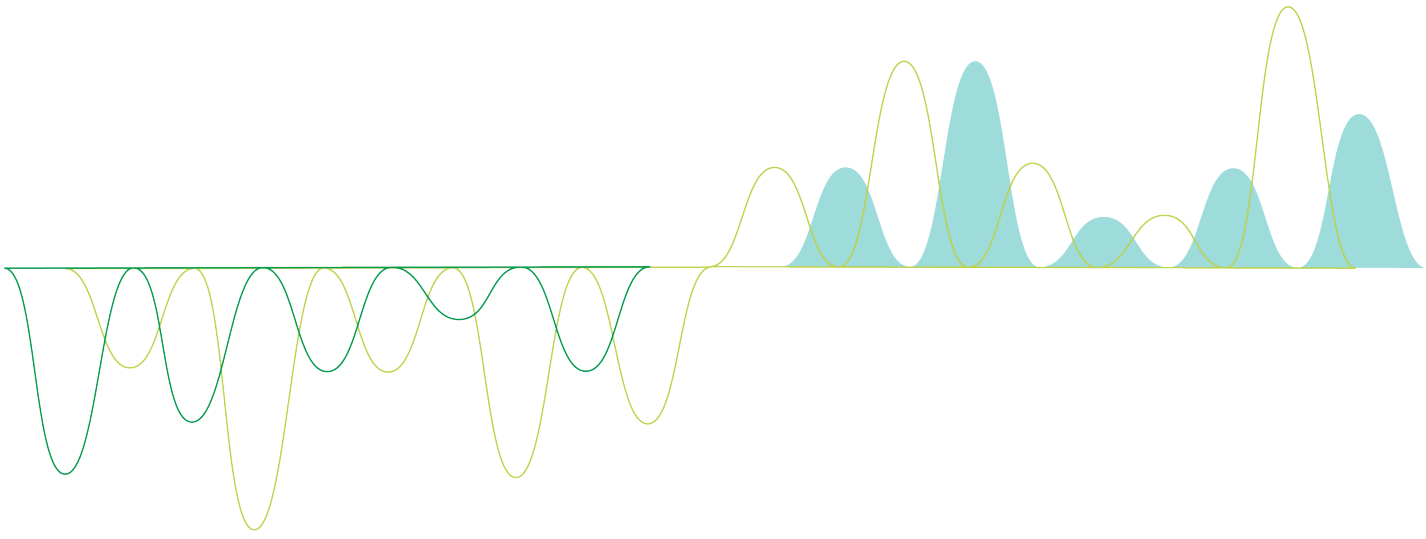
Składnia skryptów i funkcje wykresów

Qlik Sense®
November 2022
Copyright © 1993-2023 QlikTech International AB. Wszelkie prawa zastrzeżone.



© 2023 QlikTech International AB. All rights reserved. All company and/or product names may be trade names, trademarks and/or registered trademarks of the respective owners with which they are associated.

Contents

1 Charakterystyka programu Qlik Sense

Program Qlik Sense to platforma służąca do analizy danych. Dzięki Qlik Sense można analizować dane i odkrywać ukryte w nich prawidłowości. Ponadto można dzielić się wiedzą i analizować dane w grupach i w ramach całych organizacji. Qlik Sense pozwala użytkownikom zadawać własne pytania, odnajdywać na nie odpowiedzi i formułować wnioski we własny sposób. Za pomocą programu Qlik Sense łatwo jest podejmować decyzje wspólnie ze współpracownikami.

1.1 Korzystanie z programu Qlik Sense

Większość produktów Business Intelligence (BI) ułatwia uzyskanie odpowiedzi na pytania, które można wcześniej przewidzieć. Co jednak z odpowiedziami na pytania uzupełniające, czyli takie, które nasuwają się dopiero po zapoznaniu się z raportem lub wizualizacją? Dzięki udostępnianej przez program Qlik Sense funkcji asocjacji można odpowiadać na kolejne pytania uzupełniające na drodze do sedna poczynionej obserwacji. Za pomocą programu Qlik Sense można swobodnie badać dostępne dane, uzyskiwać na ich podstawie wiedzę na każdym etapie takiego badania oraz odkrywać nowe kierunki analizy na podstawie wcześniejszych obserwacji.

1.2 Zasady działania programu Qlik Sense

Program Qlik Sense generuje w czasie rzeczywistym odpowiednie widoki danych. Program Qlik Sense nie wymaga wstępnie zdefiniowanych i statycznych raportów ani nie uzależnia użytkownika od innych użytkowników. Wystarczy klikać i zdobywać wiedzę. Po każdym kliknięciu program Qlik Sense natychmiastowo aktualizuje wszystkie wizualizacje i widoki Qlik Sense w aplikacji nowo obliczonymi zbiorami danych i wizualizacjami właściwymi dla danej selekcji.

Model aplikacji

Nie trzeba już wdrażać dużych aplikacji biznesowych i zarządzać nimi. Wystarczy bowiem utworzenie własnych aplikacji Qlik Sense, które można następnie używać, modyfikować i udostępniać innym osobom. Model aplikacji ułatwia zadawanie pytań i udzielanie odpowiedzi na kolejne pytania w sposób samodzielny bez konieczności zwracania się z prośbą do eksperta o nowe raporty lub wizualizacje.

Funkcja asocjacji

Program Qlik Sense automatycznie zarządza wszystkimi relacjami między danymi i prezentuje informacje za pomocą kolorów **green/white/gray**. Selekcje są podświetlane na zielono, dane powiązane – na biało, a dane wykluczone (niepowiązane) – na szaro. Ten błyskawiczny system komunikacji pozwala na łatwiejsze formułowanie kolejnych pytań i dalsze swobodne eksplorowanie i wykrywanie danych.

Obsługa współpracy i urządzeń przenośnych

Program Qlik Sense pozwala na współpracę z innymi użytkownikami w dowolnym momencie i z dowolnego miejsca. Wszystkie funkcje programu Qlik Sense, w tym funkcje asocjacji i współpracy, są dostępne na urządzeniach przenośnych. Za pomocą programu Qlik Sense można zadawać i odpowiadać na wszystkie pojawiające się w toku analizy pytania, także w ramach współpracy z innymi użytkownikami.

1.3 Wdrażanie programu Qlik Sense

Można wdrożyć dwie wersje programu Qlik Sense – Qlik Sense Desktop i Qlik Sense Enterprise.

Qlik Sense Desktop

Jest to łatwa w instalacji wersja dla jednego użytkownika, która jest zazwyczaj wdrażana na lokalnym komputerze.

Qlik Sense Enterprise

Jest to wersja służąca do wdrażania lokacji Qlik Sense. Lokacja jest to co najmniej jeden serwer podłączony do wspólnego repozytorium logicznego lub węzła centralnego.

1.4 Administrowanie i zarządzanie lokacją Qlik Sense

Za pomocą programu Konsola zarządzania Qlik można łatwo i intuicyjnie konfigurować i monitorować lokacje Qlik Sense oraz zarządzać nimi. Można m.in. zarządzać licencjami, dostępem i regułami zabezpieczeń, konfigurować węzły i połączenia ze źródłami danych oraz synchronizować treści i profile użytkowników w ramach wielu działań i zasobów.

1.5 Rozszerzanie programu Qlik Sense i dostosowywanie go do własnych potrzeb

Program Qlik Sense zapewnia elastyczne interfejsy API i zestawy SDK służące do tworzenia własnych rozszerzeń oraz dostosowania programu Qlik Sense do własnych potrzeb takich jak:

Tworzenie rozszerzeń i aplikacji typu mashup

Za pomocą języka JavaScript można tworzyć rozszerzenia będące niestandardowymi wizualizacjami w aplikacjach Qlik Sense. Za pomocą interfejsów API aplikacji typu „mashup” można też tworzyć witryny internetowe zawierające zawartość z programu Qlik Sense.

Tworzenie klientów

Można tworzyć klienty w obiektach .NET i wbudowanych obiektach Qlik Sense w ramach własnych aplikacji. Ponadto można w dowolnym języku programowania tworzyć klienty natywne obsługujące komunikację WebSocket za pomocą protokołu klienta Qlik Sense.

Tworzenie narzędzi serwerowych

Za pomocą interfejsów API usługi i katalogów użytkownika można tworzyć własne narzędzia służące do administrowania i zarządzania lokacjami Qlik Sense.

Nawiązywanie połączenia z innymi źródłami danych

W celu pobrania danych z niestandardowych źródeł danych można utworzyć odpowiednie łączniki programu Qlik Sense.

2 Omówienie składni skryptów

2.1 Wprowadzenie do składni skryptów

Skrypt zawiera definicje nazwy źródła danych, nazw tabel i nazw pól przetwarzanych przez logikę skryptu. W skrypcie są też zdefiniowane pola z definicji praw dostępu. Skrypt składa się z pewnej liczby instrukcji wykonywanych sekwencyjnie.

Składnia wiersza poleceń aplikacji Qlik Sense i składnia skryptu są opisywane w notacji określanej jako notacja Backus-Naur lub kod BNF.

Po utworzeniu nowego pliku Qlik Sense kilka pierwszych wierszy kodu jest generowanych automatycznie. Wartości domyślne tych zmiennych interpretacji liczb są określane na podstawie ustawień regionalnych systemu operacyjnego.

Skrypt składa się z pewnej liczby instrukcji skryptu i słów kluczowych, które są wykonywane sekwencyjnie. Wszystkie instrukcje skryptowe muszą kończyć się znakiem średnika „;”.

W celu przekształcania załadowanych danych można używać wyrażeń i funkcji w instrukcjach **LOAD**.

W przypadku pliku tabeli rozdzielanego przecinkami, tabulatorami lub średnikami można korzystać z instrukcji **LOAD**. Domyślnie instrukcja **LOAD** załaduje wszystkie pola w pliku.

Dostęp do ogólnych baz danych można uzyskać za pośrednictwem łączników baz danych ODBC lub OLE DB. W tym przypadku używane są standardowe instrukcje SQL. Akceptowana składnia SQL różni się w zależności od sterowników ODBC.

Dodatkowo dostęp do innych źródeł danych można uzyskiwać, korzystając z łączników niestandardowych.

2.2 Czym jest notacja Backus-Naur?

Składnia wiersza poleceń aplikacji Qlik Sense i składnia skryptu są opisywane w notacji określanej jako notacja Backus-Naur, zwanej również kodem BNF.

W poniższej tabeli zamieszczono listę symboli używanych w kodzie BNF wraz z opisem ich interpretacji:

Symbole

Symbol	Opisu
|	Alternatywa logiczna OR: można użyć jednego lub drugiego z symboli rozdzielonych tym znakiem.
( )	Nawiasy definiujące pierwszeństwo: używane do określania struktury składni BNF.
[ ]	Nawiasy kwadratowe: elementy w takich nawiasach są opcjonalne.
{ }	Nawiasy klamrowe: elementy w takich nawiasach mogą być powtarzane dowolną liczbę razy, w tym zero.

Symbol	Opisu
Symbol	Nieterminalna kategoria składniowa: symbol może się dzielić na inne symbole. Na przykład połączenia powyższych, inne symbole nieterminalne, ciągi tekstowe itp.
::=	Wskazuje początek bloku definiującego symbol.
LOAD	Symbol terminalny w postaci ciągu znaków. Należy go bez zmian umieścić w skrypcie.

Wszystkie symbole terminalne są zapisywane czcionką **bold face**. Na przykład znak „(” należy interpretować jako nawias określający pierwszeństwo, natomiast znak „**(**” jako znak wpisywany bezpośrednio w skrypcie.

Przykład:

Opis instrukcji alias wygląda następująco:

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

Zapis należy interpretować tak: ciąg znaków „alias”, potem dowolna nazwa pola, potem ciąg znaków „as”, a potem dowolny alias. Można podać dowolną liczbę dodatkowych kombinacji wartości „fieldname as alias”, rozdzielając je przecinkami.

Następujące instrukcje są poprawne:

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

Następujące instrukcje nie są poprawne:

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

2 Instrukcje i słowa kluczowe skryptu

Skrypt Qlik Sense składa się z pewnej liczby instrukcji. Instrukcje dzielą się na zwykłe instrukcje skryptu oraz instrukcje sterowania skryptem. Niektóre instrukcje można poprzedzać prefiksami.

Zwykłe instrukcje służą zazwyczaj do wykonywania operacji na danych. Każda taka instrukcja może obejmować w skrypcie dowolną liczbę wierszy i musi zawsze być zakończona średnikiem, czyli znakiem „;”.

Instrukcje sterowania służą zazwyczaj do sterowania przepływem wykonania skryptu. Każda klauzula instrukcji sterowania musi mieścić się w jednym wierszu skryptu i może być zakończona albo średnikiem, albo znakiem końca linii.

Prefiksy można stosować z obsługującymi je instrukcjami zwykłymi, ale nigdy z instrukcjami sterowania. Wyjątek stanowią prefiksy **when** i **unless**, których można używać jako sufiksów klauzul kilku ściśle określonych instrukcji sterowania.

W następnym podrozdziale zostanie podany wykaz alfabetyczny wszystkich instrukcji skryptowych, instrukcji sterowania i prefiksów.

Słowa kluczowe w skrypcie mogą być wpisywane z użyciem dowolnych kombinacji małych i wielkich liter. Wielkość liter jest natomiast uwzględniana w nazwach pól i zmiennych używanych w instrukcjach.

2.3 Instrukcje sterowania skryptem

Skrypt Qlik Sense składa się z pewnej liczby instrukcji. Instrukcje dzielą się na zwykłe instrukcje skryptu oraz instrukcje sterowania skryptem.

Instrukcje sterowania służą zazwyczaj do sterowania przepływem wykonania skryptu. Każda klauzula instrukcji sterowania musi mieścić się w jednym wierszu skryptu i może być zakończona albo średnikiem, albo znakiem końca linii.

Do instrukcji sterowania nie mają zastosowania prefiksy, z wyjątkiem prefiksów **when** i **unless**, których można używać z kilkoma ściśle określonymi instrukcjami sterowania.

Słowa kluczowe w skrypcie mogą być wpisywane z użyciem dowolnych kombinacji małych i wielkich liter.

Przegląd instrukcji sterowania skryptem

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

Call

Instrukcja sterowania **call** wywołuje procedurę zdefiniowaną we wcześniejszej instrukcji **sub**.

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

Do..loop

Instrukcja sterowania **do..loop** to rodzaj iteracji skryptu, który wykonuje co najmniej jedną instrukcję aż do momentu spełnienia warunku logicznego.

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

Exit script

Instrukcja ta zatrzymuje wykonanie skryptu. Można ją wstawić w dowolnym miejscu skryptu.

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

For each ..next

Instrukcja sterowania **for each..next** to rodzaj iteracji skryptu służący do wykonania co najmniej jednej instrukcji dla każdej wartości na liście rozdzielonej przecinkami. Dla każdej wartości na liście wykonane zostaną instrukcje wewnątrz pętli między argumentami **for** i **next**.

```
For each..next var in list
[statements]
[exit for [ ( when | unless ) condition ]
[statements]
next [var]
```

For..next

Instrukcja sterowania **for..next** to rodzaj iteracji skryptu z licznikiem. Dla każdej wartości zmiennej licznika mieszczącej się w określonym limicie wykonane zostaną instrukcje wewnątrz pętli między wartościami **for** i **next**.

```
For..next counter = expr1 to expr2 [ stepexpr3 ]
[statements]
[exit for [ ( when | unless ) condition ]
[statements]
Next [counter]
```

If..then

Instrukcja sterowania **if..then** jest rodzajem selekcji skryptu, który wymusza wykonanie skryptu według różnych ścieżek w zależności od co najmniej jednego warunku logicznego.

> *Instrukcja **if..then** jest instrukcją sterowania i jako taka kończy się średnikiem lub symbolem końca linii, a żadna z jej czterech możliwych klauzul (**if..then**,**elseif..then**, **else** i **end if**) nie może wykraczać poza granicę linii.*

```
If..then..elseif..else..end if condition then
  [ statements ]
{ elseif condition then
  [ statements ] }
[ else
  [ statements ] ]
```

```
end if
```

Sub

Instrukcja sterowania **sub..end sub** definiuje podprogram, który można wywołać z instrukcji **call**.

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

Switch

Instrukcja sterowania **switch** jest rodzajem selekcji skryptu, który wymusza wykonanie skryptu według różnych ścieżek w zależności od wartości wyrażenia.

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

Call

Instrukcja sterowania **call** wywołuje procedurę zdefiniowaną we wcześniejszej instrukcji **sub**.

Składnia:

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

Argumenty:

Argumenty

Argument	Opis
name	Nazwa podprogramu.
paramlist	Do podprogramu przekazana zostanie rozdzielana przecinkami lista rzeczywistych parametrów. Każdy z elementów listy może być nazwą pola, zmienną lub dowolnym wyrażeniem.

Podprogram wywoływany instrukcją **call** musi być zdefiniowany w instrukcji **sub** napotkanej na wcześniejszym etapie wykonywania skryptu.

Parametry są kopiowane do podprogramu, a jeśli parametrem instrukcji **call** jest zmienna, a nie wyrażenie, to po wyjściu z podprogramu parametry są ponownie kopiowane na zewnątrz.

Ograniczenia:

- Jako instrukcja sterowania instrukcja **call** kończy się na średniku lub znaku nowego wiersza, nie może zatem obejmować wielu wierszy.
- Podczas definiowania podprogramu za pomocą `Sub..end sub` wewnątrz instrukcji sterującej, na przykład `if..then`, można wywołać podprogram tylko z tej samej instrukcji sterującej.

Przykład:

W tym przykładzie powstaje lista wszystkich plików związanych z aplikacją Qlik w folderze oraz jego podfolderach, a następnie informacje o plikach są zapisywane w tabeli. Przyjęto, że użytkownik utworzył powiązania między danymi o nazwie Apps do tego folderu.

Podprogram DoDir jest wywoływany z odniesieniem do folderu 'lib://Apps' jako parametrem. W podprogramie istnieje wywołanie rekurencyjne `Call DoDir (Dir)`, które powoduje, że funkcja wyszukuje pliki rekurencyjnie w podfolderach.

```
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

Instrukcja sterowania **do..loop** to rodzaj iteracji skryptu, który wykonuje co najmniej jedną instrukcję aż do momentu spełnienia warunku logicznego.

Składnia:

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

*Instrukcja **do..loop** jest instrukcją sterowania i jako taka kończy się średnikiem lub symbolem końca linii, żadna z jej trzech możliwych klauzul (**do**,**exit do** i **loop**) nie może zatem wykraczać poza granicę linii.*

Argumenty:

Argumenty

Argument	Opis
condition	Wyrażenie logiczne, którego ocena zwraca True lub False.
statements	Dowolna grupa zawierająca co najmniej jedną instrukcję skryptu Qlik Sense.
while / until	Klauzula warunkowa **while** lub **until** może pojawiać się tylko raz w instrukcji **do..loop**, tj. po instrukcji **do** lub po instrukcji **loop**. Każdy warunek jest interpretowany tylko przy pierwszym napotkaniu, ale jest oceniany każdorazowo przy napotkaniu w pętli.
exit do	Jeśli wewnątrz pętli napotkana zostanie klauzula **exit do**, wykonanie skryptu zostanie przeniesione do pierwszej instrukcji po klauzuli **loop** wskazującej na koniec pętli. Klauzula **exit do** może zostać ustawiona jako warunkowa przez opcjonalne użycie sufiksu **when** lub **unless**.

Przykład:

```
// LOAD files file1.csv..file9.csv
Set a=1;
Do while a<10
LOAD * from file$(a).csv;
Let a=a+1;
Loop
```

End

Słowo kluczowe skryptu **End** służy do zamykania klauzul **If**, **Sub** i **Switch**.

Exit

Słowo kluczowe skryptu **Exit** jest częścią instrukcji **Exit Script**, ale może także zostać użyte do zamykania klauzul **Do**, **For** lub **Sub**.

Exit script

Instrukcja ta zatrzymuje wykonanie skryptu. Można ją wstawić w dowolnym miejscu skryptu.

Składnia:

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

Jako instrukcja sterowania instrukcja **exit script** kończy się na średniku lub znaku nowego wiersza, nie może zatem obejmować wielu wierszy.

Argumenty:

Argumenty

Argument	Opis
condition	Wyrażenie logiczne, którego ocena zwraca True lub False.
when / unless	Instrukcja **exit script** może zostać ustawiona jako warunkowa przez opcjonalne użycie klauzuli **when** lub **unless**.

Przykłady:

```
//Exit script
Exit Script;

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

For..next

Instrukcja sterowania **for..next** to rodzaj iteracji skryptu z licznikiem. Dla każdej wartości zmiennej licznika mieszczącej się w określonym limicie wykonane zostaną instrukcje wewnątrz pętli między wartościami **for** i **next**.

Składnia:

```
For counter = expr1 to expr2 [ step expr3 ]
[statements]
[exit for [ ( when | unless ) condition ]
[statements]
Next [counter]
```

Wyrażenia *expr1*, *expr2* i *expr3* są oceniane tylko przy pierwszym wejściu do pętli. Wartość zmiennej counter można zmienić przy użyciu instrukcji wewnątrz pętli, ale nie jest to zalecana praktyka programowania.

Jeśli wewnątrz pętli napotkana zostanie klauzula **exit for**, wykonanie skryptu zostanie przeniesione do pierwszej instrukcji po klauzuli **next** wskazującej na koniec pętli. Klauzula **exit for** może zostać ustawiona jako warunkowa przez opcjonalne użycie sufiksu **when** lub **unless**.

> *Instrukcja **for..next** jest instrukcją sterowania i jako taka kończy się średnikiem lub symbolem końca linii, żadna z jej trzech możliwych klauzul (**for..to..step**,**exit for** i **next**) nie może zatem wykraczać poza granicę linii.*

Argumenty:

Argumenty

Argument	Opis
counter	Nazwa zmiennej. Jeśli argument *counter* określono po argumencie **next**, musi on mieć taką samą nazwę zmiennej jak nazwa po **for**.
expr1	Wyrażenie określające pierwszą wartość zmiennej *counter*, dla której pętla powinna zostać wykonana.
expr2	Wyrażenie określające ostatnią wartość zmiennej *counter*, dla której pętla powinna zostać wykonana.
expr3	Wyrażenie określające wartość wskazującą na przyrost zmiennej *counter* podczas każdego wykonania pętli.
condition	wyrażenie logiczne, którego ocena zwraca True lub False.
statements	Dowolna grupa zawierająca co najmniej jedną instrukcję skryptu Qlik Sense.

Example 1: Ładowanie sekwencji plików

```
// LOAD files file1.csv..file9.csv
for a=1 to 9

      LOAD * from file$(a).csv;

next
```

Example 2: Ładowanie losowej liczby plików

W tym przykładzie zakładamy istnienie plików danych *x1.csv, x3.csv, x5.csv, x7.csv* i *x9.csv*. Ładowanie jest zatrzymywane w losowym punkcie przy użyciu warunku `if rand( )<0.5 then`.

```
for counter=1 to 9 step 2

      set filename=x$(counter).csv;
      if rand( )<0.5 then

            exit for unless counter=1

      end if
      LOAD a,b from $(filename);

next
```

For each..next

Instrukcja sterowania **for each..next** to rodzaj iteracji skryptu służący do wykonania co najmniej jednej instrukcji dla każdej wartości na liście rozdzielonej przecinkami. Dla każdej wartości na liście wykonane zostaną instrukcje wewnątrz pętli między argumentami **for** i **next**.

Składnia:

Dzięki specjalnej składni możliwe jest generowanie list z nazwami plików i katalogów w katalogu bieżącym.

```
for each var in list
[statements]
[exit for [ ( when | unless ) condition ]
[statements]
next [var]
```

Argumenty:

Argumenty

Argument	Opis
var	Nazwa zmiennej skryptu pobierającej nową wartość z listy dla każdego wykonania pętli. Jeśli argument **var** określono po argumencie **next**, musi on mieć taką samą nazwę zmiennej jak nazwa po **for each**.

Wartość zmiennej **var** można zmienić przy użyciu instrukcji wewnątrz pętli, ale nie jest to zalecana praktyka programowania.

Jeśli wewnątrz pętli napotkana zostanie klauzula **exit for**, wykonanie skryptu zostanie przeniesione do pierwszej instrukcji po klauzuli **next** wskazującej na koniec pętli. Klauzula **exit for** może zostać ustawiona jako warunkowa przez opcjonalne użycie sufiksu **when** lub **unless**.

> *Instrukcja **for each..next** jest instrukcją sterowania i jako taka kończy się średnikiem lub symbolem końca linii, żadna z jej trzech możliwych klauzul (**for each**,**exit for** i **next**) nie może zatem wykraczać poza granicę linii.*

Składnia:

```
list := item { , item }
item := constant | (expression) | filelist mask | dirlist mask |
fieldvaluelist mask
```

Argumenty

Argument	Opis
constant	Dowolna liczba lub ciąg znaków. Należy pamiętać, że ciąg znaków wpisany bezpośrednio w skrypt musi być ujęty w pojedyncze cudzysłowy. Ciąg znaków bez pojedynczych cudzysłowów zostanie zinterpretowany jako zmienna i będzie użyta wartość zmiennej. Liczb nie trzeba ujmować w pojedyncze cudzysłowy.
expression	Dowolne wyrażenie.
mask	Maska nazwy pliku lub folderu, która może zawierać dowolne znaki dozwolone w nazwie pliku, a także standardowe symbole wieloznaczne, na przykład ***** i **?**. Można używać bezwzględnych ścieżek do plików lub ścieżek lib://.
condition	Wyrażenie logiczne, którego ocena zwraca True lub False.
statements	Dowolna grupa zawierająca co najmniej jedną instrukcję skryptu Qlik Sense.
filelist mask	Składnia tworzy rozdzieloną przecinkami listę wszystkich plików w bieżącym katalogu zgodnym z maską nazwy pliku. *Ten argument obsługuje tylko połączenia bibliotek w trybie standardowym.*
dirlist mask	Składnia tworzy rozdzieloną przecinkami listę wszystkich folderów w bieżącym folderze zgodnym z maską nazwy folderu. *Ten argument obsługuje tylko połączenia bibliotek w trybie standardowym.*
fieldvaluelist mask	Ta składnia jest iterowana przez wartości pola już załadowane do aplikacji Qlik Sense.

Połączenia z Qlik Łączniki do dostawcy magazynowania w sieci Web i inne połączenia DataFiles nie obsługują masek filtrów, które używają symboli wieloznacznych (i ?).*

Example 1: Ładowanie listy plików

```
// 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: Tworzenie listy plików na dysku

W tym przykładzie ładowana jest lista wszystkich plików powiązanych z aplikacją Qlik Sense w folderze.

```
 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: Iteracja przez wartości pola

W tym przykładzie iteracja przeprowadzana jest przez listę załadowanych wartości FIELD i generowane jest nowe pole, NEWFIELD. Dla każdej wartości FIELD utworzone zostaną dwa rekordy NEWFIELD.

```
 load * inline [ FIELD one two three ];  FOR Each a in FieldValueList('FIELD') LOAD '$(a)' &'-
'&RecNo() as NEWFIELD AutoGenerate 2; NEXT a
```

Wynikowa tabela wygląda następująco:

Example table

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

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

Instrukcja sterowania **if..then** jest rodzajem selekcji skryptu, który wymusza wykonanie skryptu według różnych ścieżek w zależności od co najmniej jednego warunku logicznego.

Instrukcje sterowania służą zazwyczaj do sterowania przepływem wykonania skryptu. W wyrażeniu wykresu użyj funkcji warunkowej **if**.

Składnia:

```
If  condition then
```

```
  [ statements ]
{ elseif condition then
  [ statements ] }
[ else
  [ statements ] ]
end if
```

Instrukcja **if..then** jest instrukcją sterowania i jako taka kończy się średnikiem lub symbolem końca linii, a żadna z jej czterech możliwych klauzul (**if..then**,**elseif..then**, **else** i **end if**) nie może wykraczać poza granicę linii.

Argumenty:

Argumenty

Argument	Opis
condition	Wyrażenie logiczne dające w wyniku wartości True lub False.
statements	Dowolna grupa zawierająca co najmniej jedną instrukcję skryptu Qlik Sense.

Example 1:

```
if a=1 then

      LOAD * from abc.csv;
      SQL SELECT e, f, g from tab1;

end if
```

Example 2:

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

Example 3:

```
if x>0 then

      LOAD * from pos.csv;

elseif x<0 then

      LOAD * from neg.csv;

else

      LOAD * from zero.txt;

end if
```

Next

Słowo kluczowe skryptu **Next** służy do zamykania pętli **For**.

Sub..end sub

Instrukcja sterowania **sub..end sub** definiuje podprogram, który można wywołać z instrukcji **call**.

Składnia:

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

Argumenty są kopiowane do podprogramu, a jeśli odpowiadające parametry rzeczywiste w instrukcji **call** są nazwami zmiennych, wówczas są kopiowane ponownie na zewnątrz po wyjściu z podprogramu.

Jeśli podprogram zawiera więcej parametrów formalnych niż rzeczywistych przekazywanych przez instrukcję **call**, wówczas parametry dodatkowe zostaną zainicjowane na NULL i możliwe będzie ich użycie jako zmiennych lokalnych w podprogramie.

Argumenty:

Argumenty

Argument	Opis
name	Nazwa podprogramu.
paramlist	Rozdzielana przecinkami lista nazw zmiennych dla parametrów formalnych podprogramu. Te zmienne mogą być używane jak dowolna zmienna w podprogramie.
statements	Dowolna grupa zawierająca co najmniej jedną instrukcję skryptu Qlik Sense.

Ograniczenia:

- Instrukcja **sub** jest instrukcją sterowania i jako taka kończy się średnikiem lub symbolem końca linii – żadna z jej dwóch klauzul (**sub** i **end sub**) nie może zatem wykraczać poza granicę linii.
- Podczas definiowania podprogramu za pomocą `Sub..end sub` wewnątrz instrukcji sterującej, na przykład `if..then`, można wywołać podprogram tylko z tej samej instrukcji sterującej.

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: – transfer parametru

```
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)
```

Wynik powyższej składni będzie następujący: lokalnie, wewnątrz podprogramu, A zostanie zainicjowane na 1, B zostanie zainicjowane na 4, a C zostanie zainicjowane na NULL.

Po wyjściu z podprogramu zmienna globalna A otrzyma 2 jako wartość (kopiowaną z powrotem z podprogramu). Drugi parametr rzeczywisty „(X+1)*2” nie zostanie skopiowany z powrotem, ponieważ nie jest zmienną. Ostatecznie wywołanie podprogramu nie wpływa na zmienną globalną C.

Switch..case..default..end switch

Instrukcja sterowania **switch** jest rodzajem selekcji skryptu, który wymusza wykonanie skryptu według różnych ścieżek w zależności od wartości wyrażenia.

Składnia:

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

*Instrukcja **switch** jest instrukcją sterowania i jako taka kończy się średnikiem lub symbolem końca linii, a żadna z jej czterech możliwych klauzul (**switch**,**case**, **default** i **end switch**) nie może wykraczać poza granicę linii.*

Argumenty:

Argumenty

Argument	Opis
expression	Dowolne wyrażenie.
valuelist	Rozdzielana przecinkami listą wartości, z którymi będzie porównywana wartość wyrażenia. Wykonywanie skryptu będzie kontynuowane z instrukcjami w pierwszej napotkanej grupie, w której wartość z listy valuelist jest równa wartości expression. Każda wartość z listy valuelist może być dowolnym wyrażeniem. Jeśli w żadnej klauzuli **case** nie zostanie znalezione żadne dopasowanie, wówczas zostaną wykonane instrukcje z klauzuli **default**, jeśli zostały określone.
statements	Dowolna grupa zawierająca co najmniej jedną instrukcję skryptu Qlik Sense.

Przykład:

```
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

Słowo kluczowe **To** jest używane w wielu instrukcjach skryptu.

2.4 Prefiksy skryptu

Prefiksy można stosować z obsługującymi je instrukcjami zwykłymi, ale nigdy z instrukcjami sterowania. Wyjątek stanowią prefiksy **when** i **unless**, których można używać jako sufiksów klauzul kilku ściśle określonych instrukcji sterowania.

Słowa kluczowe w skrypcie mogą być wpisywane z użyciem dowolnych kombinacji małych i wielkich liter. Wielkość liter jest natomiast uwzględniana w nazwach pól i zmiennych używanych w instrukcjach.

Przegląd prefiksów skryptu

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

Add

Do dowolnej instrukcji **LOAD** lub **SELECT** w skrypcie można dodać prefiks **Add**, aby określić, że powinna ona dodawać rekordy do innej tabeli. Określa on również, że ta instrukcja powinna być uruchamiana podczas częściowego ładowania. Prefiksu **Add** można też użyć w instrukcji **Map**.

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

Buffer

Pliki QVD można tworzyć i utrzymywać automatycznie z użyciem prefiksu **buffer**. Prefiksu tego można używać w większości instrukcji **LOAD** i **SELECT** w skryptach. Sygnalizuje on, że do buforowania wyniku instrukcji używane są pliki QVD.

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

Concatenate

Nawet jeśli dwie tabele przeznaczone do konkatenacji zawierają różne zestawy pól, konkatenacja dwóch tabel może zostać wymuszona za pomocą prefiksu **Concatenate**.

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

Crosstable

Prefiks ładowania **crosstable** służy do transpozycji danych strukturalnych typu „tabela krzyżowa” lub „tabela przestawna”. Dane o takiej strukturze są często spotykane podczas pracy ze źródłami w postaci arkuszy kalkulacyjnych. Wynikiem i celem prefiksu ładowania **crosstable** jest transpozycja takich struktur do zwykłego odpowiednika tabeli z kolumnami, ponieważ ta struktura jest ogólnie lepiej przystosowana do analizy w Qlik Sense.

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

First

Prefiks First w instrukcji LOAD lub SELECT (SQL) służy do ładowania z tabeli źródła danych ustalonej maksymalnej liczby rekordów.

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

Generic

Prefiks ładowania Generic umożliwia konwersję danych w modelu encja-atrybut-wartość (EAV) na tradycyjną, znormalizowaną, relacyjną strukturę tabeli. Modelowanie EAV jest alternatywnie określane jako „ogólne modelowanie danych” lub „schemat otwarty”.

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

Hierarchy

Prefiks hierarchy służy do przekształcenia tabeli hierarchii nadrzędne-podrzędne w tabelę nadającą się do zastosowania w modelu danych Qlik Sense. Podanie go przed instrukcją LOAD lub SELECT sprawi, że wynik instrukcji ładującej będzie używany jako dane wejściowe dla transformacji tabeli.

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

HierarchBelongsTo

Prefiks ten służy do przekształcenia tabeli hierarchii nadrzędne-podrzędne w tabelę nadającą się do użycia w modelu danych Qlik Sense. Podanie go przed instrukcją LOAD lub SELECT sprawi, że wynik instrukcji ładującej będzie używany jako dane wejściowe dla transformacji tabeli.

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

Inner

Prefiksy join i keep mogą być poprzedzone prefiksem inner.

Podanie go przed prefiksem join oznacza, że ma być używane sprzężenie wewnętrzne. Wynikowa tabela będzie wtedy zawierać tylko te kombinacje wartości pól z pierwotnych tabel danych, dla których wartości pola łączącego są obecne w obu tabelach. Podanie go przed prefiksem keep oznacza, że przed zapisaniem w aplikacji Qlik Sense obie tabele surowych danych mają zostać zredukowane do części wspólnej danych.

.

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

IntervalMatch

Prefiks IntervalMatch służy do utworzenia tabeli dopasowującej dyskretne wartości liczbowe do jednego lub wielu interwałów liczbowych, opcjonalnie dopasowując wartości jednego lub kilku kluczy dodatkowych.

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

Join

Prefiks **join** powoduje sprzężenie załadowanej tabeli z istniejącą tabelą nazwaną lub ostatnio utworzoną tabelą danych.

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

Keep

Prefiks **keep** jest podobny do prefiksu **join**. Podobnie jak prefiks **join** powoduje on porównanie załadowanej tabeli do istniejącej tabeli nazwanej lub ostatnio utworzonej tabeli danych. Zamiast jednak sprzęgać tabelę załadowaną z istniejącą, powoduje zredukowanie jednej lub obu tabel do części wspólnej danych przed zapisaniem tabel w aplikacji Qlik Sense. Wykonywane porównanie jest równoważne wykonaniu sprzężenia naturalnego po wszystkich wspólnych polach, czy przebiega tak samo, jak przy analogicznej instrukcji join. Tabele nie są jednak sprzęgane i będą przechowywane w aplikacji Qlik Sense jako dwie odrębne tabele nazwane.

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

Left

Prefiksy **Join** i **Keep** mogą być poprzedzone prefiksem **left**.

Podanie go przed prefiksem **join** oznacza, że ma być używane lewe sprzężenie. Wynikowa tabela będzie zawierać tylko te kombinacje wartości pól z pierwotnych tabel danych, dla których wartości pola łączącego są obecne w pierwszej tabeli. Podanie go przed prefiksem **keep** oznacza, że przed zapisaniem w aplikacji Qlik Sense druga tabela surowych danych ma zostać zredukowana do części wspólnej z pierwszą tabelą.

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

Mapping

Prefiks **mapping** służy do utworzenia tabeli mapowania, której można używać na przykład do zastępowania wartości i nazw pól podczas wykonywania skryptu.

```
Mapowanie  ( loadstatement | selectstatement )
```

Merge

Do dowolnej instrukcji **LOAD** lub **SELECT** w skrypcie można dodać prefiks **Merge**, aby określić, że ładowana tabela powinna zostać scalona w innej tabeli. Określa on również, że ta instrukcja powinna być uruchamiana podczas częściowego ładowania.

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

NoConcatenate

Prefiks **NoConcatenate** wymusza traktowanie dwóch załadowanych tabel z identycznym zestawem pól jako dwóch oddzielnych tabel wewnętrznych, podczas gdy w przeciwnym wypadku automatycznie zostałaby wobec nich zastosowana konkatenacja.

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

Outer

Jawny prefiks **Join** może być poprzedzony prefiksem **Outer**, aby określić sprzężenie zewnętrzne. W przypadku sprzężenia zewnętrznego generowane są wszystkie kombinacje między dwiema tabelami. Otrzymana tabela zawiera zatem kombinacje wartości pól z tabel samych danych, gdzie łączące wartości pola są reprezentowane w jednej lub obu tabelach. Słowo **Outer** jest opcjonalne i stanowi domyślny typ sprzężenia używany w sytuacji, gdy nie określono prefiksu sprzężenia.

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

Partial reload

Pełne ładowanie zawsze rozpoczyna się od usunięcia wszystkich tabel w istniejącym modelu danych, a następnie uruchamiany jest skrypt ładowania.

Nie umożliwia tego *Częściowe ładowanie (page 98)*. Zamiast tego zachowuje ono w modelu danych wszystkie tabele, a następnie wykonuje tylko instrukcje **Load** i **Select** poprzedzone prefiksem **Add**, **Merge** lub **Replace**. Polecenie to nie wpływa na inne tabele danych. Argument **only** oznacza, że instrukcja powinna być wykonywana tylko podczas częściowego ładowania i pomijana podczas pełnego ładowania. W poniższej tabeli podsumowano wykonanie instrukcji dla częściowego i pełnego ładowania.

Replace

Prefiks **Replace** można dodać do dowolnej instrukcji **LOAD** lub **SELECT** w skrypcie, aby określić, że ładowana tabela powinna zastąpić inną tabelę. Określa on również, że ta instrukcja powinna być uruchamiana podczas częściowego ładowania. Prefiksu **Replace** można też użyć w instrukcji **Map**.

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

Right

Prefiksy **Join** i **Keep** mogą być poprzedzone prefiksem **right**.

Podanie go przed prefiksem **join** oznacza, że ma być używane prawe sprzężenie. Wynikowa tabela będzie zawierać tylko te kombinacje wartości pól z pierwotnych tabel danych, dla których wartości pola łączącego są obecne w drugiej tabeli. Podanie go przed prefiksem **keep** oznacza, że przed zapisaniem w aplikacji Qlik Sense pierwsza tabela surowych danych ma zostać zredukowana do części wspólnej z drugą tabelą.

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

Sample

Prefiks **sample** w instrukcji **LOAD** lub **SELECT** służy do ładowania ze źródła danych losowej próbki rekordów.

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

Semantic

Prefiks **semantic** umożliwia ładowanie tabel zawierających relacje między rekordami. Mogą to być na przykład odwołania własne w obrębie tabeli, w których jeden rekord wskazuje na inny: nadrzędny, do którego należy, lub poprzedni.

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

Unless

Prefiks i sufiks **unless** jest używany do tworzenia klauzuli warunkowej określającej, czy dana instrukcja lub klauzula wyjściowa ma być sprawdzana czy też nie. Mogą być one traktowane jako wygodna alternatywa pełnej instrukcji **if..end if**.

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

When

Prefiks i sufiks **when** jest używany do tworzenia klauzuli warunkowej określającej, czy dana instrukcja lub klauzula wyjściowa ma być wykonywana czy też nie. Mogą być one traktowane jako wygodna alternatywa pełnej instrukcji **if..end if**.

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

Add

Do dowolnej instrukcji **LOAD** lub **SELECT** w skrypcie można dodać prefiks **Add**, aby określić, że powinna ona dodawać rekordy do innej tabeli. Określa on również, że ta instrukcja powinna być uruchamiana podczas częściowego ładowania. Prefiksu **Add** można też użyć w instrukcji **Map**.

> *Aby częściowe ładowanie działało poprawnie, aplikacja musi zostać otwarta z danymi przed jego uruchomieniem.*

Wykonaj częściowe ładowanie za pomocą przycisku **Ładuj**. Możesz także użyć Qlik Engine JSON API.

Składnia:

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

```
Add [only] mapstatement
```

Podczas normalnego (nie częściowego) ładowania konstrukcja **Add LOAD** będzie działać jako normalna instrukcja **LOAD**. Rekordy zostaną wygenerowane i zapisane w tabeli.

Jeśli będzie używany prefiks **Concatenate** lub jeśli będzie istnieć tabela z tym samym zestawem pól, rekordy zostaną dołączone do odpowiedniej istniejącej tabeli. W przeciwnym razie konstrukcja **Add LOAD** utworzy nową tabelę.

Częściowe ładowanie da takie same rezultaty. Jedyna różnica polega na tym, że konstrukcja **Add LOAD** nigdy nie utworzy nowej tabeli. Zawsze istnieje odpowiednia tabela z poprzedniego wykonania skryptu, do której należy dołączyć rekordy.

Nie jest wykonywane sprawdzanie duplikatów. W związku z tym instrukcja z prefiksem **Add** zazwyczaj zawiera kwalifikator distinct lub klauzulę where eliminującą duplikaty.

W przypadku instrukcji**Add Map...Using** mapowanie jest przeprowadzane również podczas częściowego wykonywania skryptu.

Argumenty:

Argumenty

Argument	Opis
only	Opcjonalny kwalifikator wskazujący, że instrukcja ma być wykonywana tylko podczas częściowego ładowania. Należy go pominąć podczas normalnych (nie częściowych) ładowań.

Przykłady i wyniki:

Przykład	Wynik

`Tab1:`
 	
`LOAD Name, Number FROM Persons.csv;`
 	
`Add LOAD Name, Number FROM newPersons.csv;`
	Podczas normalnego przeładowania dane są ładowane z pliku *Persons.csv* i zapisywane w tabeli Qlik Sense o nazwie Tab1. Dane z pliku *NewPersons.csv* są następnie konkatenowane z tą samą tabelą Qlik Sense.
	
	Podczas częściowego przeładowania dane są ładowane z pliku *NewPersons.csv* i dołączane do tabeli Qlik Sense o nazwie Tab1. Nie jest wykonywane sprawdzanie duplikatów.

`Tab1:`
 	
`SQL SELECT Name, Number FROM Persons.csv;`
 	
`Add LOAD Name, Number FROM NewPersons.csv where not exists (Name);`
	Wykonywane jest sprawdzenie duplikatów polegające na wyszukaniu, czy wartość pola Name już istnieje we wcześniej załadowanych danych tabeli.
	
	Podczas normalnego przeładowania dane są ładowane z pliku *Persons.csv* i zapisywane w tabeli Qlik Sense o nazwie Tab1. Dane z pliku *NewPersons.csv* są następnie konkatenowane z tą samą tabelą Qlik Sense.
	
	Podczas częściowego przeładowania dane są ładowane z pliku *NewPersons.csv*, który jest dołączany do tabeli Qlik Sense o nazwie Tab1. Wykonywane jest sprawdzenie duplikatów polegające na wyszukaniu, czy wartość pola Name już istnieje we wcześniej załadowanych danych tabeli.

`Tab1:`
 	
`LOAD Name, Number FROM Persons.csv;`
 	
`Add Only LOAD Name, Number FROM NewPersons.csv where not exists (Name);`
	Podczas normalnego przeładowania dane są ładowane z pliku *Persons.csv* i zapisywane w tabeli Qlik Sense o nazwie Tab1. Instrukcja ładująca plik *NewPersons.csv* jest ignorowana.
	
	Podczas częściowego przeładowania dane są ładowane z pliku *NewPersons.csv*, który jest dołączany do tabeli Qlik Sense o nazwie Tab1. Wykonywane jest sprawdzenie duplikatów polegające na wyszukaniu, czy wartość pola Name już istnieje we wcześniej załadowanych danych tabeli.

Buffer

Pliki QVD można tworzyć i utrzymywać automatycznie z użyciem prefiksu **buffer**. Prefiksu tego można używać w większości instrukcji **LOAD** i **SELECT** w skryptach. Sygnalizuje on, że do buforowania wyniku instrukcji używane są pliki QVD.

Składnia:

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

Jeśli nie zostanie podana żadna opcja, bufor QVD utworzony przez pierwsze wykonanie skryptu będzie używany przez czas nieokreślony.

Plik bufora jest zapisywany w podfolderze *Buffers* zwykle w ścieżce *C:\ProgramData\Qlik\Sense\Engine\Buffers* (instalacja serwera) lub w ścieżce *C:\Users\{user}\Documents\Qlik\Sense\Buffers* (Qlik Sense Desktop).

Nazwa pliku QVD jest wyliczanym 160-bitowym skrótem szesnastkowym całej następującej po niej instrukcji **LOAD** lub **SELECT** i innych informacji wyróżniających. Oznacza to, że każda zmiana w następującej później instrukcji **LOAD** lub **SELECT** spowoduje, że bufor QVD stanie się niepoprawny.

Bufory QVD są zazwyczaj usuwane, gdy nie ma już do nich żadnych odniesień po wykonaniu całego skryptu w aplikacji tworzącej bufor lub gdy aplikacja tworząca bufor przestaje istnieć.

Argumenty:

Argumenty

Argument	Opis
incremental	Opcja incremental umożliwia odczytywanie tylko części pliku bazowego. Poprzedni rozmiar pliku jest zapisany w nagłówku XML w pliku QVD. Jest to szczególnie przydatne w przypadku plików dziennika. Wszystkie rekordy załadowane wcześniej są wczytywane z pliku QVD, natomiast dalsze nowe rekordy są wczytywane z oryginalnego pliku źródłowego. Na koniec tworzony jest zaktualizowany plik QVD. Opcji incremental można używać tylko z instrukcjami **LOAD** i plikami tekstowymi. Ładowania przyrostowego nie można używać w przypadku zmiany lub usunięcia starych danych.
stale [after] amount [(days | hours)]	amount to liczba określająca okres. Mogą być używane liczby dziesiętne. Jeśli parametr nie zostanie określony, przyjmowana jest wartość dni. Opcja stale after jest zazwyczaj używana w przypadku źródeł bazodanowych, gdy nie istnieje prosty znacznik czasu oryginalnych danych. Zamiast tego określa się maksymalny dopuszczalny wiek migawki QVD. Klauzula stale after jedynie określa czas od momentu utworzenia bufora QVD, po którego upływie bufor będzie uważany za nieważny. Przed upływem tego czasu źródłem danych będzie bufor QVD, natomiast po upływie tego czasu używane będzie oryginalne źródło danych. Plik bufora QVD zostanie automatycznie zaktualizowany i rozpocznie się nowy okres.

Ograniczenia:

Istnieje szereg ograniczeń, przede wszystkim wymaganie, aby sednem wszelkich instrukcji złożonych była instrukcja **LOAD** lub **SELECT** odnosząca się do pliku.

Example 1:

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

Example 2:

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

Example 3:

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

Concatenate

`Concatenate` to prefiks ładowania skryptu, który umożliwia dołączenie zestawu danych do tabeli znajdującej się w pamięci. Często używa się go w celu dołączania różnych zestawów danych transakcyjnych do jednej centralnej tabeli faktów lub do budowy wspólnych zestawów danych odniesienia określonego typu, które pochodzą z wielu źródeł. Jego funkcjonalność jest podobna do sposobu działania operatora SQL UNION.

Tabela wyników operacji `concatenate` będzie zawierała oryginalny zestaw danych z nowymi wierszami danych dołączonymi na końcu. Tabele źródłowa i docelowa mogą zawierać różne pola. Tam, gdzie pola się różnią, tabela wynikowa zostanie rozszerzona w celu zaprezentowania połączonego wyniku wszystkich pól obecnych zarówno w tabeli źródłowej, jak i docelowej.

Składnia:

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

Argumenty

Argument	Opis
`tablename`	Nazwa istniejącej tabeli. Wymieniona tabela będzie tabelą docelową operacji `Concatenate` i wszystkie rekordy załadowanych danych zostaną dodane do tej tabeli. Jeśli parametr `tablename` nie zostanie użyty, tabela docelowa będzie ostatnią załadowaną tabelą przed tym poleceniem.
`loadstatement/selectstatement`	Argument `loadstatement/selectstatement` znajdujący się za argumentem `tablename` zostanie dołączony do określonej tabeli.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład funkcji

Przykład	Wynik
`Concatenate (Transactions) Load …. ;`	Dane załadowane w instrukcji LOAD pod przedrostkiem `Concatenate` zostaną dodane do istniejącej w pamięci tabeli o nazwie `Transactions` (przy założeniu, że tabela o nazwie `Transactions` została wcześniej załadowana w skrypcie ładowania).

Przykład 1 – Dołączanie kilku zestawów danych do tabeli docelowej za pomocą przedrostka ładowania Concatenate

Skrypt ładowania i wyniki

Przegląd

W tym przykładzie załadujemy dwa skrypty po kolei.

- Pierwszy skrypt ładowania zawiera początkowy zestaw danych zawierający daty i kwoty, który został wysłany do tabeli o nazwie `Transactions`.
- Drugi skrypt ładowania zawiera następujące elementy:
    - Drugi zestaw danych, który jest dołączony do pierwszego za pomocą przedrostka `Concatenate`. Ten zestaw danych ma dodatkowe pole, `type`, którego nie ma w pierwszym zestawie danych.
    - Prefiks `Concatenate`.

Otwórz edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Pierwszy skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

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

Tabela wyników pierwszego skryptu ładowania

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

Tabela przedstawia pierwszy zestaw danych.

Drugi skrypt ładowania

Otwórz edytor ładowania danych i poniżej dodaj skrypt ładowania.

```
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
];
```

Wyniki

Załaduj dane i przejdź do arkusza. Utwórz to pole jako wymiar:

- `type`

Tabela wyników drugiego skryptu ładowania

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

Zwróć uwagę na puste wartości w polu `type` dla siedmiu pierwszych załadowanych rekordów, w których nie zdefiniowano argumentu `type`.

Przykład 2 – Dołączanie kilku zestawów danych do tabeli docelowej za pomocą konkatenacji niejawnej

Skrypt ładowania i wyniki

Przegląd

Typowy przypadek użycia niejawnego dodawania danych to sytuacja, w której ładujemy kilka plików danych o identycznej strukturze i chcemy dołączyć je wszystkie do tabeli docelowej.

Na przykład przy użyciu `wildcards` w nazwach plików przy użyciu następującej składni:

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

lub w pętlach wykorzystujących następujące konstrukcje:

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

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

next file
```

Konkatenacja niejawna zostanie wykonana między każdymi dwiema załadowanymi tabelami, które mają pola o identycznych nazwach, nawet jeśli nie są one zdefiniowane jedna po drugiej w skrypcie. To może doprowadzić do nieumyślnego dodania danych do tabel. Jeśli nie chcesz, aby dodatkowa tabela z identycznymi polami została w ten sposób dołączona, użyj prefiksu ładowania `NoConcatenate`*. Zmiana nazwy tabeli przez dodanie alternatywnego znacznika nie wystarczy, aby zapobiec niejawnej konkatenacji. Więcej informacji zawiera temat NoConcatenate (page 88).*

W tym przykładzie załadujemy dwa skrypty po kolei.

- Pierwszy skrypt ładowania zawiera początkowy zestaw danych zawierający cztery pola, który został wysłany do tabeli o nazwie `Transactions`.
- Drugi skrypt ładowania zawiera zestaw danych zawierający te same pola, co pierwszy zestaw danych.

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Pierwszy skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

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

Tabela wyników pierwszego skryptu ładowania

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

Tabela przedstawia pierwszy zestaw danych.

Drugi skrypt ładowania

Otwórz edytor ładowania danych i poniżej dodaj skrypt ładowania.

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

Wyniki

Załaduj dane i przejdź do arkusza.

Tabela wyników drugiego skryptu ładowania

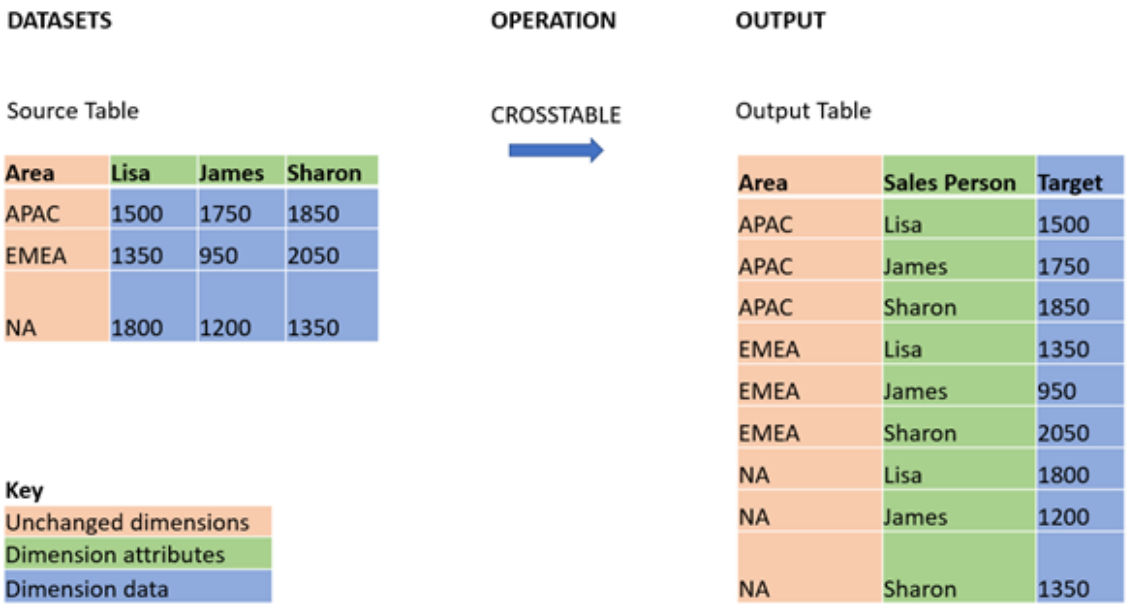
id	date	type	amount
3758	10/01/2018	Wewnętrzne	164.27
3759	10/03/2018	Zewnętrzne	384.00
3760	10/06/2018	Wewnętrzne	25.82
3761	10/09/2018	Wewnętrzne	312.00
3762	10/15/2018	Wewnętrzne	4.56
3763	10/16/2018	Wewnętrzne	90.24
3764	10/18/2018	Zewnętrzne	19.32
3765	11/03/2018	Wewnętrzne	129.40
3766	11/05/2018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

Drugi zestaw danych został niejawnie dołączony do pierwszego zestawu, ponieważ oba te zestawy zawierają identyczne pola.

Crosstable

Prefiks ładowania **crosstable** służy do transpozycji danych strukturalnych typu „tabela krzyżowa” lub „tabela przestawna”. Dane o takiej strukturze są często spotykane podczas pracy ze źródłami w postaci arkuszy kalkulacyjnych. Wynikiem i celem prefiksu ładowania **crosstable** jest transpozycja takich struktur do zwykłego odpowiednika tabeli z kolumnami, ponieważ ta struktura jest ogólnie lepiej przystosowana do analizy w Qlik Sense.

Przykład danych ustrukturyzowanych jako tabela krzyżowa i odpowiednik jej struktury po transformacji tabeli krzyżowej



Składnia:

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

Argumenty

Argument	Opis
attribute field name	Nazwa pola żądanych danych wyjściowych opisująca wymiar zorientowany poziomo, który ma być transponowany (wiersz nagłówka).
data field name	Nazwa pola żądanych danych wyjściowych opisująca wymiar zorientowany poziomo, który ma być transponowany (macierz wartości danych pod wierszem nagłówka).
n	Liczba poprzedzających tabelę pól kwalifikatora, czyli niezmienionych wymiarów, które zostaną przekształcone do postaci ogólnej. Wartością domyślną jest 1.

Ta funkcja skryptu jest powiązana z następującymi funkcjami:

Powiązane funkcje

Funkcja	Interakcja
Generic (page 56)	Prefiks ładowania transformacji, który powoduje pobranie uporządkowanego zestawu danych jednostka-atrybut-wartość i przekształcenie go w zwykłą relacyjną strukturę tabeli, oddzielając każdy napotkany atrybut w nowym polu lub kolumnie danych.

Przykład 1 – Przekształcenie przestawnych danych sprzedaży (proste)

Skrypty ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj pierwszy skrypt ładowania poniżej do nowej karty.

Pierwszy skrypt ładowania zawiera zestaw danych, do którego prefiks skryptu `crosstable` zostanie zastosowany później, a sekcja dotycząca zastosowania `crosstable` została wykomentowana. Oznacza to, że do wyłączenia tej sekcji w skrypcie ładowania użyto składni komentarza.

Drugi skrypt ładowania jest taki sam jak pierwszy, ale z zastosowaniem `crosstable` bez wykomentowania (włączone przez usunięcie składni komentarza). Skrypty są pokazane w ten sposób, aby podkreślić wartość tej funkcji skryptowej w przekształcaniu danych.

Pierwszy skrypt ładowania (funkcja nie jest stosowana)

```
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;
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `Product`
- `Jan 2021`
- `Feb 2021`

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

Tabela wynikowa

Produkt	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

Skrypt umożliwia tworzenie tabeli krzyżowej z jedną kolumną na każdy miesiąc i jednym wierszem na produkt. Analiza tych danych w obecnym formacie nie jest łatwa. O wiele lepiej byłoby mieć wszystkie liczby w jednym polu, a wszystkie miesiące w innym, w trzykolumnowej tabeli. W następnej sekcji wyjaśniono, jak wykonać tę transformację tabeli krzyżowej.

Drugi skrypt ładowania (funkcja nie jest stosowana)

Usuń komentarz ze skryptu, usuwając `//`. Skrypt ładowania powinien wyglądać następująco:

```
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;
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- Product
- Month
- Sales

Tabela wynikowa

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

Po zastosowaniu prefiksu skryptu tabela krzyżowa jest przekształcana w tabelę prostą z jedną kolumną `Month`, a drugą — `Sales`. Zwiększa to czytelność danych.

Przykład 2 – przekształcenie przestawianych danych celów sprzedaży w pionową strukturę tabeli (średnio zaawansowany)

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie Targets.
- Prefiks ładowania `crosstable`, który transponuje imiona i nazwiska sprzedawców w przestawieniu

na własne pole oznaczone etykietą `Sales Person`.

- Powiązane dane celu sprzedaży, które są uporządkowane w polu o nazwie `Target`.

Skrypt ładowania

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

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `Area`
- `Sales Person`

Dodaj tę miarę:

```
=Sum(Target)
```

Tabela wynikowa

Obszar	Sprzedawca	=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

Jeśli chcesz powielić wyświetlanie danych jako przestawną tabelę wejściową, możesz utworzyć równoważną tabelę przestawną w arkuszu.

Wykonaj następujące czynności:

1. Skopiuj i wklej właśnie utworzoną tabelę do arkusza.
2. Przeciągnij obiekt wykresu **tabeli przestawnej** na nowo utworzoną kopię tabeli. Wybierz **Konwertuj**.
3. Kliknij przycisk ✔ **Koniec edycji**.
4. Przeciągnij pole `Sales Person` z pionowej półki kolumnowej na poziomą półkę kolumnową.

Poniższa tabela przedstawia dane w ich początkowej formie tabeli, tak jak są one wyświetlane w Qlik Sense:

Oryginalna tabela wyników, tak jak jest widoczna w Qlik Sense

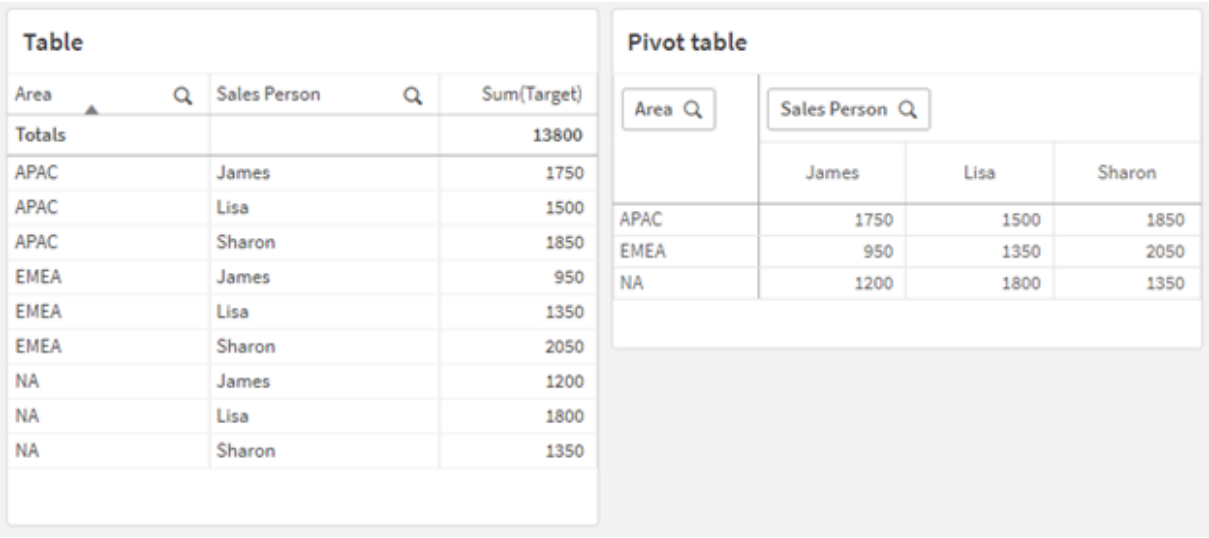
Obszar	Sprzedawca	=Sum(Target)
Sumy	-	13800
APAC	James	1750
APAC	Lisa	1500
APAC	Sharon	1850
EMEA	James	950
EMEA	Lisa	1350
EMEA	Sharon	2050
NA	James	1200
NA	Lisa	1800
NA	Sharon	1350

Odpowiednia tabela przestawna wygląda podobnie do poniższej, przy czym kolumna z imieniem i nazwiskiem każdego sprzedawcy znajduje się w większym wierszu dla `Sales Person`:

Równoważna tabela przestawna z polem `Sales Person` przestawionym w poziomie

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

Przykład danych wyświetlanych jako tabela i odpowiadająca jej tabela przestawna z polem `Sales Person` przestawionym poziomo



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

Przykład 3 – przekształcenie przestawianych danych celów sprzedaży w pionową strukturę tabeli (zaawansowany)

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych reprezentujący dane dotyczące sprzedaży i celów, uporządkowany według obszaru i miesiąca roku. Ten zestaw danych jest ładowany do tabeli o nazwie `SalesAndTargets`.
- Prefiks ładowania `crosstable`. Służy do przestawiania wymiaru `Month Year` do dedykowanego pola, a także do transpozycji macierzy sprzedaży i kwot docelowych do dedykowanego pola o nazwie `Amount`.
- Konwersja pola `Month Year` z tekstu na właściwą datę przy użyciu funkcji konwersji tekstu na datę `date#`. To pole `Month Year` przekonwertowane na datę jest dołączane z powrotem do tabeli `SalesAndTarget` za pomocą prefiksu ładowania `Join`.

Skrypt ładowania

```
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;
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `Area`
- `Month Year`

Utwórz następującą miarę z etykietą `Actual`:

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

Utwórz także następującą miarę z etykietą `Target`:

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

Tabela wyników (przycięte)

Area	Month Year	Actual	Target
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

Area	Month Year	Actual	Target
APAC	Dec-22	397	425
EMEA	Jan-22	363.5	362.5
EMEA	Feb-22	359.5	362.5

Jeśli chcesz powielić wyświetlanie danych jako przestawną tabelę wejściową, możesz utworzyć równoważną tabelę przestawną w arkuszu.

Wykonaj następujące czynności:

1. Skopiuj i wklej właśnie utworzoną tabelę do arkusza.
2. Przeciągnij obiekt wykresu **tabeli przestawnej** na nowo utworzoną kopię tabeli. Wybierz **Konwertuj**.
3. Kliknij przycisk ✔ **Koniec edycji**.
4. Przeciągnij pole `Month Year` z pionowej półki kolumnowej na poziomą półkę kolumnową.
5. Przeciągnij element `Values` z poziomej półki kolumnowej na pionową półkę kolumnową.

Poniższa tabela przedstawia dane w ich początkowej formie tabeli, tak jak są one wyświetlane w Qlik Sense:

Oryginalna tabela wyników (przycięta), tak jak jest widoczna w Qlik Sense

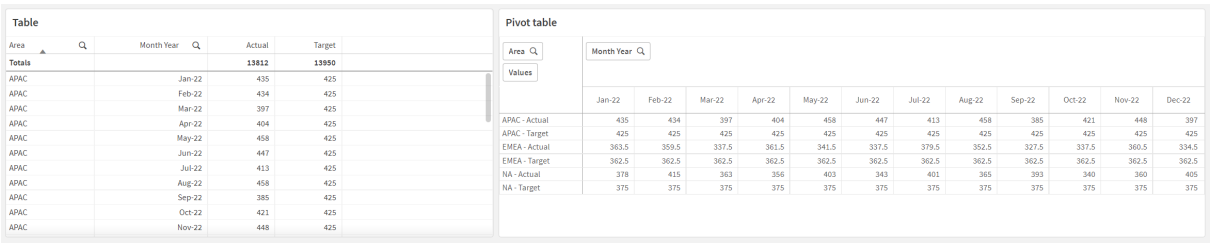
Area	Month Year	Actual	Target
Sumy	-	13812	13950
APAC	Jan-22	435	425
APAC	Feb-22	434	425
APAC	Mar-22	397	425
APAC	Apr-22	404	425
APAC	May-22	458	425
APAC	Jun-22	447	425
APAC	Jul-22	413	425
APAC	Aug-22	458	425
APAC	Sep-22	385	425
APAC	Oct-22	421	425
APAC	Nov-22	448	425
APAC	Dec-22	397	425
EMEA	Jan-22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

Odpowiednia tabela przestawna wygląda podobnie do poniższej, przy czym kolumna z poszczególnymi miesiącami znajduje się w większym wierszu dla `Month Year`:

Równoważna tabela przestawna (przycięta) z polem `Month Year` przestawionym w poziomie

Area (wartości)	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 - Actual	435	434	397	404	458	447	413	458	385	421	448	397
APAC - Target	425	425	425	425	425	425	425	425	425	425	425	425
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
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
NA - Actual	378	415	363	356	403	343	401	365	393	340	360	405
NA - Target	375	375	375	375	375	375	375	375	375	375	375	375

Przykład danych wyświetlanych jako tabela i odpowiadająca jej tabela przestawna z polem `Month Year` przestawionym poziomo



First

Prefiks `First` w instrukcji `LOAD` lub `SELECT` (SQL) służy do ładowania z tabeli źródła danych ustalonej maksymalnej liczby rekordów. Typowym zastosowaniem prefiksu `First` jest pobranie małego podzbioru rekordów na etapie dużego lub powolnego ładowania danych. Po załadowaniu zdefiniowanej liczby rekordów „`n`” etap ładowania kończy się przedwcześnie, a reszta wykonywania skryptu jest kontynuowana normalnie.

Składnia:

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

Argumenty

Argument	Opis
n	Dowolne wyrażenie, którego wynik oceny jest liczbą całkowitą wskazującą maksymalną liczbę rekordów do odczytania. n można też zamknąć w nawiasie: (n).
loadstatement | selectstatement	load statement/select statement po argumencie n zdefiniuje określoną tabelę, którą należy załadować z ustawioną maksymalną liczbą rekordów.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji SET DateFormat w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
FIRST 10 LOAD * from abc.csv;	Ten przykład pobierze pierwsze dziesięć wierszy z pliku programu Excel.
FIRST (1) SQL SELECT * from Orders;	Ten przykład pobierze pierwszy wybrany wiersz z zestawu danych Orders.

Przykład – Ładowanie pierwszych pięciu wierszy

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych dat z pierwszych dwóch tygodni 2020 roku.
- Zmienna First, która nakazuje aplikacji ładowanie tylko pierwszych pięciu rekordów.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj `Date` jako pole oraz `sum(sales)` jako miarę.

Tabela wynikowa

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

Skrypt ładuje tylko pierwszych pięć rekordów tabeli `Sales`.

Generic

Prefiks ładowania **Generic** umożliwia konwersję danych w modelu encja-atrybut-wartość (EAV) na tradycyjną, znormalizowaną, relacyjną strukturę tabeli. Modelowanie EAV jest alternatywnie określane jako „ogólne modelowanie danych” lub „schemat otwarty”.

Przykład danych modelowanych EAV i równoważnej zdenormalizowanej tabeli relacyjnej

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

Przykład danych modelowanych EAV i równoważnego zestawu znormalizowanych tabel relacyjnych

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

Chociaż jest technicznie możliwe ładowanie i analizowanie danych modelowanych EAV w Qlik, często łatwiej jest pracować z równoważną tradycyjną relacyjną strukturą danych.

Składnia:

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

W pracy z tą funkcją mogą Ci pomóc poniższe tematy:

Tematy pokrewne

Temat	Opis
Crosstable (page 45)	Prefiks ładowania `Crosstable` przekształca dane zorientowane poziomo na dane zorientowane pionowo. Z czysto funkcjonalnego punktu widzenia wykonuje transformację przeciwną niż prefiks ładowania `Generic`, chociaż prefiksy zazwyczaj służą do zupełnie innych zastosowań.
Ogólne bazy danych w *Zarządzanie danymi*	Ustrukturyzowane modele danych EAV są dokładniej opisane tutaj.

Przykład 1 – przekształcenie uporządkowanych danych EAV z prefiksem ładowania Generic

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera zestaw danych, który jest ładowany do tabeli o nazwie `Transactions`. Zestaw danych zawiera pole daty. Używana jest domyślna definicja `MonthNames`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `Color`.

Dodaj tę miarę:

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

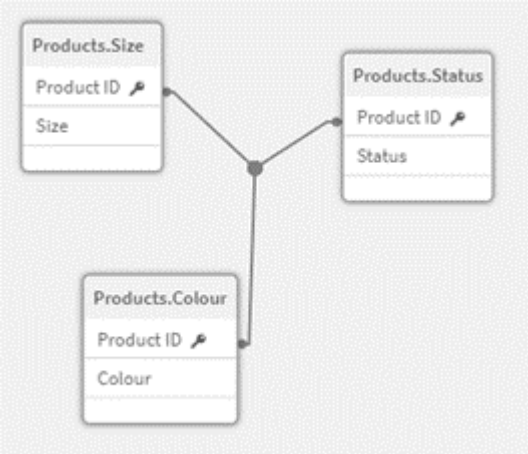
Teraz możesz sprawdzić liczbę produktów według koloru.

Tabela wynikowa

Color	=Count([Product ID])
Brown	4
White	2

Zwróć uwagę na kształt modelu danych, w którym każdy atrybut został podzielony na osobną tabelę o nazwie zgodnej z oryginalnym znacznikiem tabeli docelowej `Product`. Każda tabela ma atrybut jako sufiks. Jednym z przykładów jest `Product.Color`. Wynikowe rekordy wyjściowe atrybutu produktu są kojarzone na podstawie `Product ID`.

Reprezentacja wyników w przeglądarce modelu danych



Wynikowa tabela rekordów: Products.Status

Product ID	Status
13	Discontinued
2	Discontinued

Wynikowa tabela rekordów: Products.Size

Product ID	Rozmiar
13	13-15
20	16-18
45	16-18

Wynikowa tabela rekordów: Products.Color

Product ID	Color
13	Brown
5	Brown
44	Brown

Product ID	Color
45	Brown
20	White
2	White

Przykład 2 – przekształcenie uporządkowanych danych EAV bez prefiksu ładowania Generic

Skrypt ładowania i wyrażenie wykresu

Przegląd

Ten przykład pokazuje, jak analizować uporządkowane dane EAV w ich oryginalnej formie.

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera zestaw danych, który jest ładowany do tabeli o nazwie `Products` w strukturze EAV.

W tym przykładzie nadal liczymy produkty według atrybutu koloru. Aby przeanalizować dane o takiej strukturze, musisz zastosować filtrowanie na poziomie wyrażenia produktów o wartości atrybutu `Color`.

Ponadto pojedyncze atrybuty nie są dostępne do wybrania jako wymiary lub pola, co utrudnia określenie sposobu tworzenia skutecznych wizualizacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `Value`.

Utwórz następującą miarę:

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

Teraz możesz sprawdzić liczbę produktów według koloru.

Wynikowa tabela rekordów: Products.Status

Wartość	=Count({<Attribute={'Color'}>} [Product ID])
Brown	4
White	2

Przykład 3 – denormalizacja wynikowych tabel wyjściowych z ładowania Generic (zaawansowany)

Skrypt ładowania i wyrażenie wykresu

Przegląd

W tym przykładzie pokazujemy, jak znormalizowaną strukturę danych utworzoną przez prefiks ładowania `Generic` można zdenormalizować z powrotem do skonsolidowanej tabeli wymiaru `Product`. Jest to zaawansowana technika modelowania, którą można zastosować w ramach dostrajania wydajności modelu danych.

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania

```
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;
```

Wyniki

Otwórz Przeglądarkę modelu danych i zauważ kształt wynikowego modelu danych. Obecna jest tylko jedna zdenormalizowana tabela. Jest to połączenie trzech pośrednich tabel wyników: `Products.Size`, `Products.Status`i `Products.Color`.

Wynikowy wewnętrzny model danych

Products
Product ID
Status
Color
Size

Wynikowa tabela rekordów: Products

Product ID	Status	Color	Size
13	Discontinued	Brown	13-15
20	-	White	16-18
2	Discontinued	White	-
5	-	Brown	-
44	-	Brown	-
45	-	Brown	16-18

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `Color`.

Dodaj tę miarę:

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

Tabela wynikowa

Color	=Count([Product ID])
Brown	4
White	2

Hierarchy

Prefiks **hierarchy** służy do przekształcenia tabeli hierarchii nadrzędne-podrzędne w tabelę nadającą się do zastosowania w modelu danych Qlik Sense. Podanie go przed instrukcją **LOAD** lub **SELECT** sprawi, że wynik instrukcji ładującej będzie używany jako dane wejściowe dla transformacji tabeli.

Prefiks tworzy tabelę węzłów rozszerzonych, która zwykle zawiera tyle samo wierszy co tabela wejściowa, ale dodatkowo w osobnym polu zapisane są poziomy hierarchii. Pole ścieżki może być użyte w strukturze drzewa.

Składnia:

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

Tabela wejściowa musi być tabelą węzłów sąsiadujących. Tabela węzłów sąsiadujących to tabela, w której każdy rekord odpowiada jednemu węzłowi i zawiera pole z odwołaniem do węzła macierzystego. W takiej tabeli każdy węzeł jest zapisany tylko w jednym rekordzie, ale mimo to może mieć dowolną liczbę węzłów potomnych. Tabela może oczywiście zawierać dodatkowe pola opisujące atrybuty węzłów.

Prefiks tworzy tabelę węzłów rozszerzonych, która zwykle zawiera tyle samo wierszy co tabela wejściowa, ale dodatkowo w osobnym polu zapisane są poziomy hierarchii. Pole ścieżki może być użyte w strukturze drzewa.

Tabela wejściowa zawiera zwykle dokładnie jeden rekord na każdy węzeł i w takim przypadku tabela wyjściowa będzie zawierać tyle samo rekordów. Niekiedy istnieją też jednak węzły o wielu elementach macierzystych, przez co jednemu węzłowi odpowiada kilka rekordów w tabeli wejściowej. W takim przypadku tabela wyjściowa może zawierać więcej rekordów od tabeli wejściowej.

Wszystkie węzły bez identyfikatora węzła nadrzędnego w kolumnie identyfikatora węzła (w tym węzły z brakującym identyfikatorem węzła nadrzędnego) zostaną uznane za węzły najwyższego poziomu. Ponadto zostaną załadowane jedynie węzły posiadające połączenie (bezpośrednie lub pośrednie) z węzłem najwyższego poziomu, co pozwala unikać odwołań cyklicznych.

Można tworzyć dodatkowe pola zawierające nazwę węzła nadrzędnego, ścieżkę węzła i głębokość węzła.

Argumenty:

Argumenty

Argument	Opis
NodeID	Nazwa pola zawierającego identyfikator węzła. To pole musi istnieć w tabeli wejściowej.
ParentID	Nazwa pola zawierającego identyfikator węzła nadrzędnego. To pole musi istnieć w tabeli wejściowej.

Argument	Opis
NodeName	Nazwa pola zawierającego nazwę węzła. To pole musi istnieć w tabeli wejściowej.
ParentName	Ciąg znaków używany jako nazwa nowego pola **ParentName**. W przypadku pominięcia to pole nie zostanie utworzone.
ParentSource	Nazwa pola zawierającego nazwę węzła użytą podczas budowania ścieżki do węzła. Parametr opcjonalny. W razie jego pominięcia zostanie użyte pole **NodeName**.
PathName	Ciąg znaków używany do nazwania nowego pola **Path**, które zawiera ścieżkę od elementu najwyższego poziomu do węzła. Parametr opcjonalny. W przypadku pominięcia to pole nie zostanie utworzone.
PathDelimiter	Ciąg znaków używany jako separator w nowym polu **Path**. Parametr opcjonalny. W razie jego pominięcia zostanie użyty znak „/”.
Depth	Ciąg znaków używany do nazwania nowego pola **Depth**, które zawiera głębokość węzła w hierarchii. Parametr opcjonalny. W przypadku pominięcia to pole nie zostanie utworzone.

Przykład:

```
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

Prefiks ten służy do przekształcenia tabeli hierarchii nadrzędne-podrzędne w tabelę nadającą się do użycia w modelu danych Qlik Sense. Podanie go przed instrukcją **LOAD** lub **SELECT** sprawi, że wynik instrukcji ładującej będzie używany jako dane wejściowe dla transformacji

tabeli.

Prefiks powoduje utworzenie tabeli zawierającej wszystkie relacje nadrzędny-podrzędny w całej hierarchii. Wybierając pola nadrzędne, można następnie wybierać całe drzewa hierarchii. Tabela wyjściowa zwykle zawiera kilka rekordów dla każdego węzła.

Składnia:

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

Tabela wejściowa musi być tabelą węzłów sąsiadujących. Tabela węzłów sąsiadujących to tabela, w której każdy rekord odpowiada jednemu węzłowi i zawiera pole z odwołaniem do węzła macierzystego. W takiej tabeli każdy węzeł jest zapisany tylko w jednym rekordzie, ale mimo to może mieć dowolną liczbę węzłów potomnych. Tabela może oczywiście zawierać dodatkowe pola opisujące atrybuty węzłów.

Prefiks powoduje utworzenie tabeli zawierającej wszystkie relacje nadrzędny-podrzędny w całej hierarchii. Wybierając pola nadrzędne, można następnie wybierać całe drzewa hierarchii. Tabela wyjściowa zwykle zawiera kilka rekordów dla każdego węzła.

Można utworzyć dodatkowe pole opisujące różnice między głębokością węzłów.

Argumenty:

Argumenty

Argument	Opis
NodeID	Nazwa pola zawierającego identyfikator węzła. To pole musi istnieć w tabeli wejściowej.
ParentID	Nazwa pola zawierającego identyfikator węzła nadrzędnego. To pole musi istnieć w tabeli wejściowej.
NodeName	Nazwa pola zawierającego nazwę węzła. To pole musi istnieć w tabeli wejściowej.
AncestorID	Ciąg znaków używany do nazwania nowego pola identyfikatora węzła nadrzędnego, które zawiera identyfikator węzła nadrzędnego.
AncestorName	Ciąg znaków używany do nazwania nowego pola węzła nadrzędnego, które zawiera nazwę węzła nadrzędnego.
DepthDiff	Ciąg znaków używany do nazwania nowego pola **DepthDiff**, które zawiera głębokość węzła w hierarchii względem węzła nadrzędnego. Parametr opcjonalny. W przypadku pominięcia to pole nie zostanie utworzone.

Przykład:

```
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

Prefiksy **join** i **keep** mogą być poprzedzone prefiksem **inner**. Podanie go przed prefiksem **join** oznacza, że ma być używane sprzężenie wewnętrzne. Wynikowa tabela będzie wtedy zawierać tylko te kombinacje wartości pól z pierwotnych tabel danych, dla których wartości pola łączącego są obecne w obu tabelach. Podanie go przed prefiksem **keep** oznacza, że przed zapisaniem w aplikacji Qlik Sense obie tabele surowych danych mają zostać zredukowane do części wspólnej danych.

Składnia:

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

Argumenty:

Argumenty

Argument	Opis
tablename	Tabela nazwana, która ma być porównana do załadowanej tabeli.
loadstatementlub selectstatement	Instrukcja **LOAD** lub **SELECT** dla załadowanej tabeli.

Przykład

Skrypt ładowania

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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 ];
```

Wynik

Tabela wynikowa

Column1	Column2	Column3
A	B	C
1	aa	xx

Objaśnienie

Ten przykład ilustruje dane wyjściowe sprzężenia wewnętrznego, w którym łączone są tylko wartości obecne w pierwszej (lewej) i drugiej (prawej) tabeli.

IntervalMatch

Prefiks **IntervalMatch** służy do utworzenia tabeli dopasowującej dyskretne wartości liczbowe do jednego lub wielu interwałów liczbowych, opcjonalnie dopasowując wartości jednego lub kilku kluczy dodatkowych.

Składnia:

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

Prefiks **IntervalMatch** musi być podany przed instrukcją **LOAD** lub **SELECT** ładującą interwały. Pole zawierające dyskretne punkty danych (w poniższym przykładzie jest to pole Time) i dodatkowe klucze musi zostać załadowane do aplikacji Qlik Sense przed instrukcją z prefiksem **IntervalMatch**. Sam prefiks nie odczytuje tego pola z tabeli w bazie danych. Działanie prefiksu polega na przekształceniu załadowanej tabeli interwałów i kluczy w tabelę zawierającą dodatkową kolumnę dyskretnych punktów danych liczbowych. Dodatkowo zwiększana jest liczba rekordów, nowa tabela ma zatem jeden rekord na każdą kombinację dyskretnego punktu danych, interwału i wartości pola klucza (lub wielu pól klucza).

Interwały mogą na siebie zachodzić, a wartości dyskretne zostaną powiązane ze wszystkimi pasującymi interwałami.

Gdy prefiks IntervalMatch jest rozszerzony o pola klucza, wówczas służy do utworzenia tabeli dopasowującej dyskretne wartości liczbowe do jednego lub wielu interwałów liczbowych, jednocześnie dopasowując przy tym wartości jednego lub kilku kluczy dodatkowych.

Aby uniknąć odrzucania niezdefiniowanych limitów interwału, niekiedy może być konieczne zezwolenie na mapowanie wartości NULL na inne pola określające dolne lub górne limity interwału. Można to osiągnąć za pomocą instrukcji **NullAsValue** lub w wyniku zastosowania jawnego sprawdzenia, w ramach którego wartości NULL będą zastępowane wartością liczbową znacznie większą (lub mniejszą) od jakiegokolwiek z dyskretnych punktów danych liczbowych.

Argumenty:

Argumenty

Argument	Opis
matchfield	Pole zawierające dyskretne wartości liczbowe, które zostaną powiązane z interwałami.
keyfield	Pola z dodatkowymi atrybutami, które będą dopasowywane w ramach przekształcenia.
loadstatement orselectstatement	Wynikiem musi być tabela, w której pierwsze pole zawiera dolny limit każdego interwału, drugie pole górny limit każdego interwału, a przypadku używania dopasowania kluczy trzecie i każde kolejne pole zawiera słowa kluczowe obecne w instrukcji **IntervalMatch**. Interwały są zawsze zamknięte, tj. punkty końcowe są zawarte w interwale. Podanie limitów nieliczbowych powoduje odrzucenie interwału (będzie on niezdefiniowany).

Example 1:

Pierwsza z dwóch poniższych tabel zawiera listę zdarzeń dyskretnych, a druga definiuje czasy rozpoczęcia i zakończenia produkcji różnych zamówień. Za pomocą prefiksu **IntervalMatch** można zdefiniować połączenie logiczne dwóch tabel na przykład w celu ustalenia, która zmiana realizowała poszczególne zamówienia lub których zamówień dotyczyły zakłócenia.

```
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;
```

Tabela **OrderLog** zawiera teraz dodatkową kolumnę: *Time*. Liczba rekordów także zostaje rozwinięta.

Table with additional column

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

Example 2: (przy użyciu wartości keyfield)

Ten sam przykład co powyżej, w wyniku którego dodaje się *ProductionLine* jako pole klucza.

```
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;
```

Można teraz utworzyć następującą tabelę:

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

Prefiks **join** powoduje sprzężenie załadowanej tabeli z istniejącą tabelą nazwaną lub ostatnio utworzoną tabelą danych.

Efektem sprzężenia danych jest rozszerzenie tabeli docelowej o dodatkowy zestaw pól lub atrybutów, czyli tych, których nie ma jeszcze w tabeli docelowej. Wszelkie wspólne nazwy pól między źródłowym zestawem danych a tabelą docelową są używane do określenia sposobu powiązania nowych rekordów przychodzących. Jest to powszechnie określane jako „sprzężenie naturalne”. Operacja sprzężenia Qlik może prowadzić do tego, że wynikowa tabela docelowa będzie miała więcej lub mniej rekordów niż na początku, w zależności od unikatowości asocjacji łączenia i typu zastosowanego sprzężenia.

Istnieją cztery rodzaje sprzężeń:

Left join

Sprzężenia Left join są najczęściej występującym typem sprzężenia. Na przykład, jeśli masz zestaw danych transakcyjnych i chcesz połączyć go z zestawem danych referencyjnych, zwykle użyjesz `Left Join`. Najpierw należy załadować tabelę transakcji, a następnie załadować zestaw danych referencyjnych podczas sprzęgania jej za pomocą prefiksu `Left Join` z już załadowaną tabelą transakcji. `Left Join` zachowałoby wszystkie transakcje bez zmian i dodałoby dodatkowe pola danych referencyjnych, w których znaleziono dopasowanie.

Inner join

Jeśli masz dwa zestawy danych, w których interesują Cię tylko wyniki z pasującą asocjacją, rozważ użycie `Inner Join`. Spowoduje to wyeliminowanie wszystkich rekordów zarówno z załadowanych danych źródłowych, jak i z tabeli docelowej, jeśli nie zostanie znalezione dopasowanie. W rezultacie może to spowodować pozostawienie w tabeli docelowej mniejszej liczby rekordów niż przed wykonaniem operacji sprzęgania.

Outer join

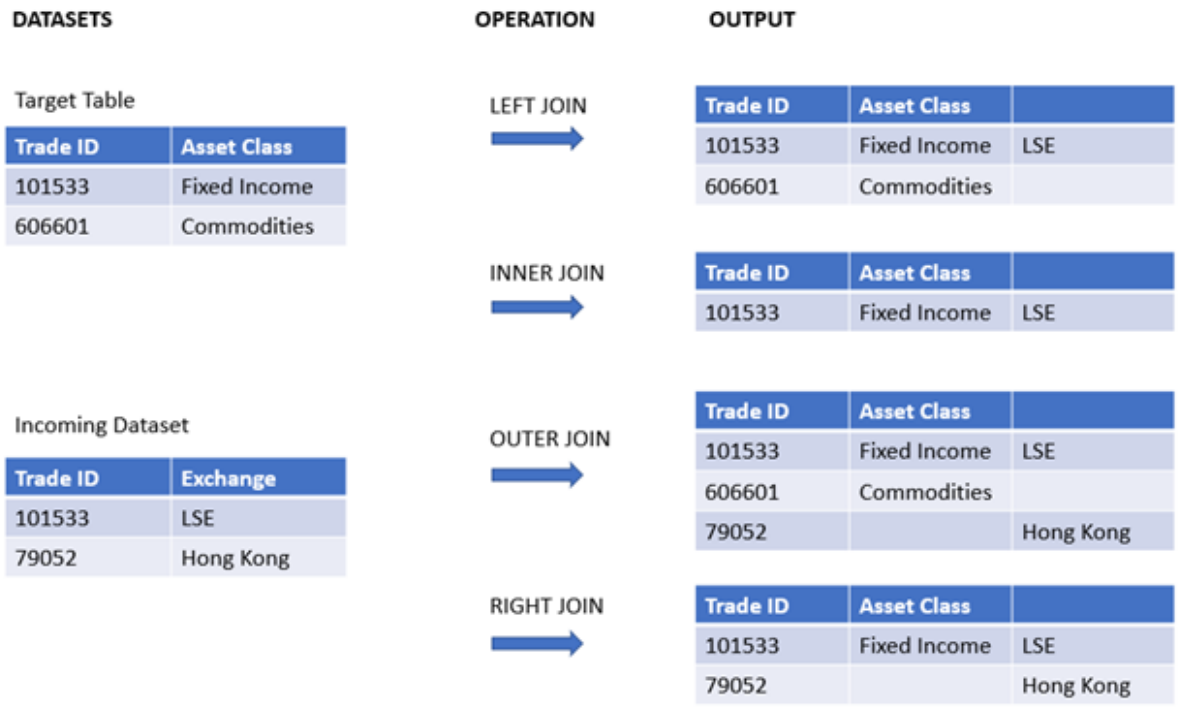
Jeśli musisz zachować zarówno rekordy docelowe, jak i wszystkie rekordy przychodzące, użyj `Outer Join`. Jeśli nie zostanie znalezione żadne dopasowanie, każdy zestaw rekordów jest nadal zachowywany, podczas gdy pola po przeciwnej stronie sprzężenia pozostaną niewypełnione (null). Sprzężenia Outer join mają na ogół niewiele praktycznych zastosowań.

Right join

Ten typ sprzężenia zachowuje wszystkie rekordy, które mają zostać załadowane, jednocześnie redukując rekordy w tabeli będącej celem sprzężenia tylko do tych rekordów, dla których istnieje dopasowanie asocjacji w rekordach przychodzących. Jest to niszowy typ sprzężenia, który jest czasami używany jako środek do przycinania już załadowanej tabeli rekordów do wymaganego podzestawu.

Jeśli słowo kluczowe type zostanie pominięte, `Inner Join` jest domyślnym typem sprzężenia.

Przykładowe zestawy wyników z różnych typów operacji sprzęgania



Jeśli nie ma wspólnych nazw pól między źródłem i celem operacji sprzężenia, da ona w wyniku iloczyn kartezjański wszystkich wierszy - nazywa się to „sprzężeniem krzyżowym”.

Przykładowy zestaw wyników z operacji „sprzężenia krzyżowego”

Składnia:

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

Argumenty

Argument	Opis
tablename	Tabela nazwana, która ma być porównana do załadowanej tabeli.
loadstatementlub selectstatement	Instrukcja **LOAD** lub **SELECT** dla załadowanej tabeli.

W pracy z tą funkcją mogą Ci pomóc poniższe tematy:

Tematy pokrewne

Temat	Opis
Łączenie tabel operatorami Join i Keep w *Zarządzanie danymi*	Ten temat zawiera dalsze wyjaśnienia pojęć „sprzęgania” i „zachowywania” zbiorów danych.
Keep (page 80)	Prefiks ładowania `Keep` jest podobny do prefiksu `Join`, ale nie łączy źródłowego i docelowego zestawu danych. Zamiast tego przycina każdy zestaw danych zgodnie z typem przyjętej operacji (inner, outer, left lub right).

Przykład 1 — Left join: Wzbogacenie tabeli docelowej referencyjnym zestawem danych

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych reprezentujący rekordy zmian, który jest ładowany do tabeli o nazwie `Changes`. Zawiera pole Status ID.
- Drugi zestaw danych reprezentujący statusy zmian, który jest ładowany i łączony z oryginalnymi rekordami zmian przez sprzężenie go przy użyciu prefiksu ładowania Left `Join`.

To sprzężenie Left join zapewnia, że rekordy zmian pozostaną nienaruszone podczas dodawania atrybutów statusu, w przypadku gdy dopasowanie w przychodzących rekordach statusu zostanie znalezione na podstawie wspólnego identyfikatora statusu.

Skrypt ładowania

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

Wyniki

Otwórz Przeglądarkę modelu danych i zauważ kształt modelu danych. Obecna jest tylko jedna zdenormalizowana tabela. Jest to połączenie wszystkich oryginalnych rekordów zmian, z pasującymi atrybutami statusu dołączonymi do każdego rekordu zmiany.

Wynikowy wewnętrzny model danych

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

Jeśli rozwiniesz okno podglądu w Przeglądarce modelu danych, zobaczysz część tego pełnego zestawu wyników zorganizowaną w formie tabeli:

Podgląd tabeli Zmiany w Przeglądarce modelu danych

Change ID	Status ID	Scheduled Start Date	Scheduled End Date	Business Impact	Status	Sub Status
10015	3	04/01/2022	15/02/2022	Low	Closed	Complete
10030	4	19/01/2022	23/02/2022	None	Closed	Cancelled
10031	3	20/01/2022	25/03/2022	Low	Closed	Complete
10040	1	29/01/2022	22/04/2022	None	Open	Not Started
10103	1	02/04/2022	29/05/2022	Średni	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
10138	2	07/05/2022	03/08/2022	None	Open	Started
10185	2	23/06/2022	08/09/2022	None	Open	Started
10187	2	25/06/2022	24/08/2022	Low	Open	Started
10220	2	28/07/2022	06/09/2022	None	Open	Started
10264	1	10/09/2022	17/10/2022	Średni	Open	Not Started
10323	1	08/11/2022	26/11/2022	Wysoka	Open	Not Started

Change ID	Status ID	Scheduled Start Date	Scheduled End Date	Business Impact	Status	Sub Status
10326	2	11/11/2022	05/12/2022	None	Open	Started
10334	2	19/11/2022	06/02/2023	Low	Open	Started

Wróć do Edytora ładowania danych. Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `Status`.

Dodaj tę miarę:

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

Teraz możesz sprawdzić liczbę zmian według statusu.

Tabela wynikowa

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

Przykład 2 – Inner join: Łączenie tylko pasujących rekordów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych reprezentujący rekordy zmian, który jest ładowany do tabeli o nazwie `Changes`.
- Drugi zestaw danych reprezentujący rekordy zmian pochodzące z systemu źródłowego `JIRA`. Jest ładowany i łączony z oryginalnymi rekordami zmian przez sprzężenie go przy użyciu prefiksu ładowania `Inner Join`.

`Inner Join` zapewnia, że zachowanych jest tylko pięć rekordów zmian, które znajdują się w obu zestawach danych.

Skrypt ładowania

```
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
10030  JIRA
10323  JIRA
10134  JIRA
10334  JIRA
10220  JIRA
10187  JIRA
] (delimiter is '\t');
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

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

Teraz możesz sprawdzić pięć wynikowych rekordów.

Tabela wynikowa

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

Przykład 3 – Outer join: Łączenie nakładających się zestawów rekordów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych reprezentujący rekordy zmian, który jest ładowany do tabeli o nazwie `Changes`.
- Drugi zestaw danych reprezentujący rekordy zmian pochodzące z systemu źródłowego `JIRA`. Jest ładowany i łączony z oryginalnymi rekordami zmian przez sprzężenie go przy użyciu prefiksu ładowania `Outer Join`.

Zapewnia to zachowanie wszystkich nakładających się rekordów zmian z obu zestawów danych.

Skrypt ładowania

```
// 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');
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

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

Teraz możesz sprawdzić 10 wynikowych rekordów.

Tabela wynikowa

Source System	Change ID	Business Impact
JIRA	10030	None
JIRA	10134	Low
JIRA	10220	None
JIRA	10323	-
JIRA	10334	Low
JIRA	10597	-
-	10015	Low
-	10031	Low
-	10040	None
-	10138	None

Przykład 4 – Right join: Przycinanie tabeli docelowej przez dodatkowy główny zbiór danych

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych reprezentujący rekordy zmian, który jest ładowany do tabeli o nazwie `Changes`.
- Drugi zestaw danych reprezentujący rekordy zmian pochodzące z systemu źródłowego `Teamwork`. Jest ładowany i łączony z oryginalnymi rekordami zmian przez sprzężenie go przy użyciu prefiksu ładowania `Right Join`.

Gwarantuje to, że przechowywane są tylko rekordy zmian `Teamwork`, bez utraty żadnych rekordów `Teamwork`, jeśli tabela docelowa nie ma pasującego `Change ID`.

Skrypt ładowania

```
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');
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

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

Teraz możesz sprawdzić pięć wynikowych rekordów.

Tabela wynikowa

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

Keep

Prefiks **keep** jest podobny do prefiksu **join**. Podobnie jak prefiks **join** powoduje on porównanie załadowanej tabeli do istniejącej tabeli nazwanej lub ostatnio utworzonej tabeli danych. Zamiast jednak sprzęgać tabelę załadowaną z istniejącą, powoduje zredukowanie jednej lub obu tabel do części wspólnej danych przed zapisaniem tabel w aplikacji Qlik Sense. Wykonywane porównanie jest równoważne wykonaniu sprzężenia naturalnego po wszystkich wspólnych polach, czy przebiega tak samo, jak przy analogicznej instrukcji join. Tabele nie są jednak sprzęgane i będą przechowywane w aplikacji Qlik Sense jako dwie odrębne tabele nazwane.

Składnia:

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

Prefiks **keep** musi być poprzedzony jednym z następujących prefiksów: **inner**, **left** lub **right**.

Jawne użycie prefiksu **join** w języku skryptowym Qlik Sense powoduje wykonanie pełnego sprzężenia dwóch tabel. Wynikiem jest pojedyncza tabela. W wielu przypadkach wykonanie sprzężenia powoduje utworzenie bardzo dużej tabeli. Jedną z kluczowych zalet aplikacji Qlik Sense jest możliwość tworzenia asocjacji między wieloma tabelami zamiast wykonywania sprzężeń. Pozwala to ograniczyć wymagania pamięciowe, przyspiesza działanie programu i zapewnia ogromną elastyczność działania. W skryptach Qlik Sense warto zatem unikać wykonywania jawnych sprzężeń. Funkcję keep zaprojektowano z myślą o ograniczeniu liczby sytuacji wymagających używania jawnych sprzężeń.

Argumenty:

Argumenty

Argument	Opis
tablename	Tabela nazwana, która ma być porównana do załadowanej tabeli.
loadstatementlub selectstatement	Instrukcja **LOAD** lub **SELECT** dla załadowanej tabeli.

Przykład:

```
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

Prefiksy **Join** i **Keep** mogą być poprzedzone prefiksem **left**.

Podanie go przed prefiksem **join** oznacza, że ma być używane lewe sprzężenie. Wynikowa tabela będzie zawierać tylko te kombinacje wartości pól z pierwotnych tabel danych, dla których wartości pola łączącego są obecne w pierwszej tabeli. Podanie go przed prefiksem **keep** oznacza, że przed zapisaniem w aplikacji Qlik Sense druga tabela surowych danych ma zostać zredukowana do części wspólnej z pierwszą tabelą.

Szukasz funkcji ciągu znaków według tej samej nazwy? Zob.: Left (page 1420)

Składnia:

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

Argumenty:

Argumenty

Argument	Opis
tablename	Tabela nazwana, która ma być porównana do załadowanej tabeli.
loadstatementlub selectstatement	Instrukcja **LOAD** lub **SELECT** dla załadowanej tabeli.

Przykład

Skrypt ładowania

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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 ];
```

Wynik

Tabela wynikowa

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

Objaśnienie

Ten przykład ilustruje dane wyjściowe lewego sprzężenia, w którym łączone są tylko wartości obecne w pierwszej (lewej) tabeli.

Mapowanie

Prefiks **mapping** służy do utworzenia tabeli mapowania, której można używać na przykład do zastępowania wartości i nazw pól podczas wykonywania skryptu.

Składnia:

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

Podanie prefiksu **mapping** przed instrukcją **LOAD** lub **SELECT** spowoduje zapisanie wyniku instrukcji ładującej w postaci tabeli mapowania. Mapowanie to skuteczny sposób na podstawianie wartości pola podczas wykonywania skryptu, na przykład zastępowania słów US, U.S. lub Ameryka słowem USA. Tabela mapowania zawiera dwie kolumny, z których pierwsza zawiera wartości porównywane, a druga docelowe wartości mapowania. Tabele mapowania są tymczasowo przechowywane w pamięci i automatycznie usuwane po wykonaniu skryptu.

Dostęp do zawartości tabeli mapowania można uzyskać na przykład z instrukcji **Map … Using**, instrukcji **Rename Field**, funkcji **Applymap()** lub funkcji **Mapsubstring()**.

Przykład:

W tym przykładzie ładowana jest lista sprzedawców z kodem kraju reprezentującym ich kraj zamieszkania. Używana jest tabela mapująca kod kraju na kraj, aby zastąpić kod kraju jego nazwą. W tabeli mapowania zdefiniowano tylko trzy kraje, inne kody krajów są mapowane na wartość 'Rest of the world'.

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

```
// We don't need the CCode anymore
Drop Field 'CCode';
```
Wynikowa tabela wygląda następująco:

Mapping table

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

Merge

Do dowolnej instrukcji **LOAD** lub **SELECT** w skrypcie można dodać prefiks **Merge**, aby określić, że ładowana tabela powinna zostać scalona w innej tabeli. Określa on również, że ta instrukcja powinna być uruchamiana podczas częściowego ładowania.

Typowym użyciem jest załadowanie dziennika zmian i wykorzystanie go do zastosowania inserts, updates oraz deletes do istniejącej tabeli.

> *Aby częściowe ładowanie działało poprawnie, aplikacja musi zostać otwarta z danymi przed jego uruchomieniem.*

Wykonaj częściowe ładowanie za pomocą przycisku **Ładuj**. Możesz także użyć Qlik Engine JSON API.

Składnia:

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

Argumenty:

Argumenty

Argument	Opis
only	Opcjonalny kwalifikator wskazujący, że instrukcja ma być wykonywana tylko podczas częściowego ładowania. Tę instrukcję należy pominąć podczas normalnych (nie częściowych) ładowań.
SequenceNoField	Nazwa pola zawierającego znacznik czasu lub numer kolejny, który definiuje kolejność operacji.

Argument	Opis
SequenceNoVar	Nazwa zmiennej, której przypisywana jest maksymalna wartość SequenceNoField scalanej tabeli.
ListOfKeys	Lista nazw pól oddzielonych przecinkami, określająca klucz podstawowy.
Operation	Pierwsze pole instrukcji load musi zawierać operację jako ciąg tekstowy: „Insert”, „Update” lub „Delete”. Akceptowane są także „i”, „u” i „d”.

Ogólna funkcjonalność

Podczas normalnego (nie częściowego) ładowania konstrukcja **Merge LOAD** działa jako normalna instrukcja **Load**, ale z dodatkową funkcjonalnością polegającą na usuwaniu starszych, zbędnych rekordów oraz rekordów oznaczonych do usunięcia. Pierwsze pole instrukcji **Load** musi zawierać informacje o operacji: Insert, Update lub Delete.

W przypadku każdego załadowanego rekordu identyfikator rekordu jest porównywany z wcześniej załadowanymi rekordami i zachowany zostanie tylko najnowszy rekord (zgodnie z numeracją). Jeśli najnowszy rekord jest oznaczony Delete, żadne rekordy nie zostaną zachowane.

Tabela docelowa

Wybór tabeli do zmodyfikowania zależy od zestawu pól. Jeśli istnieje już tabela z tym samym zestawem pól (z wyjątkiem pierwszego pola, operacji), będzie to odpowiednia tabela do zmodyfikowania. Alternatywnie do określenia tabeli można użyć prefiksu **Concatenate**. Jeśli tabela docelowa nie jest określona, wynik konstrukcji **Merge LOAD** jest zapisywany w nowej tabeli.

Jeśli używany jest przedrostek Concatenate, wynikowa tabela zawiera zestaw pól stanowiący połączenie zbiorów istniejącej tabeli i danych wejściowych do scalenia. W związku z tym tabela docelowa może mieć więcej pól niż dziennik zmian, który jest używany jako dane wejściowe do scalania.

Częściowe ładowanie daje takie same rezultaty jak pełne. Różnica polega na tym, że częściowe ładowanie rzadko tworzy nową tabelę. O ile nie użyto klauzuli **Only**, tabela docelowa z tym samym zestawem pól z poprzedniego wykonania skryptu zawsze istnieje.

Numer sekwencji

Jeśli załadowany dziennik zmian jest dziennikiem akumulowanym, to znaczy zawiera zmiany, które zostały już załadowane, można użyć parametru SequenceNoVar w klauzuli **Where** w celu ograniczenia ilości danych wejściowych. Konstrukcji **Merge LOAD** można wówczas użyć do załadowania tylko rekordów, w których wartość pola SequenceNoField jest większa niż SequenceNoVar. Po zakończeniu konstrukcja **Merge LOAD** przypisuje nową wartość do SequenceNoVar z maksymalną wartością widoczną w polu SequenceNoField.

Operacje

Konstrukcja **Merge LOAD** może mieć mniej pól niż tabela docelowa. Różne operacje inaczej traktują brakujące pola:

Insert: Pola brakujące w **Merge LOAD**, ale istniejące w tabeli docelowej, otrzymują wartość NULL w tabeli docelowej.

Delete: Brakujące pola nie wpływają na wynik. Odpowiednie rekordy są i tak usuwane.

Update: Pola wymienione w **Merge LOAD** są aktualizowane w tabeli docelowej. Brakujące pola nie są zmieniane. Oznacza to, że dwie poniższe instrukcje nie są identyczne:

- 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 ...;

Pierwsza instrukcja aktualizuje wymienione rekordy i zmienia F2 na NULL. Druga nie zmienia F2, ale zamiast tego pozostawia wartości w tabeli docelowej.

Przykłady

Przykład 1: Proste scalanie z określoną tabelą

W tym przykładzie tabela z danymi wpisanymi w skrypcie o nazwie Persons jest ładowana z trzema wierszami. Następnie **Merge** zmienia tę tabelę w następujący sposób:

- Dodaje wiersz *Mary, 4*.
- Usuwa wiersz *Steven, 3*.
- Przypisuje liczbę *5* do *Jake*.

Po wykonaniu **Merge** zmienna *LastChangeDate* jest ustawiana na maksymalną wartość w kolumnie *ChangeDate*.

Skrypt ładowania

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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
];
```

Wynik

Przed wykonaniem **Merge Load** tabela wyników wygląda następująco:

Resulting table

Name	Number
Jake	3
Jill	2
Steven	3

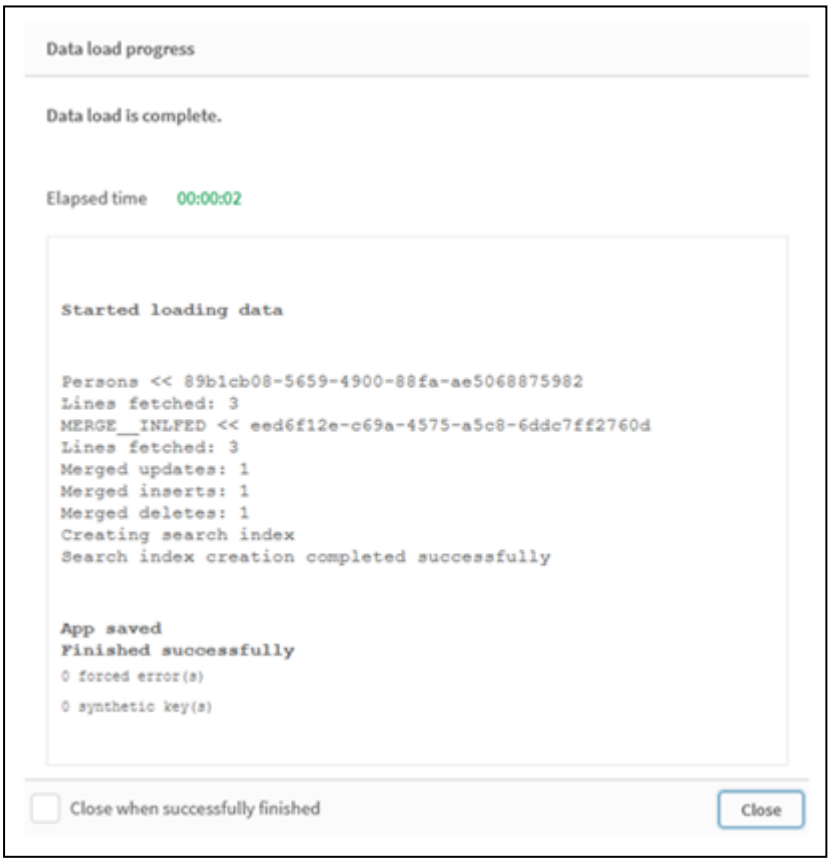
Po wykonaniu **Merge Load** tabela wygląda następująco:

Resulting table

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

Podczas ładowania danych okno dialogowe **Postęp ładowania danych** przedstawia wykonywane operacje:

Okno dialogowe Postęp ładowania danych



Przykład 2: Skrypt ładowania danych z brakującymi polami

W tym przykładzie ładowane są te same dane co powyżej, ale teraz z identyfikatorem dla każdej osoby.

Merge zmienia tabelę w następujący sposób:

- Dodaje wiersz *Mary, 4*.
- Usuwa wiersz *Steven, 3*.
- Przypisuje liczbę *5* do *Jake*.
- Przypisuje liczbę *6* do *Jill*.

Skrypt ładowania

Tutaj używamy dwóch instrukcji **Merge Load** – jednej dla „Insert” i „Delete”, a drugiej dla „Update”.

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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
];
```

Wynik

Po wykonaniu instrukcji **Merge Load** tabela wygląda następująco:

Resulting table

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

Zauważ, że druga instrukcja **Merge** nie zawiera pola **Name**, w związku z czym imiona nie zostały zmienione.

Przykład 3: Skrypt ładowania danych – częściowe ładowanie przy użyciu klauzuli Where z ChangeDate

W poniższym przykładzie argument **Only** określa, że polecenie **Merge** jest wykonywane tylko podczas częściowego ładowania. Aktualizacje są filtrowane na podstawie wcześniej przechwyconych wartości zmiennej LastChangeDate. Po zakończeniu wykonywania instrukcji **Merge** zmiennej LastChangeDate jest przypisywana maksymalna wartość ChangeDate kolumny przetwarzanej podczas scalania.

Skrypt ładowania

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

NoConcatenate

Prefiks **NoConcatenate** wymusza traktowanie dwóch załadowanych tabel z identycznym zestawem pól jako dwóch oddzielnych tabel wewnętrznych, podczas gdy w przeciwnym wypadku automatycznie zostałaby wobec nich zastosowana konkatenacja.

Składnia:

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

Domyślnie, jeśli do wcześniej załadowanej do skryptu tabeli zostanie załadowana tabela zawierająca identyczną liczbę pól o takich samych nazwach, to Qlik Sense automatycznie połączy te tabele. Stanie się tak nawet wtedy, gdy druga tabela będzie miała inną nazwę.

Jeśli jednak przed instrukcją LOAD lub przed instrukcją select drugiej tabeli zostanie dodany prefiks skryptu `NoConcatenate`, to tabele zostaną załadowane osobno.

Typowym przypadkiem użycia `NoConcatenate` jest sytuacja, gdy chcemy utworzyć tymczasową kopię tabeli w celu wykonania na tej kopii pewnych tymczasowych przekształceń, zachowując oryginalne dane. Prefiks `NoConcatenate` zapobiegnie niejawnemu dodaniu tej kopii do tabeli źródłowej.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład funkcji

Przykład	Wynik
`Source: LOAD A,B from file1.csv; CopyOfSource: NoConcatenate LOAD A,B resident Source;`	Została załadowana tabela z `A` i `B` jako miarami. Druga tabela z takimi samymi polami została załadowana osobno przy użyciu zmiennej `NoConcatenate`.

Przykład 1 – Konkatenacja niejawna

Skrypt ładowania i wyniki

Przegląd

W tym przykładzie dodamy dwa skrypty ładowania po kolei.

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Początkowy zestaw danych zawierający daty i kwoty, który został wysłany do tabeli o nazwie `Transactions`.

Pierwszy skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

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

Pierwsza tabela wyników

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

Drugi skrypt ładowania

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Drugi zestaw danych z identycznymi polami został wysłany do tabeli `Sales`.

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

Wyniki

Załaduj dane i przejdź do tabeli.

Druga tabela wyników

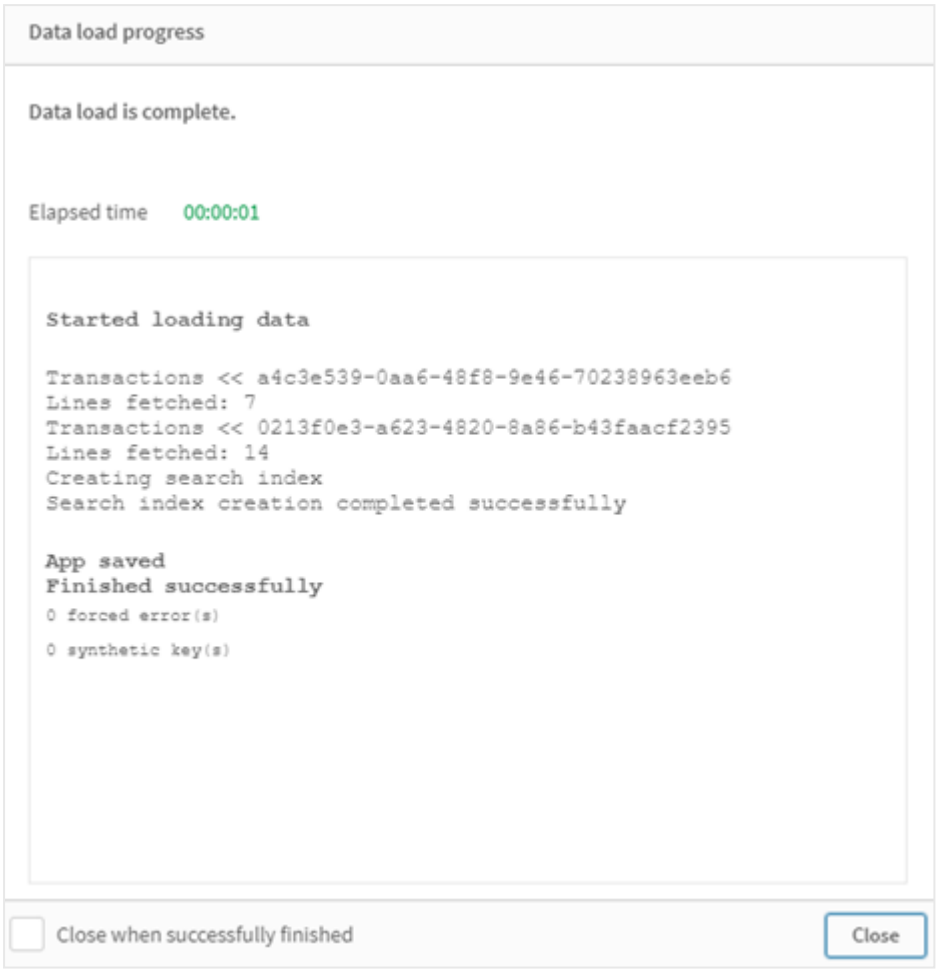
id	date	amount
1	08/30/2018	23.56
2	09/07/2018	556.31
3	09/16/2018	5.75
4	09/22/2018	125.00
5	09/22/2018	484.21

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

Kiedy skrypt zostanie uruchomiony, tabela `Sales` zostanie niejawnie dołączona do istniejącej tabeli `Transactions`, ponieważ oba zestawy danych mają identyczną liczbę pól o takich samym nazwach. Dzieje się tak mimo tego, że znacznik nazwy drugiej tabeli próbuje nadać zestawowi wynikowemu nazwę `'Sales'`.

Aby przekonać się, że zestaw danych Sales został niejawnie dołączony, można zajrzeć do dziennika **postępu ładowania danych**.

Dziennik postępu ładowania danych pokazujący niejawnie dołączone dane z tabeli Transactions.



Przykład 2 – Przypadek użycia

Skrypt ładowania i wyniki

Przegląd

W tym przypadku mamy następujące elementy:

- Zestaw danych transakcji zawierający:
  - id
  - date
  - amount (w GBP)
- Tabela walut zawierająca:
  - Kursy wymiany USD na GBP
- Drugi zestaw danych transakcji:
  - id

- date
- amount (w USD)

Załadujemy pięć skryptów po kolei.

- Pierwszy skrypt ładowania zawiera początkowy zestaw danych zawierający daty i kwoty w GBP, który został wysłany do tabeli o nazwie `Transactions`.
- Drugi skrypt ładowania zawiera następujące elementy:
  - Drugi zestaw danych zawierający daty i kwoty w USD, który został wysłany do tabeli o nazwie `Transactions_in_USD`.
  - Prefiks `noconcatenate`, który został umieszczony przed instrukcją LOAD zestawu danych `Transactions_in_USD`, aby zapobiec niejawnej konkatenacji.
- Trzeci skrypt ładowania zawiera prefiks `join`, który zostanie użyty w celu utworzenia kursu wymiany walut między walutami GBP i USD w tabeli `Transactions_in_USD`.
- Czwarty skrypt ładowania zawiera prefiks `concatenate`, który doda `Transactions_in_USD` do początkowej tabeli `Transactions`.
- Piąty skrypt ładowania zawiera instrukcję `drop table`, która usunie tabelę `Transactions_in_USD` po dodaniu jej danych do tabeli `Transactions`.

Pierwszy skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

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

Wyniki pierwszego skryptu ładowania

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

Tabela przedstawia pierwszy zestaw danych z kwotami w GBP.

Drugi skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i przejdź do tabeli.

Wyniki drugiego skryptu ładowania

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

Zauważysz, że drugi zestaw danych z tabeli `Transactions_in_USD` został dodany.

Trzeci skrypt ładowania

Ten skrypt ładowania łączy kurs wymiany walut z USD na GBP z tabelą `Transactions_in_USD`.

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

Wyniki

Załaduj dane i przejdź do przeglądarki modelu danych. Wybierz tabelę `Transactions_in_USD`, aby zobaczyć, że każdy rekord ma pole „rate” o wartości 0,7.

Czwarty skrypt ładowania

Ten skrypt ładowania, wykorzystujący instrukcję LOAD z predykatem Resident, dołączy tabelę `Transactions_in_USD` do tabeli `Transactions` po wymianie kwot na USD.

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

Wyniki

Załaduj dane i przejdź do tabeli. Zobaczysz nowe pozycje z kwotami w GBP w wierszach od ósmego do czternastego.

Wyniki czwartego skryptu ładowania

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

id	date	amount
5	12/22/2018	484.21
6	12/22/2018	59.18
7	12/23/2018	177.42
8	01/01/2019	114.989
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

Piąty skrypt ładowania

Ten skrypt ładowania usunie duplikaty pozycji z tabeli wyników czwartego skryptu ładowania, pozostawiając tylko pozycje z kwotami w GBP.

```
drop tables Transactions_in_USD;
```

Wyniki

Załaduj dane i przejdź do tabeli.

Wyniki piątego skryptu ładowania

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
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

Po załadowaniu piątego skryptu ładowania, tabela wyników pokazuje wszystkie czternaście transakcji, które znajdowały się w obu zestawach danych transakcji. Jednak w przypadku transakcji 8-14 nastąpiła wymiana waluty na GBP.

Jeśli usuniemy prefiks `NoConcatenate`, który został użyty przed `Transactions_in_USD` w drugim skrypcie ładowania, skrypt spowoduje następujący błąd: „Table '`Transactions_in_USD`' not found”. Stałoby się tak, ponieważ tabela `Transactions_in_USD` zostałaby automatycznie dołączona do oryginalnej tabeli `Transactions`.

Only

Słowo kluczowe skryptu **Only** jest używane jako funkcja agregacji lub jako część składni w prefiksach częściowego przeładowania **Add**, **Replace** i **Merge**.

Outer

Jawny prefiks **Join** może być poprzedzony prefiksem **Outer**, aby określić sprzężenie zewnętrzne. W przypadku sprzężenia zewnętrznego generowane są wszystkie kombinacje między dwiema tabelami. Otrzymana tabela zawiera zatem kombinacje wartości pól z tabel samych danych, gdzie łączące wartości pola są reprezentowane w jednej lub obu tabelach. Słowo **Outer** jest opcjonalne i stanowi domyślny typ sprzężenia używany w sytuacji, gdy nie określono prefiksu sprzężenia.

Składnia:

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

Argumenty:

Argumenty

Argument	Opis
tablename	Tabela nazwana, która ma być porównana do załadowanej tabeli.
loadstatementlub selectstatement	Instrukcja **LOAD** lub **SELECT** dla załadowanej tabeli.

Przykład

Skrypt ładowania

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

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

Tabela wynikowa

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

Objaśnienie

W tym przykładzie dwie tabele, Tabela1 i Tabela2, są scalane w jedną tabelę o nazwie Tabela1. W takich przypadkach prefiks **outer** jest często używany do łączenia kilku tabel w jedną tabelę w celu wykonywania agregacji wartości z pojedynczej tabeli.

Częściowe ładowanie

Pełne ładowanie zawsze rozpoczyna się od usunięcia wszystkich tabel w istniejącym modelu danych, a następnie uruchamiany jest skrypt ładowania.

Inaczej jest w przypadku częściowego ładowania. Zamiast tego zachowuje ono w modelu danych wszystkie tabele, a następnie wykonuje tylko instrukcje **Load** i **Select** poprzedzone prefiksem **Add**, **Merge** lub **Replace**. Polecenie to nie wpływa na inne tabele danych. Argument **only** oznacza, że instrukcja powinna być wykonywana tylko podczas częściowego ładowania i pomijana podczas pełnego ładowania. W poniższej tabeli podsumowano wykonanie instrukcji dla częściowego i pełnego ładowania.

Instrukcja	Pełne ładowanie	Częściowe ładowanie
Load ...	Instrukcja zostanie uruchomiona	Instrukcja nie zostanie uruchomiona
Add/Replace/Merge Load ...	Instrukcja zostanie uruchomiona	Instrukcja zostanie uruchomiona
Ładowanie z instrukcjami Add/Replace/Merge Only...	Instrukcja nie zostanie uruchomiona	Instrukcja zostanie uruchomiona

Częściowe ładowanie ma w porównaniu z pełnym kilka zalet:

- Jest szybsze, ponieważ wystarczy załadować tylko ostatnio zmienione dane. Przy dużych zestawach danych różnica jest znacząca.
- Wykorzystuje się mniej pamięci, ponieważ ładuje się mniej danych.
- Jest bardziej niezawodne, ponieważ zapytania wysyłane do danych źródłowych działają szybciej, co zmniejsza ryzyko problemów z siecią.

Aby częściowe ładowanie działało poprawnie, aplikacja musi zostać otwarta z danymi przed jego uruchomieniem.

Wykonaj częściowe ładowanie za pomocą przycisku **Ładuj**. Możesz także użyć Qlik Engine JSON API.

Ograniczenie

Operacja częściowego ładowania może usunąć wartości z danych. Nie zostanie to jednak odzwierciedlone na liście wartości odrębnych, która jest tabelą obsługiwaną wewnętrznie. Zatem po wykonaniu operacji częściowego przeładowania lista będzie zawierać wszystkie wartości odrębne, które istniały w polu od ostatniego pełnego przeładowania Może być ich więcej niż istnieje obecnie po częściowym przeładowaniu. Ma to wpływ na wyniki funkcji FieldValueCount() i FieldValue(). Funkcja FieldValueCount() teoretycznie może zwrócić liczbę większą niż aktualna liczba wartości pól.

Przykład

Przykład 1:

Skrypt ładowania

Dodaj przykładowy skrypt do aplikacji i wykonaj częściowe ładowanie. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

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

Wynik

Resulting table

Num	Count(Num)
11	1
12	1
13	1
14	1
15	1
16	1
17	1

Num	Count(Num)
18	1
19	1
20	1

Objaśnienie

Instrukcja jest wykonywana tylko podczas częściowego ładowania. Jeśli prefiks „distinct” zostanie pominięty, liczba w polu **Num** będzie się zwiększać z każdym kolejnym częściowym ładowaniem.

Przykład 2

Skrypt ładowania

Dodaj przykładowy skrypt do aplikacji. Wykonaj pełne ładowanie i zobacz wynik. Następnie wykonaj częściowe ładowanie i zobacz wynik. Aby zobaczyć wyniki, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

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

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

Wynik

Output table after full reload

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

Output table after partial reload

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

Objaśnienie

Pierwsza tabela jest ładowana podczas pełnego ładowania, a druga tabela po prostu zastępuje pierwszą tabelę podczas częściowego ładowania.

Replace

Słowo kluczowe skryptu **Replace** jest używane jako funkcja ciągu znaków albo jako prefiks w częściowym przeładowaniu.

Replace

Prefiks **Replace** można dodać do dowolnej instrukcji **LOAD** lub **SELECT** w skrypcie, aby określić, że ładowana tabela powinna zastąpić inną tabelę. Określa on również, że ta instrukcja powinna być uruchamiana podczas częściowego ładowania. Prefiksu **Replace** można też użyć w instrukcji **Map**.

Aby częściowe ładowanie działało poprawnie, aplikacja musi zostać otwarta z danymi przed jego uruchomieniem.

Wykonaj częściowe ładowanie za pomocą przycisku **Ładuj**. Możesz także użyć Qlik Engine JSON API.

Składnia:

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

```
Replace [only] mapstatement
```

Podczas normalnego (nie częściowego) ładowania konstrukcja **Replace LOAD** będzie działać jako normalna instrukcja **LOAD**, ale poprzedzona **Drop Table**. Najpierw stara tabela zostanie usunięta, a następnie rekordy zostaną wygenerowane i zapisane jako nowa tabela.

Jeśli będzie używany prefiks **Concatenate** lub jeśli będzie istnieć tabela z tym samym zestawem pól, będzie to odpowiednia tabela do usunięcia. W przeciwnym razie nie będzie tabeli do usunięcia, a konstrukcja **Replace LOAD** będzie identyczna jak normalna instrukcja **LOAD**.

Częściowe ładowanie da takie same rezultaty. Jedyna różnica polega na tym, że zawsze istnieje tabela do usunięcia z poprzedniego wykonania skryptu. Konstrukcja **Replace LOAD** zawsze najpierw usunie starą tabelę, a następnie utworzy nową.

W przypadku instrukcji**Replace Map...Using** mapowanie jest przeprowadzane również podczas częściowego wykonywania skryptu.

Argumenty:

Argumenty

Argument	Opisu
only	Opcjonalny kwalifikator wskazujący, że instrukcja ma być wykonywana tylko podczas częściowego ładowania. Należy go pominąć podczas normalnych (nie częściowych) ładowań.

Przykłady i wyniki:

Przykład	Wynik
`Tab1: Replace LOAD * from File1.csv;`	Podczas normalnego i częściowego przeładowania tabela Qlik Sense o nazwie Tab1 jest najpierw usuwana. Nowe dane są następnie ładowane z pliku File1.csv i zapisywane w tabeli Tab1.
`Tab1: Replace only LOAD * from File1.csv;`	Podczas normalnego przeładowania ta instrukcja jest pomijana. Podczas częściowego przeładowania tabela Qlik Sense o poprzedniej nazwie Tab1 jest najpierw usuwana. Nowe dane są następnie ładowane z pliku File1.csv i zapisywane w tabeli Tab1.
`Tab1: LOAD a,b,c from File1.csv; Replace LOAD a,b,c from File2.csv;`	Podczas normalnego przeładowania File1.csv jest wczytywany do tabeli Qlik Sense o nazwie Tab1, która następnie zostaje usunięta i zastąpiona nowymi danymi załadowanymi z pliku File2.csv. Następuje utrata wszystkich danych z pliku File1.csv. Podczas częściowego przeładowania tabela Qlik Sense o nazwie Tab1 jest najpierw usuwana. Następnie jest ona zastępowana nowymi danymi załadowanymi z pliku File2.csv.
`Tab1: LOAD a,b,c from File1.csv; Replace only LOAD a,b,c from File2.csv;`	Podczas normalnego przeładowania dane są ładowane z pliku File1.csv i zapisywane w tabeli Qlik Sense o nazwie Tab1. Plik File2.csv jest pomijany. Podczas częściowego przeładowania tabela Qlik Sense o nazwie Tab1 jest najpierw usuwana. Następnie jest ona zastępowana nowymi danymi załadowanymi z pliku File2.csv. Następuje utrata wszystkich danych z pliku File1.csv.

Right

Prefiksy **Join** i **Keep** mogą być poprzedzone prefiksem **right**.

Podanie go przed prefiksem **join** oznacza, że ma być używane prawe sprzężenie. Wynikowa tabela będzie zawierać tylko te kombinacje wartości pól z pierwotnych tabel danych, dla których wartości pola łączącego są obecne w drugiej tabeli. Podanie go przed prefiksem **keep** oznacza, że przed zapisaniem w aplikacji Qlik Sense pierwsza tabela surowych danych ma zostać zredukowana do części wspólnej z drugą tabelą.

Szukasz funkcji ciągu znaków według tej samej nazwy? Zob.: Right (page 1429)

Składnia:

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

Argumenty:

Argumenty

Argument	Opis
tablename	Tabela nazwana, która ma być porównana do załadowanej tabeli.
loadstatementlub selectstatement	Instrukcja **LOAD** lub **SELECT** dla załadowanej tabeli.

Przykład

Skrypt ładowania

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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 ];
```

Wynik

Tabela wynikowa

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

Objaśnienie

Ten przykład ilustruje dane wyjściowe prawego sprzężenia, w którym łączone są tylko wartości obecne w drugiej (prawej) tabeli.

Sample

Prefiks **sample** w instrukcji **LOAD** lub **SELECT** służy do ładowania ze źródła danych losowej próbki rekordów.

Składnia:

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

Wyrażenie, które jest obliczane, nie definiuje, jaki procent rekordów z zestawu danych zostanie załadowany do aplikacji Qlik Sense, tylko prawdopodobieństwo załadowania do aplikacji każdego z wczytywanych rekordów. Innymi słowy, wartość `p = 0.5` nie oznacza, że zostanie załadowane 50% wszystkich rekordów, tylko że każdy rekord ma 50% szans na zostanie załadowanym do aplikacji Qlik Sense.

Argumenty

Argument	Opis
p	Dowolne wyrażenie, które zwraca wartość liczbową większą od 0, a mniejszą lub równą 1. Liczba określa prawdopodobieństwo odczytania konkretnego rekordu. Wszystkie rekordy zostaną odczytane, ale tylko niektóre z nich zostaną załadowane do aplikacji Qlik Sense.

Kiedy używać

Funkcja Sample jest przydatna, kiedy chcemy pobrać próbkę danych z dużej tabeli, aby zrozumieć ich naturę, rozkład lub zawartość pola. Dzięki temu, że pobierany jest podzbiór danych, są one szybciej ładowane, co umożliwia szybsze testowanie skryptów. W odróżnieniu od funkcji `First`, funkcja `Sample` pobiera dane z całej tabeli, a nie tylko z kilku pierwszych wierszy. W niektórych przypadkach pozwala to uzyskać dokładniejszą reprezentację danych.

Poniższe przykłady pokazują dwa możliwe zastosowania prefiksu skryptu `Sample`:

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

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Próbka z tabeli wbudowanej

Skrypt ładowania i wyniki

Przegląd

W tym przykładzie skrypt ładuje próbkę danych z zestawu zawierającego siedem rekordów do tabeli o nazwie `Transactions` z tabeli wbudowanej.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `amount`

Dodaj następującą miarę:

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

Tabela wynikowa

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

Podczas iteracji w tym przykładzie wczytano wszystkie siedem rekordów, ale do tabeli danych zostały załadowane tylko cztery rekordy. Za każdym razem zostałaby wczytana inna liczba rekordów oraz zostałby załadowany do aplikacji inny zestaw rekordów.

Przykład 2 – Próbka z automatycznie wygenerowanej tabeli

Skrypt ładowania i wyniki

Przegląd

W tym przykładzie, z użyciem `Autogenerate`, został utworzony zestaw danych zawierający 100 rekordów z polami `date`, `id` i `amount`. Jednak został użyty prefiks `Sample` o wartości 0,1.

Skrypt ładowania

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

Autogenerate(100);
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `amount`

Dodaj następującą miarę:

Tabela wynikowa

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

W iteracji ładowania w tym przykładzie zostało załadowanych siedem rekordów z utworzonego zestawu danych. W tym przypadku także za każdym razem zostałaby wczytana inna liczba rekordów oraz zostałby załadowany do aplikacji inny zestaw rekordów.

Semantic

Prefiks ładowania `semantic` tworzy specjalny typ pola, którego można używać w Qlik Sense do łączenia i zarządzania danymi relacyjnymi, takimi jak struktury drzewiaste, samoodwołujące się dane strukturalne nadrzędne-podrzędne lub dane, które można opisać w postaci wykresu.

Należy zauważyć, że ładowanie `semantic` może działać podobnie do prefiksów *Hierarchy (page 63)* i *HierarchyBelongsTo (page 64)*. Wszystkie trzy prefiksy mogą być używane jako bloki do budowy efektywnych rozwiązań front-end do przechodzenia przez dane relacyjne.

Składnia:

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

Ładunek Semantic oczekuje danych wejściowych o szerokości dokładnie trzech lub czterech pól z dokładną definicją tego, co reprezentuje każde uporządkowane pole, jak pokazano w poniższej tabeli:

Pola ładowania Semantic

Nazwa pola	Opis pola
1st Field:	Ten znacznik jest reprezentacją pierwszego z dwóch obiektów, między którymi istnieje relacja.
2nd Field:	Znacznik ten będzie używany do opisania relacji „do przodu” między pierwszym a drugim obiektem. Jeśli pierwszy obiekt jest podrzędny, a drugi nadrzędny, możesz utworzyć kartę relacji, która wskazuje „nadrzędny” lub „nadrzędny dla”, tak jak w przypadku śledzenia relacji od elementu podrzędnego do nadrzędnego.
3rd Field:	Ten znacznik jest reprezentacją drugiego z dwóch obiektów, między którymi istnieje relacja.
4th Field:	To pole jest opcjonalne. Znacznik ten będzie używany do opisania relacji „do tyłu” lub „odwrotnej” między pierwszym a drugim obiektem. Jeśli pierwszy obiekt jest podrzędny, a drugi nadrzędny, możesz utworzyć kartę relacji, która wskazuje „podrzędny” lub „podrzędny dla”, tak jak w przypadku śledzenia relacji od elementu nadrzędnego do podrzędnego. Jeśli nie dodasz czwartego pola, do opisu relacji w obu kierunkach zostanie użyty znacznik drugiego pola. W takim przypadku symbol strzałki jest automatycznie dodawany jako część znacznika.

Poniższy kod jest przykładem prefiksu `semantic`.

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

Dozwoloną i typową praktyką jest oznaczanie trzeciego pola tak samo jak pierwszego pola. Tworzy to samoodwołujące się wyszukiwanie, dzięki czemu można podążać za obiektami do powiązanych obiektów relacja po relacji. Jeśli trzecie pole nie nosi tej samej nazwy, wynikiem końcowym będzie proste wyszukiwanie od obiektów do ich bezpośrednich relacyjnych sąsiadów tylko o krok, co jest wynikiem mało praktycznym.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Powiązane funkcje

Funkcje	Interakcja
Hierarchy (page 63)	Prefiks ładowania Hierarchy służy do dzielenia i organizowania węzłów w strukturach danych typu nadrzędny-podrzędny i innych podobnych do wykresów oraz do przekształcania ich w tabele.
HierarchyBelongsTo (page 64)	Prefiks ładowania HierarchyBelongsTo służy do znajdowania i organizowania elementów nadrzędnych w strukturach danych typu nadrzędny-podrzędny i innych podobnych do wykresów oraz do przekształcania ich w tabele.

Przykład - Tworzenie specjalnego pola do łączenia relacji za pomocą prefiksu Semantic

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych reprezentujący rekordy relacji geograficznych, który jest ładowany do tabeli o nazwie `GeographyTree`.

    - Każdy wpis ma ID na początku linii i ParentID na końcu linii.
- Prefiks `semantic`, który doda jedno specjalne pole zachowania o nazwie `Relation`.

Skrypt ładowania

```
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;
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `Id`
- `Geography`

Następnie utwórz panel filtrowania z `Relation` jako wymiarem. Kliknij przycisk **Koniec edycji**.

Tabela wynikowa

Id	Geography
1	World
2	Europe

Id	Geography
3	Asia
4	North America
5	South America
6	UK
7	Germany
8	Sweden
9	Korea Południowa
10	North Korea
11	Chiny
12	London
13	Birmingham

Panel filtrowania

Relation
Child
Parent

Kliknij **Europe** w wymiarze `Geography` w tabeli i kliknij **Child** w wymiarze `Relation` w panelu filtrowania. Zwróć uwagę na oczekiwany wynik w tabeli:

Tabela wyników pokazująca elementy podrzędne regionu geograficznego Europe

Id	Geography
6	UK
7	Germany
8	Sweden

Ponowne kliknięcie **Child** spowoduje pokazanie miejsc, które są elementami podrzędnymi regionu UK, o jeden krok dalej.

Tabela wyników pokazująca elementy podrzędne regionu geograficznego UK

Id	Geography
12	London
13	Birmingham

Unless

Prefiks i sufiks **unless** jest używany do tworzenia klauzuli warunkowej określającej, czy dana instrukcja lub klauzula wyjściowa ma być sprawdzana czy też nie. Mogą być one traktowane jako wygodna alternatywa pełnej instrukcji **if..end if**.

Składnia:

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

Instrukcja **statement** lub **exitstatement** zostanie wykonana tylko wtedy, gdy **condition** przyjmie wartość False.

Prefiks **unless** może być używany do instrukcji, które już składają się z jednej lub wielu innych instrukcji, włącznie z dodatkowymi prefiksami **when** lub **unless**.

Argumenty

Argument	Opis
condition	Wyrażenie logiczne, którego ocena zwraca True lub False.
statement	Dowolna instrukcja skryptu aplikacji Qlik Sense poza instrukcjami sterowania.
exitstatement	Klauzula **exit for**, **exit do** lub **exit sub** albo instrukcja **exit script**.

Kiedy używać

Instrukcja `Unless` zwraca wynik logiczny. Normalnie tego typu funkcje są używane jako warunek, gdy użytkownik chce warunkowo załadować lub wykluczyć wybrane części skryptu.

Trzy poniższe wiersze pokazują różne możliwości użycia funkcji `Unless`:

```
exit script unless A=1;
unless A=1 LOAD * from myfile.csv;
unless A=1 when B=2 drop table Tab1;
```

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Prefiks Unless

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Utworzenie zmiennej A, której zostaje nadana wartość 1.
- Zestaw danych, który jest ładowany do tabeli o nazwie Transactions, chyba że zmienna A = 2.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

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

Tabela wynikowa

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

Ponieważ na początku skryptu zmiennej A została przypisana wartość 1, następuje sprawdzenie warunku znajdującego się za prefiksem `Unless`, czego wynikiem jest wartość logiczna `FALSE`. W efekcie skrypt kontynuuje wykonywanie instrukcji `Load`. W tabeli wyników widać wszystkie rekordy z tabeli `Transactions`.

Jeśli wartość tej zmiennej zostanie ustawiona na 2, do modelu danych nie zostaną załadowane żadne dane.

Przykład 2 – Sufiks Unless

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zaczyna od załadowania początkowego zestawu danych do tabeli o nazwie `Transactions`. Następnie skrypt zostaje zamknięty, chyba że w tabeli `Transactions` jest mniej niż dziesięć rekordów.

Jeśli ten warunek nie spowoduje zamknięcia skryptu, do tabeli `Transactions` zostaje dołączony dalszy zestaw transakcji i proces zostaje powtórzony.

Skrypt ładowania

```
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 ;
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- id
- date
- amount

Tabela wynikowa

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

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

W każdym z trzech zestawów danych skryptu ładowania znajduje się siedem rekordów.

Pierwszy zestaw danych (z `id` transakcji od 1 do 7) zostaje załadowany do aplikacji. Warunek `Unless` sprawdza, czy w tabeli `Transactions` znajduje się mniej niż 10 wierszy. Zostaje zwrócona wartość `TRUE`, więc do aplikacji zostaje załadowany drugi zestaw danych (z transakcjami o `id` od 8 do 14). Drugi warunek `Unless` sprawdza, czy w tabeli `Transactions` znajduje się mniej niż 10 rekordów. Zostaje zwrócona wartość `FALSE`, więc skrypt kończy działanie.

Przykład 3 – Kilka prefiksów Unless

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

W tym przykładzie jest tworzony zestaw danych w formie tabeli o nazwie `Transactions` zawierającej jedną transakcję. Następnie zostaje wykonana pętla „for” zawierająca dwie zagnieżdżone instrukcje Unless, które sprawdzają następujące warunki:

1. Czy w tabeli `Transactions` znajduje się więcej niż 100 rekordów
2. Czy liczba rekordów w tabeli `Transactions` jest wielokrotnością liczby 6

Jeśli te warunki nie są spełnione (`FALSE`), następuje wygenerowanie siedmiu kolejnych rekordów, które zostają dołączone do istniejącej tabeli `Transactions`. Ten proces jest powtarzany aż jedna z dwóch transakcji zwróci wartość `TRUE`.

Skrypt ładowania

```
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
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:`id`.

Tabela wynikowa

id
0
1
2
3
4
5
Jeszcze 30 wierszy

Instrukcje Unless zagnieżdżone w pętli „for” sprawdzają następujące warunki:

1. Czy w tabeli `Transactions` znajduje się więcej niż 100 wierszy?
2. Czy suma rekordów w tabeli `Transactions` jest wielokrotnością liczby 6?

Jeśli obie instrukcje Unless zwracają wartość `FALSE`, następuje wygenerowanie siedmiu kolejnych rekordów, które zostają dołączone do istniejącej tabeli `Transactions`.

Instrukcje te zwracają wartość `FALSE` pięć razy, po czym liczba wierszy w tabeli `Transactions` wynosi 36.

W tym momencie druga instrukcja `Unless` zwraca wartość `TRUE`, przez co znajdująca się za nią instrukcja ładowania nie zostaje wykonana.

When

Prefiks i sufiks **when** jest używany do tworzenia klauzuli warunkowej określającej, czy dana instrukcja lub klauzula wyjściowa ma być wykonywana czy też nie. Mogą być one traktowane jako wygodna alternatywa pełnej instrukcji **if..end if**.

Składnia:

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

Typ zwracanych danych: Wartość logiczna

W Qlik Sense wartość logiczna Prawda jest reprezentowana przez -1, a wartość Fałsz jest reprezentowana przez 0.

Instrukcja **statement** lub **exitstatement** zostanie wykonana tylko wtedy, gdy warunek przyjmie wartość TRUE.

Prefiks `when` może być używany do instrukcji, które już składają się z jednej lub wielu innych instrukcji, włącznie z dodatkowymi prefiksami `when` lub `Unless`.

Kiedy używać

Instrukcja `when` zwraca wynik logiczny. Normalnie tego typu funkcje są używane jako warunek, gdy użytkownik chce załadować lub wykluczyć wybrane części skryptu.

Argumenty

Argument	Opis
condition	Wyrażenie logiczne zwracające wartość TRUE lub FALSE
statement	Dowolna instrukcja skryptu aplikacji Qlik Sense poza instrukcjami sterowania.
exitstatement	Klauzula **exit for**, **exit do** lub **exit sub** albo instrukcja **exit script**.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
`exit script when A=1;`	Kiedy instrukcja `A=1` zwraca wartość TRUE, następuje zakończenie wykonywania skryptu.
`when A=1 LOAD * from myfile.csv;`	Kiedy instrukcja `A=1` zwraca wartość TRUE, następuje załadowanie pliku `myfile.csv`.
`when A=1 unless B=2 drop table Tab1;`	Kiedy instrukcja `A=1` zwraca wartość TRUE, a `B=2` zwraca FALSE, następuje usunięcie tabeli `Tab1`.

Przykład 1 – Prefiks When

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający daty i kwoty, który został wysłany do tabeli o nazwie `Transactions`.
- Instrukcja `Let`, która oświadcza, że zmienna `A` jest utworzona i ma wartość `1`.
- Warunek `When` według którego, jeśli `A` równa się `1`, to skrypt ma kontynuować ładowanie.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- id
- date
- amount

Tabela wynikowa

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

Ponieważ na początku skryptu zmiennej `A` została przypisana wartość `1`, następuje sprawdzenie warunku znajdującego się za prefiksem `When`, czego wynikiem jest wartość logiczna TRUE. Ponieważ zostaje zwrócona wartość TRUE, skrypt kontynuuje wykonywanie instrukcji ładowania. Widać wszystkie rekordy z tabeli wyników.

Jeśli wartość tej zmiennej zostanie ustawiona na wartość różną od `1`, do modelu danych nie zostaną załadowane żadne dane.

Przykład 2 – sufiks When

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Trzy zestawy danych zawierające daty i kwoty, które zostały wysłany do tabeli o nazwie `Transactions`.
    - Pierwszy zestaw zawiera transakcje 1-7.
    - Pierwszy zestaw zawiera transakcje 8-14.
    - Pierwszy zestaw zawiera transakcje 15-21.
- Warunek `When`, który określa, czy tabela `Transactions` zawiera więcej niż dziesięć wierszy. Jeśli którakolwiek z instrukcji `When` zwraca wartość TRUE, następuje zakończenie wykonywania skryptu. Ten warunek został umieszczony na końcu każdego z trzech zestawów danych.

Skrypt ładowania

```
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 ;
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- id
- date
- amount

Tabela wynikowa

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

W każdym z trzech zestawów danych znajduje się siedem transakcji. Pierwszy zestaw danych zawiera transakcje 1-7 i zostaje załadowany do aplikacji. Warunek `When` znajdujący się za tą instrukcją ładowania zwraca wartość FALSE, ponieważ w tabeli `Transactions` znajduje się mniej niż dziesięć wierszy. Skrypt ładowania przechodzi do następnego zestawu danych.

Drugi zestaw danych zawiera transakcje 8-14 i zostaje załadowany do aplikacji. Drugi warunek `When` zwraca wartość TRUE, ponieważ w tabeli `Transactions` znajduje się więcej niż dziesięć wierszy. W efekcie skrypt kończy działanie.

Przykład 3 – Kilka prefiksów When

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Został utworzony zestaw danych zawierający jedną transakcję w postaci tabeli o nazwie `Transactions`.
- Pętla `For` zawiera dwa zagnieżdżone warunki `When`, które sprawdzają, czy:

1. W tabeli `Transactions` jest mniej niż `100` rekordów.
2. Liczba rekordów w tabeli `Transactions` jest wielokrotnością liczby `6`.

Skrypt ładowania

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

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

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

- `id`

Tabela wyników pokazuje tylko pięć pierwszych identyfikatorów transakcji, ale skrypt ładowania tworzy 36 wierszy, po czym kończy działanie, gdy zostaje spełniony warunek `When`.

Tabela wynikowa

id
0
1
2
3
4
5
Jeszcze 30 wierszy

Zagnieżdżone warunki `When` w pętli `For` dają odpowiedzi na następujące pytania:

- Czy w tabeli `Transactions` znajduje się mniej niż 100 wierszy?
- Czy suma rekordów w tabeli `Transactions` nie jest wielokrotnością liczby 6?

Jeśli oba warunki `When` zwracają wartość TRUE, następuje wygenerowanie siedmiu kolejnych rekordów, które zostają dołączone do istniejącej tabeli `Transactions`.

Warunki `When` zwracają wartość TRUE pięć razy. W tym momencie w tabeli `Transactions` znajduje się 36 wierszy danych.

Kiedy w tabeli `Transactions` jest 36 wierszy danych, druga instrukcja `When` zwraca wartość FALSE, przez co znajdująca się za nią instrukcja ładowania nie zostaje wykonana.

2.5 Zwykłe instrukcje skryptu

Zwykłe instrukcje służą zazwyczaj do wykonywania operacji na danych. Każda taka instrukcja może obejmować w skrypcie dowolną liczbę wierszy i musi zawsze być zakończona średnikiem, czyli znakiem „;”.

Słowa kluczowe w skrypcie mogą być wpisywane z użyciem dowolnych kombinacji małych i wielkich liter. Wielkość liter jest natomiast uwzględniana w nazwach pól i zmiennych używanych w instrukcjach.

Przegląd zwykłych instrukcji skryptu

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

Alias

Instrukcja **alias** służy do ustawiania aliasu, który będzie używany jako nazwa pola we wszystkich wystąpieniach tego pola w skrypcie.

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

Autonumber

Ta instrukcja zwraca niepowtarzalną wartość całkowitą dla każdej odrębnie przetwarzanej wartości w polu napotkanym podczas wykonywania skryptu.

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

Binary

Instrukcja **binary** służy do ładowania danych z innego dokumentu QlikView, w tym danych z instrukcji Section Access.

```
Binary  [path] filename
```

comment

Udostępnia sposób wyświetlania komentarzy do pola (metadanych) z baz danych i arkuszy kalkulacyjnych. Nazwy pól nieobecne w aplikacji są ignorowane. W przypadku znalezienia wielu wystąpień nazwy pola używana jest ostatnia wartość.

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

comment table

Udostępnia sposób wyświetlania komentarzy do tabeli (metadanych) z baz danych lub arkuszy kalkulacyjnych.

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

Connect

Ta funkcjonalność jest niedostępna w środowisku Qlik Sense SaaS.

Instrukcja **CONNECT** służy do określania dostępu aplikacji Qlik Sense do ogólnej bazy danych przy użyciu interfejsu OLE DB/ODBC. W przypadku ODBC źródło danych najpierw należy określić za pomocą narzędzia administracyjnego ODBC.

```
ODBC Connect TO connect-string [ ( access_info ) ]
OLEDB CONNECT TO connect-string [ ( access_info ) ]
CUSTOM CONNECT TO connect-string [ ( access_info ) ]
LIB CONNECT TO connection
```

Declare

Instrukcja **Declare** służy do tworzenia definicji pól, w których można określić relacje między polami a funkcjami. Definicje zestawów pól można wykorzystać do automatycznego generowania pochodnych pól, których można użyć jako wymiarów. Można na przykład utworzyć definicję kalendarza i użyć jej do wygenerowania z pola daty powiązanych wymiarów takich jak rok, miesiąc, tydzień i dzień.

```
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

Instrukcja **Derive** służy do generowania pochodnych pól na podstawie definicji pola utworzonej przy użyciu instrukcji **Declare**. Można określić, dla których pól danych mają być tworzone pochodne pola, lub tworzyć je jawnie lub niejawnie na podstawie znaczników pól.

```
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
```

Directory

Instrukcja **Directory** określa, w którym katalogu należy szukać plików danych w kolejnych instrukcjach **LOAD**, do momentu wydania nowej instrukcji **Directory**.

```
Directory [path]
```

Disconnect

Instrukcja **Disconnect** przerywa bieżące połączenieODBC/OLE DB/Niestandardowe. Ta instrukcja jest opcjonalna.

```
Disconnect
```

drop field

W dowolnym momencie podczas wykonywania skryptu można usunąć z modelu danych, a zarazem z pamięci, pola aplikacji Qlik Sense, korzystając w tym celu z instrukcji **drop field**.

> *Dozwolone są obydwie formy **drop field** i **drop fields**, a użycie każdej z nich zapewnia ten sam efekt. Jeśli żadna tabela nie zostanie określona, pole zostanie usunięte z wszystkich tabel, w których występuje.*

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

drop table

W dowolnym momencie podczas wykonywania skryptu można usunąć z modelu danych, a zarazem z pamięci, wewnętrzne tabele aplikacji Qlik Sense, korzystając w tym celu z instrukcji **drop table**.

> *Akceptowane są zarówno formularze **drop table**, jak i **drop tables**.*

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

Execute

Instrukcja **Execute** służy do uruchamiania innych programów w czasie, gdy aplikacja Qlik Sense ładuje dane. Na przykład w celu wykonania niezbędnych przekształceń.

```
Execute commandline
```

FlushLog

Instrukcja **FlushLog** wymusza w aplikacji Qlik Sense zapisywanie zawartości bufora skryptu do pliku dziennika skryptu.

```
FlushLog
```

Force

Instrukcja **force** wymusza w aplikacji Qlik Sense interpretację nazw i wartości pól kolejnych instrukcji **LOAD** i **SELECT** jako pisanych tylko wielkimi literami, tylko małymi literami, jak nazwy własne lub tak, jak są wyświetlane (mieszane). Ta instrukcja umożliwia powiązanie wartości pól z tabeli utworzonych zgodnie z różnymi konwencjami.

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

LOAD

Instrukcja **LOAD** ładuje pola z pliku, z danych zdefiniowanych w skrypcie, z wcześniej załadowanej tabeli, ze strony internetowej, z wyniku późniejszej instrukcji **SELECT** lub przez automatyczne wygenerowanie danych. Możliwe jest również ładowanie danych z połączeń analitycznych.

```
 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

Instrukcja **let**, uzupełniająca instrukcję **set**, służy do określania zmiennych skryptu. Instrukcja **let** w przeciwieństwie do instrukcji **set** ocenia wyrażenie po prawej stronie znaku „=” w czasie wykonywania skryptu, zanim zostanie przypisane do zmiennej.

```
Let variablename=expression
```

Loosen Table

Instrukcja **Loosen Table** umożliwia jawne deklarowanie wewnętrznych tabel danych Qlik Sense jako luźno powiązanych na etapie wykonywania skryptu. Gdy tabela jest luźno powiązana, wszystkie asocjacje między wartościami pola w tabeli są usuwane. Podobny efekt można uzyskać przez załadowanie poszczególnych pól luźno powiązanej tabeli jako niezależnych, niepołączonych tabel. Luźne powiązanie może być użyteczne podczas testowania w celu tymczasowego izolowania różnych części struktury danych. Luźno powiązaną tabelę można zidentyfikować w przeglądarce tabel na podstawie linii przerywanych. Jeśli w skrypcie występuje jakakolwiek instrukcja **Loosen Table**, aplikacja Qlik Sense zignoruje wszelkie ustawienia tabel luźno powiązanych obowiązujące przed wykonaniem skryptu.

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

Map ... using

Instrukcja **map ... using** służy do mapowania określonej wartości pola lub wyrażenia na wartości we wskazanej tabeli mapowania. Tabelę mapowania tworzy się instrukcją **Mapping**.

```
Map  *fieldlist Using  mapname
```

NullAsNull

Instrukcja **NullAsNull** wyłącza konwersję wartości NULL na wartości ciągów znaków ustawione wcześniej przez instrukcję **NullAsValue**.

```
NullAsNull *fieldlist
```

NullAsValue

Instrukcja **NullAsValue** określa, dla których pól wartość NULL powinna być przekształcona w wartość.

```
NullAsValue *fieldlist
```

Qualify

Instrukcja **Qualify** służy do przełączania kwalifikacji nazw pól, na przykład aby nazwy te przyjmowały jako prefiks nazwę tabeli.

```
Qualify *fieldlist
```

Rem

Instrukcja **rem** służy do wstawiania komentarzy do skryptu lub tymczasowego dezaktywowania instrukcji w skrypcie bez usuwania ich.

```
Rem string
```

Rename Field

Ta funkcja skryptu zmienia nazwy istniejących pól aplikacji Qlik Sense po ich załadowaniu.

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

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

Rename Table

Ta funkcja skryptu zmienia nazwy istniejących tabel wewnętrznych aplikacji Qlik Sense po ich załadowaniu.

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

Section

Instrukcja **section** pozwala określić, czy następujące po niej instrukcje **LOAD** i **SELECT** mają być traktowane jako dane czy jako definicje praw dostępu.

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

Select

Pola są wybierane ze źródła danych ODBC lub dostawcy OLE DB z użyciem standardowych instrukcji **SELECT** języka SQL. Akceptacja instrukcji **SELECT** zależy jednak od używanego sterownika ODBC lub dostawcy OLE DB.

```
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

Instrukcja **set** jest używana do określania zmiennych skryptu. Mogą one służyć do zastępowania ciągów znaków, ścieżek, dysków itp.

```
Set variablename=string
```

Sleep

Instrukcja **sleep** wstrzymuje wykonanie skryptu przez określony czas.

```
Sleep n
```

SQL

Instrukcja **SQL** umożliwia wysłanie dowolnego polecenia SQL przez połączenie ODBC lub OLE DB.

```
SQL sql_command
```

SQLColumns

Instrukcja **sqlcolumns** zwraca zestaw pól opisujących kolumny źródła danych ODBC lub OLE DB, do którego utworzono połączenie **connect**.

```
SQLColumns
```

SQLTables

Instrukcja **sqltables** zwraca zestaw pól opisujących tabele źródła danych ODBC lub OLE DB, do którego utworzono połączenie **connect**.

```
SQLTables
```

SQLTypes

Instrukcja **sqltypes** zwraca zestaw pól opisujących typy źródła danych ODBC lub OLE DB, do którego utworzono połączenie **connect**.

```
SQLTypes
```

Star

Ciąg danych używany na potrzeby reprezentacji zestawu wszystkich wartości pola w bazie danych można ustawić przy użyciu instrukcji **star**. Wpływa on na następne instrukcje **LOAD** i **SELECT**.

```
Star is [ string ]
```

Store

Instrukcja **Store** tworzy plik QVD, CSV lub text.

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

Tag

Ta instrukcja skryptu umożliwia przypisywanie znaczników do jednego lub większej liczby pól lub tabel. W przypadku próby oznaczenia pola lub tabeli niewystępującej w aplikacji oznaczanie zostanie zignorowane. W przypadku znalezienia niezgodnych wystąpień pola lub znacznika używana jest ostatnia wartość.

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

```
Tag table tablelist with tagname
```

Trace

Instrukcja **trace** zapisuje ciąg znaków w oknie **Postęp wykonania skryptu** oraz w pliku dziennika skryptu (jeśli jest używany). Może być bardzo przydatna do celów debugowania. Komunikat taki można dostosować przy użyciu rozszerzeń „$" zmiennych obliczanych przed instrukcją **trace**.

```
Trace string
```

Unmap

Instrukcja **Unmap** powoduje dezaktywację mapowania wartości pola określonej w poprzedniej instrukcji **Map … Using** dla kolejno ładowanych pól.

```
Unmap *fieldlist
```

Unqualify

Instrukcja **Unqualify** służy do wyłączania kwalifikowania nazw pól, które zostało wcześniej włączone przy użyciu instrukcji **Qualify**.

```
Unqualify *fieldlist
```

Untag

Ta instrukcja skryptu umożliwia usuwanie znaczników z pól lub tabel. W przypadku próby usunięcia oznaczenia pola lub tabeli niewystępującej w aplikacji usunięcie oznaczenia zostanie zignorowane.

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

Alias

Instrukcja **alias** służy do ustawiania aliasu, który będzie używany jako nazwa pola we wszystkich wystąpieniach tego pola w skrypcie.

Składnia:

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

Argumenty:

Argumenty

Argument	Opis
fieldname	Nazwa pola w danych źródłowych
aliasname	Alias, który ma być użyty zamiast niej

Przykłady i wyniki:

Przykład	Wynik
`Alias ID_N as NameID;`	
`Alias A as Name, B as Number, C as Date;`	Zmiany nazw zdefiniowane tą instrukcją są używane we wszystkich dalszych instrukcjach **SELECT** i **LOAD**. W dowolnym dalszym miejscu skryptu można też zdefiniować nowy alias nazwy pola, podając nową instrukcję **alias**.

AutoNumber

Ta instrukcja zwraca niepowtarzalną wartość całkowitą dla każdej odrębnie przetwarzanej wartości w polu napotkanym podczas wykonywania skryptu.

Gdy funkcja *autonumber (page 562)* jest używana w instrukcji **LOAD**, ale w tym przypadku obowiązują pewne ograniczenia dotyczące używania ładowania zoptymalizowanego. Ładowanie zoptymalizowane można utworzyć, najpierw ładując dane z pliku **QVD**, a następnie używając instrukcji **AutoNumber** celem przekształcenia wartości na klucze symboli.

Składnia:

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

Argumenty:

Argumenty

Argument	Opis
*fieldlist	Rozdzielana przecinkami lista pól, w których wartości powinny być zastąpione unikatową wartością liczby całkowitej. Aby uwzględnić wszystkie pola o zgodnych nazwach, można użyć symbolu wieloznacznego ? lub * w nazwach pól. W celu uwzględnienia wszystkich pól można użyć znaku *. Jeśli używane są symbole wieloznaczne, wówczas nazwy pól należy umieszczać w cudzysłowach.
namespace	**Używanie argumentu** `namespace` jest opcjonalne. Tej opcji można używać, gdy wymagane jest utworzenie przestrzeni nazw, w której identyczne wartości w różnych polach będą miały ten sam klucz. Jeśli nie użyjesz tej opcji, wszystkie pola będą miały osobny indeks klucza.

Ograniczenia:

Gdy w jednym skrypcie istnieje kilka instrukcji **LOAD**, wówczas instrukcję **AutoNumber** należy umieścić po końcowej instrukcji **LOAD**.

Przykład – skrypt z instrukcją AutoNumber

Przykład skryptu

W tym przykładzie dane są najpierw ładowane bez instrukcji **AutoNumber**. Następnie dodawana jest instrukcja **AutoNumber**, aby pokazać efekt.

Dane zastosowane w tym przykładzie

Załaduj następujące dane w edytorze ładowania danych jako ładowanie wbudowane, aby utworzyć poniższy przykład skryptu. Na razie zostaw instrukcję **AutoNumber** w komentarzu.

```
RegionSales: LOAD *,  Region &'|'& Year &'|'& Month as KeyToOtherTable INLINE [ Region, Year,
Month, Sales North,     2014,   May,    245 North,      2014,   May,    347 North,      2014,   June,   127 S
June,   645 South,      2013,   May,    367 South,      2013,   May,    221 ];
&'|'& 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;
```

Tworzenie wizualizacji

Utwórz dwie wizualizacje tabel w arkuszu Qlik Sense. Dodaj **KeyToOtherTable**, **Region**, **Year**, **Month** i **Sales** jako wymiary pierwszej tabeli. Dodaj **KeyToOtherTable**, **Region**, **Year**, **Month** i **Budget** jako wymiary drugiej tabeli.

Wynik

Tabela RegionSales

KeyToOtherTable	Region	Year	Month	Sales
North|2014|June	North	2014	June	127
North|2014|May	North	2014	May	245
North|2014|May	North	2014	May	347
South|2013|May	South	2013	May	221
South|2013|May	South	2013	May	367
South|2014|June	South	2014	June	645

Tabela Budget

KeyToOtherTable	Region	Year	Month	Budget
North|2014|June	North	2014	June	150
North|2014|May	North	2014	May	200
North|2014|May	North	2014	May	350
South|2013|May	South	2013	May	200
South|2013|May	South	2013	May	300
South|2014|June	South	2014	June	500

Objaśnienie

W przykładzie pokazano pole złożone **KeyToOtherTable**, które łączy obie tabele. Instrukcja **AutoNumber** nie jest używana. Zwróć uwagę na długość wartości pola **KeyToOtherTable**.

Dodawanie instrukcji AutoNumber

Usuń oznaczenie komentarza instrukcji **AutoNumber** w skrypcie ładowania.

```
AutoNumber KeyToOtherTable;
```

Wynik

Tabela RegionSales

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

Tabela Budget

KeyToOtherTable	Region	Year	Month	Budget
1	North	2014	June	150
1	North	2014	May	200
2	North	2014	May	350
3	South	2013	May	200
4	South	2013	May	300
4	South	2014	June	500

Objaśnienie

Wartości pola **KeyToOtherTable** zostały zastąpione unikatowymi wartościami liczb całkowitych, w wyniku czego zmniejszono długość wartości pól, oszczędzając w ten sposób pamięć. Instrukcja **AutoNumber** wpływa na pola kluczowe w obu tabelach, a tabele pozostają połączone. Na potrzeby prezentacji przykład jest krótki, ale funkcja ta sprawdza się szczególnie w przypadku tabel zawierających wiele wierszy.

Binary

Instrukcja **binary** służy do ładowania danych z dokumentu z innej aplikacji Qlik Sense lub QlikView, w tym danych dostępu do sekcji. Inne elementy aplikacji nie są uwzględnione, na przykład arkusze, narracje, wizualizacje, elementy główne ani zmienne.

W skrypcie może występować tylko jedna instrukcja **binary**. Instrukcja **binary** musi być pierwszą instrukcją skryptu, mimo że na początku skryptu zazwyczaj umieszczane są instrukcje SET.

Składnia:

```
binary [path] filename
```

Argumenty:

Argumenty

Argument	Opis
path	Ścieżka do pliku powinna być odniesieniem do połączenia do danych folderu. Jest to wymagane, jeśli plik nie został załadowany do katalogu roboczego programu Qlik Sense. **Przykład:** *'lib://Table Files/'* W dotychczasowym trybie tworzenia skryptów obsługiwane są również następujące formaty ścieżek: • bezwzględna **Przykład:** *c:**data*\ • względna wobec aplikacji zawierającej tę linię skryptu. **Przykład:** *data*\
filename	Nazwa pliku z uwzględnieniem rozszerzenia .qvw lub .qvf.

Ograniczenia:

Instrukcji **binary** nie można używać w celu ładowania danych z aplikacji w tym samym wdrożeniu programu Qlik Sense Enterprise, tworząc odwołanie do identyfikatora aplikacji. Dane można ładować tylko z pliku *.qvf*.

Przykłady

Ciąg znaków	Opis
`Binary lib://DataFolder/customer.qvw;`	W tym przykładzie plik musi znajdować się w połączeniu danych **Folder**. Może to być na przykład folder, który administrator tworzy na serwerze Qlik Sense. Kiknij **Utwórz nowe połączenie** w edytorze ładowania danych a następnie w opcji **Lokalizacje plików** wybierz **Folder**.
`Binary customer.qvf;`	W tym przykładzie plik musi być zlokalizowany w katalogu roboczym Qlik Sense.
`Binary c:\qv\customer.qvw;`	Ten przykład z użyciem bezwzględnej ścieżki do pliku zadziała tylko w dotychczasowym trybie tworzenia skryptów.

Comment field

Udostępnia sposób wyświetlania komentarzy do pola (metadanych) z baz danych i arkuszy kalkulacyjnych. Nazwy pól nieobecne w aplikacji są ignorowane. W przypadku znalezienia wielu wystąpień nazwy pola używana jest ostatnia wartość.

Składnia:

```
comment [fields] *fieldlist using mapname
comment [field] fieldname with comment
```

Używana tabela mapowania powinna mieć dwie kolumny – pierwsza powinna zawierać nazwy pól, a druga komentarze.

Argumenty:

Argumenty

Argument	Opis
**fieldlist*	Rozdzielana przecinkami lista pól przeznaczonych do opatrzenia komentarzem. Użycie znaku * jako listy pól oznacza wszystkie pola. W nazwach pól dozwolone jest korzystanie z symboli wieloznacznych * i ?. Cytowanie nazw pól może być konieczne, gdy używane są symbole wieloznaczne.
mapname	Nazwa tabeli mapowania, która została poprzednio odczytana w instrukcji **LOAD** lub **SELECT** mapowania.
fieldname	Nazwa pola przeznaczonego do opatrzenia komentarzem.
comment	Komentarz, jaki powinien zostać dodany do pola.

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

Udostępnia sposób wyświetlania komentarzy do tabeli (metadanych) z baz danych lub arkuszy kalkulacyjnych.

Nazwy tabel nieobecne w aplikacji są ignorowane. W przypadku znalezienia wielu wystąpień nazwy tabeli używana jest ostatnia wartość. To słowo kluczowe może być używane w celu odczytywania komentarzy ze źródła danych.

Składnia:

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

Argumenty:

Argumenty

Argument	Opis
tablelist	(table{,table})
mapname	Nazwa tabeli mapowania, która została poprzednio odczytana w instrukcji **LOAD** lub **SELECT** mapowania.
tablename	Nazwa tabeli przeznaczonej do opatrzenia komentarzem.
comment	Komentarz, jaki powinien zostać dodany do tabeli.

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

Instrukcja **CONNECT** służy do określania dostępu aplikacji Qlik Sense do ogólnej bazy danych przy użyciu interfejsu OLE DB/ODBC. W przypadku ODBC źródło danych najpierw należy określić za pomocą narzędzia administracyjnego ODBC.

Ta funkcjonalność jest niedostępna w środowisku Qlik Sense SaaS.

Ta instrukcja obsługuje tylko powiązania do danych w folderze w trybie standardowym.

Składnia:

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

Argumenty:

Argumenty

Argument	Opis

connect-string	`connect-string ::= datasourcename { ; conn-spec-item }`
	Ciąg połączenia jest nazwą bazy danych oraz opcjonalną listą co najmniej jednego elementu specyfikacji połączenia. Jeśli nazwa źródła danych zawiera spacje albo jeśli podana zostanie lista elementów specyfikacji połączenia, wówczas ciąg połączenia należy ująć w cudzysłowy.
	
	datasourcename musi być zdefiniowanym źródłem danych ODBC albo ciągiem znaków, który definiuje dostawcę OLE DB.
	
	`conn-spec-item ::=`**`DBQ=`**`database_specifier |`**`DriverID=`**`driver_specifier |`**`UID=`**`userid |`**`PWD=`**`password`
	
	W przypadku różnych baz danych możliwe elementy specyfikacji połączenia mogą się różnić. W przypadku niektórych baz danych możliwe są także elementy inne niż określone powyżej. W przypadku bazy danych OLE DB niektóre elementy dotyczące połączenia są obowiązkowe, nie opcjonalne.

connection	Nazwa połączenia do danych zapisywana w edytorze ładowania danych.

Jeśli źródło **ODBC** zostanie umieszczone przed instrukcją **CONNECT**, wówczas będzie używany interfejs ODBC; w przeciwnym wypadku będzie używany interfejs OLE DB.

Za pomocą instrukcji **LIB CONNECT TO** nawiązuje połączenie z bazą danych przy użyciu zapisanego połączenia danych utworzonego w edytorze ładowania danych.

Example 1:

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

Źródło danych zdefiniowane przez tę instrukcję jest używane przez kolejne instrukcje **Select (SQL)**, aż do napotkania nowej instrukcji **CONNECT**.

Example 2:

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

Connect32

Ta instrukcja jest używana w taki sam sposób, jak instrukcja **CONNECT**, ale zmusza system 64-bitowy do użycia 32-bitowego dostawcy ODBC/OLE DB. Nie dotyczy to niestandardowej instrukcji connect.

Connect64

Ta instrukcja jest używana w taki sam sposób, jak instrukcja **CONNECT**, ale wymusza użycie 64-bitowego dostawcy. Nie dotyczy to niestandardowej instrukcji connect.

Declare

Instrukcja **Declare** służy do tworzenia definicji pól, w których można określić relacje między polami a funkcjami. Definicje zestawów pól można wykorzystać do automatycznego generowania pochodnych pól, których można użyć jako wymiarów. Można na przykład utworzyć definicję kalendarza i użyć jej do wygenerowania z pola daty powiązanych wymiarów takich jak rok, miesiąc, tydzień i dzień.

Instrukcji **Declare** można używać albo do skonfigurowania nowej definicji pola, albo do utworzenia definicji pola na podstawie już istniejącej definicji.

Konfigurowanie nowej definicji pola

Składnia:

```
definition_name:
Declare [Field[s]] Definition [Tagged tag_list ]
[Parameters parameter_list ]
Fields field_list
```

Argumenty:

Argument	Opis
definition_name	Nazwa definicji pola, zakończona dwukropkiem.  *Jako nazwy definicji pola nie należy używać autoCalendar, gdyż jest to nazwa zarezerwowana dla automatycznie generowanych szablonów kalendarza.* **Przykład:** `Calendar:`
tag_list	Rozdzielana przecinkami lista znaczników stosowanych do pól pochodnych tworzonych na podstawie tej definicji pola. Stosowanie znaczników nie jest obowiązkowe, ale jeśli nie zostaną podane znaczniki określające kolejność sortowania, na przykład $date, $numeric lub $text, pole pochodne będzie domyślnie sortowane według kolejności ładowania. **Przykład:** `'$date'Thank you for bringing this to our attention, and apologies for the inconvenience.`
parameter_list	Rozdzielana przecinkami lista parametrów. Parametr ma definicję w postaci `name=value` i otrzymuje wartość początkową, którą można zastąpić przy ponownym wykorzystaniu definicji pola. Opcjonalne. **Przykład:** `first_month_of_year = 1`
field_list	Rozdzielana przecinkami lista pól generowanych, gdy używana jest definicja pola. Definicja pola ma postać `<expression> As field_name tagged tag`. Symbol `$1` umożliwia odwoływanie się do pola danych, z którego mają być generowane pola pochodne. **Przykład:** `Year($1) As Year tagged ('$numeric')`

Przykład:

```
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')
;
```

Kalendarz został zdefiniowany. Teraz można stosować go do załadowanych pól daty, w tym przypadku OrderDate i ShippingDate, używając klauzuli **Derive**.

Ponowne używanie istniejącej definicji pola

Składnia:

```
<definition name>:
Declare [Field][s] Definition
Using <existing_definition>
[With <parameter_assignment> ]
```

Argumenty:

Argument	Opis
definition_ name	Nazwa definicji pola, zakończona dwukropkiem. **Przykład:** `MyCalendar:`
existing_ definition	Definicja pola, która zostanie ponownie użyta podczas tworzenia nowej definicji pola. Nowa definicja pola będzie działać analogicznie do definicji, na której jest oparta, chyba że zostanie podany argument parameter_assignment zmieniający wartość używaną w wyrażeniach pola. **Przykład:** `Using Calendar`
parameter_ assignment	Rozdzielana przecinkami lista przypisań parametrów. Przypisanie parametru ma postać `name=value` i powoduje zastąpienie wartości parametru ustawionej w bazowej definicji pola. Opcjonalne. **Przykład:** `first_month_of_year = 4`

Przykład:

W tym przykładzie ponownie użyjemy definicji kalendarza utworzonej w poprzednim przykładzie. W tym przypadku chcemy używać roku obrachunkowego zaczynającego się w kwietniu. Można to osiągnąć, przypisując parametrowi first_month_of_year wartość 4, co wpłynie na wartość zdefiniowanego pola

DayNumberOfYear.

Posłużymy się tutaj danymi i definicją pola z poprzedniego przykładu.

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

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

Po przeładowaniu skryptu ładowania danych pola wygenerowane będą dostępne w edytorze arkusza pod nazwami OrderDate.MyCalendar.* i ShippingDate.MyCalendar.*.

Derive

Instrukcja **Derive** służy do generowania pochodnych pól na podstawie definicji pola utworzonej przy użyciu instrukcji **Declare**. Można określić, dla których pól danych mają być tworzone pochodne pola, lub tworzyć je jawnie lub niejawnie na podstawie znaczników pól.

Składnia:

```
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
```

Argumenty:

Argumenty

Argument	Opis
definition	Nazwa definicji pola używana podczas wyznaczania pól pochodnych. **Przykład:** `Calendar`
field_list	Rozdzielana przecinkami lista pól danych, z których powinny zostać wygenerowane pola pochodne – na podstawie definicji pól. Pola danych powinny być polami, które zostały już załadowane w skrypcie. **Przykład:** `OrderDate, ShippingDate`
tag_list	Rozdzielana przecinkami lista znaczników. Pola pochodne zostaną wygenerowane dla wszystkich pól danych, które zawierają dowolne znaczniki z listy. Lista znaczników powinna być otoczona nawiasami okrągłymi. **Przykład:** `('$date','$timestamp')`

Przykłady:

- Wyznaczanie pól pochodnych dla konkretnych pól danych.
  W tym przypadku podajemy pola OrderDate i ShippingDate.
  `DERIVE FIELDS FROM FIELDS OrderDate,ShippingDate USING Calendar;`
- Wyznaczanie pól pochodnych dla wszystkich pól z konkretnym znacznikiem.

W tym przypadku wyznaczamy pola pochodne na podstawie pola Calendar dla wszystkich pól ze znacznikiem `$date`.
`DERIVE FIELDS FROM EXPLICIT TAGS ('$date') USING Calendar;`

- Wyznaczanie pól pochodnych dla wszystkich pól ze znacznikiem definicji pola.
  W tym przypadku wyznaczamy pola pochodne dla wszystkich pól danych z takim samym znacznikiem, jak definicja pola Calendar – tym przypadku jest to znacznik `$date`.
  `DERIVE FIELDS FROM IMPLICIT TAG USING Calendar;`

Directory

Instrukcja **Directory** określa, w którym katalogu należy szukać plików danych w kolejnych instrukcjach **LOAD**, do momentu wydania nowej instrukcji **Directory**.

Składnia:

```
Directory[path]
```

Jeśli instrukcja **Directory** jest wydana bez parametru **path** lub pozostawiona sama sobie, aplikacja Qlik Sense będzie prowadzić wyszukiwanie w katalogu roboczym Qlik Sense.

Argumenty:

Argumenty

Argument	Opis
path	Tekst, który może być interpretowany jako ścieżka do pliku data. Ścieżka jest ścieżką do pliku. Dostępne są następujące opcje: • bezwzględna **Przykład:** *c:\data* • względna wobec katalogu roboczego aplikacji Qlik Sense. **Przykład:** *data* • Adres URL (HTTP lub FTP), wskazujący lokalizację w Internecie lub intranecie. **Przykład:** *http://www.qlik.com*

Przykłady:

```
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

Instrukcja **Disconnect** przerywa bieżące połączenieODBC/OLE DB/Niestandardowe. Ta instrukcja jest opcjonalna.

Składnia:

```
Disconnect
```

Połączenie zostanie automatycznie przerwane w przypadku wykonania nowej instrukcji **connect** lub po zakończeniu wykonywania skryptu.

Przykład:

```
Disconnect;
```

Drop

Słowo kluczowe skryptu **Drop** może być używane do upuszczania tabel lub pól z bazy danych.

Drop field

W dowolnym momencie podczas wykonywania skryptu można usunąć z modelu danych, a zarazem z pamięci, pola aplikacji Qlik Sense, korzystając w tym celu z instrukcji **drop field**.

> *Dozwolone są obydwie formy **drop field** i **drop fields**, a użycie każdej z nich zapewnia ten sam efekt. Jeśli żadna tabela nie zostanie określona, pole zostanie usunięte z wszystkich tabel, w których występuje.*

Składnia:

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

Przykłady:

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

Drop table

W dowolnym momencie podczas wykonywania skryptu można usunąć z modelu danych, a zarazem z pamięci, wewnętrzne tabele aplikacji Qlik Sense, korzystając w tym celu z instrukcji **drop table**.

Składnia:

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

*Akceptowane są zarówno formularze **drop table**, jak i **drop tables**.*

W rezultacie tej operacji zostaną utracone następujące elementy:

- Rzeczywiste tabele.
- Wszystkie pola, które nie stanowią części tabel pozostałych.
- Wartości pola w polach pozostałych, które pochodzą bezpośrednio z usuniętych tabel.

Przykłady i wyniki:

Przykład	Wynik
`drop table Orders, Salesmen, T456a;`	Ta linia spowoduje usunięcie z pamięci trzech tabel.

`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;`
	Po utworzeniu tabeli *Tab2* tabela *Tab1* jest usuwana.

Drop table

W dowolnym momencie podczas wykonywania skryptu można usunąć z modelu danych, a zarazem z pamięci, wewnętrzne tabele aplikacji Qlik Sense, korzystając w tym celu z instrukcji **drop table**.

Składnia:

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

*Akceptowane są zarówno formularze **drop table**, jak i **drop tables**.*

W rezultacie tej operacji zostaną utracone następujące elementy:

- Rzeczywiste tabele.
- Wszystkie pola, które nie stanowią części tabel pozostałych.
- Wartości pola w polach pozostałych, które pochodzą bezpośrednio z usuniętych tabel.

Przykłady i wyniki:

Przykład	Wynik
`drop table Orders, Salesmen, T456a;`	Ta linia spowoduje usunięcie z pamięci trzech tabel.

`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;`
	Po utworzeniu tabeli *Tab2* tabela *Tab1* jest usuwana.

Execute

Instrukcja **Execute** służy do uruchamiania innych programów w czasie, gdy aplikacja Qlik Sense ładuje dane. Na przykład w celu wykonania niezbędnych przekształceń.

Ta funkcjonalność jest niedostępna w środowisku Qlik Sense SaaS.

Ta instrukcja nie jest obsługiwana w trybie standardowym.

Składnia:

```
execute commandline
```

Argumenty:

Argumenty

Argument	Opis
commandline	Tekst, który może zostać zinterpretowany przez system operacyjny jako linia poleceń. Możesz utworzyć odwołanie do bezwzględnej ścieżki do pliku albo do ścieżki lib:// folderu.

Jeśli wymagane jest użycie instrukcji **Execute**, wówczas muszą być spełnione następujące warunki:

- Należy uruchomić program w trybie zgodności (dotyczy programu Qlik Sense oraz programu Qlik Sense Desktop).
- Dla parametru OverrideScriptSecurity należy ustawić wartość 1 w pliku *Settings.ini* (dotyczy programu Qlik Sense).
  Plik *Settings.ini* znajduje się w folderze *C:\ProgramData\Qlik\Sense\Engine* i zwykle jest plikiem pustym.

*Jeśli ustawisz parametr OverrideScriptSecurity w celu włączenia instrukcji **Execute**, wówczas każdy użytkownik będzie mógł uruchamiać pliki na serwerze. Na przykład użytkownik może dołączyć plik wykonywalny do aplikacji, a następnie wykonać w plik w skrypcie ładowania danych.*

Wykonaj następujące czynności:

1. Utwórz kopię pliku *Settings.ini* i otwórz ją w edytorze tekstu.
2. Upewnij się, że pierwszy wiersz pliku zawiera *[Settings 7]*.
3. Wstaw nowy wiersz i napisz *OverrideScriptSecurity=1*.
4. Wstaw pusty wiersz na końcu pliku.
5. Zapisz plik.
6. Zastąp plik *Settings.ini* plikiem, który właśnie został zmieniony.
7. Ponownie uruchom program Qlik Sense Engine Service (QES).

Jeśli program Qlik Sense działa jako usługa, wówczas niektóre polecenia mogą działać niezgodnie z oczekiwaniami.

Przykład:

```
Execute C:\Program Files\Office12\Excel.exe;

Execute lib://win\notepad.exe // win is a folder connection referring to c:\windows
```

Field/Fields

Słowa kluczowe skryptu **Field** i **Fields** są używane w instrukcjach **Declare**, **Derive**, **Drop**, **Comment**, **Rename** i **Tag/Untag**.

FlushLog

Instrukcja **FlushLog** wymusza w aplikacji Qlik Sense zapisywanie zawartości bufora skryptu do pliku dziennika skryptu.

Składnia:

```
FlushLog
```

Zawartość tego buforu jest zapisywana do pliku dziennika. To polecenie może być użyteczne na potrzeby debugowania, ponieważ umożliwia uzyskiwanie danych, które w przeciwnym wypadku mogły zostać utracone w przypadku nieudanego wykonania skryptu.

Przykład:

```
FlushLog;
```

Force

Instrukcja **force** wymusza w aplikacji Qlik Sense interpretację nazw i wartości pól kolejnych instrukcji **LOAD** i **SELECT** jako pisanych tylko wielkimi literami, tylko małymi literami, jak nazwy własne lub tak, jak są wyświetlane (mieszane). Ta instrukcja umożliwia powiązanie wartości pól z tabeli utworzonych zgodnie z różnymi konwencjami.

Składnia:

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

Jeśli nie określono żadnych wartości, przyjmuje się wymuszanie mieszanej wielkości liter. Instrukcja Force obowiązuje do nowej instrukcji Force.

Instrukcja **force** nie działa w sekcji dostępu; wielkość liter nie ma znaczenia w żadnych ładowanych wartościach pól.

Przykłady i wyniki

Przykład	Wynik

Na tym przykładzie pokazano, jak wymuszać pisownię z wielkiej litery.
`FORCE Capitalization;`
`Capitalization:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	Tabela **Capitalization** zawiera następujące wartości:
	`Ab`
	`Cd`
	`Ef`
	`Gh`
	Wszystkie wartości są pisane z wielkiej litery.

W tym przykładzie pokazano, jak wymuszać pisownię wielkimi literami.
`FORCE Case Upper;`
`CaseUpper:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	Tabela **CaseUpper** zawiera następujące wartości:
	`AB`
	`CD`
	`EF`
	`GH`
	Wszystkie wartości są pisane wielkimi literami.

Przykład	Wynik

W tym przykładzie pokazano, jak wymuszać pisownię małymi literami.
 	
`FORCE Case Lower;`
`CaseLower:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	Tabela **CaseLower** zawiera następujące wartości:
	
	`ab`
	`cd`
	`ef`
	`gh`
	Wszystkie wartości są pisane małymi literami.

W tym przykładzie pokazano, jak wymuszać pisownię literami mieszanej wielkości.
 	
`FORCE Case Mixed;`
`CaseMixed:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	Tabela **CaseMixed** zawiera następujące wartości:
	
	`ab`
	`Cd`
	`eF`
	`GH`
	Wszystkie te wartości są pisane tak, jak są wyświetlane w skrypcie.

Zob. także:

From

Słowo kluczowe skryptu **From** jest używane w instrukcjach **Load** jako odwołanie do pliku, a w instrukcjach **Select** jako odwołanie do tabeli lub widoku bazy danych.

Load

Instrukcja **LOAD** ładuje pola z pliku, z danych zdefiniowanych w skrypcie, z wcześniej załadowanej tabeli, ze strony internetowej, z wyniku późniejszej instrukcji **SELECT** lub przez automatyczne wygenerowanie danych. Możliwe jest również ładowanie danych z połączeń analitycznych.

Składnia:

```
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 ]
```

Argumenty:

Argumenty

Argument	Opisu
distinct	Instrukcja **distinct** może być używana jako predykat dotyczący ładowania tylko unikatowych rekordów. Jeśli istnieją zduplikowane rekordy, wówczas załadowane zostanie tylko pierwsze wystąpienie. Jeśli używane są poprzedzające polecenia Load, wówczas polecenie **distinct** należy umieścić w pierwszej instrukcji Load, ponieważ **distinct** wpływa tylko na tabelę docelową.

Argument	Opisu
fieldlist	*fieldlist ::= (* | field {, * | field })* Lista ładowanych pól. Użycie znaku * jako listy pól oznacza wszystkie pola w tabeli. *field ::= (fieldref | expression)* [**as** *aliasname]* Definicja pola musi zawsze zawierać literał, odniesienie do istniejącego pola lub wyrażenie. *fieldref ::= (fieldname |@fieldnumber |@startpos:endpos [* **I** *|* **U** *|* **R** *|* **B** *|* **T**]) *fieldname* jest tekstem identycznym z nazwą pola w tabeli. Jeśli nazwa pola zawiera np. spacje, musi być ujęta w proste podwójne cudzysłowy lub nawiasy kwadratowe. Niekiedy nie są dostępne jawne nazwy pól. Wtedy należy użyć innego zapisu: *@fieldnumber* reprezentuje numer pola w rozdzielanym pliku tabeli. Musi to być dodatnia liczba całkowita poprzedzona znakiem „@”. Numeracja zawsze zaczyna się od 1, a kończy na liczbie pól. *@startpos*:*endpos* reprezentuje pozycję początkową i końcową pola w pliku z rekordami o stałej długości. Obie pozycje muszą być dodatnimi liczbami całkowitymi. Liczby muszą być poprzedzone znakiem „@” i rozdzielone dwukropkiem. Numeracja zawsze zaczyna się od 1, a kończy na liczbie pozycji. W ostatnim polu **n** jest używane jako pozycja końcowa. • Jeśli bezpośrednio po zapisie *@startpos*:*endpos* następuje znak **I** lub **U**, odczytane bajty będą interpretowane jako dane binarne zawierające liczby całkowite, odpowiednio ze znakiem (**I**) lub bez znaku (**U**) (kolejność bajtów Intel). Liczba odczytanych pozycji musi wynosić 1, 2 lub 4. • Jeśli bezpośrednio po zapisie *@startpos*:*endpos* następuje znak **R**, odczytane bajty będą interpretowane jako dane binarne zawierające liczby rzeczywiste (32-bitowe lub 64-bitowe wartości zmiennopozycyjne IEEE). Liczba odczytanych pozycji musi wynosić 4 lub 8. • Jeśli bezpośrednio po zapisie *@startpos*:*endpos* następuje znak **B**, odczytane bajty będą interpretowane jako wartości BCD (Binary Coded Decimal) według standardu COMP-3. Można podać dowolną liczbę bajtów. *expression* może być funkcją liczbową lub znakową operującą na polach tej samej tabeli. Więcej informacji na ten temat zawiera sekcja na temat składni wyrażeń. Klauzula **as** służy do przypisania polu nowej nazwy.

Argument	Opisu
from	**from** jest używane, jeśli dane powinny być ładowane z pliku przy użyciu połączenia do danych pliku webowego. *file ::= [path] filename* **Przykład:** ***'lib://Table Files/'*** Pominięcie ścieżki spowoduje, że aplikacja Qlik Sense będzie wyszukiwać plik w katalogu wskazanym instrukcją **Directory**. W przypadku braku instrukcji **Directory** aplikacja Qlik Sense prowadzi wyszukiwanie w katalogu roboczym, którym jest *C:\Users\{user}\Documents\Qlik\Sense\Apps*. *W instalacji serwera Qlik Sense katalog roboczy jest określony w usłudze Qlik Sense Repository Service. Domyślnie jest to C:\ProgramData\Qlik\Sense\Apps.* *filename* może zawierać standardowe symbole wieloznaczne DOS, czyli * i ?. Ich użycie spowoduje załadowanie wszystkich pasujących plików w określonym katalogu. *format-spec ::= (fspec-item { , fspec-item })* Na specyfikację formatu składają się pozycje specyfikacji formatu podane w nawiasach jako lista. **Dotychczasowy tryb tworzenia skryptów** W dotychczasowym trybie tworzenia skryptów obsługiwane są również następujące formaty ścieżek: • bezwzględna **Przykład:** ***c:\data*** • względna wobec katalogu roboczego aplikacji Qlik Sense. **Przykład:** ***data*** • Adres URL (HTTP lub FTP), wskazujący lokalizację w Internecie lub intranecie. **Przykład:** ***http://www.qlik.com***

Argument	Opisu
from_field	Klauzula **from_field** umożliwia załadowanie danych z wcześniej załadowanego pola. *fieldassource::=*(*tablename, fieldname)* Wskazywana jest nazwa pola odpowiadająca nazwie tabeli *tablename* i nazwie pola *fieldname*, która została poprzednio załadowana. *format-spec ::= (fspec-item {, fspec-item })* Na specyfikację formatu składają się pozycje specyfikacji formatu podane w nawiasach jako lista.
inline	Klauzula **inline** wskazuje dane wpisywane bezpośrednio w skrypcie, a nie ładowane z pliku. *data ::= [text]* Dane wprowadzane w ramach klauzuli **inline** muszą być ujęte w podwójne cudzysłowy lub w nawiasy kwadratowe. Tekst umieszczony między tymi znakami zostanie zinterpretowany tak samo, jak zawartość pliku. Oznacza to między innymi, że w tekście klauzuli **inline** należy wstawiać nowe wiersze w tych samych miejscach, w których występowałyby w pliku tekstowym, naciskając klawisz Enter podczas wpisywania skryptu. Liczba kolumn jest zdefiniowana w pierwszej linii. *format-spec ::= (fspec-item {, fspec-item })* Na specyfikację formatu składają się pozycje specyfikacji formatu podane w nawiasach jako lista.
resident	Klauzula **resident** umożliwia załadowanie danych z wcześniej załadowanej tabeli. *table label* to etykieta poprzedzająca instrukcje **LOAD** lub **SELECT** tworzące pierwotną tabelę. Na końcu etykiety musi się znajdować dwukropek.
autogenerate	Klauzula **autogenerate** jest używana w przypadku danych, które mają być automatycznie generowane przez aplikację Qlik Sense. *size ::= number* *Number* to liczba całkowita wskazująca liczbę generowanych rekordów. Lista pól nie może zawierać wyrażeń, które wymagają danych z zewnętrznego źródła danych ani wcześniej załadowanej tabeli, chyba że przy użyciu funkcji **Peek** utworzono odwołanie do wartości pojedynczego pola we wcześniej załadowanej tabeli.

Argument	Opisu
extension	Dane można ładować z połączeń analitycznych. Należy użyć klauzuli **extension**, aby wywołać funkcję zdefiniowaną we wtyczce rozszerzeń po stronie serwera (SSE) albo dokonać ewaluacji skryptu. Do wtyczki SSE można wysłać pojedynczą tabelę i zostanie zwrócona pojedyncza tabela danych. Jeśli wtyczka nie określa nazw pól, które są zwracane, wówczas pola otrzymają nazwy Field1, Field2 itd. `Extension pluginname.functionname( tabledescription );` • Ładowanie danych za pomocą funkcji we wtyczce SSE *tabledescription ::= (table { ,tablefield})* Jeśli pola tabeli nie zostaną określone, wówczas pola będą używane w kolejności ładowania. • Ładowanie danych poprzez ocenę skryptu we wtyczce SSE *tabledescription ::= (script, table { ,tablefield})* **Sposób obsługi typów danych w definicji pola tabeli** Typy danych są automatycznie wykrywane w połączeniach analitycznych. Jeśli dane nie zawierają żadnych wartości liczbowych, a zawierają co najmniej jeden ciąg znaków inny niż NULL, wówczas takie pole jest traktowane jako tekstowe. W każdym innym przypadku jest traktowane jako liczbowe. Typ danych można wymusić, umieszczając nazwę pola w funkcji **String()** lub **Mixed()**. • Funkcja **String()** wymusza traktowanie pola jako tekstowego. Jeśli pole jest liczbowe, wówczas wyodrębniana jest część tekstowa wartości podwójnej – nie jest wykonywane żadne przekształcenie. • Funkcja **Mixed()** wymusza traktowanie pola jako wartości podwójnej. Funkcje **String()** i **Mixed()** nie mogą być używane poza definicjami pola tabeli w klauzuli **extension**, a ponadto w definicji pola tabeli nie można używać innych funkcji Qlik Sense. **Więcej informacji o połączeniach analitycznych** Zanim możliwe będzie używanie połączeń analitycznych, należy je skonfigurować.
where	**where** to klauzula określająca, czy rekord ma być uwzględniony w selekcji, czy też nie. Selekcja jest uwzględniana, jeśli warunek *criterion* ma wartość True. *criterion* jest wyrażeniem logicznym.

Argument	Opisu
while	**while** to klauzula określająca, czy rekord ma być wielokrotnie odczytywany. Ten sam rekord będzie odczytywany, dopóki warunek *criterion* ma wartość True. W praktyce korzystanie z klauzuli **while** zwykle wymaga również użycia funkcji **IterNo( )**. *criterion* jest wyrażeniem logicznym.
group by	Klauzula **group by** służy do określania pól, według których dane mają być agregowane (grupowane). Pola agregujące powinny być w jakiś sposób dołączone w ładowanych wyrażeniach. Poza obrębem funkcji agregacji w ładowanych wyrażeniach wolno używać wyłącznie pól agregujących. *groupbyfieldlist ::= (fieldname { ,fieldname })*
order by	Klauzula **order by** służy do sortowania rekordów tabeli rezydentnej przed ich przetworzeniem przez instrukcję **load**. Tabelę rezydentną można sortować według dowolnej liczby pól w kolejności rosnącej lub malejącej. Sortowanie jest wykonywane w pierwszej kolejności według wartości liczbowej, a w drugiej kolejności według porządku leksykograficznego z uwzględnieniem znaków diakrytycznych. Klauzuli można używać tylko wtedy, gdy źródłem danych jest tabela rezydentna. Pola porządkujące określają, według którego pola należy sortować tabelę rezydentną. Pole można wskazać nazwą lub numerem w tabeli rezydentnej (pierwsze pole ma numer 1). orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } Kolejność *sortorder* może mieć wartość *asc* (rosnąco) lub *desc* (malejąco). Jeśli nie określono *sortorder*, przyjmuje się wartość *asc*. *fieldname*, *path*, *filename* i *aliasname* to ciągi tekstowe reprezentujące odpowiednio nazwę pola, ścieżkę, nazwę pliku i alias. Jako *fieldname* można podać dowolne pole z tabeli źródłowej. Pola utworzone klauzulą as (*aliasname*) są jednak poza zakresem i nie można ich używać w tej samej instrukcji **load**.

Jeśli nie zostanie wskazane żadne źródło danych klauzulą **from**, **inline**, **resident**, **from_field, extension** lub **autogenerate**, dane będą ładowane z wyniku instrukcji **SELECT** lub **LOAD** występującej bezpośrednio po bieżącej instrukcji. Następna instrukcja nie powinna mieć żadnego prefiksu.

Przykłady:

Ładowanie różnych formatów plików
Załaduj rozdzielony plik danych z opcjami domyślnymi:

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

Załaduj rozdzielony plik danych z połączenia z biblioteką (DataFiles):

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

Załaduj wszystkie rozdzielone pliki danych z połączenia z biblioteką (DataFiles):

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

Załaduj rozdzielony plik z przecinkiem jako znakiem rozdzielającym i z wbudowanymi etykietami:

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

Załaduj rozdzielony plik z tabulacją jako znakiem rozdzielającym i z wbudowanymi etykietami:

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

Załaduj plik dif z wbudowanymi nagłówkami:

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

Załaduj trzy pola ze stałego pliku rekordu bez nagłówków:

```
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);
```

Załaduj plik QVX, określając ścieżkę bezwzględną:

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

Ładowanie plików webowych
Załaduj z domyślnego adresu URL ustawionego w połączeniu do danych z plików webowych:

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

Załaduj z konkretnego adresu URL i zastąp adres URL ustawiony w połączeniu do danych z plików webowych:

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

Załaduj z konkretnego adresu URL ustawionego w zmiennej, korzystając z rozszerzenia przez znak dolara:

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

Wybieranie tylko niektórych pól, zmiana nazw i obliczanie pól
Załaduj tylko trzy określone pola z rozdzielonego pliku:

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

Zmień nazwę pierwszego pola na A, a drugiego na B podczas ładowania pliku bez etykiet:

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

Załaduj Name jako konkatenację FirstName, znaku spacji i LastName:

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

Załaduj Quantity, Price i Value (wynik Quantity i Price):

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

Wybieranie tylko niektórych rekordów
Załaduj tylko unikatowe rekordy (powielone rekordy zostaną zignorowane):

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

Załaduj tylko te rekordy, w których pole Litres ma wartość powyżej zera:

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

Ładowanie danych nieznajdujących się w pliku i danych wygenerowanych automatycznie:
Załaduj tabelę z danymi wbudowanymi, dwoma polami o nazwach CatID i Category:

```
LOAD * Inline
[CatID, Category
0,Regular
1,Occasional
2,Permanent];
```

Załaduj tabelę z danymi wbudowanymi, trzema polami o nazwach UserID, Password i Access:

```
LOAD * Inline [UserID, Password, Access
A, ABC456, User
B, VIP789, Admin];
```

Załaduj tabelę z 10 000 wierszami. Pole A będzie zawierać numer odczytanego rekordu (1, 2, 3, 4, 5...), a pole B losową liczbę między 0 i 1:

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

Nawiasy po autogenerate są dozwolone, ale niewymagane.

Ładowanie danych z wcześniej załadowanej tabeli
Najpierw ładujemy rozdzielony plik tabeli i nadajemy mu nazwę tab1:

```
tab1:
SELECT A,B,C,D from 'lib://DataFiles/data1.csv';
```

Załaduj pola z wcześniej załadowanej tabeli tab1 jako tab2:

```
tab2:
LOAD A,B,month(C),A*B+D as E resident tab1;
```

Załaduj pola z wcześniej załadowanej tabeli tab1, ale tylko rekordy, w których wartość A jest większa niż B:

```
tab3:
LOAD A,A+B+C resident tab1 where A>B;
```

Załaduj pola z wcześniej załadowanej tabeli tab1 uporządkowane według A:

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

Załaduj pola z wcześniej załadowanej tabeli tab1, uporządkowane według pierwszego pola, a następnie drugiego pola:

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

Załaduj pola z wcześniej załadowanej tabeli tab1 uporządkowane według C malejąco, a następnie B rosnąco, po czym według pierwszego pola malejąco.

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

Ładowanie danych z wcześniej załadowanych pól
Załaduj Types pól z wcześniej załadowanej tabeli Characters jako A:

```
LOAD A from_field (Characters, Types);
```

Ładowanie danych z następnej tabeli (przed ładowaniem)
Załaduj A, B oraz obliczone pola X i Y z tabeli Table1 ładowanej w następnej instrukcji **SELECT**:

```
LOAD A, B, if(C>0,'positive','negative') as X, weekday(D) as Y;
SELECT A,B,C,D from Table1;
```

Grupowanie danych
Załaduj pola pogrupowane (zagregowane) według ArtNo:

```
LOAD ArtNo, round(Sum(TransAmount),0.05) as ArtNoTotal from table.csv group by ArtNo;
```

Załaduj pola pogrupowane (zagregowane) według Week i ArtNo:

```
LOAD Week, ArtNo, round(Avg(TransAmount),0.05) as WeekArtNoAverages from table.csv group by
Week, ArtNo;
```

Wielokrotne wczytywanie tego samego rekordu
W tym przykładzie mamy plik wejściowy Grades.csv zawierający oceny wszystkich uczniów zebrane w jednym polu.

```
Student,Grades
Mike,5234
John,3345
Pete,1234
Paul,3352
```

Oceny, w skali od 1 do 5, reprezentują następujące przedmioty: Math, English, Science i History. Możemy rozdzielić oceny w oddzielne wartości, kilkakrotnie odczytując każdy rekord z klauzulą **while** przy użyciu funkcji **IterNo( )** jako licznika. W każdym odczycie ocena jest wyodrębniana przy użyciu funkcji **Mid** i zapisywana w Grade, a temat jest wybierany za pomocą funkcji **pick** i zapisywany w Subject. Ostatnia klauzula **while** zawiera test celem sprawdzenia, czy wszystkie oceny zostały odczytane (w tym przypadku cztery na ucznia), co oznacza, że należy odczytać rekord następnego ucznia.

```
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));
```

W efekcie otrzymuje się tabelę zawierającą następujące dane:

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

Ładowanie z połączeń analitycznych
Używane są następujące dane przykładowe.

```
Values:
Load
 Rand() as A,
 Rand() as B,
 Rand() as C
AutoGenerate(50);
```

Ładowanie danych za pomocą funkcji

W tych przykładach zakładamy, że mamy wtyczkę połączeń analitycznych o nazwie *P*, która zawiera funkcję niestandardową *Calculate(Parameter1, Parameter2)*. Ta funkcja zwraca tabelę *Results*, która zawiera pola *Field1* i *Field2*.

```
Load * Extension P.Calculate( Values{A, C} );
```
Załaduj wszystkie pola, które są zwracane po wysłaniu pól A i C do funkcji.

```
Load Field1 Extension P.Calculate( Values{A, C} );
```
W przypadku wysłania pól A i C do funkcji załaduj tylko pole Field1.

```
Load * Extension P.Calculate( Values );
```
Załaduj wszystkie pola, które są zwracane po wysłaniu pól A i B do funkcji. Pola nie są określone, dlatego używane są pola A i B, ponieważ są pierwsze w kolejności w tabeli.

```
Load * Extension P.Calculate( Values {C, C});
```
Załaduj wszystkie pola, które są zwracane po wysłaniu pola C do obu parametrów funkcji.

```
Load * Extension P.Calculate( Values {String(A), Mixed(B)});
```

Załaduj wszystkie pola zwracane w przypadku wysłania do funkcji pola A, względem którego wymuszono traktowanie go jako ciągu znaków, oraz pola B, względem którego wymuszono traktowanie go jako liczbowego.

Ładowanie danych poprzez ocenę skryptu

```
Load A as A_echo, B as B_echo Extension R.ScriptEval( 'q;', Values{A, B} );
```
Załaduj tabelę zwracaną przez skrypt q w przypadku wysłania wartości A i B.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{A, B} );
```
Załaduj tabelę zwracaną przez skrypt zapisywany w zmiennej My_R_Script w przypadku wysłania wartości A i B.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{B as D, *} );
```
Załaduj tabelę zwracaną przez skrypt zapisywany w zmiennej My_R_Script w przypadku wysłania wartości B, której nazwę zmieniono na D, A i C. Użycie * powoduje wysłanie pozostałych pól, do których nie istnieją odwołania.

W rozszerzeniach plików powiązań DataFiles rozróżniana jest wielkość liter. Na przykład: .qvd.

Elementy specyfikacji formatu

Każdy element specyfikacji formatu określa pewną właściwość pliku tabeli:

fspec-item ::= [**ansi | oem | mac | UTF-8 | Unicode | txt | fix | dif | biff | ooxml | html | xml | kml | qvd | qvx | 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`]

Zestaw znaków

Zestaw znaków jest określnikiem pliku dla instrukcji **LOAD**, która definiuje zestaw znaków używany w pliku.

Specyfikatory **ansi**, **oem** i **mac** były używane w programie QlikView i nadal działają. Jednak nie są generowane podczas tworzenia instrukcji **LOAD** w programie Qlik Sense.

Składnia:
```
utf8 | unicode | ansi | oem | mac | codepage is
```

Argumenty:

Argumenty

Argument	Opis
utf8	Zestaw znaków UTF-8
unicode	Zestaw znaków Unicode

Argument	Opis
ansi	Windows, strona kodowa 1252
oem	DOS, OS/2, AS400 i inne
mac	Strona kodowa 10000
codepage is	Przy użyciu określnika **codepage** można używać dowolnej strony kodowej Windows jako *N*.

Ograniczenia:

Konwersja z zestawu znaków **oem** nie została wdrożona na potrzeby systemu MacOS. Jeśli żadna opcja nie zostanie określona, w systemie Windows zostanie przyjęta strona kodowa 1252.

Przykład:

```
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)
```

Zob. także:

p *Load (page 147)*

Format tabeli

Format tabeli jest określnikiem pliku dla instrukcji **LOAD**, która określa typ pliku. Jeśli nie określono inaczej, wówczas przyjmowane jest założenie, że plik jest w formacie *.txt*.

Typy formatu tabeli

Type	Opisu
txt	W rozdzielanym pliku tekstowym kolumny w tabeli są rozdzielane znakiem.
fix	W pliku o stałej długości rekordu każde pole zawiera określoną liczbę znaków. Zazwyczaj wiele plików o stałej długości rekordu zawiera rekordy rozdzielone znakiem nowego wiersza, ale istnieją bardziej zaawansowane opcje, aby określić wielkość rekordu w bajtach lub uwzględnić kilka wierszy przy użyciu parametru **Record is**. *Jeśli dane zawierają znaki wielobajtowe, podziały pól mogą być nierówne, ponieważ format opiera się na stałej długości w bajtach.*
dif	W pliku *.dif* (Data Interchange Format) używany jest specjalny format w celu definiowania tabeli.

Type	Opisu
biff	Aplikacja Qlik Sense może także interpretować dane w standardowych plikach Excel za pośrednictwem formatu *biff* (Binary Interchange File Format).
ooxml	W programie Excel 2007 i nowszych wersjach używany jest format ooxml *.xslx*.
html	Jeśli tabela jest częścią strony lub pliku html, wówczas powinien być używany format html.
xml	xml (Extensible Markup Language) to popularny język znaczników stosowany do przedstawiania struktur danych w formacie tekstowym.
qvd	Format *qvd* jest zastrzeżonym formatem plików QVD, eksportowanych z aplikacji Qlik Sense.
qvx	*qvx* jest formatem pliku/strumienia, który zapewnia wysoką wydajność dostarczania danych wyjściowych do aplikacji Qlik Sense.

Delimiter is

W przypadku rozdzielanych plików tabel można określić dowolny ogranicznik przy użyciu określnika **delimiter is**. Określnik ten ma zastosowanie tylko do rozdzielanych plików .txt.

Składnia:

```
delimiter is char
```

Argumenty:

Argumenty

Argument	Opis
char	Określa pojedynczy znak ze znaków 127 ASCII.

Ponadto można użyć następujących wartości:

Wartości opcjonalne

Wartość	Opisu
'\t'	reprezentuje znak tabulacji, z cudzysłowami lub bez.
'\\'	reprezentuje ukośnik odwrotny (\ ).
'spaces'	reprezentuje wszystkie kombinacje co najmniej jednej spacji. Znaki niedrukowalne o wartości ASCII poniżej 32, z wyjątkiem CR i LF, będą interpretowane jako spacje.

Jeśli nie określono inaczej, przyjmuje się założenie, że plik jest w formacie **delimiter is ','**.

Przykład:

```
LOAD * from a.txt (utf8, txt, delimiter is ',' , embedded labels);
```

Zob. także:

p *Load (page 147)*

No eof

Określnik **no eof** służy do ignorowania znaku na końcu pliku podczas ładowania rozdzielanych plików **.txt**.

Składnia:

```
no eof
```

W przypadku użycia określnika **no eof** znaki z pozycją kodu 26, które w przeciwnym razie oznaczają koniec pliku, są ignorowane i mogą być częścią wartości ciągu.

Dotyczy on tylko dla rozdzielanych plików tekstowych.

Przykład:

```
LOAD * from a.txt (txt, utf8, embedded labels, delimiter is ' ', no eof);
```

Zob. także:

p *Load (page 147)*

Labels

Określnik pliku **Labels** jest używany w instrukcji **LOAD** do zdefiniowania, gdzie w pliku można znaleźć nazwy pól.

Składnia:

```
embedded labels|explicit labels|no labels
```

Nazwy pól mogą występować w różnych miejscach pliku. Jeśli nazwy pól znajdują się w pierwszym rekordzie, należy użyć ustawienia **embedded labels**. Jeśli w ogóle nie ma nazw pól, należy użyć ustawienia **no labels**. W plikach *dif* jest niekiedy używana odrębna sekcja nagłówka z jawnie podanymi nazwami pól. W takim przypadku należy użyć ustawienia **explicit labels**. Jeśli nie zostanie podana żadna wartość, zostanie przyjęte ustawienie **embedded labels**, również w przypadku plików *dif*.

Example 1:

```
LOAD * from a.txt (unicode, txt, delimiter is ',' , embedded labels
```

Example 2:

```
LOAD * from a.txt (codePage is 1252, txt, delimiter is ',' , no labels)
```

Zob. także:

p *Load (page 147)*

Header is

Określa rozmiar nagłówka w plikach tabeli. Określnik **header is** umożliwia określenie dowolnej długości nagłówka. Nagłówek to sekcja tekstowa nieużywana przez aplikację Qlik Sense.

Składnia:

```
header is n
header is line
header is n lines
```

Długość nagłówka można podać w bajtach (**header is n**) lub w liniach (**header is line** lub **header is n lines**). Wartość **n** musi być dodatnią liczbą całkowitą, która będzie reprezentować długość nagłówka. Jeśli nie określono inaczej, przyjmuje się wartość **header is 0**. Określnik **header is** dotyczy wyłącznie plików tabeli.

Przykład:

Jest to przykład tabeli źródła danych zawierającej linię tekstu nagłówka, której aplikacja Qlik Sense nie powinna interpretować jako danych.

```
*Header line
Col1,Col2
a,B
c,D
```

Podczas korzystania z określnika **header is 1 lines** pierwsza linia nie zostanie załadowana jako dane. Na tym przykładzie określnik **embedded labels** informuje aplikację Qlik Sense, aby pierwsza niewykluczona linia była interpretowana jako linia zawierająca etykiety pól.

```
LOAD Col1, Col2
FROM 'lib://files/header.txt'
(txt, embedded labels, delimiter is ',', msq, header is 1 lines);
```

W efekcie otrzymuje się tabelę z dwoma polami, Col1 i Col2.

Zob. także:

p *Load (page 147)*

Record is

W przypadku plików o stałej długości wiersza konieczne jest określenie tej długości przy użyciu określnika **record is**.

Składnia:

```
Record is n
Record is line
Record is n lines
```

Argumenty:

Argumenty

Argument	Opis
n	Określa długość rekordu w bajtach.
line	Określa długość rekordu jako jedną linię.
n lines	Określa długość rekordu w liniach, gdzie n jest dodatnią liczbą całkowitą reprezentującą długość rekordu.

Ograniczenia:

Określnik **record is** dotyczy wyłącznie plików **fix**.

Zob. także:

p *Load (page 147)*

Quotes

Quotes to określnik pliku do instrukcji **LOAD**, który określa, czy dozwolone jest używanie cudzysłowów, i definiuje kolejność pierwszeństwa cudzysłowów i separatorów. Dotyczy tylko plików tekstowych.

Składnia:

```
no quotes
msq
```

Jeśli określnik zostanie pominięty, używane będą standardowe reguły cytowania: można używać cudzysłowów " " lub ' ', ale tylko pod warunkiem, że stanowią one w wartości pola pierwszy i ostatni znak niepusty.

Argumenty:

Argumenty

Argument	Opis
no quotes	Używany, gdy w pliku tekstowym cudzysłowy mają nie być akceptowane.
msq	Używany w celu określenia cytowania w stylu nowoczesnym, umożliwiającego stosowanie w polach zawartości obejmującej wiele wierszy. Pola zawierające znaki końca wiersza muszą być ujęte w podwójne cudzysłowy. Jednym z ograniczeń przy używaniu opcji msq jest interpretowanie samotnego cudzysłowu podwójnego (") będącego pierwszym lub ostatnim znakiem zawartości pola jako początku lub końca zawartości wielowierszowej. Może to dawać nieoczekiwane wyniki w załadowanym zestawie danych. W takim przypadku należy pominąć określnik i używać standardowych reguł cytowania.

XML

Ten określnik skryptu jest używany podczas ładowania plików xml. Poprawne opcje dla określnika **XML** zostały wymienione w składni.

W programie Qlik Sense nie można ładować plików DTD.

Składnia:

```
xmlsimple
```

Zob. także:

p *Load (page 147)*

KML

Ten określnik skryptu jest używany podczas ładowania plików KML do użycia w wizualizacji danych na mapie.

Składnia:

```
kml
```

Plik KML może reprezentować dane obszaru (na przykład kraje lub regiony) reprezentowane przez wielokąty, dane linii (na przykład tory lub drogi) albo dane punktu (na przykład miasta lub miejsca) reprezentowane przez punkty w postaci [długość geograficzna, szerokość geograficzna].

URL is

Ten określnik skryptu jest używany do ustawiania adresu URL połączenia do danych z pliku webowego podczas ładowania pliku webowego.

Składnia:

```
URL is string
```

Argumenty:

Argumenty

Argument	Opis
string	Określa adres URL pliku do załadowania. Spowoduje nadpisanie adresu URL ustawionego w używanym połączeniu do pliku sieciowym.

Ograniczenia:

Określnik **URL is** dotyczy wyłącznie plików webowych. Wymagane jest użycie istniejącego połączenia do danych pliku webowego.

Zob. także:

p *Load (page 147)*

userAgent is

Ten określnik skryptu jest używany do ustawiania agenta użytkownika przeglądarki podczas ładowania pliku webowego.

Składnia:

```
userAgent is string
```

Argumenty:

Argumenty

Argument	Opis
string	Określa ciąg znaków agenta użytkownika przeglądarki. Spowoduje zastąpienie domyślnego agenta "Mozilla/5.0" użytkownika przeglądarki.

Ograniczenia:

Określnik **userAgent is** dotyczy wyłącznie plików webowych.

Zob. także:

p *Load (page 147)*

Let

Instrukcja **let**, uzupełniająca instrukcję **set**, służy do określania zmiennych skryptu. Instrukcja **let** w przeciwieństwie do instrukcji **set** ocenia wyrażenie po prawej stronie znaku „=” w czasie wykonywania skryptu, zanim zostanie przypisane do zmiennej.

Składnia:

```
Let variablename=expression
```

Przykłady i wyniki:

Przykład	Wynik

`Set x=3+4;`
 	
`Let y=3+4;`
 	
`z=$(y)+1;`
	Wynikiem oceny `$(x)` będzie „3+4 ”
	
	Wynikiem oceny `$(y)` będzie „ 7 ”
	
	Wynikiem oceny `$(z)` będzie „ 8 ”
	
	Zwróć uwagę na różnice między instrukcjami **Set** i **Let**. Instrukcja **Set** przypisuje zmiennej ciąg „3+4”, podczas gdy instrukcja **Let** oblicza ciąg i przypisuje do zmiennej 7.

Let T=now( );	`$(T)` otrzyma wartość bieżącego czasu.

Loosen Table

Instrukcja **Loosen Table** umożliwia jawne deklarowanie wewnętrznych tabel danych Qlik Sense jako luźno powiązanych na etapie wykonywania skryptu. Gdy tabela jest luźno powiązana, wszystkie asocjacje między wartościami pola w tabeli są usuwane. Podobny efekt można uzyskać przez załadowanie poszczególnych pól luźno powiązanej tabeli jako niezależnych, niepołączonych tabel. Luźne powiązanie może być użyteczne podczas testowania w celu tymczasowego izolowania różnych części struktury danych. Luźno powiązaną tabelę można zidentyfikować w przeglądarce tabel na podstawie linii przerywanych. Jeśli w skrypcie występuje jakakolwiek instrukcja **Loosen Table**, aplikacja Qlik Sense zignoruje wszelkie ustawienia tabel luźno powiązanych obowiązujące przed wykonaniem skryptu.

Składnia:

```
Loosen Tabletablename [ , tablename2 ...]
Loosen Tablestablename [ , tablename2 ...]
```

Można użyć instrukcji **Loosen Table** lub **Loosen Tables**.

*Jeśli aplikacja Qlik Sense znajdzie w strukturze danych odwołania cykliczne, których nie da się przerwać z użyciem deklaracji tabel luźno powiązanych zdefiniowanych interaktywnie lub jawnie w skrypcie, zostanie wymuszone luźne powiązanie kolejnej tabeli lub tabel, aż do wyeliminowania odwołań cyklicznych. W takiej sytuacji zostanie wyświetlone ostrzeżenie w oknie dialogowym **Ostrzeżenie o pętli**.*

Przykład:

```
Tab1:
SELECT * from Trans;
Loosen Table Tab1;
```

Map

Instrukcja **map ... using** służy do mapowania określonej wartości pola lub wyrażenia na wartości we wskazanej tabeli mapowania. Tabelę mapowania tworzy się instrukcją **Mapping**.

Składnia:

```
Map fieldlist Using  mapname
```

Automatyczne mapowanie jest stosowane do pól załadowanych od wystąpienia instrukcji **Map … Using** do końca skryptu lub do wystąpienia instrukcji **Unmap**.

Mapowanie jest wykonywane jako ostatnia z czynności w ciągu zdarzeń poprzedzającym zapisanie pola w tabeli wewnętrznej aplikacji Qlik Sense. Oznacza to, że mapowanie nie jest wykonywane przy każdym natrafieniu na nazwę pola w ramach wyrażenia, a jedynie podczas zapisywania wartości pod nazwą pola w tabeli wewnętrznej. Jeśli wymagane jest mapowanie na poziomie wyrażenia, należy użyć funkcji **Applymap()**.

Argumenty:

Argumenty

Argument	Opis
fieldlist	Rozdzielana przecinkami lista pól, które mają być mapowane od tego miejsca w skrypcie. Użycie znaku * jako listy pól oznacza wszystkie pola. W nazwach pól dozwolone jest korzystanie z symboli wieloznacznych * i ?. Cytowanie nazw pól może być konieczne, gdy używane są symbole wieloznaczne.
mapname	Nazwa tabeli mapowania, która została poprzednio odczytana w instrukcji **mapping load** lub **mapping select**.

Przykłady i wyniki:

Przykład	Wynik
Map Country Using Cmap;	Umożliwia mapowanie pola Country przy użyciu mapy Cmap.
Map A, B, C Using X;	Umożliwia mapowanie pól A, B i C przy użyciu mapy X.
Map * Using GenMap;	Umożliwia mapowanie wszystkich pól przy użyciu mapy GenMap.

NullAsNull

Instrukcja **NullAsNull** wyłącza konwersję wartości NULL na wartości ciągów znaków ustawione wcześniej przez instrukcję **NullAsValue**.

Składnia:

```
NullAsNull *fieldlist
```

Instrukcja **NullAsValue** działa na zasadzie przełącznika i można ją kilkakrotnie włączać i wyłączać w skrypcie przy użyciu instrukcji **NullAsValue** lub **NullAsNull**.

Argumenty:

Argumenty

Argument	Opis
*fieldlist	Rozdzielana przecinkami lista pól, w odniesieniu do których instrukcja **NullAsNull** powinna być włączona. Użycie znaku * jako listy pól oznacza wszystkie pola. W nazwach pól dozwolone jest korzystanie z symboli wieloznacznych * i ?. Cytowanie nazw pól może być konieczne, gdy używane są symbole wieloznaczne.

Przykład:

```
NullAsNull A,B;
LOAD A,B from x.csv;
```

NullAsValue

Instrukcja **NullAsValue** określa, dla których pól wartość NULL powinna być przekształcona w wartość.

Składnia:

```
NullAsValue *fieldlist
```

Domyślnie program Qlik Sense uznaje wartości NULL za wystąpienia brakujące lub nieokreślone. Niektóre konteksty bazy danych zakładają jednak uwzględnianie wartości NULL jako wartości specjalnych, zamiast uznawać je za brakujące. To, że wartości NULL zazwyczaj nie wolno łączyć z innymi wartościami NULL, można pominąć, korzystając z instrukcji **NullAsValue**.

Instrukcja **NullAsValue** funkcjonuje jak przełącznik i będzie działać na kolejnych instrukcjach ładowania. Można ją ponownie wyłączyć przy użyciu instrukcji **NullAsNull**.

Argumenty:

Argumenty

Argument	Opis
*fieldlist	Rozdzielana przecinkami lista pól, w odniesieniu do których instrukcja **NullAsValue** powinna być włączona. Użycie znaku * jako listy pól oznacza wszystkie pola. W nazwach pól dozwolone jest korzystanie z symboli wieloznacznych * i ?. Cytowanie nazw pól może być konieczne, gdy używane są symbole wieloznaczne.

Przykład:

```
NullAsValue A,B;
```

```
Set NullValue = 'NULL';
LOAD A,B from x.csv;
```

Qualify

Instrukcja **Qualify** służy do przełączania kwalifikacji nazw pól, na przykład aby nazwy te przyjmowały jako prefiks nazwę tabeli.

Składnia:

```
Qualify *fieldlist
```

Przy użyciu instrukcji **qualify**, kwalifikującej nazwę pola z nazwą tabeli, można zawiesić automatyczne sprzężenie pól o tej samej nazwie w różnych tabelach. W przypadku kwalifikacji nazwy pól po znalezieniu ich w tabeli zostaną zmienione. Nowa nazwa będzie miała formę *tablename.fieldname*. *Tablename* to odpowiednik etykiety bieżącej tabeli lub, w przypadku braku etykiety, nazwy pojawiającej się po określeniu **from** w instrukcjach **LOAD** i **SELECT**.

Kwalifikacja zostanie przeprowadzona dla wszystkich pól załadowanych po instrukcji **qualify**.

Kwalifikacja jest zawsze domyślnie wyłączona na początku wykonania skryptu. Kwalifikację nazwy pola można aktywować w dowolnym momencie przy użyciu instrukcji **qualify**. Kwalifikację można wyłączyć w dowolnym momencie przy użyciu instrukcji **Unqualify**.

*Instrukcji **qualify** nie należy stosować w połączeniu z częściowym przeładowaniem.*

Argumenty:

Argumenty

Argument	Opis
*fieldlist	Rozdzielana przecinkami lista pól, w odniesieniu do których kwalifikacja powinna być włączona. Użycie znaku * jako listy pól oznacza wszystkie pola. W nazwach pól dozwolone jest korzystanie z symboli wieloznacznych * i ?. Cytowanie nazw pól może być konieczne, gdy używane są symbole wieloznaczne.

Example 1:

```
Qualify B;
LOAD A,B from x.csv;
LOAD A,B from y.csv;
```

Dwie tabele **x.csv** i **y.csv** są powiązane tylko przez **A**. Rezultatem będą trzy pola: A, x.B, y.B.

Example 2:

W przypadku nieznanej bazy danych zwykle warto zacząć od upewnienia się, że powiązano tylko jedno pole lub kilka pól, jak pokazano na następującym przykładzie:

```
qualify *;
unqualify TransID;
```

```
SQL SELECT * from tab1;
SQL SELECT * from tab2;
SQL SELECT * from tab3;
```

Na potrzeby asocjacji między tabelami *tab1*, *tab2* i *tab3* będzie użyte tylko **TransID**.

Rem

Instrukcja **rem** służy do wstawiania komentarzy do skryptu lub tymczasowego dezaktywowania instrukcji w skrypcie bez usuwania ich.

Składnia:

```
Rem string
```

Wszystko od słowa **rem** do najbliższego średnika **;** jest uznawane za komentarz.

Istnieją dwie inne metody oznaczania komentarzy w skrypcie:

1. Umieszczenie odpowiedniej sekcji między znakami **/*** i ***/** umożliwia utworzenie komentarza w dowolnym miejscu skryptu (z wyjątkiem miejsc znajdujących się wewnątrz cudzysłowów).
2. Wpisanie znaków **//** w skrypcie powoduje zaznaczenie jako komentarza całego tekstu na prawo od nich aż do końca wiersza. (Wyjątkiem jest ciąg znaków //: występujący w adresach internetowych).

Argumenty:

Argumenty

Argument	Opis
string	Dowolny tekst.

Przykład:

```
Rem ** This is a comment **;
/* This is also a comment */
// This is a comment as well
```

Rename

Słowo kluczowe skryptu **Rename** może służyć do zmieniania nazw tabel lub pól, które zostały już załadowane.

Rename field

Ta funkcja skryptu zmienia nazwy istniejących pól aplikacji Qlik Sense po ich załadowaniu.

> *Nie zaleca się nadawania zmiennej nazwy takiej samej, jaką ma pole lub funkcja w programie Qlik Sense.*

Można użyć instrukcji **rename field** lub **rename fields**.

Składnia:

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

Argumenty:

Argument	Opis
mapname	Nazwa wcześniej załadowanej tabeli mapowania zawierającej co najmniej jedną parę starej i nowej nazwy pola.
oldname	Stara nazwa pola.
newname	Nowa nazwa pola.

Ograniczenia:

Nie można zmienić nazw dwóch pól na tę samą nazwę.

Example 1:

```
Rename Field XAZ0007 to Sales;
```

Example 2:

```
FieldMap:
Mapping SQL SELECT oldnames, newnames from datadictionary;
Rename Fields using FieldMap;
```

Rename table

Ta funkcja skryptu zmienia nazwy istniejących tabel wewnętrznych aplikacji Qlik Sense po ich załadowaniu.

Można użyć instrukcji **rename table** lub **rename tables**.

Składnia:

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

Argumenty:

Argumenty

Argument	Opis
mapname	Nazwa wcześniej załadowanej tabeli mapowania zawierającej co najmniej jedną parę starej i nowej nazwy tabeli.
oldname	Stara nazwa tabeli.
newname	Nowa nazwa tabeli.

Ograniczenia:

Nie można nadać tej samej nazwy dwóm tabelom, które poprzednio miały różne nazwy. Skrypt wywoła błąd, jeśli spróbujesz zmienić nazwę tabeli na taką samą, jak nazwa istniejącej tabeli.

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

Instrukcja **Search** służy do uwzględniania albo wykluczania pól w ramach inteligentnego wyszukiwania.

Składnia:

```
Search Include *fieldlist
Search Exclude *fieldlist
```

Można użyć wielu instrukcji Search, aby zawęzić selekcję pól do uwzględnienia. Instrukcje są oceniane od góry do dołu.

Argumenty:

Argumenty

Argument	Opis
*fieldlist	Rozdzielana przecinkami lista pól do uwzględnienia w wyszukiwaniu lub wykluczenia z wyszukiwania inteligentnego. Użycie znaku * jako listy pól oznacza wszystkie pola. W nazwach pól dozwolone jest korzystanie z symboli wieloznacznych * i ?. Cytowanie nazw pól może być konieczne, gdy używane są symbole wieloznaczne.

Przykład:

Przykłady wyszukiwania

Instrukcja	Opisu
`Search Include *;`	Uwzględnienie wszystkich pól w inteligentnych wyszukiwaniach.

Instrukcja	Opisu
`Search Exclude [*ID];`	Wykluczenie z inteligentnego wyszukiwania wszystkich pól kończących się na ID.
`Search Exclude '*ID';`	Wykluczenie z inteligentnego wyszukiwania wszystkich pól kończących się na ID.
`Search Include ProductID;`	Uwzględnienie pola ProductID w inteligentnych wyszukiwaniach.

Zgodnie z połączonym wynikiem tych trzech instrukcji w tej sekwencji wszystkie pola kończące się na ID – inne niż ProductID – są wykluczane z inteligentnych wyszukiwań.

Section

Instrukcja **section** pozwala określić, czy następujące po niej instrukcje **LOAD** i **SELECT** mają być traktowane jako dane czy jako definicje praw dostępu.

Składnia:

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

Jeśli nie określono inaczej, przyjmuje się założenie, że plik jest w formacie **section application**. Definicja **section** obowiązuje do momentu podania nowej instrukcji **section**.

Przykład:

```
Section access;
Section application;
```

Select

Pola są wybierane ze źródła danych ODBC lub dostawcy OLE DB z użyciem standardowych instrukcji **SELECT** języka SQL. Akceptacja instrukcji **SELECT** zależy jednak od używanego sterownika ODBC lub dostawcy OLE DB. Używanie instrukcji **SELECT** wymaga otwartego połączenia danych ze źródłem.

Składnia:

```
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 ]
```

Istnieje też możliwość konkatenacji kilku instrukcji **SELECT** za pomocą operatora **union**:

```
selectstatement Union selectstatement
```

Instrukcja **SELECT** jest interpretowana przez sterownik ODBC lub dostawcę OLE DB, w zależności od zakresu funkcji danych sterowników SQL lub dostawców ODBC mogą zatem występować różnice w porównaniu z ogólną składnią OLE DB, na przykład:

- Klauzula **as** niekiedy nie jest dozwolona, czyli *aliasname* musi następować bezpośrednio po *fieldname*.
- Klauzula **as** niekiedy jest obowiązkowa, gdy używane jest *aliasname*.
- Klauzule **distinct**, **as**,**where**, **group by**, **order by** lub **union** czasami nie są obsługiwane.
- Niektóre sterowniki ODBC nie akceptują niektórych z wymienionych powyżej znaków cudzysłowów.

*Nie jest to pełny opis instrukcji **SELECT** języka SQL! Instrukcje **SELECT** mogą na przykład być zagnieżdżane, jedna instrukcja **SELECT** może zawierać kilka sprzężeń, liczba funkcji dozwolonych w wyrażeniach bywa bardzo duża itd.*

Argumenty:

Argumenty

Argument	Opisu
distinct	**distinct** to predykat używany, jeśli powtarzające się kombinacje wartości w wybranych polach mają być ładowane tylko raz.
distinctrow	**distinctrow** to predykat używany, jeśli powtarzające się rekordy w tabeli źródłowej mają być ładowane tylko raz.
fieldlist	**fieldlist ::= (*| field) {, field }** Lista wybieranych pól. Użycie znaku * jako listy pól oznacza wszystkie pola tabeli. **fieldlist ::= field {, field }** Lista pól, rozdzielana przecinkami. **field ::= (fieldref| expression) [as aliasname]** Jako wyrażenie expression można podać np. funkcję liczbową lub funkcję ciągu znaków opartą na jednym polu lub wielu polach. Niektóre zwykle akceptowane operatory i funkcje są następujące: +, -, *, /, & (konkatenacja ciągu znaków), sum(fieldname), count (fieldname), avg(fieldname)(average), month(fieldname) itp. Więcej informacji na ten temat można znaleźć w dokumentacji sterownika ODBC. **fieldref ::= [tablename.] fieldname** Argumenty **tablename** i **fieldname** to ciągi znaków określające odpowiednio nazwę tabeli i nazwę pola. Jeśli zawierają np. odstępy, muszą być ujęte w proste cudzysłowy podwójne. Klauzula **as** służy do przypisania polu nowej nazwy.

Argument	Opisu
from	**tablelist ::= table {, table }** Lista tabel, z których będą wybierane pola. **table ::= tablename [[as] aliasname]** Argument **tablename** może, ale nie musi być ujęty w cudzysłowy.
where	**where** to klauzula określająca, czy rekord ma być uwzględniony w selekcji, czy też nie. **criterion** to wyrażenie logiczne, które może niekiedy być bardzo złożone. Akceptowane operatory to między innymi operatory i funkcje liczbowe, **=**, **<>** lub **#(**znak nierówności), **>**, **>=**, **<**, **<=**, **and**, **or**,**not**, **exists**,**some**, **all**,**in** oraz nowe instrukcje **SELECT**. Więcej informacji na ten temat można znaleźć w dokumentacji sterownika ODBC lub dostawcy OLE DB.
group by	**group by** to klauzula do agregowania (grupowania) wielu rekordów w jeden. W obrębie jednej grupy wszystkie rekordy dla określonego pola muszą mieć tę samą wartość – w przeciwnym razie pola można używać jedynie z poziomu wyrażenia, np. jako sumy lub średniej. Wyrażenie to może odnosić się do jednego lub wielu pól i jest zdefiniowane w wyrażeniu symbolu pola.
having	**having** to klauzula używana do kwalifikowania grup, działająca analogicznie do klauzuli **where** kwalifikującej rekordy.
order by	**order by** to klauzula do określania kolejności sortowania tabeli wynikowej generowanej przez instrukcję **SELECT**.
join	**join** to kwalifikator określający sprzężenie kilku tabel w jedną. Wszelkie nazwy pól i tabel zawierające odstępy lub znaki diakrytyczne muszą być ujęte w cudzysłowy. W przypadku automatycznego generowania skryptu przez aplikację Qlik Sense używane są znaki cudzysłowu preferowane przez sterownik ODBC lub dostawcę OLE DB z definicji źródła danych podanej w instrukcji **Connect**.

Example 1:

```
SELECT * FROM `Categories`;
```

Example 2:

```
SELECT `Category ID`, `Category Name` FROM `Categories`;
```

Example 3:

```
SELECT `Order ID`, `Product ID`,

`Unit Price` * Quantity * (1-Discount) as NetSales

FROM `Order Details`;
```

Example 4:

```
SELECT `Order Details`.`Order ID`,

Sum(`Order Details`.`Unit Price` * `Order Details`.Quantity) as `Result`

FROM `Order Details`, Orders

where Orders.`Order ID` = `Order Details`.`Order ID`

group by `Order Details`.`Order ID`;
```

Set

Instrukcja **set** jest używana do określania zmiennych skryptu. Mogą one służyć do zastępowania ciągów znaków, ścieżek, dysków itp.

Składnia:

```
Set variablename=string
```

Example 1:

```
Set FileToUse=Data1.csv;
```

Example 2:

```
Set Constant="My string";
```

Example 3:

```
Set BudgetYear=2012;
```

Sleep

Instrukcja **sleep** wstrzymuje wykonanie skryptu przez określony czas.

Składnia:

```
Sleep n
```

Argumenty:

Argument	Opis
n	Określany w milisekundach, gdzie *n* jest dodatnią liczbą całkowitą nie większą niż 3600000 (tzn. 1 godzina). Wartość może być wyrażeniem.

Example 1:

```
Sleep 10000;
```

Example 2:

```
sleep t*1000;
```

SQL

Instrukcja **SQL** umożliwia wysłanie dowolnego polecenia SQL przez połączenie ODBC lub OLE DB.

Składnia:

```
SQL sql_command
```

Wysłanie instrukcji SQL, które powodują aktualizację bazy danych, spowoduje zwrócenie błędu, jeśli aplikacja Qlik Sense otworzyła połączenie ODBC w trybie tylko do odczytu.

Składnia:

```
SQL SELECT * from tab1;
```

jest dozwolona i ze względu na spójność jest składnią preferowaną dla instrukcji **SELECT**. Prefiks SQL zostanie jednak opcjonalny dla instrukcji **SELECT**.

Argumenty:

Argument	Opis
sql_command	Poprawne polecenie SQL.

Example 1:

```
SQL leave;
```

Example 2:

```
SQL Execute <storedProc>;
```

SQLColumns

Instrukcja **sqlcolumns** zwraca zestaw pól opisujących kolumny źródła danych ODBC lub OLE DB, do którego utworzono połączenie **connect**.

Składnia:

```
SQLcolumns
```

Pola mogą być łączone z polami wygenerowanymi przez polecenia **sqltables** i **sqltypes** w celu zapewnienia prawidłowego przeglądu konkretnej bazy danych. Poniżej przedstawiono dwanaście standardowych pól:

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

COLUMN_NAME

DATA_TYPE

TYPE_NAME

PRECISION

LENGTH

SCALE

RADIX

NULLABLE

REMARKS

Szczegółowy opis tych pól zawiera podręcznik referencyjny ODBC.

Przykład:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLcolumns;
```

Niektóre sterowniki ODBC mogą nie obsługiwać tego polecenia. Niektóre sterowniki ODBC mogą generować dodatkowe pola.

SQLTables

Instrukcja **sqltables** zwraca zestaw pól opisujących tabele źródła danych ODBC lub OLE DB, do którego utworzono połączenie **connect**.

Składnia:

```
SQLTables
```

Pola mogą być łączone z polami wygenerowanymi przez polecenia **sqlcolumns** i **sqltypes** w celu zapewnienia prawidłowego przeglądu konkretnej bazy danych. Poniżej przedstawiono pięć standardowych pól:

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

TABLE_TYPE

REMARKS

Szczegółowy opis tych pól zawiera podręcznik referencyjny ODBC.

Przykład:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTables;
```

Niektóre sterowniki ODBC mogą nie obsługiwać tego polecenia. Niektóre sterowniki ODBC mogą generować dodatkowe pola.

SQLTypes

Instrukcja **sqltypes** zwraca zestaw pól opisujących typy źródła danych ODBC lub OLE DB, do którego utworzono połączenie **connect**.

Składnia:

```
SQLTypes
```

Pola mogą być łączone z polami wygenerowanymi przez polecenia **sqlcolumns** i **sqltables** w celu zapewnienia prawidłowego przeglądu konkretnej bazy danych. Poniżej przedstawiono piętnaście standardowych pól:

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

Szczegółowy opis tych pól zawiera podręcznik referencyjny ODBC.

Przykład:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTypes;
```

Niektóre sterowniki ODBC mogą nie obsługiwać tego polecenia. Niektóre sterowniki ODBC mogą generować dodatkowe pola.

Star

Ciąg danych używany na potrzeby reprezentacji zestawu wszystkich wartości pola w bazie danych można ustawić przy użyciu instrukcji **star**. Wpływa on na następne instrukcje **LOAD** i **SELECT**.

Składnia:

```
Star is[ string ]
```

Argumenty:

Argumenty

Argument	Opis
string	Dowolny tekst. Należy pamiętać, że jeśli ciąg znaków zawiera spacje, wówczas musi zostać ujęty w cudzysłowy. Jeśli żadna opcja nie zostanie wybrana, przyjmuje się wartość **star is;**, to znaczy, że symbol gwiazdki nie jest dostępny, chyba że zostało to wyraźnie określone. Ta definicja obowiązuje do momentu podania nowej instrukcji **star**.

Nie zaleca się stosowania instrukcji **Star is** w części skryptu zawierającej dane (w sekcji zainicjowanej instrukcją **Section Application**), jeśli jest używany dostęp do sekcji. Znak gwiazdki jest jednak w pełni obsługiwany w odniesieniu do pól chronionych w części **Section Access** (dostęp do sekcji) skryptu. W tym przypadku nie ma konieczności używania jawnej instrukcji **Star is**, ponieważ w dostępie do sekcji jest ona zawsze domniemana.

Ograniczenia

- Nie można używać znaku gwiazdki z polami kluczy, czyli polami, które tworzą powiązania tabel.
- Nie można używać znaku gwiazdki z żadnymi polami objętymi instrukcją **Unqualify**, ponieważ może to mieć wpływ na pola, które tworzą powiązania tabel.
- Nie można używać znaku gwiazdki z tabelami innymi niż logiczne, na przykład tabelami typu „info-load” lub „mapping-load”.
- Kiedy znak gwiazdki jest używany w polu kompresującym (polu tworzącym powiązania z danymi) w dostępie do sekcji, reprezentuje on wartości wymienione w tym polu w dostępie do sekcji. Nie reprezentuje natomiast innych wartości, które mogą istnieć w danych, ale nie są wymienione w dostępie do sekcji.

- Nie można używać znaku gwiazdki z polami objętymi jakąkolwiek formą kompresji danych poza obszarem **Section Access** (dostęp do sekcji).

Przykład

Poniższy przykład zawiera wyodrębnienie skryptu ładowania danych z dostępem do sekcji.

```
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

];
```

Obowiązują następujące ustalenia:

- Znak *Star* to *.
- Użytkownik *ADMIN* widzi wszystkie pola. Nic nie jest pominięte.
- Użytkownik *USER1* nie widzi pola *SALES*.
- Użytkownik *USER2* nie widzi pola *WAREHOUSE* .

- Użytkownik *USER3* nie widzi pola *EMPLOYEES*.
- Użytkownik *USER4* jest dodawany dwukrotnie do rozwiązania, aby OMINĄĆ dla pola dotyczące tego użytkownika: *SALES* i *WAREHOUSE*.
- *USER5* ma dodany znak „*”, co oznacza, że wszystkie pola w polu OMIT są niedostępne, czyli użytkownik *USER5* nie widzi pól *SALES*, *WAREHOUSE* i *EMPLOYEES* , ale widzi pole *ORDERS*.

Store

Instrukcja **Store** tworzy plik QVD, CSV lub text.

Składnia:

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

Ta instrukcja utworzy jawnie nazwany plik QVD, CSV lub TXT.

Instrukcja może eksportować pola tylko z jednej tabeli danych. W przypadku eksportowania pól z kilku tabel należy wcześniej wykonać w skrypcie jawną instrukcję join, aby utworzyć tabelę danych do wyeksportowania.

Wartości tekstowe są eksportowane do pliku CSV w formacie UTF-8. Możliwe jest określenie ogranicznika – zobacz opis instrukcji **LOAD**. Instrukcja **store** do pliku CSV nie obsługuje eksportu BIFF.

Argumenty:

Argumenty polecenia Store

Argument	Opis
*fieldlist::= (* | field) { , field })*	Lista wybieranych pól. Użycie znaku * jako listy pól oznacza wszystkie pola. *field::= fieldname [***as** *aliasname]* *fieldname* jest tekstem identycznym z nazwą pola w tabeli *table*. (Jeśli nazwa pola zawiera spacje lub inne znaki niestandardowe, musi być ujęta w proste podwójne cudzysłowy lub nawiasy kwadratowe). *aliasname* jest alternatywną nazwą dla pola, która będzie używana w wynikowym pliku QVD lub CSV.
table	Etykieta skryptu reprezentująca już załadowaną tabelę do użycia jako źródło danych.

Argument	Opis
filename	Nazwa pliku docelowego wraz z poprawną ścieżką połączenia do danych istniejącego folderu. **Przykład:** ***'lib://Table Files/target.qvd'*** W dotychczasowym trybie tworzenia skryptów obsługiwane są również następujące formaty ścieżek: • bezwzględna **Przykład:** ***c:\data\sales.qvd*** • względna wobec katalogu roboczego aplikacji Qlik Sense. **Przykład:** ***data\sales.qvd*** Pominięcie ścieżki spowoduje, że aplikacja Qlik Sense zapisze plik w katalogu wskazanym instrukcją **Directory**. W przypadku braku instrukcji **Directory** aplikacja Qlik Sense zapisuje plik w katalogu roboczym, którym jest *C:\Users\{user}\Documents\Qlik\Sense\Apps*.
*format-spec ::=***(** (**txt** | **qvd**) **)**	Specyfikacja formatu obejmuje tekst **txt** dla plików tekstowych albo tekst **qvd** dla plików qvd. Jeśli specyfikacja formatu będzie pominięta, wówczas zostanie przyjęty format **qvd**.

Przykłady:

```
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 * from mytable into 'lib://FolderConnection/myfile.qvd';
```

W rozszerzeniach plików powiązań DataFiles rozróżniana jest wielkość liter. Na przykład: .qvd.

Table/Tables

Słowa kluczowe skryptu **Table** i **Tables** są używane w instrukcjach **Drop**, **Comment** i **Rename**, analogicznie jak określnik formatu w instrukcjach **Load**.

Tag

Ta instrukcja skryptu umożliwia przypisywanie znaczników do jednego lub większej liczby pól lub tabel. W przypadku próby oznaczenia pola lub tabeli niewystępującej w aplikacji oznaczanie zostanie zignorowane. W przypadku znalezienia niezgodnych wystąpień pola lub znacznika używana jest ostatnia wartość.

Składnia:

```
Tag [field|fields] fieldlist with tagname
```

```
Tag [field|fields] fieldlist using mapname
```

```
Tag table tablelist with tagname
```

Argumenty

Argument	Opisu
fieldlist	Jedno lub kilka pól do opatrzenia znacznikiem na liście rozdzielanej przecinkami.
mapname	Nazwa tabeli mapowania, która została poprzednio załadowana w instrukcji **mapping Load** lub **mapping Select**.
tablelist	Rozdzielana przecinkami lista pól do opatrzenia znacznikiem.
tagname	Nazwa znacznika, który powinien zostać zastosowany względem pola.

Example 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
];
tag fields using tagmap;
```

Example 2:

```
tag field Alpha with 'MyTag2';
```

Trace

Instrukcja **trace** zapisuje ciąg znaków w oknie **Postęp wykonania skryptu** oraz w pliku dziennika skryptu (jeśli jest używany). Może być bardzo przydatna do celów debugowania. Komunikat taki można dostosować przy użyciu rozszerzeń „$” zmiennych obliczanych przed instrukcją **trace**.

Składnia:

```
Trace string
```

Example 1:

Poniższej instrukcji można użyć zaraz po instrukcji Load, która ładuje tabelę „Main”.

```
Trace Main table loaded;
```

Spowoduje to wyświetlenie tekstu „Załadowano tabelę Main” w oknie dialogowym wykonywania skryptu oraz w pliku dziennika.

Example 2:

Poniższych instrukcji można użyć zaraz po instrukcji Load, która ładuje tabelę „Main”.

```
Let MyMessage = NoOfRows('Main') & ' rows in Main table';

Trace $(MyMessage);
```

Spowoduje to wyświetlenie tekstu pokazującego liczbę wierszy w oknie dialogowym wykonywania skryptu oraz w pliku dziennika, na przykład „265 391 wierszy w tabeli Main”.

Unmap

Instrukcja **Unmap** powoduje dezaktywację mapowania wartości pola określonej w poprzedniej instrukcji **Map … Using** dla kolejno ładowanych pól.

Składnia:

```
Unmap *fieldlist
```

Argumenty:

Argumenty

Argument	Opis
*fieldlist	Rozdzielana przecinkami lista pól, które nie powinny być już mapowane od tego miejsca w skrypcie. Użycie znaku * jako listy pól oznacza wszystkie pola. W nazwach pól dozwolone jest korzystanie z symboli wieloznacznych * i ?. Cytowanie nazw pól może być konieczne, gdy używane są symbole wieloznaczne.

Przykłady i wyniki:

Przykład	Wynik
Unmap Country;	Wyłącza mapowanie pola Country.
Unmap A, B, C;	Wyłącza mapowanie pól A, B i C.
Unmap * ;	Wyłącza mapowanie wszystkich pól.

Unqualify

Instrukcja **Unqualify** służy do wyłączania kwalifikowania nazw pól, które zostało wcześniej włączone przy użyciu instrukcji **Qualify**.

Składnia:

```
Unqualify *fieldlist
```

Argumenty:

Argumenty

Argument	Opis
*fieldlist	Rozdzielana przecinkami lista pól, w odniesieniu do których kwalifikacja powinna być włączona. Użycie znaku * jako listy pól oznacza wszystkie pola. W nazwach pól dozwolone jest korzystanie z symboli wieloznacznych * i ?. Cytowanie nazw pól może być konieczne, gdy używane są symbole wieloznaczne. Więcej informacji na ten temat można znaleźć w dokumentacji dotyczącej instrukcji **Qualify**.

Example 1:

W przypadku nieznanej bazy danych zwykle warto zacząć od upewnienia się, że powiązano tylko jedno pole lub kilka pól, jak pokazano na następującym przykładzie:

```
qualify *;
unqualify TransID;
SQL SELECT * from tab1;
SQL SELECT * from tab2;
SQL SELECT * from tab3;
```

Najpierw włącza się kwalifikacje dla wszystkich pól.
Następnie kwalifikacja jest wyłączana dla **TransID**.
Na potrzeby asocjacji między tabelami *tab1*, *tab2* i *tab3* będzie użyte tylko **TransID**. Wszystkie pozostałe pola zostaną zakwalifikowane z nazwą tabeli.

Untag

Ta instrukcja skryptu umożliwia usuwanie znaczników z pól lub tabel. W przypadku próby usunięcia oznaczenia pola lub tabeli niewystępującej w aplikacji usunięcie oznaczenia zostanie zignorowane.

Składnia:

```
Untag [field|fields] fieldlist with tagname
```

```
Untag [field|fields] fieldlist using mapname
```

```
Untag table tablelist with tagname
```

Argumenty:

Argumenty

Argument	Opis
fieldlist	Jedno lub kilka pól, których znaczniki powinny zostać usunięte, na liście rozdzielanej przecinkami.
mapname	Nazwa tabeli mapowania, która została poprzednio załadowana w instrukcji **LOAD** lub **SELECT** mapowania.
tablelist	Rozdzielana przecinkami lista tabel, których znaczniki mają zostać usunięte.
tagname	Nazwa znacznika, który powinien zostać usunięty z pola.

Example 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
];
Untag fields using tagmap;
```

Example 2:

```
Untag field Alpha with MyTag2;
```

2.6 Katalog roboczy

W przypadku odwoływania się do pliku w instrukcji skryptu, gdy ścieżka jest pominięta, aplikacja Qlik Sense wyszukuje pliki w następującej kolejności:

1. Katalog określony przez instrukcję **Directory** (obsługiwane tylko w dotychczasowym trybie tworzenia skryptów).
2. W przypadku braku instrukcji **Directory** aplikacja Qlik Sense prowadzi wyszukiwanie w katalogu roboczym.

Katalog roboczy Qlik Sense Desktop

W aplikacji Qlik Sense Desktop katalog roboczy to *C:\Users\{user}\Documents\Qlik\Sense\Apps*.

Katalog roboczy Qlik Sense

W instalacji serwera Qlik Sense katalog roboczy jest określony w usłudze Qlik Sense Repository Service. Domyślnie jest to *C:\ProgramData\Qlik\Sense\Apps*. Więcej informacji zawiera pomoc do konsoli Konsola zarządzania Qlik.

2 Praca ze zmiennymi w edytorze ładowania danych

Zmienna w aplikacji Qlik Sense to kontener, w którym przechowywana jest wartość statyczna lub obliczenie, na przykład wartość liczbowa lub alfanumeryczna. W przypadku użycia tej zmiennej w aplikacji, wszelkie dokonywane w niej zmiany zostaną zastosowane wszędzie tam, gdzie jest ona używana. Zmienne można definiować z użyciem podglądu zmiennych albo w skrypcie przy użyciu edytora ładowania danych. Do ustawiania wartości zmiennej służą instrukcje **Let** i **Set** w skrypcie ładowania danych.

Praca ze zmiennymi aplikacji Qlik Sense możliwa jest również z podglądu zmiennych podczas edycji arkusza.

2.7 Przegląd

Jeśli pierwszy znak wartości zmiennej będzie znakiem równości „=", aplikacja Qlik Sense podejmie próbę obliczenia formuły (wyrażenia Qlik Sense) i wyświetlenia lub zwrócenia wyniku takiego obliczenia zamiast samej treści formuły.

W chwili użycia nazwa zmiennej jest zastępowana jej wartością. Zmiennych można używać w skrypcie na potrzeby rozszerzenia przez znak dolara i różnych instrukcji sterowania. Jest to szczególnie przydatne, gdy w wielu miejscach skryptu powtarza się ten sam ciąg znaków (np. ścieżka).

Niektóre specjalne zmienne systemowe są ustawiane przez aplikację Qlik Sense przy rozpoczęciu wykonywania skryptu niezależnie od ich wcześniejszych wartości.

2.8 Definiowanie zmiennej

Zmienne umożliwiają przechowywanie wartości statycznych lub wyników obliczeń. Podczas definiowania zmiennej użyj następującej składni:

```
set variablename = string
```

lub

```
let variable = expression
```

Instrukcja **Set** służy do przypisywania ciągów. Powoduje przypisanie zmiennej wartości tekstowej podanej na prawo od znaku równości. Instrukcja **Let** ocenia wyrażenie po prawej stronie znaku równości w czasie wykonywania skryptu i przypisuje wynik wyrażenia do zmiennej.

W nazwach zmiennych rozróżniana jest wielkość liter.

Nie zaleca się nadawania zmiennej nazwy takiej samej, jaką ma pole lub funkcja w programie Qlik Sense.

Przykłady:

`set x = 3 + 4; //` zmienna otrzyma ciąg „3 + 4” jako wartość.

`let x = 3 + 4; //` zwraca 7 jako wartość.

`set x = Today(); //` zwraca „Today()” jako wartość.

`let x = Today(); //` zwraca dzisiejszą datę jako wartość, na przykład „27.09.2021”.

2.9 Usuwanie zmiennej

Po usunięciu zmiennej ze skryptu i przeładowaniu danych zmienna pozostanie w aplikacji. Jeśli wymagane jest całkowite usunięcie zmiennej z aplikacji, należy również usunąć tę zmienną z okna dialogowego zmiennych.

2.10 Ładowanie wartości zmiennej jako wartości pola

Jeśli chcesz załadować wartość zmiennej jako wartość pola w instrukcji **LOAD** , a wynik rozszerzenia przez znak dolara jest tekstem, a nie jest liczbą ani wyrażeniem, wówczas wymagane jest umieszczenie rozwiniętej zmiennej w pojedynczych cudzysłowach.

Przykład:

W tym przykładzie do tabeli ładowana jest zmienna systemowa zawierająca listę błędów skryptu. Można zauważyć, że rozszerzenie zmiennej ScriptErrorCount w klauzuli **If** nie wymaga cudzysłowów, a rozszerzenie zmiennej ScriptErrorList wymaga cudzysłowów.

```
IF $(ScriptErrorCount) >= 1 THEN

   LOAD '$(ScriptErrorList)' AS Error AutoGenerate 1; END IF
```

2.11 Obliczanie zmiennej

Istnieje kilka sposobów używania zmiennych z obliczonymi wartościami w aplikacji Qlik Sense, a wynik zależy od sposobu ich określenia oraz wywoływania w wyrażeniu.

W tym przykładzie ładujemy dane wbudowane:

```
LOAD * INLINE [
    Dim, Sales
    A, 150
    A, 200
    B, 240
    B, 230
    C, 410
```

```
    C, 330
];
```

Zdefiniujmy dwie zmienne:

```
Let vSales  = 'Sum(Sales)' ;
Let vSales2  = '=Sum(Sales)' ;
```

W drugiej zmiennej dodajemy znak równości przed wyrażeniem. Spowoduje to obliczenie zmiennej przed jej rozwinięciem i ocenę wyrażenia.

W przypadku użycia zmiennej vSales w niezmienionej postaci, na przykład w mierze, wynikiem będzie ciąg Sum(Sales), co oznacza, że obliczenie nie jest wykonywane.

W przypadku dodania rozszerzenia przez znak dolara i wywołania $(vSales) w wyrażeniu, zmienna zostaje rozwinięta i wyświetlana jest suma Sales.

Z kolei po wywołaniu $(vSales2) zmienna będzie obliczona przed rozwinięciem. Oznacza to, że wyświetlony wynik jest sumą całkowitą Sales. Różnica między użyciem =$(vSales) i =$(vSales2) jako wyrażeń miary jest widoczna w tym wykresie ukazującym wyniki.

Wyniki

Dim	$(vSales)	$(vSales2)
A	350	1560
B	470	1560
C	740	1560

Jak widać, $(vSales) daje w wyniku sumę częściową dotyczącą wartości wymiaru, natomiast $(vSales2) – sumę całkowitą.

Dostępne są następujące zmienne skryptu:

- *Zmienne błędu (page 262)*
- *Zmienne interpretacji liczb (page 199)*
- *Zmienne systemowe (page 190)*
- *Zmienne obsługi wartości (page 197)*

2.12 Zmienne systemowe

Zmienne systemowe, z których część jest definiowana przez system, udostępniają informacje na temat systemu i aplikacji Qlik Sense.

Przegląd zmiennych systemowych

Niektóre z funkcji są po podsumowaniu opisane bardziej szczegółowo. W przypadku tych funkcji można kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

Floppy

Zwraca literę napędu pierwszego znalezionego napędu dyskietek (zwykle jest to *a:*). Jest to zmienna definiowana przez system.

```
Floppy
```

Ta zmienna nie jest obsługiwana w trybie standardowym.

CD

Zwraca literę napędu pierwszego znalezionego napędu CD-ROM. Jeśli żaden napęd CD-ROM nie zostanie znaleziony, wówczas zostanie zwrócona litera *c:*. Jest to zmienna definiowana przez system.

```
CD
```

Ta zmienna nie jest obsługiwana w trybie standardowym.

Include

Zmienna **Include/Must_Include** określa plik, który zawiera tekst, jaki powinien zostać umieszczony w skrypcie i oceniony jako kod skryptu. Nie służy do dodawania danych. Można przechowywać części kodu skryptu w oddzielnym pliku tekstowym i używać ich wielokrotnie w wielu aplikacjach. Jest to zmienna definiowana przez użytkownika.

```
$(Include=filename)
$(Must_Include=filename)
```

HidePrefix

Nazwy wszystkich plików zaczynające się od tego ciągu tekstowego zostaną ukryte w taki sam sposób, jak pola systemowe. Jest to zmienna definiowana przez użytkownika.

```
HidePrefix
```

HideSuffix

Nazwy wszystkich plików kończące się tym ciągiem tekstowym zostaną ukryte w taki sam sposób, jak pola systemowe. Jest to zmienna definiowana przez użytkownika.

```
HideSuffix
```

QvPath

Zwraca ciąg przeglądania ścieżki pliku wykonywalnego Qlik Sense. Jest to zmienna definiowana przez system.

```
QvPath
```

Ta zmienna nie jest obsługiwana w trybie standardowym.

QvRoot

Zwraca katalog główny pliku wykonywalnego programu Qlik Sense. Jest to zmienna definiowana przez system.

```
QvRoot
```

Ta zmienna nie jest obsługiwana w trybie standardowym.

QvWorkPath

Zwraca ciąg znaków przeglądania do bieżącej aplikacji Qlik Sense. Jest to zmienna definiowana przez system.

```
QvWorkPath
```

Ta zmienna nie jest obsługiwana w trybie standardowym.

QvWorkRoot

Zwraca katalog główny bieżącej aplikacji Qlik Sense. Jest to zmienna definiowana przez system.

```
QvWorkRoot
```

Ta zmienna nie jest obsługiwana w trybie standardowym.

StripComments

Jeśli ta zmienna zostanie ustawiona na 0, wówczas obcinanie komentarzy /*..*/ i // w skrypcie będzie zablokowane. Jeśli ta zmienna nie zostanie zdefiniowana, wówczas obcinanie komentarzy będzie zawsze wykonywane.

```
StripComments
```

Verbatim

Zazwyczaj przed załadowaniem do bazy danych Qlik Sense z wszystkich wartości pola automatycznie obcinane są spacje wiodące i końcowe (ASCII 32). Ustawienie tej zmiennej na 1 powoduje zawieszenie obcinania spacji. Znaki tabulacji (ASCII 9) i twardej spacji (ANSI 160) nigdy nie są obcinane.

```
Verbatim
```

OpenUrlTimeout

Ta zmienna definiuje limit czasu w sekundach, jaki aplikacja Qlik Sense powinna uwzględniać podczas pobierania danych ze źródeł URL (np. stron HTML). Jeśli zostanie pominięta, wówczas limit czasu będzie ustawiony na około 20 minut.

```
OpenUrlTimeout
```

WinPath

Zwraca ciąg znaków przeglądania do systemu Windows. Jest to zmienna definiowana przez system.

```
WinPath
```

Ta zmienna nie jest obsługiwana w trybie standardowym.

WinRoot

Zwraca katalog główny systemu Windows. Jest to zmienna definiowana przez system.

```
WinRoot
```

Ta zmienna nie jest obsługiwana w trybie standardowym.

CollationLocale

Określa, jakie ustawienia regionalne mają być użyte na potrzeby porządku sortowania i dopasowywana wyszukiwania. Wartość jest nazwą kulturową ustawienia regionalnego, na przykład „en-US”.Jest to zmienna definiowana przez system.

```
CollationLocale
```

CreateSearchIndexOnReload

Ta zmienna określa, czy podczas przeładowania danych mają być tworzone pliki indeksów wyszukiwania.

```
CreateSearchIndexOnReload
```

CreateSearchIndexOnReload

Ta zmienna określa, czy podczas przeładowania danych mają być tworzone pliki indeksów wyszukiwania.

Składnia:

```
CreateSearchIndexOnReload
```

Można określić, czy pliki indeksu wyszukiwania powinny być tworzone podczas przeładowania danych, czy po wysłaniu przez użytkownika pierwszego żądania wyszukiwania. Zaletą tworzenia plików indeksów wyszukiwania podczas przeładowania danych jest skrócenie czasu oczekiwania podczas pierwszego wyszukiwania prowadzonego przez użytkownika. Wadą jest wydłużenie czasu przeładowania danych o czas potrzebny na utworzenie indeksu – należy zrównoważyć te aspekty.

Jeśli ta zmienna zostanie pominięta, pliki indeksów wyszukiwania nie będą tworzone podczas przeładowania danych.

W przypadku aplikacji sesji pliki indeksów wyszukiwania nie będą tworzone podczas przeładowania danych bez względu na ustawienie tej zmiennej.

Example 1: Tworzenie plików indeksów wyszukiwania podczas przeładowania danych

```
set CreateSearchIndexOnReload=1;
```

Example 2: Tworzenie plików indeksów wyszukiwania po pierwszym żądaniu wyszukiwania

```
set CreateSearchIndexOnReload=0;
```

HidePrefix

Nazwy wszystkich plików zaczynające się od tego ciągu tekstowego zostaną ukryte w taki sam sposób, jak pola systemowe. Jest to zmienna definiowana przez użytkownika.

Składnia:

```
HidePrefix
```

Przykład:

```
set HidePrefix='_' ;
```

Jeśli pola systemowe są ukryte, wówczas po użyciu tej instrukcji nazwy pól zaczynające się od znaku podkreślenia nie będą pokazywane na listach nazw plików.

HideSuffix

Nazwy wszystkich plików kończące się tym ciągiem tekstowym zostaną ukryte w taki sam sposób, jak pola systemowe. Jest to zmienna definiowana przez użytkownika.

Składnia:

```
HideSuffix
```

Przykład:

```
set HideSuffix='%';
```

Jeśli pola systemowe są ukryte, wówczas po użyciu tej instrukcji nazwy pól kończące się znakiem procentu nie będą pokazywane na listach nazw plików.

Include

Zmienna **Include/Must_Include** określa plik, który zawiera tekst, jaki powinien zostać umieszczony w skrypcie i oceniony jako kod skryptu. Nie służy do dodawania danych. Można przechowywać części kodu skryptu w oddzielnym pliku tekstowym i używać ich wielokrotnie w wielu aplikacjach. Jest to zmienna definiowana przez użytkownika.

Ta zmienna obsługuje tylko powiązania do danych w folderze w trybie standardowym.

Składnia:

```
$(Include=filename)
```

```
$(Must_Include=filename)
```

Istnieją dwie wersje zmiennej:

- **Include** nie generuje błędu, jeśli nie można znaleźć pliku; nie pojawia się żadna informacja o niepowodzeniu.
- **Must_Include** generuje błąd, jeśli nie można znaleźć pliku.

Jeśli ścieżka nie zostanie określona, nazwa pliku będzie określana względem katalogu roboczego aplikacji Qlik Sense. Można również określić bezwzględną ścieżkę do pliku albo ścieżkę do połączenia do folderu lib://. Przed znakiem równości ani za nim nie należy umieszczać znaku spacji.

*Konstrukcja **set Include =**filename nie ma zastosowania.*

Przykłady:

```
$(Include=abc.txt);
```

```
$(Must_Include=lib://DataFiles/abc.txt);
```

Ograniczenia

Ograniczona kompatybilność między systemami w przypadku plików zakodowanych w UTF-8 w systemie Windows i Linux.

Używanie kodowania UTF-8 z BOM (Byte Order Mark) jest opcjonalne. BOM może kolidować podczas używania UTF-8 w oprogramowaniu, które nie oczekuje bajtów innych niż ASCII na początku pliku, ale mogłoby obsłużyć strumień tekstowy bez BOM.

- Systemy Windows używają BOM w UTF-8 do identyfikacji, że plik jest zakodowany w UTF-8, pomimo tego, że kolejność bajtów jest jednoznaczna.
- Systemy Unix i Linux używają UTF-8 do Unicode, ale nie używają BOM, ponieważ koliduje to ze składnią plików poleceń.

Ma to pewne konsekwencje dla Qlik Sense.

- W systemie Windows każdy plik rozpoczynający się od BOM kodowania UTF-8 jest uważany za plik skryptu w formacie UTF-8. W przeciwnym razie zakłada się kodowanie ANSI.
- W systemie Linux domyślną systemową 8-bitową stroną kodową jest UTF-8. Dlatego UTF-8 działa, chociaż nie zawiera BOM.

W rezultacie nie można zagwarantować przenośności. Nie zawsze jest możliwe utworzenie w systemie Windows pliku, który można zinterpretować w systemie Linux i odwrotnie. Nie ma wzajemnej kompatybilności między tymi dwoma systemami w zakresie plików zakodowanych w UTF-8 ze względu na różną obsługę BOM.

OpenUrlTimeout

Ta zmienna definiuje limit czasu w sekundach, jaki aplikacja Qlik Sense powinna uwzględniać podczas pobierania danych ze źródeł URL (np. stron HTML). Jeśli zostanie pominięta, wówczas limit czasu będzie ustawiony na około 20 minut.

Składnia:

```
OpenUrlTimeout
```

Przykład:

```
set OpenUrlTimeout=10;
```

StripComments

Jeśli ta zmienna zostanie ustawiona na 0, wówczas obcinanie komentarzy /*..*/ i // w skrypcie będzie zablokowane. Jeśli ta zmienna nie zostanie zdefiniowana, wówczas obcinanie komentarzy będzie zawsze wykonywane.

Składnia:

```
StripComments
```

W niektórych sterownikach bazy danych używane są komentarze /*..*/ jako podpowiedzi dotyczące optymalizacji w instrukcjach **SELECT**. W tym przypadku komentarze nie powinny być obcinane przed wysłaniem instrukcji **SELECT** do sterownika bazy danych.

Zalecane jest przywrócenie dla tej zmiennej wartości 1 bezpośrednio po instrukcjach, w których jest wymagana.

Przykład:

```
set StripComments=0;
SQL SELECT * /* <optimization directive> */ FROM Table ;
set StripComments=1;
```

Verbatim

Zazwyczaj przed załadowaniem do bazy danych Qlik Sense z wszystkich wartości pola automatycznie obcinane są spacje wiodące i końcowe (ASCII 32). Ustawienie tej zmiennej na 1 powoduje zawieszenie obcinania spacji. Znaki tabulacji (ASCII 9) i twardej spacji (ANSI 160) nigdy nie są obcinane.

Składnia:

```
Verbatim
```

Przykład:

```
set Verbatim = 1;
```

2.13 Zmienne obsługi wartości

W tej sekcji opisano zmienne używane do obsługi wartości NULL i innych.

Przegląd zmiennych obsługi wartości

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

NullDisplay
Ten zdefiniowany symbol zastąpi wszystkie wartości NULL z ODBC i łączników na najniższym poziomie danych. Jest to zmienna definiowana przez użytkownika.

```
NullDisplay
```

NullInterpret
Ten zdefiniowany symbol, jeśli wystąpi w pliku tekstowym, pliku programu NULL lub w instrukcji wbudowanej, zostanie zinterpretowany jako wartość Excel. Jest to zmienna definiowana przez użytkownika.

```
NullInterpret
```

NullValue
W przypadku użycia instrukcji **NullAsValue** ten zdefiniowany symbol zastąpi wszystkie wartości NULL w polach określonych parametrem **NullAsValue** określonym ciągiem znaków.

```
NullValue
```

OtherSymbol
Definiuje symbol, który ma być traktowany jak „wszystkie inne wartości” przed instrukcją **LOAD/SELECT**. Jest to zmienna definiowana przez użytkownika.

```
OtherSymbol
```

NullDisplay

Ten zdefiniowany symbol zastąpi wszystkie wartości NULL z ODBC i łączników na najniższym poziomie danych. Jest to zmienna definiowana przez użytkownika.

Składnia:

```
NullDisplay
```

Przykład:

```
set NullDisplay='<NULL>';
```

NullInterpret

Ten zdefiniowany symbol, jeśli wystąpi w pliku tekstowym, pliku programu NULL lub w instrukcji wbudowanej, zostanie zinterpretowany jako wartość Excel. Jest to zmienna definiowana przez użytkownika.

Składnia:

```
NullInterpret
```

Przykłady:

```
set NullInterpret=' ';
set NullInterpret =;
```

nie zwróci wartości NULL dla pustych wartości w programie Excel (ale stanie się tak w przypadku pliku tekstowego CSV).

```
set NullInterpret ='';
```

zwróci wartości NULL dla wartości pustych w programie Excel.

NullValue

W przypadku użycia instrukcji **NullAsValue** ten zdefiniowany symbol zastąpi wszystkie wartości NULL w polach określonych parametrem **NullAsValue** określonym ciągiem znaków.

Składnia:

```
NullValue
```

Przykład:

```
NullAsValue Field1, Field2;
set NullValue='<NULL>';
```

OtherSymbol

Definiuje symbol, który ma być traktowany jak „wszystkie inne wartości” przed instrukcją **LOAD/SELECT**. Jest to zmienna definiowana przez użytkownika.

Składnia:

```
OtherSymbol
```

Przykład:

```
set OtherSymbol='+';
LOAD * inline
[X, Y
a, a
b, b];
LOAD * inline
[X, Z
a, a
```

```
+, c];
```
Wartość pola Y='b' będzie teraz połączona z Z='c' przez inny symbol.

2.14 Zmienne interpretacji liczb

Zmienne interpretacji liczb są definiowane przez system. Zmienne są uwzględniane na górze skryptu i stosują ustawienia formatowania liczb w momencie wykonania skryptu. Można je usuwać, edytować lub powielać.

Zmienne interpretacji liczb są generowane automatycznie zgodnie z bieżącymi ustawieniami regionalnymi systemu operacyjnego podczas tworzenia nowej aplikacji. W Qlik Sense Desktop odbywa się to zgodnie z ustawieniami systemu operacyjnego komputera. W Qlik Sense odbywa się to zgodnie z ustawieniami systemu operacyjnego serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Formatowanie waluty

MoneyDecimalSep

Zdefiniowany separator dziesiętny zastępuje symbol dziesiętny waluty na podstawie ustawień regionalnych.

```
MoneyDecimalSep
```

MoneyFormat

Zdefiniowany symbol zastępuje symbol waluty na podstawie ustawień regionalnych.

```
MoneyFormat
```

MoneyThousandSep

Zdefiniowany separator tysięcy zastępuje symbol grupowania cyfr dotyczący waluty na podstawie ustawień regionalnych.

```
MoneyThousandSep
```

Formatowanie liczb

DecimalSep

Zdefiniowany separator dziesiętny zastępuje symbol dziesiętny na podstawie ustawień regionalnych.

```
DecimalSep
```

ThousandSep

Zdefiniowany separator tysięcy zastępuje symbol grupowania cyfr w systemie operacyjnym (ustawienia regionalne).

```
ThousandSep
```

NumericalAbbreviation

Skróty liczbowe to skróty używane dla przedrostków skali wartości liczbowych – na przykład M dla mega oraz miliona (10^6) i µ dla mikro (10^{-6}).

```
NumericalAbbreviation
```

Formatowanie czasu

DateFormat

Ta zmienna środowiskowa definiuje format daty używany jako domyślny w aplikacji. Format ten służy zarówno do interpretacji, jak i formatowania dat. Jeśli zmienna nie jest zdefiniowana, po uruchomieniu skryptu zostanie pobrany format daty ustawień regionalnych systemu operacyjnego.

```
DateFormat
```

TimeFormat

Zdefiniowany format zastępuje format czasu w systemie operacyjnym (ustawienia regionalne).

```
TimeFormat
```

TimestampFormat

Zdefiniowany format zastępuje formaty daty i godziny w systemie operacyjnym (ustawienia regionalne).

```
TimestampFormat
```

MonthNames

Zdefiniowany format zastępuje konwencję zapisu nazw miesięcy na podstawie ustawień regionalnych.

```
MonthNames
```

LongMonthNames

Zdefiniowany format zastępuje konwencję zapisu długich nazw miesięcy na podstawie ustawień regionalnych.

```
LongMonthNames
```

DayNames

Zdefiniowany format zastępuje konwencję zapisu nazw dni tygodnia systemu na podstawie ustawień regionalnych.

```
DayNames
```

LongDayNames

Zdefiniowany format zastępuje konwencję zapisu długich nazw dni tygodnia na podstawie ustawień regionalnych.

```
LongDayNames
```

FirstWeekDay

Liczba całkowita określająca dzień, który ma być pierwszym dniem tygodnia.

```
FirstWeekDay
```

BrokenWeeks

Ustawienie to określa, czy tygodnie są podzielone czy nie.

```
BrokenWeeks
```

ReferenceDay

Ustawienie to określa, który dzień w styczniu ma zostać ustawiony jako dzień referencyjny w celu określenia tygodnia nr 1.

```
ReferenceDay
```

FirstMonthOfYear

Ustawienie to określa miesiąc używany jako pierwszy miesiąc roku, który może zostać wykorzystany do zdefiniowania lat obrotowych używających przesunięcia miesięcznego, na przykład rozpoczynających się 1 kwietnia.

To ustawienia aktualnie nie jest używane, ale jest zarezerwowane do użytku w przyszłości.

Poprawne ustawienia to od 1 (styczeń) do 12 (grudzień). Ustawienie domyślne to 1.

Składnia:

```
FirstMonthOfYear
```

Przykład:

```
Set FirstMonthOfYear=4; //Sets the year to start in April
```

BrokenWeeks

Ustawienie to określa, czy tygodnie są podzielone czy nie.

Składnia:

```
BrokenWeeks
```

Domyślnie w funkcjach Qlik Sense stosuje się podzielone tygodnie. Oznacza to, że:

- W niektórych latach tydzień 1 zaczyna się w grudniu, a w innych tydzień 52 lub 53 przeciąga się na styczeń.
- Co najmniej cztery dni tygodnia 1 przypadają w styczniu.

Alternatywnym wyjściem jest użycie podzielonych tygodni:

- Tydzień 52 ani 53 nie przeciąga się na styczeń.
- Tydzień 1 zaczyna się 1 stycznia i w większości przypadków nie jest pełnym tygodniem.

Można użyć następujących wartości:

- 0 (=użyj niepodzielonych tygodni)
- 1 (=użyj podzielonych tygodni)

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady:

```
Set BrokenWeeks=0; //(use unbroken weeks)
Set BrokenWeeks=1; //(use broken weeks)
```

DateFormat

Ta zmienna środowiskowa definiuje format daty używany jako domyślny w aplikacji przez funkcje zwracające datę, takie jak `date()` i `date#()`. Format ten służy do interpretacji i formatowania dat. Jeśli zmienna nie jest zdefiniowana, po uruchomieniu skryptu zostanie pobrany format daty ustawień regionalnych.

Składnia:

```
DateFormat
```

Przykłady funkcji DateFormat

Przykład	Wynik
`Set DateFormat='M/D/YY'; //(US format)`	To użycie funkcji `DateFormat` definiuje datę w formacie amerykańskim, miesiąc/dzień/rok.
`Set DateFormat='DD/MM/YY'; //(UK date format)`	To użycie funkcji `DateFormat` definiuje datę w formacie brytyjskim, dzień/miesiąc/rok.
`Set DateFormat='YYYY/MM/DD'; //(ISO date format)`	To użycie funkcji `DateFormat` definiuje datę w formacie ISO, rok/miesiąc/dzień.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Domyślne zmienne systemowe

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zbiór danych dat.
- Funkcja `DateFormat`, która będzie używać amerykańskiego formatu daty.

W tym przykładzie zestaw danych jest ładowany do tabeli o nazwie „`Transactions`”. Zawiera ona pole `date`. Zastosowano definicję amerykańską `DateFormat`. Ten wzorzec będzie używany do niejawnej konwersji tekstu na datę po załadowaniu dat w formacie tekstowym.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `month`

Utwórz tę miarę:

```
=sum(amount)
```

Tabela wynikowa

date	miesiąc	=sum(amount)
01/01/2022	Jan	1000
02/01/2022	Feb	2123
03/01/2022	Mar	4124
04/01/2022	Apr	2431

Do niejawnej konwersji tekstu na daty jest używana definicja `DateFormat` MM/DD/RRRR, dlatego pole `date` jest prawidłowo interpretowane jako data. Jak pokazano w tabeli wyników, ten sam format jest używany do wyświetlania daty.

Przykład 2 – Zmiana zmiennej systemowej

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zbiór danych, co w poprzednim przykładzie.
- Funkcja `DateFormat`, która pobiera dane w formacie „DD/MM/RRRR”.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- date
- month

Utwórz tę miarę:

```
=sum(amount)
```

Tabela wynikowa

date	miesiąc	=sum(amount)
01/01/2022	Jan	1000
02/01/2022	Jan	2123
03/01/2022	Jan	4124
04/01/2022	Jan	2431

Ponieważ definicja `DateFormat` została ustawiona na „DD/MM/RRRR”, dwie pierwsze cyfry za pierwszym znakiem „/” zostały zinterpretowane jako numer miesiąca i w efekcie wszystkie rekordy są z miesiąca stycznia.

Przykład 3 – Interpretacja daty

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zbiór danych z datami w formacie numerycznym.
- Zmienna `DateFormat`, która będzie używać formatu „DD/MM/RRRR”.
- Zmienna `date()`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `month`

Utwórz tę miarę:

```
=sum(amount)
```

Tabela wynikowa

date	miesiąc	=sum(amount)
06/03/2022	Jun	1000
06/04/2022	Jun	2123
06/05/2022	Jun	4124
06/07/2022	Jun	2431

W skrypcie ładowania użyj funkcji `date()`, aby przekonwertować datę numeryczną na format daty. Ponieważ nie określono formatu w drugim argumencie funkcji, używa się `DateFormat`. W wyniku tego pole daty używa formatu „MM/DD/RRRR”.

Przykład 4 – Formatowanie obcych dat

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zbiór danych dat.
- Zmienna `DateFormat`, która używa formatu „DD/MM/RRRR”, ale nie jest wykomentowana przy użyciu ukośników.

Skrypt ładowania

```
// 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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `month`

Utwórz tę miarę:

```
=sum(amount)
```

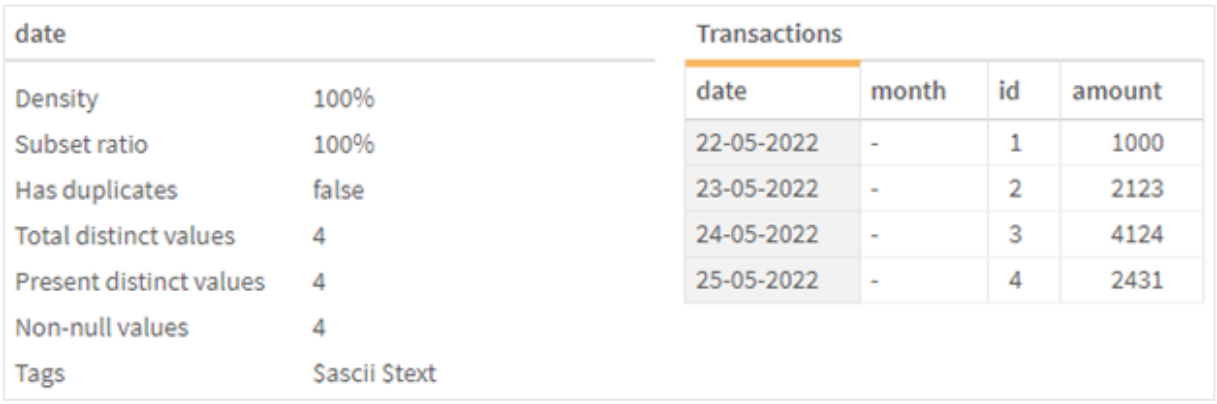
Tabela wynikowa

date	miesiąc	=sum(amount)
22-05-2022	-	1000
23-05-2022	-	2123
24-05-2022	-	4124
25-05-2022	-	2431

W pierwszym skrypcie ładowania używany `DateFormat` to domyślny „MM/DD/RRRR”. Ponieważ pole `date` w zestawie danych transakcji jest w innym formacie, pole nie jest interpretowane jako data. Widać to w tabeli wyników, w której wartości pola `month` są null.

Interpretowane typy danych można zweryfikować w Przeglądarce modelu danych we właściwościach „Tags” pola `date`:

Podgląd tabeli `Transactions`. Zwróć uwagę na „Tags” w polu `date` wskazujące, że tekstowe dane wejściowe nie zostały niejawnie przekonwertowane na datę/znacznik czasu.



Można to rozwiązać, włączając zmienną systemową `DateFormat`:

```
// SET DateFormat='DD/MM/YYYY';
```

Usuń podwójne ukośniki i ponownie załaduj dane.

Podgląd tabeli `Transactions`. Zwróć uwagę na „Tags" w polu `date` wskazujące, że tekstowe dane wejściowe zostały niejawnie przekonwertowane na datę/znacznik czasu.

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

Zdefiniowany format zastępuje konwencję zapisu nazw dni tygodnia systemu na podstawie ustawień regionalnych.

Składnia:

`DayNames`

Podczas modyfikowania zmiennej wymagany jest średnik ; do oddzielenia poszczególnych wartości.

Przykłady funkcji DayName

Przykład funkcji	Definicja wyniku
`Set DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';`	To użycie funkcji DayNames definiuje nazwy dni w ich skróconej formie.
`Set DayNames='M;Tu;W;Th;F;Sa;Su';`	To użycie funkcji DayNames definiuje nazwy dni przy użyciu ich pierwszych liter.

Funkcja `DayNames` jest często używana w połączeniu z następującymi funkcjami:

Powiązane funkcje

Funkcja	Interakcja
weekday (page 1051)	Funkcja skryptu do zwracania `DayNames` jako wartości pola.
Date (page 1209)	Funkcja skryptu zwracająca `DayNames` jako wartości pól.
LongDayNames (page 219)	Długie formy wartości `DayNames`.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Domyślne zmienne systemowe

Skrypt ładowania i wyniki

Przegląd

W tym przykładzie daty w zbiorze danych są ustawione w formacie MM/DD/RRRR.

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych z datami, który zostanie załadowany do tabeli o nazwie `Transactions`.
- Pole `date`.
- Domyślna definicja `DayNames`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `dayname`

Utwórz tę miarę:

```
sum(amount)
```

Tabela wynikowa

date	dayname	sum(amount)
01/01/2022	Sat	1000
02/01/2022	Wto	2123
03/01/2022	Wto	4124
04/01/2022	Fri	2431

W skrypcie ładowania funkcja `WeekDay` jest używana z polem `date` jako podanym argumentem. W tabeli wyników dane wyjściowe tej funkcji `WeekDay` wyświetlają dni tygodnia w formacie definicji `DayNames`.

Przykład 2 – Zmiana zmiennej systemowej

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty. Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak na początku skryptu definicja `DayNames` została zmodyfikowana, aby używać skróconych dni tygodnia w języku afrikaans.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `dayname`

Utwórz tę miarę:

```
sum(amount)
```

Tabela wynikowa

date	dayname	sum(amount)
01/01/2022	Sa	1000
02/01/2022	Di	2123
03/01/2022	Di	4124
04/01/2022	Vr	2431

W tabeli wyników dane wyjściowe tej funkcji `WeekDay` wyświetlają dni tygodnia w formacie definicji `DayNames`.

Należy pamiętać, że jeśli język `DayNames` zostanie zmodyfikowany, tak jak w tym przykładzie, `LongDayNames` nadal będzie zawierać dni tygodnia w języku angielskim. Należałoby to również zmodyfikować, jeśli w aplikacji używane są obie zmienne.

Przykład 3 – Funkcja daty

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych z datami, który zostanie załadowany do tabeli o nazwie `Transactions`.
- Pole `date`.
- Domyślna definicja `DayNames`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `dayname`

Utwórz tę miarę:

```
sum(amount)
```

Tabela wynikowa

date	dayname	sum(amount)
01/01/2022	Sat	1000
02/01/2022	Wto	2123
03/01/2022	Wto	4124
04/01/2022	Fri	2431

Używana jest domyślna definicja `DayNames`. W skrypcie ładowania funkcja `Date` jest używana z polem `date` jako pierwszym argumentem. Drugi argument to `WWW`. To formatowanie konwertuje wynik na wartości przechowywane w definicji `DayNames`. Jest to wyświetlane w wyjściu tabeli wyników.

DecimalSep

Zdefiniowany separator dziesiętny zastępuje symbol dziesiętny na podstawie ustawień regionalnych.

Qlik Sense automatycznie interpretuje tekst jako liczby za każdym razem, gdy napotka rozpoznawalny wzorzec liczb. Zmienne systemowe `ThousandSep` i `DecimalSep` określają skład wzorców stosowanych podczas analizowania tekstu jako liczb. Zmienne `ThousandSep` i `DecimalSep` ustawiają domyślny wzorzec formatu liczb podczas wizualizacji zawartości liczbowej na wykresach i tabelach front-endowych. Oznacza to, że ma bezpośredni wpływ na opcje **formatowania liczb** dla dowolnego wyrażenia front-endowego.

Zakładając, że separatorem tysięcy jest przecinek „,”, a separatorem dziesiętnym „.”, oto przykłady wzorców, które zostałyby niejawnie przekonwertowane na równoważne wartości liczbowe:

```
0,000.00
```

```
0000.00
```

`0,000`

Są to przykłady wzorców, które pozostaną niezmienione jako tekst; czyli nie zostaną przekonwertowane na liczbowe:

`0.000,00`

`0,00`

Składnia:

`DecimalSep`

Przykłady funkcji

Przykład	Wynik
`Set DecimalSep='.';`	Ustawia „.” jako separator dziesiętny.
`Set DecimalSep=',';`	Ustawia „,” jako separator dziesiętny.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład – Wpływ ustawienia zmiennych separatora liczb na różne dane wejściowe

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych sum i dat z sumami ustawionymi w różnych wzorcach formatu.
- Tabela o nazwie `Transactions`.
- Zmienna `DecimalSep`, która jest ustawiona na „.”.

- Zmienna `ThousandSep`, która jest ustawiona na „,”.
- Zmienna `delimiter` ustawiona jako znak „|”, aby oddzielić różne pola w linii.

Skrypt ładowania

```
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 '|');
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `amount`.

Utwórz tę miarę:

```
=sum(amount)
```

Tabela wynikowa

Amount	=Sum(amount)
Sumy	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

Amount	=Sum(amount)	
	4132.5246	4132.5246
	4,787	4787

Każda wartość, która nie jest interpretowana jako liczba, pozostaje tekstem i jest domyślnie wyrównywana do lewej. Wszystkie pomyślnie przekonwertowane wartości są wyrównywane do prawej, zachowując oryginalny format wejściowy.

Kolumna wyrażenia zawiera odpowiednik liczbowy, który jest domyślnie sformatowany tylko z separatorem dziesiętnym „.”. Można to zastąpić ustawieniem z listy rozwijanej **Formatowanie liczb** w konfiguracji wyrażenia.

FirstWeekDay

Liczba całkowita określająca dzień, który ma być pierwszym dniem tygodnia.

Składnia:

```
FirstWeekDay
```

Zmienne systemowe Qlik Sense definiują domyślnie `FirstWeekDay=6`. Oznacza to, że pierwszym dniem tygodnia jest niedziela.

Wartości, które można ustawić dla FirstWeekDay

Wartość	Dzień
0	poniedziałek
1	wtorek
2	środa
3	czwartek
4	piątek
5	sobota
6	niedziela

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień

regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – użycie wartości domyślnej (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

W tym przykładzie skrypt ładowania używa domyślnej Qlik Sense wartości zmiennej systemowej – `FirstWeekDay=6`. To są dane dotyczące czternastu pierwszych dni 2020 roku.

Skrypt ładowania

```
// 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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `week`
- `weekday`

Tabela wynikowa

Data	tydzień	weekday
01/01/2021	1	Śro
01/02/2021	1	Thu
01/03/2021	1	Fri
01/04/2021	1	Sat
01/05/2021	2	Nie
01/06/2020	2	Pon
01/07/2020	2	Wto
01/08/2020	2	Śro
01/09/2020	2	Thu
01/10/2020	2	Fri
01/11/2020	2	Sat
01/12/2020	3	Nie
01/13/2020	3	Pon
01/14/2020	3	Wto

Ponieważ używane są domyślne ustawienia, zmienna systemowa `FirstWeekDay` została ustawiona na 6. W tabeli wyników widać, że każdy tydzień zaczyna się od niedzieli (5 i 12 stycznia).

Przykład 2 – Zmiana zmiennej FirstWeekDay (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

W tym przykładzie dane zawierają informacje dotyczące pierwszych 14 dni 2020 roku. Na początku skryptu ustawiamy zmienną `FirstWeekDay` na 3.

Skrypt ładowania

```
// 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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `week`
- `weekday`

Tabela wynikowa

Data	tydzień	weekday
01/01/2021	52	Śro
01/02/2021	1	Thu
01/03/2021	1	Fri
01/04/2021	1	Sat
01/05/2021	1	Nie
01/06/2020	1	Pon
01/07/2020	1	Wto
01/08/2020	1	Śro
01/09/2020	2	Thu
01/10/2020	2	Fri
01/11/2020	2	Sat
01/12/2020	2	Nie
01/13/2020	2	Pon
01/14/2020	2	Wto

Jako że zmienna systemowa `FirstWeekDay` jest ustawiona na 3, pierwszym dniem każdego tygodnia będzie czwartek. W tabeli wyników widać, że każdy nowy tydzień zaczyna się w czwartek (2 i 9 stycznia).

LongDayNames

Zdefiniowany format zastępuje konwencję zapisu długich nazw dni tygodnia na podstawie ustawień regionalnych.

Składnia:

`LongDayNames`

Poniższy przykład funkcji `LongDayNames` definiuje pełne nazwy dni:

```
Set LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';
```

Podczas modyfikowania zmiennej wymagany jest średnik `;` do oddzielenia poszczególnych wartości.

Funkcji `LongDayNames` można używać w połączeniu z funkcją *Date (page 1209)*, która zwraca `DayNames` jako wartości pola.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Domyślna zmienna systemowa

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych z datami, który zostanie załadowany do tabeli o nazwie `Transactions`.
- Pole `date`.
- Domyślna definicja `LongDayNames`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `dayname`

Utwórz tę miarę:

```
=sum(amount)
```

Tabela wynikowa

date	dayname	=sum(amount)
01/01/2022	sobota	1000
02/01/2022	wtorek	2123
03/01/2022	wtorek	4124
04/01/2022	piątek	2431

W celu utworzenia pola o nazwie `dayname` w skrypcie ładowania funkcja `Date` jest używana z polem `date` jako pierwszym argumentem. Drugim argumentem w funkcji jest formatowanie `WWWW`.

Użycie tego formatowania powoduje konwersję wartości z pierwszego argumentu na odpowiednią pełną nazwę dnia, która jest ustawiona w zmiennej `LongDayNames`. W tabeli wyników wyświetlane są wartości pól naszego utworzonego pola `dayname`.

Przykład 2 – Zmiana zmiennej systemowej

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie. Jednak na początku skryptu definicja `LongDayNames` została zmodyfikowana, aby używać dni tygodnia w języku hiszpańskim.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `dayname`

Utwórz tę miarę:

```
=sum(amount)
```

Tabela wynikowa

date	dayname	=sum(amount)
01/01/2022	Sábado	1000
02/01/2022	Martes	2123
03/01/2022	Martes	4124
04/01/2022	Viernes	2431

W skrypcie ładowania zmienna `LongDayNames` została zmodyfikowana tak, aby wyświetlała dni tygodnia w języku hiszpańskim.

Następnie tworzy się pole o nazwie `dayname`, które jest funkcją `Date` używaną z polem `date` jako pierwszym argumentem.

Drugim argumentem w funkcji jest formatowanie `WWWW`. Użycie tego formatowania powoduje konwersję przez Qlik Sense wartości z pierwszego argumentu na odpowiednią pełną nazwę dnia, która jest ustawiona w zmiennej `LongDayNames`.

W tabeli wyników wartości pól utworzonego przez nas pola `dayname` wyświetlają dni tygodnia napisane w języku hiszpańskim i w całości.

LongMonthNames

Zdefiniowany format zastępuje konwencję zapisu długich nazw miesięcy na podstawie ustawień regionalnych.

Składnia:

```
LongMonthNames
```

Podczas modyfikowania zmiennej należy używać ; do oddzielania poszczególnych wartości.

Poniższy przykład funkcji LongMonthNames definiuje pełne nazwy miesięcy:

```
Set
LongMonthNames='January;February;March;April;May;June;July;August;September;October;November;D
ecember';
```

Funkcja `LongMonthNames` jest często używana w połączeniu z następującymi funkcjami:

Powiązane funkcje

Funkcja	Interakcja
Date (page 1209)	Funkcja skryptu zwracająca `DayNames` jako wartości pól.
LongDayNames (page 219)	Długie formy wartości `DayNames`.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Domyślne zmienne systemowe

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający daty, który jest załadowany do tabeli o nazwie `Transactions`.
- Pole `date`.
- Domyślna definicja `LongMonthNames`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `monthname`

Utwórz tę miarę:

```
=sum(amount)
```

Tabela wynikowa

date	monthname	sum(amount)
01/01/2022	Styczeń	1000.45
01/02/2022	Styczeń	2123.34
01/03/2022	Styczeń	4124.35
01/04/2022	Styczeń	2431.36

date	monthname	sum(amount)
01/05/2022	Styczeń	4787.78
01/06/2022	Styczeń	2431.84
01/07/2022	Styczeń	2854.83
01/08/2022	Styczeń	3554.28
01/09/2022	Styczeń	3756.17
01/10/2022	Styczeń	3454.35

Używana jest domyślna definicja `LongMonthNames`. W celu utworzenia pola o nazwie `month` w skrypcie ładowania funkcja `Date` jest używana z polem `date` jako pierwszym argumentem. Drugim argumentem w funkcji jest formatowanie `MMMM`.

Przy użyciu tego formatowania Qlik Sense konwertuje wartości z pierwszego argumentu na odpowiadające im pełne nazwy miesięcy ustawione w zmiennej `LongMonthNames`. W tabeli wyników wyświetlane są wartości pól naszego utworzonego pola `month`.

Przykład 2 – Zmiana zmiennej systemowej

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający daty, który jest załadowany do tabeli o nazwie `Transactions`.
- Pole `date`.
- Zmienna `LongMonthNames`, która została zmodyfikowana tak, aby używała skróconych dni tygodnia w języku hiszpańskim.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj `sum(amount)` jako miarę oraz te pola jako wymiary:

- `date`
- `monthname`

Utwórz tę miarę:

```
=sum(amount)
```

Tabela wynikowa

date	monthname	sum(amount)
01/01/2022	Enero	1000.45
01/02/2022	Enero	2123.34
01/03/2022	Enero	4124.35
01/04/2022	Enero	2431.36
01/05/2022	Enero	4787.78
01/06/2022	Enero	2431.84
01/07/2022	Enero	2854.83
01/08/2022	Enero	3554.28
01/09/2022	Enero	3756.17
01/10/2022	Enero	3454.35

W skrypcie ładowania zmienna `LongMonthNames` została zmodyfikowana tak, aby wyświetlała miesiące w języku hiszpańskim. Następnie w celu utworzenia pola `monthname` funkcja `Date` jest używana z polem `date` jako pierwszym argumentem. Drugim argumentem w funkcji jest formatowanie `MMMM`.

Przy użyciu tego formatowania Qlik Sense konwertuje wartości z pierwszego argumentu na odpowiadające im pełne nazwy miesięcy ustawione w zmiennej `LongMonthNames`. W tabeli wyników wartości pól naszego utworzonego pola `monthname` przedstawiają nazwę miesiąca zapisaną w języku hiszpańskim.

MoneyDecimalSep

Zdefiniowany separator dziesiętny zastępuje symbol dziesiętny waluty na podstawie ustawień regionalnych.

Qlik Sense domyślnie wyświetla inaczej liczby i tekst na wykresach tabel. Liczby są wyrównane do prawej, a tekst do lewej. Ułatwia to znalezienie problemów z konwersją tekstu na liczbę. Wszystkie tabele na tej stronie, które pokazują wyniki Qlik Sense, będą używać tego formatowania.

Składnia:

MoneyDecimalSep

Qlik Sense Aplikacje będą interpretować pola tekstowe w tym formacie jako wartości pieniężne. Pole tekstowe musi zawierać symbol waluty, który jest zdefiniowany w zmiennej systemowej `MoneyFormat`. Funkcja `MoneyDecimalSep` jest szczególnie przydatna w przypadku pracy ze źródłami danych reprezentującymi różne ustawienia regionalne.

Poniżej znajduje się przykład użycia zmiennej systemowej `MoneyDecimalSep`:

```
Set MoneyDecimalSep='.';
```

Ta funkcja jest często używana w połączeniu z następującymi funkcjami:

Powiązane funkcje

Funkcja	Interakcja
`MoneyFormat`	W przypadku interpretacji pola tekstowego symbol `MoneyFormat` jest używany w ramach interpretacji. W przypadku formatowania liczb formatowanie `MoneyFormat` zostanie użyte przez Qlik Sense w obiektach wykresów.
`MoneyThousandSep`	W przypadku interpretacji pól tekstowych należy przestrzegać także zasad funkcji `MoneyThousandSep`.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Notacja z kropką (.) MoneyDecimalSep

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Transactions`.
- Dostarczone dane zawierające pole walutowe w formacie tekstowym z kropką w roli separatora dziesiętnego. Ponadto prawie każdy rekord jest poprzedzony symbolem „$”. Wyjątkiem jest ostatni rekord, który ma z przodu symbol „£”.

Pamiętaj, że zmienna systemowa `MoneyFormat` definiuje dolara („$”) jako walutę domyślną.

Skrypt ładowania

```
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'
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:`amount`.

Dodaj następujące miary:

- `isNum(amount)`
- `sum(amount)`

Przyjrzyj się poniższym wynikom pokazujący, że poprawnie zostały zinterpretowane tylko wartości z symbolem dolara.

Tabela wynikowa

amount	=isNum(amount)	=Sum(amount)
Sumy	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

Powyższa tabela wyników pokazuje, że pole `amount` zostało prawidłowo zinterpretowane dla wszystkich wartości z symbolem dolara z przodu. Natomiast wartość z symbolem funta (£) z przodu `amount` nie zostało przekonwertowane na wartość walutową.

Przykład 2 – Notacja z przecinkiem (,) MoneyDecimalSep

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Transactions`.
- Dostarczone dane zawierające pole walutowe w formacie tekstowym z przecinkiem w roli separatora dziesiętnego. Ponadto każdy rekord jest poprzedzony symbolem „$”. Ostatni rekord jest wyjątkowy, ponieważ błędnie zastosowano w nim kropkę w roli separatora dziesiętnego.

Pamiętaj, że zmienna systemowa `MoneyFormat` definiuje dolara („$”) jako walutę domyślną.

Skrypt ładowania

```
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'
];
```

Wyniki

Tekst akapitu dotyczący wyników.

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:`amount`.

Dodaj następujące miary:

- `isNum(amount)`
- `sum(amount)`

W poniższych wynikach widać, że prawie wszystkie wartości zostały poprawnie zinterpretowane. Wyjątkiem jest kwota, w której jako separatora dziesiętnego użyto kropki. W tym przypadku powinien zostać użyty przecinek.

Tabela wynikowa

amount	=isNum(amount)	=Sum(amount)
Sumy	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

amount	=isNum(amount)	=Sum(amount)
$249,36	-1	$249.36
$545,36	-1	$545.36
$2.814,32	-1	$2814.32

MoneyFormat

Ta zmienna systemowa definiuje wzorzec formatu używany przez Qlik do automatycznej translacji tekstu na liczbę, gdy liczba ta jest poprzedzona symbolem waluty. Ponadto definiuje sposób prezentacji w obiektach wykresów miar, których właściwości formatowania liczbowego są ustawione na „Money”.

Symbol zdefiniowany jako część wzorca formatu w zmiennej systemowej `MoneyFormat` zastępuje symbol waluty wybrany w ustawieniach regionalnych.

Qlik Sense domyślnie wyświetla inaczej liczby i tekst na wykresach tabel. Liczby są wyrównane do prawej, a tekst do lewej. Ułatwia to znalezienie problemów z konwersją tekstu na liczbę. Wszystkie tabele na tej stronie, które pokazują wyniki Qlik Sense, będą używać tego formatowania.

Składnia:

MoneyFormat

```
Set MoneyFormat='$ #,##0.00; ($ #,##0.00)';
```

To formatowanie będzie stosowane w obiektach wykresów, gdy własność `Number Formatting` pola liczbowego będzie ustawiona na `Money`. Ponadto, kiedy numeryczne pola tekstowe są interpretowane przez Qlik Sense, jeśli symbol waluty pola tekstowego zgadza się z symbolem waluty zdefiniowanym w zmiennej `MoneyFormat`, Qlik Sense zinterpretuje to pole jako wartość walutową.

Ta funkcja jest często używana w połączeniu z następującymi funkcjami:

Powiązane funkcje

Funkcja	Interakcja
MoneyDecimalSep (page 225)	W przypadku formatowania liczb do formatowania pól obiektów używana jest zmienna `MoneyDecimalSep`.
MoneyThousandSep (page 234)	W przypadku formatowania liczb do formatowania pól obiektów używana jest zmienna `MoneyThousandSep`.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne

czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – MoneyFormat

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera zestaw danych, który jest ładowany do tabeli o nazwie `Transactions`. Używana jest domyślna definicja zmiennej `MoneyFormat`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `amount`

Dodaj tę miarę:

```
=Sum(amount)
```

W obszarze **Formatowanie liczb** wybierz pozycję **Pieniądze**, aby ustawić `Sum(amount)` jako wartość walutową.

Tabela wynikowa

date	Amount	=Sum(amount)
Sumy		$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

Używana jest domyślna definicja `MoneyFormat`. Wygląda ona następująco: `$###0.00;-$###0.00`. W tabeli wyników format pola `amount` obejmuje symbol waluty i separator dziesiętny oraz miejsca dziesiętne.

Przykład 2 – MoneyFormat z separatorem tysięcy i mieszanymi formatami wejściowymi

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający mieszankę formatów wejściowych, który został załadowany do tabeli o nazwie `Transactions`, z separatorami tysięcy i dziesiętnym.
- Zmodyfikowana wersja definicji `MoneyFormat` zawiera przecinek jako separator tysięcy.

- Jeden z wierszy ma pomyłkowo wstawione separatory tysięcy (przecinki) w niewłaściwych miejscach. Zwróć uwagę, że ta kwota została zinterpretowana jako tekst, nie jako liczba.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `amount`

Dodaj tę miarę:

```
=Sum(amount)
```

W obszarze **Formatowanie liczb** wybierz pozycję **Pieniądze**, aby ustawić `Sum(amount)` jako wartość walutową.

Tabela wynikowa

date	Amount	=Sum(amount)
Sumy		$119,548,811,911.90
01/01/2022	$10,000,000,441.45	$10,000,000,441.45
01/02/2022	$212,3749,24,32.23	$0.00

date	Amount	=Sum(amount)
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

Na początku skryptu zmienna systemowa `MoneyFormat` została zmodyfikowana, aby definiowała przecinek jako separator tysięcy. W tabeli Qlik Sense widać, że ten separator jest używany do formatowania liczb. Jednak wiersz zawierający błędny separator nie został poprawnie zinterpretowany i ma postać zwykłego tekstu. Dlatego też nie został uwzględniony w kwocie łącznej.

MoneyThousandSep

Zdefiniowany separator tysięcy zastępuje symbol grupowania cyfr dotyczący waluty na podstawie ustawień regionalnych.

Qlik Sense domyślnie wyświetla inaczej liczby i tekst na wykresach tabel. Liczby są wyrównane do prawej, a tekst do lewej. Ułatwia to znalezienie problemów z konwersją tekstu na liczbę. Wszystkie tabele na tej stronie, które pokazują wyniki Qlik Sense, będą używać tego formatowania.

Składnia:

`MoneyThousandSep`

Qlik Sense Aplikacje będą interpretować pola tekstowe w tym formacie jako wartości pieniężne. Pole tekstowe musi zawierać symbol waluty, który jest zdefiniowany w zmiennej systemowej `MoneyFormat`. Funkcja `MoneyThousandSep` jest szczególnie przydatna w przypadku pracy ze źródłami danych reprezentującymi różne ustawienia regionalne.

Poniżej znajduje się przykład użycia zmiennej systemowej `MoneyThousandSep`:

```
Set MoneyDecimalSep=',';
```

Ta funkcja jest często używana w połączeniu z następującymi funkcjami:

Powiązane funkcje

Funkcja	Interakcja
`MoneyFormat`	W przypadku interpretacji pola tekstowego w ramach interpretacji będzie używany symbol `MoneyFormat`. W przypadku formatowania liczb formatowanie `MoneyFormat` zostanie użyte przez Qlik Sense w obiektach wykresów.
`MoneyDecimalSep`	W przypadku interpretacji pól tekstowych należy przestrzegać także zasad funkcji `MoneyDecimalSep`.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Notacja z przecinkiem (,) MoneyThousandSep

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Transactions`.
- Dostarczone dane zawierające pole walutowe w formacie tekstowym z przecinkiem w roli separatora tysięcy. Ponadto każdy rekord jest poprzedzony symbolem „$”.

Pamiętaj, że zmienna systemowa `MoneyFormat` definiuje dolara („$”) jako walutę domyślną.

Skrypt ładowania

```
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'
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:`amount`.

Dodaj następujące miary:

- `isNum(amount)`
- `sum(amount)`

Spójrz na poniższe wyniki. Tabela przedstawia poprawną interpretację wartości zawierających przecinek w roli separatora tysięcy.

Pole `amount` zostało zinterpretowane poprawnie dla wszystkich wartości z wyjątkiem jednego, w którym jako separatora tysięcy użyto kropki.

Tabela wynikowa

amount	=isNum(amount)	=Sum(amount)
Sumy	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

Przykład 2 – Notacja z kropką (.) MoneyThousandSep

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Transactions`.
- Dostarczone dane zawierające pole walutowe w formacie tekstowym z kropką w roli separatora tysięcy. Ponadto każdy rekord jest poprzedzony symbolem „$”.

Pamiętaj, że zmienna systemowa `MoneyFormat` definiuje dolara („$”) jako walutę domyślną.

Skrypt ładowania

```
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'
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:`amount`.

Dodaj następujące miary:

- `isNum(amount)`
- `sum(amount)`

Spójrz na poniższe wyniki przedstawiające poprawną interpretację wszystkich wartości zawierających kropkę w roli separatora tysięcy.

Pole `amount` zostało zinterpretowane poprawnie dla wszystkich wartości z wyjątkiem jednego, w którym jako separatora tysięcy użyto przecinka.

Tabela wynikowa

amount	=isNum(amount)	=Sum(amount)
Sumy	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

Zdefiniowany format zastępuje konwencję zapisu nazw miesięcy na podstawie ustawień regionalnych.

Składnia:

```
MonthNames
```

Podczas modyfikowania zmiennej należy używać ; do oddzielania poszczególnych wartości.

Przykłady funkcji

Przykład	Wyniki
`Set MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';`	To użycie funkcji `MonthNames` definiuje nazwy miesięcy w języku angielskim i w skróconą formę.
`Set MonthNames='Enero;Feb;Marzo;Abr;Mayo;Jun;Jul;Agosto;Set;Oct;Nov;Dic';`	To użycie funkcji `MonthNames` definiuje nazwy miesięcy w języku

Przykład	Wyniki
	hiszpańskim i w skróconą formę.

Funkcja `MonthNames` może być używana w połączeniu z następującymi funkcjami:

Powiązane funkcje

Funkcja	Interakcja
month (page 893)	Funkcja skryptu do zwracania wartości zdefiniowanych w `MonthNames` jako wartości pola
Date (page 1209)	Funkcja skryptu do zwracania wartości zdefiniowanych w `MonthNames` jako wartości pola na podstawie podanego argumentu formatowania
LongMonthNames (page 222)	Długie formy wartości `MonthNames`

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Domyślne zmienne systemowe

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający daty, który jest załadowany do tabeli o nazwie `Transactions`.
- Pole `date`.
- Domyślna definicja `MonthNames`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `monthname`

Utwórz tę miarę:

```
=sum(amount)
```

Tabela wynikowa

date	monthname	sum(amount)
01/01/2022	Jan	1000.45
01/02/2022	Jan	2123.34
01/03/2022	Jan	4124.35
01/04/2022	Jan	2431.36
01/05/2022	Jan	4787.78
01/06/2022	Jan	2431.84
01/07/2022	Jan	2854.83
01/08/2022	Jan	3554.28

date	monthname	sum(amount)
01/09/2022	Jan	3756.17
01/10/2022	Jan	3454.35

Używana jest domyślna definicja `MonthNames`. W skrypcie ładowania funkcja `Month` jest używana z polem `date` jako podanym argumentem.

W tabeli wyników dane wyjściowe tej funkcji `Month` wyświetlają miesiące w formacie definicji `MonthNames`.

Przykład 2 – Zmiana zmiennej systemowej

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający daty, który jest załadowany do tabeli o nazwie `Transactions`.
- Pole `date`.
- Zmienna `MonthNames` , która została zmodyfikowana tak, aby używała skróconych nazw miesięcy w języku hiszpańskim.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `monthname`

Utwórz tę miarę:

```
=sum(amount)
```

Tabela wynikowa

date	monthname	sum(amount)
01/01/2022	Enero	1000.45
01/02/2022	Enero	2123.34
01/03/2022	Enero	4124.35
01/04/2022	Enero	2431.36
01/05/2022	Enero	4787.78
01/06/2022	Enero	2431.84
01/07/2022	Enero	2854.83
01/08/2022	Enero	3554.28
01/09/2022	Enero	3756.17
01/10/2022	Enero	3454.35

W skrypcie ładowania zmienna `MonthNames` została zmodyfikowana tak, aby wyświetlała skrócone nazwy miesięcy w języku hiszpańskim. Funkcja `Month` jest używana z polem `date` jako podanym argumentem.

W tabeli wyników dane wyjściowe tej funkcji `Month` wyświetlają miesiące w formacie definicji `MonthNames`.

Należy pamiętać, że jeśli język zmiennej `MonthNames` zostanie zmodyfikowany, tak jak w tym przykładzie, zmienna `LongMonthNames` nadal będzie zawierać miesiące w języku angielskim. Zmienną `LongMonthNames` należałoby zmodyfikować, jeśli w aplikacji używane są obie zmienne.

Przykład 3 – Funkcja daty

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający daty, który jest załadowany do tabeli o nazwie `Transactions`.
- Pole `date`.
- Domyślna definicja `MonthNames`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `monthname`

Utwórz tę miarę:

```
=sum(amount)
```

Tabela wynikowa

date	monthname	sum(amount)
01/01/2022	Jan	1000.45
01/02/2022	Jan	2123.34
01/03/2022	Jan	4124.35
01/04/2022	Jan	2431.36
01/05/2022	Jan	4787.78
01/06/2022	Jan	2431.84
01/07/2022	Jan	2854.83
01/08/2022	Jan	3554.28
01/09/2022	Jan	3756.17
01/10/2022	Jan	3454.35

Używana jest domyślna definicja `MonthNames`. W skrypcie ładowania funkcja `Date` jest używana z polem `date` jako pierwszym argumentem. Drugi argument to .MMM

Przy użyciu tego formatowania Qlik Sense konwertuje wartości z pierwszego argumentu na odpowiadające im nazwy miesięcy ustawione w zmiennej `MonthNames`. W tabeli wyników wyświetlane są wartości pól naszego utworzonego pola `month`.

NumericalAbbreviation

Skróty liczbowe to skróty używane dla przedrostków skali wartości liczbowych – na przykład M dla mega oraz miliona (10^6) i µ dla mikro (10^{-6}).

Składnia:

```
NumericalAbbreviation
```

Zmienną `NumericalAbbreviation` należy ustawić na łańcuch zawierający listę par definicji skrótów rozdzielanych średnikiem. Każda para definicji skrótów powinna zawierać skalę (wykładnik w postaci podstawy dziesiętnej) oraz skrót po dwukropku, na przykład `6:M` dla miliona.

Ustawieniem domyślnym jest `'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'`.

Przykłady:

To ustawienie zmieni prefiks dla tysięcy na t, a prefiks dla miliardów na B. Taki zapis może być użyteczny w zastosowaniach finansowych, w których można oczekiwać skrótów, takich jak t$, M$ oraz B$.

```
Set NumericalAbbreviation='3:t;6:M;9:B;12:T;15:P;18:E;21:Z;24:Y;-3:m;-6:µ;-9:n;-12:p;-15:f;-18:a;-21:z;-24:y';
```

ReferenceDay

To ustawienie określa, który dzień stycznia ma być ustawiony jako dzień odniesienia, aby zdefiniować tydzień 1. Innymi słowy, to ustawienie określa, ile dni w tygodniu 1 musi być datami styczniowymi.

Składnia:

```
ReferenceDay
```

`ReferenceDay` określa liczbę dni w pierwszym tygodniu roku. `ReferenceDay` można ustawić na dowolną wartość z zakresu od `1` do `7`. Każda wartość spoza zakresu `1-7` jest interpretowana jako środek tygodnia (`4`), co jest równoważne ustawieniu `ReferenceDay` na `4`.

Jeśli nie wybierzesz wartości dla ustawienia `ReferenceDay`, domyślna wartość będzie pokazywać `ReferenceDay=0`, co będzie interpretowane jako środek tygodnia (`4`), jak widać w tabeli wartości `ReferenceDay` poniżej.

Funkcja `ReferenceDay` jest często używana w połączeniu z następującymi funkcjami:

Powiązane funkcje

Zmienna	Interakcja
BrokenWeeks (page 201)	Jeśli aplikacja Qlik Sense działa z niepodzielonymi tygodniami, ustawienie zmiennej `ReferenceDay` zostanie wymuszone. Jeśli jednak używane są podzielone tygodnie, tydzień 1 rozpocznie się 1 stycznia i zakończy się w połączeniu z ustawieniem zmiennej `FirstWeekDay` oraz zignoruje flagę `ReferenceDay`.
FirstWeekDay (page 215)	Liczba całkowita określająca dzień, który ma być pierwszym dniem tygodnia.

Qlik Sense umożliwia ustawienie następujących wartości dla `ReferenceDay`:

Wartości ReferenceDay

Wartość	Dzień referencyjny
0 (domyślnie)	January 4
1	January 1
2	Styczeń 2 r.
3	January 3
4	January 4
5	January 5
6	January 6
7	January 7

W poniższym przykładzie `ReferenceDay = 3` definiuje 3 stycznia jako dzień odniesienia:

```
SET ReferenceDay=3; //(set January 3 as the reference day)
```

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – skrypt ładowania używający wartości domyślnej; ReferenceDay=0

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zmienna `ReferenceDay`, która jest ustawiona na `0`.
- Zmienna `BrokenWeeks` ustawiona na `0`, która zmusza aplikację do używania niepodzielonych

tygodni.

- Zestaw danych dat od końca 2019 r. do początku 2020 r.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `week`
- `weekday`

Tabela wynikowa

date	tydzień	weekday
12/27/2019	52	Fri
12/28/2019	52	Sat
12/29/2019	1	Nie
12/30/2019	1	Pon

date	tydzień	weekday
12/31/2019	1	Wto
01/01/2020	1	Śro
01/02/2020	1	Thu
01/03/2020	1	Fri
01/04/2020	1	Sat
01/05/2020	2	Nie
01/06/2020	2	Pon
01/07/2020	2	Wto
01/08/2020	2	Śro
01/09/2020	2	Thu
01/10/2020	2	Fri
01/11/2020	2	Sat

Tydzień 52 kończy się w sobotę 28 grudnia. Ponieważ `ReferenceDay` wymaga uwzględnienia 4 stycznia w tygodniu 1, tydzień 1 rozpoczyna się 29 grudnia i kończy w sobotę 4 stycznia.

Przykład – zmienna ReferenceDay ustawiona na 5

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zmienna `ReferenceDay`, która jest ustawiona na `5`.
- Zmienna `BrokenWeeks` ustawiona na `0`, która zmusza aplikację do używania niepodzielonych tygodni.
- Zestaw danych dat od końca 2019 r. do początku 2020 r.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `week`
- `weekday`

Tabela wynikowa

date	tydzień	weekday
12/27/2019	52	Fri
12/28/2019	52	Sat
12/29/2019	53	Nie
12/30/2019	53	Pon
12/31/2019	53	Wto
01/01/2020	53	Śro
01/02/2020	53	Thu
01/03/2020	53	Fri
01/04/2020	53	Sat
01/05/2020	1	Nie
01/06/2020	1	Pon
01/07/2020	1	Wto
01/08/2020	1	Śro
01/09/2020	1	Thu
01/10/2020	1	Fri
01/11/2020	1	Sat

Tydzień 52 kończy się w sobotę 28 grudnia. Zmienna `BrokenWeeks` zmusza aplikację do używania niepodzielonych tygodni. Wartość dnia referencyjnego `5` wymaga uwzględnienia 5 stycznia w tygodniu 1.

Jest to jednak osiem dni po zakończeniu 52. tygodnia poprzedniego roku. Dlatego 53. tydzień rozpoczyna się 29 grudnia i kończy 4 stycznia. Tydzień 1. zaczyna się w niedzielę 5 stycznia.

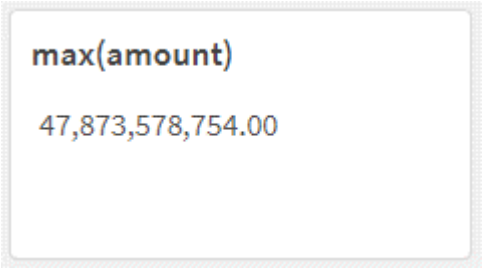
ThousandSep

Zdefiniowany separator tysięcy zastępuje symbol grupowania cyfr w systemie operacyjnym (ustawienia regionalne).

Składnia:

`ThousandSep`

Qlik Sense Obiekt używający zmiennej `ThousandSep` (z separatorem tysięcy)



Qlik Sense Aplikacje interpretują pola tekstowe zgodne z tym formatem jako liczby. To formatowanie będzie stosowane w obiektach wykresów, gdy własność **Number formatting** pola liczbowego będzie ustawiona na **Number**.

`ThousandSep` pomaga w przetwarzaniu źródeł danych o zróżnicowanych ustawieniach regionalnych.

> *Jeśli zmienna `ThousandSep` zostanie zmodyfikowana po utworzeniu i sformatowaniu obiektów w aplikacji, użytkownik będzie musiał ponownie sformatować każde interesujące go pole przez usunięcie zaznaczenia i ponowne wybranie własności **formatowania liczb Number**.*

Poniżej znajdują się przykłady użycia zmiennej systemowej `ThousandSep`:

```
Set ThousandSep=','; //(for example, seven billion will be displayed as: 7,000,000,000)
```

```
Set ThousandSep=' '; //(for example, seven billion will be displayed as: 7 000 000 000)
```

W pracy z tą funkcją mogą Ci pomóc poniższe tematy:

Tematy pokrewne

Temat	Opis
DecimalSep (page 212)	W przypadkach interpretacji pól tekstowych należy także uwzględnić ustawienia separatora dziesiętnego określone przez tę funkcję. W przypadku formatowania liczb w razie potrzeby Qlik Sense używa **DecimalSep**.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Domyślne zmienne systemowe

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Transactions`.
- Użycie domyślnej definicji zmiennej `ThousandSep`.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:`date`.
2. Dodaj następującą miarę:
   `=sum(amount)`
3. W panelu właściwości, w obszarz **Data** wybierz miarę.
4. W obszarze **Formatowanie liczb** wybierz **Liczba**.

Dostosowywanie formatowania liczb dla miary wykresu

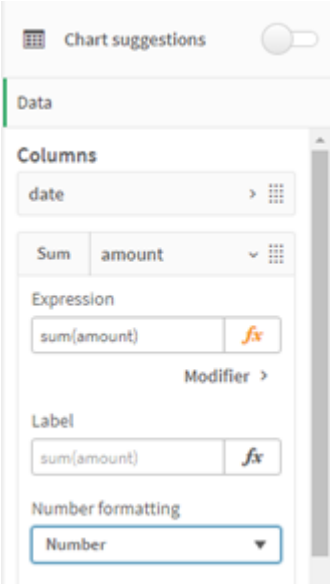


Tabela wynikowa

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

W tym przykładzie została użyta domyślna definicja `ThousandSep`, która jest ustawiona na format z przecinkiem (,). W tabeli wyników format pola kwoty sprawia, że grupy tysięczne są oddzielane przecinkami.

Przykład 2 – Zmiana zmiennej systemowej

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych z poprzedniego przykładu, który został załadowany do tabeli o nazwie `Transactions`.
- Modyfikacja definicji `ThousandSep`, na początku skryptu, aby grupy tysięczne oddzielać znakiem „*”. Jest to ekstremalny przykład, który ma na celu jedynie zademonstrować funkcjonalność opisywanej zmiennej.

Modyfikacja zastosowana w tym przykładzie jest ekstremalna i nie jest normalnie używana, ale pozwala zademonstrować funkcjonalność opisywanej zmiennej.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:`date`.
2. Dodaj następującą miarę:
   `=sum(amount)`
3. W panelu właściwości, w obszarz **Data** wybierz miarę.
4. W obszarze **Formatowanie liczb** wybierz **Niestandardowe**.

Tabela wynikowa

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

Na początku skryptu zmiennej systemowej `ThousandSep` została przypisana wartość '*'. W tabeli wyników został zastosowany format kwoty, w którym grupy tysięczne są oddzielane znakiem „*”.

Przykład 3 – Interpretacja tekstu

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Transactions`.
- Dane, które mają pole liczbowe w formacie tekstowym, z przecinkiem w roli separatora tysięcy.
- Użycie domyślnej zmiennej systemowej `ThousandSep`.

Skrypt ładowania

```
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'
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:`date`.
2. Dodaj następującą miarę:
   `=sum(amount)`
3. W panelu właściwości, w obszarz **Data** wybierz miarę.
4. W obszarze **Formatowanie liczb** wybierz **Liczba**.
5. Dodaj następującą miarę sprawdzającą, czy pole kwoty zawiera wartość liczbową:
   `=isnum(amount)`

Tabela wynikowa

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

date	=sum(amount)	=isnum(amount)
01/09/2022	37,565,817,436.00	-1
01/10/2022	3*454*343*566.00	-1

Po załadowaniu danych widzimy, że Qlik Sense zinterpretowała pole kwoty jako wartość liczbową dzięki temu, że dane te spełniają warunki zmiennej `ThousandSep`. Demonstruje to funkcja `isnum()`, która interpretuje każdy wpis jako -1, czyli `TRUE`.

W Qlik Sense wartość logiczna Prawda jest reprezentowana przez -1, a wartość Fałsz jest reprezentowana przez 0.

TimeFormat

Zdefiniowany format zastępuje format czasu w systemie operacyjnym (ustawienia regionalne).

Składnia:

```
TimeFormat
```

Przykład:

```
Set TimeFormat='hh:mm:ss';
```

TimestampFormat

Zdefiniowany format zastępuje formaty daty i godziny w systemie operacyjnym (ustawienia regionalne).

Składnia:

```
TimestampFormat
```

Przykład:

W poniższych przykładach wartość *1983-12-14T13:15:30Z* jest używana jako znacznik czasu, aby pokazać wyniki różnych instrukcji **SET TimestampFormat**. Używanym formatem daty jest **YYYYMMDD**, a formatem godziny jest **h:mm:ss TT**. Format daty jest określony w instrukcji **SET DateFormat**, a format godziny jest określony w instrukcji **SET TimeFormat** u góry skryptu ładowania danych.

Wyniki

Przykład	Wynik
`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

Przykłady: Skrypt ładowania

Przykład: Skrypt ładowania

W pierwszym skrypcie ładowania używana jest instrukcja *SET TimestampFormat='DD/MM/YYYY h:mm:ss [.fff] TT'*. W drugim skrypcie ładowania format znacznika czasu jest zmieniony na *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'*. Różne wyniki pokazują, w jaki sposób działa instrukcja **SET TimeFormat** z różnymi formatami danych czasu.

Poniższa tabela pokazuje zestaw danych używany w skryptach ładowania po tej tabeli. Druga kolumna tabeli pokazuje format każdego znacznika czasu w zestawie danych. Pierwsze pięć znaczników czasu jest zgodnych z regułami ISO 8601, ale szósty nie jest.

Zestaw danych

Tabela pokazująca używane dane czasu oraz format dla każdego znacznika czasu w zestawie danych.

transaction_timestamp	time data format
2018-08-30	YYYY-MM-DD
20180830T193614.857	YYYYMMDDhhmmss.sss
20180830T193614.857+0200	YYYYMMDDhhmmss.sss±hhmm
2018-09-16T12:30-02:00	YYYY-MM-DDhh:mm±hh:mm
2018-09-16T13:15:30Z	YYYY-MM-DDhh:mmZ
9/30/18 19:36:14	M/D/YY hh:mm:ss

W **edytorze ładowania danych** utwórz nową sekcję, a następnie dodaj skrypt przykładowy i uruchom go. Następnie dodaj do arkusza w swojej aplikacji co najmniej pola wyszczególnione w kolumnie wyników, aby wyświetlić wynik.

Skrypt ładowania

```
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 ];
```

Wyniki

Tabela aplikacji Qlik Sense pokazująca wyniki zmiennej interpretacji TimestampFormat używanej w skrypcie ładowania. Ostatni znacznik czasu w zestawie danych nie zwraca poprawnej daty.

transaction_id	transaction_timestamp	LogTimeStamp
3750	2018-08-30	2018-08-30 00:00:00
3751	20180830T193614.857	2018-08-30 19:36:14
3752	20180830T193614.857+0200	2018-08-30 17:36:14
3753	2018-09-16T12:30-02:00	2018-09-16 14:30:00
3754	2018-09-16T13:15:30Z	2018-09-16 13:15:30
3755	9/30/18 19:36:14	-

Następny skrypt ładowania używa tego samego zestawu danych. Jednak w nim użyto instrukcji *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'* w celu zapewnienia zgodności z formatem szóstego znacznika czasu, który jest w formacie innym niż ISO 8601.

W **edytorze ładowania danych** zastąp poprzedni skrypt przykładowy poniższym i uruchom go. Następnie dodaj do arkusza w swojej aplikacji co najmniej pola wyszczególnione w kolumnie wyników, aby wyświetlić wynik.

Skrypt ładowania

```
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 ];
```

Wyniki

Tabela aplikacji Qlik Sense pokazująca wyniki zmiennej interpretacji TimestampFormat używanej w skrypcie ładowania.

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

transaction_id	transaction_timestamp	LogTimeStamp
3753	2018-09-16T12:30-02:00	2018-09-16 14:30:00
3754	2018-09-16T13:15:30Z	2018-09-16 13:15:30
3755	9/30/18 19:36:14	2018-09-16 19:36:14

2.15 Zmienne Direct Discovery

Zmienne systemowe Direct Discovery

DirectCacheSeconds

Dla wyników zapytań Direct Discovery dotyczących wizualizacji można ustawić limit buforowania. Po osiągnięciu tego limitu Qlik Sense czyści bufor, gdy pojawią się nowe zapytania Direct Discovery. Qlik Sense skieruje zapytanie do źródła danych właściwego dla dokonanej selekcji i utworzy ponownie bufor na czas ograniczony limitem. Wynik każdej kombinacji selekcji będzie buforowany niezależnie. Oznacza to, że bufor jest odświeżany niezależnie przy każdej selekcji, czyli jedna selekcja odświeża bufor tylko dla pól poddanych selekcji, a druga selekcja odświeża bufor tylko pod względem poddanych jej pól. Jeśli druga selekcja obejmuje pola, które zostały odświeżone w pierwszej selekcji, nie są one ponownie aktualizowane w buforze, jeśli limit buforowania nie został osiągnięty.

Buforowanie Direct Discovery nie dotyczy wizualizacji typu **Tabela**. Każda selekcja w tabelach kieruje zapytanie do źródła danych.

Limit czasu musi być ustawiony w sekundach. Domyślny limit buforowania wynosi 1800 sekund (30 minut).

Wartość używana dla parametru **DirectCacheSeconds** jest wartością ustawioną w czasie wykonywania instrukcji **DIRECT QUERY**. Wartości tej nie można zmienić w środowisku wykonawczym.

Przykład:

```
SET DirectCacheSeconds=1800;
```

DirectConnectionMax

Do bazy danych można kierować asynchroniczne wywołania jednoczesne, korzystając z możliwości buforowania połączeń. Składnia skryptu ładowania przeznaczonego do ustawiania buforowania połączeń jest następująca:

```
SET DirectConnectionMax=10;
```

Ustawienie liczbowe określa maksymalną liczbę połączeń bazy danych, z których kod Direct Discovery może korzystać podczas aktualizowania arkusza. Ustawieniem domyślnym jest 1.

Z tej zmiennej należy korzystać ostrożnie. Jeśli nada się jej wartość większą niż 1, podczas połączenia z Microsoft SQL Server występują problemy.

DirectUnicodeStrings
Funkcja Direct Discovery może obsługiwać selekcję danych rozszerzonych Unicode przy użyciu standardowego formatu SQL dla rozszerzonych literałów ciągów znaków (N'<ciąg rozszerzony>'), co jest wymagane przez niektóre bazy danych (w tym przez SQL Server). Zastosowanie tej składni można włączyć dla Direct Discovery przy użyciu używanej w skryptach zmiennej **DirectUnicodeStrings**.

Ustawienie tej zmiennej na wartość „true" umożliwi użycie standardowego znacznika ANSI "N" dla znaków dwubajtowych przed literałami ciągów. Nie wszystkie bazy danych obsługują ten standard. Ustawieniem domyślnym jest „false".

DirectDistinctSupport
Jeśli w obiekcie Qlik Sense wybrana jest wartość pola **DIMENSION**, wówczas dla źródłowej bazy danych generowane jest zapytanie. Jeśli takie zapytanie wymaga grupowania, Direct Discovery korzysta ze słowa kluczowego **DISTINCT** w celu selekcji tylko wartości niepowtarzalnych. Niektóre bazy danych wymagają jednak słowa kluczowego **GROUP BY**. Ustaw wartość parametru **DirectDistinctSupport** na 'false', aby wygenerować **GROUP BY** zamiast **DISTINCT** w zapytaniach dotyczących wartości niepowtarzalnych.

```
SET DirectDistinctSupport='false';
```
Jeśli parametr DirectDistinctSupport ma wartość true, wówczas stosowane jest słowo kluczowe **DISTINCT**. Jeśli parametr ten nie zostanie ustawiony, wówczas domyślnym działaniem będzie zastosowanie słowa kluczowego **DISTINCT**.

DirectEnableSubquery
W przypadku scenariuszy obejmujących wiele tabel i dużą liczbę wierszy można generować w ramach zapytania SQL zapytania podrzędne, zamiast dużych klauzul IN. Aby włączyć tę funkcję, należy ustawić wartość funkcji **DirectEnableSubquery** na 'true'. Wartością domyślną jest 'false'.

> *Po włączeniu funkcji **DirectEnableSubquery** nie można już ładować tabel, które nie są w trybie Direct Discovery.*

```
SET DirectEnableSubquery='true';
```

Zmienne podziału zapytań na przedziały w oprogramowaniu Teradata

Podział zapytań na przedziały w oprogramowaniu Teradata to funkcja, która umożliwia aplikacjom korporacyjnym współpracę ze źródłową bazą danych Teradata w celu zapewnienia sprawniejszego ewidencjonowania, lepszej priorytetyzacji i efektywniejszego zarządzania procesami. Korzystając z podziału zapytań na przedziały można zawijać metadane, takie jak poświadczenia użytkownika, wokół zapytania.

Dostępne są dwie zmienne. Obie są ciągami znaków, które są oceniane i wysyłane do bazy danych.

SQLSessionPrefix
Ten ciąg jest wysyłany po nawiązaniu połączenia z bazą danych.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR SESSION;';
```
Jeśli **OSuser()** zwróci na przykład wynik *WA\sbt*, fragment ten będzie oceniany jako `SET QUERY_BAND = 'Who=WA\sbt;' FOR SESSION;` i taka postać zostanie wysłana do bazy danych po nawiązaniu połączenia.

SQLQueryPrefix
Ten ciąg jest wysyłany w przypadku każdego zapytania.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR TRANSACTION;';
```

Zmienne znakowe Direct Discovery

DirectFieldColumnDelimiter

Dla baz danych, które jako separatora pola wymagają znaku innego niż przecinek, można ustawić znak używany jako separator pola w instrukcjach **Direct Query**. Podany znak należy ująć w pojedyncze cudzysłowy w instrukcji **SET**.

```
SET DirectFieldColumnDelimiter= '|'
```

DirectStringQuoteChar

Możliwe jest określenie znaku, który będzie używany w celu cytowania ciągów w generowanych zapytaniach. Domyślnie jest to pojedynczy cudzysłów. Podany znak należy ująć w pojedyncze cudzysłowy w instrukcji **SET**.

```
SET DirectStringQuoteChar= '"';
```

DirectIdentifierQuoteStyle

W celu cytowania identyfikatorów w generowanych zapytaniach można zezwolić na stosowanie cudzysłowów w formacie innym niż ANSI. Aktualnie jedynym cudzysłowem w formacie innym niż ANSI jest GoogleBQ. Wartością domyślną jest ANSI. Dozwolony jest zapis literami wielkimi, literami małymi oraz literami o różnej wielkości (ANSI, ansi, Ansi).

```
SET DirectIdentifierQuoteStyle="GoogleBQ";
```

Na przykład cytowanie w formacie ANSI ma miejsce w następującej instrukcji **SELECT**:

```
SELECT [Quarter] FROM [qvTest].[sales] GROUP BY [Quarter]
```

Jeśli **DirectIdentifierQuoteStyle** ma wartość "GoogleBQ", wówczas w instrukcji **SELECT** cytowanie miałoby następującą postać:

```
SELECT [Quarter] FROM [qvTest.sales] GROUP BY [Quarter]
```

DirectIdentifierQuoteChar

Możliwe jest określenie znaku, który będzie kontrolował cytowanie identyfikatorów w generowanych zapytaniach. Może to być jeden znak (np. znak podwójnego cudzysłowu) albo dwa znaki (np. para nawiasów kwadratowych). Domyślnie jest to znak podwójnego cudzysłowu.

```
SET DirectIdentifierQuoteChar='[]';
SET DirectIdentifierQuoteChar='``';
SET DirectIdentifierQuoteChar=' ';
SET DirectIdentifierQuoteChar='""';
```

DirectTableBoxListThreshold

Jeśli pola Direct Discovery są używane w wizualizacji typu **Tabela**, wówczas ustawiona jest wartość progowa, która ogranicza liczbę wyświetlanych wierszy. Ustawieniem domyślnym jest 1000 wierszy. Aby zmienić domyślne ustawienie limitu, należy w skrypcie ładowania odpowiednio ustawić zmienną

DirectTableBoxListThreshold. Na przykład:

```
SET DirectTableBoxListThreshold=5000;
```

Ustawienie limitu obowiązuje tylko względem wizualizacji typu **Tabela**, która zawiera pola Direct Discovery. Wizualizacje typu **Tabela** zawierającej jedynie pola oparte na pamięci głównej nie podlegają ograniczeniu wartości określonym przez funkcję **DirectTableBoxListThreshold**.

W wizualizacji typu **Tabela** wyświetlane są pola dopiero wtedy, gdy selekcja obejmuje mniej rekordów niż wartość limitu.

Zmienne interpretacji liczb Direct Discovery

DirectMoneyDecimalSep

Zdefiniowany separator dziesiętny zastępuje symbol dziesiętny dotyczący waluty w instrukcji SQL wygenerowanej w celu ładowania danych przy użyciu Direct Discovery. Ten znak musi być zgodny ze znakiem użytym w funkcji **DirectMoneyFormat**.

Wartość domyślna to `'.'`

Przykład:

```
Set DirectMoneyDecimalSep='.';
```

DirectMoneyFormat

Zdefiniowany symbol zastępuje format waluty w instrukcji SQL wygenerowanej w celu ładowania danych przy użyciu funkcji Direct Discovery. Nie należy uwzględniać symbolu waluty w odniesieniu do separatora tysięcy.

Wartość domyślna to `'#.0000'`

Przykład:

```
Set DirectMoneyFormat='#.0000';
```

DirectTimeFormat

Zdefiniowany format czasu zastępuje format czasu w instrukcji SQL wygenerowanej w celu ładowania danych przy użyciu Direct Discovery.

Przykład:

```
Set DirectTimeFormat='hh:mm:ss';
```

DirectDateFormat

Zdefiniowany format daty zastępuje format daty w instrukcji SQL wygenerowanej w celu ładowania danych przy użyciu Direct Discovery.

Przykład:

```
Set DirectDateFormat='MM/DD/YYYY';
```

DirectTimeStampFormat

Zdefiniowany format zastępuje format daty i godziny w instrukcji SQL wygenerowanej w celu ładowania danych przy użyciu Direct Discovery.

Przykład:

```
Set DirectTimestampFormat='M/D/YY hh:mm:ss[.fff]';
```

2.16 Zmienne błędu

Po wykonaniu skryptu wartości wszystkich zmiennych zostaną zachowane. Pierwsza zmienna, ErrorMode, jest zmienną wejściową podawaną przez użytkownika, a ostatnie trzy są uzyskiwane z aplikacji Qlik Sense i zawierają informacje na temat błędów w skrypcie.

Przegląd zmiennych błędu

Każda zmienna jest opisana dalej po przeglądzie. Możesz także kliknąć nazwę zmiennej w składni, aby natychmiast uzyskać dostęp do szczegółowych informacji o tej konkretnej zmiennej.

Więcej informacji na temat zmiennych można znaleźć w pomocy online Qlik Sense.

ErrorMode

Ta zmienna błędu określa, jakie działanie ma podjąć program Qlik Sense po napotkaniu błędu podczas wykonywania skryptu.

```
ErrorMode
```

ScriptError

Ta zmienna błędu zwraca kod błędu ostatniej wykonanej instrukcji skryptu.

```
ScriptError
```

ScriptErrorCount

Ta zmienna błędu zwraca łączną liczbę instrukcji, które spowodowały błędy podczas bieżącego wykonywania skryptu. Ta zmienna jest zawsze resetowana do wartości 0 na początku wykonywania skryptu.

```
ScriptErrorCount
```

ScriptErrorList

Ta zmienna błędu będzie zawierać połączoną listę wszystkich błędów skryptu, które wystąpiły podczas ostatniego wykonywania skryptu. Każdy błąd jest oddzielony znakiem nowego wiersza.

```
ScriptErrorList
```

ErrorMode

Ta zmienna błędu określa, jakie działanie ma podjąć program Qlik Sense po napotkaniu błędu podczas wykonywania skryptu.

Składnia:

```
ErrorMode
```

Argumenty:

Argumenty

Argument	Opis
ErrorMode=1	Ustawienie domyślne. Wykonanie skryptu zostanie zatrzymane, a użytkownik otrzyma monit o podjęcie działania (tryb niewsadowy).
ErrorMode =0	Aplikacja Qlik Sense zignoruje błąd i będzie kontynuować wykonywanie skryptu w następnej instrukcji skryptu.
ErrorMode =2	Aplikacja Qlik Sense wyzwoli komunikat o błędzie „Wykonanie skryptu nie powiodło się...” natychmiast po wystąpieniu błędu, bez wcześniejszego monitowania użytkownika o podjęcie działania.

Przykład:

```
set ErrorMode=0;
```

ScriptError

Ta zmienna błędu zwraca kod błędu ostatniej wykonanej instrukcji skryptu.

Składnia:

```
ScriptError
```

Po każdej pomyślnie wykonanej instrukcji skryptu ta zmienna zostanie zresetowana do wartości 0. Jeśli wystąpi błąd, zostanie ona ustawiona na wewnętrzny kod błędu aplikacji Qlik Sense. Kody błędów to podwójne wartości składające się z elementu liczbowego i tekstowego. Istnieją następujące kody błędów:

Kody błędów skryptu

Kod błędu	Opis
0	Brak błędu. Tekst o wartości podwójnej jest pusty.
1	Błąd ogólny.
2	Błąd składni.
3	Błąd ogólny ODBC.
4	Błąd ogólny OLE DB.
5	Ogólny niestandardowy błąd bazy danych.
6	Błąd ogólny XML.
7	Błąd ogólny HTML.

Kod błędu	Opis
8	Plik nie został znaleziony.
9	Baza danych nie została znaleziona.
10	Tabela nie została znaleziona.
11	Pole nie zostało znalezione.
12	Plik ma nieprawidłowy format.
16	Błąd semantyczny.

Przykład:

```
set ErrorMode=0;

LOAD * from abc.qvf;

if ScriptError=8 then

exit script;

//no file;

end if
```

ScriptErrorCount

Ta zmienna błędu zwraca łączną liczbę instrukcji, które spowodowały błędy podczas bieżącego wykonywania skryptu. Ta zmienna jest zawsze resetowana do wartości 0 na początku wykonywania skryptu.

Składnia:

```
ScriptErrorCount
```

ScriptErrorList

Ta zmienna błędu będzie zawierać połączoną listę wszystkich błędów skryptu, które wystąpiły podczas ostatniego wykonywania skryptu. Każdy błąd jest oddzielony znakiem nowego wiersza.

Składnia:

```
ScriptErrorList
```

2 Wyrażenia skryptu

Wyrażeń można używać zarówno w instrukcjach **LOAD**, jak i w instrukcjach **SELECT**. Opisane tu reguły składniowe mają zastosowanie do instrukcji **LOAD**, ale nie do instrukcji **SELECT**, gdyż ta jest interpretowana przez sterownik ODBC, a nie przez aplikację Qlik Sense. Mimo to większość sterowników ODBC poprawnie interpretuje wiele z niżej opisanych funkcji.

Wyrażenia składają się z funkcji, pól i operatorów, połączonych w składni.

Wszystkie wyrażenia w skrypcie Qlik Sense zwracają liczbę lub ciąg znaków, stosownie do sytuacji. Funkcje logiczne i operatory logiczne zwracają 0 dla wartości False i -1 dla wartości True. Konwersje liczb na ciągi znaków i w odwrotnym kierunku są niejawne. Operatory i funkcje logiczne interpretują 0 jako wartość False, a wszystkie inne liczby jako wartość True.

Ogólna składnia wyrażenia wygląda następująco:

Składnia ogólna

Wyrażenie	Pola	Operator
expression ::= (constant	constant	|
expression ::= (constant	fieldref	|
expression ::= (constant	operator1 expression	|
expression ::= (constant	expression operator2 expression	|
expression ::= (constant	function	|
expression ::= (constant	(expression)	)

gdzie:

- **constant** to ciąg (tekst, data lub godzina) ujęty w pojedyncze cudzysłowy proste lub liczba. Stałe są zapisywane bez separatora tysięcy i z użyciem kropki jako separatora dziesiętnego.
- **fieldref** to nazwa pola załadowanej tabeli.
- **operator1** to operator jednoargumentowy, który dotyczy tylko jednego wyrażenia (znajdującego się na prawo od operatora).
- **operator2** to operator dwuargumentowy, który dotyczy wyrażeń znajdujących się po obu jego stronach.
- **function ::= functionname(parameters)**
- **parameters ::= expression { , expression }**

Liczba i typy parametrów nie są dowolne, zależą bowiem od używanej funkcji.

Wyrażenia i funkcje można dowolnie zagnieżdżać – dopóki wyrażenie zwraca wartość dającą się zinterpretować, aplikacja Qlik Sense nie będzie zgłaszać żadnych komunikatów o błędach.

3 Wyrażenia wykresu

Wyrażenie wykresu (wizualizacji) to kombinacja funkcji, pól i operatorów matematycznych (+ * / =) oraz innych miar. Składnia wiersza poleceń aplikacji Qlik Sense i składnia skryptu są opisywane w tzw. notacji Backusa-Naura lub notacji BNF. Wyrażenia służą przetwarzaniu danych w aplikacji w celu osiągnięcia rezultatu widocznego na wizualizacji. Zastosowanie wyrażeń nie jest ograniczone do miar. Z wyrażeniami używanymi jako tytuły, podtytuły, stopki, a nawet wymiary można tworzyć bardziej dynamiczne i zaawansowane wizualizacje.

Oznacza to na przykład, że tytuł wizualizacji nie musi być tekstem statycznym, ale może być oparty na wyrażeniu i zmieniać się w zależności od dokonanych selekcji.

Szczegółowe informacje o funkcjach skryptu i funkcjach wykresu zawiera dokumentacja Składnia skryptów i funkcje wykresów.

3.1 Określenie zakresu agregacji

Istnieją zazwyczaj dwa czynniki, które łącznie określają, które rekordy mają definiować wartość agregacji w wyrażeniu. W przypadku wizualizacji są to:

- Wartość wymiaru (agregacji w wyrażeniu wykresu)
- Selekcje

Łącznie czynniki te określają zakres agregacji. Mogą zdarzyć się sytuacje, w których obliczenie ma ignorować daną selekcję lub dany wymiar. W przypadku funkcji wykresu można to osiągnąć przez zastosowanie kwalifikatora TOTAL i/lub analizy zestawów.

Agregacja: Metoda i opis

Metoda	Opis
Kwalifikator TOTAL	Zastosowanie kwalifikatora total w ramach funkcji agregacji powoduje zignorowanie wartości wymiaru. Agregacja taka zostanie wykonana dla wszystkich możliwych wartości pól. Po kwalifikatorze **TOTAL** może następować lista zawierająca co najmniej jedną nazwę pola w nawiasach trójkątnych. Te nazwy pól powinny być podzbiorem zmiennych wymiaru wykresu. W tym przypadku obliczenie jest wykonywane z pominięciem wszystkich zmiennych wymiaru wykresu z wyjątkiem zmiennych z listy, tj. dla każdej kombinacji wartości pól dla pól wymiaru z listy jest zwracana jedna wartość. Lista może zawierać także pola, które aktualnie nie są wymiarem na wykresie. Jest to użyteczne w przypadku wymiarów grupowych, gdy pola wymiarów nie są niezmienne. Lista zawiera wszystkie zmienne z grupy, co powoduje, że funkcja działa w przypadku zmiany poziomu drążenia.

Metoda	Opis
Analiza zestawów	Zastosowanie analizy zestawów w ramach agregacji powoduje zastąpienie danej selekcji. Agregacja taka zostanie wykonana dla wszystkich wartości w podziale na wymiary.
Kwalifikator TOTAL i analiza zestawów	Zastosowanie kwalifikatora **TOTAL** i analizy zestawów w ramach agregacji powoduje zastąpienie danej selekcji i zignorowanie wymiarów.

Kwalifikator ALL	Zastosowanie kwalifikatora **ALL** w ramach agregacji powoduje zignorowanie danej selekcji i wymiarów. Aby wykonać podobną operację, należy skorzystać z instrukcji analizy zestawów {1} i kwalifikatora **TOTAL**:
	`=sum(All Sales)`
	`=sum({1} Total Sales)`

Przykład: Kwalifikator TOTAL

Na poniższym przykładzie przedstawiono sposób zastosowania kwalifikatora TOTAL w celu obliczenia udziału względnego. Przy założeniu, że wybrano kwartał Q2, zastosowanie kwalifikatora TOTAL pozwala na obliczenie sumy wszystkich wartości bez uwzględnienia wymiarów.

Przykład: Kwalifikator Total

Year	Quarter	Sum(Amount)	Sum(TOTAL Amount)	Sum(Amount)/Sum(TOTAL Amount)
		3000	3000	100%
2012	Q2	1700	3000	56,7%
2013	Q2	1300	3000	43,3%

*Aby pokazać te liczby jako procenty, w panelu właściwości – dla miary, którą chcesz pokazać jako wartość procentową – w obszarze **Number formatting** wybierz opcję **Number**, a w obszarze **Formatting** wybierz opcję **Simple** i jeden z formatów %.*

Przykład: Analiza zestawów

Na poniższym przykładzie przedstawiono sposób zastosowania analizy zestawów w celu porównania zestawów danych przed dokonaniem jakiejkolwiek selekcji. Przy założeniu, że wybrano kwartał Q2, zastosowanie analizy zestawów z definicją zestawu {1} pozwala na obliczenie sumy wszystkich wartości bez uwzględnienia jakichkolwiek selekcji, ale w podziale na wymiary.

Przykład: Analiza zestawów

Year	Quarter	Sum(Amount)	Sum({1} Amount)	Sum(Amount)/Sum({1} Amount)
		3000	10800	27,8%
2012	Q1	0	1100	0%
2012	Q3	0	1400	0%
2012	Q4	0	1800	0%
2012	Q2	1700	1700	100%
2013	Q1	0	1000	0%
2013	Q3	0	1100	0%
2013	Q4	0	1400	0%
2013	Q2	1300	1300	100%

Przykład: Kwalifikator TOTAL i analiza zestawów

Na poniższym przykładzie przedstawiono sposób łącznego zastosowania analizy zestawów i kwalifikatora TOTAL w celu porównania zestawów danych przed dokonaniem jakichkolwiek selekcji i w ramach wszystkich wymiarów. Przy założeniu, że wybrano kwartał Q2, zastosowanie analizy zestawów z definicją zestawu {1} i kwalifikatora TOTAL pozwala na obliczenie sumy wszystkich wartości bez uwzględnienia jakichkolwiek selekcji i wymiarów.

Przykład: Kwalifikator TOTAL i analiza zestawów

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%

Dane zastosowane w przykładach:

```
AggregationScope: LOAD * inline [ Year Quarter Amount 2012 Q1 1100 2012 Q2 1700 2012 Q3 1400
2012 Q4 1800 2013 Q1 1000 2013 Q2 1300 2013 Q3 1100 2013 Q4 1400] (delimiter is ' ');
```

3.2 Analiza zestawów

Dokonując wyboru w aplikacji, definiujesz podzestaw rekordów w danych. Funkcje agregacji, takie jak `Sum()`, `Max()`, `Min()`, `Avg()` i `Count()` są obliczane na podstawie tego podzestawu.

Innymi słowy, Twój wybór definiuje zakres agregacji i zestaw rekordów, na których wykonywane są obliczenia.

W ramach analizy zestawów można zdefiniować zakres, który jest inny niż zestaw rekordów zdefiniowany przez bieżący wybór. Ten nowy zakres można także uznać za wybór alternatywny.

Może się to przydać do porównania bieżącego wyboru z konkretną wartością, na przykład wartością z zeszłego roku lub udziałem w rynku globalnym.

Wyrażenia zestawu

Wyrażenia zestawów mogą być używane wewnątrz i na zewnątrz funkcji agregacji oraz są ujęte w nawiasy klamrowe.

Przykład: Wewnętrzne wyrażenie zestawu

```
Sum( {$<Year={2021}>} Sales )
```

Przykład: Zewnętrzne wyrażenie zestawu

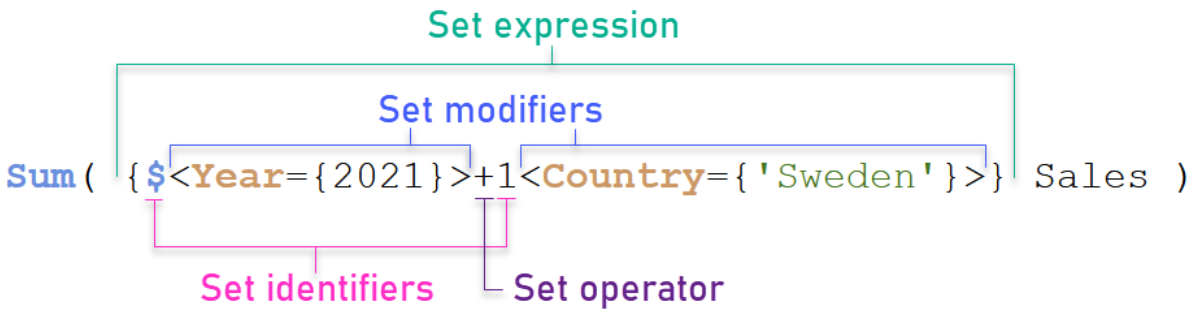
```
{<Year={2021}>} Sum(Sales) / Count(distinct Customer)
```

Wyrażenie zestawu składa się z kombinacji następujących elementów:

- **Identifiers**. Identyfikator zestawu reprezentuje wybór zdefiniowany gdzie indziej. Reprezentuje również określony zestaw rekordów w danych. Może to być bieżący wybór, wybór z zakładki lub wybór ze stanu alternatywnego. Proste wyrażenie zestawu składa się z jednego identyfikatora, na przykład znaku dolara {$}, który oznacza wszystkie rekordy w bieżącej selekcji.
  Przykłady: `$`, `1`, `BookMark1`, `State2`
- **Operators**. Operator zestawu może być używany do tworzenia sum, różnic lub części wspólnych między różnymi identyfikatorami zestawów. W ten sposób można utworzyć podzestaw lub nadzestaw wyborów definiowanych przez identyfikatory zestawów.
  Przykłady: `+`, `-`, `*`, `/`
- **Modifiers**. Do wyrażenia zestawu można dodać modyfikator zestawu, aby zmienić wybór. Modyfikator może być również używany samodzielnie, a następnie zmodyfikuje on domyślny identyfikator. Modyfikator musi być ujęty w nawiasy kątowe: `<…>`.
  Przykłady: `<Year={2020}>`, `<Supplier={ACME}>`

Elementy są łączone w celu tworzenia wyrażeń zestawu.

Elementy w wyrażeniu zestawu



Na przykład powyższe wyrażenie zestawu jest zbudowane na podstawie agregacji `Sum(Sales)`.

Pierwszy operand zwraca sprzedaż za rok `2021` dla bieżącego wyboru, na co wskazuje identyfikator zestawu `$` i modyfikator zawierający wybór roku `2021`. Drugi operand zwraca `Sales` dla `Sweden` i ignoruje bieżące zaznaczenie, które jest wskazywane przez identyfikator zestawu `1`.

Na koniec wyrażenie zwraca zestaw zawierający wszystkie rekordy, które należą do któregokolwiek z dwóch operandów zestawu, jak wskazuje to operator zestawu +.

Przykłady

Przykłady łączące powyższe elementy wyrażenia zestawu są dostępne w następujących tematach:

Zestawy naturalne

Wyrażenie zestawu reprezentuje zazwyczaj zarówno zestaw rekordów w modelu danych, jak i wybór definiujący ten podzestaw danych. W tym przypadku zestaw nazywamy zestawem naturalnym.

Identyfikatory zestawów, z modyfikatorami zestawów lub bez nich, zawsze reprezentują zestawy naturalne.

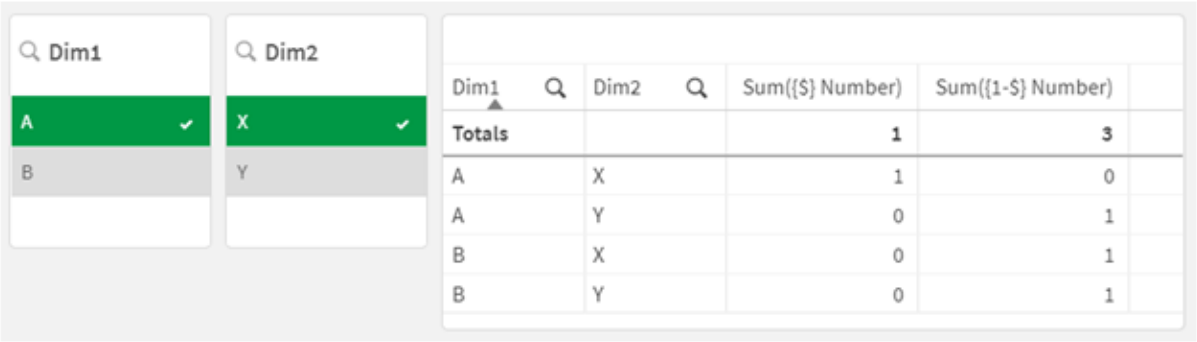
Jednak wyrażenie zestawu używające operatorów zestawu również reprezentuje podzestaw rekordów, ale zasadniczo nadal nie można go opisać za pomocą wyboru wartości pól. Takie wyrażenie jest zestawem nienaturalnym.

Na przykład zestaw określony przez `{1-$}` nie może zawsze być zdefiniowany przez wybór. Nie jest więc zestawem naturalnym. Można to pokazać, ładując następujące dane, dodając je do tabeli, a następnie dokonując wyborów za pomocą paneli filtrowania.

```
Load * Inline
[Dim1, Dim2, Number
A, X, 1
A, Y, 1
B, X, 1
B, Y, 1];
```

Dokonując wyborów dla `Dim1` i `Dim2`, otrzymujesz widok pokazany w poniższej tabeli.

Tabela z zestawami naturalnymi i nienaturalnymi



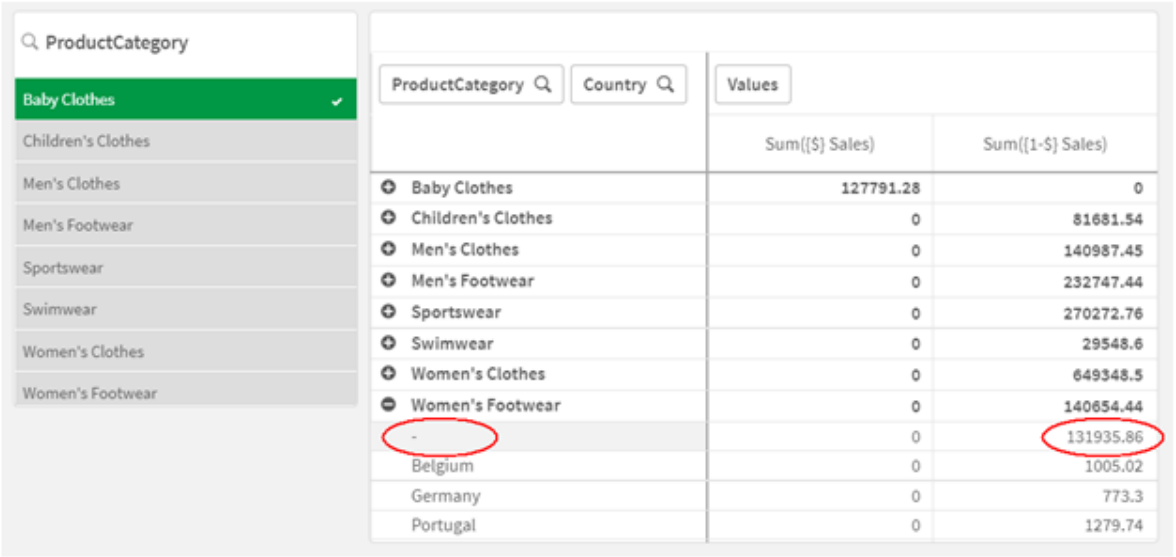
Dim1	Dim2	Sum({$} Number)	Sum({1-$} Number)
Totals		**1**	**3**
A	X	1	0
A	Y	0	1
B	X	0	1
B	Y	0	1

Wyrażenie zestawu w pierwszej mierze wykorzystuje zestaw naturalny: odpowiada wyborowi dokonanemu przez `{$}`.

Druga miara jest inna. Wykorzystuje ona `{1-$}`. Nie można dokonać wyboru odpowiadającego temu zestawowi, więc jest to zestaw nienaturalny.

To rozróżnienie ma szereg konsekwencji:

- Modyfikatory zestawów można stosować tylko do identyfikatorów zestawów. Nie można ich stosować do dowolnego wyrażenia zestawu. Na przykład nie można użyć wyrażenia zestawu takiego jak:
  `{ (BM01 * BM02) <Field={x,y}> }`
  W tym przypadku zwykłe (okrągłe) nawiasy oznaczają, że część wspólna `BM01` i `BM02` powinna zostać oceniona przed zastosowaniem modyfikatora zestawu. To dlatego, że nie ma zestawu elementów, który można zmodyfikować.
- W funkcjach elementów `P()` i `E()` nie można używać zestawów nienaturalnych. Funkcje te zwracają zestaw elementów, ale nie można wydedukować zestawu elementów z zestawu nienaturalnego.
- Miarę wykorzystującą zestaw nienaturalny nie zawsze można przypisać do właściwej wartości wymiaru, jeśli model danych zawiera wiele tabel. Na przykład na poniższym wykresie niektóre wykluczone wartości sprzedaży są przypisane do prawidłowego Country, podczas gdy inne mają NULL jako Country.
  Wykres z zestawem nienaturalnym



  Prawidłowe przypisanie zależy od modelu danych. W takim przypadku wartości nie można przypisać, jeśli dotyczy kraju wykluczonego przez wybór.

Identyfikator	Opis
1	Reprezentuje cały zestaw wszystkich rekordów w aplikacji niezależnie od dokonanych selekcji.
$	Reprezentuje rekordy bieżącej selekcji. To wyrażenie zestawu **{$}** jest zatem równoważne niepodaniu żadnego wyrażenia zestawu.
$1	Reprezentuje poprzednią selekcję. $2 reprezentuje selekcję przed poprzednią selekcją itd.
$_1	Reprezentuje następną (kolejną) selekcję. $_2 reprezentuje selekcję po następnej selekcji itd.

Identyfikator	Opis
BM01	Można zastosować dowolny identyfikator zakładki lub dowolną nazwę zakładki.
MyAltState	Można przywołać selekcje dokonane w stanie alternatywnym przez podanie nazwy tego stanu.

Przykład	Wynik
sum ({1} Sales)	Zwraca wartość łącznej sprzedaży dla danej aplikacji. Ignorowane są selekcje, ale nie wymiary.
sum ({$} Sales)	Zwraca wartość sprzedaży dla bieżącej selekcji (zwraca to samo co funkcja sum (Sales)).
sum ({$1} Sales)	Zwraca wartość sprzedaży dla poprzedniej selekcji.
sum ({BM01} Sales)	Zwraca wartość sprzedaży dla zakładki o nazwie *BM01*.

Przykład	Wynik
sum({$<OrderDate = DeliveryDate>} Sales)	Zwraca wartość sprzedaży dla bieżącej selekcji, gdzie OrderDate = DeliveryDate.
sum({1<Region = {US}>} Sales)	Zwraca wartość sprzedaży dla regionu US, ignorując bieżącą selekcję.
sum({$<Region = >} Sales)	Zwraca wartość sprzedaży dla tej selekcji, ale usuwa selekcję w wymiarze *Region*.
sum({<Region = >} Sales)	Zwraca tę samą wartość co w przykładzie powyżej. Jeśli modyfikowany zestaw zostanie pominięty, zakładany jest parametr $.
sum({$<Year={2000}, Region={"U*"}>} Sales)	Zwraca wartość sprzedaży dla bieżącej selekcji, ale z nowymi selekcjami w wymiarach *Year* i *Region*.

Identyfikatory zestawu

Identyfikator zestawu reprezentuje zestaw rekordów w danych: albo wszystkie dane, albo ich podzestaw. Jest to zestaw rekordów zdefiniowany przez wybór. Może to być bieżący wybór, wszystkie dane (bez wyboru), wybór z zakładki lub wybór ze stanu alternatywnego.

W przykładzie `Sum( {$<Year = {2009}>} Sales )` identyfikatorem jest znak dolara: `$`. Reprezentuje on bieżący wybór. Reprezentuje również wszystkie możliwe rekordy. Zestaw ten może zostać następnie zmieniony przez modyfikator zawarty w tym wyrażeniu zestawu: zostaje dodany wybór `2009` w `Year`.

W bardziej złożonym wyrażeniu zestawu dwa identyfikatory mogą być używane razem z operatorem, aby utworzyć sumę, różnicę lub część wspólną dwóch zestawów rekordów.

W poniższej tabeli przedstawiono kilka typowych identyfikatorów.

Przykłady ze wspólnymi identyfikatorami

Identyfikator	Opis
`1`	Reprezentuje cały zestaw wszystkich rekordów w aplikacji niezależnie od dokonanych selekcji.
`$`	Reprezentuje rekordy bieżącego wyboru w stanie domyślnym. Wyrażenie zestawu `{$}` jest zatem równoważne niepodaniu żadnego wyrażenia zestawu.
`$1`	Reprezentuje poprzedni wybór w stanie domyślnym. `$2` reprezentuje przedostatni wybór itd.
`$_1`	Reprezentuje następny wybór. `$_2` reprezentuje drugi w kolejności wybór i tak dalej.
`BM01`	Można zastosować dowolny identyfikator zakładki lub dowolną nazwę zakładki.
`AltState`	Stan alternatywny można przywołać przez podanie nazwy tego stanu.
`AltState::BM01`	Zakładka zawiera wybory wszystkich stanów, a do konkretnej zakładki można się odwoływać, kwalifikując jej nazwę.

W następującej tabeli przedstawiono przykłady z różnymi identyfikatorami.

Przykłady z różnymi identyfikatorami

Przykład	Wynik
`Sum ({1} Sales)`	Zwraca wartość łącznej sprzedaży dla danej aplikacji. Ignorowane są selekcje, ale nie wymiary.
`Sum ({$} Sales)`	Zwraca wartość sprzedaży dla bieżącej selekcji (zwraca to samo co funkcja `Sum(Sales)`).
`Sum ({$1} Sales)`	Zwraca wartość sprzedaży dla poprzedniej selekcji.
`Sum ({BM01} Sales)`	Zwraca wartość sprzedaży dla zakładki o nazwie `BM01`.

Operatory zestawów

Operatory zestawów służą do uwzględniania, wykluczania albo przecinania zestawów danych. Wszystkie operatory korzystają z zestawów jako operandów i zwracają zestaw jako wynik.

Operatorów zestawów można używać w dwóch sytuacjach:

- Aby wykonać operację zestawu na identyfikatorach zestawów reprezentujących zestawy rekordów w danych.
- Aby wykonać operację zestawu na zestawach elementów, na wartościach pól lub wewnątrz modyfikatora zestawu.

W poniższej tabeli przedstawiono operatory, które mogą być stosowane w wyrażeniach zestawu.

Operatory

Operator	Opis
+	Suma. Operacja dwuargumentowa zwracająca zestaw zawierający wszystkie rekordy lub elementy, które należą do któregokolwiek z dwóch zestawów wejściowych.
-	Wykluczenie. Operacja dwuargumentowa zwracająca zestaw zawierający wszystkie rekordy lub elementy, które należą do pierwszego, a nie należą do drugiego z dwóch zestawów wejściowych. W przypadku użycia jako operator jednoargumentowy zwraca dopełnienie zestawu wejściowego.
*	Część wspólna. Operacja dwuargumentowa zwracająca zestaw zawierający wszystkie rekordy lub elementy, które należą do obu zestawów wejściowych.
/	Różnica symetryczna (XOR). Operacja dwuargumentowa zwracająca zestaw zawierający wszystkie rekordy lub elementy, które należą do jednego, ale nie obu zestawów wejściowych.

W poniższej tabeli przedstawiono przykłady z operatorami.

Przykłady z operatorami

Przykład	Wynik
`Sum ({1-$} Sales)`	Zwraca wartość sprzedaży dla wszystkiego, co zostało wykluczone przez bieżący wybór.
`Sum ({$*BM01} Sales)`	Zwraca wartość sprzedaży dla części wspólnej wyboru i zakładki `BM01`.
`Sum ({-($+BM01)} Sales)`	Zwraca wartość sprzedaży wykluczoną przez wybór i zakładkę `BM01`.
`Sum ({$<Year={2009}>+1<Country={'Sweden'}>} Sales)`	Zwraca wartość sprzedaży z roku 2009 powiązaną z bieżącymi wyborami i dodaje pełny zestaw danych powiązanych z krajem `Sweden` dla wszystkich lat.
`Sum ({$<Country={"S*"}+{"*land"}>} Sales)`	Zwraca sprzedaż dla krajów, które zaczynają się na `S` lub kończą na `land`.

Modyfikatory zestawów

Wyrażenia zestawu służą do definiowania zakresu obliczenia. Centralną częścią wyrażenia zestawu jest modyfikator zestawu, który określa wybór. Służy do modyfikowania wyboru użytkownika lub wyboru w identyfikatorze zestawu, a wynik określa nowy zakres obliczeń.

Modyfikator ustawień składa się z jednej lub więcej nazw pól, po których następuje wybór, który ma zostać dokonany na danym polu. Modyfikator jest ujęty w nawiasy kątowe: `< >`

Na przykład:

- `Sum ( {$<Year = {2015}>} Sales )`
- `Count ( {1<Country = {Germany}>} distinct OrderID )`
- `Sum ( {$<Year = {2015}, Country = {Germany}>} Sales )`

Zestawy elementów

Zestaw elementów można zdefiniować, używając następujących sposobów:

- Lista wartości
- Wyszukiwanie
- Odwołanie do innego pola
- Funkcja zestawu

Jeśli definicja zestawu elementów zostanie pominięta, modyfikator zestawu wyczyści wszystkie wybory w tym polu. Na przykład:

```
Sum( {$<Year = >} Sales )
```

Przykłady: Wyrażenia wykresu do modyfikatorów zestawów na podstawie zestawów elementów

Przykłady – wyrażenia wykresu

Skrypt ładowania

Załaduj następujące dane w edytorze ładowania danych jako ładowanie wbudowane, aby utworzyć poniższe przykłady wyrażeń wykresu:

```
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
];
```

Wyrażenia wykresu

Utwórz tabelę w arkuszu Qlik Sense z następującymi wyrażeniami wykresu

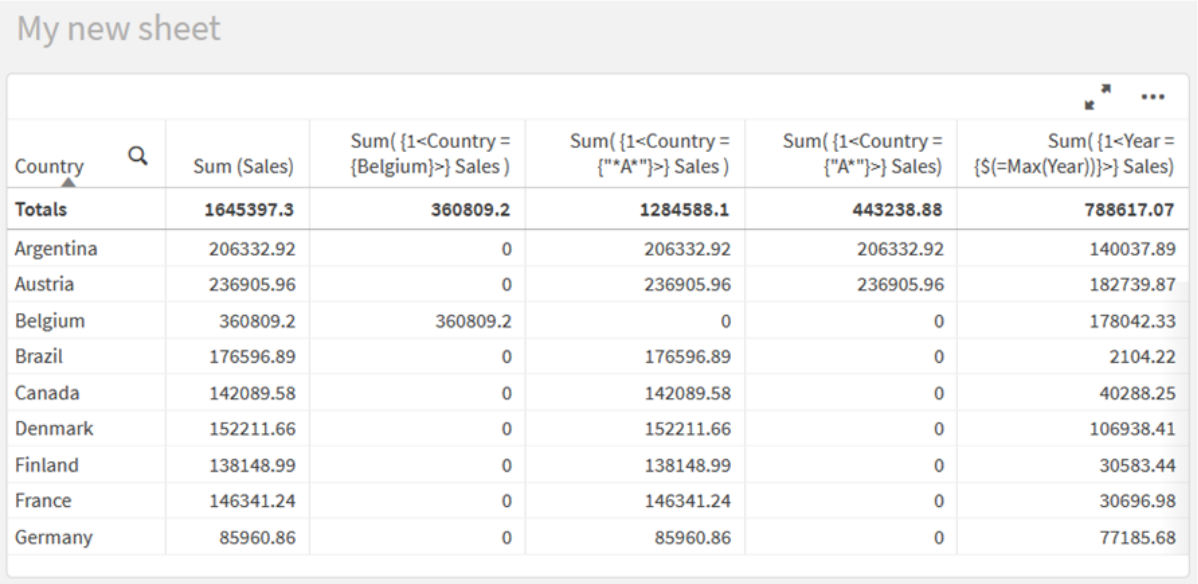
Tabela – modyfikatory zestawu oparte na zestawach elementów

Kraj	Sum(Sales)	Sum ({1<Country= {Belgium}>} Sales)	Sum ({1<Country= {"*A*"}>} Sales)	Sum ({1<Country= {"A*"}>} Sales)	Sum ({1<Year= {$(=Max (Year))}>} Sales)
Sumy	1645397.3	360809.2	1284588.1	443238.88	788617.07
Argentyna	206332.92	0	206332.92	206332.92	140037.89
Austria	236905.96	0	236905.96	236905.96	182739.87
Belgia	360809.2	360809.2	0	0	178042.33
Brazylia	176596.89	0	176596.89	0	2104.22
Kanada	142089.58	0	142089.58	0	40288.25
Dania	152211.66	0	152211.66	0	106938.41
Finlandia	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

Objaśnienie

- Wymiary:
  - `Country`
- Miary:
  - `Sum(Sales)`
    Suma `Sales` bez wyrażenia zestawu.
  - `Sum({1<Country={Belgium}>}Sales)`
    Wybierz `Belgium`, a następnie zsumuj odpowiednie `Sales`.
  - `Sum({1<Country={"*A*"}>}Sales)`
    Wybierz wszystkie kraje, które mają w nazwie `A`, a następnie zsumuj odpowiednie `Sales`.
  - `Sum({1<Country={"A*"}>}Sales)`
    Wybierz wszystkie kraje, które zaczynają się od `A`, a następnie zsumuj odpowiednie `Sales`.
  - `Sum({1<Year={$(=Max(Year))}>}Sales)`
    Oblicz `Max(Year)`, czyli `2015`, a następnie zsumuj odpowiednie `Sales`.

Modyfikatory zestawów oparte na zestawach elementów


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

Lista wartości

Najbardziej typowym przykładem zestawu elementów jest zestaw oparty na liście wartości pól ujętych w nawiasy klamrowe. Na przykład:

- `{$<Country = {Canada, Germany, Singapore}>}`
- `{$<Year = {2015, 2016}>}`

Wewnętrzne nawiasy klamrowe definiują zestaw elementów. Poszczególne wartości są rozdzielane przecinkami.

Cudzysłowy i rozróżnianie wielkości liter

Jeśli wartości zawierają spacje lub znaki specjalne, należy je umieścić w cudzysłowie. Pojedyncze cudzysłowy implikują dosłowne dopasowanie z rozróżnianiem wielkości liter, z pojedynczą wartością pola. Podwójne cudzysłowy oznaczają dopasowanie bez rozróżniania wielkości liter z co najmniej jedną wartością pola. Na przykład:

- `<Country = {'New Zealand'}>`
  Dopasowanie tylko do `New Zealand`.
- `<Country = {"New Zealand"}>`
  Dopasowanie do `New Zealand`, `NEW ZEALAND` i `new zealand`.

Daty muszą być ujęte w cudzysłów i zgodne z formatem daty w danym polu. Na przykład:

- `<ISO_Date = {'2021-12-31'}>`
- `<US_Date = {'12/31/2021'}>`
- `<UK_Date = {'31/12/2021'}>`

Cudzysłowy podwójne można zastąpić nawiasami kwadratowymi lub akcentami słabymi.

Wyszukiwania

Zestawy elementów można również tworzyć przez wyszukiwanie. Na przykład:

- `<Country = {"C*"}>`
- `<Ingredient = {"*garlic*"}>`
- `<Year = {">2015"}>`
- `<Date = {">12/31/2015"}>`

W wyszukiwaniach tekstowych można używać symboli wieloznacznych: Gwiazdka (`*`) reprezentuje dowolną liczbę znaków, a znak zapytania (`?`) reprezentuje pojedynczy znak. Do definiowania wyszukiwań liczbowych można używać operatorów relacyjnych.

Do wyszukiwania zawsze należy używać cudzysłowów. W wyszukiwaniach wielkość liter nie jest rozróżniana.

Rozszerzenia przez znak dolara

Rozszerzenia przez znak dolara są potrzebne, jeśli chcesz użyć obliczeń w swoim zestawie elementów. Jeśli na przykład chcesz spojrzeć tylko na ostatni możliwy rok, możesz użyć:

```
<Year = {$(=Max(Year))}>
```

Wybrane wartości w innych polach

Modyfikatory mogą być oparte na wybranych wartościach innego pola. Na przykład:

```
<OrderDate = DeliveryDate>
```

Taki modyfikator spowoduje pobranie wybranych wartości z pola `DeliveryDate` i zastosowanie ich jako selekcji w polu `OrderDate`. Jeśli odrębnych wartości jest wiele (więcej niż kilkaset), taka operacja bardzo obciąża procesor i nie jest zalecana.

Funkcje zestawu elementów

Zestaw elementów może też być oparty na funkcjach zestawu `P()` (wartości możliwe) i `E()` (wartości wykluczone).

Jeżeli na przykład chcesz wybrać kraje, w których jest lub był sprzedawany produkt `Cap`, możesz użyć:

```
<Country = P({1<Product={Cap}>} Country)>
```

Podobnie jeżeli chcesz wybrać kraje, w których produkt `Cap` nie był sprzedawany, możesz użyć:

```
<Country = E({1<Product={Cap}>} Country)>
```

Modyfikatory zestawów z wyszukiwaniami

Możesz tworzyć zestawy elementów przez wyszukiwanie z modyfikatorami zestawów.

Na przykład:

- `<Country = {"C*"}>`
- `<Year = {">2015"}>`
- `<Ingredient = {"*garlic*"}>`

Wyszukiwania powinny być zawsze ujęte w cudzysłowy, nawiasy kwadratowe lub akcenty słabe. Możesz użyć listy zawierającej kombinację ciągów literałów (pojedyncze cudzysłowy) i wyszukiwań (podwójne cudzysłowy). Na przykład:

```
<Product = {'Nut', "*Bolt", Washer}>
```

Wyszukiwania tekstowe

W wyszukiwaniach tekstowych można używać symboli wieloznacznych i innych:

- Gwiazdka (*) oznacza dowolną liczbę znaków.
- Znak zapytania (?) oznacza pojedynczy znak.
- Akcent cyrkumfleksowy (^) oznacza początek słowa.

Na przykład:

- `<Country = {"C*", "*land"}>`
  Dopasuj wszystkie kraje zaczynające się na `C` lub kończące na `land`.
- `<Country = {"*^z*"}>`
  Spowoduje to dopasowanie do wszystkich krajów mających w nazwie słowo zaczynające się na `z`, takich jak `New Zealand`.

Wyszukiwania liczbowe

Możesz prowadzić wyszukiwania liczbowe, korzystając z tych operatorów relacji: `>`, `>=`, `<`, `<=`

Wyszukiwanie liczbowe zawsze zaczyna się od jednego z tych operatorów. Na przykład:

- `<Year = {">2015"}>`
  Dopasowanie do roku 2016 i kolejnych lat.
- `<Date = {">=1/1/2015<1/1/2016"}>`
  Dopasowanie do wszystkich dat w 2015 roku. Zwróć uwagę na składnię do opisu zakresu czasu między dwiema datami. Format daty musi odpowiadać formatowi daty w danym polu.

Wyszukiwania według wyrażenia

Do bardziej zaawansowanych wyszukiwań możesz użyć wyszukiwania według wyrażenia. Agregacja jest obliczana dla każdej wartości pola w polu wyszukiwania. Zostaną wybrane wszystkie wartości, dla których wyrażenie wyszukiwania zwraca wartość Prawda.

Wyszukiwanie według wyrażenia zawsze zaczyna się znakiem równości: `=`

Na przykład:

```
<Customer = {"=Sum(Sales)>1000"}>
```

Spowoduje to zwrócenie wszystkich klientów o wartości sprzedaży większej niż 1000. `Sum(Sales)` oblicza się na podstawie bieżącego wyboru. Oznacza to, że jeśli masz wybór w innym polu, takim jak pole `Product`, otrzymasz klientów, którzy spełnili warunek sprzedaży tylko w przypadku wybranych produktów.

Jeśli chcesz, aby warunek był niezależny od wyboru, musisz użyć analizy zestawu wewnątrz ciągu wyszukiwania. Na przykład:

```
<Customer = {"=Sum({1} Sales)>1000"}>
```

Wyrażenia po znaku równości będą interpretowane jako wartość logiczna. Oznacza to, że jeśli zwrócona zostanie inna wartość, każda niezerowa liczba zostanie zinterpretowana jako prawda, podczas gdy zero i ciągi znaków zostaną zinterpretowane jako fałsz.

Quotes

Użyj znaków cudzysłowu, gdy ciągi wyszukiwania zawierają spacje albo znaki specjalne. Pojedyncze cudzysłowy implikują dosłowne dopasowanie z rozróżnianiem wielkości liter, z pojedynczą wartością pola. Podwójne cudzysłowy implikują dopasowanie bez rozróżniania wielkości liter, potencjalnie z wieloma wartościami pola.

Na przykład:

- `<Country = {'New Zealand'}>`
  Dopasowanie tylko do `New Zealand`.
- `<Country = {"New Zealand"}>`
  Dopasowanie do `New Zealand`, `NEW ZEALAND` i `new zealand`

Cudzysłowy podwójne można zastąpić nawiasami kwadratowymi lub akcentami słabymi.

> *W poprzednich wersjach Qlik Sense cudzysłowy pojedyncze nie były odróżniane od podwójnych i wszystkie ciągi w cudzysłowach były traktowane jak wyszukiwania. W celu zachowania zgodności z poprzednimi wersjami aplikacje utworzone w starszych wersjach Qlik Sense nadal będą działać tak samo, jak w poprzednich wersjach. Aplikacje utworzone w Qlik Sense w wersji z listopada 2017 lub w nowszych wersjach uwzględniają różnice między tymi dwoma typami cudzysłowów.*

Przykłady: Wyrażenia wykresu dla modyfikatorów zestawu z wyszukiwaniami

Przykłady – wyrażenia wykresu

Skrypt ładowania

Załaduj następujące dane w edytorze ładowania danych jako ładowanie wbudowane, aby utworzyć poniższe przykłady wyrażeń wykresu:

```
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];
```

Przykład 1: Wyrażenia wykresu z wyszukiwaniami tekstowymi

Utwórz tabelę w arkuszu Qlik Sense z następującymi wyrażeniami wykresu

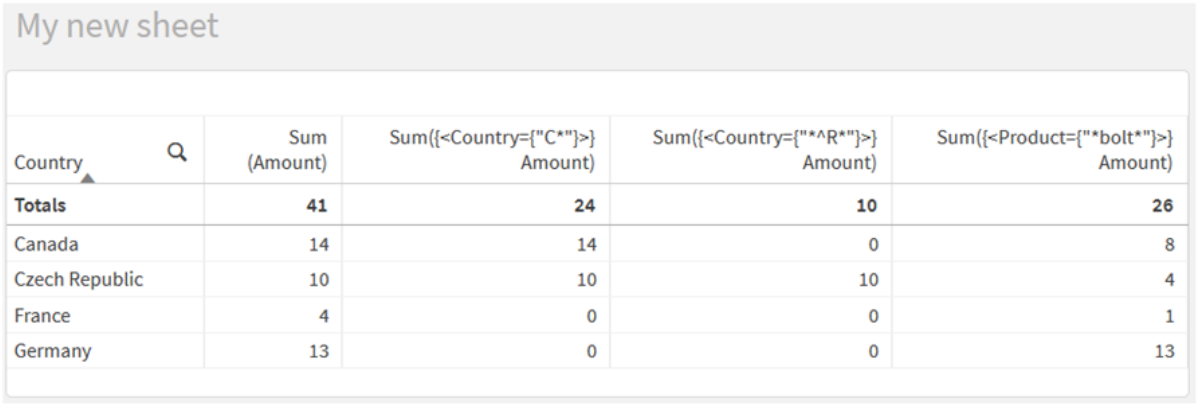
Tabela – modyfikatory zestawów z wyszukiwaniami tekstowymi

Kraj	Sum (Amount)	Sum({<Country={"C*"}>} Amount)	Sum({<Country={"*^R*"}>} Amount)	Sum({<Product={"*bolt*"}>} Amount)
Sumy	**41**	**24**	**10**	**26**
Kanada	14	14	0	8
Czechy	10	10	10	4
France	4	0	0	1
Germany	13	0	0	13

Objaśnienie

- Wymiary:
  - `Country`
- Miary:
  - `Sum(Amount)`
    Suma `Amount` bez wyrażenia zestawu.
  - `Sum({<Country={"C*"}>}Amount)`
    Suma `Amount` dla wszystkich krajów o nazwach rozpoczynających się od `C`, takich jak `Canada` i `Czech Republic`.
  - `Sum({<Country={"*^R*"}>}Amount)`
    Suma `Amount` dla wszystkich krajów mających w nazwie słowo zaczynające się od `R`, takich jak `Czech Republic`.
  - `Sum({<Product={"*bolt*"}>}Amount)`
    Suma `Amount` dla wszystkich produktów zawierających ciąg `bolt`, takich jak `Bolt` i `Anchor bolt`.

Modyfikatory zestawów z wyszukiwaniami tekstowymi


My new sheet

Country	Sum (Amount)	Sum({<Country={"C*"}>} Amount)	Sum({<Country={"*^R*"}>} Amount)	Sum({<Product={"*bolt*"}>} Amount)
Totals	**41**	**24**	**10**	**26**
Canada	14	14	0	8
Czech Republic	10	10	10	4
France	4	0	0	1
Germany	13	0	0	13

Przykład 2: Wyrażenia wykresu z wyszukiwaniami liczbowymi

Utwórz tabelę w arkuszu Qlik Sense z następującymi wyrażeniami wykresu

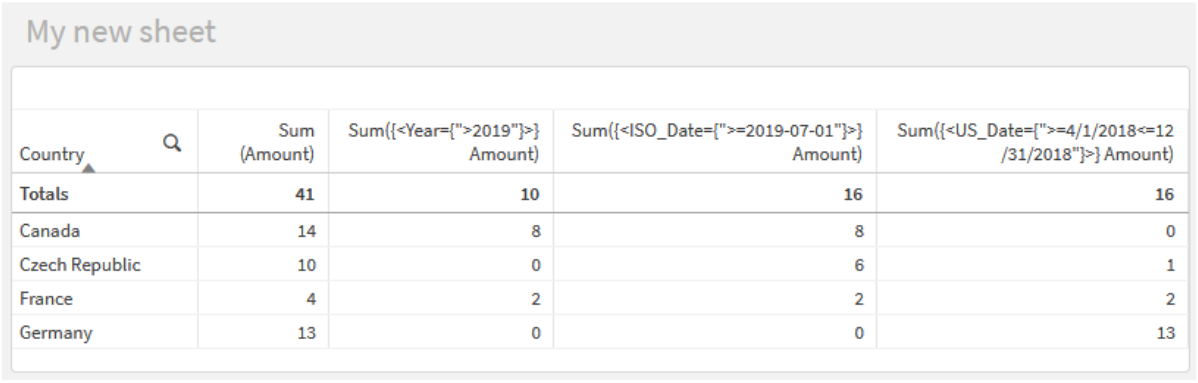
Tabela – modyfikatory zestawów z wyszukiwaniami liczbowymi

Kraj	Sum (Amount)	Sum({<Year={">2019"}>} Amount)	Sum({<ISO_Date={">=2019-07-01"}>} Amount)	Sum({<US_Date={">=4/1/2018<=12/31/2018"}>} Amount)
Sumy	**41**	**10**	**16**	**16**
Kanada	14	8	8	0
Czechy	10	0	6	1
France	4	2	2	2
Germany	13	0	0	13

Objaśnienie

- Wymiary:
  - `Country`
- Miary:
  - `Sum(Amount)`
    Suma `Amount` bez wyrażenia zestawu.
  - `Sum({<Year={">2019"}>}Amount)`
    Suma `Amount` za wszystkie lata po `2019`.
  - `Sum({<ISO_Date={">=2019-07-01"}>}Amount)`
    Suma `Amount` dla wszystkich dat od `2019-07-01` i późniejszych. Format daty w wyszukiwaniu musi być zgodny z formatem pola.
  - `Sum({<US_Date={">=4/1/2018<=12/31/2018"}>}Amount)`
    Suma `Amount` dla wszystkich dat od `4/1/2018` do `12/31/2018`, w tym dat rozpoczęcia i zakończenia. Format dat w wyszukiwaniu musi być zgodny z formatem pola.

Modyfikatory zestawów z wyszukiwaniami liczbowymi



Przykład 3: Wyrażenia wykresu z wyszukiwaniami według wyrażenia

Utwórz tabelę w arkuszu Qlik Sense z następującymi wyrażeniami wykresu

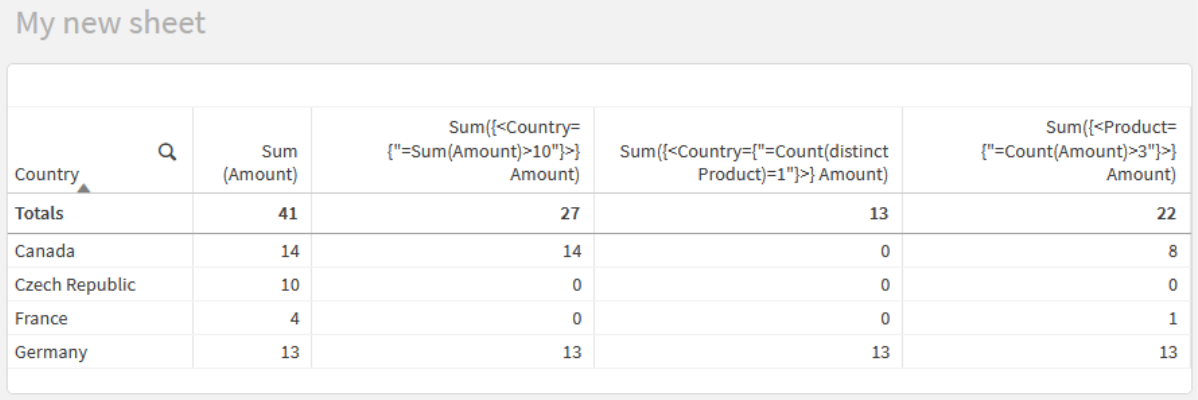
Table - Set modifiers with expression searches

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

Objaśnienie

- Wymiary:
  - `Country`
- Miary:
  - `Sum(Amount)`
    Suma `Amount` bez wyrażenia zestawu.
  - `Sum({<Country={"=Sum(Amount)>10"}>}Amount)`
    Suma `Amount` dla wszystkich krajów, których zagregowana suma `Amount` jest większa niż `10`.
  - `Sum({<Country={"=Count(distinct Product)=1"}>}Amount)`
    Suma `Amount` dla wszystkich krajów powiązanych z dokładnie jednym odrębnym produktem.
  - `Sum({<Product={"=Count(Amount)>3"}>}Amount)`
    Suma `Amount` dla wszystkich krajów, które mają więcej niż trzy transakcje w danych.

Modyfikatory zestawów z wyszukiwaniami według wyrażenia


My new sheet

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

Przykłady	Wyniki
sum({$-1<Product = {"*Internal*", "*Domestic*"}>} Sales)	Zwraca wartość sprzedaży dla bieżącego wyboru bez uwzględnienia transakcji dotyczących produktów, których nazwy zawierają ciąg „Internal" lub „Domestic".
sum({$<Customer = {"=Sum ({1<Year = {2007}>} Sales) > 1000000"}>} Sales)	Zwraca wartość sprzedaży dla bieżącej selekcji, ale z nową selekcją w polu Customer: tylko klienci, którzy w 2007 r. odnotowali sprzedaż w wysokości większej niż 1000000.

Modyfikatory zestawów z rozszerzeniami przez znak dolara

Rozszerzenia przez znak dolara to konstrukcje, które są obliczane przed przeanalizowaniem i oceną wyrażenia. Wynik jest następnie wstawiany do wyrażenia zamiast `$(...)`. Wyrażenie jest następnie obliczane na podstawie wyniku rozszerzenia przez znak dolara.

Edytor wyrażeń wyświetla podgląd rozszerzenia przez znak dolara, dzięki czemu można sprawdzić jego rezultat.

Podgląd rozszerzenia przez znak dolara w edytorze wyrażeń

Użyj rozszerzeń przez znak dolara, jeśli chcesz skorzystać z obliczenia w swoim zestawie elementów.

Jeśli na przykład chcesz spojrzeć tylko na ostatni możliwy rok, możesz użyć następującej konstrukcji:

```
<Year = {$(=Max(Year))}>
```

`Max(Year)` jest obliczane w pierwszej kolejności, a wynik jest wstawiany do wyrażenia zamiast `$(…)`.

Wynik po rozszerzeniu przez znak dolara będzie wyrażeniem takim jak:

```
<Year = {2021}>
```

Wyrażenie wewnątrz rozszerzenia przez znak dolara jest obliczane na podstawie bieżącego wyboru. Oznacza to, że jeśli masz wybór w innym polu, wpłynie to na wynik wyrażenia.

Jeśli chcesz, aby obliczenie było niezależne od wyboru, użyj analizy zestawu w rozszerzeniu przez znak dolara. Na przykład:

```
<Year = {$(=Max({1} Year))}>
```

Ciągi znaków

Jeśli chcesz, aby rozszerzenie przez znak dolara dało w wyniku ciąg znaków, obowiązują normalne zasady używania cudzysłowów. Na przykład:

```
<Country = {'$(=FirstSortedValue(Country,Date)'}>
```

Wynik po rozszerzeniu przez znak dolara będzie wyrażeniem takim jak:

```
<Country = {'New Zealand'}>
```

Jeśli nie użyjesz cudzysłowów, wystąpi błąd składni.

Liczby

Jeśli chcesz, aby rozszerzenie przez znak dolara dało w wyniku liczbę, rozszerzenie musi mieć takie samo formatowanie jak pole. Oznacza to, że czasami trzeba opakować wyrażenie w funkcję formatującą.

Na przykład:

```
<Amount = {$(=Num(Max(Amount), '###0.00'))}>
```

Wynik po rozszerzeniu przez znak dolara będzie wyrażeniem takim jak:

```
<Amount = {12362.00}>
```

Użyj skrótu, aby wymusić używanie przez rozszerzenie zawsze przecinka dziesiętnego bez separatora tysięcy. Na przykład:

```
<Amount = {$(#=Max(Amount))}>
```

Daty

Jeśli chcesz, aby rozszerzenie przez znak dolara dało w wyniku datę, rozszerzenie musi mieć prawidłowe formatowanie. Oznacza to, że czasami trzeba opakować wyrażenie w funkcję formatującą.

Na przykład:

```
<Date = {'$(=Date(Max(Date)))'}>
```

Wynik po rozszerzeniu przez znak dolara będzie wyrażeniem takim jak:

```
<Date = {'12/31/2015'}>
```

Podobnie jak w przypadku ciągów musisz użyć prawidłowych cudzysłowów.

Typowym zastosowaniem jest ograniczenie obliczeń do ostatniego miesiąca (lub roku). Następnie możesz użyć wyszukiwania liczbowego w połączeniu z funkcją `AddMonths()`.

Na przykład:

```
<Date = {">=$(=AddMonths(Today(),-1))"}>
```

Wynik po rozszerzeniu przez znak dolara będzie wyrażeniem takim jak:

```
<Date = {">=9/31/2021"}>
```

Spowoduje to wybranie wszystkich zdarzeń, które wystąpiły w ostatnim miesiącu.

Przykład: Wyrażenia wykresu dla modyfikatorów zestawu z rozszerzeniami przez znak dolara

Przykład – wyrażenia wykresu

Skrypt ładowania

Załaduj następujące dane w edytorze ładowania danych jako ładowanie wbudowane, aby utworzyć poniższe przykłady wyrażeń wykresu:

```
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];
```

Wyrażenia wykresu z rozszerzeniami przez znak dolara

Utwórz tabelę w arkuszu Qlik Sense z następującymi wyrażeniami wykresu

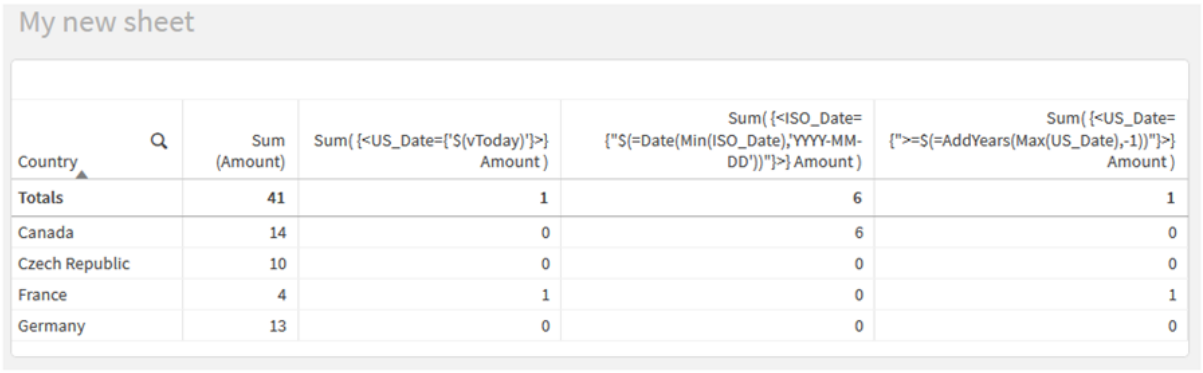
Tabela – modyfikatory zestawów z rozszerzeniami przez znak dolara

Kraj	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)
Sumy	**41**	**1**	**6**	**1**
Kanada	14	0	6	0
Czechy	10	0	0	0
France	4	1	0	1
Germany	13	0	0	0

Objaśnienie

- Wymiary:
  - `Country`
- Miary:
  - `Sum(Amount)`
    Suma `Amount` bez wyrażenia zestawu.
  - `Sum({<US_Date={'$(vToday)'}>}Amount)`
    Suma `Amount` dla wszystkich rekordów, w których wartość `US_Date` jest taka sama jak w zmiennej `vToday`.
  - `Sum({<ISO_Date={"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>}Amount)`
    Suma `Amount` dla wszystkich rekordów, w których wartość `ISO_Date` jest taka sama jak pierwsza (najmniejsza) możliwa wartość `ISO_Date`. Funkcja `Date()` jest potrzebna, aby zapewnić zgodność formatu daty z formatem pola.
  - `Sum({<US_Date={">=$(=AddYears(Max(US_Date),-1))"}>}Amount)`
    Suma `Amount` dla wszystkich rekordów, które mają `US_Date` po lub na rok przed najpóźniejszą (największą) możliwą datą `US_Date`. Funkcja `AddYears()` zwróci datę w formacie określonym przez zmienną `DateFormat`, która musi być zgodna z formatem pola `US_Date`.

Modyfikatory zestawów z rozszerzeniami przez znak dolara



Przykłady	Wyniki
sum({$<Year = {$(#vLastYear)}>} Sales)	Zwraca wartość sprzedaży z roku poprzedzającego bieżącą selekcję. W tym przykładzie rozszerzenie przez znak dolara obejmuje zmienną vLastYear zawierającą odpowiedni rok.
sum({$<Year = {$(#=Only(Year)-1)}>} Sales)	Zwraca wartość sprzedaży z roku poprzedzającego bieżącą selekcję. W tym przykładzie rozszerzenie przez znak dolara jest używane do obliczenia poprzedniego roku.

Modyfikatory zestawów z operatorami zestawów

Operatory zestawów służą do uwzględniania, wykluczania albo tworzenia części wspólnych zestawów danych. Łączą one różne metody definiowania zestawów elementów.

Operatory są takie same jak te używane do identyfikatorów zestawu.

Operatory

Operator	Opis
+	Suma. Operacja dwuargumentowa zwracająca zestaw zawierający wszystkie rekordy lub elementy, które należą do któregokolwiek z dwóch zestawów wejściowych.
-	Wykluczenie. Operacja dwuargumentowa zwracająca zestaw zawierający wszystkie rekordy lub elementy, które należą do pierwszego, a nie należą do drugiego z dwóch zestawów wejściowych. W przypadku użycia jako operator jednoargumentowy zwraca dopełnienie zestawu wejściowego.
*	Część wspólna. Operacja dwuargumentowa zwracająca zestaw zawierający wszystkie rekordy lub elementy, które należą do obu zestawów wejściowych.
/	Różnica symetryczna (XOR). Operacja dwuargumentowa zwracająca zestaw zawierający wszystkie rekordy lub elementy, które należą do jednego, ale nie obu zestawów wejściowych.

Na przykład następujące dwa modyfikatory definiują ten sam zestaw wartości pól:

- `<Year = {1997, "20*"}>`
- `<Year = {1997} + {"20*"}>`

Oba wyrażenia wybierają rok `1997` i lata zaczynające się od `20`. Innymi słowy jest to suma dwóch warunków.

Operatory zestawów umożliwiają również tworzenie bardziej złożonych definicji. Na przykład:

```
<Year = {1997, "20*"} - {2000}>
```

Wyrażenie to wybierze te same lata co powyżej, ale dodatkowo wykluczy rok `2000`.

.

Przykłady: Wyrażenia wykresu dla modyfikatorów zestawu z operatorami zestawu

Przykłady – wyrażenia wykresu

Skrypt ładowania

Załaduj następujące dane w edytorze ładowania danych jako ładowanie wbudowane, aby utworzyć poniższe przykłady wyrażeń wykresu:

```
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];
```

Wyrażenia wykresu

Utwórz tabelę w arkuszu Qlik Sense z następującymi wyrażeniami wykresu

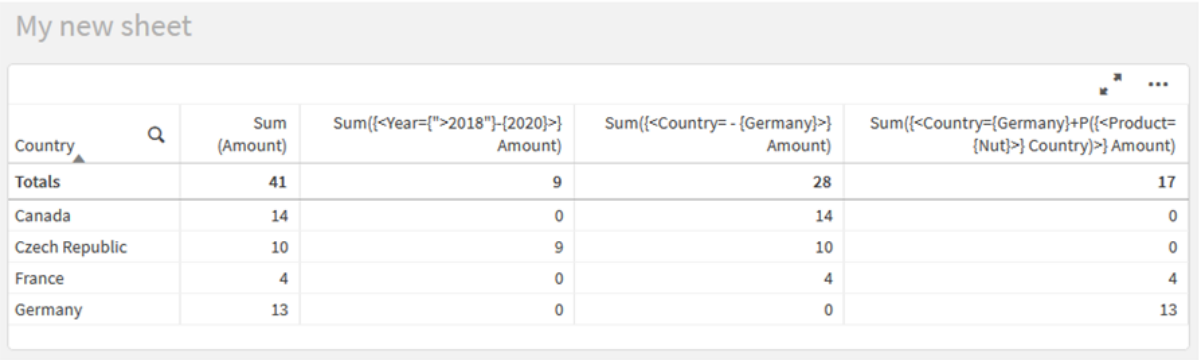
Tabela – modyfikatory zestawów z operatorami zestawów

Kraj	Sum (Amount)	Sum({<Year={">2018"}-{2020}>} Amount)	Sum ({<Country=-{Germany}>} Amount)	Sum({<Country={Germany}+P({<Product={Nut}>}Country)>} Amount)
Sumy	**41**	**9**	**28**	**17**
Kanada	14	0	14	0
Czechy	10	9	10	0
France	4	0	4	4
Germany	13	0	0	13

Objaśnienie

- Wymiary:
  - `Country`
- Miary:
  - `Sum(Amount)`
    Suma `Amount` bez wyrażenia zestawu.
  - `Sum({<Year={">2018"}-{2020}>}Amount)`
    Suma `Amount` za wszystkie lata po `2018`, z wyjątkiem `2020`.
  - `Sum({<Country=-{Germany}>}Amount)`
    Suma `Amount` dla wszystkich krajów z wyjątkiem `Germany`. Zwróć uwagę na jednoargumentowy operator wykluczenia.
  - `Sum({<Country={Germany}+P({<Product={Nut}>}Country)>}Amount)`
    Suma `Amount` dla `Germany` i wszystkich krajów powiązanych z produktem `Nut`.

Modyfikatory zestawów z operatorami zestawów



Country	Sum (Amount)	Sum({<Year={">2018"}-{2020}>} Amount)	Sum({<Country= - {Germany}>} Amount)	Sum({<Country={Germany}+P({<Product={Nut}>} Country)>} Amount)
Totals	**41**	**9**	**28**	**17**
Canada	14	0	14	0
Czech Republic	10	9	10	0
France	4	0	4	4
Germany	13	0	0	13

Przykłady	Wyniki
sum({$<Product = Product + {OurProduct1} - {OurProduct2} >} Sales)	Zwraca wartość sprzedaży dla bieżącej selekcji, ale po dodaniu produktu OurProduct1 do listy wybranych produktów, a usunięciu z niej produktu OurProduct2.
sum({$<Year = Year + ({“20*”,1997} - {2000}) >} Sales)	Zwraca wartość sprzedaży dla bieżącej selekcji, ale z dodatkowymi selekcjami w polu „Year”: 1997 i wszystkie lata zaczynające się na „20”, ale nie sam rok 2000. Jeśli jednak rok 2000 jest uwzględniony w bieżącej selekcji, to po zastosowaniu modyfikacji będzie nadal uwzględniony.
sum({$<Year = (Year + {“20*”,1997}) - {2000} >} Sales)	Zwraca prawie ten sam wynik, co wyrażenie w poprzednim przykładzie, ale w tym przypadku rok 2000 zostanie wykluczony, nawet jeśli początkowo zawierał się w początkowej selekcji. Ten przykład ilustruje, jak ważne w niektórych przypadkach mogą być nawiasy określające pierwszeństwo.

Przykłady	Wyniki
sum({$<Year = {“*”} - {2000}, Product = {“*bearing*”} >} Sales)	Zwraca wartość sprzedaży dla bieżącego wyboru ale z nowym wyborem w polu „Year”: wszystkie lata poza rokiem 2000 i tylko dla produktów zawierających ciąg „bearing”.

Modyfikatory zestawu z niejawnymi operatorami zestawu

Standardowym sposobem zapisywania wyborów w modyfikatorze zestawu jest użycie znaku równości. Na przykład:

```
Year = {">2015"}
```

Wyrażenie po prawej stronie znaku równości w modyfikatorze zestawu nazywa się zestawem elementów. Definiuje ono zestaw odrębnych wartości pól, czyli wybór.

Notacja ta umożliwia definiowanie nowego wyboru z odrzuceniem bieżącego wyboru w danym polu. Jeżeli więc identyfikator zestawu zawiera wybór w tym polu, stary wybór zostanie zastąpiony przez ten z zestawu elementów.

Jeśli chcesz oprzeć swój wybór na bieżącym wyborze w polu, musisz użyć innego wyrażenia

Na przykład, jeśli chcesz uwzględnić stary wybór i dodać wymaganie, że rok jest po 2015, możesz napisać:

```
Year = Year * {">2015"}
```

Gwiazdka jest operatorem zestawu definiującym część wspólną, więc otrzymasz część wspólną bieżącego wyboru w `Year` i dodatkowego wymagania, że rok ma przypadać po `2015`. Alternatywny sposób zapisania tego jest następujący:

```
Year *= {">2015"}
```

Oznacza to, że operator przypisania (`*=`) niejawnie definiuje część wspólną.

Podobnie można definiować niejawne sumy, części wspólne i różnice symetryczne, używając: `+=`, `-=`, `/=`

Przykłady: Wyrażenia wykresu dla modyfikatorów zestawu z niejawnymi operatorami zestawu

Przykłady – wyrażenia wykresu

Skrypt ładowania

Załaduj następujące dane w edytorze ładowania danych jako ładowanie wbudowane, aby utworzyć poniższe przykłady wyrażeń wykresu:

```
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];
```

Wyrażenia wykresu z niejawnymi operatorami zestawu

Utwórz tabelę w arkuszu Qlik Sense z następującymi wyrażeniami wykresu

Wybierz `Canada` i `Czech Republic` z listy krajów.

Tabela – wyrażenia wykresu z niejawnymi operatorami zestawu

Kraj	Sum (Amount)	Sum({<Country*= {Canada}>} Amount)	Sum({<Country-= {Canada}>} Amount)	Sum({<Country+= {France}>} Amount)
Sumy	**24**	**14**	**10**	**28**
Kanada	14	14	0	14
Czechy	10	0	10	10
France	0	0	0	4

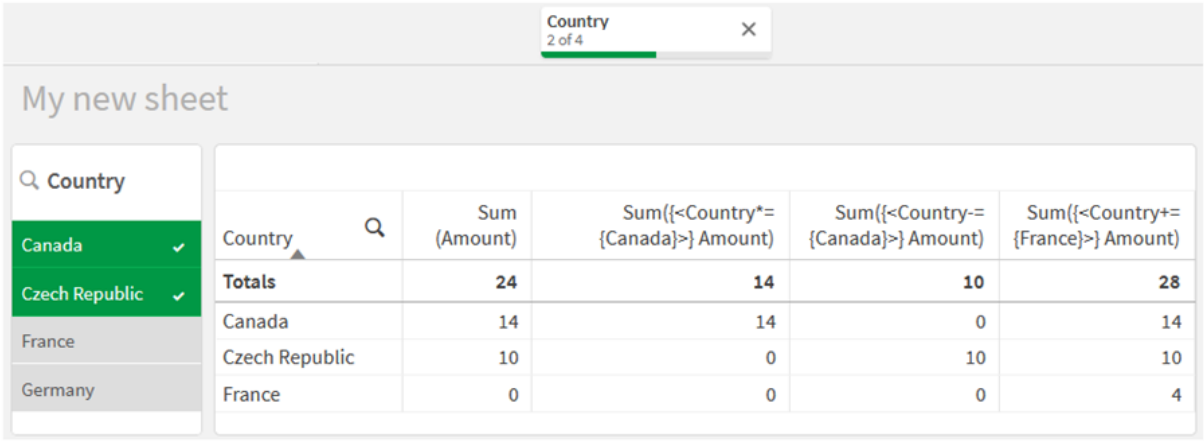
Objaśnienie

- Wymiary:
  - `Country`
- Miary:
  - `Sum(Amount)`
    Suma `Amount` dla bieżącego wyboru. Zauważ, że tylko `Canada` i `Czech Republic` mają wartości niezerowe.
  - `Sum({<Country*={Canada}>}Amount)`
    Część wspólna sumy `Amount` dla bieżącego wyboru z wymogiem, aby wartością `Country` było `Canada`. Jeśli `Canada` nie należy do wyboru użytkownika, wyrażenie zestawu zwraca pusty zestaw, a kolumna będzie miała `0` we wszystkich wierszach.
  - `Sum({<Country-={Canada}>}Amount)`

Suma `Amount` dla bieżącego wyboru, ale z uprzednim wykluczeniem `Canada` z wyboru `Country`. Jeśli `Canada` nie należy do wyboru użytkownika, wyrażenie zestawu nie zmieni żadnych liczb.

- `Sum({<Country+={France}>}Amount)`
  Suma `Amount` dla bieżącego wyboru, ale z uprzednim dodaniem `France` do wyboru `Country`. Jeśli `France` należy już do wyboru użytkownika, wyrażenie zestawu nie zmieni żadnych liczb.

Modyfikatory zestawu z niejawnymi operatorami zestawu



Przykłady	Wyniki
sum({$<Product += {OurProduct1, OurProduct2} >} Sales)	Zwraca wartość sprzedaży dla bieżącej selekcji, ale przy użyciu operatora niejawnej sumy dodaje do listy wybranych produktów produktyOurProduct1 i OurProduct2.
sum({$<Year += {“20*”,1997} - {2000} >} Sales)	Zwraca wartość sprzedaży dla bieżącej selekcji, ale przy użyciu operatora niejawnej sumy dodaje liczbę lat w selekcji: 1997 i wszystkie lata zaczynające się na „20”, ale nie sam rok 2000. Jeśli jednak rok 2000 jest uwzględniony w bieżącej selekcji, to po zastosowaniu modyfikacji będzie nadal uwzględniony. Działa tak samo jak wyrażenie `<Year=Year + ({“20*”,1997}-{2000})>`.
sum({$<Product *= {OurProduct1} >} Sales)	Zwraca wartość sprzedaży dla bieżącej selekcji, ale tylko dla części wspólnej aktualnie wybranych produktów i produktu OurProduct1.

Modyfikatory zestawów z wykorzystaniem funkcji zestawu

Czasami trzeba zdefiniować zestaw wartości pól za pomocą zagnieżdżonej definicji zestawu. Na przykład możesz chcieć wybrać wszystkich klientów, którzy kupili określony produkt, bez wybierania produktu.

W takich przypadkach użyj funkcji zestawu elementów `P()` i `E()`. Zwracają one zestawy elementów z odpowiednio możliwymi i wykluczonymi wartościami pola. W nawiasie możesz określić dane pole oraz wyrażenie zestawu, które definiuje zakres. Na przykład:

```
P({1<Year = {2021}>} Customer)
```

Spowoduje to zwrócenie zestawu klientów, którzy mieli transakcje w 2021 r. Możesz następnie użyć tego w modyfikatorze zestawu. Na przykład:

```
Sum({<Customer = P({1<Year = {2021}>} Customer)>} Amount)
```

To wyrażenie zestawu wybierze tych klientów, ale nie ograniczy wyboru do 2021 r.

Funkcji tych nie można używać w innych wyrażeniach.

Dodatkowo wewnątrz funkcji zestawu elementów można stosować tylko zestawy naturalne. Jest to zestaw rekordów, które można zdefiniować za pomocą prostego wyboru.

Przykład: zestaw zdefiniowany przez `{1-$}` nie może zawsze być określony poprzez wybór, nie jest więc zestawem naturalnym. Użycie tych funkcji w odniesieniu do zestawów innych niż naturalne zwróci nieoczekiwane wyniki.

Przykłady: Wyrażenia wykresu dla modyfikatorów zestawu z funkcjami zestawu

Przykłady – wyrażenia wykresu

Skrypt ładowania

Załaduj następujące dane w edytorze ładowania danych jako ładowanie wbudowane, aby utworzyć poniższe przykłady wyrażeń wykresu:

```
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];
```

Wyrażenia wykresu

Utwórz tabelę w arkuszu Qlik Sense z następującymi wyrażeniami wykresu

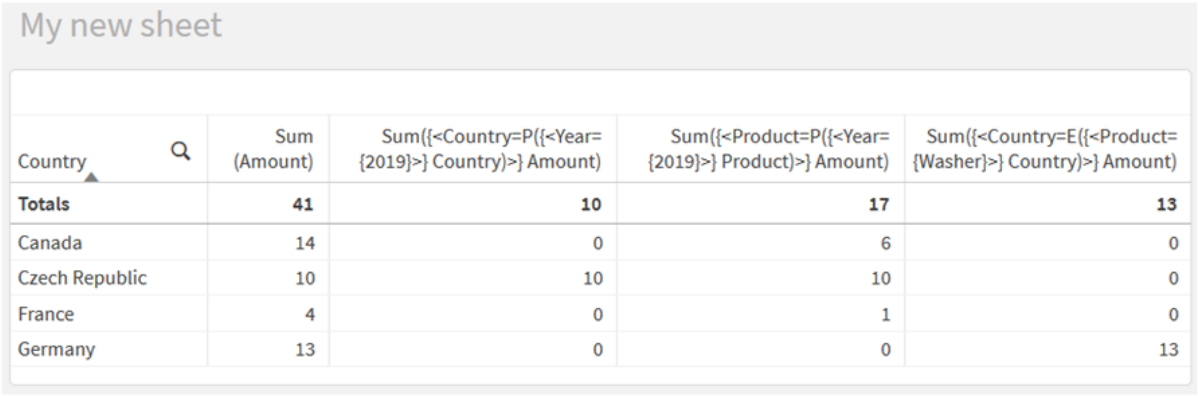
Tabela – wyrażenia zestawu z wykorzystaniem funkcji zestawu

Kraj	Sum (Amount)	Sum({<Country=P({<Year={2019}>}Country)>} Amount)	Sum({<Product=P({<Year={2019}>}Product)>} Amount)	Sum({<Country=E({<Product={Washer}>}Country)>} Amount)
Sumy	**41**	**10**	**17**	**13**
Kanada	14	0	6	0
Czechy	10	10	10	0
France	4	0	1	0
Germany	13	0	0	13

Objaśnienie

- Wymiary:
  - `Country`
- Miary:
  - `Sum(Amount)`
    Suma `Amount` bez wyrażenia zestawu.
  - `Sum({<Country=P({<Year={2019}>} Country)>} Amount)`
    Suma `Amount` dla krajów powiązanych z rokiem `2019`. Nie ograniczy to jednak obliczeń do `2019`.
  - `Sum({<Product=P({<Year={2019}>} Product)>} Amount)`
    Suma `Amount` dla produktów związanych z rokiem `2019`. Nie ograniczy to jednak obliczeń do `2019`.
  - `Sum({<Country=E({<Product={Washer}>} Country)>} Amount)`
    Suma `Amount` dla krajów, które nie są powiązane z produktem `Washer`.

Modyfikatory zestawu z wykorzystaniem funkcji zestawu



Przykłady	Wyniki
sum({$<Customer = P({1<Product= {'Shoe'}>} Customer)>} Sales)	Zwraca wartość sprzedaży dla bieżącego wyboru, ale tylko tych klientów, którzy co najmniej raz kupili produkt „Shoe”. W tym przypadku funkcja elementowa P() zwraca listę możliwych klientów, których wskazuje wybór „Shoe” w polu Product.
sum({$<Customer = P({1<Product= {'Shoe'}>})>} Sales)	Tak samo jak powyżej. Jeśli pole w funkcji elementowej zostanie pominięte, funkcja zwróci możliwe wartości pola wskazanego w przypisaniu zewnętrznym.
sum({$<Customer = P({1<Product= {'Shoe'}>} Supplier)>} Sales)	Zwraca wartość sprzedaży dla bieżącego wyboru, ale tylko tych klientów, którzy co najmniej raz dostarczyli produkt „Shoe”. To znaczy, ze klient jest również dostawcą. W tym przypadku funkcja elementowa P() zwraca listę możliwych dostawców, których wskazuje wybór „Shoe” w polu Product. Lista dostawców jest następnie stosowana jako wybór w polu Customer.
sum({$<Customer = E({1<Product= {'Shoe'}>})>} Sales)	Zwraca wartość sprzedaży dla bieżącego wyboru, ale tylko tych klientów, którzy nigdy nie kupili produktu „Shoe”. W tym przypadku funkcja elementowa E() zwraca listę klientów wykluczonych wyborem produktu „Shoe” w polu Product.

Wewnętrzne i zewnętrzne wyrażenia zestawu

Wyrażenia zestawów mogą być używane wewnątrz i na zewnątrz funkcji agregacji oraz są ujęte w nawiasy klamrowe.

Wyrażenie zestawu użyte w funkcji agregacji może wyglądać tak:

Przykład: Wewnętrzne wyrażenie zestawu

```
Sum( {$<Year={2021}>} Sales )
```

Jeśli masz wyrażenia z kilkom agregacjami i chcesz uniknąć konieczności pisania tego samego wyrażenia zestawu w każdej funkcji agregacji, użyj wyrażenia zestawu na zewnątrz funkcji agregacji.

Zewnętrzne wyrażenie zestawu musi znajdować się na początku zakresu.

Przykład: Zewnętrzne wyrażenie zestawu

```
{<Year={2021}>} Sum(Sales) / Count(distinct Customer)
```

Jeśli używasz wyrażenia zestawu na zewnątrz funkcji agregacji, to możesz ją zastosować także do istniejących miar głównych.

Przykład: Zewnętrzne wyrażenie zestawu zastosowane do miary głównej

```
{<Year={2021}>} [Master Measure]
```

Wyrażenie zestawu użyte na zewnątrz funkcji agregacji ma wpływ na całe wyrażenie, chyba że zostanie ujęte w nawias, który określa zakres dostępności. W poniższym przykładzie leksykalnego zakresu wyrażenie zestawu jest stosowane tylko do agregacji w nawiasie.

Przykład: Leksykalne ograniczenie zakresu

```
( {<Year={2021}>} Sum(Amount) / Count(distinct Customer) ) - Avg(CustomerSales)
```

Reguły

Zakres leksykalny

Wyrażenie zestawu ma wpływ na całe wyrażenie, chyba że znajduje się w nawiasie. W takim przypadku nawias określa zakres leksykalny.

pozycja;

Wyrażenie zestawu należy umieścić na początku zakresu leksykalnego.

Kontekst

Kontekstem jest selekcja odpowiednia dla danego wyrażenia. Zwyczajowo kontekstem zawsze jest domyślny stan obecnej selekcji. Jeśli jednak obiekt zostanie ustawiony na alternatywny stan, kontekstem jest stan alternatywny obecnej selekcji.

Ponadto kontekst można zdefiniować w postaci zewnętrznego wyrażenia zestawu.

Dziedziczenie

Wewnętrzne wyrażenia zestawów mają pierwszeństwo przed zewnętrznymi wyrażeniami zestawów. Jeśli wewnętrzne wyrażenie zestawu zawiera identyfikator zestawu, to zastępuje kontekst. W przeciwnym razie kontekst i wyrażenie zestawu są łączone.

- `{$<SetExpression>}` - przesłania zewnętrzne wyrażenie zestawu
- `{<SetExpression>}` - scalenie z zewnętrznym wyrażeniem zestawu

Przypisanie zestawu elementów

Przypisanie zestawu elementów określa sposób scalania dwóch selekcji. Jeśli użyty jest normalny znak równości, to pierwszeństwo ma wewnętrzne wyrażenie zestawu. W przeciwnym przypadku zostanie użyty niejawny operator zestawu.

- `{<Field={value}>}` - ta wewnętrzna selekcja zastępuje selekcję zewnętrzną w "`Field`".
- `{<Field+={value}>}` - ta selekcja wewnętrzna zostaje scalona za pomocą operatora sumy z selekcją zewnętrzną w "`Field`".
- `{<Field*={value}>}` - ta selekcja wewnętrzna zostaje scalona za pomocą operatora przecięcia z selekcją zewnętrzną w "`Field`".

Dziedziczenie w przypadku wielu kroków

Dziedziczenie może wystąpić w przypadku wielu kroków. Przykłady:

- Bieżący wybór → `Sum(Amount)`
  Funkcja agregacji użyje kontekstu, którym w tym przypadku jest aktualna selekcja.
- Obecna selekcja → `{<Set1>} Sum(Amount)`
  `Set1` odziedziczy po obecnej selekcji, a wynikiem będzie kontekst dla funkcji agregacji.
- Obecna selekcja → `{<Set1>} ({<Set2>} Sum(Amount))`
  `Set2` odziedziczy po `Set1`, która z kolei dziedziczy z obecnej selekcji, a wynikiem będzie kontekst dla funkcji agregacji.

Funkcja Aggr()

Funkcja `Aggr()` tworzy zagnieżdżoną agregację, która ma dwie niezależne agregacje. W poniższym przykładzie `Count()` jest obliczana dla każdej wartości `Dim`, a tablica wynikowa jest agregowana przy użyciu funkcji `Sum()`.

Przykład:

`Sum(Aggr(Count(X),Dim))`
`Count()` jest wewnętrzną agregacją, a `Sum()` jest zewnętrzną agregacją.

- Agregacja wewnętrzna nie dziedziczy żadnego kontekstu z agregacji zewnętrznej.
- Agregacja wewnętrzna dziedziczy kontekst z funkcji `Aggr()`, która może zawierać wyrażenie zestawu.
- Zarówno funkcja `Aggr()`, jak i zewnętrzna funkcja agregacji, dziedziczą kontekst z zewnętrznego wyrażenia zestawu.

Kurs – Tworzenie wyrażenia zestawu

Wyrażenia zestawu możesz tworzyć na potrzeby analizy danych. W tym kontekście analizę często nazywa się analizą zestawu. W ramach analizy zestawów można zdefiniować zakres, który jest inny niż zestaw rekordów zdefiniowany przez bieżący wybór w aplikacji.

Czego się nauczysz?

Ten kurs obejmuje wyrażenia danych i wykresów do tworzenia wyrażeń zestawów przy użyciu modyfikatorów, identyfikatorów i operatorów zestawów.

Kto powinien ukończyć ten kurs

Ten kurs jest przeznaczony dla twórców aplikacji, którzy potrafią pracować z edytorem skryptu i wyrażeniami wykresów.

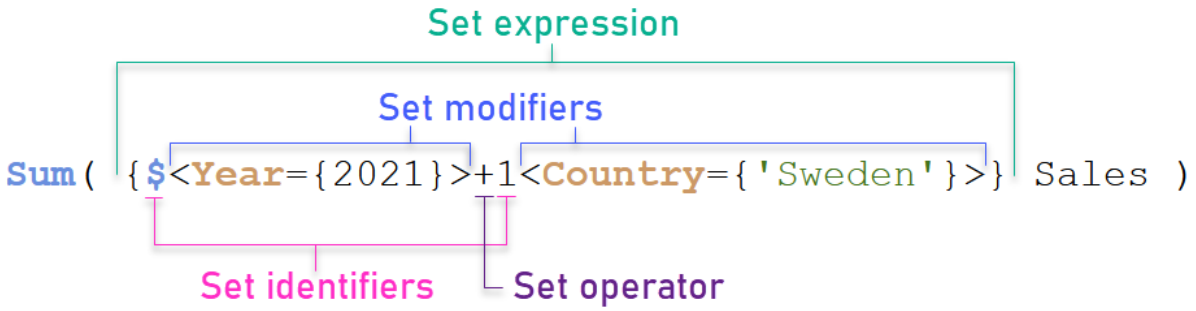
Co należy zrobić przed rozpoczęciem?

Dostęp Professional w Qlik Sense Enterprise umożliwiający ładowanie danych i tworzenie aplikacji.

Elementy w wyrażeniu zestawu

Wyrażenia zestawu są zawarte w funkcji agregacji, takiej jak `Sum()`, `Max()`, `Min()`, `Avg()` lub `Count()`. Wyrażenia zestawu składają się z bloków konstrukcyjnych znanych jako elementy. Elementy te obejmują modyfikatory, identyfikatory i operatory zestawów.

Elementy w wyrażeniu zestawu



Na przykład powyższe wyrażenie zestawu jest zbudowane na podstawie agregacji `Sum(Sales)`. Wyrażenie zestawu jest ujęte w zewnętrzny nawias klamrowy: `{ }`

Pierwszy operand w wyrażeniu to: `$<Year={2021}>`

Operand ten zwraca wartość sprzedaży za rok `2021` dla bieżącego wyboru. Modyfikator, `<Year={2021}>`, zawiera wybór roku 2021. Identyfikator zestawu `$` wskazuje, że wyrażenie zestawu jest oparte na bieżącym wyborze.

Drugi operand w wyrażeniu to: `1<Country={'Sweden'}>`

Ten operand zwraca Sales dla Sweden. Modyfikator, `<Country={'Sweden'}>`, zawiera wybór kraju Sweden. Identyfikator zestawu `1` wskazuje, że wybory dokonane w aplikacji zostaną zignorowane.

I wreszcie operator zestawu `+` wskazuje, że wyrażenie zwraca zestaw zawierający rekordy, które należą do któregokolwiek z dwóch operandów zestawu.

Kurs – Tworzenie wyrażenia zestawu

Wykonaj następujące procedury, aby utworzyć wyrażenia zestawu pokazane w tym kursie.

Tworzenie nowej aplikacji i ładowanie danych

Wykonaj następujące czynności:

1. Utwórz nową aplikację.
2. Kliknij **Edytor skryptów**. Możesz zamiast tego kliknąć **Przygotuj** > **Edytor ładowania danych** na pasku nawigacyjnym.
3. Utwórz nową sekcję w **Edytorze ładowania danych**.
4. Skopiuj następujące dane i wklej je w nowej sekcji: *Dane kursu o wyrażeniach zestawu (page 306)*
5. Kliknij polecenie **Ładuj dane**. Dane są ładowane w ramach ładowania wbudowanego.

Tworzenie wyrażeń zestawów za pomocą modyfikatorów

Modyfikator ustawień składa się z jednej lub więcej nazw pól, po których następuje wybór, który ma zostać dokonany na danym polu. Modyfikator jest ujęty w nawiasy kątowe. Na przykład w tym wyrażeniu zestawu:

```
Sum ( {<Year = {2015}>} Sales )
```

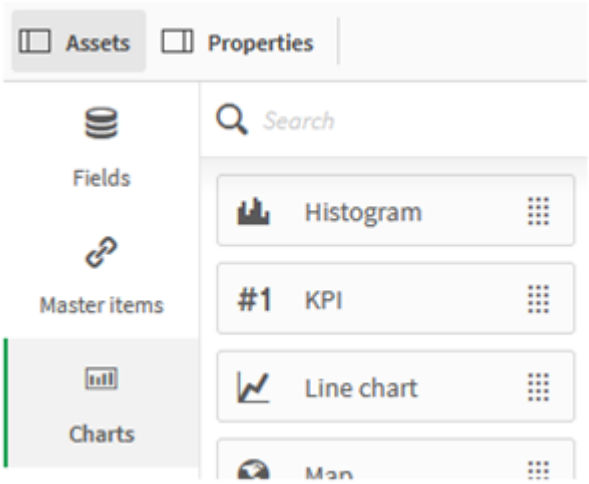
Modyfikator to:

```
<Year = {2015}>
```

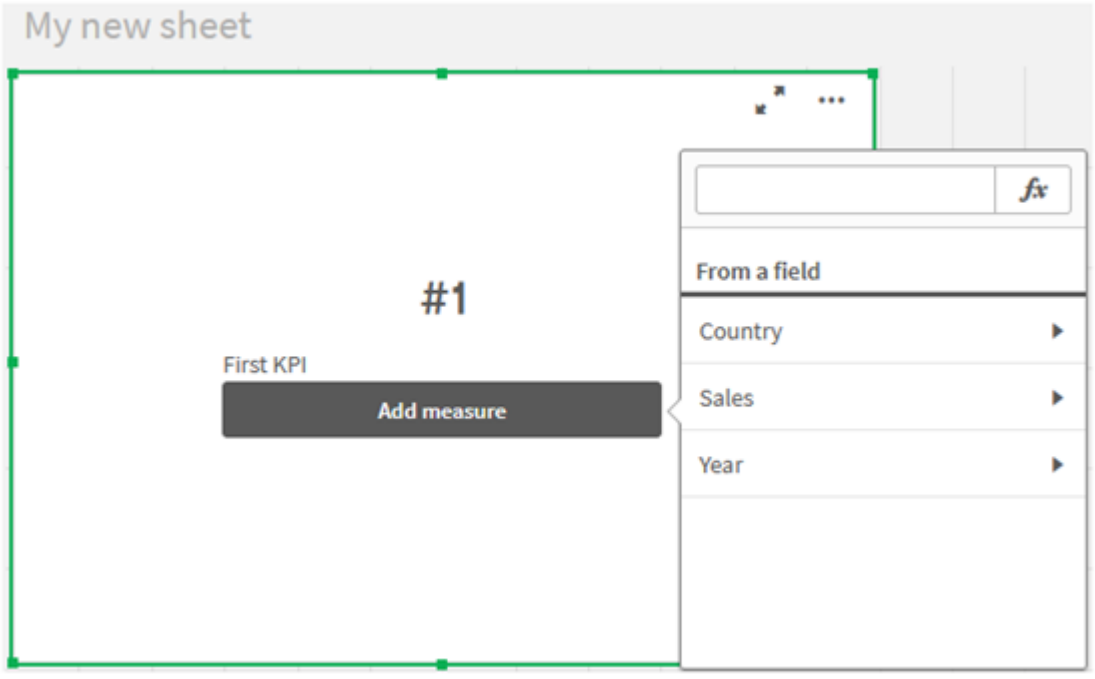
Modyfikator ten określa, że wybrane zostaną dane z roku 2015. Nawias klamrowy, w którym zawarty jest modyfikator, wskazuje wyrażenie zestawu.

Wykonaj następujące czynności:

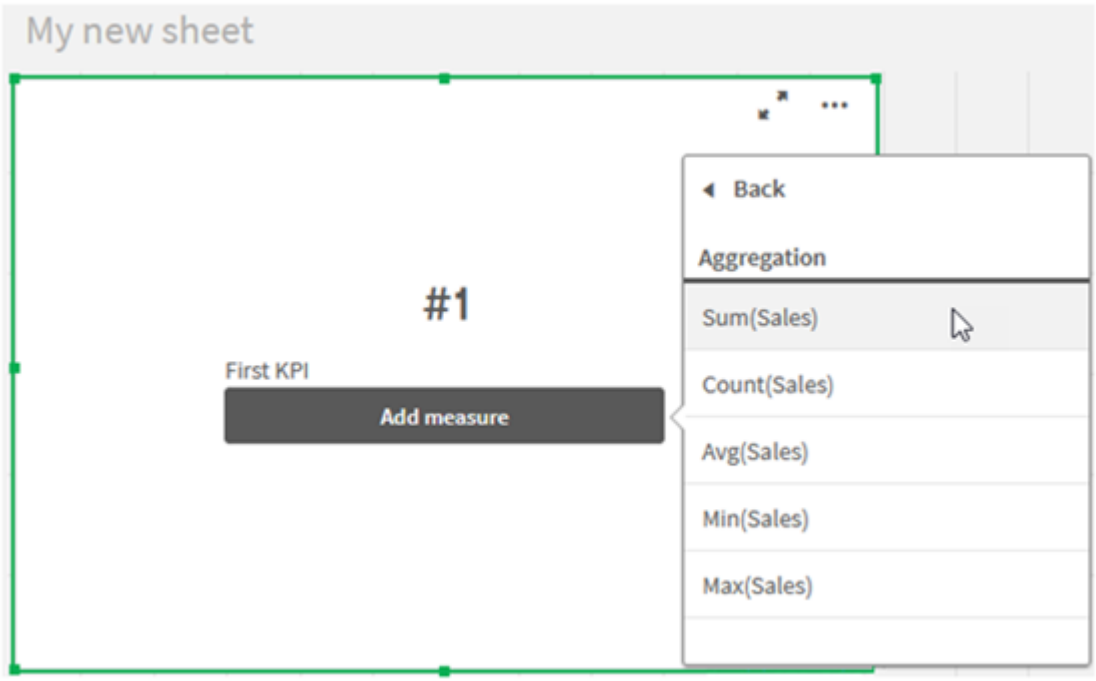
1. W arkuszu otwórz panel **Zasoby** z paska nawigacyjnego, a następnie kliknij **Wykresy**.



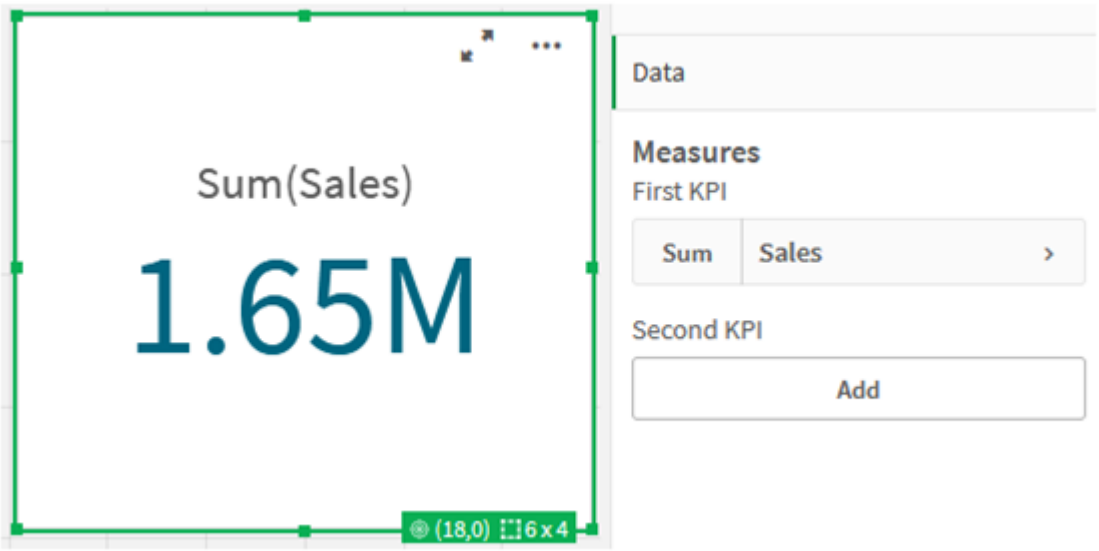
2. Przeciągnij wskaźnik **KPI** na arkusz, a następnie kliknij **Dodaj miarę**.



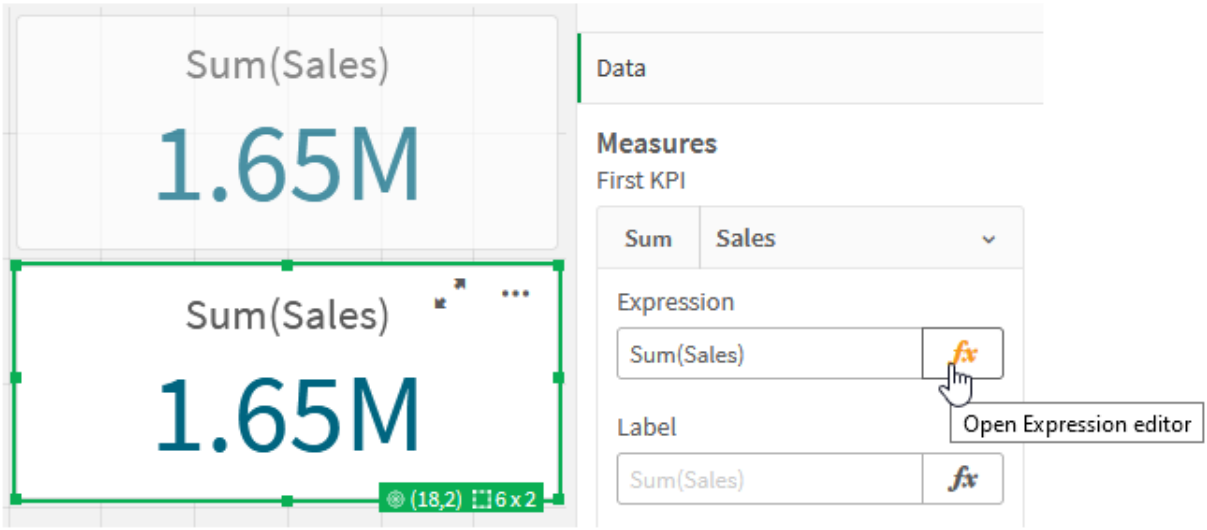
3. Kliknij `Sales`, a następnie wybierz `Sum(Sales)` do agregacji.



KPI przedstawia sumę sprzedaży za wszystkie lata.



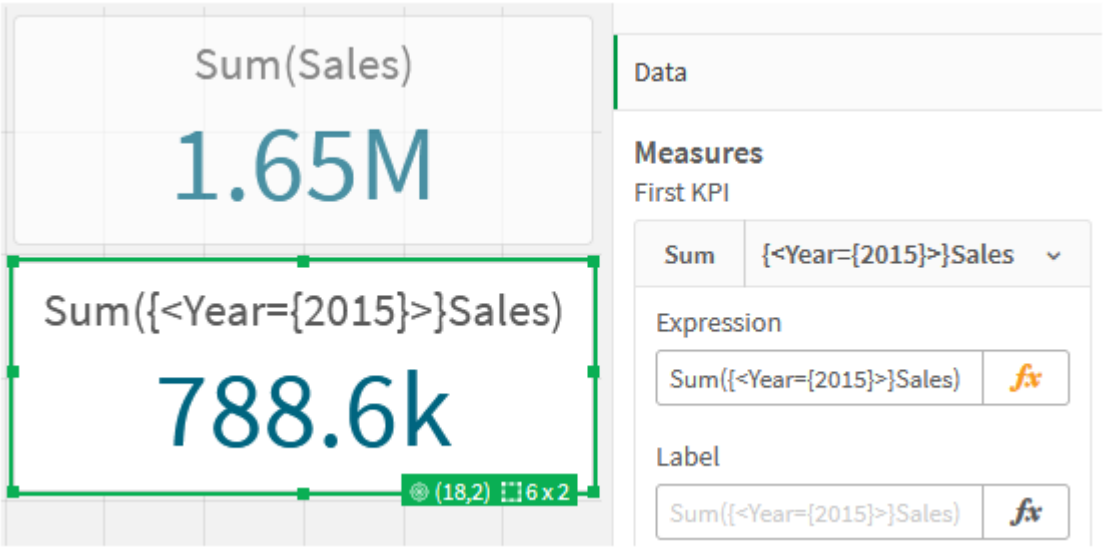
4. Skopiuj i wklej KPI, aby utworzyć nowy wskaźnik KPI.
5. Kliknij nowy wskaźnik KPI, kliknij **Sales** w obszarze **Miary**, a następnie kliknij **Otwórz edytor wyrażeń**.



Edytor wyrażeń otwiera się z agregacją `Sum(Sales)`.

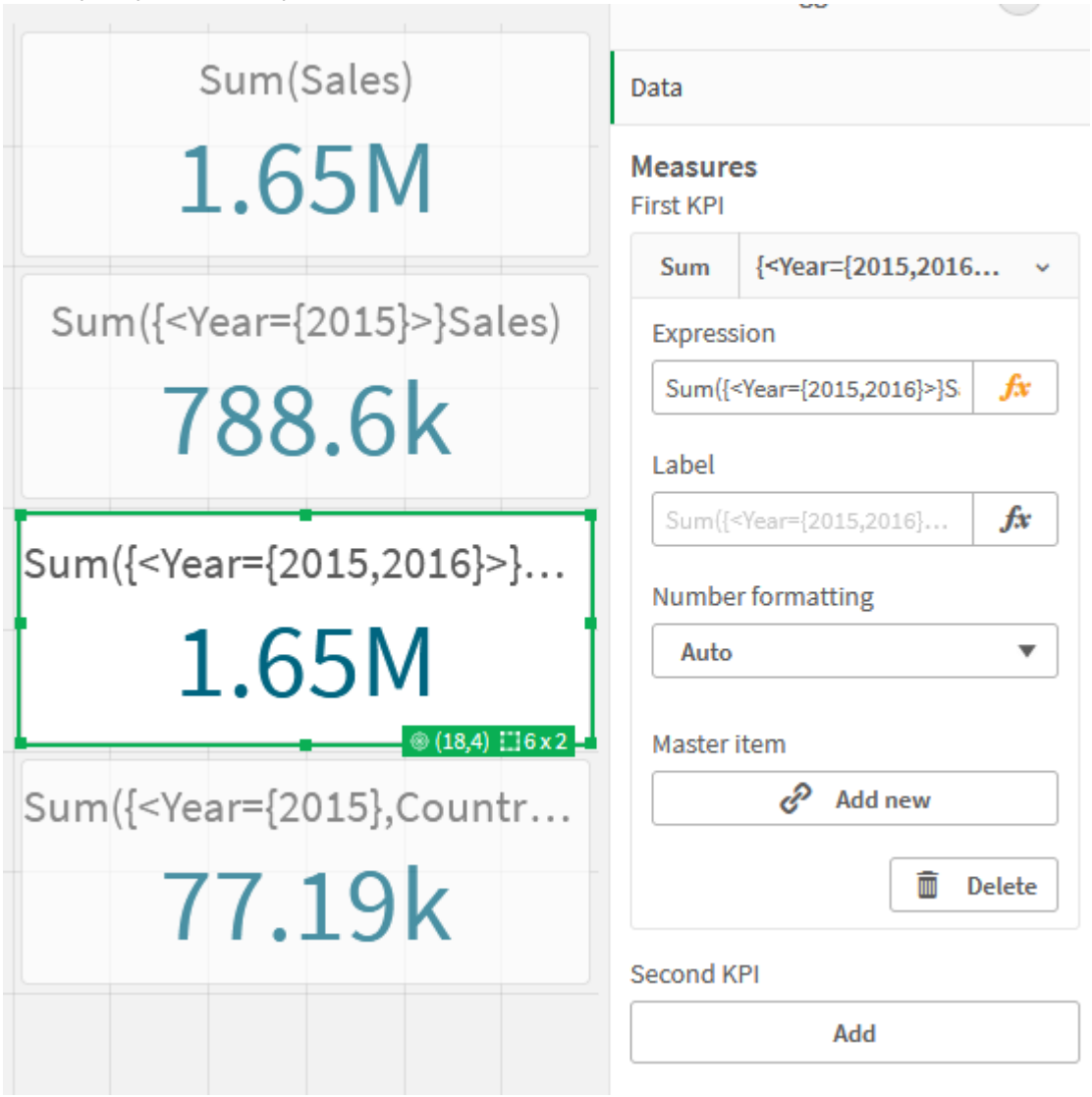
6. W edytorze wyrażeń utwórz wyrażenie sumujące Sales tylko za rok 2015:
   i. Dodaj nawiasy klamrowe, aby wskazać wyrażenie zestawu: `Sum({}Sales)`
   i. Dodaj nawiasy kątowe, aby wskazać modyfikator zestawu: `Sum({<>}Sales)`
   ii. W nawiasie kątowym dodaj pole do wybrania, w tym przypadku `Year`, a po nim znak równości. Następnie ujmij rok 2015 w innym zestawie nawiasów klamrowych. Wynikowy modyfikator zestawu to: `{<Year={2015}>}`.
   Całe wyrażenie to:
   `Sum({<Year={2015}>}Sales)`

iii. Kliknij **Zastosuj**, aby zapisać wyrażenie i zamknąć edytor wyrażeń. W KPI pokazana jest suma Sales za 2015 rok.



7. Utwórz dwa dodatkowe wskaźniki KPI za pomocą następujących wyrażeń:
   `Sum({<Year={2015,2016}>}Sales)`
   Modyfikatorem powyżej jest `<Year={2015,2016}>`. Wyrażenie zwróci sumę Sales za rok 2015 i 2016.
   `Sum({<Year={2015},Country={'Germany'}>} Sales)`
   Modyfikatorem powyżej jest `<Year={2015}, Country={'Germany'}>`. Wyrażenie zwróci sumę Sales za rok 2015, gdzie 2015 ma część wspólną z Germany.

KPI z wykorzystaniem modyfikatorów ustawień



Dodawanie identyfikatorów zestawu

Powyższe wyrażenia zestawu będą używać bieżących wyborów jako podstawy, ponieważ nie użyto identyfikatora. Następnie dodaj identyfikatory, aby określić działanie podczas dokonywania wyborów.

Wykonaj następujące czynności:

W arkuszu utwórz lub skopiuj następujące wyrażenia zestawu:

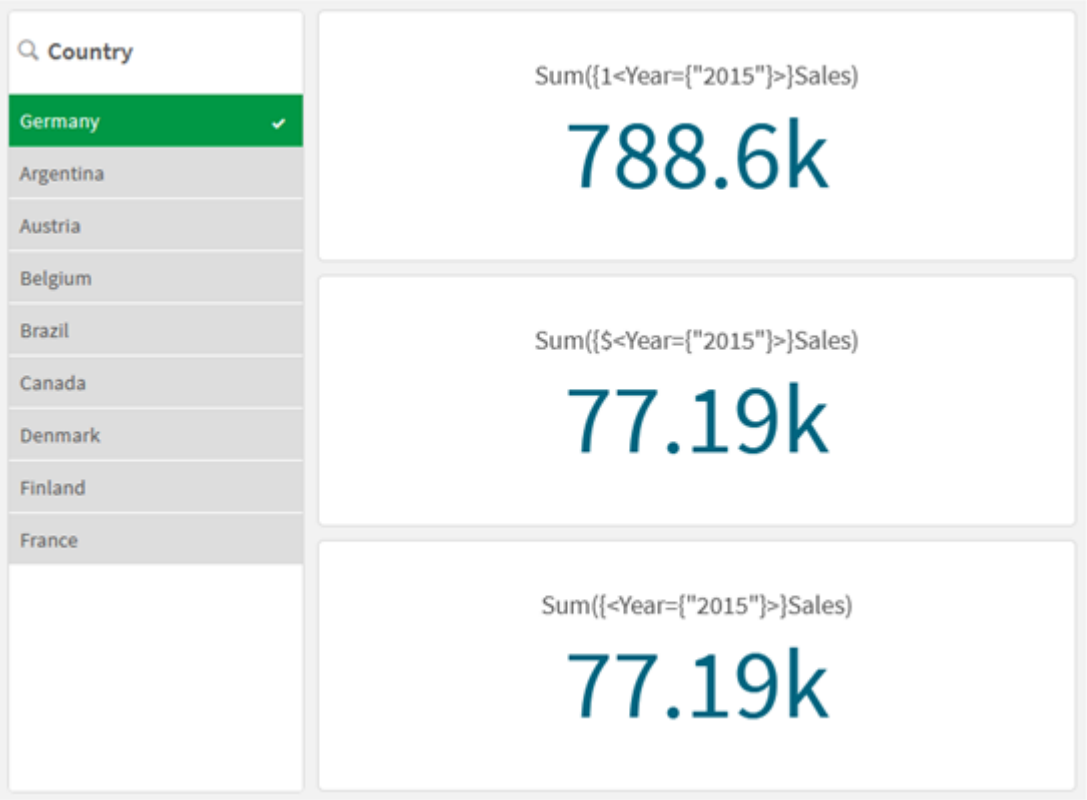
```
Sum({$<Year={"2015"}>}Sales
```

Identyfikator `$` będzie opierać wyrażenie zestawu na bieżących wyborach dokonanych w danych. Jest to również domyślne zachowanie, gdy identyfikator nie jest używany.

```
Sum({1<Year={"2015"}>}Sales)
```

Identyfikator `1` spowoduje, że agregacja `Sum(Sales)` w 2015 roku będzie ignorować bieżący wybór. Wartość agregacji nie zmieni się, gdy użytkownik dokona innych wyborów. Na przykład po wybraniu poniżej Germany wartość sumy zagregowanej z roku 2015 nie zmieni się.

Wskaźniki KPI z wykorzystaniem modyfikatorów zestawu i identyfikatorów



Dodawanie operatorów

Operatory zestawów służą do uwzględniania, wykluczania albo przecinania zestawów danych. Wszystkie operatory korzystają z zestawów jako operandów i zwracają zestaw jako wynik.

Operatorów zestawów można używać w dwóch sytuacjach:

- Aby wykonać operację zestawu na identyfikatorach zestawów reprezentujących zestawy rekordów w danych.
- Aby wykonać operację zestawu na zestawach elementów, na wartościach pól lub wewnątrz modyfikatora zestawu.

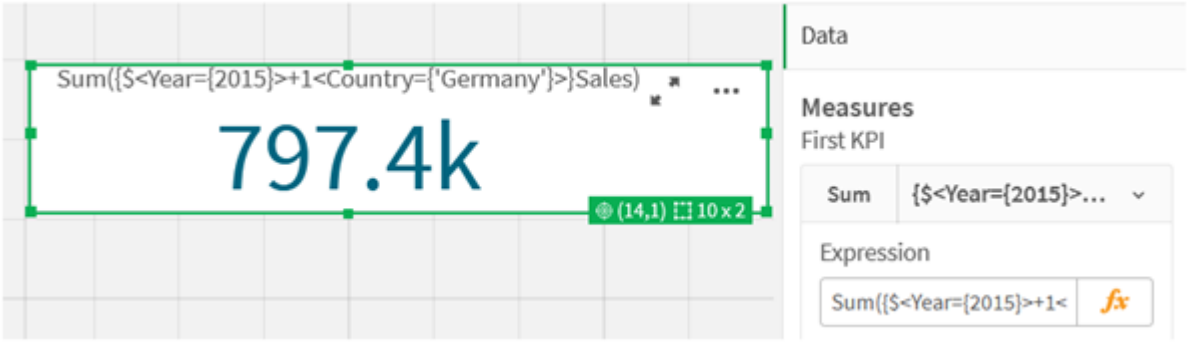
Wykonaj następujące czynności:

W arkuszu utwórz lub skopiuj następujące wyrażenie zestawu:

```
Sum({$<Year={2015}>+1<Country={'Germany'}>}Sales)
```

W tym przypadku operator znaku plusa (+) tworzy sumę zestawów danych dla 2015 i Germany. Jak wyjaśniono w przypadku identyfikatorów zestawu powyżej, identyfikator w postaci znaku dolara ($) oznacza, że bieżące wybory będą stosowane dla pierwszego operandu, <Year={2015}>. Identyfikator 1 oznacza, że wybór zostanie zignorowany w przypadku drugiego operandu, <Country={'Germany'}>.

KPI z operatorem znaku plusa (+)

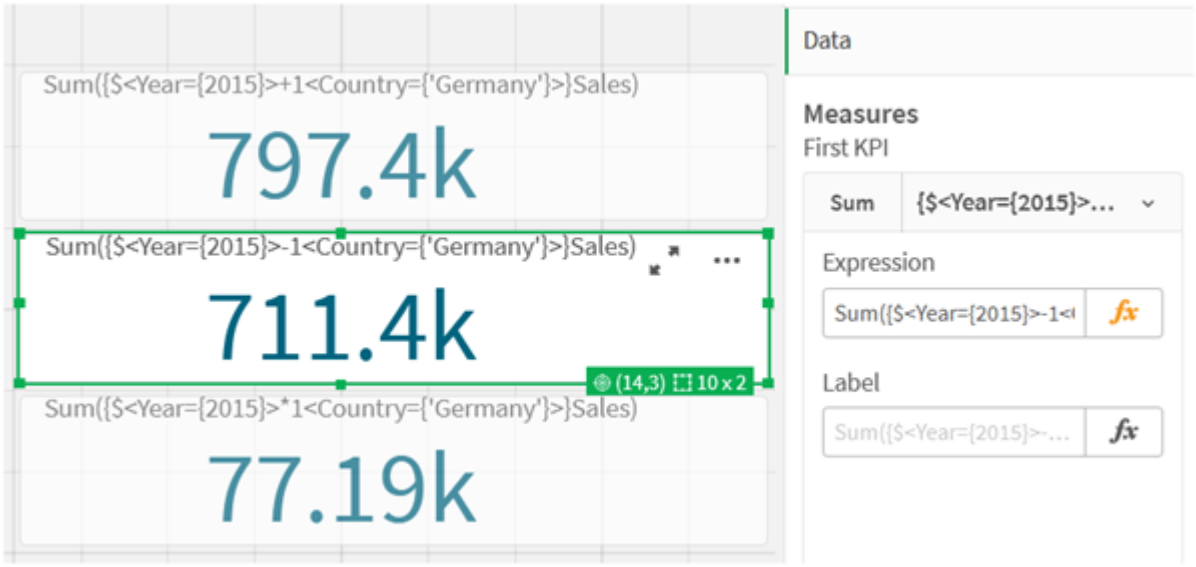


Alternatywnie użyj znaku minusa (-), aby zwrócić zestaw danych składający się z rekordów należących do roku 2015, ale nie do Germany. Możesz też użyć gwiazdki (*), aby zwrócić zestaw zawierający wszystkie rekordy należące do obu zestawów.

```
Sum({$<Year={2015}>-1<Country={'Germany'}>}Sales)
```

```
Sum({$<Year={2015}>*1<Country={'Germany'}>}Sales)
```

Wskaźniki KPI z wykorzystaniem operatorów



Dane kursu o wyrażeniach zestawu

Skrypt ładowania

Załaduj następujące dane jako ładowanie wbudowane, a następnie utwórz wyrażenia wykresu w kursie.

```
//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
];
```

Składnia wyrażeń zestawu

Pełna składnia (bez opcjonalnego stosowania nawiasów do definiowania pierwszeństwa operacji) jest opisana za pomocą notacji Backus-Naur:

```
set_expression ::= { set_entity { set_operator set_entity } }
set_entity ::= set_identifier [ set_modifier ] | set_modifier
set_identifier ::= 1 | $ | $N | $_N | bookmark_id | bookmark_name
set_operator ::= + | - | * | /
set_modifier ::= < field_selection {, field_selection } >
field_selection ::= field_name [ = | += | -= | *= | /= ] element_set_
expression
element_set_expression ::= [ - ] element_set { set_operator element_set }
element_set ::= [ field_name ] | { element_list } | element_function
element_list ::= element { , element }
element_function ::= ( P | E ) ( [set_expression] [field_name] )
element ::= field_value | " search_mask "
```

3.3 Składnia ogólna wyrażeń wykresu

Dla wyrażenia wykresu można użyć następującej ogólnej struktury składni, z wieloma opcjonalnymi parametrami:

```
expression ::= ( constant | expressionname | operator1 expression | expression operator2
expression | function | aggregation function | (expression ) )
```

gdzie:

constant to ciąg (tekst, data lub godzina) ujęty w pojedyncze cudzysłowy proste lub liczba. Stałe są zapisywane bez separatora tysięcy (z kropką jako separatorem dziesiętnym).

expressionname to nazwa (etykieta) innego wyrażenia na tym samym wykresie.

operator1 to operator jednoargumentowy, który dotyczy tylko jednego wyrażenia (znajdującego się na prawo od operatora).

operator2 to operator dwuargumentowy, który dotyczy wyrażeń znajdujących się po obu jego stronach.

```
function ::= functionname ( parameters )
parameters ::= expression { , expression }
```
Liczba i typy parametrów nie są dowolne, zależą bowiem od używanej funkcji.

```
aggregationfunction ::= aggregationfunctionname ( parameters2 )
parameters2 ::= aggrexpression { , aggrexpression }
```
Liczba i typy parametrów nie są dowolne, zależą bowiem od używanej funkcji.

3.4 Składnia ogólna agregacji

Dla agregacji można użyć następującej ogólnej struktury składni, z wieloma opcjonalnymi parametrami:

```
aggrexpression ::= ( fieldref | operator1 aggrexpression | aggrexpression operator2
aggrexpression | functioninaggr | ( aggrexpression ) )
```
fieldref to nazwa pola.

```
functionaggr ::= functionname ( parameters2 )
```
Wyrażenia i funkcje można w ten sposób dowolnie zagnieżdżać, o ile nazwa pola **fieldref** jest zawsze ujęta w ramach dokładnie jednej funkcji agregacji. Jeśli takie wyrażenie zwraca wartość dającą się zinterpretować, Qlik Sense nie wyświetla żadnych komunikatów o błędach.

4 Operatory

W tej sekcji opisano operatory używane w aplikacji Qlik Sense. Istnieją dwa rodzaje operatorów:

- operatory jednoelementowe (zawierają tylko jeden operand);
- operatory dwuargumentowe (zawierają dwa operandy).

Większość operatorów to operatory dwuargumentowe.

Można zdefiniować następujące operatory:

- Operatory bitowe
- Operatory logiczne
- Operatory liczbowe
- Operatory relacyjne
- Operatory ciągów znaków

4.1 Operatory bitowe

Wszystkie operatory bitowe dokonują konwersji (obcięcia) operandów na liczby całkowite ze znakiem (32-bitowe) i zwracają wynik w taki sam sposób. Wszystkie operacje są wykonywane na poszczególnych bitach. Jeśli operandu nie można zinterpretować jako liczby, operacja zwróci wartość NULL.

Operatory bitowe

Operator	Pełna nazwa	Opisu
bitnot	Odwrotność bitowa.	Operator jednoargumentowy. Operacja zwraca logiczną odwrotność operandu obliczoną na poszczególnych bitach. **Przykład:** `bitnot 17` zwraca -18
bitand	Bitowy operator AND.	Operacja zwraca wartość logicznej operacji AND na operandach obliczoną na poszczególnych bitach. **Przykład:** `17 bitand 7` zwraca 1
bitor	Bitowy operator OR.	Operacja zwraca wartość logicznej operacji OR na operandach obliczoną na poszczególnych bitach. **Przykład:** `17 bitor 7` zwraca 23

Operator	Pełna nazwa	Opisu
bitxor	Bitowy operator XOR.	Operacja zwraca wartość logicznej operacji XOR na operandach obliczoną na poszczególnych bitach. **Przykład:** `17 bitxor 7` zwraca 22
>>	Przesunięcie bitów w prawo.	Operacja zwraca wartość pierwszego operandu przesuniętą w prawo. Liczba kroków jest określana w drugim operandzie. **Przykład:** `8 >> 2` zwraca 2
<<	Przesunięcie bitów w lewo.	Operacja zwraca wartość pierwszego operandu przesuniętą w lewo. Liczba kroków jest określana w drugim operandzie. **Przykład:** `8 << 2` zwraca 32

4.2 Operatory logiczne

Wszystkie operatory logiczne interpretują wartości operandów jako wartości logiczne i zwracają w wyniku True (–1) lub False (0).

Operatory logiczne

Operator	Opisu
not	Odwrotność logiczna. Jeden z nielicznych operatorów jednoargumentowych. Operacja zwraca logiczną odwrotność operandu.
and	Koniunkcja logiczna (AND). Operacja zwraca wynik logicznej operacji AND na operandach.
or	Alternatywa logiczna (OR). Operacja zwraca wynik logicznej operacji OR na operandach.
Xor	Logiczna alternatywa wykluczająca (XOR). Operacja zwraca wynik logicznej operacji XOR na operandach. Jest to operacja podobna do OR logicznego, ale z tą różnicą, że daje w wyniku False, jeśli oba operandy mają wartość True.

4.3 Operatory liczbowe

Wszystkie operatory liczbowe używają wartości liczbowych operandów i zwracają jako wynik wartość liczbową.

Operatory liczbowe

Operator	Opisu
+	Znak liczby dodatniej (operator jednoargumentowy) lub dodawanie arytmetyczne. Operacja dwuargumentowa zwraca sumę dwóch operandów.
-	Znak liczby ujemnej (operator jednoargumentowy) lub odejmowanie arytmetyczne. Operacja jednoargumentowa zwraca operand pomnożony przez -1, a dwuargumentowa różnicę między dwoma operandami.
*	Mnożenie arytmetyczne. Operacja zwraca iloczyn dwóch operandów.
/	Dzielenie arytmetyczne. Operacja zwraca iloraz dwóch operandów.

4.4 Operatory relacyjne

Wszystkie operatory relacyjne porównują wartości operandów i zwracają w wyniku True (-1) lub False (0). Wszystkie operatory relacyjne są binarne.

Operatory relacyjne

Operator	Opisu
<	Mniejsze niż. Porównanie liczbowe jest wykonywane, jeśli obydwa operandy mogą być interpretowane jako liczby. Operacja zwraca wartość logiczną oceny porównania.
<=	Mniejsze lub równe. Porównanie liczbowe jest wykonywane, jeśli obydwa operandy mogą być interpretowane jako liczby. Operacja zwraca wartość logiczną oceny porównania.
>	Większe niż. Porównanie liczbowe jest wykonywane, jeśli obydwa operandy mogą być interpretowane jako liczby. Operacja zwraca wartość logiczną oceny porównania.
>=	Większe lub równe. Porównanie liczbowe jest wykonywane, jeśli obydwa operandy mogą być interpretowane jako liczby. Operacja zwraca wartość logiczną oceny porównania.
=	Równe. Porównanie liczbowe jest wykonywane, jeśli obydwa operandy mogą być interpretowane jako liczby. Operacja zwraca wartość logiczną oceny porównania.
<>	Nierównoważne. Porównanie liczbowe jest wykonywane, jeśli obydwa operandy mogą być interpretowane jako liczby. Operacja zwraca wartość logiczną oceny porównania.

Operator	Opisu
precedes	W odróżnieniu od operatora **<** przed wykonaniem porównania nie jest podejmowana próba liczbowej interpretacji wartości argumentów. Operacja zwraca wartość true, jeśli reprezentacja tekstowa wartości po lewej stronie operatora poprzedza według porównania ciągu znaków reprezentację tekstową wartości po prawej stronie operatora. **Przykład:** `'1 ' precedes ' 2'` zwraca wartość FALSE. `' 1' precedes ' 2'` zwraca wartość TRUE. ponieważ wartość ASCII spacji (' ') jest niższa niż wartość ASCII liczby. Przykład podobny: `'1 ' < ' 2'` zwraca TRUE `' 1' < ' 2'` zwraca wartość TRUE.
follows	W odróżnieniu od operatora **>** przed wykonaniem porównania nie jest podejmowana próba liczbowej interpretacji wartości argumentów. Operacja zwraca wartość true, jeśli reprezentacja tekstowa wartości po lewej stronie operatora następuje według porównania ciągu znaków po reprezentacji tekstowej wartości po prawej stronie operatora. **Przykład:** `' 2' follows '1'` zwraca FALSE ' 2' follows ' 1' zwraca wartość TRUE. ponieważ wartość ASCII spacji (' ') jest niższa niż wartość ASCII liczby. Przykład podobny: `' 2' > ' 1'` zwraca wartość TRUE. `' 2' > '1 '` zwraca wartość TRUE.

4.5 Operatory ciągów znaków

Istnieją dwa operatory ciągów znaków. Jeden z nich używa wartości ciągów znaków operandów i zwraca ciąg znaków jako wynik. Drugi porównuje operandy i zwraca wartość logiczną w celu wskazania wyniku porównania.

&

Konkatenacja ciągu znaków. Operacja zwraca ciąg tekstowy, który obejmuje dwa ciągi znaków operandów, jeden po drugim.

Przykład:

`'abc' & 'xyz'` zwraca wartość „abcxyz”

like

Porównanie ciągów znaków z użyciem symboli wieloznacznych. Operacja zwraca logiczną wartość True (-1), jeśli ciąg znaków przed operatorem jest taki sam jak ciąg znaków po operatorze. Drugi ciąg znaków może zawierać symbole wieloznaczne * (zastępuje dowolną liczbę dowolnych znaków) albo ? (zastępuje jeden dowolny znak)

Przykład:

`'abc' like 'a*'` zwraca True (-1)

`'abcd' like 'a?c*'` zwraca True (-1)

`'abc' like 'a??bc'` zwraca False (0)

5 Funkcje skryptów i wykresów

Dane można przekształcać i agregować przy użyciu funkcji w skryptach ładowania danych, jak i wyrażeniach wykresu.

Wiele funkcji można stosować w ten sam sposób zarówno w skryptach ładowania danych, jak i wyrażeniach wykresu, ale istnieje kilka wyjątków:

- Niektóre funkcje można stosować tylko w skryptach ładowania danych; są one oznaczone jako funkcje skryptu.
- Niektóre funkcje można stosować tylko w wyrażeniach wykresu; są one oznaczone jako funkcje wykresu.
- Niektóre funkcje można stosować w skryptach ładowania danych i wyrażeniach wykresu, ale z różnicami w parametrach i zastosowaniach. Są one opisane w oddzielnych tematach oznaczonych jako funkcja skryptu lub funkcja wykresu.

5.1 Połączenia analityczne dla rozszerzeń po stronie serwera (SSE)

Funkcje włączone przez połączenia analityczne będą widoczne tylko po skonfigurowaniu połączeń analitycznych i uruchomieniu programu Qlik Sense.

Połączenia analityczne można konfigurować w QMC – informacje zawiera temat „Tworzenie połączenia analitycznego” w podręczniku Zarządzanie lokacjami Qlik Sense.

W programie Qlik Sense Desktop połączenia analityczne są konfigurowane poprzez edycję pliku *Settings.ini* – informacje zawiera temat „Configuring analytic connections in Qlik Sense Desktop” („Konfigurowanie połączeń analitycznych w programie ”) w podręczniku Qlik Sense Desktop.

5.2 Funkcje agregacji

Rodzina funkcji określanych jako funkcje agregacji składa się z funkcji pobierających wiele wartości pola jako dane wejściowe i zwracających jeden wynik na grupę. Grupowanie jest natomiast definiowane przez wymiar wykresu lub klauzulę **group by** w instrukcji skryptu.

Do funkcji agregacji należą: **Sum()**, **Count()**, **Min()**, **Max()** i wiele innych.

Większość funkcji agregacji może być użyta zarówno w skrypcie ładowania danych, jak i w wyrażeniach wykresu, ale różnią się one składnią.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Podczas nazywania elementu unikaj nadawania tej samej nazwy więcej niż jednemu polu, zmiennej lub mierze. Podczas rozwiązywania konfliktów między obiektami o identycznych nazwach istnieje ścisła kolejność. Kolejność ta jest odzwierciedlana we wszystkich obiektach lub kontekstach, w których są używane te elementy. Kolejność jest następująca:

- Wewnątrz agregacji pole ma pierwszeństwo przed zmienną. Etykiety miar nie są istotne w agregacjach i nie mają priorytetu.
- Poza agregacją etykieta miary ma pierwszeństwo przed zmienną, która z kolei ma pierwszeństwo przed polem.
- Dodatkowo, poza agregacją, miary można użyć ponownie przez odwołanie się do jej etykiety, chyba że etykieta jest w rzeczywistości obliczana. W takiej sytuacji miara traci na znaczeniu w celu zmniejszenia ryzyka odwoływania się do samego siebie i wtedy nazwa zawsze będzie interpretowana po pierwsze jako etykieta miary, po drugie jako nazwa pola, a po trzecie jako nazwa zmiennej.

Używanie funkcji agregacji w skrypcie ładowania danych

Funkcji agregacji można używać tylko w instrukcjach **LOAD** i **SELECT**.

Używanie funkcji agregacji w wyrażeniach wykresu

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Funkcja agregacji wykonuje agregacje na zestawie możliwych rekordów zdefiniowanym przez selekcję. Można jednak zdefiniować alternatywny zestaw rekordów za pomocą wyrażenia set w analizie zestawów.

Jak obliczane są agregacje

Funkcja agregacji agreguje w pętli rekordy określonej tabeli. Na przykład **Count**(<Field>) spowoduje policzenie rekordów w tabeli, w których znajduje się <Field>. Jeśli chcesz zagregować tylko odrębne wartości pól, musisz użyć klauzuli **distinct**, na przykład **Count**(**distinct**<Field>).

Jeśli funkcja agregacji będzie zawierać pola z różnych tabel, będzie przetwarzać w pętli rekordy iloczynu wektorowego tabel pól składowych. Ma to negatywny wpływ na wydajność i z tego powodu należy unikać takich agregacji, szczególnie w przypadku dużych ilości danych.

Agregacja pól kluczowych

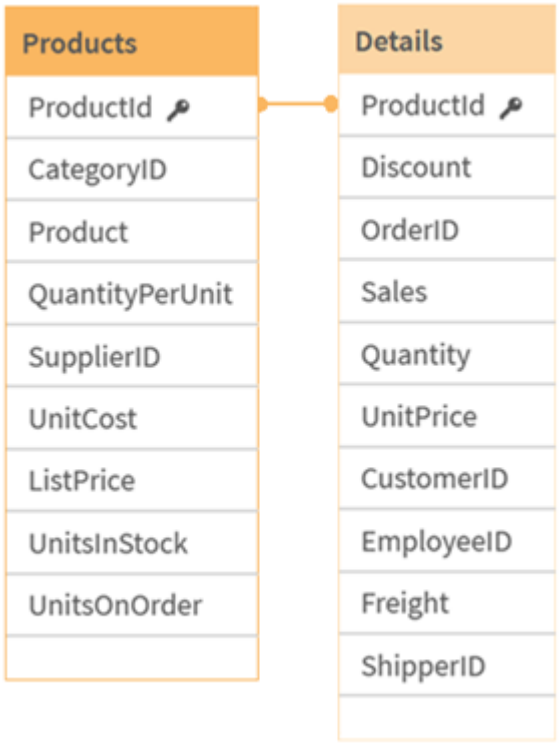
Sposób obliczania agregacji oznacza, że nie można agregować pól kluczowych, ponieważ nie jest jasne, która tabela ma być używana do agregacji. Na przykład, jeśli pole <Key> łączy dwie tabele, nie jest jasne, czy **Count**(<Key>) ma zwrócić liczbę rekordów z pierwszej czy drugiej tabeli.

Jeśli jednak użyjesz klauzuli **distinct**, agregacja będzie dobrze zdefiniowana i można ją będzie obliczyć.

Jeżeli więc użyjesz pola kluczowego wewnątrz funkcji agregacji bez klauzuli **distinct**, Qlik Sense zwróci liczbę, która może być bez znaczenia. Rozwiązaniem jest użycie klauzuli **distinct** lub kopii klucza znajdującej się tylko w jednej tabeli.

Na przykład w poniższych tabelach ProductID jest kluczem między tabelami.

Klucz ProductID między tabelami Products i Details



Count(ProductID) można liczyć w tabeli Products (która zawiera po jednym rekordzie na produkt – ProductID jest kluczem podstawowym) lub w tabeli Details (która najprawdopodobniej ma po kilka rekordów na produkt). Jeśli chcesz policzyć liczbę odrębnych produktów, użyj Count(distinct ProductID). Jeśli chcesz policzyć liczbę wierszy w określonej tabeli, nie używaj tego klucza.

Podstawowe funkcje agregacji

Przegląd podstawowych funkcji agregacji

Podstawowe funkcje agregacji to grupa najczęściej używanych funkcji agregacji.

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

Podstawowe funkcje agregacji w skrypcie ładowania danych

FirstSortedValue

Funkcja **FirstSortedValue()** zwraca wartość z wyrażenia określonego w argumencie **value**, który odpowiada wynikowi sortowania argumentu **sort_weight**, na przykład nazwie produktu o najniższej cenie jednostkowej. N-ta wartość w kolejności sortowania może zostać określona w argumencie **rank**. W przypadku uzyskania więcej niż jednej wartości o takim samym wyniku **sort_weight** dla podanej wartości argumentu **rank** funkcja zwraca NULL. Sortowane wartości są iterowane po liczbie rekordów (zgodnie z definicją z klauzuli **group by**) lub agregowane w pełnym zestawie danych, jeśli nie określono klauzuli

group by.

```
FirstSortedValue ([ distinct ] expression, sort_weight [, rank ])
```

Max

Funkcja **Max()** znajduje najwyższą wartość liczbową w zagregowanych danych w wyrażeniu, zgodnie z definicją z klauzuli **group by**. Podanie argumentu **rank** n pozwala na znalezienie n-tej najwyższej wartości.

```
Max ( expression[, rank])
```

Min

Funkcja **Min()** zwraca najniższą wartość liczbową w zagregowanych danych w wyrażeniu, zgodnie z definicją z klauzuli **group by**. Podanie argumentu **rank** n pozwala na znalezienie n-tej najniższej wartości.

```
Min ( expression[, rank])
```

Mode

Funkcja **Mode()** zwraca najczęściej występującą wartość w zagregowanych danych w wyrażeniu (wartość modalną), zgodnie z definicją z klauzuli **group by**. Funkcja **Mode()** może zwracać zarówno wartości liczbowe, jak i tekstowe.

```
Mode (expression )
```

Only

Funkcja **Only()** zwraca wartość tylko wtedy, gdy z agregowanych danych możliwy jest dokładnie jeden wynik. Jeśli wiersze zawierają tylko jedną wartość, jest ona zwracana. W przeciwnym wypadku zwracana jest wartość NULL. W celu wyznaczenia wartości wielu wierszy należy użyć klauzuli **group by**. Funkcja **Only()** może zwracać tylko wartości liczbowe i tekstowe.

```
Only (expression )
```

Sum

Funkcja **Sum()** oblicza sumę wartości zagregowanych w wyrażeniu, zgodnie z definicją z klauzuli **group by**.

```
Sum ([distinct]expression)
```

Podstawowe funkcje agregacji w wyrażeniach wykresu

Z funkcji agregacji wykresu można korzystać tylko w polach wyrażeń wykresu. Wyrażenie argumentu jednej funkcji agregacji nie może zawierać innej funkcji agregacji.

FirstSortedValue

Funkcja **FirstSortedValue()** zwraca wartość z wyrażenia określonego w argumencie **value**, który odpowiada wynikowi sortowania argumentu **sort_weight**, na przykład nazwie produktu o najniższej cenie jednostkowej. N-ta wartość w kolejności sortowania może zostać określona w argumencie **rank**. W przypadku uzyskania więcej niż jednej wartości o takim samym wyniku **sort_weight** dla podanej wartości argumentu **rank** funkcja zwraca NULL.

```
FirstSortedValue – funkcja wykresu([{SetExpression}] [DISTINCT] [TOTAL [<fld
{,fld}>]] value, sort_weight [,rank])
```

Max

Funkcja **Max()** zwraca najwyższą wartość z agregowanych danych. Podanie argumentu **rank** n pozwala na znalezienie n-tej najwyższej wartości.

```
Max — funkcja wykresu Funkcja Max() zwraca najwyższą wartość z agregowanych
danych. Podanie argumentu rank n pozwala na znalezienie n-tej najwyższej
wartości. Warto zapoznać się z informacjami na temat funkcji FirstSortedValue
i rangemax, których działanie jest podobne do działania funkcji Max. Max
([{SetExpression}] [TOTAL [<fld {,fld}>]] expr [,rank])
numeric ArgumentyArgumentOpisexprWyrażenie lub pole zawierające mierzone
dane.rankWartość domyślna argumentu rank wynosi 1, co odpowiada najwyższej
wartości. Dla argumentu rank równego 2 zostanie zwrócona druga wartość po
wartości najwyższej. Dla argumentu rank równego 3 zostanie zwrócona trzecia
wartość po wartości najwyższej itd.SetExpressionFunkcja agregacji będzie
domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym
przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą
wyrażenia analizy zestawów. TOTALJeśli słowo TOTAL występuje przed
argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich
możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych
należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów
wykresu. Korzystając z polecenia TOTAL [<fld {.fld}>], gdzie po
kwalifikatorze TOTAL podana jest lista nazw pól stanowiących podzbiór
zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych
wartości. DaneCustomerProductUnitSalesUnitPrice
AstridaAA416AstridaAA1015AstridaBB99BetacabBB510BetacabCC220BetacabDD-
25CanutilityAA815CanutilityCC-19Przykłady i wynikiPrzykładyWynikiMax
(UnitSales)10, ponieważ jest to najwyższa wartość w kolumnie
UnitSales.Wartość zamówienia jest obliczana na podstawie liczby sprzedanych
jednostek podanej w kolumnie (UnitSales) pomnożonej przez cenę
jednostkową.Max(UnitSales*UnitPrice)150, ponieważ jest to najwyższa wartość
wynikająca z obliczeń dotyczących wszystkich pozostałych danych w ramach
formuły: UnitSales*UnitPrice.Max(UnitSales, 2)9, czyli druga najwyższa
wartość.Max(TOTAL UnitSales)10, ponieważ kwalifikator TOTAL informuje o tym,
że znaleziono najwyższą możliwą wartość bez uwzględnienia wymiarów wykresu. W
przypadku wykresu z wymiarem Customer kwalifikator TOTAL zapewnia, że
zwracana jest maksymalna wartość z całego zestawu danych, a nie maksymalna
wartość z kolumny UnitSales dla każdego klienta.Wybierz Customer B.Max({1}
TOTAL UnitSales)10, niezależnie od dokonanej selekcji, ponieważ wyrażenie Set
Analysis {1} określa zestaw rekordów, które zostaną ocenione w kategorii ALL,
bez względu na dokonaną selekcję.Dane zastosowane w
przykładach:ProductData:LOAD * inline
[Customer|Product|UnitSales|UnitPriceAstrida|AA|4|16Astrida|AA|10|15Astrida|B
B|9|9Betacab|BB|5|10Betacab|CC|2|20Betacab|DD||25Canutility|AA|8|15Canutility
|CC||19] (delimiter is '|'); FirstSortedValue RangeMax ([{SetExpression}]
[DISTINCT] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Min
Funkcja **Min()** zwraca najniższą wartość z agregowanych danych. Podanie argumentu **rank** n pozwala na znalezienie n-tej najniższej wartości.

```
Min — funkcja wykresu([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr [,rank])
```

Mode
Funkcja **Mode()** zwraca wartość najczęściej występującą w agregowanych danych (wartość modalną). Funkcja **Mode()** może przetwarzać zarówno wartości tekstowe, jak i liczbowe.

```
Mode — funkcja wykresu ({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

Only
Funkcja **Only()** zwraca wartość tylko wtedy, gdy z agregowanych danych możliwy jest dokładnie jeden wynik. Na przykład wyrażenie szukające jedynego produktu o cenie jednostkowej równej 9 zwróci NULL, jeśli istnieje więcej niż jeden produkt o cenie jednostkowej równej 9.

```
Only — funkcja wykresu([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr)
```

Sum
Funkcja **Sum()** oblicza sumę wartości z wyrażenia lub pola dla wszystkich agregowanych danych.

```
Sum — funkcja wykresu([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr])
```

FirstSortedValue

Funkcja **FirstSortedValue()** zwraca wartość z wyrażenia określonego w argumencie **value**, który odpowiada wynikowi sortowania argumentu **sort_weight**, na przykład nazwie produktu o najniższej cenie jednostkowej. N-ta wartość w kolejności sortowania może zostać określona w argumencie **rank**. W przypadku uzyskania więcej niż jednej wartości o takim samym wyniku **sort_weight** dla podanej wartości argumentu **rank** funkcja zwraca NULL. Sortowane wartości są iterowane po liczbie rekordów (zgodnie z definicją z klauzuli **group by**) lub agregowane w pełnym zestawie danych, jeśli nie określono klauzuli **group by**.

Składnia:

```
FirstSortedValue ([ distinct ] value, sort-weight [, rank ])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
value Expression	Funkcja znajduje wartość wyrażenia **value** odpowiadającą wynikowi sortowania argumentu **sort_weight**.

Argument	Opis
sort-weight Expression	Wyrażenie zawierające dane do sortowania. Znajdowana jest pierwsza (najniższa) wartość argumentu **sort_weight**, na podstawie której określana jest wartość wyrażenia podanego argumentem **value**. Jeśli przed argumentem **sort_weight** zostanie podany znak minusa, funkcja zwróci ostatnią (najwyższą) wartość z sortowania.
rank Expression	Podanie dla parametru **rank** wartości „n” większej niż 1 spowoduje zwrócenie n-tej wartości w kolejności sortowania.
distinct	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie dodaj do arkusza w swojej aplikacji co najmniej pola wyszczególnione w kolumnie wyników, aby wyświetlić wynik.

Aby uzyskać ten sam wygląd, jak w kolumnie wyników poniżej, w panelu właściwości w obszarze sortowania przełącz z wartości Autom. na Niestandardowe. Następnie usuń zaznaczenie sortowania liczbowego i alfabetycznego.

Przykłady skryptów

Przykład	Wynik

`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`
	Funkcja sortuje wartości UnitSales od najmniejszej do największej, wyszukując wartość Customer z najmniejszą wartością UnitSales, co odpowiada najmniejszemu zamówieniu.
	
	Ponieważ CC odpowiada najmniejszemu zamówieniu (wartość UnitSales=2) dla Astrida klienta. AA odpowiada najmniejszemu zamówieniu (4) dla Betacab klienta, AA odpowiada najmniejszemu zamówieniu (8) dla Canutility klienta, a DD odpowiada najmniejszemu zamówieniu (10) dla Divadip. klienta.

Przykład	Wynik

Zakładając, że tabela **Temp** została załadowana jak w poprzednim przykładzie:
`LOAD Customer,FirstSortedValue(Product, -UnitSales) as MyProductWithLargestOrderByCustomer Resident Temp Group By Customer;`
	`Customer MyProductWithLargestOrderByCustomer`
	`Astrida AA`
	`Betacab DD`
	`Canutility CC`
	`Divadip -`
	Argument sort_weight poprzedza znak minus, dlatego funkcja sortuje wartości od największych.
	
	Ponieważ AA odpowiada największemu zamówieniu (wartość UnitSales:18) dla Astrida klienta, DD odpowiada największemu zamówieniu (12) dla Betacab klienta, a CC odpowiada największemu zamówieniu (13) dla Canutility klienta. Istnieją dwie identyczne wartości największego zamówienia (16) dla Divadip, klienta, w wyniku czego uzyskuje się wynik Null.

Zakładając, że tabela **Temp** została załadowana jak w poprzednim przykładzie:
`LOAD Customer,FirstSortedValue(distinct Product, -UnitSales) as MyProductWithSmallestOrderByCustomer Resident Temp Group By Customer;`
	`Customer MyProductWithLargestOrderByCustomer`
	`Astrida AA`
	`Betacab DD`
	`Canutility CC`
	`Divadip AA`
	Sytuacja wygląda tak samo jak w poprzednim przykładzie – jedynym wyjątkiem jest użycie kwalifikatora distinct. Powoduje to zignorowanie powielonego wyniku Divadip, umożliwiając zwrot wartości innych niż Null.

FirstSortedValue – funkcja wykresu

Funkcja **FirstSortedValue()** zwraca wartość z wyrażenia określonego w argumencie **value**, który odpowiada wynikowi sortowania argumentu **sort_weight**, na przykład nazwie produktu o najniższej cenie jednostkowej. N-ta wartość w kolejności sortowania może zostać określona w argumencie **rank**. W przypadku uzyskania więcej niż jednej wartości o takim samym wyniku **sort_weight** dla podanej wartości argumentu **rank** funkcja zwraca NULL.

Składnia:

```
FirstSortedValue([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] value, sort_weight [,rank])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
value	Pole danych wyjściowych Funkcja znajduje wartość wyrażenia **value** odpowiadającą wynikowi sortowania argumentu **sort_weight**.
sort_weight	Pole wejściowe Wyrażenie zawierające dane do sortowania. Znajdowana jest pierwsza (najniższa) wartość argumentu **sort_weight**, na podstawie której określana jest wartość wyrażenia podanego argumentem **value**. Jeśli przed argumentem **sort_weight** zostanie podany znak minusa, funkcja zwróci ostatnią (najwyższą) wartość z sortowania.
rank	Podanie dla parametru **rank** wartości „n” większej niż 1 spowoduje zwrócenie n-tej wartości w kolejności sortowania.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Przykłady i wyniki:

Dane

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15
Astrida	BB	9	9
Betacab	BB	5	10
Betacab	CC	2	20

Customer	Product	UnitSales	UnitPrice
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

Przykłady i wyniki

Przykład	Wynik
`firstsortedvalue (Product, UnitPrice)`	BB, czyli `Product` z najniższą wartością w kolumnie `UnitPrice`(9).
`firstsortedvalue (Product, UnitPrice, 2)`	BB, czyli `Product` z drugą najniższą wartością w kolumnie `UnitPrice`(10).
`firstsortedvalue (Customer, -UnitPrice, 2)`	Betacab, czyli `Customer`, który ma `Product` z drugą najwyższą wartością w kolumnie `UnitPrice`(20).
`firstsortedvalue (Customer, UnitPrice, 3)`	NULL, ponieważ są dwie wartości w kolumnie `Customer` (Astrida i Canutility) z tą samą wartością rank (trzecia najniższa wartość) w kolumnie `UnitPrice`(15). Aby zapewnić, że nie zostaną niespodziewanie zwrócone wartości null, należy skorzystać z kwalifikatora distinct.
`firstsortedvalue (Customer, -UnitPrice*UnitSales, 2)`	Canutility, czyli `Customer` z drugą najwyższą wartością w kolumnie `UnitPrice` pomnożoną przez wartość w kolumnie `UnitSales` (120).

Dane zastosowane w przykładach:

```
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

Funkcja **Max()** znajduje najwyższą wartość liczbową w zagregowanych danych w wyrażeniu, zgodnie z definicją z klauzuli **group by**. Podanie argumentu **rank** n pozwala na znalezienie n-tej najwyższej wartości.

Składnia:

```
Max ( expr [, rank])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
expr Expression	Wyrażenie lub pole zawierające mierzone dane.
rank Expression	Wartość domyślna argumentu **rank** wynosi 1, co odpowiada najwyższej wartości. Dla argumentu **rank** równego 2 zostanie zwrócona druga wartość po wartości najwyższej. Dla argumentu **rank** równego 3 zostanie zwrócona trzecia wartość po wartości najwyższej itd.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie dodaj do arkusza w swojej aplikacji co najmniej pola wyszczególnione w kolumnie wyników, aby wyświetlić wynik.

Aby uzyskać ten sam wygląd, jak w kolumnie wyników poniżej, w panelu właściwości w obszarze sortowania przełącz z wartości Autom. na Niestandardowe. Następnie usuń zaznaczenie sortowania liczbowego i alfabetycznego.

Przykład:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');

Max:
LOAD Customer, Max(UnitSales) as MyMax Resident Temp Group By Customer;
```

Tabela wynikowa

Customer	MyMax
Astrida	18
Betacab	5
Canutility	8

Przykład:

Zakładając, że tabela **Temp** została załadowana jak w poprzednim przykładzie:

```
LOAD Customer, Max(UnitSales,2) as MyMaxRank2 Resident Temp Group By Customer;
```

Tabela wynikowa

Customer	MyMaxRank2
Astrida	10
Betacab	4
Canutility	-

Max – funkcja wykresu

Funkcja **Max()** zwraca najwyższą wartość z agregowanych danych. Podanie argumentu **rank** n pozwala na znalezienie n-tej najwyższej wartości.

*Warto zapoznać się z informacjami na temat funkcji **FirstSortedValue** i **rangemax**, których działanie jest podobne do działania funkcji **Max**.*

Składnia:

```
Max([{SetExpression}] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
rank	Wartość domyślna argumentu **rank** wynosi 1, co odpowiada najwyższej wartości. Dla argumentu **rank** równego 2 zostanie zwrócona druga wartość po wartości najwyższej. Dla argumentu **rank** równego 3 zostanie zwrócona trzecia wartość po wartości najwyższej itd.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.

Argument	Opis
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Przykłady i wyniki:

Dane

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

Przykłady i wyniki

Przykłady	Wyniki
`Max(UnitSales)`	10, ponieważ jest to najwyższa wartość w kolumnie `UnitSales`.
Wartość zamówienia jest obliczana na podstawie liczby sprzedanych jednostek podanej w kolumnie (`UnitSales`) pomnożonej przez cenę jednostkową. `Max(UnitSales*UnitPrice)`	150, ponieważ jest to najwyższa wartość wynikająca z obliczeń dotyczących wszystkich pozostałych danych w ramach formuły: `UnitSales*UnitPrice`.
`Max(UnitSales, 2)`	9, czyli druga najwyższa wartość.

Przykłady	Wyniki
`Max(TOTAL UnitSales)`	10, ponieważ kwalifikator TOTAL informuje o tym, że znaleziono najwyższą możliwą wartość bez uwzględnienia wymiarów wykresu. W przypadku wykresu z wymiarem Customer kwalifikator TOTAL zapewnia, że zwracana jest maksymalna wartość z całego zestawu danych, a nie maksymalna wartość z kolumny UnitSales dla każdego klienta.
Wybierz `Customer B.` `Max({1} TOTAL UnitSales)`	10, niezależnie od dokonanej selekcji, ponieważ wyrażenie Set Analysis {1} określa zestaw rekordów, które zostaną ocenione w kategorii ALL, bez względu na dokonaną selekcję.

Dane zastosowane w przykładach:

```
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 '|');
```

Zob. także:

p *FirstSortedValue – funkcja wykresu (page 321)*
p *RangeMax (page 1321)*

Min

Funkcja **Min()** zwraca najniższą wartość liczbową w zagregowanych danych w wyrażeniu, zgodnie z definicją z klauzuli **group by**. Podanie argumentu **rank** n pozwala na znalezienie n-tej najniższej wartości.

Składnia:

```
Min ( expr [, rank])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
expr Expression	Wyrażenie lub pole zawierające mierzone dane.
rank Expression	Wartość domyślna argumentu **rank** wynosi 1, co odpowiada najniższej wartości. Podanie wartości 2 argumentu **rank** spowoduje zwrócenie drugiej wartości po najniższej. Dla argumentu **rank** równego 3 zostanie zwrócona trzecia wartość po najniższej itd.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie dodaj do arkusza w swojej aplikacji co najmniej pola wyszczególnione w kolumnie wyników, aby wyświetlić wynik.

Aby uzyskać ten sam wygląd, jak w kolumnie wyników poniżej, w panelu właściwości w obszarze sortowania przełącz z wartości Autom. na Niestandardowe. Następnie usuń zaznaczenie sortowania liczbowego i alfabetycznego.

Przykład:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');
Min:
LOAD Customer, Min(UnitSales) as MyMin Resident Temp Group By Customer;
```

Tabela wynikowa

Customer	MyMin
Astrida	2
Betacab	4
Canutility	8

Przykład:

Zakładając, że tabela **Temp** została załadowana jak w poprzednim przykładzie:

```
LOAD Customer, Min(UnitSales,2) as MyMinRank2 Resident Temp Group By Customer;
```

Tabela wynikowa

Customer	MyMinRank2
Astrida	9
Betacab	5
Canutility	-

Min – funkcja wykresu

Funkcja **Min()** zwraca najniższą wartość z agregowanych danych. Podanie argumentu **rank** n pozwala na znalezienie n-tej najniższej wartości.

*Warto zapoznać się z informacjami na temat funkcji **FirstSortedValue** i **rangemin**, których działanie jest podobne do działania funkcji **Min**.*

Składnia:

```
Min({[SetExpression] [TOTAL [<fld {,fld}>]]} expr [,rank])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
rank	Wartość domyślna argumentu **rank** wynosi 1, co odpowiada najniższej wartości. Podanie wartości 2 argumentu **rank** spowoduje zwrócenie drugiej wartości po najniższej. Dla argumentu **rank** równego 3 zostanie zwrócona trzecia wartość po najniższej itd.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.

Argument	Opis
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Przykłady i wyniki:

Dane

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

Funkcja Min() musi zwracać wartość inną niż NULL z szeregu wartości określonych przez ewentualne wyrażenie. Ze względu na fakt, że w podanych przykładach wśród danych występują wartości NULL, funkcja ta zwraca pierwszą wartość niebędącą wartością NULL określoną na podstawie danego wyrażenia.

Przykłady i wyniki

Przykłady	Wyniki
`Min(UnitSales)`	2, ponieważ jest to najniższa wartość niebędąca wartością NULL w kolumnie `UnitSales`.

Przykłady	Wyniki
Wartość zamówienia jest obliczana na podstawie liczby sprzedanych jednostek podanej w kolumnie (`UnitSales`) pomnożonej przez cenę jednostkową. `Min(UnitSales*UnitPrice)`	40, ponieważ jest to najniższa wartość niebędąca wartością NULL wynikająca z obliczeń dotyczących wszystkich pozostałych danych w ramach formuły: (`UnitSales`)*(`UnitPrice`).
`Min(UnitSales, 2)`	4, czyli druga najniższa wartość (po wartościach NULL).
`Min(TOTAL UnitSales)`	2, ponieważ kwalifikator TOTAL informuje o tym, że znaleziono najniższą możliwą wartość bez uwzględnienia wymiarów wykresu. W przypadku wykresu z wymiarem Customer kwalifikator TOTAL zapewnia, że zwracana jest minimalna wartość z całego zestawu danych, a nie minimalna wartość z kolumny UnitSales dla każdego klienta.
Wybierz Customer B. `Min({1} TOTAL UnitSales)`	2, niezależnie od dokonanego wyboru Customer B. Wyrażenie {1} analizy zestawów (Set Analysis) określa zestaw rekordów, które zostaną ocenione w kategorii ALL, bez względu na dokonaną selekcję.

Dane zastosowane w przykładach:

```
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 '|');
```

Zob. także:

p *FirstSortedValue – funkcja wykresu (page 321)*
p *RangeMin (page 1325)*

Mode

Funkcja **Mode()** zwraca najczęściej występującą wartość w zagregowanych danych w wyrażeniu (wartość modalną), zgodnie z definicją z klauzuli **group by**. Funkcja **Mode()** może zwracać zarówno wartości liczbowe, jak i tekstowe.

Składnia:

```
Mode ( expr)
```

Typ zwracanych danych: dual

Argumenty

Argument	Opisu
expr Expression	Wyrażenie lub pole zawierające mierzone dane.

Ograniczenia:

Jeśli jest więcej niż jedna najczęstsza wartość, zwracane jest NULL.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie dodaj do arkusza w swojej aplikacji co najmniej pola wyszczególnione w kolumnie wyników, aby wyświetlić wynik.

Aby uzyskać ten sam wygląd, jak w kolumnie wyników poniżej, w panelu właściwości w obszarze sortowania przełącz z wartości Autom. na Niestandardowe. Następnie usuń zaznaczenie sortowania liczbowego i alfabetycznego.

Przykłady skryptów

Przykład	Wynik
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 ponieważ AA jest jedynym produktem, który został sprzedany więcej niż raz.

Mode – funkcja wykresu

Funkcja **Mode()** zwraca wartość najczęściej występującą w agregowanych danych (wartość modalną). Funkcja **Mode()** może przetwarzać zarówno wartości tekstowe, jak i liczbowe.

Składnia:

```
Mode({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Przykłady i wyniki:

Dane

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

Przykłady i wyniki

Przykłady	Wyniki
`Mode(UnitPrice)` Wybierz `Customer A`.	15, ponieważ jest to najczęściej występująca wartość w kolumnie `UnitSales`. Zwraca wartość NULL (-). Żadna wartość pojedyncza nie występuje częściej niż pozostałe.
`Mode(Product)` Wybierz `Customer A`.	AA, ponieważ jest to najczęściej występująca wartość w kolumnie `Product`. Zwraca wartość NULL (-). Żadna wartość pojedyncza nie występuje częściej niż pozostałe.
`Mode` `(TOTAL UnitPrice)`	15, ponieważ kwalifikator TOTAL informuje o tym, że najczęściej występującą wartością jest nadal 15, nawet bez uwzględnienia wymiarów wykresu.

Wybierz `Customer B`.
 	
`Mode({1}`
`TOTAL UnitPrice)`
	15, niezależnie od dokonanej selekcji, ponieważ wyrażenie Set Analysis {1} określa zestaw rekordów, które zostaną ocenione w kategorii ALL, bez względu na dokonaną selekcję.

Dane zastosowane w przykładach:

```
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 '|');
```

Zob. także:

p *Avg – funkcja wykresu (page 389)*
p *Median – funkcja wykresu (page 427)*

Only

Funkcja **Only()** zwraca wartość tylko wtedy, gdy z agregowanych danych możliwy jest dokładnie jeden wynik. Jeśli wiersze zawierają tylko jedną wartość, jest ona zwracana. W przeciwnym wypadku zwracana jest wartość NULL. W celu wyznaczenia wartości wielu wierszy należy użyć klauzuli **group by**. Funkcja **Only()** może zwracać tylko wartości liczbowe i tekstowe.

Składnia:

```
Only ( expr )
```

Typ zwracanych danych: dual

Argumenty

Argument	Opis
expr Expression	Wyrażenie lub pole zawierające mierzone dane.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie dodaj do arkusza w swojej aplikacji co najmniej pola wyszczególnione w kolumnie wyników, aby wyświetlić wynik.

Aby uzyskać ten sam wygląd, jak w kolumnie wyników poniżej, w panelu właściwości w obszarze sortowania przełącz z wartości Autom. na Niestandardowe. Następnie usuń zaznaczenie sortowania liczbowego i alfabetycznego.

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');
Only:
LOAD Customer, Only(CustomerID) as MyUniqIDCheck Resident Temp Group By Customer;
```

Tabela wynikowa

Customer	MyUniqIDCheck
Astrida	1 ponieważ tylko jeden klient Astrida ma pełne rekordy zawierające CustomerID.

Only – funkcja wykresu

Funkcja **Only()** zwraca wartość tylko wtedy, gdy z agregowanych danych możliwy jest dokładnie jeden wynik. Na przykład wyrażenie szukające jedynego produktu o cenie jednostkowej równej 9 zwróci NULL, jeśli istnieje więcej niż jeden produkt o cenie jednostkowej równej 9.

Składnia:

```
Only([{SetExpression}] [TOTAL [<fld {,fld}>]] expr)
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Z funkcji Only() należy skorzystać, jeśli w przypadku występowania w danych z próby wielu tych samych wartości ma zostać zwrócona wartość NULL.

Przykłady i wyniki:

Dane

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

Przykłady i wyniki

Przykłady	Wyniki
`Only ({<UnitPrice={9}>} Product)`	BB, ponieważ jest to jedyna wartość w kolumnie `Product`, dla której wartość w kolumnie `UnitPrice` wynosi 9.
`Only({<Product={DD}>} Customer)`	Betacab, ponieważ jest to jedyny `Customer`, który sprzedaje `Product` o nazwie „DD”.
`Only ({<UnitPrice={20}>} UnitSales)`	Wartość w kolumnie `UnitSales`, dla której wartość w kolumnie `UnitPrice` wynosi 20, to 2, ponieważ jest tylko jedna wartość w kolumnie `UnitSales`, dla której wartość w kolumnie `UnitPrice` wynosi 20.
`Only ({<UnitPrice={15}>} UnitSales)`	NULL, ponieważ istnieją dwie wartości w kolumnie `UnitSales`, dla których wartość w kolumnie `UnitPrice` wynosi 15.

Dane zastosowane w przykładach:

```
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

Funkcja **Sum()** oblicza sumę wartości zagregowanych w wyrażeniu, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
sum ( [ distinct] expr)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
distinct	Jeśli słowo **distinct** występuje przed wyrażeniem, wówczas wszystkie duplikaty zostaną pominięte.
expr Expression	Wyrażenie lub pole zawierające mierzone dane.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie dodaj do arkusza w swojej aplikacji co najmniej pola wyszczególnione w kolumnie wyników, aby wyświetlić wynik.

Aby uzyskać ten sam wygląd, jak w kolumnie wyników poniżej, w panelu właściwości w obszarze sortowania przełącz z wartości Autom. na Niestandardowe. Następnie usuń zaznaczenie sortowania liczbowego i alfabetycznego.

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');
Sum:
LOAD Customer, Sum(UnitSales) as MySum Resident Temp Group By Customer;
```

Tabela wynikowa

Customer	MySum
Astrida	39
Betacab	9
Canutility	8

Sum – funkcja wykresu

Funkcja **Sum()** oblicza sumę wartości z wyrażenia lub pola dla wszystkich agregowanych danych.

Składnia:

```
Sum([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.

Argument	Opis
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane. *Kwalifikator DISTINCT jest obsługiwany, należy jednak korzystać z niego bardzo ostrożnie, ponieważ może sprawić, że odbiorca uzna, że pokazywana jest wartość łączna, podczas gdy w rzeczywistości niektóre dane zostały pominięte.*
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Przykłady i wyniki:

Dane

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

Przykłady i wyniki

Przykłady	Wyniki
`Sum(UnitSales)`	38. Łączna suma wartości z kolumny `UnitSales`.

Przykłady	Wyniki
`Sum(UnitSales*UnitPrice)`	505. Łączna suma wartości z kolumny `UnitPrice` pomnożonych przez wartości z kolumny `UnitSales`.
`Sum (TOTAL UnitSales*UnitPrice)`	505 dla wszystkich wierszy w tabeli oraz jako łączna suma, ponieważ kwalifikator TOTAL informuje o tym, że suma nadal wynosi 505, bez uwzględnienia wymiarów wykresu.
Wybierz `Customer B`. `Sum({1} TOTAL UnitSales*UnitPrice)`	505, niezależnie od dokonanej selekcji, ponieważ wyrażenie Set Analysis {1} określa zestaw rekordów, które zostaną ocenione w kategorii ALL, bez względu na dokonaną selekcję.

Dane zastosowane w przykładach:

```
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 '|');
```

Licznikowe funkcje agregacji

Licznikowe funkcje agregacji zwracają różne typy obliczeń wyrażenia po liczbie rekordów w skrypcie ładowania danych lub liczbie wartości w wymiarze wykresu.

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

Licznikowe funkcje agregacji dla ciągów w skrypcie ładowania danych

Count

Funkcja **Count()** zwraca liczbę wartości zagregowanych w wyrażeniu, zgodnie z definicją z klauzuli **group by**.

```
Count ([distinct ] expression | * )
```

MissingCount

Funkcja **MissingCount()** zwraca liczbę brakujących wartości zagregowanych w wyrażeniu, zgodnie z definicją z klauzuli **group by**.

```
MissingCount ([ distinct ] expression)
```

NullCount

Funkcja **NullCount()** zwraca liczbę wartości NULL zagregowanych w wyrażeniu, zgodnie z definicją z klauzuli **group by**.

```
NullCount ([ distinct ] expression)
```

NumericCount

Funkcja **NumericCount()** zwraca liczbę wartości liczbowych znalezionych w wyrażeniu, zgodnie z definicją z klauzuli **group by**.

```
NumericCount ([ distinct ] expression)
```

TextCount

Funkcja **TextCount()** zwraca liczbę wartości pól będących wartościami nienumerycznymi zagregowanymi w wyrażeniu, zgodnie z definicją z klauzuli **group by**.

```
TextCount ([ distinct ] expression)
```

Licznikowe funkcje agregacji w wyrażeniach wykresu

W wykresach mogą być stosowane następujące licznikowe funkcje agregacji:

Count

Funkcja **Count()** służy do agregowania liczby wartości (tekstowych i liczbowych) w poszczególnych wymiarach wykresu.

```
Count — funkcja wykresu({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]}
expr)
```

MissingCount

Funkcja **MissingCount()** służy do agregowania liczby brakujących wartości w poszczególnych wymiarach wykresu. Wartości brakujące to wszystkie wartości nieliczbowe.

```
MissingCount — funkcja wykresu({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]] expr)
```

NullCount

Funkcja **NullCount()** służy do agregowania liczby wartości NULL w poszczególnych wymiarach wykresu.

```
NullCount — funkcja wykresu({[SetExpression][DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

NumericCount

Funkcja **NumericCount()** agreguje liczbę wartości liczbowych w poszczególnych wymiarach wykresu.

```
NumericCount — funkcja wykresu({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

TextCount

Funkcja **TextCount()** służy do agregowania liczby nieliczbowych wartości pól w poszczególnych wymiarach wykresu.

```
TextCount — funkcja wykresu({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

Count

Funkcja **Count()** zwraca liczbę wartości zagregowanych w wyrażeniu, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
Count( [distinct ] expr)
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
distinct	Jeśli przed wyrażeniem występuje słowo **distinct**, wówczas wszystkie duplikaty są pomijane.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie dodaj do arkusza w swojej aplikacji co najmniej pola wyszczególnione w kolumnie wyników, aby wyświetlić wynik.

Aby uzyskać ten sam wygląd, jak w kolumnie wyników poniżej, w panelu właściwości w obszarze sortowania przełącz z wartości Autom. na Niestandardowe. Następnie usuń zaznaczenie sortowania liczbowego i alfabetycznego.

Przykłady skryptów

Przykład	Wynik
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 Jeśli w tabeli na arkuszu uwzględniono wymiar Customer, w przeciwnym wypadku wynikiem dla OrdersByCustomer jest 3, 2.
Zakładając, że tabela **Temp** została załadowana jak w poprzednim przykładzie: `LOAD Count(OrderNumber) as TotalOrderNumber Resident Temp;`	TotalOrderNumber 10
Zakładając, że tabela **Temp** została załadowana jak w pierwszym przykładzie: `LOAD Count(distinct OrderNumber) as TotalOrderNumber Resident Temp;`	TotalOrderNumber 8 Ponieważ istnieją dwie wartości OrderNumber z tą samą wartością: 1 i jedną wartością null.

Count – funkcja wykresu

Funkcja **Count()** służy do agregowania liczby wartości (tekstowych i liczbowych) w poszczególnych wymiarach wykresu.

Składnia:

```
Count({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.

Argument	Opis
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Przykłady i wyniki:

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

O ile nie podano inaczej, w poniższych przykładach założono, że selekcja obejmuje wszystkich klientów.

Przykłady i wyniki

Przykład	Wynik
`Count(OrderNumber)`	10, ponieważ istnieje dziesięć pól, które mogłyby mieć przypisaną wartość w kolumnie OrderNumber (uwzględniane są wszystkie rekordy, nawet puste). *„0” liczy się jako wartość, a nie jako pusta komórka. Jeśli jednak agregacja miary dla pewnego wymiaru da wynik 0, wymiar ten nie będzie uwzględniany na wykresach.*
`Count(Customer)`	10, ponieważ funkcja Count ocenia liczbę wystąpień we wszystkich polach.
`Count(DISTINCT [Customer])`	4, ponieważ zastosowanie kwalifikatora Distinct oznacza, że funkcja Count ocenia wyłącznie niepowtarzalne wystąpienia.
Przy założeniu, że wybrano klienta Canutility `Count (OrderNumber)/Count ({1} TOTAL OrderNumber)`	0,2, ponieważ wyrażenie to zwraca liczbę zamówień wybranego klienta jako procent zamówień wszystkich klientów. W tym przypadku jest to 2/10.
Przy założeniu, że wybrano klientów Astrida i Canutility `Count(TOTAL <Product> OrderNumber)`	5, ponieważ jest to liczba zamówień złożonych na produkty wybranych klientów (uwzględniane są również komórki puste).

Dane zastosowane w przykładach:

```
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

Funkcja **MissingCount()** zwraca liczbę brakujących wartości zagregowanych w wyrażeniu, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
MissingCount ( [ distinct ] expr)
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
expr Expression	Wyrażenie lub pole zawierające mierzone dane.
distinct	Jeśli przed wyrażeniem występuje słowo **distinct**, wówczas wszystkie duplikaty są pomijane.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie dodaj do arkusza w swojej aplikacji co najmniej pola wyszczególnione w kolumnie wyników, aby wyświetlić wynik.

Aby uzyskać ten sam wygląd, jak w kolumnie wyników poniżej, w panelu właściwości w obszarze sortowania przełącz z wartości Autom. na Niestandardowe. Następnie usuń zaznaczenie sortowania liczbowego i alfabetycznego.

Przykłady skryptów

Przykład	Wynik
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 Druga instrukcja daje: TotalMissingCount 3 w tabeli z tym wymiarem.

Przykład	Wynik

Zakładając, że tabela **Temp** została załadowana jak w poprzednim przykładzie:
`LOAD MissingCount(distinct OrderNumber) as TotalMissingCountDistinct Resident Temp;`
	`TotalMissingCountDistinct`
	`1`
	Ponieważ brakuje tylko jednej wartości OrderNumber.

MissingCount – funkcja wykresu

Funkcja **MissingCount()** służy do agregowania liczby brakujących wartości w poszczególnych wymiarach wykresu. Wartości brakujące to wszystkie wartości nieliczbowe.

Składnia:

```
MissingCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Przykłady i wyniki:

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16
Astrida	AA	7	10	15

Customer	Product	OrderNumber	UnitSales	Unit Price
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

Przykłady i wyniki

Przykład	Wynik
`MissingCount([OrderNumber])`	3, ponieważ trzy pola z dziesięciu pól w kolumnie OrderNumber są puste. *„0” liczy się jako wartość, a nie jako pusta komórka. Jeśli jednak agregacja miary dla pewnego wymiaru da wynik 0, wymiar ten nie będzie uwzględniany na wykresach.*
`MissingCount ([OrderNumber])/MissingCount ({1} Total [OrderNumber])`	Wyrażenie to zwraca liczbę niekompletnych zamówień wybranego klienta jako ułamek niekompletnych zamówień wszystkich klientów. W kolumnie OrderNumber brakuje łącznie trzech wartości dla wszystkich klientów. Dla każdego klienta z kolumny Customer, w przypadku którego brakuje wartości w kolumnie Product, wynik wynosi zatem 1/3.

Dane zastosowane w przykładzie:

```
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

Funkcja **NullCount()** zwraca liczbę wartości NULL zagregowanych w wyrażeniu, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
NullCount ( [ distinct ] expr)
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
expr Expression	Wyrażenie lub pole zawierające mierzone dane.
distinct	Jeśli przed wyrażeniem występuje słowo **distinct**, wówczas wszystkie duplikaty są pomijane.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie dodaj do arkusza w swojej aplikacji co najmniej pola wyszczególnione w kolumnie wyników, aby wyświetlić wynik.

Aby uzyskać ten sam wygląd, jak w kolumnie wyników poniżej, w panelu właściwości w obszarze sortowania przełącz z wartości Autom. na Niestandardowe. Następnie usuń zaznaczenie sortowania liczbowego i alfabetycznego.

Przykłady skryptów

Przykład	Wynik
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 Druga instrukcja daje: TotalNullCount 1 w tabeli z tym wymiarem, ponieważ tylko jeden rekord zawiera wartość Null.

NullCount – funkcja wykresu

Funkcja **NullCount()** służy do agregowania liczby wartości NULL w poszczególnych wymiarach wykresu.

Składnia:

```
NullCount({[SetExpression][DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
set_ expression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.

Argument	Opis
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Przykłady i wyniki:

Przykłady i wyniki

Przykład	Wynik
NullCount ([OrderNumber])	1, ponieważ wprowadzono wartość null za pomocą parametru NullInterpret we wbudowanej instrukcji **LOAD**.

Dane zastosowane w przykładzie:

```
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

Funkcja **NumericCount()** zwraca liczbę wartości liczbowych znalezionych w wyrażeniu, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
NumericCount ( [ distinct ] expr)
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
expr Expression	Wyrażenie lub pole zawierające mierzone dane.
distinct	Jeśli przed wyrażeniem występuje słowo **distinct**, wówczas wszystkie duplikaty są pomijane.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie dodaj do arkusza w swojej aplikacji co najmniej pola wyszczególnione w kolumnie wyników, aby wyświetlić wynik.

Aby uzyskać ten sam wygląd, jak w kolumnie wyników poniżej, w panelu właściwości w obszarze sortowania przełącz z wartości Autom. na Niestandardowe. Następnie usuń zaznaczenie sortowania liczbowego i alfabetycznego.

Przykład skryptu

Przykład	Wynik
`LOAD NumericCount(OrderNumber) as TotalNumericCount Resident Temp;`	Druga instrukcja daje: TotalNumericCount 7 w tabeli z tym wymiarem.
Zakładając, że tabela **Temp** została załadowana jak w poprzednim przykładzie: `LOAD NumericCount(distinct OrderNumber) as TotalNumeriCCountDistinct Resident Temp;`	TotalNumericCountDistinct 6 Ponieważ istnieje jeden OrderNumber, który powiela inny, wynikiem jest 6, które nie są duplikatami.

Przykład:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|7|1|25
```

```
] (delimiter is '|');
NumCount1:
LOAD Customer,NumericCount(OrderNumber) as NumericCountByCustomer Resident Temp Group By
Customer;
```

Tabela wynikowa

Customer	NumericCountByCustomer
Astrida	3
Betacab	2
Canutility	0
Divadip	2

NumericCount – funkcja wykresu

Funkcja **NumericCount()** agreguje liczbę wartości liczbowych w poszczególnych wymiarach wykresu.

Składnia:

```
NumericCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
set_ expression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Przykłady i wyniki:

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

O ile nie podano inaczej, w poniższych przykładach założono, że selekcja obejmuje wszystkich klientów.

Przykłady i wyniki

Przykład	Wynik
`NumericCount ([OrderNumber])`	7, ponieważ trzy pola z dziesięciu pól w kolumnie OrderNumber są puste. *„0” liczy się jako wartość, a nie jako pusta komórka. Jeśli jednak agregacja miary dla pewnego wymiaru da wynik 0, wymiar ten nie będzie uwzględniany na wykresach.*
`NumericCount ([Product])`	0, ponieważ wszystkie nazwy produktów mają postać tekstową. Za pomocą tej funkcji zazwyczaj sprawdza się, czy pola tekstowe nie zawierają wartości liczbowych.
`NumericCount (DISTINCT [OrderNumber])/Count (DISTINCT [OrderNumber)]`	Zlicza liczbę wszystkich różniących się od siebie wartości liczbowych w kolumnie OrderNumber i dzieli tę liczbę przez liczbę wartości liczbowych i nieliczbowych w kolumnie OrderNumber. Jeśli wszystkie wartości w polach mają charakter liczbowy, zwracana jest wartość 1. Za pomocą tej funkcji zazwyczaj sprawdza się, czy wszystkie wartości w polach mają charakter liczbowy. W tym przykładzie w kolumnie OrderNumber występuje siedem różniących się od siebie wartości liczbowych z ośmiu różniących się od siebie wartości liczbowych i nieliczbowych. W takim przypadku wyrażenie to zwraca wartość 0,875.

Dane zastosowane w przykładzie:

```
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

Funkcja **TextCount()** zwraca liczbę wartości pól będących wartościami nienumerycznymi zagregowanymi w wyrażeniu, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
TextCount ( [ distinct ] expr)
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
expr Expression	Wyrażenie lub pole zawierające mierzone dane.
distinct	Jeśli przed wyrażeniem występuje słowo **distinct**, wówczas wszystkie duplikaty są pomijane.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie dodaj do arkusza w swojej aplikacji co najmniej pola wyszczególnione w kolumnie wyników, aby wyświetlić wynik.

Aby uzyskać ten sam wygląd, jak w kolumnie wyników poniżej, w panelu właściwości w obszarze sortowania przełącz z wartości Autom. na Niestandardowe. Następnie usuń zaznaczenie sortowania liczbowego i alfabetycznego.

Przykład:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
```

```
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
TextCount1:
LOAD Customer,TextCount(Product) as ProductTextCount Resident Temp Group By Customer;
```

Tabela wynikowa

Customer	ProductTextCount
Astrida	3
Betacab	3
Canutility	2
Divadip	2

Przykład:

```
LOAD Customer,TextCount(OrderNumber) as OrderNumberTextCount Resident Temp Group By Customer;
```

Tabela wynikowa

Customer	OrderNumberTextCount
Astrida	0
Betacab	1
Canutility	2
Divadip	0

TextCount – funkcja wykresu

Funkcja **TextCount()** służy do agregowania liczby nieliczbowych wartości pól w poszczególnych wymiarach wykresu.

Składnia:

```
TextCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.

Argument	Opis
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Przykłady i wyniki:

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

Przykłady i wyniki

Przykład	Wynik
`TextCount ([Product])`	10, ponieważ we wszystkich dziesięciu polach w kolumnie Product znajduje się tekst. *„0" liczy się jako wartość, a nie jako pusta komórka. Jeśli jednak agregacja miary dla pewnego wymiaru da wynik 0, wymiar ten nie będzie uwzględniany na wykresach. Puste komórki są traktowane jako nietekstowe i nie są zliczane przez funkcję TextCount.*
`TextCount ([OrderNumber])`	3, ponieważ zliczane są również puste komórki. Za pomocą tej funkcji zazwyczaj sprawdza się, czy w polach liczbowych występują jakiekolwiek wartości tekstowe lub zerowe.
`TextCount (DISTINCT [Product])/Count ([Product)]`	Zlicza wszystkie odrębne wartości tekstowe Product (4) i dzieli ich liczbę przez łączną liczbę wartości w Product (10). Wynikiem jest 0,4.

Dane zastosowane w przykładzie:

```
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 '|');
```

Finansowe funkcje agregacji

W tej sekcji opisano funkcje agregacji dla operacji finansowych dotyczących płatności i przepływów pieniężnych.

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

Finansowe funkcje agregacji dla ciągów w skrypcie ładowania danych

IRR

Funkcja **IRR()** zwraca zagregowany wewnętrzny współczynnik zwrotu dotyczący serii przepływów pieniężnych reprezentowanych przez liczby w wyrażeniach iterowanych względem wielu wierszy określonych przez klauzulę group by.

```
IRR (expression)
```

XIRR

Funkcja **XIRR()** zwraca zagregowany wewnętrzny współczynnik zwrotu dotyczący harmonogramu przepływów pieniężnych (niekoniecznie okresowych) reprezentowanych przez liczby zestawione w pary w wyrażeniach **pmt** i **date** iterowanych względem wielu rekordów określonych przez klauzulę group by. Wszystkie płatności są obejmowane upustem na podstawie roku zawierającego 365 dni.

```
XIRR (valueexpression, dateexpression )
```

NPV

Funkcja skryptu **NPV()** pobiera stawkę rabatu i kilka wartości uporządkowanych według okresu. W tych obliczeniach wpływy (przychody) są dodatnie, a wypływy (przyszłe płatności) powinny być wartościami ujemnymi. Występują na końcu każdego okresu.

```
NPV (rate, expression)
```

XNPV

Funkcja skryptu **XNPV()** pobiera określone daty odpowiadające każdemu dyskontowanemu przepływowi pieniężnemu z wyjątkiem stawki rabatu. Różni się od funkcji **NPV()** tym, że **NPV()** zakłada, że wszystkie okresy są równe. Dlatego **XNPV()** jest precyzyjniejsza od **NPV()**.

```
XNPV (rate, valueexpression, dateexpression)
```

Finansowe funkcje agregacji w wyrażeniach wykresu

Te finansowe funkcje agregacji mogą być stosowane w wykresach.

IRR

Funkcja **IRR()** zwraca zagregowaną wewnętrzną stopę zwrotu dla szeregu przepływów pieniężnych reprezentowanych przez liczby w wyrażeniu podanym jako **value**, iterowanych po wymiarach wykresu.

```
IRR – funkcja wykresu[TOTAL [<fld {,fld}>]] value)
```

NPV

Funkcja **NPV()** zwraca zagregowaną wartość bieżącą netto inwestycji na podstawie stopy dyskontowej (parametr **discount_rate**) dla poszczególnych okresów oraz szeregu przyszłych płatności (wartości ujemne) i przychodów (wartości dodatnie) reprezentowanych przez liczby w parametrze **value**, iterowanych po wymiarach wykresu. Przyjmuje się, że płatności i przychody występują na koniec każdego okresu.

```
NPV – funkcja wykresu([TOTAL [<fld {,fld}>]] discount_rate, value)
```

XIRR

Funkcja **XIRR()** zwraca zagregowaną wewnętrzną stopę zwrotu dotyczącą harmonogramu przepływów pieniężnych (niekoniecznie okresowych) reprezentowanych przez połączone w pary liczby zwracane przez wyrażenia podane jako **pmt** i **date**, iterowane po wymiarach wykresu. Wszystkie płatności są obejmowane upustem na podstawie roku zawierającego 365 dni.

```
XIRR — funkcja wykresu (page 372)([TOTAL [<fld {,fld}>]] pmt, date)
```

XNPV

Funkcja **XNPV()** zwraca zagregowaną wartość bieżącą netto dotyczącą harmonogramu przepływów pieniężnych (niekoniecznie okresowych) reprezentowanych przez połączone w pary liczby zwracane przez wyrażenia podane jako **pmt** i **date**, iterowane po wymiarach wykresu. Wszystkie płatności są obejmowane upustem na podstawie roku zawierającego 365 dni.

```
XNPV — funkcja wykresu([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

IRR

Funkcja **IRR()** zwraca zagregowany wewnętrzny współczynnik zwrotu dotyczący serii przepływów pieniężnych reprezentowanych przez liczby w wyrażeniach iterowanych względem wielu wierszy określonych przez klauzulę group by.

Te przepływy pieniężne nie muszą być równe, jak w przypadku rozliczeń rocznych. Jednak przepływy pieniężne muszą odbywać się w regularnych interwałach, np. miesięcznie lub rocznie. Wewnętrzna stopa zwrotu to stopa procentowa uzyskiwana dla inwestycji składającej się z płatności (wartości ujemne) i przychodów (wartości dodatnie) występujących w regularnych okresach. W celu obliczenia funkcja musi zawierać co najmniej jedną wartość dodatnią i jedną wartość ujemną.

Składnia:

```
IRR(value)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wyrażenie lub pole zawierające mierzone dane.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące są pomijane.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

Przykłady i wyniki:

Przykłady i wyniki

Przykład	Rok	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 – funkcja wykresu

Funkcja **IRR()** zwraca zagregowaną wewnętrzną stopę zwrotu dla szeregu przepływów pieniężnych reprezentowanych przez liczby w wyrażeniu podanym jako **value**, iterowanych po wymiarach wykresu.

Te przepływy pieniężne nie muszą być równe, jak w przypadku rozliczeń rocznych. Jednak przepływy pieniężne muszą odbywać się w regularnych interwałach, np. miesięcznie lub rocznie. Wewnętrzna stopa zwrotu to stopa procentowa uzyskiwana dla inwestycji składającej się z płatności (wartości ujemne) i przychodów (wartości dodatnie) występujących w regularnych okresach. W celu obliczenia funkcja musi zawierać co najmniej jedną wartość dodatnią i jedną wartość ujemną.

Składnia:

```
IRR([TOTAL [<fld {,fld}>]] value)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wyrażenie lub pole zawierające mierzone dane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Wartości tekstowe, wartości NULL i wartości brakujące są pomijane.

Przykłady i wyniki:

Przykłady i wyniki

Przykład	Wynik
`IRR (Payments)`	0.1634 Przyjmuje się, że płatności mają charakter okresowy, na przykład miesięczny. *Pole Data jest używane w przykładzie XIRR, w którym płatności mogą być nieokresowe, pod warunkiem że zostaną podane daty dokonania płatności.*

Dane zastosowane w przykładach:

```
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 '|');
```

Zob. także:

p *XIRR – funkcja wykresu (page 372)*
p *Aggr – funkcja wykresu (page 532)*

NPV

Funkcja skryptu **NPV()** pobiera stawkę rabatu i kilka wartości uporządkowanych według okresu. W tych obliczeniach wpływy (przychody) są dodatnie, a wypływy (przyszłe płatności) powinny być wartościami ujemnymi. Występują na końcu każdego okresu.

Wartość bieżąca netto (Net Present Value – NPV) służy do obliczania bieżącej wartości łącznej przyszłego strumienia przepływów pieniężnych. Aby obliczyć NPV, należy oszacować przyszłe przepływy pieniężne dla każdego okresu i określić poprawną stopę dyskontową. Funkcja skryptu **NPV()** pobiera stawkę rabatu i kilka wartości uporządkowanych według okresu. W tych obliczeniach wpływy (przychody) są dodatnie, a wypływy (przyszłe płatności) powinny być wartościami ujemnymi. Występują na końcu każdego okresu.

Składnia:

```
NPV(discount_rate, value)
```

Typ zwracanych danych: liczbowy. Domyślnie wynik zostanie sformatowany jako waluta.

Wzór na obliczenie bieżącej wartości netto:

$$NPV = \sum_{t=1}^{n} \frac{R_t}{(1+i)^t}$$

gdzie:

- R_t = wpływy pieniężne netto - wypływy podczas pojedynczego okresu `t`
- `i` = stopa dyskontowa lub zwrot, jaki uzyskano by z alternatywnych inwestycji
- `t` = liczba okresów licznika czasu

Argumenty

Argument	Opis
discount_ rate	**discount_rate** jest wartością procentową zastosowanego rabatu. Wartość 0,1 oznaczałaby stopę dyskontową w wysokości 10%.
value	To pole przechowuje wartości dla wielu okresów uporządkowane według okresu. Przyjmuje się, że pierwsza wartość reprezentuje przepływ pieniężny na koniec okresu 1 itd.

Ograniczenia:

Funkcja `NPV()` ma następujące ograniczenia:

- Wartości tekstowe, wartości NULL i wartości brakujące są pomijane.
- Wartości przepływów pieniężnych muszą być uporządkowane według okresu w kolejności rosnącej.

Kiedy używać

`NPV()` jest funkcją finansową umożliwiającą sprawdzenie rentowności projektu oraz derywowanie innych miar. Znajduje zastosowanie w przypadku, gdy przepływy pieniężne są podane w nieprzetworzonej formie.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Pojedyncza płatność (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych jednego projektu i jego przepływy pieniężne dla jednego okresu, który został załadowany do tabeli o nazwie `CashFlow`.
- Polecenie LOAD z predykatem Resident z tabeli `CashFlow`, które jest używane do obliczania pola NPV dla projektu w tabeli o nazwie `NPV`.
- Wpisana w kod stopa dyskontowa w wysokości 10%, która jest używana w obliczeniach NPV.
- Polecenie `Group By`, które jest używane do grupowania wszystkich płatności dotyczących projektu.

Skrypt ładowania

```
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;
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `PrjId`
- `NPV`

Tabela wynikowa

PrjId	NPV
1	$909.09

Aby na koniec okresu otrzymać pojedynczą płatność na kwotę 1000 $, przy stopie dyskontowej 10% na okres, wartość NPV musi być równa 1000 $ podzielone przez (1 + stopa dyskontowa). Efektywna wartość NPV wynosi 909,09 $

Przykład 2 – Kilka płatności (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych jednego projektu i jego przepływy pieniężne dla wielu okresów, który został załadowany do tabeli o nazwie `CashFlow`.
- Polecenie LOAD z predykatem Resident z tabeli `CashFlow`, które jest używane do obliczania pola NPV dla projektu w tabeli o nazwie `NPV`.
- Wpisana w kod stopa dyskontowa w wysokości 10% (0,1), która jest używana w obliczeniach NPV.
- Polecenie `Group By`, które jest używane do grupowania wszystkich płatności dotyczących projektu.

Skrypt ładowania

```
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;
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- PrjId
- NPV

Tabela wynikowa

PrjId	NPV
1	$1735.54

Aby na koniec dwóch okresów otrzymać płatności na kwotę 1000 $, przy stopie dyskontowej 10% na okres, efektywna wartość NPV musi być równa 1735,54 $.

Przykład 3 – Kilka płatności (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Stopy dyskontowe dla dwóch projektów, które są załadowane do tabeli o nazwie `Project`.
- Przepływy pieniężne dla kilku okresów dla każdego projektu według identyfikatora projektu i identyfikatora okresu. Identyfikatora okresu można użyć do uporządkowania rekordów, jeśli dane są nieuporządkowane.
- Kombinacja `NoConcatenate`, poleceń LOAD z predykatem Resident i funkcji `Left Join` tworząca tymczasową tabelę `tmpNPV`. Tabela ta łączy rekordy z tabel `Project` i `CashFlow` w jednej płaskiej tabeli. Będzie ona zawierać stopy dyskontowe dla każdego okresu.
- Polecenie LOAD z predykatem Resident z tabeli `tmpNPV`, które jest używane do obliczania pola NPV dla każdego projektu w tabeli o nazwie `NPV`.
- Pojedyncza wartość stopy dyskontowej związana z każdym projektem. Jest otrzymywana za pomocą funkcji `only()` i jest używana w obliczeniach NPV dla każdego projektu.
- Polecenie `Group By`, które jest używane do grupowania wszystkich płatności dotyczących każdego projektu według identyfikatora projektu.

Aby uniknąć załadowania do modelu danych syntetycznych lub redundantnych, na końcu skryptu tabela `tmpNPV` zostaje usunięta.

Skrypt ładowania

```
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;
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- PrjId
- NPV

Tabela wynikowa

PrjId	NPV
1	$2486.85
2	$2042.12

W projekcie o identyfikatorze 1 oczekiwane są płatności na kwotę 1000 $ na koniec trzech okresów, przy stopie dyskontowej 10% na okres. Zatem efektywna wartość NPV wynosi 2486,85 $.

W projekcie o identyfikatorze 2 oczekiwane są dwie płatności na kwotę 500 $ i dwie kolejne na kwotę 1000 $ w czterech okresach, przezy stopie dyskontowej 15%. Zatem efektywna wartość NPV wynosi 2042,12 $.

Przykład 4 – Przykład obliczania rentowności projektu (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Stopy dyskontowe i początkowe wartości inwestycji (okres 0) dla dwóch projektów, które są załadowane do tabeli o nazwie `Project`.
- Przepływy pieniężne dla kilku okresów dla każdego projektu według identyfikatora projektu i identyfikatora okresu. Identyfikatora okresu można użyć do uporządkowania rekordów, jeśli dane są nieuporządkowane.
- Kombinacja `NoConcatenate`, poleceń LOAD z predykatem Resident i funkcji `Left Join` tworząca tymczasową tabelę `tmpNPV`. Tabela ta łączy rekordy z tabel `Project` i `CashFlow` w jednej płaskiej tabeli. Będzie ona zawierać stopy dyskontowe dla każdego okresu.
- Pojedyncza stopa dyskontowa powiązana z każdym projektem, która jest otrzymywana za pomocą funkcji `only()` i jest używana w obliczeniach NPV dla każdego projektu.
- Polecenie LOAD z predykatem Resident z tabeli tmpNPV jest używane do obliczania pola NPV dla każdego projektu w tabeli o nazwie NPV.
- Zostaje utworzone dodatkowe pole, które dzieli NPV przez początkową wartość inwestycji każdego projektu, w celu obliczenia indeksu rentowności projektu.
- Polecenie group by, grupowanie według identyfikatora projektu, jest używane do grupowania wszystkich płatności dotyczących każdego projektu.

Aby uniknąć załadowania do modelu danych syntetycznych lub redundantnych, na końcu skryptu tabela `tmpNPV` zostaje usunięta.

Skrypt ładowania

```
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;
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `PrjId`
- `NPV`

Utwórz następującą miarę:

```
=only(Profitability_Index)
```

Tabela wynikowa

PrjId	NPV	=only(Profitability_Index)
1	$87039.82	0.87
2	$123513.71	1.24

Projekt o identyfikatorze 1 ma efektywną wartość NPV 87039,82 $, a początkowa wartość inwestycji wynosi 100000 $. Zatem indeks rentowności wynosi 0,87. Ponieważ jego wartość jest niższa od 1, ten projekt jest nierentowny.

Projekt o identyfikatorze 2 ma efektywną wartość NPV 123513,71 $, a początkowa wartość inwestycji wynosi 100000 $. Zatem indeks rentowności wynosi 1,24. Ponieważ jego wartość jest większa od 1, ten projekt jest rentowny.

NPV – funkcja wykresu

Funkcja **NPV()** zwraca zagregowaną wartość bieżącą netto inwestycji na podstawie stopy dyskontowej (parametr **discount_rate**) dla poszczególnych okresów oraz szeregu przyszłych płatności (wartości ujemne) i przychodów (wartości dodatnie) reprezentowanych przez liczby w parametrze **value**,

iterowanych po wymiarach wykresu. Przyjmuje się, że płatności i przychody występują na koniec każdego okresu.

Składnia:

```
NPV([TOTAL [<fld {,fld}>]] discount_rate, value)
```

Typ zwracanych danych: numeric Domyślnie wynik zostanie sformatowany jako waluta.

Argumenty:

Argumenty

Argument	Opisu
discount_rate	**discount_rate** is the rate of discount over the length of the period.**discount_rate** jest wartością procentową zastosowanego rabatu.
value	Wyrażenie lub pole zawierające mierzone dane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości. Po kwalifikatorze **TOTAL** może następować lista zawierająca co najmniej jedną nazwę pola w nawiasach trójkątnych. Te nazwy pól powinny być podzbiorem zmiennych wymiaru wykresu. W tym przypadku obliczenie jest wykonywane z pominięciem wszystkich zmiennych wymiaru wykresu z wyjątkiem zmiennych z listy, tj. dla każdej kombinacji wartości pól dla pól wymiaru z listy jest zwracana jedna wartość. Lista może zawierać także pola, które aktualnie nie są wymiarem na wykresie. Jest to użyteczne w przypadku wymiarów grupowych, gdy pola wymiarów nie są niezmienne. Lista zawiera wszystkie zmienne z grupy, co powoduje, że funkcja działa w przypadku zmiany poziomu drążenia.

Ograniczenia:

Argumenty **discount_rate** i **value** nie mogą zawierać funkcji agregacji, chyba że takie wewnętrzne funkcje agregacji zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Wartości tekstowe, wartości NULL i wartości brakujące są pomijane.

Przykłady i wyniki:

Przykłady i wyniki

Przykład	Wynik
`NPV(Discount, Payments)`	-$540.12

Dane zastosowane w przykładach:

```
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 '|');
```

Zob. także:

p *XNPV – funkcja wykresu (page 379)*
p *Aggr – funkcja wykresu (page 532)*

XIRR

Funkcja **XIRR()** zwraca zagregowany wewnętrzny współczynnik zwrotu dotyczący harmonogramu przepływów pieniężnych (niekoniecznie okresowych) reprezentowanych przez liczby zestawione w pary w wyrażeniach **pmt** i **date** iterowanych względem wielu rekordów określonych przez klauzulę group by. Wszystkie płatności są obejmowane upustem na podstawie roku zawierającego 365 dni.

Składnia:

```
XIRR(pmt, date )
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
pmt	Płatności. Wyrażenie lub pole zawierające płatności w ramach przepływów pieniężnych odpowiadające harmonogramowi płatności podanemu argumentem **date**.
date	Wyrażenie lub pole zawierające harmonogram dat odpowiadających płatnościom w ramach przepływów pieniężnych podanym argumentem **pmt**.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej albo obydwu częściach pary danych spowodują pominięcie całej pary danych.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

Przykłady i wyniki

Przykład	Rok	XIRR2013
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,XIRR(Payments, Date) as XIRR2013 Resident Cashflow Group By Year;	2013	0.5385

XIRR – funkcja wykresu

Funkcja **XIRR()** zwraca zagregowaną wewnętrzną stopę zwrotu dotyczącą harmonogramu przepływów pieniężnych (niekoniecznie okresowych) reprezentowanych przez połączone w pary liczby zwracane przez wyrażenia podane jako **pmt** i **date**, iterowane po wymiarach wykresu. Wszystkie płatności są obejmowane upustem na podstawie roku zawierającego 365 dni.

Składnia:

```
XIRR([TOTAL [<fld {,fld}>]] pmt, date)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
pmt	Płatności. Wyrażenie lub pole zawierające płatności w ramach przepływów pieniężnych odpowiadające harmonogramowi płatności podanemu argumentem **date**.
date	Wyrażenie lub pole zawierające harmonogram dat odpowiadających płatnościom w ramach przepływów pieniężnych podanym argumentem **pmt**.

Argument	Opisu
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Argumenty **pmt** i **date** nie mogą zawierać funkcji agregacji, chyba że takie wewnętrzne funkcje agregacji zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Przykłady i wyniki:

Przykłady i wyniki

Przykład	Wynik
`XIRR(Payments, Date)`	0.5385

Dane zastosowane w przykładach:

```
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 '|');
```

Zob. także:

p *IRR – funkcja wykresu (page 361)*
p *Aggr – funkcja wykresu (page 532)*

XNPV

Funkcja skryptu **XNPV()** pobiera określone daty odpowiadające każdemu dyskontowanemu przepływowi pieniężnemu z wyjątkiem stawki rabatu. Różni się od funkcji **NPV()** tym, że **NPV ()** zakłada, że wszystkie okresy są równe. Dlatego **XNPV()** jest precyzyjniejsza od **NPV()**.

Składnia:

```
XNPV(discount_rate, pmt, date)
```

Typ zwracanych danych: liczbowy. Domyślnie wynik zostanie sformatowany jako waluta.

Wzór na obliczanie XNPV:

$$XNPV = \sum_{i=1}^{n} \frac{P_i}{(1+rate)^{(di-d1)/365}}$$

gdzie:

- P_i = wpływy pieniężne netto - wypływy podczas pojedynczego okresu `i`
- d_1 = data pierwszej płatności
- d_i = data `i`-tej płatności
- `rate` = stopa dyskontowa

Wartość bieżąca netto (Net Present Value – NPV) służy do obliczania bieżącej wartości łącznej przyszłego strumienia przepływów pieniężnych. Aby obliczyć NPV, należy oszacować przyszłe przepływy pieniężne dla każdego okresu i określić poprawną stopę dyskontową.

Funkcja `XNPV()` pobiera stopę dyskontową i kilka wartości uporządkowanych według okresu. Wpływy (przychody) są dodatnie, a wypływy (przyszłe płatności) powinny być wartościami ujemnymi. Występują na końcu każdego okresu.

Argumenty

Argument	Opis
discount_rate	**discount_rate** jest wartością procentową zastosowanego rabatu. Wartość 0,1 oznaczałaby stopę dyskontową w wysokości 10%.
value	To pole przechowuje wartości przepływów pieniężnych. Przyjmuje się, że pierwsza wartość reprezentuje przepływ pieniężny na początku, a odnośna data służy jako punkt odniesienia do obliczenia wartości bieżącej dla wszystkich przyszłych przepływów pieniężnych. *__XNPV()__ nie dyskontuje początkowego przepływu pieniężnego. Kolejne płatności są dyskontowane na podstawie roku zawierającego 365 dni. W funkcji __NPV()__ jest inaczej, ponieważ w jej przypadku każda płatność jest dyskontowana.*
date	To pole zawiera datę dokonania przepływu pieniężnego (**value**, drugi parametr). Pierwsza wartość określa datę początkową do obliczania przesunięć dla przyszłych przepływów pieniężnych.

Ograniczenia:

Jeśli w jednej lub obu częściach pary danych pojawi się wartość tekstowa, wartości NULL, lub zabraknie którejś z tych wartości, dana para zostanie pominięta.

Kiedy używać

- Funkcja `XNPV()` jest używana w modelowaniu finansowym do obliczania bieżącej wartości netto (NPV) potencjalnej inwestycji.
- Funkcja XNPV jest dokładniejsza i dlatego jest preferowana w porównaniu z NPV we wszystkich typach modeli finansowych.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Pojedyncza płatność (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych jednego projektu i jego przepływy pieniężne dla jednego roku, w tabeli o nazwie `CashFlow`. Początkowa data do obliczeń jest ustawiona na 1 lipca 2022 r., a przepływ pieniężny netto wynosi 0. Po roku pojawia się przepływ pieniężny w wysokości 1000 $.
- Polecenie LOAD z predykatem Resident z tabeli `CashFlow`, które jest używane do obliczania pola `XNPV` dla projektu w tabeli o nazwie `XNPV`.
- Wpisana w kod stopa dyskontowa w wysokości 10% (0,1), która jest używana w obliczeniach XNPV.
- Polecenie `Group By` użyte do zgrupowania wszystkich płatności dotyczących projektu.

Skrypt ładowania

```
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;
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- PrjId
- XNPV

Tabela wynikowa

PrjId	XNPV
1	$909.09

Zgodnie ze wzorem, wartość XNPV dla pierwszego rekordu wynosi 0, a wartość XNPV dla drugiego rekordu wynosi 909,09 $. Zatem całkowita wartość XNPV wynosi 909,09 $.

Przykład 2 – Kilka płatności (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych jednego projektu i jego przepływy pieniężne dla jednego roku, w tabeli o nazwie CashFlow.
- Polecenie LOAD z predykatem Resident z tabeli CashFlow, które jest używane do obliczania pola XNPV dla projektu w tabeli o nazwie XNPV.

- Wpisana w kod stopa dyskontowa w wysokości 10% (0,1), która jest używana w obliczeniach XNPV.
- Polecenie `Group By` użyte do zgrupowania wszystkich płatności dotyczących projektu.

Skrypt ładowania

```
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;
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `PrjId`
- `XNPV`

Tabela wynikowa

PrjId	XNPV
1	$1322.21

W tym przykładzie otrzymano płatność w wysokości 1000 $ na koniec pierwszego roku i płatność w wysokości 500 $ na koniec drugiego roku. Przy stopie dyskontowej na poziomie 10% na okres efektywna wartość XNPV wynosi 1322,21 $.

Należy zauważyć, że tylko pierwszy wiersz danych powinien odnosić się do daty bazowej do obliczeń. W przypadku pozostałych wierszy kolejność jest nieistotna, ponieważ do obliczania okresu, który upłynął, jest używany parametr date.

Przykład 3 – Kilka płatności w nieregularnych przepływach pieniężnych (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Stopy dyskontowe dla dwóch projektów w tabeli o nazwie `Project`.
- Przepływy pieniężne dla kilku okresów dla każdego projektu według identyfikatora projektu i dat. Pole `Dates` jest używane do obliczania czasu, przez jaki stopa dyskontowa jest stosowana do przepływu pieniężnego. Nie licząc pierwszego rekordu (początkowy przepływ pieniężny i data), kolejność rekordów jest nieistotna, i jej zmiana nie powinna mieć wpływu na wynik obliczeń.
- Przy użyciu kombinacji `NoConcatenate`, poleceń LOAD z predykatem Resident oraz funkcji `Left Join` zostaje utworzona tymczasowa tabela `tmpNPV`, która łączy rekordy z tabel `Project` i `CashFlow` w jedną płaską tabelę. Będzie ona zawierać stopy dyskontowe dla każdego przepływu pieniężnego.
- Polecenie LOAD z predykatem Resident z tabeli `tmpNPV`, które jest używane do obliczania pola `XNPV` dla każdego projektu w tabeli o nazwie `XNPV`.
- Pojedyncza stopa dyskontowa powiązana z każdym projektem jest pobierana za pomocą funkcji `only()` i jest używana w obliczeniach `XNPV` dla każdego projektu.
- Polecenie `Group By`, grupowanie według identyfikatora projektu, jest używane do grupowania wszystkich płatności i odnośnych dat dla każdego projektu.
- Aby uniknąć załadowania do modelu danych syntetycznych lub redundantnych, na końcu skryptu tabela `tmpXNPV` zostaje usunięta.

Skrypt ładowania

```
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;
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- PrjId
- XNPV

Tabela wynikowa

PrjId	XNPV
1	$1735.54
2	$278.36

Projekt o identyfikatorze 1 ma początkowy przepływ pieniężny w wysokości 0 $ na dzień 1 lipca 2021 r. Na koniec dwóch kolejnych lat mają zostać odebrane dwie płatności po 1000 $ ze stopą dyskontową 10% na okres. Zatem efektywna wartość XNPV wynosi 1735,54 $.

Projekt o identyfikatorze 2 ma początkowy wypływ pieniężny w wysokości 1000 $ (stąd znak minus) 1 lipca 2020 r. Po dwóch latach oczekiwana jest płatność w wysokości 500 $. Po trzech latach oczekiwana jest kolejna płatność w wysokości 500 $. Na koniec, 1 lipca 2024 r., oczekiwana jest płatność w wysokości 1000 $. Przy stopie dyskontowej na poziomie 15% efektywna wartość XNPV wynosi 278,36 $.

XNPV – funkcja wykresu

Funkcja **XNPV()** zwraca zagregowaną wartość bieżącą netto dotyczącą harmonogramu przepływów pieniężnych (niekoniecznie okresowych) reprezentowanych przez połączone w pary liczby zwracane przez wyrażenia podane jako **pmt** i **date**, iterowane po wymiarach wykresu. Wszystkie płatności są obejmowane upustem na podstawie roku zawierającego 365 dni.

Składnia:

```
XNPV([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

Typ zwracanych danych: numeric Domyślnie wynik zostanie sformatowany jako waluta.

Argumenty:

Argumenty

Argument	Opisu
discount_rate	**discount_rate** is the rate of discount over the length of the period.**discount_rate** jest wartością procentową zastosowanego rabatu.
pmt	Płatności. Wyrażenie lub pole zawierające płatności w ramach przepływów pieniężnych odpowiadające harmonogramowi płatności podanemu argumentem **date**.
date	Wyrażenie lub pole zawierające harmonogram dat odpowiadających płatnościom w ramach przepływów pieniężnych podanym argumentem **pmt**.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Argumenty **discount_rate**,**pmt** i **date** nie mogą zawierać funkcji agregacji, chyba że takie wewnętrzne funkcje agregacji zawierają kwalifikatory **TOTAL** lub **ALL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Przykłady i wyniki:

Przykłady i wyniki

Przykład	Wynik
`XNPV(Discount, Payments, Date)`	-$3164.35

Dane zastosowane w przykładach:

```
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 '|');
```

Zob. także:

p *NPV – funkcja wykresu (page 369)*
p *Aggr – funkcja wykresu (page 532)*

Statystyczne funkcje agregacji

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

Statystyczne funkcje agregacji dla ciągów w skrypcie ładowania danych

W skryptach mogą być stosowane następujące statystyczne funkcje agregacji:

Avg
Funkcja **Avg()** wyszukuje średnią wartość zagregowanych danych w wyrażeniu, zgodnie z definicją z klauzuli **group by**.

```
Avg  ([distinct] expression)
```

Correl
Funkcja **Correl()** zwraca zagregowany współczynnik korelacji dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
Correl (x-expression, y-expression)
```

Fractile
Funkcja **Fractile()** wyszukuje wartość odpowiadającą fraktylowi (kwantylowi) z przedziału zamkniętego zagregowanych danych w wyrażeniu iterowanym po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
Fractile (expression, fractile)
```

FractileExc
Funkcja **FractileExc()** wyszukuje wartość odpowiadającą fraktylowi (kwantylowi) z przedziału otwartego zagregowanych danych w wyrażeniu iterowanym po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
FractileExc (expression, fractile)
```

Kurtosis
Funkcja **Kurtosis()** zwraca kurtozę danych w wyrażeniu iterowanym po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
Kurtosis  ([distinct ] expression )
```

LINEST_B

Funkcja **LINEST_B()** zwraca zagregowaną wartość b (punkt przecięcia z osią Y) regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
LINEST_B (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_df

Funkcja **LINEST_DF()** zwraca zagregowane stopnie swobody regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
LINEST_DF  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_f

Ta funkcja skryptu zwraca zagregowaną statystykę F ($r^2/(1-r^2)$) regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
LINEST_F (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_m

Funkcja **LINEST_M()** zwraca zagregowaną wartość m (nachylenie) regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
LINEST_M  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_r2

Funkcja **LINEST_R2()** zwraca zagregowaną wartość r^2 (współczynnik determinacji) regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
LINEST_R2 (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_seb

Funkcja **LINEST_SEB()** zwraca zagregowany błąd standardowy wartości b regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
LINEST_SEB  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sem

Funkcja **LINEST_SEM()** zwraca zagregowany błąd standardowy wartości m regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
LINEST_SEM (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sey

Funkcja **LINEST_SEY()** zwraca zagregowany błąd standardowy oszacowania y regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
LINEST_SEY (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_ssreg

Funkcja **LINEST_SSREG()** zwraca zagregowaną sumę kwadratów regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
LINEST_SSREG (y-expression, x-expression [, y0 [, x0 ]])
```

Linest_ssresid

Funkcja **LINEST_SSRESID()** zwraca zagregowaną sumę kwadratów reszt regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
LINEST_SSRESID (y-expression, x-expression [, y0 [, x0 ]])
```

Median

Funkcja **Median()** zwraca zagregowaną medianę wartości w wyrażeniu iterowanym po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
Median (expression)
```

Skew

Funkcja **Skew()** zwraca skośność wyrażenia iterowanego po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
Skew ([ distinct] expression)
```

Stdev

Funkcja **Stdev()** zwraca odchylenie standardowe wartości podane w wyrażeniu iterowanym po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
Stdev ([distinct] expression)
```

Sterr

Funkcja **Sterr()** zwraca zagregowany błąd standardowy (stdev/sqrt(n)) dla serii wartości reprezentowanych przez wyrażenie iterowane po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
Sterr  ([distinct] expression)
```

STEYX

Funkcja **STEYX()** zwraca zagregowany błąd standardowy przewidywanej wartości y dla każdej wartości x w regresji dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

```
STEYX  (y-expression, x-expression)
```

Statystyczne funkcje agregacji w wyrażeniach wykresu

W wykresach mogą być stosowane następujące statystyczne funkcje agregacji:

Avg

Funkcja **Avg()** zwraca zagregowaną średnią wartość wyrażenia lub pola iterowaną po wymiarach wykresu.

```
Avg — funkcja wykresu({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]}
expr)
```

Correl

Funkcja **Correl()** zwraca zagregowany współczynnik korelacji dwóch zestawów danych. Funkcja korelacji stanowi miarę związku między zestawami danych, a agregacja jest wykonywana dla par wartości (x,y) iterowanych po wykresach wymiaru.

```
Correl — funkcja wykresu({[SetExpression] [TOTAL [<fld {, fld}>]]} value1,
value2 )
```

Fractile

Funkcja **Fractile()** zwraca wartość odpowiadającą fraktylowi (kwantylowi) z przedziału zamkniętego zagregowanych danych w zakresie podanym wyrażeniem iterowanym po wymiarach wykresu.

```
Fractile — funkcja wykresu({[SetExpression] [TOTAL [<fld {, fld}>]]} expr,
fraction)
```

FractileExc

Funkcja **FractileExc()** zwraca wartość odpowiadającą fraktylowi (kwantylowi) z przedziału otwartego zagregowanych danych w zakresie podanym wyrażeniem iterowanym po wymiarach wykresu.

```
FractileExc — funkcja wykresu({[SetExpression] [TOTAL [<fld {, fld}>]]} expr,
fraction)
```

Kurtosis

Funkcja **Kurtosis()** zwraca kurtozę zakresu danych zagregowanych w wyrażeniu lub polu iterowanym po wymiarach wykresu.

```
Kurtosis — funkcja wykresu({[SetExpression] [DISTINCT] [TOTAL [<fld{,
fld}>]]} expr)
```

LINEST_b

Funkcja **LINEST_B()** zwraca zagregowaną wartość b (przecięcie osi Y) regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowaną po wymiarach wykresu.

```
LINEST_R2 – funkcja wykresu({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_value,
x_value[, y0_const[, x0_const]])
```

LINEST_df

Funkcja **LINEST_DF()** zwraca zagregowaną wartość stopni swobody regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowaną po wymiarach wykresu.

```
LINEST_DF – funkcja wykresu({[SetExpression] [TOTAL [<fld{, fld}>]]} y_value,
x_value [, y0_const [, x0_const]])
```

LINEST_f

Funkcja **LINEST_F()** zwraca zagregowaną wartość statystyki F (r2/(1-r2)) regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowaną po wymiarach wykresu.

```
LINEST_F – funkcja wykresu({[SetExpression] [TOTAL[<fld{, fld}>]]} y_value,
x_value [, y0_const [, x0_const]])
```

LINEST_m

Funkcja **LINEST_M()** zwraca zagregowaną wartość m (nachylenie) regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowaną po wymiarach wykresu.

```
LINEST_M – funkcja wykresu({[SetExpression] [TOTAL[<fld{, fld}>]]} y_value,
x_value [, y0_const [, x0_const]])
```

LINEST_r2

Funkcja **LINEST_R2()** zwraca zagregowaną wartość r2 (współczynnik determinacji) regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowaną po wymiarach wykresu.

```
LINEST_R2 – funkcja wykresu({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_value,
x_value[, y0_const[, x0_const]])
```

LINEST_seb

Funkcja **LINEST_SEB()** zwraca zagregowany błąd standardowy wartości b regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowany po wymiarach wykresu.

```
LINEST_SEB – funkcja wykresu({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_sem

Funkcja **LINEST_SEM()** zwraca zagregowany błąd standardowy wartości m regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowany po wymiarach wykresu.

```
LINEST_SEM – funkcja wykresu([{set_expression}][ distinct ] [total [<fld
{,fld}>] ] y-expression, x-expression [, y0 [, x0 ]] )
```

LINEST_sey

Funkcja **LINEST_SEY()** zwraca zagregowany błąd standardowy szacowanej wartości y regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowany po wymiarach wykresu.

```
LINEST_SEY – funkcja wykresu({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_ssreg

Funkcja **LINEST_SSREG()** zwraca zagregowaną sumę kwadratów regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowaną po wymiarach wykresu.

```
LINEST_SSREG – funkcja wykresu({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_ssresid

Funkcja **LINEST_SSRESID()** zwraca zagregowaną sumę kwadratów reszt regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez pary liczb w wyrażeniach określanych przez **x_value** i **y_value** iterowane po wymiarach wykresu.

```
LINEST_SSRESID – funkcja wykresu Funkcja LINEST_SSRESID() zwraca zagregowaną
sumę kwadratów reszt regresji liniowej zdefiniowanej równaniem y=mx+b dla
serii współrzędnych reprezentowanych przez pary liczb w wyrażeniach
określanych przez x_value i y_value iterowane po wymiarach wykresu.  LINEST_
SSRESID([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const[, x0_const]]) numeric ArgumentyArgumentOpisy_valueWyrażenie lub
pole zawierające zakres wartości y do zmierzenia.x_valueWyrażenie lub pole
zawierające zakres wartości x do zmierzenia.y0, x0Opcjonalnie można podać
wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w
danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby
linia regresji przechodziła przez pojedynczą ustaloną współrzędną. Funkcja na
potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że
podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para
danych. SetExpressionFunkcja agregacji będzie domyślnie dokonywać agregacji
na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny
zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCTJeśli przed argumentami funkcji występuje słowo DISTINCT, wówczas
duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTALJeśli słowo TOTAL występuje przed argumentami funkcji, wówczas
```

obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia TOTAL [<fld {.fld}>], gdzie po kwalifikatorze TOTAL podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator TOTAL. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w Aggrpołączeniu z określonym wymiarem. Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych. An example of how to use linest functionsavg([SetExpression] [**TOTAL** [<fld{ ,fld}>]] }y_value, x_value[, y0_const[, x0_const]]**)**

Median

Funkcja **Median()** zwraca wartość mediany z zakresu wartości zagregowanych w wyrażeniu iterowanym po wymiarach wykresu.

Median — funkcja wykresu([SetExpression] [**TOTAL** [<fld{, fld}>]]} expr**)**

MutualInfo

MutualInfo oblicza informacje wzajemne pomiędzy dwoma polami lub wartościami zagregowanymi w **Aggr()**.

MutualInfo — funkcja wykresu (page 428)[SetExpression] [**DISTINCT**] [**TOTAL** target, driver [, datatype [, breakdownbyvalue [, samplesize]]]**)**

Skew

Funkcja **Skew()** zwraca zagregowaną skośność wartości wyrażenia lub pola iterowanych po wymiarach wykresu.

Skew — funkcja wykresu[SetExpression] [**DISTINCT**] [**TOTAL** [<fld{ ,fld}>]]} expr**)**

Stdev

Funkcja **Stdev()** zwraca odchylenie standardowe zakresu danych zagregowanych w wyrażeniu lub polu iterowanym po wymiarach wykresu.

Stdev — funkcja wykresu([SetExpression] [**DISTINCT**] [**TOTAL** [<fld{, fld}>]]} expr**)**

Sterr

Funkcja **Sterr()** zwraca wartość błędu standardowego wartości średniej (stdev/sqrt(n)) dla szeregu wartości zagregowanych w wyrażeniu, iterowaną po wymiarach wykresu.

```
Sterr — funkcja wykresu({[SetExpression] [DISTINCT] [TOTAL[<fld{, fld}>]]}
expr)
```

STEYX
Funkcja **STEYX()** zwraca zagregowany błąd standardowy przy prognozowaniu wartości y dla każdej wartości x w regresji liniowej określonej szeregiem współrzędnych reprezentowanych przez pary liczb w wyrażeniu podanym argumentami **y_value** i **x_value**.

```
STEYX — funkcja wykresu{[SetExpression] [TOTAL [<fld{, fld}>]]} y_value, x_
value)
```

Avg

Funkcja **Avg()** wyszukuje średnią wartość zagregowanych danych w wyrażeniu, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
Avg([DISTINCT] expr)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
DISTINCT	Jeśli słowo **distinct** występuje przed wyrażeniem, wówczas wszystkie duplikaty zostaną pominięte.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

Dane wynikowe

Przykład	Wynik
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 Można to sprawdzić w arkuszu, tworząc tabelę zawierającą miarę: Sum(Sales)/12
Zakładając, że tabela **Temp** została załadowana jak w poprzednim przykładzie: 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 Zliczane są tylko wartości odrębne. Sumę należy podzielić przez liczbę wartości, które nie są zduplikowane.

Avg – funkcja wykresu

Funkcja **Avg()** zwraca zagregowaną średnią wartość wyrażenia lub pola iterowaną po wymiarach wykresu.

Składnia:

```
Avg([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
expr	Wyrażenie lub pole zawierające mierzone dane.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.

Argument	Opisu
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Przykłady i wyniki:

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

Przykłady funkcji

Przykład	Wynik
`Avg(Sales)`	W przypadku tabeli zawierającej wymiar `Customer` i miarę `Avg([Sales])`, jeśli pokazane są wartości **Totals**, zwracany jest wynik 2566.
`Avg([TOTAL (Sales))`	53,458333 dla wszystkich wartości wymiaru `Customer`, ponieważ kwalifikator TOTAL sprawia, że wymiary są ignorowane.
`Avg (DISTINCT (Sales))`	51,862069 dla sumy, ponieważ zastosowanie kwalifikatora Distinct sprawia, że oceniane są tylko niepowtarzalne wartości `Sales` dla każdej wartości wymiaru `Customer`.

Dane zastosowane w przykładach:

```
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 '|');
```

Zob. także:

p *Aggr – funkcja wykresu (page 532)*

Correl

Funkcja **Correl()** zwraca zagregowany współczynnik korelacji dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
Correl(value1, value2)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
value1, value2	Wyrażenia lub pola zawierające dwa zestawy prób, dla których będzie obliczany współczynnik korelacji.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

Dane wynikowe

Przykład	Wynik
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;	W tabeli z wymiarem `Correl_Salary` wynikiem obliczenia Correl() w skrypcie ładowania danych jest zostanie przedstawiony wynik: 0,9270611.

Correl – funkcja wykresu

Funkcja **Correl()** zwraca zagregowany współczynnik korelacji dwóch zestawów danych. Funkcja korelacji stanowi miarę związku między zestawami danych, a agregacja jest wykonywana dla par wartości (x,y) iterowanych po wykresach wymiaru.

Składnia:

```
Correl([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] value1, value2 )
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
value1, value2	Wyrażenia lub pola zawierające dwa zestawy prób, dla których będzie obliczany współczynnik korelacji.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.

Argument	Opis
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Przykłady i wyniki:

Przykłady funkcji

Przykład	Wynik
`Correl (Age, Salary)`	W przypadku tabeli zawierającej wymiar `Employee name` i miarę `Correl(Age, Salary)`, wynik wynosi 0,9270611. Wynik ten jest wyświetlany tylko dla komórki sumy.
`Correl (TOTAL Age, Salary))`	0.927. W celu zapewnienia większej przejrzystości ten i następne wyniki są wyświetlane z trzema miejscami po przecinku. Jeśli utworzysz panel filtrowania z wymiarem Gender, a następnie dokonasz w nim selekcji, wówczas pojawi się wynik 0,951 w przypadku selekcji Female oraz wynik 0,939 w przypadku selekcji Male. Przyczyną jest to, że selekcja wyklucza wszystkie wyniki, które nie należą do drugiej wartości wymiaru Gender.
`Correl({1} TOTAL Age, Salary))`	0.927. Niezależnie od selekcji. Przyczyną jest to, że wyrażenie zestawu {1} pomija wszystkie selekcje i wymiary.
`Correl (TOTAL <Gender> Age, Salary))`	0,927 w komórce sumy, 0,939 dla wszystkich wartości Male oraz 0,951 dla wszystkich wartości Female. To odpowiada wynikom w przypadku selekcji w panelu filtrowania na podstawie wymiaru Gender.

Dane zastosowane w przykładach:

```
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 '|');
```

Zob. także:

p *Aggr – funkcja wykresu (page 532)*
p *Avg – funkcja wykresu (page 389)*
p *RangeCorrel (page 1313)*

Fractile

Funkcja **Fractile()** wyszukuje wartość odpowiadającą fraktylowi (kwantylowi) z przedziału zamkniętego zagregowanych danych w wyrażeniu iterowanym po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Do obliczenia fraktylu z przedziału otwartego można użyć funkcji FractileExc (page 398).

Składnia:

```
Fractile(expr, fraction)
```

Typ zwracanych danych: numeric

Funkcja zwraca wartość odpowiadającą klasyfikacji zdefiniowanej przez `rank = fraction * (N-1) + 1`, gdzie `N` jest liczbą wartości w wyrażeniu `expr`. Jeśli `rank` jest liczbą niecałkowitą, dokonuje się interpolacji między dwiema najbliższymi wartościami.

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające dane do wykorzystania przy obliczaniu fraktylu.
fraction	Liczba z przedziału od 0 do 1 odpowiadająca obliczanemu fraktylowi (kwantylowi wyrażonemu ułamkiem).

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

Dane wynikowe

Przykład	Wynik

`Table1:`
`crosstable 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;`
	W tabeli z wymiarami `Type` i `MyFractile` wynikami obliczeń wymiaru Fractile() w skrypcie ładowania danych są:
	
	`Type MyFractile`
	`Comparison 27.5`
	`Observation 36`

Fractile – funkcja wykresu

Funkcja **Fractile()** zwraca wartość odpowiadającą fraktylowi (kwantylowi) z przedziału zamkniętego zagregowanych danych w zakresie podanym wyrażeniem iterowanym po wymiarach wykresu.

Do obliczenia fraktylu z przedziału otwartego można użyć funkcji FractileExc – funkcja wykresu (page 399).

Składnia:

```
Fractile([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr, fraction)
```

Typ zwracanych danych: numeric

Funkcja zwraca wartość odpowiadającą klasyfikacji zdefiniowanej przez `rank = fraction * (N-1) + 1`, gdzie `N` jest liczbą wartości w wyrażeniu `expr`. Jeśli `rank` jest liczbą niecałkowitą, dokonuje się interpolacji między dwiema najbliższymi wartościami.

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające dane do wykorzystania przy obliczaniu fraktylu.
fraction	Liczba z przedziału od 0 do 1 odpowiadająca obliczanemu fraktylowi (kwantylowi wyrażonemu ułamkiem).
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Przykłady i wyniki:

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

Przykłady funkcji

Przykład	Wynik
`Fractile (Sales, 0.75)`	W przypadku tabeli zawierającej wymiar `Customer` i miarę `Fractile([Sales])`, jeśli pokazane są wartości **Totals**, zwracany jest wynik 71,75. Wartość ta wskazuje punkt w rozkładzie wartości `Sales`, poniżej którego przypada 75% wartości.
`Fractile (TOTAL Sales, 0.75))`	71,75 dla wszystkich wartości wymiaru `Customer`, ponieważ kwalifikator TOTAL sprawia, że wymiary są ignorowane.
`Fractile (DISTINCT Sales, 0.75)`	70 dla sumy, ponieważ zastosowanie kwalifikatora DISTINCT sprawia, że oceniane są tylko niepowtarzalne wartości `Sales` dla każdej wartości wymiaru `Customer`.

Dane zastosowane w przykładach:

```
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 '|');
```

Zob. także:

p *Aggr – funkcja wykresu (page 532)*

FractileExc

Funkcja **FractileExc()** wyszukuje wartość odpowiadającą fraktylowi (kwantylowi) z przedziału otwartego zagregowanych danych w wyrażeniu iterowanym po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Do obliczenia fraktylu z przedziału zamkniętego można użyć funkcji Fractile (page 394).

Składnia:

```
FractileExc(expr, fraction)
```

Typ zwracanych danych: numeric

Funkcja zwraca wartość odpowiadającą klasyfikacji zdefiniowanej przez `rank = fraction * (N +1)` , gdzie `N` jest liczbą wartości w wyrażeniu `expr`. Jeśli `rank` jest liczbą niecałkowitą, dokonuje się interpolacji między dwiema najbliższymi wartościami.

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające dane do wykorzystania przy obliczaniu fraktylu.
fraction	Liczba z przedziału od 0 do 1 odpowiadająca obliczanemu fraktylowi (kwantylowi wyrażonemu ułamkiem).

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

Dane wynikowe

Przykład	Wynik
Table1: crosstable 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;	W tabeli z wymiarami `Type` i `MyFractile` wynikami obliczeń wymiaru FractileExc() w skrypcie ładowania danych są: Type MyFractile Comparison 28.5 Observation 38

FractileExc – funkcja wykresu

Funkcja **FractileExc()** zwraca wartość odpowiadającą fraktylowi (kwantylowi) z przedziału otwartego zagregowanych danych w zakresie podanym wyrażeniem iterowanym po wymiarach wykresu.

Do obliczenia fraktylu z przedziału zamkniętego można użyć funkcji Fractile – funkcja wykresu (page 395).

Składnia:

```
FractileExc([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr, fraction)
```

Typ zwracanych danych: numeric

Funkcja zwraca wartość odpowiadającą klasyfikacji zdefiniowanej przez `rank = fraction * (N +1)` , gdzie `N` jest liczbą wartości w wyrażeniu `expr`. Jeśli `rank` jest liczbą niecałkowitą, dokonuje się interpolacji między dwiema najbliższymi wartościami.

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające dane do wykorzystania przy obliczaniu fraktylu.
fraction	Liczba z przedziału od 0 do 1 odpowiadająca obliczanemu fraktylowi (kwantylowi wyrażonemu ułamkiem).
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Przykłady i wyniki:

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

Przykłady funkcji

Przykład	Wynik
`FractileExc (Sales, 0.75)`	W przypadku tabeli zawierającej wymiar `Customer` i miarę `FractileExc([Sales])`, jeśli pokazane są wartości **Totals**, zwracany jest wynik 75.25. Wartość ta wskazuje punkt w rozkładzie wartości `Sales`, poniżej którego przypada 75% wartości.
`FractileExc (TOTAL Sales, 0.75))`	75.25 dla wszystkich wartości wymiaru `Customer`, ponieważ kwalifikator TOTAL sprawia, że wymiary są ignorowane.
`FractileExc (DISTINCT Sales, 0.75)`	73,50 dla sumy, ponieważ zastosowanie kwalifikatora DISTINCT sprawia, że oceniane są tylko niepowtarzalne wartości `Sales` dla każdej wartości wymiaru `Customer`.

Dane zastosowane w przykładach:

```
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 '|');
```

Zob. także:

p *Aggr – funkcja wykresu (page 532)*

Kurtosis

Funkcja **Kurtosis()** zwraca kurtozę danych w wyrażeniu iterowanym po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
Kurtosis([distinct ] expr )
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
distinct	Jeśli słowo **distinct** występuje przed wyrażeniem, wówczas wszystkie duplikaty zostaną pominięte.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

Dane wynikowe

Przykład	Wynik
Table1: crosstable 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;	W tabeli z wymiarami Type, MyKurtosis1 i MyKurtosis2 wynikami obliczeń wymiaru Kurtosis() w skrypcie ładowania danych są: Type MyKurtosis1 MyKurtosis2 Comparison -1.1612957 -1.4982366 Observation -1.1148768 -0.93540144

Kurtosis – funkcja wykresu

Funkcja **Kurtosis()** zwraca kurtozę zakresu danych zagregowanych w wyrażeniu lub polu iterowanym po wymiarach wykresu.

Składnia:

```
Kurtosis([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
expr	Wyrażenie lub pole zawierające mierzone dane.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Przykłady i wyniki:

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

Przykłady funkcji

Przykład	Wynik
`Kurtosis (Value)`	W przypadku tabeli zawierającej wymiar `Type` i miarę `Kurtosis(Value)`, jeśli pokazane są wartości **Totals** dla tej tabeli, a format liczb przewiduje trzy cyfry znaczące, zwracany jest wynik 1,252. W przypadku wymiaru `Comparison` jest to 1,161, a w przypadku wymiaru `Observation` – 1,115.
`Kurtosis (TOTAL Value))`	1,252 dla wszystkich wartości wymiaru `Type`, ponieważ kwalifikator TOTAL sprawia, że wymiary są ignorowane.

Dane zastosowane w przykładach:

```
Table1:
crosstable 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 '|');
```

Zob. także:

p *Avg – funkcja wykresu (page 389)*

LINEST_B

Funkcja **LINEST_B()** zwraca zagregowaną wartość b (punkt przecięcia z osią Y) regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
LINEST_B (y_value, x_value[, y0 [, x0 ]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y(0), x(0)	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*

LINEST_B — funkcja wykresu

Funkcja **LINEST_B()** zwraca zagregowaną wartość b (przecięcie osi Y) regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowaną po wymiarach wykresu.

Składnia:

```
LINEST_B([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [ , x0_const]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.

Argument	Opis
y0_const, x0_const	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. *Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.*
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*
p *Avg – funkcja wykresu (page 389)*

LINEST_DF

Funkcja **LINEST_DF()** zwraca zagregowane stopnie swobody regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
LINEST_DF (y_value, x_value[, y0 [, x0 ]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y(0), x(0)	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*

LINEST_DF – funkcja wykresu

Funkcja **LINEST_DF()** zwraca zagregowaną wartość stopni swobody regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowaną po wymiarach wykresu.

Składnia:

```
LINEST_DF([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value [, y0_const [, x0_const]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.

Argument	Opis
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y0, x0	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. *Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.*
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*
p *Avg – funkcja wykresu (page 389)*

LINEST_F

Ta funkcja skryptu zwraca zagregowaną statystykę F ($r^2/(1-r^2)$) regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z

definicją z klauzuli **group by**.

Składnia:

```
LINEST_F (y_value, x_value[, y0 [, x0 ]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y(0), x(0)	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*

LINEST_F — funkcja wykresu

Funkcja **LINEST_F()** zwraca zagregowaną wartość statystyki F (r2/(1-r2)) regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowaną po wymiarach wykresu.

Składnia:

```
LINEST_F([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y0, x0	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. *Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.*
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*

p *Avg – funkcja wykresu (page 389)*

LINEST_M

Funkcja **LINEST_M()** zwraca zagregowaną wartość m (nachylenie) regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
LINEST_M (y_value, x_value[, y0 [, x0 ]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y(0), x(0)	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*

LINEST_M – funkcja wykresu

Funkcja **LINEST_M()** zwraca zagregowaną wartość m (nachylenie) regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowaną po wymiarach wykresu.

Składnia:

```
LINEST_M([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y0, x0	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. *Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.*
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*

p *Avg – funkcja wykresu (page 389)*

LINEST_R2

Funkcja **LINEST_R2()** zwraca zagregowaną wartość r^2 (współczynnik determinacji) regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
LINEST_R2 (y_value, x_value[, y0 [, x0 ]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y(0), x(0)	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*

LINEST_R2 – funkcja wykresu

Funkcja **LINEST_R2()** zwraca zagregowaną wartość r2 (współczynnik determinacji) regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowaną po wymiarach wykresu.

Składnia:

```
LINEST_R2([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y0, x0	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. *Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.*
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*

p *Avg – funkcja wykresu (page 389)*

LINEST_SEB

Funkcja **LINEST_SEB()** zwraca zagregowany błąd standardowy wartości b regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
LINEST_SEB (y_value, x_value[, y0 [, x0 ]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y(0), x(0)	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*

LINEST_SEB – funkcja wykresu

Funkcja **LINEST_SEB()** zwraca zagregowany błąd standardowy wartości b regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowany po wymiarach wykresu.

Składnia:

```
LINEST_SEB([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y0, x0	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. *Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.*
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*

p *Avg – funkcja wykresu (page 389)*

LINEST_SEM

Funkcja **LINEST_SEM()** zwraca zagregowany błąd standardowy wartości m regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
LINEST_SEM (y_value, x_value[, y0 [, x0 ]])
```

Typ zwracanych danych: numeric

Argumenty:

Argument	Opisu
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y(0), x(0)	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*

LINEST_SEM – funkcja wykresu

Funkcja **LINEST_SEM()** zwraca zagregowany błąd standardowy wartości m regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowany po wymiarach wykresu.

Składnia:

```
LINEST_SEM([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y0, x0	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. *Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.*
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*

p *Avg – funkcja wykresu (page 389)*

LINEST_SEY

Funkcja **LINEST_SEY()** zwraca zagregowany błąd standardowy oszacowania y regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
LINEST_SEY (y_value, x_value[, y0 [, x0 ]])
```

Typ zwracanych danych: numeric

Argumenty:

Argument	Opisu
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y(0), x(0)	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*

LINEST_SEY – funkcja wykresu

Funkcja **LINEST_SEY()** zwraca zagregowany błąd standardowy szacowanej wartości y regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowany po wymiarach wykresu.

Składnia:

```
LINEST_SEY([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y0, x0	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. *Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.*
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*

p *Avg – funkcja wykresu (page 389)*

LINEST_SSREG

Funkcja **LINEST_SSREG()** zwraca zagregowaną sumę kwadratów regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
LINEST_SSREG (y_value, x_value[, y0 [, x0 ]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y(0), x(0)	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*

LINEST_SSREG – funkcja wykresu

Funkcja **LINEST_SSREG()** zwraca zagregowaną sumę kwadratów regresji liniowej zdefiniowanej równaniem y=mx+b dla szeregu współrzędnych reprezentowanych przez pary liczb w wyrażeniach podanych argumentami **x_value** i **y_value**, iterowaną po wymiarach wykresu.

Składnia:

```
LINEST_SSREG([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y0, x0	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. *Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.*
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*

p *Avg – funkcja wykresu (page 389)*

LINEST_SSRESID

Funkcja **LINEST_SSRESID()** zwraca zagregowaną sumę kwadratów reszt regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
LINEST_SSRESID (y_value, x_value[, y0 [, x0 ]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y(0), x(0)	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*

LINEST_SSRESID – funkcja wykresu

Funkcja **LINEST_SSRESID()** zwraca zagregowaną sumę kwadratów reszt regresji liniowej zdefiniowanej równaniem y=mx+b dla serii współrzędnych reprezentowanych przez pary liczb w wyrażeniach określanych przez **x_value** i **y_value** iterowane po wymiarach wykresu.

Składnia:

```
LINEST_SSRESID([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value,
x_value[, y0_const[, x0_const]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.
y0, x0	Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną. *Funkcja na potrzeby obliczenia wymaga co najmniej dwóch poprawnych par danych, chyba że podano y0 i x0. Jeśli określono y0 i x0, wówczas wystarczy jedna para danych.*
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Opcjonalnie można podać wartość y0 w celu wymuszenia, aby linia regresji przechodziła przez oś Y w danym punkcie. Po określeniu wartości y0 i x0 możliwe jest wymuszenie, aby linia regresji przechodziła przez pojedynczą ustaloną współrzędną.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Zob. także:

p *Przykłady stosowania funkcji linest (page 446)*
p *Avg – funkcja wykresu (page 389)*

Median

Funkcja **Median()** zwraca zagregowaną medianę wartości w wyrażeniu iterowanym po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
Median (expr)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.

Przykład: Wyrażenia skryptu używające funkcji Median

Przykład – wyrażenie skryptu

Skrypt ładowania

Załaduj następujące dane wbudowane i wyrażenie skryptu w edytorze ładowania danych na potrzeby poniższego przykładu.

```
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 Grou
```

Tworzenie wizualizacji

Utwórz wizualizację tabeli w arkuszu Qlik Sense z wymiarami **Letter** i **MyMedian**.

Wynik

Letter	MyMedian
A	3.5
B	8

Objaśnienie

Mediana jest uważana za liczbę „środkową”, gdy liczby są posortowane w kolejności od najmniejszej do największej. Jeśli zestaw danych ma parzystą liczbę wartości, funkcja zwraca średnią z dwóch środkowych wartości. W tym przykładzie mediana jest obliczana dla każdego zestawu wartości **A** oraz **B** i wynosi odpowiednio 3,5 oraz 8.

Median – funkcja wykresu

Funkcja **Median()** zwraca wartość mediany z zakresu wartości zagregowanych w wyrażeniu iterowanym po wymiarach wykresu.

Składnia:

```
Median([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
expr	Wyrażenie lub pole zawierające mierzone dane.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Przykład: Wyrażenie wykresu używające funkcji Median

Przykład – wyrażenie wykresu

Skrypt ładowania

Załaduj następujące dane w edytorze ładowania danych jako ładowanie wbudowane, aby utworzyć poniższy przykład wyrażeń wykresu.

```
Load RecNo() as RowNo, Letter, Number Inline [Letter, Number A,1 A,3 A,4 A,9 B,2 B,8 B,9];
```

Tworzenie wizualizacji

Utwórz wizualizację tabeli w arkuszu Qlik Sense z wymiarem **Letter**.

Wyrażenie wykresu

Dodaj w tabeli następujące wyrażenie jako miarę:

```
Median(Number)
```

Wynik

Letter	Median(Number)
Totals	4
A	3.5
B	8

Objaśnienie

Mediana jest uważana za liczbę „środkową”, gdy liczby są posortowane w kolejności od najmniejszej do największej. Jeśli zestaw danych ma parzystą liczbę wartości, funkcja zwraca średnią z dwóch środkowych wartości. W tym przykładzie mediana jest obliczana dla każdego zestawu wartości **A** oraz **B** i wynosi odpowiednio 3,5 oraz 8.

Mediana z **Totals** jest obliczana ze wszystkich wartości, co daje 4.

Zob. także:

p *Avg – funkcja wykresu (page 389)*

MutualInfo – funkcja wykresu

MutualInfo oblicza informacje wzajemne pomiędzy dwoma polami lub wartościami zagregowanymi w **Aggr()**.

MutualInfo zwraca zagregowaną informację wzajemną o dwóch zestawach danych. Pozwala to na analizę kluczowych czynników wpływających między polem a potencjalnym czynnikiem wpływającym. Informacja wzajemna mierzy relacje między zestawami danych i jest agregowana dla wartości par (x, y) iterowanych po wymiarach wykresu. Informacja wzajemna jest mierzona w zakresie od 0 do 1 i może być formatowana jako wartość percentyla. **MutualInfo** definiuje się przy użyciu wyborów lub wyrażenia set.

MutualInfo umożliwia różne rodzaje analizy informacji wzajemnej:

- Informacja wzajemna parami: Obliczenie informacji wzajemnej między polem driver a polem target.
- Podział pól driver według wartości: Informacja wzajemna jest obliczana między poszczególnymi wartościami pól driver i target.
- Wybór funkcji: Użyj **MutualInfo** w wykresie siatkowym, aby wygenerować macierz, w której wszystkie pola są porównywane ze sobą na podstawie informacji wzajemnej.

MutualInfo niekoniecznie wskazuje na związek przyczynowy między polami dzielącymi informację wzajemną. Dwa pola mogą dzielić między sobą informację wzajemną, ale mogą nie być dla siebie równorzędnymi czynnikami wpływającymi. Na przykład, porównując sprzedaż lodów i temperaturę zewnętrzną, **MutualInfo** pokaże informację wzajemną między nimi. Nie wskaże natomiast, czy to temperatura zewnętrzna wpływa na sprzedaż lodów, co jest prawdopodobne, czy też to sprzedaż lodów ma wpływ na temperaturę zewnętrzną, co jest mało prawdopodobne.

Podczas obliczania informacji wzajemnej asocjacje wpływają na powiązanie i częstotliwość wartości z pól pochodzących z różnych tabel.

Zwracane wartości dla tych samych pól lub wyborów mogą się nieznacznie różnić. Wynika to z tego, że każde wywołanie **MutualInfo** działa na losowo wybranej próbce, a także z nieodłącznej losowości algorytmu **MutualInfo**.

MutualInfo można zastosować do funkcji **Aggr()**.

Składnia:

```
MutualInfo({SetExpression}] [DISTINCT] [TOTAL] field1, field2 , datatype [,
breakdownbyvalue [, samplesize ]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
field1, field2	Wyrażenia lub pola zawierające dwa zestawy próbek, dla których będzie obliczana informacja wzajemna.
datatype	Typy danych zawarte w polu driver i target, 1 lub 'dd' – dyskretne:dyskretne 2 lub 'cc' – ciągłe:ciągłe 3 lub 'cd' – ciągłe:dyskretne 4 lub 'dc' – dyskretne:ciągłe W typach danych nie jest uwzględniana wielkość liter.
breakdownbyvalue	Wartość statyczna odpowiadająca wartości w polu driver. Jeśli podano, obliczony zostanie udział informacji wzajemnej w tej wartości. Możesz użyć **ValueList()** lub **ValueLoop()**. W przypadku dodania **Null()** obliczona zostanie ogólna informacja wzajemna dla wszystkich wartości w polu driver. Podział według wartości wymaga, aby pole driver zawierało dane dyskretne.

Argument	Opis
samplesize	Liczba wartości do próbkowania z pól target i driver. Pobieranie próbek jest losowe. **MutualInfo** wymaga minimalnej wielkości próbki 80. **MutualInfo** domyślnie próbkuje tylko do 10 000 par danych, ponieważ działanie **MutualInfo** może intensywnie wykorzystywać zasoby. Można określić większą liczbę par danych w próbce. Jeśli **MutualInfo** spowoduje przekroczenie limitu czasu, zmniejsz rozmiar próbki.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

Przykłady funkcji

Przykład	Wynik
`mutualinfo (Age, Salary, 1)`	W przypadku tabeli zawierającej wymiar `Employee name` i miarę `mutualinfo(Age, Salary, 1)` zwracany jest wynik 0.99820986. Wynik ten jest wyświetlany tylko dla komórki sumy.
`mutualinfo (TOTAL Age, Salary, 1, null(), 81)`	Jeśli utworzysz panel filtrowania z wymiarem Gender, a następnie dokonasz wyborów z niego, wówczas pojawi się wynik 0,99805677 w przypadku wyboru Female oraz wynik 0,99847373 w przypadku wyboru Male. Przyczyną jest to, że selekcja wyklucza wszystkie wyniki, które nie należą do drugiej wartości wymiaru Gender.

Przykład	Wynik
`mutualinfo (TOTAL Age, Gender, 1, ValueLoop (25,35))`	0.68196996. Wybór dowolnej wartości z Gender spowoduje zmianę na 0.
`mutualinfo ({1} TOTAL Age, Salary, 1, null())`	0.99820986. Zależy to od wyborów. Wyrażenie set {1} ignoruje wszystkie wybory i wymiary.

Dane zastosowane w przykładach:

```
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

Funkcja **Skew()** zwraca skośność wyrażenia iterowanego po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
Skew([ distinct] expr)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
DISTINCT	Jeśli słowo **distinct** występuje przed wyrażeniem, wówczas wszystkie duplikaty zostaną pominięte.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie utwórz tabelę prostą z wymiarami `Type` i *MySkew* .

Dane wynikowe

Przykład	Wynik

`Table1:`
`crosstable 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;`
	Wyniki obliczenia mediany są następujące Skew():
	• `Type` wynosi `MySkew`
	• `Comparison` wynosi 0.86414768
	• `Observation` wynosi 0.32625351

Skew – funkcja wykresu

Funkcja **Skew()** zwraca zagregowaną skośność wartości wyrażenia lub pola iterowanych po wymiarach wykresu.

Składnia:

```
Skew([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
expr	Wyrażenie lub pole zawierające mierzone dane.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie utwórz tabelę prostą z wymiarem `Type` i miarą `Skew(Value)`.

Funkcja `Totals` powinna być włączona we właściwościach tabeli.

Przykład	Wynik
Table1: crosstable 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 '|');	Wyniki obliczenia mediany są następujące Skew(Value): • Total wynosi 0.23522195 • Comparison wynosi 0.86414768 • Observation wynosi 0.32625351

Zob. także:

p *Avg – funkcja wykresu (page 389)*

Stdev

Funkcja **Stdev()** zwraca odchylenie standardowe wartości podane w wyrażeniu iterowanym po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
Stdev([distinct] expr)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
distinct	Jeśli słowo **distinct** występuje przed wyrażeniem, wówczas wszystkie duplikaty zostaną pominięte.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie utwórz tabelę prostą z wymiarami `Type` i `MyStdev`.

Dane wynikowe

Przykład	Wynik

`Table1:`
`crosstable 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;`
	Wyniki obliczenia mediany są następujące Stdev():
	• `Type` wynosi `MyStdev`
	• `Comparison` wynosi 14.61245
	• `Observation` wynosi 12.507997

Stdev – funkcja wykresu

Funkcja **Stdev()** zwraca odchylenie standardowe zakresu danych zagregowanych w wyrażeniu lub polu iterowanym po wymiarach wykresu.

Składnia:

```
Stdev([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
expr	Wyrażenie lub pole zawierające mierzone dane.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie utwórz tabelę prostą z wymiarem `Type` i miarą `Stdev(Value)`.

Funkcja `Totals` powinna być włączona we właściwościach tabeli.

Przykład	Wynik

`Stdev(Value)`
`Table1:`
`crosstable 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`
`'|');`
	Wyniki obliczenia mediany są następujące Stdev(Value):
	• `Total` wynosi 15.47529
	• `Comparison` wynosi 14.61245
	• `Observation` wynosi 12.507997

Zob. także:

p *Avg – funkcja wykresu (page 389)*
p *STEYX – funkcja wykresu (page 444)*

Sterr

Funkcja **Sterr()** zwraca zagregowany błąd standardowy (stdev/sqrt(n)) dla serii wartości reprezentowanych przez wyrażenie iterowane po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
Sterr ([distinct] expr)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
expr	Wyrażenie lub pole zawierające mierzone dane.
distinct	Jeśli słowo **distinct** występuje przed wyrażeniem, wówczas wszystkie duplikaty zostaną pominięte.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące są pomijane.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

Dane wynikowe

Przykład	Wynik

`Table1:`
`crosstable 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;`
	W tabeli z wymiarami `Type` i `MySterr` wynikami obliczenia wymiaru Sterr() w skrypcie ładowania danych są:
	
	`Type MySterr`
	`Comparison 3.2674431`
	`Observation 2.7968733`

Sterr – funkcja wykresu

Funkcja **Sterr()** zwraca wartość błędu standardowego wartości średniej (stdev/sqrt(n)) dla szeregu wartości zagregowanych w wyrażeniu, iterowaną po wymiarach wykresu.

Składnia:

```
Sterr([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
expr	Wyrażenie lub pole zawierające mierzone dane.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Wartości tekstowe, wartości NULL i wartości brakujące są pomijane.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie utwórz tabelę prostą z wymiarem `Type` i miarą `Sterr(Value)`.

Funkcja `Totals` powinna być włączona we właściwościach tabeli.

Przykład	Wynik
Table1: crosstable 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 '|');	Wyniki obliczenia mediany są następujące Sterr(Value): • Total wynosi 2.4468583 • Comparison wynosi 3.2674431 • Observation wynosi 2.7968733

Zob. także:

p *Avg – funkcja wykresu (page 389)*
p *STEYX – funkcja wykresu (page 444)*

STEYX

Funkcja **STEYX()** zwraca zagregowany błąd standardowy przewidywanej wartości y dla każdej wartości x w regresji dla serii współrzędnych reprezentowanych przez liczby zestawione w pary w wyrażeniach x-expression i y-expression iterowanych po liczbie rekordów, zgodnie z definicją z klauzuli **group by**.

Składnia:

```
STEYX (y_value, x_value)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
y_value	Wyrażenie lub pole zawierające zakres wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres wartości x do zmierzenia.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

Dane wynikowe

Przykład	Wynik
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;	W tabeli z wymiarem MySTEYX wynikiem obliczenia STEYX() w skrypcie ładowania danych jest 2,0714764.

STEYX – funkcja wykresu

Funkcja **STEYX()** zwraca zagregowany błąd standardowy przy prognozowaniu wartości y dla każdej wartości x w regresji liniowej określonej szeregiem współrzędnych reprezentowanych przez pary liczb w wyrażeniu podanym argumentami **y_value** i **x_value**.

Składnia:

```
STEYX([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
y_value	Wyrażenie lub pole zawierające zakres znanych wartości y do zmierzenia.
x_value	Wyrażenie lub pole zawierające zakres znanych wartości x do zmierzenia.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Parametr funkcji agregacji nie może zawierać innych funkcji agregacji, chyba że takie wewnętrzne agregacje zawierają kwalifikator **TOTAL**. W przypadku bardziej zaawansowanych agregacji zagnieżdżonych należy stosować zaawansowaną funkcję w **Aggr**połączeniu z określonym wymiarem.

Wartości tekstowe, wartości NULL i wartości brakujące w dowolnej części pary danych powodują pominięcie całej pary danych.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie utwórz tabelę prostą z wymiarami `KnownY` i `KnownX` i miarą `Steyx(KnownY,KnownX)`.

Funkcja `Totals` powinna być włączona we właściwościach tabeli.

Przykład	Wynik
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 '|');	Wynik obliczenia funkcji STEYX(KnownY,KnownX) to 2,071 (jeśli format liczb przewiduje trzy miejsca dziesiętne).

Zob. także:

p *Avg – funkcja wykresu (page 389)*
p *Sterr – funkcja wykresu (page 441)*

Przykłady stosowania funkcji linest

Funkcje linest są stosowane w celu znajdowania wartości powiązanych z analizą regresji liniowej. W tej sekcji opisano sposoby tworzenia wizualizacji na podstawie danych z próby w celu znalezienia wartości funkcji linest dostępnych w Qlik Sense. Funkcje linest mogą być stosowane w skryptach ładowania danych i wyrażeniach wykresu.

Więcej informacji o składni i argumentach poszczególnych funkcji wykresów i skryptów linest można znaleźć w dotyczących ich tematach.

Wyrażenia danych i skryptów używane w przykładach

Załaduj następujące dane wbudowane i wyrażenia skryptu w edytorze ładowania danych na potrzeby poniższych przykładów zastosowania funkcji linest().

```
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;
```

Przykład 1: Wyrażenia skryptu używające funkcji linest

Przykład: Wyrażenia skryptu

Utwórz wizualizację z obliczeń skryptu ładowania danych

Utwórz wizualizację tabeli w arkuszu Qlik Sense z następującymi polami jako kolumnami:

- Linest_B
- Linest_DF
- Linest_F
- Linest_M
- Linest_R2
- Linest_SEB
- Linest_SEM
- Linest_SEY
- Linest_SSREG
- Linest_SSRESID

Wynik

Tabela zawierająca wyniki obliczeń funkcji linest dokonanych w ramach skryptu ładowania danych powinna wyglądać następująco:

Tabela wynikowa

Linest_B	Linest_DF	Linest_F	Linest_M	Linest_R2	Linest_SEB
-35.047	18	20.788	8.605	0.536	22.607

Tabela wynikowa

Linest_SEM	Linest_SEY	Linest_SSREG	Linest_SSRESID
1.887	48.666	49235.014	42631.186

Przykład 2: Wyrażenia wykresu używające funkcji linest

Przykład: Wyrażenia wykresu

Utwórz wizualizację tabeli w arkuszu Qlik Sense z następującymi polami jako wymiarami:

```
ValueList('Linest_b', 'Linest_df','Linest_f', 'Linest_m','Linest_r2','Linest_SEB','Linest_
SEM','Linest_SEY','Linest_SSREG','Linest_SSRESID')
```

To wyrażenie korzysta z funkcji wymiarów syntetycznych w celu utworzenia etykiet do wymiarów z nazwami funkcji linest. Z uwagi na oszczędność miejsca można zmienić nazwę tej etykiety na **Linest functions**.

Dodaj w tabeli następujące wyrażenie jako miarę:

```
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) )
```

Wyrażenie to wyświetla wartość wyniku każdej funkcji linest w kontekście odpowiedniej nazwy w wymiarze syntetycznym. Wynik funkcji `Linest_b(Y,X)` zostanie wyświetlony obok pola **linest_b** itd.

Wynik

Tabela wynikowa

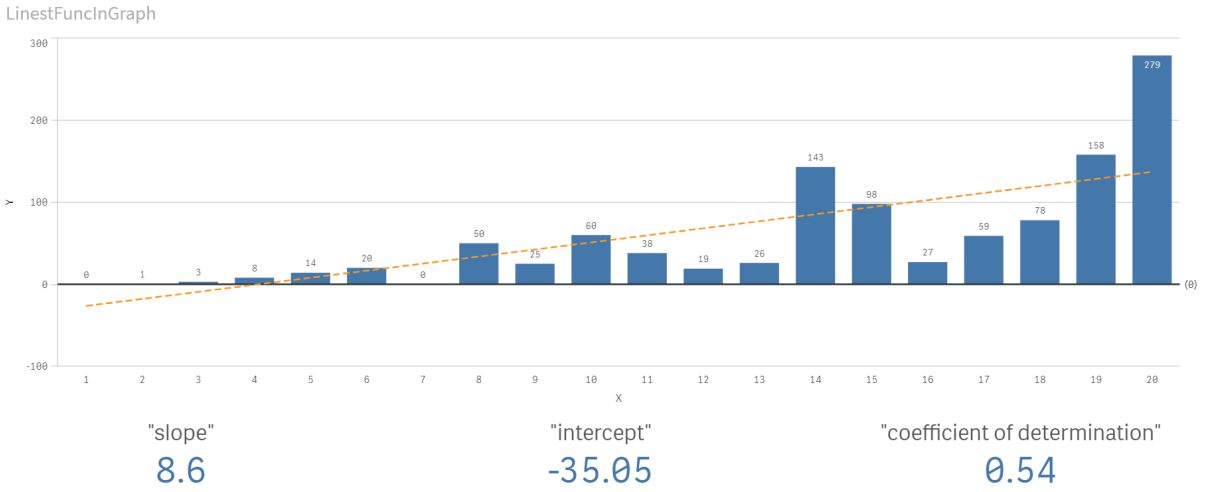
Linest functions	Linest function results
Linest_b	-35.047
Linest_df	18
Linest_f	20.788
Linest_m	8.605
Linest_r2	0.536
Linest_SEB	22.607
Linest_SEM	1.887
Linest_SEY	48.666
Linest_SSREG	49235.014
Linest_SSRESID	42631.186

Przykład 3: Wyrażenia wykresu używające funkcji linest

Przykład: Wyrażenia wykresu

1. Utwórz wizualizację wykresu słupkowego w arkuszu Qlik Sense z **X** jako wymiarem i **Y** jako miarą.
2. Dodaj linię trendu liniowego do miary Y.
3. Dodaj wizualizację wskaźnika KPI do arkusza.
    1. Dodaj *slope* jako etykietę wskaźnika KPI.
    2. Dodaj `sum(Linest_M)` jako wyrażenie do wskaźnika KPI.
4. Dodaj do arkusza drugą wizualizację wskaźnika KPI.
    1. Dodaj *intercept* jako etykietę wskaźnika KPI.
    2. Dodaj `Sum(Linest_B)` jako wyrażenie do wskaźnika KPI.
5. Dodaj do arkusza trzecią wizualizację wskaźnika KPI.
    1. Dodaj *coefficient of determination* jako etykietę wskaźnika KPI.
    2. Dodaj `Sum(Linest_R2)` jako wyrażenie do wskaźnika KPI.

Wynik



Objaśnienie

Wykres słupkowy stanowi wykres danych X i Y. Odpowiednie funkcje linest() dostarczają wartości równania regresji liniowej, na którym oparta jest linia trendu, a mianowicie y = m * x + b. W równaniu zastosowano metodę najmniejszych kwadratów do obliczenia linii prostej (linii trendu) przez zwrócenie tablicy opisującej linię, która najlepiej pasuje do danych.

Wskaźniki KPI wyświetlają wyniki funkcji linest() **sum(Linest_M)** dla nachylenia i **sum(Linest_B)** dla punktu przecięcia Y, które są zmiennymi w równaniu regresji liniowej, oraz odpowiadającą im zagregowaną wartość R2 dla współczynnika determinacji.

Funkcje testów statystycznych

Funkcji testów statystycznych można używać zarówno w skrypcie ładowania danych, jak i w wyrażeniach wykresu, ale różnią się one składnią.

Funkcje testu chi-kwadrat

Zwykle używany przy analizie zmiennych jakościowych. Możliwe jest porównywanie zaobserwowanych częstości z jednokierunkowej tabeli częstości z częstościami oczekiwanymi lub analizowanie związków między dwiema zmiennymi w tabeli tymczasowej.

Funkcje testu t

Funkcje testu t służą do statystycznego analizowania wartości średnich z dwóch populacji. Test t na dwóch próbach pozwala ustalić, czy próby te są różne. Typowe zastosowania tego testu to badanie dwóch rozkładów normalnych o nieznanych wariancjach oraz analiza eksperymentów z nieliczną próbą.

Funkcje testu Z

Badanie statystyczne średnich z dwóch populacji. Test z na dwóch próbach pozwala ustalić, czy próby te są różne. Typowe zastosowania to badanie dwóch rozkładów normalnych o znanych wariancjach oraz analiza eksperymentów z liczną próbą.

Funkcje testu Chi-kwadrat

Zwykle używany przy analizie zmiennych jakościowych. Możliwe jest porównywanie zaobserwowanych częstości z jednokierunkowej tabeli częstości z częstościami oczekiwanymi lub analizowanie związków między dwiema zmiennymi w tabeli tymczasowej.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.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Chi2Test_chi2
Funkcja **Chi2Test_chi2()** zwraca zagregowaną wartość testu chi^2 dla jednego lub dwóch szeregów wartości.

```
 Funkcja Chi2Test_chi2() zwraca zagregowaną wartość testu chi2 dla jednego
lub dwóch szeregów wartości.(col, row, actual_value[, expected_value])
```

Chi2Test_df
Funkcja **Chi2Test_df()** zwraca zagregowaną wartość df (stopni swobody) testu chi^2 dla jednego lub dwóch szeregów wartości.

```
 Funkcja Chi2Test_df() zwraca zagregowaną wartość df (stopni swobody) testu
chi2 dla jednego lub dwóch szeregów wartości.(col, row, actual_value[,
expected_value])
```

Chi2Test_p
Funkcja **Chi2Test_p()** zwraca zagregowaną wartość p (istotności) testu chi^2 dla jednego lub dwóch szeregów wartości.

```
Chi2Test_p — funkcja wykresu(col, row, actual_value[, expected_value])
```

Zob. także:

p *Funkcje testu t (page 453)*
p *Funkcje testu Z (page 489)*

Chi2Test_chi2

Funkcja **Chi2Test_chi2()** zwraca zagregowaną wartość testu chi^2 dla jednego lub dwóch szeregów wartości.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

W Qlik Sense wszystkie funkcje testu chi^2 mają takie same argumenty.

Składnia:

```
Chi2Test_chi2(col, row, actual_value[, expected_value])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
col, row	Określona kolumna i wiersz w macierzy testowanych wartości.
actual_value	Wartość zaobserwowana w określonej macierzy **col** i **row**.
expected_value	Wartość oczekiwana rozkładu w określonej macierzy **col** i **row**.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
Chi2Test_chi2( Grp, Grade, Count )
Chi2Test_chi2( Gender, Description, Observed, Expected )
```

Zob. także:

p *Przykłady stosowania funkcji chi2-test w wykresach (page 506)*
p *Przykłady stosowania funkcji chi2-test w skrypcie ładowania danych (page 508)*

Chi2Test_df

Funkcja **Chi2Test_df()** zwraca zagregowaną wartość df (stopni swobody) testu chi^2 dla jednego lub dwóch szeregów wartości.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

W Qlik Sense wszystkie funkcje testu chi^2 mają takie same argumenty.

Składnia:

```
Chi2Test_df(col, row, actual_value[, expected_value])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
col, row	Określona kolumna i wiersz w macierzy testowanych wartości.
actual_value	Wartość zaobserwowana w określonej macierzy **col** i **row**.
expected_value	Wartość oczekiwana rozkładu w określonej macierzy **col** i **row**.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
Chi2Test_df( Grp, Grade, Count )
Chi2Test_df( Gender, Description, Observed, Expected )
```

Zob. także:

p *Przykłady stosowania funkcji chi2-test w wykresach (page 506)*
p *Przykłady stosowania funkcji chi2-test w skrypcie ładowania danych (page 508)*

Chi2Test_p – funkcja wykresu

Funkcja **Chi2Test_p()** zwraca zagregowaną wartość p (istotności) testu chi^2 dla jednego lub dwóch szeregów wartości. Test można przeprowadzić względem wartości **actual_value** podczas testowania wariacji w określonych macierzach **col** i **row**, albo poprzez porównanie wartości **actual_value** z odpowiednimi wartościami **expected_value** (jeśli zostały określone).

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

W Qlik Sense wszystkie funkcje testu chi^2 mają takie same argumenty.

Składnia:

```
Chi2Test_p(col, row, actual_value[, expected_value])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
col, row	Określona kolumna i wiersz w macierzy testowanych wartości.
actual_value	Wartość zaobserwowana w określonej macierzy **col** i **row**.
expected_value	Wartość oczekiwana rozkładu w określonej macierzy **col** i **row**.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
Chi2Test_p( Grp, Grade, Count )
Chi2Test_p( Gender, Description, Observed, Expected )
```

Zob. także:

p *Przykłady stosowania funkcji chi2-test w wykresach (page 506)*
p *Przykłady stosowania funkcji chi2-test w skrypcie ładowania danych (page 508)*

Funkcje testu t

Funkcje testu t służą do statystycznego analizowania wartości średnich z dwóch populacji. Test t na dwóch próbach pozwala ustalić, czy próby te są różne. Typowe zastosowania tego testu to badanie dwóch rozkładów normalnych o nieznanych wariancjach oraz analiza eksperymentów z nieliczną próbą.

W następujących sekcjach funkcje testów statystycznych t zostały pogrupowane zgodnie z testem t Studenta dla próby, który ma zastosowanie do każdego typu funkcji.

Tworzenie typowego raportu t-test (page 510)

Testy t z dwóch niezależnych prób

Poniższe funkcje mają zastosowanie do testów t Studenta dla dwóch prób niezależnych.

ttest_conf
Funkcja **TTest_conf** zwraca zagregowaną wartość przedziału ufności testu t dla dwóch niezależnych prób.

```
 Funkcja TTest_conf zwraca zagregowaną wartość przedziału ufności testu t dla
dwóch niezależnych prób. ( grp, value [, sig[, eq_var]])
```

ttest_df
Funkcja **TTest_df()** zwraca zagregowaną wartość df (stopni swobody) testu t-Studenta dla dwóch niezależnych szeregów wartości.

```
Funkcja TTest_df() zwraca zagregowaną wartość df (stopni swobody) testu t-Studenta dla dwóch niezależnych szeregów wartości. (grp, value [, eq_var)
```

ttest_dif
TTest_dif() to funkcja liczbowa, która zwraca zagregowaną wartość średniej różnicy testu t-Studenta dla dwóch niezależnych szeregów wartości.

```
TTest_dif() to funkcja liczbowa, która zwraca zagregowaną wartość średniej różnicy testu t-Studenta dla dwóch niezależnych szeregów wartości. (grp, value)
```

ttest_lower
Funkcja **TTest_lower()** zwraca zagregowaną wartość dolnego końca przedziału ufności dla dwóch niezależnych szeregów wartości.

```
Funkcja TTest_lower() zwraca zagregowaną wartość dolnego końca przedziału ufności dla dwóch niezależnych szeregów wartości. (grp, value [, sig[, eq_var]])
```

ttest_sig
Funkcja **TTest_sig()** zwraca zagregowaną wartość dwustronnego poziomu istotności testu t-Studenta dla dwóch niezależnych szeregów wartości.

```
Funkcja TTest_sig() zwraca zagregowaną wartość dwustronnego poziomu istotności testu t-Studenta dla dwóch niezależnych szeregów wartości. (grp, value [, eq_var])
```

ttest_sterr
Funkcja **TTest_sterr()** zwraca zagregowany błąd standardowy średniej różnicy testu t-Studenta dla dwóch niezależnych szeregów wartości.

```
Funkcja TTest_sterr() zwraca zagregowany błąd standardowy średniej różnicy testu t-Studenta dla dwóch niezależnych szeregów wartości. (grp, value [, eq_var])
```

ttest_t
Funkcja **TTest_t()** zwraca zagregowaną wartość t dla dwóch niezależnych szeregów wartości.

```
Funkcja TTest_t() zwraca zagregowaną wartość t dla dwóch niezależnych szeregów wartości. (grp, value [, eq_var])
```

ttest_upper
Funkcja **TTest_upper()** zwraca zagregowaną wartość górnego końca przedziału ufności dla dwóch niezależnych szeregów wartości.

Funkcja TTest_upper() zwraca zagregowaną wartość górnego końca przedziału ufności dla dwóch niezależnych szeregów wartości. (grp, value [, sig [, eq_var]]**)**

Testy t z dwóch niezależnych prób ważonych

Poniższe funkcje mają zastosowanie do testów t Studenta z dwóch niezależnych prób, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym.

ttestw_conf

Funkcja **TTestw_conf()** zwraca zagregowaną wartość t dla dwóch niezależnych szeregów wartości.

Funkcja TTestw_conf() zwraca zagregowaną wartość t dla dwóch niezależnych szeregów wartości. (weight, grp, value [, sig[, eq_var]]**)**

ttestw_df

Funkcja **TTestw_df()** zwraca zagregowaną wartość df (stopni swobody) testu t-Studenta dla dwóch niezależnych szeregów wartości.

Funkcja TTestw_df() zwraca zagregowaną wartość df (stopni swobody) testu t-Studenta dla dwóch niezależnych szeregów wartości. (weight, grp, value [, eq_var]**)**

ttestw_dif

Funkcja **TTestw_dif()** zwraca zagregowaną średnią różnicę testu t-Studenta dla dwóch niezależnych szeregów wartości.

Funkcja TTestw_dif() zwraca zagregowaną średnią różnicę testu t-Studenta dla dwóch niezależnych szeregów wartości. (weight, grp, value**)**

ttestw_lower

Funkcja **TTestw_lower()** zwraca zagregowaną wartość dolnego końca przedziału ufności dla dwóch niezależnych szeregów wartości.

Funkcja TTestw_lower() zwraca zagregowaną wartość dolnego końca przedziału ufności dla dwóch niezależnych szeregów wartości. (weight, grp, value [, sig [, eq_var]]**)**

ttestw_sig

Funkcja **TTestw_sig()** zwraca zagregowaną wartość dwustronnego poziomu istotności testu t-Studenta dla dwóch niezależnych szeregów wartości.

Funkcja TTestw_sig() zwraca zagregowaną wartość dwustronnego poziomu istotności testu t-Studenta dla dwóch niezależnych szeregów wartości. (weight, grp, value [, eq_var]**)**

ttestw_sterr

Funkcja **TTestw_sterr()** zwraca zagregowany błąd standardowy średniej różnicy testu t-Studenta dla dwóch niezależnych szeregów wartości.

Funkcja TTestw_sterr() zwraca zagregowany błąd standardowy średniej różnicy testu t-Studenta dla dwóch niezależnych szeregów wartości. (weight, grp, value [, eq_var]**)**

ttestw_t
Funkcja **TTestw_t()** zwraca zagregowaną wartość t dla dwóch niezależnych szeregów wartości.

Funkcja TTestw_t() zwraca zagregowaną wartość t dla dwóch niezależnych szeregów wartości. (weight, grp, value [, eq_var]**)**

ttestw_upper
Funkcja **TTestw_upper()** zwraca zagregowaną wartość górnego końca przedziału ufności dla dwóch niezależnych szeregów wartości.

Funkcja TTestw_upper() zwraca zagregowaną wartość górnego końca przedziału ufności dla dwóch niezależnych szeregów wartości. (weight, grp, value [, sig [, eq_var]]**)**

Testy t z jednej próby

Poniższe funkcje mają zastosowanie do testów t Studenta dla jednej próby.

ttest1_conf
Funkcja **TTest1_conf()** zwraca zagregowaną wartość przedziału ufności dla jednego szeregu wartości.

Funkcja TTest1_conf() zwraca zagregowaną wartość przedziału ufności dla jednego szeregu wartości. (value [, sig]**)**

ttest1_df
Funkcja **TTest1_df()** zwraca zagregowaną wartość df (stopni swobody) testu t-Studenta dla jednego szeregu wartości.

Funkcja TTest1_df() zwraca zagregowaną wartość df (stopni swobody) testu t-Studenta dla jednego szeregu wartości. (value**)**

ttest1_dif
Funkcja **TTest1_dif()** zwraca zagregowaną średnią różnicę testu t-Studenta dla jednego szeregu wartości.

Funkcja TTest1_dif() zwraca zagregowaną średnią różnicę testu t-Studenta dla jednego szeregu wartości. (value**)**

ttest1_lower
Funkcja **TTest1_lower()** zwraca zagregowaną wartość dolnego końca przedziału ufności dla jednego szeregu wartości.

Funkcja TTest1_lower() zwraca zagregowaną wartość dolnego końca przedziału ufności dla jednego szeregu wartości. (value [, sig]**)**

ttest1_sig
Funkcja **TTest1_sig()** zwraca zagregowaną wartość dwustronnego poziomu istotności testu t-Studenta dla jednego szeregu wartości.

```
Funkcja TTest1_sig() zwraca zagregowaną wartość dwustronnego poziomu
istotności testu t-Studenta dla jednego szeregu wartości. (value)
```

ttest1_sterr
Funkcja **TTest1_sterr()** zwraca zagregowany błąd standardowy średniej różnicy testu t-Studenta dla jednego szeregu wartości.

```
Funkcja TTest1_sterr() zwraca zagregowany błąd standardowy średniej różnicy
testu t-Studenta dla jednego szeregu wartości. (value)
```

ttest1_t
Funkcja **TTest1_t()** zwraca zagregowaną wartość t dla jednego szeregu wartości.

```
Funkcja TTest1_t() zwraca zagregowaną wartość t dla jednego szeregu
wartości. (value)
```

ttest1_upper
Funkcja **TTest1_upper()** zwraca zagregowaną wartość górnego końca przedziału ufności dla jednego szeregu wartości.

```
Funkcja TTest1_upper() zwraca zagregowaną wartość górnego końca przedziału
ufności dla jednego szeregu wartości. (value [, sig])
```

Testy t z jednej próby ważonej

Poniższe funkcje mają zastosowanie do testów t Studenta z jednej próby, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym.

ttest1w_conf
TTest1w_conf() to funkcja **liczbowa**, która zwraca zagregowaną wartość przedziału ufności dla jednego szeregu wartości.

```
TTest1w_conf() to funkcja liczbowa, która zwraca zagregowaną wartość
przedziału ufności dla jednego szeregu wartości. (weight, value [, sig])
```

ttest1w_df
Funkcja **TTest1w_df()** zwraca zagregowaną wartość df (stopni swobody) testu t-Studenta dla jednego szeregu wartości.

```
Funkcja TTest1w_df() zwraca zagregowaną wartość df (stopni swobody) testu t-
Studenta dla jednego szeregu wartości. (weight, value)
```

ttest1w_dif
Funkcja **TTest1w_dif()** zwraca zagregowaną średnią różnicę testu t-Studenta dla jednego szeregu wartości.

```
Funkcja TTest1w_dif() zwraca zagregowaną średnią różnicę testu t-Studenta
dla jednego szeregu wartości. (weight, value)
```

ttest1w_lower
Funkcja **TTest1w_lower()** zwraca zagregowaną wartość dolnego końca przedziału ufności dla jednego szeregu wartości.

```
Funkcja TTest1w_lower() zwraca zagregowaną wartość dolnego końca przedziału ufności dla jednego szeregu wartości. (weight, value [, sig])
```

ttest1w_sig
Funkcja **TTest1w_sig()** zwraca zagregowaną wartość dwustronnego poziomu istotności testu t-Studenta dla jednego szeregu wartości.

```
Funkcja TTest1w_sig() zwraca zagregowaną wartość dwustronnego poziomu istotności testu t-Studenta dla jednego szeregu wartości. (weight, value)
```

ttest1w_sterr
Funkcja **TTest1w_sterr()** zwraca zagregowany błąd standardowy średniej różnicy testu t-Studenta dla jednego szeregu wartości.

```
Funkcja TTest1w_sterr() zwraca zagregowany błąd standardowy średniej różnicy testu t-Studenta dla jednego szeregu wartości. (weight, value)
```

ttest1w_t
Funkcja **TTest1w_t()** zwraca zagregowaną wartość t dla jednego szeregu wartości.

```
Funkcja TTest1w_t() zwraca zagregowaną wartość t dla jednego szeregu wartości. ( weight, value)
```

ttest1w_upper
Funkcja **TTest1w_upper()** zwraca zagregowaną wartość górnego końca przedziału ufności dla jednego szeregu wartości.

```
Funkcja TTest1w_upper() zwraca zagregowaną wartość górnego końca przedziału ufności dla jednego szeregu wartości. (weight, value [, sig])
```

TTest_conf

Funkcja **TTest_conf** zwraca zagregowaną wartość przedziału ufności testu t dla dwóch niezależnych prób.

Ta funkcja ma zastosowanie do testów t-Studenta dla prób niezależnych.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest_conf ( grp, value [, sig [, eq_var]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTest_conf( Group, Value )
TTest_conf( Group, Value, Sig, false )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest_df

Funkcja **TTest_df()** zwraca zagregowaną wartość df (stopni swobody) testu t-Studenta dla dwóch niezależnych szeregów wartości.

Ta funkcja ma zastosowanie do testów t-Studenta dla prób niezależnych.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest_df (grp, value [, eq_var])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTest_df( Group, Value )
TTest_df( Group, Value, false )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest_dif

TTest_dif() to funkcja liczbowa, która zwraca zagregowaną wartość średniej różnicy testu t-Studenta dla dwóch niezależnych szeregów wartości.

Ta funkcja ma zastosowanie do testów t-Studenta dla prób niezależnych.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest_dif (grp, value [, eq_var] )
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTest_dif( Group, Value )
TTest_dif( Group, Value, false )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest_lower

Funkcja **TTest_lower()** zwraca zagregowaną wartość dolnego końca przedziału ufności dla dwóch niezależnych szeregów wartości.

Ta funkcja ma zastosowanie do testów t-Studenta dla prób niezależnych.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest_lower (grp, value [, sig [, eq_var]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTest_lower( Group, Value )
TTest_lower( Group, Value, Sig, false )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest_sig

Funkcja **TTest_sig()** zwraca zagregowaną wartość dwustronnego poziomu istotności testu t-Studenta dla dwóch niezależnych szeregów wartości.

Ta funkcja ma zastosowanie do testów t-Studenta dla prób niezależnych.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest_sig (grp, value [, eq_var])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTest_sig( Group, Value )
TTest_sig( Group, Value, false )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest_sterr

Funkcja **TTest_sterr()** zwraca zagregowany błąd standardowy średniej różnicy testu t-Studenta dla dwóch niezależnych szeregów wartości.

Ta funkcja ma zastosowanie do testów t-Studenta dla prób niezależnych.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest_sterr (grp, value [, eq_var])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTest_sterr( Group, Value )
TTest_sterr( Group, Value, false )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest_t

Funkcja **TTest_t()** zwraca zagregowaną wartość t dla dwóch niezależnych szeregów wartości.

Ta funkcja ma zastosowanie do testów t-Studenta dla prób niezależnych.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest_t(grp, value[, eq_var])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
TTest_t( Group, Value, false )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest_upper

Funkcja **TTest_upper()** zwraca zagregowaną wartość górnego końca przedziału ufności dla dwóch niezależnych szeregów wartości.

Ta funkcja ma zastosowanie do testów t-Studenta dla prób niezależnych.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest_upper (grp, value [, sig [, eq_var]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTest_upper( Group, Value )
TTest_upper( Group, Value, sig, false )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTestw_conf

Funkcja **TTestw_conf()** zwraca zagregowaną wartość t dla dwóch niezależnych szeregów wartości.

Ta funkcja ma zastosowanie do testów t-Studenta z dwóch niezależnych prób, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTestw_conf (weight, grp, value [, sig [, eq_var]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
weight	Każda wartość w parametrze **value** może być zliczana co najmniej raz odpowiednio do wartości wagi w parametrze **weight**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTestw_conf( Weight, Group, Value )
TTestw_conf( Weight, Group, Value, sig, false )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTestw_df

Funkcja **TTestw_df()** zwraca zagregowaną wartość df (stopni swobody) testu t-Studenta dla dwóch niezależnych szeregów wartości.

Ta funkcja ma zastosowanie do testów t-Studenta z dwóch niezależnych prób, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTestw_df (weight, grp, value [, eq_var])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
weight	Każda wartość w parametrze **value** może być zliczana co najmniej raz odpowiednio do wartości wagi w parametrze **weight**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTestw_df( Weight, Group, Value )
TTestw_df( Weight, Group, Value, false )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTestw_dif

Funkcja **TTestw_dif()** zwraca zagregowaną średnią różnicę testu t-Studenta dla dwóch niezależnych szeregów wartości.

Ta funkcja ma zastosowanie do testów t-Studenta z dwóch niezależnych prób, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTestw_dif (weight, grp, value)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
weight	Każda wartość w parametrze **value** może być zliczana co najmniej raz odpowiednio do wartości wagi w parametrze **weight**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTestw_dif( Weight, Group, Value )
TTestw_dif( Weight, Group, Value, false )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTestw_lower

Funkcja **TTestw_lower()** zwraca zagregowaną wartość dolnego końca przedziału ufności dla dwóch niezależnych szeregów wartości.

Ta funkcja ma zastosowanie do testów t-Studenta z dwóch niezależnych prób, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTestw_lower (weight, grp, value [, sig [, eq_var]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
weight	Każda wartość w parametrze **value** może być zliczana co najmniej raz odpowiednio do wartości wagi w parametrze **weight**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTestw_lower( Weight, Group, Value )
TTestw_lower( Weight, Group, Value, sig, false )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTestw_sig

Funkcja **TTestw_sig()** zwraca zagregowaną wartość dwustronnego poziomu istotności testu t-Studenta dla dwóch niezależnych szeregów wartości.

Ta funkcja ma zastosowanie do testów t-Studenta z dwóch niezależnych prób, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTestw_sig ( weight, grp, value [, eq_var])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
weight	Każda wartość w parametrze **value** może być zliczana co najmniej raz odpowiednio do wartości wagi w parametrze **weight**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTestw_sig( Weight, Group, Value )
```

```
TTestw_sig( Weight, Group, Value, false )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTestw_sterr

Funkcja **TTestw_sterr()** zwraca zagregowany błąd standardowy średniej różnicy testu t-Studenta dla dwóch niezależnych szeregów wartości.

Ta funkcja ma zastosowanie do testów t-Studenta z dwóch niezależnych prób, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTestw_sterr (weight, grp, value [, eq_var])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
weight	Każda wartość w parametrze **value** może być zliczana co najmniej raz odpowiednio do wartości wagi w parametrze **weight**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTestw_sterr( Weight, Group, Value )
TTestw_sterr( Weight, Group, Value, false )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTestw_t

Funkcja **TTestw_t()** zwraca zagregowaną wartość t dla dwóch niezależnych szeregów wartości.

Ta funkcja ma zastosowanie do testów t-Studenta z dwóch niezależnych prób, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
ttestw_t (weight, grp, value [, eq_var])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
weight	Każda wartość w parametrze **value** może być zliczana co najmniej raz odpowiednio do wartości wagi w parametrze **weight**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTestw_t( Weight, Group, Value )
TTestw_t( Weight, Group, Value, false )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTestw_upper

Funkcja **TTestw_upper()** zwraca zagregowaną wartość górnego końca przedziału ufności dla dwóch niezależnych szeregów wartości.

Ta funkcja ma zastosowanie do testów t-Studenta z dwóch niezależnych prób, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTestw_upper (weight, grp, value [, sig [, eq_var]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
weight	Każda wartość w parametrze **value** może być zliczana co najmniej raz odpowiednio do wartości wagi w parametrze **weight**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.

Argument	Opisu
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTestw_upper( Weight, Group, Value )
TTestw_upper( Weight, Group, Value, sig, false )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest1_conf

Funkcja **TTest1_conf()** zwraca zagregowaną wartość przedziału ufności dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów t-Studenta dla jednej próby.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest1_conf (value [, sig ])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Próby, z których będą obliczane wartości. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.

Argument	Opisu
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTest1_conf( Value )
TTest1_conf( Value, 0.005 )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest1_df

Funkcja **TTest1_df()** zwraca zagregowaną wartość df (stopni swobody) testu t-Studenta dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów t-Studenta dla jednej próby.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest1_df (value)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Próby, z których będą obliczane wartości. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
TTest1_df( Value )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest1_dif

Funkcja **TTest1_dif()** zwraca zagregowaną średnią różnicę testu t-Studenta dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów t-Studenta dla jednej próby.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest1_dif (value)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Próby, z których będą obliczane wartości. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
TTest1_dif( Value )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest1_lower

Funkcja **TTest1_lower()** zwraca zagregowaną wartość dolnego końca przedziału ufności dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów t-Studenta dla jednej próby.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest1_lower (value [, sig])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Próby, z których będą obliczane wartości. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTest1_lower( Value )
TTest1_lower( Value, 0.005 )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest1_sig

Funkcja **TTest1_sig()** zwraca zagregowaną wartość dwustronnego poziomu istotności testu t-Studenta dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów t-Studenta dla jednej próby.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest1_sig (value)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Próby, z których będą obliczane wartości. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
TTest1_sig( Value )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest1_sterr

Funkcja **TTest1_sterr()** zwraca zagregowany błąd standardowy średniej różnicy testu t-Studenta dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów t-Studenta dla jednej próby.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest1_sterr (value)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Próby, z których będą obliczane wartości. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
TTest1_sterr( Value )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest1_t

Funkcja **TTest1_t()** zwraca zagregowaną wartość t dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów t-Studenta dla jednej próby.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest1_t (value)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Próby, z których będą obliczane wartości. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
TTest1_t( Value )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest1_upper

Funkcja **TTest1_upper()** zwraca zagregowaną wartość górnego końca przedziału ufności dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów t-Studenta dla jednej próby.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest1_upper (value [, sig])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Próby, z których będą obliczane wartości. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTest1_upper( Value )
TTest1_upper( Value, 0.005 )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest1w_conf

TTest1w_conf() to funkcja **liczbowa**, która zwraca zagregowaną wartość przedziału ufności dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów t-Studenta z jednej próby, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym:

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest1w_conf (weight, value [, sig ])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Próby, z których będą obliczane wartości. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
weight	Każda wartość w parametrze **value** może być zliczana co najmniej raz odpowiednio do wartości wagi w parametrze **weight**.
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTest1w_conf( Weight, Value )
```

```
TTest1w_conf( Weight, Value, 0.005 )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest1w_df

Funkcja **TTest1w_df()** zwraca zagregowaną wartość df (stopni swobody) testu t-Studenta dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów t-Studenta z jednej próby, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym:

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest1w_df (weight, value)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Próby, z których będą obliczane wartości. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
weight	Każda wartość w parametrze **value** może być zliczana co najmniej raz odpowiednio do wartości wagi w parametrze **weight**.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
TTest1w_df( Weight, Value )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest1w_dif

Funkcja **TTest1w_dif()** zwraca zagregowaną średnią różnicę testu t-Studenta dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów t-Studenta z jednej próby, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym:

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest1w_dif (weight, value)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Próby, z których będą obliczane wartości. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
weight	Każda wartość w parametrze **value** może być zliczana co najmniej raz odpowiednio do wartości wagi w parametrze **weight**.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
TTest1w_dif( Weight, Value )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest1w_lower

Funkcja **TTest1w_lower()** zwraca zagregowaną wartość dolnego końca przedziału ufności dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów t-Studenta z jednej próby, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym:

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest1w_lower (weight, value [, sig ])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Próby, z których będą obliczane wartości. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
weight	Każda wartość w parametrze **value** może być zliczana co najmniej raz odpowiednio do wartości wagi w parametrze **weight**.
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTest1w_lower( Weight, Value )
TTest1w_lower( Weight, Value, 0.005 )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest1w_sig

Funkcja **TTest1w_sig()** zwraca zagregowaną wartość dwustronnego poziomu istotności testu t-Studenta dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów t-Studenta z jednej próby, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym:

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest1w_sig (weight, value)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Próby, z których będą obliczane wartości. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
weight	Każda wartość w parametrze **value** może być zliczana co najmniej raz odpowiednio do wartości wagi w parametrze **weight**.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
TTest1w_sig( Weight, Value )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest1w_sterr

Funkcja **TTest1w_sterr()** zwraca zagregowany błąd standardowy średniej różnicy testu t-Studenta dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów t-Studenta z jednej próby, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym:

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest1w_sterr (weight, value)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Próby, z których będą obliczane wartości. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
weight	Każda wartość w parametrze **value** może być zliczana co najmniej raz odpowiednio do wartości wagi w parametrze **weight**.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
TTest1w_sterr( Weight, Value )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest1w_t

Funkcja **TTest1w_t()** zwraca zagregowaną wartość t dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów t-Studenta z jednej próby, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym:

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest1w_t ( weight, value)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Próby, z których będą obliczane wartości. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
weight	Każda wartość w parametrze **value** może być zliczana co najmniej raz odpowiednio do wartości wagi w parametrze **weight**.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
TTest1w_t( Weight, Value )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

TTest1w_upper

Funkcja **TTest1w_upper()** zwraca zagregowaną wartość górnego końca przedziału ufności dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów t-Studenta z jednej próby, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym:

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
TTest1w_upper (weight, value [, sig])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Próby, z których będą obliczane wartości. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
weight	Każda wartość w parametrze **value** może być zliczana co najmniej raz odpowiednio do wartości wagi w parametrze **weight**.
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
TTest1w_upper( Weight, Value )
TTest1w_upper( Weight, Value, 0.005 )
```

Zob. także:

p *Tworzenie typowego raportu t-test (page 510)*

Funkcje testu Z

Badanie statystyczne średnich z dwóch populacji. Test z na dwóch próbach pozwala ustalić, czy próby te są różne. Typowe zastosowania to badanie dwóch rozkładów normalnych o znanych wariancjach oraz analiza eksperymentów z liczną próbą.

Funkcje testów statystycznych z zostały pogrupowane zgodnie z typem szeregu danych wejściowych mającym zastosowanie do danej funkcji.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Przykłady stosowania funkcji z-test (page 514)

Funkcje w formacie jednokolumnowym

W przypadku testów z z prostym szeregiem danych wejściowych zastosowanie mają następujące funkcje.

ztest_conf

Funkcja **ZTest_conf()** zwraca zagregowaną wartość z dla jednego szeregu wartości.

```
Funkcja ZTest_conf() zwraca zagregowaną wartość z dla jednego szeregu
wartości. (value [, sigma [, sig ])
```

ztest_dif

Funkcja **ZTest_dif()** zwraca zagregowaną średnią różnicę testu z dla jednego szeregu wartości.

```
Funkcja ZTest_dif() zwraca zagregowaną średnią różnicę testu z dla jednego
szeregu wartości. (value [, sigma])
```

ztest_sig

Funkcja **ZTest_sig()** zwraca zagregowaną wartość dwustronnego poziomu istotności testu z dla jednego szeregu wartości.

```
Funkcja ZTest_sig() zwraca zagregowaną wartość dwustronnego poziomu
istotności testu z dla jednego szeregu wartości. (value [, sigma])
```

ztest_sterr

Funkcja **ZTest_sterr()** zwraca zagregowany błąd standardowy średniej różnicy testu z dla jednego szeregu wartości.

```
Funkcja ZTest_sterr() zwraca zagregowany błąd standardowy średniej różnicy
testu z dla jednego szeregu wartości. (value [, sigma])
```

ztest_z

Funkcja **ZTest_z()** zwraca zagregowaną wartość z dla jednego szeregu wartości.

```
Funkcja ZTest_z() zwraca zagregowaną wartość z dla jednego szeregu wartości.
(value [, sigma])
```

ztest_lower

Funkcja **ZTest_lower()** zwraca zagregowaną wartość dolnego końca przedziału ufności dla dwóch niezależnych szeregów wartości.

```
Funkcja ZTest_lower() zwraca zagregowaną wartość dolnego końca przedziału
ufności dla dwóch niezależnych szeregów wartości. (grp, value [, sig [, eq_
var]])
```

ztest_upper

Funkcja **ZTest_upper()** zwraca zagregowaną wartość górnego końca przedziału ufności dla dwóch niezależnych szeregów wartości.

```
Funkcja ZTest_upper() zwraca zagregowaną wartość górnego końca przedziału
ufności dla dwóch niezależnych szeregów wartości. (grp, value [, sig [, eq_
var]])
```

Funkcje w formacie ważonym dwukolumnowym

Poniższe funkcje mają zastosowanie do testów z, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym.

ztestw_conf

Funkcja **ZTestw_conf()** zwraca zagregowaną wartość przedziału ufności z dla jednego szeregu wartości.

```
Funkcja ZTestw_conf() zwraca zagregowaną wartość przedziału ufności z dla jednego szeregu wartości. (weight, value [, sigma [, sig]])
```

ztestw_dif

Funkcja **ZTestw_dif()** zwraca zagregowaną średnią różnicę testu z dla jednego szeregu wartości.

```
Funkcja ZTestw_dif() zwraca zagregowaną średnią różnicę testu z dla jednego szeregu wartości.  (weight, value [, sigma])
```

ztestw_lower

Funkcja **ZTestw_lower()** zwraca zagregowaną wartość dolnego końca przedziału ufności dla dwóch niezależnych szeregów wartości.

```
Funkcja ZTestw_lower() zwraca zagregowaną wartość dolnego końca przedziału ufności dla dwóch niezależnych szeregów wartości. (weight, value [, sigma])
```

ztestw_sig

Funkcja **ZTestw_sig()** zwraca zagregowaną wartość dwustronnego poziomu istotności testu z dla jednego szeregu wartości.

```
Funkcja ZTestw_sig() zwraca zagregowaną wartość dwustronnego poziomu istotności testu z dla jednego szeregu wartości. (weight, value [, sigma])
```

ztestw_sterr

Funkcja **ZTestw_sterr()** zwraca zagregowany błąd standardowy średniej różnicy testu z dla jednego szeregu wartości.

```
Funkcja ZTestw_sterr() zwraca zagregowany błąd standardowy średniej różnicy testu z dla jednego szeregu wartości. (weight, value [, sigma])
```

ztestw_upper

Funkcja **ZTestw_upper()** zwraca zagregowaną wartość górnego końca przedziału ufności dla dwóch niezależnych szeregów wartości.

```
Funkcja ZTestw_upper() zwraca zagregowaną wartość górnego końca przedziału ufności dla dwóch niezależnych szeregów wartości. (weight, value [, sigma])
```

ztestw_z

Funkcja **ZTestw_z()** zwraca zagregowaną wartość z dla jednego szeregu wartości.

```
Funkcja ZTestw_z() zwraca zagregowaną wartość z dla jednego szeregu wartości. (weight, value [, sigma])
```

ZTest_z

Funkcja **ZTest_z()** zwraca zagregowaną wartość z dla jednego szeregu wartości.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
ZTest_z(value[, sigma])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Przyjmowana jest średnia w populacji równa 0. Jeśli test ma być wykonywany względem innej średniej, należy odjąć jej wartość od wartości próby.
sigma	Jeśli odchylenie standardowe jest znane, wówczas można je określić w parametrze **sigma**. Jeśli parametr **sigma** zostanie pominięty, wówczas stosowane będzie rzeczywiste odchylenie standardowe dla próby.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
ZTest_z( Value-TestValue )
```

Zob. także:

p *Przykłady stosowania funkcji z-test (page 514)*

ZTest_sig

Funkcja **ZTest_sig()** zwraca zagregowaną wartość dwustronnego poziomu istotności testu z dla jednego szeregu wartości.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
ZTest_sig(value[, sigma])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Przyjmowana jest średnia w populacji równa 0. Jeśli test ma być wykonywany względem innej średniej, należy odjąć jej wartość od wartości próby.
sigma	Jeśli odchylenie standardowe jest znane, wówczas można je określić w parametrze **sigma**. Jeśli parametr **sigma** zostanie pominięty, wówczas stosowane będzie rzeczywiste odchylenie standardowe dla próby.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
ZTest_sig(Value-TestValue)
```

Zob. także:

p *Przykłady stosowania funkcji z-test (page 514)*

ZTest_dif

Funkcja **ZTest_dif()** zwraca zagregowaną średnią różnicę testu z dla jednego szeregu wartości.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
ZTest_dif(value[, sigma])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Przyjmowana jest średnia w populacji równa 0. Jeśli test ma być wykonywany względem innej średniej, należy odjąć jej wartość od wartości próby.
sigma	Jeśli odchylenie standardowe jest znane, wówczas można je określić w parametrze **sigma**. Jeśli parametr **sigma** zostanie pominięty, wówczas stosowane będzie rzeczywiste odchylenie standardowe dla próby.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
ZTest_dif(Value-TestValue)
```

Zob. także:

p *Przykłady stosowania funkcji z-test (page 514)*

ZTest_sterr

Funkcja **ZTest_sterr()** zwraca zagregowany błąd standardowy średniej różnicy testu z dla jednego szeregu wartości.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
ZTest_sterr(value[, sigma])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Przyjmowana jest średnia w populacji równa 0. Jeśli test ma być wykonywany względem innej średniej, należy odjąć jej wartość od wartości próby.
sigma	Jeśli odchylenie standardowe jest znane, wówczas można je określić w parametrze **sigma**. Jeśli parametr **sigma** zostanie pominięty, wówczas stosowane będzie rzeczywiste odchylenie standardowe dla próby.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
ZTest_sterr(Value-TestValue)
```

Zob. także:

p *Przykłady stosowania funkcji z-test (page 514)*

ZTest_conf

Funkcja **ZTest_conf()** zwraca zagregowaną wartość z dla jednego szeregu wartości.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
ZTest_conf(value[, sigma[, sig]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Przyjmowana jest średnia w populacji równa 0. Jeśli test ma być wykonywany względem innej średniej, należy odjąć jej wartość od wartości próby.
sigma	Jeśli odchylenie standardowe jest znane, wówczas można je określić w parametrze **sigma**. Jeśli parametr **sigma** zostanie pominięty, wówczas stosowane będzie rzeczywiste odchylenie standardowe dla próby.
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
ZTest_conf(Value-TestValue)
```

Zob. także:

p *Przykłady stosowania funkcji z-test (page 514)*

ZTest_lower

Funkcja **ZTest_lower()** zwraca zagregowaną wartość dolnego końca przedziału ufności dla dwóch niezależnych szeregów wartości.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
ZTest_lower (grp, value [, sig [, eq_var]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
ZTest_lower( Group, Value )
ZTest_lower( Group, Value, sig, false )
```

Zob. także:

p *Przykłady stosowania funkcji z-test (page 514)*

ZTest_upper

Funkcja **ZTest_upper()** zwraca zagregowaną wartość górnego końca przedziału ufności dla dwóch niezależnych szeregów wartości.

Ta funkcja ma zastosowanie do testów t-Studenta dla prób niezależnych.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
ZTest_upper (grp, value [, sig [, eq_var]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
ZTest_upper( Group, Value )
ZTest_upper( Group, Value, sig, false )
```

Zob. także:

p *Przykłady stosowania funkcji z-test (page 514)*

ZTestw_z

Funkcja **ZTestw_z()** zwraca zagregowaną wartość z dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów z, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym:

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
ZTestw_z (weight, value [, sigma])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości powinny być zwracane przez parametr **value**. Przyjmowana jest średnia w próbie równa 0. Jeśli konieczne jest wykonanie testu względem innej średniej, odejmij tę wartość od wartości próby.
weight	Każda wartość próbki w argumencie **value** może być zliczana raz lub więcej razy, w zależności od wartości wagi w argumencie **weight**.
sigma	Jeśli odchylenie standardowe jest znane, wówczas można je określić w parametrze **sigma**. Jeśli parametr **sigma** zostanie pominięty, wówczas stosowane będzie rzeczywiste odchylenie standardowe dla próby.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
ZTestw_z( Weight, Value-TestValue)
```

Zob. także:

p *Przykłady stosowania funkcji z-test (page 514)*

ZTestw_sig

Funkcja **ZTestw_sig()** zwraca zagregowaną wartość dwustronnego poziomu istotności testu z dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów z, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym:

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
ZTestw_sig (weight, value [, sigma])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości powinny być zwracane przez parametr **value**. Przyjmowana jest średnia w próbie równa 0. Jeśli konieczne jest wykonanie testu względem innej średniej, odejmij tę wartość od wartości próby.
weight	Każda wartość próbki w argumencie **value** może być zliczana raz lub więcej razy, w zależności od wartości wagi w argumencie **weight**.
sigma	Jeśli odchylenie standardowe jest znane, wówczas można je określić w parametrze **sigma**. Jeśli parametr **sigma** zostanie pominięty, wówczas stosowane będzie rzeczywiste odchylenie standardowe dla próby.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
ZTestw_sig( Weight, Value-TestValue)
```

Zob. także:

p *Przykłady stosowania funkcji z-test (page 514)*

ZTestw_dif

Funkcja **ZTestw_dif()** zwraca zagregowaną średnią różnicę testu z dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów z, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym:

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
ZTestw_dif ( weight, value [, sigma])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości powinny być zwracane przez parametr **value**. Przyjmowana jest średnia w próbie równa 0. Jeśli konieczne jest wykonanie testu względem innej średniej, odejmij tę wartość od wartości próby.
weight	Każda wartość próbki w argumencie **value** może być zliczana raz lub więcej razy, w zależności od wartości wagi w argumencie **weight**.
sigma	Jeśli odchylenie standardowe jest znane, wówczas można je określić w parametrze **sigma**. Jeśli parametr **sigma** zostanie pominięty, wówczas stosowane będzie rzeczywiste odchylenie standardowe dla próby.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
ZTestw_dif( Weight, Value-TestValue)
```

Zob. także:

p *Przykłady stosowania funkcji z-test (page 514)*

ZTestw_sterr

Funkcja **ZTestw_sterr()** zwraca zagregowany błąd standardowy średniej różnicy testu z dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów z, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym:

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
ZTestw_sterr (weight, value [, sigma])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości powinny być zwracane przez parametr **value**. Przyjmowana jest średnia w próbie równa 0. Jeśli konieczne jest wykonanie testu względem innej średniej, odejmij tę wartość od wartości próby.
weight	Każda wartość próbki w argumencie **value** może być zliczana raz lub więcej razy, w zależności od wartości wagi w argumencie **weight**.
sigma	Jeśli odchylenie standardowe jest znane, wówczas można je określić w parametrze **sigma**. Jeśli parametr **sigma** zostanie pominięty, wówczas stosowane będzie rzeczywiste odchylenie standardowe dla próby.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
ZTestw_sterr( Weight, Value-TestValue)
```

Zob. także:

p *Przykłady stosowania funkcji z-test (page 514)*

ZTestw_conf

Funkcja **ZTestw_conf()** zwraca zagregowaną wartość przedziału ufności z dla jednego szeregu wartości.

Ta funkcja ma zastosowanie do testów z, w których szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym:

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
ZTest_conf(weight, value[, sigma[, sig]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Przyjmowana jest średnia w populacji równa 0. Jeśli test ma być wykonywany względem innej średniej, należy odjąć jej wartość od wartości próby.
weight	Każda wartość próbki w argumencie **value** może być zliczana raz lub więcej razy, w zależności od wartości wagi w argumencie **weight**.
sigma	Jeśli odchylenie standardowe jest znane, wówczas można je określić w parametrze **sigma**. Jeśli parametr **sigma** zostanie pominięty, wówczas stosowane będzie rzeczywiste odchylenie standardowe dla próby.
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykład:

```
ZTestw_conf( Weight, Value-TestValue)
```

Zob. także:

p *Przykłady stosowania funkcji z-test (page 514)*

ZTestw_lower

Funkcja **ZTestw_lower()** zwraca zagregowaną wartość dolnego końca przedziału ufności dla dwóch niezależnych szeregów wartości.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
ZTestw_lower (grp, value [, sig [, eq_var]])
```

Typ zwracanych danych: numeric

Argumenty:

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
ZTestw_lower( Group, Value )
ZTestw_lower( Group, Value, sig, false )
```

Zob. także:

p *Przykłady stosowania funkcji z-test (page 514)*

ZTestw_upper

Funkcja **ZTestw_upper()** zwraca zagregowaną wartość górnego końca przedziału ufności dla dwóch niezależnych szeregów wartości.

Ta funkcja ma zastosowanie do testów t-Studenta dla prób niezależnych.

Jeśli funkcja jest używana w skrypcie ładowania danych, wartości są iterowane po liczbie rekordów, zgodnie z definicją z klauzuli group by.

Jeśli funkcja jest używana w wyrażeniu wykresu, wartości są iterowane po wymiarach wykresu.

Składnia:

```
ZTestw_upper (grp, value [, sig [, eq_var]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
value	Wartości próby, z których będą obliczane wartości. Wartości próby muszą być pogrupowane logicznie jak to zostało określone przez dokładnie dwie wartości w grupie **group**. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego wartości próby, polu temu automatycznie zostanie nadana nazwa **Value**.
grp	Pole zawierające nazwy każdej z dwóch grup prób. Jeśli w skrypcie ładowania nie podano nazwy pola dotyczącego grupy, polu temu automatycznie zostanie nadana nazwa **Type**.
sig	W parametrze **sig** można określić dwustronny poziom istotności. Jeśli poziom zostanie pominięty, wówczas wartość **sig** zostanie ustawiona na 0,025, co oznacza 95% przedział ufności.
eq_var	Jeśli wartość **eq_var** zostanie określona jakoFalse (0), wówczas zostanie przyjęte założenie osobnych wariancji dwóch prób. Jeśli wartość **eq_var** zostanie określona jakoTrue (1), wówczas zostanie przyjęte założenie równych wariancji prób.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące w wyniku wyrażenia powodują zwrócenie wartości NULL.

Przykłady:

```
ZTestw_upper( Group, Value )
ZTestw_upper( Group, Value, sig, false )
```

Zob. także:

p *Przykłady stosowania funkcji z-test (page 514)*

Przykłady funkcji testów statystycznych

W tej sekcji przedstawiono przykłady funkcji testów statystycznych stosowanych w wykresach i skryptach ładowania danych.

Przykłady stosowania funkcji chi2-test w wykresach

Funkcje chi2-test są stosowane w celu znajdowania wartości powiązanych z analizą statystyczną chi-kwadrat.

W tej sekcji opisano sposoby tworzenia wizualizacji na podstawie danych z próby w celu znalezienia wartości funkcji testu zgodności chi-kwadrat dostępnych w Qlik Sense. Więcej informacji o składni i argumentach dotyczących funkcji wykresów chi2-test można znaleźć w stosownych opisach.

Ładowanie danych na potrzeby prób

Do skryptu powinny zostać załadowane trzy zestawy danych z próby opisujące trzy różne próby statystyczne.

Wykonaj następujące czynności:

1. Utwórz nową aplikację.
2. W procedurze ładowania danych dodaj następujące wiersze:

```
// Sample_1 data is pre-aggregated... Note: make sure you set your DecimalSep='.' at the
top of the script.
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. Kliknij aby załadować dane.

Tworzenie wizualizacji funkcji wykresu chi2-test

Przykład: Próba 1

Wykonaj następujące czynności:

1. W Edytorze ładowania danych kliknij w celu przejścia do widoku aplikacji, a następnie kliknij arkusz utworzony poprzednio.
   Zostanie otwarty widok arkusza.
2. Kliknij **Edytuj arkusz**, aby rozpocząć edycję arkusza.
3. Z obszaru **Wykresy** dodaj tabelę, a z obszaru **Pola** dodaj Grp, Grade i Count jako wymiary.
   W tej tabeli znajdują się dane z próby.
4. Dodaj kolejną tabelę z następującym wyrażeniem jako wymiarem:
   `ValueList('p','df','Chi2')`
   Wyrażenie to korzysta z funkcji wymiarów syntetycznych w celu utworzenia etykiet dla wymiarów z nazwami trzech funkcjichi2-test.
5. Dodaj w tabeli następujące wyrażenie jako miarę:
   ```
   IF(ValueList('p','df','Chi2')='p',Chi2Test_p(Grp,Grade,Count),
   IF(ValueList('p','df','Chi2')='df',Chi2Test_df(Grp,Grade,Count),
   Chi2Test_Chi2(Grp,Grade,Count)))
   ```
   Wyrażenie to sprawi, że wartości wyników każdej funkcji chi2-test zostaną wstawione w tabeli obok powiązanego z nimi wymiaru syntetycznego.
6. Ustaw **Formatowanie liczb** miary na wartość **Liczba** oraz **3Cyfry znaczące**.

W wyrażeniu dla miary możesz użyć w zamian następującego wyrażenia: `Pick(Match(ValueList('p','df','Chi2'),'p','df','Chi2'),Chi2Test_p(Grp,Grade,Count),Chi2Test_df(Grp,Grade,Count),Chi2Test_Chi2(Grp,Grade,Count))`

Wynik:

Tabela docelowa dla funkcji chi2-test dla danych z próby 1 będzie zawierać następujące wartości:

Tabela wynikowa

p	df	Chi2
0.820	5	2.21

Przykład: Próba 2

Wykonaj następujące czynności:

1. W arkuszu edytowanym w przykładzie Próba 1 dodaj tabelę z obszaru **Wykresy**, a z obszaru **Pola** dodaj Sex, Opinion i OpCount jako wymiary.
2. Utwórz kopię tabeli wyników z Próby 1, korzystając z poleceń **Kopiuj** i **Wklej**. Edytuj wyrażenie w mierze i zastąp argumenty we wszystkich trzech funkcjach chi2-test nazwami pól użytych w danych z próby 2, na przykład: `Chi2Test_p(Sex,Opinion,OpCount)`.

Wynik:

Tabela docelowa dla funkcji chi2-test dla danych z próby 2 będzie zawierać następujące wartości:

Tabela wynikowa

p	df	Chi2
0.000309	2	16.2

Przykład: Próba 3

Wykonaj następujące czynności:

1. Utwórz jeszcze dwie tabele w ten sam sposób, jak w przykładach dotyczących danych z próby 1 i próby 2. W tabeli wymiarów użyj następujących pól jako wymiarów: Gender, Description, Actual oraz Expected.
2. W docelowej tabeli zastosuj nazwy pól użytych w danych z próby 3, na przykład: `Chi2Test_p (Gender,Description,Actual,Expected)`.

Wynik:

Tabela docelowa dla funkcji chi2-test dla danych z próby 3 będzie zawierać następujące wartości:

Tabela wynikowa

p	df	Chi2
0.000308	2	16.2

Przykłady stosowania funkcji chi2-test w skrypcie ładowania danych

Funkcje chi2-test są stosowane w celu znajdowania wartości powiązanych z analizą statystyczną chi-kwadrat. W tej sekcji opisano, jak korzystać z funkcji testu zgodności chi-

kwadrat dostępnych w aplikacji Qlik Sense w skrypcie ładowania danych. Więcej informacji o składni i argumentach dotyczących funkcji skryptu chi2-test można znaleźć w stosownych opisach.

W tym przykładzie wykorzystywana jest tabela zawierająca liczbę studentów uzyskujących ocenę (A-F) w dwóch grupach (I i II).

Data table

Group	A	B	C	D	E	F
I	15	7	9	20	26	19
II	10	11	7	15	21	16

Ładowanie danych z próby

Wykonaj następujące czynności:

1. Utwórz nową aplikację.
2. Otwórz edytor ładowania danych i wprowadź:
```
// Sample_1 data is pre-aggregated... Note: make sure you set your DecimalSep='.' at the top of the script.
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. Kliknij aby załadować dane.

Dane z próby zostały załadowane.

Ładowanie wartości funkcji chi2-test

Teraz załadujemy wartości chi2-test na podstawie danych z próby w nową tabelę, pogrupowaną według wartości Grp.

Wykonaj następujące czynności:

1. W edytor ładowania danych dodaj następujące informacje na końcu skryptu:
```
// 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. Kliknij aby załadować dane.

Wartości chi2-test zostały załadowane do tabeli o nazwie Chi2_table.

Wyniki

Otrzymane wartości chi2-test można wyświetlić w przeglądarce modelu danych w obszarze **Podgląd**. Powinny one wyglądać następująco:

Results

Grp	chi2	df	p
I	16.00	5	0.007
II	9.40	5	0.094

Tworzenie typowego raportu t-test

Typowy raport dotyczący testu t-test Studenta może obejmować tabele z wynikami dla grup **Group Statistics** i **Independent Samples Test**.

W następujących sekcjach takie tabele zostaną utworzone za pomocą funkcji Qlik Senset-test zastosowanych w odniesieniu do dwóch niezależnych grup prób: Observation i Comparison. Odpowiednie tabele dla tych prób wyglądałyby następująco:

Statystyki grup

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

Test niezależnych prób

Type	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval of the Difference (Lower)	95% Confidence Interval of the Difference (Upper)
Equal Variance not Assumed	3.534	37.116717335823	0.001	15.2	4.30101	6.48625	23.9137

Type	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval of the Difference (Lower)	95% Confidence Interval of the Difference (Upper)
Equal Variance Assumed	3.534	38	0.001	15.2	4.30101	6.49306	23.9069

Ładowanie danych z próby

Wykonaj następujące czynności:

1. Należy utworzyć aplikację przy użyciu nowego arkusza i otworzyć ten arkusz.
2. W edytorze ładowania danych wprowadź:
```
Table1:
crosstable 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 '|');
```
W tym skrypcie ładowania uwzględniona jest funkcja**recno()**, ponieważ tabela**crosstable** wymaga trzech argumentów. Funkcja **recno()** zapewnia zatem dodatkowy argument (w tym przypadku ID dla każdego wiersza). Bez tego argumentu przykładowe wartości **Comparison** nie zostałyby załadowane.
3. Kliknij aby załadować dane.

Tworzenie tabeli Group Statistics

Wykonaj następujące czynności:

1. W Edytorze ładowania danych kliknij w celu przejścia do widoku aplikacji, a następnie kliknij arkusz utworzony poprzednio.
To spowoduje otwarcie widoku arkusza.

2. Kliknij **Edytuj arkusz**, aby rozpocząć edycję arkusza.
3. Z obszaru **Wykresy** dodaj tabelę, a z obszaru **Pola** dodaj następujące wyrażenia jako miary:

Przykłady wyrażeń

Etykieta	Wyrażenie
N	Count(Value)
Mean	Avg(Value)
Standard Deviation	Stdev(Value)
Standard Error Mean	Sterr(Value)

4. Dodaj Type jako wymiar do tabeli.
5. Kliknij opcję **Sortowanie** i przesuń Type na szczyt listy sortowania.

Wynik:

Tabela Group Statistics dla tych prób wyglądałaby następująco:

Statystyki grup

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

Tworzenie tabeli Two Independent Sample Student's T-test

Wykonaj następujące czynności:

1. Kliknij **Edytuj arkusz**, aby rozpocząć edycję arkusza.
2. Dodaj do tabeli następujące wyrażenie jako wymiar. `=ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1))`

3. Z obszaru **Wykresy** dodaj tabelę z następującymi wyrażeniami jako miarami:

Przykłady wyrażeń

Etykieta	Wyrażenie
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 of the Difference (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 of the Difference (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))

Wynik:

Test niezależnych prób

Type	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval of the Difference (Lower)	95% Confidence Interval of the Difference (Upper)
Equal Variance not Assumed	3.534	37.116717335823	0.001	15.2	4.30101	6.48625	23.9137
Equal Variance Assumed	3.534	38	0.001	15.2	4.30101	6.49306	23.9069

Przykłady stosowania funkcji z-test

Funkcje z-test są stosowane w celu znajdowania wartości powiązanych z analizą statystyczną z-test dla dużych prób (powyżej 30 obserwacji) i przy znanej wariancji.

W tej sekcji opisano sposoby tworzenia wizualizacji na podstawie danych z próby w celu znalezienia wartości funkcji z-test dostępnych w Qlik Sense. Więcej informacji o składni i argumentach dotyczących funkcji wykresów z-test można znaleźć w stosownych opisach.

Ładowanie danych z próby

Użyte tutaj dane z próby są takie same jak dane użyte w przykładach dotyczących funkcji t-test. Rozmiar próby byłby w zwykłych okolicznościach uznawany za zbyt mały dla testu z, ale jest wystarczające na potrzeby zilustrowania sposobu korzystania z różnych funkcji z-test w Qlik Sense.

Wykonaj następujące czynności:

1. Należy utworzyć aplikację przy użyciu nowego arkusza i otworzyć ten arkusz.

   Jeśli utworzono aplikację dla funkcji t-test można na jej podstawie utworzyć nowy arkusz dla tych funkcji.

2. Otwórz edytor ładowania danych i wprowadź:

```
Table1:
crosstable 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 '|');
```

   W tym skrypcie ładowania uwzględniona jest funkcja**recno()**, ponieważ tabela**crosstable** wymaga trzech argumentów. Funkcja **recno()** zapewnia zatem dodatkowy argument (w tym przypadku ID dla każdego wiersza). Bez tego argumentu przykładowe wartości **Comparison** nie zostałyby załadowane.

3. Kliknij aby załadować dane.

Tworzenie wizualizacji funkcji wykresu z-test

Wykonaj następujące czynności:

1. W Edytorze ładowania danych kliknij w celu przejścia do widoku aplikacji, a następnie kliknij arkusz utworzony podczas ładowania danych.
   Zostanie otwarty widok arkusza.
2. Kliknij **Edytuj arkusz**, aby rozpocząć edycję arkusza.
3. Z obszaru **Wykresy** dodaj tabelę, a z obszaru **Pola** dodaj Type jako wymiar.
4. Dodaj do tabeli następujące wyrażenia jako miary.

Przykłady wyrażeń

Etykieta	Wyrażenie
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)

*Możesz zmienić sposób formatowania liczb z miar w celu uzyskania bardziej czytelnych wyników. Tabelę będzie łatwiej odczytać, jeśli w większości miar formatowanie liczb zostanie ustawione na **Liczba>Proste**, zamiast na **Auto**. Ale na przykład dla funkcji ZTest Sig należy używać formatowania liczb: **Niestandardowe**, a następnie dostosować wzorzec formatu do **# ##**.*

Wynik:

Tabela docelowa dla funkcji z-test dla danych z próby będzie zawierać następujące wartości:

Tabela wynikowa

Type	ZTest Conf	ZTest Dif	ZTest Sig	ZTest Sterr	ZTest Z
Comparison	6.40	11.95	0.000123	3.27	3.66
Value	5.48	27.15	0.001	2.80	9.71

Tworzenie wizualizacji funkcji wykresu z-testw

Funkcje z-testw są stosowane w przypadku, gdy szeregi danych wejściowych są podane w formacie ważonym dwukolumnowym. Wyrażenia takie wymagają podania wartości argumentu weight. W poniższych przykładach użyto wszędzie wartości 2, ale można w tym celu użyć wyrażenia, które określi wartość argumentu weight dla każdej obserwacji.

Przykłady i wyniki:

W przypadku zastosowania tych samych danych z próby i formatów liczb co w funkcjach z-test tabela docelowa dla funkcji z-testw będzie zawierać następujące wartości:

Tabela wynikowa

Type	ZTestw Conf	ZTestw Dif	ZTestw Sig	ZTestw Sterr	ZTestw Z
Comparison	3.53	2.95	5.27e-005	1.80	3.88
Value	2.97	34.25	0	4.52	20.49

Funkcje agregacji dla ciągów znaków

W tej sekcji opisano funkcje agregacji powiązane z ciągami znaków.

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

Funkcje agregacji dla ciągów w skrypcie ładowania danych

Concat

Funkcja **Concat()** służy do łączenia wartości ciągów znaków. Funkcja skryptu zwraca zagregowaną konkatenację ciągu wszystkich wartości wyrażenia iterowanego po liczbie rekordów zgodnie z definicją w klauzuli **group by**.

```
Concat ([ distinct ] expression [, delimiter [, sort-weight]])
```

FirstValue

Funkcja **FirstValue()** zwraca wartość załadowaną jako pierwsza z rekordów określonych przez wyrażenie, posortowanych przez klauzulę **group by**.

Ta funkcja jest dostępna tylko jako funkcja skryptu.

```
FirstValue (expression)
```

LastValue

Funkcja **LastValue()** zwraca wartość załadowaną jako ostatnia z rekordów określonych przez wyrażenie, posortowanych przez klauzulę **group by**.

Ta funkcja jest dostępna tylko jako funkcja skryptu.

```
LastValue (expression)
```

MaxString

FunkcjaMaxString() wyszukuje wartości ciągu w wyrażeniu i zwraca ostatnią wartość tekstową posortowaną w porządku alfabetycznym według liczby rekordów zgodnie z definicją w klauzuli group by.

```
MaxString (expression )
```

MinString

FunkcjaMinString() wyszukuje wartości ciągu w wyrażeniu i zwraca ostatnią wartość tekstową posortowaną w porządku alfabetycznym według liczby rekordów zgodnie z definicją w klauzuli group by.

```
MinString (expression )
```

Funkcje agregacji dla ciągów znaków w wykresach

Poniższe funkcje wykresów są dostępne w odniesieniu do agregowania ciągów znaków w wykresach.

Concat

Funkcja Concat() służy do łączenia wartości ciągów znaków. Funkcja ta zwraca zagregowaną konkatenację ciągów znaków ze wszystkich wartości wyrażenia obliczonych dla poszczególnych wymiarów.

```
Concat — funkcja wykresu({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]
string[, delimiter[, sort_weight]])
```

MaxString

Funkcja MaxString() znajduje wartości łańcuchowe w wyrażeniu lub polu i zwraca ostatnią wartość tekstową w porządku alfabetycznym.

```
MaxString — funkcja wykresu({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MinString

Funkcja MinString() znajduje wartości łańcuchowe w wyrażeniu lub polu i zwraca ostatnią wartość tekstową w porządku alfabetycznym.

```
MinString — funkcja wykresu({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

Concat

Funkcja Concat() służy do łączenia wartości ciągów znaków. Funkcja skryptu zwraca zagregowaną konkatenację ciągu wszystkich wartości wyrażenia iterowanego po liczbie rekordów zgodnie z definicją w klauzuli group by.

Składnia:

```
Concat ([ distinct ] string [, delimiter [, sort-weight]])
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Wyrażenie lub pole zawierające ciąg znaków do przetworzenia.

Argumenty

Argument	Opis
string	Wyrażenie lub pole zawierające ciąg znaków do przetworzenia.
delimiter	Wartości mogą być rozdzielane ciągiem znaków określonym argumentem delimiter.
sort-weight	Kolejność konkatenacji można określić opcjonalnym argumentem wymiaru **sort-weight**. Ciąg znaków odpowiadający najniższej wartości sortowania będzie występować na początku konkatenacji..
distinct	Jeśli przed wyrażeniem występuje słowo **distinct**, wówczas wszystkie duplikaty są pomijane.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

Przykłady i wyniki

Przykład	Wynik	Wyniki po dodaniu do arkusza

`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

Zakładając, że tabela **TeamData** została załadowana jak w poprzednim przykładzie:
 	
`LOAD SalesGroup,Concat(distinct Team,'-')`
`as TeamConcat2 Resident TeamData Group By`
`SalesGroup;`
	SalesGroup
	East
	West
		TeamConcat2
		Alpha-Beta-Delta-Gamma
		Epsilon-Eta-Theta-Zeta

Przykład	Wynik	Wyniki po dodaniu do arkusza
Zakładając, że tabela **TeamData** została załadowana jak w poprzednim przykładzie. Dodano argument **sort-weight**, wyniki są zatem porządkowane według wartości wymiaru Amount: `LOAD SalesGroup,Concat(distinct Team,'-',Amount) as TeamConcat2 Resident TeamData Group By SalesGroup;`	SalesGroup East West	TeamConcat2 Delta-Beta-Gamma-Alpha Eta-Epsilon-Zeta-Theta

Concat – funkcja wykresu

Funkcja **Concat()** służy do łączenia wartości ciągów znaków. Funkcja ta zwraca zagregowaną konkatenację ciągów znaków ze wszystkich wartości wyrażenia obliczonych dla poszczególnych wymiarów.

Składnia:

```
Concat({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} string[, delimiter [, sort_weight]])
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
string	Wyrażenie lub pole zawierające ciąg znaków do przetworzenia.
delimiter	Wartości mogą być rozdzielane ciągiem znaków określonym argumentem delimiter.
sort-weight	Kolejność konkatenacji można określić opcjonalnym argumentem wymiaru **sort-weight**. Ciąg znaków odpowiadający najniższej wartości sortowania będzie występować na początku konkatenacji..
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli przed argumentami funkcji występuje słowo **DISTINCT**, wówczas duplikaty wynikające z wyników obliczenia argumentów funkcji są pomijane.
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Przykłady i wyniki:

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

Przykłady funkcji

Przykład	Wynik
`Concat(Team)`	Tabela została utworzona na podstawie wymiarów SalesGroup i Amountoraz odchyleń od miary Concat(Team). Ignorując wyniki Totals, należy zauważyć, że nawet jeśli dostępne są dane dla ośmiu wartości w kolumnie Team podzielonych względem dwóch wartości w kolumnie SalesGroup, jedyny wynik miary Concat (Team), który dokonuje konkatenacji więcej niż jednej wartości ciągu Team w tej tabeli, jest w wierszu zawierającym wymiar Amount o wartości 20000, co zwraca wynik BetaGammaGamma. Dzieje się tak, ponieważ występują trzy wartości dla wymiaru Amount 20000 w danych wejściowych. Wszystkie pozostałe wyniki nie zostają poddane konkatenacji, gdy miara obejmuje wiele wymiarów, ponieważ jest tylko jedna wartość Team dla każdej kombinacji wymiarów SalesGroup i Amount.
`Concat (DISTINCT Team,', ')`	Beta, Gamma, ponieważ kwalifikator DISTINCT sprawia, że powtarzający się wynik Gamma zostaje zignorowany. Ponadto argument ogranicznika jest zdefiniowany jako przecinek, po którym następuje spacja.
`Concat (TOTAL <SalesGroup> Team)`	Wszystkie wartości ciągów dla wszystkich wartości Team są poddane konkatenacji, jeśli zastosowany został kwalifikator TOTAL. W przypadku wyboru pola <SalesGroup> wyniki zostają podzielone między dwie wartości wymiaru SalesGroup. W przypadku SalesGroupEast zwracane są wyniki AlphaBetaDeltaGammaGamma. W przypadku SalesGroupWest zwracane są wyniki EpsilonEtaThetaZeta.
`Concat (TOTAL <SalesGroup> Team,';', Amount)`	Dodanie argumentu **sort-weight**: Amount powoduje, że wyniki są uporządkowane według wartości wymiaru Amount. Zwracane są zatem wyniki DeltaBetaGammaGammaAlpha i EtaEpsilonZEtaTheta.

Dane zastosowane w przykładzie:

```
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

Funkcja **FirstValue()** zwraca wartość załadowaną jako pierwsza z rekordów określonych przez wyrażenie, posortowanych przez klauzulę **group by**.

Ta funkcja jest dostępna tylko jako funkcja skryptu.

Składnia:

```
FirstValue ( expr)
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opisu
expr	Wyrażenie lub pole zawierające mierzone dane.

Ograniczenia:

Jeśli nie zostanie znaleziona żadna wartość tekstowa, zwracana jest wartość NULL.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

Dane wynikowe

Przykład	Wynik	Wyniki na arkuszu
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

Funkcja **LastValue()** zwraca wartość załadowaną jako ostatnia z rekordów określonych przez wyrażenie, posortowanych przez klauzulę **group by**.

Ta funkcja jest dostępna tylko jako funkcja skryptu.

Składnia:

```
LastValue ( expr )
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opisu
expr	Wyrażenie lub pole zawierające mierzone dane.

Ograniczenia:

Jeśli nie zostanie znaleziona żadna wartość tekstowa, zwracana jest wartość NULL.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Następnie dodaj do arkusza w swojej aplikacji co najmniej pola wyszczególnione w kolumnie wyników, aby wyświetlić wynik.

Aby uzyskać ten sam wygląd, jak w kolumnie wyników poniżej, w panelu właściwości w obszarze sortowania przełącz z wartości Autom. na Niestandardowe. Następnie usuń zaznaczenie sortowania liczbowego i alfabetycznego.

Przykład	Wynik	Wynik z niestandardowym sortowaniem

`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

Funkcja**MaxString()** wyszukuje wartości ciągu w wyrażeniu i zwraca ostatnią wartość tekstową posortowaną w porządku alfabetycznym według liczby rekordów zgodnie z definicją w klauzuli **group by**.

Składnia:

```
MaxString ( expr )
```

Typ zwracanych danych: podwójny

Argumenty:

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.

Ograniczenia:

Jeśli nie zostanie znaleziona żadna wartość tekstowa, zwracana jest wartość NULL.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

Przykład	Wynik	
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
Zakładając, że tabela **TeamData** została załadowana jak w poprzednim przykładzie, a skrypt ładowania danych zawiera instrukcję SET: SET DateFormat='DD/MM/YYYY';: LOAD SalesGroup,MaxString(Date) as MaxString2 Resident TeamData Group By SalesGroup;	SalesGroup East West	MaxString2 01/11/2013 01/12/2013

MaxString – funkcja wykresu

Funkcja **MaxString()** znajduje wartości łańcuchowe w wyrażeniu lub polu i zwraca ostatnią wartość tekstową w porządku alfabetycznym.

Składnia:

```
MaxString({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opisu
expr	Wyrażenie lub pole zawierające mierzone dane.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.

Argument	Opisu
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Ograniczenia:

Jeśli wyrażenie nie zawiera wartości z reprezentacją ciągu znaków, zwracana jest wartość NULL.

Przykłady i wyniki:

Tabela wynikowa

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

Przykłady funkcji

Przykład	Wynik
`MaxString (Team)`	Dla wymiaru Amount określone są trzy wartości 20000: dwie dotyczą zespołu Gamma (w różnych datach), a jedna – zespołu Beta. Dla miary MaxString (Team) zwracana jest zatem wartość Gamma, ponieważ jest to najwyżej znajdująca się wartość w posortowanych ciągach.
`MaxString (Date)`	2013/11/01 jest największą wartością Date z trzech wartości powiązanych z wymiarem Amount. Ponadto przyjęto założenie, że skrypt zawiera instrukcję SET `SET DateFormat='YYYY-MM-DD';'`

Dane zastosowane w przykładzie:

```
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

FunkcjaMinString() wyszukuje wartości ciągu w wyrażeniu i zwraca ostatnią wartość tekstową posortowaną w porządku alfabetycznym według liczby rekordów zgodnie z definicją w klauzuli group by.

Składnia:

```
MinString ( expr )
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opisu
expr	Wyrażenie lub pole zawierające mierzone dane.

Ograniczenia:

Jeśli nie zostanie znaleziona żadna wartość tekstowa, zwracana jest wartość NULL.

Przykłady i wyniki:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

Dane wynikowe

Przykład	Wynik	
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
Zakładając, że tabela **TeamData** została załadowana jak w poprzednim przykładzie, a skrypt ładowania danych zawiera instrukcję SET: SET DateFormat='DD/MM/YYYY';: LOAD SalesGroup,MinString(Date) as MinString2 Resident TeamData Group By SalesGroup;	SalesGroup East West	MinString2 01/05/2013 01/06/2013

MinString – funkcja wykresu

Funkcja **MinString()** znajduje wartości łańcuchowe w wyrażeniu lub polu i zwraca ostatnią wartość tekstową w porządku alfabetycznym.

Składnia:

```
MinString({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.

Argument	Opis
TOTAL	Jeśli słowo **TOTAL** występuje przed argumentami funkcji, wówczas obliczenie jest wykonywane względem wszystkich możliwych wartości z uwzględnieniem bieżących selekcji, nie tylko tych należących do bieżącej wartości wymiaru, to znaczy z pominięciem wymiarów wykresu. Korzystając z polecenia **TOTAL [<fld {.fld}>]**, gdzie po kwalifikatorze **TOTAL** podana jest lista nazw pól stanowiących podzbiór zmiennych wymiarów wykresu, można utworzyć podzbiór wszystkich możliwych wartości.

Przykłady i wyniki:

Dane przykładowe

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

Przykłady funkcji

Przykłady	Wyniki
`MinString (Team)`	Dla wymiaru Amount określone są trzy wartości 20000: dwie dotyczą zespołu Gamma (w różnych datach), a jedna – zespołu Beta. Dla miary MinString (Team) zwracana jest zatem wartość Beta, ponieważ jest to pierwsza wartość w posortowanych ciągach.
`MinString (Date)`	2013/11/01 jest najwcześniejszą wartością Date z trzech wartości powiązanych z wymiarem Amount. Ponadto przyjęto założenie, że skrypt zawiera instrukcję SET `SET DateFormat='YYYY-MM-DD';'`

Dane zastosowane w przykładzie:

```
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 '|');
```

Funkcje wymiarów syntetycznych

Wymiar syntetyczny jest tworzony w aplikacji na podstawie wartości generowanych przez funkcje wymiarów syntetycznych, a nie bezpośrednio na podstawie pól w modelu danych. Gdy wartości wygenerowane przez funkcję wymiarów syntetycznych zostaną użyte na wykresie jako wymiar wyliczany, powstaje wymiar syntetyczny. Wymiary syntetyczne pozwalają na tworzenie m.in. wykresów z wymiarami z wartościami wynikającymi z podanych danych, czyli wymiarami dynamicznymi.

Selekcje nie mają wpływu na wymiary syntetyczne.

W wykresach mogą być stosowane następujące funkcje wymiarów syntetycznych.

ValueList
Funkcja **ValueList()** zwraca zestaw wymienionych wartości, które w przypadku zastosowania w wymiarze wyliczanym utworzą wymiar syntetyczny.

```
ValueList – funkcja wykresu (v1 {, Expression})
```

ValueLoop
Funkcja ValueLoop() zwraca zestaw wartości iterowanych, które w przypadku zastosowania w wymiarze wyliczanym utworzą wymiar syntetyczny.

```
ValueLoop – funkcja wykresu(from [, to [, step ]])
```

ValueList – funkcja wykresu

Funkcja **ValueList()** zwraca zestaw wymienionych wartości, które w przypadku zastosowania w wymiarze wyliczanym utworzą wymiar syntetyczny.

*W wykresach z wymiarem syntetycznym utworzonym za pomocą funkcji **ValueList** możliwe jest przywołanie wartości wymiaru odpowiadającej konkretnej komórce wyrażenia. W tym celu należy zrestartować funkcję **ValueList** z tymi samymi parametrami w wyrażeniu wykresu. Ta funkcja może być oczywiście używana w dowolnym miejscu układu, ale oprócz sytuacji, gdy jest używana dla wymiarów syntetycznych, będzie znaczącą tylko wewnątrz funkcji agregacji.*

Selekcje nie mają wpływu na wymiary syntetyczne.

Składnia:

```
ValueList(v1 {,...})
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
v1	Wartość statyczna (zazwyczaj ciąg, ale może to być również liczba).
{,...}	Opcjonalna lista wartości statycznych.

Przykłady i wyniki:

Przykłady funkcji

Przykład	Wynik
`ValueList ('Number of Orders', 'Average Order Size', 'Total Amount')`	Gdy jest stosowane na przykład w celu utworzenia wymiaru w tabeli, wyrażenie to zwraca trzy ciągi znaków jako etykiety wierszy w tabeli. Ciągi te mogą być następnie przywoływane przez dowolne wyrażenie.
`=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) ))`	Wyrażenie to pobiera wartości z utworzonego wymiaru i przywołuje je w zagnieżdżonej instrukcji IF jako dane wejściowe dla trzech funkcji agregacji: **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

Dane zastosowane w przykładach:

```
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 – funkcja wykresu

Funkcja ValueLoop() zwraca zestaw wartości iterowanych, które w przypadku zastosowania w wymiarze wyliczanym utworzą wymiar syntetyczny.
Wygenerowane wartości będą rozpoczynać się od wartości **from** i kończyć na wartości **to** wraz z wartościami pośrednimi zgodnymi z przyrostem kroku.

*W wykresach z wymiarem syntetycznym utworzonym za pomocą funkcji **ValueLoop** możliwe jest przywołanie wartości wymiaru odpowiadającej konkretnej komórce wyrażenia. W tym celu należy zrestartować funkcję **ValueLoop** z tymi samymi parametrami w wyrażeniu wykresu. Ta funkcja może być oczywiście używana w dowolnym miejscu układu, ale oprócz sytuacji, gdy jest używana dla wymiarów syntetycznych, będzie znaczącą tylko wewnątrz funkcji agregacji.*

Selekcje nie mają wpływu na wymiary syntetyczne.

Składnia:

```
ValueLoop(from [, to [, step ]])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argumenty	Opis
from	Wartość początkowa w generowanym zakresie wartości.
to	Wartość końcowa w generowanym zakresie wartości.
step	Rozmiar kroku między wartościami.

Przykłady i wyniki:

Przykłady funkcji

Przykład	Wynik
`ValueLoop (1, 10)`	Tworzy w tabeli wymiar, który może być zastosowany w celach takich jak tworzenie numerowanych etykiet. W tym przykładzie zwrócone zostaną wartości ponumerowane od 1 do 10. Wartości te mogą być następnie przywoływane przez dowolne wyrażenie.
`ValueLoop (2, 10,2)`	W tym przykładzie zwracane są wartości ponumerowane cyframi 2, 4, 6, 8 i 10, ponieważ argument step ma wartość 2.

Agregacje zagnieżdżone

Mogą pojawić się sytuacje, w których należy zastosować agregację w odniesieniu do wyników innej agregacji. Sytuacja taka jest określana mianem „agregacji zagnieżdżonej”.

W większości wyrażeń wykresu nie można zagnieżdżać agregacji. Można jednak zagnieżdżać agregacje, jeśli używa się kwalifikatora **TOTAL** w wewnętrznej funkcji agregacji.

Dozwolonych jest maksymalnie sto poziomów zagnieżdżenia.

Agregacje zagnieżdżone z kwalifikatorem TOTAL

Przykład:

Należy na przykład obliczyć sumę wartości pól **Sales**, ale tylko z uwzględnieniem transakcji z wartością **OrderDate** z ubiegłego roku. Wartość taką można uzyskać dzięki zastosowaniu funkcji agregacji **`Max (TOTAL Year(OrderDate))`**.

Następująca agregacja zwróci oczekiwany wynik:

```
Sum(If(Year(OrderDate)=Max(TOTAL Year(OrderDate)), Sales))
```

Qlik Sense wymaga włączenia kwalifikatora **TOTAL** w tego typu zagnieżdżeniach. Jest to konieczne do potrzebnego porównania. Tego typu zagnieżdżenie jest często stosowane i bardzo przydatne.

Zob. także:

p *Aggr – funkcja wykresu (page 532)*

5.3 Aggr – funkcja wykresu

Funkcja **Aggr()** zwraca tablicę wartości wyrażenia obliczonego po wskazanych wymiarach. Może to na przykład być wartość maksymalna sprzedaży według klienta i regionu.

Funkcja **Aggr** jest przeznaczona do agregacji zagnieżdżonych, w których pierwszy parametr (agregacja wewnętrzna) jest obliczany raz na każdą wartość wymiaru. Wymiary są określone w drugim i kolejnych parametrach.

Ponadto funkcja **Aggr** powinna być zamknięta w zewnętrznej funkcji agregacji wykorzystującej tablicę wyników z funkcji **Aggr** jako dane wejściowe do agregacji, w której jest zagnieżdżona.

Składnia:

```
Aggr({SetExpression}[DISTINCT] [NODISTINCT ] expr, StructuredParameter{,
StructuredParameter})
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie zawierające funkcję agregacji. Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję.
StructuredParameter	Argument StructuredParameter składa się z wymiaru i opcjonalnie z kryterium sortowania w następującym formacie: `(Dimension(Sort-type, Ordering))` Wymiar musi być pojedynczym polem i nie może być wyrażeniem. Na podstawie wymiaru zostanie ustalona tablica wartości, z których będzie wyliczana wartość wyrażenia Aggr. Jeśli podane są kryteria sortowania, obliczona dla wymiaru tablica wartości utworzona przez funkcję Aggr jest sortowana. Jest to istotne, jeśli kolejność sortowania wpływa na wynik wyrażenia zawierającego funkcję Aggr. Szczegółowe informacje na temat używania kryteriów sortowania zawiera temat Dodawanie kryteriów sortowania do wymiaru podanego argumentem StructuredParameter.
SetExpression	Funkcja agregacji będzie domyślnie dokonywać agregacji na zbiorze możliwych wierszy zdefiniowanym przez selekcję. Alternatywny zestaw wierszy można zdefiniować za pomocą wyrażenia analizy zestawów.
DISTINCT	Jeśli argument wyrażenia jest poprzedzony kwalifikatorem **distinct** lub w ogóle nie ma kwalifikatora, dla każdej odrębnej kombinacji wartości wymiarów będzie generowana tylko jedna wartość zwracana. W ten sposób są standardowo dokonywane agregacje. Każda odrębna kombinacja wartości wymiarów będzie generować jeden wiersz na wykresie.
NODISTINCT	Jeśli argument wyrażenia jest poprzedzony kwalifikatorem **nodistinct**, każda kombinacja wartości wymiarów może generować więcej niż jedną wartość zwracaną (w zależności od bazowej struktury danych). Jeśli występuje tylko jeden wymiar, funkcja **aggr** zwróci tablicę z tą samą liczbą elementów co liczba wierszy w źródle danych.

Podstawowe funkcje agregacji, takie jak **Sum**, **Min** i **Avg**, zwracają jedną wartość liczbową, natomiast działanie funkcji Aggr() można porównać do utworzenia tymczasowego zestawu wyników (tabeli wirtualnej), na którym wykonywana jest kolejna agregacja. Przykładowe zastosowanie to obliczenie średniej wartości sprzedaży poprzez zsumowanie sprzedaży według klientów w ramach instrukcji **Aggr()**, a następnie obliczenie średniej zsumowanych wyników: **Avg(TOTAL Aggr(Sum(Sales),Customer))**.

Funkcji Aggr() można używać w wymiarach wyliczanych, do tworzenia zagnieżdżanych agregacji wykresów na wielu poziomach.

Ograniczenia:

Każdy wymiar w funkcji Aggr() musi być pojedynczym polem i nie może być wyrażeniem (wymiarem wyliczanym).

Dodawanie kryteriów sortowania do wymiaru podanego argumentem StructuredParameter

W podstawowej postaci argument StructuredParameter w składni funkcji Aggr jest pojedynczym wymiarem. Wyrażenie Aggr(Sum(Sales, Month)) znajduje łączną wartość sprzedaży w każdym miesiącu. Użycie go w obrębie innej funkcji agregacji może jednak dać nieoczekiwane wyniki, jeśli nie zostaną określone kryteria sortowania. Wynika to stąd, że niektóre wymiary można sortować na różne sposoby (w kolejności liczbowej, alfabetycznie itd.).

Argument StructuredParameter funkcji Aggr umożliwia określenie kryteriów sortowania wymiaru w ramach wyrażenia. Pozwala to wymusić określoną kolejność sortowania tabeli wirtualnej generowanej przez funkcję Aggr.

Składnia argumentu StructuredParameter jest następująca:

```
(FieldName, (Sort-type, Ordering))
```

Argumenty StructuredParameter można zagnieżdżać:

```
(FieldName, (FieldName2, (Sort-type, Ordering)))
```

Typem sortowania może być: NUMERIC, TEXT, FREQUENCY lub LOAD_ORDER.

Z poszczególnymi typami sortowania powiązane są następujące kolejności:

Dozwolone typy określania kolejności

Typ sortowania	Dozwolone kolejności
NUMERIC	ASCENDING, DESCENDING lub REVERSE
TEXT	ASCENDING, A2Z, DESCENDING, REVERSE lub Z2A
FREQUENCY	DESCENDING, REVERSE lub ASCENDING
LOAD_ORDER	ASCENDING, ORIGINAL, DESCENDING lub REVERSE

Kolejności REVERSE i DESCENDING są równoważne.

W przypadku typu sortowania TEXT kolejności ASCENDING i A2Z są sobie równoważne. Również kolejności DESCENDING, REVERSE i Z2A są sobie równoważne.

W przypadku typu sortowania LOAD_ORDER kolejności ASCENDING i ORIGINAL są sobie równoważne.

Przykłady: Wyrażenia wykresu używające funkcji Aggr

Przykłady – wyrażenia wykresu

Przykład wyrażenia wykresu 1

Skrypt ładowania

Załaduj następujące dane w edytorze ładowania danych jako ładowanie wbudowane, aby utworzyć poniższy przykład wyrażeń wykresu.

```
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 '|');
```

Wyrażenie wykresu

Utwórz wizualizację KPI w arkuszu Qlik Sense. Dodaj do KPI następujące wyrażenie jako miarę:

```
Avg(Aggr(Sum(UnitSales*UnitPrice), Customer))
```

Wynik

376.7

Objaśnienie

Wyrażenie `Aggr(Sum(UnitSales*UnitPrice), Customer)` znajduje łączną wartość sprzedaży według pola **Customer** i zwraca tablicę wartości: 295, 715 i 120 dla trzech wartości **Customer**.

W efekcie uzyskaliśmy tymczasową listę wartości bez konieczności jawnego tworzenia tabeli lub kolumny zawierającej te wartości.

Wartości te są używane jako dane wejściowe dla funkcji **Avg()**, aby znaleźć średnią wartość sprzedaży, wynoszącą 376.7.

Przykład wyrażenia wykresu 2

Skrypt ładowania

Załaduj następujące dane w edytorze ładowania danych jako ładowanie wbudowane, aby utworzyć poniższy przykład wyrażeń wykresu.

```
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 '|');
```

Wyrażenie wykresu

Utwórz wizualizację tabeli w arkuszu Qlik Sense z wymiarami **Customer**, **Product**, **UnitPrice** i **UnitSales**. Dodaj w tabeli następujące wyrażenie jako miarę:

```
Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
```

Wynik

Customer	Product	UnitPrice	UnitSales	Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
Astrida	AA	15	10	16
Astrida	AA	16	4	16
Astrida	BB	9	9	15
Astrida	BB	15	10	15
Betacab	BB	10	5	12
Betacab	BB	12	7	12
Betacab	CC	20	4	22
Betacab	CC	22	2	22
Betacab	DD	25	25	25
Canutility	AA	11	5	15
Canutility	AA	15	8	15
Canutility	CC	19	0	19

Objaśnienie

Tablica wartości: 16, 16, 15, 15, 12, 12, 22, 22, 25, 15, 15 i 19. Kwalifikator **nodistinct** oznacza, że tablica zawiera jeden element dla każdego wiersza w danych źródłowych – w tym przypadku będzie to maksymalna wartość pola **UnitPrice** dla każdego pola **Customer** i **Product**.

Przykład wyrażenia wykresu 3

Skrypt ładowania

Załaduj następujące dane w edytorze ładowania danych jako ładowanie wbudowane, aby utworzyć poniższy przykład wyrażeń wykresu.

```
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;
```

Wyrażenia wykresu

Utwórz wizualizację tabeli w arkuszu Qlik Sense z wymiarami **Year** i **Month**. Dodaj do tabeli następujące wyrażenia jako miary.

- `Sum(Sales)`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))` oznaczone jako Structured Aggr() w tabeli.

Wynik

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

Objaśnienie

W tym przykładzie wyświetlane są zagregowane wartości w okresie dwunastu miesięcy każdego roku w porządku chronologicznym rosnącym, stąd parametry strukturalne (Numeric, Ascending) wyrażenia **Aggr ()**. Jako parametry strukturalne wymagane są dwa określone wymiary: **Year** i **Month**, posortowane (1) **Year** (liczbowy) i (2) **Month** (liczbowy). Te dwa wymiary muszą być używane w wizualizacji tabeli lub wykresu. Jest to konieczne, aby lista wymiarów funkcji **Aggr()** odpowiadała wymiarom obiektu użytego w wizualizacji.

Różnicę między tymi miarami można porównać w tabeli albo na osobnych wykresach liniowych:

- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year), (Month) ))`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))`

Powinno być jasne, że tylko to drugie wyrażenie wykonuje pożądaną akumulację zagregowanych wartości.

Zob. także:

p *Podstawowe funkcje agregacji (page 316)*

5.4 Funkcje koloru

Funkcje te mogą być stosowane w wyrażeniach związanych z ustawieniami i oceną właściwości kolorów obiektów wykresu oraz w skryptach ładowania danych.

*Qlik Sense obsługuje funkcje koloru **Color()**, **qliktechblue** i **qliktechgray** ze względu na zgodność z poprzednimi wersjami, ale korzystanie z tych funkcji nie jest zalecane.*

ARGB

Funkcja **ARGB()** jest stosowana w wyrażeniach w celu ustawienia lub obliczenia wartości właściwości koloru obiektu wykresu, przy czym kolor jest zdefiniowany wartościami składowej czerwonej **r**, składowej zielonej **g** i składowej niebieskiej **b** oraz współczynnikiem alfa (przezroczystości) **alpha**.

```
ARGB (alpha, r, g, b)
```

HSL

Funkcja **HSL()** służy w wyrażeniach do ustawienia lub obliczenia wartości właściwości koloru obiektu wykresu, przy czym kolor jest zdefiniowany wartościami składowych **hue** (odcień), **saturation** (nasycenie) i **luminosity** (jasność) z zakresu od 0 do 1.

```
HSL  (hue, saturation, luminosity)
```

RGB

RGB() zwraca liczbę całkowitą odpowiadającą kodowi koloru określonemu przez trzy parametry: składnik czerwony r, składnik zielony g i składnik niebieski b. Te składniki muszą mieć wartości w postaci liczb całkowitych z przedziału od 0 do 255. Funkcji tej można używać w wyrażeniach do ustawiania lub oceny właściwości koloru obiektu wykresu.

```
RGB  (r, g, b)
```

Colormix1

Funkcja **Colormix1()** jest stosowana w wyrażeniach w celu zwracania reprezentacji koloru ARGB z gradientu dwóch kolorów, na podstawie wartości z zakresu od 0 do 1.

```
Colormix1 (Value , ColorZero , ColorOne)
```

Parametr Value jest liczbą rzeczywistą z zakresu od 0 do 1.

- Jeśli parametr Value wynosi 0, wówczas zwracana jest wartość ColorZero .
- Jeśli parametr Value wynosi 1, wówczas zwracana jest wartość ColorOne .
- Jeśli parametr Value jest z zakresu od 0 do 1, wówczas zwracany jest odpowiedni odcień pośredni.

Funkcja ColorZero jest poprawną reprezentacją koloru RGB dla koloru, który zostanie powiązany z dolnym końcem przedziału.

Funkcja ColorOne jest poprawną reprezentacją koloru RGB dla koloru, który zostanie powiązany z górnym końcem przedziału.

Przykład:

```
Colormix1(0.5, red(), blue())
```
zwraca:

```
ARGB(255,64,0,64) (purple)
```

Colormix2
Funkcja **Colormix2()** jest stosowana w wyrażeniach w celu zwracania reprezentacji koloru ARGB z gradientu dwóch kolorów, na podstawie wartości z zakresu od -1 do 1, z możliwością określenia koloru pośredniego dla pozycji centralnej (0).

```
Colormix2 (Value ,ColorMinusOne , ColorOne[ , ColorZero])
```
Parametr Value jest liczbą rzeczywistą z zakresu od -1 do 1.

- Jeśli parametr Value wynosi -1, wówczas zwracany jest pierwszy kolor.
- Jeśli parametr Value wynosi 1, wówczas zwracany jest drugi kolor.
- Jeśli parametr Value jest z zakresu od -1 do 1, wówczas zwracany jest odpowiedni kolor pośredni.

Funkcja ColorMinusOne jest poprawną reprezentacją koloru RGB dla koloru, który zostanie powiązany z dolnym końcem przedziału.

Funkcja ColorOne jest poprawną reprezentacją koloru RGB dla koloru, który zostanie powiązany z górnym końcem przedziału.

Funkcja ColorZero jest opcjonalną poprawną reprezentacją koloru RGB dla koloru, który zostanie powiązany ze środkiem przedziału.

SysColor
Funkcja **SysColor()** zwraca reprezentację koloru ARGB dla koloru nr w systemie Windows, gdzie nr odpowiada parametrowi dla funkcji API **GetSysColor(nr)** w systemie Windows.

```
SysColor (nr)
```

ColorMapHue
Funkcja **ColorMapHue()** zwraca wartość ARGB koloru z mapy kolorów, która zmienia składnik odcienia w modelu koloru HSV. Mapa kolorów zaczyna się od koloru czerwonego, a następnie kolejno przechodzi w kolor żółty, zielony, błękitny, niebieski, amarantowy, aby powrócić do koloru czerwonego. Parametr x musi być określony jako wartość w zakresie od 0 do 1.

```
ColorMapHue (x)
```

ColorMapJet

Funkcja **ColorMapJet()** zwraca wartość ARGB dla koloru z mapy kolorów, która zaczyna się od koloru niebieskiego, a następnie kolejno przechodzi w kolor błękitny, żółty i pomarańczowy, aby powrócić do koloru czerwonego. Parametr x musi być określony jako wartość w zakresie od 0 do 1.

```
ColorMapJet (x)
```

Wstępnie zdefiniowane funkcje koloru

W wyrażeniach dotyczących wstępnie zdefiniowanych kolorów mogą być stosowane następujące funkcje. Każda z tych funkcji zwraca reprezentację koloru RGB.

Opcjonalnie można podać parametr określający współczynnik alfa. W takim przypadku zwrócona zostanie reprezentacja koloru ARGB. Współczynnik alfa o wartości 0 oznacza pełną przezroczystość, a o wartości 255 – pełną nieprzezroczystość. Jeśli wartość alfa nie zostanie wprowadzona, przyjmowana jest wartość 255.

Wstępnie zdefiniowane funkcje koloru

Funkcje koloru	RGB value
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)

Przykłady i wyniki:

Przykłady i wyniki

Przykłady	Wyniki
`Blue()`	RGB(0,0,128)
`Blue(128)`	ARGB(128,0,0,128)

ARGB

Funkcja **ARGB()** jest stosowana w wyrażeniach w celu ustawienia lub obliczenia wartości właściwości koloru obiektu wykresu, przy czym kolor jest zdefiniowany wartościami składowej czerwonej **r**, składowej zielonej **g** i składowej niebieskiej **b** oraz współczynnikiem alfa (przezroczystości) **alpha**.

Składnia:

```
ARGB(alpha, r, g, b)
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
alpha	Wartość przezroczystości w zakresie od 0 do 255. 0 oznacza pełną przezroczystość, natomiast 255 – pełną nieprzezroczystość.
r, g, b	Wartości składowej czerwonej, zielonej i niebieskiej. Wartość składowej 0 oznacza brak wkładu, a 255 – pełny wkład.

Wszystkie argumenty muszą być wyrażeniami, które zwracają liczby całkowite w zakresie od 0 do 255.

W przypadku interpretowania składnika liczbowego i przekształcania go na zapis szesnastkowy wartości składników koloru są bardziej czytelne. Na przykład kolorowi jasnozielonemu odpowiada liczba 4 278 255 360, która w zapisie szesnastkowym ma postać FF00FF00. Pierwsze dwie pozycje „FF” (255) wskazują kanał **alpha**. Następne dwie pozycje „00” oznaczają wartość składowej czerwonej (**red**), kolejne dwie pozycje „FF” wartość składowej zielonej (**green**), a ostatnie dwie pozycje „00” wartość składowej niebieskiej (**blue**).

RGB

RGB() zwraca liczbę całkowitą odpowiadającą kodowi koloru określonemu przez trzy parametry: składnik czerwony r, składnik zielony g i składnik niebieski b. Te składniki muszą mieć wartości w postaci liczb całkowitych z przedziału od 0 do 255. Funkcji tej można używać w wyrażeniach do ustawiania lub oceny właściwości koloru obiektu wykresu.

Składnia:

```
RGB (r, g, b)
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
r, g, b	Wartości składowej czerwonej, zielonej i niebieskiej. Wartość składowej 0 oznacza brak wkładu, a 255 – pełny wkład.

Wszystkie argumenty muszą być wyrażeniami, które zwracają liczby całkowite w zakresie od 0 do 255.

W przypadku interpretowania składnika liczbowego i przekształcania go na zapis szesnastkowy wartości składników koloru są bardziej czytelne. Na przykład kolorowi jasnozielonemu odpowiada liczba 4 278 255 360, która w zapisie szesnastkowym ma postać FF00FF00. Pierwsze dwie pozycje „FF” (255) wskazują kanał **alpha**. W przypadku funkcji **RGB** i **HSL** będzie to zawsze wartość „FF” (nieprzezroczysta). Następne dwie pozycje „00” oznaczają wartość składowej czerwonej (**red**), kolejne dwie pozycje „FF” wartość składowej zielonej (**green**), a ostatnie dwie pozycje „00” wartość składowej niebieskiej (**blue**).

Przykład: Wyrażenie wykresu
W tym przykładzie stosuje się niestandardowy kolor do wykresu:

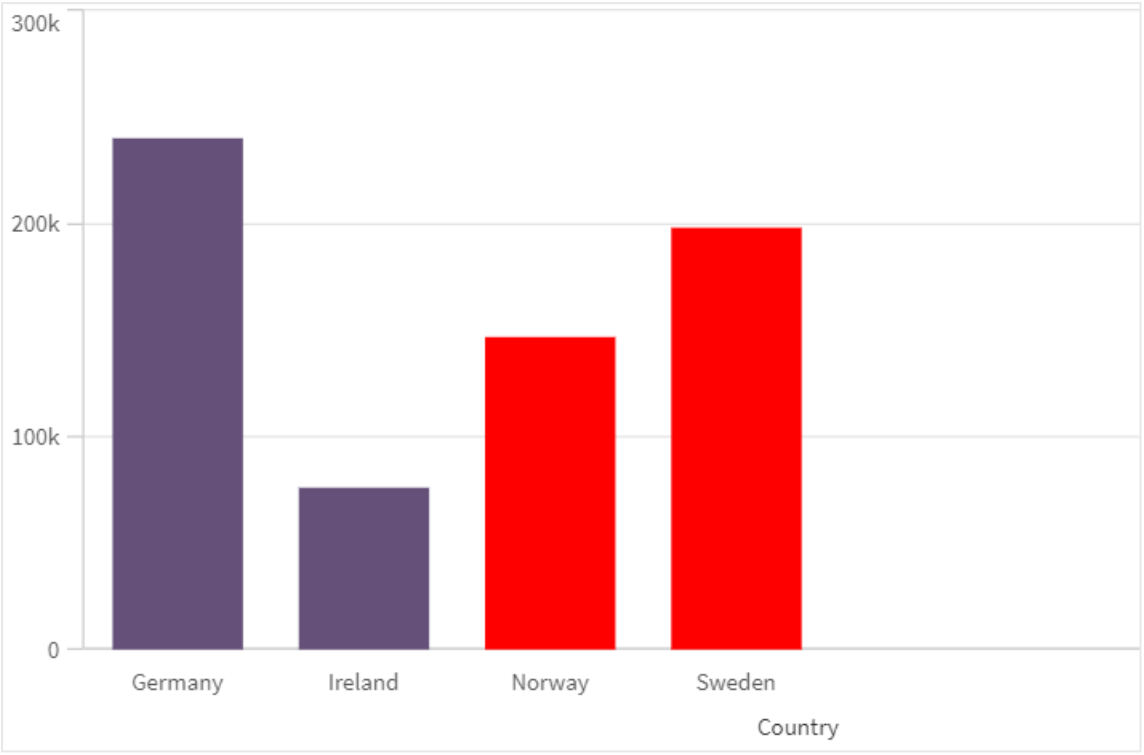
Dane zastosowane w tym przykładzie:

```
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 ',');
```

Wprowadź następujące wyrażenie w panelu właściwości **Kolory i legenda**:

```
If (Sum(Sales)>Sum(Budget),RGB(255,0,0),RGB(100,80,120))
```

Wynik:



Przykład: Skrypt ładowania

Poniższy przykład prezentuje wartości RGB równoważne wartościom w formacie szesnastkowym.

```
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];
```

Wynik:

Tekst	Kolor
010203	RGB(1,2,3)
AABBCC	RGB(170,187,204)

HSL

Funkcja **HSL()** służy w wyrażeniach do ustawienia lub obliczenia wartości właściwości koloru obiektu wykresu, przy czym kolor jest zdefiniowany wartościami składowych **hue** (odcień), **saturation** (nasycenie) i **luminosity** (jasność) z zakresu od 0 do 1.

Składnia:

```
HSL (hue, saturation, luminosity)
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
hue, saturation, luminosity	Wartości składników hue, saturation i luminosity zawierają się w zakresie od 0 do 1.

Wszystkie argumenty muszą być wyrażeniami, które zwracają liczby całkowite w zakresie od 0 do 1.

W przypadku interpretowania składnika liczbowego i przekształcania go na zapis szesnastkowy wartości RGB składników koloru są bardziej czytelne. Na przykład kolorowi jasnozielonemu odpowiada liczba 4 278 255 360, która w zapisie szesnastkowym ma postać FF00FF00 oraz RGB (0,255,0). Odpowiada to zapisowi HSL (80/240, 240/240, 120/240) – wartości HSL odpowiadającej zapisowi (0.33, 1, 0.5).

5.5 Funkcje warunkowe

Wszystkie funkcje warunkowe oceniają warunek, a następnie zwracają różne odpowiedzi w zależności od wartości warunku. Funkcje te mogą być stosowane w skryptach ładowania danych i wyrażeniach wykresu.

Przegląd funkcji warunkowych

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

alt

Funkcja **alt** zwraca pierwszy z parametrów o poprawnej reprezentacji liczbowej. Jeśli taki parametr nie zostanie znaleziony, wówczas zwrócony zostanie ostatni parametr. Można użyć dowolnej liczby parametrów.

```
alt (expr1[ , expr2 , expr3 , ...] , else)
```

class

Funkcja **class** przydziela pierwszy parametr do interwału klasy. Wynikiem jest wartość podwójna z a<=x<b jako wartością tekstową, gdzie argumenty a ib są górnym i dolnym limitami przedziału, oraz dolną granicą jako wartością liczbową.

```
class (expression, interval [ , label [ , offset ]])
```

coalesce

Funkcja **coalesce** zwraca pierwszy z parametrów o poprawnej reprezentacji non-NULL. Można użyć dowolnej liczby parametrów.

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

if

Funkcja **if** zwraca wartość w zależności od tego, czy warunek określony w ramach tej funkcji uwzględnia wartości True czy False.

```
if (condition , then , else)
```

match

Funkcja **match** porównuje pierwszy parametr z wszystkimi kolejnymi parametrami i zwraca liczbową lokalizację z wyrażeń dopasowanych. Wielkość liter w porównaniu ma znaczenie.

```
match ( str, expr1 [ , expr2,...exprN ])
```

mixmatch

Funkcja **mixmatch** porównuje pierwszy parametr z wszystkimi kolejnymi parametrami i zwraca liczbową lokalizację z wyrażeń dopasowanych. Wielkość liter w porównaniu nie ma znaczenia.

```
mixmatch ( str, expr1 [ , expr2,...exprN ])
```

pick

Funkcja pick zwraca *n*-te wyrażenie na liście.

```
pick (n, expr1[ , expr2,...exprN])
```

wildmatch

Funkcja **wildmatch** porównuje pierwszy parametr ze wszystkimi kolejnymi parametrami i zwraca numer pasującego wyrażenia. Funkcja ta zezwala na stosowanie symboli wieloznacznych (***** i **?**) w ciągach znaków porównania. ***** stanowi dopasowanie dla dowolnej sekwencji znaków. **?** jest dopasowaniem dla dowolnego pojedynczego znaku. Wielkość liter w porównaniu nie ma znaczenia.

```
wildmatch ( str, expr1 [ , expr2,...exprN ])
```

alt

Funkcja **alt** zwraca pierwszy z parametrów o poprawnej reprezentacji liczbowej. Jeśli taki parametr nie zostanie znaleziony, wówczas zwrócony zostanie ostatni parametr. Można użyć dowolnej liczby parametrów.

Składnia:

```
alt(expr1[ , expr2 , expr3 , ...] , else)
```

Argumenty:

Argumenty

Argument	Opis
expr1	Pierwsze wyrażenie do sprawdzenia poprawnej reprezentacji liczbowej.
expr2	Drugie wyrażenie do sprawdzenia poprawnej reprezentacji liczbowej.
expr3	Trzecie wyrażenie do sprawdzenia poprawnej reprezentacji liczbowej.
else	Wartość do zwrócenia, jeśli żaden z poprzednich parametrów nie ma poprawnej reprezentacji liczbowej.

Funkcja alt jest często używana z funkcjami interpretacji liczby lub daty. W ten sposób aplikacja Qlik Sense może sprawdzać różne formaty danych w kolejności opartej na priorytetach. Można jej także użyć do obsługi wartości NULL w wyrażeniach liczbowych.

Przykłady:

Przykłady

Przykład	Wynik

`alt( date#( dat , 'YYYY/MM/DD' ),`
`date#( dat , 'MM/DD/YYYY' ),`
`date#( dat , 'MM/DD/YY' ),`
`'No valid date' )`
	To wyrażenie sprawdzi, czy pole date zawiera datę zgodną z dowolnym z trzech określonych formatów. Jeśli tak, zostanie zwrócona wartość podwójna zawierająca pierwotny ciąg znaków i poprawną reprezentację liczbową daty. Jeśli dopasowanie nie zostanie znalezione, zostanie zwrócony tekst 'No valid date' (bez żadnej poprawnej reprezentacji liczbowej).

`alt(Sales,0) + alt(Margin,0)`	To wyrażenie dodaje pola Sales i Margin, zastępując wszelkie brakujące wartości (NULL) liczbą 0.

class

Funkcja **class** przydziela pierwszy parametr do interwału klasy. Wynikiem jest wartość podwójna z a<=x<b jako wartością tekstową, gdzie argumenty a ib są górnym i dolnym limitami przedziału, oraz dolną granicą jako wartością liczbową.

Składnia:

```
class(expression, interval [ , label [ , offset ]])
```

Argumenty:

Argumenty

Argument	Opis
interval	Liczba określająca szerokość przedziału.

Argument	Opis
label	Dowolny ciąg znaków, który może zastąpić „x” w tekście wyniku.
offset	Liczba, która może zostać użyta jako przesunięcie od domyślnego punktu początkowego klasyfikacji. Zazwyczaj domyślny punkt startowy wynosi 0.

Przykłady:

Przykłady

Przykład	Wynik
`class( var,10 ) z var = 23`	zwraca wartość '20<=x<30'
`class( var,5,'value' ) z var = 23`	zwraca wartość '20<= value <25'
`class( var,10,'x',5 ) z var = 23`	zwraca wartość '15<=x<25'

Przykład – skrypt ładowania używający class

Przykład: skrypt ładowania

Skrypt ładowania

W tym przykładzie ładujemy tabelę zawierającą nazwiska i wiek osób. Chcemy dodać pola klasyfikujące każdą osobę zgodnie z grupą wiekową z interwałem dziesięciu lat. Oryginalna tabela źródłowa wygląda następująco.

Wyniki

Name	Age
John	25
Karen	42
Yoshi	53

Aby dodać pole klasyfikacji grupy wiekowej, można dodać poprzedzającą instrukcję ładowania przy użyciu funkcji **class**.

Utwórz nową kartę w edytorze ładowania danych, a następnie załaduj następujące dane jako ładowanie wbudowane. Utwórz tabelę poniżej w Qlik Sense, aby zobaczyć wyniki.

```
 LOAD *, class(Age, 10, 'age') As Agegroup;  LOAD * INLINE [ Age, Name 25, John 42, Karen 53, Yoshi];
```

Wyniki

Wyniki

Name	Age	Agegroup
John	25	20 <= age < 30
Karen	42	40 <= age < 50
Yoshi	53	50 <= age < 60

coalesce

Funkcja **coalesce** zwraca pierwszy z parametrów o poprawnej reprezentacji non-NULL. Można użyć dowolnej liczby parametrów.

Składnia:

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

Argumenty:

Argumenty

Argument	Opisu
expr1	Pierwsze wyrażenie do sprawdzenia poprawnej reprezentacji innej niż NULL.
expr2	Drugie wyrażenie do sprawdzenia poprawnej reprezentacji innej niż NULL.
expr3	Trzecie wyrażenie do sprawdzenia poprawnej reprezentacji innej niż NULL.

Przykłady:

Przykłady

Przykład	Wynik
	To wyrażenie zmienia wszystkie wartości NULL pola na „N/A”.
`Coalesce(ProductDescription, ProductName, ProductCode, 'no description available')`	To wyrażenie wybierze spośród trzech różnych pól opisu produktu, gdy niektóre pola mogą nie zawierać wartości dla produktu. Zwrócone zostanie pierwsze z pól z wartością różną od null w podanej kolejności. Jeśli żadne z pól nie będzie zawierać wartości, wynikiem będzie „brak opisu”.
`Coalesce(TextBetween(FileName, '"', '"'), FileName)`	To wyrażenie spowoduje obcięcie potencjalnych otaczającego cudzysłowu z pola *FileName*. Jeśli *FileName* będzie w cudzysłowie, zostanie on usunięty, a zwrócona zostanie ujęta w nim wartość *FileName* bez cudzysłowu. Jeśli funkcja *TextBetween* nie znajdzie ograniczników, zwróci wartość null, którą odrzuci funkcja **Coalesce**, zwracając zamiast tego nieprzetworzoną wartość *FileName*.

if

Funkcja **if** zwraca wartość w zależności od tego, czy warunek określony w ramach tej funkcji uwzględnia wartości True czy False.

Składnia:

```
if(condition , then [, else])
```

Argumenty

Argument	Opis
condition	Wyrażenie, które jest interpretowane logicznie.
then	Wyrażenie, które może być dowolnego typu. Jeśli wyrażenie*condition* ma wartość True, wówczas funkcja if zwraca wartość wyrażenia *then*.
else	Wyrażenie, które może być dowolnego typu. Jeśli wyrażenie*condition* ma wartość False, wówczas funkcja if zwraca wartość wyrażenia *else*. Ten parametr jest opcjonalny. Jeśli warunek *condition* jest False, zwracana jest wartość NULL, jeśli nie określono else.

Przykład

Przykład	Wynik
`if( Amount>= 0, 'OK', 'Alarm' )`	To wyrażenie sprawdza, czy ilość jest liczbą dodatnią (0 lub większą), i zwraca wówczas 'OK'. Jeśli ilość jest liczbą mniejszą od 0, zwracane jest 'Alarm'.

Przykład – skrypt ładowania używający if

Przykład: Skrypt ładowania

Skrypt ładowania

Funkcja If może być używana w skrypcie ładowania z innymi metodami i obiektami, w tym również ze zmiennymi. Jeśli na przykład ustawisz zmienną *threshold* (próg) i chcesz uwzględnić pole w modelu danych na podstawie tego progu, możesz zrobić to tak:

Utwórz nową kartę w edytorze ładowania danych, a następnie załaduj następujące dane jako ładowanie wbudowane. Utwórz tabelę poniżej w Qlik Sense, aby zobaczyć wyniki.

```
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;
```

Wyniki

Tabela programu Qlik Sense przedstawiająca wynik użycia funkcji *if* w skrypcie ładowania.

transaction_id	W porównaniu do progu
3750	Mniej niż 100
3751	Więcej niż 100
3752	Mniej niż 100
3753	Więcej niż 100
3754	Więcej niż 100
3756	Mniej niż 100
3757	Więcej niż 100

Przykłady – wyrażenia wykresu używające if

Przykłady: Wyrażenia wykresu

Wyrażenie wykresu 1

Skrypt ładowania

Utwórz nową kartę w edytorze ładowania danych, a następnie załaduj następujące dane jako ładowanie wbudowane. Po załadowaniu danych utwórz poniższe przykłady wyrażeń wykresu w tabeli Qlik Sense.

```
MyTable:
LOAD * inline [Date, Location, Incidents
1/3/2016, Beijing, 0
1/3/2016, Boston, 12
1/3/2016, Stockholm, 3
1/3/2016, Toronto, 0
```

```
1/4/2016, Beijing, 0
1/4/2016, Boston, 8];
```

Tabela aplikacji Qlik Sense z przykładami funkcji *if* w wyrażeniu wykresu.

Date	Lokalizacja	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	Ostrzeżenie
1/3/2016	Toronto	0	Ok	Ok
1/4/2016	Beijing	0	Ok	Ok
1/4/2016	Boston	8	Ok	Ostrzeżenie

Wyrażenie wykresu 2

W nowej aplikacji dodaj następujący skrypt w nowej karcie w edytorze ładowania danych, a następnie załaduj dane. Następnie możesz utworzyć tabelę z poniższymi wyrażeniami wykresu.

```
SET FirstWeekDay=0;
Load
Date(MakeDate(2022)+RecNo()-1) as Date
Autogenerate 14;
```

Tabela aplikacji Qlik Sense z przykładem funkcji *if* w wyrażeniu wykresu.

Data	WeekDay(Date)	If(WeekDay (Date)>=5,'WeekEnd','Normal Day')
1/1/2022	Sat	WeekEnd
1/2/2022	Nie	WeekEnd
1/3/2022	Pon	Normal Day
1/4/2022	Wto	Normal Day
1/5/2022	Śro	Normal Day
1/6/2022	Thu	Normal Day
1/7/2022	Fri	Normal Day
1/8/2022	Sat	WeekEnd
1/9/2022	Nie	WeekEnd
1/10/2022	Pon	Normal Day
1/11/2022	Wto	Normal Day

Data	WeekDay(Date)	If(WeekDay (Date)>=5,'WeekEnd','Normal Day')
1/12/2022	Śro	Normal Day
1/13/2022	Thu	Normal Day
1/14/2022	Fri	Normal Day

match

Funkcja **match** porównuje pierwszy parametr z wszystkimi kolejnymi parametrami i zwraca liczbową lokalizację z wyrażeń dopasowanych. Wielkość liter w porównaniu ma znaczenie.

Składnia:

```
match( str, expr1 [ , expr2,...exprN ])
```

*Aby dokonywać porównań ignorujących wielkość liter, należy użyć funkcji **mixmatch**. Aby dokonywać porównań ignorujących wielkość liter i symboli wieloznacznych, należy użyć funkcji **wildmatch**.*

Przykład: Skrypt ładowania używający match

Przykład: Skrypt ładowania

Skrypt ładowania

Funkcji match można użyć, aby załadować podzbiór danych. Można na przykład zwrócić wartość liczbową dla wyrażenia w funkcji. Następnie można ograniczyć załadowane dane na podstawie tej wartości liczbowej. W przypadku braku dopasowania funkcja Match zwraca 0. Wszystkie wyrażenia, dla których nie zostanie znalezione dopasowanie, zwrócą w tym przykładzie 0 i zostaną wykluczone z danych ładowanych przez instrukcję WHERE.

Utwórz nową kartę w edytorze ładowania danych, a następnie załaduj następujące dane jako ładowanie wbudowane. Utwórz tabelę poniżej w Qlik Sense, aby zobaczyć wyniki.

```
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;
```

Wyniki

Tabela programu Qlik Sense przedstawiająca wynik użycia funkcji match w skrypcie ładowania

Color Code Blue and Black	Customer
Black	203521
Black	3036491
Blue	2038593

Przykłady – wyrażenia wykresu używające match

Przykłady: Wyrażenia wykresu

Wyrażenie wykresu 1

Skrypt ładowania

Utwórz nową kartę w edytorze ładowania danych, a następnie załaduj następujące dane jako ładowanie wbudowane. Po załadowaniu danych utwórz poniższe przykłady wyrażeń wykresu w tabeli Qlik Sense.

```
MyTable: Load * inline [Cities, Count Toronto, 123 Toronto, 234 Toronto, 231 Boston, 32
Boston, 23 Boston, 1341 Beijing, 234 Beijing, 45 Beijing, 235 Stockholm, 938 Stockholm, 39
Stockholm, 189 zurich, 2342 zurich, 9033 zurich, 0039];
```

Pierwsze wyrażenie z tabeli poniżej zwraca 0 dla Stockholm, ponieważ „Stockholm” nie znajduje się na liście wyrażeń w funkcji **match**. Zwraca również 0 dla „Zurich”, ponieważ porównanie **match** rozróżnia wielkość liter.

Tabela aplikacji Qlik Sense z przykładami funkcji *match* w wyrażeniu wykresu.

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

Wyrażenie wykresu 2

Funkcji match można użyć, aby wykonać niestandardowe sortowanie dla wyrażenia.

Domyślnie kolumny są posortowane w porządku liczbowym lub alfabetycznym – w zależności od danych.

Tabela aplikacji Qlik Sense pokazująca przykład domyślnego porządku sortowania

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Aby zmienić porządek, wykonaj następujące czynności:

1. W panelu **Właściwości** otwórz sekcję **Sortowanie** dla wykresu.
2. Wyłącz automatyczne sortowanie dla kolumny, względem której chcesz wykonać sortowanie niestandardowe.
3. Usuń zaznaczenie opcji **Sortuj w kolejności liczbowej** i **Sortuj alfabetycznie**.
4. Zaznacz opcję **Sortuj wg wyrażenia**, a następnie wprowadź wyrażenie podobne do następującego:
   `=match( Cities, 'Toronto','Boston','Beijing','Stockholm','zurich')`
   Porządek sortowania w kolumnie Cities ulegnie zmianie.

Tabela aplikacji Qlik Sense pokazująca przykład zmiany porządku sortowania za pomocą funkcji *match*

Cities
Toronto
Boston
Beijing
Stockholm
zurich

Możesz także wyświetlić zwróconą wartość liczbową.

Tabela aplikacji Qlik Sense pokazująca przykład wartości liczbowych, które są zwracane przez funkcję *match*

Miasta	Cities & ' - ' & match (Cities, 'Toronto','Boston', 'Beijing','Stockholm','zurich')
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

mixmatch

Funkcja **mixmatch** porównuje pierwszy parametr z wszystkimi kolejnymi parametrami i zwraca liczbową lokalizację z wyrażeń dopasowanych. Wielkość liter w porównaniu nie ma znaczenia.

Składnia:

```
mixmatch( str, expr1 [ , expr2,...exprN ])
```

Aby zamiast tego dokonać porównania uwzględniającego wielkość liter, należy użyć funkcji **match**. Aby dokonywać porównań ignorujących wielkość liter i symboli wieloznacznych, należy użyć funkcji **wildmatch**.

Przykład – skrypt ładowania używający mixmatch

Przykład: Skrypt ładowania

Skrypt ładowania

Funkcji mixmatch można użyć, aby załadować podzbiór danych. Można na przykład zwrócić wartość liczbową dla wyrażenia w funkcji. Następnie można ograniczyć załadowane dane na podstawie tej wartości liczbowej. W przypadku braku dopasowania funkcja Mixmatch zwraca 0. Wszystkie wyrażenia, dla których nie zostanie znalezione dopasowanie, zwrócą w tym przykładzie 0 i zostaną wykluczone z danych ładowanych przez instrukcję WHERE.

Utwórz nową kartę w edytorze ładowania danych, a następnie załaduj następujące dane jako ładowanie wbudowane. Utwórz tabelę poniżej w Qlik Sense, aby zobaczyć wyniki.

```
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;
```

Wyniki

Tabela programu Qlik Sense przedstawiająca wynik użycia funkcji mixmatch w skrypcie ładowania.

Color Code Black, Blue, blue	Customer
Black	203521
Black	3036491

Color Code Black, Blue, blue	Customer
Blue	2038593
Blue	5646471

Przykłady – wyrażenia wykresu używające mixmatch

Przykłady: Wyrażenia wykresu
Utwórz nową kartę w edytorze ładowania danych, a następnie załaduj następujące dane jako ładowanie wbudowane. Po załadowaniu danych utwórz poniższe przykłady wyrażeń wykresu w tabeli Qlik Sense.

Wyrażenie wykresu 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];
```

Pierwsze wyrażenie z tabeli poniżej zwraca 0 dla Stockholm, ponieważ „Stockholm” nie znajduje się na liście wyrażeń w funkcji **mixmatch**. Zwraca 4 dla „Zurich”, ponieważ porównanie **mixmatch** nie rozróżnia wielkości liter.

Tabela aplikacji Qlik Sense z przykładami funkcji *mixmatch* w wyrażeniu wykresu.

Cities	mixmatch(Cities,'Toronto','Boston','Beijing','Zurich')	mixmatch(Cities,'Toronto','Boston','Beijing','Stockholm','Zurich')
Beijing	3	3
Boston	2	2
Stockholm	0	4
Toronto	1	1
zurich	4	5

Wyrażenie wykresu 2

Funkcji mixmatch można użyć, aby wykonać niestandardowe sortowanie dla wyrażenia.

Domyślnie kolumny są posortowane w porządku alfabetycznym lub liczbowym – w zależności od danych.

Tabela aplikacji Qlik Sense pokazująca przykład domyślnego porządku sortowania

Cities
Beijing
Boston
Stockholm

Cities
Toronto
zurich

Aby zmienić porządek, wykonaj następujące czynności:

1. W panelu **Właściwości** otwórz sekcję **Sortowanie** dla wykresu.
2. Wyłącz automatyczne sortowanie dla kolumny, względem której chcesz wykonać sortowanie niestandardowe.
3. Usuń zaznaczenie opcji **Sortuj w kolejności liczbowej** i **Sortuj alfabetycznie**.
4. Wybierz **Sortuj wg wyrażenia**, a następnie wprowadź następujące wyrażenie:
   `=mixmatch( Cities, 'Toronto','Boston','Beijing','Stockholm','Zurich')`
   Porządek sortowania w kolumnie Cities ulegnie zmianie.

Tabela aplikacji Qlik Sense pokazująca przykład zmiany porządku sortowania za pomocą funkcji *mixmatch*.

Cities
Toronto
Boston
Beijing
Stockholm
zurich

Możesz także wyświetlić zwróconą wartość liczbową.

Tabela aplikacji Qlik Sense pokazująca przykład wartości liczbowych, które są zwracane przez funkcję *mixmatch*.

Miasta	Cities & ' - ' & mixmatch (Cities, 'Toronto','Boston', 'Beijing','Stockholm','Zurich')
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

pick

Funkcja pick zwraca *n*-te wyrażenie na liście.

Składnia:

```
pick(n, expr1[ , expr2,...exprN])
```

Argumenty:

Argumenty

Argument	Opis
n	*n* jest liczbą całkowitą z zakresu od 1 do N.

Przykład:

Przykład

Przykład	Wynik
`pick( N, 'A','B',4, 6 )`	zwraca 'B', jeśli N = 2 zwraca 4, jeśli N = 3

wildmatch

Funkcja **wildmatch** porównuje pierwszy parametr ze wszystkimi kolejnymi parametrami i zwraca numer pasującego wyrażenia. Funkcja ta zezwala na stosowanie symboli wieloznacznych (* i ?) w ciągach znaków porównania. * stanowi dopasowanie dla dowolnej sekwencji znaków. ? jest dopasowaniem dla dowolnego pojedynczego znaku. Wielkość liter w porównaniu nie ma znaczenia.

Składnia:

```
wildmatch( str, expr1 [ , expr2,...exprN ])
```

Aby dokonywać porównań między symbolami wieloznacznymi, należy użyć funkcji **match** lub **mixmatch**.

Przykład: Skrypt ładowania używający wildmatch

Przykład: Skrypt ładowania

Skrypt ładowania

Funkcji wildmatch można użyć, aby załadować podzbiór danych. Można na przykład zwrócić wartość liczbową dla wyrażenia w funkcji. Następnie można ograniczyć załadowane dane na podstawie tej wartości liczbowej. W przypadku braku dopasowania funkcja Wildmatch zwraca 0. Wszystkie wyrażenia, dla których nie zostanie znalezione dopasowanie, zwrócą w tym przykładzie 0 i zostaną wykluczone z danych ładowanych przez instrukcję WHERE.

Utwórz nową kartę w edytorze ładowania danych, a następnie załaduj następujące dane jako ładowanie wbudowane. Utwórz tabelę poniżej w Qlik Sense, aby zobaczyć wyniki.

```
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;
```

Wyniki

Tabela programu Qlik Sense przedstawiająca wynik użycia funkcji *wildmatch* w skrypcie ładowania

Color Code Black, Blue, blue, Red	Customer
Black	203521
Black	3036491
Blue	2038593
Blue	5646471
Red	049681
Red	2038593

Przykłady: Wyrażenia wykresu używające wildmatch

Przykład: Wyrażenie wykresu

Wyrażenie wykresu 1

Utwórz nową kartę w edytorze ładowania danych, a następnie załaduj następujące dane jako ładowanie wbudowane. Po załadowaniu danych utwórz poniższe przykłady wyrażeń wykresu w tabeli Qlik Sense.

```
MyTable: Load * inline [Cities, Count Toronto, 123 Toronto, 234 Toronto, 231 Boston, 32 Boston, 23 Boston, 1341 Beijing, 234 Beijing, 45 Beijing, 235 Stockholm, 938 Stockholm, 39 Stockholm, 189 zurich, 2342 zurich, 9033 zurich, 0039];
```

Pierwsze wyrażenie z tabeli poniżej zwraca 0 dla Stockholm, ponieważ „Stockholm” nie znajduje się na liście wyrażeń w funkcji **wildmatch**. Zwraca również 0 dla „Boston”, ponieważ dopasowaniem dla ? jest tylko pojedynczy znak.

Tabela aplikacji Qlik Sense z przykładami funkcji *wildmatch* w wyrażeniu wykresu.

Cities	wildmatch(Cities,'Tor*','?ton','Beijing','*urich')	wildmatch(Cities,'Tor*','???ton','Beijing','Stockholm','*urich')
Beijing	3	3
Boston	0	2
Stockholm	0	4

Cities	wildmatch(Cities,'Tor*','?ton','Beijing','*urich')	wildmatch(Cities,'Tor*','???ton','Beijing','Stockholm','*urich')
Toronto	1	1
zurich	4	5

Wyrażenie wykresu 2

Funkcji wildmatch można użyć, aby wykonać niestandardowe sortowanie dla wyrażenia.

Domyślnie kolumny są posortowane w porządku liczbowym lub alfabetycznym – w zależności od danych.

Tabela aplikacji Qlik Sense pokazująca przykład domyślnego porządku sortowania

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Aby zmienić porządek, wykonaj następujące czynności:

1. W panelu **Właściwości** otwórz sekcję **Sortowanie** dla wykresu.
2. Wyłącz automatyczne sortowanie dla kolumny, względem której chcesz wykonać sortowanie niestandardowe.
3. Usuń zaznaczenie opcji **Sortuj w kolejności liczbowej** i **Sortuj alfabetycznie**.
4. Zaznacz opcję **Sortuj wg wyrażenia**, a następnie wprowadź wyrażenie podobne do następującego:
   `=wildmatch( Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')`
   Porządek sortowania w kolumnie Cities ulegnie zmianie.

Tabela aplikacji Qlik Sense pokazująca przykład zmiany porządku sortowania za pomocą funkcji *wildmatch*.

Cities
Toronto
Boston
Beijing
Stockholm
zurich

Możesz także wyświetlić zwróconą wartość liczbową.

Tabela aplikacji Qlik Sense pokazująca przykład wartości liczbowych, które są zwracane przez funkcję *wildmatch*

Miasta	Cities & ' - ' & wildmatch (Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

5.6 Funkcje licznikowe

W tej sekcji opisano funkcje powiązane z licznikami rekordów podczas oceny instrukcji **LOAD** w skrypcie ładowania danych. Jedyną funkcją, której można użyć w wyrażeniach wykresu, jest **RowNo()**.

Niektóre funkcje licznikowe nie muszą zawierać parametrów, ale wciąż wymagany jest nawias końcowy.

Przegląd funkcji licznikowych

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

autonumber

Ta funkcja skryptu zwraca niepowtarzalną wartość całkowitą dla każdej odrębnie przetwarzanej wartości *expression* napotkanej podczas wykonywania skryptu. Ta funkcja może zostać użyta np. w celu utworzenia kompaktowej reprezentacji klucza złożonego w pamięci.

```
autonumber (expression[ , AutoID])
```

autonumberhash128

Ta funkcja skryptu oblicza 128-bitowy skrót połączonych wartości wyrażenia wejściowego oraz zwraca niepowtarzalną wartość całkowitą dla każdej odrębnej wartości skrótu napotkanej podczas wykonywania skryptu. Ta funkcja może zostać użyta np. w celu utworzenia kompaktowej reprezentacji klucza złożonego w pamięci.

```
autonumberhash128 (expression {, expression})
```

autonumberhash256

Ta funkcja skryptu oblicza 256-bitowy skrót połączonych wartości wyrażenia wejściowego oraz zwraca niepowtarzalną wartość całkowitą dla każdej odrębnej wartości skrótu napotkanej podczas wykonywania skryptu. Ta funkcja może zostać użyta np. w celu utworzenia kompaktowej reprezentacji klucza złożonego w pamięci.

```
autonumberhash256 (expression {, expression})
```

IterNo

Ta funkcja skryptu zwraca liczbę całkowitą, która wskazuje numer oceny pojedynczego wiersza w instrukcji a **LOAD** z klauzulą **while**. Pierwsza iteracja ma numer 1. Funkcja **IterNo** jest przydatna tylko wtedy, gdy jest stosowana razem z klauzulą **while**.

```
IterNo ( )
```

RecNo

Ta funkcja skryptu zwraca liczbę całkowitą dla numeru aktualnie odczytanego wiersza bieżącej tabeli. Pierwszy wiersz ma numer 1.

```
RecNo ( )
```

RowNo - script function

Ta funkcja zwraca liczbę całkowitą dla pozycji bieżącego wiersza w wynikowej tabeli wewnętrznej Qlik Sense. Pierwszy wiersz ma numer 1.

```
RowNo ( )
```

RowNo - chart function

Funkcja **RowNo()** zwraca numer bieżącego wiersza w bieżącym segmencie kolumn tabeli. W przypadku wykresów bitmapowych funkcja **RowNo()** zwraca numer bieżącego wiersza w tabeli prostej odpowiadającej wykresowi.

```
RowNo — funkcja wykresu([TOTAL])
```

autonumber

Ta funkcja skryptu zwraca niepowtarzalną wartość całkowitą dla każdej odrębnie przetwarzanej wartości *expression* napotkanej podczas wykonywania skryptu. Ta funkcja może zostać użyta np. w celu utworzenia kompaktowej reprezentacji klucza złożonego w pamięci.

> *Można łączyć tylko te klucze **autonumber**, które zostały wygenerowane w tym samym ładowaniu danych, ponieważ liczba całkowita jest generowana zgodnie z kolejnością odczytu tabeli. Aby użyć kluczy stałych w różnych ładowaniach danych, niezależnie od sortowania danych źródłowych, należy skorzystać z funkcji **hash128**, **hash160** lub **hash256**.*

Składnia:

```
autonumber(expression[ , AutoID])
```

Argumenty:

Argument	Opis
AutoID	W celu utworzenia wielu instancji licznika, gdy funkcja **autonumber** jest używana na różnych kluczach w skrypcie, można użyć opcjonalnego parametru *AutoID* przeznaczonego do nazywania poszczególnych liczników.

Przykład: Tworzenie klucza złożonego

W tym przykładzie utworzymy klucz złożony przy użyciu funkcji **autonumber** w celu ochrony pamięci. Na potrzeby prezentacji przykład jest krótki, ale funkcja ta sprawdza się szczególnie w przypadku tabel zawierających wiele wierszy.

Przykładowe dane

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

Dane źródłowe są ładowane przy użyciu danych wbudowanych. Następnie dodajemy wcześniejsze ładowanie, które tworzy klucz złożony z pól Region, Year i Month.

```
RegionSales:
LOAD *,
AutoNumber(Region&Year&Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
South, 2013,   May, 367
South, 2013,   May,    221
];
```

Wynikowa tabela wygląda następująco:

Tabela wynikowa

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

Aby połączyć kolejną tabelę, w tym przykładzie można odwołać się do klucza RYMkey, na przykład 1, zamiast do ciągu znaków „North2014May”.

Teraz w podobny sposób ładujemy źródłową tabelę kosztów. Pola Region, Year i Month są wykluczone we wcześniejszym ładowaniu, aby uniknąć tworzenia klucza syntetycznego. Tworzony jest już bowiem klucz złożony z funkcją **autonumber**, łączącą tabele.

```
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
];
```

Teraz możemy dodać do arkusza wizualizację tabeli oraz pola Region, Year i Month, jak również miary Sum dla sprzedaży i kosztów. Tabela będzie wyglądać następująco:

Tabela wynikowa

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

Ta funkcja skryptu oblicza 128-bitowy skrót połączonych wartości wyrażenia wejściowego oraz zwraca niepowtarzalną wartość całkowitą dla każdej odrębnej wartości skrótu napotkanej podczas wykonywania skryptu. Ta funkcja może zostać użyta np. w celu utworzenia kompaktowej reprezentacji klucza złożonego w pamięci.

*Można łączyć tylko te klucze **autonumberhash128**, które zostały wygenerowane w tym samym ładowaniu danych, ponieważ liczba całkowita jest generowana zgodnie z kolejnością odczytu tabeli. Aby użyć kluczy stałych w różnych ładowaniach danych, niezależnie od sortowania danych źródłowych, należy skorzystać z funkcji **hash128**, **hash160** lub **hash256**.*

Składnia:

```
autonumberhash128(expression {, expression})
```

Przykład: Tworzenie klucza złożonego

W tym przykładzie utworzymy klucz złożony przy użyciu funkcji **autonumberhash128** w celu ochrony pamięci. Na potrzeby prezentacji przykład jest krótki, ale funkcja ta sprawdza się szczególnie w przypadku tabel zawierających wiele wierszy.

Przykładowe dane

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

Dane źródłowe są ładowane przy użyciu danych wbudowanych. Następnie dodajemy wcześniejsze ładowanie, które tworzy klucz złożony z pól Region, Year i Month.

```
RegionSales:
LOAD *,
AutoNumberHash128(Region, Year, Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
South, 2013,   May, 367
South, 2013,   May,    221
];
```

Wynikowa tabela wygląda następująco:

Tabela wynikowa

Region	Year	Month	Sales	RYMkey
North	2014	May	245	1
North	2014	May	347	1
North	2014	June	127	2

Region	Year	Month	Sales	RYMkey
South	2014	June	645	3
South	2013	May	367	4
South	2013	May	221	4

Aby połączyć kolejną tabelę, w tym przykładzie można odwołać się do klucza RYMkey, na przykład 1, zamiast do ciągu znaków „North2014May”.

Teraz w podobny sposób ładujemy źródłową tabelę kosztów. Pola Region, Year i Month są wykluczone we wcześniejszym ładowaniu, aby uniknąć tworzenia klucza syntetycznego. Tworzony jest już bowiem klucz złożony z funkcją **autonumberhash128**, łączącą tabele.

```
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
];
```

Teraz możemy dodać do arkusza wizualizację tabeli oraz pola Region, Year i Month, jak również miary Sum dla sprzedaży i kosztów. Tabela będzie wyglądać następująco:

Tabela wynikowa

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

Ta funkcja skryptu oblicza 256-bitowy skrót połączonych wartości wyrażenia wejściowego oraz zwraca niepowtarzalną wartość całkowitą dla każdej odrębnej wartości skrótu napotkanej podczas wykonywania skryptu. Ta funkcja może zostać użyta np. w celu utworzenia kompaktowej reprezentacji klucza złożonego w pamięci.

*Można łączyć tylko te klucze **autonumberhash256**, które zostały wygenerowane w tym samym ładowaniu danych, ponieważ liczba całkowita jest generowana zgodnie z kolejnością odczytu tabeli. Aby użyć kluczy stałych w różnych ładowaniach danych, niezależnie od sortowania danych źródłowych, należy skorzystać z funkcji **hash128**, **hash160** lub **hash256**.*

Składnia:

```
autonumberhash256(expression {, expression})
```

Przykład: Tworzenie klucza złożonego

W tym przykładzie utworzymy klucz złożony przy użyciu funkcji **autonumberhash256** w celu ochrony pamięci. Na potrzeby prezentacji przykład jest krótki, ale funkcja ta sprawdza się szczególnie w przypadku tabel zawierających wiele wierszy.

Tabela przykładowa

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

Dane źródłowe są ładowane przy użyciu danych wbudowanych. Następnie dodajemy wcześniejsze ładowanie, które tworzy klucz złożony z pól Region, Year i Month.

```
RegionSales:
LOAD *,
AutoNumberHash256(Region, Year, Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
South, 2013,   May, 367
South, 2013,   May,    221
];
```

Wynikowa tabela wygląda następująco:

Tabela wynikowa

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

Aby połączyć kolejną tabelę, w tym przykładzie można odwołać się do klucza RYMkey, na przykład 1, zamiast do ciągu znaków „North2014May”.

Teraz w podobny sposób ładujemy źródłową tabelę kosztów. Pola Region, Year i Month są wykluczone we wcześniejszym ładowaniu, aby uniknąć tworzenia klucza syntetycznego. Tworzony jest już bowiem klucz złożony z funkcją **autonumberhash256**, łączącą tabele.

```
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
];
```

Teraz możemy dodać do arkusza wizualizację tabeli oraz pola Region, Year i Month, jak również miary Sum dla sprzedaży i kosztów. Tabela będzie wyglądać następująco:

Tabela wynikowa

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

Ta funkcja skryptu zwraca liczbę całkowitą, która wskazuje numer oceny pojedynczego wiersza w instrukcji a **LOAD** z klauzulą **while**. Pierwsza iteracja ma numer 1. Funkcja **IterNo** jest przydatna tylko wtedy, gdy jest stosowana razem z klauzulą **while**.

Składnia:

```
IterNo( )
```

Przykłady i wyniki:

Przykład:

```
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
];
```

Ta instrukcja **LOAD** spowoduje wygenerowanie jednego rekordu na datę w ramach zakresu określonego przez wartości **StartDate** i **EndDate**.

W tym przykładzie otrzymana tabela będzie wyglądać następująco:

Tabela wynikowa

Day	Date
1	2014-01-22
2	2014-01-23
3	2014-01-24
4	2014-01-25
5	2014-01-26

RecNo

Ta funkcja skryptu zwraca liczbę całkowitą dla numeru aktualnie odczytanego wiersza bieżącej tabeli. Pierwszy wiersz ma numer 1.

Składnia:

```
RecNo( )
```

W odróżnieniu od funkcji **RowNo( )**, która zlicza wiersze w docelowej tabeli Qlik Sense, funkcja **RecNo( )** zlicza rekordy w tabeli danych nieprzetworzonych i jest resetowana, gdy tabela danych nieprzetworzonych jest konkatenowana z inną.

Przykład: Skrypt ładowania danych

Ładowanie tabeli danych nieprzetworzonych:

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

Ładowanie numerów rekordów i wierszy dla wybranych wierszy:

```
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;
```

Docelowa tabela wewnętrzna Qlik Sense:

Tabela wynikowa

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo

Ta funkcja zwraca liczbę całkowitą dla pozycji bieżącego wiersza w wynikowej tabeli wewnętrznej Qlik Sense. Pierwszy wiersz ma numer 1.

Składnia:

```
RowNo( [TOTAL])
```

W przeciwieństwie do funkcji **RecNo( )**, która zlicza rekordy w tabeli danych nieprzetworzonych, funkcja **RowNo( )** nie zlicza rekordów wykluczonych przez klauzule **where** i nie jest resetowana w przypadku konkatenacji tabeli danych nieprzetworzonych z inną tabelą.

*Jeśli używane jest wcześniejsze ładowanie, tzn. określona liczba skumulowanych instrukcji **LOAD** dokonuje odczytu z tej samej tabeli, wówczas funkcji **RowNo( )** można użyć tylko w najbardziej zewnętrznej instrukcji **LOAD** . Jeśli funkcja **RowNo( )** jest używana w kolejnych instrukcjach **LOAD** , zwracane jest 0.*

Przykład: Skrypt ładowania danych

Ładowanie tabeli danych nieprzetworzonych:

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

Ładowanie numerów rekordów i wierszy dla wybranych wierszy:

```
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;
```

Docelowa tabela wewnętrzna Qlik Sense:

Tabela wynikowa

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo – funkcja wykresu

Funkcja **RowNo()** zwraca numer bieżącego wiersza w bieżącym segmencie kolumn tabeli. W przypadku wykresów bitmapowych funkcja **RowNo()** zwraca numer bieżącego wiersza w tabeli prostej odpowiadającej wykresowi.

Jeśli tabela lub równoważnik tabeli zawiera wiele wymiarów pionowych, wówczas segment bieżącej kolumny będzie zawierał tylko wiersze z takimi samymi wartościami we wszystkich kolumnach wymiaru jak bieżący wiersz, ale bez kolumny przedstawiającej ostatni wymiar w kolejności sortowania między polami.

Segmenty kolumn

Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji.

Składnia:

```
RowNo([TOTAL])
```

Typ zwracanych danych: integer

Argumenty:

Argument	Opisu
TOTAL	Jeśli tabela jest jednowymiarowa lub jako argument zostanie podany kwalifikator **TOTAL**, bieżący segment kolumny jest zawsze równy całej kolumnie.

Przykład: Wyrażenie wykresu używające funkcji RowNo

Przykład – wyrażenie wykresu

Skrypt ładowania

Załaduj następujące dane w edytorze ładowania danych jako ładowanie wbudowane, aby utworzyć poniższe przykłady wyrażeń wykresu:

```
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 '|');
```

Wyrażenie wykresu

Utwórz wizualizację tabeli w arkuszu Qlik Sense z wymiarami **Customer** i **UnitSales**. Dodaj `RowNo( )` i `RowNo(TOTAL)` jako miary zatytułowane odpowiednio **Row in Segment** i **Row Number**. Dodaj w tabeli następujące wyrażenie jako miarę:

```
If( RowNo( )=1, 0, UnitSales / Above( UnitSales ))
```

Wynik

Customer	UnitSales	Row in Segment	Row Number	If(RowNo( )=1, 0, UnitSales / Above(UnitSales))
Astrida	4	1	1	0
Astrida	9	2	2	2.25
Astrida	10	3	3	1.1111111111111
Betacab	2	1	4	0
Betacab	5	2	5	2.5
Betacab	25	3	6	5
Canutility	4	1	7	0
Canutility	8	2	8	2

Customer	UnitSales	Row in Segment	Row Number	If(RowNo( )=1, 0, UnitSales / Above(UnitSales))
Divadip	1	1	9	0
Divadip	4	2	10	4

Objaśnienie

W kolumnie **Row in Segment** prezentowane są wyniki 1,2,3 dla segmentu kolumny z wartościami UnitSales dla klienta Astrida. Numerowanie wierszy rozpoczyna się wówczas ponownie od wartości 1 dla następnego segmentu kolumny, czyli dla Betacab.

Kolumna **Row Number** ignoruje wymiary ze względu na argument `TOTAL` dla `RowNo()` i służy do liczenia wierszy w tabeli.

Wyrażenie to zwraca wartość 0 dla pierwszego wiersza w każdym segmencie kolumny. W efekcie w kolumnie są wyświetlane następujące wartości:

0, 2,25, 1,1111111, 0, 2,5, 5, 0, 2, 0 i 4.

Zob. także:

p *Above – funkcja wykresu (page 1253)*

5.7 Funkcje daty i czasu

Funkcje daty i godziny Qlik Sense umożliwiają przekształcanie i konwertowanie wartości daty i godziny. Wszystkie funkcje mogą być stosowane zarówno w skryptach ładowania danych, jak i wyrażeniach wykresu.

Funkcje te bazują na liczbie kolejnej daty i godziny, wyrażającej liczbę dni od 30 grudnia 1899 r. Wartość całkowita reprezentuje datę, a wartość ułamkowa godzinę.

W aplikacji Qlik Sense używana jest numeryczna wartość parametru, dlatego numer jest prawidłowym parametrem także wówczas, gdy nie jest sformatowany jako data lub godzina. Jeśli parametr nie odpowiada wartości numerycznej, na przykład dlatego, że jest ciągiem znaków, aplikacja Qlik Sense podejmuje próbę interpretacji ciągu znaków zgodnie ze zmiennymi środowiska daty i godziny.

Jeśli używany w parametrze format czasu nie odpowiada formatowi ustawionemu w zmiennych środowiska, aplikacja Qlik Sense nie dokona prawidłowej interpretacji. W takim przypadku trzeba zmienić ustawienia lub użyć funkcji interpretacji.

W przykładach dla każdej funkcji jako domyślne formaty godziny i daty przyjmuje się hh:mm:ss iYYYY-MM-DD (ISO 8601).

Podczas przetwarzania znacznika czasu przy użyciu funkcji daty lub godziny program Qlik Sense ignoruje wszelkie parametry czasu letniego, chyba że funkcja daty lub godziny uwzględnia położenie geograficzne.

Na przykład w przypadku zapisu `ConvertToLocalTime( filetime('Time.qvd'), 'Paris')` *czas letni będzie stosowany, a w przypadku zapisu* `ConvertToLocalTime(filetime('Time.qvd'), 'GMT-01:00')` *parametry czasu letniego nie będą stosowane.*

Przegląd funkcji daty i godziny

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

Wyrażenia godziny w liczbach całkowitych

second

Ta funkcja zwraca liczbę całkowitą reprezentującą sekundy, gdy ułamek **expression** jest interpretowany jako czas zgodnie ze standardową interpretacją liczb.

```
second (expression)
```

minute

Ta funkcja zwraca liczbę całkowitą reprezentującą minuty, gdy ułamek **expression** jest interpretowany jako czas zgodnie ze standardową interpretacją liczb.

```
minute (expression)
```

hour

Ta funkcja zwraca liczbę całkowitą reprezentującą godzinę, gdy ułamek wyrażenia **expression** jest interpretowany jako czas zgodnie ze standardową interpretacją liczb.

```
hour (expression)
```

day

Ta funkcja zwraca liczbę całkowitą reprezentującą dzień, gdy ułamek wyrażenia **expression** jest interpretowany jako data zgodnie ze standardową interpretacją liczb.

```
day (expression)
```

week

Ta funkcja zwraca liczbę całkowitą reprezentującą numer tygodnia zgodnie z normą ISO 8601. Numer miesiąca jest obliczany na podstawie interpretacji daty z wyrażenia zgodnie ze standardową interpretacją liczb.

```
week (expression)
```

month

Ta funkcja zwraca wartość podwójną z nazwą miesiąca, jak to zostało określone w zmiennej środowiskowej **MonthNames**, oraz liczbę całkowitą z przedziału 1-12. Miesiąc jest obliczany na podstawie interpretacji daty z wyrażenia zgodnie ze standardową interpretacją liczb.

```
month (expression)
```

year

Ta funkcja zwraca liczbę całkowitą reprezentującą rok, gdy wyrażenie **expression** jest interpretowane jako data zgodnie ze standardową interpretacją liczb.

```
year (expression)
```

weekyear

Ta funkcja zwraca rok, do którego należy numer tygodnia zgodnie z normą ISO 8601. Numer tygodnia należy do zakresu od 1 do około 52.

```
weekyear (expression)
```

weekday

Ta funkcja zwraca wartość podwójną:

- Nazwę dnia zdefiniowaną w zmiennej środowiskowej **DayNames**.
- Wartość całkowitą od 0 do 6, która odpowiada nominalnemu dniu tygodnia (0-6).

```
weekday (date)
```

Funkcje Timestamp

now

Ta funkcja zwraca znacznik bieżącego czasu. Funkcja zwraca wartości w formacie zmiennej systemowej **TimeStamp**. Wartością domyślną **timer_mode** jest 1.

```
now ([ timer_mode])
```

today

Ta funkcja zwraca bieżącą datę. Funkcja zwraca wartości w formacie zmiennej systemowej `DateFormat`.

```
today ([timer_mode])
```

LocalTime

Ta funkcja zwraca znacznik bieżącego czasu dla podanej strefy czasowej.

```
localtime ([timezone [, ignoreDST ]])
```

Funkcje Make

makedate

Ta funkcja zwraca datę obliczoną na podstawie parametrów rok **YYYY**, miesiąc **MM** i dzień **DD**.

```
makedate (YYYY [ , MM [ , DD ] ])
```

makeweekdate

Ta funkcja zwraca datę obliczoną na podstawie parametrów rok **YYYY**, tydzień **WW** i dzień tygodnia **D**.

```
makeweekdate (YYYY [ , WW [ , D ] ])
```

maketime

Ta funkcja zwraca godzinę obliczoną na podstawie parametrów godzina **hh**, minuta **mm** i sekunda **ss**.

```
maketime (hh [ , mm [ , ss [ .fff ] ] ])
```

Inne funkcje daty

AddMonths

Ta funkcja zwraca datę występującą **n** miesięcy po dacie **startdate** lub, jeśli **n** jest wartością ujemną, **n** miesięcy przed datą **startdate**.

```
addmonths (startdate, n , [ , mode])
```

AddYears

Ta funkcja zwraca datę występującą **n** lat po dacie **startdate** lub, jeśli **n** jest wartością ujemną, **n** lat przed datą **startdate**.

```
addyears (startdate, n)
```

yeartodate

Ta funkcja sprawdza, czy wejściowy znacznik czasu przypada w roku daty dnia, w którym skrypt został ostatnio załadowany, i zwraca True, jeśli przypada, a False, jeśli nie przypada.

```
yeartodate (date [ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Funkcje Timezone

timezone

Ta funkcja zwraca strefę czasową zdefiniowaną na komputerze, na którym działa aparat Qlik.

```
timezone ( )
```

GMT

Ta funkcja zwraca aktualny czas Greenwich Mean Time ustalony na podstawie ustawień regionalnych.

```
GMT ( )
```

UTC

Zwraca bieżącą wartość Coordinated Universal Time.

```
UTC ( )
```

daylightsaving

Zwraca aktualną korektę czasu letniego zdefiniowaną w systemie Windows.

```
daylightsaving ( )
```

converttolocaltime

Przekształca znacznik czasu UTC lub GMT na czas lokalny w postaci wartości podwójnej. Parametr place może być dowolnym miastem, miejscem albo strefą czasową z całego świata.

```
converttolocaltime (timestamp [, place [, ignore_dst=false]])
```

Funkcje Set time

setdateyear

Ta funkcja przyjmuje jako dane wejściowe wartości z pól **timestamp** i **year** i aktualizuje pole **timestamp** wartością **year** określoną w danych wejściowych.

```
setdateyear  (timestamp, year)
```

setdateyearmonth

Ta funkcja przyjmuje jako dane wejściowe wartości z pól **timestamp**, **month** i **year** i aktualizuje pole **timestamp** wartościami **year** i **month** określonymi w danych wejściowych.

```
setdateyearmonth  (timestamp, year, month)
```

Funkcje In...

inyear

Ta funkcja zwraca wartość True, jeśli znacznik czasu **timestamp** należy do roku zawierającego wartość **base_date**.

```
inyear (date, basedate , shift [, first_month_of_year = 1])
```

inyeartodate

Ta funkcja zwraca wartość True, jeśli wartość **timestamp** należy do części roku zawierającego wartość **base_date**, z dokładnością do ostatniej milisekundy wartości **base_date** włącznie.

```
inyeartodate  (date, basedate , shift [, first_month_of_year = 1])
```

inquarter

Ta funkcja zwraca wartość True, jeśli wartość **timestamp** należy do kwartału zawierającego wartość **base_date**.

```
inquarter  (date, basedate , shift [, first_month_of_year = 1])
```

inquartertodate

Ta funkcja zwraca wartość True, jeśli wartość **timestamp** należy do części kwartału zawierającego wartość **base_date**, z dokładnością do ostatniej milisekundy wartości **base_date** włącznie.

```
inquartertodate  (date, basedate , shift [, first_month_of_year = 1])
```

inmonth

Ta funkcja zwraca wartość True, jeśli wartość **timestamp** należy do miesiąca zawierającego wartość **base_date**.

```
inmonth (date, basedate , shift)
```

inmonthtodate

Zwraca wartość True, jeśli wartość **date** należy do części miesiąca zawierającego wartość **basedate** z dokładnością do jednej milisekundy wartości **basedate** włącznie.

```
inmonthtodate (date, basedate , shift)
```

inmonths

Ta funkcja sprawdza, czy znacznik czasu mieści się w tym samym okresie miesięcznym, dwumiesięcznym, kwartalnym, czteromiesięcznym lub półrocznym, jako data bazowa. Można także wyszukać, czy znacznik czasu przypada w okresie poprzednim lub następnym.

```
inmonths (n, date, basedate , shift [, first_month_of_year = 1])
```

inmonthstodate

Ta funkcja wyszukuje, czy znacznik czasu przypada w części okresu miesiąca, dwóch miesięcy, kwartału, czterech miesięcy lub półrocza z dokładnością do jednej milisekundy wartości `base_date` włącznie. Można także wyszukać, czy znacznik czasu przypada w okresie poprzednim lub następnym.

```
inmonthstodate (n, date, basedate , shift [, first_month_of_year = 1])
```

inweek

Ta funkcja zwraca wartość True, jeśli wartość **timestamp** należy do tygodnia zawierającego wartość **base_date**.

```
inweek (date, basedate , shift [, weekstart])
```

inweektodate

Ta funkcja zwraca wartość True, jeśli wartość **timestamp** należy do części tygodnia zawierającego wartość **base_date**, z dokładnością do ostatniej milisekundy wartości **base_date** włącznie.

```
inweektodate (date, basedate , shift [, weekstart])
```

inlunarweek

Ta funkcja sprawdza, czy wartość **timestamp** należy do tygodnia księżycowego zawierającego wartość **base_date**. Tygodnie księżycowe w Qlik Sense są zdefiniowane przez uznanie 1 stycznia za pierwszy dzień tygodnia. Każdy tydzień, z wyjątkiem ostatniego tygodnia roku, będzie zawierał dokładnie siedem dni.

```
inlunarweek (date, basedate , shift [, weekstart])
```

inlunarweektodate

Ta funkcja sprawdza, czy wartość **timestamp** należy do części tygodnia księżycowego do ostatniej milisekundy wartości **base_date** włącznie. Tygodnie księżycowe w Qlik Sense są zdefiniowane przez uznanie 1 stycznia za pierwszy dzień tygodnia i każdy tydzień, z wyjątkiem ostatniego tygodnia roku, będzie zawierał dokładnie siedem dni.

```
inlunarweektodate (date, basedate , shift [, weekstart])
```

inday

Ta funkcja zwraca wartość True, jeśli znacznik czasu **timestamp** należy do dnia zawierającego wartość **base_timestamp**.

```
inday (timestamp, basetimestamp , shift [, daystart])
```

indaytotime

Ta funkcja zwraca wartość True, jeśli wartość **timestamp** należy do części dnia zawierającego wartość **base_timestamp**, z dokładnością do jednej milisekundy wartości **base_timestamp** włącznie.

```
indaytotime (timestamp, basetimestamp , shift [, daystart])
```

Funkcje Start ... end

yearstart

Ta funkcja zwraca znacznik czasu odpowiadający rozpoczęciu się pierwszego dnia roku zawierającego wartość **date**. Domyślnym formatem wyjściowym będzie format **DateFormat** skonfigurowany w skrypcie.

```
yearstart ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearend

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu ostatniej milisekundy ostatniego dnia roku zawierającego wartość **date**. Domyślnym formatem wyjściowym będzie format **DateFormat** skonfigurowany w skrypcie.

```
yearend ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearname

Ta funkcja zwraca rok w zapisie czterocyfrowym jako wartość wyświetlaną z bazową wartością liczbową odpowiadającą znacznikowi czasu pierwszej milisekundy pierwszego dnia roku zawierającego datę **date**.

```
yearname (date [, shift = 0 [, first_month_of_year = 1]] )
```

quarterstart

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu pierwszej milisekundy kwartału zawierającego wartość **date**. Domyślnym formatem wyjściowym będzie format **DateFormat** skonfigurowany w skrypcie.

```
quarterstart (date [, shift = 0 [, first_month_of_year = 1]])
```

quarterend

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu ostatniej milisekundy kwartału zawierającego wartość **date**. Domyślnym formatem wyjściowym będzie format **DateFormat** skonfigurowany w skrypcie.

```
quarterend (date [, shift = 0 [, first_month_of_year = 1]])
```

quartername

Ta funkcja zwraca wartość pokazującą miesiące kwartału (sformatowane zgodnie ze stosowaną w skryptach zmienną **MonthNames**) oraz rok z bazową wartością liczbową odpowiadającą znacznikowi czasu pierwszej milisekundy pierwszego dnia tego kwartału.

```
quartername (date [, shift = 0 [, first_month_of_year = 1]])
```

monthstart

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu pierwszej milisekundy ostatniego dnia miesiąca zawierającego wartość **date**. Domyślnym formatem wyjściowym będzie format **DateFormat** skonfigurowany w skrypcie.

```
monthstart (date [, shift = 0])
```

monthend

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu ostatniej milisekundy ostatniego dnia miesiąca zawierającego wartość `date`. Domyślnym formatem wyjściowym będzie format `DateFormat` skonfigurowany w skrypcie.

```
monthend (date [, shift = 0])
```

monthname

Ta funkcja zwraca wartość pokazującą miesiąc (sformatowany zgodnie ze stosowaną w skryptach zmienną **MonthNames**) oraz rok z bazową wartością liczbową odpowiadającą znacznikowi czasu pierwszej milisekundy pierwszego dnia tego miesiąca.

```
monthname (date [, shift = 0])
```

monthsstart

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu pierwszej milisekundy okresu miesiąca, dwóch miesięcy, kwartału, czterech miesięcy lub półrocza, który zawiera datę bazową. Można także wyszukać znacznik czasu dla okresu poprzedniego lub następnego.Domyślnym formatem wyjściowym będzie format **DateFormat** skonfigurowany w skrypcie.

```
monthsstart (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsend

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu ostatniej milisekundy okresu miesiąca, dwóch miesięcy, kwartału, czterech miesięcy lub półrocza, który zawiera datę bazową. Można także wyszukać znacznik czasu dla okresu poprzedniego lub następnego.

```
monthsend (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsname

Ta funkcja zwraca wartość reprezentującą zakres miesięcy w okresie (sformatowany zgodnie ze stosowaną w skryptach zmienną **MonthNames**), a także rok. Bazowa wartość liczbowa odpowiada znacznikowi czasu pierwszej milisekundy okresu miesiąca, dwóch miesięcy, kwartału, czterech miesięcy lub półrocza, który zawiera datę bazową.

```
monthsname (n, date [, shift = 0 [, first_month_of_year = 1]])
```

weekstart

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu pierwszej milisekundy pierwszego dnia tygodnia kalendarzowego zawierającego wartość **date**. Domyślnym formatem wyjściowym będzie format **DateFormat** skonfigurowany w skrypcie.

```
weekstart (date [, shift = 0 [,weekoffset = 0]])
```

weekend

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu ostatniej milisekundy ostatniego dnia (niedzieli) tygodnia kalendarzowego, który zawiera wartość **date**. Domyślnym formatem wyjściowym będzie format **DateFormat** skonfigurowany w skrypcie.

```
weekend (date [, shift = 0 [,weekoffset = 0]])
```

weekname

Ta funkcja zwraca wartość pokazującą rok i numer tygodnia z bazową wartością liczbową odpowiadającą znacznikowi czasu pierwszej milisekundy pierwszego dnia tygodnia, który zawiera datę **date**.

```
weekname (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekstart

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu pierwszej milisekundy pierwszego dnia tygodnia księżycowego zawierającego wartość **date**. Tygodnie księżycowe w Qlik Sense są zdefiniowane przez uznanie 1 stycznia za pierwszy dzień tygodnia i każdy tydzień, z wyjątkiem ostatniego tygodnia roku, będzie zawierał dokładnie siedem dni.

```
lunarweekstart (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekend

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu ostatniej milisekundy ostatniego dnia tygodnia księżycowego zawierającego wartość **date**. Tygodnie księżycowe w Qlik Sense są zdefiniowane przez uznanie 1 stycznia za pierwszy dzień tygodnia i każdy tydzień, z wyjątkiem ostatniego tygodnia roku, będzie zawierał dokładnie siedem dni.

```
lunarweekend (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekname

Ta funkcja zwraca wartość pokazującą rok i numer tygodnia księżycowego odpowiadający znacznikowi czasu pierwszej milisekundy pierwszego dnia tygodnia księżycowego zawierającego wartość **date**. Tygodnie księżycowe w Qlik Sense są zdefiniowane przez uznanie 1 stycznia za pierwszy dzień tygodnia i każdy tydzień, z wyjątkiem ostatniego tygodnia roku, będzie zawierał dokładnie siedem dni.

```
lunarweekname (date [, shift = 0 [,weekoffset = 0]])
```

daystart

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu dla pierwszej milisekundy dnia zawartego w argumencie **time**. Domyślnym formatem wyjściowym będzie format **TimestampFormat** skonfigurowany w skrypcie.

```
daystart (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayend

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu dla ostatniej milisekundy dnia określonego przez parametr **time**. Domyślnym formatem wyjściowym będzie format **TimestampFormat** skonfigurowany w skrypcie.

```
dayend (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayname

Ta funkcja zwraca wartość pokazującą datę z bazową wartością liczbową odpowiadającą znacznikowi czasu dla pierwszej milisekundy dnia określonego przez parametr **time**.

```
dayname (timestamp [, shift = 0 [, dayoffset = 0]])
```

Funkcje numerowania dni

age

Funkcja **age** zwraca wiek w momencie określonym przez parametr **timestamp** (w liczbie ukończonych lat) osoby urodzonej w dniu **date_of_birth**.

```
age (timestamp, date_of_birth)
```

networkdays

Funkcja **networkdays** zwraca liczbę dni roboczych (poniedziałek-piątek) od **start_date** do **end_date** włącznie z uwzględnieniem opcjonalnych dni wolnych (**holiday**).

```
networkdays  (start:date, end_date {, holiday})
```

firstworkdate

Funkcja **firstworkdate** zwraca najpóźniejszą datę rozpoczęcia, gdy możliwe jest uzyskanie parametru **no_of_workdays** (poniedziałek-piątek) z końcem nie później niż w dniu określonym przez parametr **end_date** oraz z uwzględnieniem wszelkich opcjonalnie wyszczególnionych dni wolnych. Parametry **end_date** i **holiday** powinny być poprawnymi datami lub znacznikami czasu.

```
firstworkdate (end_date, no_of_workdays {, holiday} )
```

lastworkdate

Funkcja **lastworkdate** zwraca najwcześniejszą datę zakończenia, gdy możliwe jest uzyskanie parametru **no_of_workdays** (poniedziałek-piątek) z początkiem w dniu **start_date** z uwzględnieniem wszelkich opcjonalnie wyszczególnionych dni wolnych (**holiday**). Parametry **start_date** i **holiday** powinny być poprawnymi datami lub znacznikami czasu.

```
lastworkdate (start_date, no_of_workdays {, holiday})
```

daynumberofyear

Ta funkcja oblicza numer dnia roku, w którym przypada znacznik czasu. Obliczenie jest wykonywane od pierwszej milisekundy pierwszego dnia roku, ale pierwszy miesiąc może być przesunięty.

```
daynumberofyear (date[,firstmonth])
```

daynumberofquarter
Ta funkcja oblicza numer dnia kwartału, w którym przypada znacznik czasu. Ta funkcja służy do tworzenia kalendarza głównego.

```
daynumberofquarter (date[,firstmonth])
```

addmonths

Ta funkcja zwraca datę występującą **n** miesięcy po dacie **startdate** lub, jeśli **n** jest wartością ujemną, **n** miesięcy przed datą **startdate**.

Składnia:

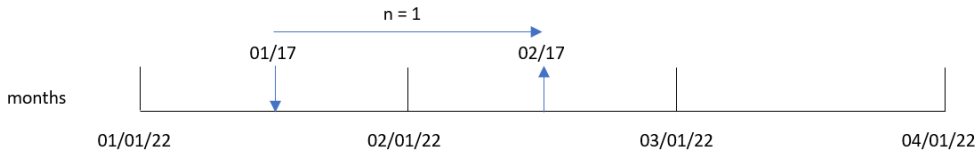
```
AddMonths(startdate, n , [ , mode])
```

Typ zwracanych danych: dual

Funkcja `addmonths()` dodaje lub odejmuje zdefiniowaną liczbę miesięcy, `n`, od `startdate` i zwraca datę otrzymaną w wyniku.

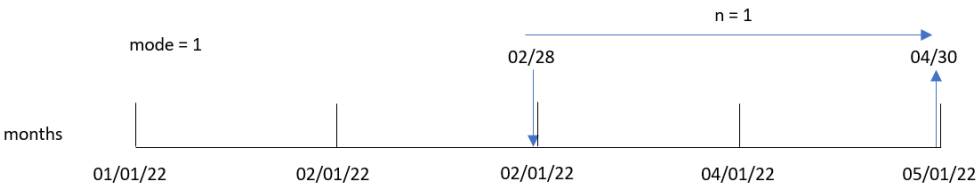
Argument `mode` ma wpływ na zwracane przez funkcję `startdate` wartości obejmujące dni od 28 do końca miesiąca. Ustawienie argumentu `mode` na 1 powoduje, że funkcja `addmonths()` zwraca datę odległą od końca miesiąca o tyle samo dni, co data `startdate`.

Przykładowy schemat przedstawiający działanie funkcji `addmonths()`



Na przykład 28 lutego jest ostatnim dniem miesiąca. Jeśli za pomocą funkcji `addmonths()`, z argumentem `mode` ustawionym na 1, przesuniemy datę o dwa miesiące do przodu, to otrzymamy ostatni dzień kwietnia, czyli 30 kwietnia.

Przykładowy schemat działania funkcji `addmonths()` z argumentem `mode=1`



Argumenty

Argument	Opis
startdate	Data rozpoczęcia jako znacznik czasu, na przykład „2012-10-12”.

Argument	Opis
n	Liczba miesięcy jako dodatnia lub ujemna liczba całkowita.
mode	Określa, czy miesiąc jest dodawany względem początku miesiąca, czy końca miesiąca. Trybem domyślnym jest 0 w przypadku operacji dodawania względem początku miesiąca. W przypadku operacji dodawania względem końca miesiąca należy ustawić tryb na 1. Jeśli tryb jest ustawiony na 1, a datą wejściową jest 28. lub data późniejsza, wówczas funkcja sprawdza, ile dni zostało do osiągnięcia końca miesiąca od daty startdate. Taka sama liczba dni do końca miesiąca jest ustawiana w zwróconej dacie.

Kiedy używać

Funkcja `addmonths()` może być używana w różnych wyrażeniach, w których trzeba znaleźć datę późniejszą lub wcześniejszą o określoną liczbę miesięcy od podanej.

Na przykład, za pomocą funkcji `addmonths()` można znaleźć datę zakończenia obowiązywania umowy na usługi telefonii komórkowej.

Przykłady funkcji

Przykład	Wynik
`addmonths ('01/29/2003' ,3)`	Zwraca wartość „04/29/2003”.
`addmonths ('01/29/2003',3,0)`	Zwraca wartość „04/29/2003”.
`addmonths ('01/29/2003',3,1)`	Zwraca wartość „04/28/2003”.
`addmonths ('01/29/2003',1,0)`	Zwraca wartość „02/28/2003”.
`addmonths ('01/29/2003',1,1)`	Zwraca wartość „02/26/2003”.
`addmonths ('02/28/2003',1,0)`	Zwraca wartość „03/28/2003”.
`addmonths ('02/28/2003',1,1)`	Zwraca wartość „03/31/2003”.
`addmonths ('01/29/2003',-3)`	Zwraca wartość „10/29/2002”.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za lata 2020-2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Tworzenie pola, `two_months_later`, które zwraca datę o dwa miesiące późniejszą od daty zawarcia transakcji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

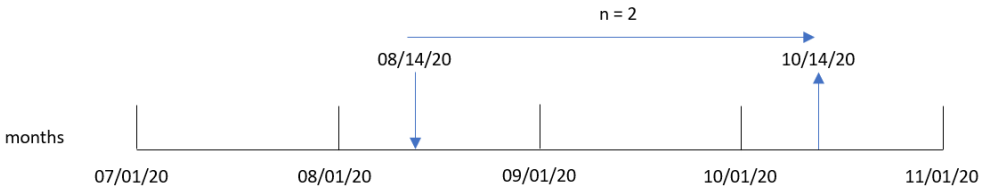
- `date`
- `two_months_later`

Tabela wynikowa

date	two_months_later
01/10/2020	03/10/2020
02/28/2020	04/28/2020
04/09/2020	06/09/2020
04/16/2020	06/16/2020
05/21/2020	07/21/2020
08/14/2020	10/14/2020
10/07/2020	12/07/2020
12/05/2020	02/05/2021
01/22/2021	03/22/2021
02/03/2021	04/03/2021
03/17/2021	05/17/2021
04/23/2021	06/23/2021
05/04/2021	07/04/2021
06/30/2021	08/30/2021
07/26/2021	09/26/2021
12/27/2021	02/27/2022
02/02/2022	04/02/2022
02/26/2022	04/26/2022
03/07/2022	05/07/2022
03/11/2022	05/11/2022

Pole `two_months_later` jest tworzone w poprzedniej instrukcji load przy użyciu funkcji `addmonths()`. Pierwszy podany argument określa datę wyjściową. Drugi reprezentuje liczbę miesięcy, jaka ma zostać dodana do daty `startdate` lub od niej odjęta. W tym przypadku przekazano wartość 2.

Diagram przedstawiający przykład użycia funkcji `addmonths()` bez dodatkowych argumentów



Transakcja 8193 miała miejsca 14 sierpnia. W związku z tym funkcja `addmonths()` dla pola `two_months_later` zwraca datę 14 października 2020 r.

Przykład 2 – względny koniec miesiąca

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zbiór transakcji na koniec miesiąca w 2022 roku, który jest ładowany do tabeli o nazwie `Transactions`.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Tworzenie pola, `relative_two_months_prior`, które zwraca względną w odniesieniu do końca miesiąca datę o dwa miesiące wcześniejszą od daty zawarcia transakcji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

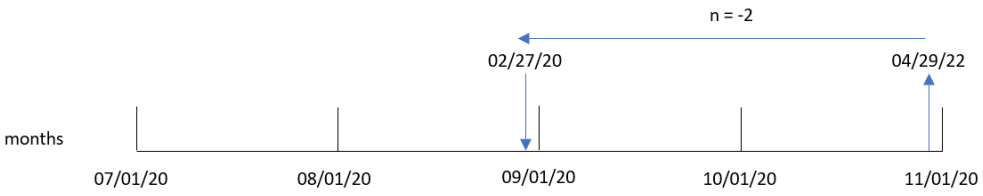
- `date`
- `relative_two_months_prior`

Tabela wynikowa

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

Pole `relative_two_months_prior` jest tworzone w poprzedniej instrukcji load przy użyciu funkcji `addmonths()`. Pierwszy podany argument określa datę wyjściową. Drugi reprezentuje liczbę miesięcy, jaka ma zostać dodana do daty `startdate` lub od niej odjęta. W tym przypadku przekazano wartość -2. Ostatni argument to mode, któremu nadano wartość 1, co zmusza funkcję do obliczenia daty względnej w odniesieniu do końca miesiąca dla wszystkich dat od 28. do ostatniego dnia miesiąca.

Diagram funkcji `addmonths()`, przykład z `n=-2`



Transakcja 8191 ma miejsce 29 kwietnia 2022 r. Wstępnie cofnięcie o dwa miesiące oznacza ustawienie daty na luty. Następnie, ze względu na to, że przypisano trzeciemu argumentowi wartość 1 oraz na to, że data jest późniejsza niż 27. dzień miesiąca, funkcja oblicza wartość względną w odniesieniu do końca miesiąca. Funkcja stwierdza, że 29. jest przedostatnim dniem kwietnia, w związku z czym zwraca przedostatni dzień lutego, czyli 27.

Przykład 3 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenia zwracające datę o dwa miesiące późniejszą od daty zawarcia transakcji są tworzone jako miara w obiekcie wykresu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Utwórz następującą miarę:

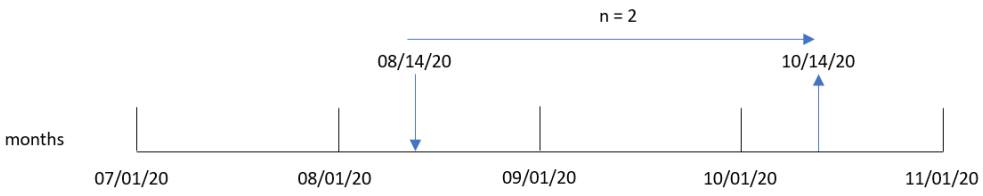
```
=addmonths(date,2)
```

Tabela wynikowa

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

Miara `two_months_later` jest tworzona w obiekcie wykresu przy użyciu funkcji `addmonths()`. Pierwszy podany argument określa datę wyjściową. Drugi reprezentuje liczbę miesięcy, jaka ma zostać dodana do daty `startdate` lub od niej odjęta. W tym przypadku przekazano wartość 2.

Diagram funkcji `addmonths()`, przykład obiektu wykresu



Transakcja 8193 miała miejsca 14 sierpnia. W związku z tym funkcja `addmonths()` zwraca datę 14 października 2020 r. dla pola `two_months_later`.

Przykład 4 – Scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Mobile_Plans`.
- Informacje obejmujące identyfikator umowy, datę rozpoczęcia, długość trwania umowy oraz miesięczną opłatę.

Użytkownik potrzebuje obiektu wykresu pokazującego datę zakończenia każdej umowy telefonicznej według identyfikatora umowy.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- contract_id
- start_date
- contract_length

Utwórz następującą miarę, aby obliczyć datę zakończenia każdej umowy:

```
=addmonths(start_date,contract_length, 0)
```

Tabela wynikowa

contract_id	start_date	contract_length	=addmonths(start_date,contract_length,0)
8188	01/13/2020	18	07/13/2021
8189	02/26/2020	24	02/26/2022
8190	03/27/2020	36	03/27/2023
8191	04/16/2020	24	04/16/2022
8192	05/21/2020	24	05/21/2022
8193	08/14/2020	12	08/14/2021
8194	10/07/2020	18	04/07/2022
8195	12/05/2020	12	12/05/2021
8196	01/22/2021	12	01/22/2022
8197	02/03/2021	18	08/03/2022
8198	03/17/2021	24	03/17/2023
8199	04/23/2021	24	04/23/2023
8200	05/04/2021	12	05/04/2022
8201	06/30/2021	12	06/30/2022
8202	07/26/2021	12	07/26/2022
8203	12/27/2021	36	12/27/2024
8204	06/06/2022	24	06/06/2024
8205	07/18/2022	12	07/18/2023
8206	11/14/2022	12	11/14/2023
8207	12/12/2022	18	06/12/2024

addyears

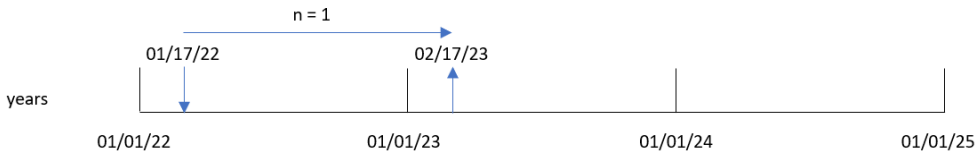
Ta funkcja zwraca datę występującą **n** lat po dacie **startdate** lub, jeśli **n** jest wartością ujemną, **n** lat przed datą **startdate**.

Składnia:

```
AddYears(startdate, n)
```

Typ zwracanych danych: dual

Przykładowy schemat przedstawiający działanie funkcji `addyears()`



Funkcja `addyears()` dodaje lub odejmuje zdefiniowaną liczbę lat, `n`, od `startdate`. Następnie zwraca datę wynikową.

Argumenty

Argument	Opis
startdate	Data rozpoczęcia jako znacznik czasu, na przykład „2012-10-12”.
n	Liczba lat jako dodatnia lub ujemna liczba całkowita.

Przykłady funkcji

Przykład	Wynik
`addyears ('01/29/2010',3)`	Zwraca wartość „01/29/2013”.
`addyears ('01/29/2010',-1)`	Zwraca wartość „01/29/2009”.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – prosty przykład

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za lata 2020-2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Tworzenie pola, `two_years_later`, które zwraca datę o dwa lata późniejszą od daty zawarcia transakcji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

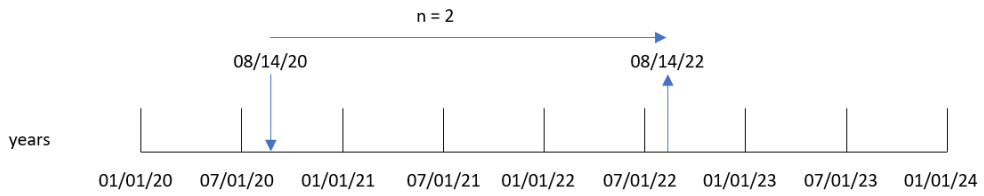
- `date`
- `two_years_later`

Tabela wynikowa

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

Pole `two_years_later` jest tworzone w poprzedniej instrukcji load przy użyciu funkcji `addyears()`. Pierwszy podany argument określa datę wyjściową. Drugi reprezentuje liczbę lat, jaka ma zostać dodana do daty początkowej lub od niej odjęta. W tym przypadku przekazano wartość 2.

Diagram funkcji `addyears()`, przykład podstawowy



Transakcja 8193 miała miejsca 14 sierpnia 2020 roku. W związku z tym funkcja `addyears()` zwraca datę 14 października 2022 r. dla pola `two_years_later`.

Przykład 2 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za lata 2020-2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).

W obiekcie wykresu utwórz miarę, `prior_year_date`, która zwraca datę o jeden rok wcześniejszą od daty zawarcia transakcji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Utwórz następującą miarę, aby obliczyć datę o rok wcześniejszą od daty zawarcia każdej transakcji:

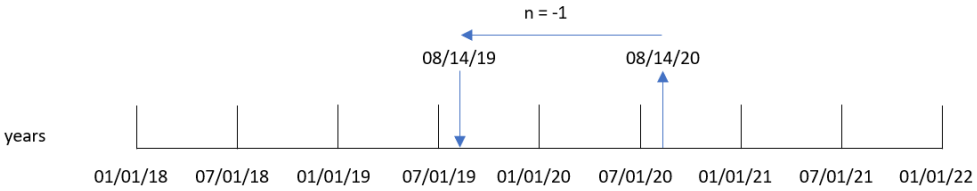
```
=addyears(date,-1)
```

Tabela wynikowa

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

Miara `one_year_prior` jest tworzona w obiekcie wykresu przy użyciu funkcji `addyears()`. Pierwszy podany argument określa datę wyjściową. Drugi reprezentuje liczbę lat, jaka ma zostać dodana do daty `startdate` lub od niej odjęta. W tym przypadku przekazano wartość -1.

Diagram funkcji `addyears()`, przykład obiektu wykresu



Transakcja 8193 miała miejsca 14 sierpnia. W związku z tym funkcja `addyears()` zwraca datę 14 sierpnia 2019 r. dla pola `one_year_prior`.

Przykład 3 – Scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Warranties`.
- Informacje obejmujące identyfikator produktu, datę zakupu, okres gwarancji i cenę zakupu.

Użytkownik potrzebuje obiektu wykresu pokazującego datę wygaśnięcia gwarancji każdego produktu według identyfikatorów produktów.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `product_id`
- `purchase_date`
- `warranty_length`

Utwórz następującą miarę, aby obliczyć datę zakończenia okresu gwarancji każdego produktu:

```
=addyears(purchase_date,warranty_length)
```

Tabela wynikowa

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

product_id	purchase_date	warranty_length	=addyears(purchase_date,warranty_length)
8204	06/06/2022	2	06/06/2024
8205	07/18/2022	1	07/18/2023
8206	11/14/2022	1	11/14/2023
8207	12/12/2022	4	12/12/2026

age

Funkcja **age** zwraca wiek w momencie określonym przez parametr **timestamp** (w liczbie ukończonych lat) osoby urodzonej w dniu **date_of_birth**.

Składnia:

```
age(timestamp, date_of_birth)
```

Może być wyrażeniem.

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
timestamp	Znacznik czasu lub wyrażenie, którego wynikiem jest znacznik czasu, do którego ma być obliczana ukończona liczba lat.
date_of_birth	Data urodzenia osoby, której wiek jest obliczany. Może być wyrażeniem.

Przykłady i wyniki:
W tych przykładach używany jest format daty **DD/MM/YYYY**. Format daty jest określony w instrukcji **SET DateFormat** u góry skryptu ładowania danych. Format zastosowany w przykładach można zmienić, aby dostosować go do konkretnych potrzeb.

Przykłady skryptów

Przykład	Wynik
`age('25/01/2014', '29/10/2012')`	Zwraca wartość 1.
`age('29/10/2014', '29/10/2012')`	Zwraca wartość 2.

Przykład:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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;
```

Tabela wynikowa przedstawia zwrócone wartości age dla każdego z rekordów w tabeli.

Tabela wynikowa

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

Przekształca znacznik czasu UTC lub GMT na czas lokalny w postaci wartości podwójnej. Parametr place może być dowolnym miastem, miejscem albo strefą czasową z całego świata.

Składnia:

```
ConvertToLocalTime(timestamp [, place [, ignore_dst=false]])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
timestamp	Znacznik czasu lub wyrażenie dające w wyniku znacznik czasu, które należy przekształcić.
place	Miejsce lub strefa czasowa z tabeli poprawnych miejsc i stref czasowych poniżej. W celu określenia czasu lokalnego można także użyć parametru GMT lub UTC. Prawidłowe są następujące wartości i zakresy przesunięcia czasu: • 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  *Można użyć wyłącznie standardowych przesunięć czasu. Nie można użyć dowolnego przesunięcia czasu, na przykład GMT-04:27.*
ignore_dst	Aby zignorować DST (czas letni), należy ustawić wartość True.

Otrzymany czas jest korygowany do czasu letniego, chyba że parametr **ignore_dst** ma wartość True.

Poprawne miejsca i strefy czasowe

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

A-C	D-K	L-R	S-Z
Auckland	Greenwich Mean Time : Dublin	Melbourne	Sofia
Azores	Guadalajara	Mexico City	Solomon Is.
Baghdad	Guam	Mid-Atlantic	Sri Jayawardenepura
Baku	Hanoi	Minsk	St. Petersburg
Bangkok	Harare	Monrovia	Stockholm
Beijing	Hawaii	Monterrey	Sydney
Belgrade	Helsinki	Moscow	Taipei
Berlin	Hobart	Mountain Time (US & Canada)	Tallinn
Bern	Hong Kong	Mumbai	Tashkent
Bogota	Indiana (East)	Muscat	Tbilisi
Brasilia	International Date Line West	Nairobi	Tehran
Bratislava	Irkutsk	New Caledonia	Tokyo
Brisbane	Islamabad	New Delhi	Urumqi
Brussels	Istanbul	Newfoundland	Warsaw
Bucharest	Jakarta	Novosibirsk	Wellington
Budapest	Jerusalem	Nuku'alofa	West Central Africa
Buenos Aires	Kabul	Osaka	Vienna
Cairo	Kamchatka	Pacific Time (US & Canada)	Vilnius
Canberra	Karachi	Paris	Vladivostok
Cape Verde Is.	Kathmandu	Perth	Volgograd
Caracas	Kolkata	Port Moresby	Yakutsk
Casablanca	Krasnoyarsk	Prague	Yerevan
Central America	Kuala Lumpur	Pretoria	Zagreb
Central Time (US & Canada)	Kuwait	Quito	-
Chennai	Kyiv	Riga	-
Chihuahua	-	Riyadh	-

A-C	D-K	L-R	S-Z
Chongqing	-	Rome	-
Copenhagen	-	-	-

Przykłady i wyniki:

Przykłady skryptów

Przykład	Wynik
`ConvertToLocalTime('2007-11-10 23:59:00','Paris')`	Zwraca wartość „2007-11-11 00:59:00” oraz odpowiadającą wewnętrzną reprezentację znacznika czasu.
`ConvertToLocalTime(UTC(), 'GMT-05:00')`	Zwraca czas dla wschodniego wybrzeża Ameryki Północnej, na przykład czas Nowego Jorku.
`ConvertToLocalTime(UTC(), 'GMT-05:00', True)`	Zwraca czas dla wschodniego wybrzeża Ameryki Północnej, na przykład czas Nowego Jorku, bez korekty do czasu letniego.

day

Ta funkcja zwraca liczbę całkowitą reprezentującą dzień, gdy ułamek wyrażenia **expression** jest interpretowany jako data zgodnie ze standardową interpretacją liczb.

Funkcja ta zwraca dzień miesiąca dla określonej daty. Jest powszechnie używana do wyprowadzania pola dnia jako części wymiaru kalendarza.

Składnia:

```
day(expression)
```

Typ zwracanych danych: integer

Przykłady funkcji

Przykład	Wynik
`day( 1971-10-12 )`	zwraca 12
`day( 35648 )`	zwraca 6, ponieważ 35648 = 1997-08-06

Przykład 1 – zbiór danych DateFormat (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Zbiór danych dat o nazwie `Master_Calendar`. Zmienna systemowa `DateFormat` jest ustawiona na DD/MM/RRRR.
- Poprzednie ładowanie, które tworzy dodatkowe pole, o nazwie `day_of_month`, używające funkcji `day()`.
- Dodatkowe pole, o nazwie `long_date`, używające funkcji `date()`, aby wyrazić pełną nazwę miesiąca.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `long_date`
- `day_of_month`

Tabela wynikowa

data	long_date	day_of_month
03/11/2022	11 marca 2022	11
03/12/2022	12 marca 2022	12
03/13/2022	13 marca 2022	13
03/14/2022	14 marca 2022	14
03/15/2022	15 marca 2022	15
03/16/2022	16 marca 2022	16
03/17/2022	17 marca 2022	17
03/18/2022	18 marca 2022	18
03/19/2022	19 marca 2022	19
03/20/2022	20 marca 2022	20
03/21/2022	21 marca 2022	21

Dzień miesiąca jest prawidłowo obliczany przez funkcję `day()` w skrypcie.

Przykład 2 – daty ANSI (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Zbiór danych dat o nazwie `Master_Calendar`. Zmienn The DateFormat system variable DD/MM/YYYY is used. Natomiast daty w zbiorze danych są w standardowym formacie daty ANSI.
- Poprzednie ładowanie, które tworzy dodatkowe pole, o nazwie `day_of_month`, używające funkcji `date()`.
- Dodatkowe pole, o nazwie `long_date`, używające funkcji `date()`, aby wyrazić datę z pełną nazwą miesiąca.

Skrypt ładowania

```
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
 ];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `long_date`
- `day_of_month`

Tabela wynikowa

data	long_date	day_of_month
03/11/2022	11 marca 2022	11
03/12/2022	12 marca 2022	12
03/13/2022	13 marca 2022	13
03/14/2022	14 marca 2022	14
03/15/2022	15 marca 2022	15
03/16/2022	16 marca 2022	16
03/17/2022	17 marca 2022	17
03/18/2022	18 marca 2022	18
03/19/2022	19 marca 2022	19
03/20/2022	20 marca 2022	20
03/21/2022	21 marca 2022	21

Dzień miesiąca jest prawidłowo obliczany przez funkcję `day()` w skrypcie.

Przykład 3 – niesformatowane daty (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Zbiór danych dat o nazwie `Master_Calendar`. Używana jest zmienna systemowa `DateFormat` DD/MM/RRRR.
- Poprzednie ładowanie, które tworzy dodatkowe pole, o nazwie `day_of_month`, używające funkcji `day()`.
- Pierwotna niesformatowana data o nazwie `unformatted_date`.
- Dodatkowe pole o nazwie `long_date`, używające `date()`, zostało użyte do konwersji numerycznej daty na sformatowane pole daty.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- unformatted_date
- long_date
- day_of_month

Tabela wynikowa

unformatted_date	long_date	day_of_month
44868	03 listopada 2022	3
44898	03 grudnia 2022	3
44928	02 stycznia 2023	2
44958	01 lutego 2023	1
44988	03 marca 2023	3
45008	23 marca 2023	23
45018	02 kwietnia 2023	2
45038	22 kwietnia 2023	22
45048	02-May- 2023	2
45068	22-May- 2023	22
45078	01 czerwca 2023	1

Dzień miesiąca jest prawidłowo obliczany przez funkcję `day()` w skrypcie.

Przykład 4 – Obliczanie miesiąca wygaśnięcia (wykres)

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Zbiór danych zamówień złożonych w marcu o nazwie `Orders`. Tabela zawiera trzy pola:
  - id
  - order_date
  - amount

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `order_date`.

Aby obliczyć datę dostawy, utwórz następującą miarę: `=day(order_date+5)`.

Tabela wynikowa

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

Funkcja `day()` prawidłowo stwierdza, że zamówienie złożone 11 marca zostanie dostarczone 16 marca przy założeniu pięciodniowego okresu dostawy.

dayend

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu dla ostatniej milisekundy dnia określonego przez parametr **time**. Domyślnym formatem wyjściowym będzie format **TimestampFormat** skonfigurowany w skrypcie.

Składnia:

```
DayEnd(time[, [period_no[, day_start]])
```

Kiedy używać

Funkcja `dayend()` jest powszechnie używana jako część wyrażenia, gdy użytkownik chce, by w obliczeniach użyto ułamka dnia, który jeszcze nie nastąpił. Na przykład, aby obliczyć całkowite wydatki do poniesienia w ciągu dnia.

Typ zwracanych danych: dual

Argumenty

Argument	Opis
time	Znacznik czasu do oceny.
period_no	Parametr **period_no** jest liczbą całkowitą lub wyrażeniem, którego wynikiem jest liczba całkowita, gdzie wartość 0 wskazuje dzień zawierający wartość **time**. Wartości ujemne parametru **period_no** oznaczają dni poprzednie, a wartości dodatnie – dni następne.
day_start	Aby określić, że dni nie zaczynają się o północy, należy wskazać przesunięcie w postaci części dnia w parametrze **day_start**. Na przykład 0,125 oznacza godzinę trzecią rano. Innymi słowy, aby utworzyć przesunięcie, podziel godzinę rozpoczęcia przez 24 godziny. Na przykład, jeśli dzień ma się zaczynać o godzinie 7:00, użyj ułamka 7/24.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
dayend('01/25/2013 16:45:00')	Returns 01/25/2013 23:59:59. PM
dayend('01/25/2013 16:45:00', -1)	Zwraca 01/24/2013 23:59:59. PM
dayend('25/01/2013 16:45:00', 0, 0.5)	Returns 01/26/2013 11:59:59. PM

Przykład 1 – prosty skrypt

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Zbiór danych zawierający listę dat, który jest załadowany do tabeli o nazwie „Calendar”.
- Domyślna `DateFormat` zmienna systemowa `(MM/DD/YYYY)`.
- Poprzedni ładunek mający na celu utworzenia dodatkowego pola, „`EOD_timestamp`”, przy użyciu funkcji `dayend()`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `EOD_timestamp`

Tabela wynikowa

data	EOD_timestamp
03/11/2022 1:47:15 AM	3/11/2022 11:59:59 PM
03/12/2022 4:34:58 AM	3/12/2022 11:59:59 PM
03/13/2022 5:15:55 AM	3/13/2022 11:59:59 PM
03/14/2022 9:25:14 AM	3/14/2022 11:59:59 PM
03/15/2022 10:06:54 AM	3/15/2022 11:59:59 PM
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

Jak widać w powyższej tabeli, dla każdej daty w naszym zbiorze danych został wygenerowany znacznik czasu końca dnia. Znacznik ten ma format zmiennej systemowej `TimestampFormat M/D/YYYY h:mm:ss [.fff] TT`.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Załadujesz zbiór danych zawierający rezerwacje usług do tabeli o nazwie „Services”.

Zbiór ten zawiera następujące pola:

- `service_id`
- `service_date`
- `amount`

Utworzysz dwa nowe pola w tabeli:

- `deposit_due_date`: Data, kiedy powinien zostać odebrany depozyt. Jest to koniec dnia na trzy dni przed `service_date`.
- `final_payment_due_date`: Data, kiedy powinna zostać otrzymana ostateczna płatność. Jest to koniec dnia siedem dni po `service_date`.

Dwa powyższe pola zostały utworzone w poprzednim ładowaniu przy użyciu funkcji `dayend()` i dostarczają one dwa pierwsze parametry: `time` i `period_no`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- service_date
- deposit_due_date
- final_payment_due_date

Tabela wynikowa

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
03/17/2022 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

Wartości nowych pól znajdują się w `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Jako że użyto funkcji `dayend()`, wszystkie wartości znacznika czasu oznaczają ostatnią milisekundę dnia.

Wartości terminu depozytu określają datę trzy dni przed datą usługi, ponieważ drugi argument przekazany do funkcji `dayend()` jest ujemny.

Wartości ostatecznego terminu płatności określają datę siedem dni po dacie usługi, ponieważ drugi argument przekazany do funkcji `dayend()` jest ujemny.

Przykład 3 – skrypt day_start

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania na nowej karcie poniżej.

Użyte w tym przykładzie zbiór danych i scenariusz są takie same, jak w poprzednim przykładzie.

Tak jak w poprzednim przykładzie, zostaną utworzone dwa nowe pola:

- `deposit_due_date`: Data, kiedy powinien zostać odebrany depozyt. Jest to koniec dnia na trzy dni przed `service_date`.
- `final_payment_due_date`: Data, kiedy powinna zostać otrzymana ostateczna płatność. Jest to koniec dnia siedem dni po `service_date`.

Twoja firma chce jednak pracować według zasady, zgodnie z którą dzień roboczy zaczyna się o godzinie 17:00 jednego dnia i kończy się o godzinie 17:00 następnego dnia. Dzięki temu może monitorować transakcje mające miejsce w tych godzinach roboczych.

Aby spełnić te wymogi, w poprzednim ładowaniu zostały utworzone dwa powyższe pola przy użyciu funkcji `dayend()` i stosowane są wszystkie trzy argumenty: `time`, `period_no` i `day_start`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- service_date
- deposit_due_date
- final_payment_due_date

Tabela wynikowa

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
03/17/2022 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

Choć daty pozostały takie same jak w przykładzie 2, teraz mają znacznik czasu ostatniej milisekundy przed godziną 17:00, ponieważ w trzecim argumencie, `day_start`, do funkcji `dayend()` przekazano wartość 17/24.

Przykład – przykład z wykresem

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Użyte w tym przykładzie zbiór danych i scenariusz są takie same, jak w dwóch poprzednich przykładach. Firma chce pracować według zasady, zgodnie z którą dzień roboczy zaczyna się o godzinie 17:00 jednego dnia i kończy się o godzinie 17:00 następnego dnia.

Tak jak w poprzednim przykładzie, zostaną utworzone dwa nowe pola:

- `deposit_due_date`: Data, kiedy powinien zostać odebrany depozyt. Jest to koniec dnia na trzy dni przed `service_date`.
- `final_payment_due_date`: Data, kiedy powinna zostać otrzymana ostateczna płatność. Jest to koniec dnia siedem dni po `service_date`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

`service_date`.

Aby utworzyć pole `deposit_due_date`, należy utworzyć następującą miarę:

`=dayend(service_date,-3,17/24).`

Następnie, aby utworzyć pole `final_payment_due_date`, należy utworzyć następującą miarę:

`=dayend(service_date,7,17/24).`

Tabela wynikowa

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

Wartości nowych pól znajdują się w `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Jako że użyto funkcji `dayend()`, wszystkie wartości znacznika czasu oznaczają ostatnią milisekundę dnia.

Wartości terminu płatności określają datę trzy dni przed datą usługi, ponieważ drugi argument przekazany do funkcji `dayend()` jest ujemny.

Wartości ostatecznego terminu płatności określają datę siedem dni po dacie usługi, ponieważ drugi argument przekazany do funkcji `dayend()` jest ujemny.

Daty mają znacznik czasu ostatniej milisekundy przed godziną 17:00, ponieważ w trzecim argumencie, `day_start`, do funkcji `dayend()` przekazano wartość 17/24.

daylightsaving

Zwraca aktualną korektę czasu letniego zdefiniowaną w systemie Windows.

Składnia:

```
DaylightSaving( )
```

Typ zwracanych danych: podwójny

Przykład:

```
daylightsaving( )
```

dayname

Ta funkcja zwraca wartość pokazującą datę z bazową wartością liczbową odpowiadającą znacznikowi czasu dla pierwszej milisekundy dnia określonego przez parametr **time**.

Składnia:

```
DayName(time[, period_no [, day_start]])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
time	Znacznik czasu do oceny.
period_no	Parametr **period_no** jest liczbą całkowitą lub wyrażeniem, którego wynikiem jest liczba całkowita, gdzie wartość 0 wskazuje dzień zawierający wartość **time**. Wartości ujemne parametru **period_no** oznaczają dni poprzednie, a wartości dodatnie – dni następne.
day_start	Aby określić, że dni nie zaczynają się o północy, należy wskazać przesunięcie w postaci części dnia w parametrze **day_start**. Na przykład 0,125 oznacza godzinę trzecią rano.

Przykłady i wyniki:
W tych przykładach używany jest format daty **DD/MM/YYYY**. Format daty jest określony w instrukcji **SET DateFormat** u góry skryptu ładowania danych. Format zastosowany w przykładach można zmienić, aby dostosować go do konkretnych potrzeb.

Przykłady skryptów

Przykład	Wynik
`dayname('25/01/2013 16:45:00')`	Zwraca wartość 25/01/2013.
`dayname('25/01/2013 16:45:00', -1)`	Zwraca wartość 24/01/2013.
`dayname('25/01/2013 16:45:00', 0, 0.5 )`	Zwraca wartość 25/01/2013. Wyświetlany pełny znacznik czasu przedstawia bazową wartość liczbową odpowiadającą 25/01/2013 12:00:00.000..

Przykład:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

Na tym przykładzie nazwa dnia jest tworzona ze znacznika czasu oznaczającego początek dnia po każdej dacie faktury w tabeli.

```
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;
```

Otrzymana tabela zawiera pierwotne daty i kolumnę ze zwracaną wartością funkcji dayname(). Można wyświetlić pełny znacznik czasu, określając formatowanie w panelu właściwości.

Tabela wynikowa

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

Ta funkcja oblicza numer dnia kwartału, w którym przypada znacznik czasu. Ta funkcja służy do tworzenia kalendarza głównego.

Składnia:

```
DayNumberOfQuarter(timestamp[,start_month])
```

Typ zwracanych danych: integer

Argumenty

Argument	Opis
timestamp	Data lub znacznik czasu do oszacowania.
start_month	Przez określenie wartości **start_month** z zakresu od 2 do 12 (1 w przypadku pominięcia) można przesunąć początek roku do pierwszego dnia dowolnego miesiąca. Na przykład w celu określenia początku roku obrotowego na 1 marca należy podać wartość **start_month** = 3.

W tych przykładach używany jest format daty **DD/MM/YYYY**. Format daty jest określony w instrukcji **SET DateFormat** u góry skryptu ładowania danych. Format zastosowany w przykładach można zmienić, aby dostosować go do konkretnych potrzeb.

Przykłady funkcji

Przykład	Wynik
`DayNumberOfQuarter('12/09/2014')`	Zwraca 74, numer dnia w bieżącym kwartale.
`DayNumberOfQuarter ('12/09/2014',3)`	Zwraca 12, numer dnia w bieżącym kwartale. W tym przypadku pierwszy kwartał zaczyna się od marca (ponieważ wartość start_month określono jako 3). Oznacza to, że bieżącym kwartałem jest trzeci kwartał, który rozpoczął się 1 września.

Przykład 1 – styczeń, początek roku (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Prosty zbiór danych zawierający listę dat, który jest załadowany do tabeli o nazwie `Calendar`. Została użyta domyślna zmienna systemowa `DateFormat` MM/DD/RRRR.

- Poprzednie ładowanie, które tworzy dodatkowe pole, o nazwie `DayNrQtr`, używające funkcji `DayNumberOfQuarter()`.

Oprócz daty do funkcji nie przekazano żadnych innych parametrów.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `daynrqtr`

Tabela wynikowa

data	daynrqtr
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
02/28/2022	59
03/01/2022	61

data	daynrqtr
03/31/2022	91
04/01/2022	1

Pierwszym dniem roku jest 1 stycznia, ponieważ do funkcji `DayNumberOfQuarter()` nie przekazano drugiego argumentu.

1 stycznia to pierwszy dzień kwartału, natomiast 1 lutego to 32. dzień kwartału. 31 marca to 91. i ostatni dzień kwartału, natomiast 1 kwietnia to pierwszy dzień drugiego kwartału.

Przykład 2 – luty, początek roku (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Ten sam zbiór danych, co w pierwszym przykładzie.
- Została użyta domyślna zmienna systemowa `DateFormat` MM/DD/RRRR.
- Argument `start_month` oznacza początek pierwszego lutego. W ten sposób początek roku obrotowego został ustawiony na 1 lutego.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `daynrqtr`

Tabela wynikowa

data	daynrqtr
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	1
02/10/2022	10
02/28/2022	28
03/01/2022	30
03/31/2022	60
04/01/2022	61

Pierwszym dniem roku jest 1 lutego, ponieważ w drugim argumencie do funkcji `DayNumberOfQuarter()` przekazano wartość 2.

Pierwszy kwartał roku obejmuje miesiące od lutego do kwietnia, a czwarty – od listopada do stycznia. Widać to w tabeli wyników, w której 1 lutego jest pierwszym dniem kwartału, a 31 stycznia jest 92. i ostatnim dniem kwartału.

Przykład 3 – styczeń, początek roku (wykres)

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Ten sam zbiór danych, co w pierwszym przykładzie.
- Została użyta domyślna zmienna systemowa `DateFormat` MM/DD/RRRR.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Wartość dnia kwartału jest obliczana przez miarę w obiekcie wykresu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Utwórz następującą miarę:

```
=daynumberofquarter(date)
```

Tabela wynikowa

data	=daynumberofquarter(date)
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
02/28/2022	59
03/01/2022	61
03/31/2022	91
04/01/2022	1

Pierwszym dniem roku jest 1 stycznia, ponieważ do funkcji `DayNumberOfQuarter()` nie przekazano drugiego argumentu.

1 stycznia to pierwszy dzień kwartału, natomiast 1 lutego to 32 dzień kwartału. 31 marca to 91. i ostatni dzień kwartału, natomiast 1 kwietnia to pierwszy dzień drugiego kwartału.

Przykład 4 – luty, początek roku (wykres)

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Ten sam zbiór danych, co w pierwszym przykładzie.
- Została użyta domyślna zmienna systemowa `DateFormat` MM/DD/RRRR.
- Rok obrotowy trwa od 1 lutego do 31 stycznia.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Wartość dnia kwartału jest obliczana przez miarę w obiekcie wykresu.

Skrypt ładowania

```
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
];
```

Obiekt wykresu

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Utwórz następującą miarę:

```
=daynumberofquarter(date,2)
```

Wyniki

Tabela wynikowa

data	=daynumberofquarter(date,2)
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	1
02/10/2022	10
02/28/2022	28
03/01/2022	30
03/31/2022	60
04/01/2022	61

Pierwszym dniem roku jest 1 stycznia, ponieważ w drugim argumencie do funkcji `DayNumberOfQuarter()` przekazano wartość 2.

Pierwszy kwartał roku obejmuje miesiące od lutego do kwietnia, a czwarty – od listopada do stycznia. Widać to w tabeli wyników, w której 1 lutego jest pierwszym dniem kwartału, a 31 stycznia jest 92. i ostatnim dniem kwartału.

daynumberofyear

Ta funkcja oblicza numer dnia roku, w którym przypada znacznik czasu. Obliczenie jest wykonywane od pierwszej milisekundy pierwszego dnia roku, ale pierwszy miesiąc może być przesunięty.

Składnia:

```
DayNumberOfYear(timestamp[,start_month])
```

Typ zwracanych danych: integer

Argumenty

Argument	Opis
timestamp	Data lub znacznik czasu do oszacowania.
start_month	Przez określenie wartości **start_month** z zakresu od 2 do 12 (1 w przypadku pominięcia) można przesunąć początek roku do pierwszego dnia dowolnego miesiąca. Na przykład w celu określenia początku roku obrotowego na 1 marca należy podać wartość **start_month** = 3.

W tych przykładach używany jest format daty **DD/MM/YYYY**. Format daty jest określony w instrukcji **SET DateFormat** u góry skryptu ładowania danych. Format zastosowany w przykładach można zmienić, aby dostosować go do konkretnych potrzeb.

Przykłady funkcji

Przykład	Wynik
`DayNumberOfYear( '12/09/2014' )`	Zwraca 256, numer dnia liczony od pierwszego dnia roku.
`DayNumberOfYear( '12/09/2014',3 )`	Zwraca 196, numer dnia liczony od 1 marca.

Przykład 1 – styczeń, początek roku (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Prosty zbiór danych zawierający listę dat, który jest załadowany do tabeli o nazwie `Calendar`. Została użyta domyślna zmienna systemowa `DateFormat` MM/DD/RRRR.
- Poprzednie ładowanie, które tworzy dodatkowe pole, o nazwie `daynryear`, używające funkcji `DayNumberOfYear()`.

Oprócz daty do funkcji nie przekazano żadnych innych parametrów.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `daynryear`

Tabela wynikowa

data	daynryear
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
06/30/2022	182
07/26/2022	208
10/31/2022	305
11/01/2022	306
12/31/2022	366

Pierwszym dniem roku jest 1 stycznia, ponieważ do funkcji `DayNumberOfYear()` nie przekazano drugiego argumentu.

1 stycznia to pierwszy dzień kwartału, natomiast 1 lutego to 32 dzień roku. 30 czerwca to 182., a 31 grudnia to 366. i ostatni dzień roku.

Przykład 2 – listopad, początek roku (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Ten sam zbiór danych, co w pierwszym przykładzie.
- Została użyta domyślna zmienna systemowa `DateFormat` MM/DD/RRRR
- Argument `start_month` oznacza początek 1 listopada. W ten sposób początek roku obrotowego został ustawiony na 1 listopada.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `daynryear`

Tabela wynikowa

data	daynryear
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	93
02/10/2022	102
06/30/2022	243
07/26/2022	269
10/31/2022	366
11/01/2022	1
12/31/2022	61

Pierwszym dniem roku jest 1 listopada, ponieważ w drugim argumencie do funkcji `DayNumberOfYear()` przekazano wartość 11.

1 stycznia to pierwszy dzień kwartału, natomiast 1 lutego to 32 dzień roku. 30 czerwca to 182., a 31 grudnia to 366. i ostatni dzień roku.

Przykład 3 – styczeń, początek roku (wykres)

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Ten sam zbiór danych, co w pierwszym przykładzie.
- Została użyta domyślna zmienna systemowa `DateFormat` MM/DD/RRRR.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Wartość dnia kwartału jest obliczana przez miarę w obiekcie wykresu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Utwórz następującą miarę:

```
=daynumberofyear(date)
```

Tabela wynikowa

data	=daynumberofyear(date)
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
06/30/2022	182
07/26/2022	208
10/31/2022	305
11/01/2022	306
12/31/2022	366

Pierwszym dniem roku jest 1 stycznia, ponieważ do funkcji `DayNumberOfYear()` nie przekazano drugiego argumentu.

1 stycznia to pierwszy dzień roku, natomiast 1 lutego to 32 dzień roku. 30 czerwca to 182., a 31 grudnia to 366. i ostatni dzień roku.

Przykład 4 – listopad, początek roku (wykres)

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Ten sam zbiór danych, co w pierwszym przykładzie.
- Została użyta domyślna zmienna systemowa `DateFormat` MM/DD/RRRR.
- Rok obrotowy trwa od 1 listopada do 31 października.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Wartość dnia roku jest obliczana przez miarę w obiekcie wykresu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Utwórz następującą miarę:

```
=daynumberofyear(date)
```

Tabela wynikowa

data	=daynumberofyear(date,11)
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	93
02/10/2022	102
06/30/2022	243
07/26/2022	269
10/31/2022	366
11/01/2022	1
12/31/2022	61

Pierwszym dniem roku jest 1 listopada, ponieważ w drugim argumencie do funkcji `DayNumberOfYear()` przekazano wartość 11.

Rok obrotowy trwa od listopada do października. Widać to w tabeli wyników, w której 1 listopada jest pierwszym dniem roku, a 31 października jest 366. i ostatnim dniem roku.

daystart

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu dla pierwszej milisekundy dnia zawartego w argumencie **time**. Domyślnym formatem wyjściowym będzie format **TimestampFormat** skonfigurowany w skrypcie.

Składnia:

```
DayStart(time[, [period_no[, day_start]])
```

Typ zwracanych danych: dual

Argumenty

Argument	Opis
time	Znacznik czasu do oceny.
period_no	Parametr **period_no** jest liczbą całkowitą lub wyrażeniem, którego wynikiem jest liczba całkowita, gdzie wartość 0 wskazuje dzień zawierający wartość **time**. Wartości ujemne parametru **period_no** oznaczają dni poprzednie, a wartości dodatnie – dni następne.
day_start	Aby określić, że dni nie zaczynają się o północy, należy wskazać przesunięcie w postaci części dnia w parametrze **day_start**. Na przykład 0,125 oznacza godzinę trzecią rano. Innymi słowy, aby utworzyć przesunięcie, podziel godzinę rozpoczęcia przez 24 godziny. Na przykład, jeśli dzień ma się zaczynać o godzinie 7:00, użyj ułamka 7/24.

Kiedy używać

Funkcja `daystart()` jest zwykle używana jako część wyrażenia, gdy użytkownik chce, by w obliczeniach użyto ułamka dnia, który upłynął do tej pory. Za jej pomocą można na przykład obliczyć łączną kwotę zarobioną przez pracowników do tej pory w ciągu dnia.

W tych przykładach jest używany format znacznika czasu `'M/D/YYYY h:mm:ss[.fff] TT'`. Format znacznika czasu jest określony w instrukcji `SET TimeStamp` u góry skryptu ładowania danych. Format zastosowany w przykładach można zmienić, aby dostosować go do konkretnych potrzeb.

Przykłady funkcji

Przykład	Wynik
`daystart('01/25/2013 4:45:00 PM')`	Zwraca wartość 1/25/2013 12:00:00 AM.
`daystart('1/25/2013 4:45:00 PM', -1)`	Zwraca wartość 1/24/2013 12:00:00 AM.
`daystart('1/25/2013 16:45:00',0,0.5 )`	Zwraca wartość 1/25/2013 12:00:00 PM.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień

regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – prosty przykład

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Prosty zbiór danych zawierający listę dat, który jest załadowany do tabeli o nazwie `Calendar`.
- Została użyta domyślna zmienna systemowa `TimeStampFormat` (`(M/D/YYYY h:mm:ss[.fff] TT`).
- Poprzednie ładowanie, które tworzy dodatkowe pole, o nazwie `SOD_timestamp`, używające funkcji `daystart()`.

Oprócz daty do funkcji nie przekazano żadnych innych parametrów.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- date
- SOD_timestamp

Tabela wynikowa

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

Jak widać w powyższej tabeli, dla każdej daty w naszym zbiorze danych został wygenerowany znacznik czasu końca dnia. Znacznik ten ma format zmiennej systemowej `TimestampFormat M/D/YYYY h:mm:ss [.fff] TT`.

Przykład 2 - period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zbiór danych zawierający mandaty za parkowanie, który jest załadowany do tabeli o nazwie `Fines`. Zbiór ten zawiera następujące pola:
  - id
  - due_date
  - number_plate
  - amount
- Poprzednie ładowanie, w którym użyto funkcji `daystart()` i przekazano wszystkie trzy parametry: `time`, `period_no` i `day_start`. To poprzednie ładowanie tworzy następujące dwa nowe pola daty:

- Pole daty `early_repayment_period`, zaczynające się siedem dni przed terminem płatności.
- Pole daty `late_penalty_period`, zaczynające się czternaście dni po terminie płatności.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `due_date`
- `early_repayment_period`
- `late_penalty_period`

Tabela wynikowa

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

Wartości nowych pól znajdują się w `TimestampFormat M/DD/YYYY tt`. Jako że użyto funkcji `daystart()`, wszystkie wartości znacznika czasu oznaczają pierwszą milisekundę dnia.

Wartości przedwczesnego terminu spłaty określają datę o siedem dni wcześniejszą od terminu spłaty, ponieważ drugi argument przekazany do funkcji `daystart()` jest ujemny.

Wartości spóźnionego terminu spłaty określają datę o czternaście dni późniejszą od terminu spłaty, ponieważ drugi argument przekazany do funkcji `daystart()` jest dodatni.

Przykład 3 – day_start

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w poprzednim przykładzie.
- To samo uprzednie ładowanie, co w poprzednim przykładzie.

W tym przykładzie ustawiliśmy początek i koniec dnia roboczego na godzinę 7:00 każdego dnia.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- due_date
- early_repayment_period
- late_penalty_period

Tabela wynikowa

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

Teraz daty mają znacznik czasu godziny 7:00, ponieważ argument `day_start`, przekazany do funkcji `daystart()`, ma wartość 7/24. To ustawia początek dnia na godzinę 7:00.

Ponieważ pole `due_date` nie ma znacznika czasu, jest traktowane jako godzina 12:00, która wciąż należy do poprzedniego dnia, ponieważ dni zaczynają i kończą się o godzinie 7:00. W związku z tym okres przedwczesnej spłaty mandatu o terminie płatności 11 lutego zaczyna się 3 lutego o godzinie 7:00.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

W tym przykładzie użyto tego samego zestawu danych i scenariusza co w poprzednim przykładzie.

Jednak do aplikacji załadowano tylko oryginalną tabelę `Fines`, a dwa dodatkowe terminy należności są obliczane w obiekcie wykresu.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `due_date`.
2. Aby utworzyć pole `early_repayment_period`, należy utworzyć następującą miarę: `=daystart(due_date,-7,7/24)`
3. Aby utworzyć pole `late_penalty_period`, należy utworzyć następującą miarę: =daystart(due_date,14,7/24)

Tabela wynikowa

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

Wartości nowych pól znajdują się w `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Ponieważ użyto funkcji `daystart()`, wszystkie wartości znacznika czasu odpowiadają pierwszej milisekundzie dnia.

Wartości przedwczesnego terminu spłaty określają datę o siedem dni wcześniejszą od terminu spłaty, ponieważ drugi argument przekazany do funkcji `daystart()` był ujemny.

Wartości spóźnionego terminu spłaty określają datę o czternaście dni późniejszą od terminu spłaty, ponieważ drugi argument przekazany do funkcji `daystart()` był dodatni.

Daty mają znacznik czasu godziny 7:00, ponieważ trzeci argument (`day_start`) przekazany do funkcji `daystart()` ma wartość 7/24.

firstworkdate

Funkcja **firstworkdate** zwraca najpóźniejszą datę rozpoczęcia, gdy możliwe jest uzyskanie parametru **no_of_workdays** (poniedziałek-piątek) z końcem nie później niż w dniu określonym przez parametr **end_date** oraz z uwzględnieniem wszelkich opcjonalnie wyszczególnionych dni wolnych. Parametry **end_date** i **holiday** powinny być poprawnymi datami lub znacznikami czasu.

Składnia:

```
firstworkdate(end_date, no_of_workdays {, holiday} )
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
end_date	Znacznik czasu daty końcowej do oceny.
no_of_ workdays	Liczba dni roboczych do uzyskania.
holiday	Okresy wolne od pracy wyłączane z dni roboczych. Dzień wolny jest określany jako data stała w formie ciągu. Można określić większą liczbę dat dni wolnych, rozdzielając je przecinkami. **Przykład:** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Przykłady i wyniki:
W tych przykładach używany jest format daty **DD/MM/YYYY**. Format daty jest określony w instrukcji **SET DateFormat** u góry skryptu ładowania danych. Format zastosowany w przykładach można zmienić, aby dostosować go do konkretnych potrzeb.

Przykłady skryptów

Przykład	Wynik
`firstworkdate ('29/12/2014', 9)`	Zwraca wartość 17/12/2014.
`firstworkdate ('29/12/2014', 9, '25/12/2014', '26/12/2014')`	Zwraca 15/12/2014, ponieważ brany jest pod uwagę dwudniowy okres urlopowy.

Przykład:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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;
```

Tabela wynikowa przedstawia zwrócone wartości FirstWorkDate dla każdego z rekordów w tabeli.

Tabela wynikowa

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

Ta funkcja zwraca aktualny czas Greenwich Mean Time ustalony na podstawie ustawień regionalnych. Funkcja zwraca wartości w formacie zmiennej systemowej `TimestampFormat`.

Po każdym przeładowaniu aplikacji następuje korekta tabeli skryptu ładowania, zmiennej lub obiektu wykresu, które używają funkcji `GMT`, pod kątem aktualnego czasu Greenwich Mean Time pobranego od zegara systemowego.

Składnia:

```
GMT( )
```

Typ zwracanych danych: dual

W tych przykładach jest używany format znacznika czasu `M/D/YYYY h:mm:ss[.fff] TT`. Format daty jest określony w instrukcji `SET TimestampFormat` u góry skryptu ładowania danych. Format zastosowany w przykładach można zmienić, aby dostosować go do konkretnych potrzeb.

Przykłady funkcji

Przykład	Wynik
`GMT()`	3/28/2022 2:47:36 PM

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – zmienna (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty. Ten przykład ustawia aktualny czas Greenwich Mean Time jako zmienną w skrypcie ładowania za pomocą funkcji `GMT`.

Skrypt ładowania

```
LET vGMT = GMT();
```

Wyniki

Załaduj dane i utwórz arkusz. Utwórz pole tekstowe za pomocą obiektu wykresu **Tekst i obraz**.

Dodaj następującą miarę do pola tekstowego:

```
=vGMT
```

Pole tekstowe powinno zawierać wiersz tekstu z datą i godziną, jak w poniższym przykładzie:

```
3/28/2022 2:47:36 PM
```

Przykład 2 – listopad, początek roku (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zbiór danych zawierający niezwrócone w terminie książki, który jest załadowany do tabeli o nazwie `Overdue`. Została użyta domyślna zmienna systemowa `DateFormat` MM/DD/RRRR.
- Utworzenie nowego pola o nazwie `days_overdue`, które oblicza, o ile dni spóźniony jest zwrot każdej książki.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `due_date`
- `book_id`
- `days_overdue`

Tabela wynikowa

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

due_date	book_id	days_overdue
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

Wartości w polu `days_overdue` są obliczane przez znalezienie różnicy między aktualnym czasem Greenwich Mean Time, za pomocą funkcji `GMT()`, i oryginalnym terminem zwrotu. Aby obliczyć tylko liczbę dni, wyniki są zaokrąglane do najbliższej liczby całkowitej za pomocą funkcji `Floor()`.

Przykład 3 – obiekt wykresu (wykres)

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty. Skrypt ładowania zawiera ten sam zestaw danych co w poprzednim przykładzie. Została użyta domyślna zmienna systemowa `DateFormat` MM/DD/RRRR.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Wartość liczby dni spóźnienia jest obliczana przez miarę w obiekcie wykresu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- due_date
- book_id

Utwórz następującą miarę:

```
=Floor(GMT() - due_date)
```

Tabela wynikowa

due_date	book_id	=Floor(GMT()-due_date)
01/01/2021	4	455
01/10/2021	24	446
01/31/2021	173	425
02/01/2021	281	424
02/10/2021	265	415
06/30/2021	465	275
07/26/2021	537	249
10/31/2021	275	152
11/01/2021	455	151
12/31/2021	46	91

Wartości w polu days_overdue są obliczane przez znalezienie różnicy między aktualnym czasem Greenwich Mean Time, za pomocą funkcji GMT(), i oryginalnym terminem zwrotu. Aby obliczyć tylko liczbę dni, wyniki są zaokrąglane do najbliższej liczby całkowitej za pomocą funkcji Floor().

hour

Ta funkcja zwraca liczbę całkowitą reprezentującą godzinę, gdy ułamek wyrażenia **expression** jest interpretowany jako czas zgodnie ze standardową interpretacją liczb.

Składnia:

```
hour(expression)
```

Typ zwracanych danych: integer

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji SET DateFormat w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne

czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
`hour( '09:14:36' )`	Dostarczony ciąg tekstowy jest niejawnie konwertowany na znacznik czasu, ponieważ jest zgodny z formatem znacznika czasu zdefiniowanym w zmiennej TimestampFormat. Wyrażenie zwraca 9.
`hour( '0.5555' )`	Wyrażenie zwraca 13 (ponieważ 0.5555 = 13:19:55)

Przykład 1 – zmienna (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Zbiór danych zawierający transakcje wg znacznika czasu
- Domyślna `TimeStamp` zmienna systemowa `(M/D/YYYY h:mm:ss[.fff] TT)`.

Utwórz pole „`hour`”, obliczając, kiedy miały miejsce zakupy.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `hour`

Tabela wynikowa

data	godzina
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

Wartości w polu godziny są tworzone za pomocą funkcji `hour()` i przez przekazanie daty jako wyrażenia w poprzedniej instrukcji ładowania.

Przykład 2 – obiekt wykresu (wykres)

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Ten sam zbiór danych, co w pierwszym przykładzie.
- Domyślna `TimeStamp` zmienna systemowa `(M/D/YYYY h:mm:ss[.fff] TT)`.

Jednak w tym przykładzie zbiór danych, niezmieniony, jest ładowany do aplikacji. Wartości „hour” są obliczane przez miarę w obiekcie wykresu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Aby obliczyć „hour”, należy utworzyć następującą miarę:

```
=hour(date)
```

Tabela wynikowa

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

Wartości dla „hour” są tworzone za pomocą funkcji `hour()` i przez przekazanie daty jako wyrażenia w mierze dla obiektu wykresu.

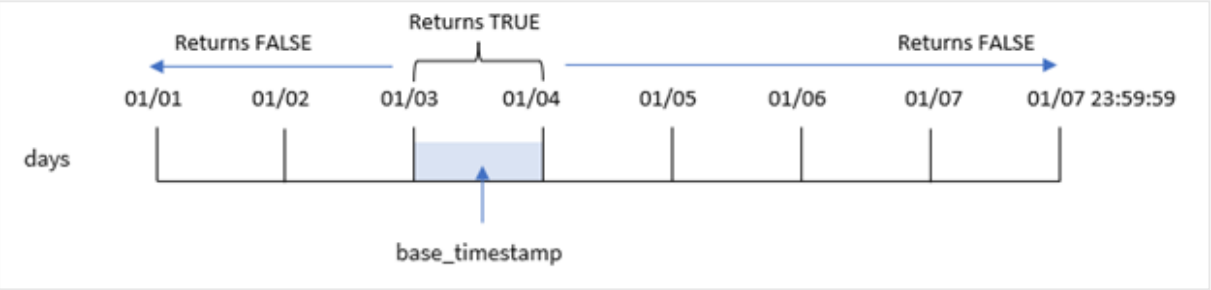
inday

Ta funkcja zwraca wartość True, jeśli znacznik czasu **timestamp** należy do dnia zawierającego wartość **base_timestamp**.

Składnia:

```
InDay (timestamp, base_timestamp, period_no[, day_start])
```

Schemat funkcji `inday`



Funkcja `inday()` używa argumentu `base_timestamp` do określenia, w którym dniu przypada znacznik czasu. Godzina rozpoczęcia dnia to domyślnie północ, ale możesz zmienić godzinę rozpoczęcia dnia, używając argumentu `day_start` funkcji `inday()`. Po zdefiniowaniu tego dnia funkcja będzie zwracać wyniki logiczne podczas porównywania określonych wartości znaczników czasu z tym dniem.

Kiedy używać

Funkcja `inday()` zwraca wynik logiczny. Zazwyczaj ten typ funkcji będzie używany jako warunek w `if expression`. Zwraca to agregację lub obliczenia zależne od tego, czy oceniana data nastąpiła w dniu danego znacznika czasu.

Na przykład funkcja `inday()` może służyć do identyfikacji całego sprzętu wyprodukowanego w danym dniu.

Typ zwracanych danych: wartość logiczna

W Qlik Sense wartość logiczna Prawda jest reprezentowana przez -1, a wartość Fałsz jest reprezentowana przez 0.

Argumenty

Argument	Opis
`timestamp`	Data i godzina, która ma być porównana z wartością `base_timestamp`.
`base_timestamp`	Data i godzina używane do oceny znacznika czasu.

Argument	Opis
period_no	Dzień może zostać przesunięty o wartość period_no. period_no jest liczbą całkowitą, gdzie 0 oznacza dzień zawierający wartość base_timestamp. Wartości ujemne parametru period_no oznaczają dni poprzednie, a wartości dodatnie – dni następne.
day_start	Jeśli wymagane jest korzystanie z dni, które nie zaczynają się o północy, należy wskazać przesunięcie w postaci ułamka dnia w parametrze day_start, np. 0,125 będzie oznaczać godzinę trzecią rano.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji SET DateFormat w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	Zwraca prawdę
inday ('01/12/2006 12:23:00 PM', '01/13/2006 12:00:00 AM', 0)	Zwraca fałsz
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	Zwraca fałsz
inday ('01/11/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	Zwraca prawdę
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0, 0.5)	Zwraca fałsz
inday ('01/12/2006 11:23:00 AM', '01/12/2006 12:00:00 AM', 0, 0.5)	Zwraca prawdę

Przykład 1 – instrukcja ładowania (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zbiór danych zawierający transakcje wg znacznika czasu, który jest ładowany do tabeli o nazwie „Transactions”.
- Pole daty, które jest przekazywane w formacie `TimeStamp` zmiennej systemowej `(M/D/YYYY h:mm:ss [.fff] TT)`.
- Ładunek poprzedzający, zawierający funkcję `inday()`, która jest ustawiona jako pole `in_day`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `in_day`

Tabela wynikowa

data	in_day
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	0
01/04/2022 6:49:38 PM	0

data	in_day
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

Pole `in_day` zostało utworzone w poprzedniej instrukcji ładowania za pomocą funkcji `inday()`, której jako argumenty przekazano pole daty, wpisany w kod znacznik czasu oznaczający 5 stycznia, i `period_no` o wartości `0`.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Skrypt ładowania wykorzystuje ten sam zbiór danych i scenariusz, które były używane w pierwszym przykładzie.

Jednak w tym przykładzie chodzi o obliczenie, czy data transakcji wypada dwa dni przed 5 stycznia.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `in_day`

Tabela wynikowa

data	in_day
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	-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

W tym przypadku, w którym funkcji `inday()` przekazano argument przesunięcia `period_no` o wartości `-2`, funkcja ta sprawdza, czy każda transakcja miała miejsce 3 stycznia. Można to zweryfikować w tabeli wyników, w której jedna transakcja zwraca wartość logiczną PRAWDA.

Przykład 3 – day_start

Skrypt ładowania i wyniki

Przegląd

Skrypt ładowania wykorzystuje ten sam zbiór danych i scenariusz, które były używane w poprzednich przykładach.

Jednak w tym przykładzie zgodnie z polityką firmy dzień roboczy zaczyna i kończy się o godzinie 7:00.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `in_day`

Tabela wynikowa

data	in_day
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	0
01/04/2022 6:49:38 PM	-1
01/04/2022 10:58:34 PM	-1
01/05/2022 5:40:49 AM	-1
01/05/2022 11:29:38 AM	0
01/05/2022 7:04:57 PM	0
01/06/2022 8:49:09 AM	0

Ponieważ funkcji `inday()` przekazano argument `start_day` o wartości 7/24, oznaczającej godzinę 7:00, funkcja ta sprawdza, czy każda transakcja miała miejsce między godziną 7:00 4 stycznia a godziną 7:00 5 stycznia.

Można to zweryfikować w tabeli wynikowej, w której transakcje po godzinie 7:00 4 stycznia zwracają wartość logiczną PRAWDA, natomiast transakcje, które miały miejsce po godzinie 7:00 5 stycznia zwracają wartość logiczną FAŁSZ.

Przykład 4 – obiekt wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Skrypt ładowania wykorzystuje ten sam zbiór danych i scenariusz, które były używane w poprzednich przykładach.

Jednak w tym przykładzie zbiór danych pozostaje bez zmian i jest ładowany do aplikacji. Będą wykonywane obliczenia mające na celu stwierdzenie, czy transakcja ma miejsce 5 stycznia, przez utworzenie miary w obiekcie wykresu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

- `date`

Aby obliczyć, czy transakcja ma miejsce 5 stycznia, utwórz następującą miarę:

```
=inday(date,'01/05/2022 12:00:00 AM',0)
```

Tabela wynikowa

data	inday(date,'01/05/2022 12:00:00 AM',0)
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0

data	inday(date,'01/05/2022 12:00:00 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

Przykład 5 – scenariusz

Skrypt ładowania i wyniki

Przegląd

W tym przykładzie stwierdzono, że z powodu błędu sprzętowego produkty wytworzone 5 stycznia były wadliwe. Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał według dat status produktów: „defective” (wadliwe) lub „faultless” (bez wad) oraz koszt produktów wytworzonych 5 stycznia.

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych ładowany do tabeli „Products”.
- Tabela zawiera następujące pola:
    - product ID
    - manufacture time
    - cost price

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

```
=dayname(manufacture_date)
```

Utwórz następujące miary:

- `=if(only(InDay(manufacture_date,makedate(2022,01,05),0)),'Defective','Faultless')`
- `=sum(cost_price)`

Ustaw **Formatowanie liczb** miary na **Waluta**.

W sekcji **Wygląd** wyłącz **Sumy**.

Tabela wynikowa

dayname (manufacture_date)	=if(only(InDay(manufacture_date,makedate (2022,01,05),0)),'Defective','Faultless')	=sum(cost_ price)
01/01/2022	Faultless	44.67
01/02/2022	Faultless	36.34
01/03/2022	Faultless	51.75
01/04/2022	Faultless	89.69
01/05/2022	Defective	170.78
01/06/2022	Faultless	74.23

Funkcja `inday()` zwraca wartość logiczną podczas oceny dat wytworzenia każdego z produktów. W przypadku każdego produktu wyprodukowanego 5 stycznia funkcja `inday()` zwraca wartość logiczną PRAWDA i oznacza produkty jako „Defective” (Wadliwe). W przypadku każdego produktu zwracającego wartość FALSE, a zatem niewyprodukowanego w tym dniu, oznacza ona produkty jako „Faultless” (Bez wad).

indaytotime

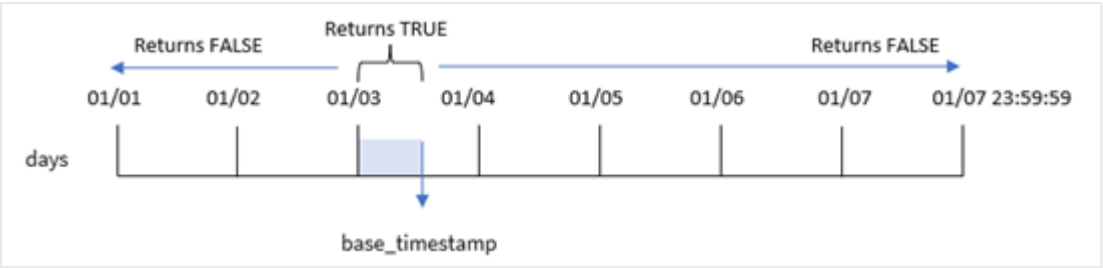
Ta funkcja zwraca wartość True, jeśli wartość **timestamp** należy do części dnia zawierającego wartość **base_timestamp**, z dokładnością do jednej milisekundy wartości **base_timestamp** włącznie.

Składnia:

```
InDayToTime (timestamp, base_timestamp, period_no[, day_start])
```

Funkcja `indaytotime()` zwraca wartość logiczną określoną na podstawie tego, czy wartość znacznika czasu mieści się w danym odcinku dnia. Granicą początkową tego odcinka jest początek dnia. Domyślna wartość to północ, ale można ją zmienić za pomocą argumentu `day_start` funkcji `indaytotime()`. Granicę końcową odcinka dnia określa argument funkcji `base_timestamp`.

Schemat funkcji `indaytotime`.



Kiedy używać

Funkcja `indaytotime()` zwraca wynik logiczny. Zazwyczaj ten typ funkcji będzie używany jako warunek w `if expression`. Funkcja `indaytotime()` zwraca agregację lub obliczenie w zależności od tego, czy znacznik czasu wystąpił w segmencie dnia do momentu podstawowego znacznika czasu włącznie.

Na przykład funkcja `indaytotime()` może służyć do pokazania sumy sprzedaży biletów na seanse, które odbyły się do tej pory w dniu dzisiejszym.

Typ zwracanych danych: wartość logiczna

W Qlik Sense wartość logiczna Prawda jest reprezentowana przez -1, a wartość Fałsz jest reprezentowana przez 0.

Argumenty

Argument	Opis
`timestamp`	Data i godzina, które mają być porównane z wartością `base_timestamp`.
`base_timestamp`	Data i godzina używane do oceny znacznika czasu.
`period_no`	Dzień może zostać przesunięty o wartość `period_no`. `period_no` jest liczbą całkowitą, gdzie 0 oznacza dzień zawierający wartość `base_timestamp`. Wartości ujemne parametru `period_no` oznaczają dni poprzednie, a wartości dodatnie – dni następne.
`day_start`	Opcjonalnie, jeśli wymagane jest korzystanie z dni, które nie zaczynają się o północy, należy wskazać przesunięcie w postaci ułamka dnia w parametrze `day_start`. Na przykład 0,125 będzie oznaczać godzinę 3:00.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień

regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
`indaytotime` ('01/12/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', 0)	Zwraca prawdę
`indaytotime` ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	Zwraca fałsz
`indaytotime` '01/11/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', -1)	Zwraca prawdę

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zbiór transakcji z okresu między 4 i 5 stycznia jest ładowany do tabeli o nazwie „Transactions”.
- Pole daty, które jest przekazywane w formacie `TimeStamp` zmiennej systemowej `(M/D/YYYY h:mm:ss [.fff] TT)`.
- Poprzednie ładowanie zawierające funkcję `indaytotime()`, która jest ustawiona jako `'in_day_to_time'`, pole określające, czy każda z transakcji miała miejsce przed godziną 9:00.

Skrypt ładowania

```
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
];
```

Wyniki

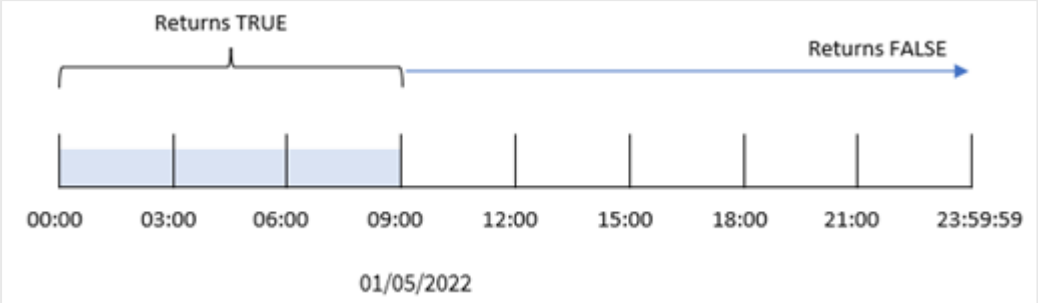
Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `in_day_to_time`

Tabela wynikowa

data	in_day_to_time
01/04/2022 3:41:54 AM	0
01/04/2022 4:19:43 AM	0
01/04/2022 04:53:47 AM	0
01/04/2022 8:38:53 AM	0
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	-1
01/05/2022 8:44:36 AM	-1
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Przykład 1 - schemat funkcji `indaytotime` z limitem do godziny 9:00.



Pole `in_day_to_time field` zostało utworzone w poprzedniej instrukcji ładowania za pomocą funkcji `indaytotime()`, której jako argumenty przekazano pole daty, wpisany w kod znacznik czasu oznaczający 5 stycznia, i przesunięcie o wartości 0. Wszystkie transakcje mające miejsce między północą a godziną 9:00 5 stycznia dają wynik prawdziwy.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Skrypt ładowania wykorzystuje ten sam zbiór danych i scenariusz, które były używane w pierwszym przykładzie.

Jednak w tym przykładzie chodzi o obliczenie, czy data transakcji wypada dzień przed godziną 9:00 5 stycznia.

Skrypt ładowania

```
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
];
```

Wyniki

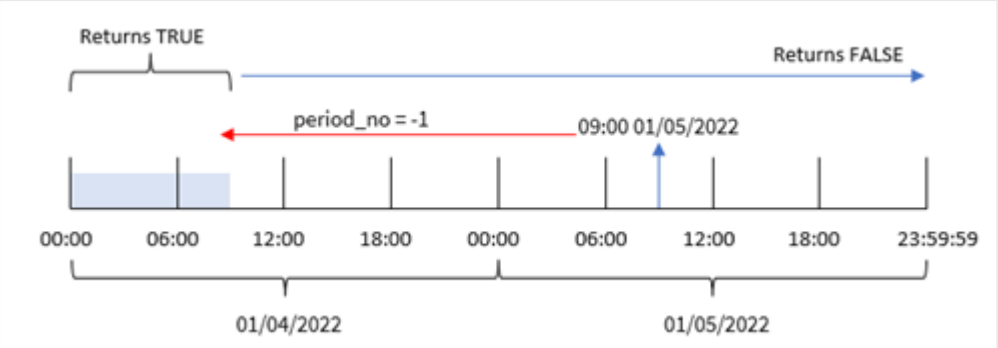
Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- date
- in_day_to_time

Tabela wynikowa

data	in_day_to_time
01/04/2022 3:41:54 AM	-1
01/04/2022 4:19:43 AM	-1
01/04/2022 04:53:47 AM	-1
01/04/2022 8:38:53 AM	-1
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	0
01/05/2022 8:44:36 AM	0
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Przykład 2 - schemat funkcji `indaytotime` z transakcjami od 4 stycznia.



W tym przykładzie, w którym funkcji `indaytotime()` przekazano argument przesunięcia o wartości -1, funkcja ta sprawdza, czy każda transakcja miała miejsce przed godziną 9:00 4 stycznia. Można to zweryfikować w tabeli wyników, w której transakcja zwraca wartość logiczną oznaczającą prawdę.

Przykład 3 – day_start

Skrypt ładowania i wyniki

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie zgodnie z polityką firmy dzień roboczy zaczyna i kończy się o godzinie 8:00.

Skrypt ładowania

```
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
];
```

Wyniki

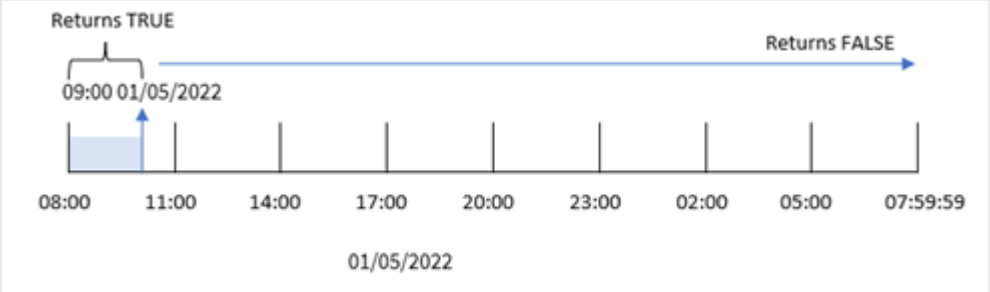
Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `in_day_to_time`

Tabela wynikowa

data	in_day_to_time
01/04/2022 3:41:54 AM	0
01/04/2022 4:19:43 AM	0
01/04/2022 04:53:47 AM	0
01/04/2022 8:38:53 AM	0
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	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

Przykład 3 - schemat funkcji `indaytotime` z transakcjami od godziny 8:00 do 9:00.



Ponieważ w funkcji `indaytotime()` użyto argumentu `start_day` 8/24, który odpowiada godzinie 8:00, każdy dzień zaczyna się i kończy o 8:00. Dlatego funkcja `indaytotime()` zwróci wynik logiczny PRAWDA dla każdej transakcji, która wystąpiła między 8:00 a 9:00 5 stycznia.

Przykład 4 – obiekt wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie zbiór danych pozostaje bez zmian i jest ładowany do aplikacji. Będą wykonywane obliczenia mające na celu stwierdzenie, czy transakcja ma miejsce 5 stycznia przed godziną 9:00, przez utworzenie miary w obiekcie wykresu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

`date`.

Aby sprawdzić, czy transakcja ma miejsce 5 stycznia przed godziną 9:00, utwórz następującą miarę:

```
=indaytotime(date,'01/05/2022 9:00:00 AM',0)
```

Tabela wynikowa

data	=indaytotime(date,'01/05/2022 9:00:00 AM',0)
01/04/2022 3:41:54 AM	0
01/04/2022 4:19:43 AM	0
01/04/2022 04:53:47 AM	0
01/04/2022 8:38:53 AM	0
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	-1
01/05/2022 8:44:36 AM	-1
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Pole `in_day_to_time` zostało utworzone w poprzedniej instrukcji ładowania za pomocą funkcji `indaytotime ()`, której jako argumenty przekazano pole daty, wpisany w kod znacznik czasu oznaczający 5 stycznia, i przesunięcie o wartości `0`. Wszystkie transakcje mające miejsce między północą a godziną 9:00 5 stycznia dają wynik prawdziwy. Można to zweryfikować w tabeli wyników.

Przykład 5 – scenariusz

Skrypt ładowania i wyniki

Przegląd

W tym przykładzie zestaw danych zawierający dane sprzedaży biletów do lokalnego kina jest ładowany do tabeli o nazwie Ticket_Sales. Dziś jest 3 maja 2022 r. i jest godzina 11:00.

Użytkownik chciałby, aby obiekt wykresu wskaźników KPI pokazywał przychody uzyskane ze wszystkich dotychczasowych seansów.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Utwórz obiekt wskaźnik KPI.
2. Utwórz miarę, która pokaże sumę sprzedaży wszystkich biletów na seanse, które odbyły się dzisiaj do tej pory, korzystając z funkcji `indaytotime()`:

```
=sum(if(indaytotime([show time],'05/03/2022 11:00:00 AM',0),[ticket price],0))
```

3. Utwórz etykietę dla obiektu wskaźnika KPI „Current Revenue”.
4. Ustaw **Formatowanie liczb** miary na **Waluta**.

Suma wpływów ze sprzedaży biletów do godziny 11:00 3 maja 2022 r. to 52,50 USD.

Funkcja `indaytotime ()` zwraca wartość logiczną podczas porównywania godzin seansów każdej sprzedaży biletów z czasem bieżącym (’05/03/2022 11:00:00’). W przypadku każdego seansu 3 maja przed godziną 11:00 funkcja `indaytotime ()` zwraca wartość logiczną PRAWDA, a cena biletu zostanie uwzględniona w sumie łącznej.

inlunarweek

Ta funkcja sprawdza, czy wartość **timestamp** należy do tygodnia księżycowego zawierającego wartość **base_date**. Tygodnie księżycowe w Qlik Sense są zdefiniowane przez uznanie 1 stycznia za pierwszy dzień tygodnia. Każdy tydzień, z wyjątkiem ostatniego tygodnia roku, będzie zawierał dokładnie siedem dni.

Składnia:

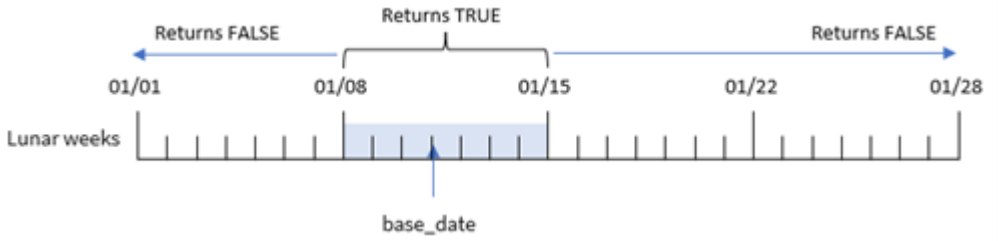
```
InLunarWeek (timestamp, base_date, period_no[, first_week_day])
```

Typ zwracanych danych: Wartość logiczna

> *W Qlik Sense wartość logiczna Prawda jest reprezentowana przez -1, a wartość Fałsz jest reprezentowana przez 0.*

Funkcja `inlunarweek()` sprawdza, do którego tygodnia księżycowego należy `base_date`. Następnie zwraca wynik logiczny po określeniu, czy każdy znacznik czasu wypada w tym samym tygodniu księżycowym co `base_date`.

Schemat funkcji `inlunarweek()`



Kiedy używać

Funkcja `inlunarweek()` zwraca wynik logiczny. Zazwyczaj tego typu funkcji używa się jako warunku w wyrażeniu IF. Spowoduje to zwrócenie agregacji lub obliczenia w zależności od tego, czy oceniana data nastąpiła w danym tygodniu księżycowym.

Na przykład funkcja `inlunarweek()` może służyć do identyfikacji całego sprzętu wyprodukowanego w danym tygodniu księżycowym.

Argumenty

Argument	Opis
timestamp	Data, która ma być porównana z wartością **base_date**.
base_date	Data używana do oceny tygodnia księżycowego.
period_no	Tydzień księżycowy może zostać przesunięty o wartość **period_no**. period_no jest liczbą całkowitą, gdzie 0 oznacza tydzień księżycowy zawierający wartość **base_date**. Wartości ujemne parametru **period_no** oznaczają poprzednie tygodnie księżycowe, a wartości dodatnie – następne tygodnie księżycowe.
first_week_day	Przesunięcie może być większe lub mniejsze od zera. Zmienia to początek roku o określoną liczbę dni lub części dnia.

Przykłady funkcji

Przykład	Wynik
`inlunarweek ('01/12/2013', '01/14/2013', 0)`	Zwraca `TRUE`, ponieważ wartość `timestamp`, 01/12/2013, wypada w tygodniu od 01/08/2013 do 01/14/2013.
`inlunarweek ('01/12/2013', '01/07/2013', 0)`	Zwraca `FALSE`, ponieważ `base_date` 01/07/2013 wypada w tygodniu księżycowym zdefiniowanym jako obejmujący dni od 01/01/2013 do 01/07/2013.
`inlunarweek ('01/12/2013', '01/14/2013', -1)`	Zwraca wartość `FALSE`. Nadanie argumentowi period_no wartości -1 powoduje przesunięcie tygodnia do poprzedniego tygodnia, od 01/01/2013 do 01/07/2013.
`inlunarweek ('01/07/2013', 01/14/2013', -1)`	Zwraca wartość `TRUE`. W odróżnieniu od poprzedniego przykładu, `timestamp` przypada w następnym tygodniu po uwzględnieniu przesunięcia do tyłu.
`inlunarweek ('01/11/2006', '01/08/2006', 0, 3)`	Zwraca wartość `FALSE`. Jeśli argumentowi `first_week_day` przypiszemy wartość 3, to początek roku będzie obliczany od 01/04/2013. W związku z tym wartość `base_` date wypada w pierwszym tygodniu, a wartość `timestamp` wypada w tygodniu od 01/11/2013 do 01/17/2013.

Funkcja `inlunarweek()` jest często używana w połączeniu z następującymi funkcjami:

Powiązane funkcje

Funkcja	Interakcja
lunarweekname (page 842)	Ta funkcja służy do określania numeru tygodnia księżycowego roku, w którym wypada data przekazana na wejściu.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych transakcji zawartych w styczniu, który jest ładowany do tabeli o nazwie `Transactions`.
- Pole daty zostało podane w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).

Utwórz pole, `in_lunar_week`, określające czy transakcje miały miejsce w tym samym tygodniu księżycowym, w którym wypada data 10 stycznia.

Skrypt ładowania

```
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
];
```

Wyniki

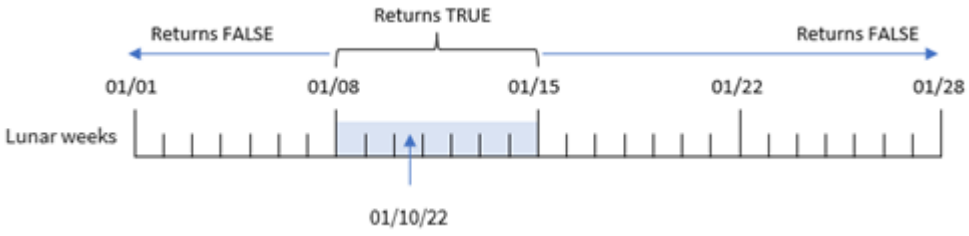
Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `in_lunar_week`

Tabela wynikowa

date	in_lunar_week
1/5/2022	0
1/6/2022	0
1/7/2022	0
1/8/2022	-1
1/9/2022	-1
1/10/2022	-1
1/11/2022	-1
1/12/2022	-1
1/13/2022	-1
1/14/2022	-1
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	0
1/23/2022	0

Funkcja `inlunarweek()`, przykład podstawowy



Pole `in_lunar_week` jest tworzone w poprzedzającej instrukcji ładowania za pomocą funkcji `inlunarweek()`. Następnie do funkcji zostają przekazane następujące argumenty:

- Pole `date`
- Data 10 stycznia wpisana bezpośrednio w kod jako `base_date`
- Argument `period_no` o wartości 0

Ponieważ tygodnie księżycowe zaczynają się 1 stycznia, data 10 stycznia wypadłaby w tygodniu księżycowym zaczynającym się 8 stycznia i kończącym się 14 stycznia. W związku z tym wszelkie transakcje, które mają miejsce między tymi dwiema styczniowymi datami, zwracałyby wartość `TRUE`. Można to zweryfikować w tabeli wyników.

Przykład 2 – period_no

Przykłady i wyniki:

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Pole daty zostało podane w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).

Jednak w tym przykładzie chodzi o utworzenie pola, `2_lunar_weeks_later`, które będzie sprawdzało, czy transakcje miały miejsce dwa tygodnie księżycowe po 10 stycznia.

Skrypt ładowania

```
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
];
```

Wyniki

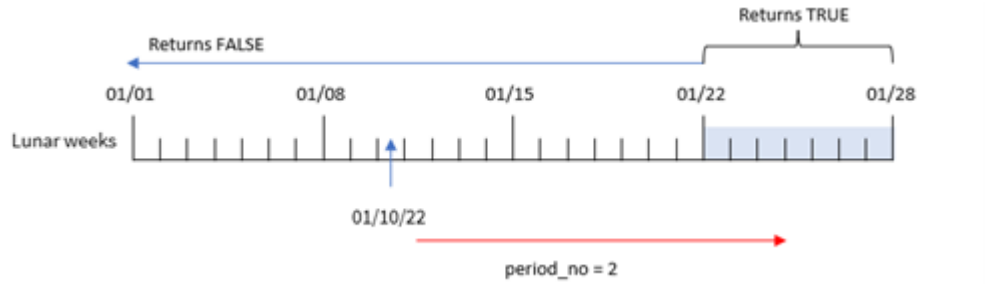
Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `2_lunar_weeks_later`

Tabela wynikowa

date	2_lunar_weeks_later
1/5/2022	0
1/6/2022	0
1/7/2022	0
1/8/2022	0
1/9/2022	0
1/10/2022	0
1/11/2022	0
1/12/2022	0
1/13/2022	0
1/14/2022	0
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	-1
1/23/2022	-1

Funkcja `inlunarweek()`, przykład z użyciem argumentu period_no



W tym przykładzie, ponieważ do funkcji `inlunarweek()` jako wartość przesunięcia przekazano argument `period_no` o wartości 2, jako tydzień, według którego mają być sprawdzane transakcje, został zdefiniowany tydzień zaczynający się dnia 22 stycznia. W związku z tym każda transakcja, która ma miejsce między 22 a 28 stycznia zwróci wynik logiczny `TRUE`.

Przykład 3 – first_week_day

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania używa tego samego zestawu danych i scenariusza co w pierwszym przykładzie. Jednak w tym przykładzie ustawiamy początek tygodni księżycowych na 6 stycznia.

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Została użyta domyślna zmienna systemowa `DateFormat` MM/DD/RRRR.
- Argument `first_week_day` o wartości. W ten sposób ustawiamy początek tygodni księżycowych na dzień 5 stycznia.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

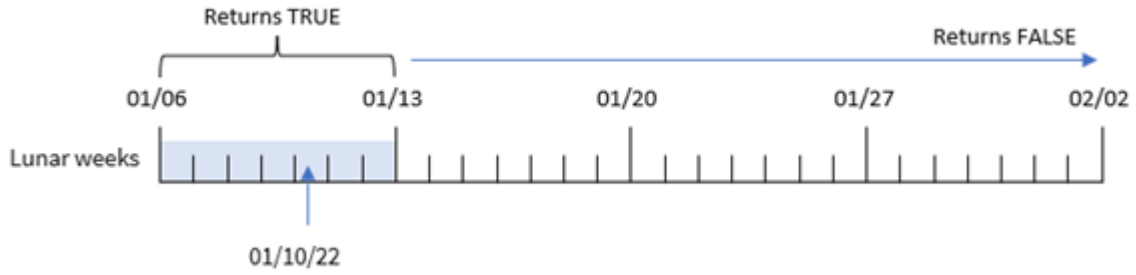
- `date`
- `in_lunar_week`

Tabela wynikowa

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

date	in_lunar_week
1/20/2022	0
1/21/2022	0
1/22/2022	0
1/23/2022	0

Funkcja `inlunarweek()`, przykład z argumentem first_week_day



W tym przykładzie przekazaliśmy funkcji `inlunarweek()` argument `first_week_date` o wartości 5, który przesuwa początek tygodnia księżycowego na 6 stycznia. W efekcie 10 stycznia wypada w tygodniu księżycowym zaczynającym się 6 stycznia i kończącym się 12 stycznia. Wszelkie transakcje mające miejsce między tymi dwiema datami będą zwracać logiczną wartość `TRUE`.

Przykład 4 – obiekt wykresu

Skrypt ładowania i wyrażenie wykresu:

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Pole daty zostało podane w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenia określające, czy transakcje miały miejsce w tym samym tygodniu księżycowym, w którym wypada 10 stycznia, jest tworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Aby obliczyć, czy transakcja ma miejsce w tygodniu księżycowym obejmującym 10 stycznia, utwórz następującą miarę:

```
= inlunarweek(date,'01/10/2022', 0)
```

Tabela wynikowa

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

date	=inlunarweek(date,'01/10/2022', 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

Przykład 5 – scenariusz

Skrypt ładowania i wyrażenie wykresu:

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Products`.
- Informacje obejmujące identyfikator produktu, datę produkcji i cenę.

Stwierdzono, że z powodu błędu sprzętowego produkty wytworzone w tygodniu księżycowym obejmującym 12 stycznia były wadliwe. Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał według tygodni księżycowych informację, czy produkty były wadliwe, czy bez wad oraz koszt produktów wytworzonych w danym miesiącu.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę.
2. Utwórz wymiar, aby pokazać nazwy tygodni:
   `=lunarweekname(manufacture_date)`
3. Utwórz miarę, aby określić, które produkty są wadliwe, a które bez wad za pomocą funkcji `inlunarweek()`:
   `=if(only(inlunarweek(manufacture_date,makedate(2022,01,12),0)), 'Defective','Faultless')`
4. Utwórz miarę, aby zsumować `cost_price` produktów:
   `=sum(cost_price)`
5. Ustaw **Formatowanie liczb** miary na **Waluta**.
6. W sekcji **Wygląd** wyłącz **Sumy**.

Tabela wynikowa

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

Funkcja `inlunarweek()` zwraca wartość logiczną podczas oceny dat wytworzenia każdego z produktów. W przypadku każdego produktu wyprodukowanego w tygodniu obejmującym 10 stycznia funkcja `inlunarweek()` zwraca wartość logiczną `TRUE` i oznacza produkty jako wadliwe. Każdy produkt, dla którego jest zwracana wartość `FALSE`, a zatem niewyprodukowany w tym tygodniu, oznacza jako pozbawiony wad.

inlunarweektodate

Ta funkcja sprawdza, czy wartość **timestamp** należy do części tygodnia księżycowego do ostatniej milisekundy wartości **base_date** włącznie. Tygodnie księżycowe w Qlik Sense są zdefiniowane przez uznanie 1 stycznia za pierwszy dzień tygodnia i każdy tydzień, z wyjątkiem ostatniego tygodnia roku, będzie zawierał dokładnie siedem dni.

Składnia:

```
InLunarWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Typ zwracanych danych: Wartość logiczna

> *W Qlik Sense wartość logiczna Prawda jest reprezentowana przez -1, a wartość Fałsz jest reprezentowana przez 0.*

Przykładowy schemat funkcji `inlunarweektodate()`

Funkcja `inlunarweektodate()` działa jako punkt końcowy tygodnia księżycowego. Natomiast funkcja `inlunarweek()` sprawdza, do którego tygodnia księżycowego należy `base_date`. Gdyby na przykład argument `base_date` określał datę 5 stycznia, to każdy znacznik czasu określający datę między 1 a 5 stycznia powodowałby zwrot wartości logicznej `TRUE`, natomiast dla dat od 6 stycznia byłaby zwracana wartość logiczna `FALSE`.

Argumenty

Argument	Opis
timestamp	Data, która ma być porównana z wartością **base_date**.
base_date	Data używana do oceny tygodnia księżycowego.
period_no	Tydzień księżycowy może zostać przesunięty o wartość **period_no**. period_no jest liczbą całkowitą, gdzie 0 oznacza tydzień księżycowy zawierający wartość **base_date**. Wartości ujemne parametru **period_no** oznaczają poprzednie tygodnie księżycowe, a wartości dodatnie – następne tygodnie księżycowe.
first_week_day	Przesunięcie może być większe lub mniejsze od zera. Zmienia to początek roku o określoną liczbę dni lub części dnia.

Kiedy używać

Funkcja `inlunarweektodate()` zwraca wynik logiczny. Zazwyczaj ten typ funkcji jest używany jako warunek w wyrażeniu IF. Funkcja `inlunarweektodate()` mogłaby zostać użyta w przypadku, gdy użytkownik chciałby uzyskać w wyniku obliczeń agregację lub kalkulację, zależnie od tego czy ewaluowana data wypada w określonym segmencie interesującego go tygodnia.

Na przykład funkcja `inlunarweektodate()` może służyć do identyfikacji całego sprzętu wyprodukowanego w danym tygodniu do określonej daty włącznie.

Przykłady funkcji

Przykład	Wynik
`inlunarweektodate ('01/12/2013', '01/13/2013', 0)`	Zwraca `TRUE`, ponieważ wartość `timestamp`, 01/12/2013, wypada w części tygodnia od 01/08/2013 do 01/13/2013.
`inlunarweektodate ('01/12/2013', '01/11/2013', 0)`	Zwraca `FALSE`, ponieważ wartość `timestamp` jest późniejsza niż wartość `base_date`, mimo że obie te daty wypadają w tym samym tygodniu księżycowym przed 01/12/2012.
`inlunarweektodate ('01/12/2006', '01/05/2006', 1)`	Zwraca wartość `TRUE`. Określenie wartości 1 dla pola `period_no` powoduje przesunięcie `base_date` o jeden tydzień w przód, a więc wartość `timestamp` przypada w części tygodnia księżycowego.

Funkcja `inlunarweektodate()` jest często używana w połączeniu z następującymi funkcjami:

Powiązane funkcje

Funkcja	Interakcja
lunarweekname (page 842)	Ta funkcja służy do określania numeru tygodnia księżycowego roku, w którym wypada data przekazana na wejściu.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zbiór styczniowych transakcji, który jest ładowany do tabeli o nazwie `Transactions`. Została użyta domyślna zmienna systemowa `DateFormat` MM/DD/RRRR.
- Utwórz pole, `in_lunar_week_to_date`, określające które transakcje miały miejsce w tygodniu księżycowym do daty 10 stycznia.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- date
- in_lunar_week_to_date

Tabela wynikowa

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

Funkcja `inlunarweektodate()`, bez dodatkowych argumentów

Pole `in_lunar_week_to_date` zostało utworzone w poprzedniej instrukcji ładowania za pomocą funkcji `inlunarweektodate()`, której jako argumenty przekazano pole `date`, wpisaną w kod datę 10 stycznia jako `base_date` oraz przesunięcie o wartości 0.

Ponieważ tygodnie księżycowe zaczynają się 1 stycznia, data 10 stycznia wypadłaby w tygodniu księżycowym zaczynającym się 8 stycznia, a ponieważ używamy funkcji `inlunarweektodate()`, ten tydzień księżycowy kończyłby się 10 dnia. W związku z tym wszelkie transakcje, które mają miejsce między tymi dwiema styczniowymi datami, zwracałyby wartość `TRUE`. Można to zweryfikować w tabeli wyników.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie. Jednak w tym przykładzie chodzi o utworzenie pola, `2_lunar_weeks_later`, które będzie sprawdzało, czy transakcje miały miejsce dwa tygodnie po tygodniu księżycowym do daty 1 stycznia.

Skrypt ładowania

```
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
];
```

Wyniki

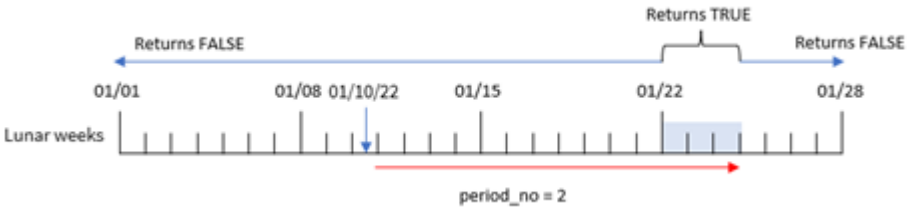
Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `2_lunar_weeks_later`

Tabela wynikowa

date	2_lunar_weeks_later
1/1/2022	0
1/4/2022	0
1/10/2022	0
1/11/2022	0
1/12/2022	0
1/15/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	-1
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

Funkcja `inlunarweektodate()`, przykład z użyciem argumentu period_no



W tym przykładzie funkcja `inlunarweektodate()` stwierdza, że tydzień księżycowy do 10 stycznia trwa trzy dni (8, 9 i 10 stycznia). Ponieważ jako przesunięcia użyto argumentu `period_no` o wartości 2, ten tydzień księżycowy zostaje przesunięty o 14 dni. To oznacza, że trzydniowy tydzień księżycowy obejmuje dni 22, 23 i 24 stycznia. Każda transakcja, która ma miejsce między 22 a 24 stycznia zwróci wynik logiczny `TRUE`.

Przykład 3 – first_week_day

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Została użyta domyślna zmienna systemowa `DateFormat` MM/DD/RRRR.
- Argument `first_week_date` o wartości 3. To sprawia, że tygodnie księżycowe zaczynają się od 3 stycznia.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

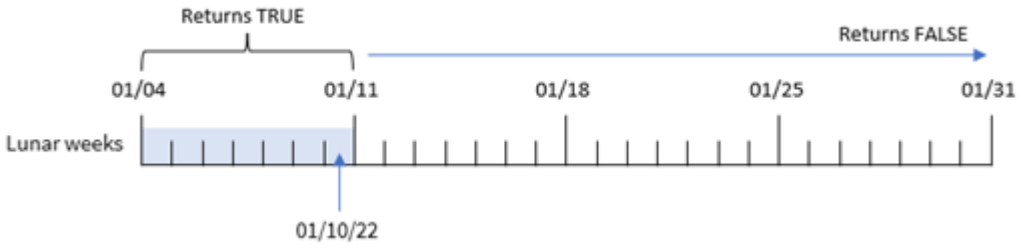
- `date`
- `in_lunar_week_to_date`

Tabela wynikowa

date	in_lunar_week_to_date
1/1/2022	0
1/4/2022	-1

date	in_lunar_week_to_date
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

Funkcja `inlunarweektodate()`, przykład z argumentem first_week_day



W tym przykładzie przekazanie argumentu `the first_week_date` o wartości 3 do do funkcji `inlunarweek()` sprawiło, że pierwszy tydzień księżycowy będzie obejmował dni od 3 do 10 stycznia. Ponieważ 10 stycznia jest także wartością `base_date`, wszelkie transakcje mające miejsce między tymi dwiema datami będą powodować zwrócenie wartości logicznej `TRUE`.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenia określające, czy transakcje miały miejsce w tygodniu księżycowym kończącym się 10 stycznia, jest tworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Utwórz następującą miarę:

```
=inlunarweektodate(date,'01/10/2022', 0)
```

Tabela wynikowa

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

date	=inlunarweektodate(date,'01/10/2022', 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

Funkcja `inlunarweektodate()`, przykład obiektu wykresu

Miara `in_lunar_week_to_date` została utworzona w obiekcie wykresu za pomocą funkcji `inlunarweektodate()` i przez przekazanie jej jako argumentów pola daty, wpisanej w kod daty 10 stycznia jako `base_date` oraz przesunięcia o wartości 0.

Ponieważ tygodnie księżycowe zaczynają się 1 stycznia, data 10 stycznia wypada w tygodniu księżycowym zaczynającym się 8 stycznia, a ponieważ używamy funkcji `inlunarweektodate()`, ten tydzień księżycowy kończyłby się 10 dnia. W związku z tym wszelkie transakcje, które mają miejsce między tymi dwiema styczniowymi datami, zwracałyby wartość `TRUE`. Można to zweryfikować w tabeli wyników.

Przykład 5 – scenariusz

Skrypt ładowania i wyrażenia wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Products`.
- Informacje obejmujące identyfikator produktu, datę produkcji i cenę.

Stwierdzono, że z powodu błędu sprzętowego produkty wytworzone w tygodniu księżycowym obejmującym 12 stycznia były wadliwe. Problem został rozwiązany 13 stycznia. Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał według tygodni informację, czy produkty były wadliwe, czy bez wad oraz koszt produktów wytworzonych w danym tygodniu.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę.
2. Utwórz wymiar, aby pokazać nazwy tygodni:
   `=weekname(manufacture_date)`
3. Następnie utwórz wymiar określający za pomocą funkcji `inlunarweektodate()`, które produkty są wadliwe, a które są bez wad:
   `=if(inlunarweektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')`
4. Utwórz miarę, aby zsumować `cost_price` produktów:
   `=sum(cost_price)`
5. Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

=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

Funkcja `inlunarweektodate()` zwraca wartość logiczną podczas oceny dat wytworzenia każdego z produktów. Dla tych, które zwracają wartość logiczną , oznacza produkty jako .TRUE‘Defective’ W przypadku każdego produktu zwracającego wartość `FALSE`, a zatem niewyprodukowanego w tygodniu księżycowym do 12 stycznia, oznacza ona produkty jako `‘Faultless’`.

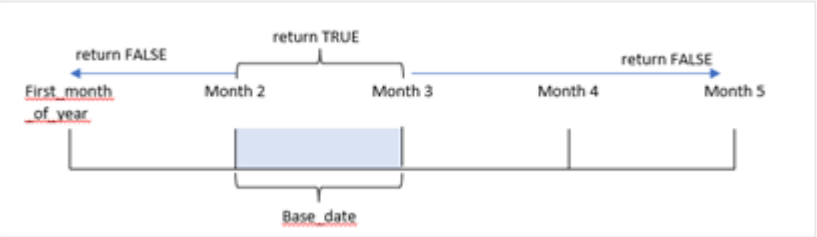
inmonth

Ta funkcja zwraca wartość True, jeśli wartość **timestamp** należy do miesiąca zawierającego wartość **base_date**.

Składnia:

```
InMonth (timestamp, base_date, period_no)
```

Schemat funkcji `indaytotime`.



Innymi słowy, funkcja `inmonth()` określa, czy zestaw dat przypada na ten miesiąc, i zwraca wartość logiczną na podstawie `base_date`, która identyfikuje miesiąc.

Kiedy używać

Funkcja `inmonth()` zwraca wynik logiczny. Zazwyczaj ten typ funkcji będzie używany jako warunek w `if expression`. Zwraca to agregację lub obliczenie w zależności od tego, czy data nastąpiła w danym miesiącu, w tym data, której dotyczy zapytanie.

Na przykład funkcja `inmonth()` może służyć do identyfikacji całego sprzętu wyprodukowanego w danym miesiącu.

Typ zwracanych danych: Wartość logiczna

W Qlik Sense wartość logiczna Prawda jest reprezentowana przez -1, a wartość Fałsz jest reprezentowana przez 0.

Argumenty

Argument	Opis
timestamp	Data, która ma być porównana z wartością `base_date`.
base_date	Data używana do oceny miesiąca. Należy pamiętać, że `base_date` może być dowolnym dniem w miesiącu.
period_no	Miesiąc może zostać przesunięty o wartość `period_no`. `period_no` jest liczbą całkowitą, gdzie 0 oznacza miesiąc zawierający `base_date`. Wartości ujemne parametru `period_no` oznaczają miesiące poprzednie, a wartości dodatnie – miesiące następne.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
`inmonth ('25/01/2013', '01/01/2013', 0)`	Zwraca prawdę
`inmonth('25/01/2013', '23/04/2013', 0)`	Zwraca fałsz
`inmonth ('25/01/2013', '01/01/2013', -1)`	Zwraca fałsz
`inmonth ('25/12/2012', '17/01/2013', -1)`	Zwraca prawdę

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji z pierwszej połowy 2022 r.
- Ładowanie poprzedzające z dodatkową zmienną „in_month”, która określa, czy transakcje miały miejsce w kwietniu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `in_month`

Przykłady funkcji

data	in_month
1/10/2022	0
1/14/2022	0
1/20/2022	0

data	in_month
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

Pole `in_month` zostało utworzone w poprzedniej instrukcji ładowania za pomocą funkcji `inmonth()`, której jako argumenty przekazano pole daty, wpisaną w kod datę 1 kwietnia, jako `base_date` oraz `period_no` o wartości `0`.

`base_date` określa miesiąc, który zwróci wynik logiczny TRUE. Dlatego wszystkie transakcje, które miały miejsce w kwietniu, zwracają TRUE, co jest weryfikowane w tabeli wyników.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie utworzysz pole „`2_months_prior`”, które określa, czy transakcje miały miejsce dwa miesiące przed kwietniem.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `2_months_prior`

Przykłady funkcji

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

Użycie -2 jako argumentu [`period_no` w funkcji `inmonth()` przesuwa miesiąc zdefiniowany przez argument `base_date` dwa miesiące wcześniej. W tym przykładzie zmienia zdefiniowany miesiąc z kwietnia na luty.

Dlatego każda transakcja, która ma miejsce w lutym, zwróci wynik logiczny TRUE.

Przykład 3 – obiekt wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w poprzednim przykładzie.

Jednak w tym przykładzie zbiór danych pozostaje bez zmian i jest ładowany do aplikacji. Obliczenie określające, czy transakcje miały miejsce w kwietniu, jest tworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Obiekt wykresu

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

```
date
```

Aby obliczyć, czy transakcja ma miejsce w kwietniu, utwórz następującą miarę:

```
=inmonth(date,'04/01/2022', 0)
```

Wyniki

	Przykłady funkcji
date	**=inmonth(date,'04/01/2022', 0)**
1/10/2022	0
1/14/2022	0
1/20/2022	0
1/22/2022	0
2/1/2022	0
2/2/2022	0
2/20/2022	0
4/11/2022	-1
4/13/2022	-1
4/15/2022	-1
4/25/2022	-1
5/20/2022	0
5/22/2022	0
6/19/2022	0
6/22/2022	0

Przykład 4 – Scenariusz

Skrypt ładowania i wyniki

Przegląd

W tym przykładzie zestaw danych jest ładowany do tabeli o nazwie `Products`. Tabela zawiera następujące pola:

- Product ID
- Manufacture date
- Cost price

Z powodu błędu sprzętowego produkty wyprodukowane w lipcu 2022 r. były wadliwe. Problem został rozwiązany 27 lipca 2022 r.

Użytkownik końcowy chciałby uzyskać wykres wyświetlający według miesięc status produktów: „defective” (wadliwe; wartość logiczna TRUE) lub „faultless” (bez wad; wartość logiczna FALSE) oraz koszt produktów wytworzonych w danym miesiącu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

```
=monthname(manufacture_date)
```

Utwórz następujące miary:

- `=sum(cost_price)`
- `=if(only(inmonth(manufacture_date,makedate(2022,07,01),0)),'Defective','Faultless')`

1. Ustaw **Formatowanie liczb** miary na **Waluta**.
2. W sekcji **Wygląd** wyłącz **Sumy**.

Tabela wynikowa

monthname (manufacture_date)	=if(only(inmonth(manufacture_date,makedate (2022,07,01),0)),'Defective','Faultless')	sum(cost_ price)
Jan 2022	Faultless	$54.40
Feb 2022	Faultless	$145.69
Mar 2022	Faultless	$53.80

monthname (manufacture_date)	=if(only(inmonth(manufacture_date,makedate (2022,07,01),0)),'Defective','Faultless')	sum(cost_ price)
Apr 2022	Faultless	$82.06
May 2022	Faultless	$127.60
Jun 2022	Faultless	$141.82
Jul 2022	Defective	$214.64
Aug 2022	Faultless	$147.46
Sep 2022	Faultless	$84.21
Oct 2022	Faultless	$163.91

Funkcja `inmonth()` zwraca wartość logiczną podczas oceny dat wytworzenia każdego z produktów. W przypadku każdego produktu wyprodukowanego w lipcu 2022 funkcja `inmonth()` zwraca wartość logiczną True i oznacza produkty jako „Defective" (Wadliwe). W przypadku każdego produktu zwracającego wartość False, a zatem niewyprodukowanego w lipcu, oznacza ona produkty jako „Faultless" (Bez wad).

inmonths

Ta funkcja sprawdza, czy znacznik czasu mieści się w tym samym okresie miesięcznym, dwumiesięcznym, kwartalnym, czteromiesięcznym lub półrocznym, jako data bazowa. Można także wyszukać, czy znacznik czasu przypada w okresie poprzednim lub następnym.

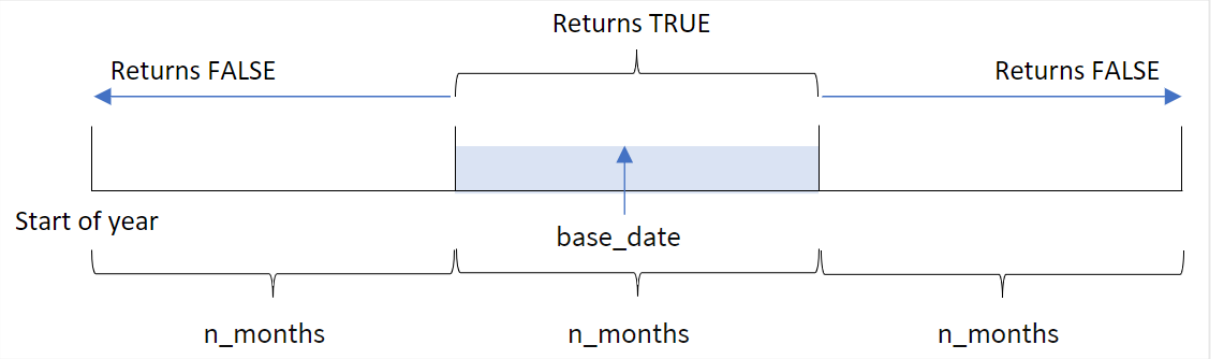
Składnia:

```
InMonths(n_months, timestamp, base_date, period_no [, first_month_of_year])
```

Typ zwracanych danych: Wartość logiczna

W Qlik Sense wartość logiczna Prawda jest reprezentowana przez -1, a wartość Fałsz jest reprezentowana przez 0.

Schemat funkcji `inmonths()`



Funkcja `inmonths()` dzieli rok na segmenty na podstawie przekazanego argumentu `n_months`. Następnie określa, czy każdy z ocenianych znaczników czasu wypada w tym samym segmencie, co argument `base_date`. Jeśli jednak zostanie przekazany argument `period_no`, funkcja określa, czy znaczniki czasu wypadają w poprzednim lub następnym okresie od `base_date`.

Następujące segmenty roku są dostępne w funkcji jako argumenty `n_month`.

Argumenty funkcji `n_month`

Okres	Liczba miesięcy
miesiąc	1
dwa miesiące	2
kwartał	3
cztery miesiące	4
pół roku	6

Kiedy używać

Funkcja `inmonths()` zwraca wynik logiczny. Zazwyczaj ten typ funkcji będzie używany jako warunek w `if expression`. Za pomocą funkcji `inmonths()` możesz wybrać okres, który chcesz sprawdzić. Na przykład możesz pozwolić użytkownikowi zidentyfikować produkty wytworzone w miesiącu, kwartale lub półroczu określonego okresu.

Typ zwracanych danych: wartość logiczna

W Qlik Sense wartość logiczna Prawda jest reprezentowana przez -1, a wartość Fałsz jest reprezentowana przez 0.

Argumenty

Argument	Opis
n_months	Liczba miesięcy określająca okres. Wartość całkowita lub wyrażenie, którego wynikiem jest jedna z następujących wartości całkowitych: 1 (równoważnik funkcji inmonth()), 2 (dwa miesiące), 3 (równoważnik funkcji inquarter()), 4 (cztery miesiące) lub 6 (pół roku).
timestamp	Data, która ma być porównana z wartością **base_date**.
base_date	Data używana do oceny okresu.
period_no	Okres może być przesunięty o wartość **period_no** – liczbę całkowitą lub wyrażenie, którego wynikiem jest liczba całkowita, gdzie wartość 0 wskazuje dzień zawierający wartość **base_date**. Wartości ujemne parametru **period_no** oznaczają okresy poprzednie, a wartości dodatnie – okresy następne.
first_month_of_year	Jeśli użytkownik zamierza korzystać z lat (obrotowych), które nie zaczynają się w styczniu, powinien wskazać wartość od 2 do 12 jako parametr **first_month_of_year**.

Aby ustawić pierwszy miesiąc roku w argumencie `first_month_of_year`, możesz użyć następujących wartości:

first_month_of_year values

Miesiąc	Wartość
Luty	2
Marzec	3
Kwiecień	4
May	5
Czerwiec	6
Lipiec	7
Sierpień	8
Wrzesień	9
Październik	10
Listopad	11
Grudzień	12

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
`inmonths(4, '01/25/2013', '04/25/2013', 0)`	Zwraca wartość TRUE. Ponieważ wartość znacznika, 01/25/2013, zawiera się w okresie czteromiesięcznym od 01/01/2013 do 04/30/2013, w którym przypada `base_date`, 04/25/2013.

Przykład	Wynik
`inmonths(4, '05/25/2013', '04/25/2013', 0)`	Zwraca wartość FALSE. Ponieważ data 05/25/2013 przypada poza tym samym okresem co poprzedni przykład.
`inmonths(4, '11/25/2012', '02/01/2013', -1 )`	Zwraca wartość TRUE. Ponieważ wartość `period_no`, `-1` powoduje przesunięcie okresu wyszukiwania w tył o jeden okres czterech miesięcy (wartość n-months), przez co okres wyszukiwania przypada od 09/01/2012 do 12/31/2012.
`inmonths(4, '05/25/2006', '03/01/2006', 0, 3)`	Zwraca wartość TRUE. Ponieważ wartość `first_month_of_year` jest ustawiona na `3`, co sprawia, że okres wyszukiwania jest od 03/01/2006 do 07/30/2006 zamiast od 01/01/2006 do 04/30/2006.

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Poprzedzające ładowanie z dodatkową zmienną `in_months`, która określa, które transakcje miały miejsce w tym samym kwartale, w którym wypada 15 maja 2022 roku.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `in_months`

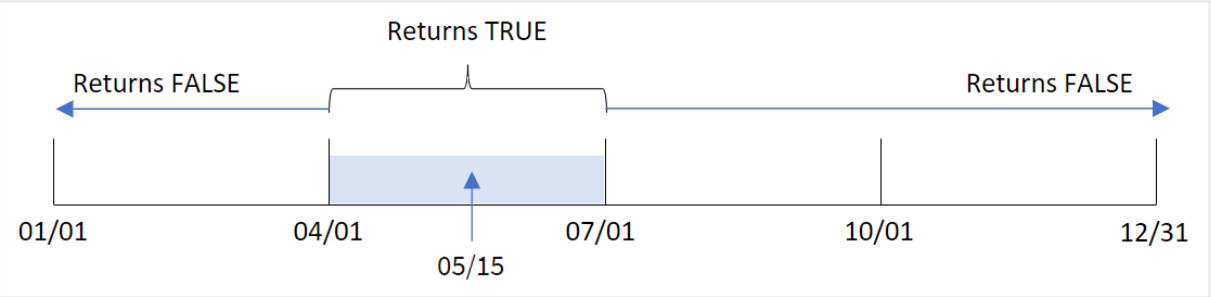
Tabela wynikowa

date	in_months
2/19/2022	0
3/7/2022	0
3/30/2022	0
4/5/2022	-1
4/16/2022	-1
5/1/2022	-1
5/7/2022	-1
5/22/2022	-1
6/15/2022	-1
6/26/2022	-1
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0

date	in_months
9/26/2022	0
10/14/2022	0
10/29/2022	0

Pole `in_months` jest tworzone w poprzedniej instrukcji ładowania przy użyciu funkcji `inmonths()`. Pierwszy podany argument to `3`. Dzieli on rok na kwartały. Drugi argument określa, które pole jest ewaluowane - pole daty w tym przypadku. Trzeci argument to wpisana bezpośrednio w kod data 15 maja, która jest datą `base_date`. Ostatnim argumentem jest `period_no` o wartości `0`.

Schemat funkcji `inmonths()` z segmentami kwartalnymi



Maj wypada w drugim kwartale roku. W związku z tym każda transakcja, która ma miejsce między 1 kwietnia a 30 czerwca, spowoduje zwrócenie w wyniku wartości logicznej TRUE. Można to zweryfikować w tabeli wyników.

Przykład 2 - period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Poprzedzające ładowanie z dodatkową zmienną `previous_quarter`, które określa, czy transakcje miały miejsce w kwartale przed 15 maja 2022 roku.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `previous_quarter`

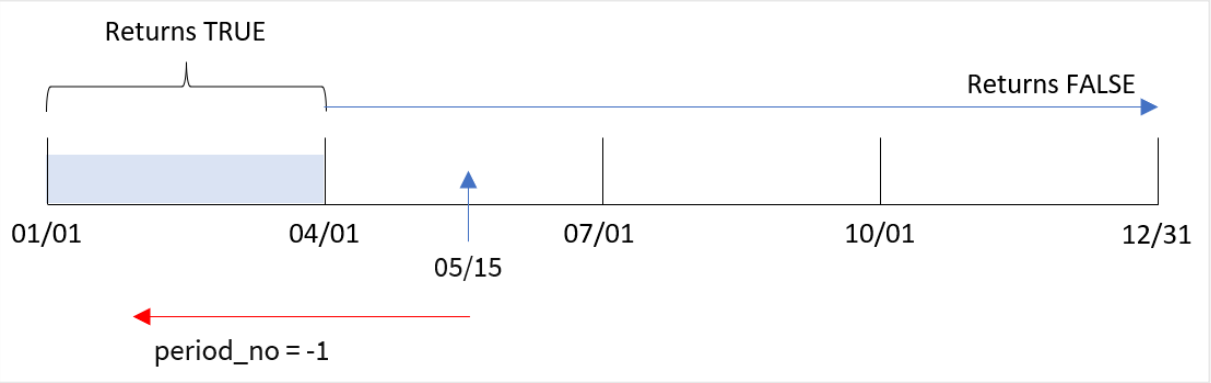
Tabela wynikowa

date	poprzedni kwartał
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	poprzedni kwartał
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

Funkcja sprawdza, czy transakcje zostały zawarte w pierwszym kwartale roku za pomocą wywołania funkcji `inmonths()` z argumentem `period_no` o wartości `-1`. 15 maja to data `base_date`, która wypada w drugim kwartale roku (kwiecień-czerwiec).

Schemat funkcji `inmonths()` z segmentami kwartalnymi i argumentem `period_no` ustawionym na `-1`



W związku z tym każda transakcja zawarta między styczniem i marcem spowoduje zwrócenie w wyniku logicznej prawdy.

Przykład 3 – first_month_of_year

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Poprzedzające ładowanie z dodatkową zmienną `in_months`, która określa, które transakcje miały miejsce w tym samym kwartale, w którym wypada 15 maja 2022 roku.

W tym przykładzie przyjęto, że marzec ma być pierwszym miesiącem roku podatkowego.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

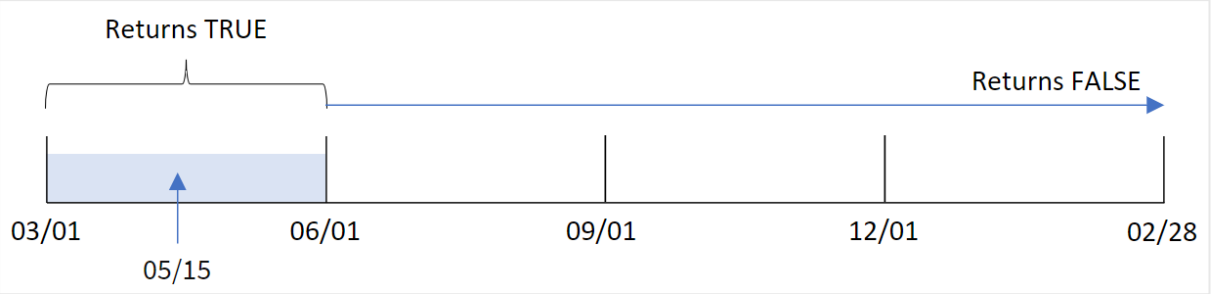
- `date`
- `in_months`

Tabela wynikowa

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

Przekazując wartość `3` jako argument `first_month_of_year` funkcji `inmonths()`, sprawiamy że funkcja ta uznaje za początek roku 1 marca. Następnie funkcja `inmonths()` dzieli rok na kwartały: marzec-maj, czerwiec-sierpień, wrzesień-listopad, grudzień-luty. W związku z tym 15 maja wypada w pierwszym kwartale roku (marzec-maj).

Diagram funkcji `inmonths()` z marcem ustawionym jako pierwszy miesiąc roku.



Każda transakcja zawarta w tych miesiącach spowoduje zwrócenie logicznej prawy.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie zestaw danych pozostaje bez zmian i jest ładowany do aplikacji. Obliczenia określające, czy transakcje miały miejsce w tym samym kwartale, w którym wypada 15 maja, jest tworzone jako miara w wykresie w aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

- `date`

Aby obliczyć, czy transakcje zostały zawarte w tym samym kwartale, w którym wypada 15 maja, utwórz następującą miarę:

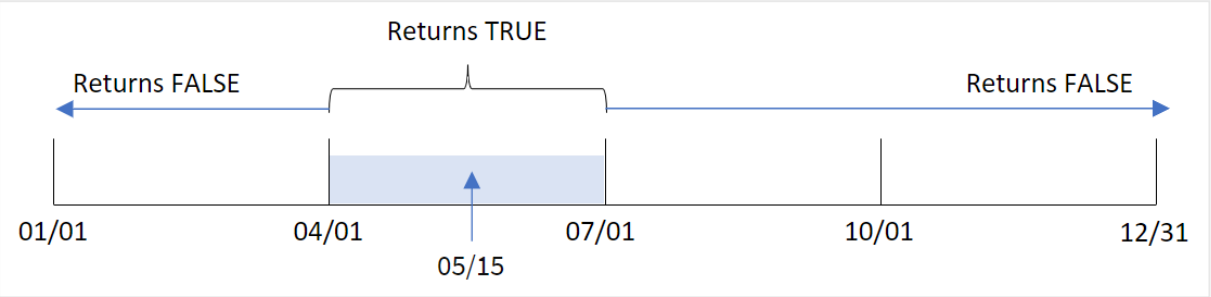
```
=inmonths(3,date,'05/15/2022', 0)
```

Tabela wynikowa

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

Pole `in_months` jest tworzone w wykresie za pomocą funkcji `inmonths()`. Pierwszy podany argument to `3`. Dzieli on rok na kwartały. Drugi argument określa, które pole jest ewaluowane – pole daty w tym przypadku. Trzeci argument to wpisana bezpośrednio w kod data 15 maja, która jest datą `base_date`. Ostatnim argumentem jest `period_no` o wartości `0`.

Schemat funkcji `inmonths()` z segmentami kwartalnymi



Maj wypada w drugim kwartale roku. W związku z tym każda transakcja, która ma miejsce między 1 kwietnia a 30 czerwca, spowoduje zwrócenie w wyniku wartości logicznej TRUE. Można to zweryfikować w tabeli wyników.

Przykład 5 – scenariusz

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Products`.
- Tabela zawiera następujące pola:
  - identyfikator produktu,
  - typ produktu,
  - data produkcji,
  - cena.

Użytkownik chciałby, aby wykres wyświetlał według typu produktu koszt produktów wyprodukowanych w pierwszym segmencie roku 2021. Użytkownik chciałby móc zdefiniować długość tego segmentu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz.

Na początku skryptu ładowania została utworzona zmienna `vPeriod`, która jest powiązana z kontrolką wejściową zmiennej.

Wykonaj następujące czynności:

1. W panelu zasobów kliknij opcję **Obiekty niestandardowe**.
2. Wybierz **pakiet Qlik Dashboard** i utwórz obiekt **wprowadzania zmiennych**.
3. Wprowadź tytuł obiektu wykresu.
4. W polu **Zmienna** wybierz **vPeriod** jako nazwę i ustaw obiekt tak, aby był wyświetlany jako **lista rozwijana**.
5. W sekcji **Wartości** kliknij **Wartości** dynamiczne. Wprowadź:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`
6. Dodaj nową tabelę do arkusza.
7. W sekcji **Dane** w okienku właściwości dodaj `product_type` jako wymiar.
8. Dodaj następujące wyrażenie jako miarę:
   `=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))`
9. Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

product_type	**=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))**
product A	$88.27

product_type	=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))
product B	$37.23
product C	$17.17
product D	$0.00

Funkcja `inmonths()` używa danych wprowadzonych przez użytkownika jako swojego argumentu określającego rozmiar początkowego segmentu roku. Funkcja przekazuje datę produkcji każdego z produktów jako drugi argument funkcji `inmonths()`. Przekazanie daty 1 stycznia jako trzeciego argumentu funkcji `inmonths()` powoduje, że produkty o dacie produkcji wypadającej w otwierającym segmencie roku będą powodować zwrot w wyniku logicznej prawdy, co będzie skutkowało sumowaniem ich kosztów.

inmonthstodate

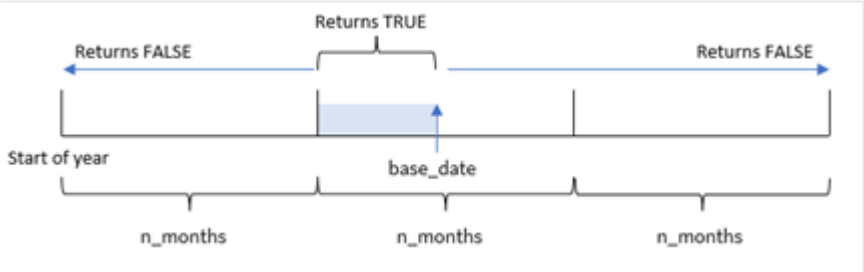
Ta funkcja wyszukuje, czy znacznik czasu przypada w części okresu miesiąca, dwóch miesięcy, kwartału, czterech miesięcy lub półrocza z dokładnością do jednej milisekundy wartości `base_date` włącznie. Można także wyszukać, czy znacznik czasu przypada w okresie poprzednim lub następnym.

Składnia:

```
InMonths (n_months, timestamp, base_date, period_no[, first_month_of_year ])
```

Typ zwracanych danych: Wartość logiczna

Schemat funkcji `inmonthstodate`.



Argumenty

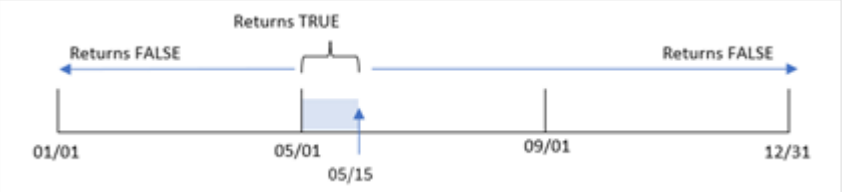
Argument	Opis
n_months	Liczba miesięcy określająca okres. Wartość całkowita lub wyrażenie, którego wynikiem jest jedna z następujących wartości całkowitych: 1 (równoważnik funkcji inmonth()), 2 (dwa miesiące), 3 (równoważnik funkcji inquarter()), 4 (cztery miesiące) lub 6 (pół roku).
timestamp	Data, która ma być porównana z wartością **base_date**.
base_date	Data używana do oceny okresu.

Argument	Opis
period_no	Okres może być przesunięty o wartość **period_no** – liczbę całkowitą lub wyrażenie, którego wynikiem jest liczba całkowita, gdzie wartość 0 wskazuje dzień zawierający wartość **base_date**. Wartości ujemne parametru **period_no** oznaczają okresy poprzednie, a wartości dodatnie – okresy następne.
first_month_of_year	Jeśli użytkownik zamierza korzystać z lat (obrotowych), które nie zaczynają się w styczniu, powinien wskazać wartość od 2 do 12 jako parametr **first_month_of_year**.

W funkcji `inmonthstodate()` data `base_date` pełni rolę punktu końcowego określonego segmentu roku, którego jest częścią.

Na przykład, jeśli rok zostanie podzielony na trzy części, a data `base_date` zostanie ustawiona na 15 maja, to każdy znacznik czasu mieszczący się między początkiem stycznia a końcem kwietnia będzie powodował zwrócenie logicznego fałszu. Daty między 1 maja i 15 maja będą dawać w wyniku prawdę. Dla pozostałej części roku będzie zwracany fałsz.

Schemat wyników logicznych zwracanych przez funkcję `inmonthstodate`.



Następujące segmenty roku są dostępne w funkcji jako argumenty `n_month`:

Argumenty funkcji `n_month`

Okres	Liczba miesięcy
miesiąc	1
dwa miesiące	2
kwartał	3
cztery miesiące	4
pół roku	6

Kiedy używać

Funkcja `inmonthstodate()` zwraca wynik logiczny. Zazwyczaj ten typ funkcji jest używany jako warunek w `if expression`. Za pomocą funkcji `inmonthstodate()` możesz wybrać okres, który chcesz sprawdzić. Na przykład, przekazując zmienną wejściową, która pozwala użytkownikowi zidentyfikować produkty wytworzone w miesiącu, kwartale lub półroczu okresu do określonej daty.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
`inmonthstodate(4, '01/25/2013', '04/25/2013', 0)`	Zwraca True, ponieważ wartość timestamp, 01/25/2013, mieści się w czteromiesięcznym okresie od 01/01/2013 do końca 04/25/2013, który obejmuje wartość 04/25/2013 argumentu base_date.
`inmonthstodate(4, '04/26/2013', '04/25/2006', 0)`	Zwraca False, ponieważ 04/26/2013 znajduje się poza okresem z poprzedniego przykładu.
`inmonthstodate(4, '09/25/2005', '02/01/2006', -1)`	Zwraca True, ponieważ wartość period_no, -1, przesuwa okres wyszukiwania w tył o jeden czteromiesięczny okres (wartość n-months), co sprawia, że okres wyszukiwania obejmuje daty od 01/09/2005 do 02/01/2006.
`inmonthstodate(4, '04/25/2006', '06/01/2006', 0, 3)`	Zwraca True, ponieważ wartość first_month_of_year jest ustawiona na 3, co zmienia okres wyszukiwania na zakres dat od 03/01/2006 do 06/01/2006 zamiast od 05/01/2006 do 06/01/2006.

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Pole daty, które jest przekazywane w formacie `(MM/DD/YYYY)` zmiennej systemowej `DateFormat`.
- Poprzedzająca instrukcja ładowania zawiera:

- Funkcję `inmonthstodate()`, która jest ustawiona jako pole `in_months_to_date`. Określa ona transakcje, które zostały zawarte w kwartale do 15 maja 2022 roku.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `in_months_to_date`

Tabela wynikowa

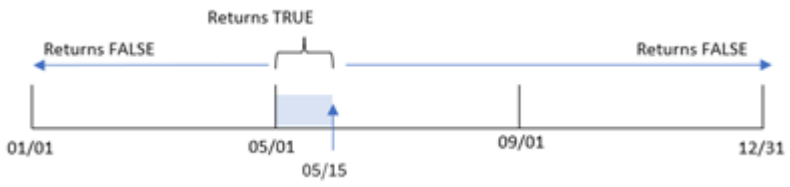
date	in_months_to_date
1/7/2022	0
1/19/2022	0

date	in_months_to_date
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

Pole `in_months_to_date` jest tworzone w poprzedniej instrukcji ładowania przy użyciu funkcji `inmonthstodate()`.

Pierwszy podany argument to `3`. Dzieli on rok na kwartały. Pierwszy podany argument określa, które pole jest oceniane. Trzeci argument to wpisana bezpośrednio w kod data 15 maja – jest to argument `base_date`, który określa datę końcową segmentu. Ostatni argument to `period_no` o wartości `0`.

Diagram funkcji `inmonthstodate` bez dodatkowych argumentów.



Każda transakcja, która ma miejsce w okresie od 1 kwietnia do 15 maja, zwraca w wyniku logiczną prawdę. Daty wypadające poza tym okresem zwracają fałsz.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie chodzi o utworzenie pola, `previous_qtr_to_date`, które będzie sprawdzało, czy transakcje były zawierane kwartał przed 15 maja.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `previous_qtr_to_date`

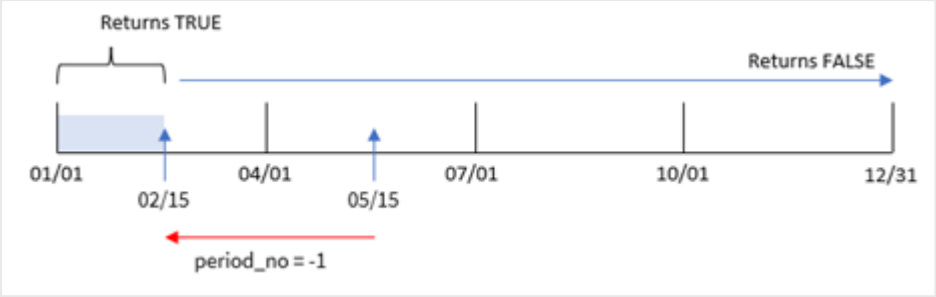
Tabela wynikowa

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

Po otrzymaniu wartości `-1` jako argumentu `period_no` funkcja `inmonthstodate()` przesuwa granice segmentu roku porównawczego o kwartał.

15 maja wypada w drugim kwartale roku, a więc początkowo segment obejmuje daty od 1 kwietnia do 15 maja. Argument `period_no` przesuwa ten segment wstecz o trzy miesiące. Zakres dat zmienia się na okres od 1 stycznia do 15 lutego.

Diagram funkcji `inmonthstodate` z argumentem period_no ustawionym na -1.



W związku z tym każda transakcja, która ma miejsce między 1 stycznia a 15 lutego, zwróci w wyniku logiczną prawdę.

Przykład 3 – first_month_of_year

Skrypt ładowania i wyniki

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

W tym przykładzie przyjęto, że marzec ma być pierwszym miesiącem roku podatkowego.

Utwórz pole, `in_months_to_date`, które określa, jakie transakcje miały miejsce w tym samym kwartale do 15 maja 2022 r.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `in_months_to_date`

Tabela wynikowa

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

Przekazując wartość `3` jako argument `first_month_of_year` funkcji `inmonthstodate()`, sprawiamy że funkcja ta uznaje za początek roku 1 marca, a następnie dzieli rok na kwartały na podstawie pierwszego przekazanego argumentu. Dlatego segmenty kwartałów to:

- marzec-maj,
- czerwiec-sierpień,
- wrzesień-listopad,
- grudzień-luty.

Argument `base_date` o wartości 15 maja dzieli następnie kwartał trwający od marca do maja przez ustawienie jego daty granicznej na 15 maja.

Diagram funkcji `inmonthstodate` z marcem ustawionym jako pierwszy miesiąc roku.

W związku z tym każda transakcja zawarta między 1 marca a 15 maja spowoduje zwrót w wyniku logicznej prawdy, podczas gdy transakcje z datami poza tymi granicami spowodują zwrot fałszu.

Przykład – przykład z wykresem

Skrypt ładowania i wyrażenie wykresu

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

W tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Zadanie polega na stworzeniu obliczeń określających, czy transakcje miały miejsce w tym samym kwartale, w którym mieści się 15 maja, jako miary na wykresie aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

```
date
```

Aby obliczyć, czy transakcje zostały zawarte w tym samym kwartale, w którym wypada 15 maja, utwórz następującą miarę:

```
=inmonthstodate(3,date,'05/15/2022', 0)
```

Tabela wynikowa

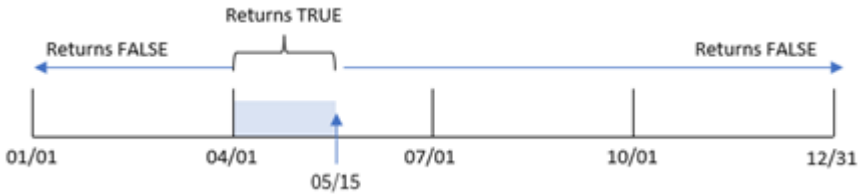
date	=inmonthstodate(3,date,'05/15/2022', 0)
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0

date	=inmonthstodate(3,date,'05/15/2022', 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

Miara `'in_months_to_date` jest tworzona na wykresie przy użyciu funkcji `inmonthstodate()`.

Pierwszy podany argument to `3`. Dzieli on rok na kwartały. Pierwszy podany argument określa, które pole jest oceniane. Trzeci argument to wpisana bezpośrednio w kod data 15 maja – jest to argument `base_date`, który określa datę końcową segmentu. Ostatni argument to `period_no` o wartości `0`.

Schemat funkcji `inmonthstodate` z segmentami kwartalnymi.



Każda transakcja, która ma miejsce w okresie od 1 kwietnia do 15 maja, spowoduje zwrot w wyniku logicznej prawdy. Daty wypadające poza tym okresem powodują zwrot fałszu.

Przykład 5 – scenariusz

Skrypt ładowania i wyniki

Przegląd

W tym przykładzie zestaw danych jest ładowany do tabeli o nazwie `Sales`. Tabela zawiera następujące pola:

- identyfikator produktu,
- typ produktu,
- data sprzedaży,
- cena sprzedaży.

Użytkownik chce, aby wykres pokazywał według typu produktu sprzedaż produktów w okresie do 24 grudnia 2022 roku. Użytkownik chciałby móc zdefiniować długość tego okresu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz.

Na początku skryptu ładowania została utworzona zmienna `vPeriod`, która jest powiązana z kontrolką wejściową zmiennej.

Wykonaj następujące czynności:

1. W panelu zasobów kliknij opcję **Obiekty niestandardowe**.
2. Wybierz **pakiet Qlik Dashboard** i dodaj **Wprowadzanie zmiennych** do swojego arkusza.
3. Wpisz tytuł wykresu.
4. W polu **Zmienna** wybierz **vPeriod** jako nazwę i ustaw obiekt tak, aby był wyświetlany jako **lista rozwijana**.
5. W sekcji **Wartości** kliknij **Wartości** dynamiczne. Wprowadź: `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`
6. Dodaj nową tabelę do arkusza.
7. W sekcji **Dane** w okienku właściwości dodaj `product_type` jako wymiar.

8. Dodaj następujące wyrażenie jako miarę:
   `=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24),0),sales_price,0))`
9. Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

product_type	=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24),0),sales_price,0))
product A	$186.48
product B	$190.52
product C	$220.43
product D	$261.46

Funkcja `inmonthstodate()` używa danych wprowadzonych przez użytkownika jako swojego argumentu określającego rozmiar początkowego segmentu roku.

Funkcja przekazuje datę sprzedaży każdego z produktów jako drugi argument funkcji `inmonthstodate()`. Przekazanie daty 24 grudnia jako trzeciego argumentu do funkcji `inmonthstodate()` powoduje, że funkcja ta zwraca w wyniku logiczną prawdę dla produktów, które zostały sprzedane w zdefiniowanym okresie do 24 grudnia włącznie. Funkcja sum sumuje wartości sprzedaży tych produktów.

inmonthtodate

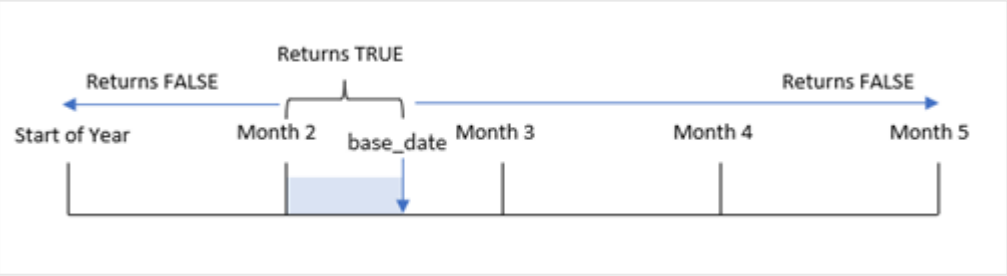
Zwraca wartość True, jeśli wartość **date** należy do części miesiąca zawierającego wartość **basedate** z dokładnością do jednej milisekundy wartości **basedate** włącznie.

Składnia:

```
InMonthToDate (timestamp, base_date, period_no)
```

Typ zwracanych danych: Wartość logiczna

Schemat funkcji `inmonthtodate`.



Funkcja `inmonthtodate()` identyfikuje określony miesiąc jako segment. Granica początkowa to początek miesiąca. Granica końcowa może zostać ustawiona na późniejszą datę w miesiącu. Następnie sprawdza, czy zestaw dat mieści się w tym zakresie, zwracając w wyniku logiczną prawdę lub logiczny fałsz.

Argumenty

Argument	Opis
timestamp	Data, która ma być porównana z wartością **base_date**.
base_date	Data używana do oceny miesiąca.
period_no	Miesiąc może zostać przesunięty o wartość **period_no**. **period_no** jest liczbą całkowitą, gdzie 0 oznacza miesiąc zawierający wartość **base_date**. Wartości ujemne parametru **period_no** oznaczają miesiące poprzednie, a wartości dodatnie – miesiące następne.

Kiedy używać

Funkcja `inmonthtodate()` zwraca wynik logiczny. Zazwyczaj ten typ funkcji jest używany jako warunek w `if expression`. Funkcja `inmonthtodate()` zwraca agregację lub obliczenia, które zależą od tego, czy data wypada w miesiącu do interesującej nas daty włącznie.

Na przykład funkcja `inmonthtodate()` może służyć do identyfikacji całego sprzętu wyprodukowanego w miesiącu do określonej daty.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
`inmonthtodate ('01/25/2013', '25/01/2013', 0)`	Zwraca wartość True
`inmonthtodate ('01/25/2013', '24/01/2013', 0)`	Zwraca wartość False
`inmonthtodate ('01/25/2013', '28/02/2013', -1)`	Zwraca wartość True

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Pole daty zostało przekazane w formacie zmiennej systemowej `DateFormat` (`MM/DD/YYYY`).
- Poprzedzająca instrukcja ładowania zawiera:
    - funkcję `inmonthtodate()`, która jest ustawiona jako pole `in_month_to_date`. Pozwala to sprawdzić, które transakcje zostały zawarte między 1 a 26 lipca 2022 roku.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `in_month_to_date`

Tabela wynikowa

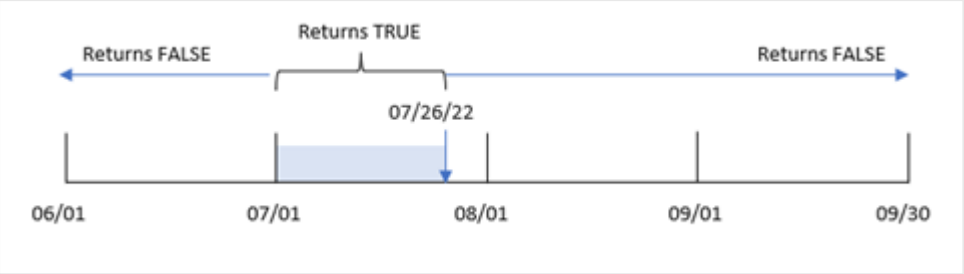
date	in_month_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	-1
7/22/2022	-1
7/23/2022	-1
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Pole `in_month_to_date` jest tworzone w poprzedniej instrukcji load przy użyciu funkcji `inmonthtodate()`.

Pierwszy argument określa, które pole jest oceniane. Drugi argument to wpisana bezpośrednio w kod data 26 lipca, która jest wartością argumentu `base_date`. Ten argument `base_date` określa, który miesiąc jest segmentowany i końcową granicę tego segmentu.

Argument `period_no` o wartości `0` jest ostatni, co oznacza, że funkcja nie porównuje miesięcy poprzedzających segmentowany miesiąc ani następujących po nim.

Diagram funkcji `inmonthtodate` bez dodatkowych argumentów.



W efekcie wszelkie transakcje, które występują między 1 a 26 lipca, powodują zwrot w wyniku logicznej prawdy. Każda transakcja zawarta w lipcu po 26 lipca powoduje zwrot logicznego fałszu, podobnie jak każda transakcja zawarta w jakimkolwiek innym miesiącu roku.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

W tym przykładzie chodzi o utworzenie pola, `six_months_prior`, sprawdzającego, które transakcje zostały zawarte pełne sześć miesięcy przed 1 i 26 lipca.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

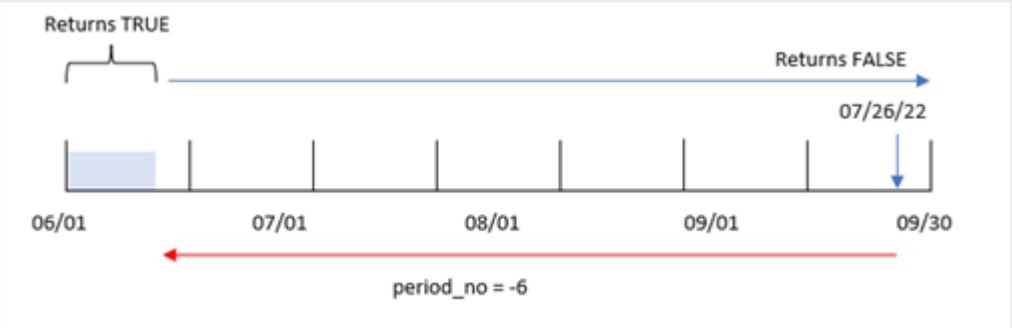
- `date`
- `six_months_prior`

Tabela wynikowa

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

Po otrzymaniu wartości `-6` jako argumentu `period_no` funkcja `inmonthtodate()` przesuwa granice segmentu miesiąca porównawczego o sześć miesięcy. Początkowo segment miesiąca jest równy zakresowi dat od 1 do 26 lipca. Następnie `period_no` przesuwa go wstecz o sześć miesięcy, w efekcie czego zostają przesunięte granice dat, które od tej pory wypadają między 1 i 26 stycznia.

Diagram funkcji `inmonthtodate` z argumentem period_no ustawionym na -6.



W efekcie wszelkie transakcje zawarte między 1 i 26 stycznia będą powodować zwrot w wyniku logicznej prawdy.

Przykład 3 – przykład z wykresem

Skrypt ładowania i wyrażenie wykresu

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

W tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Zadanie polega na utworzeniu obliczeń określających, czy transakcje zostały zawarte między 1 a 26 lipca, jako miary w wykresie aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

```
date
```

Aby obliczyć, czy transakcje zostały zawarte między 1 a 26 lipca, utwórz następującą miarę:

```
=inmonthtodate(date,'07/26/2022', 0)
```

Tabela wynikowa

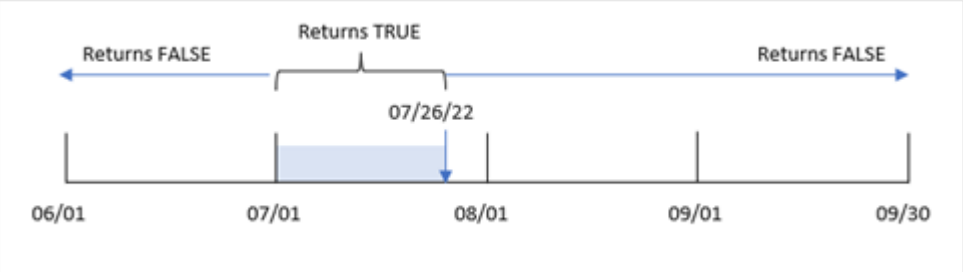
date	=inmonthtodate(date,'07/26/2022', 0)
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	-1
7/22/2022	-1
7/23/2022	-1
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0

date	=inmonthtodate(date,'07/26/2022', 0)
9/26/2022	0
10/14/2022	0
10/29/2022	0

Miara pola `in_month_to_date` została utworzona na wykresie za pomocą funkcji `inmonthtodate()`.

Pierwszy argument określa, które pole jest oceniane. Drugi argument to wpisana bezpośrednio w kod data 26 lipca, która jest wartością argumentu `base_date`. Ten argument base_date określa, który miesiąc jest segmentowany i końcową granicę tego segmentu. Ostatni argument to `period_no` o wartości 0. To znaczy, że funkcja nie porównuje miesięcy poprzedzających segmentowany miesiąc, ani po nim następujących.

Diagram funkcji `inmonthtodate` bez dodatkowych argumentów.



W efekcie wszelkie transakcje, które występują między 1 a 26 lipca, powodują zwrot w wyniku logicznej prawdy. Każda transakcja zawarta w lipcu po 26 lipca powoduje zwrot logicznego fałszu, podobnie jak każda transakcja zawarta w jakimkolwiek innym miesiącu roku.

Przykład 4 – Scenariusz

Skrypt ładowania i wyniki

Przegląd

W tym przykładzie zestaw danych jest ładowany do tabeli o nazwie `Products`. Tabela zawiera następujące pola:

- Product ID
- Manufacture date
- Cost price

Z powodu błędu sprzętowego produkty wyprodukowane w lipcu 2022 r. były wadliwe. Problem został rozwiązany 27 lipca 2022 r.

Użytkownik końcowy chciałby uzyskać wykres wyświetlający według miesięc status produktów: „defective” (wadliwe; wartość logiczna TRUE) lub „faultless” (bez wad; wartość logiczna FALSE) oraz koszt produktów wytworzonych w danym miesiącu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `=monthname(manufacture_date)`
- `=if(Inmonthtodate(manufacture_date,makedate(2022,07,26),0),'Defective','Faultless')`

Aby obliczyć sumę kosztów produktów, utwórz następującą miarę:

```
=sum(cost_price)
```

Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

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

monthname (manufacture_date)	if(Inmonthtodate(manufacture_date,makedate (2022,07,26),0),'Defective','Faultless')	Sum(cost_ price)
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

Funkcja `inmonthtodate()` zwraca wartość logiczną podczas oceny dat wytworzenia każdego z produktów.

Dla dat powodujących zwrot logicznej prawdy produkt jest oznaczany jako wadliwy (Defective). Dla każdego produktu powodującego zwrot wartości FALSE, a zatem niewyprodukowanego w lipcu do 26. dnia tego miesiąca włącznie, funkcja dodaje oznaczenie „Faultless” (Bez wad).

inquarter

Ta funkcja zwraca wartość True, jeśli wartość **timestamp** należy do kwartału zawierającego wartość **base_date**.

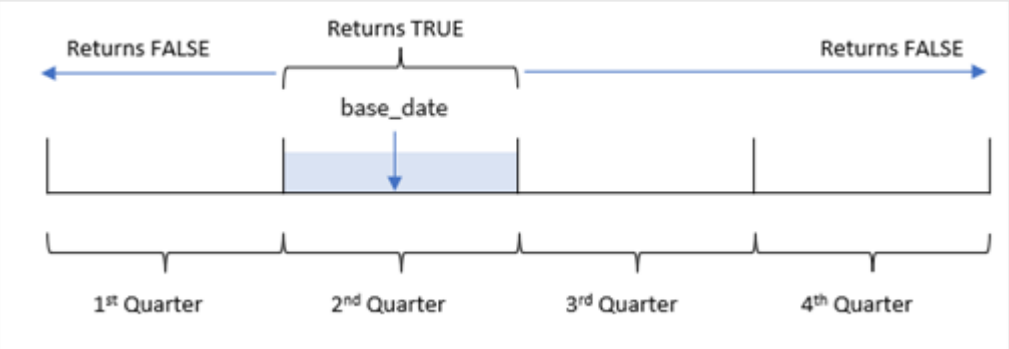
Składnia:

```
InQuarter (timestamp, base_date, period_no[, first_month_of_year])
```

Typ zwracanych danych: Wartość logiczna

W Qlik Sense wartość logiczna Prawda jest reprezentowana przez -1, a wartość Fałsz jest reprezentowana przez 0.

Diagram zakresu funkcji `inquarter()`



Innymi słowy, funkcja `inquarter()` dzieli rok na cztery równe kwartały między 1 stycznia a 31 grudnia. Za pomocą argumentu `first_month_of_year` można zmienić pierwszy miesiąc w aplikacji i wówczas kwartały zostaną odpowiednio dostosowane. Argument `base_date` – funkcja określa, który kwartał powinien zostać użyty jako komparator dla funkcji. Na koniec funkcja zwraca wynik logiczny porównywania wartości dat z tym segmentem kwartału.

Kiedy używać

Funkcja `inquarter()` zwraca wynik logiczny. Zazwyczaj ten typ funkcji będzie używany jako warunek w `if expression`. W wyniku zwraca agregację lub obliczenia zależne od tego, czy data wypada w określonym kwartale.

Na przykład za pomocą funkcji `inquarter()` można zidentyfikować cały sprzęt wyprodukowany w segmencie kwartału na podstawie dat produkcji tego sprzętu.

Argumenty

Argument	Opis
timestamp	Data, która ma być porównana z wartością **base_date**.
base_date	Data używana do oceny kwartału.
period_no	Kwartał może zostać przesunięty o wartość **period_no**. **period_no** jest liczbą całkowitą, gdzie 0 oznacza kwartał zawierający wartość **base_date**. Wartości ujemne parametru **period_no** oznaczają kwartały poprzednie, a wartości dodatnie – kwartały następne.
first_month_of_year	Jeśli użytkownik zamierza korzystać z lat (obrotowych), które nie zaczynają się w styczniu, powinien wskazać wartość od 2 do 12 jako parametr **first_month_of_year**.

Aby ustawić pierwszy miesiąc roku w argumencie `first_month_of_year`, możesz użyć następujących wartości:

first_month_of_year values

Miesiąc	Wartość
Luty	2
Marzec	3
Kwiecień	4
May	5
Czerwiec	6
Lipiec	7
Sierpień	8
Wrzesień	9
Październik	10
Listopad	11
Grudzień	12

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
inquarter ('01/25/2013', '01/01/2013', 0)	Zwraca TRUE
inquarter ('01/25/2013', '04/01/2013', 0)	Zwraca FALSE
inquarter ('01/25/2013', '01/01/2013', -1)	Zwraca FALSE
inquarter ('12/25/2012', '01/01/2013', -1)	Zwraca TRUE
inquarter ('01/25/2013', '03/01/2013', 0, 3)	Zwraca FALSE
inquarter ('03/25/2013', '03/01/2013', 0, 3)	Zwraca TRUE

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Poprzedzające ładowanie zawiera funkcję `inquarter()` ustawioną jako pole `in_quarter` i określa, które transakcje zostały zawarte w tym samym kwartale, do którego należy data 15 maja 2022 roku.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `in_quarter`

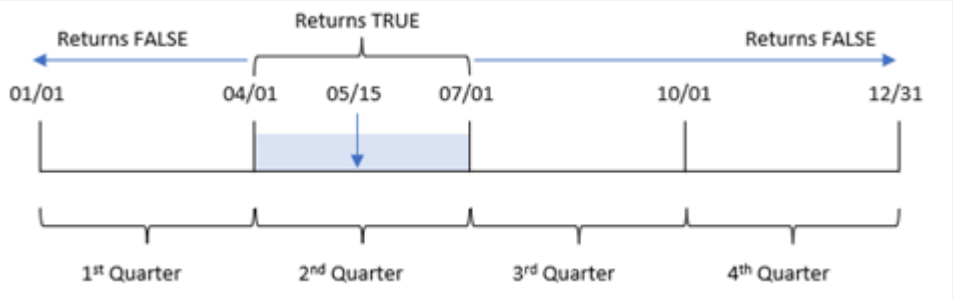
Tabela wynikowa

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

date	in_quarter
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

Pole `in_quarter` jest tworzone w poprzedniej instrukcji load przy użyciu funkcji `inquarter()`. Pierwszy argument określa, które pole jest oceniane. Drugi argument to wpisana bezpośrednio w kod data 15 maja określająca, który kwartał ma zostać wybrany jako porównawczy. Na końcu przekazano wartość `0` jako argument `period_no`, aby funkcja `inquarter()` nie porównywała kwartałów poprzedzających segmentowany kwartał lub następujących po nim.

Diagram funkcji `inquarter()` z datą 15 maja jako datą bazową



Każda transakcja zawarta w okresie od 1 kwietnia do końca 30 czerwca powoduje zwrot logicznej prawdy w wyniku.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Poprzedzające ładowanie zawiera funkcję `inquarter()` ustawioną jako pole `previous_quarter` i określa, które transakcje zostały zawarte w kwartale poprzedzającym kwartał obejmujący datę 15 maja 2022 roku.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

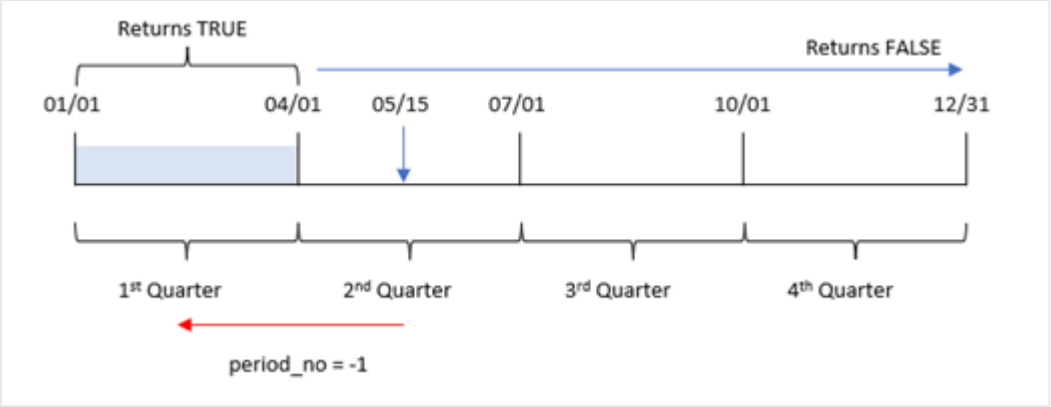
- `date`
- `previous_qtr`

Tabela wynikowa

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

Przekazanie funkcji `inquarter()` wartości `-1` jako argumentu `period_no` powoduje przesunięcie granic kwartału porównawczego wstecz o pełny kwartał. 15 maja wypada w drugim kwartale roku, a więc początkowo segment obejmuje daty od 1 kwietnia do 30 czerwca. Argument `period_no` przesuwa ten segment o trzy miesiące wstecz, powodując zmianę dat granicznych na 1 stycznia i 30 marca.

Diagram funkcji `inquarter()` z datą 15 maja jako datą bazową



W związku z tym każda transakcja zawarta między 1 stycznia a 30 marca spowoduje zwrócenie w wyniku logicznej prawdy.

Przykład 3 – first_month_of_year

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Poprzedzające ładowanie zawiera funkcję `inquarter()` ustawioną jako pole `in_quarter` i określa, które transakcje zostały zawarte w tym samym kwartale, do którego należy data 15 maja 2022 roku.

W tym przykładzie przyjęto jednak, że marzec ma być pierwszym miesiącem roku podatkowego.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `previous_qtr`

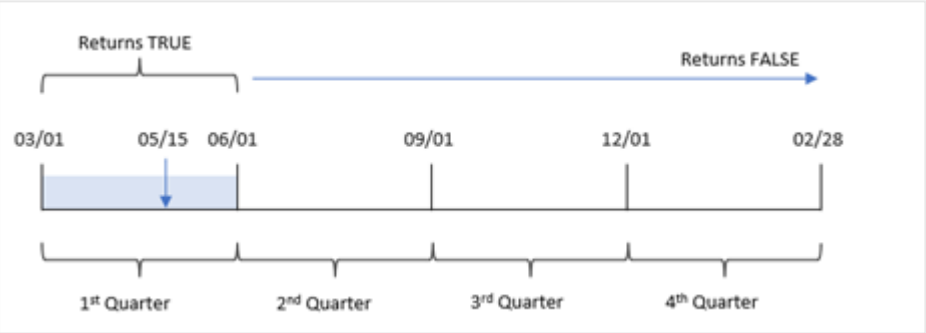
Tabela wynikowa

date	previous_qtr
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	-1
4/1/2022	-1
5/7/2022	-1
5/16/2022	-1
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0

date	previous_qtr
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Przekazanie wartości `3` jako argumentu `first_month_of_year` funkcji `inquarter()` powoduje ustawienie 1 marca jako początku roku, a następnie jego podzielenie na cztery kwartały. W efekcie kwartały obejmują następujące okresy: marzec-maj, czerwiec-sierpień, wrzesień-listopad, grudzień-luty. Wartość 15 maja argumentu `base_date` ustawia kwartał marzec-maj jako kwartał porównawczy dla funkcji.

Diagram funkcji `inquarter()` z marcem jako pierwszym miesiącem roku



W związku z tym każda transakcja zawarta między 1 marca a 31 maja spowoduje zwrócenie w wyniku logicznej prawdy.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Poprzedzające ładowanie zawiera funkcję `inquarter()` ustawioną jako pole `in_quarter` i określa, które transakcje zostały zawarte w tym samym kwartale, do którego należy data 15 maja 2022 roku.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

- `date`

Aby obliczyć, czy transakcje zostały zawarte w tym samym kwartale, w którym wypada 15 maja, utwórz następującą miarę:

```
=inquarter(date,'05/15/2022', 0)
```

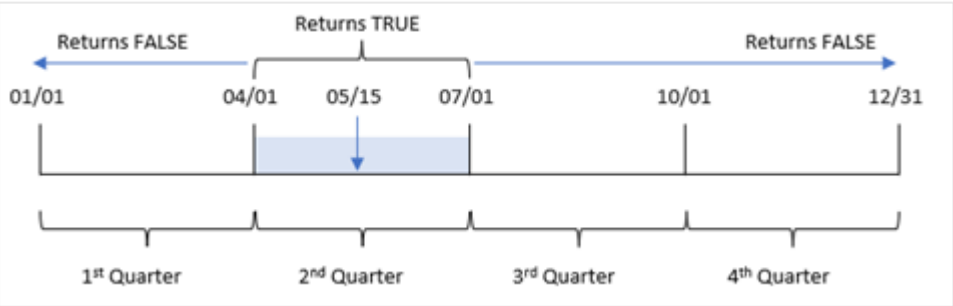
Tabela wynikowa

date	in_quarter
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0

date	in_quarter
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

Miara `in_quarter` jest tworzona na wykresie przy użyciu funkcji `inquarter()`. Pierwszy argument określa, które pole jest oceniane. Drugi argument to wpisana bezpośrednio w kod data 15 maja określająca, który kwartał ma zostać wybrany jako porównawczy. Na końcu przekazano wartość `0` jako argument `period_no`, aby funkcja `inquarter()` nie porównywała kwartałów poprzedzających segmentowany kwartał lub następujących po nim.

Diagram funkcji `inquarter()` z datą 15 maja jako datą bazową



Każda transakcja zawarta w okresie od 1 kwietnia do końca 30 czerwca powoduje zwrot logicznej prawdy w wyniku.

Przykład 5 – scenariusz

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Products`.
- Tabela zawiera następujące pola:
    - identyfikator produktu,
    - typ produktu,
    - data produkcji,
    - cena.

Stwierdzono, że z powodu błędu sprzętowego produkty wytworzone w kwartale obejmującym datę 15 maja 2022 r. są wadliwe. Użytkownik końcowy chciałby, aby wykres wyświetlał według nazwy kwartału informację na temat tego, które z wyprodukowanych produktów były wadliwe, a które bez wad, oraz koszt produktów wytworzonych w danym kwartale.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

```
=quartername(manufacture_date)
```

Utwórz następujące miary:

- `=if(only(InQuarter(manufacture_date,makedate(2022,05,15),0)),'Defective','Faultless')` - określa, które produkty są wadliwe, a które bez wad, za pomocą funkcji `inquarter()`.
- `=sum(cost_price)` - pokazuje sumę kosztów wszystkich produktów.

Wykonaj następujące czynności:

1. Ustaw **Formatowanie liczb** miary na **Waluta**.
2. W sekcji **Wygląd** wyłącz **Sumy**.

Tabela wynikowa

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

Funkcja `inquarter()` zwraca wartość logiczną podczas oceny dat wytworzenia każdego z produktów. Dla każdego produktu wyprodukowanego w kwartale obejmującym 15 maja funkcja `inquarter()` zwraca wartość logiczną TRUE i oznacza go jako wadliwy. Dla każdego produktu zwracającego wartość FALSE, a zatem niewyprodukowanego w danym kwartale, oznacza go jako pozbawiony wad (Faultless).

inquartertodate

Ta funkcja zwraca wartość True, jeśli wartość **timestamp** należy do części kwartału zawierającego wartość **base_date**, z dokładnością do ostatniej milisekundy wartości **base_ date** włącznie.

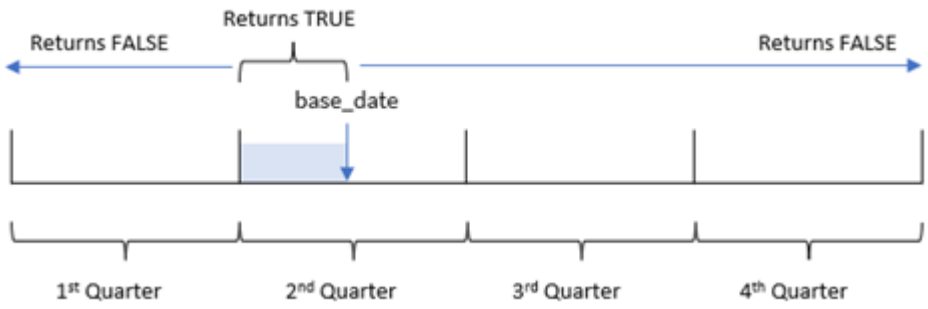
Składnia:

```
InQuarterToDate (timestamp, base_date, period_no [, first_month_of_year])
```

Typ zwracanych danych: Wartość logiczna

> *W Qlik Sense wartość logiczna Prawda jest reprezentowana przez -1, a wartość Fałsz jest reprezentowana przez 0.*

Diagram funkcji inquartertodate



Funkcja `inquartertodate()` dzieli rok na cztery równe kwartały między 1 stycznia a 31 grudnia (lub zdefiniowanym przez użytkownika początkiem roku a odpowiadającą mu datą zakończenia). Używając `base_date`, funkcja dokona segmentacji określonego kwartału, przy czym `base_date` identyfikuje zarówno kwartał, jak i maksymalną dozwoloną datę dla tego segmentu kwartału. Na koniec funkcja zwraca wynik logiczny podczas porównywania określonych wartości dat z tym segmentem.

Argumenty

Argument	Opis
timestamp	Data, która ma być porównana z wartością **base_date**.
base_date	Data używana do oceny kwartału.
period_no	Kwartał może zostać przesunięty o wartość **period_no**. **period_no** jest liczbą całkowitą, gdzie 0 oznacza kwartał zawierający wartość **base_date**. Wartości ujemne parametru **period_no** oznaczają kwartały poprzednie, a wartości dodatnie – kwartały następne.
first_month_of_year	Jeśli użytkownik zamierza korzystać z lat (obrotowych), które nie zaczynają się w styczniu, powinien wskazać wartość od 2 do 12 jako parametr **first_month_of_year**.

Kiedy używać

Funkcja `inquartertodate()` zwraca wynik logiczny. Zazwyczaj ten typ funkcji będzie używany jako warunek w wyrażeniu `if`. Można użyć funkcji `inquartertodate()`, aby zwróciła agregację lub obliczenia w zależności od tego, czy oceniana data nastąpiła w kwartale do danej daty włącznie.

Na przykład funkcja `inquartertodate()` może służyć do identyfikacji całego sprzętu wyprodukowanego w kwartale do określonej daty.

Przykłady funkcji

Przykład	Wynik
`inquartertodate ('01/25/2013', '03/25/2013', 0)`	Zwraca `TRUE`, ponieważ wartość `timestamp`, 01/25/2013, przypada w okresie trzech miesięcy od 01/01/2013 do 03/25/2013, w którym przypada wartość `base_date`, 03/25/2013.

Przykład	Wynik
`inquartertodate ('04/26/2013', '03/25/2013', 0)`	Zwraca `FALSE`, ponieważ 04/26/2013 przypada poza tym samym okresem co poprzedni przykład.
`inquartertodate ('02/25/2013', '06/09/2013', -1)`	Zwraca `TRUE`, ponieważ wartość`period_no`, -1, cofa okres wyszukiwania o jeden okres trzymiesięczny (jeden kwartał roku). W związku z tym okres wyszukiwania jest od 01/01/2013 do 03/09/2013.
`inquartertodate ('03/25/2006', '04/15/2006', 0, 2)`	Zwraca `TRUE`, ponieważ wartość `first_month_of_year` jest ustawiona na 2, co sprawia, że okres wyszukiwania jest od 02/01/2006 do 04/15/2006 zamiast od 04/01/2006 do 04/15/2006.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „`Transactions`”.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie pola, `in_quarter_to_date`, które określa, jakie transakcje miały miejsce w kwartale do 15 maja 2022 r.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `in_quarter_to_date`

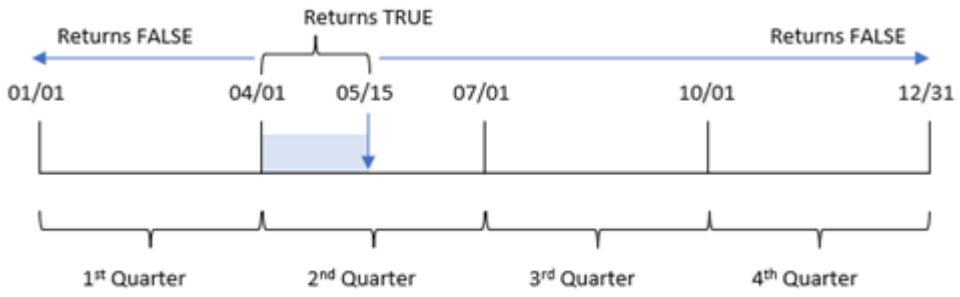
Tabela wynikowa

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

date	in_quarter_to_date
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

Pole `in_quarter_to_date` jest tworzone w poprzedniej instrukcji load przy użyciu funkcji `inquartertodate()`. Pierwszy podany argument określa, które pole jest oceniane. Drugi argument to ustalona data 15 maja – jest to argument `base_date`, który wskazuje kwartał do posegmentowania i definiuje granicę końcową tego segmentu. Argument `period_no` o wartości 0 jest ostatnim argumentem, co oznacza, że funkcja nie porównuje kwartałów poprzedzających segmentowany kwartał ani następujących po nim.

Diagram funkcji inquartertodate, bez dodatkowych argumentów



Każda transakcja, która ma miejsce w okresie od 1 kwietnia do 15 maja, zwraca wynik logiczny `TRUE`. Daty transakcji 16 maja i późniejsze zwrócą `FALSE`, podobnie jak wszelkie transakcje sprzed 1 kwietnia.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola, `previous_qtr_to_date`, które określa, jakie transakcje miały miejsce w pełnym kwartale przed segmentem kwartału kończącego się 15 maja 2022 r.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

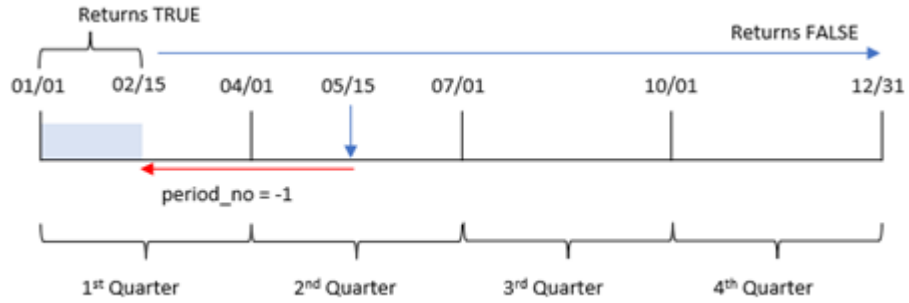
- `date`
- `previous_qtr_to_date`

Tabela wynikowa

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

Wartość `period_no` równa -1 wskazuje, że funkcja `inquartertodate ()` porównuje wejściowy segment kwartału z poprzednim kwartałem. 15 maja przypada na drugi kwartał roku, więc segment początkowo równa się od 1 kwietnia do 15 maja. Wartość `period_no` następnie przesuwa ten segment o trzy miesiące wcześniej, powodując, że datami granicznymi są 1 stycznia i 15 lutego.

Diagram funkcji inquartertodate, przykład period_no



W związku z tym każda transakcja, która ma miejsce między 1 stycznia a 15 lutego, zwróci wynik logicznyTRUE.

Przykład 3 – first_month_of_year

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola, `in_quarter_to_date`, które określa, jakie transakcje miały miejsce w tym samym kwartale do 15 maja 2022 r.

W tym przykładzie jako pierwszy miesiąc roku obrachunkowego ustawiliśmy marzec.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `in_quarter_to_date`

Tabela wynikowa

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

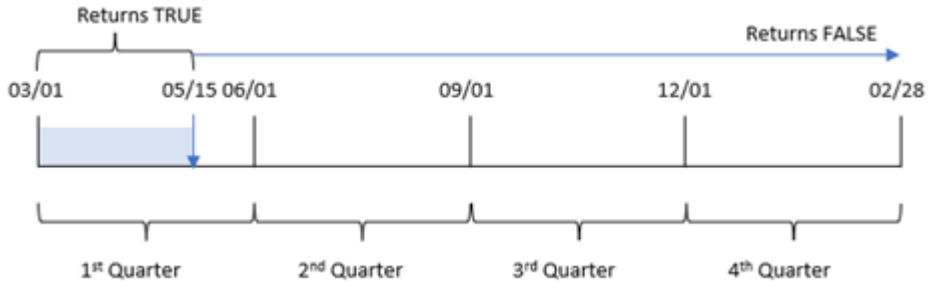
date	in_quarter_to_date
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Używając 3 jako argumentu `first_month_of_year` w funkcji `inquartertodate()`, funkcja rozpoczyna rok 1 marca, a następnie dzieli rok na kwartały. Dlatego segmenty kwartałów to:

- Od marca do maja
- Od czerwca do sierpnia
- Od września do listopada
- Od grudnia do lutego

Wartość `base_date` z 15 maja dzieli kwartał od marca do maja, ustawiając granicę końcową na 15 maja.

Diagram funkcji inquartertodate, przykład first_month_of_year



W związku z tym każda transakcja, która ma miejsce między 1 marca a 15 maja, zwróci wynik logiczny `TRUE`, podczas gdy transakcje z datami poza tymi granicami zwrócą wartość `FALSE`.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie. Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenie określające, które transakcje miały miejsce w tym samym kwartale co 15 maja, jest tworzone jako miara w obiekcie wykresu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:`date`.

Utwórz następującą miarę:

```
=inquartertodate(date,'05/15/2022', 0)
```

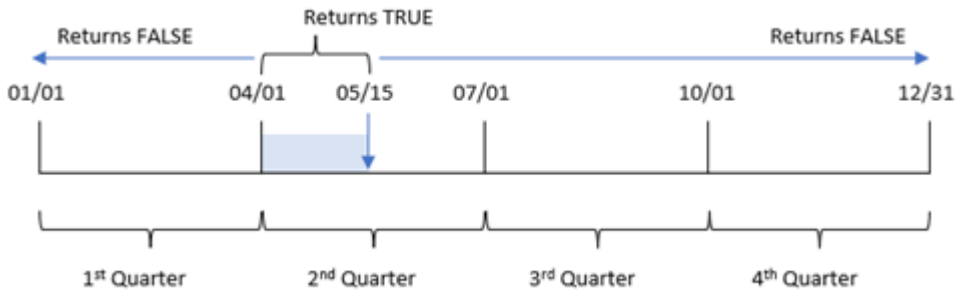
Tabela wynikowa

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

date	=inquartertodate(date,'05/15/2022', 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

Miara `in_quarter_to_date` jest tworzona w obiekcie wykresu przy użyciu funkcji `inquartertodate()`. Pierwszym argumentem jest oceniane pole daty. Drugi argument to ustalona data 15 maja – jest to argument `base_date`, który wskazuje kwartał do posegmentowania i definiuje granicę końcową tego segmentu. Argument `period_no` o wartości 0 jest ostatnim argumentem, co oznacza, że funkcja nie porównuje kwartałów poprzedzających segmentowany kwartał ani następujących po nim.

Diagram funkcji inquartertodate, przykład obiektu wykresu



Każda transakcja, która ma miejsce w okresie od 1 kwietnia do 15 maja, zwraca wynik logiczny `TRUE`. Transakcje z 16 maja i późniejsze zwrócą `FALSE`, podobnie jak wszelkie transakcje sprzed 1 kwietnia.

Przykład 5 – scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Products`.
- Informacje dotyczące identyfikatora produktu, daty produkcji i kosztu.

15 maja 2022 r. w procesie produkcyjnym zidentyfikowano i naprawiono błąd sprzętowy. Produkty wyprodukowane w tym kwartale do tego dnia będą wadliwe. Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał według nazw kwartałów status produktów: „defective” (wadliwe) lub „faultless” (bez wad) oraz koszt produktów wytworzonych w danym kwartale do daty bieżącej.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę. Utwórz wymiar, aby pokazać nazwy kwartałów:
   `=quartername(manufacture_date)`
2. Następnie utwórz wymiar, aby określić, które produkty są wadliwe, a które bez wad:
   `=if(inquartertodate(manufacture_date,makedate(2022,05,15),0),'Defective','Faultless')`
3. Utwórz miarę, aby zsumować `cost_price` produktów:
   `=sum(cost_price)`
4. Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

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

Funkcja `inquartertodate()` zwraca wartość logiczną podczas oceny dat wytworzenia każdego z produktów. Dla tych, które zwracają wartość logiczną `TRUE`, oznacza produkty jako `'Defective'`. W przypadku każdego produktu zwracającego wartość `FALSE`, a zatem niewyprodukowanego w kwartale do 15 maja włącznie, oznacza ona produkty jako `'Faultless'`.

inweek

Ta funkcja zwraca wartość True, jeśli wartość **timestamp** należy do tygodnia zawierającego wartość **base_date**.

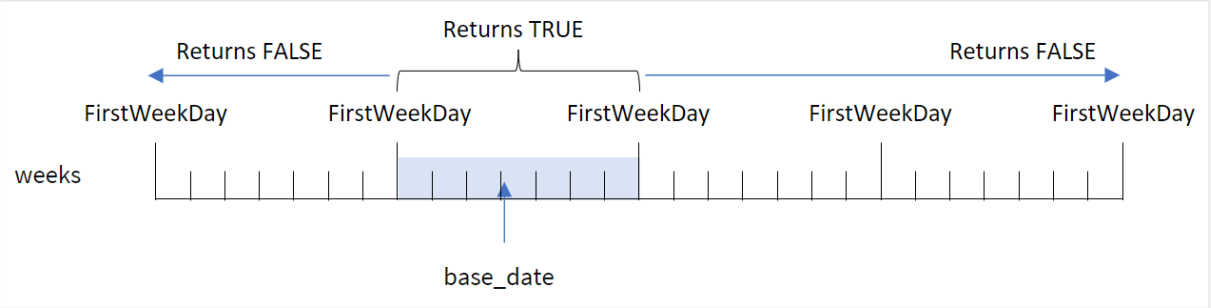
Składnia:

```
InWeek (timestamp, base_date, period_no[, first_week_day])
```

Typ zwracanych danych: Wartość logiczna

W Qlik Sense wartość logiczna Prawda jest reprezentowana przez -1, a wartość Fałsz jest reprezentowana przez 0.

Diagram zakresu funkcji `inweek()`



Funkcja `inweek()` używa argumentu `base_date` do identyfikacji, w którym siedmiodniowym okresie wypada określona data. Dzień początkowy tygodnia jest określany na podstawie zmiennej systemowej `FirstWeekDay`. Możesz jednak również zmienić pierwszy dzień tygodnia, używając argumentu `first_week_day` funkcji `inweek()`.

Kiedy zostanie zdefiniowany wybrany tydzień, funkcja zwraca w wyniku wartość logiczną na podstawie porównania określonych wartości dat z danym segmentem tygodnia.

Kiedy używać

Funkcja `inweek()` zwraca wynik logiczny. Zazwyczaj ten typ funkcji będzie używany jako warunek w `if expression`. Funkcja `inweek()` zwraca agregację lub obliczenia zależne od tego, czy dana data wypada w tygodniu obejmującym wybraną datę argumentu `base_date`.

Na przykład funkcja `inweek()` może służyć do identyfikacji całego sprzętu wyprodukowanego w danym tygodniu.

Argumenty

Argument	Opis
timestamp	Data, która ma być porównana z wartością **base_date**.
base_date	Data używana do oceny tygodnia.
period_no	Tydzień może zostać przesunięty o wartość **period_no**. **period_no** jest liczbą całkowitą, gdzie 0 oznacza tydzień zawierający wartość **base_date**. Wartości ujemne parametru **period_no** oznaczają tygodnie poprzednie, a wartości dodatnie – tygodnie następne.
first_week_day	Domyślnie pierwszym dniem tygodnia jest niedziela (zgodnie ze zmienną systemową FirstWeekDay), począwszy od północy między sobotą a niedzielą. Parametr **first_week_day** zastępuje zmienną **FirstWeekDay**. Aby wskazać, że tydzień zaczyna się innego dnia, należy określić flagę w zakresie od 0 do 6.

first_week_day values

Dzień	Wartość
poniedziałek	0
wtorek	1
środa	2
czwartek	3
piątek	4
sobota	5
niedziela	6

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
`inweek ('01/12/2006', '01/14/2006', 0)`	Zwraca TRUE
`inweek ('01/12/2006', '01/20/2006', 0 )`	Zwraca FALSE
`inweek ('01/12/2006', '01/14/2006', -1 )`	Zwraca FALSE
`inweek ('01/07/2006', '01/14/2006', -1)`	Zwraca TRUE
`inweek ('01/12/2006', '01/09/2006', 0, 3)`	Zwraca FALSE, ponieważ argument `first_week_day` ma wartość 3 (czwartek), co sprawia, że 12.01.2006 r. jest pierwszym dniem tygodnia następującego po tygodniu obejmującym datę 09.01.2006 r.

W pracy z tą funkcją mogą Ci pomóc poniższe tematy:

Tematy pokrewne

Temat	Flaga domyślna / Wartość	Opis
FirstWeekDay (page 215)	6 / niedziela	Definiuje dzień rozpoczęcia każdego tygodnia.

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za styczeń 2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Zmienna systemowa `FirstWeekDay`, która jest ustawiona na `6` (niedziela).
- Ładunek poprzedzający, który zawiera następujące elementy:
    - Funkcja `inweek()`, ustawiona jako pole `in_week` określające, które transakcje zostały zawarte w tygodniu obejmującym datę 14 lutego 2022 r.
    - Funkcja `weekday()`, ustawiona jako pole `week_day` pokazujące, który dzień tygodnia odpowiada każdej dacie.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- date
- week_day
- in_week

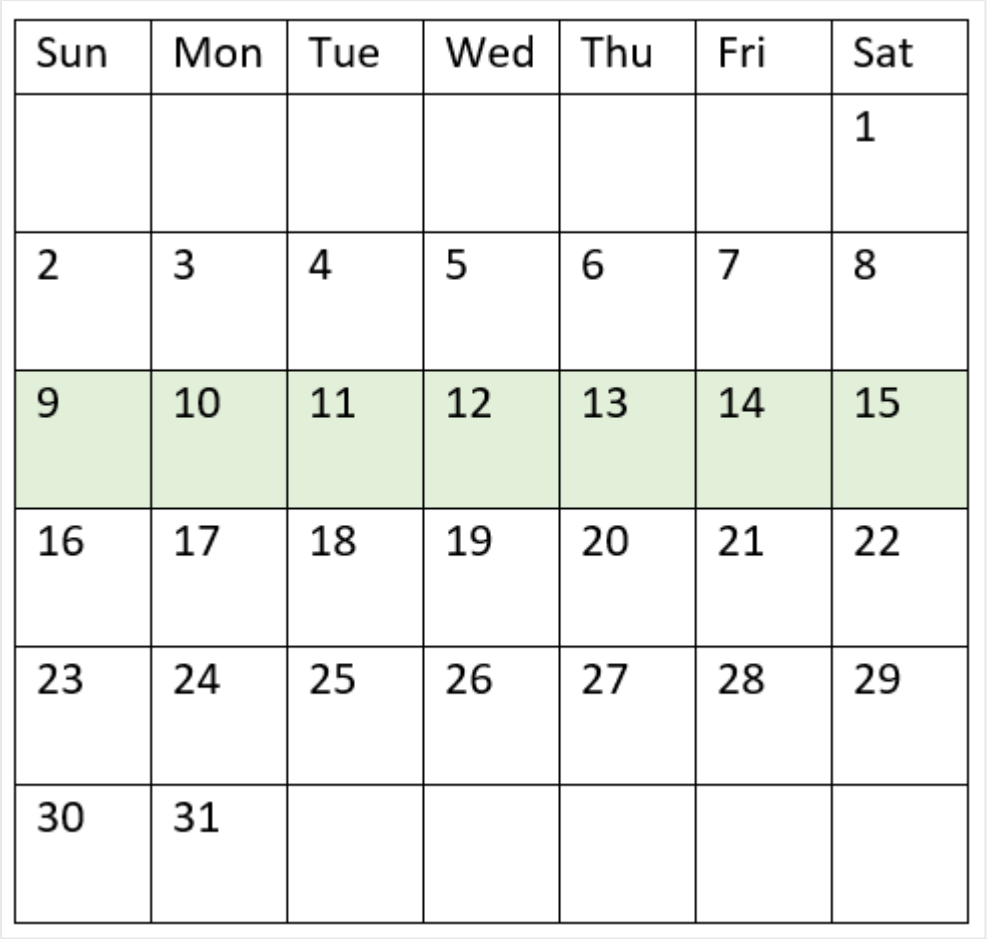
Tabela wynikowa

date	week_day	in_week
01/02/2022	Nie	0
01/05/2022	Śro	0
01/06/2022	Thu	0
01/08/2022	Sat	0
01/09/2022	Nie	-1
01/10/2022	Pon	-1
01/11/2022	Wto	-1
01/12/2022	Śro	-1
01/13/2022	Thu	-1
01/14/2022	Fri	-1
01/15/2022	Sat	-1
01/16/2022	Nie	0
01/17/2022	Pon	0
01/18/2022	Wto	0
01/26/2022	Śro	0
01/27/2022	Thu	0
01/28/2022	Fri	0
01/29/2022	Sat	0
01/30/2022	Nie	0
01/31/2022	Pon	0

Pole `in_week` jest tworzone w poprzedniej instrukcji load przy użyciu funkcji `inweek()`. Pierwszy argument określa, które pole jest oceniane. Drugi argument to wpisana bezpośrednio w kod data 14 stycznia, która jest wartością argumentu `base_date`. Argument `base_date` w połączeniu ze zmienną systemową `FirstWeekDay` identyfikuje tydzień porównawczy. Argument `period_no` o wartości `0` – oznaczający, że funkcja nie porównuje tygodni poprzedzających segmentowany tydzień ani następujących po nim – jest ostatni.

Zmienna systemowa `FirstWeekDay` określa, że tygodnie zaczynają się w niedzielę i kończą w sobotę. W związku z tym styczeń zostałby podzielony na tygodnie zgodnie z poniższym wykresem, a daty między 9 a 15 stycznia stanowiłyby prawidłowy okres dla obliczeń funkcji `inweek()`:

Diagram kalendarza z zaznaczonym zakresem funkcji `inweek()`



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					

Wszelkie transakcje zawarte między 9 a 15 stycznia powodują zwrócenie logicznej wartości `TRUE`.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych zawierający zestaw transakcji za rok 2022 zostaje załadowany do tabeli o nazwie `Transactions`.
- Zmienna systemowa `FirstWeekDay`, która jest ustawiona na `6` (niedziela).
- Ładunek poprzedzający, który zawiera następujące elementy:
  - Funkcja `inweek ()`, ustawiona jako pole `prev_week` określające, które transakcje zostały zawarte cały tydzień przed tygodniem obejmującym datę 14 lutego 2022 r.

- Funkcja `weekday()`, ustawiona jako pole `week_day` pokazujące, który dzień tygodnia odpowiada każdej dacie.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

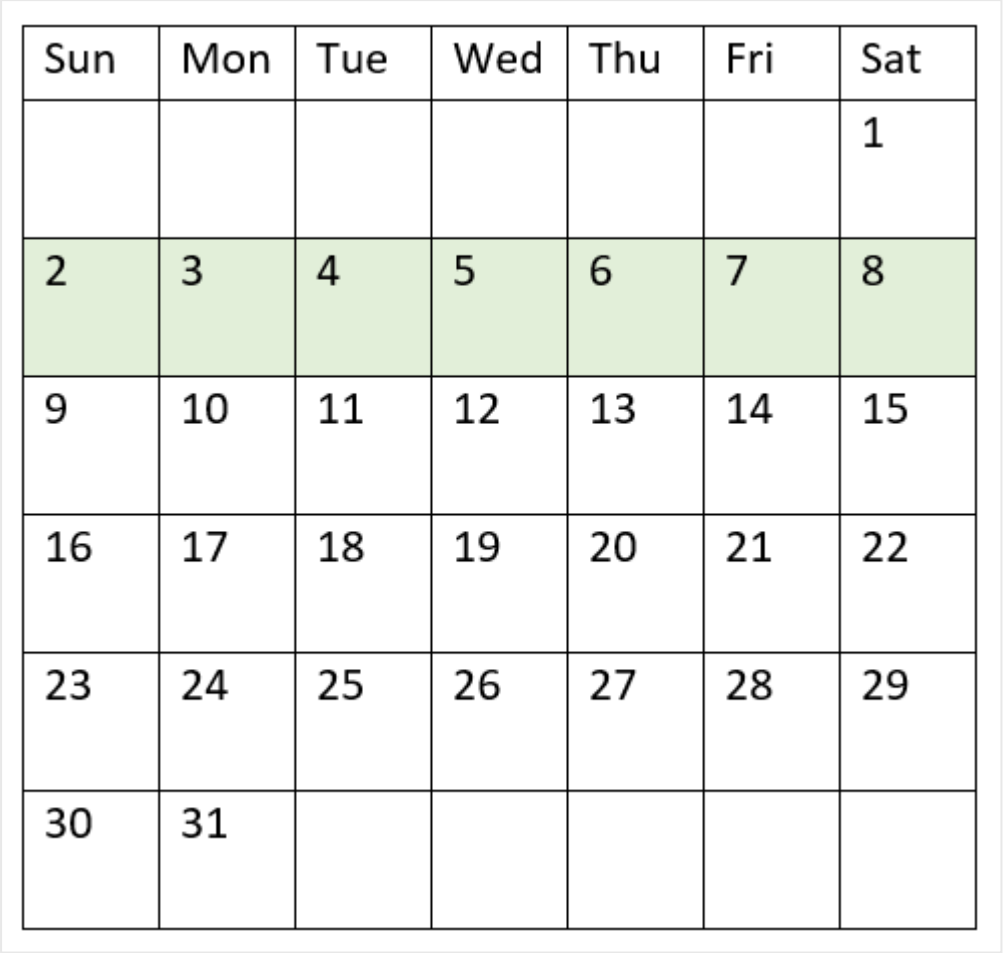
- `date`
- `week_day`
- `prev_week`

Tabela wynikowa

date	week_day	prev_week
01/02/2022	Nie	-1
01/05/2022	Śro	-1
01/06/2022	Thu	-1
01/08/2022	Sat	-1
01/09/2022	Nie	0
01/10/2022	Pon	0
01/11/2022	Wto	0
01/12/2022	Śro	0
01/13/2022	Thu	0
01/14/2022	Fri	0
01/15/2022	Sat	0
01/16/2022	Nie	0
01/17/2022	Pon	0
01/18/2022	Wto	0
01/26/2022	Śro	0
01/27/2022	Thu	0
01/28/2022	Fri	0
01/29/2022	Sat	0
01/30/2022	Nie	0
01/31/2022	Pon	0

Przekazanie funkcji `inweek()` wartości `-1` jako argumentu `period_no` powoduje przesunięcie granic tygodnia porównawczego wstecz o pełne siedem dni. W przypadku argumentu `period_no` o wartości `0` tydzień obejmowałby dni od 9 do 15 stycznia. Jednak w tym przykładzie argument `period_no` o wartości `-1` przesuwa początek i koniec tego segmentu o tydzień wstecz. Zakres dat zmienia się na okres od 2 do 8 stycznia.

Diagram kalendarza z zaznaczonym zakresem funkcji `inweek()`



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					

W związku z tym każda transakcja, która ma miejsce między 2 a 8 stycznia spowoduje zwrócenie logicznej prawdy.

Przykład 3 – first_week_day

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych zawierający zestaw transakcji za rok 2022 zostaje załadowany do tabeli o nazwie `Transactions`.
- Zmienna systemowa `FirstWeekDay`, która jest ustawiona na `6` (niedziela).
- Ładunek poprzedzający, który zawiera następujące elementy:
    - Funkcja `inweek()`, ustawiona jako pole `in_week` określające, które transakcje zostały zawarte w tygodniu obejmującym datę 14 lutego 2022 r.

- Funkcja `weekday()`, ustawiona jako pole `week_day` pokazujące, który dzień tygodnia odpowiada każdej dacie.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `week_day`
- `in_week`

Tabela wynikowa

date	week_day	in_week
01/02/2022	Nie	0
01/05/2022	Śro	0
01/06/2022	Thu	0
01/08/2022	Sat	0
01/09/2022	Nie	0
01/10/2022	Pon	-1
01/11/2022	Wto	-1
01/12/2022	Śro	-1
01/13/2022	Thu	-1
01/14/2022	Fri	-1
01/15/2022	Sat	-1
01/16/2022	Nie	-1
01/17/2022	Pon	0
01/18/2022	Wto	0
01/26/2022	Śro	0
01/27/2022	Thu	0
01/28/2022	Fri	0
01/29/2022	Sat	0
01/30/2022	Nie	0
01/31/2022	Pon	0

Przekazanie wartości `0` jako argumentu `first_week_day` funkcji `inweek()` powoduje zastąpienie zmiennej systemowej `FirstWeekDay` i ustawienie poniedziałku jako pierwszego dnia tygodnia.

Diagram kalendarza z zaznaczonym zakresem funkcji inweek()

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					

W związku z tym każda transakcja zawarta między 10 a 16 stycznia spowoduje zwrócenie w wyniku logicznej prawdy.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie zestaw danych pozostaje bez zmian i jest ładowany do aplikacji. Utwórz miarę w tabeli wyników znajdującą transakcje, które zostały zawarte w tygodniu obejmującym datę 14 stycznia 2022 r.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

- `date`

Utwórz następujące miary:

- `=inweek (date,'01/14/2022',0)`, aby obliczyć, czy transakcje zostały zawarte w tym samym tygodniu, w którym wypada 14 stycznia.
- `=weekday(date)`, aby pokazać, który dzień tygodnia odpowiada każdej dacie.

Tabela wynikowa

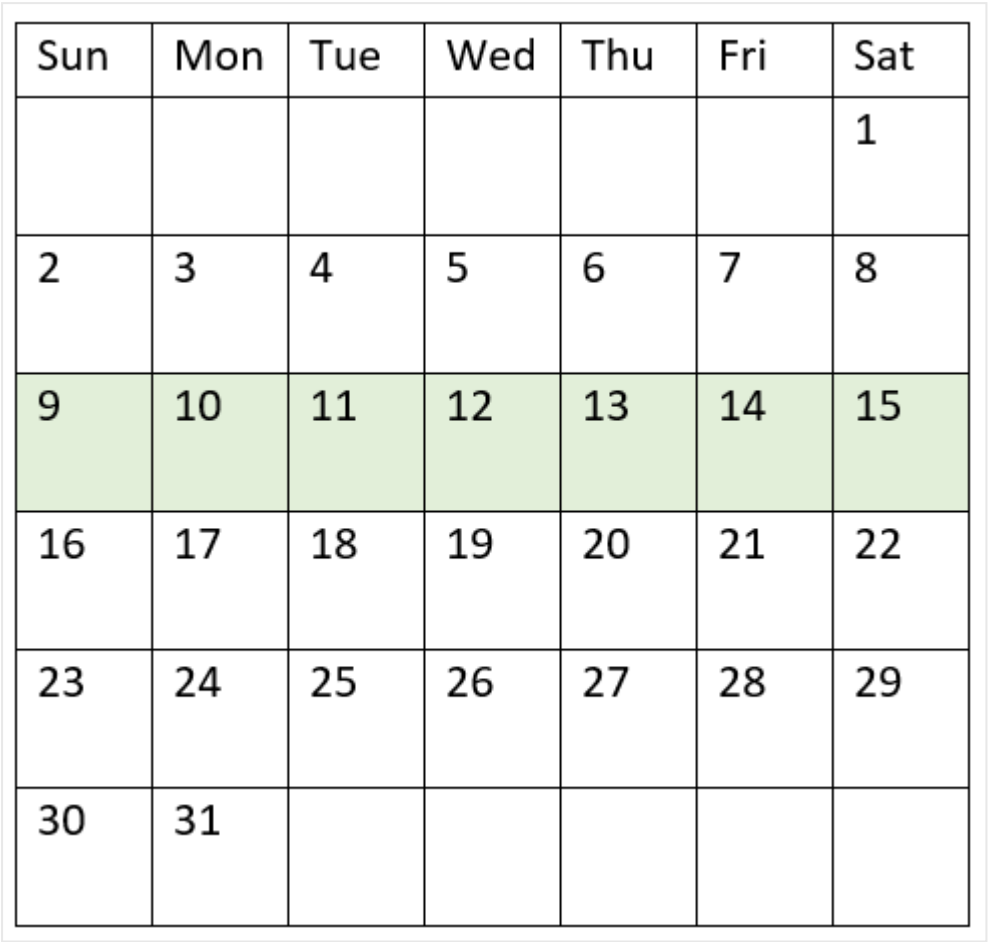
date	week_day	=inweek (date,'01/14/2022',0)
01/02/2022	Nie	0
01/05/2022	Śro	0
01/06/2022	Thu	0
01/08/2022	Sat	0
01/09/2022	Nie	-1
01/10/2022	Pon	-1

date	week_day	=inweek (date,'01/14/2022',0)
01/11/2022	Wto	-1
01/12/2022	Śro	-1
01/13/2022	Thu	-1
01/14/2022	Fri	-1
01/15/2022	Sat	-1
01/16/2022	Nie	0
01/17/2022	Pon	0
01/18/2022	Wto	0
01/26/2022	Śro	0
01/27/2022	Thu	0
01/28/2022	Fri	0
01/29/2022	Sat	0
01/30/2022	Nie	0
01/31/2022	Pon	0

Miara `in_week` jest tworzona na wykresie przy użyciu funkcji `inweek()`. Pierwszy argument określa, które pole jest oceniane. Drugi argument to wpisana bezpośrednio w kod data 14 stycznia, która jest wartością argumentu `base_date`. Argument `base_date` w połączeniu ze zmienną systemową `FirstWeekDay` identyfikuje tydzień porównawczy. Ostatni argument to `period_no` o wartości `0`.

Zmienna systemowa `FirstWeekDay` określa, że tygodnie zaczynają się w niedzielę i kończą w sobotę. W związku z tym styczeń zostałby podzielony na tygodnie zgodnie z poniższym wykresem, a daty między 9 a 15 stycznia stanowiłyby prawidłowy okres dla obliczeń funkcji `inweek()`:

Diagram kalendarza z zaznaczonym zakresem funkcji inweek()



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					

Wszelkie transakcje zawarte między 9 a 15 stycznia powodują zwrócenie logicznej wartości TRUE.

Przykład 5 – scenariusz

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Products`.
- Tabela zawiera następujące pola:
  - identyfikator produktu,
  - typ produktu,
  - data produkcji,
  - cena.

Stwierdzono, że z powodu błędu sprzętowego produkty wytworzone w tygodniu z 12 stycznia były wadliwe. Użytkownik końcowy chciałby, aby wykres wyświetlał według tygodni informację, czy produkty były wadliwe, czy bez wad oraz koszt produktów wytworzonych w danym tygodniu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

- `=weekname(manufacture_date)`

Utwórz następujące miary:

- `=if(only(inweek(manufacture_date,makedate(2022,01,12),0)),'Defective','Faultless')` - określa, które produkty są wadliwe, a które bez wad, za pomocą funkcji `inweek()`.
- `=sum(cost_price)` - pokazuje sumę kosztów wszystkich produktów.

Wykonaj następujące czynności:

1. Ustaw **Formatowanie liczb** miary na **Waluta**.
2. W sekcji **Wygląd** wyłącz **Sumy**.

Tabela wynikowa

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

Funkcja `inweek()` zwraca wartość logiczną podczas oceny dat wytworzenia każdego z produktów. W przypadku każdego produktu wyprodukowanego w tygodniu obejmującym 12 stycznia funkcja `inweek()` zwraca wartość logiczną PRAWDA i oznacza produkty jako „Defective” (Wadliwe). Każdy produkt, dla którego jest zwracana wartość FALSE, a zatem niewyprodukowanego w tym tygodniu, oznacza jako pozbawiony wad.

inweektodate

Ta funkcja zwraca wartość True, jeśli wartość **timestamp** należy do części tygodnia zawierającego wartość **base_date**, z dokładnością do ostatniej milisekundy wartości **base_ date** włącznie.

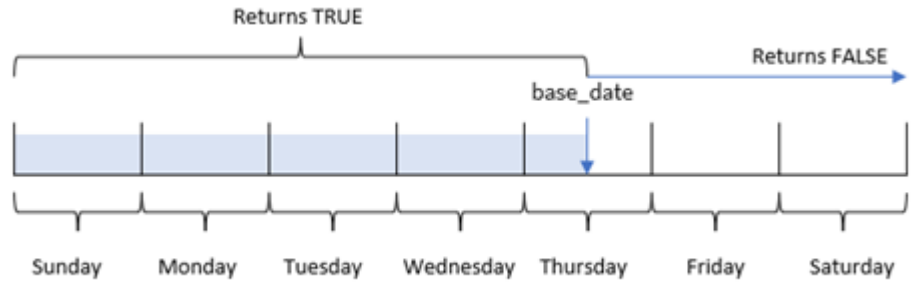
Składnia:

```
InWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Typ zwracanych danych: Wartość logiczna

> *W Qlik Sense wartość logiczna Prawda jest reprezentowana przez -1, a wartość Fałsz jest reprezentowana przez 0.*

Diagram funkcji inweektodate



Funkcja `inweektodate()` używa parametru `base_date` do identyfikacji maksymalnej daty granicznej segmentu tygodnia, a także odpowiadającej mu daty początku tygodnia, która jest oparta na zmiennej systemowej `FirstWeekDay` (lub parametrze `first_week_day` zdefiniowanym przez użytkownika). Kiedy segment tygodnia zostanie zdefiniowany, funkcja zwraca wyniki logiczne podczas porównywania określonych wartości dat z tym segmentem.

Kiedy używać

Funkcja `inweektodate()` zwraca wynik logiczny. Zazwyczaj ten typ funkcji będzie używany jako warunek w wyrażeniu `if`. Spowoduje to zwrócenie agregacji lub obliczenia w zależności od tego, czy oceniana data nastąpiła w danym tygodniu do określonej daty włącznie.

Na przykład funkcja `inweektodate()` może służyć do obliczania wszystkich sprzedaży dokonanych w określonym tygodniu do określonej daty.

Argumenty

Argument	Opis
timestamp	Data, która ma być porównana z wartością **base_date**.
base_date	Data używana do oceny tygodnia.
period_no	Tydzień może zostać przesunięty o wartość **period_no**. **period_no** jest liczbą całkowitą, gdzie 0 oznacza tydzień zawierający wartość **base_date**. Wartości ujemne parametru **period_no** oznaczają tygodnie poprzednie, a wartości dodatnie – tygodnie następne.
first_week_day	Domyślnie pierwszym dniem tygodnia jest niedziela (zgodnie ze zmienną systemową FirstWeekDay), począwszy od północy między sobotą a niedzielą. Parametr **first_week_day** zastępuje zmienną **FirstWeekDay**. Aby wskazać, że tydzień zaczyna się innego dnia, należy określić flagę w zakresie od 0 do 6. W przypadku tygodnia rozpoczynającego się w poniedziałek i kończącego w niedzielę użyj flagi 0 dla poniedziałku, 1 dla wtorku, 2 dla środy, 3 dla czwartku, 4 dla piątku, 5 dla soboty i 6 dla niedzieli.

Przykłady funkcji

Przykład	Interakcja
`inweektodate ('01/12/2006', '01/12/2006', 0)`	Zwraca wartość `TRUE`.
`inweektodate ('01/12/2006', '01/11/2006', 0)`	Zwraca wartość `FALSE`.
`inweektodate ('01/12/2006', '01/18/2006', -1)`	Zwraca wartość `FALSE`. Ponieważ `period_no` określono jako -1, dlatego data obowiązywania, względem której mierzy się `timestamp`, jest równa 01/11/2006.

Przykład	Interakcja
`inweektodate ('01/11/2006', '01/12/2006', 0, 3 )`	Zwraca `FALSE`, ponieważ wartość `first_week_day` określono jako `3` (czwartek), przez co 01/12/2006 staje się pierwszym dniem tygodnia następującego po tygodniu zawierającym 01/12/2006.

W pracy z tą funkcją mogą Ci pomóc poniższe tematy:

Tematy pokrewne

Temat	Flaga domyślna / Wartość	Opis
FirstWeekDay (page 215)	6 / niedziela	Definiuje dzień rozpoczęcia każdego tygodnia.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za styczeń 2022, który jest ładowany do tabeli o nazwie „`Transactions`”.
- Pole danych podane w formacie `TimestampFormat='M/D/YYYY h:mm:ss[.fff]'`.
- Utworzenie pola, `in_week_to_date`, które określa, jakie transakcje miały miejsce w tygodniu do 14 stycznia 2022 r.
- Tworzenie dodatkowego pola o nazwie `weekday` przy użyciu funkcji `weekday()`. To nowe pole jest tworzone, aby pokazać, który dzień tygodnia odpowiada poszczególnym datom.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `week_day`
- `in_week_to_date`

Tabela wynikowa

date	week_day	in_week_to_date
2022-01-02 12:22:06	Nie	0
2022-01-05 01:02:30	Śro	0

date	week_day	in_week_to_date
2022-01-06 15:36:20	Thu	0
2022-01-08 10:58:35	Sat	0
2022-01-09 08:53:32	Nie	-1
2022-01-10 21:13:01	Pon	-1
2022-01-11 00:57:13	Wto	-1
2022-01-12 09:26:02	Śro	-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	Nie	0
2022-01-17 10:44:16	Pon	0
2022-01-18 18:48:17	Wto	0
2022-01-26 04:36:03	Śro	0
2022-01-27 08:07:49	Thu	0
2022-01-28 12:24:29	Fri	0
2022-01-30 11:56:56	Nie	0
2022-01-30 14:40:19	Nie	0
2022-01-31 05:28:21	Pon	0

Pole `in_week_to_date` jest tworzone w poprzedniej instrukcji load przy użyciu funkcji `inweektodate()`. Pierwszy podany argument określa, które pole jest oceniane. Drugi argument to ustalona data 14 stycznia – jest to argument `base_date`, który wskazuje tydzień do posegmentowania i identyfikuje granicę końcową tego segmentu. Argument `period_no` o wartości 0 jest ostatnim argumentem, co oznacza, że funkcja nie porównuje tygodni poprzedzających segmentowany tydzień ani następujących po nim.

Zmienna systemowa `FirstWeekDay` określa, że tygodnie zaczynają się w niedzielę i kończą w sobotę. W związku z tym styczeń zostałby podzielony na tygodnie zgodnie z poniższym wykresem, a daty między 9 a 14 stycznia stanowiłyby ważny okres dla obliczenia `inweekdodate()`:

Diagram kalendarza pokazujący daty transakcji, które zwrócą wynik logiczny TRUE

Sun	Mon	Tue	Wed	Thur	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Wszelkie transakcje, które występują między 9 a 14 stycznia, zwracają jako wynik wartość logiczną TRUE. Transakcje przed i po tych datach zwracają wynik logiczny FALSE.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola, `prev_week_to_date`, które określa, jakie transakcje miały miejsce w pełnym tygodniu przed segmentem tygodnia kończącego się 14 stycznia 2022 r.
- Tworzenie dodatkowego pola o nazwie `weekday` przy użyciu funkcji `weekday()`. Pokazuje, który dzień tygodnia odpowiada poszczególnym datom.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `week_day`
- `prev_week_to_date`

Tabela wynikowa

date	week_day	prev_week_to_date
2022-01-02 12:22:06	Nie	-1
2022-01-05 01:02:30	Śro	-1
2022-01-06 15:36:20	Thu	-1
2022-01-08 10:58:35	Sat	0
2022-01-09 08:53:32	Nie	0
2022-01-10 21:13:01	Pon	0
2022-01-11 00:57:13	Wto	0
2022-01-12 09:26:02	Śro	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

date	week_day	prev_week_to_date
2022-01-16 06:39:27	Nie	0
2022-01-17 10:44:16	Pon	0
2022-01-18 18:48:17	Wto	0
2022-01-26 04:36:03	Śro	0
2022-01-27 08:07:49	Thu	0
2022-01-28 12:24:29	Fri	0
2022-01-30 11:56:56	Nie	0
2022-01-30 14:40:19	Nie	0
2022-01-31 05:28:21	Pon	0

Wartość `period_no` równa -1 wskazuje, że funkcja `inweektodate ()` porównuje wejściowy segment kwartału z poprzednim tygodniem. Segment tygodnia początkowo oznacza okres od 9 do 14 stycznia. Wartość `period_no` następnie przesuwa zarówno początkową, jak i końcową granicę tego segmentu do jednego tygodnia wcześniej, powodując, że datami granicznymi stają się 2 stycznia i 7 stycznia.

Diagram kalendarza pokazujący daty transakcji, które zwrócą wynik logiczny TRUE

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

W związku z tym każda transakcja, która ma miejsce między 2 a 8 stycznia (ale bez uwzględniania 8 stycznia), zwróci wynik logicznyTRUE.

Przykład 3 – first_week_day

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola, `in_week_to_date`, które określa, jakie transakcje miały miejsce w tygodniu do 14 stycznia 2022 r.
- Tworzenie dodatkowego pola o nazwie `weekday` przy użyciu funkcji `weekday()`. Pokazuje, który dzień tygodnia odpowiada poszczególnym datom.

Na tym przykładzie jako pierwszy dzień tygodnia używany jest poniedziałek.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `week_day`

- `in_week_to_date`

Tabela wynikowa

date	week_day	in_week_to_date
2022-01-02 12:22:06	Nie	0
2022-01-05 01:02:30	Śro	0
2022-01-06 15:36:20	Thu	0
2022-01-08 10:58:35	Sat	0
2022-01-09 08:53:32	Nie	0
2022-01-10 21:13:01	Pon	-1
2022-01-11 00:57:13	Wto	-1
2022-01-12 09:26:02	Śro	-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	Nie	0
2022-01-17 10:44:16	Pon	0
2022-01-18 18:48:17	Wto	0
2022-01-26 04:36:03	Śro	0
2022-01-27 08:07:49	Thu	0
2022-01-28 12:24:29	Fri	0
2022-01-30 11:56:56	Nie	0
2022-01-30 14:40:19	Nie	0
2022-01-31 05:28:21	Pon	0

Kiedy używa się 0 jako argumentu `first_week_day`] w funkcji `inweektodate()`, argument funkcji zastępuje zmienną systemową `FirstWeekDay` i ustawia poniedziałek jako pierwszy dzień tygodnia.

Diagram kalendarza pokazujący daty transakcji, które zwrócą wynik logiczny TRUE

Mon	Tue	Wed	Thu	Fri	Sat	Sun
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	17
24	25	26	27	28	29	30
31						

W związku z tym każda transakcja, która ma miejsce między 10 a 14 stycznia, zwróci wynik logiczny TRUE, podczas gdy transakcje z datami poza tymi granicami zwrócą wartość FALSE.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie. Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenie określające, które transakcje miały miejsce w tygodniu do 14 stycznia 2022 r., jest tworzone jako miara w obiekcie wykresu.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.
2. Aby obliczyć, czy transakcje miały miejsce w tym samym tygodniu do 14 stycznia, utwórz następującą miarę:
   `=inweektodate(date,'01/14/2022',0)`
3. Aby pokazać, który dzień tygodnia odpowiada poszczególnym datom, utwórz dodatkową miarę:
   `=weekday(date)`

Tabela wynikowa

date	week_day	in_week_to_date
2022-01-02 12:22:06	Nie	0
2022-01-05 01:02:30	Śro	0
2022-01-06 15:36:20	Thu	0
2022-01-08 10:58:35	Sat	0
2022-01-09 08:53:32	Nie	-1
2022-01-10 21:13:01	Pon	-1
2022-01-11 00:57:13	Wto	-1
2022-01-12 09:26:02	Śro	-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	Nie	0
2022-01-17 10:44:16	Pon	0
2022-01-18 18:48:17	Wto	0

date	week_day	in_week_to_date
2022-01-26 04:36:03	Śro	0
2022-01-27 08:07:49	Thu	0
2022-01-28 12:24:29	Fri	0
2022-01-30 11:56:56	Nie	0
2022-01-30 14:40:19	Nie	0
2022-01-31 05:28:21	Pon	0

Pole `in_week_to_date` jest tworzone jako miara w obiekcie wykresu przy użyciu funkcji `inweektodate()`. Pierwszy podany argument określa, które pole jest oceniane. Drugi argument to ustalona data 14 stycznia – jest to argument `base_date`, który wskazuje tydzień do posegmentowania i identyfikuje granicę końcową tego segmentu. Argument `period_no` o wartości 0 jest ostatnim argumentem, co oznacza, że funkcja nie porównuje tygodni poprzedzających segmentowany tydzień ani następujących po nim.

Zmienna systemowa `FirstWeekDay` określa, że tygodnie zaczynają się w niedzielę i kończą w sobotę. W związku z tym styczeń zostałby podzielony na tygodnie zgodnie z poniższym wykresem, a daty między 9 a 14 stycznia stanowiłyby ważny okres dla obliczenia `inweekdodate()`:

Diagram kalendarza pokazujący daty transakcji, które zwrócą wynik logiczny TRUE

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Wszelkie transakcje, które występują między 9 a 14 stycznia, zwracają jako wynik wartość logiczną `TRUE`. Transakcje przed i po tych datach zwracają wynik logiczny `FALSE`.

Przykład 5 – scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Products`.
- Informacje dotyczące identyfikatora produktu, daty produkcji i kosztu.

Stwierdzono, że z powodu błędu sprzętowego produkty wytworzone w tygodniu z 12 stycznia były wadliwe. Problem został rozwiązany 13 stycznia. Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał według tygodni status produktów: „defective” (wadliwe) lub „faultless” (bez wad) oraz koszt produktów wytworzonych w danym tygodniu.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę. Utwórz wymiar, aby pokazać nazwy tygodni:
   `=weekname(manufacture_date)`
2. Następnie utwórz wymiar, aby określić, które produkty są wadliwe, a które bez wad:
   `=if(inWeektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')`
3. Utwórz miarę, aby zsumować `cost_price` produktów:
   `=sum(cost_price)`
4. Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

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

Funkcja `inweektodate()` zwraca wartość logiczną podczas oceny dat wytworzenia każdego z produktów. Dla tych, które zwracają wartość logiczną , oznacza produkty jako .TRUE'Defective' W przypadku każdego produktu zwracającego wartość `FALSE`, a zatem niewyprodukowanego w tygodniu do 12 stycznia, oznacza ona produkty jako `'Faultless'`.

inyear

Ta funkcja zwraca wartość True, jeśli znacznik czasu **timestamp** należy do roku zawierającego wartość **base_date**.

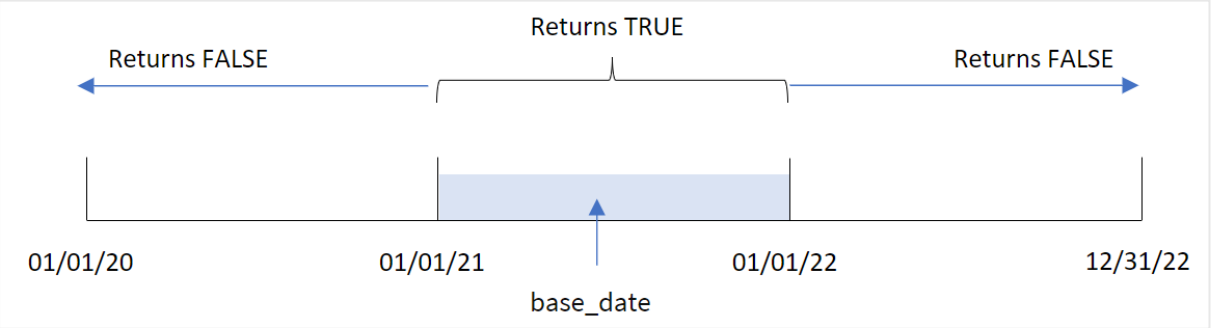
Składnia:

```
InYear (timestamp, base_date, period_no [, first_month_of_year])
```

Typ zwracanych danych: Wartość logiczna

W Qlik Sense wartość logiczna Prawda jest reprezentowana przez -1, a wartość Fałsz jest reprezentowana przez 0.

Diagram zakresu funkcji `inyear()`



Funkcja `inyear()` zwraca wartość logiczną na podstawie porównania wybranych wartości dat z rokiem zdefiniowanym przez `base_date`.

Kiedy używać

Funkcja `inyear()` zwraca wynik logiczny. Zazwyczaj ten typ funkcji będzie używany jako warunek w `if expression`. Wartością zwrotną są obliczenia lub agregacja zależne od tego, czy oceniana data wypada w rozważanym roku. Na przykład, za pomocą funkcji `inyear()` można zidentyfikować całą sprzedaż, jaka miała miejsce w zdefiniowanym roku.

Argumenty

Argument	Opis
timestamp	Data, która ma być porównana z wartością **base_date**.
base_date	Data używana do oceny roku.
period_no	Rok może zostać przesunięty o wartość **period_no**. **period_no** jest liczbą całkowitą, gdzie 0 oznacza rok zawierający wartość **base_date**. Wartości ujemne parametru **period_no** oznaczają lata poprzednie, a wartości dodatnie – lata następne.
first_month_of_year	Jeśli użytkownik zamierza korzystać z lat (obrotowych), które nie zaczynają się w styczniu, powinien wskazać wartość od 2 do 12 jako parametr **first_month_of_year**.

Aby ustawić pierwszy miesiąc roku w argumencie `first_month_of_year`, możesz użyć następujących wartości:

first_month_of_year values

Miesiąc	Wartość
Luty	2
Marzec	3
Kwiecień	4
May	5
Czerwiec	6
Lipiec	7
Sierpień	8
Wrzesień	9
Październik	10
Listopad	11
Grudzień	12

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne

czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
`inyear ('01/25/2013', '01/01/2013', 0 )`	Zwraca TRUE
`inyear ('01/25/2012', '01/01/2013', 0)`	Zwraca FALSE
`inyear ('01/25/2013', '01/01/2013', -1)`	Zwraca FALSE
`inyear ('01/25/2012', '01/01/2013', -1 )`	Zwraca TRUE
`inyear ('01/25/2013', '01/01/2013', 0, 3)`	Zwraca TRUE Wartości argumentów `base_date` i `first_month_of_year` oznaczają, że znacznik czasu musi mieścić się w przedziale od 03.01.2012 r. do 28.02.2013 r.
`inyear ('03/25/2013', '07/01/2013', 0, 3 )`	Zwraca TRUE

Przykład 1 – podstawowy przykład

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za lata 2020-2022, który jest ładowany do tabeli o nazwie `Transactions`.

- Poprzedzające ładowanie zawiera funkcję `inyear()` ustawioną jako pole `in_year` określającą, które transakcje zostały zawarte w tym samym roku, w którym wypada data 26 lipca 2021.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `in_year`

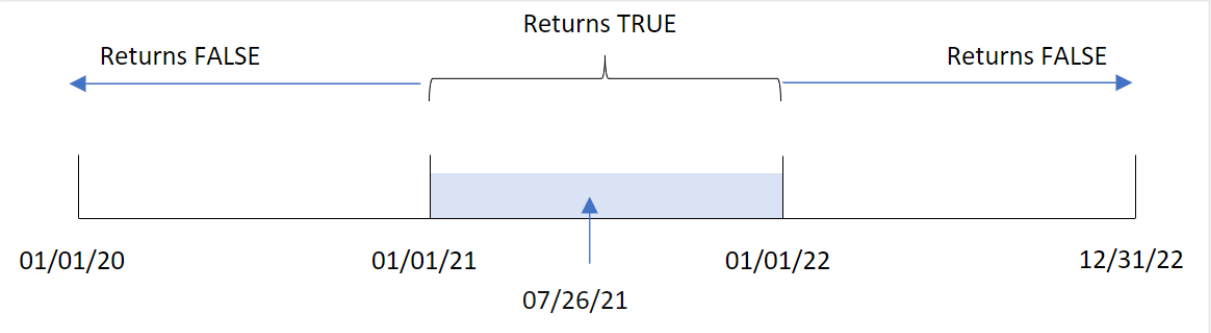
Tabela wynikowa

date	in_year
01/13/2020	0
02/26/2020	0

date	in_year
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

Pole `in_year` jest tworzone w poprzedniej instrukcji load przy użyciu funkcji `inyear()`. Pierwszy argument określa, które pole jest oceniane. Drugi argument to wpisana bezpośrednio w kod data 26 lipca 2021 r. – jest to argument `base_date` określający rok porównawczy. Ostatni jest argument `period_no` o wartości `0`, który oznacza, że funkcja `inyear()` nie porównuje lat poprzednich ani następnych.

Diagram zakresu funkcji `inyear()` z 26 lipca ustawionym jako data bazowa



Każda transakcja zawarta w 2021 r. da w wyniku logiczną prawdę.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za lata 2020-2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Poprzedzające ładowanie zawiera funkcję `inyear()` ustawioną jako pole `previous_year` określającą, które transakcje zostały zawarte w tym roku poprzedzającym rok obejmujący datę 26 lipca 2021.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

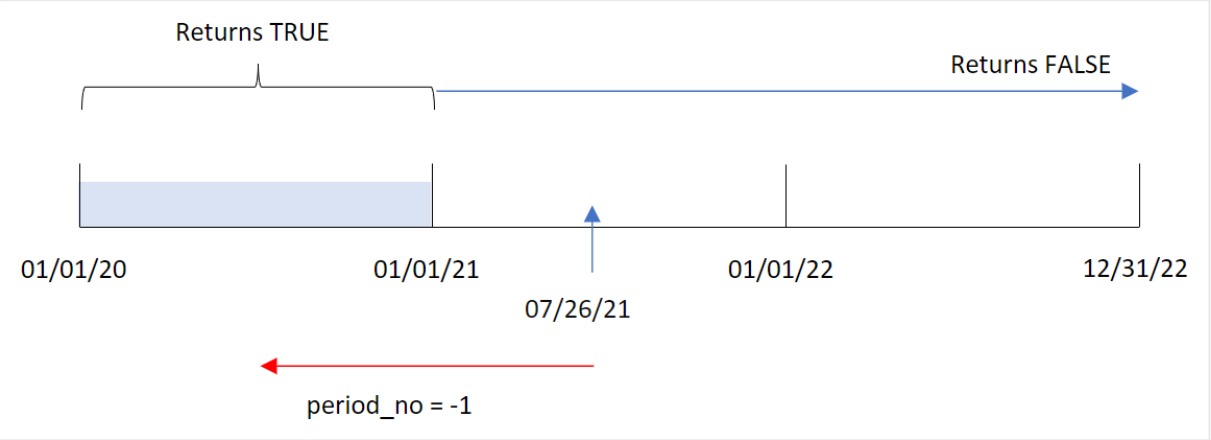
- `date`
- `previous_year`

Tabela wynikowa

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
11/14/2022	0
12/12/2022	0

Przekazanie funkcji `inyear()` argumentu `period_no` o wartości `-1` powoduje przesunięcie granic roku porównawczego o jeden pełny rok. Początkowo rok 2021 jest określony jako porównawczy. `period_no` przesuwa rok porównawczy o jeden, na rok 2020.

Diagram zakresu funkcji `inyear()` z argumentem period_no ustawionym na -1



W efekcie każda transakcja zawarta w 2020 r. daje w wyniku logiczną prawdę.

Przykład 3 – first_month_of_year

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za lata 2020-2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Poprzedzające ładowanie zawiera funkcję `inyear()` ustawioną jako pole `in_year` określającą, które transakcje zostały zawarte w tym samym roku, w którym wypada data 26 lipca 2021.

W tym przykładzie przyjęto jednak, że marzec ma być pierwszym miesiącem roku podatkowego.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `in_year`

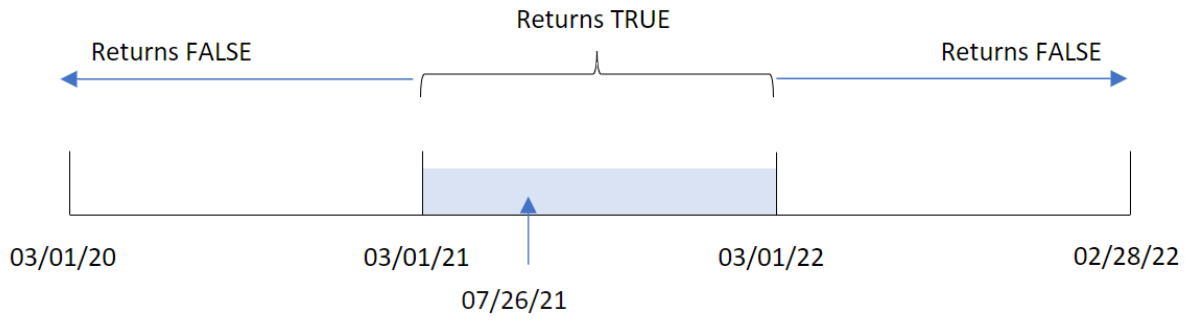
Tabela wynikowa

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	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

date	in_year
12/27/2021	-1
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Przekazanie funkcji `inyear()` argumentu `first_month_of_year` o wartości `3` ustawia początek roku na 1 marca, a koniec - na ostatni dzień lutego.

Diagram zakresu funkcji `inyear()` z marcem jako pierwszym miesiącem roku



W związku z tym każda transakcja zawarta między 1 marca 2021 r. a 1 marca 2022 r. spowoduje zwrócenie w wyniku logicznej prawdy.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie zestaw danych pozostaje bez zmian i jest ładowany do aplikacji. Obliczenia określające, czy transakcje miały miejsce w tym samym roku, w którym wypada data 26 lipca 2021 r., zostały utworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

- `date`

Aby obliczyć, czy transakcje zostały zawarte w tym samym roku, w którym wypada data 26 lipca 2021 r., utwórz następującą miarę:

- `=inyear(date,'07/26/2021', 0)`

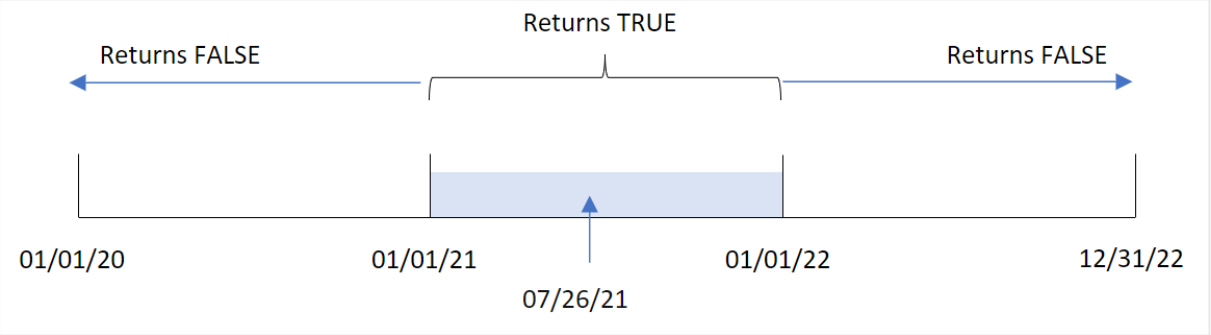
Tabela wynikowa

date	=inyear(date,'07/26/2021',0)
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	-1
02/03/2021	-1

date	=inyear(date,'07/26/2021',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

Pole `in_year` jest tworzone na wykresie za pomocą funkcji `inyear()`. Pierwszy argument określa, które pole jest oceniane. Drugi argument to wpisana bezpośrednio w kod data 26 lipca 2021 r. – jest to argument `base_date` określający rok porównawczy. Ostatni jest argument `period_no` o wartości `0`, który oznacza, że funkcja `inyear()` nie porównuje lat poprzednich ani następnych.

Diagram zakresu funkcji `inyear()` z 27 lipca ustawionym jako data bazowa



Każda transakcja zawarta w 2021 r. da w wyniku logiczną prawdę.

Przykład 5 – scenariusz

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Products`.
- Tabela zawiera następujące pola:

- identyfikator produktu,
- typ produktu,
- data produkcji,
- cena.

Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał według typu produktu koszt produktów wyprodukowanych w 2021 r.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

- `product_type`

Aby obliczyć sumę wszystkich produktów wyprodukowanych w 2021 r., utwórz następującą miarę:

- `=sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))`

Wykonaj następujące czynności:

1. Ustaw **Formatowanie liczb** miary na **Waluta**.
2. W sekcji **Wygląd** wyłącz **Sumy**.

Tabela wynikowa

product_type	=sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))
product A	$95.93
product B	$128.66
product C	$61.89
product D	$171.21

Funkcja `inyear()` zwraca wartość logiczną podczas oceny dat wytworzenia każdego z produktów. Dla każdego produktu wyprodukowanego w 2021 r. funkcja `inyear()` zwraca wartość logiczną TRUE i pokazuje sumę wartości `cost_price`.

inyeartodate

Ta funkcja zwraca wartość True, jeśli wartość **timestamp** należy do części roku zawierającego wartość **base_date**, z dokładnością do ostatniej milisekundy wartości **base_date** włącznie.

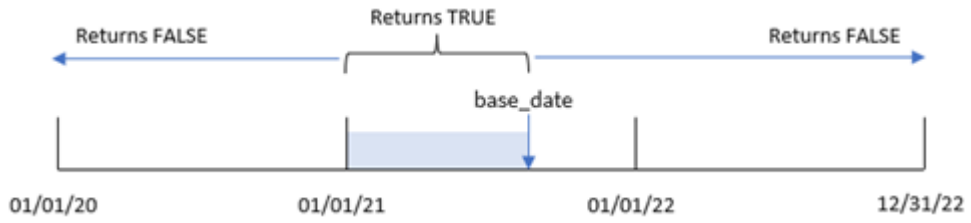
Składnia:

```
InYearToDate (timestamp, base_date, period_no[, first_month_of_year])
```

Typ zwracanych danych: Wartość logiczna

> *W Qlik Sense wartość logiczna Prawda jest reprezentowana przez -1, a wartość Fałsz jest reprezentowana przez 0.*

Diagram funkcji inyeartodate



Funkcja `inyeartodate()` dokona segmentacji określonej części roku przy użyciu `base_date`, identyfikując maksymalną dozwoloną datę dla tego segmentu roku. Następnie funkcja ocenia, czy pole lub wartość daty należy do tego segmentu, i zwraca wynik logiczny.

Argumenty

Argument	Opis
timestamp	Data, która ma być porównana z wartością **base_date**.
base_date	Data używana do oceny roku.
period_no	Rok może zostać przesunięty o wartość **period_no**. **period_no** jest liczbą całkowitą, gdzie 0 oznacza rok zawierający wartość **base_date**. Wartości ujemne parametru **period_no** oznaczają lata poprzednie, a wartości dodatnie – lata następne.
first_month_of_year	Jeśli użytkownik zamierza korzystać z lat (obrotowych), które nie zaczynają się w styczniu, powinien wskazać wartość od 2 do 12 jako parametr **first_month_of_year**.

Kiedy używać

Funkcja `inyeartodate()` zwraca wynik logiczny. Zazwyczaj ten typ funkcji będzie używany jako warunek w wyrażeniu `if`. Spowoduje to zwrócenie agregacji lub obliczenia w zależności od tego, czy oceniana data nastąpiła w roku do danej daty włącznie.

Na przykład funkcja `inyeartodate()` może służyć do identyfikacji całego sprzętu wyprodukowanego w roku do określonej daty.

W tych przykładach używany jest format daty MM/DD/YYYY. Format daty jest określony w instrukcji `SET DateFormat` u góry skryptu ładowania danych. Format zastosowany w przykładach można zmienić, aby dostosować go do konkretnych potrzeb.

Przykłady funkcji

Przykład	Wynik
`inyeartodate ('01/25/2013', '02/01/2013', 0)`	Zwraca wartość `TRUE`.
`inyeartodate ('01/25/2012', '01/01/2013', 0)`	Zwraca wartość `FALSE`.
`inyeartodate ('01/25/2012', '02/01/2013', -1)`	Zwraca wartość `TRUE`.
`inyeartodate ('11/25/2012', '01/31/2013', 0, 4)`	Zwraca wartość `TRUE`. Wartość `timestamp` przypada w roku obrotowym zaczynającym się w czwartym kwartale, przed wartością `base_date`.
`inyeartodate ('3/31/2013', '01/31/2013', 0, 4 )`	Zwraca wartość `FALSE`. W porównaniu z poprzednim przykładem wartość `timestamp` wciąż mieści się w roku obrotowym, ale jest po wartości `base_date`, dlatego przypada poza częścią roku.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za lata 2020-2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie pola, `in_year_to_date`, które określa, jakie transakcje miały miejsce w roku do 26 lipca 2021 r.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `in_year_to_date`

Tabela wynikowa

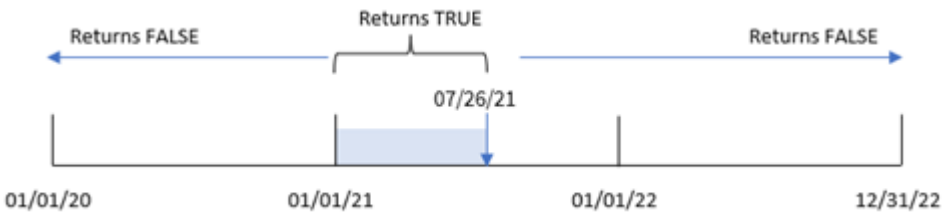
date	in_year_to_date
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
06/14/2020	0
08/07/2020	0
09/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1

date	in_year_to_date
07/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Pole `in_year_to_date` jest tworzone w poprzedniej instrukcji load przy użyciu funkcji `inyeartodate()`. Pierwszy podany argument określa, które pole jest oceniane.

Drugi argument to ustalona data 26 lipca 2021 r. – jest to argument `base_date`, który wskazuje granicę końcową segmentu roku. Argument `period_no` o wartości 0 jest ostatnim argumentem, co oznacza, że funkcja nie porównuje lat poprzedzających segmentowany rok ani następujących po nim.

Diagram funkcji inyeartodate, bez dodatkowych argumentów



Wszelkie transakcje, które występują między 1 stycznia a 26 lipca, zwracają jako wynik wartość logiczną `TRUE`. Daty transakcji sprzed 2021 roku i po 26 lipca 2021 r. zwracają `FALSE`.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola, `previous_year_to_date`, które określa, jakie transakcje miały miejsce w pełnym roku przed segmentem roku kończącego się 26 lipca 2021 r.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `previous_year_to_date`

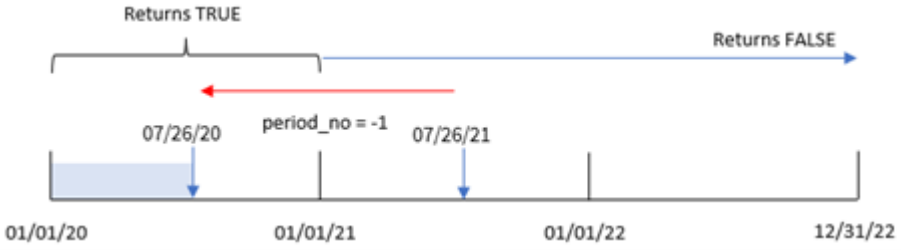
Tabela wynikowa

date	previous_year_to_date
01/13/2020	-1
02/26/2020	-1
03/27/2020	-1
04/16/2020	-1
05/21/2020	-1
06/14/2020	-1
08/07/2020	0

date	previous_year_to_date
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

Wartość `period_no` równa -1 wskazuje, że funkcja `inyeartodate ()` porównuje wejściowy segment kwartału z poprzednim rokiem. Z datą wejściową 26 lipca 2021 r. segment od 1 stycznia 2021 r. do 26 lipca 2021 r. został wstępnie zidentyfikowany jako od początku roku. Wartość `period_no` następnie przesuwa ten segment o cały rok wcześniej, powodując, że datami granicznymi są 1 stycznia i 26 lipca.

Diagram funkcji inyeartodate, przykład period_no



W związku z tym każda transakcja, która ma miejsce między 1 stycznia a 26 lipca, zwróci wynik logicznyTRUE.

Przykład 3 – first_month_of_year

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola, `in_year_to_date`, które określa, jakie transakcje miały miejsce w tym samym roku do 26 lipca 2021 r.

W tym przykładzie jako pierwszy miesiąc roku obrachunkowego ustawiliśmy marzec.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

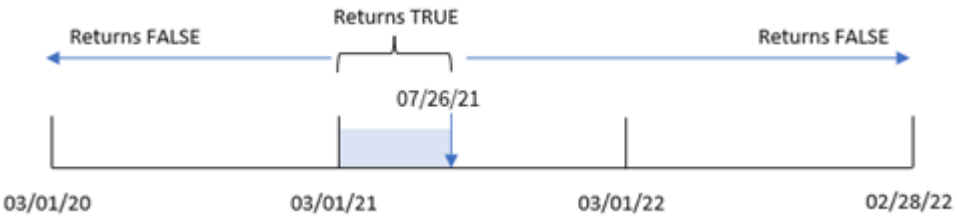
- `date`
- `in_year_to_date`

Tabela wynikowa

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

Kiedy używa się 3 jako argumentu `first_month_of_year` w funkcji `inyeartodate()`, funkcja rozpoczyna rok 1 marca. Wartość `base_date` 26 lipca 2021 r. następnie ustawia datę końcową dla tego segmentu roku.

Diagram funkcji inyeartodate, przykład first_month_of_year



W związku z tym każda transakcja, która ma miejsce między 1 marca a 26 lipca 2021 r., zwróci wynik logiczny TRUE, podczas gdy transakcje z datami poza tymi granicami zwrócą wartość FALSE.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie. Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenie określające, które transakcje miały miejsce w tym samym roku do 26 lipca 2021 r., jest tworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:`date`.

Utwórz następującą miarę:

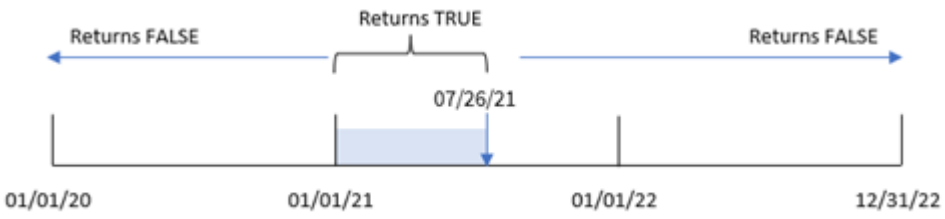
```
=inyeartodate(date,'07/26/2021', 0)
```

Tabela wynikowa

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

Miara `in_year_to_date` jest tworzona w obiekcie wykresu przy użyciu funkcji `inyeartodate()`. Pierwszy podany argument określa, które pole jest oceniane. Drugi argument to ustalona data 26 lipca 2021 r. – jest to argument `base_date`, który wskazuje granicę końcową segmentu roku komparatora. Argument `period_no` o wartości 0 jest ostatnim argumentem, co oznacza, że funkcja nie porównuje lat poprzedzających segmentowany rok ani następujących po nim.

Diagram funkcji inyeartodate, przykład obiektu wykresu



Wszelkie transakcje, które występują między 1 stycznia a 26 lipca 2021 r., zwracają jako wynik wartość logiczną TRUE. Daty transakcji sprzed 2021 roku i po 26 lipca 2021 r. zwracają FALSE.

Przykład 5 – scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Products`.
- Informacje dotyczące identyfikatora produktu, typu produktu, daty produkcji i kosztu.

Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał według typu produktu koszt produktów wyprodukowanych w 2021 r. do 26 lipca.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:`product_type`.

Utwórz miarę, która oblicza sumę każdego produktu wyprodukowanego w 2021 r. przed 27 lipca:

```
=sum(if(inyeartodate(manufacture_date,makedate(2021,07,26),0),cost_price,0))
```

Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

product_type	=sum(if(inyeartodate(manufacture_date,makedate (2021,07,26),0),cost_price,0))
product A	$95.93
product B	$128.66
product C	$61.89
product D	$146.09

Funkcja `inyeartodate()` zwraca wartość logiczną podczas oceny dat wytworzenia każdego z produktów. W przypadku każdego produktu wyprodukowanego w 2021 r. przed 27 lipca funkcja `inyeartodate()` zwraca wartość logiczną `TRUE` i sumuje `cost_price`.

Produkt D to jedyny produkt, który został wyprodukowany również po 26 lipca 2021 roku. Pozycja z `product_ID` 8203 została wyprodukowana 27 grudnia i kosztowała 25,12 USD. W związku z tym ten koszt nie został uwzględniony w sumie produktu D w obiekcie wykresu.

lastworkdate

Funkcja **lastworkdate** zwraca najwcześniejszą datę zakończenia, gdy możliwe jest uzyskanie parametru **no_of_workdays** (poniedziałek-piątek) z początkiem w dniu **start_date** z uwzględnieniem wszelkich opcjonalnie wyszczególnionych dni wolnych (**holiday**). Parametry **start_date** i **holiday** powinny być poprawnymi datami lub znacznikami czasu.

Składnia:

```
lastworkdate(start_date, no_of_workdays {, holiday})
```

Typ zwracanych danych: integer

Kalendarz pokazujący sposób użycia funkcji `lastworkdate()`

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			

Ograniczenia

Nie ma metody pozwalającej zmodyfikować funkcję `lastworkdate()` dla regionów lub scenariuszy uwzględniających cokolwiek innego niż tydzień roboczy rozpoczynający się w poniedziałek i kończący się w piątek.

Parametr holiday musi być stałą łańcuchową. Nie może być wyrażeniem.

Kiedy używać

Funkcja `lastworkdate()` jest używana w wyrażeniach, za pomocą których użytkownik chce obliczyć proponowaną datę zakończenia projektu lub zadania na podstawie daty jego rozpoczęcia i wypadających w tym okresie świąt.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Argumenty

Argument	Opis
start_date	Data rozpoczęcia do oceny.
no_of_ workdays	Liczba dni roboczych do uzyskania.
holiday	Okresy wolne od pracy wyłączane z dni roboczych. Dzień wolny jest określany jako data stała w formie ciągu. Można określić większą liczbę dat dni wolnych, rozdzielając je przecinkami. **Przykład:** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Przykład 1 – podstawowy przykład

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający identyfikatory projektów, daty rozpoczęcia projektów oraz szacowaną wymaganą ilość pracy w dniach. Zestaw danych jest ładowany do tabeli o nazwie `Projects`.
- Poprzedzające ładowanie zawiera funkcję `lastworkdate()` ustawioną jako pole `end_date`, które identyfikuje zaplanowaną datę zakończenia każdego projektu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `start_date`
- `effort`
- `end_date`

Tabela wynikowa

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

Ponieważ nie ma zaplanowanych świąt, funkcja dodaje zdefiniowaną liczbę dni roboczych, od poniedziałku do piątku, do daty początkowej i w ten sposób oblicza najwcześniejszy możliwy termin zakończenia.

Poniższy kalendarz pokazuje datę rozpoczęcia i zakończenia dla projektu 3. Dni robocze są zaznaczone na zielono.

Kalendarz pokazujący datę rozpoczęcia i zakończenia projektu 3

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17 Start Date	18	19	20	21
22	23 End Date	24	25	26	27	28
29	30	31				

Przykład 2 – jeden dzień świąteczny

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający identyfikatory projektów, daty rozpoczęcia projektów oraz szacowaną wymaganą ilość pracy w dniach. Zestaw danych jest ładowany do tabeli o nazwie `Projects`.
- Poprzedzające ładowanie zawiera funkcję `lastworkdate()` ustawioną jako pole `end_date`, które identyfikuje zaplanowaną datę zakończenia każdego projektu.

Dnia 18 maja 2022 jest zaplanowane święto. Funkcja `lastworkdate()` w poprzednim ładowaniu uwzględnia to święto w trzecim argumencie, aby określić termin zakończenia każdego projektu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `start_date`
- `effort`
- `end_date`

Tabela wynikowa

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

Dzień świąteczny został wprowadzony jako trzeci argument do funkcji `lastworkdate()`. W efekcie data zakończenia projektu 3 zostaje przesunięta o jeden dzień do przodu, ponieważ święto zajmuje jeden dzień roboczy przed datą zakończenia.

Poniższy kalendarz pokazuje daty rozpoczęcia i zakończenia projektu 3 oraz to, że święto zmienia datę zakonczenia projektu o jeden dzień.

Kalendarz pokazujący datę rozpoczęcia i zakończenia projektu 3 z uwzględnieniem święta 18 maja

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				

Przykład 3 – kilka dni świątecznych

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający identyfikatory projektów, daty rozpoczęcia projektów oraz szacowaną wymaganą ilość pracy w dniach. Zestaw danych jest ładowany do tabeli o nazwie `Projects`.
- Poprzedzające ładowanie zawiera funkcję `lastworkdate()` ustawioną jako pole `end_date`, które identyfikuje zaplanowaną datę zakończenia każdego projektu.

Należy uwzględnić cztery dni świąteczne wypadające 19, 20, 21 i 22 maja. Funkcja `lastworkdate()` w poprzednim ładowaniu uwzględnia te święta w trzecim argumencie, aby określić termin zakończenia każdego projektu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `start_date`
- `effort`
- `end_date`

Tabela wynikowa

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

Cztery dni świąteczne zostają przekazane jako lista argumentów do funkcji `lastworkdate()` po dacie rozpoczęcia i liczbie dni roboczych.

Poniższy kalendarz pokazuje daty rozpoczęcia i zakończenia projektu 3 oraz to, że święta zmieniają datę zakończenia projektu o trzy dni.

Kalendarz pokazujący datę rozpoczęcia i zakończenia projektu 3 z uwzględnieniem świąt od 19 do 22 maja

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				

Przykład 4 – jeden dzień świąteczny (wykres)

Skrypt ładowania i wyrażenie wykresu

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

W tym przykładzie zestaw danych pozostaje bez zmian i jest ładowany do aplikacji. Pole `end_date` jest obliczane jako miara na wykresie.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `start_date`
- `effort`

Aby obliczyć datę zakończenia (end_date), należy utworzyć następującą miarę:

- `=LastWorkDate(start_date,effort,'05/18/2022')`

Tabela wynikowa

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

Pojedynczy dzień świąteczny został wprowadzony jako miara na wykresie. W efekcie data zakończenia projektu 3 zostaje przesunięta o jeden dzień do przodu, ponieważ święto zajmuje jeden dzień roboczy przed datą zakończenia.

Poniższy kalendarz pokazuje daty rozpoczęcia i zakończenia projektu 3 oraz to, że święto zmienia datę zakonczenia projektu o jeden dzień.

Kalendarz pokazujący datę rozpoczęcia i zakończenia projektu 3 z uwzględnieniem święta 18 maja

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

Ta funkcja zwraca znacznik bieżącego czasu dla podanej strefy czasowej.

Składnia:

```
LocalTime([timezone [, ignoreDST ]])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
timezone	**timezone** określa się jako ciąg znaków zawierający dowolne miejsca geograficzne wymienione w sekcji **Time Zone** w **Windows Control Panel** w odniesieniu do **Date and Time** lub jako ciąg znaków w formacie „GMT+hh:mm”. Jeśli nie określono żadnej strefy czasowej, wówczas zwrócony zostanie czas lokalny.
ignoreDST	Jeśli **ignoreDST** ma wartość -1 (True), czas letni będzie ignorowany.

Przykłady i wyniki:

Poniższe przykłady opierają się na funkcji wywoływanej w dniu 2014-10-22 12:54:47 czasu lokalnego, w lokalnej strefie czasowej GMT+01:00.

Przykłady skryptów

Przykład	Wynik
`localtime ()`	Zwraca czas lokalny 2014-10-22 12:54:47.
`localtime ('London')`	Zwraca czas lokalny w Londynie, 2014-10-22 11:54:47.
`localtime ('GMT+02:00')`	Zwraca czas lokalny w strefie czasowej GMT+02:00, 2014-10-22 13:54:47.
`localtime ('Paris','-1')`	Zwraca czas lokalny w Paryżu bez uwzględniania czasu letniego, 2014-10-22 11:54:47.

lunarweekend

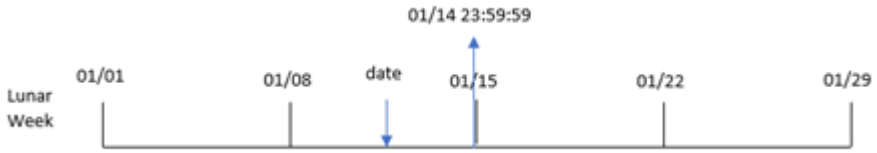
Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu ostatniej milisekundy ostatniego dnia tygodnia księżycowego zawierającego wartość **date**. Tygodnie księżycowe w Qlik Sense są zdefiniowane przez uznanie 1 stycznia za pierwszy dzień tygodnia i każdy tydzień, z wyjątkiem ostatniego tygodnia roku, będzie zawierał dokładnie siedem dni.

Składnia:

```
LunarweekEnd(date[, period_no[, first_week_day]])
```

Typ zwracanych danych: dual

Schemat funkcji `lunarweekend()`



Funkcja `lunarweekend()` sprawdza, do którego tygodnia księżycowego należy `date`. Następnie zwraca znacznik czasu w formacie daty dla ostatniej milisekundy tego tygodnia.

Argumenty

Argument	Opis
date	Data lub znacznik czasu do oszacowania.

Argument	Opis
period_no	Parametr **period_no** jest liczbą całkowitą lub wyrażeniem, którego wynikiem jest liczba całkowita, gdzie wartość 0 wskazuje tydzień księżycowy zawierający wartość **date**. Wartości ujemne parametru **period_no** oznaczają poprzednie tygodnie księżycowe, a wartości dodatnie – następne tygodnie księżycowe.
first_week_day	Przesunięcie może być większe lub mniejsze od zera. Zmienia to początek roku o określoną liczbę dni lub części dnia.

Kiedy używać

Funkcja `lunarweekend()` jest powszechnie używana jako część wyrażenia, gdy użytkownik chce, by w obliczeniach użyto ułamka tygodnia, który jeszcze nie nastąpił. Inaczej niż w przypadku funkcji `weekend()`, ostatni tydzień księżycowy każdego roku kalendarzowego będzie się kończył 31 grudnia. Na przykład, za pomocą funkcji `lunarweekend()` można obliczyć odsetki, które jeszcze nie zostały zapłacone w tym tygodniu.

Przykłady funkcji

Przykład	Wynik
`lunarweekend('01/12/2013')`	Zwraca wartość 01/14/2013 23:59:59.
`lunarweekend('01/12/2013', -1)`	Zwraca wartość 01/07/2013 23:59:59.
`lunarweekend('01/12/2013', 0, 1)`	Zwraca wartość 01/15/2013 23:59:59.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „Transactions”.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie pola `end_of_week` zwracającego znacznik czasu końca tygodnia księżycowego, w którym zostały zawarte transakcje.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `end_of_week`

- end_of_week_timestamp

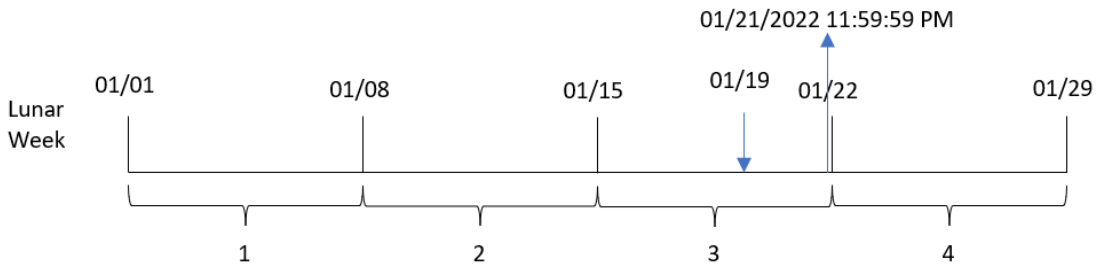
Tabela wynikowa

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

Pole `end_of_week` jest tworzone w poprzedzającej instrukcji ładowania przez użycie funkcji `lunarweekend()` i przekazanie pola `date` jako argumentu funkcji.

Funkcja `lunarweekend()` identyfikuje, w którym tygodniu księżycowym przypada wartość daty, i zwraca znacznik czasu ostatniej milisekundy danego tygodnia.

Diagram przedstawiający przykład użycia funkcji `lunarweekend()` bez dodatkowych argumentów



Transakcja 8189 miała miejsce 19 stycznia. Funkcja `lunarweekend()` oblicza, że tydzień księżycowy zaczyna się 15 stycznia. W związku z tym wartość `end_of_week` dla tamtej transakcji zwraca ostatnią milisekundę tygodnia księżycowego (21 stycznia o godz. 23:59:59).

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola `previous_lunar_week_end` zwracającego znacznik czasu końca tygodnia księżycowego przed transakcją.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `previous_lunar_week_end`
- `previous_lunar_week_end_timestamp`

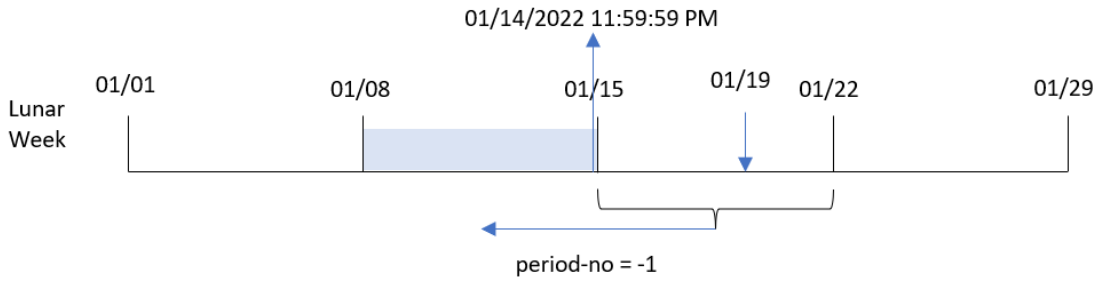
Tabela wynikowa

date	previous_lunar_week_end	previous_lunar_week_end_timestamp
1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
1/19/2022	01/14/2022	1/14/2022 11:59:59 PM
2/5/2022	02/04/2022	2/4/2022 11:59:59 PM
2/28/2022	02/25/2022	2/25/2022 11:59:59 PM
3/16/2022	03/11/2022	3/18/2022 11:59:59 PM
4/1/2022	03/25/2022	3/25/2022 11:59:59 PM
5/7/2022	05/06/2022	5/6/2022 11:59:59 PM
5/16/2022	05/13/2022	5/13/2022 11:59:59 PM
6/15/2022	06/10/2022	6/10/2022 11:59:59 PM
6/26/2022	06/24/2022	6/24/2022 11:59:59 PM
7/9/2022	07/08/2022	7/8/2022 11:59:59 PM
7/22/2022	07/15/2022	7/15/2022 11:59:59 PM
7/23/2022	07/22/2022	7/22/2022 11:59:59 PM
7/27/2022	07/22/2022	7/22/2022 11:59:59 PM
8/2/2022	07/29/2022	7/29/2022 11:59:59 PM

date	previous_lunar_week_end	previous_lunar_week_end_timestamp
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

W tym przypadku, ponieważ wartości `period_no` -1 użyto jako argumentu przesunięcia w funkcji `lunarweekend()`, funkcja najpierw identyfikuje tydzień księżycowy, w którym zawarto transakcje. Następnie przesuwa zakres o tydzień wstecz i identyfikuje ostatnią milisekundę tego tygodnia księżycowego.

Diagram funkcji `lunarweekend()`, przykład z argumentem period_no



Transakcja 8189 miała miejsce 19 stycznia. Funkcja `lunarweekend()` oblicza, że tydzień księżycowy zaczyna się 15 stycznia. W efekcie poprzedni tydzień księżycowy rozpoczynał się 8 stycznia i kończył 14 stycznia o godzinie 23:59:59. Ta wartość jest zwracana dla pola `previous_lunar_week_end`.

Przykład 3 – first_week_day

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie. W tym przykładzie ustawiamy początek tygodni księżycowych na 5 stycznia.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `end_of_week`
- `end_of_week_timestamp`

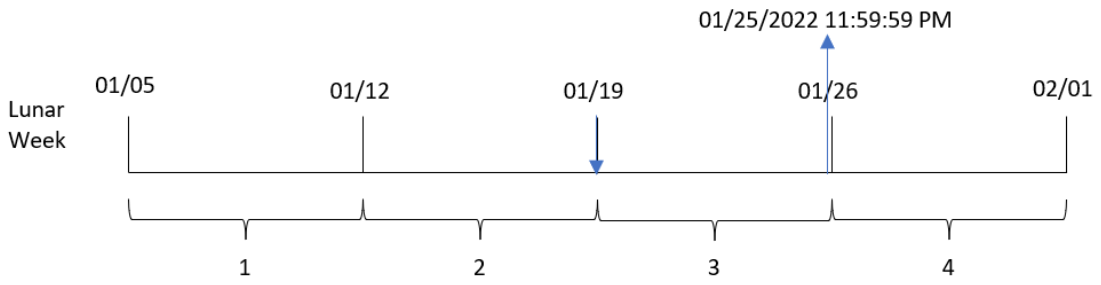
Tabela wynikowa

date	end_of_week	end_of_week_timestamp
1/7/2022	01/11/2022	1/11/2022 11:59:59 PM
1/19/2022	01/25/2022	1/25/2022 11:59:59 PM
2/5/2022	02/08/2022	2/8/2022 11:59:59 PM
2/28/2022	03/01/2022	3/1/2022 11:59:59 PM
3/16/2022	03/22/2022	3/22/2022 11:59:59 PM
4/1/2022	04/05/2022	4/5/2022 11:59:59 PM
5/7/2022	05/10/2022	5/10/2022 11:59:59 PM
5/16/2022	05/17/2022	5/17/2022 11:59:59 PM

date	end_of_week	end_of_week_timestamp
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

W tym przypadku przekazano wartość 4 jako argument `first_week_date` funkcji `lunarweekend()`, w wyniku czego początek roku został przesunięty z 1 stycznia na 5 stycznia.

Diagram funkcji `lunarweekend()`, przykład z argumentem first_week_day



Transakcja 8189 miała miejsce 19 stycznia. Ponieważ tygodnie księżycowe zaczynają się 5 stycznia, funkcja `lunarweekend()` oblicza, że tydzień księżycowy obejmujący datę 19 stycznia zaczyna się właśnie w tym dniu. W efekcie koniec tego tygodnia księżycowego wypada 25 stycznia o godzinie 23:59:59. Ta wartość zostaje zwrócona dla pola `end_of_week`.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenia zwracające znacznik czasu końca tygodnia księżycowego, w którym zawarto transakcje, są tworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Dodaj następujące miary:

```
=lunarweekend(date)
```

```
=timestamp(lunarweekend(date))
```

Tabela wynikowa

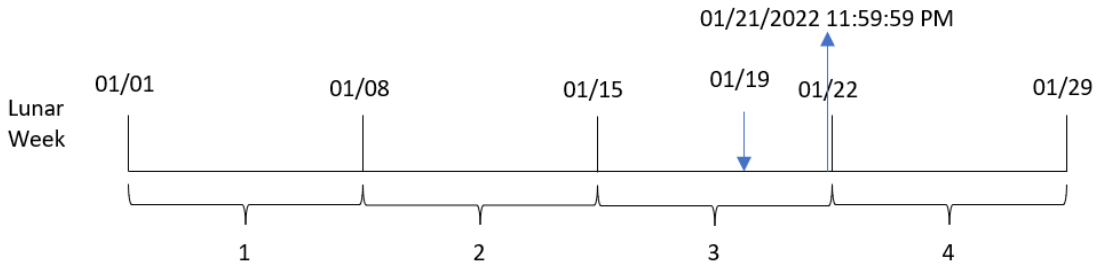
date	=lunarweekend(date)	=timestamp(lunarweekend(date))
1/7/2022	01/07/2022	1/7/2022 11:59:59 PM
1/19/2022	01/21/2022	1/21/2022 11:59:59 PM

date	=lunarweekend(date)	=timestamp(lunarweekend(date))
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

Miarę `end_of_week` tworzy się w obiekcie wykresu, używając funkcji `lunarweekend()` i przekazując pole daty `date` jako jej argument.

Funkcja `lunarweekend()` identyfikuje, w którym tygodniu księżycowym przypada wartość daty, i zwraca znacznik czasu ostatniej milisekundy danego tygodnia.

Diagram funkcji `lunarweekend()`, przykład obiektu wykresu



Transakcja 8189 miała miejsce 19 stycznia. Funkcja `lunarweekend()` oblicza, że tydzień księżycowy zaczyna się 15 stycznia. W związku z tym wartość `end_of_week` dla tamtej transakcji zwraca ostatnią milisekundę tygodnia księżycowego (21 stycznia o godz. 23:59:59).

Przykład 5 – scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Employee_Expenses`.
- Identyfikatory pracowników, imiona i nazwiska pracowników oraz średnie dzienne roszczenia z tytułu wydatków każdego pracownika.

Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał, według identyfikatora pracownika oraz imienia i nazwiska pracownika, szacowane roszczenia z tytułu wydatków do poniesienia przez pozostałą część tygodnia księżycowego.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę.
2. Dodaj następujące pola jako wymiary:
    - `employee_id`
    - `employee_name`
3. Następnie utwórz następującą miarę, aby obliczyć narosłe odsetki:
   `=(lunarweekend(today(1))-today(1))*avg_daily_claim`
4. Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

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

Funkcja `lunarkweekend()`, przyjmująca jako argument tylko dzisiejszą datę, zwraca datę końcową bieżącego tygodnia księżycowego. Następnie wyrażenie odejmuje dzisiejszą datę od daty zakończenia tygodnia księżycowego i zwraca liczbę dni pozostałych w tym tygodniu.

Wartość ta jest następnie mnożona przez średnie dzienne roszczenie z tytułu wydatków przez każdego pracownika, aby obliczyć szacunkową wartość roszczeń, które każdy pracownik złoży w pozostałej części tygodnia księżycowego.

lunarweekname

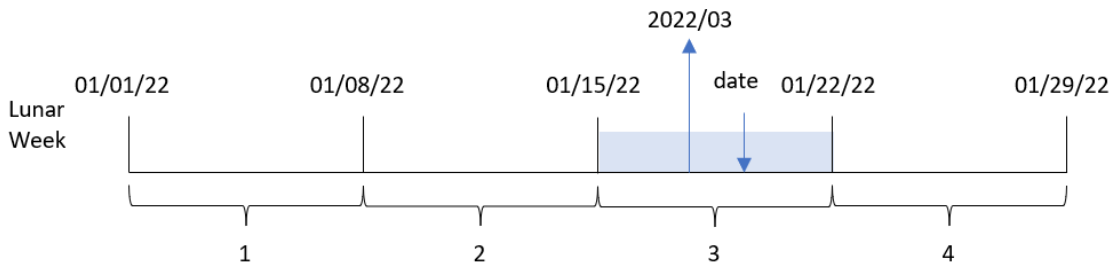
Ta funkcja zwraca wartość pokazującą rok i numer tygodnia księżycowego odpowiadający znacznikowi czasu pierwszej milisekundy pierwszego dnia tygodnia księżycowego zawierającego wartość **date**. Tygodnie księżycowe w Qlik Sense są zdefiniowane przez uznanie 1 stycznia za pierwszy dzień tygodnia i każdy tydzień, z wyjątkiem ostatniego tygodnia roku, będzie zawierał dokładnie siedem dni.

Składnia:

```
LunarWeekName(date [, period_no[, first_week_day]])
```

Typ zwracanych danych: dual

Schemat funkcji `lunarweekname()`



Funkcja `lunarweekname()` określa, w którym tygodniu księżycowym wypada ta data, zaczynając liczenie tygodni od 1 stycznia. Następnie zwraca wartość złożoną z `year/weekcount`.

Argumenty

Argument	Opis
date	Data lub znacznik czasu do oszacowania.
period_no	Parametr **period_no** jest liczbą całkowitą lub wyrażeniem, którego wynikiem jest liczba całkowita, gdzie wartość 0 wskazuje tydzień księżycowy zawierający wartość **date**. Wartości ujemne parametru **period_no** oznaczają poprzednie tygodnie księżycowe, a wartości dodatnie – następne tygodnie księżycowe.
first_week_day	Przesunięcie może być większe lub mniejsze od zera. Zmienia to początek roku o określoną liczbę dni lub części dnia.

Kiedy używać

Funkcja `lunarweekname()` jest przydatna, gdy chcesz porównać agregacje według tygodni księżycowych. Na przykład, za pomocą tej funkcji można obliczyć całkowitą wielkość sprzedaży produktów według tygodnia księżycowego. Tygodnie księżycowe są przydatne, gdy chcemy mieć pewność, że wszystkie wartości znajdujące się w pierwszym tygodniu roku obejmują tylko wartości nie wcześniejsze niż z 1 stycznia.

Te wymiary można utworzyć w skrypcie ładowania za pomocą funkcji tworzenia pola w tabeli kalendarza głównego. Tej funkcji można także użyć bezpośrednio na wykresie jako wymiaru wyliczanego.

Przykłady funkcji

Przykład	Wynik
`lunarweekname('01/12/2013')`	Zwraca wartość 2006/02.
`lunarweekname('01/12/2013', -1)`	Zwraca wartość 2006/01.
`lunarweekname('01/12/2013', 0, 1)`	Zwraca wartość 2006/02.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – data bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „Transactions”.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie pola , `lunar_week_name`, które zwraca rok i numer tygodnia dla tygodnia księżycowego, w którym zawarto transakcję.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `lunar_week_name`

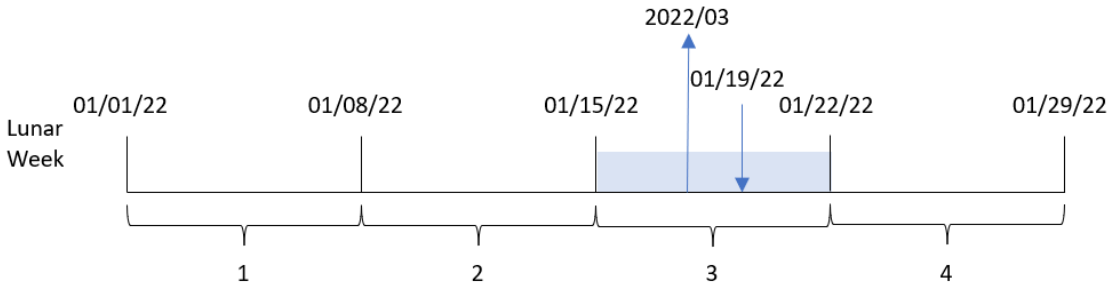
Tabela wynikowa

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
10/14/2022	2022/41
10/29/2022	2022/44

Pole `lunar_week_name` jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `lunarweekname()` i przekazanie pola `date` jako argumentu funkcji.

Funkcja `lunarweekname()` sprawdza, w którym tygodniu księżycowym wypada wartość daty oraz zwraca rok i numer tygodnia obejmujące tę datę.

Diagram przedstawiający przykład użycia funkcji `lunarweekname()` bez dodatkowych argumentów



Transakcja 8189 miała miejsce 19 stycznia. Funkcja `lunarweekname()` oblicza, że ta data wypada w tygodniu księżycowym zaczynającym się 15 stycznia, czyli trzecim tygodniu księżycowym roku. W efekcie dla tej transakcji zostaje zwrócona wartość `lunar_week_name` 2022/03.

Przykład 2 – data z argumentem period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola, `previous_lunar_week_name`, które zwraca rok i numer tygodnia dla tygodnia księżycowego, w którym zawarto transakcję.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `previous_lunar_week_name`

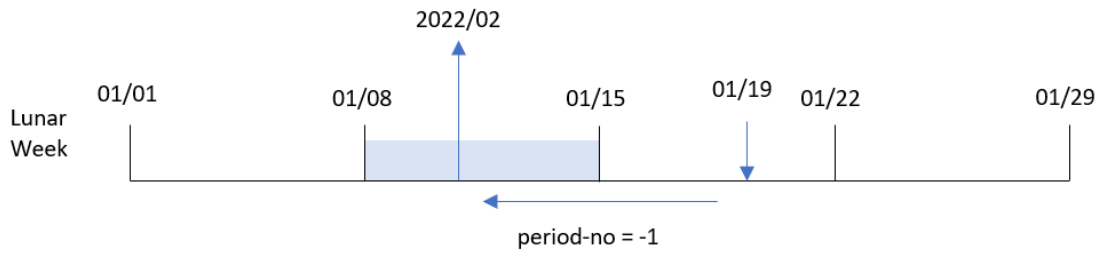
Tabela wynikowa

date	previous_lunar_week_name
1/7/2022	2021/52
1/19/2022	2022/02
2/5/2022	2022/05
2/28/2022	2022/08
3/16/2022	2022/10
4/1/2022	2022/12
5/7/2022	2022/18
5/16/2022	2022/19
6/15/2022	2022/23
6/26/2022	2022/25
7/9/2022	2022/27
7/22/2022	2022/28
7/23/2022	2022/29
7/27/2022	2022/29
8/2/2022	2022/30
8/8/2022	2022/31
8/19/2022	2022/32

date	previous_lunar_week_name
9/26/2022	2022/38
10/14/2022	2022/40
10/29/2022	2022/43

W tym przypadku, ponieważ wartości `period_no` -1 użyto jako argumentu przesunięcia w funkcji `lunarweekname()`, funkcja najpierw identyfikuje tydzień księżycowy, w którym zawarto transakcje. Następnie zwraca rok i numer poprzedniego tygodnia.

Diagram funkcji `lunarweekname()`, przykład z argumentem period_no



Transakcja 8189 miała miejsce 19 stycznia. Funkcja `lunarweekname()` oblicza, że ta transakcja została zawarta w trzecim tygodniu księżycowym roku, więc zwraca rok i wartość dla tygodnia wcześniejszego o jeden, 02.2022, dla pola `previous_lunar_week_name`.

Example 3 – data z argumentem first_week_day

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie. W tym przykładzie ustawiamy początek tygodni księżycowych na 5 stycznia.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

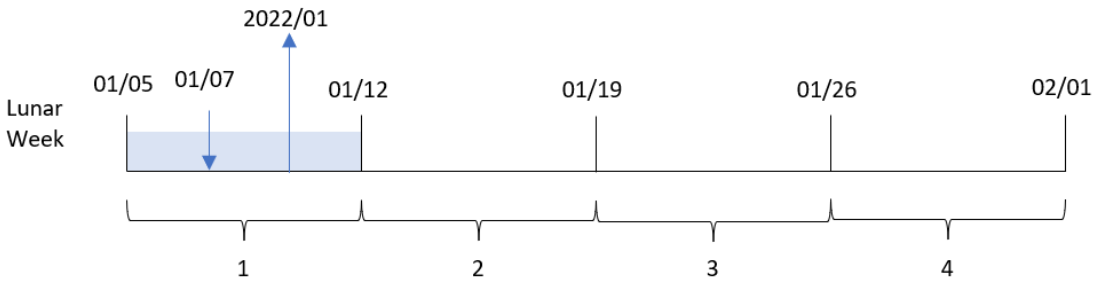
- `date`
- `lunar_week_name`

Tabela wynikowa

date	lunar_week_name
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/05
2/28/2022	2022/08
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/18
5/16/2022	2022/19
6/15/2022	2022/24
6/26/2022	2022/25
7/9/2022	2022/27

date	lunar_week_name
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

Diagram funkcji `lunarweekname()`, przykład z argumentem first_week_day



W tym przypadku przekazano wartość 4 jako argument `first_week_date` funkcji `lunarweekname()`, w wyniku czego początek tygodni księżycowych został przesunięty z 1 stycznia na 5 stycznia.

Transakcja 8188 została zawarta 7 stycznia. Ponieważ tygodnie księżycowe zaczynają się 5 stycznia, funkcja `lunarweekname()` oblicza, że tydzień księżycowy obejmujący 7 stycznia jest pierwszym tygodniem księżycowym roku. W efekcie dla tej transakcji zostaje zwrócona wartość `lunar_week_name` 2022/01.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenia zwracające numer tygodnia księżycowego i rok, w którym zostały zawarte transakcje, są tworzone jako miara na obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Aby obliczyć datę początkową tygodnia księżycowego, kiedy ma miejsce transakcja, utwórz następującą miarę:

```
=lunarweekname(date)
```

Tabela wynikowa

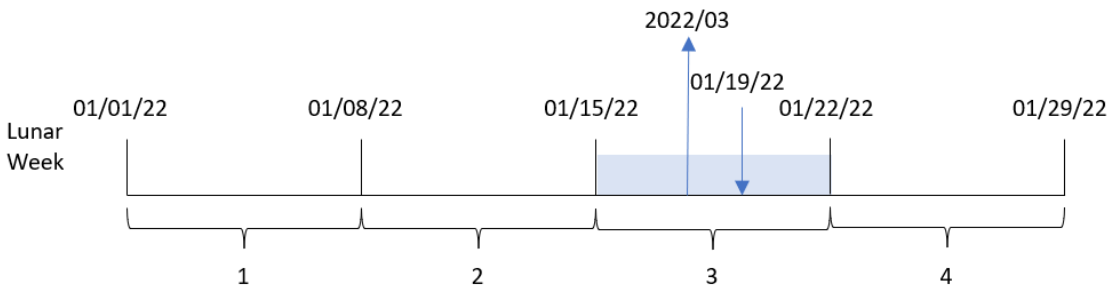
date	=lunarweekname(date)
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/06
2/28/2022	2022/09
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/19

date	=lunarweekname(date)
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

Miarę „`lunar_week_name`" tworzy się w obiekcie wykresu, używając funkcji `lunarweekname()` i przekazując pole daty `date` jako jej argument.

Funkcja `lunarweekname()` sprawdza, w którym tygodniu księżycowym wypada wartość daty oraz zwraca rok i numer tygodnia obejmujące tę datę.

Diagram funkcji `lunarweekname()`, przykład obiektu wykresu



Transakcja 8189 miała miejsce 19 stycznia. Funkcja `lunarweekname()` oblicza, że ta data wypada w tygodniu księżycowym zaczynającym się 15 stycznia, czyli trzecim tygodniu księżycowym roku. W efekcie wartość `lunar_week_name` dla tej transakcji to 2022/03.

Przykład 5 – scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „`Transactions`”.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).

Użytkownik końcowy chciałby otrzymać obiekt wykresu przedstawiający łączną sprzedaż w ujęciu tygodniowym dla bieżącego roku. Tydzień 1, o długości siedmiu dni, powinien zaczynać się 1 stycznia. Można to osiągnąć nawet, jeśli ten wymiar jest nieodstępny w modelu danych przy użyciu funkcji `lunarweekname()` jako wymiaru wyliczanego na wykresie.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę.
2. Utwórz wymiar wyliczany, używając następującego wyrażenia:
   `=lunarweekname(date)`
3. Aby obliczyć łączną sprzedaż, utwórz następującą miarę agregacji:
   `=sum(amount)`
4. Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

=lunarweekname(date)	=sum(amount)
2022/01	$17.17
2022/03	$37.23
2022/06	$57.42
2022/09	$88.27
2022/11	$53.80
2022/13	$82.06
2022/19	$40.39
2022/20	$87.21
2022/24	$95.93
2022/26	$45.89
2022/28	$36.23
2022/29	$25.66
2022/30	$152.75
2022/31	$76.11
2022/32	$25.12
2022/33	$46.23
2022/39	$84.21
2022/41	$96.24
2022/44	$67.67

lunarweekstart

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu pierwszej milisekundy pierwszego dnia tygodnia księżycowego zawierającego wartość **date**. Tygodnie księżycowe w Qlik Sense są zdefiniowane przez uznanie 1 stycznia za pierwszy dzień tygodnia i każdy tydzień, z wyjątkiem ostatniego tygodnia roku, będzie zawierał dokładnie siedem dni.

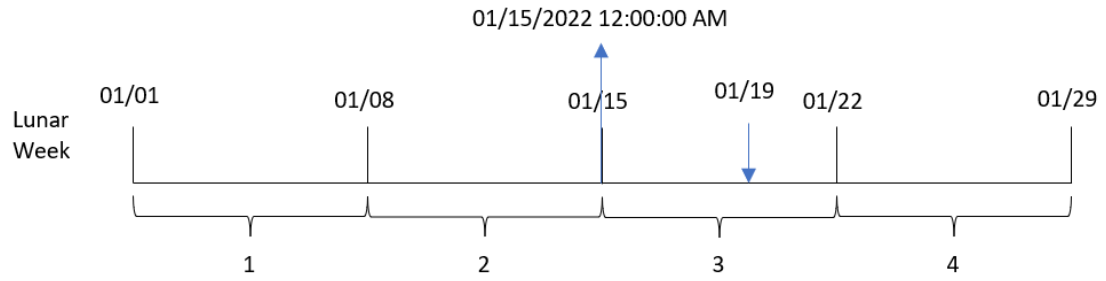
Składnia:

```
LunarweekStart(date[, period_no[, first_week_day]])
```

Typ zwracanych danych: dual

Funkcja `lunarweekstart()` sprawdza, do którego tygodnia księżycowego należy `date`. Następnie zwraca znacznik czasu w formacie daty dla pierwszej milisekundy tego tygodnia.

Przykładowy diagram funkcji `lunarweekstart()`



Argumenty

Argument	Opis
date	Data lub znacznik czasu do oszacowania.
period_no	Parametr **period_no** jest liczbą całkowitą lub wyrażeniem, którego wynikiem jest liczba całkowita, gdzie wartość 0 wskazuje tydzień księżycowy zawierający wartość **date**. Wartości ujemne parametru **period_no** oznaczają poprzednie tygodnie księżycowe, a wartości dodatnie – następne tygodnie księżycowe.
first_week_day	Przesunięcie może być większe lub mniejsze od zera. Zmienia to początek roku o określoną liczbę dni lub części dnia.

Kiedy używać

Funkcja `lunarweekstart()` jest zwykle używana jako część wyrażenia, gdy użytkownik chce, by w obliczeniach użyto ułamka tygodnia, który upłynął do tej pory. Inaczej niż w przypadku funkcji `weekstart()`, na początku każdego nowego roku kalendarzowego, tygodnie zaczynają się od 1 stycznia i każdy kolejny tydzień zaczyna się o siedem dni później. Funkcja`lunarweekstart()` nie uwzględnia zmiennej systemowej `FirstWeekDay`.

Na przykład, za pomocą funkcji `lunarweekstart()` można obliczyć odsetki narosłe w ciągu tygodnia do podanej daty.

Przykłady funkcji

Przykład	Wynik
`lunarweekstart ('01/12/2013')`	Zwraca wartość 01/08/2013.
`lunarweekstart ('01/12/2013', -1)`	Zwraca wartość 01/01/2013.
`lunarweekstart ('01/12/2013', 0, 1 )`	Zwraca 01/09/2013, ponieważ ustawienie argumentu first_week_day na 1 oznacza zmianę początku roku na 01/02/2013.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „`Transactions`”.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie pola `start_of_week` zwracającego znacznik czasu początku tygodnia księżycowego, w którym zostały zawarte transakcje.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

Tabela wynikowa

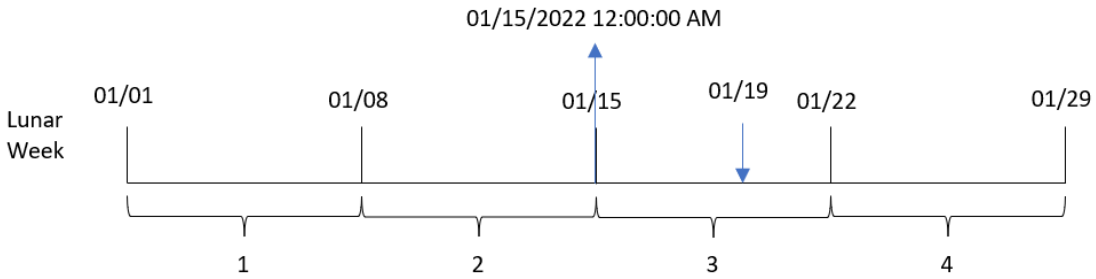
date	start_of_week	start_of_week_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/15/2022	1/15/2022 12:00:00 AM
2/5/2022	02/05/2022	2/5/2022 12:00:00 AM
2/28/2022	02/26/2022	2/26/2022 12:00:00 AM
3/16/2022	03/12/2022	3/12/2022 12:00:00 AM

date	start_of_week	start_of_week_timestamp
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

Pole `start_of_week` jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `lunarweekstart()` i przekazanie pola `date` jako argumentu funkcji.

Funkcja `lunarweekstart()` identyfikuje, w którym tygodniu księżycowym przypada określona data, i zwraca znacznik czasu pierwszej milisekundy danego tygodnia.

Diagram przedstawiający przykład użycia funkcji `lunarweekstart()` bez dodatkowych argumentów



Transakcja 8189 miała miejsce 19 stycznia. Funkcja `lunarweekstart()` oblicza, że tydzień księżycowy zaczyna się 15 stycznia. W związku z tym wartość `start_of_week` dla tamtej transakcji zwraca pierwszą milisekundę tamtego dnia, czyli 15 stycznia, godz. 00:00:00.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola `previous_lunar_week_start` zwracającego znacznik czasu początku tygodnia księżycowego przed transakcją.

Skrypt ładowania

```
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
];
```

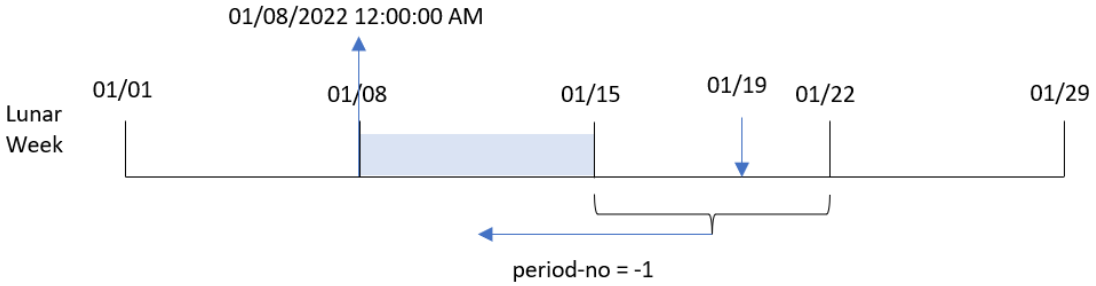
Wyniki

Tabela wynikowa

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

W tym przypadku, ponieważ wartości `period_no` -1 użyto jako argumentu przesunięcia w funkcji `lunarweekstart()`, funkcja najpierw identyfikuje tydzień księżycowy, w którym odbywają się transakcje. Następnie przesuwa zakres o tydzień wstecz i identyfikuje pierwszą milisekundę tego tygodnia księżycowego.

Diagram funkcji `lunarweekstart()`, przykład z argumentem period_no



Transakcja 8189 miała miejsce 19 stycznia. Funkcja `lunarweekstart()` oblicza, że tydzień księżycowy zaczyna się 15 stycznia. W efekcie poprzedni tydzień księżycowy rozpoczynał się 8 stycznia o godzinie 00:00:00. Ta wartość jest zwracana dla pola `previous_lunar_week_start`.

Przykład 3 – first_week_day

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie. W tym przykładzie ustawiamy początek tygodni księżycowych na 5 stycznia.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

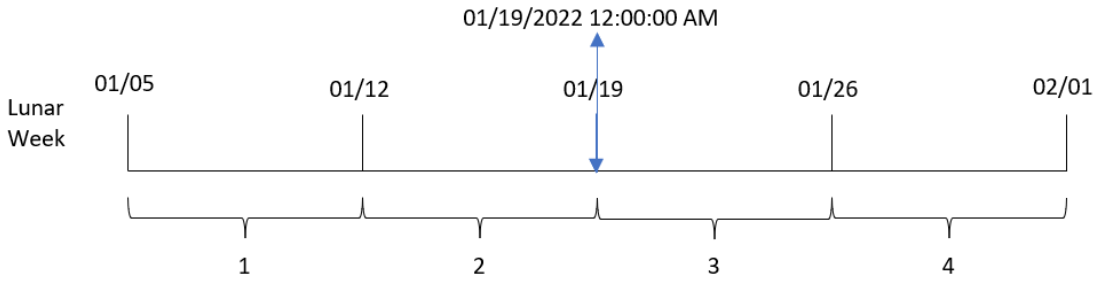
Tabela wynikowa

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

W tym przypadku przekazano wartość 4 jako argument `first_week_date` funkcji `lunarweekstart()`, w wyniku czego początek roku został przesunięty z 1 stycznia na 5 stycznia.

Diagram funkcji `lunarweekstart()`, przykład z argumentem first_week_day



Transakcja 8189 miała miejsce 19 stycznia. Ponieważ tygodnie księżycowe zaczynają się 5 stycznia, funkcja `lunarweekstart()` oblicza, że tydzień księżycowy obejmujący datę 19 stycznia także zaczyna się 19 stycznia o godzinie 00:00:00. W efekcie taka jest wartość zwracana dla pola `start_of_week`.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenia zwracające znacznik czasu początku tygodnia księżycowego, w którym zawarto transakcje, są tworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Dodaj następujące miary:

```
=lunarweekstart(date)
```

```
=timestamp(lunarweekstart(date))
```

Tabela wynikowa

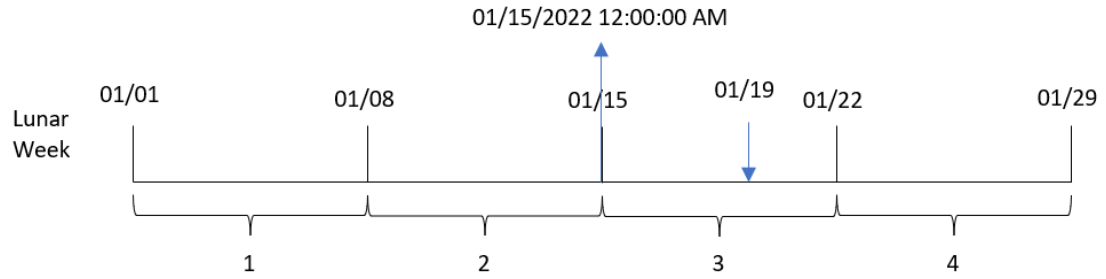
date	=lunarweekstart(date)	=timestamp(lunarweekstart(date))
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/15/2022	1/15/2022 12:00:00 AM
2/5/2022	02/05/2022	2/5/2022 12:00:00 AM
2/28/2022	02/26/2022	2/26/2022 12:00:00 AM
3/16/2022	03/12/2022	3/12/2022 12:00:00 AM
4/1/2022	03/26/2022	3/26/2022 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

date	=lunarweekstart(date)	=timestamp(lunarweekstart(date))
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

Miarę `start_of_week` tworzy się w obiekcie wykresu, używając funkcji `lunarweekstart()` i przekazując pole daty jako jej argument.

Funkcja `lunarweekstart()` identyfikuje, w którym tygodniu księżycowym przypada wartość daty, i zwraca znacznik czasu ostatniej milisekundy danego tygodnia.

Diagram funkcji `lunarweekstart()`, przykład obiektu wykresu



Transakcja 8189 miała miejsce 19 stycznia. Funkcja `lunarweekstart()` oblicza, że tydzień księżycowy zaczyna się 15 stycznia. W związku z tym wartość `start_of_week` dla tamtej transakcji jest pierwszą milisekundą tamtego dnia, czyli 15 stycznia, godz. 00:00:00.

Przykład 5 – scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw sald kredytów, który jest ładowany do tabeli o nazwie `Loans`.
- Dane zawierające identyfikatory kredytów, saldo na początku tygodnia i prostą stopę procentową naliczaną od każdego kredytu rocznie.

Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał według identyfikatora pożyczki bieżące odsetki naliczone od każdej pożyczki w bieżącym tygodniu.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę.
2. Dodaj następujące pola jako wymiary:
    - `loan_id`
    - `start_balance`
3. Następnie utwórz następującą miarę, aby obliczyć narosłe odsetki:
   `=start_balance*(rate*(today(1)-lunarweekstart(today(1)))/365)`
4. Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

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

Używając dzisiejszej daty jako jedynego argumentu, funkcja `lunarweekstart()` zwraca datę początkową bieżącego roku. Odejmując ten wynik od bieżącej daty, wyrażenie zwraca liczbę dni, które upłynęły do tej pory w tym tygodniu.

Wartość ta jest następnie mnożona przez stopę procentową i dzielona przez 365, aby uzyskać wysokość odsetek nagromadzonych do tego momentu tygodnia. Wynik jest następnie mnożony przez saldo początkowe pożyczki, aby zwrócić odsetki naliczone do tej pory w tym tygodniu.

makedate

Ta funkcja zwraca datę obliczoną na podstawie parametrów rok **YYYY**, miesiąc **MM** i dzień **DD**.

Składnia:

```
MakeDate(YYYY [ , MM [ , DD ] ])
```

Typ zwracanych danych: dual

Argumenty

Argument	Opis
YYYY	Rok jest liczbą całkowitą.
MM	Miesiąc jest liczbą całkowitą. Jeśli nie zostanie podany miesiąc, przyjmowana jest wartość 1 (styczeń).
DD	Dzień jest liczbą całkowitą. Jeśli nie zostanie podany dzień, przyjmowana jest wartość 1 (pierwszy).

Kiedy używać

Funkcja `makedate()` znajduje zastosowanie do generowania kalendarzy w skryptach do generowania danych. Ponadto można jej użyć, gdy pole daty nie jest bezpośrednio dostępne jako data, ale potrzebuje pewnych transformacji, aby pobrać rok, miesiąc i dzień.

W tych przykładach używany jest format daty MM/DD/YYYY. Format daty jest określony w instrukcji `SET DateFormat` u góry skryptu ładowania danych. Format zastosowany w przykładach można zmienić, aby dostosować go do konkretnych potrzeb.

Przykłady funkcji

Przykład	Wynik
`makedate(2012)`	Zwraca wartość 01/01/2012.
`makedate(12)`	Zwraca wartość 01/01/2012.
`makedate(2012,12)`	Zwraca wartość 12/01/2012.
`makedate(2012,2,14)`	Zwraca wartość 02/14/2012.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Przykład podstawowy

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2018, który jest ładowany do tabeli o nazwie „`Transactions`”.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie pola `transaction_date`, które zwraca datę w formacie MM/DD/RRRR.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `transaction_year`
- `transaction_month`
- `transaction_day`
- `transaction_date`

Tabela wynikowa

transaction_year	transaction_month	transaction_day	transaction_date
2018	08	30	08/30/2018
2018	09	07	09/07/2018
2018	09	16	09/16/2018
2018	09	22	09/22/2018
2018	09	23	09/23/2018

Pole `transaction_date` jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `makedate()` i przekazanie jej jako argumentów roku, miesiąca oraz dnia.

Następnie funkcja ta łączy i konwertuje te wartości na pole daty oraz zwraca wynik w formacie zmiennej systemowej `DateFormat`.

Przykład 2 – zmodyfikowana zmienna systemowa DateFormat

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola `transaction_date` w formacie DD/MM/RRRR bez modyfikacji zmiennej systemowej `DateFormat`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `transaction_year`
- `transaction_month`
- `transaction_day`
- `transaction_date`

Tabela wynikowa

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

W tym przypadku funkcja `makedate()` jest zagnieżdżona w funkcji `date()`. Drugi argument funkcji `date()` ustawia format wyników funkcji `makedate()` na wymagany DD/MM/RRRR.

Przykład 3 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2018, który jest ładowany do tabeli o nazwie „`Transactions`”.
- Daty transakcji przekazane w dwóch polach: `year` i `month`.

Utwórz miarę obiektu wykresu, `transaction_date`, która zwraca datę w formacie MM/DD/RRRR.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `year`
- `month`

Aby obliczyć `transaction_date`, utwórz następującą miarę:

```
=makedate(transaction_year,transaction_month)
```

Tabela wynikowa

transaction_year	transaction_month	transaction_date
2018	08	08/01/2018
2018	09	09/01/2018

Miara `transaction_date` jest tworzona w obiekcie wykresu za pomocą funkcji `makedate()`, której jako argumenty przekazano pola roku i miesiąca.

Następnie funkcja ta łączy te wartości, jak również zakładaną wartość dnia 01. Następnie te wartości zostają przekonwertowane na pole daty oraz zostaje zwrócony wynik w formacie zmiennej systemowej `DateFormat`.

Przykład 4 – Scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Utwórz zestaw danych kalendarza dla roku kalendarzowego 2022.

Skrypt ładowania

```
SET DateFormat='MM/DD/YYYY';

Calendar:
       load
                *
    where year(date)=2022;
load
       date(recno()+makedate(2021,12,31)) as date
AutoGenerate 400;
```

Wyniki

Tabela wynikowa

date
01/01/2022
01/02/2022
01/03/2022
01/04/2022
01/05/2022
01/06/2022
01/07/2022
01/08/2022
01/09/2022
01/10/2022
01/11/2022
01/12/2022
01/13/2022
01/14/2022
01/15/2022
01/16/2022
01/17/2022
01/18/2022
01/19/2022
01/20/2022
01/21/2022

date
01/22/2022
01/23/2022
01/24/2022
01/25/2022
Jeszcze 340 wierszy

Funkcja `makedate()` tworzy wartość daty 31 grudnia 2021 r. Funkcja `recno()` dostarcza numer bieżącego rekordu, który jest obecnie ładowany do tabeli (numeracja zaczyna się od 1). W efekcie pierwszy rekord ma datę 1 stycznia 2022 r. Każde kolejne wywołanie funkcji `recno()` będzie zwiększać tę datę o 1. To wyrażenie znajduje się w funkcji `date()`, która konwertuje wartość na datę. Ten proces zostaje wykonany 400 razy przez funkcję `autogenerate`. Na koniec, wykorzystując poprzednią instrukcję LOAD, można użyć warunku `where`, aby załadować tylko daty z roku 2022. Ten skrypt generuje kalendarz zawierający każdą datę z 2022 roku.

maketime

Ta funkcja zwraca godzinę obliczoną na podstawie parametrów godzina **hh**, minuta **mm** i sekunda **ss**.

Składnia:

```
MakeTime(hh [ , mm [ , ss ] ])
```

Typ zwracanych danych: dual

Argumenty

Argument	Opis
hh	Godzina jest liczbą całkowitą.
mm	Minuta jest liczbą całkowitą. Jeśli nie zostanie podana minuta, przyjmowana jest wartość 00.
ss	Sekunda jest liczbą całkowitą. Jeśli nie zostanie podana sekunda, przyjmowana jest wartość 00.

Kiedy używać

Funkcja `maketime()` znajduje zastosowanie do generowania pól czasu w skryptach do generowania danych. Czasami, gdy pole czasu jest określane na podstawie tekstu wejściowego, tej funkcji można użyć do skonstruowania wartości czasu z jej elementów.

W tych przykładach jest używany format czasu `h:mm:ss`. Format czasu jest określony w instrukcji `SET TimeFormat` na początku skryptu ładowania danych. Format zastosowany w przykładach można zmienić, aby dostosować go do konkretnych potrzeb.

Przykłady funkcji

Przykład	Wynik
`maketime(22)`	Zwraca wartość 22:00:00.
`maketime(22, 17)`	Zwraca wartość 22:17:00.
`maketime(22,17,52 )`	Zwraca wartość 22:17:52.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – funkcja maketime()

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji, który jest ładowany do tabeli o nazwie `Transactions`.
- Czasy transakcji określone przez trzy pola: `hours`, `minutes` i `seconds`.
- Utworzenie pola `transaction_time`, które zwraca czas w formacie zmiennej systemowej `TimeFormat`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `transaction_hour`
- `transaction_minute`
- `transaction_second`
- `transaction_time`

Tabela wynikowa

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

Pole `transaction_time` jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `maketime()` i przekazanie jej jako argumentów godziny, minuty i sekundy.

Następnie funkcja ta łączy i konwertuje te wartości na pole czasu oraz zwraca wynik w formacie czasu zmiennej systemowej `TimeFormat`.

Przykład 2 – funkcja time()

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola `transaction_time`, które umożliwi nam pokazanie wyników w 24-godzinnym formacie czasu bez modyfikacji zmiennej systemowej `TimeFormat`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `transaction_hour`
- `transaction_minute`
- `transaction_second`
- `transaction_time`

Tabela wynikowa

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

W tym przypadku funkcja `maketime()` jest zagnieżdżona w funkcji `time()`. Drugi argument funkcji `time()` ustawia format wyników funkcji `maketime()` na wymagany `h:mm:ss`.

Przykład 3 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji, który jest ładowany do tabeli o nazwie `Transactions`.
- Godziny transakcji określone przez dwa pola: `hours` i `minutes`.
- Utworzenie pola `transaction_time`, które zwraca czas w formacie zmiennej systemowej `TimeFormat`.

Utwórz miarę obiektu wykresu, `transaction_time`, która zwraca godzinę w formacie `h:mm:ss TT`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `transaction_hour`
- `transaction_minute`

Aby obliczyć `transaction_time`, należy utworzyć następującą miarę:

```
=maketime(transaction_hour,transaction_minute)
```

Tabela wynikowa

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

Miara `transaction_time` jest tworzona w obiekcie wykresu za pomocą funkcji `maketime()`, której jako argumenty przekazano pola godziny i minuty.

Następnie funkcja ta łączy te wartości oraz przyjmuje 00 jako liczbę sekund. Następnie te wartości zostają przekonwertowane na pole godziny oraz zostaje zwrócony wynik w formacie zmiennej systemowej `TimeFormat`.

Przykład 4 – Scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Utwórz zestaw danych kalendarza dla stycznia 2022 r., podzielony na ośmiogodzinne rosnące bloki.

Skrypt ładowania

```
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;
```

Wyniki

Tabela wynikowa

znacznik czasu
1/1/2022 12:00:00 AM
1/1/2022 8:00:00 AM
1/1/2022 4:00:00 PM
1/2/2022 12:00:00 AM
1/2/2022 8:00:00 AM
1/2/2022 4:00:00 PM
1/3/2022 12:00:00 AM
1/3/2022 8:00:00 AM
1/3/2022 4:00:00 PM
1/4/2022 12:00:00 AM
1/4/2022 8:00:00 AM
1/4/2022 4:00:00 PM
1/5/2022 12:00:00 AM
1/5/2022 8:00:00 AM
1/5/2022 4:00:00 PM
1/6/2022 12:00:00 AM
1/6/2022 8:00:00 AM
1/6/2022 4:00:00 PM
1/7/2022 12:00:00 AM
1/7/2022 8:00:00 AM
1/7/2022 4:00:00 PM
1/8/2022 12:00:00 AM
1/8/2022 8:00:00 AM
1/8/2022 4:00:00 PM
1/9/2022 12:00:00 AM
Jeszcze 68 wierszy

Początkowo funkcja `autogenerate` tworzy kalendarz zawierający wszystkie daty stycznia w tabeli o nazwie `tmpCalendar`.

Zostaje utworzona druga tabela, zawierająca trzy rekordy. Dla każdego rekordu wykonywane jest działanie `recno() - 1` (wartości 0, 1, 2) i wynik jest mnożony przez 8. W efekcie generowane są wartości 0, 8 i 16. Wartości te są używane jako parametr godziny w funkcji `maketime()`, z wartościami minuty i sekundy wynoszącymi 0. W efekcie tabela zawiera trzy pola godziny: 12:00:00 AM, 8:00:00 AM i 4:00:00 PM.

Ta tabela jest połączona z tabelą `tmpCalendar`. Ponieważ między tymi dwiema łączonymi tabelami nie ma pasujących pól, wiersze godziny zostają dodane do każdego wiersza daty. W efekcie każdy wiersz daty występuje teraz w trzech egzemplarzach z każdą wartością godziny.

Na koniec zostaje utworzona tabela Calendar z już załadowanej tabeli `tmpCalendar`. Pola daty i godziny zostają połączone i opakowane w funkcję `timestamp()` w celu utworzenia pola znacznika czasu.

Następnie tabela `tmpCalendar` zostaje usunięta.

makeweekdate

Ta funkcja zwraca datę obliczoną na podstawie parametrów rok **YYYY**, tydzień **WW** i dzień tygodnia **D**.

Składnia:

```
MakeWeekDate(YYYY [ , WW [ , D ] ])
```

Typ zwracanych danych: dual

Funkcja `makeweekdate()` jest dostępna zarówno jako skrypt, jak i funkcja wykresu. Oblicza ona datę na podstawie otrzymanych argumentów. Jeśli parametr dnia tygodnia zostanie pominięty, funkcja ta zwróci datę poniedziałku w tym tygodniu.

Funkcja `makeweekdate()` nie uwzględnia zmiennych systemowych `BrokenWeek`, `ReferenceDay` ani `FirstWeekDay`. Tydzień 1 zaczyna się w pierwszy poniedziałek stycznia. Na przykład, w 2022 r. tydzień pierwszy zaczyna się 3 stycznia.

Argumenty

Argument	Opis
YYYY	Rok jest liczbą całkowitą.
WW	Tydzień jest liczbą całkowitą. Wartość tygodnia może być dodatnia lub ujemna oraz może być większa niż 52, aby można było zwracać daty w różnych latach.
D	Dzień tygodnia jest liczbą całkowitą. Jeśli nie zostanie podany dzień tygodnia, przyjmowana jest wartość 0 (poniedziałek). Pozostałe dni tygodnia mają przypisane następujące wartości: wtorek – 1, środa – 2, czwartek – 3, piątek – 4, sobota – 5 oraz niedziela – 6.

Kiedy używać

Funkcja `makeweekdate()` może być używana w skryptach do generowania danych w celu wygenerowania listy dat lub utworzenia daty na podstawie podanych wartości roku, tygodnia i dnia tygodnia.

Przykłady funkcji

Przykład	Wynik
`makeweekdate(2014,6,6)`	zwraca wartość 02/09/2014
`makeweekdate(2014,6,1)`	zwraca wartość 02/04/2014
`makeweekdate(2014,6)`	zwraca 02/03/2014 (przyjmuje się dzień tygodnia 0)

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – uwzględnienie dnia

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający sumy sprzedaży w ujęciu tygodniowym za 2022 rok w tabeli o nazwie `Sales`.
- Daty transakcji określone przez trzy pola: `year`, `week` i `sales`.
- Poprzednie ładowanie, użyte do utworzenia miary, `end_of_week`, wykorzystuje funkcję `makeweekdate()`, aby zwrócić datę piątku w tamtym tygodniu w formacie MM/DD/RRRR.

Aby udowodnić, że zwrócona data to piątek, wyrażenie `end_of_week` zostało zapakowane w funkcję `weekday()`, pokazującą dzień tygodnia.

Skrypt ładowania

```
SET DateFormat='MM/DD/YYYY';

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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `transaction_year`
- `transaction_week`
- `end_of_week`
- `week_day`

Tabela wynikowa

transaction_year	transaction_week	end_of_week	week_day
2022	01	01/07/2022	Fri
2022	02	01/14/2022	Fri
2022	03	01/21/2022	Fri
2022	04	01/28/2022	Fri
2022	05	02/04/2022	Fri
2022	06	02/11/2022	Fri
2022	07	02/18/2022	Fri

Pole `end_of_week` jest tworzone w poprzedniej instrukcji load przy użyciu funkcji `makeweekdate()`. Pola `transaction_year` i `transaction_week` są przekazywane przez funkcję jako argumenty określające rok i tydzień. Argumentowi dnia przypisano wartość 4.

Następnie funkcja ta łączy i konwertuje te wartości na pole daty oraz zwraca wynik w formacie zmiennej systemowej `DateFormat`.

Funkcja `makeweekdate()` wraz z argumentami także jest zapakowana w funkcję `weekday()`, która zwraca pole `week_day`. Jak widać w powyższej tabeli, pole `week_day` pokazuje, że te daty rzeczywiście wypadają w piątek.

Przykład 2 – dzień nieuwzględniony

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający sumy sprzedaży w ujęciu tygodniowym za 2022 rok w tabeli o nazwie `Sales`.
- Daty transakcji określone przez trzy pola: `year`, `week` i `sales`.
- Poprzednie ładowanie, które zostało użyte do utworzenia miary, `first_day_of_week`, używa funkcji `makeweekdate()`. Zostanie zwrócona data poniedziałku w tym tygodniu w formacie MM/DD/RRRR.

Aby udowodnić, że zwrócona data wypada w poniedziałek, wyrażenie `first_day_of_week` zostało zapakowane w funkcję `weekday()`, pokazującą dzień tygodnia.

Skrypt ładowania

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;

Transactions:
    Load
        *,
        makeweekdate(transaction_year, transaction_week) as first_day_of_week,
        day(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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- transaction_year
- transaction_week
- first_day_of_week
- week_day

Tabela wynikowa

transaction_year	transaction_week	first_day_of_week	week_day
2022	01	01/03/2022	Pon
2022	02	01/10/2022	Pon
2022	03	01/17/2022	Pon
2022	04	01/24/2022	Pon
2022	05	01/31/2022	Pon
2022	06	02/07/2022	Pon
2022	07	02/14/2022	Pon

Pole first_day_of_week jest tworzone w poprzedniej instrukcji load przy użyciu funkcji makeweekdate(). Parametry transaction_year i transaction_week zostały przekazane jako argumenty funkcji, a parametr dnia pozostawiono pusty.

Następnie funkcja ta łączy i konwertuje te wartości na pole daty oraz zwraca wynik w formacie zmiennej systemowej DateFormat.

Funkcja makeweekdate() wraz z argumentami także została zapakowana w funkcję weekday(), która zwraca pole week_day. Jak widać w powyższej tabeli, pole week_day pokazuje, że te daty rzeczywiście wypadają w poniedziałek (mimo że zmienna FirstWeekDay ustawia niedzielę jako pierwszy dzień tygodnia), ponieważ parametr dnia w funkcji makeweekdate() pozostawiono pusty.

Przykład 3 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający sumy sprzedaży w ujęciu tygodniowym za 2022 rok w tabeli o nazwie Sales.
- Daty transakcji określone przez trzy pola: year, week i sales.

W tym przykładzie obiekt wykresu zostanie użyty w celu utworzenia miary równoważnej obliczeniom end_of_week z pierwszego przykładu. Miara ta użyje funkcji makeweekdate(), aby zwrócić datę piątku w tym tygodniu w formacie MM/DD/RRRR.

Aby udowodnić, że zwrócona data wypada w piątek, zostaje utworzona druga miara, która zwraca dzień tygodnia.

Skrypt ładowania

```
SET DateFormat='MM/DD/YYYY';

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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:
   - `transaction_year`
   - `transaction_week`
2. Aby wykonać obliczenia równoważne tym, które wykonuje pole `end_of_week` w pierwszym przykładzie, utwórz następującą miarę:
   `=makeweekdate(transaction_year,transaction_week,4)`
3. Aby obliczyć dzień tygodnia każdej transakcji, utwórz następującą miarę:
   `=weekday(makeweekdate(transaction_year,transaction_week,4))`

Tabela wynikowa

transaction_ year	transaction_ week	=makeweekdate (transaction_ year,transaction_week,4)	=weekday(makeweekdate (transaction_year,transaction_ week,4))
2022	01	01/07/2022	Fri
2022	02	01/14/2022	Fri
2022	03	01/21/2022	Fri
2022	04	01/28/2022	Fri
2022	05	02/04/2022	Fri
2022	06	02/11/2022	Fri
2022	07	02/18/2022	Fri

Pole równoważne `end_of_week` zostaje utworzone w obiekcie wykresu jako miara przy użyciu funkcji `makeweekdate()`. Pola `transaction_year` i `transaction_week` zostały przekazane jako argumenty roku i tygodnia. Argumentowi dnia przypisano wartość 4.

Następnie funkcja ta łączy i konwertuje te wartości na pole daty oraz zwraca wynik w formacie zmiennej systemowej `DateFormat`.

Funkcja `makeweekdate()` wraz z argumentami także została zapakowana w funkcję `weekday()`, która zwraca obliczenia równoważne obliczeniom pola `week_day` z pierwszego przykładu. Jak widać w powyższej tabeli, ostatnia kolumna z prawej pokazuje, że te daty rzeczywiście wypadają w piątek.

Przykład 4 – Scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

W tym przykładzie utwórz listę dat zawierających wszystkie piątki 2022 roku.

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania

```
SET DateFormat='MM/DD/YYYY';

Calendar:
    load
           *,
        weekday(date) as weekday
    where year(date)=2022;
load
    makeweekdate(2022,recno()-2,4) as date
AutoGenerate 60;
```

Wyniki

Tabela wynikowa

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

date	weekday
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
Jeszcze 27 wierszy	

Funkcja `makeweekdate()` znajduje wszystkie piątki w 2022 roku. Przypisanie parametrowi tygodnia wartości -2 daje pewność, że żadna data nie zostanie przeoczona. Poprzednia instrukcja LOAD dla jasności tworzy dodatkowe pole `weekday`, aby pokazać, że każda wartość `date` jest piątkiem.

minute

Ta funkcja zwraca liczbę całkowitą reprezentującą minuty, gdy ułamek **expression** jest interpretowany jako czas zgodnie ze standardową interpretacją liczb.

Składnia:

```
minute(expression)
```

Typ zwracanych danych: integer

Kiedy używać

Funkcja `minute()` jest przydatna, gdy chcesz porównać agregacje według minut. Za pomocą tej funkcji można na przykład sprawdzić rozkład liczb aktywności w ujęciu minutowym.

Te wymiary można utworzyć w skrypcie ładowania za pomocą funkcji tworzenia pola w tabeli kalendarza głównego. Ewentualnie można ich użyć bezpośrednio na wykresie jako wymiaru wyliczanego.

Przykłady funkcji

Przykład	Wynik
`minute ( '09:14:36' )`	Zwraca wartość 14
`minute ( '0.5555' )`	zwraca 19 (ponieważ 0,5555 = 13:19:55)

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – zmienna (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zbiór danych zawierający transakcje wg znacznika czasu, który jest ładowany do tabeli o nazwie `Transactions`.
- Została użyta domyślna zmienna systemowa `TimeStamp` (`M/D/YYYY h:mm:ss[.fff] TT`).
- Utworzenie pola `minute` obliczającego czas zawierania transakcji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `timestamp`
- `minute`

Tabela wynikowa

znacznik czasu	minuta
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

Wartości w polu `minute` są tworzone za pomocą funkcji `minute()` i przez przekazanie `timestamp` jako wyrażenia w poprzedniej instrukcji ładowania.

Przykład 2 – obiekt wykresu (wykres)

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Została użyta domyślna zmienna systemowa `TimeStamp` (`M/D/YYYY h:mm:ss[.fff] TT`).

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Wartości `minute` są obliczane przez miarę w obiekcie wykresu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `timestamp`.

Utwórz następującą miarę:

```
=minute(timestamp)
```

Tabela wynikowa

znacznik czasu	minuta
2022-01-01 22:10:22	10
2022-01-02 08:35:54	35

znacznik czasu	minuta
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

Wartości dla `minute` są tworzone za pomocą funkcji `minute()` i przez przekazanie `timestamp` jako wyrażenia w mierze dla obiektu wykresu.

Przykład 3 – Scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw znaczników czasu wygenerowanych w celu reprezentowania przejść przez bramkę biletową.
- Informacje z każdym `timestamp` i odpowiadającym mu `id`, które są załadowane do tabeli o nazwie `Ticket_Barrier_Tracker`.
- Została użyta domyślna zmienna systemowa `TimeStamp` (`M/D/YYYY h:mm:ss[.fff] TT`).

Użytkownik chciałby, aby obiekt wykresu pokazywał liczbę przejść przez bramkę biletową w ujęciu minutowym.

Skrypt ładowania

```
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;
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę.
2. Utwórz wymiar wyliczany, używając następującego wyrażenia:
   `=minute(timestamp)`
3. Dodaj następującą miarę agregacji, aby obliczyć sumę przejść:
   `=count(id)`
4. Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

minute(timestamp)	=count(id)
0	174
1	171
2	175
3	165
4	188
5	176
6	158
7	187
8	178
9	178
10	197
11	161
12	166
13	184
14	159
15	161
16	152

minute(timestamp)	=count(id)
17	160
18	176
19	164
20	170
21	170
22	142
23	145
24	155
Jeszcze 35 wierszy	

month

Ta funkcja zwraca wartość podwójną z nazwą miesiąca, jak to zostało określone w zmiennej środowiskowej **MonthNames**, oraz liczbę całkowitą z przedziału 1-12. Miesiąc jest obliczany na podstawie interpretacji daty z wyrażenia zgodnie ze standardową interpretacją liczb.

Funkcja ta zwraca nazwę miesiąca w formacie zmiennej systemowej `MonthName` dla określonej daty. Jest powszechnie używana do tworzenia pola dnia jako wymiaru w kalendarzu głównym.

Składnia:

```
month(expression)
```

Typ zwracanych danych: integer

Przykłady funkcji

Przykład	Wynik
`month( 2012-10-12 )`	Zwraca wartość Oct
`month( 35648 )`	Zwraca wartość Aug, ponieważ 35648 = 1997-08-06

Przykład 1 – zbiór danych DateFormat (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Zbiór danych dat o nazwie `Master_Calendar`. Zmienna systemowa `DateFormat` jest ustawiona na DD/MM/RRRR.
- Poprzednie ładowanie, które tworzy dodatkowe pole, o nazwie`month_name`, używające funkcji `month()`.
- Dodatkowe pole, o nazwie `long_date`, używające funkcji `date()`, aby wyrazić pełną datę.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `long_date`
- `month_name`

Tabela wynikowa

data	long_date	month_name
03/01/2022	03 stycznia 2022	Jan
03/02/2022	03 lutego 2022	Feb
03/03/2022	03 marca 2022	Mar
03/04/2022	03 kwietnia 2022	Apr
03/05/2022	03-May- 2022	May

data	long_date	month_name
03/06/2022	03 czerwca 2022	Jun
03/07/2022	03 lipca 2022	Jul
03/08/2022	03 sierpnia 2022	Aug
03/09/2022	03 września 2022	Sep
03/10/2022	03 października 2022	Oct
03/11/2022	03 listopada 2022	Nov

Nazwa miesiąca jest prawidłowo obliczana przez funkcję `month()` w skrypcie.

Przykład 2 – daty ANSI (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Zbiór danych dat o nazwie `Master_Calendar`. Zmienn The DateFormat system variable DD/MM/YYYY is used. Natomiast daty w zbiorze danych są w standardowym formacie daty ANSI.
- Poprzednie ładowanie, które tworzy dodatkowe pole, o nazwie`month_name`, za pomocą funkcji month().
- Dodatkowe pole, o nazwie`long_date`, używające funkcji date() w celu wyrażenia pełnej daty.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `long_date`
- `month_name`

Tabela wynikowa

data	long_date	month_name
03/11/2022	11 marca 2022	11
03/12/2022	12 marca 2022	12
03/13/2022	13 marca 2022	13
03/14/2022	14 marca 2022	14
03/15/2022	15 marca 2022	15
03/16/2022	16 marca 2022	16
03/17/2022	17 marca 2022	17
03/18/2022	18 marca 2022	18
03/19/2022	19 marca 2022	19
03/20/2022	20 marca 2022	20
03/21/2022	21 marca 2022	21

Nazwa miesiąca jest prawidłowo obliczana przez funkcję `month()` w skrypcie.

Przykład 3 – niesformatowane daty (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Zbiór danych dat o nazwie `Master_Calendar`. Używana jest zmienna systemowa `DateFormat` DD/MM/RRRR.
- Poprzednie ładowanie, które tworzy dodatkowe pole, o nazwie`month_name`, używające funkcji `month()`.

- Pierwotna niesformatowana data o nazwie `unformatted_date`.
- Dodatkowe pole, o nazwie `long_date`, używające funkcji `date()`, aby wyrazić pełną datę.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `unformatted_date`
- `long_date`
- `month_name`

Tabela wynikowa

unformatted_date	long_date	month_name
44868	03 stycznia 2022	Jan
44898	03 lutego 2022	Feb
44928	03 marca 2022	Mar
44958	03 kwietnia 2022	Apr
44988	03-May- 2022	May
45018	03 czerwca 2022	Jun

unformatted_date	long_date	month_name
45048	03 lipca 2022	Jul
45078	03 sierpnia 2022	Aug
45008	03 września 2022	Sep
45038	03 października 2022	Oct
45068	03 listopada 2022	Nov

Nazwa miesiąca jest prawidłowo obliczana przez funkcję `month()` w skrypcie.

Przykład 4 – Obliczanie miesiąca wygaśnięcia

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i dodaj skrypt ładowania do nowej karty poniżej.

Skrypt ładowania zawiera:

- Zbiór danych zamówień złożonych w marcu o nazwie `Subscriptions`. Tabela zawiera trzy pola:
  - id
  - order_date
  - amount

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `order_date`.

Aby obliczyć miesiąc, w którym wygaśnie zamówienie, należy utworzyć następującą miarę: `=month(order_date+180)`.

Tabela wynikowa

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	Jan
03/09/2022	Mar
03/10/2022	Apr
03/11/2022	May

Funkcja `month()` prawidłowo stwierdza, że zamówienie złożone 11 marca wygaśnie w lipcu.

monthend

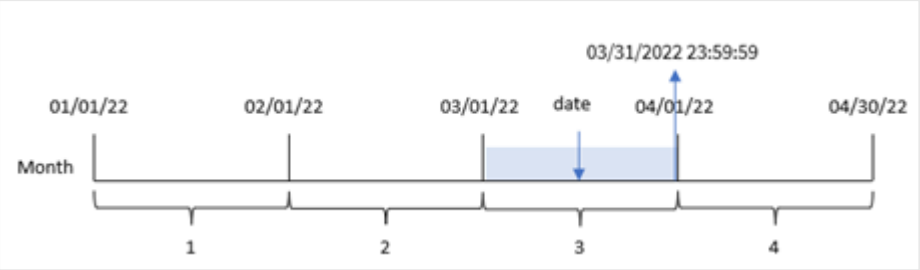
Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu ostatniej milisekundy ostatniego dnia miesiąca zawierającego wartość `date`. Domyślnym formatem wyjściowym będzie format `DateFormat` skonfigurowany w skrypcie.

Składnia:

```
MonthEnd(date[, period_no])
```

Innymi słowy, funkcja `monthend()` określa, na który miesiąc przypada data. Następnie zwraca znacznik czasu w formacie daty dla ostatniej milisekundy tego miesiąca.

Schemat funkcji monthend.



Kiedy używać

Funkcja `monthend()` jest powszechnie używana jako część wyrażenia, gdy użytkownik chce, by w obliczeniach użyto ułamka miesiąca, który jeszcze nie nastąpił. Na przykład, jeśli chcesz obliczyć łączne odsetki, które nie zostały jeszcze naliczone w ciągu miesiąca.

Typ zwracanych danych: dual

Argumenty

Argument	Opis
date	Data lub znacznik czasu do oszacowania.
period_no	**period_no** jest liczbą całkowitą, która, jeśli ma wartość 0 lub jest pominięta, oznacza miesiąc, który zawiera **date**. Wartości ujemne parametru **period_no** oznaczają miesiące poprzednie, a wartości dodatnie – miesiące następne.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
`monthend('02/19/2012')`	Zwraca wartość 02/29/2012 23:59:59.
`monthend('02/19/2001', -1)`	Zwraca wartość 01/31/2001 23:59:59.

Przykład 1 – Przykład podstawowy

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Pole daty, które jest przekazywane w formacie `MM/DD/YYYY` zmiennej systemowej `DateFormat`.
- Poprzedzająca instrukcja ładowania zawiera:
    - funkcję `monthend()`, która jest ustawiona jako pole `end_of_month`.
    - funkcję `timestamp`, która jest ustawiona jako pole `end_of_month_timestamp`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `end_of_month`
- `end_of_month_timestamp`

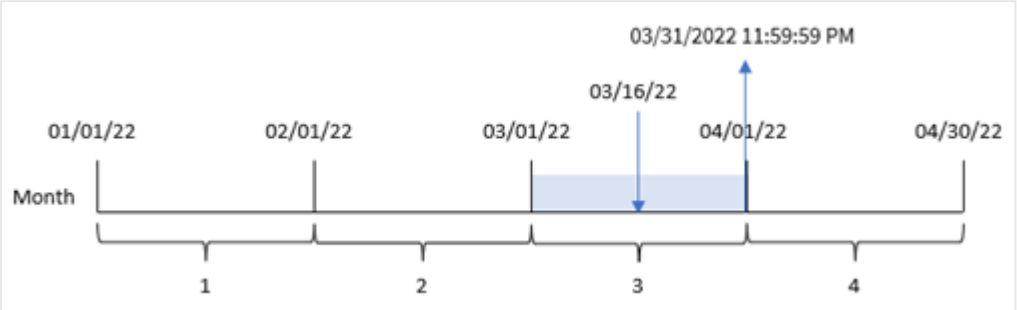
Tabela wynikowa

id	date	end_of_month	end_of_month_timestamp
8188	1/7/2022	01/31/2022	1/31/2022 11:59:59 PM
8189	1/19/2022	01/31/2022	1/31/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8194	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8195	5/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	07/31/2022	7/31/2022 11:59:59 PM
8199	7/22/2022	07/31/2022	7/31/2022 11:59:59 PM
8200	7/23/2022	07/31/2022	7/31/2022 11:59:59 PM
8201	7/27/2022	07/31/2022	7/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
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

Pole „`end_of_month`” jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `monthend()` i przekazanie pola daty jako jej argumentu.

Funkcja `monthend()` identyfikuje, w którym miesiącu przypada wartość daty, i zwraca znacznik czasu ostatniej milisekundy danego miesiąca.

Diagram funkcji `monthend` z wybranym marcem.



Transakcja 8192 miała miejsce 16 marca. Funkcja `monthend()` zwraca ostatnią milisekundę tego miesiąca, czyli 31 marca o godz 23:59:59 (11:59:59 PM).

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

W tym przykładzie zadaniem jest utworzenie pola `previous_month_end` zwracającego znacznik czasu końca miesiąca przed transakcją.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `previous_month_end`
- `previous_month_end_timestamp`

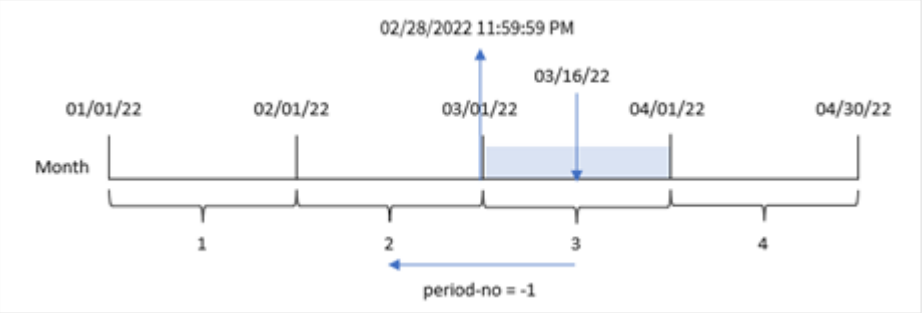
Tabela wynikowa

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

id	date	previous_month_end	previous_month_end_timestamp
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

Funkcja `monthend()` najpierw identyfikuje miesiąc, w którym miały miejsce transakcje, ponieważ jako argument przesunięcia `period_no` przekazano wartość `-1`. Następnie przesuwa zakres o miesiąc wstecz i identyfikuje ostatnią milisekundę tego miesiąca.

Diagram funkcji `monthend` ze zmienną period_no.



Transakcja 8192 miała miejsce 16 marca. Funkcja `monthend()` stwierdza, że miesiąc poprzedzający miesiąc transakcji to luty. Następnie zwraca ostatnią milisekundę tego miesiąca – 28 lutego o godz. 23:59:59.

Przykład 3 – przykład z wykresem

Skrypt ładowania i wyrażenie wykresu

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

W tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Zadanie polega na utworzeniu obliczeń zwracających znacznik czasu końca miesiąca, w którym zawarto transakcje, jako miary na wykresie aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `id`

Aby obliczyć koniec miesiąca, w którym zawarto transakcję, utwórz następujące miary:

- `=monthend(date)`
- `=timestamp(monthend(date))`

Tabela wynikowa

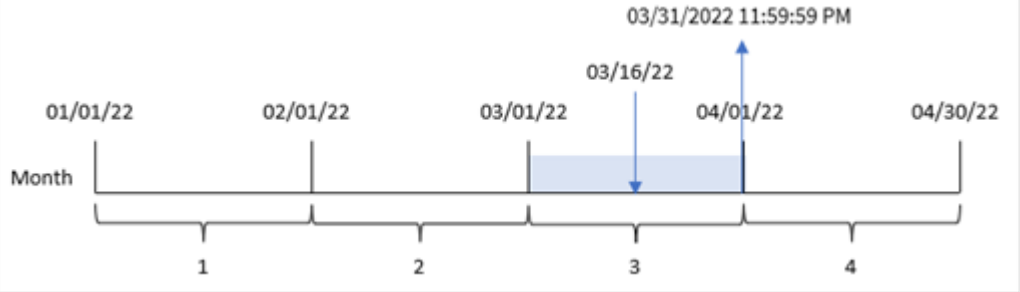
id	date	=monthend(date)	=timestamp(monthend(date))
8188	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8189	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM
8190	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8191	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8192	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8193	8/19/2022	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

id	date	=monthend(date)	=timestamp(monthend(date))
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

Miarę „`end_of_month`" tworzy się na wykresie, używając funkcji `monthend()` i przekazując pole daty jako jej argument.

Funkcja `monthend()` identyfikuje, w którym miesiącu przypada wartość daty, i zwraca znacznik czasu ostatniej milisekundy danego miesiąca.

Diagram funkcji `monthend` ze zmienną period_no.



Transakcja 8192 miała miejsce 16 marca. Funkcja `monthend()` zwraca ostatnią milisekundę tego miesiąca, czyli 31 marca o godz 23:59:59 (11:59:59 PM).

Przykład 4 – Scenariusz

Skrypt ładowania i wyniki

Przegląd

W tym przykładzie zestaw danych jest ładowany do tabeli o nazwie `Employee_Expenses`. Tabela zawiera następujące pola:

- Identyfikatory pracowników
- Imiona i nazwiska pracownika

- Średnie dzienne rozliczenia wydatków każdego pracownika.

Użytkownik końcowy chciałby, aby wykres wyświetlał, według identyfikatora pracownika oraz imienia i nazwiska pracownika, szacowane roszczenia z tytułu wydatków do poniesienia przez pozostałą część miesiąca.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `employee_id`
- `employee_name`

Aby obliczyć narosłe odsetki, należy utworzyć następującą miarę:

```
=floor(monthend(today(1),0)-today(1))*avg_daily_claim
```

> *ta miera jest dynamiczna i zwracać różne wyniki do tabeli w zależności od daty załadowania danych.*

Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

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

Funkcja `monthend()` zwraca datę końcową bieżącego miesiąca na podstawie aktualnej daty przekazanej jako jej jedyny argument. Wyrażenie zwraca liczbę dni pozostałych w tym miesiącu przez odjęcie bieżącej daty od daty zakończenia miesiąca.

Wartość ta jest następnie mnożona przez średnie dzienne roszczenie z tytułu wydatków przez każdego pracownika, aby obliczyć szacunkową wartość roszczeń, które każdy pracownik złoży w pozostałej części miesiąca.

monthname

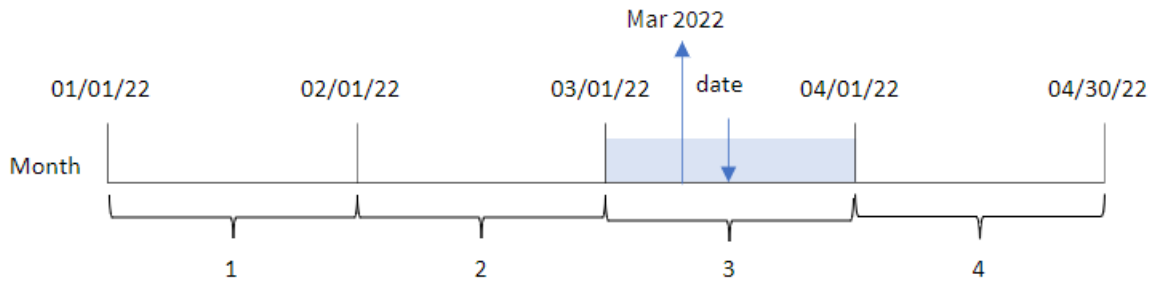
Ta funkcja zwraca wartość pokazującą miesiąc (sformatowany zgodnie ze stosowaną w skryptach zmienną **MonthNames**) oraz rok z bazową wartością liczbową odpowiadającą znacznikowi czasu pierwszej milisekundy pierwszego dnia tego miesiąca.

Składnia:

```
MonthName(date[, period_no])
```

Typ zwracanych danych: dual

Diagram funkcji monthname



Argumenty

Argument	Opis
date	Data lub znacznik czasu do oszacowania.
period_no	**period_no** jest liczbą całkowitą, która, jeśli ma wartość 0 lub jest pominięta, oznacza miesiąc, który zawiera **date**. Wartości ujemne parametru **period_no** oznaczają miesiące poprzednie, a wartości dodatnie – miesiące następne.

Przykłady funkcji

Przykład	Wynik
`monthname('10/19/2013')`	Zwraca wartość Oct 2013
`monthname('10/19/2013', -1)`	Zwraca wartość Sep 2013

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Przykład podstawowy

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „`Transactions`”.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie pola `transaction_month` zwracającego miesiąc, w którym miały miejsce transakcje.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `transaction_month`

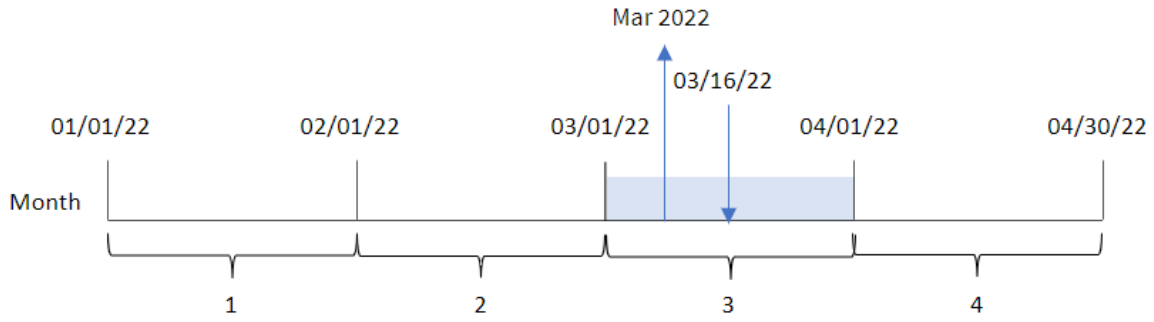
Tabela wynikowa

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

date	transaction_month
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

Pole `transaction_month` jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `monthname()` i przekazanie pola `date` jako argumentu funkcji.

Diagram funkcji monthname, przykład podstawowy



Funkcja `monthname()` identyfikuje, że transakcja 8192 miała miejsce w marcu 2022 r., i zwraca tę wartość przy użyciu zmiennej systemowej `MonthNames`.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam wbudowany zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola `transaction_previous_month` zwracającego znacznik czasu końca miesiąca przed transakcją.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `transaction_previous_month`

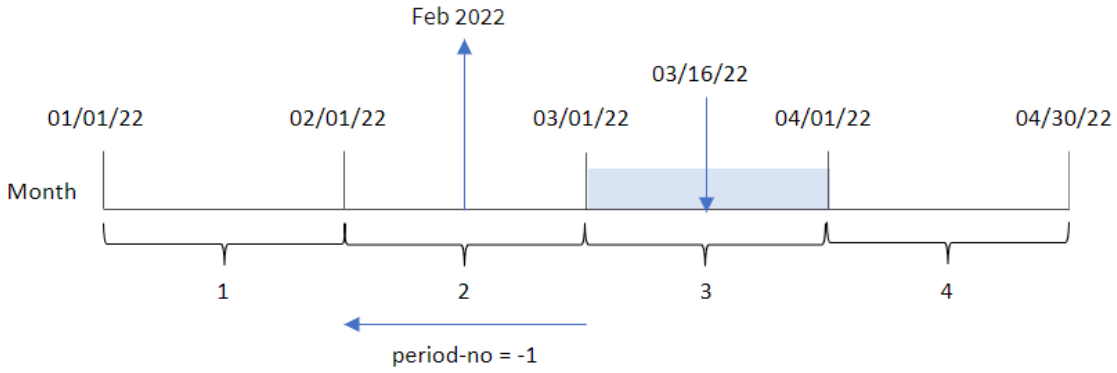
Tabela wynikowa

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

date	transaction_previous_month
5/16/2022	Apr 2022
6/15/2022	May 2022
6/26/2022	May 2022
7/9/2022	Jun 2022
7/22/2022	Jun 2022
7/23/2022	Jun 2022
7/27/2022	Jun 2022
8/2/2022	Jul 2022
8/8/2022	Jul 2022
8/19/2022	Jul 2022
9/26/2022	Aug 2022
10/14/2022	Sep 2022
10/29/2022	Sep 2022

W tym przypadku, ponieważ wartości `period_no` -1 użyto jako argumentu przesunięcia w funkcji `monthname()`, funkcja najpierw identyfikuje miesiąc, w którym odbywają się transakcje. Następnie powoduje przesunięcie o miesiąc wcześniej i zwraca nazwę miesiąca oraz rok.

Diagram funkcji monthname, przykład period_no



Transakcja 8192 miała miejsce 16 marca. Funkcja `monthname()` identyfikuje, że miesiącem poprzedzającym transakcję był luty, i zwraca miesiąc w formacie zmiennej systemowej `MonthNames` wraz z rokiem 2022.

Przykład 3 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam wbudowany zestaw danych i scenariusz co w pierwszym przykładzie. Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenie zwracające znacznik czasu końca miesiąca, kiedy wystąpiły transakcje, jest tworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:`date`.

Utwórz następującą miarę:

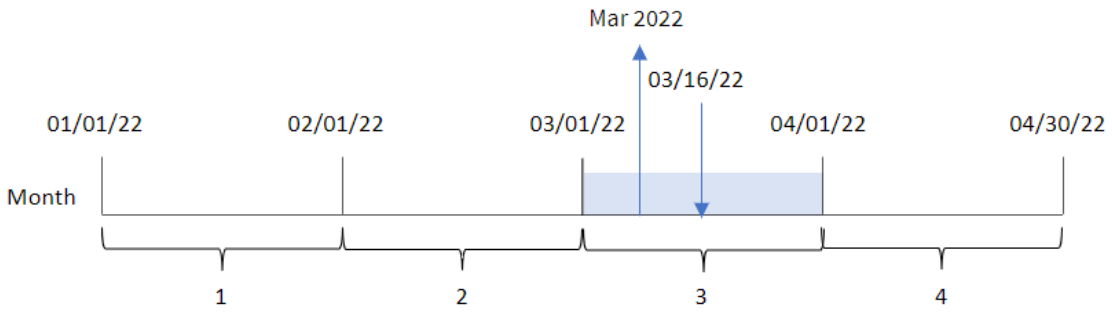
```
=monthname(date)
```

Tabela wynikowa

date	=monthname(date)
1/7/2022	Jan 2022
1/19/2022	Jan 2022
2/5/2022	Feb 2022
2/28/2022	Feb 2022
3/16/2022	Mar 2022
4/1/2022	Apr 2022
5/7/2022	May 2022
5/16/2022	May 2022
6/15/2022	Jun 2022
6/26/2022	Jun 2022
7/9/2022	Jul 2022
7/22/2022	Jul 2022
7/23/2022	Jul 2022
7/27/2022	Jul 2022
8/2/2022	Aug 2022
8/8/2022	Aug 2022
8/19/2022	Aug 2022
9/26/2022	Sep 2022
10/14/2022	Oct 2022
10/29/2022	Oct 2022

Miarę „`month_name`" tworzy się w obiekcie wykresu, używając funkcji `monthname()` i przekazując pole daty `date` jako jej argument.

Diagram funkcji monthname, przykład obiektu wykresu



Funkcja `monthname()` identyfikuje, że transakcja 8192 miała miejsce w marcu 2022 r., i zwraca tę wartość przy użyciu zmiennej systemowej `MonthNames`.

monthsend

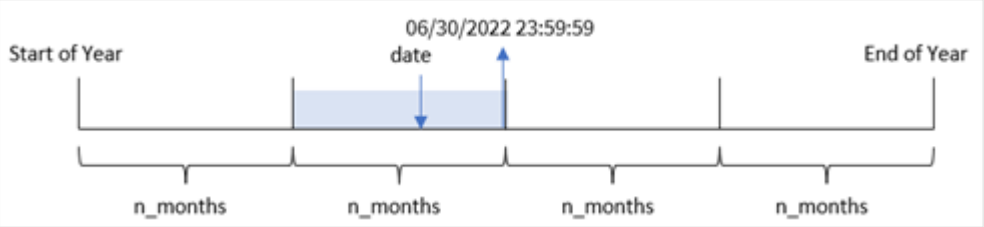
Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu ostatniej milisekundy okresu miesiąca, dwóch miesięcy, kwartału, czterech miesięcy lub półrocza, który zawiera datę bazową. Można także wyszukać znacznik czasu końca okresu poprzedniego lub następnego. Domyślnym formatem wyjściowym będzie format `DateFormat` skonfigurowany w skrypcie.

Składnia:

```
MonthsEnd(n_months, date[, period_no [, first_month_of_year]])
```

Typ zwracanych danych: wartość podwójna

Schemat funkcji `monthsend`.



Argumenty

Argument	Opis
n_months	Liczba miesięcy określająca okres. Wartość całkowita lub wyrażenie, którego wynikiem jest jedna z następujących wartości całkowitych: 1 (równoważnik funkcji inmonth()), 2 (dwa miesiące), 3 (równoważnik funkcji inquarter()), 4 (cztery miesiące) lub 6 (pół roku).
date	Data lub znacznik czasu do oszacowania.
period_no	Okres może być przesunięty o wartość **period_no** – liczbę całkowitą lub wyrażenie, którego wynikiem jest liczba całkowita, gdzie wartość 0 wskazuje dzień zawierający wartość **base_date**. Wartości ujemne parametru **period_no** oznaczają okresy poprzednie, a wartości dodatnie – okresy następne.
first_month_of_year	Jeśli użytkownik zamierza korzystać z lat (obrotowych), które nie zaczynają się w styczniu, powinien wskazać wartość od 2 do 12 jako parametr **first_month_of_year**.

Funkcja `monthsend()` dzieli rok na segmenty na podstawie przekazanego argumentu `n_months`. Następnie sprawdza, do którego segmentu należy każda z przekazanych dat, i zwraca ostatnią milisekundę, w formacie daty, tego segmentu. Funkcja może zwrócić końcowy znacznik czasu z poprzednich lub następnych segmentów, jak również może przedefiniować pierwszy miesiąc roku.

Następujące segmenty roku są dostępne w funkcji jako argumenty `n_month`:

Argumenty funkcji `n_month`

Okres	Liczba miesięcy
miesiąc	1
dwa miesiące	2
kwartał	3
cztery miesiące	4
pół roku	6

Kiedy używać

Funkcja `monthsend()` jest zwykle jako część wyrażenia, gdy użytkownik chce, by w obliczeniach użyto ułamka miesiąca, który upłynął do tej pory. Użytkownik może, przy użyciu zmiennej, wybrać dowolny okres. Na przykład, funkcja `monthsend()` może dostarczyć zmienną wejściową umożliwiającą użytkownikowi obliczenie sumy odsetek jeszcze nie zapłaconych w miesiącu, kwartale lub półroczu.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
`monthsend(4, '07/19/2013')`	Zwraca wartość 08/31/2013.
`monthsend(4, '10/19/2013', -1)`	Zwraca wartość 08/31/2013.
`monthsend(4, '10/19/2013', 0, 2)`	Zwraca wartość 01/31/2014. Ponieważ początkiem roku staje się miesiąc 2.

Przykład 1 – Przykład podstawowy

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Pole daty przekazane w formacie zmiennej systemowej `DateFormat` (`(MM/DD/YYYY)`).
- Poprzedzająca instrukcja ładowania zawiera:
    - funkcję `monthsend`, która jest ustawiona jako pole `bi_monthly_end`. Grupuje transakcje w dwumiesięczne segmenty.
    - Funkcja `timestamp`, która zwraca początkowy znacznik czasu segmentu dla każdej transakcji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

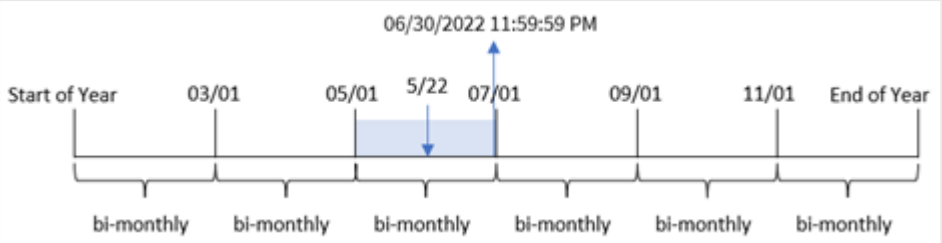
- id
- date
- bi_monthly_end
- bi_monthly_end_timestamp

Tabela wynikowa

id	date	bi_monthly_end	bi_monthly_end_timestamp
8188	1/7/2022	02/28/2022	2/28/2022 11:59:59 PM
8189	1/19/2022	02/28/2022	2/28/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	04/30/2022	4/30/2022 11:59:59 PM
8193	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	10/31/2022	10/31/2022 11:59:59 PM
8206	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8207	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

Pole `bi_monthly_end` jest tworzone w poprzedniej instrukcji load przy użyciu funkcji `monthsend()`. Pierwszy podany argument - `2` - dzieli rok na segmenty dwumiesięczne. Pierwszy podany argument określa, które pole jest oceniane.

Schemat funkcji `monthsend` z segmentami dwumiesięcznymi.



Transakcja 8195 ma miejsce 22 maja. Funkcja `monthsend()` początkowo dzieli rok na segmenty dwumiesięczne. Transakcja 8195 należy do segmentu od 1 maja do 30 czerwca. W wyniku funkcja zwraca ostatnią milisekundę tego segmentu – 30.06.2022, godz. 23:59:59.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

W tym przykładzie zadaniem jest utworzenie pola `prev_bi_monthly_end` zwracającego pierwszą milisekundę dwumiesięcznego segmentu przed zawarciem transakcji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `prev_bi_monthly_end`
- `prev_bi_monthly_end_timestamp`

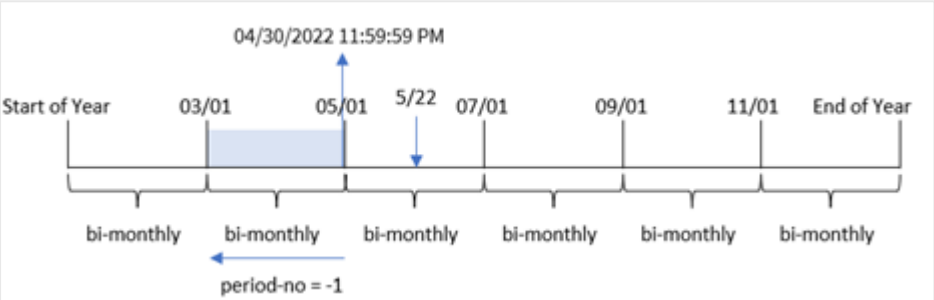
Tabela wynikowa

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

id	date	prev_bi_monthly_end	prev_bi_monthly_end_timestamp
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

Jeśli po podzieleniu roku na segmenty dwumiesięczne do funkcji `monthsend()` przekażemy wartość `-1` jako argument `period_no`, funkcja ta zwróci ostatnią milisekundę dwumiesięcznego segmentu poprzedzającego ten, w którym zawarto transakcję.

Diagram funkcji `monthsend` zwracającej poprzedni dwumiesięczny segment.



Transakcja 8195 należy do segmentu od maja do czerwca. W efekcie poprzedni dwumiesięczny segment obejmował okres od 1 marca do 30 kwietnia, więc funkcja zwraca ostatnią milisekundę właśnie tego segmentu, czyli odpowiadającą dacie 30.04.2022, godz. 23:59:59.

Przykład 3 – first_month_of_year

Skrypt ładowania i wyniki

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

W tym przykładzie przyjęto, że kwiecień ma być pierwszym miesiącem roku podatkowego.

Utwórz pole `bi_monthly_end`, które grupuje transakcje w segmenty dwumiesięczne i zwraca znacznik czasu ostatniej milisekundy segmentu dla każdej transakcji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `bi_monthly_end`
- `bi_monthly_end_timestamp`

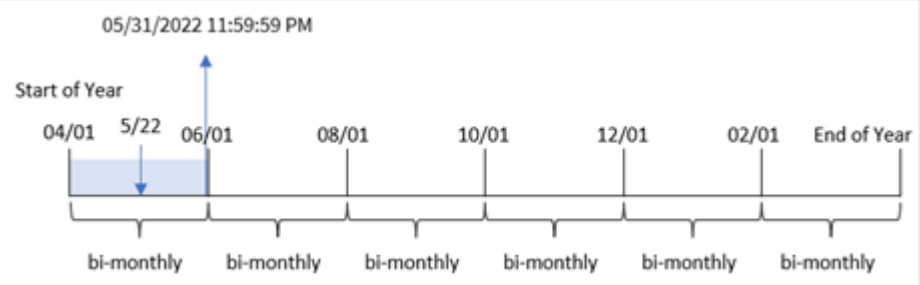
Tabela wynikowa

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

id	date	bi_monthly_end	bi_monthly_end_timestamp
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

Kiedy jako argument `first_month_of_year` funkcji `monthsend()` zostaje przekazana wartość `4`, funkcja ta rozpoczyna rok 1 kwietnia, a następnie dzieli go na segmenty dwumiesięczne: kwi-maj, cze-lip, sie-wrz, paź-lis, gru-sty, lut-mar.

Diagram funkcji `monthsend` z kwietniem ustawionym jako pierwszy miesiąc roku.



Transakcja 8195 miała miejsce 22 maja i należy do segmentu od 1 kwietnia do 31 maja. W wyniku funkcja zwraca ostatnią milisekundę tego segmentu – 31.05.2022, godz. 23:59:59.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie. Jednak w tym przykładzie zestaw danych pozostaje bez zmian i jest ładowany do aplikacji.

W tym przykładzie zadanie polega na utworzeniu obliczeń grupujących transakcje w segmenty dwumiesięczne i zwracających znacznik czasu ostatniej milisekundy segmentu dla każdej transakcji jako miary w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

```
date
```

Aby pobrać znacznik czasu ostatniej milisekundy dwumiesięcznego segmentu, w którym zawarto transakcję, utwórz następujące miary:

- `=monthsEnd(2,date)`
- `=timestamp(monthsend(2,date))`

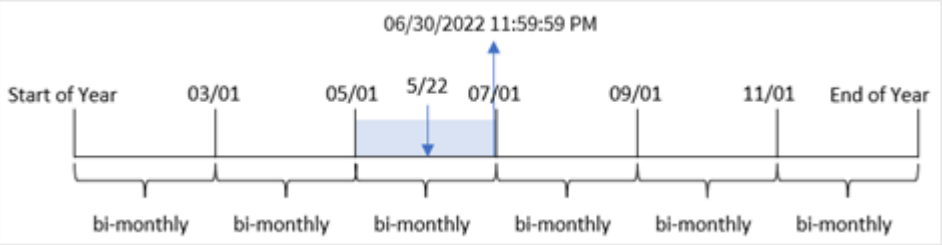
Tabela wynikowa

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

id	date	=monthsend(2,date)	=timestamp(monthsend(2,date))
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

Pole `bi_monthly_end` jest tworzone jako miara w obiekcie wykresu przy użyciu funkcji `monthsend()`. Pierwszy podany argument ma wartość `2`, która dzieli rok na segmenty dwumiesięczne. Pierwszy podany argument określa, które pole jest oceniane.

Schemat funkcji `monthsend` z segmentami dwumiesięcznymi.



Transakcja 8195 ma miejsce 22 maja. Funkcja `monthsend()` początkowo dzieli rok na segmenty dwumiesięczne. Transakcja 8195 należy do segmentu od 1 maja do 30 czerwca. W wyniku funkcja zwraca pierwszą milisekundę tego segmentu - 30.06.2022, godz. 23:59:59.

Przykład 5 – scenariusz

Skrypt ładowania i wyniki

Przegląd

Otwórz edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

W tym przykładzie zestaw danych jest ładowany do tabeli o nazwie `Employee_Expenses`. Tabela zawiera następujące pola:

- Identyfikatory pracowników
- Imiona i nazwiska pracownika
- Średnie dzienne rozliczenia wydatków każdego pracownika.

Użytkownik końcowy chciałby, aby wykres wyświetlał, według identyfikatora pracownika oraz imienia i nazwiska pracownika, szacowane roszczenia z tytułu wydatków do poniesienia przez pozostałą część wybranego okresu. Rok obrotowy rozpoczyna się w styczniu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz nowy arkusz.

Na początku skryptu ładowania została utworzona zmienna `vPeriod`, która jest powiązana z kontrolką wejściową zmiennej.

Wykonaj następujące czynności:

1. W panelu zasobów kliknij opcję **Obiekty niestandardowe**.
2. Wybierz **pakiet Qlik Dashboard** i utwórz obiekt **wprowadzania zmiennych**.
3. Wprowadź tytuł obiektu wykresu.
4. W polu **Zmienna** wybierz **vPeriod** jako nazwę i ustaw obiekt tak, aby był wyświetlany jako **lista rozwijana**.
5. W sekcji **Wartości** kliknij **Wartości** dynamiczne. Wprowadź:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`

Utwórz nową tabelę i te pola jako wymiary:

- employee_id
- employee_name

Aby obliczyć narosłe odsetki, należy utworzyć następującą miarę:

```
=floor(monthsend($(vPeriod),today(1))-today(1))*avg_daily_claim
```

ta miera jest dynamiczna i zwracać różne wyniki do tabeli w zależności od daty załadowania danych.

Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

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

Funkcja `monthsend()` przyjmuje dane od użytkownika jako pierwszy argument i dzisiejszą datę jako drugi argument. Zwraca datę końcową dla okresu wybranego przez użytkownika. Następnie wyrażenie zwraca liczbę dni pozostałych w wybranym okresie przez odjęcie bieżącej daty od daty zakończenia.

Wartość ta jest następnie mnożona przez średnie dzienne roszczenie z tytułu wydatków przez każdego pracownika, aby obliczyć szacunkową wartość roszczeń, które każdy pracownik złoży w pozostałych dniach tego okresu.

monthsname

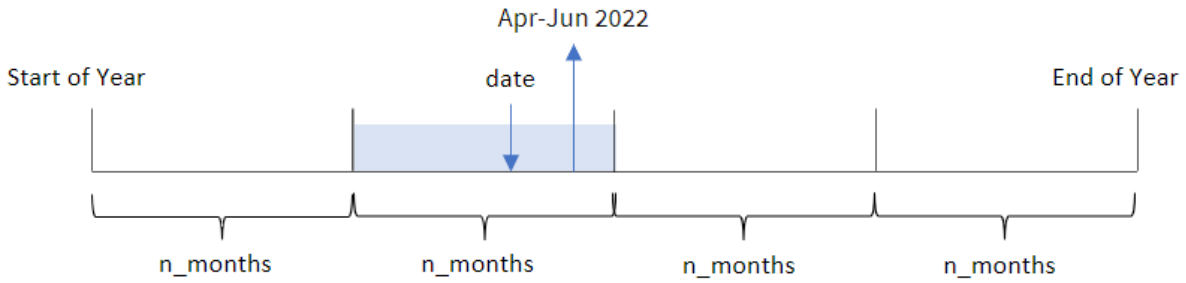
Ta funkcja zwraca wartość reprezentującą zakres miesięcy w okresie (sformatowany zgodnie ze stosowaną w skryptach zmienną **MonthNames**), a także rok. Bazowa wartość liczbowa odpowiada znacznikowi czasu pierwszej milisekundy okresu miesiąca, dwóch miesięcy, kwartału, czterech miesięcy lub półrocza, który zawiera datę bazową.

Składnia:

```
MonthsName(n_months, date[, period_no[, first_month_of_year]])
```

Typ zwracanych danych: dual

Diagram funkcji monthsname



Funkcja `monthsname()` dzieli rok na segmenty na podstawie podanego argumentu `n_months`. Następnie ocenia segment, do którego należy każda podana `date`, i zwraca nazwę miesiąca początkowego i końcowego tego segmentu, a także rok. Funkcja daje również możliwość zwracania tych granic z poprzednich lub następnych segmentów, a także redefiniowania, który jest pierwszym miesiącem roku.

Następujące segmenty roku są dostępne w funkcji jako argumenty `n_month`:

Możliwe argumenty `n_month`

Okresy	Liczba miesięcy
miesiąc	1
dwa miesiące	2
kwartał	3
cztery miesiące	4
pół roku	6

Argumenty

Argument	Opis
n_months	Liczba miesięcy określająca okres. Wartość całkowita lub wyrażenie, którego wynikiem jest jedna z następujących wartości całkowitych: 1 (równoważnik funkcji inmonth()), 2 (dwa miesiące), 3 (równoważnik funkcji inquarter()), 4 (cztery miesiące) lub 6 (pół roku).
date	Data lub znacznik czasu do oszacowania.
period_no	Okres może być przesunięty o wartość **period_no** – liczbę całkowitą lub wyrażenie, którego wynikiem jest liczba całkowita, gdzie wartość 0 wskazuje dzień zawierający wartość **base_date**. Wartości ujemne parametru **period_no** oznaczają okresy poprzednie, a wartości dodatnie – okresy następne.
first_ month_of_ year	Jeśli użytkownik zamierza korzystać z lat (obrotowych), które nie zaczynają się w styczniu, powinien wskazać wartość od 2 do 12 jako parametr **first_month_of_year**.

Kiedy używać

Funkcja `monthsname()` jest przydatna, gdy chcesz udostępnić użytkownikowi funkcjonalność porównywania agregacji według wybranego samodzielnie okresu. Na przykład możesz podać zmienną wejściową, aby użytkownik mógł zobaczyć całkowitą sprzedaż produktów według miesiąca, kwartału lub półrocza.

Te wymiary można utworzyć w skrypcie ładowania, dodając funkcję jako pole w tabeli kalendarza głównego, lub tworząc wymiar bezpośrednio na wykresie jako wymiar obliczany.

Przykłady funkcji

Przykład	Wynik
`monthsname(4, '10/19/2013')`	Zwraca „Sep-Dec 2013”. W tym i pozostałych przykładach instrukcja **SET Monthnames** jest ustawiona na wartość Jan;Feb;Mar itd.
`monthsname(4, '10/19/2013', -1)`	Zwraca wartość „May-Aug 2013”.
`monthsname(4, '10/19/2013', 0, 2)`	Zwraca „Oct-Jan 2014”, ponieważ rok jest określony jako rozpoczynający się w miesiącu 2. W związku z tym okres czterech miesięcy kończy się w pierwszym miesiącu roku następnego.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Przykład podstawowy

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „Transactions”.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie pola `bi_monthly_range`, które grupuje transakcje w segmenty dwumiesięczne i zwraca nazwy granic tego segmentu dla każdej transakcji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

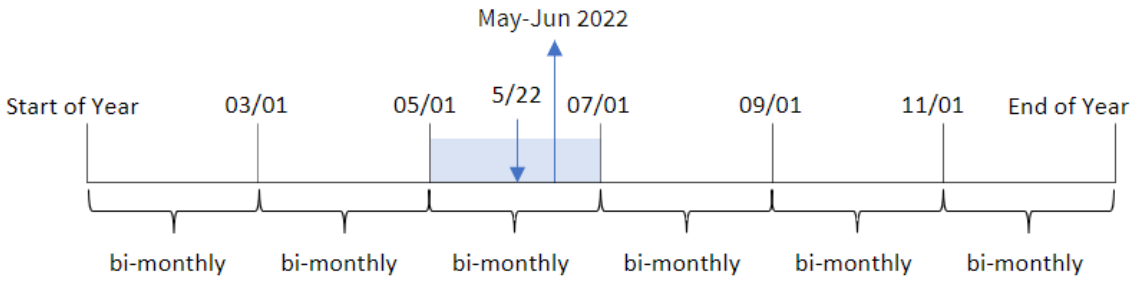
- `date`
- `bi_monthly_range`

Tabela wynikowa

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

Pole `bi_monthly_range` jest tworzone w poprzedniej instrukcji load przy użyciu funkcji `monthsname()`. Pierwszy podany argument to 2, dzielący rok na segmenty dwumiesięczne. Pierwszy podany argument określa, które pole jest oceniane.

Diagram funkcji monthsname, przykład podstawowy



Transakcja 8195 ma miejsce 22 maja. Funkcja `monthsname()` początkowo dzieli rok na segmenty dwumiesięczne. Transakcja 8195 należy do segmentu od 1 maja do 30 czerwca. Dlatego funkcja zwraca te miesiące w formacie zmiennej systemowej `MonthNames`, a także rok, May-Jun 2022.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam wbudowany zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola `prev_bi_monthly_range`, które grupuje transakcje w segmenty dwumiesięczne i zwraca nazwy granic poprzedniego segmentu dla każdej transakcji.

Dodaj tutaj swój inny tekst, w razie potrzeby, z listami itp.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `prev_bi_monthly_range`

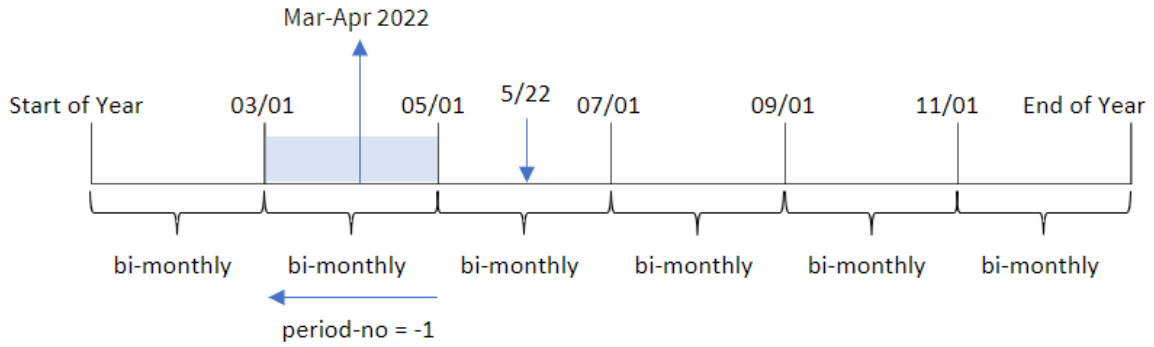
Tabela wynikowa

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

W tym przykładzie wartość -1 jest używana jako argument `period_no` w funkcji `monthsname()`. Po początkowym podzieleniu roku na segmenty dwumiesięczne funkcja zwraca granice poprzedniego segmentu, gdy ma miejsce transakcja.

Diagram funkcji monthsname, przykład period_no



Transakcja 8195 należy do segmentu od maja do czerwca. W związku z tym poprzedni segment dwumiesięczny miał miejsce między 1 marca a 30 kwietnia, więc funkcja zwraca Mar-Apr 2022.

Przykład 3 – first_month_of_year

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam wbudowany zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie innego pola, `bi_monthly_range`, które grupuje transakcje w segmenty dwumiesięczne i zwraca granice segmentu dla każdej transakcji.

Jednak w tym przykładzie musimy również ustawić kwiecień jako pierwszy miesiąc roku obrotowego.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `bi_monthly_range`

Tabela wynikowa

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

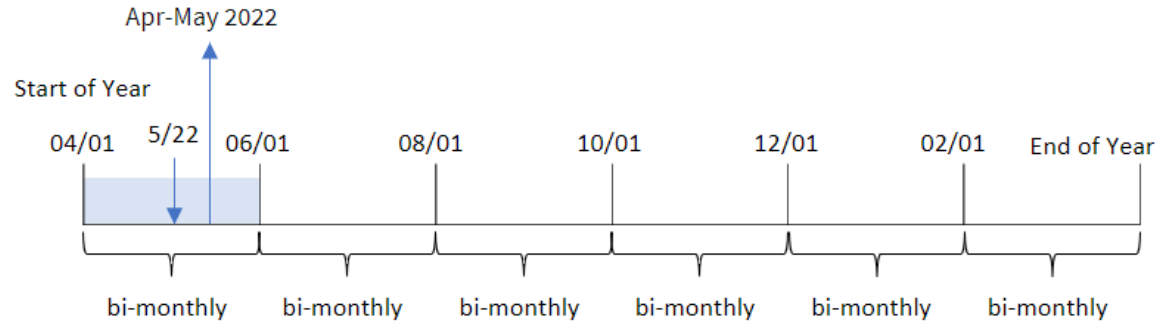
date	bi_monthly_range
4/16/2022	Apr-May 2022
5/1/2022	Apr-May 2022
5/7/2022	Apr-May 2022
5/22/2022	Apr-May 2022
6/15/2022	Jun-Jul 2022
6/26/2022	Jun-Jul 2022
7/9/2022	Jun-Jul 2022
7/22/2022	Jun-Jul 2022
7/23/2022	Jun-Jul 2022
7/27/2022	Jun-Jul 2022
8/2/2022	Aug-Sep 2022
8/8/2022	Aug-Sep 2022
8/19/2022	Aug-Sep 2022
9/26/2022	Aug-Sep 2022
10/14/2022	Oct-Nov 2022
10/29/2022	Oct-Nov 2022

Kiedy wartość 4 jest używana jako argument `first_month_of_year` w funkcji `monthsname()`, funkcja rozpoczyna rok 1 kwietnia, a następnie dzieli rok na segmenty dwumiesięczne: Apr-May,Jun-Jul,Aug-Sep,Oct-Nov,Dec-Jan,Feb-Mar.

Tekst akapitu dotyczący wyników.

Transakcja 8195 miała miejsce 22 maja i należy do segmentu od 1 kwietnia do 31 maja. W związku z tym funkcja zwraca Apr-May 2022.

Diagram funkcji monthsname, przykład first_month_of_year



Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam wbudowany zestaw danych i scenariusz co w pierwszym przykładzie. Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenie, które grupuje transakcje w segmenty dwumiesięczne i zwraca granice segmentu dla każdej transakcji zostaje utworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:`date`.

Utwórz następującą miarę:

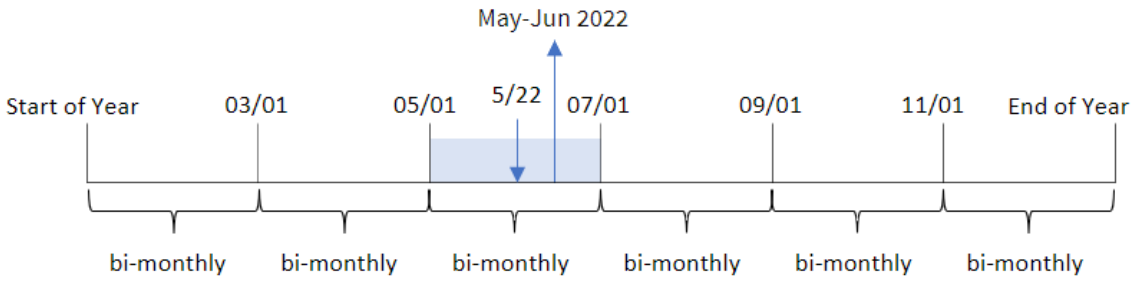
```
=monthsname(2,date)
```

Tabela wynikowa

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

Pole `bi_monthly_range` jest tworzone jako miara w obiekcie wykresu przy użyciu funkcji `monthsname()`. Pierwszy podany argument to 2, dzielący rok na segmenty dwumiesięczne. Pierwszy podany argument określa, które pole jest oceniane.

Diagram funkcji monthsname, przykład obiektu wykresu



Transakcja 8195 ma miejsce 22 maja. Funkcja `monthsname()` początkowo dzieli rok na segmenty dwumiesięczne. Transakcja 8195 należy do segmentu od 1 maja do 30 czerwca. Dlatego funkcja zwraca te miesiące w formacie zmiennej systemowej `MonthNames`, a także rok, May-Jun 2022.

Przykład 5 – scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający transakcje za rok 2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).

Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał całkowitą sprzedaż w wybranym przez siebie okresie. Można to osiągnąć nawet wtedy, gdy ten wymiar nie jest dostępny w modelu danych, używając funkcji `monthsname()` jako wymiaru obliczanego, który jest dynamicznie modyfikowany przy użyciu elementu sterującego do wprowadzania zmiennych.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz.

Na początku skryptu ładowania została utworzona zmienna (`vPeriod`), która zostanie powiązana z kontrolką wejściową zmiennej. Następnie skonfiguruj zmienną jako obiekt niestandardowy w arkuszu.

Wykonaj następujące czynności:

1. W panelu zasobów kliknij opcję **Obiekty niestandardowe**.
2. Wybierz **Qlik Dashboard bundle** i utwórz obiekt **Wprowadzanie zmiennych**.
3. Wprowadź tytuł obiektu wykresu.
4. W polu **Zmienna** wybierz **vPeriod** jako nazwę i ustaw obiekt tak, aby był wyświetlany jako **lista rozwijana**.
5. W obszarze **Wartości** skonfiguruj obiekt tak, aby używał wartości dynamicznych. Wprowadź:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'`

Następnie utwórz tabelę wyników.

Wykonaj następujące czynności:

1. Utwórz nową tabelę i dodaj następujący wymiar wyliczany:
   `=monthsname($(vPeriod),date)`
2. Dodaj tę miarę, aby obliczyć całkowitą sprzedaż:
   `=sum(amount)`
3. Ustaw **Formatowanie liczb** miary na **Waluta**. Kliknij przycisk ✔ **Koniec edycji**. Możesz teraz modyfikować dane pokazane w tabeli, dostosowując segment czasu w obiekcie zmiennych.

Tak będzie wyglądać tabela wyników po wybraniu opcji `tertial` :

Tabela wynikowa

monthsname($(vPeriod),date)	=sum(amount)
Jan-Mar 2022	253.89
May-Aug 2022	713.58
Sep-Dec 2022	248.12

monthsstart

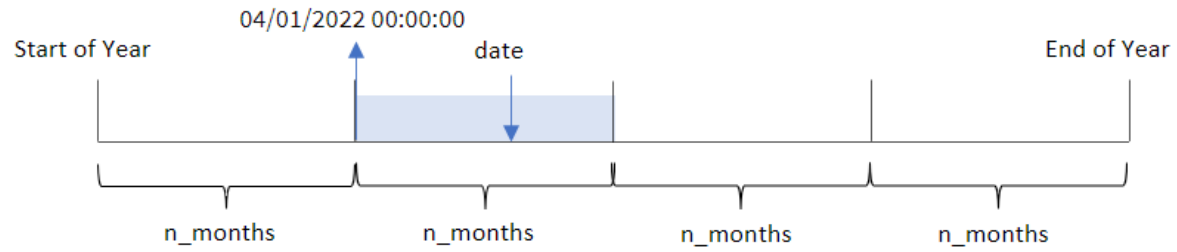
Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu pierwszej milisekundy okresu miesiąca, dwóch miesięcy, kwartału, czterech miesięcy lub półrocza, który zawiera datę bazową. Można także wyszukać znacznik czasu dla okresu poprzedniego lub następnego.Domyślnym formatem wyjściowym będzie format **DateFormat** skonfigurowany w skrypcie.

Składnia:

```
MonthsStart(n_months, date[, period_no [, first_month_of_year]])
```

Typ zwracanych danych: dual

Schemat funkcji `monthsstart()`



Funkcja `monthsstart()` dzieli rok na segmenty na podstawie przekazanego argumentu `n_months`. Następnie sprawdza, do którego segmentu należy każda z przekazanych dat, i zwraca pierwszą milisekundę tego segmentu w formacie daty. Funkcja daje również możliwość zwracania początkowego znacznika czasu z poprzednich lub następnych segmentów, a także redefiniowania pierwszego miesiąca roku.

Następujące segmenty roku są dostępne w funkcji jako argumenty `n_month`:

Możliwe argumenty `n_month`

Okresy	Liczba miesięcy
miesiąc	1
dwa miesiące	2
kwartał	3

Okresy	Liczba miesięcy
cztery miesiące	4
pół roku	6

Argumenty

Argument	Opis
n_months	Liczba miesięcy określająca okres. Wartość całkowita lub wyrażenie, którego wynikiem jest jedna z następujących wartości całkowitych: 1 (równoważnik funkcji inmonth()), 2 (dwa miesiące), 3 (równoważnik funkcji inquarter()), 4 (cztery miesiące) lub 6 (pół roku).
date	Data lub znacznik czasu do oszacowania.
period_no	Okres może być przesunięty o wartość **period_no** – liczbę całkowitą lub wyrażenie, którego wynikiem jest liczba całkowita, gdzie wartość 0 wskazuje dzień zawierający wartość **base_date**. Wartości ujemne parametru **period_no** oznaczają okresy poprzednie, a wartości dodatnie – okresy następne.
first_month_of_year	Jeśli użytkownik zamierza korzystać z lat (obrotowych), które nie zaczynają się w styczniu, powinien wskazać wartość od 2 do 12 jako parametr **first_month_of_year**.

Kiedy używać

Funkcja `monthsstart()` jest powszechnie używana jako część wyrażenia, gdy użytkownik chce, by w obliczeniach użyto ułamka okresu, który jeszcze nie nastąpił. Można to wykorzystać na przykład do dostarczenia zmiennej wejściowej umożliwiającej użytkownikowi obliczenie sumy odsetek nagromadzonych w miesiącu, kwartale lub półroczu.

Przykłady funkcji

Przykład	Wynik
`monthsstart(4, '10/19/2013')`	Zwraca wartość 09/01/2013.
`monthsstart(4, '10/19/2013, -1)`	Zwraca wartość 05/01/2013.
`monthsstart(4, '10/19/2013', 0, 2 )`	Zwraca 10/01/2013, ponieważ początek roku został zmieniony na miesiąc 2.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „Transactions”.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie pola `bi_monthly_start`, które grupuje transakcje w segmenty dwumiesięczne i zwraca początkowy znacznik czasu segmentu dla każdej transakcji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

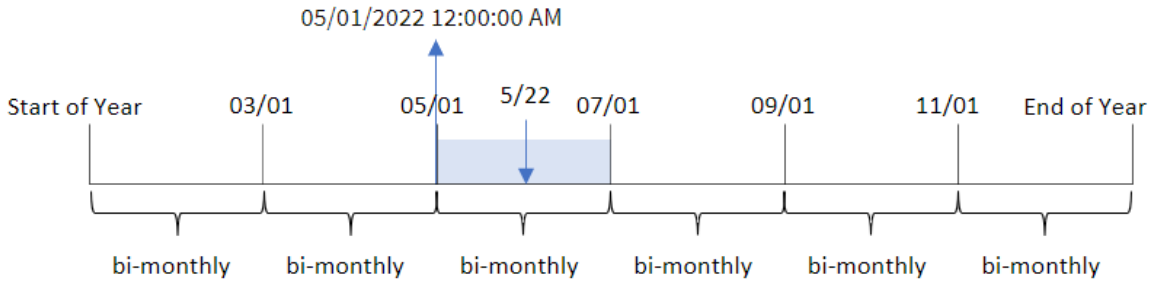
- `date`
- `bi_monthly_start`
- `bi_monthly_start_timestamp`

Tabela wynikowa

date	bi_monthly_start	bi_monthly_start_timestamp
2/19/2022	01/01/2022	1/1/2022 12:00:00 AM
3/7/2022	03/01/2022	3/1/2022 12:00:00 AM
3/30/2022	03/01/2022	3/1/2022 12:00:00 AM
4/5/2022	03/01/2022	3/1/2022 12:00:00 AM
4/16/2022	03/01/2022	3/1/2022 12:00:00 AM
5/1/2022	05/01/2022	5/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/22/2022	05/01/2022	5/1/2022 12:00:00 AM
6/15/2022	05/01/2022	5/1/2022 12:00:00 AM
6/26/2022	05/01/2022	5/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
10/14/2022	09/01/2022	9/1/2022 12:00:00 AM
10/29/2022	09/01/2022	9/1/2022 12:00:00 AM

Pole `bi_monthly_start` jest tworzone w poprzedniej instrukcji load przy użyciu funkcji `monthsstart()`. Pierwszy podany argument to 2, dzielący rok na segmenty dwumiesięczne. Pierwszy podany argument określa, które pole jest oceniane.

Diagram funkcji `monthsstart()`, przykład bez dodatkowych argumentów



Transakcja 8195 ma miejsce 22 maja. Funkcja `monthsstart()` początkowo dzieli rok na segmenty dwumiesięczne. Transakcja 8195 należy do segmentu od 1 maja do 30 czerwca. W związku z tym funkcja zwraca pierwszą milisekundę tego segmentu, 1 maja 2022 r., godz 00:00:00.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola, `prev_bi_monthly_start`, które zwraca pierwszą milisekundę dwumiesięcznego segmentu przed zawarciem transakcji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `prev_bi_monthly_start`
- `prev_bi_monthly_start_timestamp`

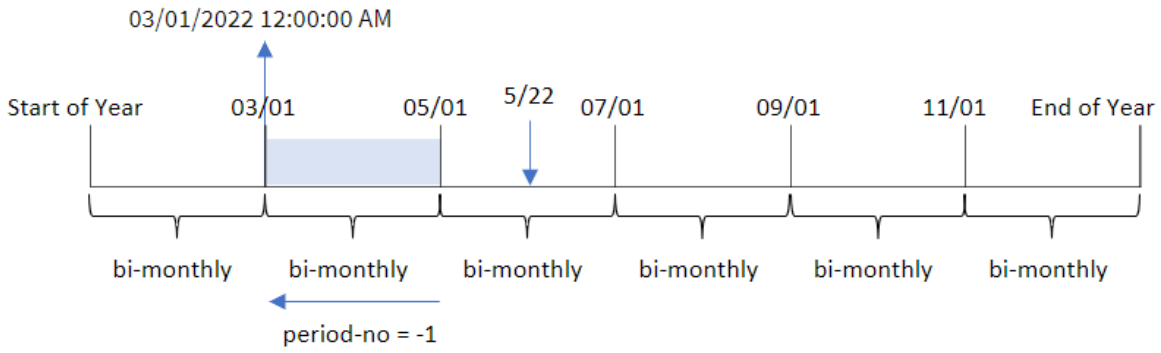
Tabela wynikowa

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

date	prev_bi_monthly_start	prev_bi_monthly_start_timestamp
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

Jeśli po podzieleniu roku na segmenty dwumiesięczne do funkcji `monthsstart()` jako argument `period_no` przekażemy wartość -1, funkcja ta zwróci ostatnią milisekundę dwumiesięcznego segmentu poprzedzającego ten, w którym zawarto transakcję.

Diagram funkcji `monthsstart()`, przykład z argumentem period_no



Transakcja 8195 należy do segmentu od maja do czerwca. W związku z tym poprzedni dwumiesięczny segment obejmował okres od 1 marca do 30 kwietnia, więc funkcja zwraca pierwszą milisekundę tego segmentu – 1 marca 2022 r., godz 00:00:00.

Przykład 3 – first_month_of_year

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola `bi_monthly_start`, które grupuje transakcje w segmenty dwumiesięczne i zwraca początkowy znacznik czasu zestawu dla każdej transakcji.

Jednak w tym przykładzie musimy również ustawić kwiecień jako pierwszy miesiąc roku obrotowego.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

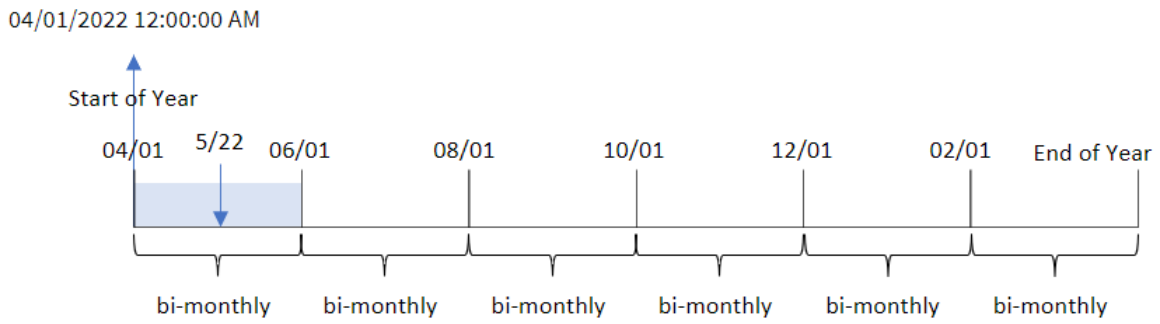
- `date`
- `bi_monthly_start`
- `bi_monthly_start_timestamp`

Tabela wynikowa

date	bi_monthly_start	bi_monthly_start_timestamp
2/19/2022	02/01/2022	2/1/2022 12:00:00 AM
3/7/2022	02/01/2022	2/1/2022 12:00:00 AM
3/30/2022	02/01/2022	2/1/2022 12:00:00 AM
4/5/2022	04/01/2022	4/1/2022 12:00:00 AM
4/16/2022	04/01/2022	4/1/2022 12:00:00 AM
5/1/2022	04/01/2022	4/1/2022 12:00:00 AM
5/7/2022	04/01/2022	4/1/2022 12:00:00 AM
5/22/2022	04/01/2022	4/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	06/01/2022	6/1/2022 12:00:00 AM
7/9/2022	06/01/2022	6/1/2022 12:00:00 AM
7/22/2022	06/01/2022	6/1/2022 12:00:00 AM
7/23/2022	06/01/2022	6/1/2022 12:00:00 AM
7/27/2022	06/01/2022	6/1/2022 12:00:00 AM
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	8/1/2022 12:00:00 AM
8/19/2022	08/01/2022	8/1/2022 12:00:00 AM
9/26/2022	08/01/2022	8/1/2022 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM

Kiedy wartość 4 jest używana jako argument `first_month_of_year` w funkcji `monthsstart()`, funkcja rozpoczyna rok 1 kwietnia, a następnie dzieli rok na segmenty dwumiesięczne: Apr-May,Jun-Jul,Aug-Sep,Oct-Nov,Dec-Jan,Feb-Mar.

Diagram funkcji `monthsstart()`, przykład z użyciem argumentu first_month_of_year



Transakcja 8195 miała miejsce 22 maja i należy do segmentu od 1 kwietnia do 31 maja. W związku z tym funkcja zwraca pierwszą milisekundę tego segmentu, 1 kwietnia 2022 r., godz 00:00:00.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Zostają utworzone obliczenia grupujące transakcje w segmenty dwumiesięczne i zwracające początkowy znacznik czasu zestawu dla każdej transakcji jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Utwórz następujące miary:

```
=monthsstart(2,date)
```

```
=timestamp(monthsstart(2,date))
```

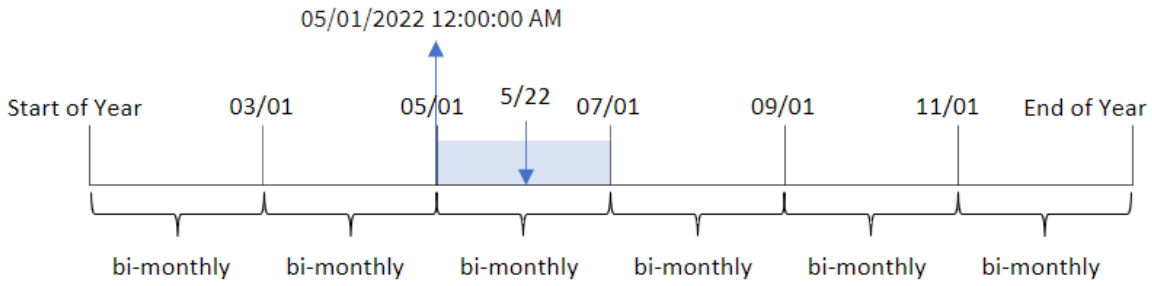
Te obliczenia pobiorą początkowy znacznik czasu dwumiesięcznego segmentu, w którym została zawarta każda transakcja.

Tabela wynikowa

date	=monthsstart(2,date)	=timestamp(monthsstart(2,date))
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
10/14/2022	09/01/2022	9/1/2022 12:00:00 AM
10/29/2022	09/01/2022	9/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
5/1/2022	05/01/2022	5/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/22/2022	05/01/2022	5/1/2022 12:00:00 AM
6/15/2022	05/01/2022	5/1/2022 12:00:00 AM
6/26/2022	05/01/2022	5/1/2022 12:00:00 AM

date	=monthsstart(2,date)	=timestamp(monthsstart(2,date))
3/7/2022	03/01/2022	3/1/2022 12:00:00 AM
3/30/2022	03/01/2022	3/1/2022 12:00:00 AM
4/5/2022	03/01/2022	3/1/2022 12:00:00 AM
4/16/2022	03/01/2022	3/1/2022 12:00:00 AM
2/19/2022	01/01/2022	1/1/2021 12:00:00 AM

Diagram funkcji `monthsstart()`, przykład obiektu wykresu



Transakcja 8195 miała miejsce 22 maja. Funkcja `monthsstart()` początkowo dzieli rok na segmenty dwumiesięczne. Transakcja 8195 należy do segmentu od 1 maja do 30 czerwca. W związku z tym funkcja zwraca pierwszą milisekundę tego segmentu, 01.05.2022 r., godz 00:00:00.

Przykład 5 – scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw sald kredytów, który jest ładowany do tabeli o nazwie `Loans`.
- Dane zawierające identyfikatory kredytów, saldo na początku miesiąca i prostą stopę procentową naliczaną od każdego kredytu rocznie.

Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał według identyfikatora pożyczki bieżące odsetki naliczone od każdej pożyczki w wybranym okresie. Rok obrotowy rozpoczyna się w styczniu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz.

Na początku skryptu ładowania została utworzona zmienna (`vPeriod`), która zostanie powiązana z kontrolką wejściową zmiennej. Następnie skonfiguruj zmienną jako obiekt niestandardowy w arkuszu.

Wykonaj następujące czynności:

1. W panelu zasobów kliknij opcję **Obiekty niestandardowe**.
2. Wybierz **Qlik Dashboard bundle** i utwórz obiekt **Wprowadzanie zmiennych**.
3. Wprowadź tytuł obiektu wykresu.
4. W polu **Zmienna** wybierz **vPeriod** jako nazwę i ustaw obiekt tak, aby był wyświetlany jako **lista rozwijana**.
5. W obszarze **Wartości** skonfiguruj obiekt tak, aby używał wartości dynamicznych. Wprowadź: `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'`

Następnie utwórz tabelę wyników.

Wykonaj następujące czynności:

1. Utwórz nową tabelę. Dodaj następujące pola jako wymiary:
   - `employee_id`
   - `employee_name`
2. Utwórz miarę, aby obliczyć narosłe odsetki:
   `=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)`
3. Ustaw **Formatowanie liczb** miary na **Waluta**. Kliknij przycisk ✔ **Koniec edycji**. Możesz teraz modyfikować dane pokazane w tabeli, dostosowując segment czasu w obiekcie zmiennych.

Tak będzie wyglądać tabela wyników po wybraniu opcji okresu `month`:

Tabela wynikowa

loan_id	start_balance	=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)
8188	$10000.00	$7.95
8189	$15000.00	$67.93

loan_id	start_balance	=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)
8190	$17500.00	$33.37
8191	$21000.00	$56.73
8192	$90000.00	$600.66

Funkcja `monthsstart()` przyjmuje dane od użytkownika jako pierwszy argument i bieżącą datę jako drugi argument, po czym zwraca datę początkową okresu wybranego przez użytkownika. Odejmując ten wynik od bieżącej daty, wyrażenie zwraca liczbę dni, które upłynęły do tej pory w tym okresie.

Wartość ta jest następnie mnożona przez stopę procentową i dzielona przez 365, aby uzyskać wysokość odsetek nagromadzonych do tego momentu tygodnia. Wynik jest następnie mnożony przez saldo początkowe pożyczki, aby zwrócić odsetki naliczone do tej pory w tym okresie.

monthstart

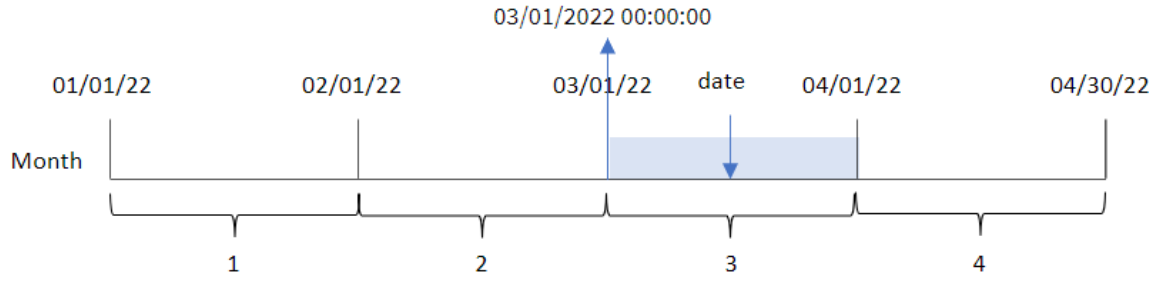
Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu pierwszej milisekundy ostatniego dnia miesiąca zawierającego wartość **date**. Domyślnym formatem wyjściowym będzie format **DateFormat** skonfigurowany w skrypcie.

Składnia:

```
MonthStart(date[, period_no])
```

Typ zwracanych danych: dual

Schemat funkcji `monthstart()`



Funkcja `monthstart()` określa, w którym miesiącu wypada data. Następnie zwraca znacznik czasu w formacie daty dla pierwszej milisekundy tego miesiąca.

Argumenty

Argument	Opis
date	Data lub znacznik czasu do oszacowania.
period_no	**period_no** jest liczbą całkowitą, która, jeśli ma wartość 0 lub jest pominięta, oznacza miesiąc, który zawiera **date**. Wartości ujemne parametru **period_no** oznaczają miesiące poprzednie, a wartości dodatnie – miesiące następne.

Kiedy używać

Funkcja `monthstart()` jest zwykle używana jako część wyrażenia, gdy użytkownik chce, by w obliczeniach użyto ułamka miesiąca, który upłynął do tej pory. Przy jej użyciu można na przykład obliczyć odsetki narosłe w ciągu miesiąca do określonej daty.

Przykłady funkcji

Przykład	Wynik
`monthstart('10/19/2001')`	Zwraca wartość 10/01/2001.
`monthstart('10/19/2001', -1)`	Zwraca wartość 09/01/2001.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „`Transactions`”.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie pola, `start_of_month`, zwracającego znacznik czasu początku miesiąca, w którym zostały zawarte transakcje.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `start_of_month`
- `start_of_month_timestamp`

Tabela wynikowa

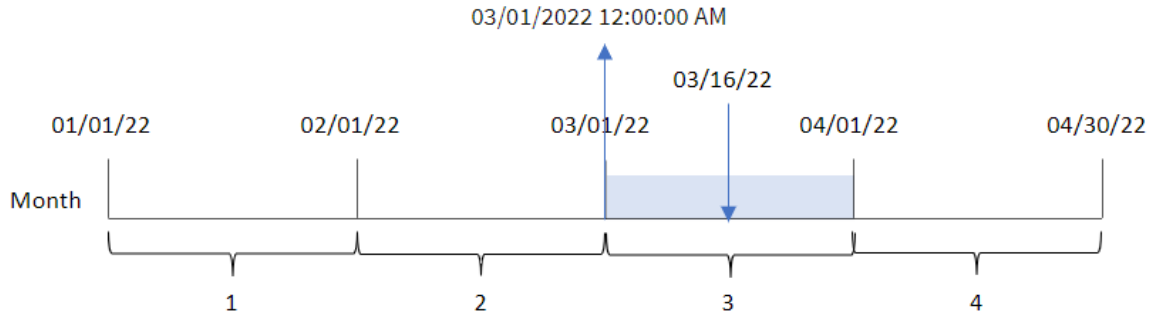
date	start_of_month	start_of_month_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM
2/5/2022	02/01/2022	2/1/2022 12:00:00 AM
2/28/2022	02/01/2022	2/1/2022 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM

date	start_of_month	start_of_month_timestamp
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

Pole `start_of_month` jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `monthstart()` i przekazanie pola daty jako jej argumentu.

Funkcja `monthstart()` identyfikuje, w którym miesiącu przypada wartość daty, i zwraca znacznik czasu pierwszej milisekundy danego miesiąca.

Diagram funkcji `monthstart()`, przykład bez dodatkowych argumentów



Transakcja 8192 miała miejsce 16 marca. Funkcja `monthstart()` zwraca pierwszą milisekundę tego miesiąca, czyli 1 marca o godz 00:00:00.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola, `previous_month_start`, zwracającego znacznik czasu początku miesiąca, przed którym została zawarta transakcja.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

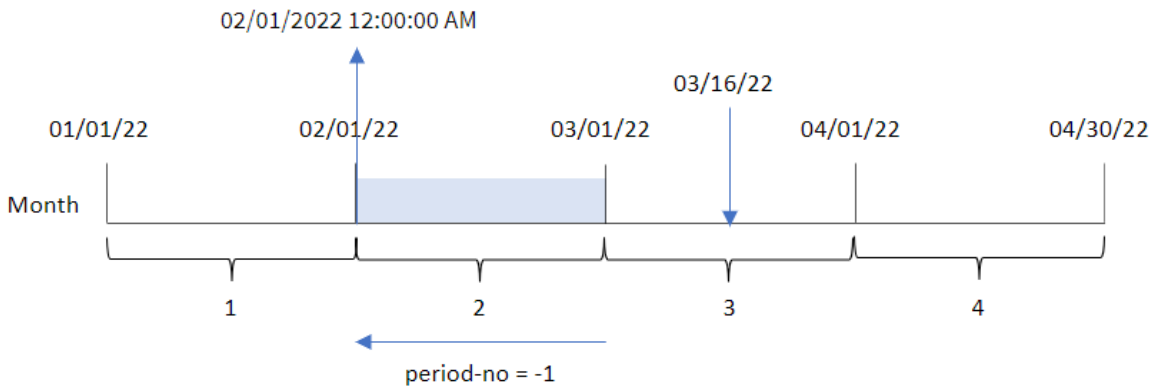
- `date`
- `previous_month_start`
- `previous_month_start_timestamp`

Tabela wynikowa

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
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

W tym przypadku, ponieważ wartości `period_no` -1 użyto jako argumentu przesunięcia w funkcji `monthstart()`, funkcja najpierw identyfikuje miesiąc, w którym odbywają się transakcje. Następnie przesuwa zakres o miesiąc wstecz i identyfikuje pierwszą milisekundę tego miesiąca.

Diagram funkcji `monthstart()`, przykład z argumentem period_no



Transakcja 8192 miała miejsce 16 marca. Funkcja `monthstart()` stwierdza, że miesiąc poprzedzający miesiąc transakcji to luty. Następnie zwraca pierwszą milisekundę tego miesiąca – 1 lutego o godz. 00:00:00.

Przykład 3 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenie zwracające znacznik czasu początku miesiąca, kiedy wystąpiły transakcje, jest tworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Aby obliczyć początek miesiąca, w którym zawarto transakcję, utwórz następujące miary:

- `=monthstart(date)`
- `=timestamp(monthstart(date))`

Tabela wynikowa

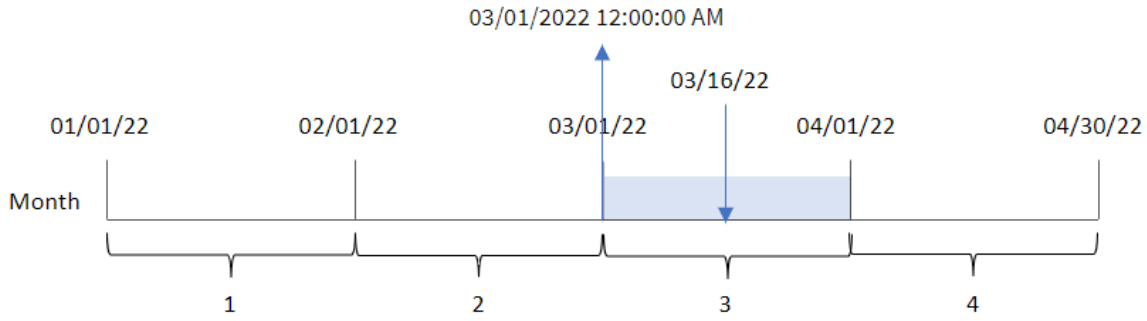
date	=monthstart(date)	=timestamp(monthstart(date))
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	8/1/2022 12:00:00 AM
8/19/2022	08/01/2022	8/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	06/01/2022	6/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/16/2022	05/01/2022	5/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM

date	=monthstart(date)	=timestamp(monthstart(date))
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

Miarę `start_of_month` tworzy się w obiekcie wykresu, używając funkcji `monthstart()` i przekazując pole daty jako jej argument.

Funkcja `monthstart()` identyfikuje, w którym miesiącu przypada wartość daty, i zwraca znacznik czasu pierwszej milisekundy danego miesiąca.

Diagram funkcji `monthstart()`, przykład obiektu wykresu



Transakcja 8192 miała miejsce 16 marca. Funkcja `monthstart()` stwierdza, że pierwsza transakcja miała miejsce w marcu i zwraca pierwszą milisekundę tego miesiąca, czyli 1 marca o godz 00:00:00.

Przykład 4 – Scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw sald kredytów, który jest ładowany do tabeli o nazwie `Loans`.
- Dane zawierające identyfikatory kredytów, saldo na początku miesiąca i prostą stopę procentową naliczaną od każdego kredytu rocznie.

Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał według identyfikatora pożyczki bieżące odsetki naliczone od każdej pożyczki w bieżącym miesiącu.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:
   - `loan_id`
   - `start_balance`
2. Następnie utwórz miarę, aby obliczyć narosłe odsetki:
   `=start_balance*(rate*(today(1)-monthstart(today(1)))/365)`
3. Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

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

Używając dzisiejszej daty jako jedynego argumentu, funkcja `monthstart()` zwraca datę początkową bieżącego miesiąca. Odejmując ten wynik od bieżącej daty, wyrażenie zwraca liczbę dni, które upłynęły do tej pory w tym miesiącu.

Wartość ta jest następnie mnożona przez stopę procentową i dzielona przez 365, aby uzyskać wysokość odsetek nagromadzonych do tego momentu tygodnia. Wynik jest następnie mnożony przez saldo początkowe pożyczki, aby zwrócić odsetki naliczone do tej pory w tym miesiącu.

networkdays

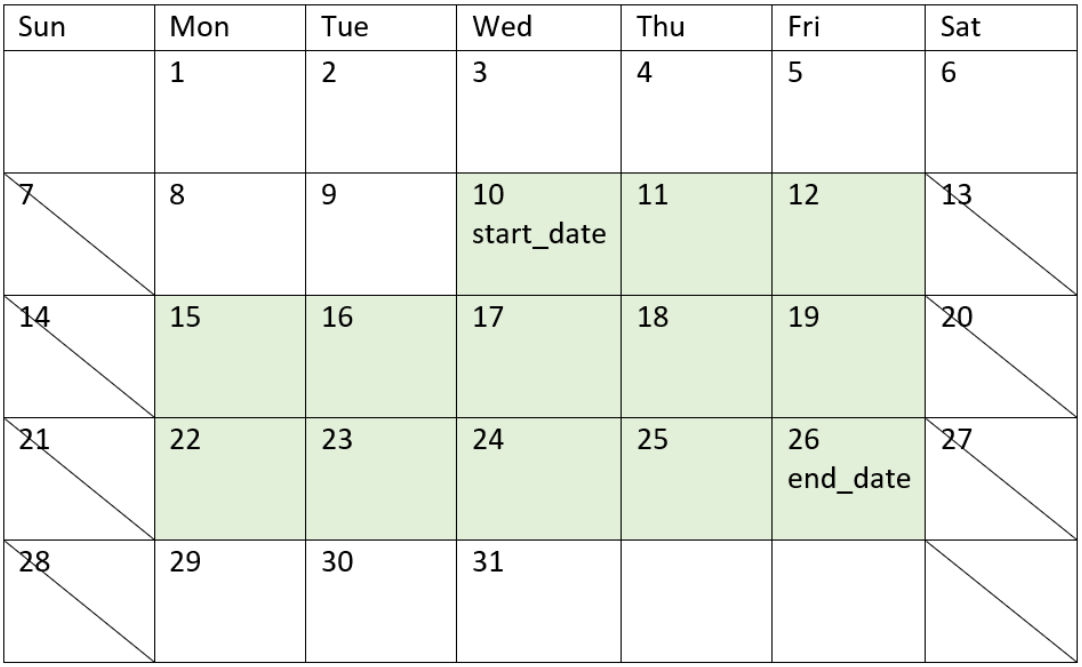
Funkcja **networkdays** zwraca liczbę dni roboczych (poniedziałek-piątek) od **start_date** do **end_date** włącznie z uwzględnieniem opcjonalnych dni wolnych (**holiday**).

Składnia:

```
networkdays (start_date, end_date [, holiday])
```

Typ zwracanych danych: integer

Diagram kalendarza przedstawiający zakres dat zwracany przez funkcję networkdays



Funkcja `networkdays` ma następujące ograniczenia:

- Nie ma sposobu na zmodyfikowanie dni roboczych. Innymi słowy, nie da się zmodyfikować funkcji pod kątem regionów lub sytuacji obejmujących cokolwiek innego niż dni robocze od poniedziałku do piątku.
- Parametr `holiday` musi być stałą łańcuchową. Wyrażenia nie są akceptowane.

Argumenty

Argument	Opis
start_date	Data rozpoczęcia do oceny.
end_date	Data zakończenia do oceny.

Argument	Opis
holiday	Okresy wolne od pracy wyłączane z dni roboczych. Dzień wolny jest określany jako data stała w formie ciągu. Można określić większą liczbę dat dni wolnych, rozdzielając je przecinkami. **Przykład:** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Kiedy używać

Funkcja `networkdays()` jest używana jako część wyrażenia, gdy użytkownik chce, aby w obliczeniach użyto liczbę dni roboczych między dwiema datami. Można jej użyć na przykład do obliczenia sumy zarobków pracownika zatrudnionego na zasadach PAYE (pay-as-you-earn).

Przykłady funkcji

Przykład	Wynik
`networkdays ('12/19/2013', '01/07/2014')`	Zwraca wartość 14 W tym przykładzie okresy wolne od pracy nie są uwzględnione.
`networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013')`	Zwraca wartość 12 W tym przykładzie uwzględniony jest okres wolny od pracy od 12/25/2013 do 12/26/2013.
`networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014')`	Zwraca wartość 10 W tym przykładzie uwzględnione są dwa okresy wolne od pracy.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Przykład podstawowy

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający identyfikatory projektów oraz ich daty rozpoczęcia i zakończenia. Te informacje są ładowane do tabeli o nazwie `Projects`.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie dodatkowego pola, `net_work_days`, aby obliczyć liczbę dni roboczych objętych przez każdy projekt.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `start_date`
- `end_date`
- `net_work_days`

Tabela wynikowa

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

Ponieważ nie ma zaplanowanych dni wolnych (byłyby określone w trzecim argumencie funkcji `networkdays ()`), funkcja oblicza liczbę dni roboczych między dwiema datami przez odjęcie `start_date` od `end_date`.

Diagram kalendarza z zaznaczonymi dniami roboczymi dla projektu 5 (brak dni wolnych)

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			

Powyższy kalendarz wizualnie zaznacza projekt o `id` 5. Projekt 5 zaczyna się w środę 10 sierpnia 2022 r. i kończy 26 sierpnia 2022 r. Po odjęciu wszystkich sobót i niedziel między tymi dwiema datami włącznie pozostaje 13 dni roboczych.

Przykład 2 – jeden dzień wolny

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w poprzednim przykładzie.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie dodatkowego pola, `net_work_days`, aby obliczyć liczbę dni roboczych objętych przez każdy projekt.

W tym przykładzie występuje jeden dzień wolny przypadający 19 sierpnia 2022 r.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `start_date`
- `end_date`
- `net_work_days`

Tabela wynikowa

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

Dzień świąteczny został wprowadzony jako trzeci argument do funkcji `networkdays()`.

Diagram kalendarza z zaznaczonymi dniami roboczymi dla projektu 5 (jeden dzień wolny)

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			

Powyższy kalendarz wizualnie pokazuje ramy projektu 5, demonstrując korektę uwzględniającą dzień świąteczny. Ten dzień wypada w trakcie trwania projektu 5 w piątek 19 sierpnia 2022 r. W efekcie całkowita wartość `net_work_days` dla projektu 5 zostaje zmniejszona o jeden dzień, czyli z 13 do 12 dni.

Przykład 3 – kilka dni świątecznych

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie dodatkowego pola, `net_work_days`, aby obliczyć liczbę dni roboczych objętych przez każdy projekt.

Jednak w tym przykładzie są cztery dni robocze zaplanowane między 18 a 21 sierpnia 2022 r.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `start_date`
- `end_date`
- `net_work_days`

Tabela wynikowa

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

Cztery zaplanowane dni wolne zostały wpisane jako lista elementów rozdzielanych przecinkami, od trzeciego argumentu w funkcji `networkdays()`.

Diagram kalendarza z zaznaczonymi dniami roboczymi dla projektu 5 (kilka dni wolnych)

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			

Powyższy kalendarz wizualnie pokazuje ramy projektu 5, demonstrując korektę uwzględniającą te dni świąteczne. Okres zaplanowanych dni wolnych przypada w trakcie trwania projektu 5, z czego dwa z tych dni wypadają w czwartek i piątek. W efekcie całkowita wartość `net_work_days` dla projektu 5 zostaje zmniejszona z 13 do 11 dni.

Przykład 4 – jeden dzień wolny

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/RRRR).

Jest jeden dzień wolny przypadający 19 sierpnia 2022 r.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Pole `net_work_days` jest obliczane jako miara w obiekcie wykresu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `start_date`
- `end_date`

Utwórz następującą miarę:

```
= networkdays(start_date,end_date,'08/19/2022')
```

Tabela wynikowa

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

Dzień świąteczny został wprowadzony jako trzeci argument do funkcji `networkdays()`.

Diagram kalendarza pokazujący dni robocze netto z jednym dniem wolnym (obiekt wykresu)

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			

Powyższy kalendarz wizualnie pokazuje ramy projektu 5, demonstrując korektę uwzględniającą dzień świąteczny. Ten dzień wypada w trakcie trwania projektu 5 w piątek 19 sierpnia 2022 r. W efekcie całkowita wartość `net_work_days` dla projektu 5 zostaje zmniejszona o jeden dzień, czyli z 13 do 12 dni.

now

Ta funkcja zwraca znacznik bieżącego czasu. Funkcja zwraca wartości w formacie zmiennej systemowej **TimeStamp**. Wartością domyślną **timer_mode** jest 1.

Składnia:

```
now([ timer_mode])
```

Typ zwracanych danych: dual

Funkcji `now()` można użyć w skrypcie ładowania lub w obiektach wykresów.

Argumenty

Argument	Opis
timer_mode	Może przyjmować następujące wartości: 0 (czas ostatnio zakończonego ładowania danych) 1 (czas w momencie wywołania funkcji) 2 (czas w momencie otwarcia aplikacji) *W przypadku użycia funkcji w skrypcie ładowania danych parametr **timer_mode=0** da wynik będący czasem ostatnio zakończonego ładowania danych, z kolei parametr **timer_mode=1** zwróci czas wywołania funkcji w bieżącym ładowaniu danych.*

Kiedy używać

Funkcja `now()` jest używana jako element wyrażenia. Za jej pomocą można na przykład obliczyć pozostały czas cyklu życia produktu. Gdyby wyrażenie wymagało użycia ułamka dnia, zamiast funkcji `today()` użyto by funkcji `now()`.

Poniższa tabela zawiera objaśnienie wyniku zwracanego przez funkcję `now()` dla różnych wartości argumentu `timer_mode`:

Przykłady funkcji

timer_mode value	Wynik w przypadku użycia w skrypcie ładowania	Wynik w przypadku użycia w obiekcie wykresu
0	Zwraca znacznik czasu, w formacie zmiennej systemowej `TimeStamp`, ostatniego udanego przeładowania danych przed ostatnim załadowaniem danych.	Zwraca znacznik czasu, w formacie zmiennej systemowej `TimeStamp`, ostatniego przeładowania danych.
1	Zwraca znacznik czasu, w formacie zmiennej systemowej `TimeStamp`, ostatniego przeładowania danych.	Zwraca znacznik czasu, w formacie zmiennej systemowej `TimeStamp`, wywołania funkcji.
2	Zwraca znacznik czasu, w formacie zmiennej systemowej `TimeStamp`, początku sesji użytkownika w aplikacji. Zostanie on zaktualizowany dopiero po ponownym załadowaniu skryptu przez użytkownika.	Zwraca znacznik czasu, w formacie zmiennej systemowej `TimeStamp`, początku sesji użytkownika w aplikacji. Będzie on odświeżany na początku sesji lub po ponownym załadowaniu danych do aplikacji.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne

czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – generowanie obiektów przy użyciu skryptu ładowania

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Ten przykład tworzy trzy zmienne za pomocą funkcji `now()`. Każda zmienna używa jednej z opcji `timer_mode` w celu zademonstrowania tego efektu.

Aby zademonstrować przeznaczenie zmiennych, przeładuj skrypt, a następnie, po krótkim czasie, przeładuj go po raz drugi. Spowoduje to, że zmienne `now(0)` i `now(1)` będą miały różne wartości, co będzie stanowiło właściwą demonstrację ich przeznaczenia.

Skrypt ładowania

```
LET vPreviousDataLoad = now(0);
LET vCurrentDataLoad = now(1);
LET vApplicationOpened = now(2);
```

Wyniki

Gdy dane zostaną załadowane po raz drugi, utwórz trzy pola tekstowe, korzystając z poniższych wskazówek.

Najpierw utwórz pole tekstowe dla danych, które zostały wcześniej załadowane.

Wykonaj następujące czynności:

1. Używając obiektu wykresu **Tekst i grafika**, utwórz pole tekstowe.
2. Dodaj następującą miarę do obiektu:
   `=vPreviousDataLoad`
3. W sekcji **Wygląd** wybierz **Show titles** i dodaj do obiektu tytuł „Previous Reload Time”.

Następnie utwórz pole tekstowe dla danych, które są aktualnie ładowane.

Wykonaj następujące czynności:

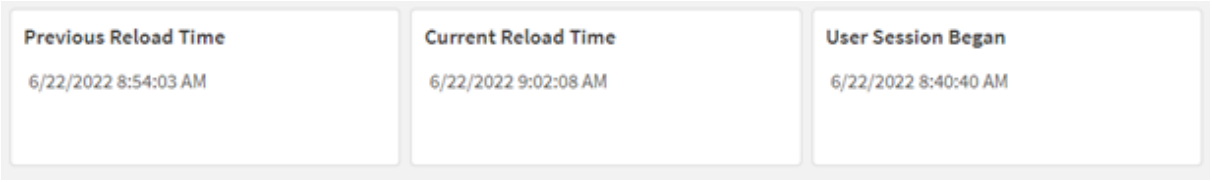
1. Używając obiektu wykresu **Tekst i grafika**, utwórz pole tekstowe.
2. Dodaj następującą miarę do obiektu:
   `=vCurrentDataLoad`
3. W sekcji **Wygląd** wybierz **Show titles** i dodaj do obiektu tytuł „Current Reload Time”.

Utwórz ostatnie pole tekstowe, aby pokazać, kiedy sesja użytkownika w aplikacji została rozpoczęta.

Wykonaj następujące czynności:

1. Używając obiektu wykresu **Tekst i grafika**, utwórz pole tekstowe.
2. Dodaj następującą miarę do obiektu:
   `=vApplicationOpened`
3. W sekcji **Wygląd** wybierz **Show titles** i dodaj do obiektu tytuł „User Session Started”.

Zmienne skryptu ładowania `now()`



Powyższy obraz pokazuje przykładowe wartości dla każdej z utworzonych zmiennych. Na przykład, te wartości mogą być następujące:

- Czas poprzedniego przeładowania: 22.06.2022 r., godz. 8:54:03
- Czas bieżącego przeładowania: 22.06.2022 r., godz. 9:02:08
- Początek sesji użytkownika: 22.06.2022 r., godz. 8:40:40

Przykład 2 – generowanie obiektów bez skryptu ładowania

Skrypt ładowania i wyrażenie wykresu

Przegląd

W tym przykładzie utworzymy trzy obiekty wykresu za pomocą funkcji `now()`, bez ładowania jakichkolwiek zmiennych lub danych do aplikacji. Każdy obiekt wykresu używa jednej z opcji `timer_mode` w celu zademonstrowania tego efektu.

W tym przykładzie nie ma skryptu ładowania.

Wykonaj następujące czynności:

1. Otwórz edytor ładowania danych.
2. Kliknij pozycję **Załaduj dane**, nie zmieniając istniejącego skryptu ładowania.

3. Po krótkim czasie załaduj skrypt drugi raz.

Wyniki

Po załadowaniu danych drugi raz utwórz trzy pola tekstowe.

Najpierw utwórz pole tekstowe dla ostatniego przeładowania danych.

Wykonaj następujące czynności:

1. Używając obiektu wykresu **Tekst i grafika**, utwórz pole tekstowe.
2. Dodaj następującą miarę:
   `=now(0)`
3. W sekcji **Wygląd** wybierz **Show titles** i dodaj do obiektu tytuł „Ostatnie przeładowanie danych”.

Następnie utwórz pole tekstowe, aby pokazać bieżący czas.

Wykonaj następujące czynności:

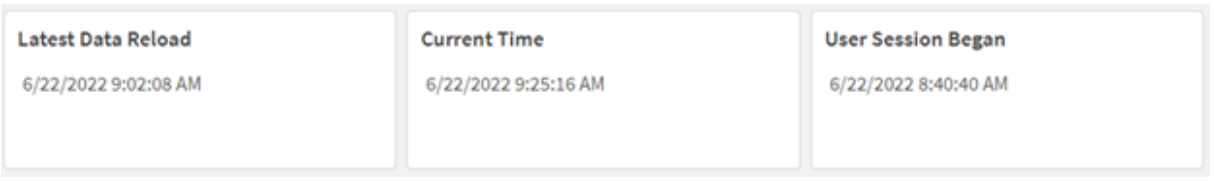
1. Używając obiektu wykresu **Tekst i grafika**, utwórz pole tekstowe.
2. Dodaj następującą miarę:
   `=now(1)`
3. W sekcji **Wygląd** wybierz **Show titles** i dodaj do obiektu tytuł „Bieżący czas”.

Utwórz ostatnie pole tekstowe, aby pokazać, kiedy sesja użytkownika w aplikacji została rozpoczęta.

Wykonaj następujące czynności:

1. Używając obiektu wykresu **Tekst i grafika**, utwórz pole tekstowe.
2. Dodaj następującą miarę:
   `=now(2)`
3. W sekcji **Wygląd** wybierz **Show titles** i dodaj do obiektu tytuł „Sesja użytkownika rozpoczęta”.

Przykłady obiektu wykresu `now()`



Powyższy obraz pokazuje przykładowe wartości dla każdego z utworzonych obiektów. Na przykład, te wartości mogą być następujące:

- Ostatnie przeładowanie danych: 22.06.2022 r., godz. 9:02:08
- Bieżący czas: 22.06.2022 r., godz. 9:25:16
- Początek sesji użytkownika: 22.06.2022 r., godz. 8:40:40

Obiekt wykresu „Ostatnie przeładowanie danych” używa argumentu `timer_mode` o wartości 0. Powoduje to zwrócenie znacznika czasu ostatniego udanego przeładowania danych.

Obiekt wykresu „Bieżący czas” używa argumentu `timer_mode` o wartości 1. Powoduje to zwrócenie bieżącego czasu według zegara systemowego. Jeśli arkusz lub obiekt zostanie odświeżony, ta wartość zostanie zaktualizowana.

Obiekt wykresu „Początek sesji użytkownika” używa argumentu `timer_mode` o wartości 2. Powoduje to zwrócenie znacznika czasu otwarcia aplikacji i rozpoczęcia sesji użytkownika.

Przykład 3 – Scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający magazyn operacji kopania kryptowalut, który jest załadowany do tabeli o nazwie `Inventory`.
- Dane z następującymi polami: `id`, `purchase_date` i `wph` (waty na godzinę).

Użytkownik chciałby otrzymać tabelę pokazującą, według `id`, sumę kosztów wygenerowanych przez każdy sprzęt wydobywczy w ciągu miesiąca do tej pory, wyrażoną według zużycia energii.

Ta wartość powinna być aktualizowana po każdym odświeżeniu obiektu wykresu. Obecnie cena energii elektrycznej wynosi 0,0678 dolara za kWh.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `id`.

Utwórz następującą miarę:

```
=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
```

Gdyby obiekt wykresu odświeżono 22.06.2022 o godz. 10:39:05, zwróciłby następujący wynik:

Tabela wynikowa

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

Użytkownik chciałby, aby wyniki obiektu były odświeżane po każdym odświeżeniu tego obiektu. Dlatego przekazano argument `timer_mode` dla egzemplarzy funkcji `now()` w wyrażeniu. Znacznik czasu początku miesiąca, zidentyfikowany przez użycie funkcji `now()` jako argumentu znacznika czasu funkcji `monthstart()`, zostaje odjęty od bieżącego czasu, który jest identyfikowany przez funkcję `now()`. W wyniku otrzymujemy sumę czasu w dniach, jaki upłynął do tej pory od początku miesiąca.

Ta wartość zostaje pomnożona przez 24 (liczba godzin w dobie), a następnie przez wartość obecną w polu `wph`.

Aby zamienić waty na godzinę na kilowaty na godzinę, wynik zostaje podzielony przez tysiąc przed wykonaniem mnożenia przez cenę jednej kilowatogodziny.

quarterend

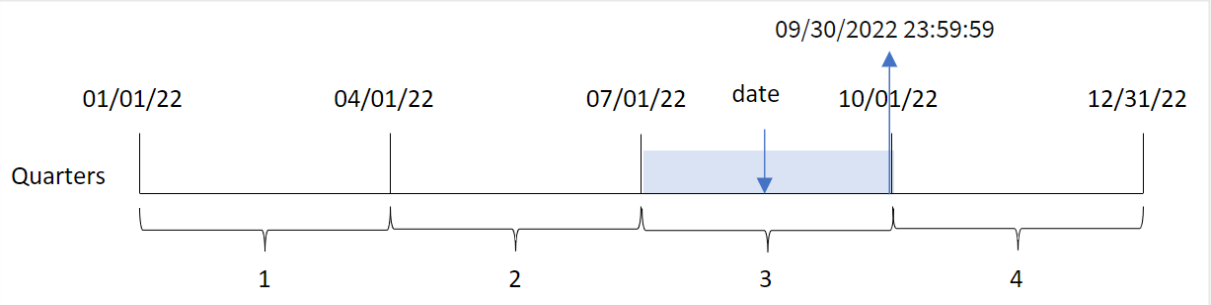
Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu ostatniej milisekundy kwartału zawierającego wartość **date**. Domyślnym formatem wyjściowym będzie format **DateFormat** skonfigurowany w skrypcie.

Składnia:

```
QuarterEnd(date[, period_no[, first_month_of_year]])
```

Typ zwracanych danych: dual

Diagram funkcji `quarterend()`



Funkcja `quarterend()` określa, w którym kwartale wypada data. Następnie zwraca znacznik czasu, w formacie daty, ostatniej milisekundy ostatniego miesiąca tego kwartału\. Pierwszym miesiącem roku jest domyślnie styczeń. Możesz jednak również zmienić pierwszy dzień tygodnia, używając argumentu `first_month_of_year` w funkcji `quarterend()`.

> *Funkcja `quarterend()` nie uwzględnia zmiennej systemowej `FirstMonthOfYear`. Rok rozpoczyna się 1 stycznia, chyba że do jego zmiany użyto argumentu `first_month_of_year`.*

Kiedy używać

Funkcja `quarterend()` jest używana jako część wyrażenia, gdy użytkownik chce, by w obliczeniach użyto ułamka kwartału, który jeszcze nie nastąpił. Na przykład, jeśli chcesz obliczyć łączne odsetki, które nie zostały jeszcze naliczone w ciągu kwartału.

Argumenty

Argument	Opis
date	Data lub znacznik czasu do oszacowania.
period_no	**period_no** jest liczbą całkowitą, gdzie 0 oznacza kwartał, który zawiera datę **date**. Wartości ujemne parametru **period_no** oznaczają kwartały poprzednie, a wartości dodatnie – kwartały następne.
first_month_of_year	Jeśli użytkownik zamierza korzystać z lat (obrotowych), które nie zaczynają się w styczniu, powinien wskazać wartość od 2 do 12 jako parametr **first_month_of_year**.

Aby ustawić pierwszy miesiąc roku w argumencie `first_month_of_year`, możesz użyć następujących wartości:

first_month_of_year values

Miesiąc	Wartość
Luty	2
Marzec	3
Kwiecień	4
May	5
Czerwiec	6
Lipiec	7
Sierpień	8
Wrzesień	9
Październik	10
Listopad	11
Grudzień	12

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień

regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
`quarterend('10/29/2005')`	Returns 12/31/2005 23:59:59.
`quarterend('10/29/2005', -1)`	Returns 09/30/2005 23:59:59.
`quarterend('10/29/2005', 0, 3)`	Returns 11/30/2005 23:59:59.

Przykład 1 – Przykład podstawowy

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Ładunek poprzedzający, który zawiera następujące elementy:
  - Funkcja `quarterend()`, która jest ustawiona jako pole `end_of_quarter` i zwraca znacznik czasu końca kwartału, w którym miały miejsce transakcje.
  - Funkcja `timestamp()`, która jest ustawiona jako pole `end_of_quarter_timestamp` i zwraca dokładny znacznik czasu końca wybranego kwartału.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `end_of_quarter`
- `end_of_quarter_timestamp`

Tabela wynikowa

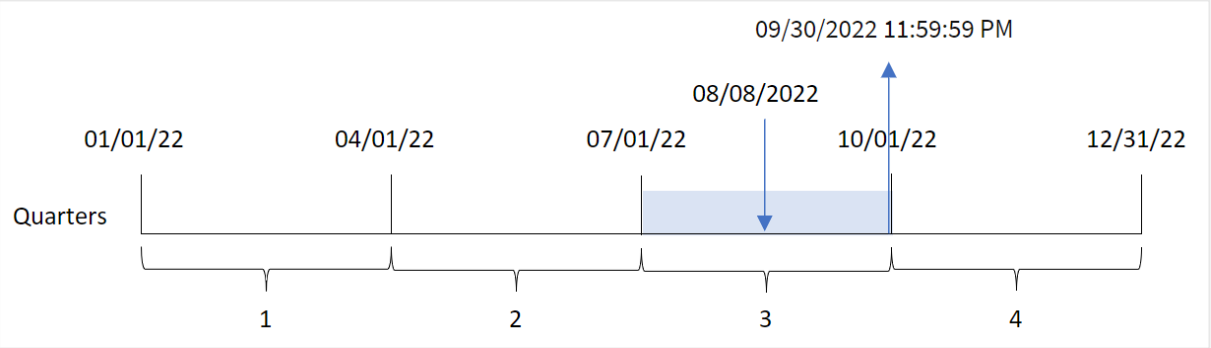
id	date	end_of_quarter	end_of_quarter_timestamp
8188	1/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8189	1/19/2022	03/31/2022	3/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	06/30/2022	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

id	date	end_of_quarter	end_of_quarter_timestamp
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

Pole „`end_of_quarter`" jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `quarterend()` i przekazanie pola daty jako jej argumentu.

Funkcja `quarterend()` początkowo identyfikuje, w którym kwartale przypada wartość daty, i zwraca znacznik czasu ostatniej milisekundy danego kwartału.

Diagram funkcji `quarterend()` ze zidentyfikowanym końcem kwartału transakcji 8203



Transakcja 8203 miała miejsce 8 sierpnia. Funkcja `quarterend()` stwierdza, że transakcja miała miejsce w trzecim kwartale oraz zwraca ostatnią milisekundę tego kwartału – 30 września, godz. 23:59:59.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Ładunek poprzedzający, który zawiera następujące elementy:

- Funkcja `quarterend()`, która jest ustawiona jako pole `previous_quarter_end` i zwraca znacznik czasu końca kwartału przed tym, w którym miała miejsce transakcja.
- Funkcja `timestamp()`, która jest ustawiona jako pole `previous_end_of_quarter_timestamp` i zwraca dokładny znacznik czasu końca kwartału przed tym, w którym miała miejsce transakcja.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

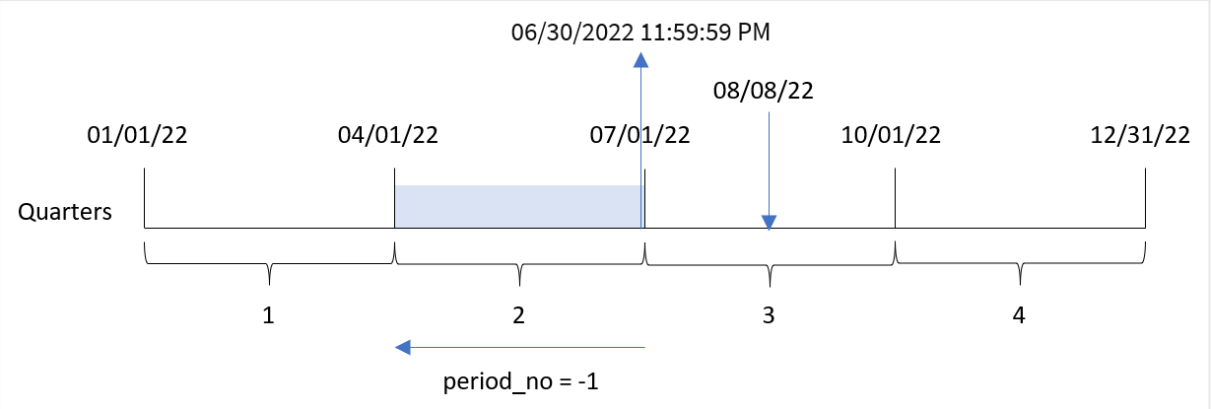
- `id`
- `date`
- `previous_quarter_end`
- `previous_quarter_end_timestamp`

Tabela wynikowa

id	date	previous_quarter_end	previous_quarter_end_timestamp
8188	1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
8189	1/19/2022	12/31/2021	12/31/2021 11:59:59 PM
8190	2/5/2022	12/31/2021	12/31/2021 11:59:59 PM
8191	2/28/2022	12/31/2021	12/31/2021 11:59:59 PM
8192	3/16/2022	12/31/2021	12/31/2021 11:59:59 PM
8193	4/1/2022	03/31/2022	3/31/2022 11:59:59 PM
8194	5/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8195	5/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8196	6/15/2022	03/31/2022	3/31/2022 11:59:59 PM
8197	6/26/2022	03/31/2022	3/31/2022 11:59:59 PM
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
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

Ponieważ `period_no` o wartości `-1` używa się jako argumentu przesunięcia w funkcji `quarterend()`, funkcja najpierw identyfikuje kwartał, w którym odbywają się transakcje. Następnie przesuwa zakres o kwartał wstecz i identyfikuje ostatnią milisekundę tego kwartału.

Diagram funkcji `quarterend()` z argumentem period_no o wartości -1.



Transakcja 8203 miała miejsce 8 sierpnia. Funkcja `quarterend()` stwierdza, że kwartał przed datą transakcji obejmował okres od 1 kwietnia do 30 czerwca. Funkcja zwraca ostatnią milisekundę tego kwartału – 30 czerwca, godz. 23:59:59.

Przykład 3 – first_month_of_year

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Ładunek poprzedzający, który zawiera następujące elementy:
  - Funkcja `quarterend()`, która jest ustawiona jako pole `end_of_quarter` i zwraca znacznik czasu końca kwartału, w którym miały miejsce transakcje.
  - Funkcja `timestamp()`, która jest ustawiona jako pole `end_of_quarter_timestamp` i zwraca dokładny znacznik czasu końca wybranego kwartału.

Jednak w tym przykładzie zgodnie z polityką firmy rok podatkowy zaczyna się 1 marca.

Skrypt ładowania

```
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
];
```

Wyniki

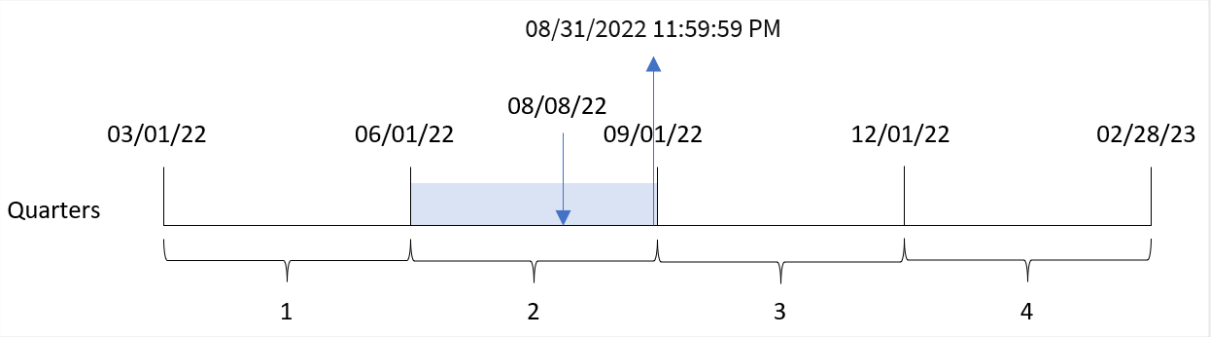
Tabela wynikowa

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

id	date	end_of_quarter	end_of_quarter_timestamp
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

Ponieważ jako argument `first_month_of_year` funkcji `quarterend()` przekazano wartość `3`, początek roku został przesunięty z 1 stycznia na 1 marca.

Diagram funkcji `quarterend()` z marcem jako pierwszym miesiącem roku



Transakcja 8203 miała miejsce 8 sierpnia. Ponieważ rok zaczyna się 1 marca, kwartały w nim obejmują miesiące marzec-maj, czerwiec-sierpień, wrzesień-listopad oraz grudzień-luty.

Funkcja `quarterend()` stwierdza, że transakcja miała miejsce w kwartale od czerwca do sierpnia i zwraca ostatnią milisekundę tego kwartału – 31 sierpnia, godz. 23:59:59.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie zestaw danych pozostaje bez zmian i jest ładowany do aplikacji. Obliczenia zwracające znacznik czasu końca kwartału, w którym wystąpiły transakcje, jest tworzone jako miara na wykresie w aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`

Aby obliczyć koniec kwartału, w którym zawarto transakcję, utwórz następujące miary:

- `=quarterend(date)`
- `=timestamp(quarterend(date))`

Tabela wynikowa

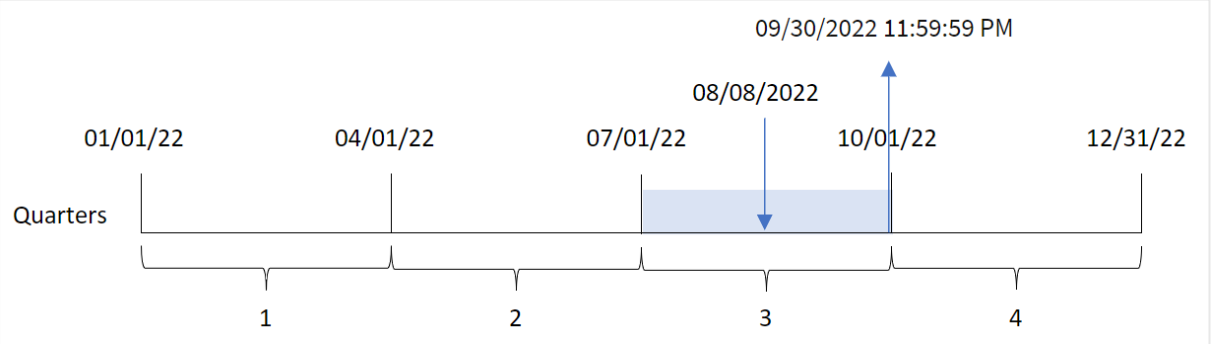
id	date	=quarterend(date)	=timestamp(quarterend(date))
8188	1/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8189	1/19/2022	03/31/2022	3/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM

id	date	=quarterend(date)	=timestamp(quarterend(date))
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

Pole „`end_of_quarter`” jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `quarterend()` i przekazanie pola daty jako jej argumentu.

Funkcja `quarterend()` początkowo identyfikuje, w którym kwartale przypada wartość daty, i zwraca znacznik czasu ostatniej milisekundy danego kwartału.

Diagram funkcji `quarterend()` ze zidentyfikowanym końcem kwartału transakcji 8203



Transakcja 8203 miała miejsce 8 sierpnia. Funkcja `quarterend()` stwierdza, że transakcja miała miejsce w trzecim kwartale oraz zwraca ostatnią milisekundę tego kwartału – 30 września, godz. 23:59:59.

Przykład 5 – scenariusz

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych jest ładowany do tabeli o nazwie `Employee_Expenses`. Tabela zawiera następujące pola:
    - Identyfikatory pracowników
    - Imiona i nazwiska pracownika
    - Średnie dzienne rozliczenia wydatków każdego pracownika.

Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał, według identyfikatora pracownika oraz imienia i nazwiska pracownika, szacowane roszczenia z tytułu wydatków do poniesienia przez pozostałą część kwartału. Rok obrotowy rozpoczyna się w styczniu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `employee_id`
- `employee_name`

Aby obliczyć narosłe odsetki, należy utworzyć następującą miarę:

- `=(quarterend(today(1))-today(1))*avg_daily_claim`

Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

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

Funkcja `quarterend()` używa bieżącej daty jako swojego jedynego argumentu i zwraca datę końcową bieżącego miesiąca. Następnie odejmuje bieżącą datę od daty końca roku i wyrażenie zwraca liczbę dni pozostałych w tym miesiącu.

Wartość ta jest następnie mnożona przez średnie dzienne roszczenie z tytułu wydatków przez każdego pracownika, aby obliczyć szacunkową wartość roszczeń, które każdy pracownik złoży w pozostałej części kwartału.

quartername

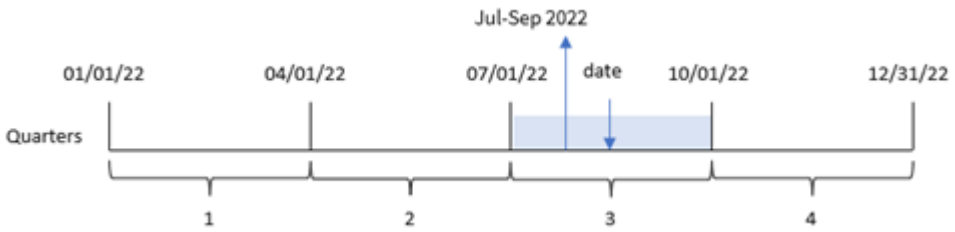
Ta funkcja zwraca wartość pokazującą miesiące kwartału (sformatowane zgodnie ze stosowaną w skryptach zmienną **MonthNames**) oraz rok z bazową wartością liczbową odpowiadającą znacznikowi czasu pierwszej milisekundy pierwszego dnia tego kwartału.

Składnia:

```
QuarterName(date[, period_no[, first_month_of_year]])
```

Typ zwracanych danych: dual

Schemat funkcji `quartername()`



Funkcja `quartername()` określa, w którym kwartale wypada data. Następnie zwraca wartość pokazującą pierwszy i ostatni miesiąc tego kwartału, jak również roku. Wartością, na podstawie której został zwrócony ten wynik, jest pierwsza milisekunda kwartału.

Argumenty

Argument	Opis
date	Data lub znacznik czasu do oszacowania.

Argument	Opis
period_no	**period_no** jest liczbą całkowitą, gdzie 0 oznacza kwartał, który zawiera datę **date**. Wartości ujemne parametru **period_no** oznaczają kwartały poprzednie, a wartości dodatnie – kwartały następne.
first_month_of_year	Jeśli użytkownik zamierza korzystać z lat (obrotowych), które nie zaczynają się w styczniu, powinien wskazać wartość od 2 do 12 jako parametr **first_month_of_year**.

Kiedy używać

Funkcja `quartername()` jest przydatna, gdy chcesz porównać agregacje według kwartału. Na przykład, jeśli chcesz zobaczyć całkowitą sprzedaż produktów według kwartału.

Tej funkcji można użyć w skrypcie ładowania w celu utworzenia pola w tabeli kalendarza głównego. Ewentualnie można jej użyć bezpośrednio na wykresie jako wymiaru wyliczanego.

W tych przykładach używany jest format daty MM/DD/YYYY. Format daty jest określony w instrukcji `SET DateFormat` u góry skryptu ładowania danych. Format zastosowany w przykładach można zmienić, aby dostosować go do konkretnych potrzeb.

Przykłady funkcji

Przykład	Wynik
`quartername('10/29/2013')`	Zwraca wartość Oct-Dec 2013.
`quartername('10/29/2013', -1)`	Zwraca wartość Jul-Sep 2013.
`quartername('10/29/2013', 0, 3)`	Zwraca wartość Sep-Nov 2013.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – data bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „Transactions”.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie pola `transaction_quarter` zwracającego kwartał, w którym miały miejsce transakcje.

Dodaj tutaj swój inny tekst, w razie potrzeby, z listami itp.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `transaction_quarter`

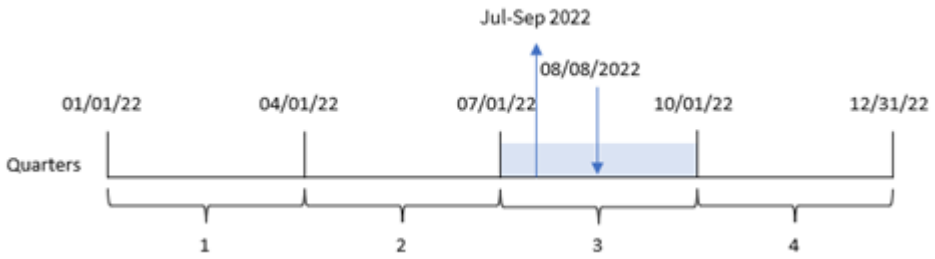
Tabela wynikowa

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
9/26/2022	Jul-Sep 2022
10/14/2022	Oct-Dec 2022
10/29/2022	Oct-Dec 2022

Pole „`transaction_quarter`" jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `quartername()` i przekazanie pola daty jako jej argumentu.

Funkcja `quartername()` początkowo identyfikuje kwartał, w którym przypada wartość daty. Następnie zwraca wartość pokazującą pierwszy i ostatni miesiąc tego kwartału, jak również roku.

Diagram funkcji `quartername()`, przykład bez dodatkowych argumentów



Transakcja 8203 miała miejsca 8 sierpnia 2022 roku. Funkcja `quartername()` stwierdza, że transakcja miała miejsce w trzecim kwartale, więc zwraca lip-wrz 2022 r. Miesiące są ukazane w takim samym formacie, jak zmienna systemowa `MonthNames`.

Przykład 2 – data z argumentem period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola `previous_quarter` zwracającego kwartał poprzedzający ten, w którym miały miejsce transakcje.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `previous_quarter`

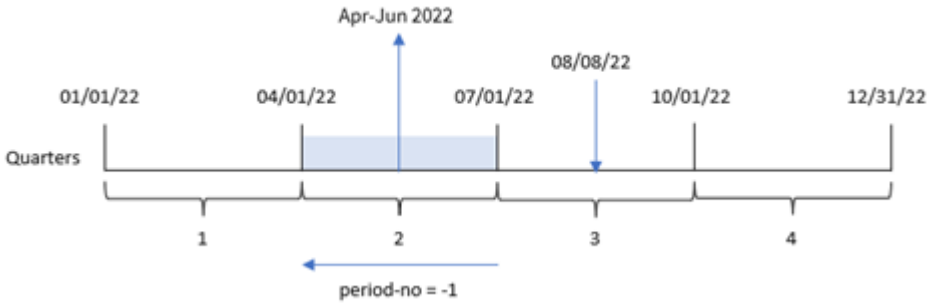
Tabela wynikowa

date	previous_quarter
1/7/2022	Oct-Dec 2021
1/19/2022	Oct-Dec 2021
2/5/2022	Oct-Dec 2021
2/28/2022	Oct-Dec 2021
3/16/2022	Oct-Dec 2021
4/1/2022	Jan-Mar 2022
5/7/2022	Jan-Mar 2022
5/16/2022	Jan-Mar 2022
6/15/2022	Jan-Mar 2022
6/26/2022	Jan-Mar 2022
7/9/2022	Apr-Jun 2022
7/22/2022	Apr-Jun 2022
7/23/2022	Apr-Jun 2022
7/27/2022	Apr-Jun 2022
8/2/2022	Apr-Jun 2022
8/8/2022	Apr-Jun 2022
8/19/2022	Apr-Jun 2022

date	previous_quarter
9/26/2022	Apr-Jun 2022
10/14/2022	Jul-Sep 2022
10/29/2022	Jul-Sep 2022

W tym przypadku, ponieważ do funkcji `quartername()` jako argument przesunięcia `period_no` przekazano wartość -1, funkcja ta najpierw stwierdza, że transakcje miały miejsce w trzecim kwartale. Następnie dokonuje przesunięcia o jeden kwartał wstecz i zwraca wartość pokazującą pierwszy i ostatni miesiąc tego kwartału, jak również roku.

Diagram funkcji `quartername()`, przykład z argumentem period_no



Transakcja 8203 miała miejsce 8 sierpnia. Funkcja `quartername()` stwierdza, że kwartał przed datą transakcji obejmował okres od 1 kwietnia do 30 czerwca. W związku z tym zwraca kwi-cze 2022 r.

Przykład 3 – data z argumentem first_week_day

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie. Jednak w tym przykładzie musimy również ustawić 1 marca jako pierwszy miesiąc roku podatkowego.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `transaction_quarter`

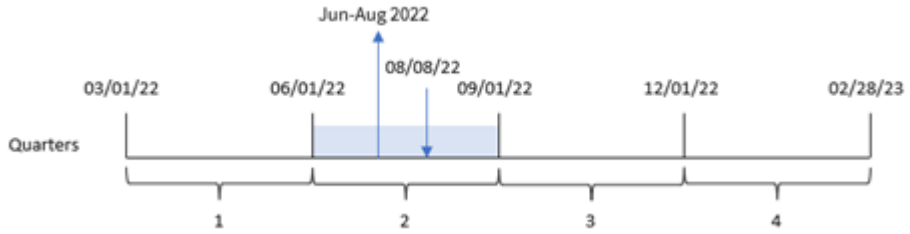
Tabela wynikowa

date	transaction_quarter
1/7/2022	Gru-lut 2021
1/19/2022	Gru-lut 2021
2/5/2022	Gru-lut 2021
2/28/2022	Gru-lut 2021
3/16/2022	Mar-maj 2022
4/1/2022	Mar-maj 2022
5/7/2022	Mar-maj 2022
5/16/2022	Mar-maj 2022
6/15/2022	Cze-sie 2022
6/26/2022	Cze-sie 2022

date	transaction_quarter
7/9/2022	Cze-sie 2022
7/22/2022	Cze-sie 2022
7/23/2022	Cze-sie 2022
7/27/2022	Cze-sie 2022
8/2/2022	Cze-sie 2022
8/8/2022	Cze-sie 2022
8/19/2022	Cze-sie 2022
9/26/2022	Wrz-lis 2022
10/14/2022	Wrz-lis 2022
10/29/2022	Wrz-lis 2022

W tym przypadku do funkcji `quartername()` przekazano wartość 3 jako argument `first_month_of_year`, w efekcie czego początek roku został przesunięty z 1 stycznia na 1 marca. To spowodowało, że kwartały w tym roku zmieniły się na mar-maj, cze-sie, wrz-lis oraz gru-lut.

Diagram funkcji `quartername()`, przykład z argumentem first_week_day



Transakcja 8203 miała miejsce 8 sierpnia. Funkcja `quartername()` stwierdza, że to był drugi kwartał obejmujący dni od początku czerwca do końca sierpnia. W związku z tym zwraca cze-sie 2022 r.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenia zwracające znacznik czasu końca kwartału, w którym wystąpiły transakcje, są tworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Utwórz następującą miarę:

```
=quartername(date)
```

Tabela wynikowa

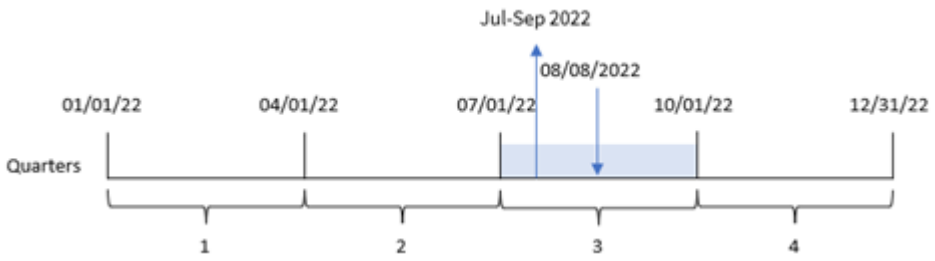
date	=quartername(date)
1/7/2022	Jan-Mar 2022
1/19/2022	Jan-Mar 2022
2/5/2022	Jan-Mar 2022
2/28/2022	Jan-Mar 2022
3/16/2022	Jan-Mar 2022
4/1/2022	Apr-Jun 2022
5/7/2022	Apr-Jun 2022

date	=quartername(date)
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

Miarę „`transaction_quarter`" tworzy się w obiekcie wykresu, używając funkcji `quartername()` i przekazując pole daty `date` jako jej argument.

Funkcja `quartername()` początkowo identyfikuje kwartał, w którym przypada wartość daty. Następnie zwraca wartość pokazującą pierwszy i ostatni miesiąc tego kwartału, jak również roku.

Diagram funkcji `quartername()`, przykład obiektu wykresu



Transakcja 8203 miała miejsca 8 sierpnia 2022 roku. Funkcja `quartername()` stwierdza, że transakcja miała miejsce w trzecim kwartale, więc zwraca lip-wrz 2022 r. Miesiące są ukazane w takim samym formacie, jak zmienna systemowa `MonthNames`.

Przykład 5 – scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „Transactions”.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).

Użytkownik końcowy chciałby otrzymać obiekt wykresu przedstawiający łączną sprzedaż w ujęciu kwartalnym dla transakcji. Można to osiągnąć nawet wtedy, gdy ten wymiar nie jest dostępny w modelu danych, używając funkcji `quartername()` jako wymiaru obliczanego na wykresie.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę.
2. Utwórz wymiar wyliczany, używając następującego wyrażenia:
   `=quartername(date)`
3. Następnie oblicz łączną sprzedaż przy użyciu następującej miary agregacji:
   `=sum(amount)`
4. Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

=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

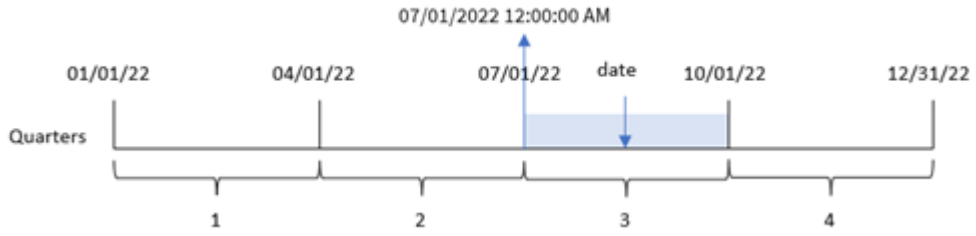
Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu pierwszej milisekundy kwartału zawierającego wartość **date**. Domyślnym formatem wyjściowym będzie format **DateFormat** skonfigurowany w skrypcie.

Składnia:

```
QuarterStart(date[, period_no[, first_month_of_year]])
```

Typ zwracanych danych: dual

Schemat funkcji `quarterstart()`



Funkcja `quarterstart()` określa, w którym kwartale wypada `date`. Następnie zwraca znacznik czasu, w formacie daty, pierwszej milisekundy pierwszego miesiąca tego kwartału.

Argumenty

Argument	Opis
date	Data lub znacznik czasu do oszacowania.
period_no	**period_no** jest liczbą całkowitą, gdzie 0 oznacza kwartał, który zawiera datę **date**. Wartości ujemne parametru **period_no** oznaczają kwartały poprzednie, a wartości dodatnie – kwartały następne.
first_month_of_year	Jeśli użytkownik zamierza korzystać z lat (obrotowych), które nie zaczynają się w styczniu, powinien wskazać wartość od 2 do 12 jako parametr **first_month_of_year**.

Kiedy używać

Funkcja `quarterstart()` jest zwykle używana jako część wyrażenia, gdy użytkownik chce, by w obliczeniach użyto ułamka kwartału, który upłynął do tej pory. Przy jej użyciu użytkownik może na przykład obliczyć odsetki narosłe w ciągu kwartału do określonej daty.

Przykłady funkcji

Przykład	Wynik
`quarterstart('10/29/2005')`	Zwraca wartość 10/01/2005.
`quarterstart('10/29/2005', -1 )`	Zwraca wartość 07/01/2005.
`quarterstart('10/29/2005', 0, 3)`	Zwraca wartość 09/01/2005.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „Transactions”.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie pola, `start_of_quarter`, zwracającego znacznik czasu początku kwartału, w którym zostały zawarte transakcje.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `start_of_quarter`

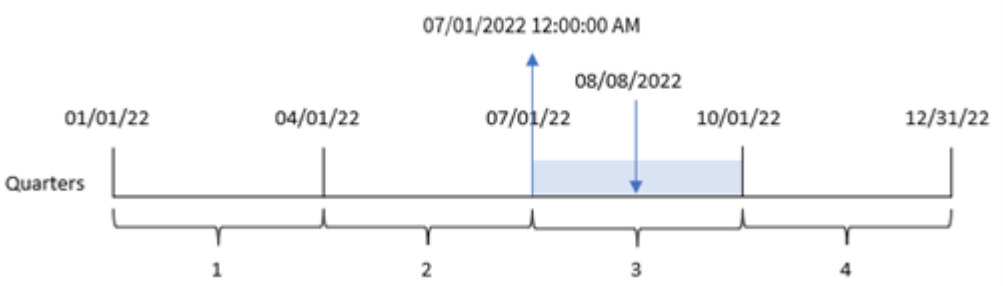
- start_of_quarter_timestamp

Tabela wynikowa

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

Pole `start_of_quarter` jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `quarterstart()` i przekazanie pola daty jako jej argumentu. Funkcja `uarterstart()` początkowo identyfikuje kwartał, w którym przypada wartość daty. Następnie zwraca znacznik czasu pierwszej milisekundy tego kwartału.

Diagram funkcji `quarterstart()`, przykład bez dodatkowych argumentów



Transakcja 8203 miała miejsce 8 sierpnia. Funkcja `quarterstart()` stwierdza, że transakcja miała miejsce w trzecim kwartale oraz zwraca pierwszą milisekundę tego kwartału – 1 lipca, godz. 00:00:00.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola `previous_quarter_start` zwracającego znacznik czasu początku kwartału, przed tym, w którym miała miejsce transakcja.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `previous_quarter_start`
- `previous_quarter_start_timestamp`

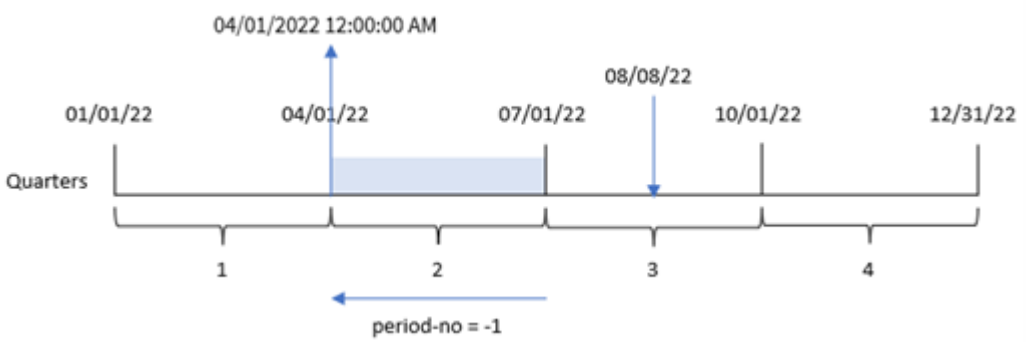
Tabela wynikowa

date	previous_quarter_start	previous_quarter_start_timestamp
1/7/2022	10/01/2021	10/1/2021 12:00:00 AM
1/19/2022	10/01/2021	10/1/2021 12:00:00 AM
2/5/2022	10/01/2021	10/1/2021 12:00:00 AM
2/28/2022	10/01/2021	10/1/2021 12:00:00 AM
3/16/2022	10/01/2021	10/1/2021 12:00:00 AM
4/1/2022	01/01/2022	1/1/2022 12:00:00 AM
5/7/2022	01/01/2022	1/1/2022 12:00:00 AM
5/16/2022	01/01/2022	1/1/2022 12:00:00 AM
6/15/2022	01/01/2022	1/1/2022 12:00:00 AM
6/26/2022	01/01/2022	1/1/2022 12:00:00 AM
7/9/2022	04/01/2022	4/1/2021 12:00:00 AM
7/22/2022	04/01/2022	4/1/2021 12:00:00 AM
7/23/2022	04/01/2022	4/1/2021 12:00:00 AM
7/27/2022	04/01/2022	4/1/2021 12:00:00 AM
8/2/2022	04/01/2022	4/1/2021 12:00:00 AM
8/8/2022	04/01/2022	4/1/2021 12:00:00 AM

date	previous_quarter_start	previous_quarter_start_timestamp
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

W tym przypadku, ponieważ wartości `period_no` -1 użyto jako argumentu przesunięcia w funkcji `quarterstart()`, funkcja najpierw identyfikuje kwartał, w którym odbywają się transakcje. Następnie przesuwa zakres o kwartał wstecz i identyfikuje pierwszą milisekundę tego kwartału.

Diagram funkcji `quarterstart()`, przykład z argumentem period_no



Transakcja 8203 miała miejsce 8 sierpnia. Funkcja `quarterstart()` stwierdza, że kwartał przed datą transakcji obejmował okres od 1 kwietnia do 30 czerwca. Następnie zwraca pierwszą milisekundę tego kwartału – 1 kwietnia o godz. 00:00:00.

Przykład 3 – first_month_of_year

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie. Jednak w tym przykładzie musimy również ustawić 1 marca jako pierwszy miesiąc roku podatkowego.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `start_of_quarter`
- `start_of_quarter_timestamp`

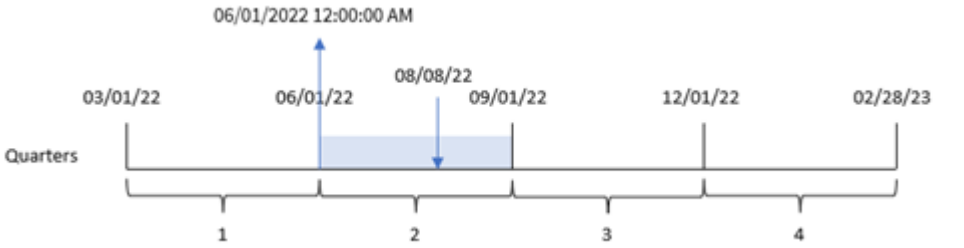
Tabela wynikowa

date	start_of_quarter	start_of_quarter_timestamp
1/7/2022	12/01/2021	12/1/2021 12:00:00 AM
1/19/2022	12/01/2021	12/1/2021 12:00:00 AM
2/5/2022	12/01/2021	12/1/2021 12:00:00 AM
2/28/2022	12/01/2021	12/1/2021 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
4/1/2022	03/01/2022	3/1/2022 12:00:00 AM
5/7/2022	03/01/2022	3/1/2022 12:00:00 AM
5/16/2022	03/01/2022	3/1/2022 12:00:00 AM

date	start_of_quarter	start_of_quarter_timestamp
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

W tym przypadku, ponieważ jako argument `first_month_of_year` funkcji `quarterstart()` przekazano wartość 3, początek roku został przesunięty z 1 stycznia na 1 marca.

Diagram funkcji `quarterstart()`, przykład z użyciem argumentu first_month_of_year



Transakcja 8203 miała miejsce 8 sierpnia. Ponieważ rok zaczyna się 1 marca, kwartały w nim obejmują miesiące marzec-maj, czerwiec-sierpień, wrzesień-listopad oraz grudzień-luty. Funkcja `quarterstart()` stwierdza, że transakcja miała miejsce w kwartale od czerwca do sierpnia i zwraca pierwszą milisekundę tego kwartału – 1 czerwca, godz. 00:00:00.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenia zwracające znacznik czasu końca kwartału, w którym wystąpiły transakcje, są tworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Dodaj następujące miary:

- `=quarterstart(date)`
- `=timestamp(quarterstart(date))`

Tabela wynikowa

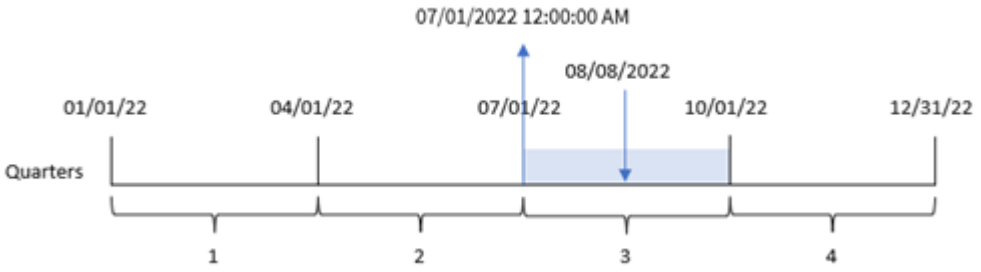
date	=quarterstart(date)	=timestamp(quarterstart(date))
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM

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

Miarę „`start_of_quarter`” tworzy się w obiekcie wykresu, używając funkcji `quarterstart()` i przekazując pole daty `date` jako jej argument.

Funkcja `quarterstart()` identyfikuje, w którym kwartale przypada określona data, i zwraca znacznik czasu pierwszej milisekundy danego kwartału.

Diagram funkcji `quarterstart()`, przykład obiektu wykresu



Transakcja 8203 miała miejsce 8 sierpnia. Funkcja `quarterstart()` stwierdza, że transakcja miała miejsce w trzecim kwartale oraz zwraca pierwszą milisekundę tego kwartału. Ta zwrócona wartość to 1 lipca, godz. 00:00:00.

Przykład 5 – scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw sald kredytów, który jest ładowany do tabeli o nazwie `Loans`.
- Dane zawierające identyfikatory kredytów, saldo na początku kwartału i prostą stopę procentową naliczaną od każdego kredytu rocznie.

Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał według identyfikatora pożyczki bieżące odsetki naliczone od każdej pożyczki w bieżącym kwartale.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:
   - `loan_id`
   - `start_balance`
2. Następnie utwórz tę miarę, aby obliczyć narosłe odsetki:
   `=start_balance*(rate*(today(1)-quarterstart(today(1)))/365)`
3. Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

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

Używając dzisiejszej daty jako jedynego argumentu, funkcja `quarterstart()` zwraca datę początkową bieżącego roku. Odejmując ten wynik od bieżącej daty, wyrażenie zwraca liczbę dni, które upłynęły do tej pory w tym kwartale.

Wartość ta jest następnie mnożona przez stopę procentową i dzielona przez 365, aby uzyskać wysokość odsetek nagromadzonych do tego momentu tygodnia. Wynik jest następnie mnożony przez saldo początkowe pożyczki, aby zwrócić odsetki naliczone do tej pory w tym kwartale.

second

Ta funkcja zwraca liczbę całkowitą reprezentującą sekundy, gdy ułamek **expression** jest interpretowany jako czas zgodnie ze standardową interpretacją liczb.

Składnia:

```
second (expression)
```

Typ zwracanych danych: integer

Kiedy używać

Funkcja `second()` jest przydatna, gdy chcesz porównać agregacje według sekundy. Za pomocą tej funkcji można na przykład sprawdzić rozkład liczb aktywności w ujęciu sekundowym.

Te wymiary można utworzyć w skrypcie ładowania, używając tej funkcji w celu utworzenia pola w tabeli kalendarza głównego lub bezpośrednio na wykresie jako wymiaru obliczanego.

Przykłady funkcji

Przykład	Wynik
`second( '09:14:36' )`	zwraca 36
`second( '0.5555' )`	zwraca 55 (ponieważ 0,5555 = 13:19:55)

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – zmienna

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zbiór danych zawierający transakcje wg znacznika czasu, który jest ładowany do tabeli o nazwie `Transactions`.
- Została użyta domyślna zmienna systemowa `TimeStamp` (`M/D/YYYY h:mm:ss[.fff] TT`).
- Utworzenie pola `second` obliczającego czas dokonywania zakupów.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- date
- second

Tabela wynikowa

date	drugie
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

Wartości w polu second są tworzone za pomocą funkcji second() i przez przekazanie daty jako wyrażenia w poprzedniej instrukcji ładowania.

Przykład 2 – obiekt wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie. Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Wartości second są obliczane przez miarę w obiekcie wykresu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:`date`.

Utwórz następującą miarę:

```
=second(date)
```

Tabela wynikowa

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

Wartości dla `second` są tworzone za pomocą funkcji `second()` i przez przekazanie daty jako wyrażenia w mierze dla obiektu wykresu.

Przykład 3 – Scenariusz

Skrypt ładowania i wyrażenia wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych znaczników czasu, który reprezentuje ruch do określonej strony internetowej sprzedaży biletów na festiwal. Te znaczniki i odpowiedni `id` zostają załadowane do tabeli o nazwie `Web_Traffic`.
- Używana jest zmienna systemowa `TimeStamp M/D/YYYY h:mm:ss[.fff] TT`.

W tym przypadku było 10000 biletów, które wystawiono na sprzedaż 20 maja 2021 r. o godz. 9:00. Minutę później bilety były wyprzedane.

Użytkownik chciałby otrzymać obiekt wykresu pokazujący liczbę wizyt w witrynie w ujęciu sekundowym.

Skrypt ładowania

```
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;
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę.
2. Następnie utwórz wymiary wyliczany za pomocą następującego wyrażenia:
   `=second(timestamp)`
3. Utwórz miarę agregacji, aby obliczyć sumę wejść:
   `=count(id)`

Tabela wyników będzie wyglądać podobnie do poniższej tabeli, tylko będzie zawierała inne wartości dla miary agregacji:

Tabela wynikowa

second(timestamp)	=count(id)
0	150
1	184

second(timestamp)	=count(id)
2	163
3	178
4	179
5	158
6	177
7	169
8	149
9	186
10	169
11	179
12	186
13	182
14	180
15	153
16	191
17	203
18	158
19	159
20	163
Jeszcze 39 wierszy	

setdateyear

Ta funkcja przyjmuje jako dane wejściowe wartości z pól **timestamp** i **year** i aktualizuje pole **timestamp** wartością **year** określoną w danych wejściowych.

Składnia:

```
setdateyear (timestamp, year)
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
timestamp	Standardowy znacznik czasu Qlik Sense (często jest to tylko data).
year	Rok w zapisie czterocyfrowym.

Przykłady i wyniki:
W tych przykładach używany jest format daty **DD/MM/YYYY**. Format daty jest określony w instrukcji **SET DateFormat** u góry skryptu ładowania danych. Format zastosowany w przykładach można zmienić, aby dostosować go do konkretnych potrzeb.

Przykłady skryptów

Przykład	Wynik
`setdateyear ('29/10/2005', 2013)`	Zwraca wartość 29/10/2013
`setdateyear ('29/10/2005 04:26:14', 2013)`	Zwraca wartość '29/10/2013 04:26:14' Aby wyświetlić w wizualizacji część znacznika czasu odpowiadającą godzinie, należy ustawić formatowanie liczb na Date i wybrać wartość formatowania, która spowoduje wyświetlenie wartości godziny.

Przykład:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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
];
```

Otrzymana tabela zawiera pierwotne daty i kolumnę, w której rok musi być ustawiony na wartość 2013.

Tabela wynikowa

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

Ta funkcja przyjmuje jako dane wejściowe wartości z pól **timestamp**, **month** i **year** i aktualizuje pole **timestamp** wartościami **year** i **month** określonymi w danych wejściowych. .

Składnia:

```
SetDateYearMonth (timestamp, year, month)
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
timestamp	Standardowy znacznik czasu Qlik Sense (często jest to tylko data).
year	Rok w zapisie czterocyfrowym.
month	Miesiąc w zapisie jedno- lub dwucyfrowym.

Przykłady i wyniki:
W tych przykładach używany jest format daty **DD/MM/YYYY**. Format daty jest określony w instrukcji **SET DateFormat** u góry skryptu ładowania danych. Format zastosowany w przykładach można zmienić, aby dostosować go do konkretnych potrzeb.

Przykłady skryptów

Przykład	Wynik
`setdateyearmonth ('29/10/2005', 2013, 3)`	Zwraca wartość 29/03/2013
`setdateyearmonth ('29/10/2005 04:26:14', 2013, 3)`	Zwraca wartość '29/03/2013 04:26:14' Aby wyświetlić w wizualizacji część znacznika czasu odpowiadającą godzinie, należy ustawić formatowanie liczb na Date i wybrać wartość formatowania, która spowoduje wyświetlenie wartości godziny.

Przykład:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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
];
```

Otrzymana tabela zawiera pierwotne daty i kolumnę, w której rok musi być ustawiony na wartość 2013.

Tabela wynikowa

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

testdates	NewYearMonth
7/7/2014	7/3/2013
4/8/2014	4/3/2013

timezone

Ta funkcja zwraca strefę czasową zdefiniowaną na komputerze, na którym działa aparat Qlik.

Składnia:

```
TimeZone( )
```

Typ zwracanych danych: dual

Przykład:

```
timezone( )
```

Jeśli chcesz zobaczyć inną strefę czasową w mierze w swojej aplikacji, możesz użyć funkcji `localtime()` w mierze.

today

Ta funkcja zwraca bieżącą datę. Funkcja zwraca wartości w formacie zmiennej systemowej `DateFormat`.

Składnia:

```
today([ timer_mode])
```

Typ zwracanych danych: dual

Funkcji `today()` można użyć w skrypcie ładowania lub w obiektach wykresów.

Wartością domyślną `timer_mode` jest 1.

Argumenty

Argument	Opis
timer_mode	Może przyjmować następujące wartości: 0 (dzień ostatnio zakończonego ładowania danych) 1 (dzień w momencie wywołania funkcji) 2 (dzień w momencie otwarcia aplikacji) *W przypadku użycia funkcji w skrypcie ładowania **timer_mode=0** da wynik będący dniem ostatnio zakończonego ładowania danych, z kolei **timer_mode=1** zwróci dzień wywołania funkcji w bieżącym ładowaniu danych.*

Przykłady funkcji

timer_mode value	Wynik w przypadku użycia w skrypcie ładowania	Wynik w przypadku użycia w obiekcie wykresu
0	Zwraca datę, w formacie zmiennej systemowej `DateFormat`, ostatniego udanego przeładowania danych przed ostatnim załadowaniem danych.	Zwraca datę, w formacie zmiennej systemowej `DateFormat`, ostatniego przeładowania danych.
1	Zwraca datę, w formacie zmiennej systemowej `DateFormat`, ostatniego przeładowania danych.	Zwraca datę, w formacie zmiennej systemowej `DateFormat`, wywołania funkcji.
2	Zwraca datę, w formacie zmiennej systemowej `DateFormat`, początku sesji użytkownika w aplikacji. Zostanie on zaktualizowany dopiero po ponownym załadowaniu skryptu przez użytkownika.	Zwraca datę, w formacie zmiennej systemowej `DateFormat`, początku sesji użytkownika w aplikacji. Będzie on odświeżany na początku sesji lub po ponownym załadowaniu danych do aplikacji.

Kiedy używać

Funkcja `today()` jest używana jako element wyrażenia. Przy jej użyciu można na przykład obliczyć odsetki narosłe w ciągu miesiąca do bieżącej daty.

Poniższa tabela zawiera objaśnienie wyniku zwracanego przez funkcję `today()` dla różnych wartości argumentu `timer_mode`:

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – generowanie obiektów przy użyciu skryptu ładowania

Skrypt ładowania i wyniki

Przegląd

Poniższy przykład tworzy trzy zmienne za pomocą funkcji `today()`. Każda zmienna używa jednej z opcji `timer_mode` w celu zademonstrowania tego efektu.

Aby zademonstrować przeznaczenie zmiennych, przeładuj skrypt, a następnie, po 24 godzinach, przeładuj go po raz drugi. Spowoduje to, że zmienne `today(0)` i `today(1)` będą miały różne wartości, co będzie stanowiło właściwą demonstrację ich przeznaczenia.

Skrypt ładowania

```
LET vPreviousDataLoad = today(0);
LET vCurrentDataLoad = today(1);
LET vApplicationOpened = today(2);
```

Wyniki

Gdy dane zostaną załadowane po raz drugi, utwórz trzy pola tekstowe, korzystając z poniższych wskazówek.

Najpierw utwórz pole tekstowe dla danych, które zostały wcześniej załadowane.

Wykonaj następujące czynności:

1. Używając obiektu wykresu **Tekst i grafika**, utwórz pole tekstowe.
2. Dodaj następującą miarę do obiektu:
   `=vPreviousDataLoad`
3. W sekcji **Wygląd** wybierz **Show titles** i dodaj do obiektu tytuł „Previous Reload Time”.

Następnie utwórz pole tekstowe dla danych, które są aktualnie ładowane.

Wykonaj następujące czynności:

1. Używając obiektu wykresu **Tekst i grafika**, utwórz pole tekstowe.
2. Dodaj następującą miarę do obiektu:
   `=vCurrentDataLoad`
3. W sekcji **Wygląd** wybierz **Show titles** i dodaj do obiektu tytuł „Current Reload Time”.

Utwórz ostatnie pole tekstowe, aby pokazać, kiedy sesja użytkownika w aplikacji została rozpoczęta.

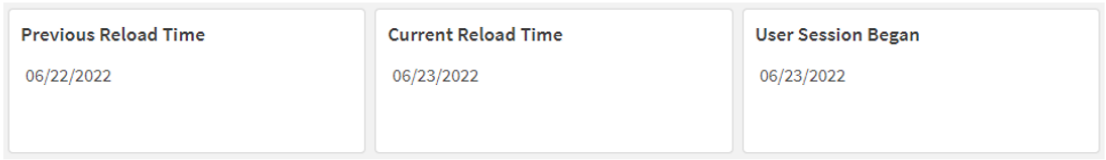
Wykonaj następujące czynności:

1. Używając obiektu wykresu **Tekst i grafika**, utwórz pole tekstowe.
2. Dodaj następującą miarę do obiektu:

```
=vApplicationOpened
```

3. W sekcji **Wygląd** wybierz **Show titles** i dodaj do obiektu tytuł „User Session Started”.

Diagram zmiennych utworzonych przy użyciu funkcji `today()` w skrypcie ładowania



Powyższy obraz pokazuje przykładowe wartości dla każdej z utworzonych zmiennych. Na przykład, te wartości mogą być następujące:

- Czas poprzedniego przeładowania: 22.06.2022 r.
- Czas bieżącego przeładowania: 23.06.2022 r.
- Początek sesji użytkownika: 23.06.2022 r.

Przykład 2 – generowanie obiektów bez skryptu ładowania

Skrypt ładowania i wyrażenie wykresu

Przegląd

Poniższy przykład tworzy trzy obiekty wykresu za pomocą funkcji `today()`. Każdy obiekt wykresu używa jednej z opcji `timer_mode` w celu zademonstrowania tego efektu.

W tym przykładzie nie ma skryptu ładowania.

Wyniki

Po załadowaniu danych drugi raz utwórz trzy pola tekstowe.

Najpierw utwórz pole tekstowe dla ostatniego przeładowania danych.

Wykonaj następujące czynności:

1. Używając obiektu wykresu **Tekst i grafika**, utwórz pole tekstowe.
2. Dodaj następującą miarę:
   `=today(0)`
3. W sekcji **Wygląd** wybierz **Show titles** i dodaj do obiektu tytuł „Ostatnie przeładowanie danych”.

Następnie utwórz pole tekstowe, aby pokazać bieżący czas.

Wykonaj następujące czynności:

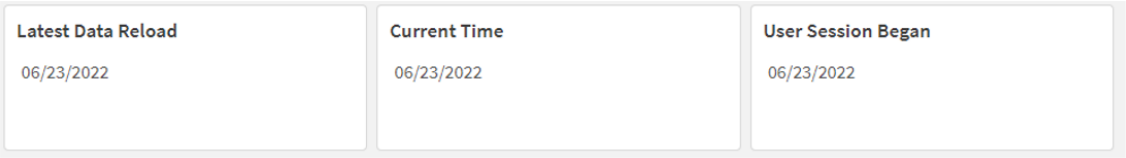
1. Używając obiektu wykresu **Tekst i grafika**, utwórz pole tekstowe.
2. Dodaj następującą miarę:
   `=today(1)`
3. W sekcji **Wygląd** wybierz **Show titles** i dodaj do obiektu tytuł „Bieżący czas”.

Utwórz ostatnie pole tekstowe, aby pokazać, kiedy sesja użytkownika w aplikacji została rozpoczęta.

Wykonaj następujące czynności:

1. Używając obiektu wykresu **Tekst i grafika**, utwórz pole tekstowe.
2. Dodaj następującą miarę:
   `=today(2)`
3. W sekcji **Wygląd** wybierz **Show titles** i dodaj do obiektu tytuł „Sesja użytkownika rozpoczęta”.

Diagram obiektów utworzonych przy użyciu funkcji `today()` bez skryptu ładowania



Powyższy obraz pokazuje przykładowe wartości dla każdego z utworzonych obiektów. Na przykład, te wartości mogą być następujące:

- Ostatnie przeładowanie danych: 23.06.2022 r.
- Bieżący czas: 23.06.2022 r.
- Początek sesji użytkownika: 23.06.2022 r.

Obiekt wykresu „Ostatnie przeładowanie danych” używa argumentu `timer_mode` o wartości 0. Powoduje to zwrócenie znacznika czasu ostatniego udanego przeładowania danych.

Obiekt wykresu „Bieżący czas” używa argumentu `timer_mode` o wartości 1. Powoduje to zwrócenie bieżącego czasu według zegara systemowego. Jeśli arkusz lub obiekt zostanie odświeżony, ta wartość zostanie zaktualizowana.

Obiekt wykresu „Początek sesji użytkownika” używa argumentu `timer_mode` o wartości 2. Powoduje to zwrócenie znacznika czasu otwarcia aplikacji i rozpoczęcia sesji użytkownika.

Przykład 3 – Scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw sald kredytów, który jest ładowany do tabeli o nazwie `Loans`.
- Tabela danych z polami identyfikatora kredytu, salda na początku miesiąca i prostej stopy procentowej naliczanej od każdego kredytu rocznie.

Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał według identyfikatora pożyczki bieżące odsetki naliczone od każdej pożyczki w bieżącym miesiącu. Mimo że aplikacja jest przeładowywana tylko raz na tydzień, użytkownik chciałby, aby wyniki były odświeżane przy każdym odświeżeniu obiektu lub aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę.
2. Dodaj następujące pola jako wymiary:
    - `loan_id`
    - `start_balance`
3. Następnie utwórz miarę, aby obliczyć narosłe odsetki:
   `=start_balance*(rate*(today(1)-monthstart(today(1)))/365)`
4. Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

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

Funkcja `monthstart()` zwraca datę początkową bieżącego miesiąca, używając funkcji `today()` zwracającej bieżącą datę jako jedynego argumentu. Odejmując ten wynik od bieżącej daty, uzyskanej za pomocą funkcji `today()`, wyrażenie to zwraca liczbę dni, które upłynęły do tej pory w tym miesiącu.

Wartość ta jest następnie mnożona przez stopę procentową i dzielona przez 365, aby uzyskać wysokość odsetek nagromadzonych do tego momentu tygodnia. Wynik jest następnie mnożony przez saldo początkowe pożyczki, aby zwrócić odsetki naliczone do tej pory w tym miesiącu.

Ponieważ w wyrażeniu funkcjom `today()` jako argument `timer_mode` przekazano wartość 1, przy każdym odświeżeniu obiektu wykresu (przez otwarcie aplikacji, odświeżenie strony, przejście między arkuszami itd.) będzie zwracana bieżąca data i wyniki będą odpowiednio odświeżane.

UTC

Zwraca bieżącą wartość Coordinated Universal Time.

Składnia:

```
UTC( )
```

Typ zwracanych danych: podwójny

Przykład:

```
utc( )
```

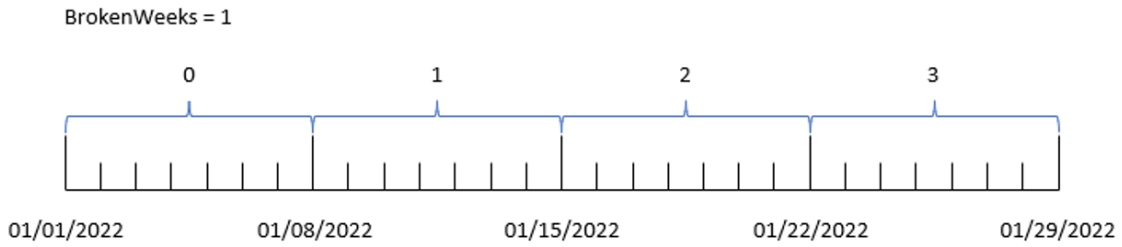
week

Ta funkcja zwraca liczbę całkowitą reprezentującą numer tygodnia zgodnie z normą ISO 8601. Numer miesiąca jest obliczany na podstawie interpretacji daty z wyrażenia zgodnie ze standardową interpretacją liczb.

Składnia:

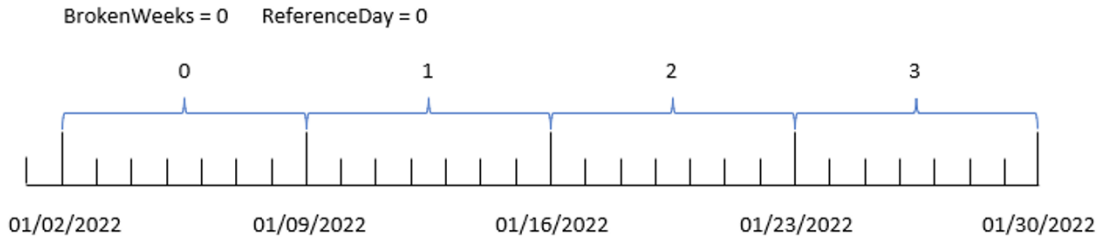
```
week(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

Przykładowy diagram funkcji week() z podzielonymi tygodniami



Numerowanie tygodni zaczyna się 1 stycznia (ponieważ domyślnie Qlik Sense ustawiono, aby używać dzielonych tygodni). Pierwszy tydzień kończy się w dniu poprzedzającym datę określaną przez zmienną systemową `FirstWeekDay` niezależnie od tego, ile dni upłynęło w tym tygodniu. Zmienną systemową `FirstWeekDay` można zastąpić w funkcji `week()` argumentem `first_week_day`.

Przykładowy diagram funkcji week() z niedzielonymi tygodniami i argumentem ReferenceDay=0



Funkcja `week()` także umożliwia określenie, czy mają być używane tygodnie dzielone czy niedzielone. Służy do tego argument `broken_weeks`. Jeśli zostaną wybrane tygodnie dzielone, tydzień 1 musi zawierać określoną liczbę dni w styczniu zgodnie z definicją w zmiennej `ReferenceDay`. W związku z tym tydzień 1 może zaczynać się w grudniu lub ewentualnie tygodnie 52 lub 53 mogą być kontynuowane w styczniu. Argument `reference_day` umożliwia funkcji przesłonięcie zmiennej systemowej `ReferenceDay`.

W odróżnieniu od funkcji `weekname()`, funkcja `week()` nie zwraca wartości roku. To umożliwia tworzenie agregacji porównujących tygodnie między latami.

Ta funkcja przyjmuje cztery argumenty.

Argument 1: timestamp

Data traktowana jako znacznik czasu lub wyrażenie, którego wynikiem jest znacznik czasu, przeznaczone do konwersji, na przykład '2012-10-12'.

Argument 2: first_week_day

Jeśli wartość **first_week_day** nie zostanie określona, jako pierwszy dzień tygodnia zostanie zastosowana wartość zmiennej **FirstWeekDay**.

Aby użyć innego dnia jako pierwszego dnia tygodnia, ustaw **first_week_day** na wartość:

- 0 dla poniedziałku
- 1 dla wtorku
- 2 dla środy
- 3 dla czwartku
- 4 dla piątku
- 5 dla soboty
- 6 dla niedzieli

Liczba całkowita zwrócona przez funkcję będzie teraz używać pierwszego dnia tygodnia ustawionego za pomocą **first_week_day**.

Argument 3: broken_weeks

Jeśli wartość **broken_weeks** nie zostanie określona, wówczas wartość zmiennej **BrokenWeeks** zostanie zastosowana w celu zdefiniowania, czy tygodnie są dzielone, czy nie.

Domyślnie w funkcjach Qlik Sense stosuje się tygodnie niedzielone. Oznacza to, że:

- W niektórych latach tydzień 1 zaczyna się w grudniu, a w innych tydzień 52 lub 53 przeciąga się na styczeń.
- Co najmniej cztery dni tygodnia 1 przypadają w styczniu.

Alternatywnym wyjściem jest użycie podzielonych tygodni.

- Tydzień 52 ani 53 nie przeciąga się na styczeń.
- Tydzień 1 zaczyna się 1 stycznia i w większości przypadków nie jest pełnym tygodniem.

Można użyć następujących wartości:

- 0 (=użyj niepodzielonych tygodni)
- 1 (=użyj podzielonych tygodni)

Argument 4: reference_day

Jeśli wartość **reference_day** nie zostanie określona, wówczas w celu zdefiniowania dnia w styczniu, który zostanie ustawiony jako dzień odniesienia do zdefiniowania tygodnia 1, zostanie użyta wartość zmiennej **ReferenceDay**. Domyślnie w funkcjach programu Qlik Sense za dzień odniesienia uznaje się 4. Oznacza to, że pierwszy tydzień musi zawierać datę 4 stycznia lub, ujmując rzecz inaczej, pierwszy tydzień musi zawsze zawierać co najmniej 4 dni w styczniu.

Do ustawiania innych dni referencyjnych można użyć następujących wartości:

- 1 (= 1 stycznia)
- 2 (= 2 stycznia)
- 3 (= 3 stycznia)
- 4 (= 4 stycznia)
- 5 (= 5 stycznia)

- 6 (= 6 stycznia)
- 7 (= 7 stycznia)

Kiedy używać

Funkcja `The week()` jest przydatna, gdy chcesz porównać agregacje według tygodni. Można jej na przykład użyć, aby wyświetlić całkowitą sprzedaż produktów w ujęciu tygodniowym. Funkcja `week()` jest wybierana zamiast funkcji `weekname()`, gdy użytkownik nie chce, aby w obliczeniach były wykorzystywane zmienne systemowe aplikacji `BrokenWeeks`, `FirstWeekDay` lub `ReferenceDay`.

Ponadto funkcja `week()` jest wybierana, gdy chcemy dokonywać porównań między latami. Przy użyciu funkcji `week()` użytkownik może utworzyć własną kombinację tych zmiennych do użycia w przypadkach, w których używana jest ta funkcja.

Te wymiary można utworzyć w skrypcie ładowania, używając tej funkcji w celu utworzenia pola w tabeli kalendarza głównego lub bezpośrednio na wykresie jako wymiaru obliczanego.

Przykłady funkcji

Przykład	Wynik
`week( '10/12/2012')`	Zwraca 41.
`week( '35648')`	Zwraca 32, ponieważ 35648 = 06.08.1997.
`week('10/12/2012', 0, 1)`	zwraca 42.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Domyślne zmienne systemowe

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za ostatni tydzień 2021 r., który jest ładowany do tabeli o nazwie `Transactions`.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie pola, `week_number`, które zwraca rok i numer tygodnia, w którym zawarto transakcję.
- Utworzenie pola o nazwie `week_day`, pokazującego wartość dnia tygodnia każdej daty transakcji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`

- week_day
- week_number

Tabela wynikowa

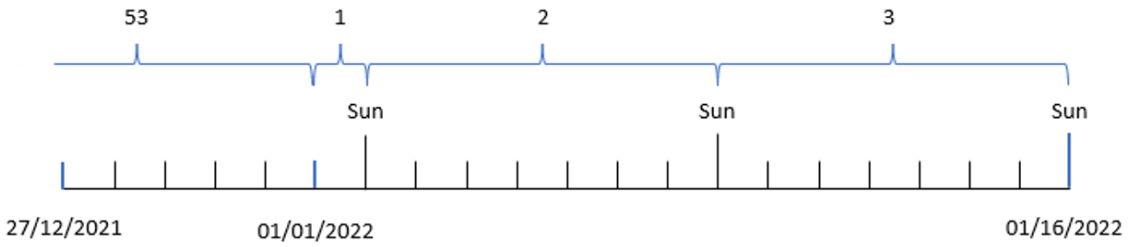
id	date	week_day	week_number
8183	12/27/2021	Pon	53
8184	12/28/2021	Wto	53
8185	12/29/2021	Śro	53
8186	12/30/2021	Thu	53
8187	12/31/2021	Fri	53
8188	01/01/2022	Sat	1
8189	01/02/2022	Nie	2
8190	01/03/2022	Pon	2
8191	01/04/2022	Wto	2
8192	01/05/2022	Śro	2
8193	01/06/2022	Thu	2
8194	01/07/2022	Fri	2
8195	01/08/2022	Sat	2
8196	01/09/2022	Nie	3
8197	01/10/2022	Pon	3
8198	01/11/2022	Wto	3
8199	01/12/2022	Śro	3
8200	01/13/2022	Thu	3
8201	01/14/2022	Fri	3

Pole `week_number` jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `week()` i przekazanie pola `date` jako argumentu funkcji.

Do funkcji nie są przekazywane żadne inne parametry, w związku z czym obowiązują następujące domyślne zmienne wpływające na funkcję `week()`:

- `BrokenWeeks`: Numerowanie tygodni zaczyna się od 1 stycznia
- `FirstWeekDay`: Pierwszy dzień tygodnia to niedziela

Diagram funkcji week() używającej domyślnych zmiennych systemowych



Ponieważ aplikacja używa domyślnej zmiennej systemowej `BrokenWeeks`, tydzień pierwszy zaczyna się 1 stycznia, czyli w sobotę.

Ze względu na domyślną zmienną systemową `FirstWeekDay` tygodnie zaczynają się w niedzielę. Pierwsza niedziela po 1 stycznia przypada 2 stycznia, więc tego dnia zaczyna się tydzień drugi.

Przykład 2 – first_week_day

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Utworzenie pola, `week_number`, które zwraca rok i numer tygodnia, w którym zawarto transakcję.
- Utworzenie pola o nazwie `week_day`, pokazującego wartość dnia tygodnia każdej daty transakcji.

W tym przykładzie chcemy ustawić początek tygodnia roboczego na wtorek.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `week_day`
- `week_number`

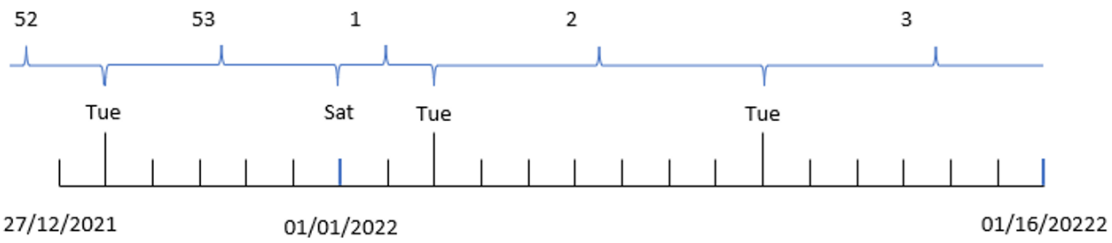
Tabela wynikowa

id	date	week_day	week_number
8183	12/27/2021	Pon	52
8184	12/28/2021	Wto	53
8185	12/29/2021	Śro	53
8186	12/30/2021	Thu	53
8187	12/31/2021	Fri	53
8188	01/01/2022	Sat	1
8189	01/02/2022	Nie	1
8190	01/03/2022	Pon	1
8191	01/04/2022	Wto	2
8192	01/05/2022	Śro	2
8193	01/06/2022	Thu	2
8194	01/07/2022	Fri	2

id	date	week_day	week_number
8195	01/08/2022	Sat	2
8196	01/09/2022	Nie	2
8197	01/10/2022	Pon	2
8198	01/11/2022	Wto	3
8199	01/12/2022	Śro	3
8200	01/13/2022	Thu	3
8201	01/14/2022	Fri	3

Aplikacja nadal używa podzielonych tygodni. Jednak argument `first_week_day` w funkcji `week()` został ustawiony na 1. To ustawia wtorek jako pierwszy dzień tygodnia.

Diagram funkcji `week()`, przykład z argumentem first_week_day



Aplikacja używa domyślnej zmiennej systemowej `BrokenWeeks`, więc tydzień pierwszy zaczyna się 1 stycznia, czyli w sobotę.

Argument `first_week_day` funkcji `week()` ustawia pierwszy dzień tygodnia na wtorek. W efekcie tydzień 53 zaczyna się 28 grudnia 2021 r.

Ponieważ jednak funkcja nadal używa dzielonych tygodni, tydzień pierwszy będzie miał tylko dwa dni, ponieważ pierwszy wtorek po 1 stycznia wypada 3 stycznia.

Przykład 3 – unbroken_weeks

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

W tym przykładzie używamy niepodzielonych tygodni.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `week_day`
- `week_number`

Diagram funkcji week(), przykład obiektu wykresu

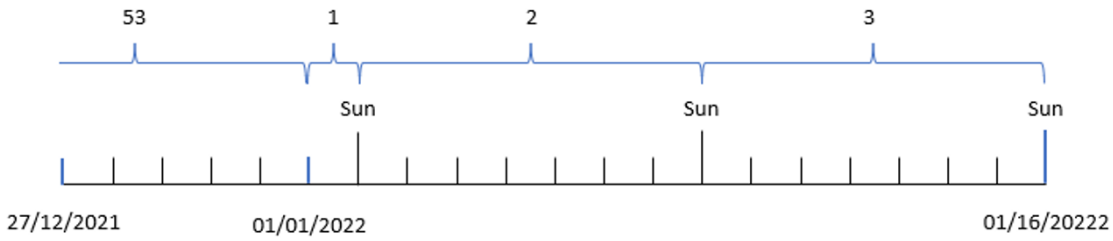


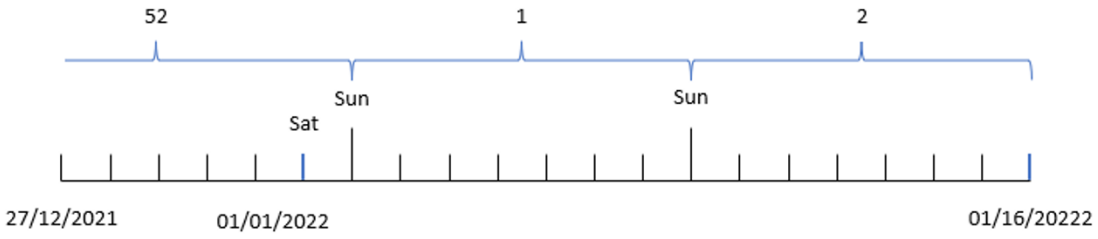
Tabela wynikowa

id	date	week_day	week_number
8183	12/27/2021	Pon	52
8184	12/28/2021	Wto	52
8185	12/29/2021	Śro	52
8186	12/30/2021	Thu	52
8187	12/31/2021	Fri	52
8188	01/01/2022	Sat	52
8189	01/02/2022	Nie	1
8190	01/03/2022	Pon	1
8191	01/04/2022	Wto	1
8192	01/05/2022	Śro	1
8193	01/06/2022	Thu	1
8194	01/07/2022	Fri	1
8195	01/08/2022	Sat	1
8196	01/09/2022	Nie	2
8197	01/10/2022	Pon	2
8198	01/11/2022	Wto	2
8199	01/12/2022	Śro	2
8200	01/13/2022	Thu	2
8201	01/14/2022	Fri	2

Parametr first_week_date jest ustawiony na 1, co czyni wtorek pierwszym dniem tygodnia. Parametr broken_weeks jest ustawiony na 0, co zmusza funkcję do używania niedzielonych tygodni. Trzeci parametr ustawia argument reference_day na 2.

Parametr `first_week_date` jest ustawiony na 6, co czyni niedzielę pierwszym dniem tygodnia. Parametr `broken_weeks` jest ustawiony na 0, co zmusza funkcję do używania niepodzielonych tygodni.

Diagram funkcji week(), przykład użycia niedzielonych tygodni



Dzięki użyciu niedzielonych tygodni pierwszy tydzień nie musi zaczynać się 1 stycznia. Zamiast tego musi mieć przynajmniej pięć dni. W związku z tym, w zestawie danych, tydzień 52 kończy się w sobotę 1 stycznia 2022 r. Zatem pierwszy tydzień zaczyna się w dniu wskazywanym przez zmienną systemową `FirstWeekDay`, który jest niedzielą 2 stycznia. Ten tydzień kończy się w następną sobotę 8 stycznia.

Przykład 4 – reference_day

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w trzecim przykładzie.
- Utworzenie pola, `week_number`, które zwraca rok i numer tygodnia, w którym zawarto transakcję.
- Utworzenie pola o nazwie `week_day`, pokazującego wartość dnia tygodnia każdej daty transakcji.

Ponadto muszą być spełnione następujące warunki:

- Tydzień roboczy zaczyna się we wtorek.
- Firma używa niepodzielonych tygodni.
- Wartość `reference_day` wynosi 2. Innymi słowy, minimalna liczba dni w styczniu w tygodniu pierwszym wynosi 2.

Skrypt ładowania

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transactions:
    Load
```

```
        *,
        WeekDay(date) as week_day,
        Week(date,1,0,2) as week_number
        ;
Load
*
Inline
[
id,date,amount
8183,12/27/2022,58.27
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `week_day`
- `week_number`

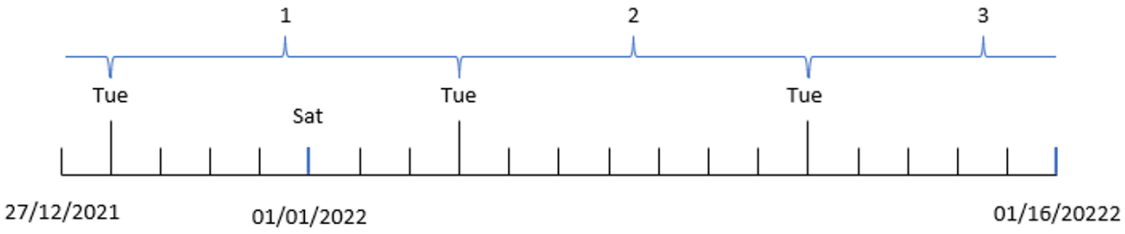
Tabela wynikowa

id	date	week_day	week_number
8183	12/27/2021	Pon	52
8184	12/28/2021	Wto	1
8185	12/29/2021	Śro	1
8186	12/30/2021	Thu	1
8187	12/31/2021	Fri	1
8188	01/01/2022	Sat	1

id	date	week_day	week_number
8189	01/02/2022	Nie	1
8190	01/03/2022	Pon	1
8191	01/04/2022	Wto	2
8192	01/05/2022	Śro	2
8193	01/06/2022	Thu	2
8194	01/07/2022	Fri	2
8195	01/08/2022	Sat	2
8196	01/09/2022	Nie	2
8197	01/10/2022	Pon	2
8198	01/11/2022	Wto	3
8199	01/12/2022	Śro	3
8200	01/13/2022	Thu	3
8201	01/14/2022	Fri	3

Parametr `first_week_date` jest ustawiony na 1, co czyni wtorek pierwszym dniem tygodnia. Parametr `broken_weeks` jest ustawiony na 0, co zmusza funkcję do używania niepodzielonych tygodni. Trzeci parametr ustawia parametr `reference_day` na 2.

Diagram funkcji `week()`, przykład z argumentem reference_day



W przypadku tej funkcji używającej niedzielonych tygodni i argumentu `reference_day` o wartości 2, pierwszy tydzień w styczniu musi mieć tylko dwa dni. Ponieważ pierwszy dzień tygodnia to wtorek, pierwszy tydzień zaczyna się 28 grudnia 2021 r. i kończy się w poniedziałek, 3 stycznia 2022 r.

Przykład 5 – Przykład z użyciem obiektu wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenia zwracające numer tygodnia zostały utworzone jako miara w obiekcie wykresu.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę.
2. Dodaj następujące pola jako wymiary:
    - `id`
    - `date`
3. Następnie utwórz następującą miarę:
   `=week (date)`
4. Utwórz miarę , `week_day`, aby pokazać wartość dnia tygodnia każdej daty transakcji:
   `=weekday(date)`

Tabela wynikowa

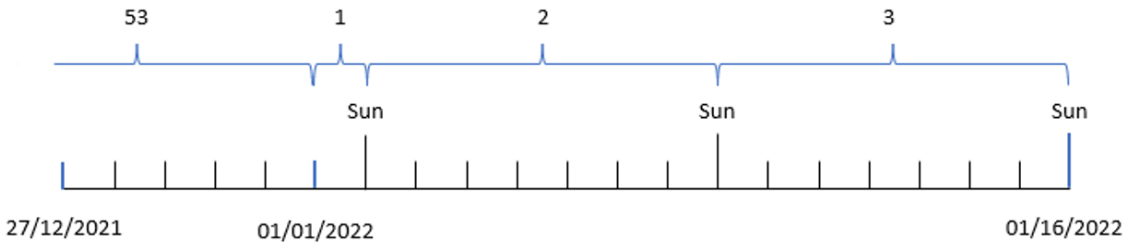
id	date	=week(date)	=weekday(date)
8183	12/27/2021	53	Pon
8184	12/28/2021	53	Wto
8185	12/29/2021	53	Śro
8186	12/30/2021	53	Thu
8187	12/31/2021	53	Fri
8188	01/01/2022	1	Sat
8189	01/02/2022	2	Nie
8190	01/03/2022	2	Pon
8191	01/04/2022	2	Wto
8192	01/05/2022	2	Śro
8193	01/06/2022	2	Thu
8194	01/07/2022	2	Fri
8195	01/08/2022	2	Sat
8196	01/09/2022	3	Nie
8197	01/10/2022	3	Pon
8198	01/11/2022	3	Wto
8199	01/12/2022	3	Śro
8200	01/13/2022	3	Thu
8201	01/14/2022	3	Fri

Pole `week_number` jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `week()` i przekazanie pola `date` jako argumentu funkcji.

Do funkcji nie są przekazywane żadne inne parametry, w związku z czym obowiązują następujące domyślne zmienne wpływające na funkcję `week()`:

- `BrokenWeeks`: Numerowanie tygodni zaczyna się od 1 stycznia
- `FirstWeekDay`: Pierwszy dzień tygodnia to niedziela

Diagram funkcji week(), przykład obiektu wykresu



Ponieważ aplikacja używa domyślnej zmiennej systemowej `BrokenWeeks`, tydzień pierwszy zaczyna się 1 stycznia, czyli w sobotę.

Ze względu na domyślną zmienną systemową `FirstWeekDay` tygodnie zaczynają się w niedzielę. Pierwsza niedziela po 1 stycznia przypada 2 stycznia, więc tego dnia zaczyna się tydzień drugi.

Przykład 6 – Scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za ostatni tydzień 2019 r. i pierwsze dwa tygodnie 2020 r., który jest ładowany do tabeli o nazwie `Transactions`.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).

Aplikacja używa przede wszystkim podzielonych tygodni na swoim pulpicie. Jednak użytkownik końcowy chciałby otrzymać obiekt wykresu przedstawiający łączną sprzedaż w ujęciu tygodniowym przy użyciu tygodni niedzielonych. Dniem odniesienia powinien być 2 stycznia, przy tygodniach zaczynających się we wtorek. Można to osiągnąć nawet wtedy, gdy ten wymiar nie jest dostępny w modelu danych, używając funkcji `week()` jako wymiaru obliczanego na wykresie.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę.
2. Utwórz następujący wymiar wyliczany:
   `=week(date)`
3. Następnie utwórz następującą miarę agregacji:
   `=sum(amount)`
4. Ustaw **Formatowanie liczb** miary na **Waluta**.
5. Wybierz menu **Sortowanie** i dla wymiaru wyliczanego usuń sortowanie niestandardowe.
6. Usuń zaznaczenie opcji **Sortuj w kolejności liczbowej** i **Sortuj alfabetycznie**.

Tabela wynikowa

week(date)	sum(amount)
52	$125.69
53	$146.42
1	$200.09
2	$347.57
3	$122.01

weekday

Ta funkcja zwraca wartość podwójną:

- Nazwę dnia zdefiniowaną w zmiennej środowiskowej **DayNames**.
- Wartość całkowitą od 0 do 6, która odpowiada nominalnemu dniu tygodnia (0-6).

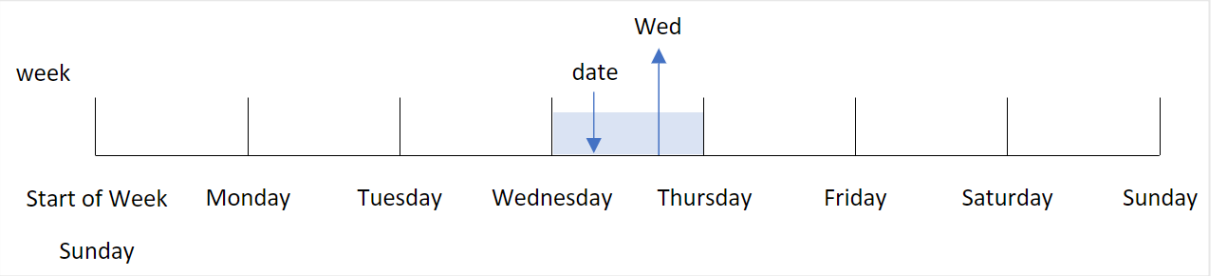
Składnia:

```
weekday(date [,first_week_day=0])
```

Typ zwracanych danych: dual

Funkcja `weekday()` określa, w którym dniu tygodnia wypada dana data. Następnie zwraca łańcuch reprezentujący ten dzień.

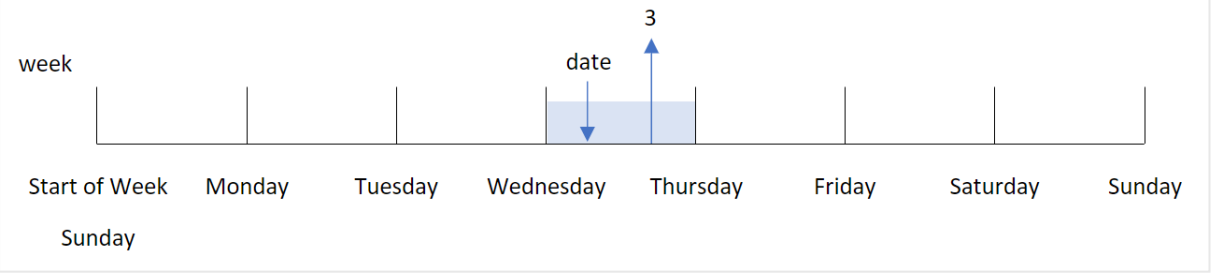
Diagram funkcji `weekday()`, która zwraca nazwę dnia, w którym przypada określona data



W wyniku jest zwracana wartość liczbowa odpowiadająca temu dniu tygodnia (0-6), na podstawie pierwszego dnia tygodnia. Na przykład, jeśli pierwszy dzień tygodnia jest ustawiony na niedzielę, dla środy zostanie zwrócona wartość `3`. Ten dzień początkowy jest określony przez zmienną systemową `FirstWeekDay` lub przez parametr funkcji `first_week_day`.

Tej wartości liczbowej można użyć w wyrażeniu arytmetycznym. Można ją na przykład pomnożyć przez `1`, aby zwrócić ją samą.

Diagram funkcji `weekday()` z pokazaną wartością liczbową dnia zamiast nazwy dnia



Kiedy używać

Funkcja `weekday()` jest przydatna, gdy chcesz porównać agregacje według dnia tygodnia. Można jej na przykład użyć, aby porównać średnią sprzedaży produktów według dni tygodnia.

Te wymiary można utworzyć w skrypcie ładowania, używając tej funkcji w celu utworzenia pola w tabeli **kalendarza głównego** lub bezpośrednio na wykresie jako miary wyliczanej.

Tematy pokrewne

Tematy	Interakcja
FirstWeekDay (page 215)	Definiuje dzień rozpoczęcia każdego tygodnia.

Argumenty

Argument	Opis
date	Data lub znacznik czasu do oszacowania.
first_week_day	Określa dzień początku tygodnia. Jeśli ten argument zostanie pominięty, wówczas zostanie użyta wartość zmiennej **FirstWeekDay**. *FirstWeekDay (page 215)*

Aby ustawić dzień, w którym zaczyna się tydzień w argumencie `first_week_day`, możesz użyć następujących wartości:

first_week_day values

Dzień	Wartość
poniedziałek	0
wtorek	1
środa	2
czwartek	3
piątek	4
sobota	5
niedziela	6

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Jeśli nie określono inaczej, w tych przykładach zmienna `FirstWeekDay` jest ustawiona na `0`.

Przykłady funkcji

Przykład	Wynik
`weekday('10/12/1971')`	zwraca „Tue” i 1.

Przykład	Wynik
`weekday('10/12/1971' , 6)`	zwraca „True” i 2. Na tym przykładzie pierwszym dniem tygodnia jest niedziela (6).

`SET FirstWeekDay=6;`
`...`
 	
`weekday('10/12/1971')`
	Zwraca „True” i 2.

Przykład 1 – Dzień tygodnia jako łańcuch

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „`Transactions`”.
- Zmienna systemowa `FirstWeekDay`, która jest ustawiona na 6 (niedziela).
- Zmienna `DayNames`, która jest ustawiona na używanie domyślnych nazw dni.
- Poprzednie polecenie LOAD, które zawiera funkcję `weekday()`, która jest ustawiona jako pole `week_day` i zwraca dzień tygodnia, w którym odbyły się transakcje.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `week_day`

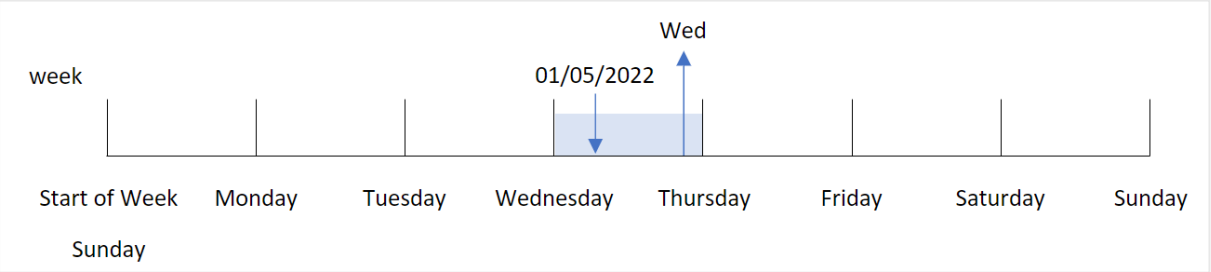
Tabela wynikowa

id	date	week_day
8188	01/01/2022	Sat
8189	01/02/2022	Nie
8190	01/03/2022	Pon
8191	01/04/2022	Wto
8192	01/05/2022	Śro
8193	01/06/2022	Thu
8194	01/07/2022	Fri

Pole „`week_day`" jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `weekday()` i przekazanie pola daty jako jej argumentu.

Funkcja `weekday()` zwraca wartość łańcuchową określającą dzień tygodnia, tzn. zwraca nazwę dnia tygodnia, która jest ustawiona przez zmienną systemową `DayNames`.

Diagram funkcji `weekday()`, która zwraca środę jako dzień tygodnia dla transakcji 8192



Transakcja 8192 miała miejsce 5 stycznia. Zmienna systemowa `FirstWeekDay` ustawia pierwszy dzień tygodnia na niedzielę. Transakcja funkcji `weekday()` miała miejsce w środę i funkcja ta zwraca tę wartość w skróconej formie zmiennej systemowej `DayNames`, w polu `week_day`.

Wartości w polu `week_day` są wyrównane w kolumnie do prawej, ponieważ to pole zawiera podwójny wynik złożony z liczby i tekstu (środa, 3). Aby zamienić wartość pola na liczbowy odpowiednik, pole to można umieścić w funkcji `num()`. Na przykład, w transakcji 8192 środa zostałaby zamieniona na liczbę 3

Przykład 2 – first_week_day

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „Transactions”.
- Zmienna systemowa `FirstWeekDay`, która jest ustawiona na 6 (niedziela).
- Zmienna `DayNames`, która jest ustawiona na używanie domyślnych nazw dni.
- Poprzednie polecenie LOAD, które zawiera funkcję `weekday()`, która jest ustawiona jako pole `week_day` i zwraca dzień tygodnia, w którym odbyły się transakcje.

Skrypt ładowania

```
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
];
```

Wyniki

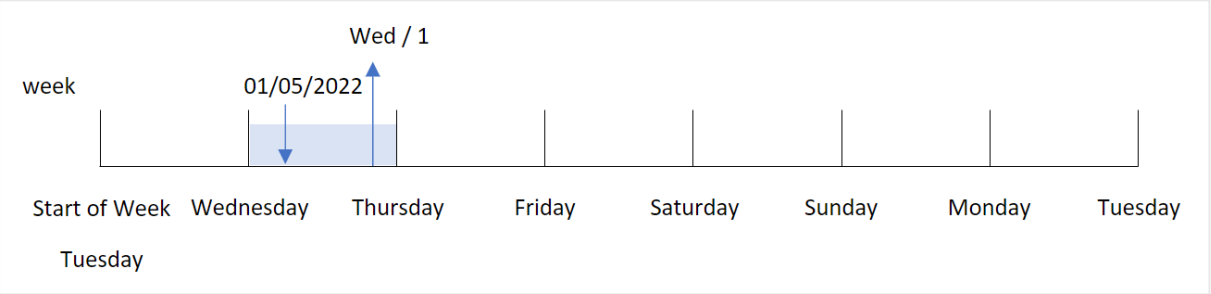
Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `week_day`

Tabela wynikowa

id	date	week_day
8188	01/01/2022	Sat
8189	01/02/2022	Nie
8190	01/03/2022	Pon
8191	01/04/2022	Wto
8192	01/05/2022	Śro
8193	01/06/2022	Thu
8194	01/07/2022	Fri

Diagram funkcji `weekday()`, który pokazuje, że środa ma wartość liczbową 1



Ponieważ funkcji `weekday()` przekazano wartość `1` jako argument `first_week_day`, pierwszym dniem tygodnia jest wtorek. W związku z tym wszystkie transakcje, które mają miejsce we wtorek będą miały podwójną wartość liczbową `0`.

Transakcja 8192 miała miejsce 5 stycznia. Funkcja `weekday()` stwierdza, że jest to środa, więc wyrażenie zwróciłoby podwójną wartość liczbową `1`.

Przykład 3 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „`Transactions`”.
- Zmienna systemowa `FirstWeekDay`, która jest ustawiona na `6` (niedziela).
- Zmienna `DayNames`, która jest ustawiona na używanie domyślnych nazw dni.

Jednak w tym przykładzie zestaw danych pozostaje bez zmian i jest ładowany do aplikacji. Obliczenia identyfikujące wartość dnia tygodnia są utworzone jako miara na wykresie w aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`

Aby obliczyć wartość dnia tygodnia, należy utworzyć następującą miarę:

- `=weekday(date)`

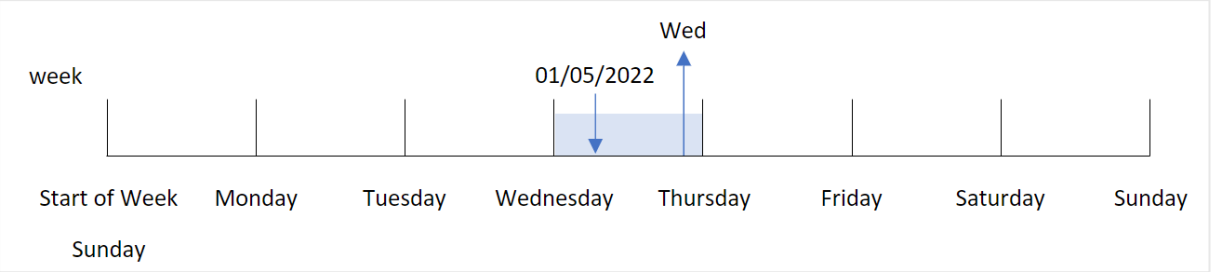
Tabela wynikowa

id	date	=weekday(date)
8188	01/01/2022	Sat
8189	01/02/2022	Nie
8190	01/03/2022	Pon
8191	01/04/2022	Wto
8192	01/05/2022	Śro
8193	01/06/2022	Thu
8194	01/07/2022	Fri

Miara `=weekday(date)` jest tworzona na wykresie przy użyciu funkcji `weekday()`, której jako argument przekazano pole daty.

Funkcja `weekday()` zwraca wartość łańcuchową określającą dzień tygodnia, tzn. zwraca nazwę dnia tygodnia, która jest ustawiona przez zmienną systemową `DayNames`.

Diagram funkcji `weekday()`, która zwraca środę jako dzień tygodnia dla transakcji 8192



Transakcja 8192 miała miejsce 5 stycznia. Zmienna systemowa `FirstWeekDay` ustawia pierwszy dzień tygodnia na niedzielę. Transakcja funkcji `weekday()` miała miejsce w środę i funkcja ta zwraca tę wartość w skróconej formie zmiennej systemowej `DayNames`, w polu `=weekday(date)`.

Przykład 4 – Scenariusz

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „`Transactions`”.
- Zmienna systemowa `FirstWeekDay`, która jest ustawiona na `6` (niedziela).
- Zmienna `DayNames`, która jest ustawiona na używanie domyślnych nazw dni.

Użytkownik końcowy chciałby otrzymać wykres przedstawiający średnią sprzedaż według dnia tygodnia dla transakcji.

Skrypt ładowania

```
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);
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `=weekday(date)`
- `=avg(amount)`

Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

=weekday(date)	Avg(amount)
Nie	$536.96
Pon	$500.80
Wto	$515.63
Śro	$509.21
Thu	$482.70
Fri	$441.33
Sat	$505.22

weekend

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu ostatniej milisekundy ostatniego dnia (niedzieli) tygodnia kalendarzowego, który zawiera wartość **date**. Domyślnym formatem wyjściowym będzie format **DateFormat** skonfigurowany w skrypcie.

Składnia:

```
WeekEnd(date [, period_no[, first_week_day]])
```

Typ zwracanych danych: dual

Przykładowy diagram funkcji `weekend()`

Funkcja `weekend()` określa, w którym tygodniu wypada data. Następnie zwraca znacznik czasu w formacie daty dla ostatniej milisekundy tego tygodnia. Pierwszy dzień tygodnia jest określany przez zmienną systemową `FirstWeekDay`. Można ją jednak zastąpić argumentem `first_week_day` funkcji `weekend()`.

Argumenty

Argument	Opis
date	Data lub znacznik czasu do oszacowania.

Argument	Opis
period_no	**shift** jest liczbą całkowitą, gdzie 0 oznacza tydzień, który zawiera datę **date**. Wartości ujemne parametru shift oznaczają poprzednie tygodnie, a wartości dodatnie – tygodnie następne.
first_week_day	Określa dzień początku tygodnia. Jeśli ten argument zostanie pominięty, wówczas zostanie użyta wartość zmiennej **FirstWeekDay**. Możliwe wartości argumentu **first_week_day** to poniedziałek – 0, wtorek – 1, środa – 2, czwartek – 3, piątek – 4, sobota – 5 oraz niedziela – 6. Aby uzyskać więcej informacji o tej zmiennej systemowej, zobacz *FirstWeekDay (page 215)*
broken_weeks	Jeśli wartość **broken_weeks** nie zostanie określona, wówczas wartość zmiennej **BrokenWeeks** zostanie zastosowana w celu zdefiniowania, czy tygodnie są dzielone, czy nie.

Kiedy używać

Funkcja `weekend()` jest używana jako część wyrażenia, gdy użytkownik chce, aby w obliczeniach użyto liczby pozostałych dni tygodnia dla określonej daty. Na przykład, można jej użyć, jeśli użytkownik chce obliczyć sumę odsetek, które nie zostały jeszcze naliczone w ciągu tygodnia.

Przykład	Wynik
`weekend('01/10/2013')`	Zwraca wartość 01/12/2013 23:59:59.
`weekend('01/10/2013', -1)`	Zwraca wartość 01/05/2013 23:59:59..
`weekend('01/10/2013', 0, 1)`	Zwraca wartość 01/14/2013 23:59:59.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 — Przykład podstawowy

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „Transactions”.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie pola `end_of_week` zwracającego znacznik czasu końca tygodnia, w którym miały miejsce transakcje.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `end_of_week`
- `end_of_week_timestamp`

Tabela wynikowa

date	end_of_week	end_of_week_timestamp
1/7/2022	01/08/2022	1/8/2022 11:59:59 PM
1/19/2022	01/22/2022	1/22/2022 11:59:59 PM
2/5/2022	02/05/2022	2/5/2022 11:59:59 PM
2/28/2022	03/05/2022	3/5/2022 11:59:59 PM
3/16/2022	03/19/2022	3/19/2022 11:59:59 PM
4/1/2022	04/02/2022	4/2/2022 11:59:59 PM
5/7/2022	05/07/2022	5/7/2022 11:59:59 PM
5/16/2022	05/21/2022	5/21/2022 11:59:59 PM
6/15/2022	06/18/2022	6/18/2022 11:59:59 PM
6/26/2022	07/02/2022	7/2/2022 11:59:59 PM
7/9/2022	07/09/2022	7/9/2022 11:59:59 PM
7/22/2022	07/23/2022	7/23/2022 11:59:59 PM
7/23/2022	07/23/2022	7/23/2022 11:59:59 PM
7/27/2022	07/30/2022	7/30/2022 11:59:59 PM
8/2/2022	08/06/2022	8/6/2022 11:59:59 PM
8/8/2022	08/13/2022	8/13/2022 11:59:59 PM
8/19/2022	08/20/2022	8/20/2022 11:59:59 PM
9/26/2022	10/01/2022	10/1/2022 11:59:59 PM
10/14/2022	10/15/2022	10/15/2022 11:59:59 PM
10/29/2022	10/29/2022	10/29/2022 11:59:59 PM

Pole `end_of_week` jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `weekend()` i przekazanie pola daty jako jej argumentu.

Funkcja `weekend()` identyfikuje, w którym tygodniu wypada wartość daty, i zwraca znacznik czasu ostatniej milisekundy danego tygodnia.

Diagram funkcji weekend(), przykład podstawowy

Transakcja 8191 miała miejsce 5 lutego. Zmienna systemowa `FirstWeekDay` ustawia pierwszy dzień tygodnia na niedzielę. Funkcja `weekend()` dowiaduje się, że pierwsza sobota po 5 lutego – a więc po końcu tygodnia – wypadała 5 lutego. W związku z tym wartość `end_of_week` dla tej transakcji zwraca ostatnią milisekundę tego dnia – 5 lutego, godz. 23:59:59.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola `previous_week_end` zwracającego znacznik czasu początku tygodnia przed transakcją.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `previous_week_end`
- `previous_week_end_timestamp`

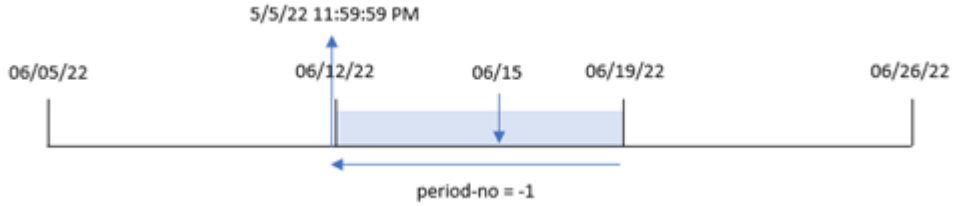
Tabela wynikowa

date	end_of_week	end_of_week_timestamp
1/7/2022	01/01/2022	1/1/2022 11:59:59 PM
1/19/2022	01/15/2022	1/15/2022 11:59:59 PM
2/5/2022	01/29/2022	1/29/2022 11:59:59 PM
2/28/2022	02/26/2022	2/26/2022 11:59:59 PM
3/16/2022	03/12/2022	3/12/2022 11:59:59 PM
4/1/2022	03/26/2022	3/26/2022 11:59:59 PM
5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
5/16/2022	05/14/2022	5/14/2022 11:59:59 PM
6/15/2022	06/11/2022	6/11/2022 11:59:59 PM
6/26/2022	06/25/2022	6/25/2022 11:59:59 PM
7/9/2022	07/02/2022	7/2/2022 11:59:59 PM
7/22/2022	07/16/2022	7/16/2022 11:59:59 PM
7/23/2022	07/16/2022	7/16/2022 11:59:59 PM
7/27/2022	07/23/2022	7/23/2022 11:59:59 PM
8/2/2022	07/30/2022	7/30/2022 11:59:59 PM
8/8/2022	08/06/2022	8/6/2022 11:59:59 PM
8/19/2022	08/13/2022	8/13/2022 11:59:59 PM

date	end_of_week	end_of_week_timestamp
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

W tym przypadku, ponieważ wartości `period_no` -1 użyto jako argumentu przesunięcia w funkcji `weekend()`, funkcja najpierw identyfikuje tydzień, w którym zawarto transakcje. Następnie identyfikuje ostatnią milisekundę poprzedniego tygodnia.

Diagram funkcji `weekend()`, przykład z argumentem period_no



Transakcja 8196 miała miejsce 15 czerwca. Funkcja `weekend()` oblicza, że tydzień zaczyna się 12 czerwca. W związku z tym poprzedni tydzień kończy się 11 czerwca o godz. 23:59:59. Ta wartość jest zwracana dla pola `previous_week_end`.

Przykład 3 – first_week_day

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie. Jednak w tym przykładzie musimy ustawić wtorek jako pierwszy dzień tygodnia roboczego.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `end_of_week`
- `end_of_week_timestamp`

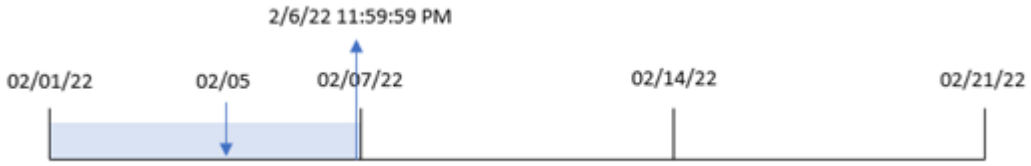
Tabela wynikowa

date	end_of_week	end_of_week_timestamp
1/7/2022	01/10/2022	1/10/2022 11:59:59 PM
1/19/2022	01/24/2022	1/24/2022 11:59:59 PM
2/5/2022	02/07/2022	2/7/2022 11:59:59 PM
2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
3/16/2022	03/21/2022	3/21/2022 11:59:59 PM
4/1/2022	04/04/2022	4/4/2022 11:59:59 PM
5/7/2022	05/09/2022	5/9/2022 11:59:59 PM
5/16/2022	05/16/2022	5/16/2022 11:59:59 PM
6/15/2022	06/20/2022	6/20/2022 11:59:59 PM
6/26/2022	06/27/2022	6/27/2022 11:59:59 PM
7/9/2022	07/11/2022	7/11/2022 11:59:59 PM

date	end_of_week	end_of_week_timestamp
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

W tym przypadku, ponieważ funkcji `weekend()` przekazano wartość 1 jako argument `first_week_date`, funkcja ta ustawia wtorek jako pierwszy dzień tygodnia.

Diagram funkcji `weekend()`, przykład z argumentem first_week_day



Transakcja 8191 miała miejsce 5 lutego. Funkcja `weekend()` oblicza, że pierwszy poniedziałek po tej dacie – a zatem także koniec tygodnia i zwracana wartość – przypadł na 6 lutego o godz. 23:59:59.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie. Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenia zwracające znacznik czasu końca tygodnia, w którym wystąpiły transakcje, są tworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Aby obliczyć początek tygodnia, kiedy ma miejsce transakcja, dodaj następujące miary:

- `=weekend(date)`
- `=timestamp(weekend(date))`

Tabela wynikowa

date	=weekend(date)	=timestamp(weekend(date))
1/7/2022	01/08/2022	1/8/2022 11:59:59 PM
1/19/2022	01/22/2022	1/22/2022 11:59:59 PM
2/5/2022	02/05/2022	2/5/2022 11:59:59 PM
2/28/2022	03/05/2022	3/5/2022 11:59:59 PM
3/16/2022	03/19/2022	3/19/2022 11:59:59 PM
4/1/2022	04/02/2022	4/2/2022 11:59:59 PM
5/7/2022	05/07/2022	5/7/2022 11:59:59 PM
5/16/2022	05/21/2022	5/21/2022 11:59:59 PM
6/15/2022	06/18/2022	6/18/2022 11:59:59 PM
6/26/2022	07/02/2022	7/2/2022 11:59:59 PM

date	=weekend(date)	=timestamp(weekend(date))
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

Miarę „`end_of_week`" tworzy się w obiekcie wykresu, używając funkcji `weekend()` i przekazując pole daty jako jej argument. Funkcja `weekend()` identyfikuje, w którym tygodniu przypada wartość daty, i zwraca znacznik czasu ostatniej milisekundy danego tygodnia.

Diagram funkcji `weekend()`, przykład obiektu wykresu

Transakcja 8191 miała miejsce 5 lutego. Zmienna systemowa `FirstWeekDay` ustawia pierwszy dzień tygodnia na niedzielę. Funkcja `weekend()` dowiaduje się, że pierwsza sobota po 5 lutego – a więc po końcu tygodnia – wypadała 5 lutego. W związku z tym wartość `end_of_week` dla tej transakcji zwraca ostatnią milisekundę tego dnia – 5 lutego, godz. 23:59:59.

Przykład 5 – scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Employee_Expenses`.
- Dane zawierające identyfikatory pracowników, imiona i nazwiska pracowników oraz średnie dzienne roszczenia z tytułu wydatków każdego pracownika.

Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał, według identyfikatora pracownika oraz imienia i nazwiska pracownika, szacowane roszczenia z tytułu wydatków do poniesienia przez pozostałą część tygodnia.

Skrypt ładowania

```
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
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:
   - `employee_id`
   - `employee_name`
2. Następnie utwórz miarę, aby obliczyć narosłe odsetki:
   `=(weekend(today(1))-today(1))*avg_daily_claim`
3. Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

employee_id	employee_name	=(weekend(today(1))-today(1))*avg_daily_claim
182	Mark	$90.00
183	Deryck	$75.00
184	Dexter	$75.00
185	Sydney	$162.00
186	Agatha	$108.00

Funkcja `weekend()`, przyjmująca jako argument tylko dzisiejszą datę, zwraca datę końcową bieżącego tygodnia. Następnie wyrażenie odejmuje dzisiejszą datę od daty zakończenia tygodnia i zwraca liczbę dni pozostałych w tym tygodniu.

Wartość ta jest następnie mnożona przez średnie dzienne roszczenie z tytułu wydatków przez każdego pracownika, aby obliczyć szacunkową wartość roszczeń, które każdy pracownik złoży w pozostałej części tygodnia.

weekname

Ta funkcja zwraca wartość pokazującą rok i numer tygodnia z bazową wartością liczbową odpowiadającą znacznikowi czasu pierwszej milisekundy pierwszego dnia tygodnia, który zawiera datę **date**.

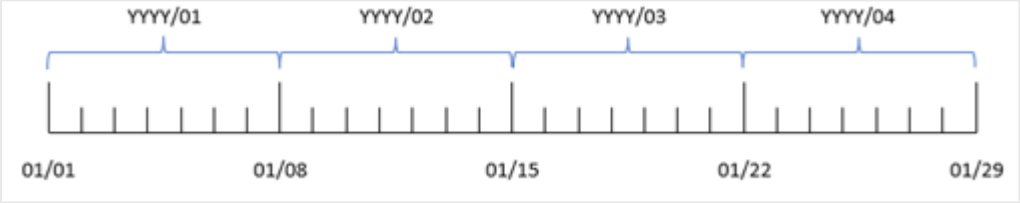
Składnia:

```
WeekName(date[, period_no[,first_week_day]])
```

Funkcja `weekname()` określa, w którym tygodniu przypada data, i zwraca numer tygodnia oraz rok tego tygodnia. Pierwszy dzień tygodnia jest określany przez zmienną systemową `FirstWeekDay`. Możesz jednak również zmienić pierwszy dzień tygodnia, używając argumentu `first_week_day` w funkcji `weekname()`.

Domyślnie aplikacje Qlik Sense używają podzielonych tygodni (zdefiniowanych przez zmienną systemową `BrokenWeeks`), dlatego numerowanie tygodni zaczyna się 1 stycznia i kończy w dniu poprzedzającym zmienną systemową `FirstWeekDay`, niezależnie od liczby dni.

Schemat funkcji weekname.



Jeśli jednak Twoja aplikacja korzysta z niepodzielonych tygodni, tydzień 1 może rozpocząć się w poprzednim roku lub w pierwszych dniach stycznia. Zależy to od sposobu używania zmiennych systemowych `ReferenceDay` i `FirstWeekDay`.

Kiedy używać

Funkcja `weekname()` jest przydatna, gdy chcesz porównać agregacje według tygodni.

Na przykład, jeśli chcesz zobaczyć całkowitą sprzedaż produktów w tygodniu. Aby zachować spójność ze zmienną środowiskową `BrokenWeeks` w aplikacji, użyj `weekname()` zamiast `lunarweekname()`. Jeśli aplikacja używa niepodzielonych tygodni, tydzień 1 może zawierać daty z grudnia poprzedniego roku lub wykluczać daty ze stycznia bieżącego roku. Jeśli aplikacja używa podzielonych tygodni, tydzień 1 może zawierać mniej niż siedem dni.

Typ zwracanych danych: dual

Argumenty

Argument	Opis
date	Data lub znacznik czasu do oszacowania.

Argument	Opis
period_no	**shift** jest liczbą całkowitą, gdzie 0 oznacza tydzień, który zawiera datę **date**. Wartości ujemne parametru shift oznaczają poprzednie tygodnie, a wartości dodatnie – tygodnie następne.
first_week_day	Określa dzień początku tygodnia. Jeśli ten argument zostanie pominięty, wówczas zostanie użyta wartość zmiennej **FirstWeekDay**. Możliwe wartości **first_week_day** to poniedziałek – 0, wtorek – 1, środa – 2, czwartek – 3, piątek – 4, sobota – 5 oraz niedziela – 6. Aby uzyskać więcej informacji o zmiennej systemowej, zobacz *FirstWeekDay (page 215)*.

Aby ustawić dzień, w którym zaczyna się tydzień w argumencie `first_week_day`, możesz użyć następujących wartości:

first_week_day values

Dzień	Wartość
poniedziałek	0
wtorek	1
środa	2
czwartek	3
piątek	4
sobota	5
niedziela	6

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
weekname('01/12/2013')	Zwraca 2013/02.
weekname('01/12/2013', -1)	Returns 2013/01.
weekname('01/12/2013', 0, 1)	Zwraca 2013/02.

Przykład 1 – data bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za ostatni tydzień 2021 r. jest ładowany do tabeli o nazwie „Transactions”.
- Zmienna systemowa `DateFormat`, która jest ustawiona na format `MM/DD/YYYY`.
- Zmienna systemowa `BrokenWeeks`, która jest ustawiona na `1`.
- Zmienna systemowa `FirstWeekDay`, która jest ustawiona na `6`.
- Ładunek poprzedzający, który zawiera następujące elementy:
  - Funkcja `weekday()` ustawiona jako pole „`week_number`”, która zwraca rok i numer tygodnia, w którym miały miejsce transakcje.
  - Funkcja `weekname()`, która jest ustawiona jako pole o nazwie „`week_day`”, aby pokazywać wartość dnia tygodnia dla każdej daty transakcji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `week_day`
- `week_number`

Tabela wynikowa

id	date	week_day	week_number
8183	12/27/2021	Pon	2021/53
8184	12/28/2021	Wto	2021/53
8185	12/29/2021	Śro	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	Nie	2022/02
8190	01/03/2022	Pon	2022/02
8191	01/04/2022	Wto	2022/02
8192	01/05/2022	Śro	2022/02
8193	01/06/2022	Thu	2022/02
8194	01/07/2022	Fri	2022/02
8195	01/08/2022	Sat	2022/02

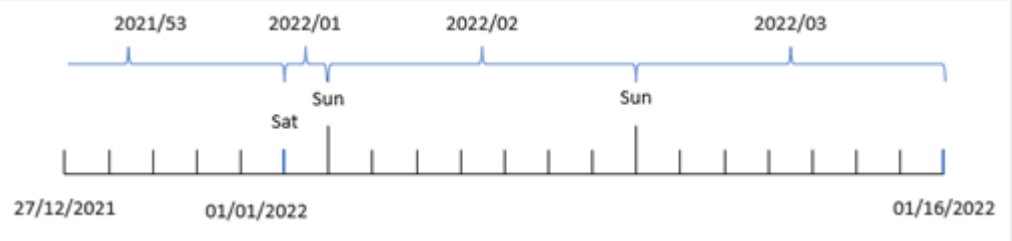
id	date	week_day	week_number
8196	01/09/2022	Nie	2022/03
8197	01/10/2022	Pon	2022/03
8198	01/11/2022	Wto	2022/03
8199	01/12/2022	Śro	2022/03
8200	01/13/2022	Thu	2022/03
8201	01/14/2022	Fri	2022/03

Pole „week_number” jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `weekname()` i przekazanie pola daty jako jej argumentu.

Funkcja `weekname()` początkowo identyfikuje tydzień, w którym przypada wartość daty, i zwraca liczbę numerów tygodni oraz rok, w którym ma miejsce transakcja.

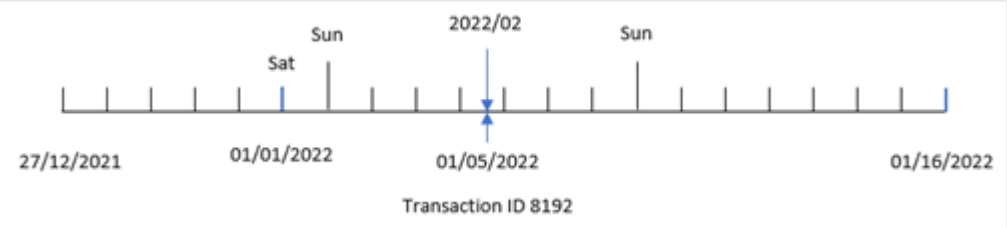
Zmienna systemowa `FirstWeekDay` ustawia niedzielę jako pierwszy dzień tygodnia. Zmienna systemowa `BrokenWeeks` ustawia aplikację tak, aby używała podzielonych tygodni, co oznacza, że tydzień 1 rozpocznie się 1 stycznia.

Diagram funkcji `weekname()` ze zmiennymi domyślnymi.



Tydzień 1 rozpoczyna się 1 stycznia, czyli w sobotę, dlatego transakcje występujące w tym dniu zwracają wartość `2022/01` (rok i numer tygodnia).

Diagram funkcji `weekname()` identyfikującej numer tygodnia transakcji 8192.



Ponieważ aplikacja używa podzielonych tygodni, a pierwszym dniem tygodnia jest niedziela, transakcje występujące od 2 do 8 stycznia zwracają wartość `2022/02` (tydzień 2 w 2022 r.). Przykładem może być transakcja 8192, która miała miejsce 5 stycznia, i zwraca wartość `2022/02` dla pola „week_number”.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie zadaniem jest utworzenie pola „previous_week_number”, które zwraca rok i numer tygodnia sprzed transakcji.

Otwórz Edytor ładowania danych i dodaj następujący skrypt ładowania do nowej karty.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

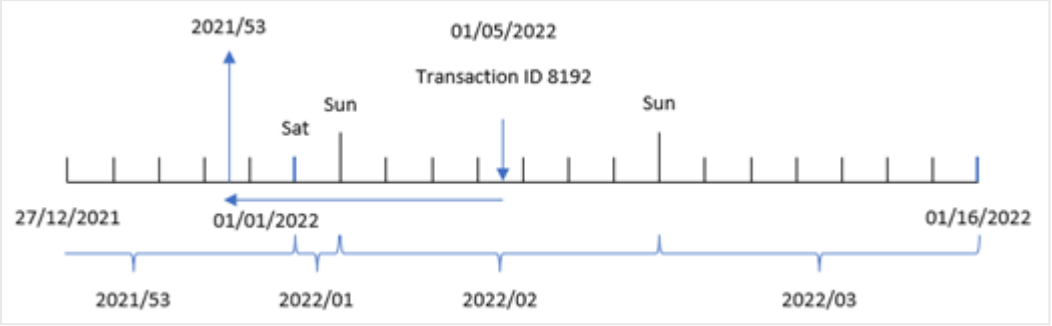
- id
- date
- week_day
- week_number

Tabela wynikowa

id	date	week_day	week_number
8183	12/27/2021	Pon	2021/52
8184	12/28/2021	Wto	2021/52
8185	12/29/2021	Śro	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	Nie	2021/53
8190	01/03/2022	Pon	2021/53
8191	01/04/2022	Wto	2021/53
8192	01/05/2022	Śro	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	Nie	2022/02
8197	01/10/2022	Pon	2022/02
8198	01/11/2022	Wto	2022/02
8199	01/12/2022	Śro	2022/02
8200	01/13/2022	Thu	2022/02
8201	01/14/2022	Fri	2022/02

Ponieważ `period_no` o wartości `-1` użyto jako argumentu przesunięcia w funkcji `weekname()`, funkcja najpierw identyfikuje tydzień, w którym odbywają się transakcje. Następnie identyfikuje pierwszą milisekundę poprzedniego tygodnia.

Diagram funkcji `weekname()` z przesunięciem `period_no` równym -1.



Transakcja 8192 miała miejsce 5 stycznia 2022 r. Funkcja `weekname()` zwraca numer tygodnia i rok dla daty 30 grudnia 2021 r. z poprzedniego tygodnia – `2021/53`.

Przykład 3 – first_week_day

Skrypt ładowania i wyniki

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie zgodnie z polityką firmową tydzień roboczy rozpoczyna się we wtorek.

Otwórz Edytor ładowania danych i dodaj następujący skrypt ładowania do nowej karty.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

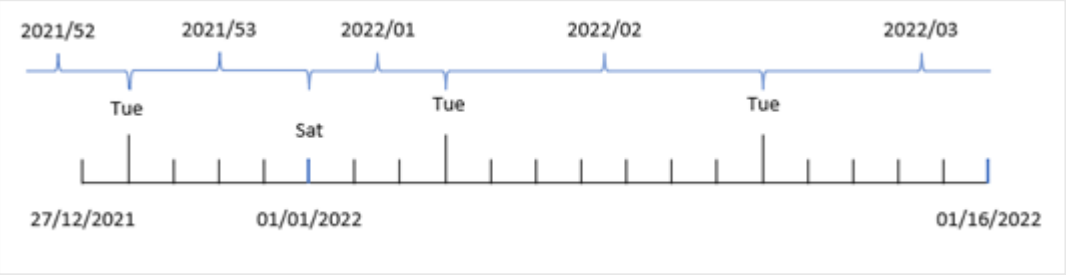
- `id`
- `date`
- `week_day`
- `week_number`

Tabela wynikowa

id	date	week_day	week_number
8183	12/27/2021	Pon	2021/52
8184	12/28/2021	Wto	2021/53
8185	12/29/2021	Śro	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	Nie	2022/01
8190	01/03/2022	Pon	2022/01
8191	01/04/2022	Wto	2022/02
8192	01/05/2022	Śro	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	Nie	2022/02
8197	01/10/2022	Pon	2022/02
8198	01/11/2022	Wto	2022/03
8199	01/12/2022	Śro	2022/03

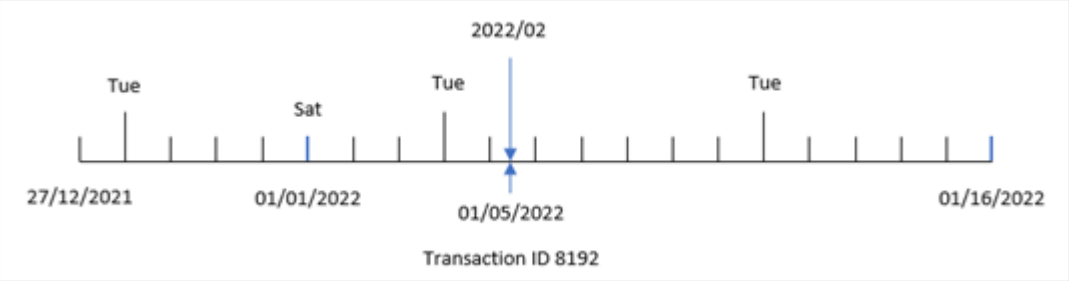
id	date	week_day	week_number
8200	01/13/2022	Thu	2022/03
8201	01/14/2022	Fri	2022/03

Schemat funkcji `weekname()` z wtorkiem jako pierwszym dniem tygodnia.



Ponieważ argument `first_week_date` o wartości `1` jest używany w funkcji `weekname()`, używa ona wtorku jako pierwszego dnia tygodnia. Funkcja ta określa więc, że 53. tydzień 2021 r. rozpoczyna się we wtorek 28 grudnia, a ze względu na to, że aplikacja używa niepodzielonych tygodni, tydzień 1. rozpoczyna się 1 stycznia 2022 r., a kończy w ostatnią milisekundę poniedziałku 3 stycznia 2022 r.

Diagram przedstawiający numer tygodnia transakcji 8192 z wtorkiem jako pierwszym dniem tygodnia.



Transakcja 8192 miała miejsce 5 stycznia 2022 r. W związku z tym, używając wtorku jako parametru `first_week_day`, funkcja `weekname()` zwraca wartość `2022/02` dla pola „`week_number`”.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie zestaw danych pozostaje bez zmian i jest ładowany do aplikacji. Obliczenie zwracające numer roku tygodnia, kiedy wystąpiły transakcje, jest tworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `=week_day (date)`

Aby obliczy początek tygodnia, kiedy ma miejsce transakcja, utwórz następującą miarę:

```
=weekname(date)
```

Tabela wynikowa

id	date	=weekday(date)	=weekname(date)
8183	12/27/2021	Pon	2021/53
8184	12/28/2021	Wto	2021/53
8185	12/29/2021	Śro	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	Nie	2022/02
8190	01/03/2022	Pon	2022/02
8191	01/04/2022	Wto	2022/02
8192	01/05/2022	Śro	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	Nie	2022/03
8197	01/10/2022	Pon	2022/03
8198	01/11/2022	Wto	2022/03
8199	01/12/2022	Śro	2022/03
8200	01/13/2022	Thu	2022/03
8201	01/14/2022	Fri	2022/03

Miarę „week_number” tworzy się w obiekcie wykresu, używając funkcji `weekname()` i przekazując pole daty jako jej argument.

Funkcja `weekname()` początkowo identyfikuje tydzień, w którym przypada wartość daty, i zwraca liczbę numerów tygodni oraz rok, w którym ma miejsce transakcja.

Zmienna systemowa `FirstWeekDay` ustawia niedzielę jako pierwszy dzień tygodnia. Zmienna systemowa `BrokenWeeks` ustawia aplikację tak, aby używała podzielonych tygodni, co oznacza, że tydzień 1 rozpoczyna się 1 stycznia.

Diagram przedstawiający numer tygodnia z sobotą jako pierwszym dniem tygodnia.

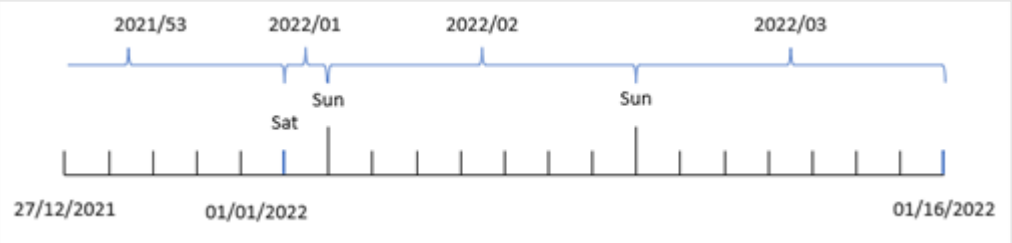
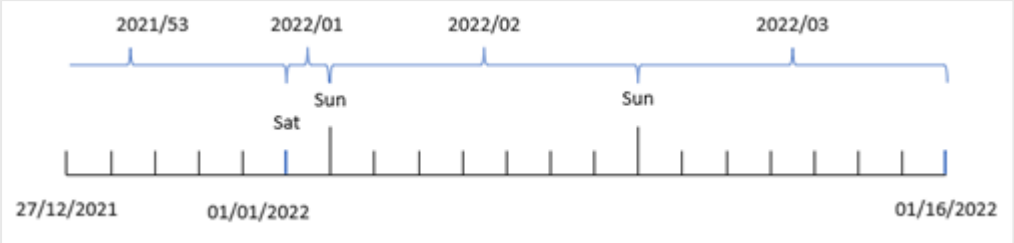


Diagram pokazujący, że transakcja 8192 miała miejsce w drugim tygodniu.



Ponieważ aplikacja używa podzielonych tygodni, a pierwszym dniem tygodnia jest niedziela, transakcje występujące od 2 do 8 stycznia zwracają wartość 2022/02 (tydzień 2 w 2022 r.). Zauważ, że transakcja 8192 miała miejsce 5 stycznia i zwraca wartość 2022/02 dla pola „week_number”.

Przykład 5 – scenariusz

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za ostatni tydzień 2019 r. i pierwsze dwa tygodnie 2020 r. jest ładowany do tabeli o nazwie „Transactions”.
- Zmienna systemowa BrokenWeeks, która jest ustawiona na 0.
- Zmienna systemowa ReferenceDay, która jest ustawiona na 2.
- Zmienna systemowa DateFormat, która jest ustawiona na format MM/DD/YYYY.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę.

Utwórz wymiar wyliczany, używając następującego wyrażenia:

```
=weekname(date)
```

Aby obliczyć łączną sprzedaż, utworzyć następującą miarę agregacji:

```
=sum(amount)
```

Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

weekname(date)	=sum(amount)
2019/52	$125.69
2020/01	$346.51
2020/02	$347.57
2020/03	$122.01

Aby zademonstrować wyniki użycia funkcji weekname() w tym scenariuszu, dodaj następujące pole jako wymiar:

```
date
```

Tabela wyników z polem daty

weekname(date)	date	=sum(amount)
2019/52	12/27/2019	$58.27
2019/52	12/28/2019	$67.42
2020/01	12/29/2019	$23.80
2020/01	12/30/2019	$82.06
2020/01	12/31/2019	$40.56
2020/01	01/01/2020	$37.23

weekname(date)	date	=sum(amount)
2020/01	01/02/2020	$17.17
2020/01	01/03/2020	$88.27
2020/01	01/04/2020	$57.42
2020/02	01/05/2020	$53.80
2020/02	01/06/2020	$82.06
2020/02	01/07/2020	$40.56
2020/02	01/08/2020	$53.67
2020/02	01/09/2020	$26.63
2020/02	01/10/2020	$72.48
2020/02	01/11/2020	$18.37
2020/03	01/12/2020	$45.26
2020/03	01/13/2020	$58.23
2020/03	01/14/2020	$18.52

Ponieważ aplikacja używa niepodzielonych tygodni, a tydzień 1. wymaga co najmniej dwóch dni w styczniu ze względu na zmienną systemową `ReferenceDay`, tydzień 1. roku 2020 obejmuje transakcje z 29 grudnia 2019.

weekstart

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu pierwszej milisekundy pierwszego dnia tygodnia kalendarzowego zawierającego wartość **date**. Domyślnym formatem wyjściowym będzie format **DateFormat** skonfigurowany w skrypcie.

Składnia:

```
WeekStart(date [, period_no[, first_week_day]])
```

Typ zwracanych danych: dual

Schemat funkcji `weekstart()`

Funkcja `weekstart()` określa, w którym tygodniu wypada data. Następnie zwraca znacznik czasu w formacie daty dla pierwszej milisekundy tego tygodnia. Pierwszy dzień tygodnia jest określany przez zmienną systemową `FirstWeekDay`. Można ją jednak zastąpić argumentem `first_week_day` funkcji `weekstart()`.

Argumenty

Argument	Opis
date	Data lub znacznik czasu do oszacowania.
period_no	**shift** jest liczbą całkowitą, gdzie 0 oznacza tydzień, który zawiera datę **date**. Wartości ujemne parametru shift oznaczają poprzednie tygodnie, a wartości dodatnie – tygodnie następne.
first_week_day	Określa dzień początku tygodnia. Jeśli ten argument zostanie pominięty, wówczas zostanie użyta wartość zmiennej **FirstWeekDay**. Możliwe wartości **first_week_day** to poniedziałek – 0, wtorek – 1, środa – 2, czwartek – 3, piątek – 4, sobota – 5 oraz niedziela – 6. Aby uzyskać więcej informacji o zmiennej systemowej, zobacz *FirstWeekDay (page 215)*.

Kiedy używać

Funkcja `weekstart()` jest zwykle używana jako część wyrażenia, gdy użytkownik chce, by w obliczeniach użyto ułamka tygodnia, który upłynął do tej pory. Za jej pomocą można na przykład obliczyć łączną kwotę zarobioną przez pracowników do tej pory w ciągu tygodnia.

Przykłady funkcji

Przykład	Wynik
`weekstart('01/12/2013')`	Zwraca wartość 01/07/2013.
`weekstart('01/12/2013', -1 )`	Zwraca wartość 11/31/2012.
`weekstart('01/12/2013', 0, 1)`	Zwraca wartość 01/08/2013.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za rok 2022, który jest ładowany do tabeli o nazwie „Transactions”.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie pola `start_of_week` zwracającego znacznik czasu początku tygodnia, w którym miały miejsce transakcje.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

Tabela wynikowa

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

Pole `start_of_week` jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `weekstart()` i przekazanie pola daty jako jej argumentu.

Funkcja `weekstart()` najpierw identyfikuje, w którym tygodniu przypada wartość daty, i zwraca znacznik czasu pierwszej milisekundy danego tygodnia.

Diagram funkcji `weekstart()`, przykład bez dodatkowych argumentów

Transakcja 8191 miała miejsce 5 lutego. Zmienna systemowa `FirstWeekDay` ustawia pierwszy dzień tygodnia na niedzielę. Funkcja `weekstart()` oblicza, że pierwsza niedziela przed 5 lutego – a więc początkiem tygodnia – wypadała 30 stycznia. W związku z tym wartość `start_of_week` dla tej transakcji zwraca pierwszą milisekundę tego dnia – 30 stycznia, godz. 00:00:00.

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola `previous_week_start` zwracającego znacznik czasu początku kwartału, przed tym, w którym miała miejsce transakcja.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `previous_week_start`
- `previous_week_start_timestamp`

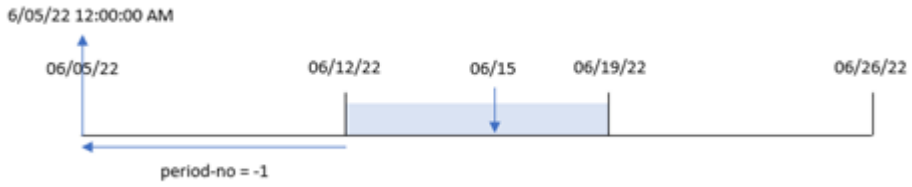
Tabela wynikowa

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

date	previous_week_start	previous_week_start_timestamp
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

W tym przypadku, ponieważ wartości `period_no` -1 użyto jako argumentu przesunięcia w funkcji `weekstart()`, funkcja najpierw identyfikuje tydzień, w którym odbywają się transakcje. Następnie identyfikuje pierwszą milisekundę poprzedniego tygodnia.

Diagram funkcji `weekstart()`, przykład z argumentem period_no



Transakcja 8196 miała miejsce 15 czerwca. Funkcja `weekstart()` oblicza, że tydzień zaczyna się 12 czerwca. W efekcie poprzedni tydzień rozpoczynał się 5 czerwca o godzinie 00:00:00. Ta wartość jest zwracana dla pola `previous_week_start`.

Przykład 3 – first_week_day

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie. Jednak w tym przykładzie musimy ustawić wtorek jako pierwszy dzień tygodnia roboczego.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

Tabela wynikowa

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

date	start_of_week	start_of_week_timestamp
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

W tym przypadku, ponieważ funkcji `weekstart()` przekazano wartość 1 jako argument `first_week_date`, funkcja ta ustawia wtorek jako pierwszy dzień tygodnia.

Diagram funkcji `weekstart()`, przykład z argumentem first_week_day

Transakcja 8191 miała miejsce 5 lutego. Funkcja `weekstart()` oblicza, że pierwszy wtorek przed tą datą - a zatem początek tygodnia i zwracana wartość - przypadł na 1 lutego o godz. 00:00:00.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenia zwracające znacznik czasu początku tygodnia, w którym wystąpiły transakcje, są tworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Aby obliczyć początek tygodnia, w którym ma miejsce transakcja, dodaj następujące miary:

- `=weekstart(date)`
- `=timestamp(weekstart(date))`

Tabela wynikowa

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

date	start_of_week	start_of_week_timestamp
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

Miarę „start_of_week" tworzy się w obiekcie wykresu, używając funkcji `weekstart()` i przekazując pole daty `date` jako jej argument.

Funkcja `weekstart()` najpierw identyfikuje, w którym tygodniu przypada wartość daty, i zwraca znacznik czasu pierwszej milisekundy danego tygodnia.

Diagram funkcji `weekstart()`, przykład obiektu wykresu

Transakcja 8191 miała miejsce 5 lutego. Zmienna systemowa `FirstWeekDay` ustawia pierwszy dzień tygodnia na niedzielę. Funkcja `weekstart()` oblicza, że pierwsza niedziela przed 5 lutego – a więc początek tygodnia – wypadała 30 stycznia. W związku z tym wartość `start_of_week` dla tamtej transakcji zwraca pierwszą milisekundę tamtego dnia, czyli 30 stycznia, godz. 00:00:00.

Przykład 5 – scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych załadowany do tabeli o nazwie `Payroll`.
- Dane zawierające identyfikatory pracowników, imiona i nazwiska pracowników oraz dzienne zarobki każdego pracownika.

Pracownik zaczyna pracę w poniedziałek i pracuje sześć dni w tygodniu. Zmienna systemowa `FirstWeekDay` nie może być modyfikowana.

Użytkownik końcowy potrzebuje obiektu wykresu, który pokazuje zarobki do określonego dnia tygodnia według identyfikatora oraz imienia i nazwiska pracownika.

Skrypt ładowania

```
Payroll:
Load
*
Inline
[
employee_id,employee_name,day_rate
182,Mark, $150
183,Deryck, $125
184,Dexter, $125
185,Sydney,$270
186,Agatha,$128
];
```

Wyniki

Wykonaj następujące czynności:

1. Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:
   - `employee_id`
   - `employee_name`
2. Następnie utwórz miarę obliczającą zarobki do określonego dnia tygodnia: `=if(today(1)-weekstart(today(1),0,0)<7,(today(1)-weekstart(today(1),0,0))*day_rate,day_rate*6)`
3. Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

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

Funkcja `weekstart()`, przyjmująca bieżącą datę jako pierwszy argument i 0 jako trzeci argument, ustawia poniedziałek jako pierwszy dzień tygodnia i zwraca datę początku bieżącego tygodnia. Odejmując ten wynik od bieżącej daty, wyrażenie zwraca liczbę dni, które upłynęły do tej pory w tym tygodniu.

Następnie warunek sprawdza, czy w tym tygodniu było więcej niż sześć dni. Jeśli tak, wartość `day_rate` pracownika zostaje pomnożona przez 6 dni. W przeciwnym razie wartość `day_rate` zostaje pomnożona przez liczbę dni, jakie upłynęły do tej pory w tym tygodniu.

weekyear

Ta funkcja zwraca rok, do którego należy numer tygodnia zgodnie z normą ISO 8601. Numer tygodnia należy do zakresu od 1 do około 52.

Składnia:

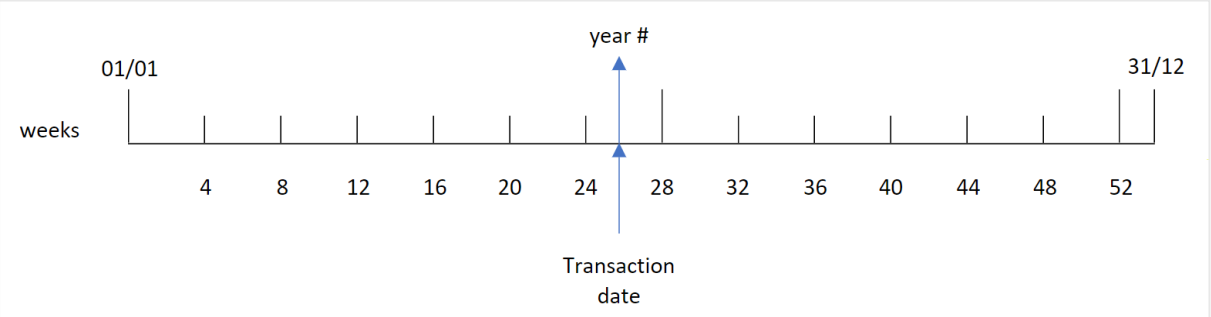
```
weekyear(expression)
```

Typ zwracanych danych: integer

Funkcja `weekyear()` określa, w którym tygodniu roku wypada określona data. Zwraca rok odpowiadający danemu numerowi tygodnia.

Domyślnie aplikacje Qlik używają tygodni dzielonych (zdefiniowanych przez zmienną systemową `BrokenWeeks`) i pierwszy tydzień zaczyna się 1 stycznia, a rok kończy się po tygodniu 52. W związku z tym, jeśli aplikacja używa tygodni dzielonych, funkcja `weekyear()` zawsze zwraca taką samą wartość, jak funkcja `week()`.

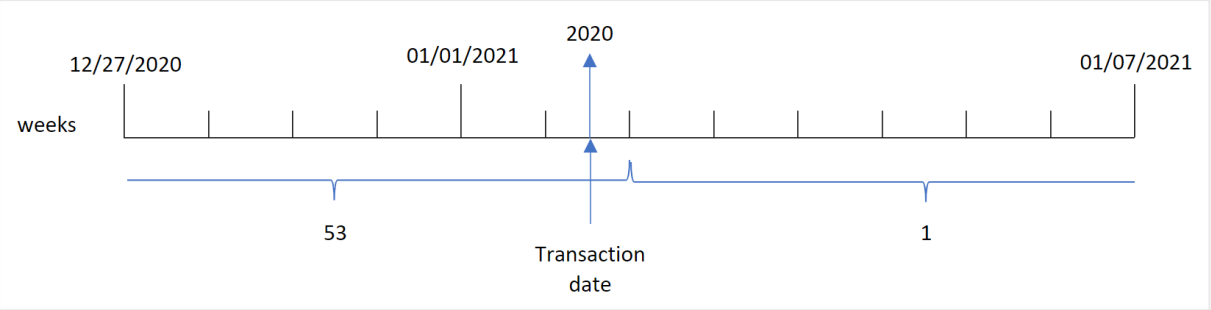
Diagram zakresu funkcji `weekyear()`



Jeśli jednak zmienna systemowa `BrokenWeeks` zostanie ustawiona na używanie tygodni niedzielonych, pierwszy tydzień będzie musiał zawierać tylko pewną liczbę dni w styczniu, określoną przez wartość zmiennej systemowej `ReferenceDay`.

Na przykład, jeśli argumentowi `ReferenceDay` zostanie nadana wartość `4`, pierwszy tydzień będzie musiał zawierać przynajmniej cztery dni w styczniu. Pierwszy tydzień może obejmować daty grudniowe z poprzedniego roku, a ostatni tydzień może obejmować daty ze stycznia następnego roku. W takich sytuacjach funkcja `weekyear()` zwróci inną wartość niż funkcja `year()`.

Diagram zakresu funkcji weekyear(), kiedy używane są tygodnie niedzielone



Kiedy używać

Funkcja `weekyear()` jest przydatna, gdy chcesz porównać agregacje według lat. Na przykład, jeśli chcesz zobaczyć całkowitą sprzedaż produktów w ujęciu rocznym. Funkcję `weekyear()` należy wybrać zamiast funkcji `year()`, gdy użytkownik chce zachować spójność ze zmienną systemową `BrokenWeeks` w aplikacji.

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
`weekyear ('12/30/1996')`	Zwraca 1997, ponieważ pierwszy tydzień 1997 roku zaczyna się dnia 30.12.1996 roku.
`weekyear ('01/02/1997')`	Zwraca wartość 1997
`weekyear ('12/28/1997')`	Zwraca wartość 1997
`weekyear ('12/30/1997')`	Zwraca 1998, ponieważ pierwszy tydzień 1998 roku zaczyna się 29.12.1997 roku.
`weekyear ('01/02/1999')`	zwraca 1998, ponieważ 53. tydzień 1998 roku kończy się 03.01.1990 roku.

Tematy pokrewne

Temat	Interakcja
week (page 1034)	Zwraca liczbę całkowitą reprezentującą numer tygodnia zgodnie z normą ISO 8601.
year (page 1107)	Zwraca liczbę całkowitą reprezentującą rok, gdy wyrażenie jest interpretowane jako data zgodnie ze standardową interpretacją liczb.

Przykład 1 – Tygodnie dzielone

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za ostatni tydzień 2020 r. i pierwszy tydzień 2021 r., który jest ładowany do tabeli o nazwie „`Transactions`”.
- Zmienna `BrokenWeeks`, która jest ustawiona na 1.
- Ładunek poprzedzający, który zawiera następujące elementy:
  - Funkcja `weekyear()` ustawiona jako pole „`week_year`”, która zwraca rok, w którym miały miejsce transakcje.
  - Funkcja `week()`, ustawiona jako pole `week` pokazujące numer tygodnia każdej daty transakcji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `week`
- `week_year`

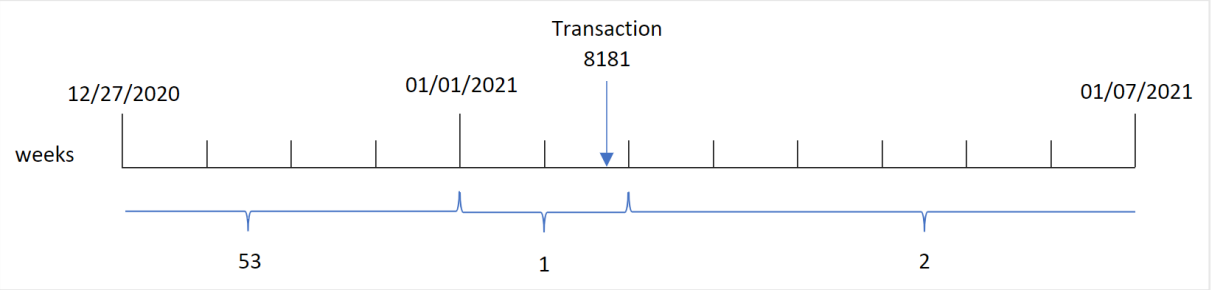
Tabela wynikowa

id	date	tydzień	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

Pole „`week_year`" jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `weekyear()` i przekazanie pola daty jako jej argumentu.

Zmienna systemowa `BrokenWeeks` jest ustawiona na `1`, co znaczy, że aplikacja używa tygodni dzielonych. Pierwszy tydzień zaczyna się 1 stycznia.

Diagram zakresu funkcji `weekyear()`, kiedy używane są tygodnie dzielone



Transakcja 8181 ma miejsce 2 stycznia, czyli w dniu należącym do pierwszego tygodnia. W związku z tym zwraca wartość 2021 dla pola „`week_year`”.

Przykład 2 – Tygodnie niedzielone

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za ostatni tydzień 2020 r. i pierwszy tydzień 2021 r., który jest ładowany do tabeli o nazwie „`Transactions`”.
- Zmienna `BrokenWeeks`, która jest ustawiona na `0`.
- Ładunek poprzedzający, który zawiera następujące elementy:
  - Funkcja `weekyear()` ustawiona jako pole „`week_year`”, która zwraca rok, w którym miały miejsce transakcje.
  - Funkcja `week()`, ustawiona jako pole `week` pokazujące numer tygodnia każdej daty transakcji.

Jednak w tym przykładzie polityką firmy jest używać tygodni niedzielonych.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `week`
- `week_year`

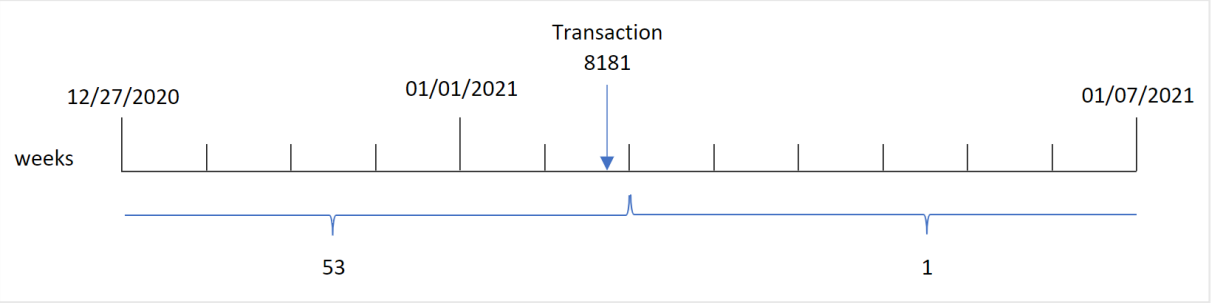
Tabela wynikowa

id	date	tydzień	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

Zmienna systemowa `BrokenWeeks` jest ustawiona na `0`, co znaczy, że aplikacja używa tygodni niedzielonych. W związku z tym pierwszy tydzień nie musi zaczynać się 1 stycznia.

Tydzień 53. 2020 roku trwa do końca 2 stycznia 2021 roku, a tydzień 1. 2020 roku zaczyna się w niedzielę 3 stycznia 2021 roku.

Diagram zakresu funkcji weekyear(), kiedy używane są tygodnie niedzielone



Transakcja 8181 ma miejsce 2 stycznia, czyli w dniu należącym do pierwszego tygodnia. W związku z tym zwraca wartość 2021 dla pola „week_year”.

Przykład 3 – Przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie zestaw danych pozostaje bez zmian i jest ładowany do aplikacji. Obliczenia zwracające numer tygodnia roku, w którym miały miejsce transakcje, są tworzone jako miara na wykresie w aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- id
- date

Aby obliczyć tydzień, w którym ma miejsce transakcja, utwórz następującą miarę:

- =week(date)

Aby obliczyć rok, w którym ma miejsce transakcja, na podstawie numeru tygodnia, utwórz następującą miarę:

- =weekyear(date)

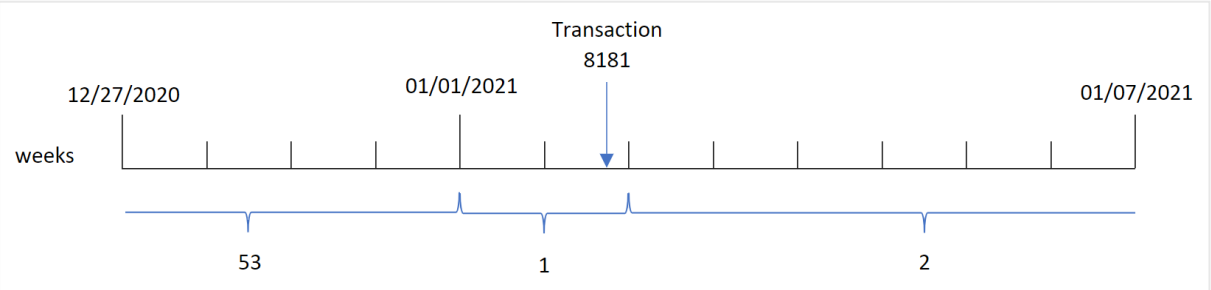
Tabela wynikowa

id	date	tydzień	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

Pole „week_year" jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `weekyear()` i przekazanie pola daty jako jej argumentu.

Zmienna systemowa `BrokenWeeks` jest ustawiona na `1`, co znaczy, że aplikacja używa tygodni dzielonych. Pierwszy tydzień zaczyna się 1 stycznia.

Diagram zakresu funkcji `weekyear()`, kiedy używane są tygodnie dzielone



Transakcja 8181 ma miejsce 2 stycznia, czyli w dniu należącym do pierwszego tygodnia. W związku z tym zwraca wartość 2021 dla pola „week_year”.

Przykład 4 – Scenariusz

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za ostatni tydzień 2020 r. i pierwszy tydzień 2021 r., który jest ładowany do tabeli o nazwie „Transactions”.
- Zmienna BrokenWeeks, która jest ustawiona na 0. Zo znaczy, że aplikacja będzie używać tygodni niedzielonych.
- Zmienna ReferenceDay, która jest ustawiona na 2. To znaczy, że rok rozpocznie się 2 stycznia i będzie miał przynajmniej dwa dni w styczniu.
- Zmienna FirstWeekDay, która jest ustawiona na 1. To znaczy, że pierwszym dniem tygodnia będzie wtorek.

Firma przyjęła zasadę, że używa tygodni dzielonych. Użytkownik końcowy chciałby otrzymać wykres przedstawiający łączną sprzedaż w ujęciu rocznym. Aplikacja używa tygodni niedzielonych i pierwszy tydzień musi mieć przynajmniej dwa dni w styczniu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę.

Aby obliczyć rok, w którym ma miejsce transakcja, na podstawie numeru tygodnia, utwórz następującą miarę:

- `=weekyear(date)`

Aby obliczyć łączną sprzedaż, utworzyć następującą miarę:

- `sum(amount)`

Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

weekyear(date)	=sum(amount)
2020	19.42
2021	373.37

year

Ta funkcja zwraca liczbę całkowitą reprezentującą rok, gdy wyrażenie **expression** jest interpretowane jako data zgodnie ze standardową interpretacją liczb.

Składnia:

```
year(expression)
```

Typ zwracanych danych: integer

Funkcja `year()` jest dostępna zarówno jako skrypt, jak i funkcja wykresu. Funkcja ta zwraca rok dla określonej daty. Jest używana do tworzenia pola roku jako wymiaru w kalendarzu głównym.

Kiedy używać

Funkcja `year()` jest przydatna, gdy chcesz porównać agregacje w ujęciu rocznym. Można jej na przykład użyć, aby wyświetlić całkowitą sprzedaż produktów w ujęciu rocznym.

Te wymiary można utworzyć w skrypcie ładowania za pomocą funkcji tworzenia pola w tabeli kalendarza głównego. Ewentualnie można jej użyć bezpośrednio na wykresie jako wymiaru wyliczanego.

Przykłady funkcji

Przykład	Wynik
`year( '2012-10-12' )`	zwraca 2012
`year( '35648' )`	zwraca 1997, ponieważ 35648 = 1997-08-06

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – zbiór danych DateFormat (skrypt)

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający daty, który jest załadowany do tabeli o nazwie `Master Calendar`.
- Została użyta domyślna zmienna systemowa `DateFormat` MM/DD/RRRR.
- Poprzednie polecenie LOAD, tworzące dodatkowe pole (`year`) przy użyciu funkcji `year()`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `year`

Tabela wynikowa

date	rok
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

Przykład 2 – Daty ANSI

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający daty, który jest załadowany do tabeli o nazwie `Master Calendar`.
- Została użyta domyślna zmienna systemowa `DateFormat` MM/DD/RRRR. Natomiast daty w zbiorze danych są w standardowym formacie daty ANSI.
- Poprzednie polecenie LOAD, tworzące dodatkowe pole (o nazwie `year`) przy użyciu funkcji `year()`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `year`

Tabela wynikowa

date	rok
2020-12-28	2020
2020-12-29	2020
2020-12-30	2020
2020-12-31	2020
2021-01-01	2021
2021-01-02	2021
2021-01-03	2021
2021-01-04	2021
2021-01-05	2021

date	rok
2021-01-06	2021
2021-01-07	2021

Przykład 3 – Niesformatowane daty

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający daty w formacie numerycznym, który jest załadowany do tabeli o nazwie `Master Calendar`.
- Została użyta domyślna zmienna systemowa `DateFormat` MM/DD/RRRR.
- Poprzednie polecenie LOAD, tworzące dodatkowe pole (`year`) przy użyciu funkcji `year()`.

Oryginalna niesformatowana data zostaje załadowana i nazwana `unformatted_date`. Ponadto, dla jasności, zostało użyte dodatkowe pole, o nazwie `long_date`, które służy do konwersji numerycznej daty na sformatowane pole daty za pomocą funkcji `date()`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `unformatted_date`
- `long_date`
- `year`

Tabela wynikowa

unformatted_date	long_date	rok
44868	11/03/2022	2022
44898	12/03/2022	2022
44928	01/02/2023	2023
44958	02/01/2023	2023
44988	03/03/2023	2023
45008	03/23/2023	2023
45018	04/02/2023	2023
45038	04/22/2023	2023
45048	05/02/2023	2023
45068	05/22/2023	2023
45078	06/01/2023	2023

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

W tym przykładzie zestaw danych złożonych zamówień jest ładowany do tabeli o nazwie Sales. Tabela zawiera trzy pola:

- `id`
- `sales_date`
- `amount`

Sprzedawane produkty są objęte dwuletnią gwarancją od daty sprzedaży. Zadanie dotyczy utworzenia na wykresie miary określającej rok zakończenia okresu każdej gwarancji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `sales_date`.

Utwórz następującą miarę:

```
=year(sales_date+365*2)
```

Tabela wynikowa

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

Wyniki tej miary można zobaczyć w powyższej tabeli. Aby dodać dwa lata do daty, należy pomnożyć 365 przez 2 i dodać wynik do daty sprzedaży. W związku z tym, w przypadku produktu sprzedanego w 2020 roku termin zakończenia wypada w 2022 roku.

yearend

Ta funkcja zwraca wartość odpowiadającą znacznikowi czasu ostatniej milisekundy ostatniego dnia roku zawierającego wartość **date**. Domyślnym formatem wyjściowym będzie format **DateFormat** skonfigurowany w skrypcie.

Składnia:

```
YearEnd( date[, period_no[, first_month_of_year = 1]])
```

Innymi słowy, funkcja `yearend()` określa, na który rok przypada data. Następnie zwraca znacznik czasu w formacie daty dla ostatniej milisekundy tego roku. Pierwszym miesiącem roku jest domyślnie styczeń. Możesz jednak również zmienić pierwszy dzień tygodnia, używając argumentu `first_month_of_year` w funkcji `yearend()`.

> *Funkcja `yearend()` nie uwzględnia zmiennej systemowej `FirstMonthOfYear`. Rok rozpoczyna się 1 stycznia, chyba że do jego zmiany użyto argumentu `first_month_of_year`.*

Schemat funkcji `yearend()`.

Kiedy używać

Funkcja `yearend()` jest powszechnie używana jako część wyrażenia, gdy użytkownik chce, by w obliczeniach użyto ułamka roku, który jeszcze nie nastąpił. Na przykład, jeśli chcesz obliczyć łączne odsetki, które nie zostały jeszcze naliczone w ciągu roku.

Typ zwracanych danych: dual

Argumenty

Argument	Opis
date	Data lub znacznik czasu do oszacowania.
period_no	**period_no** jest liczbą całkowitą, gdzie 0 oznacza rok, który zawiera datę **date**. Wartości ujemne parametru **period_no** oznaczają lata poprzednie, a wartości dodatnie – lata następne.

Argument	Opis
first_month_of_year	Jeśli użytkownik zamierza korzystać z lat (obrotowych), które nie zaczynają się w styczniu, powinien wskazać wartość od 2 do 12 jako parametr **first_month_of_year**.

Aby ustawić pierwszy miesiąc roku w argumencie `first_month_of_year`, możesz użyć następujących wartości:

first_month_of_year values

Miesiąc	Wartość
Luty	2
Marzec	3
Kwiecień	4
May	5
Czerwiec	6
Lipiec	7
Sierpień	8
Wrzesień	9
Październik	10
Listopad	11
Grudzień	12

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
`yearend('10/19/2001')`	Returns 12/31/2001 23:59:59.

Przykład	Wynik
`yearend('10/19/2001', -1)`	Returns 12/31/2000 23:59:59.
`yearend('10/19/2001', 0, 4)`	Returns 03/31/2002 23:59:59.

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za lata 2020-2022 jest ładowany do tabeli o nazwie „`Transactions`”.
- Pole daty zostało podane w formacie zmiennej systemowej `DateFormat (MM/DD/YYYY)`.
- Poprzedzająca instrukcja LOAD, która zawiera:
  - Funkcję `yearend()`, która jest ustawiona jako pole `year_end`.
  - Funkcję `Timestamp()`, która jest ustawiona jako pole `year_end_timestamp`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `year_end`
- `year_end_timestamp`

Tabela wynikowa

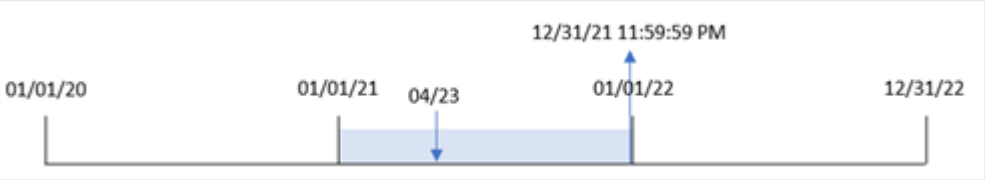
id	date	year_end	year_end_timestamp
8188	01/13/2020	12/31/2020	12/31/2020 11:59:59 PM
8189	02/26/2020	12/31/2020	12/31/2020 11:59:59 PM
8190	03/27/2020	12/31/2020	12/31/2020 11:59:59 PM
8191	04/16/2020	12/31/2020	12/31/2020 11:59:59 PM
8192	05/21/2020	12/31/2020	12/31/2020 11:59:59 PM
8193	08/14/2020	12/31/2020	12/31/2020 11:59:59 PM
8194	10/07/2020	12/31/2020	12/31/2020 11:59:59 PM
8195	12/05/2020	12/31/2020	12/31/2020 11:59:59 PM
8196	01/22/2021	12/31/2021	12/31/2021 11:59:59 PM
8197	02/03/2021	12/31/2021	12/31/2021 11:59:59 PM
8198	03/17/2021	12/31/2021	12/31/2021 11:59:59 PM
8199	04/23/2021	12/31/2021	12/31/2021 11:59:59 PM
8200	05/04/2021	12/31/2021	12/31/2021 11:59:59 PM
8201	06/30/2021	12/31/2021	12/31/2021 11:59:59 PM
8202	07/26/2021	12/31/2021	12/31/2021 11:59:59 PM
8203	12/27/2021	12/31/2021	12/31/2021 11:59:59 PM
8204	06/06/2022	12/31/2022	12/31/2022 11:59:59 PM

id	date	year_end	year_end_timestamp
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

Pole „year_end" jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `yearend()` i przekazanie pola daty jako jej argumentu.

Funkcja `yearend()` początkowo identyfikuje, w którym roku przypada wartość daty, i zwraca znacznik czasu ostatniej milisekundy danego roku.

Diagram funkcji `yearend()` z wybraną transakcją 8199.



Transakcja 8199 miała miejsce 23 kwietnia 2021 r. Funkcja `yearend()` zwraca ostatnią milisekundę tego roku, czyli 31 grudnia o godz 23:59:59 (11:59:59 PM).

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie zadaniem jest utworzenie pola „previous_year_end", które zwraca znacznik czasu daty końcowej roku poprzedzającego rok, w którym miała miejsce transakcja.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `previous_year_end`
- `previous_ year_end_timestamp`

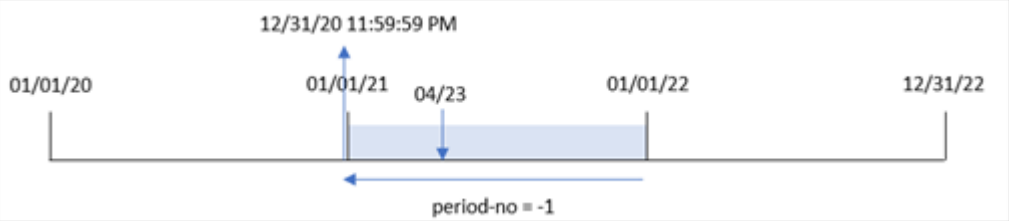
Tabela wynikowa

id	date	previous_year_end	previous_year_end_timestamp
8188	01/13/2020	12/31/2019	12/31/2019 11:59:59 PM
8189	02/26/2020	12/31/2019	12/31/2019 11:59:59 PM
8190	03/27/2020	12/31/2019	12/31/2019 11:59:59 PM
8191	04/16/2020	12/31/2019	12/31/2019 11:59:59 PM
8192	05/21/2020	12/31/2019	12/31/2019 11:59:59 PM
8193	08/14/2020	12/31/2019	12/31/2019 11:59:59 PM
8194	10/07/2020	12/31/2019	12/31/2019 11:59:59 PM
8195	12/05/2020	12/31/2019	12/31/2019 11:59:59 PM
8196	01/22/2021	12/31/2020	12/31/2020 11:59:59 PM
8197	02/03/2021	12/31/2020	12/31/2020 11:59:59 PM
8198	03/17/2021	12/31/2020	12/31/2020 11:59:59 PM
8199	04/23/2021	12/31/2020	12/31/2020 11:59:59 PM

id	date	previous_year_end	previous_year_end_timestamp
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

Ponieważ `period_no` o wartości `-1` użyto jako argumentu przesunięcia w funkcji `yearend()`, funkcja najpierw identyfikuje rok, w którym odbywają się transakcje. Następnie identyfikuje ostatnią milisekundę poprzedniego roku.

Diagram funkcji `yearend()` z period_no o wartości -1.



Transakcja 8199 ma miejsce 23 kwietnia 2021 r. Funkcja `yearend()` zwraca ostatnią milisekundę poprzedniego roku, czyli 31 grudnia 2020 r. o godz 23:59:59 (11:59:59 PM), dla pola „`previous_year_end`”.

Przykład 3 – first_month_of_year

Skrypt ładowania i wyniki

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

W tym przykładzie jednak, zgodnie z polityką firmy, rok zaczyna się 1 kwietnia.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `year_end`
- `year_end_timestamp`

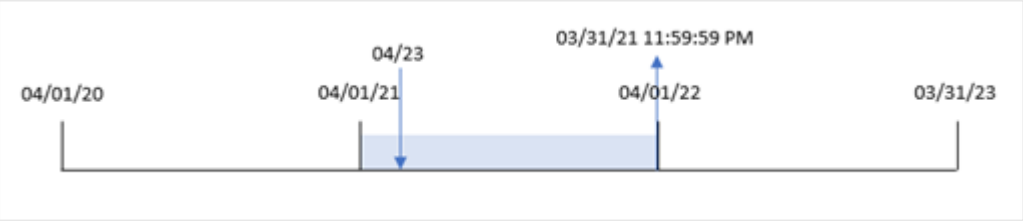
Tabela wynikowa

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

id	date	year_end	year_end_timestamp
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

Ponieważ argument `first_month_of_year` o wartości `4` jest używany w funkcji `yearend()`, ustawia pierwszy dzień roku na 1 kwietnia, a ostatni dzień roku na 31 marca.

Wykres funkcji `yearend()` z kwietniem jako pierwszym miesiącem roku.



Transakcja 8199 ma miejsce 23 kwietnia 2021 r. Ponieważ funkcja `yearend()` ustawia początek roku na 1 kwietnia, zwraca 31 marca 2022 r. jako wartość „`year_end`” dla transakcji.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie zestaw danych pozostaje bez zmian i jest ładowany do aplikacji. Obliczenie zwracające znacznik czasu końcowej daty roku, kiedy wystąpiła transakcja, jest tworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`

Aby obliczyć, w którym roku miała miejsce transakcja, utwórz następujące miary:

- `=yearend(date)`
- `=timestamp(yearend(date))`

Tabela wynikowa

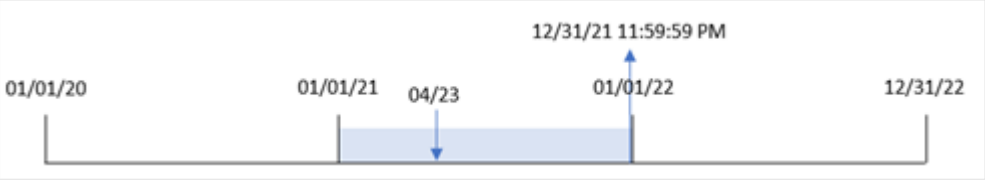
id	date	=yearend(date)	=timestamp(yearend(date))
8188	01/13/2020	12/31/2020	12/31/2020 11:59:59 PM
8189	02/26/2020	12/31/2020	12/31/2020 11:59:59 PM
8190	03/27/2020	12/31/2020	12/31/2020 11:59:59 PM
8191	04/16/2020	12/31/2020	12/31/2020 11:59:59 PM

id	date	=yearend(date)	=timestamp(yearend(date))
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

Miarę „end_of_year” tworzy się w obiekcie wykresu, używając funkcji `yearend()` i przekazując pole daty jako jej argument.

Funkcja `yearend()` początkowo identyfikuje, w którym roku przypada wartość daty, i zwraca znacznik czasu ostatniej milisekundy danego roku.

Diagram funkcji `yearend()` pokazujący, że transakcja 8199 miała miejsce w kwietniu.



Transakcja 8199 ma miejsce 23 kwietnia 2021 r. Funkcja `yearend()` zwraca ostatnią milisekundę tego roku, czyli 31 grudnia o godz 23:59:59 (11:59:59 PM).

Przykład 5 – scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych jest ładowany do tabeli o nazwie `Employee_Expenses`. Tabela zawiera następujące pola:
    - identyfikatory pracowników
    - imię i nazwisko pracownika
    - średnie dzienne rozliczenia wydatków każdego pracownika

Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał, według identyfikatora pracownika oraz imienia i nazwiska pracownika, szacowane roszczenia z tytułu wydatków do poniesienia przez pozostałą część roku. Rok obrotowy rozpoczyna się w styczniu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `employee_id`
- `employee_name`

Aby obliczyć przewidywane roszczenia z tytułu wydatków, utwórz następującą miarę:

```
=(yearend(today(1))-today(1))*avg_daily_claim
```

Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

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

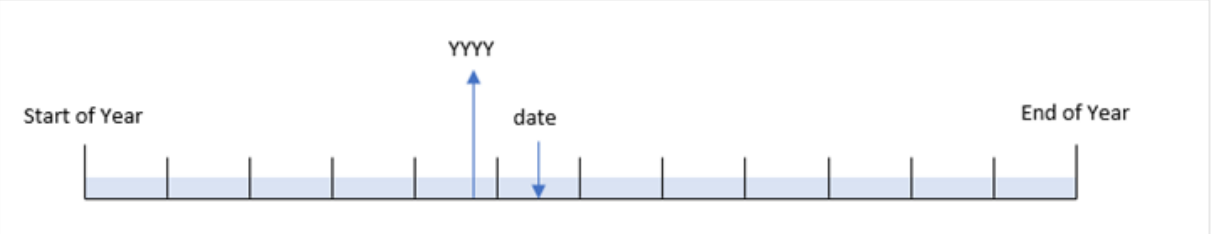
Używając dzisiejszej daty jako jedynego argumentu, funkcja `yearend()` zwraca datę końcową bieżącego roku. Następnie, odejmując dzisiejszą datę od daty zakończenia roku, wyrażenie zwraca liczbę dni pozostałych w tym roku.

Wartość ta jest następnie mnożona przez średnie dzienne roszczenie z tytułu wydatków przez każdego pracownika, aby obliczyć szacunkową wartość roszczeń, które każdy pracownik złoży w pozostałej części roku.

yearname

Ta funkcja zwraca rok w zapisie czterocyfrowym jako wartość wyświetlaną z bazową wartością liczbową odpowiadającą znacznikowi czasu pierwszej milisekundy pierwszego dnia roku zawierającego datę **date**.

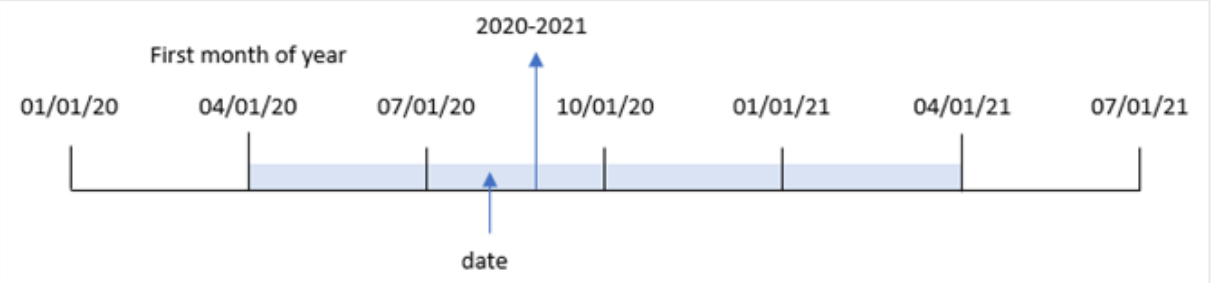
Wykres zakresu czasu funkcji `yearname()`.



Funkcja `yearname()` różni się od funkcji `year()`, ponieważ umożliwia przesunięcie daty, którą chcesz ocenić, oraz ustawienie pierwszego miesiąca roku.

Jeśli pierwszym miesiącem roku nie jest styczeń, funkcja zwróci dwa czterocyfrowe lata z dwunastomiesięcznego okresu zawierającego datę. Na przykład, jeśli początek roku to kwiecień, a ocenianą datą jest 06/30/2020, zwróconym wynikiem będzie `2020-2021`.

Wykres funkcji `yearname()` z kwietniem jako pierwszym miesiącem roku.



Składnia:

```
YearName(date[, period_no[, first_month_of_year]] )
```

Typ zwracanych danych: dual

Argument	Opis
date	Data lub znacznik czasu do oszacowania.
period_no	**period_no** jest liczbą całkowitą, gdzie 0 oznacza rok, który zawiera datę **date**. Wartości ujemne parametru **period_no** oznaczają lata poprzednie, a wartości dodatnie – lata następne.
first_month_of_year	Jeśli użytkownik zamierza korzystać z lat (obrotowych), które nie zaczynają się w styczniu, powinien wskazać wartość od 2 do 12 jako parametr **first_month_of_year**. Wartość wyświetlana będzie ciągiem znaków przedstawiającym dwa lata.

Aby ustawić pierwszy miesiąc roku w argumencie `first_month_of_year`, możesz użyć następujących wartości:

first_month_of_year values

Miesiąc	Wartość
Luty	2
Marzec	3
Kwiecień	4
May	5
Czerwiec	6
Lipiec	7
Sierpień	8
Wrzesień	9
Październik	10
Listopad	11
Grudzień	12

Kiedy używać

Funkcja `yearname()` przydaje się do porównywania agregacji według roku. Na przykład, jeśli chcesz zobaczyć całkowitą sprzedaż produktów według roku.

Te wymiary można utworzyć w skrypcie ładowania za pomocą funkcji tworzenia pola w tabeli kalendarza głównego. Można je również tworzyć na wykresie jako wymiary obliczone

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
`yearname('10/19/2001')`	Returns '2001.'
`yearname('10/19/2001',-1)`	Zwraca „2000”.
`yearname('10/19/2001',0,4)`	Zwraca „2001-2002”.

Tematy pokrewne

Temat	Opis
year (page 1107)	Ta funkcja zwraca liczbę całkowitą reprezentującą rok, gdy wyrażenie jest interpretowane jako data zgodnie ze standardową interpretacją liczb.

Przykład 1 – bez dodatkowych argumentów

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za lata 2020-2022 jest ładowany do tabeli o nazwie „Transactions”.
- Zmienna systemowa DateFormat, która jest ustawiona na „MM/DD/YYYY”.
- Poprzednie ładowanie używające funkcji yearname(), która jest ustawiona jako pole year_name.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- date
- year_name

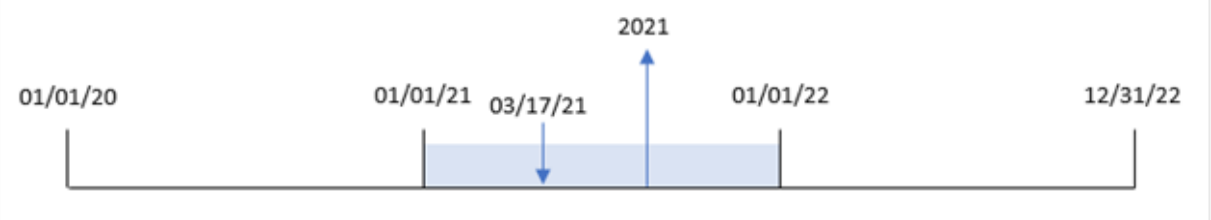
Tabela wynikowa

date	year_name
01/13/2020	2020
02/26/2020	2020
03/27/2020	2020
04/16/2020	2020
05/21/2020	2020
08/14/2020	2020
10/07/2020	2020
12/05/2020	2020
01/22/2021	2021
02/03/2021	2021
03/17/2021	2021
04/23/2021	2021
05/04/2021	2021
06/30/2021	2021
07/26/2021	2021
12/27/2021	2021
06/06/2022	2022
07/18/2022	2022
11/14/2022	2022
12/12/2022	2022

Pole „year_name" jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `yearname()` i przekazanie pola daty jako jej argumentu.

Funkcja `yearname()` identyfikuje rok, w którym przypada wartość daty, i zwraca go jako czterocyfrową wartość roku.

Schemat funkcji `yearname()`, która pokazuje 2021 jako wartość roku.



Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za lata 2020-2022 jest ładowany do tabeli o nazwie „Transactions”.
- Zmienna systemowa DateFormat, która jest ustawiona na „MM/DD/YYYY”.
- Poprzednie ładowanie używające funkcji `yearname()`, która jest ustawiona jako pole `year_name`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

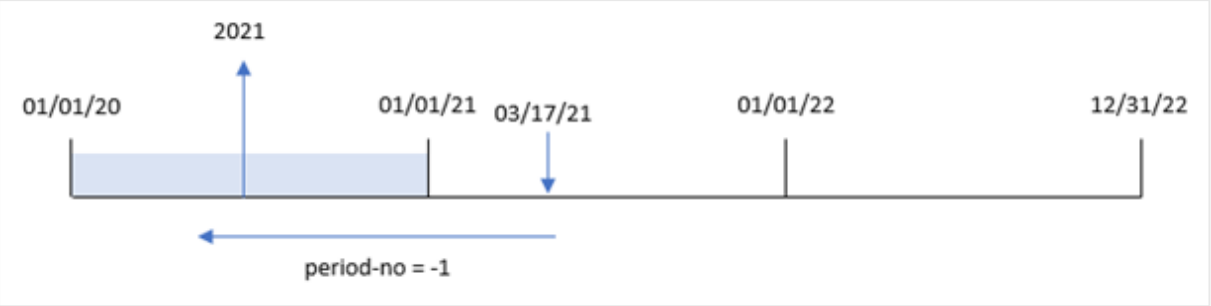
- `date`
- `prior_year_name`

Tabela wynikowa

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

Ponieważ `period_no` o wartości `-1` używa się jako argumentu przesunięcia w funkcji `yearname()`, funkcja najpierw identyfikuje rok, w którym odbywają się transakcje. Następnie funkcja dokonuje przesunięcia o rok wcześniej i zwraca wynikowy rok.

Diagram funkcji `yearname()` z `period_no` o wartości -1.



Przykład 3 – first_month_of_year

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zbiór danych, co w pierwszym przykładzie.
- Zmienna systemowa `DateFormat`, która jest ustawiona na „`MM/DD/YYYY`”.
- Poprzednie ładowanie używające funkcji `yearname()`, która jest ustawiona jako pole `year_name`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`
- `year_name`

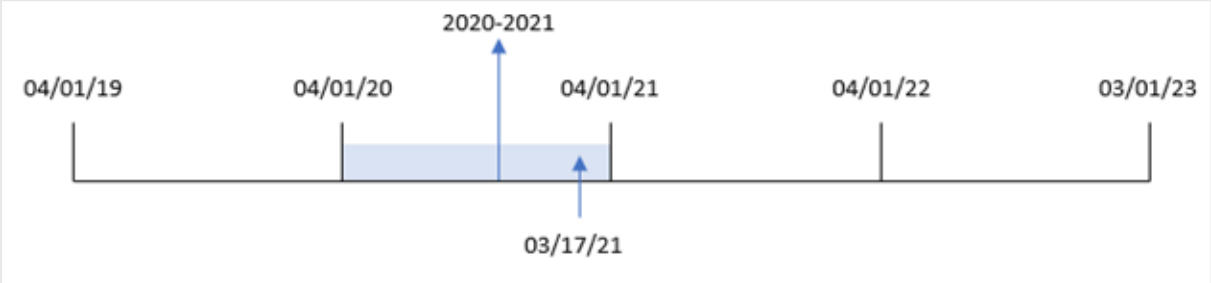
Tabela wynikowa

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

Ponieważ argument [`first_month_of_year` o wartości `4` jest używany w funkcji `yearname()`, początek roku przesuwa się z 1 stycznia na 1 kwietnia. Dlatego każdy okres dwunastu miesięcy przecina dwa lata kalendarzowe, a funkcja `yearname()` zwraca dwa czterocyfrowe lata dla ocenianych dat.

Transakcja 8198 ma miejsce 17 marca 2021 r. Funkcja `yearname()` ustawia początek roku na 1 kwietnia i koniec na 30 marca. W związku z tym transakcja 8198 miała miejsce w okresie od 1 kwietnia 2020 r. do 30 marca 2021 r. W rezultacie funkcja `yearname()` zwraca wartość 2020-2021.

Diagram funkcji `yearname()` z marcem jako pierwszym miesiącem roku.



Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zbiór danych, co w pierwszym przykładzie.
- Zmienna systemowa `DateFormat`, która jest ustawiona na „`MM/DD/YYYY`”.

Jednak pole zwracające rok, w którym miała miejsce transakcja, jest tworzone jako miara w obiekcie wykresu.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar:

`date`

Aby obliczyć pole `year_name`, należy utworzyć następującą miarę:

```
=yearname(date)
```

Tabela wynikowa

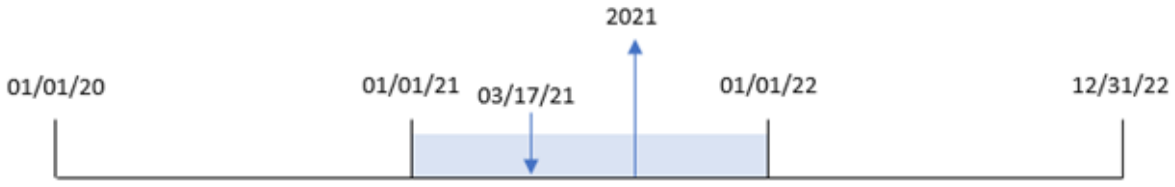
date	=yearname(date)
01/13/2020	2020
02/26/2020	2020
03/27/2020	2020
04/16/2020	2020
05/21/2020	2020
08/14/2020	2020
10/07/2020	2020
12/05/2020	2020
01/22/2021	2021
02/03/2021	2021
03/17/2021	2021
04/23/2021	2021
05/04/2021	2021
06/30/2021	2021
07/26/2021	2021
12/27/2021	2021

date	=yearname(date)
06/06/2022	2022
07/18/2022	2022
11/14/2022	2022
12/12/2022	2022

Miarę „year_name" tworzy się w obiekcie wykresu, używając funkcji `yearname()` i przekazując pole daty jako jej argument.

Funkcja `yearname()` identyfikuje rok, w którym przypada wartość daty, i zwraca go jako czterocyfrową wartość roku.

Diagram funkcji `yearname()`, który pokazuje 2021 jako wartość roku.



Przykład 5 – scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zbiór danych, co w pierwszym przykładzie.
- Zmienna systemowa `DateFormat`, która jest ustawiona na „`MM/DD/YYYY`".

Użytkownik końcowy chciałby otrzymać wykres przedstawiający łączną sprzedaż w ujęciu kwartalnym dla transakcji. Użyj funkcji `yearname()` jako wymiaru wyliczanego, aby utworzyć ten wykres, gdy wymiar `yearname()` nie jest dostępny w modelu danych.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę.

Aby porównać agregacje według roku, utwórz ten wyliczony wymiar:

```
=yearname(date)
```

Utwórz tę miarę:

```
=sum(amount)
```

Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

yearname(date)	=sum(amount)
2020	$463.55
2021	$457.69
2022	$294.35

yearstart

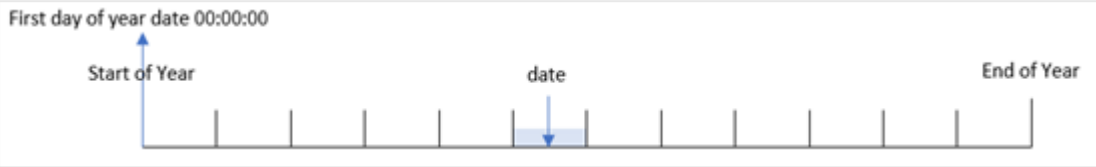
Ta funkcja zwraca znacznik czasu odpowiadający rozpoczęciu się pierwszego dnia roku zawierającego wartość **date**. Domyślnym formatem wyjściowym będzie format **DateFormat** skonfigurowany w skrypcie.

Składnia:

```
YearStart(date[, period_no[, first_month_of_year]])
```

Innymi słowy, funkcja `yearstart()` określa, na który rok przypada data. Następnie zwraca znacznik czasu w formacie daty dla pierwszej milisekundy tego roku. Pierwszym miesiącem roku jest domyślnie styczeń, ale możesz zmienić pierwszy dzień tygodnia, używając argumentu `first_month_of_year` w funkcji `yearstart()`.

Diagram funkcji `yearstart()` pokazujący zakres czasu, który funkcja może objąć.



Kiedy używać

Funkcja `yearstart()` jest używana jako część wyrażenia, gdy użytkownik chce, by w obliczeniach użyto ułamka roku, który już upłynął. Na przykład, jeśli chcesz obliczyć odsetki narosłe w ciągu ostatniego roku.

Typ zwracanych danych: dual

Argumenty

Argument	Opis
date	Data lub znacznik czasu do oszacowania.
period_no	**period_no** jest liczbą całkowitą, gdzie 0 oznacza rok, który zawiera datę **date**. Wartości ujemne parametru **period_no** oznaczają lata poprzednie, a wartości dodatnie – lata następne.
first_month_of_year	Jeśli użytkownik zamierza korzystać z lat (obrotowych), które nie zaczynają się w styczniu, powinien wskazać wartość od 2 do 12 jako parametr **first_month_of_year**.

Kolejne miesiące można wykorzystać w `first_month_of_year argument`:

first_month_of_year values

Miesiąc	Wartość
Luty	2
Marzec	3
Kwiecień	4
May	5
Czerwiec	6
Lipiec	7
Sierpień	8
Wrzesień	9

Miesiąc	Wartość
Październik	10
Listopad	11
Grudzień	12

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykłady funkcji

Przykład	Wynik
`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.

Przykład 1 – Przykład podstawowy

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za lata 2020-2022 jest ładowany do tabeli o nazwie „`Transactions`”.
- Pole daty zostało podane w formacie zmiennej systemowej `DateFormat` (`MM/DD/YYYY`).
- Poprzedzająca instrukcja LOAD, która zawiera:
  - Funkcję `yearstart()`, która jest ustawiona jako pole `year_start`.
  - Funkcję `Timestamp()`, która jest ustawiona jako pole `year_start_timestamp`.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- id
- date
- year_start
- year_start_timestamp

Tabela wynikowa

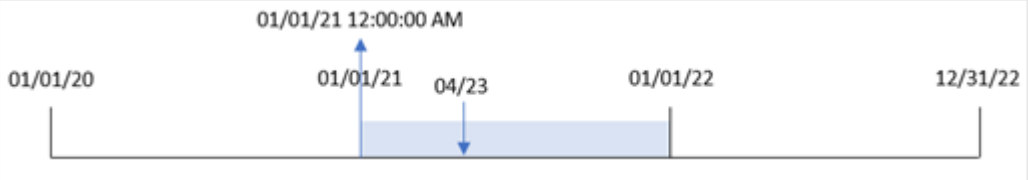
id	date	year_start	year_start_timestamp
8188	01/13/2020	01/01/2020	1/1/2020 12:00:00 AM

id	date	year_start	year_start_timestamp
8189	02/26/2020	01/01/2020	1/1/2020 12:00:00 AM
8190	03/27/2020	01/01/2020	1/1/2020 12:00:00 AM
8191	04/16/2020	01/01/2020	1/1/2020 12:00:00 AM
8192	05/21/2020	01/01/2020	1/1/2020 12:00:00 AM
8193	08/14/2020	01/01/2020	1/1/2020 12:00:00 AM
8194	10/07/2020	01/01/2020	1/1/2020 12:00:00 AM
8195	12/05/2020	01/01/2020	1/1/2020 12:00:00 AM
8196	01/22/2021	01/01/2021	1/1/2021 12:00:00 AM
8197	02/03/2021	01/01/2021	1/1/2021 12:00:00 AM
8198	03/17/2021	01/01/2021	1/1/2021 12:00:00 AM
8199	04/23/2021	01/01/2021	1/1/2021 12:00:00 AM
8200	05/04/2021	01/01/2021	1/1/2021 12:00:00 AM
8201	06/30/2021	01/01/2021	1/1/2021 12:00:00 AM
8202	07/26/2021	01/01/2021	1/1/2021 12:00:00 AM
8203	12/27/2021	01/01/2021	1/1/2021 12:00:00 AM
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

Pole „year_start” jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `yearstart()` i przekazanie pola daty jako jej argumentu.

Funkcja `yearstart()` początkowo identyfikuje, w którym roku przypada wartość daty, i zwraca znacznik czasu pierwszej milisekundy danego roku.

Diagram funkcji `yearstart()` i transakcji 8199.



Transakcja 8199 miała miejsce 23 kwietnia 2021 r. Funkcja `yearstart()` zwraca pierwszą milisekundę tego roku, czyli 1 stycznia o godz 12:00:00 AM (00:00:00)

Przykład 2 – period_no

Skrypt ładowania i wyniki

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie zadaniem jest utworzenie pola „previous_year_start”, które zwraca znacznik czasu daty początkowej roku poprzedzającego rok, w którym miała miejsce transakcja.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

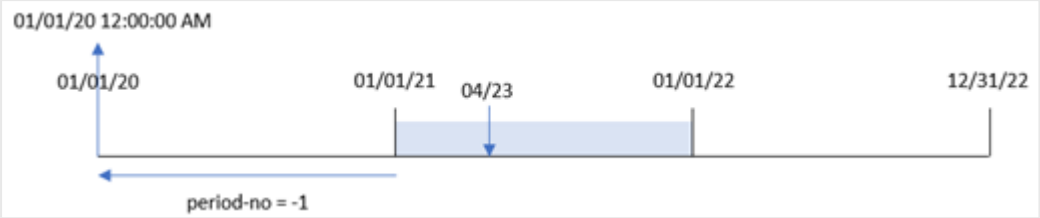
- id
- date
- previous_year_start
- previous_ year_start_timestamp

Tabela wynikowa

id	date	previous_year_start	previous_year_start_timestamp
8188	01/13/2020	01/01/2019	1/1/2019 12:00:00 AM
8189	02/26/2020	01/01/2019	1/1/2019 12:00:00 AM
8190	03/27/2020	01/01/2019	1/1/2019 12:00:00 AM
8191	04/16/2020	01/01/2019	1/1/2019 12:00:00 AM
8192	05/21/2020	01/01/2019	1/1/2019 12:00:00 AM
8193	08/14/2020	01/01/2019	1/1/2019 12:00:00 AM
8194	10/07/2020	01/01/2019	1/1/2019 12:00:00 AM
8195	12/05/2020	01/01/2019	1/1/2019 12:00:00 AM
8196	01/22/2021	01/01/2020	1/1/2020 12:00:00 AM
8197	02/03/2021	01/01/2020	1/1/2020 12:00:00 AM
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

W tym przypadku, ponieważ `period_no` o wartości `-1` używa się jako argumentu przesunięcia w funkcji `yearstart()`, funkcja najpierw identyfikuje rok, w którym odbywają się transakcje. Następnie identyfikuje pierwszą milisekundę poprzedniego roku.

Diagram funkcji `yearstart()` z `period_no` o wartości -1.



Transakcja 8199 miała miejsce 23 kwietnia 2021 r. Funkcja `yearstart()` zwraca pierwszą milisekundę poprzedniego roku, czyli 1 stycznia 2020 r. o godz 00:00:00 (12:00:00 AM), dla pola „`previous_year_start`”.

Przykład 3 – first_month_of_year

Skrypt ładowania i wyniki

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

W tym przykładzie jednak, zgodnie z polityką firmy, rok zaczyna się 1 kwietnia.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

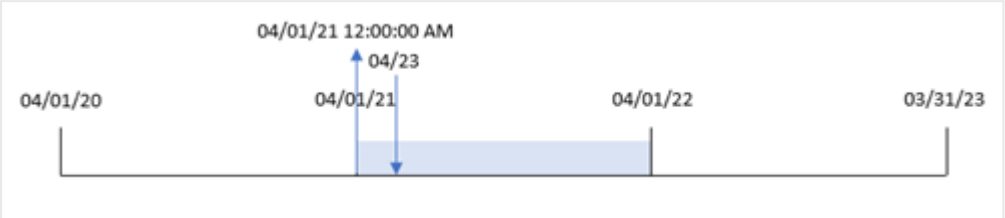
- id
- date
- year_start
- year_start_timestamp

Tabela wynikowa

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

W tym przypadku, ponieważ argument `first_month_of_year` o wartości `4` jest używany w funkcji `yearstart()`, ustawia pierwszy dzień roku na 1 kwietnia, a ostatni dzień roku na 31 marca.

Diagram funkcji `yearstart()` z kwietniem jako pierwszym miesiącem roku.



Transakcja 8199 miała miejsce 23 kwietnia 2021 r. Ponieważ funkcja `yearstart()` ustawia początek roku na 1 kwietnia i zwraca go jako wartość „`year_start`" dla transakcji.

Przykład 4 – przykład z obiektem wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Używany jest ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie zestaw danych pozostaje bez zmian i jest ładowany do aplikacji. Obliczenie zwracające znacznik czasu początkowej daty roku, kiedy wystąpiła transakcja, jest tworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `id`
- `date`

Aby obliczyć, w którym roku miała miejsce transakcja, utwórz następujące miary:

- `=yearstart(date)`
- `=timestamp(yearstart(date))`

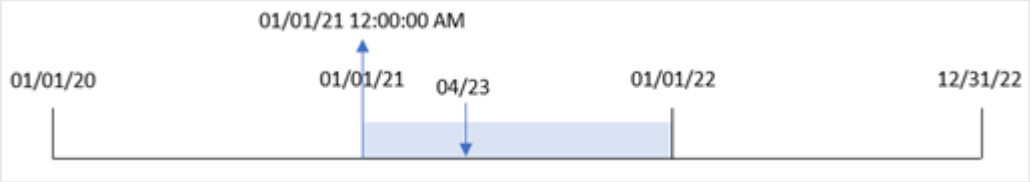
Tabela wynikowa

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

Miarę „start_of_year" tworzy się w obiekcie wykresu, używając funkcji `yearstart()` i przekazując pole daty jako jej argument.

Funkcja `yearstart()` początkowo identyfikuje, w którym roku przypada wartość daty, i zwraca znacznik czasu pierwszej milisekundy danego roku.

Diagram funkcji `yearstart()` i transakcji 8199.



Transakcja 8199 miała miejsce 23 kwietnia 2021 r. Funkcja `yearstart()` zwraca pierwszą milisekundę tego roku, czyli 1 stycznia o godz 12:00:00 AM (00:00:00)

Przykład 5 – scenariusz

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych jest ładowany do tabeli o nazwie `Loans`. Tabela zawiera następujące pola:
  - Loan IDs.
  - Bilans na początek roku.
  - Prosta stopa procentowa naliczana od każdej pożyczki w skali roku.

Użytkownik końcowy chciałby, aby obiekt wykresu wyświetlał według identyfikatora pożyczki bieżące odsetki naliczone od każdej pożyczki w bieżącym roku.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `loan_id`
- `start_balance`

Aby obliczyć narosłe odsetki, należy utworzyć następującą miarę:

```
=start_balance*(rate*(today(1)-yearstart(today(1)))/365)
```

Ustaw **Formatowanie liczb** miary na **Waluta**.

Tabela wynikowa

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

Używając dzisiejszej daty jako jedynego argumentu, funkcja `yearstart()` zwraca datę początkową bieżącego roku. Odejmując ten wynik od bieżącej daty, wyrażenie zwraca liczbę dni, które upłynęły do tej pory w tym roku.

Wartość ta jest następnie mnożona przez stopę procentową i dzielona przez 365, aby uzyskać efektywną stopę procentową za dany okres. Efektywna stopa procentowa za dany okres jest następnie mnożona przez saldo początkowe pożyczki, aby zwrócić odsetki naliczone do tej pory w tym roku.

yeartodate

Ta funkcja sprawdza, czy wejściowy znacznik czasu przypada w roku daty dnia, w którym skrypt został ostatnio załadowany, i zwraca True, jeśli przypada, a False, jeśli nie przypada.

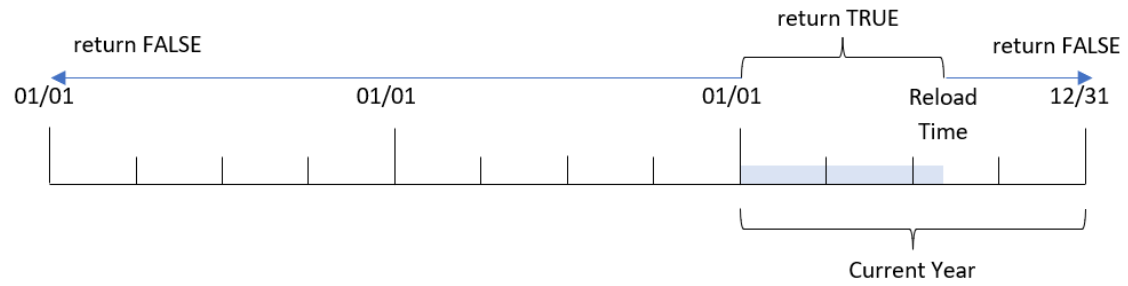
Składnia:

```
YearToDate(timestamp[ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Typ zwracanych danych: Wartość logiczna

> *W Qlik Sense wartość logiczna Prawda jest reprezentowana przez -1, a wartość Fałsz jest reprezentowana przez 0.*

Przykładowy diagram funkcji `yeartodate()`



Jeśli żaden z parametrów opcjonalnych nie jest używany, wówczas data od początku roku oznacza dowolną datę w ciągu roku kalendarzowego od 1 stycznia aż do daty ostatniego wykonania skryptu włącznie.

Innymi słowy, funkcja `yeartodate()`, jeśli zostanie wywołana bez żadnych dodatkowych parametrów, pobiera znacznik czasu i zwraca wartość logiczną zależną od tego, czy dana data wypadła w roku kalendarzowym do daty przeładowania włącznie.

Istnieje jednak także możliwość zmiany daty początkowej roku za pomocą argumentu `firstmonth`, jak również dokonywania porównań z poprzednimi lub następnymi latami za pomocą argumentu `yearoffset`.

Ponadto, w przypadku historycznych zestawów danych, funkcja `yeartodate()` udostępnia parametr umożliwiający ustawienie argumentu `todaydate`, co spowoduje porównanie znacznika czasu z rokiem kalendarzowym do daty przekazanej w argumencie `todaydate` włącznie.

Argumenty

Argument	Opis
timestamp	Znacznik czasu do porównania, np. '10/12/2012'.
yearoffset	Po określeniu wartości **yearoffset** parametr **yeartodate** zwraca wartość True dla tego samego okresu w innym roku. Ujemna wartość **yearoffset** wskazuje poprzedni rok, a dodatnie przesunięcie – przyszły rok. Najnowszą datę od początku roku uzyskuje się przez określenie yearoffset = -1. W przypadku pominięcia przyjmuje się wartość 0.
firstmonth	Przez określenie wartości **firstmonth** z zakresu od 1 do 12 (1 w przypadku pominięcia) można przesunąć początek roku do pierwszego dnia dowolnego miesiąca. Na przykład w celu ustalenia roku obrotowego z początkiem 1 maja należy określić **firstmonth** = 5. Wartość 1 oznaczałaby rok podatkowy zaczynający się 1 stycznia, a wartość 12 – rok podatkowy zaczynający się 1 grudnia.
todaydate	Określenie parametru **todaydate** (w przypadku pominięcia jest to znacznik czasu ostatniego wykonania skryptu) umożliwia przeniesienie dnia używanego jako górna granica okresu.

Kiedy używać

Funkcja `yeartodate()` zwraca wynik logiczny. Zazwyczaj ten typ funkcji będzie używany jako warunek w wyrażeniu if. Spowoduje to zwrócenie agregacji lub obliczeń w zależności od tego, czy przetwarzana data wypada w roku do daty ostatniego przeładowania aplikacji włącznie.

Na przykład funkcja YearToDate() może zostać użyta do identyfikacji całego sprzętu wyprodukowanego do danego momentu w roku.

W poniższych przykładach przyjmuje się, że ostatnie przeładowanie miało miejsce 18.11.2011 r.

Przykłady funkcji

Przykład	Wynik
`yeartodate( '11/18/2010')`	zwraca wartość False
`yeartodate( '02/01/2011')`	zwraca wartość True
`yeartodate( '11/18/2011')`	zwraca wartość True
`yeartodate( '11/19/2011')`	zwraca wartość False
`yeartodate( '11/19/2011', 0, 1, '12/31/2011')`	zwraca wartość True
`yeartodate( '11/18/2010', -1)`	zwraca wartość True
`yeartodate( '11/18/2011', -1)`	zwraca wartość False
`yeartodate( '04/30/2011', 0, 5)`	zwraca wartość False
`yeartodate( '05/01/2011', 0, 5)`	zwraca wartość True

Ustawienia regionalne

Jeżeli nie podano inaczej, w przykładach w tym temacie stosowany jest następujący format daty: MM/DD/RRRR. Format daty jest określony w instrukcji `SET DateFormat` w skrypcie ładowania danych. Domyślny format daty w Twoim systemie może być inny ze względu na ustawienia regionalne i inne czynniki. Formaty zastosowane w przykładach można zmienić, aby dostosować je do własnych wymagań. Zamiast tego można też zmienić formaty w skrypcie ładowania, aby pasowały do tych przykładów.

Domyślne ustawienia regionalne w aplikacjach są oparte na regionalnych ustawieniach systemu komputera lub serwera, na którym zainstalowano Qlik Sense. Jeśli serwer Qlik Sense, do którego uzyskujesz dostęp, jest ustawiony na Szwecję, Edytor ładowania danych użyje szwedzkich ustawień regionalnych dla dat, godziny i waluty. Te ustawienia regionalne nie są związane z językiem wyświetlanym w interfejsie użytkownika Qlik Sense. Interfejs Qlik Sense będzie wyświetlany w tym samym języku co używana przeglądarka.

Przykład 1 – Przykład podstawowy

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za lata 2020-2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).
- Utworzenie pola, `year_to_date`, określającego, które transakcje miały miejsce w roku kalendarzowym do daty ostatniego przeładowania.

Aktualnie ta data to 26 kwietnia 2022 r.

Skrypt ładowania

```
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
];
```

Wyniki

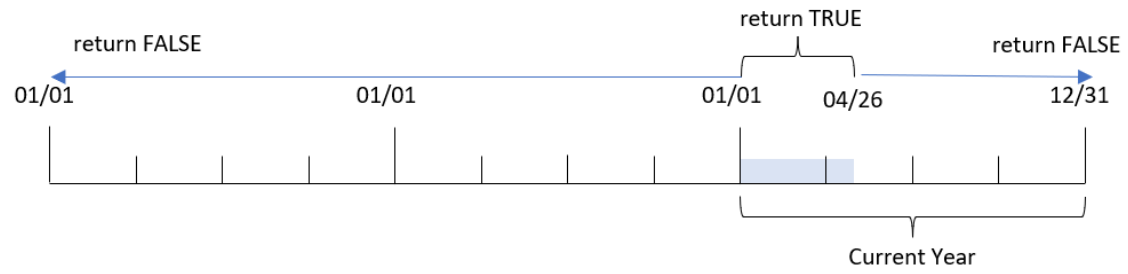
Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `year_to_date`

Tabela wynikowa

date	year_to_date
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

Diagram funkcji `yeartodate()`, przykład podstawowy



Pole `year_to_date` jest tworzone w poprzedzającej instrukcji LOAD przez użycie funkcji `yeartodate()` i przekazanie pola `date` jako argumentu funkcji.

Ponieważ do funkcji nie zostały przekazane żadne dalsze parametry, funkcja `yeartodate()` najpierw identyfikuje datę przeładowania, a więc granice bieżącego roku kalendarzowego (zaczynającego się 1 stycznia), które zwrócą logiczną wartość `TRUE`.

W związku z tym każda transakcja, która ma miejsce między 1 stycznia a 26 kwietnia, zwróci wynik logiczny `TRUE`. Każda transakcja mająca miejsce przed początkiem roku 2022 zwróci logiczną wartość `FALSE`.

Przykład 2 – yearoffset

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola `two_years_prior` określającego, które transakcje miały miejsce dwa pełne lata przed datą kalendarzową.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

- `date`
- `two_years_prior`

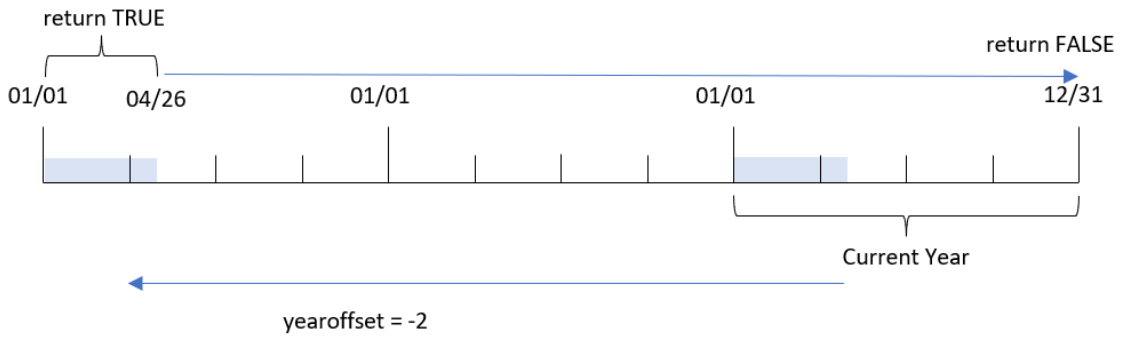
Tabela wynikowa

date	two_years_prior
01/10/2020	-1
02/28/2020	-1
04/09/2020	-1
04/16/2020	-1
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0

date	two_years_prior
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

Przekazanie wartości -2 jako argument `yearoffset` funkcji `yeartodate()` powoduje przesunięcie granic porównawczego segmentu roku kalendarzowego o dwa pełne lata. Początkowo segment roku obejmuje dni między 1 stycznia a 26 kwietnia 2022 roku. Argument `yearoffset` przesuwa ten segment o dwa lata wstecz. W efekcie zakres dat zmienia się na przedział od 1 stycznia do 26 kwietnia 2020 roku.

Diagram funkcji `yeartodate()`, przykład z użyciem argumentu yearoffset



W związku z tym każda transakcja, która ma miejsce między 1 stycznia a 26 kwietnia, zwróci wynik logiczny TRUE. Wszystkie transakcje mające miejsce przed tym segmentem lub po nim, spowodują zwrócenie wartości FALSE.

Przykład 3 – firstmonth

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola, `year_to_date`, określającego, które transakcje miały miejsce w roku kalendarzowym do daty ostatniego przeładowania.

W tym przykładzie ustawiamy początek roku podatkowego na 1 lipca.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

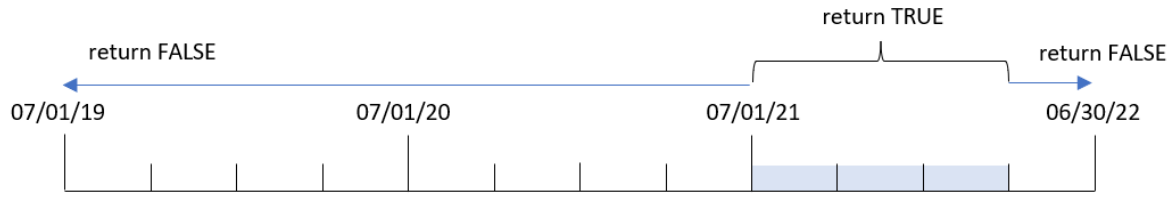
- `date`
- `year_to_date`

Tabela wynikowa

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

W tym przypadku, ponieważ jako argument `firstmonth` funkcji `yeartodate()` przekazano wartość 7, funkcja ta ustawia pierwszy dzień roku na 1 lipca, a ostatni dzień roku na 30 czerwca.

Diagram funkcji `yeartodate()`, przykład z argumentem firstmonth



W związku z tym każda transakcja, która ma miejsce między 1 lipca 2021 roku a 26 kwietnia 2022 roku, spowoduje zwrócenie wartości logicznej TRUE. Każda transakcja mająca miejsce przed 1 lipca spowoduje zwrócenie wartości logicznej FALSE.

Przykład 4 – todaydate

Skrypt ładowania i wyniki

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Ten sam zestaw danych i scenariusz co w pierwszym przykładzie.
- Utworzenie pola, year_to_date, określającego, które transakcje miały miejsce w roku kalendarzowym do daty ostatniego przeładowania.

Jednak w tym przykładzie musimy zidentyfikować wszystkie transakcje, które miały miejsce w roku kalendarzowym do 1 marca 2022 r. włącznie.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj te pola jako wymiary:

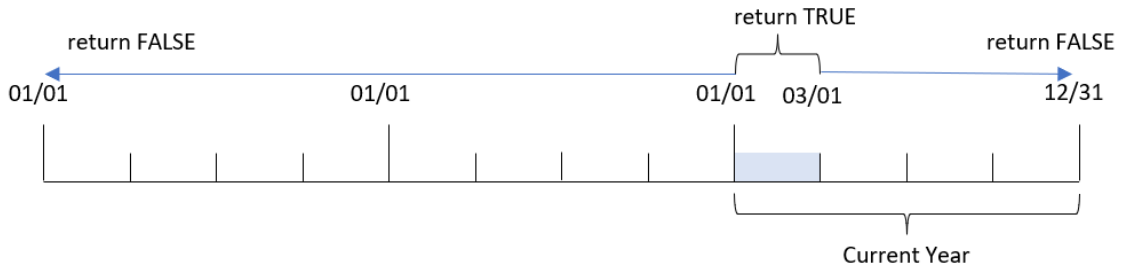
- `date`
- `year_to_date`

Tabela wynikowa

date	year_to_date
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	-1
02/26/2022	-1
03/07/2022	0
03/11/2022	0

W tym przypadku funkcji `yeartodate()` przekazano datę 01.03.2022 jako argument `todaydate`, w związku z czym funkcja ta ustawia granicę końcową porównawczego segmentu roku kalendarzowego na 1 marca 2022 r. Przekazanie parametru `firstmonth` (o wartości z przedziału od 1 do 12) ma krytyczne znaczenie. Jeśli tego nie zrobimy, funkcja będzie zwracać puste wyniki.

Diagram funkcji `yeartodate()`, przykład z użyciem argumentu todaydate



W związku z tym każda transakcja zawarta między 1 stycznia a 1 marca 2022 roku, parametr `todaydate`, spowoduje zwrócenie logicznej wartości `TRUE`. Każda transakcja mająca miejsce przed 1 stycznia i po 1 marca 2022 roku spowoduje zwrócenie wartości logicznej `FALSE`.

Przykład 5 – Przykład z użyciem obiektu wykresu

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera ten sam zestaw danych i scenariusz co w pierwszym przykładzie.

Jednak w tym przykładzie do aplikacji został załadowany niezmieniony zbiór danych. Obliczenia określające, które transakcje miały miejsce w roku kalendarzowym do daty ostatniego przeładowania są tworzone jako miara w obiekcie wykresu aplikacji.

Skrypt ładowania

```
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
];
```

Wyniki

Załaduj dane i otwórz arkusz. Utwórz nową tabelę i dodaj to pole jako wymiar: `date`.

Dodaj następującą miarę:

```
=yeartodate(date)
```

Tabela wynikowa

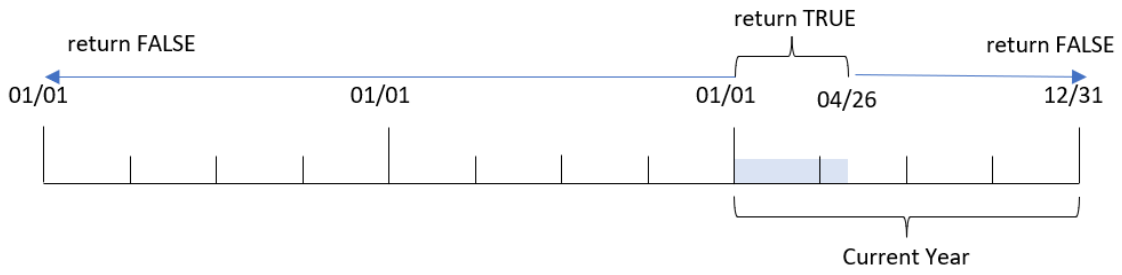
date	=yeartodate(date)
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	-1

date	=yeartodate(date)
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

Miara `year_to_date` została utworzona w obiekcie wykresu przy użyciu funkcji `yeartodate()`, której przekazano pole daty `date` jako argument.

Ponieważ do funkcji nie zostały przekazane żadne dalsze parametry, funkcja `yeartodate()` najpierw identyfikuje datę przeładowania, a więc granice bieżącego roku kalendarzowego (zaczynającego się 1 stycznia), które zwrócą logiczną wartość `TRUE`.

Diagram funkcji `yeartodate()`, przykład z użyciem obiektu wykresu



Każda transakcja, która ma miejsce między 1 stycznia a 26 kwietnia, zwróci wynik logiczny `TRUE`. Każda transakcja mająca miejsce przed początkiem roku 2022 zwróci logiczną wartość `FALSE`.

Przykład 6 – Scenariusz

Skrypt ładowania i wyrażenie wykresu

Przegląd

Otwórz Edytor ładowania danych i poniżej dodaj skrypt ładowania do nowej karty.

Skrypt ładowania zawiera:

- Zestaw danych zawierający zestaw transakcji za lata 2020-2022, który jest ładowany do tabeli o nazwie `Transactions`.
- Pole danych w formacie zmiennej systemowej `DateFormat` (MM/DD/YYYY).

Użytkownik potrzebuje obiektu wskaźnika KPI przedstawiającego łączną sprzedaż za okres 2021 roku analogiczny do okresu od początku bieżącego roku do daty ostatniego przeładowania.

Obecnie ta data to 16 czerwca 2022 r.

Skrypt ładowania

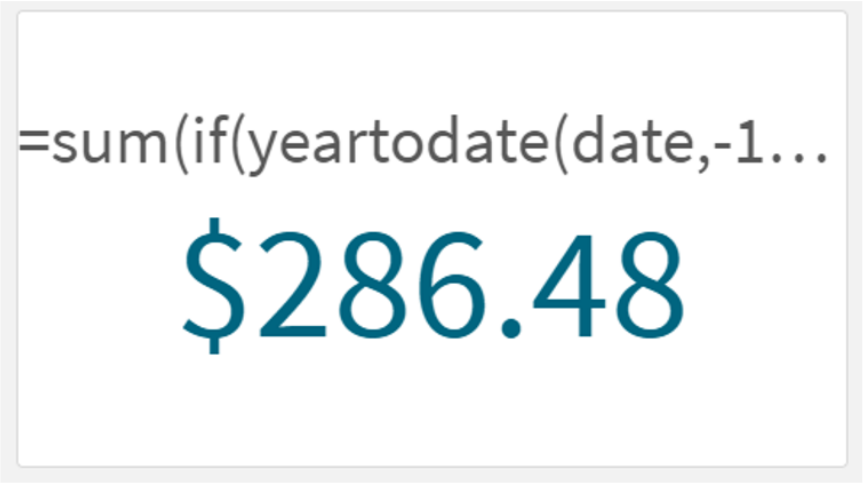
```
SET DateFormat='MM/DD/YYYY';

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
];
```

Wyniki

Wykonaj następujące czynności:

1. Utwórz obiekt wskaźnik KPI.
2. Utwórz następującą miarę agregacji, aby obliczyć łączną sprzedaż
   `=sum(if(yeartodate(date,-1),amount,0))`
3. Ustaw **Formatowanie liczb** miary na **Waluta**.

Wykres KPI funkcji `yeartodate()` za 2021 rok



Funkcja `yeartodate()` zwraca wartość logiczną na podstawie ocen dat transakcji określonych za pomocą identyfikatorów. Ponieważ przeładowanie miało miejsce 16 czerwca 2022 roku, funkcja `yeartodate` wydziela segment roku obejmujący okres od 01.01.2022 do 16.06.2022. Ponieważ jednak funkcji przekazano wartość -1 jako argument `period_no`, te granice zostały przesunięte do poprzedniego roku. W związku z tym, dla każdej transakcji mającej miejsce między 01.01.2021 a 16.06.2021 funkcja `yeartodate()` zwraca wartość logiczną `TRUE` i sumuje kwotę.

5.8 Funkcje wykładnicze i logarytmiczne

W tej sekcji opisano funkcje związane z obliczeniami wykładniczymi i logarytmicznymi. Wszystkie funkcje mogą być stosowane zarówno w skryptach ładowania danych, jak i wyrażeniach wykresu.

W funkcjach poniżej parametry są wyrażeniami, w których wartości **x** i **y** powinny być interpretowane jako liczby o wartości rzeczywistej.

exp

Naturalna funkcja wykładnicza, e^x, przy użyciu logarytmu naturalnego **e** jako podstawy. Wynik jest liczbą dodatnią.

```
exp(x )
```

Przykłady i wyniki:

`exp(3)` zwraca 20,085

log

Logarytm naturalny **x**. Funkcja jest określona tylko wtedy, gdy **x**> 0. Wynik jest liczbą.

```
log(x )
```

Przykłady i wyniki:

`log(3)` zwraca 1,0986

log10

Logarytm wspólny (podstawa 10) **x**. Funkcja jest określona tylko wtedy, gdy **x**> 0. Wynik jest liczbą.

```
log10(x )
```

Przykłady i wyniki:

`log10(3)` zwraca 0,4771

pow

Zwraca **x** do potęgi **y**. Wynik jest liczbą.

```
pow(x,y )
```

Przykłady i wyniki:

`pow(3, 3)` zwraca 27

sqr

x kwadrat (**x** do potęgi 2). Wynik jest liczbą.

```
sqr (x )
```

Przykłady i wyniki:

`sqr(3)` zwraca 9

sqrt

Pierwiastek kwadratowy z **x**. Funkcja jest określona tylko wtedy, gdy **x** >= 0. Wynik jest liczbą dodatnią.

```
sqrt(x )
```

Przykłady i wyniki:

`sqrt(3)` zwraca 1,732

5.9 Funkcje pól

Te funkcje mogą być używane tylko w wyrażeniach wykresu.

Funkcje pól zwracają liczby całkowite albo ciągi określające różne aspekty dotyczące selekcji pól.

Funkcje licznikowe

GetAlternativeCount

Funkcja **GetAlternativeCount()** służy do określenia liczby wartości alternatywnych (jasnoszarych) w podanym polu.

```
GetAlternativeCount – funkcja wykresu (field_name)
```

GetExcludedCount

GetExcludedCount() zwraca liczbę wykluczonych wartości odrębnych w podanym polu. Wartości wykluczone zawierają pola alternatywne (jasnoszare), wykluczone (ciemnoszare) oraz wybrane wykluczone (ciemnoszare ze znacznikiem wyboru).

```
GetExcludedCount – funkcja wykresu (page 1172)(field_name)
```

GetNotSelectedCount

Ta funkcja wykresu zwraca liczbę niewybranych wartości w polu o nazwie **fieldname**. Ta funkcja obowiązuje, pod warunkiem że dla pola ustawiono tryb And.

```
GetNotSelectedCount – funkcja wykresu(fieldname [, includeexcluded=false])
```

GetPossibleCount

Funkcja **GetPossibleCount()** służy do określenia liczby możliwych wartości w podanym polu. Jeśli podane pole zawiera selekcje, liczone są pola wybrane (zielone). W przeciwnym razie liczone są wartości powiązane (białe).

```
GetPossibleCount – funkcja wykresu(field_name)
```

GetSelectedCount

Funkcja **GetSelectedCount()** zwraca liczbę wybranych (zielonych) wartości w polu.

```
GetSelectedCount – funkcja wykresu (field_name [, include_excluded])
```

Funkcje pól i selekcji

GetCurrentSelections

Funkcja **GetCurrentSelections()** zwraca listę bieżących wyborów w aplikacji. Jeśli zamiast tego wybory są wykonywane z użyciem szukanego ciągu w polu wyszukiwania, wówczas funkcja **GetCurrentSelections()** zwraca szukany ciąg.

```
GetCurrentSelections – funkcja wykresu([record_sep [,tag_sep [,value_sep
[,max_values]]]])
```

GetFieldSelections

Funkcja **GetFieldSelections()** zwraca ciąg znaków (**string**) zawierający bieżące selekcje w danym polu.

```
GetFieldSelections – funkcja wykresu ( field_name [, value_sep [, max_
values]])
```

GetObjectDimension
Funkcja **GetObjectDimension()** zwraca nazwę wymiaru. **Index** jest opcjonalną liczbą całkowitą wskazującą, który z wymiarów powinien być zwracany.

```
GetObjectDimension – funkcja wykresu ([index])
```

GetObjectField
Funkcja **GetObjectField()** zwraca nazwę wymiaru. **Index** jest opcjonalną liczbą całkowitą wskazującą, który z wymiarów powinien być zwracany.

```
GetObjectField – funkcja wykresu ([index])
```

GetObjectMeasure
Funkcja **GetObjectMeasure()** zwraca nazwę miary. **Index** jest opcjonalną liczbą całkowitą wskazującą, która z miar powinna być zwracana.

```
GetObjectMeasure – funkcja wykresu ([index])
```

GetAlternativeCount – funkcja wykresu

Funkcja **GetAlternativeCount()** służy do określenia liczby wartości alternatywnych (jasnoszarych) w podanym polu.

Składnia:

```
GetAlternativeCount (field_name)
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
field_name	Pole zawierające mierzony zakres danych.

Przykłady i wyniki:

W poniższym przykładzie użyto pola **First name** załadowanego do panelu filtrowania.

Przykłady i wyniki

Przykłady	Wyniki
Jeśli na liście **First name** wybrano wartość **John**. `GetAlternativeCount ([First name])`	4, ponieważ na liście **First name** występują cztery niepowtarzalne i wykluczone (wyszarzone) wartości.

Przykłady	Wyniki
Jeśli wybrano wartości **John** i **Peter**. `GetAlternativeCount ([First name])`	3, ponieważ na liście **First name** występują trzy niepowtarzalne i wykluczone (wyszarzone) wartości.
Jeśli na liście **First name** nie wybrano żadnych wartości. `GetAlternativeCount ([First name])`	0, ponieważ nie dokonano żadnej selekcji.

Dane zastosowane w przykładzie:

```
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 – funkcja wykresu

Funkcja **GetCurrentSelections()** zwraca listę bieżących wyborów w aplikacji. Jeśli zamiast tego wybory są wykonywane z użyciem szukanego ciągu w polu wyszukiwania, wówczas funkcja **GetCurrentSelections()** zwraca szukany ciąg.

W razie stosowania opcji należy określić separator record_sep. Aby określić nowy wiersz, należy nadać separatorowi **record_sep** wartość **chr(13)&chr(10)**.

Jeśli wybrano wszystkie wartości oprócz jednej albo dwóch, zastosowany zostanie odpowiednio format NOT x,y albo NOT y. Jeśli wybrano wszystkie wartości i liczba tych wszystkich wartości będzie większa niż max_values, zwrócona zostanie wartość ALL.

Składnia:

```
GetCurrentSelections ([record_sep [, tag_sep [, value_sep [, max_values [, state_name]]]]])
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argumenty	Opis
record_sep	Separator umieszczany między rekordami w polu. Domyślnym separatorem jest wartość <CR><LF> oznaczająca nowy wiersz.
tag_sep	Separator umieszczany między znacznikiem nazwy pola i wartościami pola. Domyślnym separatorem jest „: ”.
value_sep	Separator do umieszczenia między wartościami pola. Separator do umieszczenia między wartościami pola. Wartością domyślną jest ', '.
max_values	Maksymalna liczba wartości pola, które mogą zostać osobno podane. Jeśli zostanie wybrana większa liczba wartości, zostanie zastosowany format „x z y wartości”. Wartością domyślną jest 6.
state_name	Nazwa stanu alternatywnego, który został wybrany dla danej wizualizacji. Jeśli używany jest argument **state_name**, uwzględniane są tylko wybory powiązane z określoną nazwą stanu.

Przykłady i wyniki:

W poniższym przykładzie wykorzystano dwa pola załadowane do różnych paneli filtrowania: jedno pole to **First name**, a drugie pole to **Initials**.

Przykłady i wyniki

Przykłady	Wyniki
Jeśli na liście **First name** wybrano wartość **John**. `GetCurrentSelections ()`	'First name: John'
Jeśli na liście **First name** wybrano wartości **John** i **Peter**. `GetCurrentSelections ()`	'First name: John, Peter'
Jeśli na liście **First name** wybrano wartości **John** oraz **Peter**, a na liście **Initials** wybrano wartość **JA**. `GetCurrentSelections ()`	'First name: John, Peter Initials: JA'
Jeśli na liście **First name** wybrano wartość **John** i na liście **Initials** wybrano wartość **JA**. `GetCurrentSelections ( chr(13)&chr(10) , ' = ' )`	'First name = John Initials = JA'
Jeśli na liście **First name** wybrano wszystkie imiona oprócz Sue i nie wybrano nic na liście **Initials**. `GetCurrentSelections (chr(13)&chr(10), '=', ',' ,3)`	'First name=NOT Sue'

Dane zastosowane w przykładzie:

```
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 – funkcja wykresu

GetExcludedCount() zwraca liczbę wykluczonych wartości odrębnych w podanym polu. Wartości wykluczone zawierają pola alternatywne (jasnoszare), wykluczone (ciemnoszare) oraz wybrane wykluczone (ciemnoszare ze znacznikiem wyboru).

Składnia:

```
GetExcludedCount (field_name)
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argumenty	Opisu
field_name	Pole zawierające mierzony zakres danych.

Przykłady i wyniki:

W poniższym przykładzie wykorzystano trzy pola wczytane do różnych paneli filtrowania: jedno pole to **First name**, drugie pole to **Last name**, a trzecie pole to **Initials**.

Przykłady i wyniki

Przykłady	Wyniki
Jeśli na liście **First name** nie wybrano żadnych wartości.	`GetExcludedCount (Initials) = 0` Nie ma żadnych wyborów.
Jeśli wybrano **John** w na liście **First name**.	`GetExcludedCount (Initials) = 5` Na liście **Initials** jest 5 wykluczonych wartości w kolorze ciemnoszarym. Szósta komórka (JA) będzie biała, ponieważ jest powiązana z wybraną wartością John na liście **First name**.
Jeśli wybrano wartości **John** oraz **Peter**.	`GetExcludedCount (Initials) = 3` Wartość John jest powiązana z 1 wartością, a wartość Peter jest powiązana z 2 wartościami na liście **Initials**.

Przykłady	Wyniki
Jeśli wartości **John** i **Peter** są wybrane na liście **First name**, a następnie wartość **Franc** zostanie wybrana na liście **Last name**.	`GetExcludedCount ([First name]) = 4` W polu **First name** są 4 wykluczone wartości koloru ciemnoszarego. Funkcja **GetExcludedCount()** ocenia pola z wykluczonymi wartościami, uwzględniając pola alternatywne i wybrane wykluczone.
Jeśli wartości **John** i **Peter** wybrano na liście **First name**, a następnie wartości **Franc** i **Anderson** wybrano na liście **Last name**.	`GetExcludedCount (Initials) = 4` Na liście **Initials** są 4 wykluczone wartości w kolorze ciemnoszarym. Pozostałe dwie komórki (JA i PF) będą białe, ponieważ są powiązane z wyborami John i Peter na liście **First name**.
Jeśli wartości **John** i **Peter** wybrano na liście **First name**, a następnie wartości **Franc** i **Anderson** wybrano na liście **Last name**.	`GetExcludedCount ([Last name]) = 4` Na liście **Initials** są 4 wykluczone wartości. Wartość Devonshire ma kolor jasnoszary, a wartości Brown, Carr oraz Elliot są ciemnoszare.

Dane zastosowane w przykładzie:

```
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 – funkcja wykresu

Funkcja **GetFieldSelections()** zwraca ciąg znaków (**string**) zawierający bieżące selekcje w danym polu.

Jeśli wybrano wszystkie wartości oprócz jednej albo dwóch, zastosowany zostanie odpowiednio format NOT x,y albo NOT y. Jeśli wybrano wszystkie wartości i liczba tych wszystkich wartości będzie większa niż max_values, zwrócona zostanie wartość ALL.

Składnia:

```
GetFieldSelections ( field_name [, value_sep [, max_values [, state_name]]])
```

Typ zwracanych danych: ciąg znaków

Formaty zwracanych ciągów

Format	Opis
'a, b, c'	Jeśli liczba wybranych wartości wynosi max_values lub jest mniejsza, zwracany ciąg jest listą wybranych wartości. Wartości rozdziela value_sep jako ogranicznik.

Format	Opis
'NOT a, b, c'	Jeśli liczba niewybranych wartości wynosi max_values lub jest mniejsza, zwracany ciąg jest listą niewybranych wartości z prefiksem NOT. Wartości rozdziela value_sep jako ogranicznik.
'x of y'	x = liczba wybranych wartości y = łączna liczba wartości Jest to zwracane, gdy max_values < x < (y - max_values).
'ALL'	Zwracane, gdy wybrane są wszystkie wartości.
'-'	Zwracane, gdy nie ma żadnych wybranych wartości.
<search string>	Jeżeli wybrano, używając wyszukiwania, zostanie zwrócony ciąg wyszukiwania.

Argumenty:

Argumenty

Argumenty	Opisu
field_name	Pole zawierające mierzony zakres danych.
value_sep	Separator do umieszczenia między wartościami pola. Separator do umieszczenia między wartościami pola. Wartością domyślną jest ', '.
max_values	Maksymalna liczba wartości pola, które mogą zostać osobno podane. Jeśli zostanie wybrana większa liczba wartości, zostanie zastosowany format „x z y wartości”. Wartością domyślną jest 6.
state_name	Nazwa stanu alternatywnego, który został wybrany dla danej wizualizacji. Jeśli używany jest argument **state_name**, uwzględniane są tylko wybory powiązane z określoną nazwą stanu.

Przykłady i wyniki:

W poniższym przykładzie użyto pola **First name** załadowanego do panelu filtrowania.

Przykłady i wyniki

Przykłady	Wyniki
Jeśli na liście **First name** wybrano wartość **John**. `GetFieldSelections ([First name])`	'John'

Przykłady	Wyniki
Jeśli wybrano wartości **John** i **Peter**. `GetFieldSelections ([First name])`	'John,Peter'
Jeśli wybrano wartości **John** i **Peter**. `GetFieldSelections ([First name],'; ')`	'John; Peter'
Jeśli na liście **First name**. wybrano wartości **John**, **Sue** i **Mark**. `GetFieldSelections ([First name],';',2)`	NOT Jane;Peter, ponieważ wartość 2 jest wartością argumentu max_ values. W przeciwnym razie zwrócona zostałaby wartość John; Sue; Mark.

Dane zastosowane w przykładzie:

```
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 – funkcja wykresu

Ta funkcja wykresu zwraca liczbę niewybranych wartości w polu o nazwie **fieldname**. Ta funkcja obowiązuje, pod warunkiem że dla pola ustawiono tryb And.

Składnia:

```
GetNotSelectedCount(fieldname [, includeexcluded=false])
```

Argumenty:

Argumenty

Argument	Opis
fieldname	Nazwa pola przeznaczonego do oceny.
includeexcluded	Jeśli parametr **includeexcluded** jest określony jako True, wówczas zliczanie obejmie wartości wybrane, które są wykluczone przez selekcje w innym polu.

Przykłady:

```
GetNotSelectedCount( Country )
```

```
GetNotSelectedCount( Country, true )
```

GetObjectDimension – funkcja wykresu

Funkcja **GetObjectDimension()** zwraca nazwę wymiaru. **Index** jest opcjonalną liczbą całkowitą wskazującą, który z wymiarów powinien być zwracany.

Tej funkcji można używać w wykresie w następujących miejscach: tytuł, podtytuł, stopka, wyrażenie linii odniesienia.

Za pomocą Object ID nie możesz utworzyć w innym obiekcie odniesienia do nazwy wymiaru lub miary.

Składnia:

```
GetObjectDimension ([index])
```

Przykład:

```
GetObjectDimension(1)
```

Przykład: wyrażenie wykresu

Tabela aplikacji Qlik Sense z przykładami funkcji GetObjectDimension w wyrażeniu wykresu.

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

W celu zawrócenia nazwy miary zamiast tej funkcji należy użyć funkcji **GetObjectMeasure**.

GetObjectField – funkcja wykresu

Funkcja **GetObjectField()** zwraca nazwę wymiaru. **Index** jest opcjonalną liczbą całkowitą wskazującą, który z wymiarów powinien być zwracany.

Tej funkcji można używać w wykresie w następujących miejscach: tytuł, podtytuł, stopka, wyrażenie linii odniesienia.

Za pomocą Object ID nie możesz utworzyć w innym obiekcie odniesienia do nazwy wymiaru lub miary.

Składnia:

```
GetObjectField ([index])
```

Przykład:

```
GetObjectField(1)
```

Przykład: wyrażenie wykresu

Tabela aplikacji Qlik Sense z przykładami funkcji GetObjectField w wyrażeniu wykresu.

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

W celu zawrócenia nazwy miary zamiast tej funkcji należy użyć funkcji **GetObjectMeasure**.

GetObjectMeasure – funkcja wykresu

Funkcja **GetObjectMeasure()** zwraca nazwę miary. **Index** jest opcjonalną liczbą całkowitą wskazującą, która z miar powinna być zwracana.

Tej funkcji można używać w wykresie w następujących miejscach: tytuł, podtytuł, stopka, wyrażenie linii odniesienia.

Za pomocą Object ID nie możesz utworzyć w innym obiekcie odniesienia do nazwy wymiaru lub miary.

Składnia:

```
GetObjectMeasure ([index])
```

Przykład:

```
GetObjectMeasure(1)
```

Przykład: wyrażenie wykresu

Tabela aplikacji Qlik Sense z przykładami funkcji GetObjectMeasure w wyrażeniu wykresu.

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)

W celu zawrócenia nazwy wymiaru zamiast tej funkcji należy użyć funkcji **GetObjectField**.

GetPossibleCount – funkcja wykresu

Funkcja **GetPossibleCount()** służy do określenia liczby możliwych wartości w podanym polu. Jeśli podane pole zawiera selekcje, liczone są pola wybrane (zielone). W przeciwnym razie liczone są wartości powiązane (białe). .

W przypadku pól wybranych funkcja **GetPossibleCount()** zwraca liczbę pól wybranych (zielonych).

Typ zwracanych danych: integer

Składnia:

```
GetPossibleCount (field_name)
```

Argumenty:

Argumenty

Argumenty	Opisu
field_name	Pole zawierające mierzony zakres danych.

Przykłady i wyniki:

W poniższym przykładzie wykorzystano dwa pola załadowane do różnych paneli filtrowania: jedno pole to **First name**, a drugie pole to **Initials**.

Przykłady i wyniki

Przykłady	Wyniki
Jeśli na liście **First name** wybrano wartość **John**. `GetPossibleCount ([Initials])`	1, ponieważ na liście Initials jest jedna wartość powiązana z wyborem wartości (**John**) na liście **First name**.
Jeśli na liście **First name** wybrano wartość **John**. `GetPossibleCount ([First name])`	1, ponieważ wybrano jedną wartość (**John**) na liście **First name**.

Przykłady	Wyniki
Jeśli na liście **First name** wybrano wartość **Peter**. `GetPossibleCount ([Initials])`	2, ponieważ wartość Peter jest powiązana z dwoma wartościami na liście **Initials**.
Jeśli na liście **First name** nie wybrano żadnych wartości. `GetPossibleCount ([First name])`	5, ponieważ nie ma selekcji, a na liście **First name** jest pięć niepowtarzalnych wartości.
Jeśli na liście **First name** nie wybrano żadnych wartości. `GetPossibleCount ([Initials])`	6, ponieważ nie ma selekcji, a na liście **Initials** jest sześć niepowtarzalnych wartości.

Dane zastosowane w przykładzie:

```
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 – funkcja wykresu

Funkcja **GetSelectedCount()** zwraca liczbę wybranych (zielonych) wartości w polu.

Składnia:

```
GetSelectedCount (field_name [, include_excluded [, state_name]])
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argumenty	Opis
field_name	Pole zawierające mierzony zakres danych.
include_ excluded	W przypadku ustawienia wartości **True()**, wówczas zliczanie obejmie wartości wybrane, które są aktualnie wykluczone przez selekcje w innych polach. Jeśli parametr ten ma wartość False lub został pominięty, wartości te nie zostaną uwzględnione.
state_name	Nazwa stanu alternatywnego, który został wybrany dla danej wizualizacji. Jeśli używany jest argument **state_name**, uwzględniane są tylko wybory powiązane z określoną nazwą stanu.

Przykłady i wyniki:

W poniższym przykładzie wykorzystano trzy pola wczytane do różnych paneli filtrowania: jedno pole to **First name**, drugie pole to **Initials**, a trzecie pole to **Has cellphone**.

Przykłady i wyniki

Przykłady	Wyniki
Jeśli na liście **First name** wybrano wartość **John**. `GetSelectedCount ([First name])`	1, ponieważ na liście **First name** wybrano jedną wartość.
Jeśli na liście **First name** wybrano wartość **John**. `GetSelectedCount ([Initials])`	0, ponieważ na liście **Initials** nie wybrano żadnej wartości.
Nie wybieraj żadnych wartości na liście **First name**, ale wybierz wszystkie wartości na liście **Initials**, a następnie zaznacz wartość **Yes** na liście **Has cellphone**. `GetSelectedCount ([Initials], True())`	6. Wybrane na liście wartości **Initials**MC i PD mają na liście **Has cellphone** przypisaną wartość **No**, ale zwracana jest nadal liczba 6, ponieważ argument include_excluded ma wartość True().

Dane zastosowane w przykładzie:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

5.10 Funkcje pliku

Funkcje plików (dostępne tylko w wyrażeniach skryptu) zwracają informacje o aktualnie odczytywanym pliku tabeli. Funkcje te zwrócą wartość NULL w odniesieniu do wszystkich źródeł danych poza plikami tabeli (wyjątek: **ConnectString( )**).

Przegląd funkcji plików

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

Attribute

Ta funkcja skryptu zwraca wartość metaznaczników różnych plików multimedialnych w postaci tekstu. Obsługiwane są następujące formaty plików: MP3, WMA, WMV, PNG oraz JPG. Jeśli parametr **filename** pliku nie istnieje, nie jest w obsługiwanym formacie pliku albo nie zawiera metaznacznika o nazwie **attributename**, zwrócone zostanie NULL.

```
Attribute (filename, attributename)
```

ConnectString

Funkcja **ConnectString()** zwraca nazwę aktywnego połączenia do danych dla połączeń ODBC lub OLE DB. Funkcja zwraca pusty ciąg znaków, jeśli nie wykonano instrukcji **connect** lub po instrukcji **disconnect**.

```
ConnectString ()
```

FileBaseName

Funkcja **FileBaseName** zwraca ciąg znaków zawierający nazwę aktualnie odczytywanego pliku tabeli, bez ścieżki ani rozszerzenia.

```
FileBaseName ()
```

FileDir

Funkcja **FileDir** zwraca ciąg znaków zawierający ścieżkę do katalogu aktualnie odczytywanego pliku tabeli.

```
FileDir ()
```

FileExtension

Funkcja **FileExtension** zwraca ciąg znaków zawierający rozszerzenie aktualnie odczytywanego pliku tabeli.

```
FileExtension ()
```

FileName

Funkcja **FileName** zwraca ciąg znaków zawierający nazwę aktualnie odczytywanego pliku tabeli, bez ścieżki, ale z uwzględnieniem rozszerzenia.

```
FileName ()
```

FilePath

Funkcja **FilePath** zwraca ciąg znaków zawierający pełną ścieżkę do aktualnie odczytywanego pliku tabeli.

```
FilePath ()
```

FileSize

Funkcja **FileSize** zwraca liczbę całkowitą zawierającą rozmiar (w bajtach) pliku filename albo, jeśli nie określono parametru filename, aktualnie odczytywanego pliku tabeli.

```
FileSize ()
```

FileTime

Funkcja **FileTime** zwraca znacznik czasu daty i godziny w UTC ostatniej modyfikacji pliku filename. Jeśli nie określono parametru filename, funkcja odwoła się do aktualnie odczytywanego pliku tabeli.

```
FileTime ([ filename ])
```

GetFolderPath

Funkcja **GetFolderPath** zwraca wartość funkcji Microsoft Windows *SHGetFolderPath*. Ta funkcja przyjmuje na wejściu nazwę folderu Microsoft Windows i zwraca pełną ścieżkę do tego folderu.

```
GetFolderPath ()
```

QvdCreateTime

Ta funkcja skryptu zwraca znacznik czasu z nagłówkiem XML z ewentualnego pliku QVD. W przeciwnym wypadku zwraca wartość NULL. W znaczniku czasu jest określany czas UTC.

```
QvdCreateTime (filename)
```

QvdFieldName

Funkcja skryptu zwraca nazwę numeru pola (**fieldno**) w pliku QVD. Jeśli nie istnieje, zwracana jest wartość NULL.

```
QvdFieldName (filename , fieldno)
```

QvdNoOfFields

Ta funkcja skryptu zwraca liczbę pól w pliku QVD.

```
QvdNoOfFields (filename)
```

QvdNoOfRecords

Ta funkcja skryptu zwraca liczbę rekordów aktualnie istniejących w pliku QVD.

```
QvdNoOfRecords (filename)
```

QvdTableName

Ta funkcja skryptu zwraca nazwę tabeli zapisaną w pliku QVD.

```
QvdTableName (filename)
```

Attribute

Ta funkcja skryptu zwraca wartość metaznaczników różnych plików multimedialnych w postaci tekstu. Obsługiwane są następujące formaty plików: MP3, WMA, WMV, PNG oraz JPG. Jeśli parametr **filename** pliku nie istnieje, nie jest w obsługiwanym formacie pliku albo nie zawiera metaznacznika o nazwie **attributename**, zwrócone zostanie NULL.

Składnia:

```
Attribute(filename, attributename)
```

Istnieje możliwość odczytu wielu metaznaczników Na przykładach w tym temacie pokażemy, które znaczniki mogą być odczytywane dla odpowiednich obsługiwanych typów plików.

> *Można odczytać tylko metaznaczniki zapisane w pliku zgodnie z odpowiednią specyfikacją, na przykład ID2v3 dla plików MP3 lub EXIF dla plików JPG, a nie dane znaczników zapisane w **Eksploratorze plików w systemie Windows**.*

Argumenty:

Argumenty

Argument	Opis
filename	Nazwa pliku multimedialnego z ewentualnym uwzględnieniem ścieżki jako połączenie do danych w folderze. **Przykład:** *'lib://Table Files/'* W dotychczasowym trybie tworzenia skryptów obsługiwane są również następujące formaty ścieżek: • bezwzględna **Przykład:** *c:\data* • względna wobec katalogu roboczego aplikacji Qlik Sense. **Przykład:** *data*
attributename	Nazwa metaznacznika.

W przykładach w celu znalezienia ścieżek do plików multimedialnych używana jest funkcja **GetFolderPath**. Funkcja **GetFolderPath** jest obsługiwana tylko w trybie zgodności, dlatego odniesienia do funkcji **GetFolderPath** należy zastąpić ścieżką połączenia do danych lib://, gdy funkcja jest używana w trybie standardowym lub Qlik Sense SaaS.

Ograniczenie dostępu do systemu plików (page 1447)

Example 1: Pliki MP3

Ten skrypt odczytuje wszystkie możliwe metaznaczniki MP3 w folderze *MyMusic*.

```
// Script to read MP3 meta tags for each vExt in 'mp3' for each vFoundFile in filelist(
GetFolderPath('MyMusic') & '\*.'& vExt ) FileList: LOAD FileLongName,     subfield
(FileLongName,'\',-1) as FileShortName,     num(FileSize(FileLongName),'# ### ### ###',',','
') as FileSize,     FileTime(FileLongName) as FileTime,      // ID3v1.0 and ID3v1.1 tags
Attribute(FileLongName, 'Title') as Title,     Attribute(FileLongName, 'Artist') as Artist,
 Attribute(FileLongName, 'Album') as Album,     Attribute(FileLongName, 'Year') as Year,
Attribute(FileLongName, 'Comment') as Comment,     Attribute(FileLongName, 'Track') as Track,
   Attribute(FileLongName, 'Genre') as Genre,
```

```
    // ID3v2.3 tags     Attribute(FileLongName, 'AENC') as AENC, // Audio encryption
Attribute(FileLongName, 'APIC') as APIC, // Attached picture     Attribute(FileLongName,
'COMM') as COMM, // Comments     Attribute(FileLongName, 'COMR') as COMR, // Commercial frame
   Attribute(FileLongName, 'ENCR') as ENCR, // Encryption method registration     Attribute
(FileLongName, 'EQUA') as EQUA, // Equalization     Attribute(FileLongName, 'ETCO') as ETCO,
// Event timing codes     Attribute(FileLongName, 'GEOB') as GEOB, // General encapsulated
object     Attribute(FileLongName, 'GRID') as GRID, // Group identification registration
Attribute(FileLongName, 'IPLS') as IPLS, // Involved people list     Attribute(FileLongName,
'LINK') as LINK, // Linked information     Attribute(FileLongName, 'MCDI') as MCDI, // Music
CD identifier     Attribute(FileLongName, 'MLLT') as MLLT, // MPEG location lookup table
Attribute(FileLongName, 'OWNE') as OWNE, // Ownership frame     Attribute(FileLongName,
'PRIV') as PRIV, // Private frame     Attribute(FileLongName, 'PCNT') as PCNT, // Play counter
    Attribute(FileLongName, 'POPM') as POPM, // Popularimeter

    Attribute(FileLongName, 'POSS') as POSS, // Position synchronisation frame     Attribute
(FileLongName, 'RBUF') as RBUF, // Recommended buffer size     Attribute(FileLongName, 'RVAD')
as RVAD, // Relative volume adjustment     Attribute(FileLongName, 'RVRB') as RVRB, // Reverb
   Attribute(FileLongName, 'SYLT') as SYLT, // Synchronized lyric/text     Attribute
(FileLongName, 'SYTC') as SYTC, // Synchronized tempo codes     Attribute(FileLongName,
'TALB') as TALB, // Album/Movie/Show title     Attribute(FileLongName, 'TBPM') as TBPM, // BPM
(beats per minute)     Attribute(FileLongName, 'TCOM') as TCOM, // Composer     Attribute
(FileLongName, 'TCON') as TCON, // Content type     Attribute(FileLongName, 'TCOP') as TCOP,
// Copyright message     Attribute(FileLongName, 'TDAT') as TDAT, // Date     Attribute
(FileLongName, 'TDLY') as TDLY, // Playlist delay

    Attribute(FileLongName, 'TENC') as TENC, // Encoded by     Attribute(FileLongName,
'TEXT') as TEXT, // Lyricist/Text writer     Attribute(FileLongName, 'TFLT') as TFLT, // File
type     Attribute(FileLongName, 'TIME') as TIME, // Time     Attribute(FileLongName, 'TIT1')
as TIT1, // Content group description     Attribute(FileLongName, 'TIT2') as TIT2, //
Title/songname/content description     Attribute(FileLongName, 'TIT3') as TIT3, //
Subtitle/Description refinement     Attribute(FileLongName, 'TKEY') as TKEY, // Initial key
 Attribute(FileLongName, 'TLAN') as TLAN, // Language(s)     Attribute(FileLongName, 'TLEN')
as TLEN, // Length     Attribute(FileLongName, 'TMED') as TMED, // Media type

    Attribute(FileLongName, 'TOAL') as TOAL, // Original album/movie/show title     Attribute
(FileLongName, 'TOFN') as TOFN, // Original filename     Attribute(FileLongName, 'TOLY') as
TOLY, // Original lyricist(s)/text writer(s)     Attribute(FileLongName, 'TOPE') as TOPE, //
Original artist(s)/performer(s)     Attribute(FileLongName, 'TORY') as TORY, // Original
release year     Attribute(FileLongName, 'TOWN') as TOWN, // File owner/licensee     Attribute
(FileLongName, 'TPE1') as TPE1, // Lead performer(s)/Soloist(s)     Attribute(FileLongName,
'TPE2') as TPE2, // Band/orchestra/accompaniment

    Attribute(FileLongName, 'TPE3') as TPE3, // Conductor/performer refinement     Attribute
(FileLongName, 'TPE4') as TPE4, // Interpreted, remixed, or otherwise modified by
Attribute(FileLongName, 'TPOS') as TPOS, // Part of a set     Attribute(FileLongName, 'TPUB')
as TPUB, // Publisher     Attribute(FileLongName, 'TRCK') as TRCK, // Track number/Position in
set     Attribute(FileLongName, 'TRDA') as TRDA, // Recording dates     Attribute
(FileLongName, 'TRSN') as TRSN, // Internet radio station name     Attribute(FileLongName,
'TRSO') as TRSO, // Internet radio station owner

    Attribute(FileLongName, 'TSIZ') as TSIZ, // Size     Attribute(FileLongName, 'TSRC') as
TSRC, // ISRC (international standard recording code)     Attribute(FileLongName, 'TSSE') as
TSSE, // Software/Hardware and settings used for encoding     Attribute(FileLongName, 'TYER')
as TYER, // Year     Attribute(FileLongName, 'TXXX') as TXXX, // User defined text information
frame     Attribute(FileLongName, 'UFID') as UFID, // Unique file identifier     Attribute
```

```
(FileLongName, 'USER') as USER, // Terms of use     Attribute(FileLongName, 'USLT') as USLT,
// Unsychronized lyric/text transcription     Attribute(FileLongName, 'WCOM') as WCOM, //
Commercial information     Attribute(FileLongName, 'WCOP') as WCOP, // Copyright/Legal
information

     Attribute(FileLongName, 'WOAF') as WOAF, // Official audio file webpage     Attribute
(FileLongName, 'WOAR') as WOAR, // Official artist/performer webpage     Attribute
(FileLongName, 'WOAS') as WOAS, // Official audio source webpage     Attribute(FileLongName,
'WORS') as WORS, // Official internet radio station homepage     Attribute(FileLongName,
'WPAY') as WPAY, // Payment     Attribute(FileLongName, 'WPUB') as WPUB, // Publishers
official webpage     Attribute(FileLongName, 'WXXX') as WXXX; // User defined URL link frame
LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no labels); Next vFoundFile Next vExt
```

Example 2: JPEG

Ten skrypt odczytuje wszystkie możliwe metaznaczniki EXIF z plików JPG w folderze *MyPictures*.

```
// Script to read Jpeg Exif meta tags for each vExt in 'jpg', 'jpeg', 'jpe', 'jfif', 'jif',
'jfi' for each vFoundFile in filelist( GetFolderPath('MyPictures') & '\*.'& vExt )

 FileList: LOAD FileLongName,     subfield(FileLongName,'\',-1) as FileShortName,     num
(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,     FileTime(FileLongName) as
FileTime,      // ************   Exif Main (IFD0) Attributes   ************     Attribute
(FileLongName, 'ImageWidth') as ImageWidth,     Attribute(FileLongName, 'ImageLength') as
ImageLength,     Attribute(FileLongName, 'BitsPerSample') as BitsPerSample,     Attribute
(FileLongName, 'Compression') as Compression,

    //  examples: 1=uncompressed, 2=CCITT, 3=CCITT 3, 4=CCITT 4,

    //5=LZW, 6=JPEG (old style), 7=JPEG, 8=Deflate, 32773=PackBits RLE,      Attribute
(FileLongName, 'PhotometricInterpretation') as PhotometricInterpretation,

    //  examples: 0=WhiteIsZero, 1=BlackIsZero, 2=RGB, 3=Palette, 5=CMYK, 6=YCbCr,
Attribute(FileLongName, 'ImageDescription') as ImageDescription,     Attribute(FileLongName,
'Make') as Make,     Attribute(FileLongName, 'Model') as Model,     Attribute(FileLongName,
'StripOffsets') as StripOffsets,     Attribute(FileLongName, 'Orientation') as Orientation,

    //  examples: 1=TopLeft, 2=TopRight, 3=BottomRight, 4=BottomLeft,

    // 5=LeftTop, 6=RightTop, 7=RightBottom, 8=LeftBottom,      Attribute(FileLongName,
'SamplesPerPixel') as SamplesPerPixel,     Attribute(FileLongName, 'RowsPerStrip') as
RowsPerStrip,     Attribute(FileLongName, 'StripByteCounts') as StripByteCounts,     Attribute
(FileLongName, 'XResolution') as XResolution,     Attribute(FileLongName, 'YResolution') as
YResolution,     Attribute(FileLongName, 'PlanarConfiguration') as PlanarConfiguration,

    //  examples: 1=chunky format, 2=planar format,      Attribute(FileLongName,
'ResolutionUnit') as ResolutionUnit,

    //  examples: 1=none, 2=inches, 3=centimeters,      Attribute(FileLongName,
'TransferFunction') as TransferFunction,     Attribute(FileLongName, 'Software') as Software,
   Attribute(FileLongName, 'DateTime') as DateTime,     Attribute(FileLongName, 'Artist') as
Artist,     Attribute(FileLongName, 'HostComputer') as HostComputer,     Attribute
(FileLongName, 'WhitePoint') as WhitePoint,     Attribute(FileLongName,
'PrimaryChromaticities') as PrimaryChromaticities,     Attribute(FileLongName,
```

```
'YCbCrCoefficients') as YCbCrCoefficients,     Attribute(FileLongName, 'YCbCrSubSampling') as
YCbCrSubSampling,     Attribute(FileLongName, 'YCbCrPositioning') as YCbCrPositioning,

    //  examples: 1=centered, 2=co-sited,      Attribute(FileLongName, 'ReferenceBlackWhite')
as ReferenceBlackWhite,     Attribute(FileLongName, 'Rating') as Rating,     Attribute
(FileLongName, 'RatingPercent') as RatingPercent,     Attribute(FileLongName,
'ThumbnailFormat') as ThumbnailFormat,

    //  examples: 0=Raw Rgb, 1=Jpeg,      Attribute(FileLongName, 'Copyright') as Copyright,
  Attribute(FileLongName, 'ExposureTime') as ExposureTime,     Attribute(FileLongName,
'FNumber') as FNumber,     Attribute(FileLongName, 'ExposureProgram') as ExposureProgram,

    //  examples: 0=Not defined, 1=Manual, 2=Normal program, 3=Aperture priority, 4=Shutter
priority,

    // 5=Creative program, 6=Action program, 7=Portrait mode, 8=Landscape mode, 9=Bulb,
Attribute(FileLongName, 'ISOSpeedRatings') as ISOSpeedRatings,     Attribute(FileLongName,
'TimeZoneOffset') as TimeZoneOffset,     Attribute(FileLongName, 'SensitivityType') as
SensitivityType,

    //  examples: 0=Unknown, 1=Standard output sensitivity (SOS), 2=Recommended exposure index
(REI),

    // 3=ISO speed, 4=Standard output sensitivity (SOS) and Recommended exposure index (REI),

    //5=Standard output sensitivity (SOS) and ISO Speed, 6=Recommended exposure index (REI)
and ISO Speed,

    // 7=Standard output sensitivity (SOS) and Recommended exposure index (REI) and ISO speed,
     Attribute(FileLongName, 'ExifVersion') as ExifVersion,     Attribute(FileLongName,
'DateTimeOriginal') as DateTimeOriginal,     Attribute(FileLongName, 'DateTimeDigitized') as
DateTimeDigitized,     Attribute(FileLongName, 'ComponentsConfiguration') as
ComponentsConfiguration,

    //  examples: 1=Y, 2=Cb, 3=Cr, 4=R, 5=G, 6=B,      Attribute(FileLongName,
'CompressedBitsPerPixel') as CompressedBitsPerPixel,     Attribute(FileLongName,
'ShutterSpeedValue') as ShutterSpeedValue,     Attribute(FileLongName, 'ApertureValue') as
ApertureValue,     Attribute(FileLongName, 'BrightnessValue') as BrightnessValue, //
examples: -1=Unknown,      Attribute(FileLongName, 'ExposureBiasValue') as ExposureBiasValue,
   Attribute(FileLongName, 'MaxApertureValue') as MaxApertureValue,     Attribute
(FileLongName, 'SubjectDistance') as SubjectDistance,

    //  examples: 0=Unknown, -1=Infinity,      Attribute(FileLongName, 'MeteringMode') as
MeteringMode,

    //  examples: 0=Unknown, 1=Average, 2=CenterWeightedAverage, 3=Spot,

    // 4=MultiSpot, 5=Pattern, 6=Partial, 255=Other,      Attribute(FileLongName,
'LightSource') as LightSource,

    //  examples: 0=Unknown, 1=Daylight, 2=Fluorescent, 3=Tungsten, 4=Flash, 9=Fine weather,

    // 10=Cloudy weather, 11=Shade, 12=Daylight fluorescent,

    // 13=Day white fluorescent, 14=Cool white fluorescent,
```

```
    // 15=White fluorescent, 17=Standard light A, 18=Standard light B, 19=Standard light C,

    // 20=D55, 21=D65, 22=D75, 23=D50, 24=ISO studio tungsten, 255=other light source,
Attribute(FileLongName, 'Flash') as Flash,     Attribute(FileLongName, 'FocalLength') as
FocalLength,     Attribute(FileLongName, 'SubjectArea') as SubjectArea,     Attribute
(FileLongName, 'MakerNote') as MakerNote,     Attribute(FileLongName, 'UserComment') as
UserComment,     Attribute(FileLongName, 'SubSecTime') as SubSecTime,

     Attribute(FileLongName, 'SubsecTimeOriginal') as SubsecTimeOriginal,     Attribute
(FileLongName, 'SubsecTimeDigitized') as SubsecTimeDigitized,     Attribute(FileLongName,
'XPTitle') as XPTitle,     Attribute(FileLongName, 'XPComment') as XPComment,

     Attribute(FileLongName, 'XPAuthor') as XPAuthor,     Attribute(FileLongName,
'XPKeywords') as XPKeywords,     Attribute(FileLongName, 'XPSubject') as XPSubject,
Attribute(FileLongName, 'FlashpixVersion') as FlashpixVersion,     Attribute(FileLongName,
'ColorSpace') as ColorSpace, //  examples: 1=sRGB, 65535=Uncalibrated,      Attribute
(FileLongName, 'PixelXDimension') as PixelXDimension,     Attribute(FileLongName,
'PixelYDimension') as PixelYDimension,     Attribute(FileLongName, 'RelatedSoundFile') as
RelatedSoundFile,

     Attribute(FileLongName, 'FocalPlaneXResolution') as FocalPlaneXResolution,     Attribute
(FileLongName, 'FocalPlaneYResolution') as FocalPlaneYResolution,     Attribute(FileLongName,
'FocalPlaneResolutionUnit') as FocalPlaneResolutionUnit,

    //  examples: 1=None, 2=Inch, 3=Centimeter,      Attribute(FileLongName, 'ExposureIndex')
as ExposureIndex,     Attribute(FileLongName, 'SensingMethod') as SensingMethod,

    //  examples: 1=Not defined, 2=One-chip color area sensor, 3=Two-chip color area sensor,

    // 4=Three-chip color area sensor, 5=Color sequential area sensor,

    // 7=Trilinear sensor, 8=Color sequential linear sensor,      Attribute(FileLongName,
'FileSource') as FileSource,

    //  examples: 0=Other, 1=Scanner of transparent type,

    // 2=Scanner of reflex type, 3=Digital still camera,      Attribute(FileLongName,
'SceneType') as SceneType,

    //  examples: 1=A directly photographed image,      Attribute(FileLongName, 'CFAPattern')
as CFAPattern,     Attribute(FileLongName, 'CustomRendered') as CustomRendered,

    //  examples: 0=Normal process, 1=Custom process,      Attribute(FileLongName,
'ExposureMode') as ExposureMode,

    //  examples: 0=Auto exposure, 1=Manual exposure, 2=Auto bracket,      Attribute
(FileLongName, 'WhiteBalance') as WhiteBalance,

    //  examples: 0=Auto white balance, 1=Manual white balance,      Attribute(FileLongName,
'DigitalZoomRatio') as DigitalZoomRatio,     Attribute(FileLongName, 'FocalLengthIn35mmFilm')
as FocalLengthIn35mmFilm,     Attribute(FileLongName, 'SceneCaptureType') as SceneCaptureType,

    //  examples: 0=Standard, 1=Landscape, 2=Portrait, 3=Night scene,      Attribute
(FileLongName, 'GainControl') as GainControl,
```

```
    //  examples: 0=None, 1=Low gain up, 2=High gain up, 3=Low gain down, 4=High gain down,
  Attribute(FileLongName, 'Contrast') as Contrast,

    //  examples: 0=Normal, 1=Soft, 2=Hard,      Attribute(FileLongName, 'Saturation') as
Saturation,

    //  examples: 0=Normal, 1=Low saturation, 2=High saturation,      Attribute(FileLongName,
'Sharpness') as Sharpness,

    //  examples: 0=Normal, 1=Soft, 2=Hard,      Attribute(FileLongName,
'SubjectDistanceRange') as SubjectDistanceRange,

    //  examples: 0=Unknown, 1=Macro, 2=Close view, 3=Distant view,      Attribute
(FileLongName, 'ImageUniqueID') as ImageUniqueID,     Attribute(FileLongName,
'BodySerialNumber') as BodySerialNumber,     Attribute(FileLongName, 'CMNT_GAMMA') as CMNT_
GAMMA,     Attribute(FileLongName, 'PrintImageMatching') as PrintImageMatching,     Attribute
(FileLongName, 'OffsetSchema') as OffsetSchema,

     // ************   Interoperability Attributes   ************     Attribute(FileLongName,
'InteroperabilityIndex') as InteroperabilityIndex,     Attribute(FileLongName,
'InteroperabilityVersion') as InteroperabilityVersion,     Attribute(FileLongName,
'InteroperabilityRelatedImageFileFormat') as InteroperabilityRelatedImageFileFormat,
Attribute(FileLongName, 'InteroperabilityRelatedImageWidth') as
InteroperabilityRelatedImageWidth,     Attribute(FileLongName,
'InteroperabilityRelatedImageLength') as InteroperabilityRelatedImageLength,     Attribute
(FileLongName, 'InteroperabilityColorSpace') as InteroperabilityColorSpace,

    //  examples: 1=sRGB, 65535=Uncalibrated,      Attribute(FileLongName,
'InteroperabilityPrintImageMatching') as InteroperabilityPrintImageMatching,     //
************   GPS Attributes   ************     Attribute(FileLongName, 'GPSVersionID') as
GPSVersionID,     Attribute(FileLongName, 'GPSLatitudeRef') as GPSLatitudeRef,     Attribute
(FileLongName, 'GPSLatitude') as GPSLatitude,     Attribute(FileLongName, 'GPSLongitudeRef')
as GPSLongitudeRef,     Attribute(FileLongName, 'GPSLongitude') as GPSLongitude,     Attribute
(FileLongName, 'GPSAltitudeRef') as GPSAltitudeRef,

    //  examples: 0=Above sea level, 1=Below sea level,      Attribute(FileLongName,
'GPSAltitude') as GPSAltitude,     Attribute(FileLongName, 'GPSTimeStamp') as GPSTimeStamp,
 Attribute(FileLongName, 'GPSSatellites') as GPSSatellites,     Attribute(FileLongName,
'GPSStatus') as GPSStatus,     Attribute(FileLongName, 'GPSMeasureMode') as GPSMeasureMode,
 Attribute(FileLongName, 'GPSDOP') as GPSDOP,     Attribute(FileLongName, 'GPSSpeedRef') as
GPSSpeedRef,

     Attribute(FileLongName, 'GPSSpeed') as GPSSpeed,     Attribute(FileLongName,
'GPSTrackRef') as GPSTrackRef,     Attribute(FileLongName, 'GPSTrack') as GPSTrack,
Attribute(FileLongName, 'GPSImgDirectionRef') as GPSImgDirectionRef,     Attribute
(FileLongName, 'GPSImgDirection') as GPSImgDirection,     Attribute(FileLongName,
'GPSMapDatum') as GPSMapDatum,     Attribute(FileLongName, 'GPSDestLatitudeRef') as
GPSDestLatitudeRef,

     Attribute(FileLongName, 'GPSDestLatitude') as GPSDestLatitude,     Attribute
(FileLongName, 'GPSDestLongitudeRef') as GPSDestLongitudeRef,     Attribute(FileLongName,
'GPSDestLongitude') as GPSDestLongitude,     Attribute(FileLongName, 'GPSDestBearingRef') as
GPSDestBearingRef,     Attribute(FileLongName, 'GPSDestBearing') as GPSDestBearing,
Attribute(FileLongName, 'GPSDestDistanceRef') as GPSDestDistanceRef,
```

```
    Attribute(FileLongName, 'GPSDestDistance') as GPSDestDistance,    Attribute
(FileLongName, 'GPSProcessingMethod') as GPSProcessingMethod,    Attribute(FileLongName,
'GPSAreaInformation') as GPSAreaInformation,    Attribute(FileLongName, 'GPSDateStamp') as
GPSDateStamp,    Attribute(FileLongName, 'GPSDifferential') as GPSDifferential;

   //  examples: 0=No correction, 1=Differential correction,  LOAD @1:n as FileLongName
Inline "$(vFoundFile)" (fix, no labels); Next vFoundFile Next vExt
```

Example 3: Pliki multimedialne Windows

Ten skrypt odczytuje wszystkie możliwe metaznaczniki WMA/WMV ASF w folderze *MyMusic*.

```
/ Script to read WMA/WMV ASF meta tags for each vExt in 'asf', 'wma', 'wmv' for each
vFoundFile in filelist( GetFolderPath('MyMusic') & '\*.'& vExt )

 FileList: LOAD FileLongName,    subfield(FileLongName,'\',-1) as FileShortName,    num
(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,    FileTime(FileLongName) as
FileTime,     Attribute(FileLongName, 'Title') as Title,    Attribute(FileLongName,
'Author') as Author,    Attribute(FileLongName, 'Copyright') as Copyright,    Attribute
(FileLongName, 'Description') as Description,

    Attribute(FileLongName, 'Rating') as Rating,    Attribute(FileLongName, 'PlayDuration')
as PlayDuration,    Attribute(FileLongName, 'MaximumBitrate') as MaximumBitrate,
Attribute(FileLongName, 'WMFSDKVersion') as WMFSDKVersion,    Attribute(FileLongName,
'WMFSDKNeeded') as WMFSDKNeeded,    Attribute(FileLongName, 'IsVBR') as IsVBR,    Attribute
(FileLongName, 'ASFLeakyBucketPairs') as ASFLeakyBucketPairs,

    Attribute(FileLongName, 'PeakValue') as PeakValue,    Attribute(FileLongName,
'AverageLevel') as AverageLevel; LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no
labels); Next vFoundFile Next vExt
```

Example 4: PNG

Ten skrypt odczytuje wszystkie możliwe metaznaczniki PNG w folderze *MyPictures*.

```
// Script to read PNG meta tags for each vExt in 'png' for each vFoundFile in filelist(
GetFolderPath('MyPictures') & '\*.'& vExt )

 FileList: LOAD FileLongName,    subfield(FileLongName,'\',-1) as FileShortName,    num
(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,    FileTime(FileLongName) as
FileTime,     Attribute(FileLongName, 'Comment') as Comment,

    Attribute(FileLongName, 'Creation Time') as Creation_Time,    Attribute(FileLongName,
'Source') as Source,    Attribute(FileLongName, 'Title') as Title,    Attribute
(FileLongName, 'Software') as Software,    Attribute(FileLongName, 'Author') as Author,
Attribute(FileLongName, 'Description') as Description,

    Attribute(FileLongName, 'Copyright') as Copyright; LOAD @1:n as FileLongName Inline
"$(vFoundFile)" (fix, no labels); Next vFoundFile Next vExt
```

ConnectString

Funkcja **ConnectString()** zwraca nazwę aktywnego połączenia do danych dla połączeń ODBC lub OLE DB. Funkcja zwraca pusty ciąg znaków, jeśli nie wykonano instrukcji **connect** lub po instrukcji **disconnect**.

Składnia:

```
ConnectString()
```

Przykłady i wyniki:

Przykłady skryptów

Przykład	Wynik

`LIB CONNECT TO 'Tutorial ODBC';`
`ConnectString:`
`Load ConnectString() as ConnectString AutoGenerate 1;`
	Zwraca 'Tutorial ODBC' w polu ConnectString.
	
	Na potrzeby tego przykładu przyjęto założenie, że dostępne jest połączenie między danymi o nazwie Tutorial ODBC.

FileBaseName

Funkcja **FileBaseName** zwraca ciąg znaków zawierający nazwę aktualnie odczytywanego pliku tabeli, bez ścieżki ani rozszerzenia.

Składnia:

```
FileBaseName()
```

Przykłady i wyniki:

Przykłady skryptów

Przykład	Wynik
`LOAD *, filebasename( ) as X from C:\UserFiles\abc.txt`	Zwróci „abc” w polu X w każdym odczytanym rekordzie.

FileDir

Funkcja **FileDir** zwraca ciąg znaków zawierający ścieżkę do katalogu aktualnie odczytywanego pliku tabeli.

Składnia:

```
FileDir()
```

Ta funkcja obsługuje tylko powiązania do danych w folderze w trybie standardowym.

Przykłady i wyniki:

Przykłady skryptów

Przykład	Wynik

`Load *, filedir( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Zwróci „C:\UserFiles” w polu X w każdym odczytanym rekordzie.

FileExtension

Funkcja **FileExtension** zwraca ciąg znaków zawierający rozszerzenie aktualnie odczytywanego pliku tabeli.

Składnia:

```
FileExtension()
```

Przykłady i wyniki:

Przykłady skryptów

Przykład	Wynik
`LOAD *, FileExtension( ) as X from C:\UserFiles\abc.txt`	Zwróci „txt” w polu X w każdym odczytanym rekordzie.

FileName

Funkcja **FileName** zwraca ciąg znaków zawierający nazwę aktualnie odczytywanego pliku tabeli, bez ścieżki, ale z uwzględnieniem rozszerzenia.

Składnia:

```
FileName()
```

Przykłady i wyniki:

Przykłady skryptów

Przykład	Wynik

`LOAD *, FileName( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Zwróci 'abc.txt' w polu X w każdym odczytanym rekordzie.

FilePath

Funkcja **FilePath** zwraca ciąg znaków zawierający pełną ścieżkę do aktualnie odczytywanego pliku tabeli.

Składnia:

```
FilePath()
```

Ta funkcja obsługuje tylko powiązania do danych w folderze w trybie standardowym.

Przykłady i wyniki:

Przykłady skryptów

Przykład	Wynik

`Load *, FilePath( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Zwróci 'C:\UserFiles\abc.txt' w polu X w każdym odczytanym rekordzie.

FileSize

Funkcja **FileSize** zwraca liczbę całkowitą zawierającą rozmiar (w bajtach) pliku filename albo, jeśli nie określono parametru filename, aktualnie odczytywanego pliku tabeli.

Składnia:

`FileSize(``[filename]`**`)`**

Argumenty:

Argumenty

Argument	Opis
filename	Nazwa pliku w razie potrzeby ze ścieżką, jako folder lub połączenia do danych pliku webowego. Jeśli nazwa pliku nie zostanie określona, użyty zostanie aktualnie odczytywany plik tabeli. **Przykład:** *'lib://Table Files/'* W dotychczasowym trybie tworzenia skryptów obsługiwane są również następujące formaty ścieżek: • bezwzględna **Przykład:** *c:\data* • względna wobec katalogu roboczego aplikacji Qlik Sense. **Przykład:** *data* • Adres URL (HTTP lub FTP), wskazujący lokalizację w Internecie lub intranecie. **Przykład:** *http://www.qlik.com*

Przykłady i wyniki:

Przykłady skryptów

Przykład	Wynik
`LOAD *, FileSize( ) as X from abc.txt;`	Zwróci rozmiar określonego pliku (abc.txt) jako liczbę całkowitą w polu X w każdym odczytywanym rekordzie.
`FileSize( 'lib://DataFiles/xyz.xls' )`	Zwróci rozmiar pliku xyz.xls.

FileTime

Funkcja **FileTime** zwraca znacznik czasu daty i godziny w UTC ostatniej modyfikacji pliku filename. Jeśli nie określono parametru filename, funkcja odwoła się do aktualnie odczytywanego pliku tabeli.

Składnia:

```
FileTime([ filename ])
```

Argumenty:

Argumenty

Argument	Opis
filename	Nazwa pliku w razie potrzeby ze ścieżką, jako folder lub połączenia do danych pliku webowego. **Przykład:** *'lib://Table Files/'* W dotychczasowym trybie tworzenia skryptów obsługiwane są również następujące formaty ścieżek: • bezwzględna **Przykład:** *c:\data* • względna wobec katalogu roboczego aplikacji Qlik Sense. **Przykład:** *data* • Adres URL (HTTP lub FTP), wskazujący lokalizację w Internecie lub intranecie. **Przykład:** *http://www.qlik.com*

Przykłady i wyniki:

Przykłady skryptów

Przykład	Wynik
`LOAD *, FileTime( ) as X from abc.txt;`	Zwróci datę i godzinę ostatniej modyfikacji pliku (abc.txt) jako znacznik czasu w polu X w każdym odczytywanym rekordzie.
FileTime('xyz.xls')	Zwróci znacznik czasu ostatniej modyfikacji pliku xyz.xls.

GetFolderPath

Funkcja **GetFolderPath** zwraca wartość funkcji Microsoft Windows *SHGetFolderPath*. Ta funkcja przyjmuje na wejściu nazwę folderu Microsoft Windows i zwraca pełną ścieżkę do tego folderu.

Ta funkcja nie jest obsługiwana w trybie standardowym.

Składnia:

```
GetFolderPath(foldername)
```

Argumenty:

Argumenty

Argument	Opis
foldername	Nazwa folderu Microsoft Windows. Nazwa folderu nie powinna zawierać spacji. Wszystkie spacje w nazwie folderu wyświetlanego w programie Windows Explorer powinny być usunięte. Przykłady: *MyMusic* *MyDocuments*

Przykłady i wyniki:

Celem w tym przykładzie jest pobranie ścieżek następujących folderów w systemie Microsoft Windows: *MyMusic*, *MyPictures* oraz *Windows*. Dodaj przykładowy skrypt do aplikacji i przeładuj ją.

```
LOAD   GetFolderPath('MyMusic') as MyMusic,  GetFolderPath('MyPictures') as MyPictures,
GetFolderPath('Windows') as Windows AutoGenerate 1;
```

Po przeładowaniu aplikacji pola *MyMusic*, *MyPictures* i *Windows* zostały dodane do modelu danych. W każdym polu znajduje się ścieżka do folderu zdefiniowanego w danych wejściowych. Na przykład:

- *C:\Users\smu\Music* for the folder *MyMusic*
- *C:\Users\smu\Pictures* for the folder *MyPictures*
- *C:\Windows* for the folder *Windows*

QvdCreateTime

Ta funkcja skryptu zwraca znacznik czasu z nagłówkiem XML z ewentualnego pliku QVD. W przeciwnym wypadku zwraca wartość NULL. W znaczniku czasu jest określany czas UTC.

Składnia:

```
QvdCreateTime(filename)
```

Argumenty:

Argumenty

Argument	Opis
filename	Nazwa pliku QVD, w razie potrzeby ze ścieżką, jako folder lub połączenia sieciowe do danych. **Przykład:** *'lib://Table Files/'* W dotychczasowym trybie tworzenia skryptów obsługiwane są również następujące formaty ścieżek: • bezwzględna **Przykład:** *c:\data* • względna wobec katalogu roboczego aplikacji Qlik Sense. **Przykład:** *data* • Adres URL (HTTP lub FTP), wskazujący lokalizację w Internecie lub intranecie. **Przykład:** *http://www.qlik.com*

Przykład:

```
QvdCreateTime('MyFile.qvd')

QvdCreateTime('C:\MyDir\MyFile.qvd')

QvdCreateTime('lib://DataFiles/MyFile.qvd')
```

QvdFieldName

Funkcja skryptu zwraca nazwę numeru pola (**fieldno**) w pliku QVD. Jeśli nie istnieje, zwracana jest wartość NULL.

Składnia:

```
QvdFieldName(filename , fieldno)
```

Argumenty:

Argumenty

Argument	Opis
filename	Nazwa pliku QVD, w razie potrzeby ze ścieżką, jako folder lub połączenia sieciowe do danych. **Przykład:** *'lib://Table Files/'* W dotychczasowym trybie tworzenia skryptów obsługiwane są również następujące formaty ścieżek: • bezwzględna **Przykład:** *c:\data* • względna wobec katalogu roboczego aplikacji Qlik Sense. **Przykład:** *data* • Adres URL (HTTP lub FTP), wskazujący lokalizację w Internecie lub intranecie. **Przykład:** *http://www.qlik.com*
fieldno	Numer pola w ramach tabeli zawartej w pliku QVD.

Przykłady:

```
QvdFieldName ('MyFile.qvd', 5)

QvdFieldName ('C:\MyDir\MyFile.qvd', 5)

QvdFieldName ('lib://DataFiles/MyFile.qvd', 5)
```

Wszystkie trzy przykłady zwracają nazwę piątego pola tabeli zawartego w pliku QVD.

QvdNoOfFields

Ta funkcja skryptu zwraca liczbę pól w pliku QVD.

Składnia:

```
QvdNoOfFields(filename)
```

Argumenty:

Argumenty

Argument	Opis
filename	Nazwa pliku QVD, w razie potrzeby ze ścieżką, jako folder lub połączenia sieciowe do danych. **Przykład:** *'lib://Table Files/'* W dotychczasowym trybie tworzenia skryptów obsługiwane są również następujące formaty ścieżek: • bezwzględna **Przykład:** *c:\data* • względna wobec katalogu roboczego aplikacji Qlik Sense. **Przykład:** *data* • Adres URL (HTTP lub FTP), wskazujący lokalizację w Internecie lub intranecie. **Przykład:** *http://www.qlik.com*

Przykłady:

```
QvdNoOfFields ('MyFile.qvd')

QvdNoOfFields ('C:\MyDir\MyFile.qvd')

QvdNoOfFields ('lib://DataFiles/MyFile.qvd')
```

QvdNoOfRecords

Przykład: Ta funkcja skryptu zwraca liczbę rekordów aktualnie istniejących w pliku QVD.

Składnia:

```
QvdNoOfRecords(filename)
```

Argumenty:

Argumenty

Argument	Opis
filename	Nazwa pliku QVD, w razie potrzeby ze ścieżką, jako folder lub połączenia sieciowe do danych. **Przykład:** *'lib://Table Files/'* W dotychczasowym trybie tworzenia skryptów obsługiwane są również następujące formaty ścieżek: • bezwzględna **Przykład:** *c:\data* • względna wobec katalogu roboczego aplikacji Qlik Sense. **Przykład:** *data* • Adres URL (HTTP lub FTP), wskazujący lokalizację w Internecie lub intranecie. **Przykład:** *http://www.qlik.com*

Przykłady:

```
QvdNoOfRecords ('MyFile.qvd')

QvdNoOfRecords ('C:\MyDir\MyFile.qvd')

QvdNoOfRecords ('lib://DataFiles/MyFile.qvd')
```

QvdTableName

Ta funkcja skryptu zwraca nazwę tabeli zapisaną w pliku QVD.

Składnia:

```
QvdTableName(filename)
```

Argumenty:

Argumenty

Argument	Opis
filename	Nazwa pliku QVD, w razie potrzeby ze ścieżką, jako folder lub połączenia sieciowe do danych. **Przykład:** *'lib://Table Files/'* W dotychczasowym trybie tworzenia skryptów obsługiwane są również następujące formaty ścieżek: • bezwzględna **Przykład:** *c:\data* • względna wobec katalogu roboczego aplikacji Qlik Sense. **Przykład:** *data* • Adres URL (HTTP lub FTP), wskazujący lokalizację w Internecie lub intranecie. **Przykład:** *http://www.qlik.com*

Przykłady:

```
QvdTableName ('MyFile.qvd')
QvdTableName ('C:\MyDir\MyFile.qvd')
QvdTableName ('lib://data\MyFile.qvd')
```

5.11 Funkcje finansowe

Funkcje finansowe mogą być używane w skrypcie ładowania danych i w wyrażeniach wykresu w celu obliczania płatności i stóp procentowych.
W odniesieniu do wszystkich argumentów wypłacane środki są przedstawiane jako liczby ujemne. Środki otrzymane są przedstawiane jako liczby dodatnie.
Poniżej wymieniono argumenty używane w funkcjach finansowych (poza argumentami zaczynającymi się od **range**-).

*W przypadku wszystkich funkcji finansowych bardzo ważne jest spójne używanie jednostek na potrzeby określania argumentów **rate** i **nper**. W przypadku dokonywania miesięcznych płatności tytułem pięcioletniego kredytu z rocznym oprocentowaniem 6% należy użyć wartości 0,005 (6%/12) dla argumentu **rate** i 60 (5*12) dla argumentu **nper**. Jeśli w odniesieniu do tego samego kredytu płatności są dokonywane rocznie, należy użyć wartości 6% dla argumentu **rate** i 5 dla parametru **nper**.*

Przegląd funkcji finansowych

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

FV

Ta funkcja zwraca przyszłą wartość inwestycji na podstawie okresowych stałych płatności i prostego oprocentowania rocznego.

```
FV (rate, nper, pmt [ ,pv [ , type ] ])
```

nPer

Ta funkcja zwraca liczbę okresów inwestycji na podstawie okresowych stałych płatności i stałej stopy procentowej.

```
nPer (rate, pmt, pv [ ,fv [ , type ] ])
```

Pmt

Ta funkcja zwraca płatność tytułem kredytu na podstawie okresowych stałych płatności i stałej stopy procentowej. Jej wartość nie może się zmieniać w okresie obowiązywania rozliczenia rocznego. Płatność jest wyrażona jako liczba ujemna, na przykład -20.

```
Pmt (rate, nper, pv [ ,fv [ , type ] ] )
```

PV

Ta funkcja zwraca bieżącą wartość inwestycji.

```
PV (rate, nper, pmt [ ,fv [ , type ] ])
```

Rate

Ta funkcja zwraca stopę procentową dla okresu w rozliczeniu rocznym. Wynik ma następujący domyślny format liczby: **Fix** – dwa miejsca po przecinku i %.

```
Rate (nper, pmt , pv [ ,fv [ , type ] ])
```

BlackAndSchole

Model Black and Scholes to model matematyczny dla instrumentów pochodnych istniejących na rynkach finansowych. Wzór umożliwia obliczenie teoretycznej wartości opcji finansowej. Funkcja **BlackAndSchole** w aplikacji Qlik Sense zwraca wartość według niemodyfikowanego wzoru Black and Scholes (opcje europejskie).

```
BlackAndSchole(strike , time_left , underlying_price , vol , risk_free_rate ,
type)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
strike	Przyszła cena zakupu akcji.
time_left	Liczba pozostałych okresów.
underlying_price	Obecna wartość akcji.
vol	Współczynnik zmienności (kursu giełdowego) wyrażony jako procent w postaci dziesiętnej, na okres czasu.
risk_free_rate	Wysokość stopy procentowej wolnej od ryzyka wyrażona jako procent w postaci dziesiętnej, na okres czasu.
call_or_put	Typ opcji: 'c', 'call' lub niezerowa wartość liczbowa w przypadku opcji kupna 'p', 'put' lub 0 w przypadku opcji sprzedaży.

Ograniczenia:

Wartości strike, time_left i underlying_price muszą być >0.

Wartości vol i risk_free_rate muszą być: <0 lub >0.

Przykłady i wyniki:

Przykłady skryptów

Przykład	Wynik
`BlackAndSchole(130, 4, 68.5, 0.4, 0.04, 'call')` Oblicza teoretyczną cenę opcji na zakup akcji wartej dziś 68,5, przy wartości 130 w ciągu 4 lat. W tym wzorze używany jest współczynnik zmienności równy 0,4 (40%) na rok i stopa procentowa wolna od ryzyka równa 0,04 (4%).	Zwraca wartość 11,245

FV

Ta funkcja zwraca przyszłą wartość inwestycji na podstawie okresowych stałych płatności i prostego oprocentowania rocznego.

Składnia:

```
FV(rate, nper, pmt [ ,pv [ , type ] ])
```

Typ zwracanych danych: liczbowy. Domyślnie wynik zostanie sformatowany jako waluta..

Argumenty:

Argumenty

Argument	Opis
rate	Stopa procentowa za okres.
nper	Łączna liczba okresów płatności w rozliczeniu rocznym.
pmt	Płatność dokonywana w każdym okresie. Jej wartość nie może się zmieniać w okresie obowiązywania rozliczenia rocznego. Płatność jest wyrażona jako liczba ujemna, na przykład -20.
pv	Wartość bieżąca lub łączna kwota odpowiadająca aktualnej wartości serii przyszłych płatności. W przypadku pominięcia argumentu **pv** przyjmuje się, że jego wartość wynosi 0 (zero).
type	Powinien mieć wartość 0, jeśli termin płatności przypada na koniec okresu, lub 1, jeśli termin płatności przypada na początek okresu. W przypadku pominięcia argumentu **type** przyjmuje się, że jego wartość wynosi 0.

Przykłady i wyniki:

Przykład skryptu

Przykład	Wynik
Spłacasz nowe urządzenie AGD w 36 miesięcznych ratach w wysokości 20 USD. Stopa procentowa wynosi 6% rocznie. Rachunek przychodzi na koniec każdego miesiąca. Jaka jest łączna wartość inwestycji po uregulowaniu ostatniego rachunku? `FV(0.005,36,-20)`	Zwraca wartość $786.72

nPer

Ta funkcja zwraca liczbę okresów inwestycji na podstawie okresowych stałych płatności i stałej stopy procentowej.

Składnia:

```
nPer(rate, pmt, pv [ ,fv [ , type ] ])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
rate	Stopa procentowa za okres.
nper	Łączna liczba okresów płatności w rozliczeniu rocznym.
pmt	Płatność dokonywana w każdym okresie. Jej wartość nie może się zmieniać w okresie obowiązywania rozliczenia rocznego. Płatność jest wyrażona jako liczba ujemna, na przykład -20.
pv	Wartość bieżąca lub łączna kwota odpowiadająca aktualnej wartości serii przyszłych płatności. W przypadku pominięcia argumentu **pv** przyjmuje się, że jego wartość wynosi 0 (zero).
fv	Wartość przyszła lub saldo środków pieniężnych, które chce się uzyskać po dokonaniu ostatniej płatności. W przypadku pominięcia argumentu **fv** przyjmuje się, że jego wartość wynosi 0.
type	Powinien mieć wartość 0, jeśli termin płatności przypada na koniec okresu, lub 1, jeśli termin płatności przypada na początek okresu. W przypadku pominięcia argumentu **type** przyjmuje się, że jego wartość wynosi 0.

Przykłady i wyniki:

Przykład skryptu

Przykład	Wynik
Chcesz sprzedać sprzęt AGD w miesięcznych ratach w wysokości 20 USD. Stopa procentowa wynosi 6% rocznie. Rachunek przychodzi na koniec każdego miesiąca. Ile potrzeba okresów, jeśli wartość środków po uregulowaniu ostatniego rachunku powinna wynieść 800 USD? `nPer(0.005,-20,0,800)`	Zwraca wartość 36,56

Pmt

Ta funkcja zwraca płatność tytułem kredytu na podstawie okresowych stałych płatności i stałej stopy procentowej. Jej wartość nie może się zmieniać w okresie obowiązywania rozliczenia rocznego. Płatność jest wyrażona jako liczba ujemna, na przykład -20.

```
Pmt(rate, nper, pv [ ,fv [ , type ] ] )
```

Typ zwracanych danych: liczbowy. Domyślnie wynik zostanie sformatowany jako waluta..

Aby otrzymać łączną kwotę zapłaconą w okresie kredytu, należy pomnożyć zwróconą wartość argumentu **pmt** przez **nper**.

Argumenty:

Argumenty

Argument	Opis
rate	Stopa procentowa za okres.
nper	Łączna liczba okresów płatności w rozliczeniu rocznym.
pv	Wartość bieżąca lub łączna kwota odpowiadająca aktualnej wartości serii przyszłych płatności. W przypadku pominięcia argumentu **pv** przyjmuje się, że jego wartość wynosi 0 (zero).
fv	Wartość przyszła lub saldo środków pieniężnych, które chce się uzyskać po dokonaniu ostatniej płatności. W przypadku pominięcia argumentu **fv** przyjmuje się, że jego wartość wynosi 0.
type	Powinien mieć wartość 0, jeśli termin płatności przypada na koniec okresu, lub 1, jeśli termin płatności przypada na początek okresu. W przypadku pominięcia argumentu **type** przyjmuje się, że jego wartość wynosi 0.

Przykłady i wyniki:

Przykłady skryptów

Przykład	Wynik
Poniższa formuła zwraca miesięczną płatność tytułem kredytu w wysokości $20 000 z roczną stopą procentową wynoszącą 10%, który należy spłacić w ciągu 8 miesięcy: `Pmt(0.1/12,8,20000)`	Zwraca wartość -$2,594.66
Jeśli w przypadku tego samego kredytu termin płatności przypada na początku miesiąca, płatność wynosi: `Pmt(0.1/12,8,20000,0,1)`	Zwraca wartość -$2,573.21

PV

Ta funkcja zwraca bieżącą wartość inwestycji.

```
PV(rate, nper, pmt [ ,fv [ , type ] ])
```

Typ zwracanych danych: liczbowy. Domyślnie wynik zostanie sformatowany jako waluta..

Wartość bieżąca to łączna kwota odpowiadająca aktualnej wartości serii przyszłych płatności. Na przykład w przypadku pożyczania od kogoś pieniędzy bieżącą wartością dla kredytodawcy jest kwota kredytu.

Argumenty:

Argumenty

Argument	Opis
rate	Stopa procentowa za okres.
nper	Łączna liczba okresów płatności w rozliczeniu rocznym.
pmt	Płatność dokonywana w każdym okresie. Jej wartość nie może się zmieniać w okresie obowiązywania rozliczenia rocznego. Płatność jest wyrażona jako liczba ujemna, na przykład -20.
fv	Wartość przyszła lub saldo środków pieniężnych, które chce się uzyskać po dokonaniu ostatniej płatności. W przypadku pominięcia argumentu **fv** przyjmuje się, że jego wartość wynosi 0.
type	Powinien mieć wartość 0, jeśli termin płatności przypada na koniec okresu, lub 1, jeśli termin płatności przypada na początek okresu. W przypadku pominięcia argumentu **type** przyjmuje się, że jego wartość wynosi 0.

Przykłady i wyniki:

Przykład skryptu

Przykład	Wynik
Jaka jest wartość bieżąca długu, jeśli musisz zapłacić 100 USD na koniec każdego miesiąca w okresie pięcioletnim, przyjmując roczną stopę procentową w wysokości 7%? `PV(0.07/12,12*5,-100,0,0)`	Zwraca wartość $5,050.20

Rate

Ta funkcja zwraca stopę procentową dla okresu w rozliczeniu rocznym. Wynik ma następujący domyślny format liczby: **Fix** – dwa miejsca po przecinku i %.

Składnia:

```
Rate(nper, pmt , pv [ ,fv [ , type ] ])
```

Typ zwracanych danych: liczbowy.

Argument **rate** jest obliczany przez iterację i może mieć zero lub kilka rozwiązań. Jeśli otrzymane wyniki dotyczące argumentu **rate** nie są zbieżne, zwracana jest wartość NULL.

Argumenty:

Argumenty

Argument	Opis
nper	Łączna liczba okresów płatności w rozliczeniu rocznym.
pmt	Płatność dokonywana w każdym okresie. Jej wartość nie może się zmieniać w okresie obowiązywania rozliczenia rocznego. Płatność jest wyrażona jako liczba ujemna, na przykład -20.
pv	Wartość bieżąca lub łączna kwota odpowiadająca aktualnej wartości serii przyszłych płatności. W przypadku pominięcia argumentu **pv** przyjmuje się, że jego wartość wynosi 0 (zero).
fv	Wartość przyszła lub saldo środków pieniężnych, które chce się uzyskać po dokonaniu ostatniej płatności. W przypadku pominięcia argumentu **fv** przyjmuje się, że jego wartość wynosi 0.
type	Powinien mieć wartość 0, jeśli termin płatności przypada na koniec okresu, lub 1, jeśli termin płatności przypada na początek okresu. W przypadku pominięcia argumentu **type** przyjmuje się, że jego wartość wynosi 0.

Przykłady i wyniki:

Przykład skryptu

Przykład	Wynik
Jaka jest stopa procentowa pięcioletniego kredytu w wysokości $10 000 spłacanego w równych ratach z miesięcznymi płatnościami w wysokości $300? `Rate(60,-300,10000)`	Zwraca wartość 2.00%

5.12 Funkcje formatowania

Funkcje formatowania narzucają format wyświetlania wejściowych pól lub wyrażeń liczbowych. W zależności od typu danych można określić znaki separatora dziesiętnego, separatora tysięcy itp.

Wszystkie funkcje zwracają wartość podwójną zawierającą zarówno ciąg znaków, jak i wartość liczbową, ale można je rozpatrywać jako przekształcenie liczby na ciąg znaków. Szczególnym przypadkiem jest funkcja **Dual()**, ale pozostałe funkcje formatowania pobierają wartość liczbową wyrażenia wejściowego i generują ciąg znaków reprezentujący liczbę.

Z kolei funkcje interpretacji działają odwrotnie, odczytując ciąg znaków z wyrażenia jako liczbę o określonym formacie wynikowym.

Funkcje te mogą być stosowane zarówno w skryptach ładowania danych, jak i wyrażeniach wykresu.

Wszystkie reprezentacje liczb są podawane z kropką jako separatorem dziesiętnym.

Przegląd funkcji formatowania

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

ApplyCodepage

ApplyCodepage() stosuje inny zestaw znaków strony kodowej względem pola albo tekstu określonego w wyrażeniu. Argument **codepage** musi być w formacie liczbowym.

```
ApplyCodepage (text, codepage)
```

Date

Funkcja **Date()** formatuje wyrażenie jako datę przy użyciu formatu skonfigurowanego w zmiennych systemowych w skrypcie ładowania danych, systemie operacyjnym lub ewentualnie w formacie ciągu znaków.

```
Date (number[, format])
```

Dual

Funkcja **Dual()** łączy liczbę i ciąg znaków w jeden rekord, dzięki czemu liczbowa reprezentacja rekordu może zostać użyta na potrzeby sortowania i obliczania (wartość ciągu znaków może natomiast zostać użyta do celów wyświetlania).

```
Dual (text, number)
```

Interval

Funkcja **Interval()** formatuje liczbę jako interwał czasowy przy użyciu formatu w zmiennych systemowych w skrypcie ładowania danych, systemie operacyjnym lub formacie ciągu znaków, jeśli został podany.

```
Interval (number[, format])
```

Money

Funkcja **Money()** formatuje wyrażenie liczbowo jako wartość pieniężną w formacie ustawionym w zmiennych systemowych w skrypcie ładowania danych lub systemie operacyjnym, chyba że podano format ciągu znaków oraz opcjonalny separator dziesiętny i separator tysięcy.

```
Money (number[, format[, dec_sep [, thou_sep]]])
```

Num

Funkcja **Num()** formatuje liczbę, to znaczy konwertuje wartość liczbową danych wejściowych na wyświetlany tekst przy użyciu formatu określonego w drugim parametrze. Jeśli drugi parametr zostanie pominięty, używa separatorów dziesiętnych i tysięcznych ustawionych w skrypcie ładowania danych. Parametrami opcjonalnymi są niestandardowe symbole separatora dziesiętnego i separatora tysięcy.

```
Num (number[, format[, dec_sep [, thou_sep]]])
```

Time

Funkcja **Time()** formatuje wyrażenie jako wartość czasu przy użyciu formatu czasu ustawionego w zmiennych systemowych w skrypcie ładowania danych lub systemie operacyjnym, chyba że podano format ciągu znaków.

```
Time (number[, format])
```

Timestamp

Funkcja **TimeStamp()** formatuje wyrażenie jako wartość daty i godziny przy użyciu formatu znacznika czasu ustawionego w zmiennych systemowych w skrypcie ładowania danych lub systemie operacyjnym, chyba że podano format ciągu znaków.

```
Timestamp (number[, format])
```

Zob. także:

p *Funkcje interpretacji (page 1241)*

ApplyCodepage

ApplyCodepage() stosuje inny zestaw znaków strony kodowej względem pola albo tekstu określonego w wyrażeniu. Argument **codepage** musi być w formacie liczbowym.

> *Polecenie ApplyCodepage może być używane w wyrażeniach wykresu, ale częściej jest używane jako funkcja skryptu w edytorze ładowania danych. Na przykład podczas ładowania plików, które mogły zostać zapisane w różnych zestawach znaków poza kontrolą użytkownika, można zastosować stronę kodową reprezentującą wymagany zestaw znaków.*

Składnia:

```
ApplyCodepage(text, codepage)
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
text	Pole lub tekst, względem którego wymagane jest zastosowanie innej strony kodowej, określone przez argument **codepage**.
codepage	Numer reprezentujący stronę kodową do zastosowania względem pola lub wyrażenia określonego przez **text**.

Przykłady i wyniki:

Przykłady skryptów

Przykład	Wynik
`LOAD ApplyCodepage (ROWX,1253) as GreekProduct, ApplyCodepage (ROWY, 1255) as HebrewProduct, ApplyCodepage (ROWZ, 65001) as EnglishProduct; SQL SELECT ROWX, ROWY, ROWZ From Products;`	Podczas ładowania z SQL źródło może zawierać różne zestawy znaków: cyrylicę, języka hebrajskiego itp. – z formatu UTF-8. W przypadku takich zestawów konieczne jest ładowanie wiersz po wierszu z zastosowaniem innej strony kodowej dla każdego wiersza. W **codepage** wartość 1253 reprezentuje zestaw znaków Windows: Grecki, wartość 1255 reprezentuje Hebrajski, a wartość 65001 reprezentuje standardowe znaki UTF-8 języka łacińskiego.

Zob. także: *Zestaw znaków (page 158)*

Date

Funkcja **Date()** formatuje wyrażenie jako datę przy użyciu formatu skonfigurowanego w zmiennych systemowych w skrypcie ładowania danych, systemie operacyjnym lub ewentualnie w formacie ciągu znaków.

Składnia:

```
Date(number[, format])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
number	Liczba do sformatowania.
format	Ciąg znaków opisujący format otrzymywanego ciągu. Jeśli ciąg znaków formatu nie zostanie podany, wówczas będzie używany format daty ustawiony w zmiennych systemowych w skrypcie ładowania danych lub w systemie operacyjnym.

Przykłady i wyniki:
W przypadku poniższych przykładów przyjęto następujące ustawienia domyślne:

- Ustawienie daty 1: YY-MM-DD
- Ustawienie daty 2: M/D/YY

Przykład:

```
Date( A )
```
gdzie A=35648

Tabela wynikowa

Wyniki	Ustawienie 1	Ustawienie 2
Ciąg znaków:	97-08-06	8/6/97
Liczba:	35648	35648

Przykład:

```
Date( A, 'YY.MM.DD' )
```
gdzie A=35648

Tabela wynikowa

Wyniki	Ustawienie 1	Ustawienie 2
Ciąg znaków:	97.08.06	97.08.06
Liczba:	35648	35648

Przykład:

```
Date( A, 'DD.MM.YYYY' )
```
gdzie A=35648.375

Tabela wynikowa

Wyniki	Ustawienie 1	Ustawienie 2
Ciąg znaków:	06.08.1997	06.08.1997
Liczba:	35648.375	35648.375

Przykład:

```
Date( A, 'YY.MM.DD' )
```
gdzie A=8/6/97

Tabela wynikowa

Wyniki	Ustawienie 1	Ustawienie 2
Ciąg znaków:	NULL (nic)	97.08.06
Liczba:	NULL	35648

Dual

Funkcja **Dual()** łączy liczbę i ciąg znaków w jeden rekord, dzięki czemu liczbowa reprezentacja rekordu może zostać użyta na potrzeby sortowania i obliczania (wartość ciągu znaków może natomiast zostać użyta do celów wyświetlania).

Składnia:

```
Dual(text, number)
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
text	Wartość ciągu znaków używana w kombinacji z argumentem liczbowym.
number	Liczba używana w kombinacji z ciągiem w argumencie ciągu.

W aplikacji Qlik Sense wszystkie wartości pola są potencjalnie wartościami podwójnymi. Oznacza to, że wartości pola mogą mieć zarówno wartość liczbową, jak i tekstową. Przykładem może być data, która może mieć wartość liczbową 40908 i reprezentację tekstową '2011-12-31'.

Gdy kilka elementów danych wczytanych do jednego pola ma różne reprezentacje w postaci ciągu, ale tę samą poprawną reprezentację liczbową, wówczas wszystkie one będą mieć tę samą pierwszą napotkaną reprezentację w postaci ciągu.

*Funkcja **dual** jest zwykle używana wcześnie w skrypcie, zanim dane zostaną wczytane do określonego pola, w celu utworzenia pierwszej reprezentacji w postaci znakowej, która będzie pokazana w panelach filtrowania.*

Przykłady i wyniki:

Przykłady skryptów

Przykład	Opis
Dodaj poniższe przykłady do skryptu i uruchom go. `Load dual ( NameDay,NumDay ) as DayOfWeek inline [ NameDay,NumDay Monday,0 Tuesday,1 Wednesday,2 Thursday,3 Friday,4 Saturday,5 Sunday,6 ];`	Pole DayOfWeek może być zastosowane w wizualizacji, na przykład jako wymiar. W tabeli z dniami tygodnia dni są automatycznie sortowane według właściwego numeru kolejnego, nie w porządku alfabetycznym.
`Load Dual('Q' & Ceil (Month(Now())/3), Ceil(Month(Now ())/3)) as Quarter AutoGenerate 1;`	Na tym przykładzie wyszukiwany jest bieżący kwartał. Jest on wyświetlany jako Q1, gdy funkcja **Now()** jest uruchamiana w pierwszych trzech miesiącach roku, Q2 w drugich trzech miesiącach itd. Jeśli jednak pole Quarter jest używane w sortowaniu, będzie zachowywać się jak jego wartość liczbowa: od 1 do 4.
`Dual('Q' & Ceil (Month(Date)/3), Ceil(Month(Date)/3)) as Quarter`	Podobnie jak w poprzednim przykładzie pole Quarter jest tworzone z wartościami tekstowymi od 'Q1' do 'Q4' i są do niego przypisywane wartości liczbowe od 1 do 4. Aby użyć go w skrypcie, należy załadować wartości dla Date.
`Dual(WeekYear(Date) & '-W' & Week(Date), WeekStart(Date)) as YearWeek`	W tym przykładzie tworzone jest pole YearWeek z wartościami tekstowymi formularza '2012-W22', a jednocześnie przypisywana jest wartość liczbowa odpowiadająca numerowi daty pierwszego dnia tygodnia, na przykład: 41057. Aby użyć go w skrypcie, należy załadować wartości dla Date.

Interval

Funkcja **Interval()** formatuje liczbę jako interwał czasowy przy użyciu formatu w zmiennych systemowych w skrypcie ładowania danych, systemie operacyjnym lub formacie ciągu znaków, jeśli został podany.

Interwały mogą być formatowane jako wartości czasu, jako dni lub jako kombinacja dni, godzin, minut, sekund i ułamków sekund.

Składnia:

```
Interval(number[, format])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
number	Liczba do sformatowania.
format	Ciąg znaków opisujący sposób formatowania otrzymywanego ciągu interwału. W przypadku pominięcia użyty jest krótki format daty, format godziny i separator dziesiętny ustawiony w systemie operacyjnym.

Przykłady i wyniki:
W przypadku poniższych przykładów przyjęto następujące ustawienia domyślne:

- Ustawienie formatu daty 1: YY-MM-DD
- Ustawienie formatu daty 2: hh:mm:ss
- Separator dziesiętny liczb: .

Tabela wynikowa

Przykład	Ciąg znaków	Liczba
`Interval( A )` ,gdzie A=0,375	09:00:00	0.375
`Interval( A )` ,gdzie A=1,375	33:00:00	1.375
`Interval( A, 'D hh:mm' )` ,gdzie A=1,375	1 09:00	1.375
`Interval( A-B, 'D hh:mm' )` ,gdzie A=97-08-06 09:00:00 i B=96-08-06 00:00:00	365 09:00	365.375

Money

Funkcja **Money()** formatuje wyrażenie liczbowo jako wartość pieniężną w formacie ustawionym w zmiennych systemowych w skrypcie ładowania danych lub systemie operacyjnym, chyba że podano format ciągu znaków oraz opcjonalny separator dziesiętny i separator tysięcy.

Składnia:

```
Money(number[, format[, dec_sep[, thou_sep]]])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
number	Liczba do sformatowania.
format	Ciąg znaków opisujący sposób formatowania otrzymywanego ciągu waluty.
dec_sep	Ciąg znaków określający separator dziesiętny.
thou_sep	Ciąg znaków określający separator tysięcy.

W przypadku pominięcia argumentów 2-4 zostanie użyty format waluty ustawiony w systemie operacyjnym.

Przykłady i wyniki:
W przypadku poniższych przykładów przyjęto następujące ustawienia domyślne:

- Ustawienie MoneyFormat 1: kr ##0,00, MoneyThousandSep' '
- Ustawienie MoneyFormat 2: $ #,##0.00, MoneyThousandSep','

Przykład:

```
Money( A )
```
,gdzie A=35648

Tabela wynikowa

Wyniki	Ustawienie 1	Ustawienie 2
Ciąg znaków:	kr 35 648,00	$ 35,648.00
Liczba:	35648.00	35648.00

Przykład:

```
Money( A, '#,##0 ¥', '.' , ',' )
```
,gdzie A=3564800

Tabela wynikowa

Wyniki	Ustawienie 1	Ustawienie 2
Ciąg znaków:	3,564,800 ¥	3,564,800 ¥
Liczba:	3564800	3564800

Num

Funkcja **Num()** formatuje liczbę, to znaczy konwertuje wartość liczbową danych wejściowych na wyświetlany tekst przy użyciu formatu określonego w drugim parametrze. Jeśli drugi parametr zostanie pominięty, używa separatorów dziesiętnych i tysięcznych ustawionych w skrypcie ładowania danych. Parametrami opcjonalnymi są niestandardowe symbole separatora dziesiętnego i separatora tysięcy.

Składnia:

```
Num(number[, format[, dec_sep [, thou_sep]]])
```

Typ zwracanych danych: dual

Funkcja Num zwraca wartość podwójną z ciągiem i wartością liczbową. Funkcja pobiera wartość liczbową wyrażenia wejściowego i generuje ciąg reprezentujący liczbę.

Argumenty:

Argumenty

Argument	Opis
number	Liczba do sformatowania.
format	Ciąg określający sposób formatowania otrzymywanego ciągu. W przypadku pominięcia zostaną użyte separatory dziesiętne i tysięcy ustawione w skrypcie ładowania danych.
dec_sep	Ciąg znaków określający separator dziesiętny. W przypadku pominięcia zostanie użyta wartość zmiennej DecimalSep ustawiona w skrypcie ładowania danych.
thou_sep	Ciąg znaków określający separator tysięcy. W przypadku pominięcia zostanie użyta wartość zmiennej ThousandSep ustawiona w skrypcie ładowania danych.

Przykład: Wyrażenie wykresu

Przykład:

Poniższa tabela przedstawia wyniki, gdy wartość pola A równa się 35648.312.

Wyniki

Symbol	Wynik
Num(A)	35648.312 (zależy od zmiennych środowiskowych w skrypcie)
Num(A, '0.0', '.')	35648.3
Num(A, '0,00', ',')	35648,31
Num(A, '#,##0.0', '.', ',')	35,648.3
Num(A, '# ##0', ',', ' ')	35 648

Przykład: Skrypt ładowania

Skrypt ładowania

Funkcja *Num* może być używana w skrypcie ładowania w celu formatowania liczb, nawet jeśli separatory tysięcy i dziesiętne są już ustawione w skrypcie. Poniższy skrypt ładowania zawiera konkretne separatory tysięcy i dziesiętne, a następnie używa funkcji *Num* w celu formatowania danych na inne sposoby.

W **edytorze ładowania danych** utwórz nową sekcję, a następnie dodaj skrypt przykładowy i uruchom go. Następnie dodaj do arkusza w swojej aplikacji co najmniej pola wyszczególnione w kolumnie wyników, aby wyświetlić wynik.

```
SET ThousandSep=','; SET DecimalSep='.'; Transactions: Load *, Num(transaction_amount) as [No
formatting], Num(transaction_amount,'0') as [0], Num(transaction_amount,'#,##0') as [#,##0],
Num(transaction_amount,'# ###,00') as [# ###,00], Num(transaction_amount,'# ###,00',',',' ')
as [# ###,00 , ',' , ' '], Num(transaction_amount,'#,###.00','.',',') as [#,###.00 , '.' ,
','], Num(transaction_amount,'$#,###.00') as [$#,###.00], ; Load * Inline [ transaction_id,
transaction_date, transaction_amount, transaction_quantity, discount, customer_id, size,
color_code 3750, 20180830, 12423.56, 23, 0,2038593, L, Red 3751, 20180907, 5356.31, 6, 0.1,
203521, m, orange 3752, 20180916, 15.75, 1, 0.22, 5646471, S, blue 3753, 20180922, 1251, 7, 0,
3036491, l, Black 3754, 20180922, 21484.21, 1356, 75, 049681, xs, Red 3756, 20180922, -59.18,
2, 0.333333333333333, 2038593, M, Blue 3757, 20180923, 3177.4, 21, .14, 203521, XL, Black ];
```

Tabela aplikacji Qlik Sense pokazująca wyniki różnych zastosowań funkcji *Num* w skrypcie ładowania. Czwarta kolumna tabeli zawiera niepoprawne użycie formatowania – do celów przykładowych.

Brak formatowania	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

Przykład: Skrypt ładowania

Skrypt ładowania

Funkcja *Num* może być używana w skrypcie ładowania w celu sformatowania liczby jako procentu.

W **edytorze ładowania danych** utwórz nową sekcję, a następnie dodaj skrypt przykładowy i uruchom go. Następnie dodaj do arkusza w swojej aplikacji co najmniej pola wyszczególnione w kolumnie wyników, aby wyświetlić wynik.

```
SET ThousandSep=','; SET DecimalSep='.'; Transactions: Load *, Num(discount,'#,##0%') as
[Discount #,##0%] ; Load * Inline [ transaction_id, transaction_date, transaction_amount,
transaction_quantity, discount, customer_id, size, color_code 3750, 20180830, 12423.56, 23,
```

```
0,2038593, L, Red 3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange 3752, 20180916, 15.75, 1,
0.22, 5646471, S, blue 3753, 20180922, 1251, 7, 0, 3036491, l, Black 3754, 20180922, 21484.21,
1356, 75, 049681, xs, Red 3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue 3757,
20180923, 3177.4, 21, .14, 203521, XL, Black ];
```

Tabela aplikacji Qlik Sense pokazująca wyniki funkcji *Num* używanej w skrypcie ładowania w celu formatowania procentów.

Rabat	Discount #,##0%
0.333333333333333	33%
0.22	22%
0	0%
.14	14%
0.1	10%
0	0%
75	7,500%

Time

Funkcja **Time()** formatuje wyrażenie jako wartość czasu przy użyciu formatu czasu ustawionego w zmiennych systemowych w skrypcie ładowania danych lub systemie operacyjnym, chyba że podano format ciągu znaków.

Składnia:

```
Time(number[, format])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
number	Liczba do sformatowania.
format	Ciąg znaków opisujący sposób formatowania otrzymywanego ciągu godziny. W przypadku pominięcia użyty jest krótki format daty, format godziny i separator dziesiętny ustawiony w systemie operacyjnym.

Przykłady i wyniki:
W przypadku poniższych przykładów przyjęto następujące ustawienia domyślne:

- Ustawienie formatu godziny 1: hh:mm:ss
- Ustawienie formatu godziny 2: hh.mm.ss

Przykład:

```
Time( A )
```
,gdzie A=0,375

Tabela wynikowa

Wyniki	Ustawienie 1	Ustawienie 2
Ciąg znaków:	09:00:00	09.00.00
Liczba:	0.375	0.375

Przykład:

```
Time( A )
```
,gdzie A=35648,375

Tabela wynikowa

Wyniki	Ustawienie 1	Ustawienie 2
Ciąg znaków:	09:00:00	09.00.00
Liczba:	35648.375	35648.375

Przykład:

```
Time( A, 'hh-mm' )
```
,gdzie A=0,99999

Tabela wynikowa

Wyniki	Ustawienie 1	Ustawienie 2
Ciąg znaków:	23-59	23-59
Liczba:	0.99999	0.99999

Timestamp

Funkcja **TimeStamp()** formatuje wyrażenie jako wartość daty i godziny przy użyciu formatu znacznika czasu ustawionego w zmiennych systemowych w skrypcie ładowania danych lub systemie operacyjnym, chyba że podano format ciągu znaków.

Składnia:

```
Timestamp(number[, format])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
number	Liczba do sformatowania.
format	Ciąg znaków opisujący sposób formatowania otrzymywanego ciągu znacznika czasu. W przypadku pominięcia użyty jest krótki format daty, format godziny i separator dziesiętny ustawiony w systemie operacyjnym.

Przykłady i wyniki:
W przypadku poniższych przykładów przyjęto następujące ustawienia domyślne:

- Ustawienie TimeStampFormat 1: YY-MM-DD hh:mm:ss
- Ustawienie TimeStampFormat 2: M/D/YY hh:mm:ss

Przykład:

```
Timestamp( A )
```
,gdzie A=35648,375

Tabela wynikowa

Wyniki	Ustawienie 1	Ustawienie 2
Ciąg znaków:	97-08-06 09:00:00	8/6/97 09:00:00
Liczba:	35648.375	35648.375

Przykład:

```
Timestamp( A,'YYYY-MM-DD hh.mm')
```
,gdzie A=35648

Tabela wynikowa

Wyniki	Ustawienie 1	Ustawienie 2
Ciąg znaków:	1997-08-06 00.00	1997-08-06 00.00
Liczba:	35648	35648

5.13 Ogólne funkcje liczbowe

W tych ogólnych funkcjach liczbowych argumenty są wyrażeniami, w których wartość **x** powinna być interpretowana jako liczba rzeczywista. Wszystkie te funkcje mogą być stosowane zarówno w skryptach ładowania danych, jak i wyrażeniach wykresów.

Przegląd ogólnych funkcji liczbowych

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

bitcount

BitCount() zwraca liczbę bitów ustawionych na 1 w reprezentacji dwójkowej podanej liczby. Innymi słowy, funkcja zwraca liczbę ustawionych bitów w argumencie **integer_number**, gdzie wartość **integer_number** jest interpretowana jako 32-bitowa liczba całkowita ze znakiem.

```
BitCount(integer_number)
```

div

Funkcja **Div()** zwraca część całkowitą wyniku dzielenia arytmetycznego pierwszego argumentu przez drugi. Oba parametry są interpretowane jako liczby rzeczywiste, czyli nie muszą być liczbami całkowitymi.

```
Div (integer_number1, integer_number2)
```

fabs

Funkcja **Fabs()** zwraca wartość bezwzględną argumentu **x**. Wynik jest liczbą dodatnią.

```
Fabs (x)
```

fact

Funkcja **Fact()** zwraca silnię dodatniej liczby całkowitej **x**.

```
Fact (x)
```

frac

Funkcja **Frac()** zwraca część ułamkową argumentu **x**.

```
Frac (x)
```

sign

Funkcja **Sign()** zwraca 1, 0 lub -1 zależnie od tego, czy **x** jest liczbą dodatnią, zerem czy liczbą ujemną.

```
Sign (x)
```

Funkcje kombinacji i permutacji

combin

Funkcja **Combin()** zwraca liczbę kombinacji **q** elementów, jakie można wybrać z grupy **p** elementów. Zgodnie z formułą: `Combin(p,q) = p! / q!(p-q)!` Kolejność wybierania elementów nie jest istotna.

```
Combin (p, q)
```

permut

Funkcja **Permut()** zwraca liczbę permutacji **q** elementów, jakie można wybrać z grupy **p** elementów. Zgodnie z formułą: `Permut(p,q) = (p)! / (p - q)!` Kolejność wybierania elementów jest istotna.

```
Permut (p, q)
```

Funkcje modulo

fmod

fmod() to ogólna funkcja modulo, która w przypadku liczb całkowitych zwraca resztę z dzielenia pierwszego argumentu (dzielnej) przez drugi argument (dzielnik). Wynik jest liczbą rzeczywistą. Oba argumenty są interpretowane jako liczby rzeczywiste, czyli nie muszą być liczbami całkowitymi.

```
Fmod (a, b)
```

mod

Mod() to funkcja matematyczna modulo zwracająca nieujemną resztę z dzielenia liczb całkowitych. Pierwszy argument jest dzielną, a drugi dzielnikiem. Oba argumenty muszą być liczbami całkowitymi.

```
Mod (integer_number1, integer_number2)
```

Funkcje parzystości

even

Funkcja **Even()** zwraca True (-1), jeśli argument **integer_number** jest parzystą liczbą całkowitą lub zerem. Funkcja zwraca False (0), jeśli **integer_number** jest nieparzystą liczbą całkowitą, a NULL jeśli **integer_number** nie jest liczbą całkowitą.

```
Even (integer_number)
```

odd

Funkcja **Odd()** zwraca True (-1), jeśli argument **integer_number** jest nieparzystą liczbą całkowitą lub zerem. Funkcja zwraca False (0), jeśli **integer_number** jest parzystą liczbą całkowitą, a NULL jeśli **integer_number** nie jest liczbą całkowitą.

```
Odd (integer_number)
```

Funkcje zaokrąglania

ceil

Funkcja **Ceil()** zaokrągla liczbę w górę do najbliższej wielokrotności wartości **step** z przesunięciem **offset** .

```
Ceil (x[, step[, offset]])
```

floor

Funkcja **Floor()** zaokrągla liczbę w dół do najbliższej wielokrotności wartości **step** z przesunięciem **offset** .

```
Floor (x[, step[, offset]])
```

round

Funkcja **Round()** zwraca wynik zaokrąglania liczby w górę lub w dół do najbliższej wielokrotności **step** przesuniętej o liczbę **offset** .

```
Round ( x [ , step [ , offset ]])
```

BitCount

BitCount() zwraca liczbę bitów ustawionych na 1 w reprezentacji dwójkowej podanej liczby. Innymi słowy, funkcja zwraca liczbę ustawionych bitów w argumencie **integer_number**, gdzie wartość **integer_number** jest interpretowana jako 32-bitowa liczba całkowita ze znakiem.

Składnia:

```
BitCount(integer_number)
```

Typ zwracanych danych: integer

Przykłady i wyniki:

Przykłady i wyniki

Przykłady	Wyniki
`BitCount ( 3 )`	3 w układzie dwójkowym to 11, zwrócona zostanie zatem wartość 2
`BitCount ( -1 )`	-1 w układzie dwójkowym zawiera 64 jedynki, zwrócona zostanie zatem wartość 64.

Ceil

Funkcja **Ceil()** zaokrągla liczbę w górę do najbliższej wielokrotności wartości **step** z przesunięciem **offset** .

Funkcja podobna do funkcji **floor**, która zaokrągla wprowadzane liczby w dół.

Składnia:

```
Ceil(x[, step[, offset]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
x	Liczba wejściowa.
step	Przyrost przedziału. Wartością domyślną jest 1.
offset	Definiuje podstawę przedziału. Wartością domyślną jest 0.

Przykłady i wyniki:

Przykłady i wyniki

Przykłady	Wyniki
`Ceil(2.4 )`	Zwraca wartość 3 W tym przykładzie rozmiar kroku wynosi 1, a podstawą przedziału kroku jest 0. Przedziały są następujące: ...0 < x <=1, 1 < x <= 2, **2< x <=3**, 3< x <=4...
`Ceil(4.2 )`	Zwraca wartość 5
`Ceil(3.88 ,0.1)`	Zwraca wartość 3,9 W tym przykładzie rozmiar przedziału wynosi 0,1, a podstawą przedziału jest 0. Przedziały są następujące: ... 3.7 < x <= 3.8, **3.8 < x <= 3.9**, 3.9 < x <= 4.0...
`Ceil(3.88 ,5)`	Zwraca wartość 5
`Ceil(1.1 ,1)`	Zwraca wartość 2
`Ceil(1.1 ,1,0.5)`	Zwraca wartość 1,5 W tym przykładzie rozmiar kroku wynosi 1, a podstawą przesunięcia jest 0,5. Oznacza to, że podstawą przedziału kroku jest 0,5, a nie 0. Przedziały są następujące: ...**0.5 < x <=1.5**, 1.5 < x <= 2.5, 2.5< x <=3.5, 3.5< x <=4.5...
`Ceil(1.1 ,1,-0.01)`	Zwraca wartość 1,99 Przedziały są następujące: ...-0.01< x <= 0.99, **0.99< x <= 1.99**, 1.99 < x <=2.99...

Combin

Funkcja **Combin()** zwraca liczbę kombinacji **q** elementów, jakie można wybrać z grupy **p** elementów. Zgodnie z formułą: `Combin(p,q) = p! / q!(p-q)!` Kolejność wybierania elementów nie jest istotna.

Składnia:

```
Combin(p, q)
```

Typ zwracanych danych: integer

Ograniczenia:

Elementy niebędące liczbami całkowitymi zostaną obcięte.

Przykłady i wyniki:

Przykłady i wyniki

Przykłady	Wyniki
Ile kombinacji siedmiu liczb można utworzyć z 35 liczb dostępnych w grze liczbowej? `Combin( 35,7 )`	Zwraca wartość 6 724 520

Div

Funkcja **Div()** zwraca część całkowitą wyniku dzielenia arytmetycznego pierwszego argumentu przez drugi. Oba parametry są interpretowane jako liczby rzeczywiste, czyli nie muszą być liczbami całkowitymi.

Składnia:

```
Div(integer_number1, integer_number2)
```

Typ zwracanych danych: integer

Przykłady i wyniki:

Przykłady i wyniki

Przykłady	Wyniki
`Div( 7,2 )`	Zwraca wartość 3
`Div( 7.1,2.3 )`	Zwraca wartość 3
`Div( 9,3 )`	Zwraca wartość 3
`Div( -4,3 )`	Zwraca wartość -1
`Div( 4,-3 )`	Zwraca wartość -1
`Div( -4,-3 )`	Zwraca wartość 1

Even

Funkcja **Even()** zwraca True (-1), jeśli argument **integer_number** jest parzystą liczbą całkowitą lub zerem. Funkcja zwraca False (0), jeśli **integer_number** jest nieparzystą liczbą całkowitą, a NULL jeśli **integer_number** nie jest liczbą całkowitą.

Składnia:

```
Even(integer_number)
```

Typ zwracanych danych: wartość logiczna

Przykłady i wyniki:

Przykłady i wyniki

Przykłady	Wyniki
`Even( 3 )`	Zwraca wartość 0, False
`Even( 2 * 10 )`	Zwraca wartość -1, True
`Even( 3.14 )`	Zwraca wartość NULL

Fabs

Funkcja **Fabs()** zwraca wartość bezwzględną argumentu **x**. Wynik jest liczbą dodatnią.

Składnia:

```
fabs(x)
```

Typ zwracanych danych: numeric

Przykłady i wyniki:

Przykłady i wyniki

Przykłady	Wyniki
`fabs( 2.4 )`	Zwraca wartość 2,4
`fabs( -3.8 )`	Zwraca wartość 3,8

Fact

Funkcja **Fact()** zwraca silnię dodatniej liczby całkowitej **x**.

Składnia:

```
Fact(x)
```

Typ zwracanych danych: integer

Ograniczenia:

Jeśli liczba **x** nie jest liczbą całkowitą, zostanie obcięta. W przypadku liczb niedodatnich zwrócona zostanie wartość NULL.

Przykłady i wyniki:

Przykłady i wyniki

Przykłady	Wyniki
`Fact( 1 )`	Zwraca wartość 1
`Fact( 5 )`	Zwraca wartość 120 (1 * 2 * 3 * 4 * 5 = 120)
`Fact( -5 )`	Zwraca wartość NULL

Floor

Funkcja **Floor()** zaokrągla liczbę w dół do najbliższej wielokrotności wartości **step** z przesunięciem **offset** .

Funkcja podobna do funkcji **ceil**, która zaokrągla wprowadzane liczby w górę.

Składnia:

```
Floor(x[, step[, offset]])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
x	Liczba wejściowa.
step	Przyrost przedziału. Wartością domyślną jest 1.
offset	Definiuje podstawę przedziału. Wartością domyślną jest 0.

Przykłady i wyniki:

Przykłady i wyniki

Przykłady	Wyniki
`Floor(2.4)`	Zwraca wartość 2 In this example, the size of the step is 1 and the base of the step interval is 0. The intervals are ...0 <= x <1, 1 <= x < 2, **2<= x <3**, 3<= x <4....
`Floor(4.2)`	Zwraca wartość 4

Przykłady	Wyniki
`Floor(3.88 ,0.1)`	Zwraca wartość 3,8 W tym przykładzie rozmiar przedziału wynosi 0,1, a podstawą przedziału jest 0. Przedziały są następujące: ... 3.7 <= x < 3.8, **3.8 <= x < 3.9**, 3.9 <= x < 4.0...
`Floor(3.88 ,5)`	Zwraca 0
`Floor(1.1 ,1)`	Zwraca wartość 1
`Floor(1.1 ,1,0.5)`	Zwraca wartość 0,5 W tym przykładzie rozmiar kroku wynosi 1, a podstawą przesunięcia jest 0,5. Oznacza to, że podstawą przedziału kroku jest 0,5, a nie 0. Przedziały są następujące: ...**0.5 <= x <1.5**, 1.5 <= x < 2.5, 2.5<= x <3.5,...

Fmod

fmod() to ogólna funkcja modulo, która w przypadku liczb całkowitych zwraca resztę z dzielenia pierwszego argumentu (dzielnej) przez drugi argument (dzielnik). Wynik jest liczbą rzeczywistą. Oba argumenty są interpretowane jako liczby rzeczywiste, czyli nie muszą być liczbami całkowitymi.

Składnia:

```
fmod(a, b)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
a	Dzielna
b	Dzielnik

Przykłady i wyniki:

Przykłady i wyniki

Przykłady	Wyniki
`fmod( 7,2 )`	Zwraca wartość 1
`fmod( 7.5,2 )`	Zwraca wartość 1,5
`fmod( 9,3 )`	Zwraca wartość 0
`fmod( -4,3 )`	Zwraca wartość -1

Przykłady	Wyniki
`fmod( 4,-3 )`	Zwraca wartość 1
`fmod( -4,-3 )`	Zwraca wartość -1

Frac

Funkcja **Frac()** zwraca część ułamkową argumentu **x**.

Ułamek jest definiowany w następujący sposób: `Frac(x ) + Floor(x ) = x`. Oznacza to, że część ułamkowa liczby dodatniej to różnica między liczbą (x) a liczbą całkowitą poprzedzającą część ułamkową.

Na przykład: część ułamkowa liczby 11,43 to 11,43 - 11 = 0,43

W przypadku liczb ujemnych, na przykład -1.4, Floor(-1.4) = -2, co daje następujący wynik:

Część ułamkowa liczby -1,4 to -1,4 - (-2) = -1,4 + 2 = 0,6

Składnia:

```
Frac(x)
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
x	Liczba, której ułamek jest zwracany.

Przykłady i wyniki:

Przykłady i wyniki

Przykłady	Wyniki
`Frac( 11.43 )`	Zwraca wartość 0,43
`Frac( -1.4 )`	Zwraca wartość 0,6
Wyodrębnij składnik godziny z liczbowej reprezentacji znacznika czasu, pomijając w ten sposób datę. `Time(Frac(44518.663888889))`	Zwraca 3:56:00 PM

Mod

Mod() to funkcja matematyczna modulo zwracająca nieujemną resztę z dzielenia liczb całkowitych. Pierwszy argument jest dzielną, a drugi dzielnikiem. Oba argumenty muszą być liczbami całkowitymi.

Składnia:

```
Mod(integer_number1, integer_number2)
```

Typ zwracanych danych: integer

Ograniczenia:

Argument **integer_number2** musi być większy od 0.

Przykłady i wyniki:

Przykłady i wyniki

Przykłady	Wyniki
`Mod( 7,2 )`	Zwraca wartość 1
`Mod( 7.5,2 )`	Zwraca wartość NULL
`Mod( 9,3 )`	Zwraca wartość 0
`Mod( -4,3 )`	Zwraca wartość 2
`Mod( 4,-3 )`	Zwraca wartość NULL
`Mod( -4,-3 )`	Zwraca wartość NULL

Odd

Funkcja **Odd()** zwraca True (-1), jeśli argument **integer_number** jest nieparzystą liczbą całkowitą lub zerem. Funkcja zwraca False (0), jeśli **integer_number** jest parzystą liczbą całkowitą, a NULL jeśli **integer_number** nie jest liczbą całkowitą.

Składnia:

```
Odd(integer_number)
```

Typ zwracanych danych: wartość logiczna

Przykłady i wyniki:

Przykłady i wyniki

Przykłady	Wyniki
`Odd( 3 )`	Zwraca wartość -1, True
`Odd( 2 * 10 )`	Zwraca wartość 0, False
`Odd( 3.14 )`	Zwraca wartość NULL

Permut

Funkcja **Permut()** zwraca liczbę permutacji **q** elementów, jakie można wybrać z grupy **p** elementów. Zgodnie z formułą: `Permut(p,q) = (p)! / (p - q)!` Kolejność wybierania elementów jest istotna.

Składnia:

```
Permut(p, q)
```

Typ zwracanych danych: integer

Ograniczenia:

Argumenty niebędące liczbami całkowitymi zostaną obcięte.

Przykłady i wyniki:

Przykłady i wyniki

Przykłady	Wyniki
Na ile sposobów można rozdzielić medale złoty, srebrny i brązowy w finale na 100 metrów między ośmiu uczestników? `Permut( 8,3 )`	Zwraca wartość 336

Round

Funkcja **Round()** zwraca wynik zaokrąglania liczby w górę lub w dół do najbliższej wielokrotności **step** przesuniętej o liczbę **offset** .

Jeśli wartość do zaokrąglenia przypada dokładnie w połowie przedziału, jest zaokrąglana w górę.

Składnia:

```
Round(x[, step[, offset]])
```

Typ zwracanych danych: numeric

W przypadku zaokrąglania liczb zmiennopozycyjnych mogą pojawiać się błędne wyniki. Te błędy zaokrąglania wynikają z tego, że liczby zmiennopozycyjne są reprezentowane przez skończoną liczbę cyfr dwójkowych (bitów). W efekcie wyniki są obliczane z użyciem liczby, która jest już zaokrąglona. W efekcie wyniki są obliczane z użyciem liczby, która jest już zaokrąglona. Jeśli te błędy zaokrąglania wpływają na przebieg pracy, należy pomnożyć liczby, aby przed zaokrąglaniem przekształcić je na liczby całkowite.

Argumenty:

Argumenty

Argument	Opis
x	Liczba wejściowa.
step	Przyrost przedziału. Wartością domyślną jest 1.
offset	Definiuje podstawę przedziału. Wartością domyślną jest 0.

Przykłady i wyniki:

Przykłady i wyniki

Przykłady	Wyniki
`Round(3.8 )`	Zwraca wartość 4 W tym przykładzie rozmiar kroku wynosi 1, a podstawą przedziału kroku jest 0. Przedziały są następujące: ...0 <= x <1, 1 <= x < 2, 2<= x <3, **3<= x <4**...
`Round(3.8,4 )`	Zwraca wartość 4
`Round(2.5 )`	Zwraca wartość 3 W tym przykładzie rozmiar kroku wynosi 1, a podstawą przedziału kroku jest 0. Przedziały są następujące: ...0 <= x <1, 1 <= x <2, **2<= x <3**...
`Round(2,4 )`	Zwraca wartość 4. Zaokrąglone w górę, ponieważ wartość 2 przypada dokładnie w połowie odstępu wynoszącego 4. W tym przykładzie rozmiar kroku wynosi 4, a podstawą przedziału kroku jest 0. Przedziały są następujące: ...**0 <= x <4**, 4 <= x <8, 8<= x <12...
`Round(2,6 )`	Zwraca wartość 0. Zaokrąglone w dół, ponieważ wartość 2 jest mniejsza niż wartość połowy odstępu wynoszącego 6. W tym przykładzie rozmiar kroku wynosi 6, a podstawą przedziału kroku jest 0. Przedziały są następujące: ...**0 <= x <6**, 6 <= x <12, 12<= x <18...
`Round(3.88 ,0.1)`	Zwraca wartość 3,9 W tym przykładzie rozmiar kroku wynosi 0,1, a podstawą przedziału kroku jest 0. Przedziały są następujące: ... 3.7 <= x <3.8, **3.8 <= x <3.9**, 3.9 <= x < 4.0...

Przykłady	Wyniki
`Round (3.88875,1/1000)`	Zwraca wartość 3,889 W tym przykładzie rozmiar kroku wynosi 0,001, co zaokrągla liczbę i ogranicza ją do trzech miejsc po przecinku.
`Round(3.88 ,5)`	Zwraca wartość 5
`Round(1.1 ,1,0.5)`	Zwraca wartość 1,5 W tym przykładzie rozmiar kroku wynosi 1, a podstawą przedziału kroku jest 0,5. Przedziały są następujące: ...**0.5 <= x <1.5**, 1.5 <= x <2.5, 2.5<= x <3.5...

Sign

Funkcja **Sign()** zwraca 1, 0 lub -1 zależnie od tego, czy **x** jest liczbą dodatnią, zerem czy liczbą ujemną.

Składnia:

```
Sign(x)
```

Typ zwracanych danych: numeric

Ograniczenia:

Jeśli nie zostanie znaleziona żadna wartość liczbowa, zostanie zwrócona wartość NULL.

Przykłady i wyniki:

Przykłady i wyniki

Przykłady	Wyniki
`Sign( 66 )`	Zwraca wartość 1
`Sign( 0 )`	Zwraca wartość 0
`Sign( - 234 )`	Zwraca wartość -1

5.14 Funkcje geoprzestrzenne

Te funkcje są używane do obsługi danych geoprzestrzennych na wizualizacjach map. Program Qlik Sense jest zgodny ze specyfikacjami GeoJSON dotyczącymi danych geoprzestrzennych i obsługuje następujące kolekcje:

- Point
- Linestring
- Polygon
- Multipolygon

Więcej informacji na temat specyfikacji GeoJSON można znaleźć w:

≤ GeoJSON.org

Przegląd funkcji geoprzestrzennych

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

Istnieją dwie kategorie funkcji geoprzestrzennych: agregacyjne i nieagregacyjne.

Funkcje agregacyjne przyjmują na wejściu zestaw geometrii (punkty lub obszary) i zwracają pojedynczą geometrię. Na przykład wiele obszarów może zostać połączonych ze sobą, a na mapie może zostać wyświetlona jedna granica dla agregacji.

Funkcje nieagregacyjne przyjmują na wejściu jedną geometrię i zwracają jedną. Jeśli na przykład dla funkcji GeoGetPolygonCenter() na wejściu ustawiona jest geometria granicy jednego z obszarów, wówczas zwracana geometria punktu (długość geograficzna i szerokość geograficzna) dla środka tego obszaru.

Następujące funkcje są agregacyjne:

GeoAggrGeometry

Funkcja **GeoAggrGeometry()** służy do agregacji obszarów w jeden większy obszar, na przykład w celu agregacji kilku podregionów w jeden region.

```
GeoAggrGeometry (field_name)
```

GeoBoundingBox

Funkcja **GeoBoundingBox()** służy do agregacji geometrii do obszaru oraz do obliczenia najmniejszego pola obwiedni, które zawiera wszystkie współrzędne.

```
GeoBoundingBox (field_name)
```

GeoCountVertex

Funkcja **GeoCountVertex()** służy do znalezienia liczby wierzchołków wielokąta.

```
GeoCountVertex(field_name)
```

GeoInvProjectGeometry

Funkcja **GeoInvProjectGeometry()** służy do agregacji geometrii do obszaru oraz do zastosowania odwrotności projekcji.

```
GeoInvProjectGeometry(type, field_name)
```

GeoProjectGeometry

Funkcja **GeoProjectGeometry()** służy do agregacji geometrii do obszaru oraz do stosowania projekcji.

```
GeoProjectGeometry(type, field_name)
```

GeoReduceGeometry

Funkcja **GeoReduceGeometry()** służy do zmniejszenia liczby wierzchołków figury geometrycznej oraz do agregacji obszarów w jeden obszar, przy czym nadal wyświetlane są linie graniczne poszczególnych obszarów.

```
GeoReduceGeometry (geometry)
```

Następujące funkcje są nieagregacyjne:

GeoGetBoundingBox

Funkcja **GeoGetBoundingBox()** jest używana w skryptach i wyrażeniach wykresów w celu obliczenia najmniejszego geoprzestrzennego pola obwiedni, które zawiera wszystkie współrzędne konkretnej geometrii.

```
GeoGetBoundingBox (geometry)
```

GeoGetPolygonCenter

Funkcja **GeoGetPolygonCenter()** jest używana w skryptach i wyrażeniach wykresu w celu obliczania i zwracania punktu środka geometrii.

```
GeoGetPolygonCenter (geometry)
```

GeoMakePoint

Funkcja **GeoMakePoint()** jest używana w skryptach i wyrażeniach wykresu w celu utworzenia punktu i oznaczania go szerokością i długością geograficzną.

```
GeoMakePoint (lat_field_name, lon_field_name)
```

GeoProject

Funkcja **GeoProject()** jest używana w skryptach i wyrażeniach wykresu w celu stosowania projekcji względem geometrii.

```
GeoProject (type, field_name)
```

GeoAggrGeometry

Funkcja **GeoAggrGeometry()** służy do agregacji obszarów w jeden większy obszar, na przykład w celu agregacji kilku podregionów w jeden region.

Składnia:

```
GeoAggrGeometry(field_name)
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
field_name	Pole lub wyrażenie odwołujące się do pola zawierającego geometrię przeznaczoną do reprezentowania. Może to być punkt (albo zestaw punktów) charakteryzujący się długością i szerokością geograficzną, albo obszar.

Funkcja **GeoAggrGeometry()** może zwykle być stosowana w celu łączenia danych granic geoprzestrzennych. Na przykład mogą istnieć obszary przypisane do kodów pocztowych dla dzielnic podmiejskich w mieście oraz przychody ze sprzedaży dla poszczególnych obszarów. Jeśli terytorium sprzedawcy obejmuje kilka obszarów kodów pocztowych, użyteczne może być przedstawienie sprzedaży łącznej według terytoriów sprzedaży, a nie dla poszczególnych obszarów, a następnie pokazanie wyników na mapie z pokolorowanymi polami.

Za pomocą funkcji **GeoAggrGeometry()** można obliczyć agregację geometrii poszczególnych dzielnic i wygenerować geometrię scalonego terytorium w modelu danych. Jeśli następnie granice terytorium zostaną zmienione, wówczas do danych zostaną przeładowane nowe scalone granice, a przychody zostaną odzwierciedlone na mapie.

GeoAggrGeometry() jest funkcją agregacji, dlatego użycie jej w skrypcie powoduje konieczność użycia instrukcji **LOAD** z klauzulą **Group by**.

*Za pomocą linii granic na mapach utworzonych z użyciem funkcji **GeoAggrGeometry()** przedstawione są linie obszarów po scaleniu. Jeśli wymagane jest wyświetlenie poszczególnych granic obszarów przed agregacją, należy użyć funkcji **GeoReduceGeometry ()**.*

Przykłady:
W tym przykładzie ładowany jest plik KML z danymi obszaru, a następnie tabela ze zagregowanymi danymi obszaru.

```
[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

Funkcja **GeoBoundingBox()** służy do agregacji geometrii do obszaru oraz do obliczenia najmniejszego pola obwiedni, które zawiera wszystkie współrzędne.

Pole GeoBoundingBox jest reprezentowane przez listę czterech wartości: lewa strona, prawa strona, góra, dół.

Składnia:

```
GeoBoundingBox(field_name)
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
field_name	Pole lub wyrażenie odwołujące się do pola zawierającego geometrię przeznaczoną do reprezentowania. Może to być punkt (albo zestaw punktów) charakteryzujący się długością i szerokością geograficzną, albo obszar.

Funkcja GeoBoundingBox() agreguje zestaw geometrii i zwraca cztery współrzędne dla najmniejszego prostokąta, który zawiera wszystkie współrzędne zagregowanej geometrii.

W celu wizualizacji wyniku mapy należy przenieść wynikowy ciąg znaków czterech współrzędnych do formatu wielokąta, oznakować przeniesione pole znacznikiem formatu geopolygon, a następnie przeciągnąć i upuścić to pole do obiektu mapy. Prostokątne pola zostaną wówczas wyświetlone w wizualizacji mapy.

GeoCountVertex

Funkcja **GeoCountVertex()** służy do znalezienia liczby wierzchołków wielokąta.

Składnia:

```
GeoCountVertex(field_name)
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
field_name	Pole lub wyrażenie odwołujące się do pola zawierającego geometrię przeznaczoną do reprezentowania. Może to być punkt (albo zestaw punktów) charakteryzujący się długością i szerokością geograficzną, albo obszar.

GeoGetBoundingBox

Funkcja **GeoGetBoundingBox()** jest używana w skryptach i wyrażeniach wykresów w celu obliczenia najmniejszego geoprzestrzennego pola obwiedni, które zawiera wszystkie współrzędne konkretnej geometrii.

Geoprzestrzenne pole obwiedni utworzone za pomocą funkcji GeoBoundingBox() jest reprezentowane przez listę czterech wartości: lewa strona, prawa strona, góra, dół.

Składnia:

```
GeoGetBoundingBox(field_name)
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
field_name	Pole lub wyrażenie odwołujące się do pola zawierającego geometrię przeznaczoną do reprezentowania. Może to być punkt (albo zestaw punktów) charakteryzujący się długością i szerokością geograficzną, albo obszar.

*W edytorze ładowania danych nie należy używać klauzuli **Group by** z tą funkcją ani z innymi funkcjami geoprzestrzennymi, które nie dokonują agregacji, ponieważ spowoduje to błąd podczas ładowania.*

GeoGetPolygonCenter

Funkcja **GeoGetPolygonCenter()** jest używana w skryptach i wyrażeniach wykresu w celu obliczania i zwracania punktu środka geometrii.

W niektórych sytuacjach konieczne jest umieszczenie na mapie kropki zamiast wypełnienia kolorem. Jeśli istniejące dane geoprzestrzenne są dostępne tylko w postaci geometrii obszaru (na przykład granica), wówczas należy użyć funkcji **GeoGetPolygonCenter()**, aby uzyskać parę współrzędnych (długość geograficzna i szerokość geograficzna) dla środka obszaru.

Składnia:

```
GeoGetPolygonCenter(field_name)
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
field_name	Pole lub wyrażenie odwołujące się do pola zawierającego geometrię przeznaczoną do reprezentowania. Może to być punkt (albo zestaw punktów) charakteryzujący się długością i szerokością geograficzną, albo obszar.

*W edytorze ładowania danych nie należy używać klauzuli **Group by** z tą funkcją ani z innymi funkcjami geoprzestrzennymi, które nie dokonują agregacji, ponieważ spowoduje to błąd podczas ładowania.*

GeoInvProjectGeometry

Funkcja **GeoInvProjectGeometry()** służy do agregacji geometrii do obszaru oraz do zastosowania odwrotności projekcji.

Składnia:

```
GeoInvProjectGeometry(type, field_name)
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
type	Typ rzutowania używany podczas transformacji geometrii mapy. Może przyjmować jedną z dwóch wartości: „unit”, (domyślnie), co powoduje rzutowanie 1:1, lub „mercator” – w tym przypadku używane jest standardowe rzutowanie Merkatora.
field_name	Pole lub wyrażenie odwołujące się do pola zawierającego geometrię przeznaczoną do reprezentowania. Może to być punkt (albo zestaw punktów) charakteryzujący się długością i szerokością geograficzną, albo obszar.

Przykład:

Przykład skryptu

Przykład	Wynik
W instrukcji `Load`: `GeoInvProjectGeometry ('mercator',AreaPolygon) as InvProjectGeometry`	Geometria załadowana jako **AreaPolygon** jest przekształcana przy użyciu odwróconej transformacji rzutowania Merkatora i zachowywana jako **InvProjectGeometry** z przeznaczeniem do użycia w wizualizacjach.

GeoMakePoint

Funkcja **GeoMakePoint()** jest używana w skryptach i wyrażeniach wykresu w celu utworzenia punktu i oznaczania go szerokością i długością geograficzną. GeoMakePoint zwraca punkty w kolejności według długości geograficznej i szerokości geograficznej.

Składnia:

```
GeoMakePoint(lat_field_name, lon_field_name)
```

Typ zwracanych danych: ciąg, sformatowany [długość geograficzna, szerokość geograficzna]

Argumenty:

Argumenty

Argument	Opis
lat_field_name	Pole lub wyrażenie odwołujące się do pola reprezentującego szerokość geograficzną punktu.
lon_field_name	Pole lub wyrażenie odwołujące się do pola reprezentującego długość geograficzną punktu.

*W edytorze ładowania danych nie należy używać klauzuli **Group by** z tą funkcją ani z innymi funkcjami geoprzestrzennymi, które nie dokonują agregacji, ponieważ spowoduje to błąd podczas ładowania.*

GeoProject

Funkcja **GeoProject()** jest używana w skryptach i wyrażeniach wykresu w celu stosowania projekcji względem geometrii.

Składnia:

```
GeoProject(type, field_name)
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
type	Typ rzutowania używany podczas transformacji geometrii mapy. Może przyjmować jedną z dwóch wartości: „unit”, (domyślnie), co powoduje rzutowanie 1:1, lub „mercator” – w tym przypadku używane jest webowe rzutowanie Merkatora.
field_name	Pole lub wyrażenie odwołujące się do pola zawierającego geometrię przeznaczoną do reprezentowania. Może to być punkt (albo zestaw punktów) charakteryzujący się długością i szerokością geograficzną, albo obszar.

*W edytorze ładowania danych nie należy używać klauzuli **Group by** z tą funkcją ani z innymi funkcjami geoprzestrzennymi, które nie dokonują agregacji, ponieważ spowoduje to błąd podczas ładowania.*

Przykład:

Przykłady skryptów

Przykład	Wynik
W instrukcji `Load`: `GeoProject ('mercator',Area) as GetProject`	Rzutowanie Merkatora jest stosowane względem geometrii załadowanej jako **Area**, a wynik jest zapisywany jako **GetProject**.

GeoProjectGeometry

Funkcja **GeoProjectGeometry()** służy do agregacji geometrii do obszaru oraz do stosowania projekcji.

Składnia:

```
GeoProjectGeometry(type, field_name)
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
type	Typ rzutowania używany podczas transformacji geometrii mapy. Może przyjmować jedną z dwóch wartości: „unit”, (domyślnie), co powoduje rzutowanie 1:1, lub „mercator” – w tym przypadku używane jest webowe rzutowanie Merkatora.
field_name	Pole lub wyrażenie odwołujące się do pola zawierającego geometrię przeznaczoną do reprezentowania. Może to być punkt (albo zestaw punktów) charakteryzujący się długością i szerokością geograficzną, albo obszar.

Przykład:

Przykład	Wynik
W instrukcji `Load`: `GeoProjectGeometry ('mercator',AreaPolygon) as ProjectGeometry`	Geometria załadowana jako **AreaPolygon** jest przekształcana przy użyciu rzutowania Merkatora i zachowywana jako **ProjectGeometry** z przeznaczeniem do użycia w wizualizacjach.

GeoReduceGeometry

Funkcja **GeoReduceGeometry()** służy do zmniejszenia liczby wierzchołków figury geometrycznej oraz do agregacji obszarów w jeden obszar, przy czym nadal wyświetlane są linie graniczne poszczególnych obszarów.

Składnia:

```
GeoReduceGeometry(field_name[, value])
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
field_name	Pole lub wyrażenie odwołujące się do pola zawierającego geometrię przeznaczoną do reprezentowania. Może to być punkt (albo zestaw punktów) charakteryzujący się długością i szerokością geograficzną, albo obszar.
value	Stopień redukcji do zastosowania względem geometrii. Zakres wynosi od 0 do 1, przy czym 0 reprezentuje brak redukcji, a 1 reprezentuje maksymalną redukcję wierzchołków.  *Użycie wartości (value) równej 0,9 lub wyższej ze złożonym zestawem danych może spowodować zmniejszenie liczby wierzchołków do poziomu, w którym reprezentacja wizualna będzie niedokładna.*

Działanie funkcji **GeoReduceGeometry()** jest podobne do działania funkcji **GeoAggrGeometry()**, ponieważ za jej pomocą można zagregować szereg obszarów w jeden obszar. Różnica polega na tym, że w przypadku użycia funkcji **GeoReduceGeometry()** poszczególne linie graniczne z danych sprzed agregacji są wyświetlane na mapie.

GeoReduceGeometry() jest funkcją agregacji, dlatego użycie jej w skrypcie powoduje konieczność użycia instrukcji **LOAD** z klauzulą **Group by**.

Przykłady:
W tym przykładzie ładowany jest plik KML z danymi obszaru, a następnie tabela ze zredukowanymi i zagregowanymi danymi obszaru.

```
[MapSource]: LOAD [world.Name], [world.Point], [world.Area] FROM [lib://Downloads/world.kml]
(kml, Table is [World.shp/Features]); Map: LOAD world.Name, GeoReduceGeometry(world.Area,0.5)
as [ReducedArea] resident MapSource Group By world.Name;

 Drop Table MapSource;
```

5.15 Funkcje interpretacji

Funkcje interpretacji oceniają zawartość wejściowych pól tekstowych lub wyrażeń oraz narzucają określony format danych otrzymywanej wartości liczbowej. Przy użyciu tych funkcji można określić format liczby zgodnie z jego typem danych, w tym takie atrybuty, jak separator dziesiętny, separator tysięcy i format daty.

Wszystkie funkcje interpretacji zwracają wartość podwójną zawierającą zarówno ciąg znaków, jak i wartość liczbową, ale można je rozpatrywać jako przekształcenie ciągu znaków na liczbę. Funkcje pobierają wartość tekstową wyrażenia wejściowego i generują liczbę reprezentującą ciąg znaków.

Natomiast funkcje formatowania przeprowadzają odwrotną operację: pobierają wyrażenia liczbowe i oceniają je jako ciągi znaków, określając format wyświetlania uzyskiwanego tekstu.

Jeśli nie są podane żadne funkcje interpretacji, aplikacja Qlik Sense interpretuje dane jako kombinację liczb, dat, godzin, znaczników czasu i ciągów znaków, wykorzystując domyślne ustawienie formatów liczb, daty i godziny zdefiniowane w zmiennych skryptu i ustawieniach systemu operacyjnego.

Wszystkie funkcje interpretacji mogą być stosowane zarówno w skryptach ładowania danych, jak i wyrażeniach wykresu.

Wszystkie reprezentacje liczb są podawane z kropką jako separatorem dziesiętnym.

Przegląd funkcji interpretacji

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

Date#

Date# ocenia wyrażenie jako datę w formacie określonym w ewentualnym drugim argumencie. W przypadku pominięcia argumentu format-code zostanie użyty domyślny format daty ustawiony w systemie operacyjnym.

```
Date# (page 1243)(text[, format])
```

Interval#

Funkcja **Interval#()** ocenia wyrażenie tekstowe jako interwał czasowy w formacie ustawionym domyślnie w systemie operacyjnym albo w formacie określonym w ewentualnym drugim argumencie.

```
Interval# (page 1244)(text[, format])
```

Money#

Funkcja **Money#()** konwertuje ciąg znaków na wartość pieniężną w formacie ustawionym w skrypcie ładowania lub systemie operacyjnym, chyba że podano format ciągu znaków. Parametrami opcjonalnymi są niestandardowe symbole separatora dziesiętnego i separatora tysięcy.

```
Money# (page 1245)(text[, format[, dec_sep[, thou_sep ] ] ])
```

Num#

Funkcja **Num#()** interpretuje ciąg tekstowy jako wartość liczbową, to znaczy konwertuje ciąg wejściowy na liczbę przy użyciu formatu określonego w drugim parametrze. Jeśli drugi parametr zostanie pominięty, używa separatorów dziesiętnych i tysięcznych ustawionych w skrypcie ładowania danych. Parametrami opcjonalnymi są niestandardowe symbole separatora dziesiętnego i separatora tysięcy.

```
Num# (page 1246)(text[ , format[, dec_sep[ , thou_sep]]])
```

Text

Funkcja **Text()** wymusza traktowanie wyrażenia jako tekstu, nawet jeśli możliwa jest jego interpretacja liczbowa.

```
Text(expr)
```

Time#

Funkcja **Time#()** ocenia wyrażenie jako wartość czasu w formacie czasu ustawionym w skrypcie ładowania danych lub systemie operacyjnym, chyba że podano format ciągu znaków..

```
Time# (page 1247)(text[, format])
```

Timestamp#

Funkcja **Timestamp#()** ocenia wyrażenie jako wartość daty i czasu w formacie znacznika czasu ustawionym w skrypcie ładowania danych lub systemie operacyjnym, chyba że podano format ciągu znaków.

```
Timestamp# (page 1248)(text[, format])
```

Zob. także:

p *Funkcje formatowania (page 1206)*

Date#

Date# ocenia wyrażenie jako datę w formacie określonym w ewentualnym drugim argumencie.

Składnia:

```
Date#(text[, format])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
text	Tekstowy ciąg znaków do oceny.
format	Ciąg znaków opisujący format przetwarzanego ciągu tekstowego. Jeśli zostanie pominięty, wówczas będzie używany format daty ustawiony w zmiennych systemowych w skrypcie ładowania danych lub w systemie operacyjnym.

Przykłady i wyniki:
W poniższym przykładzie użyto formatu daty **M/D/YYYY**. Format daty jest określony w instrukcji **SET DateFormat** u góry skryptu ładowania danych.

Dodaj ten przykładowy skrypt do aplikacji i uruchom ją.

```
Load *,
Num(Date#(StringDate)) as Date;
LOAD * INLINE [
StringDate
8/7/97
8/6/1997
]
```

W przypadku utworzenia tabeli z wymiarami **StringDate**i **Date** otrzymane wyniki będą następujące:

Wyniki

StringDate	Data
8/7/97	35649
8/6/1997	35648

Interval#

Funkcja **Interval#()** ocenia wyrażenie tekstowe jako interwał czasowy w formacie ustawionym domyślnie w systemie operacyjnym albo w formacie określonym w ewentualnym drugim argumencie.

Składnia:

```
Interval#(text[, format])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
text	Tekstowy ciąg znaków do oceny.
format	Ciąg znaków opisujący spodziewany format danych wejściowych, który ma być użyty podczas konwertowania tego ciągu na interwał liczbowy. W przypadku pominięcia użyty jest krótki format daty, format godziny i separator dziesiętny ustawiony w systemie operacyjnym.

Funkcja **interval#** konwertuje tekstowy interwał czasowy na jego ekwiwalent liczbowy.

Przykłady i wyniki:
W przypadku poniższych przykładów przyjęto następujące ustawienia systemu operacyjnego:

- Krótki format daty: YY-MM-DD
- Format godziny: M/D/YY
- Separator dziesiętny liczb: .

Wyniki

Przykład	Wynik
`Interval#( A, 'D hh:mm' )` gdzie A='1 09:00'	1.375

Money#

Funkcja **Money#()** konwertuje ciąg znaków na wartość pieniężną w formacie ustawionym w skrypcie ładowania lub systemie operacyjnym, chyba że podano format ciągu znaków. Parametrami opcjonalnymi są niestandardowe symbole separatora dziesiętnego i separatora tysięcy.

Składnia:

```
Money#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
text	Tekstowy ciąg znaków do oceny.
format	Ciąg znaków opisujący spodziewany format danych wejściowych, który ma być użyty podczas konwertowania tego ciągu na interwał liczbowy. W przypadku pominięcia zostanie użyty format waluty ustawiony w systemie operacyjnym.
dec_sep	Ciąg znaków określający separator dziesiętny. W przypadku pominięcia zostanie użyta wartość MoneyDecimalSep ustawiona w skrypcie ładowania danych.
thou_sep	Ciąg znaków określający separator tysięcy. W przypadku pominięcia zostanie użyta wartość MoneyThousandSep ustawiona w skrypcie ładowania danych.

Funkcja **money#** zazwyczaj działa analogicznie do funkcji **num#**, ale pobierając domyślne wartości separatora dziesiętnego i separatora tysięcy ze zmiennych skryptu określających format lub ustawień systemowych waluty.

Przykłady i wyniki:
W przypadku poniższych przykładów przyjęto założenie, że wybrane są dwa następujące ustawienia systemu operacyjnego:

- Domyślne ustawienie formatu waluty 1: kr # ##0,00
- Domyślne ustawienie formatu waluty 2: $ #,##0.00

```
Money#(A , '# ##0,00 kr' )
```
,gdzie A=35 648.37 kr

Wyniki

Wyniki	Ustawienie 1	Ustawienie 2
Ciąg znaków	35 648.37 kr	35 648.37 kr
Liczba	35648.37	3564837

```
Money#( A, ' $#', '.', ',' )
```
,gdzie A= $35 648.37

Wyniki

Wyniki	Ustawienie 1	Ustawienie 2
Ciąg znaków	$35,648.37	$35,648.37
Liczba	35648.37	35648.37

Num#

Funkcja **Num#()** interpretuje ciąg tekstowy jako wartość liczbową, to znaczy konwertuje ciąg wejściowy na liczbę przy użyciu formatu określonego w drugim parametrze. Jeśli drugi parametr zostanie pominięty, używa separatorów dziesiętnych i tysięcznych ustawionych w skrypcie ładowania danych. Parametrami opcjonalnymi są niestandardowe symbole separatora dziesiętnego i separatora tysięcy.

Składnia:

```
Num#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Typ zwracanych danych: dual

Funkcja **Num#()** zwraca wartość podwójną z ciągiem i wartością liczbową. Funkcja pobiera reprezentację tekstową wyrażenia wejściowego i generuje liczbę. Nie zmienia formatu liczby: dane wyjściowe są sformatowane tak samo jak dane wejściowe.

Argumenty:

Argumenty

Argument	Opis
text	Tekstowy ciąg znaków do oceny.
format	Ciąg określający format liczb używany w pierwszym parametrze. W przypadku pominięcia zostaną użyte separatory dziesiętne i tysięcy ustawione w skrypcie ładowania danych.
dec_sep	Ciąg znaków określający separator dziesiętny. W przypadku pominięcia zostanie użyta wartość zmiennej DecimalSep ustawiona w skrypcie ładowania danych.
thou_sep	Ciąg znaków określający separator tysięcy. W przypadku pominięcia zostanie użyta wartość zmiennej ThousandSep ustawiona w skrypcie ładowania danych.

Przykłady i wyniki:
W poniższej tabeli przedstawiono wynik *Num#(A, '#', '.' , ',')* dla różnych wartości A.

Wyniki

Symbol	Reprezentacja ciągu	Wartość liczbowa (tutaj wyświetlana z kropką dziesiętną)
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

Funkcja **Text()** wymusza traktowanie wyrażenia jako tekstu, nawet jeśli możliwa jest jego interpretacja liczbowa.

Składnia:

```
Text (expr)
```

Typ zwracanych danych: dual

Przykład:

```
Text( A )
```
,gdzie A=1234

Wyniki

Ciąg znaków	Liczba
1234	-

Przykład:

```
Text( pi( ) )
```

Wyniki

Ciąg znaków	Liczba
3.1415926535898	-

Time#

Funkcja **Time#()** ocenia wyrażenie jako wartość czasu w formacie czasu ustawionym w skrypcie ładowania danych lub systemie operacyjnym, chyba że podano format ciągu znaków..

Składnia:

```
time#(text[, format])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
text	Tekstowy ciąg znaków do oceny.
format	Ciąg znaków opisujący format przetwarzanego ciągu tekstowego. W przypadku pominięcia użyty jest krótki format daty, format godziny i separator dziesiętny ustawiony w systemie operacyjnym.

Przykład:

- Domyślne ustawienie formatu godziny 1: hh:mm:ss
- Domyślne ustawienie formatu godziny 2: hh.mm.ss

```
time#( A )
```
gdzie A=09:00:00

Wyniki

Wyniki	Ustawienie 1	Ustawienie 2
Ciąg znaków:	09:00:00	09:00:00
Liczba:	0.375	-

Przykład:

- Domyślne ustawienie formatu godziny 1: hh:mm:ss
- Domyślne ustawienie formatu godziny 2: hh.mm.ss

```
time#( A, 'hh.mm' )
```
,gdzie A=09,00

Wyniki

Wyniki	Ustawienie 1	Ustawienie 2
Ciąg znaków:	09.00	09.00
Liczba:	0.375	0.375

Timestamp#

Funkcja **Timestamp#()** ocenia wyrażenie jako wartość daty i czasu w formacie znacznika czasu ustawionym w skrypcie ładowania danych lub systemie operacyjnym, chyba że podano format ciągu znaków.

Składnia:

```
timestamp#(text[, format])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
text	Tekstowy ciąg znaków do oceny.
format	Ciąg znaków opisujący format przetwarzanego ciągu tekstowego. W przypadku pominięcia użyty jest krótki format daty, format godziny i separator dziesiętny ustawiony w systemie operacyjnym. Norma ISO 8601 jest obsługiwana w zakresie znaczników czasu.

Przykład:

W poniższym przykładzie użyto formatu daty **M/D/YYYY**. Format daty jest określony w instrukcji **SET DateFormat** u góry skryptu ładowania danych.

Dodaj ten przykładowy skrypt do aplikacji i uruchom ją.

```
Load *,
Timestamp(Timestamp#(String)) as TS;
LOAD * INLINE [
Ciąg znaków
2015-09-15T12:13:14
1952-10-16T13:14:00+0200
1109-03-01T14:15
];
```

W przypadku utworzenia tabeli z wymiarami **String** i **TS** otrzymane wyniki będą następujące:

Wyniki

Ciąg znaków	TS
2015-09-15T12:13:14	9/15/2015 12:13:14 PM
1952-10-16T13:14:00+0200	10/16/1952 11:14:00 AM
1109-03-01T14:15	3/1/1109 2:15:00 PM

5.16 Funkcje międzywierszowe

Funkcje międzywierszowe są stosowane:

- W skryptach ładowania danych, gdy ocena bieżącego rekordu wymaga pobrania wartości z wcześniej załadowanych rekordów danych.

- W wyrażeniu wykresu, gdy wymagane jest pobranie kolejnej wartości z zestawu danych wizualizacji.

Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, gdy w dowolnym z wyrażeń wykresu stosowana jest funkcja międzywierszowa wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz międzywierszowej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji zostanie przywrócone do posortowania danych wejściowych funkcji międzywierszowej. To ograniczenie nie dotyczy równoważnej funkcji skryptu, jeśli taka istnieje.

Definicje wyrażeń z odwołaniami do samych siebie mogą być niezawodnie wykonywane w tabelach zawierających mniej niż 100 wierszy, ale może to być zależne od sprzętu, na którym działa silnik Qlik.

Funkcje wierszy

Te funkcje mogą być używane tylko w wyrażeniach wykresu.

Above

Funkcja **Above()** oblicza wartość wyrażenia w wierszu znajdującym się nad bieżącym wierszem w ramach segmentu kolumny w tabeli. Wiersz wybierany do obliczeń zależy od wartości opcjonalnego parametru **offset** – domyślnie jest to wiersz bezpośrednio poprzedzający bieżący. W przypadku wykresów niebędących tabelami funkcja **Above()** oblicza wartość dla wiersza poprzedzającego bieżący w tabeli prostej odpowiadającej wykresowi.

```
Above — funkcja wykresu([TOTAL [<fld{,fld}>]] expr [ , offset [,count]])
```

Below

Funkcja **Below()** oblicza wartość wyrażenia w wierszu znajdującym się pod bieżącym wierszem w ramach segmentu kolumny w tabeli. Wiersz wybierany do obliczeń zależy od wartości opcjonalnego parametru **offset** – domyślnie jest to wiersz bezpośrednio po wierszu bieżącym. W przypadku wykresów niebędących tabelami funkcja **Below()** oblicza wartość dla wiersza znajdującego się bezpośrednio po wierszu bieżącym w tabeli prostej odpowiadającej wykresowi.

```
Below — funkcja wykresu([TOTAL[<fld{,fld}>]] expression [ , offset [,count
 ]])
```

Bottom

Funkcja **Bottom()** oblicza wartość wyrażenia w ostatnim (najniższym) wierszu segmentu kolumny w tabeli. Wiersz wybierany do obliczeń zależy od wartości opcjonalnego parametru **offset** – domyślnie jest to wiersz najniższy. W przypadku wykresów niebędących tabelami obliczenie jest wykonywane na ostatnim wierszu bieżącej kolumny w tabeli prostej odpowiadającej wykresowi.

```
Bottom — funkcja wykresu([TOTAL[<fld{,fld}>]] expr [ , offset [,count ]])
```

Top

Funkcja **Top()** oblicza wartość wyrażenia w pierwszym (najwyższym) wierszu segmentu kolumny w tabeli. Wiersz wybierany do obliczeń zależy od wartości opcjonalnego parametru **offset** – domyślnie jest to wiersz najwyższy. W przypadku wykresów niebędących tabelami wynik funkcji **Top()** jest obliczany na pierwszym wierszu bieżącej kolumny w tabeli prostej odpowiadającej wykresowi.

```
Top — funkcja wykresu([TOTAL [<fld{,fld}>]] expr [ , offset [,count ]])
```

NoOfRows

Funkcja **NoOfRows()** zwraca liczbę wierszy w bieżącym segmencie kolumn tabeli. W przypadku wykresów bitmapowych funkcja **NoOfRows()** zwraca liczbę wierszy w tabeli prostej odpowiadającej wykresowi.

```
NoOfRows — funkcja wykresu([TOTAL])
```

Funkcje kolumn

Te funkcje mogą być używane tylko w wyrażeniach wykresu.

Column

Funkcja **Column()** zwraca wartość z kolumny określonej jako **ColumnNo** w tabeli prostej, bez uwzględniania wymiarów. Na przykład wyrażenie **Column(2)** zwraca wartość z drugiej kolumny miary.

```
Column — funkcja wykresu(ColumnNo)
```

Dimensionality

Funkcja **Dimensionality()** zwraca liczbę wymiarów bieżącego wiersza. W przypadku tabel przestawnych funkcja zwraca łączną liczbę kolumn wymiaru zawierających dane nieagregowane, czyli niezawierających sum częściowych ani zwiniętych agregacji.

```
Dimensionality — funkcja wykresu ( )
```

Secondarydimensionality

Funkcja **SecondaryDimensionality()** zwraca liczbę wierszy wymiaru tabeli przestawnej, które zawierają treść niepodlegającą agregacji, czyli nie zawierają sum częściowych ani zwiniętych agregacji. Ta funkcja jest odpowiednikiem funkcji **dimensionality()** dla poziomych wymiarów tabeli przestawnej.

```
SecondaryDimensionality — funkcja wykresu ( )
```

Funkcje pól

FieldIndex

Funkcja **FieldIndex()** zwraca pozycję wartości pola **value** w polu **field_name** (według kolejności ładowania).

```
FieldIndex (field_name , value)
```

FieldValue

Funkcja **FieldValue()** zwraca wartość znalezioną na pozycji **elem_no** pola **field_name** (według kolejności ładowania).

```
FieldValue (field_name , elem_no)
```

FieldValueCount

FieldValueCount() to funkcja zwracająca **liczbę całkowitą**, która znajduje liczbę odrębnych wartości w polu.

```
FieldValueCount (field_name)
```

Funkcje tabeli przestawnych

Te funkcje mogą być używane tylko w wyrażeniach wykresu.

After

Funkcja **After()** zwraca wartość wyrażenia obliczoną na podstawie wartości wymiaru tabeli przestawnej występujących w kolumnie znajdującej się za kolumną bieżącą w ramach segmentu wierszy tabeli przestawnej.

```
After — funkcja wykresu([TOTAL] expression [ , offset [,n]])
```

Before

Funkcja **Before()** zwraca wartość wyrażenia obliczoną na podstawie wartości wymiaru tabeli przestawnej występujących w kolumnie znajdującej się przed kolumną bieżącą w ramach segmentu wierszy tabeli przestawnej.

```
Before — funkcja wykresu([TOTAL] expression [ , offset [,n]])
```

First

Funkcja **First()** zwraca wartość wyrażenia obliczoną na podstawie wartości wymiaru tabeli przestawnej występujących w pierwszej kolumnie bieżącego segmentu wierszy tabeli przestawnej. Ta funkcja zwraca wartość NULL we wszystkich typach wykresów z wyjątkiem tabel przestawnych.

```
First — funkcja wykresu([TOTAL] expression [ , offset [,n]])
```

Last

Funkcja **Last()** zwraca wartość wyrażenia obliczoną na podstawie wartości wymiaru tabeli przestawnej występujących w ostatniej kolumnie bieżącego segmentu wierszy tabeli przestawnej. Ta funkcja zwraca wartość NULL we wszystkich typach wykresów z wyjątkiem tabel przestawnych.

```
Last — funkcja wykresu([TOTAL] expression [ , offset [,n]])
```

ColumnNo

Funkcja **ColumnNo()** zwraca numer bieżącej kolumny w bieżącym segmencie wierszy tabeli przestawnej. Pierwsza kolumna ma numer 1.

```
ColumnNo — funkcja wykresu([TOTAL])
```

NoOfColumns

Funkcja **NoOfColumns()** zwraca liczbę kolumn w bieżącym segmencie wierszy tabeli przestawnej.

```
NoOfColumns — funkcja wykresu([TOTAL])
```

Funkcje międzywierszowe w skrypcie ładowania danych

Exists

Funkcja **Exists()** określa, czy podana wartość pola została już załadowana w polu w skrypcie ładowania danych. Funkcja zwraca wartość TRUE lub FALSE, dzięki czemu może zostać użyta w klauzuli **where** instrukcji **LOAD** lub instrukcji **IF**.

```
Exists (field_name [, expr])
```

LookUp

Funkcja **Lookup()** sprawdza załadowaną tabelę i zwraca wartość **field_name** odpowiadającą pierwszemu wystąpieniu wartości **match_field_value** w polu **match_field_name**. Tabela ta może być tabelą bieżącą lub inną wcześniej załadowaną tabelą.

```
LookUp (field_name, match_field_name, match_field_value [, table_name])
```

Peek

Peek() wyszukuje wartość pola w tabeli z wiersza, który został już załadowany. Numer wiersza może być określony, podobnie jak tabela. Jeśli nie określono numeru wiersza, zostanie użyty ostatnio załadowany rekord.

```
Peek (field_name[, row_no[, table_name ] ])
```

Previous

Funkcja **Previous()** wyszukuje wartość wyrażenia **expr** przy użyciu danych z poprzedniego rekordu wejściowego, który nie został odrzucony z powodu klauzuli **where**. W przypadku pierwszego wiersza tabeli wewnętrznej funkcja zwróci wartość NULL.

```
Previous (page 1287)(expr)
```

Zob. także:

p *Funkcje zakresu (page 1308)*

Above – funkcja wykresu

Funkcja **Above()** oblicza wartość wyrażenia w wierszu znajdującym się nad bieżącym wierszem w ramach segmentu kolumny w tabeli. Wiersz wybierany do obliczeń zależy od wartości opcjonalnego parametru **offset** – domyślnie jest to wiersz bezpośrednio poprzedzający bieżący. W przypadku wykresów niebędących tabelami funkcja **Above()** oblicza wartość dla wiersza poprzedzającego bieżący w tabeli prostej odpowiadającej wykresowi.

Składnia:

```
Above([TOTAL] expr [ , offset [,count]])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
offset	Określenie wartości offsetn większej niż 0 umożliwia przeniesienie oceny wyrażenia o n wierszy w górę z bieżącego wiersza. Podanie argumentu offset równego 0 spowoduje ocenę wartości wyrażenia w bieżącym wierszu. Określenie ujemnej wartości argumentu offset sprawia, że funkcja Above działa tak samo jak funkcja Below z odpowiednią dodatnią wartością argumentu offset.
count	Jeśli trzeci argument **count** będzie większy od 1, funkcja zwróci zakres wartości **count**, po jednej dla każdego z **count** wierszy tabeli liczonych w górę od pierwotnej komórki. W tej postaci funkcja może być używana jako argument dla specjalnych funkcji zakresu. *Funkcje zakresu (page 1308)*
TOTAL	Jeśli tabela jest jednowymiarowa lub jako argument zostanie podany kwalifikator **TOTAL**, bieżący segment kolumny jest zawsze równy całej kolumnie.

W przypadku pierwszego wiersza segmentu kolumny zostanie zwrócona wartość NULL, ponieważ nie ma żadnego wiersza nad tym wierszem.

Segment kolumny definiuje się jako ciągły podzbiór komórek o tych samych wartościach dla wymiarów w bieżącej kolejności sortowania. Międzywierszowe funkcje wykresu są obliczane w segmencie kolumny z pominięciem skrajnego prawego wymiaru w równoważnym wykresie tabeli prostej. Jeśli wykres ma tylko jeden wymiar lub jeśli podano kwalifikator TOTAL, wartość wyrażenia jest obliczana na pełnej tabeli.

Jeśli tabela lub równoważnik tabeli zawiera wiele wymiarów pionowych, wówczas segment bieżącej kolumny będzie zawierał tylko wiersze z takimi samymi wartościami we wszystkich kolumnach wymiaru jak bieżący wiersz, ale bez kolumny przedstawiającej ostatni wymiar w kolejności sortowania między polami.

Ograniczenia:

- Wywołania rekurencyjne będą zwracać NULL.
- Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji.

Przykłady i wyniki:

Example 1:

Wizualizacja tabeli dla przykładu 1

Customer	Sum([Sales])	Above(Sum(Sales))	Sum(Sales)+Above(Sum(Sales))	Above offset 3	Higher?
	2566	-	-	-	-
Astrida	587	-	-	-	-
Betacab	539	587	1126	-	-
Canutility	683	539	1222	-	Higher
Divadip	757	683	1440	1344	Higher

Na zrzucie ekranu w tym przykładzie widoczne jest, że stosowna wizualizacja tabeli została utworzona na podstawie wymiaru **Customer** oraz miar `Sum(Sales)` i `Above(Sum(Sales))`.

W kolumnie Above(Sum(Sales)) zwracane są wartościNULL dla wiersza wymiaru **Customer** zawierającego wartość **Astrida**, ponieważ nie ma żadnego wiersza nad tym wierszem. Wynik dla klienta **Betacab** pokazuje wartość Sum(Sales) dla klienta **Astrida**, wynik dla klienta **Canutility** pokazuje wartość **Sum(Sales)** dla klienta **Betacab** itd.

W kolumnie zatytułowanej Sum(Sales)+Above(Sum(Sales)) wiersz klienta **Betacab** pokazuje wynik dodawania wartości **Sum(Sales)** z wierszy klientów **Betacab** i **Astrida** (539+587). Wynik w wierszu klienta **Betacab** pokazuje wynik dodawania wartości **Sum(Sales)** z wierszy klientów **Canutility** i **Canutility** (683+539).

Miara zatytułowana Above offset 3 utworzona za pomocą wyrażenia `Sum(Sales)+Above(Sum(Sales), 3)` ma argument **offset** o wartości 3, co oznacza, że pobiera wartość z wiersza znajdującego się o trzy wiersze powyżej bieżącego wiersza. W efekcie jej działania wartość **Sum(Sales)** dla bieżącego wiersza w kolumnie **Customer** jest dodawana do wartości wiersza w kolumnie **Customer** znajdującej się o trzy wiersze powyżej. Dla pierwszych trzech wierszy w kolumnie **Customer** zwracane są wartości null.

W tabeli tej przedstawione są również bardziej złożone miary: jedna utworzona na podstawie wyrażenia `Sum(Sales)+Above(Sum(Sales))` i jedna zatytułowana **Higher?**, utworzona na podstawie wyrażenia `IF(Sum(Sales)>Above(Sum(Sales)), 'Higher')`.

Tej funkcji można też używać w wykresach innego rodzaju niż tabele, na przykład wykresach słupkowych.

W przypadku innych typów wykresu należy przekształcić wykres w odpowiadającą mu tabelę prostą, aby umożliwić łatwe interpretowanie wiersza, do którego odnosi się funkcja.

Example 2:

Na zrzutach ekranu tabel przedstawionych w tym przykładzie do wizualizacji zostały dodane kolejne wymiary: **Month** i **Product**. W przypadku wykresów wielowymiarowych wyniki wyrażeń zawierających funkcje **Above**, **Below**, **Top** i **Bottom** są zależne od kolejności sortowania wymiarów kolumn przez Qlik Sense. Qlik Sense oblicza wartości funkcji na podstawie segmentów kolumny uzyskanych z wymiaru sortowanego jako ostatni. Kolejność sortowania kolumn określa się w panelu właściwości w sekcji **Sortowanie** i może ona być inna od kolejności wyświetlania kolumn w tabeli.

W wizualizacji tabeli dotyczącej przykładu 2 ostatnim sortowanym wymiarem jest **Month**, funkcja **Above** dokonuje zatem oceny na podstawie miesięcy. Dla każdej wartości w kolumnie **Product** istnieje szereg wyników dla każdego miesiąca (od **Jan** do **Aug**) – segment kolumny. Następnie pojawia się szereg dla następnego segmentu kolumny: dla każdego miesiąca **Month** dla następnej wartości z kolumny **Product**. Będzie istnieć segment kolumny dla każdej wartości w kolumnie **Customer** dla każdej wartości z kolumny **Product**.

Wizualizacja tabeli dla przykładu 2

Customer	Product	Month	Sum([Sales])	Above(Sum(Sales))
			2566	-
Astrida	AA	Jan	46	-
Astrida	AA	Feb	60	46
Astrida	AA	Mar	70	60
Astrida	AA	Apr	13	70
Astrida	AA	May	78	13
Astrida	AA	Jun	20	78
Astrida	AA	Jul	45	20
Astrida	AA	Aug	65	45

Example 3:

W wizualizacji tabeli dotyczącej przykładu 3 ostatnim sortowanym wymiarem jest **Product**. Jest to wynikiem przesunięcia wymiaru Product do pozycji 3 na karcie Sortuj w panelu właściwości. Funkcja **Above** jest oceniana dla każdej wartości w kolumnie **Product**. Występują tam tylko dwa produkty (**AA** i **BB**), w każdym szeregu jest zatem tylko jeden wynik niebędący wartością null. W wierszu produktu **BB** dla

miesiąca **Jan** wartość wyrażenia **Above(Sum(Sales))** wynosi 46. W przypadku wiersza **AA** wartość wynosi null. Wartość w każdym wierszu **AA** dla dowolnego miesiąca zawsze będzie null, ponieważ nie ma żadnej wartości **Product** powyżej AA. Drugi szereg jest oceniany dla wartości **AA** i **BB** za miesiąc **Feb** dla znajdującego się w kolumnie **Customer** klienta **Astrida**. Po dokonaniu oceny wszystkich miesięcy dla klienta **Astrida** sekwencja ta zostaje powtórzona dla kolejnego klienta (**Customer**) Betacab itd.

Wizualizacja tabeli dla przykładu 3

Customer	Product	Month	Sum([Sales])	Above(Sum(Sales))
			2566	-
Astrida	AA	Jan	46	-
Astrida	BB	Jan	46	46
Astrida	AA	Feb	60	-
Astrida	BB	Feb	60	60
Astrida	AA	Mar	70	-
Astrida	BB	Mar	70	70
Astrida	AA	Apr	13	-
Astrida	BB	Apr	13	13

Przykład 4

Example 4:	**Wynik**
Funkcję Above można stosować jako źródło danych wejściowych dla funkcji zakresu. Na przykład: `RangeAvg (Above(Sum (Sales),1,3))`.	W funkcji Above() argument offset ma wartość 1, a argument count ma wartość 3. Funkcja ta znajduje wyniki wyrażeniaSum(Sales) w trzech wierszach znajdujących się bezpośrednio nad bieżącym wierszem w segmencie kolumny (o ile jest tam wiersz). Wspomniane trzy wartości są stosowane jako dane wejściowe dla funkcji RangeAvg(), która znajduje średnią wartości w określonym szeregu liczbowym. Tabela z kolumną Customer jako wymiarem zwraca następujące wyniki dla wyrażenia RangeAvg(). Astrida - Betacab 587 Canutility 563 Divadip: 603

Dane zastosowane w przykładach:

```
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 '|');
```

Zob. także:

p *Below – funkcja wykresu (page 1258)*
p *Bottom – funkcja wykresu (page 1262)*
p *Top – funkcja wykresu (page 1288)*
p *RangeAvg (page 1311)*

Below – funkcja wykresu

Funkcja **Below()** oblicza wartość wyrażenia w wierszu znajdującym się pod bieżącym wierszem w ramach segmentu kolumny w tabeli. Wiersz wybierany do obliczeń zależy od wartości opcjonalnego parametru **offset** – domyślnie jest to wiersz bezpośrednio po wierszu bieżącym. W przypadku wykresów niebędących tabelami funkcja **Below()** oblicza wartość dla wiersza znajdującego się bezpośrednio po wierszu bieżącym w tabeli prostej odpowiadającej wykresowi.

Składnia:

```
Below([TOTAL] expr [ , offset [,count ]])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.

Argument	Opis
offset	Określenie wartości **offset**n większej niż 1 umożliwia przeniesienie oceny wyrażenia o n wierszy w dół z bieżącego wiersza. Podanie argumentu offset równego 0 spowoduje ocenę wartości wyrażenia w bieżącym wierszu. Określenie ujemnej wartości argumentu offset sprawia, że funkcja **Below** działa tak samo jak funkcja **Above** z odpowiednią dodatnią wartością argumentu offset.
count	Jeśli trzeci parametr **count** będzie większy od 1, funkcja zwróci zakres wartości **count**, po jednej dla każdego z **count** wierszy tabeli liczonych w dół od pierwotnej komórki. W tej postaci funkcja może być używana jako argument dla specjalnych funkcji zakresu. *Funkcje zakresu (page 1308)*
TOTAL	Jeśli tabela jest jednowymiarowa lub jako argument zostanie podany kwalifikator **TOTAL**, bieżący segment kolumny jest zawsze równy całej kolumnie.

W przypadku ostatniego wiersza segmentu kolumny zostanie zwrócona wartość NULL, ponieważ nie ma żadnego wiersza pod nim.

Segment kolumny definiuje się jako ciągły podzbiór komórek o tych samych wartościach dla wymiarów w bieżącej kolejności sortowania. Międzywierszowe funkcje wykresu są obliczane w segmencie kolumny z pominięciem skrajnego prawego wymiaru w równoważnym wykresie tabeli prostej. Jeśli wykres ma tylko jeden wymiar lub jeśli podano kwalifikator TOTAL, wartość wyrażenia jest obliczana na pełnej tabeli.

Jeśli tabela lub równoważnik tabeli zawiera wiele wymiarów pionowych, wówczas segment bieżącej kolumny będzie zawierał tylko wiersze z takimi samymi wartościami we wszystkich kolumnach wymiaru jak bieżący wiersz, ale bez kolumny przedstawiającej ostatni wymiar w kolejności sortowania między polami.

Ograniczenia:

- Wywołania rekurencyjne będą zwracać NULL.
- Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji.

Przykłady i wyniki:

Example 1:

Wizualizacja tabeli dla przykładu 1

Customer	Sum([Sales])	Below(Sum(Sales))	Sum(Sales)+Below(Sum(Sales))	Below + Offset 3	Higher
	2566	-	-	-	-
Astrida	587	539	1126	1344	Higher
Betacab	539	683	1222	-	-
Canutility	683	757	1440	-	-
Divadip	757	-	-	-	-

Na zrzucie ekranu w przykładzie 1 widoczne jest, że stosowna wizualizacja tabeli została utworzona na podstawie wymiaru **Customer** oraz miar: `Sum(Sales)` i `Below(Sum(Sales))`.

W kolumnie **Below(Sum(Sales))** zwracane są wartościNULL dla wiersza wymiaru **Customer** zawierającego wartość **Divadip**, ponieważ nie ma żadnego wiersza pod tym wierszem. Wynik dla klienta **Canutility** pokazuje wartość Sum(Sales) dla klienta **Divadip**, wynik dla klienta **Betacab** pokazuje wartość **Sum(Sales)** dla klienta **Canutility** itd.

W tabeli tej przedstawione są również bardziej złożone miary, które znajdują się w kolumnach zatytułowanych `Sum(Sales)+Below(Sum(Sales))`, **Below +Offset 3** i **Higher?**. Wyrażenia te działają w sposób opisany w poniższych sekcjach.

W kolumnie zatytułowanej **Sum(Sales)+Below(Sum(Sales))** wiersz klienta **Astrida** pokazuje wynik dodawania wartości **Sum(Sales)** z wierszy klientów **Betacab** i **Astrida** (539+587). Wynik w wierszu klienta **Betacab** pokazuje wynik dodawania wartości **Sum(Sales)** z wierszy klientów **Canutility** i **Betacab** (539+683).

Miara zatytułowana **Below +Offset 3** utworzona za pomocą wyrażenia `Sum(Sales)+Below(Sum(Sales), 3)` ma argument **offset** o wartości 3, co oznacza, że pobiera wartość z wiersza znajdującego się o trzy wiersze poniżej bieżącego wiersza. W efekcie jej działania wartość **Sum(Sales)** dla bieżącego wiersza w kolumnie **Customer** jest dodawana do wartości wiersza w kolumnie **Customer** znajdującej się o trzy wiersze poniżej. Dla ostatnich trzech wierszy w kolumnie **Customer** zwracane są wartości null.

Miara zatytułowana **Higher?** została utworzona na podstawie wyrażenia `IF(Sum(Sales)>Below(Sum(Sales)), 'Higher')`. Wyrażenie to porównuje wartości z bieżącego wiersza w mierze **Sum(Sales)** z wartościami z wiersza poniżej. Jeśli wartość w bieżącym wierszu jest większa, zwracany jest tekst Higher.

Tej funkcji można też używać w wykresach innego rodzaju niż tabele, na przykład wykresach słupkowych.

W przypadku innych typów wykresu należy przekształcić wykres w odpowiadającą mu tabelę prostą, aby umożliwić łatwe interpretowanie wiersza, do którego odnosi się funkcja.

W przypadku wykresów wielowymiarowych wyniki wyrażeń zawierających funkcje **Above**, **Below**, **Top** i **Bottom** są zależne od kolejności sortowania wymiarów kolumn przez Qlik Sense. Qlik Sense oblicza wartości funkcji na podstawie segmentów kolumny uzyskanych z wymiaru sortowanego jako ostatni. Kolejność sortowania kolumn określa się w panelu właściwości w sekcji **Sortowanie** i może ona być inna od kolejności wyświetlania kolumn w tabeli.Więcej informacji na ten temat zawiera przykład 2 w funkcji **Above**.

Przykład 2

Example 2:	Wynik
Funkcję **Below** można stosować jako źródło danych wejściowych dla funkcji zakresu. Na przykład: `RangeAvg (Below(Sum(Sales),1,3))`.	W funkcji **Below()** argument offset ma wartość 1, a argument count ma wartość 3. Funkcja ta znajduje wyniki wyrażenia **Sum(Sales)** w trzech wierszach znajdujących się bezpośrednio pod bieżącym wierszem w segmencie kolumny (o ile jest tam wiersz). Wspomniane trzy wartości są stosowane jako dane wejściowe dla funkcji RangeAvg(), która znajduje średnią wartości w określonym szeregu liczbowym. Tabela z kolumną **Customer** jako wymiarem zwraca następujące wyniki dla wyrażenia RangeAvg().
	Astrida 659.67 Betacab 720 Canutility 757 Divadip: -

Dane zastosowane w przykładach:

```
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 '|');
```

Zob. także:

p *Above – funkcja wykresu (page 1253)*
p *Bottom – funkcja wykresu (page 1262)*
p *Top – funkcja wykresu (page 1288)*
p *RangeAvg (page 1311)*

Bottom – funkcja wykresu

Funkcja **Bottom()** oblicza wartość wyrażenia w ostatnim (najniższym) wierszu segmentu kolumny w tabeli. Wiersz wybierany do obliczeń zależy od wartości opcjonalnego parametru **offset** – domyślnie jest to wiersz najniższy. W przypadku wykresów niebędących tabelami obliczenie jest wykonywane na ostatnim wierszu bieżącej kolumny w tabeli prostej odpowiadającej wykresowi.

Składnia:

```
Bottom([TOTAL] expr [ , offset [,count ]])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
offset	Określenie wartości **offset**n większej niż 1 umożliwia przeniesienie oceny wyrażenia do góry o n wierszy ponad najniższy wiersz. Określenie ujemnej wartości argumentu offset sprawia, że funkcja **Bottom** działa tak samo jak funkcja **Top** z odpowiednią dodatnią wartością argumentu offset.
count	Jeśli trzeci parametr **count** będzie większy od 1, funkcja zwróci zamiast jednej wartości cały zakres wartości **count**, po jednej dla każdego z **count** ostatnich wierszy bieżącego segmentu kolumny. W tej postaci funkcja może być używana jako argument dla specjalnych funkcji zakresu. *Funkcje zakresu (page 1308)*
TOTAL	Jeśli tabela jest jednowymiarowa lub jako argument zostanie podany kwalifikator **TOTAL**, bieżący segment kolumny jest zawsze równy całej kolumnie.

Segment kolumny definiuje się jako ciągły podzbiór komórek o tych samych wartościach dla wymiarów w bieżącej kolejności sortowania. Międzywierszowe funkcje wykresu są obliczane w segmencie kolumny z pominięciem skrajnego prawego wymiaru w równoważnym wykresie tabeli prostej. Jeśli wykres ma tylko jeden wymiar lub jeśli podano kwalifikator TOTAL, wartość wyrażenia jest obliczana na pełnej tabeli.

Jeśli tabela lub równoważnik tabeli zawiera wiele wymiarów pionowych, wówczas segment bieżącej kolumny będzie zawierał tylko wiersze z takimi samymi wartościami we wszystkich kolumnach wymiaru jak bieżący wiersz, ale bez kolumny przedstawiającej ostatni wymiar w kolejności sortowania między polami.

Ograniczenia:

- Wywołania rekurencyjne będą zwracać NULL.
- Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji.

Przykłady i wyniki:

Wizualizacja tabeli dla przykładu 1

Customer	Sum([Sales])	Bottom(Sum(Sales))	Sum(Sales)+Bottom(Sum(Sales))	Bottom offset 3
	2566	**757**	**3323**	**3105**
Astrida	587	757	1344	1126
Betacab	539	757	1296	1078
Canutility	683	757	1440	1222
Divadip	757	757	1514	1296

Na zrzucie ekranu w tym przykładzie widoczne jest, że stosowna wizualizacja tabeli została utworzona na podstawie wymiaru **Customer** oraz miar `Sum(Sales)` i `Bottom(Sum(Sales))`.

Kolumna **Bottom(Sum(Sales))** zwraca wartość 757 dla wszystkich wierszy, ponieważ jest to wartość najniższego wiersza: **Divadip**.

W tabeli tej przedstawione są również bardziej złożone miary: jedna utworzona na podstawie wyrażenia `Sum(Sales)+Bottom(Sum(Sales))` i jedna zatytułowana **Bottom offset 3**, utworzona na podstawie wyrażenia `Sum(Sales)+Bottom(Sum(Sales), 3)` z argumentem **offset** o wartości 3. W efekcie jej działania wartość **Sum(Sales)** dla bieżącego wiersza jest dodawana do wartości z trzeciego wiersza liczonego od najniższego wiersza (wartość z wiersza bieżącego plus wartość dla klienta **Betacab**).

Przykład: 2

Na zrzutach ekranu tabel przedstawionych w tym przykładzie do wizualizacji zostały dodane kolejne wymiary: **Month** i **Product**. W przypadku wykresów wielowymiarowych wyniki wyrażeń zawierających funkcje **Above**, **Below**, **Top** i **Bottom** są zależne od kolejności sortowania wymiarów kolumn przez Qlik Sense. Qlik Sense oblicza wartości funkcji na podstawie segmentów kolumny uzyskanych z wymiaru sortowanego jako ostatni. Kolejność sortowania kolumn określa się w panelu właściwości w sekcji **Sortowanie** i może ona być inna od kolejności wyświetlania kolumn w tabeli.

W pierwszej tabeli wyrażenie jest oceniane na podstawie wymiaru **Month**, a w drugiej tabeli – na podstawie wymiaru **Product**. Miara **End value** zawiera wyrażenie `Bottom(Sum(Sales))`. Najniższy wiersz dla miary **Month** to Dec, a wartość dla Dec i dla **Product** wynosi 22. (W celu oszczędności miejsca niektóre wiersze zostały wymazane).

Pierwsza tabela z przykładu 2. Wartość funkcji Bottom dla miary End value na podstawie wymiaru Month (Dec).

Customer	Product	Month	Sum(Sales)	End value
			2566	-
Astrida	AA	Jan	46	22
Astrida	AA	Feb	60	22
Astrida	AA	Mar	70	22
Astrida	AA	Sep	78	22
Astrida	AA	Oct	12	22
Astrida	AA	Nov	78	22
Astrida	AA	Dec	22	22
Astrida	BB	Jan	46	22

Druga tabela z przykładu 2. Wartość funkcji Bottom dla miary End value na podstawie wymiaru Product (BB dla klienta Astrida).

Customer	Product	Month	Sum(Sales)	End value
			2566	-
Astrida	AA	Jan	46	46
Astrida	BB	Jan	46	46
Astrida	AA	Feb	60	60
Astrida	BB	Feb	60	60
Astrida	AA	Mar	70	70
Astrida	BB	Mar	70	70
Astrida	AA	Apr	13	13
Astrida	BB	Apr	13	13

Więcej informacji na ten temat zawiera przykład 2 w funkcji **Above**.

Przykład 3

Przykład: 3	Wynik
Funkcję **Bottom** można stosować jako źródło danych wejściowych dla funkcji zakresu. Na przykład: `RangeAvg (Bottom(Sum(Sales),1,3))`.	W funkcji **Bottom()** argument offset ma wartość 1, a argument count ma wartość 3. Funkcja ta znajduje wyniki wyrażenia **Sum(Sales)** w trzech wierszach począwszy od wiersza nad najniższym wierszem w segmencie kolumny (ponieważ offset=1), oraz i w dwóch wierszach nad nim (o ile jest tam wiersz). Wspomniane trzy wartości są stosowane jako dane wejściowe dla funkcji RangeAvg(), która znajduje średnią wartości w określonym szeregu liczbowym. Tabela z kolumną **Customer** jako wymiarem zwraca następujące wyniki dla wyrażenia RangeAvg().
	Astrida 659.67 Betacab 659.67 Canutility 659.67 Divadip: 659.67

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Zob. także:

p *Top – funkcja wykresu (page 1288)*

Column – funkcja wykresu

Funkcja **Column()** zwraca wartość z kolumny określonej jako **ColumnNo** w tabeli prostej, bez uwzględniania wymiarów. Na przykład wyrażenie **Column(2)** zwraca wartość z drugiej kolumny miary.

Składnia:

```
Column(ColumnNo)
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
ColumnNo	Numer kolumny w tabeli zawierającej miarę. *Funkcja Column() ignoruje kolumny z wymiarem.*

Ograniczenia:

- Wywołania rekurencyjne będą zwracać NULL.
- Jeśli argument **ColumnNo** odwołuje się do kolumny, dla której nie istnieje miara, wówczas zwracana jest wartość NULL.
- Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji.

Przykłady i wyniki:

Przykład: Procent łącznej sprzedaży

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
A	AA	15	10	150	505	29.70
A	AA	16	4	64	505	12.67
A	BB	9	9	81	505	16.04

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
B	BB	10	5	50	505	9.90
B	CC	20	2	40	505	7.92
B	DD	25	-	0	505	0.00
C	AA	15	8	120	505	23.76
C	CC	19	-	0	505	0.00

Przykład: Procent sprzedaży dla wybranego klienta

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

Przykłady i wyniki

Przykłady	Wyniki
Do tabeli dodano miarę Order Value z wyrażeniem `Sum (UnitPrice*UnitSales)`. Total Sales Value dodano jako miarę z wyrażeniem: `Sum(TOTAL UnitPrice*UnitSales)` Do tabeli dodano miarę % Sales z wyrażeniem `100*Column (1)/Column(2)`.	Wynik funkcji Column(1) jest pobierany z kolumny Order Value, ponieważ jest to pierwsza kolumna z miarą. Wynik funkcji Column(2) jest pobierany z kolumny Total Sales Value, ponieważ jest to druga kolumna z miarą. Sprawdź wyniki w kolumnie % Sales na przykładzie zatytułowanym *Procent łącznej sprzedaży (page 1266)*
Wybierz Customer A.	Selekcja ta zmienia wartość w kolumnie Total Sales Value oraz w efekcie w kolumnie %Sales. Zobacz przykład *Procent sprzedaży dla wybranego klienta (page 1267)*

Dane zastosowane w przykładach:

```
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 – funkcja wykresu

Funkcja **Dimensionality()** zwraca liczbę wymiarów bieżącego wiersza. W przypadku tabel przestawnych funkcja zwraca łączną liczbę kolumn wymiaru zawierających dane nieagregowane, czyli niezawierających sum częściowych ani zwiniętych agregacji.

Składnia:

```
Dimensionality ( )
```

Typ zwracanych danych: integer

Ograniczenia:

Ta funkcja jest dostępna tylko w przypadku wykresów. Dla wszystkich typów wykresów – oprócz tabeli przestawnej – funkcja ta zwróci liczbę wymiarów we wszystkich wierszach oprócz sumy, która będzie wynosić 0.

Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji.

Przykład: Wyrażenie wykresu używające funkcji Dimensionality

Przykład: Wyrażenie wykresu
Funkcji Dimensionality() można używać z tabelą przestawną jako wyrażenia wykresu, w którym chcesz zastosować różne formatowania komórek w zależności od liczby wymiarów w wierszu zawierającym dane niezagregowane. W tym przykładzie użyto funkcji Dimensionality() w celu zastosowania koloru tła do komórek tabeli, które pasują do danego warunku.

Skrypt ładowania

Załaduj następujące dane w edytorze ładowania danych jako ładowanie wbudowane, aby utworzyć poniższy przykład wyrażeń wykresu.

```
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 ',');
```

Wyrażenie wykresu

Utwórz wizualizację tabeli przestawnej w arkuszu Qlik Sense z wymiarami **Country** i **Product**. Dodaj **Sum (Sales)**, **Sum(Budget)** i **Dimensionality()** jako miary.

W panelu **Właściwości** wprowadź następujące wyrażenie jako **Wyrażenie koloru tła** dla miary **Sum (Sales)**:

```
If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156),
If(Dimensionality()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)
))
```

Wynik:

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

Objaśnienie

Wyrażenie `If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156), If(Dimensionality()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)))` zawiera instrukcje warunkowe, które sprawdzają wartość funkcji Dimensionality oraz Sum(Sales) i Sum(Budget) dla każdego produktu. Jeśli warunki są spełnione, do wartości Sum(Sales) jest stosowany kolor tła.

Exists

Funkcja **Exists()** określa, czy podana wartość pola została już załadowana w polu w skrypcie ładowania danych. Funkcja zwraca wartość TRUE lub FALSE, dzięki czemu może zostać użyta w klauzuli **where** instrukcji **LOAD** lub instrukcji **IF**.

*Można również użyć funkcji **Not Exists()** w celu określenia, czy wartość pola nie została załadowana, ale zalecana jest ostrożność w przypadku używania funkcji **Not Exists()** w klauzuli where. Funkcja **Exists()** sprawdza zarówno poprzednio załadowane tabele, jak i poprzednio załadowane wartości w bieżącej tabeli. Z tego powodu zostanie załadowane tylko pierwsze wystąpienie. W momencie napotkania drugiego wystąpienia wartość jest już załadowana. Więcej informacji można znaleźć w przedstawionych przykładach.*

Składnia:

```
Exists(field_name [, expr])
```

Typ zwracanych danych: wartość logiczna

Argumenty:

Argumenty

Argument	Opis
field_name	Nazwa pola, w którym ma być wyszukiwana wartość. Można użyć jawnej nazwy pola bez cudzysłowów. Pole musi być już załadowane przez skrypt. Oznacza to, że nie można odwoływać się do pola, które jest ładowane przez klauzulę znajdującą się w dalszej części skryptu.
expr	Wartość, która ma zostać sprawdzona pod kątem tego, czy istnieje. Można użyć wartości jawnej albo wyrażenia, które odwołuje się do jednego lub wielu pól w bieżącej instrukcji LOAD. *Nie można odwoływać się do pól, które nie są uwzględnione w bieżącej instrukcji LOAD.* Ten argument jest opcjonalny. Jeśli zostanie pominięty, funkcja będzie sprawdzać, czy wartość **field_name** w bieżącym rekordzie już istnieje.

Przykłady i wyniki:

Przykład 1:

```
Exists (Employee)
```

Zwraca -1 (True), jeśli wartość pola **Employee** w bieżącym rekordzie już istnieje w jakimkolwiek wcześniej wczytanym rekordzie zawierającym to pole.

Instrukcje `Exists (Employee, Employee)` i `Exists (Employee)` są równoważne.

Przykład 2

```
Exists(Employee, 'Bill')
```

Zwraca -1 (True), jeśli wartość pola **'Bill'** występuje w bieżącej zawartości pola **Employee**.

Przykład 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;
```

W efekcie otrzymuje się tabelę, której można użyć w wizualizacji tabeli przy użyciu wymiarów Employee i Address.

Klauzula where w postaci `where Exists (Employee)` oznacza, że tylko nazwy z tabeli Citizens, które znajdują się także w tabeli Employees, są ładowane do nowej tabeli. Instrukcja Drop usuwa tabelę Employees w celu uniknięcia pomyłek.

Wyniki

Employee	Address
Bill	New York
John	Miami
Steve	Chicago

Przykład 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;
```

Klauzula where zawiera not: `where not Exists (Employee)`.

To oznacza, że tylko te nazwy z tabeli Citizens, które nie znajdują się w tabeli Employees, są ładowane do nowej tabeli.

Zwróć uwagę, że istnieją dwie wartości dla Lucy w tabeli Citizens, ale tylko jedna jest uwzględniana w tabeli wynikowej. Po załadowaniu pierwszego wiersza z wartością Lucy, jest ona uwzględniona w polu Employee. W związku z tym, gdy sprawdzany jest drugi wiersz, wartość już istnieje.

Wyniki

Pracownik	**Adres**
Mary	London
Lucy	Madrid

Przykład 5

Ten przykład pokazuje, jak załadować wszystkie wartości.

```
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;
```

Aby móc uzyskać wszystkie wartości dla Lucy, zmieniono dwie rzeczy:

- Poprzednie ładowanie do tabeli Employees zostało wstawione tam, gdzie zmieniono nazwę Employee na Name.
  `Load Employee As Name;`
- Warunek Where w Citizens został zmieniony na:
  `not Exists (Name, Employee).`

Powoduje to tworzenie pól dla Name i Employee. Kiedy sprawdzany jest drugi wiersz z Lucy, to wartość ta nadal nie istnieje w Name.

Wyniki

Pracownik	Adres
Mary	London
Lucy	Madrid
Lucy	Paris

FieldIndex

Funkcja **FieldIndex()** zwraca pozycję wartości pola **value** w polu **field_name** (według kolejności ładowania).

Składnia:

```
FieldIndex(field_name , value)
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
field_name	Nazwa pola, dla którego wymagany jest indeks. Na przykład kolumna w tabeli. Argument musi być podany jako wartość ciągu znaków. Oznacza to, że nazwa pola musi być ujęta w pojedyncze cudzysłowy.
value	Wartość w polu **field_name**.

Ograniczenia:

- Jeśli nie można znaleźć wartości **value** wśród wartości pola **field_name**, zwracana jest wartość 0.
- Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji. To ograniczenie nie dotyczy równoważnej funkcji skryptu.

Przykłady i wyniki:

W poniższych przykładach stosowane jest pole: **First name** z tabeli **Names**.

Przykłady i wyniki

Przykłady	Wyniki
Dodaj przykładowe dane do aplikacji i uruchom ją.	Załadowano tabelę **Names**, tak jak w danych z próby.
Funkcja wykresu: w tabeli zawierającej wymiar First name dodaj jako miarę:	
`FieldIndex ('First name','John')`	1, ponieważ wartość John pojawia się jako pierwsza w kolejności ładowania pola **First name**. Warto zauważyć, że w panelu filtrowania imię **John** pojawiłoby się jako drugie od góry, ponieważ podlega sortowaniu alfabetycznemu, nie tak, jak w kolejności ładowania.
`FieldIndex ('First name','Peter')`	4, ponieważ funkcja **FieldIndex()** zwraca tylko jedną wartość (pierwsze wystąpienie w kolejności ładowania).
Funkcja skryptu: przy założeniu, że tabela **Names** jest załadowana, jak w danych przykładowych:	

`John1:`
`Load FieldIndex('First name','John') as MyJohnPos`
`Resident Names;`
	`MyJohnPos=`1, ponieważ wartość John pojawia się jako pierwsza w kolejności ładowania pola **First name**. Warto zauważyć, że w panelu filtrowania imię **John** pojawiłoby się jako drugie od góry, ponieważ podlega sortowaniu alfabetycznemu, nie tak, jak w kolejności ładowania.

`Peter1:`
`Load FieldIndex('First name','Peter') as MyPeterPos`
`Resident Names;`
	`MyPeterPos=`4, ponieważ funkcja **FieldIndex()** zwraca tylko jedną wartość (pierwsze wystąpienie w kolejności ładowania).

Dane zastosowane w przykładzie:

```
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

Funkcja **FieldValue()** zwraca wartość znalezioną na pozycji **elem_no** pola **field_name** (według kolejności ładowania).

Składnia:

```
FieldValue(field_name , elem_no)
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
field_name	Nazwa pola, dla którego wymagana jest wartość. Na przykład kolumna w tabeli. Argument musi być podany jako wartość ciągu znaków. Oznacza to, że nazwa pola musi być ujęta w pojedyncze cudzysłowy.
elem_no	Zgodny z kolejnością ładowania numer pozycji (elementu) pola, dla którego zwracana jest wartość. Może on dotyczyć wiersza w tabeli, ale zależy od kolejności, w jakiej elementy (wiersze) są ładowane.

Ograniczenia:

- Jeśli wartość argumentu **elem_no** jest większa niż liczba wartości pól, wówczas zwracana jest wartość NULL.
- Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji. To ograniczenie nie dotyczy równoważnej funkcji skryptu.

Przykład

Skrypt ładowania

Załaduj następujące dane w edytorze ładowania danych jako ładowanie wbudowane, aby utworzyć poniższy przykład.

```
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;
```

Tworzenie wizualizacji

Utwórz wizualizację tabeli w arkuszu Qlik Sense. Dodaj do tabeli pola **First name**, **MyPos1** i **MyPos2**.

Wynik

First name	MyPos1	MyPos2
Jane	John	Jane
John	John	Jane
Mark	John	Jane
Peter	John	Jane
Sue	John	Jane

Objaśnienie

FieldValue('First name','1') powoduje, że John jest wartością **MyPos1** dla wszystkich imion, ponieważ „John” pojawia się jako pierwsze w kolejności ładowania pola **Name**. Warto zauważyć, że w panelu filtrowania imię John pojawiłoby się jako drugie od góry, po imieniu Jane, ponieważ podlega sortowaniu alfabetycznemu, nie tak, jak w kolejności ładowania.

FieldValue('First name','5') powoduje, że Jane jest wartością **MyPos2** dla wszystkich imion, ponieważ „Jane” pojawia się jako piąte w kolejności ładowania pola **First name**.

FieldValueCount

FieldValueCount() to funkcja zwracająca **liczbę całkowitą**, która znajduje liczbę odrębnych wartości w polu.

Częściowe ładowanie może usunąć z danych wartości, które nie zostaną odzwierciedlone w zwracanej liczbie. Zwrócona liczba będzie odpowiadać wszystkim odrębnym wartościom, które zostały załadowane podczas początkowego ładowania lub dowolnego kolejnego ładowania częściowego.

Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji. To ograniczenie nie dotyczy równoważnej funkcji skryptu.

Składnia:

```
FieldValueCount(field_name)
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
field_name	Nazwa pola, dla którego wymagana jest wartość. Na przykład kolumna w tabeli. Argument musi być podany jako wartość ciągu znaków. Oznacza to, że nazwa pola musi być ujęta w pojedyncze cudzysłowy.

Przykłady i wyniki:

W poniższych przykładach stosowane jest pole: **First name** z tabeli **Names**.

Przykłady i wyniki

Przykłady	Wyniki
Dodaj przykładowe dane do aplikacji i uruchom ją.	Załadowano tabelę **Names**, tak jak w danych z próby.
Funkcja wykresu: w tabeli zawierającej wymiar First name dodaj jako miarę:	
`FieldValueCount('First name')`	5, ponieważ wartość **Peter** pojawia się dwukrotnie.
`FieldValueCount('Initials')`	6, ponieważ wartość **Initials** ma tylko dwie odrębne wartości.

Przykłady	Wyniki
Funkcja skryptu: przy założeniu, że tabela **Names** jest załadowana, jak w danych przykładowych:	

`FieldCount1:`
`Load FieldValueCount('First name') as MyFieldCount1`
`Resident Names;`
	`MyFieldCount1`=5, ponieważ wartość Peter pojawia się dwukrotnie.

`FieldCount2:`
`Load FieldValueCount('Initials') as MyInitialsCount1`
`Resident Names;`
	`MyFieldCount1`=6, ponieważ wartość Initials ma tylko dwie odrębne wartości.

Dane zastosowane w przykładach:

```
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

Funkcja **Lookup()** sprawdza załadowaną tabelę i zwraca wartość **field_name** odpowiadającą pierwszemu wystąpieniu wartości **match_field_value** w polu **match_field_name**. Tabela ta może być tabelą bieżącą lub inną wcześniej załadowaną tabelą.

Składnia:

```
lookup(field_name, match_field_name, match_field_value [, table_name])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
field_name	Nazwa pola, dla którego wymagana jest wartość zwracana. Wartość wejściowa musi być podana jako ciąg znaków (np. jako literał w cudzysłowie).

Argument	Opis
match_ field_name	Nazwa pola, w którym wyszukiwana jest wartość **match_field_value**. Wartość wejściowa musi być podana jako ciąg znaków (np. jako literał w cudzysłowie).
match_ field_value	Wartość wyszukiwana w polu **match_field_name**.
table_name	Nazwa tabeli, w której wyszukiwana jest wartość. Wartość wejściowa musi być podana jako ciąg znaków (np. jako literał w cudzysłowie). Jeśli argument **table_name** zostanie pominięty, przyjmowana jest bieżąca tabela.

Argumenty bez cudzysłowów odwołują się do bieżącej tabeli. Aby odwołać się do innych tabel, należy ująć argument w pojedyncze cudzysłowy.

Ograniczenia:

Kolejność wyszukiwania odpowiada kolejności ładowania, chyba że tabela jest wynikiem złożonych operacji, na przykład sprzężeń – w takim przypadku kolejność nie jest dobrze zdefiniowana. **field_name** i **match_field_name** muszą określać pola w tej samej tabeli, wskazanej argumentem **table_name**.

W razie braku dopasowania zwracane jest NULL.

Przykład

Skrypt ładowania

Załaduj następujące dane w edytorze ładowania danych jako ładowanie wbudowane, aby utworzyć poniższy przykład.

```
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;
```

Tworzenie wizualizacji

Utwórz wizualizację tabeli w arkuszu Qlik Sense. Dodaj pola **ProductID**, **InvoiceID**, **CustomerID**, **Units** i **CategoryID** do tabeli.

Wynik

Tabela wynikowa

ProductID	InvoiceID	CustomerID	Jednostki:	CategoryID
1	1	Astrida	8	1
2	1	Astrida	6	1

ProductID	InvoiceID	CustomerID	Jednostki:	CategoryID
3	2	Betacab	10	2
3	3	Divadip	5	2
4	4	Divadip	10	3

Objaśnienie

Przykładowe dane używają funkcji **Lookup()** w następującej formie:

```
Lookup('Category', 'ProductID', ProductID, 'ProductList')
```

W pierwszej kolejności ładowana jest tabela**ProductList**.

Przy użyciu funkcji **Lookup()** tworzona jest tabela **OrderData**. Określa ona trzeci argument jako **ProductID**. Jest to pole, w odniesieniu którego należy wyszukać wartość w drugim argumencie **'ProductID'** w tabeli **ProductList**, na co wskazuje ujęcie w pojedyncze cudzysłowy.

Funkcja zwraca wartość „**Category**" (w tabeli **ProductList**), załadowaną jako **CategoryID**.

Instrukcja **drop** usuwa tabelę **ProductList** z modelu danych, ponieważ nie jest wymagana, w wyniku czego pozostaje następująca wynikowa tabela **OrderData**.

Funkcja Lookup() jest elastyczna i może uzyskać dostęp do każdej wcześniej załadowanej tabeli. Jest jednak wolna w porównaniu z funkcją Applymap().

Zob. także:

p *ApplyMap (page 1300)*

NoOfRows – funkcja wykresu

Funkcja **NoOfRows()** zwraca liczbę wierszy w bieżącym segmencie kolumn tabeli. W przypadku wykresów bitmapowych funkcja **NoOfRows()** zwraca liczbę wierszy w tabeli prostej odpowiadającej wykresowi.

Jeśli tabela lub równoważnik tabeli zawiera wiele wymiarów pionowych, wówczas segment bieżącej kolumny będzie zawierał tylko wiersze z takimi samymi wartościami we wszystkich kolumnach wymiaru jak bieżący wiersz, ale bez kolumny przedstawiającej ostatni wymiar w kolejności sortowania między polami.

Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji.

Składnia:

```
NoOfRows([TOTAL])
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
TOTAL	Jeśli tabela jest jednowymiarowa lub jako argument zostanie podany kwalifikator **TOTAL**, bieżący segment kolumny jest zawsze równy całej kolumnie.

Przykład: Wyrażenie wykresu używające funkcji NoOfRows

Przykład – wyrażenie wykresu

Skrypt ładowania

Załaduj następujące dane w edytorze ładowania danych jako ładowanie wbudowane, aby utworzyć poniższe przykłady wyrażeń wykresu:

```
        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 '|');
```

Wyrażenie wykresu

Utwórz wizualizację tabeli w arkuszu Qlik Sense z wymiarami **Region** i **SubRegion**. Dodaj `RowNo( )`, `NoOfRows()` i `NoOfRows(Total)` jako miary.

Wynik

Region	SubRegion	RowNo()	NoOfRows()	NoOfRows (Total)
Africa	Eastern	1	2	9
Africa	Western	2	2	9
Americas	Central	1	2	9

Region	SubRegion	RowNo()	NoOfRows()	NoOfRows (Total)
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

Objaśnienie

W tym przykładzie sortowanie odbywa się według pierwszego wymiaru, Region. W rezultacie każdy segment kolumny składa się z grupy regionów o tej samej wartości, na przykład Africa.

Kolumna **RowNo()** pokazuje numery wierszy dla każdego segmentu kolumny, na przykład istnieją dwa wiersze dla regionu Africa. Numerowanie wierszy rozpoczyna się wówczas ponownie od wartości 1 dla następnego segmentu kolumny, czyli dla Americas.

Kolumna **NoOfRows()** zlicza liczbę wierszy w każdym segmencie kolumny, na przykład Europe ma trzy wiersze w segmencie kolumny.

Kolumna **NoOfRows(Total)** ignoruje wymiary ze względu na argument `TOTAL` dla `NoOfRows()` i służy do liczenia wierszy w tabeli.

Gdyby tabela została posortowana według drugiego wymiaru, SubRegion, segmenty kolumn opierałyby się na tym wymiarze, więc numeracja wierszy zmieniałaby się dla każdej wartości wymiaru SubRegion.

Zob. także:

p *RowNo – funkcja wykresu (page 572)*

Peek

Peek() wyszukuje wartość pola w tabeli z wiersza, który został już załadowany. Numer wiersza może być określony, podobnie jak tabela. Jeśli nie określono numeru wiersza, zostanie użyty ostatnio załadowany rekord.

Funkcja peek() służy najczęściej do znajdowania odpowiednich granic we wcześniej załadowanej tabeli, czyli pierwszej lub ostatniej wartości określonego pola. W większości przypadków ta wartość jest przechowywana w zmiennej do późniejszego wykorzystania, na przykład jako warunek w pętli do-while.

Składnia:

```
Peek(
field_name
[, row_no[, table_name ] ])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
field_name	Nazwa pola, dla którego wymagana jest wartość zwracana.Wartość wejściowa musi być podana jako ciąg znaków (np. jako literał w cudzysłowie).
row_no	Wiersz w tabeli określający wymagane pole. Może być wyrażeniem, ale musi dawać w wyniku liczbę całkowitą. 0 oznacza pierwszy rekord, 1 drugi rekord itd. Liczby ujemne określają kolejność od końca tabeli. -1 oznacza ostatni wczytany rekord. Jeśli argument **row_no** nie zostanie podany, przyjmowana jest wartość -1.
table_name	Etykieta tabeli bez końcowego dwukropka. Jeśli argument **table_name** nie zostanie podany, przyjmowana jest bieżąca tabela. Wartość **table_name** musi być podana w przypadku użycia poza instrukcją **LOAD** lub odnoszenia się do innej tabeli.

Ograniczenia:

Funkcja ta może zwracać tylko wartości z już załadowanych rekordów. Oznacza to, że w pierwszym rekordzie tabeli wywołanie używające -1 jako row_no zwróci NULL.

Przykłady i wyniki:

Przykład 1:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
EmployeeDates: Load * Inline [ EmployeeCode|StartDate|EndDate 101|02/11/2010|23/06/2012
102|01/11/2011|30/11/2013 103|02/01/2012| 104|02/01/2012|31/03/2012 105|01/04/2012|31/01/2013
106|02/11/2013| ] (delimiter is '|');       First_last_Employee: Load  EmployeeCode,  Peek
('EmployeeCode',0,'EmployeeDates') As FirstCode,  Peek('EmployeeCode',-1,'EmployeeDates') As
LastCode Resident EmployeeDates;
```

Tabela wynikowa

Kod pracownika	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, ponieważ `Peek('EmployeeCode',0, 'EmployeeDates')` zwraca pierwszą wartość EmployeeCode w tabeli EmployeeDates.

LastCode = 106, ponieważ `Peek('EmployeeCode',-1, 'EmployeeDates')` zwraca ostatnią wartość EmployeeCode w tabeli EmployeeDates.

Podstawienie wartości argumentu **row_no** zwraca wartości innych wierszy w tabeli w następujący sposób:

`Peek('EmployeeCode',2, 'EmployeeDates')` zwraca trzecią wartość w tabeli, 103, jako FirstCode.

Należy jednak pamiętać, że bez określenia tabeli jako trzeciego argumentu **table_name** w tych przykładach funkcja odwołuje się do bieżącej tabeli (w tym przypadku – wewnętrznej).

Przykład 2

Aby uzyskać dostęp do danych znajdujących się dalej w tabeli, należy to zrobić w dwóch krokach: najpierw załaduj całą tabelę do tabeli tymczasowej, a następnie posortuj ją ponownie podczas używania funkcji **Peek()**.

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
T1: LOAD * inline [ ID|Value 1|3 1|4 1|6 3|7 3|8 2|1 2|11 5|2 5|78 5|13 ] (delimiter is '|');
T2:  LOAD *, IF(ID=Peek('ID'), Peek('List')&','&Value,Value) AS List RESIDENT T1 ORDER BY ID
ASC; DROP TABLE T1;
```

Create a table in a sheet in your app with **ID**, **List**, and **Value** as the dimensions.

Tabela wynikowa

Identyfikator	Lista	Wartość
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

Instrukcja **IF()** jest tworzona z tabeli tymczasowej T1.
`Peek('ID')` odwołuje się do pola ID w poprzednim wierszu w bieżącej tabeli T2.
`Peek('List')` odwołuje się do pola List w poprzednim wierszu w tabeli T2, która obecnie jest tworzona podczas oceny wyrażenia.

Wartość instrukcji jest wyznaczana w następujący sposób:
Jeśli bieżąca wartość ID jest taka sama jak poprzednia wartość ID, należy zapisać wartość Peek('List') skonkatenowaną z bieżącą wartością Value. W przeciwnym wypadku należy zapisać tylko bieżącą wartość Value.

Jeśli funkcja Peek('List') już zawiera skonkatenowany wynik, nowy wynik Peek('List') będzie z nim skonkatenowany.

*Warto zwrócić uwagę na klauzulę **Order by**. Określa ona sposób porządkowania tabeli (rosnąco według ID). Bez niej funkcja Peek() użyje dowolnego porządkowania zawartego w tabeli wewnętrznej, co może prowadzić do nieprzewidywalnych wyników.*

Przykład 3

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
Amounts: Load  Date#(Month,'YYYY-MM') as Month, Amount, Peek(Amount) as AmountMonthBefore
Inline [Month,Amount 2022-01,2 2022-02,3 2022-03,7 2022-04,9 2022-05,4 2022-06,1];
```

Tabela wynikowa

Amount	AmountMonthBefore	Miesiąc
1	4	2022-06
2	-	2022-01
3	2	2022-02
4	9	2022-05
7	3	2022-03
9	7	2022-04

W polu AmountMonthBefore będzie przechowywana kwota z poprzedniego miesiąca.

W tym przypadku parametry row_no i table_name są pomijane, więc używane są wartości domyślne. W tym przykładzie następujące trzy wywołania funkcji są równoważne:

- Peek(Amount)
- Peek(Amount,-1)
- Peek(Amount,-1,'Amounts')

Użycie -1 jako row_no oznacza, że zostanie użyta wartość z poprzedniego wiersza. Podstawiając tę wartość, można pobrać wartości innych wierszy w tabeli:

Peek(Amount,2) zwraca trzecią wartość w tabeli: 7.

Przykład 4

Aby można było uzyskać prawidłowe wyniki, dane muszą być poprawnie posortowane, ale niestety nie zawsze tak jest. Ponadto funkcja Peek() nie może być używana w celu utworzenia odniesienia do danych, które nie zostały jeszcze załadowane. Problemów takich można uniknąć, korzystając z tabel tymczasowych i uruchamiając wiele przejść przez dane.

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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;
```

Objaśnienie

Początkowa tabela jest posortowana według miesięcy, co oznacza, że funkcja peek() w wielu przypadkach zwróciłaby kwotę za niewłaściwy produkt. Dlatego należy ponownie posortować tę tabelę. W tym celu uruchamia się drugie przejście przez dane, tworząc nową tabelę tmp2Amounts. Warto zwrócić uwagę na klauzulę Order by. Porządkuje ona rekordy najpierw według produktu, a następnie według miesiąca w porządku rosnącym.

Funkcja If() jest potrzebna, ponieważ AmountMonthBefore należy obliczyć tylko wtedy, gdy poprzedni wiersz zawiera dane dotyczące tego samego produktu, ale za poprzedni miesiąc. Porównując produkt w bieżącym wierszu z produktem w poprzednim wierszu, można zweryfikować ten warunek.

Po utworzeniu drugiej tabeli pierwsza tabela tmp1Amounts jest usuwana za pomocą instrukcji Drop Table.

Na koniec następuje trzecie przejście przez dane, ale tym razem z miesiącami posortowanymi w kolejności odwrotnej. W ten sposób można również obliczyć AmountMonthAfter.

Porządkowanie przy użyciu klauzul umożliwia określenie, jak tabela ma być uporządkowana. Bez tego funkcja Peek() użyje dowolnego porządkowania, jakie istnieje w tabeli wewnętrznej, co może prowadzić do nieprzewidywalnych wyników.

Wynik

Tabela wynikowa

Miesiąc	Produkt	Amount	AmountMonthBefore	AmountMonthAfter
2022-01	Symbol	8	-	6
2022-02	B	3	-	4
2022-03	Symbol	6	8	6

Miesiąc	Produkt	Amount	AmountMonthBefore	AmountMonthAfter
2022-04	B	4	3	1
2022-05	Symbol	6	6	5
2022-06	B	1	4	5
2022-01	Symbol	5	6	7
2022-02	B	5	1	6
2022-03	Symbol	7	5	4
2022-04	B	6	5	8
2022-05	Symbol	4	7	-
2022-06	B	8	6	-

Przykład 5

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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;
```

Wynik

Tabela wynikowa

Kwartał	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

Objaśnienie

Instrukcja LOAD **Load *, rangesum(SumVal,peek('AccSumVal')) as AccSumVal** zawiera wywołanie rekursywne, w którym poprzednie wartości są dodawane do wartości bieżącej. Ta operacja służy do obliczania akumulacji wartości w skrypcie.

Zob. także:

Previous

Funkcja **Previous()** wyszukuje wartość wyrażenia **expr** przy użyciu danych z poprzedniego rekordu wejściowego, który nie został odrzucony z powodu klauzuli **where**. W przypadku pierwszego wiersza tabeli wewnętrznej funkcja zwróci wartość NULL.

Składnia:

```
Previous(expr)
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane. Wyrażenie może zawierać zagnieżdżone funkcje **previous()** w celu uzyskiwania dostępu do bardziej odległych rekordów. Dane są pobierane bezpośrednio ze źródła danych wejściowych, co umożliwia odwoływanie się również do pól, które nie zostały załadowane do aplikacji Qlik Sense (nie zostały zapisane w asocjacyjnej bazie danych programu).

Ograniczenia:

W przypadku pierwszego rekordu tabeli wewnętrznej funkcja zwraca wartość NULL.

Przykład:

Wprowadź następujące dane do swojego skryptu ładowania

```
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 '|');
```

Korzystając z funkcji **Previous()** w instrukcji **Load**, możemy porównać bieżącą wartość Sales z poprzednią wartością i użyć jej w trzecim polu, Increase.

Tabela wynikowa

Miesiąc	Sales	Zwiększ
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 – funkcja wykresu

Funkcja **Top()** oblicza wartość wyrażenia w pierwszym (najwyższym) wierszu segmentu kolumny w tabeli. Wiersz wybierany do obliczeń zależy od wartości opcjonalnego parametru **offset** – domyślnie jest to wiersz najwyższy. W przypadku wykresów niebędących tabelami wynik funkcji **Top()** jest obliczany na pierwszym wierszu bieżącej kolumny w tabeli prostej odpowiadającej wykresowi.

Składnia:

```
Top([TOTAL] expr [ , offset [,count ]])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.

Argument	Opis
offset	Określenie wartości **offset**n większej niż 1 umożliwia przeniesienie oceny wyrażenia w dół o n wierszy poniżej najwyższego wiersza. Określenie ujemnej wartości argumentu offset sprawia, że funkcja **Top** działa tak samo jak funkcja **Bottom** z odpowiednią dodatnią wartością argumentu offset.
count	Jeśli trzeci parametr **count** będzie większy od 1, funkcja zwróci zakres wartości **count**, po jednej dla każdego z **count** ostatnich wierszy bieżącego segmentu kolumny. W tej postaci funkcja może być używana jako argument dla specjalnych funkcji zakresu. *Funkcje zakresu (page 1308)*
TOTAL	Jeśli tabela jest jednowymiarowa lub jako argument zostanie podany kwalifikator **TOTAL**, bieżący segment kolumny jest zawsze równy całej kolumnie.

Segment kolumny definiuje się jako ciągły podzbiór komórek o tych samych wartościach dla wymiarów w bieżącej kolejności sortowania. Międzywierszowe funkcje wykresu są obliczane w segmencie kolumny z pominięciem skrajnego prawego wymiaru w równoważnym wykresie tabeli prostej. Jeśli wykres ma tylko jeden wymiar lub jeśli podano kwalifikator TOTAL, wartość wyrażenia jest obliczana na pełnej tabeli.

Jeśli tabela lub równoważnik tabeli zawiera wiele wymiarów pionowych, wówczas segment bieżącej kolumny będzie zawierał tylko wiersze z takimi samymi wartościami we wszystkich kolumnach wymiaru jak bieżący wiersz, ale bez kolumny przedstawiającej ostatni wymiar w kolejności sortowania między polami.

Ograniczenia:

- Wywołania rekurencyjne będą zwracać NULL.
- Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji.

Przykłady i wyniki:

Przykład: 1

Na zrzucie ekranu w tym przykładzie widoczne jest, że stosowna wizualizacja tabeli została utworzona na podstawie wymiaru **Customer** oraz miar `Sum(Sales)` i `Top(Sum(Sales))`.

Kolumna **Top(Sum(Sales))** zwraca wartość 587 dla wszystkich wierszy, ponieważ jest to wartość najwyższego wiersza: **Astrida**.

W tabeli tej przedstawione są również bardziej złożone miary: jedna utworzona na podstawie wyrażenia `Sum(Sales)+Top(Sum(Sales))` i jedna zatytułowana **Top offset 3**, utworzona na podstawie wyrażenia `Sum(Sales)+Top(Sum(Sales), 3)` z argumentem **offset** o wartości 3. W efekcie jej działania wartość **Sum (Sales)** dla bieżącego wiersza jest dodawana do wartości z trzeciego wiersza liczonego od najwyższego wiersza (wartość z wiersza bieżącego plus wartość dla klienta **Canutility**).

Przykład 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

Przykład: 2

Na zrzutach ekranu tabel przedstawionych w tym przykładzie do wizualizacji zostały dodane kolejne wymiary: **Month** i **Product**. W przypadku wykresów wielowymiarowych wyniki wyrażeń zawierających funkcje **Above**, **Below**, **Top** i **Bottom** są zależne od kolejności sortowania wymiarów kolumn przez Qlik Sense. Qlik Sense oblicza wartości funkcji na podstawie segmentów kolumny uzyskanych z wymiaru sortowanego jako ostatni. Kolejność sortowania kolumn określa się w panelu właściwości w sekcji **Sortowanie** i może ona być inna od kolejności wyświetlania kolumn w tabeli.

Pierwsza tabela z przykładu 2. Wartość funkcji Top dla miary First value na podstawie wymiaru Month (Jan).

Customer	Product	Month	Sum(Sales)	First value
			2566	-
Astrida	AA	Jan	46	46
Astrida	AA	Feb	60	46
Astrida	AA	Mar	70	46
Astrida	AA	Apr	13	46
Astrida	AA	May	78	46
Astrida	AA	Jun	20	46
Astrida	AA	Jul	45	46
Astrida	AA	Aug	65	46
Astrida	AA	Sep	78	46
Astrida	AA	Oct	12	46
Astrida	AA	Nov	78	46
Astrida	AA	Dec	22	46

Druga tabela z przykładu 2. Wartość funkcji Top dla miary First value na podstawie wymiaru Product (AA dla klienta Astrida).

Customer	Product	Month	Sum(Sales)	First value
			2566	-
Astrida	AA	Jan	46	46
Astrida	BB	Jan	46	46
Astrida	AA	Feb	60	60
Astrida	BB	Feb	60	60
Astrida	AA	Mar	70	70
Astrida	BB	Mar	70	70
Astrida	AA	Apr	13	13
Astrida	BB	Apr	13	13

Więcej informacji na ten temat zawiera przykład 2 w funkcji **Above**.

Przykład 3

Przykład: 3	**Wynik**
Funkcję **Top** można stosować jako źródło danych wejściowych dla funkcji zakresu. Na przykład: `RangeAvg (Top(Sum(Sales),1,3)).`	W funkcji **Top()** argument offset ma wartość 1, a argument count ma wartość 3. Funkcja ta znajduje wyniki wyrażenia **Sum(Sales)** w trzech wierszach począwszy od wiersza pod najwyższym wierszem w segmencie kolumny (ponieważ offset=1), oraz i w dwóch wierszach pod nim (o ile jest tam wiersz). Wspomniane trzy wartości są stosowane jako dane wejściowe dla funkcji RangeAvg(), która znajduje średnią wartości w określonym szeregu liczbowym. Tabela z kolumną **Customer** jako wymiarem zwraca następujące wyniki dla wyrażenia RangeAvg().
	Astrida 603 Betacab 603 Canutility 603 Divadip: 603

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
```

```
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Zob. także:

p *Bottom – funkcja wykresu (page 1262)*
p *Above – funkcja wykresu (page 1253)*
p *Sum – funkcja wykresu (page 338)*
p *RangeAvg (page 1311)*
p *Funkcje zakresu (page 1308)*

SecondaryDimensionality – funkcja wykresu

Funkcja **SecondaryDimensionality()** zwraca liczbę wierszy wymiaru tabeli przestawnej, które zawierają treść niepodlegającą agregacji, czyli nie zawierają sum częściowych ani zwiniętych agregacji. Ta funkcja jest odpowiednikiem funkcji **dimensionality()** dla poziomych wymiarów tabeli przestawnej.

Składnia:

```
SecondaryDimensionality( )
```

Typ zwracanych danych: integer

Ograniczenia:

- Funkcja **SecondaryDimensionality** zawsze zwraca 0, chyba że jest używana w tabelach przestawnych.
- Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji.

After – funkcja wykresu

Funkcja **After()** zwraca wartość wyrażenia obliczoną na podstawie wartości wymiaru tabeli przestawnej występujących w kolumnie znajdującej się za kolumną bieżącą w ramach segmentu wierszy tabeli przestawnej.

Składnia:

```
after([TOTAL] expr [, offset [, count ]])
```

Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji.

Ta funkcja zwraca wartość NULL we wszystkich typach wykresów z wyjątkiem tabel przestawnych.

Argumenty:

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
offset	Określenie wartości **offset** n większej niż 1 umożliwia przeniesienie oceny wyrażenia o n wierszy w prawo od bieżącego wiersza. Podanie argumentu offset równego 0 spowoduje ocenę wartości wyrażenia w bieżącym wierszu. Określenie ujemnej wartości argumentu offset sprawia, że funkcja **After** działa tak samo jak funkcja **Before** z odpowiednią dodatnią wartością argumentu offset.
count	Jeśli trzeci parametr **count** będzie większy od 1, funkcja zwróci zakres wartości, po jednej dla każdego z wierszy tabeli do wartości **count**, liczonych w prawo od pierwotnej komórki.
TOTAL	Jeśli tabela jest jednowymiarowa lub jako argument zostanie podany kwalifikator **TOTAL**, bieżący segment kolumny jest zawsze równy całej kolumnie.

W przypadku ostatniej kolumny segmentu wiersza zostanie zwrócona wartość NULL, ponieważ za tą kolumną nie ma żadnej kolumny.

Jeśli tabela przestawna zawiera wiele wymiarów poziomych, wówczas bieżący segment wiersza będzie zawierać tylko kolumny z takimi samymi wartościami, co bieżąca kolumna we wszystkich wierszach wymiaru, z wyjątkiem wiersza przedstawiającego ostatni wymiar poziomy w kolejności sortowania między polami. Kolejność sortowania między polami dla wymiarów poziomych w tabelach przestawnych jest zdefiniowana przez kolejność wymiarów od góry do dołu..

Przykład:

```
after( sum( Sales ))
after( sum( Sales ), 2 )
after( total sum( Sales ))
```

Wyrażenie `rangeavg (after(sum(x),1,3))` zwraca średnią z trzech wyników funkcji **sum(x)** ocenianej w trzech kolumnach znajdujących się bezpośrednio po prawej stronie kolumny bieżącej.

Before – funkcja wykresu

Funkcja **Before()** zwraca wartość wyrażenia obliczoną na podstawie wartości wymiaru tabeli przestawnej występujących w kolumnie znajdującej się przed kolumną bieżącą w ramach segmentu wierszy tabeli przestawnej.

Składnia:

```
before([TOTAL] expr [, offset [, count]])
```

Ta funkcja zwraca wartość NULL we wszystkich typach wykresów z wyjątkiem tabel przestawnych.

Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji.

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
offset	Określenie wartości **offset** n większej niż 1 umożliwia przeniesienie oceny wyrażenia o n wierszy w lewo od bieżącego wiersza. Podanie argumentu offset równego 0 spowoduje ocenę wartości wyrażenia w bieżącym wierszu. Określenie ujemnej wartości argumentu offset sprawia, że funkcja **Before** działa tak samo jak funkcja **After** z odpowiednią dodatnią wartością argumentu offset.
count	Jeśli trzeci parametr **count** będzie większy od 1, funkcja zwróci zakres wartości, po jednej dla każdego z wierszy tabeli do wartości **count**, liczonych w lewo od pierwotnej komórki.
TOTAL	Jeśli tabela jest jednowymiarowa lub jako argument zostanie podany kwalifikator **TOTAL**, bieżący segment kolumny jest zawsze równy całej kolumnie.

W przypadku pierwszej kolumny segmentu wiersza zostanie zwrócona wartość NULL, ponieważ tej kolumny nie poprzedza żadna kolumna.

Jeśli tabela przestawna zawiera wiele wymiarów poziomych, wówczas bieżący segment wiersza będzie zawierać tylko kolumny z takimi samymi wartościami, co bieżąca kolumna we wszystkich wierszach wymiaru, z wyjątkiem wiersza przedstawiającego ostatni wymiar poziomy w kolejności sortowania między polami.Kolejność sortowania między polami dla wymiarów poziomych w tabelach przestawnych jest zdefiniowana przez kolejność wymiarów od góry do dołu..

Przykłady:

```
before( sum( Sales ))
before( sum( Sales ), 2 )
before( total sum( Sales ))
```

Wyrażenie `rangeavg (before(sum(x),1,3))` zwraca średnią z trzech wyników funkcji **sum(x)** ocenianej w trzech kolumnach znajdujących się bezpośrednio po lewej stronie kolumny bieżącej.

First – funkcja wykresu

Funkcja **First()** zwraca wartość wyrażenia obliczoną na podstawie wartości wymiaru tabeli przestawnej występujących w pierwszej kolumnie bieżącego segmentu wierszy tabeli przestawnej. Ta funkcja zwraca wartość NULL we wszystkich typach wykresów z wyjątkiem tabel przestawnych.

Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji.

Składnia:

```
first([TOTAL] expr [, offset [, count]])
```

Argumenty:

Argumenty

Argument	Opis
expression	Wyrażenie lub pole zawierające mierzone dane.
offset	Określenie wartości **offset** n większej niż 1 umożliwia przeniesienie oceny wyrażenia o n wierszy w prawo od bieżącego wiersza. Podanie argumentu offset równego 0 spowoduje ocenę wartości wyrażenia w bieżącym wierszu. Określenie ujemnej wartości argumentu offset sprawia, że funkcja **First** działa tak samo jak funkcja **Last** z odpowiednią dodatnią wartością argumentu offset.
count	Jeśli trzeci parametr **count** będzie większy od 1, funkcja zwróci zakres wartości, po jednej dla każdego z wierszy tabeli do wartości **count**, liczonych w prawo od pierwotnej komórki.

Argument	Opis
TOTAL	Jeśli tabela jest jednowymiarowa lub jako argument zostanie podany kwalifikator **TOTAL**, bieżący segment kolumny jest zawsze równy całej kolumnie.

Jeśli tabela przestawna zawiera wiele wymiarów poziomych, wówczas bieżący segment wiersza będzie zawierać tylko kolumny z takimi samymi wartościami, co bieżąca kolumna we wszystkich wierszach wymiaru, z wyjątkiem wiersza przedstawiającego ostatni wymiar poziomy w kolejności sortowania między polami.Kolejność sortowania między polami dla wymiarów poziomych w tabelach przestawnych jest zdefiniowana przez kolejność wymiarów od góry do dołu..

Przykłady:

```
first( sum( Sales ))
first( sum( Sales ), 2 )
first( total sum( Sales )
```

`rangeavg (first(sum(x),1,5))` zwraca średnią z wyników funkcji **sum(x)** ocenianej w pięciu kolumnach skrajnie po lewej stronie bieżącego segmentu wiersza.

Last – funkcja wykresu

Funkcja **Last()** zwraca wartość wyrażenia obliczoną na podstawie wartości wymiaru tabeli przestawnej występujących w ostatniej kolumnie bieżącego segmentu wierszy tabeli przestawnej. Ta funkcja zwraca wartość NULL we wszystkich typach wykresów z wyjątkiem tabel przestawnych.

Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji.

Składnia:

```
last([TOTAL] expr [, offset [, count]])
```

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.

Argument	Opis
offset	Określenie wartości **offset** n większej niż 1 umożliwia przeniesienie oceny wyrażenia o n wierszy w lewo od bieżącego wiersza. Podanie argumentu offset równego 0 spowoduje ocenę wartości wyrażenia w bieżącym wierszu. Określenie ujemnej wartości argumentu offset sprawia, że funkcja **First** działa tak samo jak funkcja **Last** z odpowiednią dodatnią wartością argumentu offset.
count	Jeśli trzeci parametr **count** będzie większy od 1, funkcja zwróci zakres wartości, po jednej dla każdego z wierszy tabeli do wartości **count**, liczonych w lewo od pierwotnej komórki.
TOTAL	Jeśli tabela jest jednowymiarowa lub jako argument zostanie podany kwalifikator **TOTAL**, bieżący segment kolumny jest zawsze równy całej kolumnie.

Jeśli tabela przestawna zawiera wiele wymiarów poziomych, wówczas bieżący segment wiersza będzie zawierać tylko kolumny z takimi samymi wartościami, co bieżąca kolumna we wszystkich wierszach wymiaru, z wyjątkiem wiersza przedstawiającego ostatni wymiar poziomy w kolejności sortowania między polami.Kolejność sortowania między polami dla wymiarów poziomych w tabelach przestawnych jest zdefiniowana przez kolejność wymiarów od góry do dołu..

Przykład:

```
last( sum( Sales ))
last( sum( Sales ), 2 )
last( total sum( Sales )
```

`rangeavg (last(sum(x),1,5))` zwraca średnią z wyników funkcji **sum(x)** ocenianej w pięciu kolumnach skrajnie po prawej stronie bieżącego segmentu wiersza.

ColumnNo – funkcja wykresu

Funkcja **ColumnNo()** zwraca numer bieżącej kolumny w bieżącym segmencie wierszy tabeli przestawnej. Pierwsza kolumna ma numer 1.

Składnia:

```
ColumnNo([total])
```

Argumenty:

Argumenty

Argument	Opis
TOTAL	Jeśli tabela jest jednowymiarowa lub jako argument zostanie podany kwalifikator **TOTAL**, bieżący segment kolumny jest zawsze równy całej kolumnie.

Jeśli tabela przestawna zawiera wiele wymiarów poziomych, wówczas bieżący segment wiersza będzie zawierać tylko kolumny z takimi samymi wartościami, co bieżąca kolumna we wszystkich wierszach wymiaru, z wyjątkiem wiersza przedstawiającego ostatni wymiar poziomy w kolejności sortowania między polami.Kolejność sortowania między polami dla wymiarów poziomych w tabelach przestawnych jest zdefiniowana przez kolejność wymiarów od góry do dołu..

Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji.

Przykład:

```
if( ColumnNo( )=1, 0, sum( Sales ) / before( sum( Sales )))
```

NoOfColumns – funkcja wykresu

Funkcja **NoOfColumns()** zwraca liczbę kolumn w bieżącym segmencie wierszy tabeli przestawnej.

Sortowanie według wartości Y w wykresach albo sortowanie według kolumn wyrażeń w tabelach jest niedozwolone, jeśli w dowolnym z wyrażeń wykresu używana jest ta funkcja wykresu. W takiej sytuacji te opcje sortowania są automatycznie wyłączone. Gdy użyjesz tej funkcji wykresu w wizualizacji lub tabeli, sortowanie wizualizacji powróci do posortowanych danych wejściowych tej funkcji.

Składnia:

```
NoOfColumns([total])
```

Argumenty:

Argumenty

Argument	Opis
TOTAL	Jeśli tabela jest jednowymiarowa lub jako argument zostanie podany kwalifikator **TOTAL**, bieżący segment kolumny jest zawsze równy całej kolumnie.

Jeśli tabela przestawna zawiera wiele wymiarów poziomych, wówczas bieżący segment wiersza będzie zawierać tylko kolumny z takimi samymi wartościami, co bieżąca kolumna we wszystkich wierszach wymiaru, z wyjątkiem wiersza przedstawiającego ostatni wymiar poziomy w kolejności sortowania między polami.Kolejność sortowania między polami dla wymiarów poziomych w tabelach przestawnych jest zdefiniowana przez kolejność wymiarów od góry do dołu..

Przykład:

```
if( ColumnNo( )=NoOfColumns( ), 0, after( sum( Sales )))
```

5.17 Funkcje logiczne

W tej sekcji opisano funkcje obsługujące operacje logiczne. Wszystkie funkcje mogą być stosowane zarówno w skryptach ładowania danych, jak i wyrażeniach wykresu.

IsNum

Zwraca wartość -1 (True), jeśli wyrażenie można zinterpretować jako liczbę, a 0 (False) w przeciwnym przypadku.

```
IsNum( expr )
```

IsText

Zwraca -1 (True), jeśli wyrażenie ma reprezentację tekstową, a 0 (False) w przeciwnym przypadku.

```
IsText( expr )
```

*Zarówno **IsNum**, jak i **IsText** zwracają 0, gdy wyrażenie to NULL.*

Przykład:

W poniższym przykładzie ładowana jest tabela wbudowana z mieszanymi wartościami tekstowymi i liczbowymi oraz dodawane są dwa pola celem sprawdzenia, czy wartość jest odpowiednio wartością liczbową czy tekstową.

```
Load *, IsNum(Value), IsText(Value)
Inline [
Value
23
Green
Blue
12
33Red];
```

Otrzymana tabela wygląda następująco:

Resulting table

Value	IsNum(Value)	IsText(Value)
23	-1	0
Green	0	-1
Blue	0	-1
12	-1	0
33Red	0	-1

5.18 Funkcje mapowania

W tej sekcji opisano funkcje obsługujące tabele mapowania. Tabele mapowania można wykorzystać do zastępowania wartości pól lub nazw pól podczas wykonywania skryptu.

Funkcje mapowania mogą być używane tylko w skrypcie ładowania danych.

Przegląd funkcji mapowania

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

ApplyMap

Funkcja skryptu **ApplyMap** służy do mapowania danych wyjściowych wyrażenia na wcześniej załadowaną tabelę mapowania.

```
ApplyMap ('mapname', expr [ , defaultexpr ] )
```

MapSubstring

Funkcja skryptu **MapSubstring** umożliwia mapowanie części dowolnego wyrażenia według wcześniej załadowanej tabeli mapowania. Mapowanie uwzględnia wielkość liter i nie jest iteracyjne. Ciągi podrzędne są mapowane od lewej do prawej.

```
MapSubstring ('mapname', expr)
```

ApplyMap

Funkcja skryptu **ApplyMap** służy do mapowania danych wyjściowych wyrażenia na wcześniej załadowaną tabelę mapowania.

Składnia:

```
ApplyMap('map_name', expression [ , default_mapping ] )
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
map_name	Nazwa tabeli mapowania, która została utworzona przy użyciu instrukcji **mapping load** lub **mapping select**. Nazwa musi być ujęta w pojedyncze proste cudzysłowy.  *Jeśli ta funkcja zostanie użyta w zmiennej rozszerzonej przez makro i zostanie utworzone odniesienie do tabeli mapowania, która nie istnieje, wówczas wywołanie funkcji nie powiedzie się, a pole nie zostanie utworzone.*
expression	Wyrażenie, którego wynik powinien być mapowany.
default_ mapping	W przypadku określenia tej wartości będzie ona używana jako wartość domyślna, jeśli tabela mapowania nie zawiera pasującej wartości wyrażenia expression. Jeśli nie zostanie określona, wartość expression zostanie zwrócona w istniejącej formie.

 *Pole wyjściowe funkcji ApplyMap nie powinno mieć tej samej nazwy, jak jedno z jej pól wejściowych. To może spowodować nieoczekiwane wyniki. Przykład nieprzeznaczony do użycia:* `ApplyMap('Map', A) as A`.

Przykład:

W tym przykładzie ładowana jest lista sprzedawców z kodem kraju reprezentującym ich kraj zamieszkania. Używana jest tabela mapująca kod kraju na kraj, aby zastąpić kod kraju jego nazwą. W tabeli mapowania zdefiniowano tylko trzy kraje, inne kody krajów są mapowane na wartość 'Rest of the world'.

```
// Load mapping table of country codes:
map1:
mapping LOAD *
Inline [
CCode, Country
Sw, Sweden
Dk, Denmark
No, Norway
] ;

// Load list of salesmen, mapping country code to country
// If the country code is not in the mapping table, put Rest of the world
Salespersons:
LOAD *,
ApplyMap('map1', CCode,'Rest of the world') As Country
Inline [
CCode, Salesperson
```

```
Sw, John
Sw, Mary
Sw, Per
Dk, Preben
Dk, Olle
No, Ole
Sf, Risttu
] ;

// We don't need the CCode anymore
Drop Field 'CCode';
```

Tabela wynikowa (Salespersons) wygląda następująco:

Resulting table

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

MapSubstring

Funkcja skryptu **MapSubstring** umożliwia mapowanie części dowolnego wyrażenia według wcześniej załadowanej tabeli mapowania. Mapowanie uwzględnia wielkość liter i nie jest iteracyjne. Ciągi podrzędne są mapowane od lewej do prawej.

Składnia:

```
MapSubstring('map_name', expression)
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
map_name	Nazwa tabeli mapowania, która została poprzednio odczytana w instrukcji **mapping load** lub **mapping select**. Nazwa musi być otoczona pojedynczymi prostymi cudzysłowami. *Jeśli ta funkcja zostanie użyta w zmiennej rozszerzonej przez makro i zostanie utworzone odniesienie do tabeli mapowania, która nie istnieje, wówczas wywołanie funkcji nie powiedzie się, a pole nie zostanie utworzone.*
expression	Wyrażenie, którego wynik będzie mapowany według fragmentów tekstu.

Przykład:

Na tym przykładzie ładujemy listę modeli produktu. Każdy model zawiera zestaw atrybutów opisanych przez kod złożony. Przy użyciu tabeli mapowania z MapSubstring można rozwinąć kody atrybutów do opisu.

```
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';
```

Wynikowa tabela wygląda następująco:

Resulting table

Model	Description
Twixie	Red Cotton Small
Boomer	Blue Polyester Large
Raven	Yellow Polyester Medium
Seedling	Red Cotton Large
SeedlingPlus	Red Cotton Large with hood
Younger	Blue Cotton with patch
MultiStripe	Red Yellow Blue Cotton Small/Medium/Large

5.19 Funkcje matematyczne

W tej sekcji opisano funkcje dotyczące stałych matematycznych i wartości logicznych. Funkcje te nie zawierają parametrów, ale wciąż wymagane są nawiasy.

Wszystkie funkcje mogą być stosowane zarówno w skryptach ładowania danych, jak i wyrażeniach wykresu.

e

Funkcja zwraca podstawę logarytmów naturalnych, **e** (2,71828...).

```
e( )
```

false

Funkcja zwraca wartość podwójną o wartości tekstowej 'False' i wartości liczbowej 0. Wyniku można używać w wyrażeniach jako fałszu logicznego.

```
false( )
```

pi

Funkcja zwraca wartość π (3,14159...).

```
pi( )
```

rand

Funkcja zwraca losową liczbę z zakresu od 0 do 1. Można jej użyć do tworzenia przykładowych danych.

```
rand( )
```

Przykład:

Ten przykładowy skrypt tworzy tabelę tysiąca rekordów z losowo wybranymi wielkimi literami, to znaczy znakami mieszczącymi się w zakresie od 65 do 91 (65+26).

```
Load
     Chr( Floor(rand() * 26) + 65) as UCaseChar,
     RecNo() as ID
     Autogenerate 1000;
```

true
Funkcja zwraca wartość podwójną o wartości tekstowej 'True' i wartości liczbowej -1. Wyniku można używać w wyrażeniach jako prawdy logicznej.

```
true( )
```

5.20 Funkcje NULL

W tej sekcji opisano funkcje dotyczące zwracania lub wykrywania wartości NULL.

Wszystkie funkcje mogą być stosowane zarówno w skryptach ładowania danych, jak i wyrażeniach wykresu.

Przegląd funkcji NULL

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

EmptyIsNull
Funkcja **EmptyIsNull** konwertuje puste ciągi na NULL. W związku z tym zwraca NULL, jeśli parametr jest pustym ciągiem. W przeciwnym razie zwraca parametr.

```
EmptyIsNull (expr )
```

IsNull
Funkcja **IsNull** sprawdza, czy wyrażenie ma wartość NULL i w takim wypadku zwraca wartość -1 (True). W przeciwnym razie zwracana jest wartość 0 (False).

```
IsNull (expr )
```

Null
Funkcja **Null** zwraca wartość NULL.

```
NULL( )
```

EmptyIsNull

Funkcja **EmptyIsNull** konwertuje puste ciągi na NULL. W związku z tym zwraca NULL, jeśli parametr jest pustym ciągiem. W przeciwnym razie zwraca parametr.

Składnia:

```
EmptyIsNull(exp )
```

Przykłady i wyniki:

Przykłady skryptów

Przykład	Wynik
`EmptyIsNull(AdditionalComments)`	To wyrażenie zwróci wszystkie puste wartości ciągu pola *AdditionalComments* jako null zamiast pustych ciągów. Zwracane są niepuste ciągi i liczby.
`EmptyIsNull(PurgeChar (PhoneNumber, ' -()'))`	To wyrażenie usunie wszelkie myślniki, spacje i nawiasy z pola *PhoneNumber*. Jeśli nie pozostaną już żadne znaki, funkcja EmptyIsNull zwraca pusty ciąg jako null. Pusty numer telefonu jest tym samym co brak numeru telefonu.

IsNull

Funkcja **IsNull** sprawdza, czy wyrażenie ma wartość NULL i w takim wypadku zwraca wartość -1 (True). W przeciwnym razie zwracana jest wartość 0 (False).

Składnia:

```
IsNull(expr )
```

*Ciąg znaków o zerowej długości nie jest uznawany za NULL i spowoduje zwrócenie przez funkcję **IsNull** wartości False.*

Przykład: Skrypt ładowania danych

W tym przykładzie ładowana jest tabela wbudowana z czterema wierszami, w której pierwsze trzy linie nie zawierają nic albo zawierają wartość - albo 'NULL' w kolumnie Value. Przekształcamy te wartości na rzeczywiste reprezentacje wartości NULL przy użyciu środkowej wartości poprzedzającej instrukcję **LOAD**, za pomocą funkcji **Null**.

Pierwsza wartość poprzedzająca instrukcję **LOAD** dodaje pole sprawdzające, czy jest to wartość NULL, przy użyciu funkcji **IsNull**.

```
NullsDetectedAndConverted:

LOAD *,
If(IsNull(ValueNullConv), 'T', 'F') as IsItNull;

LOAD *,
If(len(trim(Value))= 0 or Value='NULL' or Value='-', Null(), Value ) as ValueNullConv;

LOAD * Inline
[ID, Value
0,
```

```
1,NULL
2,-
3,Value];
```

Oto otrzymana tabela. W kolumnie ValueNullConv wartości NULL są reprezentowane przez -.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

NULL

Funkcja **Null** zwraca wartość NULL.

Składnia:

```
Null( )
```

Przykład: Skrypt ładowania danych

W tym przykładzie ładowana jest tabela wbudowana z czterema wierszami, w której pierwsze trzy linie nie zawierają nic albo zawierają wartość - albo 'NULL' w kolumnie Value. Chcemy przekształcić te wartości na rzeczywiste reprezentacje wartości NULL.

Środkowa poprzedzająca instrukcję **LOAD** wykonuje przekształcenie przy użyciu funkcji **Null**.

Pierwsza poprzedzająca instrukcję **LOAD** dodaje pole sprawdzające, czy wartość wynosi NULL, w tym przykładzie tylko do celów ilustracyjnych.

```
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];
```

Oto otrzymana tabela. W kolumnie ValueNullConv wartości NULL są reprezentowane przez -.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

5.21 Funkcje zakresu

Funkcje zakresu to funkcje, które pobierają szereg wartości i zwracają wartość pojedynczą. Wszystkie funkcje zakresu mogą być stosowane zarówno w skryptach ładowania danych, jak i wyrażeniach wykresów.

Na przykład w przypadku wizualizacji funkcja zakresu może posłużyć do obliczenia wartości pojedynczej na podstawie szeregu międzywierszowego. W przypadku skryptu ładowania danych funkcja zakresu może posłużyć do obliczenia wartości pojedynczej na podstawie szeregu wartości w tabeli wewnętrznej.

*Funkcje zakresu zastępują następujące ogólne funkcje liczbowe: **numsum**,**numavg**, **numcount**,**nummin** i **nummax**, które powinny być teraz traktowane jako niepotrzebne.*

Podstawowe funkcje zakresu

RangeMax
Funkcja **RangeMax()** zwraca najwyższe wartości liczbowe znalezione w wyrażeniu lub polu.

```
RangeMax (first_expr[, Expression])
```

RangeMaxString
Funkcja **RangeMaxString()** zwraca ostatnią z posortowanych tekstowo wartości znalezionych w wyrażeniu lub polu.

```
RangeMaxString (first_expr[, Expression])
```

RangeMin
Funkcja **RangeMin()** zwraca najniższe wartości liczbowe znalezione w wyrażeniu lub polu.

```
RangeMin (first_expr[, Expression])
```

RangeMinString
Funkcja **RangeMinString()** zwraca pierwszą z posortowanych tekstowo wartości znalezionych w wyrażeniu lub polu.

```
RangeMinString (first_expr[, Expression])
```

RangeMode
Funkcja **RangeMode()** zwraca wartość najczęściej występującą w wyrażeniu lub polu (wartość modalną).

```
RangeMode (first_expr[, Expression])
```

RangeOnly

RangeOnly() to funkcja podwójna zwracająca wartość, jeśli wynikiem wyrażenia jest dokładnie jedna, unikatowa wartość. W przeciwnym razie zwracana jest wartość **NULL**.

```
RangeOnly (first_expr[, Expression])
```

RangeSum

RangeSum() zwraca sumę zakresu wartości. Wszystkie wartości nienumeryczne są traktowane jako 0.

```
RangeSum (first_expr[, Expression])
```

Licznikowe funkcje zakresu

RangeCount

Funkcja **RangeCount()** zwraca liczbę wartości tekstowych i liczbowych w wyrażeniu lub polu.

```
RangeCount (first_expr[, Expression])
```

RangeMissingCount

Funkcja **RangeMissingCount()** zwraca liczbę wartości nieliczbowych (w tym wartości NULL) w wyrażeniu lub polu.

```
RangeMissingCount (first_expr[, Expression])
```

RangeNullCount

Funkcja **RangeNullCount()** zwraca liczbę wartości NULL w wyrażeniu lub polu.

```
RangeNullCount (first_expr[, Expression])
```

RangeNumericCount

Funkcja **RangeNumericCount()** zwraca liczbę wartości liczbowych w wyrażeniu lub polu.

```
RangeNumericCount (first_expr[, Expression])
```

RangeTextCount

Funkcja **RangeTextCount()** zwraca liczbę wartości tekstowych w wyrażeniu lub polu.

```
RangeTextCount (first_expr[, Expression])
```

Statystyczne funkcje zakresu

RangeAvg

Funkcja **RangeAvg()** zwraca średnią wartość z zakresu. Argumentem wejściowym funkcji może być zakres wartości lub wyrażenie.

```
RangeAvg (first_expr[, Expression])
```

RangeCorrel

Funkcja **RangeCorrel()** zwraca współczynnik korelacji dwóch zestawów danych. Współczynnik korelacji to miara stopnia powiązania między zestawami danych.

```
RangeCorrel (x_values , y_values[, Expression])
```

RangeFractile
Funkcja **RangeFractile()** zwraca wartość odpowiadającą n-temu kwantylowi (**fractile**) z zakresu liczb.

```
RangeFractile (fractile, first_expr[ ,Expression])
```

RangeKurtosis
Funkcja **RangeKurtosis()** zwraca wartość odpowiadającą kurtozie zakresu liczb.

```
RangeKurtosis (first_expr[, Expression])
```

RangeSkew
Funkcja **RangeSkew()** zwraca wartość odpowiadającą skośności zakresu liczb.

```
RangeSkew (first_expr[, Expression])
```

RangeStdev
Funkcja **RangeStdev()** określa odchylenie standardowe zakresu liczb.

```
RangeStdev (expr1[, Expression])
```

Finansowe funkcje zakresu

RangeIRR

Funkcja **RangeIRR()** zwraca wewnętrzny współczynnik zwrotu dla serii przepływów pieniężnych reprezentowanych przez wartości wejściowe.

```
RangeIRR (value[, value][, Expression])
```

RangeNPV

Funkcja **RangeNPV()** zwraca wartość bieżącą netto inwestycji na podstawie stopy dyskontowej oraz serii przyszłych okresowych płatności (wartości ujemnych) i dochodów (wartości dodatnich). Wynik ma następujący domyślny format liczby: **money**.

```
RangeNPV (discount_rate, value[, value][, Expression])
```

RangeXIRR

Funkcja **RangeXIRR()** zwraca wewnętrzną stopę zwrotu dla zaplanowanych przepływów pieniężnych, niekoniecznie okresowych. Do obliczania wewnętrznej stopy zwrotu dla serii okresowych przepływów pieniężnych należy używać funkcji **RangeIRR**.

```
RangeXIRR (values, dates[, Expression])
```

RangeXNPV

Współczynnik **RangeXNPV()** zwraca wartość bieżącą netto zaplanowanych przepływów pieniężnych, niekoniecznie okresowych. Wynik ma następujący domyślny format liczby: waluta. Do obliczania bieżącej wartości netto serii okresowych przepływów pieniężnych należy używać funkcji **RangeNPV**.

```
RangeXNPV (discount_rate, values, dates[, Expression])
```

Zob. także:

p *Funkcje międzywierszowe (page 1249)*

RangeAvg

Funkcja **RangeAvg()** zwraca średnią wartość z zakresu. Argumentem wejściowym funkcji może być zakres wartości lub wyrażenie.

Składnia:

```
RangeAvg(first_expr[, Expression])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty tej funkcji mogą zawierać funkcje międzywierszowe, które same zwracają listę wartości.

Argumenty

Argument	Opisu
first_expr	Wyrażenie lub pole zawierające mierzone dane.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Ograniczenia:

Jeśli nie zostanie znaleziona żadna wartość liczbowa, zostanie zwrócona wartość NULL.

Przykłady i wyniki:

Przykłady skryptów

Przykłady	Wyniki
`RangeAvg (1,2,4)`	Zwraca wartość 2,33333333
`RangeAvg (1,'xyz')`	Zwraca wartość 1
`RangeAvg (null( ), 'abc')`	Zwraca wartość NULL

Przykład:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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
];
```

Tabela wynikowa przedstawia zwrócone wartości MyRangeAvg dla każdego z rekordów w tabeli.

Tabela wynikowa

RangeID	MyRangeAvg
1	7
2	4
3	6
4	12.666
5	6.333
6	5

Przykład z wyrażeniem:

```
RangeAvg (Above(MyField),0,3))
```

Zwraca średnią kroczącą wyniku zakresu trzech wartości z kolumny **MyField** obliczaną dla bieżącego wiersza oraz dwóch wierszy powyżej niego. Jeśli trzeci argument ma wartość 3, funkcja **Above()** zwraca trzy wartości (o ile istnieje wystarczająca liczba wierszy powyżej), które stanowią dane wejściowe dla funkcji **RangeAvg()**.

Dane zastosowane w przykładach:

*Aby przykład działał zgodnie z oczekiwaniami, wyłącz sortowanie pola **MyField**.*

Dane przykładowe

MyField	RangeAvg (Above (MyField,0,3))	Comments
10	10	Jest to najwyższy wiersz, zakres obejmuje tylko jedną wartość.
2	6	Nad tym wierszem jest tylko jeden wiersz, zakres jest zatem następujący: 10,2.
8	6.6666666667	Odpowiednik funkcji RangeAvg(10,2,8)
18	9.333333333	-
5	10. 333333333	-
9	10.6666666667	-

```
RangeTab:
LOAD * INLINE [
MyField
```

```
10
2
8
18
5
9
] ;
```

Zob. także:

p *Avg – funkcja wykresu (page 389)*
p *Count – funkcja wykresu (page 343)*

RangeCorrel

Funkcja **RangeCorrel()** zwraca współczynnik korelacji dwóch zestawów danych. Współczynnik korelacji to miara stopnia powiązania między zestawami danych.

Składnia:

```
RangeCorrel(x_value , y_value[, Expression])
```

Typ zwracanych danych: numeric

Szeregi danych powinny zostać wprowadzone jako pary (x,y). Aby na przykład ocenić dwa szeregi danych, szereg 1 i szereg 2, gdy szereg 1 to 2,6,9, a szereg 2 to 3,8,4, należy zastosować funkcję `RangeCorrel (2,3,6,8,9,4)`, która zwróci wartość 0,269.

Argumenty:

Argumenty

Argument	Opis
x-value, y-value	Każda wartość jest wartością pojedynczą lub zakresem wartości, jaką/jaki zwracają funkcje międzywierszowe z trzecim parametrem opcjonalnym. Każda wartość lub zakres wartości musi odpowiadać wartości **x-value** lub zakresowi wartości **y-values**.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Ograniczenia:

Obliczenie wartości funkcji wymaga podania co najmniej dwóch par współrzędnych.

Wartości tekstowe, wartości NULL i wartości brakujące zwracają wartość NULL.

Przykłady i wyniki:

Przykłady funkcji

Przykłady	Wyniki
`RangeCorrel (2,3,6,8,9,4,8,5)`	Zwraca wartość 0,2492 Ta funkcja może być ładowana w skrypcie lub dodawana do wizualizacji w edytorze wyrażeń.

Przykład:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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 '|');
```

W tabeli, w której wymiarem jest ID1, a miarą jest: RangeCorrel(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6)), funkcja **RangeCorrel()** znajduje wartość **Correl** w zakresie sześciu par x,y, dla każdej z wartości ID1.

Tabela wynikowa

ID1	MyRangeCorrel
01	-0.9517
02	-0.5209
03	-0.5209
04	-0.1599

Przykład:

```
XY:
LOAD recno() as RangeID, * Inline [
X|Y
2|3
6|8
9|4
8|5
](delimiter is '|');
```

W tabeli, w której wymiarem jest RangeID, a miarą jest: RangeCorrel(Below(X,0,4,BelowY,0,4)), funkcja **RangeCorrel()** używa wyników funkcji **Below()**, które z powodu trzeciego argumentu (count) ustawionego na 4, zwracają zakres czterech wartości x-y z ładowanej tabeli XY.

Tabela wynikowa

RangeID	MyRangeCorrel2
01	0.2492
02	-0.9959
03	-1.0000
04	-

Wartość dla RangeID 01 jest taka sama, jak w przypadku ręcznego wprowadzenia ciągu znaków RangeCorrel(2,3,6,8,9,4,8,5). W przypadku innych wartości RangeID serie zwracane przez funkcję Below () są następujące: (6,8,9,4,8,5), (9,4,8,5) i (8,5), przy czym ostatnia seria wywołuje wynik null.

Zob. także:

p *Correl – funkcja wykresu (page 392)*

RangeCount

Funkcja **RangeCount()** zwraca liczbę wartości tekstowych i liczbowych w wyrażeniu lub polu.

Składnia:

```
RangeCount(first_expr[, Expression])
```

Typ zwracanych danych: integer

Argumenty:

Argumenty tej funkcji mogą zawierać funkcje międzywierszowe, które same zwracają listę wartości.

Argumenty

Argument	Opis
first_expr	Wyrażenie lub pole zawierające dane do zliczenia.
Expression	Opcjonalne wyrażenia lub pola zawierające zliczany zakres danych.

Ograniczenia:

Wartości NULL nie są zliczane.

Przykłady i wyniki:

Przykłady funkcji

Przykłady	Wyniki
`RangeCount (1,2,4)`	Zwraca wartość 3

Przykłady	Wyniki
`RangeCount (2,'xyz')`	Zwraca wartość 2
`RangeCount (null( ))`	Zwraca wartość 0
`RangeCount (2,'xyz', null())`	Zwraca wartość 2

Przykład:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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
];
```

Tabela wynikowa przedstawia zwrócone wartości MyRangeCount dla każdego z rekordów w tabeli.

Tabela wynikowa

RangeID	MyRangeCount
1	3
2	3
3	3
4	3
5	3
6	3

Przykład z wyrażeniem:

```
RangeCount (Above(MyField,1,3))
```

Zwraca liczbę wartości zawartych w trzech wynikach z kolumny **MyField**. Jeśli pierwszy argument funkcji **Above()** będzie mieć wartość 1, a drugi argument wartość 3, funkcja zwróci wartości z trzech pierwszych pól ponad bieżącym wierszem (o ile istnieje wystarczająca liczba wierszy powyżej), które stanowią dane wejściowe dla funkcji **RangeCount()**.

Dane zastosowane w przykładach:

Dane przykładowe

MyField	RangeCount(Above(MyField,1,3))
10	0
2	1
8	2
18	3
5	3
9	3

Dane zastosowane w przykładach:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Zob. także:

p *Count – funkcja wykresu (page 343)*

RangeFractile

Funkcja **RangeFractile()** zwraca wartość odpowiadającą n-temu kwantylowi (**fractile**) z zakresu liczb.

Funkcja RangeFractile() korzysta z liniowej interpolacji między najbliższymi klasyfikacjami podczas obliczania fraktala.

Składnia:

```
RangeFractile(fractile, first_expr[, Expression])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty tej funkcji mogą zawierać funkcje międzywierszowe, które same zwracają listę wartości.

Argumenty

Argument	Opis
fractile	Liczba z przedziału od 0 do 1 odpowiadająca obliczanemu fraktylowi (kwantylowi wyrażonemu ułamkiem).
first_expr	Wyrażenie lub pole zawierające mierzone dane.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Przykłady i wyniki:

Przykłady funkcji

Przykłady	Wyniki
`RangeFractile (0.24,1,2,4,6)`	Zwraca wartość 1,72
`RangeFractile(0.5,1,2,3,4,6)`	Zwraca wartość 3
`RangeFractile (0.5,1,2,5,6)`	Zwraca wartość 3,5

Przykład:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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
];
```

Tabela wynikowa przedstawia zwrócone wartości MyRangeFrac dla każdego z rekordów w tabeli.

Tabela wynikowa

RangeID	MyRangeFrac
1	6
2	3
3	8
4	11
5	5
6	4

Przykład z wyrażeniem:

```
RangeFractile (0.5, Above(Sum(MyField),0,3))
```

W tym przykładzie funkcja międzywierszowa **Above()** zawiera opcjonalne argumenty offset i count. W efekcie zwracany jest zakres wyników, który może stanowić źródło danych wejściowych dla dowolnej funkcji zakresu. W tym przypadku funkcja `Above(Sum(MyField),0,3)` zwraca wartości z kolumny `MyField` dla bieżącego wiersza i dwóch wierszy powyżej. Wartości te są danymi wejściowymi dla funkcji **RangeFractile()**. W przypadku najniższego wiersza w tabeli poniżej jest ona zatem odpowiednikiem funkcji `RangeFractile(0.5, 3,4,6)`, obliczającą fraktal 0,5 dla szeregu 3, 4 i 6. W przypadku pierwszych dwóch wierszy w poniższej tabeli liczba wartości w zakresie jest stosownie ograniczona, ponieważ nie ma wierszy ponad bieżącym wierszem. Podobne wyniki są zwracane w przypadku innych funkcji międzywierszowych.

Dane przykładowe

MyField	RangeFractile(0.5, Above(Sum(MyField),0,3))
1	1
2	1.5
3	2
4	3
5	4
6	5

Dane zastosowane w przykładach:

```
RangeTab:
LOAD * INLINE [
MyField
1
2
3
4
5
6
] ;
```

Zob. także:

p *Above – funkcja wykresu (page 1253)*
p *Fractile – funkcja wykresu (page 395)*

RangeIRR

Funkcja **RangeIRR()** zwraca wewnętrzny współczynnik zwrotu dla serii przepływów pieniężnych reprezentowanych przez wartości wejściowe.

Wewnętrzna stopa zwrotu to stopa procentowa uzyskiwana dla inwestycji składającej się z płatności (wartości ujemne) i przychodów (wartości dodatnie) występujących w regularnych okresach.

Składnia:

```
RangeIRR(value[, value][, Expression])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
value	Pojedyncza wartość lub zakres wartości, jakie zwraca funkcja międzywierszowa z trzecim parametrem opcjonalnym. Obliczenie funkcji wymaga podania co najmniej jednej wartości dodatniej i jednej ujemnej.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące są pomijane.

Tabela przykładowa

Przykłady	Wyniki
`RangeIRR(-70000,12000,15000,18000,21000,26000)`	Zwraca wartość 0,0866

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.
`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 '|');`
	Tabela wynikowa przedstawia zwrócone wartości RangeIRR dla każdego z rekordów w tabeli.
	RangeID RangeIRR
	1 0.0639
	2 0.8708
	3 -
	4 5.8419
	5 0.9318
	6 -0.2566

Zob. także:

p *Funkcje międzywierszowe (page 1249)*

RangeKurtosis

Funkcja **RangeKurtosis()** zwraca wartość odpowiadającą kurtozie zakresu liczb.

Składnia:

```
RangeKurtosis(first_expr[, Expression])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty tej funkcji mogą zawierać funkcje międzywierszowe, które same zwracają listę wartości.

Argumenty

Argument	Opisu
first_expr	Wyrażenie lub pole zawierające mierzone dane.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Ograniczenia:

Jeśli nie zostanie znaleziona żadna wartość liczbowa, zostanie zwrócona wartość NULL.

Przykłady i wyniki:

Przykłady funkcji

Przykłady	Wyniki
`RangeKurtosis (1,2,4,7)`	Zwraca wartość -0,28571428571429

Zob. także:

p *Kurtosis – funkcja wykresu (page 403)*

RangeMax

Funkcja **RangeMax()** zwraca najwyższe wartości liczbowe znalezione w wyrażeniu lub polu.

Składnia:

```
RangeMax(first_expr[, Expression])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
first_expr	Wyrażenie lub pole zawierające mierzone dane.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Ograniczenia:

Jeśli nie zostanie znaleziona żadna wartość liczbowa, zostanie zwrócona wartość NULL.

Przykłady i wyniki:

Przykłady funkcji

Przykłady	Wyniki
`RangeMax (1,2,4)`	Zwraca wartość 4
`RangeMax (1,'xyz')`	Zwraca wartość 1
`RangeMax (null( ), 'abc')`	Zwraca wartość NULL

Przykład:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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
];
```

Tabela wynikowa przedstawia zwrócone wartości MyRangeMax dla każdego z rekordów w tabeli.

Tabela wynikowa

RangeID	MyRangeMax
1	10
2	7

RangeID	MyRangeMax
3	8
4	18
5	9
6	9

Przykład z wyrażeniem:

```
RangeMax (Above(MyField,0,3))
```

Zwraca wartość maksymalną w zakresie trzech wartości z kolumny **MyField** obliczaną dla bieżącego wiersza oraz dwóch wierszy powyżej niego. Jeśli trzeci argument ma wartość 3, funkcja **Above()** zwraca trzy wartości (o ile istnieje wystarczająca liczba wierszy powyżej), które stanowią dane wejściowe dla funkcji **RangeMax()**.

Dane zastosowane w przykładach:

*Aby przykład działał zgodnie z oczekiwaniami, wyłącz sortowanie pola **MyField**.*

Dane przykładowe

MyField	RangeMax (Above(Sum(MyField),1,3))
10	10
2	10
8	10
18	18
5	18
9	18

Dane zastosowane w przykładach:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

RangeMaxString

Funkcja **RangeMaxString()** zwraca ostatnią z posortowanych tekstowo wartości znalezionych w wyrażeniu lub polu.

Składnia:

```
RangeMaxString(first_expr[, Expression])
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty tej funkcji mogą zawierać funkcje międzywierszowe, które same zwracają listę wartości.

Argumenty

Argument	Opis
first_expr	Wyrażenie lub pole zawierające mierzone dane.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Przykłady i wyniki:

Przykłady funkcji

Przykłady	Wyniki
`RangeMaxString (1,2,4)`	Zwraca wartość 4
`RangeMaxString ('xyz','abc')`	Zwraca wartość 'xyz'
`RangeMaxString (5,'abc')`	Zwraca wartość 'abc'
`RangeMaxString (null( ))`	Zwraca wartość NULL

Przykład z wyrażeniem:

```
RangeMaxString (Above(MaxString(MyField),0,3))
```

Zwraca ostatni (w kolejności sortowania tekstu) z trzech wyników funkcji **MaxString(MyField)** dla bieżącego wiersza oraz dwóch wierszy powyżej niego.

Dane zastosowane w przykładach:

*Aby przykład działał zgodnie z oczekiwaniami, wyłącz sortowanie pola **MyField**.*

Dane przykładowe

MyField	RangeMaxString(Above(MaxString(MyField),0,3))
10	10
abc	abc
8	abc
def	def

MyField	RangeMaxString(Above(MaxString(MyField),0,3))
xyz	xyz
9	xyz

Dane zastosowane w przykładach:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

Zob. także:

p *MaxString – funkcja wykresu (page 524)*

RangeMin

Funkcja **RangeMin()** zwraca najniższe wartości liczbowe znalezione w wyrażeniu lub polu.

Składnia:

```
RangeMin(first_expr[, Expression])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opisu
first_expr	Wyrażenie lub pole zawierające mierzone dane.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Ograniczenia:

Jeśli nie zostanie znaleziona żadna wartość liczbowa, zostanie zwrócona wartość NULL.

Przykłady i wyniki:

Przykłady funkcji

Przykłady	Wyniki
`RangeMin (1,2,4)`	Zwraca wartość 1

Przykłady	Wyniki
`RangeMin (1,'xyz')`	Zwraca wartość 1
`RangeMin (null( ), 'abc')`	Zwraca wartość NULL

Przykład:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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
];
```

Tabela wynikowa przedstawia zwrócone wartości MyRangeMin dla każdego z rekordów w tabeli.

Tabela wynikowa

RangeID	MyRangeMin
1	5
2	2
3	2
4	9
5	5
6	2

Przykład z wyrażeniem:

```
RangeMin (Above(MyField,0,3)
```

Zwraca wartość minimalną w zakresie trzech wartości z kolumny **MyField** obliczaną dla bieżącego wiersza oraz dwóch wierszy powyżej niego. Jeśli trzeci argument ma wartość 3, funkcja **Above()** zwraca trzy wartości (o ile istnieje wystarczająca liczba wierszy powyżej), które stanowią dane wejściowe dla funkcji **RangeMin()**.

Dane zastosowane w przykładach:

Dane przykładowe

MyField	RangeMin(Above(MyField,0,3))
10	10

MyField	RangeMin(Above(MyField,0,3))
2	2
8	2
18	2
5	5
9	5

Dane zastosowane w przykładach:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Zob. także:

p *Min – funkcja wykresu (page 329)*

RangeMinString

Funkcja **RangeMinString()** zwraca pierwszą z posortowanych tekstowo wartości znalezionych w wyrażeniu lub polu.

Składnia:

```
RangeMinString(first_expr[, Expression])
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty tej funkcji mogą zawierać funkcje międzywierszowe, które same zwracają listę wartości.

Argumenty

Argument	Opis
first_expr	Wyrażenie lub pole zawierające mierzone dane.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Przykłady i wyniki:

Przykłady funkcji

Przykłady	Wyniki
`RangeMinString (1,2,4)`	Zwraca wartość 1
`RangeMinString ('xyz','abc')`	Zwraca wartość 'abc'
`RangeMinString (5,'abc')`	Zwraca wartość 5
`RangeMinString (null( ))`	Zwraca wartość NULL

Przykład z wyrażeniem:

```
RangeMinString (Above(MinString(MyField),0,3))
```

Zwraca pierwszy (w kolejności sortowania tekstu) z trzech wyników funkcji **MinString(MyField)** dla bieżącego wiersza oraz dwóch wierszy powyżej niego.

Dane zastosowane w przykładach:

*Aby przykład działał zgodnie z oczekiwaniami, wyłącz sortowanie pola **MyField**.*

Dane przykładowe

MyField	RangeMinString(Above(MinString(MyField),0,3))
10	10
abc	10
8	8
def	8
xyz	8
9	9

Dane zastosowane w przykładach:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

Zob. także:

p *MinString – funkcja wykresu (page 527)*

RangeMissingCount

Funkcja **RangeMissingCount()** zwraca liczbę wartości nieliczbowych (w tym wartości NULL) w wyrażeniu lub polu.

Składnia:

```
RangeMissingCount(first_expr[, Expression])
```

Typ zwracanych danych: integer

Argumenty:

Argumenty tej funkcji mogą zawierać funkcje międzywierszowe, które same zwracają listę wartości.

Argumenty

Argument	Opis
first_expr	Wyrażenie lub pole zawierające dane do zliczenia.
Expression	Opcjonalne wyrażenia lub pola zawierające zliczany zakres danych.

Przykłady i wyniki:

Przykłady funkcji

Przykłady	Wyniki
`RangeMissingCount (1,2,4)`	Zwraca wartość 0
`RangeMissingCount (5,'abc')`	Zwraca wartość 1
`RangeMissingCount (null( ))`	Zwraca wartość 1

Przykład z wyrażeniem:

```
RangeMissingCount (Above(MinString(MyField),0,3))
```

Zwraca liczbę wartości nieliczbowych z trzech wyników funkcji **MinString(MyField)** dla bieżącego wiersza oraz dwóch wierszy powyżej niego.

*Aby przykład działał zgodnie z oczekiwaniami, wyłącz sortowanie pola **MyField**.*

Dane przykładowe

MyField	RangeMissingCount (Above(MinString (MyField),0,3))	Explanation
10	2	Zwraca wartość 2, ponieważ nie ma wierszy powyżej tego wiersza, w wyniku czego brakuje dwóch z trzech wartości.
abc	2	Zwraca wartość 2, ponieważ nad bieżącym wierszem jest tylko jeden wiersz, a bieżący wiersz zawiera wartość nieliczbową ('abc').
8	1	Zwraca wartość 1, ponieważ jeden z trzech wierszy zawiera wartość nieliczbową ('abc').
def	2	Zwraca wartość 2, ponieważ dwa z trzech wierszy zawierają wartości nieliczbowe ('def' i 'abc').
xyz	2	Zwraca wartość 2, ponieważ dwa z trzech wierszy zawierają wartości nieliczbowe (' xyz' i 'def').
9	2	Zwraca wartość 2, ponieważ dwa z trzech wierszy zawierają wartości nieliczbowe (' xyz' i 'def').

Dane zastosowane w przykładach:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

Zob. także:

p *MissingCount – funkcja wykresu (page 347)*

RangeMode

Funkcja **RangeMode()** zwraca wartość najczęściej występującą w wyrażeniu lub polu (wartość modalną).

Składnia:

```
RangeMode(first_expr {, Expression})
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty tej funkcji mogą zawierać funkcje międzywierszowe, które same zwracają listę wartości.

Argumenty

Argument	Opisu
first_expr	Wyrażenie lub pole zawierające mierzone dane.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Ograniczenia:

Jeśli z najwyższą częstością występuje więcej niż jedna wartość, wówczas zwracana jest wartość NULL.

Przykłady i wyniki:

Przykłady funkcji

Przykłady	Wyniki
`RangeMode (1,2,9,2,4)`	Zwraca wartość 2
`RangeMode ('a',4,'a',4)`	Zwraca wartość NULL
`RangeMode (null( ))`	Zwraca wartość NULL

Przykład:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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
];
```

Tabela docelowa przedstawia zwrócone wartości funkcji**MyRangeMode** dla każdego z rekordów w tabeli.

Tabela wynikowa

RangeID	MyRangMode
1	-

RangeID	MyRangMode
2	-
3	8
4	-
5	5
6	-

Przykład z wyrażeniem:

```
RangeMode (Above(MyField,0,3))
```

Zwraca najczęściej występującą wartość z trzech wyników funkcji **MyField** dla bieżącego wiersza oraz dwóch wierszy powyżej niego. Jeśli trzeci argument ma wartość 3, funkcja **Above()** zwraca trzy wartości (o ile istnieje wystarczająca liczba wierszy powyżej), które stanowią dane wejściowe dla funkcji **RangeMode ()**.

Dane zastosowane w przykładzie:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

*Aby przykład działał zgodnie z oczekiwaniami, wyłącz sortowanie pola **MyField**.*

Dane przykładowe

MyField	RangeMode(Above(MyField,0,3))
10	Zwraca wartość 10, ponieważ nie ma wierszy powyżej, zatem ta wartość pojedyncza jest wartością najczęściej występującą.
2	-
8	-
18	-
5	-
9	-

Zob. także:

p *Mode – funkcja wykresu (page 332)*

RangeNPV

Funkcja **RangeNPV()** zwraca wartość bieżącą netto inwestycji na podstawie stopy dyskontowej oraz serii przyszłych okresowych płatności (wartości ujemnych) i dochodów (wartości dodatnich). Wynik ma następujący domyślny format liczby: **money**.

W przypadku przepływów pieniężnych (niekoniecznie okresowych) zob. *RangeXNPV (page 1344)*.

Składnia:

```
RangeNPV(discount_rate, value[,value][, Expression])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
discount_rate	Stopa procentowa za okres.
value	Płatność lub dochód na zakończenie każdego okresu. Każda wartość może być wartością pojedynczą lub zakresem wartości, jakie zwracają funkcje międzywierszowe z trzecim parametrem opcjonalnym.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące są pomijane.

Przykłady	Wyniki
`RangeNPV(0.1,-10000,3000,4200,6800)`	Zwraca wartość 1188,44

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.
`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 '|');`
	Tabela wynikowa przedstawia zwrócone wartości RangeNPV dla każdego z rekordów w tabeli.
	RangeID: 1 — RangeNPV: $-49.13
	2 — $777.78
	3 — $98.77
	4 — $25.51
	5 — $250.83
	6 — $20.40

Zob. także:

p *Funkcje międzywierszowe (page 1249)*

RangeNullCount

Funkcja **RangeNullCount()** zwraca liczbę wartości NULL w wyrażeniu lub polu.

Składnia:

```
RangeNullCount(first_expr [, Expression])
```

Typ zwracanych danych: integer

Argumenty:

Argumenty tej funkcji mogą zawierać funkcje międzywierszowe, które same zwracają listę wartości.

Argumenty

Argument	Opis
first_expr	Wyrażenie lub pole zawierające mierzone dane.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Przykłady i wyniki:

Przykłady funkcji

Przykłady	Wyniki
`RangeNullCount (1,2,4)`	Zwraca wartość 0
`RangeNullCount (5,'abc')`	Zwraca wartość 0
`RangeNullCount (null( ), null( ))`	Zwraca wartość 2

Przykład z wyrażeniem:

```
RangeNullCount (Above(Sum(MyField),0,3))
```

Zwraca liczbę wartości NULL z trzech wyników funkcji **Sum(MyField)** dla bieżącego wiersza oraz dwóch wierszy powyżej niego.

*Skopiowanie kolumny **MyField** w poniższym przykładzie nie spowoduje zwrócenia wartości NULL.*

Dane przykładowe

MyField	RangeNullCount(Above(Sum(MyField),0,3))
10	Zwraca wartość 2, ponieważ nie ma wierszy powyżej tego wiersza, w wyniku czego brakuje dwóch z trzech wartości (czyli mają wartość NULL).
'abc'	Zwraca wartość 1, ponieważ nad bieżącym wierszem jest tylko jeden wiersz, w wyniku czego brakuje jednej z trzech wartości (czyli ma ona wartość NULL).
8	Zwraca wartość 0, ponieważ żaden z tych trzech wierszy nie zawiera wartości NULL.

Dane zastosowane w przykładach:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
] ;
```

Zob. także:

p *NullCount – funkcja wykresu (page 350)*

RangeNumericCount

Funkcja **RangeNumericCount()** zwraca liczbę wartości liczbowych w wyrażeniu lub polu.

Składnia:

```
RangeNumericCount(first_expr[, Expression])
```

Typ zwracanych danych: integer

Argumenty:

Argumenty tej funkcji mogą zawierać funkcje międzywierszowe, które same zwracają listę wartości.

Argumenty

Argument	Opis
first_expr	Wyrażenie lub pole zawierające mierzone dane.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Przykłady i wyniki:

Przykłady funkcji

Przykłady	Wyniki
`RangeNumericCount (1,2,4)`	Zwraca wartość 3

Przykłady	Wyniki
`RangeNumericCount (5,'abc')`	Zwraca wartość 1
`RangeNumericCount (null( ))`	Zwraca wartość 0

Przykład z wyrażeniem:

```
RangeNumericCount (Above(MaxString(MyField),0,3))
```

Zwraca liczbę wartości liczbowych z trzech wyników funkcji **MaxString(MyField)** dla bieżącego wiersza oraz dwóch wierszy powyżej niego.

*Aby przykład działał zgodnie z oczekiwaniami, wyłącz sortowanie pola **MyField**.*

Dane przykładowe

MyField	RangeNumericCount(Above(MaxString(MyField),0,3))
10	1
abc	1
8	2
def	1
xyz	1
9	1

Dane zastosowane w przykładach:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
def
xyz
9
] ;
```

Zob. także:

p *NumericCount – funkcja wykresu (page 353)*

RangeOnly

RangeOnly() to funkcja podwójna zwracająca wartość, jeśli wynikiem wyrażenia jest dokładnie jedna, unikatowa wartość. W przeciwnym razie zwracana jest wartość **NULL**.

Składnia:

```
RangeOnly(first_expr[, Expression])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty tej funkcji mogą zawierać funkcje międzywierszowe, które same zwracają listę wartości.

Argument	Opis
first_expr	Wyrażenie lub pole zawierające mierzone dane.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Przykłady i wyniki:

Przykłady	Wyniki
`RangeOnly (1,2,4)`	Zwraca wartość NULL
`RangeOnly (5,'abc')`	Zwraca wartość NULL
`RangeOnly (null( ), 'abc')`	Zwraca wartość 'abc'
`RangeOnly(10,10,10)`	Zwraca wartość 10

Zob. także:

p *Only – funkcja wykresu (page 335)*

RangeSkew

Funkcja **RangeSkew()** zwraca wartość odpowiadającą skośności zakresu liczb.

Składnia:

```
RangeSkew(first_expr[, Expression])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty tej funkcji mogą zawierać funkcje międzywierszowe, które same zwracają listę wartości.

Argumenty

Argument	Opisu
first_expr	Wyrażenie lub pole zawierające mierzone dane.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Ograniczenia:

Jeśli nie zostanie znaleziona żadna wartość liczbowa, zostanie zwrócona wartość NULL.

Przykłady i wyniki:

Przykłady funkcji

Przykłady	Wyniki
`rangeskew (1,2,4)`	Zwraca wartość 0,93521952958283
`rangeskew (above (SalesValue,0,3))`	Zwraca kroczącą skośność zakresu trzech wartości zwróconych przez funkcję above() dla bieżącego wiersza oraz dwóch wierszy powyżej niego.

Dane zastosowane w przykładzie:

Dane przykładowe

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
] ;
```

Zob. także:

p *Skew – funkcja wykresu (page 435)*

RangeStdev

Funkcja **RangeStdev()** określa odchylenie standardowe zakresu liczb.

Składnia:

```
RangeStdev(first_expr[, Expression])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty tej funkcji mogą zawierać funkcje międzywierszowe, które same zwracają listę wartości.

Argumenty

Argument	Opisu
first_expr	Wyrażenie lub pole zawierające mierzone dane.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Ograniczenia:

Jeśli nie zostanie znaleziona żadna wartość liczbowa, zostanie zwrócona wartość NULL.

Przykłady i wyniki:

Przykłady funkcji

Przykłady	Wyniki
`RangeStdev (1,2,4)`	Zwraca wartość 1,5275252316519
`RangeStdev (null( ))`	Zwraca wartość NULL
`RangeStdev (above (SalesValue),0,3))`	Zwraca kroczące odchylenie standardowe zakresu trzech wartości zwróconych przez funkcję above() dla bieżącego wiersza oraz dwóch wierszy powyżej niego.

Dane zastosowane w przykładzie:

Dane przykładowe

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
] ;
```

Zob. także:

p *Stdev – funkcja wykresu (page 438)*

RangeSum

RangeSum() zwraca sumę zakresu wartości. Wszystkie wartości nienumeryczne są traktowane jako 0.

Składnia:

```
RangeSum(first_expr[, Expression])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty tej funkcji mogą zawierać funkcje międzywierszowe, które same zwracają listę wartości.

Argumenty

Argument	Opisu
first_expr	Wyrażenie lub pole zawierające mierzone dane.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Ograniczenia:

Funkcja **RangeSum** interpretuje wszelkie wartości nieliczbowe jako 0.

Przykłady i wyniki:

Przykłady

Przykłady	Wyniki
`RangeSum (1,2,4)`	Zwraca wartość 7
`RangeSum (5,'abc')`	Zwraca wartość 5
`RangeSum (null( ))`	Zwraca wartość 0

Przykład:

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.

```
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
];
```

Tabela wynikowa przedstawia zwrócone wartości MyRangeSum dla każdego z rekordów w tabeli.

Tabela wynikowa

RangeID	MyRangeSum
1	21
2	12
3	18
4	38
5	19
6	15

Przykład z wyrażeniem:

```
RangeSum (Above(MyField,0,3))
```

Zwraca wartość sumy trzech wartości z kolumny **MyField)** dla bieżącego wiersza oraz dwóch wierszy powyżej niego. Jeśli trzeci argument ma wartość 3, funkcja **Above()** zwraca trzy wartości (o ile istnieje wystarczająca liczba wierszy powyżej), które stanowią dane wejściowe dla funkcji **RangeSum()**.

Dane zastosowane w przykładach:

*Aby przykład działał zgodnie z oczekiwaniami, wyłącz sortowanie pola **MyField**.*

Dane przykładowe

MyField	RangeSum(Above(MyField,0,3))
10	10
2	12
8	20
18	28
5	31
9	32

Dane zastosowane w przykładach:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Zob. także:

p *Sum – funkcja wykresu (page 338)*
p *Above – funkcja wykresu (page 1253)*

RangeTextCount

Funkcja **RangeTextCount()** zwraca liczbę wartości tekstowych w wyrażeniu lub polu.

Składnia:

```
RangeTextCount(first_expr[, Expression])
```

Typ zwracanych danych: integer

Argumenty:

Argumenty tej funkcji mogą zawierać funkcje międzywierszowe, które same zwracają listę wartości.

Argument

Argument	Opis
first_expr	Wyrażenie lub pole zawierające mierzone dane.
Expression	Opcjonalne wyrażenia lub pola zawierające mierzony zakres danych.

Przykłady i wyniki:

Przykłady funkcji

Przykłady	Wyniki
`RangeTextCount (1,2,4)`	Zwraca wartość 0
`RangeTextCount (5,'abc')`	Zwraca wartość 1
`RangeTextCount (null( ))`	Zwraca wartość 0

Przykład z wyrażeniem:

```
RangeTextCount (Above(MaxString(MyField),0,3))
```

Zwraca liczbę wartości tekstowych z trzech wyników funkcji **MaxString(MyField)** dla bieżącego wiersza oraz dwóch wierszy powyżej niego.

Dane zastosowane w przykładach:

*Aby przykład działał zgodnie z oczekiwaniami, wyłącz sortowanie pola **MyField**.*

Przykładowe dane

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

Dane zastosowane w przykładach:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
null()
'xyz'
9
] ;
```

Zob. także:

p *TextCount – funkcja wykresu (page 356)*

RangeXIRR

Funkcja **RangeXIRR()** zwraca wewnętrzną stopę zwrotu dla zaplanowanych przepływów pieniężnych, niekoniecznie okresowych. Do obliczania wewnętrznej stopy zwrotu dla serii okresowych przepływów pieniężnych należy używać funkcji **RangeIRR**.

Składnia:

```
RangeXIRR(value, date{, value, date})
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
value	Przepływ pieniężny lub seria przepływów pieniężnych odpowiadające płatnościom zaplanowanym w konkretnych terminach. Seria wartości musi zawierać co najmniej jedną wartość dodatnią i jedną ujemną.
date	Data płatności lub harmonogram dat płatności powiązane z płatnościami w ramach przepływów pieniężnych.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące są pomijane.

Wszystkie płatności są obejmowane upustem na podstawie roku zawierającego 365 dni.

Przykłady	Wyniki
`RangeXIRR(-2500,'2008-01-01',2750,'2008-09-01')`	Zwraca wartość 0,1532

Zob. także:

p *RangeIRR (page 1319)*

RangeXNPV

Współczynnik **RangeXNPV()** zwraca wartość bieżącą netto zaplanowanych przepływów pieniężnych, niekoniecznie okresowych. Wynik ma następujący domyślny format liczby: waluta. Do obliczania bieżącej wartości netto serii okresowych przepływów pieniężnych należy używać funkcji **RangeNPV**.

Składnia:

```
RangeXNPV(discount_rate, values, dates[, Expression])
```

Typ zwracanych danych: numeric

Argumenty:

Argumenty

Argument	Opis
discount_rate	Stopa procentowa za okres.
values	Przepływ pieniężny lub seria przepływów pieniężnych odpowiadające płatnościom zaplanowanym w konkretnych terminach. Każda wartość może być wartością pojedynczą lub zakresem wartości, jakie zwracają funkcje międzywierszowe z trzecim parametrem opcjonalnym. Seria wartości musi zawierać co najmniej jedną wartość dodatnią i jedną ujemną.
dates	Data płatności lub harmonogram dat płatności powiązane z płatnościami w ramach przepływów pieniężnych.

Ograniczenia:

Wartości tekstowe, wartości NULL i wartości brakujące są pomijane.

Wszystkie płatności są obejmowane upustem na podstawie roku zawierającego 365 dni.

Tabela przykładowa

Przykłady	Wyniki
`RangeXNPV(0.1, -2500,'2008-01-01',2750,'2008-09-01')`	Zwraca wartość 80,25

Dodaj skrypt przykładowy do aplikacji i uruchom ją. Aby zobaczyć wynik, dodaj do arkusza w swojej aplikacji pola wyszczególnione w kolumnie wyników.
`RangeTab3:`
`LOAD *,`
`recno() as RangeID,`
`RangeXNPV(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 '|');`
	Tabela wynikowa przedstawia zwrócone wartości RangeXNPV dla każdego z rekordów w tabeli.
	RangeID: RangeXNPV
	1: $-49.13
	2: $777.78
	3: $98.77
	4: $25.51
	5: $250.83
	6: $20.40

5.22 Funkcje klasyfikacji i grupowania

Te funkcje mogą być używane tylko w wyrażeniach wykresu.

Funkcje klasyfikacji w wykresach

Gdy takie funkcje są używane, wówczas pomijanie wartości zerowych jest automatycznie wyłączone. Wartości NULL są ignorowane.

Rank

Funkcja **Rank()** oblicza wartości wierszy wykresu w wyrażeniu i dla każdego wiersza zwraca względną pozycję wartości wymiaru obliczanego w wyrażeniu. Obliczając wartość wyrażenia, funkcja porównuje wynik z wynikiem dla innych wierszy zawierających bieżący segment kolumny i zwraca klasyfikację bieżącego wiersza w ramach segmentu.

```
Rank — funkcja wykresu([TOTAL [<fld {, fld}>]] expr[, mode[, fmt]])
```

HRank

Funkcja **HRank()** oblicza wartość wyrażenia i porównuje otrzymany wynik z wynikami z innych kolumn zawierających bieżący segment wierszy tabeli przestawnej. Funkcja zwraca następnie klasyfikację bieżącej kolumny w ramach segmentu.

```
HRank — funkcja wykresu([TOTAL] expr[, mode[, fmt]])
```

Funkcje grupowania w wykresach

KMeans2D

Grupa właściwości **Licencja na lokację** zawiera właściwości związane z licencją na system Qlik Sense. Wszystkie pola są obowiązkowe i nie mogą być puste.

Właściwości licencji na lokację

Property name	Opis
Imię i nazwisko/nazwa właściciela	Nazwa użytkownika właściciela produktu Qlik Sense.
Organizacja właściciela	Nazwa organizacji, której członkiem jest właściciel produktu Qlik Sense.
Numer seryjny	Numer seryjny przypisany do oprogramowania Qlik Sense.
Numer kontrolny	Numer kontrolny przypisany do oprogramowania Qlik Sense.
Dostęp przy użyciu LEF	Plik License Enabler File (LEF) przypisany do oprogramowania Qlik Sense.

KMeans2D() poddaje ocenie wiersze wykresu, stosując algorytm centroidów oraz wyświetlając dla każdego wiersza wykresu identyfikator klastra, do którego został przypisany ten punkt danych. Kolumny wykorzystywane przez algorytm grupowania są określane przez odpowiednio parametry coordinate_1 i coordinate_2. Oba te parametry są agregacjami. Liczba tworzonych klastrów jest określana przez parametr num_clusters. Dane można opcjonalnie znormalizować za pomocą parametru norm.

```
KMeans2D — funkcja wykresu(num_clusters, coordinate_1, coordinate_2 [, norm])
```

KMeansND

KMeansND() poddaje ocenie wiersze wykresu, stosując algorytm centroidów oraz wyświetlając dla każdego wiersza wykresu identyfikator klastra, do którego został przypisany ten punkt danych. Kolumny wykorzystywane przez algorytm grupowania są określone przez odpowiednio parametry coordinate_1 i coordinate_2 itd., aż do n kolumn. Wszystkie te parametry są agregacjami. Liczba tworzonych klastrów jest określana przez parametr num_clusters.

```
KMeansND – funkcja wykresu(num_clusters, num_iter, coordinate_1, coordinate_2
[,coordinate_3 [, ...]])
```

KMeansCentroid2D

KMeansCentroid2D() poddaje ocenie wiersze wykresu, stosując algorytm centroidów oraz wyświetlając dla każdego wiersza wykresu oczekiwaną współrzędną klastra, do którego został przypisany ten punkt danych. Kolumny wykorzystywane przez algorytm grupowania są określone przez odpowiednio parametry coordinate_1 i coordinate_2. Oba te parametry są agregacjami. Liczba tworzonych klastrów jest określana przez parametr num_clusters. Dane można opcjonalnie znormalizować za pomocą parametru norm.

```
KMeansCentroid2D – funkcja wykresu(num_clusters, coordinate_no, coordinate_1,
coordinate_2 [, norm])
```

KMeansCentroidND

KMeansCentroidND() poddaje ocenie wiersze wykresu, stosując algorytm centroidów oraz wyświetlając dla każdego wiersza wykresu oczekiwaną współrzędną klastra, do którego został przypisany ten punkt danych. Kolumny wykorzystywane przez algorytm grupowania są określone przez odpowiednio parametry coordinate_1, coordinate_2 itd., aż do n kolumn. Wszystkie te parametry są agregacjami. Liczba tworzonych klastrów jest określana przez parametr num_clusters.

```
KMeansCentroidND – funkcja wykresu(num_clusters, num_iter, coordinate_no,
coordinate_1, coordinate_2 [,coordinate_3 [, ...]])
```

Rank – funkcja wykresu

Funkcja **Rank()** oblicza wartości wierszy wykresu w wyrażeniu i dla każdego wiersza zwraca względną pozycję wartości wymiaru obliczanego w wyrażeniu. Obliczając wartość wyrażenia, funkcja porównuje wynik z wynikiem dla innych wierszy zawierających bieżący segment kolumny i zwraca klasyfikację bieżącego wiersza w ramach segmentu.

Segmenty kolumn

	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

W przypadku wykresów innych niż tabele bieżący segment kolumny jest definiowany tak, jak pojawia się w odpowiedniku tabeli prostej dla takiego wykresu.

Składnia:

```
Rank([TOTAL] expr[, mode[, fmt]])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
mode	Określa reprezentację liczbową wyniku funkcji.
fmt	Określa reprezentację tekstową wyniku funkcji.
TOTAL	Jeśli wykres jest jednowymiarowy lub wyrażenie jest poprzedzone kwalifikatorem **TOTAL**, funkcja będzie obliczana w całej kolumnie. Jeśli tabela lub równoważnik tabeli zawiera wiele wymiarów pionowych, wówczas segment bieżącej kolumny będzie zawierał tylko wiersze z takimi samymi wartościami we wszystkich kolumnach wymiaru jak bieżący wiersz, ale bez kolumny przedstawiającej ostatni wymiar w kolejności sortowania między polami.

Klasyfikacja jest zwracana jako wartość podwójna, która w sytuacji, gdy każdy wiersz ma niepowtarzalną klasyfikację, będzie liczbą całkowitą z zakresu od 1 do liczby wierszy w bieżącym segmencie kolumny.

Jeśli kilka wierszy ma tę samą klasyfikację, reprezentację tekstową i liczbową można kontrolować za pomocą argumentów **mode** i **fmt**.

mode
Drugi argument, **mode**, może mieć następujące wartości:

mode – przykłady

Wartość	Opis
0 (domyślnie)	Jeśli wszystkie klasyfikacje w grupie o takich samych klasyfikacjach należą do dolnego zakresu wartości środkowej całej klasyfikacji, wówczas wszystkie wiersze uzyskują najniższą klasyfikację w tej grupie. Jeśli wszystkie klasyfikacje w grupie o takich samych klasyfikacjach należą do górnego zakresu wartości środkowej całej klasyfikacji, wówczas wszystkie wiersze uzyskują najwyższą klasyfikację w tej grupie. Jeśli klasyfikacje w grupie o takich samych klasyfikacjach obejmują środek zakresu całej klasyfikacji, wszystkie wiersze otrzymują wartość odpowiadającą średniej klasyfikacji górnej i dolnej w całym segmencie kolumny.
1	Najniższa klasyfikacja we wszystkich wierszach.
2	Średnia klasyfikacja we wszystkich wierszach.
3	Najwyższa klasyfikacja we wszystkich wierszach.

Wartość	Opis
4	Najniższa klasyfikacja w pierwszym wierszu, następnie zwiększana o jeden dla każdego wiersza.

fmt

Trzeci argument, **fmt**, może mieć następujące wartości:

fmt – przykłady

Wartość	Opis
0 (domyślnie)	Niska wartość **-** wysoka wartość we wszystkich wierszach (na przykład 3-4).
1	Niska wartość we wszystkich wierszach.
2	Niska wartość w pierwszym wierszu, pusta w kolejnych wierszach.

Kolejność wierszy dla argumentów **mode** 4 i **fmt** 2 jest określona przez kolejność sortowania wymiarów wykresu.

Przykłady i wyniki:

Utwórz dwie wizualizacje z wymiarów Product i Sales oraz kolejną wizualizację z wymiarów Product i UnitSales. Dodaj miary zgodnie z poniższą tabelą.

Przykłady klasyfikacji

Przykłady	Wyniki
Przykład 1. Utwórz tabelę z wymiarami `Customer` i `Sales` oraz miarą `Rank (Sales)`	Wynik zależy od kolejności sortowania wymiarów. Jeśli tabele są sortowane według wymiaru Customer, w tabeli zostaną wyszczególnione wszystkie wartości z kolumny Sales dla klienta Astrida, a następnie dla klienta Betacab itd. Wyniki w kolumnie Rank(Sales) będą następujące: 10 dla wartości Sales 12, 9 dla wartości Sales 13 itd., z wartością klasyfikacji 1 zwracaną dla wartości Sales 78. Następny segment kolumny rozpoczyna się wartością Betacab, dla której pierwszą wartością w kolumnie Sales w tym segmencie jest 12. Wartość klasyfikacji Rank(Sales) w tym przypadku wynosi 11. Jeśli tabela zostanie posortowana według wymiaru Sales, segmenty kolumny będą mieć wartości wymiaru Sales i odpowiadające im wartości wymiaru Customer. Ze względu na występowanie dwóch wartości 12 dla wymiaru Sales (dla klientów Astrida i Betacab) wartość miary Rank(Sales) dla tego segmentu kolumny wynosi 1-2, dla każdej wartości wymiaru Customer. Wynika to z faktu, że istnieją dwie wartości wymiaru Customer dla wartości Sales 12. Gdyby były cztery wartości, wynik wynosiłby 1-4 dla wszystkich wierszy. Ten przykład pokazuje, jak wyglądałyby wyniki dla wartości domyślnej (0) argumentu fmt.
Przykład 2. Zastąp wymiar Customer wymiarem Product i dodaj miarę `Rank (Sales,1,2)`	Zwrócona zostanie wartość 1 w pierwszym wierszu każdego segmentu kolumny, a wszystkie pozostałe wiersze będą puste, ponieważ argumenty **mode** i **fmt** mają odpowiednio wartości 1 i 2.

Wyniki przykładu 1, w tabeli posortowanej według wymiaru Customer:

Tabela wynikowa

Customer	Sales	Rank(Sales)
Astrida	12	10
Astrida	13	9
Astrida	20	8
Astrida	22	7
Astrida	45	6
Astrida	46	5
Astrida	60	4

Customer	Sales	Rank(Sales)
Astrida	65	3
Astrida	70	2
Astrida	78	1
Betcab	12	11

Wyniki przykładu 1, w tabeli posortowanej według wymiaru Sales:

Tabela wynikowa

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

Dane zastosowane w przykładach:

```
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 '|');
```

Zob. także:

p *Sum – funkcja wykresu (page 338)*

HRank – funkcja wykresu

Funkcja **HRank()** oblicza wartość wyrażenia i porównuje otrzymany wynik z wynikami z innych kolumn zawierających bieżący segment wierszy tabeli przestawnej. Funkcja zwraca następnie klasyfikację bieżącej kolumny w ramach segmentu.

Składnia:

```
HRank([ TOTAL ] expr [ , mode [, fmt ] ])
```

Typ zwracanych danych: dual

> *Ta funkcja działa tylko względem tabel przestawnych. We wszystkich innych typach wykresów zwraca wartość NULL.*

Argumenty:

Argumenty

Argument	Opis
expr	Wyrażenie lub pole zawierające mierzone dane.
mode	Określa reprezentację liczbową wyniku funkcji.
fmt	Określa reprezentację tekstową wyniku funkcji.
TOTAL	Jeśli wykres jest jednowymiarowy lub wyrażenie jest poprzedzone kwalifikatorem **TOTAL**, funkcja będzie obliczana w całej kolumnie. Jeśli tabela lub równoważnik tabeli zawiera wiele wymiarów pionowych, wówczas segment bieżącej kolumny będzie zawierał tylko wiersze z takimi samymi wartościami we wszystkich kolumnach wymiaru jak bieżący wiersz, ale bez kolumny przedstawiającej ostatni wymiar w kolejności sortowania między polami.

Jeśli tabela przestawna jest jednowymiarowa lub jeśli wyrażenie jest poprzedzone kwalifikatorem **total**, wówczas bieżący segment wiersza jest zawsze równy całemu wierszowi. Jeśli tabela przestawna zawiera wiele wymiarów poziomych, wówczas bieżący segment wiersza będzie zawierać tylko kolumny z takimi samymi wartościami, co bieżąca kolumna we wszystkich wierszach wymiaru, z wyjątkiem wiersza przedstawiającego ostatni wymiar poziomy w kolejności sortowania między polami.

Klasyfikacja jest zwracana jako wartość podwójna, która w sytuacji, gdy każda kolumna ma unikalną klasyfikację będzie liczbą całkowitą z zakresu od 1 do liczby kolumn w bieżącym segmencie wiersza.

Jeśli kilka kolumn ma tę samą klasyfikację, reprezentację tekstową i liczbową można kontrolować za pomocą argumentów **mode** i **format**.

Drugi argument, **mode**, określa reprezentację liczbową wyniku funkcji:

mode – przykłady

Wartość	Opis
0 (domyślnie)	Jeśli wszystkie klasyfikacje w grupie o takich samych klasyfikacjach należą do dolnego zakresu wartości środkowej całej klasyfikacji, wówczas wszystkie kolumny uzyskują najniższą klasyfikację w tej grupie. Jeśli wszystkie klasyfikacje w grupie o takich samych klasyfikacjach należą do górnego zakresu wartości środkowej całej klasyfikacji, wówczas wszystkie kolumny uzyskują najwyższą klasyfikację w tej grupie. Jeśli klasyfikacje w grupie o takich samych klasyfikacjach obejmują środek zakresu całej klasyfikacji, wszystkie wiersze otrzymują wartość odpowiadającą średniej klasyfikacji górnej i dolnej w całym segmencie kolumny.
1	Najniższa klasyfikacja z wszystkich kolumn w grupie.
2	Średnia klasyfikacja z wszystkich kolumn w grupie.
3	Najwyższa klasyfikacja z wszystkich kolumn w grupie.
4	Najniższa klasyfikacja w pierwszej kolumnie, następnie zwiększana o jeden dla każdej kolumny w grupie.

Trzeci argument, **format**, określa reprezentację tekstową wyniku funkcji:

format – przykłady

Wartość	Opis
0 (domyślnie)	Niska wartość **&' - '&** wysoka wartość we wszystkich kolumnach w grupie (na przykład 3-4).
1	Niska wartość we wszystkich kolumnach w grupie.
2	Niska wartość w pierwszej kolumnie, pusta w kolejnych kolumnach w grupie.

Kolejność kolumn dla argumentów **mode** 4 i **format** 2 jest określona przez kolejność sortowania wymiarów wykresu.

Przykłady:

```
HRank( sum( Sales ))
HRank( sum( Sales ), 2 )
HRank( sum( Sales ), 0, 1 )
```

Optymalizacja za pomocą KMeans: Rzeczywisty przykład

Poniższy przykład przedstawia rzeczywisty przypadek użycia, w którym funkcje klastrowania KMeans i Centroid są stosowane do zestawu danych. Funkcja KMeans segreguje podobne punkty danych w klastry. Klastry stają się bardziej zwarte i zróżnicowane, gdy algorytm KMeans jest stosowany w konfigurowalnej liczbie iteracji.

KMeans to funkcja używana w wielu dziedzinach do różnych zastosowań. Niektóre przykłady klastrowania obejmują segmentację klientów, wykrywanie oszustw, przewidywanie utraty klientów, targetowanie zachęt dla klientów, identyfikację cyberprzestępstw i optymalizację tras dostaw. Algorytm klastrowania KMeans jest coraz częściej używany przez przedsiębiorstwa, które próbują wywnioskować wzorce i zoptymalizować ofertę usług.

Qlik Sense Funkcje KMeans i Centroid

Qlik Sense udostępnia dwie funkcje KMeans, które grupują punkty danych w klastry na podstawie podobieństwa. Zobacz *KMeans2D — funkcja wykresu (page 1363)* i *KMeansND — funkcja wykresu (page 1376)*. Funkcja **KMeans2D** przyjmuje dwa wymiary i sprawdza się przy wizualizacji wyników przy użyciu **wykresu punktowego**. Funkcja **KMeansND** przyjmuje więcej niż dwa wymiary. Ponieważ łatwo jest przedstawić koncepcję wyniku 2D na standardowych wykresach, w następującej demonstracji stosowana jest funkcja KMeans do **wykresu punktowego** przy użyciu dwóch wymiarów. Klastrowanie KMeans można wizualizować przez kolorowanie na podstawie wyrażenia lub według wymiaru, jak opisano w tym przykładzie.

Funkcje centroidów w Qlik Sense określają średnią arytmetyczną pozycji wszystkich punktów danych w klastrze i identyfikują punkt centralny, czyli centroid danego klastra. Dla każdego wiersza wykresu (lub rekordu) funkcja centroid wyświetla współrzędną klastra, do którego przypisano ten punkt danych. Zobacz *KMeansCentroid2D — funkcja wykresu (page 1389)* i *KMeansCentroidND — funkcja wykresu (page 1390)*.

Przegląd zastosowania i przykładu

Poniższy przykład przedstawia symulowany scenariusz w świecie rzeczywistym. Firma odzieżowa w stanie Nowy Jork w USA musi zmniejszyć wydatki, minimalizując koszty dostaw. Jednym ze sposobów na to jest przeniesienie magazynów bliżej dystrybutorów. Firma zatrudnia 118 dystrybutorów w całym stanie Nowy Jork. Poniższa demonstracja symuluje, w jaki sposób dyrektor operacyjny może posegmentować dystrybutorów na pięć skupionych obszarów geograficznych za pomocą funkcji KMeans, a następnie zidentyfikować pięć optymalnych lokalizacji magazynowych centralnych dla tych klastrów za pomocą funkcji centroid. Ma to na celu odkrycie współrzędnych mapowania, które można wykorzystać do identyfikacji pięciu lokalizacji magazynów centralnych.

Zestaw danych

Zestaw danych jest oparty na losowo generowanych nazwach i adresach w stanie Nowy Jork z rzeczywistymi współrzędnymi szerokości i długości geograficznej. Zestaw danych zawiera następujące dziesięć kolumn: id, first_name, last_name, telephone, address, city, state, zip, latitude, longitude. Zestaw danych jest dostępny poniżej jako plik, który można pobrać lokalnie, a następnie przesłać do Qlik Sense lub edytować w edytorze ładowania danych. Tworzona aplikacja nosi nazwę *Distributors KMeans and Centroid*, a pierwszy arkusz w aplikacji nazywa się *Distribution cluster analysis*.

Wybierz poniższe łącze, aby pobrać przykładowy plik danych: DistributorData.csv

Zestaw danych Distributor: Wbudowane ładowanie dla edytora ładowania danych w Qlik Sense (page 1361)

Tytuł: DistributorData

Całkowita liczba rekordów: 118

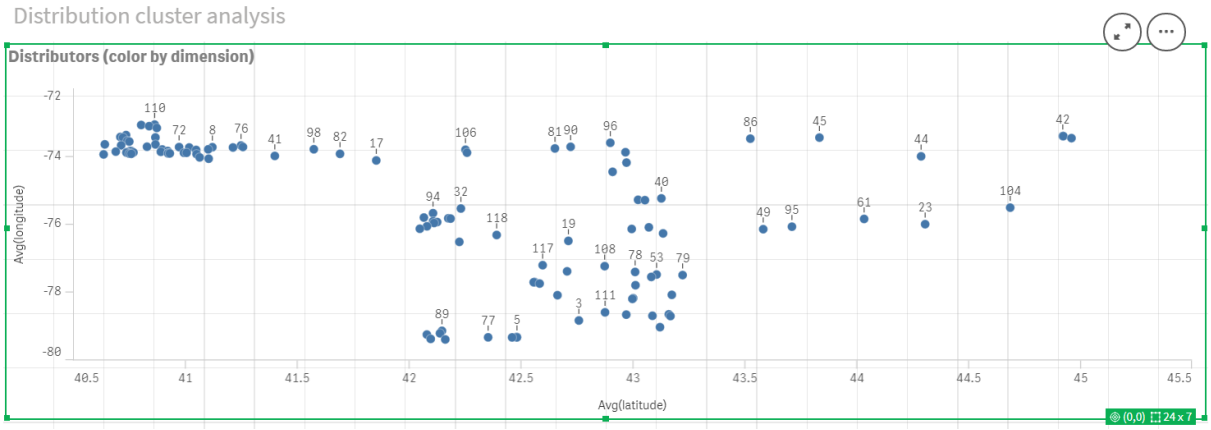
Stosowanie funkcji KMeans2D

W tym przykładzie konfiguracja **wykresu punktowego** jest pokazana przy użyciu zestawu danych *DistributorData*, stosowana jest funkcja **KMeans2D**, a wykres jest pokolorowany według wymiarów.

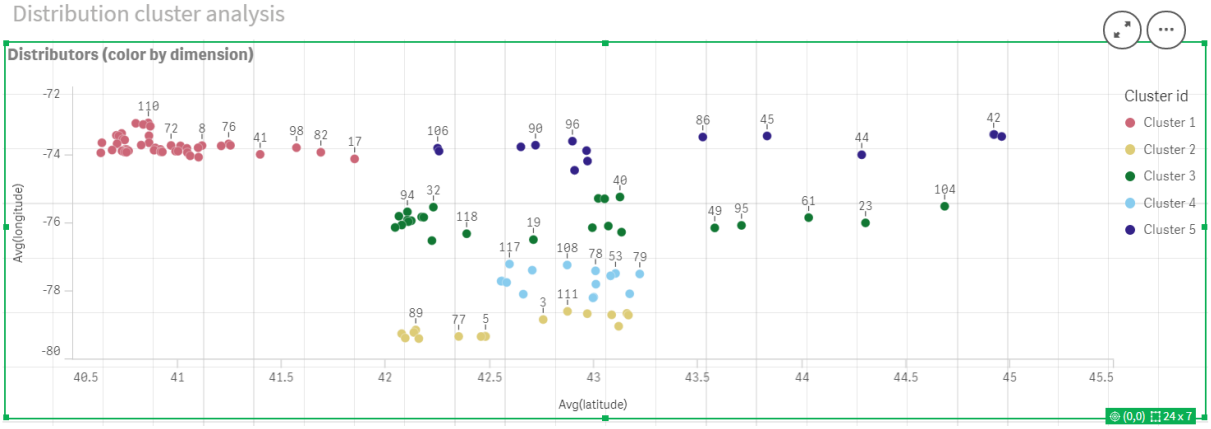
Uwaga: funkcje KMeans Qlik Sense obsługują automatyczne grupowanie za pomocą metody zwanej różnicą głębokości (DeD). Gdy użytkownik ustawia liczbę klastrów na 0, określana jest optymalna liczba klastrów dla tego zestawu danych. W tym przykładzie jednak jest tworzona zmienna dla argumentu **num_clusters** (informacje o składni zawiera *KMeans2D – funkcja wykresu (page 1363)*). W związku z tym pożądaną liczbę klastrów (k=5) określa zmienna.

1. **Wykres punktowy** zostaje przeciągnięty na arkusz i nazwany *Distributors (by dimension)*.
2. Tworzona jest **zmienna** do określenia liczby klastrów. **Zmienna** nosi nazwę *vDistClusters*. Dla zmiennej **Definition** wprowadza się *5*.
3. Konfiguracja **Danych** dla wykresu:
    a. Pod obszarem **Wymiary** pole *id* jest wybierane jako wartość **Bąbelek**. *Cluster id* wprowadza się jako wartość **Etykieta**.
    b. W sekcji **Miary**, *Avg([latitude])* jest wyrażeniem dla **osi X**.
    c. W sekcji **Miary**, *Avg([longitude])* jest wyrażeniem dla **osi Y**.
4. Konfiguracja **Wyglądu**:
    a. W sekcji **Kolory i legenda** wybierana jest wartość **Niestandardowe** dla **Kolorów**.
    b. Do kolorowania wykresu wybierana jest opcja **Według wymiaru**.
    c. Wprowadzane jest następujące wyrażenie: *=pick(aggr(KMeans2D(vDistClusters,only (latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    d. Pole wyboru dla **Trwałe kolory** jest zaznaczone.

Wykres punktowy przed zastosowaniem kolorowania KMeans według wymiaru



Wykres punktowy po zastosowaniu kolorowania KMeans według wymiaru



Dodawanie **tabeli**: *Distributors*

Warto mieć pod ręką tabelę, aby uzyskać szybki dostęp do odpowiednich danych. **Wykres punktowy** pokazuje wartości *id*, chociaż w celach informacyjnych dodano tabelę z nazwami odpowiednich dystrybutorów.

1. **Tabela** o nazwie *Distributors* jest przeciągana na arkusz z następującymi dodanymi **kolumnami** (wymiarami): *id*, *first_name*, i *last_name*.

Tabela: Distributor names

Distributors

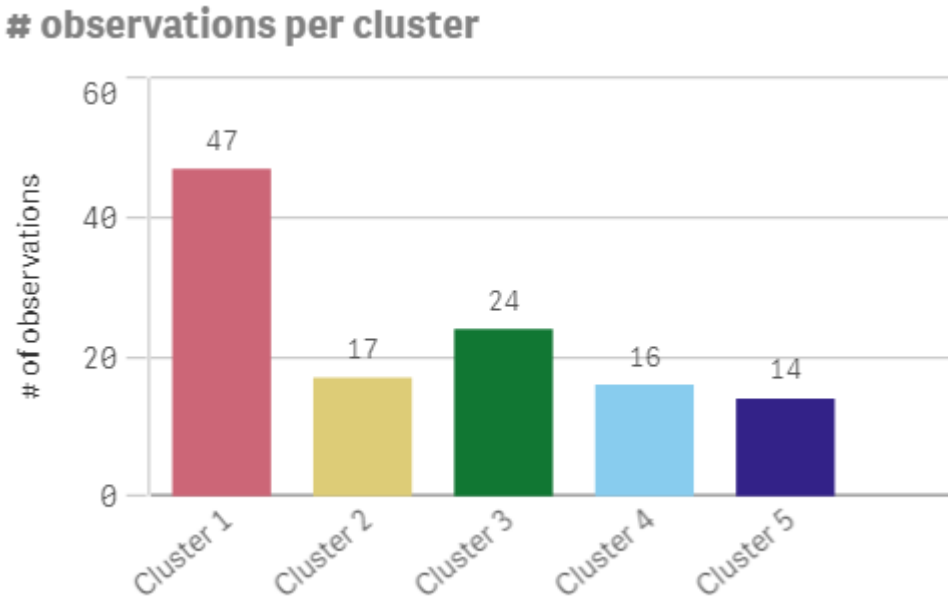
id	first_name	last_name
1	Kaiya	Snow
2	Dean	Roy
3	Eden	Paul
4	Bryanna	Higgins
5	Elisabeth	Lee
6	Skylar	Robinson
7	Cody	Bailey
8	Dario	Sims
9	Deacon	Hood

Dodawanie **wykresu słupkowego**: *# observations per cluster*

W przypadku scenariusza dystrybucji magazynowej warto wiedzieć, ilu dystrybutorów będzie obsługiwanych przez każdy magazyn. Dlatego tworzony jest **wykres słupkowy**, który mierzy, ilu dystrybutorów jest przypisanych do każdego klastra.

1. **Wykres słupkowy** jest przeciągany na arkusz. Wykres nosi nazwę: *# observations per cluster*.
2. Konfiguracja **Danych** do **wykresu słupkowego**:
    a. Zostanie dodany **wymiar** z etykietą *Clusters* (etykietę można dodać po zastosowaniu wyrażenia). Wprowadzane jest następujące wyrażenie: *=pick(aggr(KMeans2D (vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    b. Dodana zostaje **miara** z etykietą *# of observations*. Wprowadzane jest następujące wyrażenie: *=count(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id))*
3. Konfiguracja **Wyglądu**:
    a. W sekcji **Kolory i legenda** wybierana jest wartość **Niestandardowe** dla **Kolorów**.
    b. Do kolorowania wykresu wybierana jest opcja **Według wymiaru**.
    c. Wprowadzane jest następujące wyrażenie: *=pick(aggr(KMeans2D(vDistClusters,only (latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    d. Pole wyboru dla **Trwałe kolory** jest zaznaczone.
    e. Ustawienie **Pokaż legendę** jest wyłączone.
    f. W obszarze **Prezentacja** ustawienie **Etykiety wartości** ma wartość **Autom**.
    g. Pod **Osią X**: zaznaczono **Clusters**, **Tylko etykiety**.

Wykres słupkowy: # observations per cluster



Stosowanie funkcji **Centroid2D**

Druga tabela została dodana dla funkcji **Centroid2D**, która zidentyfikuje współrzędne potencjalnych lokalizacji magazynów. Ta tabela pokazuje centralną lokalizację (wartości centroidów) dla pięciu zidentyfikowanych grup dystrybutorów.

1. Na arkusz przeciągana jest **tabela**. Nosi nazwę *Cluster centroids* i ma dodane następujące kolumny:
    a. Dodany zostaje **wymiar** z etykietą *Clusters*. Wprowadzane jest następujące wyrażenie: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1,'Warehouse 1','Warehouse 2','Warehouse 3','Warehouse 4','Warehouse 5'*)
    b. Zostaje dodana **miara** z etykietą *latitude (D1)*. Wprowadzane jest następujące wyrażenie: *=only(aggr(KMeansCentroid2D(vDistClusters,0,only(latitude),only(longitude)),id))* Zauważ, że parametr **coordinate_no** odpowiada pierwszemu wymiarowi dimension(0). W tym przypadku wymiar *latitude* jest wykreślany względem osi x. Gdybyśmy pracowali z funkcją **CentroidND** i było do sześciu wymiarów, te parametry mogłyby mieć dowolną z sześciu wartości: 0,1,2,3,4 lub 5.
    c. Zostaje dodana **miara** z etykietą *longitude (D2)*. Wprowadzane jest następujące wyrażenie: *=only(aggr(KMeansCentroid2D(vDistClusters,1,only(latitude),only(longitude)),id))* Parametr **coordinate_no** w tym wyrażeniu odpowiada drugiemu wymiarowi dimension(1). Wymiar *longitude* jest wykreślany względem osi y.

Tabela: Cluster centroid calculations

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

Mapowanie centroidów

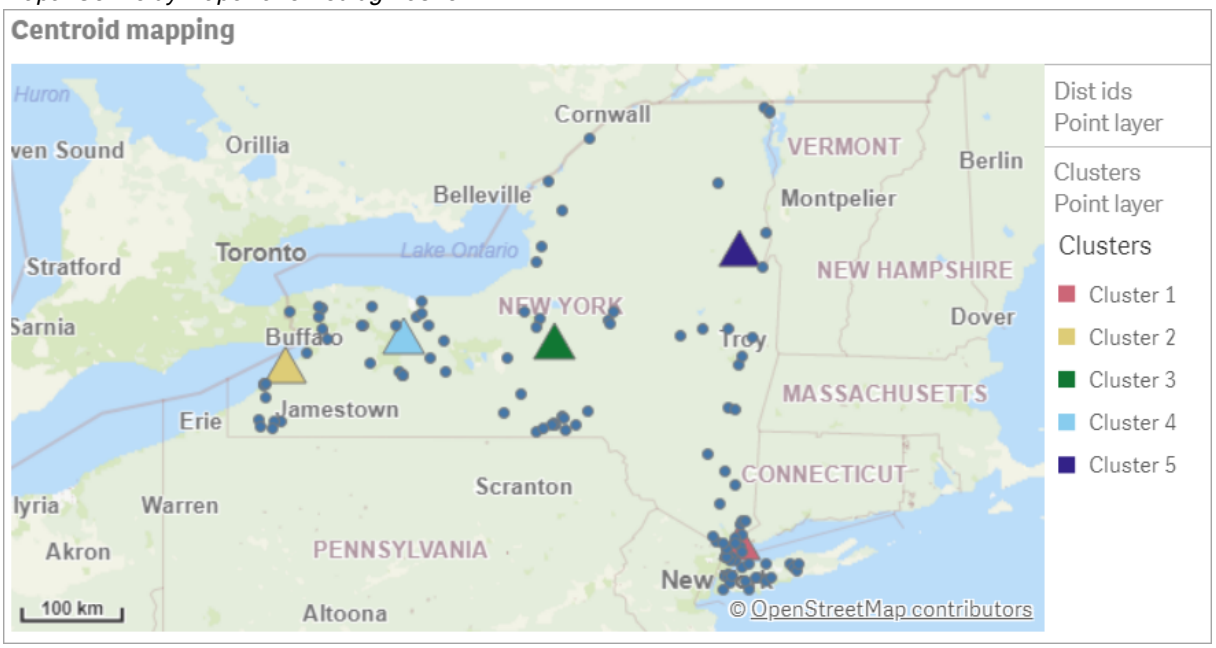
Następnym krokiem jest mapowanie centroidów. Od dewelopera aplikacji zależy, czy wizualizacja ma być umieszczona na osobnych arkuszach.

1. Na arkusz zostaje przeciągnięta **mapa** o nazwie *Centroid mapping*.
2. W sekcji **Warstwy** Wybiera się pozycję **Dodaj warstwę**, a następnie **Warstwa punktów**.
    a. Wybiera się **Field** *id* i dodaje **etykietę** *Dist ids*.
    b. W sekcji **Lokalizacja** zaznaczone jest pole wyboru **Pola długości i szerokości geograficznej**.
    c. W przypadku **Szerokości geograficznej** wybrane jest pole *latitude*.
    d. W przypadku **Długości geograficznej** wybrane jest pole *longitude*.
    e. W sekcji **Rozmiar i kształt** jako **Kształt** wybrana jest wartość **Bąbelek**, a **Rozmiar** jest zmniejszany zgodnie z preferencjami na suwaku.
    f. W sekcji **Kolory** wybrany jest **Pojedynczy kolor** i wartość niebieski jako **Kolor**, a szary jako **Kolor konturu** (te wybory są również kwestią preferencji).
3. W sekcji **Warstwy** druga **Warstwa punktów** jest dodawana przez wybranie **Dodaj warstwę**, a następnie **Warstwa punktów**.
    a. Wprowadzane jest następujące wyrażenie: *=aggr(KMeans2D(vDistClusters,only (latitude),only(longitude)),id)*
    b. Zostaje dodana **etykieta** *Clusters*.
    c. W sekcji **Lokalizacja** zaznaczone jest pole wyboru **Pola długości i szerokości geograficznej**.
    d. W przypadku **Szerokości geograficznej**, która jest tu wykreślana wzdłuż osi x, dodaje się następujące wyrażenie: *=aggr(KMeansCentroid2D(vDistClusters,0,only(latitude),only (longitude)),id)*
    e. W przypadku **Długości geograficznej**, która jest tu wykreślana wzdłuż osi y dodaje się następujące wyrażenie: *=aggr(KMeansCentroid2D(vDistClusters,1,only(latitude),only (longitude)),id)*

f. W sekcji **Rozmiar i kształt** jako **Kształt** wybierany jest **Trójkąt**, a **Rozmiar** jest zmniejszany zgodnie z preferencjami na suwaku.
g. W sekcji **Kolory i legenda** wybierana jest wartość **Niestandardowe** dla **Kolorów**.
h. Do kolorowania wykresu wybierana jest opcja **Według wymiaru**. Wprowadzane jest następujące wyrażenie: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only (longitude)),id)+1,'Cluster 1','Cluster 2','Cluster 3','Cluster 4','Cluster 5')*
i. Wymiar otrzymuje etykietę *Clusters*.

4. W **Ustawieniach mapy** wybierana jest wartość **Adaptacyjne** jako ustawienie **Rzutowanie**. Jako **Jednostki miary** wybierana jest wartość **Metryczne**.

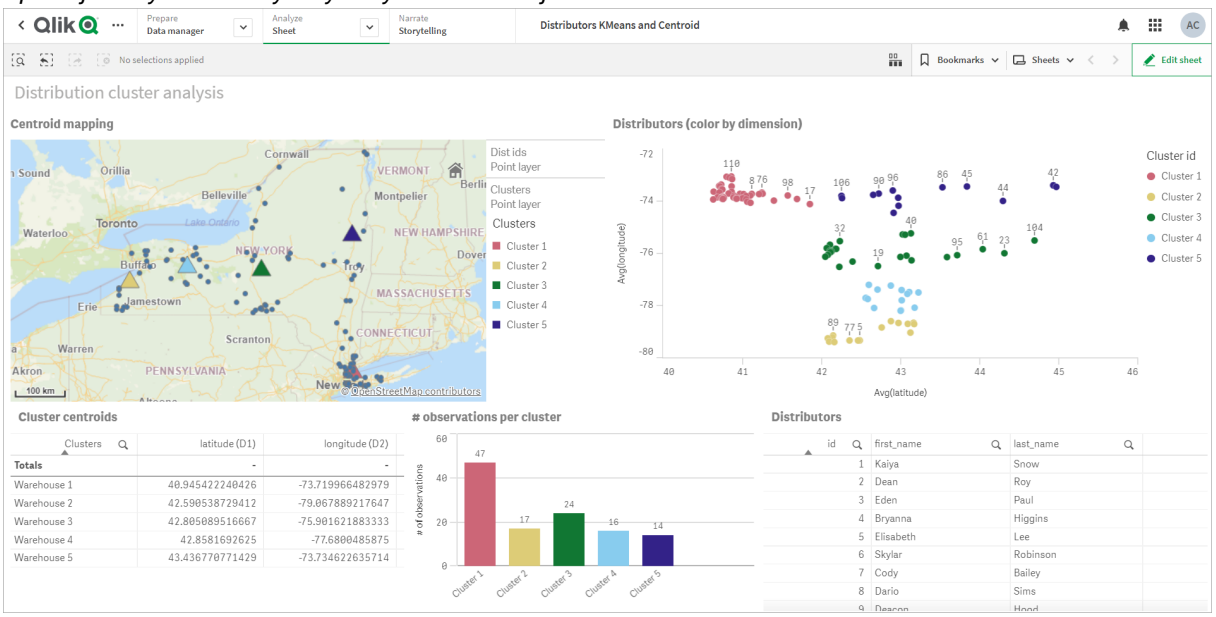
Mapa: Centroidy mapowane według klastrów



Podsumowanie

Przy użyciu funkcji KMeans w tym rzeczywistym scenariuszu dystrybutorów podzielono na podobne grupy, czyli klastry na podstawie podobieństwa – w tym przypadku bliskości siebie. Do tych klastrów zastosowano funkcję Centroid, aby zidentyfikować pięć współrzędnych mapowania. Te współrzędne wskazują początkową centralną lokalizację, w której można budować lub lokalizować magazyny. Funkcja centroid jest stosowana do wykresu **mapy**, dzięki czemu użytkownicy aplikacji mogą wizualizować położenie centroidów względem otaczających punktów danych klastra. Wynikowe współrzędne reprezentują potencjalne lokalizacje magazynów, które mogą zminimalizować koszty dostawy do dystrybutorów w stanie Nowy Jork.

Aplikacja: Przykład analizy z wykorzystaniem funkcji KMeans i centroid



Zestaw danych Distributor: Wbudowane ładowanie dla edytora ładowania danych w Qlik Sense

```
DistributorData: Load * Inline [ id,first_name,last_
name,telephone,address,city,state,zip,latitude,longitude 1,Kaiya,Snow,(716) 201-1212,6231
Tonawanda Creek Rd #APT 308,Lockport,NY,14094,43.08926,-78.69313 2,Dean,Roy,(716) 201-
1588,6884 E High St,Lockport,NY,14094,43.16245,-78.65036 3,Eden,Paul,(716) 202-4596,4647
Southwestern Blvd #APT 350,Hamburg,NY,14075,42.76003,-78.83194 4,Bryanna,Higgins,(716) 203-
7041,418 Park Ave,Dunkirk,NY,14048,42.48279,-79.33088 5,Elisabeth,Lee,(716) 203-7043,36 E
Courtney St,Dunkirk,NY,14048,42.48299,-79.31928 6,Skylar,Robinson,(716) 203-7166,26 Greco
Ln,Dunkirk,NY,14048,42.4612095,-79.3317925 7,Cody,Bailey,(716) 203-7201,114 Lincoln
Ave,Dunkirk,NY,14048,42.4801269,-79.322232 8,Dario,Sims,(408) 927-1606,N Castle
Dr,Armonk,NY,10504,41.11979,-73.714864 9,Deacon,Hood,(410) 244-6221,4856 44th
St,Woodside,NY,11377,40.748372,-73.905445 10,Zackery,Levy,(410) 363-8874,61 Executive
Blvd,Farmingdale,NY,11735,40.7197457,-73.430239 11,Rey,Hawkins,(412) 344-8687,4585 Shimerville
Rd,Clarence,NY,14031,42.972075,-78.6592452 12,Phillip,Howard,(413) 269-4049,464 Main St
#101,Port Washington,NY,11050,40.8273756,-73.7009971 13,Shirley,Tyler,(434) 985-8943,114 Glann
Rd,Apalachin,NY,13732,42.0482515,-76.1229725 14,Aniyah,Jarvis,(440) 244-1808,87 N Middletown
Rd,Pearl River,NY,10965,41.0629,-74.0159 15,Alayna,Woodard,(478) 335-3704,70 W Red Oak Ln,West
Harrison,NY,10604,41.0162722,-73.7234926 16,Jermaine,Lambert,(508) 561-9836,24 Kellogg Rd,New
Hartford,NY,13413,43.0555739,-75.2793197 17,Harper,Gibbs,(239) 466-0238,Po Box
33,Cottekill,NY,12419,41.853392,-74.106082 18,Osvaldo,Graham,(252) 246-0816,6878 Sand Hill
Rd,East Syracuse,NY,13057,43.073215,-76.081448 19,Roberto,Wade,(270) 469-1211,3936 Holley
Rd,Moravia,NY,13118,42.713044,-76.481227 20,Kate,Mcguire,(270) 788-3080,6451 State 64 Rte
#3,Naples,NY,14512,42.707366,-77.380489 21,Dale,Andersen,(281) 480-5690,205 W Service
Rd,Champlain,NY,12919,44.9645392,-73.4470831 22,Lorelai,Burch,(302) 644-2133,1 Brewster
St,Glen Cove,NY,11542,40.865177,-73.633019 23,Amiyah,Flowers,(303) 223-0055,46600 Us
Interstate 81 Rte,Alexandria Bay,NY,13607,44.309626,-75.988365 24,Mckinley,Clements,(303) 918-
3230,200 Summit Lake Dr,Valhalla,NY,10595,41.101145,-73.778298 25,Marc,Gibson,(607) 203-
1233,25 Robinson St,Binghamton,NY,13901,42.107416,-75.901614 26,Kali,Norman,(607) 203-1400,1
Ely Park Blvd #APT 15,Binghamton,NY,13905,42.125866,-75.925026 27,Laci,Cain,(607) 203-1437,16
Zimmer Road,Kirkwood,NY,13795,42.066516,-75.792627 28,Mohammad,Perez,(607) 203-1652,71
```

Endicott Ave #APT 12,Johnson City,NY,13790,42.111894,-75.952187 29,Izabelle,Pham,(607) 204-0392,434 State 369 Rte,Port Crane,NY,13833,42.185838,-75.823074 30,Kiley,Mays,(607) 204-0870,244 Ballyhack Rd #14,Port Crane,NY,13833,42.175612,-75.814917 31,Peter,Trevino,(607) 205-1374,125 Melbourne St.,Vestal,NY,13850,42.080254,-76.051124 32,Ani,Francis,(607) 208-4067,48 Caswell St,Afton,NY,13730,42.232065,-75.525674 33,Jared,Sheppard,(716) 386-3002,4709 430th Rte,Bemus Point,NY,14712,42.162175,-79.39176 34,Dulce,Atkinson,(914) 576-2266,501 Pelham Rd,New Rochelle,NY,10805,40.895449,-73.782602 35,Jayla,Beasley,(716) 526-1054,5010 474th Rte,Ashville,NY,14710,42.096859,-79.375561 36,Dane,Donovan,(718) 545-3732,5014 31st Ave,Woodside,NY,11377,40.756967,-73.909506 37,Brendon,Clay,(585) 322-7780,133 Cummings Ave,Gainesville,NY,14066,42.664309,-78.085651 38,Asia,Nunez,(718) 426-1472,2407 Gilmore ,East Elmhurst,NY,11369,40.766662,-73.869185 39,Dawson,Odonnell,(718) 342-2179,5019 H Ave,Brooklyn,NY,11234,40.633245,-73.927591 40,Kyle,Collins,(315) 733-7078,502 Rockhaven Rd,Utica,NY,13502,43.129184,-75.226726 41,Eliza,Hardin,(315) 331-8072,502 Sladen Place,West Point,NY,10996,41.3993,-73.973003 42,Kasen,Klein,(518) 298-4581,2407 Lake Shore Rd,Chazy,NY,12921,44.925561,-73.387373 43,Reuben,Bradford,(518) 298-4581,33 Lake Flats Dr,Champlain,NY,12919,44.928092,-73.387884 44,Henry,Grimes,(518) 523-3990,2407 Main St,Lake Placid,NY,12946,44.291487,-73.98474 45,Kyan,Livingston,(518) 585-7364,241 Alexandria Ave,Ticonderoga,NY,12883,43.836553,-73.43155 46,Kaitlyn,Short,(516) 678-3189,241 Chance Dr,Oceanside,NY,11572,40.638534,-73.63079 47,Damaris,Jacobs,(914) 664-5331,241 Claremont Ave,Mount Vernon,NY,10552,40.919852,-73.827848 48,Alivia,Schroeder,(315) 469-4473,241 Lafayette Rd,Syracuse,NY,13205,42.996446,-76.12957 49,Bridget,Strong,(315) 298-4355,241 Maltby Rd,Pulaski,NY,13142,43.584966,-76.136317 50,Francis,Lee,(585) 201-7021,166 Ross St,Batavia,NY,14020,43.0031502,-78.17487 51,Makaila,Phelps,(585) 201-7422,58 S Main St,Batavia,NY,14020,42.99941,-78.1939285 52,Jazlynn,Stephens,(585) 203-1087,1 Sinclair Dr,Pittsford,NY,14534,43.084157,-77.545452 53,Ryann,Randolph,(585) 203-1519,331 Eaglehead Rd,East Rochester,NY,14445,43.10785,-77.475552 54,Rosa,Baker,(585) 204-4011,42 Ossian St,Dansville,NY,14437,42.560761,-77.70088 55,Marcel,Barry,(585) 204-4013,42 Jefferson St,Dansville,NY,14437,42.557735,-77.702983 56,Dennis,Schmitt,(585) 204-4061,750 Dansville Mount Morris Rd,Dansville,NY,14437,42.584458,-77.741648 57,Cassandra,Kim,(585) 204-4138,3 Perine Ave APT1,Dansville,NY,14437,42.562865,-77.69661 58,Kolton,Jacobson,(585) 206-5047,4925 Upper Holly Rd,Holley,NY,14470,43.175957,-78.074465 59,Nathanael,Donovan,(718) 393-3501,9604 57th Ave,Corona,NY,11373,40.736077,-73.864858 60,Robert,Frazier,(718) 271-3067,300 56th Ave,Corona,NY,11373,40.735304,-73.873997 61,Jessie,Mora,(315) 405-8991,9607 Forsyth Loop,Watertown,NY,13603,44.036466,-75.833437 62,Martha,Rollins,(347) 242-2642,22 Main St,Corona,NY,11373,40.757727,-73.829331 63,Emely,Townsend,(718) 699-0751,60 Sanford Ave,Corona,NY,11373,40.755466,-73.831029 64,Kylie,Cooley,(347) 561-7149,9608 95th Ave,Ozone Park,NY,11416,40.687564,-73.845715 65,Wendy,Cameron,(585) 571-4185,9608 Union St,Scottsville,NY,14546,43.013327,-77.7907839 66,Kayley,Peterson,(718) 654-5027,961 E 230th St,Bronx,NY,10466,40.889275,-73.850555 67,Camden,Ochoa,(718) 760-8699,59 Vark St,Yonkers,NY,10701,40.929322,-73.89957 68,Priscilla,Castillo,(910) 326-7233,9359 Elm St,Chadwicks,NY,13319,43.024902,-75.26886 69,Dana,Schultz,(913) 322-4580,99 Washington Ave,Hastings on Hudson,NY,10706,40.99265,-73.879748 70,Blaze,Medina,(914) 207-0015,60 Elliott Ave,Yonkers,NY,10705,40.921498,-73.896682 71,Finnegan,Tucker,(914) 207-0015,90 Hillside Drive,Yonkers,NY,10705,40.922514,-73.892911 72,Pranav,Palmer,(914) 214-8376,5 Bruce Ave,Harrison,NY,10528,40.970916,-73.711493 73,Kolten,Wong,(914) 218-8268,70 Barker St,Mount Kisco,NY,10549,41.211993,-73.723202 74,Jasiah,Vazquez,(914) 231-5199,30 Broadway,Dobbs Ferry,NY,10522,41.004629,-73.879825 75,Lamar,Pierce,(914) 232-0380,68 Ridge Rd,Katonah,NY,10536,41.256662,-73.707964 76,Carla,Coffey,(914) 232-0469,197 Beaver Dam Rd,Katonah,NY,10536,41.247934,-73.664363 77,Brooklynn,Harmon,(716) 595-3227,8084 Glasgow Rd,Cassadega,NY,14718,42.353861,-79.329558 78,Raquel,Hodges,(585) 398-8125,809 County Road ,Victor,NY,14564,43.011745,-77.398806 79,Jerimiah,Gardner,(585) 787-9127,809 Houston Rd,Webster,NY,14580,43.224204,-77.491353 80,Clarence,Hammond,(720) 746-1619,809 Pierpont Ave,Piermont,NY,10968,41.0491181,-73.918622 81,Rhys,Gill,(518) 427-7887,81 Columbia St,Albany,NY,12210,42.652824,-73.752096 82,Edith,Parrish,(845) 452-7621,81 Glenwood Ave,Poughkeepsie,NY,12603,41.691058,-73.910829 83,Kobe,Mcintosh,(845) 371-1101,81 Heitman Dr,Spring Valley,NY,10977,41.103227,-74.054396 84,Ayden,Waters,(516) 796-2722,81 Kingfisher

```
Rd,Levittown,NY,11756,40.738939,-73.52826 85,Francis,Rogers,(631) 427-7728,81 Knollwood
Ave,Huntington,NY,11743,40.864905,-73.426107 86,Jaden,Landry,(716) 496-4038,12839 39th
Rte,Chaffee,NY,14030,43.527396,-73.462786 87,Giancarlo,Campos,(518) 885-5717,1284 Saratoga
Rd,Ballston Spa,NY,12020,42.968594,-73.862847 88,Eduardo,Contreras,(716) 285-8987,1285
Saunders Sett Rd,Niagara Falls,NY,14305,43.122963,-79.029274 89,Gabriela,Davidson,(716) 267-
3195,1286 Mee Rd,Falconer,NY,14733,42.147339,-79.137976 90,Evangeline,Case,(518) 272-9435,1287
2nd Ave,Watervliet,NY,12189,42.723132,-73.703818 91,Tyrone,Ellison,(518) 843-4691,1287 Midline
Rd,Amsterdam,NY,12010,42.9730876,-74.1700608 92,Bryce,Bass,(518) 943-9549,1288 Leeds Athens
Rd,Athens,NY,12015,42.259381,-73.876897 93,Londyn,Butler,(518) 922-7095,129 Argersinger
Rd,Fultonville,NY,12072,42.910969,-74.441917 94,Graham,Becker,(607) 655-1318,129 Baker
Rd,Windsor,NY,13865,42.107271,-75.66408 95,Rolando,Fitzgerald,(315) 465-4166,17164 County 90
Rte,Mannsville,NY,13661,43.713443,-76.06232 96,Grant,Hoover,(518) 692-8363,1718 County 113
Rte,Schaghticote,NY,12154,42.900648,-73.585036 97,Mark,Goodwin,(631) 584-6761,172 Cambon
Ave,Saint James,NY,11780,40.871152,-73.146032 98,Deacon,Cantu,(845) 221-7940,172 Carpenter
Rd,Hopewell Junction,NY,12533,41.57388,-73.77609 99,Tristian,Walsh,(516) 997-4750,172 E Cabot
Ln,Westbury,NY,11590,40.7480397,-73.54819 100,Abram,Alexander,(631) 588-3817,172 Lorenzo
Cir,Ronkonkoma,NY,11779,40.837123,-73.09367 101,Lesly,Bush,(516) 489-3791,172 Nassau
Blvd,Garden City,NY,11530,40.71147,-73.660753 102,Pamela,Espinoza,(716) 201-1520,172 Niagara
St ,Lockport,NY,14094,43.169871,-78.70093 103,Bryanna,Newton,(914) 328-4332,172 Warren
Ave,White Plains,NY,10603,41.047207,-73.79572 104,Marcelo,Schmitt,(315) 393-4432,319 Mansion
Ave,Ogdensburg,NY,13669,44.690246,-75.49992 105,Layton,Valenzuela,(631) 676-2113,319
Singingwood Dr,Holbrook,NY,11741,40.801391,-73.058993 106,Roderick,Rocha,(518) 671-6037,319
Warren St,Hudson,NY,12534,42.252527,-73.790629 107,Camryn,Terrell,(315) 635-1680,3192 Olive
Dr,Baldinsville,NY,13027,43.136843,-76.260303 108,Summer,Callahan,(585) 394-4195,3192 Smith
Road,Canandaigua,NY,14424,42.875457,-77.228039 109,Pierre,Novak,(716) 665-2524,3194 Falconer
Kimball Stand Rd,Falconer,NY,14733,42.138439,-79.211091 110,Kennedi,Fry,(315) 543-2301,32
College Rd,Selden,NY,11784,40.861624,-73.04757 111,Wyatt,Pruitt,(716) 681-4042,277 Ransom
Rd,Lancaster ,NY,14086,42.87702,-78.591302 112,Lilly,Jensen,(631) 841-0859,2772 Schliegel
Blvd,Amityville,NY,11701,40.708021,-73.413015 113,Tristin,Hardin,(631) 920-0927,278 Fulton
Street,West Babylon,NY,11704,40.733578,-73.357321 114,Tanya,Stafford,(716) 484-0771,278
Sampson St,Jamestown,NY,14701,42.0797,-79.247805 115,Paris,Cordova,(607) 589-4857,278 Washburn
Rd,Spencer,NY,14883,42.225046,-76.510257 116,Alfonso,Morse,(718) 359-5582,200 Colden
St,Flushing,NY,11355,40.750403,-73.822752 117,Maurice,Hooper,(315) 595-6694,4435 Italy Hill
Rd,Branchport,NY,14418,42.597957,-77.199267 118,Iris,Wolf,(607) 539-7288,444 Harford
Rd,Brooktondale,NY,14817,42.392164,-76.30756 ];
```

KMeans2D – funkcja wykresu

KMeans2D() poddaje ocenie wiersze wykresu, stosując algorytm centroidów oraz wyświetlając dla każdego wiersza wykresu identyfikator klastra, do którego został przypisany ten punkt danych. Kolumny wykorzystywane przez algorytm grupowania są określane przez odpowiednio parametry coordinate_1 i coordinate_2. Oba te parametry są agregacjami. Liczba tworzonych klastrów jest określana przez parametr num_clusters. Dane można opcjonalnie znormalizować za pomocą parametru norm.

KMeans2D zwraca jedną wartość na punkt danych. Zwrócona wartość jest podwójna i stanowi wartość liczby całkowitej odpowiadającej klastrowi, do którego został przypisany każdy punkt danych.

Składnia:

```
KMeans2D(num_clusters, coordinate_1, coordinate_2 [, norm])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
num_clusters	Liczba całkowita określająca liczbę klastrów.
coordinate_1	Agregacja obliczająca pierwszą współrzędną, zwykle osi X wykresu punktowego, która może zostać utworzona z wykresu. Dodatkowy paramert, coordinate_2, oblicza drugą współrzędną.
norm	Opcjonalna metoda normalizacji stosowana do zestawów danych przed algorytmem centroidów. Możliwe wartości: 0 lub „none” w przypadku braku normalizacji 1 lub „zscore” w przypadku normalizacji z-score 2 lub „minmax” dla normalizacji min.-maks. Jeśli nie podano żadnego parametru lub jeśli podany parametr jest nieprawidłowy, żadna normalizacja nie jest stosowana. Normalizacja z-score normalizuje dane w oparciu o średnią cechy i odchylenie standardowe. Normalizacja z-score nie gwarantuje, że każda cecha będzie mieć taką samą skalę, ale jest lepszym podejściem niż min.-maks. w przypadku wartości odstających. Normalizacja min.-maks. zapewnia, że cechy mają tę samą skalę, biorąc minimalne i maksymalne wartości każdej z nich i przeliczając każdy punkt danych.

Przykład: wyrażenie wykresu
W tym przykładzie ukazane jest tworzenie wykresu punktowego z użyciem zestawu danych *Iris*, a następnie wykorzystanie KMeans do kolorowania danych według wyrażenia.

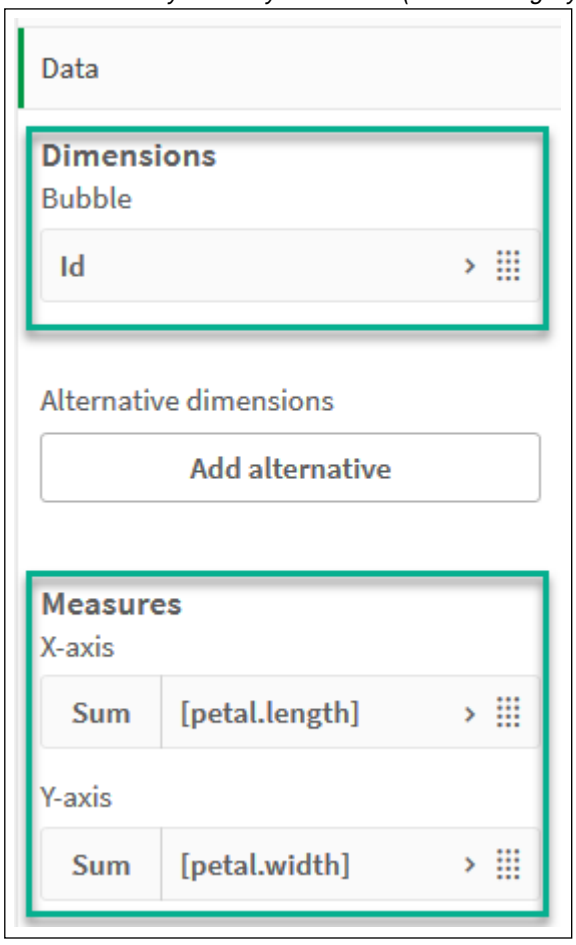
Można także utworzyć argument *num_clusters*, a następnie użyć pola wprowadzania zmiennej, aby zmienić liczbę klastrów.

Zestaw danych *Iris* jest ogólnodostępny w różnorodnych formatach. Zapewniono dane w formie wbudowanej tabeli do ładowania z użyciem edytora ładowania danych w Qlik Sense. Należy zauważyć, że w tym przykładzie do tabeli danych dodano *identyfikator* kolumny.

Po załadowaniu danych w Qlik Sense należy podjąć następujące kroki:

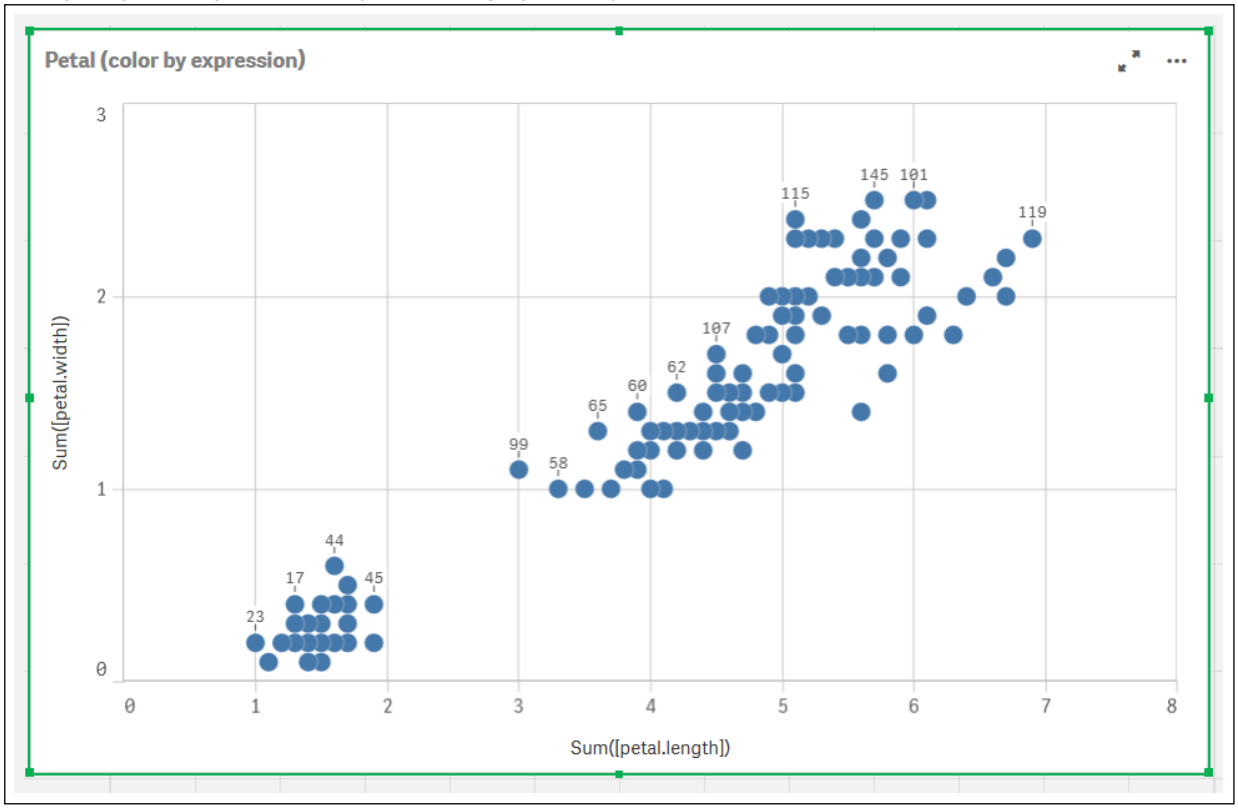
1. Przeciągnij wykres **punktowy** do nowego arkusza. Nazwij wykres *Petal (kolor według wyrażenia)*.
2. Utwórz zmienną, aby określić liczbę klastrów. Dla zmiennej **Nazwa** wprowadź *KmeansPetalClusters*. Dla zmiennej **Definicja** wprowadź *=2*.
3. Skonfiguruj opcję **Dane** dla wykresu:
   i. Pod obszarem **Wymiary** wybierz *identyfikator* dla pola **Bąbelek**. Wprowadź identyfikator klastra dla Etykiety.
   ii. Pod obszarem **Miary** wybierz *Sum([petal.length])* dla wyrażenia **Oś X**.
   iii. Pod obszarem **Miary** wybierz *Sum([petal.width])* dla wyrażenia **Oś Y**.

   Ustawienia danych dla wykresu Petal (kolor według wyrażenia)



Punkty danych są naniesione na wykres.

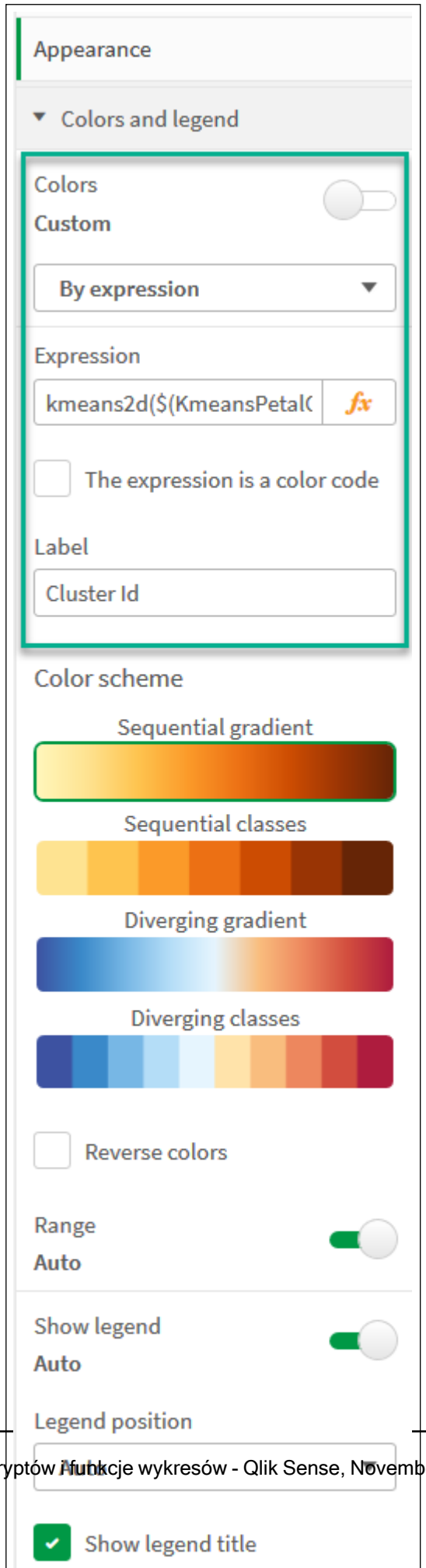
Punkty danych na wykresie Petal (kolor według wyrażenia)



4. Skonfiguruj opcję **Wygląd** dla wykresu:
    i. W opcji **Kolory i legenda** wybierz wartość **Niestandardowe** dla **Kolorów**.
    ii. Wybierz opcję kolorowania wykresu **Według wyrażenia**.
    iii. Wprowadź następującą wartość dla **Wyrażenia**: *kmeans2d($(KmeansPetalClusters), Sum ([petal.length]), Sum([petal.width]))*
    Należy zauważyć, że wartość *KmeansPetalClusters* jest zmienną ustawianą na *2*.
    Alternatywnie wprowadź następującą wartość: *kmeans2d(2, Sum([petal.length]), Sum ([petal.width]))*
    iv. Usuń zaznaczenie pola wyboru **Wyrażenie jest kodem koloru**.

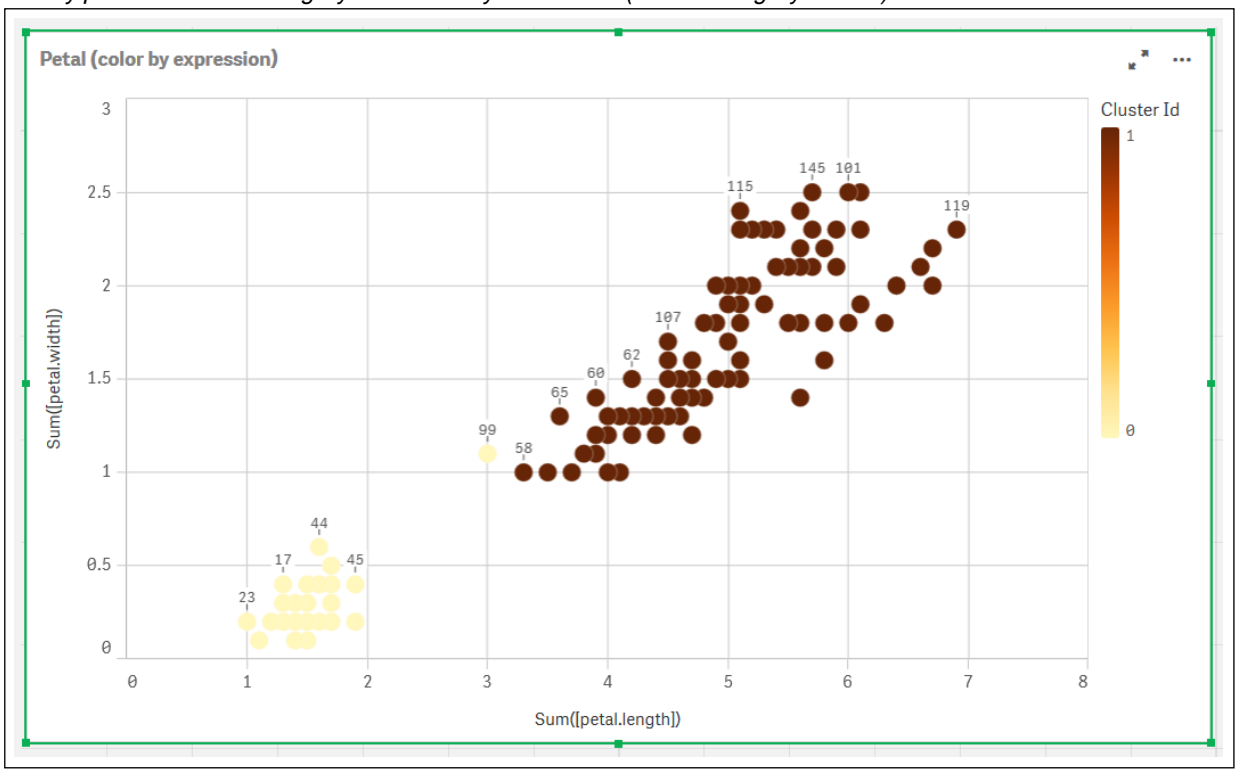
v. Wprowadź następujące dla opcji **Etykieta**: *Identyfikator klastra*

Ustawienia wyglądu dla wykresu Petal (kolor według wyrażenia)


Appearance
Colors and legend
Colors
Custom
By expression
Expression
kmeans2d($(KmeansPetal(
fx
The expression is a color code
Label
Cluster Id
Color scheme
Sequential gradient
Sequential classes
Diverging gradient
Diverging classes
Reverse colors
Range
Auto
Show legend
Auto
Legend position
Auto
Show legend title

Dwa klastry na wykresie są pokolorowane według wyrażenia KMeans.

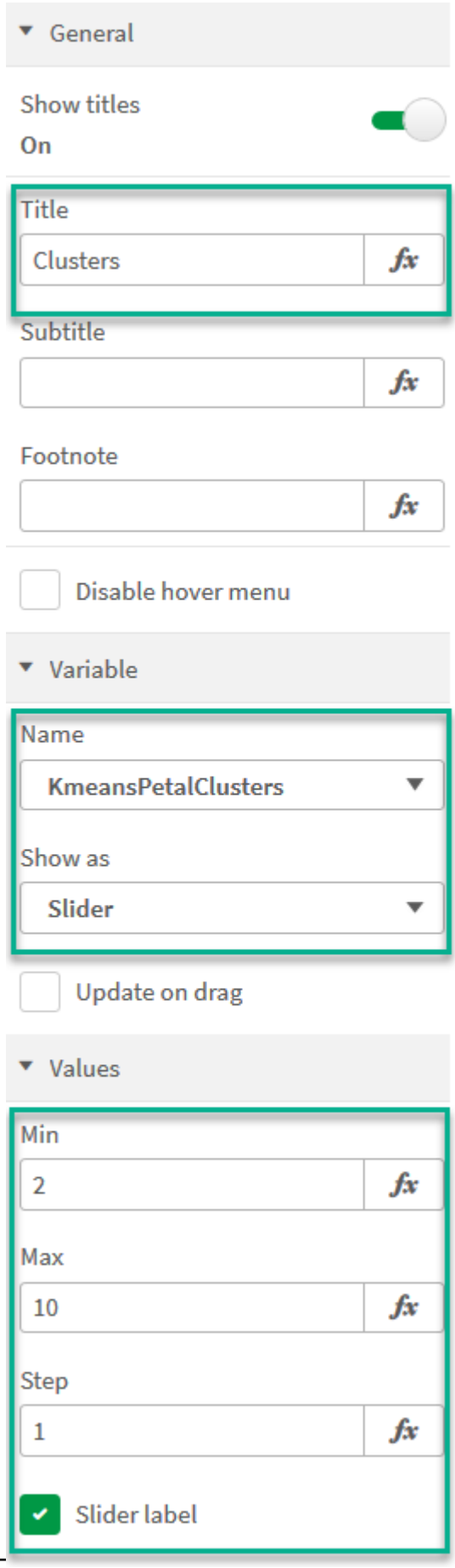
Klastry pokolorowane według wyrażenia na wykresie Petal (kolor według wyrażenia)



5. Dodaj pole **Wprowadzanie zmiennych** dla liczby klastrów.
    i. Pod obszarem **Obiekty niestandardowe** w panelu **Zasoby** wybierz **pakiet Qlik Dashboard bundle**. W przypadku braku dostępu do pakietu dashboard bundle nadal można zmieniać liczbę klastrów, korzystając z utworzonej zmiennej lub bezpośrednio jako liczbę całkowitą w wyrażeniu.
    ii. Przeciągnij pole **Wprowadzanie zmiennych** do arkusza.
    iii. W sekcji **Wygląd** kliknij pozycję **Ogólny**.
    iv. Dla obszaru **Tytuł** wprowadź następujące dane: *Klastry*
    v. Kliknij pozycję **Zmienna**.
    vi. Wybierz następującą zmienną dla obszaru **Nazwa**: *KmeansPetalClusters*.
    vii. Wybierz **Suwak** dla opcji **Pokaż jako**.

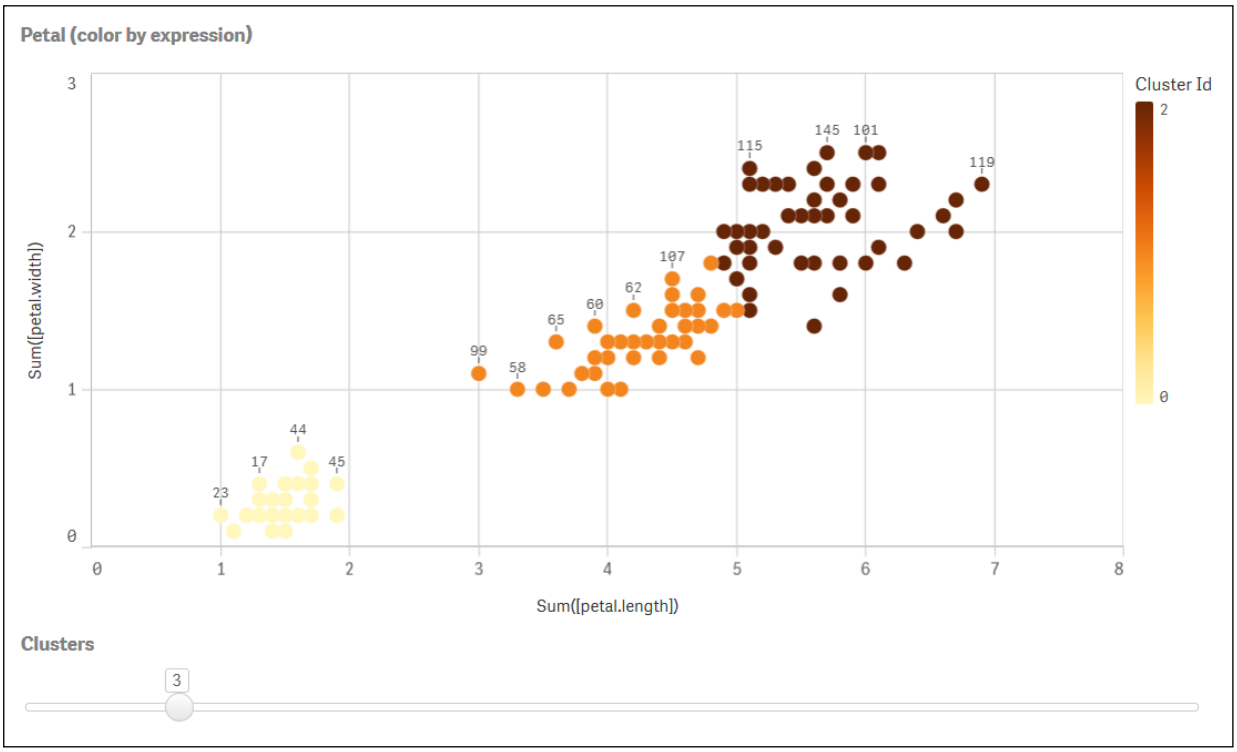
viii. Wybierz **Wartości** i skonfiguruj ustawienia zgodnie z wymaganiami

Wygląd dla pola wprowadzania zmiennych Klastry


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

Po zakończeniu edycji można zmienić liczbę klastrów za pomocą suwaka w polu wprowadzania zmiennych *Klastry*.

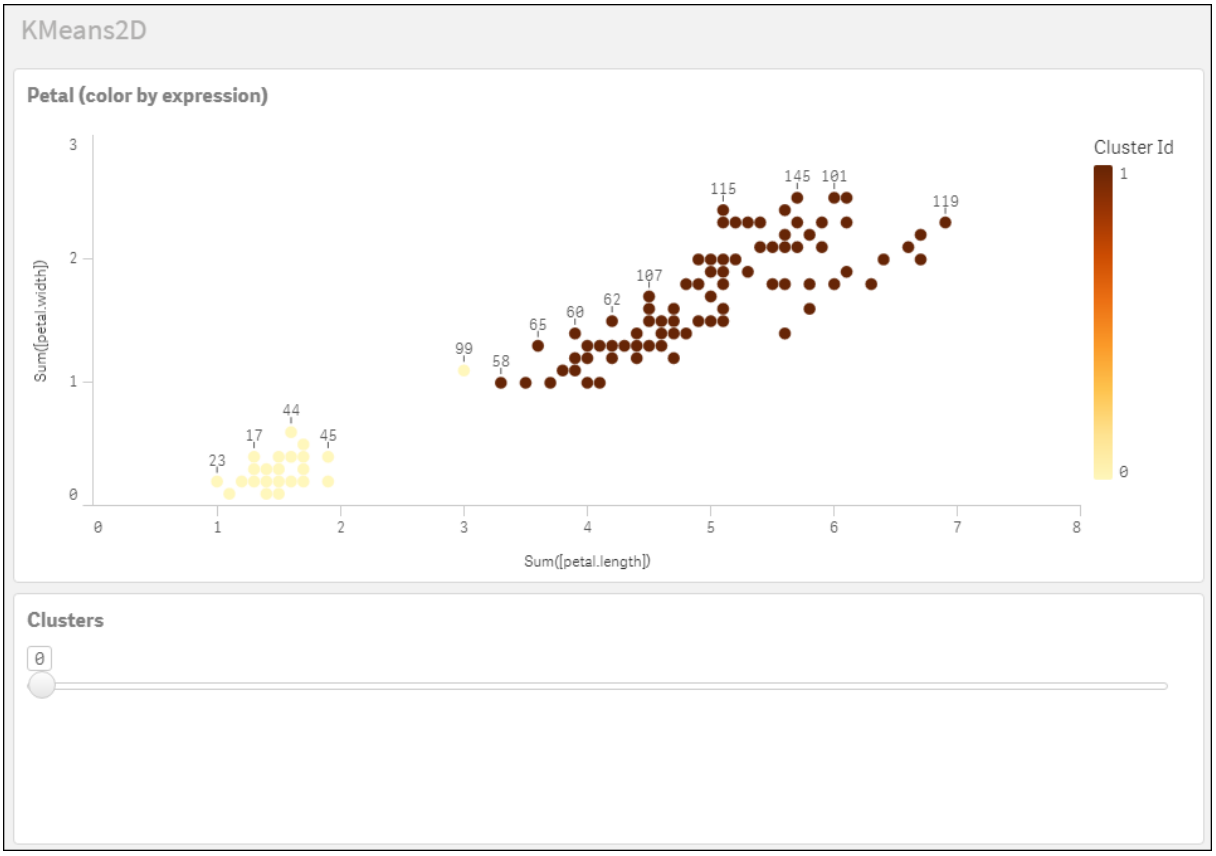
Klastry pokolorowane według wyrażenia na wykresie Petal (kolor według wyrażenia)



Automatyczne grupowanie

Funkcje **KMeans** obsługują automatyczne grupowanie za pomocą metody zwanej różnicą głębokości (DeD). Gdy użytkownik ustawia liczbę klastrów na 0, określana jest optymalna liczba klastrów dla tego zestawu danych. Należy zauważyć, że liczba całkowita dla liczby klastrów (*k*) nie jest wyraźnie zwracana, ale obliczana w ramach algorytmu KMeans. Na przykład jeśli 0 jest określone w funkcji dla wartości *KmeansPetalClusters* lub ustalone poprzez pole wprowadzania zmiennych, przypisania klastrów są automatyczne obliczane dla zestawu danych opartego na optymalnej liczbie klastrów.

Metoda różnicy głębokości KMeans określa optymalną liczbę klastrów, gdy (k) ma wartość 0



Zestaw danych Iris: Wbudowane ładowanie dla edytora ładowania danych w Qlik Sense

```
IrisData: Load * Inline [ sepal.length, sepal.width, petal.length, petal.width, variety, id
5.1, 3.5, 1.4, 0.2, Setosa, 1 4.9, 3, 1.4, 0.2, Setosa, 2 4.7, 3.2, 1.3, 0.2, Setosa, 3 4.6,
3.1, 1.5, 0.2, Setosa, 4 5, 3.6, 1.4, 0.2, Setosa, 5 5.4, 3.9, 1.7, 0.4, Setosa, 6 4.6, 3.4,
1.4, 0.3, Setosa, 7 5, 3.4, 1.5, 0.2, Setosa, 8 4.4, 2.9, 1.4, 0.2, Setosa, 9 4.9, 3.1, 1.5,
0.1, Setosa, 10 5.4, 3.7, 1.5, 0.2, Setosa, 11 4.8, 3.4, 1.6, 0.2, Setosa, 12 4.8, 3, 1.4,
0.1, Setosa, 13 4.3, 3, 1.1, 0.1, Setosa, 14 5.8, 4, 1.2, 0.2, Setosa, 15 5.7, 4.4, 1.5, 0.4,
Setosa, 16 5.4, 3.9, 1.3, 0.4, Setosa, 17 5.1, 3.5, 1.4, 0.3, Setosa, 18 5.7, 3.8, 1.7, 0.3,
Setosa, 19 5.1, 3.8, 1.5, 0.3, Setosa, 20 5.4, 3.4, 1.7, 0.2, Setosa, 21 5.1, 3.7, 1.5, 0.4,
Setosa, 22 4.6, 3.6, 1, 0.2, Setosa, 23 5.1, 3.3, 1.7, 0.5, Setosa, 24 4.8, 3.4, 1.9, 0.2,
Setosa, 25 5, 3, 1.6, 0.2, Setosa, 26 5, 3.4, 1.6, 0.4, Setosa, 27 5.2, 3.5, 1.5, 0.2, Setosa,
28 5.2, 3.4, 1.4, 0.2, Setosa, 29 4.7, 3.2, 1.6, 0.2, Setosa, 30 4.8, 3.1, 1.6, 0.2, Setosa,
31 5.4, 3.4, 1.5, 0.4, Setosa, 32 5.2, 4.1, 1.5, 0.1, Setosa, 33 5.5, 4.2, 1.4, 0.2, Setosa,
34 4.9, 3.1, 1.5, 0.1, Setosa, 35 5, 3.2, 1.2, 0.2, Setosa, 36 5.5, 3.5, 1.3, 0.2, Setosa, 37
4.9, 3.1, 1.5, 0.1, Setosa, 38 4.4, 3, 1.3, 0.2, Setosa, 39 5.1, 3.4, 1.5, 0.2, Setosa, 40 5,
3.5, 1.3, 0.3, Setosa, 41 4.5, 2.3, 1.3, 0.3, Setosa, 42 4.4, 3.2, 1.3, 0.2, Setosa, 43 5,
3.5, 1.6, 0.6, Setosa, 44 5.1, 3.8, 1.9, 0.4, Setosa, 45 4.8, 3, 1.4, 0.3, Setosa, 46 5.1,
3.8, 1.6, 0.2, Setosa, 47 4.6, 3.2, 1.4, 0.2, Setosa, 48 5.3, 3.7, 1.5, 0.2, Setosa, 49 5,
3.3, 1.4, 0.2, Setosa, 50 7, 3.2, 4.7, 1.4, Versicolor, 51 6.4, 3.2, 4.5, 1.5, Versicolor, 52
6.9, 3.1, 4.9, 1.5, Versicolor, 53 5.5, 2.3, 4, 1.3, Versicolor, 54 6.5, 2.8, 4.6, 1.5,
Versicolor, 55 5.7, 2.8, 4.5, 1.3, Versicolor, 56 6.3, 3.3, 4.7, 1.6, Versicolor, 57 4.9, 2.4,
3.3, 1, Versicolor, 58 6.6, 2.9, 4.6, 1.3, Versicolor, 59 5.2, 2.7, 3.9, 1.4, Versicolor, 60
5, 2, 3.5, 1, Versicolor, 61 5.9, 3, 4.2, 1.5, Versicolor, 62 6, 2.2, 4, 1, Versicolor, 63
6.1, 2.9, 4.7, 1.4, Versicolor, 64 5.6, 2.9, 3.6, 1.3, Versicolor, 65 6.7, 3.1, 4.4, 1.4,
Versicolor, 66 5.6, 3, 4.5, 1.5, Versicolor, 67 5.8, 2.7, 4.1, 1, Versicolor, 68 6.2, 2.2,
```

```
4.5, 1.5, Versicolor, 69 5.6, 2.5, 3.9, 1.1, Versicolor, 70 5.9, 3.2, 4.8, 1.8, Versicolor, 71
6.1, 2.8, 4, 1.3, Versicolor, 72 6.3, 2.5, 4.9, 1.5, Versicolor, 73 6.1, 2.8, 4.7, 1.2,
Versicolor, 74 6.4, 2.9, 4.3, 1.3, Versicolor, 75 6.6, 3, 4.4, 1.4, Versicolor, 76 6.8, 2.8,
4.8, 1.4, Versicolor, 77 6.7, 3, 5, 1.7, Versicolor, 78 6, 2.9, 4.5, 1.5, Versicolor, 79 5.7,
2.6, 3.5, 1, Versicolor, 80 5.5, 2.4, 3.8, 1.1, Versicolor, 81 5.5, 2.4, 3.7, 1, Versicolor,
82 5.8, 2.7, 3.9, 1.2, Versicolor, 83 6, 2.7, 5.1, 1.6, Versicolor, 84 5.4, 3, 4.5, 1.5,
Versicolor, 85 6, 3.4, 4.5, 1.6, Versicolor, 86 6.7, 3.1, 4.7, 1.5, Versicolor, 87 6.3, 2.3,
4.4, 1.3, Versicolor, 88 5.6, 3, 4.1, 1.3, Versicolor, 89 5.5, 2.5, 4, 1.3, Versicolor, 90
5.5, 2.6, 4.4, 1.2, Versicolor, 91 6.1, 3, 4.6, 1.4, Versicolor, 92 5.8, 2.6, 4, 1.2,
Versicolor, 93 5, 2.3, 3.3, 1, Versicolor, 94 5.6, 2.7, 4.2, 1.3, Versicolor, 95 5.7, 3, 4.2,
1.2, Versicolor, 96 5.7, 2.9, 4.2, 1.3, Versicolor, 97 6.2, 2.9, 4.3, 1.3, Versicolor, 98 5.1,
2.5, 3, 1.1, Versicolor, 99 5.7, 2.8, 4.1, 1.3, Versicolor, 100 6.3, 3.3, 6, 2.5, Virginica,
101 5.8, 2.7, 5.1, 1.9, Virginica, 102 7.1, 3, 5.9, 2.1, Virginica, 103 6.3, 2.9, 5.6, 1.8,
Virginica, 104 6.5, 3, 5.8, 2.2, Virginica, 105 7.6, 3, 6.6, 2.1, Virginica, 106 4.9, 2.5,
4.5, 1.7, Virginica, 107 7.3, 2.9, 6.3, 1.8, Virginica, 108 6.7, 2.5, 5.8, 1.8, Virginica, 109
7.2, 3.6, 6.1, 2.5, Virginica, 110 6.5, 3.2, 5.1, 2, Virginica, 111 6.4, 2.7, 5.3, 1.9,
Virginica, 112 6.8, 3, 5.5, 2.1, Virginica, 113 5.7, 2.5, 5, 2, Virginica, 114 5.8, 2.8, 5.1,
2.4, Virginica, 115 6.4, 3.2, 5.3, 2.3, Virginica, 116 6.5, 3, 5.5, 1.8, Virginica, 117 7.7,
3.8, 6.7, 2.2, Virginica, 118 7.7, 2.6, 6.9, 2.3, Virginica, 119 6, 2.2, 5, 1.5, Virginica,
120 6.9, 3.2, 5.7, 2.3, Virginica, 121 5.6, 2.8, 4.9, 2, Virginica, 122 7.7, 2.8, 6.7, 2,
Virginica, 123 6.3, 2.7, 4.9, 1.8, Virginica, 124 6.7, 3.3, 5.7, 2.1, Virginica, 125 7.2, 3.2,
6, 1.8, Virginica, 126 6.2, 2.8, 4.8, 1.8, Virginica, 127 6.1, 3, 4.9, 1.8, Virginica, 128
6.4, 2.8, 5.6, 2.1, Virginica, 129 7.2, 3, 5.8, 1.6, Virginica, 130 7.4, 2.8, 6.1, 1.9,
Virginica, 131 7.9, 3.8, 6.4, 2, Virginica, 132 6.4, 2.8, 5.6, 2.2, Virginica, 133 6.3, 2.8,
5.1, 1.5, Virginica, 134 6.1, 2.6, 5.6, 1.4, Virginica, 135 7.7, 3, 6.1, 2.3, Virginica, 136
6.3, 3.4, 5.6, 2.4, Virginica, 137 6.4, 3.1, 5.5, 1.8, Virginica, 138 6, 3, 4.8, 1.8,
Virginica, 139 6.9, 3.1, 5.4, 2.1, Virginica, 140 6.7, 3.1, 5.6, 2.4, Virginica, 141 6.9, 3.1,
5.1, 2.3, Virginica, 142 5.8, 2.7, 5.1, 1.9, Virginica, 143 6.8, 3.2, 5.9, 2.3, Virginica, 144
6.7, 3.3, 5.7, 2.5, Virginica, 145 6.7, 3, 5.2, 2.3, Virginica, 146 6.3, 2.5, 5, 1.9,
Virginica, 147 6.5, 3, 5.2, 2, Virginica, 148 6.2, 3.4, 5.4, 2.3, Virginica, 149 5.9, 3, 5.1,
1.8, Virginica, 150 ];
```

KMeansND – funkcja wykresu

KMeansND() poddaje ocenie wiersze wykresu, stosując algorytm centroidów oraz wyświetlając dla każdego wiersza wykresu identyfikator klastra, do którego został przypisany ten punkt danych. Kolumny wykorzystywane przez algorytm grupowania są określone przez odpowiednio parametry coordinate_1 i coordinate_2 itd., aż do n kolumn. Wszystkie te parametry są agregacjami. Liczba tworzonych klastrów jest określana przez parametr num_clusters.

KMeansND zwraca jedną wartość na punkt danych. Zwrócona wartość jest podwójna i stanowi wartość liczby całkowitej odpowiadającej klastrowi, do którego został przypisany każdy punkt danych.

Składnia:

```
KMeansND(num_clusters, num_iter, coordinate_1, coordinate_2 [,coordinate_3 [,
...]])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opis
num_clusters	Liczba całkowita określająca liczbę klastrów.
num_iter	Liczba iteracji grupowania z ponownie inicjalizowanymi centrami klastrów.
coordinate_1	Agregacja obliczająca pierwszą współrzędną, zwykle osi X (wykresu punktowego, która może zostać utworzona z wykresu). Dodatkowe parametry obliczają kolejne współrzędne.

Przykład: wyrażenie wykresu
W tym przykładzie ukazane jest tworzenie wykresu punktowego z użyciem zestawu danych *Iris*, a następnie wykorzystanie KMeans do kolorowania danych według wyrażenia.

Można także utworzyć argument *num_clusters*, a następnie użyć pola wprowadzania zmiennej, aby zmienić liczbę klastrów.

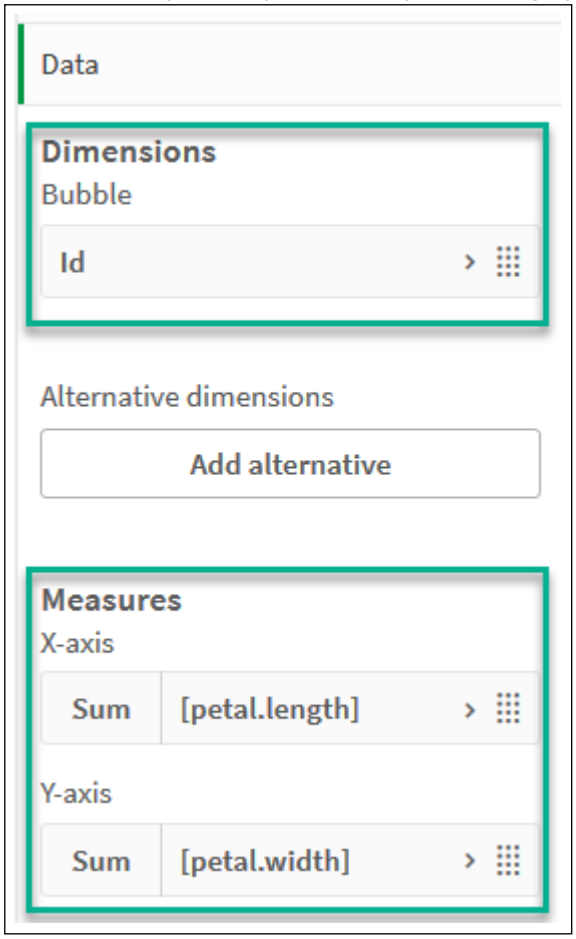
Dodatkowo można utworzyć zmienną dla argumentu *num_iter*, a następnie użyć drugiego pola wprowadzania zmiennej, aby zmienić liczbę iteracji.

Zestaw danych *Iris* jest ogólnodostępny w różnorodnych formatach. Zapewniono dane w formie wbudowanej tabeli do ładowania z użyciem edytora ładowania danych w Qlik Sense. Należy zauważyć, że w tym przykładzie do tabeli danych dodano *identyfikator* kolumny.

Po załadowaniu danych w Qlik Sense należy podjąć następujące kroki:

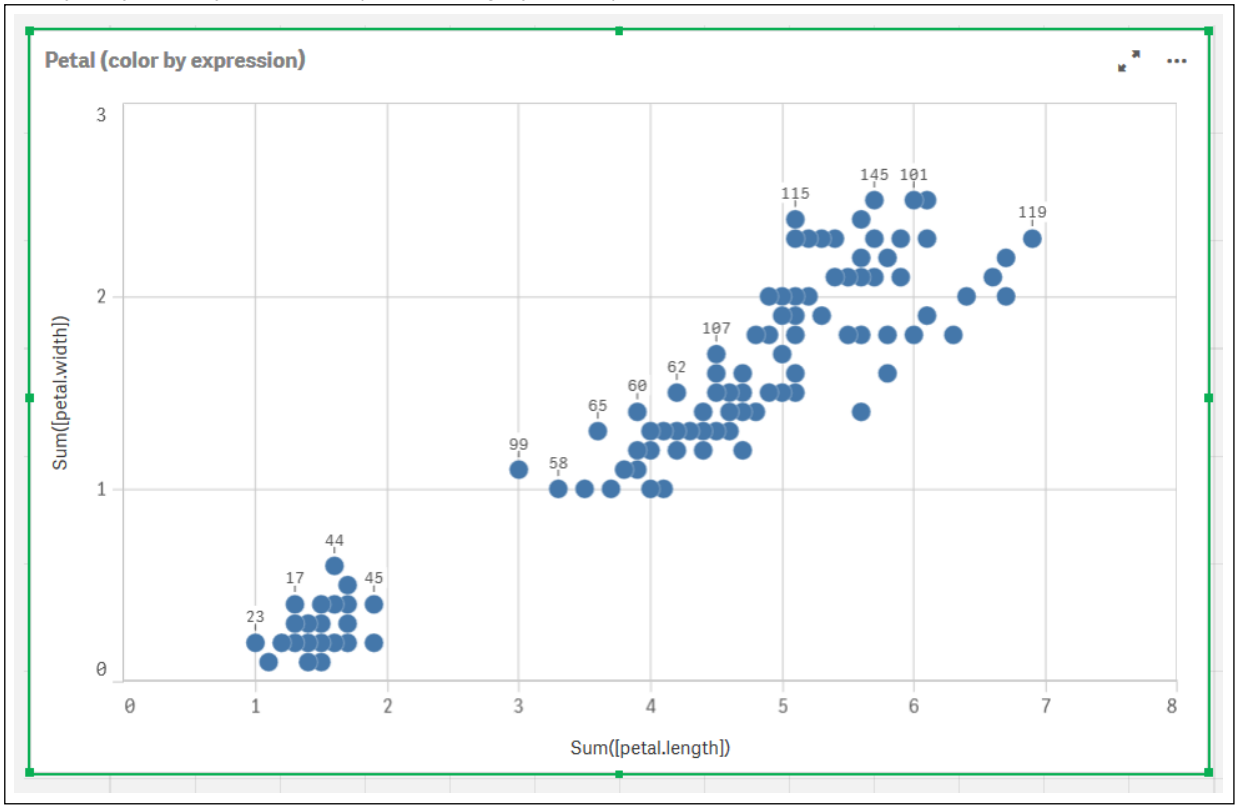
1. Przeciągnij wykres **punktowy** do nowego arkusza. Nazwij wykres *Petal (kolor według wyrażenia)*.
2. Utwórz zmienną, aby określić liczbę klastrów. Dla zmiennej **Nazwa** wprowadź *KmeansPetalClusters*. Dla zmiennej **Definicja** wprowadź *=2*.
3. Utwórz zmienną, aby określić liczbę iteracji. Dla zmiennej **Nazwa** wprowadź *KMeansNumberIterations*. Dla zmiennej **Definicja** wprowadź *=1*.
4. Skonfiguruj opcję **Dane** dla wykresu:
    i. Pod obszarem **Wymiary** wybierz *identyfikator* dla pola **Bąbelek**. Wprowadź identyfikator klastra dla Etykiety.
    ii. Pod obszarem **Miary** wybierz *Sum([petal.length])* dla wyrażenia **Oś X**.
    iii. Pod obszarem **Miary** wybierz *Sum([petal.width])* dla wyrażenia **Oś Y**.

Ustawienia danych dla wykresu Petal (kolor według wyrażenia)



Punkty danych są naniesione na wykres.

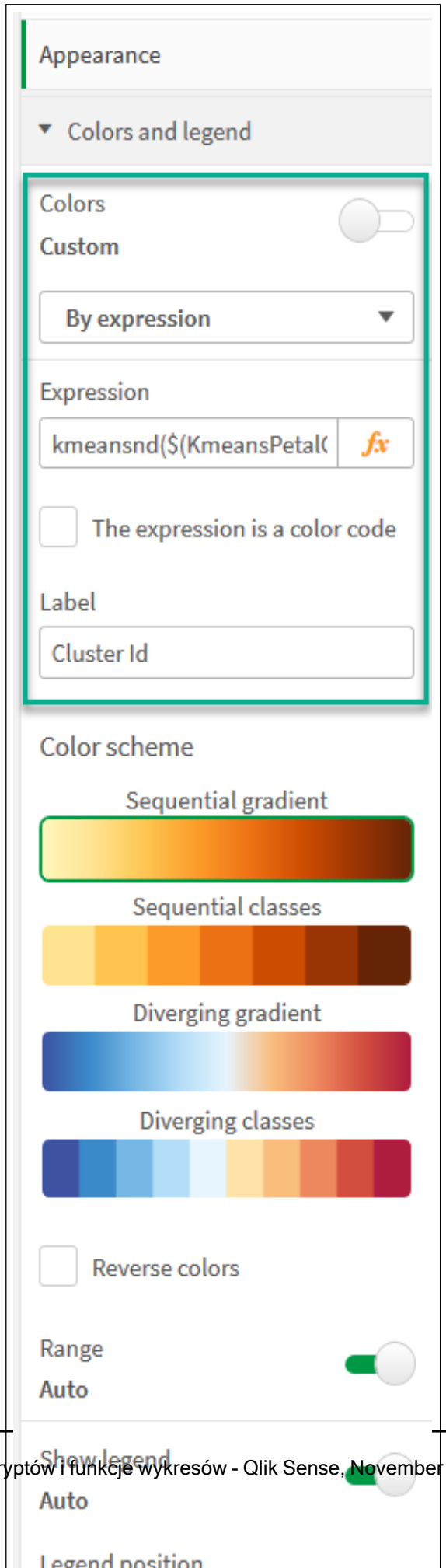
Punkty danych na wykresie Petal (kolor według wyrażenia)



5. Skonfiguruj opcję **Wygląd** dla wykresu:
    i. W opcji **Kolory i legenda** wybierz wartość **Niestandardowe** dla **Kolorów**.
    ii. Wybierz opcję kolorowania wykresu **Według wyrażenia**.
    iii. Wprowadź następujące **Wyrażenie**: *kmeansnd ($(KmeansPetalClusters),$(KmeansNumberIterations), Sum([petal.length]), Sum ([petal.width]),Sum([sepal.length]), Sum([sepal.width]))*
    Należy zauważyć, że wartość *KmeansPetalClusters* jest zmienną ustawianą na *2*. *KmeansNumberIterations* jest zmienną ustawioną na *1*.
    Alternatywnie wprowadź następującą wartość: *kmeansnd(2, 2, Sum([petal.length]), Sum ([petal.width]),Sum([sepal.length]), Sum([sepal.width]))*
    iv. Usuń zaznaczenie pola wyboru **Wyrażenie jest kodem koloru**.

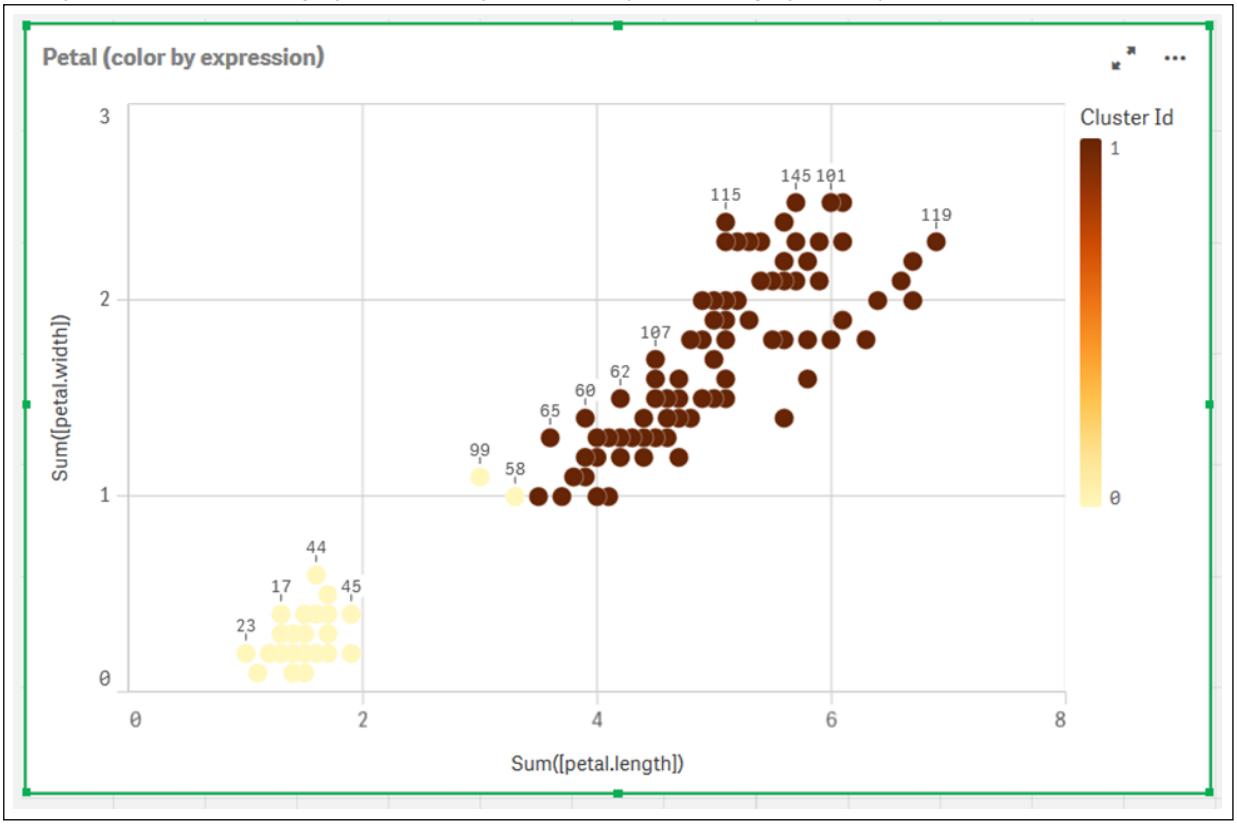
v. Wprowadź następujące dla opcji **Etykieta**: *Identyfikator klastra*

Ustawienia wyglądu dla wykresu Petal (kolor według wyrażenia)


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

Dwa klastry na wykresie są pokolorowane według wyrażenia KMeans.

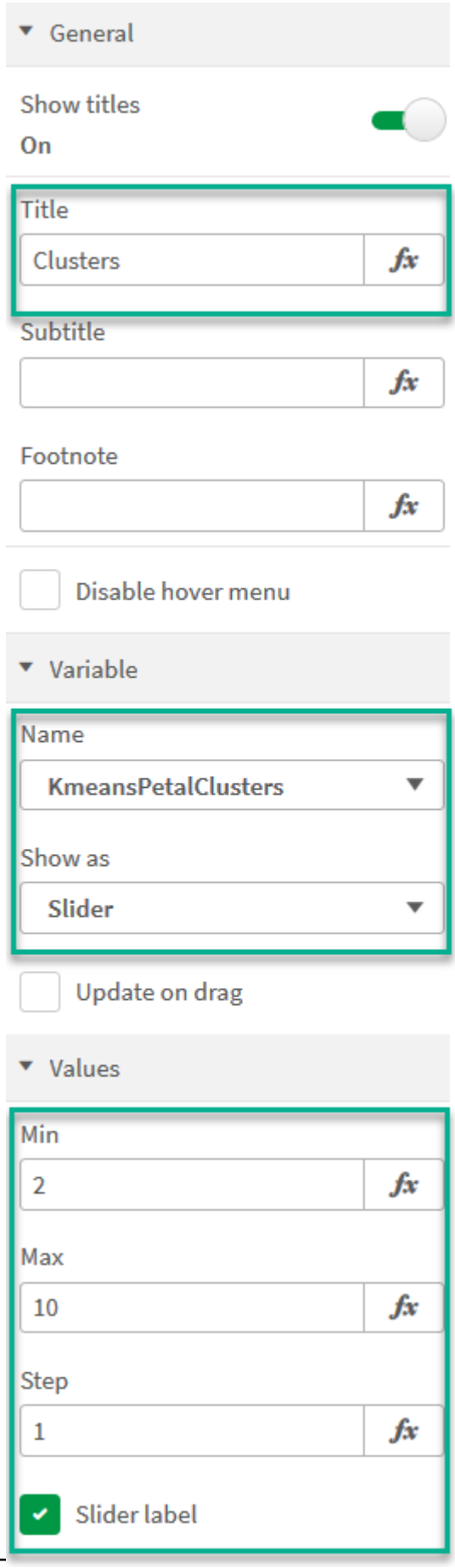
Klastry pokolorowane według wyrażenia na wykresie Petal (kolor według wyrażenia)



6. Dodaj pole **Wprowadzanie zmiennych** dla liczby klastrów.
    i. Pod obszarem **Obiekty niestandardowe** w panelu **Zasoby** wybierz **pakiet Qlik Dashboard bundle**. W przypadku braku dostępu do pakietu dashboard bundle nadal można zmieniać liczbę klastrów, korzystając z utworzonej zmiennej lub bezpośrednio jako liczbę całkowitą w wyrażeniu.
    ii. Przeciągnij pole **Wprowadzanie zmiennych** do arkusza.
    iii. W sekcji **Wygląd** kliknij pozycję **Ogólny**.
    iv. Dla obszaru **Tytuł** wprowadź następujące dane: *Klastry*
    v. Kliknij pozycję **Zmienna**.
    vi. Wybierz następującą zmienną dla obszaru **Nazwa**: *KmeansPetalClusters*.
    vii. Wybierz **Suwak** dla opcji **Pokaż jako**.

viii. Wybierz **Wartości** i skonfiguruj ustawienia zgodnie z wymaganiami

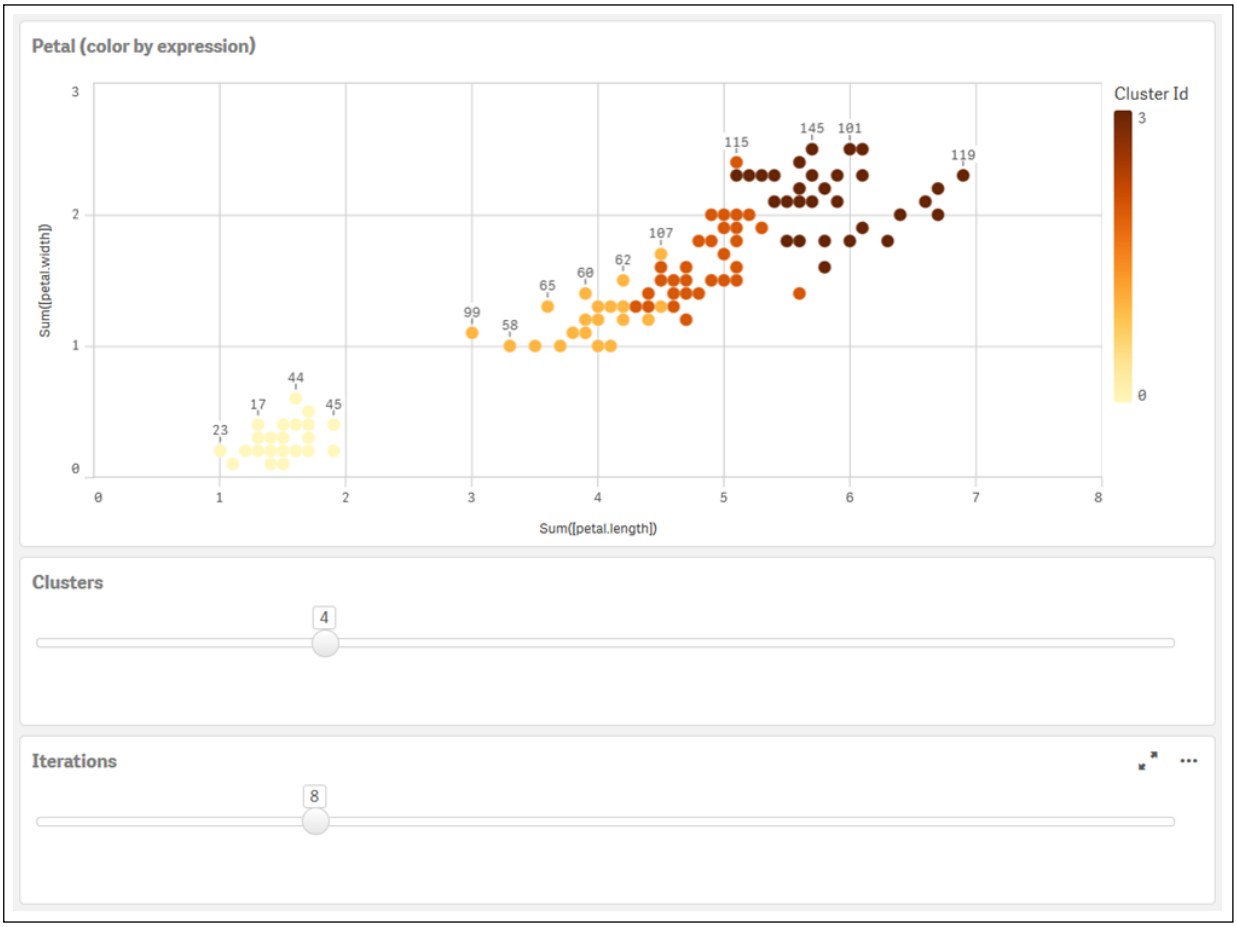
Wygląd dla pola wprowadzania zmiennych Klastry


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. Dodaj pole **Wprowadzanie zmiennych** dla liczby iteracji.
    i. Przeciągnij pole **Wprowadzanie zmiennych** do arkusza.
    ii. Pod obszarem **Wygląd** wybierz **Ogólne**.
    iii. Dla obszaru **Tytuł** wprowadź następujące dane: *Iteracje*
    iv. Pod obszarem **Wygląd** wybierz **Zmienna**.
    v. Wybierz następującą zmienną pod obszarem **Nazwa**: *KmeansNumberIterations*.
    vi. Skonfiguruj dodatkowe ustawienia zgodnie z wymaganiami

    Teraz można zmienić liczbę klastrów i iteracji z użyciem suwaków w polach wprowadzania zmiennych.

    Klastry pokolorowane według wyrażenia na wykresie Petal (kolor według wyrażenia)

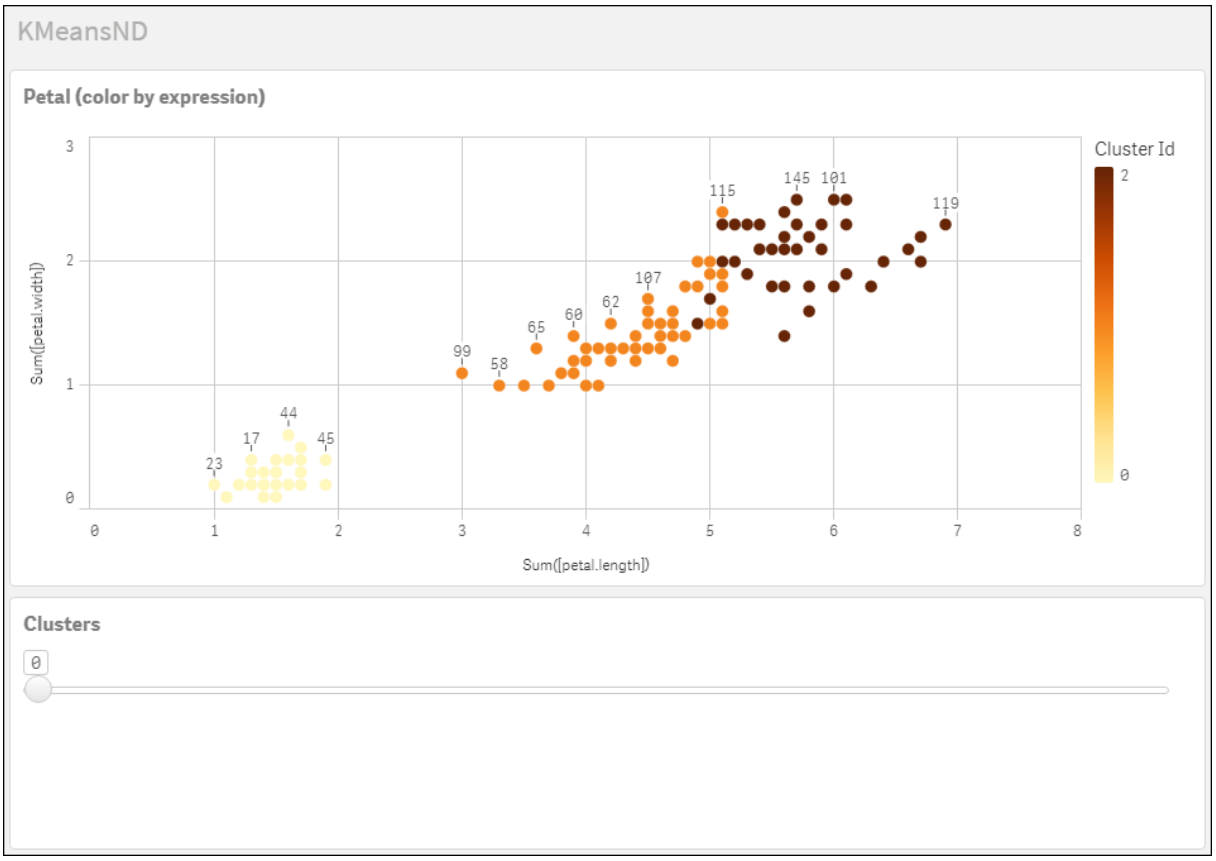


Automatyczne grupowanie

Funkcje **KMeans** obsługują automatyczne grupowanie za pomocą metody zwanej różnicą głębokości (DeD). Gdy użytkownik ustawia liczbę klastrów na 0, określana jest optymalna liczba klastrów dla tego zestawu danych. Należy zauważyć, że liczba całkowita dla liczby klastrów (*k*) nie jest wyraźnie zwracana, ale obliczana w ramach algorytmu KMeans. Na przykład jeśli 0 jest określone w funkcji dla wartości *KmeansPetalClusters* lub ustalone poprzez pole wprowadzania zmiennych, przypisania klastrów są

automatyczne obliczane dla zestawu danych opartego na optymalnej liczbie klastrów. Biorąc pod uwagę zestaw danych Iris, jeśli jako liczbę klastrów wybrano 0, algorytm określi (automatycznie pogrupuje) optymalną liczbę klastrów (3) dla tego zestawu danych.

Metoda różnicy głębokości KMeans określa optymalną liczbę klastrów, gdy (k) ma wartość 0.



Zestaw danych Iris: Wbudowane ładowanie dla edytora ładowania danych w Qlik Sense

```
IrisData: Load * Inline [ sepal.length, sepal.width, petal.length, petal.width, variety, id
5.1, 3.5, 1.4, 0.2, Setosa, 1 4.9, 3, 1.4, 0.2, Setosa, 2 4.7, 3.2, 1.3, 0.2, Setosa, 3 4.6,
3.1, 1.5, 0.2, Setosa, 4 5, 3.6, 1.4, 0.2, Setosa, 5 5.4, 3.9, 1.7, 0.4, Setosa, 6 4.6, 3.4,
1.4, 0.3, Setosa, 7 5, 3.4, 1.5, 0.2, Setosa, 8 4.4, 2.9, 1.4, 0.2, Setosa, 9 4.9, 3.1, 1.5,
0.1, Setosa, 10 5.4, 3.7, 1.5, 0.2, Setosa, 11 4.8, 3.4, 1.6, 0.2, Setosa, 12 4.8, 3, 1.4,
0.1, Setosa, 13 4.3, 3, 1.1, 0.1, Setosa, 14 5.8, 4, 1.2, 0.2, Setosa, 15 5.7, 4.4, 1.5, 0.4,
Setosa, 16 5.4, 3.9, 1.3, 0.4, Setosa, 17 5.1, 3.5, 1.4, 0.3, Setosa, 18 5.7, 3.8, 1.7, 0.3,
Setosa, 19 5.1, 3.8, 1.5, 0.3, Setosa, 20 5.4, 3.4, 1.7, 0.2, Setosa, 21 5.1, 3.7, 1.5, 0.4,
Setosa, 22 4.6, 3.6, 1, 0.2, Setosa, 23 5.1, 3.3, 1.7, 0.5, Setosa, 24 4.8, 3.4, 1.9, 0.2,
Setosa, 25 5, 3, 1.6, 0.2, Setosa, 26 5, 3.4, 1.6, 0.4, Setosa, 27 5.2, 3.5, 1.5, 0.2, Setosa,
28 5.2, 3.4, 1.4, 0.2, Setosa, 29 4.7, 3.2, 1.6, 0.2, Setosa, 30 4.8, 3.1, 1.6, 0.2, Setosa,
31 5.4, 3.4, 1.5, 0.4, Setosa, 32 5.2, 4.1, 1.5, 0.1, Setosa, 33 5.5, 4.2, 1.4, 0.2, Setosa,
34 4.9, 3.1, 1.5, 0.1, Setosa, 35 5, 3.2, 1.2, 0.2, Setosa, 36 5.5, 3.5, 1.3, 0.2, Setosa, 37
4.9, 3.1, 1.5, 0.1, Setosa, 38 4.4, 3, 1.3, 0.2, Setosa, 39 5.1, 3.4, 1.5, 0.2, Setosa, 40 5,
3.5, 1.3, 0.3, Setosa, 41 4.5, 2.3, 1.3, 0.3, Setosa, 42 4.4, 3.2, 1.3, 0.2, Setosa, 43 5,
3.5, 1.6, 0.6, Setosa, 44 5.1, 3.8, 1.9, 0.4, Setosa, 45 4.8, 3, 1.4, 0.3, Setosa, 46 5.1,
3.8, 1.6, 0.2, Setosa, 47 4.6, 3.2, 1.4, 0.2, Setosa, 48 5.3, 3.7, 1.5, 0.2, Setosa, 49 5,
3.3, 1.4, 0.2, Setosa, 50 7, 3.2, 4.7, 1.4, Versicolor, 51 6.4, 3.2, 4.5, 1.5, Versicolor, 52
6.9, 3.1, 4.9, 1.5, Versicolor, 53 5.5, 2.3, 4, 1.3, Versicolor, 54 6.5, 2.8, 4.6, 1.5,
```

```
Versicolor, 55 5.7, 2.8, 4.5, 1.3, Versicolor, 56 6.3, 3.3, 4.7, 1.6, Versicolor, 57 4.9, 2.4,
3.3, 1, Versicolor, 58 6.6, 2.9, 4.6, 1.3, Versicolor, 59 5.2, 2.7, 3.9, 1.4, Versicolor, 60
5, 2, 3.5, 1, Versicolor, 61 5.9, 3, 4.2, 1.5, Versicolor, 62 6, 2.2, 4, 1, Versicolor, 63
6.1, 2.9, 4.7, 1.4, Versicolor, 64 5.6, 2.9, 3.6, 1.3, Versicolor, 65 6.7, 3.1, 4.4, 1.4,
Versicolor, 66 5.6, 3, 4.5, 1.5, Versicolor, 67 5.8, 2.7, 4.1, 1, Versicolor, 68 6.2, 2.2,
4.5, 1.5, Versicolor, 69 5.6, 2.5, 3.9, 1.1, Versicolor, 70 5.9, 3.2, 4.8, 1.8, Versicolor, 71
6.1, 2.8, 4, 1.3, Versicolor, 72 6.3, 2.5, 4.9, 1.5, Versicolor, 73 6.1, 2.8, 4.7, 1.2,
Versicolor, 74 6.4, 2.9, 4.3, 1.3, Versicolor, 75 6.6, 3, 4.4, 1.4, Versicolor, 76 6.8, 2.8,
4.8, 1.4, Versicolor, 77 6.7, 3, 5, 1.7, Versicolor, 78 6, 2.9, 4.5, 1.5, Versicolor, 79 5.7,
2.6, 3.5, 1, Versicolor, 80 5.5, 2.4, 3.8, 1.1, Versicolor, 81 5.5, 2.4, 3.7, 1, Versicolor,
82 5.8, 2.7, 3.9, 1.2, Versicolor, 83 6, 2.7, 5.1, 1.6, Versicolor, 84 5.4, 3, 4.5, 1.5,
Versicolor, 85 6, 3.4, 4.5, 1.6, Versicolor, 86 6.7, 3.1, 4.7, 1.5, Versicolor, 87 6.3, 2.3,
4.4, 1.3, Versicolor, 88 5.6, 3, 4.1, 1.3, Versicolor, 89 5.5, 2.5, 4, 1.3, Versicolor, 90
5.5, 2.6, 4.4, 1.2, Versicolor, 91 6.1, 3, 4.6, 1.4, Versicolor, 92 5.8, 2.6, 4, 1.2,
Versicolor, 93 5, 2.3, 3.3, 1, Versicolor, 94 5.6, 2.7, 4.2, 1.3, Versicolor, 95 5.7, 3, 4.2,
1.2, Versicolor, 96 5.7, 2.9, 4.2, 1.3, Versicolor, 97 6.2, 2.9, 4.3, 1.3, Versicolor, 98 5.1,
2.5, 3, 1.1, Versicolor, 99 5.7, 2.8, 4.1, 1.3, Versicolor, 100 6.3, 3.3, 6, 2.5, Virginica,
101 5.8, 2.7, 5.1, 1.9, Virginica, 102 7.1, 3, 5.9, 2.1, Virginica, 103 6.3, 2.9, 5.6, 1.8,
Virginica, 104 6.5, 3, 5.8, 2.2, Virginica, 105 7.6, 3, 6.6, 2.1, Virginica, 106 4.9, 2.5,
4.5, 1.7, Virginica, 107 7.3, 2.9, 6.3, 1.8, Virginica, 108 6.7, 2.5, 5.8, 1.8, Virginica, 109
7.2, 3.6, 6.1, 2.5, Virginica, 110 6.5, 3.2, 5.1, 2, Virginica, 111 6.4, 2.7, 5.3, 1.9,
Virginica, 112 6.8, 3, 5.5, 2.1, Virginica, 113 5.7, 2.5, 5, 2, Virginica, 114 5.8, 2.8, 5.1,
2.4, Virginica, 115 6.4, 3.2, 5.3, 2.3, Virginica, 116 6.5, 3, 5.5, 1.8, Virginica, 117 7.7,
3.8, 6.7, 2.2, Virginica, 118 7.7, 2.6, 6.9, 2.3, Virginica, 119 6, 2.2, 5, 1.5, Virginica,
120 6.9, 3.2, 5.7, 2.3, Virginica, 121 5.6, 2.8, 4.9, 2, Virginica, 122 7.7, 2.8, 6.7, 2,
Virginica, 123 6.3, 2.7, 4.9, 1.8, Virginica, 124 6.7, 3.3, 5.7, 2.1, Virginica, 125 7.2, 3.2,
6, 1.8, Virginica, 126 6.2, 2.8, 4.8, 1.8, Virginica, 127 6.1, 3, 4.9, 1.8, Virginica, 128
6.4, 2.8, 5.6, 2.1, Virginica, 129 7.2, 3, 5.8, 1.6, Virginica, 130 7.4, 2.8, 6.1, 1.9,
Virginica, 131 7.9, 3.8, 6.4, 2, Virginica, 132 6.4, 2.8, 5.6, 2.2, Virginica, 133 6.3, 2.8,
5.1, 1.5, Virginica, 134 6.1, 2.6, 5.6, 1.4, Virginica, 135 7.7, 3, 6.1, 2.3, Virginica, 136
6.3, 3.4, 5.6, 2.4, Virginica, 137 6.4, 3.1, 5.5, 1.8, Virginica, 138 6, 3, 4.8, 1.8,
Virginica, 139 6.9, 3.1, 5.4, 2.1, Virginica, 140 6.7, 3.1, 5.6, 2.4, Virginica, 141 6.9, 3.1,
5.1, 2.3, Virginica, 142 5.8, 2.7, 5.1, 1.9, Virginica, 143 6.8, 3.2, 5.9, 2.3, Virginica, 144
6.7, 3.3, 5.7, 2.5, Virginica, 145 6.7, 3, 5.2, 2.3, Virginica, 146 6.3, 2.5, 5, 1.9,
Virginica, 147 6.5, 3, 5.2, 2, Virginica, 148 6.2, 3.4, 5.4, 2.3, Virginica, 149 5.9, 3, 5.1,
1.8, Virginica, 150 ];
```

KMeansCentroid2D – funkcja wykresu

KMeansCentroid2D() poddaje ocenie wiersze wykresu, stosując algorytm centroidów oraz wyświetlając dla każdego wiersza wykresu oczekiwaną współrzędną klastra, do którego został przypisany ten punkt danych. Kolumny wykorzystywane przez algorytm grupowania są określone przez odpowiednio parametry coordinate_1 i coordinate_2. Oba te parametry są agregacjami. Liczba tworzonych klastrów jest określana przez parametr num_clusters. Dane można opcjonalnie znormalizować za pomocą parametru norm.

KMeansCentroid2D zwraca jedną wartość na punkt danych. Zwrócona wartość jest podwójna i stanowi jedną ze współrzędnych pozycji odpowiadającej centrum klastra, do którego został przypisany punkt danych.

Składnia:

```
KMeansCentroid2D(num_clusters, coordinate_no, coordinate_1, coordinate_2 [,
norm])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opisu
num_clusters	Liczba całkowita określająca liczbę klastrów.
coordinate_no	Oczekiwana liczba współrzędnych centroidów (np. odpowiednio do osi x, y lub z).
coordinate_1	Agregacja obliczająca pierwszą współrzędną, zwykle osi X wykresu punktowego, która może zostać utworzona z wykresu. Dodatkowy paramert, coordinate_2, oblicza drugą współrzędną.
norm	Opcjonalna metoda normalizacji stosowana do zestawów danych przed algorytmem centroidów. Możliwe wartości: 0 lub „none” w przypadku braku normalizacji 1 lub „zscore” w przypadku normalizacji z-score 2 lub „minmax” dla normalizacji min.-maks. Jeśli nie podano żadnego parametru lub jeśli podany parametr jest nieprawidłowy, żadna normalizacja nie jest stosowana. Normalizacja z-score normalizuje dane w oparciu o średnią cechy i odchylenie standardowe. Normalizacja z-score nie gwarantuje, że każda cecha będzie mieć taką samą skalę, ale jest lepszym podejściem niż min.-maks. w przypadku wartości odstających. Normalizacja min.-maks. zapewnia, że cechy mają tę samą skalę, biorąc minimalne i maksymalne wartości każdej z nich i przeliczając każdy punkt danych.

Automatyczne grupowanie

Funkcje **KMeans** obsługują automatyczne grupowanie za pomocą metody zwanej różnicą głębokości (DeD). Gdy użytkownik ustawia liczbę klastrów na 0, określana jest optymalna liczba klastrów dla tego zestawu danych. Należy zauważyć, że liczba całkowita dla liczby klastrów (*k*) nie jest wyraźnie zwracana, ale obliczana w ramach algorytmu KMeans. Na przykład jeśli 0 jest określone w funkcji dla wartości *KmeansPetalClusters* lub ustalone poprzez pole wprowadzania zmiennych, przypisania klastrów są automatyczne obliczane dla zestawu danych opartego na optymalnej liczbie klastrów.

KMeansCentroidND – funkcja wykresu

KMeansCentroidND() poddaje ocenie wiersze wykresu, stosując algorytm centroidów oraz wyświetlając dla każdego wiersza wykresu oczekiwaną współrzędną klastra, do którego został przypisany ten punkt danych. Kolumny wykorzystywane przez algorytm grupowania są określone przez odpowiednio parametry

coordinate_1, coordinate_2 itd., aż do n kolumn. Wszystkie te parametry są agregacjami. Liczba tworzonych klastrów jest określana przez parametr num_clusters.

KMeansCentroidND zwraca jedną wartość na wiersz. Zwrócona wartość jest podwójna i stanowi jedną ze współrzędnych pozycji odpowiadającej centrum klastra, do którego został przypisany punkt danych.

Składnia:

```
KMeansCentroidND((num_clusters, num_iter, coordinate_no, coordinate_1,
coordinate_2 [,coordinate_3 [, ...]])
```

Typ zwracanych danych: dual

Argumenty:

Argumenty

Argument	Opisu
num_clusters	Liczba całkowita określająca liczbę klastrów.
num_iter	Liczba iteracji grupowania z ponownie inicjalizowanymi centrami klastrów.
coordinate_no	Oczekiwana liczba współrzędnych centroidów (np. odpowiednio do osi x, y lub z).
coordinate_1	Agregacja obliczająca pierwszą współrzędną, zwykle osi X (wykresu punktowego, która może zostać utworzona z wykresu). Dodatkowe parametry obliczają kolejne współrzędne.

Automatyczne grupowanie

Funkcje **KMeans** obsługują automatyczne grupowanie za pomocą metody zwanej różnicą głębokości (DeD). Gdy użytkownik ustawia liczbę klastrów na 0, określana jest optymalna liczba klastrów dla tego zestawu danych. Należy zauważyć, że liczba całkowita dla liczby klastrów (*k*) nie jest wyraźnie zwracana, ale obliczana w ramach algorytmu KMeans. Na przykład jeśli 0 jest określone w funkcji dla wartości *KmeansPetalClusters* lub ustalone poprzez pole wprowadzania zmiennych, przypisania klastrów są automatyczne obliczane dla zestawu danych opartego na optymalnej liczbie klastrów.

5.23 Funkcje rozkładu statystycznego

Funkcje rozkładu statystycznego zwracają prawdopodobieństwa wystąpienia różnych możliwych wyników dla danej zmiennej wejściowej. Możesz używać tych funkcji do obliczania potencjalnych wartości punktów danych.

Trzy opisane poniżej grupy funkcji rozkładu statystycznego są implementowane w aplikacji Qlik Sense przy użyciu biblioteki funkcji Cephes. Referencje i szczegóły dotyczące użytych algorytmów, dokładności itp. można znaleźć na stronie: ≤ Cephes library. Biblioteka funkcji Cephes jest używana na podstawie pozwolenia.

- Funkcje prawdopodobieństwa obliczają prawdopodobieństwo w punkcie rozkładu określonym przez podaną wartość.
    - Funkcje częstotliwości są używane do rozkładów dyskretnych.
    - Funkcje gęstości są używane do funkcji ciągłych.
- Funkcje Dist obliczają skumulowane prawdopodobieństwo rozkładu w punkcie rozkładu określonym przez podaną wartość.
- Funkcje Inv obliczają wartość odwrotną z uwzględnieniem skumulowanego prawdopodobieństwa rozkładu.

Wszystkie funkcje mogą być stosowane zarówno w skryptach ładowania danych, jak i wyrażeniach wykresu.

Przegląd funkcji rozkładu statystycznego

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

BetaDensity
`BetaDensity()` zwraca prawdopodobieństwo rozkładu Beta.

```
BetaDensity (value, alpha, beta)
```

BetaDist
`BetaDist()` zwraca zakumulowane prawdopodobieństwo rozkładu Beta.

```
BetaDist (value, alpha, beta)
```

BetaInv
`BetaINV()` zwraca odwrotność zakumulowanego prawdopodobieństwa rozkładu Beta.

```
BetaInv (prob, alpha, beta)
```

BinomDist
`BinomDist()` zwraca zakumulowane prawdopodobieństwo rozkładu dwumianowego.

```
BinomDist (value, trials, trial_probability)
```

BinomFrequency
`BinomFrequency()` zwraca dwumianową dystrybucję prawdopodobieństwa.

```
BinomFrequency (value, trials, trial_probability)
```

BinomInv
`BinomInv()` zwraca odwrotność zakumulowanego prawdopodobieństwa rozkładu dwumianowego.

```
BinomInv (prob, trials, trial_probability)
```

ChiDensity

Funkcja `ChiDensity()` zwraca jednostronne prawdopodobieństwo rozkładu chi^2. Funkcja gęstości chi^2 jest powiązana z testem chi^2.

```
ChiDensity (value, degrees_freedom)
```

ChiDist

Funkcja `ChiDist()` zwraca jednostronne prawdopodobieństwo rozkładu chi^2. Rozkład chi^2 jest powiązany z testem chi^2.

```
ChiDist (value, degrees_freedom)
```

ChiInv

Funkcja `ChiInv()` zwraca odwrotność jednostronnego prawdopodobieństwa rozkładu chi^2.

```
ChiInv (prob, degrees_freedom)
```

FDensity

`FDensity()` zwraca prawdopodobieństwo rozkładu `F`.

```
FDensity (value, degrees_freedom1, degrees_freedom2)
```

FDist

`FDist()` zwraca zakumulowane prawdopodobieństwo rozkładu `F`.

```
FDist (value, degrees_freedom1, degrees_freedom2)
```

FInv

`FInv()` zwraca odwrotność zakumulowanego prawdopodobieństwa rozkładu `F`.

```
FInv (prob, degrees_freedom1, degrees_freedom2)
```

GammaDensity

`GammaDensity()` zwraca prawdopodobieństwo rozkładu Gamma.

```
GammaDensity (value, k, θ)
```

GammaDist

`GammaDist()` zwraca zakumulowane prawdopodobieństwo rozkładu Gamma.

```
GammaDist (value, k, θ)
```

GammaInv

`GammaInv()` zwraca odwrotność zakumulowanego prawdopodobieństwa rozkładu Gamma.

```
GammaInv (prob, k, θ)
```

NormDist

Funkcja `NormDist()` zwraca skumulowany rozkład normalny dla określonej średniej i odchylenia standardowego. Jeśli wartość mean = 0, a wartość standard_dev = 1, wówczas funkcja zwraca standardowy rozkład normalny.

```
NormDist (value, mean, standard_dev)
```

NormInv

Funkcja `NormInv()` zwraca odwrotność skumulowanego rozkładu normalnego dla określonej średniej i odchylenia standardowego.

```
NormInv (prob, mean, standard_dev)
```

PoissonDist

`PoissonDist()` zwraca zakumulowane prawdopodobieństwo rozkładu Poissona.

```
PoissonDist (value, mean)
```

PoissonFrequency

`PoissonFrequency()` zwraca dystrybucję prawdopodobieństwa Poissona.

```
PoissonFrequency (value, mean)
```

PoissonInv

`PoissonInv()` zwraca odwrotność zakumulowanego prawdopodobieństwa rozkładu Poissona.

```
PoissonInv (prob, mean)
```

TDensity

`TDensity()` zwraca wartość funkcji gęstości `t` studenta, gdzie wartość numeryczna jest wartością obliczoną `t`, dla której ma zostać obliczone prawdopodobieństwo.

```
TDensity (value, degrees_freedom, tails)
```

TDist

Funkcja `TDist()` zwraca prawdopodobieństwo dla rozkładu `t` Studenta, gdzie wartość liczbowa jest wartością wyliczaną `t`, dla której prawdopodobieństwo ma zostać obliczone.

```
TDist (value, degrees_freedom, tails)
```

TInv

Funkcja `TInv()` zwraca wartość `t` rozkładu `t` studenta jako funkcję prawdopodobieństwa i stopni swobody.

```
TInv (prob, degrees_freedom)
```

Zob. także:

p *Statystyczne funkcje agregacji (page 381)*

BetaDensity

`BetaDensity()` zwraca prawdopodobieństwo rozkładu Beta.

Składnia:

```
BetaDensity(value, alpha, beta)
```

Typ zwracanych danych: liczba

Argumenty

Argument	Opis
value	Wartość, przy której wymagana jest ocena rozkładu. Wartość musi się zawierać między 0 a 1.
alpha	Liczba dodatnia definiująca pierwszy parametr kształtu. Jest wykładnikiem potęgi zmiennej losowej
beta	Liczba dodatnia definiująca drugi parametr kształtu. Wskazuje liczbę stopni swobody mianownika.

BetaDist

`BetaDist()` zwraca zakumulowane prawdopodobieństwo rozkładu Beta.

Składnia:

```
BetaDist(value, alpha, beta)
```

Typ zwracanych danych: liczba

Argumenty

Argument	Opis
value	Wartość, przy której wymagana jest ocena rozkładu. Wartość musi się zawierać między 0 a 1.
alpha	Liczba dodatnia definiująca pierwszy parametr kształtu. Jest wykładnikiem potęgi zmiennej losowej
beta	Liczba dodatnia definiująca drugi parametr kształtu. Jest wykładnikiem potęgi kontrolującym kształt rozkładu.

Ta funkcja jest powiązana z funkcją `BetaInv` w następujący sposób:
`If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value`

BetaInv

`BetaINV()` zwraca odwrotność zakumulowanego prawdopodobieństwa rozkładu Beta.

Składnia:

```
BetaInv(prob, alpha, beta)
```

Typ zwracanych danych: liczba

Argumenty

Argument	Opis
prob	Prawdopodobieństwo powiązane z rozkładem prawdopodobieństwa beta. Musi być liczbą z zakresu od 0 do 1.
alpha	Liczba dodatnia definiująca pierwszy parametr kształtu. Jest wykładnikiem potęgi zmiennej losowej
beta	Liczba dodatnia definiująca drugi parametr kształtu. Jest wykładnikiem potęgi kontrolującym kształt rozkładu.

Ta funkcja jest powiązana z funkcją `BetaDist` w następujący sposób:
`If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value`

BinomDist

`BinomDist()` zwraca zakumulowane prawdopodobieństwo rozkładu dwumianowego.

Składnia:

```
BinomDist(value, trials, trial_probability)
```

Typ zwracanych danych: liczba

Argumenty

Argument	Opis
value	Wartość, przy której wymagana jest ocena rozkładu. Wartość musi być liczbą całkowitą nie mniejszą niż zero i nie większą niż liczba prób.
trials	Dodatnia liczba całkowita wskazująca liczbę prób.
trial_ probability	Prawdopodobieństwo powodzenia każdej próby. Jest zawsze liczbą z zakresu od 0 do 1.

Ta funkcja jest powiązana z funkcją `BinomInv` w następujący sposób:
`If prob = BinomDIST(value, trials, trial_probability), then BinomInv(prob, trials, trial_ probability) = value`

BinomFrequency

`BinomFrequency()` zwraca dwumianową dystrybucję prawdopodobieństwa.

Składnia:

```
BinomFrequency(value, trials, trial_probability)
```

Typ zwracanych danych: liczba

Argumenty

Argument	Opis
value	Wartość, przy której wymagana jest ocena rozkładu. Wartość musi być liczbą całkowitą nie mniejszą niż zero i nie większą niż liczba prób.
trials	Dodatnia liczba całkowita wskazująca liczbę prób
trial_probability	Prawdopodobieństwo powodzenia każdej próby. Jest zawsze liczbą z zakresu od 0 do 1.

BinomInv

`BinomInv()` zwraca odwrotność zakumulowanego prawdopodobieństwa rozkładu dwumianowego.

Składnia:

```
BinomInv(prob, trials, trial_probability)
```

Typ zwracanych danych: liczba

Argumenty

Argument	Opis
prob	Prawdopodobieństwo powiązane z dwumianowym rozkładem prawdopodobieństwa. Musi być liczbą z zakresu od 0 do 1.
trials	Dodatnia liczba całkowita wskazująca liczbę prób.
trial_probability	Prawdopodobieństwo powodzenia każdej próby. Jest zawsze liczbą z zakresu od 0 do 1.

Ta funkcja jest powiązana z funkcją `BinomDist` w następujący sposób:
`If prob = BinomDist(value, trials, trial_probability), then BinomInv(prob, trials, trial_probability) = value`

ChiDensity

Funkcja `ChiDensity()` zwraca jednostronne prawdopodobieństwo rozkładu chi^2. Funkcja gęstości chi^2 jest powiązana z testem chi^2.

Składnia:

```
ChiDensity(value, degrees_freedom)
```

Typ zwracanych danych: liczba

Argumenty

Argument	Opis
`value`	Wartość, przy której wymagana jest ocena rozkładu. Wartość nie może być ujemna.
`degrees_ freedom`	Dodatnia liczba całkowita określająca liczbę liczników ułamka określającego stopnie swobody.

ChiDist

Funkcja `ChiDist()` zwraca jednostronne prawdopodobieństwo rozkładu chi^2. Rozkład chi^2 jest powiązany z testem chi^2.

Składnia:

```
CHIDIST(value, degrees_freedom)
```

Typ zwracanych danych: liczba

Argumenty:

Argumenty

Argument	Opis
value	Wartość, przy której wymagana jest ocena rozkładu. Wartość nie może być ujemna.
degrees_ freedom	Dodatnia liczba całkowita określająca liczbę stopni swobody.

Ta funkcja jest powiązana z funkcją **ChiInv** w następujący sposób:
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

Ograniczenia:

Wszystkie argumenty muszą być liczbowe – w przeciwnym wypadku zwracana jest wartość NULL.

Przykłady i wyniki:

Przykład	Wynik
CHIDIST(8, 15)	Zwraca wartość 0,9238

ChiInv

Funkcja `ChiInv()` zwraca odwrotność jednostronnego prawdopodobieństwa rozkładu chi^2.

Składnia:

```
CHIINV(prob, degrees_freedom)
```

Typ zwracanych danych: liczba

Argumenty:

Argumenty

Argument	Opis
prob	Prawdopodobieństwo powiązane z rozkładem chi^2. Musi być liczbą z zakresu od 0 do 1.
degrees_ freedom	Liczba całkowita określająca liczbę stopni swobody.

Ta funkcja jest powiązana z funkcją **ChiDist** w następujący sposób:
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

Ograniczenia:

Wszystkie argumenty muszą być liczbowe – w przeciwnym wypadku zwracana jest wartość NULL.

Przykłady i wyniki:

Przykład	Wynik
CHIINV(0.9237827, 15)	Zwraca wartość 8,0000

FDensity

`FDensity()` zwraca prawdopodobieństwo rozkładu `F`.

Składnia:

```
FDensity(value, degrees_freedom1, degrees_freedom2)
```

Typ zwracanych danych: liczba

Argumenty

Argument	Opis
`value`	Wartość, przy której wymagana jest ocena rozkładu. Wartość nie może być ujemna.
`degrees_ freedom1`	Dodatnia liczba całkowita określająca liczbę liczników ułamka określającego stopnie swobody.
`degrees_ freedom2`	Dodatnia liczba całkowita określająca liczbę mianowników ułamka określającego stopnie swobody.

FDist

`FDist()` zwraca zakumulowane prawdopodobieństwo rozkładu `F`.

Składnia:

```
FDist(value, degrees_freedom1, degrees_freedom2)
```

Typ zwracanych danych: liczba

Argumenty:

Argumenty

Argument	Opis
value	Wartość, przy której wymagana jest ocena rozkładu. Wartość nie może być ujemna.
degrees_ freedom1	Dodatnia liczba całkowita określająca liczbę liczników ułamka określającego stopnie swobody.
degrees_ freedom2	Dodatnia liczba całkowita określająca liczbę mianowników ułamka określającego stopnie swobody.

Ta funkcja jest powiązana z funkcją **FInv** w następujący sposób:
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

Ograniczenia:

Wszystkie argumenty muszą być liczbowe – w przeciwnym wypadku zwracana jest wartość NULL.

Przykłady i wyniki:

Przykład	Wynik
FDIST(15, 8, 6)	Zwraca wartość 0,0019

FInv

`FInv()` zwraca odwrotność zakumulowanego prawdopodobieństwa rozkładu `F`.

Składnia:

```
FInv(prob, degrees_freedom1, degrees_freedom2)
```

Typ zwracanych danych: liczba

Argumenty:

Argumenty

Argument	Opis
prob	Prawdopodobieństwo powiązane z rozkładem prawdopodobieństwa F, musi być liczbą od 0 do 1.
degrees_ freedom	Liczba całkowita określająca liczbę stopni swobody.

Ta funkcja jest powiązana z funkcją **FDist** w następujący sposób:
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

Ograniczenia:

Wszystkie argumenty muszą być liczbowe – w przeciwnym wypadku zwracana jest wartość NULL.

Przykłady i wyniki:

Przykład	Wynik
FINV(0.0019369, 8, 6)	Zwraca wartość 15,0000

GammaDensity

`GammaDensity()` zwraca prawdopodobieństwo rozkładu Gamma.

Składnia:

```
GammaDensity(value, k, θ)
```

Typ zwracanych danych: liczba

Argumenty

Argument	Opis
value	Wartość, przy której wymagana jest ocena rozkładu. Wartość nie może być ujemna.
k	Liczba dodatnia definiująca parametr kształtu.
θ	Liczba dodatnia definiująca parametr skali.

GammaDist

`GammaDist()` zwraca zakumulowane prawdopodobieństwo rozkładu Gamma.

Składnia:

```
GammaDist(value, k, θ)
```

Typ zwracanych danych: liczba

Argumenty

Argument	Opis
value	Wartość, przy której wymagana jest ocena rozkładu. Wartość nie może być ujemna.
k	Liczba dodatnia definiująca parametr kształtu.
θ	Liczba dodatnia definiująca parametr skali.

Ta funkcja jest powiązana z funkcją `GammaINV` w następujący sposób:
`If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value`

GammaInv

`GammaInv()` zwraca odwrotność zakumulowanego prawdopodobieństwa rozkładu Gamma.

Składnia:

```
GammaInv(prob, k, θ)
```

Typ zwracanych danych: liczba

Argumenty

Argument	Opis
prob	Prawdopodobieństwo powiązane z rozkładem prawdopodobieństwa gamma. Musi być liczbą z zakresu od 0 do 1.
k	Liczba dodatnia definiująca parametr kształtu.
θ	Liczba dodatnia definiująca parametr skali.

Ta funkcja jest powiązana z funkcją `GammaDist` w następujący sposób:
`If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value`

NormDist

Funkcja `NormDist()` zwraca skumulowany rozkład normalny dla określonej średniej i odchylenia standardowego. Jeśli wartość mean = 0, a wartość standard_dev = 1, wówczas funkcja zwraca standardowy rozkład normalny.

Składnia:

```
NORMDIST(value, [mean], [standard_dev], [cumulative])
```

Typ zwracanych danych: liczba

Argumenty:

Argumenty

Argument	Opis
value	Wartość, przy której wymagana jest ocena rozkładu.
mean	Wartość opcjonalna określająca średnią arytmetyczną dla rozkładu. Jeśli ten argument nie zostanie podany, jego wartością domyślną będzie 0.
standard_dev	Opcjonalna wartość dodatnia określająca odchylenie standardowe rozkładu. Jeśli ten argument nie zostanie podany, jego wartością domyślną będzie 1.

Argument	Opis
cumulative	Opcjonalnie można wybrać użycie standardowego rozkładu normalnego lub rozkładu skumulowanego. 0 = standardowy rozkład normalny 1 = rozkład skumulowany (domyślny)

Ta funkcja jest powiązana z funkcją **NormInv** w następujący sposób:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Ograniczenia:

Wszystkie argumenty muszą być liczbowe – w przeciwnym wypadku zwracana jest wartość NULL.

Przykłady i wyniki:

Przykład	Wynik
NORMDIST(0.5, 0, 1)	Zwraca wartość 0,6915

NormInv

Funkcja `NormInv()` zwraca odwrotność skumulowanego rozkładu normalnego dla określonej średniej i odchylenia standardowego.

Składnia:

```
NORMINV(prob, mean, standard_dev)
```

Typ zwracanych danych: liczba

Argumenty:

Argumenty

Argument	Opis
prob	Prawdopodobieństwo powiązane z rozkładem normalnym. Musi być liczbą z zakresu od 0 do 1.
mean	Wartość określająca średnią arytmetyczną dla rozkładu.
standard_ dev	Wartość dodatnia określająca odchylenie standardowe rozkładu.

Ta funkcja jest powiązana z funkcją **NormDist** w następujący sposób:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Ograniczenia:

Wszystkie argumenty muszą być liczbowe – w przeciwnym wypadku zwracana jest wartość NULL.

Przykłady i wyniki:

Przykład	Wynik
NORMINV(0.6914625, 0, 1)	Zwraca wartość 0,5000

PoissonDist

`PoissonDist()` zwraca zakumulowane prawdopodobieństwo rozkładu Poissona.

Składnia:

```
PoissonDist(value, mean)
```

Typ zwracanych danych: liczba

Argumenty

Argument	Opis
`value`	Wartość, przy której wymagana jest ocena rozkładu. Wartość nie może być ujemna.
`mean`	Liczba dodatnia definiująca średni wynik.

Ta funkcja jest powiązana z funkcją `PoissonInv` w następujący sposób:
`If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value`

PoissonFrequency

`PoissonFrequency()` zwraca dystrybucję prawdopodobieństwa Poissona.

Składnia:

```
PoissonFrequency(value, mean)
```

Typ zwracanych danych: liczba

Argumenty

Argument	Opis
`value`	Wartość, przy której wymagana jest ocena rozkładu. Wartość nie może być ujemna.
`mean`	Liczba dodatnia definiująca średni wynik.

PoissonInv

`PoissonInv()` zwraca odwrotność zakumulowanego prawdopodobieństwa rozkładu Poissona.

Składnia:

```
PoissonInv(prob, mean)
```

Typ zwracanych danych: liczba

Argumenty

Argument	Opis
prob	Prawdopodobieństwo powiązane z rozkładem prawdopodobieństwa Poissona. Musi być liczbą z zakresu od 0 do 1.
mean	Liczba dodatnia definiująca średni wynik.

Ta funkcja jest powiązana z funkcją PoissonDIST w następujący sposób:
`If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value`

TDensity

TDensity() zwraca wartość funkcji gęstości t studenta, gdzie wartość numeryczna jest wartością obliczoną t, dla której ma zostać obliczone prawdopodobieństwo.

Składnia:

```
TDensity(value, degrees_freedom)
```

Typ zwracanych danych: liczba

Argumenty

Argument	Opis
value	Wartość, przy której wymagana jest ocena rozkładu. Wartość nie może być ujemna.
degrees_freedom	Dodatnia liczba całkowita określająca liczbę stopni swobody.

TDist

Funkcja TDist() zwraca prawdopodobieństwo dla rozkładu t Studenta, gdzie wartość liczbowa jest wartością wyliczaną t, dla której prawdopodobieństwo ma zostać obliczone.

Składnia:

```
TDist(value, degrees_freedom, tails)
```

Typ zwracanych danych: liczba

Argumenty:

Argumenty

Argument	Opis
value	Wartość, przy której wymagana jest ocena rozkładu. Wartość nie może być ujemna.

Argument	Opis
degrees_ freedom	Dodatnia liczba całkowita określająca liczbę stopni swobody.
tails	Musi być równa 1 (rozkład jednostronny) lub 2 (rozkład dwustronny).

Ta funkcja jest powiązana z funkcją **TInv** w następujący sposób:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value`

Ograniczenia:

Wszystkie argumenty muszą być liczbowe – w przeciwnym wypadku zwracana jest wartość NULL.

Przykłady i wyniki:

Przykład	Wynik
TDIST(1, 30, 2)	Zwraca wartość 0,3253

TInv

Funkcja `TInv()` zwraca wartość `t` rozkładu `t` studenta jako funkcję prawdopodobieństwa i stopni swobody.

Składnia:

```
TINV(prob, degrees_freedom)
```

Typ zwracanych danych: liczba

Argumenty:

Argumenty

Argument	Opis
prob	Prawdopodobieństwo dwustronne powiązane z rozkładem t. Musi być liczbą z zakresu od 0 do 1.
degrees_ freedom	Liczba całkowita określająca liczbę stopni swobody.

Ograniczenia:

Wszystkie argumenty muszą być liczbowe – w przeciwnym wypadku zwracana jest wartość NULL.

Ta funkcja jest powiązana z funkcją **TDist** w następujący sposób:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value.`

Przykłady i wyniki:

Przykład	Wynik
TINV(0.3253086, 30)	Zwraca wartość 1,0000

5.24 Funkcje ciągów znaków

W tej sekcji opisano funkcje do obsługi i modyfikacji ciągów znaków.

Wszystkie funkcje mogą być stosowane zarówno w skryptach ładowania danych, jak i wyrażeniach wykresu, poza funkcją **Evaluate**, która może być używana tylko w skrypcie ładowania danych.

Przegląd funkcji ciągów znaków

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

Capitalize

Funkcja **Capitalize()** zwraca ciąg znaków ze wszystkimi wyrazami pisanymi wielką literą.

```
Capitalize (text)
```

Chr

Funkcja **Chr()** zwraca znak Unicode odpowiadający wejściowej liczbie całkowitej.

```
Chr (int)
```

Evaluate

Funkcja **Evaluate()** wyszukuje, czy wejściowy ciąg tekstowy może zostać oceniony jako poprawne wyrażenie Qlik Sense, i jeśli tak, zwraca wartość wyrażenia jako ciąg. Jeśli ciąg wejściowy nie jest poprawnym wyrażeniem, zostanie zwrócona wartość NULL.

```
Evaluate (expression_text)
```

FindOneOf

Funkcja **FindOneOf()** przeszukuje ciąg znaków, aby znaleźć pozycję wystąpienia dowolnego znaku z zestawu udostępnionych znaków. Zwracana jest pozycja pierwszego wystąpienia dowolnego znaku z zestawu wyszukiwania, chyba że podano trzeci argument (o wartości większej niż 1). W razie braku dopasowania zwracane jest **0**.

```
FindOneOf (text, char_set[, count])
```

Hash128

Funkcja **Hash128()** zwraca 128-bitowy skrót połączonych wartości wyrażenia wejściowego. Wynik jest ciągiem 22 znaków.

```
Hash128 (expr{, expression})
```

Hash160

Funkcja **Hash160()** zwraca 160-bitowy skrót połączonych wartości wyrażenia wejściowego. Wynik jest ciągiem 27 znaków.

```
Hash160 (expr{, expression})
```

Hash256

Funkcja **Hash256()** zwraca 256-bitowy skrót połączonych wartości wyrażenia wejściowego. Wynik jest ciągiem 43 znaków.

```
Hash256 (expr{, expression})
```

Index

Funkcja **Index()** przeszukuje ciąg, aby znaleźć pozycję początkową n-tego wystąpienia podanego fragmentu tekstu. Opcjonalny trzeci argument dostarcza wartość n, która w przypadku pominięcia wynosi 1. Podanie wartości ujemnej powoduje wyszukiwanie od końca ciągu znaków. Pozycje w ciągu znaków są numerowane od **1** w górę.

```
Index  (text, substring[, count])
```

IsJson

IsJson() sprawdza, czy określony ciąg zawiera prawidłowe dane JSON (JavaScript Object Notation). Można także sprawdzić poprawność określonego typu danych JSON.

```
IsJson (json [, type])
```

JsonGet

JsonGet() zwraca ścieżkę ciągu danych JSON (JavaScript Object Notation). Dane JSON muszą mieć poprawny format JSON, ale mogą zawierać dodatkowe spacje i znaki nowego wiersza.

```
JsonGet (json, path)
```

JsonSet

JsonSet() modyfikuje ciąg zawierający dane JSON (JavaScript Object Notation). Może ustawić lub wstawić wartość JSON z nową lokalizacją określoną przez ścieżkę. Dane JSON muszą mieć poprawny format JSON, ale mogą zawierać dodatkowe spacje i znaki nowego wiersza.

```
JsonSet(json, path, value)
```

KeepChar

Funkcja **KeepChar()** zwraca ciąg znaków składający się z pierwszego ciągu, „text”, minus wszystkie znaki, których NIE ZAWIERA drugi ciąg, „keep_chars”.

```
KeepChar (text, keep_chars)
```

Left

Left() zwraca ciąg składający się z pierwszych (z lewej strony) znaków ciągu wejściowego, a liczbę tych znaków określa drugi argument.

```
Left (text, count)
```

Len

Funkcja **Len()** zwraca długość ciągu wejściowego.

```
Len (text)
```

LevenshteinDist

LevenshteinDist () zwraca odległość Levenshteina między dwoma ciągami. Jest ona definiowana jako minimalna liczba jednoznakowych edycji (wstawień, usunięć lub podstawień) wymaganych do zmiany jednego ciągu na drugi. Funkcja jest przydatna do porównań rozmytych ciągów.

```
LevenshteinDist (text1, text2)
```

Lower

Funkcja **Lower()** zamienia wszystkie znaki w ciągu wejściowym na małe litery.

```
Lower (text)
```

LTrim

Funkcja **LTrim()** zwraca ciąg wejściowy pozbawiony wszelkich spacji wiodących.

```
LTrim (text)
```

Mid

Funkcja **Mid()** zwraca część ciągu wejściowego zaczynającą się w pozycji znaku określonej przez drugi argument („start"), zwracając liczbę znaków określoną przez trzeci argument („count"). Jeśli parametr „count" zostanie pominięty, wówczas zostanie zwrócona reszta ciągu wejściowego. Pierwszy znak ciągu wejściowego ma numer 1.

```
Mid (text, start[, count])
```

Ord

Funkcja **Ord()** zwraca numer pozycji kodu Unicode pierwszego znaku w ciągu wejściowym.

```
Ord (text)
```

PurgeChar

Funkcja **PurgeChar()** zwraca ciąg znaków składający się ze znaków zawartych w ciągu wejściowym („text") z wyłączeniem tych, które pojawiają się w drugim argumencie („remove_chars").

```
PurgeChar (text, remove_chars)
```

Repeat

Funkcja **Repeat()** tworzy ciąg znaków składający się z ciągu wejściowego powtórzonego liczbę razy określoną przez drugi argument.

```
Repeat (text[, repeat_count])
```

Replace

Funkcja **Replace()** zwraca ciąg znaków po zastąpieniu wszystkich wystąpień danego fragmentu tekstu w ciągu wejściowym innym fragmentem tekstu. Ta funkcja jest nierekurencyjna i działa od lewej do prawej.

```
Replace (text, from_str, to_str)
```

Right

Right() zwraca ciąg składający się z ostatnich (z prawej strony) znaków ciągu wejściowego, a liczbę tych znaków określa drugi argument.

```
Right (text, count)
```

RTrim

Funkcja **RTrim()** zwraca ciąg wejściowy pozbawiony wszelkich spacji końcowych.

```
RTrim (text)
```

SubField

Funkcja **Subfield()** służy do wyodrębniania składników fragmentu tekstu z nadrzędnego pola ciągu znaków, gdy pierwotne pola rekordów składają się z co najmniej dwóch części rozdzielonych ogranicznikiem.

```
SubField (text, delimiter[, field_no ])
```

SubStringCount

Funkcja **SubStringCount()** zwraca liczbę wystąpień określonego fragmentu tekstu w tekście ciągu wejściowego. W przypadku braku dopasowań zwracane jest 0.

```
SubStringCount (text, substring)
```

TextBetween

Funkcja **TextBetween()** zwraca tekst w ciągu wejściowym, który występuje miedzy znakami określonymi jako ograniczniki.

```
TextBetween (text, delimiter1, delimiter2[, n])
```

Trim

Funkcja **Trim()** zwraca ciąg wejściowy pozbawiony wszelkich spacji wiodących i końcowych.

```
Trim (text)
```

Upper

Funkcja **Upper()** zamienia wszystkie znaki w ciągu wejściowym na wielkie litery w odniesieniu do wszystkich znaków tekstowych w wyrażeniu. Liczby i symbole są ignorowane.

```
Upper (text)
```

Capitalize

Funkcja **Capitalize()** zwraca ciąg znaków ze wszystkimi wyrazami pisanymi wielką literą.

Składnia:

```
Capitalize(text)
```

Typ zwracanych danych: ciąg znaków

Przykład: Wyrażenia wykresu

Przykład	Wynik
`Capitalize ( 'star trek' )`	Zwraca wartość 'Star Trek'
`Capitalize ( 'AA bb cC Dd')`	Zwraca wartość 'Aa Bb Cc Dd'

Przykład: Skrypt ładowania

```
Load  String, Capitalize(String) Inline [String rHode iSland washingTon d.C. new york];
```

Wynik

Ciąg znaków	Capitalize(String)
rHode iSland	Rhode Island
washingTon d.C.	Washington D.C.
new york	New York

Chr

Funkcja **Chr()** zwraca znak Unicode odpowiadający wejściowej liczbie całkowitej.

Składnia:

```
Chr(int)
```

Typ zwracanych danych: ciąg znaków

Przykłady i wyniki:

Przykład	Wynik
Chr(65)	Zwraca ciąg znaków 'A'
Chr(163)	Zwraca ciąg znaków '£'
Chr(35)	Zwraca ciąg znaków '#'

Evaluate

Funkcja **Evaluate()** wyszukuje, czy wejściowy ciąg tekstowy może zostać oceniony jako poprawne wyrażenie Qlik Sense, i jeśli tak, zwraca wartość wyrażenia jako ciąg. Jeśli ciąg wejściowy nie jest poprawnym wyrażeniem, zostanie zwrócona wartość NULL.

Składnia:

```
Evaluate(expression_text)
```

Typ zwracanych danych: dual

Tej funkcji ciągu znaków nie można używać w wyrażeniach wykresu.

Przykłady i wyniki:

Przykład funkcji	Wynik
`Evaluate ( 5 * 8 )`	Zwraca wartość '40'

Przykład skryptu ładowania

```
Load  Evaluate(String) as Evaluated, String Inline [String 4 5+3 0123456789012345678 Today()
];
```

Wynik

Ciąg znaków	Oceniany
4	4
5+3	8
0123456789012345678	0123456789012345678
Today()	2022-02-02

FindOneOf

Funkcja **FindOneOf()** przeszukuje ciąg znaków, aby znaleźć pozycję wystąpienia dowolnego znaku z zestawu udostępnionych znaków. Zwracana jest pozycja pierwszego wystąpienia dowolnego znaku z zestawu wyszukiwania, chyba że podano trzeci argument (o wartości większej niż 1). W razie braku dopasowania zwracane jest **0**.

Składnia:

```
FindOneOf(text, char_set[, count])
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
text	Pierwotny ciąg znaków.
char_set	Zestaw znaków do wyszukania w tekście text.
count	Określa wystąpienie każdego znaku, który ma zostać wyszukany. Na przykład wartość dwóch wyszukań dla drugiego wystąpienia.

Przykład: Wyrażenia wykresu

Przykład	Wynik
`FindOneOf( 'my example text string', 'et%s')`	Zwraca „4”, ponieważ „e” jest czwartym znakiem w przykładowym ciągu.
`FindOneOf( 'my example text string', 'et%s', 3)`	Zwraca „12”, ponieważ wyszukiwanie dotyczy dowolnego z następujących znaków: e, t, % lub s, a „t” to trzecie wystąpienie na pozycji 12 przykładowego ciągu.
`FindOneOf( 'my example text string', '¤%&')`	Zwraca „0”, ponieważ żaden ze znaków ¤, %, lub & nie istnieje w przykładowym ciągu.

Przykład: Skrypt ładowania

```
Load * Inline [SearchFor, Occurrence et%s,1 et%s,3 ¤%&,1]
```

Wynik

SearchFor	Occurrence	FindOneOf('my example text string', SearchFor, Occurrence)
et%s	1	4
et%s	3	12
¤%&	1	0

Hash128

Funkcja **Hash128()** zwraca 128-bitowy skrót połączonych wartości wyrażenia wejściowego. Wynik jest ciągiem 22 znaków.

Składnia:

```
Hash128(expr{, expression})
```

Typ zwracanych danych: ciąg znaków

Przykład: Wyrażenia wykresu

Przykład	Wynik
Hash128 ('abc', 'xyz', '123')	Zwraca „MA&5]6+3=:>:>G%S<U*S2+”.
Hash128 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Zwraca „G7*=6GKPJ(Z+)^KM?<$'A+”.

Przykład: Skrypt ładowania

```
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 ];
```

Wynik

Region	Rok	Miesiąc	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

Funkcja **Hash160()** zwraca 160-bitowy skrót połączonych wartości wyrażenia wejściowego. Wynik jest ciągiem 27 znaków.

Składnia:

```
Hash160(expr{, expression})
```

Typ zwracanych danych: ciąg znaków

Przykład: Wyrażenia wykresu

Przykład	Wynik
Hash160 ('abc', 'xyz', '123')	Zwraca „MA&5]6+3=:>;>G%S<U*S2I:`=X*”.
Hash160 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Zwraca „G7*=6GKPJ (Z+)^KM?<$'AI.)?U$”.

Przykład: Skrypt ładowania

```
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 ];
```

Wynik

Region	Rok	Miesiąc	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

Funkcja **Hash256()** zwraca 256-bitowy skrót połączonych wartości wyrażenia wejściowego. Wynik jest ciągiem 43 znaków.

Składnia:

```
Hash256(expr{, expression})
```

Typ zwracanych danych: ciąg znaków

Przykład: Wyrażenia wykresu

Przykład	Wynik
Hash256 ('abc', 'xyz', '123')	Zwraca „MA&5]6+3=:>;>G%S<U*S2I:`=X*A.IO*8N\%Y7Q;YEJ”.
Hash256 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Zwraca „G7*=6GKPJ(Z+)^KM?<$'AI.)?U$#X2RB [:0ZP=+Z`F:”.

Przykład: Skrypt ładowania

```
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 ];
```

Wynik

Region	Rok	Miesiąc	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

Funkcja **Index()** przeszukuje ciąg, aby znaleźć pozycję początkową n-tego wystąpienia podanego fragmentu tekstu. Opcjonalny trzeci argument dostarcza wartość n, która w przypadku pominięcia wynosi 1. Podanie wartości ujemnej powoduje wyszukiwanie od końca ciągu znaków. Pozycje w ciągu znaków są numerowane od **1** w górę.

Składnia:

```
Index(text, substring[, count])
```

Typ zwracanych danych: integer

Argumenty:

Argumenty

Argument	Opis
text	Pierwotny ciąg znaków.
substring	Ciąg znaków do wyszukania w tekście text.
count	Określa wystąpienie **substring** do wyszukania. Na przykład wartość dwóch wyszukań dla drugiego wystąpienia.

Przykłady i wyniki:

Przykład	Wynik
Index('abcdefg', 'cd')	Zwraca wartość 3
Index('abcdabcd', 'b', 2)	Zwraca 6 (drugie wystąpienie „b”)
Index('abcdabcd', 'b',-2)	Zwraca 2 (drugie wystąpienie „b” od końca)
`Left( Date, Index( Date,'-') -1 )` where **Date** = 1997-07-14	Zwraca wartość 1997
`Mid( Date, Index( Date, '-', 2 ) -2, 2 )` where **Date** = 1997-07-14	Zwraca wartość 07

Przykład: Skrypt

```
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() sprawdza, czy określony ciąg zawiera prawidłowe dane JSON (JavaScript Object Notation). Można także sprawdzić poprawność określonego typu danych JSON.

Składnia:

```
value IsJson(json [, type])
```

Typ zwracanych danych: dual

Argumenty

Argument	Opis
json	Ciąg do przetestowania. Może zawierać dodatkowe spacje lub znaki nowego wiersza.

Argument	Opis
type	Opcjonalny argument określający typ danych JSON do przetestowania. • 'value' (domyślna) • 'object' • 'array' • 'string' • 'number' • 'Boolean' • 'null'

Przykład: Prawidłowy format JSON i typ

Przykład	Wynik
`IsJson('null')`	Zwraca wartość -1 (true)
`IsJson('"abc"', 'value')`	Zwraca wartość -1 (true)
`IsJson('"abc"', 'string')`	Zwraca wartość -1 (true)
`IsJson(123, 'number')`	Zwraca wartość -1 (true)

Przykład: Nieprawidłowy format JSON lub typ

Przykład	Wynik	Opis
`IsJson('text')`	Zwraca wartość 0 (false)	'text' nie jest prawidłową wartością JSON
`IsJson('"text"', 'number')`	Zwraca wartość 0 (false)	'"text"' nie jest prawidłową liczbą JSON
`IsJson('"text"', 'text')`	Zwraca wartość 0 (false)	'text' nie jest prawidłowym typem JSON

JsonGet

JsonGet() zwraca ścieżkę ciągu danych JSON (JavaScript Object Notation). Dane JSON muszą mieć poprawny format JSON, ale mogą zawierać dodatkowe spacje i znaki nowego wiersza.

Składnia:

```
value JsonGet(json, path)
```

Typ zwracanych danych: dual

Argumenty

Argument	Opis
json	Ciąg zawierający dane JSON.
path	Ścieżka musi spełniać warunki normy ≤ RFC 6901. Umożliwi to wyszukiwanie właściwości w danych w formacie JSON bez użycia skomplikowanych funkcji operujących na fragmentach tekstu lub indeksujących.

Przykład: Prawidłowe format JSON i ścieżka

Przykład	Wynik
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '')`	Zwraca wartość '{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/a')`	Zwraca wartość '{"foo":"bar"}'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/a/foo')`	Zwraca wartość '"bar"'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b')`	Zwraca wartość '[123,"abc","ABC"]'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/0')`	Zwraca wartość '123'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/1')`	Zwraca wartość '"abc"'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/2')`	Zwraca wartość '"ABC"'

Przykład: Nieprawidłowe format JSON lub ścieżka

Przykład	Wynik	Opis
`JsonGet('{"a":"b"}','/b')`	Zwraca wartość null	Ścieżka nie wskazuje prawidłowej części danych w formacie JSON.
`JsonGet('{"a"}','/a')`	Zwraca wartość null	Nieprawidłowy format danych JSON (składowa „a” nie ma wartości).

JsonSet

JsonSet() modyfikuje ciąg zawierający dane JSON (JavaScript Object Notation). Może ustawić lub wstawić wartość JSON z nową lokalizacją określoną przez ścieżkę. Dane JSON muszą mieć poprawny format JSON, ale mogą zawierać dodatkowe spacje i znaki nowego wiersza.

Składnia:

```
value JsonSet(json, path, value)
```

Typ zwracanych danych: dual

Argumenty

Argument	Opis
json	Ciąg zawierający dane JSON.
path	Ścieżka musi spełniać warunki normy ≤ RFC 6901. Umożliwia to budowanie właściwości w danych w formacie JSON bez użycia skomplikowanych funkcji operujących na fragmentach tekstu lub indeksujących oraz instrukcji konkatenacji.
value	Nowa wartość ciągu w formacie JSON.

Przykład: Prawidłowe format JSON, ścieżka i wartość

Przykład	Wynik
`JsonSet('{}','/a','"b"')`	Zwraca wartość '{"a":"b"}'
`JsonSet('[]','/0','"x"')`	Zwraca wartość '["x"]'
`JsonSet('"abc"','','123')`	Zwraca wartość 123

Przykład: Nieprawidłowe format JSON, ścieżka lub wartość

Przykład	Wynik	Opis
`JsonSet('"abc"','/x','123')`	Zwraca wartość null	Ścieżka nie wskazuje prawidłowej części danych w formacie JSON.
`JsonSet('{"a": {"b":"c"}}','a/b','"x"')`	Zwraca wartość null	Nieprawidłowa ścieżka.
`JsonSet(' {"a":"b"}','/a','abc')`	Zwraca wartość null	Nieprawidłowy format JSON wartości. Ciąg musi być ujęty w cudzysłów.

KeepChar

Funkcja **KeepChar()** zwraca ciąg znaków składający się z pierwszego ciągu, „text”, minus wszystkie znaki, których NIE ZAWIERA drugi ciąg, „keep_chars”.

Składnia:

```
KeepChar(text, keep_chars)
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
text	Pierwotny ciąg znaków.
keep_chars	Ciąg znaków zawierający znaki w tekście text do przechowania.

Przykład: Wyrażenia wykresu

Przykład	Wynik
`KeepChar ( 'a1b2c3','123' )`	Zwraca „123”.
`KeepChar ( 'a1b2c3','1234' )`	Zwraca „123”.
`KeepChar ( 'a1b22c3','1234' )`	Zwraca „1223”.
`KeepChar ( 'a1b2c3','312' )`	Zwraca „123”.

Przykład: Skrypt ładowania

```
T1: Load  *,  keepchar(String1, String2) as KeepChar;   Load * inline [ String1, String2
'a1b2c3', '123' ];
```

Wyniki

Tabela programu Qlik Sense przedstawiająca wynik użycia funkcji *KeepChar* w skrypcie ładowania.

String1	String2	KeepChar
a1b2c3	123	123

Zob. także:

p *PurgeChar (page 1426)*

Left

Left() zwraca ciąg składający się z pierwszych (z lewej strony) znaków ciągu wejściowego, a liczbę tych znaków określa drugi argument.

Składnia:

```
Left(text, count)
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argument	Opis
text	Pierwotny ciąg znaków.
count	Określa liczbę znaków do uwzględnienia od lewej części ciągu znaków **text**.

Przykład: Wyrażenie wykresu

Przykład	Wynik
`Left('abcdef', 3)`	Zwraca „abc”

Przykład: Skrypt ładowania

```
T1: Load  *,  left(Text,Start) as Left;        Load * inline [ Text, Start 'abcdef', 3 '2021-07-14', 4 '2021-07-14', 2 ];
```

Wynik

Tabela programu Qlik Sense przedstawiająca wynik użycia funkcji *Left* w skrypcie ładowania.

Tekst	Początek	Left
abcdef	3	abc
2021-07-14	4	2021
2021-07-14	2	20

p Zobacz też *Index (page 1415)*, co umożliwia bardziej złożoną analizę ciągów.

Len

Funkcja **Len()** zwraca długość ciągu wejściowego.

Składnia:

```
Len(text)
```

Typ zwracanych danych: integer

Przykład: Wyrażenie wykresu

Przykład	Wynik
`Len('Peter')`	Zwraca „5”.

Przykład: Skrypt ładowania

```
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 ];
```

Wynik

Ciąg znaków	NewString
this is a sample text string	This is a sample text string
capitalize first letter only	Capitalize first letter only

LevenshteinDist

LevenshteinDist () zwraca odległość Levenshteina między dwoma ciągami. Jest ona definiowana jako minimalna liczba jednoznakowych edycji (wstawień, usunięć lub podstawień) wymaganych do zmiany jednego ciągu na drugi. Funkcja jest przydatna do porównań rozmytych ciągów.

Składnia:

```
LevenshteinDist(text1, text2)
```

Typ zwracanych danych: integer

Przykład: Wyrażenie wykresu

Przykład	Wynik
`LevenshteinDist('Kitten','Sitting')`	Zwraca wartość 3

Przykład: Skrypt ładowania

Skrypt ładowania

```
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;
```

Wynik

Identyfikator	String_1	String_2	LevenshteinDistance
1	Silver	Sliver	2
2	Silver	SSiver	2
3	Silver	SSiveer	3
4	Gold	Pogrubienie	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

Funkcja **Lower()** zamienia wszystkie znaki w ciągu wejściowym na małe litery.

Składnia:

```
Lower(text)
```

Typ zwracanych danych: ciąg znaków

Przykład: Wyrażenie wykresu

Przykład	Wynik
`Lower('abcD')`	Zwraca wartość 'abcd'

Przykład: Skrypt ładowania

```
Load  String, Lower(String) Inline [String rHode iSland washingTon d.C. new york];
```

Wynik

Ciąg znaków	Lower(String)
rHode iSland	rhode island
washingTon d.C.	washington d.c.
new york	new york

LTrim

Funkcja **LTrim()** zwraca ciąg wejściowy pozbawiony wszelkich spacji wiodących.

Składnia:

```
LTrim(text)
```

Typ zwracanych danych: ciąg znaków

Przykład: Wyrażenia wykresu

Przykład	Wynik
`LTrim( ' abc' )`	Zwraca wartość 'abc'
`LTrim( 'abc ' )`	Zwraca wartość 'abc '

Przykład: Skrypt ładowania

```
Set verbatim=1;  T1:  Load *, len(LtrimString) as LtrimStringLength;       Load *, ltrim
(String) as LtrimString; Load *, len(String) as StringLength;                Load * Inline [
String '   abc    ' ' def  '];
```

Instrukcja „Set verbatim=1” jest zawarta w przykładzie, aby zapobiec automatycznemu przycięciu spacji przed demonstracją funkcji ltrim. Więcej informacji zawiera temat Verbatim (page 196).

Wynik

Ciąg znaków	StringLength	LtrimStringLength
def	6	5
abc	10	7

Zob. także:

p *RTrim (page 1429)*

Mid

Funkcja **Mid()** zwraca część ciągu wejściowego zaczynającą się w pozycji znaku określonej przez drugi argument („start”), zwracając liczbę znaków określoną przez trzeci argument („count”). Jeśli parametr „count” zostanie pominięty, wówczas zostanie zwrócona reszta ciągu wejściowego. Pierwszy znak ciągu wejściowego ma numer 1.

Składnia:

```
Mid(text, start[, count])
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
text	Pierwotny ciąg znaków.
start	Liczba całkowita określająca pozycję pierwszego znaku w tekście text do uwzględnienia.
count	Określa długość wyjściowego ciągu znaków. W przypadku pominięcia uwzględniane są wszystkie znaki określone przez wartość **start**.

Przykład: Wyrażenia wykresu

Przykład	Wynik
`Mid('abcdef',3 )`	Zwraca wartość 'cdef'
`Mid('abcdef',3, 2 )`	Zwraca wartość 'cd'

Przykład: Skrypt ładowania

```
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 ];
```

Wynik

Tabela programu Qlik Sense przedstawiająca wynik użycia funkcji *Mid* w skrypcie ładowania.

Tekst	Początek	Mid1	Łącznie	Mid2
abcdef	2	bcdef	3	bcd
abcdef	3	cdef	2	cd
210714	2	10714	3	107
210714	3	0714	2	07

Zob. także:

p *Index (page 1415)*

Ord

Funkcja **Ord()** zwraca numer pozycji kodu Unicode pierwszego znaku w ciągu wejściowym.

Składnia:

```
Ord(text)
```

Typ zwracanych danych: integer

Przykłady i wyniki:

Przykład: Wyrażenie wykresu

Przykład	Wynik
`Ord('A')`	Zwraca liczbę całkowitą 65.
`Ord('Ab')`	Zwraca liczbę całkowitą 65.

Przykład: Skrypt ładowania

```
//Guqin (Chinese: 古琴) - 7-stringed zithers T2: Load *,  ord(Chinese) as OrdUnicode,
        ord(Western) as OrdASCII;                    Load * inline [ Chinese, Western 古琴,
Guqin ];
```

Wynik:

Chinese	Western	OrdASCII	OrdUnicode
古琴	Guqin	71	21476

PurgeChar

Funkcja **PurgeChar()** zwraca ciąg znaków składający się ze znaków zawartych w ciągu wejściowym („text”) z wyłączeniem tych, które pojawiają się w drugim argumencie („remove_chars”).

Składnia:

```
PurgeChar(text, remove_chars)
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
text	Pierwotny ciąg znaków.
remove_chars	Ciąg znaków zawierający znaki w tekście text do usunięcia.

Typ zwracanych danych: ciąg znaków

Przykład: Wyrażenia wykresu

Przykład	Wynik
`PurgeChar ( 'a1b2c3','123' )`	Zwraca „abc”.
`PurgeChar ( 'a1b2c3','312' )`	Zwraca „abc”.

Przykład: Skrypt ładowania

```
T1: Load  *,  purgechar(String1, String2) as PurgeChar;   Load * inline [ String1, String2
'a1b2c3', '123' ];
```

Wyniki

Tabela programu Qlik Sense przedstawiająca wynik użycia funkcji *PurgeChar* w skrypcie ładowania.

String1	String2	PurgeChar
a1b2c3	123	abc

Zob. także:

p *KeepChar (page 1419)*

Repeat

Funkcja **Repeat()** tworzy ciąg znaków składający się z ciągu wejściowego powtórzonego liczbę razy określoną przez drugi argument.

Składnia:

```
Repeat(text[, repeat_count])
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
text	Pierwotny ciąg znaków.
repeat_count	Określa, ile razy znak w tekście **text** ciągu ma być powtórzony w ciągu wyjściowym.

Przykład: Wyrażenie wykresu

Przykład	Wynik
`Repeat( ' * ', rating )` when **rating** = 4	Zwraca wartość '****'

Przykład: Skrypt ładowania

```
T1: Load *, repeat(String,2) as Repeat; Load * inline [ String hello world! hOw aRe you? ];
```

Wynik

Ciąg znaków	Powtarzaj
hello world!	hello world!hello world!
hOw aRe you?	hOw aRe you?hOw aRe you?

Replace

Funkcja **Replace()** zwraca ciąg znaków po zastąpieniu wszystkich wystąpień danego fragmentu tekstu w ciągu wejściowym innym fragmentem tekstu. Ta funkcja jest nierekurencyjna i działa od lewej do prawej.

Składnia:

```
Replace(text, from_str, to_str)
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
text	Pierwotny ciąg znaków.
from_str	Ciąg znaków, który może występować co najmniej raz w ciągu znaków **text**.
to_str	Ciąg znaków, który zastąpi wszystkie wystąpienia **from_str** w tekście **text** ciągu znaków.

Przykłady i wyniki:

Przykład	Wynik
`Replace('abccde','cc','xyz')`	Zwraca wartość 'abxyzde'

Zob. także:

Right

Right() zwraca ciąg składający się z ostatnich (z prawej strony) znaków ciągu wejściowego, a liczbę tych znaków określa drugi argument.

Składnia:

```
Right(text, count)
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
text	Pierwotny ciąg znaków.
count	Określa liczbę znaków do uwzględnienia od prawej części ciągu **text**.

Przykład: Wyrażenie wykresu

Przykład	Wynik
`Right('abcdef', 3)`	Zwraca wartość 'def'

Przykład: Skrypt ładowania

```
T1: Load  *,  right(Text,Start) as Right;        Load * inline [ Text, Start 'abcdef', 3
'2021-07-14', 4 '2021-07-14', 2 ];
```

Wynik

Tabela programu Qlik Sense przedstawiająca wynik użycia funkcji *Right* w skrypcie ładowania.

Tekst	Początek	Right
abcdef	3	def
2021-07-14	4	7-14
2021-07-14	2	14

RTrim

Funkcja **RTrim()** zwraca ciąg wejściowy pozbawiony wszelkich spacji końcowych.

Składnia:

```
RTrim(text)
```

Typ zwracanych danych: ciąg znaków

Przykład: Wyrażenia wykresu

Przykład	Wynik
`RTrim( ' abc' )`	Zwraca wartość ' abc'
`RTrim( 'abc ' )`	Zwraca wartość 'abc'

Przykład: Skrypt ładowania

```
Set verbatim=1;    T1:  Load *, len(RtrimString) as RtrimStringLength;         Load *, rtrim
(String) as RtrimString; Load *, len(String) as StringLength;              Load * Inline [
String '   abc    ' ' def  '];
```

Instrukcja „Set verbatim=1" jest zawarta w przykładzie, aby zapobiec automatycznemu przycięciu spacji przed demonstracją funkcji rtrim. Więcej informacji zawiera temat Verbatim (page 196).

Wynik

Ciąg znaków	StringLength	RtrimStringLength
def	6	4
abc	10	6

Zob. także:

p *LTrim (page 1424)*

SubField

Funkcja **Subfield()** służy do wyodrębniania składników fragmentu tekstu z nadrzędnego pola ciągu znaków, gdy pierwotne pola rekordów składają się z co najmniej dwóch części rozdzielonych ogranicznikiem.

Funkcja **Subfield()** może być używana na przykład do wyodrębniania imienia i nazwiska z listy rekordów składających się z nazwisk, elementów nazwy ścieżki, lub wyodrębniania danych z tabel rozdzielanych przecinkami.

W przypadku korzystania z funkcji **Subfield()** w instrukcji **LOAD** z pozostawionym opcjonalnym parametrem field_no, dla każdego ciągu podrzędnego zostanie wygenerowany jeden pełny rekord. Jeśli przy użyciu funkcji **Subfield()** załadowano kilka pól, wówczas zostanie wygenerowany iloczyn kartezjański wszystkich kombinacji.

Składnia:

```
SubField(text, delimiter[, field_no ])
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argumenty

Argument	Opis
text	Pierwotny ciąg znaków. Może to być tekst ustalony, zmienna, rozszerzenie przez znak dolara lub inne wyrażenie.
delimiter	Znak w tekście wejściowym **text**, który dzieli ciąg znaków na elementy.
field_no	Opcjonalny trzeci argument jest liczbą całkowitą, określającą, które z ciągów podrzędnych tekstu nadrzędnego **text** ciągu znaków mają zostać zwrócone. Użycie wartości 1 spowoduje zwrócenie pierwszego fragmentu tekstu, a wartości 2 zwrócenie drugiego fragmentu tekstu, i tak dalej. • Jeśli **field_no** jest wartością dodatnią, wówczas podciągi są wyodrębniane od lewej do prawej. • Jeśli **field_no** jest wartością ujemną, wówczas podciągi są wyodrębniane od prawej do lewej.

Funkcja SubField() może być stosowana zamiast złożonych kombinacji funkcji, takich jak Len (), Right(), Left(), Mid() i inne funkcje ciągów znaków.

Przykłady: Wyrażenia skryptów i wykresów używające funkcji SubField

Przykłady – wyrażenia skryptów i wykresów

Przykłady podstawowe

Przykład	Wynik
`SubField(S, ';' ,2)`	Zwraca 'cde', jeśli **S** wynosi 'abc;cde;efg'.
`SubField(S, ';' ,1)`	Zwraca pusty ciąg znaków, jeśli **S** jest pustym ciągiem znaków.
`SubField(S, ';' ,1)`	Zwraca pusty ciąg znaków, jeśli **S** jest równe ';'.

Przykład	Wynik
Załóżmy, że istnieje zmienna, która przechowuje nazwę ścieżki vMyPath, `Set vMyPath=\Users\ext_jrb\Documents\Qlik\Sense\Apps;.`	Na wykresie typu Tekst i grafika można dodać miarę, taką jak: `SubField(vMyPath, '\',-3)`, czego wynikiem będzie „Qlik”, ponieważ jest to fragment ciągu trzeci od prawej strony zmiennej vMyPath.

Przykład skryptu 1

Skrypt ładowania

Załaduj następujące wyrażenia skryptu i dane w edytorze ładowania danych.

```
FullName:                                    LOAD * inline [ Name 'Dave Owen' 'Joe Tem' ];  SepNames:  Lo
(Name, ' ',1) as FirstName, SubField(Name, ' ',-1) as Surname Resident FullName; Drop Table
FullName;
```

Tworzenie wizualizacji

Utwórz wizualizację tabeli w arkuszu Qlik Sense z wymiarami **Name**, **FirstName** i **SurName**.

Wynik

Name	FirstName	SurName
Dave Owen	Dave	Owen
Joe Tem	Joe	Tem

Objaśnienie

Funkcja **SubField()** wyodrębnia pierwszy podciąg z **Name**, ustawiając argument **field_no** na 1. Ponieważ wartość **field_no** jest dodatnia, w celu wyodrębnienia podciągu stosuje się kolejność od lewej do prawej. Drugie wywołanie funkcji wyodrębnia drugi podciąg, ustawiając argument **field _no** na -1, co powoduje wyodrębnienie podciągu w kolejności od prawej do lewej.

Przykład skryptu 2

Skrypt ładowania

Załaduj następujące wyrażenia skryptu i dane w edytorze ładowania danych.

```
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 '|');
```

Tworzenie wizualizacji

Utwórz wizualizację tabeli w arkuszu Qlik Sense z wymiarami **Instrument**, **Player** i **Project**

Wynik

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

Objaśnienie

Ten przykład przedstawia, jak używając wielu instancji funkcji **Subfield()** (w każdej pomijając parametr field_no) z tej samej instrukcji **LOAD** powoduje utworzenie iloczynów kartezjańskich wszystkich kombinacji. Opcja **DISTINCT** służy do unikania tworzenia zduplikowanych rekordów.

SubStringCount

Funkcja **SubStringCount()** zwraca liczbę wystąpień określonego fragmentu tekstu w tekście ciągu wejściowego. W przypadku braku dopasowań zwracane jest 0.

Składnia:

```
SubStringCount(text, sub_string)
```

Typ zwracanych danych: integer

Argumenty:

Argument	Opis
text	Pierwotny ciąg znaków.
sub_string	Ciąg znaków, który może występować co najmniej raz w tekście **text** wejściowego ciągu znaków.

Przykład: Wyrażenia wykresu

Przykład	Wynik
`SubStringCount ( 'abcdefgcdxyz', 'cd' )`	Zwraca „2”.
`SubStringCount ( 'abcdefgcdxyz', 'dc' )`	Zwraca „0”.

Przykład: Skrypt ładowania

```
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 ];
```

Wynik

Ciągi znaków	SubStringCount_AB
aB/cd/ef/gh/Abc/abandoned	3
ABC:DEF:GHI:AB:CD:EF:GH	2

TextBetween

Funkcja **TextBetween()** zwraca tekst w ciągu wejściowym, który występuje miedzy znakami określonymi jako ograniczniki.

Składnia:

```
TextBetween(text, delimiter1, delimiter2[, n])
```

Typ zwracanych danych: ciąg znaków

Argumenty:

Argument	Opis
text	Pierwotny ciąg znaków.
delimiter1	Określa pierwszy rozdzielający znak (lub ciąg znaków) do wyszukania w tekście **text**.
delimiter2	Określa drugi rozdzielający znak (lub ciąg znaków) do wyszukania w tekście **text**.
n	Określa, między którym wystąpieniem pary znaków rozdzielających należy przeprowadzić wyszukiwanie. Na przykład wartość 2 zwraca znaki pomiędzy drugim wystąpieniem delimiter1 a drugim wystąpieniem delimiter2.

Przykład: Wyrażenia wykresu

Przykład	Wynik
`TextBetween('<abc>', '<', '>')`	Zwraca wartość 'abc'
`TextBetween('<abc><de>', '<', '>',2)`	Zwraca wartość 'de'

Przykład	Wynik
`TextBetween('abc', '<', '>')` `TextBetween('<a<b', '<', '>')`	Oba przykłady zwracają wartość NULL. Jeśli w ciągu nie zostanie znaleziony żaden separator, zostanie zwrócona wartość NULL.
`TextBetween('<>', '<', '>')`	Zwraca ciąg o długości zerowej.
`TextBetween('<abc>', '<', '>', 2)`	Zwraca NULL, ponieważ n jest większe od liczby wystąpień separatorów.

Przykład: Skrypt ładowania

```
Load *, textbetween(Text,'<','>') as TextBetween, textbetween(Text,'<','>',2) as
SecondTextBetween; Load * inline [ Text <abc><de> <def><ghi><jkl> ];
```

Wynik

Tekst	TextBetween	SecondTextBetween
<abc><de>	abc	de
<def><ghi><jkl>	def	ghi

Trim

Funkcja **Trim()** zwraca ciąg wejściowy pozbawiony wszelkich spacji wiodących i końcowych.

Składnia:

```
Trim(text)
```

Typ zwracanych danych: ciąg znaków

Przykłady i wyniki:

Przykład: Wyrażenie wykresu

Przykład	Wynik
`Trim( ' abc' )`	Zwraca wartość 'abc'
`Trim( 'abc ' )`	Zwraca wartość 'abc'
`Trim( ' abc ' )`	Zwraca wartość 'abc'

Przykład: Skrypt ładowania

```
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');
```

Instrukcja „Set verbatim=1" jest zawarta w przykładzie, aby zapobiec automatycznemu przycięciu spacji przed demonstracją funkcji trim. Więcej informacji zawiera temat Verbatim (page 196).

Wynik:

Ciąg znaków	StringLength	TrimStringLength
def	6	3
abc	10	3

Upper

Funkcja **Upper()** zamienia wszystkie znaki w ciągu wejściowym na wielkie litery w odniesieniu do wszystkich znaków tekstowych w wyrażeniu. Liczby i symbole są ignorowane.

Składnia:

```
Upper(text)
```

Typ zwracanych danych: ciąg znaków

Przykład: Wyrażenie wykresu

Przykład	Wynik
`Upper(' abcD')`	Zwraca wartość 'ABCD'

Przykład: Skrypt ładowania

```
Load  String,Upper(String) Inline [String rHode iSland washingTon d.C. new york];
```

Wynik

Ciąg znaków	Upper(String)
rHode iSland	RHODE ISLAND
washingTon d.C.	WASHINGTON D.C.
new york	NEW YORK

5.25 Funkcje systemowe

Funkcje systemowe udostępniają funkcje dotyczące dostępu do właściwości systemu, urządzenia i aplikacji Qlik Sense.

Przegląd funkcji systemowych

Niektóre z funkcji są po podsumowaniu opisane bardziej szczegółowo. W przypadku tych funkcji można kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

Author()

Ta funkcja zwraca ciąg znaków zawierający właściwość autora bieżącej aplikacji. Może być używane w skrypcie ładowania danych oraz w wyrażeniu wykresu.

Właściwość Author nie może być ustawiana w bieżącej wersji programu Qlik Sense. W przypadku migracji dokumentu QlikView właściwość author zostanie zachowana.

ClientPlatform()

Ta funkcja zwraca ciąg znaków agenta użytkownika dotyczący przeglądarki klienta. Może być używane w skrypcie ładowania danych oraz w wyrażeniu wykresu.

Przykład:

```
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/35.0.1916.114 Safari/537.36
```

ComputerName

Ta funkcja zwraca ciąg znaków zawierający nazwę komputera, jaką zwraca system operacyjny. Może być używane w skrypcie ładowania danych oraz w wyrażeniu wykresu.

Jeśli nazwa komputera zawiera więcej niż 15 znaków, wówczas ciąg znaków będzie zawierał tylko 15 pierwszych znaków.

```
ComputerName( )
```

DocumentName

Ta funkcja zwraca ciąg znaków zawierający nazwę bieżącej aplikacji Qlik Sense bez ścieżki, ale z rozszerzeniem. Może być używane w skrypcie ładowania danych oraz w wyrażeniu wykresu.

```
DocumentName( )
```

DocumentPath

Ta funkcja zwraca ciąg znaków zawierający pełną ścieżkę do bieżącej aplikacji Qlik Sense. Może być używane w skrypcie ładowania danych oraz w wyrażeniu wykresu.

```
DocumentPath( )
```

Ta funkcja nie jest obsługiwana w trybie standardowym.

DocumentTitle

Ta funkcja zwraca ciąg znaków zawierający tytuł bieżącej aplikacji Qlik Sense. Może być używane w skrypcie ładowania danych oraz w wyrażeniu wykresu.

```
DocumentTitle( )
```

EngineVersion

Ta funkcja zwraca pełną wersję silnika Qlik Sense jako ciąg znaków.

```
EngineVersion ()
```

GetCollationLocale

Ta funkcja skryptu zwraca nazwę kulturową stosowanej leksykografii z uwzględnieniem znaków diakrytycznych. Jeśli nie ustawiono zmiennej CollationLocale, zwracane są rzeczywiste ustawienia regionalne komputera użytkownika.

```
GetCollationLocale( )
```

GetObjectField

Funkcja **GetObjectField()** zwraca nazwę wymiaru. **Index** jest opcjonalną liczbą całkowitą wskazującą, który z wymiarów powinien być zwracany.

```
GetObjectField – funkcja wykresu([index])
```

GetRegistryString

Ta funkcja zwraca wartość klucza w rejestrze Windows. Może być używane w skrypcie ładowania danych oraz w wyrażeniu wykresu.

```
GetRegistryString(path, key)
```

Ta funkcja nie jest obsługiwana w trybie standardowym.

IsPartialReload

Ta funkcja zwraca - 1 (True), jeśli bieżące przeładowanie było częściowe, a 0 (False) w przeciwnym przypadku.

```
IsPartialReload ()
```

OSUser

Ta funkcja zwraca ciąg znaków zawierający nazwę użytkownika, który aktualnie jest podłączony. Może być używane w skrypcie ładowania danych oraz w wyrażeniu wykresu.

```
OSUser( )
```

W programach Qlik Sense Desktop i Qlik Sense Mobile Client Managed ta funkcja zawsze zwraca „Personal\Me”.

ProductVersion

Ta funkcja zwraca pełne oznaczenie wersji i numer kompilacji aplikacji Qlik Sense jako ciąg znaków.

Ta funkcja jest przestarzała i została zastąpiona funkcją **EngineVersion()**.

```
ProductVersion ()
```

ReloadTime

Ta funkcja zwraca znacznik czasu odnoszący się do zakończenia ostatniego ładowania danych. Może być używane w skrypcie ładowania danych oraz w wyrażeniu wykresu.

```
ReloadTime( )
```

StateName

Funkcja **StateName()** zwraca nazwę stanu alternatywnego wizualizacji, w której jest używana. Funkcji StateName można na przykład użyć do tworzenia wizualizacji z dynamicznym tekstem i kolorami, aby odzwierciedlić zmianę stanu wizualizacji. Tę funkcję można wykorzystać w wyrażeniach wykresów, ale nie można jej użyć do określenia stanu, do którego odwołuje się dane wyrażenie.

```
StateName – funkcja wykresu()
```

EngineVersion

Ta funkcja zwraca pełną wersję silnika Qlik Sense jako ciąg znaków.

Składnia:

```
EngineVersion()
```

IsPartialReload

Ta funkcja zwraca - 1 (True), jeśli bieżące przeładowanie było częściowe, a 0 (False) w przeciwnym przypadku.

Składnia:

```
IsPartialReload()
```

ProductVersion

Ta funkcja zwraca pełne oznaczenie wersji i numer kompilacji aplikacji Qlik Sense jako ciąg znaków. Ta funkcja jest przestarzała i została zastąpiona funkcją **EngineVersion()**.

Składnia:

```
ProductVersion()
```

StateName – funkcja wykresu

Funkcja **StateName()** zwraca nazwę stanu alternatywnego wizualizacji, w której jest używana. Funkcji StateName można na przykład użyć do tworzenia wizualizacji z dynamicznym tekstem i kolorami, aby odzwierciedlić zmianę stanu wizualizacji. Tę funkcję można wykorzystać w wyrażeniach wykresów, ale nie można jej użyć do określenia stanu, do którego odwołuje się dane wyrażenie.

Składnia:

```
StateName ()
```

Example 1:

Tekst dynamiczny

```
='Region - ' & if(StateName() = '$', 'Default', StateName())
```

Example 2:

Dynamiczne kolory

```
if(StateName() = 'Group 1', rgb(152, 171, 206),
      if(StateName() = 'Group 2', rgb(187, 200, 179),
            rgb(210, 210, 210)
      )
)
```

5.26 Funkcje tabeli

Funkcje tabeli zwracają informacje na temat aktualnie odczytywanej tabeli danych. Jeśli nie określono nazwy tabeli i funkcja jest używana w instrukcji **LOAD**, przyjmowana jest bieżąca tabela.

Wszystkie funkcje mogą być użyte w skrypcie ładowania danych i tylko funkcji **NoOfRows** można użyć w wyrażeniu wykresu.

Przegląd funkcji tabeli

Niektóre z funkcji są po podsumowaniu opisane bardziej szczegółowo. W przypadku tych funkcji można kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

FieldName

Funkcja skryptu **FieldName** zwraca nazwę pola o określonym numerze w poprzednio załadowanej tabeli. Jeśli funkcja jest używana w instrukcji **LOAD**, nie może odwoływać się do tabeli, która jest już ładowana.

```
FieldName (field_number ,table_name)
```

FieldNumber

Funkcja skryptu **FieldNumber** zwraca numer określonego pola w poprzednio załadowanej tabeli. Jeśli funkcja jest używana w instrukcji **LOAD**, nie może odwoływać się do tabeli, która jest już ładowana.

```
FieldNumber (field_name ,table_name)
```

NoOfFields

Funkcja skryptu **NoOfFields** zwraca liczbę pól w poprzednio załadowanej tabeli. Jeśli funkcja jest używana w instrukcji **LOAD**, nie może odwoływać się do tabeli, która jest już ładowana.

```
NoOfFields (table_name)
```

NoOfRows

Funkcja **NoOfRows** zwraca liczbę wierszy (rekordów) w poprzednio załadowanej tabeli. Jeśli funkcja jest używana w instrukcji **LOAD**, nie może odwoływać się do tabeli, która jest już ładowana.

```
NoOfRows (table_name)
```

NoOfTables

Ta funkcja skryptu zwraca liczbę poprzednio załadowanych tabel.

```
NoOfTables()
```

TableName

Ta funkcja skryptu zwraca nazwę tabeli o określonym numerze.

```
TableName(table_number)
```

TableNumber

Ta funkcja skryptu zwraca numer określonej tabeli. Pierwsza tabela ma numer 0.

Jeśli table_name nie istnieje, zwracana jest wartość NULL.

```
TableNumber(table_name)
```

Przykład:

W tym przykładzie chcemy utworzyć tabelę z informacjami na temat załadowanych tabel i pól.

Najpierw ładujemy przykładowe dane. Tworzone są dwie tabele, które posłużą do przedstawienia funkcji tabeli opisanych w tej sekcji.

```
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 ;
```

Następnie przeprowadzany iterację przez załadowane tabele przy użyciu funkcji **NoOfTables** oraz przez pola w poszczególnych tabelach przy użyciu funkcji **NoOfFields** i ładujemy informacje za pomocą funkcji tabeli.

```
//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;
```

Otrzymana tabela Tables będzie wyglądać następująco:

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

Funkcja skryptu **FieldName** zwraca nazwę pola o określonym numerze w poprzednio załadowanej tabeli. Jeśli funkcja jest używana w instrukcji **LOAD**, nie może odwoływać się do tabeli, która jest już ładowana.

Składnia:

```
FieldName(field_number ,table_name)
```

Argumenty:

Argumenty

Argument	Opis
field_number	Numer pola, do którego użytkownik chce się odwołać.
table_name	Tabela zawierająca pole, do którego użytkownik chce się odwołać.

Przykład:

```
LET a = FieldName(4,'tab1');
```

FieldNumber

Funkcja skryptu **FieldNumber** zwraca numer określonego pola w poprzednio załadowanej tabeli. Jeśli funkcja jest używana w instrukcji **LOAD**, nie może odwoływać się do tabeli, która jest już ładowana.

Składnia:

```
FieldNumber(field_name ,table_name)
```

Argumenty:

Argumenty

Argument	Opis
field_name	Nazwa pola.
table_name	Nazwa tabeli zawierającej pole.

Jeśli pole field_name nie istnieje w table_name lub table_name nie istnieje, funkcja zwraca 0.

Przykład:

```
LET a = FieldNumber('Customer','tab1');
```

NoOfFields

Funkcja skryptu **NoOfFields** zwraca liczbę pól w poprzednio załadowanej tabeli. Jeśli funkcja jest używana w instrukcji **LOAD**, nie może odwoływać się do tabeli, która jest już ładowana.

Składnia:

```
NoOfFields(table_name)
```

Argumenty:

Argumenty

Argument	Opis
table_name	Nazwa tabeli.

Przykład:

```
LET a = NoOfFields('tab1');
```

NoOfRows

Funkcja **NoOfRows** zwraca liczbę wierszy (rekordów) w poprzednio załadowanej tabeli. Jeśli funkcja jest używana w instrukcji **LOAD**, nie może odwoływać się do tabeli, która jest już ładowana.

Składnia:

```
NoOfRows(table_name)
```

Argumenty:

Argumenty

Argument	Opis
table_name	Nazwa tabeli.

Przykład:

```
LET a = NoOfRows('tab1');
```

5.27 Funkcje trygonometryczne i hiperboliczne

W tej sekcji opisano funkcje dotyczące wykonywania operacji trygonometrycznych i hiperbolicznych. We wszystkich funkcjach argumenty są wyrażeniami dającymi w wyniku kąty mierzone w radianach, gdzie **x** należy interpretować jako liczbę rzeczywistą.

Wszystkie kąty są mierzone w radianach.

Wszystkie funkcje mogą być stosowane zarówno w skryptach ładowania danych, jak i wyrażeniach wykresu.

cos
Cosinus **x**. Wynik jest liczbą z zakresu od -1 do 1.

```
cos( x )
```

acos
Odwrotność cosinusa **x**. Funkcja jest zdefiniowana wyłącznie, jeśli $-1 \leq x \leq 1$. Wynik jest liczbą z zakresu od 0 do π.

```
acos( x )
```

sin
Sinus **x**. Wynik jest liczbą z zakresu od -1 do 1.

```
sin( x )
```

asin
Odwrotność sinusa **x**. Funkcja jest zdefiniowana wyłącznie, jeśli $-1 \leq x \leq 1$. Wynik jest liczbą z zakresu od $-\pi/2$ do $\pi/2$.

```
asin( x )
```

tan
Tangens **x**. Wynik jest liczbą rzeczywistą.

```
tan( x )
```

atan
Odwrotność tangensa **x**. Wynik jest liczbą z zakresu od $-\pi/2$ do $\pi/2$.

```
atan( x )
```

atan2

Dwuwymiarowe uogólnienie funkcji odwrotności tangensa. Zwraca kąt między wartością pierwotną a punktem określonym współrzędnymi **x** i **y**. Wynik jest liczbą z zakresu od -π do +π.

```
atan2( y,x )
```

cosh

Cosinus hiperboliczny **x**. Wynik jest dodatnią liczbą rzeczywistą.

```
cosh( x )
```

sinh

Sinus hiperboliczny **x**. Wynik jest liczbą rzeczywistą.

```
sinh( x )
```

tanh

Tangens hiperboliczny **x**. Wynik jest liczbą rzeczywistą.

```
tanh( x )
```

acosh

Odwrotność cosinusa hiperbolicznego **x**. Wynik jest dodatnią liczbą rzeczywistą.

```
acosh( x )
```

asinh

Odwrotność sinusa hiperbolicznego **x**. Wynik jest liczbą rzeczywistą.

```
asinh( x )
```

atanh

Odwrotność tangensa hiperbolicznego **x**. Wynik jest liczbą rzeczywistą.

```
atanh( x )
```

Przykłady:

Poniższy kod skryptu ładuje tabelę przykładową, a następnie ładuje tabelę zawierającą obliczone operacje trygonometryczne i hiperboliczne na wartościach.

```
SampleData:
LOAD * Inline
[Value
-1
0
1];

Results:
Load *,
cos(Value),
acos(Value),
sin(Value),
asin(Value),
```

```
tan(Value),
atan(Value),
atan2(Value, Value),
cosh(Value),
sinh(Value),
tanh(Value)
RESIDENT SampleData;

Drop Table SampleData;
```

6 Ograniczenie dostępu do systemu plików

Ze względów bezpieczeństwa Qlik Sense w trybie standardowym nie obsługuje ścieżek w skrypcie ładowania danych, podobnie jak funkcji ani zmiennych ujawniających informacje o systemie plików.

Ponieważ jednak ścieżki systemu plików były obsługiwane w QlikView, można jednak wyłączyć tryb standardowy i używać trybu zgodności w celu korzystania z istniejących skryptów ładowania QlikView.

Wyłączenie trybu standardowego może stworzyć zagrożenie przez ujawnienie informacji o systemie plików.

Wyłączanie trybu standardowego (page 1454)

6.1 Kwestie bezpieczeństwa związane z połączeniami do danych ODBC i OLE DB w plikach

Nawiązywanie połączeń do danych ODBC i OLE DB z wykorzystaniem sterowników plikowych powoduje ujawnienie ścieżki do połączonego pliku danych w ciągu połączenia. Ścieżka ta może być ujawniona podczas edytowania połączenia, w oknie dialogowym selekcji danych lub w określonym zapytaniu SQL. Dotyczy to zarówno trybu standardowego, jak i trybu zgodności.

Jeśli ujawnianie ścieżki do pliku danych jest niepożądane, zaleca się w miarę możliwości nawiązywanie połączenia z plikiem danych w ramach połączenia z danymi folderu.

6.2 Ograniczenia trybu standardowego

Niektóre instrukcje, zmienne i funkcje są w trybie standardowym niedostępne lub dostępne w ograniczonym zakresie. Użycie nieobsługiwanej instrukcji w skrypcie ładowania danych spowoduje błąd przy wykonywaniu skryptu ładowania. Komunikaty o błędach można znaleźć w pliku dziennika skryptu. Użycie nieobsługiwanej zmiennej lub funkcji nie jest sygnalizowane żadnym komunikatem o błędzie ani wpisem w pliku dziennika. Funkcja zwróci jedynie wartość NULL.

Podczas edytowania skryptu ładowania danych nie są wyświetlane żadne informacje o użyciu nieobsługiwanej zmiennej, instrukcji lub funkcji.

Zmienne systemowe

Zmienne systemowe

Zmienna	Tryb standardowy	Tryb zgodności	Definicja
Floppy	Nieobsługiwane	Obsługiwane	Zwraca literę napędu pierwszego znalezionego napędu dyskietek (zwykle jest to *a:*).
CD	Nieobsługiwane	Obsługiwane	Zwraca literę napędu pierwszego znalezionego napędu CD-ROM. Jeśli żaden napęd CD-ROM nie zostanie znaleziony, wówczas zostanie zwrócona litera *c:*.
QvPath	Nieobsługiwane	Obsługiwane	Zwraca ciąg przeglądania ścieżki pliku wykonywalnego Qlik Sense.
QvRoot	Nieobsługiwane	Obsługiwane	Zwraca katalog główny pliku wykonywalnego programu Qlik Sense.
QvWorkPath	Nieobsługiwane	Obsługiwane	Zwraca ciąg znaków przeglądania do bieżącej aplikacji Qlik Sense.
QvWorkRoot	Nieobsługiwane	Obsługiwane	Zwraca katalog główny bieżącej aplikacji Qlik Sense.
WinPath	Nieobsługiwane	Obsługiwane	Zwraca ciąg znaków przeglądania do systemu Windows.
WinRoot	Nieobsługiwane	Obsługiwane	Zwraca katalog główny systemu Windows.

Zmienna	Tryb standardowy	Tryb zgodności	Definicja
$(include=...)	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką lub systemu plików	Zmienna **Include/Must_Include** określa plik, który zawiera tekst, jaki powinien zostać umieszczony w skrypcie i oceniony jako kod skryptu. Nie służy do dodawania danych. Można przechowywać części kodu skryptu w oddzielnym pliku tekstowym i używać ich wielokrotnie w wielu aplikacjach. Jest to zmienna definiowana przez użytkownika.

Zwykłe instrukcje skryptu

Zwykłe instrukcje skryptu

Instrukcja	Tryb standardowy	Tryb zgodności	Definicja
Binary	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką lub systemu plików	Instrukcja **binary** służy do ładowania danych z innej aplikacji.
Connect	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką lub systemu plików	Instrukcja **CONNECT** służy do określania dostępu aplikacji Qlik Sense do ogólnej bazy danych przy użyciu interfejsu OLE DB/ODBC. W przypadku ODBC źródło danych najpierw należy określić za pomocą narzędzia administracyjnego ODBC.

Instrukcja	Tryb standardowy	Tryb zgodności	Definicja
Directory	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką lub systemu plików	Instrukcja **Directory** określa, w którym katalogu należy szukać plików danych w kolejnych instrukcjach **LOAD**, do momentu wydania nowej instrukcji **Directory**.
Execute	Nieobsługiwane	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką lub systemu plików	Instrukcja **Execute** służy do uruchamiania innych programów w czasie, gdy aplikacja Qlik Sense ładuje dane. Na przykład w celu wykonania niezbędnych przekształceń.
LOAD from ...	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką lub systemu plików	Instrukcja **LOAD** ładuje pola z pliku, z danych zdefiniowanych w skrypcie, z wcześniej załadowanej tabeli, ze strony internetowej, z wyniku późniejszej instrukcji **SELECT** lub przez automatyczne wygenerowanie danych.
Store into ...	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką lub systemu plików	Instrukcja **Store** tworzy plik QVD, CSV lub text.

Instrukcje sterowania skryptem

Instrukcje sterowania skryptem

Instrukcja	Tryb standardowy	Tryb zgodności	Definicja
For each... filelist mask/dirlist mask	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką Zwracane dane wyjściowe: połączenie z biblioteką	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką lub systemu plików Zwracane dane wyjściowe: Połączenie z biblioteką lub ścieżka w systemie plików w zależności od danych wejściowych	Konstrukcja filelist mask tworzy rozdzieloną przecinkami listę wszystkich plików w bieżącym katalogu zgodnych z **filelist mask**. Konstrukcja **dirlist mask** tworzy rozdzieloną przecinkami listę wszystkich katalogów w bieżącym katalogu zgodnych z maską nazwy katalogu.

Funkcje pliku

Funkcje pliku

Funkcja	Tryb standardowy	Tryb zgodności	Definicja
Attribute()	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką lub systemu plików	Zwraca wartość metaznaczników różnych plików multimedialnych w postaci tekstu.
ConnectString()	Zwracane dane wyjściowe: Nazwa połączenia biblioteki	Nazwa połączenia biblioteki lub samo połączenie, zależnie od danych wejściowych	Zwraca aktywne parametry połączenia dla połączeń ODBC lub OLE DB.
FileDir()	Zwracane dane wyjściowe: połączenie z biblioteką	Zwracane dane wyjściowe: Połączenie z biblioteką lub ścieżka w systemie plików w zależności od danych wejściowych	Funkcja **FileDir** zwraca ciąg znaków zawierający ścieżkę do katalogu aktualnie odczytywanego pliku tabeli.

Funkcja	Tryb standardowy	Tryb zgodności	Definicja
FilePath()	Zwracane dane wyjściowe: połączenie z biblioteką	Zwracane dane wyjściowe: Połączenie z biblioteką lub ścieżka w systemie plików w zależności od danych wejściowych	Funkcja **FilePath** zwraca ciąg znaków zawierający pełną ścieżkę do aktualnie odczytywanego pliku tabeli.
FileSize()	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką lub systemu plików	Funkcja **FileSize** zwraca liczbę całkowitą zawierającą rozmiar (w bajtach) pliku filename albo, jeśli nie określono parametru filename, aktualnie odczytywanego pliku tabeli.
FileTime()	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką lub systemu plików	Funkcja **FileTime** zwraca znacznik czasu daty i godziny w UTC ostatniej modyfikacji pliku filename. Jeśli nie określono parametru filename, funkcja odwoła się do aktualnie odczytywanego pliku tabeli.
GetFolderPath()	Nieobsługiwane	Zwracane dane wyjściowe: Ścieżka bezwzględna	Funkcja **GetFolderPath** zwraca wartość funkcji Microsoft Windows *SHGetFolderPath*. Ta funkcja przyjmuje na wejściu nazwę folderu Microsoft Windows i zwraca pełną ścieżkę do tego folderu.

Funkcja	Tryb standardowy	Tryb zgodności	Definicja
QvdCreateTime()	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką lub systemu plików	Ta funkcja skryptu zwraca znacznik czasu z nagłówkiem XML z ewentualnego pliku QVD. W przeciwnym wypadku zwraca wartość NULL. W znaczniku czasu jest określany czas UTC.
QvdFieldName()	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką lub systemu plików	Funkcja skryptu zwraca nazwę numeru pola (**fieldno**) w pliku QVD. Jeśli nie istnieje, zwracana jest wartość NULL.
QvdNoOfFields()	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką lub systemu plików	Ta funkcja skryptu zwraca liczbę pól w pliku QVD.
QvdNoOfRecords()	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką lub systemu plików	Ta funkcja skryptu zwraca liczbę rekordów aktualnie istniejących w pliku QVD.
QvdTableName()	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką	Obsługiwane dane wejściowe: Ścieżka używająca połączenia z biblioteką lub systemu plików	Ta funkcja skryptu zwraca nazwę tabeli zapisaną w pliku QVD.

Funkcje systemowe

Funkcje systemowe

Funkcja	Tryb standardowy	Tryb zgodności	Definicja
DocumentPath()	Nieobsługiwane	Zwracane dane wyjściowe: Ścieżka bezwzględna	Ta funkcja zwraca ciąg znaków zawierający pełną ścieżkę do bieżącej aplikacji Qlik Sense.

Funkcja	Tryb standardowy	Tryb zgodności	Definicja
GetRegistryString()	Nieobsługiwane	Obsługiwane	Zwraca wartość nazwanego klucza rejestru o podanej ścieżce rejestru. Funkcji tej można używać zarówno w wykresach, jak i w skryptach.

6.3 Wyłączanie trybu standardowego

Wyłączenie trybu standardowego, czyli innymi słowy włączenie trybu zgodności, umożliwia korzystanie z istniejących skryptów ładowania programu QlikView zawierających odniesienia do względnych i bezwzględnych ścieżek plików i połączeń bibliotek.

Wyłączenie trybu standardowego może stworzyć zagrożenie przez ujawnienie informacji o systemie plików.

Qlik Sense

W programie Qlik Sense wyłączenie trybu standardowego umożliwia właściwość **Tryb standardowy** w konsoli QMC.

Qlik Sense Desktop

W przypadku programu Qlik Sense Desktop wybór trybu standardowego lub trybu zgodności wymaga modyfikacji pliku *Settings.ini*.

Jeśli program Qlik Sense Desktop został zainstalowany przy użyciu domyślnej lokalizacji instalacji, wówczas plik *Settings.ini* znajduje się w ścieżce *C:\Users\{user}\Documents\Qlik\Sense\Settings.ini*. Jeśli program Qlik Sense Desktop został zainstalowany w folderze wybranym przez użytkownika, wówczas plik *Settings.ini* znajduje się w folderze *Engine* w ścieżce instalacji.

Wykonaj następujące czynności:

1. Otwórz plik *Settings.ini* w edytorze tekstu.
2. Zmień parametr *StandardReload=1* na *StandardReload=0*.
3. Zapisz plik i uruchom program Qlik Sense Desktop.

Program Qlik Sense Desktop zostanie uruchomiony w trybie zgodności.

Ustawienia

Możliwe wartości parametru StandardReload:

- 1 (tryb standardowy)
- 0 (tryb zgodności)

6 Skrypty na poziomie wykresu

Podczas modyfikowania danych wykresu używa się podzestawu skryptu Qlik Sense, który składa się z pewnej liczby instrukcji. Instrukcje dzielą się na zwykłe instrukcje skryptu oraz instrukcje sterowania skryptem. Niektóre instrukcje można poprzedzać prefiksami.

Zwykłe instrukcje służą zazwyczaj do wykonywania operacji na danych. Każda taka instrukcja może obejmować w skrypcie dowolną liczbę wierszy i musi zawsze być zakończona średnikiem, czyli znakiem „;”.

Instrukcje sterowania służą zazwyczaj do sterowania przepływem wykonania skryptu. Każda klauzula instrukcji sterowania musi mieścić się w jednym wierszu skryptu i może być zakończona albo średnikiem, albo znakiem końca linii.

Prefiksy można stosować z obsługującymi je instrukcjami zwykłymi, ale nigdy z instrukcjami sterowania.

Słowa kluczowe w skrypcie mogą być wpisywane z użyciem dowolnych kombinacji małych i wielkich liter. Wielkość liter jest natomiast uwzględniana w nazwach pól i zmiennych używanych w instrukcjach.

W tej sekcji można znaleźć alfabetyczną listę wszystkich instrukcji skryptu, instrukcji sterowania i prefiksów dostępnych w podzestawie skryptu używanym podczas modyfikowania danych wykresu.

6.4 Instrukcje sterowania

Podczas modyfikowania danych wykresu używa się podzestawu skryptu Qlik Sense, który składa się z pewnej liczby instrukcji. Instrukcje dzielą się na zwykłe instrukcje skryptu oraz instrukcje sterowania skryptem.

Instrukcje sterowania służą zazwyczaj do sterowania przepływem wykonania skryptu. Każda klauzula instrukcji sterowania musi mieścić się w jednym wierszu skryptu i może być zakończona albo średnikiem, albo znakiem końca linii.

Prefiksy nigdy nie są stosowane do instrukcji sterowania.

Słowa kluczowe w skrypcie mogą być wpisywane z użyciem dowolnych kombinacji małych i wielkich liter.

Przegląd instrukcji sterowania modyfikatora wykresu

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

Call

Instrukcja sterowania **call** wywołuje procedurę zdefiniowaną we wcześniejszej instrukcji **sub**.

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

Do..loop

Instrukcja sterowania **do..loop** to rodzaj iteracji skryptu, który wykonuje co najmniej jedną instrukcję aż do momentu spełnienia warunku logicznego.

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

End

Słowo kluczowe skryptu **End** służy do zamykania klauzul **If**, **Sub** i **Switch**.

Exit

Słowo kluczowe skryptu **Exit** jest częścią instrukcji **Exit Script**, ale może także zostać użyte do zamykania klauzul **Do**, **For** lub **Sub**.

Exit script

Instrukcja ta zatrzymuje wykonanie skryptu. Można ją wstawić w dowolnym miejscu skryptu.

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

For..next

Instrukcja sterowania **for..next** to rodzaj iteracji skryptu z licznikiem. Dla każdej wartości zmiennej licznika mieszczącej się w określonym limicie wykonane zostaną instrukcje wewnątrz pętli między wartościami **for** i **next**.

```
For..next  counter = expr1 to expr2 [ stepexpr3 ]
[statements]
[exit for [ ( when | unless ) condition ]
[statements]
Next [counter]
```

For each ..next

Instrukcja sterowania **for each..next** to rodzaj iteracji skryptu służący do wykonania co najmniej jednej instrukcji dla każdej wartości na liście rozdzielonej przecinkami. Dla każdej wartości na liście wykonane zostaną instrukcje wewnątrz pętli między argumentami **for** i **next**.

```
For each..next var in list
[statements]
[exit for [ ( when | unless ) condition ]
[statements]
next [var]
```

If..then

Instrukcja sterowania **if..then** jest rodzajem selekcji skryptu, który wymusza wykonanie skryptu według różnych ścieżek w zależności od co najmniej jednego warunku logicznego.

*Instrukcja **if..then** jest instrukcją sterowania i jako taka kończy się średnikiem lub symbolem końca linii, a żadna z jej czterech możliwych klauzul (**if..then**,**elseif..then**, **else** i **end if**) nie może wykraczać poza granicę linii.*

```
If..then..elseif..else..end if condition then
  [ statements ]
{ elseif condition then
  [ statements ] }
[ else
  [ statements ] ]
end if
```

Next

Słowo kluczowe skryptu **Next** służy do zamykania pętli **For**.

Sub

Instrukcja sterowania **sub..end sub** definiuje podprogram, który można wywołać z instrukcji **call**.

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

Switch

Instrukcja sterowania **switch** jest rodzajem selekcji skryptu, który wymusza wykonanie skryptu według różnych ścieżek w zależności od wartości wyrażenia.

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

To

Słowo kluczowe **To** jest używane w wielu instrukcjach skryptu.

Call

Instrukcja sterowania **call** wywołuje procedurę zdefiniowaną we wcześniejszej instrukcji **sub**.

Składnia:

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

Argumenty:

Argumenty

Argument	Opis
name	Nazwa podprogramu.
paramlist	Do podprogramu przekazana zostanie rozdzielana przecinkami lista rzeczywistych parametrów. Każdy z elementów listy może być nazwą pola, zmienną lub dowolnym wyrażeniem.

Podprogram wywoływany instrukcją **call** musi być zdefiniowany w instrukcji **sub** napotkanej na wcześniejszym etapie wykonywania skryptu.

Parametry są kopiowane do podprogramu, a jeśli parametrem instrukcji **call** jest zmienna, a nie wyrażenie, to po wyjściu z podprogramu parametry są ponownie kopiowane na zewnątrz.

Ograniczenia:

- Jako instrukcja sterowania instrukcja **call** kończy się na średniku lub znaku nowego wiersza, nie może zatem obejmować wielu wierszy.
- Podczas definiowania podprogramu za pomocą `Sub..end sub` wewnątrz instrukcji sterującej, na przykład `if..then`, można wywołać podprogram tylko z tej samej instrukcji sterowania.

Do..loop

Instrukcja sterowania **do..loop** to rodzaj iteracji skryptu, który wykonuje co najmniej jedną instrukcję aż do momentu spełnienia warunku logicznego.

Składnia:

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

*Instrukcja **do..loop** jest instrukcją sterowania i jako taka kończy się średnikiem lub symbolem końca linii, żadna z jej trzech możliwych klauzul (**do**,**exit do** i **loop**) nie może zatem wykraczać poza granicę linii.*

Argumenty:

Argumenty

Argument	Opis
condition	Wyrażenie logiczne, którego ocena zwraca True lub False.
statements	Dowolna grupa zawierająca co najmniej jedną instrukcję skryptu Qlik Sense.
while / until	Klauzula warunkowa **while** lub **until** może pojawiać się tylko raz w instrukcji **do..loop**, tj. po instrukcji **do** lub po instrukcji **loop**. Każdy warunek jest interpretowany tylko przy pierwszym napotkaniu, ale jest oceniany każdorazowo przy napotkaniu w pętli.
exit do	Jeśli wewnątrz pętli napotkana zostanie klauzula **exit do**, wykonanie skryptu zostanie przeniesione do pierwszej instrukcji po klauzuli **loop** wskazującej na koniec pętli. Klauzula **exit do** może zostać ustawiona jako warunkowa przez opcjonalne użycie sufiksu **when** lub **unless**.

End

Słowo kluczowe skryptu **End** służy do zamykania klauzul **If**, **Sub** i **Switch**.

Exit

Słowo kluczowe skryptu **Exit** jest częścią instrukcji **Exit Script**, ale może także zostać użyte do zamykania klauzul **Do**, **For** lub **Sub**.

Exit script

Instrukcja ta zatrzymuje wykonanie skryptu. Można ją wstawić w dowolnym miejscu skryptu.

Składnia:

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

Jako instrukcja sterowania instrukcja **exit script** kończy się na średniku lub znaku nowego wiersza, nie może zatem obejmować wielu wierszy.

Argumenty:

Argumenty

Argument	Opis
condition	Wyrażenie logiczne, którego ocena zwraca True lub False.
when / unless	Instrukcja **exit script** może zostać ustawiona jako warunkowa przez opcjonalne użycie klauzuli **when** lub **unless**.

Przykłady:

```
//Exit script
Exit Script;

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

For..next

Instrukcja sterowania **for..next** to rodzaj iteracji skryptu z licznikiem. Dla każdej wartości zmiennej licznika mieszczącej się w określonym limicie wykonane zostaną instrukcje wewnątrz pętli między wartościami **for** i **next**.

Składnia:

```
For counter = expr1 to expr2 [ step expr3 ]
[statements]
[exit for [ ( when | unless ) condition ]
[statements]
Next [counter]
```

Wyrażenia *expr1*, *expr2* i *expr3* są oceniane tylko przy pierwszym wejściu do pętli. Wartość zmiennej counter można zmienić przy użyciu instrukcji wewnątrz pętli, ale nie jest to zalecana praktyka programowania.

Jeśli wewnątrz pętli napotkana zostanie klauzula **exit for**, wykonanie skryptu zostanie przeniesione do pierwszej instrukcji po klauzuli **next** wskazującej na koniec pętli. Klauzula **exit for** może zostać ustawiona jako warunkowa przez opcjonalne użycie sufiksu **when** lub **unless**.

*Instrukcja **for..next** jest instrukcją sterowania i jako taka kończy się średnikiem lub symbolem końca linii, żadna z jej trzech możliwych klauzul (**for..to..step**,**exit for** i **next**) nie może zatem wykraczać poza granicę linii.*

Argumenty:

Argumenty

Argument	Opis
counter	Nazwa zmiennej. Jeśli argument *counter* określono po argumencie **next**, musi on mieć taką samą nazwę zmiennej jak nazwa po **for**.
expr1	Wyrażenie określające pierwszą wartość zmiennej *counter*, dla której pętla powinna zostać wykonana.
expr2	Wyrażenie określające ostatnią wartość zmiennej *counter*, dla której pętla powinna zostać wykonana.
expr3	Wyrażenie określające wartość wskazującą na przyrost zmiennej *counter* podczas każdego wykonania pętli.
condition	wyrażenie logiczne, którego ocena zwraca True lub False.
statements	Dowolna grupa zawierająca co najmniej jedną instrukcję skryptu Qlik Sense.

For each..next

Instrukcja sterowania **for each..next** to rodzaj iteracji skryptu służący do wykonania co najmniej jednej instrukcji dla każdej wartości na liście rozdzielonej przecinkami. Dla każdej wartości na liście wykonane zostaną instrukcje wewnątrz pętli między argumentami **for** i **next**.

Składnia:
Dzięki specjalnej składni możliwe jest generowanie list z nazwami plików i katalogów w katalogu bieżącym.

```
for each var in list
[statements]
[exit for [ ( when | unless ) condition ]
[statements]
next [var]
```

Argumenty:

Argumenty

Argument	Opis
var	Nazwa zmiennej skryptu pobierającej nową wartość z listy dla każdego wykonania pętli. Jeśli argument **var** określono po argumencie **next**, musi on mieć taką samą nazwę zmiennej jak nazwa po **for each**.

Wartość zmiennej **var** można zmienić przy użyciu instrukcji wewnątrz pętli, ale nie jest to zalecana praktyka programowania.

Jeśli wewnątrz pętli napotkana zostanie klauzula **exit for**, wykonanie skryptu zostanie przeniesione do pierwszej instrukcji po klauzuli **next** wskazującej na koniec pętli. Klauzula **exit for** może zostać ustawiona jako warunkowa przez opcjonalne użycie sufiksu **when** lub **unless**.

*Instrukcja **for each..next** jest instrukcją sterowania i jako taka kończy się średnikiem lub symbolem końca linii, żadna z jej trzech możliwych klauzul (**for each**,**exit for** i **next**) nie może zatem wykraczać poza granicę linii.*

Składnia:

```
list := item { , item }
item := constant | (expression) | filelist mask | dirlist mask |
fieldvaluelist mask
```

Argumenty

Argument	Opis
constant	Dowolna liczba lub ciąg znaków. Należy pamiętać, że ciąg znaków wpisany bezpośrednio w skrypt musi być ujęty w pojedyncze cudzysłowy. Ciąg znaków bez pojedynczych cudzysłowów zostanie zinterpretowany jako zmienna i będzie użyta wartość zmiennej. Liczb nie trzeba ujmować w pojedyncze cudzysłowy.
expression	Dowolne wyrażenie.
mask	Maska nazwy pliku lub folderu, która może zawierać dowolne znaki dozwolone w nazwie pliku, a także standardowe symbole wieloznaczne, na przykład ***** i **?**. Można używać bezwzględnych ścieżek do plików lub ścieżek lib://.
condition	Wyrażenie logiczne, którego ocena zwraca True lub False.
statements	Dowolna grupa zawierająca co najmniej jedną instrukcję skryptu Qlik Sense.

Argument	Opis
filelist mask	Składnia tworzy rozdzieloną przecinkami listę wszystkich plików w bieżącym katalogu zgodnym z maską nazwy pliku. *Ten argument obsługuje tylko połączenia bibliotek w trybie standardowym.*
dirlist mask	Składnia tworzy rozdzieloną przecinkami listę wszystkich folderów w bieżącym folderze zgodnym z maską nazwy folderu. *Ten argument obsługuje tylko połączenia bibliotek w trybie standardowym.*
fieldvaluelist mask	Ta składnia jest iterowana przez wartości pola już załadowane do aplikacji Qlik Sense.

Połączenia z Qlik Łączniki do dostawcy magazynowania w sieci Web i inne połączenia DataFiles nie obsługują masek filtrów, które używają symboli wieloznacznych (i ?).*

Example 1: Ładowanie listy plików

```
// 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: Tworzenie listy plików na dysku

W tym przykładzie ładowana jest lista wszystkich plików powiązanych z aplikacją Qlik Sense w folderze.

```
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: Iteracja przez wartości pola

W tym przykładzie iteracja przeprowadzana jest przez listę załadowanych wartości FIELD i generowane jest nowe pole, NEWFIELD. Dla każdej wartości FIELD utworzone zostaną dwa rekordy NEWFIELD.

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

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

Wynikowa tabela wygląda następująco:

Example table

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

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

Instrukcja sterowania **if..then** jest rodzajem selekcji skryptu, który wymusza wykonanie skryptu według różnych ścieżek w zależności od co najmniej jednego warunku logicznego.

Instrukcje sterowania służą zazwyczaj do sterowania przepływem wykonania skryptu. W wyrażeniu wykresu użyj funkcji warunkowej **if**.

Składnia:

```
If  condition then
  [ statements ]
{ elseif condition then
  [ statements ] }
[ else
  [ statements ] ]
end if
```

Instrukcja **if..then** jest instrukcją sterowania i jako taka kończy się średnikiem lub symbolem końca linii, a żadna z jej czterech możliwych klauzul (**if..then**,**elseif..then**, **else** i **end if**) nie może wykraczać poza granicę linii.

Argumenty:

Argumenty

Argument	Opis
condition	Wyrażenie logiczne dające w wyniku wartości True lub False.
statements	Dowolna grupa zawierająca co najmniej jedną instrukcję skryptu Qlik Sense.

Example 1:

```
if a=1 then

      LOAD * from abc.csv;
      SQL SELECT e, f, g from tab1;

end if
```

Example 2:

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

Example 3:

```
if x>0 then

      LOAD * from pos.csv;

elseif x<0 then

      LOAD * from neg.csv;

else

      LOAD * from zero.txt;

end if
```

Next

Słowo kluczowe skryptu **Next** służy do zamykania pętli **For**.

Sub..end sub

Instrukcja sterowania **sub..end sub** definiuje podprogram, który można wywołać z instrukcji **call**.

Składnia:

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

Argumenty są kopiowane do podprogramu, a jeśli odpowiadające parametry rzeczywiste w instrukcji **call** są nazwami zmiennych, wówczas są kopiowane ponownie na zewnątrz po wyjściu z podprogramu.

Jeśli podprogram zawiera więcej parametrów formalnych niż rzeczywistych przekazywanych przez instrukcję **call**, wówczas parametry dodatkowe zostaną zainicjowane na NULL i możliwe będzie ich użycie jako zmiennych lokalnych w podprogramie.

Argumenty:

Argumenty

Argument	Opis
name	Nazwa podprogramu.
paramlist	Rozdzielana przecinkami lista nazw zmiennych dla parametrów formalnych podprogramu. Te zmienne mogą być używane jak dowolna zmienna w podprogramie.
statements	Dowolna grupa zawierająca co najmniej jedną instrukcję skryptu Qlik Sense.

Ograniczenia:

- Instrukcja **sub** jest instrukcją sterowania i jako taka kończy się średnikiem lub symbolem końca linii – żadna z jej dwóch klauzul (**sub** i **end sub**) nie może zatem wykraczać poza granicę linii.
- Podczas definiowania podprogramu za pomocą `Sub..end sub` wewnątrz instrukcji sterującej, na przykład `if..then`, można wywołać podprogram tylko z tej samej instrukcji sterowania.

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: – transfer parametru

```
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)
```

Wynik powyższej składni będzie następujący: lokalnie, wewnątrz podprogramu, A zostanie zainicjowane na 1, B zostanie zainicjowane na 4, a C zostanie zainicjowane na NULL.

Po wyjściu z podprogramu zmienna globalna A otrzyma 2 jako wartość (kopiowaną z powrotem z podprogramu). Drugi parametr rzeczywisty „(X+1)*2” nie zostanie skopiowany z powrotem, ponieważ nie jest zmienną. Ostatecznie wywołanie podprogramu nie wpływa na zmienną globalną C.

Switch..case..default..end switch

Instrukcja sterowania **switch** jest rodzajem selekcji skryptu, który wymusza wykonanie skryptu według różnych ścieżek w zależności od wartości wyrażenia.

Składnia:

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

*Instrukcja **switch** jest instrukcją sterowania i jako taka kończy się średnikiem lub symbolem końca linii, a żadna z jej czterech możliwych klauzul (**switch**,**case**, **default** i **end switch**) nie może wykraczać poza granicę linii.*

Argumenty:

Argumenty

Argument	Opis
expression	Dowolne wyrażenie.
valuelist	Rozdzielana przecinkami listą wartości, z którymi będzie porównywana wartość wyrażenia. Wykonywanie skryptu będzie kontynuowane z instrukcjami w pierwszej napotkanej grupie, w której wartość z listy valuelist jest równa wartości expression. Każda wartość z listy valuelist może być dowolnym wyrażeniem. Jeśli w żadnej klauzuli **case** nie zostanie znalezione żadne dopasowanie, wówczas zostaną wykonane instrukcje z klauzuli **default**, jeśli zostały określone.
statements	Dowolna grupa zawierająca co najmniej jedną instrukcję skryptu Qlik Sense.

Przykład:

```
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

Słowo kluczowe **To** jest używane w wielu instrukcjach skryptu.

6.5 Prefiksy

Prefiksy można stosować z obsługującymi je instrukcjami zwykłymi, ale nigdy z instrukcjami sterowania.

Słowa kluczowe w skrypcie mogą być wpisywane z użyciem dowolnych kombinacji małych i wielkich liter. Wielkość liter jest natomiast uwzględniana w nazwach pól i zmiennych używanych w instrukcjach.

Przegląd prefiksów modyfikatorów wykresów

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

Add

Do dowolnej instrukcji **LOAD** lub **SELECT** w skrypcie można dodać prefiks **Add**, aby określić, że powinna ona dodawać rekordy do innej tabeli. Określa on również, że ta instrukcja powinna być uruchamiana podczas częściowego ładowania. Prefiksu **Add** można też użyć w instrukcji **Map**.

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

Replace

Prefiks **Replace** można dodać do dowolnej instrukcji **LOAD** lub **SELECT** w skrypcie, aby określić, że ładowana tabela powinna zastąpić inną tabelę. Określa on również, że ta instrukcja powinna być uruchamiana podczas częściowego ładowania. Prefiksu **Replace** można też użyć w instrukcji **Map**.

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

Add

W kontekście modyfikacji wykresu prefiks **Add** jest używany z **LOAD** do dołączania wartości do tabeli *HC1* reprezentującej hiperkostkę obliczoną przez Qlik associative engine. Możesz określić jedną lub kilka kolumn. Brakujące wartości są automatycznie uzupełniane przez Qlik associative engine.

Składnia:

```
Add loadstatement
```

Przykład:

Ten przykład dodaje dwa wiersze do kolumn *Dates* i *Sales* z wbudowanej instrukcji

```
Add Load
x as Dates,
y as Sales
Inline
[
Dates,Sales
2001/09/1,1000
2001/09/10,-300
]
```

Replace

W kontekście modyfikacji wykresu prefiks **Replace** zmienia wszystkie wartości tabeli *HC1* na obliczoną wartość zdefiniowaną przez skrypt.

Składnia:

```
Replace loadstatement
```

Przykład:

Ten przykład zastępuje wszystkie wartości w kolumnie `z` sumą `x` i `y`.

```
Replace Load
x+y as z
Resident HC1;
```

6.6 Zwykłe instrukcje

Zwykłe instrukcje służą zazwyczaj do wykonywania operacji na danych. Każda taka instrukcja może obejmować w skrypcie dowolną liczbę wierszy i musi zawsze być zakończona średnikiem, czyli znakiem „;”.

Słowa kluczowe w skrypcie mogą być wpisywane z użyciem dowolnych kombinacji małych i wielkich liter. Wielkość liter jest natomiast uwzględniana w nazwach pól i zmiennych używanych w instrukcjach.

Przegląd zwykłych instrukcji z modyfikatorami wykresów

Po podsumowaniu każda funkcja jest opisana szczegółowo. Można też kliknąć nazwę funkcji w opisie składni, aby natychmiast wyświetlić szczegółowe informacje o tej funkcji.

LOAD

W kontekście modyfikacji wykresu instrukcja **LOAD** ładuje do hiperkostki dodatkowe dane z danych zdefiniowanych w skrypcie lub z wcześniej załadowanej tabeli. Możliwe jest również ładowanie danych z połączeń analitycznych.

*Instrukcja **LOAD** musi mieć prefiks **Replace** lub **Add** – w przeciwnym razie zostanie odrzucona.*

```
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

Instrukcja **let**, uzupełniająca instrukcję **set**, służy do określania zmiennych skryptu. Instrukcja **let** w przeciwieństwie do instrukcji **set** ocenia wyrażenie po prawej stronie znaku „=" w czasie wykonywania skryptu, zanim zostanie przypisane do zmiennej.

```
Let variablename=expression
```

Set

Instrukcja **set** jest używana do określania zmiennych skryptu. Mogą one służyć do zastępowania ciągów znaków, ścieżek, dysków itp.

```
Set variablename=string
```

Put

Instrukcja **Put** służy do ustawienia pewnej wartości liczbowej w hiperkostce.

HCValue

Instrukcja **HCValue** służy do pobierania wartości w wierszu określonej kolumny.

Load

W kontekście modyfikacji wykresu instrukcja **LOAD** ładuje do hiperkostki dodatkowe dane z danych zdefiniowanych w skrypcie lub z wcześniej załadowanej tabeli. Możliwe jest również ładowanie danych z połączeń analitycznych.

> *Instrukcja **LOAD** musi mieć prefiks **Replace** lub **Add** – w przeciwnym razie zostanie odrzucona.*

Składnia:

```
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 ]
```

Argumenty:

Argumenty

Argument	Opis
fieldlist	*fieldlist ::= (* | field {, * | field })* Lista ładowanych pól. Użycie znaku * jako listy pól oznacza wszystkie pola w tabeli. *field ::= (fieldref | expression)* [**as** *aliasname]* Definicja pola musi zawsze zawierać literał, odniesienie do istniejącego pola lub wyrażenie. *fieldref ::= (fieldname |@fieldnumber |@startpos:endpos [**I** | **U** | **R** | **B** | **T**])* *fieldname* jest tekstem identycznym z nazwą pola w tabeli. Jeśli nazwa pola zawiera np. spacje, musi być ujęta w proste podwójne cudzysłowy lub nawiasy kwadratowe. Niekiedy nie są dostępne jawne nazwy pól. Wtedy należy użyć innego zapisu: *@fieldnumber* reprezentuje numer pola w rozdzielanym pliku tabeli. Musi to być dodatnia liczba całkowita poprzedzona znakiem „@”. Numeracja zawsze zaczyna się od 1, a kończy na liczbie pól. *@startpos*:*endpos* reprezentuje pozycję początkową i końcową pola w pliku z rekordami o stałej długości. Obie pozycje muszą być dodatnimi liczbami całkowitymi. Liczby muszą być poprzedzone znakiem „@” i rozdzielone dwukropkiem. Numeracja zawsze zaczyna się od 1, a kończy na liczbie pozycji. W ostatnim polu **n** jest używane jako pozycja końcowa. • Jeśli bezpośrednio po zapisie *@startpos*:*endpos* następuje znak **I** lub **U**, odczytane bajty będą interpretowane jako dane binarne zawierające liczby całkowite, odpowiednio ze znakiem (**I**) lub bez znaku (**U**) (kolejność bajtów Intel). Liczba odczytanych pozycji musi wynosić 1, 2 lub 4. • Jeśli bezpośrednio po zapisie *@startpos*:*endpos* następuje znak **R**, odczytane bajty będą interpretowane jako dane binarne zawierające liczby rzeczywiste (32-bitowe lub 64-bitowe wartości zmiennopozycyjne IEEE). Liczba odczytanych pozycji musi wynosić 4 lub 8. • Jeśli bezpośrednio po zapisie *@startpos*:*endpos* następuje znak **B**, odczytane bajty będą interpretowane jako wartości BCD (Binary Coded Decimal) według standardu COMP-3. Można podać dowolną liczbę bajtów. *expression* może być funkcją liczbową lub znakową operującą na polach tej samej tabeli. Więcej informacji na ten temat zawiera sekcja na temat składni wyrażeń. Klauzula **as** służy do przypisania polu nowej nazwy.

Argument	Opis
inline	Klauzula **inline** wskazuje dane wpisywane bezpośrednio w skrypcie, a nie ładowane z pliku. *data ::= [text]* Dane wprowadzane w ramach klauzuli **inline** muszą być ujęte w podwójne cudzysłowy lub w nawiasy kwadratowe. Tekst umieszczony między tymi znakami zostanie zinterpretowany tak samo, jak zawartość pliku. Oznacza to między innymi, że w tekście klauzuli **inline** należy wstawiać nowe wiersze w tych samych miejscach, w których występowałyby w pliku tekstowym, naciskając klawisz Enter podczas wpisywania skryptu. Liczba kolumn jest zdefiniowana w pierwszej linii. *format-spec ::= (fspec-item {, fspec-item })* Na specyfikację formatu składają się pozycje specyfikacji formatu podane w nawiasach jako lista.
resident	Klauzula **resident** umożliwia załadowanie danych z wcześniej załadowanej tabeli. *table label* to etykieta poprzedzająca instrukcję **LOAD**, która utworzyła pierwotną tabelę. Na końcu etykiety musi się znajdować dwukropek.

Argument	Opis
extension	Dane można ładować z połączeń analitycznych. Należy użyć klauzuli **extension**, aby wywołać funkcję zdefiniowaną we wtyczce rozszerzeń po stronie serwera (SSE) albo dokonać ewaluacji skryptu. Do wtyczki SSE można wysłać pojedynczą tabelę i zostanie zwrócona pojedyncza tabela danych. Jeśli wtyczka nie określa nazw pól, które są zwracane, wówczas pola otrzymają nazwy Field1, Field2 itd. `Extension pluginname.functionname( tabledescription );` • Ładowanie danych za pomocą funkcji we wtyczce SSE *tabledescription ::= (table { ,tablefield})* Jeśli pola tabeli nie zostaną określone, wówczas pola będą używane w kolejności ładowania. • Ładowanie danych poprzez ocenę skryptu we wtyczce SSE *tabledescription ::= (script, table { ,tablefield})* **Sposób obsługi typów danych w definicji pola tabeli** Typy danych są automatycznie wykrywane w połączeniach analitycznych. Jeśli dane nie zawierają żadnych wartości liczbowych, a zawierają co najmniej jeden ciąg znaków inny niż NULL, wówczas takie pole jest traktowane jako tekstowe. W każdym innym przypadku jest traktowane jako liczbowe. Typ danych można wymusić, umieszczając nazwę pola w funkcji **String()** lub **Mixed()**. • Funkcja **String()** wymusza traktowanie pola jako tekstowego. Jeśli pole jest liczbowe, wówczas wyodrębniana jest część tekstowa wartości podwójnej – nie jest wykonywane żadne przekształcenie. • Funkcja **Mixed()** wymusza traktowanie pola jako wartości podwójnej. Funkcje **String()** i **Mixed()** nie mogą być używane poza definicjami pola tabeli w klauzuli **extension**, a ponadto w definicji pola tabeli nie można używać innych funkcji Qlik Sense.
where	**where** to klauzula określająca, czy rekord ma być uwzględniony w selekcji, czy też nie. Selekcja jest uwzględniana, jeśli warunek *criterion* ma wartość True. *criterion* jest wyrażeniem logicznym.
while	**while** to klauzula określająca, czy rekord ma być wielokrotnie odczytywany. Ten sam rekord będzie odczytywany, dopóki warunek *criterion* ma wartość True. W praktyce korzystanie z klauzuli **while** zwykle wymaga również użycia funkcji **IterNo( )**. *criterion* jest wyrażeniem logicznym.

Argument	Opis
group by	Klauzula **group by** służy do określania pól, według których dane mają być agregowane (grupowane). Pola agregujące powinny być w jakiś sposób dołączone w ładowanych wyrażeniach. Poza obrębem funkcji agregacji w ładowanych wyrażeniach wolno używać wyłącznie pól agregujących. *groupbyfieldlist ::= (fieldname { ,fieldname })*
order by	Klauzula **order by** służy do sortowania rekordów tabeli rezydentnej przed ich przetworzeniem przez instrukcję **load**. Tabelę rezydentną można sortować według dowolnej liczby pól w kolejności rosnącej lub malejącej. Sortowanie jest wykonywane w pierwszej kolejności według wartości liczbowej, a w drugiej kolejności według porządku leksykograficznego z uwzględnieniem znaków diakrytycznych. Klauzuli można używać tylko wtedy, gdy źródłem danych jest tabela rezydentna. Pola porządkujące określają, według którego pola należy sortować tabelę rezydentną. Pole można wskazać nazwą lub numerem w tabeli rezydentnej (pierwsze pole ma numer 1). orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } Kolejność *sortorder* może mieć wartość *asc* (rosnąco) lub *desc* (malejąco). Jeśli nie określono *sortorder*, przyjmuje się wartość *asc*. *fieldname*, *path*, *filename* i *aliasname* to ciągi tekstowe reprezentujące odpowiednio nazwę pola, ścieżkę, nazwę pliku i alias. Jako *fieldname* można podać dowolne pole z tabeli źródłowej. Pola utworzone klauzulą as (*aliasname*) są jednak poza zakresem i nie można ich używać w tej samej instrukcji **load**.

Let

Instrukcja **let**, uzupełniająca instrukcję **set**, służy do określania zmiennych skryptu. Instrukcja **let** w przeciwieństwie do instrukcji **set** ocenia wyrażenie po prawej stronie znaku „=” w czasie wykonywania skryptu, zanim zostanie przypisane do zmiennej.

Składnia:

```
Let variablename=expression
```

Przykłady i wyniki:

Przykład	Wynik

`Set x=3+4;`
`Let y=3+4;`
`z=$(y)+1;`
	Wynikiem oceny `$(x)` będzie „3+4 ”
	Wynikiem oceny `$(y)` będzie „ 7 ”
	Wynikiem oceny `$(z)` będzie „ 8 ”
	Zwróć uwagę na różnice między instrukcjami **Set** i **Let**. Instrukcja **Set** przypisuje zmiennej ciąg „3+4”, podczas gdy instrukcja **Let** oblicza ciąg i przypisuje do zmiennej 7.

Let T=now( );	`$(T)` otrzyma wartość bieżącego czasu.

Set

Instrukcja **set** jest używana do określania zmiennych skryptu. Mogą one służyć do zastępowania ciągów znaków, ścieżek, dysków itp.

Składnia:

```
Set variablename=string
```

Example 1:

```
Set FileToUse=Data1.csv;
```

Example 2:

```
Set Constant="My string";
```

Example 3:

```
Set BudgetYear=2012;
```

Put

Instrukcja **put** służy do ustawienia pewnej wartości liczbowej w hiperkostce.

Dostęp do kolumn można uzyskać przy użyciu etykiet. Możesz również uzyskać dostęp do kolumn i wierszy według kolejności deklaracji. Aby uzyskać więcej informacji, zobacz przykłady poniżej.

Składnia:

```
put column(position)=value
```

Example 1:

Dostęp do kolumn można uzyskać przy użyciu etykiet.

Ten przykład ustawi wartość `1` na pierwszej pozycji kolumny o nazwie *Sales*.

```
Put Sales(1) = 1;
```

Example 2:

Dostęp do kolumn miar można uzyskać według kolejności deklaracji przy użyciu formatu `#hc1.measure` dla miar.

W tym przykładzie zostanie ustawiona wartość 1000 na dziesiątej pozycji ostatecznej, posortowanej hiperkostki.

```
Put #hc1.measure.2(10) = 1000;
```

Example 3:

Dostęp do wierszy wymiarów można uzyskać według kolejności deklaracji przy użyciu formatu `#hc1.dimension` dla wymiarów.

Ten przykład umieszcza wartość stałej Pi w piątym wierszu trzeciego zadeklarowanego wymiaru.

```
Put #hc1.dimension.3(5) = Pi();
```

Jeżeli nie ma takich wymiarów lub wyrażeń w wartości lub etykietach, zwracany jest błąd wskazujący, że nie znaleziono kolumny. Jeżeli indeks kolumny wykracza poza zakres, nie jest wyświetlany żaden błąd.

HCValue

Funkcja **HCValue** służy do pobierania wartości w wierszu określonej kolumny.

Składnia:

```
HCValue(column,position)
```

Example 1:

Ten przykład zwraca wartość na pierwszej pozycji kolumny z etykietą „Sales”.

```
HCValue(Sales,1)
```

Example 2:

Ten przykład zwraca wartość na dziesiątej pozycji posortowanej hiperkostki.

```
HCValue(#hc1.measure.2,10)
```

Example 3:

Ten przykład zwraca wartość w piątym wierszu w trzecim wymiarze.

```
HCValue(#hc1.dimension.3,5)
```

Jeżeli nie ma takich wymiarów lub wyrażeń w wartości lub etykietach, zwracany jest błąd wskazujący, że nie znaleziono kolumny. Jeżeli indeks kolumny wykracza poza zakres, zwracana jest wartość NULL.

7 Funkcje i instrukcje programu QlikView, które nie są obsługiwane w programie Qlik Sense

Funkcje i instrukcje, które mogą być używane w skryptach ładowania i wyrażeniach wykresu w programie QlikView, są w większości obsługiwane w programie Qlik Sense, ale istnieją pewne wyjątki opisane poniżej.

7.1 Instrukcje skryptów, które nie są obsługiwane w programie Qlik Sense

Instrukcje skryptów QlikView, które nie są obsługiwane w programie Qlik Sense

Instrukcja	Komentarze
Command	Zastąp instrukcją **SQL**.
InputField	

7.2 Funkcje nieobsługiwane w programie Qlik Sense

Na tej liście znajdują się funkcje skryptu i wykresu programu QlikView, które nie są obsługiwane w programie Qlik Sense.

- **GetCurrentField**
- **GetExtendedProperty**
- **Input**
- **InputAvg**
- **InputSum**
- **MsgBox**
- **NoOfReports**
- **ReportComment**
- **ReportId**
- **ReportName**
- **ReportNumber**

7.3 Prefiksy nieobsługiwane w programie Qlik Sense

Na tej liście znajdują się prefiksy programu QlikView, które nie są obsługiwane w programie Qlik Sense.

- **Bundle**
- **Image_Size**
- **Info**

8 Funkcje i instrukcje niezalecane w programie Qlik Sense

Funkcje i instrukcje dozwolone w skryptach ładowania i wyrażeniach wykresów w programie QlikView są w większości obsługiwane również w programie Qlik Sense, ale używanie niektórych z nich w programie Qlik Sense jest niezalecane. Istnieją również funkcje i instrukcje, które były dostępne w poprzednich wersjach programu Qlik Sense, które zostały zaniechane.

Dla zachowania kompatybilności będą one nadal działać poprawnie, ale zaleca się zaktualizowanie kodu zgodnie z rekomendacjami podanymi w tej sekcji, ponieważ w przyszłych wersjach mogą one zostać usunięte.

8.1 Instrukcje skryptu niezalecane w programie Qlik Sense

Poniższa tabela zawiera instrukcje skryptu, których używanie w programie Qlik Sense jest niezalecane.

Niezalecane instrukcje skryptu

Instrukcja	Zalecenie
Command	Zastąp instrukcją **SQL**.
CustomConnect	Zastąp instrukcją **Custom Connect**.

8.2 Parametry instrukcji skryptu niezalecane w programie Qlik Sense

Poniższa tabela zawiera parametry instrukcji skryptu, których używanie w programie Qlik Sense jest niezalecane.

Parametry instrukcji skryptu, które nie są zalecane

Instrukcja	Parametry
Buffer	Używaj **Incremental** zamiast: • **Inc** (niezalecane) • **Incr** (niezalecane)

Instrukcja	Parametry
LOAD	Następujące słowa kluczowe parametrów są generowane przez kreatory przekształcania plików QlikView. Po przeładowaniu danych instrukcje będą nadal działać prawidłowo, ale program Qlik Sense nie będzie oferować pomocy ani kreatorów do generowania instrukcji zawierających te parametry: • Bottom • Cellvalue • Col • Colmatch • Colsplit • Colxtr • Compound • Contain • Equal • Every • Expand • Filters • Intarray • Interpret • Length • Longer • Numerical • Pos • Remove • Rotate • Row • Rowcnd • Shorter • Start • Strcnd • Top • Transpose • Unwrap • XML: XMLSAX and Pattern is Path

8.3 Funkcje niezalecane w programie Qlik Sense

Poniższa tabela zawiera funkcje skryptu i wykresów, których używanie w programie Qlik Sense jest niezalecane.

Funkcje niezalecane

Funkcja	Zalecenie
NumAvg **NumCount** **NumMax** **NumMin** **NumSum**	Zastąp funkcjami zakresu. *Funkcje zakresu (page 1308)*
Color() **QliktechBlue** **QliktechGray**	Zastąp innymi funkcjami koloru. Identyczne kolory do funkcji **QliktechBlue()** i **QliktechGray** daje użycie odpowiednio funkcji **RGB(8, 18, 90)** i **RGB(158, 148, 137)**. *Funkcje koloru (page 538)*
QlikViewVersion	Zastąp instrukcją **EngineVersion**. *EngineVersion (page 1439)*
ProductVersion	Zastąp instrukcją **EngineVersion**. *EngineVersion (page 1439)*
QVUser	
Year2Date	Zastąp instrukcją **YearToDate**.
Vrank	Zastąp instrukcją **Rank**.
WildMatch5	Zastąp instrukcją **WildMatch**.

Kwalifikator ALL

W programie QlikView kwalifikator **ALL** może występować przed wyrażeniem. Jest to równoważne zapisowi **{1} TOTAL**. W takim przypadku obliczenie jest wykonywane na wszystkich wartościach pól dokumentu, bez względu na wymiary wykresu i bieżące selekcje. Niezależnie od stanu logicznego dokumentu będzie zawsze zwracana ta sama wartość. Użycie kwalifikatora **ALL** wyklucza użycie wyrażenia zestawu, ponieważ sam kwalifikator **ALL** definiuje zestaw. W celu zachowania kompatybilności kwalifikator **ALL** będzie nadal działać w bieżącej wersji Qlik Sense, ale może zostać usunięty w przyszłych wersjach.