指令碼語法和圖表函數

Qlik Sense®

February 2024

Contents

1 什麼是 Qlik Sense?

Qlik Sense 是進行資料分析的平台。透過 Qlik Sense，您可自行分析資料及探索資料。您可以在群組中及組織之間分享知識及分析資料。Qlik Sense 可供您自行提問及回答問題，並隨著自己的步調深入瞭解。Qlik Sense 讓您及您的同事可合力做出決策。

1.1 您可以在 Qlik Sense 中做什麼？

大部分商業智慧 (BI) 產品可以幫助您回答進一步理解的問題。但是您的後續問題呢？其他人閱讀您的報告或觀看您的視覺化後提出的問題？藉由 Qlik Sense 關聯經驗，您可以逐個回答問題，按照您自己的步調瞭解該產品。使用 Qlik Sense，您只需按幾下即可自由探索資料，瞭解該方法的每一個步驟，並根據先前的發現項目進行後續步驟。

1.2 Qlik Sense 如何運作？

Qlik Sense 會在執行中為您產生資訊檢視。Qlik Sense 不需要預先定義的靜態報告，或者您依賴於其他的使用者 - 您只需按一下並瞭解相關情況。每當您按一下時，Qlik Sense 會立即回應，更新應用程式中的每個 Qlik Sense 視覺化與檢視，以及與您的選項特定相關的一組新計算的資料與視覺化。

應用程式模型

不同於部署和管理大型的商業應用程式，您可以建立能夠重複使用、修改並與其他人共用的個人專屬 Qlik Sense 應用程式。應用程式模型幫助您獨立詢問並回答問題，而無需返回向專家要求新報告或視覺化。

關聯體驗

Qlik Sense 會自動管理資料中的所有關係，並使用 **green/white/gray** 方式向您呈現資訊。選項以綠色醒目提示，關聯的資料以白色顯示，而已排除 (未關聯) 的資料以灰色顯示。這個即時回饋可讓您考慮新的問題，並繼續瀏覽及探索。

共同作業及行動性

Qlik Sense 可讓您進一步與同事共同作業，無需考慮時間及他們身處何地。所有 Qlik Sense 功能，包括關聯體驗及共同作業，皆位於行動裝置上。透過 Qlik Sense，您可與您的同事詢問及回答問題，而無論你們身處何地。

1.3 您可以如何部署 Qlik Sense?

有兩種版本的 Qlik Sense 可供部署，即 Qlik Sense Desktop 及 Qlik Sense Enterprise。

Qlik Sense Desktop

這是易於安裝的單一使用者版本，通常安裝在本機電腦上。

Qlik Sense Enterprise

此版本用來部署 Qlik Sense 網站。網站是連接至一般邏輯存放庫或中心節點的一個或多個伺服器機器的集合。

1.4 如何管理 Qlik Sense 網站

使用 Qlik 管理主控台, 您可以使用簡單直觀的方法設定、管理及監視 Qlik Sense 網站。您可以管理授權、存取及安全規則, 設定節點及資料來源連線, 以及同步許多其他活動與資源的內容及使用者。

1.5 展開 Qlik Sense 並根據您自己的目的進行調整

Qlik Sense 為您提供靈活的 API 與 SDK, 讓您可以開發自己的延伸, 並針對不同的目的調整及整合 Qlik Sense, 例如:

建置延伸及 mashup

您可以使用 JavaScript 執行網頁開發, 以建置作為 Qlik Sense 應用程式中自訂視覺化的延伸, 或者使用 Mashup API 來建置包含 Qlik Sense 內容的網站。

建立用戶端

您可以在自己應用程式中的 .NET 與內嵌 Qlik Sense 物件內建置用戶端。您也可以利用 Qlik Sense 用戶端協定, 使用任何可以處理 WebSocket 通訊的程式設計語言來建置原生用戶端。

建立伺服器工具

使用服務與使用者目錄 API, 您可以建置自己的工具以管理 Qlik Sense 網站。

連接至其他資料來源

建立 Qlik Sense 連接器, 以從自訂資料來源擷取資料。

2 指令碼語法概述

2.1 指令碼語法簡介

在指令碼中，會定義邏輯中所包含的資料來源名稱、表格名稱及欄位名稱。此外，指令碼中也會定義存取權定義中的欄位。指令碼是由一些連續執行的陳述式所組成。

Qlik Sense 命令行語法和指令碼語法是以稱為 Backus-Naur 形式論 (或稱 BNF 代碼) 的標記法來描述。

在建立新的 Qlik Sense 檔案時，會先產生前面幾行的程式碼。這些數字解譯變數的預設值是衍生自作業系統的區域設定。

指令碼是由一些連續執行的指令碼陳述式和關鍵字所組成。所有的指令碼陳述式都必須以分號 "**;**" 為結束。

您可以在 **LOAD** 陳述式中使用運算式和函數來轉換已載入的資料。

對於使用逗號、定位點或分號作為分隔符號的表格檔案，可以使用 **LOAD** 陳述式。**LOAD** 陳述式預設會載入檔案的所有欄位。

一般資料庫均可透過 ODBC 或 OLE DB 資料庫連線器存取。在此會使用標準 SQL 陳述式。接受的 SQL 語法在不同的 ODBC 驅動程式之間有所不同。

此外，您可以使用自訂連線器存取其他資料來源。

2.2 什麼是 Backus-Naur 形式論？

Qlik Sense 指令行語法和指令碼語法使用名為 Backus-Naur 形式論的標記法進行說明，該標記法也稱為 BNF 代碼。

下表提供 BNF 代碼中使用的符號清單，以及如何解譯它們的描述：

符號

符號	描述
|	邏輯 OR：此符號可用於任一側。
( )	定義優先順序的括弧：用於架構 BNF 語法。
[ ]	方括弧：括住的項目為選用。
{ }	大括弧：括住的項目可重複零次或更多次。
符號	非終端語法類別：可進一步分割為其他符號。例如上述符號的組合、其他非終端符號、文字字串等。
::=	標示用於定義符號的區塊開端。
LOAD	包含文字字串的終端符號。應依原樣寫入指令碼中。

所有終端符號都會使用 **bold face** 字型予以列印。例如，"(" 應解譯為定義優先順序的括弧，而 "**(**" 則應解譯為要寫入指令碼中的字元。

範例：

alias 陳述式的描述如下：

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

這應解譯為文字字串 "alias"，後面接著一個任意的欄位名稱，再接著文字字串 "as"，然後接著一個任意的別名名稱。可另外加上任何數目的 "fieldname as alias" 組合 (以逗號分隔)。

下列都是正確的陳述式：

```
alias a as first;

alias a as first, b as second;

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

而下列都是不正確的陳述式：

```
alias a as first b as second;

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

3 指令碼陳述式和關鍵字

Qlik Sense 指令碼由多個陳述式組成。陳述式可以是一般的指令碼陳述式或是指令碼控制陳述式。某些陳述式的前面可加上前置詞。

一般陳述式通常用於以某種方式操縱資料。這些陳述式可在指令碼中撰寫為任意行數, 但是一定必須以分號 (";") 終止。

控制陳述式通常用於控制指令碼執行的流程。控制陳述式的每個子句都必須保持在一個指令碼行內, 並可以分號或行結尾終止。

前置詞可套用於適用的一般陳述式, 但絕不可套用於控制陳述式。**when** 和 **unless** 前置詞可作為一些特定控制陳述式子句的後置詞。

在下一個子章節中, 將按字母順序列出所有的指令碼陳述式、控制陳述式及前置詞。

所有的指令碼關鍵字皆可以小寫和大寫字元的任意組合輸入。但用於陳述式中的欄位和變數名稱則會區分大小寫。

3.1 指令碼控制陳述式

Qlik Sense 指令碼由多個陳述式組成。陳述式可以是一般的指令碼陳述式或是指令碼控制陳述式。

控制陳述式通常用於控制指令碼執行的流程。控制陳述式的每個子句都必須保持在一個指令碼行內, 並可以分號或行結尾終止。

控制陳述式絕不可套用前置詞, 但前置詞 **when** 和 **unless** 是例外, 它們可用於一些特定的控制陳述式。

所有的指令碼關鍵字皆可以小寫和大寫字元的任意組合輸入。

指令碼控制陳述式概述

概述之後, 會進一步描述每個函數。您還可以在語法中按一下函數名稱, 以立即存取該特定函數的詳細資料。

Call

call 控制陳述式會呼叫必須由先前的 **sub** 陳述式定義的副程式。

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

Do..loop

do..loop 控制陳述式是指令碼反覆運算建構, 這會執行一或數個陳述式, 直到符合邏輯條件為止。

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

Exit script

此控制陳述式可停止指令碼執行。它可以插入指令碼的任一處。

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

For each ..next

for each..next 控制陳述式是指令碼反覆運算建構，這會對於逗號分隔清單中的各個值執行一或數個陳述式。將會為清單的各個值，執行由 **for** 與 **next** 括住之迴圈內的陳述式。

```
For each..next var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

For..next

for..next 控制陳述式是包含計數器的指令碼反覆運算建構。將對於所指定上下限之間的各個 counter 變數值，執行 **for** 與 **next** 所括住迴圈之內的陳述式。

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

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

If..then

if..then 控制陳述式是一種陳述式選項建構，會根據一或數個邏輯條件強制指令碼執行遵照不同的路徑。

> *由於 **if..then** 陳述式是控制陳述式，因而會以分號或行尾來結束，所以這四個可能的子句 (**if..then**、**elseif..then**、**else** 和 **end if**) 都不能超過行邊界。*

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

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]
```

```
end if
```

Sub

sub..end sub 控制陳述式會定義可從 **call** 陳述式呼叫的副程式。

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

Switch

switch 控制陳述式是一個指令碼選項建構，會根據運算式的值強制指令碼執行遵照不同的路徑。

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

Call

call 控制陳述式會呼叫必須由先前的 **sub** 陳述式定義的副程式。

語法：

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

引數：

引數

引數	描述
name	副程式的名稱。
paramlist	要傳送到副程式的實際參數的逗號分隔清單。清單中的各個項目可能是欄位名稱、變數或任意的運算式。

call 陳述式呼叫的副程式必須由指令碼執行期間較早出現的 **sub** 加以定義。

參數會複製到副程式中，而且如果 **call** 陳述式中的參數是變數而不是運算式，將在結束副程式時再次複製回參數。

限制：

- 由於 **call** 陳述式是控制陳述式，而且結尾是分號或行結尾，因此不可超出行邊界。
- 透過控制陳述式內部的 `Sub..end sub` 定義副程式時，例如 `if..then`，您只能從相同的控制陳述式內部叫用副程式。

範例：

此範例列出了在一個資料夾及其子資料夾中的所有 Qlik 相關檔案，並在表格中儲存檔案資訊。它假定您已建立一個名為 Apps、連接該資料夾的資料連線。

將透過參考該資料夾，以 'lib://Apps' 為參數呼叫 DoDir 副程式。在副程式內，有一個循環呼叫 `Call DoDir (Dir)`，它可使函數在子資料夾中循環搜尋檔案。

```
sub DoDir (Root)
  For Each Ext in 'qvw', 'qvo', 'qvs', 'qvt', 'qvd', 'qvc', 'qvf'
```

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

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

Do..loop

do..loop 控制陳述式是指令碼反覆運算建構，這會執行一或數個陳述式，直到符合邏輯條件為止。

語法：

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

*由於 **do..loop** 陳述式是控制陳述式，因而會以分號或行結尾來結束，所以這三個可能的子句 (**do**、**exit do**、與 **loop**) 都不能超過行邊界。*

引數：

引數

引數	描述
condition	評估為 True 或 False 的邏輯運算式。
statements	是一或多個 Qlik Sense 指令碼陳述式的任何群組。
while / until	**while** 或 **until** 條件子句必須只在任何 **do..loop** 陳述式 (也就是在 **do** 之後或 **loop** 之後) 中出現一次。只有在各個條件第一次出現時，才會予以解譯，不過每次在迴圈中出現時，都會予以評估。
exit do	如果 **exit do** 子句出現在迴圈中，指令碼的執行將傳輸到 **loop** 子句後的第一個陳述式，代表迴圈結束。選用 **when** 或 **unless** 後置詞，即可將 **exit do** 子句設定為條件子句。

範例：

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

Set a=1;
```

```
Do while a<10

LOAD * from file$(a).csv;

Let a=a+1;

Loop
```

End

End 指令碼關鍵字用來結束 **If**、**Sub** 和 **Switch** 子句。

Exit

Exit 指令碼關鍵字屬於 **Exit Script** 陳述式，但是也可以用來結束 **Do**、**For** 或 **Sub** 子句。

Exit script

此控制陳述式可停止指令碼執行。它可以插入指令碼的任一處。

語法：

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

由於 **exit script** 陳述式是控制陳述式，而且結尾是分號或行結尾，因此不可超出行邊界。

引數：

引數

引數	描述
condition	評估為 True 或 False 的邏輯運算式。
when / unless	選用 **when** 或 **unless** 子句可將 **exit script** 陳述式設定為條件式。

範例：

```
//Exit script
Exit Script;

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

For..next

for..next 控制陳述式是包含計數器的指令碼反覆運算建構。將對於所指定上下限之間的各個 counter 變數值，執行 **for** 與 **next** 所括住迴圈之內的陳述式。

語法：

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

```
[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

只有在第一次進入迴圈時，才會評估 *expr1*、*expr2* 和 *expr3* 運算式。迴圈內的陳述式可變更 counter 變數的值，不過在程式設計上這不見得是好方法。

如果 **exit for** 子句出現在迴圈中，指令碼的執行將傳輸到 **next** 子句後的第一個陳述式，代表迴圈結束。選用 **when** 或 **unless** 後置詞，即可將 **exit for** 子句設定為條件子句。

*由於 **for..next** 陳述式是控制陳述式，因而會以分號或行結尾來結束，所以這三個可能的子句 (**for..to..step**、**exit for**、與 **next**) 都不能超過行邊界。*

引數：

引數

引數	描述
counter	變數名稱。如果在 **next** 之後指定 *counter*，它的變數名稱必須與對應的 **for** 之後出現的變數名稱相同。
expr1	決定應對其執行迴圈的第一個 *counter* 變數值所用的運算式。
expr2	決定應對其執行迴圈的最後一個 *counter* 變數值所用的運算式。
expr3	決定每次執行迴圈時，表示 *counter* 變數遞增的值所用的運算式。
condition	評估為 True 或 False 的邏輯運算式。
statements	是一或多個 Qlik Sense 指令碼陳述式的任何群組。

Example 1: 載入一連串檔案

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

for a=1 to 9

      LOAD * from file$(a).csv;

next
```

Example 2: 載入隨機數目的檔案

在此範例中，我們假設存在下列資料檔案 *x1.csv*、*x3.csv*、*x5.csv*、*x7.csv* 和 *x9.csv*。使用 `if rand( )<0.5 then` 條件在隨機點停止載入。

```
for counter=1 to 9 step 2
```

```
	set filename=x$(counter).csv;

	if rand( )<0.5 then

		exit for unless counter=1

	end if

	LOAD a,b from $(filename);

next
```

For each..next

for each..next 控制陳述式是指令碼反覆運算建構，這會對於逗號分隔清單中的各個值執行一或數個陳述式。將會為清單的各個值，執行由 **for** 與 **next** 括住之迴圈內的陳述式。

語法：

特殊語法能夠以目前目錄中的檔案和目錄名稱產生清單。

```
for each var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

引數：

引數

引數	描述
var	為各個迴圈執行從清單取得新值的指令碼變數名稱。如果在 **next** 之後指定 **var**，它的變數名稱必須與對應的 **for each** 之後出現的變數名稱相同。

迴圈內的陳述式可變更 **var** 變數的值，不過在程式設計上這不見得是好方法。

如果 **exit for** 子句出現在迴圈中，指令碼的執行將傳輸到 **next** 子句後的第一個陳述式，代表迴圈結束。選用 **when** 或 **unless** 後置詞，即可將 **exit for** 子句設定為條件子句。

*由於 **for each..next** 陳述式是控制陳述式，因而會以分號或行結尾來結束，所以這三個可能的子句 (**for each**、**exit for**、與 **next**) 都不能超過行邊界。*

語法：

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

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

引數

引數	描述
constant	任何數字或字串。請注意，直接在指令碼中寫入的字串必須以單引號括住。沒有單引號的字串將被解譯為變數，因而將使用變數的值。數字不需要以單引號括住。
expression	任意運算式。
mask	檔案名稱或目錄名稱遮罩，其中可能包含任何有效的檔案名稱字元，以及標準的萬用字元 ***** 和 **?**。 您可以使用絕對檔案路徑或 lib:// 路徑。
condition	評估為 True 或 False 的邏輯運算式。
statements	是一或多個 Qlik Sense 指令碼陳述式的任何群組。
filelist mask	此語法會對於符合檔案名稱遮罩的目前目錄中所有的檔案產生逗號分隔清單。 *此引數僅在標準模式下支援物件庫連線。*
dirlist mask	此語法會對於符合資料夾名稱遮罩的目前資料夾中所有的資料夾產生逗號分隔清單。 *此引數僅在標準模式下支援物件庫連線。*
fieldvaluelist mask	此語法透過已載入至 Qlik Sense 的欄位值反覆運算。

*Qlik 網頁儲存空間提供者連接器 和其他 DataFiles 連線不支援使用萬用字元 (***** 和 **?**) 的篩選遮罩。*

Example 1: **載入檔案清單**

```
// 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: **在磁碟上建立檔案清單**

此範例會載入資料夾中所有 Qlik Sense 相關檔案的清單。

```
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: 透過欄位值反覆運算

此範例會透過 FIELD 的已載入值清單反覆運算，並產生新的欄位 NEWFIELD。對於 FIELD 的每一個值，將建立兩筆 NEWFIELD 記錄。

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

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

產生的表格如下所示：

Example table

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

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

if..then 控制陳述式是一種陳述式選項建構，會根據一或數個邏輯條件強制指令碼執行遵照不同的路徑。

控制陳述式通常用於控制指令碼執行的流程。在圖表運算式中，請改用 **if** 條件式函數。

語法：

```
If  condition then

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

由於 **if..then** 陳述式是控制陳述式，因而會以分號或行尾來結束，所以這四個可能的子句 (**if..then**、**elseif..then**、**else** 和 **end if**) 都不能超過行邊界。

引數：

引數

引數	描述
condition	可評估為 True 或 False 的邏輯運算式。
statements	是一或多個 Qlik Sense 指令碼陳述式的任何群組。

Example 1:

```
if a=1 then

     LOAD * from abc.csv;

     SQL SELECT e, f, g from tab1;

end if
```

Example 2:

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

Example 3:

```
if x>0 then

      LOAD * from pos.csv;

elseif x<0 then

      LOAD * from neg.csv;

else

      LOAD * from zero.txt;

end if
```

Next

Next 指令碼用來結束 **For** 迴圈。

Sub..end sub

sub..end sub 控制陳述式會定義可從 **call** 陳述式呼叫的副程式。

語法:

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

引數會複製到副程式中，而且如果 **call** 陳述式中的對應實際參數是變數名稱，將在結束副程式時再次複製回引數。

如果副程式擁有的型式參數比 **call** 陳述式傳遞的實際參數多，則會將額外的參數初始化為 NULL，且可在副程式內做為本機變數。

引數:

引數

引數	描述
name	副程式的名稱。
paramlist	以逗號分隔副程式型式參數之變數名稱的清單。其可在副程式內做為任何變數。
statements	是一或多個 Qlik Sense 指令碼陳述式的任何群組。

限制:

- 由於 **sub** 陳述式是控制陳述式，因而會以分號或行尾來結束，所以這兩個子句 (**sub** 與 **end sub**) 都不能超過行邊界。
- 透過控制陳述式內部的 `Sub..end sub` 定義副程式時，例如 `if..then`，您只能從相同的控制陳述式內部叫用副程式。

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: - 參數傳送

```
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)
```

以上範例的結果會是在本機的副程式內, A 將初始化為 1、B 將初始化為 4, 而 C 將初始化為 NULL。

結束副程式後, 全域變數 A 會取得 2 做為值 (從副程式複製回來)。第二個實際參數 “(X+1)*2” 因為不是變數, 所以不會被複製回來。最後, 全域變數 C 不會受到副程式呼叫的影響。

Switch..case..default..end switch

switch 控制陳述式是一個指令碼選項建構, 會根據運算式的值強制指令碼執行遵照不同的路徑。

語法:

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

*由於 **switch** 陳述式是控制陳述式, 因而會以分號或行尾來結束, 所以這四個可能的子句 (**switch**、**case**、**default** 和 **end switch**) 都不能超過行邊界。*

引數:

引數

引數	描述
expression	任意運算式。
valuelist	是以逗號分隔的值清單, 其會與運算式的值進行比較。指令碼將繼續執行, 且遇到 valuelist 值的第一個群組中的陳述式等於運算式中的值。valuelist 中的每個值可以是任意運算式。如果在所有 **case** 子句中都找不到任何相符項目, 將會執行 **default** 子句下的陳述式 (若已指定)。
statements	是一或多個 Qlik Sense 指令碼陳述式的任何群組。

範例:

```
Switch I

Case 1

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

Case 2

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

Default

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

End Switch
```

To

To 指令碼關鍵字用於數個指令碼陳述式中。

3.2 指令碼前置詞

前置詞可套用於適用的一般陳述式, 但絕不可套用於控制陳述式。**when** 和 **unless** 前置詞可作為一些特定控制陳述式子句的後置詞。

所有的指令碼關鍵字皆可以小寫和大寫字元的任意組合輸入。但用於陳述式中的欄位和變數名稱則會區分大小寫。

指令碼前置詞概述

概述之後, 會進一步描述每個函數。您還可以在語法中按一下函數名稱, 以立即存取該特定函數的詳細資料。

Add

Add 前置詞可新增至指令碼中的任何 **LOAD** 或 **SELECT** 陳述式，以指定這應將記錄新增至另一個表格。這也能指定應在部分載入中執行此陳述式。**Add** 前置詞也能用於 **Map** 陳述式。

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

```
Add [ Only ] mapstatement
```

Buffer

透過 **buffer** 前置詞可以自動建立和維護 QVD 檔案。這個前置詞可以用於指令碼的大多數 **LOAD** 和 **SELECT** 陳述式。其表示 QVD 檔案會用於快取/緩衝陳述式的結果。

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

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

Concatenate

如果將串連的兩個表格有不同的欄位集合，仍然使用 **Concatenate** 前置詞強制串連兩個表格。

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

Crosstable

crosstable 載入前置詞用來轉置「跨維度資料表」或「樞紐分析表」結構資料。使用試算表來源時通常會遇到以此方式結構化的資料。**crosstable** 載入前置詞的輸出和目標是將這類結構轉置為一般欄導向的表格對等項目，因為此結構通常更適合 Qlik Sense 中的分析。

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

First

LOAD 或 **SELECT (SQL)** 陳述式的 **First** 前置詞可用來載入資料來源表格的一組最多筆數的記錄。

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

Generic

Generic 載入前置詞允許將實體-屬性-值模型化資料 (EAV) 轉換為傳統的標準化關係表格結構。EAV 模型化也稱為「一般資料模型化」或「開啟結構描述」。

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

Hierarchy

hierarchy 前置詞用來將上下層式階層表格轉換為可以在 Qlik Sense 資料模型中使用的表格。它可以放在 **LOAD** 或 **SELECT** 陳述式的前面，而且將使用載入陳述式的結果做為表格轉換的輸入。

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

HierarchBelongsTo

此前置詞用來將上下層式階層表格轉換為可以在 Qlik Sense 資料模型中使用的表格。它可以放在 **LOAD** 或 **SELECT** 陳述式的前面，而且將使用載入陳述式的結果做為表格轉換的輸入。

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

Inner

join 及 **keep** 前置詞前面可以加上 **inner** 前置詞。

如果加在 **join** 之前，會指定應該使用 inner join。產生的表格會因此只包含原始資料表中的欄位值組合，其中兩個表格會呈現連結欄位值。如果加在 **keep** 之前，會指定兩個原始資料表應先減少到其共同交集之後，然後才儲存到 Qlik Sense 中。
。

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

IntervalMatch

延伸的 **IntervalMatch** 前置詞可用來建立一個表格，比對離散數值與一個或多個數值間隔，並選擇性地比對一或數個其他索引鍵的值。

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

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

Join

join 前置詞能夠將載入的表格與現有的具名表格或先前最後建立的資料表聯結。

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

Keep

keep 前置詞類似於 **join** 前置詞。與 **join** 前置詞一樣，它會比較已載入的表格與現有具名表格或最後一個先前建立的資料表格，而不會聯結已載入的表格與現有表格，它會在將一個或兩個表格儲存在 Qlik Sense 中之前，根據表格資料的交集進行減少。進行比較相當於在所有通用欄位上進行自然聯結，也就是如同對應聯結中的方法。不過，兩個表格並未聯結，而是以兩個個別的具名表格保留在 Qlik Sense 中。

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

Left

Join 及 **Keep** 前置詞前面可以加上 **left** 前置詞。

如果加在 **join** 之前，會指定應該使用 left join。產生的表格只會包含原始資料表中的欄位值組合，其中第一個表格會呈現連結欄位值。如果加在 **keep** 之前，會指定第二個原始資料表應先減少到與第一個表格的共同交集之後，然後才儲存到 Qlik Sense 中。

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

Mapping

mapping 前置詞可用來建立對應表，例如，對應表可用於在指令碼執行期間取代欄位值和欄位名稱。

```
對應 ( loadstatement | selectstatement )
```

Merge

Merge 前置詞可新增至指令碼中的任何 **LOAD** 或 **SELECT** 陳述式，以指定載入的表格應合併至另一個表格中。這也能指定應在部分載入中執行此陳述式。

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

NoConcatenate

NoConcatenate 前置詞會強制將具有相同欄位集的兩個載入表格在自動串連時視為兩個個別的內部表格。

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

Outer

明確的 **Join** 前置詞可接在前置詞 **Outer** 之後，以指定外部聯結。在外部聯結中，會在兩個表格間產生所有組合。產生的表格會因此包含原始資料表格中的欄位值組合，其中一或兩個表格會呈現連結欄位值。**Outer** 關鍵字是選用項目，是未指定聯結前置詞時使用的預設連結。

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

Partial reload

完整載入永遠從刪除現有資料模型中的所有表格開始，然後執行載入指令碼。

部分載入 (page 95) 將不會這麼做。反而這會保留資料模型中的所有表格，然後僅執行前面有 **Add**、**Merge** 或 **Replace** 前置詞的 **Load** 和 **Select** 陳述式。其他資料表格不會受到命令影響。**only** 引數表示只應在部分載入期間執行陳述式，且應在完整載入期間忽略。下列表格概述部分和完整載入的陳述式執行。

Replace

Replace 前置詞 可新增至指令碼中的任何 **LOAD** 或 **SELECT** 陳述式，以指定載入的表格應取代另一個表格。這也能指定應在部分載入中執行此陳述式。**Replace** 前置詞也能用於 **Map** 陳述式。

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

```
Replace [only] mapstatement
```

Right

Join 及 **Keep** 前置詞前面可以加上 **right** 前置詞。

如果加在 **join** 之前，會指定應該使用 right join。產生的表格只會包含原始資料表格中的欄位值組合，其中第二個表格會呈現連結欄位值。如果加在 **keep** 之前，會指定第一個原始資料表應先減少到與第二個表格的共同交集之後，然後才儲存到 Qlik Sense 中。

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

Sample

LOAD 或 **SELECT** 陳述式的 **sample** 前置詞可用來載入資料來源的隨機記錄樣本。

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

Semantic

透過 **semantic** 前置詞，即可載入包含記錄之間關係的表格。例如，這可以在表格內自我參考，其中一筆記錄指向另一個，例如上層、屬於或前任。

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

Unless

unless 前置詞及後置詞用來建立條件式子句，其可決定是否應該評估陳述式或結束子句。它可視為完整 **if..end if** 陳述式的精簡替代選項。

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

When

when 前置詞及後置詞用來建立條件式子句，其可決定是否應該執行陳述式或結束子句。它可視為完整 **if..end if** 陳述式的精簡替代選項。

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

Add

Add 前置詞可新增至指令碼中的任何 **LOAD** 或 **SELECT** 陳述式，以指定這應將記錄新增至另一個表格。這也能指定應在部分載入中執行此陳述式。**Add** 前置詞也能用於 **Map** 陳述式。

為了讓部分載入正常運作，必須在觸發部分載入之前以資料開啟應用程式。

使用**載入**按鈕執行部分載入。您也可以使用 Qlik Engine JSON API。

語法：

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

```
Add [only] mapstatement
```

在一般 (非部分) 載入期間，**Add LOAD** 建構將會作為一般 **LOAD** 陳述式運作。將會產生記錄並儲存在表格中。

若使用 **Concatenate** 前置詞，或者若存在具有相同欄位集的表格，記錄將會附加至相關現有表格。否則，**Add LOAD** 建構將會建立新的表格。

部分載入將會進行相同事項。唯一的差異是 **Add LOAD** 建構永遠不會建立新的表格。永遠存在來自先前指令碼執行的相關表格，記錄應附加於此。

其中不會檢查重複項目。因此，使用 **Add** 前置詞的陳述式通常包含 distinct 限定詞或規範重複項目的 where 子句。

Add Map...Using 陳述式也會使得對應在部分指令碼執行期間進行。

引數:

引數

引數	描述
only	選用的限定詞，表示只應在部分載入期間執行該陳述式。這應在一般(非部分)載入期間忽略。

範例與結果:

範例	結果

`Tab1:`
 	
`LOAD Name, Number FROM Persons.csv;`
 	
`Add LOAD Name, Number FROM newPersons.csv;`
	在正常重新載入期間，會從 *Persons.csv* 載入資料並儲存到 Qlik Sense 表格 Tab1。來自 *NewPersons.csv* 的資料接著將串連到同一個 Qlik Sense 表格中。
	
	在部分重新載入期間，資料是從 *NewPersons.csv* 載入，並附加到 Qlik Sense 表格 Tab1 中。其中不會檢查重複項目。

`Tab1:`
 	
`SQL SELECT Name, Number FROM Persons.csv;`
 	
`Add LOAD Name, Number FROM NewPersons.csv where not exists(Name);`
	查看 Name 是否存在於先前載入的表格資料中，即可檢查重複項目。
	
	在正常重新載入期間，會從 *Persons.csv* 載入資料並儲存到 Qlik Sense 表格 Tab1。來自 *NewPersons.csv* 的資料接著將串連到同一個 Qlik Sense 表格中。
	
	在部分重新載入期間，資料是從 *NewPersons.csv* 載入，並附加到 Qlik Sense 表格 Tab1 中。查看 Name 是否存在於先前載入的表格資料中，即可檢查重複項目。

`Tab1:`
 	
`LOAD Name, Number FROM Persons.csv;`
 	
`Add Only LOAD Name, Number FROM NewPersons.csv where not exists(Name);`
	在正常重新載入期間，會從 *Persons.csv* 載入資料並儲存到 Qlik Sense 表格 Tab1。略過陳述式載入 *NewPersons.csv*。
	
	在部分重新載入期間，資料是從 *NewPersons.csv* 載入，並附加到 Qlik Sense 表格 Tab1 中。查看 Name 是否存在於先前載入的表格資料中，即可檢查重複項目。

Buffer

透過 **buffer** 前置詞可以自動建立和維護 QVD 檔案。這個前置詞可以用於指令碼的大多數 **LOAD** 和 **SELECT** 陳述式。其表示 QVD 檔案會用於快取/緩衝陳述式的結果。

語法:

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

如果不使用任何選項，則指令碼第一次執行所建立的 QVD 緩衝區就無使用期限。

緩衝區檔案儲存在 *Buffers* 子資料夾中，通常位於 *C:\ProgramData\Qlik\Sense\Engine\Buffers* (伺服器安裝) 或 *C:\Users\{user}\Documents\Qlik\Sense\Buffers* (Qlik Sense Desktop)。

QVD 檔案的名稱是一個計算名稱，整個後續 **LOAD** 或 **SELECT** 陳述式和其他區別資訊的 160 位元的十六進位雜湊。這表示 QVD 緩衝區會被後續 **LOAD** 或 **SELECT** 陳述式中的任何變更視為無效。

建立 QVD 緩衝區的應用程式之中的完整指令碼執行完全不再參考該緩衝區時，或者建立該緩衝區的應用程式不再存在時，一般都會移除該緩衝區。

引數：

引數

引數	描述
累加	incremental 選項能夠唯讀基礎檔案的一部分。先前的檔案大小儲存於 XML 檔案的 QVD 標頭中。這特別適用於記錄檔。先前載入的所有記錄都是從 QVD 檔案讀取的，而後續的新記錄則是從原始來源讀取的，最後會建立更新的 QVD 檔案。 incremental 選項只能用於 **LOAD** 陳述式和文字檔案。若已變更或已刪除舊資料，則無法使用增量載入。
stale [after] amount [(days | hours)]	amount 是指定時間週期的數字。可以使用小數。如果未指定單位，一律假設單位為天數。 stale after 選項一般會用於其中原始資料沒有簡單時間戳記的資料庫來源。而非指定 QVD 快照可以用多久時間。stale after 子句只是陳述從建立 QVD 緩衝區之後經過多久才會失效的時間週期。在這段時間之前，將使用 QVD 緩衝區做為資料的來源，過了這段時間之後，則將使用原始資料來源。此後 QVD 緩衝區檔案將自動更新，並且重啟新的週期。

限制：

其中有許多限制，最顯著的是任何複雜陳述式的核心都必須要有 **LOAD** 檔案或 **SELECT** 陳述式。

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 是指令碼載入前置詞，可讓資料集附加至已存在的記憶體內表格。這經常用於將不同組的交易資料附加至單一個中央的事實資料表，或是建置源自多項來源之特定類型的常見參考資料集。這類似 SQL UNION 運算子的功能。

concatenate 運算的結果表格包含原始資料集，其具有附加至表格底部的新資料列。來源和目標表格可能會出現不同的欄位。如果欄位不同，即會擴大結果表格以代表出現在來源表格和目標表格之所有欄位的組合結果。

語法：

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

引數

引數	描述
tablename	現有表格的名稱。命名的表格會是 Concatenate 運算的目標，且載入的任何資料記錄都會附加至該表格。如果未使用 tablename 參數，則目標表格會是此陳述式前的最後一個載入表格。
loadstatement/selectstatement	追蹤 tablename 引數的 loadstatement/selectstatement 引述會串連至特定表格。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
Concatenate (Transactions) Load …. ;	載入到 Concatenate 前置詞下方的資料會附加至名為 Transactions 現有記憶體內表格 (假設在載入指令碼的這個點前已載入名為 Transactions 的表格)。

範例 1 – 將多組資料附加至具有串連載入前置詞的目標表格

載入指令碼和結果

概覽

在此範例中，您會以連續順序載入兩個指令碼。

- 第一個載入指令碼包含具有日期和金額的初始資料集，其傳送至名為 Transactions 的表格中。
- 第二個載入指令碼包含：
  - 第二個資料集使用 Concatenate 前置詞附加至初始資料集。此資料集具有其他欄位 type，且不位於初始資料集。
  - Concatenate 前置詞。

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

第一個載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date
- amount

第一個載入指令碼結果表格

id	日期	金額
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

表格顯示初始資料集。

第二個載入指令碼

開啟資料載入編輯器，然後在下方新增載入指令碼。

```
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
];
```

結果

載入資料並前往工作表。將此欄位建立為維度使用。

- type

第二個載入指令碼結果表格

id	日期	金額	類型
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	內部
3759	10/03/2018	384.00	外部
3760	10/06/2018	25.82	內部
3761	10/09/2018	312.00	內部
3762	10/15/2018	4.56	內部
3763	10/16/2018	90.24	內部
3764	10/18/2018	19.32	外部

請注意 `type` 欄位中的 Null 值，前七筆記錄載入時未定義 `type`。

範例 2 – 使用隱含串連將多組資料附加至目標表格

載入指令碼和結果

概覽

隱含附加資料的一般使用案例即是當您載入相同結構化資料的多個檔案，並想要全部附加至目標表格的時候。

例如，將 `wildcards` 用於具有以下語法的檔案名稱中：

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

或是用於使用以下建構的迴圈中：

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

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

next file
```

> *隱含串連會發生在任兩個載入相同名稱之欄位的表格之間，即使它們並未在指令碼中互相定義。這會導致資料在無意間附加至表格中。若您不想以此方式附加具有相同欄位的第二個表格，請使用 `NoConcatenate` 載入前置詞。即使將該表格重新命名為替代的表格名稱標記，還是無法預防隱含串連發生。如需詳細資訊，請參閱NoConcatenate (page 85)。*

在此範例中，您會以連續順序載入兩個指令碼。

- 第一個載入指令碼包含具有四個欄位的初始資料集，其傳送至名為 `Transactions` 的表格中。
- 第二個載入指令碼包含的資料集具有與第一個資料集相同的欄位。

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

第一個載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- id
- date
- amount
- type

第一個載入指令碼結果表格

id	日期	類型	金額
3758	10/01/2018	內部	164.27
3759	10/03/2018	外部	384.00
3760	10/06/2018	內部	25.82
3761	10/09/2018	內部	312.00
3762	10/15/2018	內部	4.56
3763	10/16/2018	內部	90.24
3764	10/18/2018	外部	19.32

表格顯示初始資料集。

第二個載入指令碼

開啟資料載入編輯器,然後在下方新增載入指令碼。

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

結果

載入資料並前往工作表。

第二個載入指令碼結果表格

id	日期	類型	金額
3758	10/01/2018	內部	164.27
3759	10/03/2018	外部	384.00
3760	10/06/2018	內部	25.82
3761	10/09/2018	內部	312.00
3762	10/15/2018	內部	4.56

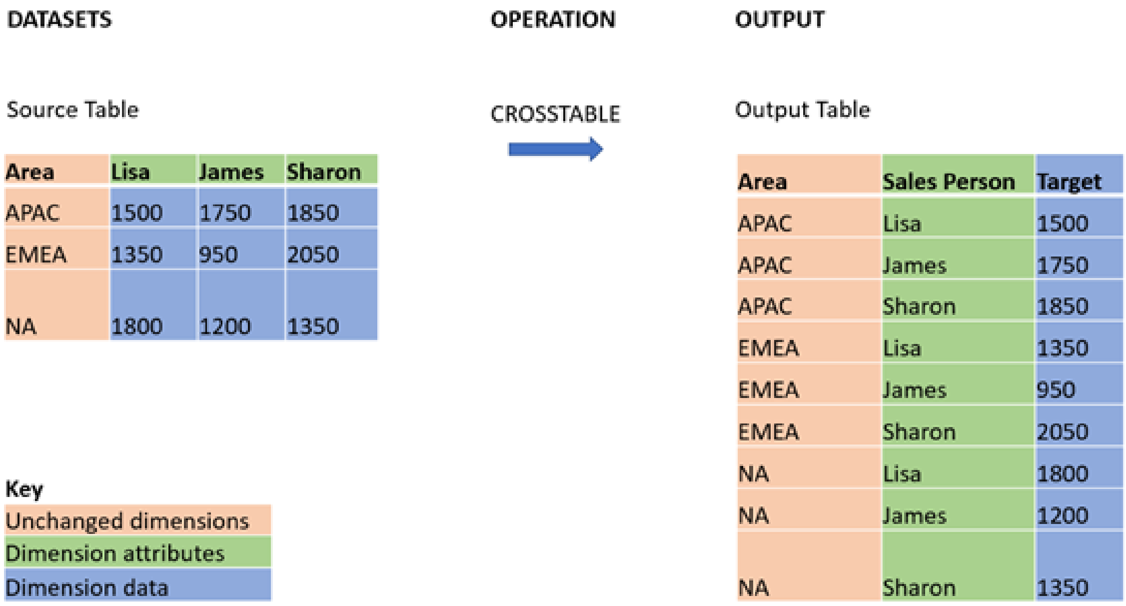
id	日期	類型	金額
3763	10/16/2018	內部	90.24
3764	10/18/2018	外部	19.32
3765	11/03/2018	內部	129.40
3766	11/05/2018	外部	638.50

第二個資料集隱含串連至初始資料集，因為它們具有相同的欄位。

Crosstable

crosstable 載入前置詞用來轉置「跨維度資料表」或「樞紐分析表」結構資料。使用試算表來源時通常會遇到以此方式結構化的資料。**crosstable** 載入前置詞的輸出和目標是將這類結構轉置為一般欄導向的表格對等項目，因為此結構通常更適合 Qlik Sense 中的分析。

結構化為跨維度資料表的資料及其在跨維度資料表轉換後的對等結構範例



語法：

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

引數

引數	描述
attribute field name	所需的輸出欄位名稱描述要轉置的水平方向維度 (標頭列)。

引數	描述
data field name	所需的輸出欄位名稱描述要轉置的水平方向維度資料 (標頭列之下的資料值矩陣)。
n	將轉換為一般表單的表格前面加上的限定詞欄位或未變更的維度數目。預設值為 1。

此指令碼函數與下列函數有關:

相關函數

函數	互動
Generic (page 55)	轉換載入前置詞採用實體-屬性-值結構化資料集, 並轉換為一般關係表格結構, 將遇到的每個屬性分為新的欄位或資料欄。

範例 1 – 轉換樞紐銷售資料 (簡單)

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的第一個載入指令碼新增至新的索引標籤。

第一個載入指令碼包含之後將套用 `crosstable` 指令碼前置詞的資料集, 其中套用 `crosstable` 的區段標記為註解。這表示註解語法已在載入指令碼中用來停用此區段。

第二個載入指令碼與第一個相同, 但 `crosstable` 的應用程式未標記為註解 (透過移除註解語法來啟用)。指令碼以此方式顯示, 以在轉換資料中醒目提示此指令碼函數的值。

第一個載入指令碼 (未套用函數)

```
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;
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

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

結果表格

產品	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

此指令碼允許建立跨維度資料表，其中每個月份都有一欄，而每個產品都有一列。在目前格式中，此資料不容易分析。比較好的做法會是某個欄位中具備所有數字以及另一個欄位 (即三欄表格) 中具備所有月份。下一區段說明如何對跨維度資料表進行此轉換。

第二個載入指令碼 (已套用函數)

移除 // 以為指令碼取消註解。載入指令碼應如下所示：

```
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;
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- `Product`
- `Month`
- `Sales`

結果表格

產品	月	銷售額
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

套用指令碼前置詞後，跨維度資料表已轉換為直式表格，其中 `Month` 為一欄，`Sales` 為另一欄。這可改善資料可讀性。

範例 2 – 將樞紐銷售目標資料轉換為垂直表格結構 (中繼)

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入到稱為 Targets 之表格的資料集。
- 將樞紐銷售人員名稱轉置為自己欄位的 `crosstable` 載入前置詞具有 `Sales Person` 標籤。
- 關聯銷售目標資料，這結構化為稱為 `Target` 的欄位。

載入指令碼

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

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- `Area`
- `Sales Person`

新增此量值：

`=Sum(Target)`

結果表格

區域	銷售人員	=Sum(Target)
APAC	James	1750
APAC	Lisa	1500
APAC	Sharon	1850
EMEA	James	950
EMEA	Lisa	1350
EMEA	Sharon	2050
不適用	James	1200
不適用	Lisa	1800
不適用	Sharon	1350

若您想要複寫資料顯示作為樞紐輸入表格，可以在工作表中建立對等的樞紐分析表。

請執行下列動作：

1. 複製並貼上您剛在工作表中建立的表格。
2. 將**樞紐分析表**圖表物件拖曳到新建立的表格副本頂端。選取**轉換**。
3. 按一下 ✔ **完成編輯**。
4. 將 `Sales Person` 欄位從垂直欄架拖曳到水平欄架。

下表以初始表格形式顯示資料，如 Qlik Sense 中顯示：

原始結果表格，如 Qlik Sense 中所示

區域	**銷售人員**	**=Sum(Target)**
總計	-	13800
APAC	James	1750
APAC	Lisa	1500
APAC	Sharon	1850
EMEA	James	950
EMEA	Lisa	1350
EMEA	Sharon	2050
不適用	James	1200
不適用	Lisa	1800
不適用	Sharon	1350

對等的樞紐分析表看起來與下列內容類似，在 `Sales Person` 的較大列內包含每個銷售人員名稱的欄：

`Sales Person` 欄位水平樞紐的對等樞紐分析表

區域	**James**	**Lisa**	**Sharon**
APAC	1750	1500	1850
EMEA	950	1350	2050
不適用	1350	1350	1350

顯示為表格的資料和 Sales Person 欄位水平樞紐之對等樞紐分析表的範例

Table

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

Pivot table

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

範例 3 – 將樞紐銷售和目標資料轉換為垂直表格結構 (進階)

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 呈現銷售和目標資料的資料集,按區域和一年中的月份組織。這載入到稱為 SalesAndTargets 的表格。
- crosstable 載入前置詞。這用來在專用欄位內取消 Month Year 維度樞紐,以及將銷售和目標金額的矩陣轉置為稱為 Amount 的專用欄位。
- Month Year 欄位使用文字至日期轉換函數 date# 從文字轉換為正確日期。此日期轉換的 Month Year 欄位透過 Join 載入前置詞聯結回到 SalesAndTarget 表格。

載入指令碼

```
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;
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- Area
- Month Year

透過標籤 `Actual` 建立下列量值:

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

也透過標籤 `Target` 建立此量值:

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

結果表格 (已裁剪)

區域	月 年	實際	目標
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

若您想要複寫資料顯示作為樞紐輸入表格,可以在工作表中建立對等的樞紐分析表。

請執行下列動作:

1. 複製並貼上您剛在工作表中建立的表格。
2. 將**樞紐分析表**圖表物件拖曳到新建立的表格副本頂端。選取**轉換**。
3. 按一下✔ **完成編輯**。
4. 將 `Month Year` 欄位從垂直欄架拖曳到水平欄架。
5. 將 `Values` 項目從水平欄架拖曳到垂直欄架。

下表以初始表格形式顯示資料, 如 Qlik Sense 中顯示:

原始結果表格 (已裁剪), 如 Qlik Sense 中所示

區域	**月 年**	**實際**	**目標**
總計	-	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

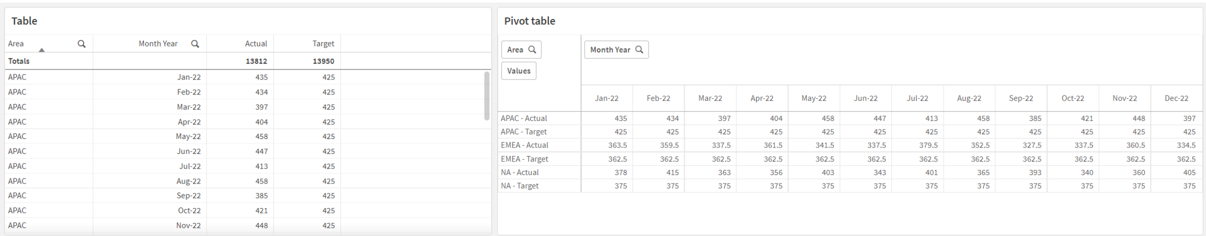
對等的樞紐分析表看起來與下列內容類似, 在 `Month Year` 的較大列內包含該年每個個別月份的欄:

`Month Year` 欄位水平樞紐的對等樞紐分析表 (已裁剪)

區域 (值)	**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 - 實際	435	434	397	404	458	447	413	458	385	421	448	397

區域(值)	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 - 目標	425	425	425	425	425	425	425	425	425	425	425	425
EMEA - 實際	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 - 目標	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 - 實際	378	415	363	356	403	343	401	365	393	340	360	405
NA - 目標	375	375	375	375	375	375	375	375	375	375	375	375

顯示為表格的資料和 `Month Year` 欄位水平樞紐之對等樞紐分析表的範例



First

`LOAD` 或 `SELECT` (SQL) 陳述式的 `First` 前置詞可用來載入資料來源表格的一組最多筆數的記錄。使用 `First` 前置詞的常見使用案例是若您想要從大型和/或慢速資料載入步驟擷取小的記錄子集。載入定義的「n」記錄數量後，載入步驟會永久終止，而指令碼執行的剩餘部分會照常繼續。

語法：

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

引數

引數	描述
n	一個任意運算式，會評估為整數，表示將讀取的最大記錄筆數。n 也可以括在括弧中：`(n)`。
`loadstatement | selectstatement`	在 n 引數之後的 `load statement/select statement` 將會以設定的最大記錄數量定義必須載入的指定表格。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
`FIRST 10 LOAD * from abc.csv;`	此範例將會從 excel 檔案擷取前十行。
`FIRST (1) SQL SELECT * from Orders;`	此範例將會從 `Orders` 資料集擷取選取的第一行。

範例 – 載入前五列

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 來自 2020 年前兩週的日期資料集。
- 指示應用程式僅載入前五筆記錄的 `First` 變數。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格，將 `Date` 新增為欄位，並將 `sum(sales)` 新增為量值：

結果表格

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

該指令碼僅載入 `Sales` 表格的前五筆記錄。

Generic

Generic 載入前置詞允許將實體–屬性–值模型化資料 (EAV) 轉換為傳統的標準化關係表格結構。EAV 模型化也稱為「一般資料模型化」或「開啟結構描述」。

EAV 模型化資料和對等去正規化關係表格的範例

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

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

EAV 模型化資料和正規化關係表格之對等集合的範例

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

雖然技術上可以在 Qlik 中載入並分析 EAV 模型化資料，但使用對等的傳統關係資料結構通常更簡單。

語法：

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

這些主題可協助您使用此函數：

相關主題

主題	描述
Crosstable *(page 44)*	`Crosstable` 載入前置詞將水平方向的資料轉換為垂直方向的資料。從純粹的功能性觀點來看，這對 `Generic` 載入前置詞執行相對轉換，雖然前置詞通常用於完全不同的使用案例。
*管理資料*中的 **泛型資料庫**	在此進一步說明 EAV 結構化資料模型。

範例 1 – 轉換含泛型載入前置詞的 EAV 結構化資料

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含載入到名為 `Transactions` 之表格的資料集。資料集包括日期欄位。使用預設 `MonthNames` 定義。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`Color`。

新增此量值：

`=Count([Product ID])`

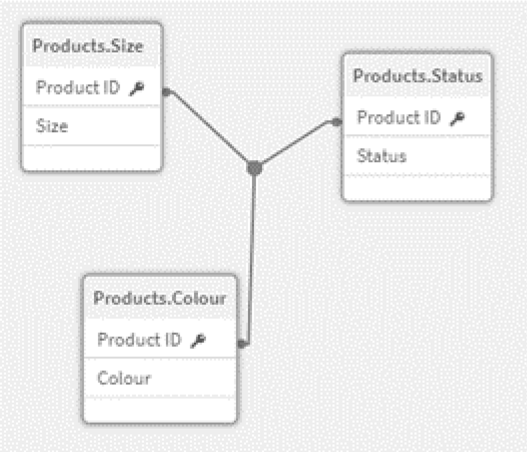
現在您可以按色彩檢查產品數量。

結果表格

色彩	=Count([Product ID])
棕色	4
白色	2

注意資料模型的形狀，其中每個屬性已分為根據原始目標表格標記 `Product` 命名的獨立表格。每個表格具有屬性作為後置詞。其中一個範例是 `Product.Color`。產生的產品屬性輸出記錄透過 `Product ID` 關聯。

資料模型檢視器結果呈現



產生的記錄表格：Products.Status

產品 ID	狀態
13	已中斷
2	已中斷

產生的記錄表格：Products.Size

產品 ID	大小
13	13-15
20	16-18
45	16-18

產生的記錄表格：Products.Color

產品 ID	色彩
13	棕色
5	棕色
44	棕色
45	棕色
20	白色
2	白色

範例 2 – 分析不含泛型載入前置詞的 EAV 結構化資料

載入指令碼和圖表運算式

概覽

此範例顯示如何以原始形式分析 EAV 結構化資料。

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含載入到 EAV 結構中名為 `Products` 之表格的資料集。

在此範例中，我們仍在按色彩屬性計算產品數。若要分析以此方式結構化的資料，您將需要套用具有屬性值 `Color` 之產品的運算式層級篩選。

此外，個別屬性無法用來選取作為維度或欄位，因此更難判定如何建置有效的視覺化。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`Value`。

建立下列量值：

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

現在您可以按色彩檢查產品數量。

產生的記錄表格：Products.Status

值	=Count({<Attribute={'Color'}>} [Product ID])
棕色	4
白色	2

範例 3 – 對從泛型載入 (進階) 產生的輸出表格去正規化

載入指令碼和圖表運算式

概覽

在此範例中，我們顯示透過 Generic 載入前置詞產生的正規化資料結構可以如何去正規化回到合併的 Product 維度表格。這是進階模型化技術，可作為資料模型效能微調的一部分採用。

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼

```
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;
```

結果

開啟資料模型檢視器並注意產生的資料模型形狀。僅存在一個去正規化的表格。這是三個中繼輸出表格的組合：Products.Size、Products.Status 和 Products.Color。

產生內部
資料模型

產品
產品 ID

產品
狀態
色彩
大小

產生的記錄表格：產品

產品 ID	狀態	色彩	大小
13	已中斷	棕色	13-15
20	-	白色	16-18
2	已中斷	白色	-
5	-	棕色	-
44	-	棕色	-
45	-	棕色	16-18

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`Color`。

新增此量值：

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

結果表格

色彩	=Count([Product ID])
棕色	4
白色	2

Hierarchy

hierarchy 前置詞用來將上下層式階層表格轉換為可以在 Qlik Sense 資料模型中使用的表格。它可以放在 **LOAD** 或 **SELECT** 陳述式的前面，而且將使用載入陳述式的結果做為表格轉換的輸入。

此前置詞會建立展開的節點表格，這一般會有與輸入表格相同的記錄數，不過除此之外階層中的各個層級會儲存在個別的欄位中。路徑欄位可以在樹狀結構中使用。

語法：

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

此輸入表格必須是相鄰節點表格。相鄰節點表格是各個記錄對應於一個節點的表格，其中有欄位包含上層節點的參考。在這類表格中，節點僅儲存在一筆記錄內，但是節點仍可有任何數目的子節點。當然，表格可包含其他用來描述節點屬性的欄位。

此前置詞會建立展開的節點表格，這一般會有與輸入表格相同的記錄數，不過除此之外階層中的各個層級會儲存在個別的欄位中。路徑欄位可以在樹狀結構中使用。

輸入表格通常對於各個節點只有一筆記錄，因此，輸出表格將包含相同筆數的記錄。不過，有時候節點會有多個上層節點，也就是一個節點由輸入表格中的多筆記錄表示。若是如此，輸出表格的記錄會比輸入表格多。

上層識別碼不在節點資料行中的所有節點 (包括遺失上層識別碼的節點) 將被視為根節點。另外，只有直接或間接連線到根節點的節點才會載入，以避免循環參照。

可建立包含上層節點名稱、節點路徑和節點深度的其他欄位。

引數:

引數

引數	描述
NodeID	包含節點識別碼的欄位名稱。此欄位必須存在於輸入表格中。
ParentID	包含上層節點識別碼的欄位名稱。此欄位必須存在於輸入表格中。
NodeName	包含節點名稱的欄位名稱。此欄位必須存在於輸入表格中。
ParentName	用於命名新 **ParentName** 欄位的字串。如果省略，將不會建立此欄位。
ParentSource	建立節點路徑所用節點名稱包含在其中的欄位名稱。選用參數。如果省略，將使用 **NodeName**。
PathName	用於命名新 **Path** (路徑) 欄位的字串，其中包含從根節點到該節點的路徑。選用參數。如果省略，將不會建立此欄位。
PathDelimiter	在新的 **Path** (路徑) 欄位中作為分隔符號的字串。選用參數。如果省略，將使用 '/'。
Depth	用於命名新 **Depth** (深度) 欄位的字串，此欄位包含節點在階層中的深度。選用參數。如果省略，將不會建立此欄位。

範例:

```
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

此前置詞用來將上下層式階層表格轉換為可以在 Qlik Sense 資料模型中使用的表格。它可以放在 **LOAD** 或 **SELECT** 陳述式的前面，而且將使用載入陳述式的結果做為表格轉換的輸入。

此前置詞會建立一個表格，包含階層的所有上階和下階關係。上階欄位可用來選取階層中的整個樹狀結構。在大多數情況下，此輸出表格會包含各個節點的數筆記錄。

語法：

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

此輸入表格必須是相鄰節點表格。相鄰節點表格是各個記錄對應於一個節點的表格，其中有欄位包含上層節點的參考。在這類表格中，節點僅儲存在一筆記錄內，但是節點仍可有任何數目的子節點。當然，表格可包含其他用來描述節點屬性的欄位。

此前置詞會建立一個表格，包含階層的所有上階和下階關係。上階欄位可用來選取階層中的整個樹狀結構。在大多數情況下，此輸出表格會包含各個節點的數筆記錄。

可建立包含節點深度差異的其他欄位。

引數：

引數

引數	描述
NodeID	包含節點識別碼的欄位名稱。此欄位必須存在於輸入表格中。
ParentID	包含上層節點識別碼的欄位名稱。此欄位必須存在於輸入表格中。
NodeName	包含節點名稱的欄位名稱。此欄位必須存在於輸入表格中。

引數	描述
AncestorID	用於命名新上階識別碼欄位的字串，其中包含上階節點的識別碼。
AncestorName	用於命名新上階欄位的字串，其中包含上階節點的名稱。
DepthDiff	用於命名新 **DepthDiff** 欄位的字串，其中包含節點在階層中相對於上階節點的深度。選用參數。如果省略，將不會建立此欄位。

範例：

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

NodeID, AncestorID, NodeName

1, 4, London

2, 3, Munich

3, 5, Germany

4, 5, UK

5, , Europe

];
```

Results

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

Inner

join 及 **keep** 前置詞前面可以加上 **inner** 前置詞。如果加在 **join** 之前，會指定應該使用 inner join。產生的表格會因此只包含原始資料表中的欄位值組合，其中兩個表格會呈現連結欄位值。如果加在 **keep** 之前，會指定兩個原始資料表應先減少到其共同交集之後，然後才儲存到 Qlik Sense 中。

語法：

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

引數：

引數

引數	描述
tablename	要與所載入表格比較的具名表格。
loadstatement或 selectstatement	已載入表格的 **LOAD** 或 **SELECT** 陳述式。

範例

載入指令碼

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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 ];
```

結果

結果表格

Column1	Column2	Column3
A	B	C
1	aa	xx

解釋

此範例展示內部聯結輸出，其中只會聯結同時出現在第一個 (左) 和第二個 (右) 表格中的值。

IntervalMatch

延伸的 **IntervalMatch** 前置詞可用來建立一個表格，比對離散數值與一個或多個數值間隔，並選擇性地比對一或數個其他索引鍵的值。

語法：

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

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

IntervalMatch 前置詞必須放在載入間隔的 **LOAD** 或 **SELECT** 陳述式之前。包含離散資料點 (在本例中為時間) 和其他索引鍵的欄位必須先載入到 Qlik Sense 之中含 **IntervalMatch** 前置詞的陳述式之前。此前置詞不會自行從資料庫表格讀取此欄位。此前置詞會將載入的間隔和索引鍵表格轉換為包含額外一個資料行的表格:離散數值資料點。它也會延伸記錄筆數, 讓新的表格在索引鍵欄位的離散資料點、間隔和值三者的各個可能組合中都有一筆記錄。

間隔可能會重疊, 所以離散值會連結到所有符合的間隔。

當 IntervalMatch 前置詞使用索引鍵欄位作為延伸時, 它可用於建立一個表格, 比對離散數值與一個或多個數值間隔, 同時比對一個或數個其他索引鍵的值。

為了避免忽略到未定義的間隔限制, 可能必須允許 NULL 值對應到構成間隔上下限的其他欄位。這種情況可由 **NullAsValue** 陳述式處理, 或由任何離散數值資料點前後以數值取代 NULL 的明確測試來處理。

引數:

引數

引數	描述
matchfield	包含離散數值的欄位, 這些離散數值將連結至間隔。
keyfield	包含其他屬性的欄位, 其他屬性將在轉換中進行比對。
loadstatement orselectstatement	必須產生表格, 其中第一個欄位包含每一個間隔的下限, 第二個欄位包含每一個間隔的上限, 在使用索引鍵配對時, 第三個和任何後續欄位都包含 **IntervalMatch** 陳述式中呈現的索引鍵欄位。間隔一定是封閉的, 也就是說, 結束點是包括在間隔內。非數值的限制會使得間隔被忽略 (未定義)。

Example 1:

在下方的兩個表格中, 第一個表格列出一系列離散事件, 第二個表格定義不同順序產生的開始和結束時間。透過 **IntervalMatch** 前置詞, 可以按照邏輯順序連接兩個表格, 以找出受干擾影響的順序, 以及哪個偏移處理哪些順序。

```
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;
```

表格 **OrderLog** 現在包含其他資料行 : *Time* 還會展開記錄數目。

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: (使用 keyfield)

與上述範例相同, 新增 *ProductionLine* 作為索引鍵欄位。

```
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;
```

現可建立如下的表格方塊：

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

ProductionLine	Time	Event	Comment	Order	Start	End
P1	08:00	4	Start of shift 2	C	03:04	10:27
P2	09:00	4	Start of shift 2	D	07:23	11:43
P1	11:43	5	End of production	-	-	-
P2	11:43	5	End of production	D	07:23	11:43

Join

join 前置詞能夠將載入的表格與現有的具名表格或先前最後建立的資料表聯結。

聯結資料的效果是透過其他欄位或屬性集合延伸目標表格，亦即尚未出現在目標表格中的內容。來源資料集和目標表格之間的任何一般欄位名稱用來理解如何關聯新的傳入記錄。這通常稱為「自然聯結」。Qlik 聯結操作會導致產生的目標表格有比一開始更多或更少的記錄，取決於聯結關聯的獨特性和採用的聯結類型。

有四種聯結類型：

左聯結

左聯結是最常見的聯結類型。例如，若您有交易資料集並想要組合參考資料集，您通常會使用 `Left Join`。您會先載入交易表格，然後載入參考資料集，同時透過 `Left Join` 前置詞聯結到已載入的交易表格。`Left Join` 會將所有交易保持原狀並新增找到相符項目的補充參考資料欄位。

內聯結

若您有兩個資料集，而您只在乎有相符關聯的任何結果，請考慮使用 `Inner Join`。若找不到相符內容，這將會從載入的來源資料和目標表格排除所有記錄。因此，這可能會讓目標表格的記錄比聯結操作發生之前更少。

外聯結

若您需要保留兩個目標記錄和所有傳入記錄，請使用 `Outer Join`。若找不到相符內容，仍會保留每個記錄集合，而來自聯結另一側的欄位將會保持未填入 (null)。

若省略類型關鍵字，預設聯結類型是外部聯結。

右聯結

此聯結類型保留所有即將載入的記錄，同時將透過聯結設為目標的記錄減少為只有在傳入記錄中有相符關聯的記錄。此為合適的聯結類型，有時候用來將已預先載入的記錄表格裁剪為所需的子集。

來自不同聯結操作類型的結果集合範例

DATASETS

Target Table

Trade ID	Asset Class
101533	Fixed Income
606601	Commodities

Incoming Dataset

Trade ID	Exchange
101533	LSE
79052	Hong Kong

OPERATION / **OUTPUT**

LEFT JOIN →

Trade ID	Asset Class	
101533	Fixed Income	LSE
606601	Commodities	

INNER JOIN →

Trade ID	Asset Class	
101533	Fixed Income	LSE

OUTER JOIN →

Trade ID	Asset Class	
101533	Fixed Income	LSE
606601	Commodities	
79052		Hong Kong

RIGHT JOIN →

Trade ID	Asset Class	
101533	Fixed Income	LSE
79052		Hong Kong

> *若在聯結操作的來源和目標之間沒有共同的欄位名稱，聯結將會產生所有列的笛卡兒乘積——這稱為「交叉聯結」。*

來自「交叉聯結」操作的結果集合範例

DATASETS

Target Table

Trade ID	Base Currency	Amount
101533	EUR	1250
606601	EUR	1650

Incoming Dataset

Target Currency	Rate
USD	1.08
GBP	0.84

OPERATION / **OUTPUT**

JOIN (any type) →

Trade ID	Base Currency	Amount	Target Currency	Rate
101533	EUR	1250	USD	1.08
101533	EUR	1250	GBP	0.84
606601	EUR	1650	USD	1.08
606601	EUR	1650	GBP	0.84

語法：

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

引數

引數	描述
tablename	要與所載入表格比較的具名表格。
loadstatement或 selectstatement	已載入表格的 **LOAD** 或 **SELECT** 陳述式。

這些主題可協助您使用此函數:

相關主題

主題	描述
在 *管理資料* 中 **使用 Join 與 Keep 合併表格**	此主題進一步說明「聯結」和「保留」資料集的概念。
Keep (page 78)	`Keep` 載入前置詞類似於 `Join` 前置詞,但這不會合併來源和目標資料集。而是會根據採用的操作類型 (內、外、左或右) 裁剪每個資料集。

範例 1 - 左聯結:以參考資料集豐富目標表格

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 呈現變更記錄的資料集,這載入到名為 `Changes` 的表格中。這包括狀態 ID 索引鍵欄位。
- 呈現變更狀態的第二個資料集,這聯結左側 `Join` 載入前置詞,以載入並合併原始變更記錄。

此左聯結確保變更記錄在新增狀態屬性時保持不變,這會根據共同狀態 ID 找到傳入狀態記錄中的相符內容。

載入指令碼

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

```
10220	2	28/07/2022	06/09/2022	None
10264	1	10/09/2022	17/10/2022	Medium
10116	1	15/04/2022	24/04/2022	None
10187	2	25/06/2022	24/08/2022	Low
] (delimiter is '\t');

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

結果

開啟資料模型檢視器並注意資料模型形狀。僅存在一個去正規化的表格。這合併所有原始變更記錄，並在每個變更記錄上聯結相符的狀態屬性。

產生內部資料模型

變更
變更 ID
狀態 ID
已排程開始日期
已排程結束日期
商務影響
狀態
子狀態

若您在資料模型檢視器中展開預覽視窗，您將會看見組織為表格之完整結果集合的一部分：

資料模型檢視器中的變更表格預覽

變更 ID	狀態 ID	已排程開始日期	已排程結束日期	商務影響	狀態	子狀態
10030	4	19/01/2022	23/02/2022	無	已關閉	已取消
10031	3	20/01/2022	25/03/2022	低	已關閉	已完成
10015	3	04/01/2022	15/02/2022	低	已關閉	已完成
10103	1	02/04/2022	29/05/2022	中等	開啟	未開始
10116	1	15/04/2022	24/04/2022	無	開啟	未開始
10134	1	03/05/2022	08/07/2022	低	開啟	未開始

變更 ID	狀態 ID	已排程開始日期	已排程結束日期	商務影響	狀態	子狀態
10264	1	10/09/2022	17/10/2022	中等	開啟	未開始
10040	1	29/01/2022	22/04/2022	無	開啟	未開始
10323	1	08/11/2022	26/11/2022	高	開啟	未開始
10187	2	25/06/2022	24/08/2022	低	開啟	已開始
10185	2	23/06/2022	08/09/2022	無	開啟	已開始
10220	2	28/07/2022	06/09/2022	無	開啟	已開始
10326	2	11/11/2022	05/12/2022	無	開啟	已開始
10138	2	07/05/2022	03/08/2022	無	開啟	已開始
10334	2	19/11/2022	06/02/2023	低	開啟	已開始

由於狀態表格中的第五列 (狀態 ID:「5」, 狀態:「已關閉」, 子狀態:「已過時」) 沒有對應至變更表格中的任何記錄, 此列的資訊不會出現在上述的結果集合中。

返回資料載入編輯器。載入資料並開啟工作表。建立新的表格並將此欄位新增為維度: `Status`。

新增此量值:

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

現在您可以依狀態檢查變更數量。

結果表格

狀態	=Count([Change ID])
開啟	12
已關閉	3

範例 2 – 內聯結:僅合併相符記錄

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 呈現變更記錄的資料集, 這載入到名為 `Changes` 的表格中。
- 呈現變更記錄的第二個資料集, 這源自來源系統 `JIRA`。這聯結 `Inner Join` 載入前置詞, 以載入並合併原始記錄。

此 `Inner Join` 確保僅保留在兩個資料集中找到的五個變更記錄。

載入指令碼

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

JIRA_changes:
Inner Join (Changes)
Load
  [Ticket ID] AS [Change ID],
  [Source System]
inline
[
Ticket ID	Source System
10000	JIRA
10030	JIRA
10323	JIRA
10134	JIRA
10334	JIRA
10220	JIRA
20000	TFS
] (delimiter is '\t');
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- Source System
- Change ID
- Business Impact

現在您可以檢查五個產生的記錄。來自 Inner Join 的結果表格只包含在兩個資料集中具有相符資訊的記錄。

結果表格

來源系統	變更 ID	商務影響
JIRA	10030	無
JIRA	10134	低
JIRA	10220	無
JIRA	10323	高
JIRA	10334	低

範例 3 – 外聯結:合併重疊記錄集合

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 呈現變更記錄的資料集,這載入到名為 `Changes` 的表格中。
- 呈現變更記錄的第二個資料集,這源自來源系統 `JIRA`,這聯結 `Outer Join` 載入前置詞,以載入並合併原始記錄。

這確保保留來自兩個資料集的所有重疊變更記錄。

載入指令碼

```
// 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');
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- Source System
- Change ID
- Business Impact

現在您可以檢查 10 個產生的記錄。

結果表格

來源系統	變更 ID	商務影響
JIRA	10030	無
JIRA	10134	低
JIRA	10220	無
JIRA	10323	-
JIRA	10334	低
JIRA	10597	-
-	10015	低
-	10031	低
-	10040	無
-	10138	無

範例 4 – 右聯結：按第二主要資料集裁剪目標表格

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 呈現變更記錄的資料集，這載入到名為 Changes 的表格中。
- 第二個資料集呈現源自來源系統 Teamwork 的變更記錄。這與 Right Join 載入前置詞聯結，以透過原始記錄載入並合併。

若目標表格沒有相符的 Change ID，這確保僅保留 Teamwork 變更記錄，同時不失去任何 Teamwork 記錄。

載入指令碼

```
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');
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- Source System
- Change ID
- Business Impact

現在您可以檢查五個產生的記錄。

結果表格

來源系統	變更 ID	商務影響
團隊合作	10015	低
團隊合作	10031	低
團隊合作	10040	無
團隊合作	10103	中等
團隊合作	50231	-

Keep

keep 前置詞類似於 **join** 前置詞。與 **join** 前置詞一樣，它會比較已載入的表格與現有具名表格或最後一個先前建立的資料表格，而不會聯結已載入的表格與現有表格，它會在將一個或兩個表格儲存在 Qlik Sense 中之前，根據表格資料的交集進行減少。進行比較相當於在所有通用欄位上進行自然聯結，也就是如同對應聯結中的方法。不過，兩個表格並未聯結，而是以兩個個別的具名表格保留在 Qlik Sense 中。

語法：

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

keep 前置詞前面可以加上 **inner**、**left** 或 **right** 前置詞的其中一個。

Qlik Sense 指令碼語言中明確的 **join** 前置詞會執行兩個表格的完整聯結。結果會是一個表格。在許多情況下，這樣的聯結會產生非常大的表格。Qlik Sense 的其中一個主要特色是可在多個表格之間建立關聯而不是聯結，這能夠大幅減少記憶體使用、提高處理速度，並提供極大的彈性。因此，在 Qlik Sense 指令碼中通常應避免明確聯結。keep 功能是專門用於減少需要使用明確聯結的情況。

引數：

引數

引數	描述
tablename	要與所載入表格比較的具名表格。
loadstatement或 selectstatement	已載入表格的 **LOAD** 或 **SELECT** 陳述式。

範例：

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

Left Keep SELECT * from table1;

tab1:

LOAD * from file1.csv;
```

```
tab2:

LOAD * from file2.csv;

.. .. ..

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

Left

Join 及 **Keep** 前置詞前面可以加上 **left** 前置詞。

如果加在 **join** 之前，會指定應該使用 left join。產生的表格只會包含原始資料表中的欄位值組合，其中第一個表格會呈現連結欄位值。如果加在 **keep** 之前，會指定第二個原始資料表應先減少到與第一個表格的共同交集之後，然後才儲存到 Qlik Sense 中。

*您在尋找使用相同名稱的字串函數嗎？請參閱：**Left (page 1376)*

語法：

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

引數：

引數

引數	描述
tablename	要與所載入表格比較的具名表格。
loadstatement或 selectstatement	已載入表格的 **LOAD** 或 **SELECT** 陳述式。

範例

載入指令碼

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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 ];
```

結果

結果表格

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

解釋

此範例展示左側聯結輸出，其中只會聯結出現在第一個 (左) 表格中的值。

對應

mapping 前置詞可用來建立對應表，例如，對應表可用於在指令碼執行期間取代欄位值和欄位名稱。

語法：

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

mapping 前置詞可以放在 **LOAD** 或 **SELECT** 陳述式的前面，而且將載入陳述式的結果另存為對應表格。對應提供有效的方法在指令碼執行期間替換欄位值，例如，將 US、U.S. 或 America 替換為 USA。對應表由兩個資料行組成，第一個包含比較值，第二個包含所需的對應值。對應表會暫時儲存在記憶體中，並且在指令碼執行後自動捨棄。

例如，使用 **Map ... Using** 陳述式、**Rename Field** 陳述式、**Applymap()** 函數或 **Mapsubstring()** 函數，即可存取對應表格的內容。

範例：

在此範例中，我們載入銷售人員清單，並使用國家/地區代碼表示其居住國家/地區。我們使用表格將國家/地區代碼對應至國家/地區，將國家/地區代碼取代為國家/地區名稱。對應表格中僅定義三個國家/地區，其他國家/地區代碼對應至 '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';
```

產生的表格如下所示：

Mapping table

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

合併

Merge 前置詞可新增至指令碼中的任何 **LOAD** 或 **SELECT** 陳述式，以指定載入的表格應合併至另一個表格中。這也能指定應在部分載入中執行此陳述式。

常見使用情況是當您載入變更記錄並想要用來將 inserts、updates 和 deletes 套用至現有表格時。

為了讓部分載入正常運作，必須在觸發部分載入之前以資料開啟應用程式。

使用**載入**按鈕執行部分載入。您也可以使用 Qlik Engine JSON API。

語法：

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

引數：

引數

引數	描述
only	選用的限定詞，表示只應在部分載入期間執行該陳述式。陳述式在一般 (非部分) 載入期間忽略。
SequenceNoField	包含時間戳記的欄位名稱或定義操作順序的序號。
SequenceNoVar	獲得指派合併表格的 SequenceNoField 最大值的變數名稱。
ListOfKeys	以逗號分隔、指定主要金鑰的欄位名稱清單。
Operation	LOAD 陳述式的第一個欄位必須包含操作作為文字字串：'Insert'、'Update' 或 'Delete'。也接受 'i'、'u' 和 'd'。

一般功能

在一般 (非部分) 載入期間，**Merge LOAD** 建構會作為一般 **Load** 陳述式運作，但具有附加功能，可移除較舊過時記錄和標記要刪除的記錄。**LOAD** 陳述式的第一個欄位必須保留關於操作的資訊：Insert、Update 或 Delete。

對於每個載入的記錄，會比較記錄識別碼與先前載入的記錄，只會保留最新記錄 (根據序號)。若以 Delete 標記最新記錄，將不會保留任何內容。

目標表格

要修改哪個表格由欄位組決定。若具有相同欄位組 (第一個欄位除外；操作) 的表格已存在，這將會是要修改的相關表格。或者，**串連**前置詞可用來指定表格。若未決定目標表格，**Merge LOAD** 建構的結果會儲存在新的表格中。

若使用串連前置詞，產生的表格具有一組對應至現有表格聯集和要合併之輸入的欄位。因此，相較於作為要合併之輸入使用的變更記錄，目標表格可能會取得更多欄位。

部分載入會進行與完整載入相同的事項。差異是部分載入很少建立新的表格。除非您已使用 **Only** 子句，否則永遠存在具有相同欄位組 (來自先前指令碼執行) 的目標表格。

序號

若載入的變更記錄是累積記錄，亦即這包含已經載入的變更，則參數 SequenceNoVar 可用於 **Where** 子句，以限制輸入資料量。然後可以讓 **Merge LOAD** 僅載入記錄，其中欄位 SequenceNoField 大於 SequenceNoVar。完成後，**Merge LOAD** 會將新的值指派至 SequenceNoVar，並可在 SequenceNoField 欄位中看見最大值。

操作

Merge LOAD 可以有比目標表格更少的欄位。不同的操作以不同的方式處理遺失的欄位：

插入：在 **Merge LOAD** 中遺失但存在於目標表格的欄位，會在目標表格中取得 NULL。

刪除：遺失的欄位不會影響結果。仍會刪除相關記錄。

更新：列於 **Merge LOAD** 的欄位會在目標表格中更新。不會變更遺失的欄位。這表示下列兩個陳述式不相同：

- 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 ...;

第一個陳述式更新列出的記錄並將 F2 變更為 NULL。第二個不會變更 F2，但會在目標表格中留下值。

範例

範例 1：與指定的表格簡單合併

在此範例中，載入了名為 Persons 的內嵌表格，其中含有三列。然後 **Merge** 會變更表格如下：

- 新增列 *Mary, 4*。
- 刪除列 *Steven, 3*。
- 將數字 *5* 指派至 *Jake*。

執行 **Merge** 後，*LastChangeDate* 變數會設定為 *ChangeDate* 欄中的最大值。

載入指令碼

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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
];
```

結果

在 **Merge Load** 之前，產生的表格顯示如下：

Resulting table

Name	Number
Jake	3
Jill	2
Steven	3

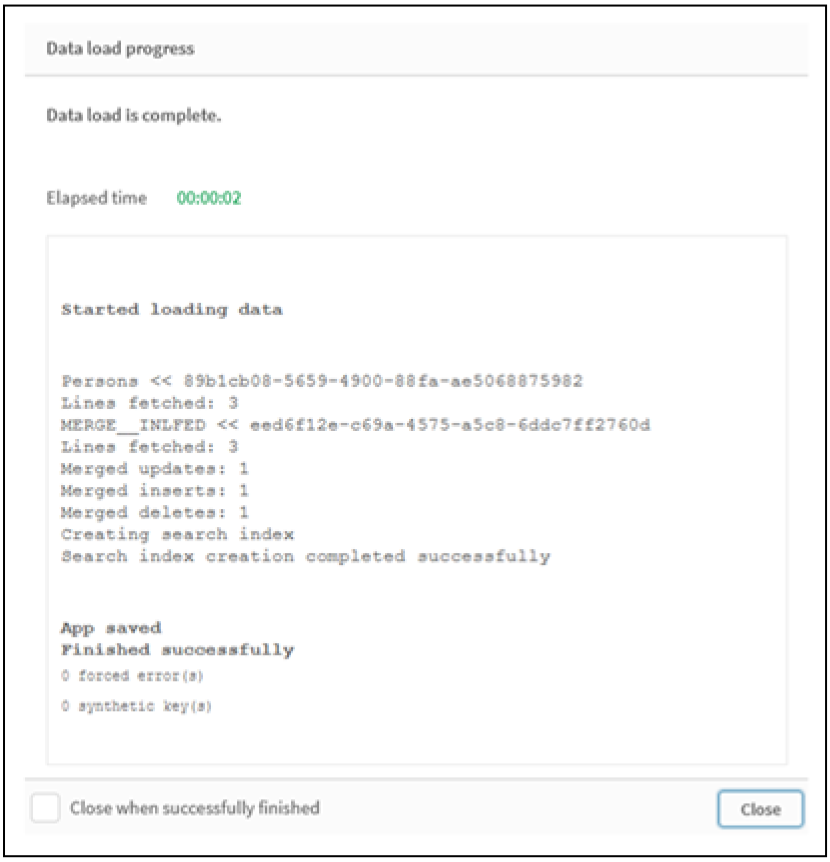
在 **Merge Load** 之後，表格顯示如下：

Resulting table

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

載入資料時，**資料載入進度** 對話方塊會顯示執行的操作：

資料載入進度對話方塊



範例 2：含有遺失欄位的資料載入指令碼

在此範例中，會載入與上述相同的資料，但現在含有每個人的 ID。

Merge 會變更表格如下：

- 新增列 *Mary, 4*。
- 刪除列 *Steven, 3*。
- 將數字 *5* 指派至 *Jake*。
- 將數字 *6* 指派至 *Jill*。

載入指令碼

在此我們使用兩個 **Merge Load** 陳述式，一個用於「插入」和「刪除」，第二個用於「更新」。

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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
];
```

結果

在 **Merge Load** 陳述式之後，表格顯示如下：

Resulting table

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

請注意，第二個 **Merge** 陳述式不包括欄位 **Name**，因此尚未變更名稱。

範例 3：資料載入指令碼 - 使用 Where 子句與 ChangeDate 的部分載入

在以下範例中，**Only** 引數指定只能在部分載入期間執行 **Merge** 命令。會根據先前擷取的 LastChangeDate 篩選更新。完成 **Merge** 之後，會向 LastChangeDate 變數指派合併期間處理的 ChangeDate 欄的最大值。

載入指令碼

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

NoConcatenate

NoConcatenate 前置詞會強制將具有相同欄位集的兩個載入表格在自動串連時視為兩個個別的內部表格。

語法：

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

根據預設，若表格載入後包含相同數量的欄位，將欄位名稱對應稍早載入到指令碼中的表格時，Qlik Sense 即會自動串連這兩個表格。即使第二個表格的名稱不同，還是會發生此情況。

但是，若指令碼前置詞 `NoConcatenate` 在 LOAD 陳述式前納入，或是選取第二個表格的陳述式，即會分開載入這兩個表格。

NoConcatenate 的一般使用案例即是當您必須建立臨時的表格副本，才能針對該副本執行一些臨時轉換，同時保留原始資料的副本。NoConcatenate 確保您可以在不隱含新增回來源表格的情況下，製作該副本。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
Source: LOAD A,B from file1.csv; CopyOfSource: NoConcatenate LOAD A,B resident Source;	已載入將 A 和 B 作為量值的表格。第二個具有相同欄位的表格係使用 NoConcatenate 變數分開載入。

範例 1 – 隱含串連

載入指令碼和結果

概覽

在此範例中，您會以連續順序新增兩個載入指令碼。

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 傳送至名為 Transactions 之表格中具有日期和金額的初始資料集。

第一個載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- id
- date
- amount

第一個結果表格

id	日期	金額
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

第二個載入指令碼

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 具有相同欄位的第二個資料集傳送至名為 Sales 的表格中。

```
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
];
```

結果

載入資料並前往表格。

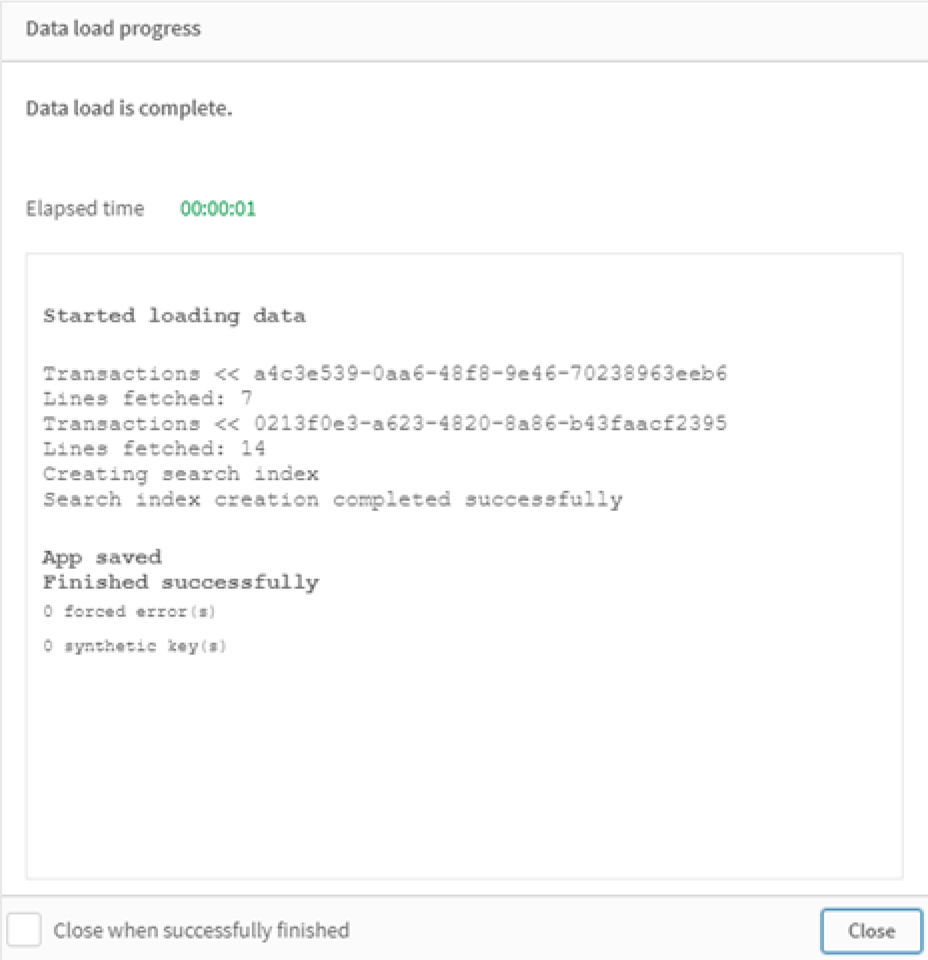
第二個結果表格

id	日期	金額
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

指令碼執行時，因為兩個資料集共用相同數量的欄位且欄位名稱相同，所以 `Sales` 表格會隱含串連至現有的 `Transactions` 表格上。雖然第二個表格名稱標記嘗試命名結果集 ‘`Sales`’，還是會發生此情況。

您可以透過觀察**資料載入進度**記錄，發現銷售資料集隱含串連。

資料載入進度記錄顯示交易資料隱含串連。



範例 2 – 使用案例情境

載入指令碼和結果

概覽

在此案例情境，您有：

- 具有以下項目的交易資料集：
  - id
  - 日期
  - 金額(單位為 GBP)
- 具有以下項目的貨幣表格：
  - USD 至 GBP 的轉換匯率
- 具有以下項目的第二個交易資料集：
  - id

- 日期
- 金額(單位為 USD)

您會以連續順序載入五個指令碼。

- 第一個載入指令碼包含具有日期和 GBP 金額的初始資料集,其傳送至名為 `Transactions` 的表格中。
- 第二個載入指令碼包含:
  - 第二個具有日期和 USD 金額的資料集,其傳送至名為 `Transactions_in_USD` 的表格中。
  - `noconcatenate` 前置詞放置在 `Transactions_in_USD` 資料集的 LOAD 陳述式前,以預防隱含串連的情況。
- 第三個載入指令碼包含用於建立 `Transactions_in_USD` 表格中 GBP 和 USD 之間貨幣匯率的 `join` 前置詞。
- 第四個載入指令碼包含將 `Transactions_in_USD` 新增至初始 `Transactions` 表格的 `concatenate` 前置詞。
- 第五個載入指令碼包含將其資料串連至 `Transactions` 表格之 `Transactions_in_USD` 表格移除的 `drop table` 陳述式。

第一個載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- id
- date
- amount

第一個載入指令碼結果

id	日期	金額
1	12/30/2018	23.56
2	12/07/2018	556.31
3	12/16/2018	5.75

id	日期	金額
4	12/22/2018	125.00
5	12/22/2018	484.21
6	12/22/2018	59.18
7	12/23/2018	177.42

表格顯示具有 GBP 金額的初始資料集。

第二個載入指令碼

```
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
];
```

結果

載入資料並前往表格。

第二個載入指令碼結果

id	日期	金額
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
12	01/15/2019	4.56

id	日期	金額
13	01/16/2019	90.24
14	01/18/2019	19.32

您會看到 `Transactions_in_USD` 表格的第二個資料集已新增。

第三個載入指令碼

此載入指令碼將 USD 至 GBP 的貨幣匯率聯結至 `Transactions_in_USD` 表格。

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

結果

載入資料並前往資料模型檢視器。選取 `Transactions_in_USD` 表格，然後您會看到每筆現有記錄都有 0.7 的「匯率」欄位值。

第四個載入指令碼

使用 Resident 載入時，此載入指令碼即會在金額轉換至 USD 後將 `Transactions_in_USD` 表格串連至 `Transactions` 表格。

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

結果

載入資料並前往表格。您會看到第八至第十四行中 GBP 金額的新項目。

第四個載入指令碼結果

id	日期	金額
1	12/30/2018	23.56
2	12/07/2018	556.31
3	12/16/2018	5.75
4	12/22/2018	125.00
5	12/22/2018	484.21
6	12/22/2018	59.18

id	日期	金額
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

第五個載入指令碼

此載入指令碼會捨棄第四個載入指令碼結果表格中的複製項目，只留下金額為 GBP 的項目。

```
drop tables Transactions_in_USD;
```

結果

載入資料並前往表格。

第五個載入指令碼結果

id	日期	金額
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

id	日期	金額
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

載入第五個載入指令碼後，結果表格會顯示存在兩個交易資料集中完整的十四筆交易；但是，交易 8 至 14 會將其金額轉換為 GBP。

如果我們移除用於第二個載入指令碼中 `Transactions_in_USD` 前的 `NoConcatenate` 前置詞，指令碼將會失敗並出現錯誤：「找不到表格 'Transactions_in_USD'」。這是因為 `Transactions_in_USD` 表格已自動串連至原始的 `Transactions` 表格上。

Only

Only 指令碼關鍵字用作一個彙總函數，或在部分重新載入前置詞 **Add**、**Replace** 和 **Merge** 中作為語法的一部分。

Outer

明確的 **Join** 前置詞可接在前置詞 **Outer** 之後，以指定外部聯結。在外部聯結中，會在兩個表格間產生所有組合。產生的表格會因此包含原始資料表格中的欄位值組合，其中一或兩個表格會呈現連結欄位值。**Outer** 關鍵字是選用項目，是未指定聯結前置詞時使用的預設連結。

語法：

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

引數：

引數

引數	描述
tablename	要與所載入表格比較的具名表格。
loadstatement或 selectstatement	已載入表格的 **LOAD** 或 **SELECT** 陳述式。

範例

載入指令碼

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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 ];
```

結果表格

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

解釋

在此範例中，兩個表格 Table1 和 Table2 合併為具有 Table1 標籤的單一表格。在這樣的情況下，**outer** 前置詞通常用來將數個表格聯結為單一表格，以透過單一表格的值執行彙總。

部分載入

完整載入永遠從刪除現有資料模型中的所有表格開始，然後執行載入指令碼。

部分載入將不會這麼做。反而這會保留資料模型中的所有表格，然後僅執行前面有 **Add**、**Merge** 或 **Replace** 前置詞的 **Load** 和 **Select** 陳述式。其他資料表格不會受到命令影響。**only** 引數表示只應在部分載入期間執行陳述式，且應在完整載入期間忽略。下列表格概述部分和完整載入的陳述式執行。

陳述式	完整載入	部分載入
Load ...	陳述式將會執行	陳述式將不會執行
新增/取代/合併載入 ...	陳述式將會執行	陳述式將會執行
新增/取代/合併僅載入 ...	陳述式將不會執行	陳述式將會執行

相較於完整載入，部分載入有幾個優點：

- 更快速，因為僅需載入最近變更的資料。具有大型資料集，差異顯著。
- 耗用的記憶體更少，因為載入的資料較少。
- 更可靠，因為查詢來源資料的執行速度更快，降低網路問題風險。

為了讓部分載入正常運作，必須在觸發部分載入之前以資料開啟應用程式。

使用**載入**按鈕執行部分載入。您也可以使用 Qlik Engine JSON API。

限制

若有命令參考存在於完全載入期間、而不存在於部分載入期間的表格，則部分載入將會失敗。

範例

範例命令

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

其中 <Table_removed_after_full_reload> 是存在於完全載入、而不存在於部分載入的表格。

因應措施

作為因應措施，您可以使用下列 if 陳述式處理命令：

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

部分載入可以從資料中移除值。不過，這不會反映在相異值的清單中，這是內部維護的表格。因此，在部分載入之後，清單將包含自上次完整載入以來存在於欄位中的所有相異值，這可能會在部分載入後多於目前存在的內容。這會影響 FieldValueCount() 和 FieldValue() 函數的輸出。FieldValueCount() 可能會傳回大於目前欄位值數量的數字。

範例

範例 1

載入指令碼

將範例指令碼新增到您的應用程式中並進行部分載入。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

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

結果

Resulting table

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

Num	Count(Num)
16	1
17	1
18	1
19	1
20	1

解釋

只有在部分載入期間才會執行陳述式。若省略「distinct」前置詞，**Num** 欄位的計數將會隨之後每次部分載入而增加。

範例 2

載入指令碼

將範例指令碼新增到您的應用程式中。進行完整載入並檢視結果。接下來，進行部分載入並檢視結果。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

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

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

結果

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

解釋

第一個表格在完整載入期間載入，而第二個表格只是在部分載入期間取代了第一個表格。

Replace

Replace 指令碼關鍵字可用作字串函數，或在部分重新載入中用作前置詞。

Replace

Replace 前置詞 可新增至指令碼中的任何 **LOAD** 或 **SELECT** 陳述式，以指定載入的表格應取代另一個表格。這也能指定應在部分載入中執行此陳述式。**Replace** 前置詞也能用於 **Map** 陳述式。

為了讓部分載入正常運作，必須在觸發部分載入之前以資料開啟應用程式。

使用**載入**按鈕執行部分載入。您也可以使用 Qlik Engine JSON API。

語法：

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

```
Replace [only] mapstatement
```

在一般 (非部分) 載入期間， **Replace LOAD** 建構將會作為一般 **LOAD** 陳述式運作，但以 **Drop Table** 作為前置詞。首先，將會捨棄舊的表格，然後將會產生記錄並儲存為新表格。

若使用 **Concatenate** 前置詞，或者若存在具有相同欄位集的表格，這將會是要捨棄的相關表格。否則，沒有要捨棄的表格，且 **Replace LOAD** 建構將會與一般 **LOAD** 相同。

部分載入將會進行相同事項。唯一的差異是永遠有來自先前指令碼執行的表格以供捨棄。**Replace LOAD** 建構將永遠先捨棄舊的表格，然後建立新的表格。

Replace Map...Using 陳述式也會使得對應在部分指令碼執行期間進行。

引數：

引數

引數	描述
only	選用的限定詞，表示只應在部分載入期間執行該陳述式。這應在一般 (非部分) 載入期間忽略。

範例與結果：

範例	結果
`Tab1:` `Replace LOAD * from File1.csv;`	在正常與部分重新載入期間，一開始就會捨棄 Qlik Sense 表格 Tab1。之後，會從 File1.csv 載入新資料並儲存到 Tab1。
`Tab1:` `Replace only LOAD * from File1.csv;`	在正常重新載入期間，會忽略此陳述式。 在部分重新載入期間，一開始就會捨棄之前命名為 Tab1 的所有 Qlik Sense 表格。之後，會從 File1.csv 載入新資料並儲存到 Tab1。

`Tab1:`
`LOAD a,b,c from File1.csv;`
`Replace LOAD a,b,c from File2.csv;`
	在正常重新載入期間，會先將 File1.csv 檔案讀取到 Qlik Sense 表格 Tab1，但是會立即捨棄該檔案，並以從 File2.csv 載入的新資料取代。File1.csv 中的所有資料會遺失。
	
	在部分重新載入期間，一開始就會捨棄整個 Qlik Sense 表格 Tab1。之後，會以從 File2.csv 載入的新資料取代。

`Tab1:`
`LOAD a,b,c from File1.csv;`
`Replace only LOAD a,b,c from File2.csv;`
	在正常重新載入期間，會從 File1.csv 載入資料並儲存到 Qlik Sense 表格 Tab1。捨棄 File2.csv。
	
	在部分重新載入期間，一開始就會捨棄整個 Qlik Sense 表格 Tab1。之後，會以從 File2.csv 載入的新資料取代。File1.csv 中的所有資料會遺失。

Right

Join 及 **Keep** 前置詞前面可以加上 **right** 前置詞。

如果加在 **join** 之前，會指定應該使用 right join。產生的表格只會包含原始資料表格中的欄位值組合，其中第二個表格會呈現連結欄位值。如果加在 **keep** 之前，會指定第一個原始資料表應先減少到與第二個表格的共同交集之後，然後才儲存到 Qlik Sense 中。

*您在尋找使用相同名稱的字串函數嗎？ 請參閱：**Right (page 1383)*

語法：

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

引數:

引數

引數	描述
tablename	要與所載入表格比較的具名表格。
loadstatement或 selectstatement	已載入表格的 **LOAD** 或 **SELECT** 陳述式。

範例

載入指令碼

將範例指令碼新增至您的應用程式並予以執行。若要查看結果,將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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 ];
```

結果

結果表格

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

解釋

此範例展示右側聯結輸出,其中只會聯結出現在第二個 (右) 表格中的值。

Sample

LOAD 或 **SELECT** 陳述式的 **sample** 前置詞可用來載入資料來源的隨機記錄樣本。

語法:

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

評估的運算式未定義載入到 Qlik Sense 應用程式的資料集記錄百分比,但會定義讀取為載入到應用程式之每筆記錄的可能性。換言之,指定 `p = 0.5` 的值不表示會載入 50% 的記錄總數,相反地,每筆記錄有 50% 載入到 Qlik Sense 應用程式的機會。

引數

引數	描述
p	評估為大於 0 且小於或等於 1 的數字所用的任意運算式。數字指示讀取指定記錄的可能性。 所有記錄都會予以讀取,不過只有其中一部分會載入 Qlik Sense。

什麼情況下使用

當您想從大型表格取得樣本資料，以便了解資料特性、分佈情況或欄位內容時，樣本很實用。顯示資料子集時，資料會加速載入並允許更快速的指令碼測試。不同於 `First`，`Sample` 函數會顯示整個表格的資料，而不是僅限於前幾列。這在某些情況下會提供更準確的資料呈現。

下列範例顯示使用 `Sample` 指令碼前置詞的兩種可能案例：

```
Sample 0.15 SQL SELECT * from Longtable;

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

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 內嵌表格的樣本

載入指令碼和結果

概覽

在此範例中，指令碼會從包含七筆記錄的資料集，將資料樣本集載入到內嵌表格中名為 `Transactions` 的表格。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- id
- amount

新增下列量值:

=sum(amount)8

結果表格

id	日期	=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

在用於此範例中的載入反覆項目中,讀取到全部的七筆記錄,但只有四筆記錄載入到資料表格中。任何重新執行的載入可能導致不同數字,並將不同的記錄集載入到應用程式中。

範例 2 – 自動產生表格的樣本

載入指令碼和結果

概覽

在此使用 Autogenerate 的範例中,100 筆記錄的資料集是使用欄位 date、id 及 amount 所建立。但是,使用了 Sample 前置詞並有 0.1 的值。

載入指令碼

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

Autogenerate(100);
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- id
- amount

新增下列量值:

結果表格

id	日期	=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

在此範例使用的載入反覆項目中,有七筆記錄是從建立的資料集載入。任何重新執行的載入可能再度導致不同數字,並將不同的記錄集載入到應用程式中。

Semantic

semantic 載入前置詞建立可用於 Qlik Sense 的特殊欄位類型,以連線並管理關係資料,例如樹狀結構、自我參考的父子結構化資料和/或可描述為圖表的資料。

請注意,semantic 載入的運作方式可類似於 *Hierarchy (page 61)* 和 *HierarchyBelongsTo (page 63)* 前置詞。所有三個前置詞可在有效的前端解決方案作為組建區塊使用以周遊關係資料。

語法:

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

語意載入預期輸入內容確切為三個或四個欄位寬,以及嚴格定義每個排序欄位呈現的內容,如下表所示:

語意載入欄位

欄位名稱	欄位描述
第 1 個欄位:	此標記呈現兩個有關係之物件中的第一個。
第 2 個欄位:	此標記將用來描述第一個和第二個物件之間的「向前」關係。若第一個物件是子項,而第二個物件是父項,則您可以建立表示「父項」或「其父項」的關係索引標籤,如同您追蹤從子項到父項的關係。
第 3	此標記呈現兩個有關係之物件中的第二個。

欄位名稱	欄位描述
個欄位:	
第 4 個欄位:	此欄位為選用。此標記描述第一個和第二個物件之間的「向後」或「反向」關係。若第一個物件是子項，而第二個物件是父項，則關係索引標籤可表示「子項」或「其子項」，如同您追蹤從父項到子項的關係。若您沒有新增第四個欄位，則第二個欄位標記將用來以任一方向描述關係。在此情況下，會自動新增箭頭符號作為標記的一部分。

下列代碼是 `semantic` 前置詞的範例。

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

這為第三個欄位加上標籤的方式與第一個欄位相同，是允許且常見的做法。這建立自我參考查閱，讓您可以一次追蹤從物件到差一個關係步驟的相關物件。若第 3 個欄位不具相同名稱，則最終結果將會是從物件到差一個步驟之直接關係鄰近項目的簡單查閱，亦即小規模實際使用的輸出。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

相關函數

函數	互動
Hierarchy (page 61)	Hierarchy 載入前置詞用來以父子和其他類圖表資料結構分割並組織節點以及轉換為表格。
HierarchyBelongsTo (page 63)	HierarchyBelongsTo 載入前置詞用來定位並組織父子和其他類圖表資料結構的上階以及轉換為表格。

範例 - 建立特殊欄位以使用語意前置詞連接關係

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 呈現地理關係記錄的資料集，這載入到名為 GeographyTree 的表格中。
    - 每個項目的 ID 位於行的開頭，而 ParentID 位於行的結尾。
- semantic 前置詞將會新增一個加上標籤的特殊行為欄位 Relation。

載入指令碼

```
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;
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度。

- Id
- Geography

然後，以 Relation 建立篩選窗格作為維度。按一下**完成編輯**。

結果表格

Id	地理
1	世界
2	歐洲
3	亞洲
4	北美
5	南美
6	英國
7	德國
8	Sweden
9	南韓
10	北韓
11	中國
12	倫敦
13	伯明罕

篩選窗格

關係
子項
父項

從表格中的 `Geography` 維度按一下**歐洲**並從篩選窗格中的 `Relation` 維度按一下**子項**。注意表格中預期的結果:

結果表格顯示歐洲的「子項」

Id	地理
6	英國
7	德國
8	Sweden

再次按一下**子項**將會顯示屬於英國「子項」的地點,亦即更進一步。

結果表格顯示英國的「子項」

Id	地理
12	倫敦
13	伯明罕

Unless

unless 前置詞及後置詞用來建立條件式子句，其可決定是否應該評估陳述式或結束子句。它可視為完整 **if..end if** 陳述式的精簡替代選項。

語法：

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

只有在 **condition** 評估為 False 時，才會執行 **statement** 或 **exitstatement**。

unless 前置詞可用在已有一或數個其他陳述式的陳述式上，包含其他 **unless** 或 **when** 前置詞。

引數

引數	描述
condition	評估為 True 或 False 的邏輯運算式。
statement	除了控制陳述式之外的任何 Qlik Sense 指令碼陳述式。
exitstatement	**exit for**、**exit do** 或 **exit sub** 子句，或者 **exit script** 陳述式。

什麼情況下使用

`Unless` 陳述式會傳回布林值結果。一般來說，當使用者想要有條件地載入或排除指令碼部份時，此函數類型會作為條件使用。

下列行顯示三個可能如何使用 `Unless` 函數的範例：

```
exit script unless A=1;

unless A=1 LOAD * from myfile.csv;

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

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – Unless 前置詞

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 建立變數 A, 其會有 1 的值。
- 資料集載入到名為「交易」的表格中, 除非變數 A = 2。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- id
- date
- amount

結果表格

id	日期	金額
1	08/30/2018	23.56
2	09/07/2018	556.31
3	09/16/2018	5.75
4	09/22/2018	125.00

id	日期	金額
5	09/22/2018	484.21
6	09/22/2018	59.18
7	09/23/2018	177.42

因為變數 A 在指令碼開始時被指派 1 的值，所以評估了追蹤 `Unless` 前置詞的條件，並傳回 `FALSE` 的結果。因此，指令碼還是會執行 `Load` 陳述式。在結果表格中，可看到 `Transactions` 表格的所有記錄。

若此變數值設為等於 2，即不會將任何資料載入到資料模型中。

範例 2 – Unless 後置詞

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼開始於將初始資料集載入到名為 `Transactions` 的表格中。然後該指令碼會終止，除非 `Transactions` 表格中的記錄少於 10 筆。

若此條件不會讓指令碼終止，更多的交易集就會串連至 `Transactions` 表格中，並且重覆此流程。

載入指令碼

```
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 ;
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date
- amount

結果表格

id	日期	金額
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

id	日期	金額
11	10/09/2018	312.00
12	10/15/2018	4.56
13	10/16/2018	90.24
14	10/18/2018	19.32

在載入指令碼的三個資料集中，每一個都有七筆記錄。

第一個資料集(從交易 id 1 到 7) 會載入到應用程式中。Unless 條件會評估 Transactions 表格是否少於 10 列。這評估為 TRUE，而且第二個資料集(從交易 id 8 到 14) 會載入到應用程式中。第二個 Unless 條件會評估 Transactions 表格中是否少於 10 筆記錄。這評估為 FALSE，指令碼也會因此終止。

範例 3 – 多個 Unless 前置詞

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

在此範例中，包含一筆交易的資料集已建立為名為 Transactions 的表格。接著觸發「for」迴圈，其中會有兩個巢狀 Unless 陳述式評估：

1. 除非 Transactions 表格中有超過 100 筆記錄
2. 除非 Transactions 表格中的記錄數為 6 的倍數

如果這些條件是 FALSE，即會產生另外七筆記錄並串連至現有的 Transactions 表格中。在兩筆交易其中一筆傳回 TRUE 的值前，此流程會重覆進行。

載入指令碼

```
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
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：id。

結果表格

id
0
1
2
3
4
5
其他 30 多列

「For」迴圈發生的巢狀 Unless 陳述式會評估以下項目：

1. `Transactions` 表格中是否超過 100 列？
2. `Transactions` 表格中的記錄總數是否為 6 的倍數？

每次 Unless 陳述式傳回 `FALSE` 的值時，即會產生另外七筆記錄並串連至現有的 `Transactions` 表格中。

這些陳述式會傳回 `FALSE` 的值五次，此時 `Transactions` 表格中共有 36 列資料。

接下來，第二個 `Unless` 陳述式會傳回 `TRUE` 的值，因此再也不會執行此項之後的 LOAD 陳述式。

When

when 前置詞及後置詞用來建立條件式子句，其可決定是否應該執行陳述式或結束子句。它可視為完整 **if..end if** 陳述式的精簡替代選項。

語法：

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

傳回的資料類型： 布林

在 Qlik Sense 中，布林值 true 值以 -1 代表，而 false 值以 0 代表。

只有在條件評估為 TRUE 時，才會執行 **statement** 或 **exitstatement**。

`Unless` 前置詞可用在已有一或數個其他陳述式的陳述式上，包含其他 `When` 或 `When` 前置詞。

什麼情況下使用

`When` 陳述式會傳回布林值結果。一般來說，當使用者想要載入或排除指令碼部份時，此函數類型會作為條件使用。

引數

引數	描述
condition	評估為 TRUE 或 FALSE 的邏輯運算式
statement	除了控制陳述式之外的任何 Qlik Sense 指令碼陳述式。
exitstatement	**exit for**、**exit do** 或 **exit sub** 子句，或者 **exit script** 陳述式。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
`exit script when A=1;`	當陳述式 `A=1` 評估為 TRUE 時，即會停止指令碼。
`when A=1 LOAD * from myfile.csv;`	當陳述式 `A=1` 評估為 TRUE 時，即會載入 `myfile.csv`。
`when A=1 unless B=2 drop table Tab1;`	當指令碼 `A=1` 評估為 TRUE 時，如果 `B=2` 評估為 FALSE，即會捨棄 `Tab1` 表格。

範例 1 – When 前置詞

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 傳送至名為「`Transactions`」之表格中具有日期和金額的資料集。
- `Let` 陳述式表示變數 `A` 已建立並具有 `1` 的值。
- `When` 條件提供如果 `A` 等於 `1`，即會繼續載入指令碼的條件。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date
- amount

結果表格

id	日期	金額
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

因為變數 A 在指令碼開始時被指派 1 的值，所以評估了追蹤 when 前置詞的條件，並傳回 TRUE 的結果。因為其傳回 TRUE 的結果，所以指令碼還是會執行 LOAD 陳述式。可看到結果表格中的所有記錄。

若此變數值設為不等於 1 的任意值，即不會將任何資料載入到資料模型中。

範例 2 – When 後置詞

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 傳送至名為「Transactions」之表格中具有日期和金額的三個資料集。
    - 第一個資料集包含交易 1 至 7。
    - 第三個資料集包含交易 8 至 14。
    - 第三個資料集包含交易 15 至 21。
- when 條件判斷「Transactions」表格是否包含超過十列。若任何 when 陳述式評估為 TRUE，即會停止載入指令碼。此條件會放在三個資料集的結尾處。

載入指令碼

```
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 ;
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- id
- date
- amount

結果表格

id	日期	金額
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

在三個資料集中,每一個都有七筆交易。第一筆資料集包含交易 1 至 7 並載入到應用程式中。此 LOAD 陳述式之後的 `when` 條件評估為 FALSE,因為「`Transactions`」表格中少於十列。LOAD 指令碼繼續至下一個資料集。

第二個資料集包含交易 8 至 14 並載入到應用程式中。第二個 `when` 條件評估為 TRUE,因為「`Transactions`」表格超過十列。因此,指令碼會終止。

範例 3 – 多個 When 前置詞

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含單一筆交易的資料集已建立為名為「Transactions」的表格。
- 觸發的 For 迴圈包含兩個巢狀的 when 條件,其評估是否有以下情況:
    1. 「Transactions」表格中有少於 100 筆記錄。
    2. 「Transactions」中的記錄數不是 6 的倍數。

載入指令碼

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

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度:

- id

結果表格只會顯示前五個交易 ID,但載入指令碼會建立 36 列並在符合 when 條件後終止。

結果表格

id
0
1
2
3
4
5
其他 30 多列

`For` 迴圈中的巢狀 `When` 條件會評估下列問題：

- 「`Transactions`」表格中是否少於 100 列？
- 「`Transactions`」表格中的記錄總數是否為 6 的倍數？

每次 `When` 陳述式傳回 TRUE 的值時，即會產生另外七筆記錄並串連至現有的「`Transactions`」表格中。

`When` 條件會傳回 TRUE 值五次。此時，「`Transactions`」表格共有 36 列。

當「`Transactions`」建立 36 列資料時，第二個 `When` 陳述式會傳回 FALSE 的值，因此再也不會執行此項之後的 LOAD 陳述式。

3.3 指令碼一般陳述式

一般陳述式通常用於以某種方式操縱資料。這些陳述式可在指令碼中撰寫為任意行數，但是一定必須以分號 (";") 終止。

所有的指令碼關鍵字皆可以小寫和大寫字元的任意組合輸入。但用於陳述式中的欄位和變數名稱則會區分大小寫。

指令碼一般陳述式概述

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

Alias

alias 陳述式用來設定別名，只要欄位出現在下列指令碼，就會按照這個陳述式重新命名欄位。

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

Autonumber

此陳述式會針對在指令碼執行期間出現的每個相異的評估值，建立唯一的整數值。

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

Binary

binary 陳述式用來載入其他 QlikView 文件的資料，包括 Section Access 資料。

```
Binary  [path] filename
```

comment

可用來顯示資料庫和試算表的欄位註解 (中繼資料)。應用程式中未出現的欄位名稱將予以忽略。如果一個欄位名稱出現多次，會使用最後一個值。

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

comment table

可用來顯示資料庫或試算表的表格註解 (中繼資料)。

```
Comment table tablelist using mapname
```

```
Comment table tablename with comment
```

Connect

此功能無法用於 Qlik Sense SaaS。

CONNECT 陳述式用來定義透過 OLE DB/ODBC 介面對一般資料庫的 Qlik Sense 存取。若是 ODBC，首先必須使用 ODBC 管理員指定資料來源。

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

Declare

Declare 陳述式用來建立欄位定義，您可以從中定義欄位或函數之間的關係。一組欄位定義可用來自動產生衍生的欄位，它們可用作維度。例如，您可以建立行事曆定義，並用以從日期欄位中產生相關維度，例如年份、月份、週和日。

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

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

Derive

Derive 陳述式用來根據使用 **Declare** 陳述式建立的欄位定義以產生衍生的欄位。您可以指定要為其衍生欄位的資料欄位，或者根據欄位標記明確或隱含地衍生它們。

```
Derive [Field[s]] From [Field[s]] field_list Using definition
Derive [Field[s]] From Explicit [Tag[s]] (tag_list) Using definition
Derive [Field[s]] From Implicit [Tag[s]] Using definition
```

Direct Query

DIRECT QUERY 陳述式允許您透過使用 Direct Discovery 功能的 ODBC 或 OLE DB 連線，來存取表格。

```
Direct Query [path]
```

Directory

Directory 陳述式會在後續 **LOAD** 陳述式中定義要在其中尋找資料檔的目錄，直到發出新的 **Directory** 陳述式為止。

```
Directory [path]
```

Disconnect

Disconnect 陳述式會終止目前 ODBC/OLE DB/自訂連線。此陳述式是選用的。

```
Disconnect
```

drop field

透過 **drop field** 陳述式，可以隨時在指令碼執行期間，從資料模型和記憶體捨棄一或數個 Qlik Sense 欄位。在 **drop field** 陳述式後移除表格的「相異」屬性。

> ***drop field** 和 **drop fields** 兩者均可，效果並無不同。如果未指定表格，將從出現欄位的所有表格中捨棄欄位。*

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

drop table

透過 **drop table** 陳述式，可以隨時在指令碼執行期間，從資料模型和記憶體捨棄一或數個 Qlik Sense 內部表格。

> *同時接受表格 **drop table** 與 **drop tables**。*

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

Execute

Execute 陳述式用來在 Qlik Sense 載入資料時，執行其他程式。例如，進行必要的轉換。

```
Execute commandline
```

FlushLog

FlushLog 陳述式會強制 Qlik Sense 將指令碼緩衝的內容寫入至指令碼記錄檔。

```
FlushLog
```

Force

force 陳述式會強制 Qlik Sense 將後續 **LOAD** 及 **SELECT** 陳述式的欄位名稱與欄位值解譯為僅以大寫字母、僅小寫字母、字首一律大寫或大小寫混合 (混用) 來寫入。此陳述式能夠讓經過不同轉換的表格之中的欄位值產生關聯。

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

LOAD

LOAD 陳述式可以從檔案、指令碼中定義的資料、先前載入的表格、網頁、後續 **SELECT** 陳述式的結果或自動產生的資料來載入欄位。這也可以從分析連線載入資料。

```
 Load [ distinct ] *fieldlist
```

```
[( from file [ format-spec ] |
from_field fieldassource [format-spec]
inline data [ format-spec ] |
resident table-label |
autogenerate size )]
[ where criterion | while criterion ]
[ group_by groupbyfieldlist ]
[order_by orderbyfieldlist ]
[extension pluginname.functionname(tabledescription)]
```

Let

let 陳述式是 **set** 陳述式的補集，用來定義指令碼變數。相對於 **set** 陳述式，**let** 陳述式會先在指令碼執行時間評估 '=' 右邊的運算式，然後才將運算式指派給變數。

```
Let variablename=expression
```

Loosen Table

使用 **Loosen Table** 陳述式，可以在指令碼執行期間將一或多個 Qlik Sense 內部資料表明確宣告為鬆散耦合表格。當表格鬆散耦合時，會移除表格中欄位值之間的所有關聯。可以透過將鬆散耦合表格的每個欄位作為獨立、未連接的表格載入，從而實現類似的效果。在測試以暫時隔離資料結構的不同部分時，鬆散耦合可能有用。鬆散耦合的表格可以在表格檢視器中使用虛線進行識別。在指令碼中使用一或多個 **Loosen Table** 陳述式將造成 Qlik Sense 忽略指令碼執行前使表格成為鬆散耦合表格的任何設定。

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

Map ... using

map ... using 陳述式用來將特定欄位值或運算式對應到特定對應表的值。對應表是透過 **Mapping** 陳述式建立的。

```
Map  *fieldlist Using  mapname
```

NullAsNull

NullAsNull 陳述式會關閉先前由 **NullAsValue** 陳述式設定將 NULL 變成字串值的轉換。

```
NullAsNull *fieldlist
```

NullAsValue

NullAsValue 陳述式會指定應該將出現 NULL 的哪些欄位轉換為值。

```
NullAsValue *fieldlist
```

Qualify

Qualify 陳述式用來切換限定欄位名稱，亦即欄位名稱將獲得表格名稱做為前置詞。

```
Qualify *fieldlist
```

Rem

rem 陳述式用來將備註或註解插入指令碼中，或暫時停用指令碼陳述式，但不移除陳述式。

```
Rem string
```

Rename Field

此指令碼函數會在載入一或多個現有 Qlik Sense 欄位之後，為其重新命名。

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

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

Rename Table

此指令碼函數會在載入一或多個現有 Qlik Sense 內部表格之後，為其重新命名。

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

Section

透過 **section** 陳述式，可以定義是否應該將後續的 **LOAD** 和 **SELECT** 陳述式視為資料或存取權限的定義。

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

Select

透過標準 SQL **SELECT** 陳述式可選取來自 ODBC 資料來源或 OLE DB 提供者的欄位。然而，是否接受 **SELECT** 陳述式取決於所使用的 ODBC 驅動程式或 OLE DB 提供者。

```
Select [all | distinct | distinctrow | top n [percent] ] *fieldlist

From tablelist

[Where criterion ]

[Group by fieldlist [having criterion ] ]

[Order by fieldlist [asc | desc] ]

[ (Inner | Left | Right | Full)Join tablename on fieldref = fieldref ]
```

Set

set 陳述式用來定義指令碼變數。這些可用來取代字串、路徑、磁碟機等。

```
Set variablename=string
```

Sleep

sleep 陳述式會暫停所指定時間的指令碼執行。

```
Sleep n
```

SQL

SQL 陳述式可讓您透過 ODBC 或 OLE DB 連線傳送任意 SQL 命令。

```
SQL sql_command
```

SQLColumns

sqlcolumns 陳述式會傳回一組描述 ODBC 或 OLE DB 資料來源資料行的欄位，該欄位已經進行 **connect**。

```
SQLColumns
```

SQLTables

sqltables 陳述式會傳回一組描述 ODBC 或 OLE DB 資料來源表格的欄位，該欄位已經進行 **connect**。

```
SQLTables
```

SQLTypes

sqltypes 陳述式會傳回一組描述 ODBC 或 OLE DB 資料來源類型的欄位，該欄位已經進行 **connect**。

```
SQLTypes
```

Star

用來代表資料庫中某個欄位之所有值集合的字串可以透過 **star** 陳述式進行設定。會影響後續的 **LOAD** 與 **SELECT** 陳述式。

```
Star is [ string ]
```

Store

Store 陳述式建立 QVD、Parquet、CSV 或 TXT 檔案。

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

Tag

此指令碼陳述式提供將標記指派至一個或多個欄位或表格的方式。如果已嘗試將欄位或表格標記為不呈現在應用程式中，則會忽略該標記。若發現欄位或標記名稱衝突，會使用最後一個值。

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

Trace

使用 **trace** 陳述式可將字串寫入**指令碼執行進度**視窗以及指令碼記錄檔。在偵錯方面非常有幫助。使用在 **trace** 陳述式前計算的 $-expansions 變數，即可自訂訊息。

```
Trace string
```

Unmap

Unmap 陳述式會為後續載入的欄位，停用先前 **Map ... Using** 陳述式指定的欄位值對應。

```
Unmap *fieldlist
```

Unqualify

Unqualify 陳述式用來關閉先前由 **Qualify** 陳述式開啟的欄位名稱限定。

```
Unqualify *fieldlist
```

Untag

此指令碼陳述式提供從欄位或表格移除標記的方式。如果已嘗試將欄位或表格取消標記為不呈現在應用程式中，則會忽略該取消標記。

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

Alias

alias 陳述式用來設定別名，只要欄位出現在下列指令碼，就會按照這個陳述式重新命名欄位。

語法：

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

引數：

引數

引數	描述
fieldname	您的來源資料的欄位名稱
aliasname	您要改用的別名名稱

範例與結果：

範例	結果
`Alias ID_N as NameID;`	
`Alias A as Name, B as Number, C as Date;`	透過這個陳述式定義的名稱變更會用於所有後續的 **SELECT** 和 **LOAD** 陳述式。指令碼中任何後續位置的新 **alias** 陳述式都可以定義欄位名稱的新別名。

AutoNumber

此陳述式會針對在指令碼執行期間出現的每個相異的評估值，建立唯一的整數值。

您也可以使用 **LOAD** 陳述式中的 *autonumber (page 538)* 函數，但若您想要使用最佳載入，這會有一些限制。您可以先從 **QVD** 檔案載入資料，以建立最佳載入，然後使用 **AutoNumber** 陳述式以將值轉換為符號鍵。

語法：

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

引數：

引數

引數	描述
*欄位清單	以逗號分隔的欄位清單，在此值應被唯一整數值取代。 您可以在欄位名稱中使用萬用字元 ? 和 *，以納入名稱符合的所有欄位。您也可以使用 * 以納入所有欄位。您需要在使用萬用字元時引用欄位名稱。
命名空間	可選擇 **Using** 命名空間。若您想要建立命名空間，可以使用此選項，其中不同欄位中的相同值共用相同的金鑰。 若您不使用此選項，所有欄位都會有獨立的金鑰索引。

限制：

若您在指令碼中有數個 **LOAD** 陳述式，則需要在最終的 **LOAD** 陳述式之後置放 **AutoNumber** 陳述式。

範例 - 含 AutoNumber 的指令碼

指令碼範例

在此範例中，首先以不含 **AutoNumber** 陳述式的方式載入資料。然後再新增 **AutoNumber** 陳述式以顯示效果。

範例中使用的資料

在資料載入編輯器中載入下列資料作為內嵌載入，以建立以下的指令碼範例。目前先忽略已註解的 **AutoNumber** 陳述式。

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

Budget:
LOAD Budget,
Region &'|'& Year &'|'& Month as KeyToOtherTable
INLINE
[Region, Year, Month, Budget
North,  2014,   May,    200
North,  2014,   May,    350
North,  2014,   June,   150
South,  2014,   June,   500
```

```
South,  2013,   May,    300
South,  2013,   May,    200
];

//AutoNumber KeyToOtherTable;
```

建立視覺化

在 Qlik Sense 工作表中建立兩個表格視覺化。將 **KeyToOtherTable**、**Region**、**Year**、**Month** 和 **Sales** 作為維度新增至第一個表格。將 **KeyToOtherTable**、**Region**、**Year**、**Month** 和 **Budget** 作為維度新增至第二個表格。

結果

RegionSales 表格

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

Budget 表格

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

解釋

範例顯示連結兩個表格的複合欄位 **KeyToOtherTable**。未使用 **AutoNumber**。注意 **KeyToOtherTable** 值的長度。

新增 AutoNumber 陳述式

在載入指令碼中取消註解 **AutoNumber** 陳述式：

```
AutoNumber KeyToOtherTable;
```

結果

RegionSales 表格

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

Budget 表格

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

解釋

已使用唯一整數值取代 **KeyToOtherTable** 欄位值，因此，已減少欄位值的長度，而能節省記憶體。兩個表格中的索引鍵欄位受到 **AutoNumber** 影響，並且表格保持連結。該範例僅供示範之用，在處理含大量列的表格時具有意義。

Binary

binary 陳述式用來載入其他 Qlik Sense 應用程式或 QlikView 文件的資料，包括區段存取資料。不會包括應用程式的其他元素，例如，工作表、故事、視覺化、主項目或變數。

指令碼中僅允許一個 **binary** 陳述式。**binary** 陳述式必須是指令碼中的第一個陳述式，即使當 SET 陳述式通常位於指令碼的開頭亦是如此。

語法：

```
binary [path] filename
```

引數:

引數

引數	描述
path	應成為參考資料夾資料連線的檔案路徑如果檔案不位於 Qlik Sense 的工作目錄中，則需要此路徑。 **範例: *'lib://Table Files/'*** 在舊式指令碼模式下，還支援下列路徑格式: • 絕對路徑 **範例: *c:\data*** • 與包含此指令碼行之應用程式相關。 **範例: *data***
filename	檔案的名稱，其中包括副檔名 .qvw 或 .qvf。

限制:

您無法使用 **binary**，參考應用程式 ID 以從相同 Qlik Sense Enterprise 部署上的應用程式載入資料。您只能從 *.qvf* 檔案載入。

範例

字串	描述
`Binary lib://DataFolder/customer.qvw;`	在此範例中，檔案必須位於**資料夾**資料連線中。例如，這可能是管理員在 Qlik Sense 伺服器上建立的資料夾。按一下資料載入編輯器中的**建立新連線**，然後選取**檔案位置**之下的**資料夾**。
`Binary customer.qvf;`	在此範例中，檔案必須位於 Qlik Sense 工作目錄中。
`Binary c:\qv\customer.qvw;`	此範例使用了一個僅在舊指令碼模式下才生效的絕對檔案路徑。

Comment field

可用來顯示資料庫和試算表的欄位註解 (中繼資料)。應用程式中未出現的欄位名稱將予以忽略。如果一個欄位名稱出現多次，會使用最後一個值。

語法:

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

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

使用的對應表應該有兩個資料行，第一個包含欄位名稱，第二個包含註解。

引數：

引數

引數	描述
**fieldlist*	以逗號分隔的欄位清單，其中欄位將會加上註解。使用 * 作為表示所有欄位的欄位清單。可在欄位名稱中使用萬用字元 * 與 ?。使用萬用字元時，可能需要為欄位名稱加上引號。
mapname	先前在對應 **LOAD** 或對應 **SELECT** 陳述式中讀取的對應表格名稱。
fieldname	應加上註解的欄位名稱。
comment	應新增到欄位的註解。

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

可用來顯示資料庫或試算表的表格註解 (中繼資料)。

應用程式中未出現的表格名稱將予以忽略。如果一個表格名稱出現多次，會使用最後一個值。可使用關鍵字讀取資料來源的註解。

語法：

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

引數：

引數

引數	描述
tablelist	(table{,table})
mapname	先前在對應 **LOAD** 或對應 **SELECT** 陳述式中讀取的對應表格名稱。
tablename	應加上註解的表格名稱。
comment	應新增到表格的註解。

Example 1:

```
Commentmap:
mapping LOAD * inline [
a,b
Main,This is the fact table
Currencies, Currency helper table
];
comment tables using Commentmap;
```

Example 2:

```
comment table Main with 'Main fact table';
```

Connect

CONNECT 陳述式用來定義透過 OLE DB/ODBC 介面對一般資料庫的 Qlik Sense 存取。若是 ODBC，首先必須使用 ODBC 管理員指定資料來源。

此功能無法用於 Qlik Sense SaaS。

此陳述式僅在標準模式下支援資料夾資料連線。

語法：

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

引數：

引數

引數	描述

connect-string	`connect-string ::= datasourcename { ; conn-spec-item }`
	連線字串是資料來源名稱，也是一或多個連線規格項目的選用清單。如果資料來源名稱包含空白，或者列出任何連線規格項目，則連線字串必須以引號括住。
	
	datasourcename 必須是定義的 ODBC 資料來源或定義 OLE DB 提供者的字串。
	
	`conn-spec-item ::=`**`DBQ=`**`database_specifier |`**`DriverID=`**`driver_specifier |`**`UID=`**`userid |`**`PWD=`**`password`
	
	不同資料庫之間的可能連線規格項目可能有所不同。對於某些資料庫，也可能是上述之外的其他項目。對於 OLE DB，有些連線特定項目是必要的項目，而非選用項目。

connection	資料載入編輯器中儲存的資料連線名稱。

如果 **ODBC** 加在 **CONNECT** 前面，將使用 ODBC 介面，否則將使用 OLE DB。

利用在資料載入編輯器中建立的已儲存資料連線，使用 **LIB CONNECT TO** 連線至資料庫。

Example 1:

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

透過此陳述式定義的資料來源將由後續的 **Select (SQL)** 陳述式使用，直到執行新的 **CONNECT** 陳述式為止。

Example 2:

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

Connect32

此陳述式的使用方式與 **CONNECT** 陳述式相同，不過會強制 64 位元系統使用 32 位元 ODBC/OLE DB 提供者。不適用於自訂連線。

Connect64

此陳述式的使用方式與 **CONNECT** 陳述式相同，不過會強制使用 64 位元提供者。不適用於自訂連線。

Declare

Declare 陳述式用來建立欄位定義，您可以從中定義欄位或函數之間的關係。一組欄位定義可用來自動產生衍生的欄位，它們可用作維度。例如，您可以建立行事曆定義，並用以從日期欄位中產生相關維度，例如年份、月份、週和日。

您可以使用 **Declare** 設定新的欄位定義，也可以根據現有定義建立欄位定義。

設定新的欄位定義

語法：

```
definition_name:

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

[Parameters parameter_list ]

Fields field_list
```

引數：

引數	描述
definition_name	欄位定義的名稱，以冒號結尾。 *不要將 autoCalendar 用作欄位定義的名稱，因為該名稱要保留給自動產生的行事曆範本使用。* **範例：** `Calendar:`
tag_list	用逗號分隔的標記清單，套用至從欄位定義衍生的欄位。套用標記是可選的，但是若您不想套用用於指定排序順序的標記，比如 $date、$numeric 或 $text，衍生欄位在預設情況下會按照載入順序排序。 **範例：** `'$date'Thank you for bringing this to our attention, and apologies for the inconvenience.`
parameter_list	用逗號分隔的參數清單。參數定義為以下格式 `name=value`，並獲指派了一個起始值，當重新使用欄位定義時可覆寫該值。選用。 **範例：** `first_month_of_year = 1`
field_list	當使用欄位定義時要產生的用逗號分隔的欄位清單。欄位定義為以下格式：`<expression> As field_name tagged tag`。使用 `$1` 參照資料欄位，衍生欄位應從該資料欄位產生。 **範例：** `Year($1) As Year tagged ('$numeric')`

範例：

```
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')
;
```

行事曆現已定義，您可以使用 **Derive** 子句將其套用至已經載入的日期欄位，在此情況下為 OrderDate 和 ShippingDate。

重新使用現有欄位定義

語法：

```
<definition name>:

Declare [Field][s] Definition

Using <existing_definition>

[With <parameter_assignment> ]
```

引數：

引數	描述
definition_name	欄位定義的名稱，以冒號結尾。 **範例：** `MyCalendar:`
existing_definition	建立新的欄位定義時要重新使用的欄位定義。新欄位定義的運作方式與其所基於的定義相同，但是如果您使用 parameter_assignment 來變更欄位運算式中使用的值則屬例外。 **範例：** `Using Calendar`

引數	描述
parameter_ assignment	用逗號分隔的參數指派清單。參數指派定義為以下格式 name=value，並覆寫在基本欄位定義中設定的參數值。選用。 **範例：** `first_month_of_year = 4`

範例：

在此範例中，我們重新使用在之前的範例中建立的行事曆定義。這種情況下，我們想使用一個從 4 月開始的財務年度。將數值 4 指派至 first_month_of_year 參數即可實現，此操作會影響已定義的 DayNumberOfYear 欄位。

此範例假定您使用的是來自上一個範例的樣本資料和欄位定義。

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



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

當您重新載入資料指令碼後，產生的欄位在工作表編輯器中可用，且名稱為 OrderDate.MyCalendar.* 和 ShippingDate.MyCalendar.*。

Derive

Derive 陳述式用來根據使用 **Declare** 陳述式建立的欄位定義以產生衍生的欄位。您可以指定要為其衍生欄位的資料欄位，或者根據欄位標記明確或隱含地衍生它們。

語法：

```
Derive [fields]] From [Field[s]] field_list Using definition
```

```
Derive [Field[s]] From Explicit [Tag[s]] tag_list Using definition
```

```
Derive [Field[s]] From Implicit [Tag[s]] Using definition
```

引數：

引數

引數	描述
definition	衍生欄位時要使用的欄位定義名稱。 **範例：** `Calendar`

引數	描述
field_list	資料欄位的逗號分隔清單，根據欄位定義，衍生欄位應從此資料欄位產生。資料欄位應為您已在指令碼中載入的欄位。 **範例：** `OrderDate, ShippingDate`
tag_list	標記的逗號分隔清單。將對包含任何列出標記的所有資料欄位產生衍生欄位。標記清單應以圓括弧括起。 **範例：** `('$date','$timestamp')`

範例：

- 為特定資料欄位衍生欄位。
  這種情況下，我們指定 OrderDate 和 ShippingDate 欄位。
  `DERIVE FIELDS FROM FIELDS OrderDate,ShippingDate USING Calendar;`
- 為含有特定標記的所有欄位衍生欄位。
  這種情況下，我們基於 Calendar 為所有含有 `$date` 標記的欄位衍生欄位。
  `DERIVE FIELDS FROM EXPLICIT TAGS ('$date') USING Calendar;`
- 為所有包含欄位定義標記的欄位衍生欄位。
  在這種情況下，我們為包含與 Calendar 欄位定義相同的標記 (此情況下為 `$date`).的所有資料欄位衍生欄位。
  `DERIVE FIELDS FROM IMPLICIT TAG USING Calendar;`

Direct Query

DIRECT QUERY 陳述式允許您透過使用 Direct Discovery 功能的 ODBC 或 OLE DB 連線，來存取表格。

語法：

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

DIMENSION、**MEASURE** 及 **DETAIL** 關鍵字可以按任何順序使用。

所有的 **DIRECT QUERY** 陳述式都需要有 **DIMENSION** 和 **FROM** 關鍵字子句。**FROM** 關鍵字必須出現在 **DIMENSION** 關鍵字之後。

緊接著 **DIMENSION** 關鍵字之後所指定的欄位會載入於記憶體內，並可用於在記憶體內資料和 Direct Discovery 資料之間建立關聯。

__DIRECT QUERY__ 陳述式不能包含 __DISTINCT__ 或 __GROUP BY__ 子句。

使用 **MEASURE** 關鍵字，您可以定義 Qlik Sense 在「中繼層級」才會識別的欄位。在資料載入處理程序期間，量值欄位的實際資料僅位於資料庫上，並且由視覺化中使用的圖表運算式以臨機操作為基礎進行擷取。

一般來說, 含離散值、要作為維度的欄位應以 **DIMENSION** 關鍵字載入, 而僅要用於彙總的數字應以 **MEASURE** 關鍵字選取。

DETAIL 欄位可提供資訊或詳細資料 (如 [註解] 欄位), 使用者可顯示在向下探查詳細資料的表格方塊中。**DETAIL** 欄位無法用於圖表運算式。

設計上, **DIRECT QUERY** 陳述式對於支援 SQL 的資料來源採取資料來源中性的態度。因此, 相同的 **DIRECT QUERY** 陳述式不須變更即可用於不同的 SQL 資料庫。Direct Discovery 會視需要產生適合資料庫的查詢。

當使用者知道要查詢的資料庫為何, 並且想要在 SQL 中利用資料庫特定的延伸, 就可使用原生資料來源語法。原生資料來源語法支援：

- 在 **DIMENSION** 和 **MEASURE** 子句中作為欄位運算式
- 作為 **WHERE** 子句的內容

範例:

```
DIRECT QUERY

    DIMENSION Dim1, Dim2
    MEASURE

        NATIVE ('X % Y') AS X_MOD_Y

FROM TableName
DIRECT QUERY

    DIMENSION Dim1, Dim2
    MEASURE X, Y
    FROM TableName
    WHERE NATIVE ('EMAIL MATCHES "\*.EDU"')
```

下列詞彙作為關鍵字使用, 因此無法用作資料行或欄位名稱, 以引用的方式除外：and, as, detach, detail, dimension, distinct, from, in, is, like, measure, native, not, or, where

引數:

引數	描述
fieldlist	欄位規格的逗號分隔清單, *fieldname {, fieldname}*. 欄位規格可以是欄位名稱, 在此情況下, 相同名稱會用於資料庫資料行名稱和 Qlik Sense 欄位名稱。欄位規格也可以是「欄位別名」, 在此情況下, 資料庫運算式或資料行名稱會使用 Qlik Sense 欄位名稱。
tablelist	要從其中載入資料之資料庫中表格或檢視的名稱清單。這通常是其中包含在資料庫上執行 JOIN 的檢視。

引數	描述

where_ clause	資料庫 **WHERE** 子句的完整語法不限於此，但大部分的 SQL「關聯運算式」都可使用，包括使用函數呼叫、用於字串的 **LIKE** 運算子、**IS NULL** 和 **IS NOT NULL**, **IN. BETWEEN** 不包括在內。
	
	NOT 是一元運算子，而不是在某些關鍵字上的修飾詞。
	
	範例：
	
	`WHERE x > 100 AND "Region Code" IN ('south', 'west')`
	`WHERE Code IS NOT NULL and Code LIKE '%prospect'`
	`WHERE NOT X in (1,2,3)`
	最後一個範例不可編寫為：
	
	`WHERE X NOT in (1,2,3)`

範例：

此範例會使用稱為 TableName 的資料庫表格，其中包含欄位 Dim1、Dim2、Num1、Num2 和 Num3。Dim1 和 Dim2 會載入 Qlik Sense 資料集中。

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

Dim1 和 Dim2 將可用作維度。Num1, Num2 和 Num3 將可用於彙總。Dim1 和 Dim2 也可用於彙總。Dim1 和 Dim2 可用於彙總的類型取決於其資料類型。例如，許多情況下，**維度** 欄位包含如名稱或帳戶號碼的字串資料。這些欄位無法加總，但可計數：`count(Dim1)`。

__DIRECT QUERY__ 陳述式直接在指令碼編輯器中進行編寫。為了簡化 __DIRECT QUERY__ 陳述式的建構，您可以從資料連線中產生一個 __SELECT__ 陳述式，然後編輯所產生的指令碼，將它變更為 __DIRECT QUERY__ 陳述式。
例如，__SELECT__ 陳述式：

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

可以變更為下列 __DIRECT QUERY__ 陳述式：

```
DIRECT QUERY
 DIMENSION
 SalesOrderID,
 RevisionNumber

MEASURE
 SubTotal,
 TaxAmt

DETAIL
OrderDate

FROM MyDB.Sales.SalesOrderHeader;
```

Direct Discovery 欄位清單

欄位清單是以逗號分隔的欄位規格清單，*fieldname {, fieldname}*。欄位規格可以是欄位名稱，在此情況下，相同名稱會用於資料庫資料行名稱和欄位名稱。欄位規格也可以是「欄位別名」，在此情況下，資料庫運算式或資料行名稱會使用 Qlik Sense 欄位名稱。

欄位名稱可以是簡單名稱或引用名稱。簡單名稱以一個字母 Unicode 字元開頭，後面接著字母字元、數字字元或底線的任意組合。引用名稱以一個雙引號開頭，然後包含任何的字元序列。如果引用名稱包含雙引號，這些雙引號會以兩個相鄰的雙引號表示。

Qlik Sense 欄位名稱區分大小寫。資料庫欄位名稱不一定區分大小寫，需視資料庫而定。Direct Discovery 查詢會保留所有欄位識別碼和別名的大小寫。以下範例中，別名 "MyState" 會在內部使用，以儲存來自資料庫資料行 "STATEID" 的資料。

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

這跟使用別名的 **SQL Select** 陳述式的結果不同。如果別名未明確引用，則結果會包含目標資料庫傳回之資料行的預設大小寫。以下範例中，針對 Oracle 資料庫的 **SQL Select** 陳述式會建立 "MYSTATE," (全大寫字母) 作為內部 Qlik Sense 別名，雖然指定的別名是大小寫混合的。**SQL Select** 陳述式會使用資料庫傳回的資料行名稱，在 Oracle 的情況下為全大寫。

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

若要避免這種行為，可使用 LOAD 陳述式來指定別名。

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

在此範例中，Qlik Sense 會將 "STATEID" 資料行內部儲存為 "MyState"。

大部分的資料庫純量運算式都允許作為欄位規格。函數呼叫也可用於欄位規格。運算式可包含布林值常數、數值常數，或置於單引號內的字串常數 (內嵌單引號以相鄰的單引號表示)。

範例：

```
DIRECT QUERY

   DIMENSION

      SalesOrderID, RevisionNumber

   MEASURE

      SubTotal AS "Sub Total"

FROM AdventureWorks.Sales.SalesOrderHeader;


DIRECT QUERY

   DIMENSION

      "SalesOrderID" AS "Sales Order ID"

   MEASURE

      SubTotal,TaxAmt,(SubTotal-TaxAmt) AS "Net Total"

FROM AdventureWorks.Sales.SalesOrderHeader;


DIRECT QUERY

   DIMENSION

      (2*Radius*3.14159) AS Circumference,

      Molecules/6.02e23 AS Moles

   MEASURE
```

```
        Num1 AS numA

    FROM TableName;
```

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

Direct Discovery 不支援在 **LOAD** 陳述式中使用彙總。若使用彙總，結果會無法預期。不得使用下列 **LOAD** 陳述式：

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

SUM 不得在 **LOAD** 陳述式中。

Direct Discovery 也不支援 Qlik Sense 函數用於 **Direct Query** 陳述式中。例如，當 "Mth" 欄位用作視覺化中的維度時，以下 **DIMENSION** 欄位的規格會導致失敗：

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

Directory

Directory 陳述式會在後續 **LOAD** 陳述式中定義要在其中尋找資料檔的目錄，直到發出新的 **Directory** 陳述式為止。

語法：

```
Directory[path]
```

如果發出 **Directory** 陳述式，但是沒有 **path** 或已省略，則 Qlik Sense 將查閱 Qlik Sense 工作目錄。

引數：

引數

引數	描述
path	可以解譯為 data 檔路徑的文字。 路徑是檔案的路徑，為以下兩者之一： • 絕對路徑 **範例： *c:\data*** • 與 Qlik Sense 應用程式工作目錄相關。 **範例： *data*** • 指向網際網路位置或內部網路位置的 URL 位址 (HTTP 或 FTP)。 **範例： *http://www.qlik.com***

範例:

```
DIRECTORY C:\userfiles\data; // OR -> DIRECTORY data\

LOAD * FROM
[data1.csv] // ONLY THE FILE NAME CAN BE SPECIFIED HERE (WITHOUT THE FULL PATH)
(ansi, txt, delimiter is ',', embedded labels);

LOAD * FROM
[data2.txt] // ONLY THE FILE NAME CAN BE SPECIFIED HERE UNTIL A NEW DIRECTORY STATEMENT IS
MADE
(ansi, txt, delimiter is '\t', embedded labels);
```

Disconnect

Disconnect 陳述式會終止目前 ODBC/OLE DB/自訂連線。此陳述式是選用的。

語法:

```
Disconnect
```

執行新的 **connect** 陳述式或指令碼執行完成時,連線將自動終止。

範例:

```
Disconnect;
```

Drop

Drop 指令碼關鍵字可以用來從資料庫中捨棄表格或欄位。

Drop field

透過 **drop field** 陳述式,可以隨時在指令碼執行期間,從資料模型和記憶體捨棄一或數個 Qlik Sense 欄位。在 **drop field** 陳述式後移除表格的「相異」屬性。

> ***drop field*** *和* ***drop fields*** *兩者均可,效果並無不同。如果未指定表格,將從出現欄位的所有表格中捨棄欄位。*

語法:

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

範例:

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

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

Drop table

透過 **drop table** 陳述式，可以隨時在指令碼執行期間，從資料模型和記憶體捨棄一或數個 Qlik Sense 內部表格。

語法：

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

*同時接受表格 **drop table** 與 **drop tables**。*

下列項目將因此捨去：

- 實際表格。
- 不屬於剩餘表格的所有欄位。
- 完全來自於捨棄表格的剩餘欄位中的欄位值。

範例與結果：

範例	結果
drop table Orders, Salesmen, T456a;	這一行將導致從記憶體捨棄三個表格。
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;	建立表格 *Tab2* 後，將捨棄表格 *Tab1*。

Drop table

透過 **drop table** 陳述式，可以隨時在指令碼執行期間，從資料模型和記憶體捨棄一或數個 Qlik Sense 內部表格。

語法：

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

*同時接受表格 **drop table** 與 **drop tables**。*

下列項目將因此捨去:

- 實際表格。
- 不屬於剩餘表格的所有欄位。
- 完全來自於捨棄表格的剩餘欄位中的欄位值。

範例與結果:

範例	結果
`drop table Orders, Salesmen, T456a;`	這一行將導致從記憶體捨棄三個表格。

`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;`
	建立表格 *Tab2* 後,將捨棄表格 *Tab1*。

Execute

Execute 陳述式用來在 Qlik Sense 載入資料時,執行其他程式。例如,進行必要的轉換。

此功能無法用於 Qlik Sense SaaS。

標準模式下不支援這個陳述式。

語法:

```
execute commandline
```

引數:

引數

引數	描述
commandline	可以由作業系統解讀為命令行的文字。您可以參考絕對檔案路徑或 lib:// 資料夾路徑。

如要使用 **Execute**,需要符合以下條件:

- 您必須在舊版模式 (適用於 Qlik Sense 和 Qlik Sense Desktop) 下執行。
- 您需要在 *Settings.ini* 中將 OverrideScriptSecurity 設定為 1 (適用於 Qlik Sense)。
  Settings.ini 位於 *C:\ProgramData\Qlik\Sense\Engine*,通常情況下是一個空檔案。

*如果您設定 OverrideScriptSecurity 以啟用 **Execute**，任何使用者均可在伺服器上執行檔案。例如，使用者可以附加一個可執行檔案到應用程式中，然後在資料載入指令碼中執行此檔案。*

請執行下列動作：

1. 複製 *Settings.ini* 並在文字編輯器中開啟。
2. 檢查檔案的第一行是否包含 *[Settings 7]*。
3. 插入新行並輸入 *OverrideScriptSecurity=1*。
4. 在檔案結束處插入空行。
5. 儲存檔案。
6. 使用您的已編輯檔案取代 *Settings.ini*。
7. 重新啟動 Qlik Sense Engine Service (QES)。

如果 Qlik Sense 正在作為一項服務執行，某些命令可能無法如期執行。

範例：

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

Field/Fields

Field 和 **Fields** 指令碼關鍵字用在 **Declare**、**Derive**、**Drop**、**Comment**、**Rename** 和 **Tag/Untag** 陳述式中。

FlushLog

FlushLog 陳述式會強制 Qlik Sense 將指令碼緩衝的內容寫入至指令碼記錄檔。

語法：

```
FlushLog
```

緩衝區的內容會被寫入記錄檔案中。此命令對於偵錯可能會很有用，因為您將會收到在指令碼執行失敗時可能丟失的資料。

範例：

```
FlushLog;
```

Force

force 陳述式會強制 Qlik Sense 將後續 **LOAD** 及 **SELECT** 陳述式的欄位名稱與欄位值解譯為僅以大寫字母、僅小寫字母、字首一律大寫或大小寫混合 (混用) 來寫入。此陳述式能夠讓經過不同轉換的表格之中的欄位值產生關聯。

語法:

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

若未指定任何項目,將採用強制大小寫混合。force 陳述式的效力會持續到進行新的 force 陳述式為止。

force 陳述式對於存取區段沒有作用:載入的所有欄位值都不區分大小寫。

範例與結果

範例	結果

此範例顯示如何強制大寫。
`FORCE Capitalization;`
`Capitalization:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	Capitalization 表格包含下列值:
	`Ab`
	`Cd`
	`Ef`
	`Gh`
	所有值都是大寫的。

此範例顯示如何強制大寫。
`FORCE Case Upper;`
`CaseUpper:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	CaseUpper 表格包含下列值:
	`AB`
	`CD`
	`EF`
	`GH`
	所有值都是大寫的。

範例	**結果**

此範例顯示如何強制小寫。
 	
`FORCE Case Lower;`
 	
`CaseLower:`
 	
`LOAD * Inline [`
 	
`ab`
 	
`Cd`
 	
`eF`
 	
`GH`
 	
`];`
	CaseLower 表格包含下列值：
	
	`ab`
	
	`cd`
	
	`ef`
	
	`gh`
	所有值都是小寫的。

此範例顯示如何強制大小寫混合。
 	
`FORCE Case Mixed;`
 	
`CaseMixed:`
 	
`LOAD * Inline [`
 	
`ab`
 	
`Cd`
 	
`eF`
 	
`GH`
 	
`];`
	CaseMixed 表格包含下列值：
	
	`ab`
	
	`Cd`
	
	`eF`
	
	`GH`
	所有值都保持指令碼中的大小寫不變。

另請參見：

From

From 指令碼在 **Load** 陳述式中用來參照檔案，並在 **Select** 陳述式中用來參照資料庫表格或視圖。

Load

LOAD 陳述式可以從檔案、指令碼中定義的資料、先前載入的表格、網頁、後續 **SELECT** 陳述式的結果或自動產生的資料來載入欄位。這也可以從分析連線載入資料。

語法：

```
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 ]
```

引數:

引數

引數	描述
distinct	若您只想載入獨特記錄，則可以使用 **distinct** 作為述詞。若有重複的記錄，將會載入第一個執行個體。 若您正在使用前置載入，您需要將 **distinct** 置於第一個 LOAD 陳述式，如同 **distinct** 僅影響目的地表格。

引數	描述
fieldlist	*fieldlist ::= (* | field{**,** * | field })* 要載入的欄位清單。使用 * 做為欄位清單表示表格中的所有欄位。 *field ::= (fieldref | expression) [**as** aliasname]* 欄位定義必須一律包含常值、現有欄位的參考或運算式。 *fieldref ::= (fieldname |@fieldnumber |@startpos:endpos [**I** | **U** | **R** | **B** | **T**])* *fieldname* 是與表格中欄位名稱相同的文字。請注意，欄位名稱如果包含空格則必須以一般雙引號或方括弧括住。有時候不一定會有檔案名稱。這時請用不同的標記法： *@fieldnumber* 代表分隔表格檔案中的欄位編號。其必須是正整數，前面加上 "@"。編號一律會從 1 開始，一直編號到欄位的數目為止。 *@startpos*:*endpos* 代表欄位在固定長度記錄的檔案中開始和結束的位置。這些位置必須是正整數。這兩個編號前面必須加上 "@"，並且以冒號分隔。編號一律會從 1 開始，一直編號到位置的數目為止。在最後一個欄位中，**n** 用作結束位置。 • 如果 *@startpos*:*endpos* 後面緊接著 **I** 或 **U** 字元，會將讀取的位元組解譯為二元帶正負號 (**I**) 或不帶正負號 (**U**) 的整數 (Intel 位元組順序)。讀取的位置編號必須是 1、2 或 4。 • 如果 *@startpos*:*endpos* 後面緊接著 **R** 字元，會將讀取的位元組解譯為二進位實數 (IEEE 32 位元或 64 位元浮點)。讀取的位置編號必須是 4 或 8。 • 如果 *@startpos*:*endpos* 後面緊接著 **B** 字元，會按照 COMP-3 標準將讀取的位元組解譯為 BCD (Binary Coded Decimal) 編號。可指定任何數目的位元組。 *expression* 可以是以同一個表格中的其他一或數個欄位為基礎的數值函數或字串函數。如需進一步資訊，請參閱運算式的語法。 **as** 用來指派欄位的新名稱。

<table>
<tr><th>引數</th><th>描述</th></tr>
<tr><td>from</td><td>
from 在應使用資料夾或網頁檔案資料連線從檔案中載入資料的情況下使用
file ::= [path] filename
範例：'lib://Table Files/'
如果省略路徑，Qlik Sense 將在 Directory 陳述式指定的目錄中搜尋該檔案。如果沒有 Directory 陳述式，則 Qlik Sense 會在工作目錄 C:\Users\{user}\Documents\Qlik\Sense\Apps 中進行搜尋。
在 Qlik Sense 伺服器安裝中，工作目錄在 Qlik Sense Repository Service 中指定，預設情況下它位於 C:\ProgramData\Qlik\Sense\Apps。
filename 可包含標準 DOS 萬用字元 (* 和 ?)。這將載入指定的目錄中所有相符的檔案。
format-spec ::= (fspec-item { , fspec-item })
格式規格包含括弧內數個格式規格項目的清單。
舊版指令碼模式
在舊式指令碼模式下，還支援下列路徑格式：

絕對路徑
範例：c:\data\
與 Qlik Sense 應用程式工作目錄相關。
範例：data\
指向網際網路位置或內部網路位置的 URL 位址 (HTTP 或 FTP)。
範例：http://www.qlik.com

</td></tr>
</table>

引數	描述
from_field	如果應該從先前載入的欄位載入資料，可使用 **from_field**。 *fieldassource::=*(*tablename, fieldname)* 此欄位是先前載入的 *tablename* 和 *fieldname* 名稱。 *format-spec ::= (fspec-item {, fspec-item })* 格式規格包含括弧內數個格式規格項目的清單。如需詳細資訊，請參閱 *格式規格項目 (page 157)*。 *分隔表格中的欄位時，**from_field** 僅支援逗號作為清單分隔符號。*
inline	如果應該在指令碼中輸入資料，而不是從檔案載入資料，可使用 **inline**。 *data ::= [text]* 透過 **inline** 子句輸入的資料必須以雙引號或方括弧括住。這些之間的文字將以檔案內容的相同方式加以解譯。因此，在文字檔中要插入新行的位置，也應該在 **inline** 子句的文字中插入新行，也就是在輸入指令碼時按下 Enter 鍵。資料行數量以第一行定義。 *format-spec ::= (fspec-item {, fspec-item })* 格式規格包含括弧內數個格式規格項目的清單。如需詳細資訊，請參閱 *格式規格項目 (page 157)*。
resident	如果應該從先前載入的表格載入資料，可使用 **resident**。 *table label* 是建立原始表格的 **LOAD** 或 **SELECT** 陳述式之前的標籤。標籤結尾應該加上冒號。
autogenerate	如果 Qlik Sense 應該自動產生資料，可使用 **autogenerate**。 *size ::= number* *Number* 是表示要產生記錄筆數的整數。 此欄位清單不得包含需要外部資料來源或先前已載入的表格中所含資料的運算式，除非您參考的是先前已透過 **Peek** 函數載入的表格中的單一欄位值。

<table>
<tr><th>引數</th><th>描述</th></tr>
<tr><td>extension</td><td>您可以從分析連線載入資料。您需要使用延伸子句以叫用伺服器端延伸 (SSE) 外掛程式中定義的函數，或評估指令碼。

您可以將單一表格傳送至 SSE 外掛程式，就會傳回單一資料表格。若外掛程式沒有指定傳回的欄位名稱，則欄位必須命名為 Field1, Field2 等。

<code>Extension pluginname.functionname(tabledescription);</code>

使用 SSE 外掛程式中的函數載入資料
tabledescription ::= (table { ,tablefield})
若您沒有說明表格欄位，則必須以載入順序使用欄位。
評估 SSE 外掛程式中的指令碼以載入資料
tabledescription ::= (script, table { ,tablefield})

表格欄位定義中的資料類型處理

分析連線中會自動偵測資料類型。若資料沒有數值，且有至少一個非 NULL 文字字串，則欄位可被視為文字。在任何其他情況下，這被視為數字。

您可以 String() 或 Mixed() 括起欄位名稱，以強制資料類型。

String() 可強制欄位為文字。若欄位為數字，則會擷取雙值的文字部分，不會執行任何轉換。
Mixed() 可強制欄位為雙值。

String() 或 Mixed() 無法在延伸表格欄位定義之外使用，您無法在表格欄位定義中使用其他 Qlik Sense 函數。

更多分析連線相關資訊

您需要先設定分析連線，才能使用。</td></tr>
<tr><td>where</td><td>where 是指明選項中是否應該包含記錄的子句。如果 criterion 為 True，將包含選項。
criterion 是邏輯運算式。</td></tr>
<tr><td>while</td><td>while 是指明是否應該重複讀取記錄所用的子句。只要 criterion 為 True，就會讀取相同的記錄。若要使用，while 子句一般必須包含 IterNo( ) 函數。

criterion 是邏輯運算式。</td></tr>
<tr><td>group by</td><td>group by 是定義應該對哪些欄位彙總 (分組) 資料所用的子句。彙總欄位應該以某些方式包含在載入的運算式中。只有彙總欄位才能在載入的運算式中的彙總函數之外使用。

groupbyfieldlist ::= (fieldname { ,fieldname })</td></tr>
</table>

引數	描述
order by	**order by** 是 **load** 陳述式處理常駐表格記錄前將這些記錄排序所用的子句。常駐表格可按照一或多個欄位以遞增或遞減順序排序。排序主要以數值進行，其次以國家排序順序進行。只有在資料來源是常駐表格時，才能使用此子句。 排序欄位可指定常駐表格按照哪些欄位排序。可以按照常駐表格的名稱或編號 (第一個欄位的編號是 1) 指定欄位。 orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } *sortorder* 以 *asc* 表示遞增，以 *desc* 表示遞減。如果未指定 *sortorder*，將會採用 *asc*。 *fieldname*、*path*、*filename* 和 *aliasname* 是指出個別名稱代表什麼意義的文字字串。來源表格的任何欄位均可做為 *fieldname*。不過，透過 as 子句 (*aliasname*) 建立的欄位若不在範圍內，無法在同一個 **load** 陳述式內使用。

注意！ 如果並未透過 **from**、**inline**、**resident**、**from_field**、**延伸** 或 **autogenerate** 子句指定資料的來源，將從後面的 **SELECT** 或 **LOAD** 陳述式結果載入資料。後面的陳述式不應該有前置詞。

範例：

載入不同的檔案格式
載入具有預設選項的分隔符號資料檔：

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

從物件庫連線 (DataFiles) 中載入分隔符號資料檔案：

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

從物件庫連線 (DataFiles) 中載入所有分隔符號資料檔案：

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

載入分隔符號檔案，將逗點指定為分隔符號，並具有內嵌的標籤：

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

載入分隔符號檔案，將定位點指定為分隔符號，並具有內嵌的標籤：

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

載入含有內嵌標頭的 dif 檔：

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

從沒有標頭的固定記錄檔載入三個欄位：

```
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);
```

載入 QVX 檔，指定絕對路徑：

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

載入 Web 檔案
從 Web 檔案資料連線中設定的預設 URL 載入：

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

從特定 URL 載入，並覆寫 Web 檔案資料連線中設定的 URL：

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

使用貨幣符號擴充從變數中設定的特定 URL 載入：

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

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

選取特定欄位，重新命名並計算欄位
僅從分隔符號檔案載入三個特定欄位：

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

載入沒有標籤的檔案時，將第一個欄位重新命名為 A，並將第二個欄位重新命名為 B。

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

載入 Name 作為 FirstName、空格字元和 LastName 的串連：

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

載入 Quantity、Price 和 Value (Quantity 和 Price 的產品)：

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

選取特定記錄
僅載入唯一的記錄，將捨棄重複的記錄：

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

僅載入欄位 Litres 擁有 0 以上值的記錄：

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

載入未在檔案上的資料以及自動產生的資料
載入含有內嵌資料的表格，兩個欄為名為 CatID 和 Category：

```
LOAD * Inline

[CatID, Category

0,Regular
```

```
1,Occasional

2,Permanent];
```

載入具有內嵌資料的表格，三個欄位名為 UserID、Password 和 Access：

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

A, ABC456, User

B, VIP789, Admin];
```

載入含有 10 000 列的表格。欄位 A 將包含讀取記錄數 (1,2,3,4,5...)，並且欄位 B 將包含 0 與 1 之間的隨機數字：

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

autogenerate 之後可以加括弧，但是並非必要。

從先前載入的表格載入資料
首先我們載入分隔表格檔案，並將它命名為 tab1：

```
tab1:

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

從已載入的 tab1 表格中，載入欄位作為 tab2：

```
tab2:

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

從已載入的表格 tab1 中載入欄位，但是僅載入 A 大於 B 的記錄：

```
tab3:

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

從已載入的表格 tab1 中載入欄位，按 A 排序：

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

從已載入的表格 tab1 中載入欄位，按第一個欄位排序，然後按第二個欄位排序：

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

從已載入的表格 tab1 中載入欄位，按 C 遞減順序排序，然後按 B 遞增順序排序，然後按第一個欄位遞減順序排序：

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

從先前載入的欄位載入資料

從先前已載入的表格 Characters 中載入欄位 Types 以作為 A：

```
LOAD A from_field (Characters, Types);
```

從後面的表格中載入資料 (前置載入)

從後面以 **SELECT** 陳述式載入的 Table1 中，載入 A、B 以及已計算的欄位 X 和 Y：

```
LOAD A, B, if(C>0,'positive','negative') as X, weekday(D) as Y;

SELECT A,B,C,D from Table1;
```

將資料分組

載入按 ArtNo 分組 (彙總) 的欄位：

```
LOAD ArtNo, round(Sum(TransAmount),0.05) as ArtNoTotal from table.csv group by ArtNo;
```

載入按 Week 和 ArtNo 分組 (彙總) 的欄位：

```
LOAD Week, ArtNo, round(Avg(TransAmount),0.05) as WeekArtNoAverages from table.csv group by
Week, ArtNo;
```

重複讀取一個記錄

在此範例中，我們擁有一個輸入檔 Grades.csv，其中包含在一個欄位中壓縮的每一個學生等級：

```
Student,Grades

Mike,5234

John,3345

Pete,1234

Paul,3352
```

等級為 1-5，代表主題 Math、English、Science 和 History。我們可以使用 **while** 子句讀取每一筆記錄數次，並將 **IterNo( )** 函數用作計數器，將等級分為數個值。在每一筆記錄中，使用 **Mid** 函數擷取等級並儲存在 Grade 中，使用 **pick** 函數選取主題並儲存在 Subject 中。最終 **while** 子句包含測試，檢查是否已讀取所有等級 (在此案例中每個學生四個等級)，這表示應該讀取接下來的學生記錄。

```
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));
```

結果是包含此資料的表格:

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

從分析連線載入
已使用下列樣本資料。

```
Values:
Load
 Rand() as A,
 Rand() as B,
 Rand() as C
AutoGenerate(50);
```

使用函數載入資料

在這些範例中,假定我們具有名稱為 *P* 的分析連線外掛程式,其中包含自訂函數 *Calculate (Parameter1, Parameter2)*。函數傳回表格*結果*,其中包含欄位 *Field1* 和 *Field2*。

```
Load * Extension P.Calculate( Values{A, C} );
```
載入在將欄位 A 和 C 傳送至函數時傳回的所有欄位。

```
Load Field1 Extension P.Calculate( Values{A, C} );
```
將欄位 A 和 C 傳送至函數時僅載入 Field1 欄位。

```
Load * Extension P.Calculate( Values );
```
載入在將欄位 A 和 B 傳送至函數時傳回的所有欄位。若沒有指定欄位,A 和 B 會作為表格中的第一順序使用。

```
Load * Extension P.Calculate( Values {C, C});
```
載入在將欄位 C 傳送至函數的兩個參數時傳回的所有欄位。

```
Load * Extension P.Calculate( Values {String(A), Mixed(B)});
```
載入在將強制作為字串的欄位 A 和強制作為數字的 B 傳送至函數時傳回的所有欄位。

評估指令碼以載入資料

```
Load A as A_echo, B as B_echo Extension R.ScriptEval( 'q;', Values{A, B} );
```
在傳送 A 和 B 的值時依指令碼 q 載入傳回的表格。

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{A, B} );
```
在傳送 A 和 B 的值時依儲存於 My_R_Script 變數的指令碼載入傳回的表格。

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{B as D, *} );
```
在傳送重新命名為 D、A 和 C 的 B 值時時依儲存於 My_R_Script 變數的指令碼載入傳回的表格。使用 * 傳送剩餘的未參考欄位。

DataFiles 連線的檔案延伸區分大小寫。例如:.qvd。

格式規格項目

各個格式規格項目都會定義表格檔案的特定屬性:

fspec-item ::= [**ansi | oem | mac | UTF-8 | Unicode | txt | fix | dif | biff | ooxml | html | xml | kml | qvd | qvx | parquet | delimiter is** char | **no eof | embedded labels | explicit labels | no labels | table is** [tablename] | **header is** n | **header is** line | **header is** n **lines** | **comment is** string | **record is** n | **record is** line | **record is** n **lines** | **no quotes |msq** | **URL is** string | **userAgent is** string]

字元集

字元集是 **LOAD** 陳述式的檔案規範,可定義檔案中使用的字元集。

ansi、**oem** 和 **mac** 規範用於 QlikView 中且仍然有效。但是,當使用 Qlik Sense 建立 **LOAD** 陳述式時,將無法產生這些規範。

語法:

```
utf8 | unicode | ansi | oem | mac | codepage is
```

引數:

引數

引數	描述
utf8	UTF-8 字元集
unicode	Unicode 字元集
ansi	Windows, 字碼頁 1252
oem	DOS、OS/2、AS400 及其他
mac	字碼頁 10000
codepage is	使用 **codepage** 規範,可以將任何 Windows 字碼頁用作 *N*。

限制：

從 **oem** 字元集的轉換未針對 macOS 予以實施。如果未指定任何項目，Windows 將假設使用字碼頁 1252。

範例：

```
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)
```

另請參見：

Load (page 146)

表格格式

表格格式是 **LOAD** 陳述式的檔案規格，可定義檔案類型。若未指定任何項目，將採用 *.txt* 檔案。

表格格式類型

類型	描述
txt	在分隔文字檔中，表格中的資料行會以分隔符號字元分隔。
fix	在固定記錄檔案中，每個欄位的寬度恰好是固定字元數。 通常，許多固定記錄長度檔包含換行符號分隔的記錄，但是有更進階的選項來指定記錄大小 (以位元組為單位)，或者使用 **Record is** 跨越多行。 *如果資料包含多位元組字元，則欄位分隔符號可能變成不對齊，因為格式是基於固定的長度 (以位元組數為單位)。*
dif	在 *.dif* 檔案中 (Data Interchange Format)，會使用特別的格式來定義表格。
biff	Qlik Sense 也可以使用 *biff* 格式 (Binary Interchange File Format)，解譯標準 Excel 檔案中的資料。
ooxml	Excel 2007 以及更新版本使用 ooxml *.xslx* 格式。
html	如果表格是 html 頁面或檔案的一部分，應使用 html。
xml	xml (可延伸標記語言) 是一種常見標記語言，用來使用文字格式代表資料結構。
qvd	*qvd* 格式是從 Qlik Sense 應用程式匯出的專屬 QVD 檔案格式。
qvx	*qvx* 是用於 Qlik Sense 高效能輸出的檔案/串流格式。
parquet	Apache Parquet 是單欄式儲存格式，對於儲存和查詢大量資料集的效率很高。

Delimiter is

若是分隔的表格檔案，可透過 **delimiter is** 規範來指定任意的分隔符號。此規範只與分隔的 .txt 檔有關。

語法：

```
delimiter is char
```

引數：

引數

引數	描述
char	指定 127 ASCII 字元中的單一字元。

此外，可使用下列值：

選用值

值	描述
'\t'	代表定位點符號，包含引號與否均可。
'\\'	代表反斜線 (\) 字元。
'spaces'	代表一或多個空格的所有組合。ASCII 值在 32 以下的不可列印字元 (CR 和 LF 除外) 將解譯為空格。

如果未指定任何字元，將採用 **delimiter is ','**。

範例：

```
LOAD * from a.txt (utf8, txt, delimiter is ',' , embedded labels);
```

另請參見：

Load (page 146)

No eof

no eof 規範用來在載入分隔的 **.txt** 檔案時忽略檔案結尾的字元。

語法：

```
no eof
```

如果使用 **no eof** 規範，則具有程式碼點 26 的字元 (否則代表檔案結束) 會被捨棄，並且可以是欄位值的一部分。

此規範只與分隔的文字檔有關。

範例:

```
LOAD * from a.txt (txt, utf8, embedded labels, delimiter is ' ', no eof);
```

另請參見:

Load (page 146)

Labels

Labels 是 **LOAD** 陳述式的檔案規範,會定義可以在檔案的什麼位置找到檔案名稱。

語法:

```
embedded labels|explicit labels|no labels
```

在檔案的不同位置都可以找到欄位名稱。如果第一筆記錄包含欄位名稱,應該使用 **embedded labels**。如果找不到欄位名稱,則應該使用 **no labels**。在 *dif* 檔案中,有時候會使用含明確欄位名稱的個別標頭區段。在這種情況下,應該使用 **explicit labels**。若未指定任何項目,也將同樣對 *dif* 檔案採用 **embedded labels**。

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)
```

另請參見:

Load (page 146)

Header is

指定表格檔案的標頭大小。可以透過 **header is** 規範指定任意的標頭長度。標頭是 Qlik Sense 不使用的文字區段。

語法:

```
header is n
```

```
header is line
```

```
header is n lines
```

標頭長度可以透過位元組 (**header is n**) 或以線條 (**header is line** 或 **header is n lines**) 指定。**n** 必須是正整數,代表標頭長度。如果未指定,將採用 **header is 0**。**header is** 規範只與表格檔案有關。

範例:

這是包含標頭文字行的資料來源表格範例,這些文字行不應該由 Qlik Sense 解譯為資料。

```
*Header line
Col1,Col2
a,B
c,D
```

使用 **header is 1 lines** 規範，第一行將不會作為資料載入。在範例中，**embedded labels** 規範告知 Qlik Sense 將第一個非排除行解譯為包含欄位標籤。

```
LOAD Col1, Col2
FROM 'lib://files/header.txt'
(txt, embedded labels, delimiter is ',', msq, header is 1 lines);
```

結果是包含兩個欄位的表格 (Col1 和 Col2)。

另請參見：

Load (page 146)

Record is

若是固定記錄長度檔案，必須透過 **record is** 規範指定記錄長度。

語法：

```
Record is n
Record is line
Record is n lines
```

引數：

引數

引數	描述
n	指定記錄長度 (以位元組為單位)。
line	指定記錄長度 (作為一行)。
n lines	指定記錄長度 (以行數為單位)，其中 n 是代表記錄長度的正整數。

限制：

record is 規範只與 **fix** 檔案有關。

另請參見：

Load (page 146)

Quotes

Quotes 是 **LOAD** 陳述式的檔案規範，可定義是否可使用引號，以及引號與分隔符號的優先順序。僅適用於文字檔。

語法:

```
no quotes
```

```
msq
```

如果省略該規範,可使用標準引號,如 " " 或 ' ',但僅限這些引號是欄位值的第一個和最後一個非空白字元的情況下。

引數:

引數

引數	描述
no quotes	如果文字檔中不接受引號,則予以使用。
msq	用來指定現代樣式引號,可允許欄位中的多行內容。包含行結尾字元的欄位必須以雙引號括住。 msq 選項的其中一個限制是,當欄位內容的第一個或最後一個字元出現單一雙引號 (") 字元時,將會解譯為多行內容的開始或結束,因此可能會在載入的資料集中造成意外的結果。在此情況下,您應改用標準引號,並省略規範。

XML

載入 xml 檔案時使用此指令碼規範。**XML** 規範的有效選項在語法中列出。

您無法在 Qlik Sense 中載入 DTD 檔案。

語法:

```
xmlsimple
```

另請參見:

Load (page 146)

KML

載入 KML 檔案以在地圖視覺化中使用時使用此指令碼規範。

語法:

```
kml
```

KML 檔案可以表示以多邊形、線路資料 (例如軌道或道路) 呈現的區域資料 (例如,國家或地區),或以 [經度, 緯度] 形式呈現的點資料 (例如,城市或地方)。

URL is

此指令碼規範用來在載入 Web 檔案時，設定 Web 檔案資料連線的 URL。

語法：

```
URL is string
```

引數：

引數

引數	描述
string	指定要載入的檔案 URL。這將會覆寫所使用的 Web 檔案連線中設定的 URL。

限制：

URL is 規範只與 Web 檔案有關。您需要使用現有的 Web 檔案資料連線。

另請參見：

Load (page 146)

userAgent is

此指令碼規範用來在載入 Web 檔案時設定瀏覽器使用者代理程式。

語法：

```
userAgent is string
```

引數：

引數

引數	描述
string	指定瀏覽器使用者代理程式字串。這將會覆寫預設瀏覽器使用者代理程式 "Mozilla/5.0"。

限制：

userAgent is 規範只與 Web 檔案有關。

另請參見：

Load (page 146)

Let

let 陳述式是 **set** 陳述式的補集，用來定義指令碼變數。相對於 **set** 陳述式，**let** 陳述式會先在指令碼執行時間評估 '=' 右邊的運算式，然後才將運算式指派給變數。

語法：

```
Let variablename=expression
```

範例與結果：

範例	結果
Set x=3+4; Let y=3+4; z=$(y)+1;	$(x) 將評估為 '3+4' $(y) 將評估為 ' 7' $(z) 將評估為 ' 8' 注意 **Set** 和 **Let** 陳述式之間的差異。**Set** 陳述式將字串 '3+4' 指派至變數，而 **Let** 陳述式則評估字串並將 7 指派至變數。
Let T=now( );	$(T) 將得到目前時間的值。

Loosen Table

使用 **Loosen Table** 陳述式，可以在指令碼執行期間將一或多個 Qlik Sense 內部資料表明確宣告為鬆散耦合表格。當表格鬆散耦合時，會移除表格中欄位值之間的所有關聯。可以透過將鬆散耦合表格的每個欄位作為獨立、未連接的表格載入，從而實現類似的效果。在測試以暫時隔離資料結構的不同部分時，鬆散耦合可能有用。鬆散耦合的表格可以在表格檢視器中使用虛線進行識別。在指令碼中使用一或多個 **Loosen Table** 陳述式將造成 Qlik Sense 忽略指令碼執行前使表格成為鬆散耦合表格的任何設定。

語法：

```
Loosen Tabletablename [ , tablename2 ...]
```

```
Loosen Tablestablename [ , tablename2 ...]
```

可使用以下任一語法：**Loosen Table** 或 **Loosen Tables**。

*當 Qlik Sense 發現資料結構有循環參照，但是無法在指令碼中以互動或明確的方式解除宣告為鬆散耦合表格的表格，將會強制其他一或多個表格成為鬆散耦合表格，直到沒有循環參照為止。發生這種情況時，**循環警告**對話方塊將發出警告。*

範例：

```
Tab1:

SELECT * from Trans;
```

```
Loosen Table Tab1;
```

Map

map ... using 陳述式用來將特定欄位值或運算式對應到特定對應表的值。對應表是透過 **Mapping** 陳述式建立的。

語法：

```
Map fieldlist Using  mapname
```

對於 **Map ... Using** 陳述式之後載入的欄位，會進行自動對應，直到指令碼結束或出現 **Unmap** 陳述式為止。

對應是這一串連鎖事件中最後執行的動作，之後欄位即儲存在 Qlik Sense 內部表格中。這表示並不是每次在運算式中遇到欄位名稱就會進行對應，而是在要根據欄位名稱將值儲存在內部表格時才會進行對應。如果需要運算式層級的對應，必須改用 **Applymap()** 函數。

引數：

引數

引數	描述
fieldlist	以逗號分隔的欄位清單，其應從指令碼的此點中進行對應。使用 * 作為表示所有欄位的欄位清單。可在欄位名稱中使用萬用字元 * 與 ?。使用萬用字元時，可能需要為欄位名稱加上引號。
mapname	之前在 **mapping load** 或 **mapping select** 陳述式中讀取的對應表格名稱。

範例與結果：

範例	結果
Map Country Using Cmap;	啟用使用對應 Cmap，對應欄位 Country。
Map A, B, C Using X;	啟用使用對應 X，對應欄位 A、B 和 C。
Map * Using GenMap;	使用 GenMap 啟用所有欄位的對應。

NullAsNull

NullAsNull 陳述式會關閉先前由 **NullAsValue** 陳述式設定將 NULL 變成字串值的轉換。

語法：

```
NullAsNull *fieldlist
```

NullAsValue 陳述式可做為參數，而且在指令碼中使用 **NullAsValue** 或 **NullAsNull** 陳述式即可多次開啟或關閉。

引數：

引數

引數	描述
*fieldlist	逗號分隔的欄位清單，其 **NullAsNull** 應該開啟。使用 * 作為表示所有欄位的欄位清單。可在欄位名稱中使用萬用字元 * 與 ?。使用萬用字元時，可能需要為欄位名稱加上引號。

範例：

```
NullAsNull A,B;
LOAD A,B from x.csv;
```

NullAsValue

NullAsValue 陳述式會指定應該將出現 NULL 的哪些欄位轉換為值。

語法：

```
NullAsValue *fieldlist
```

Qlik Sense 預設認定 NULL 值是遺漏或未定義的實體。不過，某些資料庫內容則暗示將 NULL 值認定為特殊值，而不是單純的遺漏值。透過 **NullAsValue** 陳述式，可暫停一般不允許 NULL 值連結到其他 NULL 值的情況。

NullAsValue 陳述式可做為參數，用在後續的載入陳述式中。透過 **NullAsNull** 陳述式可以將它再次關閉。

引數：

引數

引數	描述
*fieldlist	逗號分隔的欄位清單，其 **NullAsValue** 應該開啟。使用 * 作為表示所有欄位的欄位清單。可在欄位名稱中使用萬用字元 * 與 ?。使用萬用字元時，可能需要為欄位名稱加上引號。

範例：

```
NullAsValue A,B;
Set NullValue = 'NULL';
LOAD A,B from x.csv;
```

Qualify

Qualify 陳述式用來切換限定欄位名稱，亦即欄位名稱將獲得表格名稱做為前置詞。

語法：

```
Qualify *fieldlist
```

使用 **qualify** 陳述式可擱置不同表格中擁有相同名稱之欄位的自動聯結，此陳述式可透過其表格名稱限定欄位名稱。如果已限定，當在表格中找到欄位名稱時，將予以重新命名。新的名稱格式將為 *tablename.fieldname*。*Tablename* 相等於目前表格的標籤，或如果沒有標籤存在，則相等於 **LOAD** 與 **SELECT** 陳述式中出現在 **from** 之後的名稱。

會針對 **qualify** 陳述式後載入的所有欄位進行限定。

一開始執行指令碼時，預設一律會關閉限定。可使用 **qualify** 陳述式隨時啟動欄位名稱的限定。可使用 **Unqualify** 陳述式隨時關閉限定。

__qualify__ 陳述式不得與部分重新載入一起使用！

引數：

引數

引數	描述
*fieldlist	逗號分隔的欄位清單，其限定應該開啟。使用 * 作為表示所有欄位的欄位清單。可在欄位名稱中使用萬用字元 * 與 ?。使用萬用字元時，可能需要為欄位名稱加上引號。

Example 1:

```
Qualify B;

LOAD A,B from x.csv;

LOAD A,B from y.csv;
```

兩個表格 **x.csv** 與 **y.csv** 僅透過 **A** 建立關聯。三個欄位將產生：A、x.B、y.B。

Example 2:

在不熟悉的資料庫中，通常較實用的方法是先確定只有一或少數欄位互相關聯，如下列範例所示：

```
qualify *;
```

```
unqualify TransID;

SQL SELECT * from tab1;

SQL SELECT * from tab2;

SQL SELECT * from tab3;
```

只有 **TransID** 用於表格 *tab1*、*tab2* 與 *tab3* 之間的關聯。

Rem

rem 陳述式用來將備註或註解插入指令碼中，或暫時停用指令碼陳述式，但不移除陳述式。

語法：

```
Rem string
```

rem 與下一個分號 **;** 之間的所有文字都會視為註解。

有兩個替代方法可用來在指令碼中建立註解：

1. 可將有問題的區段放置在 **/*** 與 ***/** 之間，在指令碼的任意處建立註解，但兩個引號之間除外。
2. 在指令碼中輸入 **//** 時，同一列右邊之後的所有文字都會變成註解。(請注意，例外的 //: 可能會用來表示部分網際網路位址。)

引數：

引數

引數	描述
string	任意文字。

範例：

```
Rem ** This is a comment **;
/* This is also a comment */
// This is a comment as well
```

Rename

Rename 指令碼關鍵字可以用來重新命名已載入的表格或欄位。

Rename field

此指令碼函數會在載入一或多個現有 Qlik Sense 欄位之後，為其重新命名。

不建議將變數命名為與 Qlik Sense 中某個欄位或函數相同的名稱。

可使用以下任一語法：**rename field** 或 **rename fields**。

語法:

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

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

引數:

引數	描述
mapname	先前所載入的對應表名稱, 該表包含一組或多組舊欄位名稱與新欄位名稱。
oldname	舊欄位名稱。
newname	新欄位名稱。

限制:

您不能重新命名兩個欄位為相同的名稱。

Example 1:

```
Rename Field XAZ0007 to Sales;
```

Example 2:

```
FieldMap:

Mapping SQL SELECT oldnames, newnames from datadictionary;

Rename Fields using FieldMap;
```

Rename table

此指令碼函數會在載入一或多個現有 Qlik Sense 內部表格之後, 為其重新命名。

可使用以下任一語法:**rename table** 或 **rename tables**。

語法:

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

引數:

引數

引數	描述
mapname	先前所載入的對應表名稱, 該表包含一組或多組舊表格名稱與新表格名稱。
oldname	舊表格名稱。
newname	新表格名稱

限制:

不能將兩個不同名稱的表格重新命名為相同的名稱。如果您嘗試使用與現有表格相同的名稱重新命名圖表,指令碼將會產生錯誤。

Example 1:

```
Tab1:
SELECT * from Trans;
Rename Table Tab1 to Xyz;
```

Example 2:

```
TabMap:
Mapping LOAD oldnames, newnames from tabnames.csv;
Rename Tables using TabMap;
```

Search

Search 陳述式用來在智慧型搜尋中包括或排除欄位。

語法:

```
Search Include *fieldlist
Search Exclude *fieldlist
```

您可以使用數個 Search 陳述式,來調整要包括之欄位的選擇範圍。從上到下評估陳述式。

引數:

引數

引數	描述
*fieldlist	要在智慧型搜尋中納入或排除的逗點分隔欄位清單。使用 * 作為表示所有欄位的欄位清單。可在欄位名稱中使用萬用字元 * 與 ?。使用萬用字元時,可能需要為欄位名稱加上引號。

範例:

搜尋範例

陳述式	描述
`Search Include *;`	在智慧型搜尋中納入所有欄位。
`Search Exclude [*ID];`	從智慧型搜尋中排除以 ID 結尾的所有欄位。
`Search Exclude '*ID';`	從智慧型搜尋中排除以 ID 結尾的所有欄位。
`Search Include ProductID;`	在智慧型搜尋中納入欄位 ProductID。

這三個陳述式按此順序的結合結果是從智慧型搜尋中排除以 ID 結束的所有欄位 (ProductID 除外)。

Section

透過 **section** 陳述式, 可以定義是否應該將後續的 **LOAD** 和 **SELECT** 陳述式視為資料或存取權限的定義。

語法:

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

如果未指定任何字元, 將採用 **section application**。**section** 定義的效力會持續到進行新的 **section** 陳述式為止。

範例:

```
Section access;
```

```
Section application;
```

Select

透過標準 SQL **SELECT** 陳述式可選取來自 ODBC 資料來源或 OLE DB 提供者的欄位。然而, 是否接受 **SELECT** 陳述式取決於所使用的 ODBC 驅動程式或 OLE DB 提供者。使用 **SELECT** 陳述式需要對來源有開放的資料連線。

語法:

```
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 ]
```

此外, 數個 **SELECT** 陳述式有時可透過使用 **union** 運算子串連成一個。

```
selectstatement Union selectstatement
```

SELECT 陳述式是由 ODBC 驅動程式或 OLE DB 提供者解譯, 因此, 根據 ODBC 驅動程式或 OLE DB 提供者的能力, 可能發生一般 SQL 語法的偏差, 例如:

- 有時候不允許 **as**, 例如 *aliasname* 必須緊接在 *fieldname* 後。
- 如果使用 *aliasname*, 則有時會強制使用 **as**。

- 有時不支援 **distinct**、**as**、**where**、**group by**、**order by** 或 **union**。
- ODBC 驅動程式有時不接受所有上列的不同引號。

> *這不是 SQL **SELECT** 陳述式的完整描述！例如，**SELECT** 陳述式可為巢狀、可在一個 **SELECT** 陳述式中進行數個聯結、運算式中允許的函數有時會非常大量等等。*

引數：

引數

引數	描述
distinct	**distinct** 是一個述詞，於僅載入一次所選欄位中的重複值組合時才會使用。
distinctrow	**distinctrow** 是一個述詞，於僅載入一次來源表格中的重複記錄時才會使用。

fieldlist	`fieldlist ::= (*| field ) {, field }`
	要選取的欄位清單。使用 * 做為欄位清單表示表格中的所有欄位。
	`fieldlist ::= field {, field }`
	一或多個欄位的清單，以逗號分隔。
	`field ::= ( fieldref| expression ) [as aliasname ]`
	運算式可以是基於一或數個其他欄位的數值或字串函數。常接受的一些運算子和函數包括：+、-、*、/、& (字串串連)、sum(fieldname)、count(fieldname)、avg(fieldname) (average)、month(fieldname) 等。如需詳細資訊，請參閱 ODBC 的文件。
	`fieldref ::= [ tablename. ] fieldname`
	tablename 與 **fieldname** 是名副其實的文字字串。如果其包含空格，則必須由直雙引號括住。
	as 子句用來指派欄位的新名稱。

from	`tablelist ::= table {, table }`
	要從其中選取欄位的表格清單。
	`table ::= tablename [ [as ] aliasname ]`
	tablename 不一定會在引號內。

where	**where** 是指明選項中是否應該包含記錄的子句。 **criterion** 是一個邏輯運算式，有些時候可能會非常複雜。接受的部分運算子為：數值運算子和函數、**=**、**<>** 或 **#(**不等於)、**>**、**>=**、**<**、**<=**、**and**、**or**、**not**、**exists**、**some**、**all**、**in** 以及新的 **SELECT** 陳述式。如需詳細資訊，請參閱 ODBC 驅動程式或 OLE DB 提供者的文件。
group by	**group by** 是一個子句，用來將數筆記錄彙總 (分組) 成一筆記錄。在一個群組內的某些欄位中，所有記錄都必須擁有相同的值，否則只能從運算式內使用該欄位，例如做為加總或平均。基於一或數個欄位的運算式會定義於欄位符號的運算式中。
having	**having** 是一個子句，用來限定群組，方式與如何使用 **where** 子句來限定記錄類似。
order by	**order by** 是一個子句，用來表示 **SELECT** 陳述式所產生表格的排序順序。

引數	描述
join	**join** 是一個限定詞，表示是否要將數個表格聯結成一個表格。如果欄位名稱與表格名稱包含空格或國家字元集的字母，則必須以引號括住。當 Qlik Sense 自動產生指令碼時，使用的引號會是 ODBC 驅動程式或 OLE DB 提供者的慣用引號，並且它們是由**Connect** 陳述式中，資料來源的資料來源定義所指定。

Example 1:

```
SELECT * FROM `Categories`;
```

Example 2:

```
SELECT `Category ID`, `Category Name` FROM `Categories`;
```

Example 3:

```
SELECT `Order ID`, `Product ID`,
`Unit Price` * Quantity * (1-Discount) as NetSales
FROM `Order Details`;
```

Example 4:

```
SELECT `Order Details`.`Order ID`,
Sum(`Order Details`.`Unit Price` * `Order Details`.Quantity) as `Result`
FROM `Order Details`, Orders
where Orders.`Order ID` = `Order Details`.`Order ID`
group by `Order Details`.`Order ID`;
```

Set

set 陳述式用來定義指令碼變數。這些可用來取代字串、路徑、磁碟機等。

語法：

```
Set variablename=string
```

Example 1:

```
Set FileToUse=Data1.csv;
```

Example 2:

```
Set Constant="My string";
```

Example 3:

```
Set BudgetYear=2012;
```

Sleep

sleep 陳述式會暫停所指定時間的指令碼執行。

語法：

```
Sleep n
```

引數：

引數	描述
n	以毫秒說明，其中 *n* 是不超過 3600000 (即 1 個小時) 的正整數。該值可以是運算式。

Example 1:

```
Sleep 10000;
```

Example 2:

```
Sleep t*1000;
```

SQL

SQL 陳述式可讓您透過 ODBC 或 OLE DB 連線傳送任意 SQL 命令。

語法：

```
SQL sql_command
```

如果 Qlik Sense 已經以唯讀模式開啟 ODBC 連線，傳送可更新資料庫的 SQL 陳述式將傳回錯誤。

允許下列語法：

```
SQL SELECT * from tab1;
```

且為了保持一致，是 **SELECT** 的慣用語法。不過，SQL 前置詞仍是 **SELECT** 陳述式的選用項目。

引數：

引數	描述
sql_command	有效的 SQL 指令。

Example 1:

```
SQL leave;
```

Example 2:

```
SQL Execute <storedProc>;
```

SQLColumns

sqlcolumns 陳述式會傳回一組描述 ODBC 或 OLE DB 資料來源資料行的欄位，該欄位已經進行 **connect**。

語法：

```
SQLcolumns
```

這些欄位可與由 **sqltables** 及 **sqltypes** 命令產生的欄位結合，以方便綜觀指定資料庫。這十二個標準欄位為：

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

COLUMN_NAME

DATA_TYPE

TYPE_NAME

PRECISION

LENGTH

SCALE

RADIX

NULLABLE

REMARKS

如需這些欄位的詳細描述，請參閱《ODBC 參考手冊》。

範例：

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLcolumns;
```

部分 ODBC 驅動程式可能不支援此指令。部分 ODBC 驅動程式可能產生其他欄位。

SQLTables

sqltables 陳述式會傳回一組描述 ODBC 或 OLE DB 資料來源表格的欄位，該欄位已經進行 **connect**。

語法：

```
SQLTables
```

這些欄位可與由 **sqlcolumns** 及 **sqltypes** 命令產生的欄位結合，以方便綜觀指定資料庫。這五個標準欄位為：

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

TABLE_TYPE

REMARKS

如需這些欄位的詳細描述，請參閱《ODBC 參考手冊》。

範例：

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTables;
```

部分 ODBC 驅動程式可能不支援此指令。部分 ODBC 驅動程式可能產生其他欄位。

SQLTypes

sqltypes 陳述式會傳回一組描述 ODBC 或 OLE DB 資料來源類型的欄位，該欄位已經進行 **connect**。

語法：

```
SQLTypes
```

這些欄位可與由 **sqlcolumns** 及 **sqltables** 命令產生的欄位結合，以方便綜觀指定資料庫。這十五個標準欄位為：

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

如需這些欄位的詳細描述,請參閱《ODBC 參考手冊》。

範例:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTypes;
```

部分 ODBC 驅動程式可能不支援此指令。部分 ODBC 驅動程式可能產生其他欄位。

Star

用來代表資料庫中某個欄位之所有值集合的字串可以透過 **star** 陳述式進行設定。會影響後續的 **LOAD** 與 **SELECT** 陳述式。

語法:

```
Star is[ string ]
```

引數:

引數

引數	描述
string	任意文字。請注意,如果字串包含空白,則必須以引號括住。 如果未指定任何項目,則採用 **star is;**,即沒有可用的星號符號,除非特別指定。此定義的效力會持續到進行新的 **star** 陳述式為止。

若使用區段存取,則 **Star is** 陳述式不建議用於指令碼的資料部分 (在**區段應用程式**之下)。不過,星號字元在指令碼的**區段存取**部分的受保護欄位完全受到支援。在此情況下,您不需要使用明確的 **Star is** 陳述式,因為這一直隱含在區段存取中。

限制

- 您無法透過索引鍵欄位使用星號字元;亦即連結表格的欄位。
- 您無法透過任何受到 **Unqualify** 陳述式影響的欄位使用星號字元,因為這會影響連結表格的欄位。
- 您無法透過非邏輯表格使用星號字元,例如資訊載入表格或對應載入表格。
- 在區段存取的減少欄位 (連結至資料的欄位) 中使用星號字元時,這呈現區段存取中列於此欄位的值。這不會呈現可能存在於資料中、但沒有列於區段存取中的其他值。
- 您無法透過受到**區段存取**區域之外任何形式的資料減少所影響的任何欄位使用星號字元。

範例

下例是描述區段存取的資料載入指令碼的摘要。

```
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

];
```

以下適用於:

- *Star* 號為「*」。
- 使用者 *ADMIN* 可查看所有欄位。不會省略任何內容。
- 使用者 *USER1* 無法查看欄位 *SALES*。
- 使用者 *USER2* 無法查看欄位 *WAREHOUSE* 。
- 使用者 *USER3* 無法查看欄位 *EMPLOYEES*。
- 兩次新增使用者 *USER4* 到解決方案中以忽略該使用者的兩個欄位 *SALES* 和 *WAREHOUSE*。

- *USER5* 新增了「*」，這代表 OMIT 中的所有列出欄位都無法使用，亦即使用者 *USER5* 無法查看欄位 *SALES*、*WAREHOUSE* 和 *EMPLOYEES*，但此使用者可以查看欄位 *ORDERS*。

Store

Store 陳述式建立 QVD、Parquet、CSV 或 TXT 檔案。

語法：

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

該陳述式將建立明確命名的 QVD、Parquet 或文字檔案。

陳述式只能從一個資料表格匯出欄位。如果要匯出數個表格中的欄位，則必須在指令碼中提前進行明確 join，以建立應該匯出的資料表格。

文字值會以 UTF-8 格式匯出到 CSV 檔案中。可指定分隔符號，請參閱 **LOAD**。在 CSV 檔案中使用 **store** 陳述式不支援 BIFF 匯出。

> *在某些資料格式不正確的情況下，將會以雙引號括起欄位，以確保正確解譯資料。例如，若欄位包含引號、逗號、空格或分行符號等字元，就可能發生此情況。*

引數：

儲存命令引數

引數	描述
*fieldlist::= (* | field) { , field })*	要選取的欄位清單。使用 * 做為欄位清單表示所有欄位。 *field::= fieldname [***as** *aliasname]* *fieldname* 是與 *table* 中欄位名稱相同的文字。(請注意，欄位名稱如果包含空格或其他非標準字元，則必須以一般雙引號或方括弧括住。) *aliasname* 是在所產生 QVD 或 CSV 檔案中要使用之欄位的替代名稱。
table	代表已經載入的表格 (用作資料來源) 的指令碼標籤。

引數	描述
filename	包括通向現有資料夾資料連線的有效路徑的目標檔案的名稱。 **範例：*'lib://Table Files/target.qvd'*** 在舊式指令碼模式下，還支援下列路徑格式： • 絕對路徑 **範例：*c:\data\sales.qvd*** • 與 Qlik Sense 應用程式工作目錄相關。 **範例：*data\sales.qvd*** 如果省略路徑，Qlik Sense 將在 **Directory** 陳述式指定的目錄中儲存該檔案。如果沒有 **Directory** 陳述式，則 Qlik Sense 會在工作目錄 (*C:\Users\{user}\Documents\Qlik\Sense\Apps*) 中儲存該檔案。 •
*format-spec ::=***(** (**txt** | **qvd** | **parquet**), **compression is** *轉碼器***)**	您可以將格式規格設定為其中任一個檔案格式。若省略格式規格，會採用 **qvd**。 • CSV 和 TXT 檔案的 **txt**。 • **qvd** 適用於 QVD 檔案。 • **parquet** 適用於 Parquet 檔案。 若您使用 **parquet**，您也可以設定要搭配 **compression is** 使用哪個壓縮轉碼器。若您沒有透過 **compression is** 指定壓縮轉碼器，則會使用 `snappy`。提供下列壓縮設定： • `uncompressed` • `snappy` • `gzip` • `lz4` • `brotli` • `zstd` • `lz4_hadoop` 範例： `Store mytable into [lib://DataFiles/myfile.parquet] (parquet, compression is lz4);`

範例:

```
Store mytable into xyz.qvd (qvd);

Store * from mytable into 'lib://FolderConnection/myfile.qvd';

Store Name, RegNo from mytable into xyz.qvd;

Store Name as a, RegNo as b from mytable into 'lib://FolderConnection/myfile.qvd';

Store mytable into myfile.txt (txt);

Store mytable into myfile.parquet (parquet);

Store * from mytable into 'lib://FolderConnection/myfile.qvd';
```

DataFiles 連線的檔案延伸區分大小寫。例如:.qvd。

儲存在 Parquet 檔案中

Parquet 是強力輸入的檔案格式，其中每個欄位包含單一特定資料類型 (例如 in32、雙重、時間戳記或文字)。Qlik Sense 將內部資料儲存為鬆散輸入的雙重內容，其中來自不同來源的資料可以混合到相同的欄位中。由於 Parquet 的每個欄位中只能儲存一個雙重部分，知道每個欄位包含的內容很重要。依照預設，Qlik Sense 使用欄位類型決定應如何儲存欄位。以特定格式在 Parquet 檔案中儲存資料時，您必須指定欄位在載入時是什麼資料類型。若您嘗試以 Parquet 檔案將資料儲存到不相容的欄位中，例如在文字欄位中儲存數字或在時間戳記欄位中儲存文字，結果會是 Null 值。

載入您打算以 Parquet 儲存的資料時，可以變更預設行為。您可以格式化以變更資料類型，或標記以在 Parquet 強制特定欄類型。

格式化資料以儲存於 Parquet

您可以使用 Qlik Sense 格式化功能以分類資料。例如，**Text()**、**Num()**、**Interval()** 或 **Timestamp()** 可以在以 Parquet 儲存資料時強制資料格式。Qlik Sense 可以將資料儲存到將近 20 個資料類型，視欄位屬性和自動欄位標記而定。如需詳細資訊，請參閱 *解譯函數 (page 1192)*

範例: 以 Num() 和 Text() 格式化資料

下列範例展示準備資料以儲存於 Parquet。**Num()** 套用至數字欄位。**Text()** 套用至文字和混合欄位。在混合情況下，**Text()** 可防止像 Parquet 中的數字欄位那樣來處理，並有變更為 Null 值的文字值。

```
Data:
LOAD * INLINE [
num, text, mixed
123.321, abc, 123
456.654, def, xyz
789.987, ghi, 321
];

Format:
NoConcatenate
```

```
LOAD num, text, Text(mixed) as mixed RESIDENT Data;
STORE Format INTO [lib://DataFiles/Tmp.parquet] (parquet);
```

標記資料以儲存於 Parquet

您使用 `$parquet` 標記來標記資料，以便在以 Parquet 儲存資料時強制特定欄類型。新增對應的控制標記以強制每個資料類型。例如，若要在 Parquet 將欄位儲存為 INT32，在載入指令碼中以 `$parquet-int32` 標記。根據資料類型，將會儲存雙重資料的字串或數字呈現。

下列 Parquet 控制標記可用來標記欄位，以儲存於 Parquet 檔案。

Parquet 控制標記

控制標記	雙值	實際類型	邏輯類型	轉換類型
`$parquet-boolean`	數字	布林	無	無
`$parquet-int32`	數字	INT32	無	無
`$parquet-int64`	數字	INT64	無	無
`$parquet-float`	數字	浮動	無	無
`$parquet-double`	數字	雙重	無	無
`$parquet-bytearray`	字串	BYTE_ARRAY	無	UTF8
`$parquet-bytearrayfix`	數字	FIXED_LEN_BYTE_ARRAY	無	小數
`$parquet-decimal`	數字	INT64	小數	小數
`$parquet-date`	數字	INT32	日期	日期
`$parquet-time`	數字	INT64	時間	TIME_MICROS
`$parquet-timestamp`	數字	INT64	時間戳記	TIMESTAMP_MICROS
`$parquet-string`	字串	BYTE_ARRAY	字串	UTF8
`$parquet-enum`	字串	BYTE_ARRAY	ENUM	ENUM
`$parquet-interval`	數字	FIXED_LEN_BYTE_ARRAY	間隔	間隔
`$parquet-json`	字串	BYTE_ARRAY	JSON	JSON
`$parquet-bson`	字串	BYTE_ARRAY	BSON	BSON
`$parquet-uuid`	字串	FIXED_LEN_BYTE_ARRAY	UUID	無

範例：標記資料以儲存於 Parquet

在此範例中，兩個標記用來定義 Parquet 的資料。欄位 *num* 以 `$parquet-int32` 標記，以定義為將會在 Parquet 設定為 INT32 的數字欄位。

```
Data:
LOAD * INLINE [
num, text,
123.321, abc
456.654, def
789.987, ghi
];
```

```
TAG num WITH '$parquet-int32';
STORE Data INTO [lib://DataFiles/Tmp.parquet] (parquet);
```

Table/Tables

Table 和 **Tables** 指令碼關鍵字使用在 **Drop**、**Comment** 和 **Rename** 陳述式中，以及在 **Load** 陳述式中用作格式規範。

Tag

此指令碼陳述式提供將標記指派至一個或多個欄位或表格的方式。如果已嘗試將欄位或表格標記為不呈現在應用程式中，則會忽略該標記。若發現欄位或標記名稱衝突，會使用最後一個值。

語法：

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

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

```
Tag table tablelist with tagname
```

引數

引數	描述
fieldlist	應標記的一個或數個欄位，位於以逗號分隔的清單中。
mapname	之前在 **mapping Load** 或 **mapping Select** 陳述式中載入的對應表格名稱。
tablelist	應標記並以逗號分隔的表格清單。
tagname	應套用到欄位的標記名稱。

Example 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
];
tag fields using tagmap;
```

Example 2:

```
tag field Alpha with 'MyTag2';
```

Trace

使用 **trace** 陳述式可將字串寫入**指令碼執行進度**視窗以及指令碼記錄檔。在偵錯方面非常有幫助。使用在 **trace** 陳述式前計算的 $-expansions 變數，即可自訂訊息。

語法:

```
Trace string
```

Example 1:

可在載入 'Main' 表格的 LOAD 陳述式後使用下列陳述式。

```
Trace Main table loaded;
```

這將會在指令碼執行對話方塊和記錄檔中顯示文字 'Main table loaded' (主表格已載入)。

Example 2:

可在載入 'Main' 表格的 LOAD 陳述式後使用下列陳述式。

```
Let MyMessage = NoOfRows('Main') & ' rows in Main table';

Trace $(MyMessage);
```

這將會顯示文字,呈現指令碼執行對話方塊和記錄檔中的列數,例如 '265,391 rows in Main table' (主表格中的 265,391 列)。

Unmap

Unmap 陳述式會為後續載入的欄位,停用先前 **Map ... Using** 陳述式指定的欄位值對應。

語法:

```
Unmap *fieldlist
```

引數:

引數

引數	描述
*fieldlist	以逗號分隔的欄位清單,其不得再從指令碼的此點中對應。使用 * 作為表示所有欄位的欄位清單。可在欄位名稱中使用萬用字元 * 與 ?。使用萬用字元時,可能需要為欄位名稱加上引號。

範例與結果:

範例	結果
Unmap Country;	停用欄位 Country 的對應。
Unmap A, B, C;	停用欄位 A、B 及 C 的對應。
Unmap * ;	停用所有欄位的對應。

Unqualify

Unqualify 陳述式用來關閉先前由 **Qualify** 陳述式開啟的欄位名稱限定。

語法：

```
Unqualify *fieldlist
```

引數：

引數

引數	描述
*fieldlist	逗號分隔的欄位清單，其限定應該開啟。使用 * 作為表示所有欄位的欄位清單。可在欄位名稱中使用萬用字元 * 與 ?。使用萬用字元時，可能需要為欄位名稱加上引號。 如需進一步資訊，請參閱 **Qualify** 陳述式文件。

Example 1:

在不熟悉的資料庫中，通常較實用的方法是先確定只有一或少數欄位互相關聯，如下列範例所示：

```
qualify *;
unqualify TransID;
SQL SELECT * from tab1;
SQL SELECT * from tab2;
SQL SELECT * from tab3;
```

首先，為所有欄位開啟限定。
然後為 **TransID** 關閉限定。
只有 **TransID** 用於表格 *tab1*、*tab2* 與 *tab3* 之間的關聯。將以表格名稱限定所有其他欄位。

Untag

此指令碼陳述式提供從欄位或表格移除標記的方式。如果已嘗試將欄位或表格取消標記為不呈現在應用程式中，則會忽略該取消標記。

語法：

```
Untag [field|fields] fieldlist with tagname
```

```
Untag [field|fields] fieldlist using mapname
```

```
Untag table tablelist with tagname
```

引數：

引數

引數	描述
fieldlist	應移除標記的一個或數個欄位，位於以逗號分隔的清單中。
mapname	先前在對應 **LOAD** 或對應 **SELECT** 陳述式中載入的對應表名稱。
tablelist	應取消標記並以逗號分隔的表格清單。

引數	描述
tagname	應從欄位移除的標記名稱。

Example 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
];
Untag fields using tagmap;
```

Example 2:

```
Untag field Alpha with MyTag2;
```

3.4 工作目錄

如果您參考指令碼陳述式中的檔案，並省略路徑，則 Qlik Sense 會搜尋下列順序的檔案：

1. **Directory** 陳述式指定的目錄 (僅在舊版指令碼模式下受支援)。
2. 如果沒有 **Directory** 陳述式，則 Qlik Sense 會在工作目錄中進行搜尋。

Qlik Sense Desktop工作目錄

在 Qlik Sense Desktop 中，工作目錄是 *C:\Users\{user}\Documents\Qlik\Sense\Apps*。

Qlik Sense工作目錄

在 Qlik Sense 伺服器安裝中，工作目錄在 Qlik Sense Repository Service 中指定，預設情況下它位於 *C:\ProgramData\Qlik\Sense\Apps*。如需更多資訊，請參閱 Qlik 管理主控台 說明。

4 在資料載入編輯器中使用變數

Qlik Sense 中的變數是一個容器，儲存靜態值或計算，例如數值或英數值。當您在應用程式中使用變數時，對變數進行的任何變更都會套用至使用變數的每個位置。您可以在變數概述或在指令碼中使用資料載入編輯器定義變數。您可在資料載入指令碼中使用 **Let** 或 **Set** 陳述式設定變數值。

編輯工作表時，您也可以使用來自變數概覽中的 Qlik Sense 變數。

4.1 概述

如果變數值的第一個字元為等號「=」，Qlik Sense 會嘗試將值評估為公式 (Qlik Sense 運算式)，然後顯示或傳回結果，而非傳回實際的公式文字。

使用時，變數值會替代其變數。變數可用於指令碼中以用作貨幣符號展開，也可用於各種控制陳述式中。如果相同的字串在指令碼中重複多次 (如路徑)，這就非常適用。

Qlik Sense 會在指令碼執行的開端，設定某些特別的系統變數，而不論其先前的值為何。

4.2 定義變數

變數提供了儲存靜態值或計算結果的能力。定義變數時，請使用下列語法：

```
set variablename = string
```

或

```
let variable = expression
```

Set 陳述式用於指派字串。它將等號右側的文字指派給變數。**Let** 陳述式在指令碼執行時計算等號右側的運算式，並將運算式的結果指派給變數。

變數會區分大小寫。

不建議將變數命名為與 Qlik Sense 中某個欄位或函數相同的名稱。

範例：

`set x = 3 + 4;` // 變數將取得字串 '3 + 4' 作為值。

`let x = 3 + 4;` // 傳回 7 作為值。

`set x = Today();` // 傳回 'Today()' 作為值。

`let x = Today();` // 傳回今天的日期作為值，例如 ‘9/27/2021’。

4.3 刪除變數

若您從指令碼中移除變數並重新載入資料，則變數會留在應用程式中。如果您要從應用程式中完全移除變數，則還必須從變數對話方塊中刪除變數。

4.4 載入變數值作為欄位值

如果您想要在 **LOAD** 陳述式中載入變數值作為欄位值，且美元展開的結果是文字而不是數字或運算式，那麼您需要在單括號中附上展開變數。

範例：

此範例將包含指令碼錯誤清單的系統變數載入表格中。您可能注意到，**If** 子句中的 ScriptErrorCount 展開不需要引號，但 ScriptErrorList 的展開需要引號。

```
IF $(ScriptErrorCount) >= 1 THEN

   LOAD '$(ScriptErrorList)' AS Error AutoGenerate 1;
END IF
```

4.5 變數計算

有數種方法可搭配使用變數與 Qlik Sense 中計算的值，則結果會取決於您定義它的方式，以及您在運算式中調用它的方式。

在此範例中，我們載入部分內嵌資料：

```
LOAD * INLINE [
    Dim, Sales
    A, 150
    A, 200
    B, 240
    B, 230
    C, 410
    C, 330
];
```

讓我們定義兩個變數：

```
Let vSales  = 'Sum(Sales)' ;
Let vSales2  = '=Sum(Sales)' ;
```

在第二個變數中，我們在運算式之前新增等號。這將導致在變數展開之前對它進行計算，並且會評估運算式。

如果您按原樣使用 vSales 變數，那麼在量值中該結果將是字串 Sum(Sales)，也就是說，不會執行任何計算。

如果您新增貨幣符號展開，並在運算式中調用 $(vSales)，則會展開變數，並顯示 Sales 的總和。

最終，如果您調用 $(vSales2)，則會在變數展開之前對它進行計算。這表示所顯示的結果是 Sales 的總和。使用 =$(vSales) 與 =$(vSales2) 作為量值運算式之間的差異可在此圖表中看到，顯示結果：

結果

Dim	$(vSales)	$(vSales2)
A	350	1560
B	470	1560
C	740	1560

如您可以看到的，$(vSales) 產生維度值的部分總和，而 $(vSales2) 結果產生總和。

下列指令碼變數可用：

- *錯誤變數 (page 257)*
- *數字解譯變數 (page 196)*
- *系統變數 (page 189)*
- *值處理變數 (page 195)*

4.6 系統變數

系統變數，其中部分是系統定義的，提供系統和 Qlik Sense 應用程式的相關資訊。

系統變數概述

概述之後，會進一步描述部分函數。對於那些函數，您可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

CreateSearchIndexOnReload

此變數定義是否應在資料重新載入期間建立搜尋索引檔案。

```
CreateSearchIndexOnReload
```

Floppy

傳回所找到第一個軟碟機的磁碟機代號，通常是 *a:*。這是系統定義的變數。

```
Floppy
```

標準模式下不支援這個變數。

CD

傳回所找到第一個光碟機的磁碟機代號。如果找不到 CD-ROM，則會傳回 *c:*。這是系統定義的變數。

```
CD
```

標準模式下不支援這個變數。

HidePrefix

以此文字字串開頭的所有欄位名稱將予以隱藏，隱藏方式與系統欄位一樣。這是使用者定義的變數。

```
HidePrefix
```

HideSuffix

以此文字字串結尾的所有欄位名稱將予以隱藏，隱藏方式與系統欄位一樣。這是使用者定義的變數。

```
HideSuffix
```

Include

Include/Must_Include 變數會指定包含應該納入指令碼並且評估為指令碼之文字的檔案。這不是用來新增資料。您可以將指令碼的各部分儲存在獨立文字檔中，並在數個應用程式中重複使用。這是使用者定義的變數。

```
$(Include=filename)
$(Must_Include=filename)
```

OpenUrlTimeout

此變數會定義 Qlik Sense 在從 URL 來源 (例如HTML 頁面) 取得資料時應遵守的逾時 (以秒為單位)。如果省略，逾時約為 20 分鐘。

```
OpenUrlTimeout
```

QvPath

會傳回 Qlik Sense 執行檔的瀏覽字串。這是系統定義的變數。

```
QvPath
```

標準模式下不支援這個變數。

QvRoot

會傳回 Qlik Sense 執行檔的根目錄。這是系統定義的變數。

```
QvRoot
```

標準模式下不支援這個變數。

QvWorkPath

將瀏覽字串傳回目前 Qlik Sense 應用程式。這是系統定義的變數。

```
QvWorkPath
```

標準模式下不支援這個變數。

QvWorkRoot

傳回目前 Qlik Sense 應用程式的根目錄。這是系統定義的變數。

```
QvWorkRoot
```

標準模式下不支援這個變數。

StripComments

如果此變數設為 0, 將會禁止在指令碼中移除 /*..*/ 與 // 註解。若未定義此變數, 將一律移除註解。

```
StripComments
```

Verbatim

通常所有欄位值在載入 Qlik Sense 資料庫前, 會先自動移除前置與尾端的空白字元 (ASCII 32)。將此變數設為 1 可暫停移除空白字元。永遠不會移除定位點分隔 (ASCII 9) 和硬空格 (ANSI 160) 字元。

```
Verbatim
```

WinPath

將瀏覽字串傳回 Windows。這是系統定義的變數。

```
WinPath
```

標準模式下不支援這個變數。

WinRoot

傳回 Windows 的根目錄。這是系統定義的變數。

```
WinRoot
```

標準模式下不支援這個變數。

CollationLocale

指定用於排序順序和搜尋相符的地區設定。該值是地區設定的文化名稱, 例如 'en-US'。這是系統定義的變數。

```
CollationLocale
```

CreateSearchIndexOnReload

此變數定義是否應在資料重新載入期間建立搜尋索引檔案。

語法:

```
CreateSearchIndexOnReload
```

您可以定義是否應在資料重新載入期間建立搜尋索引檔案，或是否應在使用者的首次搜尋要求後建立這些檔案。在資料重新載入期間建立搜尋索引檔案的優勢在於，您不必耗費在首位使用者進行搜尋所經歷的等待時間。需要與搜尋索引建立所需之更長的資料重新載入時間在一起進行權衡。

如果省略此變數，則在資料重新載入期間不會建立搜尋索引檔案。

對於工作階段應用程式，不會在資料重新載入期間建立搜尋索引檔案，無論此變數的設定為何。

Example 1: 在資料重新載入期間建立搜尋索引欄位

```
set CreateSearchIndexOnReload=1;
```

Example 2: 在首次搜尋要求後建立搜尋索引欄位

```
set CreateSearchIndexOnReload=0;
```

HidePrefix

以此文字字串開頭的所有欄位名稱將予以隱藏，隱藏方式與系統欄位一樣。這是使用者定義的變數。

語法:

```
HidePrefix
```

範例:

```
set HidePrefix='_' ;
```

若使用此陳述式，則隱藏系統欄位時，以底線開頭的欄位名稱將不會顯示在欄位名稱清單中。

HideSuffix

以此文字字串結尾的所有欄位名稱將予以隱藏，隱藏方式與系統欄位一樣。這是使用者定義的變數。

語法:

```
HideSuffix
```

範例:

```
set HideSuffix='%';
```

若使用此陳述式，則隱藏系統欄位時，以百分比符號結尾的欄位名稱將不會顯示在欄位名稱清單中。

Include

Include/Must_Include 變數會指定包含應該納入指令碼並且評估為指令碼之文字的檔案。這不是用來新增資料。您可以將指令碼的各部分儲存在獨立文字檔中，並在數個應用程式中重複使用。這是使用者定義的變數。

> *此變數僅在標準模式下支援資料夾資料連線。*

語法：

```
$(Include=filename)
```

```
$(Must_Include=filename)
```

存在兩個版本的變數：

- **Include** 在找不到檔案時不會產生錯誤，而是將自動失敗。
- **Must_Include** 在找不到檔案時產生錯誤。

如果您未指定路徑，而檔案名稱將與 Qlik Sense 應用程式工作目錄相關。您也可以指定一個絕對檔案路徑，或指向 lib:// 資料夾連線的路徑。請勿在等號之前或之後放置空格字元。

> *建構 **set Include =**filename 不適用。*

範例：

```
$(Include=abc.txt);
```

```
$(Must_Include=lib://DataFiles/abc.txt);
```

限制

Windows 與 Linux 之下的 UTF-8 編碼檔案之間的交叉相容性有限。

可以選擇使用 UTF-8 與 BOM (位元組順序標記)。BOM 會在啟動檔案時影響沒有預期非 ASCII 位元組的軟體中 UTF-8 的使用，但仍可以處理文字串流。

- Windows 系統在 UTF-8 中使用 BOM 以確認檔案為 UTF-8 編碼，雖然在位元組儲存方面並沒有模糊性。
- Unix/Linux 對 Unicode 使用 UTF-8，但不使用 BOM，因為這會影響命令檔案的語法。

這對 Qlik Sense 有一些含意。

- 在 Windows 中，任何以 UTF-8 BOM 開頭的檔案都被視為 UTF-8 指令碼檔案。否則就會假設為 ANSI 編碼。
- 在 Linux 中，系統預設 8 位元程式碼頁面為 UTF-8。因此雖然沒有包含 BOM，但 UTF-8 仍可運作。

因此，無法保證可攜性。不一定可以在 Linux 能解譯的 Windows 建立檔案，反之亦然。由於對 BOM 的處理方式不同，兩個系統之間對於 UTF-8 編碼檔案沒有交叉相容性。

OpenUrlTimeout

此變數會定義 Qlik Sense 在從 URL 來源 (例如HTML 頁面) 取得資料時應遵守的逾時 (以秒為單位)。如果省略，逾時約為 20 分鐘。

語法:

```
OpenUrlTimeout
```

範例:

```
set OpenUrlTimeout=10;
```

StripComments

如果此變數設為 0，將會禁止在指令碼中移除 /*..*/ 與 // 註解。若未定義此變數，將一律移除註解。

語法:

```
StripComments
```

某些資料庫驅動程式使用 /*..*/ 作為 **SELECT** 陳述式中的最佳化提示。在此情況下，在將 **SELECT** 陳述式傳送至資料庫驅動程式之前，註解不應移除。

建議在需要的陳述式後立即將此變數重設為 1。

範例:

```
set StripComments=0;
SQL SELECT * /* <optimization directive> */ FROM Table ;
set StripComments=1;
```

Verbatim

通常所有欄位值在載入 Qlik Sense 資料庫前，會先自動移除前置與尾端的空白字元 (ASCII 32)。將此變數設為 1 可暫停移除空白字元。永遠不會移除定位點分隔 (ASCII 9) 和硬空格 (ANSI 160) 字元。

語法:

```
Verbatim
```

範例:

```
set Verbatim = 1;
```

4.7 值處理變數

本節描述用來處理 NULL 及其他值的變數。

值處理變數概述

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

NullDisplay

定義的符號將取代來自 ODBC 及最低層級資料連接器的所有 NULL 值。這是使用者定義的變數。

```
NullDisplay
```

NullInterpret

當定義的符號出現在文字檔、Excel 檔或 inline 陳述式時，會解譯為 NULL。這是使用者定義的變數。

```
NullInterpret
```

NullValue

如果使用 **NullAsValue** 陳述式，則已定義的符號會將 **NullAsValue** 指定欄位中的所有 NULL 值取代為指定的字串。

```
NullValue
```

OtherSymbol

定義符號使其成為在 **LOAD/SELECT** 陳述式前的「所有其他值」。這是使用者定義的變數。

```
OtherSymbol
```

NullDisplay

定義的符號將取代來自 ODBC 及最低層級資料連接器的所有 NULL 值。這是使用者定義的變數。

語法：

```
NullDisplay
```

範例：

```
set NullDisplay='<NULL>';
```

NullInterpret

當定義的符號出現在文字檔、Excel 檔或 inline 陳述式時，會解譯為 NULL。這是使用者定義的變數。

語法：

```
NullInterpret
```

範例：

```
set NullInterpret=' ';
set NullInterpret =;
```

將不會針對 Excel (但會針對 CSV 文字檔) 中的空白值傳回 NULL 值

```
set NullInterpret ='';
```

將針對 Excel 中的空白值，傳回 NULL 值。

NullValue

如果使用 **NullAsValue** 陳述式，則已定義的符號會將 **NullAsValue** 指定欄位中的所有 NULL 值取代為指定的字串。

語法：

```
NullValue
```

範例：

```
NullAsValue Field1, Field2;
set NullValue='<NULL>';
```

OtherSymbol

定義符號使其成為在 **LOAD/SELECT** 陳述式前的「所有其他值」。這是使用者定義的變數。

語法：

```
OtherSymbol
```

範例：

```
set OtherSymbol='+';
LOAD * inline
[X, Y
a, a
b, b];
LOAD * inline
[X, Z
a, a
+, c];
```

現在，欄位值 Y='b' 將透過其他符號連結至 Z='c'。

4.8 數字解譯變數

數字解譯變數由系統定義。變數包含在載入指令碼的上層，而且可以在指令碼執行時套用數字格式設定。可以刪除、編輯或複製這些變數。

數字解譯變數是在建立新應用程式時，會自動按照目前作業系統區域設定自動產生。在 Qlik Sense Desktop 中，這是根據電腦作業系統的設定而定。在Qlik Sense 中，這是根據安裝 Qlik Sense 的伺服器作業系統而定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

貨幣格式設定

MoneyDecimalSep

此定義的小數點分隔符號可取代區域設定所設定的貨幣小數點符號。

```
MoneyDecimalSep
```

MoneyFormat

此定義的符號可取代區域設定所設定的貨幣符號。

```
MoneyFormat
```

MoneyThousandSep

此定義的千位分隔符號可取代區域設定所設定的的貨幣位數分組符號。

```
MoneyThousandSep
```

數字格式設定

DecimalSep

此定義的小數點分隔符號可取代區域設定所設定的小數符號。

```
DecimalSep
```

ThousandSep

此定義的千位分隔符號可取代作業系統的位數分組符號 (區域設定)。

```
ThousandSep
```

NumericalAbbreviation

數字縮寫可設定要在數字的比例前置詞使用哪個縮寫，例如 M 用於 mega (兆) 或 million (百萬) (10^6)，而 µ 用於 micro (百萬分之一) (10^{-6}).

```
NumericalAbbreviation
```

時間格式設定

DateFormat

此環境變數定義作為應用程式中預設使用的日期格式。該格式用來解譯和格式化日期。若未定義變數，將會在指令碼執行時擷取作業系統區域設定的日期格式。

```
DateFormat
```

TimeFormat

此定義的格式可取代作業系統的時間格式 (區域設定)。

`TimeFormat`

TimestampFormat

此定義的格式可取代作業系統的日期和時間格式 (區域設定)。

`TimestampFormat`

MonthNames

此定義的格式可取代區域設定的月份名稱慣例。

`MonthNames`

LongMonthNames

此定義的格式可取代區域設定中的月份長名稱慣例。

`LongMonthNames`

DayNames

此定義的格式可取代區域設定所設定的星期幾名稱慣例。

`DayNames`

LongDayNames

此定義的格式可取代區域設定中的星期幾長名稱慣例。

`LongDayNames`

FirstWeekDay

定義要用作一週第一天的日子的整數。

`FirstWeekDay`

BrokenWeeks

此設定會定義週是否中斷。

`BrokenWeeks`

ReferenceDay

該設定會定義一月份中哪天設為參考日來定義第一週。

`ReferenceDay`

FirstMonthOfYear

此設定會定義哪個月要作為年度的第一個月，可用於定義使用月份位移的會計年度，例如開始於 4 月 1 日。

此設定目前未被使用，但已被保留以供日後使用。

有效設定為 1 (一月份) 至 12 (十二月份)。預設設定是 1。

語法:

```
FirstMonthOfYear
```

範例:

```
Set FirstMonthOfYear=4; //Sets the year to start in April
```

BrokenWeeks

此設定會定義週是否中斷。

語法:

```
BrokenWeeks
```

在 Qlik Sense 中,建立應用程式時會擷取地區設定,對應的設定會儲存在指令碼中,作為環境變數。

北美應用程式開發人員通常會在指令碼中取得 `Set BrokenWeeks=1;`,對應至中斷的週。歐洲應用程式開發人員通常會在指令碼中取得 `Set BrokenWeeks=0;`,對應至未中斷的週。

未中斷的週表示:

- 在某些年中,第 1 週從 12 月開始,而在其他年中,上一年的最後一週延續至 1 月。
- 根據 ISO 8601,第 1 週至少有四天總是在 1 月。在 Qlik Sense 中,這可以使用 `ReferenceDay` 變數設定。

中斷的週表示:

- 一年的最後一週不會延續至 1 月。
- 第 1 週從 1 月 1 日開始,並且在大多數情況下,不是完整的週。

可使用下列值:

- 0 (=使用未中斷的週)
- 1 (= 使用中斷的週)

區域設定

除非另有說明,否則此主題中的範例皆使用下列日期格式:MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素,您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式,以滿足您的需求。或者,您可以在載入指令碼中變更格式,以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典,資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例:

若您想要週和週數的 ISO 設定,確認指令碼中有下列內容:

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;   //(use unbroken weeks)
Set ReferenceDay=4;
```

若您想要 US 設定，確認指令碼中有下列內容：

```
Set FirstWeekDay=6;
Set BrokenWeeks=1;   //(use broken weeks)
Set ReferenceDay=1;
```

DateFormat

此環境變數定義作為應用程式中預設值使用的日期格式，並按照傳回 `date()` 和 `date#()` 等函數的日期。格式用來解譯和格式化日期。若未定義變數，將會在指令碼執行時擷取區域設定所設定的日期格式。

語法：

```
DateFormat
```

DateFormat 函數範例

範例	結果
`Set DateFormat='M/D/YY'; //(US format)`	這樣的 `DateFormat` 函數使用方式將日期定義為 US 格式，即月/日/年。
`Set DateFormat='DD/MM/YY'; //(UK date format)`	這樣的 `DateFormat` 函數使用方式將日期定義為 UK 格式，即日/月/年。
`Set DateFormat='YYYY/MM/DD'; //(ISO date format)`	這樣的 `DateFormat` 函數使用方式將日期定義為 ISO 格式，即年/月/日。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 系統變數預設

載入指令碼和結果

概述

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 日期資料集。
- DateFormat 函數，這將會使用美國日期格式。

在此範例中，資料集會載入到名為「Transactions」的表格中。這包括 date 欄位。使用 US DateFormat 定義。此模式將在載入文字日期時用於隱含文字到日期轉換。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- month

建立此量值：

=sum(amount)

結果表格

日期	月	=sum(amount)
01/01/2022	一月	1000
02/01/2022	二月	2123
03/01/2022	三月	4124
04/01/2022	四月	2431

DateFormat 定義 MM/DD/YYYY 用於隱含的文字到日期轉換，因此 date 欄位可正確解譯為日期。相同格式用來顯示日期，如結果表格中所示。

範例 2 – 變更系統變數

載入指令碼和結果

概述

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 來自先前範例的相同資料集。
- DateFormat 函數,這將會使用「DD/MM/YYYY」格式。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- month

建立此量值:

=sum(amount)

結果表格

日期	月	=sum(amount)
01/01/2022	一月	1000
02/01/2022	一月	2123
03/01/2022	一月	4124
04/01/2022	一月	2431

因為 DateFormat 定義已設定為「DD/MM/YYYY'」,您可以看見第一個「/」符號之後的兩位數字已解譯為月份,產生來自一月的所有記錄。

範例 3 – 日期解譯

載入指令碼和結果

概述

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 資料集中的日期為數字格式。
- DateFormat 變數,這將會使用「DD/MM/YYYY」格式。
- date() 變數。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- month

建立此量值:

=sum(amount)

結果表格

日期	月	=sum(amount)
06/03/2022	六月	1000
06/04/2022	六月	2123

日期	月	=sum(amount)
06/05/2022	六月	4124
06/07/2022	六月	2431

在載入指令碼中，您使用 `date()` 函數將數值日期轉換為日期格式。因為您沒有提供指定格式作為函數中的第二引數，所以會使用 `DateFormat`。這產生使用格式「MM/DD/YYYY」的日期欄位。

範例 4 – 外國日期格式

載入指令碼和結果

概述

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 日期資料集。
- `DateFormat` 變數使用「DD/MM/YYYY」格式但以正斜線取消註解。

載入指令碼

```
// 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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- `date`
- `month`

建立此量值：

```
=sum(amount)
```

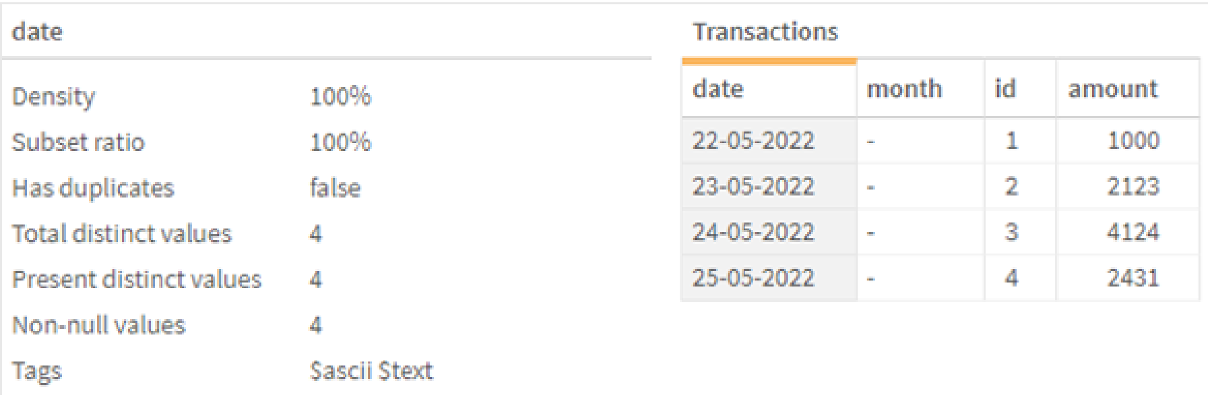
結果表格

日期	月	=sum(amount)
22-05-2022	-	1000
23-05-2022	-	2123
24-05-2022	-	4124
25-05-2022	-	2431

在初始載入指令碼中，正在使用的 `DateFormat` 是預設「MM/DD/YYYY」。因為交易資料集中的 `date` 欄位不屬於此格式，所以欄位不會解譯為日期。這顯示在結果表格中，其中 `month` 欄位值為 Null。

您可以檢查 `date` 欄位的「標記」屬性，以在資料模型檢視器中驗證解譯的資料類型。

`Transactions` 表格的預覽。對指示文字輸入資料未隱含轉換至日期/時間戳記的 `date` 欄位記下「標記」。



date

Density	100%
Subset ratio	100%
Has duplicates	false
Total distinct values	4
Present distinct values	4
Non-null values	4
Tags	$ascii $text

Transactions

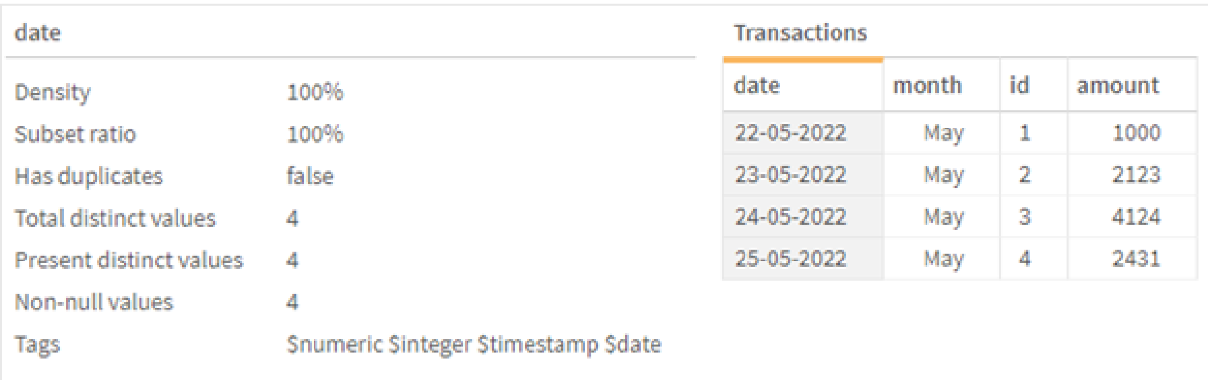
date	month	id	amount
22-05-2022	-	1	1000
23-05-2022	-	2	2123
24-05-2022	-	3	4124
25-05-2022	-	4	2431

這可以透過啟用 `DateFormat` 系統變數來解決：

```
// SET DateFormat='DD/MM/YYYY';
```

移除雙正斜線並載入資料。

`Transactions` 表格的預覽。對指示文字輸入資料已隱含轉換至日期/時間戳記的 `date` 欄位記下「標記」。



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

此定義的格式可取代區域設定所設定的星期幾名稱慣例。

語法:

`DayNames`

修改變數時,需要分號 ; 才能分隔個別的值 。

DayName 函數範例

函數範例	結果定義
`Set DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';`	這種 DayNames 函數的使用方式以縮寫形式定義日子名稱。
`Set DayNames='M;Tu;W;Th;F;Sa;Su';`	這種 DayNames 函數的使用方式以第一個字母定義日子名稱。

`DayNames` 函數通常用於組合下列函數:

相關函數

函數	互動
weekday (page 1009)	要作為欄位值傳回 `DayNames` 的指令碼函數。
Date (page 1162)	要作為欄位值傳回 `DayNames` 的指令碼函數。
LongDayNames (page 216)	`DayNames` 的長形式值。

區域設定

除非另有說明,否則此主題中的範例皆使用下列日期格式:MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素,您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式,以滿足您的需求。或者,您可以在載入指令碼中變更格式,以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典,資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 - 系統變數預設

載入指令碼和結果

概覽

在此範例中,以 MM/DD/YYYY 格式設定資料集中的日期。

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 具有日期的資料集將會載入到名稱為 Transactions 的表格中。
- date 欄位。
- 預設 DayNames 定義。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- dayname

建立此量值:

sum(amount)

結果表格

日期	dayname	sum(amount)
01/01/2022	星期六	1000
02/01/2022	星期二	2123
03/01/2022	星期二	4124
04/01/2022	星期五	2431

在載入指令碼中, WeekDay 函數作為提供的引數搭配 date 欄位使用。在結果表格中, 此 WeekDay 函數的輸出以 DayNames 定義的格式顯示一週中的日子。

範例 2 - 變更系統變數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。使用與第一個範例相同的資料集和情境。

不過，在指令碼開頭，已修改 `DayNames` 定義以使用南非文縮寫的一週中日子。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- dayname

建立此量值：

`sum(amount)`

結果表格

日期	dayname	sum(amount)
01/01/2022	Sa	1000
02/01/2022	Di	2123
03/01/2022	Di	4124
04/01/2022	Vr	2431

在結果表格中，此 `WeekDay` 函數的輸出以 `DayNames` 定義的格式顯示一週中的日子。

重要的是，應記住若 `DayNames` 的語言按照此範例修改，則 `LongDayNames` 仍會包含以英文表示的一週中的日子。若兩種變數都用於應用程式中，則這也需要修改。

範例 3 – 日期函數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 具有日期的資料集將會載入到名稱為 `Transactions` 的表格中。
- `date` 欄位。
- 預設 `DayNames` 定義。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- dayname

建立此量值:

`sum(amount)`

結果表格

日期	dayname	sum(amount)
01/01/2022	星期六	1000
02/01/2022	星期二	2123

日期	dayname	sum(amount)
03/01/2022	星期二	4124
04/01/2022	星期五	2431

使用預設 `DayNames` 定義。在載入指令碼中，`Date` 函數作為第一個引數搭配 `date` 欄位使用。第二個引數是 `WWW`。此格式將結果轉換為儲存在 `DayNames` 定義中的值。這顯示在結果表格的輸出。

DecimalSep

此定義的小數點分隔符號可取代區域設定所設定的小數符號。

只要遇到可辨識的數字模式，Qlik Sense 就會自動將文字解譯為數字。`ThousandSep` 和 `DecimalSep` 系統變數判定在將文字剖析為數字時套用的模式標記。`ThousandSep` 和 `DecimalSep` 變數設定以前端圖表和表格視覺化數字內容時的預設數字格式模式。亦即，這直接影響任何前端運算式的**數字格式**選項。

假設千分位分隔符號是逗號「,」，而小數點分隔符號是「.」，則以下模式範例會隱含轉換為對等數值：

`0,000.00`

`0000.00`

`0,000`

以下模式範例會讓文字保持不變；亦即不會轉換為數字：

`0.000,00`

`0,00`

語法：

`DecimalSep`

函數範例

範例	結果
`Set DecimalSep='.';`	將「.」設定為小數點分隔符號。
`Set DecimalSep=',';`	將「,」設定為小數點分隔符號。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 – 設定數字分隔符號變數對不同輸入資料的影響

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 以不同的格式模式設定總和的總和和日期資料集。
- 名稱為 `Transactions` 的表格。
- 設定為「.」的 `DecimalSep` 變數。
- 設定為「,」的 `ThousandSep` 變數。
- 設定為「|」字元以分隔一行中不同欄位的 `delimiter` 變數。

載入指令碼

```
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 '|');
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度 `amount`。

建立此量值:

```
=sum(amount)
```

結果表格

金額	=Sum(amount)
總計	20814.7086
1.000-45	
3.756,178	
4124,35	
23.344	23.344
2431.36	2431.36
2431.84	2431.84
3,454.356	3454.356
3554.284	3554.284
4132.5246	4132.5246
4,787	4787

未解譯為數字的任何值保持為文字，依照預設會靠左對齊。任何成功轉換的值會靠右對齊，保留原始輸入格式。

運算式欄顯示對等數值，依照預設僅以小數點分隔符號「.」格式化。這可以在運算式設定中使用**數字格式**下拉式清單設定覆寫。

FirstWeekDay

定義要用作一週第一天的日子的整數。

語法：

```
FirstWeekDay
```

根據日期和時間的國際標準呈現方式 ISO 8601，星期一是一週的第一天。星期一在一些國家也作為一週的第一天使用，例如英國、法國、德國和瑞典。

在其他國家，例如美國和加拿大，星期日被視為一週的開始。

在 Qlik Sense 中，建立應用程式時會擷取地區設定，對應的設定會儲存在指令碼中，作為環境變數。

北美應用程式開發人員通常會在指令碼中取得 `Set FirstWeekDay=6;`，對應至星期日。歐洲應用程式開發人員通常會在指令碼中取得 `Set FirstWeekDay=0;`，對應至星期一。

可為 FirstWeekDay 設定的值

值	日
0	星期一
1	星期二

值	日
2	星期三
3	星期四
4	星期五
5	星期六
6	星期日

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例：

若您想要週和週數的 ISO 設定，確認指令碼中有下列內容：

```
Set FirstWeekDay=0;  // Monday as first week day
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

若您想要 US 設定，確認指令碼中有下列內容：

```
Set FirstWeekDay=6;  // Sunday as first week day
Set BrokenWeeks=1;
Set ReferenceDay=1;
```

範例 1 – 使用預設值 (指令碼)

載入指令碼和結果

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

在此範例中，載入指令碼使用預設 Qlik Sense 系統變數值 `FirstWeekDay=6`。此資料包含 2020 年前 14 天的資料。

載入指令碼

```
// 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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- week
- weekday

結果表格

日期	週	weekday
01/01/2021	1	星期三
01/02/2021	1	星期四
01/03/2021	1	星期五
01/04/2021	1	星期六
01/05/2021	2	星期日
01/06/2020	2	星期一
01/07/2020	2	星期二
01/08/2020	2	星期三
01/09/2020	2	星期四

日期	週	weekday
01/10/2020	2	星期五
01/11/2020	2	星期六
01/12/2020	3	星期日
01/13/2020	3	星期一
01/14/2020	3	星期二

因為正在使用預設設定, `FirstWeekDay` 系統變數設定為 6。在結果表格中, 可以看見新的一週皆從星期日開始 (一月 5 日 和 12 日)。

範例 2 – 變更 FirstWeekDay 變數 (指令碼)

載入指令碼和結果

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

在此範例中, 資料包含 2020 年前 14 天。在指令碼開頭, 我們將 `FirstWeekDay` 變數設定為 3。

載入指令碼

```
// 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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- week
- weekday

結果表格

日期	週	weekday
01/01/2021	52	星期三
01/02/2021	1	星期四
01/03/2021	1	星期五
01/04/2021	1	星期六
01/05/2021	1	星期日
01/06/2020	1	星期一
01/07/2020	1	星期二
01/08/2020	1	星期三
01/09/2020	2	星期四
01/10/2020	2	星期五
01/11/2020	2	星期六
01/12/2020	2	星期日
01/13/2020	2	星期一
01/14/2020	2	星期二

因為 `FirstWeekDay` 系統變數設定為 3，所以每週的第一天將會是星期四。在結果表格中，可以看見新的一週皆從星期四開始 (一月 2 日和 9 日)。

LongDayNames

此定義的格式可取代區域設定中的星期幾長名稱慣例。

語法：

LongDayNames

下列 `LongDayNames` 函數範例完整定義日子名稱：

```
Set LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';
```

修改變數時，需要分號 ; 才能分隔個別的值 。

`LongDayNames` 函數可組合 *Date (page 1162)* 函數使用，這會傳回 `DayNames` 作為欄位值。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 - 系統變數預設

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 具有日期的資料集將會載入到名稱為 `Transactions` 的表格中。
- `date` 欄位。
- 預設 `LongDayNames` 定義。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- dayname

建立此量值:

=sum(amount)

結果表格

日期	dayname	=sum(amount)
01/01/2022	星期六	1000
02/01/2022	星期二	2123
03/01/2022	星期二	4124
04/01/2022	星期五	2431

在載入指令碼中,若要建立稱為 dayname 的欄位,Date 函數會作為第一個引數搭配 date 欄位使用。該函數中的第二個引數格式為 WWWW。

使用此格式將值從第一個引數轉換為變數 LongDayNames 中設定的對應日子全名。在結果表格中,策劃欄位 dayname 的欄位值會顯示此。

範例 2 – 變更系統變數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

使用與第一個範例相同的資料集和情境。不過,在指令碼開頭,已修改 LongDayNames 定義以使用西班牙文一週中的日子。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- dayname

建立此量值：

```
=sum(amount)
```

結果表格

日期	dayname	=sum(amount)
01/01/2022	Sábado	1000
02/01/2022	Martes	2123
03/01/2022	Martes	4124
04/01/2022	Viernes	2431

在載入指令碼中，已修改 LongDayNames 變數以列出西班牙文一週中的日子。

然後，您可建立稱為 dayname 的欄位，這是作為第一個引數搭配 date 欄位使用的 Date 函數。

該函數中的第二個引數格式為 WWWW。使用此格式 Qlik Sense 將值從第一個引數轉換為變數 LongDayNames 中設定的對應日子全名。

在結果表格中，策劃欄位 dayname 的欄位值會顯示以西班牙文書寫的一週中日子全名。

LongMonthNames

此定義的格式可取代區域設定中的月份長名稱慣例。

語法：

LongMonthNames

修改變數時，需要使用 ; 才能分隔個別的值 。

下列 LongMonthNames 函數範例完整定義月份名稱：

```
Set
LongMonthNames='January;February;March;April;May;June;July;August;September;October;November;D
ecember';
```

LongMonthNames 函數通常用於組合下列函數：

相關函數

函數	互動
Date (page 1162)	要作為欄位值傳回 DayNames 的指令碼函數。
LongDayNames (page 216)	DayNames 的長形式值。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 - 系統變數預設

載入指令碼和結果

概述

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入到名為 Transactions 之表格中的日期資料集。
- date 欄位。
- 預設 LongMonthNames 定義。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度。

- date
- monthname

建立此量值

```
=sum(amount)
```

結果表格

日期	monthname	sum(amount)
01/01/2022	一月	1000.45
01/02/2022	一月	2123.34
01/03/2022	一月	4124.35
01/04/2022	一月	2431.36
01/05/2022	一月	4787.78
01/06/2022	一月	2431.84
01/07/2022	一月	2854.83
01/08/2022	一月	3554.28
01/09/2022	一月	3756.17
01/10/2022	一月	3454.35

使用預設 LongMonthNames 定義。在載入指令碼中,若要建立稱為 month 的欄位,Date 函數會作為第一個引數搭配 date 欄位使用。該函數中的第二個引數格式為 MMMM。

使用此格式 Qlik Sense 將值從第一個引數轉換為變數 LongMonthNames 中設定的對應月份全名。在結果表格中,策劃欄位 month 的欄位值會顯示此。

範例 2 - 變更系統變數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 載入到名為 Transactions 之表格中的日期資料集。
- date 欄位。

- 經過修改以使用西班牙文縮寫的一週中日子的 LongMonthNames 變數。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格，將 sum(amount) 新增為量值，並將這些欄位新增為維度：

- date
- monthname

建立此量值：

=sum(amount)

結果表格

日期	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

在載入指令碼中，已修改 `LongMonthNames` 變數以列出西班牙文一年中的月份。然後，若要建立稱為 `monthname` 的欄位，`Date` 函數可作為第一個引數搭配 `date` 欄位使用的。該函數中的第二個引數格式為 `MMMM`。

使用此格式 Qlik Sense 將值從第一個引數轉換為變數 `LongMonthNames` 中設定的對應月份全名。在結果表格中，策劃欄位 `monthname` 的欄位值會顯示以西班牙文書寫的月份名稱。

MoneyDecimalSep

此定義的小數點分隔符號可取代區域設定所設定的貨幣小數點符號。

> *依照預設，Qlik Sense 在表格圖表中以不同的方式顯示數字和文字。數字會靠右對齊，文字則靠左對齊。如此可以輕鬆找到文字對數字的轉換問題。此頁面此顯示 Qlik Sense 結果的任何表格將會使用此格式。*

語法：

```
MoneyDecimalSep
```

Qlik Sense 應用程式會解譯符合此貨幣值格式的文字欄位。文字欄位必須包含在 `MoneyFormat` 系統變數中定義的貨幣符號。當處理從多項不同區域設定的資料來源時，`MoneyDecimalSep` 特別實用。

下列範例顯示 `MoneyDecimalSep` 系統變數的可能使用情況：

```
Set MoneyDecimalSep='.';
```

此函數通常與下列函數搭配使用：

相關函數

函數	互動
`MoneyFormat`	在文字欄位轉譯的例子中，`MoneyFormat` 符號會作為轉譯一部分使用。對於數字格式設定，圖表物件中的 Qlik Sense 會使用 `MoneyFormat` 格式設定。
`MoneyThousandSep`	在文字欄位轉譯的例子中，亦須遵守 `MoneyThousandSep` 函數。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 - MoneyDecimalSep 點 (.) 標記法

載入指令碼和結果

概述

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 載入到名為 Transactions 之表格中的資料集。
- 提供的資料具有文字格式的貨幣欄位,還有作為小數點分隔符號使用的點「.」。每筆記錄也會以「$」符號為前置詞,而最後一筆記錄除外,其前置詞為「£」符號。

請記住,MoneyFormat 系統變數將美元「$」定義為預設貨幣。

載入指令碼

```
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'
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度:amount。

新增下列量值:

- isNum(amount)
- sum(amount)

檢閱下方結果,且內容僅展示所有美元「$」值的正確轉譯。

結果表格

金額	=isNum(amount)	=Sum(amount)
總計	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

上方結果表格顯示如何為所有以美元 ($) 為前置詞的值正確轉譯 `amount` 欄位，且以英鎊 (£) 為前置詞的 `amount` 並未轉換為貨幣值。

範例 2 - MoneyDecimalSep 逗號 (,) 標記法

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入到名為 `Transactions` 之表格中的資料集。
- 提供的資料具有文字格式的貨幣欄位，還有作為小數點分隔符號使用的逗號「,」。每筆記錄也會以「$」符號為前置詞，而最後一筆記錄除外，其前置詞為「£」符號。

請記住，`MoneyFormat` 系統變數將美元「$」定義為預設貨幣。

載入指令碼

```
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'
];
```

結果

結果的段落文字。

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度:`amount`。

新增下列量值:

- `isNum(amount)`
- `sum(amount)`

檢閱下方結果,且內容展示所有值的正確轉譯,除了小數點分隔符號使用點「.」標記法的金額以外。在該情況下,應改用逗號。

結果表格

金額	=isNum(amount)	=Sum(amount)
總計	0	$3905.98
$345.66	0	$0.00
$3,436	-1	$3.44
$3,55	-1	$3.55
$7,54	-1	$7.54
$14,41	-1	$14.41
$24,37	-1	$24.37
$243,63	-1	$243.63
$249,36	-1	$249.36
$545,36	-1	$545.36
$2.814,32	-1	$2814.32

MoneyFormat

此系統變數定義 Qlik 用於將文字自動翻譯為數字的格式模式，且該數字以貨幣符號為前置詞。它也會定義圖表物件如何顯示數字格式設定屬性設為「貨幣」的量值。

在 `MoneyFormat` 系統變數中定義為格式模式一部分的符號會取代您區域設定所設定的貨幣符號。

> *依照預設，Qlik Sense 在表格圖表中以不同的方式顯示數字和文字。數字會靠右對齊，文字則靠左對齊。如此可以輕鬆找到文字對數字的轉換問題。此頁面此顯示 Qlik Sense 結果的任何表格將會使用此格式。*

語法：

MoneyFormat

```
Set MoneyFormat='$ #,##0.00; ($ #,##0.00)';
```

當數字欄位的 `Number Formatting` 屬性設為 `Money` 時，圖表物件即會顯示此格式設定。而且，當數字文字欄位經由 Qlik Sense 轉譯時，如果文字欄位的貨幣符號符合 `MoneyFormat` 變數中定義的符號，Qlik Sense 即會將此欄位轉譯為貨幣值。

此函數通常與下列函數搭配使用：

相關函數

函數	互動
MoneyDecimalSep (page 223)	對於數字格式設定，物件欄位格式設定中會使用 `MoneyDecimalSep`。
MoneyThousandSep (page 231)	對於數字格式設定，物件欄位格式設定中會使用 `MoneyThousandSep`。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 - MoneyFormat

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含載入到名為 Transactions 之表格的資料集。已使用預設的 MoneyFormat 變數定義。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- amount

新增此量值:

```
=Sum(amount)
```

在「**數字格式設定**」下, 選取「**貨幣**」將 Sum(amount) 設為貨幣值。

結果表格

日期	金額	=Sum(amount)
總計		$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

使用預設 `MoneyFormat` 定義。這看起來如下所示：`$###0.00;-$###0.00`。在結果表格中，`amount` 欄位的格式顯示貨幣符號和小數點，並已包括小數位數。

範例 2 - 具有千位分隔符號和混合輸入格式的 MoneyFormat

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 混合輸入格式資料集，其載入到名為 `Transactions` 的表格中並散佈著千位分隔符號和小數點分隔符號。
- `MoneyFormat` 定義的修改經修改後包括逗號並作為千位分隔符號。
- 其中一個資料列在不正確的位置錯誤地用千位分隔符號分隔。請注意此金額如何保留為文字且未轉譯為數字。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- amount

新增此量值：

```
=Sum(amount)
```

在「**數字格式設定**」下，選取「**貨幣**」將 Sum(amount) 設為貨幣值。

結果表格

日期	金額	=Sum(amount)
總計		$119,548,811,911.90
01/01/2022	$10,000,000,441.45	$10,000,000,441.45
01/02/2022	$212,3749,24,32.23	$0.00
01/03/2022	$249475336.45	$249,475,336.45
01/04/2022	$24	$24.00
01/05/2022	$7873578754	$7,873,578,754.00
01/06/2022	$24313884663	$24,313,884,663.00
01/07/2022	$545883436	$545,883,436.00
01/08/2022	$35545828255	$35,545,828,255.00
01/09/2022	$37565817436	$37,565,817,436.00
01/10/2022	$3454343566	$3,454,343,566.00

在指令碼開始時，MoneyFormat 系統變數經修改後包括逗點並作為千位分隔符號。在 Qlik Sense 表格中，可查看格式設定以包括此分隔符號。此外，錯誤分隔符號的列未正確轉譯並保留為文字。這就是為什麼其對金額加總沒有貢獻。

MoneyThousandSep

此定義的千位分隔符號可取代區域設定所設定的的貨幣位數分組符號。

> *依照預設，Qlik Sense 在表格圖表中以不同的方式顯示數字和文字。數字會靠右對齊，文字則靠左對齊。如此可以輕鬆找到文字對數字的轉換問題。此頁面此顯示 Qlik Sense 結果的任何表格將會使用此格式。*

語法：

```
MoneyThousandSep
```

Qlik Sense 應用程式會解譯符合此貨幣值格式的文字欄位。文字欄位必須包含在 `MoneyFormat` 系統變數中定義的貨幣符號。當處理從多項不同區域設定的資料來源時，`MoneyThousandSep` 特別實用。

下列範例顯示 `MoneyThousandSep` 系統變數的可能使用情況：

```
Set MoneyDecimalSep=',';
```

此函數通常與下列函數搭配使用：

相關函數

函數	互動
`MoneyFormat`	在文字欄位轉譯的例子中，`MoneyFormat` 符號會作為轉譯一部分使用。對於數字格式設定，圖表物件中的 Qlik Sense 會使用 `MoneyFormat` 格式設定。
`MoneyDecimalSep`	在文字欄位轉譯的例子中，亦須遵守 `MoneyDecimalSep` 函數。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 - MoneyThousandSep 逗號 (,) 標記法

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入到名為 `Transactions` 之表格中的資料集。
- 提供的資料具有文字格式的貨幣欄位，還有作為千位分隔符號使用的逗號。每筆記錄也會以「$」符號為前置詞。

請記住，`MoneyFormat` 系統變數將美元「$」定義為預設貨幣。

載入指令碼

```
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'
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`amount`。

新增下列量值：

- `isNum(amount)`
- `sum(amount)`

檢閱下方結果。該表格將逗號「,」標記法作為千位分隔符號使用，展示所有值的正確轉譯。

已為所有值正確轉譯 `amount` 欄位，除了將點「.」作為千位分隔符號使用的一個值以外。

結果表格

金額	=isNum(amount)	=Sum(amount)
總計	0	$161645330590.00

金額	=isNum(amount)	=Sum(amount)
$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

範例 2 - MoneyThousandSep 點 (.) 標記法

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入到名為 `Transactions` 之表格中的資料集。
- 提供的資料具有文字格式的貨幣欄位，還有作為千位分隔符號使用的點「.」。每筆記錄也會以「$」符號為前置詞。

請記住，`MoneyFormat` 系統變數將美元「$」定義為預設貨幣。

載入指令碼

```
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'
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：amount。

新增下列量值：

- isNum(amount)
- sum(amount)

檢閱下方結果，且內容將點「.」作為千位分隔符號使用，展示所有值的正確轉譯。

已為所有值正確轉譯 amount 欄位，除了將逗號「,」作為千位分隔符號使用的一個值以外。

結果表格

金額	=isNum(amount)	=Sum(amount)
總計	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

此定義的格式可取代區域設定的月份名稱慣例。

語法：

```
MonthNames
```

修改變數時，需要使用 ; 才能分隔個別的值 。

函數範例

範例	結果
`Set MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';`	這種 `MonthNames` 函數的使用方式以英文縮寫形式定義月份名稱。
`Set MonthNames='Enero;Feb;Marzo;Abr;Mayo;Jun;Jul;Agosto;Set;Oct;Nov;Dic';`	這種 `MonthNames` 函數的使用方式以西班牙文縮寫形式定義月份名稱。

`MonthNames` 函數可用於組合下列函數:

相關函數

函數	互動
month (page 857)	要作為欄位值傳回 `MonthNames` 中定義的值的指令碼函數
Date (page 1162)	要根據提供的格式引數作為欄位值傳回 `MonthNames` 中定義的值的指令碼函數
LongMonthNames (page 219)	`MonthNames` 的長形式值

區域設定

除非另有說明,否則此主題中的範例皆使用下列日期格式:MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素,您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式,以滿足您的需求。或者,您可以在載入指令碼中變更格式,以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典,資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 系統變數預設

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 載入到名為 `Transactions` 之表格中的日期資料集。
- `date` 欄位。
- 預設 `MonthNames` 定義。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- `date`
- `monthname`

建立此量值:

`=sum(amount)`

結果表格

日期	monthname	sum(amount)
01/01/2022	一月	1000.45
01/02/2022	一月	2123.34
01/03/2022	一月	4124.35
01/04/2022	一月	2431.36
01/05/2022	一月	4787.78

日期	monthname	sum(amount)
01/06/2022	一月	2431.84
01/07/2022	一月	2854.83
01/08/2022	一月	3554.28
01/09/2022	一月	3756.17
01/10/2022	一月	3454.35

使用預設 `MonthNames` 定義。在載入指令碼中， `Month` 函數作為提供的引數搭配 `date` 欄位使用。

在結果表格中，此 `Month` 函數的輸出以 `MonthNames` 定義的格式顯示一年中的月份。

範例 2 - 變更系統變數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入到名為 `Transactions` 之表格中的日期資料集。
- `date` 欄位。
- 經過修改以使用西班牙文縮寫月份的 `MonthNames` 變數。

載入指令碼

```
                            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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- monthname

建立此量值:

=sum(amount)

結果表格

日期	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

在載入指令碼中，首先已修改 MonthNames 變數以列出西班牙文縮寫的一年中月份。Month 函數作為提供的引數搭配 date 欄位使用。

在結果表格中，此 Month 函數的輸出以 MonthNames 定義的格式顯示一年中的月份。

重要的是，應記住若 MonthNames 變數的語言按照此範例修改，則 LongMonthNames 變數仍會包含以英文表示的一年中的月份。若兩種變數都用於應用程式中，也必須修改 LongMonthNames 變數。

範例 3 – 日期函數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 載入到名為 Transactions 之表格中的日期資料集。
- date 欄位。
- 預設 MonthNames 定義。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- monthname

建立此量值:

=sum(amount)

結果表格

日期	monthname	sum(amount)
01/01/2022	一月	1000.45
01/02/2022	一月	2123.34
01/03/2022	一月	4124.35
01/04/2022	一月	2431.36
01/05/2022	一月	4787.78
01/06/2022	一月	2431.84
01/07/2022	一月	2854.83
01/08/2022	一月	3554.28
01/09/2022	一月	3756.17
01/10/2022	一月	3454.35

使用預設 `MonthNames` 定義。在載入指令碼中，`Date` 函數作為第一個引數搭配 `date` 欄位使用。第二個引數是 `MMM`。

使用此格式 Qlik Sense 將值從第一個引數轉換為變數 `MonthNames` 中設定的對應月份名稱。在結果表格中，策劃欄位 `month` 的欄位值會顯示此。

NumericalAbbreviation

數字縮寫可設定要在數字的比例前置詞使用哪個縮寫，例如 M 用於 mega (兆) 或 million (百萬) (10^6)，而 µ 用於 micro (百萬分之一) (10^{-6}).

語法：

`NumericalAbbreviation`

您將 `NumericalAbbreviation` 變數設定為包含縮寫定義配對清單的字串，並以分號分隔。每個縮寫定義配對應包含位數 (以十進位為基底的指數) 以及以冒號分隔的縮寫，例如 `6:M` 代表百萬。

預設設定為 `'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'`。

範例：

此設定會將一千的前置詞變更為 t，並將十億的前置詞變更為 B。對於預期會使用 t$、M$ 和 B$ 等縮寫的財務應用程式，這十分實用。

```
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

該設定定義要將 1 月中的哪一天設為參考日以定義第 1 週。換言之，此設定規定第 1 週有多少天必須是落在 1 月內的日期。

語法：

`ReferenceDay`

`ReferenceDay` 設定一年的第一週包含多少天。`ReferenceDay` 可以設定為 1 和 7 之間的任何值。任何落在 1-7 範圍之外的值會解譯為一週的中點 (4)，這對等於設定為 4 的 `ReferenceDay`。

若您沒有為 `ReferenceDay` 設定選取值，則預設值將顯示 `ReferenceDay=0`，這將解譯為一週的中點 (4)，如以下 `ReferenceDay` 值表格中所見。

`ReferenceDay` 函數通常用於組合下列函數：

相關函數

變數	互動
BrokenWeeks (page 199)	若 Qlik Sense 應用程式使用未中斷的週，將會強制執行 `ReferenceDay` 變數。不過，若使用中斷的週，則第 1 週將於 1 月 1 日開始並在接合 `FirstWeekDay` 變數設定時終止，並忽略 `ReferenceDay` 旗標。
FirstWeekDay (page 212)	定義要用作一週第一天的日子的整數。

Qlik Sense 允許為 `ReferenceDay` 設定下列值:

ReferenceDay 值

值	參考日
0 (預設)	1 月 4 日
1	1 月 1 日
2	January 2
3	1 月 3 日
4	1 月 4 日
5	1 月 5 日
6	1 月 6 日
7	1 月 7 日

在下列範例中, `ReferenceDay = 3` 將 1 月 3 日定義為參考日:

```
SET ReferenceDay=3; //(set January 3 as the reference day)
```

區域設定

除非另有說明, 否則此主題中的範例皆使用下列日期格式:MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素, 您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式, 以滿足您的需求。或者, 您可以在載入指令碼中變更格式, 以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典, 資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例:

若您想要週和週數的 ISO 設定, 確認指令碼中有下列內容:

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;    // Jan 4th is always in week 1
```

若您想要 US 設定, 確認指令碼中有下列內容:

```
Set FirstWeekDay=6;
Set BrokenWeeks=1;
Set ReferenceDay=1;     // Jan 1st is always in week 1
```

範例 1 - 使用預設值的載入指令碼; ReferenceDay=0

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 設定為 0 的 ReferenceDay 變數。
- 設定為 0 的 BrokenWeeks 變數, 這強制應用程式使用未中斷的週。
- 從 2019 年結束至 2020 年開始的日期資料集。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- week
- weekday

結果表格

日期	週	weekday
12/27/2019	52	星期五
12/28/2019	52	星期六
12/29/2019	1	星期日
12/30/2019	1	星期一
12/31/2019	1	星期二
01/01/2020	1	星期三
01/02/2020	1	星期四
01/03/2020	1	星期五
01/04/2020	1	星期六
01/05/2020	2	星期日
01/06/2020	2	星期一
01/07/2020	2	星期二
01/08/2020	2	星期三
01/09/2020	2	星期四
01/10/2020	2	星期五
01/11/2020	2	星期六

第 52 週結束於 12 月 28 日星期六。因為 `ReferenceDay` 要求在第 1 週中納入 1 月 4 日，所以第 1 週於 12 月 29 日開始，並於 1 月 4 日星期六結束。

範例 - 設定為 5 的 ReferenceDay 變數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 設定為 5 的 `ReferenceDay` 變數。
- 設定為 0 的 `BrokenWeeks` 變數，這強制應用程式使用未中斷的週。
- 從 2019 年結束至 2020 年開始的日期資料集。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- week
- weekday

結果表格

日期	週	weekday
12/27/2019	52	星期五
12/28/2019	52	星期六
12/29/2019	53	星期日
12/30/2019	53	星期一
12/31/2019	53	星期二
01/01/2020	53	星期三
01/02/2020	53	星期四
01/03/2020	53	星期五
01/04/2020	53	星期六
01/05/2020	1	星期日

日期	週	weekday
01/06/2020	1	星期一
01/07/2020	1	星期二
01/08/2020	1	星期三
01/09/2020	1	星期四
01/10/2020	1	星期五
01/11/2020	1	星期六

第 52 週結束於 12 月 28 日星期六。`BrokenWeeks` 變數強制應用程式使用未中斷的週。5 的參考日值要求在第 1 週中納入 1 月 5 日。

不過，此為前一年第 52 週結束的八天後。因此，第 53 週於 12 月 29 日開始，並於 1 月 4 日結束。第 1 週於 1 月 5 日星期日開始。

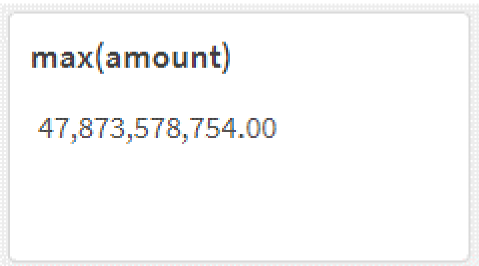
ThousandSep

此定義的千位分隔符號可取代作業系統的位數分組符號 (區域設定)。

語法：

```
ThousandSep
```

使用 ThousandSep 變數的 Qlik Sense 物件(具有千位分隔符號)



Qlik Sense 應用程式會解譯符合此數字格式設定的文字欄位。當數字欄位的**數字格式設定**屬性設為**數字**時，圖表物件即會顯示此格式設定。

當處理從多項區域設定收到的資料來源時，`ThousandSep` 很實用。

> *如果 ThousandSep 變數在應用程式中已建立物件並進行格式設定後修改，則使用者必須取消選取並重新選取**數字格式設定**屬性**數字**，藉此對每個相關欄位重新進行格式設定。*

下列範例顯示 `ThousandSep` 系統變數的可能使用情況：

```
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)
```

這些主題可協助您使用此函數：

相關主題

主題	描述
DecimalSep (page 210)	在此文字欄位轉譯的例子中,亦須遵守此函數提供的小數點分隔符號設定。對於數字格式設定,Qlik Sense 會在需要時使用 **DecimalSep**。

區域設定

除非另有說明,否則此主題中的範例皆使用下列日期格式:MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素,您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式,以滿足您的需求。或者,您可以在載入指令碼中變更格式,以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典,資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 - 預設系統變數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 載入到名為 `Transactions` 之表格中的資料集。
- 使用預設的 `ThousandSep` 變數定義。

載入指令碼

```
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
```

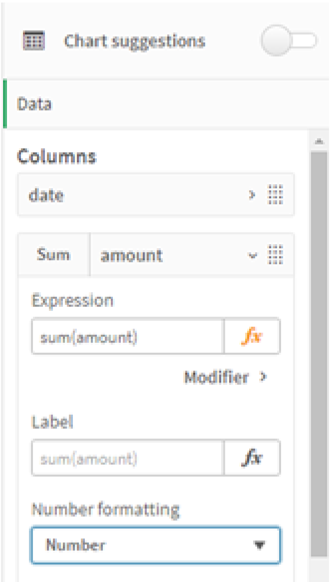
```
01/10/2022,10,3454343566
];
```

結果

請執行下列動作：

1. 載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`date`。
2. 新增下列量值：
   `=sum(amount)`
3. 在屬性面板中，選取**資料**下方的量值。
4. 在「**數字格式設定**」下方，選取「**數字**」。

調整圖表量值的數字格式設定



結果表格

在結果表格中，金額欄位的格式顯示千位分組之間的逗號。	=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

在結果表格中，金額欄位的格式顯示千位分組之間的逗號。	=sum(amount)
01/09/2022	37,565,817,436.00
01/10/2022	3,454,343,566.00

在此範例中，已使用設為逗號格式(「,」)的預設 ThousandSep 定義。在結果表格中，金額欄位的格式顯示千位分組之間的逗號。

範例 2 - 變更系統變數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 第一個範例的相容資料集，其載入到名為 Transactions 的表格。
- 指令碼開始時的 ThousandSep 定義修改顯示「*」字元作為千位分隔符號。這是極限的範例，並僅用於展示變數的功能。

此範例中使用的修改為極限且不常使用，但在此顯示為展示變數的功能。

載入指令碼

```
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
];
```

結果

請執行下列動作:

1. 載入資料並開啟工作表。建立新的表格並將此欄位新增為維度:`date`。
2. 新增下列量值:
   `=sum(amount)`
3. 在屬性面板中,選取「**資料**」下方的量值。
4. 在「**數字格式設定**」下方,選取「**自訂**」。

結果表格

在結果表格中,金額欄位的格式顯示千位分組之間的逗號。	=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

在指令碼開始時,`ThousandSep` 系統變數會修改為「`*`」。在結果表格中,可看到金額欄位的格式顯示千位分組之間的「`*`」。

範例 3 - 文字轉譯

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 載入到名為 `Transactions` 之表格中的資料集。
- 資料具有文字格式的數字欄位,還有作為千位分隔符號使用的逗號。
- 使用預設的 `ThousandSep` 系統變數。

載入指令碼

```
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'
];
```

結果

請執行下列動作:

1. 載入資料並開啟工作表。建立新的表格並將此欄位新增為維度:`date`。
2. 新增下列量值:
   `=sum(amount)`
3. 在屬性面板中,選取「**資料**」下方的量值。
4. 在「**數字格式設定**」下方,選取「**數字**」。
5. 新增下列量值以評估金額欄位是否為數字值:
   `=isnum(amount)`

結果表格

在結果表格中,金額欄位的格式顯示千位分組之間的逗號。	=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

在結果表格中，金額欄位的格式顯示千位分組之間的逗號。	=sum(amount)	=isnum (amount)
01/08/2022	545,828,255.00	-1
01/09/2022	37,565,817,436.00	-1
01/10/2022	3*454*343*566.00	-1

載入資料後，我們可以看到 Qlik Sense 已將金額欄位轉譯為數字值，因為該資料符合 `ThousandSep` 變數。這可由 `isnum()` 函數展示，其評估每個項目到 -1 或 `TRUE`。

在 Qlik Sense 中，布林值 true 值以 -1 代表，而 false 值以 0 代表。

TimeFormat

此定義的格式可取代作業系統的時間格式 (區域設定)。

語法：

```
TimeFormat
```

範例：

```
Set TimeFormat='hh:mm:ss';
```

TimestampFormat

此定義的格式可取代作業系統的日期和時間格式 (區域設定)。

語法：

```
TimestampFormat
```

範例：

下列範例使用 *1983-12-14T13:15:30Z* 作為時間戳記資料，以顯示不同的 **SET TimestampFormat** 陳述式的結果。使用的日期格式是 **YYYYMMDD**，而時間格式是 **h:mm:ss TT**。日期格式在資料載入指令碼頂部的 **SET DateFormat** 陳述式中指定，而時間格式在資料載入指令碼頂部的 **SET TimeFormat** 陳述式中指定。

結果

範例	結果
`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

範例:載入指令碼

範例:載入指令碼

在第一個載入指令碼中,會使用 *SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT'*。在第二個載入指令碼中,時間戳記格式會變更為 *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'*。不同的結果顯示 **SET TimeFormat** 陳述式搭配不同時間資料格式的運作方式。

以下表格顯示用於所遵守的載入指令碼中的資料集。表格的第二欄顯示資料集中每個時間戳記的格式。前五個時間戳記遵守 ISO 8601 規則,但第六個沒有遵守。

資料集

表格顯示使用的時間資料以及資料集中每個時間戳記的格式。

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

在**資料載入編輯器**中,建立新的區段,然後新增範例指令碼並執行。然後,至少將結果資料行中列出的欄位新增至您應用程式中的工作表以查看結果。

載入指令碼

```
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
];
```

結果

Qlik Sense 表格顯示載入指令碼中使用的 TimestampFormat 解譯變數結果。資料集中的最後一個時間戳記不會傳回正確的日期。

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	-

下一個載入指令碼使用相同的資料集。不過，這使用 *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'* 以符合第六個時間戳記的非 ISO 8601 格式。

在**資料載入編輯器**中，以下列項目之一取代先前的範例指令碼並執行。然後，至少將結果資料行中列出的欄位新增至您應用程式中的工作表以查看結果。

載入指令碼

```
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
];
```

結果

Qlik Sense 表格顯示載入指令碼中使用的 TimestampFormat 解譯變數結果。

transaction_id	transaction_timestamp	LogTimeStamp
3750	2018-08-30	2018-08-30 00:00:00
3751	20180830T193614.857	2018-08-30 19:36:14
3752	20180830T193614.857+0200	2018-08-30 17:36:14
3753	2018-09-16T12:30-02:00	2018-09-16 14:30:00
3754	2018-09-16T13:15:30Z	2018-09-16 13:15:30
3755	9/30/18 19:36:14	2018-09-16 19:36:14

4.9 Direct Discovery 變數

Direct Discovery 系統變數

DirectCacheSeconds

您可針對視覺化設定 Direct Discovery 查詢結果的快取限制。一旦達到此時間限制，Qlik Sense 會在執行新的 Direct Discovery 查詢時清除快取。Qlik Sense 會針對選項查詢來源資料，然後為指定的時間限制再次建立快取。每個選項組合的結果都會個別快取。也就是說，各個選項會個別重新整理快取；所以第一個選項只針對所選的欄位重新整理快取，而第二個選項則針對其相關欄位重新整理快取。如果第二個選項包含第一個選項已重新整理的欄位，則在快取限制尚未達到前，這些欄位不會在快取中再次更新。

Direct Discovery 快取不會套用至**表格**視覺化。表格選項每次都會查詢資料來源。

限制值必須以秒為單位進行設定。預設的快取限制為 1800 秒 (30 分鐘)。

用於 **DirectCacheSeconds** 的值是在執行 **DIRECT QUERY** 陳述式當下所設定的值。該值在執行階段無法變更。

範例：

```
SET DirectCacheSeconds=1800;
```

DirectConnectionMax

透過連接共用功能，即可對資料庫執行非同步的平行呼叫。設定共用功能的載入指令碼語法如下所示：

```
SET DirectConnectionMax=10;
```

該數值設定會指定 Direct Discovery 程式碼在更新工作表時，可使用之資料庫連接的數目上限。預設設定為 1。

應謹慎使用此變數。已知將其設為大於 1 會在連線至 Microsoft SQL Server 時引起問題。

DirectUnicodeStrings

透過使用針對延伸字元字串常值的 SQL 標準格式 (N'<延伸字串>'), Direct Discovery 就可支援對延伸 Unicode 資料的選取;這是部分資料庫 (尤其是 SQL Server) 的要求。Direct Discovery 可透過指令碼變數 **DirectUnicodeStrings** 來使用這個語法。

設定此變數為 'True', 即可在字串常值前使用 ANSI 標準寬字元標記 “N”。並非所有資料庫都支援此標準。預設設定為 'False'。

DirectDistinctSupport

在 Qlik Sense 物件中選取 **DIMENSION** 欄位值時, 即會針對來源資料庫產生查詢。當查詢需要分組時, Direct Discovery 會使用 **DISTINCT** 關鍵字, 以僅選取唯一值。不過, 部分資料庫必須使用 **GROUP BY** 關鍵字。將 **DirectDistinctSupport** 設為 'false', 即可在用於唯一值的查詢中產生 **GROUP BY**, 而非 **DISTINCT**。

```
SET DirectDistinctSupport='false';
```

如果將 DirectDistinctSupport 設為 True, 則會使用 **DISTINCT**。若未設定, 預設行為是使用 **DISTINCT**。

DirectEnableSubquery

在高基數的多表格案例中, 可能會在 SQL 查詢中產生子查詢, 而非產生較大的 IN 子句。這可透過將 **DirectEnableSubquery** 設定為 'true' 啟用。預設值為 'false'。

*當啟用 **DirectEnableSubquery** 時, 您不能載入不處於 Direct Discovery 模式的表格。*

```
SET DirectEnableSubquery='true';
```

Teradata 查詢級區變數

Teradata 查詢級區是能讓企業應用程式與基礎 Teradata 資料庫共同作業的功能, 可提供更有效的計量、優先順序及工作負載管理。透過查詢級區, 您可在查詢周圍包覆中繼資料, 例如使用者認證。

有兩個變數可供使用, 這兩個變數都是經過評估並傳送至資料庫的字串。

SQLSessionPrefix

此字串會在建立資料庫的連線時傳送。

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR SESSION;';
```

舉例來說, 如果 **OSuser()** 傳回 *WA\sbt*, 這將會被評估為 `SET QUERY_BAND = 'Who=WA\sbt;' FOR SESSION;`, 並於建立連線時傳送至資料庫。

SQLQueryPrefix

此字串會針對每一個查詢傳送。

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR TRANSACTION;';
```

Direct Discovery 字元變數

DirectFieldColumnDelimiter

您可設定 **Direct Query** 陳述式中作為欄位分隔符號的字元，以用於需要逗號之外的字元作為欄位分隔符號的資料庫。指定的字元必須在 **SET** 陳述式中以單引號括住。

```
SET DirectFieldColumnDelimiter= '|'
```

DirectStringQuoteChar

您可指定在產生的查詢中要用於引用字串的字元。預設為單引號。指定的字元必須在 **SET** 陳述式中以單引號括住。

```
SET DirectStringQuoteChar= '"';
```

DirectIdentifierQuoteStyle

您可以指定在產生的查詢中要使用的識別碼非 ANSI 引號。目前唯一可用的非 ANSI 引號在 GoogleBQ 中提供。預設為 ANSI。大寫、小寫及大小寫混合皆可使用 (ANSI, ansi, Ansi)。

```
SET DirectIdentifierQuoteStyle="GoogleBQ";
```

例如，以下 **SELECT** 陳述式會使用 ANSI 引號：

```
SELECT [Quarter] FROM [qvTest].[sales] GROUP BY [Quarter]
```

若 **DirectIdentifierQuoteStyle** 設為 "GoogleBQ"，則 **SELECT** 陳述式會使用如下的引號：

```
SELECT [Quarter] FROM [qvTest.sales] GROUP BY [Quarter]
```

DirectIdentifierQuoteChar

您可指定在產生的查詢中控制識別碼引號的字元。這可設為一個字元 (如雙引號) 或兩個字元 (如一對方括弧)。預設為雙引號。

```
SET DirectIdentifierQuoteChar='[]';
SET DirectIdentifierQuoteChar='``';
SET DirectIdentifierQuoteChar=' ';
SET DirectIdentifierQuoteChar='"';
```

DirectTableBoxListThreshold

在**表格**視覺化中使用 Direct Discovery 欄位時，會設定臨界值以限制所顯示的列數。預設臨界值為 1000 個記錄。預設臨界值設定可透過在載入指令碼中設定 **DirectTableBoxListThreshold** 進行變更。例如：

```
SET DirectTableBoxListThreshold=5000;
```

此臨界值設定僅會套用至包含 Direct Discovery 欄位的**表格**視覺化。僅包含記憶體內欄位的**表格**視覺化不會受限於 **DirectTableBoxListThreshold** 設定。

直到選項比臨界值限制的記錄要少的時候，**表格**視覺化才會顯示欄位。

Direct Discovery 數字解譯變數

DirectMoneyDecimalSep

此定義的小數點分隔符號可取代將使用 Direct Discovery 載入資料而產生的 SQL 陳述式中所用的貨幣小數符號。此字元必須符合 **DirectMoneyFormat** 中使用的字元。

預設值為 `'.'`

範例：

```
Set DirectMoneyDecimalSep='.';
```

DirectMoneyFormat

此定義的符號可取代將使用 Direct Discovery 載入資料而產生的 SQL 陳述式中所用的貨幣格式。不應該包含千位分隔符號的貨幣符號。

預設值為 '#.0000'

範例：

```
Set DirectMoneyFormat='#.0000';
```

DirectTimeFormat

此定義的時間格式可取代將使用 Direct Discovery 載入資料而產生的 SQL 陳述式中所用的時間格式。

範例：

```
Set DirectTimeFormat='hh:mm:ss';
```

DirectDateFormat

此定義的日期格式可取代將使用 Direct Discovery 載入資料而產生的 SQL 陳述式中所用的日期格式。

範例：

```
Set DirectDateFormat='MM/DD/YYYY';
```

DirectTimeStampFormat

此定義的格式可取代將使用 Direct Discovery 載入資料而產生的 SQL 陳述式中所用的日期和時間格式。

範例：

```
Set DirectTimestampFormat='M/D/YY hh:mm:ss[.fff]';
```

4.10 錯誤變數

所有錯誤變數的值將會在指令碼執行後存在。第一個變數 ErrorMode 是從使用者輸入，並且最後三個是從具有指令碼中錯誤資訊的 Qlik Sense 輸出。

錯誤變數概述

會在概述後進一步描述每個變數。您還可以在語法中按一下變數名稱，以立即存取該特定變數的詳細資料。

請參閱 Qlik Sense 線上說明，以獲得有關變數的進一步詳細資料。

ErrorMode

此錯誤變數會決定指令碼執行期間發生錯誤時，Qlik Sense 將採取什麼動作。

```
ErrorMode
```

ScriptError

此錯誤變數會傳回最後執行指令碼陳述式的錯誤碼。

```
ScriptError
```

ScriptErrorCount

此錯誤變數會傳回目前指令碼執行期間造成錯誤的陳述式總數。此變數一律在指令碼開始執行時重設為 0。

```
ScriptErrorCount
```

ScriptErrorList

此錯誤變數將包含一份串連清單，其中列出上次指令碼執行期間發生的所有指令碼錯誤。每個錯誤都會以換行字元分隔。

```
ScriptErrorList
```

ErrorMode

此錯誤變數會決定指令碼執行期間發生錯誤時，Qlik Sense 將採取什麼動作。

語法：

```
ErrorMode
```

引數：

引數

引數	描述
ErrorMode=1	預設設定。指令碼執行將終止，並提示使用者採取動作 (非批次模式)。
ErrorMode =0	Qlik Sense 將直接忽略失敗，並且在下一個指令碼陳述式繼續執行指令碼。
ErrorMode =2	Qlik Sense 將在失敗時立即觸發「指令碼執行失敗...」錯誤訊息，而不事先提示使用者採取動作。

範例：

```
set ErrorMode=0;
```

ScriptError

此錯誤變數會傳回最後執行指令碼陳述式的錯誤碼。

語法：

```
ScriptError
```

每次成功執行指令碼陳述式之後，此變數將重設為 0。如果發生錯誤，它將設定為內部 Qlik Sense 錯誤碼。錯誤碼是有數值和文字部分的雙值。會出現下列錯誤碼：

指令碼錯誤碼

錯誤碼	描述
0	無錯誤。雙值文字是空的。
1	一般錯誤。
2	語法錯誤。
3	一般 ODBC 錯誤。
4	一般 OLE DB 錯誤。
5	一般自訂資料庫錯誤。
6	一般 XML 錯誤。
7	一般 HTML 錯誤。
8	找不到檔案。
9	找不到資料庫。
10	找不到表格。
11	找不到欄位。
12	檔案格式錯誤。
16	語意錯誤。

範例:

```
set ErrorMode=0;

LOAD * from abc.qvf;

if ScriptError=8 then

exit script;

//no file;

end if
```

ScriptErrorCount

此錯誤變數會傳回目前指令碼執行期間造成錯誤的陳述式總數。此變數一律在指令碼開始執行時重設為 0。

語法:

```
ScriptErrorCount
```

ScriptErrorList

此錯誤變數將包含一份串連清單，其中列出上次指令碼執行期間發生的所有指令碼錯誤。每個錯誤都會以換行字元分隔。

語法：

```
ScriptErrorList
```

5 指令碼運算式

LOAD 陳述式和 **SELECT** 陳述式中都可使用運算式。此處所述的語法和函數適用於 **LOAD** 陳述式，但不適用於 **SELECT** 陳述式，因為後者是由 ODBC 驅動程式所解譯，而非由 Qlik Sense 所解譯。不過，大部分的 ODBC 驅動程式通常都可解譯以下所述的一些函數。

運算式包含函數、欄位和運算子，結合在一個語法中。

Qlik Sense 指令碼中所有的運算式皆會傳回一個數字及/或字串，依適合的情況而定。邏輯函數和運算子對於 False 會傳回 0，對於 True 會傳回 -1。數字和字串之間的轉換是隱含的。邏輯運算子和函數會將 0 解譯為 False，而將其他都解譯為 True。

運算式的一般語法為：

一般語法

運算式	**欄位**	**運算子**
expression ::= (constant	constant	|
expression ::= (constant	fieldref	|
expression ::= (constant	operator1 expression	|
expression ::= (constant	expression operator2 expression	|
expression ::= (constant	function	|
expression ::= (constant	(expression)	)

其中：

- **constant** 為以一般單引號括住的字串 (文字、日期或時間)，或為數字。常數的寫法不會包含千位分隔符號，但會加上小數點作為小數點分隔符號。
- **fieldref** 為載入表格的欄位名稱。
- **operator1** 為一元運算子 (會影響右側的一個運算式)。
- **operator2** 為二元運算子 (會影響兩側各一個運算式)。
- **function ::= functionname(parameters)**

- **parameters ::= expression { , expression }**

參數的數目和類型非為任意的，會視所用的函數而定。

因此，運算式和函數可任意構成巢狀，只要運算式傳回可解譯的值，Qlik Sense 就不會產生任何錯誤訊息。

6 圖表運算式

圖表 (視覺化) 運算式是函數、欄位與數學運算子 (+ * / =) 及其他量值的組合。運算式可用來處理應用程式中的資料, 以便產生可在視覺化中查看的結果。運算式不限於在量值中使用。您可使用標題、副標題、註腳甚至是維度的運算式, 建置更加動態且功能更強大的視覺化。

這就表示, 舉例來說, 視覺化標題並非靜態文字, 而是可以從運算式得出, 且運算式的結果會根據選取的選項而改變。

如需有關指令碼函數及圖表函數的詳細參考, 請參閱 指令碼語法和圖表函數。

6.1 定義彙總範圍

通常有兩個因素會共同決定要使用哪些記錄來定義中運算式的彙總值。使用視覺化時, 這些因素為:

- 維度值 (圖表運算式中彙總的維度值)
- 選項

這些因素共同定義了彙總範圍。您可能會遇到需在計算中忽略選項、維度或同時忽略兩者的情形。在圖表函數中, 使用 TOTAL 限定詞、集合分析或結合這兩者就可以達成目的。

彙總:方法和描述

方法	描述
TOTAL 限定詞	在彙總函數中使用 total 限定詞即可忽略維度值。 彙總會針對所有可能欄位值執行。 **TOTAL** 限定詞後面可以加上以角括弧括住的一或多個欄位名稱清單。這些欄位名稱應該是圖表維度變數的子集。在這種情況下, 將忽略所列圖表維度變數以外的所有圖表維度變數進行計算, 也就是說, 將對於所列維度欄位中欄位值的各個組合傳回一個值。另外, 清單可以包含目前在圖表中並不是維度的欄位。對於維度欄位不固定的群組維度而言, 這會相當實用。列出群組的所有變數會使得函數在向下探查層級變更時產生作用。
集合分析	在彙總中使用集合分析即可覆寫選項。彙總會針對所有跨維度的值執行。
TOTAL 限定詞與集合分析	在彙總中使用 **TOTAL** 限定詞和集合分析即可覆寫選項並忽略維度。

方法	描述

ALL 限定詞	在彙總中使用 **ALL** 限定詞即可忽略選項及維度。使用 {1} 集合分析陳述式和 **TOTAL** 限定詞也可實現相同目的：
	`=sum(All Sales)`
	`=sum({1} Total Sales)`

範例：TOTAL 限定詞

以下範例說明如何使用 TOTAL 來計算相對份額。假設已選取 Q2，使用 TOTAL 可忽略維度並計算所有值的加總。

範例：Total 限定詞

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%

若要將數字顯示為一個百分數，在屬性面板中，對於想要顯示為百分數值的量值，在 ***Number formatting*** *下選取* ***Number****，然後從* ***Formatting*** *中選擇* ***Simple*** *以及其中一種 % 格式。*

範例：集合分析

以下範例說明如何在選取任何內容前使用集合分析來比較不同資料集。假設已選取 Q2，可使用集合定義為 {1} 的集合分析，忽略任何選項但依維度分割來計算所有值的加總。

範例：集合分析

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%

範例:TOTAL 限定詞與集合分析

以下範例說明如何在選取任何內容前及跨所有維度,結合使用集合分析和 TOTAL 限定詞來比較不同資料集。假設已選取 Q2,可使用集合定義為 {1} 的集合分析及 TOTAL 限定詞,忽略任何選項及維度來計算所有值的加總。

範例:TOTAL 限定詞與集合分析

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%

範例中使用的資料:

```
AggregationScope:
LOAD * inline [
Year Quarter Amount
2012 Q1 1100
2012 Q2 1700
2012 Q3 1400
2012 Q4 1800
2013 Q1 1000
2013 Q2 1300
2013 Q3 1100
2013 Q4 1400] (delimiter is ' ');
```

6.2 集合分析

在應用程式中進行選取時,您會定義資料中的記錄子集。會根據此子集計算彙總函數,例如 `Sum()`、`Max()`、`Min()`、`Avg()` 和 `Count()`。

換言之,您的選項定義彙總範圍;這定義進行計算的記錄集合。

集合分析提供定義範圍的方法,與由目前選項定義的記錄集合不同。此新範圍也可被視為替代選項。

若您想要比較目前選項與特殊值,例如去年的值或全球市場占比,則這會很實用。

集合運算式

集合運算式可用於彙總函數內部和外部,並以大括弧括住。

範例:內部集合運算式

```
Sum( {$<Year={2021}>} Sales )
```

範例：外部集合運算式

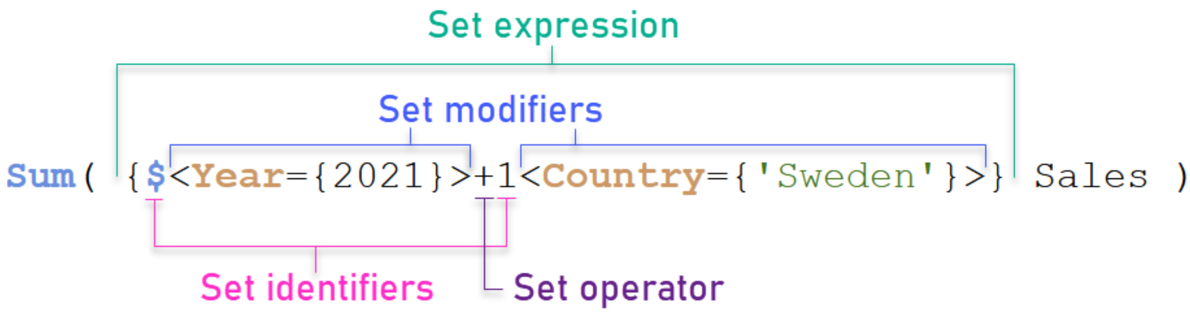
```
{<Year={2021}>} Sum(Sales) / Count(distinct Customer)
```

集合運算式包含以下元素的組合：

- **識別碼**。集合識別碼代表其他地方定義的選項。這也代表資料中的特定記錄集合。這可以是目前選項、來自書籤的選項或來自替代狀態的選項。簡單的集合運算式中包含單一識別碼，例如貨幣符號 {$}，表示目前選項中的所有記錄。
  範例：$、1、BookMark1、State2
- **運算子**。集合運算式可用來建立不同集合識別碼之間的聯集、差異或交集。以此方式，您可以建立由集合識別碼定義的選項子集或超集。
  範例：+、-、*、/
- **修飾詞**。集合修飾詞可新增至集合識別碼以變更其選項。修飾詞也能單獨使用，這將會修改預設識別碼。修飾詞必須以角括弧 <…> 括起。
  範例：<Year={2020}>、<Supplier={ACME}>

合併元素以形成集合運算式。

集合運算式中的元素



例如，以上集合運算式從彙總 Sum(Sales) 建置。

第一個運算元傳回目前選項的 2021 年銷售額，這由 $ 集合識別碼和包含 2021 年選項的修飾詞指示。第二個運算元傳回 Sweden 的 Sales，並忽略目前選項，這由 1 集合識別碼指示。

最終，運算式傳回包含記錄的集合，這些記錄屬於兩個集合運算元的任一個，由 + 集合運算子指示。

範例

合併以上集合運算式元素的範例在下列主題中提供：

自然集合

通常，集合運算式代表資料模型中的記錄集合，以及定義此資料子集的選項。在此情況下，集合稱為自然集合。

含有或不含集合修飾詞的集合識別碼一律代表自然集合。

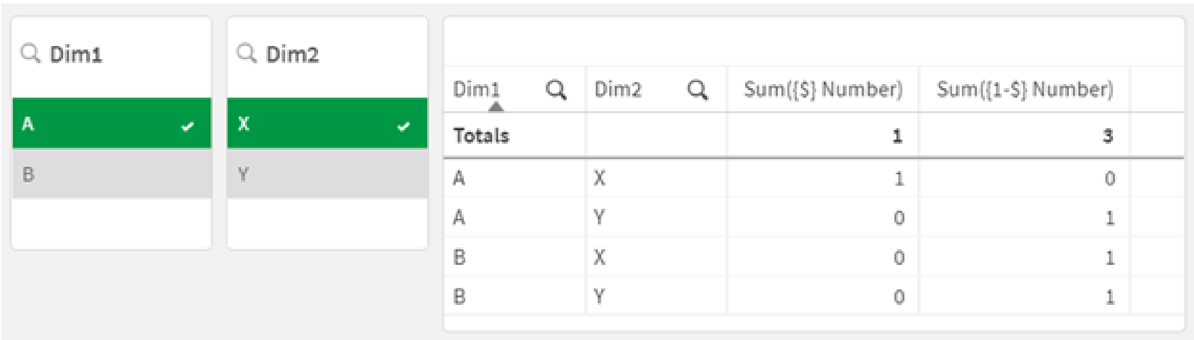
不過，使用集合運算子的集合運算式也代表記錄子集，但通常仍無法使用欄位值的選項來描述。這樣的運算式是非自然集合。

例如，`{1-$}` 提供的集合不可永遠由選項定義。因此這不是自然集合。可以載入下列資料、新增至表格然後使用篩選窗格進行選取，以顯示此。

```
Load * Inline
[Dim1, Dim2, Number
A, X, 1
A, Y, 1
B, X, 1
B, Y, 1];
```

為 `Dim1` 和 `Dim2` 進行選取後，就會取得顯示在下列表格中的檢視畫面。

含有自然和非自然集合的表格



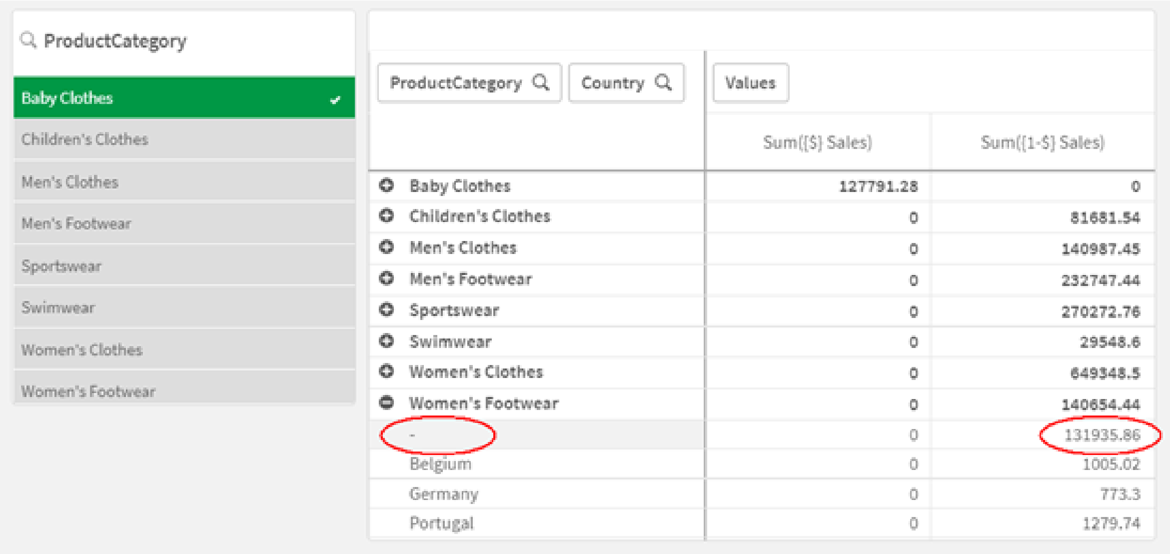
第一個量值中的集合運算式使用自然集合：這對應至進行 `{$}` 的選項。

第二個量值則不同。這使用 `{1-$}`。無法選取對應至此集合的選項，因此這不是自然集合。

此差異會造成一些後果：

- 集合修飾詞只能套用至集合識別碼。這些無法套用至任意集合運算式。例如，無法使用如下的集合運算式：
  `{ (BM01 * BM02) <Field={x,y}> }`
  在此，一般 (圓) 括弧暗示應在套用集合修飾詞之前評估 `BM01` 和 `BM02` 之間的交集。因為沒有可以修改的元素集。
- 您無法在 `P()` 和 `E()` 元素函數內部使用非自然集合。這些函數傳回元素集，但無法從非自然集合推算元素集。
- 若資料模型有許多表格，使用非自然集合的量值無法一律歸類到合適的維度值。例如，在下列圖表中，一些排除的銷售數字歸類到正確的 Country，而其他則有 NULL 作為 Country。

含有非自然集合的圖表



是否正確進行指派取決於資料模型。在此情況下，若這涉及選項排除的國家，則無法指派數字。

識別碼	描述
1	代表應用程式中所有記錄的完整集合，不考慮任何已選取的選項。
$	代表目前選項的記錄。因此，集合運算式 **{$}** 等同於未指明集合運算式。
$1	代表前一個選項。$2 代表前面一個選項，以此類推。
$_1	代表下一個 (前進) 選項。$_2 代表下面一個選項，以此類推。
BM01	您可使用任何書籤識別碼或書籤名稱。
MyAltState	您可以按其狀態名稱引用在輪替狀態機進行的選擇。

範例	結果
sum ({1} Sales)	傳回應用程式的總銷售額，會忽略選項，但不會忽略維度。
sum ({$} Sales)	會傳回目前選項的銷售額，亦即與 sum(Sales) 相同。
sum ({$1} Sales)	會傳回前一個選項的銷售。
sum ({BM01} Sales)	會傳回書籤 *BM01* 的銷售。

範例	結果
sum({$<OrderDate = DeliveryDate>} Sales)	會傳回目前選項的銷售，其中 OrderDate = DeliveryDate。

範例	**結果**
sum({1<Region = {US}>} Sales)	會傳回 US 區域的銷售額，忽略目前選項。
sum({$<Region = >} Sales)	會傳回目前選項的銷售額，但移除 *Region* 選項。
sum({<Region = >} Sales)	會傳回與上例相同的結果。當省略要修改的集合時，會採用 $。
sum({$<Year={2000}, Region={“U*”}>} Sales)	會傳回目前選項的銷售額，但為*Year* 和 *Region* 兩者中的新選項。

集合識別碼

集合識別碼代表資料中的記錄集合；可以是所有資料或資料子集。這是可由選項定義的記錄集合。這可以是目前選項、所有資料 (沒有選項)、來自書籤的選項或來自替代狀態的選項。

在範例 `Sum( {$<Year = {2009}>} Sales )` 中，識別碼是貨幣符號：`$`。這表示目前選項。這也表示所有可能的記錄。接著會用集合運算式中的修飾詞部分改變此集合：新增 `Year` 中的選項 `2009`。

在更複雜的集合運算式中，兩個識別碼可以與運算子一起使用，以形成兩個記錄集合的聯集、差異或交集。

下列表格顯示一些常見識別碼。

使用常見識別碼的範例

識別碼	**描述**
`1`	代表應用程式中所有記錄的完整集合，不考慮任何已選取的選項。
`$`	代表預設狀態中目前選項的記錄。因此，集合運算式 `{$}` 通常等同於未指明集合運算式。
`$1`	代表預設狀態中的上一個選項。$2 代表上面一個選項，以此類推。
`$_1`	代表下一個 (向前)選項。$_2 代表下面一個選項，以此類推。
`BM01`	您可使用任何書籤識別碼或書籤名稱。
`AltState`	您可以按其狀態名稱引用輪替狀態。
`AltState::BM01`	書籤包含所有狀態的選項，您可以讓書籤名稱合格，以參考特定書籤。

下列表格顯示使用不同識別碼的範例。

使用不同識別碼的範例

範例	**結果**
`Sum ({1} Sales)`	傳回應用程式的總銷售額，會忽略選項，但不會忽略維度。
`Sum ({$} Sales)`	會傳回目前選項的銷售額，亦即與 `Sum(Sales)` 相同。

範例	結果
Sum ({$1} Sales)	會傳回前一個選項的銷售。
Sum ({BM01} Sales)	會傳回書籤 BM01 的銷售。

集合運算子

集合運算子用來納入、排除或交集資料集。所有運算子都將集合作為運算元使用，並傳回一個集合作為結果。

您可以在兩種不同的情況下使用集合運算子：

- 在集合識別碼執行集合操作，表示資料中的記錄集合。
- 在元素集、欄位值或集合修飾詞內部執行集合操作。

下列表格顯示可在集合運算式中使用的運算子。

運算子

運算子	描述
+	聯集。此二元運算會傳回一個集合，其中包含的記錄或元件屬於兩個集合運算元中的任一個集合。
-	差集。此二元運算會傳回一個集合，其中包含的記錄或元件屬於兩個集合算元中的第一個集合，但不屬於第二個集合。此外，當作為一元運算子時，會傳回補集。
*	交集。此二元運算會傳回一個集合，其中包含的記錄或元件同時屬於兩個集合運算元。
/	對稱差 (XOR)。此二元運算會傳回一個集合，其中包含的記錄或元件屬於任一個集合運算元，但不同時屬於這兩個集合運算元。

下列表格顯示使用運算子的範例。

使用運算子的範例

範例	結果
Sum ({1-$} Sales)	傳回透過目前選取排除的所有項目的銷售額。
Sum ({$*BM01} Sales)	傳回選取與書籤 #160;BM01 之間交集的銷售額。
Sum ({-($+BM01)} Sales)	傳回選取與書籤 BM01 排除的銷售額。
Sum ({$<Year={2009}>+1<Country={'Sweden'}>} Sales)	傳回與目前選取相關的 2009 年銷售額，並新增與國家 Sweden 相關的所有年份的完整資料集。
Sum ({$<Country={"S*"}+{"*land"}>} Sales)	傳回以 S 開頭或以 land 結尾的國家銷售。

集合修飾詞

集合運算式用來定義計算範圍。集合運算式的中央部分是指定選項的集合修飾詞。這用來修改使用者選項，或集合識別碼中的選項，而結果會定義新的計算範圍。

集合修飾詞的組成為一個或多個欄位名稱、各個欄位的後面都加上要對該欄位做出的選項。修飾詞括在角括號內: `< >`

例如：

- `Sum ( {$<Year = {2015}>} Sales )`
- `Count ( {1<Country = {Germany}>} distinct OrderID )`
- `Sum ( {$<Year = {2015}, Country = {Germany}>} Sales )`

元素集

可以使用下列內容定義元素集:

- 值清單
- 搜尋
- 另一個欄位參考
- 集合函數

若省略元素集定義，集合修飾詞將會清除此欄位中的任何選項。例如：

```
Sum( {$<Year = >} Sales )
```

範例:根據元素集用於集合修飾詞的圖表運算式

範例 - 圖表運算式

載入指令碼

在資料載入編輯器中載入下列資料作為內嵌載入，以建立以下的圖表運算式範例。

```
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
];
```

圖表運算式

透過下列圖表運算式在 Qlik Sense 工作表中建立表格。

表格 - 以元素集為基礎的集合修飾詞

國家/地區	Sum(Sales)	Sum({1<Country={Belgium}>} Sales)	Sum ({1<Country={"*A*"}>} Sales)	Sum ({1<Country={"A*"}>} Sales)	Sum({1<Year={$(=Max(Year))}>} Sales)
總計	1645397.3	360809.2	1284588.1	443238.88	788617.07
阿根廷	206332.92	0	206332.92	206332.92	140037.89
奧地利	236905.96	0	236905.96	236905.96	182739.87
比利時	360809.2	360809.2	0	0	178042.33
巴西	176596.89	0	176596.89	0	2104.22
加拿大	142089.58	0	142089.58	0	40288.25
丹麥	152211.66	0	152211.66	0	106938.41
芬蘭	138148.99	0	138148.99	0	30583.44
法國	146341.24	0	146341.24	0	30696.98
德國	85960.86	0	85960.86	0	77185.68

解釋

- 維度:
  - `Country`
- 量值:
  - `Sum(Sales)`
    不含集合運算式的 `Sales` 總和。
  - `Sum({1<Country={Belgium}>}Sales)`
    選取 `Belgium`, 然後選取對應 `Sales` 的總和。
  - `Sum({1<Country={"*A*"}>}Sales)`
    選取具有 `A` 的所有國家, 然後選取對應 `Sales` 的總和。
  - `Sum({1<Country={"A*"}>}Sales)`

選取以 `A` 開頭的所有國家，然後選取對應 `Sales` 的總和。

- `Sum({1<Year={$(=Max(Year))}>}Sales)`
  計算 `Max(Year)`，這是 `2015`，然後選取對應 `Sales` 的總和。

以元素集合為基礎的集合修飾詞

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

列出的值

元素集的常見情況是會以括在大括弧內的欄位值清單為基礎。例如：

- `{$<Country = {Canada, Germany, Singapore}>}`
- `{$<Year = {2015, 2016}>}`

大括弧內部定義元素集。各個值以逗號分隔。

引號和區分大小寫

若值包含空白或特殊字元，則值需要加上引號。單引號將是常值、區分大小寫並符合單一欄位值。雙引號暗示區分大小寫並符合一個或數個欄位值。例如：

- `<Country = {'New Zealand'}>`
  僅符合 `New Zealand`。
- `<Country = {"New Zealand"}>`
  符合 `New Zealand`、`NEW ZEALAND` 和 `new zealand`。

日期必須括在引號內，並使用有問題的欄位的日期格式。例如：

- `<ISO_Date = {'2021-12-31'}>`
- `<US_Date = {'12/31/2021'}>`
- `<UK_Date = {'31/12/2021'}>`

方括弧或重音符號可以替代雙引號。

搜尋

也可以透過搜尋建立元素集。例如：

- <Country = {"C*"}>
- <Ingredient = {"*garlic*"}>
- <Year = {">2015"}>
- <Date = {">12/31/2015"}>

萬用字元可用於文字搜尋:星號 (*) 表示任何數量的字元, 而問號 (?) 表示單一字元。關係運算子可用來定義數字搜尋。

您應永遠使用雙引號進行搜尋。搜尋不區分大小寫。

貨幣擴充

若您想要在元素集內部使用計算, 則需要貨幣擴充。例如, 若您只想要查看可用的上一年, 可以使用:

```
<Year = {$(=Max(Year))}>
```

其他欄位中已選取的值

修飾詞能夠以另一個欄位的選取值為基礎。例如：

```
<OrderDate = DeliveryDate>
```

此修飾詞會取用 DeliveryDate 的選取值, 然後將其套用為 OrderDate 的選項。如果有許多相異值 (超過幾百個), 則此運算會需要大量的 CPU, 應避免使用。

元素集函數

元素集也可以根據集合功能 P() (可能值) 和 E() (排除值)。

例如, 若您想要選取銷售產品 Cap 的國家, 可以使用:

```
<Country = P({1<Product={Cap}>} Country)>
```

同樣地, 若您想要挑選出尚未銷售產品 Cap 的國家, 可以使用:

```
<Country = E({1<Product={Cap}>} Country)>
```

含搜尋的集合修飾詞

您可以使用集合修飾詞透過搜尋建立元素集。

例如：

- <Country = {"C*"}>
- <Year = {">2015"}>
- <Ingredient = {"*garlic*"}>

應永遠以雙引號、方括號或重音符號括起搜尋。您可以混用常值字串 (單引號) 和搜尋 (雙引號) 以使用清單。例如：

```
<Product = {'Nut', "*Bolt", Washer}>
```

文字搜尋

萬用字元和其他符號可用於文字搜尋：

- 星號 (*) 將代表任何字元數量。
- 問號 (?) 將代表單一字元。
- 抑揚符號 (^) 將標記一個字的開頭。

例如：

- `<Country = {"C*", "*land"}>`
  比對所有以 `C` 開頭或以 `land` 結尾的國家。
- `<Country = {"*^z*"}>`
  這將比對含有以 `z` 開頭之字的所有國家，例如 `New Zealand`。

數值搜尋

您可以使用這些關係運算子進行數值搜尋：`>`、`>=`、`<`、`<=`

數值搜尋永遠以這些運算子之一開頭。例如：

- `<Year = {">2015"}>`
  比對 2016 和後續年份。
- `<Date = {">=1/1/2015<1/1/2016"}>`
  在 2015 年期間比對所有日期。記下描述兩個日期之間時間範圍的語法。日期格式需要符合問題中欄位的日期格式。

運算式搜尋

您可以使用運算式搜尋以進行更進階的搜尋。然後將針對搜尋欄位中的每個欄位值評估彙總。將選取搜尋運算式傳回 true 的所有值。

運算式搜尋通常以等號開頭：`=`

例如：

```
<Customer = {"=Sum(Sales)>1000"}>
```

這將會傳回銷售值大於 1000 的所有客戶。`Sum(Sales)` 在目前選項上計算。這表示，若您在另一個欄位中有選項，例如 `Product` 欄位，則您只會取得滿足所選產品銷售條件的客戶。

若您希望條件獨立於選項，需要在搜尋字串內部使用集合分析。例如：

```
<Customer = {"=Sum({1} Sales)>1000"}>
```

等號之後的運算式將會解譯為布林值。這表示，若這對其他內容進行評估，則任何非零數字將會解譯為 true，而零和字串將會解譯為 false。

引號

若搜尋字串包含空白或特殊字元，請使用引號。單引號暗示常值、區分大小寫並符合單一欄位值。雙引號暗示區分大小寫的搜尋，這可能符合多個欄位值。

例如：

- `<Country = {'New Zealand'}>`
  僅符合 New Zealand。
- `<Country = {"New Zealand"}>`
  符合 New Zealand、NEW ZEALAND 和 new zealand

方括弧或重音符號可以替代雙引號。

> *在先前版本的 Qlik Sense 中，沒有區分單引號和雙引號，所有引號內的字串都視為搜尋。為維持向下相容性，透過舊版 Qlik Sense 建立的應用程式將會依照先前版本的方式繼續運作。透過 Qlik Sense November 2017 或更高版本建立的應用程式將會顧及兩種引號類型之間的差異。*

範例：含搜尋的集合修飾詞的圖表運算式

範例 - 圖表運算式

載入指令碼

在資料載入編輯器中載入下列資料作為內嵌載入，以建立以下的圖表運算式範例。

```
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];
```

範例 1：含文字搜尋的圖表運算式

透過下列圖表運算式在 Qlik Sense 工作表中建立表格。

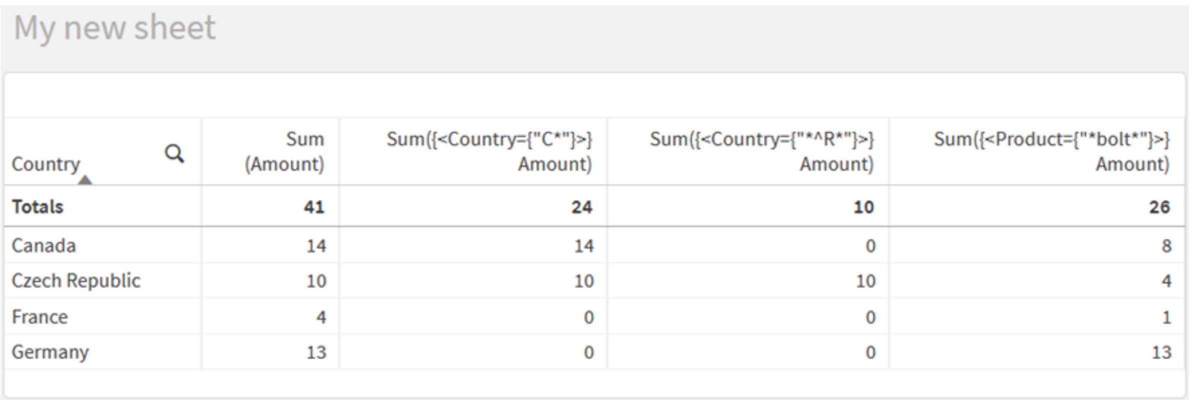
表格 - 含文字搜尋的集合修飾詞

國家/地區	Sum (Amount)	Sum({<Country={"C*"}>} Amount)	Sum({<Country={"*^R*"}>} Amount)	Sum({<Product={"*bolt*"}>} Amount)
總計	**41**	**24**	**10**	**26**
加拿大	14	14	0	8
捷克	10	10	10	4
法國	4	0	0	1
德國	13	0	0	13

解釋

- 維度：
  - Country
- 量值：
  - Sum(Amount)
    不含集合運算式的 Amount 總和。
  - Sum({<Country={"C*"}>}Amount)
    以 C 開頭的所有國家的總和 Amount，例如 Canada 和 Czech Republic。
  - Sum({<Country={"*^R*"}>}Amount)
    有個字以 R 開頭的所有國家的總和 Amount，例如 Czech Republic。
  - Sum({<Product={"*bolt*"}>}Amount)
    包含字串 bolt 的所有產品的總和 Amount，例如 Bolt 和 Anchor bolt。

含文字搜尋的集合修飾詞


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

範例 2：含數值搜尋的圖表運算式

透過下列圖表運算式在 Qlik Sense 工作表中建立表格。

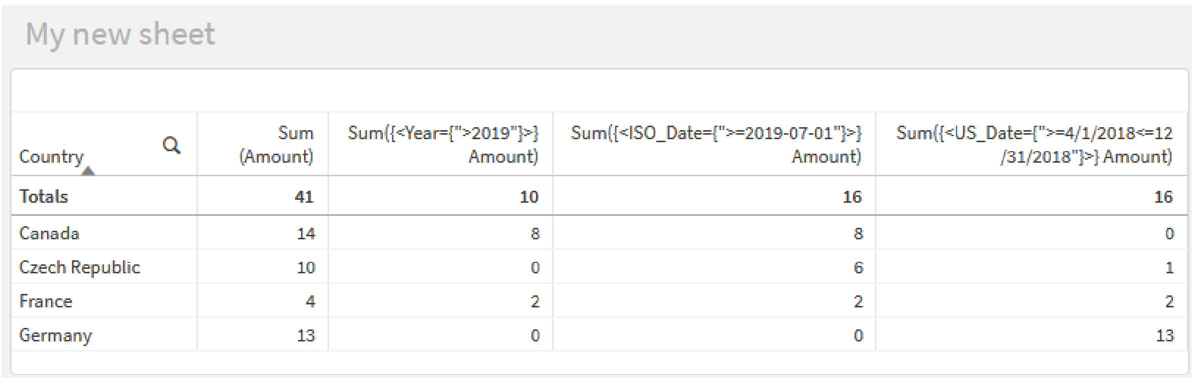
表格 - 含數值搜尋的集合修飾詞

國家/地區	Sum (Amount)	Sum({<Year={">2019"}>} Amount)	Sum({<ISO_Date={">=2019-07-01"}>} Amount)	Sum({<US_Date={">=4/1/2018<=12/31/2018"}>} Amount)
總計	**41**	**10**	**16**	**16**
加拿大	14	8	8	0
捷克	10	0	6	1
法國	4	2	2	2
德國	13	0	0	13

解釋

- 維度：
    - Country
- 量值：
    - Sum(Amount)
      不含集合運算式的 Amount 總和。
    - Sum({<Year={">2019"}>}Amount)
      2019 之後所有年份的總和 Amount。
    - Sum({<ISO_Date={">=2019-07-01"}>}Amount)
      2019-07-01 當日或之後所有日期的總和 Amount。搜尋中的日期格式必須符合欄位格式。
    - Sum({<US_Date={">=4/1/2018<=12/31/2018"}>}Amount)
      從 4/1/2018 至 12/31/2018 所有日期的總和 Amount，包括開始和結束日期。搜尋中的日期格式必須符合欄位格式。

含數值搜尋的集合修飾詞


My new sheet

Country	Sum (Amount)	Sum({<Year={">2019"}>} Amount)	Sum({<ISO_Date={">=2019-07-01"}>} Amount)	Sum({<US_Date={">=4/1/2018<=12 /31/2018"}>} Amount)
Totals	41	10	16	16
Canada	14	8	8	0
Czech Republic	10	0	6	1
France	4	2	2	2
Germany	13	0	0	13

範例 3：含運算式搜尋的圖表運算式

透過下列圖表運算式在 Qlik Sense 工作表中建立表格。

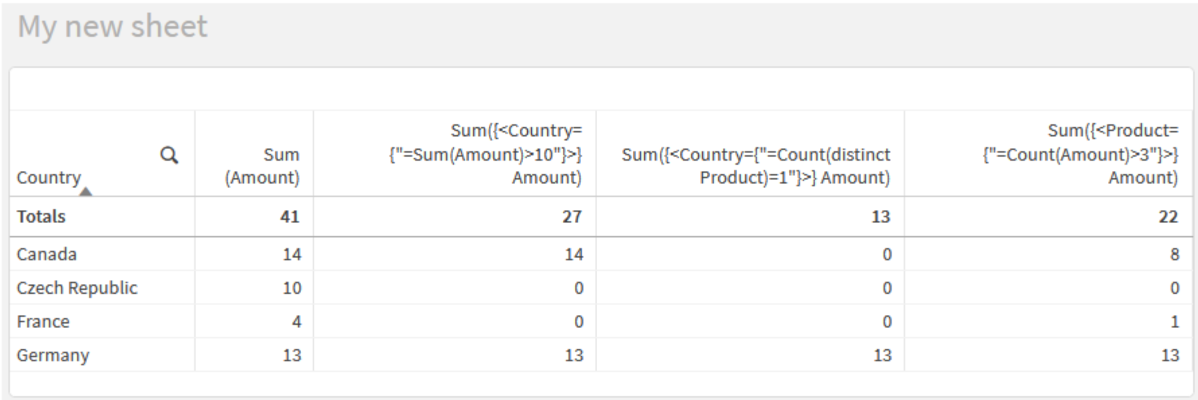
Table - Set modifiers with expression searches

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

解釋

- 維度：
    - `Country`
- 量值：
    - `Sum(Amount)`
      不含集合運算式的 `Amount` 總和。
    - `Sum({<Country={"=Sum(Amount)>10"}>}Amount)`
      `Amount` 彙總總和大於 10 的所有國家的總和 `Amount`。
    - `Sum({<Country={"=Count(distinct Product)=1"}>}Amount)`
      與確切一個相異產品相關的所有國家的總和 `Amount`。
    - `Sum({<Product={"=Count(Amount)>3"}>}Amount)`
      在資料中有超過三筆交易的所有國家的總和 `Amount`。

含運算式搜尋的集合修飾詞


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

範例	結果
sum({$–1<Product = {“*Internal*”, “*Domestic*”}>} Sales)	會傳回目前選項的銷售額，但排除與產品名稱中含字串「Internal」或「Domestic」之產品相關的交易。

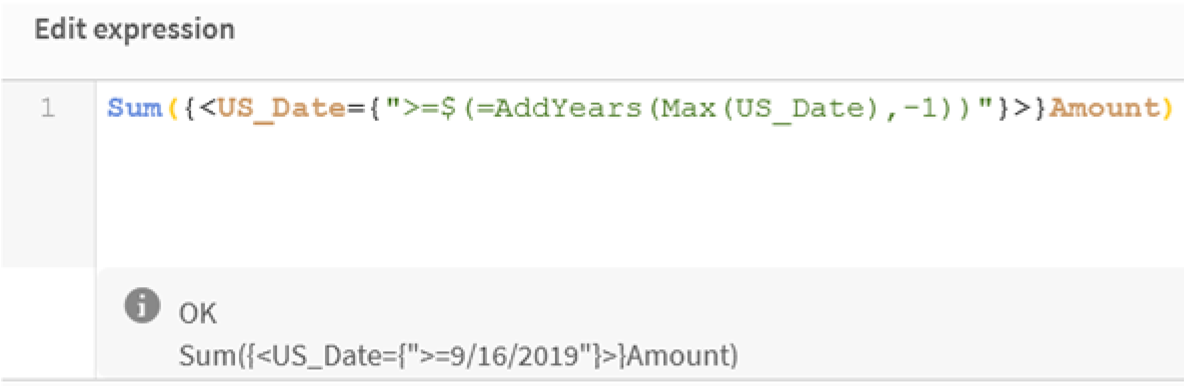
範例	結果
sum({$<Customer = {“=Sum({1<Year = {2007}>} Sales) > 1000000”}>} Sales)	會傳回目前選項的銷售額，但包含「Customer」欄位中新的選項：僅針對在 2007 年期間總銷售額超過 1000000 的客戶。

含貨幣符號展開的集合修飾詞

貨幣符號展開是在剖析並評估運算式之前計算的建構。結果則插入到運算式中，而非 `$(…)`。然後會使用貨幣展開的結果進行運算式的計算。

運算式編輯器會顯示貨幣展開預覽，讓您可以驗證貨幣符號展開要評估的內容。

運算式編輯器中的貨幣符號展開預覽



若您想要在元素集內部使用計算，請使用貨幣符號展開。

例如，若您希望僅查看上次可能的年份，可以使用下列建構：

```
<Year = {$(=Max(Year))}>
```

會先計算 `Max(Year)`，結果會插入運算式中，而非 `$(…)`。

貨幣展開後的結果將會是如下的運算式：

```
<Year = {2021}>
```

會根據目前選項計算貨幣展開內部的運算式。這表示，若您在另一個欄位有選項，將會影響運算式的結果。

若您希望計算獨立於選項，請在貨幣展開內部使用集合分析。例如：

```
<Year = {$(=Max({1} Year))}>
```

字串

若您希望貨幣展開產生字串，則一般引用規則適用。例如：

```
<Country = {'$(=FirstSortedValue(Country,Date)'}>
```

貨幣展開後的結果將會是如下的運算式：

```
<Country = {'New Zealand'}>
```

若您沒有使用引號，將會取得語法錯誤。

數字

若您希望貨幣展開產生數字，請確保展開取得與欄位相同的格式。這表示，您有時候需要以格式化函數包裹運算式。

例如：

```
<Amount = {$(=Num(Max(Amount), '###0.00'))}>
```

貨幣展開後的結果將會是如下的運算式：

```
<Amount = {12362.00}>
```

使用雜湊以強制展開永遠使用小數點，而非千位分隔符。例如：

```
<Amount = {$(#=Max(Amount))}>
```

日期

若您希望貨幣展開產生日期，請確保展開具有正確的格式。這表示，您有時候需要以格式化函數包裹運算式。

例如：

```
<Date = {'$(=Date(Max(Date)))'}>
```

貨幣展開後的結果將會是如下的運算式：

```
<Date = {'12/31/2015'}>
```

如同使用字串，您需要使用正確的引號。

常用情況是您希望計算限制為上個月 (或去年)。則您可以結合 `AddMonths()` 函數使用數值搜尋。

例如：

```
<Date = {">=$(=AddMonths(Today(),-1))"}>
```

貨幣展開後的結果將會是如下的運算式：

```
<Date = {">=9/31/2021"}>
```

這將會挑選出上個月發生的所有事件。

範例：含貨幣符號展開的集合修飾詞的圖表運算式

範例 - 圖表運算式

載入指令碼

在資料載入編輯器中載入下列資料作為內嵌載入，以建立以下的圖表運算式範例。

```
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];
```

含貨幣符號展開的圖表運算式

透過下列圖表運算式在 Qlik Sense 工作表中建立表格。

表格 - 含貨幣符號展開的集合修飾詞

國家/地區	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)
總計	**41**	**1**	**6**	**1**
加拿大	14	0	6	0
捷克	10	0	0	0
法國	4	1	0	1
德國	13	0	0	0

解釋

- 維度:
  - Country
- 量值:
  - Sum(Amount)
    不含集合運算式的 Amount 總和。
  - Sum({<US_Date={'$(vToday)'}>}Amount)
    US_Date 與變數 vToday 相同的所有記錄的總和 Amount。
  - Sum({<ISO_Date={"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>}Amount)
    ISO_Date 與第一個 (最小) 可能的 ISO_Date 相同的所有記錄的總和 Amount。需要 Date() 函數才能確保日期格式符合欄位格式。
  - Sum({<US_Date={">=$(=AddYears(Max(US_Date),-1))"}>}Amount)

在最晚 (最大) 可能的 US_Date 之前的年份日期之後或當日具有 US_Date 的所有記錄的總和 Amount。AddYears() 函數將會以變數 DateFormat 指定的格式傳回日期，而這需要符合欄位 US_Date 的格式。

含貨幣符號展開的集合修飾詞

My new sheet

Country	Sum (Amount)	Sum({<US_Date={'$(vToday)'}>} Amount)	Sum({<ISO_Date= {"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>} Amount)	Sum({<US_Date= {">=$(=AddYears(Max(US_Date),-1))"}>} Amount)
Totals	**41**	**1**	**6**	**1**
Canada	14	0	6	0
Czech Republic	10	0	0	0
France	4	1	0	1
Germany	13	0	0	0

範例	結果
sum({$<Year = {$(#vLastYear)}>} Sales)	會傳回與目前選項相關的前一年銷售。此處，貨幣符號展開是用於包含相關年份的變數 vLastYear。
sum({$<Year = {$(#=Only(Year)-1)}>} Sales)	會傳回與目前選項相關的前一年銷售。此處，貨幣符號展開是用於計算前一年。

含集合運算子的集合修飾詞

集合運算子用來納入、排除或交集不同的元素集。這結合不同的方法來定義元素集。

運算子與用於集合識別碼的內容相同。

運算子

運算子	描述
+	聯集。此二元運算會傳回一個集合，其中包含的記錄或元件屬於兩個集合運算元中的任一個集合。
-	差集。此二元運算會傳回一個集合，其中包含的記錄或元件屬於兩個集合算元中的第一個集合，但不屬於第二個集合。此外，當作為一元運算子時，會傳回補集。
*	交集。此二元運算會傳回一個集合，其中包含的記錄或元件同時屬於兩個集合運算元。
/	對稱差 (XOR)。此二元運算會傳回一個集合，其中包含的記錄或元件屬於任一個集合運算元，但不同時屬於這兩個集合運算元。

例如，以下兩個修飾詞定義相同的欄位值集合：

- <Year = {1997, "20*"}>
- <Year = {1997} + {"20*"}>

兩個運算式都選取 1997 和以 20 為開頭的年份。換言之，這是兩個條件的聯集。

集合運算子也允許更複雜的定義。例如：

```
<Year = {1997, "20*"} - {2000}>
```

此運算式將會選取與上述相同的年份，但也會排除 2000 年。

。

範例：含集合運算子的集合修飾詞的圖表運算式

範例 - 圖表運算式

載入指令碼

在資料載入編輯器中載入下列資料作為內嵌載入，以建立以下的圖表運算式範例。

```
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];
```

圖表運算式

透過下列圖表運算式在 Qlik Sense 工作表中建立表格。

表格 - 含集合運算子的集合修飾詞

國家/地區	Sum(Amount)	Sum({<Year={">2018"}-{2020}>}Amount)	Sum({<Country=-{Germany}>}Amount)	Sum({<Country={Germany}+P({<Product={Nut}>}Country)>}Amount)
總計	**41**	**9**	**28**	**17**
加拿大	14	0	14	0
捷克	10	9	10	0

國家/地區	Sum (Amount)	Sum({<Year={">2018"}-{2020}>} Amount)	Sum({<Country=-{Germany}>} Amount)	Sum({<Country={Germany}+P({<Product={Nut}>}Country)>} Amount)
總計	**41**	**9**	**28**	**17**
法國	4	0	4	4
德國	13	0	0	13

解釋

- 維度：
  - `Country`
- 量值：
  - `Sum(Amount)`
    不含集合運算式的 `Amount` 總和。
  - `Sum({<Year={">2018"}-{2020}>}Amount)`
    2018 之後所有年份的總和 `Amount`，2020 除外。
  - `Sum({<Country=-{Germany}>}Amount)`
    所有國家的總和 `Amount`，`Germany` 除外。記下一元排除運算子。
  - `Sum({<Country={Germany}+P({<Product={Nut}>}Country)>}Amount)`
    `Germany` 以及所有與產品 `Nut` 相關國家的總和 `Amount`。

含集合運算子的集合修飾詞

My new sheet

Country	Sum (Amount)	Sum({<Year={">2018"}-{2020}>} Amount)	Sum({<Country= - {Germany}>} Amount)	Sum({<Country={Germany}+P({<Product={Nut}>} Country)>} Amount)
Totals	41	9	28	17
Canada	14	0	14	0
Czech Republic	10	9	10	0
France	4	0	4	4
Germany	13	0	0	13

範例	結果
sum({$<Product = Product + {OurProduct1} – {OurProduct2} >} Sales)	會傳回目前選項的銷售額，但「OurProduct1」產品會加入所選產品的清單中，而「OurProduct2」則會從所選產品的清單中移除。
sum({$<Year = Year + ({“20*”,1997} – {2000}) >} Sales)	會傳回目前選項的銷售額，但加上「Year」欄位中的其他選項：1997 年和以「20」開頭的所有年份，不過不包含 2000 年。 請注意，如果 2000 年包含在目前選項中，則修改後仍會包含在內。

範例	結果
sum({$<Year = (Year + {“20*”,1997}) – {2000} >} Sales)	會傳回與上例幾乎相同的結果，但此處會排除 2000 年，即使起初包含在目前選項中。此範例顯示了有時候使用括弧定義優先順序的重要性。
sum({$<Year = {“*”} – {2000}, Product = {“*bearing*”} >} Sales)	會傳回目前選項的銷售額，但包含「Year」中的新選項：除了 2000 年之外的所有年份；而且僅針對包含字串「bearing」的產品。

含隱含集合運算子的集合修飾詞

在集合修飾詞中撰寫選項的標準方式是使用等號。例如：

```
Year = {">2015"}
```

集合修飾詞中的等號右側的運算式稱為元素集。這定義一組相異欄位值，換言之即選項。

此標記法會定義新的選項，而忽略欄位中的目前選項。因此，若集合識別碼包含此欄位中的選項，則舊的選項將被元素集中的選項取代。

若您想要以欄位中的目前選項作為選取基礎，需要使用不同的運算式

例如，若您想要採用舊的選項，並新增年份晚於 2015 年的要求，可以撰寫下列內容：

```
Year = Year * {">2015"}
```

星號是定義交集的集合運算子，因此您將會取得 Year 中目前選項之間的交集，以及年份晚於 2015 的附加要求。替代方式是以下列內容撰寫此部分：

```
Year *= {">2015"}
```

也就是說，指派運算子 (*=) 隱含性地定義交集。

類似地，可以使用下列內容定義隱含的聯集、排除和對稱差：+=、–=、/=

範例：含隱含集合運算子的集合修飾詞的圖表運算式

範例 - 圖表運算式

載入指令碼

在資料載入編輯器中載入下列資料作為內嵌載入，以建立以下的圖表運算式範例。

```
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];
```

含隱含集合運算子的圖表運算式

透過下列圖表運算式在 Qlik Sense 工作表中建立表格。

從國家清單中選取 Canada 和 Czech Republic。

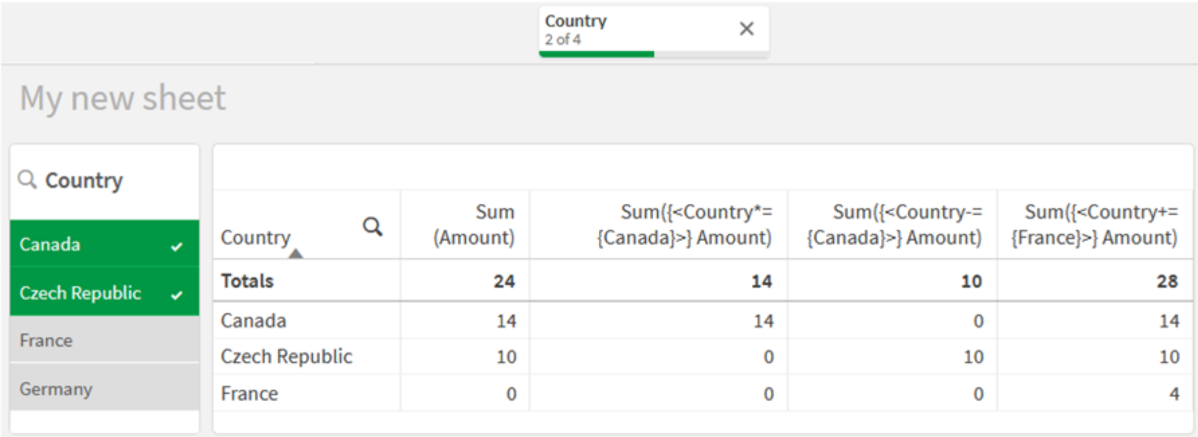
表格 - 含隱含集合運算子的圖表運算式

國家/地區	Sum (Amount)	Sum({<Country*={Canada}>} Amount)	Sum({<Country-={Canada}>} Amount)	Sum({<Country+={France}>} Amount)
總計	**24**	**14**	**10**	**28**
加拿大	14	14	0	14
捷克	10	0	10	10
法國	0	0	0	4

解釋

- 維度:
  - Country
- 量值:
  - Sum(Amount)
    目前選項的總格 Amount 請注意, 只有 Canada 和 Czech Republic 有非零值。
  - Sum({<Country*={Canada}>}Amount)
    目前選項的總和 Amount, 與 Country 為 Canada 的要求交集。若 Canada 不屬於使用者選項, 則集合運算式會傳回空的集合, 該欄在所有列都會是 0。
  - Sum({<Country-={Canada}>}Amount)
    目前選項的總和 Amount, 但首先從 Country 選項中排除 Canada。若 Canada 不屬於使用者選項, 則集合運算式不會變更任何數字。
  - Sum({<Country+={France}>}Amount)
    目前選項的總和 Amount, 但首先將 France 新增至 Country 選項。若 France 已屬於使用者選項, 則集合運算式不會變更任何數字。

含隱含集合運算子的集合修飾詞



範例	結果
sum({$<Product += {OurProduct1, OurProduct2} >} Sales)	會傳回目前選項的銷售額，但使用隱含聯集將「OurProduct1」和「OurProduct2」產品加入所選產品的清單中。
sum({$<Year += {“20*”,1997} – {2000} >} Sales)	會傳回目前選項的銷售額，但使用隱含聯集將一些年份加入選項中：1997 年和以「20」開頭的所有年份，不過不包含 2000 年。 請注意，如果 2000 年包含在目前選項中，則修改後仍會包含在內。相同於 `<Year=Year + ({“20*”,1997}–{2000})>`。
sum({$<Product *= {OurProduct1} >} Sales)	會傳回目前選項的銷售額，但僅針對目前所選產品和 OurProduct1 產品的交集。

使用集合函數的集合修飾詞

有時候您需要使用巢狀集合定義來定義欄位值集合。例如，您可能希望選取已購買特定產品的所有客戶，而不選取該產品。

在這類情況下，使用元素集函數 `P()` 和 `E()`。這會分別傳回欄位可能值和排除值的元素集。在括弧內，可以在問題中指定欄位，以及定義範圍的集合運算式。例如：

```
P({1<Year = {2021}>} Customer)
```

這將會傳回在 2021 年曾有交易的客戶集合。然後您可以在集合修飾詞中使用此。例如：

```
Sum({<Customer = P({1<Year = {2021}>} Customer)>} Amount)
```

此集合運算式將會選取這些客戶，但不會將選項限制為 2021。

這些函數無法用於其他運算式。

此外，在元素集函數內部只能使用自然集合。即可由單一選項定義的一組記錄。

例如，`{1-$}` 指定的集合無法一直透過選取來定義，因此不是自然集合。將這些函數用於非自然集合會傳回非預期的結果。

範例:使用集合函數的集合修飾詞的圖表運算式

範例 - 圖表運算式

載入指令碼

在資料載入編輯器中載入下列資料作為內嵌載入,以建立以下的圖表運算式範例。

```
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];
```

圖表運算式

透過下列圖表運算式在 Qlik Sense 工作表中建立表格。

表格 - 使用集合函數的集合修飾詞

國家/地區	Sum (Amount)	Sum({<Country=P({<Year={2019}>}Country)>} Amount)	Sum({<Product=P({<Year={2019}>}Product)>} Amount)	Sum({<Country=E({<Product={Washer}>}Country)>} Amount)
總計	**41**	**10**	**17**	**13**
加拿大	14	0	6	0
捷克	10	10	10	0
法國	4	0	1	0
德國	13	0	0	13

解釋

- 維度：
  - `Country`
- 量值：
  - `Sum(Amount)`
    不含集合運算式的 `Amount` 總和。
  - `Sum({<Country=P({<Year={2019}>} Country)>} Amount)`
    與年份 2019 相關國家的總和 `Amount`。不過，這不會將計算限制為 2019。
  - `Sum({<Product=P({<Year={2019}>} Product)>} Amount)`
    與年份 2019 相關產品的總和 `Amount`。不過，這不會將計算限制為 2019。
  - `Sum({<Country=E({<Product={Washer}>} Country)>} Amount)`
    與產品 `Washer` 相關國家的總和 `Amount`。

使用集合函數的集合修飾詞

My new sheet

Country	Sum (Amount)	Sum({<Country=P({<Year={2019}>} Country)>} Amount)	Sum({<Product=P({<Year={2019}>} Product)>} Amount)	Sum({<Country=E({<Product={Washer}>} Country)>} Amount)
Totals	**41**	**10**	**17**	**13**
Canada	14	0	6	0
Czech Republic	10	10	10	0
France	4	0	1	0
Germany	13	0	0	13

範例	**結果**
sum({$<Customer = P({1<Product={'Shoe'}>} Customer)>} Sales)	會傳回目前選項的銷售額，但僅針對曾經買過「Shoe」產品的客戶。元素函數 P( ) 在此會傳回可能客戶的清單，亦即以欄位 Product 中選項為「Shoe」所表示的客戶。
sum({$<Customer = P({1<Product={'Shoe'}>})>} Sales)	同上。如果省略元素函數中的欄位，則函數會傳回外部指派中所指定之欄位的可能值。
sum({$<Customer = P({1<Product={'Shoe'}>} Supplier)>} Sales)	會傳回目前選項的銷售額，但僅針對曾經供應過產「Shoe」品的客戶，亦即客戶也是供應商。元素函數 P( ) 在此會傳回可能客戶的清單，亦即以欄位 Product 中選項為「Shoe」所表示的供應商。應商的清單接著會做為欄位 Customer 中的選項。
sum({$<Customer = E({1<Product={'Shoe'}>})>} Sales)	會傳回目前選項的銷售額，但僅針對不曾買過「Shoe」產品的客戶。元素函數 P( ) 在此會傳回已排除客戶的清單，亦即以欄位 Product 中選項為「Shoe」所排除的客戶。

內部和外部集合運算式

集合運算式可用於彙總函數內部和外部, 並以大括弧括住。

使用彙總函數內部的集合運算式時, 看起來會是如此:

範例: 內部集合運算式

```
Sum( {$<Year={2021}>} Sales )
```

若您的運算式有多個彙總並且想要避免在每個彙總函數寫入相同的集合運算式, 請使用彙總函數外部的集合運算式。

若您使用外部集合運算式, 必須置於範圍開頭。

範例: 外部集合運算式

```
{<Year={2021}>} Sum(Sales) / Count(distinct Customer)
```

若您使用彙總函數外部的集合運算式, 也可以在現有的主量值套用。

範例: 套用至主量值的外部集合運算式

```
{<Year={2021}>} [Master Measure]
```

彙總函數外部使用的集合運算式會影響整個運算式, 除非這以括弧括住, 然後括弧定義範圍。在下面的語彙範圍範例中, 集合運算式僅適用於括弧內部的彙總。

範例: 語彙範圍

```
( {<Year={2021}>} Sum(Amount) / Count(distinct Customer) ) – Avg(CustomerSales)
```

規則

語彙範圍

集合運算式會影響整個運算式, 除非這以括弧括住。若如此, 括弧會定義語彙範圍。

位置

集合運算式必須置於語彙範圍開頭。

內容

內容是與運算式相關的選項。傳統上, 內容一直是目前選項的預設狀態。但若物件設定為替代狀態, 則內容是目前選項的替代狀態。

您也能以外部集合運算式的形式定義內容。

繼承

內部集合運算式優先於外部集合運算式。若內部集合運算式包含集合識別碼, 則會取代內容。否則, 將會合併內容和集合運算式。

- {$<SetExpression>} - 覆寫外部集合運算式
- {<SetExpression>} - 與外部集合運算式合併

元素集合指派

元素集合指派決定如何合併兩個選項。若使用一般等號，則以內部集合運算式中的選項為優先。否則，將會使用隱含的集合運算子。

- `{<Field={value}>}` - 此內部選項會取代 “`Field`” 中的任何外部選項。
- `{<Field+={value}>}` - 此內部選項使用聯集運算子與 “`Field`” 中的外部選項合併。
- `{<Field*={value}>}` - 此內部選項使用交集運算子與 “`Field`” 中的外部選項合併。

多個步驟中的繼承

繼承可發生於多個步驟。範例：

- 目前選項 → `Sum(Amount)`
  彙總函數將會使用內容，在此為目前選項。
- 目前選項 → `{<Set1>} Sum(Amount)`
  `Set1` 將繼承自目前選項，結果將會是彙總函數的內容。
- 目前選項 → `{<Set1>} ({<Set2>} Sum(Amount))`
  `Set2` 將繼承自 `Set1`，這依次繼承自目前選項，結果將會是彙總函數的內容。

Aggr() 函數

`Aggr()` 函數建立具有兩個獨立彙總的巢狀彙總。在以下範例中，會為 `Dim` 的每個值計算 `Count()`，並使用 `Sum()` 函數彙總產生的陣列。

範例：

`Sum(Aggr(Count(X),Dim))`
`Count()` 是內部彙總，而 `Sum()` 是外部彙總。

- 內部彙總沒有從外部彙總繼承任何內容。
- 內部彙總從 `Aggr()` 函數繼承內容，這可能包含集合運算式。
- `Aggr()` 函數和外部彙總函數都從外部集合運算式繼承內容。

教學課程 - 建立集合運算式

您可以在 Qlik Sense 中建置集合運算式以支援資料分析。在此脈絡中，分析通常稱為集合分析。集合分析提供定義範圍的方法，與由應用程式中的目前選項定義的記錄集合不同。

您將會學習的內容

本教學課程提供資料和圖表運算式，以建置使用集合修飾詞、識別碼和運算子的集合運算式。

誰應完成本教學課程

本教學課程適合可順利使用指令碼編輯器和圖表運算式的應用程式開發人員。

開始前需要進行的事項

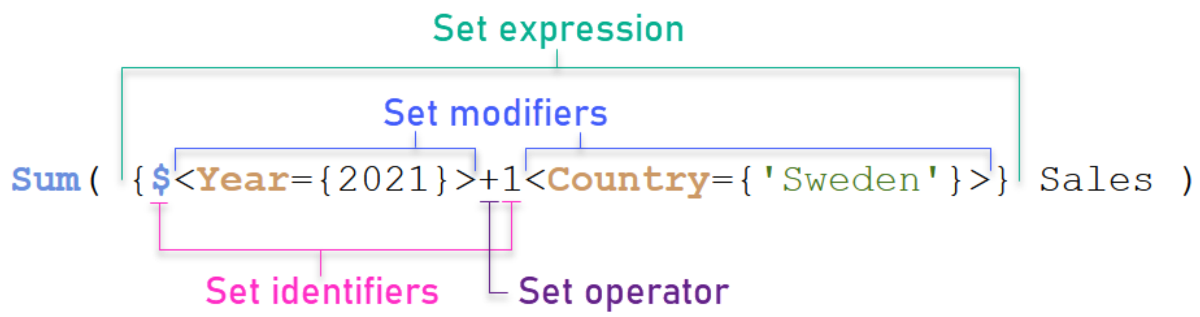
Qlik Sense Enterprise Professional 存取配置，這允許您載入資料並建立應用程式。

- [集合分析第 1 部分：初學者的簡介]
- [集合分析第 2 部分]

集合運算式中的元素

集合運算式括在彙總函數中，例如 Sum()、Max()、Min()、Avg() 或 Count()。集合運算式從已知為元素的建置組塊建構。這些元素是集合修飾詞、識別碼和運算子。

集合運算式中的元素



例如，以上集合運算式從彙總 Sum(Sales) 建置。集合運算式括在外側大括弧內：{ }

運算式中的第一個運算元：$<Year={2021}>

此運算元傳回目前選項 2021 年份的銷售。修飾詞 <Year={2021}> 包含 2021 年的選項。$ 集合識別碼指示集合運算式基於目前選項。

運算式中的第二個運算元：1<Country={'Sweden'}>

此運算元傳回 Sweden 的 Sales。修飾詞 <Country={'Sweden'}> 包含 Sweden 國家的選項。1 集合識別碼指示將會忽略應用程式中選取的選項。

最終，+ 集合運算子指示運算式傳回包含記錄的集合，這些記錄屬於兩個集合運算元的任一個。

建立集合運算式教學課程

完成下列程序以建立顯示在本教學課程的集合運算式。

建立新的應用程式並載入資料

請執行下列動作：

1. 建立新應用程式。
2. 按一下**指令碼編輯器**。或者，按一下導覽列中的**準備 > 資料載入編輯器**。
3. 在**資料載入編輯器**中建立新的區段。
4. 複製下列資料，並在新的區段中貼上：*集合運算式教學課程資料 (page 300)*
5. 按一下**載入資料**。資料作為內嵌載入來載入。

使用修飾詞建立集合運算式

集合修飾詞的組成為一個或多個欄位名稱、各個欄位的後面都加上要對該欄位做出的選項。修飾詞括在角括號內。例如，在此集合運算式中：

```
Sum ( {<Year = {2015}>} Sales )
```

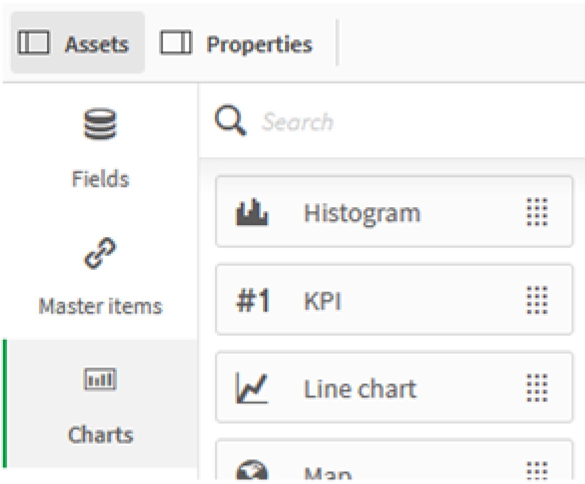
修飾詞為：

```
<Year = {2015}>
```

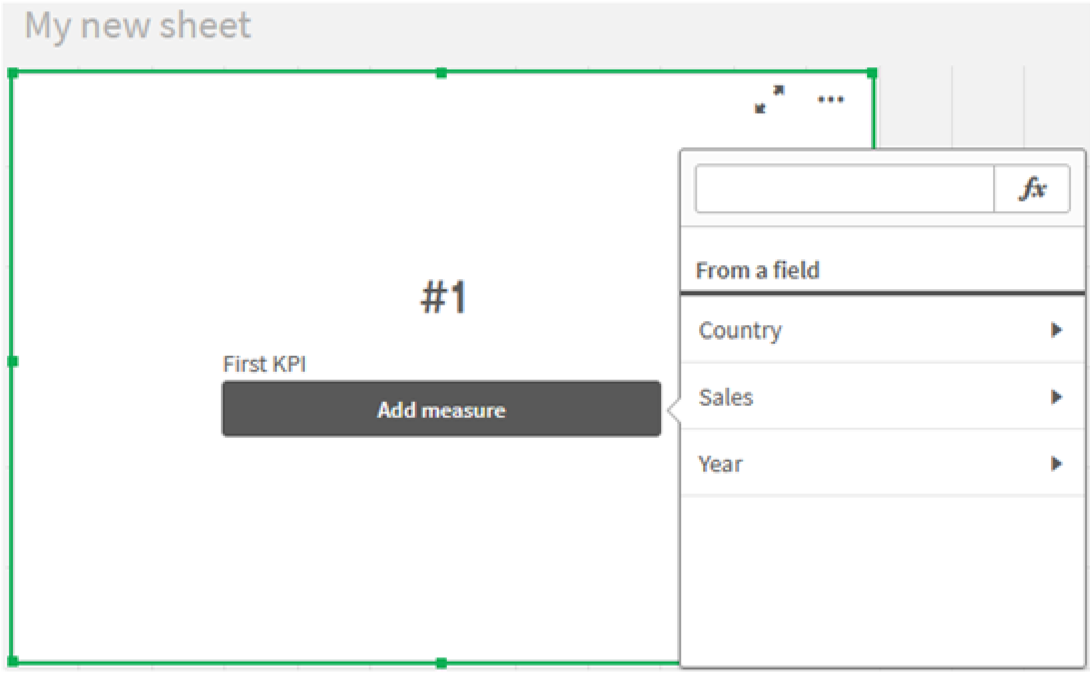
此修飾詞指定將會選取 2015 年起的資料。括住修飾詞的大括弧指示集合運算式。

請執行下列動作：

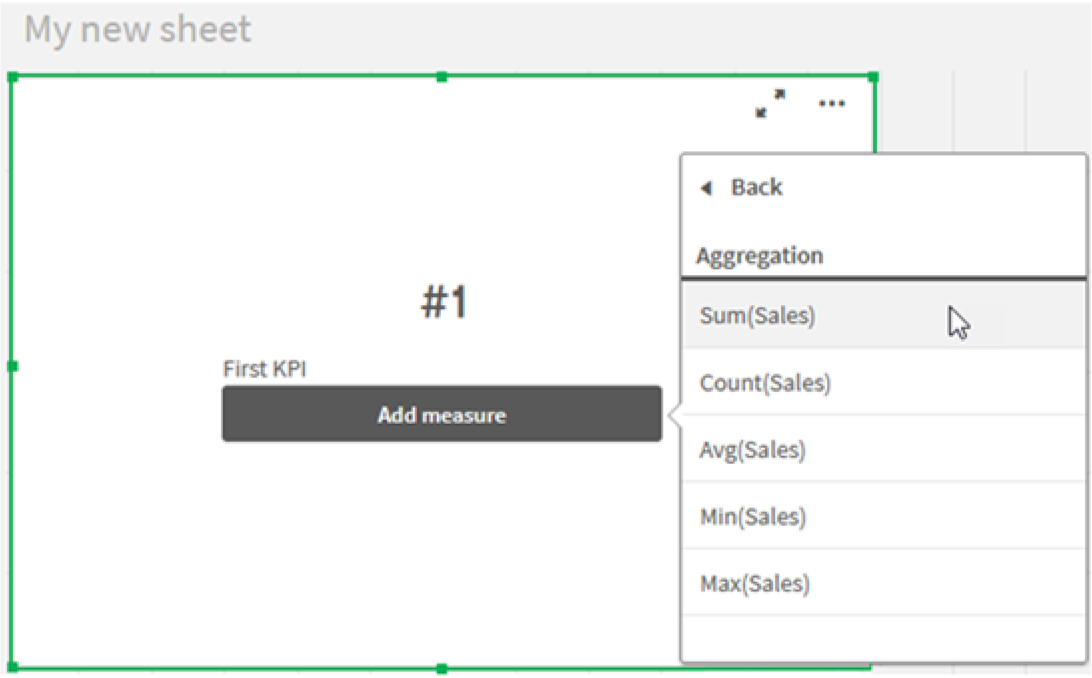
1. 在工作表中，從導覽列開啟**資產**面板，然後按一下**圖表**。



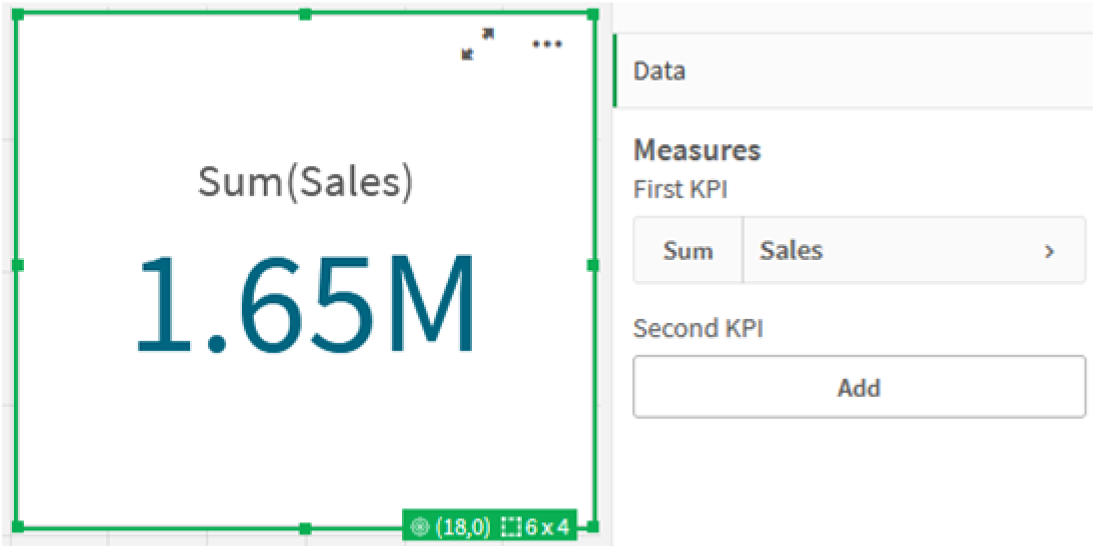
2. 將 **KPI** 拖曳到工作表，然後按一下**新增量值**。



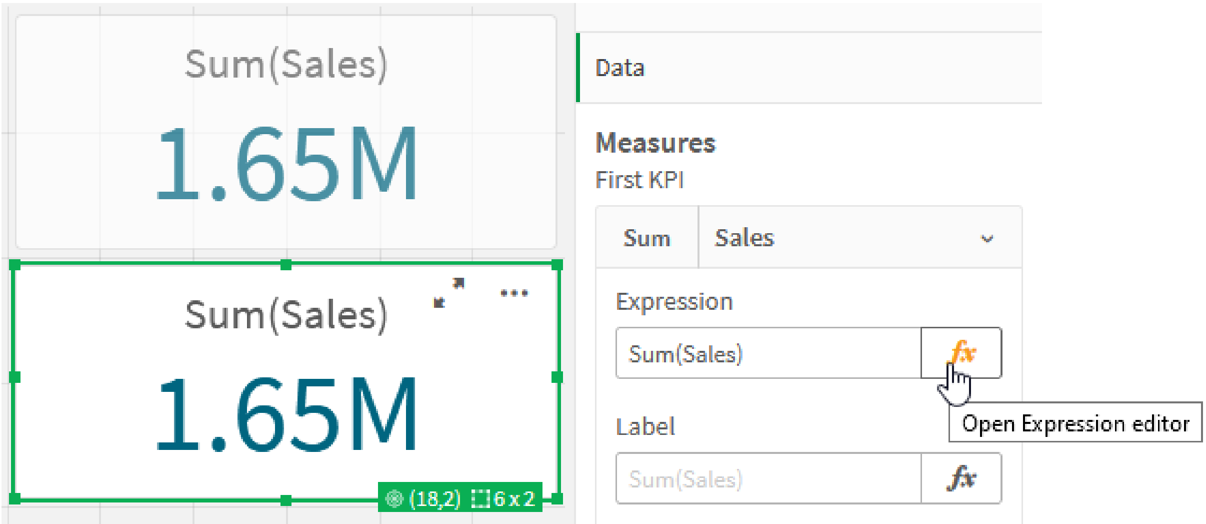
3. 按一下 `Sales`，然後為彙總選取 `Sum(Sales)`。



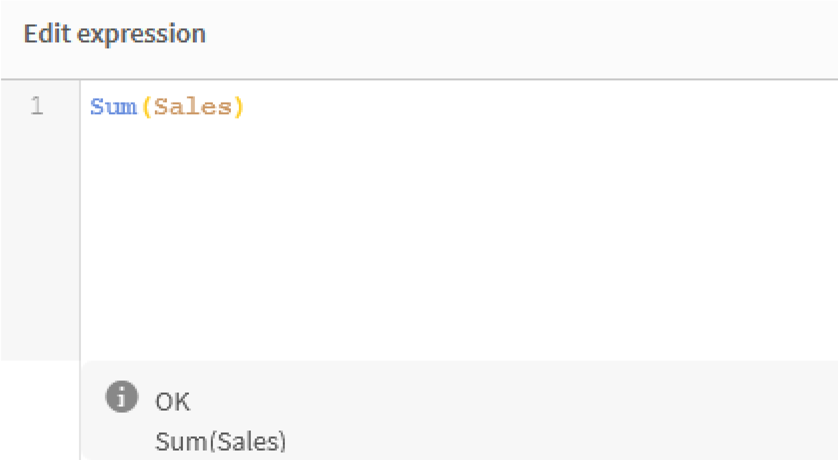
KPI 顯示所有年份的銷售總和。



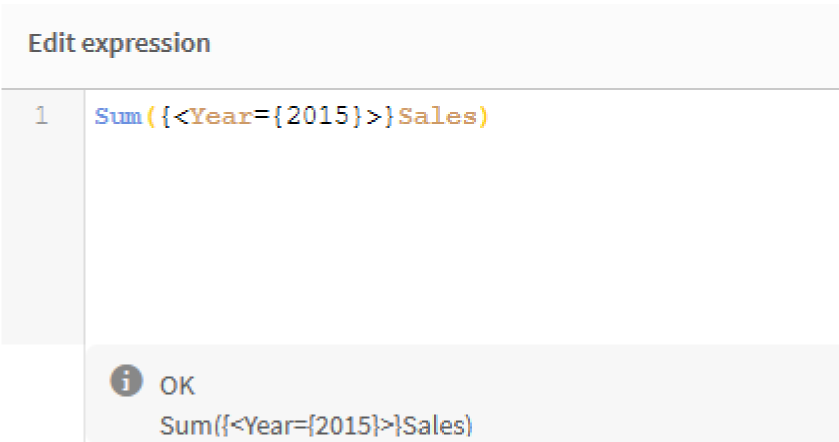
4. 複製並貼上 KPI 以建立新的 KPI。
5. 按一下新的 KPI，按一下**量值**之下的**銷售**，然後按一下**開啟運算式編輯器**。



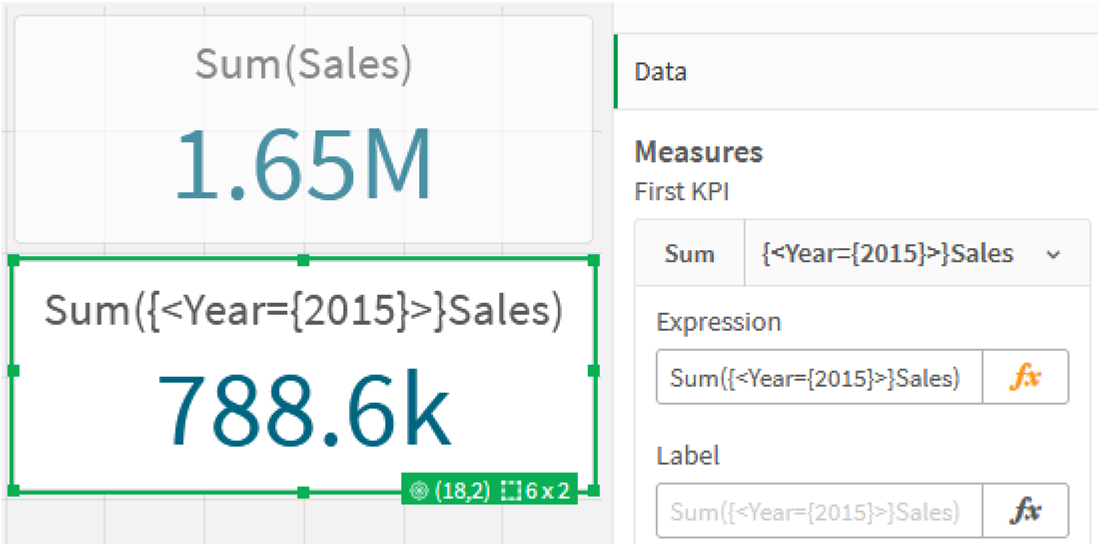
運算式編輯器會以彙總 `sum(Sales)` 開啟。



6. 在運算式編輯器中，建立運算式，以僅加總 2015 年 Sales：
   i. 新增大括號以指示集合運算式：`Sum({}Sales)`
   i. 新增角括號以指示集合修飾詞：`Sum({<>}Sales)`
   ii. 在角括號中，新增要選取的欄位，此在情況下欄位為 `Year`，並在其後加上等號。接著，以另一組大括弧括起 2015。產生的集合修飾詞為：`{<Year={2015}>}`。
   整個運算式是：
   `Sum({<Year={2015}>}Sales)`

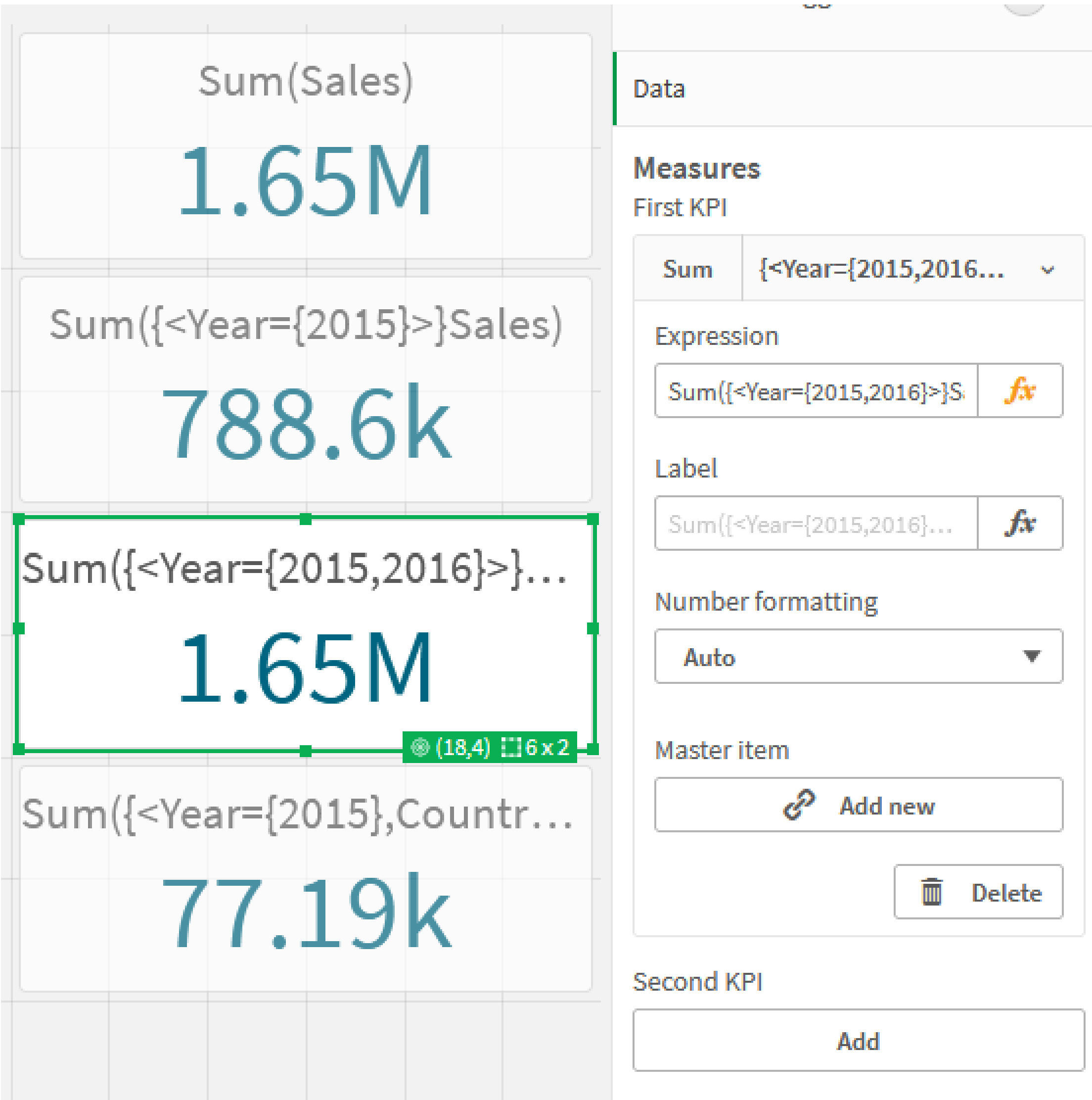


   iii. 按一下**套用**以儲存運算式並關閉運算式編輯器。2015 年 Sales 總和顯示在 KPI 中。



7. 使用下列運算式再建立兩個 KPI:
`Sum({<Year={2015,2016}>}Sales)`
以上的修飾詞為 `<Year={2015,2016}>`。運算式將會傳回 2015 和 2016 年 Sales 的總和。
`Sum({<Year={2015},Country={'Germany'}>} Sales)`
以上的修飾詞為 `<Year={2015}, Country={'Germany'}>`。運算式將會傳回 2015 年 Sales 總和，其中 2015 年與 Germany 有交集。

使用集合修飾詞的 KPI



新增集合識別碼

以上的集合運算式將會使用目前選項作為基礎，因為未使用識別碼。接下來，新增識別碼以在進行選取時指定行為。

請執行下列動作：

在工作表上，建置或複製下列集合運算式：

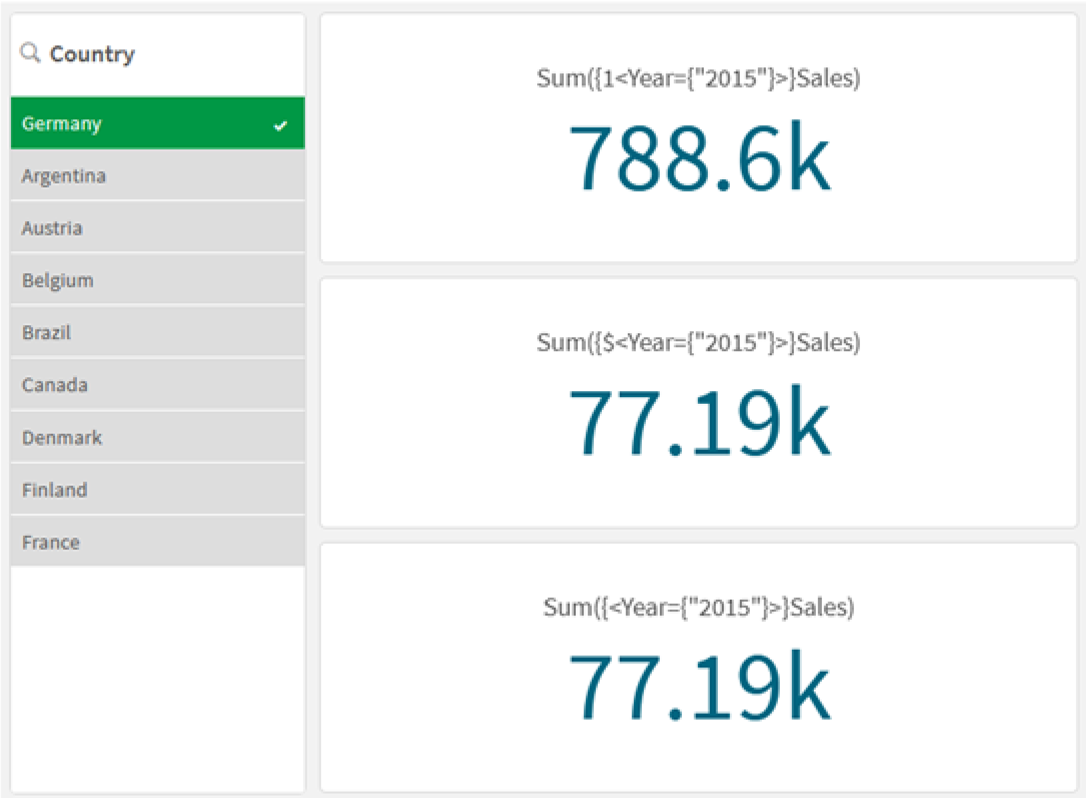
```
Sum({$<Year={"2015"}>}Sales)
```

$ 識別碼將會讓集合運算式立基於資料中選取的目前選項。這也是不使用識別碼時的預設行為。

```
Sum({1<Year={"2015"}>}Sales)
```

1 識別碼將會導致 2015 年的 `Sum(Sales)` 彙總，以忽略目前選項。使用者進行其他選取時，彙總值將不會變更。例如，若在以下選取 Germany，2015 年的彙總總和值不會變更。

使用集合修飾詞和識別碼的 KPI



新增運算子

集合運算子用來納入、排除或交集資料集。所有運算子都將集合作為運算元使用，並傳回一個集合作為結果。

您可以在兩種不同的情況下使用集合運算子：

- 在集合識別碼執行集合操作，表示資料中的記錄集合。
- 在元素集、欄位值或集合修飾詞內部執行集合操作。

請執行下列動作：

在工作表上，建置或複製下列集合運算式：

```
Sum({$<Year={2015}>+1<Country={'Germany'}>}Sales)
```

加號 `(+)` 運算子會為 2015 和 Germany 產生資料集的聯集。如以上集合識別碼的說明，貨幣符號 `($)` 識別碼代表將會採用第一個運算元 `<Year={2015}>` 使用的目前選項。`1` 識別碼代表將會忽略第二個運算元 `<Country={'Germany'}>` 的選項。

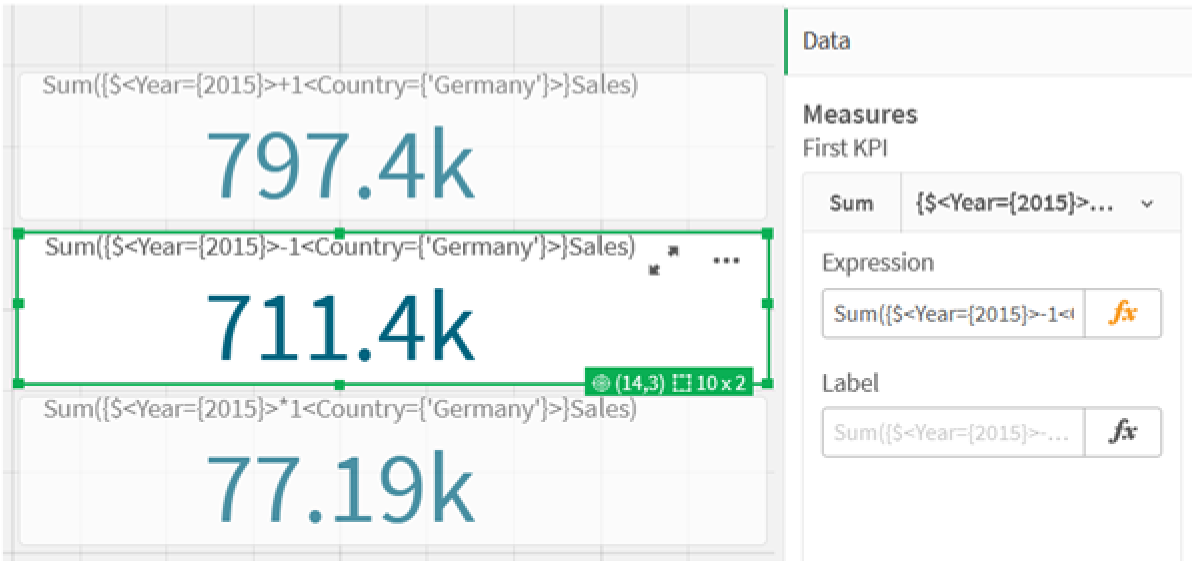
使用加號 (+) 運算子的 KPI

或者，使用減號 (-) 以傳回資料集，該資料集包括屬於 2015、但不屬於 Germany 的記錄。或者，使用星號 (*) 傳回包括屬於兩個集合之記錄的集合。

```
Sum({$<Year={2015}>-1<Country={'Germany'}>}Sales)
```

```
Sum({$<Year={2015}>*1<Country={'Germany'}>}Sales)
```

使用運算子的 KPI



集合運算式教學課程資料

載入指令碼

載入下列資料作為內嵌載入，然後在教學課程中建立圖表運算式。

```
//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
];
```

集合運算式的語法

使用 Backus-Naur 形式論說明完整語法 (不包含選用標準括弧以定義優先順序):

```
set_expression ::= { set_entity { set_operator set_entity } }
set_entity ::= set_identifier [ set_modifier ] | set_modifier
set_identifier ::= 1 | $ | $N | $_N | bookmark_id | bookmark_name
set_operator ::= + | - | * | /
set_modifier ::= < field_selection {, field_selection } >
field_selection ::= field_name [ = | += | –= | *= | /= ] element_set_
expression
element_set_expression ::= [ - ] element_set { set_operator element_set }
element_set ::= [ field_name ] | { element_list } | element_function
element_list ::= element { , element }
element_function ::= ( P | E ) ( [set_expression] [field_name] )
element ::= field_value | " search_mask "
```

6.3 圖表運算式一般語法

下列一般語法結構可用於圖表運算式, 並透過許多選用參數:

```
expression ::= ( constant | expressionname | operator1 expression | expression operator2
expression | function | aggregation function | (expression ) )
```

其中:

constant 為以一般單引號括住的字串 (文字、日期或時間), 或為數字。常數的寫法不包含千位分隔符號, 但會加上小數點作為小數分隔符號。

expressionname 是同一個圖表中另一個運算式的名稱 (標籤)。

operator1 為一元運算子 (會影響右側的一個運算式)。

operator2 為二元運算子 (會影響兩側各一個運算式)。

```
function ::= functionname ( parameters )
parameters ::= expression { , expression }
```

參數的數目和類型非為任意的, 會視所用的函數而定。

```
aggregationfunction ::= aggregationfunctionname ( parameters2 )
parameters2 ::= aggrexpression { , aggrexpression }
```

參數的數目和類型非為任意的，會視所用的函數而定。

6.4 彙總的一般語法：

下列一般語法結構可用於彙總，並透過許多選用參數：

```
aggrexpression ::= ( fieldref | operator1 aggrexpression | aggrexpression operator2
aggrexpression | functioninaggr | ( aggrexpression ) )
```

fieldref 是欄位名稱。

```
functionaggr ::= functionname ( parameters2 )
```

因此，只要 **fieldref** 始終恰好以一個彙總函數括住，就可以任意將運算式和函數設為巢狀；而且，如果運算式傳回無法解讀的值，Qlik Sense 就不會產生任何錯誤訊息。

7 運算子

本節描述可以在 Qlik Sense 中使用的運算子。有兩種類型的運算子：

- 一元運算子 (僅採用一個運算元)
- 二元運算子 (採用兩個運算元)

大部分運算子都是二元。

可定義下列運算子:

- 位元運算子
- 邏輯運算子
- 數值運算子
- 關聯運算子
- 字串運算子

7.1 位元運算子

所有位元運算子都會將運算元轉換 (截斷) 為帶正負號的整數 (32 位元), 並且以相同的方式傳回結果。所有運算都是依每個位元逐一執行的。如果運算元無法解譯為數字, 則該運算將傳回 NULL。

位元運算子

運算子	全名	描述
bitnot	位元反向。	一元運算子。該運算會傳回依每個位元逐一執行的運算元邏輯反向。 **範例:** `bitnot 17` 傳回 -18
bitand	位元 and。	該運算會傳回依每個位元逐一執行的運算元邏輯 AND。 **範例:** `17 bitand 7` 傳回 1
bitor	位元 or。	該運算會傳回依每個位元逐一執行的運算元邏輯 OR。 **範例:** `17 bitor 7` 傳回 23

運算子	全名	描述
bitxor	位元 exclusive or。	該運算會傳回依每個位元逐一執行的運算元邏輯 exclusive or。 **範例:** `17 bitxor 7` 傳回 22
>>	位元向右偏移。	該運算會傳回第一個向右偏移的運算元。在第二個運算元中定義步驟數目。 **範例:** `8 >> 2` 傳回 2
<<	位元向左偏移。	該運算會傳回第一個向左偏移的運算元。在第二個運算元中定義步驟數目。 **範例:** `8 << 2` 傳回 32

7.2 邏輯運算子

所有邏輯運算子都會以邏輯方式解譯運算元，並傳回 True (-1) 或 False (0) 做為結果。

邏輯運算子

運算子	描述
not	邏輯反向。幾個一元運算子的其中一個。該運算會傳回運算元的邏輯反向。
and	邏輯 and。該運算會傳回運算元的邏輯 and。
or	邏輯 or。該運算會傳回運算元的邏輯 or。
Xor	邏輯 exclusive or。該運算會傳回運算元的邏輯 exclusive or。也就是和邏輯 or 一樣，差別在於如果兩個運算元為 True 時，結果則為 False。

7.3 數值運算子

所有數值運算子均使用運算元的數值，並傳回數值做為結果。

數值運算子

運算子	描述
+	正數 (一元運算子) 或算數加法的符號。二元運算會傳回兩個運算元的加總。

運算子	描述
-	負數 (一元運算子) 或算數減法的符號。一元運算會傳回乘以 -1 的運算元，二元運算會傳回兩個運算相減的差。
*	算數乘法。該運算會傳回兩個運算元的積。
/	算數除法。該運算會傳回兩個運算元的比例。

7.4 關聯運算子

所有關係運算子會比較運算元的值，並傳回 True (-1) 或 False (0) 結果。所有關係運算子都是二進位檔。

關聯運算子

運算子	描述
<	小於。如果這兩個運算元都可解譯成數字，將進行數字比較。這項運算會傳回比較評估的邏輯值。
<=	小於或等於。如果這兩個運算元都可解譯成數字，將進行數字比較。這項運算會傳回比較評估的邏輯值。
>	大於。如果這兩個運算元都可解譯成數字，將進行數字比較。這項運算會傳回比較評估的邏輯值。
>=	大於或等於。如果這兩個運算元都可解譯成數字，將進行數字比較。這項運算會傳回比較評估的邏輯值。
=	等於。如果這兩個運算元都可解譯成數字，將進行數字比較。這項運算會傳回比較評估的邏輯值。
<>	不等於。如果這兩個運算元都可解譯成數字，將進行數字比較。這項運算會傳回比較評估的邏輯值。
precedes	與 **<** 運算子不同，在比較前不會嘗試進行引數值的數字解譯。如果運算子左邊的值擁有文字表示法，且在字串比較中大於右邊值的文字表示法，則運算會傳回 True。 **範例：** `'1 ' precedes ' 2'` 傳回 FALSE `' 1' precedes ' 2'` 傳回 TRUE 因為空格 (' ') 的 ASCII 值少於數字的 ASCII 值。 相較於： `'1 ' < ' 2'` 傳回 TRUE `' 1' < ' 2'` 傳回 TRUE

運算子	描述
follows	與 **>** 運算子不同，在比較前不會嘗試進行引數值的數字解譯。如果運算子左邊的值擁有文字表示法，且在字串比較中小於右邊值的文字表示法，則運算會傳回 True。 **範例：** `' 2' follows '1'` 傳回 FALSE ' 2' follows ' 1' 傳回 TRUE 因為空格 (' ') 的 ASCII 值少於數字的 ASCII 值。 相較於： `' 2' > ' 1'` 傳回 TRUE `' 2' > '1 '` 傳回 TRUE

7.5 字串運算子

共有兩個字串運算子。一個會使用運算元的字串值並傳回字串做為結果。另一個會比較運算元並傳回布林值來表示相符。

&

字串串連。這項操作會傳回文字字串，該文字字串會由兩個緊接著的運算元字串組成。

範例：

`'abc' & 'xyz'` 傳回 'abcxyz'

like

與萬用字元進行字串比較。如果運算子前的字串符合運算子後的字串，操作會傳回傳回布林值 True (-1)。第二個字串可能會包含萬用字元 * (任何數量的任意字元) 或？ (一個任意字元)。

範例：

`'abc' like 'a*'` 傳回 True (-1)

`'abcd' like 'a?c*'` 傳回 True (-1)

`'abc' like 'a??bc'` 傳回 False (0)

8 指令碼和圖表函數

在資料載入指令碼和圖表運算式中使用函數轉換並彙總資料。

許多函數在資料載入指令碼和圖表運算式可以相同的方式使用，但是有許多例外：

- 部分函數只能在資料載入指令碼中使用，由 - script 函數表示。
- 部分函數只能在圖表運算式中使用，由 - chart 函數表示。
- 部分函數可以同時在資料載入指令碼和圖表運算式中使用，但是在參數和應用程式中有差異。它們將在單獨的主題中說明，由 - script 函數或 - chart 函數表示。

8.1 伺服器方延伸的分析連線 (SSE)

只有在已設定分析連線且 Qlik Sense 已啟動的情況下，才能看見分析連線啟用的函數。

您可在 QMC 中設定分析連線，請參閱指南 管理 Qlik Sense 網站 中的「建立分析連線」主題。

在 Qlik Sense Desktop 中，您可以編輯 *Settings.ini* 檔案以設定分析連線，請參閱指南 Qlik Sense Desktop 中的「在 Qlik Sense Desktop 中設定分析連線」主題。

8.2 彙總函數

稱為彙總函數的函數系列，包含將多個欄位值作為其輸入值並傳回每個群組單一結果的函數，其中分組是由圖表維度或指令碼陳述式中的 **group by** 子句所定義。

彙總函數包括 **Sum()**、**Count()**、**Min()**、**Max()** 等。

大多數彙總函數可以同時用在資料載入指令碼和圖表運算式中，但是語法有所不同。

限制：

除非內部彙總包含 **TOTAL** 限定詞，否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總，請結合使用進階函數 **Aggr** 與指定維度。

命名實體時，避免將相同名稱指派至多個欄位、變數或量值。解決名稱相同的實體之間的衝突時有嚴格的優先順序。此順序反映在使用這些實體的任何物件或內容中。此優先順序如下：

- 在彙總內部，欄位優先於變數。量值標籤在彙總中不相關，沒有優先度。
- 在彙總外部，量值標籤優先於變數，結果會優先於欄位名稱。
- 此外，在彙總外部，可以參考其標籤以重新使用量值，除非該標籤實際上是計算出的標籤。在該情況下，量值的重要性會下降，以降低自行參考的風險，在此情況下，首先會一律將名稱解譯為量值標籤，第二會解譯為欄位名稱，第三會解譯為變數名稱。

在資料載入指令碼中使用彙總函數

彙總函數只能在 **LOAD** 和 **SELECT** 陳述式內部使用。

在圖表運算式中使用彙總函數

除非內部彙總包含 **TOTAL** 限定詞，否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總，請結合使用進階函數 **Aggr** 與指定維度。

彙總函數會將選項所定義之一組可能記錄加以彙總。然而，可以在集合分析中使用集合運算式，來定義替代的一組記錄。

如何計算彙總

彙總會透過特定表格的記錄來循環，彙總其中的記錄。例如，**Count**(<Field>) 將會計算 <Field> 所在表格中的記錄數量。若您只想要彙總相異欄位值，您需要使用 **distinct** 子句，例如 **Count**(**distinct** <Field>)。

若彙總函數包含來自不同表格的欄位，彙總函數將會透過構成欄位的表格交叉乘積的記錄來循環。這對效能有負面影響，因此應避免這類彙總，尤其是在您有大量資料時應避免。

索引鍵欄位彙總

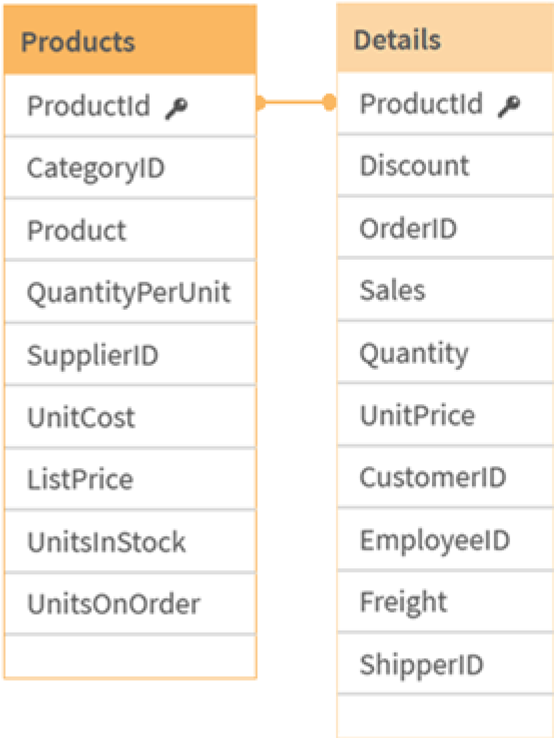
計算彙總的方式代表您無法彙總索引鍵欄位，因為不清楚哪個表格應用於彙總。例如，若欄位 <Key> 連結兩個表格，則不清楚 **Count**(<Key>) 應從第一個或第二個表格傳回記錄數量。

不過，若您使用 **distinct** 子句，可妥善定義並計算彙總。

因此，若您在彙總函數內部使用索引鍵欄位，而沒有 **distinct** 子句，則 Qlik Sense 將會傳回可能無意義的數字。解決方案是使用 **distinct** 子句或使用索引鍵副本，亦即僅位於一個表格中的副本。

例如，在下列表格中，ProductID 是表格之間的金鑰。

產品和詳細資料表格之間的 ProductID 金鑰



Count(ProductID) 可以在 Products 表格 (每個產品只有一個記錄 – ProductID 是主要金鑰) 中計算，或者可以在 Details 表格 (每個產品很可能有數個記錄) 中計算。若您想要計算相異產品的數量，應使用 Count(distinct ProductID)。若您想要計算特定表格的列數，不應使用金鑰。

基本彙總函數

基本彙總函數概述

基本彙總函數是一組最常見的彙總函數。

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

資料載入指令碼中的基本彙總函數

FirstSortedValue

FirstSortedValue() 從運算式中傳回 **value** 中指定的值，該值對應於排序 **sort_weight** 引數的結果，例如，具有最低單價之產品的名稱。可以在 **rank** 中指定排序順序中的第 n 個值。如果對於指定的 **rank**，有多個產生的值共用同一 **sort_weight**，則該函數會傳回 NULL。已排序的值會在許多記錄上反覆運算，如 **group by** 子句所定義，或者如果未定義任何 **group by** 子句則在整個資料集上彙總。

```
FirstSortedValue ([ distinct ] expression, sort_weight [, rank ])
```

Max

Max() 會尋找運算式中已彙總資料的最高數值，如 **group by** 子句所定義。透過指定 **rank** n，可以找到第 n 個最高的值。

```
Max ( expression[, rank])
```

Min

Min() 會傳回運算式中已彙總資料的最低數值，如 **group by** 子句所定義。透過指定 **rank** n，可以找到第 n 個最低的值。

```
Min ( expression[, rank])
```

Mode

Mode() 會傳回運算式中已彙總資料的最常出現值 (模式值)，如 **group by** 子句所定義。**Mode()** 函數可傳回數值和文字值。

```
Mode (expression )
```

Only

如果已彙總的資料中有一個可能的結果並且只有一個，則 **Only()** 會傳回一個值。如果記錄僅包含一個值，則傳回該值，否則傳回 NULL。使用 **group by** 子句，可在多筆記錄上評估。**Only()** 函數可傳回數值和文字值。

```
Only (expression )
```

Sum

Sum() 會計算運算式中已彙總值的總計，如 **group by** 子句所定義。

```
Sum ([distinct]expression)
```

圖表運算式中的基本彙總函數

圖表彙總函數只能用於圖表運算式中的欄位。一個彙總函數的引數運算式不能包含另一個彙總函數。

FirstSortedValue

FirstSortedValue() 從運算式中傳回 **value** 中指定的值，該值對應於排序 **sort_weight** 引數的結果，例如，具有最低單價之產品的名稱。可以在 **rank** 中指定排序順序中的第 n 個值。如果對於指定的 **rank**，有多個產生的值共用同一 **sort_weight**，則該函數會傳回 NULL。

```
FirstSortedValue - 圖表函數([{SetExpression}] [DISTINCT] [TOTAL [<fld
{,fld}>]] value, sort_weight [,rank])
```

Max

Max() 會找到已彙總之資料中的最高值。透過指定 **rank** n，可以找到第 n 個最高的值。

Max - 圖表函數Max() 會找到已彙總之資料中的最高值。透過指定 rank n，可以找到第 n 個最高的值。 您可能也想瞭解 FirstSortedValue 和 rangemax，它們具有類似 Max 函數的功能。 Max([{SetExpression}] [TOTAL [<fld {,fld}>]] expr [,rank]) 數值 引數引數描述 expr包含待測量之資料的運算式或欄位。rankrank 預設值為 1，這對應於最大值。將 rank 指定為 2，將傳回第二個最大值。如果 rank 為 3，將傳回第三個最大值，依此類推。 SetExpression依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。 TOTAL如果單字 TOTAL 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 TOTAL [<fld {.fld}>] (其中 TOTAL 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。 資料CustomerProductUnitSalesUnitPrice

```
AstridaAA416AstridaAA1015AstridaBB99BetacabBB510BetacabCC220BetacabDD-25CanutilityAA815CanutilityCC-19範例與結果範例結果Max(UnitSales)10,因為這是 UnitSales 中的最高值。訂單值的計算方式為售出單位數 (UnitSales) 乘以單位價格。Max (UnitSales*UnitPrice)150,因為計算所有 (UnitSales)*(UnitPrice) 可能值結果的最高值。Max(UnitSales, 2)9,即為第二高值。Max(TOTAL UnitSales)10,因為 TOTAL 限定詞表示找到最高的可能值,忽略圖表維度。針對具有 Customer 作為維度的圖表,TOTAL 限定詞會確認傳回完整資料集間的最大值,而非各客戶的最大 UnitSales。選取 Customer B。Max({1} TOTAL UnitSales)10,獨立於選取的選項,因為 Set Analysis 運算式 {1} 會定義要評估為 ALL 的記錄組合,無論選取的選項為何。範例中使用的資料:ProductData:LOAD * inline [Customer|Product|UnitSales|UnitPriceAstrida|AA|4|16Astrida|AA|10|15Astrida|BB|9|9Betacab|BB|5|10Betacab|CC|2|20Betacab|DD||25Canutility|AA|8|15Canutility|CC||19] (delimiter is '|'); FirstSortedValue RangeMax ([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Min

Min() 會找到已彙總之資料中的最低值。透過指定 **rank** n,可以找到第 n 個最低的值。

```
Min - 圖表函數([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Mode

Mode()在已彙總之資料中找到最常發生的值,即模式值。**Mode()** 函數可以處理文字值以及數值。

```
Mode - 圖表函數 ({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

Only

如果已彙總的資料中有一個可能的結果並且只有一個,則 **Only()** 會傳回一個值。例如,如果多個產品有單價 9,則僅搜尋單價 = 9 的產品將傳回 NULL。

```
Only - 圖表函數([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr)
```

Sum

Sum() 會計算已彙總的資料中,運算式或欄位提供的值總計。

```
Sum - 圖表函數([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr])
```

FirstSortedValue

FirstSortedValue() 從運算式中傳回 **value** 中指定的值,該值對應於排序 **sort_weight** 引數的結果,例如,具有最低單價之產品的名稱。可以在 **rank** 中指定排序順序中的第 n 個值。如果對於指定的 **rank**,有多個產生的值共用同一 **sort_weight**,則該函數會傳回 NULL。已排序的值會在許多記錄上反覆運算,如 **group by** 子句所定義,或者如果未定義任何 **group by** 子句則在整個資料集上彙總。

語法:

```
FirstSortedValue ([ distinct ] value, sort-weight [, rank ])
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
value Expression	函數找到對應於 **sort_weight** 之排序結果的運算式 **value** 的值。
sort-weight Expression	包含待排序之資料的運算式。會找到 **sort_weight** 的第一個 (最低) 值，從中判定 **value** 運算式的對應值。如果您在 **sort_weight** 前面放置一個負號，則該函數會改為傳回最後一個 (最高) 的已排序值。
rank Expression	透過指明 **rank** "n" 大於 1，可取得第 n 個已排序的值。
distinct	如果 **DISTINCT** 這個字出現在函數引數之前，會忽略評估函數引數所產生的重複項目。

範例與結果：

將範例指令碼新增至您的應用程式並予以執行。然後，至少將結果資料行中列出的欄位新增至我們應用程式中的工作表以查看結果。

若要取得與下方結果資料行中相同的外觀，請在屬性面板的「排序」下方，從「自動」切換至「自訂」，然後取消選取按數字和字母排序。

指令碼處理範例

範例	結果

`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`
	該函數將 UnitSales 從最小到最大排序，尋找具有最小 UnitSales 值、最小順序的 Customer 值。
	
	因為 CC 對應於客戶 Astrida 的最小順序 (UnitSales 的值=2)。AA 對應於客戶 Betacab 的最小順序 (4)，AA 對應於客戶 Canutility 的最小順序 (8)，並且 DD 對應於客戶 Divadip. 的最小順序 (10)。

範例	結果

假設 **Temp** 表格已如前一個範例中所示載入：
`LOAD Customer,FirstSortedValue(Product, -UnitSales) as MyProductWithLargestOrderByCustomer Resident Temp Group By Customer;`
	`Customer`
	`MyProductWithLargestOrderByCustomer`
	`Astrida AA`
	`Betacab DD`
	`Canutility CC`
	`Divadip -`
	減號位於 sort_weight 引數前面，因此函數將最大的排在最前面。
	因為 AA 對應於客戶 Astrida 的最大順序 (UnitSales 的值：18)，DD 對應於客戶 Betacab 的最大順序 (12)，並且 CC 對應於客戶 Canutility 的最大順序 (13)。客戶 Divadip 的最大順序 (16) 具有兩個相同的值，因此這會產生 Null 值結果。

假設 **Temp** 表格已如前一個範例中所示載入：
`LOAD Customer,FirstSortedValue(distinct Product, -UnitSales) as MyProductWithSmallestOrderByCustomer Resident Temp Group By Customer;`
	`Customer`
	`MyProductWithLargestOrderByCustomer`
	`Astrida AA`
	`Betacab DD`
	`Canutility CC`
	`Divadip AA`
	這與先前的範例相同，已使用 distinct 限定詞的除外。這會導致忽略 Divadip 的重複結果，允許傳回非 Null 值。

FirstSortedValue - 圖表函數

FirstSortedValue() 從運算式中傳回 **value** 中指定的值，該值對應於排序 **sort_weight** 引數的結果，例如，具有最低單價之產品的名稱。可以在 **rank** 中指定排序順序中的第 n 個值。如果對於指定的 **rank**，有多個產生的值共用同一 **sort_weight**，則該函數會傳回 NULL。

語法：

```
FirstSortedValue([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] value,
sort_weight [,rank])
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
value	輸出欄位。函數找到對應於 **sort_weight** 之排序結果的運算式 **value** 的值。
sort_weight	輸入欄位。包含待排序之資料的運算式。會找到 **sort_weight** 的第一個 (最低) 值，從中判定 **value** 運算式的對應值。如果您在 **sort_weight** 前面放置一個負號，則該函數會改為傳回最後一個 (最高) 的已排序值。

引數	描述
rank	透過指明 **rank** "n" 大於 1，可取得第 n 個已排序的值。
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前，會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

範例與結果：

資料

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

範例與結果

範例	結果
`firstsortedvalue (Product, UnitPrice)`	BB，即為 `UnitPrice` 最低的 `Product`(9)。
`firstsortedvalue (Product, UnitPrice, 2)`	BB，即為 `UnitPrice` 第二低的 `Product`(10)。
`firstsortedvalue (Customer, -UnitPrice, 2)`	Betacab，即為擁有第二高 `UnitPrice` 的 `Product` 之 `Customer` (20)。
`firstsortedvalue (Customer, UnitPrice, 3)`	NULL，因為有兩個 `Customer` 值 (Astrida 和 Canutility) 具有相同 rank (第三低) `UnitPrice`(15)。 使用 distinct 限定詞以確定不會發生非預期的 NULL 結果。

範例	結果
`firstsortedvalue (Customer, -UnitPrice*UnitSales, 2)`	Canutility，即為擁有第二高銷售訂單值 `UnitPrice` 乘以 `UnitSales` 的 `Customer`(120)。

範例中使用的資料：

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Max

Max() 會尋找運算式中已彙總資料的最高數值，如 **group by** 子句所定義。透過指定 **rank** n，可以找到第 n 個最高的值。

語法：

```
Max ( expr [, rank])
```

傳回的資料類型：數值

引數：

引數

引數	描述
expr Expression	包含待測量之資料的運算式或欄位。
rank Expression	**rank** 預設值為 1，這對應於最大值。將 **rank** 指定為 2，將傳回第二個最大值。如果 **rank** 為 3，將傳回第三個最大值，依此類推。

範例與結果：

將範例指令碼新增至您的應用程式並予以執行。然後，至少將結果資料行中列出的欄位新增至我們應用程式中的工作表以查看結果。

若要取得與下方結果資料行中相同的外觀，請在屬性面板的「排序」下方，從「自動」切換至「自訂」，然後取消選取按數字和字母排序。

範例：

```
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;
```

結果表格

Customer	MyMax
Astrida	18
Betacab	5
Canutility	8

範例:

假設 **Temp** 表格已如前一個範例中所示載入:

```
LOAD Customer, Max(UnitSales,2) as MyMaxRank2 Resident Temp Group By Customer;
```

結果表格

Customer	MyMaxRank2
Astrida	10
Betacab	4
Canutility	-

Max - 圖表函數

Max() 會找到已彙總之資料中的最高值。透過指定 **rank** n, 可以找到第 n 個最高的值。

*您可能也想瞭解 **FirstSortedValue** 和 **rangemax**, 它們具有類似 **Max** 函數的功能。*

語法:

Max([{SetExpression}] [TOTAL [<fld {,fld}>]] expr [,rank]**)**

傳回的資料類型: 數值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
rank	**rank** 預設值為 1, 這對應於最大值。將 **rank** 指定為 2, 將傳回第二個最大值。如果 **rank** 為 3, 將傳回第三個最大值, 依此類推。
SetExpression	依預設, 彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
TOTAL	如果單字 **TOTAL** 位於函數引數之前, 則會在提供現行選項的所有可能值上進行計算, 而不僅僅是關於目前維度值的那些選項, 也就是說, 它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單), 您可建立總可能值的子集。

範例與結果:

資料

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

範例與結果

範例	結果
`Max(UnitSales)`	10, 因為這是 `UnitSales` 中的最高值。

範例	結果
訂單值的計算方式為售出單位數 (UnitSales) 乘以單位價格。 `Max (UnitSales*UnitPrice)`	150, 因為計算所有 (UnitSales)*(UnitPrice) 可能值結果的最高值。
`Max(UnitSales, 2)`	9, 即為第二高值。
`Max(TOTAL UnitSales)`	10, 因為 TOTAL 限定詞表示找到最高的可能值, 忽略圖表維度。針對具有 Customer 作為維度的圖表, TOTAL 限定詞會確認傳回完整資料集間的最大值, 而非各客戶的最大 UnitSales。
選取 Customer B。 `Max({1} TOTAL UnitSales)`	10, 獨立於選取的選項, 因為 Set Analysis 運算式 {1} 會定義要評估為 ALL 的記錄組合, 無論選取的選項為何。

範例中使用的資料:

```
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 '|');
```

另請參見:

- *FirstSortedValue - 圖表函數 (page 313)*
- *RangeMax (page 1268)*

Min

Min() 會傳回運算式中已彙總資料的最低數值, 如 **group by** 子句所定義。透過指定 **rank** n, 可以找到第 n 個最低的值。

語法:

```
Min ( expr [, rank])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
expr Expression	包含待測量之資料的運算式或欄位。
rank Expression	**rank** 預設值為 1,這對應於最小值。將 **rank** 指定為 2,將傳回第二個最小值。如果 **rank** 為 3,將傳回第三個最小值,依此類推。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。然後,至少將結果資料行中列出的欄位新增至我們應用程式中的工作表以查看結果。

若要取得與下方結果資料行中相同的外觀,請在屬性面板的「排序」下方,從「自動」切換至「自訂」,然後取消選取按數字和字母排序。

範例:

```
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;
```

結果表格

Customer	MyMin
Astrida	2
Betacab	4
Canutility	8

範例:

假設 **Temp** 表格已如前一個範例中所示載入:

```
LOAD Customer, Min(UnitSales,2) as MyMinRank2 Resident Temp Group By Customer;
```

結果表格

Customer	MyMinRank2
Astrida	9
Betacab	5
Canutility	-

Min - 圖表函數

Min() 會找到已彙總之資料中的最低值。透過指定 **rank** n，可以找到第 n 個最低的值。

*您可能也想瞭解 **FirstSortedValue** 和 **rangemin**，它們具有類似 **Min** 函數的功能。*

語法：

```
Min({[SetExpression] [TOTAL [<fld {,fld}>]]} expr [,rank])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
rank	**rank** 預設值為 1，這對應於最小值。將 **rank** 指定為 2，將傳回第二個最小值。如果 **rank** 為 3，將傳回第三個最小值，依此類推。
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

範例與結果：

資料

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15
Astrida	BB	9	9

Customer	Product	UnitSales	UnitPrice
Betacab	BB	5	10
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

Min() 函數必須從運算式指定的值陣列傳回非 NULL 值 (若有)。因此在範例中，因為資料中有 NULL 值，所以函數回傳透過運算式評估的第一個非 NULL 值。

範例與結果

範例	結果
`Min(UnitSales)`	2，因為這是 `UnitSales` 中最低的非 NULL 值。
訂單值的計算方式為售出單位數 (`UnitSales`) 乘以單位價格。 `Min (UnitSales*UnitPrice)`	40，因為這是計算所有 (`UnitSales`)*(`UnitPrice`) 可能值的最低非 NULL 值結果。
`Min(UnitSales, 2)`	4，即為第二低值 (次於 NULL 值)。
`Min(TOTAL UnitSales)`	2，因為 TOTAL 限定詞表示找到最低的可能值，忽略圖表維度。針對具有 Customer 作為維度的圖表，TOTAL 限定詞會確認傳回完整資料集間的最小值，而非各客戶的最小 UnitSales。
選取 Customer B。 `Min({1} TOTAL UnitSales)`	2，獨立於 Customer B 的選項。 Set Analysis 運算式 {1} 會定義要評估為 ALL 的記錄組合，無論選取的選項為何。

範例中使用的資料：

```
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 '|');
```

另請參見:

- *FirstSortedValue - 圖表函數 (page 313)*
- *RangeMin (page 1272)*

Mode

Mode() 會傳回運算式中已彙總資料的最常出現值 (模式值), 如 **group by** 子句所定義。**Mode()** 函數可傳回數值和文字值。

語法:

```
Mode ( expr)
```

傳回的資料類型: 雙值

引數

引數	描述
expr Expression	包含待測量之資料的運算式或欄位。

限制:

如果多個值同樣經常出現, 將傳回 NULL。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。然後, 至少將結果資料行中列出的欄位新增至我們應用程式中的工作表以查看結果。

若要取得與下方結果資料行中相同的外觀, 請在屬性面板的「排序」下方, 從「自動」切換至「自訂」, 然後取消選取按數字和字母排序。

指令碼處理範例

<table>
<tr><th>範例</th><th>結果</th></tr>
<tr><td><pre>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;</pre></td><td>MyMostOftenSoldProduct

AA

因為 AA 是唯一一種銷售一次以上的產品。</td></tr>
</table>

Mode - 圖表函數

Mode()在已彙總之資料中找到最常發生的值，即模式值。**Mode()** 函數可以處理文字值以及數值。

語法：

```
Mode({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

範例與結果：

資料

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

範例與結果

範例	結果
`Mode(UnitPrice)` 選取 `Customer A`。	15，因為這是 `UnitSales` 中最常出現的值。 傳回 NULL (-)。沒有任何單一值會比其他值更常出現。
`Mode(Product)` 選取 `Customer A`	AA，因為這是 `Product` 中最常出現的值。 傳回 NULL (-)。沒有任何單一值會比其他值更常出現。
`Mode (TOTAL UnitPrice)`	15，因為 TOTAL 限定詞表示即使忽略圖表維度，最常出現的值仍為 15。
選取 `Customer B`。 `Mode({1} TOTAL UnitPrice)`	15，獨立於選取的選項，因為 Set Analysis 運算式 {1} 會定義要評估為 ALL 的記錄組合，無論選取的選項為何。

範例中使用的資料：

```
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 '|');
```

另請參見：

- *Avg - 圖表函數 (page 378)*
- *Median - 圖表函數 (page 412)*

Only

如果已彙總的資料中有一個可能的結果並且只有一個，則 **Only()** 會傳回一個值。如果記錄僅包含一個值，則傳回該值，否則傳回 NULL。使用 **group by** 子句，可在多筆記錄上評估。**Only()** 函數可傳回數值和文字值。

語法：

```
Only ( expr )
```

傳回的資料類型： 雙值

引數

引數	描述
expr Expression	包含待測量之資料的運算式或欄位。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。然後,至少將結果資料行中列出的欄位新增至我們應用程式中的工作表以查看結果。

若要取得與下方結果資料行中相同的外觀,請在屬性面板的「排序」下方,從「自動」切換至「自訂」,然後取消選取按數字和字母排序。

```
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;
```

結果表格

Customer	MyUniqIDCheck
Astrida	1 因為只有客戶 Astrida 擁有包括 CustomerID 的完整記錄。

Only - 圖表函數

如果已彙總的資料中有一個可能的結果並且只有一個,則 **Only()** 會傳回一個值。例如,如果多個產品有單價 9,則僅搜尋單價 = 9 的產品將傳回 NULL。

語法:

Only([{SetExpression}] [TOTAL [<fld {,fld}>]] expr**)**

傳回的資料類型: 雙值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
SetExpression	依預設,彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。

引數	描述
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

樣本資料中有多個可能值時若想得到 NULL 結果，請使用 Only()。

範例與結果：

資料

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

範例與結果

範例	結果
`Only({<UnitPrice={9}>} Product)`	BB，因為這是唯一 `UnitPrice` 為「9」的 `Product`。
`Only({<Product={DD}>} Customer)`	Betacab，因為這是唯一銷售名為「DD」`Product` 的 `Customer`。
`Only({<UnitPrice={20}>} UnitSales)`	`UnitSales` 的數目，其中 `UnitPrice` 為 20，因為只有一個 `UnitSales` 值，其中 `UnitPrice` =20。
`Only({<UnitPrice={15}>} UnitSales)`	NULL，因為有兩個 `UnitSales` 值，其中 `UnitPrice` =15。

範例中使用的資料：

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
```

```
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Sum

Sum() 會計算運算式中已彙總值的總計，如 **group by** 子句所定義。

語法：

```
sum ( [ distinct] expr)
```

傳回的資料類型： 數值

引數：

引數

引數	描述
distinct	如果 **distinct** 出現在運算式之前，將會忽略所有的重複值。
expr Expression	包含待測量之資料的運算式或欄位。

範例與結果：

將範例指令碼新增至您的應用程式並予以執行。然後，至少將結果資料行中列出的欄位新增至我們應用程式中的工作表以查看結果。

若要取得與下方結果資料行中相同的外觀，請在屬性面板的「排序」下方，從「自動」切換至「自訂」，然後取消選取按數字和字母排序。

```
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;
```

結果表格

Customer	MySum
Astrida	39
Betacab	9
Canutility	8

Sum - 圖表函數

Sum() 會計算已彙總的資料中，運算式或欄位提供的值總計。

語法：

```
Sum([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前，會忽略評估函數引數所產生的重複項目。 *雖然支援 DISTINCT 限定詞，但使用時需非常小心，因它可能會誤導讀取器認為部分資料遭省略時會顯示總值。*
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

範例與結果：

資料

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15

Customer	Product	UnitSales	UnitPrice
Astrida	BB	9	9
Betacab	BB	5	10
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

範例與結果

範例	結果
`Sum(UnitSales)`	38.`UnitSales` 中的總值。
`Sum(UnitSales*UnitPrice)`	505.`UnitPrice` 總值乘以彙總的 `UnitSales`。
`Sum (TOTAL UnitSales*UnitPrice)`	505 適用表格中所有列以及總值，因為 TOTAL 限定詞表示即使忽略圖表維度，加總仍是 505。
選取 `Customer B`。 `Sum({1} TOTAL UnitSales*UnitPrice)`	505，獨立於選取的選項，因為 Set Analysis 運算式 {1} 會定義要評估為 ALL 的記錄組合，無論選取的選項為何。

範例中使用的資料：

```
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 '|');
```

計數器彙總函數

計數器彙總函數針對資料載入指令碼中的大量記錄或者圖表維度中的大量值，傳回各種類型的運算式計數。

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

資料載入指令碼中的計數器彙總函數

Count

Count() 會傳回運算式中已彙總的值數目，如 **group by** 子句所定義。

```
Count ([distinct ] expression | * )
```

MissingCount

MissingCount() 會傳回運算式中已彙總的遺漏值數目，如 **group by** 子句所定義。

```
MissingCount ([ distinct ] expression)
```

NullCount

NullCount() 會傳回運算式中已彙總的 NULL 值數目，如 **group by** 子句所定義。

```
NullCount ([ distinct ] expression)
```

NumericCount

NumericCount() 會傳回運算式中找到的數值數目，如 **group by** 子句所定義。

```
NumericCount ([ distinct ] expression)
```

TextCount

TextCount() 會傳回運算式中已彙總的非數值欄位值數目，如 **group by** 子句所定義。

```
TextCount ([ distinct ] expression)
```

圖表運算式中的計數器彙總函數

下列計數器彙總函數可用於圖表中。

Count

Count() 用來彙總每一個圖表維度中值、文字及數值的數目。

```
Count - 圖表函數({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

MissingCount

MissingCount() 用來彙總每一個圖表維度中遺漏值的數目。遺失值全部都是非數值。

```
MissingCount - 圖表函數({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr)
```

NullCount

NullCount() 用來彙總每一個圖表維度中 NULL 值的數目。

```
NullCount - 圖表函數({[SetExpression][DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

NumericCount

NumericCount() 會彙總每一個圖表維度中數值的數目。

```
NumericCount - 圖表函數({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]}
expr)
```

TextCount

TextCount() 用來彙總每一個圖表維度中非數值的欄位值數目。

TextCount - 圖表函數({[SetExpression] [**DISTINCT**] [**TOTAL** [<fld {,fld}>]]} expr**)**

Count

Count() 會傳回運算式中已彙總的值數目, 如 **group by** 子句所定義。

語法:

```
Count( [distinct ] expr)
```

傳回的資料類型: 整數

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
distinct	若 **distinct** 一字出現在運算式之前, 會忽略所有的重複值。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。然後, 至少將結果資料行中列出的欄位新增至我們應用程式中的工作表以查看結果。

若要取得與下方結果資料行中相同的外觀, 請在屬性面板的「排序」下方, 從「自動」切換至「自訂」, 然後取消選取按數字和字母排序。

指令碼處理範例

範例	結果

`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`
	只要維度 Customer 包括在工作表上的表格中，否則 OrdersByCustomer 的結果就是 3, 2。

假設 **Temp** 表格已如前一個範例中所示載入：
 	
`LOAD Count(OrderNumber) as TotalOrderNumber Resident Temp;`
	`TotalOrderNumber`
	
	`10`

假設 **Temp** 表格已如第一個範例中所示載入：
 	
`LOAD Count(distinct OrderNumber) as TotalOrderNumber Resident Temp;`
	`TotalOrderNumber`
	
	`8`
	因為有兩個 OrderNumber 的值有相同的值 1，所以一個是 Null 值。

Count - 圖表函數

Count() 用來彙總每一個圖表維度中值、文字及數值的數目。

語法：

```
Count({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

傳回的資料類型： 整數

引數：

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。

引數	描述
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前，會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

範例與結果：

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

以下範例假設選取了所有客戶，例外之處會另行說明。

範例與結果

範例	結果
`Count(OrderNumber)`	10，因為有 10 個欄位可能具有 OrderNumber 的值，且所有記錄 (包含空白者) 都納入計數。 *"0" 計為某個值，而不是空的儲存格。然而，如果某個維度的量值彙總為 0，則該維度不會包括在圖表中。*
`Count(Customer)`	10，因為 Count 會評估所有欄位的出現次數。
`Count(DISTINCT [Customer])`	4，因為使用 Distinct 限定詞，Count 只會評估唯一的出現次數。

範例	**結果**
假定已選取客戶 Canutility `Count(OrderNumber)/Count({1} TOTAL OrderNumber)`	0.2，因為運算式會從所選客戶傳回訂單數，作為來自所有客戶的訂單百分比。此情況下為 2 / 10。
假定已選取客戶 Astrida 和 Canutility `Count(TOTAL <Product> OrderNumber)`	5，因為這是僅針對所選客戶產品下訂的訂單數，且空白儲存格納入計數。

範例中使用的資料：

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB|1|25| 25
Canutility|AA|3|8|15
Canutility|CC|||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

MissingCount

MissingCount() 會傳回運算式中已彙總的遺漏值數目，如 **group by** 子句所定義。

語法：

```
MissingCount ( [ distinct ] expr)
```

傳回的資料類型： 整數

引數：

引數

引數	**描述**
expr Expression	包含待測量之資料的運算式或欄位。
distinct	若 **distinct** 一字出現在運算式之前，會忽略所有的重複值。

範例與結果：

將範例指令碼新增至您的應用程式並予以執行。然後，至少將結果資料行中列出的欄位新增至我們應用程式中的工作表以查看結果。

若要取得與下方結果資料行中相同的外觀,請在屬性面板的「排序」下方,從「自動」切換至「自訂」,然後取消選取按數字和字母排序。

指令碼處理範例

範例	結果

`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`
	
	第二個陳述式提供:
	
	`TotalMissingCount`
	
	`3`
	在具有該維度的表格中。

假設 **Temp** 表格已如前一個範例中所示載入:
 	
`LOAD MissingCount(distinct OrderNumber) as TotalMissingCountDistinct Resident Temp;`
	`TotalMissingCountDistinct`
	
	`1`
	因為只有一個 OrderNumber 一個遺漏值。

MissingCount - 圖表函數

MissingCount() 用來彙總每一個圖表維度中遺漏值的數目。遺失值全部都是非數值。

語法:

```
MissingCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

傳回的資料類型: 整數

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
SetExpression	依預設,彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。

引數	描述
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前，會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

範例與結果：

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16
Astrida	AA	7	10	15
Astrida	BB	4	9	9
Betacab	BB	6	5	10
Betacab	CC	5	2	20
Betacab	DD			25
Canutility	AA			15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

範例與結果

範例	結果
`MissingCount([OrderNumber])`	3，因為 10 個 OrderNumber 欄位中有 3 個為空白 *"0" 計為某個值，而不是空的儲存格。然而，如果某個維度的量值彙總為 0，則該維度不會包括在圖表中。*
`MissingCount ([OrderNumber])/MissingCount ({1} Total [OrderNumber])`	運算式會從所選客戶傳回不完整訂單數，作為來自所有客戶的不完整訂單部分。所有客戶總共有 3 個 OrderNumber 遺漏值。因此，針對每個具有 Product 遺漏值的 Customer，結果為 1/3。

範例中使用的資料：

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

NullCount

NullCount() 會傳回運算式中已彙總的 NULL 值數目，如 **group by** 子句所定義。

語法：

```
NullCount ( [ distinct ] expr)
```

傳回的資料類型： 整數

引數：

引數

引數	描述
expr Expression	包含待測量之資料的運算式或欄位。
distinct	若 **distinct** 一字出現在運算式之前，會忽略所有的重複值。

範例與結果：

將範例指令碼新增至您的應用程式並予以執行。然後，至少將結果資料行中列出的欄位新增至我們應用程式中的工作表以查看結果。

若要取得與下方結果資料行中相同的外觀，請在屬性面板的「排序」下方，從「自動」切換至「自訂」，然後取消選取按數字和字母排序。

指令碼處理範例

<table>
<tr><th>範例</th><th>結果</th></tr>
<tr><td><pre>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;</pre></td><td><code>Customer NullOrdersByCustomer</code>

<code>Astrida 0</code>

<code>Betacab 0</code>

<code>Canutility 1</code>

第二個陳述式提供:

<code>TotalNullCount</code>

<code>1</code>
在具有該維度的表格中,因為只有一筆記錄包含 NULL 值。</td></tr>
</table>

NullCount - 圖表函數

NullCount() 用來彙總每一個圖表維度中 NULL 值的數目。

語法:

```
NullCount({[SetExpression][DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

傳回的資料類型: 整數

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
set_ expression	依預設,彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前,會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前,則會在提供現行選項的所有可能值上進行計算,而不僅僅是關於目前維度值的那些選項,也就是說,它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單),您可建立總可能值的子集。

範例與結果:

範例與結果

範例	結果
NullCount ([OrderNumber])	1, 因為我們使用 inline **LOAD** 陳述式中的 NullInterpret 引入一個 NULL 值。

範例中使用的資料:

```
Set NULLINTERPRET = NULL;
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD|||
Canutility|AA|3|8|
Canutility|CC|NULL||
] (delimiter is '|');
Set NULLINTERPRET=;
```

NumericCount

NumericCount() 會傳回運算式中找到的數值數目, 如 **group by** 子句所定義。

語法:

```
NumericCount ( [ distinct ] expr)
```

傳回的資料類型: 整數

引數:

引數

引數	描述
expr Expression	包含待測量之資料的運算式或欄位。
distinct	若 **distinct** 一字出現在運算式之前, 會忽略所有的重複值。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。然後, 至少將結果資料行中列出的欄位新增至我們應用程式中的工作表以查看結果。

若要取得與下方結果資料行中相同的外觀, 請在屬性面板的「排序」下方, 從「自動」切換至「自訂」, 然後取消選取按數字和字母排序。

指令碼處理範例

範例	結果
`LOAD NumericCount(OrderNumber) as TotalNumericCount Resident Temp;`	第二個陳述式提供: TotalNumericCount 7 在具有該維度的表格中。
假設 **Temp** 表格已如前一個範例中所示載入: `LOAD NumericCount(distinct OrderNumber) as TotalNumericCCountDistinct Resident Temp;`	TotalNumericCountDistinct 6 因為有一個 OrderNumber 複製另一個,因此結果是不重複的 6。

範例:

```
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;
```

結果表格

Customer	NumericCountByCustomer
Astrida	3
Betacab	2
Canutility	0
Divadip	2

NumericCount - 圖表函數

NumericCount() 會彙總每一個圖表維度中數值的數目。

語法:

```
NumericCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

傳回的資料類型: 整數

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
set_ expression	依預設,彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前,會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前,則會在提供現行選項的所有可能值上進行計算,而不僅僅是關於目前維度值的那些選項,也就是說,它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單),您可建立總可能值的子集。

範例與結果:

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

Customer	Product	OrderNumber	UnitSales	Unit Price
Betacab	CC	5	2	20
Betacab	DD			25
Canutility	AA			15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

以下範例假設選取了所有客戶，例外之處會另行說明。

範例與結果

範例	結果
`NumericCount ([OrderNumber])`	7，因為 OrderNumber 的 10 個欄位中有 3 個為空白。 *"0" 計為某個值，而不是空的儲存格。然而，如果某個維度的量值彙總為 0，則該維度不會包括在圖表中。*
`NumericCount ([Product])`	0，因為所有產品名稱皆為文字。通常您可用此函數檢查文字欄位中沒有任何數值內容。
`NumericCount (DISTINCT [OrderNumber])/Count (DISTINCT [OrderNumber)]`	計算所有相異數值訂單編號的數量，再除以訂單編號數值與非數值的數目。若所有欄位值皆為數值則為 1。通常您可用此函數檢查所有欄位值皆為數值。在範例中，OrderNumber 的 8 個相異數值與非數值中有 7 個相異的數值，因此運算式傳回 0.875。

範例中使用的資料：

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

TextCount

TextCount() 會傳回運算式中已彙總的非數值欄位值數目，如 **group by** 子句所定義。

語法:

```
TextCount ( [ distinct ] expr)
```

傳回的資料類型: 整數

引數:

引數

引數	描述
expr Expression	包含待測量之資料的運算式或欄位。
distinct	若 **distinct** 一字出現在運算式之前, 會忽略所有的重複值。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。然後, 至少將結果資料行中列出的欄位新增至我們應用程式中的工作表以查看結果。

若要取得與下方結果資料行中相同的外觀, 請在屬性面板的「排序」下方, 從「自動」切換至「自訂」, 然後取消選取按數字和字母排序。

範例:

```
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;
```

結果表格

Customer	ProductTextCount
Astrida	3
Betacab	3
Canutility	2
Divadip	2

範例：

```
LOAD Customer,TextCount(OrderNumber) as OrderNumberTextCount Resident Temp Group By Customer;
```

結果表格

Customer	OrderNumberTextCount
Astrida	0
Betacab	1
Canutility	2
Divadip	0

TextCount - 圖表函數

TextCount() 用來彙總每一個圖表維度中非數值的欄位值數目。

語法：

```
TextCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

傳回的資料類型： 整數

引數：

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前，會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

範例與結果：

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	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

範例與結果

範例	結果
`TextCount([Product])`	10，因為 Product 的 10 個欄位全部皆為文字。 *"0" 計為某個值，而不是空的儲存格。然而，如果某個維度的量值彙總為 0，則該維度不會包括在圖表中。空白儲存格會評估為非文字，且不會被 TextCount 納入計數。*
`TextCount ([OrderNumber])`	3，因為空白儲存格納入計數。通常您可用此函數檢查數值欄位中沒有任何文字值或為非零值。
`TextCount (DISTINCT [Product])/Count ([Product)]`	計入 Product (4) 的所有不同文字值的數字，並將它除以 Product (10) 中的值總數。結果為 0.4。

範例中使用的資料：

```
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 '|');
```

財務彙總函數

本節說明有關付款和現金流之財務運作的彙總函數。

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

資料載入指令碼中的財務彙總函數

IRR

IRR() 會針對由 group by 子句所定義之若干記錄上反覆運算的運算式中數字所代表的一組現金流量，傳回彙總的內部報酬率。

```
IRR (expression)
```

XIRR

XIRR() 會針對由 group by 子句所定義之若干記錄上反覆運算的配對數字 **pmt** 和 **date** 所代表的現金流量排程 (不一定是定期)，傳回彙總的內部報酬率 (每年)。所有支出均按照 1 年 365 天攤算。

```
XIRR (valueexpression, dateexpression )
```

NPV

NPV() 指令碼函數採用折扣率和依期間排序的多個值。對於這些計算，流入量 (收入) 為正值，而流出量 (未來付款) 假設為負值。這發生在每個期間結束時。

```
NPV (rate, expression)
```

XNPV

XNPV() 會針對配對數字 (以 **pmt** 和 **date** 表示) 所代表的現金流量排程 (不一定是定期)，傳回彙總淨現值。所有支出均按照 1 年 365 天攤算。

```
XNPV (rate, valueexpression, dateexpression)
```

圖表運算式中的財務彙總函數

這些財務彙總函數可用於圖表中

IRR

IRR() 會針對圖表維度上反覆運算的運算式中的數字 (以 **value** 表示) 所代表的一組現金流量，傳回彙總的內部報酬率。

```
IRR - 圖表函數[TOTAL [<fld {,fld}>]] value)
```

NPV

NPV() 會根據圖表維度上反覆運算的數字 (以 **value** 表示) 代表的各期間的 **discount_rate** 以及一系列未來支出 (負值) 和收入 (正值)，傳回投資的彙總淨現值。支出與收入假設於每個期間的末尾發生。

```
NPV - 圖表函數([TOTAL [<fld {,fld}>]] discount_rate, value)
```

XIRR

XIRR() 會針對圖表維度上反覆運算的運算式的配對數字 (以 **pmt** 和 **date** 表示) 所代表的現金流量排程 (不一定是定期) 傳回彙總的內部報酬率 (每年)。所有支出均按照 1 年 365 天攤算。

```
XIRR - 圖表函數([TOTAL [<fld {,fld}>]] pmt, date)
```

XNPV

XNPV() 會針對圖表維度上反覆運算的運算式的配對數字 (以 **pmt** 和 **date** 表示) 所代表的現金流量排程 (不一定是定期) 傳回彙總淨現值。所有支出均按照 1 年 365 天攤算。

```
XNPV - 圖表函數([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

IRR

IRR() 會針對由 group by 子句所定義之若干記錄上反覆運算的運算式中數字所代表的一組現金流量, 傳回彙總的內部報酬率。

這些現金流量不一定平均, 因為可能是年金的現金流量。不過, 現金流量必須定期產生, 例如每月或每年。內部報酬率是對於定期支出 (負值) 和收入 (正值) 的投資所收到的利率。這個函數需要至少一個正值和一個負值才能計算。

此函數使用 Newton 方法的簡化版本來計算內部報酬率 (IRR)。

語法:

```
IRR(value)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	包含待測量之資料的運算式或欄位。

限制:

文字值、NULL 值和遺漏值將予以忽略。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。若要查看結果, 將結果資料行中列出的欄位新增至您應用程式中的工作表。

範例與結果:

範例與結果

範例	年	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 - 圖表函數

IRR() 會針對圖表維度上反覆運算的運算式中的數字 (以 **value** 表示) 所代表的一組現金流量, 傳回彙總的內部報酬率。

這些現金流量不一定平均, 因為可能是年金的現金流量。不過, 現金流量必須定期產生, 例如每月或每年。內部報酬率是對於定期支出 (負值) 和收入 (正值) 的投資所收到的利率。這個函數需要至少一個正值和一個負值才能計算。

此函數使用 Newton 方法的簡化版本來計算內部報酬率 (IRR)。

語法:

```
IRR([TOTAL [<fld {,fld}>]] value)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	包含待測量之資料的運算式或欄位。
TOTAL	如果單字 **TOTAL** 位於函數引數之前, 則會在提供現行選項的所有可能值上進行計算, 而不僅僅是關於目前維度值的那些選項, 也就是說, 它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單), 您可建立總可能值的子集。

限制:

除非內部彙總包含 **TOTAL** 限定詞, 否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總, 請結合使用進階函數 **Aggr** 與指定維度。

文字值、NULL 值和遺漏值將予以忽略。

範例與結果:

範例與結果

範例	結果
IRR (Payments)	0.1634 假設付款的本質為週期性,例如每月。 *在 XIRR 範例中使用了「日期」欄位,其中只要您提供了付款的日期,付款也可以是非週期性的。*

範例中使用的資料:

```
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 '|');
```

另請參見:

- *XIRR - 圖表函數 (page 360)*
- *Aggr - 圖表函數 (page 509)*

NPV

NPV() 指令碼函數採用折扣率和依期間排序的多個值。對於這些計算,流入量 (收入) 為正值,而流出量 (未來付款) 假設為負值。這發生在每個期間結束時。

淨現值 (或 NPV) 用於計算未來現金流目前的總值。若要計算 NPV,我們必須估計每段期間的未來現金流,並判斷正確的折扣率。**NPV()** 指令碼函數採用折扣率和依期間排序的多個值。對於這些計算,流入量 (收入) 為正值,而流出量 (未來付款) 假設為負值。這發生在每個期間結束時。

語法:

```
NPV(discount_rate, value)
```

傳回的資料類型: 數值。依照預設,結果將會格式化為貨幣。

計算淨現值的公式為:

$$NPV = \sum_{t=1}^{n} \frac{R_t}{(1+i)^t}$$

其中：

- R_t = 單一期間 t 的淨現金流入-流出量
- i = 折扣率或可在替代投資中賺取的收益
- t = 計時器週期數

引數

引數	描述
discount_ rate	**discount_rate** 是套用的折扣百分比。 0.1 的值表示 10% 的折扣率。
value	此欄位保留了多個依期間排序的期間價值。第一個值會假設為期間 1 結束時的現金流，以此類推。

限制：

NPV() 函數具有下列限制：

- 文字值、NULL 值和遺漏值將予以忽略。
- 現金流的值必須為遞增期間的順序。

什麼情況下使用

NPV() 是用於檢查專案收益性並衍生其他量值的財務函數。當現金流可作為原始資料使用時，此函數很實用。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 單筆付款 (指令碼)

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一項專案及其單一期間現金流的資料集，其載入到名為 `CashFlow` 的表格中。
- `CashFlow` 表格的 Resident 載入，其用於針對 `NPV` 表格中的專案計算 NPV 欄位。
- 10% 的硬式編碼折扣率，其用於 NPV 計算。
- `Group By` 陳述式，其用於分組專案的所有付款。

載入指令碼

```
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;
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- `PrjId`
- `NPV`

結果表格

PrjId	NPV
1	$909.09

對於每個期間 10% 折扣率並預計在單一期間結束時收到的 $1000 單筆付款，該 NPV 等於 $1000 除以 (1 + 折扣率)。有效 NPV 等於 $909.09

範例 2 – 多筆付款 (指令碼)

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一項專案及其多期間現金流的資料集，其載入到名為 `CashFlow` 的表格中。
- `CashFlow` 表格的 Resident 載入，其用於針對 `NPV` 表格中的專案計算 NPV 欄位。
- 10% (0.1) 的硬式編碼折扣率用於 NPV 計算。
- `Group By` 陳述式，其用於分組專案的所有付款。

載入指令碼

```
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;
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- `PrjId`
- `NPV`

結果表格

PrjId	NPV
1	$1735.54

對於每個期間 10% 折扣率並預計在兩個期間結束時收到的 $1000 付款，有效 NPV 等於 $1735.54。

範例 3 – 多筆付款 (指令碼)

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 兩項專案的折扣率，其載入到名為 Project 的表格中。
- 每項專案依專案 ID 和期間 ID 劃分的多期間現金流。如果資料未排序，此期間 ID 可用於排序記錄。
- NoConcatenate、Resident 載入及 Left Join 函數的組合建立臨時表格 tmpNPV。該表格將 Project 和 CashFlow 表格的記錄結合成一個扁平表格。此表格會重複每個期間的折扣率。
- tmpNPV 表格的 Resident 載入，其用於針對 NPV 表格的每項專案計算 NPV 欄位。
- 單值折扣率與每項專案相關聯。這會使用 only() 函數擷取並用於每項專案的 NPV 計算。
- Group By 陳述式，其用於依專案 ID 分組每項專案的所有付款。

若要避免任何合成或冗餘資料載入到資料模型，指令碼結束時會捨棄 tmpNPV 表格。

載入指令碼

```
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;
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- `PrjId`
- `NPV`

結果表格

PrjId	NPV
1	$2486.85
2	$2042.12

專案 ID 1 預計在三個期間結束時收到每個期間 10% 折扣率的 $1000 付款。因此，有效 NPV 等於 $2486.85。

專案 ID 2 預計在四個期間中會有折扣率為 15% 的兩筆 $500 付款，以及另外兩筆 $1000 付款。因此，有效 NPV 等於 $2042.12。

範例 4 – 專案收益性範例 (指令碼)

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 兩項專案的折扣率和初始付款 (期間 0)，其載入到名為 `Project` 的表格中。
- 每項專案依專案 ID 和期間 ID 劃分的多期間現金流。如果資料未排序，此期間 ID 可用於排序記錄。
- `NoConcatenate`、Resident 載入及 `Left Join` 函數的組合建立臨時表格 `tmpNPV`。該表格將 `Project` 和 `CashFlow` 表格的記錄結合成一個扁平表格。此表格會重複每個期間的折扣率。
- 與每項專案相關聯的單值折扣率會使用 `only()` 函數擷取並用於每項專案的 NPV 計算。
- tmpNPV 表格的 Resident 載入用於針對 NPV 表格的每項專案計算 NPV 欄位。
- 將 NPV 除以每項專案初始付款的其他欄位已建立並用於計算專案收益性指數。
- 依專案 ID 分組的分組依據陳述式用於分組每項專案的所有付款。

若要避免任何合成或冗餘資料載入到資料模型，指令碼結束時會捨棄 `tmpNPV` 表格。

載入指令碼

```
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;
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- PrjId
- NPV

建立下列量值：

=only(Profitability_Index)

結果表格

PrjId	NPV	=only(Profitability_Index)
1	$87039.82	0.87
2	$123513.71	1.24

專案 ID 1 的有效 NPV 為 $87039.82 而初始付款為 $100000。因此，收益性指數等於 0.87。少於 1 的關係，該專案無法獲利。

專案 ID 2 的有效 NPV 為 $123513.71 而初始投資為 $100000。因此，收益性指數等於 1.24。大於 1 的關係，該專案可以獲利。

NPV - 圖表函數

NPV() 會根據圖表維度上反覆運算的數字 (以 **value** 表示) 代表的各期間的 **discount_rate** 以及一系列未來支出 (負值) 和收入 (正值)，傳回投資的彙總淨現值。支出與收入假設於每個期間的末尾發生。

語法：

```
NPV([TOTAL [<fld {,fld}>]] discount_rate, value)
```

傳回的資料類型： 數值 依照預設，結果將會格式化為貨幣。

引數：

引數

引數	描述
discount_rate	**discount_rate** 是套用的折扣百分比。
value	包含待測量之資料的運算式或欄位。
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。 **TOTAL** 限定詞後面可以加上以角括弧括住的一或多個欄位名稱清單。這些欄位名稱應該是圖表維度變數的子集。在這種情況下，將忽略所列圖表維度變數以外的所有圖表維度變數進行計算，也就是說，將對於所列維度欄位中欄位值的各個組合傳回一個值。另外，清單可以包含目前在圖表中並不是維度的欄位。對於維度欄位不固定的群組維度而言，這會相當實用。列出群組的所有變數會使得函數在向下探查層級變更時產生作用。

限制：

除非內部彙總包含 **TOTAL** 限定詞，否則 **discount_rate** 與 **value** 不可包含彙總函數。如需更進階的巢狀彙總，請結合使用進階函數 **Aggr** 與指定維度。

文字值、NULL 值和遺漏值將予以忽略。

範例與結果:

範例與結果

範例	結果
`NPV(Discount, Payments)`	-$540.12

範例中使用的資料:

```
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 '|');
```

另請參見:

- *XNPV - 圖表函數 (page 368)*
- *Aggr - 圖表函數 (page 509)*

XIRR

XIRR() 會針對由 group by 子句所定義之若干記錄上反覆運算的配對數字 **pmt** 和 **date** 所代表的現金流量排程 (不一定是定期), 傳回彙總的內部報酬率 (每年)。所有支出均按照 1 年 365 天攤算。

Qlik 的 XIRR 功能 (**XIRR()** 和 **RangeXIRR()** 函數) 使用解開 `Rate` 值的下列方程式, 以判定正確的 XIRR 值:

`XNPV(Rate, pmt, date) = 0`

會使用 Newton 方法的簡化版本解開方程式。

語法:

```
XIRR(pmt, date )
```

傳回的資料類型: 數值

引數

引數	描述
pmt	付款。包含對應於 **date** 中提供之付款排程的現金流量的運算式或欄位。
date	包含對應於 **pmt** 中提供之現金流量付款的日期排程的運算式或欄位。

使用此函數時, 適用下列限制:

- 若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值，會導致整個資料配對被忽略。
- 此函數需要至少一個有效負值和至少一個有效正值付款 (連同對應的有效日期)。若未提供這些付款，會傳回 NULL 值。

這些主題可能可協助您使用此函數：

- *XNPV (page 363)*: 使用此函數計算現金流排程的彙總淨現值。
- *RangeXIRR (page 1290)*:**RangeXIRR()** 是 **XIRR()** 函數的同等範圍函數。

在 Qlik Sense 用戶端管理 的不同版本，此函數使用的基礎演算法有變化。如需關於演算法最近更新的資訊，請參閱支援文章 XIRR 函數修正和更新。

範例

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 一系列現金流的交易資料。
- 使用 **XIRR()** 函數運算這些現金流的內部年度報酬率。

載入指令碼

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Payments
2013-01-01|-10000
2013-03-01|3000
2013-10-30|4200
2014-02-01|6800
] (delimiter is '|');

Cashflow1:
LOAD Year,XIRR(Payments, Date) as XIRR2013  Resident Cashflow Group By Year;
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- Year
- XIRR2013

結果表格

年	**XIRR2013**
2013	0.5385

解譯 XIRR 傳回值

XIRR 功能通常用來分析投資，這在開始時有支出 (負) 款項，隨後則有一系列金額較小的收入 (正) 款項。以下是簡化範例，只有一個負值和一個正值款項：

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|-100
2024-01-01|110
] (delimiter is '|');
```

我們的初始付款是 100，並在一年後取回 110。這代表每年 10% 的報酬率。`XIRR(Payments, Date)` 傳回值 0.1。

XIRR 功能的傳回值可以是正值或負值。在投資情況下，負值結果表示投資有損失。透過付款欄位進行總和彙總，即可簡單計算獲得或損失的金額。

在上述範例中，我們將自己的金錢借出一年。報酬率可以想成利息。若您是交易的另一方 (例如，若您是借入者而非借出者)，也可以使用 XIRR 功能。

考慮此範例：

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|100
2024-01-01|-110
] (delimiter is '|');
```

這與第一個範例相同，但情況相反。在此，我們借貸了 100 一年，償還時需附帶 10% 利息。在此範例中，XIRR 計算傳回 0.1 (10%)，與第一個範例的值相同。

請注意，在第一個範例中，我們收到了利潤 10，而在第二個範例中，我們損失了 10，但 XIRR 功能的傳回值在兩個範例中都是正值。這是因為無論您是交易哪一方，XIRR 功能都會計算交易中的隱藏利息。

多個解決方案下的限制

Qlik 的 XIRR 功能透過下列方程式定義，其中會解出 `Rate` 值：

```
XNPV(Rate, pmt, date) = 0
```

有時候此方程式可以有多個解決方案。這稱為「多 IRR 問題」，由於非正常現金流 (也稱為不尋常的現金流) 所造成。下列載入指令碼顯示此範例：

```
Cashflow:
LOAD * inline [
Date|Payments
2021-01-01|-200
2022-01-01|500
```

```
2023-01-01|-250
] (delimiter is '|');
```

在此範例中，有一個負值解決方案和一個正值解決方案 (`Rate` = -0.3 和 `Rate` = 0.8)。**XIRR()** 將會傳回 0.8。

當 Qlik 的 XIRR 功能搜尋解決方案時，這從 `Rate` = 0 開始，並逐步增加比率，直到找到解決方案為止。若有多個正值解決方案，將會傳回第一個出現的內容。若找不到正值解決方案，會將 `Rate` 重設為零並開始以負向搜尋解決方案。

請注意，「正常」現金流保證只有一個解決方案。「正常」現金流表示具有相同符號 (正或負) 的所有款項都在連續的群組中。

另請參見：

- *XNPV (page 363)*
- *RangeXIRR (page 1290)*
- XIRR 函數修正和更新

XIRR - 圖表函數

XIRR() 會針對圖表維度上反覆運算的運算式的配對數字 (以 **pmt** 和 **date** 表示) 所代表的現金流量排程 (不一定是定期) 傳回彙總的內部報酬率 (每年)。所有支出均按照 1 年 365 天攤算。

Qlik 的 XIRR 功能 (**XIRR()** 和 **RangeXIRR()** 函數) 使用解開 `Rate` 值的下列方程式，以判定正確的 XIRR 值：

`XNPV(Rate, pmt, date) = 0`

會使用 Newton 方法的簡化版本解開方程式。

語法：

```
XIRR([TOTAL [<fld {,fld}>]] pmt, date)
```

傳回的資料類型： 數值

引數

引數	描述
pmt	付款。包含對應於 **date** 中提供之付款排程的現金流量的運算式或欄位。
date	包含對應於 **pmt** 中提供之現金流量付款的日期排程的運算式或欄位。
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

使用此函數時，適用下列限制：

- 除非內部彙總包含 **TOTAL**限定詞，否則 **pmt** 與 **date** 不可包含彙總函數。如需更進階的巢狀彙總，請結合使用進階函數 **Aggr** 與指定維度。
- 若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值，會導致整個資料配對被忽略。
- 此函數需要至少一個有效負值和至少一個有效正值付款 (連同對應的有效日期)。若未提供這些付款，會傳回 NULL 值。

這些主題可能可協助您使用此函數：

- *XNPV - 圖表函數 (page 368)*: 使用此函數計算現金流排程的彙總淨現值。
- *RangeXIRR (page 1290)*:**RangeXIRR()** 是 **XIRR()** 函數的同等範圍函數。

在 Qlik Sense 用戶端管理 的不同版本，此函數使用的基礎演算法有變化。如需關於演算法最近更新的資訊，請參閱支援文章 XIRR 函數修正和更新。

範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含現金流交易的資料集。
- 儲存在稱為 `Cashflow` 之表格中的資料。

載入指令碼

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Payments
2013-01-01|-10000
2013-03-01|3000
2013-10-30|4200
2014-02-01|6800
] (delimiter is '|');
```

結果

請執行下列動作：

載入資料並開啟工作表。建立新的表格並新增下列計算作為量值：

```
=XIRR(Payments, Date)
```

結果表格

=XIRR(Payments, Date)
0.5385

解譯 XIRR 傳回值

XIRR 功能通常用來分析投資，這在開始時有支出 (負) 款項，隨後則有一系列金額較小的收入 (正) 款項。以下是簡化範例，只有一個負值和一個正值款項：

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|-100
2024-01-01|110
] (delimiter is '|');
```

我們的初始付款是 100，並在一年後取回 110。這代表每年 10% 的報酬率。`XIRR(Payments, Date)` 傳回值 0.1。

XIRR 功能的傳回值可以是正值或負值。在投資情況下，負值結果表示投資有損失。透過付款欄位進行總和彙總，即可簡單計算獲得或損失的金額。

在上述範例中，我們將自己的金錢借出一年。報酬率可以想成利息。若您是交易的另一方 (例如，若您是借入者而非借出者)，也可以使用 XIRR 功能。

考慮此範例：

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|100
2024-01-01|-110
] (delimiter is '|');
```

這與第一個範例相同，但情況相反。在此，我們借貸了 100 一年，償還時需附帶 10% 利息。在此範例中，XIRR 計算傳回 0.1 (10%)，與第一個範例的值相同。

請注意，在第一個範例中，我們收到了利潤 10，而在第二個範例中，我們損失了 10，但 XIRR 功能的傳回值在兩個範例中都是正值。這是因為無論您是交易哪一方，XIRR 功能都會計算交易中的隱藏利息。

多個解決方案下的限制

Qlik 的 XIRR 功能透過下列方程式定義，其中會解出 `Rate` 值：

```
XNPV(Rate, pmt, date) = 0
```

有時候此方程式可以有多個解決方案。這稱為「多 IRR 問題」，由於非正常現金流 (也稱為不尋常的現金流) 所造成。下列載入指令碼顯示此範例：

```
Cashflow:
LOAD * inline [
Date|Payments
2021-01-01|-200
2022-01-01|500
```

```
2023-01-01|-250
] (delimiter is '|');
```

在此範例中，有一個負值解決方案和一個正值解決方案 (`Rate` = -0.3 和 `Rate` = 0.8)。**XIRR()** 將會傳回 0.8。

當 Qlik 的 XIRR 功能搜尋解決方案時，這從 `Rate` = 0 開始，並逐步增加比率，直到找到解決方案為止。若有多個正值解決方案，將會傳回第一個出現的內容。若找不到正值解決方案，會將 `Rate` 重設為零並開始以負向搜尋解決方案。

請注意，「正常」現金流保證只有一個解決方案。「正常」現金流表示具有相同符號 (正或負) 的所有款項都在連續的群組中。

另請參見：

- *IRR - 圖表函數 (page 348)*
- *Aggr - 圖表函數 (page 509)*
- XIRR 函數修正和更新

XNPV

XNPV() 會針對配對數字 (以 **pmt** 和 **date** 表示) 所代表的現金流量排程 (不一定是定期)，傳回彙總淨現值。所有支出均按照 1 年 365 天攤算。

語法：

```
XNPV(discount_rate, pmt, date)
```

傳回的資料類型： 數值

依照預設，結果將會格式化為貨幣。

XNPV 的計算公式顯示如下：

XNPV 彙總公式

$$XNPV = \sum_{i=1}^{n} \frac{P_i}{(1+rate)^{(di-d1)/365}}$$

其中：

- P_i = 單一期間 i 的淨現金流入-流出量
- d_1 = 第一筆付款日期
- d_i = 第 i 筆付款日期
- rate = 折扣率

淨現值 (或 NPV) 用於計算在指定折扣率下未來現金流目前的總值。若要計算 XNPV，我們需要以對應的日期估算未來現金流。在此之後，對於每筆付款，我們會根據付款日期套用複合折扣率。

透過一系列的付款執行 XNPV 彙總類似於透過這些付款執行總和彙總。差別在於，會根據所選的折扣率 (類似利率) 和付款距離未來多久來修改 (或「折扣」) 每筆金額。使用設定為零的 **discount_rate** 參數執行 XNPV 將會讓 XNPV 等同於總和操作 (在加總之前不會修改付款)。一般而言，設定的 **discount_rate** 越接近零，XNPV 結果就會越類似於總和彙總的結果。

引數

引數	描述
discount_rate	**discount_rate** 是付款應據此折扣的年折扣率。 0.1 的值表示 10% 的折扣率。
pmt	付款。包含對應於 **date** 中提供之付款排程的現金流量的運算式或欄位。正值假設為收入，而負值則假設為支出。 *__XNPV()__ 不會折扣初始現金流，因為這一律發生在開始日期。後續付款均按照 1 年 365 天攤算。這不同於也會攤算第一筆付款的 __NPV ()__。*
date	包含對應於 **pmt** 中提供之現金流量付款的日期排程的運算式或欄位。第一個值作為計算未來現金流偏移的開始日期使用。

使用此函數時，適用下列限制：

- 若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值，會導致整個資料配對被忽略。

什麼情況下使用

- XNPV() 用於計算投資機會淨現值 (NPV) 的財務模型。
- 由於其精確度較高，因此與 NPV 相比偏好為所有財務模型類型採用 XNPV。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 單筆付款 (指令碼)

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一項專案及其一年現金流的資料集,位於名稱為 CashFlow 的表格中。初始計算日期設為 2022 年 7 月 1 日,且淨現金流為 0。一年後會有 $1000 的現金流。
- CashFlow 表格的 Resident 載入,其用於針對名稱為 XNPV 的表格中的專案計算 XNPV 欄位。
- 10% (0.1) 的硬式編碼折扣率用於 XNPV 計算。
- Group By 陳述式用於分組專案的所有付款。

載入指令碼

```
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;
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- PrjId
- XNPV

結果表格

PrjId	XNPV
1	$909.09

根據公式,第一筆記錄的 XNPV 值為 0,而第二筆記錄的 XNPV 值為 $909.09;因此,XNPV 總值為 $909.09。

範例 2 – 多筆付款 (指令碼)

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一項專案及其一年現金流的資料集,位於 CashFlow 的表格中。
- CashFlow 表格的 Resident 載入,其用於針對 XNPV 表格中的專案計算 XNPV 欄位。
- 10% (0.1) 的硬式編碼折扣率用於 XNPV 計算。
- Group By 陳述式用於分組專案的所有付款。

載入指令碼

```
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;
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- PrjId
- XNPV

結果表格

PrjId	XNPV
1	$1322.21

在此範例中,第一年結束時收到 $1,000 的付款,而第二年結束時收到 $500 的付款。在每個期間折扣率為 10% 的情況下,有效 XNPV 等於 $1322.21。

請注意，只有資料第一列應參照計算的基準日期。剩餘列的順序則不重要，因為日期參數用於計算已經過的期間。

範例 3 – 多筆付款和不規則現金流 (指令碼)

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- `Project` 表格中兩項專案的折扣率。
- 每項專案依專案 ID 和日期劃分的多期間現金流。`Dates` 欄位用於計算將折扣率套用至現金流的持續時間。第一筆記錄 (初始現金流和日期) 除外的記錄順序不重要，而且變更後不會影響計算。
- 使用 `NoConcatenate`、Resident 載入及 `Left Join` 函數建立臨時表格 `tmpNPV` 時，該表格會將 `Project` 和 `CashFlow` 表格的記錄結合至一個扁平表格中。此表格會重複每個現金流的折扣率。
- `tmpNPV` 表格的 Resident 載入，其用於針對 `XNPV` 表格中的每項專案計算 `XNPV` 欄位。
- 與每項專案相關聯的單值折扣率會使用 `only()` 函數擷取並用於每項專案的 `XNPV` 計算。
- 依專案 ID 分組的 `Group By` 陳述式用於分組每項專案的所有付款和對應日期。
- 若要避免任何合成或冗餘資料載入到資料模型，指令碼結束時會捨棄 `tmpXNPV` 表格。

載入指令碼

```
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;
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- PrjId
- XNPV

結果表格

PrjId	XNPV
1	$1735.54
2	$278.36

專案 ID 1 在 2021 年 7 月 1 日的初始現金流為 $0。連續兩年結束時預計收到每個期間 10% 折扣率的 $1,000 付款。因此，有效 XNPV 等於 $1735.54。

專案 ID 2 在 2020 年 7 月 1 日的初始流出量為 $1000 (因此為負號)。兩年後，預計有 $500 的付款。三年後，預計有另一筆 $500 的付款。最後在 2024 年 7 月 1 日，預計有 $1,000 的付款。在折扣率為 15% 的情況下，有效 XNPV 等於 $278.36。

另請參見：

- *Drop table (page 142)*
- *group by (page 151)*
- *Join (page 69)*
- *Max (page 315)*
- *NoConcatenate (page 85)*
- *NPV - 圖表函數 (page 356)*
- *Only (page 324)*

XNPV - 圖表函數

XNPV() 會針對圖表維度上反覆運算的運算式的配對數字 (以 **pmt** 和 **date** 表示) 所代表的現金流量排程 (不一定是定期) 傳回彙總淨現值。所有支出均按照 1 年 365 天攤算。

語法：

```
XNPV([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

傳回的資料類型: 數值

> *依照預設, 結果將會格式化為貨幣。*

XNPV 的計算公式顯示如下:

XNPV 彙總公式

$$XNPV = \sum_{i=1}^{n} \frac{P_i}{(1+rate)^{(di-d1)/365}}$$

其中:

- P_i = 單一期間 i 的淨現金流入-流出量
- d_1 = 第一筆付款日期
- d_i = 第 i 筆付款日期
- rate = 折扣率

淨現值 (或 NPV) 用於計算在指定折扣率下未來現金流目前的總值。若要計算 XNPV, 我們需要以對應的日期估算未來現金流。在此之後, 對於每筆付款, 我們會根據付款日期套用複合折扣率。

透過一系列的付款執行 XNPV 彙總類似於透過這些付款執行總和彙總。差別在於, 會根據所選的折扣率 (類似利率) 和付款距離未來多久來修改 (或「折扣」) 每筆金額。使用設定為零的 **discount_rate** 參數執行 XNPV 將會讓 XNPV 等同於總和操作 (在加總之前不會修改付款)。一般而言, 設定的 **discount_rate** 越接近零, XNPV 結果就會越類似於總和彙總的結果。

引數

引數	描述
discount_rate	**discount_rate** 是付款應據此折扣的年折扣率。 0.1 的值表示 10% 的折扣率。
pmt	付款。包含對應於 **date** 中提供之付款排程的現金流量的運算式或欄位。正值假設為收入, 而負值則假設為支出。 *__XNPV()__ 不會折扣初始現金流, 因為這一律發生在開始日期。後續付款均按照 1 年 365 天攤算。這不同於也會攤算第一筆付款的 __NPV()__。*
date	包含對應於 **pmt** 中提供之現金流量付款的日期排程的運算式或欄位。第一個值作為計算未來現金流偏移的開始日期使用。
TOTAL	如果單字 **TOTAL** 位於函數引數之前, 則會在提供現行選項的所有可能值上進行計算, 而不僅僅是關於目前維度值的那些選項, 也就是說, 它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單), 您可建立總可能值的子集。

使用此函數時，適用下列限制：

- 除非內部彙總包含 **TOTAL** 或 **ALL** 限定詞，否則 **discount_rate**、**pmt** 與 **date** 不可包含彙總函數。如需更進階的巢狀彙總，請結合使用進階函數 **Aggr** 與指定維度。
- 若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值，會導致整個資料配對被忽略。

什麼情況下使用

- XNPV() 用於計算投資機會淨現值 (NPV) 的財務模型。
- 由於其精確度較高，因此與 NPV 相比偏好為所有財務模型類型採用 XNPV。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含現金流交易的資料集。
- 儲存在稱為 Cashflow 之表格中的資訊。

載入指令碼

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Payments
2013-01-01|-10000
2013-03-01|3000
2013-10-30|4200
2014-02-01|6800
] (delimiter is '|');
```

結果

請執行下列動作:

載入資料並開啟工作表。建立新的表格並新增下列計算作為量值:

```
=XNPV(0.09, Payments, Date)
```

結果表格

=XNPV(0.09, Payments, Date)
$3062.49

另請參見:

- *NPV - 圖表函數 (page 356)*
- *Aggr - 圖表函數 (page 509)*

統計彙總函數

概述之後,會進一步描述每個函數。您還可以在語法中按一下函數名稱,以立即存取該特定函數的詳細資料。

資料載入指令碼中的統計彙總函數

下列統計彙總函數可用於指令碼中。

Avg

Avg() 會根據 **group by** 子句所定義的若干記錄,來尋找運算式中已彙總之資料的平均值。

```
Avg ([distinct] expression)
```

Correl

Correl() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標,傳回彙總的相關係數。

```
Correl (x-expression, y-expression)
```

Fractile

Fractile() 會根據 **group by** 子句所定義的若干記錄,來尋找相當於運算式中已彙總資料的兼容分位數的值。

```
Fractile (expression, fractile)
```

FractileExc

FractileExc() 會根據 **group by** 子句所定義的若干記錄,來尋找相當於運算式中已彙總資料的互斥分位數的值。

```
FractileExc (expression, fractile)
```

Kurtosis

Kurtosis() 會根據 **group by** 子句所定義的若干記錄，來傳回運算式中資料的 kurtosis。

```
Kurtosis  ([distinct ] expression )
```

LINEST_B

LINEST_B() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 b 值 (y 截距)。

```
LINEST_B (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_df

LINEST_DF() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的自由度。

```
LINEST_DF  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_f

此指令碼函數會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 F 統計 (r^2/($1-r^2$))。

```
LINEST_F (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_m

LINEST_M() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 m 值 (斜率)。

```
LINEST_M  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_r2

LINEST_R2() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 r^2 值 (決定係數)。

```
LINEST_R2  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_seb

LINEST_SEB() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 b 值的標準誤差。

```
LINEST_SEB  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sem

LINEST_SEM() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 m 值的標準誤差。

```
LINEST_SEM (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sey

LINEST_SEY() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 y 估計值的標準誤差。

```
LINEST_SEY (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_ssreg

LINEST_SSREG() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的迴歸平方和。

```
LINEST_SSREG (y-expression, x-expression [, y0 [, x0 ]])
```

Linest_ssresid

LINEST_SSRESID() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的殘差平方和。

```
LINEST_SSRESID (y-expression, x-expression [, y0 [, x0 ]])
```

Median

Median() 會根據 **group by** 子句所定義的若干記錄，來傳回運算式中值的彙總中位數。

```
Median (expression)
```

Skew

Skew() 會針對由 **group by** 子句所定義的若干記錄，傳回運算式的偏態。

```
Skew ([ distinct] expression)
```

Stdev

Stdev() 會根據 **group by** 子句所定義的若干記錄，來傳回運算式中值的標準差。

```
Stdev ([distinct] expression)
```

Sterr

Sterr() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的一組值 (以運算式表示)，傳回彙總的標準誤差 (stdev/sqrt(n))。

```
Sterr ([distinct] expression)
```

STEYX

STEYX() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回迴歸中每個 x 值之預測 y 值的彙總標準誤差。

```
STEYX (y-expression, x-expression)
```

圖表運算式中的統計彙總函數

下列統計彙總函數可用於圖表中：

Avg

Avg() 傳回圖表維度上反覆運算的運算式或欄位的彙總平均值。

```
Avg - 圖表函數({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} expr)
```

Correl

Correl() 傳回兩個資料集的彙總相關係數。相關函數是資料集之間關係的量值，對圖表維度上反覆運算的 (x,y) 值配對進行彙總。

```
Correl - 圖表函數({[SetExpression] [TOTAL [<fld {, fld}>]]} value1, value2 )
```

Fractile

Fractile() 尋找對應於圖表維度上反覆運算之運算式提供的範圍中彙總資料之兼容分位數的值

```
Fractile - 圖表函數({[SetExpression] [TOTAL [<fld {, fld}>]]} expr, fraction)
```

FractileExc

FractileExc() 尋找對應於圖表維度上反覆運算之運算式提供的範圍中彙總資料之互斥分位數的值

```
FractileExc - 圖表函數({[SetExpression] [TOTAL [<fld {, fld}>]]} expr,
fraction)
```

Kurtosis

Kurtosis() 尋找圖表維度上反覆運算的運算式或欄位中彙總資料範圍的 kurtosis。

```
Kurtosis - 圖表函數({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} expr)
```

LINEST_b

LINEST_B() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 b 值 (y 截距)。

```
LINEST_R2 - 圖表函數({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_value, x_value
[, y0_const[, x0_const]])
```

LINEST_df

LINEST_DF() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸的彙總自由度。

```
LINEST_DF - 圖表函數({[SetExpression] [TOTAL [<fld{, fld}>]]} y_value, x_value
[, y0_const [, x0_const]])
```

LINEST_f

LINEST_F() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 F 統計量 (r2/(1-r2))。

```
LINEST_F - 圖表函數({[SetExpression] [TOTAL[<fld{, fld}>]]} y_value, x_value
[, y0_const [, x0_const]])
```

LINEST_m

LINEST_M() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 m 值 (斜率)。

```
LINEST_M - 圖表函數({[SetExpression] [TOTAL[<fld{, fld}>]]} y_value, x_value
[, y0_const [, x0_const]])
```

LINEST_r2

LINEST_R2() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 r2 值 (決定係數)。

```
LINEST_R2 - 圖表函數({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_value, x_value
[, y0_const[, x0_const]])
```

LINEST_seb

LINEST_SEB() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中 b 值的彙總標準誤差。

```
LINEST_SEB - 圖表函數({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_value, x_
value[, y0_const[, x0_const]])
```

LINEST_sem

LINEST_SEM() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中 m 值的彙總標準誤差。

```
LINEST_SEM - 圖表函數([{set_expression}][ distinct ] [total [<fld {,fld}>] ]
y-expression, x-expression [, y0 [, x0 ]] )
```

LINEST_sey

LINEST_SEY() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中 y 估計值的彙總標準誤差。

```
LINEST_SEY - 圖表函數({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_value, x_
value[, y0_const[, x0_const]])
```

LINEST_ssreg

LINEST_SSREG() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的迴歸平方和。

```
LINEST_SSREG - 圖表函數({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_value, x_
value[, y0_const[, x0_const]])
```

LINEST_ssresid

LINEST_SSRESID() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的殘差平方和。

```
LINEST_SSRESID - 圖表函數LINEST_SSRESID() 會針對圖表維度上反覆運算之運算式提供的
配對數字 (以 x_value 和 y_value 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義
之線性迴歸中彙總的殘差平方和。 LINEST_SSRESID([{SetExpression}] [DISTINCT]
[TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])數值 引數引數
```

描述y_value包含待測量之 y 值範圍的運算式或欄位。x_value包含待測量之 x 值範圍的運算式或欄位。y0, x0可指明選用值 y0,強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0,即可強制迴歸線通過單一固定座標。 除非指明 y0 和 x0,否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0,則只要一個資料配對即可計算。 SetExpression依預設,彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。 DISTINCT如果 DISTINCT 這個字出現在函數引數之前,會忽略評估函數引數所產生的重複項目。 TOTAL如果單字 TOTAL 位於函數引數之前,則會在提供現行選項的所有可能值上進行計算,而不僅僅是關於目前維度值的那些選項,也就是說,它會忽略圖表維度。 透過使用 TOTAL [<fld {.fld}>] (其中 TOTAL 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單),您可建立總可能值的子集。可指明選用值 y0,強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0,即可強制迴歸線通過單一固定座標。 除非內部彙總包含 TOTAL 限定詞,否則彙總函數的參數不可包含其他彙總函數。 如需更進階的巢狀彙總,請結合使用進階函數 Aggr 與指定維度。 若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值,會導致整個資料配對被忽略。 An example of how to use linest functionsavg({[SetExpression] [**TOTAL** [<fld{ ,fld}>]] }y_value, x_value[, y0_const[, x0_const]]**)**

Median

Median() 傳回圖表維度上反覆運算的運算式中彙總值範圍的中位值。

```
Median - 圖表函數({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MutualInfo

MutualInfo 計算兩個欄位之間或 **Aggr()** 中的彙總值之間的相互資訊 (MI)。

```
MutualInfo - 圖表函數{[SetExpression] [DISTINCT] [TOTAL target, driver [, datatype [, breakdownbyvalue [, samplesize ]]])
```

Skew

Skew() 傳回圖表維度上反覆運算的運算式或欄位的彙總 skewness。

```
Skew - 圖表函數{[SetExpression] [DISTINCT] [TOTAL [<fld{ ,fld}>]]} expr)
```

Stdev

Stdev() 尋找圖表維度上反覆運算的運算式或欄位中彙總資料範圍的標準差。

```
Stdev - 圖表函數({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} expr)
```

Sterr

Sterr() 尋找圖表維度上反覆運算之運算式中彙總的一系列值的平均值標準誤差 (stdev/sqrt(n)) 的值。

```
Sterr - 圖表函數({[SetExpression] [DISTINCT] [TOTAL[<fld{, fld}>]]} expr)
```

STEYX

STEYX() 會針對運算式提供的配對數字 (以 **y_value** 和 **x_value** 表示) 所代表的一組座標,傳回線性迴歸中為每個 x 值預測 y 值時的彙總標準誤差。

```
STEYX - 圖表函數{[SetExpression] [TOTAL [<fld{, fld}>]]} y_value, x_value)
```

Avg

Avg() 會根據 **group by** 子句所定義的若干記錄，來尋找運算式中已彙總之資料的平均值。

語法：

```
Avg([DISTINCT] expr)
```

傳回的資料類型： 數值

引數：

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
DISTINCT	如果 **distinct** 出現在運算式之前，將會忽略所有的重複值。

範例與結果：

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

結果資料

範例	結果
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 可以透過建立包含以下量值的表格，在工作表中查看此項目。 Sum(Sales)/12

範例	結果

假設 **Temp** 表格已如前一個範例中所示載入：
 	
`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`
	只會對相異值進行計數。總計除以非重複值的數量。

Avg - 圖表函數

Avg() 傳回圖表維度上反覆運算的運算式或欄位的彙總平均值。

語法：

```
Avg([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

傳回的資料類型： 數值

引數：

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前，會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

限制：

除非內部彙總包含 **TOTAL** 限定詞，否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總，請結合使用進階函數 **Aggr** 與指定維度。

範例與結果:

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

函數範例

範例	結果
`Avg(Sales)`	對於包含維度 `Customer` 和量值 `Avg([Sales])` 的表格,若顯示**總計**,則結果為 2566。
`Avg([TOTAL (Sales))`	53.458333 (對於所有 `Customer` 值),因為 TOTAL 限定詞表示維度會遭忽略。
`Avg(DISTINCT (Sales))`	總計 51.862069,因為使用 Distinct 限定詞表示只會評估每個 `Customer` 中的 `Sales` 唯一值。

範例中使用的資料:

```
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 '|');
```

另請參見:

Aggr - 圖表函數 (page 509)

Correl

Correl() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回彙總的相關係數。

語法:

```
Correl(value1, value2)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value1, value2	含有兩個待測量相關係數之樣本集合的運算式或欄位。

限制:

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值，會導致整個資料配對被忽略。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

結果資料

範例	結果
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;	在具有 Correl_Salary 維度的表格中，資料載入指令碼中 Correl() 計算的結果顯示為：0.9270611

Correl - 圖表函數

Correl() 傳回兩個資料集的彙總相關係數。相關函數是資料集之間關係的量值，對圖表維度上反覆運算的 (x,y) 值配對進行彙總。

語法：

```
Correl([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] value1, value2 )
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value1, value2	含有兩個待測量相關係數之樣本集合的運算式或欄位。
SetExpression	依預設, 彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前, 會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前, 則會在提供現行選項的所有可能值上進行計算, 而不僅僅是關於目前維度值的那些選項, 也就是說, 它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單), 您可建立總可能值的子集。

限制:

除非內部彙總包含 **TOTAL** 限定詞, 否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總, 請結合使用進階函數 **Aggr** 與指定維度。

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值, 會導致整個資料配對被忽略。

範例與結果:

函數範例

範例	結果
`Correl(Age, Salary)`	對於包含維度 `Employee name` 和量值 `Correl(Age, Salary)` 的表格, 結果為 0.9270611。只會顯示總計儲存格的結果。
`Correl(TOTAL Age, Salary))`	0.927. 此結果和以下結果會以三位小數顯示以利閱讀。 如果您使用維度 Gender 建立篩選窗格並從中進行選取, 當選取 Female 時, 您會看到結果為 0.951, 當選取 Male 時, 則結果為 0.939。這是因為選項中排除了不屬於 Gender 的其他值的所有結果。
`Correl({1} TOTAL Age, Salary))`	0.927. 獨立於選取的選項。這是因為集合運算式 {1} 忽略了所有選項和維度。
`Correl(TOTAL <Gender> Age, Salary))`	在總計儲存格中為 0.927, 對於 Male 的所有值為 0.939, 對於 Female 的所有值則為 0.951。這與從篩選窗格中基於 Gender 進行選取所得到的結果相對應。

範例中使用的資料:

`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 '|');
```

另請參見:

- *Aggr - 圖表函數 (page 509)*
- *Avg - 圖表函數 (page 378)*
- *RangeCorrel (page 1260)*

Fractile

Fractile() 會根據 **group by** 子句所定義的若干記錄，來尋找相當於運算式中已彙總資料的兼容分位數的值。

> *您可以使用 FractileExc (page 386) 計算互斥分位數*

語法:

```
Fractile(expr, fraction)
```

傳回的資料類型: 數值

該函數會傳回對應至排名的值，如 `rank = fraction * (N-1) + 1` 所定義，其中 `N` 是 `expr` 中值的數量。若 `rank` 是非整數數字，會在兩個最相近的值之間進行插補。

引數：

引數

引數	描述
expr	要在計算分位數時使用的包含資料的運算式或欄位。
fraction	介於 0 與 1 之間的數字對應於要計算的分位數 (以分數表示的分位數)。

範例與結果：

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

結果資料

範例	結果

`Table1:`
`Crosstable (Type, Value)`
`Load recno() as ID, * inline [`
`Observation|Comparison`
`35|2`
`40|27`
`12|38`
`15|31`
`21|1`
`14|19`
`46|1`
`10|34`
`28|3`
`48|1`
`16|2`
`30|3`
`32|2`
`48|1`
`31|2`
`22|1`
`12|3`
`39|29`
`19|37`
`25|2 ] (delimiter is '|');`
 	
`Fractile1:`
`LOAD Type,`
`Fractile(Value,0.75) as MyFractile`
`Resident Table1 Group By Type;`
	在具有 `Type` 和 `MyFractile` 維度的表格中，資料載入指令碼中 Fractile() 計算的結果為：
	
	`Type MyFractile`
	
	`Comparison 27.5`
	
	`Observation 36`

Fractile - 圖表函數

Fractile() 尋找對應於圖表維度上反覆運算之運算式提供的範圍中彙總資料之兼容分位數的值

您可以使用 FractileExc - 圖表函數 (page 388) 計算互斥分位數

語法:

```
Fractile([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr, fraction)
```

傳回的資料類型: 數值

該函數會傳回對應至排名的值,如 `rank = fraction * (N-1) + 1` 所定義,其中 `N` 是 `expr` 中值的數量。若 `rank` 是非整數數字,會在兩個最相近的值之間進行插補。

引數:

引數

引數	描述
expr	要在計算分位數時使用的包含資料的運算式或欄位。
fraction	介於 0 與 1 之間的數字對應於要計算的分位數 (以分數表示的分位數)。
SetExpression	依預設,彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前,會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前,則會在提供現行選項的所有可能值上進行計算,而不僅僅是關於目前維度值的那些選項,也就是說,它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單),您可建立總可能值的子集。

限制:

除非內部彙總包含 **TOTAL** 限定詞,否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總,請結合使用進階函數 **Aggr** 與指定維度。

範例與結果:

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

函數範例

範例	結果
`Fractile (Sales, 0.75)`	對於包含維度 `Customer` 和量值 `Fractile([Sales])` 的表格，若顯示**總計**，則結果為 71.75。這是 `Sales` 值分佈中的一個點，其中 75% 的值都落在這個點以下。
`Fractile(TOTAL Sales, 0.75))`	71.75 (對於所有 `Customer` 值)，因為 TOTAL 限定詞表示維度會遭忽略。
`Fractile (DISTINCT Sales, 0.75)`	總計 70，因為使用 DISTINCT 限定詞表示只會評估每個 `Customer` 中的 `Sales` 唯一值。

範例中使用的資料：

```
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 '|');
```

另請參見：

Aggr - 圖表函數 (page 509)

FractileExc

FractileExc() 會根據 **group by** 子句所定義的若干記錄，來尋找相當於運算式中已彙總資料的互斥分位數的值。

您可以使用 Fractile (page 383) 計算兼容分位數

語法:

```
FractileExc(expr, fraction)
```

傳回的資料類型: 數值

該函數會傳回對應至排名的值,如 `rank = fraction * (N+1)` 所定義,其中 `N` 是 `expr` 中值的數量。若 `rank` 是非整數數字,會在兩個最相近的值之間進行插補。

引數:

引數

引數	描述
expr	要在計算分位數時使用的包含資料的運算式或欄位。
fraction	介於 0 與 1 之間的數字對應於要計算的分位數 (以分數表示的分位數)。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。若要查看結果,將結果資料行中列出的欄位新增至您應用程式中的工作表。

結果資料

<table>
<tr><th>範例</th><th>結果</th></tr>
<tr><td>Table1:
Crosstable (Type, Value)
Load recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2] (delimiter is '|');

Fractile1:
LOAD Type,
FractileExc(Value,0.75) as MyFractile
Resident Table1 Group By Type;</td><td>在具有 Type 和 MyFractile 維度的表格中，資料載入指令碼中 FractileExc() 計算的結果為：

Type MyFractile

Comparison 28.5

Observation 38</td></tr>
</table>

FractileExc - 圖表函數

FractileExc() 尋找對應於圖表維度上反覆運算之運算式提供的範圍中彙總資料之互斥分位數的值

您可以使用 Fractile - 圖表函數 (page 384) 計算兼容分位數

語法：

```
FractileExc([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr,
fraction)
```

傳回的資料類型： 數值

該函數會傳回對應至排名的值，如 `rank = fraction * (N+1)` 所定義，其中 `N` 是 `expr` 中值的數量。若 `rank` 是非整數數字，會在兩個最相近的值之間進行插補。

引數:

引數

引數	描述
expr	要在計算分位數時使用的包含資料的運算式或欄位。
fraction	介於 0 與 1 之間的數字對應於要計算的分位數 (以分數表示的分位數)。
SetExpression	依預設, 彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前, 會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前, 則會在提供現行選項的所有可能值上進行計算, 而不僅僅是關於目前維度值的那些選項, 也就是說, 它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單), 您可建立總可能值的子集。

限制:

除非內部彙總包含 **TOTAL** 限定詞, 否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總, 請結合使用進階函數 **Aggr** 與指定維度。

範例與結果:

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

函數範例

範例	結果
`FractileExc (Sales, 0.75)`	對於包含維度 `Customer` 和量值 `FractileExc([Sales])` 的表格, 若顯示**總計**, 則結果為 75.25。這是 `Sales` 值分佈中的一個點, 其中 75% 的值都落在這個點以下。
`FractileExc (TOTAL Sales, 0.75))`	71.25 (對於所有 `Customer` 值), 因為 TOTAL 限定詞表示維度會遭忽略。
`FractileExc (DISTINCT Sales, 0.75)`	總計 73.50, 因為使用 DISTINCT 限定詞表示只會評估每個 `Customer` 中的 `Sales` 唯一值。

範例中使用的資料:

```
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 '|');
```

另請參見:

Aggr - 圖表函數 (page 509)

Kurtosis

Kurtosis() 會根據 **group by** 子句所定義的若干記錄, 來傳回運算式中資料的 kurtosis。

語法:

```
Kurtosis([distinct ] expr )
```

傳回的資料類型: 數值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
distinct	如果 **distinct** 出現在運算式之前, 將會忽略所有的重複值。

範例與結果：

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

結果資料

<table>
<tr><th>範例</th><th>結果</th></tr>
<tr><td><pre>Table1:
Crosstable (Type, Value)
Load recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2] (delimiter is '|');

Kurtosis1:
LOAD Type,
Kurtosis(Value) as MyKurtosis1,
Kurtosis(DISTINCT Value) as
MyKurtosis2
Resident Table1 Group By Type;</pre></td><td>在具有 Type、MyKurtosis1 和 MyKurtosis2 維度的表格中，資料載入指令碼中 Kurtosis() 計算的結果為：

Type MyKurtosis1 MyKurtosis2

Comparison -1.1612957 -1.4982366

Observation -1.1148768 -0.93540144</td></tr>
</table>

Kurtosis - 圖表函數

Kurtosis() 尋找圖表維度上反覆運算的運算式或欄位中彙總資料範圍的 kurtosis。

語法：

```
Kurtosis([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
SetExpression	依預設, 彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前, 會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前, 則會在提供現行選項的所有可能值上進行計算, 而不僅僅是關於目前維度值的那些選項, 也就是說, 它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單), 您可建立總可能值的子集。

限制:

除非內部彙總包含 **TOTAL** 限定詞, 否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總, 請結合使用進階函數 **Aggr** 與指定維度。

範例與結果:

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

函數範例

範例	結果
`Kurtosis (Value)`	對於包含維度 `Type` 和量值 `Kurtosis(Value)` 的表格, 若顯示表格的**總計**, 且數字格式設定設為 3 個有效數字, 則結果為 1.252。對於 `Comparison` 是 1.161, 對於 `Observation` 則為 1.115。
`Kurtosis (TOTAL Value))`	1.252 (對於所有 `Type` 值), 因為 TOTAL 限定詞表示維度會遭忽略。

範例中使用的資料:

```
Table1:
Crosstable (Type, Value)
Load recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2 ] (delimiter is '|');
```

另請參見:

Avg - 圖表函數 (page 378)

LINEST_B

LINEST_B() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 b 值 (y 截距)。

語法:

```
LINEST_B (y_value, x_value[, y0 [, x0 ]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。

引數	描述
y(0), x (0)	可指明選用值 y0，強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0，即可強制迴歸線通過單一固定座標。 除非指明 y0 和 x0，否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0，則只要一個資料配對即可計算。

限制：

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值，會導致整個資料配對被忽略。

另請參見：

linest 函數使用方式範例 (page 431)

LINEST_B - 圖表函數

LINEST_B() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 b 值 (y 截距)。

語法：

```
LINEST_B([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [ , x0_const]])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。
y0_const, x0_const	可指明選用值 y0，強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0，即可強制迴歸線通過單一固定座標。 *除非指明 y0 和 x0，否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0，則只要一個資料配對即可計算。*
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前，會忽略評估函數引數所產生的重複項目。

引數	描述
TOTAL	如果單字 **TOTAL** 位於函數引數之前, 則會在提供現行選項的所有可能值上進行計算, 而不僅僅是關於目前維度值的那些選項, 也就是說, 它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單), 您可建立總可能值的子集。

限制:

除非內部彙總包含 **TOTAL** 限定詞, 否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總, 請結合使用進階函數 **Aggr** 與指定維度。

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值, 會導致整個資料配對被忽略。

另請參見:

- *linest 函數使用方式範例 (page 431)*
- *Avg - 圖表函數 (page 378)*

LINEST_DF

LINEST_DF() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的自由度。

語法:

```
LINEST_DF (y_value, x_value[, y0 [, x0 ]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。
y(0), x (0)	可指明選用值 y0, 強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0, 即可強制迴歸線通過單一固定座標。 除非指明 y0 和 x0, 否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0, 則只要一個資料配對即可計算。

限制:

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值, 會導致整個資料配對被忽略。

另請參見:

linest 函數使用方式範例 (page 431)

LINEST_DF - 圖表函數

LINEST_DF() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸的彙總自由度。

語法:

```
LINEST_DF([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value [, y0_const [, x0_const]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。
y0, x0	可指明選用值 y0，強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0，即可強制迴歸線通過單一固定座標。 *除非指明 y0 和 x0，否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0，則只要一個資料配對即可計算。*
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前，會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

限制:

除非內部彙總包含 **TOTAL** 限定詞，否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總，請結合使用進階函數 **Aggr** 與指定維度。

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值，會導致整個資料配對被忽略。

另請參見:

- *linest 函數使用方式範例 (page 431)*
- *Avg - 圖表函數 (page 378)*

LINEST_F

此指令碼函數會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 F 統計 ($r^2/(1-r^2)$)。

語法:

```
LINEST_F (y_value, x_value[, y0 [, x0 ]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。
y(0), x(0)	可指明選用值 y0，強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0，即可強制迴歸線通過單一固定座標。 除非指明 y0 和 x0，否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0，則只要一個資料配對即可計算。

限制:

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值，會導致整個資料配對被忽略。

另請參見:

- *linest 函數使用方式範例 (page 431)*

LINEST_F - 圖表函數

LINEST_F() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 F 統計量 (r2/(1-r2))。

語法:

```
LINEST_F([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。
y0, x0	可指明選用值 y0，強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0，即可強制迴歸線通過單一固定座標。 *除非指明 y0 和 x0，否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0，則只要一個資料配對即可計算。*
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前，會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

限制：

除非內部彙總包含 **TOTAL** 限定詞，否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總，請結合使用進階函數 **Aggr** 與指定維度。

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值，會導致整個資料配對被忽略。

另請參見：

- *linest 函數使用方式範例 (page 431)*
- *Avg - 圖表函數 (page 378)*

LINEST_M

LINEST_M() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 m 值 (斜率)。

語法：

```
LINEST_M (y_value, x_value[, y0 [, x0 ]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。
y(0), x (0)	可指明選用值 y0, 強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0, 即可強制迴歸線通過單一固定座標。 除非指明 y0 和 x0, 否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0, 則只要一個資料配對即可計算。

限制:

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值, 會導致整個資料配對被忽略。

另請參見:

linest 函數使用方式範例 (page 431)

LINEST_M - 圖表函數

LINEST_M() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 m 值 (斜率)。

語法:

```
LINEST_M([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。

引數	描述
y0, x0	可指明選用值 y0，強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0，即可強制迴歸線通過單一固定座標。 *除非指明 y0 和 x0，否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0，則只要一個資料配對即可計算。*
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前，會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

限制：

除非內部彙總包含 **TOTAL** 限定詞，否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總，請結合使用進階函數 **Aggr** 與指定維度。

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值，會導致整個資料配對被忽略。

另請參見：

- *linest 函數使用方式範例 (page 431)*
- *Avg - 圖表函數 (page 378)*

LINEST_R2

LINEST_R2() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 r^2 值 (決定係數)。

語法：

```
LINEST_R2 (y_value, x_value[, y0 [, x0 ]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。
y(0), x (0)	可指明選用值 y0, 強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0, 即可強制迴歸線通過單一固定座標。 除非指明 y0 和 x0, 否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0, 則只要一個資料配對即可計算。

限制:

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值, 會導致整個資料配對被忽略。

另請參見:

linest 函數使用方式範例 (page 431)

LINEST_R2 - 圖表函數

LINEST_R2() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 r2 值 (決定係數)。

語法:

```
LINEST_R2([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。

引數	描述
y0, x0	可指明選用值 y0，強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0，即可強制迴歸線通過單一固定座標。 *除非指明 y0 和 x0，否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0，則只要一個資料配對即可計算。*
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前，會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

限制：

除非內部彙總包含 **TOTAL** 限定詞，否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總，請結合使用進階函數 **Aggr** 與指定維度。

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值，會導致整個資料配對被忽略。

另請參見：

- *linest 函數使用方式範例 (page 431)*
- *Avg - 圖表函數 (page 378)*

LINEST_SEB

LINEST_SEB() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 b 值的標準誤差。

語法：

```
LINEST_SEB (y_value, x_value[, y0 [, x0 ]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。
y(0), x (0)	可指明選用值 y0, 強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0, 即可強制迴歸線通過單一固定座標。 除非指明 y0 和 x0, 否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0, 則只要一個資料配對即可計算。

限制:

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值, 會導致整個資料配對被忽略。

另請參見:

linest 函數使用方式範例 (page 431)

LINEST_SEB - 圖表函數

LINEST_SEB() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中 b 值的彙總標準誤差。

語法:

```
LINEST_SEB([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。

引數	描述
y0, x0	可指明選用值 y0，強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0，即可強制迴歸線通過單一固定座標。 *除非指明 y0 和 x0，否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0，則只要一個資料配對即可計算。*
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前，會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

限制：

除非內部彙總包含 **TOTAL** 限定詞，否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總，請結合使用進階函數 **Aggr** 與指定維度。

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值，會導致整個資料配對被忽略。

另請參見：

- *linest 函數使用方式範例 (page 431)*
- *Avg - 圖表函數 (page 378)*

LINEST_SEM

LINEST_SEM() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 m 值的標準誤差。

語法：

```
LINEST_SEM (y_value, x_value[, y0 [, x0 ]])
```

傳回的資料類型： 數值

引數：

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。

引數	描述
x_value	包含待測量之 x 值範圍的運算式或欄位。
y(0), x (0)	可指明選用值 y0，強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0，即可強制迴歸線通過單一固定座標。 除非指明 y0 和 x0，否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0，則只要一個資料配對即可計算。

限制：

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值，會導致整個資料配對被忽略。

另請參見：

linest 函數使用方式範例 (page 431)

LINEST_SEM - 圖表函數

LINEST_SEM() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標，傳回以方程式 y=mx+b 所定義之線性迴歸中 m 值的彙總標準誤差。

語法：

```
LINEST_SEM([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。
y0, x0	可指明選用值 y0，強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0，即可強制迴歸線通過單一固定座標。 *除非指明 y0 和 x0，否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0，則只要一個資料配對即可計算。*
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前，會忽略評估函數引數所產生的重複項目。

引數	描述
TOTAL	如果單字 **TOTAL** 位於函數引數之前, 則會在提供現行選項的所有可能值上進行計算, 而不僅僅是關於目前維度值的那些選項, 也就是說, 它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單), 您可建立總可能值的子集。

限制:

除非內部彙總包含 **TOTAL** 限定詞, 否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總, 請結合使用進階函數 **Aggr** 與指定維度。

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值, 會導致整個資料配對被忽略。

另請參見:

- *linest 函數使用方式範例 (page 431)*
- *Avg - 圖表函數 (page 378)*

LINEST_SEY

LINEST_SEY() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的 y 估計值的標準誤差。

語法:

```
LINEST_SEY (y_value, x_value[, y0 [, x0 ]])
```

傳回的資料類型: 數值

引數:

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。
y(0), x (0)	可指明選用值 y0, 強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0, 即可強制迴歸線通過單一固定座標。 除非指明 y0 和 x0, 否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0, 則只要一個資料配對即可計算。

限制:

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值, 會導致整個資料配對被忽略。

另請參見:

- *linest 函數使用方式範例 (page 431)*

LINEST_SEY - 圖表函數

LINEST_SEY() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中 y 估計值的彙總標準誤差。

語法:

```
LINEST_SEY([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。
y0, x0	可指明選用值 y0, 強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0, 即可強制迴歸線通過單一固定座標。 *除非指明 y0 和 x0, 否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0, 則只要一個資料配對即可計算。*
SetExpression	依預設, 彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前, 會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前, 則會在提供現行選項的所有可能值上進行計算, 而不僅僅是關於目前維度值的那些選項, 也就是說, 它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單), 您可建立總可能值的子集。

限制:

除非內部彙總包含 **TOTAL** 限定詞, 否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總, 請結合使用進階函數 **Aggr** 與指定維度。

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值, 會導致整個資料配對被忽略。

另請參見:

- *linest 函數使用方式範例 (page 431)*
- *Avg - 圖表函數 (page 378)*

LINEST_SSREG

LINEST_SSREG() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的迴歸平方和。

語法:

```
LINEST_SSREG (y_value, x_value[, y0 [, x0 ]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。
y(0), x (0)	可指明選用值 y0, 強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0, 即可強制迴歸線通過單一固定座標。 除非指明 y0 和 x0, 否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0, 則只要一個資料配對即可計算。

限制:

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值, 會導致整個資料配對被忽略。

另請參見:

- *linest 函數使用方式範例 (page 431)*

LINEST_SSREG - 圖表函數

LINEST_SSREG() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的迴歸平方和。

語法:

```
LINEST_SSREG([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。
y0, x0	可指明選用值 y0, 強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0, 即可強制迴歸線通過單一固定座標。 *除非指明 y0 和 x0, 否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0, 則只要一個資料配對即可計算。*
SetExpression	依預設, 彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前, 會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前, 則會在提供現行選項的所有可能值上進行計算, 而不僅僅是關於目前維度值的那些選項, 也就是說, 它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單), 您可建立總可能值的子集。

限制:

除非內部彙總包含 **TOTAL** 限定詞, 否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總, 請結合使用進階函數 **Aggr** 與指定維度。

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值, 會導致整個資料配對被忽略。

另請參見:

- *linest 函數使用方式範例 (page 431)*
- *Avg - 圖表函數 (page 378)*

LINEST_SSRESID

LINEST_SSRESID() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的殘差平方和。

語法:

```
LINEST_SSRESID (y_value, x_value[, y0 [, x0 ]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。
y(0), x (0)	可指明選用值 y0, 強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0, 即可強制迴歸線通過單一固定座標。 除非指明 y0 和 x0, 否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0, 則只要一個資料配對即可計算。

限制:

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值, 會導致整個資料配對被忽略。

另請參見:

linest 函數使用方式範例 (page 431)

LINEST_SSRESID - 圖表函數

LINEST_SSRESID() 會針對圖表維度上反覆運算之運算式提供的配對數字 (以 **x_value** 和 **y_value** 表示) 所代表的一組座標, 傳回以方程式 y=mx+b 所定義之線性迴歸中彙總的殘差平方和。

語法:

```
LINEST_SSRESID([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。

引數	描述
y0, x0	可指明選用值 y0，強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0，即可強制迴歸線通過單一固定座標。 *除非指明 y0 和 x0，否則此函數需要至少兩個有效的資料配對才能計算。若指明 y0 和 x0，則只要一個資料配對即可計算。*
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前，會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

可指明選用值 y0，強制迴歸線在指定點通過 Y 軸。同時指明 y0 和 x0，即可強制迴歸線通過單一固定座標。

限制：

除非內部彙總包含 **TOTAL** 限定詞，否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總，請結合使用進階函數 **Aggr** 與指定維度。

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值，會導致整個資料配對被忽略。

另請參見：

- *linest 函數使用方式範例 (page 431)*
- *Avg - 圖表函數 (page 378)*

Median

Median() 會根據 **group by** 子句所定義的若干記錄，來傳回運算式中值的彙總中位數。

語法：

```
Median (expr)
```

傳回的資料類型： 數值

引數：

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。

範例:使用中位數的指令碼運算式

範例 - 指令碼運算式

載入指令碼

為此範例在資料載入編輯器中載入下列內嵌資料和指令碼運算式。

```
Table 1:
Load RecNo() as RowNo, Letter, Number Inline
[Letter, Number
A,1
A,3
A,4
A,9
B,2
B,8
B,9];

Median:
LOAD Letter,
Median(Number) as MyMedian
Resident Table1 Group By Letter;
```

建立視覺化

在 Qlik Sense 工作表中建立具有 **Letter** 和 **MyMedian** 作為維度的表格視覺化。

結果

Letter	MyMedian
A	3.5
B	8

解釋

當數字按照從最小到最大的順序排序時,中位數被視為「中間」數。若資料集的值數量為偶數,函數會傳回兩個中間值的平均。在此範例中,各為 **A** 和 **B** 的值集合計算了中位數,分別是 3.5 和 8。

Median - 圖表函數

Median() 傳回圖表維度上反覆運算的運算式中彙總值範圍的中位值。

語法:

```
Median([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
SetExpression	依預設,彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前,會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前,則會在提供現行選項的所有可能值上進行計算,而不僅僅是關於目前維度值的那些選項,也就是說,它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單),您可建立總可能值的子集。

限制:

除非內部彙總包含 **TOTAL** 限定詞,否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總,請結合使用進階函數 **Aggr** 與指定維度。

範例:使用中位數的圖表運算式

範例 - 圖表運算式

載入指令碼

在資料載入編輯器中載入下列資料作為內嵌載入,以建立以下的圖表運算式範例。

```
Load RecNo() as RowNo, Letter, Number Inline
[Letter, Number
A,1
A,3
A,4
A,9
B,2
B,8
B,9];
```

建立視覺化

在 Qlik Sense 工作表中建立具有 **Letter** 作為維度的表格視覺化。

圖表運算式

將以下運算式作為量值新增至表格:

```
Median(Number)
```

結果

Letter	Median(Number)
Totals	4
A	3.5
B	8

解釋

當數字按照從最小到最大的順序排序時，中位數被視為「中間」數。若資料集的值數量為偶數，函數會傳回兩個中間值的平均。在此範例中，各為 **A** 和 **B** 的值集合計算了中位數，分別是 3.5 和 8。

Totals 的中位數透過所有值計算，等於 4。

另請參見：

Avg - 圖表函數 (page 378)

MutualInfo - 圖表函數

MutualInfo 計算兩個欄位之間或 **Aggr()** 中的彙總值之間的相互資訊 (MI)。

MutualInfo 傳回兩個資料集的彙總相互資訊。這可在欄位和潛在驅動因素之間進行關鍵驅動因素分析。相互資訊衡量資料集之間的關係，對圖表維度上反覆運算的 (x,y) 配對值進行彙總。相互資訊在 0 和 1之間進行衡量，可以格式化為百分位數值。**MutualInfo** 由選項或集合運算式定義。

MutualInfo 可進行不同的 MI 分析：

- 成對 MI：在驅動程式欄位和目標欄位之間計算 MI。
- 依值進行驅動程式分解：在驅動程式和目標欄位的個別欄位值之間計算 MI。
- 功能選項：使用 **MutualInfo** 在格線圖中建立矩陣，在此根據 MI 互相比較所有欄位。

MutualInfo 不必指示共用相互資訊的欄位之間的因果關係。兩個欄位可共用相互資訊，但可能不等於彼此的驅動因素。例如，比較冰淇淋銷售量和室外溫度時，**MutualInfo** 將會顯示兩者之間的相互資訊。這不會指示室外溫度是否驅動了冰淇淋銷售量 (這有可能發生)，也不會指示冰淇淋銷售量是否驅動了室外溫度 (這不可能發生)。

計算相互資訊時，關聯會影響來自不同表格的欄位值之間的對應和頻率。

對相同欄位或選項傳回的值可能稍有不同。這是因為每個 **MutualInfo** 呼叫以隨機選取的樣本和 **MutualInfo** 演算法的固有隨機性來操作。

MutualInfo 可以套用至 **Aggr()** 函數。

語法：

```
MutualInfo({SetExpression}] [DISTINCT] [TOTAL] field1, field2 , datatype [, breakdownbyvalue [, samplesize ]])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
field1, field2	含有兩個待測量相互資訊之樣本集合的運算式或欄位。
datatype	包含在目標和驅動程式中的資料類型， 1 或 'dd' 用於 discrete:discrete 2 或 'cc' 用於 continuous:continuous 3 或 'cd' 用於 continuous:discrete 4 或 'dc' 用於 discrete:continuous 資料類型不區分大小寫。
breakdownbyvalue	靜態值對應至驅動程式中的值。若套用，計算將會計算 MI 對該值的占比。您可以使用 **ValueList()** 或 **ValueLoop()**。若新增 **Null()**，計算將會計算驅動程式中所有值的整體 MI。 依值分解需要驅動程式包含離散資料。
samplesize	要從目標和驅動程式取樣的值的數量。取樣為隨機。**MutualInfo** 需要的最低取樣大小為 80。依照預設，**MutualInfo** 取樣最多僅達 10,000 個資料對，因為 **MutualInfo** 可以是資源密集性質。您可以依樣本大小指定更大的資料對數量。若 **MutualInfo** 逾時，減少樣本大小。
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前，會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

限制：

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值，會導致整個資料配對被忽略。

範例與結果：

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

函數範例

範例	結果
`mutualinfo(Age, Salary, 1)`	對於包含維度 `Employee name` 和量值 `mutualinfo(Age, Salary, 1)` 的表格，結果為 0.99820986。只會顯示總計儲存格的結果。
`mutualinfo (TOTAL Age, Salary, 1, null (), 81)`	如果您使用維度 Gender 建立篩選窗格並從中進行選取，當選取 Female 時，您會看到結果為 0.99805677，當選取 Male 時，則結果為 0.99847373。這是因為選項中排除了不屬於 Gender 的其他值的所有結果。
`mutualinfo (TOTAL Age, Gender, 1, ValueLoop(25,35))`	0.68196996. 從 Gender 選取任何值會將此變更為 0。
`mutualinfo({1} TOTAL Age, Salary, 1, null ())`	0.99820986. 這獨立於選取的選項。集合運算式 {1} 忽略了所有選項和維度。

範例中使用的資料：

```
Salary:

LOAD * inline [

"Employee name"|Age|Gender|Salary

Aiden Charles|20|Male|25000

Ann Lindquist|69|Female|58000

Anna Johansen|37|Female|36000

Anna Karlsson|42|Female|23000

Antonio Garcia|20|Male|61000

Benjamin Smith|42|Male|27000

Bill Yang|49|Male|50000

Binh Protzmann|69|Male|21000

Bob Park|51|Male|54000

Brenda Davies|25|Male|32000
```

```
Celine Gagnon|48|Female|38000

Cezar Sandu|50|Male|46000

Charles Ingvar Jönsson|27|Male|58000

Charlotte Edberg|45|Female|56000

Cindy Lynn|69|Female|28000

Clark Wayne|63|Male|31000

Daroush Ferrara|31|Male|29000

David Cooper|37|Male|64000

David Leg|58|Male|57000

Eunice Goldblum|31|Female|32000

Freddy Halvorsen|25|Male|26000

Gauri Indu|36|Female|46000

George van Zaant|59|Male|47000

Glenn Brown|58|Male|40000

Harry Jones|38|Male|40000

Helen Brolin|52|Female|66000

Hiroshi Ito|24|Male|42000

Ian Underwood|40|Male|45000

Ingrid Hendrix|63|Female|27000

Ira Baumel|39|Female|39000

Jackie Kingsley|23|Female|28000

Jennica Williams|36|Female|48000

Jerry Tessel|31|Male|57000

Jim Bond|50|Male|58000

Joan Callins|60|Female|65000

Joan Cleaves|25|Female|61000

Joe Cheng|61|Male|41000
```

```
John Doe|36|Male|59000
John Lemon|43|Male|21000
Karen Helmkey|54|Female|25000
Karl Berger|38|Male|68000
Karl Straubaum|30|Male|40000
Kaya Alpan|32|Female|60000
Kenneth Finley|21|Male|25000
Leif Shine|63|Male|70000
Lennart Skoglund|63|Male|24000
Leona Korhonen|46|Female|50000
Lina André|50|Female|65000
Louis Presley|29|Male|36000
Luke Langston|50|Male|63000
Marcus Salvatori|31|Male|46000
Marie Simon|57|Female|23000
Mario Rossi|39|Male|62000
Markus Danzig|26|Male|48000
Michael Carlen|21|Male|45000
Michelle Tyson|44|Female|69000
Mike Ashkenaz|45|Male|68000
Miro Ito|40|Male|39000
Nina Mihn|62|Female|57000
Olivia Nguyen|35|Female|51000
Olivier Simenon|44|Male|31000
Östen Ärlig|68|Male|57000
Pamala Garcia|69|Female|29000
Paolo Romano|34|Male|45000
```

```
Pat Taylor|67|Female|69000

Paul Dupont|34|Male|38000

Peter Smith|56|Male|53000

Pierre Clouseau|21|Male|37000

Preben Jørgensen|35|Male|38000

Rey Jones|65|Female|20000

Ricardo Gucci|55|Male|65000

Richard Ranieri|30|Male|64000

Rob Carsson|46|Male|54000

Rolf Wesenlund|25|Male|51000

Ronaldo Costa|64|Male|39000

Sabrina Richards|57|Female|40000

Sato Hiromu|35|Male|21000

Sehoon Daw|57|Male|24000

Stefan Lind|67|Male|35000

Steve Cioazzi|58|Male|23000

Sunil Gupta|45|Male|40000

Sven Svensson|45|Male|55000

Tom Lindwall|46|Male|24000

Tomas Nilsson|27|Male|22000

Trinity Rizzo|52|Female|48000

Vanessa Lambert|54|Female|27000

] (delimiter is '|');
```

Skew

Skew() 會針對由 **group by** 子句所定義的若干記錄，傳回運算式的偏態。

語法：

```
Skew([ distinct] expr)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
DISTINCT	如果 **distinct** 出現在運算式之前,將會忽略所有的重複值。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。然後使用 `Type` 和 *MySkew* 作為維度建立一個連續表。

結果資料

<table>
<tr><th>範例</th><th>結果</th></tr>
<tr><td>
<pre>
Table1:
Crosstable (Type, Value)
Load recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2] (delimiter is '|');

Skew1:
LOAD Type,
Skew(Value) as MySkew
Resident Table1 Group By Type;
</pre>
</td><td>
Skew() 的計算結果如下:

<code>Type</code> 為 <code>MySkew</code>
<code>Comparison</code> 為 0.86414768
<code>Observation</code> 為 0.32625351

</td></tr>
</table>

Skew - 圖表函數

Skew() 傳回圖表維度上反覆運算的運算式或欄位的彙總 skewness。

語法:

```
Skew([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
SetExpression	依預設,彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前,會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前,則會在提供現行選項的所有可能值上進行計算,而不僅僅是關於目前維度值的那些選項,也就是說,它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單),您可建立總可能值的子集。

限制:

除非內部彙總包含 **TOTAL** 限定詞,否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總,請結合使用進階函數 **Aggr** 與指定維度。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。然後使用 `Type` 作為維度、`Skew(Value)` 作為量值建立一個連續表。

表格屬性中應啟用 `Totals`。

範例	結果
Table1: Crosstable (Type, Value) Load recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|');	Skew(Value) 的計算結果如下: • Total為 0.23522195 • Comparison 為 0.86414768 • Observation 為 0.32625351

另請參見:

Avg - 圖表函數 (page 378)

Stdev

Stdev() 會根據 **group by** 子句所定義的若干記錄,來傳回運算式中值的標準差。

語法:

```
Stdev([distinct] expr)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
distinct	如果 **distinct** 出現在運算式之前,將會忽略所有的重複值。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。然後使用 `Type` 和 `MyStdev` 作為維度建立一個連續表。

結果資料

<table>
<tr><th>範例</th><th>結果</th></tr>
<tr><td><pre>Table1:
Crosstable (Type, Value)
Load recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2] (delimiter is '|');

Stdev1:
LOAD Type,
Stdev(Value) as MyStdev
Resident Table1 Group By Type;</pre></td><td>Stdev() 的計算結果如下:
<code>Type</code> 為 <code>MyStdev</code>
<code>Comparison</code> 為 14.61245
<code>Observation</code> 為 12.507997</td></tr>
</table>

Stdev - 圖表函數

Stdev() 尋找圖表維度上反覆運算的運算式或欄位中彙總資料範圍的標準差。

語法:

```
Stdev([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
SetExpression	依預設,彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前,會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前,則會在提供現行選項的所有可能值上進行計算,而不僅僅是關於目前維度值的那些選項,也就是說,它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單),您可建立總可能值的子集。

限制:

除非內部彙總包含 **TOTAL** 限定詞,否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總,請結合使用進階函數 **Aggr** 與指定維度。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。然後使用 `Type` 作為維度、`Stdev(Value)` 作為量值建立一個連續表。

表格屬性中應啟用 `Totals`。

範例	結果

`Stdev(Value)`
`Table1:`
`Crosstable (Type, Value)`
`Load recno() as ID, * inline [`
`Observation|Comparison`
`35|2`
`40|27`
`12|38`
`15|31`
`21|1`
`14|19`
`46|1`
`10|34`
`28|3`
`48|1`
`16|2`
`30|3`
`32|2`
`48|1`
`31|2`
`22|1`
`12|3`
`39|29`
`19|37`
`25|2 ] (delimiter is '|');`
	Stdev(Value) 的計算結果如下:
	• `Total`為 15.47529
	• `Comparison` 為 14.61245
	• `Observation` 為 12.507997

另請參見:

- *Avg - 圖表函數 (page 378)*
- *STEYX - 圖表函數 (page 430)*

Sterr

Sterr() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的一組值 (以運算式表示), 傳回彙總的標準誤差 (stdev/sqrt(n))。

語法:

```
Sterr ([distinct] expr)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
distinct	如果 **distinct** 出現在運算式之前, 將會忽略所有的重複值。

限制：

文字值、NULL 值和遺漏值將予以忽略。

範例與結果：

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

結果資料

範例	結果

`Table1:`
`Crosstable (Type, Value)`
`Load recno() as ID, * inline [`
`Observation|Comparison`
`35|2`
`40|27`
`12|38`
`15|31`
`21|1`
`14|19`
`46|1`
`10|34`
`28|3`
`48|1`
`16|2`
`30|3`
`32|2`
`48|1`
`31|2`
`22|1`
`12|3`
`39|29`
`19|37`
`25|2 ] (delimiter is '|');`
 	
`Sterr1:`
`LOAD Type,`
`Sterr(Value) as MySterr`
`Resident Table1 Group By Type;`
	在具有 `Type` 和 `MySterr` 維度的表格中，資料載入指令碼中 Sterr() 計算的結果為：
	
	`Type MySterr`
	
	`Comparison 3.2674431`
	
	`Observation 2.7968733`

Sterr - 圖表函數

Sterr() 尋找圖表維度上反覆運算之運算式中彙總的一系列值的平均值標準誤差 (stdev/sqrt(n)) 的值。

語法：

```
Sterr([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
SetExpression	依預設, 彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前, 會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前, 則會在提供現行選項的所有可能值上進行計算, 而不僅僅是關於目前維度值的那些選項, 也就是說, 它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單), 您可建立總可能值的子集。

限制:

除非內部彙總包含 **TOTAL** 限定詞, 否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總, 請結合使用進階函數 **Aggr** 與指定維度。

文字值、NULL 值和遺漏值將予以忽略。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。然後使用 `Type` 作為維度、`Sterr(Value)` 作為量值建立一個連續表。

表格屬性中應啟用 `Totals`。

範例	結果
Table1: Crosstable (Type, Value) Load recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|');	Sterr(Value) 的計算結果如下: • Total為 2.4468583 • Comparison 為 3.2674431 • Observation 為 2.7968733

另請參見:

Avg - 圖表函數 (page 378)
STEYX - 圖表函數 (page 430)

STEYX

STEYX() 會針對由 **group by** 子句所定義之若干記錄上反覆運算的配對數字 (以 x-expression 和 y-expression 表示) 所代表的一組座標，傳回迴歸中每個 x 值之預測 y 值的彙總標準誤差。

語法:

```
STEYX (y_value, x_value)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
y_value	包含待測量之 y 值範圍的運算式或欄位。
x_value	包含待測量之 x 值範圍的運算式或欄位。

限制:

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值, 會導致整個資料配對被忽略。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。若要查看結果, 將結果資料行中列出的欄位新增至您應用程式中的工作表。

結果資料

範例	結果

`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;`
	在具有 MySTEYX 維度的表格中, 資料載入指令碼中 STEYX() 計算的結果是 2.0714764。

STEYX - 圖表函數

STEYX() 會針對運算式提供的配對數字 (以 **y_value** 和 **x_value** 表示) 所代表的一組座標，傳回線性迴歸中為每個 x 值預測 y 值時的彙總標準誤差。

語法：

```
STEYX([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value)
```

傳回的資料類型： 數值

引數：

引數

引數	描述
y_value	包含已知 y 值範圍的待測量運算式或欄位。
x_value	包含已知 x 值範圍的待測量運算式或欄位。
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前，會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

限制：

除非內部彙總包含 **TOTAL** 限定詞，否則彙總函數的參數不可包含其他彙總函數。如需更進階的巢狀彙總，請結合使用進階函數 **Aggr** 與指定維度。

若資料配對中有任一資料是 (或兩個都是) 文字值、NULL 值及遺漏值，會導致整個資料配對被忽略。

範例與結果：

將範例指令碼新增至您的應用程式並予以執行。然後使用 `KnownY` 和 `KnownX` 作為維度、`Steyx (KnownY,KnownX)` 作為量值建立一個連續表。

表格屬性中應啟用 `Totals`。

範例	結果

`Trend:`
 	
`LOAD * inline [`
 	
`Month|KnownY|KnownX`
 	
`Jan|2|6`
 	
`Feb|3|5`
 	
`Mar|9|11`
 	
`Apr|6|7`
 	
`May|8|5`
 	
`Jun|7|4`
 	
`Jul|5|5`
 	
`Aug|10|8`
 	
`Sep|9|10`
 	
`Oct|12|14`
 	
`Nov|15|17`
 	
`Dec|14|16`
 	
`] (delimiter is '|');`
	STEYX(KnownY,KnownX) 的計算結果為 2.071 (若數字格式設定設為 3 位小數)。

另請參見:

- *Avg - 圖表函數 (page 378)*
- *Sterr - 圖表函數 (page 426)*

linest 函數使用方式範例

linest 函數可用來得出與線性迴歸分析相關的值。本節說明如何使用樣本資料建置視覺化, 進而得出 Qlik Sense 中可用 linest 函數的值。linest 函數可用於資料載入指令碼和圖表運算式。

請參閱個別 linest 圖表函數與指令碼函數主題, 取得語法及引數的說明。

範例中使用的資料和指令碼運算式

為以下 linest() 範例在資料載入編輯器中載入下列內嵌資料和指令碼運算式。

```
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;
```

範例 1:使用 linest 的指令碼運算式

範例:指令碼運算式

從資料載入指令碼計算建立視覺化

以下列欄位作為欄在 Qlik Sense 工作表中建立表格視覺化:

- Linest_B
- Linest_DF
- Linest_F
- Linest_M

- Linest_R2
- Linest_SEB
- Linest_SEM
- Linest_SEY
- Linest_SSREG
- Linest_SSRESID

結果

包含用資料載入指令碼得出的 linest 計算結果之表格外觀應如下所示:

結果表格

Linest_B	Linest_DF	Linest_F	Linest_M	Linest_R2	Linest_SEB
-35.047	18	20.788	8.605	0.536	22.607

結果表格

Linest_SEM	Linest_SEY	Linest_SSREG	Linest_SSRESID
1.887	48.666	49235.014	42631.186

範例 2:使用 linest 的圖表運算式

範例:圖表運算式

以下列欄位作為維度在 Qlik Sense 工作表中建立表格視覺化:

```
ValueList('Linest_b', 'Linest_df','Linest_f', 'Linest_m','Linest_r2','Linest_SEB','Linest_
SEM','Linest_SEY','Linest_SSREG','Linest_SSRESID')
```

此運算式使用合成維度函數來建立含有 linest 函數名稱的維度標籤。您可將標籤變更為 **Linest functions** 以節省空間。

將以下運算式作為量值新增至表格:

```
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) )
```

此運算式顯示針對合成維度中對應名稱的各 linest 函數結果值。`Linest_b(Y,X)` 結果會顯示在 **linest_b** 旁邊,以此類推。

結果

結果表格

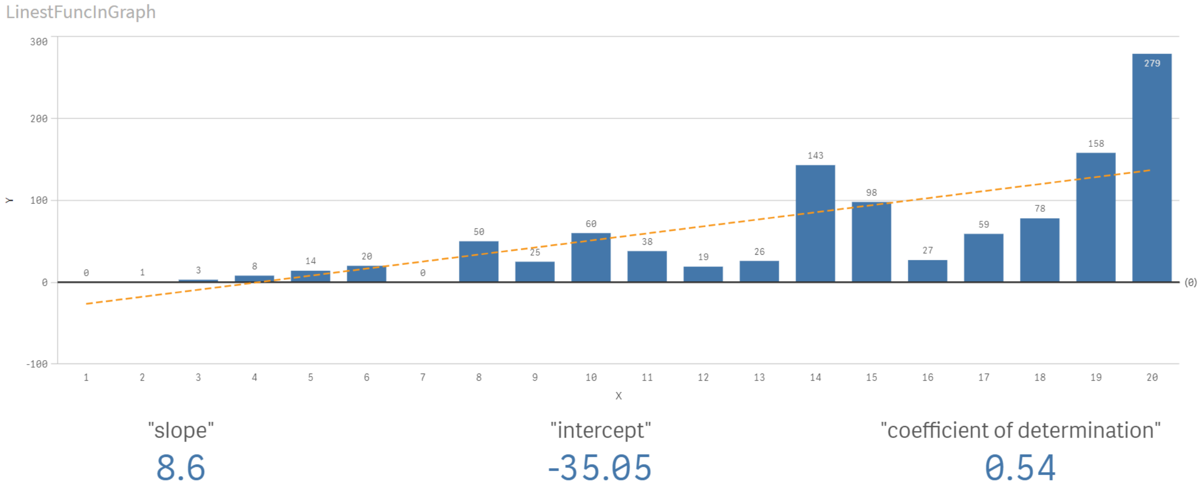
Linest functions	**Linest function results**
Linest_b	-35.047
Linest_df	18
Linest_f	20.788
Linest_m	8.605
Linest_r2	0.536
Linest_SEB	22.607
Linest_SEM	1.887
Linest_SEY	48.666
Linest_SSREG	49235.014
Linest_SSRESID	42631.186

範例 3:使用 linest 的圖表運算式

範例:圖表運算式

1. 在 Qlik Sense 工作表中建立長條圖視覺化,且 **X** 作為維度,而 **Y** 作為量值。
2. 將線性趨勢線新增至 Y 量值。
3. 將 KPI 視覺化新增至工作表。
    1. 新增*斜坡*作為 KPI 的標籤。
    2. 新增 `sum(Linest_M)` 作為 KPI 的運算式。
4. 將第二 KPI 視覺化新增至工作表。
    1. 新增*截距*作為 KPI 的標籤。
    2. 新增 `Sum(Linest_B)` 作為 KPI 的運算式。
5. 將第三 KPI 視覺化新增至工作表。
    1. 新增*決定係數*作為 KPI 的標籤。
    2. 新增 `Sum(Linest_R2)` 作為 KPI 的運算式。

結果



解釋

長條圖顯示 X 和 Y 資料的繪圖。相關的 linest() 函數提供趨勢線所根據的線性迴歸方程式的值，亦即 y = m * x + b。方程式使用「最小平方」方法，傳回描述最適合資料之線條的陣列以計算直線 (趨勢線)。

KPI 為斜坡顯示 linest() 函數的結果 **sum(Linest_M)** 並為 Y 截距顯示 **sum(Linest_B)**，這是線性迴歸方程式中的變數，以及決定係數的對應彙總 R2 值。

統計檢定函數

統計資料測試函數可同時用於資料載入指令碼和圖表運算式，但是語法有所不同。

Chi-2 檢定函數

通常用於量化變數的研究中。這項檢定可以比較在包含期望頻率的單向頻率表格中觀察到的頻率，或者研究偶然性表格中兩個變數間的連結。

T 檢定函數

t 檢定函數用於兩個母體平均數的統計檢查。雙樣本 t 檢定會檢查兩個樣本是否不同，以及是否通常在兩個常態分佈都具有未知變異數且試驗使用小型樣本時使用。

Z 檢定函數

兩個母體平均數的統計檢查。雙樣本 z 檢定會檢查兩個樣本是否不同，以及是否通常在兩個常態分佈都具有已知變異數且試驗使用大型樣本時使用。

Chi2-test 函數

通常用於量化變數的研究中。這項檢定可以比較在包含期望頻率的單向頻率表格中觀察到的頻率，或者研究偶然性表格中兩個變數間的連結。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.

如果該函數在資料載入指令碼中使用,則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用,則值會在圖表維度上反覆運算。

Chi2Test_chi2
Chi2Test_chi2() 會針對一或兩組值,傳回彙總 chi^2 檢定值。

```
Chi2Test_chi2() 會針對一或兩組值,傳回彙總 chi2 檢定值。(col, row, actual_value[, expected_value])
```

Chi2Test_df
Chi2Test_df() 會針對一或兩組值,傳回彙總 chi^2 檢定 df 值 (自由度)。

```
Chi2Test_df() 會針對一或兩組值,傳回彙總 chi2 檢定 df 值 (自由度)。(col, row, actual_value[, expected_value])
```

Chi2Test_p
Chi2Test_p() 會針對一或兩組值,傳回彙總 chi^2 檢定 p 值 (顯著)。

```
Chi2Test_p - 圖表函數(col, row, actual_value[, expected_value])
```

另請參見:

- *T 檢定函數 (page 439)*
- *Z 檢定函數 (page 469)*

Chi2Test_chi2

Chi2Test_chi2() 會針對一或兩組值,傳回彙總 chi^2 檢定值。

如果該函數在資料載入指令碼中使用,則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用,則值會在圖表維度上反覆運算。

> *全部 Qlik Sense chi^2 檢定函數具有相同的引數。*

語法:

```
Chi2Test_chi2(col, row, actual_value[, expected_value])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
col, row	正在檢定之值矩陣中指定的資料行及列。

引數	描述
actual_value	在指定 **col** 與 **row** 的資料的觀察值。
expected_value	在指定 **col** 與 **row** 的分佈的預期值。

限制:

運算式值中若有文字值、NULL 值及遺漏值, 會造成函數傳回 NULL。

範例:

```
Chi2Test_chi2( Grp, Grade, Count )
Chi2Test_chi2( Gender, Description, Observed, Expected )
```

另請參見:

- *圖表中 chi2-test 函數的使用方式範例 (page 482)*
- *資料載入指令碼中 chi2-test 函數的使用方式範例 (page 486)*

Chi2Test_df

Chi2Test_df() 會針對一或兩組值, 傳回彙總 chi^2 檢定 df 值 (自由度)。

如果該函數在資料載入指令碼中使用, 則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用, 則值會在圖表維度上反覆運算。

> *全部 Qlik Sense chi^2 檢定函數具有相同的引數。*

語法:

```
Chi2Test_df(col, row, actual_value[, expected_value])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
col, row	正在檢定之值矩陣中指定的資料行及列。
actual_value	在指定 **col** 與 **row** 的資料的觀察值。
expected_value	在指定 **col** 與 **row** 的分佈的預期值。

限制:

運算式值中若有文字值、NULL 值及遺漏值, 會造成函數傳回 NULL。

範例:

```
Chi2Test_df( Grp, Grade, Count )
Chi2Test_df( Gender, Description, Observed, Expected )
```

另請參見:

- *圖表中 chi2-test 函數的使用方式範例 (page 482)*
- *資料載入指令碼中 chi2-test 函數的使用方式範例 (page 486)*

Chi2Test_p - 圖表函數

Chi2Test_p() 會針對一或兩組值,傳回彙總 chi^2 檢定 p 值 (顯著)。檢定可針對 **actual_value** 中的值來檢定指定的 **col** 和 **row** 矩陣內的變異,或透過比較 **actual_value** 中的值與 **expected_value** 中的對應值來完成 (若已指定)。

如果該函數在資料載入指令碼中使用,則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用,則值會在圖表維度上反覆運算。

> *全部 Qlik Sense chi^2 檢定函數具有相同的引數。*

語法:

```
Chi2Test_p(col, row, actual_value[, expected_value])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
col, row	正在檢定之值矩陣中指定的資料行及列。
actual_value	在指定 **col** 與 **row** 的資料的觀察值。
expected_value	在指定 **col** 與 **row** 的分佈的預期值。

限制:

運算式值中若有文字值、NULL 值及遺漏值,會造成函數傳回 NULL。

範例:

```
Chi2Test_p( Grp, Grade, Count )
Chi2Test_p( Gender, Description, Observed, Expected )
```

另請參見：

- *圖表中 chi2-test 函數的使用方式範例 (page 482)*
- *資料載入指令碼中 chi2-test 函數的使用方式範例 (page 486)*

T 檢定函數

t 檢定函數用於兩個母體平均數的統計檢查。雙樣本 t 檢定會檢查兩個樣本是否不同，以及是否通常在兩個常態分佈都具有未知變異數且試驗使用小型樣本時使用。

在以下章節中，T 檢定統計檢定函數會根據適用於各類函數的樣本學生檢定加以分組。

建立一般 t-test 報表 (page 488)

兩組獨立的樣本 T 檢定

下列函數適用於兩組獨立樣本學生 T 檢定。

ttest_conf

TTest_conf 針對兩組獨立樣本傳回彙總 t 檢定信賴區間值。

```
TTest_conf 針對兩組獨立樣本傳回彙總 t 檢定信賴區間值。( grp, value [, sig[, eq_var]])
```

ttest_df

TTest_df() 會針對兩組獨立值，傳回彙總的學生 t 檢定值 (自由度)。

```
TTest_df() 會針對兩組獨立值，傳回彙總的學生 t 檢定值 (自由度)。(grp, value [, eq_var)
```

ttest_dif

TTest_dif() 是一個數值函數，針對兩組獨立值傳回彙總的學生 t 檢定平均差。

```
TTest_dif() 是一個數值函數，針對兩組獨立值傳回彙總的學生 t 檢定平均差。(grp, value)
```

ttest_lower

TTest_lower() 會針對兩組獨立值，傳回信賴區間下端的彙總值。

```
TTest_lower() 會針對兩組獨立值，傳回信賴區間下端的彙總值。(grp, value [, sig[, eq_var]])
```

ttest_sig

TTest_sig() 會針對兩組獨立值，傳回彙總的學生 t 檢定雙尾顯著性水平。

```
TTest_sig() 會針對兩組獨立值，傳回彙總的學生 t 檢定雙尾顯著性水平。(grp, value [, eq_var])
```

ttest_sterr

TTest_sterr() 會針對兩組獨立值，傳回彙總的學生 t 檢定平均差的標準誤差。

TTest_sterr() 會針對兩組獨立值，傳回彙總的學生 t 檢定平均差的標準誤差。(grp, value [, eq_var]**)**

ttest_t

TTest_t() 會針對兩組獨立值傳回彙總 t 值。

TTest_t() 會針對兩組獨立值傳回彙總 t 值。(grp, value [, eq_var]**)**

ttest_upper

TTest_upper() 會針對兩組獨立值，傳回信賴區間上端的彙總值。

TTest_upper() 會針對兩組獨立值，傳回信賴區間上端的彙總值。(grp, value [, sig [, eq_var]]**)**

兩組獨立的加權樣本 T 檢定

下列函數適用於兩組獨立樣本學生 T 檢定，其中輸入資料序列已指定為雙資料行加權的格式。

ttestw_conf

TTestw_conf() 會針對兩組獨立值傳回彙總 t 值。

TTestw_conf() 會針對兩組獨立值傳回彙總 t 值。(weight, grp, value [, sig[, eq_var]]**)**

ttestw_df

TTestw_df() 會針對兩組獨立值，傳回彙總的學生 t 檢定 df 值 (自由度)。

TTestw_df() 會針對兩組獨立值，傳回彙總的學生 t 檢定 df 值 (自由度)。(weight, grp, value [, eq_var]**)**

ttestw_dif

TTestw_dif() 會針對兩組獨立值，傳回彙總的學生 t 檢定平均差。

TTestw_dif() 會針對兩組獨立值，傳回彙總的學生 t 檢定平均差。(weight, grp, value**)**

ttestw_lower

TTestw_lower() 會針對兩組獨立值，傳回信賴區間下端的彙總值。

TTestw_lower() 會針對兩組獨立值，傳回信賴區間下端的彙總值。(weight, grp, value [, sig[, eq_var]]**)**

ttestw_sig

TTestw_sig() 會針對兩組獨立值，傳回彙總的學生 t 檢定雙尾顯著性水平。

TTestw_sig() 會針對兩組獨立值，傳回彙總的學生 t 檢定雙尾顯著性水平。(weight, grp, value [, eq_var]**)**

ttestw_sterr

TTestw_sterr() 會針對兩組獨立值，傳回彙總的學生 t 檢定平均差的標準誤差。

TTestw_sterr() 會針對兩組獨立值，傳回彙總的學生 t 檢定平均差的標準誤差。(weight, grp, value [, eq_var]**)**

ttestw_t

TTestw_t() 會針對兩組獨立值傳回彙總 t 值。

```
TTestw_t() 會針對兩組獨立值傳回彙總 t 值。(weight, grp, value [, eq_var])
```

ttestw_upper

TTestw_upper() 會針對兩組獨立值，傳回信賴區間上端的彙總值。

```
TTestw_upper() 會針對兩組獨立值，傳回信賴區間上端的彙總值。(weight, grp, value [, sig [, eq_var]])
```

單一樣本 T 檢定

下列函數適用於單一樣本學生 T 檢定。

ttest1_conf

TTest1_conf() 針對一組值傳回彙總信賴區間值。

```
TTest1_conf() 針對一組值傳回彙總信賴區間值。(value [, sig])
```

ttest1_df

TTest1_df() 會針對一組值，傳回彙總的學生 t 檢定 df 值 (自由度)。

```
TTest1_df() 會針對一組值，傳回彙總的學生 t 檢定 df 值 (自由度)。(value)
```

ttest1_dif

TTest1_dif() 會針對一組值，傳回彙總的學生 t 檢定平均差。

```
TTest1_dif() 會針對一組值，傳回彙總的學生 t 檢定平均差。(value)
```

ttest1_lower

TTest1_lower() 會針對一組值，傳回信賴區間下端的彙總值。

```
TTest1_lower() 會針對一組值，傳回信賴區間下端的彙總值。(value [, sig])
```

ttest1_sig

TTest1_sig() 會針對一組值，傳回彙總的學生 t 檢定雙尾顯著性水平。

```
TTest1_sig() 會針對一組值，傳回彙總的學生 t 檢定雙尾顯著性水平。(value)
```

ttest1_sterr

TTest1_sterr() 會針對一組值，傳回彙總的學生 t 檢定平均差的標準誤差。

```
TTest1_sterr() 會針對一組值，傳回彙總的學生 t 檢定平均差的標準誤差。(value)
```

ttest1_t

TTest1_t() 會針對一組值傳回彙總 t 值。

```
TTest1_t() 會針對一組值傳回彙總 t 值。(value)
```

ttest1_upper

TTest1_upper() 會針對一組值，傳回信賴區間上端的彙總值。

```
TTest1_upper() 會針對一組值，傳回信賴區間上端的彙總值。(value [, sig])
```

單一加權樣本 T 檢定

下列函數適用於單一樣本學生 T 檢定，其中輸入資料序列已指定為雙資料行加權的格式。

ttest1w_conf

TTest1w_conf() 是一個 **numeric** 函數，會針對一組值傳回彙總信賴區間值。

```
TTest1w_conf() 是一個 numeric 函數，會針對一組值傳回彙總信賴區間值。 (weight, value [, sig])
```

ttest1w_df

TTest1w_df() 會針對一組值，傳回彙總的學生 t 檢定 df 值 (自由度)。

```
TTest1w_df() 會針對一組值，傳回彙總的學生 t 檢定 df 值 (自由度)。 (weight, value)
```

ttest1w_dif

TTest1w_dif() 會針對一組值，傳回彙總的學生 t 檢定平均差。

```
TTest1w_dif() 會針對一組值，傳回彙總的學生 t 檢定平均差。 (weight, value)
```

ttest1w_lower

TTest1w_lower() 會針對一組值，傳回信賴區間下端的彙總值。

```
TTest1w_lower() 會針對一組值，傳回信賴區間下端的彙總值。 (weight, value [, sig])
```

ttest1w_sig

TTest1w_sig() 會針對一組值，傳回彙總的學生 t 檢定雙尾顯著性水平。

```
TTest1w_sig() 會針對一組值，傳回彙總的學生 t 檢定雙尾顯著性水平。 (weight, value)
```

ttest1w_sterr

TTest1w_sterr() 會針對一組值，傳回彙總的學生 t 檢定平均差的標準誤差。

```
TTest1w_sterr() 會針對一組值，傳回彙總的學生 t 檢定平均差的標準誤差。 (weight, value)
```

ttest1w_t

TTest1w_t() 會針對一組值傳回彙總 t 值。

```
TTest1w_t() 會針對一組值傳回彙總 t 值。 ( weight, value)
```

ttest1w_upper

TTest1w_upper() 會針對一組值，傳回信賴區間上端的彙總值。

```
TTest1w_upper() 會針對一組值，傳回信賴區間上端的彙總值。 (weight, value [, sig])
```

TTest_conf

TTest_conf 針對兩組獨立樣本傳回彙總 t 檢定信賴區間值。

此函數適用於獨立樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法:

```
TTest_conf ( grp, value [, sig [, eq_var]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱,則該欄位將自動命名為 **Value**。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱,則該欄位將自動命名為 **Type**。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略,**sig** 會設定為 0.025,產生 95% 信賴區間。
eq_var	如果 **eq_var** 指定為 False (0),則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1),則假設樣本之間有相等的變異。

限制:

運算式值中若有文字值、NULL 值及遺漏值,會造成函數傳回 NULL。

範例:

```
TTest_conf( Group, Value )
TTest_conf( Group, Value, Sig, false )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTest_df

TTest_df() 會針對兩組獨立值,傳回彙總的學生 t 檢定值 (自由度)。

此函數適用於獨立樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用,則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用,則值會在圖表維度上反覆運算。

語法:

```
TTest_df (grp, value [, eq_var])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱, 則該欄位將自動命名為 **Value**。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱, 則該欄位將自動命名為 **Type**。
eq_var	如果 **eq_var** 指定為 False (0), 則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1), 則假設樣本之間有相等的變異。

限制:

運算式值中若有文字值、NULL 值及遺漏值, 會造成函數傳回 NULL。

範例:

```
TTest_df( Group, Value )
TTest_df( Group, Value, false )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTest_dif

TTest_dif() 是一個數值函數, 針對兩組獨立值傳回彙總的學生 t 檢定平均差。

此函數適用於獨立樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用, 則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用, 則值會在圖表維度上反覆運算。

語法:

```
TTest_dif (grp, value [, eq_var] )
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱, 則該欄位將自動命名為 **Value**。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱, 則該欄位將自動命名為 **Type**。
eq_var	如果 **eq_var** 指定為 False (0), 則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1), 則假設樣本之間有相等的變異。

限制:

運算式值中若有文字值、NULL 值及遺漏值, 會造成函數傳回 NULL。

範例:

```
TTest_dif( Group, Value )
TTest_dif( Group, Value, false )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTest_lower

TTest_lower() 會針對兩組獨立值, 傳回信賴區間下端的彙總值。

此函數適用於獨立樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用, 則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用, 則值會在圖表維度上反覆運算。

語法:

```
TTest_lower (grp, value [, sig [, eq_var]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱,則該欄位將自動命名為 **Value**。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱,則該欄位將自動命名為 **Type**。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略,**sig** 會設定為 0.025,產生 95% 信賴區間。
eq_var	如果 **eq_var** 指定為 False (0),則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1),則假設樣本之間有相等的變異。

限制:

運算式值中若有文字值、NULL 值及遺漏值,會造成函數傳回 NULL。

範例:

```
TTest_lower( Group, Value )
TTest_lower( Group, Value, Sig, false )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTest_sig

TTest_sig() 會針對兩組獨立值,傳回彙總的學生 t 檢定雙尾顯著性水平。

此函數適用於獨立樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用,則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用,則值會在圖表維度上反覆運算。

語法:

```
TTest_sig (grp, value [, eq_var])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱, 則該欄位將自動命名為 **Value**。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱, 則該欄位將自動命名為 **Type**。
eq_var	如果 **eq_var** 指定為 False (0), 則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1), 則假設樣本之間有相等的變異。

限制:

運算式值中若有文字值、NULL 值及遺漏值, 會造成函數傳回 NULL。

範例:

```
TTest_sig( Group, Value )
TTest_sig( Group, Value, false )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTest_sterr

TTest_sterr() 會針對兩組獨立值, 傳回彙總的學生 t 檢定平均差的標準誤差。

此函數適用於獨立樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用, 則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用, 則值會在圖表維度上反覆運算。

語法:

```
TTest_sterr (grp, value [, eq_var])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱, 則該欄位將自動命名為 **Value**。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱, 則該欄位將自動命名為 **Type**。
eq_var	如果 **eq_var** 指定為 False (0), 則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1), 則假設樣本之間有相等的變異。

限制:

運算式值中若有文字值、NULL 值及遺漏值, 會造成函數傳回 NULL。

範例:

```
TTest_sterr( Group, Value )
TTest_sterr( Group, Value, false )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTest_t

TTest_t() 會針對兩組獨立值傳回彙總 t 值。

此函數適用於獨立樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用, 則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用, 則值會在圖表維度上反覆運算。

語法:

```
TTest_t(grp, value[, eq_var])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱, 則該欄位將自動命名為 **Value**。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱, 則該欄位將自動命名為 **Type**。
eq_var	如果 **eq_var** 指定為 False (0), 則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1), 則假設樣本之間有相等的變異。

限制:

運算式值中若有文字值、NULL 值及遺漏值, 會造成函數傳回 NULL。

範例:

```
TTest_t( Group, Value, false )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTest_upper

TTest_upper() 會針對兩組獨立值, 傳回信賴區間上端的彙總值。

此函數適用於獨立樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用, 則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用, 則值會在圖表維度上反覆運算。

語法:

```
TTest_upper (grp, value [, sig [, eq_var]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱, 則該欄位將自動命名為 **Value**。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱, 則該欄位將自動命名為 **Type**。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略, **sig** 會設定為 0.025, 產生 95% 信賴區間。
eq_var	如果 **eq_var** 指定為 False (0), 則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1), 則假設樣本之間有相等的變異。

限制:

運算式值中若有文字值、NULL 值及遺漏值, 會造成函數傳回 NULL。

範例:

```
TTest_upper( Group, Value )
TTest_upper( Group, Value, sig, false )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTestw_conf

TTestw_conf() 會針對兩組獨立值傳回彙總 t 值。

此函數適用於兩個獨立樣本學生 t 檢定, 其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用, 則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用, 則值會在圖表維度上反覆運算。

語法:

```
TTestw_conf (weight, grp, value [, sig [, eq_var]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱, 則該欄位將自動命名為 **Value**。
weight	**value** 中的各個值可根據 **weight** 中的對應加權值, 計數一次或多次。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱, 則該欄位將自動命名為 **Type**。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略, **sig** 會設定為 0.025, 產生 95% 信賴區間。
eq_var	如果 **eq_var** 指定為 False (0), 則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1), 則假設樣本之間有相等的變異。

限制:

運算式值中若有文字值、NULL 值及遺漏值, 會造成函數傳回 NULL。

範例:

```
TTestw_conf( Weight, Group, Value )
TTestw_conf( Weight, Group, Value, sig, false )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTestw_df

TTestw_df() 會針對兩組獨立值, 傳回彙總的學生 t 檢定 df 值 (自由度)。

此函數適用於兩個獨立樣本學生 t 檢定, 其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用, 則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用, 則值會在圖表維度上反覆運算。

語法:

```
TTestw_df (weight, grp, value [, eq_var])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
weight	**value** 中的各個值可根據 **weight** 中的對應加權值, 計數一次或多次。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱, 則該欄位將自動命名為 **Type**。
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱, 則該欄位將自動命名為 **Value**。
eq_var	如果 **eq_var** 指定為 False (0), 則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1), 則假設樣本之間有相等的變異。

限制:

運算式值中若有文字值、NULL 值及遺漏值, 會造成函數傳回 NULL。

範例:

```
TTestw_df( Weight, Group, Value )
TTestw_df( Weight, Group, Value, false )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTestw_dif

TTestw_dif() 會針對兩組獨立值, 傳回彙總的學生 t 檢定平均差。

此函數適用於兩個獨立樣本學生 t 檢定, 其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用, 則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用, 則值會在圖表維度上反覆運算。

語法:

```
TTestw_dif (weight, grp, value)
```

傳回的資料類型： 數值

引數：

引數

引數	描述
weight	**value** 中的各個值可根據 **weight** 中的對應加權值，計數一次或多次。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱，則該欄位將自動命名為 **Type**。
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
TTestw_dif( Weight, Group, Value )
TTestw_dif( Weight, Group, Value, false )
```

另請參見：

建立一般 t-test 報表 (page 488)

TTestw_lower

TTestw_lower() 會針對兩組獨立值，傳回信賴區間下端的彙總值。

此函數適用於兩個獨立樣本學生 t 檢定，其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
TTestw_lower (weight, grp, value [, sig [, eq_var]])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
weight	**value** 中的各個值可根據 **weight** 中的對應加權值，計數一次或多次。

引數	描述
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱，則該欄位將自動命名為 **Type**。
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略，**sig** 會設定為 0.025，產生 95% 信賴區間。
eq_var	如果 **eq_var** 指定為 False (0)，則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1)，則假設樣本之間有相等的變異。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
TTestw_lower( Weight, Group, Value )
TTestw_lower( Weight, Group, Value, sig, false )
```

另請參見：

建立一般 t-test 報表 (page 488)

TTestw_sig

TTestw_sig() 會針對兩組獨立值，傳回彙總的學生 t 檢定雙尾顯著性水平。

此函數適用於兩個獨立樣本學生 t 檢定，其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
TTestw_sig ( weight, grp, value [, eq_var])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
weight	**value** 中的各個值可根據 **weight** 中的對應加權值，計數一次或多次。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱，則該欄位將自動命名為 **Type**。

引數	描述
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。
eq_var	如果 **eq_var** 指定為 False (0)，則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1)，則假設樣本之間有相等的變異。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
TTestw_sig( Weight, Group, Value )
TTestw_sig( Weight, Group, Value, false )
```

另請參見：

建立一般 t-test 報表 (page 488)

TTestw_sterr

TTestw_sterr() 會針對兩組獨立值，傳回彙總的學生 t 檢定平均差的標準誤差。

此函數適用於兩個獨立樣本學生 t 檢定，其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
TTestw_sterr (weight, grp, value [, eq_var])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
weight	**value** 中的各個值可根據 **weight** 中的對應加權值，計數一次或多次。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱，則該欄位將自動命名為 **Type**。
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。
eq_var	如果 **eq_var** 指定為 False (0)，則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1)，則假設樣本之間有相等的變異。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
TTestw_sterr( Weight, Group, Value )
TTestw_sterr( Weight, Group, Value, false )
```

另請參見：

建立一般 t-test 報表 (page 488)

TTestw_t

TTestw_t() 會針對兩組獨立值傳回彙總 t 值。

此函數適用於兩個獨立樣本學生 t 檢定，其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
ttestw_t (weight, grp, value [, eq_var])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。
weight	**value** 中的各個值可根據 **weight** 中的對應加權值，計數一次或多次。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱，則該欄位將自動命名為 **Type**。
eq_var	如果 **eq_var** 指定為 False (0)，則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1)，則假設樣本之間有相等的變異。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
TTestw_t( Weight, Group, Value )
TTestw_t( Weight, Group, Value, false )
```

另請參見:

- *建立一般 t-test 報表 (page 488)*

TTestw_upper

TTestw_upper() 會針對兩組獨立值,傳回信賴區間上端的彙總值。

此函數適用於兩個獨立樣本學生 t 檢定,其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用,則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用,則值會在圖表維度上反覆運算。

語法:

```
TTestw_upper (weight, grp, value [, sig [, eq_var]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
weight	**value** 中的各個值可根據 **weight** 中的對應加權值,計數一次或多次。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱,則該欄位將自動命名為 **Type**。
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱,則該欄位將自動命名為 **Value**。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略,**sig** 會設定為 0.025,產生 95% 信賴區間。
eq_var	如果 **eq_var** 指定為 False (0),則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1),則假設樣本之間有相等的變異。

限制:

運算式值中若有文字值、NULL 值及遺漏值,會造成函數傳回 NULL。

範例:

```
TTestw_upper( Weight, Group, Value )
TTestw_upper( Weight, Group, Value, sig, false )
```

另請參見:

- *建立一般 t-test 報表 (page 488)*

TTest1_conf

TTest1_conf() 針對一組值傳回彙總信賴區間值。

此函數適用於單一樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
TTest1_conf (value [, sig ])
```

傳回的資料類型：數值

引數：

引數

引數	描述
value	即會評估樣本。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略，**sig** 會設定為 0.025，產生 95% 信賴區間。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
TTest1_conf( Value )
TTest1_conf( Value, 0.005 )
```

另請參見：

建立一般 t-test 報表 (page 488)

TTest1_df

TTest1_df() 會針對一組值，傳回彙總的學生 t 檢定 df 值 (自由度)。

此函數適用於單一樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
TTest1_df (value)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本。如果載入指令碼中未提供樣本值的欄位名稱,則該欄位將自動命名為 **Value**。

限制:

運算式值中若有文字值、NULL 值及遺漏值,會造成函數傳回 NULL。

範例:

```
TTest1_df( Value )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTest1_dif

TTest1_dif() 會針對一組值,傳回彙總的學生 t 檢定平均差。

此函數適用於單一樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用,則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用,則值會在圖表維度上反覆運算。

語法:

```
TTest1_dif (value)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本。如果載入指令碼中未提供樣本值的欄位名稱,則該欄位將自動命名為 **Value**。

限制:

運算式值中若有文字值、NULL 值及遺漏值,會造成函數傳回 NULL。

範例:

```
TTest1_dif( Value )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTest1_lower

TTest1_lower() 會針對一組值，傳回信賴區間下端的彙總值。

此函數適用於單一樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法:

```
TTest1_lower (value [, sig])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略，**sig** 會設定為 0.025，產生 95% 信賴區間。

限制:

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例:

```
TTest1_lower( Value )
TTest1_lower( Value, 0.005 )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTest1_sig

TTest1_sig() 會針對一組值，傳回彙總的學生 t 檢定雙尾顯著性水平。

此函數適用於單一樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
TTest1_sig (value)
```

傳回的資料類型： 數值

引數：

引數

引數	描述
value	即會評估樣本。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
TTest1_sig( Value )
```

另請參見：

建立一般 t-test 報表 (page 488)

TTest1_sterr

TTest1_sterr() 會針對一組值，傳回彙總的學生 t 檢定平均差的標準誤差。

此函數適用於單一樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
TTest1_sterr (value)
```

傳回的資料類型： 數值

引數：

引數

引數	描述
value	即會評估樣本。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。

限制:

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例:

```
TTest1_sterr( Value )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTest1_t

TTest1_t() 會針對一組值傳回彙總 t 值。

此函數適用於單一樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法:

TTest1_t (value**)**

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。

限制:

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例:

```
TTest1_t( Value )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTest1_upper

TTest1_upper() 會針對一組值，傳回信賴區間上端的彙總值。

此函數適用於單一樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
TTest1_upper (value [, sig])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
value	即會評估樣本。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略，**sig** 會設定為 0.025，產生 95% 信賴區間。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
TTest1_upper( Value )
TTest1_upper( Value, 0.005 )
```

另請參見：

建立一般 t-test 報表 (page 488)

TTest1w_conf

TTest1w_conf() 是一個 **numeric** 函數，會針對一組值傳回彙總信賴區間值。

此函數適用於單一樣本學生 t 檢定，其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
TTest1w_conf (weight, value [, sig ])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。
weight	**value** 中的各個值可根據 **weight** 中的對應加權值，計數一次或多次。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略，**sig** 會設定為 0.025，產生 95% 信賴區間。

限制:

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例:

```
TTest1w_conf( Weight, Value )
TTest1w_conf( Weight, Value, 0.005 )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTest1w_df

TTest1w_df() 會針對一組值，傳回彙總的學生 t 檢定 df 值 (自由度)。

此函數適用於單一樣本學生 t 檢定，其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法:

```
TTest1w_df (weight, value)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。
weight	**value** 中的各個值可根據 **weight** 中的對應加權值，計數一次或多次。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
TTest1w_df( Weight, Value )
```

另請參見：

建立一般 t-test 報表 (page 488)

TTest1w_dif

TTest1w_dif() 會針對一組值，傳回彙總的學生 t 檢定平均差。

此函數適用於單一樣本學生 t 檢定，其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
TTest1w_dif (weight, value)
```

傳回的資料類型： 數值

引數：

引數

引數	描述
value	即會評估樣本。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。
weight	**value** 中的各個值可根據 **weight** 中的對應加權值，計數一次或多次。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
TTest1w_dif( Weight, Value )
```

另請參見：

建立一般 t-test 報表 (page 488)

TTest1w_lower

TTest1w_lower() 會針對一組值，傳回信賴區間下端的彙總值。

此函數適用於單一樣本學生 t 檢定，其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
TTest1w_lower (weight, value [, sig ])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
value	即會評估樣本。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。
weight	**value** 中的各個值可根據 **weight** 中的對應加權值，計數一次或多次。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略，**sig** 會設定為 0.025，產生 95% 信賴區間。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
TTest1w_lower( Weight, Value )
TTest1w_lower( Weight, Value, 0.005 )
```

另請參見：

建立一般 t-test 報表 (page 488)

TTest1w_sig

TTest1w_sig() 會針對一組值，傳回彙總的學生 t 檢定雙尾顯著性水平。

此函數適用於單一樣本學生 t 檢定，其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
TTest1w_sig (weight, value)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本。如果載入指令碼中未提供樣本值的欄位名稱,則該欄位將自動命名為 **Value**。
weight	**value** 中的各個值可根據 **weight** 中的對應加權值,計數一次或多次。

限制:

運算式值中若有文字值、NULL 值及遺漏值,會造成函數傳回 NULL。

範例:

```
TTest1w_sig( Weight, Value )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTest1w_sterr

TTest1w_sterr() 會針對一組值,傳回彙總的學生 t 檢定平均差的標準誤差。

此函數適用於單一樣本學生 t 檢定,其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用,則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用,則值會在圖表維度上反覆運算。

語法:

```
TTest1w_sterr (weight, value)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本。如果載入指令碼中未提供樣本值的欄位名稱,則該欄位將自動命名為 **Value**。
weight	**value** 中的各個值可根據 **weight** 中的對應加權值,計數一次或多次。

限制:

運算式值中若有文字值、NULL 值及遺漏值, 會造成函數傳回 NULL。

範例:

```
TTest1w_sterr( Weight, Value )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTest1w_t

TTest1w_t() 會針對一組值傳回彙總 t 值。

此函數適用於單一樣本學生 t 檢定, 其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用, 則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用, 則值會在圖表維度上反覆運算。

語法:

```
TTest1w_t ( weight, value)
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本。如果載入指令碼中未提供樣本值的欄位名稱, 則該欄位將自動命名為 **Value**。
weight	**value** 中的各個值可根據 **weight** 中的對應加權值, 計數一次或多次。

限制:

運算式值中若有文字值、NULL 值及遺漏值, 會造成函數傳回 NULL。

範例:

```
TTest1w_t( Weight, Value )
```

另請參見:

建立一般 t-test 報表 (page 488)

TTest1w_upper

TTest1w_upper() 會針對一組值, 傳回信賴區間上端的彙總值。

此函數適用於單一樣本學生 t 檢定, 其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用, 則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用, 則值會在圖表維度上反覆運算。

語法:

```
TTest1w_upper (weight, value [, sig])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本。如果載入指令碼中未提供樣本值的欄位名稱, 則該欄位將自動命名為 **Value**。
weight	**value** 中的各個值可根據 **weight** 中的對應加權值, 計數一次或多次。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略, **sig** 會設定為 0.025, 產生 95% 信賴區間。

限制:

運算式值中若有文字值、NULL 值及遺漏值, 會造成函數傳回 NULL。

範例:

```
TTest1w_upper( Weight, Value )
TTest1w_upper( Weight, Value, 0.005 )
```

另請參見:

建立一般 t-test 報表 (page 488)

Z 檢定函數

兩個母體平均數的統計檢查。雙樣本 z 檢定會檢查兩個樣本是否不同, 以及是否通常在兩個常態分佈都具有已知變異數且試驗使用大型樣本時使用。

Z 檢定統計檢定函數會根據套用到函數的輸入資料序列類型加以分組。

如果該函數在資料載入指令碼中使用, 則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用, 則值會在圖表維度上反覆運算。

z-test 函數使用方式範例 (page 491)

單資料行格式函數

下列函數適用於具有簡單輸入資料序列的 z 檢定。

ztest_conf

ZTest_conf() 針對一組值傳回彙總的 z 值。

ZTest_conf() 針對一組值傳回彙總的 z 值。 (value [, sigma [, sig]**)**

ztest_dif

ZTest_dif() 會針對一組值, 傳回彙總的 z 檢定平均差。

ZTest_dif() 會針對一組值, 傳回彙總的 z 檢定平均差。 (value [, sigma]**)**

ztest_sig

ZTest_sig() 會針對一組值, 傳回彙總的 z 檢定雙尾顯著性水平。

ZTest_sig() 會針對一組值, 傳回彙總的 z 檢定雙尾顯著性水平。 (value [, sigma]**)**

ztest_sterr

ZTest_sterr() 會針對一組值, 傳回彙總的 z 檢定平均差的標準誤差。

ZTest_sterr() 會針對一組值, 傳回彙總的 z 檢定平均差的標準誤差。 (value [, sigma]**)**

ztest_z

ZTest_z() 針對一組值傳回彙總的 z 值。

ZTest_z() 針對一組值傳回彙總的 z 值。 (value [, sigma]**)**

ztest_lower

ZTest_lower() 會針對兩組獨立值, 傳回信賴區間下端的彙總值。

ZTest_lower() 會針對兩組獨立值, 傳回信賴區間下端的彙總值。 (grp, value [, sig [, eq_var]]**)**

ztest_upper

ZTest_upper() 會針對兩組獨立值, 傳回信賴區間上端的彙總值。

ZTest_upper() 會針對兩組獨立值, 傳回信賴區間上端的彙總值。 (grp, value [, sig [, eq_var]]**)**

加權雙資料行格式函數

下列函數適用於 z 檢定, 其中輸入資料序列已指定為雙資料行加權的格式。

ztestw_conf

ZTestw_conf() 針對一組值傳回彙總 z 信賴區間值。

ZTestw_conf() 針對一組值傳回彙總 z 信賴區間值。 (weight, value [, sigma [, sig]]**)**

ztestw_dif

ZTestw_dif() 會針對一組值, 傳回彙總的 z 檢定平均差。

ZTestw_dif() 會針對一組值,傳回彙總的 z 檢定平均差。 (weight, value [, sigma]**)**

ztestw_lower

ZTestw_lower() 會針對兩組獨立值,傳回信賴區間下端的彙總值。

ZTestw_lower() 會針對兩組獨立值,傳回信賴區間下端的彙總值。 (weight, value [, sigma]**)**

ztestw_sig

ZTestw_sig() 會針對一組值,傳回彙總的 z 檢定雙尾顯著性水平。

ZTestw_sig() 會針對一組值,傳回彙總的 z 檢定雙尾顯著性水平。 (weight, value [, sigma]**)**

ztestw_sterr

ZTestw_sterr() 會針對一組值,傳回彙總的 z 檢定平均差的標準誤差。

ZTestw_sterr() 會針對一組值,傳回彙總的 z 檢定平均差的標準誤差。 (weight, value [, sigma]**)**

ztestw_upper

ZTestw_upper() 會針對兩組獨立值,傳回信賴區間上端的彙總值。

ZTestw_upper() 會針對兩組獨立值,傳回信賴區間上端的彙總值。 (weight, value [, sigma]**)**

ztestw_z

ZTestw_z() 針對一組值傳回彙總的 z 值。

ZTestw_z() 針對一組值傳回彙總的 z 值。 (weight, value [, sigma]**)**

ZTest_z

ZTest_z() 針對一組值傳回彙總的 z 值。

如果該函數在資料載入指令碼中使用,則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用,則值會在圖表維度上反覆運算。

語法:

```
ZTest_z(value[, sigma])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本值。假設母體平均數為 0。若要對另一個平均數執行檢定,可從樣本值減去該平均數。
sigma	如果已知,可在 **sigma** 中指出標準差。如果省略 **sigma**,則會使用實際的樣本標準差。

限制:

運算式值中若有文字值、NULL 值及遺漏值,會造成函數傳回 NULL。

範例:

```
ZTest_z( Value-TestValue )
```

另請參見:

z-test 函數使用方式範例 (page 491)

ZTest_sig

ZTest_sig() 會針對一組值,傳回彙總的 z 檢定雙尾顯著性水平。

如果該函數在資料載入指令碼中使用,則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用,則值會在圖表維度上反覆運算。

語法:

```
ZTest_sig(value[, sigma])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本值。假設母體平均數為 0。若要對另一個平均數執行檢定,可從樣本值減去該平均數。
sigma	如果已知,可在 **sigma** 中指出標準差。如果省略 **sigma**,則會使用實際的樣本標準差。

限制:

運算式值中若有文字值、NULL 值及遺漏值,會造成函數傳回 NULL。

範例:

```
ZTest_sig(Value-TestValue)
```

另請參見:

z-test 函數使用方式範例 (page 491)

ZTest_dif

ZTest_dif() 會針對一組值,傳回彙總的 z 檢定平均差。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
ZTest_dif(value[, sigma])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
value	即會評估樣本值。假設母體平均數為 0。若要對另一個平均數執行檢定，可從樣本值減去該平均數。
sigma	如果已知，可在 **sigma** 中指出標準差。如果省略 **sigma**，則會使用實際的樣本標準差。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
ZTest_dif(Value-TestValue)
```

另請參見：

z-test 函數使用方式範例 (page 491)

ZTest_sterr

ZTest_sterr() 會針對一組值，傳回彙總的 z 檢定平均差的標準誤差。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
ZTest_sterr(value[, sigma])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本值。假設母體平均數為 0。若要對另一個平均數執行檢定,可從樣本值減去該平均數。
sigma	如果已知,可在 **sigma** 中指出標準差。如果省略 **sigma**,則會使用實際的樣本標準差。

限制:

運算式值中若有文字值、NULL 值及遺漏值,會造成函數傳回 NULL。

範例:

```
ZTest_sterr(Value-TestValue)
```

另請參見:

z-test 函數使用方式範例 (page 491)

ZTest_conf

ZTest_conf() 針對一組值傳回彙總的 z 值。

如果該函數在資料載入指令碼中使用,則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用,則值會在圖表維度上反覆運算。

語法:

```
ZTest_conf(value[, sigma[, sig]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本值。假設母體平均數為 0。若要對另一個平均數執行檢定,可從樣本值減去該平均數。
sigma	如果已知,可在 **sigma** 中指出標準差。如果省略 **sigma**,則會使用實際的樣本標準差。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略,**sig** 會設定為 0.025,產生 95% 信賴區間。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
ZTest_conf(Value-TestValue)
```

另請參見：

z-test 函數使用方式範例 (page 491)

ZTest_lower

ZTest_lower() 會針對兩組獨立值，傳回信賴區間下端的彙總值。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
ZTest_lower (grp, value [, sig [, eq_var]])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱，則該欄位將自動命名為 **Type**。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略，**sig** 會設定為 0.025，產生 95% 信賴區間。
eq_var	如果 **eq_var** 指定為 False (0)，則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1)，則假設樣本之間有相等的變異。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
ZTest_lower( Group, Value )
ZTest_lower( Group, Value, sig, false )
```

另請參見:

z-test 函數使用方式範例 (page 491)

ZTest_upper

ZTest_upper() 會針對兩組獨立值，傳回信賴區間上端的彙總值。

此函數適用於獨立樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法:

```
ZTest_upper (grp, value [, sig [, eq_var]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱，則該欄位將自動命名為 **Type**。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略，**sig** 會設定為 0.025，產生 95% 信賴區間。
eq_var	如果 **eq_var** 指定為 False (0)，則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1)，則假設樣本之間有相等的變異。

限制:

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例:

```
ZTest_upper( Group, Value )
ZTest_upper( Group, Value, sig, false )
```

另請參見:

z-test 函數使用方式範例 (page 491)

ZTestw_z

ZTestw_z() 針對一組值傳回彙總的 z 值。

此函數適用於 z 檢定，其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
ZTestw_z (weight, value [, sigma])
```

傳回的資料類型：數值

引數：

引數

引數	描述
value	值應會透過 **value** 傳回。假設樣本平均數為 0。若要對另一個平均數執行檢定，可從樣本值減去該值。
weight	**value** 中的各個樣本值可根據 **weight** 中的對應加權值，計數一次或多次。
sigma	如果已知，可在 **sigma** 中指出標準差。如果省略 **sigma**，則會使用實際的樣本標準差。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
ZTestw_z( Weight, Value-TestValue)
```

另請參見：

z-test 函數使用方式範例 (page 491)

ZTestw_sig

ZTestw_sig() 會針對一組值，傳回彙總的 z 檢定雙尾顯著性水平。

此函數適用於 z 檢定，其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
ZTestw_sig (weight, value [, sigma])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	值應會透過 **value** 傳回。假設樣本平均數為 0。若要對另一個平均數執行檢定，可從樣本值減去該值。
weight	**value** 中的各個樣本值可根據 **weight** 中的對應加權值，計數一次或多次。
sigma	如果已知，可在 **sigma** 中指出標準差。如果省略 **sigma**，則會使用實際的樣本標準差。

限制:

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例:

```
ZTestw_sig( Weight, Value-TestValue)
```

另請參見:

z-test 函數使用方式範例 (page 491)

ZTestw_dif

ZTestw_dif() 會針對一組值，傳回彙總的 z 檢定平均差。

此函數適用於 z 檢定，其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法:

```
ZTestw_dif ( weight, value [, sigma])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	值應會透過 **value** 傳回。假設樣本平均數為 0。若要對另一個平均數執行檢定，可從樣本值減去該值。
weight	**value** 中的各個樣本值可根據 **weight** 中的對應加權值，計數一次或多次。
sigma	如果已知，可在 **sigma** 中指出標準差。如果省略 **sigma**，則會使用實際的樣本標準差。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
ZTestw_dif( Weight, Value-TestValue)
```

另請參見：

z-test 函數使用方式範例 (page 491)

ZTestw_sterr

ZTestw_sterr() 會針對一組值，傳回彙總的 z 檢定平均差的標準誤差。

此函數適用於 z 檢定，其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
ZTestw_sterr (weight, value [, sigma])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
value	值應會透過 **value** 傳回。假設樣本平均數為 0。若要對另一個平均數執行檢定，可從樣本值減去該值。
weight	**value** 中的各個樣本值可根據 **weight** 中的對應加權值，計數一次或多次。
sigma	如果已知，可在 **sigma** 中指出標準差。如果省略 **sigma**，則會使用實際的樣本標準差。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
ZTestw_sterr( Weight, Value-TestValue)
```

另請參見：

z-test 函數使用方式範例 (page 491)

ZTestw_conf

ZTestw_conf() 針對一組值傳回彙總 z 信賴區間值。

此函數適用於 z 檢定，其中輸入資料序列已指定為雙資料行加權的格式。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
ZTest_conf(weight, value[, sigma[, sig]])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
value	即會評估樣本值。假設母體平均數為 0。若要對另一個平均數執行檢定，可從樣本值減去該平均數。
weight	**value** 中的各個樣本值可根據 **weight** 中的對應加權值，計數一次或多次。
sigma	如果已知，可在 **sigma** 中指出標準差。如果省略 **sigma**，則會使用實際的樣本標準差。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略，**sig** 會設定為 0.025，產生 95% 信賴區間。

限制：

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例：

```
ZTestw_conf( Weight, Value-TestValue)
```

另請參見：

z-test 函數使用方式範例 (page 491)

ZTestw_lower

ZTestw_lower() 會針對兩組獨立值，傳回信賴區間下端的彙總值。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法：

```
ZTestw_lower (grp, value [, sig [, eq_var]])
```

傳回的資料類型: 數值

引數:

引數	描述
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱，則該欄位將自動命名為 **Value**。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱，則該欄位將自動命名為 **Type**。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略，**sig** 會設定為 0.025，產生 95% 信賴區間。
eq_var	如果 **eq_var** 指定為 False (0)，則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1)，則假設樣本之間有相等的變異。

限制:

運算式值中若有文字值、NULL 值及遺漏值，會造成函數傳回 NULL。

範例:

```
ZTestw_lower( Group, Value )
ZTestw_lower( Group, Value, sig, false )
```

另請參見:

z-test 函數使用方式範例 (page 491)

ZTestw_upper

ZTestw_upper() 會針對兩組獨立值，傳回信賴區間上端的彙總值。

此函數適用於獨立樣本學生 t 檢定。

如果該函數在資料載入指令碼中使用，則值會在由 group by 子句所定義的若干記錄上反覆運算。

如果函數在圖表運算式中使用，則值會在圖表維度上反覆運算。

語法:

```
ZTestw_upper (grp, value [, sig [, eq_var]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
value	即會評估樣本值。樣本值必須按 **group** 中正好兩個值所指定的邏輯方式分組。如果載入指令碼中未提供樣本值的欄位名稱,則該欄位將自動命名為 **Value**。
grp	包含兩個樣本群組中每一個的名稱的欄位如果載入指令碼中未提供群組的欄位名稱,則該欄位將自動命名為 **Type**。
sig	可在 **sig** 中指定雙尾顯著性水平。如果省略,**sig** 會設定為 0.025,產生 95% 信賴區間。
eq_var	如果 **eq_var** 指定為 False (0),則會假設兩個樣本有不同的變異。如果 **eq_var** 指定為 True (1),則假設樣本之間有相等的變異。

限制:

運算式值中若有文字值、NULL 值及遺漏值,會造成函數傳回 NULL。

範例:

```
ZTestw_upper( Group, Value )
ZTestw_upper( Group, Value, sig, false )
```

另請參見:

z-test 函數使用方式範例 (page 491)

統計檢定函數範例

本節包含套用至圖表和資料載入指令碼的統計檢定函數範例。

圖表中 chi2-test 函數的使用方式範例

chi2-test 函數可用來得出與卡方統計分析相關的值。

本節說明如何使用樣本資料建置視覺化,進而得出 Qlik Sense 中可用卡方分布檢定函數的值。請參閱個別 chi2-test 圖表函數主題,取得語法及引數的說明。

載入樣本資料

共有三組樣本資料,描述三種要載入指令碼中的不同統計樣本。

請執行下列動作:

1. 建立新應用程式。
2. 在資料載入中,輸入以下內容:

```
// 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. 按一下 以載入資料。

建立 chi2-test 圖表函數視覺化

範例：樣本 1

請執行下列動作：

1. 在資料載入編輯器中，按一下 以轉至應用程式檢視，然後按一下您之前建立的工作表。
   隨即開啟工作表檢視。
2. 按一下 **編輯工作表**以編輯工作表。
3. 從**圖表**中新增一個表格，並從**欄位**中新增 Grp、Grade 和 Count 作為維度。
   此表格顯示樣本資料。
4. 新增含有以下運算式的另一個表格作為維度。
   `ValueList('p','df','Chi2')`
   使用合成維度函數來建立含有三個 chi2-test 函數名稱的維度標籤。
5. 將以下運算式作為量值新增至表格。
   ```
   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)))
   ```
   這會產生將各個 chi2-test 的結果值放在其相關合成維度旁表格中的效果。
6. 將量值的**數字格式設定**設定為**數字**和 **3** 個**有效數字**。

在量值的運算式中，應改用以下運算式：`Pick(Match(ValueList('p','df','Chi2'),'p','df','Chi2'),Chi2Test_p(Grp,Grade,Count),Chi2Test_df(Grp,Grade,Count),Chi2Test_Chi2(Grp,Grade,Count))`

結果：

針對樣本 1 資料產生的 chi2-test 函數表格會包含以下值：

結果表格

p	df	Chi2
0.820	5	2.21

範例：樣本 2

請執行下列動作：

1. 在您於範例樣本 1 中編輯的工作表裡，從 **圖表**中新增表格，並從**欄位**中新增 Sex、Opinion 和 OpCount 作為維度。
2. 使用**複製**和**貼上**命令，複製樣本 1 的結果表格。在量值中編輯運算式，並用樣本 2 資料中使用的欄位名稱取代全部三個 chi2-test 函數中的引數，例如：`Chi2Test_p(Sex,Opinion,OpCount)`。

結果：

針對樣本 2 資料產生的 chi2-test 函數表格會包含以下值：

結果表格

p	df	Chi2
0.000309	2	16.2

範例：樣本 3

請執行下列動作：

1. 依照樣本 1 和樣本 2 資料範例的同樣方式，再建立兩個表格。在維度表格中，使用以下欄位作為維度：Gender、Description、Actual 和 Expected。
2. 在結果表格中，使用樣本 3 資料中所用的欄位名稱，例如：`Chi2Test_p (Gender,Description,Actual,Expected)`。

結果：

針對樣本 3 資料產生的 chi2-test 函數表格會包含以下值：

結果表格

p	df	Chi2
0.000308	2	16.2

資料載入指令碼中 chi2-test 函數的使用方式範例

chi2-test 函數可用來得出與卡方統計分析相關的值。本節描述如何使用資料載入指令碼的 Qlik Sense 中提供的卡方分佈檢定函數。請參閱各別 chi2-test 指令碼函數主題，取得語法及引數的說明。

此範例使用一個表格，其中包含兩組學生 (I 和 II) 中達到某個等級 (A-F) 的學生數量。

Data table

Group	A	B	C	D	E	F
I	15	7	9	20	26	19
II	10	11	7	15	21	16

載入樣本資料

請執行下列動作：

1. 建立新應用程式。
2. 在資料載入編輯器中，輸入以下內容：
```
// 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. 按一下 以載入資料。

現在，您已載入樣本資料。

載入 chi2-test 函數值

現在，我們將根據新表格中的樣本資料載入 chi2-test 值，依 Grp 進行分組。

請執行下列動作：

1. 在資料載入編輯器中，在指令碼結束處新增以下內容：

```
// 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. 按一下 以載入資料。

現在，您已在名為 Chi2_table 的表格中載入 chi2-test 值。

結果

您可以在**預覽**下的資料模型檢視器中檢視產生的 chi2-test 值，它們應該如下所示：

Results

Grp	chi2	df	p
I	16.00	5	0.007
II	9.40	5	0.094

建立一般 t-test 報表

一般的學生 t-test 報表可能包含具有 **Group Statistics** 和 **Independent Samples Test** 結果的表格。

以下章節中將使用套用到兩個獨立樣本群組 Observation 和 Comparison 的 Qlik Senset-test 函數，進而建置三個表格。這些樣本的對應表格看起來應像這樣：

群組統計資料

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

獨立樣本測試

Type	conf	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval (Lower)	95% Confidence Interval (Upper)
Equal Variance not Assumed	0	3.534	37.116717335823	0.001	15.2	4.30101	6.48625	23.9137
Equal Variance Assumed	8.706939	3.534	38	0.001	15.2	4.30101	6.49306	23.9069

載入樣本資料

請執行下列動作：

1. 使用新工作表建立新應用程式。
2. 在資料載入編輯器中輸入以下內容：

```
Table1:
Crosstable (Type, Value)
Load recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2 ] (delimiter is '|');
```

在此載入指令碼中包括 **recno()**，因為 **crosstable** 需要引數。因此，**recno()** 只需提供額外的引數，在此情況下，是為每列提供一個 ID。如果未提供，則不會載入 **Comparison** 樣本值。

3. 按一下 以載入資料。

建立 Group statistics 表格

請執行下列動作：

1. 在資料載入編輯器中，按一下 以轉至應用程式檢視，然後按一下您之前建立的工作表。隨即開啟工作表檢視。
2. 按一下 **編輯工作表**以編輯工作表。
3. 從**圖表**中新增一個表格，並從**欄位**中將 Type 作為維度新增至表格。
4. 新增以下運算式作為量值。

範例運算式

標籤	運算式
N	Count(Value)
Mean	Avg(Value)
Standard Deviation	Stdev(Value)
Standard Error Mean	Sterr(Value)

5. 按一下**排序**，並確認 Type 位於排序清單的頂部。

結果：

這些樣本的 Group statistics 表格看起來應像這樣：

群組統計資料

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

建立 Independent sample test 表格

請執行下列動作：

1. 按一下 **編輯工作表**以編輯工作表。
2. 從**圖表**將具有下列運算式的表格作為維度新增至表格。`=ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1))` 並提供標籤類型。
3. 新增以下運算式作為量值：

範例運算式

標籤	運算式
conf	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_conf(Type, Value),TTest_conf(Type, Value, 0))
t	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_t(Type, Value),TTest_t(Type, Value, 0))
df	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_df(Type, Value),TTest_df(Type, Value, 0))
Sig. (2-tailed)	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_sig(Type, Value),TTest_sig(Type, Value, 0))
Mean Difference	TTest_dif(Type, Value)
Standard Error Difference	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_sterr(Type, Value),TTest_sterr(Type, Value, 0))
95% Confidence Interval (Lower)	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_lower(Type, Value,(1-(95)/100)/2),TTest_lower(Type, Value,(1-(95)/100)/2, 0))
95% Confidence Interval (Upper)	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_upper(Type, Value,(1-(95)/100)/2),TTest_upper(Type, Value,(1-(95)/100)/2, 0))

結果：

獨立樣本測試

Type	conf	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval (Lower)	95% Confidence Interval (Upper)
Equal Variance not Assumed	0	3.534	37.116717335823	0.001	15.2	4.30101	6.48625	23.9137
Equal Variance Assumed	8.706939	3.534	38	0.001	15.2	4.30101	6.49306	23.9069

z-test 函數使用方式範例

z-test 函數可用來針對較大型資料樣本 (通常大於 30 且已知變異數) 得出與 z-test 統計分析相關的值。

本節說明如何使用樣本資料建置視覺化，進而得出 Qlik Sense 中可用 z-test 函數的值。請參閱個別 z-test 圖表函數主題，取得語法及引數的說明。

載入樣本資料

這裡使用的樣本資料與 t-test 函數範例中使用的一樣。該樣本資料大小通常對 Z 檢定分析來說太小，不過已足夠用來說明 Qlik Sense 中不同的 z-test 函數用法。

請執行下列動作：

1. 使用新工作表建立新應用程式。

   若已針對 t-test 函數建立應用程式，您可使用該應用程式並為這些函數建立新的工作表。

2. 在資料載入編輯器中，輸入以下內容：
```
Table1:
Crosstable (Type, Value)
Load recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
```

```
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2 ] (delimiter is '|');
```

在此載入指令碼中包括 **recno()**,因為 **crosstable** 需要引數。因此,**recno()** 只需提供額外的引數,在此情況下,是為每列提供一個 ID。如果未提供,則不會載入 **Comparison** 樣本值。

3. 按一下 以載入資料。

建立 z-test 表格

請執行下列動作:

1. 在資料載入編輯器中,按一下 以轉至應用程式檢視,然後按一下上述建立的工作表。
   隨即開啟工作表檢視。
2. 按一下 **編輯工作表**以編輯工作表。
3. 從**圖表**中新增一個表格,並從**欄位**中新增 Type 作為維度。
4. 將以下運算式作為量值新增至表格

範例運算式

標籤	**運算式**
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)

*您可能會想調整量值的數字格式設定,以便查看更具意義的值。若您將絕大多數量值的數字格式設為 **Number>Simple**,而不是 **Auto**,這會令表格更易於閱讀。但例如對於 ZTest Sig,請使用數字格式:**Custom**,然後將格式模式調整為 **#.######**。*

結果:

針對樣本資料產生的 z-test 函數表格會包含以下值:

z-test 結果表格

Type	ZTest Conf	ZTest Dif	ZTest Sig	ZTest Sterr	ZTest Z
Comparison	6.40	11.95	0.000123	3.27	3.66
Observation	5.48	27.15	0.000000	2.80	9.71

建立 z-testw 表格

z-testw 函數適合用於輸入資料序列為加權雙資料行格式時。運算式需要一個引數 weight 的值。

此處的範例皆使用值 2，不過您可使用能針對各觀察項目定義 weight 值的運算式。

請執行下列動作：

1. 在資料載入編輯器中，按一下 以轉至應用程式檢視，然後按一下上述建立的工作表。
隨即開啟工作表檢視。
2. 按一下 **編輯工作表** 以編輯工作表。
3. 從**圖表**中新增一個表格，並從**欄位**中新增 Type 作為維度。
4. 將以下運算式作為量值新增至表格：

範例運算式

標籤	運算式
ZTestw Conf	ZTestw_conf(2,Value)
ZTestw Dif	ZTestw_dif(2,Value)
ZTestw Sig	ZTestw_sig(2,Value)
ZTestw Sterr	ZTestw_sterr(2,Value)
ZTestw Z	ZTestw_z(2,Value)

使用與 z-test 函數範例中相同的數字格式。

結果：

產生的 z-testw 函數表格會包含以下值：

z-testw 結果表格

Type	ZTestw Conf	ZTestw Dif	ZTestw Sig	ZTestw Sterr	ZTestw Z
Comparison	4.47	11.95	8.037185e-08	2.28	5.24
Observation	3.83	27.15	0	1.95	13.91

字串彙總函數

本節描述與字串相關的彙總函數。

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

資料載入指令碼中的字串彙總函數

Concat

Concat() 用來組合字串值。該指令碼函數會針對由 **group by** 子句所定義之若干記錄上反覆運算的所有運算式值，傳回彙總字串串連。

```
Concat ([ distinct ] expression [, delimiter [, sort-weight]])
```

FirstValue

FirstValue() 會傳回首先從運算式定義之記錄中載入的值，按 **group by** 子句排序。

此函數只可作為指令碼函數。

```
FirstValue  (expression)
```

LastValue

LastValue() 會傳回最後從運算式定義之記錄中載入的值，按 **group by** 子句排序。

此函數只可作為指令碼函數。

```
LastValue  (expression)
```

MaxString

MaxString() 會尋找運算式中的字串值，並傳回在若干記錄上按字母排序的最後一個文字值，如 **group by** 子句所定義。

```
MaxString (expression )
```

MinString

MinString() 會尋找運算式中的字串值，並傳回在若干記錄上按字母排序的第一個文字值，如 **group by** 子句所定義。

```
MinString (expression )
```

圖表中的字串彙總函數

下列圖表函數可用於圖表中的彙總字串

Concat

Concat() 用來組合字串值。該函數傳回對每個維度評估之運算式的所有值的彙總字串串連。

```
Concat - 圖表函數({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]] string[,
delimiter[, sort_weight]])
```

MaxString

MaxString() 尋找運算式或欄位中的字串值，並傳回按字母排序順序中的最後一個文字值。

```
MaxString - 圖表函數({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MinString

MinString() 尋找運算式或欄位中的字串值，並傳回按字母排序順序中的第一個文字值。

MinString - 圖表函數({[SetExpression] [**TOTAL** [<fld {, fld}>]]} expr**)**

Concat

Concat() 用來組合字串值。該指令碼函數會針對由 **group by** 子句所定義之若干記錄上反覆運算的所有運算式值，傳回彙總字串串連。

語法：

Concat ([distinct] string [, delimiter [, sort-weight]]**)**

傳回的資料類型： 字串

引數：

包含待處理之字串的運算式或欄位。

引數

引數	描述
string	包含待處理之字串的運算式或欄位。
delimiter	可用在 delimiter 找到的字串來將各個值隔開。
sort-weight	可以透過維度 **sort-weight** 的值來判定串連的順序，如果呈現，則對應於最低值的字串出現在串連的第一位.
distinct	若 **distinct** 一字出現在運算式之前，會忽略所有的重複值。

範例與結果：

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

範例與結果

範例	結果	曾新增至工作表的結果

`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

假設 **TeamData** 表格已如前一個範例中所示載入： `LOAD SalesGroup,Concat(distinct Team,'-') as TeamConcat2 Resident TeamData Group By SalesGroup;`	SalesGroup East West	TeamConcat2 Alpha-Beta-Delta-Gamma Epsilon-Eta-Theta-Zeta
假設 **TeamData** 表格已如前一個範例中所示載入。因為已新增 **sort-weight** 的引數，所以結果會按照維度 Amount 的值排序： `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 - 圖表函數

Concat() 用來組合字串值。該函數傳回對每個維度評估之運算式的所有值的彙總字串串連。

語法：

```
Concat({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} string[, delimiter
[, sort_weight]])
```

傳回的資料類型: 字串

引數:

引數

引數	描述
string	包含待處理之字串的運算式或欄位。
delimiter	可用在 delimiter 找到的字串來將各個值隔開。
sort-weight	可以透過維度 **sort-weight** 的值來判定串連的順序, 如果呈現, 則對應於最低值的字串出現在串連的第一位.
SetExpression	依預設, 彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 **DISTINCT** 這個字出現在函數引數之前, 會忽略評估函數引數所產生的重複項目。
TOTAL	如果單字 **TOTAL** 位於函數引數之前, 則會在提供現行選項的所有可能值上進行計算, 而不僅僅是關於目前維度值的那些選項, 也就是說, 它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單), 您可建立總可能值的子集。

範例與結果:

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

函數範例

範例	結果
`Concat(Team)`	表格是從維度 SalesGroup 和 Amount 以及量值 Concat(Team) 的變異數建立而成。請注意，若忽略總計結果，即使有八個 Team 值的資料分布在兩個 SalesGroup 值之間，串連表格中多個 Team 字串值的量值 Concat(Team) 的唯一結果，是含有維度 Amount 20000 的列，而該列提供結果 BetaGammaGamma。這是因為輸入資料中 Amount 20000 有三個值。當量值跨越維度，則所有其他結果會維持未串連，因為每個 SalesGroup 和 Amount 組合只有一個 Team 值。
`Concat (DISTINCT Team,', ')`	Beta, Gamma。因為 DISTINCT 限定詞表示重複的 Gamma 結果會遭忽略。還有，分隔符號引數定義為緊接著一個空格的逗號。
`Concat (TOTAL <SalesGroup> Team)`	若使用 TOTAL 限定詞，所有 Team 值的所有字串值皆會串連。若指定欄位選項 <SalesGroup>，會將結果分隔為兩個 SalesGroup 維度值。針對 SalesGroupEast，結果為 AlphaBetaDeltaGammaGamma。針對 SalesGroupWest，結果為 EpsilonEtaThetaZeta。
`Concat (TOTAL <SalesGroup> Team,';', Amount)`	透過新增 **sort-weight** 的引數：Amount，結果會按照維度 Amount 的值排序。結果會變成 DeltaBetaGammaGammaAlpha 和 EtaEpsilonZEtaTheta。

範例中使用的資料：

```
TeamData:
LOAD * inline [
SalesGroup|Team|Date|Amount
East|Gamma|01/05/2013|20000
East|Gamma|02/05/2013|20000
West|Zeta|01/06/2013|19000
East|Alpha|01/07/2013|25000
East|Delta|01/08/2013|14000
West|Epsilon|01/09/2013|17000
West|Eta|01/10/2013|14000
East|Beta|01/11/2013|20000
West|Theta|01/12/2013|23000
] (delimiter is '|');
```

FirstValue

FirstValue() 會傳回首先從運算式定義之記錄中載入的值，按 **group by** 子句排序。

此函數只可作為指令碼函數。

語法：

```
FirstValue ( expr)
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。

限制:

如果找不到文字值,則會傳回 NULL。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。若要查看結果,將結果資料行中列出的欄位新增至您應用程式中的工作表。

結果資料

範例	結果	工作表上的結果
TeamData: LOAD * inline [SalesGroup|Team|Date|Amount East|Gamma|01/05/2013|20000 East|Gamma|02/05/2013|20000 West|Zeta|01/06/2013|19000 East|Alpha|01/07/2013|25000 East|Delta|01/08/2013|14000 West|Epsilon|01/09/2013|17000 West|Eta|01/10/2013|14000 East|Beta|01/11/2013|20000 West|Theta|01/12/2013|23000] (delimiter is '|'); FirstValue1: LOAD SalesGroup,FirstValue(Team) as FirstTeamLoaded Resident TeamData Group By SalesGroup;	SalesGroup East West	FirstTeamLoaded Gamma Zeta

LastValue

LastValue() 會傳回最後從運算式定義之記錄中載入的值,按 **group by** 子句排序。

此函數只可作為指令碼函數。

語法:

```
LastValue ( expr )
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。

限制:

如果找不到文字值,則會傳回 NULL。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。然後,至少將結果資料行中列出的欄位新增至我們應用程式中的工作表以查看結果。

若要取得與下方結果資料行中相同的外觀,請在屬性面板的「排序」下方,從「自動」切換至「自訂」,然後取消選取按數字和字母排序。

範例	結果	使用自訂排序的結果
TeamData: LOAD * inline [SalesGroup|Team|Date|Amount East|Gamma|01/05/2013|20000 East|Gamma|02/05/2013|20000 West|Zeta|01/06/2013|19000 East|Alpha|01/07/2013|25000 East|Delta|01/08/2013|14000 West|Epsilon|01/09/2013|17000 West|Eta|01/10/2013|14000 East|Beta|01/11/2013|20000 West|Theta|01/12/2013|23000] (delimiter is '|'); LastValue1: LOAD SalesGroup,LastValue(Team) as LastTeamLoaded Resident TeamData Group By SalesGroup;	SalesGroup East West	LastTeamLoaded Beta Theta

MaxString

MaxString() 會尋找運算式中的字串值,並傳回在若干記錄上按字母排序的最後一個文字值,如 **group by** 子句所定義。

語法:

```
MaxString ( expr )
```

傳回的資料類型: 雙值

引數:

引數	描述
expr	包含待測量之資料的運算式或欄位。

限制:

如果找不到文字值,則會傳回 NULL。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。若要查看結果,將結果資料行中列出的欄位新增至您應用程式中的工作表。

<table>
<tr><th>範例</th><th>結果</th><th></th></tr>
<tr><td><pre>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;</pre></td><td>SalesGroup

East

West</td><td>MaxString1

Gamma

Zeta</td></tr>
<tr><td>假設 TeamData 表格已如前一個範例中所示載入,並且您的資料載入指令碼具有 SET 陳述式:<pre>SET DateFormat='DD/MM/YYYY';':

LOAD SalesGroup,MaxString(Date) as MaxString2 Resident TeamData
Group By SalesGroup;</pre></td><td>SalesGroup

East

West</td><td>MaxString2

01/11/2013

01/12/2013</td></tr>
</table>

MaxString - 圖表函數

MaxString() 尋找運算式或欄位中的字串值,並傳回按字母排序順序中的最後一個文字值。

語法:

```
MaxString({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
SetExpression	依預設,彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
TOTAL	如果單字 **TOTAL** 位於函數引數之前,則會在提供現行選項的所有可能值上進行計算,而不僅僅是關於目前維度值的那些選項,也就是說,它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單),您可建立總可能值的子集。

限制:

若運算式沒有包含任何含有字串表示法的值,則會傳回 NULL。

範例與結果:

結果表格

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

函數範例

範例	結果
`MaxString (Team)`	維度 Amount 共有三個 20000 值:兩個屬於 Gamma (分屬不同日期),一個屬於 Beta。因此量值 MaxString (Team) 的結果是 Gamma,因為這是已排序字串中最高的值。
`MaxString (Date)`	2013/11/01 是三個與維度 Amount 相關的 Date 值中最大的一個。這裡假設您的指令碼具有 SET 陳述式 `SET DateFormat='YYYY-MM-DD';`

範例中使用的資料:

```
TeamData:
LOAD * inline [
SalesGroup|Team|Date|Amount
East|Gamma|01/05/2013|20000
East|Gamma|02/05/2013|20000
West|Zeta|01/06/2013|19000
East|Alpha|01/07/2013|25000
East|Delta|01/08/2013|14000
West|Epsilon|01/09/2013|17000
West|Eta|01/10/2013|14000
East|Beta|01/11/2013|20000
West|Theta|01/12/2013|23000
] (delimiter is '|');
```

MinString

MinString() 會尋找運算式中的字串值，並傳回在若干記錄上按字母排序的第一個文字值，如 **group by** 子句所定義。

語法:

```
MinString ( expr )
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。

限制:

如果找不到文字值，則會傳回 NULL。

範例與結果:

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

結果資料

範例	結果	

`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

假設 **TeamData** 表格已如前一個範例中所示載入, 並且您的資料載入指令碼具有 SET 陳述式:
`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 - 圖表函數

MinString() 尋找運算式或欄位中的字串值, 並傳回按字母排序順序中的第一個文字值。

語法:

```
MinString({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
SetExpression	依預設, 彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。

引數	描述
TOTAL	如果單字 **TOTAL** 位於函數引數之前，則會在提供現行選項的所有可能值上進行計算，而不僅僅是關於目前維度值的那些選項，也就是說，它會忽略圖表維度。 透過使用 **TOTAL [<fld {.fld}>]** (其中 **TOTAL** 限定詞後面是做為圖表維度變數子集之一或更多欄位名稱的清單)，您可建立總可能值的子集。

範例與結果：

樣本資料

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

函數範例

範例	結果
`MinString (Team)`	維度 Amount 共有三個 20000 值：兩個屬於 Gamma (分屬不同日期)，一個屬於 Beta。因此量值 MinString (Team) 的結果是 Beta，因為這是已排序字串中第一個值。
`MinString (Date)`	2013/11/01 是三個與維度 Amount 相關的 Date 值中最早的一個。這裡假設您的指令碼具有 SET 陳述式 `SET DateFormat='YYYY-MM-DD';`

範例中使用的資料：

```
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 '|');
```

合成維度函數

應用程式中的合成維度是從合成維度函數所產生的值所建立，而非直接從資料模型中的欄位建立。當合成維度函數產生的值在圖表中作為計算維度使用，會建立一個合成維度。合成維度可供您建立，舉例來說，包含維度的圖表，而該維度含有從資料所得的值，亦即動態維度。

合成維度不受選項影響。

下列合成維度函數可用於圖表中。

ValueList

ValueList() 傳回一組列出的值，當用於計算維度時，可形成合成維度。

ValueList - 圖表函數 (v1 {, Expression}**)**

ValueLoop

ValueLoop() 傳回一組反覆運算的值，當用於計算維度時，可形成合成維度。

ValueLoop - 圖表函數(from [, to [, step]]**)**

ValueList - 圖表函數

ValueList() 傳回一組列出的值，當用於計算維度時，可形成合成維度。

*圖表中若有以 **ValueList** 函數建立的合成維度，可使用圖表運算式中相同的參數來重新指明 **ValueList** 函數，即可參考對應特定運算式儲存格的維度值。此函數當然可以用於版面配置中的任何地方，但除了可用於合成維度之外，只有在彙總函數內才有意義。*

合成維度不受選項影響。

語法：

ValueList(v1 {,...}**)**

傳回的資料類型：雙值

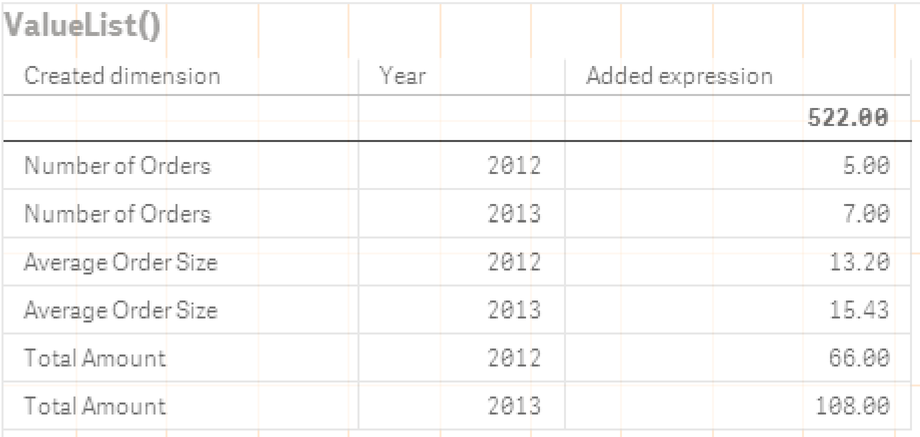
引數：

引數

引數	描述
v1	靜態值 (通常為字串，也可以是數字)。
{,...}	靜態值的選用清單。

範例與結果：

函數範例

<table>
<tr><th>範例</th><th>結果</th></tr>
<tr><td><code>ValueList ('Number of Orders', 'Average Order Size', 'Total Amount')</code></td><td>用來在表格中建立維度時，舉例來說，此函數會產生三個字串值作為表格中的列標籤。這些值可提供運算式參考。</td></tr>
<tr><td><code>=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)))</code></td><td>此運算式使用來自所建立維度的值，並在巢狀 IF 陳述式中將這些值參考為三個彙總函數的輸入：

ValueList()<table><tr><th>Created dimension</th><th>Year</th><th>Added expression</th></tr><tr><td></td><td></td><td>522.00</td></tr><tr><td>Number of Orders</td><td>2012</td><td>5.00</td></tr><tr><td>Number of Orders</td><td>2013</td><td>7.00</td></tr><tr><td>Average Order Size</td><td>2012</td><td>13.20</td></tr><tr><td>Average Order Size</td><td>2013</td><td>15.43</td></tr><tr><td>Total Amount</td><td>2012</td><td>66.00</td></tr><tr><td>Total Amount</td><td>2013</td><td>108.00</td></tr></table></td></tr>
</table>

範例中使用的資料：

```
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 - 圖表函數

ValueLoop() 傳回一組反覆運算的值，當用於計算維度時，可形成合成維度。
產生的值會從 **from** 值開始，至 **to** 值結束，包括以步階遞增的中間值。

*圖表中若有以 **ValueLoop** 函數建立的合成維度，可使用圖表運算式中相同的參數來重新指明 **ValueLoop** 函數，即可參考對應特定運算式儲存格的維度值。此函數當然可以用於版面配置中的任何地方，但除了可用於合成維度之外，只有在彙總函數內才有意義。*

合成維度不受選項影響。

語法：

```
ValueLoop(from [, to [, step ]])
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
from	要產生之一組值的開始值。
to	要產生之一組值的結束值。
step	值之間的遞增大小。

範例與結果：

函數範例

範例	結果
`ValueLoop (1, 10)`	這會在表格中建立一個維度，例如可用於編號標籤等用途的維度。這裡的範例產生編號 1 到 10 的值。這些值可提供運算式參考。
`ValueLoop (2, 10,2)`	此範例產生編號 2、4、6、8 和 10 的值，因為引數 step 的值為 2。

巢狀彙總

您可能會遇到需在將某彙總套用到另一個彙總結果的情況。此情況就稱為巢狀彙總。

您無法在大部分的圖表運算式中巢狀化彙總。不過，若您在內部彙總函數中使用 **TOTAL** 限定詞，可以巢狀化彙總。

不允許超過 100 層的巢狀結構。

含有 TOTAL 限定詞的巢狀彙總

範例:

您要計算欄位 **Sales** 的加總,但是只要包含 **OrderDate** 等於去年的交易。透過彙總函數 **Max(TOTAL** Year(OrderDate)**)** 即可取得去年的交易。

以下彙總會傳回所需結果:

```
Sum(If(Year(OrderDate)=Max(TOTAL Year(OrderDate)), Sales))
```

Qlik Sense 要求在此巢狀類型納入 **TOTAL** 限定詞。對於所需比較有此必要。這類型的巢狀結構需求相當普遍且是個好方法。

另請參見:

Aggr - 圖表函數 (page 509)

8.3 Aggr - 圖表函數

Aggr() 會傳回對所說明維度計算之運算式的值陣列。例如,每個地區,每個客戶的銷售額的最大值。

Aggr 函數用於巢狀彙總,其中每個維度值會計算一次第一個參數 (內部彙總)。會在第二個參數 (和後續參數) 指定維度。

此外,**Aggr** 函數應含括在外部彙總函數中,使用 **Aggr** 函數的結果陣列作為其巢狀所在之彙總的輸入。

語法:

```
Aggr({SetExpression}[DISTINCT] [NODISTINCT ] expr, StructuredParameter{, StructuredParameter})
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
expr	運算式包含一個彙總函數。依預設,彙總函數將彙總選項所定義的一組可能記錄。

引數	描述
StructuredParameter	StructuredParameter 由維度和該格式中的排序條件 (選用) 組成：`(Dimension(Sort-type, Ordering))` 該維度是單一欄位且不能是運算式。維度用於確定 Aggr 運算式所計算的值陣列。 如果採用排序條件，則要對 Aggr 函數建立的值陣列 (為維度而計算) 進行排序。當排序順序影響含括 Aggr 函數的運算式的結果時，這一點很重要。 如需瞭解如何使用排序條件的詳情，請參見新增排序條件至結構化參數的維度中。
SetExpression	依預設，彙總函數將彙總選項所定義的一組可能記錄。集合分析運算式可定義一組替代的記錄。
DISTINCT	如果 expression 引數前面有 **distinct** 限定詞，或者完全不使用任何限定詞，則維度值的各個相異組合只會產生一個傳回值。這是製作彙總的一般方式；每個維度值的相異組合都會呈現在圖表中的一條線上。
NODISTINCT	如果 expression 引數前面有 **nodistinct** 限定詞，則視基礎資料結構而定，維度值的各個組合可能會產生多個傳回值。如果只有一個維度，**aggr** 函數會傳回與來源資料中的列具有相同元素數的陣列。

Sum、**Min** 和 **Avg** 等基本彙總函數會傳回單一數值，而 Aggr() 函數可加以比較，藉此建立一個暫存結果集 (虛擬表格)，在該結果集的基礎上可進行其他彙總。例如，在 **Aggr()** 陳述式中按客戶加總銷售額來計算平均銷售值，然後計算加總結果的平均值：**Avg(TOTAL Aggr(Sum(Sales),Customer))**。

若要在多個層級中建立巢狀圖表彙總，請在計算維度中使用 Aggr() 函數。

限制：

在 Aggr() 函數中的每個維度都必須是單一欄位，且不能為運算式 (計算維度)。

新增排序條件至結構化參數的維度中

在其基本形式中，Aggr 函數語法中的引數 StructuredParameter 是單一維度。運算式：Aggr(Sum(Sales, Month)) 找到每個月的總銷售值。但是，如果含括在另一個彙總函數中，除非使用排序條件，否則會產生意料之外的結果。這是因為一些維度可能按數值或字母順序等排序。

在 Aggr 函數的 StructuredParameter 引數中，您可指定運算式中維度的排序條件。如此一來，您便在 Aggr 函數產生的虛擬表格中強制使用了排序順序。

引數 StructuredParameter 具有以下語法：

```
(FieldName, (Sort-type, Ordering))
```

結構化參數可構成巢狀：

```
(FieldName, (FieldName2, (Sort-type, Ordering)))
```

排序類型可以是：NUMERIC、TEXT、FREQUENCY 或 LOAD_ORDER。

與每個排序類型關聯的順序類型如下所示：

允許的順序類型

排序類型	**允許的順序類型**
NUMERIC	ASCENDING、DESCENDING 或 REVERSE
TEXT	ASCENDING、A2Z、DESCENDING、REVERSE 或 Z2A
FREQUENCY	DESCENDING、REVERSE 或 ASCENDING
LOAD_ORDER	ASCENDING、ORIGINAL、DESCENDING 或 REVERSE

順序類型 REVERSE 與 DESCENDING 相同。

對於排序類型 TEXT，順序類型 ASCENDING 和 A2Z 是相同的，而 DESCENDING、REVERSE 和 Z2A 也是相同的。

對於排序類型 LOAD_ORDER，順序類型 ASCENDING 和 ORIGINAL 是相同的。

範例：使用彙總的圖表運算式

範例 - 圖表運算式

圖表運算式範例 1

載入指令碼

在資料載入編輯器中載入下列資料作為內嵌載入，以建立以下的圖表運算式範例。

```
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 '|');
```

圖表運算式

在 Qlik Sense 工作表中建立 KPI 視覺化。將以下運算式作為量值新增至 KPI：

```
Avg(Aggr(Sum(UnitSales*UnitPrice), Customer))
```

結果

376.7

解釋

運算式 `Aggr(Sum(UnitSales*UnitPrice), Customer)` 依 **Customer** 找到總銷售值，並傳回值的陣列：295、715 和 120，用於三個 **Customer** 值。

我們已有效建立各個值的臨時清單，而不必建立包含這些值的明確表格或資料行。

這些值作為 **Avg()** 函數的輸入使用，進而得到銷售額平均值 376.7。

圖表運算式範例 2

載入指令碼

在資料載入編輯器中載入下列資料作為內嵌載入，以建立以下的圖表運算式範例。

```
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 '|');
```

圖表運算式

在 Qlik Sense 工作表中建立具有 **Customer**、**Product**、**UnitPrice** 與 **UnitSales** 作為維度的表格視覺化。將以下運算式作為量值新增至表格：

```
Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
```

結果

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

Customer	Product	UnitPrice	UnitSales	Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
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

解釋

值陣列:16、16、15、15、12、12、22、22、25、15、15 與 19。**nodistinct** 限定詞表示陣列對來源資料中的每行都含有一個元素:每個元素對於各個 **Customer** 和 **Product** 都是最大的 **UnitPrice**。

圖表運算式範例 3

載入指令碼

在資料載入編輯器中載入下列資料作為內嵌載入,以建立以下的圖表運算式範例。

```
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;
```

圖表運算式

在 Qlik Sense 工作表中建立具有 **Year** 和 **Month** 作為維度的表格視覺化。將以下運算式作為量值新增至表格:

- `Sum(Sales)`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))` 在表格中以結構化 Aggr() 為標籤。

結果

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

解釋

此範例以時間性遞增順序顯示每年十二個月期間的彙總值,亦即 **Aggr()** 運算式的結構化參數 (數值,遞增) 部分。需要兩個特定維度作為結構化參數:**Year** 和 **Month**,以 (1) **Year** (數值) 和 (2) **Month** (數值) 排序。這兩個維度必須用於表格或圖表視覺化。這對於 **Aggr()** 函數的維度清單是必要內容,以便與視覺化中使用的物件維度對應。

您可以在表格中或在獨立的折線圖中比較這些量值之間的差異:

- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year), (Month) ))`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))`

應可清楚看見只有後面的運算式執行所需的彙總值累積。

另請參見:

🗋 *基本彙總函數 (page 309)*

8.4 色彩函數

這些函數可用於設定和評估圖表物件色彩屬性的相關運算式,也可用於資料載入指令碼。

> *基於向下相容性理由,Qlik Sense 支援色彩函數* ***Color()***、***qliktechblue*** *和* ***qliktechgray***,*不過不建議使用這些函數。*

ARGB

ARGB() 用在運算式中,以設定或評估圖表物件的顏色屬性,其中顏色由紅色元件 **r**、綠色元件 **g** 及藍色元件 **b** 定義,並且 Alpha 係數 (不透明度) 為 **alpha**。

```
ARGB (alpha, r, g, b)
```

HSL

HSL() 用在運算式中,以設定或評估圖表物件的顏色屬性,其中顏色由 0 與 1 之間的 **hue**、**saturation** 及 **luminosity** 值定義。

```
HSL  (hue, saturation, luminosity)
```

RGB

RGB() 會傳回一個整數,這對應至由三個參數定義之色彩的色彩代碼:紅色元件 r、綠色元件 g 和藍色元件 b。這些元件必須有介於 0 和 255 之間的整數值。函數可用於運算式中,以設定或評估圖表物件的色彩屬性。

```
RGB  (r, g, b)
```

Colormix1

Colormix1() 用於運算式,以便根據介於 0 和 1 之間的值傳回雙色漸層的 ARGB 色彩呈現。

```
Colormix1 (Value , ColorZero , ColorOne)
```

Value 為介於 0 和 1 之間的實數。

- 若 Value = 0,會傳回 ColorZero 。
- 若 Value = 1,會傳回 ColorOne 。
- 若 0 < Value < 1,會傳回適當的中間色調。

ColorZero 是要與間隔低端關聯之色彩的有效 RGB 色彩呈現。

ColorOne 是要與間隔高端關聯之色彩的有效 RGB 色彩呈現。

範例:

```
Colormix1(0.5, red(), blue())
```

傳回：

`ARGB(255,64,0,64) (purple)`

Colormix2

Colormix2() 用於運算式，以便根據介於 -1 和 1 之間的值傳回雙色漸層的 ARGB 色彩呈現，且可針對中心 (0) 位置指定中間色。

```
Colormix2 (Value ,ColorMinusOne , ColorOne[ , ColorZero])
```

Value 為介於 -1 和 1 之間的實數。

- 若 Value = -1，會傳回第一個色彩。
- 若 Value = 1，會傳回第二個色彩。
- If -1 < Value< 1 會傳回適當的色彩混合。

ColorMinusOne 是要與間隔低端關聯之色彩的有效 RGB 色彩呈現。

ColorOne 是要與間隔高端關聯之色彩的有效 RGB 色彩呈現。

ColorZero 是要與間隔中心關聯之色彩的選用有效 RGB 色彩呈現。

SysColor

SysColor() 會傳回 Windows 系統色彩 nr 的 ARGB 色彩呈現，其中 nr 相當於 Windows API 函數 **GetSysColor(nr)** 中的參數。

```
SysColor (nr)
```

ColorMapHue

ColorMapHue() 會從色彩圖傳回色彩的 ARGB 值；該色彩圖的 HSV 色彩模式色調元件各異。色彩圖會以紅色開始，然後是黃色、綠色、青綠色、藍色、洋紅色，再回到紅色。x 必須指定為介於 0 和 1 之間的數值。

```
ColorMapHue (x)
```

ColorMapJet

ColorMapJet() 會從色彩圖傳回色彩的 ARGB 值；該色彩圖從藍色開始，然後是青綠色、黃色、橘色，再回到紅色。x 必須指定為介於 0 和 1 之間的數值。

```
ColorMapJet (x)
```

預先定義色彩函數

以下函數可用於預先定義色彩的運算式。每個函數會回一個 RGB 色彩呈現。

若傳回了 ARGB 色彩呈現，則可選用地指定 Alpha 係數的參數。Alpha 係數 0 相當於完全透明，而 Alpha 係數 255 則相當於完全不透明。如果未輸入 alpha 值，則將假設為 255。

預先定義色彩函數

色彩函數	RGB 值
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)

範例與結果:

範例與結果

範例	**結果**
`Blue()`	RGB(0,0,128)
`Blue(128)`	ARGB(128,0,0,128)

ARGB

ARGB() 用在運算式中，以設定或評估圖表物件的顏色屬性，其中顏色由紅色元件 **r**、綠色元件 **g** 及藍色元件 **b** 定義，並且 Alpha 係數 (不透明度) 為 **alpha**。

語法:

```
ARGB(alpha, r, g, b)
```

傳回的資料類型: 雙值

引數:

引數

引數	**描述**
alpha	範圍介於 0 - 255 的透明值。0 相當於完全透明，而 255 則相當於完全不透明。
r, g, b	紅色、綠色及藍色元件值。顏色元件 0 不對應於任何作用，255 對應於全部作用。

所有引數都必須是解析為範圍 0 至 255 中的整數的運算式。

如果解譯數值元件並將其格式設為十六進位標記法，色彩元件的值會更易於查看。例如，淺綠色有數字 4 278 255 360，其十六進位表示法為 FF00FF00。前兩個位置 'FF' (255) 表示 **alpha** 通道。接下來兩個位置 '00' 表示**紅色**的數量，接下來兩個位置 'FF' 表示**綠色**的數量，並且最後兩個位置 '00' 表示**藍色**的數量.

RGB

RGB() 會傳回一個整數，這對應至由三個參數定義之色彩的色彩代碼：紅色元件 r、綠色元件 g 和藍色元件 b。這些元件必須有介於 0 和 255 之間的整數值。函數可用於運算式中，以設定或評估圖表物件的色彩屬性。

語法：

```
RGB (r, g, b)
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
r, g, b	紅色、綠色及藍色元件值。顏色元件 0 不對應於任何作用，255 對應於全部作用。

所有引數都必須是解析為範圍 0 至 255 中的整數的運算式。

如果解譯數值元件並將其格式設為十六進位標記法，色彩元件的值會更易於查看。例如，淺綠色有數字 4 278 255 360，其十六進位表示法為 FF00FF00。前兩個位置 'FF' (255) 表示 **alpha** 通道。在函數 **RGB** 與 **HSL** 中，這一律是 'FF' (不透明)。接下來兩個位置 '00' 表示**紅色**的數量，接下來兩個位置 'FF' 表示**綠色**的數量，並且最後兩個位置 '00' 表示**藍色**的數量.

範例：圖表運算式
此範例將自訂色彩套用至圖表：

此範例中使用的資料：

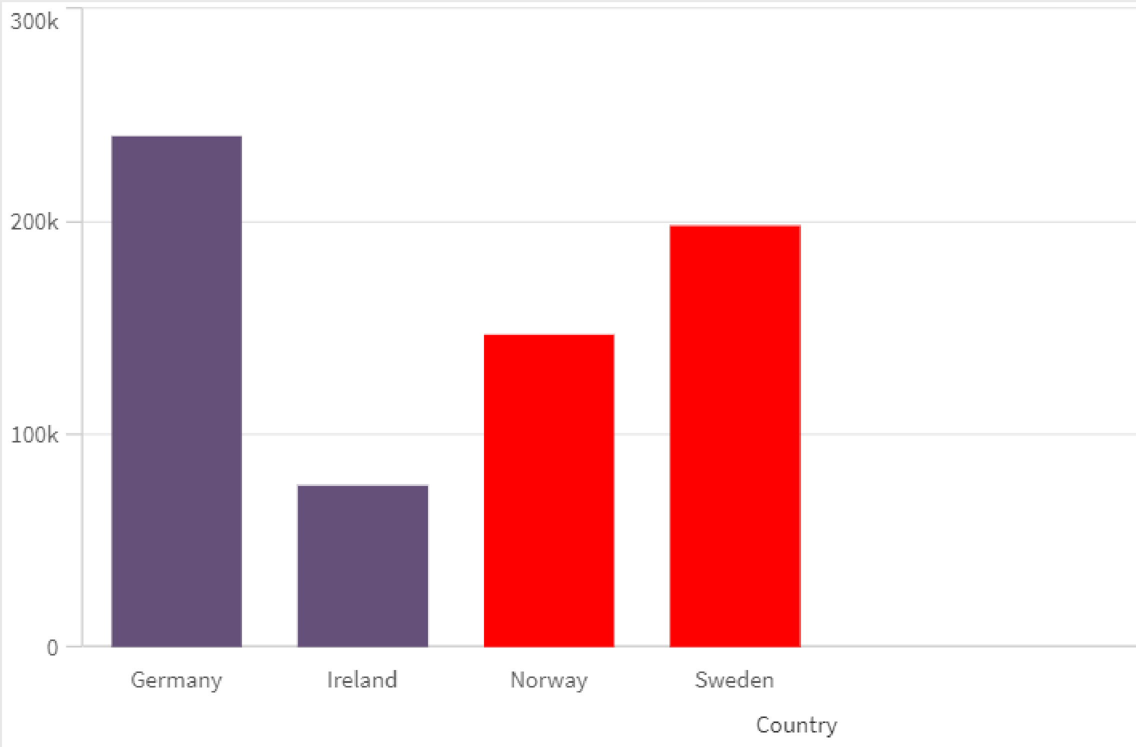
```
ProductSales:
Load * Inline
[Country,Sales,Budget
Sweden,100000,50000
Germany, 125000, 175000
Norway, 74850, 68500
Ireland, 45000, 48000
Sweden,98000,50000
Germany, 115000, 175000
Norway, 71850, 68500
```

```
Ireland, 31000, 48000
] (delimiter is ',');
```

在**色彩和圖例**屬性面板中輸入下列運算式：

```
If (Sum(Sales)>Sum(Budget),RGB(255,0,0),RGB(100,80,120))
```

結果：



範例：載入指令碼

下列範例顯示十六進位格式值的同等 RGB 值：

```
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];
```

結果：

文字	色彩
010203	RGB(1,2,3)
AABBCC	RGB(170,187,204)

HSL

HSL() 用在運算式中，以設定或評估圖表物件的顏色屬性，其中顏色由 0 與 1 之間的 **hue**、**saturation** 及 **luminosity** 值定義。

語法：

```
HSL (hue, saturation, luminosity)
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
hue, saturation, luminosity	介於 0 到 1 的 hue、saturation 和 luminosity 元件值。

所有引數都必須是解析為範圍 0 至 1 中的整數的運算式。

如果解譯數值元件並將其格式設為十六進位標記法，色彩元件的 RGB 值會更易於查看。例如，淺綠色的編號為 4 278 255 360，而其十六進位標記為 FF00FF00 和 RGB (0,255,0)。這等同於 HSL (80/240, 240/240, 120/240)，亦即 (0.33, 1, 0.5) 的 HSL 值。

8.5 條件函數

條件函數都會評估條件，然後根據條件值傳回不同回答。函數可用於資料載入指令碼和圖表運算式。

條件函數概述

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

alt

alt 函數會傳回參數中第一個具有效數字表示法的參數。如果找不到符合的參數，則會傳回最後一個參數。可使用任意數目的參數。

```
alt (expr1[ , expr2 , expr3 , ...] , else)
```

class

class 函數將第一個參數指派給類別間隔。結果是包含 a<=x<b 作為文字值的雙值，其中 a 和 b 是 bin 的上限與下限，並且下限以數值表示。

```
class (expression, interval [ , label [ , offset ]])
```

coalesce

coalesce 函數會傳回參數中第一個具有效 non-NULL 表示法的參數。可使用任意數目的參數。

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

if

if 函數會根據使用函數提供的條件是評估為 True 還是 False，來傳回值。

```
if (condition , then , else)
```

match

match 函數會比較第一個參數與所有後續參數，並傳回相符的運算式數字位置。比較區分大小寫。

```
match ( str, expr1 [ , expr2,...exprN ])
```

mixmatch

mixmatch 函數會比較第一個參數與所有後續參數，並傳回相符的運算式數字位置。比較不區分大小寫。

```
mixmatch ( str, expr1 [ , expr2,...exprN ])
```

pick

pick 函數會傳回清單中的第 *n* 個運算式。

```
pick (n, expr1[ , expr2,...exprN])
```

wildmatch

wildmatch 函數會比較第一個參數與所有後續參數，並傳回相符的運算式數目。這允許在運算式字串中使用萬用字元 (***** 和 **?**)。***** 符合任何字元順序。**?** 符合任何單一字元。比較不區分大小寫。

```
wildmatch ( str, expr1 [ , expr2,...exprN ])
```

alt

alt 函數會傳回參數中第一個具有效數字表示法的參數。如果找不到符合的參數，則會傳回最後一個參數。可使用任意數目的參數。

語法：

```
alt(expr1[ , expr2 , expr3 , ...] , else)
```

引數：

引數

引數	描述
expr1	要檢查數字表示的有效性的第一個運算式。
expr2	要檢查數字表示的有效性的第二個運算式。
expr3	要檢查數字表示的有效性的第三個運算式。
else	在前面參數的數字表示均無效的情況下要傳回的值。

alt 函數通常與數字或日期解譯函數搭配使用。使用此方法，Qlik Sense 可以按照優先順序測試不同的日期格式。它也可以用來處理數值運算式中的 NULL 值。

範例：

範例

範例	結果

`alt( date#( dat , 'YYYY/MM/DD' ),`
 	
`date#( dat , 'MM/DD/YYYY' ),`
 	
`date#( dat , 'MM/DD/YY' ),`
 	
`'No valid date' )`
	此運算式將測試 date 欄位是否包含按照三個指定日期格式中任何一個的日期。如果是，則會傳回一個雙值，其中包含原始字串和日期的有效數字表示法。如果找不到符合的項目，則會傳回 'No valid date' 文字 (並且不會傳回有效的數字表示法)。

`alt(Sales,0) + alt(Margin,0)`	此運算式會新增欄位 Sales 及 Margin，并將所有遺漏值 (NULL) 取代為 0。

class

class 函數將第一個參數指派給類別間隔。結果是包含 a<=x<b 作為文字值的雙值，其中 a 和 b 是 bin 的上限與下限，並且下限以數值表示。

語法：

```
class(expression, interval [ , label [ , offset ]])
```

引數：

引數

引數	描述
interval	指定 bin 寬度的數字。
label	可以取代結果文字中 'x' 的任意字串。
offset	可以用作與預設分類起點偏移的數字。預設起點通常是 0。

範例：

範例

範例	結果
`class( var,10 ), var = 23`	傳回 '20<=x<30'
`class( var,5,'value' ), var = 23`	傳回 '20<= value <25'
`class( var,10,'x',5 ), var = 23`	傳回 '15<=x<25'

範例 - 使用 class 的載入指令碼

範例:載入指令碼

載入指令碼

在此範例中,我們載入包含人員姓名和年齡的表格。我們想要新增一個欄位,根據以十年為間隔的年齡群組分類每一個人員。原始來源表格如下。

結果

Name	Age
John	25
Karen	42
Yoshi	53

若要新增年齡群組分類欄位,您可以使用 **class** 函數,新增前置 load 陳述式。

在資料載入編輯器中建立新的索引標籤,然後載入下列資料作為內嵌載入。在 Qlik Sense 中建立以下表格以查看結果。

```
LOAD *,
class(Age, 10, 'age') As Agegroup;

LOAD * INLINE
[ Age, Name
25, John
42, Karen
53, Yoshi];
```

結果

結果

Name	Age	Agegroup
John	25	20 <= age < 30
Karen	42	40 <= age < 50
Yoshi	53	50 <= age < 60

coalesce

coalesce 函數會傳回參數中第一個具有效 non-NULL 表示法的參數。可使用任意數目的參數。

語法:

coalesce(expr1[, expr2 , expr3 , ...]**)**

引數:

引數

引數	描述
expr1	要檢查非 NULL 表示的有效性的第一個運算式。
expr2	要檢查非 NULL 表示的有效性的第二個運算式。
expr3	要檢查非 NULL 表示的有效性的第三個運算式。

範例:

範例

範例	結果
	此運算式將欄位的所有 NULL 值變更為「不適用」。
`Coalesce(ProductDescription, ProductName, ProductCode, 'no description available')`	若某些欄位可能沒有適用於產品的值,此運算式將會在三個不同的產品描述欄位之間選取。將會傳回給定順序中的第一個欄位,其中有非 NULL 值。若沒有任何欄位包含值,結果將會是「沒有可用描述」。
`Coalesce(TextBetween(FileName, '"', '"'), FileName)`	此運算式將會從欄位 *FileName* 修剪可能加括號的引用。若給定的 *FileName* 被引用,會移除這些內容,並傳回加上括號、未引用的 *FileName*。若 *TextBetween* 函數沒有找到 **Coalesce** 拒絕、傳回 NULL 的分隔符號,會改為傳回原始 *FileName*。

if

if 函數會根據使用函數提供的條件是評估為 True 還是 False,來傳回值。

語法:

```
if(condition , then [, else])
```

引數

引數	描述
condition	邏輯上解譯的運算式。
then	運算式可以是任何類型的。如果 *condition* 是 True,則 if 函數會傳回 *then* 運算式的值。
else	運算式可以是任何類型的。如果 *condition* 是 False,則 if 函數會傳回 *else* 運算式的值。 此參數為選用。若 *condition* 為 False,則會在您未指定 else 的情況下傳回 NULL。

範例

範例	結果
`if( Amount>= 0, 'OK', 'Alarm' )`	此運算式會測試數量是否為正數 (0 或更大), 如果是, 則會傳回 'OK'。如果數量小於 0, 則會傳回 'Alarm'。

範例 - 使用 if 的載入指令碼

範例: 載入指令碼

載入指令碼

If 可在載入指令碼中與其他方法和物件中使用, 包括變數。例如, 如果設定變數 *threshold*, 並且想要根據該臨界值在資料模型中包括欄位, 您可以執行下列項目。

在資料載入編輯器中建立新的索引標籤, 然後載入下列資料作為內嵌載入。在 Qlik Sense 中建立以下表格以查看結果。

```
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;
```

結果

使用在載入指令碼中的 *if* 函數來顯示輸出的 Qlik Sense 表格。

transaction_id	與臨界值比較
3750	少於 100
3751	大於 100
3752	少於 100
3753	大於 100
3754	大於 100
3756	少於 100
3757	大於 100

範例 - 使用 if 的圖表運算式

範例: 圖表運算式

圖表運算式 1

載入指令碼

在資料載入編輯器中建立新的索引標籤, 然後載入下列資料作為內嵌載入。載入資料後, 在 Qlik Sense 表格中建立以下的圖表運算式範例。

```
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];
```

在圖表運算式中顯示 *if* 函數範例的 Qlik Sense 表格。

日期	位置	Incidents	if(Incidents>=10, 'Critical', 'Ok')	if(Incidents>=10, 'Critical', If(Incidents>=1 and Incidents<10, 'Warning', 'Ok'))
1/3/2016	北京	0	確定	確定
1/3/2016	波士頓	12	重大	重大

日期	位置	Incidents	if(Incidents>=10, 'Critical', 'Ok')	if(Incidents>=10, 'Critical', If(Incidents>=1 and Incidents<10, 'Warning', 'Ok'))
1/3/2016	斯德哥爾摩	3	確定	警告
1/3/2016	多倫多	0	確定	確定
1/4/2016	北京	0	確定	確定
1/4/2016	波士頓	8	確定	警告

圖表運算式 2

在新的應用程式中，在資料載入編輯器的新索引標籤中新增下列指令碼，然後載入資料。然後您可以使用以下圖表運算式建立表格。

```
SET FirstWeekDay=0;
Load
Date(MakeDate(2022)+RecNo()-1) as Date
Autogenerate 14;
```

在圖表運算式中顯示 *if* 函數範例的 Qlik Sense 表格。

日期	WeekDay(Date)	If(WeekDay (Date)>=5,'WeekEnd','Normal Day')
1/1/2022	星期六	WeekEnd
1/2/2022	星期日	WeekEnd
1/3/2022	星期一	Normal Day
1/4/2022	星期二	Normal Day
1/5/2022	星期三	Normal Day
1/6/2022	星期四	Normal Day
1/7/2022	星期五	Normal Day
1/8/2022	星期六	WeekEnd
1/9/2022	星期日	WeekEnd
1/10/2022	星期一	Normal Day
1/11/2022	星期二	Normal Day
1/12/2022	星期三	Normal Day
1/13/2022	星期四	Normal Day
1/14/2022	星期五	Normal Day

match

match 函數會比較第一個參數與所有後續參數，並傳回相符的運算式數字位置。比較區分大小寫。

語法：

```
match( str, expr1 [ , expr2,...exprN ])
```

> *如果您想要使用不區分大小寫的比較，請使用 **mixmatch** 函數。如果您想要使用不區分大小寫的比較及萬用字元，請使用 **wildmatch** 函數。*

範例：使用 match 的載入指令碼

範例：載入指令碼

載入指令碼

您可以使用 match 載入資料子集。例如，您可以傳回函數中運算式的數值。然後您可以限制根據數值載入的資料。若沒有相符項目，Match 會傳回 0。在此範例中相符的所有運算式將會因此傳回 0，並且將會按照 WHERE 陳述式從資料載入中排除。

在資料載入編輯器中建立新的索引標籤，然後載入下列資料作為內嵌載入。在 Qlik Sense 中建立以下表格以查看結果。

```
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;
```

結果

使用在載入指令碼中的 match 函數來顯示輸出的 Qlik Sense 表格

色彩代碼藍色和黑色	客戶
黑色	203521
黑色	3036491
藍色	2038593

範例 - 使用 match 的圖表運算式

範例:圖表運算式

圖表運算式 1

載入指令碼

在資料載入編輯器中建立新的索引標籤,然後載入下列資料作為內嵌載入。載入資料後,在 Qlik Sense 表格中建立以下的圖表運算式範例。

```
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];
```

以下表格中的第一個運算式會傳回 0 代表 Stockholm,因為 'Stockholm' 不包括在 **match** 函數的運算式清單中。這也會傳回 0 代表 'Zurich',因為 **match** 比較區分大小寫。

在圖表運算式中顯示 *match* 函數範例的 Qlik Sense 表格

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

圖表運算式 2

您可以使用相符項目，以執行運算式的自訂排序。

按照預設，會按數字或字母排序欄，視資料而定。

Qlik Sense 表格顯示預設排序的範例

Cities
Beijing
Boston
Stockholm
Toronto
zurich

若要變更順序，請進行下列事項：

1. 在**屬性**面板中開啟圖表的**排序**區段。
2. 對於您要進行自訂排序的欄，關閉自動排序。
3. 取消選取**按數字排序**和**按字母排序**。
4. 選取**按運算式排序**，然後輸入類似下列項目的運算式：
   `=match( Cities, 'Toronto','Boston','Beijing','Stockholm','zurich')`
   Cities 欄的排序順序會變更。

Qlik Sense 表格顯示使用 *match* 函數變更排序順序的範例

Cities
多倫多
Boston
Beijing
Stockholm
zurich

您也可以檢視傳回的數值。

Qlik Sense 表格顯示從 *match* 函數傳回的數值範例

Cities	Cities & ' - ' & match (Cities, 'Toronto','Boston', 'Beijing','Stockholm','zurich')
多倫多	多倫多 - 1
Boston	Boston - 2
北京	Beijing - 3
斯德哥爾摩	Stockholm - 4
蘇黎世	zurich - 5

mixmatch

mixmatch 函數會比較第一個參數與所有後續參數，並傳回相符的運算式數字位置。比較不區分大小寫。

語法：

```
mixmatch( str, expr1 [ , expr2,...exprN ])
```

如果您想要改用區分大小寫的比較，請使用 **match** 函數。如果您想要使用不區分大小寫的比較及萬用字元，請使用 **wildmatch** 函數。

範例 - 使用 mixmatch 的載入指令碼

範例：載入指令碼

載入指令碼

您可以使用 mixmatch 載入資料子集。例如，您可以傳回函數中運算式的數值。然後您可以限制根據數值載入的資料。若沒有相符項目，Mixmatch 會傳回 0。在此範例中相符的所有運算式將會因此傳回 0，並且將會按照 WHERE 陳述式從資料載入中排除。

在資料載入編輯器中建立新的索引標籤，然後載入下列資料作為內嵌載入。在 Qlik Sense 中建立以下表格以查看結果。

```
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;
```

結果

使用在載入指令碼中的 mixmatch 函數來顯示輸出的 Qlik Sense 表格。

色彩代碼黑色, 藍色, 藍色	客戶
黑色	203521
黑色	3036491
藍色	2038593
藍色	5646471

範例 - 使用 mixmatch 的圖表運算式

範例:圖表運算式

在資料載入編輯器中建立新的索引標籤, 然後載入下列資料作為內嵌載入。載入資料後, 在 Qlik Sense 表格中建立以下的圖表運算式範例。

圖表運算式 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];
```

以下表格中的第一個運算式會傳回 0 代表 Stockholm, 因為 'Stockholm' 不包括在 **mixmatch** 函數的運算式清單中。這會傳回 4 代表 'Zurich', 因為 **mixmatch** 比較區分大小寫。

在圖表運算式中顯示 *mixmatch* 函數範例的 Qlik Sense 表格。

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

圖表運算式 2

您可以使用 mixmatch 以執行運算式的自訂排序。

按照預設, 欄會按字母或數字排序, 視資料而定。

Qlik Sense 表格顯示預設排序的範例

Cities
Beijing
Boston
Stockholm
Toronto
zurich

若要變更順序，請進行下列事項：

1. 在**屬性**面板中開啟圖表的**排序**區段。
2. 對於您要進行自訂排序的欄，關閉自動排序。
3. 取消選取**按數字排序**和**按字母排序**。
4. 選取**按運算式排序**，然後輸入下列運算式：
   `=mixmatch( Cities, 'Toronto','Boston','Beijing','Stockholm','Zurich')`
   Cities 欄的排序順序會變更。

Qlik Sense 表格顯示使用 *mixmatch* 函數變更排序順序的範例。

Cities
多倫多
Boston
Beijing
Stockholm
zurich

您也可以檢視傳回的數值。

Qlik Sense 表格顯示從 *mixmatch* 函數傳回的數值範例。

Cities	Cities & ' - ' & mixmatch (Cities, 'Toronto','Boston', 'Beijing','Stockholm','Zurich')
多倫多	多倫多 - 1
Boston	Boston - 2
北京	Beijing - 3
斯德哥爾摩	Stockholm - 4
蘇黎世	zurich - 5

pick

pick 函數會傳回清單中的第 *n* 個運算式。

語法：

```
pick(n, expr1[ , expr2,...exprN])
```

引數：

引數

引數	描述
n	*n* 為介於 1 和 N 之間的整數。

範例：

範例

範例	結果
pick(N, 'A','B',4, 6)	如果 N = 2，傳回 'B' 如果 N = 3，傳回 4

wildmatch

wildmatch 函數會比較第一個參數與所有後續參數，並傳回相符的運算式數目。這允許在運算式字串中使用萬用字元 (***** 和 **?**)。***** 符合任何字元順序。**?** 符合任何單一字元。比較不區分大小寫。

語法：

```
wildmatch( str, expr1 [ , expr2,...exprN ])
```

如果您想要不用萬用字元進行比較，請使用 **match** 或 **mixmatch** 函數。

範例：使用 wildmatch 的載入指令碼

範例：載入指令碼

載入指令碼

您可以使用 wildmatch 載入資料子集。例如，您可以傳回函數中運算式的數值。然後您可以限制根據數值載入的資料。若沒有相符項目，Wildmatch 會傳回 0。在此範例中相符的所有運算式將會因此傳回 0，並且將會按照 WHERE 陳述式從資料載入中排除。

在資料載入編輯器中建立新的索引標籤，然後載入下列資料作為內嵌載入。在 Qlik Sense 中建立以下表格以查看結果。

```
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;
```

結果

使用在載入指令碼中的 *wildmatch* 函數來顯示輸出的 Qlik Sense 表格

色彩代碼黑色, 藍色, 藍色, 紅色	客戶
黑色	203521
黑色	3036491
藍色	2038593
藍色	5646471
紅色	049681
紅色	2038593

範例:使用 wildmatch 的圖表運算式

範例:圖表運算式

圖表運算式 1

在資料載入編輯器中建立新的索引標籤,然後載入下列資料作為內嵌載入。載入資料後,在 Qlik Sense 表格中建立以下的圖表運算式範例。

```
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];
```

以下表格中的第一個運算式會傳回 0 代表 Stockholm,因為 'Stockholm' 不包括在 **wildmatch** 函數的運算式清單中。這也會傳回 0 代表 'Boston',因為 ? 只符合單個字元。

在圖表運算式中顯示 *wildmatch* 函數範例的 Qlik Sense 表格。

Cities	wildmatch(Cities,'Tor*','?ton','Beijing','*urich')	wildmatch(Cities,'Tor*','???ton','Beijing','Stockholm','*urich')
Beijing	3	3
Boston	0	2
Stockholm	0	4
Toronto	1	1
zurich	4	5

圖表運算式 2

您可以使用 wildmatch 以執行運算式的自訂排序。

按照預設, 會按數字或字母排序欄, 視資料而定。

Qlik Sense 表格顯示預設排序的範例

Cities
Beijing
Boston
Stockholm
Toronto
zurich

若要變更順序, 請進行下列事項:

1. 在**屬性**面板中開啟圖表的**排序**區段。
2. 對於您要進行自訂排序的欄, 關閉自動排序。
3. 取消選取**按數字排序**和**按字母排序**。
4. 選取**按運算式排序**, 然後輸入類似下列項目的運算式:
   `=wildmatch( Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')`
   Cities 欄的排序順序會變更。

Qlik Sense 表格顯示使用 *wildmatch* 函數變更排序順序的範例。

Cities
多倫多
Boston
Beijing
Stockholm
zurich

您也可以檢視傳回的數值。

Qlik Sense 表格顯示從 *wildmatch* 函數傳回的數值範例

Cities	Cities & ' - ' & wildmatch (Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')
多倫多	多倫多 - 1
Boston	Boston - 2
北京	Beijing - 3
斯德哥爾摩	Stockholm - 4
蘇黎世	zurich - 5

8.6 計數器函數

本節描述在資料載入指令碼中的 **LOAD** 陳述式評估期間，與記錄計數器相關的函數。唯一可以用於圖表運算式的函數是 **RowNo()**。

部分計數器函數沒有任何參數，但是仍需要尾部括弧。

計數器函數概述

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

autonumber

此指令碼函數會針對在指令碼執行期間出現的每個相異的 *expression* 評估值，傳回唯一的整數值。此函數可用於如建立複雜金鑰的簡潔記憶表示法。

```
autonumber (expression[ , AutoID])
```

autonumberhash128

此指令碼函數會計算組合輸入運算式值的 128 位元雜湊，並針對在指令碼執行期間出現的每個相異的雜湊值，傳回唯一的整數值。此函數可用於如建立複雜金鑰的簡潔記憶表示法。

```
autonumberhash128 (expression {, expression})
```

autonumberhash256

此指令碼函數會計算組合輸入運算式值的 256 位元雜湊，並針對在指令碼執行期間出現的每個相異的雜湊值，傳回唯一的整數值。此函數可用於如建立複雜金鑰的簡潔記憶表示法。

```
autonumberhash256 (expression {, expression})
```

IterNo

此指令碼函數會傳回整數，代表某單筆記錄在含有 **while** 子句的 **LOAD** 陳述式中進行評估的次數。第一個反覆項目有數字 1。**IterNo** 函數僅在搭配 **while** 子句使用時才有意義。

```
IterNo ( )
```

RecNo

此指令碼函數會傳回整數，代表目前讀取列在目前表格中的編號。第一筆記錄的編號為 1。

```
RecNo ( )
```

RowNo - script function

此指令碼會傳回整數，代表目前列在所產生 Qlik Sense 內部表格中的位置。第一列的編號為 1。

```
RowNo ( )
```

RowNo - chart function

RowNo() 傳回表格中目前資料行區段內，目前列的編號。對於點陣圖圖表，**RowNo()** 傳回圖表的連續表同等表格內的目前列的編號。

```
RowNo - 圖表函數([TOTAL])
```

autonumber

此指令碼函數會針對在指令碼執行期間出現的每個相異的 *expression* 評估值，傳回唯一的整數值。此函數可用於如建立複雜金鑰的簡潔記憶表示法。

> *您只能連接已在相同資料載入中產生的 **autonumber** 索引鍵，因為根據表格的讀取順序產生了整數。如果您需要在資料載入之間使用持續存在的索引鍵，無論來源資料排序為何，您應該使用 **hash128**、**hash160** 或 **hash256** 函數。*

語法：

```
autonumber(expression[ , AutoID])
```

引數：

引數	描述
AutoID	如果 **autonumber** 函數在指令碼內用於不同的索引鍵，為了建立多個計數器執行個體，可使用選用參數 *AutoID* 來命名各個計數器。

範例：建立複合索引鍵

在此範例中，我們使用 **autonumber** 函數建立複合索引鍵以節省開支。該範例僅供示範之用，在處理含大量列的表格時具有意義。

範例資料

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

使用內嵌資料載入來源資料。然後，我們新增一個前置載入，從 Region、Year 和 Month 欄位中建立複合索引鍵。

```
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
];
```

產生的表格如下所示：

結果表格

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

在此範例中，如果您需要連結至另一個表格，則可以參照 RYMkey (對於範例 1)，替代字串 'North2014May'。

現在，我們使用類似的方法載入成本來源表格。在前置載入中排除 Region、Year 和 Month 欄位，以避免建立合成鍵，我們已使用 **autonumber** 函數建立複合索引鍵，進而連結表格。

```
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
];
```

現在，我們可以將表格視覺化新增至工作表，並新增 Region、Year 和 Month 欄位，以及銷售額和成本的 Sum 量值。該表格將如下所示：

結果表格

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199

Region	Year	Month	Sum([Sales])	Sum([Costs])
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

autonumberhash128

此指令碼函數會計算組合輸入運算式值的 128 位元雜湊，並針對在指令碼執行期間出現的每個相異的雜湊值，傳回唯一的整數值。此函數可用於如建立複雜金鑰的簡潔記憶表示法。

您只能連接已在相同資料載入中產生的 ***autonumberhash128*** *索引鍵，因為根據表格的讀取順序產生了整數。如果您需要在資料載入之間使用持續存在的索引鍵，無論來源資料排序為何，您應該使用* ***hash128***、***hash160*** *或* ***hash256*** *函數。*

語法：

```
autonumberhash128(expression {, expression})
```

範例：建立複合索引鍵

在此範例中，我們使用 **autonumberhash128** 函數建立複合索引鍵以節省開支。該範例僅供示範之用，在處理含大量列的表格時具有意義。

範例資料

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

使用內嵌資料載入來源資料。然後，我們新增一個前置載入，從 Region、Year 和 Month 欄位中建立複合索引鍵。

```
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
];
```

產生的表格如下所示：

結果表格

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

在此範例中，如果您需要連結至另一個表格，則可以參照 RYMkey (對於範例 1)，替代字串 'North2014May'。

現在，我們使用類似的方法載入成本來源表格。在前置載入中排除 Region、Year 和 Month 欄位，以避免建立合成鍵，我們已使用 **autonumberhash128** 函數建立複合索引鍵，進而連結表格。

```
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
];
```

現在，我們可以將表格視覺化新增至工作表，並新增 Region、Year 和 Month 欄位，以及銷售額和成本的 Sum 量值。該表格將如下所示：

結果表格

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199

Region	Year	Month	Sum([Sales])	Sum([Costs])
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

autonumberhash256

此指令碼函數會計算組合輸入運算式值的 256 位元雜湊，並針對在指令碼執行期間出現的每個相異的雜湊值，傳回唯一的整數值。此函數可用於如建立複雜金鑰的簡潔記憶表示法。

> *您只能連接已在相同資料載入中產生的 **autonumberhash256** 索引鍵，因為根據表格的讀取順序產生了整數。如果您需要在資料載入之間使用持續存在的索引鍵，無論來源資料排序為何，您應該使用 **hash128**、**hash160** 或 **hash256** 函數。*

語法：

```
autonumberhash256(expression {, expression})
```

範例：建立複合索引鍵

在此範例中，我們使用 **autonumberhash256** 函數建立複合索引鍵以節省開支。該範例僅供示範之用，在處理含大量列的表格時具有意義。

範例表格

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

使用內嵌資料載入來源資料。然後，我們新增一個前置載入，從 Region、Year 和 Month 欄位中建立複合索引鍵。

```
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
];
```

產生的表格如下所示:

結果表格

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

在此範例中,如果您需要連結至另一個表格,則可以參照 RYMkey (對於範例 1),替代字串 'North2014May'。

現在,我們使用類似的方法載入成本來源表格。在前置載入中排除 Region、Year 和 Month 欄位,以避免建立合成鍵,我們已使用 **autonumberhash256** 函數建立複合索引鍵,進而連結表格。

```
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
];
```

現在,我們可以將表格視覺化新增至工作表,並新增 Region、Year 和 Month 欄位,以及銷售額和成本的 Sum 量值。該表格將如下所示:

結果表格

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784

Region	Year	Month	Sum([Sales])	Sum([Costs])
North	2014	June	127	199
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

IterNo

此指令碼函數會傳回整數，代表某單筆記錄在含有 **while** 子句的 **LOAD** 陳述式中進行評估的次數。第一個反覆項目有數字 1。**IterNo** 函數僅在搭配 **while** 子句使用時才有意義。

語法：

```
IterNo( )
```

範例與結果：

範例：

```
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
];
```

這個 **LOAD** 陳述式將在 **StartDate** 和 **EndDate** 定義的範圍內，每個日期產生一筆記錄。

在此範例中，產生的表格將如下所示：

結果表格

Day	Date
1	2014-01-22
2	2014-01-23
3	2014-01-24
4	2014-01-25
5	2014-01-26

RecNo

此指令碼函數會傳回整數，代表目前讀取列在目前表格中的編號。第一筆記錄的編號為 1。

語法：

```
RecNo( )
```

與在產生的 Qlik Sense 表格中計數列的 **RowNo( )** 相比，**RecNo( )** 在原始資料表格中計數記錄，並在將原始資料表格串連至另一個時予以重設。

範例：資料載入指令碼

原始資料表格載入：

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

載入所選取列的記錄和列數目：

```
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;
```

產生的 Qlik Sense 內部表格：

結果表格

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo

此指令碼會傳回整數，代表目前列在所產生 Qlik Sense 內部表格中的位置。第一列的編號為 1。

語法：

```
RowNo( [TOTAL])
```

與計數原始資料表中之記錄的 **RecNo( )** 相比，**RowNo( )** 函數並不會計數 **where** 子句所排除的記錄，而且當原始資料表和另一個資料表串連時，也不會重設。

*注意！ 如果您使用前置載入，亦即從相同表格中讀取的堆疊 **LOAD** 陳述式編號，則您僅可使用頂端 **LOAD** 陳述式中的 **RowNo( )**。如果您使用後續 **LOAD** 陳述式中的 **RowNo( )**，則會傳回 0。*

範例：資料載入指令碼

原始資料表格載入：

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

載入所選取列的記錄和列數目：

```
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;
```

產生的 Qlik Sense 內部表格：

結果表格

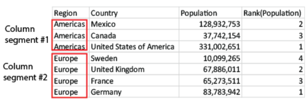
A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo - 圖表函數

RowNo() 傳回表格中目前資料行區段內，目前列的編號。對於點陣圖圖表，**RowNo()** 傳回圖表的連續表同等表格內的目前列的編號。

如果表格或同等表格有多個垂直維度，則目前資料行區段僅會包含在所有維度資料行中與目前列有相同值的列，除了顯示在欄位間排序順序中為最後一個維度的資料行之外。

資料行區段



	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

> *當任何圖表運算式使用了此圖表函數時，就不允許在圖表中依據 Y 值排序，或在表格中依據運算式資料行排序。因此，這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數，視覺化的排序將還原為對此函數的排序輸入。*

語法：

RowNo([TOTAL])

傳回的資料類型： 整數

引數：

引數	描述
TOTAL	如果表格為單一維度或 **TOTAL** 限定詞作為引數，則目前資料行區段一律等於整個資料行。

範例：使用 RowNo 的圖表運算式

範例 - 圖表運算式

載入指令碼

在資料載入編輯器中載入下列資料作為內嵌載入，以建立以下的圖表運算式範例。

```
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 '|');
```

圖表運算式

在 Qlik Sense 工作表中建立具有 **Customer** 和 **UnitSales** 作為維度的表格視覺化。各新增 `RowNo( )` 和 `RowNo(TOTAL)` 作為有**區段中的列**和 **Row Number** 標籤的量值。將以下運算式作為量值新增至表格。

```
If( RowNo( )=1, 0, UnitSales / Above( UnitSales ))
```

結果

Customer	UnitSales	Row in Segment	Row Number	If(RowNo( )=1, 0, UnitSales / Above(UnitSales))
Astrida	4	1	1	0
Astrida	9	2	2	2.25
Astrida	10	3	3	1.1111111111111
Betacab	2	1	4	0
Betacab	5	2	5	2.5
Betacab	25	3	6	5
Canutility	4	1	7	0
Canutility	8	2	8	2
Divadip	1	1	9	0
Divadip	4	2	10	4

解釋

Row in Segment 資料行顯示資料行區段結果 1、2、3，其中包含客戶 Astrida 的 UnitSales 值。接著下一個資料行區段 Betacab 的列編號會再度從 1 開始。

由於 `RowNo()` 的 `TOTAL` 引數，**Row Number** 資料行會忽略維度，並計入表格中的列。

此運算式會針對每個資料行區段的第一列傳回 `0`，因此資料行會顯示：

0、2.25、1.1111111、0、2.5、5、0、2、0 和 4。

另請參見：

Above - 圖表函數 (page 1202)

8.7 日期與時間函數

Qlik Sense 日期和時間函數用於轉換日期和時間值。所有函數皆可用於資料載入指令碼和圖表運算式。

函數以相當於從 1899 年 12 月 30 日起算的天數的日期時間序號為基礎。整數值代表天，分數值代表當天的時間。

Qlik Sense 使用參數的數值，以便當數字的格式未設定為日期或時間時，作為參數也有效。如果參數未對應於數值，例如，因為它是字串，則 Qlik Sense 會嘗試根據日期和時間環境變數來解譯字串。

如果參數中所用的時間格式未對應於環境變數中設定的時間格式，Qlik Sense 將無法正確解譯。若要解決這個問題，可變更設定或使用解譯函數。

在每一個函數的範例中，假設預設時間與日期格式是 hh:mm:ss 和 YYYY-MM-DD (ISO 8601)。

使用日期或時間函數處理時間戳記時，Qlik Sense 會忽略任何日光節約時間參數，除非日期或時間函數包括地理位置。

例如，`ConvertToLocalTime( filetime('Time.qvd'), 'Paris')` 會使用日光節約時間參數，而 `ConvertToLocalTime(filetime('Time.qvd'), 'GMT-01:00')` 則不會使用日光節約時間參數。

日期與時間函數概述

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

時間的整數運算式

second

當 **expression** 的分數根據標準數字解譯的方式可解譯為時間時，此函數會傳回代表秒的整數。

```
second (expression)
```

minute

當 **expression** 的分數根據標準數字解譯的方式可解譯為時間時，此函數會傳回代表分鐘的整數。

```
minute (expression)
```

hour

當 **expression** 的分數根據標準數字解譯的方式可解譯為時間時，此函數會傳回代表小時的整數。

```
hour (expression)
```

day

當 **expression** 的分數根據標準數字解譯的方式可解譯為日期時，此函數會傳回代表日的整數。

```
day (expression)
```

week

此函數會傳回根據 ISO 8601 代表週數的整數。週數是根據標準數字解譯，從運算式的日期解譯計算所得。

```
week (expression)
```

month

此函數會傳回雙值：一個如環境變數 **MonthNames** 中所定義的月份名稱，一個介於 1 與 12 之間的整數。月份是根據標準數字解譯，從運算式的日期解譯計算所得。

```
month (expression)
```

year

當 **expression** 根據標準數字解譯的方式可解譯為日期時，此函數會傳回代表年份的整數。

```
year (expression)
```

weekyear

此函數會根據環境變數傳回週數所屬於的年份。週數的範圍介於 1 和約 52 之間。

```
weekyear (expression)
```

weekday

此函數會傳回函下列項目的雙值：

- 如環境變數 **DayNames** 中所定義的日名稱。
- 介於 0 至 6 之間且對應於一週中正常日 (0-6) 的整數。

```
weekday (date)
```

Timestamp 函數

now

此函數會傳回目前時間的時間戳記。該函數會以 **TimeStamp** 系統變數格式傳回值。預設 **timer_mode** 值為 1。

```
now ([ timer_mode])
```

today

此函數會傳回目前日期。該函數會以 `DateFormat` 系統變數格式傳回值。

```
today ([timer_mode])
```

LocalTime

此函數會傳回指定時區目前時間的時間戳記。

```
localtime ([timezone [, ignoreDST ]])
```

Make 函數

makedate

此函數會傳回從年 **YYYY**、月 **MM** 及日 **DD** 計算而得的日期。

```
makedate (YYYY [ , MM [ , DD ] ])
```

makeweekdate

此函數會傳回從年、週數及星期幾計算而得的日期。

```
makeweekdate (YYYY [ , WW [ , D ] ])
```

maketime

此函數會傳回從小時 **hh**、分鐘 **mm** 及秒 **ss** 計算而得的時間。

```
maketime (hh [ , mm [ , ss [ .fff ] ] ])
```

Other date 函數

AddMonths

此函數會傳回 **startdate** 後 **n** 個月的日期，或者如果 **n** 是負數，則傳回 **startdate** 之前 **n** 個月的日期。

```
addmonths (startdate, n , [ , mode])
```

AddYears

此函數會傳回 **startdate** 後 **n** 年的日期，或者如果 **n** 是負數，則傳回 **startdate** 之前 **n** 年的日期。

```
addyears (startdate, n)
```

yeartodate

此功能會尋找輸入時間戳記是否位於前次載入指令碼之日期的年份內，並傳回 True (如果在其內)，False (如果不在其內)。

```
yeartodate (date [ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Timezone 函數

timezone

此函數傳回時區，如執行 Qlik 引擎的電腦上所定義。

```
timezone ( )
```

GMT

此函數會傳回目前 Greenwich Mean Time，如衍生自區域設定。

```
GMT ( )
```

UTC

傳回目前 Coordinated Universal Time。

```
UTC ( )
```

daylightsaving

傳回目前針對日光節約時間所作的調整 (如 Windows 所定義)。

```
daylightsaving ( )
```

converttolocaltime

將 UTC 或 GMT 時間戳記轉換為當地時間作為雙值。其中 place 可為全世界任何一個城市、地點及時區。

```
converttolocaltime (timestamp [, place [, ignore_dst=false]])
```

Set time 函數

setdateyear

此函數會輸入 **timestamp** 與 **year**，並使用輸入中指定的 **year** 更新 **timestamp**。

```
setdateyear  (timestamp, year)
```

setdateyearmonth

此函數會輸入 **timestamp**、**month** 與 **year**，並使用輸入中指定的 **year** 和 **month** 更新 **timestamp**。

```
setdateyearmonth (timestamp, year, month)
```

In... 函數

inyear

此函數會傳回 True，前提是如果 **timestamp** 位於包含 **base_date** 的年中。

```
inyear (date, basedate , shift [, first_month_of_year = 1])
```

inyeartodate

如果 **timestamp** 位於包含 **base_date** 的年部分內，且不超過 (可包含) **base_date** 的最後一毫秒，則此函數會傳回 True。

```
inyeartodate (date, basedate , shift [, first_month_of_year = 1])
```

inquarter

此函數會傳回 True，前提是如果 **timestamp** 位於包含 **base_date** 的季度中。

```
inquarter (date, basedate , shift [, first_month_of_year = 1])
```

inquartertodate

如果 **timestamp** 位於包含 **base_date** 的季度部分內，且不超過 (可包含) **base_date** 的最後一毫秒，則此函數會傳回 True。

```
inquartertodate (date, basedate , shift [, first_month_of_year = 1])
```

inmonth

此函數會傳回 True，前提是如果 **timestamp** 位於包含 **base_date** 的月份中。

```
inmonth (date, basedate , shift)
```

inmonthtodate

如果 **date** 位於包含 **basedate** 的月份部分內，且不超過 (可包含) **basedate** 的最後一毫秒，則傳回 True。

```
inmonthtodate (date, basedate , shift)
```

inmonths

此函數會發現時間戳記是否落在與基礎日期相同的月、雙月、季度、四月期或半年的期間內。還可以發現時間戳記是否在前一個或後一個時間期間內。

```
inmonths (n, date, basedate , shift [, first_month_of_year = 1])
```

inmonthstodate

此函數會發現時間戳記是否在月、雙月、季度、四月期或半年的期間內，且不超過 (可包含) base_date 的最後一毫秒。還可以發現時間戳記是否在前一個或後一個時間期間內。

```
inmonthstodate (n, date, basedate , shift [, first_month_of_year = 1])
```

inweek

此函數會傳回 True, 前提是如果 **timestamp** 位於包含 **base_date** 的週中。

```
inweek (date, basedate , shift [, weekstart])
```

inweektodate

如果 **timestamp** 位於包含 **base_date** 的週部分內, 且不超過 (可包含) **base_date** 的最後一毫秒, 則此函數會傳回 True。

```
inweektodate  (date, basedate , shift [, weekstart])
```

inlunarweek

此函數會判定 **timestamp** 是否位於包含 **base_date** 的農曆週內。將 1 月 1 日視為該週的第一天, 以定義 Qlik Sense 中的農曆週。除了該年的最後一週, 每週都會確切包含七天。

```
inlunarweek  (date, basedate , shift [, weekstart])
```

inlunarweektodate

此函數會發現 **timestamp** 是否位於農曆週部分內, 且不超過 (可包含) **base_date** 的最後一毫秒。將 1 月 1 日視為該週的第一天, 以定義 Qlik Sense 中的農曆週, 而除了該年的最後一週, 會確切包含七天。

```
inlunarweektodate  (date, basedate , shift [, weekstart])
```

inday

此函數會傳回 True, 前提是如果 **timestamp** 位於包含 **base_timestamp** 的日中。

```
inday (timestamp, basetimestamp , shift [, daystart])
```

indaytotime

如果 **timestamp** 位於包含 **base_timestamp** 的日部分內, 且不超過 (可包含) **base_timestamp** 的確切的毫秒, 則此函數會傳回 True。

```
indaytotime (timestamp, basetimestamp , shift [, daystart])
```

Start ... end 函數

yearstart

此函數傳回的時間戳記相當於包含 **date** 的年份的第一天開始。預設的輸出格式為指令碼中所設定的 **DateFormat**。

```
yearstart ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearend

此函數傳回的值相當於包含 **date** 的年份最後一天、最後一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 **DateFormat**。

```
yearend ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearname

此函數傳回四位數的年度為顯示值，其基礎數值相當於包含 **date** 的年度的第一天、第一毫秒的時間戳記。

```
yearname (date [, shift = 0 [, first_month_of_year = 1]] )
```

quarterstart

此函數傳回的值相當於包含 **date** 的季度的第一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 **DateFormat**。

```
quarterstart (date [, shift = 0 [, first_month_of_year = 1]])
```

quarterend

此函數傳回的值相當於包含 **date** 的季度的最後一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 **DateFormat**。

```
quarterend (date [, shift = 0 [, first_month_of_year = 1]])
```

quartername

此函數會傳回顯示當季度月份 (根據 **MonthNames** 指令碼變數進行格式設定) 和年度的顯示值，其基礎數值相當於該季度第一天、第一毫秒的時間戳記。

```
quartername (date [, shift = 0 [, first_month_of_year = 1]])
```

monthstart

此函數傳回的值相當於包含 **date** 的月份第一天、第一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 **DateFormat**。

```
monthstart (date [, shift = 0])
```

monthend

此函數傳回的值相當於包含 `date` 的月份最後一天、最後一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 `DateFormat`。

```
monthend (date [, shift = 0])
```

monthname

此函數會傳回顯示月份 (根據 **MonthNames** 指令碼變數進行格式設定) 和年度的顯示值，其基礎數值相當於該月份第一天、第一毫秒的時間戳記。

```
monthname (date [, shift = 0])
```

monthsstart

此函數傳回的值相當於包含基本日期的月、雙月、季度、四月期或半年的第一毫秒的時間戳記。還可以尋找前一個或後一個時間期間的時間戳記。預設的輸出格式為指令碼中所設定的 **DateFormat**。

```
monthsstart (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsend

此函數傳回的值相當於包含基本日期的月、雙月、季度、四月期或半年的最後一毫秒的時間戳記。還可以尋找前一個或後一個時間期間的時間戳記。

```
monthsend (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsname

此函數會傳回一個顯示值，代表期間月份 (根據 **MonthNames** 指令碼變數進行格式設定) 和年度。基礎數值相當於包含基本日期的月、雙月、季度、四月期或半年的第一毫秒的時間戳記。

```
monthsname (n, date [, shift = 0 [, first_month_of_year = 1]])
```

weekstart

此功能傳回的值相當於包含 **date**的行事曆週第一天、第一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 **DateFormat**。

```
weekstart (date [, shift = 0 [,weekoffset = 0]])
```

weekend

此功能傳回的值相當於包含 **date** 的該行事曆週最後一天的最後一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 **DateFormat**。

```
weekend (date [, shift = 0 [,weekoffset = 0]])
```

weekname

此函數會傳回顯示年度和週數的值，其基礎數值相當於包含 **date** 之週的第一天、第一毫秒的時間戳記。

```
weekname (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekstart

此函數傳回的值相當於包含 **date** 的該農曆週第一天的第一毫秒的時間戳記。將 1 月 1 日視為該週的第一天，以定義 Qlik Sense 中的農曆週，而除了該年的最後一週，會確切包含七天。

```
lunarweekstart (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekend

此函數傳回的值相當於包含 **date** 的該農曆週最後一天的最後一毫秒的時間戳記。將 1 月 1 日視為該週的第一天，以定義 Qlik Sense 中的農曆週，而除了該年的最後一週，會確切包含七天。

```
lunarweekend (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekname

此函數會傳回一個顯示值，顯示相當於包含 **date** 之農曆週第一天的第一毫秒的時間戳記的年和農曆週數。將 1 月 1 日視為該週的第一天，以定義 Qlik Sense 中的農曆週，而除了該年的最後一週，會確切包含七天。

```
lunarweekname (date [, shift = 0 [,weekoffset = 0]])
```

daystart

此函數傳回的值相當於包含在 **time** 引數中該日的第一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 **TimestampFormat**。

```
daystart (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayend

此函數傳回的值相當於包含在 **time** 中該日的最後一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 **TimestampFormat**。

```
dayend (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayname

此函數會傳回顯示日期的值，其基礎數值相當於包含 **time** 之日的第一毫秒的時間戳記。

```
dayname (timestamp [, shift = 0 [, dayoffset = 0]])
```

Day numbering 函數

age

age 函數傳回生日為 **date_of_birth** 的某人於 **timestamp** 時的年齡 (以實歲計)。

```
age (timestamp, date_of_birth)
```

networkdays

networkdays 函數會傳回介於 **start_date** 和 **end_date** 之間 (可包含) 的工作日數 (星期一至星期五)，並考慮到所有選用的列出 **holiday**。

```
networkdays  (start:date, end_date {, holiday})
```

firstworkdate

firstworkdate 函數會傳回在不晚於 **end_date** 前結束，並考慮到所有選用的列出假日，而可達到 **no_of_workdays** (星期一至星期五) 的最近開始日期。**end_date** 和 **holiday** 應該是有效的日期或時間戳記。

```
firstworkdate (end_date, no_of_workdays {, holiday} )
```

lastworkdate

lastworkdate 函數會傳回開始於 **start_date**，並考慮到所有選用的列出 **holiday**，而可達到 **no_of_workdays** (星期一至星期五) 的最早結束日期。**start_date** 和 **holiday** 應該是有效的日期或時間戳記。

```
lastworkdate (start_date, no_of_workdays {, holiday})
```

daynumberofyear

此函數會計算時間戳記在當年的第幾天。該計算從當年第一天的第一毫秒開始進行，但是第一個月可能發生位移。

```
daynumberofyear (date[,firstmonth])
```

daynumberofquarter

此函數會計算時間戳記在當季度的第幾天。在建立主要行事曆時使用此函數。

```
daynumberofquarter (date[,firstmonth])
```

addmonths

此函數會傳回 **startdate** 後 **n** 個月的日期，或者如果 **n** 是負數，則傳回 **startdate** 之前 **n** 個月的日期。

語法：

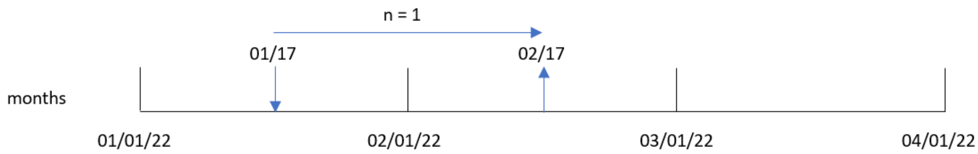
```
AddMonths(startdate, n , [ , mode])
```

傳回的資料類型： 雙值

addmonths() 函數對 startdate 加上或減去定義的月數 n，並傳回產生的日期。

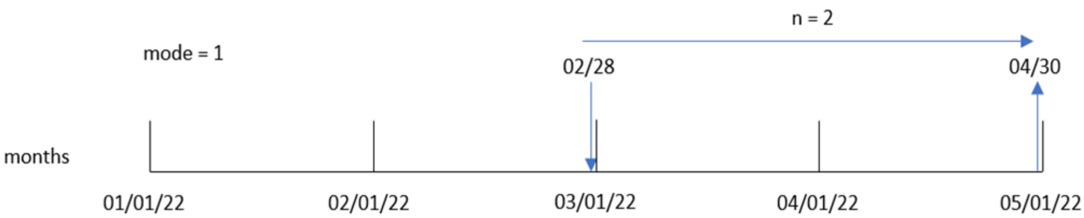
mode 引數將會影響該月 28 日當日或之後的 startdate 值。透過將 mode 引數設為 1，addmonths() 函數會傳回與該月結束時的相對距離等於 startdate 的日期。

addmonths() 函數的範例圖表



例如，2 月 28 日是該月的最後一天。若 mode 為 1 的 addmonths() 函數用來傳回兩個月後的日期，則該函數將會傳回 4 月的最後一個日期 4 月 30 日。

addmonths() 函數的範例圖表，具有 mode=1



引數

引數	描述
startdate	以時間戳記表示的開始日期，例如 '2012-10-12'。
n	以正整數或負整數表示的月份數。
mode	指定新增月份是與月初或月尾相關。對於與月初相關的新增項目，預設模式為 0。對於與月尾相關的新增項目，將模式設定為 1。若模式設定為 1，且輸入日期為 28 日或以上，則函數會檢查距離開始日期的月尾剩下多少天。在傳回的日期設定的天數，與距離月尾的天數相同。

什麼情況下使用

`addmonths()` 函數通常會用於運算式，以尋找某個期間給定月數之前或之後的日期。

例如，`addmonths()` 函數可用來識別手機聯絡人的結束日期。

函數範例

範例	結果
`addmonths ('01/29/2003' ,3)`	傳回「04/29/2003」。
`addmonths ('01/29/2003',3,0)`	傳回「04/29/2003」。
`addmonths ('01/29/2003',3,1)`	傳回「04/28/2003」。
`addmonths ('01/29/2003',1,0)`	傳回「02/28/2003」。
`addmonths ('01/29/2003',1,1)`	傳回「02/26/2003」。
`addmonths ('02/28/2003',1,0)`	傳回「03/28/2003」。
`addmonths ('02/28/2003',1,1)`	傳回「03/31/2003」。
`addmonths ('01/29/2003',-3)`	傳回「10/29/2002」。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2020 年和 2022 年之間交易的資料集，這載入到稱為「`Transactions`」的表格中。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 `two_months_later`，這傳回交易發生後兩個月的日期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- two_months_later

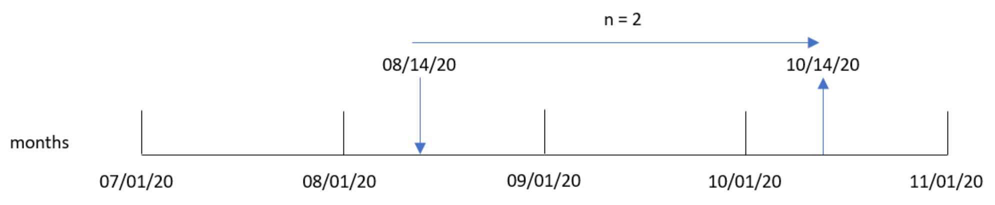
結果表格

日期	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

日期	two_months_later
05/21/2020	07/21/2020
08/14/2020	10/14/2020
10/07/2020	12/07/2020
12/05/2020	02/05/2021
01/22/2021	03/22/2021
02/03/2021	04/03/2021
03/17/2021	05/17/2021
04/23/2021	06/23/2021
05/04/2021	07/04/2021
06/30/2021	08/30/2021
07/26/2021	09/26/2021
12/27/2021	02/27/2022
02/02/2022	04/02/2022
02/26/2022	04/26/2022
03/07/2022	05/07/2022
03/11/2022	05/11/2022

「two_months_later」欄位使用 addmonths() 函數在前置 LOAD 陳述式中建立。提供的第一個引數識別正在評估哪個日期。第二個引數是要從 startdate 加上或減去的月數。在此例中，提供的值為 2。

addmonths() 函數的圖表，無其他引數的範例



交易 8193 發生於 8 月 14 日。因此， addmonths() 函數會為 two_months_later 欄位傳回 2020 年 10 月 14 日。

範例 2 – 相對月底

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年月底交易的資料集，這載入到稱為 `Transactions` 的表格中。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 `relative_two_months_prior`，這傳回交易發生前兩個月的相對月底日期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

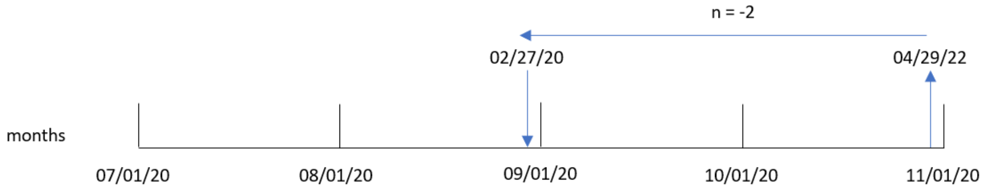
- date
- relative_two_months_prior

結果表格

日期	relative_two_months_prior
01/28/2022	11/27/2021
01/31/2022	11/30/2021
02/28/2022	12/31/2021
04/29/2022	02/27/2022
04/30/2022	02/28/2022
05/31/2022	03/31/2022
08/14/2022	06/14/2022
10/07/2022	08/07/2022

relative_two_months_prior 欄位使用 addmonths() 函數在前置 LOAD 陳述式中建立。提供的第一個引數識別正在評估哪個日期。第二個引數是要從 startdate 加上或減去的月數。在此例中，提供的值為 -2。最終引數是模式，值為 1，這強制該函數為大於或等於 28 的所有日期計算相對月底日期。

addmonths() 函數的圖表，範例具有 n=-2



交易 8191 發生在 2022 年 4 月 29 日。最初，提前兩個月會將月份設定為 2 月。然後，由於函數的第三個引數將模式設定為 1，而日期值晚於 27 日，該函數會計算相對月底值。該函數識別 29 日是 4 月的倒數第二天，因此會傳回 2 月的倒數第二天 2 月 27 日。

範例 3 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。

不過，在此範例中，不變的資料集會載入到應用程式中。會建立傳回交易發生後兩個月之日期的計算，作為圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`date`。

建立下列量值：

`=addmonths(date,2)`

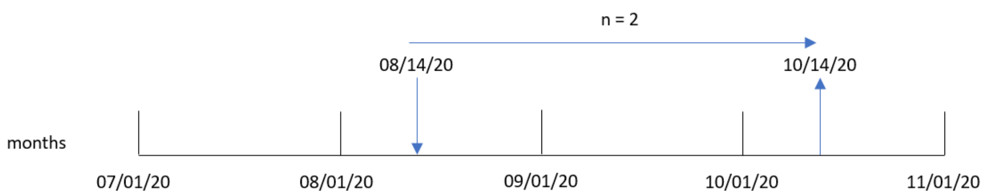
結果表格

日期	=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

日期	=addmonths(date,2)
03/07/2022	05/07/2022
03/11/2022	05/11/2022

會使用 addmonths() 函數在圖表物件中建立 two_months_later 量值。提供的第一個引數識別正在評估哪個日期。第二個引數是要從 startdate 加上或減去的月數。在此例中，提供的值為 2。

addmonths() 函數的圖表，圖表物件範例



交易 8193 發生於 8 月 14 日。因此，addmonths() 函數會為 two_months_later 欄位傳回 2020 年 10 月 14 日。

範例 4 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入到稱為 Mobile_Plans 之表格的資料集。
- 含有合約 ID、開始日期、合約長度和每月費用的資訊。

最終使用者希望圖表物件依合約 ID 顯示每份手機合約的終止日期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- contract_id
- start_date
- contract_length

建立下列量值以計算每份合約的結束日期:

```
=addmonths(start_date,contract_length, 0)
```

結果表格

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

contract_id	start_date	contract_length	=addmonths(start_date,contract_length,0)
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

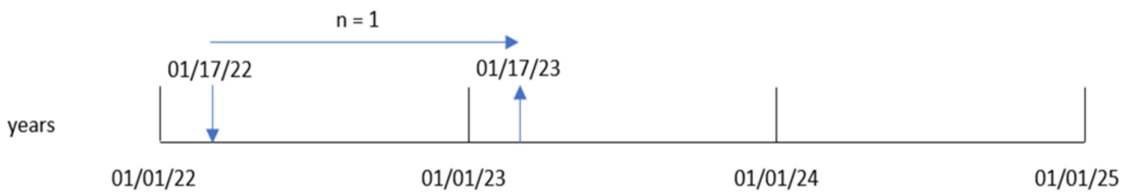
此函數會傳回 **startdate** 後 **n** 年的日期，或者如果 **n** 是負數，則傳回 **startdate** 之前 **n** 年的日期。

語法：

```
AddYears(startdate, n)
```

傳回的資料類型： 雙值

addyears() 函數的範例圖表



addyears() 函數對 startdate 加上或減去定義的年數 n。然後這會傳回產生的日期。

引數

引數	描述
startdate	以時間戳記表示的開始日期，例如 '2012-10-12'。
n	以正整數或負整數表示的年份數。

函數範例

範例	結果
`addyears ('01/29/2010',3)`	傳回「01/29/2013」。
`addyears ('01/29/2010',-1)`	傳回「01/29/2009」。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 - 簡單範例

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2020 年和 2022 年之間交易的資料集，這載入到稱為「Transactions」的表格中。
- 以 DateFormat 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 two_years_later，這傳回交易發生後兩年的日期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- two_years_later

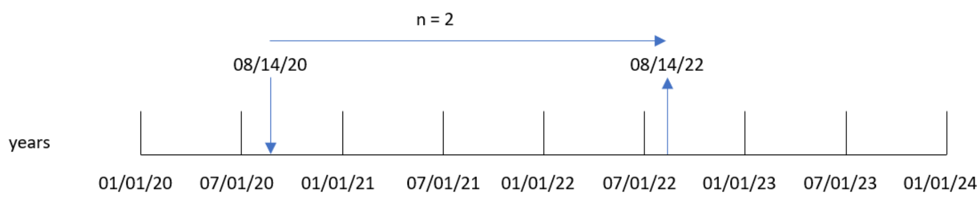
結果表格

日期	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

日期	two_years_later
03/07/2022	03/07/2024
03/11/2022	03/11/2024

「two_years_later」欄位使用 addyears() 函數在前置 LOAD 陳述式中建立。提供的第一個引數識別正在評估哪個日期。第二個引數是要從開始日期加上或減去的年數。在此例中，提供的值為 2。

addyears() 函數的圖表，基本範例



交易 8193 發生於 2020 年 8 月 14 日。因此，addyears() 函數會為 two_years_later 欄位傳回 2022 年 8 月 14 日。

範例 2 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2020 年和 2022 年之間交易的資料集，這載入到稱為 Transactions 的表格中。
- 以 DateFormat 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。

在圖表物件中，建立量值 prior_year_date，這傳回交易發生前一年的日期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`date`。

建立下列量值以計算每個交易前一年的日期：

`=addyears(date,-1)`

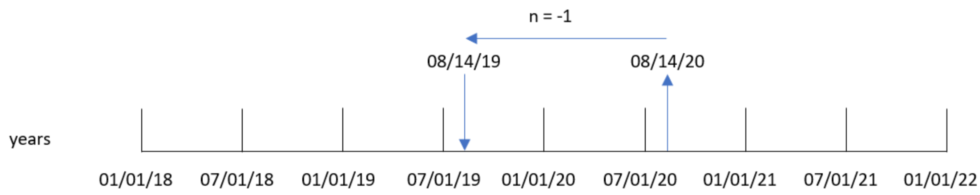
結果表格

日期	=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

日期	=addyears(date,-1)
12/27/2021	12/27/2020
02/02/2022	02/02/2021
02/26/2022	02/26/2021
03/07/2022	03/07/2021
03/11/2022	03/11/2021

會使用 `addyears()` 函數在圖表物件中建立 `one_year_prior` 量值。提供的第一個引數識別正在評估哪個日期。第二個引數是要從 `startdate` 加上或減去的年數。在此例中，提供的值為 -1。

`addyears()` 函數的圖表，圖表物件範例



交易 8193 發生於 8 月 14 日。因此，`addyears()` 函數會為 `one_year_prior` 欄位傳回 2019 年 8 月 14 日。

範例 3 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入到稱為 `Warranties` 之表格的資料集。
- 含有產品 ID、購買日期、保固長度和購買價格的資訊。

最終使用者希望圖表物件依產品 ID 顯示每個產品的保固終止日期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- product_id
- purchase_date
- warranty_length

建立下列量值以計算每個產品保固的結束日期：

=addyears(purchase_date,warranty_length)

結果表格

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

product_id	purchase_date	warranty_length	=addyears(purchase_date,warranty_length)
8199	04/23/2021	3	04/23/2024
8200	05/04/2021	4	05/04/2025
8201	06/30/2021	3	06/30/2024
8202	07/26/2021	4	07/26/2025
8203	12/27/2021	4	12/27/2025
8204	06/06/2022	2	06/06/2024
8205	07/18/2022	1	07/18/2023
8206	11/14/2022	1	11/14/2023
8207	12/12/2022	4	12/12/2026

age

age 函數傳回生日為 **date_of_birth** 的某人於 **timestamp** 時的年齡 (以實歲計)。

語法:

```
age(timestamp, date_of_birth)
```

可以是運算式。

傳回的資料類型: 數值

引數:

引數

引數	描述
timestamp	時間戳記或解析為時間戳記的運算式，計算截至該時間的已完成年份數。
date_of_birth	正在計算其年齡之人員的出生日期。可以是運算式。

範例與結果:

這些範例使用日期格式 **DD/MM/YYYY**。日期格式是在位於資料載入指令碼頂端的 **SET DateFormat** 陳述式中指定。變更範例中的格式，以滿足您的需求。

指令碼處理範例

範例	結果
age('25/01/2014', '29/10/2012')	傳回 1。
age('29/10/2014', '29/10/2012')	傳回 2。

範例：

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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;
```

產生的表格顯示表格中各記錄的 age 傳回值。

結果表格

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

將 UTC 或 GMT 時間戳記轉換為當地時間作為雙值。其中 place 可為全世界任何一個城市、地點及時區。

語法:

```
ConvertToLocalTime(timestamp [, place [, ignore_dst=false]])
```

傳回的資料類型: 雙值

引數

<table>
<tr><th>引數</th><th>描述</th></tr>
<tr><td>timestamp</td><td>要轉換的時間戳記或解析為時間戳記的運算式。</td></tr>
<tr><td>place</td><td>下方有效地點及時區表格中的地點或時區。或者,您可以使用 GMT 或 UTC 來定義當地時間。下列值及時間偏移範圍有效:

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

若您使用 DST 時差 (亦即您指定評估為 False 的 ignore_dst 引數值),您必須在 place 引數中指定一個地方,而非 GMT 時差。這是因為對日光節約時間進行調整時,除了 GMT 時差提供的經度資訊,還需要緯度資訊。如需資訊,請參閱 結合 DST 使用 GMT 時差 (page 578)。

您只能使用標準時間偏移。不能使用任意的時間偏移,例如,GMT-04:27。</td></tr>
<tr><td>ignore_dst</td><td>若此引數評估為 True,會忽略 DST (日光節約時間)。評估為 True 的有效引數值包括 <code>-1</code> 和 <code>True()</code>。

若此引數評估為 False,會為日光節約時間調整時間戳記。評估為 False 的有效引數值包括 <code>0</code> 和 <code>False()</code>。

若 ignore_dst 引數值無效,該函數會按照 ignore_dst 值評估為 True 的情境來評估運算式。若未指定 ignore_dst 引數值,該函數會按照 ignore_dst 值評估為 False 的情境來評估運算式。</td></tr>
</table>

有效的地點和時區

A-C	D-K	L-R	S-Z
Abu Dhabi	Darwin	La Paz	Samoa
Adelaide	Dhaka	Lima	Santiago
Alaska	Eastern Time (US & Canada)	Lisbon	Sapporo
Amsterdam	Edinburgh	Ljubljana	Sarajevo
Arizona	Ekaterinburg	London	Saskatchewan
Astana	Fiji	Madrid	Seoul
Athens	Georgetown	Magadan	Singapore
Atlantic Time (Canada)	Greenland	Mazatlan	Skopje
Auckland	Greenwich Mean Time : Dublin	Melbourne	Sofia
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

A-C	D-K	L-R	S-Z
Canberra	Karachi	Paris	Vladivostok
Cape Verde Is.	Kathmandu	Perth	Volgograd
Caracas	Kolkata	Port Moresby	Yakutsk
Casablanca	Krasnoyarsk	Prague	Yerevan
Central America	Kuala Lumpur	Pretoria	Zagreb
Central Time (US & Canada)	Kuwait	Quito	-
Chennai	Kyiv	Riga	-
Chihuahua	-	Riyadh	-
Chongqing	-	Rome	-
Copenhagen	-	-	-

範例與結果:

指令碼處理範例

範例	結果
`ConvertToLocalTime('2023-08-14 08:39:47','Paris')`	傳回 '2023-08-14 10:39:47' 及對應的內部時間戳記表示法。
`ConvertToLocalTime(UTC(), 'Stockholm')`	傳回斯德哥爾摩的時間，為日光節約時間進行調整。
`ConvertToLocalTime(UTC(), 'Stockholm', -1)`	傳回斯德哥爾摩的時間，不會進行日光節約時間調整。
`ConvertToLocalTime(UTC(), 'GMT-05:00')`	傳回北美東岸 (例如紐約) 的時間。不會為日光節約時間進行任何調整，因為會指定 GMT 時差，而非某個地方。
`ConvertToLocalTime(UTC(), 'New York', -1)`	傳回北美東岸 (紐約) 的時間，無日光節約時間調整。
`ConvertToLocalTime(UTC(), 'New York', True())`	傳回北美東岸 (紐約) 的時間，無日光節約時間調整。
`ConvertToLocalTime(UTC(), 'New York', 0)`	傳回北美東岸 (紐約) 的時間，為日光節約時間進行調整。
`ConvertToLocalTime(UTC(), 'New York', False())`	傳回北美東岸 (紐約) 的時間，為日光節約時間進行調整。

結合 DST 使用 GMT 時差

按照 Qlik Sense 中國際統一碼元件 (ICU) 庫的實施，結合 DST (日光節約時間) 使用 GMT (格林威治標準時間) 時差需要附加的緯度資訊。

GMT 是經度 (東西向) 時差，而 DST 則是緯度 (南北向) 時差。例如，赫爾辛基 (芬蘭) 和約翰尼斯堡 (南非) 共用相同的 GMT+02:00 時差，但沒有共用相同的 DST 時差。這表示，除了 GMT 時差，任何 DST 時差都需要當地時區的緯度位置資訊 (地理時區輸入)，才能有完整的當地 DST 條件資訊。

day

當 **expression** 的分數根據標準數字解譯的方式可解譯為日期時，此函數會傳回代表日的整數。

該函數針對特定日期傳回該月的日子。這通常用來衍生日欄位，作為行事曆維度的一部分。

語法：

```
day(expression)
```

傳回的資料類型： 整數

函數範例

範例	結果
day(1971-10-12)	傳回 12
day(35648)	傳回 6，因為 35648 = 1997-08-06

範例 1 – DateFormat 資料集 (指令碼)

載入指令碼和結果

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 名稱為 Master_Calendar 的日期資料集。DateFormat 系統變數設定為 DD/MM/YYYY。
- 使用 day() 函數建立附加欄位 (名稱為 day_of_month) 的前置載入。
- 名稱為 long_date 的附加欄位，使用 date() 函數表示月份全名。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- long_date
- day_of_month

結果表格

日期	long_date	day_of_month
03/11/2022	2022 年 3 月 11 日	11
03/12/2022	2022 年 3 月 12 日	12
03/13/2022	2022 年 3 月 13 日	13
03/14/2022	2022 年 3 月 14 日	14
03/15/2022	2022 年 3 月 15 日	15
03/16/2022	2022 年 3 月 16 日	16
03/17/2022	2022 年 3 月 17 日	17
03/18/2022	2022 年 3 月 18 日	18
03/19/2022	2022 年 3 月 19 日	19
03/20/2022	2022 年 3 月 20 日	20
03/21/2022	2022 年 3 月 21 日	21

月份中的日期目前由指令碼中的 `day()` 函數正確評估。

範例 2 – ANSI 日期 (指令碼)

載入指令碼和結果

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 名稱為 Master_Calendar 的日期資料集。使用 DateFormat 系統變數 DD/MM/YYYY。不過，資料集中包括的日期屬於 ANSI 標準日期格式。
- 使用 date() 函數建立附加欄位 (名稱為 day_of_month) 的前置載入。
- 名稱為 long_date 的附加欄位，使用 date() 函數表示日期與月份全名。

載入指令碼

```
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
 ];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- long_date
- day_of_month

結果表格

日期	long_date	day_of_month
03/11/2022	2022 年 3 月 11 日	11
03/12/2022	2022 年 3 月 12 日	12
03/13/2022	2022 年 3 月 13 日	13
03/14/2022	2022 年 3 月 14 日	14
03/15/2022	2022 年 3 月 15 日	15
03/16/2022	2022 年 3 月 16 日	16

日期	long_date	day_of_month
03/17/2022	2022 年 3 月 17 日	17
03/18/2022	2022 年 3 月 18 日	18
03/19/2022	2022 年 3 月 19 日	19
03/20/2022	2022 年 3 月 20 日	20
03/21/2022	2022 年 3 月 21 日	21

月份中的日期目前由指令碼中的 day() 函數正確評估。

範例 3 – 未格式化的日期 (指令碼)

載入指令碼和結果

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 名稱為 Master_Calendar 的日期資料集。使用 DateFormat 系統變數 DD/MM/YYYY。
- 使用 day() 函數建立附加欄位 (名稱為 day_of_month) 的前置載入。
- 原始未格式化的日期, 名稱為 unformatted_date。
- 使用 date() 且名稱為 long_date 的附加欄位用來將數值日期轉換為格式化的日期欄位。

載入指令碼

```
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
```

```
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- `unformatted_date`
- `long_date`
- `day_of_month`

結果表格

unformatted_date	long_date	day_of_month
44868	2022 年 11 月 3 日	3
44898	2022 年 12 月 3 日	3
44928	2023 年 1 月 2 日	2
44958	2023 年 2 月 1 日	1
44988	2023 年 3 月 3 日	3
45008	23-March- 2023	23
45018	2023 年 4 月 2 日	2
45038	2023 年 4 月 22 日	22
45048	2023 年 5 月 2 日	2
45068	2023 年 5 月 22 日	22
45078	2023 年 6 月 1 日	1

月份中的日期目前由指令碼中的 `day()` 函數正確評估。

範例 4 – 計算到期月份 (圖表)

載入指令碼和圖表運算式

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 名稱為 `orders` 的三月下訂訂單資料集。表格含有三個欄位：
  - id
  - order_date
  - 金額

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度:`order_date`。

若要計算遞送日期,建立此量值:`=day(order_date+5)`。

結果表格

order_date	=day(order_date+5)
03/11/2022	16
03/12/2022	17
03/13/2022	18
03/14/2022	19
03/15/2022	20
03/16/2022	21
03/17/2022	22
03/18/2022	23
03/19/2022	24
03/20/2022	25
03/21/2022	26

`day()` 函數根據 5 天運送期間正確判定在 3 月 11 日下訂的訂單會在 16 日送達。

dayend

此函數傳回的值相當於包含在 **time** 中該日的最後一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 **TimestampFormat**。

語法：

```
DayEnd(time[, [period_no[, day_start]])
```

什麼情況下使用

`dayend()` 函數通常在使用者想要計算以使用一天中尚未發生的部分時，作為運算式的一部分使用。例如，用來計算一天當中仍待產生的整體費用。

傳回的資料類型： 雙值

引數

引數	描述
time	要評估的時間戳記。
period_no	**period_no** 是一個整數，或者解析為整數的運算式，其中值 0 表示包含 **time** 的日。負值的 **period_no** 表示之前的日，正值表示之後的日。
day_start	若要指定日不從午夜開始，可在 **day_start** 中指示位移 (以一日的片段表示)。例如，0.125 表示上午 3:00。 換言之，若要建立偏移，請依 24 小時分割開始時間。例如，對於在上午 7:00 開始的日子，請使用分數 7/24。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
dayend('01/25/2013 16:45:00')	傳回 01/25/2013 23:59:59。PM
dayend('01/25/2013 16:45:00', -1)	傳回 01/24/2013 23:59:59。PM
dayend('01/25/2013 16:45:00', 0, 0.5)	傳回 01/26/2013 11:59:59。PM

範例 1 - 基本指令碼

載入指令碼和結果

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含日期清單的資料集載入到名為「Calendar」的表格中。
- 預設 DateFormat 系統變數 (MM/DD/YYYY)。
- 使用 dayend() 函數建立附加欄位 'EOD_timestamp' 的前置載入。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- EOD_timestamp

結果表格

日期	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

如同您在以上表格中所見，會為資料集中的每個日期產生日子結束時間戳記。時間戳記的格式是系統變數 `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`。

範例 2 – period_no

載入指令碼和結果

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

您會將包含服務預約的資料集載入到名稱為 'Services' 的表格。

資料集包括下列欄位：

- `service_id`
- `service_date`
- `amount`

您將會在表格中建立兩個新欄位：

- `deposit_due_date`：應收到存款的日期。這是 `service_date` 三天之前的日子結束時間。
- `final_payment_due_date`：應收到最終付款的日期。這是 `service_date` 七天之後的日子結束時間。

以上兩個欄位在前置載入中使用 `dayend()` 函數建立，並提供前兩個參數 `time` 和 `period_no`。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- service_date
- deposit_due_date
- final_payment_due_date

結果表格

service_date	deposit_due_date	final_payment_due_date
03/11/2022 9:25:14 AM	3/8/2022 11:59:59 PM	3/18/2022 11:59:59 PM
03/12/2022 10:06:54 AM	3/9/2022 11:59:59 PM	3/19/2022 11:59:59 PM
03/13/2022 10:44:42 AM	3/10/2022 11:59:59 PM	3/20/2022 11:59:59 PM
03/14/2022 11:33:30 AM	3/11/2022 11:59:59 PM	3/21/2022 11:59:59 PM
03/15/2022 12:58:14 PM	3/12/2022 11:59:59 PM	3/22/2022 11:59:59 PM
03/16/2022 4:23:12 PM	3/13/2022 11:59:59 PM	3/23/2022 11:59:59 PM

service_date	deposit_due_date	final_payment_due_date
03/17/2022 6:42:15 PM	3/14/2022 11:59:59 PM	3/24/2022 11:59:59 PM
03/18/2022 7:41:16 PM	3/15/2022 11:59:59 PM	3/25/2022 11:59:59 PM
03/19/2022 8:14:15 PM	3/16/2022 11:59:59 PM	3/26/2022 11:59:59 PM
03/20/2022 9:23:51 PM	3/17/2022 11:59:59 PM	3/27/2022 11:59:59 PM
03/21/2022 10:04:41 PM	3/18/2022 11:59:59 PM	3/28/2022 11:59:59 PM

新欄位的值位於 TimestampFormat M/D/YYYY h:mm:ss[.fff] TT 中。因為使用函數 dayend(), 時間戳記值都是一天中的最後一毫秒。

由於 dayend() 函數中傳遞的第二引數為負值, 存款到期日期值是服務日期的三天前。

由於 dayend() 函數中傳遞的第二引數為正值, 最終付款到期日期值是服務日期的七天後。

範例 3 – day_start 指令碼

載入指令碼和結果

概覽

開啟 資料載入編輯器 並在新的索引標籤中新增下面的載入指令碼。

此範例中使用的資料集和情境與上一個範例相同。

如同上一個範例, 您將會建立兩個新的欄位:

- deposit_due_date: 應收到存款的日期。這是 service_date 三天之前的日子結束時間。
- final_payment_due_date: 應收到最終付款的日期。這是 service_date 七天之後的日子結束時間。

不過, 貴公司希望根據工作日在下午 5 時開始並在次日下午 5 時結束的政策來運作。然後貴公司可以監控在這些工作時間中發生的交易。

為了達到這些要求, 以上兩個欄位在前置載入中使用 dayend() 函數建立, 並使用全部三個引數 time、period_no 和 day_start。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- service_date
- deposit_due_date
- final_payment_due_date

結果表格

service_date	deposit_due_date	final_payment_due_date
03/11/2022 9:25:14 AM	3/8/2022 4:59:59 PM	3/18/2022 4:59:59 PM
03/12/2022 10:06:54 AM	3/9/2022 4:59:59 PM	3/19/2022 4:59:59 PM
03/13/2022 10:44:42 AM	3/10/2022 4:59:59 PM	3/20/2022 4:59:59 PM
03/14/2022 11:33:30 AM	3/11/2022 4:59:59 PM	3/21/2022 4:59:59 PM
03/15/2022 12:58:14 PM	3/12/2022 4:59:59 PM	3/22/2022 4:59:59 PM
03/16/2022 4:23:12 PM	3/13/2022 4:59:59 PM	3/23/2022 4:59:59 PM
03/17/2022 6:42:15 PM	3/14/2022 4:59:59 PM	3/24/2022 4:59:59 PM
03/18/2022 7:41:16 PM	3/15/2022 4:59:59 PM	3/25/2022 4:59:59 PM
03/19/2022 8:14:15 PM	3/16/2022 4:59:59 PM	3/26/2022 4:59:59 PM
03/20/2022 9:23:51 PM	3/17/2022 4:59:59 PM	3/27/2022 4:59:59 PM
03/21/2022 10:04:41 PM	3/18/2022 4:59:59 PM	3/28/2022 4:59:59 PM

由於日期與範例 2 保持相同, 日期現在有下午 5:00 之前最後一毫秒的時間戳記, 因為傳遞至 dayend() 函數的第三引數 day_start 值為 17/24。

範例 4 – 圖表範例

載入指令碼和圖表運算式

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

此範例中使用的資料集和情境與前兩個範例相同。公司希望根據工作日在下午 5:00 開始並在次日下午 5:00 結束的政策來運作。

如同上一個範例, 您將會建立兩個新的欄位:

- deposit_due_date: 應收到存款的日期。這是 service_date 三天之前的日子結束時間。
- final_payment_due_date: 應收到最終付款的日期。這是 service_date 七天之後的日子結束時間。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度:

service_date。

若要建立 deposit_due_date 欄位, 建立此量值。

=dayend(service_date,-3,17/24)。

然後, 若要建立 final_payment_due_date 欄位, 建立此量值:

=dayend(service_date,7,17/24)。

結果表格

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

新欄位的值位於 `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT` 中。因為使用函數 `dayend()`，時間戳記值都是一天中的最後一毫秒。

由於 `dayend()` 函數中傳遞的第二引數為負值，付款到期日期值是服務日期的三天前。

由於 `dayend()` 函數中傳遞的第二引數為正值，最終付款到期日期值是服務日期的七天後。

日期有下午 5:00 之前最後一毫秒的時間戳記，因為傳遞至 `dayend()` 函數的第三引數 `day_start` 值為 17/24。

引數

引數	描述
time	要評估的時間戳記。
period_no	**period_no** 是一個整數，或者解析為整數的運算式，其中值 0 表示包含 **time** 的日。負值的 **period_no** 表示之前的日，正值表示之後的日。
day_start	若要指定日不從午夜開始，可在 **day_start** 中指示位移 (以一日的片段表示)。例如，0.125 表示上午 3:00。

daylightsaving

傳回目前針對日光節約時間所作的調整 (如 Windows 所定義)。

語法：

```
DaylightSaving( )
```

傳回的資料類型: 雙值

範例:

```
daylightsaving( )
```

dayname

此函數會傳回顯示日期的值,其基礎數值相當於包含 **time** 之日的第一毫秒的時間戳記。

語法:

```
DayName(time[, period_no [, day_start]])
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
time	要評估的時間戳記。
period_no	**period_no** 是一個整數,或者解析為整數的運算式,其中值 0 表示包含 **time** 的日。負值的 **period_no** 表示之前的日,正值表示之後的日。
day_start	若要指定日不從午夜開始,可在 **day_start** 中指示位移 (以一日的片段表示)。例如,0.125 表示上午 3:00。

範例與結果:
這些範例使用日期格式 **DD/MM/YYYY**。日期格式是在位於資料載入指令碼頂端的 **SET DateFormat** 陳述式中指定。變更範例中的格式,以滿足您的需求。

指令碼處理範例

範例	結果
dayname('25/01/2013 16:45:00')	傳回 25/01/2013。
dayname('25/01/2013 16:45:00', -1)	傳回 24/01/2013。
dayname('25/01/2013 16:45:00', 0, 0.5)	傳回 25/01/2013。 顯示完整的時間戳記會顯示對應於 '25/01/2013 12:00:00.000. 的基礎數值

範例：

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

在此範例中，在表格中的每一個發票日期之後，從標記日開始的時間戳記建立日名稱。

```
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;
```

產生的表格包含原始日期，以及具有 dayname() 函數之傳回值的資料行。您可以透過在屬性面板中指定格式設定，來顯示完整時間戳記。

結果表格

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

此函數會計算時間戳記在當季度的第幾天。在建立主要行事曆時使用此函數。

語法：

```
DayNumberOfQuarter(timestamp[,start_month])
```

傳回的資料類型： 整數

引數

引數	描述
timestamp	要評估的時間戳記。
start_month	透過指定 **start_month** 介於 2 和 12 之間 (如果省略，則為 1)，可將年初向前移到任何月份的第一天。例如，如果要使用起始於 3 月 1 日的會計年度，可指定 **start_month** = 3。

這些範例使用日期格式 **DD/MM/YYYY**。日期格式是在位於資料載入指令碼頂端的 **SET DateFormat** 陳述式中指定。變更範例中的格式，以滿足您的需求。

函數範例

範例	結果
`DayNumberOfQuarter('12/09/2014')`	傳回 74，當前季度的天數。
`DayNumberOfQuarter ('12/09/2014',3)`	傳回 12，當前季度的天數。 在此情況下，第一個季度從 3 月開始 (因為 start_month 指定為 3)。這表示當前季度是第三季度，從 9 月 1 日開始。

範例 1 – 1 月是一年的開始 (指令碼)

載入指令碼和結果

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含日期清單的簡單資料集,載入到名為 `Calendar` 的表格中。使用預設 `DateFormat` 系統變數 MM/DD/YYYY。
- 使用 `DayNumberOfQuarter()` 函數建立附加欄位 (名稱為 `DayNrQtr`) 的前置載入。

除了日期,不會向函數提供其他參數。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- `date`
- `daynrqtr`

結果表格

日期	daynrqtr
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
02/28/2022	59
03/01/2022	61
03/31/2022	91
04/01/2022	1

該年的第一天是 1 月 1 日，因為沒有第二引數傳遞至 DayNumberOfQuarter() 函數中。

1 月 1 日是該季的第 1 天，而 2 月 1 日是該季的第 32 天。3 月 31 日是該季的第 91 天和最後一天，而 4 月 1 日則是第 2 季的第 1 天。

範例 2 – 2 月是一年的開始 (指令碼)

載入指令碼和結果

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 資料集與第一個範例中相同。
- 使用預設 DateFormat 系統變數 MM/DD/YYYY。
- 在 2 月 1 日開始的 start_month 引數。這將會計年度設定為 2 月 1 日。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- daynrqtr

結果表格

日期	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

該年的第一天是 2 月 1 日, 因為傳遞至 DayNumberOfQuarter() 函數中的第二引數是 2。

該年的第一季從 2 月到 4 月, 而第四季則從 11 月到 1 月。結果表格中顯示, 2 月 1 日是該季的第 1 天, 而 1 月 31 日是該季的第 92 天和最後一天。

範例 3 – 1 月是一年的開始 (圖表)

載入指令碼和圖表運算式

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 資料集與第一個範例中相同。
- 使用預設 DateFormat 系統變數 MM/DD/YYYY。

不過，在此範例中，不變的資料集會載入到應用程式中。會透過圖表物件中的量值計算該季日子的值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：date。

建立下列量值：

=daynumberofquarter(date)

結果表格

日期	=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

該年的第一天是 1 月 1 日，因為沒有第二引數傳遞至 `DayNumberOfQuarter()` 函數中。

1 月 1 日是該季的第 1 天，而 2 月 1 日是該季的第 32 天。3 月 31 日是該季的第 91 天和最後一天，而 4 月 1 日則是第 2 季的第 1 天。

範例 4 – 2 月是一年的開始 (圖表)

載入指令碼和圖表運算式

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 資料集與第一個範例中相同。
- 使用預設 `DateFormat` 系統變數 MM/DD/YYYY。
- 會計年度自 2 月 1 日起，至 1 月 31 日止。

不過，在此範例中，不變的資料集會載入到應用程式中。會透過圖表物件中的量值計算該季日子的值。

載入指令碼

```
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
];
```

圖表物件

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`date`。

建立下列量值：

```
=daynumberofquarter(date,2)
```

結果

結果表格

日期	=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

該年的第一天是 1 月 1 日, 因為傳遞至 `DayNumberOfQuarter()` 函數中的第二引數是 2。

該年的第一季從 2 月到 4 月, 而第四季則從 11 月到 1 月。結果表格中證明, 2 月 1 日是該季的第 1 天, 而 1 月 31 日是該季的第 92 天和最後一天。

daynumberofyear

此函數會計算時間戳記在當年的第幾天。該計算從當年第一天的第一毫秒開始進行, 但是第一個月可能發生位移。

語法:

```
DayNumberOfYear(timestamp[,start_month])
```

傳回的資料類型: 整數

引數

引數	描述
timestamp	要評估的時間戳記。
start_month	透過指定 **start_month** 介於 2 和 12 之間 (如果省略, 則為 1), 可將年初向前移到任何月份的第一天。例如, 如果要使用起始於 3 月 1 日的會計年度, 可指定 **start_month** = 3。

這些範例使用日期格式 **DD/MM/YYYY**。日期格式是在位於資料載入指令碼頂端的 **SET DateFormat** 陳述式中指定。變更範例中的格式, 以滿足您的需求。

函數範例

範例	結果
`DayNumberOfYear( '12/09/2014' )`	傳回 256，從年度第一天算起的天數。
`DayNumberOfYear( '12/09/2014',3 )`	傳回 196，從 3 月 1 日開始計數的天數。

範例 1 – 1 月是一年的開始 (指令碼)

載入指令碼和結果

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含日期清單的簡單資料集，載入到名為 `Calendar` 的表格中。使用預設 `DateFormat` 系統變數 MM/DD/YYYY。
- 使用 `DayNumberOfYear()` 函數建立附加欄位 (名稱為 `daynryear`) 的前置載入。

除了日期，不會向函數提供其他參數。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- daynryear

結果表格

日期	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

該年的第一天是 1 月 1 日，因為沒有第二引數傳遞至 `DayNumberOfYear()` 函數中。

1 月 1 日是該季的第 1 天，而 2 月 1 日是該年的第 32 天。6 月 30 日是該年的第 182 天，而 12 月 31 日則是該年的第 366 天和最後一天。

範例 2 – 11 月是一年的開始 (指令碼)

載入指令碼和結果

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 資料集與第一個範例中相同。
- 使用預設 `DateFormat` 系統變數 MM/DD/YYYY
- 在 11 月 1 日開始的 `start_month` 引數。這將會計年度設定為 11 月 1 日。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- daynryear

結果表格

日期	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

該年的第一天是 11 月 1 日,因為傳遞至 `DayNumberOfYear()` 函數中的第二引數是 11。

1 月 1 日是該季的第 1 天,而 2 月 1 日是該年的第 32 天。6 月 30 日是該年的第 182 天,而 12 月 31 日則是該年的第 366 天和最後一天。

範例 3 – 1 月是一年的開始 (圖表)

載入指令碼和圖表運算式

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 資料集與第一個範例中相同。
- 使用預設 `DateFormat` 系統變數 MM/DD/YYYY。

不過, 在此範例中, 不變的資料集會載入到應用程式中。會透過圖表物件中的量值計算該季日子的值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度: `date`。

建立下列量值:

`=daynumberofyear(date)`

結果表格

日期	=daynumberofyear(date)
01/01/2022	1
01/10/2022	10

日期	=daynumberofyear(date)
01/31/2022	31
02/01/2022	32
02/10/2022	41
06/30/2022	182
07/26/2022	208
10/31/2022	305
11/01/2022	306
12/31/2022	366

該年的第一天是 1 月 1 日, 因為沒有第二引數傳遞至 `DayNumberOfYear()` 函數中。

1 月 1 日是該年的第 1 天, 而 2 月 1 日是該年的第 32 天。6 月 30 日是該年的第 182 天, 而 12 月 31 日則是該年的第 366 天和最後一天。

範例 4 – 11 月是一年的開始 (圖表)

載入指令碼和圖表運算式

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 資料集與第一個範例中相同。
- 使用預設 `DateFormat` 系統變數 MM/DD/YYYY。
- 會計年度自 11 月 1 日起, 至 10 月 31 日止。

不過, 在此範例中, 不變的資料集會載入到應用程式中。會透過圖表物件中的量值計算該年日子的值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`date`。

建立下列量值：

`=daynumberofyear(date)`

結果表格

日期	=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

該年的第一天是 11 月 1 日，因為傳遞至 `DayNumberOfYear()` 函數中的第二引數是 11。

會計年度自 11 月起，至 10 月止。結果表格中顯示，11 月 1 日是該年的第 1 天，而 10 月 31 日是該年的第 366 天和最後一天。

daystart

此函數傳回的值相當於包含在 **time** 引數中該日的第一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 **TimestampFormat**。

語法：

```
DayStart(time[, [period_no[, day_start]])
```

傳回的資料類型: 雙值

引數

引數	描述
time	要評估的時間戳記。
period_no	**period_no** 是一個整數, 或者解析為整數的運算式, 其中值 0 表示包含 **time** 的日。負值的 **period_no** 表示之前的日, 正值表示之後的日。
day_start	若要指定日不從午夜開始, 可在 **day_start** 中指示位移 (以一日的片段表示)。例如, 0.125 表示上午 3:00。 換言之, 若要建立偏移, 請依 24 小時分割開始時間。例如, 對於在上午 7:00 開始的日子, 請使用分數 7/24。

什麼情況下使用

`daystart()` 函數通常在使用者想要計算使用一天中已經過的部分時, 作為運算式的一部分使用。例如, 這可用來計算員工目前為止在一天中所賺取的總工資, 即可使用此函數。

這些範例使用時間戳記格式 `'M/D/YYYY h:mm:ss[.fff] TT'`。時間戳記格式是在位於資料載入指令碼頂端的 `SET TimeStamp` 陳述式中指定。變更範例中的格式, 以滿足您的需求。

函數範例

範例	結果
`daystart('01/25/2013 4:45:00 PM')`	傳回 1/25/2013 12:00:00 AM。
`daystart('1/25/2013 4:45:00 PM', -1)`	傳回 1/24/2013 12:00:00 AM。
`daystart('1/25/2013 16:45:00',0,0.5 )`	傳回 1/25/2013 12:00:00 PM。

區域設定

除非另有說明, 否則此主題中的範例皆使用下列日期格式:MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素, 您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式, 以滿足您的需求。或者, 您可以在載入指令碼中變更格式, 以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典, 資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 - 簡單範例

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含日期清單的簡單資料集,載入到名為 Calendar 的表格中。
- 已使用預設的 TimeStampFormat 系統變數 ((M/D/YYYY h:mm:ss[.fff] TT)。
- 使用 daystart() 函數建立附加欄位 (名稱為 SOD_timestamp) 的前置載入。

除了日期,不會向函數提供其他參數。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- SOD_timestamp

結果表格

日期	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

如同在以上表格中所見，會為資料集中的每個日期產生日子結束時間戳記。時間戳記的格式是系統變數 `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`。

範例 2 - period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含停車罰款的資料集，這載入到名為 `Fines` 的表格中。資料集包括下列欄位：
  - `id`
  - `due_date`
  - `number_plate`
  - `amount`
- 使用 `daystart()` 函數並提供所有三個參數的前置載入：`time`、`period_no` 和 `day_start`。此前置載入建立下列兩個新的日期欄位：
  - `early_repayment_period` 日期欄位，在付款到期前七天開始。
  - `late_penalty_period` 日期欄位，在付款到期後 14 天開始。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- due_date
- early_repayment_period
- late_penalty_period

結果表格

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

新欄位的值位於 TimestampFormat M/DD/YYYY tt 中。因為使用函數 daystart()，時間戳記值都是一天中的第一毫秒。

由於 daystart() 函數中傳遞的第二引數為負值，提早還款期間值是到期日期前七天。

由於 daystart() 函數中傳遞的第二引數為正值，延遲還款期間值是到期日期後 14 天。

範例 3 - day_start

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 與上一個範例相同的資料集和情境。
- 與上一個範例相同的前置載入。

在此範例中, 我們將工作日設定為在每天上午 7:00 開始和結束。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- due_date
- early_repayment_period
- late_penalty_period

結果表格

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

日期現在有上午 7:00 的時間戳記，因為傳遞至 daystart() 函數的 day_start 引數值為 7/24。這將一日的開始設定為上午 7:00 。

因為 due_date 欄位沒有時間戳記，這會被視為上午 12:00，因此仍然屬於前一天，因為每天在上午 7:00 開始和結束。因此，於 2 月 11 日到期的罰款提早還款期間在 2 月 3 日上午 7:00 開始。

範例 4 - 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

此範例使用與上一個範例相同的資料集和情境。

不過，僅原始 Fines 表格會載入到應用程式中，且圖表物件中計算了兩個額外的到期日期值。

載入指令碼

```
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
];
```

結果

請執行下列動作：

1. 載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：due_date。
2. 若要建立 early_repayment_period 欄位，建立下列量值。
   =daystart(due_date,-7,7/24)
3. 若要建立 late_penalty_period 欄位，建立下列量值：
   =daystart(due_date,14,7/24)

結果表格

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

新欄位的值位於 TimestampFormat M/D/YYYY h:mm:ss[.fff] TT 中。因為使用 daystart() 函數，時間戳記值對應至一天中的第一毫秒。

由於 daystart() 函數中傳遞的第二引數為負值，提早還款期間值是到期日期前七天。

由於 daystart() 函數中傳遞的第二引數為正值，延遲還款期間值是到期日期後 14 天。

日期有上午 7:00 的時間戳記，因為傳遞至 daystart() 函數 day_start 的第三引數值為 7/24。

firstworkdate

firstworkdate 函數會傳回在不晚於 **end_date** 前結束，並考慮到所有選用的列出假日，而可達到 **no_of_workdays** (星期一至星期五) 的最近開始日期。**end_date** 和 **holiday** 應該是有效的日期或時間戳記。

語法：

```
firstworkdate(end_date, no_of_workdays {, holiday} )
```

傳回的資料類型: 整數

引數:

引數

引數	描述
end_date	要評估之結束日期的時間戳記。
no_of_ workdays	要達到的工作日數目。
holiday	要從工作日中排除的假期。假日以字串常數日期表示。您可以指定多個假日日期,以逗點分隔。 **範例: '12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'**

範例與結果:
這些範例使用日期格式 **DD/MM/YYYY**。日期格式是在位於資料載入指令碼頂端的 **SET DateFormat** 陳述式中指定。變更範例中的格式,以滿足您的需求。

指令碼處理範例

範例	結果
`firstworkdate ('29/12/2014', 9)`	傳回 '17/12/2014。
`firstworkdate ('29/12/2014', 9, '25/12/2014', '26/12/2014')`	傳回 '15/12/2014, 原因是將兩日假期納入考量。

範例:

將範例指令碼新增至您的應用程式並予以執行。若要查看結果,將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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;
```

產生的表格顯示表格中各記錄的 FirstWorkDate 傳回值。

結果表格

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

此函數會傳回目前 Greenwich Mean Time，如衍生自區域設定。該函數會以 `TimestampFormat` 系統變數格式傳回值。

載入應用程式時，將會根據從系統時鐘取得的最新格林威治標準時間調整任何使用 `GMT` 函數的載入指令碼表格、變數或圖表物件。

語法：

```
GMT( )
```

傳回的資料類型： 雙值

這些範例使用時間戳記格式 `M/D/YYYY h:mm:ss[.fff] TT`。日期格式是在位於資料載入指令碼頂端的 `SET TimestampFormat` 陳述式中指定。變更範例中的格式，以滿足您的需求。

函數範例

範例	結果
GMT()	3/28/2022 2:47:36 PM

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 - 變數 (指令碼)

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。此範例使用 GMT 函數將目前的格林威治標準時間設定為載入指令碼中的變數。

載入指令碼

```
LET vGMT = GMT();
```

結果

載入資料並建立工作表。使用**文字與影像**圖表物件建立文字方塊。

將此量值新增至文字方塊:

```
=vGMT
```

文字方塊應包含具有日期和時間的文字行, 類似以下所示:

```
3/28/2022 2:47:36 PM
```

範例 2 - 11 月是一年的開始 (指令碼)

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含過期圖書館書籍的資料集, 這載入到名為 Overdue 的表格中。使用預設 DateFormat 系統變數 MM/DD/YYYY。
- 建立名為 days_overdue 的新欄位, 這計算每本書過期了多少天。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- due_date
- book_id
- days_overdue

結果表格

due_date	book_id	days_overdue
01/01/2021	4	455
01/10/2021	24	446
01/31/2021	173	425
02/01/2021	281	424
02/10/2021	265	415
06/30/2021	465	275
07/26/2021	537	249
10/31/2021	275	152
11/01/2021	455	151
12/31/2021	46	91

days_overdue 欄位中的值的計算方式是使用 GMT() 函數尋找目前格林威治標準時間與原始到期日期之間的差異。為了僅計算天數，會使用 Floor() 函數，將結果四捨五入為最接近的整數。

範例 3 - 圖表物件 (圖表)

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。載入指令碼包含與上一個範例相同的資料集。使用預設 `DateFormat` 系統變數 MM/DD/YYYY。

不過，在此範例中，不變的資料集會載入到應用程式中。會透過圖表物件中的量值計算過期天數的值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- `due_date`
- `book_id`

建立下列量值：

```
=Floor(GMT() - due_date)
```

結果表格

due_date	book_id	=Floor(GMT()-due_date)
01/01/2021	4	455
01/10/2021	24	446

due_date	book_id	=Floor(GMT()-due_date)
01/31/2021	173	425
02/01/2021	281	424
02/10/2021	265	415
06/30/2021	465	275
07/26/2021	537	249
10/31/2021	275	152
11/01/2021	455	151
12/31/2021	46	91

`days_overdue` 欄位中的值的計算方式是使用 `GMT()` 函數尋找目前格林威治標準時間與原始到期日期之間的差異。為了僅計算天數，會使用 `Floor()` 函數，將結果四捨五入為最接近的整數。

hour

當 **expression** 的分數根據標準數字解譯的方式可解譯為時間時，此函數會傳回代表小時的整數。

語法：

```
hour(expression)
```

傳回的資料類型： 整數

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
`hour( '09:14:36' )`	套用的文字字串隱含轉換為時間戳記，因為這符合 TimestampFormat 變數中定義的時間戳記格式。運算式傳回 9。
`hour( '0.5555' )`	運算式傳回 13 (因為 0.5555 = 13:19:55)。

範例 1 – 變數 (指令碼)

載入指令碼和結果

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含依時間戳記之交易的資料集
- 預設 TimeStamp 系統變數 (M/D/YYYY h:mm:ss[.fff] TT)

建立欄位 'hour', 在進行購買時計算。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- hour

結果表格

日期	hour
2022-01-01 22:10:22	22
2022-01-02 08:35:54	8
2022-01-03 05:40:49	5
2022-01-03 14:21:53	14
2022-01-04 18:49:38	18
2022-01-04 22:58:34	22
2022-01-05 19:04:57	19
2022-01-05 19:34:46	19
2022-01-06 08:49:09	8
2022-01-06 11:29:38	11

使用 `hour()` 函數並傳遞日期作為前置 LOAD 陳述式中的運算式，以建立小時欄位中的值。

範例 2 – 圖表物件 (圖表)

載入指令碼和圖表運算式

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 資料集與第一個範例中相同。
- 預設 `TimeStamp` 系統變數 (`M/D/YYYY h:mm:ss[.fff] TT`)。

不過，在此範例中，不變的資料集會載入到應用程式中。會透過圖表物件中的量值計算 '`hour`' 值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：date。

若要計算 'hour'，建立下列量值：

=hour(date)

結果表格

due_date	=hour(date)
2022-01-01 22:10:22	22
2022-01-02 08:35:54	8
2022-01-03 05:40:49	5
2022-01-03 14:21:53	14
2022-01-04 18:49:38	18
2022-01-04 22:58:34	22
2022-01-05 19:04:57	19
2022-01-05 19:34:46	19
2022-01-06 08:49:09	8
2022-01-06 11:29:38	11

'hour' 值的建立方式是使用 hour() 函數並傳遞日期作為圖表物件量值中的運算式。

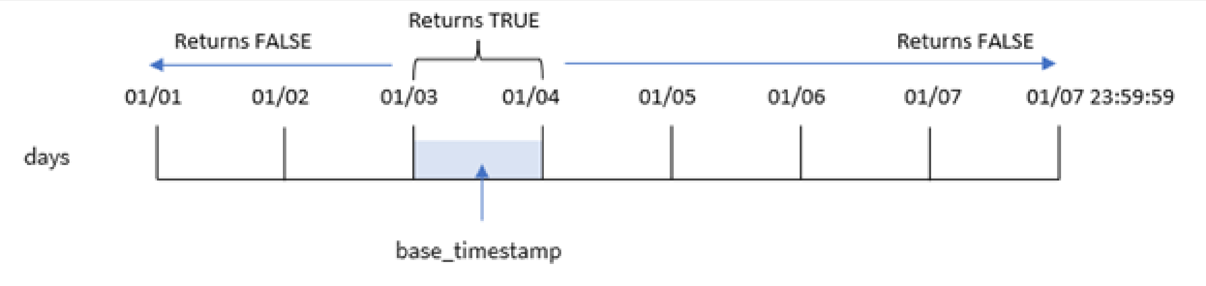
inday

此函數會傳回 True，前提是如果 **timestamp** 位於包含 **base_timestamp** 的日中。

語法：

```
InDay (timestamp, base_timestamp, period_no[, day_start])
```

inday 函數的圖表



inday() 函數使用 base_timestamp 引數來識別時間戳記落在哪一天。依照預設，一天的開始時間為午夜；但您可以使用 inday() 函數的 day_start 引數變更一天的開始時間。定義此日後，該函數將會在比較規定的時間戳記值與該日時傳回布林值結果。

什麼情況下使用

inday() 函數會傳回布林值結果。通常，此函數類型將作為 if expression 中的條件使用。這根據評估的日期是否發生於問題中的時間戳記當日，傳回彙總或計算。

例如，inday() 函數可用來識別指定日製作的所有設備。

傳回的資料類型： 布林

在 Qlik Sense 中，布林值 true 值以 -1 代表，而 false 值以 0 代表。

引數

引數	描述
timestamp	您想要與 base_timestamp 比較的日期和時間。
base_timestamp	用來評估時間戳記的日期和時間。
period_no	日可以使用 period_no 位移。period_no 是一個整數，其中值 0 表示包含 base_timestamp 的日。負值的 period_no 表示之前的日，正值表示之後的日。
day_start	如果要使用不起始於午夜的日，則可在 day_start 中指示位移 (以一日的小數表示)，例如 0.125 代表早上 3 時。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	傳回 True
inday ('01/12/2006 12:23:00 PM', '01/13/2006 12:00:00 AM', 0)	傳回 False
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	傳回 False
inday ('01/11/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	傳回 True
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0, 0.5)	傳回 False
inday ('01/12/2006 11:23:00 AM', '01/12/2006 12:00:00 AM', 0, 0.5)	傳回 True

範例 1 – LOAD 陳述式 (指令碼)

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含依時間戳記之交易的資料集，這載入到稱為 Transactions 的表格中。
- 以 TimeStamp 系統變數 (M/D/YYYY h:mm:ss[.fff] TT) 格式提供的日期欄位。
- 包含設定為 in_day 欄位之 inday() 函數的前置載入。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- in_day

結果表格

日期	in_day
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	0
01/04/2022 6:49:38 PM	0
01/04/2022 10:58:34 PM	0
01/05/2022 5:40:49 AM	-1
01/05/2022 11:29:38 AM	-1
01/05/2022 7:04:57 PM	-1
01/06/2022 8:49:09 AM	0

in_day 欄位在前置 LOAD 陳述式中的建立方式是使用 inday() 函數並傳遞日期欄位,即 1 月 5 日的硬式編碼時間戳記和值為 0 的 period_no 作為函數的引數。

範例 2 – period_no

載入指令碼和結果

概覽

載入指令碼使用與第一個範例中相同的資料集和情境。

不過,在此範例中,任務要計算交易日期是否發生在 1 月 5 日之前的兩天。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- in_day

結果表格

日期	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

在此實例中,因為值為 `-2` 的 `period_no` 作為 `inday()` 函數中的偏移引數使用,所以函數會決定每個交易日期是否發生在 1 月 3 日。這可以在輸出表格中驗證,其中有一筆交易傳回布林值結果 TRUE。

範例 3 – day_start

載入指令碼和結果

概覽

載入指令碼使用與先前範例中相同的資料集和情境。

不過，在此範例中，公司政策是工作日在上午 7 時開始和結束。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- in_day

結果表格

日期	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

日期	in_day
01/05/2022 7:04:57 PM	0
01/06/2022 8:49:09 AM	0

因為 inday() 函數中使用的 start_day 引數為 7/24, 亦即上午 7 時, 所以該函數會決定每個交易日期是否發生在 1 月 4 日上午 7 時和 1 月 5 日上午 7 時之間。

這可以在輸出表格中驗證, 其中發生在 1 月 4 日上午 7 時之後的交易傳回布林值結果 TRUE, 而發生在 1 月 5 日上午 7 時之後的交易則傳回布林值結果 。

範例 4 – 圖表物件

載入指令碼和圖表運算式

概覽

載入指令碼使用與先前範例中相同的資料集和情境。

不過, 在此範例中, 資料集保持不變並且會載入到應用程式中。您將會建立圖表物件中的量值, 藉此計算以決定交易是否發生在 1 月 5 日。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度:

- date

若要計算交易是否發生在 1 月 5 日, 建立下列量值:

```
=inday(date,'01/05/2022 12:00:00 AM',0)
```

結果表格

日期	inday(date,'01/05/2022 12:00:00 AM',0)
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	0
01/04/2022 6:49:38 PM	0
01/04/2022 10:58:34 PM	0
01/05/2022 5:40:49 AM	-1
01/05/2022 11:29:38 AM	-1
01/05/2022 7:04:57 PM	-1
01/06/2022 8:49:09 AM	0

範例 5 – 情境

載入指令碼和結果

概覽

在此範例中，這識別出由於設備錯誤，在 1 月 5 日製造的產品有瑕疵。最終使用者希望圖表物件依日期顯示哪些製造產品的狀態為「瑕疵」或「故障」，以及在 1 月 5 日製造的產品成本。

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入資料集並稱為「Products」的表格。
- 該表格含有下列欄位：
  - 產品 ID
  - 製造時間
  - 成本價格

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度:

=dayname(manufacture_date)

建立下列量值:

- =if(only(InDay(manufacture_date,makedate(2022,01,05),0)),'Defective','Faultless')
- =sum(cost_price)

將量值的**數字格式**設定為**金錢**。

在**外觀**之下,關閉**總計**。

結果表格

dayname (manufacture_date)	=if(only(InDay(manufacture_date,makedate (2022,01,05),0)),'Defective','Faultless')	=sum(cost_price)
01/01/2022	無缺失	44.67
01/02/2022	無缺失	36.34
01/03/2022	無缺失	51.75
01/04/2022	無缺失	89.69
01/05/2022	瑕疵	170.78
01/06/2022	無缺失	74.23

inday() 函數會在評估每個產品的製造日期時傳回布林值。對於在 1 月 5 日製造的任何產品, inday () 函數會傳回布林值 TRUE 並將產品標記為「瑕疵」。對於傳回 FALSE 值的任何產品 (因此這不是在該日製造), 會將產品標記為「無缺失」。

indaytotime

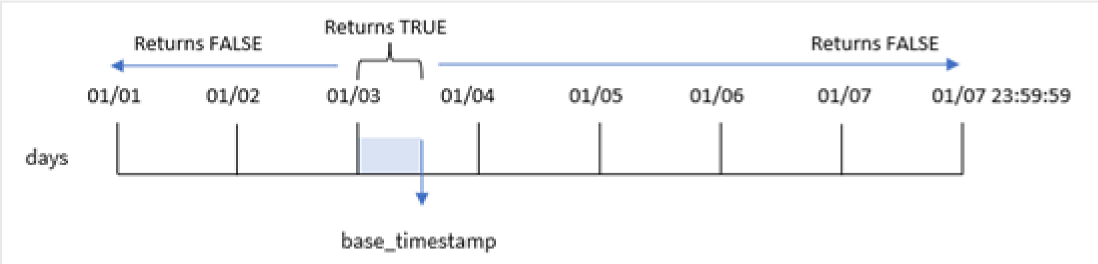
如果 **timestamp** 位於包含 **base_timestamp** 的日部分內, 且不超過 (可包含) **base_timestamp** 的確切的毫秒, 則此函數會傳回 True。

語法:

```
InDayToTime (timestamp, base_timestamp, period_no[, day_start])
```

indaytotime() 函數傳回依據時間戳記值在該日時段期間於何時發生所決定的布林值結果。此時段的開始界線是一天的開始, 預設為午夜;一天的開始可以透過 indaytotime() 函數的 day_start 引數來修改。一天時段的結束界線由函數的 base_timestamp 引數決定。

indaytotime 函數的圖表。



什麼情況下使用

indaytotime() 函數會傳回布林值結果。通常，此函數類型將作為 if expression 中的條件使用。indaytotime() 函數會截至 (含) 基本時間戳記的時間，根據時間戳記是否發生在該日時段內，傳回彙總或計算。

例如，indaytotime() 函數可用來顯示到今日為止演出之節目的票券銷售總和。

傳回的資料類型： 布林

在 Qlik Sense 中，布林值 true 值以 -1 代表，而 false 值以 0 代表。

引數

引數	描述
timestamp	您想要與 base_timestamp 比較的日期和時間。
base_timestamp	用來評估時間戳記的日期和時間。
period_no	日可以使用 period_no 偏移。period_no 是一個整數，其中值 0 表示包含 base_timestamp 的日。負值的 period_no 表示之前的日，正值表示之後的日。
day_start	(選用) 如果要使用不起始於午夜的日，則可在 day_start 中指示偏移 (以一日的小數表示)。例如，使用 0.125 表示上午 3 時。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
indaytotime ('01/12/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', 0)	傳回 True
indaytotime ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	傳回 False
indaytotime '01/11/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', -1)	傳回 True

範例 1 – 無其他引數

載入指令碼和結果

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 1 月 4 日和 1 月 5 日之間交易的資料集載入到稱為「Transactions」的表格中。
- 以 TimeStamp 系統變數 (M/D/YYYY h:mm:ss[.fff] TT) 格式提供的日期欄位。
- 包含 indaytotime() 函數的前置載入，這設定為 'in_day_to_time' 欄位，決定每筆交易是否發生在上午 9:00 之前。

載入指令碼

```
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
];
```

結果

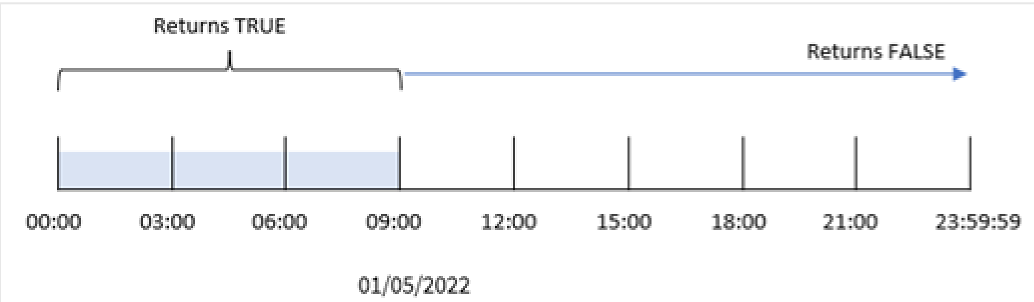
載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- in_day_to_time

結果表格

日期	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

範例 1 具有上午 9:00 限制的 `indaytotime` 函數圖表。



`in_day_to_time field` 在前置 LOAD 陳述式中的建立方式是使用 `indaytotime()` 函數並傳遞日期欄位，即 1 月 5 日上午 9:00 的硬式編碼時間戳記和值為 0 的偏移作為函數的引數。任何發生在 1 月 5 日午夜和上午 9:00 之間的交易都會傳回 TRUE。

範例 2 – period_no

載入指令碼和結果

概覽

載入指令碼使用與第一個範例中相同的資料集和情境。

不過，在此範例中，您將計算交易日期是否發生在 1 月 5 日上午 9:00 前一天。

載入指令碼

```
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
];
```

結果

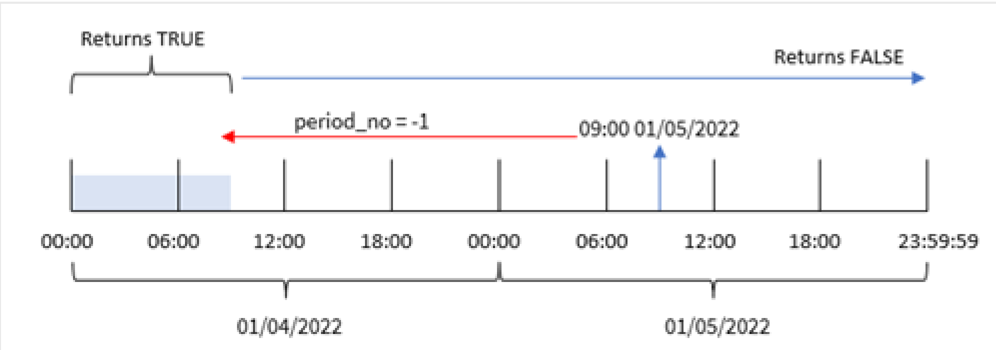
載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- in_day_to_time

結果表格

日期	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

範例 2 具有自 1 月 4 日以來交易的 `indaytotime` 函數圖表。



在此範例中，因為值為 -1 的偏移作為 `indaytotime()` 函數中的偏移引數使用，所以函數會決定每個交易日期是否發生在 1 月 4 日上午 9:00 之前。這可以在輸出表格中驗證，其中有一筆交易傳回布林值結果 TRUE。

範例 3 – day_start

載入指令碼和結果

概覽

使用與第一個範例相同的資料集和情境。

不過，在此範例中，公司政策是工作日在上午 8 時開始和結束。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

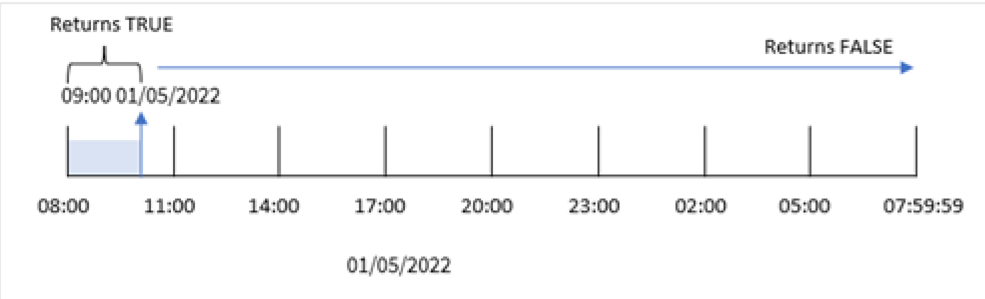
- date
- in_day_to_time

結果表格

日期	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

日期	in_day_to_time
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	0
01/05/2022 8:44:36 AM	-1
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

範例 3 具有自上午 8:00 至上午 9:00 之交易的 `indaytotime` 函數圖表



因為 `indaytotime()` 函數中使用的 `start_day` 引數為 8/24，等同於上午 8:00，所以每天於上午 8:00 開始和結束。因此，`indaytotime()` 函數將會為發生在 1 月 5 日上午 8:00 和上午 9:00 之間的任何交易傳回布林值結果 TRUE。

範例 4 – 圖表物件

載入指令碼和圖表運算式

概述

使用與第一個範例相同的資料集和情境。

不過，在此範例中，資料集保持不變並且會載入到應用程式中。您將會建立圖表物件中的量值，藉此計算以決定交易是否發生在 1 月 5 日上午 9:00 之前。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度:

date。

若要決定交易是否發生在 1 月 5 日上午 9:00 之前，建立下列量值:

=indaytotime(date,'01/05/2022 9:00:00 AM',0)

結果表格

日期	=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

in_day_to_time 量值在圖表物件中的建立方式是使用 indaytotime() 函數並傳遞日期欄位，即 1 月 5 日上午 9:00 的硬式編碼時間戳記和值為 0 的偏移作為函數的引數。任何發生在 1 月 5 日午夜和上午 9:00 之間的交易都會傳回 TRUE。這在結果表格中驗證。

範例 5 – 情境

載入指令碼和結果

概覽

在此範例中，包含當地電影院票券銷售的資料集載入到稱為 Ticket_Sales 的表格中。今日為 2022 年 5 月 3 日，現在是上午 11:00。

使用者希望 KPI 圖表物件顯示從截至今日演出的所有節目所賺取的收益。

載入指令碼

```
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
];
```

結果

請執行下列動作：

3. 建立 KPI 物件。
4. 使用 indaytotime() 函數建立將會顯示截至今日演出節目的所有票券銷售總和的量值：

```
=sum(if(indaytotime([show time],'05/03/2022 11:00:00 AM',0),[ticket price],0))
```

3. 建立 KPI 物件的標籤「目前收益」。
4. 將量值的**數字格式**設定為**金錢**。

截至 2022 年 5 月 3 日上午 11:00 的票券銷售總和總計為 $52.50。

`indaytotime ()` 函數會在比較每項票券銷售的節目時間與目前時間 ('05/03/2022 11:00:00 AM') 時傳回布林值。對於 5 月 3 日在上午 11:00 之前的任何節目，`indaytotime ()` 函數會傳回布林值 TRUE，其票券價格將會包括在總和總計中。

inlunarweek

此函數會判定 **timestamp** 是否位於包含 **base_date** 的農曆週內。將 1 月 1 日視為該週的第一天，以定義 Qlik Sense 中的農曆週。除了該年的最後一週，每週都會確切包含七天。

語法：

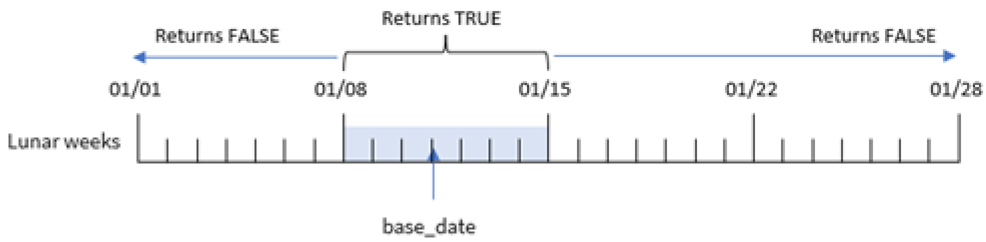
```
InLunarWeek (timestamp, base_date, period_no[, first_week_day])
```

傳回的資料類型： 布林

> *在 Qlik Sense 中，布林值 true 值以 -1 代表，而 false 值以 0 代表。*

`inlunarweek()` 函數判定 `base_date` 落在哪一個農曆週。然後，判定每個時間戳記值是否發生在與 `base_date` 相同的農曆週期間後，就會傳回布林值結果。

`inlunarweek()` 函數的圖表



什麼情況下使用

`inlunarweek()` 函數會傳回布林值結果。通常，此函數類型將作為 IF 運算式中的條件使用。這會根據評估的日期是否發生於有疑問的農曆週期間，傳回彙總或計算。

例如，`inlunarweek()` 函數可用來識別特定農曆週製作的所有設備。

引數

引數	描述
timestamp	您要與 **base_date** 比較的日期。
base_date	用來評估農曆週的日期。
period_no	農曆週可以使用 **period_no** 位移。period_no 是一個整數，其中值 0 表示包含 **base_date** 的農曆週。負值的 **period_no** 表示之前的農曆週，正值表示之後的農曆週。
first_week_day	可能大於或小於零的位移。這會使用指定的天數及/或一天的片段變更一年的開始。

函數範例

範例	結果
`inlunarweek ('01/12/2013', '01/14/2013', 0)`	傳回 `TRUE`，因為 `timestamp`、01/12/2013 的值介於週 01/08/2013 至 01/14/2013 期間。
`inlunarweek ('01/12/2013', '01/07/2013', 0)`	傳回 `FALSE`，因為 `base_date` 01/07/2013 在定義為 01/01/2013 至 01/07/2013 的農曆週。
`inlunarweek ('01/12/2013', '01/14/2013', -1)`	傳回 `FALSE`。將 period_no 的值指定為 -1，會將週變更為前一週，即 01/01/2013 變更為 01/07/2013。
`inlunarweek ('01/07/2013', 01/14/2013', -1)`	傳回 `TRUE`。與前一個範例相比較，`timestamp` 是在考量往回偏移之後的下一週內。
`inlunarweek ('01/11/2006', '01/08/2006', 0, 3)`	傳回 `FALSE`。為 `first_week_day` 指定值 3 表示會從 01/04/2013 計算一年的開始。因此，`base_date` 的值落在第一週，而 `timestamp` 的值落在週 01/11/2013 至 01/17/2013。

`inlunarweek()` 函數通常用於組合下列函數：

相關函數

函數	互動
lunarweekname (page 808)	此函數用來判定輸入日期發生的該年農曆週數。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 - 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 1 月交易的資料集, 這載入到稱為 `Transactions` 的表格中。
- 已使用 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供日期欄位。

建立欄位 `in_lunar_week`, 這判定交易是否發生在與 1 月 10 日相同的農曆週。

載入指令碼

```
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
];
```

結果

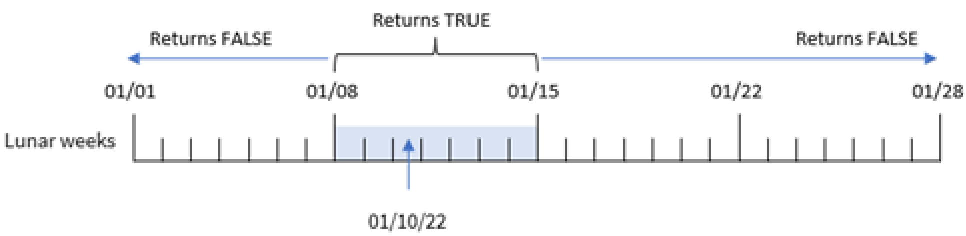
載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- in_lunar_week

結果表格

日期	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

inlunarweek() 函數，基本範例



in_lunar_week 欄位在前置 LOAD 陳述式中的建立方式是使用 inlunarweek()，然後傳遞以下內容作為函數的引數：

- date 欄位
- 1 月 10 日的硬式編碼日期, 如 base_date
- 0 的 period_no

因為農曆週於 1 月 1 日開始, 所以 1 月 10 日會落在 1 月 8 日開始、1 月 14 日結束的農曆週。因此, 發生在 1 月這兩個日期之間的任何交易會傳回布林值 TRUE。這在結果表格中驗證。

範例 2 - period_no

範例與結果:

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 與第一個範例相同的資料集和情境。
- 已使用 DateFormat 系統變數 (MM/DD/YYYY) 格式提供日期欄位。

不過, 在此範例中, 任務是建立欄位 2_lunar_weeks_later, 這判定交易是否發生在 1 月 10 日的兩個農曆週之後。

載入指令碼

```
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
];
```

結果

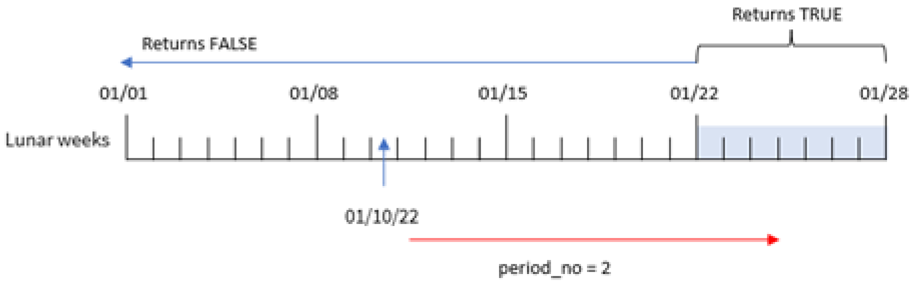
載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- 2_lunar_weeks_later

結果表格

日期	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

inlunarweek() 函數, period_no 範例



在此例中，因為值為 2 的 period_no 作為 inlunarweek() 函數中的偏移引數使用，所以該函數將 1 月 22 日開始的週定義為農曆週，以據此驗證交易。因此，任何發生在 1 月 22 日和 1 月 28 日之間的交易會傳回 TRUE 的布林值結果。

範例 3 - first_week_day

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼使用與第一個範例相同的資料集和情境。不過，在此範例中，我們將農曆週設定為在 1 月 6 月開始。

- 與第一個範例相同的資料集和情境。
- 使用預設 DateFormat 系統變數 MM/DD/YYYY。
- first_week_day 引數為 5。這將農曆週設定為在 1 月 5 日開始。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

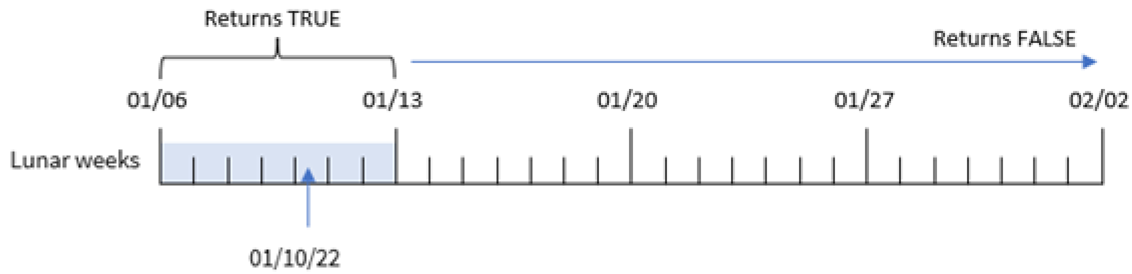
- `date`
- `in_lunar_week`

結果表格

日期	in_lunar_week
1/5/2022	0
1/6/2022	-1
1/7/2022	-1
1/8/2022	-1
1/9/2022	-1
1/10/2022	-1
1/11/2022	-1
1/12/2022	-1
1/13/2022	0
1/14/2022	0
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0

日期	in_lunar_week
1/22/2022	0
1/23/2022	0

inlunarweek() 函數, first_week_day 範例



在此例中，因為值為 5 的 `first_week_date` 引數用於 `inlunarweek()` 函數，所以這會將農曆週的開始偏移至 1 月 6 日。因此，1 月 10 日落在 1 月 6 日開始、1 月 12 日結束的農曆週。任何落在這兩個日期之間的交易會傳回 `TRUE` 的布林值。

範例 4 - 圖表物件

載入指令碼和圖表運算式：

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 已使用 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供日期欄位。

不過，在此範例中，不變的資料集會載入到應用程式中。會建立決定交易是否發生在與 1 月 10 日相同之農曆週的計算，作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`date`。

若要計算交易是否發生在包含 1 月 10 日的農曆週，建立下列量值：

`= inlunarweek(date,'01/10/2022', 0)`

結果表格

日期	=inlunarweek(date,'01/10/2022', 0)
1/5/2022	0
1/6/2022	0
1/7/2022	0
1/8/2022	-1
1/9/2022	-1
1/10/2022	-1
1/11/2022	-1
1/12/2022	-1
1/13/2022	-1
1/14/2022	-1
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0

日期	=inlunarweek(date,'01/10/2022', 0)
1/20/2022	0
1/21/2022	0
1/22/2022	0
1/23/2022	0

範例 5 - 情境

載入指令碼和圖表運算式:

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 載入到稱為 `Products` 之表格的資料集。
- 包含產品 ID、製造日期和成本價格的資訊。

這識別出由於設備錯誤, 在包含 1 月 12 日之農曆週製造的產品有瑕疵。最終使用者希望圖表物件依農曆週名稱顯示製造的產品狀態為「瑕疵」或「無缺失」, 以及在該月製造的產品成本。

載入指令碼

```
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
];
```

結果

請執行下列動作:

1. 載入資料並開啟工作表。建立新的表格。
2. 建立維度以顯示月份名稱:
   `=lunarweekname(manufacture_date)`
3. 使用 `inlunarweek()` 函數建立量值以識別哪些產品為瑕疵, 那些為無缺失:
   `=if(only(inlunarweek(manufacture_date,makedate(2022,01,12),0)), 'Defective','Faultless')`
4. 建立量值以加總產品的 cost_price:
   `=sum(cost_price)`
5. 將量值的**數字格式**設定為**金錢**。
6. 在**外觀**之下, 關閉**總計**。

結果表格

lunarweekname (manufacture_date)	=if(only(inlunarweek(manufacture_date,makedate (2022,01,12),0)), 'Defective','Faultless')	sum(cost_ price)
2022/01	無缺失	$125.79
2022/02	瑕疵	$316.38
2022/03	無缺失	$455.75
2022/04	無缺失	$146.09

`inlunarweek()` 函數會在評估每個產品的製造日期時傳回布林值。對於在包含 1 月 10 日之農曆週製造的任何產品, `inlunarweek()` 函數會傳回布林值 TRUE 並將產品標記為「瑕疵」。對於傳回 FALSE 值的任何產品 (因此這不是在該週製造), 會將產品標記為「無缺失」。

inlunarweektodate

此函數會發現 **timestamp** 是否位於農曆週部分內, 且不超過 (可包含) **base_date** 的最後一毫秒。將 1 月 1 日視為該週的第一天, 以定義 Qlik Sense 中的農曆週, 而除了該年的最後一週, 會確切包含七天。

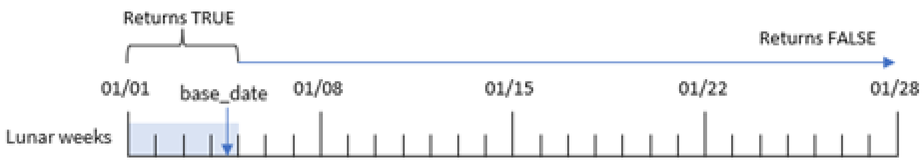
語法:

```
InLunarWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

傳回的資料類型: 布林

在 Qlik Sense 中, 布林值 true 值以 -1 代表, 而 false 值以 0 代表。

inlunarweektodate() 函數的範例圖表



inlunarweektodate() 函數作為農曆週的結束點運作。相反地，inlunarweek() 函數判定 base_date 落在哪一個農曆週。例如，若 base_date 為 1 月 5 日，則 1 月 1 日和 1 月 5 日之間的任何時間戳記會傳回布林值結果 TRUE，而 1 月 6 日和 7 日及之後的日期則會傳回布林值結果 FALSE。

引數

引數	描述
timestamp	您要與 **base_date** 比較的日期。
base_date	用來評估農曆週的日期。
period_no	農曆週可以使用 **period_no** 位移。period_no 是一個整數，其中值 0 表示包含 **base_date** 的農曆週。負值的 **period_no** 表示之前的農曆週，正值表示之後的農曆週。
first_week_day	可能大於或小於零的位移。這會使用指定的天數及/或一天的片段變更一年的開始。

什麼情況下使用

inlunarweektodate() 函數會傳回布林值結果。通常，此函數類型將作為 IF 運算式中的條件使用。若使用者希望計算根據評估的日期是否發生在有疑問的特定週區段期間，傳回彙總或計算，則會使用 inlunarweektodate() 函數。

例如，inlunarweektodate() 函數可用來識別特定週截至 (含) 特定日期前製作的所有設備。

函數範例

範例	結果
inlunarweektodate ('01/12/2013', '01/13/2013', 0)	傳回 TRUE，因為 timestamp、01/12/2013 的值介於週 01/08/2013 至 01/13/2013 的部分期間。
inlunarweektodate ('01/12/2013', '01/11/2013', 0)	傳回 FALSE，因為 timestamp 的值晚於 base_date 值，即便兩個日期都在 01/12/2012 之前的同一農曆週內。
inlunarweektodate ('01/12/2006', '01/05/2006', 1)	傳回 TRUE。為 period_no 指定值 1 會將 base_date 向前移動一週，因此 timestamp 的值在農曆週部分中。

inlunarweektodate() 函數通常用於組合下列函數：

相關函數

函數	互動
lunarweekname (page 808)	此函數用來判定輸入日期發生的該年農曆週數。

區域設定

除非另有說明,否則此主題中的範例皆使用下列日期格式:MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素,您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式,以滿足您的需求。或者,您可以在載入指令碼中變更格式,以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典,資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 - 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 1 月交易的資料集,這載入到稱為 `Transactions` 的表格中。使用預設 `DateFormat` 系統變數 MM/DD/YYYY。
- 建立欄位 `in_lunar_week_to_date`,這判定哪些交易發生在截至 1 月 10 日的農曆週。

載入指令碼

```
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
];
```

結果

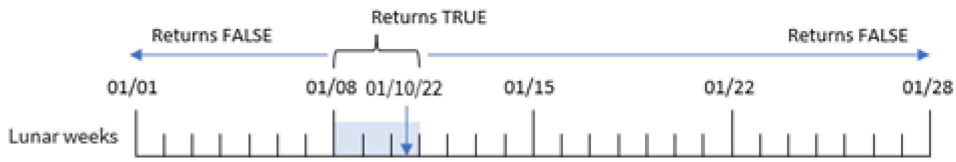
載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- in_lunar_week_to_date

結果表格

日期	in_lunar_week_to_date
1/1/2022	0
1/4/2022	0
1/10/2022	-1
1/11/2022	0
1/12/2022	0
1/15/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	0
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

inlunarweektodate() 函數，無其他引數



in_lunar_week_to_date 欄位在前置 LOAD 陳述式中的建立方式是使用 inlunarweektodate() 函數並傳遞 date 欄位，即 1 月 10 日的硬式編碼作為我們的 base_date，而 0 偏移作為函數的引數。

因為農曆週於 1 月 1 日開始，所以 1 月 10 日會落在於 1 月 8 日開始的農曆週；又因為我們使用 inlunarweektodate() 函數，所以該農曆週會結束於 10 日。因此，發生在 1 月這兩個日期之間的任何交易會傳回布林值 TRUE。這在結果表格中驗證。

範例 2 - period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。不過，在此範例中，任務是建立欄位 2_lunar_weeks_later，這判定交易是否發生在截至 1 月 1 日之農曆週的兩週之後。

載入指令碼

```
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
];
```

結果

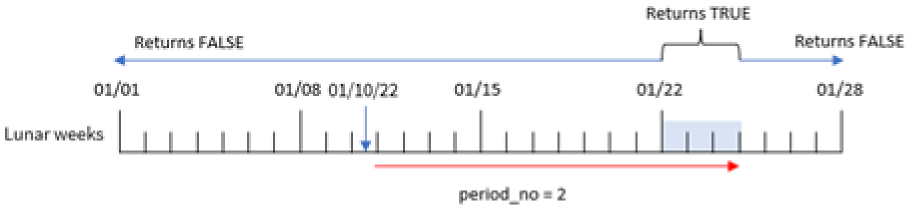
載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- 2_lunar_weeks_later

結果表格

日期	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

inlunarweektodate() 函數，period_no 範例



在此例中，inlunarweektodate() 函數判定截至 1 月 10 日的農曆週等於三天 (1 月 8 日、9 日、10 日)。由於值為 2 的 period_no 用來作為偏移引數，此農曆週偏移了 14 天。因此，這將為期三天的農曆週定義為包括 1 月 22 日、23 日和 24 日。任何發生在 1 月 22 日和 1 月 24 日之間的交易會傳回 TRUE 的布林值結果。

範例 3 - first_week_day

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 使用預設 DateFormat 系統變數 MM/DD/YYYY。
- first_week_date 引數為 3。這將農曆週設定為在 1 月 3 日開始。

載入指令碼

```
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
];
```

結果

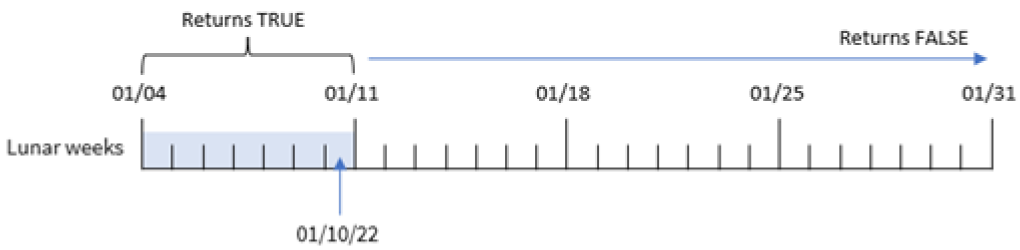
載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- `date`
- `in_lunar_week_to_date`

結果表格

日期	in_lunar_week_to_date
1/1/2022	0
1/4/2022	-1
1/10/2022	-1
1/11/2022	0
1/12/2022	0
1/15/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	0
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

`inlunarweektodate()` 函數, first_week_day 範例



在此例中,因為值為 3 的 `the first_week_date` 引數用於 `inlunarweek()` 函數,所以第一個農曆週將是自 1 月 3 日至 1 月 10 日。因為 1 月 10 日也是 `base_date`,所以任何落在這兩個日期之間的交易會傳回 `TRUE` 的布林值。

範例 4 - 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。

不過，在此範例中，不變的資料集會載入到應用程式中。會建立決定交易是否發生在截至 1 月 10 日之農曆週的計算，作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`date`。

建立下列量值：

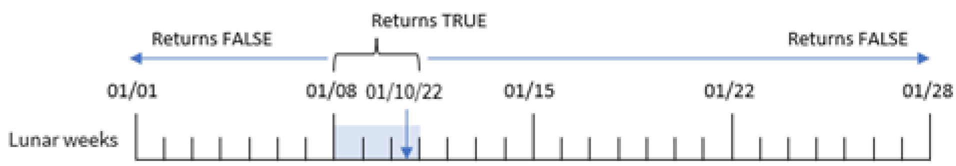
`=inlunarweektodate(date,'01/10/2022', 0)`

結果表格

日期	=inlunarweektodate(date,'01/10/2022', 0)
1/1/2022	0

日期	=inlunarweektodate(date,'01/10/2022', 0)
1/4/2022	0
1/10/2022	-1
1/11/2022	0
1/12/2022	0
1/15/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	0
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

inlunarweektodate() 函數，圖表物件範例



in_lunar_week_to_date 量值在圖表物件中的建立方式是使用 inlunarweektodate() 函數並傳遞日期欄位，即 1 月 10 日的硬式編碼日期作為 base_date，而 0 偏移作為函數的引數。

因為農曆週於 1 月 1 日開始，所以 1 月 10 日會落在於 1 月 8 日開始的農曆週，由於我們使用 inlunarweektodate() 函數，所以該農曆週會終止於 10 日。因此，發生在 1 月這兩個日期之間的任何交易會傳回布林值 TRUE。這在結果表格中驗證。

範例 5 - 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入到稱為 `Products` 之表格的資料集。
- 包含產品 ID、製造日期和成本價格的資訊。

這識別出由於設備錯誤，在 1 月 12 日該農曆週製造的產品有瑕疵。問題已在 1 月 13 日解決。最終使用者希望圖表物件依週顯示製造的產品狀態為「瑕疵」或「無缺失」，以及在該週製造的產品成本。

載入指令碼

```
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
];
```

結果

請執行下列動作：

1. 載入資料並開啟工作表。建立新的表格。
2. 建立維度以顯示週名稱：
   `=weekname(manufacture_date)`
3. 接下來，建立使用 `inlunarweektodate()` 函數的維度，以識別哪些產品為瑕疵，那些為無缺失：
   `=if(inlunarweektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')`
4. 建立量值以加總產品的 `cost_price`：
   `=sum(cost_price)`
5. 將量值的**數字格式**設定為**金錢**。

結果表格

=lunarweekname (manufacture_date)	=if(InLunarWeekToDate(manufacture_date,makedate (2022,01,12),0),'Defective','Faultless')	=Sum(cost_ price)
2022/01	無缺失	$142.67
2022/02	瑕疵	$320.88
2022/02	無缺失	$141.82
2022/03	無缺失	$214.64
2022/04	無缺失	$147.46
2022/05	無缺失	$248.12

`inlunarweektodate()` 函數會在評估每個產品的製造日期時傳回布林值。對於傳回布林值 `TRUE` 的項目，會將產品標記為 `'Defective'`。對於傳回值 `FALSE` 的任何產品，以及因此不是在該農曆週截至 1 月 12 日製作的產品，會將產品標記為 `'Faultless'`。

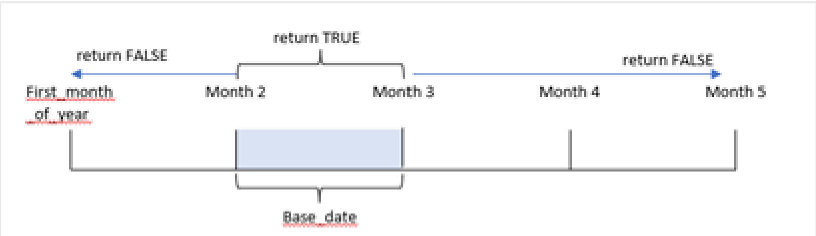
inmonth

此函數會傳回 True，前提是如果 **timestamp** 位於包含 **base_date** 的月份中。

語法：

```
InMonth (timestamp, base_date, period_no)
```

`indaytotime` 函數的圖表。



換言之，`inmonth()` 函數決定一組日期是否落在此月份當中，並根據識別該月份的 `base_date` 傳回布林值。

什麼情況下使用

`inmonth()` 函數會傳回布林值結果。通常，此函數類型將作為 `if expression` 中的條件使用。這根據日期是否發生於該月份中 (包括問題中的日期)，傳回彙總或計算。

例如，`inmonth()` 函數可用來識別在特定月份製造的所有設備。

傳回的資料類型： 布林

在 Qlik Sense 中，布林值 true 值以 -1 代表，而 false 值以 0 代表。

引數

引數	描述
時間戳記	您要與 `base_date` 比較的日期。
base_date	用來評估月份的日期。注意，`base_date` 可以是月份內的任何日子，這很重要。
period_no	月份可以使用 `period_no` 位移。`period_no` 是一個整數，其中值 0 表示包含 `base_date` 的月份。負值的 `period_no` 表示之前的月份，正值表示之後的月份。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
`inmonth ('25/01/2013', '01/01/2013', 0)`	傳回 True
`inmonth('25/01/2013', '23/04/2013', 0)`	傳回 False
`inmonth ('25/01/2013', '01/01/2013', -1)`	傳回 False
`inmonth ('25/12/2012', '17/01/2013', -1)`	傳回 True

範例 1 – 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 資料集包含 2022 年上半年的一組交易。
- 具有其他變數「`in_month`」的前置載入，可決定交易是否發生在 4 月。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- in_month

函數範例

日期	in_month
1/10/2022	0
1/14/2022	0
1/20/2022	0
1/22/2022	0
2/1/2022	0
2/2/2022	0
2/20/2022	0
4/11/2022	-1
4/13/2022	-1

日期	in_month
4/15/2022	-1
4/25/2022	-1
5/20/2022	0
5/22/2022	0
6/19/2022	0
6/22/2022	0

「in_month」欄位在前置 LOAD 陳述式中的建立方式是使用 inmonth() 函數並傳遞日期欄位，即 4 月 1 日的硬式編碼日期 (作為 base_date) 和值為 0 的 period_no 作為函數的引數。

base_date 識別將會傳回布林值結果 TRUE 的月份。因此，發生在 4 月的所有交易都會傳回 TRUE，這會在結果表格中驗證。

範例 2 – period_no

載入指令碼和結果

概覽

使用與第一個範例相同的資料集和情境。

不過，在此範例中，您將會建立欄位「2_months_prior」，這決定交易是否發生在 4 月前兩個月。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- 2_months_prior

函數範例

日期	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

使用 -2 作為 `inmonth()` 函數中的 `period_no` 引數，讓 `base_date` 引數定義的月份往前移動兩個月。在此範例中，這會將定義的月份從 4 月變更為 2 月。

因此，發生在 2 月的任何交易將會傳回布林值結果 TRUE。

範例 3 – 圖表物件

載入指令碼和圖表運算式

概述

使用與先前範例相同的資料集和情境。

不過，在此範例中，資料集保持不變並且會載入到應用程式中。會建立決定交易是否發生在 4 月的計算，作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

圖表物件

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：

`date`

若要計算交易是否發生在 4 月，建立下列量值：

`=inmonth(date,'04/01/2022', 0)`

結果

函數範例

日期	=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

日期	=inmonth(date,'04/01/2022', 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

範例 4 – 情境

載入指令碼和結果

概覽

在此範例中，資料集會載入到稱為「`Products`」的表格中。該表格含有下列欄位：

- 產品 ID
- 製造日期
- 成本價格

由於設備錯誤，在 2022 年 7 月製造的產品有瑕疵。問題已於 2022 年 7 月 27 日解決。

最終使用者希望圖表依月份顯示製造時為「瑕疵」(布林值 TURE) 或「故障」(布林值 FALSE) 的產品狀態，以及在該月份製造的產品成本。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：

=monthname(manufacture_date)
建立下列量值

- =sum(cost_price)
- =if(only(inmonth(manufacture_date,makedate(2022,07,01),0)),'Defective','Faultless')

1. 將量值的**數字格式**設定為**金錢**。
2. 在**外觀**之下，關閉**總計**。

結果表格

monthname (manufacture_date)	=if(only(inmonth(manufacture_date,makedate (2022,07,01),0)),'Defective','Faultless')	sum(cost_ price)
2022 年 1 月	無缺失	$54.40
2022 年 2 月	無缺失	$145.69
2022 年 3 月	無缺失	$53.80
2022 年 4 月	無缺失	$82.06
2022 年 5 月	無缺失	$127.60
2022 年 6 月	無缺失	$141.82
2022 年 7 月	瑕疵	$214.64
2022 年 8 月	無缺失	$147.46
2022 年 9 月	無缺失	$84.21
2022 年 10 月	無缺失	$163.91

inmonth() 函數會在評估每個產品的製造日期時傳回布林值。對於在 2022 年 7 月製造的任何產品，inmonth() 函數會傳回布林值 True 並將產品標記為「瑕疵」。對於傳回 False 值的任何產品 (因此這不是在 7 月製造)，會將產品標記為「無缺失」。

inmonths

此函數會發現時間戳記是否落在與基礎日期相同的月、雙月、季度、四月期或半年的期間內。還可以發現時間戳記是否在前一個或後一個時間期間內。

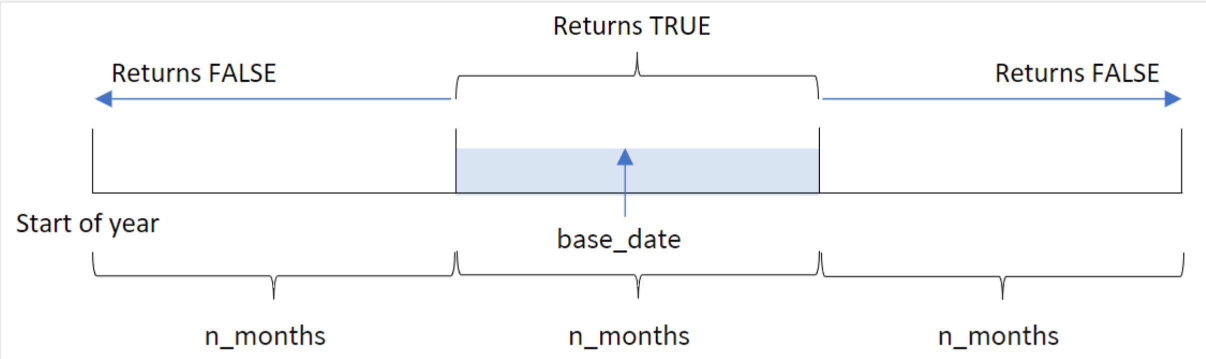
語法：

```
InMonths(n_months, timestamp, base_date, period_no [, first_month_of_year])
```

傳回的資料類型： 布林

在 Qlik Sense 中，布林值 true 值以 -1 代表，而 false 值以 0 代表。

inmonths() 函數的圖表



inmonths() 函數根據提供的 n_months 引數將該年分為幾個區段。然後這會判定評估的每個時間戳記是否落在與 base_date 引數相同的區段。不過，若提供 period_no 引數，該函數會判定時間戳記落在 base_date 的上一個或下一個期間。

一年中的下列區段可作為 n_month 引數用於函數。

n_month 引數

期間	月數
月	1
兩個月	2
季	3
四個月	4
半年	6

什麼情況下使用

inmonths() 函數會傳回布林值結果。通常，此函數類型將作為 if expression 中的條件使用。透過使用 inmonths() 函數，您可以選取想要評估的期間。例如，讓使用者識別在特定期間的月份、季度或半年製造的產品。

傳回的資料類型： 布林

在 Qlik Sense 中，布林值 true 值以 -1 代表，而 false 值以 0 代表。

引數

引數	描述
n_months	定義期間的月數。整數或解析為整數的運算式，必須是下列其中一項：1 (相當於 inmonth() 函數)、2 (雙月)、3 (相當於 inquarter() 函數)、4 (四月期) 或 6 (半年)。
timestamp	您要與 **base_date** 比較的日期。
base_date	用來評估期間的日期。
period_no	期間可以使用 **period_no** 位移，是一個整數或解譯為整數的運算式，其中值 0 表示包含 **base_date** 的期間。負值的 **period_no** 表示之前的期間，正值表示之後的期間。
first_month_of_year	如果要使用不起始於 1 月的 (會計) 年度，可在 **first_month_of_year** 中指定介於 2 和 12 之間的值。

您可以在 `first_month_of_year` 引數中使用下列值設定一年的第一個月：

first_month_of_year 值

月	值
二月	2
三月	3
四月	4
五月	5
六月	6
七月	7
八月	8
九月	9
十月	10
十一月	11
十二月	12

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
`inmonths(4, '01/25/2013', '04/25/2013', 0)`	傳回 TRUE。因為時間戳記 01/25/2013 的值在 01/01/2013 至 04/30/2013 四個月期間內，base_date 04/25/2013 的值也在該期間內。
`inmonths(4, '05/25/2013', '04/25/2013', 0)`	傳回 FALSE。因為 05/25/2013 處在與前一個範例相同的期間之外。
`inmonths(4, '11/25/2012', '02/01/2013', -1 )`	傳回 TRUE。因為 period_no, -1 的值將搜尋期間往回偏移四個月期間 (n-months 的值)，這讓搜尋期間從 09/01/2012 變更為 12/31/2012。
`inmonths(4, '05/25/2006', '03/01/2006', 0, 3)`	傳回 TRUE。因為 first_month_of_year 的值設為 3，這將搜尋期間從 03/01/2006 成為 07/30/2006，而非從 01/01/2006 成為 04/30/2006。

範例 1 - 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為「Transactions」的表格中。
- 具有其他變數「in_months」的前置載入，這判定哪些交易發生在與 2022 年 5 月 15 日相同的季度。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- in_months

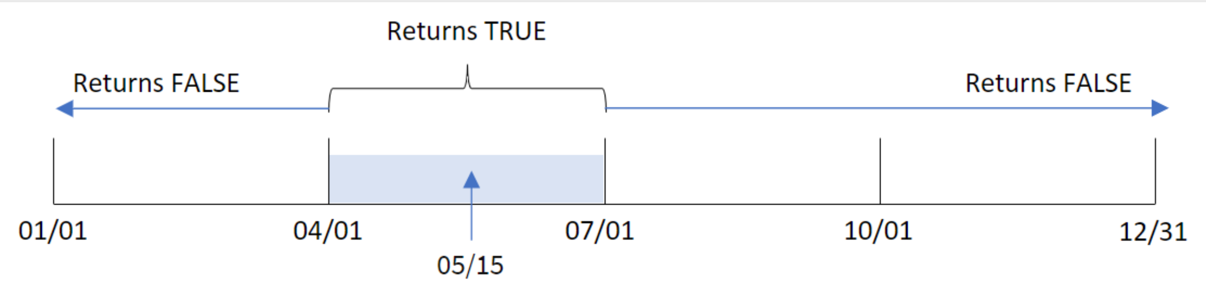
結果表格

日期	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

日期	in_months
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

「in_months」欄位使用 inmonths() 函數在前置 LOAD 陳述式中建立。提供的第一個引數為 3，這將該年分為幾個季區段。第二個引數識別正在評估哪個欄位，在此範例中是日期欄位。第三個引數是 5 月 15 日的硬式編碼日期，這是 base_date，而 0 的 period_no 是最終引數。

具有季區段的 inmonths() 函數圖表



5 月落在該年的第二季。因此，發生在 4 月 1 日和 6 月 30 日之間的任何交易將會傳回 TRUE 的布林值結果 。這在結果表格中驗證。

範例 2 - period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為「Transactions」的表格中。
- 具有其他變數「previous_quarter」的前置載入，這判定交易是否發生在 2022 年 5 月 15 日之前的季度。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- previous_quarter

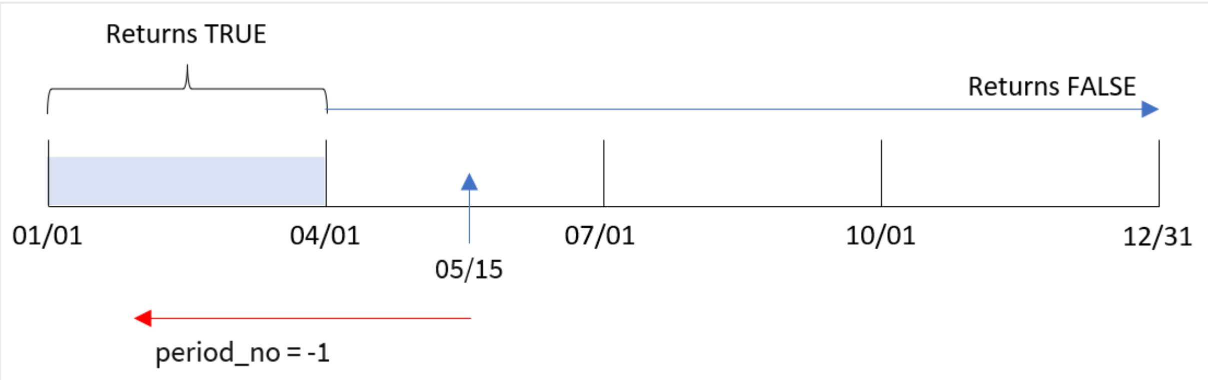
結果表格

日期	上一季
2/19/2022	-1
3/7/2022	-1
3/30/2022	-1
4/5/2022	0
4/16/2022	0
5/1/2022	0
5/7/2022	0
5/22/2022	0
6/15/2022	0
6/26/2022	0

日期	上一季
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

該函數使用 -1 作為 inmonths() 函數中的 period_no 引數，評估交易是否發生在該年的第一季。5 月 15 日是 base_date，落在該年的第二季 (4 月 - 6 月)。

具有季區段且 period_no 設為 -1 的 inmonths() 函數圖表



因此，發生在 1 月和 3 月之間的任何交易將會傳回布林值結果 TRUE。

範例 3 - first_month_of_year

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為「Transactions」的表格中。
- 具有其他變數「in_months」的前置載入，這判定哪些交易發生在與 2022 年 5 月 15 日相同的季度。

在此範例中, 組織政策適用於 3 月作為會計年度第一個月份的情況。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- in_months

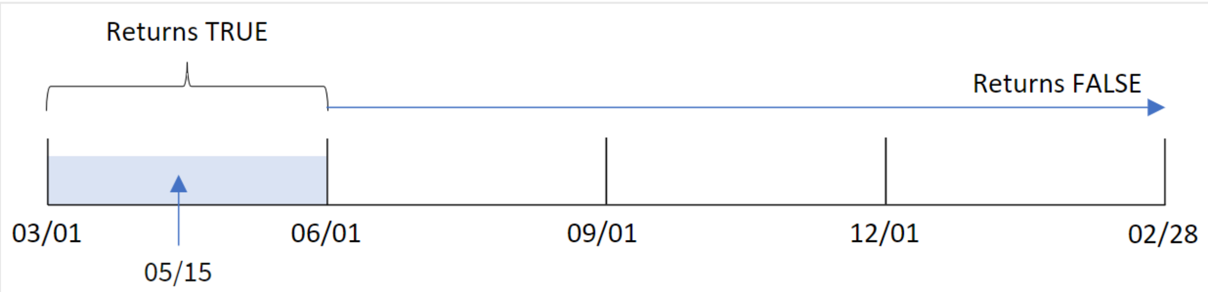
結果表格

日期	in_months
2/19/2022	0
3/7/2022	-1

日期	in_months
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

若使用 3 作為 `inmonths()` 函數中的 `first_month_of_year` 引數，該函數會讓該年從 3 月 1 日開始。然後 `inmonths()` 函數會將該年分為幾個季：3 月-5 月、6 月-8 月、9 月-11 月、12 月-2 月。因此，5 月 15 日落在該年的第一季 (3 月-5 月)。

將 3 月設定為該年第一個月的 `inmonths()` 函數圖表



任何發生在這幾個月的交易會傳回 TRUE 的布林值結果。

範例 4 - 圖表物件範例

載入指令碼和圖表運算式

概覽

使用與第一個範例相同的資料集和情境。

不過，在此範例中，資料集保持不變並且會載入到應用程式中。會建立決定交易是否發生在與 2022 年 5 月 15 日同一季的計算，作為應用程式中圖表的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：

- date

若要計算交易是否發生在與 5 月 15 日同一季，建立下列量值：

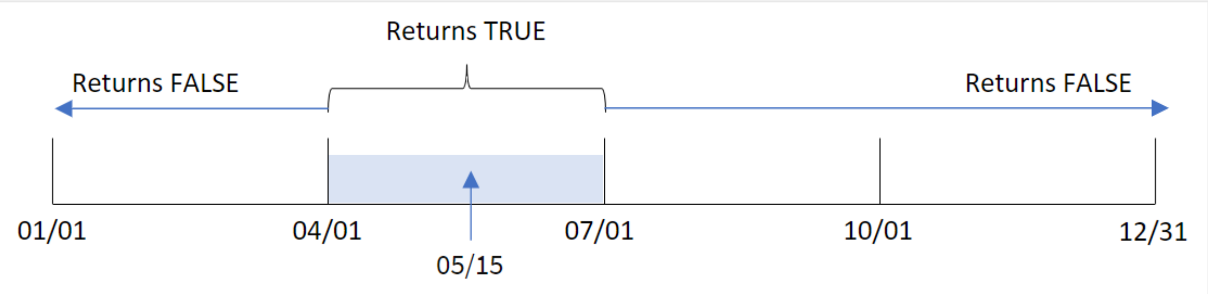
```
=inmonths(3,date,'05/15/2022', 0)
```

結果表格

日期	=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

會使用 inmonths() 函數在圖表中建立「in_months」欄位。提供的第一個引數為 3，這將該年分為幾個季區段。第二個引數識別正在評估哪個欄位，在此範例中是日期欄位。第三個引數是 5 月 15 日的硬式編碼日期，這是 base_date，而 0 的 period_no 是最終引數。

具有季區段的 inmonths() 函數圖表



5 月落在該年的第二季。因此，發生在 4 月 1 日和 6 月 30 日之間的任何交易將會傳回 TRUE 的布林值結果 。這在結果表格中驗證。

範例 5 - 情境

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入到稱為「Products」之表格的資料集。
- 該表格含有下列欄位：
    - 產品 ID
    - 產品類型
    - 製造日期
    - 成本價格

最終使用者希望圖表依產品類型顯示在 2021 年第一個區段製造的產品成本。使用者希望能夠定義此區段的長度。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。

在載入指令碼開頭，已建立與變數輸入控制繫結的變數 vPeriod。

請執行下列動作：

1. 在資產面板中，按一下**自訂物件**。
2. 選取 **Qlik 儀表板搭售**，建立**變數輸入**物件。
3. 輸入圖表物件的標題。
4. 在**變數**之下，選取 **vPeriod** 作為名稱並設定物件以顯示為**下拉式清單**。
5. 在**值**之下，按一下**動態**值。輸入下列內容：
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`
6. 將新的表格新增至工作表。
7. 在屬性面板的**資料**下方，新增 product_type 作為維度。
8. 新增以下運算式作為量值：
   `=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))`
9. 將量值的**數字格式**設定為**金錢**。

結果表格

product_type	=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))
產品 A	$88.27
產品 B	$37.23
產品 C	$17.17
產品 D	$0.00

inmonths() 函數使用使用者輸入作為其引數，以定義該年起始區段的大小。該函數傳遞每個產品的製造日期作為 inmonths() 函數的第二個引數。透過使用 1 月 1 日作為 inmonths() 函數的第三個引數，製造日期落在該年起始區段的產品將會傳回布林值 TRUE，因此 sum 函數將會加上這些產品的成本。

inmonthstodate

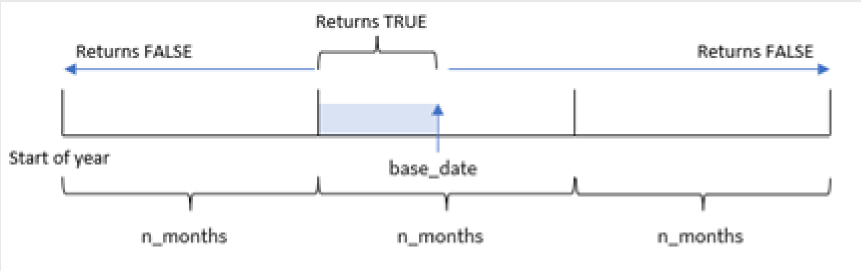
此函數會發現時間戳記是否在月、雙月、季度、四月期或半年的期間內，且不超過 (可包含) base_date 的最後一毫秒。還可以發現時間戳記是否在前一個或後一個時間期間內。

語法：

```
InMonths (n_months, timestamp, base_date, period_no[, first_month_of_year ])
```

傳回的資料類型： 布林

inmonthstodate 函數的圖表。



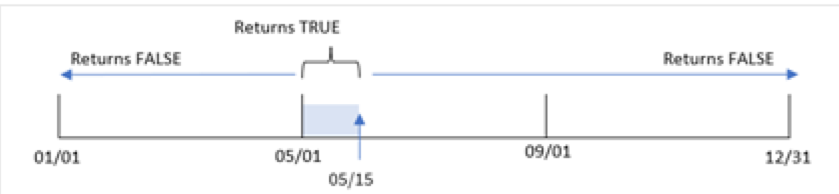
引數

引數	描述
n_months	定義期間的月數。整數或解析為整數的運算式，必須是下列其中一項：1 (相當於 inmonth() 函數)、2 (雙月)、3 (相當於 inquarter() 函數)、4 (四月期) 或 6 (半年)。
timestamp	您要與 **base_date** 比較的日期。
base_date	用來評估期間的日期。
period_no	期間可以使用 **period_no** 位移，是一個整數或解譯為整數的運算式，其中值 0 表示包含 **base_date** 的期間。負值的 **period_no** 表示之前的期間，正值表示之後的期間。
first_month_of_year	如果要使用不起始於 1 月的 (會計) 年度，可在 **first_month_of_year** 中指定介於 2 和 12 之間的值。

在 inmonthstodate() 函數中，base_date 作為其中特定年區段的結束點運作。

例如，若該年分為幾個四個月區段，而 base_date 為 5 月 15 日，則 1 月開始和 4 月結束之間的任何時間戳記都會傳回布林值結果 FALSE。5 月 1 日和 5 月 15 日之間的日期會傳回 TRUE。該年的其餘部分會傳回 FALSE。

inmonthstodate 函數的布林值結果範圍圖表。



一年中的下列區段可作為 n_month 引數用於函數。

n_month 引數

期間	月數
月	1
兩個月	2
季	3
四個月	4
半年	6

什麼情況下使用

inmonthstodate() 函數會傳回布林值結果。通常，此函數類型會作為 if expression 中的條件使用。透過使用 inmonthstodate() 函數，您可以選取想要評估的期間。例如，提供輸入變數，用來讓使用者識別在某個期間截至特定日期的月份、季度或半年製造的產品。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
`inmonthstodate(4, '01/25/2013', '04/25/2013', 0)`	傳回 True，因為 timestamp 的值 01/25/2013 在 01/01/2013 至 04/25/2013 結束的四個月期間內，base_date 的值 04/25/2013 也在該期間內。
`inmonthstodate(4, '04/26/2013', '04/25/2006', 0)`	傳回 False，因為 04/26/2013 處在與前一個範例相同的期間之外。
`inmonthstodate(4, '09/25/2005', '02/01/2006', -1)`	傳回 True，因為 period_no 的值 -1 將搜尋期間往回偏移四個月期間 (n-months 的值)，這讓搜尋期間從 01/09/2005 變更為 02/01/2006。
`inmonthstodate(4, '04/25/2006', '06/01/2006', 0, 3)`	傳回 True，因為 first_month_of_year 的值設為 3，這讓搜尋期間從 03/01/2006 變更為 06/01/2006，而不是 05/01/2006 變更為 06/01/2006。

範例 1 – 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2022 年交易的資料集,這載入到稱為「Transactions」的表格中。
- 使用 DateFormat 系統變數 (MM/DD/YYYY) 格式的日期欄位。
- 前置 LOAD 陳述式包含:
    - 設為欄位「in_months_to_date」的 inmonthstodate() 函數。這決定哪些交易發生在 2022 年該季的 5 月 15 日之前。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- in_months_to_date

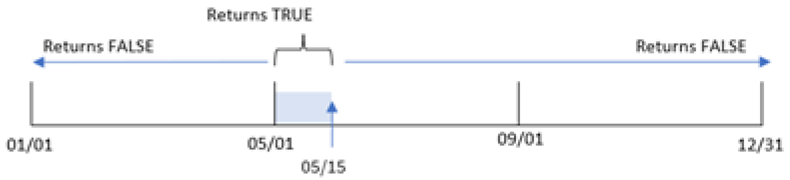
結果表格

日期	in_months_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

「in_months_to_date」欄位使用 inmonthstodate() 函數在前置 LOAD 陳述式中建立。

提供的第一個引數為 3，將該年分為幾個季區段。第二個引數識別正在評估哪個欄位。第三個引數是 5 月 15 日的硬式編碼日期，即定義該區段結束邊界的 base_date。0 的 period_no 是最終引數。

inmonthstodate 函數的圖表，無其他引數。



發生在 4 月 1 日和 5 月 15 日之間的任何交易都會傳回布林值結果 TRUE。該期間之外的交易日期會傳回 FALSE。

範例 2 – period_no

載入指令碼和結果

概覽

使用與第一個範例相同的資料集和情境。

不過，在此範例中，任務是建立欄位「previous_qtr_to_date」，這判定交易是否發生在 5 月 15 日前一季。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- previous_qtr_to_date

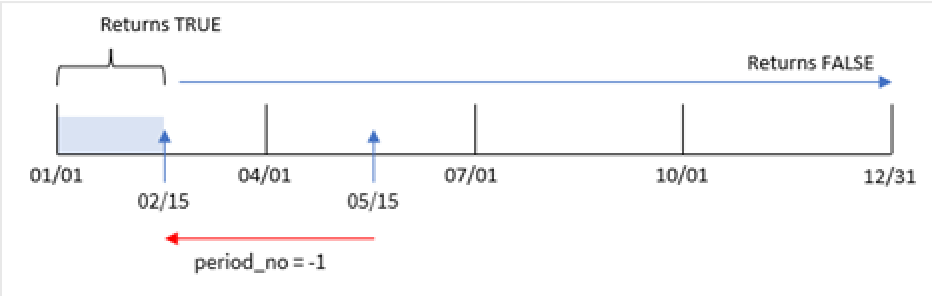
結果表格

日期	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

透過使用 -1 作為 inmonthstodate() 函數中的 period_no 引數,函數會使用一季偏移比較子年區段的界限。

5 月 15 日落在該年的第二季，因此該區段原本等於 4 月 1 日和 5 月 15 日之間。period_no 引數將此區段偏移負三個月。日期邊界變成 1 月 1 日至 2 月 15 日。

period_no 值設為 -1 的 inmonthstodate 函數圖表。



因此，發生在 1 月 1 日和 2 月 15 日之間的任何交易將會傳回布林值結果 TRUE。

範例 3 – first_month_of_year

載入指令碼和結果

概覽

使用與第一個範例相同的資料集和情境。

在此範例中，組織政策適用於 3 月作為會計年度第一個月份的情況。

建立欄位「in_months_to_date」，這決定哪些交易發生在 2022 年同一季的 5 月 15 日之前。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- `date`
- `in_months_to_date`

結果表格

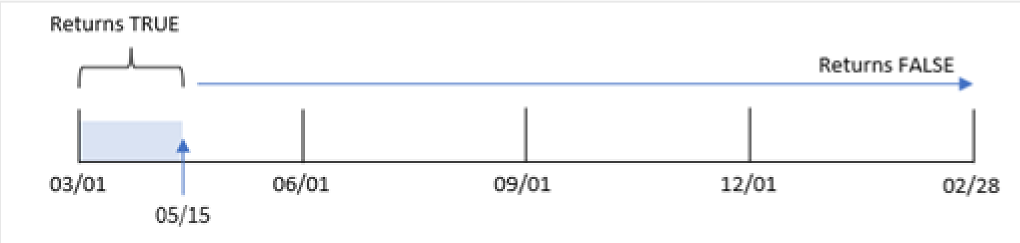
日期	previous_qtr_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	-1
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

若使用 3 作為 inmonthstodate() 函數中的 first_month_of_year引數，該函數會讓該年從 3 月 1 日開始，然後根據提供的第一個引數將該年分為幾個季。因此，季區段為：

- 3 月-5 月
- 6 月-8 月
- 9 月-11 月
- 12 月-2 月

然後值為 5 月 15 日的 base_date 會分割出 3 月 - 5 月的季，並將結束邊界設定為 5 月 15 日。

將 3 月設定為該年第一個月的 inmonthstodate 函數圖表。



因此，發生在 3 月 1 日和 5 月 15 日之間的任何交易將會傳回布林值結果 TRUE，而日期落在這些邊界之外的交易將會傳回 FALSE 的值。

範例 4 – 圖表範例

載入指令碼和圖表運算式

概覽

使用與第一個範例相同的資料集和情境。

在此範例中，資料集保持不變並且會載入到應用程式中。任務是要建立決定交易是否發生在與 5 月 15 日同一季的計算，作為應用程式圖表的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度:

`date`

若要計算交易是否發生在與 5 月 15 日同一季, 建立下列量值:

`=inmonthstodate(3,date,'05/15/2022', 0)`

結果表格

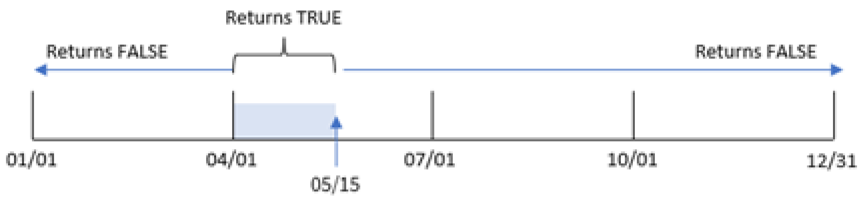
日期	=inmonths(3,date,'05/15/2022', 0)
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0

日期	=inmonths(3,date,'05/15/2022', 0)
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

會使用 `inmonthstodate()` 函數在圖表中建立 ‘`in_months_to_date` 量值。

提供的第一個引數為 3，將該年分為幾個季區段。第二個引數識別正在評估哪個欄位。第三個引數是 5 月 15 日的硬式編碼日期，即定義該區段結束邊界的 `base_date`。0 的 `period_no` 是最終引數。

具有季區段的 `inmonthstodate` 函數圖表。



發生在 4 月 1 日和 5 月 15 日之間的任何交易將會傳回布林值結果 TRUE。該區段之外的交易日期將會傳回 FALSE。

範例 5 – 情境

載入指令碼和結果

概覽

在此範例中，資料集會載入到稱為「`sales`」的表格中。該表格含有下列欄位：

- 產品 ID
- 產品類型
- 銷售日期
- 銷售價格

最終使用者希望圖表依產品類型顯示截至 2022 年 12 月 24 日期間的產品銷售額。使用者希望能夠定義此期間的長度。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。

在載入指令碼開頭，已建立與變數輸入控制繫結的變數 `vPeriod`。

請執行下列動作：

1. 在資產面板中，按一下**自訂物件**。
2. 選取 **Qlik 儀表板搭售**並將**變數輸入**新增至工作表。
3. 輸入圖表的標題。
4. 在**變數**之下，選取 **vPeriod** 作為名稱並設定物件以顯示為**下拉式清單**。
5. 在**值**之下，按一下**動態**值。輸入下列內容：
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`
6. 將新的表格新增至工作表。
7. 在屬性面板的**資料**下方，新增 `product_type` 作為維度。
8. 新增以下運算式作為量值：
   `=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24),0),sales_price,0))`
9. 將量值的**數字格式**設定為**金錢**。

結果表格

product_type	=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24)),0),sales_price,0))
產品 A	$186.48
產品 B	$190.52
產品 C	$220.43
產品 D	$261.46

inmonthstodate() 函數使用使用者輸入作為其引數，以定義該年起始區段的大小。

該函數傳遞每個產品的銷售日期作為 inmonthstodate() 函數的第二個引數。透過使用 12 月 24 日作為 inmonthstodate() 函數中的第三個引數，銷售日期發生在截至 (含) 12 月 24 日之定義期間的產品會傳回布林值 TRUE。sum 函數會加上這些產品的銷售額。

inmonthtodate

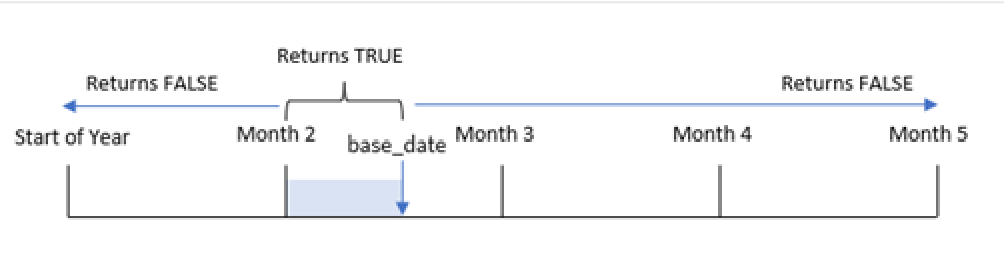
如果 **date** 位於包含 **basedate** 的月份部分內，且不超過 (可包含) **basedate** 的最後一毫秒，則傳回 True。

語法：

```
InMonthToDate (timestamp, base_date, period_no)
```

傳回的資料類型：布林

inmonthtodate 函數的圖表。



inmonthtodate() 函數識別所選月份作為區段。開始邊界是該月的開始。結束邊界可設定為該月的較晚日期。然後這會判定一組日期是否落在此區段，傳回 TRUE 或 FALSE 布林值。

引數

引數	描述
timestamp	您要與 **base_date** 比較的日期。
base_date	用來評估月份的日期。
period_no	月份可以使用**period_no** 位移。**period_no** 是一個整數，其中值 0 表示包含 **base_date** 的月份。負值的 **period_no** 表示之前的月份，正值表示之後的月份。

什麼情況下使用

inmonthtodate() 函數會傳回布林值結果。通常,此函數類型會作為 if expression 中的條件使用。inmonthtodate() 函數根據日期是否發生於截至 (含) 問題中日期的該月份中,傳回彙總或計算。

例如,inmonthtodate() 函數可用來識別某個月截至特定日期製造的所有設備。

區域設定

除非另有說明,否則此主題中的範例皆使用下列日期格式:MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素,您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式,以滿足您的需求。或者,您可以在載入指令碼中變更格式,以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典,資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
inmonthtodate ('01/25/2013', '25/01/2013', 0)	傳回 True
inmonthtodate ('01/25/2013', '24/01/2013', 0)	傳回 False
inmonthtodate ('01/25/2013', '28/02/2013', -1)	傳回 True

範例 1 – 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2022 年交易的資料集,這載入到稱為「Transactions」的表格中。
- 以 DateFormat 系統變數 (MM/DD/YYYY) 格式提供日期欄位。
- 前置 LOAD 陳述式包含:
  - 設定為欄位「in_month_to_date」的 inmonthtodate() 函數。這判定哪些交易發生在 2022 年 7 月 1 日和 7 月 26 日之間。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- in_month_to_date

結果表格

日期	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

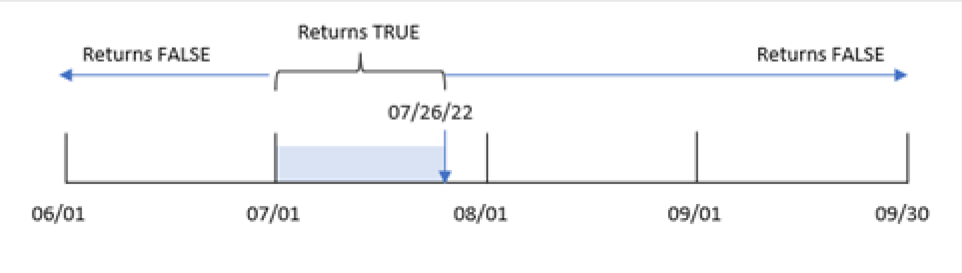
日期	in_month_to_date
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	-1
7/22/2022	-1
7/23/2022	-1
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

「in_month_to_date」欄位使用 inmonthtodate() 函數在前置 LOAD 陳述式中建立。

第一個引數識別正在評估哪個欄位。第二個引數是硬式編碼日期 7 月 26 日，這是 base_date。此 base_date 引數識別哪個月受到分段以及該區段的結束邊界。

0 的 period_no 是最終引數，表示該函數沒有比較分割月份之前或之後的月份。

inmonthtodate 函數的圖表，無其他引數。



因此，發生在 7 月 1 日和 7 月 26 日之間的任何交易都會傳回布林值結果 TRUE。發生在 7 月且晚於 7 月 26 日的任何交易會傳回布林值結果 FALSE，如同該年任何其他月份的任何交易。

範例 2 – period_no

載入指令碼和結果

概覽

使用與第一個範例相同的資料集和情境。

在此範例中,任務是建立欄位「six_months_prior」,這判定哪些交易發生在 7 月 1 日和 7 月 26 日之前的完整六個月。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- six_months_prior

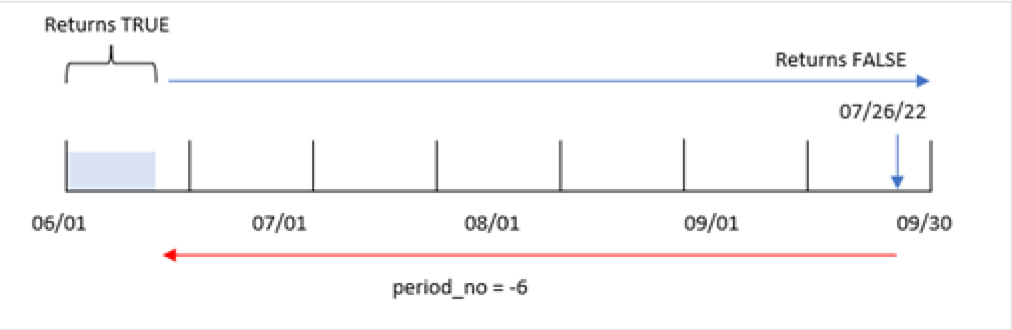
結果表格

日期	six_months_prior
1/7/2022	-1
1/19/2022	-1

日期	six_months_prior
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

透過使用 -6 作為 inmonthtodate() 函數中的 period_no 引數，比較子月份區段的邊界會偏移六個月。原本該月區段等於 7 月 1 日和 7 月 26 日之間。然後 period_no 將此區段偏移負六個月，日期邊界會偏移，並落在 1 月 1 日和 1 月 26 日之間。

period_no 值設為 -6 的 inmonthtodate 函數圖表。



因此，發生在 1 月 1 日和 1 月 26 日之間的任何交易將會傳回布林值結果 TRUE。

範例 3 – 圖表範例

載入指令碼和圖表運算式

概述

使用與第一個範例相同的資料集和情境。

在此範例中，資料集保持不變並且會載入到應用程式中。任務是要建立決定交易是否發生在 7 月 1 日和 7 月 26 日之間的計算，作為應用程式圖表中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：

date

若要計算交易是否發生在 7 月 1 日和 7 月 26 日之間，建立下列量值：

```
=inmonthtodate(date,'07/26/2022', 0)
```

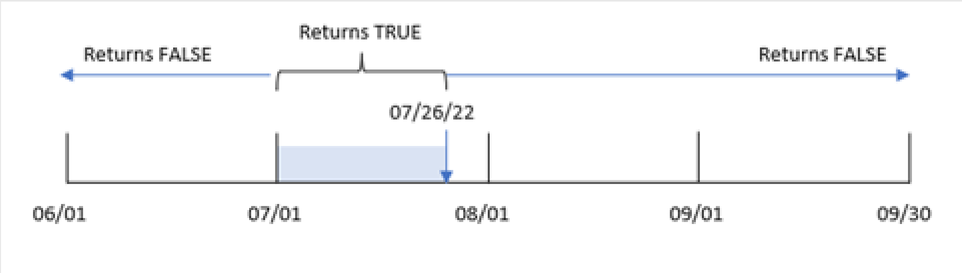
結果表格

日期	=inmonthtodate(date,'07/26/2022', 0)
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	-1
7/22/2022	-1
7/23/2022	-1
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

會使用 `inmonthtodate()` 函數在圖表中建立「`in_month_to_date`」欄位量值。

第一個引數識別正在評估哪個欄位。第二個引數是硬式編碼日期 7 月 26 日，這是 `base_date`。此 base_date 引數識別哪個月受到分段以及該區段的結束邊界。值為 0 的 `period_no` 是最終引數。這表示該函數沒有比較分割月份之前或之後的月份。

inmonthtodate 函數的圖表，無其他引數。



因此，發生在 7 月 1 日和 7 月 26 日之間的任何交易都會傳回布林值結果 TRUE。發生在 7 月且晚於 7 月 26 日的任何交易會傳回布林值結果 FALSE，如同該年任何其他月份的任何交易。

範例 4 – 情境

載入指令碼和結果

概覽

在此範例中，資料集會載入到稱為「Products」的表格中。該表格含有下列欄位：

- 產品 ID
- 製造日期
- 成本價格

由於設備錯誤，在 2022 年 7 月製造的產品有瑕疵。問題已於 2022 年 7 月 27 日解決。

最終使用者希望圖表依月份顯示製造時為「瑕疵」(布林值 TURE) 或「故障」(布林值 FALSE) 的產品狀態，以及在該月份製造的產品成本。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- =monthname(manufacture_date)
- =if(Inmonthtodate(manufacture_date,makedate(2022,07,26),0),'Defective','Faultless')

若要計算產品成本總和，建立此量值：

=sum(cost_price)

將量值的**數字格式**設定為**金錢**。

結果表格

monthname (manufacture_date)	**if(Inmonthtodate(manufacture_date,makedate (2022,07,26),0),'Defective','Faultless')**	**Sum(cost_ price)**
2022 年 1 月	無缺失	$54.40
2022 年 2 月	無缺失	$145.69
2022 年 3 月	無缺失	$53.80
2022 年 4 月	無缺失	$82.06
2022 年 5 月	無缺失	$127.60
2022 年 6 月	無缺失	$141.82
2022 年 7 月	瑕疵	$144.66
2022 年 7 月	無缺失	$69.98
2022 年 8 月	無缺失	$147.46
2022 年 9 月	無缺失	$84.21
2022 年 10 月	無缺失	$163.91

inmonthtodate() 函數會在評估每個產品的製造日期時傳回布林值。

對於傳回布林值 TRUE 的日期，產品會標記為「瑕疵」。對於傳回 FALSE 值的任何產品 (因此這不是在該月截至 (含) 7 月 26 日前製造)，會將產品標記為「無缺失」。

inquarter

此函數會傳回 True，前提是如果 **timestamp** 位於包含 **base_date** 的季度中。

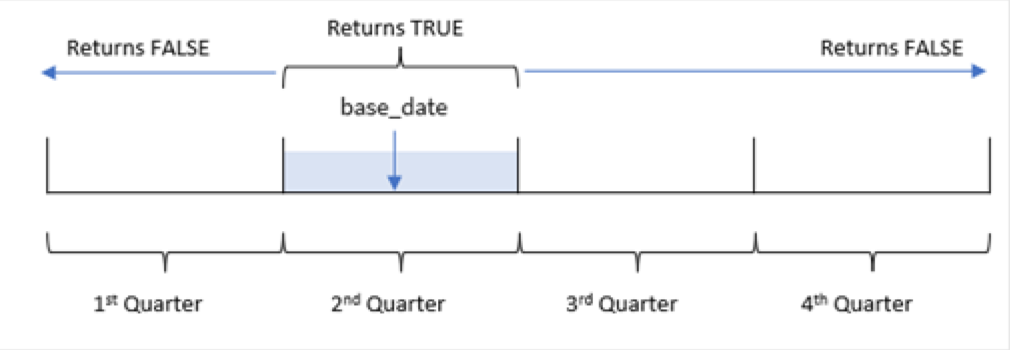
語法：

```
InQuarter (timestamp, base_date, period_no[, first_month_of_year])
```

傳回的資料類型： 布林

在 Qlik Sense 中，布林值 true 值以 -1 代表，而 false 值以 0 代表。

inquarter() 函數範圍的圖表



換言之，inquarter() 函數將該年在 1 月 1 日和 12 月 31 日之間分為四個相等的季。您可以使用 first_month_of_year 引數變更哪個月視為應用程式中的第一個月，季度將會根據該引數變更。base_date 函數識別哪一季應作為該函數的比較子使用。最後，該函數會在比較日期值與該季區段時，傳回布林值結果。

什麼情況下使用

inquarter() 函數會傳回布林值結果。通常，此函數類型將作為 if expression 中的條件使用。這根據日期是否發生於所選的季中傳回彙總或計算。

例如，inquarter() 函數可用來根據設備製造日期識別在季區段中製造的所有設備。

引數

引數	描述
timestamp	您要與 **base_date** 比較的日期。
base_date	用來評估季度的日期。
period_no	季度可以使用 **period_no** 位移。**period_no** 是一個整數，其中值 0 表示包含 **base_date** 的季度。負值的 **period_no** 表示之前的季度，正值表示之後的季度。
first_month_of_year	如果要使用不起始於 1 月的 (會計) 年度，可在 **first_month_of_year** 中指定介於 2 和 12 之間的值。

您可以在 first_month_of_year 引數中使用下列值設定一年的第一個月：

first_month_of_year 值

月	值
二月	2
三月	3
四月	4
五月	5
六月	6
七月	7
八月	8
九月	9
十月	10
十一月	11
十二月	12

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
inquarter ('01/25/2013', '01/01/2013', 0)	傳回 TRUE
inquarter ('01/25/2013', '04/01/2013', 0)	傳回 FALSE
inquarter ('01/25/2013', '01/01/2013', -1)	傳回 FALSE
inquarter ('12/25/2012', '01/01/2013', -1)	傳回 TRUE
nquarter ('01/25/2013', '03/01/2013', 0, 3)	傳回 FALSE
inquarter ('03/25/2013', '03/01/2013', 0, 3)	傳回 TRUE

範例 1 - 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2022 年交易的資料集,這載入到稱為「Transactions」的表格中。
- 包含設為「in_quarter」欄位之 inquarter() 函數的前置載入,這判定哪些交易發生在與 2022 年 5 月 15 日相同的季度。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

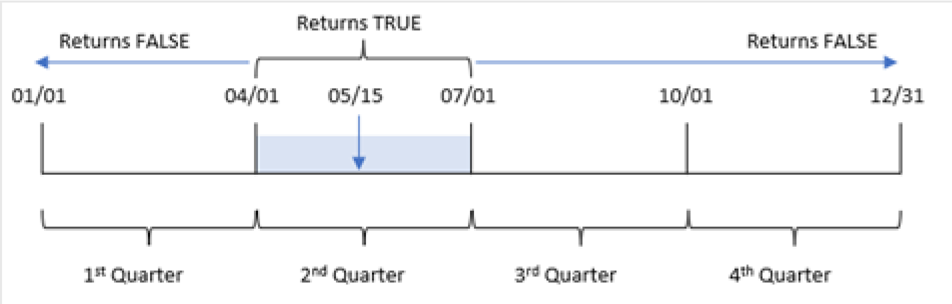
- date
- in_quarter

結果表格

日期	in_quarter
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	-1
6/15/2022	-1
6/26/2022	-1
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

「in_quarter」欄位使用 inquarter() 函數在前置 LOAD 陳述式中建立。第一個引數識別正在評估哪個欄位。第二個引數是 5 月 15 日的硬式編碼日期，這識別哪一季要定義為比較子。0 的 period_no 是最終引數，並確保 inquarter() 函數沒有比較分割的季之前或之後的季。

以 5 月 15 日作為基底日期的 inquarter() 函數圖表



發生在 4 月 1 日和 6 月 30 日結束之間的任何交易都會傳回布林值結果 TRUE。

範例 2 - period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2022 年交易的資料集,這載入到稱為「Transactions」的表格中。
- 包含設為「previous_quarter」欄位之 inquarter() 函數的前置載入,這判定哪些交易發生在 2022 年 5 月 15 日的前一季。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

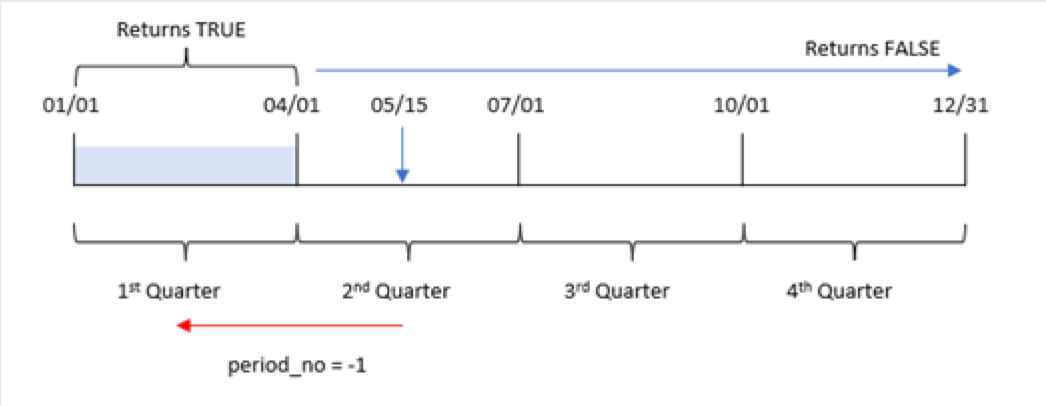
- date
- previous_qtr

結果表格

日期	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

使用 -1 作為 inquarter() 函數中的 period_no 引數，可將比較子季度的邊界往回偏移完整一季。5 月 15 日落在該年的第二季，因此該區段原本等於 4 月 1 日至 6 月 30 日該季。period_no 將此區段偏移負三個月，造成日期邊界變成 1 月 1 日到 3 月 30 日。

以 5 月 15 日作為基底日期的 inquarter() 函數圖表



因此，發生在 1 月 1 日和 3 月 30 日之間的任何交易將會傳回布林值結果 TRUE。

範例 3 - first_month_of_year

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為「Transactions」的表格中。
- 包含設為「in_quarter」欄位之 inquarter() 函數的前置載入，這判定哪些交易發生在與 2022 年 5 月 15 日相同的季度。

不過，在此範例中，組織政策適用於 3 月作為會計年度第一個月份的情況。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- previous_qtr

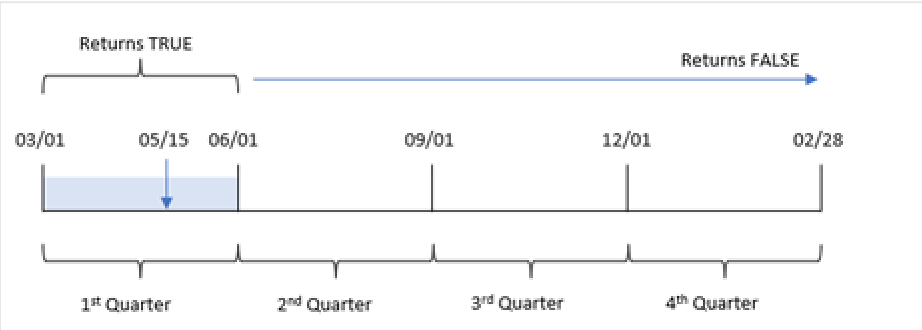
結果表格

日期	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

日期	previous_qtr
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

使用 3 作為 `inquarter()` 函數中的 `first_month_of_year` 引數，會將 3 月 1 日設為該年的開始，然後將該年分為幾個季。因此，該季區段為 3 月-5 月、6 月-8 月、9 月-11 月、12 月-2 月。值為 5 月 15 日的 `base_date` 將 3 月-5 月該季設定為該函數的比較子季度。

將 3 月設定為該年第一個月的 `inquarter()` 函數圖表



因此，發生在 3 月 1 日和 5 月 31 日之間的任何交易將會傳回布林值結果 TRUE。

範例 4 - 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為「`Transactions`」的表格中。
- 包含設為「`in_quarter`」欄位之 `inquarter()` 函數的前置載入，這判定哪些交易發生在與 2022 年 5 月 15 日相同的季度。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度:

- date

建立下列量值,以計算交易是否發生在與 5 月 15 日同一季:

```
=inquarter(date,'05/15/2022', 0)
```

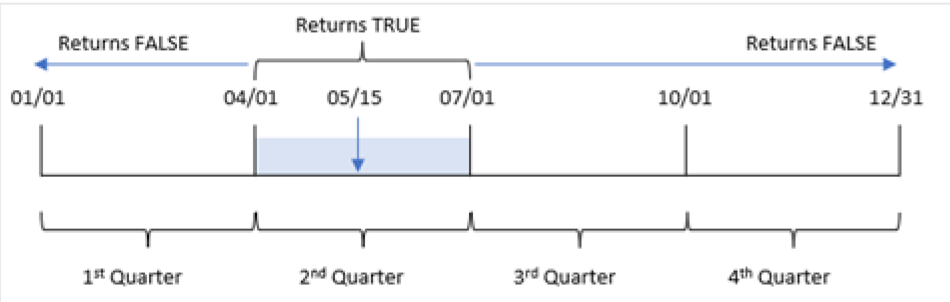
結果表格

日期	in_quarter
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	-1
6/15/2022	-1

日期	in_quarter
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

會使用 `inquarter()` 函數在圖表中建立「`in_quarter`」量值。第一個引數識別正在評估哪個欄位。第二個引數是 5 月 15 日的硬式編碼日期，這識別哪一季要定義為比較子。0 的 `period_no` 是最終引數，並確保 `inquarter()` 函數沒有比較分割的季之前或之後的季。

以 5 月 15 日作為基底日期的 `inquarter()` 函數圖表



發生在 4 月 1 日和 6 月 30 日結束之間的任何交易都會傳回布林值結果 TRUE。

範例 5 - 情境

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入到稱為「`Products`」之表格的資料集。
- 該表格含有下列欄位：

- 產品 ID
- 產品類型
- 製造日期
- 成本價格

這識別出由於設備錯誤，在 2022 年 5 月 15 日該季製造的產品有瑕疵。最終使用者希望圖表依季度名稱顯示製造的哪些產品狀態為「瑕疵」或「無缺失」，以及在該季製造的產品成本。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：

`=quartername(manufacture_date)`

建立下列量值：

- `=if(only(InQuarter(manufacture_date,makedate(2022,05,15),0)),'Defective','Faultless')`，用來使用 `inquarter()` 函數識別哪些產品為瑕疵，那些為無缺失。
- `=sum(cost_price)`，用來顯示每個產品成本的總和。

請執行下列動作：

1. 將量值的**數字格式**設定為**金錢**。
2. 在**外觀**之下，關閉**總計**。

結果表格

quartername (manufacture_date)	=if(only(InQuarter(manufacture_date,makedate (2022,05,15),0)),'Defective','Faultless')	Sum(cost_ price)
2022 年 1 月至 3 月	無缺失	253.89
2022 年 4 月至 6 月	瑕疵	351.48
2022 年 7 月至 9 月	無缺失	446.31
2022 年 10 月至 12 月	無缺失	163.91

inquarter() 函數會在評估每個產品的製造日期時傳回布林值。對於在包含 5 月 15 日該季製造的任何產品，inquarter() 函數會傳回布林值 TRUE 並將產品標記為「瑕疵」。對於傳回 FALSE 值的任何產品 (因此這不是在該季製造)，會將產品標記為「無缺失」。

inquartertodate

如果 **timestamp** 位於包含 **base_date** 的季度部分內，且不超過 (可包含) **base_date** 的最後一毫秒，則此函數會傳回 True。

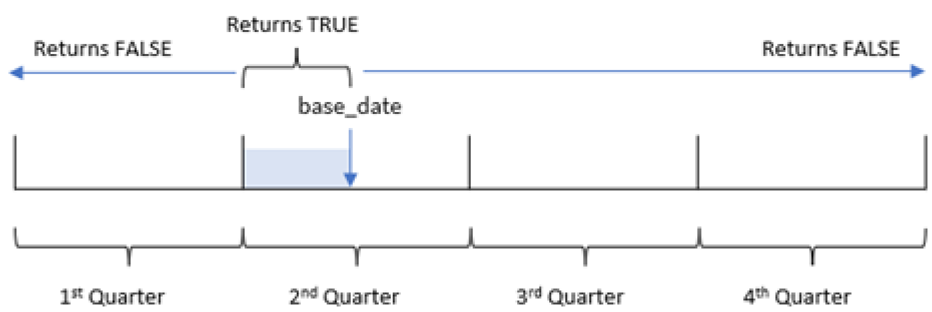
語法：

```
InQuarterToDate (timestamp, base_date, period_no [, first_month_of_year])
```

傳回的資料類型：布林

在 Qlik Sense 中，布林值 true 值以 -1 代表，而 false 值以 0 代表。

inquartertodate 函數的圖表



inquartertodate() 函數會在 1 月 1 日和 12 月 31 日 (或使用者定義的一年開始日期及其對應的結束日期) 之間將該年分為四個相等的季。使用 base_date，該函數則會分割特定的季，並透過 base_date 識別哪一季和該季區段的最大允許日期。最後，該函數會在比較規定的日期值與該區段時，傳回布林值結果。

引數

引數	描述
timestamp	您要與 **base_date** 比較的日期。
base_date	用來評估季度的日期。
period_no	季度可以使用 **period_no** 位移。**period_no** 是一個整數，其中值 0 表示包含 **base_date** 的季度。負值的 **period_no** 表示之前的季度，正值表示之後的季度。
first_month_of_year	如果要使用不起始於 1 月的 (會計) 年度，可在 **first_month_of_year** 中指定介於 2 和 12 之間的值。

什麼情況下使用

inquartertodate() 函數會傳回布林值結果。通常，此函數類型將作為 if 運算式中的條件使用。inquartertodate() 函數會用來根據評估的日期是否發生於該季截至 (含) 問題中的日期，傳回彙總或計算。

例如，inquartertodate() 函數可用來識別某季截至特定日期製造的所有設備。

函數範例

範例	結果
inquartertodate ('01/25/2013', '03/25/2013', 0)	傳回 TRUE，因為 timestamp 的值 01/25/2013 落在從 01/01/2013 到 03/25/2013 的三個月期間內，其中包含 base_date 的值 03/25/2013。
inquartertodate ('04/26/2013', '03/25/2013', 0)	傳回 FALSE，因為 04/26/2013 處在與前一個範例相同的期間之外。
inquartertodate ('02/25/2013', '06/09/2013', -1)	傳回 TRUE，因為 period_no 的值 -1 將搜尋期間往回移動一段三個月期間 (該年的一季)。這將搜尋期間從 01/01/2013 成為 03/09/2013。
inquartertodate ('03/25/2006', '04/15/2006', 0, 2)	傳回 TRUE，因為 first_month_of_year 的值設為 2，這將搜尋期間從 02/01/2006 成為 04/15/2006，而非從 04/01/2006 成為 04/15/2006。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2022 年交易的資料集，這載入到稱為 `Transactions` 的表格中。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 `in_quarter_to_date`，這決定哪些交易發生在 2022 年該季的 5 月 15 日之前。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

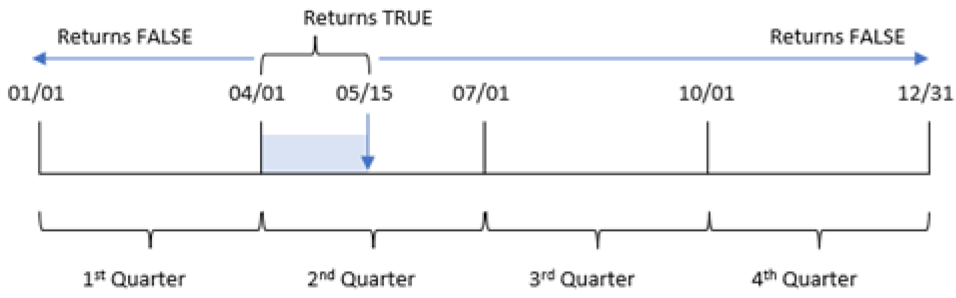
- date
- in_quarter_to_date

結果表格

日期	in_quarter_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

「in_quarter_to_date」欄位使用 inquartertodate() 函數在前置 LOAD 陳述式中建立。提供的第一個引數識別正在評估哪個欄位。第二個引數是 5 月 15 日的硬式編碼日期，即識別要分割哪一季並定義該區段結束邊界的 base_date。值為 0 的 period_no 是最終引數，表示該函數沒有比較分割的季之前或之後的季。

inquartertodate 函數的圖表，無其他引數



發生在 4 月 1 日和 5 月 15 日之間的任何交易都會傳回布林值結果 TRUE。5 月 16 日及之後的交易日期將會傳回 FALSE，如同 4 月 1 日之前的任何交易。

範例 2 – period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 建立欄位 previous_qtr_to_date，這決定哪些交易發生在結束於 2022 年 5 月 15 日之季區段的完整一季之前。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- previous_qtr_to_date

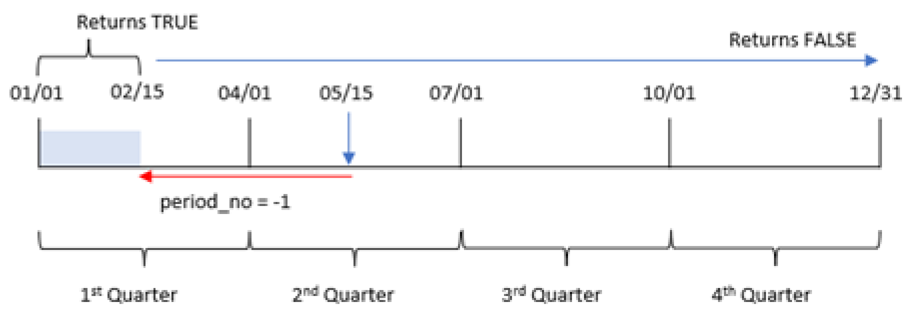
結果表格

日期	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

日期	previous_qtr_to_date
9/26/2022	0
10/14/2022	0
10/29/2022	0

period_no 值 -1 表示 inquartertodate () 函數比較輸入的季區段與上一季。5 月 15 日落在該年的第二季，因此該區段原本等於 4 月 1 日和 5 月 15 日之間。然後 period_no 將此區段往前移動三個月，造成日期邊界變成 1 月 1 日到 2 月 15 日。

inquartertodate 函數的圖表，period_no 範例



因此，發生在 1 月 1 日和 2 月 15 日之間的任何交易將會傳回布林值結果 TRUE。

範例 3 – first_month_of_year

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 建立欄位 in_quarter_to_date，這決定哪些交易發生在 2022 年同一季的 5 月 15 日之前。

在此範例中，我們將 3 月設為會計年度的第一個月。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- in_quarter_to_date

結果表格

日期	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

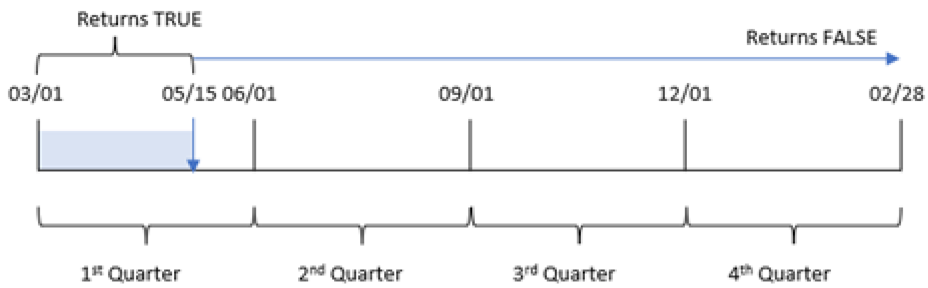
日期	in_quarter_to_date
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

若使用 3 作為 `inquartertodate()` 函數中的 `first_month_of_year` 引數，該函數會讓該年從 3 月 1 日開始，然後將該年分為幾個季。因此，季區段為：

- 3 月至 5 月
- 6 月至 8 月
- 9 月至 11 月
- 12 月至 2 月

然後值為 5 月 15 日的 `base_date` 會分割出 3 月至 5 月的季，並將結束邊界設定為 5 月 15 日。

inquartertodate 函數的圖表，first_month_of_year 範例



因此，發生在 3 月 1 日和 3 月 15 日之間的任何交易將會傳回布林值結果 TRUE，而日期落在這些邊界之外的交易將會傳回 FALSE 的值。

範例 4 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。不過，在此範例中，不變的資料集會載入到應用程式中。會建立決定哪些交易發生在與 5 月 15 日同一季的計算，作為圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`date`。

建立下列量值：

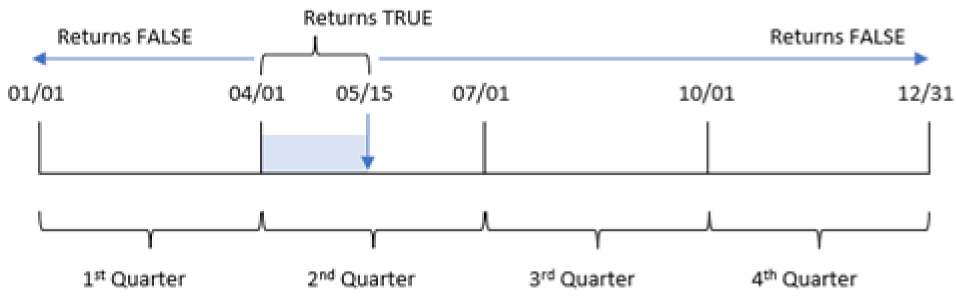
```
=inquartertodate(date,'05/15/2022', 0)
```

結果表格

日期	=inquartertodate(date,'05/15/2022', 0)
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

會使用 `inquartertodate()` 函數在圖表物件中建立 `in_quarter_to_date` 量值。第一個引數是正在評估的日期欄位。第二個引數是 5 月 15 日的硬式編碼日期，即識別要分割哪一季並定義該區段結束邊界的 `base_date`。值為 0 的 `period_no` 是最終引數，表示該函數沒有比較分割的季之前或之後的季。

inquartertodate 函數的圖表, 圖表物件範例



發生在 4 月 1 日和 5 月 15 日之間的任何交易都會傳回布林值結果 TRUE。5 月 16 日及之後的交易將會傳回 FALSE, 如同 4 月 1 日之前的任何交易。

範例 5 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 載入到稱為 Products 之表格的資料集。
- 關於產品 ID、製造日期和成本價格的資訊。

在 2022 年 5 月 15 日, 識別並解決了製造流程中的一項設備錯誤。在該季截至此日期製造的產品將會有瑕疵。最終使用者希望圖表物件依季名稱顯示產品的狀態為「瑕疵」或「故障」, 以及在該季至今製造的產品成本。

載入指令碼

```
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
];
```

結果

請執行下列動作:

1. 載入資料並開啟工作表。建立新的表格。建立維度以顯示季名稱:
   =quartername(manufacture_date)
2. 接下來,建立維度以識別哪些產品為瑕疵,那些為故障:
   =if(inquartertodate(manufacture_date,makedate(2022,05,15),0),'Defective','Faultless')
3. 建立量值以加總產品的 cost_price:
   =sum(cost_price)
4. 將量值的**數字格式**設定為**金錢**。

結果表格

quartername (manufacture_date)	if(inquartertodate(manufacture_date,makedate (2022,05,15),0),'Defective','Faultless')	Sum(cost_ price)
2022 年 1 月至 3 月	無缺失	$253.89
2022 年 4 月至 6 月	無缺失	$229.03
2022 年 4 月至 6 月	瑕疵	$122.45
2022 年 7 月至 9 月	無缺失	$446.31
2022 年 10 月至 12 月	無缺失	$163.91

inquartertodate() 函數會在評估每個產品的製造日期時傳回布林值。對於傳回布林值 TRUE 的項目,會將產品標記為 'Defective'。對於傳回值 FALSE 的任何產品,以及因此不是在該季截至 (含) 5 月 15 日製作的產品,會將產品標記為 'Faultless'。

inweek

此函數會傳回 True,前提是如果 **timestamp** 位於包含 **base_date** 的週中。

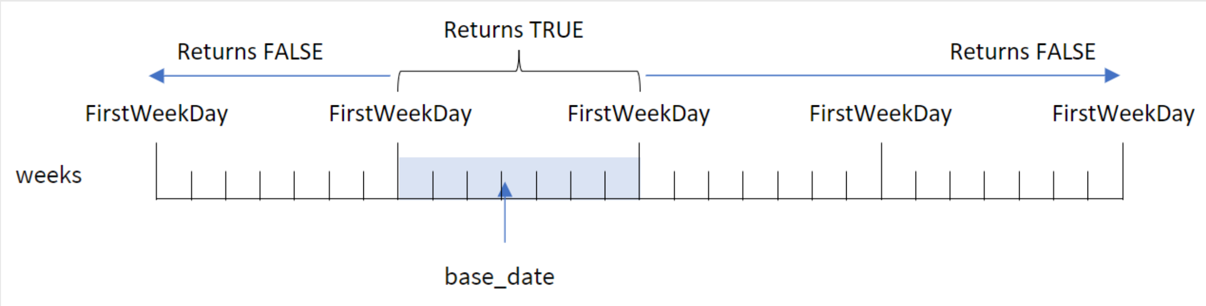
語法:

InWeek (timestamp, base_date, period_no[, first_week_day])

傳回的資料類型: 布林

在 Qlik Sense 中,布林值 true 值以 -1 代表,而 false 值以 0 代表。

inweek() 函數範圍的圖表



inweek() 函數使用 base_date 引數來識別日期落在哪一段七天期間。該週的開始日是根據 FirstWeekDay 系統變數。不過，您可以使用 inweek() 函數的 first_week_day 引數變更該週的第一天。

定義所選的週後，該函數會在比較規定的日期值與該週區段時，傳回布林值結果。

什麼情況下使用

inweek() 函數會傳回布林值結果。通常，此函數類型將作為 if expression 中的條件使用。inweek() 函數根據評估的日期是否發生在具有 base_date 引數所選日期的該週，傳回彙總或計算。

例如，inweek() 函數可用來識別在特定週製造的所有設備。

引數

引數	描述
timestamp	您要與 **base_date** 比較的日期。
base_date	用來評估週的日期。
period_no	週可以使用 **period_no** 位移。**period_no** 是一個整數，其中值 0 表示包含 **base_date** 的週。負值的 **period_no** 表示之前的週，正值表示之後的週。
first_week_day	依照預設，該週的第一天為星期日 (如 FirstWeekDay 系統變數所決定)，從星期六和星期日之間的午夜開始。**first_week_day** 參數取代 **FirstWeekDay** 變數。若要指出從另一日開始的週，請在 0 和 6 之間指定一個旗標。

first_week_day 值

日	值
星期一	0
星期二	1
星期三	2
星期四	3
星期五	4
星期六	5
星期日	6

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
`inweek ('01/12/2006', '01/14/2006', 0)`	傳回 TRUE
`inweek ('01/12/2006', '01/20/2006', 0 )`	傳回 FALSE
`inweek ('01/12/2006', '01/14/2006', -1 )`	傳回 FALSE
`inweek ('01/07/2006', '01/14/2006', -1)`	傳回 TRUE
`inweek ('01/12/2006', '01/09/2006', 0, 3)`	傳回 FALSE，因為 `first_week_day` 指定為 3 (星期四)，這使得 01/12/2006 成為包含 01/09/2006 的週之後星期的第一天。

這些主題可協助您使用此函數：

相關主題

主題	預設旗標 / 值	描述
FirstWeekDay (page 212)	6 / 星期日	定義每週開始的日子。

範例 1 - 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年 1 月交易的資料集，這載入到稱為「`Transactions`」的表格中。
- 設為 6 (星期日) 的 `FirstWeekDay` 系統變數。

- 前置載入包含下列內容：
    - 設為欄位「in_week」的 inweek() 函數，決定哪些交易發生在 2022 年 1 月 14 日該週。
    - 設為欄位「week_day」的 weekday() 函數，顯示該週的哪一天對應至每個日期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- week_day
- in_week

結果表格

日期	week_day	in_week
01/02/2022	星期日	0
01/05/2022	星期三	0
01/06/2022	星期四	0
01/08/2022	星期六	0
01/09/2022	星期日	-1
01/10/2022	星期一	-1
01/11/2022	星期二	-1
01/12/2022	星期三	-1
01/13/2022	星期四	-1
01/14/2022	星期五	-1
01/15/2022	星期六	-1
01/16/2022	星期日	0
01/17/2022	星期一	0
01/18/2022	星期二	0
01/26/2022	星期三	0
01/27/2022	星期四	0
01/28/2022	星期五	0
01/29/2022	星期六	0
01/30/2022	星期日	0
01/31/2022	星期一	0

「in_week」欄位使用 inweek() 函數在前置 LOAD 陳述式中建立。第一個引數識別正在評估哪個欄位。第二個引數是 1 月 14 日的硬式編碼日期，這是 base_date。base_date 引數搭配 FirstWeekDay 系統變數運作，以識別比較子週。0 的 period_no 是最終引數，表示該函數沒有比較分割的週之前或之後的週。

FirstWeekDay 系統變數決定週於星期日開始，於星期六結束。因此，1 月會根據下圖分為數週，其中 1 月 9 日和 15 日之間的日期提供適用於 inweek() 計算的有效期間：

醒目提示 inweek() 函數範圍的行事曆圖表

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					

發生在 1 月 9 日和 1 月 15 日之間的任何交易都會傳回布林值結果 `TRUE`。

範例 2 - period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的相同資料集，這載入到稱為「`Transactions`」的表格中。
- 設為 6 (星期日) 的 `FirstWeekDay` 系統變數。
- 前置載入包含下列內容：
    - 設為欄位「`prev_week`」的 `inweek ()` 函數，決定哪些交易發生在 2022 年 1 月 14 日該週的完整一週之前。
    - 設為欄位「`week_day`」的 `weekday()` 函數，顯示該週的哪一天對應至每個日期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- week_day
- prev_week

結果表格

日期	week_day	prev_week
01/02/2022	星期日	-1
01/05/2022	星期三	-1

日期	week_day	prev_week
01/06/2022	星期四	-1
01/08/2022	星期六	-1
01/09/2022	星期日	0
01/10/2022	星期一	0
01/11/2022	星期二	0
01/12/2022	星期三	0
01/13/2022	星期四	0
01/14/2022	星期五	0
01/15/2022	星期六	0
01/16/2022	星期日	0
01/17/2022	星期一	0
01/18/2022	星期二	0
01/26/2022	星期三	0
01/27/2022	星期四	0
01/28/2022	星期五	0
01/29/2022	星期六	0
01/30/2022	星期日	0
01/31/2022	星期一	0

使用 `-1` 作為 `inweek()` 函數中的 `period_no` 引數，可將比較子週的邊界往回偏移完整七天。透過 `0` 的 `period_no`，該週會在 1 月 9 日和 15 日之間。但在此範例中，`-1` 的 `period_no` 會將此區段的開始和結束邊界往回偏移一週。日期邊界變成 1 月 2 日至 1 月 8 日。

醒目提示 inweek() 函數範圍的行事曆圖表

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					

因此，發生在 1 月 2 日和 1 月 8 日之間的任何交易將會傳回布林值結果 TRUE。

範例 3 - first_week_day

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的相同資料集，這載入到稱為「Transactions」的表格中。
- 設為 6 (星期日) 的 FirstWeekDay 系統變數。
- 前置載入包含下列內容：
  - 設為欄位「in_week」的 inweek() 函數，決定哪些交易發生在 2022 年 1 月 14 日該週。
  - 設為欄位「week_day」的 weekday() 函數，顯示該週的哪一天對應至每個日期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- week_day
- in_week

結果表格

日期	week_day	in_week
01/02/2022	星期日	0
01/05/2022	星期三	0

日期	week_day	in_week
01/06/2022	星期四	0
01/08/2022	星期六	0
01/09/2022	星期日	0
01/10/2022	星期一	-1
01/11/2022	星期二	-1
01/12/2022	星期三	-1
01/13/2022	星期四	-1
01/14/2022	星期五	-1
01/15/2022	星期六	-1
01/16/2022	星期日	-1
01/17/2022	星期一	0
01/18/2022	星期二	0
01/26/2022	星期三	0
01/27/2022	星期四	0
01/28/2022	星期五	0
01/29/2022	星期六	0
01/30/2022	星期日	0
01/31/2022	星期一	0

使用 0 作為 `inweek()` 函數中的 `first_week_day` 引數會取代 `FirstWeekDay` 系統變數並將星期一設定為週的第一天。

醒目提示 inweek() 函數範圍的行事曆圖表

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					

因此，發生在 1 月 10 日和 16 日之間的任何交易將會傳回布林值結果 TRUE。

範例 4 - 圖表物件範例

載入指令碼和圖表運算式

概覽

使用與第一個範例相同的資料集和情境。

不過，在此範例中，資料集保持不變並且會載入到應用程式中。在結果表格中建立量值，以判定哪些交易發生在 2022 年 1 月 14 日該週。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：

- date

建立下列量值：

- =inweek (date,'01/14/2022',0)，用來計算交易是否發生在與 1 月 14 日相同的一週。
- =weekday(date)，用來顯示該週的哪個日子對應至每個日期。

結果表格

日期	week_day	=inweek (date,'01/14/2022',0)
01/02/2022	星期日	0
01/05/2022	星期三	0
01/06/2022	星期四	0
01/08/2022	星期六	0
01/09/2022	星期日	-1
01/10/2022	星期一	-1
01/11/2022	星期二	-1
01/12/2022	星期三	-1

日期	week_day	=inweek (date,'01/14/2022',0)
01/13/2022	星期四	-1
01/14/2022	星期五	-1
01/15/2022	星期六	-1
01/16/2022	星期日	0
01/17/2022	星期一	0
01/18/2022	星期二	0
01/26/2022	星期三	0
01/27/2022	星期四	0
01/28/2022	星期五	0
01/29/2022	星期六	0
01/30/2022	星期日	0
01/31/2022	星期一	0

會使用 inweek() 函數在圖表中建立「in_week」量值。第一個引數識別正在評估哪個欄位。第二個引數是 1 月 14 日的硬式編碼日期，這是 base_date。base_date 引數搭配 FirstWeekDay 系統變數運作，以識別比較子週。0 的 period_no 是最終引數。

FirstWeekDay 系統變數決定週於星期日開始，於星期六結束。因此，1 月會根據下圖分為數週，其中 1 月 9 日和 15 日之間的日期提供適用於 inweek() 計算的有效期間：

醒目提示 inweek() 函數範圍的行事曆圖表

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					

發生在 1 月 9 日和 1 月 15 日之間的任何交易都會傳回布林值結果 TRUE。

範例 5 - 情境

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入到稱為「Products」之表格的資料集。
- 該表格含有下列欄位：
  - 產品 ID
  - 產品類型
  - 製造日期
  - 成本價格

這識別出由於設備錯誤，在 1 月 12 日該週製造的產品有瑕疵。最終使用者希望圖表依週顯示製造的哪些產品狀態為「瑕疵」或「無缺失」，以及在該週製造的產品成本。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：

- =weekname(manufacture_date)

建立下列量值：

- =if(only(inweek(manufacture_date,makedate(2022,01,12),0)),'Defective','Faultless')，用來使用 inweek() 函數識別哪些產品為瑕疵，那些為無缺失。
- =sum(cost_price)，用來顯示每個產品成本的總和。

請執行下列動作：

1. 將量值的**數字格式**設定為**金錢**。
2. 在**外觀**之下，關閉**總計**。

結果表格

weekname (manufacture_date)	=if(only(inweek(manufacture_date,makedate(2022,01,12),0)), 'Defective','Faultless')	Sum(cost_price)
2022/02	無缺失	200.09
2022/03	瑕疵	441.51
2022/04	無缺失	178.41
2022/05	無缺失	231.67
2022/06	無缺失	163.91

inweek() 函數會在評估每個產品的製造日期時傳回布林值。對於在 1 月 12 日該週製造的任何產品，inweek() 函數會傳回布林值 TRUE 並將產品標記為「瑕疵」。對於傳回 FALSE 值的任何產品 (因此這不是在該週製造)，會將產品標記為「無缺失」。

inweektodate

如果 **timestamp** 位於包含 **base_date** 的週部分內，且不超過 (可包含) **base_date** 的最後一毫秒，則此函數會傳回 True。

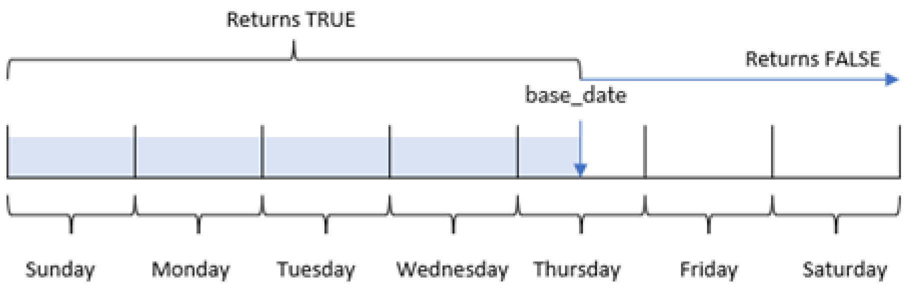
語法：

```
InWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

傳回的資料類型： 布林

在 Qlik Sense 中，布林值 true 值以 -1 代表，而 false 值以 0 代表。

inweektodate 函數的圖表



inweektodate() 函數使用 base_date 參數識別一週區段的最大邊界日期，以及該週開始的對應日期，這根據 FirstWeekDay 系統變數 (或使用者定義的 first_week_day 參數) 來進行。定義此週區段後，該函數則會在比較規定的日期值與該區段時，傳回布林值結果。

什麼情況下使用

inweektodate() 函數會傳回布林值結果。通常，此函數類型將作為 if 運算式中的條件使用。這根據評估的日期是否發生於問題中的週截至 (含) 特定日期，傳回彙總或計算。

例如，inweektodate() 函數可用來計算指定的週截至特定日期期間進行的所有銷售。

引數

引數	描述
timestamp	您要與 **base_date** 比較的日期。
base_date	用來評估週的日期。
period_no	週可以使用 **period_no** 位移。**period_no** 是一個整數，其中值 0 表示包含 **base_date** 的週。負值的 **period_no** 表示之前的週，正值表示之後的週。
first_week_day	依照預設，該週的第一天為星期日 (如 FirstWeekDay 系統變數所決定)，從星期六和星期日之間的午夜開始。**first_week_day** 參數取代 **FirstWeekDay** 變數。若要指出從另一日開始的週，請在 0 和 6 之間指定一個旗標。 對於在星期一開始並在星期日結束的週，請對星期一使用 0 的旗標，對星期二使用 1，對星期三使用 2，對星期四使用 3，對星期五使用 4，對星期六使用 5，並對星期日使用 6。

函數範例

範例	互動
inweektodate('01/12/2006', '01/12/2006', 0)	傳回 TRUE。
inweektodate('01/12/2006', '01/11/2006', 0)	傳回 FALSE。
inweektodate('01/12/2006', '01/18/2006', -1)	傳回 FALSE。 因為 period_no 指定為 -1，測量 timestamp 所針對的有效資料是 01/11/2006。
inweektodate('01/11/2006', '01/12/2006', 0, 3)	傳回 FALSE，因為 first_week_day 指定為 3 (星期四)，這使得 01/12/2006 成為包含 01/12/2006 的週之後星期的第一天。

這些主題可協助您使用此函數：

相關主題

主題	預設旗標 / 值	描述
FirstWeekDay (page 212)	6 / 星期日	定義每週開始的日子。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典,資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2022 年 1 月交易的資料集,這載入到稱為「Transactions」的表格中。
- 以 TimestampFormat='M/D/YYYY h:mm:ss[.fff]' 格式提供的日期欄位。
- 建立欄位 in_week_to_date,這決定哪些交易發生在 2022 年該週的 1 月 14 日之前。
- 使用 weekday() 函數建立附加欄位,名稱為 weekday。會建立此新欄位,以顯示該週的哪個日子對應至每個日期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- week_day
- in_week_to_date

結果表格

日期	week_day	in_week_to_date
2022-01-02 12:22:06	星期日	0
2022-01-05 01:02:30	星期三	0
2022-01-06 15:36:20	星期四	0
2022-01-08 10:58:35	星期六	0
2022-01-09 08:53:32	星期日	-1
2022-01-10 21:13:01	星期一	-1
2022-01-11 00:57:13	星期二	-1
2022-01-12 09:26:02	星期三	-1
2022-01-13 15:05:09	星期四	-1
2022-01-14 18:44:57	星期五	-1
2022-01-15 06:10:46	星期六	0
2022-01-16 06:39:27	星期日	0
2022-01-17 10:44:16	星期一	0
2022-01-18 18:48:17	星期二	0
2022-01-26 04:36:03	星期三	0
2022-01-27 08:07:49	星期四	0
2022-01-28 12:24:29	星期五	0
2022-01-30 11:56:56	星期日	0
2022-01-30 14:40:19	星期日	0
2022-01-31 05:28:21	星期一	0

「in_week_to_date」欄位使用 inweektodate() 函數在前置 LOAD 陳述式中建立。提供的第一個引數識別正在評估哪個欄位。第二個引數是 1 月 14 日的硬式編碼日期，即識別要分割哪一週並定義該區段結束邊界的 base_date。值為 0 的 period_no 是最終引數，表示該函數沒有比較分割的週之前或之後的週。

FirstWeekDay 系統變數決定週於星期日開始，於星期六結束。因此，1 月會根據下圖分為數週，其中 1 月 9 日和 14 日之間的日期提供適用於 inweekdodate() 計算的有效期間：

行事曆圖表顯示會傳回布林值結果 TRUE 的交易日期

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					

發生在 1 月 9 日和 14 日之間的任何交易都會傳回布林值結果 TRUE。該日期之前和之後的交易會傳回布林值結果 FALSE。

範例 2 – period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 建立欄位 prev_week_to_date，這決定哪些交易發生在結束於 2022 年 1 月 14 日之週區段的完整一週之前。
- 使用 weekday() 函數建立附加欄位，名稱為 weekday。這用來顯示該週的哪個日子對應至每個日期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- week_day
- prev_week_to_date

結果表格

日期	week_day	prev_week_to_date
2022-01-02 12:22:06	星期日	-1
2022-01-05 01:02:30	星期三	-1
2022-01-06 15:36:20	星期四	-1
2022-01-08 10:58:35	星期六	0
2022-01-09 08:53:32	星期日	0
2022-01-10 21:13:01	星期一	0

日期	week_day	prev_week_to_date
2022-01-11 00:57:13	星期二	0
2022-01-12 09:26:02	星期三	0
2022-01-13 15:05:09	星期四	0
2022-01-14 18:44:57	星期五	0
2022-01-15 06:10:46	星期六	0
2022-01-16 06:39:27	星期日	0
2022-01-17 10:44:16	星期一	0
2022-01-18 18:48:17	星期二	0
2022-01-26 04:36:03	星期三	0
2022-01-27 08:07:49	星期四	0
2022-01-28 12:24:29	星期五	0
2022-01-30 11:56:56	星期日	0
2022-01-30 14:40:19	星期日	0
2022-01-31 05:28:21	星期一	0

`period_no` 值 -1 表示 `inweektodate ()` 函數比較輸入的季區段與上一週。週區段原本等同於 1 月 9 日和 1 月 14 日之間。然後 `period_no` 將此區段的開始和結束邊界都往前移動一週，造成日期邊界成為 1 月 2 日至 1 月 7 日。

行事曆圖表顯示會傳回布林值結果 TRUE 的交易日期

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					

因此，發生在 1 月 2 日和 8 日之間 (不含 1 月 8 日當日) 的任何交易將會傳回布林值結果 `TRUE`。

範例 3 – first_week_day

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 與第一個範例相同的資料集和情境。
- 建立欄位 `in_week_to_date`, 這決定在 2022 年 1 月 14 日之前的該週內發生哪些交易。
- 使用 `weekday()` 函數建立附加欄位, 名稱為 `weekday`。這用來顯示該週的哪個日子對應至每個日期。

在此範例中, 我們將星期一視為週的第一天。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- week_day
- in_week_to_date

結果表格

日期	week_day	in_week_to_date
2022-01-02 12:22:06	星期日	0
2022-01-05 01:02:30	星期三	0
2022-01-06 15:36:20	星期四	0
2022-01-08 10:58:35	星期六	0
2022-01-09 08:53:32	星期日	0
2022-01-10 21:13:01	星期一	-1
2022-01-11 00:57:13	星期二	-1
2022-01-12 09:26:02	星期三	-1
2022-01-13 15:05:09	星期四	-1
2022-01-14 18:44:57	星期五	-1
2022-01-15 06:10:46	星期六	0
2022-01-16 06:39:27	星期日	0
2022-01-17 10:44:16	星期一	0
2022-01-18 18:48:17	星期二	0
2022-01-26 04:36:03	星期三	0
2022-01-27 08:07:49	星期四	0
2022-01-28 12:24:29	星期五	0
2022-01-30 11:56:56	星期日	0
2022-01-30 14:40:19	星期日	0
2022-01-31 05:28:21	星期一	0

若使用 0 作為 inweektodate() 函數中的 first_week_day 引數，該函數引數會取代 FirstWeekDay 系統變數並將星期一設定為週的第一天。

行事曆圖表顯示會傳回布林值結果 TRUE 的交易日期

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						

因此, 發生在 1 月 10 日和 14 日之間的任何交易將會傳回布林值結果 TRUE, 而日期落在這些邊界之外的交易將會傳回 FALSE 的值。

範例 4 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。不過, 在此範例中, 不變的資料集會載入到應用程式中。會建立決定哪些交易發生在該週截至 2022 年 1 月 14 日的計算, 作為圖表物件中的量值。

載入指令碼

```
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
];
```

結果

請執行下列動作:

1. 載入資料並開啟工作表。建立新的表格並將此欄位新增為維度: `date`。
2. 若要計算交易是否發生在同一週截至 1 月 14 日, 建立下列量值:
   `=inweektodate(date,'01/14/2022',0)`
3. 若要顯示該週的哪個日子對應至每個日期, 建立附加量值:
   `=weekday(date)`

結果表格

日期	week_day	in_week_to_date
2022-01-02 12:22:06	星期日	0
2022-01-05 01:02:30	星期三	0
2022-01-06 15:36:20	星期四	0
2022-01-08 10:58:35	星期六	0
2022-01-09 08:53:32	星期日	-1
2022-01-10 21:13:01	星期一	-1
2022-01-11 00:57:13	星期二	-1
2022-01-12 09:26:02	星期三	-1
2022-01-13 15:05:09	星期四	-1
2022-01-14 18:44:57	星期五	-1
2022-01-15 06:10:46	星期六	0
2022-01-16 06:39:27	星期日	0
2022-01-17 10:44:16	星期一	0
2022-01-18 18:48:17	星期二	0
2022-01-26 04:36:03	星期三	0

日期	week_day	in_week_to_date
2022-01-27 08:07:49	星期四	0
2022-01-28 12:24:29	星期五	0
2022-01-30 11:56:56	星期日	0
2022-01-30 14:40:19	星期日	0
2022-01-31 05:28:21	星期一	0

會使用 inweektodate() 函數在圖表物件中建立 in_week_to_date 欄位作為量值。提供的第一個引數識別正在評估哪個欄位。第二個引數是 1 月 14 日的硬式編碼日期，即識別要分割哪一週並定義該區段結束邊界的 base_date。值為 0 的 period_no 是最終引數，表示該函數沒有比較分割的週之前或之後的週。

FirstWeekDay 系統變數決定週於星期日開始，於星期六結束。因此，1 月會根據下圖分為數週，其中 1 月 9 日和 14 日之間的日期提供適用於 inweekdodate() 計算的有效期間：

行事曆圖表顯示會傳回布林值結果 TRUE 的交易日期

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					

發生在 1 月 9 日和 14 日之間的任何交易都會傳回布林值結果 TRUE。該日期之前和之後的交易會傳回布林值結果 FALSE。

範例 5 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入到稱為 `Products` 之表格的資料集。
- 關於產品 ID、製造日期和成本價格的資訊。

這識別出由於設備錯誤，在 1 月 12 日該週製造的產品有瑕疵。問題已在 1 月 13 日解決。最終使用者希望圖表物件依週顯示製造的產品狀態為「瑕疵」或「故障」，以及在該週製造的產品成本。

載入指令碼

```
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
];
```

結果

請執行下列動作：

1. 載入資料並開啟工作表。建立新的表格。建立維度以顯示週名稱：
   =weekname(manufacture_date)
2. 接下來，建立維度以識別哪些產品為瑕疵，那些為故障：
   =if(inWeektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')
3. 建立量值以加總產品的 cost_price：
   =sum(cost_price)
4. 將量值的**數字格式**設定為**金錢**。

結果表格

weekname(manufacture_date)	if(inweektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')	Sum(cost_price)
2022/02	無缺失	$200.09
2022/03	瑕疵	$263.46
2022/03	無缺失	$178.05
2022/04	無缺失	$178.41
2022/05	無缺失	$147.46
2022/06	無缺失	$248.12

`inweektodate()` 函數會在評估每個產品的製造日期時傳回布林值。對於傳回布林值 `TRUE` 的項目，會將產品標記為 `'Defective'`。對於傳回值 `FALSE` 的任何產品，以及因此不是在該週截至 1 月 12 日製作的產品，會將產品標記為 `'Faultless'`。

inyear

此函數會傳回 True，前提是如果 **timestamp** 位於包含 **base_date** 的年中。

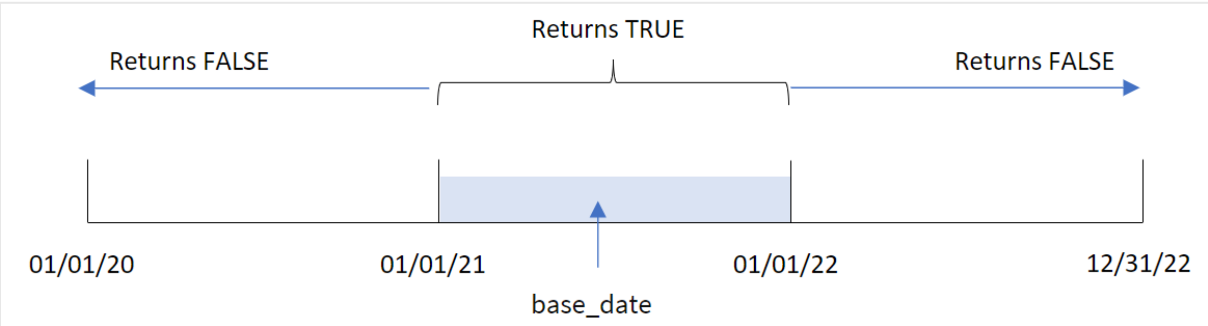
語法：

```
InYear (timestamp, base_date, period_no [, first_month_of_year])
```

傳回的資料類型： 布林

在 Qlik Sense 中，布林值 true 值以 -1 代表，而 false 值以 0 代表。

inyear() 函數範圍的圖表



`inyear()` 函數在比較所選日期值與 `base_date` 定義的年份時傳回布林值結果。

什麼情況下使用

`inyear()` 函數會傳回布林值結果。通常，此函數類型將作為 `if expression` 中的條件使用。這根據評估的日期是否發生於問題中的年份，傳回彙總或計算。例如，`inyear()` 函數可用來識別發生在定義年份的所有銷售。

引數

引數	描述
timestamp	您要與 **base_date** 比較的日期。
base_date	用來評估年的日期。
period_no	年可以使用 **period_no** 位移。**period_no** 是一個整數，其值 0 表示包含 **base_date** 的年。負值的 **period_no** 表示之前的年度，正值表示之後的年度。
first_month_of_year	如果要使用不起始於 1 月的 (會計) 年度，可在 **first_month_of_year** 中指定介於 2 和 12 之間的值。

您可以在 `first_month_of_year` 引數中使用下列值設定一年的第一個月：

first_month_of_year 值

月	值
二月	2
三月	3
四月	4
五月	5
六月	6
七月	7
八月	8
九月	9
十月	10
十一月	11
十二月	12

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
`inyear ('01/25/2013', '01/01/2013', 0 )`	傳回 TRUE
`inyear ('01/25/2012', '01/01/2013', 0)`	傳回 FALSE
`inyear ('01/25/2013', '01/01/2013', -1)`	傳回 FALSE
`inyear ('01/25/2012', '01/01/2013', -1 )`	傳回 TRUE
`inyear ('01/25/2013', '01/01/2013', 0, 3)`	傳回 TRUE `base_date` 和 `first_month_of_year` 的值指定時間戳記必須落在 01/03/2012 和 02/28/2013 內
`inyear ('03/25/2013', '07/01/2013', 0, 3 )`	傳回 TRUE

範例 1 - 基本範例

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2020 年和 2022 年之間交易的資料集,這載入到稱為「Transactions」的表格中。
- 包含設為「in_year」欄位之 inyear() 函數的前置載入,這判定哪些交易發生在與 2021 年 7 月 26 日相同的年份。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- in_year

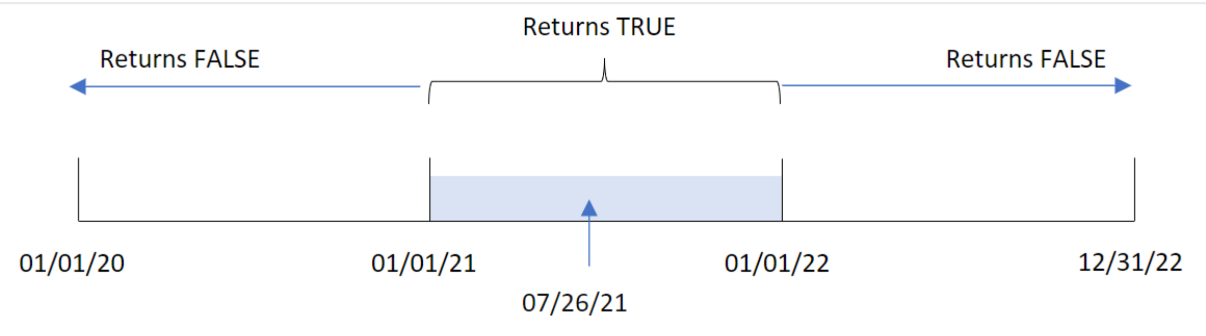
結果表格

日期	in_year
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1

日期	in_year
07/26/2021	-1
12/27/2021	-1
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

「in_year」欄位使用 `inyear()` 函數在前置 LOAD 陳述式中建立。第一個引數識別正在評估哪個欄位。第二個引數是 2021 年 7 月 26 日的硬式編碼日期，即判定比較子年份的 `base_date`。0 的 `period_no` 是最終引數，表示 `inyear()` 函數沒有比較該年之前或之後的年。

以 7 月 26 日作為基底日期的 inyear() 函數範圍圖表



任何發生在 2021 年的交易會傳回 TRUE 的布林值結果。

範例 2 - period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2020 年和 2022 年之間交易的資料集，這載入到稱為「`Transactions`」的表格中。
- 包含設為「`previous_year`」欄位之 `inyear()` 函數的前置載入，這判定哪些交易發生在包含 2021 年 7 月 26 日之年份的前一年。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- previous_year

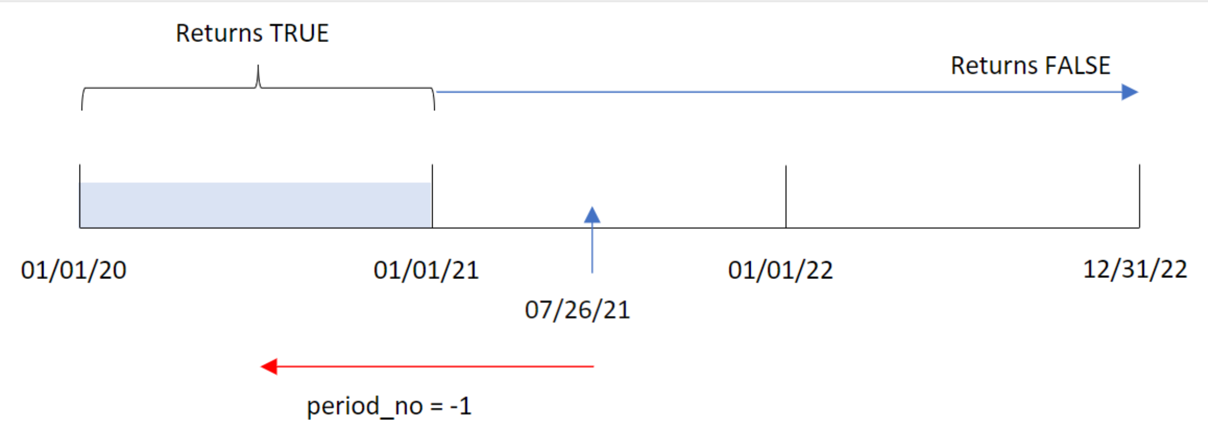
結果表格

日期	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

日期	previous_year
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

使用 -1 作為 inyear() 函數中的 period_no 引數，將比較子年份邊界往回偏移一整年。2021 年原本識別為比較子年份。period_no 將比較子年份偏移一年，讓 2020 成為比較子年份。

period_no 引數設定為 -1 的 inyear() 函數範圍圖表



因此，任何發生在 2020 年的交易會傳回 TRUE 的布林值結果。

範例 3 - first_month_of_year

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2020 年和 2022 年之間交易的資料集，這載入到稱為「Transactions」的表格中。
- 包含設為「in_year」欄位之 inyear() 函數的前置載入，這判定哪些交易發生在與 2021 年 7 月 26 日相同的年份。

不過，在此範例中，組織政策適用於 3 月作為會計年度第一個月份的情況。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

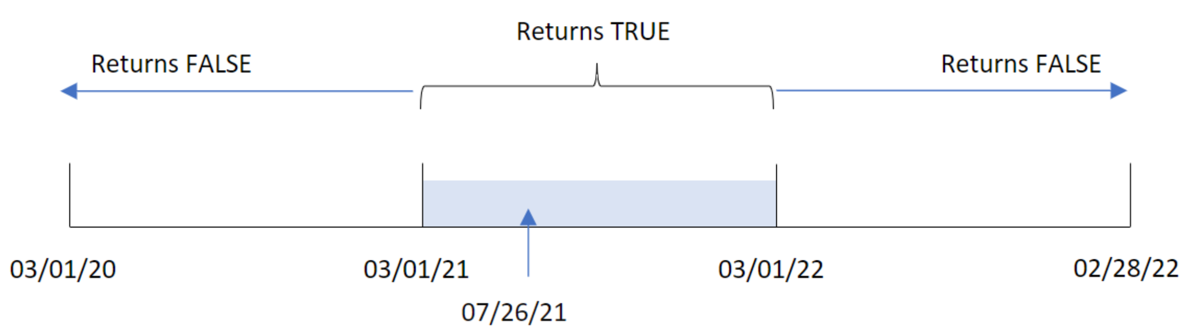
- date
- in_year

結果表格

日期	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
12/27/2021	-1
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

使用 3 作為 `inyear()` 函數中的 `first_month_of_year` 引數會讓該年於 3 月 1 日開始，並於 2 月底結束。

將 3 月設定為該年第一個月的 `inyear()` 函數範圍圖表



因此, 發生在 2021 年 3 月 1 日和 2022 年 3 月 1 日之間的任何交易將會傳回布林值結果 TRUE。

範例 4 - 圖表物件範例

載入指令碼和圖表運算式

概覽

使用與第一個範例相同的資料集和情境。

不過, 在此範例中, 資料集保持不變並且會載入到應用程式中。會建立決定交易是否發生在與 2021 年 7 月 26 日相同的一年, 作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度:

- date

若要計算交易是否發生在與 2021 年 7 月 26 日相同的一年, 建立下列量值:

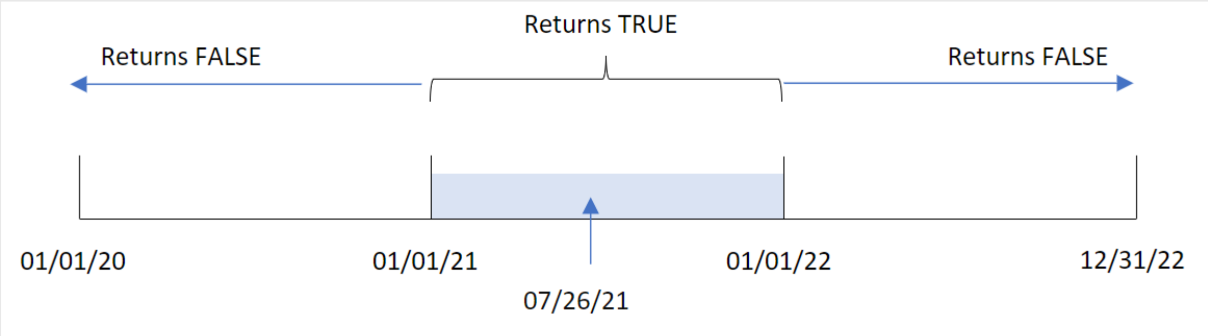
- =inyear(date,'07/26/2021', 0)

結果表格

日期	=inyear(date,'07/26/2021',0)
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
12/27/2021	-1
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

會使用 inyear() 函數在圖表中建立「in_year」欄位。第一個引數識別正在評估哪個欄位。第二個引數是 2021 年 7 月 26 日的硬式編碼日期，即判定比較子年份的 base_date。0 的 period_no 是最終引數，表示 inyear() 函數沒有比較該年之前或之後的年。

以 7 月 27 日作為基底日期的 inyear() 函數範圍圖表



任何發生在 2021 年的交易會傳回 TRUE 的布林值結果。

範例 5 - 情境

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入到稱為「Products」之表格的資料集。
- 該表格含有下列欄位：
  - 產品 ID
  - 產品類型
  - 製造日期
  - 成本價格

最終使用者希望圖表物件依產品類型顯示在 2021 年製造的產品成本。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：

- product_type

若要計算在 2021 年製造的每個產品總和，建立下列量值：

- =sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))

請執行下列動作：

1. 將量值的**數字格式**設定為**金錢**。
2. 在**外觀**之下，關閉**總計**。

結果表格

product_type	=sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))
產品 A	$95.93
產品 B	$128.66
產品 C	$61.89
產品 D	$171.21

inyear() 函數會在評估每個產品的製造日期時傳回布林值。對於在 2021 年製造的任何產品，inyear() 函數會傳回布林值 TRUE 並顯示 cost_price 的總和。

inyeartodate

如果 **timestamp** 位於包含 **base_date** 的年部分內，且不超過 (可包含) **base_date** 的最後一毫秒，則此函數會傳回 True。

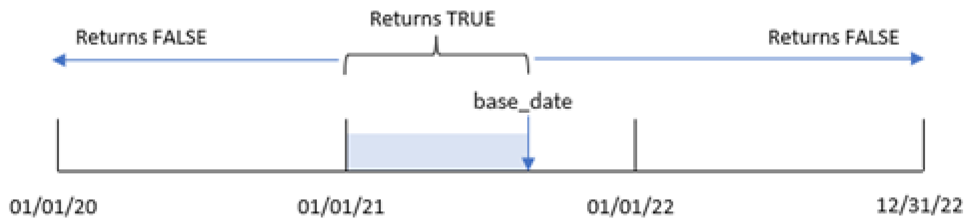
語法：

```
InYearToDate (timestamp, base_date, period_no[, first_month_of_year])
```

傳回的資料類型: 布林

> 在 Qlik Sense 中,布林值 true 值以 -1 代表,而 false 值以 0 代表。

inyeartodate 函數的圖表



`inyeartodate()` 函數將會分割一年中的特定部分,並透過 base_date 識別該年區段的最大允許日期。然後該函數會評估日期欄位或值是否落在此區段內,並傳回布林值結果。

引數

引數	描述
timestamp	您要與 **base_date** 比較的日期。
base_date	用來評估年的日期。
period_no	年可以使用 **period_no** 位移。**period_no** 是一個整數,其值 0 表示包含 **base_date** 的年。負值的 **period_no** 表示之前的年度,正值表示之後的年度。
first_month_of_year	如果要使用不起始於 1 月的 (會計) 年度,可在 **first_month_of_year** 中指定介於 2 和 12 之間的值。

什麼情況下使用

`inyeartodate()` 函數會傳回布林值結果。通常,此函數類型將作為 `if` 運算式中的條件使用。這會根據評估的日期是否發生於該年截至 (含) 問題中的日期,傳回彙總或計算。

例如,`inyeartodate()` 函數可用來識別某年截至特定日期製造的所有設備。

這些範例使用日期格式 MM/DD/YYYY。日期格式是在位於資料載入指令碼頂端的 `SET DateFormat` 陳述式中指定。變更範例中的格式,以滿足您的需求。

函數範例

範例	結果
`inyeartodate ('01/25/2013', '02/01/2013', 0)`	傳回 `TRUE`。
`inyeartodate ('01/25/2012', '01/01/2013', 0)`	傳回 `FALSE`。
`inyeartodate ('01/25/2012', '02/01/2013', -1)`	傳回 `TRUE`。

範例	結果
inyeartodate ('11/25/2012', '01/31/2013', 0, 4)	傳回 TRUE。 timestamp 的值落在從第四個月開始的會計年度內，並在 base_date 的值之前。
inyeartodate ('3/31/2013', '01/31/2013', 0, 4)	傳回 FALSE。 與先前的範例相比較，timestamp 仍在會計年度內，但是在 base_date 的值之後，因此它在年度部分之外。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2020 年和 2022 年之間交易的資料集，這載入到稱為「Transactions」的表格中。
- 以 DateFormat 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 in_year_to_date，這決定哪些交易發生在該年的 2021 年 7 月 26 日之前。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- `date`
- `in_year_to_date`

結果表格

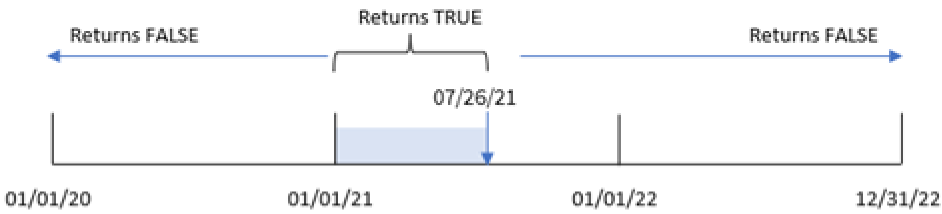
日期	in_year_to_date
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
06/14/2020	0
08/07/2020	0
09/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1

日期	in_year_to_date
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
07/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

「in_year_to_date」欄位使用 inyeartodate() 函數在前置 LOAD 陳述式中建立。提供的第一個引數識別正在評估哪個欄位。

第二個引數是 2021 年 7 月 26 日的硬式編碼日期，即識別年區段結束邊界的 base_date。值為 0 的 period_no 是最終引數，表示該函數沒有比較分割的年之前或之後的年。

inyeartodate 函數的圖表，無其他引數



發生在 1 月 1 日和 7 月 26 日之間的任何交易都會傳回布林值結果 TRUE。2021 年之前和超過 2021 年 7 月 26 日的交易日期會傳回 FALSE。

範例 2 – period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 建立欄位 previous_year_to_date，這決定哪些交易發生在結束於 2021 年 7 月 26 日之年區段的完整一年之前。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- previous_year_to_date

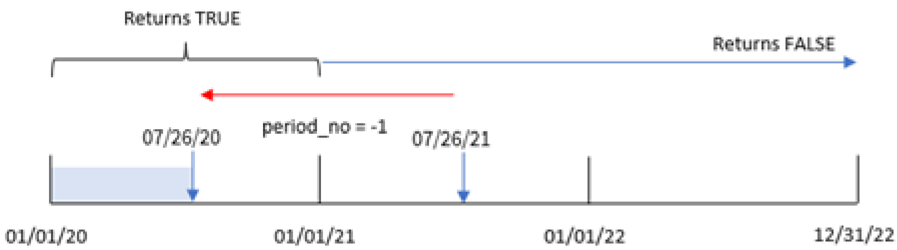
結果表格

日期	previous_year_to_date
01/13/2020	-1
02/26/2020	-1
03/27/2020	-1
04/16/2020	-1

日期	previous_year_to_date
05/21/2020	-1
06/14/2020	-1
08/07/2020	0
09/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
07/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

period_no 值 -1 表示 inyeartodate () 函數比較輸入的季區段與上一年。透過輸入日期 2021 年 7 月 26 日，從 2021 年 1 月 1 日到 2021 年 7 月 26 日的區段原本被視為年初至今。然後 period_no 將此區段往前移動一整年，造成日期邊界變成 2020 年 1 月 1 日到 7 月 26 日。

inyeartodate 函數的圖表，period_no 範例



因此，發生在 2020 年 1 月 1 日和 7 月 26 日之間的任何交易將會傳回布林值結果 TRUE。

範例 3 – first_month_of_year

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 與第一個範例相同的資料集和情境。
- 建立欄位 `in_year_to_date`, 這決定哪些交易發生在同一年的 2021 年 7 月 26 日之前。

在此範例中, 我們將 3 月設為會計年度的第一個月。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

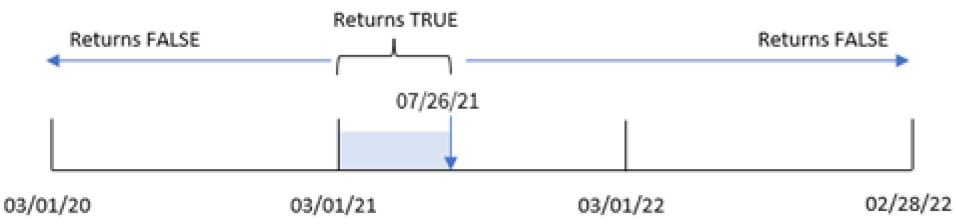
- date
- in_year_to_date

結果表格

日期	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

若使用 3 作為 `inyeartodate()` 函數中的 `first_month_of_year` 引數，該函數會讓該年從 3 月 1 日開始。值為 2021 年 7 月 26 日的 `base_date` 則會設定該年區段的結束日期。

inyeartodate 函數的圖表, first_month_of_year 範例



因此, 發生在 2021 年 3 月 1 日和 7 月 26 日之間的任何交易將會傳回布林值結果 TRUE, 而日期落在這些邊界之外的交易將會傳回 FALSE 的值。

範例 4 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。不過, 在此範例中, 不變的資料集會載入到應用程式中。會建立決定哪些交易發生在同一年截至 2021 年 7 月 26 日的計算, 作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度:date。

建立下列量值:

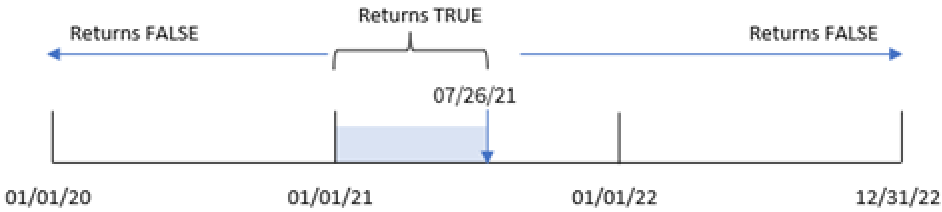
```
=inyeartodate(date,'07/26/2021', 0)
```

結果表格

日期	=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

會使用 inyeartodate() 函數在圖表物件中建立 in_year_to_date 量值。提供的第一個引數識別正在評估哪個欄位。第二個引數是 2021 年 7 月 26 日的硬式編碼日期,即識別比較子年區段結束邊界的 base_date。值為 0 的 period_no 是最終引數,表示該函數沒有比較分割的年之前或之後的年。

inyeartodate 函數的圖表，圖表物件範例



發生在 2021 年 1 月 1 日和 7 月 26 日之間的任何交易都會傳回布林值結果 TRUE。2021 年之前和 2021 年 7 月 26 日之後的交易日期會傳回 FALSE。

範例 5 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入到稱為 Products 之表格的資料集。
- 關於產品 ID、產品類型、製造日期和成本價格的資訊。

最終使用者希望圖表物件依產品類型顯示在 2021 年截至 7 月 26 日製造的產品成本。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：product_type。

建立計算在 2021 年的 7 月 27 日之前製造的每個產品總和的量值：

```
=sum(if(inyeartodate(manufacture_date,makedate(2021,07,26),0),cost_price,0))
```

將量值的**數字格式**設定為**金錢**。

結果表格

product_type	**=sum(if(inyeartodate(manufacture_date,makedate (2021,07,26),0),cost_price,0))**
產品 A	$95.93
產品 B	$128.66
產品 C	$61.89
產品 D	$146.09

inyeartodate() 函數會在評估每個產品的製造日期時傳回布林值。對於在 2021 年的 7 月 27 日之前製造的任何產品，inyeartodate() 函數會傳回布林值 TRUE 並加總 cost_price。

產品 D 是也在 2021 年 7 月 26 日之後製造的唯一產品。具有 product_ID 8203 的項目在 12 月 27 日製造且成本為 $25.12。因此，此成本不包括在圖表物件產品 D 的總計中。

lastworkdate

lastworkdate 函數會傳回開始於 **start_date**，並考慮到所有選用的列出 **holiday**，而可達到 **no_of_workdays** (星期一至星期五) 的最早結束日期。**start_date** 和 **holiday** 應該是有效的日期或時間戳記。

語法：

```
lastworkdate(start_date, no_of_workdays {, holiday})
```

傳回的資料類型： 整數

顯示如何使用 `lastworkdate()` 函數的行事曆

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			

限制

對於工作週不是星期一開始和星期五結束的相關地區或情境，沒有方法可修改 `lastworkdate()` 函數。

假日參數必須是字串常數。這不接受運算式。

什麼情況下使用

若使用者想要根據專案開始時間和將會發生在該期間的假期，計算提出的專案或指派工作結束日期，`lastworkdate()` 函數通常用來作為運算式的一部分。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

引數

引數	描述
start_date	要評估的開始日期。
no_of_ workdays	要達到的工作日數目。
holiday	要從工作日中排除的假期。假日以字串常數日期表示。您可以指定多個假日日期，以逗點分隔。 **範例：** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

範例 1 - 基本範例

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 資料集包含專案 ID、專案開始日期和專案所需的預計花費天數。資料集載入到稱為「`Projects`」的表格。
- 包含設定為「end_date」欄位之 `lastworkdate()` 函數並識別每個專案之排定結束時間的前置載入。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- start_date
- effort
- end_date

結果表格

id	start_date	花費天數	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

因為沒有已排程的假期，該函數會將定義的工作天數星期一至星期五加到開始日期，以尋找可能的最早結束日期。

下列行事曆顯示專案 3 的開始和結束日期，其中工作日以綠色醒目提示。

行事曆顯示專案 3 的開始和結束日期

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				

範例 2 - 單一假期

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 資料集包含專案 ID、專案開始日期和專案所需的預計花費天數。資料集載入到稱為「`Projects`」的表格。
- 包含設定為 end_date 欄位之 `lastworkdate()` 函數並識別每個專案之排定結束時間的前置載入。

不過,有一個排程於 2022 年 5 月 18 日的假期。前置載入中的 `lastworkdate()` 函數會在第三引數中納入該假期,以識別每個專案的排定結束時間。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- id
- start_date
- effort
- end_date

結果表格

id	start_date	花費天數	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

輸入單一排程假期作為 `lastworkdate()` 函數中的第三引數。因此,專案 3 的結束日期會往後偏移一天,因為假期發生在結束日期之前的工作天之一。

下列行事曆顯示專案 3 的開始和結束日期,並顯示假期將專案的結束日期變更一天。

行事曆顯示專案 3 的開始和結束日期, 其中假期為 5 月 18 日

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				

範例 3 - 多個假期

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 資料集包含專案 ID、專案開始日期和專案所需的預計花費天數。資料集載入到稱為「`Projects`」的表格。
- 包含設定為 end_date 欄位之 `lastworkdate()` 函數並識別每個專案之排定結束時間的前置載入。

不過, 有三個假期排程於 5 月 19、20、21 和 22 日。前置載入中的 `lastworkdate()` 函數會在第三引數中納入每個假期, 以識別每個專案的排定結束時間。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- start_date
- effort
- end_date

結果表格

id	start_date	花費天數	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

在開始日期和工作天數之後輸入四個假期作為 `lastworkdate()` 函數中的引數清單。

下列行事曆顯示專案 3 的開始和結束日期，並顯示假期將專案的結束日期變更三天。

行事曆顯示專案 3 的開始和結束日期，其中假期為 5 月 19 至 22 日

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				

範例 4 - 單一假期 (圖表)

載入指令碼和圖表運算式

概覽

使用與第一個範例相同的資料集和情境。

不過，在此範例中，資料集保持不變並且會載入到應用程式中。會計算 `end_date` 欄位作為圖表中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- `id`
- `start_date`
- `effort`

若要計算 end_date，建立下列量值：

- `=LastWorkDate(start_date,effort,'05/18/2022')`

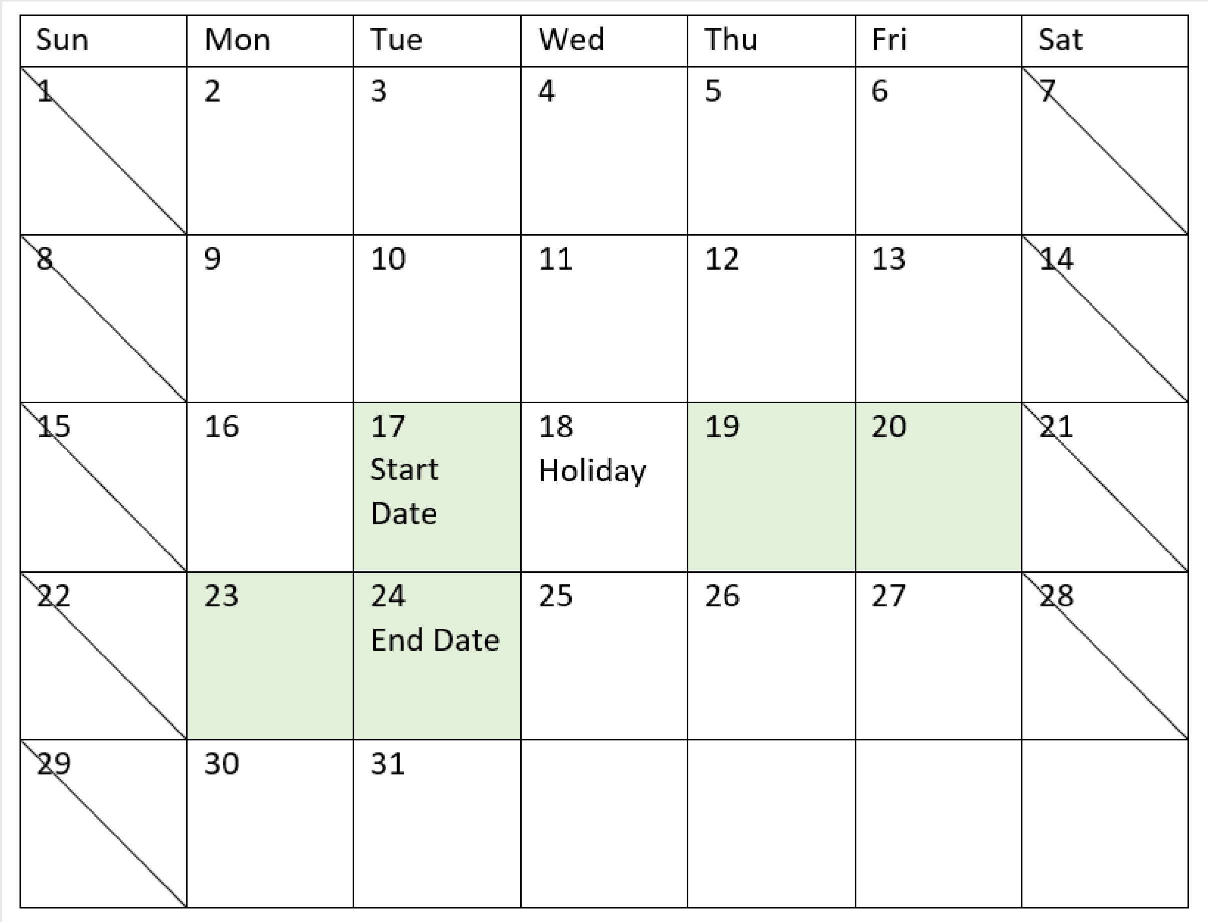
結果表格

id	start_date	花費天數	=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

輸入單一排程假期作為圖表中的量值。因此，專案 3 的結束日期會往後偏移一天，因為假期發生在結束日期之前的工作天之一。

下列行事曆顯示專案 3 的開始和結束日期，並顯示假期將專案的結束日期變更一天。

行事曆顯示專案 3 的開始和結束日期，其中假期為 5 月 18 日



localtime

此函數會傳回指定時區目前時間的時間戳記。

語法：

```
LocalTime([timezone [, ignoreDST ]])
```

傳回的資料類型: 雙值

引數

引數	描述
timezone	**timezone** 指定為字串，包含 **Windows Control Panel** 中為 **Date and Time** 列出的 **Time Zone** 下列出的任何地理位置，或者指定為 'GMT+hh:mm' 格式的字串。下表也呈現了接受的地點和時區清單。 如未指定時區，則會傳回當地時間。 *若您使用 DST 時差 (亦即您指定評估為 False 的 **ignoreDST** 引數值)，您必須在 **place** 引數中指定一個地方，而非 GMT 時差。這是因為對日光節約時間進行調整時，除了 GMT 時差提供的經度資訊，還需要緯度資訊。如需詳細資訊，請參閱結合 DST 使用 GMT 時差 (page 796)。*
ignoreDST	若此引數評估為 True，會忽略 DST (日光節約時間)。評估為 True 的有效引數值包括 `-1` 和 `True()`。 若此引數評估為 False，會為日光節約時間調整時間戳記。評估為 False 的有效引數值包括 `0` 和 `False()`。 若 **ignoreDST** 引數值無效，該函數會按照 **ignore_dst** 值評估為 True 的情境來評估運算式。若未指定 **ignoreDST** 引數值，該函數會按照 **ignore_dst** 值評估為 False 的情境來評估運算式。

有效的地點和時區

A-C	**D-K**	**L-R**	**S-Z**
Abu Dhabi	Darwin	La Paz	Samoa
Adelaide	Dhaka	Lima	Santiago
Alaska	Eastern Time (US & Canada)	Lisbon	Sapporo
Amsterdam	Edinburgh	Ljubljana	Sarajevo
Arizona	Ekaterinburg	London	Saskatchewan
Astana	Fiji	Madrid	Seoul
Athens	Georgetown	Magadan	Singapore
Atlantic Time (Canada)	Greenland	Mazatlan	Skopje
Auckland	Greenwich Mean Time : Dublin	Melbourne	Sofia
Azores	Guadalajara	Mexico City	Solomon Is.

A-C	D-K	L-R	S-Z
Baghdad	Guam	Mid-Atlantic	Sri Jayawardenepura
Baku	Hanoi	Minsk	St. Petersburg
Bangkok	Harare	Monrovia	Stockholm
Beijing	Hawaii	Monterrey	Sydney
Belgrade	Helsinki	Moscow	Taipei
Berlin	Hobart	Mountain Time (US & Canada)	Tallinn
Bern	Hong Kong	Mumbai	Tashkent
Bogota	Indiana (East)	Muscat	Tbilisi
Brasilia	International Date Line West	Nairobi	Tehran
Bratislava	Irkutsk	New Caledonia	Tokyo
Brisbane	Islamabad	New Delhi	Urumqi
Brussels	Istanbul	Newfoundland	Warsaw
Bucharest	Jakarta	Novosibirsk	Wellington
Budapest	Jerusalem	Nuku'alofa	West Central Africa
Buenos Aires	Kabul	Osaka	Vienna
Cairo	Kamchatka	Pacific Time (US & Canada)	Vilnius
Canberra	Karachi	Paris	Vladivostok
Cape Verde Is.	Kathmandu	Perth	Volgograd
Caracas	Kolkata	Port Moresby	Yakutsk
Casablanca	Krasnoyarsk	Prague	Yerevan
Central America	Kuala Lumpur	Pretoria	Zagreb
Central Time (US & Canada)	Kuwait	Quito	-
Chennai	Kyiv	Riga	-
Chihuahua	-	Riyadh	-
Chongqing	-	Rome	-
Copenhagen	-	-	-

範例與結果：

以下範例是根據在當地時間 2023-08-14 08:39:47 叫用的函數，伺服器或電腦版環境的當地時區為 GMT-05:00，並位於截至上述日期已實施日光節約時間的地區。

指令碼處理範例

範例	結果
`localtime ()`	傳回當地時間 2023-08-14 08:39:47。
`localtime ('London')`	傳回倫敦當地時間 2023-08-14 13:39:47。
`localtime ('GMT+02:00')`	傳回使用時區 GMT+02:00 的當地時間 2023-08-14 14:39:47。不會為日光節約時間進行任何調整，因為會指定 GMT 時差，而非某個地方。
`localtime ('Paris',-1)`	傳回巴黎當地時間 (忽略日光節約時間) 2023-08-14 13:39:47。
`localtime ('Paris',True())`	傳回巴黎當地時間 (忽略日光節約時間) 2023-08-14 13:39:47。
`localtime ('Paris',0)`	傳回巴黎當地時間 (考慮日光節約時間) 2023-08-14 14:39:47。
`localtime ('Paris',False ())`	傳回巴黎當地時間 (考慮日光節約時間) 2023-08-14 14:39:47。

結合 DST 使用 GMT 時差

按照 Qlik Sense 中國際統一碼元件 (ICU) 庫的實施，結合 DST (日光節約時間) 使用 GMT (格林威治標準時間) 時差需要附加的緯度資訊。

GMT 是經度 (東西向) 時差，而 DST 則是緯度 (南北向) 時差。例如，赫爾辛基 (芬蘭) 和約翰尼斯堡 (南非) 共用相同的 GMT+02:00 時差，但沒有共用相同的 DST 時差。這表示，除了 GMT 時差，任何 DST 時差都需要當地時區的緯度位置資訊 (地理時區輸入)，才能有完整的當地 DST 條件資訊。

lunarweekend

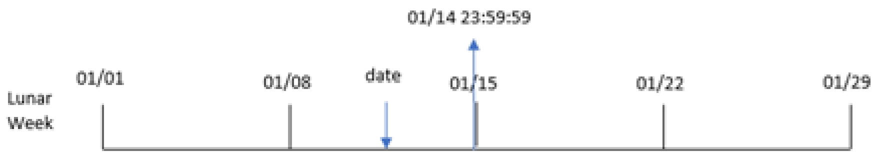
此函數傳回的值相當於包含 **date** 的該農曆週最後一天的最後一毫秒的時間戳記。將 1 月 1 日視為該週的第一天，以定義 Qlik Sense 中的農曆週，而除了該年的最後一週，會確切包含七天。

語法：

```
LunarweekEnd(date[, period_no[, first_week_day]])
```

傳回的資料類型：雙值

lunarweekend() 函數的範例圖表



lunarweekend() 函數判定 date 落在哪一個農曆週。然後以日期格式傳回該週最後一毫秒的時間戳記。

引數

引數	描述
date	要評估的時間戳記。
period_no	**period_no** 是一個整數，或者解析為整數的運算式，其中值 0 表示包含 **date** 的農曆週。負值的 **period_no** 表示之前的農曆週，正值表示之後的農曆週。
first_week_day	可能大於或小於零的位移。這會使用指定的天數及/或一天的片段變更一年的開始。

什麼情況下使用

lunarweekend() 函數通常在使用者想要計算以使用一週中尚未發生的部分時，作為運算式的一部分使用。與 weekend() 函數不同，每曆年的最終農曆週將結束於 12 月 31 日。例如，lunarweekend() 函數可用來計算該週期間尚未發生的利息。

函數範例

範例	結果
lunarweekend('01/12/2013')	傳回 01/14/2013 23:59:59。
lunarweekend('01/12/2013', -1)	傳回 01/07/2013 23:59:59。
lunarweekend('01/12/2013', 0, 1)	傳回 01/15/2013 23:59:59。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2022 年交易的資料集, 這載入到稱為 Transactions 的表格中。
- 以 DateFormat 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 end_of_week, 這傳回交易發生的該農曆週結束的時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- end_of_week
- end_of_week_timestamp

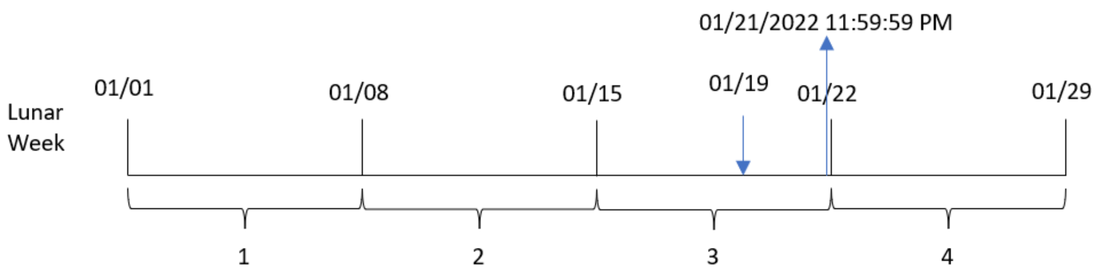
結果表格

日期	end_of_week	end_of_week_timestamp
1/7/2022	01/07/2022	1/7/2022 11:59:59 PM
1/19/2022	01/21/2022	1/21/2022 11:59:59 PM
2/5/2022	02/11/2022	2/11/2022 11:59:59 PM
2/28/2022	03/04/2022	3/4/2022 11:59:59 PM
3/16/2022	03/18/2022	3/18/2022 11:59:59 PM
4/1/2022	04/01/2022	4/1/2022 11:59:59 PM
5/7/2022	05/13/2022	5/13/2022 11:59:59 PM
5/16/2022	05/20/2022	5/20/2022 11:59:59 PM
6/15/2022	06/17/2022	6/17/2022 11:59:59 PM
6/26/2022	07/01/2022	7/1/2022 11:59:59 PM
7/9/2022	07/15/2022	7/15/2022 11:59:59 PM
7/22/2022	07/22/2022	7/22/2022 11:59:59 PM
7/23/2022	07/29/2022	7/29/2022 11:59:59 PM
7/27/2022	07/29/2022	7/29/2022 11:59:59 PM
8/2/2022	08/05/2022	8/5/2022 11:59:59 PM
8/8/2022	08/12/2022	8/12/2022 11:59:59 PM
8/19/2022	08/19/2022	8/19/2022 11:59:59 PM
9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
10/14/2022	10/14/2022	10/14/2022 11:59:59 PM
10/29/2022	11/04/2022	11/4/2022 11:59:59 PM

end_of_week 欄位在前置 LOAD 陳述式中的建立方式是使用 `lunarweekend()` 函數，並傳遞 `date` 欄位，作為函數的引數。

`lunarweekend()` 函數識別日期值落在哪一個農曆週，並傳回該週最後一毫秒的時間戳記。

lunarweekend() 函數的圖表, 無其他引數的範例



交易 8189 發生在 1 月 19 日。lunarweekend() 函數識別該農曆週開始於 1 月 15 日。因此, 該交易的 end_of_week 值傳回該農曆週的最後一毫秒, 亦即 1 月 21 日下午 11:59:59。

範例 2 – period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 與第一個範例相同的資料集和情境。
- 建立欄位 previous_lunar_week_end, 這傳回交易發生的前一個農曆週結束的時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- `date`
- `previous_lunar_week_end`
- `previous_lunar_week_end_timestamp`

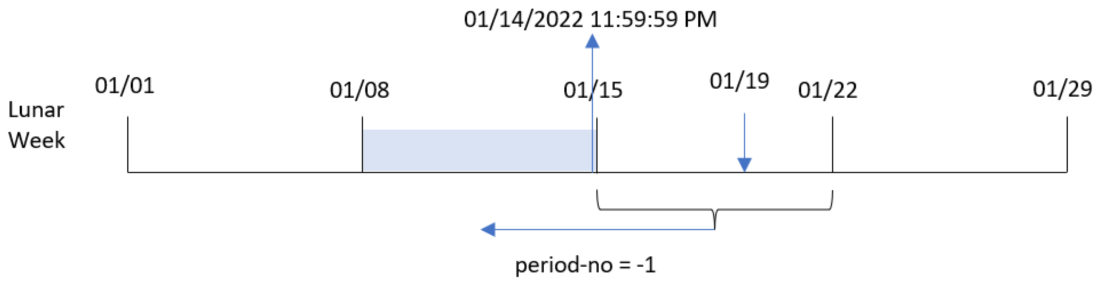
結果表格

日期	previous_lunar_week_end	previous_lunar_week_end_timestamp
1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
1/19/2022	01/14/2022	1/14/2022 11:59:59 PM
2/5/2022	02/04/2022	2/4/2022 11:59:59 PM
2/28/2022	02/25/2022	2/25/2022 11:59:59 PM
3/16/2022	03/11/2022	3/18/2022 11:59:59 PM
4/1/2022	03/25/2022	3/25/2022 11:59:59 PM
5/7/2022	05/06/2022	5/6/2022 11:59:59 PM
5/16/2022	05/13/2022	5/13/2022 11:59:59 PM
6/15/2022	06/10/2022	6/10/2022 11:59:59 PM
6/26/2022	06/24/2022	6/24/2022 11:59:59 PM
7/9/2022	07/08/2022	7/8/2022 11:59:59 PM
7/22/2022	07/15/2022	7/15/2022 11:59:59 PM
7/23/2022	07/22/2022	7/22/2022 11:59:59 PM
7/27/2022	07/22/2022	7/22/2022 11:59:59 PM
8/2/2022	07/29/2022	7/29/2022 11:59:59 PM
8/8/2022	08/05/2022	8/5/2022 11:59:59 PM
8/19/2022	08/12/2022	8/12/2022 11:59:59 PM

日期	previous_lunar_week_end	previous_lunar_week_end_timestamp
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

在此例中，因為 period_no 的值 -1 已作為 lunarweekend() 函數中的偏移引數使用，所以該函數首先會識別交易發生的農曆週。然後這會往前偏移一週並識別該農曆週的最後一毫秒。

lunarweekend() 函數的圖表，period_no 範例



交易 8189 發生在 1 月 19 日。lunarweekend() 函數識別該農曆週開始於 1 月 15 日。因此，前一農曆週開始於 1 月 8 日並結束於 1 月 14 日 11:59:59 PM；這是為 previous_lunar_week_end 欄位傳回的值。

範例 3 – first_week_day

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。在此範例中，我們將農曆週設定為在 1 月 5 月開始。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- end_of_week
- end_of_week_timestamp

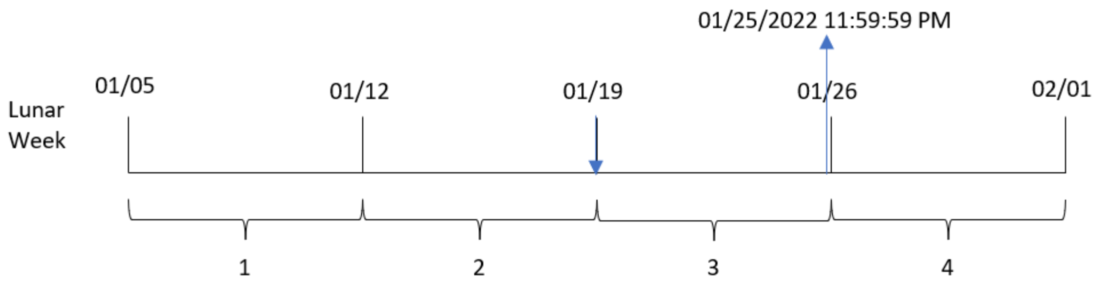
結果表格

日期	end_of_week	end_of_week_timestamp
1/7/2022	01/11/2022	1/11/2022 11:59:59 PM
1/19/2022	01/25/2022	1/25/2022 11:59:59 PM
2/5/2022	02/08/2022	2/8/2022 11:59:59 PM
2/28/2022	03/01/2022	3/1/2022 11:59:59 PM
3/16/2022	03/22/2022	3/22/2022 11:59:59 PM
4/1/2022	04/05/2022	4/5/2022 11:59:59 PM
5/7/2022	05/10/2022	5/10/2022 11:59:59 PM
5/16/2022	05/17/2022	5/17/2022 11:59:59 PM
6/15/2022	06/21/2022	6/21/2022 11:59:59 PM
6/26/2022	06/28/2022	6/28/2022 11:59:59 PM
7/9/2022	07/12/2022	7/12/2022 11:59:59 PM

日期	end_of_week	end_of_week_timestamp
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

在此例中，因為 first_week_date 引數 4 用於 lunarweekend() 函數，這會將該年的開始從 1 月 1 日偏移至 1 月 5 日。

lunarweekend() 函數的圖表，first_week_day 範例



交易 8189 發生在 1 月 19 日。由於農曆週於 1 月 5 日開始，lunarweekend() 函數識別包含 1 月 19 日的農曆週也在 1 月 19 日開始。因此，該農曆週結束於 1 月 25 日 11:59:59 PM；這是為 end_of_week 欄位傳回的值。

範例 4 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。

不過，在此範例中，不變的資料集會載入到應用程式中。會建立傳回交易發生農曆週結束時間戳記的計算，作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`date`。

新增下列量值：

`=lunarweekend(date)`

`=timestamp(lunarweekend(date))`

結果表格

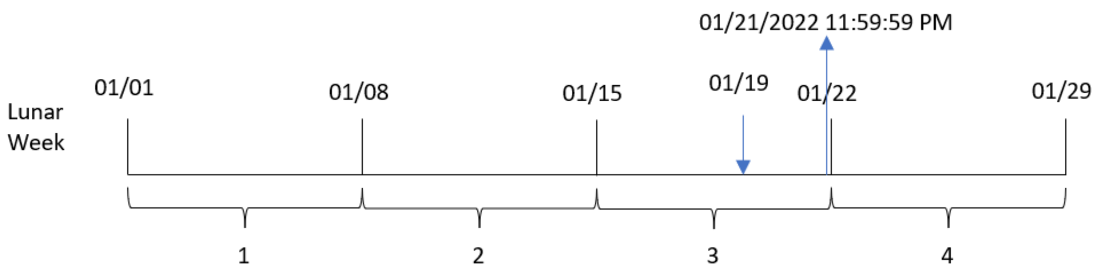
日期	=lunarweekend(date)	=timestamp(lunarweekend(date))
1/7/2022	01/07/2022	1/7/2022 11:59:59 PM
1/19/2022	01/21/2022	1/21/2022 11:59:59 PM
2/5/2022	02/11/2022	2/11/2022 11:59:59 PM
2/28/2022	03/04/2022	3/4/2022 11:59:59 PM
3/16/2022	03/18/2022	3/18/2022 11:59:59 PM
4/1/2022	04/01/2022	4/1/2022 11:59:59 PM

日期	=lunarweekend(date)	=timestamp(lunarweekend(date))
5/7/2022	05/13/2022	5/13/2022 11:59:59 PM
5/16/2022	05/20/2022	5/20/2022 11:59:59 PM
6/15/2022	06/17/2022	6/17/2022 11:59:59 PM
6/26/2022	07/01/2022	7/1/2022 11:59:59 PM
7/9/2022	07/15/2022	7/15/2022 11:59:59 PM
7/22/2022	07/22/2022	7/22/2022 11:59:59 PM
7/23/2022	07/29/2022	7/29/2022 11:59:59 PM
7/27/2022	07/29/2022	7/29/2022 11:59:59 PM
8/2/2022	08/05/2022	8/5/2022 11:59:59 PM
8/8/2022	08/12/2022	8/12/2022 11:59:59 PM
8/19/2022	08/19/2022	8/19/2022 11:59:59 PM
9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
10/14/2022	10/14/2022	10/14/2022 11:59:59 PM
10/29/2022	11/04/2022	11/4/2022 11:59:59 PM

end_of_week 量值在圖表物件中的建立方式是使用 `lunarweekend()` 函數並傳遞 date 欄位，作為函數的引數。

`lunarweekend()` 函數識別日期值落在哪一個農曆週，並傳回該週最後一毫秒的時間戳記。

`lunarweekend()` 函數的圖表，圖表物件範例



交易 8189 發生在 1 月 19 日。`lunarweekend()` 函數識別該農曆週開始於 1 月 15 日。因此，該交易的 end_of_week 值傳回該農曆週的最後一毫秒，亦即 1 月 21 日下午 11:59:59。

範例 5 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入到稱為 `Employee_Expenses` 之表格的資料集。
- 員工 ID、員工姓名及每個員工報銷的平均每日開支。

最終使用者希望圖表物件按員工 ID 和員工名稱顯示該農曆週剩餘期間仍會產生的預估開支報銷。

載入指令碼

```
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
];
```

結果

請執行下列動作：

1. 載入資料並開啟工作表。建立新的表格。
2. 新增下列欄位作為維度。
   - `employee_id`
   - `employee_name`
3. 接著，建立下列量值以計算累積的利息：
   `=(lunarweekend(today(1))-today(1))*avg_daily_claim`
4. 將量值的**數字格式**設定為**金錢**。

結果表格

employee_id	employee_name	=(lunarweekend(today(1))-today(1))*avg_daily_claim
182	Mark	$75.00

employee_id	employee_name	=(lunarweekend(today(1))-today(1))*avg_daily_claim
183	Deryck	$62.50
184	Dexter	$62.50
185	Sydney	$135.00
186	Agatha	$90.00

藉由使用今日日期作為其唯一函數，`lunarkweekend()` 函數會傳回目前農曆週的結束日期。然後，以該農曆週結束日期減去今日日期，運算式就會傳回該週剩餘天數。

然後按員工讓此值乘以平均每日開支報銷，以計算每個員工在該農曆週剩餘期間預期會有的預估報銷值。

lunarweekname

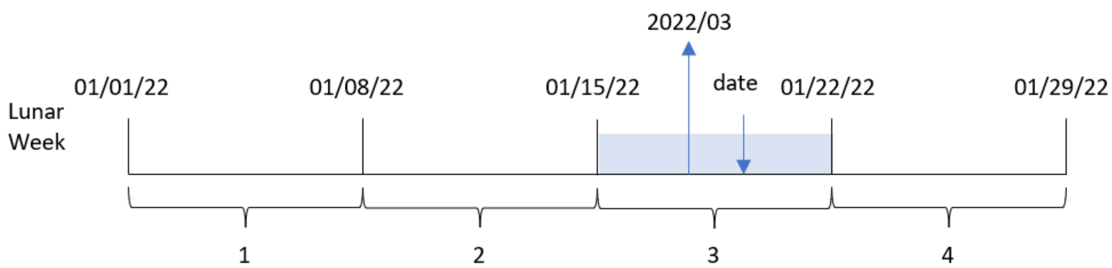
此函數會傳回一個顯示值，顯示相當於包含 **date** 之農曆週第一天的第一毫秒的時間戳記的年和農曆週數。將 1 月 1 日視為該週的第一天，以定義 Qlik Sense 中的農曆週，而除了該年的最後一週，會確切包含七天。

語法：

```
LunarWeekName(date [, period_no[, first_week_day]])
```

傳回的資料類型：雙值

`lunarweekname()` 函數的範例圖表



`lunarweekname()` 函數判定日期落在哪個農曆週，週計數開始於 1 月 1 日。然後這會傳回由 `year/weekcount` 組成的值。

引數

引數	描述
date	要評估的時間戳記。
period_no	**period_no** 是一個整數，或者解析為整數的運算式，其中值 0 表示包含 **date** 的農曆週。負值的 **period_no** 表示之前的農曆週，正值表示之後的農曆週。
first_week_day	可能大於或小於零的位移。這會使用指定的天數及/或一天的片段變更一年的開始。

什麼情況下使用

若您想要按農曆週比較彙總，則 lunarweekname() 函數很實用。例如，該函數可用來按農曆週決定產品的總銷售額。若您想要確保該年第一週包含的所有值僅包含最早從 1 月 1 日的值，則農曆週很實用。

可以使用該函數建立主要行事曆表格中的欄位，以在載入指令碼中建立這些維度。也可以直接在圖表中使用該函數作為計算維度。

函數範例

範例	結果
lunarweekname('01/12/2013')	傳回 2006/02。
lunarweekname('01/12/2013', -1)	傳回 2006/01。
lunarweekname('01/12/2013', 0, 1)	傳回 2006/02。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 無其他引數的日期

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為 Transactions 的表格中。
- 以 DateFormat 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 lunar_week_name，傳回交易發生之農曆週的年份和週數。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- lunar_week_name

結果表格

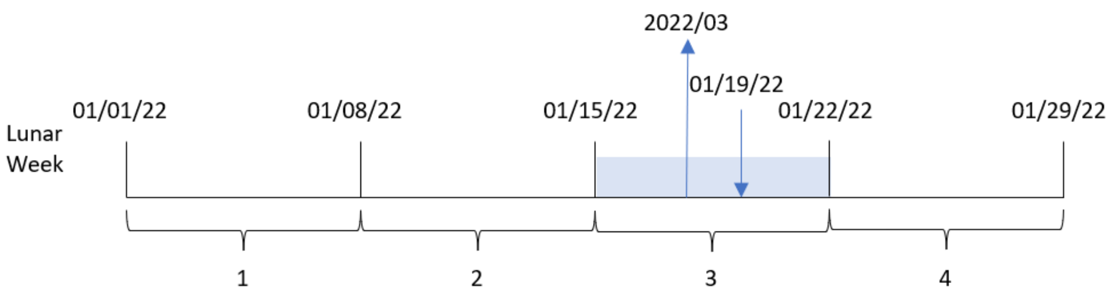
日期	lunar_week_name
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/06
2/28/2022	2022/09
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/19

日期	lunar_week_name
5/16/2022	2022/20
6/15/2022	2022/24
6/26/2022	2022/26
7/9/2022	2022/28
7/22/2022	2022/29
7/23/2022	2022/30
7/27/2022	2022/30
8/2/2022	2022/31
8/8/2022	2022/32
8/19/2022	2022/33
9/26/2022	2022/39
10/14/2022	2022/41
10/29/2022	2022/44

`lunar_week_name` 欄位在前置 LOAD 陳述式中的建立方式是使用 `lunarweekname()` 函數，並傳遞 `date` 欄位，作為函數的引數。

`lunarweekname()` 函數識別日期值落在哪一個農曆週，傳回該日期的年份和週數。

lunarweekname() 函數的圖表，無其他引數的範例



交易 8189 發生在 1 月 19 日。`lunarweekname()` 函數識別此日期落在開始於 1 月 15 日的農曆週，這是該年的第三個農曆週。因此，為該交易傳回的 `lunar_week_name` 值是 2022/03。

範 2 – 有 period_no 引數的日期

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 與第一個範例相同的資料集和情境。
- 建立欄位 previous_lunar_week_name, 傳回交易發生前農曆週的年份和週數。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

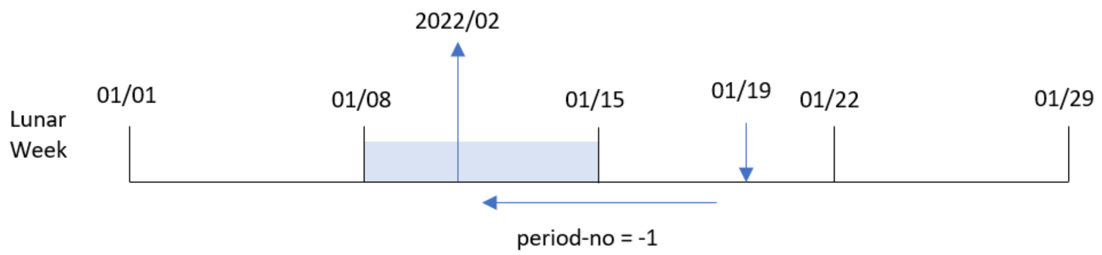
- date
- previous_lunar_week_name

結果表格

日期	previous_lunar_week_name
1/7/2022	2021/52
1/19/2022	2022/02
2/5/2022	2022/05
2/28/2022	2022/08
3/16/2022	2022/10
4/1/2022	2022/12
5/7/2022	2022/18
5/16/2022	2022/19
6/15/2022	2022/23
6/26/2022	2022/25
7/9/2022	2022/27
7/22/2022	2022/28
7/23/2022	2022/29
7/27/2022	2022/29
8/2/2022	2022/30
8/8/2022	2022/31
8/19/2022	2022/32
9/26/2022	2022/38
10/14/2022	2022/40
10/29/2022	2022/43

在此例中，因為 `period_no` 的值 `-1` 已作為 `lunarweekname()` 函數中的偏移引數使用，所以該函數首先會識別交易發生的農曆週。然後這會傳回年份和前一週的週數。

`lunarweekname()` 函數的圖表，period_no 範例



交易 8189 發生在 1 月 19 日。lunarweekname() 函數識別此交易發生在該年的第三個農曆週，因此為 previous_lunar_week_name 欄位傳回前一週的年份和值 2022/02。

範例 3 – 有 first_week_day 引數的日期

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。在此範例中，我們將農曆週設定為在 1 月 5 月開始。

載入指令碼

```
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
];
```

結果

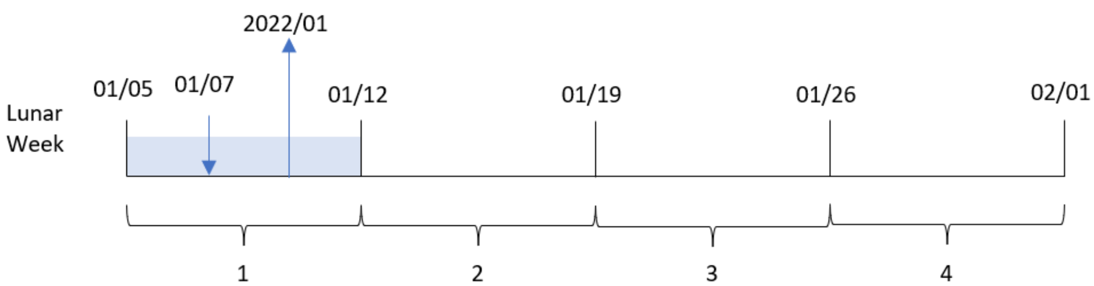
載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- `date`
- `lunar_week_name`

結果表格

日期	lunar_week_name
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/05
2/28/2022	2022/08
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/18
5/16/2022	2022/19
6/15/2022	2022/24
6/26/2022	2022/25
7/9/2022	2022/27
7/22/2022	2022/29
7/23/2022	2022/29
7/27/2022	2022/30
8/2/2022	2022/30
8/8/2022	2022/31
8/19/2022	2022/33
9/26/2022	2022/38
10/14/2022	2022/41
10/29/2022	2022/43

`lunarweekname()` 函數的圖表，first_week_day 範例



在此例中，因為 `first_week_date` 引數 4 用於 `lunarweekname()` 函數，這會將該農曆週的開始從 1 月 1 日偏移至 1 月 5 日。

交易 8188 發生在 1 月 7 日。由於農曆週開始於 1 月 5 日，`lunarweekname()` 函數識別包含 1 月 7 日的農曆週是該年的第一個農曆週。因此，為該交易傳回的 `lunar_week_name` 值是 2022/01。

範例 4 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。

不過，在此範例中，不變的資料集會載入到應用程式中。會建立傳回交易發生之農曆週數和年份的計算，作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`date`。

若要計算交易發生的該農曆週開始日期，建立下列量值：

`=lunarweekname(date)`

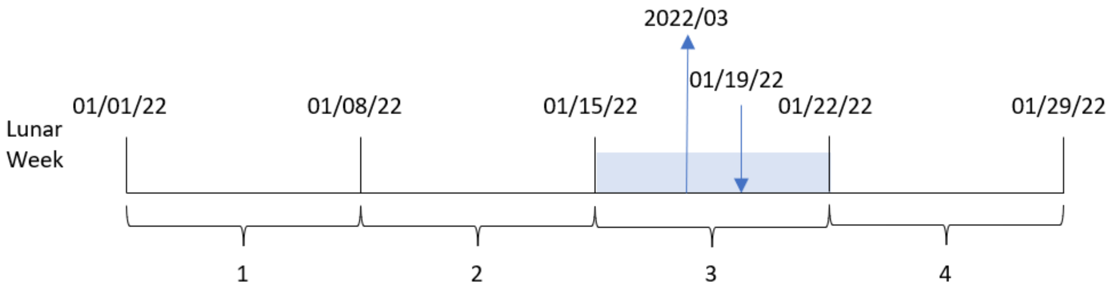
結果表格

日期	=lunarweekname(date)
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/06
2/28/2022	2022/09
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/19
5/16/2022	2022/20
6/15/2022	2022/24
6/26/2022	2022/26
7/9/2022	2022/28
7/22/2022	2022/29
7/23/2022	2022/30
7/27/2022	2022/30
8/2/2022	2022/31
8/8/2022	2022/32
8/19/2022	2022/33
9/26/2022	2022/39
10/14/2022	2022/41
10/29/2022	2022/44

`lunar_week_name` 量值在圖表物件中的建立方式是使用 `lunarweekname()` 函數並傳遞 `date` 欄位，作為函數的引數。

`lunarweekname()` 函數識別日期值落在哪一個農曆週，傳回該日期的年份和週數。

lunarweekname() 函數的圖表，圖表物件範例



交易 8189 發生在 1 月 19 日。lunarweekname() 函數識別此日期落在開始於 1 月 15 日的農曆週，這是該年的第三個農曆週。因此，該交易的 lunar_week_name 值是 2022/03。

範例 5 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為 Transactions 的表格中。
- 以 DateFormat 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。

最終使用者希望圖表物件按目前年份的週呈現總銷售額。長度七天的第 1 週應於 1 月 1 日開始。當資料模型中無法使用此維度時，在圖表中將 lunarweekname() 函數作為計算維度可達成這一點。

載入指令碼

```
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
];
```

結果

請執行下列動作:

1. 載入資料並開啟工作表。建立新的表格。
2. 使用下列運算式建立計算維度:
   `=lunarweekname(date)`
3. 使用下列彙總量值計算總銷售額:
   `=sum(amount)`
4. 將量值的**數字格式**設定為**金錢**。

結果表格

=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

=lunarweekname(date)	=sum(amount)
2022/39	$84.21
2022/41	$96.24
2022/44	$67.67

lunarweekstart

此函數傳回的值相當於包含 **date** 的該農曆週第一天的第一毫秒的時間戳記。將 1 月 1 日視為該週的第一天，以定義 Qlik Sense 中的農曆週，而除了該年的最後一週，會確切包含七天。

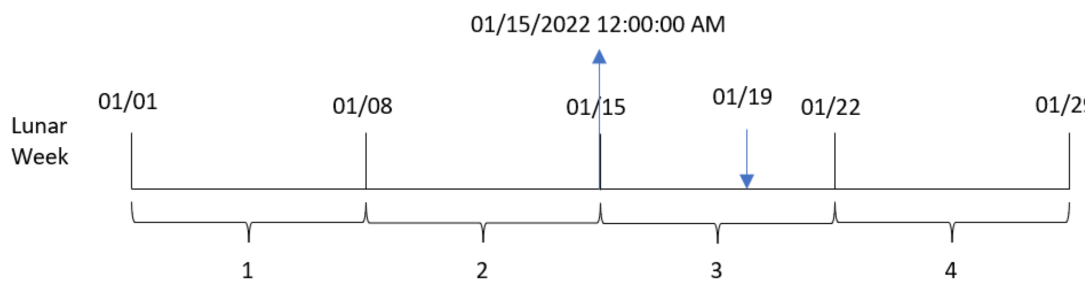
語法：

```
LunarweekStart(date[, period_no[, first_week_day]])
```

傳回的資料類型： 雙值

`lunarweekstart()` 函數判定 `date` 落在哪一個農曆週。然後以日期格式傳回該週第一毫秒的時間戳記。

`lunarweekstart()` 函數的範例圖表



引數

引數	描述
date	要評估的時間戳記。
period_no	**period_no** 是一個整數，或者解析為整數的運算式，其中值 0 表示包含 **date** 的農曆週。負值的 **period_no** 表示之前的農曆週，正值表示之後的農曆週。
first_week_day	可能大於或小於零的位移。這會使用指定的天數及/或一天的片段變更一年的開始。

什麼情況下使用

`lunarweekstart()` 函數通常在使用者想要計算使用一週中已經過的部分時，作為運算式的一部分使用。與 `weekstart()` 函數不同，在每個新曆年的開始，週開始於 1 月 1 日，之後每週都在七天後開始。`FirstWeekDay` 系統變數不會影響 `lunarweekstart()` 函數。

例如，`lunarweekstart()` 可用於計算一週至今彙總的利息。

函數範例

範例	結果
lunarweekstart('01/12/2013')	傳回 01/08/2013。
lunarweekstart('01/12/2013', -1)	傳回 01/01/2013。
lunarweekstart('01/12/2013', 0, 1)	傳回 01/09/2013, 因為將 first_week_day 設為 1 表示年的開始變更為 01/02/2013。

區域設定

除非另有說明, 否則此主題中的範例皆使用下列日期格式:MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素, 您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式, 以滿足您的需求。或者, 您可以在載入指令碼中變更格式, 以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典, 資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2022 年交易的資料集, 這載入到稱為 Transactions 的表格中。
- 以 DateFormat 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 start_of_week, 這傳回交易發生的該農曆週開始的時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- start_of_week
- start_of_week_timestamp

結果表格

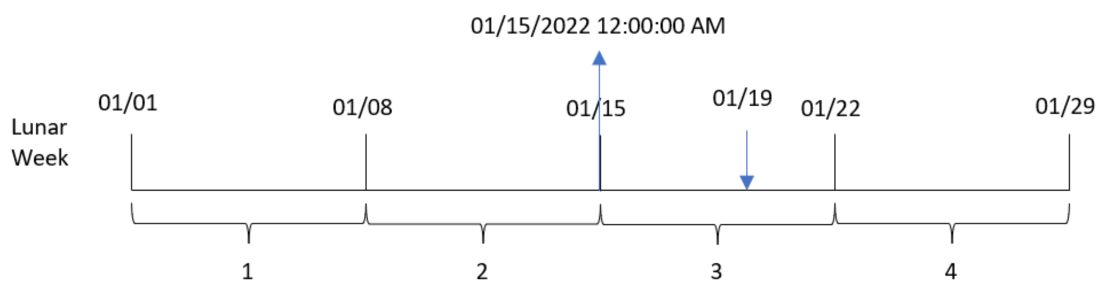
日期	start_of_week	start_of_week_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/15/2022	1/15/2022 12:00:00 AM
2/5/2022	02/05/2022	2/5/2022 12:00:00 AM
2/28/2022	02/26/2022	2/26/2022 12:00:00 AM
3/16/2022	03/12/2022	3/12/2022 12:00:00 AM
4/1/2022	03/26/2022	3/26/2022 12:00:00 AM
5/7/2022	05/07/2022	5/7/2022 12:00:00 AM
5/16/2022	05/14/2022	5/14/2022 12:00:00 AM
6/15/2022	06/11/2022	6/11/2022 12:00:00 AM
6/26/2022	06/25/2022	6/25/2022 12:00:00 AM
7/9/2022	07/09/2022	7/9/2022 12:00:00 AM

日期	start_of_week	start_of_week_timestamp
7/22/2022	07/16/2022	7/16/2022 12:00:00 AM
7/23/2022	07/23/2022	7/23/2022 12:00:00 AM
7/27/2022	07/23/2022	7/23/2022 12:00:00 AM
8/2/2022	07/30/2022	7/30/2022 12:00:00 AM
8/8/2022	08/06/2022	8/6/2022 12:00:00 AM
8/19/2022	08/13/2022	8/13/2022 12:00:00 AM
9/26/2022	09/24/2022	9/24/2022 12:00:00 AM
10/14/2022	10/08/2022	10/8/2022 12:00:00 AM
10/29/2022	10/29/2022	10/29/2022 12:00:00 AM

start_of_week 欄位在前置 LOAD 陳述式中的建立方式是使用 lunarweekstart() 函數，並傳遞 date 欄位，作為函數的引數。

lunarweekstart() 函數識別日期落在哪一個農曆週，並傳回該週第一毫秒的時間戳記。

lunarweekstart() 函數的圖表，無其他引數的範例



交易 8189 發生在 1 月 19 日。lunarweekstart() 函數識別該農曆週開始於 1 月 15 日。因此，該交易的 start_of_week 值傳回該日子的第一毫秒，亦即 1 月 15 日上午 12:00:00。

範例 2 – period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 建立欄位 previous_lunar_week_start，這傳回交易發生的前一個農曆週開始的時間戳記。

載入指令碼

```
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
];
```

結果

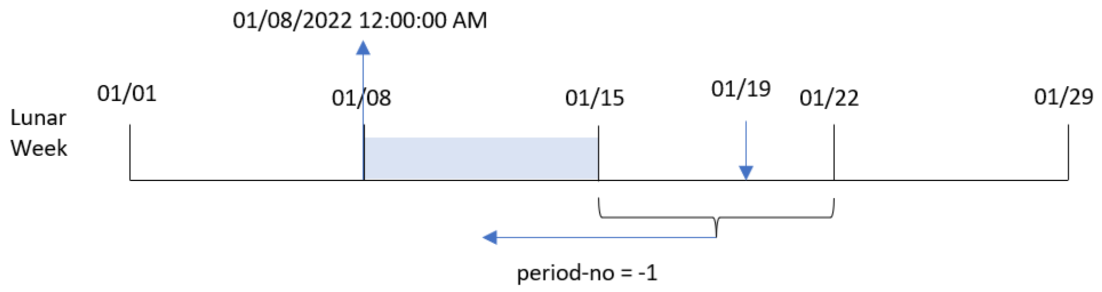
結果表格

日期	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

日期	previous_lunar_week_start	previous_lunar_week_start_timestamp
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

在此例中，因為 period_no 的值 -1 已作為 lunarweekstart() 函數中的偏移引數使用，所以該函數首先會識別交易發生的農曆週。然後這會往前偏移一週並識別該農曆週的第一毫秒。

lunarweekstart() 函數的圖表，period_no 範例



交易 8189 發生在 1 月 19 日。lunarweekstart() 函數識別該農曆週開始於 1 月 15 日。因此，前一農曆週開始於 1 月 8 日 12:00:00 AM；這是為 previous_lunar_week_start 欄位傳回的值。

範例 3 – first_week_day

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。在此範例中，我們將農曆週設定為在 1 月 5 月開始。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

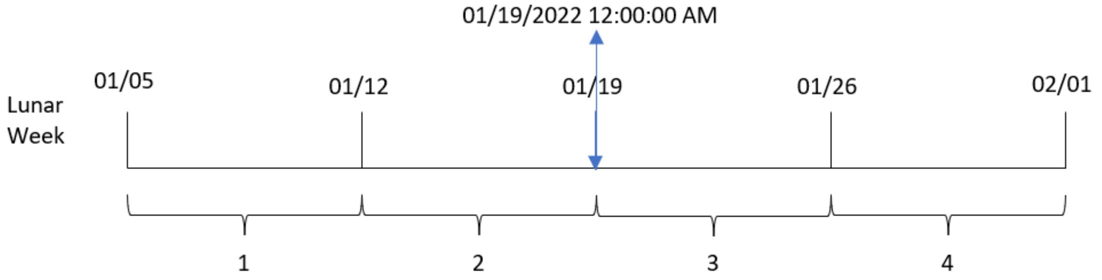
- date
- start_of_week
- start_of_week_timestamp

結果表格

日期	start_of_week	start_of_week_timestamp
1/7/2022	01/05/2022	1/5/2022 12:00:00 AM
1/19/2022	01/19/2022	1/19/2022 12:00:00 AM
2/5/2022	02/02/2022	2/2/2022 12:00:00 AM
2/28/2022	02/23/2022	2/23/2022 12:00:00 AM
3/16/2022	03/16/2022	3/16/2022 12:00:00 AM
4/1/2022	03/30/2022	3/30/2022 12:00:00 AM
5/7/2022	05/04/2022	5/4/2022 12:00:00 AM
5/16/2022	05/11/2022	5/11/2022 12:00:00 AM
6/15/2022	06/15/2022	6/15/2022 12:00:00 AM
6/26/2022	06/22/2022	6/22/2022 12:00:00 AM
7/9/2022	07/06/2022	7/6/2022 12:00:00 AM
7/22/2022	07/20/2022	7/20/2022 12:00:00 AM
7/23/2022	07/20/2022	7/20/2022 12:00:00 AM
7/27/2022	07/27/2022	7/27/2022 12:00:00 AM
8/2/2022	07/27/2022	7/27/2022 12:00:00 AM
8/8/2022	08/03/2022	8/3/2022 12:00:00 AM
8/19/2022	08/17/2022	8/17/2022 12:00:00 AM
9/26/2022	09/21/2022	9/21/2022 12:00:00 AM
10/14/2022	10/12/2022	10/12/2022 12:00:00 AM
10/29/2022	10/26/2022	10/26/2022 12:00:00 AM

在此例中，因為 `first_week_date` 引數 4 用於 `lunarweekstart()` 函數，這會將該年的開始從 1 月 1 日偏移至 1 月 5 日。

`lunarweekstart()` 函數的圖表，first_week_day 範例



交易 8189 發生在 1 月 19 日。由於農曆週於 1 月 5 日開始，`lunarweekstart()` 函數識別包含 1 月 19 日的農曆週也在 1 月 19 日 12:00:00 AM 開始。因此，這是為 `start_of_week` 欄位傳回的值。

範例 4 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。

不過，在此範例中，不變的資料集會載入到應用程式中。會建立傳回交易發生農曆週開始時間戳記的計算，作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`date`。

新增下列量值：

```
=lunarweekstart(date)
```

```
=timestamp(lunarweekstart(date))
```

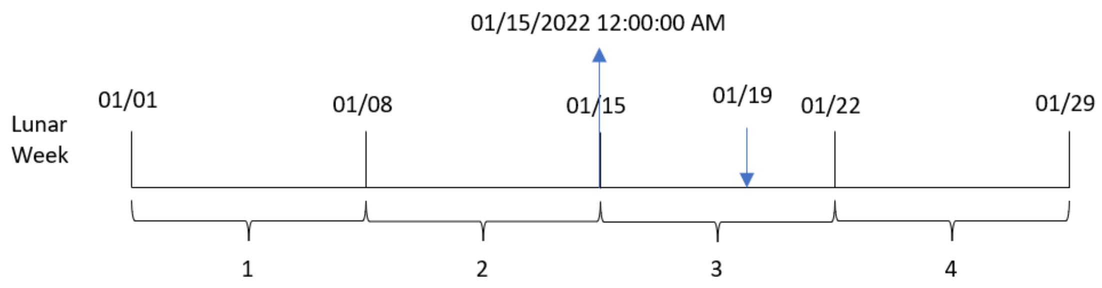
結果表格

日期	=lunarweekstart(date)	=timestamp(lunarweekstart(date))
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/15/2022	1/15/2022 12:00:00 AM
2/5/2022	02/05/2022	2/5/2022 12:00:00 AM
2/28/2022	02/26/2022	2/26/2022 12:00:00 AM
3/16/2022	03/12/2022	3/12/2022 12:00:00 AM
4/1/2022	03/26/2022	3/26/2022 12:00:00 AM
5/7/2022	05/07/2022	5/7/2022 12:00:00 AM
5/16/2022	05/14/2022	5/14/2022 12:00:00 AM
6/15/2022	06/11/2022	6/11/2022 12:00:00 AM
6/26/2022	06/25/2022	6/25/2022 12:00:00 AM
7/9/2022	07/09/2022	7/9/2022 12:00:00 AM
7/22/2022	07/16/2022	7/16/2022 12:00:00 AM
7/23/2022	07/23/2022	7/23/2022 12:00:00 AM
7/27/2022	07/23/2022	7/23/2022 12:00:00 AM
8/2/2022	07/30/2022	7/30/2022 12:00:00 AM
8/8/2022	08/06/2022	8/6/2022 12:00:00 AM
8/19/2022	08/13/2022	8/13/2022 12:00:00 AM
9/26/2022	09/24/2022	9/24/2022 12:00:00 AM
10/14/2022	10/08/2022	10/8/2022 12:00:00 AM
10/29/2022	10/29/2022	10/29/2022 12:00:00 AM

「start_of_week」量值在圖表物件中的建立方式是使用 lunarweekstart() 函數並傳遞日期欄位，作為函數的引數。

lunarweekstart() 函數識別日期值落在哪一個農曆週，並傳回該週最後一毫秒的時間戳記。

lunarweekstart() 函數的圖表，圖表物件範例



交易 8189 發生在 1 月 19 日。lunarweekstart() 函數識別該農曆週開始於 1 月 15 日。因此，該交易的 start_of_week 值是該日子的第一毫秒，亦即 1 月 15 日上午 12:00:00。

範例 5 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 資料集包含一系列貸款餘額，其位於稱為 Loans 的表格。
- 資料包含貸款 ID、週開始餘額以及每年對每項貸款收取的單利率。

最終使用者希望圖表物件按貸款 ID 顯示週初至今每筆貸款已累積的目前利息。

載入指令碼

```
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
];
```

結果

請執行下列動作：

1. 載入資料並開啟工作表。建立新的表格。
2. 新增下列欄位作為維度。
   - `loan_id`
   - `start_balance`
3. 接著，建立下列量值以計算累積的利息：
   `=start_balance*(rate*(today(1)-lunarweekstart(today(1)))/365)`
4. 將量值的**數字格式**設定為**金錢**。

結果表格

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

`lunarweekstart()` 函數使用今日日期作為其唯一引數，傳回目前年份的開始日期。以目前日期減去該結果後，運算式會傳回本週目前已經過的天數。

然後此值乘以利率並除以 365，以傳回此期間產生的有效利率。然後該結果乘以貸款的開始餘額，以傳回本週目前已累積的利息。

makedate

此函數會傳回從年 **YYYY**、月 **MM** 及日 **DD** 計算而得的日期。

語法：

```
MakeDate(YYYY [ , MM [ , DD ] ])
```

傳回的資料類型： 雙值

引數

引數	描述
YYYY	年度為整數。
MM	月份為整數。如未指定月份，則會採用 1 (1 月)。
DD	日為整數。如未指定日，則會採用 1 (1 號)。

什麼情況下使用

通常會在用於資料產生的指令碼中使用 makedate() 函數，以產生行事曆。若日期欄位無法直接作為日期使用，而需要一些轉換以擷取年份、月份和日期元件，也可以使用此。

這些範例使用日期格式 MM/DD/YYYY。日期格式是在位於資料載入指令碼頂端的 SET DateFormat 陳述式中指定。變更範例中的格式以配合您的需求。

函數範例

範例	結果
makedate(2012)	傳回 01/01/2012。
makedate(12)	傳回 01/01/2012。
makedate(2012,12)	傳回 12/01/2012。
makedate(2012,2,14)	傳回 02/14/2012。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 基本範例

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2018 年交易的資料集，這載入到稱為 Transactions 的表格中。
- 以 DateFormat 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 transaction_date，這以格式 MM/DD/YYYY 傳回日期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- transaction_year
- transaction_month
- transaction_day
- transaction_date

結果表格

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

transaction_date 欄位在前置 LOAD 陳述式中的建立方式是使用 makedate() 函數並傳遞年份、月份、日期欄位，作為函數引數。

然後該函數會將這些值合併並轉換為日期欄位，以 DateFormat 系統變數格式傳回結果。

範例 2 – 修改的 DateFormat

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 以格式 DD/MM/YYYY 建立欄位 `transaction_date` 而不修改 `DateFormat` 系統變數。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- `transaction_year`
- `transaction_month`
- `transaction_day`
- `transaction_date`

結果表格

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

在此例中, `makedate()` 函數巢狀於 `date()` 函數內部。`date()` 函數的第二引數將 `makedate()` 函數結果的格式設為所需的 DD/MM/YYYY。

範例 3 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2018 年交易的資料集,這載入到稱為 `Transactions` 的表格中。
- 跨越兩個欄位提供的交易日期:`year` 和 `month`。

建立圖表物件量值 `transaction_date`,這以格式 MM/DD/YYYY 傳回日期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- `year`
- `month`

若要決定 `transaction_date`,建立此量值:

```
=makedate(transaction_year,transaction_month)
```

結果表格

transaction_year	transaction_month	transaction_date
2018	08	08/01/2018
2018	09	09/01/2018

transaction_date 量值在圖表物件中的建立方式是使用 makedate() 函數並傳遞年份和月份欄位，作為函數引數。

然後該函數會合併這些值，以及假設日期值 01。然後這些值會轉換為日期欄位，以 DateFormat 系統變數格式傳回結果。

範例 4 – 情境

載入指令碼和圖表運算式

概覽

建立 2022 曆年的行事曆資料集。

載入指令碼

```
SET DateFormat='MM/DD/YYYY';

Calendar:
       load
               *
    where year(date)=2022;
load
       date(recno()+makedate(2021,12,31)) as date
AutoGenerate 400;
```

結果

結果表格

日期
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

日期
01/13/2022
01/14/2022
01/15/2022
01/16/2022
01/17/2022
01/18/2022
01/19/2022
01/20/2022
01/21/2022
01/22/2022
01/23/2022
01/24/2022
01/25/2022
還有 + 340 列

`makedate()` 函數建立 2021 年 12 月 31 日的日期值。`recno()` 函數提供載入到表格之目前記錄的記錄數，從 1 開始。因此，第一個記錄具有日期 2022 年 1 月 1 日。接續的每個 `recno()` 會由此日期以 1 為增量開始增加。此運算式包覆在 `date()` 函數中，用來將值轉換為日期。此流程依 `autogenerate` 函數重複 400 次。最終，透過使用前置載入，`where` 條件可用來僅載入來自 2022 年的日期。此指令碼產生包含 2022 年每個日期的行事曆。

maketime

此函數會傳回從小時 **hh**、分鐘 **mm** 及秒 **ss** 計算而得的時間。

語法：

```
MakeTime(hh [ , mm [ , ss ] ])
```

傳回的資料類型： 雙值

引數

引數	描述
hh	小時為整數。
mm	分鐘為整數。 如未指定分鐘，則會採用 00。
ss	秒為整數。 如未指定秒，則會採用 00。

什麼情況下使用

通常會在用於資料產生的指令碼中使用 `maketime()` 函數,以產生時間欄位。有時候,從輸入文字衍生時間欄位時,此函數可用來建構使用其元件的時間。

這些範例使用時間格式 `h:mm:ss`。時間格式是在位於資料載入指令碼頂端的 `SET TimeFormat` 陳述式中指定。變更範例中的格式以配合您的需求。

函數範例

範例	結果
`maketime(22)`	傳回 22:00:00。
`maketime(22, 17)`	傳回 22:17:00。
`maketime(22,17,52 )`	傳回 22:17:52。

區域設定

除非另有說明,否則此主題中的範例皆使用下列日期格式:MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素,您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式,以滿足您的需求。或者,您可以在載入指令碼中變更格式,以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典,資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – maketime()

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組交易的資料集,這載入到稱為 `Transactions` 的表格中。
- 跨越三個欄位提供的交易時間:`hours`、`minutes` 和 `seconds`。
- 建立欄位 `transaction_time`,這以 `TimeFormat` 系統變數格式傳回時間。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- transaction_hour
- transaction_minute
- transaction_second
- transaction_time

結果表格

transaction_hour	transaction_minute	transaction_second	transaction_time
2	52	22	2:52:22 AM
6	32	07	6:32:07 AM
9	25	23	9:25:23 AM
12	09	16	12:09:16 PM
17	55	22	5:55:22 PM
18	43	30	6:43:30 PM
21	43	41	9:43:41 PM

`transaction_time` 欄位在前置 LOAD 陳述式中的建立方式是使用 `maketime()` 函數並傳遞小時、分鐘和秒欄位, 作為函數引數。

然後該函數會將這些值合併並轉換為時間欄位, 以 `TimeFormat` 系統變數的時間格式傳回結果。

範例 2 – time() 函數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 建立欄位 `transaction_time`，這可讓我們以 24 小時時間格式顯示結果，而不必修改 `TimeFormat` 系統變數。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- transaction_hour
- transaction_minute
- transaction_second
- transaction_time

結果表格

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

在此例中，maketime() 函數巢狀於 time() 函數內部。time() 函數的第二引數將 maketime() 函數結果的格式設為所需的 h:mm:ss。

範例 3 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組交易的資料集，這載入到稱為 Transactions 的表格中。
- 跨越兩個欄位提供的交易時間：hours 和 minutes。
- 建立欄位 transaction_time，這以 TimeFormat 系統變數格式傳回時間。

建立圖表物件量值 transaction_time，這以格式 h:mm:ss TT 傳回時間。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- transaction_hour
- transaction_minute

若要計算 transaction_time，建立此量值：

```
=maketime(transaction_hour,transaction_minute)
```

結果表格

transaction_hour	transaction_minute	=maketime(transaction_hour, transaction_minute)
2	52	2:52:00 AM
6	32	6:32:00 AM
9	25	9:25:00 AM
12	09	12:09:00 PM
17	55	5:55:00 PM
18	43	6:43:00 PM
21	43	9:43:00 PM

`transaction_time` 量值在圖表物件中的建立方式是使用 `maketime()` 函數並傳遞小時和分鐘欄位，作為函數引數。

然後該函數會合併這些值，秒數假設為 00。然後這些值會轉換為時間欄位，以 `TimeFormat` 系統變數格式傳回結果。

範例 4 – 情境

載入指令碼和圖表運算式

概覽

建立 2022 年 1 月的行事曆資料集，分為八小時增量。

載入指令碼

```
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;
```

結果

結果表格

時間戳記
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
還有 + 68 列

初始 `autogenerate` 函數在稱為 `tmpCalendar` 的表格中建立包含 1 月所有日期的行事曆。

會建立包含三個記錄的第二個表格。對於每個記錄,會採用 `recno() - 1` (值 0、1、2),結果會乘以 8。因此,這會產生值 0、8、16。這些值作為 `maketime()` 函數中的小時參數使用,分鐘和秒鐘值為 0。因此,表格包含三個時間欄位:12:00:00 AM、8:00:00 AM 和 4:00:00 PM。

此表格聯結到 `tmpCalendar` 表格。因為兩個表格之間沒有用於聯結的相符欄位,時間列會新增至每個日期列。因此,現在每個日期列會透過每個時間值重複三次。

最終,會從 `tmpCalendar` 表格的駐留載入建立行事曆表格。日期和時間欄位串連並包覆在 `timestamp()` 函數中,以建立時間戳記欄位。

然後會捨棄 `tmpCalendar` 表格。

makeweekdate

此函數會傳回從年、週數及星期幾計算而得的日期。

語法:

```
MakeWeekDate(weekyear [, week [, weekday [, first_week_day [, broken_weeks [, reference_day]]]]])
```

傳回的資料類型: 雙值

`makeweekdate()` 函數可作為指令碼和圖表函數。該函數將根據傳遞至函數中的參數計算日期。

引數

引數	描述
weekyear	`WeekYear()` 函數為特定日期定義的年份,亦即該週數屬於的年份。 *在某些情況下,該週年份可以與日曆年份不同,例如,若第 1 週已在上一年的 12 月開始。*
week	`Week()` 函數為特定日期定義的週數。 如未指定週數,則會採用 `1`。

引數	描述
weekday	`WeekDay()` 函數為問題中的日期定義的星期幾。0 是一週的第一天，而 6 是一週的最後一天。 如未指定星期幾，則會採用 0。 *即使 0 一律代表一週的第一天，而 6 一律是最後一天，對應至星期幾仍然由* ***first_week_day*** *參數決定。如果忽略，將使用變數* ***FirstWeekDay*** *的值。* 若使用中斷的週，以及不可能的參數組合，這可能會導致不屬於所選年份的結果。 **範例：** `MakeWeekDate(2021,1,0,6,1)` 傳回「Dec 27 2020」，因為此日是指定週的第一天 (星期日)。2021 年 1 月 1 日是星期五。
first_week_day	指定一週開始的日期。如果忽略，將使用變數 **FirstWeekDay** 的值。 **first_week_day** 可能的值是對星期一使用 0、對星期二使用 1，對星期三使用 2，對星期四使用 3，對星期五使用 4，對星期六使用 5，並對星期日使用 6。 如需系統變數的更多資訊，請參閱 *FirstWeekDay (page 212)*。
broken_weeks	如果您未指定 **broken_weeks**，則變數 **BrokenWeeks** 的值會用於定義週是否中斷。
reference_day	如果您不指定 **reference_day**，變數 **ReferenceDay** 的值會用於定義要設定一月份的哪一天為參照日以定義第 1 週。

什麼情況下使用

`makeweekdate()` 函數通常用於進行資料產生的指令碼中，以產生日期清單，或用來在輸入資料中提供年份、週和星期幾時建構日期。

下列範例假設：

```
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;
```

函數範例

範例	結果
`makeweekdate(2014,6,6)`	傳回 02/09/2014
`makeweekdate(2014,6,1)`	傳回 02/04/2014
`makeweekdate(2014,6)`	傳回 02/03/2014 (採用工作日 0)

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 含日子

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 稱為 Sales 的表格中包含 2022 年每週銷售總額的資料集。
- 跨越三個欄位提供的交易日期：year、week 和 sales。
- 用來建立量值 end_of_week 的前置載入，使用 makeweekdate() 函數以格式 MM/DD/YYYY 傳回該週星期五的日期。

若要證明傳回的日期是星期五，end_of_week 運算式也包覆在 weekday() 函數中，以顯示星期幾。

載入指令碼

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;

Transactions:
    Load
        *,
        makeweekdate(transaction_year, transaction_week,4) as end_of_week,
        weekday(makeweekdate(transaction_year, transaction_week,4)) as week_day
        ;
Load * Inline [
transaction_year, transaction_week, sales
2022, 01, 10000
2022, 02, 11250
2022, 03, 9830
2022, 04, 14010
2022, 05, 28402
```

```
2022, 06, 9992
2022, 07, 7292
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- transaction_year
- transaction_week
- end_of_week
- week_day

結果表格

transaction_year	transaction_week	end_of_week	week_day
2022	01	01/07/2022	星期五
2022	02	01/14/2022	星期五
2022	03	01/21/2022	星期五
2022	04	01/28/2022	星期五
2022	05	02/04/2022	星期五
2022	06	02/11/2022	星期五
2022	07	02/18/2022	星期五

「end_of_week」欄位使用 makeweekdate() 函數在前置 LOAD 陳述式中建立。透過該函數傳遞 transaction_week 欄位 transaction_year 作為年和週引數。值 4 用於日子引數。

然後該函數會將這些值合併並轉換為日期欄位,以 DateFormat 系統變數格式傳回結果。

makeweekdate() 函數及其引數也包覆在 weekday() 函數中,以傳回 week_day 欄位;如上表所見,week_day 欄位顯示這些日期的確發生在星期五。

範例 2 – 排除日子

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 稱為 Sales 的表格中包含 2022 年每週銷售總額的資料集。
- 跨越三個欄位提供的交易日期:year、week 和 sales。

- 使用 makeweekdate() 函數建立量值 first_day_of_week 的前置載入。這將會以格式 MM/DD/YYYY 傳回該週星期一的日期。

若要證明傳回的日期是星期一，first_day_of_week 運算式也包覆在 weekday() 函數中，以顯示星期幾。

載入指令碼

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;

Transactions:
    Load
        *,
        makeweekdate(transaction_year, transaction_week) as first_day_of_week,
        weekday(makeweekdate(transaction_year, transaction_week)) as week_day
        ;
Load * Inline [
transaction_year, transaction_week, sales
2022, 01, 10000
2022, 02, 11250
2022, 03, 9830
2022, 04, 14010
2022, 05, 28402
2022, 06, 9992
2022, 07, 7292
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- transaction_year
- transaction_week
- first_day_of_week
- week_day

結果表格

transaction_year	transaction_week	first_day_of_week	week_day
2022	01	01/03/2022	星期一
2022	02	01/10/2022	星期一
2022	03	01/17/2022	星期一
2022	04	01/24/2022	星期一
2022	05	01/31/2022	星期一

transaction_year	transaction_week	first_day_of_week	week_day
2022	06	02/07/2022	星期一
2022	07	02/14/2022	星期一

first_day_of_week 欄位使用 makeweekdate() 函數在前置 LOAD 陳述式中建立。transaction_year 和 transaction_week 參數作為函數引數傳遞，而日子參數則留下空白。

然後該函數會將這些值合併並轉換為日期欄位，以 DateFormat 系統變數格式傳回結果。

makeweekdate() 函數及其引數也會包覆在 weekday() 函數中，以傳回 week_day 欄位。如上表所見，week_day 欄位在所有情況下都傳回星期一，因為該參數在 makeweekdate() 函數中為空白，預設為 0 (一週的第一天)，而一週的第一天由 FirstWeekDay 系統變數設為星期一。

範例 3 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 稱為 Sales 的表格中包含 2022 年每週銷售總額的資料集。
- 跨越三個欄位提供的交易日期：year、week 和 sales。

在此範例中，圖表物件將用來建立等同於第一個範例中 end_of_week 計算的量值。此量值將會使用 makeweekdate() 函數以格式 MM/DD/YYYY 傳回該週星期五的日期。

若要證明傳回的日期是星期五，建立第二個量值，以傳回星期幾。

載入指令碼

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;

Master_Calendar:
Load * Inline [
transaction_year, transaction_week, sales
2022, 01, 10000
2022, 02, 11250
2022, 03, 9830
2022, 04, 14010
2022, 05, 28402
2022, 06, 9992
2022, 07, 7292
];
```

結果

請執行下列動作：

1. 載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：
   - transaction_year
   - transaction_week
2. 若要執行等同於第一個範例中 end_of_week 欄位的計算，建立下列量值：
   =makeweekdate(transaction_year,transaction_week,4)
3. 若要計算每個交易是星期幾，建立下列量值：
   =weekday(makeweekdate(transaction_year,transaction_week,4))

結果表格

transaction_ year	transaction_ week	=makeweekdate (transaction_ year,transaction_week,4)	=weekday(makeweekdate (transaction_year,transaction_ week,4))
2022	01	01/07/2022	星期五
2022	02	01/14/2022	星期五
2022	03	01/21/2022	星期五
2022	04	01/28/2022	星期五
2022	05	02/04/2022	星期五
2022	06	02/11/2022	星期五
2022	07	02/18/2022	星期五

會使用 makeweekdate() 函數在圖表物件中建立等同於 end_of_week 的欄位作為量值。傳遞 transaction_year 和 transaction_week 欄位作為年和週引數。值 4 用於日子引數。

然後該函數會將這些值合併並轉換為日期欄位，以 DateFormat 系統變數格式傳回結果。

makeweekdate() 函數及其引數也會包覆在 weekday() 函數中，以傳回等同於第一個範例中 week_day 欄位的計算。如上表所見，右側最後一欄顯示這些日期的確發生在星期五。

範例 4 – 情境

載入指令碼和圖表運算式

概覽

在此範例中，建立包含 2022 年所有星期五的日期清單。

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;

Calendar:
    load
          *,
        weekday(date) as weekday
    where year(date)=2022;
load
    makeweekdate(2022,recno()-2,4) as date
AutoGenerate 60;
```

結果

結果表格

日期	weekday
01/07/2022	星期五
01/14/2022	星期五
01/21/2022	星期五
01/28/2022	星期五
02/04/2022	星期五
02/11/2022	星期五
02/18/2022	星期五
02/25/2022	星期五
03/04/2022	星期五
03/11/2022	星期五
03/18/2022	星期五
03/25/2022	星期五
04/01/2022	星期五
04/08/2022	星期五
04/15/2022	星期五
04/22/2022	星期五
04/29/2022	星期五
05/06/2022	星期五

日期	weekday
05/13/2022	星期五
05/20/2022	星期五
05/27/2022	星期五
06/03/2022	星期五
06/10/2022	星期五
06/17/2022	星期五
還有 + 27 列	

makeweekdate() 函數尋找 2022 年的每個星期五。使用週參數 -2 確保不會遺漏任何日期。最後,前置載入建立附加 weekday 欄位,以便清楚顯示每個 date 值是星期五。

minute

當 **expression** 的分數根據標準數字解譯的方式可解譯為時間時,此函數會傳回代表分鐘的整數。

語法:

```
minute(expression)
```

傳回的資料類型: 整數

什麼情況下使用

若您想要按分鐘比較彙總,則 minute() 函數很實用。例如,若您想要按分鐘查看活動計數分佈,可使用該函數。

可以使用該函數建立主要行事曆表格中的欄位,以在載入指令碼中建立這些維度。相反地,這可直接在圖表中作為計算維度使用。

函數範例

範例	結果
minute ('09:14:36')	傳回 14。
minute ('0.5555')	傳回 19 (因為 0.5555 = 13:19:55)。

區域設定

除非另有說明,否則此主題中的範例皆使用下列日期格式:MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素,您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式,以滿足您的需求。或者,您可以在載入指令碼中變更格式,以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典, 資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 變數 (指令碼)

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含依時間戳記之交易的資料集, 這載入到稱為 `Transactions` 的表格中。
- 已使用預設的 `TimeStamp` 系統變數 (`M/D/YYYY h:mm:ss[.fff] TT`)。
- 交易時用於計算的欄位 `minute` 建立。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- timestamp
- minute

結果表格

時間戳記	minute
2022-01-01 22:10:22	10
2022-01-02 08:35:54	35
2022-01-03 05:40:49	40
2022-01-03 14:21:53	21
2022-01-04 18:49:38	49
2022-01-04 22:58:34	58
2022-01-05 19:04:57	4
2022-01-05 19:34:46	34
2022-01-06 08:49:09	49
2022-01-06 11:29:38	29

使用 minute() 函數並傳遞 timestamp 作為前置 LOAD 陳述式中的運算式，以建立 minute 欄位中的值。

範例 2 – 圖表物件 (圖表)

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 已使用預設的 TimeStamp 系統變數 (M/D/YYYY h:mm:ss[.fff] TT)。

不過，在此範例中，不變的資料集會載入到應用程式中。會透過圖表物件中的量值計算 minute 值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`timestamp`。

建立下列量值：

`=minute(timestamp)`

結果表格

時間戳記	minute
2022-01-01 22:10:22	10
2022-01-02 08:35:54	35
2022-01-03 05:40:49	40
2022-01-03 14:21:53	21
2022-01-04 18:49:38	49
2022-01-04 22:58:34	58
2022-01-05 19:04:57	4
2022-01-05 19:34:46	34
2022-01-06 08:49:09	49
2022-01-06 11:29:38	29

`minute` 值的建立方式是使用 `minute()` 函數並傳遞 `timestamp` 作為圖表物件量值中的運算式。

範例 3 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 時間戳記的資料集，產生以在票閘入口呈現。
- 含每個 `timestamp` 及其對應 `id` 的資訊，這載入到稱為 `Ticket_Barrier_Tracker` 的表格中。
- 已使用預設的 `TimeStamp` 系統變數 (`M/D/YYYY h:mm:ss[.fff] TT`)。

使用者希望圖表物件依分鐘顯示票閘入口計數。

載入指令碼

```
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;
```

結果

請執行下列動作：

1. 載入資料並開啟工作表。建立新的表格。
2. 使用下列運算式建立計算維度：
   `=minute(timestamp)`
3. 新增下列彙總量值以計算總入口數：
   `=count(id)`
4. 將量值的**數字格式**設定為**金錢**。

結果表格

minute(timestamp)	=count(id)
0	174
1	171
2	175
3	165

minute(timestamp)	=count(id)
4	188
5	176
6	158
7	187
8	178
9	178
10	197
11	161
12	166
13	184
14	159
15	161
16	152
17	160
18	176
19	164
20	170
21	170
22	142
23	145
24	155
還有 + 35 列	

month

此函數會傳回雙值:一個如環境變數 **MonthNames** 中所定義的月份名稱,一個介於 1 與 12 之間的整數。月份是根據標準數字解譯,從運算式的日期解譯計算所得。

該函數針對特定日期以 `MonthName` 系統變數的格式傳回月份名稱。這通常用來建立日欄位,作為主要行事曆中的維度。

語法:

```
month(expression)
```

傳回的資料類型: 整數

函數範例

範例	**結果**
month(2012-10-12)	傳回十月
month(35648)	傳回八月，因為 35648 = 1997-08-06

範例 1 – DateFormat 資料集 (指令碼)

載入指令碼和結果

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 名稱為 Master_Calendar 的日期資料集。DateFormat 系統變數設定為 DD/MM/YYYY。
- 使用 month() 函數建立附加欄位 (名稱為 month_name) 的前置載入。
- 名稱為 long_date 的附加欄位，使用 date() 函數表示完整日期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- long_date
- month_name

結果表格

日期	long_date	month_name
03/01/2022	2022 年 1 月3 日	一月
03/02/2022	2022 年 2 月 3 日	二月
03/03/2022	2022 年 3 月 3 日	三月
03/04/2022	2022 年 4 月 3 日	四月
03/05/2022	2022 年 5 月 3 日	五月
03/06/2022	2022 年 6 月 3 日	六月
03/07/2022	2022 年 7 月 3 日	7 月
03/08/2022	2022 年 8 月 3 日	8 月
03/09/2022	2022 年 9 月 3 日	9 月
03/10/2022	2022 年 10 月 3 日	10 月
03/11/2022	2022 年 11 月 3 日	11 月

月份名稱目前由指令碼中的 month() 函數正確評估。

範例 2 – ANSI 日期 (指令碼)

載入指令碼和結果

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 名稱為 Master_Calendar 的日期資料集。使用 DateFormat 系統變數 DD/MM/YYYY。不過, 資料集中包括的日期屬於 ANSI 標準日期格式。
- 使用 month() 函數建立附加欄位 (名稱為 month_name) 的前置載入。
- 名稱為 long_date 的附加欄位, 使用 date() 函數表示完整日期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- long_date
- month_name

結果表格

日期	long_date	month_name
03/11/2022	2022 年 3 月 11 日	11
03/12/2022	2022 年 3 月 12 日	12
03/13/2022	2022 年 3 月 13 日	13
03/14/2022	2022 年 3 月 14 日	14
03/15/2022	2022 年 3 月 15 日	15
03/16/2022	2022 年 3 月 16 日	16
03/17/2022	2022 年 3 月 17 日	17
03/18/2022	2022 年 3 月 18 日	18
03/19/2022	2022 年 3 月 19 日	19
03/20/2022	2022 年 3 月 20 日	20
03/21/2022	2022 年 3 月 21 日	21

月份名稱目前由指令碼中的 month() 函數正確評估。

範例 3 – 未格式化的日期 (指令碼)

載入指令碼和結果

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 名稱為 `Master_Calendar` 的日期資料集。使用 `DateFormat` 系統變數 DD/MM/YYYY。
- 使用 `month()` 函數建立附加欄位 (名稱為 `month_name`) 的前置載入。
- 原始未格式化的日期，名稱為 `unformatted_date`。
- 名稱為 `long_date` 的附加欄位，使用 `date()` 函數表示完整日期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- `unformatted_date`
- `long_date`
- `month_name`

結果表格

unformatted_date	long_date	month_name
44868	2022 年 1 月3 日	一月
44898	2022 年 2 月3 日	二月
44928	2022 年 3 月3 日	三月
44958	2022 年 4 月3 日	四月
44988	2022 年 5 月3 日	五月
45018	2022 年 6 月3 日	六月
45048	2022 年 7 月3 日	7 月
45078	2022 年 8 月3 日	8 月
45008	2022 年 9 月3 日	9 月
45038	2022 年 10 月3 日	10 月
45068	2022 年 11 月 3 日	11 月

月份名稱目前由指令碼中的 `month()` 函數正確評估。

範例 4 – 計算到期月份

載入指令碼和圖表運算式

概述

開啟 資料載入編輯器 並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 名稱為 `Subscriptions` 的三月下訂訂單資料集。表格含有三個欄位:
  - id
  - order_date
  - 金額

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：order_date。

若要計算訂單到期的月份，建立此量值：=month(order_date+180)。

結果表格

order_date	=month(order_date+180)
03/01/2022	7 月
03/02/2022	8 月
03/03/2022	8 月
03/04/2022	9 月
03/05/2022	10 月
03/06/2022	11 月
03/07/2022	12 月
03/08/2022	一月
03/09/2022	三月
03/10/2022	四月
03/11/2022	五月

month() 函數正確判定在 3 月 11 日下訂的訂單會在 7 月到期。

monthend

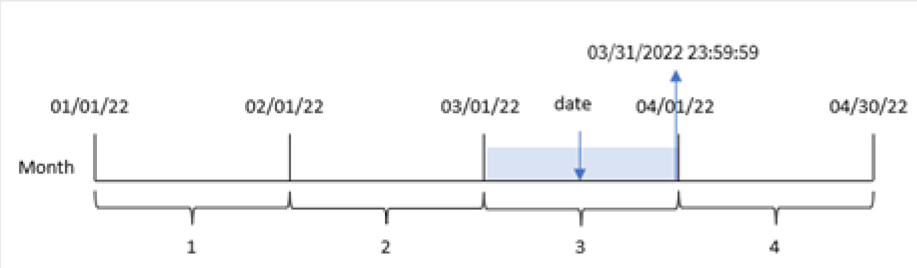
此函數傳回的值相當於包含 date 的月份最後一天、最後一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 DateFormat。

語法：

MonthEnd(date[, period_no]**)**

換言之，monthend() 函數判定日期落在哪個月。然後以日期格式傳回該月最後一毫秒的時間戳記。

monthend 函數的圖表。



什麼情況下使用

monthend() 函數在您想要計算以使用一個月中尚未發生的部分時，作為運算式的一部分使用。例如，若您想要計算該月期間尚未發生的總利息。

傳回的資料類型： 雙值

引數

引數	描述
date	要評估的時間戳記。
period_no	**period_no** 是一個整數，如果為 0 或忽略，則表示包含 **date** 的月份。負值的 **period_no** 表示之前的月份，正值表示之後的月份。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
monthend('02/19/2012')	傳回 02/29/2012 23:59:59。
monthend('02/19/2001', -1)	傳回 01/31/2001 23:59:59。

範例 1 – 基本範例

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2022 年交易的資料集,這載入到稱為「Transactions」的表格中。
- 使用 DateFormat 系統變數 (MM/DD/YYYY) 格式的日期欄位。
- 前置 LOAD 陳述式包含:
    - 設定為欄位「end_of_month」的 monthend() 函數。
    - 設定為欄位「end_of_month_timestamp」的 timestamp 函數。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- id
- date
- end_of_month
- end_of_month_timestamp

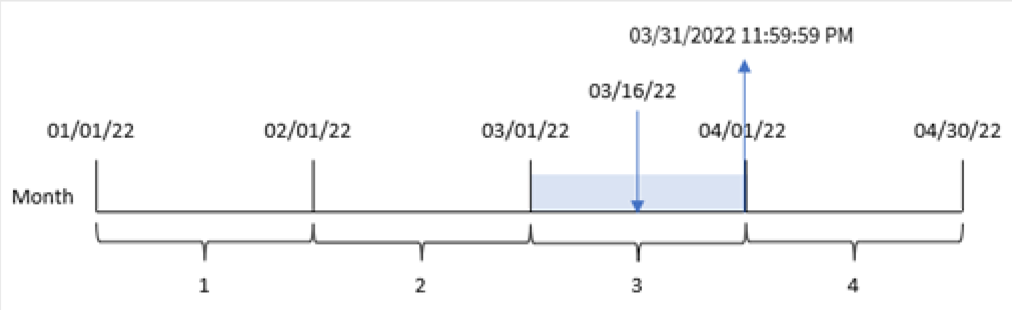
結果表格

id	日期	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

「end_of_month」欄位在前置 LOAD 陳述式中的建立方式是使用 monthend() 函數並傳遞日期欄位，作為函數的引數。

monthend() 函數識別日期值落在哪個月，並傳回該月最後一毫秒的時間戳記。

以 3 月作為所選月份的 monthend 函數圖表。



交易 8192 發生於 3 月 16 日。monthend() 函數傳回該月的最後一毫秒，亦即 3 月 31 日下午 11:59:59。

範例 2 – period_no

載入指令碼和結果

概覽

使用與第一個範例相同的資料集和情境。

在此範例中，任務是建立欄位「previous_month_end」，這傳回交易發生的前一個月結束的時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

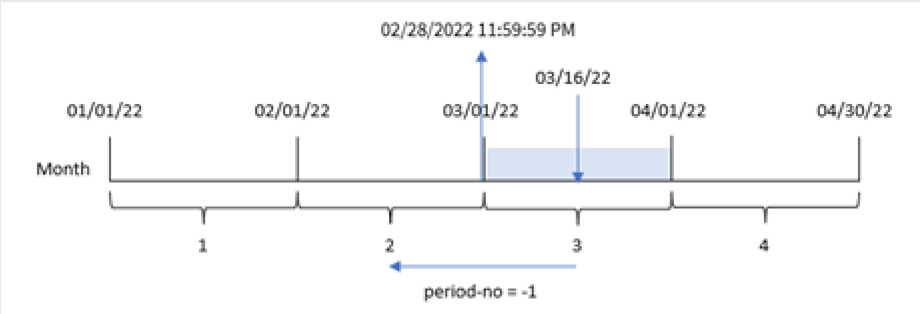
- id
- date
- previous_month_end
- previous_month_end_timestamp

結果表格

id	日期	previous_month_end	previous_month_end_timestamp
8188	1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
8189	1/19/2022	12/31/2021	12/31/2021 11:59:59 PM
8190	2/5/2022	01/31/2022	1/31/2022 11:59:59 PM
8191	2/28/2022	01/31/2022	1/31/2022 11:59:59 PM
8192	3/16/2022	02/28/2022	2/28/2022 11:59:59 PM
8193	4/1/2022	03/31/2022	3/31/2022 11:59:59 PM
8194	5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
8195	5/16/2022	04/30/2022	4/30/2022 11:59:59 PM
8196	6/15/2022	05/31/2022	5/31/2022 11:59:59 PM
8197	6/26/2022	05/31/2022	5/31/2022 11:59:59 PM
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	7/23/2022	06/30/2022	6/30/2022 11:59:59 PM
8201	7/27/2022	06/30/2022	6/30/2022 11:59:59 PM
8202	8/2/2022	07/31/2022	7/31/2022 11:59:59 PM
8203	8/8/2022	07/31/2022	7/31/2022 11:59:59 PM
8204	8/19/2022	07/31/2022	7/31/2022 11:59:59 PM
8205	9/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8206	10/14/2022	09/30/2022	9/30/2022 11:59:59 PM
8207	10/29/2022	09/30/2022	9/30/2022 11:59:59 PM

monthend() 函數首先會識別交易發生的月份，因為 -1 的 period_no 作為偏移引數使用。然後這會往前偏移一個月並識別該月的最後一毫秒。

具有 period_no 變數的 monthend 函數圖表。



交易 8192 發生於 3 月 16 日。monthend() 函數識別交易發生的前一個月為 2 月。然後這會傳回該月的最後一毫秒 2 月 28 日 11:59:59 PM。

範例 3 – 圖表範例

載入指令碼和圖表運算式

概覽

使用與第一個範例相同的資料集和情境。

在此範例中，資料集保持不變並且會載入到應用程式中。任務是要建立傳回交易發生月份結束時間戳記的計算，作為應用程式圖表中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- id

若要計算交易發生的該月結束日期，建立下列量值：

- =monthend(date)
- =timestamp(monthend(date))

結果表格

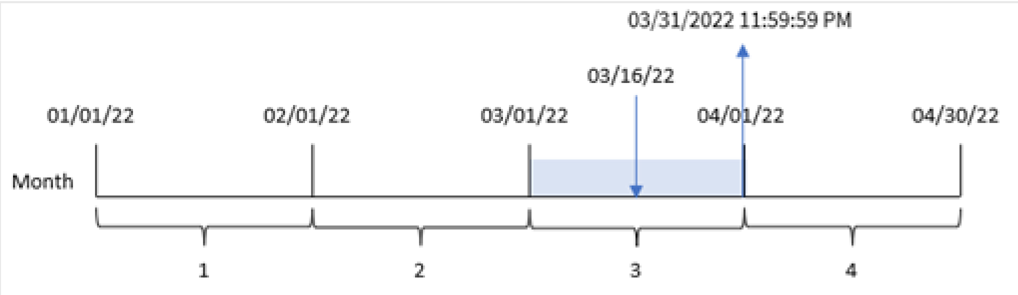
id	日期	=monthend(date)	=timestamp(monthend(date))
8188	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8189	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM
8190	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8191	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8192	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8193	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8194	7/9/2022	07/31/2022	7/31/2022 11:59:59 PM
8195	7/22/2022	07/31/2022	7/31/2022 11:59:59 PM
8196	7/23/2022	07/31/2022	7/31/2022 11:59:59 PM
8197	7/27/2022	07/31/2022	7/31/2022 11:59:59 PM
8198	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8201	5/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8202	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8203	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8204	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8205	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM

id	日期	=monthend(date)	=timestamp(monthend(date))
8206	1/7/2022	01/31/2022	1/31/2022 11:59:59 PM
8207	1/19/2022	01/31/2022	1/31/2022 11:59:59 PM

「end_of_month」量值在圖表中的建立方式是使用 monthend() 函數並傳遞日期欄位，作為函數的引數。

monthend() 函數識別日期值落在哪個月，並傳回該月最後一毫秒的時間戳記。

具有 period_no 變數的 monthend 函數圖表。



交易 8192 發生於 3 月 16 日。monthend() 函數傳回該月的最後一毫秒，亦即 3 月 31 日下午 11:59:59。

範例 4 – 情境

載入指令碼和結果

概覽

在此範例中，資料集會載入到稱為「Employee_Expenses」的表格中。該表格含有下列欄位：

- 員工 ID
- 員工名稱
- 每個員工報銷的平均每日開支。

最終使用者希望圖表按員工 ID 和員工名稱顯示該月剩餘期間的預估開支報銷。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- employee_id
- employee_name

若要計算累積的利息，建立此量值：

`=floor(monthend(today(1),0)-today(1))*avg_daily_claim`

此量值是動態性質，將會根據載入資料的日期產生不同的表格結果。

將量值的**數字格式**設定為**金錢**。

結果表格

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

藉由使用今日日期作為其唯一引數，monthend() 函數會傳回目前月份的結束日期。以該月結束日期減去今日日期，運算式就會傳回此月剩餘天數。

然後按員工讓此值乘以平均每日開支報銷，以計算每個員工在該月剩餘期間預期會有的預估報銷值。

monthname

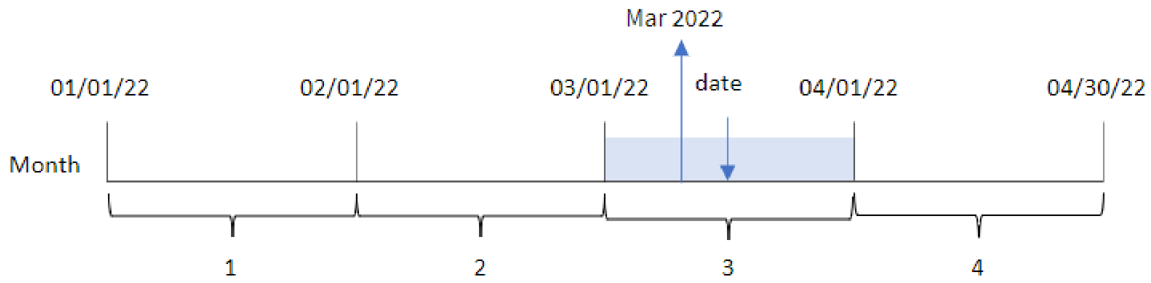
此函數會傳回顯示月份 (根據 **MonthNames** 指令碼變數進行格式設定) 和年度的顯示值，其基礎數值相當於該月份第一天、第一毫秒的時間戳記。

語法：

```
MonthName(date[, period_no])
```

傳回的資料類型： 雙值

monthname 函數的圖表



引數

引數	描述
date	要評估的時間戳記。
period_no	**period_no** 是一個整數，如果為 0 或忽略，則表示包含 **date** 的月份。負值的 **period_no** 表示之前的月份，正值表示之後的月份。

函數範例

範例	結果
`monthname('10/19/2013')`	傳回 Oct 2013
`monthname('10/19/2013', -1)`	傳回 Sep 2013

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 基本範例

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2022 年交易的資料集, 這載入到稱為 `Transactions` 的表格中。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 `transaction_month`, 這傳回交易發生的月份。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

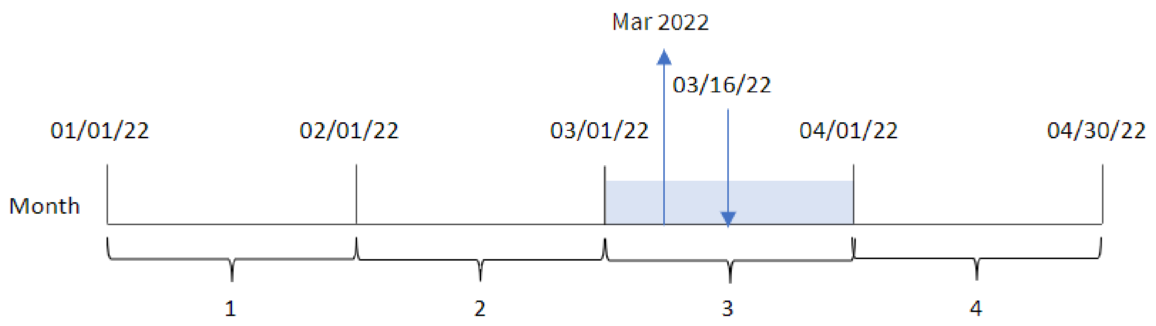
- date
- transaction_month

結果表格

日期	transaction_month
1/7/2022	2022 年 1 月
1/19/2022	2022 年 1 月
2/5/2022	2022 年 2 月
2/28/2022	2022 年 2 月
3/16/2022	2022 年 3 月
4/1/2022	2022 年 4 月
5/7/2022	2022 年 5 月
5/16/2022	2022 年 5 月
6/15/2022	2022 年 6 月
6/26/2022	2022 年 6 月
7/9/2022	2022 年 7 月
7/22/2022	2022 年 7 月
7/23/2022	2022 年 7 月
7/27/2022	2022 年 7 月
8/2/2022	2022 年 8 月
8/8/2022	2022 年 8 月
8/19/2022	2022 年 8 月
9/26/2022	2022 年 9 月
10/14/2022	2022 年 10 月
10/29/2022	2022 年 10 月

transaction_month 欄位在前置 LOAD 陳述式中的建立方式是使用 monthname() 函數，並傳遞 date 欄位，作為函數的引數。

monthname 函數的圖表，基本範例



monthname() 函數決定交易 8192 發生在 2022 年 3 月，並使用 MonthNames 系統變數傳回此值。

範例 2 – period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的內嵌資料集和情境。
- 建立欄位 transaction_previous_month，這傳回交易發生之前的月份結束的時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- transaction_previous_month

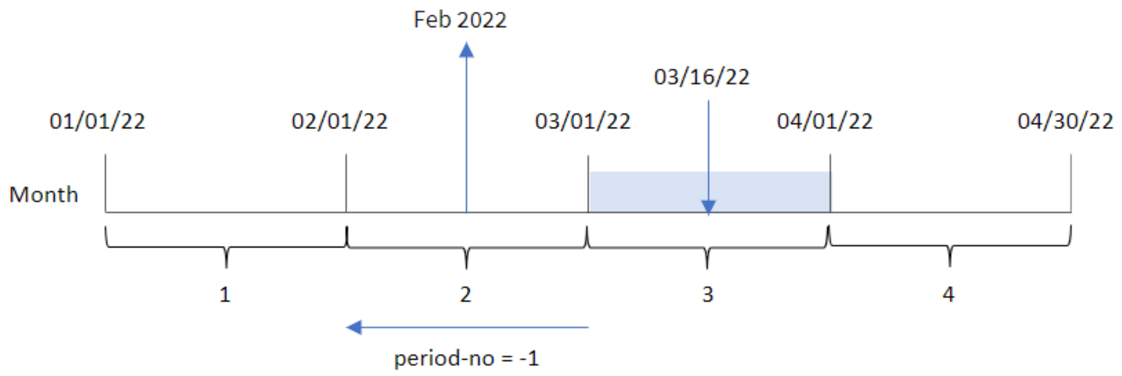
結果表格

日期	transaction_previous_month
1/7/2022	2021 年 12 月
1/19/2022	2021 年 12 月
2/5/2022	2022 年 1 月
2/28/2022	2022 年 1 月
3/16/2022	2022 年 2 月
4/1/2022	2022 年 3 月
5/7/2022	2022 年 4 月
5/16/2022	2022 年 4 月
6/15/2022	2022 年 5 月
6/26/2022	2022 年 5 月
7/9/2022	2022 年 6 月
7/22/2022	2022 年 6 月
7/23/2022	2022 年 6 月
7/27/2022	2022 年 6 月
8/2/2022	2022 年 7 月
8/8/2022	2022 年 7 月
8/19/2022	2022 年 7 月

日期	transaction_previous_month
9/26/2022	2022 年 8 月
10/14/2022	2022 年 9 月
10/29/2022	2022 年 9 月

在此例中，因為 period_no 的值 -1 已作為 monthname() 函數中的偏移引數使用，所以該函數首先會識別交易發生的月份。然後這會往前移動一個月並傳回月份名稱和年份。

monthname 函數的圖表，period_no 範例



交易 8192 發生於 3 月 16 日。monthname() 函數識別交易發生之前的月份是 2 月，並以 MonthNames 系統變數格式傳回該月份以及 2022 年。

範例 3 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的內嵌資料集和情境。不過，在此範例中，不變的資料集會載入到應用程式中。會建立傳回交易發生月份結束時間戳記的計算，作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`date`。

建立下列量值：

`=monthname(date)`

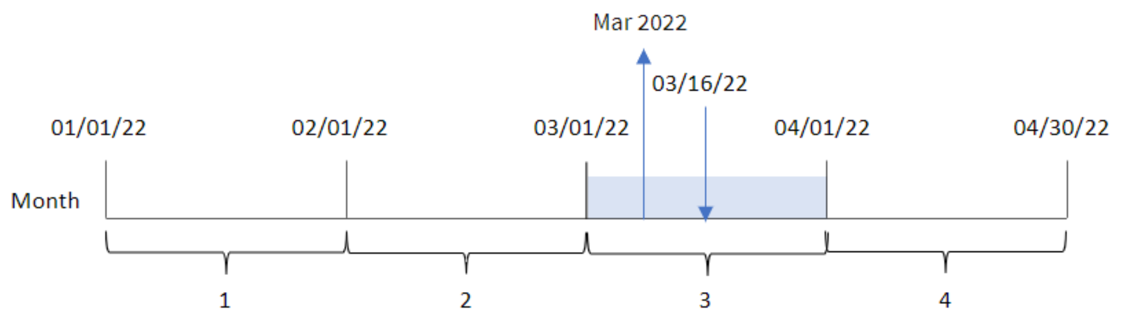
結果表格

日期	=monthname(date)
1/7/2022	2022 年 1 月
1/19/2022	2022 年 1 月
2/5/2022	2022 年 2 月
2/28/2022	2022 年 2 月
3/16/2022	2022 年 3 月
4/1/2022	2022 年 4 月
5/7/2022	2022 年 5 月
5/16/2022	2022 年 5 月
6/15/2022	2022 年 6 月
6/26/2022	2022 年 6 月
7/9/2022	2022 年 7 月
7/22/2022	2022 年 7 月

日期	=monthname(date)
7/23/2022	2022 年 7 月
7/27/2022	2022 年 7 月
8/2/2022	2022 年 8 月
8/8/2022	2022 年 8 月
8/19/2022	2022 年 8 月
9/26/2022	2022 年 9 月
10/14/2022	2022 年 10 月
10/29/2022	2022 年 10 月

month_name 量值在圖表物件中的建立方式是使用 monthname() 函數並傳遞 date 欄位，作為函數的引數。

monthname 函數的圖表，圖表物件範例



monthname() 函數決定交易 8192 發生在 2022 年 3 月，並使用 MonthNames 系統變數傳回此值。

monthsend

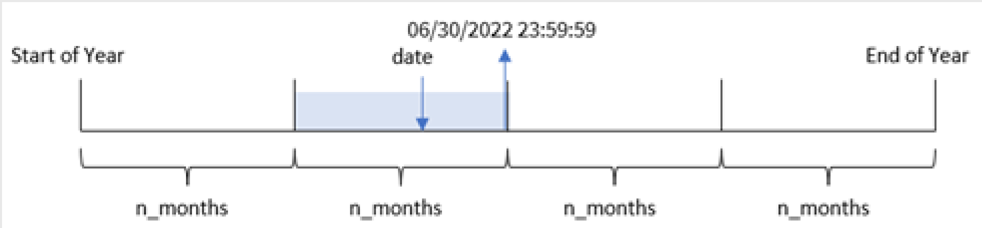
此函數傳回的值相當於包含基本日期的月、雙月、季度、四月期或半年的最後一毫秒的時間戳記。還可以尋找前一個或後一個時間期間結束時的時間戳記。預設的輸出格式為指令碼中所設定的 DateFormat。

語法：

```
MonthsEnd(n_months, date[, period_no [, first_month_of_year]])
```

傳回的資料類型: 雙值

monthsend 函數的圖表。



引數

引數	描述
n_months	定義期間的月數。整數或解析為整數的運算式,必須是下列其中一項:1 (相當於 inmonth() 函數)、2 (雙月)、3 (相當於 inquarter() 函數)、4 (四月期) 或 6 (半年)。
date	要評估的時間戳記。
period_no	期間可以使用 **period_no** 位移,是一個整數或解譯為整數的運算式,其中值 0 表示包含 **base_date** 的期間。負值的 **period_no** 表示之前的期間,正值表示之後的期間。
first_ month_ of_year	如果要使用不起始於 1 月的 (會計) 年度,可在 **first_month_of_year** 中指定介於 2 和 12 之間的值。

monthsend() 函數根據提供的 n_months 引數將該年分為幾個區段。然後這會評估提供的每個日期落在哪些區段,並以日期格式傳回該區段的最後一毫秒。該函數可以從上一個或下一個區段傳回結束時間戳記以及重新定義該年的第一個月。

一年中的下列區段可作為 n_month 引數用於函數。

n_month 引數

期間	月數
月	1
兩個月	2
季	3
四個月	4
半年	6

什麼情況下使用

monthsend() 函數在使用者想要計算使用一個月中已經過的部分時,作為運算式的一部分使用。使用者有機會使用變數選取所選期間。例如,monthsend() 可以提供輸入變數,以讓使用者計算該月、該季或半年期間尚未產生的總利息。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
`monthsend(4, '07/19/2013')`	傳回 08/31/2013。
`monthsend(4, '10/19/2013', -1)`	傳回 08/31/2013。
`monthsend(4, '10/19/2013', 0, 2)`	傳回 01/31/2014。 因為年開始變成月份 2。

範例 1 - 基本範例

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為「`Transactions`」的表格中。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 前置 LOAD 陳述式包含：
    - 設定為欄位「`bi_monthly_end`」的 `monthsend` 函數。這將交易分成兩個月區段。
    - `timestamp` 函數為每個交易傳回區段的開始時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- id
- date
- bi_monthly_end
- bi_monthly_end_timestamp

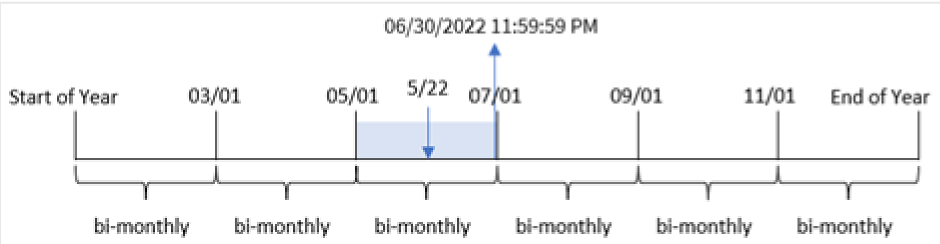
結果表格

id	日期	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

id	日期	bi_monthly_end	bi_monthly_end_timestamp
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	10/31/2022	10/31/2022 11:59:59 PM
8206	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8207	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

「bi_monthly_end」欄位使用 monthsend() 函數在前置 LOAD 陳述式中建立。提供的第一個引數為 2，將該年分為兩個月區段。第二個引數識別正在評估哪個欄位。

具有兩個月區段的 monthsend 函數圖表。



交易 8195 發生在 5 月 22 日。monthsend() 函數原本將該年分為兩個月區段。交易 8195 落在 5 月 1 日和 6 月 30 日之間的區段內。因此，該函數回傳此區段的最後一毫秒 06/30/2022 11:59:59 PM。

範例 2 - period_no

載入指令碼和結果

概覽

使用與第一個範例相同的資料集和情境。

在此範例中，任務是建立欄位「prev_bi_monthly_end」，這傳回交易發生之前的兩個月區段的第一毫秒。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date
- prev_bi_monthly_end
- prev_bi_monthly_end_timestamp

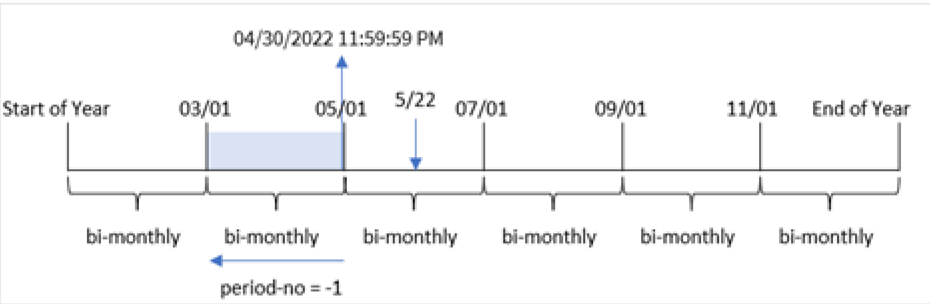
結果表格

id	日期	prev_bi_monthly_end	prev_bi_monthly_end_timestamp
8188	1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
8189	1/19/2022	12/31/2021	12/31/2021 11:59:59 PM
8190	2/5/2022	12/31/2021	12/31/2021 11:59:59 PM
8191	2/28/2022	12/31/2021	12/31/2021 11:59:59 PM

id	日期	prev_bi_monthly_end	prev_bi_monthly_end_timestamp
8192	3/16/2022	02/28/2022	2/28/2022 11:59:59 PM
8193	4/1/2022	02/28/2022	2/28/2022 11:59:59 PM
8194	5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
8195	5/22/2022	04/30/2022	4/30/2022 11:59:59 PM
8196	6/15/2022	04/30/2022	4/30/2022 11:59:59 PM
8197	6/26/2022	04/30/2022	4/30/2022 11:59:59 PM
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	7/23/2022	06/30/2022	6/30/2022 11:59:59 PM
8201	7/27/2022	06/30/2022	6/30/2022 11:59:59 PM
8202	8/2/2022	06/30/2022	6/30/2022 11:59:59 PM
8203	8/8/2022	06/30/2022	6/30/2022 11:59:59 PM
8204	8/19/2022	06/30/2022	6/30/2022 11:59:59 PM
8205	9/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8206	10/14/2022	08/31/2022	8/31/2022 11:59:59 PM
8207	10/29/2022	08/31/2022	8/31/2022 11:59:59 PM

透過使用 `-1` 作為 `monthsend()` 函數中的 `period_no` 引數，原本將年分為兩個月區段後，該函數就會針對交易發生時間傳回上一個兩個月區段的最後一毫秒。

傳回上一個兩個月區段的 `monthsend` 函數圖表。



交易 8195 發生在 5 月 和 6 月 之間的區段內。因此，上一個兩個月區段介於 3 月 1 日和 4 月 30 日之間，所以該函數會傳回此區段的最後一毫秒 4/30/2022 11:59:59 PM。

範例 3 – first_month_of_year

載入指令碼和結果

概覽

使用與第一個範例相同的資料集和情境。

在此範例中，組織政策適用於 4 月作為會計年度第一個月份的情況。

建立欄位「bi_monthly_end」，這將交易組成兩個月區段，並為每個交易傳回該區段的最後一毫秒時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date

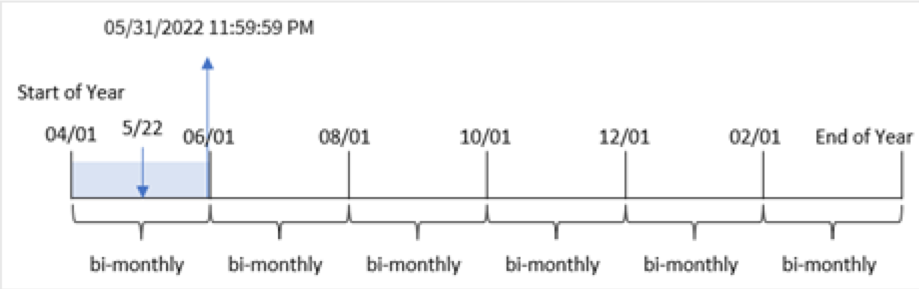
- `bi_monthly_end`
- `bi_monthly_end_timestamp`

結果表格

id	日期	bi_monthly_end	bi_monthly_end_timestamp
8188	1/7/2022	01/31/2022	1/31/2022 11:59:59 PM
8189	1/19/2022	01/31/2022	1/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	05/31/2022	5/31/2022 11:59:59 PM
8194	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8195	5/22/2022	05/31/2022	5/31/2022 11:59:59 PM
8196	6/15/2022	07/31/2022	7/31/2022 11:59:59 PM
8197	6/26/2022	07/31/2022	7/31/2022 11:59:59 PM
8198	7/9/2022	07/31/2022	7/31/2022 11:59:59 PM
8199	7/22/2022	07/31/2022	7/31/2022 11:59:59 PM
8200	7/23/2022	07/31/2022	7/31/2022 11:59:59 PM
8201	7/27/2022	07/31/2022	7/31/2022 11:59:59 PM
8202	8/2/2022	09/30/2022	9/30/2022 11:59:59 PM
8203	8/8/2022	09/30/2022	9/30/2022 11:59:59 PM
8204	8/19/2022	09/30/2022	9/30/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	11/30/2022	11/30/2022 11:59:59 PM
8207	10/29/2022	11/30/2022	11/30/2022 11:59:59 PM

若使用 4 作為 `monthsend()` 函數中的 `first_month_of_year` 引數，該函數會讓該年從 4 月 1 日開始，然後將該年分為兩個月區段：4 月-5 月、6 月-7 月、8 月-9 月、10 月-11 月、12 月-1 月、2 月-3 月。

將該年第一個月設為 4 月的 monthsend 函數圖表



交易 8195 發生在 5 月 22 日並落在 4 月 1 日和 5 月 31 日之間的區段內。因此，該函數回傳此區段的最後一毫秒 05/31/2022 11:59:59 PM。

範例 4 - 圖表物件範例

載入指令碼和圖表運算式

概覽

使用與第一個範例相同的資料集和情境。不過，在此範例中，資料集保持不變並且會載入到應用程式中。

在此範例中，任務是建立將交易組成兩個月區段的計算，並為每個交易傳回該區段的最後一毫秒時間戳記，作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：

`date`

若要擷取交易發生之兩個月區段的最後一毫秒時間戳記，建立下列量值：

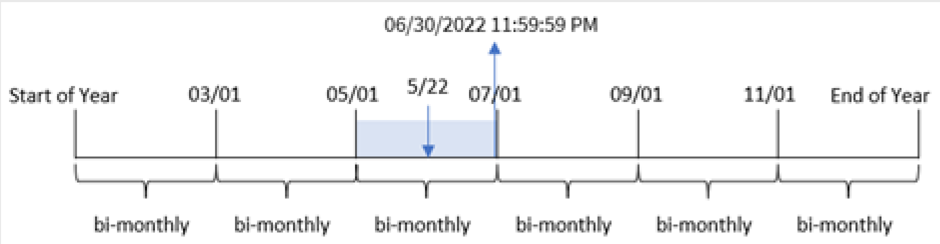
- `=monthsEnd(2,date)`
- `=timestamp(monthsend(2,date))`

結果表格

id	日期	=monthsend(2,date)	=timestamp(monthsend(2,date))
8188	1/7/2022	02/28/2022	2/28/2022 11:59:59 PM
8189	1/19/2022	02/28/2022	2/28/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	04/30/2022	4/30/2022 11:59:59 PM
8193	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	10/31/2022	10/31/2022 11:59:59 PM
8206	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8207	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

會使用 monthsend() 函數在圖表物件中建立「bi_monthly_end」欄位作為量值。提供的第一個引數為 2，將該年分為兩個月區段。第二個引數識別正在評估哪個欄位。

具有兩個月區段的 monthsend 函數圖表。



交易 8195 發生在 5 月 22 日。monthsend() 函數原本將該年分為兩個月區段。交易 8195 落在 5 月 1 日和 6 月 30 日之間的區段內。因此，該函數回傳此區段的第一毫秒 06/30/2022 11:59:59 PM。

範例 5 – 情境

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

在此範例中，資料集會載入到稱為「Employee_Expenses」的表格中。該表格含有下列欄位：

- 員工 ID
- 員工名稱
- 每個員工報銷的平均每日開支。

最終使用者希望圖表按員工 ID 和員工名稱顯示所選期間的剩餘期間的預估開支報銷。會計年度於 1 月開始。

載入指令碼

```
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
];
```

結果

載入資料並開啟新的工作表。

在載入指令碼開頭，已建立與變數輸入控制繫結的變數 `vPeriod`。

請執行下列動作：

1. 在資產面板中，按一下**自訂物件**。
2. 選取 **Qlik 儀表板搭售**，建立**變數輸入**物件。
3. 輸入圖表物件的標題。
4. 在**變數**之下，選取 **vPeriod** 作為名稱並設定物件以顯示為**下拉式清單**。
5. 在**值**之下，按一下**動態**值。輸入下列內容：
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`

建立新的表格和這些欄位作為維度：

- `employee_id`
- `employee_name`

若要計算累積的利息，建立此量值：

`=floor(monthsend($(vPeriod),today(1))-today(1))*avg_daily_claim`

此量值是動態性質，將會根據載入資料的日期產生不同的表格結果。

將量值的**數字格式**設定為**金錢**。

結果表格

employee_id	employee_name	=floor(monthsend($(vPeriod),today(1))-today(1))*avg_daily_claim
182	Mark	$1410.00
183	Deryck	$1175.00
184	Dexter	$1175.00
185	Sydney	$2538.00
186	Agatha	$1692.00

`monthsend()` 函數使用使用者輸入作為第一引數，並使用今日日期作為第二引數。這傳回使用者所選期間的結束日期。然後，運算式會以此結束日期減去今日日期，以傳回所選期間的剩餘天數。

然後按員工讓此值乘以平均每日開支報銷，以計算每個員工在此期間剩餘天數預期會有的預估報銷值。

monthsname

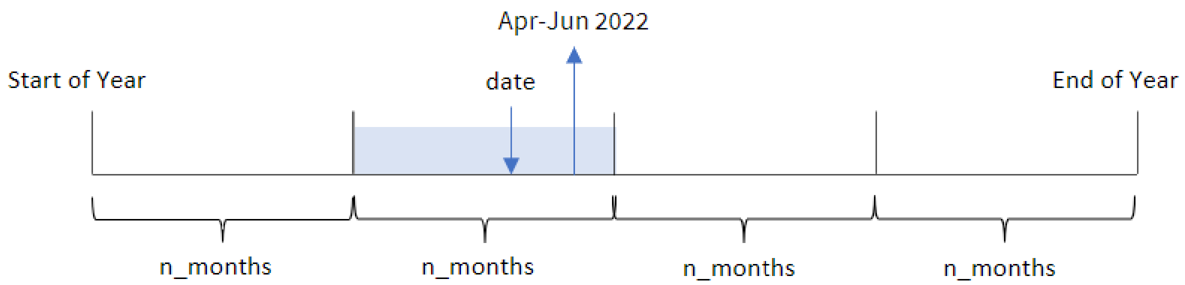
此函數會傳回一個顯示值，代表期間月份 (根據 **MonthNames** 指令碼變數進行格式設定) 和年度。基礎數值相當於包含基本日期的月、雙月、季度、四月期或半年的第一毫秒的時間戳記。

語法：

```
MonthsName(n_months, date[, period_no[, first_month_of_year]])
```

傳回的資料類型： 雙值

monthsname 函數的圖表



monthsname() 函數根據提供的 n_months 引數將該年分為幾個區段。然後這評估每個提供的 date 所屬的區段，並傳回該區段的開始和結束月份名稱以及該年。該函數也能夠從之前或之後的區段傳回這些邊界，以及重新定義哪個是該年的第一個月。

一年中的下列區段可作為 n_month 引數用於函數：

可能的 n_month 引數

期間	月數
月	1
兩個月	2
季	3
四個月	4
半年	6

引數

引數	描述
n_months	定義期間的月數。整數或解析為整數的運算式，必須是下列其中一項：1 (相當於 inmonth() 函數)、2 (雙月)、3 (相當於 inquarter() 函數)、4 (四月期) 或 6 (半年)。
date	要評估的時間戳記。

引數	描述
period_no	期間可以使用 **period_no** 位移，是一個整數或解譯為整數的運算式，其中值 0 表示包含 **base_date** 的期間。負值的 **period_no** 表示之前的期間，正值表示之後的期間。
first_month_of_year	如果要使用不起始於 1 月的 (會計) 年度，可在 **first_month_of_year** 中指定介於 2 和 12 之間的值。

什麼情況下使用

若您想要向使用者提供透過所選期間比較彙總的功能，則 `monthsname()` 函數很實用。例如，您可以提供輸入變數，以讓使用者按月、季或半年查看產品的總銷售額。

這些維度的建立方式可以是新增函數作為主要行事曆表格中的欄位，以在載入指令碼中建立，或直接在圖表中建立維度作為計算維度。

函數範例

範例	結果
`monthsname(4, '10/19/2013')`	傳回「Sep-Dec 2013」。因為在此範例和其他範例中， **SET Monthnames** 陳述式設為 Jan;Feb;Mar，以此類推。
`monthsname(4, '10/19/2013', -1)`	傳回「May-Aug 2013」。
`monthsname(4, '10/19/2013', 0, 2)`	傳回「Oct-Jan 2014」，因為該年指定以月份 2 開始。因此，四個月期間結束於下一年的第一個月。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 基本範例

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為 `Transactions` 的表格中。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 `bi_monthly_range`，這將交易組成兩個月區段，並為每個交易傳回該區段的邊界名稱。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

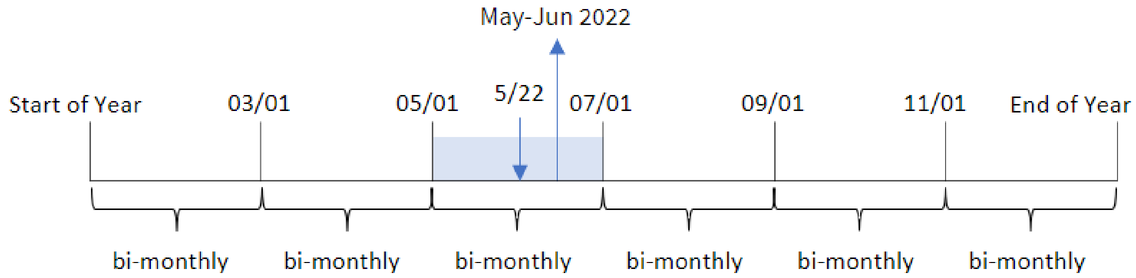
- date
- bi_monthly_range

結果表格

日期	bi_monthly_range
2/19/2022	2022 年 1 月至 2 月
3/7/2022	2022 年 3 月至 4 月
3/30/2022	2022 年 3 月至 4 月
4/5/2022	2022 年 3 月至 4 月
4/16/2022	2022 年 3 月至 4 月
5/1/2022	2022 年 5 月至 6 月
5/7/2022	2022 年 5 月至 6 月
5/22/2022	2022 年 5 月至 6 月
6/15/2022	2022 年 5 月至 6 月
6/26/2022	2022 年 5 月至 6 月
7/9/2022	2022 年 7 月至 8 月
7/22/2022	2022 年 7 月至 8 月
7/23/2022	2022 年 7 月至 8 月
7/27/2022	2022 年 7 月至 8 月
8/2/2022	2022 年 7 月至 8 月
8/8/2022	2022 年 7 月至 8 月
8/19/2022	2022 年 7 月至 8 月
9/26/2022	2022 年 9 月至 10 月
10/14/2022	2022 年 9 月至 10 月
10/29/2022	2022 年 9 月至 10 月

「`bi_monthly_range`」欄位使用 `monthsname()` 函數在前置 LOAD 陳述式中建立。提供的第一個引數為 2，將該年分為兩個月區段。第二個引數識別正在評估哪個欄位。

monthsname 函數的圖表，基本範例



交易 8195 發生在 5 月 22 日。monthsname() 函數原本將該年分為兩個月區段。交易 8195 落在 5 月 1 日和 6 月 30 日之間的區段內。因此，該函數會以 MonthNames 系統變數格式傳回這些月份以及年份，即 2022 年 5 月至 6 月。

範例 2 – period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的內嵌資料集和情境。
- 建立欄位 prev_bi_monthly_range，這將交易組成兩個月區段，並為每個交易傳回上一個區段的邊界名稱。

按需要在此新增其他文字，可附上清單等。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

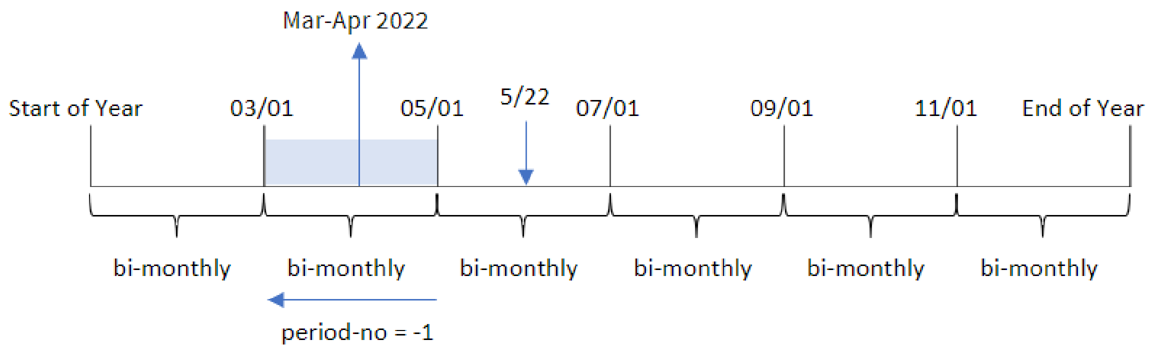
- date
- prev_bi_monthly_range

結果表格

日期	prev_bi_monthly_range
2/19/2022	2021 年 11 月至 12 月
3/7/2022	2022 年 1 月至 2 月
3/30/2022	2022 年 1 月至 2 月
4/5/2022	2022 年 1 月至 2 月
4/16/2022	2022 年 1 月至 2 月
5/1/2022	2022 年 3 月至 4 月
5/7/2022	2022 年 3 月至 4 月
5/22/2022	2022 年 3 月至 4 月
6/15/2022	2022 年 3 月至 4 月
6/26/2022	2022 年 3 月至 4 月
7/9/2022	2022 年 5 月至 6 月
7/22/2022	2022 年 5 月至 6 月
7/23/2022	2022 年 5 月至 6 月
7/27/2022	2022 年 5 月至 6 月
8/2/2022	2022 年 5 月至 6 月
8/8/2022	2022 年 5 月至 6 月
8/19/2022	2022 年 5 月至 6 月
9/26/2022	2022 年 7 月至 8 月
10/14/2022	2022 年 7 月至 8 月
10/29/2022	2022 年 7 月至 8 月

在此範例中,-1 作為 monthsname() 函數中的 period_no 引數使用。原本將年分為兩個月區段後,該函數就會針對交易發生時間傳回上一個區段邊界。

monthsname 函數的圖表，period_no 範例



交易 8195 發生在 5 月 和 6 月 之間的區段內。因此，上一個兩個月區段為 3 月 1 日和 4 月 30 日之間，因此該函數會傳回 2022 年 3 月至 4 月。

範例 3 – first_month_of_year

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的內嵌資料集和情境。
- 建立不同的欄位 `bi_monthly_range`，這將交易組成兩個月區段，並為每個交易傳回區段邊界。

不過，在此範例中，我們也需要將 4 月設定為會計年度的第一個月。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- bi_monthly_range

結果表格

日期	bi_monthly_range
2/19/2022	2021 年 2 月至 3 月
3/7/2022	2021 年 2 月至 3 月
3/30/2022	2021 年 2 月至 3 月
4/5/2022	2022 年 4 月至 5 月
4/16/2022	2022 年 4 月至 5 月
5/1/2022	2022 年 4 月至 5 月
5/7/2022	2022 年 4 月至 5 月
5/22/2022	2022 年 4 月至 5 月
6/15/2022	2022 年 6 月至 7 月
6/26/2022	2022 年 6 月至 7 月
7/9/2022	2022 年 6 月至 7 月
7/22/2022	2022 年 6 月至 7 月
7/23/2022	2022 年 6 月至 7 月
7/27/2022	2022 年 6 月至 7 月
8/2/2022	2022 年 8 月至 9 月

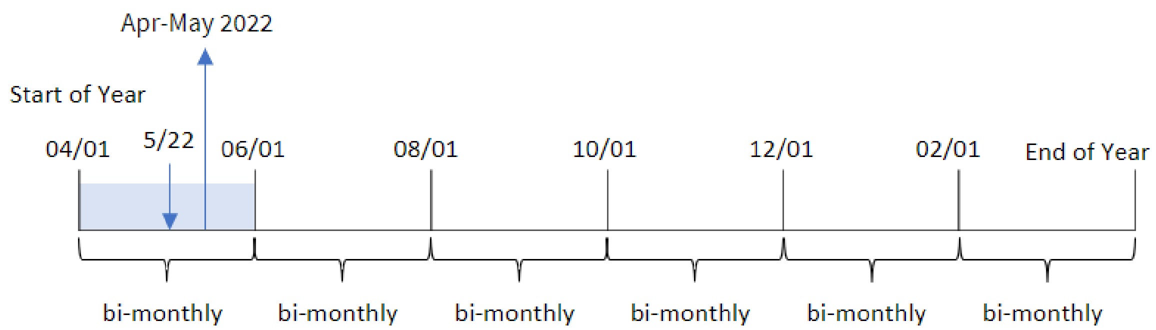
日期	bi_monthly_range
8/8/2022	2022 年 8 月至 9 月
8/19/2022	2022 年 8 月至 9 月
9/26/2022	2022 年 8 月至 9 月
10/14/2022	2022 年 10 月至 11 月
10/29/2022	2022 年 10 月至 11 月

若使用 4 作為 `monthsname()` 函數中的 `first_month_of_year` 引數，該函數會讓該年從 4 月 1 日開始，然後將該年分為兩個月區段：4 月至 5 月、6 月至 7 月、8 月至 9 月、10 月至 11 月、12 月至 1 月、2 月至 3 月。

結果的段落文字。

交易 8195 發生在 5 月 22 日並落在 4 月 1 日和 5 月 31 日之間的區段內。因此，該函數會傳回 2022 年 4 月至 5 月。

monthsname 函數的圖表，first_month_of_year 範例



範例 4 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的內嵌資料集和情境。不過，在此範例中，不變的資料集會載入到應用程式中。會建立將交易組成兩個月區段並為每個交易傳回區段邊界的計算，作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`date`。

建立下列量值：

```
=monthsname(2,date)
```

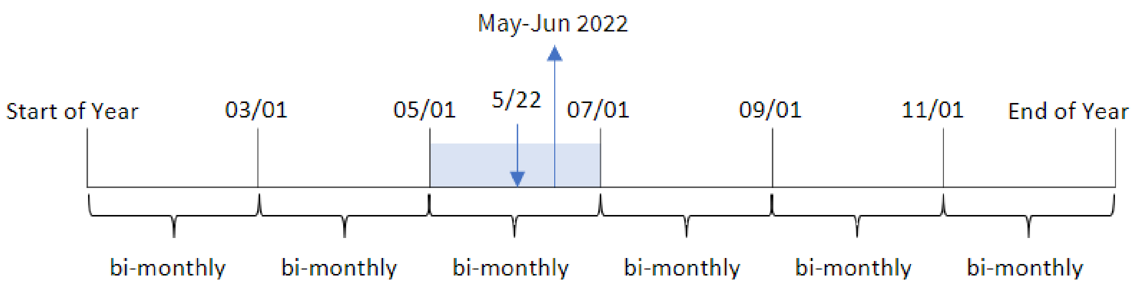
結果表格

日期	=monthsname(2,date)
2/19/2022	2022 年 1 月至 2 月
3/7/2022	2022 年 3 月至 4 月
3/30/2022	2022 年 3 月至 4 月
4/5/2022	2022 年 3 月至 4 月
4/16/2022	2022 年 3 月至 4 月
5/1/2022	2022 年 5 月至 6 月
5/7/2022	2022 年 5 月至 6 月
5/22/2022	2022 年 5 月至 6 月
6/15/2022	2022 年 5 月至 6 月

日期	=monthsname(2,date)
6/26/2022	2022 年 5 月至 6 月
7/9/2022	2022 年 7 月至 8 月
7/22/2022	2022 年 7 月至 8 月
7/23/2022	2022 年 7 月至 8 月
7/27/2022	2022 年 7 月至 8 月
8/2/2022	2022 年 7 月至 8 月
8/8/2022	2022 年 7 月至 8 月
8/19/2022	2022 年 7 月至 8 月
9/26/2022	2022 年 9 月至 10 月
10/14/2022	2022 年 9 月至 10 月
10/29/2022	2022 年 9 月至 10 月

會使用 `monthsname()` 函數在圖表物件中建立 `bi_monthly_range` 欄位作為量值。提供的第一個引數為 2，將該年分為兩個月區段。第二個引數識別正在評估哪個欄位。

monthsname 函數的圖表，圖表物件範例



交易 8195 發生在 5 月 22 日。`monthsname()` 函數原本將該年分為兩個月區段。交易 8195 落在 5 月 1 日和 6 月 30 日之間的區段內。因此，該函數會以 `MonthNames` 系統變數格式傳回這些月份以及年份，即 2022 年 5 月至 6 月。

範例 5 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含 2022 年交易的資料集，這載入到稱為 `Transactions` 的表格中。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。

最終使用者希望圖表物件按自己選擇的期間顯示總銷售額。即使此維度無法用於資料模型，這仍可以達成，方法是使用 `monthsname()` 函數作為透過變數輸入控制所動態修改的計算維度。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。

在載入指令碼開頭，已建立將會與變數輸入控制繫結的變數 (`vPeriod`)。接下來，設定變數作為工作表中的自訂物件。

請執行下列動作：

1. 在資產面板中，按一下**自訂物件**。
2. 選取 **Qlik 儀表板搭售**，並建立**變數輸入**物件。
3. 輸入圖表物件的標題。

4. 在**變數**之下，選取 **vPeriod** 作為名稱並設定物件以顯示為**下拉式清單**。
5. 在**值**之下，設定物件以使用動態值。輸入下列內容：
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'`

接下來，建立結果表格。

請執行下列動作：

1. 建立新的表格並新增下列計算維度：
   `=monthsname($(vPeriod),date)`
2. 新增此量值以計算總銷售額：
   `=sum(amount)`
3. 將量值的**數字格式**設定為**金錢**。按一下 ✔ **完成編輯**。您現在可以調整變數物件中的時間區段，以修改顯示在表格中的資料。

這是結果表格在選取 `tertial` 選項時看起來的樣子：

結果表格

monthsname($(vPeriod),date)	=sum(amount)
2022 年 1 月至 4 月	253.89
2022 年 5 月至 8 月	713.58
2022 年 9 月至 12 月	248.12

monthsstart

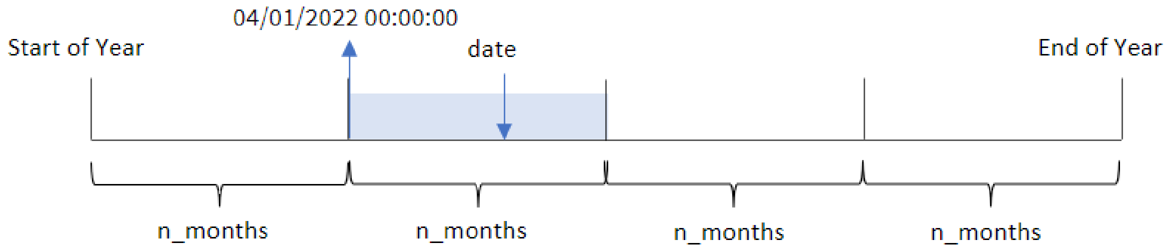
此函數傳回的值相當於包含基本日期的月、雙月、季度、四月期或半年的第一毫秒的時間戳記。還可以尋找前一個或後一個時間期間的時間戳記。預設的輸出格式為指令碼中所設定的 **DateFormat**。

語法：

```
MonthsStart(n_months, date[, period_no [, first_month_of_year]])
```

傳回的資料類型： 雙值

monthsstart() 函數的圖表



monthsstart() 函數根據提供的 n_months 引數將該年分為幾個區段。然後這會評估提供的每個日期落在哪些區段，並以日期格式傳回該區段的第一毫秒。該函數也能夠從之前或之後的區段傳回開始時間戳記，以及重新定義哪個是該年的第一個月。

一年中的下列區段可作為 n_month 引數用於函數：

可能的 n_month 引數

期間	月數
月	1
兩個月	2
季	3
四個月	4
半年	6

引數

引數	描述
n_months	定義期間的月數。整數或解析為整數的運算式，必須是下列其中一項：1 (相當於 inmonth() 函數)、2 (雙月)、3 (相當於 inquarter() 函數)、4 (四月期) 或 6 (半年)。
date	要評估的時間戳記。
period_no	期間可以使用 **period_no** 位移，是一個整數或解譯為整數的運算式，其中值 0 表示包含 **base_date** 的期間。負值的 **period_no** 表示之前的期間，正值表示之後的期間。
first_month_of_year	如果要使用不起始於 1 月的 (會計) 年度，可在 **first_month_of_year** 中指定介於 2 和 12 之間的值。

什麼情況下使用

monthsstart() 函數通常在使用者想要計算以使用一個期間中尚未發生的部分時，作為運算式的一部分使用。例如，這可以用來提供輸入變數，以讓使用計算該月、該季或半年期間目前已累積的總利息。

函數範例

範例	結果
monthsstart(4, '10/19/2013')	傳回 09/01/2013。
monthsstart(4, '10/19/2013, -1)	傳回 05/01/2013。
monthsstart(4, '10/19/2013', 0, 2)	傳回 10/01/2013，因為年開始變成月份 2。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以

符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為 `Transactions` 的表格中。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 `bi_monthly_start`，這將交易組成兩個月區段，並為每個交易傳回該區段的開始時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

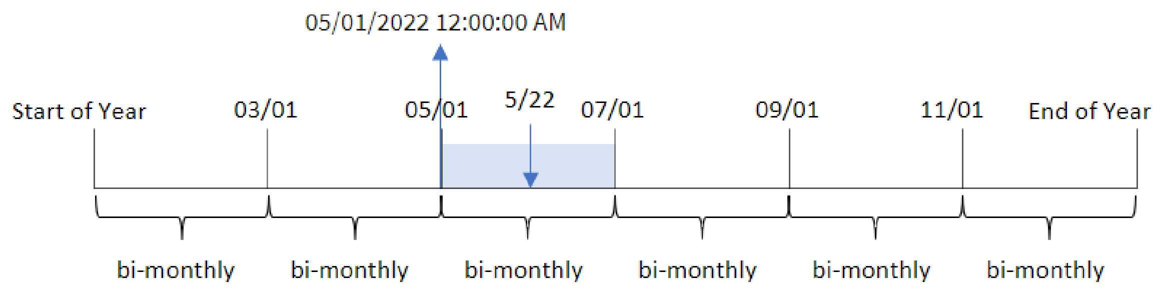
- date
- bi_monthly_start
- bi_monthly_start_timestamp

結果表格

日期	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

「bi_monthly_start」欄位使用 monthsstart() 函數在前置 LOAD 陳述式中建立。提供的第一個引數為 2，將該年分為兩個月區段。第二個引數識別正在評估哪個欄位。

monthsstart() 函數的圖表，無其他引數的範例



交易 8195 發生在 5 月 22 日。monthsstart() 函數原本將該年分為兩個月區段。交易 8195 落在 5 月 1 日和 6 月 30 日之間的區段內。因此，該函數傳回此區段的第一毫秒 2022 年 5 月 1 日 12:00:00 AM。

範例 2 – period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 建立欄位「prev_bi_monthly_start」，這傳回交易發生之前的兩個月區段的第一毫秒。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- prev_bi_monthly_start
- prev_bi_monthly_start_timestamp

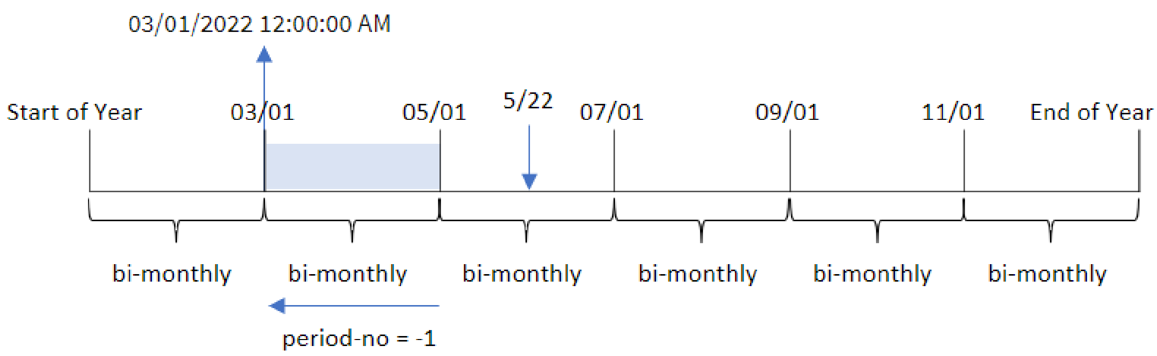
結果表格

日期	prev_bi_monthly_start	prev_bi_monthly_start_timestamp
2/19/2022	11/01/2021	11/1/2021 12:00:00 AM
3/7/2022	01/01/2022	1/1/2022 12:00:00 AM
3/30/2022	01/01/2022	1/1/2022 12:00:00 AM
4/5/2022	01/01/2022	1/1/2022 12:00:00 AM
4/16/2022	01/01/2022	1/1/2022 12:00:00 AM
5/1/2022	03/01/2022	3/1/2022 12:00:00 AM
5/7/2022	03/01/2022	3/1/2022 12:00:00 AM
5/22/2022	03/01/2022	3/1/2022 12:00:00 AM
6/15/2022	03/01/2022	3/1/2022 12:00:00 AM
6/26/2022	03/01/2022	3/1/2022 12:00:00 AM
7/9/2022	05/01/2022	5/1/2022 12:00:00 AM
7/22/2022	05/01/2022	5/1/2022 12:00:00 AM
7/23/2022	05/01/2022	5/1/2022 12:00:00 AM
7/27/2022	05/01/2022	5/1/2022 12:00:00 AM
8/2/2022	05/01/2022	5/1/2022 12:00:00 AM

日期	prev_bi_monthly_start	prev_bi_monthly_start_timestamp
8/8/2022	05/01/2022	5/1/2022 12:00:00 AM
8/19/2022	05/01/2022	5/1/2022 12:00:00 AM
9/26/2022	07/01/2022	7/1/2022 12:00:00 AM
10/14/2022	07/01/2022	7/1/2022 12:00:00 AM
10/29/2022	07/01/2022	7/1/2022 12:00:00 AM

透過使用 -1 作為 `monthsstart()` 函數中的 `period_no` 引數，原本將年分為兩個月區段後，該函數就會針對交易發生時間傳回上一個兩個月區段的第一毫秒。

`monthsstart()` 函數的圖表，period_no 範例



交易 8195 發生在 5 月 和 6 月 之間的區段內。因此，上一個兩個月區段介於 3 月 1 日和 4 月 30 日之間，所以該函數會傳回此區段的第一毫秒 2022 年 3 月 1 日 12:00:00 AM。

範例 3 – first_month_of_year

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 建立欄位 `bi_monthly_start`，這將交易組成兩個月區段，並為每個交易傳回該設定的開始時間戳記。

不過，在此範例中，我們也需要將 4 月設定為會計年度的第一個月。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- bi_monthly_start
- bi_monthly_start_timestamp

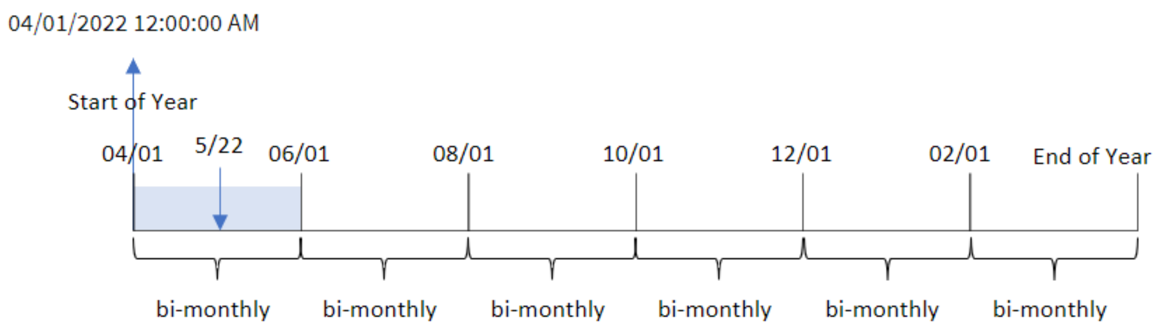
結果表格

日期	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

日期	bi_monthly_start	bi_monthly_start_timestamp
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

若使用 4 作為 `monthsstart()` 函數中的 `first_month_of_year` 引數，該函數會讓該年從 4 月 1 日開始，然後將該年分為兩個月區段：4 月至 5 月、6 月至 7 月、8 月至 9 月、10 月至 11 月、12 月至 1 月、2 月至 3 月。

`monthsstart()` 函數的圖表，first_month_of_year 範例



交易 8195 發生在 5 月 22 日並落在 4 月 1 日和 5 月 31 日之間的區段內。因此，該函數傳回此區段的第一毫秒 2022 年 4 月 1 日 12:00:00 AM。

範例 4 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。

不過,在此範例中,不變的資料集會載入到應用程式中。會建立將交易組成兩個月區段並為每個交易傳回設定之開始時間戳記的計算,作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度:`date`。

建立下列量值:

```
=monthsstart(2,date)
```

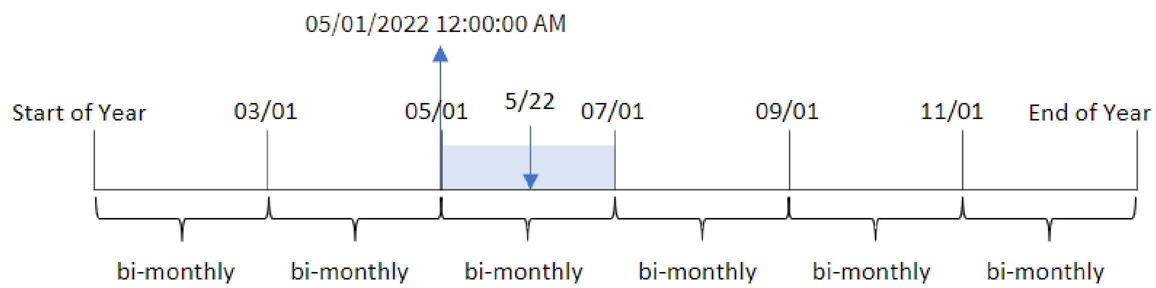
```
=timestamp(monthsstart(2,date))
```

這些計算將擷取每個交易發生之兩個月區段的開始時間戳記。

結果表格

日期	=monthsstart(2,date)	=timestamp(monthsstart(2,date))
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
10/14/2022	09/01/2022	9/1/2022 12:00:00 AM
10/29/2022	09/01/2022	9/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
5/1/2022	05/01/2022	5/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/22/2022	05/01/2022	5/1/2022 12:00:00 AM
6/15/2022	05/01/2022	5/1/2022 12:00:00 AM
6/26/2022	05/01/2022	5/1/2022 12:00:00 AM
3/7/2022	03/01/2022	3/1/2022 12:00:00 AM
3/30/2022	03/01/2022	3/1/2022 12:00:00 AM
4/5/2022	03/01/2022	3/1/2022 12:00:00 AM
4/16/2022	03/01/2022	3/1/2022 12:00:00 AM
2/19/2022	01/01/2022	1/1/2021 12:00:00 AM

monthsstart() 函數的圖表，圖表物件範例



交易 8195 發生在 5 月 22 日。monthsstart() 函數原本將該年分為兩個月區段。交易 8195 落在 5 月 1 日和 6 月 30 日之間的區段內。因此，該函數傳回此區段的第一毫秒 05/01/2022 12:00:00 AM。

範例 5 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 資料集包含一系列貸款餘額，其位於稱為 Loans 的表格。
- 資料包含貸款 ID、月份開始餘額以及每年對每項貸款收取的單利率。

最終使用者希望圖表物件按貸款 ID 顯示所選期間每筆貸款已累積的目前利息。會計年度於 1 月開始。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。

在載入指令碼開頭，已建立將會與變數輸入控制繫結的變數 (vPeriod)。接下來，設定變數作為工作表中的自訂物件。

請執行下列動作：

1. 在資產面板中，按一下**自訂物件**。
2. 選取 **Qlik 儀表板搭售**，並建立**變數輸入**物件。
3. 輸入圖表物件的標題。
4. 在**變數**之下，選取 **vPeriod** 作為名稱並設定物件以顯示為**下拉式清單**。

5. 在**值**之下，設定物件以使用動態值。輸入下列內容：
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'`

接下來，建立結果表格。

請執行下列動作：

1. 建立新的表格。新增下列欄位作為維度。
   - `employee_id`
   - `employee_name`
2. 建立量值以計算累積的利息：
   `=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)`
3. 將量值的**數字格式**設定為**金錢**。按一下 ✔ **完成編輯**。您現在可以調整變數物件中的時間區段，以修改顯示在表格中的資料。

這是結果表格在選取 `month` 期間選項時看起來的樣子：

結果表格

loan_id	start_balance	=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)
8188	$10000.00	$7.95
8189	$15000.00	$67.93
8190	$17500.00	$33.37
8191	$21000.00	$56.73
8192	$90000.00	$600.66

`monthsstart()` 函數使用使用者輸入作為第一引數並使用今日日期作為第二引數，傳回使用者所選期間的開始日期。以目前日期減去該結果後，運算式會傳回此期間目前已經過的天數。

然後此值乘以利率並除以 365，以傳回此期間產生的有效利率。然後該結果乘以貸款的開始餘額，以傳回此期間目前已累積的利息。

monthstart

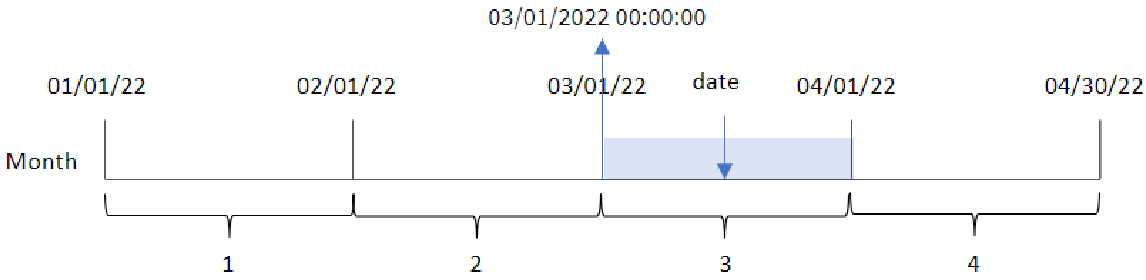
此函數傳回的值相當於包含 **date** 的月份第一天、第一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 **DateFormat**。

語法：

```
MonthStart(date[, period_no])
```

傳回的資料類型: 雙值

monthstart() 函數的圖表



monthstart() 函數判定日期落在哪個月。然後以日期格式傳回該月第一毫秒的時間戳記。

引數

引數	描述
date	要評估的時間戳記。
period_no	**period_no** 是一個整數,如果為 0 或忽略,則表示包含 **date** 的月份。負值的 **period_no** 表示之前的月份,正值表示之後的月份。

什麼情況下使用

monthstart() 函數通常在使用者想要計算使用一個月中已經過的部分時,作為運算式的一部分使用。例如,這可用於計算截至特定日期之月份內已累積的利息。

函數範例

範例	結果
monthstart('10/19/2001')	傳回 10/01/2001。
monthstart('10/19/2001', -1)	傳回 09/01/2001。

區域設定

除非另有說明,否則此主題中的範例皆使用下列日期格式:MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素,您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式,以滿足您的需求。或者,您可以在載入指令碼中變更格式,以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典,資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2022 年交易的資料集, 這載入到稱為 Transactions 的表格中。
- 以 DateFormat 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 start_of_month, 這傳回交易發生該月開始的時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- start_of_month
- start_of_month_timestamp

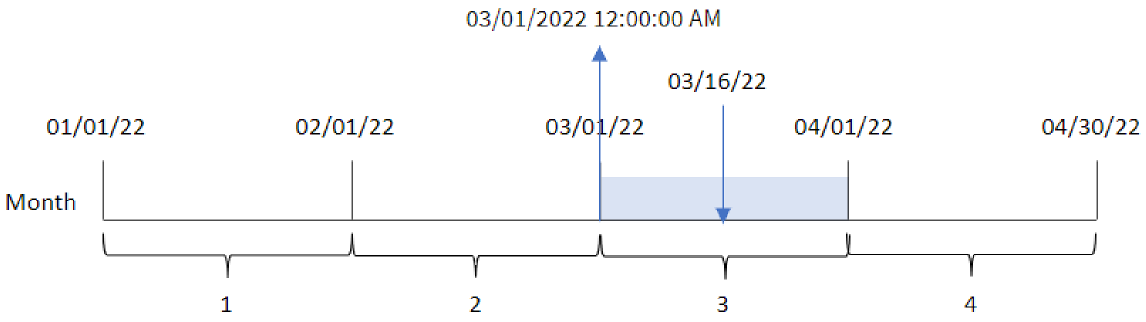
結果表格

日期	start_of_month	start_of_month_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM
2/5/2022	02/01/2022	2/1/2022 12:00:00 AM
2/28/2022	02/01/2022	2/1/2022 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/16/2022	05/01/2022	5/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	07/01/2022	6/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	8/1/2022 12:00:00 AM
8/19/2022	08/01/2022	8/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM

start_of_month 欄位在前置 LOAD 陳述式中的建立方式是使用 monthstart() 函數並傳遞日期欄位，作為函數的引數。

monthstart() 函數識別日期值落在哪個月，並傳回該月第一毫秒的時間戳記。

monthstart() 函數的圖表，無其他引數的範例



交易 8192 發生於 3 月 16 日。monthstart() 函數傳回該月的第一毫秒，亦即 3 月 1 日上午 12:00:00。

範例 2 – period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 建立欄位 previous_month_start，這傳回交易發生的前一個月開始的時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- previous_month_start
- previous_month_start_timestamp

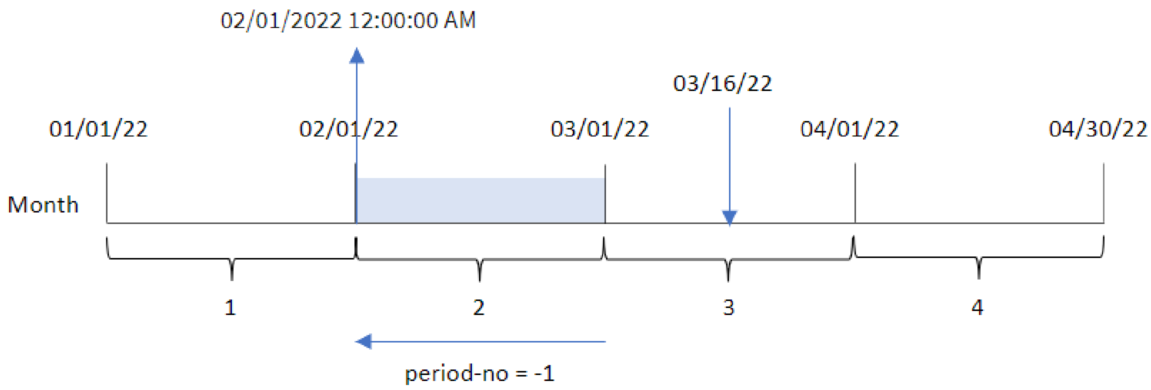
結果表格

日期	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

日期	previous_month_start	previous_month_start_timestamp
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

在此例中，因為 period_no 的值 -1 已作為 monthstart() 函數中的偏移引數使用，所以該函數首先會識別交易發生的月份。然後這會往前偏移一個月並識別該月的第一毫秒。

monthstart() 函數的圖表，period_no 範例



交易 8192 發生於 3 月 16 日。monthstart() 函數識別交易發生的前一個月為 2 月。然後這會傳回該月的第一毫秒 2 月 1 日 12:00:00 AM。

範例 3 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。

不過，在此範例中，不變的資料集會載入到應用程式中。會建立傳回交易發生月份開始時間戳記的計算，作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`date`。

若要計算交易發生的該月開始日期，建立下列量值：

- `=monthstart(date)`
- `=timestamp(monthstart(date))`

結果表格

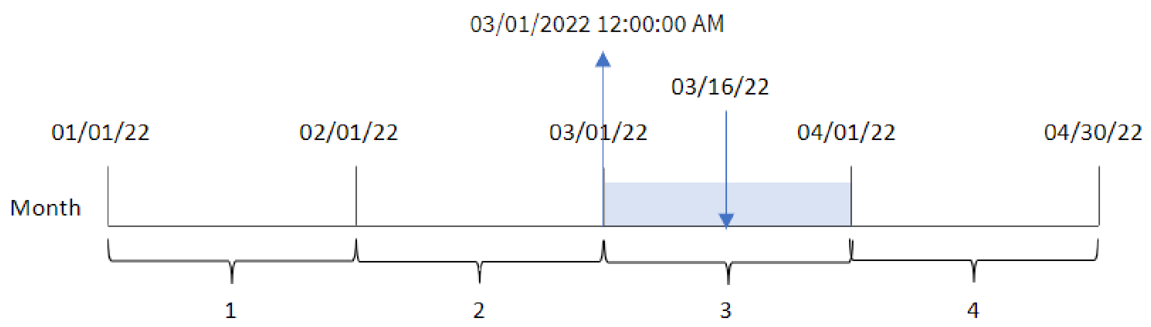
日期	=monthstart(date)	=timestamp(monthstart(date))
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	8/1/2022 12:00:00 AM
8/19/2022	08/01/2022	8/1/2022 12:00:00 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

日期	=monthstart(date)	=timestamp(monthstart(date))
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	06/01/2022	6/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/16/2022	05/01/2022	5/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
2/5/2022	02/01/2022	2/1/2022 12:00:00 AM
2/28/2022	02/01/2022	2/1/2022 12:00:00 AM
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM

`start_of_month` 量值在圖表物件中的建立方式是使用 `monthstart()` 函數並傳遞日期欄位，作為函數的引數。

`monthstart()` 函數識別日期值落在哪個月，並傳回該月第一毫秒的時間戳記。

`monthstart()` 函數的圖表，圖表物件範例



交易 8192 發生於 3 月 16 日。`monthstart()` 函數識別交易發生在 3 月，並傳回該月的第一毫秒，亦即 3 月 1 日上午 12:00:00。

範例 4 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 資料集包含一系列貸款餘額，其位於稱為 Loans 的表格。
- 資料包含貸款 ID、月份開始餘額以及每年對每項貸款收取的單利率。

最終使用者希望圖表物件按貸款 ID 顯示月初至今每筆貸款已累積的目前利息。

載入指令碼

```
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
];
```

結果

請執行下列動作：

1. 載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：
   - loan_id
   - start_balance
2. 接下來，建立量值以計算累積的利息：
   =start_balance*(rate*(today(1)-monthstart(today(1)))/365)
3. 將量值的**數字格式**設定為**金錢**。

結果表格

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

monthstart() 函數使用今日日期作為其唯一引數，傳回目前月份的開始日期。以目前日期減去該結果後，運算式會傳回此月目前已經過的天數。

然後此值乘以利率並除以 365，以傳回此期間產生的有效利率。然後該結果乘以貸款的開始餘額，以傳回本月目前已累積的利息。

networkdays

networkdays 函數會傳回介於 **start_date** 和 **end_date** 之間 (可包含) 的工作日數 (星期一至星期五), 並考慮到所有選用的列出 **holiday**。

語法:

```
networkdays (start_date, end_date [, holiday])
```

傳回的資料類型: 整數

行事曆圖表顯示 networkdays 函數傳回的日期範圍

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			

`networkdays` 函數具有下列限制:

- 沒有方法可修改工作日。換言之, 沒有方式可為工作日不是星期一至星期五的相關地區或情境修改函數。
- `holiday` 參數必須是字串常數。不接受運算式。

引數

引數	描述
start_date	要評估的開始日期。
end_date	要評估的結束日期。

引數	描述
holiday	要從工作日中排除的假期。假日以字串常數日期表示。您可以指定多個假日日期，以逗點分隔。 **範例：** **'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'**

什麼情況下使用

networkdays() 函數通常在使用者想要計算以使用發生在兩個日期之間的工作天數時，作為運算式的一部分使用。例如，若使用者想要計算員工在 PAYE (即收即付) 合約賺取的總薪資。

函數範例

範例	結果
networkdays ('12/19/2013', '01/07/2014')	傳回 14。此範例不會考慮假日。
networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013')	傳回 12。此範例會將假日 12/25/2013 至 12/26/2013 納入考量。
networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014')	傳回 10。此範例將兩個假日期間納入考量。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 基本範例

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 資料集包含專案 ID、開始日期和結束日期。此資訊載入到稱為 Projects 的表格。
- 以 DateFormat 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立附加欄位 net_work_days，以計算每個專案涉及的工作天數。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- start_date
- end_date
- net_work_days

結果表格

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

因為沒有已排程的假期 (這已存在於 networkdays() 函數的第三引數)，函數會從 end_date 減去 start_date 以及所有週末，以計算兩個日期之間的工作天數。

醒目提示專案 5 工作日的行事曆圖表 (無假期)

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			

以上行事曆以視覺方式概述 `id` 為 5 的專案。專案 5 開始於 2022 年 8 月10 日星期三，並結束於 2022 年 8 月 26 日。在忽略所有星期六和星期日的情況下，之間有 13 個工作日，並包含這兩個日期。

範例 2 – 單一假期

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與上一個範例相同的資料集和情境。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立附加欄位 `net_work_days`，以計算每個專案涉及的工作天數。

在此範例中，有一天排程於 2022 年 8 月19 日的假期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- id
- start_date
- end_date
- net_work_days

結果表格

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

輸入單一排程假期作為 `networkdays()` 函數中的第三引數。

醒目提示專案 5 工作日的行事曆圖表 (單一假期)

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			

以上行事曆以視覺方式概述專案 5，呈現此調整以納入假期。此假期發生在專案 5 期間 2022 年 8 月 19 日星期五。因此，專案 5 的總 `net_work_days` 值減少一天，從 13 天變成 12 天。

範例 3 – 多個假期

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立附加欄位 `net_work_days`，以計算每個專案涉及的工作天數。

不過，在此範例中，有四個排程於 2022 年 8 月 18 日至 8 月 21 日的假期。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- start_date
- end_date
- net_work_days

結果表格

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

從 `networkdays()` 函數中的第三引數開始，輸入四個已排程的假期作為以逗號分隔的清單。

醒目提示專案 5 工作日的行事曆圖表 (多個假期)

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			

以上行事曆以視覺方式概述專案 5, 呈現此調整以納入這些假期。此排程假期期間發生在專案 5 期間, 其中兩天發生在星期四和星期五。因此, 專案 5 的總 `net_work_days` 值從 13 天減為 11 天。

範例 4 – 單一假期

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 與第一個範例相同的資料集和情境。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。

有一天排程於 2022 年 8 月19 日的假期。

不過, 在此範例中, 不變的資料集會載入到應用程式中。會計算 `net_work_days` 欄位作為圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- id
- start_date
- end_date

建立下列量值:

```
= networkdays(start_date,end_date,'08/19/2022')
```

結果表格

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

輸入單一排程假期作為 `networkdays()` 函數中的第三引數。

行事曆圖表顯示網路工作日並具有單一假期 (圖表物件)

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			

以上行事曆以視覺方式概述專案 5，呈現此調整以納入假期。此假期發生在專案 5 期間 2022 年 8 月 19 日星期五。因此，專案 5 的總 `net_work_days` 值減少一天，從 13 天變成 12 天。

now

此函數會傳回目前時間的時間戳記。該函數會以 **TimeStamp** 系統變數格式傳回值。預設 **timer_mode** 值為 1。

語法：

```
now([ timer_mode])
```

傳回的資料類型： 雙值

`now()` 函數可用於載入指令碼或圖表物件中。

引數

引數	描述
timer_mode	可以具有下列值: 0 (最後完成資料載入的時間) 1 (函數叫用的時間) 2 (開啟應用程式的時間) *如果您在資料載入指令碼中使用函數,則 **timer_mode=0** 將產生最後完成資料載入的時間,並且 **timer_mode=1** 將提供目前資料載入中函數呼叫的時間。*

now() 函數有高效能影響,若該函數用於表格的運算式內,可能會造成捲動問題。若這樣的使用方式沒有嚴格的必要性,建議改用 today() 函數。若在版面配置中需要使用 now(),建議在可行情況下使用非預設設定 now(0) 或 now(2),因為這不需要常數重新計算

什麼情況下使用

now() 函數常用於當作運算式中的元件。例如,這可用來計算產品生命週期的剩餘時間。若運算式需要使用一天中的部分,會使用 now() 函數而非 today() 函數。

下表提供 now() 函數傳回結果的解釋,並提供 timer_mode 引數的不同數值:

函數範例

timer_mode 數值	用於載入指令碼的結果	用於圖表物件的結果
0	以 TimeStamp 系統變數格式傳回最新資料載入前,最後一筆成功資料載入的時間戳記。	以 TimeStamp 系統變數格式傳回最新資料載入的時間戳記。
1	以 TimeStamp 系統變數格式傳回最新資料載入的時間戳記。	以 TimeStamp 系統變數格式傳回函數呼叫的時間戳記。
2	以 TimeStamp 系統變數格式傳回使用者應用程式工作階段開始時的時間戳記。這不會更新內容,除非使用者重新載入指令碼。	以 TimeStamp 系統變數格式傳回使用者應用程式工作階段開始時的時間戳記。新工作階段開始或應用程式資料重新載入時,即會重新整理內容。

區域設定

除非另有說明,否則此主題中的範例皆使用下列日期格式:MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素,您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式,以滿足您的需求。或者,您可以在載入指令碼中變更格式,以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 使用載入指令碼產生物件

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

此範例使用 `now()` 函數建立三項變數。每個變數都會使用其中一個 `timer_mode` 個選項示範其效果。

對於示範用途的變數，載入指令碼並在短時間後第二次載入指令碼。這會造成 `now(0)` 和 `now(1)` 變數顯示不同數值，藉此正確示範其用途。

載入指令碼

```
LET vPreviousDataLoad = now(0);
LET vCurrentDataLoad = now(1);
LET vApplicationOpened = now(2);
```

結果

第二次載入資料後，使用以下指示建立三個文字方塊。

首先，為先前載入的資料建立文字方塊。

請執行下列動作：

1. 使用**文字與影像**圖表物件，建立文字方塊。
2. 將下列量值新增至物件：
   `=vPreviousDataLoad`
3. 在**外觀**之下，選取 **Show titles** 並將標題「Previous Reload Time」(先前載入時間) 新增至物件。

接下來，為目前正在載入的資料建立文字方塊。

請執行下列動作：

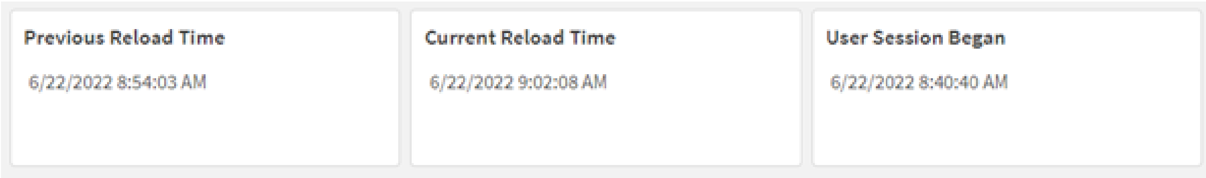
1. 使用**文字與影像**圖表物件，建立文字方塊。
2. 將下列量值新增至物件：
   `=vCurrentDataLoad`
3. 在**外觀**之下，選取 **Show titles** 並將標題「Current Reload Time」(目前載入時間) 新增至物件。

建立最後一個文字方塊，以顯示應用程式中的使用者工作階段何時開始。

請執行下列動作：

1. 使用**文字與影像**圖表物件，建立文字方塊。
2. 將下列量值新增至物件：
   `=vApplicationOpened`
3. 在**外觀**之下，選取 **Show titles** 並將標題「User Session Started」(使用者工作階段已開始) 新增至物件。

now() 載入指令碼變數



上方影像顯示每個建立變數的範例值。例如，數值可能如下所示：

- 先前的載入時間：6/22/2022 8:54:03 AM
- 目前的載入時間： 6/22/2022 9:02:08 AM
- 使用者工作階段已開始：6/22/2022 8:40:40 AM

範例 2 – 不使用載入指令碼產生物件

載入指令碼和圖表運算式

概覽

在此範例中，您將會使用 `now()` 函數建立三個圖表物件，而不需將任何變數或資料載入到應用程式中。每個圖表物件都會使用其中一個 `timer_mode` 選項示範其效果。

此範例沒有載入指令碼。

請執行下列動作：

1. 開啟資料載入編輯器。
2. 在不變更現有載入指令碼的情況下，按一下**載入資料**。
3. 在短時間後，第二次載入指令碼。

結果

第二次載入資料後，建立三個文字方塊。

首先，建立最新資料重新載入的文字方塊。

請執行下列動作：

1. 使用**文字與影像**圖表物件，建立文字方塊。
2. 新增下列量值。
   =now(0)
3. 在**外觀**之下，選取**顯示標題**並將標題「最新資料重新載入」新增至物件。

接下來，建立顯示目前時間的文字方塊。

請執行下列動作：

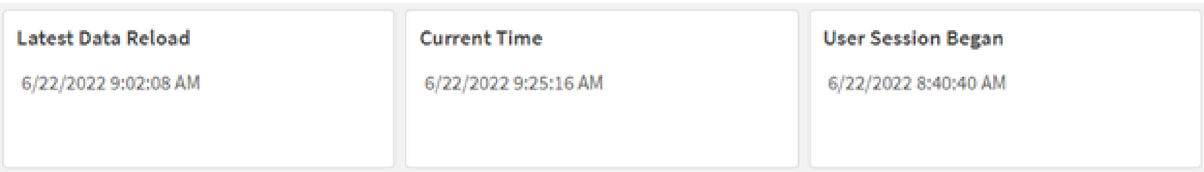
1. 使用**文字與影像**圖表物件，建立文字方塊。
2. 新增下列量值：
   =now(1)
3. 在**外觀**之下，選取**顯示標題**並將標題「目前時間」新增至物件。

建立最後一個文字方塊，以顯示應用程式中的使用者工作階段何時開始。

請執行下列動作：

1. 使用**文字與影像**圖表物件，建立文字方塊。
2. 新增下列量值：
   =now(2)
3. 在**外觀**之下，選取**顯示標題**並將標題「使用者工作階段已開始」新增至物件。

now() 圖表物件範例



上方影像顯示每個建立物件的範例值。例如，數值可能如下所示：

- 最新資料載入：6/22/2022 9:02:08 AM
- 目前時間：6/22/2022 9:25:16 AM
- 使用者工作階段已開始：6/22/2022 8:40:40 AM

「最新資料重新載入」圖表物件使用 0 的 `timer_mode` 值。這會傳回最後一次成功重新載入資料的時間戳記。

「目前時間」圖表物件使用 1 個 `timer_mode` 值。這會傳回根據系統時鐘顯示的目前時間。如果工作表或物件重新整理，即會更新此數值。

「使用者工作階段已開始」的圖表物件使用 2 的 `timer_mode` 值。這會傳回應用程式已開啟且使用者工作階段已開始的時間戳記。

範例 3 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 資料集包含加密貨幣挖掘操作的庫存,這載入到稱為 Inventory 的表格中。
- 具有下列欄位的資料:id、purchase_date 和 wph (每小時瓦數)。

使用者希望表格依 id 針對耗電量顯示每個採礦設備在該月至今產生的總成本。

此值應在每次圖表物件重新整理時更新。目前的電力成本為每 kWH $0.0678。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度:id。

建立下列量值：

```
=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
```

若圖表物件於 6/22/2022 10:39:05 AM 重新整理，會傳回下列結果：

結果表格

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

使用者希望物件結果在每次重新整理物件時重新整理。因此，會為運算式中的 `now()` 函數執行個體提供 `timer_mode` 引數。會使用 `now()` 函數作為 `monthstart()` 函數的時間戳記引數，以識別該月開始的時間戳記，並從透過 `now()` 函數識別的目前時間減去此時間戳記。這提供本月目前已經過的時間總長，以天數表示。

此值乘以 24 (一天中的時數)，然後乘以 `wph` 欄位中的值。

若要從每小時瓦數轉換為每小時千瓦，結果會先除以 1000，再乘以提供的 kWH 率。

quarterend

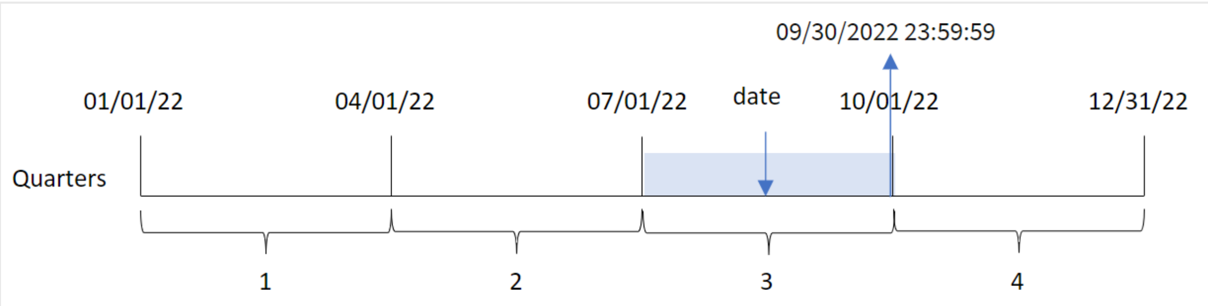
此函數傳回的值相當於包含 **date** 的季度的最後一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 **DateFormat**。

語法:

```
QuarterEnd(date[, period_no[, first_month_of_year]])
```

傳回的資料類型: 雙值

quarterend() 函數的圖表



quarterend() 函數判定日期落在哪一季。然後以日期格式傳回該季最後一個月的最後一毫秒的時間戳記。依照預設,該年的第一個月是 1 月。不過,您可以使用 quarterend() 函數中的 first_month_of_year 引數變更要將哪個月設為第一個月。

> *quarterend() 函數不考慮 FirstMonthOfYear 系統變數。該年從 1 月 1 日開始,除非使用 first_month_of_year 引數變更。*

什麼情況下使用

quarterend() 函數在您想要計算以使用一季中尚未發生的部分時,通常作為運算式的一部分使用。例如,若您想要計算該季期間尚未發生的總利息。

引數

引數	描述
date	要評估的時間戳記。
period_no	**period_no** 為整數,值 0 表示是包含 **date** 的季度。負值的 **period_no** 表示之前的季度,正值表示之後的季度。
first_month_of_year	如果要使用不起始於 1 月的 (會計) 年度,可在 **first_month_of_year** 中指定介於 2 和 12 之間的值。

您可以在 first_month_of_year 引數中使用下列值設定一年的第一個月:

first_month_of_year 值

月	值
二月	2
三月	3
四月	4
五月	5
六月	6
七月	7
八月	8
九月	9
十月	10
十一月	11
十二月	12

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
`quarterend('10/29/2005')`	傳回 12/31/2005 23:59:59。
`quarterend('10/29/2005', -1)`	傳回 09/30/2005 23:59:59。
`quarterend('10/29/2005', 0, 3)`	傳回 11/30/2005 23:59:59。

範例 1 - 基本範例

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2022 年交易的資料集, 這載入到稱為「Transactions」的表格中。
- 前置載入包含下列內容:
  - 設為「end_of_quarter」欄位並傳回交易發生該季結束之時間戳記的 quarterend() 函數。
  - 設為「end_of_quarter_timestamp」欄位並傳回所選季結束之確切時間戳記的 timestamp () 函數。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- id
- date
- end_of_quarter
- end_of_quarter_timestamp

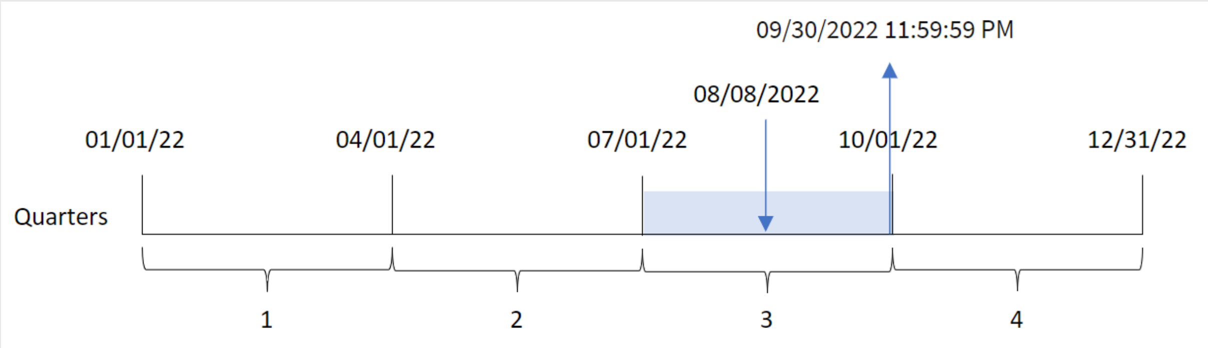
結果表格

id	日期	end_of_quarter	end_of_quarter_timestamp
8188	1/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8189	1/19/2022	03/31/2022	3/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	06/30/2022	6/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/16/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	09/30/2022	9/30/2022 11:59:59 PM
8199	7/22/2022	09/30/2022	9/30/2022 11:59:59 PM
8200	7/23/2022	09/30/2022	9/30/2022 11:59:59 PM
8201	7/27/2022	09/30/2022	9/30/2022 11:59:59 PM
8202	8/2/2022	09/30/2022	9/30/2022 11:59:59 PM
8203	8/8/2022	09/30/2022	9/30/2022 11:59:59 PM
8204	8/19/2022	09/30/2022	9/30/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	10/29/2022	12/31/2022	12/31/2022 11:59:59 PM

「end_of_quarter」欄位在前置 LOAD 陳述式中的建立方式是使用 quarterend() 函數並傳遞日期欄位,作為函數的引數。

quarterend() 函數最初識別日期值落在哪一季,然後傳回該季最後一毫秒的時間戳記。

識別交易 8203 之季度結束的 quarterend() 函數圖表



交易 8203 發生於 8 月 8 日。quarterend() 函數識別該交易發生在第三季，並傳回該季的最後一毫秒，即 9 月 30 日 11:59:59 PM。

範例 2 - period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為「Transactions」的表格中。
- 前置載入包含下列內容：
  - 設為「previous_quarter_end」欄位並傳回交易發生的前一季結束之時間戳記的 quarterend() 函數。
  - 設為「previous_end_of_quarter_timestamp」欄位並傳回交易發生的前一季結束之確切時間戳記的 timestamp() 函數。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- id
- date
- previous_quarter_end
- previous_quarter_end_timestamp

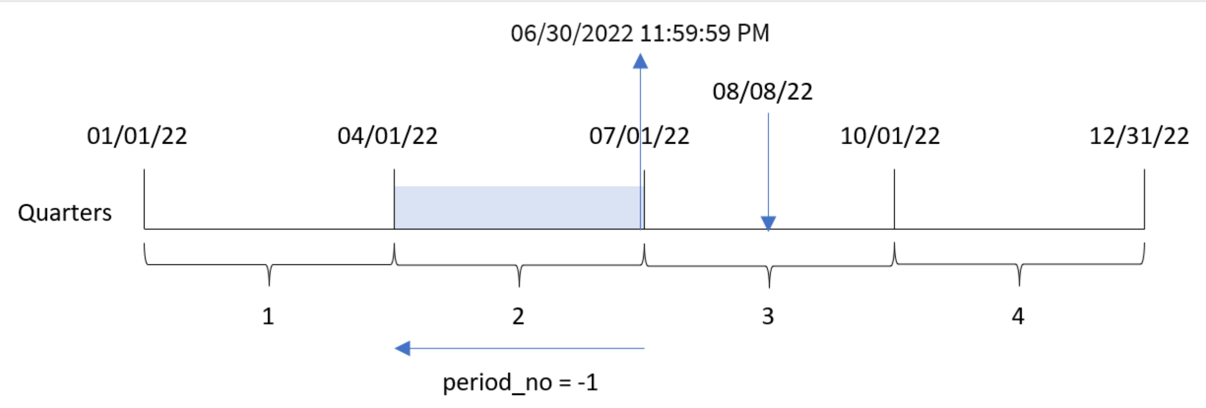
結果表格

id	日期	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

id	日期	previous_quarter_end	previous_quarter_end_timestamp
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

因為 period_no 的值 -1 作為 quarterend() 函數中的偏移引數使用，所以該函數首先會識別交易發生的季。然後這會往前偏移一季並識別該季的最後一毫秒。

具有 period_no 值 -1 的 quarterend() 函數圖表



交易 8203 發生於 8 月 8 日。quarterend() 函數識別交易發生的前一季介於 4 月 1 日和 6 月 30 日之間。然後該函數會傳回該季的最後一毫秒 6 月 30 日 11:59:59 PM。

範例 3 - first_month_of_year

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為「Transactions」的表格中。
- 前置載入包含下列內容：
  - 設為「end_of_quarter」欄位並傳回交易發生該季結束之時間戳記的 quarterend() 函數。
  - 設為「end_of_quarter_timestamp」欄位並傳回所選季結束之確切時間戳記的 timestamp () 函數。

不過，在此範例中，公司政策適用於會計年度在 3 月 1 日開始的情況。

載入指令碼

```
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
];
```

結果

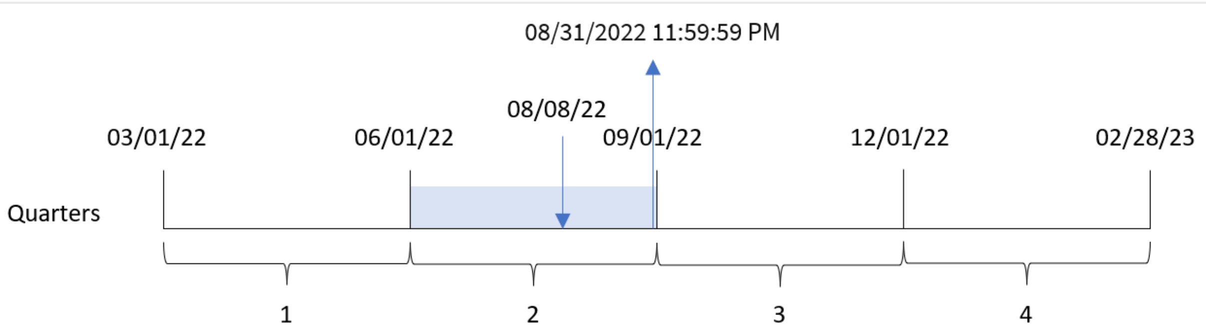
結果表格

id	日期	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

id	日期	end_of_quarter	end_of_quarter_timestamp
8193	4/1/2022	05/31/2022	5/31/2022 11:59:59 PM
8194	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8195	5/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8196	6/15/2022	08/31/2022	8/31/2022 11:59:59 PM
8197	6/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	11/30/2022	11/30/2022 11:59:59 PM
8206	10/14/2022	11/30/2022	11/30/2022 11:59:59 PM
8207	10/29/2022	11/30/2022	11/30/2022 11:59:59 PM

在此例中，因為 `first_month_of_year` 引數 3 用於 `quarterend()` 函數，該年的開始會從 1 月 1 日移至 3 月 1 日。

3 月作為該年第一個月的 quarterend() 函數圖表



交易 8203 發生於 8 月 8 日。因為該年的開始是 3 月 1 日，該年的各季發生在 3 月 - 5 月、6 月 - 8 月、9 月 - 11 月 和 12 月 - 2 月之間。

`quarterend()` 函數識別該交易發生 6 月開始和 8 月之間的季，並傳回該季的最後一毫秒，即 8 月 31 日 11:59:59 PM。

範例 4 - 圖表物件範例

載入指令碼和圖表運算式

概覽

使用與第一個範例相同的資料集和情境。

不過，在此範例中，資料集保持不變並且會載入到應用程式中。會建立傳回交易發生該季結束時間戳記的計算，作為應用程式圖表中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date

若要計算交易發生的該季結束日期，建立下列量值：

- =quarterend(date)
- =timestamp(quarterend(date))

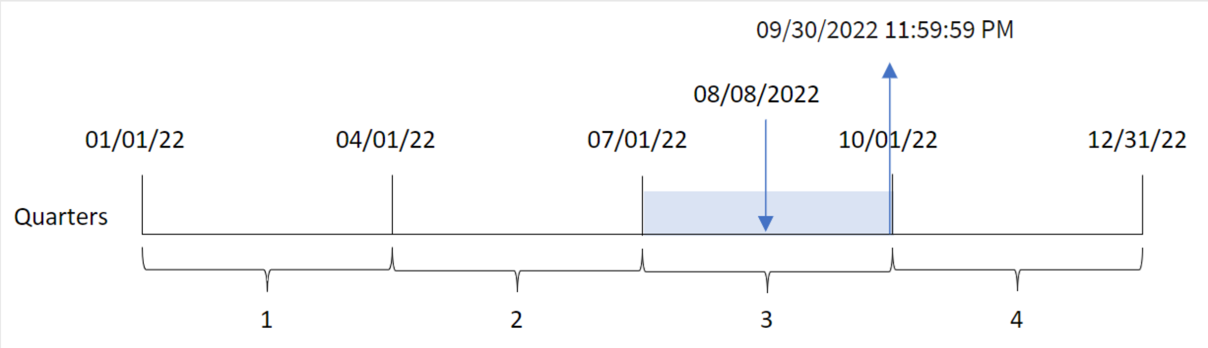
結果表格

id	日期	=quarterend(date)	=timestamp(quarterend(date))
8188	1/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8189	1/19/2022	03/31/2022	3/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	06/30/2022	6/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/16/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	09/30/2022	9/30/2022 11:59:59 PM
8199	7/22/2022	09/30/2022	9/30/2022 11:59:59 PM
8200	7/23/2022	09/30/2022	9/30/2022 11:59:59 PM
8201	7/27/2022	09/30/2022	9/30/2022 11:59:59 PM
8202	8/2/2022	09/30/2022	9/30/2022 11:59:59 PM
8203	8/8/2022	09/30/2022	9/30/2022 11:59:59 PM
8204	8/19/2022	09/30/2022	9/30/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	10/29/2022	12/31/2022	12/31/2022 11:59:59 PM

「end_of_quarter」欄位在前置 LOAD 陳述式中的建立方式是使用 quarterend() 函數並傳遞日期欄位，作為函數的引數。

quarterend() 函數最初識別日期值落在哪一季，然後傳回該季最後一毫秒的時間戳記。

識別交易 8203 之季度結束的 quarterend() 函數圖表



交易 8203 發生於 8 月 8 日。quarterend() 函數識別該交易發生在第三季，並傳回該季的最後一毫秒，即 9 月 30 日 11:59:59 PM。

範例 5 - 情境

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 資料集載入到稱為「Employee_Expenses」的表格。該表格含有下列欄位：
  - 員工 ID
  - 員工名稱
  - 每個員工報銷的平均每日開支。

最終使用者希望圖表物件按員工 ID 和員工名稱顯示該季剩餘期間仍會產生的預估開支報銷。會計年度於 1 月開始。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- `employee_id`
- `employee_name`

若要計算累積的利息，建立下列量值：

- `=(quarterend(today(1))-today(1))*avg_daily_claim`

將量值的**數字格式**設定為**金錢**。

結果表格

employee_id	employee_name	=(quarterend(today(1))-today(1))*avg_daily_claim
182	Mark	$480.00
183	Deryck	$400.00
184	Dexter	$400.00
185	Sydney	$864.00
186	Agatha	$576.00

`quarterend()` 函數使用今日日期作為其唯一引數，並傳回目前月份的結束日期。然後，以該年結束日期減去今日日期，運算式就會傳回此月剩餘天數。

然後按員工讓此值乘以平均每日開支報銷，以計算每個員工在該季剩餘期間預期會有的預估報銷值。

quartername

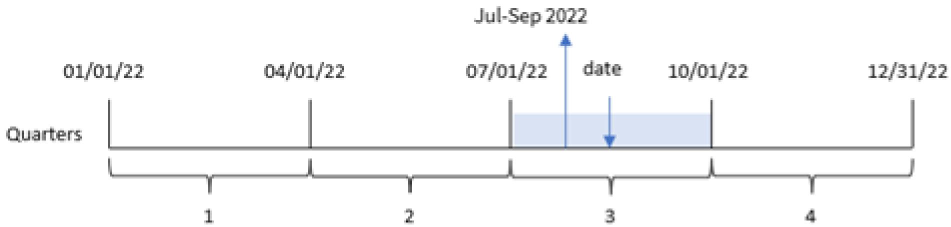
此函數會傳回顯示當季度月份 (根據 **MonthNames** 指令碼變數進行格式設定) 和年度的顯示值，其基礎數值相當於該季度第一天、第一毫秒的時間戳記。

語法：

```
QuarterName(date[, period_no[, first_month_of_year]])
```

傳回的資料類型： 雙值

quartername() 函數的圖表



quartername() 函數判定日期落在哪一季。然後這傳回值，顯示此季的開始到結束月份以及該年。此結果的基礎數值是該季的第一毫秒。

引數

引數	描述
date	要評估的時間戳記。
period_no	**period_no** 為整數，值 0 表示是包含 **date** 的季度。負值的 **period_no** 表示之前的季度，正值表示之後的季度。
first_month_of_year	如果要使用不起始於 1 月的 (會計) 年度，可在 **first_month_of_year** 中指定介於 2 和 12 之間的值。

什麼情況下使用

若您想要按季度比較彙總，則 quartername() 函數很實用。例如，若您想要按季度查看產品的總銷售額。

此函數可用於載入指令碼中，以建立主要行事曆表格中的欄位。相反地，這可直接在圖表中作為計算維度使用。

這些範例使用日期格式 MM/DD/YYYY。日期格式是在位於資料載入指令碼頂端的 SET DateFormat 陳述式中指定。變更範例中的格式，以滿足您的需求。

函數範例

範例	結果
quartername('10/29/2013')	傳回 Oct-Dec 2013。
quartername('10/29/2013', -1)	傳回 Jul-Sep 2013。
quartername('10/29/2013', 0, 3)	傳回 Sep-Nov 2013。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 無其他引數的日期

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2022 年交易的資料集, 這載入到稱為 `Transactions` 的表格中。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 `transaction_quarter`, 這傳回交易發生的季度。

按需要在此新增其他文字, 可附上清單等。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- transaction_quarter

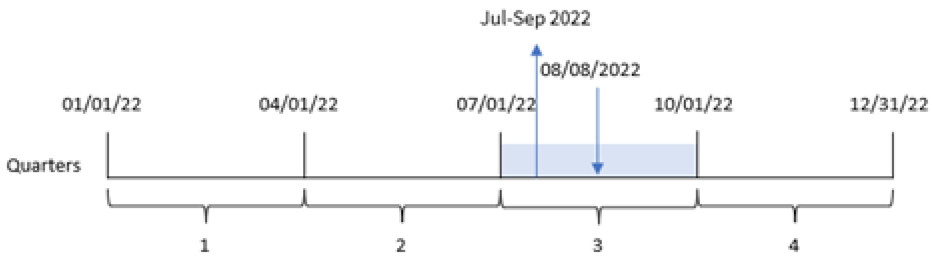
結果表格

日期	transaction_quarter
1/7/2022	2022 年 1 月至 3 月
1/19/2022	2022 年 1 月至 3 月
2/5/2022	2022 年 1 月至 3 月
2/28/2022	2022 年 1 月至 3 月
3/16/2022	2022 年 1 月至 3 月
4/1/2022	2022 年 4 月至 6 月
5/7/2022	2022 年 4 月至 6 月
5/16/2022	2022 年 4 月至 6 月
6/15/2022	2022 年 4 月至 6 月
6/26/2022	2022 年 4 月至 6 月
7/9/2022	2022 年 7 月至 9 月
7/22/2022	2022 年 7 月至 9 月
7/23/2022	2022 年 7 月至 9 月
7/27/2022	2022 年 7 月至 9 月
8/2/2022	2022 年 7 月至 9 月
8/8/2022	2022 年 7 月至 9 月
8/19/2022	2022 年 7 月至 9 月
9/26/2022	2022 年 7 月至 9 月
10/14/2022	2022 年 10 月至 12 月
10/29/2022	2022 年 10 月至 12 月

`transaction_quarter` 欄位在前置 LOAD 陳述式中的建立方式是使用 `quartername()` 函數並傳遞日期欄位，作為函數的引數。

`quartername()` 函數最初識別日期值落在的季。然後這傳回值，顯示此季的開始到結束月份以及該年。

quartername() 函數的圖表，無其他引數的範例



交易 8203 發生於 2022 年 8 月 8 日。quartername() 函數識別交易發生在第三季，因此傳回 2022 年 7 月至 9 月。月份的顯示格式與 MonthNames 系統變數相同。

範 2 – 有 period_no 引數的日期

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 建立欄位 previous_quarter，這傳回交易發生的上一季。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- previous_quarter

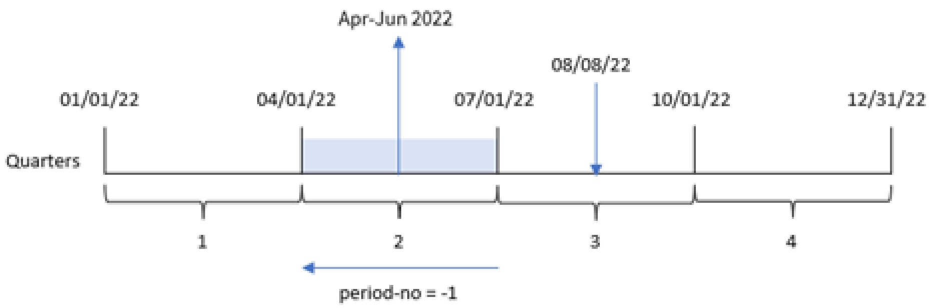
結果表格

日期	previous_quarter
1/7/2022	2021 年 10 月至 12 月
1/19/2022	2021 年 10 月至 12 月
2/5/2022	2021 年 10 月至 12 月
2/28/2022	2021 年 10 月至 12 月
3/16/2022	2021 年 10 月至 12 月
4/1/2022	2022 年 1 月至 3 月
5/7/2022	2022 年 1 月至 3 月
5/16/2022	2022 年 1 月至 3 月
6/15/2022	2022 年 1 月至 3 月
6/26/2022	2022 年 1 月至 3 月
7/9/2022	2022 年 4 月至 6 月
7/22/2022	2022 年 4 月至 6 月
7/23/2022	2022 年 4 月至 6 月
7/27/2022	2022 年 4 月至 6 月
8/2/2022	2022 年 4 月至 6 月
8/8/2022	2022 年 4 月至 6 月
8/19/2022	2022 年 4 月至 6 月
9/26/2022	2022 年 4 月至 6 月

日期	previous_quarter
10/14/2022	2022 年 7 月至 9 月
10/29/2022	2022 年 7 月至 9 月

在此例中，因為 period_no 的值 -1 已作為 quartername() 函數中的偏移引數使用，所以該函數首先會識別交易發生在第三季。然後這會往前偏移一季，並傳回值，顯示此季的開始到結束月份以及該年。

quartername() 函數的圖表，period_no 範例



交易 8203 發生於 8 月 8 日。quartername() 函數識別交易發生的前一季介於 4 月 1 日和 6 月 30 日之間。因此，這會傳回 2022 年 4 月至 6 月。

範例 3 – 有 first_week_day 引數的日期

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。不過，在此範例中，我們需要將 3 月 1 日設定為會計年度的開始。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- transaction_quarter

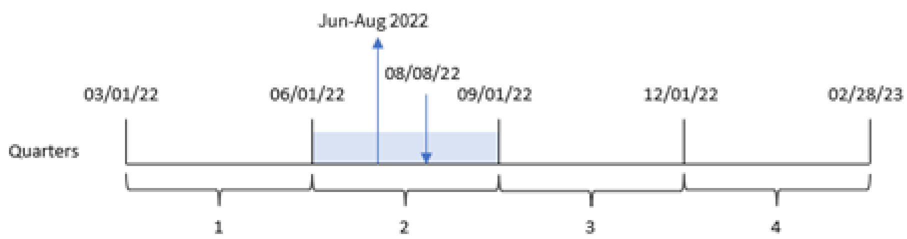
結果表格

日期	transaction_quarter
1/7/2022	2021 年 12 月至 2 月
1/19/2022	2021 年 12 月至 2 月
2/5/2022	2021 年 12 月至 2 月
2/28/2022	2021 年 12 月至 2 月
3/16/2022	2022 年 3 月至 5 月
4/1/2022	2022 年 3 月至 5 月
5/7/2022	2022 年 3 月至 5 月
5/16/2022	2022 年 3 月至 5 月
6/15/2022	2022 年 6 月至 8 月
6/26/2022	2022 年 6 月至 8 月
7/9/2022	2022 年 6 月至 8 月
7/22/2022	2022 年 6 月至 8 月

日期	transaction_quarter
7/23/2022	2022 年 6 月至 8 月
7/27/2022	2022 年 6 月至 8 月
8/2/2022	2022 年 6 月至 8 月
8/8/2022	2022 年 6 月至 8 月
8/19/2022	2022 年 6 月至 8 月
9/26/2022	2022 年 9 月至 11 月
10/14/2022	2022 年 9 月至 11 月
10/29/2022	2022 年 9 月至 11 月

在此例中，因為 `first_month_of_year` 引數 3 用於 `quartername()` 函數，所以該年的開始從 1 月 1 日移至 3 月 1 日。因此，該年的季分為 3 月 - 5 月、6 月 - 8 月、9 月 - 11 月和 12 月 - 2 月。

`quartername()` 函數的圖表，first_week_day 範例



交易 8203 發生於 8 月 8 日。`quartername()` 函數識別交易發生在第二季，介於 6 月 開始和 8 月結束之間。因此，這會傳回 2022 年 6 月至 8 月。

範例 4 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。

不過，在此範例中，不變的資料集會載入到應用程式中。會建立傳回交易發生該季結束時間戳記的計算，作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`date`。

建立下列量值：

`=quartername(date)`

結果表格

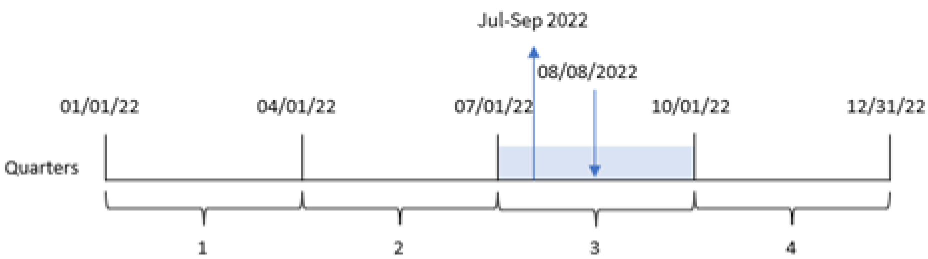
日期	=quartername(date)
1/7/2022	2022 年 1 月至 3 月
1/19/2022	2022 年 1 月至 3 月
2/5/2022	2022 年 1 月至 3 月
2/28/2022	2022 年 1 月至 3 月
3/16/2022	2022 年 1 月至 3 月
4/1/2022	2022 年 4 月至 6 月
5/7/2022	2022 年 4 月至 6 月
5/16/2022	2022 年 4 月至 6 月
6/15/2022	2022 年 4 月至 6 月
6/26/2022	2022 年 4 月至 6 月
7/9/2022	2022 年 7 月至 9 月

日期	=quartername(date)
7/22/2022	2022 年 7 月至 9 月
7/23/2022	2022 年 7 月至 9 月
7/27/2022	2022 年 7 月至 9 月
8/2/2022	2022 年 7 月至 9 月
8/8/2022	2022 年 7 月至 9 月
8/19/2022	2022 年 7 月至 9 月
9/26/2022	2022 年 7 月至 9 月
10/14/2022	2022 年 10 月至 12 月
10/29/2022	2022 年 10 月至 12 月

`transaction_quarter` 量值在圖表物件中的建立方式是使用 `quartername()` 函數並傳遞 `date` 欄位，作為函數的引數。

`quartername()` 函數最初識別日期值落在的季。然後這傳回值，顯示此季的開始到結束月份以及該年。

`quartername()` 函數的圖表，圖表物件範例



交易 8203 發生於 2022 年 8 月 8 日。`quartername()` 函數識別交易發生在第三季，因此傳回 2022 年 7 月至 9 月。月份的顯示格式與 `MonthNames` 系統變數相同。

範例 5 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為 `Transactions` 的表格中。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。

最終使用者希望圖表物件按季呈現交易的總銷售額。當資料模型中無法使用此維度時，在圖表中將 quartername() 函數作為計算維度可達成這一點。

載入指令碼

```
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
];
```

結果

請執行下列動作：

1. 載入資料並開啟工作表。建立新的表格。
2. 使用下列運算式建立計算維度：
   =quartername(date)
3. 接著，使用下列彙總量值計算總銷售額：
   =sum(amount)
4. 將量值的**數字格式**設定為**金錢**。

結果表格

=quartername(date)	=sum(amount)
2022 年 7 月至 9 月	$446.31

=quartername(date)	=sum(amount)
2022 年 4 月至 6 月	$351.48
2022 年 1 月至 3 月	$253.89
2022 年 10 月至 12 月	$163.91

quarterstart

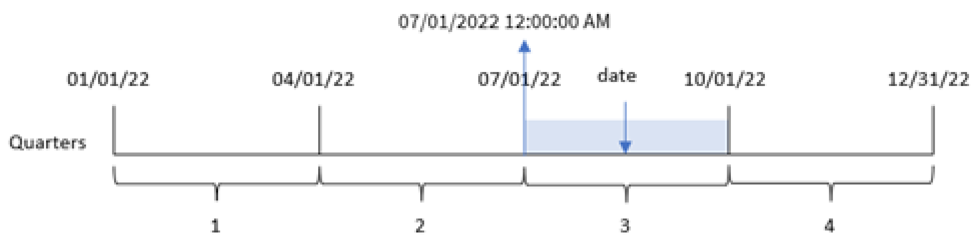
此函數傳回的值相當於包含 **date** 的季度的第一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 **DateFormat**。

語法：

```
QuarterStart(date[, period_no[, first_month_of_year]])
```

傳回的資料類型： 雙值

quarterstart() 函數的圖表



quarterstart() 函數判定 date 落在哪一季。然後以日期格式傳回該季第一個月的第一毫秒的時間戳記。

引數

引數	描述
date	要評估的時間戳記。
period_no	**period_no** 為整數，值 0 表示是包含 **date** 的季度。負值的 **period_no** 表示之前的季度，正值表示之後的季度。
first_month_of_year	如果要使用不起始於 1 月的 (會計) 年度，可在 **first_month_of_year** 中指定介於 2 和 12 之間的值。

什麼情況下使用

quarterstart() 函數通常在使用者想要計算使用一季中已經過的部分時，作為運算式的一部分使用。例如，這可讓使用者用於計算季初至今已累積的利息。

函數範例

範例	結果
quarterstart('10/29/2005')	傳回 10/01/2005。
quarterstart('10/29/2005', -1)	傳回 07/01/2005。
quarterstart('10/29/2005', 0, 3)	傳回 09/01/2005。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為 Transactions 的表格中。
- 以 DateFormat 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 start_of_quarter，這傳回交易發生該季開始的時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- start_of_quarter
- start_of_quarter_timestamp

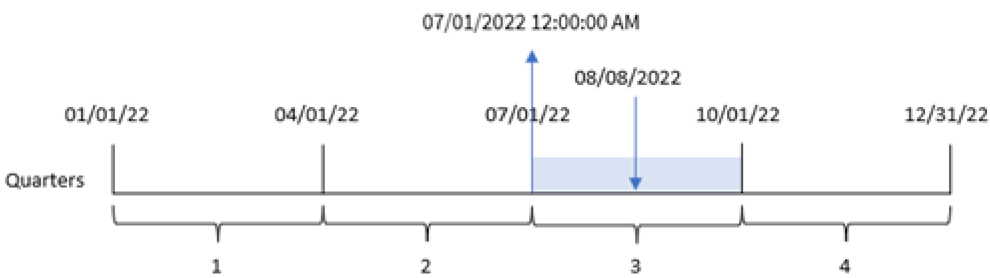
結果表格

日期	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

日期	start_of_quarter	start_of_quarter_timestamp
7/23/2022	07/01/2022	7/1/2021 12:00:00 AM
7/27/2022	07/01/2022	7/1/2021 12:00:00 AM
8/2/2022	07/01/2022	7/1/2021 12:00:00 AM
8/8/2022	07/01/2022	7/1/2021 12:00:00 AM
8/19/2022	07/01/2022	7/1/2021 12:00:00 AM
9/26/2022	07/01/2022	7/1/2021 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM

start_of_quarter 欄位在前置 LOAD 陳述式中的建立方式是使用 quarterstart() 函數並傳遞日期欄位，作為函數的引數。uarterstart() 函數最初識別日期值落在哪一季。然後傳回該季第一毫秒的時間戳記。

quarterstart() 函數的圖表，無其他引數的範例



交易 8203 發生於 8 月 8 日。quarterstart() 函數識別該交易發生在第三季，並傳回該季的第一毫秒，即 7 月 1 日 12:00:00 AM。

範例 2 – period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 建立欄位 previous_quarter_start，，這傳回交易發生之前每季開始的時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- previous_quarter_start
- previous_quarter_start_timestamp

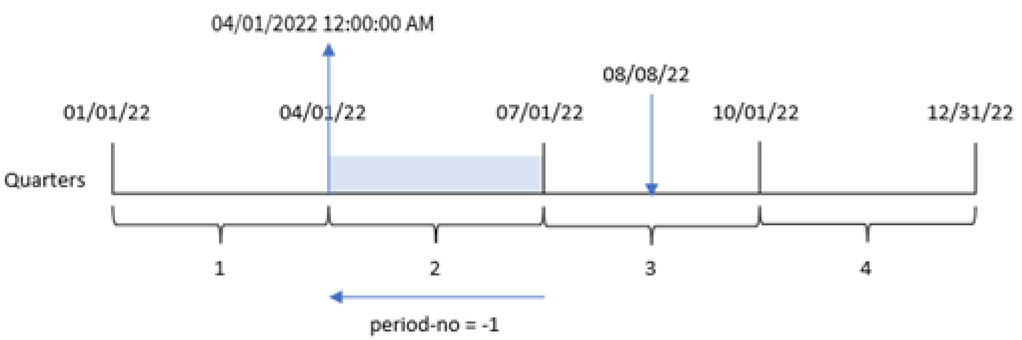
結果表格

日期	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

日期	previous_quarter_start	previous_quarter_start_timestamp
2/5/2022	10/01/2021	10/1/2021 12:00:00 AM
2/28/2022	10/01/2021	10/1/2021 12:00:00 AM
3/16/2022	10/01/2021	10/1/2021 12:00:00 AM
4/1/2022	01/01/2022	1/1/2022 12:00:00 AM
5/7/2022	01/01/2022	1/1/2022 12:00:00 AM
5/16/2022	01/01/2022	1/1/2022 12:00:00 AM
6/15/2022	01/01/2022	1/1/2022 12:00:00 AM
6/26/2022	01/01/2022	1/1/2022 12:00:00 AM
7/9/2022	04/01/2022	4/1/2021 12:00:00 AM
7/22/2022	04/01/2022	4/1/2021 12:00:00 AM
7/23/2022	04/01/2022	4/1/2021 12:00:00 AM
7/27/2022	04/01/2022	4/1/2021 12:00:00 AM
8/2/2022	04/01/2022	4/1/2021 12:00:00 AM
8/8/2022	04/01/2022	4/1/2021 12:00:00 AM
8/19/2022	04/01/2022	4/1/2021 12:00:00 AM
9/26/2022	04/01/2022	4/1/2021 12:00:00 AM
10/14/2022	07/01/2022	7/1/2022 12:00:00 AM
10/29/2022	07/01/2022	7/1/2022 12:00:00 AM

在此例中，因為 period_no 的值 -1 已作為 quarterstart() 函數中的偏移引數使用，所以該函數首先會識別交易發生的季度。然後這會往前偏移一季並識別該季的第一毫秒。

quarterstart() 函數的圖表，period_no 範例



交易 8203 發生於 8 月 8 日。quarterstart() 函數識別交易發生的前一季介於 4 月 1 日和 6 月 30 日之間。然後這會傳回該季的第一毫秒 4 月 1 日 12:00:00 AM。

範例 3 – first_month_of_year

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。不過，在此範例中，我們需要將 3 月 1 日設定為會計年度的開始。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

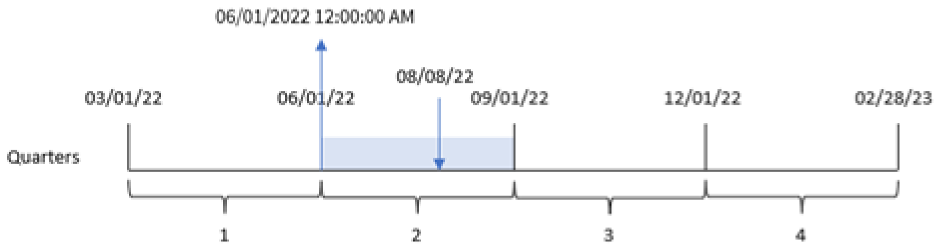
- date
- start_of_quarter
- start_of_quarter_timestamp

結果表格

日期	start_of_quarter	start_of_quarter_timestamp
1/7/2022	12/01/2021	12/1/2021 12:00:00 AM
1/19/2022	12/01/2021	12/1/2021 12:00:00 AM
2/5/2022	12/01/2021	12/1/2021 12:00:00 AM
2/28/2022	12/01/2021	12/1/2021 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
4/1/2022	03/01/2022	3/1/2022 12:00:00 AM
5/7/2022	03/01/2022	3/1/2022 12:00:00 AM
5/16/2022	03/01/2022	3/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	06/01/2022	6/1/2022 12:00:00 AM
7/9/2022	06/01/2022	6/1/2022 12:00:00 AM
7/22/2022	06/01/2022	6/1/2022 12:00:00 AM
7/23/2022	06/01/2022	6/1/2022 12:00:00 AM
7/27/2022	06/01/2022	6/1/2022 12:00:00 AM
8/2/2022	06/01/2022	6/1/2022 12:00:00 AM
8/8/2022	06/01/2022	6/1/2022 12:00:00 AM
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

因為 first_month_of_year 引數 3 用於 quarterstart() 函數，該年的開始會從 1 月 1 日移至 3 月 1 日。

quarterstart() 函數的圖表, first_month_of_year 範例



交易 8203 發生於 8 月 8 日。因為該年的開始是 3 月 1 日,該年的各季發生在 3 月 - 5 月、6 月 - 8 月、9 月 - 11 月 和 12 月 - 2 月之間。quarterstart() 函數識別該交易發生 6 月開始和 8 月之間的季,並傳回該季的第一毫秒,即 6 月 1 日 12:00:00 AM。

範例 4 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。

不過,在此範例中,不變的資料集會載入到應用程式中。會建立傳回交易發生該季結束時間戳記的計算,作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：date。

新增下列量值：

- =quarterstart(date)
- =timestamp(quarterstart(date))

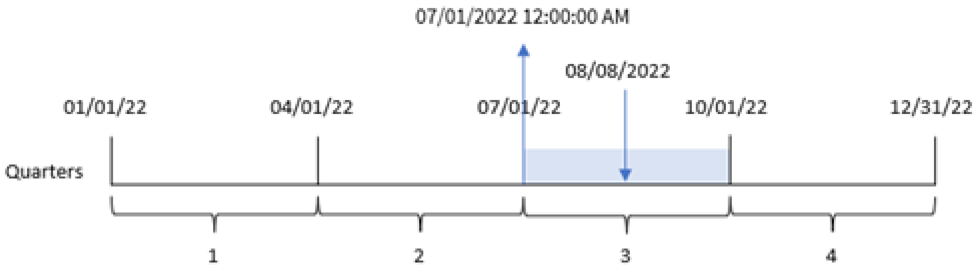
結果表格

日期	=quarterstart(date)	=timestamp(quarterstart(date))
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
9/26/2022	07/01/2022	7/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM
5/7/2022	04/01/2022	4/1/2022 12:00:00 AM
5/16/2022	04/01/2022	4/1/2022 12:00:00 AM
6/15/2022	04/01/2022	4/1/2022 12:00:00 AM
6/26/2022	04/01/2022	4/1/2022 12:00:00 AM
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM
2/5/2022	01/01/2022	1/1/2022 12:00:00 AM
2/28/2022	01/01/2022	1/1/2022 12:00:00 AM
3/16/2022	01/01/2022	1/1/2022 12:00:00 AM

start_of_quarter 量值在圖表物件中的建立方式是使用 quarterstart() 函數並傳遞 date 欄位，作為函數的引數。

quarterstart() 函數識別日期值落在哪一季，並傳回該季第一毫秒的時間戳記。

quarterstart() 函數的圖表，圖表物件範例



交易 8203 發生於 8 月 8 日。quarterstart() 函數識別該交易發生在第三季，並傳回該季的第一毫秒。傳回的值是 7 月 1 日 12:00:00 AM。

範例 5 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 資料集包含一系列貸款餘額，其位於稱為 Loans 的表格。
- 資料包含貸款 ID、季開始餘額以及每年對每項貸款收取的單利率。

最終使用者希望圖表物件按貸款 ID 顯示季初至今每筆貸款已累積的目前利息。

載入指令碼

```
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
];
```

結果

請執行下列動作:

1. 載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:
   - `loan_id`
   - `start_balance`
2. 接下來,建立此量值以計算累積的利息:
   `=start_balance*(rate*(today(1)-quarterstart(today(1)))/365)`
3. 將量值的**數字格式**設定為**金錢**。

結果表格

loan_id	start_balance	=start_balance*(rate*(today(1)-quarterstart(today(1)))/365)
8188	$10000.00	$15.07
8189	$15000.00	$128.84
8190	$17500.00	$63.29
8191	$21000.00	$107.59
8192	$90000.00	$1139.18

`quarterstart()` 函數使用今日日期作為其唯一引數,傳回目前年份的開始日期。以目前日期減去該結果後,運算式會傳回此季目前已經過的天數。

然後此值乘以利率並除以 365,以傳回此期間產生的有效利率。然後該結果乘以貸款的開始餘額,以傳回此季目前已累積的利息。

second

當 **expression** 的分數根據標準數字解譯的方式可解譯為時間時,此函數會傳回代表秒的整數。

語法:

```
second (expression)
```

傳回的資料類型: 整數

什麼情況下使用

若您想要按秒比較彙總,則 second() 函數很實用。例如,若您想要按秒查看活動計數分佈,可使用該函數。

這些維度在建立時可以在載入指令碼進行,或使用函數建立「主要行事曆」表格的欄位,或是直接在圖表中當作計算維度使用。

函數範例

範例	結果
second('09:14:36')	傳回 36
second('0.5555')	傳回 55 (因為 0.5555 = 13:19:55)

區域設定

除非另有說明,否則此主題中的範例皆使用下列日期格式:MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素,您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式,以滿足您的需求。或者,您可以在載入指令碼中變更格式,以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典,資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 變數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含依時間戳記之交易的資料集,這載入到稱為 Transactions 的表格中。
- 已使用預設的 TimeStamp 系統變數 (M/D/YYYY h:mm:ss[.fff] TT)。
- 購買時用於計算的欄位 second 建立。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- second

結果表格

日期	second
01/01/2022 10:10:22 PM	22
01/02/2022 8:35:54 AM	54
01/03/2022 5:40:49 AM	49
01/03/2022 2:21:53 PM	53
01/04/2022 6:49:38 PM	38
01/05/2022 7:04:57 PM	57
01/05/2022 7:34:46 PM	46
01/06/2022 8:49:09 AM	9
01/06/2022 11:29:38 AM	38
01/06/2022 10:58:34 PM	34

使用 second() 函數並傳遞日期作為前置 LOAD 陳述式中的運算式，以建立 second 欄位中的值。

範例 2 – 圖表物件

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。不過，在此範例中，不變的資料集會載入到應用程式中。會透過圖表物件中的量值計算 second 值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：date。

建立下列量值：

=second(date)

結果表格

日期	=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

second 值的建立方式是使用 second() 函數並傳遞日期作為圖表物件量值中的運算式。

範例 3 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 時間戳記資料集的產生可代表特定節日票券銷售網站的流量。這些時間戳記和對應的 id 位於稱為 Web_Traffic 的表格中。
- 已使用 TimeStamp 系統變數 M/D/YYYY h:mm:ss[.fff] TT。

在此情境中, 總共有 10,000 張票券, 並在 2021 年 5 月 20 日上午 9:00 開賣。票卷在一分鐘後完售。

使用者需要按秒顯示網站訪客數量的圖表物件。

載入指令碼

```
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;
```

結果

請執行下列動作:

1. 載入資料並開啟工作表。建立新的表格。
2. 接下來, 使用下列運算式建立計算維度:
   =second(timestamp)
3. 建立彙總量值以計算總項目數:
   =count(id)

結果表格與下表類似, 但具有不同的彙總量值:

結果表格

second(timestamp)	=count(id)
0	150
1	184
2	163
3	178
4	179
5	158
6	177
7	169
8	149
9	186
10	169
11	179
12	186
13	182
14	180
15	153
16	191
17	203
18	158
19	159
20	163
其他 39 列	

setdateyear

此函數會輸入 **timestamp** 與 **year**，並使用輸入中指定的 **year** 更新 **timestamp**。

語法：

```
setdateyear (timestamp, year)
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
timestamp	標準 Qlik Sense 時間戳記 (通常只是日期)。
year	四位數的年度。

範例與結果:
這些範例使用日期格式 **DD/MM/YYYY**。日期格式是在位於資料載入指令碼頂端的 **SET DateFormat** 陳述式中指定。變更範例中的格式, 以滿足您的需求。

指令碼處理範例

範例	結果
`setdateyear ('29/10/2005', 2013)`	傳回 '29/10/2013
`setdateyear ('29/10/2005 04:26:14', 2013)`	傳回 '29/10/2013 04:26:14' 若要查看視覺化中時間戳記的時間部分, 您必須將數字格式設定為「日期」, 並從顯示時間值的「格式設定」中選取一個值。

範例:

將範例指令碼新增至您的應用程式並予以執行。若要查看結果, 將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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

];

產生的表格包含原始日期及資料行，其中年已設為 2013。

結果表格

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

此函數會輸入 **timestamp**、**month** 與 **year**，並使用輸入中指定的 **year** 和 **month** 更新 **timestamp**。.

語法：

```
SetDateYearMonth (timestamp, year, month)
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
timestamp	標準 Qlik Sense 時間戳記 (通常只是日期)。
year	四位數的年度。
month	1 位數或 2 位數的月份。

範例與結果:
這些範例使用日期格式 **DD/MM/YYYY**。日期格式是在位於資料載入指令碼頂端的 **SET DateFormat** 陳述式中指定。變更範例中的格式,以滿足您的需求。

指令碼處理範例

範例	結果
`setdateyearmonth ('29/10/2005', 2013, 3)`	傳回 '29/03/2013
`setdateyearmonth ('29/10/2005 04:26:14', 2013, 3)`	傳回 '29/03/2013 04:26:14' 若要查看視覺化中時間戳記的時間部分,您必須將數字格式設定為「日期」,並從顯示時間值的「格式設定」中選取一個值。

範例:

將範例指令碼新增至您的應用程式並予以執行。若要查看結果,將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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

];
```

產生的表格包含原始日期及資料行，其中年已設為 2013。

結果表格

testdates	NewYearMonth
1/11/2012	1/3/2013
10/12/2012	10/3/2013
2/1/2012	2/3/2013
19/5/2013	19/3/2013
15/9/2013	15/3/2013
11/12/2013	11/3/2013
14/5/2014	14/3/2013
13/6/2014	13/3/2013
7/7/2014	7/3/2013
4/8/2014	4/3/2013

timezone

此函數傳回時區，如執行 Qlik 引擎的電腦上所定義。

語法：

```
TimeZone( )
```

傳回的資料類型： 雙值

範例：

`timezone( )`

若您想要在應用程式的量值中查看不同的時區，您可以在量值中使用 `localtime()` 函數。

today

此函數會傳回目前日期。該函數會以 `DateFormat` 系統變數格式傳回值。

語法:

`today([ timer_mode])`

傳回的資料類型: 雙值

`today()` 函數可用於載入指令碼或圖表物件中。

預設 `timer_mode` 值為 `1`。

引數

引數	描述
timer_mode	可以具有下列值: 0 (最後完成資料載入的日子) 1 (函數叫用的日子) 2 (開啟應用程式的日子) *如果我們在載入指令碼中使用函數,則* ***timer_mode=0*** *將產生最後完成資料載入的日子,而* ***timer_mode=1*** *將提供目前資料載入的日子。*

函數範例

timer_mode 數值	用於載入指令碼的結果	用於圖表物件的結果
0	以 `DateFormat` 系統變數格式返回最新資料重新載入前,最後一筆成功資料重新載入的日期。	以 `DateFormat` 系統變數格式返回最新資料重新載入的日期。
1	以 `DateFormat` 系統變數格式返回最新資料重新載入的日期。	以 `DateFormat` 系統變數格式返回函數呼叫的日期。
2	以 `DateFormat` 系統變數格式返回使用者應用程式工作階段開始時的日期。這不會更新內容,除非使用者重新載入指令碼。	以 `DateFormat` 系統變數格式返回使用者應用程式工作階段開始的日期。新工作階段開始或應用程式資料重新載入時,即會重新整理內容。

什麼情況下使用

`today()` 函數常用於當作運算式中的元件。例如,這可用於計算截至目前日期之月份內彙總的利息。

下表提供 `today()` 函數傳回結果的解釋,並提供 `timer_mode` 引數的不同數值:

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 使用載入指令碼產生物件

載入指令碼和結果

概覽

下列範例使用 `today()` 函數建立三項變數。每個變數都會使用其中一個 `timer_mode` 個選項示範其效果。

對於示範用途的變數，重新載入指令碼並在 24 小時後第二次重新載入指令碼。這會造成 `today(0)` 和 `today(1)` 變數顯示不同數值，藉此正確示範其用途。

載入指令碼

```
LET vPreviousDataLoad = today(0);
LET vCurrentDataLoad = today(1);
LET vApplicationOpened = today(2);
```

結果

第二次載入資料後，使用以下指示建立三個文字方塊。

首先，為先前載入的資料建立文字方塊。

請執行下列動作：

1. 使用**文字與影像**圖表物件，建立文字方塊。
2. 將下列量值新增至物件：
   `=vPreviousDataLoad`
3. 在**外觀**之下，選取 **Show titles** 並將標題「Previous Reload Time」(先前載入時間) 新增至物件。

接下來，為目前正在載入的資料建立文字方塊。

請執行下列動作：

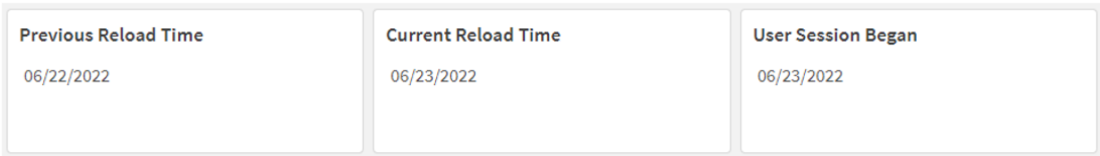
1. 使用**文字與影像**圖表物件，建立文字方塊。
2. 將下列量值新增至物件：
   `=vCurrentDataLoad`
3. 在**外觀**之下，選取 **Show titles** 並將標題「Current Reload Time」(目前載入時間) 新增至物件。

建立最後一個文字方塊，以顯示應用程式中的使用者工作階段何時開始。

請執行下列動作：

1. 使用**文字與影像**圖表物件，建立文字方塊。
2. 將下列量值新增至物件：
   `=vApplicationOpened`
3. 在**外觀**之下，選取 **Show titles** 並將標題「User Session Started」(使用者工作階段已開始) 新增至物件。

使用載入指令碼中 today() 函數建立變數的圖表



上方影像顯示每個建立變數的範例值。例如，數值可能如下所示：

- 先前的重新載入時間：06/22/2022
- 目前的重新載入時間：06/23/2022
- 使用者工作階段開始時間：06/23/2022

範例 2 – 不使用載入指令碼產生物件

載入指令碼和圖表運算式

概覽

下列範例使用 `today()` 函數建立三項圖表物件。每個圖表物件都會使用其中一個 `timer_mode` 選項示範其效果。

此範例沒有載入指令碼。

結果

第二次載入資料後，建立三個文字方塊。

首先，建立最新資料重新載入的文字方塊。

請執行下列動作:

1. 使用**文字與影像**圖表物件,建立文字方塊。
2. 新增下列量值。
   `=today(0)`
3. 在**外觀**之下,選取**顯示標題**並將標題「最新資料重新載入」新增至物件。

接下來,建立顯示目前時間的文字方塊。

請執行下列動作:

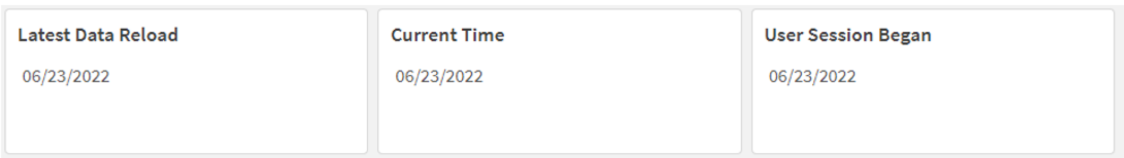
1. 使用**文字與影像**圖表物件,建立文字方塊。
2. 新增下列量值:
   `=today(1)`
3. 在**外觀**之下,選取**顯示標題**並將標題「目前時間」新增至物件。

建立最後一個文字方塊,以顯示應用程式中的使用者工作階段何時開始。

請執行下列動作:

1. 使用**文字與影像**圖表物件,建立文字方塊。
2. 新增下列量值:
   `=today(2)`
3. 在**外觀**之下,選取**顯示標題**並將標題「使用者工作階段已開始」新增至物件。

使用沒有載入指令碼的 today() 函數建立物件的圖表



上方影像顯示每個建立物件的範例值。例如,數值可能如下所示:

- 最新資料重新載入:06/23/2022
- 目前時間: 06/23/2022
- 使用者工作階段開始時間:06/23/2022

「最新資料重新載入」圖表物件使用 0 的 `timer_mode` 值。這會傳回最後一次成功重新載入資料的時間戳記。

「目前時間」圖表物件使用 1 個 `timer_mode` 值。這會傳回根據系統時鐘顯示的目前時間。如果工作表或物件重新整理,即會更新此數值。

「使用者工作階段已開始」的圖表物件使用 2 的 `timer_mode` 值。這會傳回應用程式已開啟且使用者工作階段已開始的時間戳記。

範例 3 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 資料集包含一系列貸款餘額,其位於稱為 Loans 的表格。
- 表格資料具有貸款 ID 欄位、月初餘額,以及在每年每筆貸款收取的單利率。

最終使用者希望圖表物件按貸款 ID 顯示月初至今每筆貸款已累積的目前利息。雖然應用程式每週僅重新載入一次,但使用者還是希望每次物件或應用程式重新整理後,也會重新整理結果的內容。

載入指令碼

```
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
];
```

結果

請執行下列動作:

1. 載入資料並開啟工作表。建立新的表格。
2. 新增下列欄位作為維度。
   - loan_id
   - start_balance
3. 接下來,建立量值以計算累積的利息:
   =start_balance*(rate*(today(1)-monthstart(today(1)))/365)
4. 將量值的**數字格式**設定為**金錢**。

結果表格

loan_id	start_balance	=start_balance*(rate*(today(1)-monthstart(today(1)))/365)
8188	$10000.00	$16.44

loan_id	start_balance	=start_balance*(rate*(today(1)-monthstart(today(1)))/365)
8189	$15000.00	$58.56
8190	$17500.00	$28.77
8191	$21000.00	$48.90
8192	$90000.00	$517.81

`monthstart()` 函數使用傳回今日日期的 `today()` 函數作為其唯一引數，傳回目前月份的開始日期。以目前日期減去該結果後，再次使用 `today()` 函數，運算式會傳回本月目前已經過的天數。

然後此值乘以利率並除以 365，以傳回此期間產生的有效利率。然後該結果乘以貸款的開始餘額，以傳回本月目前已累積的利息。

因為 1 的值已作為運算式內 `today()` 函數的 `timer_mode` 引數，所以每次重新整理圖形物件時 (方法是開啟應用程式、重新整理頁面、在工作表之間移動等)，傳回的日期會針對目前日期，且結果會隨之重新整理。

UTC

傳回目前 Coordinated Universal Time。

語法：

```
UTC( )
```

傳回的資料類型： 雙值

範例：

```
utc( )
```

week

此函數會傳回代表對應至輸入日期之週數的整數。

語法：

```
week(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

傳回的資料類型： 整數

引數

引數	描述
timestamp	要評估的時間戳記。

引數	描述
first_week_day	指定一週開始的日期。如果忽略，將使用變數 **FirstWeekDay** 的值。 **first_week_day** 可能的值是對星期一使用 0、對星期二使用 1，對星期三使用 2，對星期四使用 3，對星期五使用 4，對星期六使用 5，並對星期日使用 6。 如需系統變數的更多資訊，請參閱 *FirstWeekDay (page 212)*。
broken_weeks	如果您未指定 **broken_weeks**，則變數 **BrokenWeeks** 的值將用於定義週是否中斷。
reference_day	如果您不指定 **reference_day**，變數 **ReferenceDay** 的值將用於定義要設定一月份的哪一天為參照日以定義第 1 週。依照預設，Qlik Sense 函數使用 4 作為參照日。這表示第 1 週必須包含 1 月 4 日，或者換言之，第 1 週必須始終至少在 1 月擁有 4 日。

`week()` 函數判定日期落在哪一週並傳回週數。

在 Qlik Sense 中，建立應用程式時會擷取地區設定，對應的設定會儲存在指令碼中，作為環境變數。這些用來決定週數。

這表示，大部分的歐洲應用程式開發人員會取得下列環境變數，對應至 ISO 8601 定義：

```
Set FirstWeekDay =0;    // Monday as first week day
Set BrokenWeeks  =0;    // Use unbroken weeks
Set ReferenceDay =4;    // Jan 4th is always in week 1
```

北美應用程式開發人員通常會取得下列環境變數：

```
Set FirstWeekDay =6;    // Sunday as first week day
Set BrokenWeeks  =1;    // Use broken weeks
Set ReferenceDay =1;    // Jan 1st is always in week 1
```

該週的第一天由 `FirstWeekDay` 系統變數決定。您可以使用 `week()` 函數中的 `first_week_day` 引數變更該週的第一天。

若應用程式使用中斷的週，則週數計數在 1 月 1 日開始，並在 `FirstWeekDay` 系統變數的前一天結束，無論發生了多少天。

若應用程式正在使用未中斷的週，則第 1 週可以在前一年或 1 月的開頭幾天開始。這取決於您如何使用 `FirstWeekDay` 和 `ReferenceDay` 環境變數。

什麼情況下使用

若您想要按週比較彙總，則 `The week()` 函數很實用。例如，若您想要按週查看產品的總銷售額，可使用此函數。若使用者想要計算不一定要使用應用程式的 `BrokenWeeks`、`FirstWeekDay` 或 `ReferenceDay` 系統變數時，即會選擇 `week()` 函數，而不是 `weekname()`。

例如，若您想要按週查看產品的總銷售額。

若應用程式正在使用未中斷的週，則第 1 週可能包含上一年 12 月中的日期，或排除目前年份 1 月的日期。若應用程式正在使用中斷的週，則第 1 週的天數可少於七天。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

以下範例假設

```
Set DateFormat= 'MM/DD/YYYY';
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

函數範例

範例	**結果**
week('12/28/2021')	傳回 52。
week(44614)	傳回 8，因為這是 02/22/2022 的序號。
week('01/03/2021')	傳回 53。
week('01/03/2021',6)	傳回 1。

範例 1 – 預設系統變數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2021 年最後一週和 2022 年前兩週交易的資料集，這載入到稱為 `Transactions` 的表格中。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 week_number，傳回交易發生年份和週數。
- 建立稱為 week_day 的欄位，顯示每個交易日期的工作日值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date
- week_day
- week_number

結果表格

id	日期	week_day	week_number
8183	12/27/2021	星期一	53
8184	12/28/2021	星期二	53
8185	12/29/2021	星期三	53

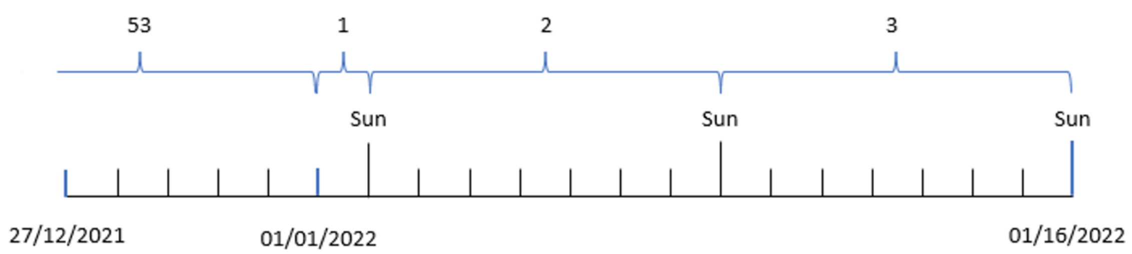
id	日期	week_day	week_number
8186	12/30/2021	星期四	53
8187	12/31/2021	星期五	53
8188	01/01/2022	星期六	1
8189	01/02/2022	星期日	2
8190	01/03/2022	星期一	2
8191	01/04/2022	星期二	2
8192	01/05/2022	星期三	2
8193	01/06/2022	星期四	2
8194	01/07/2022	星期五	2
8195	01/08/2022	星期六	2
8196	01/09/2022	星期日	3
8197	01/10/2022	星期一	3
8198	01/11/2022	星期二	3
8199	01/12/2022	星期三	3
8200	01/13/2022	星期四	3
8201	01/14/2022	星期五	3

week_number 欄位在前置 LOAD 陳述式中的建立方式是使用 week() 函數，並傳遞 date 欄位，作為函數的引數。

沒有其他參數傳遞至函數，因此下列影響 week() 函數的預設變數已生效：

- BrokenWeeks: 週計數開始於 1 月 1 日
- FirstWeekDay: 每週第一天是星期日

week() 的函數圖表，使用預設系統變數



因為應用程式使用的是預設 BrokenWeeks 系統變數，所以第 1 週開始於 1 月 1 日星期六。

因為預設 FirstWeekDay 系統變數的關係，每週開始於星期六。1 月 1 日的第一個星期日發生在 1 月 2 日，亦即第 2 週開始的時候。

範例 2 – first_week_day

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 建立欄位 week_number，傳回交易發生年份和週數。
- 建立稱為 week_day 的欄位，顯示每個交易日期的工作日值。

在此範例中，我們想要將工作週的開始設定在星期二。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

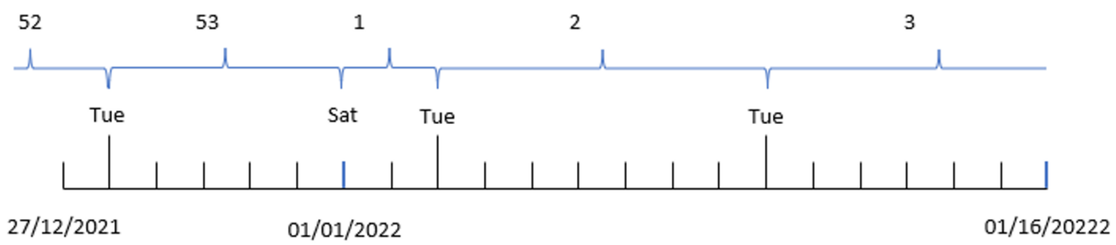
- id
- date
- week_day
- week_number

結果表格

id	日期	week_day	week_number
8183	12/27/2021	星期一	52
8184	12/28/2021	星期二	53
8185	12/29/2021	星期三	53
8186	12/30/2021	星期四	53
8187	12/31/2021	星期五	53
8188	01/01/2022	星期六	1
8189	01/02/2022	星期日	1
8190	01/03/2022	星期一	1
8191	01/04/2022	星期二	2
8192	01/05/2022	星期三	2
8193	01/06/2022	星期四	2
8194	01/07/2022	星期五	2
8195	01/08/2022	星期六	2
8196	01/09/2022	星期日	2
8197	01/10/2022	星期一	2
8198	01/11/2022	星期二	3
8199	01/12/2022	星期三	3
8200	01/13/2022	星期四	3
8201	01/14/2022	星期五	3

應用程式仍使用中斷的週。但是，first_week_day 引數已在 week() 函數中設定為 1。這會將每週第一天設定為星期二。

week() 函數的圖表, first_week_day 範例



應用程式使用的是預設 `BrokenWeeks` 系統變數, 所以第 1 週開始於 1 月 1 日星期六。

`week()` 函數的 `first_week_day` 引數將第一週設定為星期二。因此, 第 53 週開始於 2021 年 12 月 28 日。

但是, 因為函數仍使用中斷的週, 所以第 1 週只會持續兩天, 這是 1 月 1 日後的第 1 個星期二發生在 1 月 3 日的緣故。

範例 3 – unbroken_weeks

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。

在此範例中, 我們使用未中斷的週。

載入指令碼

```
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
```

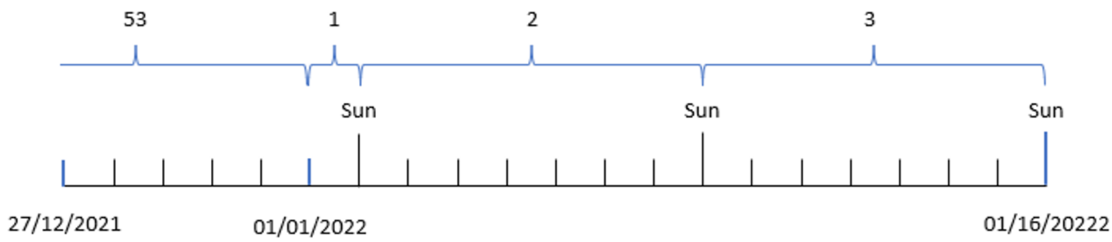
```
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date
- week_day
- week_number

week() 函數的圖表，圖表物件範例



結果表格

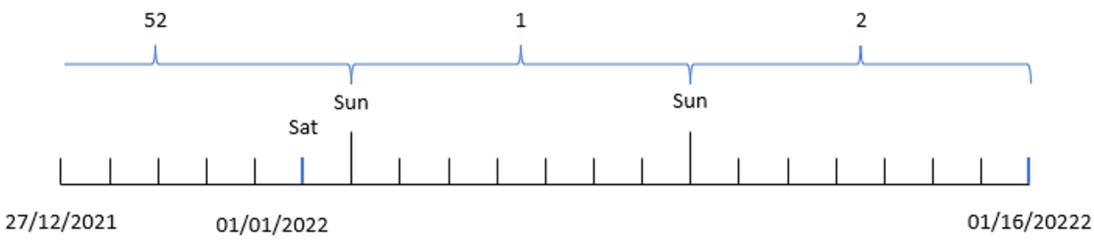
id	日期	week_day	week_number
8183	12/27/2021	星期一	52
8184	12/28/2021	星期二	52
8185	12/29/2021	星期三	52
8186	12/30/2021	星期四	52
8187	12/31/2021	星期五	52

id	日期	week_day	week_number
8188	01/01/2022	星期六	52
8189	01/02/2022	星期日	1
8190	01/03/2022	星期一	1
8191	01/04/2022	星期二	1
8192	01/05/2022	星期三	1
8193	01/06/2022	星期四	1
8194	01/07/2022	星期五	1
8195	01/08/2022	星期六	1
8196	01/09/2022	星期日	2
8197	01/10/2022	星期一	2
8198	01/11/2022	星期二	2
8199	01/12/2022	星期三	2
8200	01/13/2022	星期四	2
8201	01/14/2022	星期五	2

first_week_date 參數設定為 1，讓星期二變成每週第一天。broken_weeks 參數設定為 0，強制函數使用未中斷的週。最後，第三個參數將 reference_day 設定為 2。

`first_week_date` 參數設定為 6，讓星期日變成每週第一天。`broken_weeks` 參數設定為 0，強制函數使用未中斷的週。

week() 函數的圖表，範例使用未中斷的週



透過使用未中斷的週，第 1 週不一定會開始於 1 月 1 日；相反地，必須至少有 4 天。因此，在資料集中，第 52 週結束於 2022 年 1 月 1 日星期六。然後第 1 週開始於 `FirstWeekDay` 系統變數，亦即 1 月 2 日星期日。該週會結束於後面的 1 月 8 日星期日。

範例 4 – reference_day

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 相同資料集和情境為第三個範例。
- 建立欄位 week_number, 傳回交易發生年份和週數。
- 建立稱為 week_day 的欄位, 顯示每個交易日期的工作日值。

此外, 必須符合下列條件:

- 工作週開始於星期二。
- 公司使用未中斷的週。
- reference_day 值是 2。換句話說, 第 1 週在 1 月的最小天數為 2。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

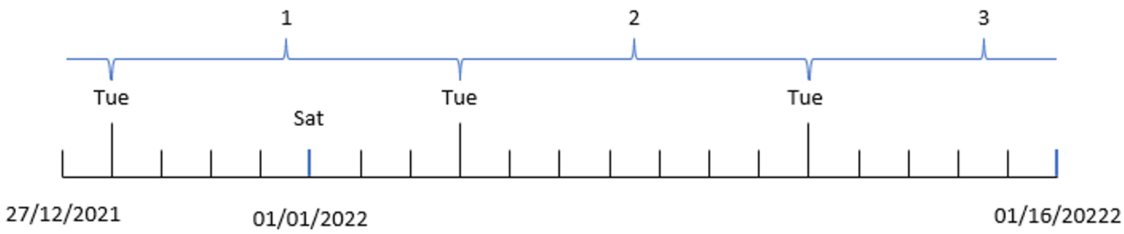
- id
- date
- week_day
- week_number

結果表格

id	日期	week_day	week_number
8183	12/27/2021	星期一	52
8184	12/28/2021	星期二	1
8185	12/29/2021	星期三	1
8186	12/30/2021	星期四	1
8187	12/31/2021	星期五	1
8188	01/01/2022	星期六	1
8189	01/02/2022	星期日	1
8190	01/03/2022	星期一	1
8191	01/04/2022	星期二	2
8192	01/05/2022	星期三	2
8193	01/06/2022	星期四	2
8194	01/07/2022	星期五	2
8195	01/08/2022	星期六	2
8196	01/09/2022	星期日	2
8197	01/10/2022	星期一	2
8198	01/11/2022	星期二	3
8199	01/12/2022	星期三	3
8200	01/13/2022	星期四	3
8201	01/14/2022	星期五	3

first_week_date 參數設定為 1，讓星期二變成每週第一天。broken_weeks 參數設定為 0，強制函數使用未中斷的週。最後，第三個參數將 reference_day 參數設定為 2。

week() 函數的圖表，reference_day example



如果函數使用未中斷的週，且參數為 reference_day 值的 2，則第 1 週只須包括 1 月的兩天。由於第一個工作日為星期二，因此第 1 週開始於 2021 年 12 月 28 日，並結束於 2022 年 1 月 3 日星期一。

範例 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。

不過，在此範例中，不變的資料集會載入到應用程式中。傳回週數的計算在圖表物件中建立為量值。

載入指令碼

```
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
];
```

結果

請執行下列動作:

1. 載入資料並開啟工作表。建立新的表格。
2. 新增下列欄位作為維度:
   - id
   - date
3. 接下來,建立下列量值:
   `=week (date)`
4. 建立量值 , `week_day` 以顯示每個交易日期的工作日值:
   `=weekday(date)`

結果表格

id	日期	=week(date)	=weekday(date)
8183	12/27/2021	53	星期一
8184	12/28/2021	53	星期二
8185	12/29/2021	53	星期三
8186	12/30/2021	53	星期四
8187	12/31/2021	53	星期五
8188	01/01/2022	1	星期六
8189	01/02/2022	2	星期日
8190	01/03/2022	2	星期一
8191	01/04/2022	2	星期二
8192	01/05/2022	2	星期三
8193	01/06/2022	2	星期四
8194	01/07/2022	2	星期五
8195	01/08/2022	2	星期六
8196	01/09/2022	3	星期日
8197	01/10/2022	3	星期一
8198	01/11/2022	3	星期二

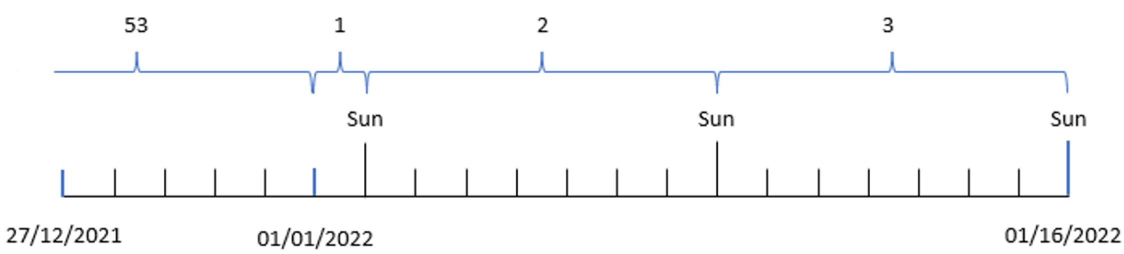
id	日期	=week(date)	=weekday(date)
8199	01/12/2022	3	星期三
8200	01/13/2022	3	星期四
8201	01/14/2022	3	星期五

`week_number` 欄位在前置 LOAD 陳述式中的建立方式是使用 `week()` 函數並傳遞 `date` 欄位,作為函數的引數。

沒有其他參數傳遞至函數,因此下列影響 `week()` 函數的預設變數已生效:

- `BrokenWeeks`: 週計數開始於 1 月 1 日
- `FirstWeekDay`: 每週第一天是星期日

week() 函數的圖表,圖表物件範例



因為應用程式使用的是預設 `BrokenWeeks` 系統變數,所以第 1 週開始於 1 月 1 日星期六。

因為預設 `FirstWeekDay` 系統變數的關係,每週開始於星期六。1 月 1 日的第一個星期日發生在 1 月 2 日,亦即第 2 週開始的時候。

範例 6 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2019 年最後一週和 2020 年前兩週交易的資料集,這載入到稱為「`Transactions`」的表格中。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。

應用程式在儀表板之間主要使用中斷的週。但是,最終使用者需要使用未中斷的週按週呈現總銷售額的圖表物件。參考日應為 1 月 2 日,且每週開始於星期二。當資料模型中無法使用此維度時,在圖表中將 `week()` 函數作為計算維度可達成這一點。

載入指令碼

```
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
];
```

結果

請執行下列動作:

1. 載入資料並開啟工作表。建立新的表格。
2. 建立下列計算維度:
   =week(date)
3. 接下來,建立下列彙總量值:
   =sum(amount)
4. 將量值的**數字格式**設定為**金錢**。
5. 選擇「**排序**」功能表,並針對計算維度移除自訂排序。
6. 取消選取「**按數字排序**」和「**按字母排序**」選項。

結果表格

week(date)	sum(amount)
52	$125.69

week(date)	sum(amount)
53	$146.42
1	$200.09
2	$347.57
3	$122.01

weekday

此函數會傳回函下列項目的雙值：

- 如環境變數 **DayNames** 中所定義的日名稱。
- 介於 0 至 6 之間且對應於一週中正常日 (0-6) 的整數。

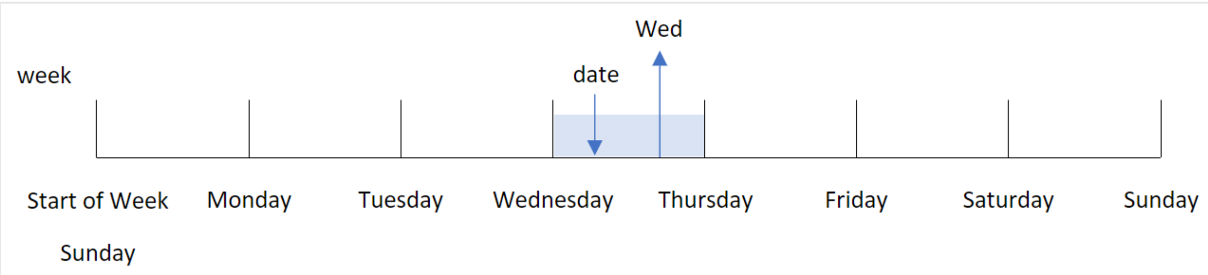
語法：

```
weekday(date [,first_week_day=0])
```

傳回的資料類型： 雙值

weekday() 函數判定日期落在哪個星期幾。然後傳回代表該日子的字串值。

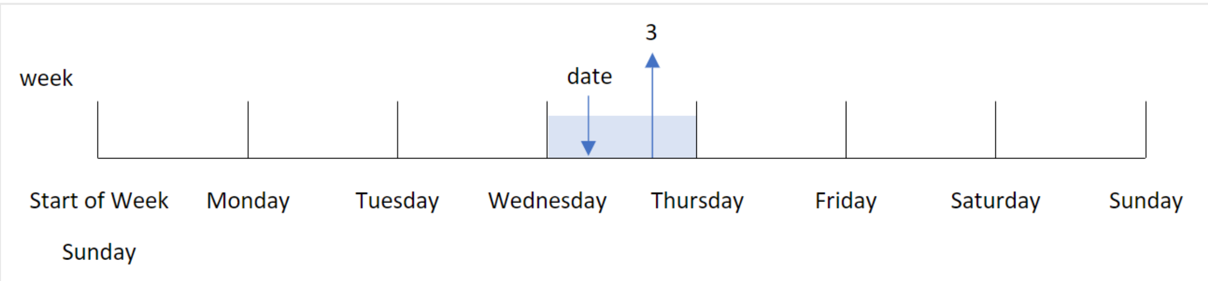
weekday() 函數的圖表，其傳回日期所在的日子名稱



根據一週的開始日期，結果會傳回與星期幾對應的數字值 (0-6)。例如，如果每週第一天設為星期日，星期三即會傳回 3 的數字值。此開始日期由 FirstWeekDay 系統變數，或是 first_week_day 函數參數判定。

您可以使用此數值作為算術運算式的一部分。例如，乘以 1 可傳回值本身。

weekday() 函數的圖表，且具有顯示日子的數字值，而不是日子名稱



什麼情況下使用

若您想要按星期幾比較會總，則 `weekday()` 函數很實用。例如，若您想按工作日比較產品平均銷售額。

這些維度在載入指令碼建立時可使用函數建立**主要行事曆**表格的欄位；或是直接在圖表中建立為計算量值。

相關主題

主題	互動
FirstWeekDay (page 212)	定義每週開始的日子。

引數

引數	描述
date	要評估的時間戳記。
first_week_day	指定一週開始的日期。如果忽略，將使用變數 **FirstWeekDay** 的值。 *FirstWeekDay (page 212)*

您可以在 `first_week_day` 引數中使用下列值設定一週開始的日期：

first_week_day 值

日	值
星期一	0
星期二	1
星期三	2
星期四	3
星期五	4
星期六	5
星期日	6

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典, 資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

除非另有說明, 否則在這些範例中 `FirstWeekDay` 會設為 0。

函數範例

範例	結果
`weekday('10/12/1971')`	傳回 'Tue' 和 1。
`weekday('10/12/1971' , 6)`	傳回 'Tue' 和 2。 在此範例中, 星期日 (6) 是一週的第一天。

`SET FirstWeekDay=6;`
`...`
 	
`weekday('10/12/1971')`
	傳回 'Tue' 和 2。

範例 1 - 工作日字串

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2022 年交易的資料集, 這載入到稱為「`Transactions`」的表格中。
- 設定為 6 (星期日) 的 `FirstWeekDay` 系統變數。
- 設為使用預設日期名稱的 `DayNames` 變數。
- 包含 `weekday()` 函數的前置載入, 且設為「`week_day`」欄位並傳回交易發生的工作日。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date
- week_day

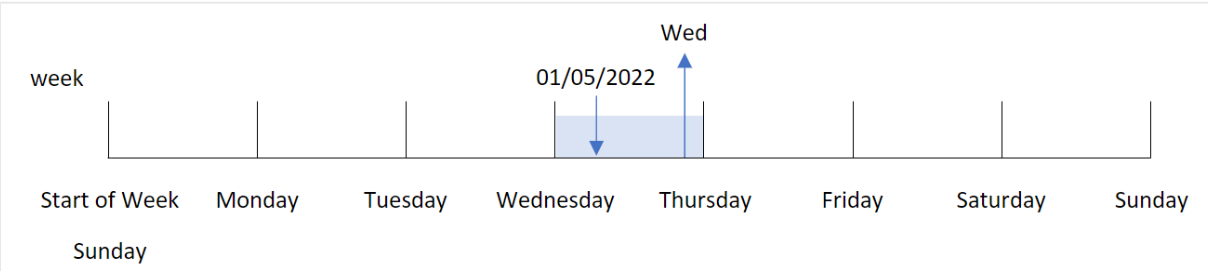
結果表格

id	日期	week_day
8188	01/01/2022	星期六
8189	01/02/2022	星期日
8190	01/03/2022	星期一
8191	01/04/2022	星期二
8192	01/05/2022	星期三
8193	01/06/2022	星期四
8194	01/07/2022	星期五

「week_day」欄位在前置 LOAD 陳述式中的建立方式是使用 weekday() 函數並傳遞日期欄位，作為函數的引數。

weekday() 函數傳回工作日字串值；亦即傳回由 DayNames 系統變數設定的工作日名稱。

weekday() 函數的圖表，並為交易 8192 工作日傳回星期三



交易 8192 發生在 1 月 5 日。FirstWeekDay 系統變數會將每週第一天設為星期日。weekday() 函數交易發生在星期三並傳回此值，且為 DayNames 系統變數的縮寫形式並位於 week_day 欄位。

「week_day」欄位中的值在資料行中向右對齊，因為欄位有雙值數字和文字結果 (星期三、3)。若要將欄位值轉換為等同數字的值，欄位即可在 num() 函數內換行。例如在交易 8192 中，星期三值會轉換為數字 3。

範例 2 – first_week_day

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為 Transactions 的表格中。
- 設定為 6 (星期日) 的 FirstWeekDay 系統變數。
- 設為使用預設日期名稱的 DayNames 變數。
- 包含 weekday() 函數的先前載入，且設為「week_day」欄位並傳回交易發生的工作日。

載入指令碼

```
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
];
```

結果

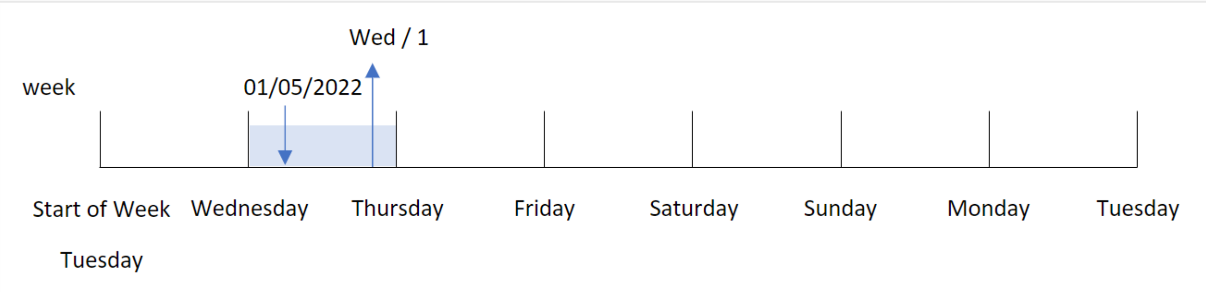
載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date
- week_day

結果表格

id	日期	week_day
8188	01/01/2022	星期六
8189	01/02/2022	星期日
8190	01/03/2022	星期一
8191	01/04/2022	星期二
8192	01/05/2022	星期三
8193	01/06/2022	星期四
8194	01/07/2022	星期五

weekday() 函數的圖表，顯示星期三有 1 的雙值數字值



因為 `first_week_day` 引數在 `weekday()` 函數中設為 `1`，所以每週第一天是星期二。因此，所有發生在星期二的交易會有 `0` 的雙值數字值。

交易 8192 發生在 1 月 5 日。`weekday()` 函數識別這是星期三，所以運算式會傳回 `1` 的雙值數字值。

範例 3 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為 `Transactions` 的表格中。
- 設為 6 (星期日) 的 `FirstWeekDay` 系統變數。
- 設為使用預設日期名稱的 `DayNames` 變數。

不過，在此範例中，資料集保持不變並且會載入到應用程式中。識別工作日值的計算在應用程式圖表中建立為量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date

若要計算工作日值，請建立下列量值：

- =weekday(date)

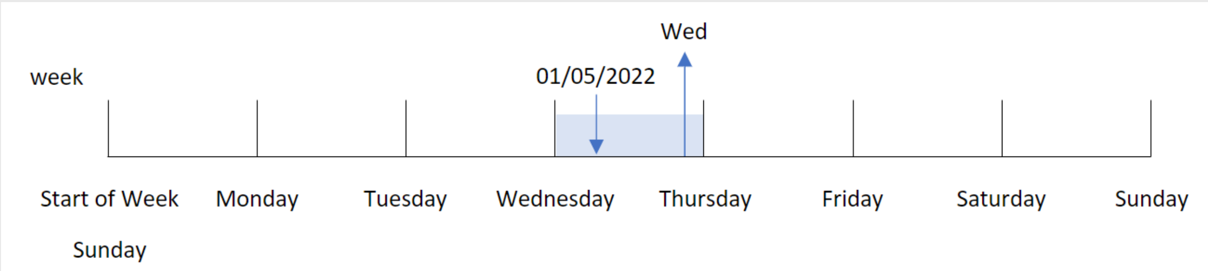
結果表格

id	日期	=weekday(date)
8188	01/01/2022	星期六
8189	01/02/2022	星期日
8190	01/03/2022	星期一
8191	01/04/2022	星期二
8192	01/05/2022	星期三
8193	01/06/2022	星期四
8194	01/07/2022	星期五

「=weekday(date)」欄位在圖表中的建立方式是使用 weekday() 函數並傳遞日期欄位，作為函數的引數。

weekday() 函數傳回工作日字串值;亦即傳回由 DayNames 系統變數設定的工作日名稱。

weekday() 函數的圖表,並為交易 8192 工作日傳回星期三



交易 8192 發生在 1 月 5 日。FirstWeekDay 系統變數會將每週第一天設為星期日。weekday() 函數交易發生在星期三並傳回此值,且為 DayNames 系統變數的縮寫形式並位於 =weekday(date) 欄位。

範例 4 – 情境

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2022 年交易的資料集,這載入到稱為 Transactions 的表格中。
- 設為 6 (星期日) 的 FirstWeekDay 系統變數。
- 設為使用預設日期名稱的 DayNames 變數。

最終使用者希望圖表按工作日呈現交易的平均銷售額。

載入指令碼

```
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);
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- =weekday(date)
- =avg(amount)

將量值的**數字格式**設定為**金錢**。

結果表格

=weekday(date)	Avg(amount)
星期日	$536.96
星期一	$500.80
星期二	$515.63
星期三	$509.21
星期四	$482.70
星期五	$441.33
星期六	$505.22

weekend

此功能傳回的值相當於包含 **date** 的該行事曆週最後一天的最後一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 **DateFormat**。

語法：

```
WeekEnd(timestamp [, period_no [, first_week_day ]])
```

傳回的資料類型： 雙值

weekend() 函數判定日期落在哪一週。然後以日期格式傳回該週最後一毫秒的時間戳記。該週的第一天由 FirstWeekDay 環境變數決定。但是，這可由 weekend() 函數中的 first_week_day 引數取代。

引數

引數	描述
timestamp	要評估的時間戳記。
period_no	**shift** 為整數，值 0 表示是包含 **date** 的週。負值的 shift 表示之前的週，正值表示之後的週。
first_week_day	指定一週開始的日期。如果忽略，將使用變數 **FirstWeekDay** 的值。 **first_week_day** 可能的值是對星期一使用 0、對星期二使用 1，對星期三使用 2，對星期四使用 3，對星期五使用 4，對星期六使用 5，並對星期日使用 6。 如需系統變數的更多資訊，請參閱 *FirstWeekDay (page 212)*

什麼情況下使用

weekend() 函數通常在使用者想要計算使用特定日期的每週剩餘日子時，作為運算式的一部分使用。例如，若使用者想要計算該週期間尚未發生的總利息，即可使用此函數。

下列範例假設：

```
SET FirstWeekDay=0;
```

範例	結果
weekend('01/10/2013')	傳回 01/12/2013 23:59:59。
weekend('01/10/2013', -1)	傳回 01/05/2013 23:59:59.。
weekend('01/10/2013', 0, 1)	傳回 01/14/2013 23:59:59。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例：

若您想要週和週數的 ISO 設定，確認指令碼中有下列內容：

```
Set DateFormat   ='YYYY-MM-DD';
Set FirstWeekDay =0;    // Monday as first week day
Set BrokenWeeks  =0;    //(use unbroken weeks)
Set ReferenceDay =4;    // Jan 4th is always in week 1
```

若您想要 US 設定，確認指令碼中有下列內容：

```
Set DateFormat   ='M/D/YYYY';
Set FirstWeekDay =6;    // Sunday as first week day
Set BrokenWeeks  =1;    //(use broken weeks)
Set ReferenceDay =1;    // Jan 1st is always in week 1
```

以上範例從 weekend() 函數產生下列內容：

Weekend 函數範例

日期	ISO 週結束	US 週結束
Sat 2020 Dec 26	2020-12-27	12/26/2020

日期	ISO 週結束	US 週結束
Sun 2020 Dec 27	2020-12-27	1/2/2021
Mon 2020 Dec 28	2021-01-03	1/2/2021
Tue 2020 Dec 29	2021-01-03	1/2/2021
Wed 2020 Dec 30	2021-01-03	1/2/2021
Thu 2020 Dec 31	2021-01-03	1/2/2021
Fri 2021 Jan 1	2021-01-03	1/2/2021
Sat 2021 Jan 2	2021-01-03	1/2/2021
Sun 2021 Jan 3	2021-01-03	1/9/2021
Mon 2021 Jan 4	2021-01-10	1/9/2021
Tue 2021 Jan 5	2021-01-10	1/9/2021

在 ISO 欄中，一週在星期日結束，而在 US 欄中，在星期六結束。

範例 1 – 基本範例

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為 `Transactions` 的表格中。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 `end_of_week`，這傳回交易發生時每週結束的時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- date
- end_of_week
- end_of_week_timestamp

結果表格

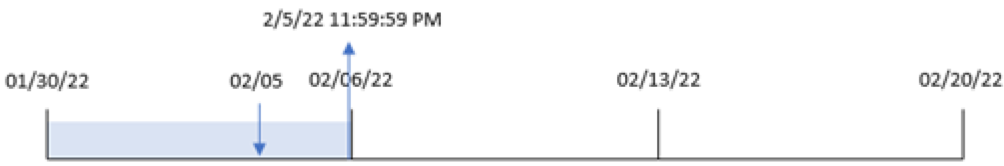
日期	end_of_week	end_of_week_timestamp
1/7/2022	01/08/2022	1/8/2022 11:59:59 PM
1/19/2022	01/22/2022	1/22/2022 11:59:59 PM
2/5/2022	02/05/2022	2/5/2022 11:59:59 PM
2/28/2022	03/05/2022	3/5/2022 11:59:59 PM
3/16/2022	03/19/2022	3/19/2022 11:59:59 PM
4/1/2022	04/02/2022	4/2/2022 11:59:59 PM
5/7/2022	05/07/2022	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

日期	end_of_week	end_of_week_timestamp
7/22/2022	07/23/2022	7/23/2022 11:59:59 PM
7/23/2022	07/23/2022	7/23/2022 11:59:59 PM
7/27/2022	07/30/2022	7/30/2022 11:59:59 PM
8/2/2022	08/06/2022	8/6/2022 11:59:59 PM
8/8/2022	08/13/2022	8/13/2022 11:59:59 PM
8/19/2022	08/20/2022	8/20/2022 11:59:59 PM
9/26/2022	10/01/2022	10/1/2022 11:59:59 PM
10/14/2022	10/15/2022	10/15/2022 11:59:59 PM
10/29/2022	10/29/2022	10/29/2022 11:59:59 PM

end_of_week 欄位在前置 LOAD 陳述式中的建立方式是使用 weekend() 函數並傳遞日期欄位，作為函數的引數。

weekend() 函數識別日期值落在哪一週，並傳回該週最後一毫秒的時間戳記。

weekend() 函數的圖表，基本範例



交易 8191 發生在 2 月 5 日。FirstWeekDay 系統變數會將每週第一天設為星期日。weekend() 函數識別 2 月 5 日後的第一個星期六 – 因此是每週的結束 – 落在 2 月 5 日。因此，該交易的 end_of_week 值會傳回該日子的最後一毫秒，亦即 2 月 5 日下午 11:59:59。

範例 2 – period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 建立欄位 previous_week_end，這傳回交易發生之前每週開始的時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- previous_week_end
- previous_week_end_timestamp

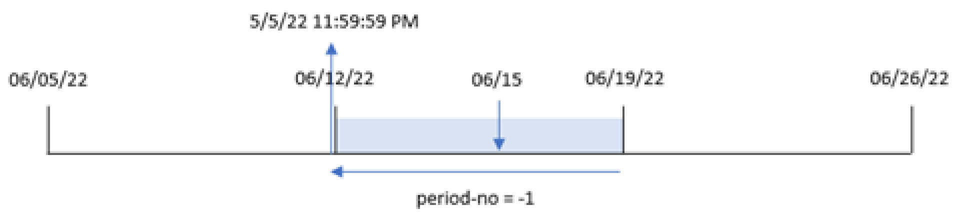
結果表格

日期	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

日期	end_of_week	end_of_week_timestamp
4/1/2022	03/26/2022	3/26/2022 11:59:59 PM
5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
5/16/2022	05/14/2022	5/14/2022 11:59:59 PM
6/15/2022	06/11/2022	6/11/2022 11:59:59 PM
6/26/2022	06/25/2022	6/25/2022 11:59:59 PM
7/9/2022	07/02/2022	7/2/2022 11:59:59 PM
7/22/2022	07/16/2022	7/16/2022 11:59:59 PM
7/23/2022	07/16/2022	7/16/2022 11:59:59 PM
7/27/2022	07/23/2022	7/23/2022 11:59:59 PM
8/2/2022	07/30/2022	7/30/2022 11:59:59 PM
8/8/2022	08/06/2022	8/6/2022 11:59:59 PM
8/19/2022	08/13/2022	8/13/2022 11:59:59 PM
9/26/2022	09/24/2022	9/24/2022 11:59:59 PM
10/14/2022	10/08/2022	10/8/2022 11:59:59 PM
10/29/2022	10/22/2022	10/22/2022 11:59:59 PM

在此例中，因為 period_no 的值 -1 已作為 weekend() 函數中的偏移引數使用，所以該函數首先會識別交易發生的週。然後這會查看前一週並識別該週的最後一毫秒。

weekend() 函數的圖表，period_no 範例



交易 8196 發生在 6 月 15 日。weekend() 函數識別該週開始於 6 月 12 日。因此，前一週結束於 6 月 11 日 11:59:59 PM；這是 previous_week_end 欄位傳回的值。

範例 3 – first_week_day

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。但是在此範例中,我們必須將星期二設為工作週第一天。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

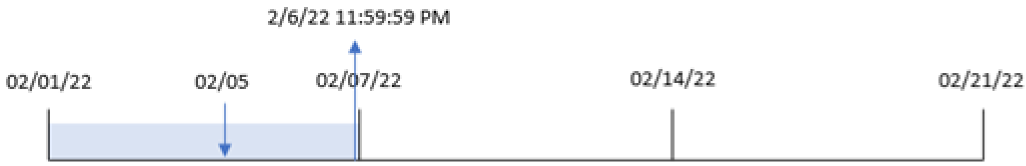
- date
- end_of_week
- end_of_week_timestamp

結果表格

日期	end_of_week	end_of_week_timestamp
1/7/2022	01/10/2022	1/10/2022 11:59:59 PM
1/19/2022	01/24/2022	1/24/2022 11:59:59 PM
2/5/2022	02/07/2022	2/7/2022 11:59:59 PM
2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
3/16/2022	03/21/2022	3/21/2022 11:59:59 PM
4/1/2022	04/04/2022	4/4/2022 11:59:59 PM
5/7/2022	05/09/2022	5/9/2022 11:59:59 PM
5/16/2022	05/16/2022	5/16/2022 11:59:59 PM
6/15/2022	06/20/2022	6/20/2022 11:59:59 PM
6/26/2022	06/27/2022	6/27/2022 11:59:59 PM
7/9/2022	07/11/2022	7/11/2022 11:59:59 PM
7/22/2022	07/25/2022	7/25/2022 11:59:59 PM
7/23/2022	07/25/2022	7/25/2022 11:59:59 PM
7/27/2022	08/01/2022	8/1/2022 11:59:59 PM
8/2/2022	08/08/2022	8/8/2022 11:59:59 PM
8/8/2022	08/08/2022	8/8/2022 11:59:59 PM
8/19/2022	08/22/2022	8/22/2022 11:59:59 PM
9/26/2022	09/26/2022	9/26/2022 11:59:59 PM
10/14/2022	10/17/2022	10/17/2022 11:59:59 PM
10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

在此例中，因為 `first_week_date` 引數的值 1 已用於 `weekend()` 函數中，所以這會將每週第一天設為星期二。

weekend() 函數的圖表，first_week_day 範例



交易 8191 發生在 2 月 5 日。`weekend()` 函數識別此日期後的第一個星期一 – 因此是每週的結束並傳回該值 – 落在 2 月 6 日 11:59:59 PM。

範例 4 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。不過，在此範例中，不變的資料集會載入到應用程式中。會建立傳回交易發生週結束時間戳記的計算，作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：date。

若要計算交易發生的該週開始時間，新增下列量值：

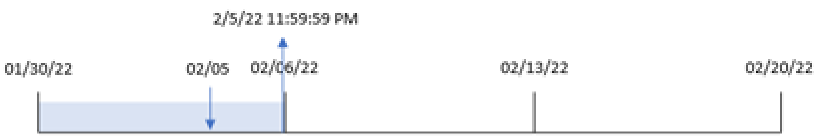
- =weekend(date)
- =timestamp(weekend(date))

結果表格

日期	=weekend(date)	=timestamp(weekend(date))
1/7/2022	01/08/2022	1/8/2022 11:59:59 PM
1/19/2022	01/22/2022	1/22/2022 11:59:59 PM
2/5/2022	02/05/2022	2/5/2022 11:59:59 PM
2/28/2022	03/05/2022	3/5/2022 11:59:59 PM
3/16/2022	03/19/2022	3/19/2022 11:59:59 PM
4/1/2022	04/02/2022	4/2/2022 11:59:59 PM
5/7/2022	05/07/2022	5/7/2022 11:59:59 PM
5/16/2022	05/21/2022	5/21/2022 11:59:59 PM
6/15/2022	06/18/2022	6/18/2022 11:59:59 PM
6/26/2022	07/02/2022	7/2/2022 11:59:59 PM
7/9/2022	07/09/2022	7/9/2022 11:59:59 PM
7/22/2022	07/23/2022	7/23/2022 11:59:59 PM
7/23/2022	07/23/2022	7/23/2022 11:59:59 PM
7/27/2022	07/30/2022	7/30/2022 11:59:59 PM
8/2/2022	08/06/2022	8/6/2022 11:59:59 PM
8/8/2022	08/13/2022	8/13/2022 11:59:59 PM
8/19/2022	08/20/2022	8/20/2022 11:59:59 PM
9/26/2022	10/01/2022	10/1/2022 11:59:59 PM
10/14/2022	10/15/2022	10/15/2022 11:59:59 PM
10/29/2022	10/29/2022	10/29/2022 11:59:59 PM

end_of_week 量值在圖表物件中的建立方式是使用 weekend() 函數並傳遞日期欄位，作為函數的引數。weekend() 函數識別日期值落在哪一週，並傳回該週最後一毫秒的時間戳記。

weekend() 函數的圖表，圖表物件範例



交易 8191 發生在 2 月 5 日。FirstWeekDay 系統變數會將每週第一天設為星期日。weekend() 函數識別 2 月 5 日後的第一個星期六 – 因此是每週的結束 – 落在 2 月 5 日。因此，該交易的 end_of_week 值會傳回該日子的最後一毫秒，亦即 2 月 5 日下午 11:59:59。

範例 5 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 載入到稱為 `Employee_Expenses` 之表格的資料集。
- 由員工 ID、員工姓名及每個員工報銷的平均每日開支組成的資料。

最終使用者希望圖表物件按員工 ID 和員工名稱顯示該週剩餘期間仍會產生的預估開支報銷。

載入指令碼

```
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
];
```

結果

請執行下列動作:

1. 載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:
   - `employee_id`
   - `employee_name`
2. 接下來,建立量值以計算累積的利息:
   `=(weekend(today(1))-today(1))*avg_daily_claim`
3. 將量值的**數字格式**設定為**金錢**。

結果表格

employee_id	employee_name	=(weekend(today(1))-today(1))*avg_daily_claim
182	Mark	$90.00
183	Deryck	$75.00
184	Dexter	$75.00

employee_id	employee_name	=(weekend(today(1))-today(1))*avg_daily_claim
185	Sydney	$162.00
186	Agatha	$108.00

藉由使用今日日期作為其唯一函數，`weekend()` 函數會傳回目前週的結束日期。然後，以該週結束日期減去今日日期，運算式就會傳回該週剩餘天數。

然後按員工讓此值乘以平均每日開支報銷，以計算每個員工在該週剩餘期間預期會有的預估報銷值。

weekname

此函數會傳回顯示年度和週數的值，其基礎數值相當於包含 **date** 之週的第一天、第一毫秒的時間戳記。

語法：

```
WeekName(date[, period_no [, first_week_day [, broken_weeks [, reference_day]]]])
```

`weekname()` 函數判定日期落在哪一週並傳回週數和該週的年份。該週的第一天由 `FirstWeekDay` 系統變數決定。不過，您可以使用 `weekname()` 函數中的 `first_week_day` 引數變更該週的第一天。

在 Qlik Sense 中，建立應用程式時會擷取地區設定，對應的設定會儲存在指令碼中，作為環境變數。

北美應用程式開發人員通常會在指令碼中取得 `Set BrokenWeeks=1;`，對應至中斷的週。歐洲應用程式開發人員通常會在指令碼中取得 `Set BrokenWeeks=0;`，對應至未中斷的週。

若應用程式使用中斷的週，則週數計數在 1 月 1 日開始，並在 `FirstWeekDay` 系統變數的前一天結束，無論發生了多少天。

不過，若應用程式正在使用未中斷的週，則第 1 週可以在前一年或 1 月的開頭幾天開始。這取決於您如何使用 `ReferenceDay` 和 `FirstWeekDay` 系統變數。

Weekname 函數範例

日期	ISO 週名稱	US 週名稱
Sat 2020 Dec 26	2020/52	2020/52
Sun 2020 Dec 27	2020/52	2020/53
Mon 2020 Dec 28	2020/53	2020/53
Tue 2020 Dec 29	2020/53	2020/53
Wed 2020 Dec 30	2020/53	2020/53
Thu 2020 Dec 31	2020/53	2020/53
Fri 2021 Jan 1	2020/53	2021/01
Sat 2021 Jan 2	2020/53	2021/01

日期	ISO 週名稱	US 週名稱
Sun 2021 Jan 3	2020/53	2021/02
Mon 2021 Jan 4	2021/01	2021/02
Tue 2021 Jan 5	2021/01	2021/02

什麼情況下使用

若您想要按週比較彙總，則 `weekname()` 函數很實用。

例如，若您想要按週查看產品的總銷售額。若要在應用程式中維護與 `BrokenWeeks` 環境變數的一致性，請使用 `weekname()` 而非 `lunarweekname()`。若應用程式正在使用未中斷的週，則第 1 週可能包含上一年 12 月中的日期，或排除目前年份 1 月的日期。若應用程式正在使用中斷的週，則第 1 週的天數可少於七天。

傳回的資料類型： 雙值

引數

引數	描述
timestamp	要評估的時間戳記。
period_no	**shift** 為整數，值 0 表示是包含 **date** 的週。負值的 shift 表示之前的週，正值表示之後的週。
first_week_day	指定一週開始的日期。如果忽略，將使用變數 **FirstWeekDay** 的值。 **first_week_day** 可能的值是對星期一使用 0、對星期二使用 1，對星期三使用 2，對星期四使用 3，對星期五使用 4，對星期六使用 5，並對星期日使用 6。 如需系統變數的更多資訊，請參閱 *FirstWeekDay (page 212)*。
broken_weeks	如果您未指定 **broken_weeks**，則變數 **BrokenWeeks** 的值將用於定義週是否中斷。
reference_day	如果您不指定 **reference_day**，變數 **ReferenceDay** 的值將用於定義要設定一月份的哪一天為參照日以定義第 1 週。依照預設，Qlik Sense 函數使用 4 作為參照日。這表示第 1 週必須包含 1 月 4 日，或者換言之，第 1 週必須始終至少在 1 月擁有 4 日。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

以下範例假設：

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

函數範例

範例	結果
weekname('01/12/2013')	傳回 2013/02。
weekname('01/12/2013', -1)	傳回 2013/01。
weekname('01/12/2013', 0, 1)	傳回 2013/02。

範例 1 – 無其他引數的日期

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2021 年最後一週和 2022 年前兩週交易的資料集，這載入到稱為「Transactions」的表格中。
- 設定為 MM/DD/YYYY 格式的 DateFormat 系統變數。
- 設定為 1 的 BrokenWeeks 系統變數。
- 設定為 6 的 FirstWeekDay 系統變數。
- 前置載入包含下列內容：
  - 設定為欄位「week_number」的 weekday() 函數，傳回交易發生年份和週數。
  - 設定為稱為「week_day」之欄位的 weekname() 函數，以顯示每個交易日期的星期值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date
- week_day
- week_number

結果表格

id	日期	week_day	week_number
8183	12/27/2021	星期一	2021/53
8184	12/28/2021	星期二	2021/53
8185	12/29/2021	星期三	2021/53
8186	12/30/2021	星期四	2021/53
8187	12/31/2021	星期五	2021/53
8188	01/01/2022	星期六	2022/01
8189	01/02/2022	星期日	2022/02

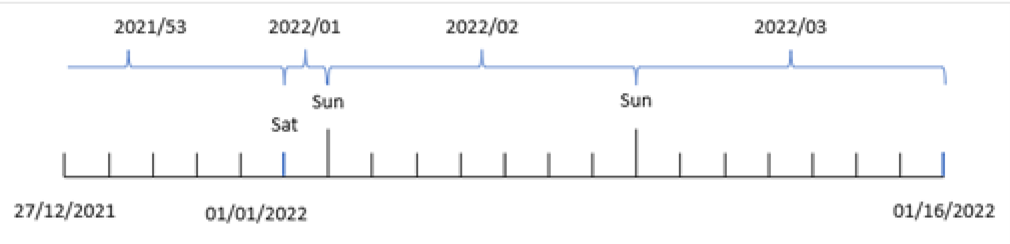
id	日期	week_day	week_number
8190	01/03/2022	星期一	2022/02
8191	01/04/2022	星期二	2022/02
8192	01/05/2022	星期三	2022/02
8193	01/06/2022	星期四	2022/02
8194	01/07/2022	星期五	2022/02
8195	01/08/2022	星期六	2022/02
8196	01/09/2022	星期日	2022/03
8197	01/10/2022	星期一	2022/03
8198	01/11/2022	星期二	2022/03
8199	01/12/2022	星期三	2022/03
8200	01/13/2022	星期四	2022/03
8201	01/14/2022	星期五	2022/03

「week_number」欄位在前置 LOAD 陳述式中的建立方式是使用 weekname() 函數並傳遞日期欄位，作為函數的引數。

weekname() 函數最初識別日期值落在哪一週並傳回交易發生的週數計數和年份。

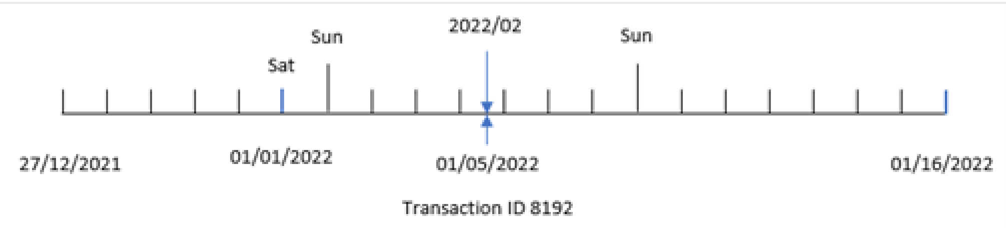
FirstWeekDay 系統變數將星期日設為該週的第一天。BrokenWeeks 系統變數設定應用程式以使用中斷的週，表示第 1 週將從 1 月 1 日開始。

具有預設變數的 weekname() 函數圖表。



第 1 週從 1 月 1 日開始，這天是星期六，因此發生在此日期的交易會傳回值 2022/01 (年份和週數)。

weekname() 函數的圖表識別交易 8192 的週數。



因為應用程式使用中斷的週且一週的第一天是星期日,所以從 1 月 2 日到 1 月 8 日發生的交易會傳回值 2022/02 (2022 年第 2 週)。這種情況的範例會是發生在 1 月 5 日的交易 8192 並傳回「week_number」欄位的值 2022/02。

範例 2 – period_no

載入指令碼和結果

概覽

使用與第一個範例相同的資料集和情境。

不過,在此範例中,任務是要建立欄位「previous_week_number」,這傳回交易發生之前的年份和週數。

開啟 資料載入編輯器 並將以下載入指令碼新增至新的索引標籤。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

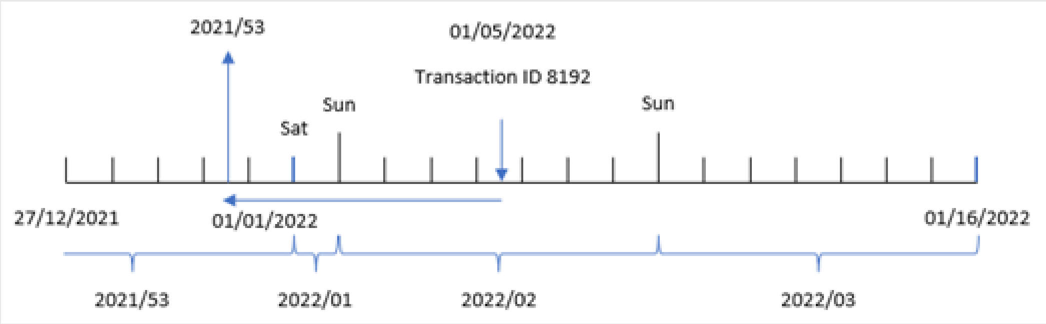
- id
- date
- week_day
- week_number

結果表格

id	日期	week_day	week_number
8183	12/27/2021	星期一	2021/52
8184	12/28/2021	星期二	2021/52
8185	12/29/2021	星期三	2021/52
8186	12/30/2021	星期四	2021/52
8187	12/31/2021	星期五	2021/52
8188	01/01/2022	星期六	2021/52
8189	01/02/2022	星期日	2021/53
8190	01/03/2022	星期一	2021/53
8191	01/04/2022	星期二	2021/53
8192	01/05/2022	星期三	2021/53
8193	01/06/2022	星期四	2021/53
8194	01/07/2022	星期五	2021/53
8195	01/08/2022	星期六	2022/01
8196	01/09/2022	星期日	2022/02
8197	01/10/2022	星期一	2022/02
8198	01/11/2022	星期二	2022/02
8199	01/12/2022	星期三	2022/02
8200	01/13/2022	星期四	2022/02
8201	01/14/2022	星期五	2022/02

因為 `period_no` 的值 `-1` 作為 `weekname()` 函數中的偏移引數使用，所以該函數首先會識別交易發生的週。然後這會查看前一週並識別該週的第一毫秒。

具有 period_no 偏移值 -1 的 weekname() 函數圖表。



交易 8192 發生在 2022 年 1 月 5 日。weekname() 函數查看前一週，即 2021 年 12 月 30 日，並傳回該日期的週數和年份 – 2021/53。

範例 3 – first_week_day

載入指令碼和結果

概覽

使用與第一個範例相同的資料集和情境。

不過，在此範例中，公司政策適用於在星期二開始的工作週。

開啟 資料載入編輯器 並將以下載入指令碼新增至新的索引標籤。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

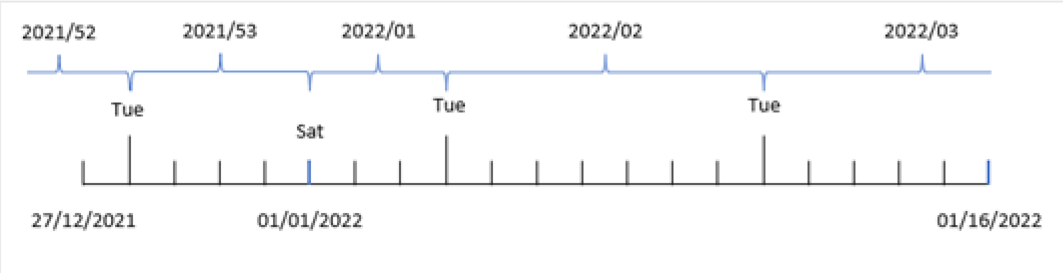
- `id`
- `date`
- `week_day`
- `week_number`

結果表格

id	日期	week_day	week_number
8183	12/27/2021	星期一	2021/52
8184	12/28/2021	星期二	2021/53
8185	12/29/2021	星期三	2021/53
8186	12/30/2021	星期四	2021/53
8187	12/31/2021	星期五	2021/53
8188	01/01/2022	星期六	2022/01
8189	01/02/2022	星期日	2022/01
8190	01/03/2022	星期一	2022/01
8191	01/04/2022	星期二	2022/02
8192	01/05/2022	星期三	2022/02
8193	01/06/2022	星期四	2022/02
8194	01/07/2022	星期五	2022/02
8195	01/08/2022	星期六	2022/02
8196	01/09/2022	星期日	2022/02
8197	01/10/2022	星期一	2022/02
8198	01/11/2022	星期二	2022/03
8199	01/12/2022	星期三	2022/03

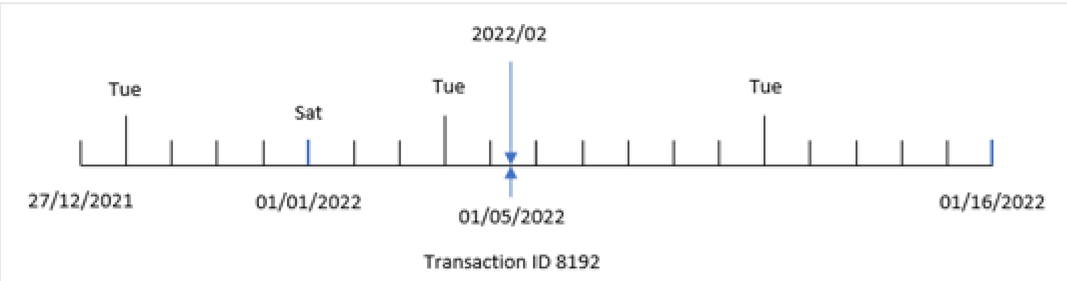
id	日期	week_day	week_number
8200	01/13/2022	星期四	2022/03
8201	01/14/2022	星期五	2022/03

weekname() 函數圖表以星期二作為該週的第一天。



因為 1 的 first_week_date 引數用於 weekname() 函數，所以這使用星期二作為該週的第一天。因此該函數判定 2021 年的第 53 週從 12 月 28 日星期二開始；此外，由於應用程式使用中斷的週，第 1 週於 2022 年 1 月 1 日開始，並於 2022 年 1 月 3 日星期一的最後一毫秒結束。

圖表顯示交易 8192 的週數並以星期二作為一週的第一天。



交易 8192 發生在 2022 年 1 月 5 日。因此，使用星期二的 first_week_day 參數，weekname() 函數會傳回「week_number」欄位的值 2022/02。

範例 4 – 圖表物件範例

載入指令碼和圖表運算式

概覽

使用與第一個範例相同的資料集和情境。

不過，在此範例中，資料集保持不變並且會載入到應用程式中。會建立傳回交易發生週數年份的計算，作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- id
- date
- =week_day (date)

若要計算交易發生的該週開始時間, 建立下列量值:

=weekname(date)

結果表格

id	日期	=weekday(date)	=weekname(date)
8183	12/27/2021	星期一	2021/53
8184	12/28/2021	星期二	2021/53
8185	12/29/2021	星期三	2021/53
8186	12/30/2021	星期四	2021/53
8187	12/31/2021	星期五	2021/53
8188	01/01/2022	星期六	2022/01
8189	01/02/2022	星期日	2022/02

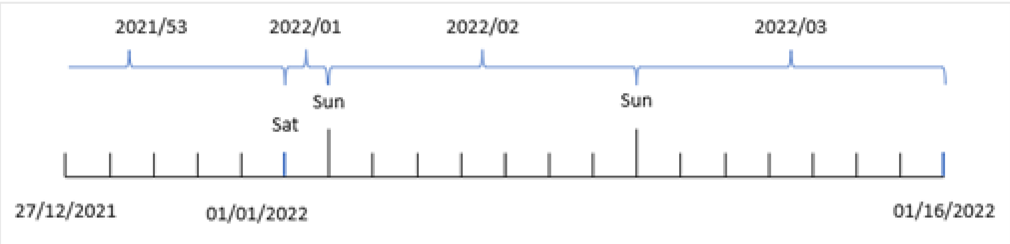
id	日期	=weekday(date)	=weekname(date)
8190	01/03/2022	星期一	2022/02
8191	01/04/2022	星期二	2022/02
8192	01/05/2022	星期三	2022/02
8193	01/06/2022	星期四	2022/02
8194	01/07/2022	星期五	2022/02
8195	01/08/2022	星期六	2022/02
8196	01/09/2022	星期日	2022/03
8197	01/10/2022	星期一	2022/03
8198	01/11/2022	星期二	2022/03
8199	01/12/2022	星期三	2022/03
8200	01/13/2022	星期四	2022/03
8201	01/14/2022	星期五	2022/03

「week_number」欄位在圖表物件中的建立為量值方式是使用 weekname() 函數並傳遞日期欄位，作為函數的引數。

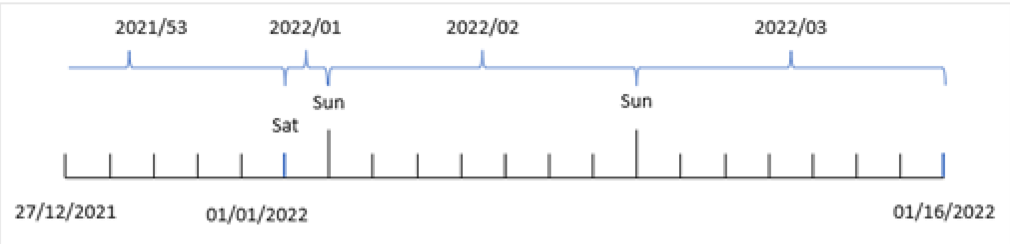
weekname() 函數最初識別日期值落在哪一週並傳回交易發生的週數計數和年份。

FirstWeekDay 系統變數將星期日設為該週的第一天。BrokenWeeks 系統變數設定應用程式以使用中斷的週，表示第 1 週從 1 月 1 日開始。

圖表顯示以星期日作為該週第一天的週數。



圖表顯示交易 8192 發生在第 2 週。



因為應用程式使用中斷的週且一週的第一天是星期日，所以從 1 月 2 日到 1 月 8 日發生的交易會傳回值 2022/02，即 2022 年第 2 週。請注意，交易 8192 發生在 1 月 5 日並為「week_number」欄位傳回值 2022/02。

範例 5 – 情境

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2019 年最後一週和 2020 年前兩週交易的資料集，這載入到稱為「Transactions」的表格中。
- 設定為 0 的 BrokenWeeks 系統變數。
- 設定為 2 的 ReferenceDay 系統變數。
- 設定為 MM/DD/YYYY 格式的 DateFormat 系統變數。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格。

使用下列運算式建立計算維度：

```
=weekname(date)
```

若要計算總銷售額，建立下列彙總量值：

```
=sum(amount)
```

將量值的**數字格式**設定為**金錢**。

結果表格

weekname(date)	=sum(amount)
2019/52	$125.69
2020/01	$346.51
2020/02	$347.57
2020/03	$122.01

若要展示在此情境下使用 weekname() 函數的結果，新增下列欄位作為維度：

```
date
```

具有日期欄位的結果表格

weekname(date)	日期	=sum(amount)
2019/52	12/27/2019	$58.27
2019/52	12/28/2019	$67.42
2020/01	12/29/2019	$23.80
2020/01	12/30/2019	$82.06
2020/01	12/31/2019	$40.56
2020/01	01/01/2020	$37.23
2020/01	01/02/2020	$17.17
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

weekname(date)	日期	=sum(amount)
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

因為應用程式使用中斷的週，且由於 `ReferenceDay` 系統變數，第 1 週需要至少有兩天落在 1 月，因此 2020 年第 1 週包括從 2019 年 12 月 29 日起的交易。

weekstart

此功能傳回的值相當於包含 **date**的行事曆週第一天、第一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 **DateFormat**。

語法：

```
WeekStart(timestamp [, period_no [, first_week_day ]])
```

傳回的資料類型：雙值

`weekstart()` 函數判定日期落在哪一週。然後以日期格式傳回該週第一毫秒的時間戳記。該週的第一天由 `FirstWeekDay` 環境變數決定。但是，這可由 `weekstart()` 函數中的 `first_week_day` 引數取代。

引數

引數	描述
timestamp	要評估的時間戳記。
period_no	**shift** 為整數，值 0 表示是包含 **date** 的週。負值的 shift 表示之前的週，正值表示之後的週。
first_week_day	指定一週開始的日期。如果忽略，將使用變數 **FirstWeekDay** 的值。 **first_week_day** 可能的值是對星期一使用 0、對星期二使用 1，對星期三使用 2，對星期四使用 3，對星期五使用 4，對星期六使用 5，並對星期日使用 6。 如需系統變數的更多資訊，請參閱 *FirstWeekDay (page 212)*。

什麼情況下使用

`weekstart()` 函數通常在使用者想要計算使用一週中已經過的部分時，作為運算式的一部分使用。例如，若使用者想要計算員工目前為止在一週中所賺取的總工資，即可使用此函數。

下列範例假設：

```
SET FirstWeekDay=0;
```

函數範例

範例	結果
weekstart('01/12/2013')	傳回 01/07/2013。
weekstart('01/12/2013', -1)	傳回 11/31/2012。
weekstart('01/12/2013', 0, 1)	傳回 01/08/2013。

區域設定

除非另有說明,否則此主題中的範例皆使用下列日期格式:MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素,您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式,以滿足您的需求。或者,您可以在載入指令碼中變更格式,以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典,資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例:

若您想要週和週數的 ISO 設定,確認指令碼中有下列內容:

```
Set DateFormat   ='YYYY-MM-DD';
Set FirstWeekDay =0;    // Monday as first week day
Set BrokenWeeks  =0;    //(use unbroken weeks)
Set ReferenceDay =4;    // Jan 4th is always in week 1
```

若您想要 US 設定,確認指令碼中有下列內容:

```
Set DateFormat   ='M/D/YYYY';
Set FirstWeekDay =6;    // Sunday as first week day
Set BrokenWeeks  =1;    //(use broken weeks)
Set ReferenceDay =1;    // Jan 1st is always in week 1
```

以上範例從 `weekstart()` 函數產生下列內容:

Weekstart 函數範例

日期	ISO 週開始	US 週開始
Sat 2020 Dec 26	2020-12-21	12/20/2020
Sun 2020 Dec 27	2020-12-21	12/27/2020
Mon 2020 Dec 28	2020-12-28	12/27/2020
Tue 2020 Dec 29	2020-12-28	12/27/2020
Wed 2020 Dec 30	2020-12-28	12/27/2020
Thu 2020 Dec 31	2020-12-28	12/27/2020
Fri 2021 Jan 1	2020-12-28	12/27/2020

日期	ISO 週開始	US 週開始
Sat 2021 Jan 2	2020-12-28	12/27/2020
Sun 2021 Jan 3	2020-12-28	1/3/2021
Mon 2021 Jan 4	2021-01-04	1/3/2021
Tue 2021 Jan 5	2021-01-04	1/3/2021

在 ISO 欄中，一週在星期一開始，而在 US 欄中，在星期日開始。

範例 1 – 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2022 年交易的資料集，這載入到稱為 `Transactions` 的表格中。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 `start_of_week`，這傳回交易發生時每週開始的時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- start_of_week
- start_of_week_timestamp

結果表格

日期	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

日期	start_of_week	start_of_week_timestamp
9/26/2022	09/25/2022	9/25/2022 12:00:00 AM
10/14/2022	10/09/2022	10/9/2022 12:00:00 AM
10/29/2022	10/23/2022	10/23/2022 12:00:00 AM

start_of_week 欄位在前置 LOAD 陳述式中的建立方式是使用 weekstart() 函數並傳遞日期欄位，作為函數的引數。

The weekstart() 函數最初識別日期值落在哪一週，並傳回該週第一毫秒的時間戳記。

weekstart() 函數的圖表，無其他引數的範例

交易 8191 發生在 2 月 5 日。FirstWeekDay 系統變數會將每週第一天設為星期日。weekstart() 函數識別 2 月 5 日前的第一個星期六 – 因此是每週的開始 – 落在 1 月 30 日。因此，該交易的 start_of_week 值會傳回該日子的第一毫秒，亦即 1 月 30 日上午 12:00:00。

範例 2 – period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 建立欄位 previous_week_start, ，這傳回交易發生之前每季開始的時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- previous_week_start
- previous_week_start_timestamp

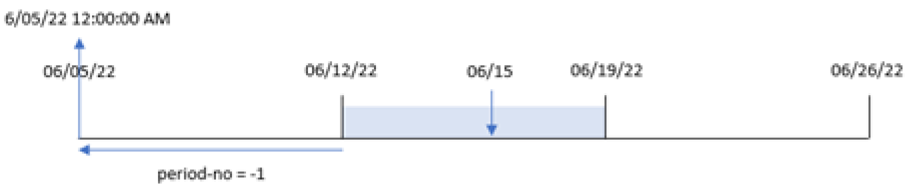
結果表格

日期	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

日期	previous_week_start	previous_week_start_timestamp
7/22/2022	07/10/2022	7/10/2022 12:00:00 AM
7/23/2022	07/10/2022	7/10/2022 12:00:00 AM
7/27/2022	07/17/2022	7/17/2022 12:00:00 AM
8/2/2022	07/24/2022	7/24/2022 12:00:00 AM
8/8/2022	07/31/2022	7/31/2022 12:00:00 AM
8/19/2022	08/07/2022	8/7/2022 12:00:00 AM
9/26/2022	09/18/2022	9/18/2022 12:00:00 AM
10/14/2022	10/02/2022	10/2/2022 12:00:00 AM
10/29/2022	10/16/2022	10/16/2022 12:00:00 AM

在此例中，因為 `period_no` 的值 -1 已作為 `weekstart()` 函數中的偏移引數使用，所以該函數首先會識別交易發生的週。然後這會查看前一週並識別該週的第一毫秒。

`weekstart()` 函數的圖表，period_no 範例



交易 8196 發生在 6 月 15 日。`weekstart()` 函數識別該週開始於 6 月 12 日。因此，前一週開始於 6 月 5 日 12:00:00 AM；這是 `previous_week_start` 欄位傳回的值。

範例 3 – first_week_day

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。但是在此範例中，我們必須將星期二設為工作週第一天。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- start_of_week
- start_of_week_timestamp

結果表格

日期	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

日期	start_of_week	start_of_week_timestamp
5/7/2022	05/03/2022	5/3/2022 12:00:00 AM
5/16/2022	05/10/2022	5/10/2022 12:00:00 AM
6/15/2022	06/14/2022	6/14/2022 12:00:00 AM
6/26/2022	06/21/2022	6/21/2022 12:00:00 AM
7/9/2022	07/05/2022	7/5/2022 12:00:00 AM
7/22/2022	07/19/2022	7/19/2022 12:00:00 AM
7/23/2022	07/19/2022	7/19/2022 12:00:00 AM
7/27/2022	07/26/2022	7/26/2022 12:00:00 AM
8/2/2022	08/02/2022	8/2/2022 12:00:00 AM
8/8/2022	08/02/2022	8/2/2022 12:00:00 AM
8/19/2022	08/16/2022	8/16/2022 12:00:00 AM
9/26/2022	09/20/2022	9/20/2022 12:00:00 AM
10/14/2022	10/11/2022	10/11/2022 12:00:00 AM
10/29/2022	10/25/2022	10/25/2022 12:00:00 AM

在此例中，因為 `first_week_date` 引數的值 1 已用於 `weekstart()` 函數中，所以這會將每週第一天設為星期二。

`weekstart()` 函數的圖表，first_week_day 範例

交易 8191 發生在 2 月 5 日。`weekstart()` 函數識別此日期前的第一個星期二 – 因此是每週的開始並傳回該值 – 落在 2 月 1 日 12:00:00 AM。

範例 4 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。

不過，在此範例中，不變的資料集會載入到應用程式中。會建立傳回交易發生週開始時間戳記的計算，作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：date。

若要計算交易發生的該週開始時間，新增下列量值：

- =weekstart(date)
- =timestamp(weekstart(date))

結果表格

日期	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

日期	start_of_week	start_of_week_timestamp
2/28/2022	02/27/2022	2/27/2022 12:00:00 AM
3/16/2022	03/13/2022	3/13/2022 12:00:00 AM
4/1/2022	03/27/2022	3/27/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/16/2022	05/15/2022	5/15/2022 12:00:00 AM
6/15/2022	06/12/2022	6/12/2022 12:00:00 AM
6/26/2022	06/26/2022	6/26/2022 12:00:00 AM
7/9/2022	07/03/2022	7/3/2022 12:00:00 AM
7/22/2022	07/17/2022	7/17/2022 12:00:00 AM
7/23/2022	07/17/2022	7/17/2022 12:00:00 AM
7/27/2022	07/24/2022	7/24/2022 12:00:00 AM
8/2/2022	07/31/2022	7/31/2022 12:00:00 AM
8/8/2022	08/07/2022	8/7/2022 12:00:00 AM
8/19/2022	08/14/2022	8/14/2022 12:00:00 AM
9/26/2022	09/25/2022	9/25/2022 12:00:00 AM
10/14/2022	10/09/2022	10/9/2022 12:00:00 AM
10/29/2022	10/23/2022	10/23/2022 12:00:00 AM

start_of_week 量值在圖表物件中的建立方式是使用 weekstart() 函數並傳遞 date 欄位，作為函數的引數。

The weekstart() 函數最初識別日期值落在哪一週，並傳回該週第一毫秒的時間戳記。

weekstart() 函數的圖表，圖表物件範例

交易 8191 發生在 2 月 5 日。FirstWeekDay 系統變數會將每週第一天設為星期日。weekstart() 函數識別 2 月 5 日前的第一個星期日 – 因此是每週開始 – 落在 1 月 30 日。因此，該交易的 start_of_week 值傳回該日子的第一毫秒，亦即 1 月 30 日上午 12:00:00。

範例 5 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 載入到稱為 Payroll 之表格的資料集。
- 由員工 ID、員工姓名及每個員工賺取的每日工資組成的資料。

員工在星期一開始工作且每週工作六天。FirstWeekDay 系統變數不得修改。

最終使用者想要圖表物件按員工 ID 和員工姓名顯示該週至今所賺取的工資。

載入指令碼

```
Payroll:
Load
*
Inline
[
employee_id,employee_name,day_rate
182,Mark, $150
183,Deryck, $125
184,Dexter, $125
185,Sydney,$270
186,Agatha,$128
];
```

結果

請執行下列動作:

1. 載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:
   - employee_id
   - employee_name
2. 接下來,建立量值以計算該週至今所賺取的工資:
   =if(today(1)-weekstart(today(1),0,0)<7,(today(1)-weekstart(today(1),0,0))*day_rate,day_rate*6)
3. 將量值的**數字格式**設定為**金錢**。

結果表格

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

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)
183	Deryck	$500.00
184	Dexter	$500.00
185	Sydney	$1080.00
186	Agatha	$512.00

藉由使用今日日期作為其第一個引數，`weekstart()` 函數會將星期一設為該週第一天並傳回目前週的開始日期。以目前日期減去該結果後，運算式會傳回本週目前已經過的天數。

然後條件就會評估本週是否超過六天。如果是的話，員工的 `day_rate` 會乘以 6 天。否則，`day_rate` 會乘以本週目前為止已經過的天數。

weekyear

此函數會根據環境變數傳回週數所屬於的年份。週數的範圍介於 1 和約 52 之間。

語法：

```
weekyear(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

傳回的資料類型： 整數

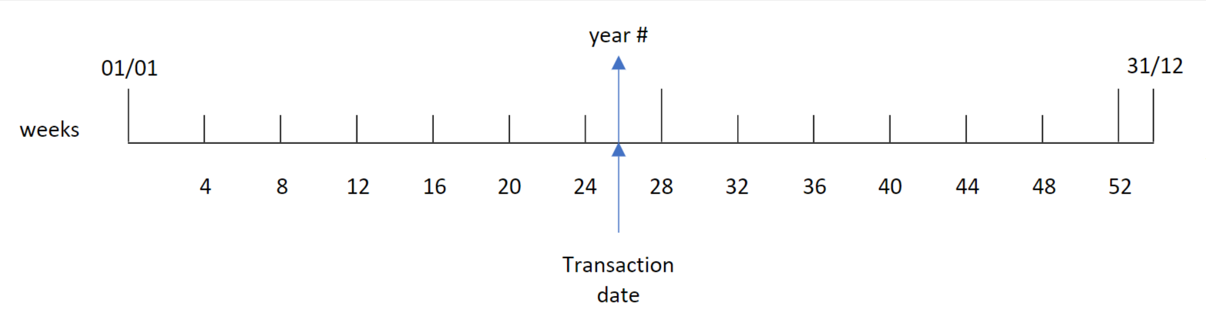
引數

引數	描述
timestamp	要評估的時間戳記。
first_week_ day	指定一週開始的日期。如果忽略，將使用變數 **FirstWeekDay** 的值。 **first_week_day** 可能的值是對星期一使用 0、對星期二使用 1，對星期三使用 2，對星期四使用 3，對星期五使用 4，對星期六使用 5，並對星期日使用 6。 如需系統變數的更多資訊，請參閱 *FirstWeekDay (page 212)*。
broken_ weeks	如果您未指定 **broken_weeks**，則變數 **BrokenWeeks** 的值將用於定義週是否中斷。
reference_ day	如果您不指定 **reference_day**，變數 **ReferenceDay** 的值將用於定義要設定一月份的哪一天為參照日以定義第 1 週。依照預設，Qlik Sense 函數使用 4 作為參照日。這表示第 1 週必須包含 1 月 4 日，或者換言之，第 1 週必須始終至少在 1 月擁有 4 日。

`weekyear()` 函數判定日期落在一年中的哪一週。然後這會傳回與該週數對應的年份。

若 `BrokenWeeks` 設定為 0 (false)，`weekyear()` 將會傳回與 `year()` 相同的內容。

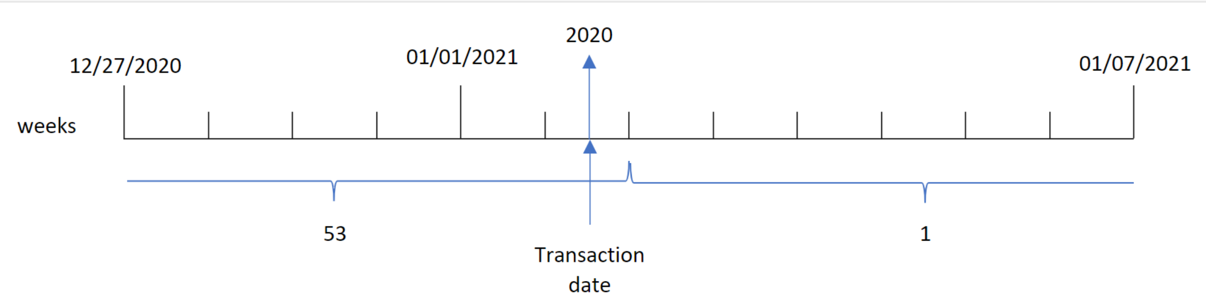
weekyear() 函數範圍的圖表



但是，若 BrokenWeeks 系統變數設為使用未中斷的週，第 1 週只能根據 ReferenceDay 系統變數中指定的值，包含 1 月中的特定天數。

例如，若使用 ReferenceDay 的值 4，第 1 週只能至少包括 1 月中的四天。第 1 週有可能包括前一年 12 月的日期，或是一年最後一週的數字有可能包括下一年 1 月的日期。在此類情況下，weekyear() 函數會傳回 year() 函數的不同值。

使用未中斷的週時，weekyear() 函數範圍的圖表



什麼情況下使用

若您想要按年份比較彙總，則 weekyear() 函數很實用。例如，若您想要按年份查看產品的總銷售額。若使用者想要在應用程式中保留與 BrokenWeeks 系統變數的一致性，即會選擇 weekyear() 函數，而不是 year()。

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
`weekyear('12/30/1996',0,0,4)`	傳回 1997，因為 1997 年第 1 週開始於 12/30/1996
`weekyear('01/02/1997',0,0,4)`	傳回 1997
`weekyear('12/28/1997',0,0,4)`	傳回 1997
`weekyear('12/30/1997',0,0,4)`	傳回 1998，因為 1998 年第 1 週開始於 12/29/1997
`weekyear('01/02/1999',0,0,4)`	傳回 1998，因為 1998 年第 53 週結束於 01/03/1999

相關主題

主題	互動
week (page 993)	傳回根據 ISO 8601 代表週數的整數
year (page 1064)	當運算式根據標準數字解譯的方式可解譯為日期時，傳回代表年份的整數。

範例 1 - 中斷的週

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2020 年最後一週和 2021 年第一週交易的資料集，這載入到稱為「`Transactions`」的表格中。
- 設為 1 的 `BrokenWeeks` 變數。
- 前置載入包含下列內容：
  - 設為欄位「`week_year`」的 `weekyear()` 函數，傳回交易發生的年份。
  - 設為欄位「`week`」的 `week()` 函數，顯示每個交易日期的週數。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date
- week
- week_year

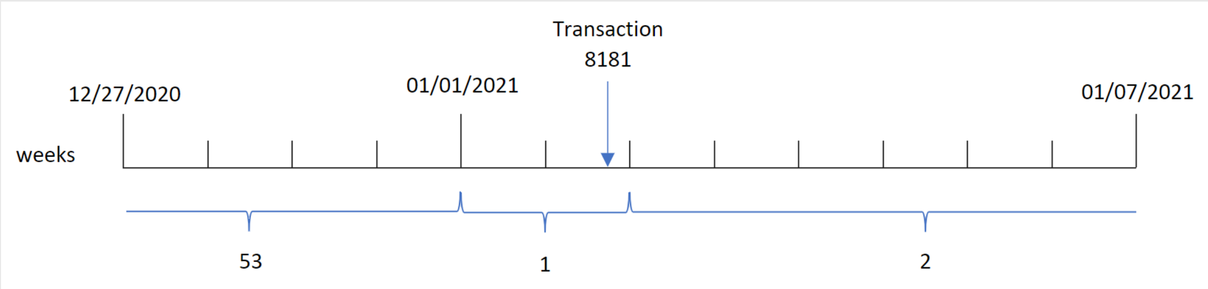
結果表格

id	日期	週	weekyear
8176	12/28/2020	53	2020
8177	12/29/2020	53	2020
8178	12/30/2020	53	2020
8179	12/31/2020	53	2020
8180	01/01/2021	1	2021
8181	01/02/2021	1	2021
8182	01/03/2021	2	2021
8183	01/04/2021	2	2021
8184	01/05/2021	2	2021
8185	01/06/2021	2	2021
8186	01/07/2021	2	2021

「week_year」欄位在前置 LOAD 陳述式中的建立方式是使用 weekyear() 函數並傳遞日期欄位，作為函數的引數。

BrokenWeeks 系統變數設為 1，表示應用程式使用中斷的週。第 1 週開始於 1 月 1 日。

使用中斷的週時，weekyear() 函數範圍的圖表



交易 8181 發生在 1 月 2 日，亦即第 1 週的一部分。因此，這會傳回「week_year」欄位在 2021 年的值。

範例 2 - 未中斷的週

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2020 年最後一週和 2021 年第一週交易的資料集，這載入到稱為「Transactions」的表格中。
- 設定為「0」的 BrokenWeeks 變數。
- 前置載入包含下列內容：
  - 設為「week_year」的 weekyear() 函數，傳回交易發生的年份。
  - 設為欄位「week」的 week() 函數，顯示每個交易日期的週數。

但是在此範例中，公司政策會使用未中斷的週。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date
- week
- week_year

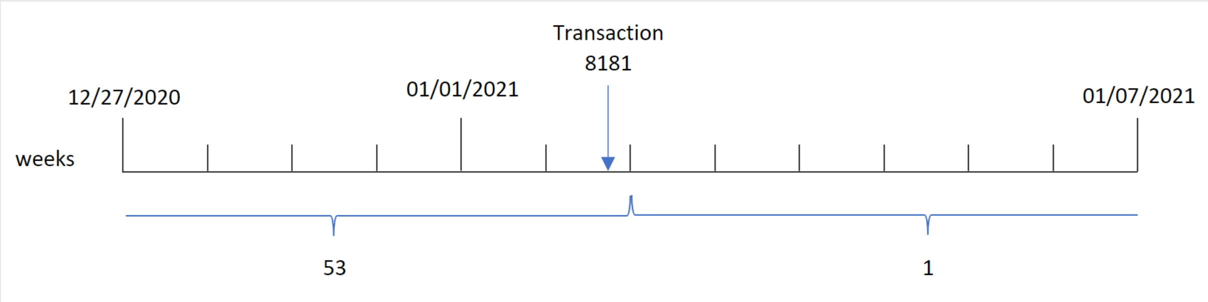
結果表格

id	日期	週	weekyear
8176	12/28/2020	53	2020
8177	12/29/2020	53	2020
8178	12/30/2020	53	2020
8179	12/31/2020	53	2020
8180	01/01/2021	53	2020
8181	01/02/2021	53	2020
8182	01/03/2021	1	2021
8183	01/04/2021	1	2021
8184	01/05/2021	1	2021
8185	01/06/2021	1	2021
8186	01/07/2021	1	2021

BrokenWeeks 系統變數設為 0，表示應用程式使用未中斷的週。因此，第 1 週不需要開始於 1 月 1 日。

2020 年第 53 週延續至 2021 年 1 月 2 日結束，而 2020 年第 1 週開始於 2021 年 1 月 3 日星期日。

使用未中斷的週時，weekyear() 函數範圍的圖表



交易 8181 發生在 1 月 2 日，亦即第 1 週的一部分。因此，這會傳回「week_year」欄位在 2021 年的值。

範例 3 – 圖表物件範例

載入指令碼和圖表運算式

概覽

使用與第一個範例相同的資料集和情境。

不過，在此範例中，資料集保持不變並且會載入到應用程式中。會建立傳回交易發生年份週數的計算，作為應用程式圖表中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date

若要計算交易發生的週，建立下列量值：

- =week(date)

若要根據週數計算交易發生的年份，建立下列量值：

- =weekyear(date)

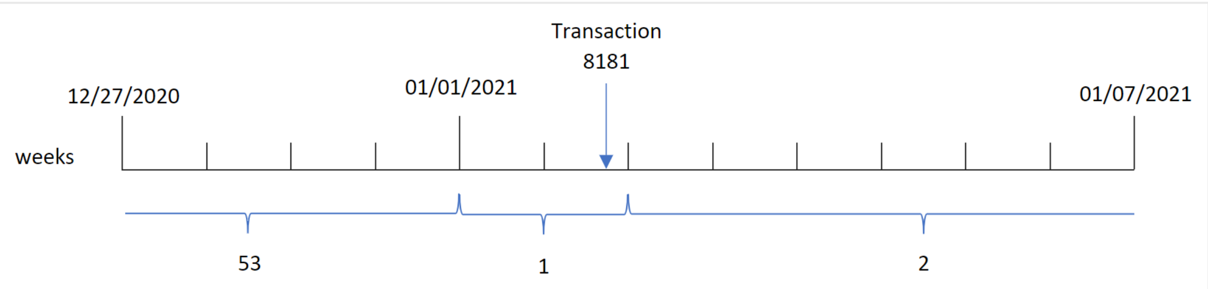
結果表格

id	日期	週	weekyear
8176	12/28/2020	53	2020
8177	12/29/2020	53	2020
8178	12/30/2020	53	2020
8179	12/31/2020	53	2020
8180	01/01/2021	1	2021
8181	01/02/2021	1	2021
8182	01/03/2021	2	2021
8183	01/04/2021	2	2021
8184	01/05/2021	2	2021
8185	01/06/2021	2	2021
8186	01/07/2021	2	2021

「week_year」欄位在前置 LOAD 陳述式中的建立方式是使用 weekyear() 函數並傳遞日期欄位，作為函數的引數。

BrokenWeeks 系統變數設為 1，表示應用程式使用中斷的週，且第 1 週開始於 1 月 1 日。

使用中斷的週時，weekyear() 函數範圍的圖表



交易 8181 發生在 1 月 2 日，亦即第 1 週的一部分。因此，這會傳回「week_year」欄位在 2021 年的值。

範例 4 – 情境

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2020 年最後一週和 2021 年第一週交易的資料集,這載入到稱為「Transactions」的表格中。
- 設定為「0」的 BrokenWeeks 變數。這表示應用程式會使用未中斷的週。
- 設定為「2」的 ReferenceDay 變數。這表示該年開始於 1 月 2 日,並至少包含 1 月中的兩天。
- 設定為「1」的 FirstWeekDay 變數。這表示每週第一天是星期二。

公司政策會使用中斷的週。最終使用者希望圖表按年份呈現總銷售額。應用程式使用未中斷的週,且第 1 週至少包含 1 月中的兩天。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格。

若要根據週數計算交易發生的年份,建立下列量值:

- =weekyear(date)

若要計算總銷售額,建立下列量值:

- sum(amount)

將量值的**數字格式**設定為**金錢**。

結果表格

weekyear(date)	=sum(amount)
2020	19.42
2021	373.37

year

當 **expression** 根據標準數字解譯的方式可解譯為日期時,此函數會傳回代表年份的整數。

語法:

```
year(expression)
```

傳回的資料類型: 整數

year() 函數可作為指令碼和圖表函數。該函數傳回特定日期的年份。這通常用來建立年份欄位,作為主要行事曆中的維度。

什麼情況下使用

若您想要按年份比較彙總,則 year() 函數很實用。例如,若您想要按年份查看產品的總銷售額,可使用此函數。

可以使用該函數建立主要行事曆表格中的欄位,以在載入指令碼中建立這些維度。相反地,這可直接在圖表中作為計算維度使用。

函數範例

範例	結果
year('2012-10-12')	傳回 2012
year('35648')	傳回 1997, 因為 35648 = 1997-08-06

區域設定

除非另有說明,否則此主題中的範例皆使用下列日期格式:MM/DD/YYYY。日期格式是在資料載入指令碼的 SET DateFormat 陳述式中指定。由於地區設定和其他因素,您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式,以滿足您的需求。或者,您可以在載入指令碼中變更格式,以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典,資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – DateFormat 資料集 (指令碼)

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 載入到名為 `Master Calendar` 之表格中的日期資料集。
- 使用預設 `DateFormat` 系統變數 MM/DD/YYYY。
- 使用 `year()` 函數建立附加欄位 `year` 的前置載入。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- `date`
- `year`

結果表格

日期	年
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

範例 2 – ANSI 日期

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 載入到名為 Master Calendar 之表格中的日期資料集。
- 使用預設 DateFormat 系統變數 MM/DD/YYYY。不過,資料集中包括的日期屬於 ANSI 標準日期格式。
- 使用 year() 函數建立附加欄位 year 的前置載入。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- `date`
- `year`

結果表格

日期	年
2020-12-28	2020
2020-12-29	2020
2020-12-30	2020
2020-12-31	2020
2021-01-01	2021
2021-01-02	2021
2021-01-03	2021
2021-01-04	2021
2021-01-05	2021
2021-01-06	2021
2021-01-07	2021

範例 3 – 未格式化的日期

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 載入到名為 Master Calendar 之表格中數字格式的日期資料集。
- 使用預設 DateFormat 系統變數 MM/DD/YYYY。
- 使用 year() 函數建立附加欄位 year 的前置載入。

名為 unformatted_date 之原始未格式化的日期已載入，且為了更加清晰，已將名為 long_date 的附加欄位用於使用 date() 函數將數字日期轉換為格式化日期欄位。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- unformatted_date
- long_date
- year

結果表格

unformatted_date	long_date	年
44868	11/03/2022	2022
44898	12/03/2022	2022
44928	01/02/2023	2023

unformatted_date	long_date	年
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

範例 4 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

在此範例中，順序資料集會載入到名為銷售的表格中。表格含有三個欄位：

- id
- sales_date
- amount

產品銷售保固自銷售日起持續兩年。任務是為了在圖表中建立量值，以便判定每次保固會到期的年份。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：`sales_date`。

建立下列量值：

=year(sales_date+365*2)

結果表格

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

上表可看到此量值的結果。若要在日期中新增兩年，將 365 乘以 2 並將結果新增至銷售日期。因此，發生在 2020 年之銷售的到期年份是 2022 年。

yearend

此函數傳回的值相當於包含 **date** 的年份最後一天、最後一毫秒的時間戳記。預設的輸出格式為指令碼中所設定的 **DateFormat**。

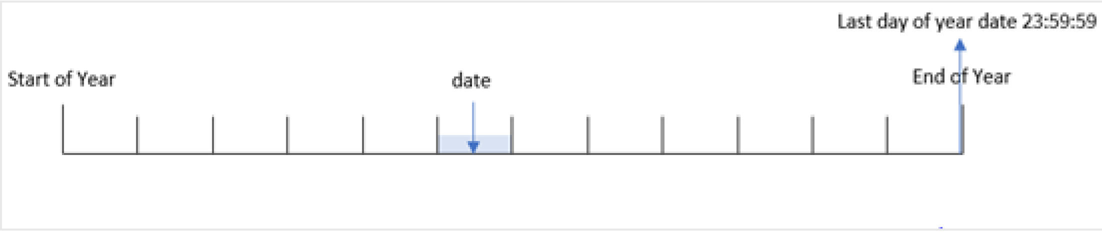
語法：

```
YearEnd( date[, period_no[, first_month_of_year = 1]])
```

換言之，`yearend()` 函數判定日期落在哪一年。然後這會以日期格式傳回該年最後一毫秒的時間戳記。依照預設，該年的第一個月是 1 月。不過，您可以使用 `yearend()` 函數中的 `first_month_of_year` 引數變更要將哪個月設為第一個月。

`yearend()` 函數不考慮 `FirstMonthOfYear` 系統變數。該年從 1 月 1 日開始，除非使用 `first_month_of_year` 引數變更。

yearend() 函數的圖表。



什麼情況下使用

yearend() 函數在您想要計算以使用一年中尚未發生的部分時，作為運算式的一部分使用。例如，若您想要計算該年期間尚未發生的總利息。

傳回的資料類型： 雙值

引數

引數	描述
date	要評估的時間戳記。
period_no	**period_no** 為整數，其中值 0 表示是包含 **date** 的年份。負值的 **period_no** 表示之前的年度，正值表示之後的年度。
first_month_of_year	如果要使用不起始於 1 月的 (會計) 年度，可在 **first_month_of_year** 中指定介於 2 和 12 之間的值。

您可以在 `first_month_of_year` 引數中使用下列值設定一年的第一個月：

first_month_of_year 值

月	值
二月	2
三月	3
四月	4
五月	5
六月	6
七月	7
八月	8
九月	9
十月	10
十一月	11
十二月	12

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
`yearend('10/19/2001')`	傳回 12/31/2001 23:59:59。
`yearend('10/19/2001', -1)`	傳回 12/31/2000 23:59:59。
`yearend('10/19/2001', 0, 4)`	傳回 03/31/2002 23:59:59。

範例 1 – 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2020 年和 2022 年之間交易的資料集，這載入到稱為「`Transactions`」的表格中。
- 已使用 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供日期欄位。
- 前置 LOAD 陳述式包含下列內容：
  - 設定為 year_end 欄位的 `yearend()` 函數。
  - 設定為 year_end_timestamp 欄位的 `Timestamp()` 函數。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date
- year_end
- year_end_timestamp

結果表格

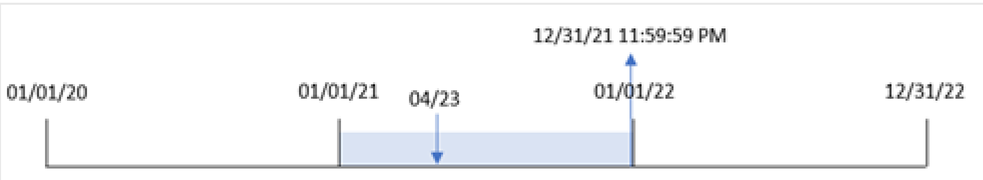
id	日期	year_end	year_end_timestamp
8188	01/13/2020	12/31/2020	12/31/2020 11:59:59 PM
8189	02/26/2020	12/31/2020	12/31/2020 11:59:59 PM
8190	03/27/2020	12/31/2020	12/31/2020 11:59:59 PM
8191	04/16/2020	12/31/2020	12/31/2020 11:59:59 PM
8192	05/21/2020	12/31/2020	12/31/2020 11:59:59 PM
8193	08/14/2020	12/31/2020	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

id	日期	year_end	year_end_timestamp
8196	01/22/2021	12/31/2021	12/31/2021 11:59:59 PM
8197	02/03/2021	12/31/2021	12/31/2021 11:59:59 PM
8198	03/17/2021	12/31/2021	12/31/2021 11:59:59 PM
8199	04/23/2021	12/31/2021	12/31/2021 11:59:59 PM
8200	05/04/2021	12/31/2021	12/31/2021 11:59:59 PM
8201	06/30/2021	12/31/2021	12/31/2021 11:59:59 PM
8202	07/26/2021	12/31/2021	12/31/2021 11:59:59 PM
8203	12/27/2021	12/31/2021	12/31/2021 11:59:59 PM
8204	06/06/2022	12/31/2022	12/31/2022 11:59:59 PM
8205	07/18/2022	12/31/2022	12/31/2022 11:59:59 PM
8206	11/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	12/12/2022	12/31/2022	12/31/2022 11:59:59 PM

「year_end」欄位在前置 LOAD 陳述式中的建立方式是使用 yearend() 函數並傳遞日期欄位，作為函數的引數。

yearend() 函數最初識別日期值落在哪一年，並傳回該年最後一毫秒的時間戳記。

選取交易 8199 的 yearend() 函數圖表。



交易 8199 發生在 2021 年 4 月 23 日。yearend() 函數傳回該年的最後一毫秒，亦即 12 月 31 日下午 11:59:59。

範例 2 – period_no

載入指令碼和結果

概覽

使用與第一個範例相同的資料集和情境。

不過，在此範例中，任務是要建立欄位「previous_year_end」，這傳回交易發生年份之前的年份結束日期時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date
- previous_year_end
- previous_ year_end_timestamp

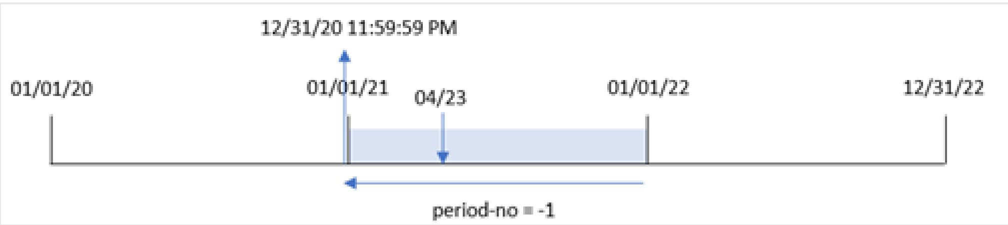
結果表格

id	日期	previous_year_end	previous_year_end_timestamp
8188	01/13/2020	12/31/2019	12/31/2019 11:59:59 PM

id	日期	previous_year_end	previous_year_end_timestamp
8189	02/26/2020	12/31/2019	12/31/2019 11:59:59 PM
8190	03/27/2020	12/31/2019	12/31/2019 11:59:59 PM
8191	04/16/2020	12/31/2019	12/31/2019 11:59:59 PM
8192	05/21/2020	12/31/2019	12/31/2019 11:59:59 PM
8193	08/14/2020	12/31/2019	12/31/2019 11:59:59 PM
8194	10/07/2020	12/31/2019	12/31/2019 11:59:59 PM
8195	12/05/2020	12/31/2019	12/31/2019 11:59:59 PM
8196	01/22/2021	12/31/2020	12/31/2020 11:59:59 PM
8197	02/03/2021	12/31/2020	12/31/2020 11:59:59 PM
8198	03/17/2021	12/31/2020	12/31/2020 11:59:59 PM
8199	04/23/2021	12/31/2020	12/31/2020 11:59:59 PM
8200	05/04/2021	12/31/2020	12/31/2020 11:59:59 PM
8201	06/30/2021	12/31/2020	12/31/2020 11:59:59 PM
8202	07/26/2021	12/31/2020	12/31/2020 11:59:59 PM
8203	12/27/2021	12/31/2020	12/31/2020 11:59:59 PM
8204	06/06/2022	12/31/2021	12/31/2021 11:59:59 PM
8205	07/18/2022	12/31/2021	12/31/2021 11:59:59 PM
8206	11/14/2022	12/31/2021	12/31/2021 11:59:59 PM
8207	12/12/2022	12/31/2021	12/31/2021 11:59:59 PM

因為 period_no 的值 -1 已作為 yearend() 函數中的偏移引數使用，所以該函數首先會識別交易發生的年份。然後這會查看前一年並識別該年的最後一毫秒。

具有 period_no 值 -1 的 yearend() 函數圖表。



交易 8199 發生在 2021 年 4 月 23 日。yearend() 函數為「previous_year_end」欄位傳回前一年的最後一毫秒，亦即 2020 年 12 月 31 日下午 11:59:59。

範例 3 – first_month_of_year

載入指令碼和結果

概覽

使用與第一個範例相同的資料集和情境。

不過，在此範例中，公司政策適用於從 4 月 1 日開始的年份。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

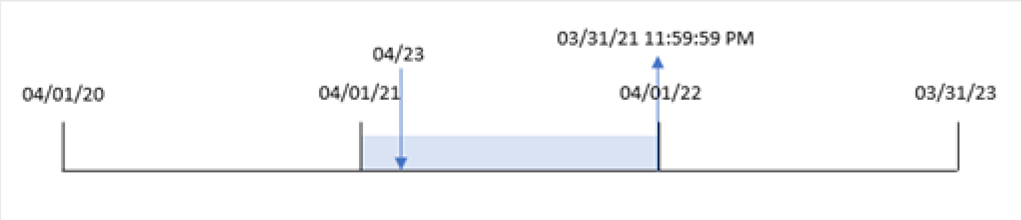
- id
- date
- year_end
- year_end_timestamp

結果表格

id	日期	year_end	year_end_timestamp
8188	01/13/2020	03/31/2020	3/31/2020 11:59:59 PM
8189	02/26/2020	03/31/2020	3/31/2020 11:59:59 PM
8190	03/27/2020	03/31/2020	3/31/2020 11:59:59 PM
8191	04/16/2020	03/31/2021	3/31/2021 11:59:59 PM
8192	05/21/2020	03/31/2021	3/31/2021 11:59:59 PM
8193	08/14/2020	03/31/2021	3/31/2021 11:59:59 PM
8194	10/07/2020	03/31/2021	3/31/2021 11:59:59 PM
8195	12/05/2020	03/31/2021	3/31/2021 11:59:59 PM
8196	01/22/2021	03/31/2021	3/31/2021 11:59:59 PM
8197	02/03/2021	03/31/2021	3/31/2021 11:59:59 PM
8198	03/17/2021	03/31/2021	3/31/2021 11:59:59 PM
8199	04/23/2021	03/31/2022	3/31/2022 11:59:59 PM
8200	05/04/2021	03/31/2022	3/31/2022 11:59:59 PM
8201	06/30/2021	03/31/2022	3/31/2022 11:59:59 PM
8202	07/26/2021	03/31/2022	3/31/2022 11:59:59 PM
8203	12/27/2021	03/31/2022	3/31/2022 11:59:59 PM
8204	06/06/2022	03/31/2023	3/31/2023 11:59:59 PM
8205	07/18/2022	03/31/2023	3/31/2023 11:59:59 PM
8206	11/14/2022	03/31/2023	3/31/2023 11:59:59 PM
8207	12/12/2022	03/31/2023	3/31/2023 11:59:59 PM

因為 4 的 first_month_of_year 引數用於 yearend() 函數，所以這將該年的第一天設定為 4 月 1 日，而該年的最後一天則為 3 月 31 日。

以 4 月作為該年第一個月的 yearend() 函數圖表。



交易 8199 發生在 2021 年 4 月 23 日。因為 yearend() 函數將該年的開始設定為 4 月 1 日，這會傳回 2022 年 3 月 31 日，作為交易的「year_end」值。

範例 4 – 圖表物件範例

載入指令碼和圖表運算式

概覽

使用與第一個範例相同的資料集和情境。

不過，在此範例中，資料集保持不變並且會載入到應用程式中。會建立傳回交易發生年份結束日期時間戳記的計算，作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date

若要計算交易發生在哪一年，建立下列量值：

- =yearend(date)
- =timestamp(yearend(date))

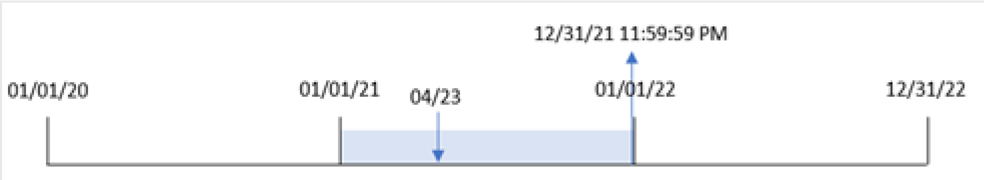
結果表格

id	日期	=yearend(date)	=timestamp(yearend(date))
8188	01/13/2020	12/31/2020	12/31/2020 11:59:59 PM
8189	02/26/2020	12/31/2020	12/31/2020 11:59:59 PM
8190	03/27/2020	12/31/2020	12/31/2020 11:59:59 PM
8191	04/16/2020	12/31/2020	12/31/2020 11:59:59 PM
8192	05/21/2020	12/31/2020	12/31/2020 11:59:59 PM
8193	08/14/2020	12/31/2020	12/31/2020 11:59:59 PM
8194	10/07/2020	12/31/2020	12/31/2020 11:59:59 PM
8195	12/05/2020	12/31/2020	12/31/2020 11:59:59 PM
8196	01/22/2021	12/31/2021	12/31/2021 11:59:59 PM
8197	02/03/2021	12/31/2021	12/31/2021 11:59:59 PM
8198	03/17/2021	12/31/2021	12/31/2021 11:59:59 PM
8199	04/23/2021	12/31/2021	12/31/2021 11:59:59 PM
8200	05/04/2021	12/31/2021	12/31/2021 11:59:59 PM
8201	06/30/2021	12/31/2021	12/31/2021 11:59:59 PM
8202	07/26/2021	12/31/2021	12/31/2021 11:59:59 PM
8203	12/27/2021	12/31/2021	12/31/2021 11:59:59 PM
8204	06/06/2022	12/31/2022	12/31/2022 11:59:59 PM
8205	07/18/2022	12/31/2022	12/31/2022 11:59:59 PM
8206	11/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	12/12/2022	12/31/2022	12/31/2022 11:59:59 PM

「end_of_year」量值在圖表物件中的建立方式是使用 yearend() 函數並傳遞日期欄位，作為函數的引數。

yearend() 函數最初識別日期值落在哪一年，傳回該年最後一毫秒的時間戳記。

yearend() 函數圖表顯示交易 8199 發生在 4 月。



交易 8199 發生在 2021 年 4 月 23 日。yearend() 函數傳回該年的最後一毫秒，亦即 12 月 31 日下午 11:59:59。

範例 5 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 資料集載入到稱為「Employee_Expenses」的表格。該表格含有下列欄位：
  - 員工 ID
  - 員工名稱
  - 每個員工報銷的平均每日開支

最終使用者希望圖表物件按員工 ID 和員工名稱顯示該年剩餘期間仍會產生的預估開支報銷。會計年度於 1 月開始。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- employee_id
- employee_name

若要計算預測的開支報銷，建立下列量值：

```
=(yearend(today(1))-today(1))*avg_daily_claim
```

將量值的**數字格式**設定為**金錢**。

結果表格

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

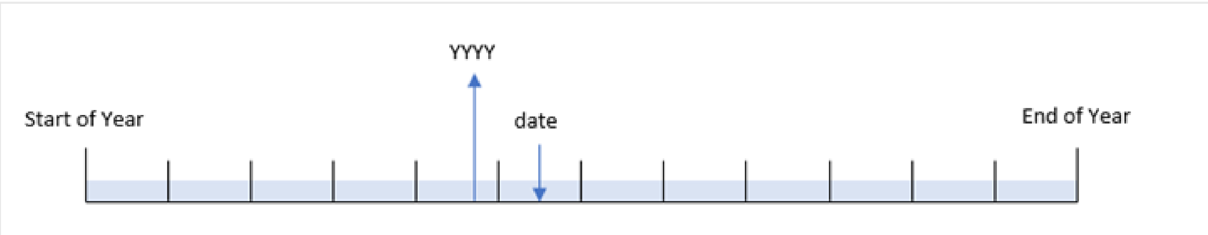
藉由使用今日日期作為其唯一引數，yearend() 函數會傳回目前年份的結束日期。然後，以該年結束日期減去今日日期，運算式就會傳回該年剩餘天數。

然後按員工讓此值乘以平均每日開支報銷，以計算每個員工在該年剩餘期間預期會有的預估報銷值。

yearname

此函數傳回四位數的年度為顯示值，其基礎數值相當於包含 **date** 的年度的第一天、第一毫秒的時間戳記。

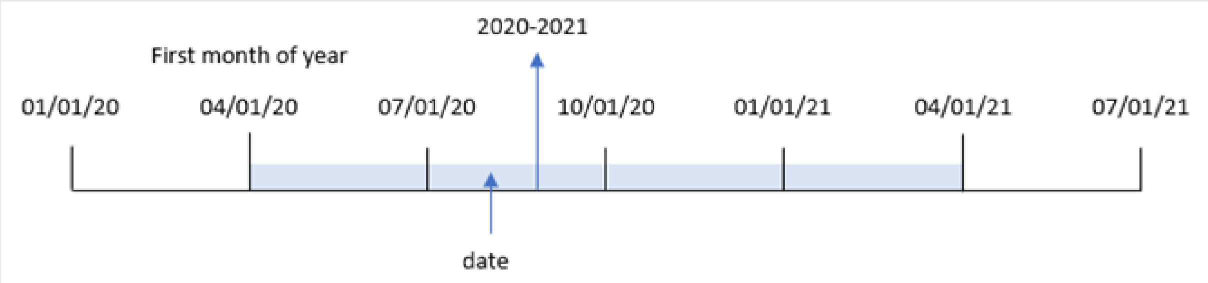
yearname() 函數的時間範圍圖表。



yearname() 函數不同於 year() 函數，因為這可讓您移動您要評估的日期，並設定該年的第一個月。

若該年的第一個月不是 1 月，該函數將會傳回跨越 12 個月期間並包含該日期的兩個四位數年份。例如，若該年於 4 月開始，而評估的日期為 2020 年 6 月 30 日，則傳回的結果會是 2020-2021。

將 4 月設定為該年第一個月的 yearname() 函數圖表。



語法:

```
YearName(date[, period_no[, first_month_of_year]] )
```

傳回的資料類型: 雙值

引數	描述
date	要評估的時間戳記。
period_no	**period_no** 為整數,其中值 0 表示是包含 **date** 的年份。負值的 **period_no** 表示之前的年度,正值表示之後的年度。
first_month_of_year	如果要使用不起始於 1 月的 (會計) 年度,可在 **first_month_of_year** 中指定介於 2 和 12 之間的值。顯示值將會是顯示兩個年度的字串。

您可以在 first_month_of_year 引數中使用下列值設定一年的第一個月:

first_month_of_year 值

月	值
二月	2
三月	3
四月	4
五月	5
六月	6
七月	7
八月	8
九月	9
十月	10
十一月	11
十二月	12

什麼情況下使用

若要按年份比較彙總，`yearname()` 函數很實用。例如，若您想要按年份查看產品的總銷售額。

可以使用該函數建立主要行事曆表格中的欄位，以在載入指令碼中建立這些維度。這也能在圖表中建立以作為計算維度

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	**結果**
`yearname('10/19/2001')`	傳回 '2001'。
`yearname('10/19/2001',-1)`	傳回 '2000'。
`yearname('10/19/2001',0,4)`	傳回 '2001-2002'。

相關主題

主題	**描述**
year (page 1064)	當運算式根據標準數字解譯的方式可解譯為日期時，此函數會傳回代表年份的整數。

範例 1 – 無其他引數

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2020 年和 2022 年之間交易的資料集，這載入到稱為「`Transactions`」的表格中。
- 設定為「`MM/DD/YYYY`」的 `DateFormat` 系統變數。
- 使用 `yearname()` 並設定為 `year_name` 欄位的前置載入。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- year_name

結果表格

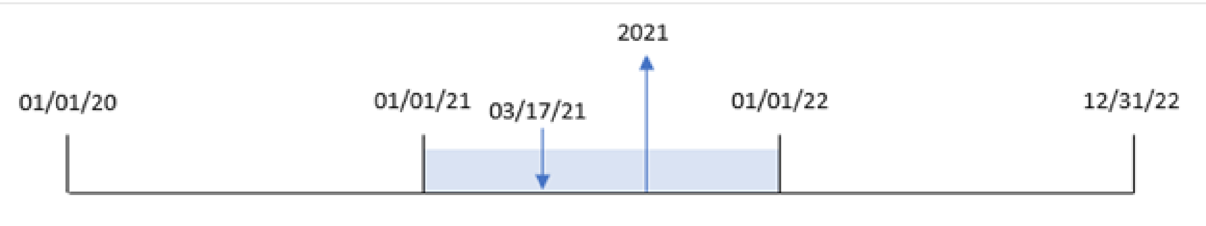
日期	year_name
01/13/2020	2020
02/26/2020	2020
03/27/2020	2020
04/16/2020	2020

日期	year_name
05/21/2020	2020
08/14/2020	2020
10/07/2020	2020
12/05/2020	2020
01/22/2021	2021
02/03/2021	2021
03/17/2021	2021
04/23/2021	2021
05/04/2021	2021
06/30/2021	2021
07/26/2021	2021
12/27/2021	2021
06/06/2022	2022
07/18/2022	2022
11/14/2022	2022
12/12/2022	2022

「year_name」欄位在前置 LOAD 陳述式中的建立方式是使用 yearname() 函數並傳遞日期欄位，作為函數的引數。

yearname() 函數識別日期值落在哪一年並傳回此作為四位數年份值。

yearname() 函數圖表顯示 2021 年為年份值。



範例 2 – period_no

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2020 年和 2022 年之間交易的資料集，這載入到稱為「Transactions」的表格中。
- 設定為「MM/DD/YYYY」的 DateFormat 系統變數。
- 使用 yearname() 並設定為 year_name 欄位的前置載入。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- prior_year_name

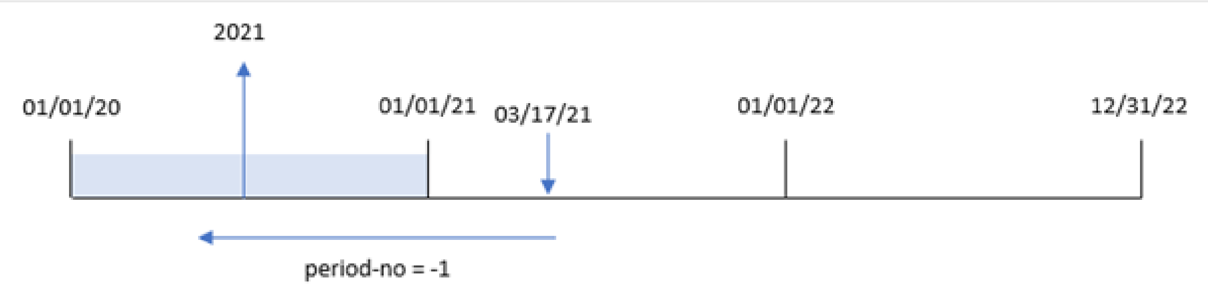
結果表格

日期	prior_year_name
01/13/2020	2019

日期	prior_year_name
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

因為 `period_no` 的值 `-1` 作為 `yearname()` 函數中的偏移引數使用，所以該函數首先會識別交易發生的年份。然後該函數會往前移動一年並傳回產生的年份。

具有設定為 `-1` 之 `period_no` 的 `yearname()` 函數圖表。



範例 3 – first_month_of_year

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 資料集與第一個範例中相同。
- 設定為「MM/DD/YYYY」的 DateFormat 系統變數。
- 使用 yearname() 並設定為 year_name 欄位的前置載入。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- `id`
- `date`
- `year_name`

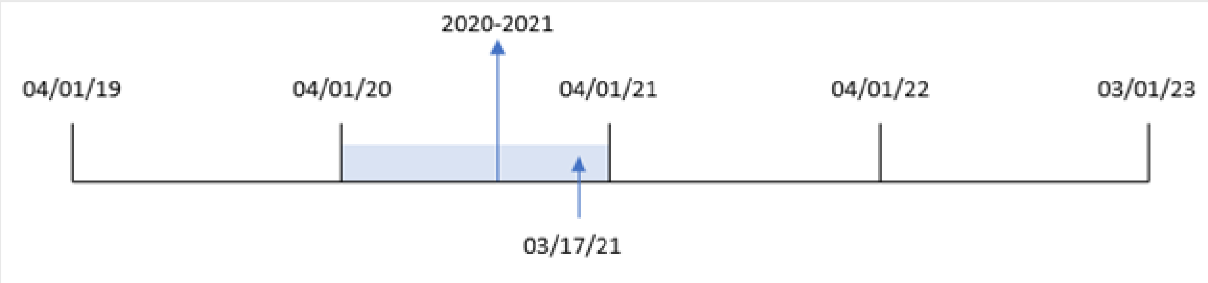
結果表格

日期	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

因為 4 的 `first_month_of_year` 引數用於 `yearname()` 函數，所以該年的開始時間從 1 月 1 日移動至 4 月 1 日。因此，每 12 個月期間會跨越兩個曆年，而 `yearname()` 函數會為評估的日期傳回兩個四位數年份。

交易 8198 發生在 2021 年 3 月 17 日。yearname() 函數將該年設為開始於 4 月 1 日，結束於 3 月 30 日。因此，交易 8198 發生在從 2020 年 4 月 1 日到 2021 年 3 月 30 日的年份期間。因此，yearname() 函數會傳回值 2020-2021。

將 3 月設定為該年第一個月的 yearname() 函數圖表。



範例 4 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 資料集與第一個範例中相同。
- 設定為「MM/DD/YYYY」的 DateFormat 系統變數。

不過，傳回交易發生年份的欄位在圖表物件中作為量值來建立。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度:

`date`

若要計算「`year_name`」欄位，建立此量值:

`=yearname(date)`

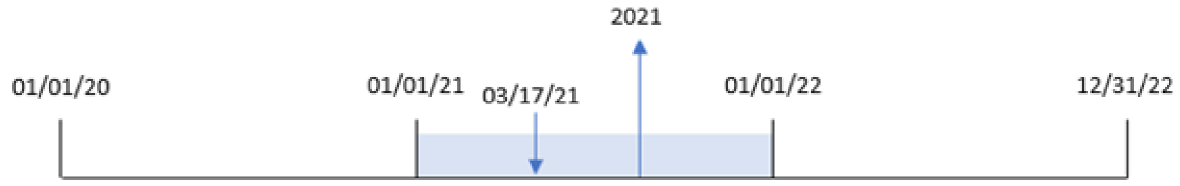
結果表格

日期	=yearname(date)
01/13/2020	2020
02/26/2020	2020
03/27/2020	2020
04/16/2020	2020
05/21/2020	2020
08/14/2020	2020
10/07/2020	2020
12/05/2020	2020
01/22/2021	2021
02/03/2021	2021
03/17/2021	2021
04/23/2021	2021
05/04/2021	2021
06/30/2021	2021
07/26/2021	2021
12/27/2021	2021
06/06/2022	2022
07/18/2022	2022
11/14/2022	2022
12/12/2022	2022

「year_name」量值在圖表物件中的建立方式是使用 yearname() 函數並傳遞日期欄位，作為函數的引數。

yearname() 函數識別日期值落在哪一年並傳回此作為四位數年份值。

yearname() 函數圖表以 2021 年為年份值。



範例 5 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 資料集與第一個範例中相同。
- 設定為「MM/DD/YYYY」的 DateFormat 系統變數。

最終使用者希望圖表按季呈現交易的總銷售額。使用 yearname() 函數作為計算維度，以在 yearname() 維度無法用於資料模型時建立此圖表。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格。

若要按年份比較彙總，建立此計算維度：

=yearname(date)

建立此量值：

=sum(amount)

將量值的**數字格式**設定為**金錢**。

結果表格

yearname(date)	=sum(amount)
2020	$463.55
2021	$457.69
2022	$294.35

yearstart

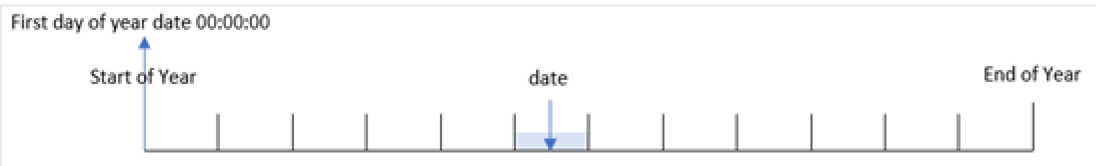
此函數傳回的時間戳記相當於包含 **date** 的年份的第一天開始。預設的輸出格式為指令碼中所設定的 **DateFormat**。

語法：

YearStart(date[, period_no[, first_month_of_year]]**)**

換言之，yearstart() 函數判定日期落在哪一年。然後這會以日期格式傳回該年第一毫秒的時間戳記。依照預設，該年的第一個月是 1 月；不過，您可以使用 yearstart() 函數中的 first_month_of_year 引數變更要將哪個月設為第一個月。

yearstart() 函數圖表顯示該函數可涵蓋的時間範圍。



什麼情況下使用

`yearstart()` 函數在您想要計算以使用一年中目前已經過的部分時，作為運算式的一部分使用。例如，若您想要計算年初至今累積的利息。

傳回的資料類型： 雙值

引數

引數	描述
date	要評估的時間戳記。
period_no	**period_no** 為整數，其中值 0 表示是包含 **date** 的年份。負值的 **period_no** 表示之前的年度，正值表示之後的年度。
first_month_of_year	如果要使用不起始於 1 月的 (會計) 年度，可在 **first_month_of_year** 中指定介於 2 和 12 之間的值。

下列月份可用於 `first_month_of_year argument`：

first_month_of_year 值

月	值
二月	2
三月	3
四月	4
五月	5
六月	6
七月	7
八月	8
九月	9
十月	10
十一月	11
十二月	12

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典,資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

函數範例

範例	結果
`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.

範例 1 – 基本範例

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 包含一組 2020 年和 2022 年之間交易的資料集,這載入到稱為「`Transactions`」的表格中。
- 已使用 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供日期欄位。
- 前置 LOAD 陳述式包含下列內容:
  - 設定為 `year_start` 欄位的 `yearstart()` 函數。
  - 設定為 `year_start_timestamp` 欄位的 `Timestamp()` 函數

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date
- year_start
- year_start_timestamp

結果表格

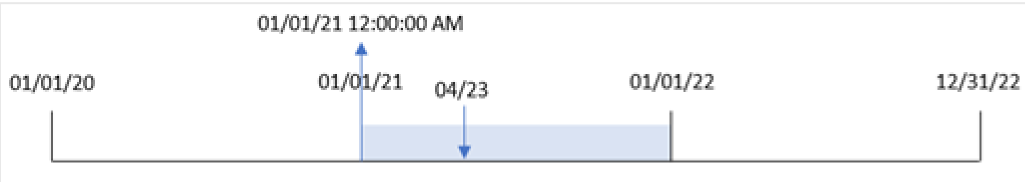
id	日期	year_start	year_start_timestamp
8188	01/13/2020	01/01/2020	1/1/2020 12:00:00 AM
8189	02/26/2020	01/01/2020	1/1/2020 12:00:00 AM
8190	03/27/2020	01/01/2020	1/1/2020 12:00:00 AM
8191	04/16/2020	01/01/2020	1/1/2020 12:00:00 AM
8192	05/21/2020	01/01/2020	1/1/2020 12:00:00 AM
8193	08/14/2020	01/01/2020	1/1/2020 12:00:00 AM
8194	10/07/2020	01/01/2020	1/1/2020 12:00:00 AM
8195	12/05/2020	01/01/2020	1/1/2020 12:00:00 AM
8196	01/22/2021	01/01/2021	1/1/2021 12:00:00 AM
8197	02/03/2021	01/01/2021	1/1/2021 12:00:00 AM
8198	03/17/2021	01/01/2021	1/1/2021 12:00:00 AM
8199	04/23/2021	01/01/2021	1/1/2021 12:00:00 AM
8200	05/04/2021	01/01/2021	1/1/2021 12:00:00 AM

id	日期	year_start	year_start_timestamp
8201	06/30/2021	01/01/2021	1/1/2021 12:00:00 AM
8202	07/26/2021	01/01/2021	1/1/2021 12:00:00 AM
8203	12/27/2021	01/01/2021	1/1/2021 12:00:00 AM
8204	06/06/2022	01/01/2022	1/1/2022 12:00:00 AM
8205	07/18/2022	01/01/2022	1/1/2022 12:00:00 AM
8206	11/14/2022	01/01/2022	1/1/2022 12:00:00 AM
8207	12/12/2022	01/01/2022	1/1/2022 12:00:00 AM

「year_start」欄位在前置 LOAD 陳述式中的建立方式是使用 yearstart() 函數並傳遞日期欄位，作為函數的引數。

yearstart() 函數最初識別日期值落在哪一年，並傳回該年第一毫秒的時間戳記。

yearstart() 函數圖表和交易 8199。



交易 8199 發生在 2021 年 4 月 23 日。yearstart() 函數傳回該年的第一毫秒，亦即 1 月 1 日上午 12:00:00。

範例 2 – period_no

載入指令碼和結果

概覽

使用與第一個範例相同的資料集和情境。

不過，在此範例中，任務是要建立欄位「previous_year_start」，這傳回交易發生年份之前的年份開始日期時間戳記。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date
- previous_year_start
- previous_ year_start_timestamp

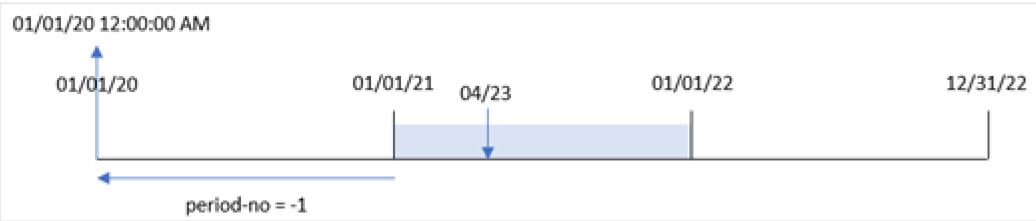
結果表格

id	日期	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

id	日期	previous_year_start	previous_year_start_timestamp
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

在此例中，因為 period_no 的值 -1 作為 yearstart() 函數中的偏移引數使用，所以該函數首先會識別交易發生的年份。然後這會查看前一年並識別該年的第一毫秒。

具有 period_no 值 -1 的 yearstart() 函數圖表。



交易 8199 發生在 2021 年 4 月 23 日。yearstart() 函數為「previous_year_start」欄位傳回前一年的第一毫秒，亦即 2020 年 1 月 1 日上午 12:00:00。

範例 3 – first_month_of_year

載入指令碼和結果

概覽

使用與第一個範例相同的資料集和情境。

不過，在此範例中，公司政策適用於從 4 月 1 日開始的年份。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- id
- date
- year_start
- year_start_timestamp

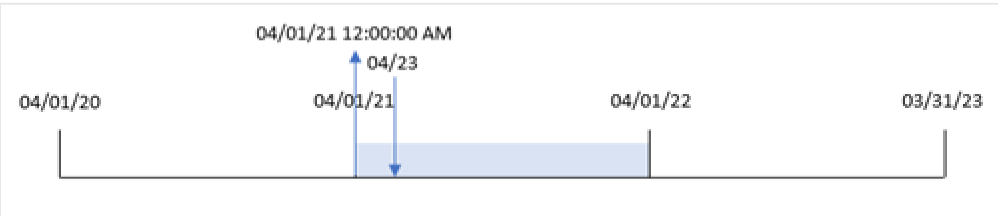
結果表格

id	日期	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

id	日期	year_start	year_start_timestamp
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

在此例中，因為 4 的 `first_month_of_year` 引數用於 `yearstart()` 函數，所以這將該年的第一天設定為 4 月 1 日，而該年的最後一天則為 3 月 31 日。

將第一個月設為 4 月的 `yearstart()` 函數圖表。



交易 8199 發生在 2021 年 4 月 23 日。因為 `yearstart()` 函數將該年的開始設定為 4 月 1 日，並傳回該日期作為交易的「`year_start`」值。

範例 4 – 圖表物件範例

載入指令碼和圖表運算式

概覽

使用與第一個範例相同的資料集和情境。

不過，在此範例中，資料集保持不變並且會載入到應用程式中。會建立傳回交易發生年份開始日期時間戳記的計算，作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- id
- date

若要計算交易發生在哪一年，建立下列量值：

- =yearstart(date)
- =timestamp(yearstart(date))

結果表格

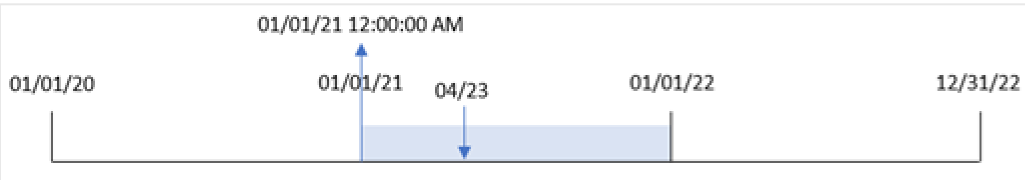
id	日期	=yearstart(date)	=timestamp(yearstart(date))
8188	06/06/2022	01/01/2022	1/1/2022 12:00:00 AM
8189	07/18/2022	01/01/2022	1/1/2022 12:00:00 AM

id	日期	=yearstart(date)	=timestamp(yearstart(date))
8190	11/14/2022	01/01/2022	1/1/2022 12:00:00 AM
8191	12/12/2022	01/01/2022	1/1/2022 12:00:00 AM
8192	01/22/2021	01/01/2021	1/1/2021 12:00:00 AM
8193	02/03/2021	01/01/2021	1/1/2021 12:00:00 AM
8194	03/17/2021	01/01/2021	1/1/2021 12:00:00 AM
8195	04/23/2021	01/01/2021	1/1/2021 12:00:00 AM
8196	05/04/2021	01/01/2021	1/1/2021 12:00:00 AM
8197	06/30/2021	01/01/2021	1/1/2021 12:00:00 AM
8198	07/26/2021	01/01/2021	1/1/2021 12:00:00 AM
8199	12/27/2021	01/01/2021	1/1/2021 12:00:00 AM
8200	01/13/2020	01/01/2020	1/1/2020 12:00:00 AM
8201	02/26/2020	01/01/2020	1/1/2020 12:00:00 AM
8202	03/27/2020	01/01/2020	1/1/2020 12:00:00 AM
8203	04/16/2020	01/01/2020	1/1/2020 12:00:00 AM
8204	05/21/2020	01/01/2020	1/1/2020 12:00:00 AM
8205	08/14/2020	01/01/2020	1/1/2020 12:00:00 AM
8206	10/07/2020	01/01/2020	1/1/2020 12:00:00 AM
8207	12/05/2020	01/01/2020	1/1/2020 12:00:00 AM

「start_of_year」量值在圖表物件中的建立方式是使用 yearstart() 函數並傳遞日期欄位，作為函數的引數。

yearstart() 函數最初識別日期值落在哪一年，並傳回該年第一毫秒的時間戳記。

yearstart() 函數圖表和交易 8199。



交易 8199 發生在 2021 年 4 月 23 日。yearstart() 函數傳回該年的第一毫秒，亦即 1 月 1 日上午 12:00:00。

範例 5 – 情境

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 資料集載入到稱為「Loans」的表格。該表格含有下列欄位:
  - 貸款 ID。
  - 該年開始時的餘額。
  - 在每年每筆貸款收取的單利率。

最終使用者希望圖表物件按貸款 ID 顯示年初至今每筆貸款已累積的目前利息。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- loan_id
- start_balance

若要計算累積的利息,建立下列量值:

```
=start_balance*(rate*(today(1)-yearstart(today(1)))/365)
```

將量值的**數字格式**設定為**金錢**。

結果表格

loan_id	start_balance	=start_balance*(rate*(today(1)-yearstart(today(1)))/365)
8188	$10000.00	$39.73
8189	$15000.00	$339.66

loan_id	start_balance	=start_balance*(rate*(today(1)-yearstart(today(1)))/365)
8190	$17500.00	$166.85
8191	$21000.00	$283.64
8192	$90000.00	$3003.29

`yearstart()` 函數使用今日日期作為其唯一引數，傳回目前年份的開始日期。以目前日期減去該結果後，運算式會傳回今年目前已經過的天數。

然後此值乘以利率並除以 365，以傳回該期間的有效利率。然後該期間的有效利率乘以貸款的開始餘額，以傳回今年目前已累積的利息。

yeartodate

此功能會尋找輸入時間戳記是否位於前次載入指令碼之日期的年份內，並傳回 True (如果在其內)，False (如果不在其內)。

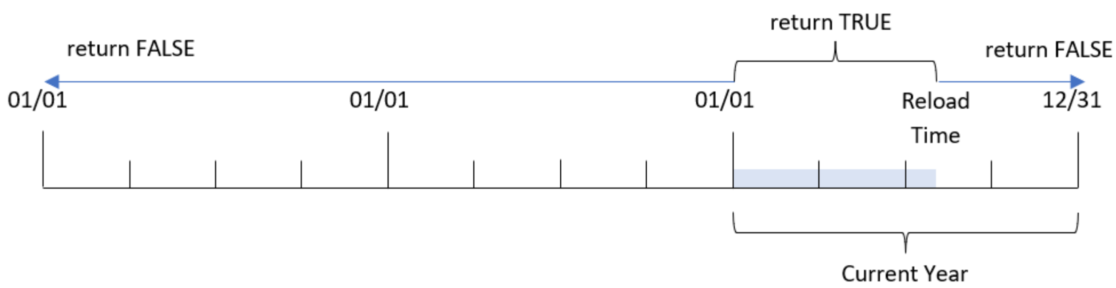
語法：

```
YearToDate(timestamp[ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

傳回的資料類型： 布林

在 Qlik Sense 中，布林值 true 值以 -1 代表，而 false 值以 0 代表。

yeartodate() 函數的範例圖表



如未使用任何選用參數，年初至今表示是從 1 月 1 日起，在一個日曆年內的任何日期，但不超過 (可包含) 上次指令碼執行的日期。

換言之，不透過其他參數觸發時，`yeartodate()` 函數會用來評估時間戳記，並根據日曆年內是否出現日期來傳回布林值結果，但不超過 (可包含) 重新載入的發生日期。

但是，這也可能使用 `firstmonth` 引數取代年份開始日期，以及使用 `yearoffset` 引數比較前一年或下一年。

最後，在歷史資料集的例子中，`yeartodate()` 函數提供設定 `todaydate` 的參數，這也會改為比較日曆年的時間戳記，但不超過 (可包含) `todaydate` 引數中提供的日期。

引數

引數	描述
timestamp	要評估的時間戳記，例如「10/12/2012」。
yearoffset	透過指定 **yearoffset**，**yeartodate** 針對另一年中的相同期間傳回 True。負數 **yearoffset** 表示前一年，正數是未來年的位移。最新的年結束日期透過指定 yearoffset = -1 實現。如果省略，則採用 0。
firstmonth	透過指定 **firstmonth** 介於 1 和 12 之間 (如果省略，則為 1)，可將年開始向前移到任何月份的第一天。例如，若您想要處理 5 月 1 日開始的會計年度，請指定 **firstmonth** = 5。1 的值表示會計年度開始於 1 月 1 日，而 12 的值表示會計年度開始於 12 月 1 日。
todaydate	透過指定 **todaydate** (如果省略，則為上次指令碼執行的時間戳記)，即可移動作為期間上限的日期。

什麼情況下使用

`yeartodate()` 函數會傳回布林值結果。通常，此函數類型將作為 if 運算式中的條件使用。這會傳回彙總或計算，其相依於每年出現的評估日期，但不超過 (可包含) 最後一次應用程式重新載入的日期。

例如，YearToDate() 函數可用來識別目前年份迄今製造的所有設備。

下列範例假設最後一個重新載入時間 = 11/18/2011。

函數範例

範例	結果
`yeartodate( '11/18/2010')`	傳回 False
`yeartodate( '02/01/2011')`	傳回 True
`yeartodate( '11/18/2011')`	傳回 True
`yeartodate( '11/19/2011')`	傳回 False
`yeartodate( '11/19/2011', 0, 1, '12/31/2011')`	傳回 True
`yeartodate( '11/18/2010', -1)`	傳回 True
`yeartodate( '11/18/2011', -1)`	傳回 False
`yeartodate( '04/30/2011', 0, 5)`	傳回 False
`yeartodate( '05/01/2011', 0, 5)`	傳回 True

區域設定

除非另有說明，否則此主題中的範例皆使用下列日期格式：MM/DD/YYYY。日期格式是在資料載入指令碼的 `SET DateFormat` 陳述式中指定。由於地區設定和其他因素，您系統中的預設日期格式可能會不同。您可以變更以下範例中的格式，以滿足您的需求。或者，您可以在載入指令碼中變更格式，以符合這些範例。

應用程式中的預設地區設定是根據安裝 Qlik Sense 之電腦或伺服器的地區系統設定。若您存取的 Qlik Sense 伺服器設定為瑞典，資料載入編輯器將會對日期、時間和貨幣使用瑞典文地區設定。這些地區格式設定與 Qlik Sense 使用者介面中顯示的語言無關。Qlik Sense 顯示的語言將與您正在使用的瀏覽器相同。

範例 1 – 基本範例

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2020 年和 2022 年之間交易的資料集，這載入到稱為「`Transactions`」的表格中。
- 以 `DateFormat` 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。
- 建立欄位 `year_to_date`，這決定哪些交易發生在截至最後一次重新載入日期的日曆年。

撰稿時，日期為 2022 年 4 月 26 日。

載入指令碼

```
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
];
```

結果

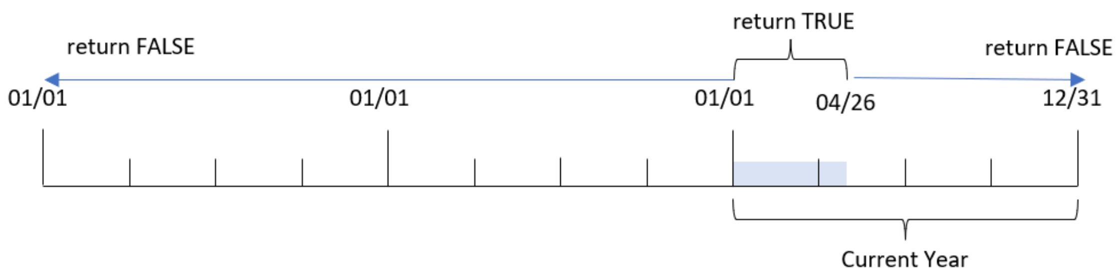
載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- year_to_date

結果表格

日期	年初至今
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

yeartodate() 函數的圖表, 基本範例



year_to_date 欄位在前置 LOAD 陳述式中的建立方式是使用 yeartodate() 函數, 並傳遞 date 欄位, 作為函數的引數。

因為沒有其他參數傳遞至函數, 所以 yeartodate() 函數最初會識別重新載入日期, 目前日曆年 (開始於 1 月 1 日) 的界限因此會傳回 TRUE 的布林值結果。

因此, 任何發生在 1 月 1 日和 4 月 26 日的交易 (重新載入日期) 會傳回 TRUE 的布林值結果。任何發生在 2022 年開始前的交易會傳回 FALSE 的布林值結果。

範例 2 – yearoffset

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 與第一個範例相同的資料集和情境。
- 建立欄位 two_years_prior, 這決定哪些交易發生在日曆年初至今的完整兩年之前。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- two_years_prior

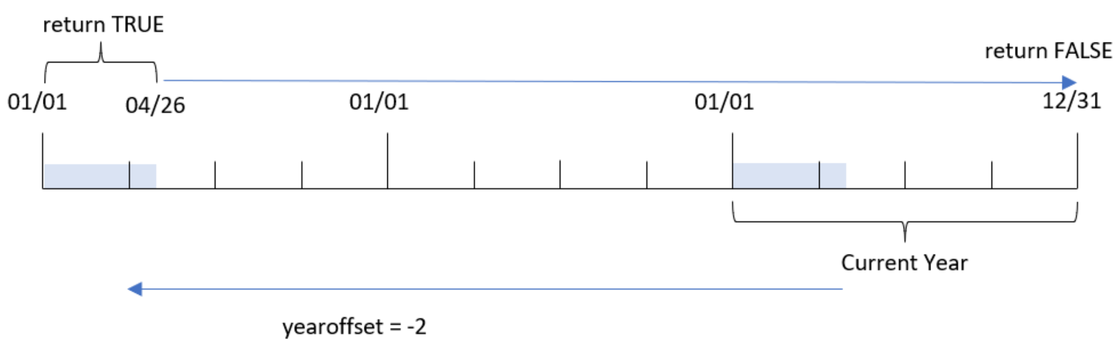
結果表格

日期	two_years_prior
01/10/2020	-1
02/28/2020	-1
04/09/2020	-1
04/16/2020	-1
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0

日期	two_years_prior
07/26/2021	0
12/27/2021	0
02/02/2022	0
02/26/2022	0
03/07/2022	0
03/11/2022	0

透過使用 -2 作為 `yeartodate()` 函數中的 `yearoffset` 引數，函數會使用完整兩年轉移比較子日曆年區段的界限。一開始，年份區段等同於 2022 年 1 月 1 日和 4 月 26 日之間。然後 `yearoffset` 引數會將此區段偏移至兩年前。然後日期界限會落在 2020 年 1 月 1 日和 4 月 26 日之間。

`yeartodate()` 函數，yearoffset 範例



因此，發生在 2020 年 1 月 1 日和 4 月 26 日之間的任何交易將會傳回 TRUE 的布林值結果 。任何發生在此區段前後的交易會傳回 FALSE。

範例 3 – firstmonth

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 與第一個範例相同的資料集和情境。
- 建立欄位 `year_to_date`，這決定哪些交易發生在截至最後一次重新載入日期的日曆年。

在此範例中，我們將會計年度開始設為 7 月 1 日。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

- date
- year_to_date

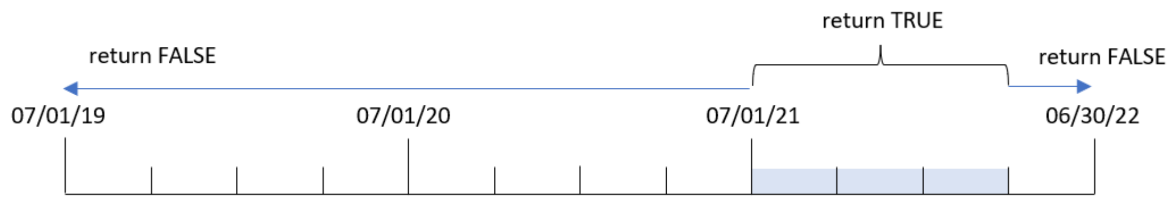
結果表格

日期	年初至今
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0

日期	年初至今
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

在此例中，因為 `firstmonth` 引數的值 7 用於 `yeartodate()` 函數，所以這將該年的第一天設為 7 月 1 日，而該年的最後一天則為 6 月 30 日。

`yeartodate()` 函數的圖表，firstmonth 範例



因此，任何發生在 2021 年 7 月 1 日和 2022 年 4 月 26 日的交易 (重新載入日期) 會傳回 TRUE 的布林值結果。任何發生在 2021 年 7 月 1 日前的交易會傳回 FALSE 的布林值結果。

範例 4 – todaydate

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 與第一個範例相同的資料集和情境。
- 建立欄位 `year_to_date`, 這決定哪些交易發生在截至最後一次重新載入日期的日曆年。

但是在此範例中, 我們必須識別所有發生在日曆年中的交易, 但不超過 (可包含) 2022 年 3 月 1 日。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度：

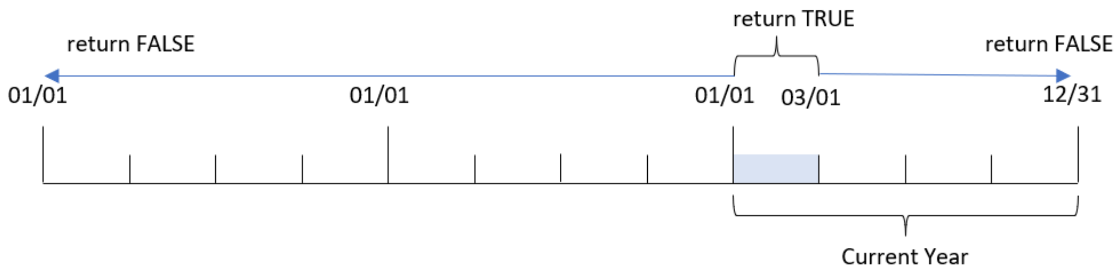
- date
- year_to_date

結果表格

日期	年初至今
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

在此例中，因為 `todaydate` 引數的值 03/01/2022 用於 `yeartodate()` 函數，所以這會將比較子日曆年區段的界限設為 2022 年 3 月 1 日。請務必提供 `firstmonth` 參數 (介於 1 和 2 之間)；否則，該函數會傳回 Null 值結果。

yeartodate() 函數的圖表，範例使用 todaydate 引數



因此，發生在 2022 年 1 月 1 日和 2022 年 3 月 1 日之間的任何交易 todaydate 參數，將會傳回 TRUE 的布林值結果。任何發生在 2022 年 1 月 1 日前和 2022 年 3 月 1 日後的交易，將會傳回 FALSE 的布林值結果。

範例 – 圖表物件範例

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含與第一個範例相同的資料集和情境。

不過，在此範例中，不變的資料集會載入到應用程式中。會建立決定哪些交易發生在日曆年但不超過 (可包含) 最後一次重新載入的計算，作為應用程式圖表物件中的量值。

載入指令碼

```
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
];
```

結果

載入資料並開啟工作表。建立新的表格並將此欄位新增為維度：date。

新增下列量值：

```
=yeartodate(date)
```

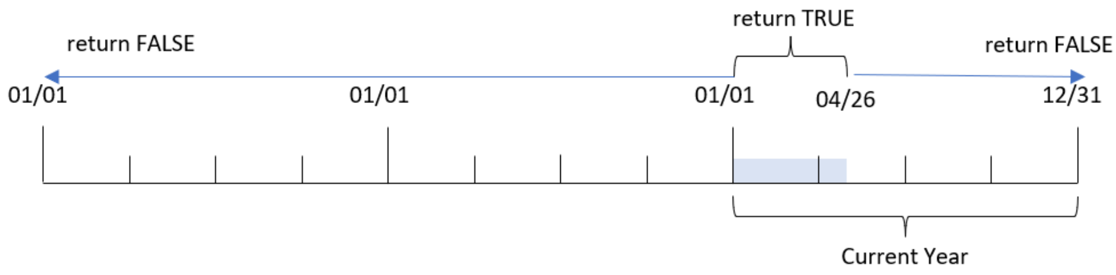
結果表格

日期	=yeartodate(date)
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

year_to_date 量值在圖表物件中的建立方式是使用 yeartodate() 函數並傳遞 date 欄位，作為函數的引數。

因為沒有其他參數傳遞至函數，所以 yeartodate() 函數最初會識別重新載入日期，目前日曆年 (開始於 1 月 1 日) 的界限因此會傳回 TRUE 的布林值結果。

yeartodate() 函數的圖表，圖表物件範例



任何發生在 1 月 1 日和 4 月 26 日的交易 (重新載入日期) 會傳回 TRUE 的布林值結果。任何發生在 2022 年開始前的交易會傳回 FALSE 的布林值結果。

範例 6 – 情境

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含：

- 包含一組 2020 年和 2022 年之間交易的資料集，這載入到稱為 Transactions 的表格中。
- 以 DateFormat 系統變數 (MM/DD/YYYY) 格式提供的日期欄位。

最終使用者想要 KPI 物件呈現總銷售額，且日期為等同於最後一次重新載入時間之目前年份至今的 2021 年期間。

撰稿時，日期為 2022 年 6 月 16。

載入指令碼

```
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
```

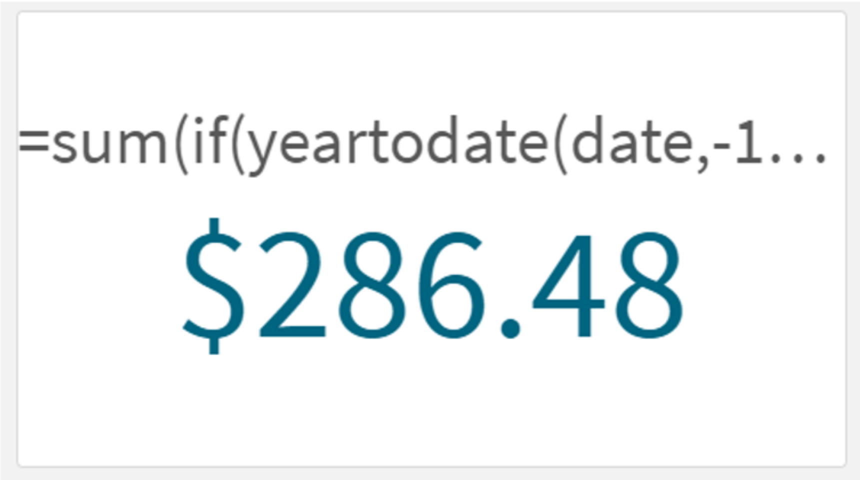
```
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

結果

請執行下列動作:

1. 建立 KPI 物件。
2. 建立下列彙總量值來計算總銷售額:
   `=sum(if(yeartodate(date,-1),amount,0))`
3. 將量值的**數字格式**設定為**金錢**。

2021 年 KPI yeartodate() 圖表



yeartodate() 函數會在評估每筆交易 ID 的日期時傳回布林值。因為重新載入發生在 2022 年 6 月 16 日，所以 yeartodate 函數會將 01/01/2022 和 06/16/2022 之間的年份期間分段。但是，由於 period_no 值的 -1 用於該函數，因此這些界限會接著轉移至前一年。因此，對於任何發生在 01/01/2021 和 06/16/2021 之間的交易，yeartodate() 函數會傳回 TRUE 的布林值結果並加總金額。

8.8 指數與對數函數

本節描述和指數與對數計算相關的函數。所有函數皆可用於資料載入指令碼和圖表運算式。

在下列函數中，參數是運算式，其中 **x** 和 **y** 應解譯為真實的值數字。

exp
自然指數函數 e^x，將自然對數 **e** 用作基數。結果是正數。

```
exp(x )
```

範例與結果：

exp(3) 傳回 20.085。

log
x 的自然對數。只有 **x**> 0 時才定義此函數。結果是數字。

```
log(x )
```

範例與結果：

log(3) 傳回 1.0986

log10
x的常用對數 (基數 10)。只有 **x**> 0 時才定義此函數。結果是數字。

```
log10(x )
```

範例與結果：

log10(3) 傳回 0.4771

pow
將 **x** 傳回到 **y** 的次方。結果是數字。

```
pow(x,y )
```

範例與結果：

pow(3, 3) 傳回 27

sqr
x 平方值 (**x** 到 2 的次方)。結果是數字。

```
sqr (x )
```

範例與結果:

`sqr(3)` 傳回 9

sqrt

x 的平方根。只有 **x** >= 0 時才定義此函數。結果是正數。

```
sqrt(x )
```

範例與結果:

`sqrt(3)` 傳回 1.732

8.9 欄位函數

這些函數僅可用於圖表運算式。

欄位函數會傳回識別欄位選項不同層面的整數或字串。

計數函數

GetAlternativeCount

GetAlternativeCount() 用來尋找已識別欄位中的替代 (淺灰色) 值的數目。

```
GetAlternativeCount - 圖表函數 (field_name)
```

GetExcludedCount

GetExcludedCount() 用來尋找已識別欄位中的已排除相異值的數目。已排除的值包括替代選項 (淺灰色)、已排除 (深灰色) 和已選取的已排除 (含有核取記號的深灰色) 欄位。

```
GetExcludedCount - 圖表函數 (page 1125)(field_name)
```

GetNotSelectedCount

此圖表函數會傳回名為 **fieldname** 的欄位中未選取值的數量。該欄位必須處於 And 模式，此函數才有意義。

```
GetNotSelectedCount - 圖表函數(fieldname [, includeexcluded=false])
```

GetPossibleCount

GetPossibleCount() 用來尋找已識別欄位中的可能值的數目。如果已識別的欄位包括選項，則會計算已選取的 (綠色) 欄位。否則，會計算相關聯的 (白色) 值。

```
GetPossibleCount - 圖表函數(field_name)
```

GetSelectedCount

GetSelectedCount() 會尋找欄位中已選取的 (綠色) 值的數目。

```
GetSelectedCount - 圖表函數 (field_name [, include_excluded])
```

欄位與選項函數

GetCurrentSelections

GetCurrentSelections() 傳回應用程式中的目前選項清單。若改為在搜尋方塊中使用搜尋字串進行選取，則 **GetCurrentSelections()** 會傳回搜尋字串。

```
GetCurrentSelections - 圖表函數([record_sep [,tag_sep [,value_sep [,max_
values]]]])
```

GetFieldSelections

GetFieldSelections() 傳回 **string** 及欄位中的目前選項。

```
GetFieldSelections - 圖表函數 ( field_name [, value_sep [, max_values]])
```

GetObjectDimension

GetObjectDimension() 會傳回維度名稱。**Index** 為選用整數，代表應傳回的維度。

```
GetObjectDimension - 圖表函數 ([index])
```

GetObjectField

GetObjectField() 會傳回維度名稱。**Index** 為選用整數，代表應傳回的維度。

```
GetObjectField - 圖表函數 ([index])
```

GetObjectMeasure

GetObjectMeasure() 可傳回量值名稱。**Index** 是代表應傳回之量值的選用整數。

```
GetObjectMeasure - 圖表函數 ([index])
```

GetAlternativeCount - 圖表函數

GetAlternativeCount() 用來尋找已識別欄位中的替代 (淺灰色) 值的數目。

語法:

```
GetAlternativeCount (field_name)
```

傳回的資料類型: 整數

引數:

引數

引數	描述
field_name	包含待測量之資料範圍的欄位。

範例與結果:

下列範例使用載入到篩選窗格的 **First name** 欄位。

範例與結果

範例	結果
假定已在 **First name** 中選取 **John**。 `GetAlternativeCount ([First name])`	4, 由於 **First name** 中有 4 個唯一且排除 (灰色) 的值。
假定已選取 **John** 和 **Peter**。 `GetAlternativeCount ([First name])`	3, 由於 **First name** 中有 3 個唯一且排除 (灰色) 的值。
假定未在 **First name** 中選取任何值。 `GetAlternativeCount ([First name])`	0, 因為無任何選項。

範例中使用的資料:

```
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 - 圖表函數

GetCurrentSelections() 傳回應用程式中的目前選項清單。若改為在搜尋方塊中使用搜尋字串進行選取, 則 **GetCurrentSelections()** 會傳回搜尋字串。

若選項已使用, 您必須指定 record_sep。若要指定新的行, 請將 **record_sep** 設為 **chr(13)&chr(10)**。

若只有兩個或一個值未選取, 會分別使用格式「NOT x,y」或「NOT y」。若選取所有值, 且所有值的計數大於 max_values, 則會傳回文字 ALL。

語法:

```
GetCurrentSelections ([record_sep [, tag_sep [, value_sep [, max_values [, state_name]]]]])
```

傳回的資料類型: 字串

引數:

引數

引數	描述
record_sep	置於欄位記錄之間的分隔符號。預設為 <CR><LF>, 表示新行。
tag_sep	置於欄位名稱標記和欄位值之間的分隔符號。預設為「: 」。

引數	描述
value_sep	要置於欄位值之間的分隔符號。預設為 ', '。
max_values	要個別列出的欄位值上限。要個別列出的欄位值上限。若選取大量的值，則會改用 'x of y values' (第 x 個值，共 y 個值) 的格式。預設為 6。
state_name	為特定視覺化選擇的替代狀態名稱。如果使用 **state_name** 引數，則只有與指定狀態名稱關聯的選項才會納入考量。

範例與結果：

以下範例使用載入至不同篩選窗格的兩個欄位，一個用於 **First name** 名字，另一個用於 **Initials**。

範例與結果

範例	結果
假定已在 **First name** 中選取 **John**。 `GetCurrentSelections ()`	'First name: John'
假定已在 **First name** 中選取 **John** 和 **Peter**。 `GetCurrentSelections ()`	'First name: John, Peter'
假定已在 **First name** 中選取 **John** 和 **Peter**、且在 **Initials** 中選取 **JA**。 `GetCurrentSelections ()`	'First name: John, Peter Initials: JA'
假定已在 **First name** 中選取 **John**、且在 **Initials** 中選取 **JA**。 `GetCurrentSelections ( chr(13)&chr(10) , ' = ' )`	'First name = John Initials = JA'
假定您已在 **First name** 中選取了 Sue 以外的所有名字，且未選取 **Initials** 中的任何選項。 `GetCurrentSelections (chr(13)&chr(10), '=', ',' ,3)`	'First name=NOT Sue'

範例中使用的資料：

```
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 - 圖表函數

GetExcludedCount() 用來尋找已識別欄位中的已排除相異值的數目。已排除的值包括替代選項 (淺灰色)、已排除 (深灰色) 和已選取的已排除 (含有核取記號的深灰色) 欄位。

語法:

```
GetExcludedCount (field_name)
```

傳回的資料類型: 字串

引數:

引數

引數	描述
field_name	包含待測量之資料範圍的欄位。

範例與結果:

以下範例使用載入至不同篩選窗格的三個欄位,一個用於 **First name**,另一個用於 **Last name**,其餘一個則用於 **Initials**。

範例與結果

範例	結果
若未在 **First name** 中選取任何值。	`GetExcludedCount (Initials) = 0` 有太多選項。
若在 **First name** 中選取了 **John**。	`GetExcludedCount (Initials) = 5` 在 **Initials** 中有 5 個已排除的值,帶有深灰色。第六個儲存格 (JA) 將是白色,因為它與 **First name** 中的選項 John 相關。
若已選取 **John** 和 **Peter**。	`GetExcludedCount (Initials) = 3` 在 **Initials** 中,John 與 1 值有關,而 Peter 與 2 值有關。
若在 **First name** 中選取 **John** 和 **Peter**,然後在 **Last name** 中選取 **Franc**。	`GetExcludedCount ([First name]) = 4` **名字**中有 4 個已排除的值,帶有深灰色。**GetExcludedCount ()** 會評估具有已排除值的欄位,包括替代選項和已選取的已排除值。
若在 **First name** 中選取 **John** 和 **Peter**,然後在 **Last name** 中選取 **Franc** 和 **Anderson**。	`GetExcludedCount (Initials) = 4` 在 **Initials** 中有 4 個已排除的值,帶有深灰色。其他兩個儲存格 (JA 和 PF) 將是白色,因為這與 **First name** 中的選項 John 和 Peter 相關。
若在 **First name** 中選取 **John** 和 **Peter**,然後在 **Last name** 中選取 **Franc** 和 **Anderson**。	`GetExcludedCount ([Last name]) = 4` **Initials** 中有 4 個已排除的值。Devonshire 有淺灰色,而 Brown、Carr 和 Elliot 則有深灰色。

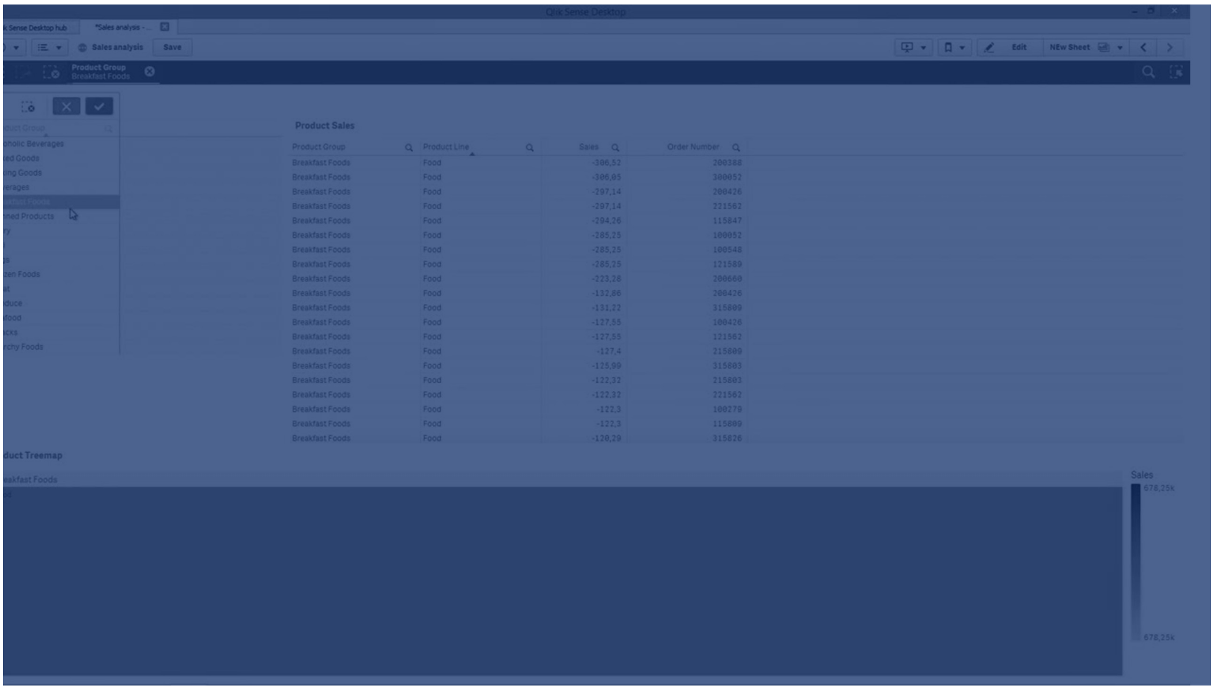
範例中使用的資料:

```
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 - 圖表函數

GetFieldSelections() 傳回 **string** 及欄位中的目前選項。



若只有兩個或一個值未選取，會分別使用格式「NOT x,y」或「NOT y」。若選取所有值，且所有值的計數大於max_values，則會傳回文字 ALL。

語法：

```
GetFieldSelections ( field_name [, value_sep [, max_values [, state_name]]])
```

傳回的資料類型： 字串

傳回字串格式

格式	描述
'a, b, c'	若所選值的數量為 max_values 或更少，傳回的字串是所選值的清單。 值以 value_sep 作為分隔符號來區隔。
'NOT a, b, c'	若未選值的數量為 max_values 或更少，傳回的字串是未選值的清單，以 NOT 為前置詞。 值以 value_sep 作為分隔符號來區隔。

格式	描述
'x of y'	x = 所選值的數量 y = 值的總數 這會在 max_values < x < (y - max_values) 時傳回。
'ALL'	若選取所有值，則會傳回。
'-'	若未選取任何值，則會傳回。
<search string>	若您已使用搜尋選取，會傳回搜尋字串。

引數：

引數

引數	描述
field_name	包含待測量之資料範圍的欄位。
value_sep	要置於欄位值之間的分隔符號。預設為 ','。
max_values	要個別列出的欄位值上限。要個別列出的欄位值上限。若選取大量的值，則會改用 'x of y values' (第 x 個值，共 y 個值) 的格式。預設為 6。
state_name	為特定視覺化選擇的替代狀態名稱。如果使用 **state_name** 引數，則只有與指定狀態名稱關聯的選項才會納入考量。

範例與結果：

下列範例使用載入到篩選窗格的 **First name** 欄位。

範例與結果

範例	結果
假定已在 **First name** 中選取 **John**。 `GetFieldSelections ([First name])`	「John」
假定已選取 **John** 和 **Peter**。 `GetFieldSelections ([First name])`	「John,Peter」
假定已選取 **John** 和 **Peter**。 `GetFieldSelections ([First name],'; ')`	「John; Peter」

範例	結果
假定已在 **First name** 中選取 **John**、**Sue**、**Mark**。 `GetFieldSelections ([First name],';',2)`	「NOT Jane;Peter」, 因為值 2 指明為 max_values 引數的值。若未指明, 結果就會是 John; Sue; Mark.

範例中使用的資料:

```
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 - 圖表函數

此圖表函數會傳回名為 **fieldname** 的欄位中未選取值的數量。該欄位必須處於 And 模式, 此函數才有意義。

語法:

```
GetNotSelectedCount(fieldname [, includeexcluded=false])
```

引數:

引數

引數	描述
fieldname	待評估欄位的名稱。
includeexcluded	若 **includeexcluded** 被描述為 True, 則計數會包括受其他欄位中的選項所排除的選取值。

範例:

```
GetNotSelectedCount( Country )
```

```
GetNotSelectedCount( Country, true )
```

GetObjectDimension - 圖表函數

GetObjectDimension() 會傳回維度名稱。**Index** 為選用整數, 代表應傳回的維度。

您無法在下列位置的圖表中使用此功能:標題、子標題、頁尾、參考線運算式和最小/最大運算式。

您無法在使用 Object ID 的另一個物件中參考維度或量值名稱。

語法:

```
GetObjectDimension ([index])
```

範例:

```
GetObjectDimension(1)
```

範例: 圖表運算式

在圖表運算式中顯示 GetObjectDimension 函數範例的 Qlik Sense 表格。

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

若您想要傳回量值名稱, 請改用 **GetObjectMeasure** 函數。

GetObjectField - 圖表函數

GetObjectField() 會傳回維度名稱。**Index** 為選用整數, 代表應傳回的維度。

您無法在下列位置的圖表中使用此功能:標題、子標題、頁尾、參考線運算式和最小/最大運算式。

您無法在使用 Object ID 的另一個物件中參考維度或量值名稱。

語法:

```
GetObjectField ([index])
```

範例:

```
GetObjectField(1)
```

範例：圖表運算式

在圖表運算式中顯示 GetObjectField 函數範例的 Qlik Sense 表格。

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

若您想要傳回量值名稱，請改用 **GetObjectMeasure** 函數。

GetObjectMeasure - 圖表函數

GetObjectMeasure() 可傳回量值名稱。**Index** 是代表應傳回之量值的選用整數。

您無法在下列位置的圖表中使用此功能：標題、子標題、頁尾、參考線運算式和最小/最大運算式。

您無法在使用 Object ID 的另一個物件中參考維度或量值名稱。

語法：

```
GetObjectMeasure ([index])
```

範例：

```
GetObjectMeasure(1)
```

範例：圖表運算式

在圖表運算式中顯示 GetObjectMeasure 函數範例的 Qlik Sense 表格。

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)

若您想要傳回維度名稱，請改用 **GetObjectField** 函數。

GetPossibleCount - 圖表函數

GetPossibleCount() 用來尋找已識別欄位中的可能值的數目。如果已識別的欄位包括選項，則會計算已選取的 (綠色) 欄位。否則，會計算相關聯的 (白色) 值。

針對選項欄位，**GetPossibleCount()** 會傳回所選 (綠色) 欄位的數目。

傳回的資料類型： 整數

語法：

```
GetPossibleCount (field_name)
```

引數：

引數

引數	描述
field_name	包含待測量之資料範圍的欄位。

範例與結果：

以下範例使用載入至不同篩選窗格的兩個欄位，一個用於 **First name** 名字，另一個用於 **Initials**。

範例與結果

範例	結果
假定已在 **First name** 中選取 **John**。 `GetPossibleCount ([Initials])`	1，因為在 **First name** 中，與選項 **John** 相關的 Initials 中有 1 個值。
假定已在 **First name** 中選取 **John**。 `GetPossibleCount ([First name])`	1，由於 **First name** 中有 1 個選項**John**。
假定已在 **First name** 中選取 **Peter**。 `GetPossibleCount ([Initials])`	2，因為 Peter 與 **Initials** 中的 2 個值相關。
假定未在 **First name** 中選取任何值。 `GetPossibleCount ([First name])`	5，因為沒有任何選項，而 **First name** 中有 5 個唯一值。
假定未在 **First name** 中選取任何值。 `GetPossibleCount ([Initials])`	6，因為沒有任何選項，而 **Initials** 中有 6 個唯一值。

範例中使用的資料：

```
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 - 圖表函數

GetSelectedCount() 會尋找欄位中已選取的 (綠色) 值的數目。

語法：

```
GetSelectedCount (field_name [, include_excluded [, state_name]])
```

傳回的資料類型： 整數

引數：

引數

引數	描述
field_name	包含待測量之資料範圍的欄位。
include_excluded	若設為 **True()**，則計數會包括目前受其他欄位中的選項所排除的選取值。若為 False 或省略的話，就不會包括這些值。
state_name	為特定視覺化選擇的替代狀態名稱。如果使用 **state_name** 引數，則只有與指定狀態名稱關聯的選項才會納入考量。

範例與結果：

以下範例使用載入至不同篩選窗格的三個欄位，一個用於 **First name** 名稱，另一個用於 **Initials**，其餘一個則用於 **Has cellphone**。

範例與結果

範例	結果
假定已在 **First name** 中選取 **John**。 `GetSelectedCount ([First name])`	1，因為在 **First name** 中選取了一個值。
假定已在 **First name** 中選取 **John**。 `GetSelectedCount ([Initials])`	0，因為沒有在 **Initials** 中選取任何值。
First name 中未選取任何選項，請選取 **Initials** 中的所有值，接著在 **Has cellphone** 中選取 **Yes**。 `GetSelectedCount ([Initials], True())`	6. 雖然 **Initials**MC 和 PD 選項已將 **Has cellphone** 設為 **No**，但結果仍是 6，因為引數 include_excluded 設定為 True()。

範例中使用的資料：

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
```

```
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

8.10 檔案函數

此檔案函數 (僅可用於指令碼運算式) 會傳回目前正在讀取之表格檔案的相關資訊。對於表格檔案之外的所有資料來源，這些函數都會傳回 NULL (例外：**ConnectString( )**)。

檔案函數概述

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

Attribute

此指令碼函數會以文字形式傳回各種媒體檔案的中繼標記值。支援下列檔案格式：MP3、WMA、WMV、PNG 和 JPG。如果檔案 **filename** 不存在、非支援的檔案格式，或不包含名為 **attributename** 的中繼標記，則會傳回 NULL。

```
Attribute (filename, attributename)
```

ConnectString

ConnectString() 函數會傳回 ODBC 或 OLE DB 連線的作用中資料連線名稱。如果尚未執行任何 **connect** 陳述式，或在 **disconnect** 陳述式之後，則該函數會傳回空字串。

```
ConnectString ()
```

FileBaseName

FileBaseName 函數會傳回一個字串，其中包含目前正在讀取之表格檔案的名稱，不含路徑或副檔名。

```
FileBaseName ()
```

FileDir

FileDir 函數會傳回一個字串，其中包含目前正在讀取之表格檔案的目錄路徑。

```
FileDir ()
```

FileExtension

FileExtension 函數會傳回一個字串，其中包含目前正在讀取之表格檔案的副檔名。

```
FileExtension ()
```

FileName

FileName 函數會傳回一個字串，其中包含目前正在讀取之表格檔案的名稱，不含路徑，但包含副檔名。

```
FileName ()
```

FilePath

FilePath 函數會傳回一個字串，其中包含目前正在讀取之表格檔案的完整路徑。

```
FilePath ()
```

FileSize

FileSize 函數會傳回一個整數，其中包含檔案 filename 或 (若未指定 filename) 目前正在讀取之表格檔案的大小 (以位元組為單位)。

```
FileSize ()
```

FileTime

FileTime 函數會以指定檔案上次修改的 UTC 格式傳回時間戳記。若未指定檔案，函數會以目前讀取表格檔案上次修改的 UTC 傳回時間戳記。

```
FileTime ([ filename ])
```

GetFolderPath

GetFolderPath 函數會傳回 Microsoft Windows *SHGetFolderPath* 函數的值。此函數會輸入 Microsoft Windows 資料夾的名稱，並傳回資料夾的完整路徑。

```
GetFolderPath ()
```

QvdCreateTime

此指令碼函數會從 QVD 檔案傳回 XML 標頭時間戳記 (如果呈現的話)，否則會傳回 NULL。在時間戳記中，所提供時間為協調世界時。

```
QvdCreateTime (filename)
```

QvdFieldName

此指令碼函數會傳回 QVD 檔案中的欄位編號 **fieldno** 名稱。如果該欄位不存在，則會傳回 NULL。

```
QvdFieldName (filename , fieldno)
```

QvdNoOfFields

此指令碼函數會傳回 QVD 檔案中的欄位數。

```
QvdNoOfFields (filename)
```

QvdNoOfRecords

此指令碼函數會傳回 QVD 檔案中的目前記錄數。

```
QvdNoOfRecords (filename)
```

QvdTableName

此指令碼函數會傳回 QVD 檔案中儲存的表格名稱。

```
QvdTableName (filename)
```

Attribute

此指令碼函數會以文字形式傳回各種媒體檔案的中繼標記值。支援下列檔案格式:MP3、WMA、WMV、PNG 和 JPG。如果檔案 **filename** 不存在、非支援的檔案格式,或不包含名為 **attributename** 的中繼標記,則會傳回 NULL。

語法:

```
Attribute(filename, attributename)
```

大量中繼標記可以讀取。本主題中的範例顯示可以從各個受支援的檔案類型中讀取的標記。

> *您僅可根據相關規格讀取儲存在檔案中的中繼標記 (例如 MP3 檔案的 ID2v3 或,JPG 檔案的 EXIF),不能讀取儲存在 **Windows 檔案總管**中的中繼資訊。*

引數:

引數

引數	描述
filename	包括路徑在內的媒體檔案名稱,用作資料夾資料連線 (如有需要的話)。 **範例: *'lib://Table Files/'*** 在舊式指令碼模式下,還支援下列路徑格式: • 絕對路徑 **範例: *c:\data*** • 與 Qlik Sense 應用程式工作目錄相關。 **範例: *data***
attributename	中繼標記的名稱。

此範例使用 **GetFolderPath** 函數來找到媒體檔案的路徑。由於 **GetFolderPath** 只在舊版模式下受支援,當您在標準模式或在 Qlik Sense SaaS 中使用此功能時,您需要使用 lib:// 資料連線路徑取代對 **GetFolderPath** 的參考。

檔案系統存取限制 (page 1421)

Example 1: MP3 檔案

此指令碼會讀取 *MyMusic* 資料夾中所有可能的 MP3 中繼標記。

```
// 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

此指令碼會讀取 *MyPictures* 資料夾內 JPG 檔案中所有可能的 EXIF 中繼標記。

```
// Script to read Jpeg Exif meta tags
for each vExt in 'jpg', 'jpeg', 'jpe', 'jfif', 'jif', 'jfi'
for each vFoundFile in filelist( GetFolderPath('MyPictures') & '\*.'& vExt )

FileList:
LOAD FileLongName,
```

```
subfield(FileLongName,'\',-1) as FileShortName,
num(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,
FileTime(FileLongName) as FileTime,
// ************   Exif Main (IFD0) Attributes   ************
Attribute(FileLongName, 'ImageWidth') as ImageWidth,
Attribute(FileLongName, 'ImageLength') as ImageLength,
Attribute(FileLongName, 'BitsPerSample') as BitsPerSample,
Attribute(FileLongName, 'Compression') as Compression,

//  examples: 1=uncompressed, 2=CCITT, 3=CCITT 3, 4=CCITT 4,

//5=LZW, 6=JPEG (old style), 7=JPEG, 8=Deflate, 32773=PackBits RLE,
Attribute(FileLongName, 'PhotometricInterpretation') as PhotometricInterpretation,

//  examples: 0=WhiteIsZero, 1=BlackIsZero, 2=RGB, 3=Palette, 5=CMYK, 6=YCbCr,
Attribute(FileLongName, 'ImageDescription') as ImageDescription,
Attribute(FileLongName, 'Make') as Make,
Attribute(FileLongName, 'Model') as Model,
Attribute(FileLongName, 'StripOffsets') as StripOffsets,
Attribute(FileLongName, 'Orientation') as Orientation,

//  examples: 1=TopLeft, 2=TopRight, 3=BottomRight, 4=BottomLeft,

// 5=LeftTop, 6=RightTop, 7=RightBottom, 8=LeftBottom,
Attribute(FileLongName, 'SamplesPerPixel') as SamplesPerPixel,
Attribute(FileLongName, 'RowsPerStrip') as RowsPerStrip,
Attribute(FileLongName, 'StripByteCounts') as StripByteCounts,
Attribute(FileLongName, 'XResolution') as XResolution,
Attribute(FileLongName, 'YResolution') as YResolution,
Attribute(FileLongName, 'PlanarConfiguration') as PlanarConfiguration,

//  examples: 1=chunky format, 2=planar format,
Attribute(FileLongName, 'ResolutionUnit') as ResolutionUnit,

//  examples: 1=none, 2=inches, 3=centimeters,
Attribute(FileLongName, 'TransferFunction') as TransferFunction,
Attribute(FileLongName, 'Software') as Software,
Attribute(FileLongName, 'DateTime') as DateTime,
Attribute(FileLongName, 'Artist') as Artist,
Attribute(FileLongName, 'HostComputer') as HostComputer,
Attribute(FileLongName, 'WhitePoint') as WhitePoint,
Attribute(FileLongName, 'PrimaryChromaticities') as PrimaryChromaticities,
Attribute(FileLongName, 'YCbCrCoefficients') as YCbCrCoefficients,
Attribute(FileLongName, 'YCbCrSubSampling') as YCbCrSubSampling,
Attribute(FileLongName, 'YCbCrPositioning') as YCbCrPositioning,

//  examples: 1=centered, 2=co-sited,
Attribute(FileLongName, 'ReferenceBlackWhite') as ReferenceBlackWhite,
Attribute(FileLongName, 'Rating') as Rating,
Attribute(FileLongName, 'RatingPercent') as RatingPercent,
Attribute(FileLongName, 'ThumbnailFormat') as ThumbnailFormat,

//  examples: 0=Raw Rgb, 1=Jpeg,
Attribute(FileLongName, 'Copyright') as Copyright,
Attribute(FileLongName, 'ExposureTime') as ExposureTime,
```

```
    Attribute(FileLongName, 'FNumber') as FNumber,
    Attribute(FileLongName, 'ExposureProgram') as ExposureProgram,

    //  examples: 0=Not defined, 1=Manual, 2=Normal program, 3=Aperture priority, 4=Shutter
priority,

    // 5=Creative program, 6=Action program, 7=Portrait mode, 8=Landscape mode, 9=Bulb,
    Attribute(FileLongName, 'ISOSpeedRatings') as ISOSpeedRatings,
    Attribute(FileLongName, 'TimeZoneOffset') as TimeZoneOffset,
    Attribute(FileLongName, 'SensitivityType') as SensitivityType,

    //  examples: 0=Unknown, 1=Standard output sensitivity (SOS), 2=Recommended exposure index
(REI),

    // 3=ISO speed, 4=Standard output sensitivity (SOS) and Recommended exposure index (REI),

    //5=Standard output sensitivity (SOS) and ISO Speed, 6=Recommended exposure index (REI)
and ISO Speed,

    // 7=Standard output sensitivity (SOS) and Recommended exposure index (REI) and ISO speed,
    Attribute(FileLongName, 'ExifVersion') as ExifVersion,
    Attribute(FileLongName, 'DateTimeOriginal') as DateTimeOriginal,
    Attribute(FileLongName, 'DateTimeDigitized') as DateTimeDigitized,
    Attribute(FileLongName, 'ComponentsConfiguration') as ComponentsConfiguration,

    //  examples: 1=Y, 2=Cb, 3=Cr, 4=R, 5=G, 6=B,
    Attribute(FileLongName, 'CompressedBitsPerPixel') as CompressedBitsPerPixel,
    Attribute(FileLongName, 'ShutterSpeedValue') as ShutterSpeedValue,
    Attribute(FileLongName, 'ApertureValue') as ApertureValue,
    Attribute(FileLongName, 'BrightnessValue') as BrightnessValue, //  examples: -1=Unknown,
    Attribute(FileLongName, 'ExposureBiasValue') as ExposureBiasValue,
    Attribute(FileLongName, 'MaxApertureValue') as MaxApertureValue,
    Attribute(FileLongName, 'SubjectDistance') as SubjectDistance,

    //  examples: 0=Unknown, -1=Infinity,
    Attribute(FileLongName, 'MeteringMode') as MeteringMode,

    //  examples: 0=Unknown, 1=Average, 2=CenterWeightedAverage, 3=Spot,

    // 4=MultiSpot, 5=Pattern, 6=Partial, 255=Other,
    Attribute(FileLongName, 'LightSource') as LightSource,

    //  examples: 0=Unknown, 1=Daylight, 2=Fluorescent, 3=Tungsten, 4=Flash, 9=Fine weather,

    // 10=Cloudy weather, 11=Shade, 12=Daylight fluorescent,

    // 13=Day white fluorescent, 14=Cool white fluorescent,

    // 15=White fluorescent, 17=Standard light A, 18=Standard light B, 19=Standard light C,

    // 20=D55, 21=D65, 22=D75, 23=D50, 24=ISO studio tungsten, 255=other light source,
    Attribute(FileLongName, 'Flash') as Flash,
    Attribute(FileLongName, 'FocalLength') as FocalLength,
    Attribute(FileLongName, 'SubjectArea') as SubjectArea,
    Attribute(FileLongName, 'MakerNote') as MakerNote,
```

```
    Attribute(FileLongName, 'UserComment') as UserComment,
    Attribute(FileLongName, 'SubSecTime') as SubSecTime,

    Attribute(FileLongName, 'SubsecTimeOriginal') as SubsecTimeOriginal,
    Attribute(FileLongName, 'SubsecTimeDigitized') as SubsecTimeDigitized,
    Attribute(FileLongName, 'XPTitle') as XPTitle,
    Attribute(FileLongName, 'XPComment') as XPComment,

    Attribute(FileLongName, 'XPAuthor') as XPAuthor,
    Attribute(FileLongName, 'XPKeywords') as XPKeywords,
    Attribute(FileLongName, 'XPSubject') as XPSubject,
    Attribute(FileLongName, 'FlashpixVersion') as FlashpixVersion,
    Attribute(FileLongName, 'ColorSpace') as ColorSpace, //  examples: 1=sRGB,
65535=Uncalibrated,
    Attribute(FileLongName, 'PixelXDimension') as PixelXDimension,
    Attribute(FileLongName, 'PixelYDimension') as PixelYDimension,
    Attribute(FileLongName, 'RelatedSoundFile') as RelatedSoundFile,

    Attribute(FileLongName, 'FocalPlaneXResolution') as FocalPlaneXResolution,
    Attribute(FileLongName, 'FocalPlaneYResolution') as FocalPlaneYResolution,
    Attribute(FileLongName, 'FocalPlaneResolutionUnit') as FocalPlaneResolutionUnit,

    //  examples: 1=None, 2=Inch, 3=Centimeter,
    Attribute(FileLongName, 'ExposureIndex') as ExposureIndex,
    Attribute(FileLongName, 'SensingMethod') as SensingMethod,

    //  examples: 1=Not defined, 2=One-chip color area sensor, 3=Two-chip color area sensor,

    // 4=Three-chip color area sensor, 5=Color sequential area sensor,

    // 7=Trilinear sensor, 8=Color sequential linear sensor,
    Attribute(FileLongName, 'FileSource') as FileSource,

    //  examples: 0=Other, 1=Scanner of transparent type,

    // 2=Scanner of reflex type, 3=Digital still camera,
    Attribute(FileLongName, 'SceneType') as SceneType,

    //  examples: 1=A directly photographed image,
    Attribute(FileLongName, 'CFAPattern') as CFAPattern,
    Attribute(FileLongName, 'CustomRendered') as CustomRendered,

    //  examples: 0=Normal process, 1=Custom process,
    Attribute(FileLongName, 'ExposureMode') as ExposureMode,

    //  examples: 0=Auto exposure, 1=Manual exposure, 2=Auto bracket,
    Attribute(FileLongName, 'WhiteBalance') as WhiteBalance,

    //  examples: 0=Auto white balance, 1=Manual white balance,
    Attribute(FileLongName, 'DigitalZoomRatio') as DigitalZoomRatio,
    Attribute(FileLongName, 'FocalLengthIn35mmFilm') as FocalLengthIn35mmFilm,
    Attribute(FileLongName, 'SceneCaptureType') as SceneCaptureType,

    //  examples: 0=Standard, 1=Landscape, 2=Portrait, 3=Night scene,
    Attribute(FileLongName, 'GainControl') as GainControl,
```

```
    //  examples: 0=None, 1=Low gain up, 2=High gain up, 3=Low gain down, 4=High gain down,
    Attribute(FileLongName, 'Contrast') as Contrast,

    //  examples: 0=Normal, 1=Soft, 2=Hard,
    Attribute(FileLongName, 'Saturation') as Saturation,

    //  examples: 0=Normal, 1=Low saturation, 2=High saturation,
    Attribute(FileLongName, 'Sharpness') as Sharpness,

    //  examples: 0=Normal, 1=Soft, 2=Hard,
    Attribute(FileLongName, 'SubjectDistanceRange') as SubjectDistanceRange,

    //  examples: 0=Unknown, 1=Macro, 2=Close view, 3=Distant view,
    Attribute(FileLongName, 'ImageUniqueID') as ImageUniqueID,
    Attribute(FileLongName, 'BodySerialNumber') as BodySerialNumber,
    Attribute(FileLongName, 'CMNT_GAMMA') as CMNT_GAMMA,
    Attribute(FileLongName, 'PrintImageMatching') as PrintImageMatching,
    Attribute(FileLongName, 'OffsetSchema') as OffsetSchema,

    // ************   Interoperability Attributes   ************
    Attribute(FileLongName, 'InteroperabilityIndex') as InteroperabilityIndex,
    Attribute(FileLongName, 'InteroperabilityVersion') as InteroperabilityVersion,
    Attribute(FileLongName, 'InteroperabilityRelatedImageFileFormat') as
InteroperabilityRelatedImageFileFormat,
    Attribute(FileLongName, 'InteroperabilityRelatedImageWidth') as
InteroperabilityRelatedImageWidth,
    Attribute(FileLongName, 'InteroperabilityRelatedImageLength') as
InteroperabilityRelatedImageLength,
    Attribute(FileLongName, 'InteroperabilityColorSpace') as InteroperabilityColorSpace,

    //  examples: 1=sRGB, 65535=Uncalibrated,
    Attribute(FileLongName, 'InteroperabilityPrintImageMatching') as
InteroperabilityPrintImageMatching,
    // ************   GPS Attributes   ************
    Attribute(FileLongName, 'GPSVersionID') as GPSVersionID,
    Attribute(FileLongName, 'GPSLatitudeRef') as GPSLatitudeRef,
    Attribute(FileLongName, 'GPSLatitude') as GPSLatitude,
    Attribute(FileLongName, 'GPSLongitudeRef') as GPSLongitudeRef,
    Attribute(FileLongName, 'GPSLongitude') as GPSLongitude,
    Attribute(FileLongName, 'GPSAltitudeRef') as GPSAltitudeRef,

    //  examples: 0=Above sea level, 1=Below sea level,
    Attribute(FileLongName, 'GPSAltitude') as GPSAltitude,
    Attribute(FileLongName, 'GPSTimeStamp') as GPSTimeStamp,
    Attribute(FileLongName, 'GPSSatellites') as GPSSatellites,
    Attribute(FileLongName, 'GPSStatus') as GPSStatus,
    Attribute(FileLongName, 'GPSMeasureMode') as GPSMeasureMode,
    Attribute(FileLongName, 'GPSDOP') as GPSDOP,
    Attribute(FileLongName, 'GPSSpeedRef') as GPSSpeedRef,

    Attribute(FileLongName, 'GPSSpeed') as GPSSpeed,
    Attribute(FileLongName, 'GPSTrackRef') as GPSTrackRef,
    Attribute(FileLongName, 'GPSTrack') as GPSTrack,
    Attribute(FileLongName, 'GPSImgDirectionRef') as GPSImgDirectionRef,
    Attribute(FileLongName, 'GPSImgDirection') as GPSImgDirection,
```

```
    Attribute(FileLongName, 'GPSMapDatum') as GPSMapDatum,
    Attribute(FileLongName, 'GPSDestLatitudeRef') as GPSDestLatitudeRef,

    Attribute(FileLongName, 'GPSDestLatitude') as GPSDestLatitude,
    Attribute(FileLongName, 'GPSDestLongitudeRef') as GPSDestLongitudeRef,
    Attribute(FileLongName, 'GPSDestLongitude') as GPSDestLongitude,
    Attribute(FileLongName, 'GPSDestBearingRef') as GPSDestBearingRef,
    Attribute(FileLongName, 'GPSDestBearing') as GPSDestBearing,
    Attribute(FileLongName, 'GPSDestDistanceRef') as GPSDestDistanceRef,

    Attribute(FileLongName, 'GPSDestDistance') as GPSDestDistance,
    Attribute(FileLongName, 'GPSProcessingMethod') as GPSProcessingMethod,
    Attribute(FileLongName, 'GPSAreaInformation') as GPSAreaInformation,
    Attribute(FileLongName, 'GPSDateStamp') as GPSDateStamp,
    Attribute(FileLongName, 'GPSDifferential') as GPSDifferential;

    //  examples: 0=No correction, 1=Differential correction,
LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no labels);
Next vFoundFile
Next vExt
```

Example 3: Windows 媒體檔案

此指令碼會讀取 *MyMusic* 資料夾中所有可能的 WMA/WMV ASF 中繼標記。

```
/ 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

此指令碼會讀取 *MyPictures* 資料夾中所有可能的 PNG 中繼標記。

```
// Script to read PNG meta tags
for each vExt in 'png'
for each vFoundFile in filelist( GetFolderPath('MyPictures') & '\*.'& vExt )

FileList:
LOAD FileLongName,
    subfield(FileLongName,'\',-1) as FileShortName,
    num(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,
    FileTime(FileLongName) as FileTime,
    Attribute(FileLongName, 'Comment') as Comment,

    Attribute(FileLongName, 'Creation Time') as Creation_Time,
    Attribute(FileLongName, 'Source') as Source,
    Attribute(FileLongName, 'Title') as Title,
    Attribute(FileLongName, 'Software') as Software,
    Attribute(FileLongName, 'Author') as Author,
    Attribute(FileLongName, 'Description') as Description,

    Attribute(FileLongName, 'Copyright') as Copyright;
LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no labels);
Next vFoundFile
Next vExt
```

ConnectString

ConnectString() 函數會傳回 ODBC 或 OLE DB 連線的作用中資料連線名稱。如果尚未執行任何 **connect** 陳述式，或在 **disconnect** 陳述式之後，則該函數會傳回空字串。

語法：

```
ConnectString()
```

範例與結果：

指令碼處理範例

範例	結果

`LIB CONNECT TO 'Tutorial ODBC';`
 	
`ConnectString:`
 	
`Load ConnectString() as ConnectString`
`AutoGenerate 1;`
	在 ConnectString 欄位中傳回 'Tutorial ODBC'。
	
	此範例假定您有一個稱為 Tutorial ODBC 的可用資料連線。

FileBaseName

FileBaseName 函數會傳回一個字串，其中包含目前正在讀取之表格檔案的名稱，不含路徑或副檔名。

語法：

```
FileBaseName()
```

範例與結果：

指令碼處理範例

範例	結果

`LOAD *, filebasename( ) as X from`
 	
`C:\UserFiles\abc.txt`
	會在每筆讀取的記錄中，於欄位 X 傳回「abc」。

FileDir

FileDir 函數會傳回一個字串，其中包含目前正在讀取之表格檔案的目錄路徑。

語法：

```
FileDir()
```

此函數僅在標準模式下支援資料夾資料連線。

範例與結果：

指令碼處理範例

範例	結果

`Load *, filedir( ) as X from`
 	
`C:\UserFiles\abc.txt`
	會在每筆讀取的記錄中，於欄位 X 傳回 'C:\UserFiles'。

FileExtension

FileExtension 函數會傳回一個字串，其中包含目前正在讀取之表格檔案的副檔名。

語法：

```
FileExtension()
```

範例與結果：

指令碼處理範例

範例	結果

`LOAD *, FileExtension( ) as X from`
 	
 `C:\UserFiles\abc.txt`
	會在每筆讀取的記錄中，於欄位 X 傳回「txt」。

FileName

FileName 函數會傳回一個字串，其中包含目前正在讀取之表格檔案的名稱，不含路徑，但包含副檔名。

語法：

```
FileName()
```

範例與結果：

指令碼處理範例

範例	結果

`LOAD *, FileName( ) as X from`
 	
 `C:\UserFiles\abc.txt`
	會在每筆讀取的記錄中，於欄位 X 傳回 'abc.txt'。

FilePath

FilePath 函數會傳回一個字串，其中包含目前正在讀取之表格檔案的完整路徑。

語法：

```
FilePath()
```

此函數僅在標準模式下支援資料夾資料連線。

範例與結果：

指令碼處理範例

範例	結果

`Load *, FilePath( ) as X from`
 	
 `C:\UserFiles\abc.txt`
	會在每筆讀取的記錄中，於欄位 X 傳回 'C:\UserFiles\abc.txt'。

FileSize

FileSize 函數會傳回一個整數，其中包含檔案 filename 或 (若未指定 filename) 目前正在讀取之表格檔案的大小 (以位元組為單位)。

語法：

```
FileSize([filename])
```

引數：

引數

引數	**描述**
filename	如果有必要納入路徑，檔案名稱作為資料夾或 Web 檔案資料連線。若您沒有指定檔案名稱，表格檔案目前準備好提供使用。 **範例：*'lib://Table Files/'*** 在舊式指令碼模式下，還支援下列路徑格式： • 絕對路徑 **範例：*c:\data*** • 與 Qlik Sense 應用程式工作目錄相關。 **範例：*data*** • 指向網際網路位置或內部網路位置的 URL 位址 (HTTP 或 FTP)。 **範例：*http://www.qlik.com***

範例與結果：

指令碼處理範例

範例	**結果**
`LOAD *, FileSize( ) as X from abc.txt;`	會在每筆讀取的記錄中，以整數的形式於欄位 X 傳回指定檔案 (abc.txt) 的大小。
`FileSize( 'lib://DataFiles/xyz.xls' )`	會傳回檔案 xyz.xls 的大小。

FileTime

FileTime 函數會以指定檔案上次修改的 UTC 格式傳回時間戳記。若未指定檔案，函數會以目前讀取表格檔案上次修改的 UTC 傳回時間戳記。

語法：

```
FileTime([ filename ])
```

引數：

引數

引數	描述
filename	如果需要，檔案的名稱將包括路徑作為資料夾或網站檔案資料連線。 **範例：*'lib://Table Files/'*** 在舊式指令碼模式下，還支援下列路徑格式： • 絕對路徑 **範例：*c:\data*** • 與 Qlik Sense 應用程式工作目錄相關。 **範例：*data*** • 指向網際網路位置或內部網路位置的 URL 位址 (HTTP 或 FTP)。 **範例：*http://www.qlik.com***

範例與結果：

指令碼範例

範例	結果
`LOAD *, FileTime( ) as X from abc.txt;`	會在每筆讀取的記錄中，於欄位 X 傳回檔案 (abc.txt) 上次修改的時間戳記。
`FileTime( 'xyz.xls' )`	會傳回檔案 xyz.xls 上次修改的時間戳記。

GetFolderPath

GetFolderPath 函數會傳回 Microsoft Windows *SHGetFolderPath* 函數的值。此函數會輸入 Microsoft Windows 資料夾的名稱，並傳回資料夾的完整路徑。

標準模式下不支援這項功能。。

語法：

```
GetFolderPath(foldername)
```

引數：

引數

引數	描述
foldername	Microsoft Windows 資料夾的名稱 資料夾名稱不應包含任何空格。Windows Explorer 中的資料夾名稱包含的任何空格均應刪去。 範例： *MyMusic* *MyDocuments*

範例與結果：

此範例的目標在於獲取以下 Microsoft Windows 資料夾的路徑：*MyMusic*、*MyPictures* 和 *Windows*。將範例指令碼新增到您的應用程式中並重新載入。

```
LOAD
 GetFolderPath('MyMusic') as MyMusic,
 GetFolderPath('MyPictures') as MyPictures,
 GetFolderPath('Windows') as Windows
AutoGenerate 1;
```

重新載入應用程式後，會新增欄位 *MyMusic*、*MyPictures* 和 *Windows* 到資料模型中。每個欄位均包含輸入中所定義資料夾的路徑。例如：

- *C:\Users\smu\Music* for the folder *MyMusic*
- *C:\Users\smu\Pictures* for the folder *MyPictures*
- *C:\Windows* for the folder *Windows*

QvdCreateTime

此指令碼函數會從 QVD 檔案傳回 XML 標頭時間戳記 (如果呈現的話)，否則會傳回 NULL。在時間戳記中，所提供時間為協調世界時。

語法：

```
QvdCreateTime(filename)
```

引數：

引數

引數	描述
filename	QVD 檔案的名稱，如果需要，則包括路徑作為資料夾或網站資料連線。 **範例：*'lib://Table Files/'*** 在舊式指令碼模式下，還支援下列路徑格式： • 絕對路徑 **範例：*c:\data*** • 與 Qlik Sense 應用程式工作目錄相關。 **範例：*data*** • 指向網際網路位置或內部網路位置的 URL 位址 (HTTP 或 FTP)。 **範例：*http://www.qlik.com***

範例：

```
QvdCreateTime('MyFile.qvd')
```

```
QvdCreateTime('C:\MyDir\MyFile.qvd')
```

```
QvdCreateTime('lib://DataFiles/MyFile.qvd')
```

QvdFieldName

此指令碼函數會傳回 QVD 檔案中的欄位編號 **fieldno** 名稱。如果該欄位不存在，則會傳回 NULL。

語法：

```
QvdFieldName(filename , fieldno)
```

引數：

引數

引數	描述
filename	QVD 檔案的名稱，如果需要，則包括路徑作為資料夾或網站資料連線。 **範例：*'lib://Table Files/'*** 在舊式指令碼模式下，還支援下列路徑格式： • 絕對路徑 **範例：*c:\data*** • 與 Qlik Sense 應用程式工作目錄相關。 **範例：*data*** • 指向網際網路位置或內部網路位置的 URL 位址 (HTTP 或 FTP)。 **範例：*http://www.qlik.com***
fieldno	QVD 檔案中包含之表格內欄位的編號。

範例：

```
QvdFieldName ('MyFile.qvd', 5)

QvdFieldName ('C:\MyDir\MyFile.qvd', 5)

QvdFieldName ('lib://DataFiles/MyFile.qvd', 5)
```

所有三個範例傳回 QVD 檔案中包含之表格的第五個欄位名稱。

QvdNoOfFields

此指令碼函數會傳回 QVD 檔案中的欄位數。

語法：

```
QvdNoOfFields(filename)
```

引數：

引數

引數	描述
filename	QVD 檔案的名稱，如果需要，則包括路徑作為資料夾或網站資料連線。 **範例： *'lib://Table Files/'*** 在舊式指令碼模式下，還支援下列路徑格式： • 絕對路徑 **範例： *c:\data*** • 與 Qlik Sense 應用程式工作目錄相關。 **範例： *data*** • 指向網際網路位置或內部網路位置的 URL 位址 (HTTP 或 FTP)。 **範例： *http://www.qlik.com***

範例：

```
QvdNoOfFields ('MyFile.qvd')

QvdNoOfFields ('C:\MyDir\MyFile.qvd')

QvdNoOfFields ('lib://DataFiles/MyFile.qvd')
```

QvdNoOfRecords

範例：此指令碼函數會傳回 QVD 檔案中的目前記錄數。

語法：

```
QvdNoOfRecords(filename)
```

引數：

引數

引數	描述
filename	QVD 檔案的名稱，如果需要，則包括路徑作為資料夾或網站資料連線。 **範例：*'lib://Table Files/'*** 在舊式指令碼模式下，還支援下列路徑格式： • 絕對路徑 **範例：*c:\data*** • 與 Qlik Sense 應用程式工作目錄相關。 **範例：*data*** • 指向網際網路位置或內部網路位置的 URL 位址 (HTTP 或 FTP)。 **範例：*http://www.qlik.com***

範例：

```
QvdNoOfRecords ('MyFile.qvd')

QvdNoOfRecords ('C:\MyDir\MyFile.qvd')

QvdNoOfRecords ('lib://DataFiles/MyFile.qvd')
```

QvdTableName

此指令碼函數會傳回 QVD 檔案中儲存的表格名稱。

語法：

```
QvdTableName(filename)
```

引數：

引數

引數	描述
filename	QVD 檔案的名稱，如果需要，則包括路徑作為資料夾或網站資料連線。 **範例：*'lib://Table Files/'*** 在舊式指令碼模式下，還支援下列路徑格式： • 絕對路徑 **範例：*c:\data*** • 與 Qlik Sense 應用程式工作目錄相關。 **範例：*data*** • 指向網際網路位置或內部網路位置的 URL 位址 (HTTP 或 FTP)。 **範例：*http://www.qlik.com***

範例：

```
QvdTableName ('MyFile.qvd')

QvdTableName ('C:\MyDir\MyFile.qvd')

QvdTableName ('lib://data\MyFile.qvd')
```

8.11 財務函數

財務函數可用於資料載入指令碼和圖表運算式中，計算付款和利率。
至於所有引數，支出的現金以負數表示。收取的現金以正數表示。
此處列出財務函數中使用的引數 (不包括以 **range-** 為開頭的引數)：

*如需所有財務函數，請務必在指定 **rate** 和 **nper** 的單位時保持一致。如果對於 6% 年利率的 5 年期貸款進行每月還款，可在 **rate** 使用 0.005 (6%/12)，並在 **nper** 使用 60 (5*12)。如果對於同一筆貸款進行每年還款，可在 **rate** 使用 6%，並在 **nper** 使用 5。*

財務函數概述

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

FV

此函數會傳回根據定期固定還款和單一年利率的未來還款值。

```
FV (rate, nper, pmt [ ,pv [ , type ] ])
```

nPer

此函數會傳回根據定期固定還款和固定利率的還款期數。

```
nPer (rate, pmt, pv [ ,fv [ , type ] ])
```

Pmt

此函數會傳回根據定期固定還款和固定利率的貸款金額。它不會隨著貸款的時間長度而變化。支付以負數表示，例如，-20。

```
Pmt (rate, nper, pv [ ,fv [ , type ] ] )
```

PV

此函數會傳回還款的現值。

```
PV (rate, nper, pmt [ ,fv [ , type ] ])
```

Rate

此函數會傳回貸款的年利率。結果具有 **Fix** 兩位小數和 % 的預設數字格式。

```
Rate (nper, pmt , pv [ ,fv [ , type ] ])
```

BlackAndSchole

Black and Scholes 模型是用於財務市場衍生工具檢測的數學模型。公式會計算選擇權的理論值。在 Qlik Sense 中，**BlackAndSchole** 函數會按照 Black and Scholes 未修改的公式 (歐式選擇權) 傳回值。

```
BlackAndSchole(strike , time_left , underlying_price , vol , risk_free_rate , type)
```

傳回的資料類型：數值

引數：

引數

引數	描述
strike	股票的未來購買價格。
time_left	剩餘的時間週期數。
underlying_price	目前的股票值。
vol	(股價) 波動根據時間週期以小數形式的百分數表示。
risk_free_rate	無風險收益率根據時間週期以小數形式的百分數表示。

引數	描述
call_or_put	選擇權的類型： 對於 call options，是 'c', 'call' 或任何非零數值 對於 put options，是 'p', 'put' 或 0。

限制：

strike、time_left 和 underlying_price 值必須 >0。

vol 和 risk_free_rate 值必須為 <0 或 >0。

範例與結果：

指令碼處理範例

範例	結果
`BlackAndSchole(130, 4, 68.5, 0.4, 0.04, 'call')` 這會計算股票每股價值目前為 68.5，每股價值 130 時 4 年內購買選擇權的理論價格。此公式使用的每年波動性為 0.4 (40%) 且無風險收益率為 0.04 (4%)。	傳回 11.245

FV

此函數會傳回根據定期固定還款和單一年利率的未來還款值。

語法：

```
FV(rate, nper, pmt [ ,pv [ , type ] ])
```

傳回的資料類型： 數值。依照預設，結果將會格式化為貨幣。.

引數：

引數

引數	描述
rate	各個期間的利率。
nper	還款總年數。
pmt	每期支付的款項。它不會隨著貸款的時間長度而變化。支付以負數表示，例如，-20。
pv	一系列未來還款目前所值的現值或一次付清金額。如果省略 **pv**，則將假設為 0 (零)。
type	如果還款截止時間是在期末，則應該為 0，如果還款截止時間是在期初，則應該為 1。如果省略 **type**，則將假設為 0。

範例與結果：

指令碼處理範例

範例	結果
您分期 36 個月支付新的家用電器，每個月繳 20 美元。年利率是 6%。帳單在每月底寄達。最後一期的帳單繳完時，還款的總金額是多少？ `FV(0.005,36,-20)`	傳回 $786.72

nPer

此函數會傳回根據定期固定還款和固定利率的還款期數。

語法：

```
nPer(rate, pmt, pv [ ,fv [ , type ] ])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
rate	各個期間的利率。
nper	還款總年數。
pmt	每期支付的款項。它不會隨著貸款的時間長度而變化。支付以負數表示，例如，-20。
pv	一系列未來還款目前所值的現值或一次付清金額。如果省略 **pv**，則將假設為 0 (零)。
fv	最後一次還款後得到的未來值或現金餘額。如果省略 **fv**，則將假設為 0。
type	如果還款截止時間是在期末，則應該為 0，如果還款截止時間是在期初，則應該為 1。如果省略 **type**，則將假設為 0。

範例與結果：

指令碼處理範例

範例	結果
您要以每月分期付 20 美元的方式出售家用電器。年利率是 6%。帳單在每月底寄達。如果最後一期帳單付清之後，金額等於 800 美元，需要繳多少期？ `nPer(0.005,-20,0,800)`	傳回 36.56

Pmt

此函數會傳回根據定期固定還款和固定利率的貸款金額。它不會隨著貸款的時間長度而變化。支付以負數表示，例如，-20。

```
Pmt(rate, nper, pv [ ,fv [ , type ] ] )
```

傳回的資料類型: 數值。依照預設, 結果將會格式化為貨幣。.

若要找出貸款期間總還款金額, 可將傳回的 **pmt** 值乘以 **nper**。

引數:

引數

引數	描述
rate	各個期間的利率。
nper	還款總年數。
pv	一系列未來還款目前所值的現值或一次付清金額。如果省略 **pv**, 則將假設為 0 (零)。
fv	最後一次還款後得到的未來值或現金餘額。如果省略 **fv**, 則將假設為 0。
type	如果還款截止時間是在期末, 則應該為 0, 如果還款截止時間是在期初, 則應該為 1。如果省略 **type**, 則將假設為 0。

範例與結果:

指令碼處理範例

範例	結果
下列公式傳回年利率 10% 且必須分期 8 個月付清的 20,000 美元貸款的按月償付金額: `Pmt(0.1/12,8,20000)`	傳回 -$2,594.66
對於同一筆貸款, 如果還款截止時間是在期初, 則金額為: `Pmt(0.1/12,8,20000,0,1)`	傳回 -$2,573.21

PV

此函數會傳回還款的現值。

```
PV(rate, nper, pmt [ ,fv [ , type ] ])
```

傳回的資料類型: 數值。依照預設, 結果將會格式化為貨幣。.

現值是一系列未來還款目前所值的總金額。例如, 進行貸款時, 貸款金額是貸方的現值。

引數:

引數

引數	描述
rate	各個期間的利率。
nper	還款總年數。

引數	描述
pmt	每期支付的款項。它不會隨著貸款的時間長度而變化。支付以負數表示, 例如, -20。
fv	最後一次還款後得到的未來值或現金餘額。如果省略 **fv**, 則將假設為 0。
type	如果還款截止時間是在期末, 則應該為 0, 如果還款截止時間是在期初, 則應該為 1。如果省略 **type**, 則將假設為 0。

範例與結果:

指令碼處理範例

範例	結果
當您必須分 5 年在每個月月底還款 100 美元時, 假設利率為 7%, 則借款現值是多少? `PV(0.07/12,12*5,-100,0,0)`	傳回 $5,050.20

Rate

此函數會傳回貸款的年利率。結果具有 **Fix** 兩位小數和 % 的預設數字格式。

語法:

```
Rate(nper, pmt , pv [ ,fv [ , type ] ])
```

傳回的資料類型: 數值。

rate 會經過反覆運算, 結果會是 0 或其他數字。如果 **rate** 的連續結果都不相同, 則將傳回 NULL 值。

引數:

引數

引數	描述
nper	還款總年數。
pmt	每期支付的款項。它不會隨著貸款的時間長度而變化。支付以負數表示, 例如, -20。
pv	一系列未來還款目前所值的現值或一次付清金額。如果省略 **pv**, 則將假設為 0 (零)。
fv	最後一次還款後得到的未來值或現金餘額。如果省略 **fv**, 則將假設為 0。
type	如果還款截止時間是在期末, 則應該為 0, 如果還款截止時間是在期初, 則應該為 1。如果省略 **type**, 則將假設為 0。

範例與結果:

指令碼處理範例

範例	結果
對於 10,000 美金的 5 年期貸款每月償付 300 美元, 利率是多少? `Rate(60,-300,10000)`	傳回 2.00%

8.12 格式設定函數

格式設定函數會強制設定輸入數值欄位或運算式的顯示格式，根據資料類型，您可以指定小數點分隔符號、千位分隔符號等的字元。

函數全部會傳回同時具有字串和數值，但是可以視為執行數字到字串轉換的雙值。**Dual()** 是一種特殊情況，但是其他格式化函數會採用輸入運算式的數值，並產生代表數字的字串。

而解譯函數執行相反的操作：它們會使用字串運算式並評估為數字，指定所產生數字的格式。

這些函數可以同時用於資料載入指令碼和圖表運算式中。

所有的數字表示法都會加入小數點作為小數點分隔符號。

格式設定函數概述

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

ApplyCodepage

ApplyCodepage() 在運算式中指明的欄位或文字上套用不同的代碼頁面字元設定。**codepage** 引數必須為數字格式。

```
ApplyCodepage (text, codepage)
```

Date

Date() 將運算式的格式設定為日期，使用在資料載入指令碼或作業系統或格式字串 (如果提供的話) 的系統變數中設定的格式。

```
Date (number[, format])
```

Dual

Dual() 將一個數字與一個字串組合至單一記錄，例如記錄的數字表示法可用於排序及計算，而字串值可用於顯示。

```
Dual (text, number)
```

Interval

Interval() 將數字的格式設定為時間間隔，使用在資料載入指令碼或作業系統或格式字串 (如果提供的話) 的系統變數中設定的格式。

```
Interval (number[, format])
```

Money

Money() 將運算式的格式以數值方式設定為金額值，使用在資料載入指令碼或作業系統 (除非提供格式字串) 的系統變數集中設定的格式，以及選用的小數與千位分隔符號。

```
Money (number[, format[, dec_sep [, thou_sep]]])
```

Num

Num() 格式化數字，亦即使用第二參數指定的格式將輸入的數值轉換為顯示文字。若省略第二參數，這會使用資料載入指令碼中設定的小數點和千位分隔符號。自訂小數與千位分隔符號是選用參數。

```
Num (number[, format[, dec_sep [, thou_sep]]])
```

Time

Time() 將運算式的格式設定為時間值，使用在資料載入指令碼或作業系統 (除非提供格式字串) 的系統變數中設定的時間格式。

```
Time (number[, format])
```

Timestamp

TimeStamp() 將運算式的格式設定為日期和時間值，使用在資料載入指令碼或作業系統 (除非提供格式字串) 的系統變數中設定的時間戳記格式。

```
Timestamp (number[, format])
```

另請參見：

解譯函數 (page 1192)

ApplyCodepage

ApplyCodepage() 在運算式中指明的欄位或文字上套用不同的代碼頁面字元設定。**codepage** 引數必須為數字格式。

> *雖然 ApplyCodepage 可用於圖表運算式，但更常用來作為資料載入編輯器中的指令碼函數。例如，當您載入儲存在超出您控制的不同字元集中的檔案，您可以套用代表您所需的字元集的字碼頁。*

語法：

```
ApplyCodepage(text, codepage)
```

傳回的資料類型： 字串

引數：

引數

引數	描述
text	您想要套用不同代碼頁面的欄位或文字，由引數 **codepage** 提供。
codepage	代表要套用至欄位或運算式之代碼頁面的數字，由 **text** 提供。

範例與結果:

指令碼處理範例

範例	結果
`LOAD ApplyCodepage(ROWX,1253) as GreekProduct, ApplyCodepage (ROWY, 1255) as HebrewProduct, ApplyCodepage (ROWZ, 65001) as EnglishProduct; SQL SELECT ROWX, ROWY, ROWZ From Products;`	從 SQL 載入時,來源可能會有不同字元設定的混合內容:Cyrillic、Hebrew 等,來自 UTF-8 格式。要逐行載入並為每行套用不同的字碼頁,這些可能是必需的。 **codepage** 值 1253 代表 Windows Greek 字元設定,值 1255 代表 Hebrew,而值 65001 代表標準 Latin UTF-8 字元。

另請參見: *字元集 (page 157)*

Date

Date() 將運算式的格式設定為日期,使用在資料載入指令碼或作業系統或格式字串 (如果提供的話) 的系統變數中設定的格式。

語法:

```
Date(number[, format])
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
number	要格式設定的數字。
format	描述所產生字串格式的字串。如果未提供任何格式字串,將會使用資料載入指令碼的系統變數或作業系統中設定的日期格式。

範例與結果:

以下的範例假設下列預設設定:

- 日期設定 1: YY-MM-DD
- 日期設定 2: M/D/YY

範例:

```
Date( A )
```

其中 A=35648

結果表格

結果	設定 1	設定 2
字串:	97-08-06	8/6/97
數字:	35648	35648

範例:

```
Date( A, 'YY.MM.DD' )
```
其中 A=35648

結果表格

結果	設定 1	設定 2
字串:	97.08.06	97.08.06
數字:	35648	35648

範例:

```
Date( A, 'DD.MM.YYYY' )
```
其中 A=35648.375

結果表格

結果	設定 1	設定 2
字串:	06.08.1997	06.08.1997
數字:	35648.375	35648.375

範例:

```
Date( A, 'YY.MM.DD' )
```
其中 A=8/6/97

結果表格

結果	設定 1	設定 2
字串:	NULL (無任何項目)	97.08.06
數字:	NULL	35648

Dual

Dual() 將一個數字與一個字串組合至單一記錄, 例如記錄的數字表示法可用於排序及計算, 而字串值可用於顯示。

語法:

```
Dual(text, number)
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
text	要與數字引數組合使用的字串值。
number	要與字串引數中字串組合使用的數字。

在 Qlik Sense 中，所有欄位值都可能是雙值。這表示欄位值可以同時具有數值和文字值。範例是可以具有數值 40908 和文字表示法 '2011-12-31' 的日期。

讀入同一個欄位的數個資料項目有不同的字串表示法，但是卻有相同的有效數字表示法時，則這些資料項目將共用第一個出現的字串表示法。

dual *函數一般用於指令碼前端，並會在其他資料讀入相關的欄位之前使用，以便建立將在篩選窗格之中顯示的第一個字串表示法。*

範例與結果：

指令碼處理範例

範例	描述

將下列範例新增至您的指令碼并予以執行。
`Load dual ( NameDay,NumDay ) as DayOfWeek inline`
`[ NameDay,NumDay`
`Monday,0`
`Tuesday,1`
`Wednesday,2`
`Thursday,3`
`Friday,4`
`Saturday,5`
`Sunday,6 ];`
	欄位 DayOfWeek 可以在視覺化中使用，例如作為維度。在具有週的表格中，日子自動排序為其正確的數字順序，而不使用字母順序。

範例	描述
`Load Dual('Q' & Ceil(Month (Now())/3), Ceil(Month(Now ())/3)) as Quarter AutoGenerate 1;`	此範例會尋找目前季度。當 **Now()** 函數在一年中的前三個月中執行時，它會顯示為 Q1，針對第二個三個月時，會顯示為 Q2，依此類推。然而，當用於排序中時，欄位 Quarter 將用作數值：1 至 4。
`Dual('Q' & Ceil(Month (Date)/3), Ceil(Month (Date)/3)) as Quarter`	如前一個範例所示，建立欄位 Quarter，使用文字值 'Q1' 至 'Q4'，并指派數值 1 至 4。為了在指令碼中使用此項目，必須載入 Date 的值。
`Dual(WeekYear(Date) & '-W' & Week(Date), WeekStart(Date)) as YearWeek`	此範例建立欄位 YearWeek，文字值來自表單 '2012-W22'，同時，指派對應於週第一天日期數的數值，例如：41057. 為了在指令碼中使用此項目，必須載入 Date 的值。

Interval

Interval() 將數字的格式設定為時間間隔，使用在資料載入指令碼或作業系統或格式字串 (如果提供的話) 的系統變數中設定的格式。

可以將間隔設定為時間、日期，或日期、小時、分鐘、秒和秒分數的組合。

語法：

```
Interval(number[, format])
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
number	要格式設定的數字。
format	說明如何對所產生間隔字串進行格式設定的字串。如果省略，則會使用作業系統中設定的簡短日期格式、時間格式和小數點分隔符號。

範例與結果：
以下的範例假設下列預設設定：

- 日期格式設定 1： YY-MM-DD
- 日期格式設定 2： hh:mm:ss
- 數字小數點分隔符號：

結果表格

範例	字串	數字
`Interval( A )` , 其中 A=0.375	09:00:00	0.375
`Interval( A )` , 其中 A=1.375	33:00:00	1.375
`Interval( A, 'D hh:mm' )` , 其中 A=1.375	1 09:00	1.375
`Interval( A-B, 'D hh:mm' )` , 其中 A=97-08-06 09:00:00 且 B=96-08-06 00:00:00	365 09:00	365.375

Money

Money() 將運算式的格式以數值方式設定為金額值, 使用在資料載入指令碼或作業系統 (除非提供格式字串) 的系統變數集中設定的格式, 以及選用的小數與千位分隔符號。

語法:

```
Money(number[, format[, dec_sep[, thou_sep]]])
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
number	要格式設定的數字。
format	說明如何對所產生貨幣字串進行格式設定的字串。
dec_sep	指定小數點位數分隔符號的字串。
thou_sep	指定千位分隔符號的字串。

如果省略引數 2-4, 將會使用作業系統中設定的貨幣格式。

範例與結果:
以下的範例假設下列預設設定:

- MoneyFormat 設定 1: kr ##0,00, MoneyThousandSep' '
- MoneyFormat 設定 2: $ #,##0.00, MoneyThousandSep','

範例:

`Money( A )`
, 其中 A=35648

結果表格

結果	設定 1	設定 2
字串：	kr 35 648,00	$ 35,648.00
數字：	35648.00	35648.00

範例：

```
Money( A, '#,##0 ¥', '.' , ',' )
```

，其中 A=3564800

結果表格

結果	設定 1	設定 2
字串：	3,564,800 ¥	3,564,800 ¥
數字：	3564800	3564800

Num

Num() 格式化數字，亦即使用第二參數指定的格式將輸入的數值轉換為顯示文字。若省略第二參數，這會使用資料載入指令碼中設定的小數點和千位分隔符號。自訂小數與千位分隔符號是選用參數。

語法：

```
Num(number[, format[, dec_sep [, thou_sep]]])
```

傳回的資料類型： 雙值

此 Num 函數會傳回含有字串和數值的雙值。這些函數採用輸入運算式的數值，並產生代表數字的字串。

引數：

引數

引數	描述
number	要格式設定的數字。
format	指定如何對所產生字串格式化的字串。如果省略，會使用資料載入指令碼中設定的小數點和千位分隔符號。
dec_sep	指定小數點位數分隔符號的字串。如果省略，則會使用資料載入指令碼中設定的變數 DecimalSep 的值。
thou_sep	指定千位分隔符號的字串。如果省略，則會使用資料載入指令碼中設定的變數 ThousandSep 的值。

範例:圖表運算式

範例:

下表顯示當欄位 A 等於 35648.312 時的結果。

結果

A	**結果**
Num(A)	35648.312 (根據指令碼中的環境變數)
Num(A, '0.0', '.')	35648.3
Num(A, '0,00', ',')	35648,31
Num(A, '#,##0.0', '.', ',')	35,648.3
Num(A, '# ##0', ',', ' ')	35 648

範例:載入指令碼

載入指令碼

Num 可用於載入指令碼,以格式化數字,即使指令碼中已設定千位和小數分隔符號。以下的載入指令碼包括特定的千位和小數分隔符號,但使用 *Num* 以不同的方式格式化資料。

在**資料載入編輯器**中,建立新的區段,然後新增範例指令碼並執行。然後,至少將結果資料行中列出的欄位新增至您應用程式中的工作表以查看結果。

```
SET ThousandSep=',';
SET DecimalSep='.';
Transactions:
Load
*,
Num(transaction_amount) as [No formatting],
Num(transaction_amount,'0') as [0],
Num(transaction_amount,'#,##0') as [#,##0],
Num(transaction_amount,'# ###,00') as [# ###,00],
Num(transaction_amount,'# ###,00',',',' ') as [# ###,00 , ',' , ' '],
Num(transaction_amount,'#,###.00','.',',') as [#,###.00 , '.' , ','],
Num(transaction_amount,'$#,###.00') as [$#,###.00],
;
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, discount,
customer_id, size, color_code
3750, 20180830, 12423.56, 23, 0,2038593, L, Red
3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange
3752, 20180916, 15.75, 1, 0.22, 5646471, S, blue
3753, 20180922, 1251, 7, 0, 3036491, l, Black
3754, 20180922, 21484.21, 1356, 75, 049681, xs, Red
3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue
3757, 20180923, 3177.4, 21, .14, 203521, XL, Black
];
```

Qlik Sense 表格顯示載入指令碼中 *Num* 函數的不同使用方式的結果。表格的第四欄包含不正確的格式使用，例如用途。

No formatting	0	#,##0	# ###,00	# ###,00 , ',' , ' '	#,###.00 , '.' , ','	$#,###.00
-59.18	-59	-59	-59###,00	-59,18	-59.18	$-59,18
15.75	16	16	16###,00	15,75	15.75	$15,75
1251	1251	1,251	1251###,00	1 251,00	1,251.00	$1,251.00
3177.4	3177	3,177	3177###,00	3 177,40	3,177.40	$3,177.40
5356.31	5356	5,356	5356###,00	5 356,31	5,356.31	$5,356.31
12423.56	12424	12,424	12424###,00	12 423,56	12,423.56	$12,423.56
21484.21	21484	21,484	21484###,00	21 484,21	21,484.21	$21,484.21

範例：載入指令碼

載入指令碼

Num 可用於載入指令碼，以百分比格式化數字。

在**資料載入編輯器**中，建立新的區段，然後新增範例指令碼並執行。然後，至少將結果資料行中列出的欄位新增至您應用程式中的工作表以查看結果。

```
SET ThousandSep=',';
SET DecimalSep='.';
Transactions:
Load
*,
Num(discount,'#,##0%') as [Discount #,##0%]
;
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, discount,
customer_id, size, color_code
3750, 20180830, 12423.56, 23, 0,2038593, L, Red
3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange
3752, 20180916, 15.75, 1, 0.22, 5646471, S, blue
3753, 20180922, 1251, 7, 0, 3036491, l, Black
3754, 20180922, 21484.21, 1356, 75, 049681, xs, Red
3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue
3757, 20180923, 3177.4, 21, .14, 203521, XL, Black
];
```

Qlik Sense 表格顯示載入指令碼中使用的 *Num* 函數結果，以格式化百分比。

Discount	Discount #,##0%
0.333333333333333	33%
0.22	22%

Discount	Discount #,##0%
0	0%
.14	14%
0.1	10%
0	0%
75	7,500%

Time

Time() 將運算式的格式設定為時間值，使用在資料載入指令碼或作業系統 (除非提供格式字串) 的系統變數中設定的時間格式。

語法：

```
Time(number[, format])
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
number	要格式設定的數字。
format	說明如何對所產生時間字串進行格式設定的字串。如果省略，則會使用作業系統中設定的簡短日期格式、時間格式和小數點分隔符號。

範例與結果：
以下的範例假設下列預設設定：

- 時間格式設定 1：hh:mm:ss
- 時間格式設定 2：hh.mm.ss

範例：

`Time( A )`
，其中 A=0.375

結果表格

結果	設定 1	設定 2
字串：	09:00:00	09.00.00
數字：	0.375	0.375

範例:

`Time( A )`
, 其中 A=35648.375

結果表格

結果	設定 1	設定 2
字串:	09:00:00	09.00.00
數字:	35648.375	35648.375

範例:

`Time( A, 'hh-mm' )`
, 其中 A=0.99999

結果表格

結果	設定 1	設定 2
字串:	23-59	23-59
數字:	0.99999	0.99999

Timestamp

TimeStamp() 將運算式的格式設定為日期和時間值, 使用在資料載入指令碼或作業系統 (除非提供格式字串) 的系統變數中設定的時間戳記格式。

語法:

```
Timestamp(number[, format])
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
number	要格式設定的數字。
format	說明如何對所產生時間戳記字串進行格式設定的字串。如果省略, 則會使用作業系統中設定的簡短日期格式、時間格式和小數點分隔符號。

範例與結果:
以下的範例假設下列預設設定:

- 時間戳記格式設定 1：YY-MM-DD hh:mm:ss
- 時間戳記格式設定 2：M/D/YY hh:mm:ss

範例：

```
Timestamp( A )
```
，其中 A=35648.375

結果表格

結果	設定 1	設定 2
字串：	97-08-06 09:00:00	8/6/97 09:00:00
數字：	35648.375	35648.375

範例：

```
Timestamp( A,'YYYY-MM-DD hh.mm')
```
，其中 A=35648

結果表格

結果	設定 1	設定 2
字串：	1997-08-06 00.00	1997-08-06 00.00
數字：	35648	35648

8.13 一般數值函數

在這些一般數值函數中，引數是運算式，其中 **x** 應解譯為真實的值數字。所有函數皆可用於資料載入指令碼和圖表運算式。

一般數值函數概述

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

bitcount

BitCount() 傳回十進位數的等效二進數中有多少個位元設定為 1。即，該函數會傳回 **integer_number** 中的設定位元數，其中 **integer_number** 會解譯為帶正負號的 32 位元整數。

```
BitCount(integer_number)
```

div

Div() 傳回第一個引數除以第二個引數的整數部分。兩個參數都會解譯為實數，亦即不必為整數。

```
Div (integer_number1, integer_number2)
```

fabs

Fabs() 傳回 **x** 的絕對值。結果是正數。

```
Fabs (x)
```

fact

Fact() 傳回正整數 **x** 階乘。

```
Fact (x)
```

frac

Frac() 傳回 **x** 的分數部分。

```
Frac (x)
```

sign

視 **x** 為正數、0 或負數而定，**Sign()** 分別傳回 1、0 或 -1。

```
Sign (x)
```

組合與排列函數

combin

Combin() 傳回可以從一組 **p** 個項目中挑選 **q** 個元素的組合數目。表示方式如公式：`Combin(p,q) = p! / q!(p-q)!` 選取項目的順序很重要。

```
Combin (p, q)
```

permut

Permut() 傳回可以從一組 **p** 個項目中選取 **q** 個元素的排列數目。表示方式如公式：`Permut(p,q) = (p)! / (p - q)!` 選取項目的順序很重要。

```
Permut (p, q)
```

模數函數

fmod

fmod() 是一個廣義模組函數，傳回整數除法中第一個引數 (被除數) 除以第二個引數 (除數) 的餘數部分。結果是實數。兩個引數都會解譯為實數，亦即不必為整數。

```
Fmod (a, b)
```

mod

Mod() 是一個數學模組函數，傳回整數除法的非負餘數。第一個引數是被除數，第二個引數是除數，兩個引數都必須是整數值。

```
Mod (integer_number1, integer_number2)
```

同位函數

even

如果 **integer_number** 是偶數整數或零，則 **Even()** 傳回 True (-1)。如果 **integer_number** 是奇數整數及 NULL，並且如果 **integer_number** 不是整數，則它傳回 False (0)。

```
Even (integer_number)
```

odd

如果 **integer_number** 是奇數整數或零，則 **Odd()** 傳回 True (-1)。如果 **integer_number** 偶整數整數及 NULL，並且如果 **integer_number** 不是整數，則它傳回 False (0)。

```
Odd (integer_number)
```

捨入函數

ceil

Ceil() 會將數字向上捨入為 **offset** 數字偏移之 **step** 的最近倍數。

```
Ceil (x[, step[, offset]])
```

floor

Floor() 會將數字向下捨入為 **offset** 數字偏移之 **step** 的最近倍數。

```
Floor (x[, step[, offset]])
```

round

Round() 傳回會將數字向上或向下捨入為 **offset** 數字偏移之 **step** 的最近倍數的結果。

```
Round ( x [ , step [ , offset ]])
```

BitCount

BitCount() 傳回十進位數的等效二進數中有多少個位元設定為 1。即，該函數會傳回 **integer_number** 中的設定位元數，其中 **integer_number** 會解譯為帶正負號的 32 位元整數。

語法：

```
BitCount(integer_number)
```

傳回的資料類型： 整數

範例與結果：

範例與結果

範例	結果
`BitCount ( 3 )`	3 的二進位為 11，因此這個函數會傳回 2
`BitCount ( -1 )`	-1 的二進位為 64 個 1，因此這個函數會傳回 64

Ceil

Ceil() 會將數字向上捨入為 **offset** 數字偏移之 **step** 的最近倍數。

相較於向下捨入輸入數字的 **floor** 函數。

語法：

```
Ceil(x[, step[, offset]])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
x	輸入數字。
step	間隔增量。預設值為 1。
offset	定義步階間隔的基數。預設值為 0。

範例與結果:

範例與結果

範例	結果
`Ceil(2.4 )`	傳回 3 在此範例中，步階的大小為 1，步階間隔的基數為 0。 間隔為 ...0 < x <=1, 1 < x <= 2, **2< x <=3**, 3< x <=4...
`Ceil(4.2 )`	傳回 5
`Ceil(3.88 ,0.1)`	傳回 3.9 在此範例中，間隔的大小為 0.1，間隔的基數為 0。 間隔為 ... 3.7 < x <= 3.8, **3.8 < x <= 3.9**, 3.9 < x <= 4.0...
`Ceil(3.88 ,5)`	傳回 5
`Ceil(1.1 ,1)`	傳回 2
`Ceil(1.1 ,1,0.5)`	傳回 1.5 在此範例中，步階的大小為 1，偏移值為 0.5。這表示步階間隔的基數為 0.5 而非 0。 間隔為 ...**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)`	傳回 1.99 間隔為 ...-0.01< x <= 0.99, **0.99< x <= 1.99**, 1.99 < x <=2.99...

Combin

Combin() 傳回可以從一組 **p** 個項目中挑選 **q** 個元素的組合數目。表示方式如公式：`Combin(p,q) = p! / q!(p-q)!` 選取項目的順序很重要。

語法:

```
Combin(p, q)
```

傳回的資料類型: 整數

限制:

非整數項目會被截斷。

範例與結果:

範例與結果

範例	**結果**
從總共 35 個樂透數字中挑選 7 個數字的可能組合有幾個? `Combin( 35,7 )`	傳回 6,724,520

Div

Div() 傳回第一個引數除以第二個引數的整數部分。兩個參數都會解譯為實數, 亦即不必為整數。

語法:

```
Div(integer_number1, integer_number2)
```

傳回的資料類型: 整數

範例與結果:

範例與結果

範例	**結果**
`Div( 7,2 )`	傳回 3
`Div( 7.1,2.3 )`	傳回 3
`Div( 9,3 )`	傳回 3
`Div( -4,3 )`	傳回 -1
`Div( 4,-3 )`	傳回 -1
`Div( -4,-3 )`	傳回 1

Even

如果 **integer_number** 是偶數整數或零, 則 **Even()** 傳回 True (-1)。如果 **integer_number** 是奇數整數及 NULL, 並且如果 **integer_number** 不是整數, 則它傳回 False (0)。

語法:

```
Even(integer_number)
```

傳回的資料類型: 布林

範例與結果:

範例與結果

範例	結果
`Even( 3 )`	傳回 0, False
`Even( 2 * 10 )`	傳回 -1, True
`Even( 3.14 )`	傳回 NULL

Fabs

Fabs() 傳回 **x** 的絕對值。結果是正數。

語法:

```
fabs(x)
```

傳回的資料類型: 數值

範例與結果:

範例與結果

範例	結果
`fabs( 2.4 )`	傳回 2.4
`fabs( -3.8 )`	傳回 3.8

Fact

Fact() 傳回正整數 **x** 階乘。

語法:

```
Fact(x)
```

傳回的資料類型: 整數

限制:

若 **x** 不是整數, 就會遭截斷。非正數會傳回 NULL。

範例與結果：

範例與結果

範例	結果
`Fact( 1 )`	傳回 1
`Fact( 5 )`	傳回 120 (1 * 2 * 3 * 4 * 5 = 120)
`Fact( -5 )`	傳回 NULL

Floor

Floor() 會將數字向下捨入為 **offset** 數字偏移之 **step** 的最近倍數。

相較於向上捨入輸入數字的 **ceil** 函數。

語法：

```
Floor(x[, step[, offset]])
```

傳回的資料類型： 數值

引數：

引數

引數	描述
x	輸入數字。
step	間隔增量。預設值為 1。
offset	定義步階間隔的基數。預設值為 0。

範例與結果：

範例與結果

範例	結果
`Floor(2.4)`	傳回 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)`	傳回 4
`Floor(3.88 ,0.1)`	傳回 3.8 在此範例中，間隔的大小為 0.1，間隔的基數為 0。 間隔為 ... 3.7 <= x < 3.8, **3.8 <= x < 3.9**, 3.9 <= x < 4.0...

範例	結果
`Floor(3.88 ,5)`	傳回 0
`Floor(1.1 ,1)`	傳回 1
`Floor(1.1 ,1,0.5)`	傳回 0.5 在此範例中，步階的大小為 1，偏移值為 0.5。這表示步階間隔的基數為 0.5 而非 0。 間隔為 ...**0.5 <= x <1.5**, 1.5 <= x < 2.5, 2.5<= x <3.5,...

Fmod

fmod() 是一個廣義模組函數，傳回整數除法中第一個引數 (被除數) 除以第二個引數 (除數) 的餘數部分。結果是實數。兩個引數都會解譯為實數，亦即不必為整數。

語法：

```
fmod(a, b)
```

傳回的資料類型： 數值

引數：

引數

引數	描述
a	被除數
b	除數

範例與結果：

範例與結果

範例	結果
`fmod( 7,2 )`	傳回 1
`fmod( 7.5,2 )`	傳回 1.5
`fmod( 9,3 )`	傳回 0
`fmod( -4,3 )`	傳回 -1
`fmod( 4,-3 )`	傳回 1
`fmod( -4,-3 )`	傳回 -1

Frac

Frac() 傳回 **x** 的分數部分。

小數的定義方式為 `Frac(x ) + Floor(x ) = x`。簡單來說，這表示一個正數的小數部分就是數字 (x) 和緊接在小數部分前整數之間的差異。

例如：11.43 的小數部分 = 11.43 - 11 = 0.43

若為負數，例如 -1.4，Floor(-1.4) = -2，這會產生下列結果：

-1.4 的小數部份 = 1.4 - (-2) = -1.4 + 2 = 0.6

語法：

```
Frac(x)
```

傳回的資料類型： 數值

引數：

引數

引數	描述
x	要為其傳回分數的數字。

範例與結果：

範例與結果

範例	結果
`Frac( 11.43 )`	傳回 0.43
`Frac( -1.4 )`	傳回 0.6
從時間戳記的數值呈現擷取時間元件，因此省略日期。 `Time(Frac(44518.663888889))`	傳回 3:56:00 PM

Mod

Mod() 是一個數學模組函數，傳回整數除法的非負餘數。第一個引數是被除數，第二個引數是除數，兩個引數都必須是整數值。

語法：

```
Mod(integer_number1, integer_number2)
```

傳回的資料類型： 整數

限制：

integer_number2 必須大於 0。

範例與結果:

範例與結果

範例	**結果**
`Mod( 7,2 )`	傳回 1
`Mod( 7.5,2 )`	傳回 NULL
`Mod( 9,3 )`	傳回 0
`Mod( -4,3 )`	傳回 2
`Mod( 4,-3 )`	傳回 NULL
`Mod( -4,-3 )`	傳回 NULL

Odd

如果 **integer_number** 是奇數整數或零，則 **Odd()** 傳回 True (-1)。如果 **integer_number** 偶整數整數及 NULL，並且如果 **integer_number** 不是整數，則它傳回 False (0)。

語法:

```
Odd(integer_number)
```

傳回的資料類型: 布林

範例與結果:

範例與結果

範例	**結果**
`Odd( 3 )`	傳回 -1, True
`Odd( 2 * 10 )`	傳回 0, False
`Odd( 3.14 )`	傳回 NULL

Permut

Permut() 傳回可以從一組 **p** 個項目中選取 **q** 個元素的排列數目。表示方式如公式:`Permut(p,q) = (p)! / (p - q)!` 選取項目的順序很重要。

語法:

```
Permut(p, q)
```

傳回的資料類型: 整數

限制:

非整數引數會被截斷。

範例與結果:

範例與結果

範例	結果
在有 8 位參賽者的 100 公尺決賽中,頒發金、銀、銅牌的可能結果有幾種? `Permut( 8,3 )`	傳回 336

Round

Round() 傳回會將數字向上或向下捨入為 **offset** 數字偏移之 **step** 的最近倍數的結果。

若待四捨五入的數字位於間隔的正中間,則會向上捨入。

語法:

```
Round(x[, step[, offset]])
```

傳回的資料類型: 數值

如果您對一個浮點數四捨五入,可能會觀察到錯誤結果。這些四捨五入錯誤是因為浮點數以二進位的有限數表示而導致。因此,結果是使用已經四捨五入的數字計算而得。如果這些四捨五入錯誤會影響您的工作,四捨五入之前,請將這些數字相乘以將它們轉換成整數。

引數:

引數

引數	描述
x	輸入數字。
step	間隔增量。預設值為 1。
offset	定義步階間隔的基數。預設值為 0。

範例與結果:

範例與結果

範例	結果
`Round(3.8 )`	傳回 4 在此範例中,步階的大小為 1,步階間隔的基數為 0。 間隔為 ...0 <= x <1, 1 <= x < 2, 2<= x <3, **3<= x <4**...
`Round(3.8,4 )`	傳回 4

範例	結果
`Round(2.5 )`	傳回 3。 在此範例中，步階的大小為 1，步階間隔的基數為 0。 間隔值為 ...0 <= x <1, 1 <= x <2, **2<= x <3**...
`Round(2,4 )`	傳回 4。向上捨入，因為 2 剛好是步階間隔 4 的一半。 在此範例中，步階的大小為 4，步階間隔的基數為 0。 間隔值為 ...**0 <= x <4**, 4 <= x <8, 8<= x <12...
`Round(2,6 )`	傳回 0。向下捨入，因為 2 小於步階間隔 6 的一半。 在此範例中，步階的大小為 6，步階間隔的基數為 0。 間隔值為 ...**0 <= x <6**, 6 <= x <12, 12<= x <18...
`Round(3.88 ,0.1)`	傳回 3.9 在此範例中，步階的大小為 0.1，步階間隔的基數為 0。 間隔為 ... 3.7 <= x <3.8, **3.8 <= x <3.9**, 3.9 <= x < 4.0...
`Round(3.88875,1/1000)`	傳回 3.889 在此範例中，步驟大小是 0.001，這捨進數字並限制為三個小數位數。
`Round(3.88 ,5)`	傳回 5
`Round(1.1 ,1,0.5)`	傳回 1.5 在此範例中，步階的大小為 1，步階間隔的基數為 0.5。 間隔為 ...**0.5 <= x <1.5**, 1.5 <= x <2.5, 2.5<= x <3.5...

Sign

視 **x** 為正數、0 或負數而定，**Sign()** 分別傳回 1、0 或 -1。

語法：

```
Sign(x)
```

傳回的資料類型： 數值

限制：

如果找不到數值，則會傳回 NULL。

範例與結果:

範例與結果

範例	**結果**
`Sign( 66 )`	傳回 1
`Sign( 0 )`	傳回 0
`Sign( - 234 )`	傳回 -1

8.14 地理空間函數

這些函數用於在地圖視覺化中處理地理空間資料。Qlik Sense 按照地理空間資料的 GeoJSON 規格並支援下列內容:

- 點
- 線串
- 多邊形
- 多個多邊形

如需更多關於 GeoJSON 規格的資訊，請參閱:

GeoJSON.org

地理空間函數概述

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

有兩種類型的地理空間函數:彙總和非彙總。

彙總函數以幾何值集 (點或區域) 作為輸入值，並傳回單一幾何值。例如，可以合併多個區域，可以在地圖上繪製彙總的單一邊界。

非彙總函數採用一個單一幾何值，並傳回一個幾何值。例如，對於函數 GeoGetPolygonCenter()，如果將某個區域的邊界幾何值設為輸入值，將傳回該區域中心的點幾何值 (經度和緯度)。

以下為彙總函數:

GeoAggrGeometry

GeoAggrGeometry() 用來將許多區域彙總至較大區域，例如，將許多子區域彙總至某個區域。

```
GeoAggrGeometry (field_name)
```

GeoBoundingBox

GeoBoundingBox() 用來將幾何圖形彙總至某個區域，並計算包含所有座標的最小邊界框。

```
GeoBoundingBox (field_name)
```

GeoCountVertex

GeoCountVertex() 用來尋找多邊幾何圖形包含的頂點數目。

```
GeoCountVertex(field_name)
```

GeoInvProjectGeometry

GeoInvProjectGeometry() 用來將幾何圖形彙總至某個區域，並套用投射的反向。

```
GeoInvProjectGeometry(type, field_name)
```

GeoProjectGeometry

GeoProjectGeometry() 用來將幾何圖形彙總至某個區域，並套用投射。

```
GeoProjectGeometry(type, field_name)
```

GeoReduceGeometry

GeoReduceGeometry() 用來減少幾何圖形的頂點數目，並將區域數目彙總至一個區域，但是仍顯示個別區域的邊界線。

```
GeoReduceGeometry (geometry)
```

以下為非彙總函數：

GeoGetBoundingBox

GeoGetBoundingBox() 用在指令碼與圖表運算式中，可計算包含幾何圖形所有座標的最小地域邊界框。

```
GeoGetBoundingBox (geometry)
```

GeoGetPolygonCenter

GeoGetPolygonCenter() 用在指令碼與圖表運算式中，可計算及傳回幾何圖形的中心點。

```
GeoGetPolygonCenter (geometry)
```

GeoMakePoint

GeoMakePoint() 用在指令碼及圖表運算式中，以在緯度和經度中建立一個點並予以標記。

```
GeoMakePoint (lat_field_name, lon_field_name)
```

GeoProject

GeoProject() 用在指令碼與圖表運算式中，可將投射套用至幾何圖形。

```
GeoProject (type, field_name)
```

GeoAggrGeometry

GeoAggrGeometry() 用來將許多區域彙總至較大區域，例如，將許多子區域彙總至某個區域。

語法：

```
GeoAggrGeometry(field_name)
```

傳回的資料類型: 字串

引數:

引數

引數	描述
field_ name	欄位或運算式是指包含要表示之幾何圖形的欄位。這可以是提供經度和緯度的點 (或點集) 或區域。

通常, **GeoAggrGeometry()** 可用於合併地理空間邊界資料。例如, 您可能擁有某個城市郊區的郵遞區號區域以及每個區域的銷售收入資料。如果某位銷售人員的銷售地區覆蓋數個郵遞區號區域, 按銷售地區而不是單個區域展示總銷售額, 並在填充色彩的地區顯示結果, 這樣可能很有用。

GeoAggrGeometry() 可以計算個別郊區幾何值的彙總, 並在資料模型中產生合併的地區幾何值。如果隨後調整了銷售地區邊界, 當重新載入資料時, 地圖中將反映新的合併邊界和收入。

由於 **GeoAggrGeometry()** 是彙總函數, 如果在指令碼中使用, 則需要 **LOAD** 陳述式 (含 **Group by** 子句)。

*使用 **GeoAggrGeometry()** 建立的地圖邊界線是那些合併區域的邊界線。如要顯示預彙總區域的個別邊界線, 請使用 **GeoReduceGeometry()**。*

範例:
此範例會載入含區域資料的 KML 檔案, 然後載入含已彙總區域資料的表格。

```
[MapSource]: LOAD [world.Name], [world.Point], [world.Area] FROM [lib://Downloads/world.kml]
(kml, Table is [World.shp/Features]); Map: LOAD world.Name, GeoAggrGeometry(world.Area) as
[AggrArea] resident MapSource Group By world.Name;

Drop Table MapSource;
```

GeoBoundingBox

GeoBoundingBox() 用來將幾何圖形彙總至某個區域, 並計算包含所有座標的最小邊界框。

GeoBoundingBox 表示為一列四個值:左、右、上、下。

語法:

```
GeoBoundingBox(field_name)
```

傳回的資料類型: 字串

引數:

引數

引數	描述
field_ name	欄位或運算式是指包含要表示之幾何圖形的欄位。這可以是提供經度和緯度的點 (或點集) 或區域。

GeoBoundingBox() 可彙總一組幾何值, 並對包含所有彙總幾何值的座標的最小矩形傳回四個座標。

要在地圖上顯示結果, 將產生的四個座標字串轉成多邊形格式, 將轉換的欄位標記為地理多邊形格式, 並將該欄位拖放到地圖物件中。然後, 矩形方塊會在地圖視覺化中顯示。

GeoCountVertex

GeoCountVertex() 用來尋找多邊幾何圖形包含的頂點數目。

語法:

```
GeoCountVertex(field_name)
```

傳回的資料類型: 整數

引數:

引數

引數	描述
field_ name	欄位或運算式是指包含要表示之幾何圖形的欄位。這可以是提供經度和緯度的點 (或點集) 或區域。

GeoGetBoundingBox

GeoGetBoundingBox() 用在指令碼與圖表運算式中, 可計算包含幾何圖形所有座標的最小地域邊界框。

由函數 GeoBoundingBox() 建立的地理空間邊界框, 表示為一列四個值:左、右、上、下。

語法:

```
GeoGetBoundingBox(field_name)
```

傳回的資料類型: 字串

引數:

引數

引數	描述
field_name	欄位或運算式是指包含要表示之幾何圖形的欄位。這可以是提供經度和緯度的點 (或點集) 或區域。

*不要在資料載入編輯器中同時使用 **Group by** 子句和此函數以及其他非彙總地理空間函數, 因此這可能會引起載入錯誤。*

GeoGetPolygonCenter

GeoGetPolygonCenter() 用在指令碼與圖表運算式中, 可計算及傳回幾何圖形的中心點。

某些情況下, 要求在地圖上繪製一個點而不是填充色彩。如果現有地理空間資料僅在採用區域幾何值的形式 (例如, 邊界) 時才可用, 請使用 **GeoGetPolygonCenter()** 擷取區域中心的一對經緯度。

語法:

```
GeoGetPolygonCenter(field_name)
```

傳回的資料類型: 字串

引數:

引數

引數	描述
field_name	欄位或運算式是指包含要表示之幾何圖形的欄位。這可以是提供經度和緯度的點 (或點集) 或區域。

*不要在資料載入編輯器中同時使用 **Group by** 子句和此函數以及其他非彙總地理空間函數, 因此這可能會引起載入錯誤。*

GeoInvProjectGeometry

GeoInvProjectGeometry() 用來將幾何圖形彙總至某個區域, 並套用投射的反向。

語法:

```
GeoInvProjectGeometry(type, field_name)
```

傳回的資料類型： 字串

引數：

引數

引數	描述
type	轉換地圖的幾何值時使用的投射類型。這可能會取兩個值之一：'unit' (預設)，會產生 1:1 的投射，或 'mercator'，會使用標準 Mercator 投射。
field_ name	欄位或運算式是指包含要表示之幾何圖形的欄位。這可以是提供經度和緯度的點 (或點集) 或區域。

範例：

指令碼處理範例

範例	結果
在 `Load` 陳述式中： `GeoInvProjectGeometry ('mercator',AreaPolygon) as InvProjectGeometry`	將使用 Mercator 投射的反向轉換對載入為 **AreaPolygon** 的幾何值進行轉換，並儲存為 **InvProjectGeometry** 以供在視覺化中使用。

GeoMakePoint

GeoMakePoint() 用在指令碼及圖表運算式中，以在緯度和經度中建立一個點並予以標記。GeoMakePoint 以經度和緯度的順序傳回點。

語法：

```
GeoMakePoint(lat_field_name, lon_field_name)
```

傳回的資料類型： 字串，已設定格式 [經度，緯度]

引數：

引數

引數	描述
lat_field_name	欄位或運算式，一個代表點的緯度的欄位。
lon_field_name	欄位或運算式，一個代表點的經度的欄位。

*不要在資料載入編輯器中同時使用 **Group by** 子句和此函數以及其他非彙總地理空間函數，因此這可能會引起載入錯誤。*

GeoProject

GeoProject() 用在指令碼與圖表運算式中，可將投射套用至幾何圖形。

語法：

```
GeoProject(type, field_name)
```

傳回的資料類型： 字串

引數：

引數

引數	描述
type	轉換地圖的幾何值時使用的投射類型。這可能會取兩個值之一：'unit' (預設)，會產生 1:1 的投射；或 'mercator'，會使用 web Mercator 投射。
field_ name	欄位或運算式是指包含要表示之幾何圖形的欄位。這可以是提供經度和緯度的點 (或點集) 或區域。

不要在資料載入編輯器中同時使用 ***Group by*** *子句和此函數以及其他非彙總地理空間函數，因此這可能會引起載入錯誤。*

範例：

指令碼範例

範例	結果
在 Load 陳述式中： `GeoProject('mercator',Area) as GetProject`	Mercator 投射套用於載入為 **Area** 的幾何值，結果將儲存為 **GetProject**。

GeoProjectGeometry

GeoProjectGeometry() 用來將幾何圖形彙總至某個區域，並套用投射。

語法：

```
GeoProjectGeometry(type, field_name)
```

傳回的資料類型： 字串

引數：

引數

引數	描述
type	轉換地圖的幾何值時使用的投射類型。這可能會取兩個值之一：'unit' (預設)，會產生 1:1 的投射；或 'mercator'，會使用 web Mercator 投射。
field_ name	欄位或運算式是指包含要表示之幾何圖形的欄位。這可以是提供經度和緯度的點 (或點集) 或區域。

範例:

範例	結果
在 Load 陳述式中: `GeoProjectGeometry ('mercator',AreaPolygon) as ProjectGeometry`	將使用 Mercator 投射對載入為 **AreaPolygon** 的幾何值進行轉換, 並儲存為 **ProjectGeometry** 以供在視覺化中使用。

GeoReduceGeometry

GeoReduceGeometry() 用來減少幾何圖形的頂點數目, 並將區域數目彙總至一個區域, 但是仍顯示個別區域的邊界線。

語法:

```
GeoReduceGeometry(field_name[, value])
```

傳回的資料類型: 字串

引數:

引數

引數	描述
field_name	欄位或運算式是指包含要表示之幾何圖形的欄位。這可以是提供經度和緯度的點 (或點集) 或區域。
value	套用至幾何值的減少量。範圍為從 0 到 1, 其中 0 表示沒有減少, 1 表示頂點的最大減少量。 *若同時使用 0.9 或更高的 value 與複雜的資料集, 可能會將頂點的數量減少到視覺顯示失準的水平。*

GeoReduceGeometry() 還可執行一個與 **GeoAggrGeometry()** 相似的函數, 此函數可將多個區域彙總稱為一個區域。不同之處在於, 如果您使用 **GeoReduceGeometry()**, 地圖上會顯示來自預先彙總資料的個別邊界線。

由於 **GeoReduceGeometry()** 是彙總函數, 如果在指令碼中使用, 則需要 **LOAD** 陳述式 (含 **Group by** 子句)。

範例:
此範例會載入含區域資料的 KML 檔案, 然後載入含已減少和已彙總區域資料的表格。

```
[MapSource]:
LOAD [world.Name],
       [world.Point],
       [world.Area]
 FROM [lib://Downloads/world.kml]
(kml, Table is [World.shp/Features]);
```

```
Map:
LOAD world.Name,
     GeoReduceGeometry(world.Area,0.5) as [ReducedArea]
resident MapSource Group By world.Name;

Drop Table MapSource;
```

8.15 解譯函數

解譯函數會評估輸入文字欄位或運算式的內容，並在產生的數值上套用指定的日期格式。使用這些函數，您可以遵循相關資料類型指定數字的格式，其中包括諸入小數點分隔符號、千位分隔符號、日期格式等屬性。

解譯函數全部會傳回同時具有字串和數值，但是可以視為執行字串到數字轉換的雙值。這些函數採用輸入運算式的文字值，並產生代表字串的數字。

比較而言，格式設定函數正相反：它們採用數值運算式，將它們作為字串進行評估，指定所產生文字的顯示格式。

若未使用解譯函數，Qlik Sense 會使用由指令碼變數和作業系統所定義之數字格式、日期格式及時間格式的預設設定，以混合數字、日期、時間、時間戳記及字串的方式來解譯資料。

所有解譯函數皆可用於資料載入指令碼和圖表運算式。

所有的數字表示法都會加入小數點作為小數點分隔符號。

解譯函數概述

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

Date#

Date# 會使用第二個引數中提供的格式 (如果提供的話)，將運算式評估為日期。如果省略格式代碼，將會使用作業系統中設定的預設日期格式。

Date# (page 1193)(text[, format])

Interval#

Interval#() 將文字運算式作為時間間隔進行評估，依預設使用作業系統中設定的格式，或者使用第二個引數種指定的格式 (如果提供的話)。

Interval# (page 1194)(text[, format])

Money#

Money#() 將文字字串轉換為金額值，使用在載入指令碼或作業系統 (除非提供格式字串) 中設定的格式。自訂小數與千位分隔符號是選用參數。

Money# (page 1195)(text[, format[, dec_sep[, thou_sep]]])

Num#

Num#() 將文字字串解譯為數值，亦即使用第二參數指定的格式將輸入字串轉換為數字。若省略第二參數，這會使用資料載入指令碼中設定的小數點和千位分隔符號。自訂小數與千位分隔符號是選用參數。

```
Num# (page 1196)(text[ , format[, dec_sep[ , thou_sep]]])
```

Text

Text() 可強制將運算式視為文字，即使可能是數值解譯。

```
Text(expr)
```

Time#

Time#() 將運算式作為時間值進行評估，使用在資料載入指令碼或作業系統 (除非提供格式字串) 中設定的時間格式。.

```
Time# (page 1197)(text[, format])
```

Timestamp#

Timestamp#() 將運算式作為日期和時間值進行評估，使用在資料載入指令碼或作業系統 (除非提供格式字串) 中設定的時間戳記格式。

```
Timestamp# (page 1198)(text[, format])
```

另請參見：

格式設定函數 (page 1160)

Date#

Date# 會使用第二個引數中提供的格式 (如果提供的話)，將運算式評估為日期。

語法：

```
Date#(text[, format])
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
text	要評估的文字字串。
format	描述待評估的文字字串格式之字串。如果忽略，將會使用資料載入指令碼的系統變數或作業系統中設定的日期格式。

範例與結果：

下例使用日期格式 **M/D/YYYY**。日期格式在資料載入指令碼頂部的 **SET DateFormat** 陳述式中指定。

新增此範例指令碼到您的應用程式中並加以執行。

```
Load *,

Num(Date#(StringDate)) as Date;

LOAD * INLINE [

StringDate

8/7/97

8/6/1997

]
```

若您使用 **StringDate**和 **Date** 作為維度建立表格，結果將如下所示：

結果

StringDate	日期
8/7/97	35649
8/6/1997	35648

Interval#

Interval#() 將文字運算式作為時間間隔進行評估，依預設使用作業系統中設定的格式，或者使用第二個引數種指定的格式 (如果提供的話)。

語法：

```
Interval#(text[, format])
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
text	要評估的文字字串。
format	描述當將字串轉換成間隔數值時預期要使用的輸入格式的字串。 如果省略，則會使用作業系統中設定的簡短日期格式、時間格式和小數點分隔符號。

interval# 函數可將文字時間間隔轉換為數字時間間隔。

範例與結果：
以下的範例假設下列作業系統設定：

- 簡短日期格式：YY-MM-DD
- 時間格式：M/D/YY
- 數字小數點分隔符號：

結果

範例	結果
`Interval#( A, 'D hh:mm' )` 其中 A='1 09:00'	1.375

Money#

Money#() 將文字字串轉換為金額值，使用在載入指令碼或作業系統 (除非提供格式字串) 中設定的格式。自訂小數與千位分隔符號是選用參數。

語法：

```
Money#(text[, format[, dec_sep [, thou_sep ] ] ])
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
text	要評估的文字字串。
format	描述當將字串轉換成間隔數值時預期要使用的輸入格式的字串。 如果省略，則將會使用作業系統中設定的貨幣格式。
dec_sep	指定小數點位數分隔符號的字串。如果省略，則會使用資料載入指令碼中設定的 MoneyDecimalSep 值。
thou_sep	指定千位分隔符號的字串。如果省略，則會使用資料載入指令碼中設定的 MoneyThousandSep 值。

money# 函數的作用一般和 **num#** 函數相同，不過會從金額格式的指令碼變數或貨幣的系統設定取得小數點分隔符號和千位分隔符號的預設值。

範例與結果：
以下的範例假設下列兩個作業系統設定：

- 貨幣格式預設設定 1：kr # ##0,00
- 貨幣格式預設設定 2：$ #,##0.00

`Money#(A , '# ##0,00 kr' )`
其中 A=35 648,37 kr

結果

結果	設定 1	設定 2
字串	35 648.37 kr	35 648.37 kr
數字	35648.37	3564837

```
Money#( A, ' $#', '.', ',' )
```
其中 A= $35,648.37

結果

結果	設定 1	設定 2
字串	$35,648.37	$35,648.37
數字	35648.37	35648.37

Num#

Num#() 將文字字串解譯為數值，亦即使用第二參數指定的格式將輸入字串轉換為數字。若省略第二參數，這會使用資料載入指令碼中設定的小數點和千位分隔符號。自訂小數與千位分隔符號是選用參數。

語法：

```
Num#(text[, format[, dec_sep [, thou_sep ] ] ])
```

傳回的資料類型： 雙值

Num#() 函數會傳回含有字串和數值的雙值。該函數採用輸入運算式的文字表示，並產生數字。這不會變更數字格式：輸出的格式化方式與輸入相同。

引數：

引數

引數	描述
text	要評估的文字字串。
format	指示第一個參數使用的數字格式的字串。如果省略，會使用資料載入指令碼中設定的小數點和千位分隔符號。
dec_sep	指定小數點位數分隔符號的字串。如果省略，則會使用資料載入指令碼中設定的變數 DecimalSep 的值。
thou_sep	指定千位分隔符號的字串。如果省略，則會使用資料載入指令碼中設定的變數 ThousandSep 的值。

範例與結果：
下表針對不同的 A 值顯示 *Num#(A, '#', '.' , ',')* 的結果。

結果

A	字串表示法	數值 (此處以小數點顯示)
35,648.31	35,648.31	35648.31
35 648.312	35 648.312	35648.312
35.648,3123	35.648,3123	-
35 648,31234	35 648,31234	-

Text

Text() 可強制將運算式視為文字，即使可能是數值解譯。

語法：

```
Text (expr)
```

傳回的資料類型： 雙值

範例：

```
Text( A )
```

，其中 A=1234

結果

字串	數字
1234	-

範例：

```
Text( pi( ) )
```

結果

字串	數字
3.1415926535898	-

Time#

Time#() 將運算式作為時間值進行評估，使用在資料載入指令碼或作業系統 (除非提供格式字串) 中設定的時間格式。.

語法：

```
time#(text[, format])
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
text	要評估的文字字串。
format	描述待評估的文字字串格式之字串。如果省略,則會使用作業系統中設定的簡短日期格式、時間格式和小數點分隔符號。

範例:

- 時間格式預設設定 1: hh:mm:ss
- 時間格式預設設定 2: hh.mm.ss

`time#( A )`
其中 A=09:00:00

結果

結果	設定 1	設定 2
字串:	09:00:00	09:00:00
數字:	0.375	-

範例:

- 時間格式預設設定 1: hh:mm:ss
- 時間格式預設設定 2: hh.mm.ss

`time#( A, 'hh.mm' )`
, 其中 A=09.00

結果

結果	設定 1	設定 2
字串:	09.00	09.00
數字:	0.375	0.375

Timestamp#

Timestamp#() 將運算式作為日期和時間值進行評估,使用在資料載入指令碼或作業系統 (除非提供格式字串) 中設定的時間戳記格式。

語法:

```
timestamp#(text[, format])
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
text	要評估的文字字串。
format	描述待評估的文字字串格式之字串。如果省略,則會使用作業系統中設定的簡短日期格式、時間格式和小數點分隔符號。ISO 8601 支援時間戳記。

範例:

下例使用日期格式 **M/D/YYYY**。日期格式在資料載入指令碼頂部的 **SET DateFormat** 陳述式中指定。

新增此範例指令碼到您的應用程式中並加以執行。

```
Load *,
Timestamp(Timestamp#(String)) as TS;
LOAD * INLINE [
String
2015-09-15T12:13:14
1952-10-16T13:14:00+0200
1109-03-01T14:15
];
```

若您使用 **String** 和 **TS** 作為維度建立表格,結果將如下所示:

結果

字串	TS
2015-09-15T12:13:14	9/15/2015 12:13:14 PM
1952-10-16T13:14:00+0200	10/16/1952 11:14:00 AM
1109-03-01T14:15	3/1/1109 2:15:00 PM

8.16 記錄間函數

使用記錄間函數的時機:

- 在資料載入指令碼中,當評估目前記錄需要先前載入的資料記錄時。
- 在圖表運算式中,當需要來自視覺化資料集的其他值時。

當任何圖表運算式使用了記錄間圖表函數時,就不允許在圖表中依據 Y 值排序,或在表格中依據運算式資料行排序。因此,這些排序替代選項會自動停用。若您在視覺化或表格中使用記錄間圖表函數,視覺化的排序將還原為對記錄間函數的排序輸入。此限制不適用於對等指令碼函數 (如有)。

自行參考運算式定義只能在少於 100 列的表格中進行，但是這將會隨 Qlik 引擎執行的硬體而異。

列函數

這些函數僅可用於圖表運算式。

Above

Above() 會在表格中某個資料行區段內的目前列上方，在某個列中評估運算式。用來計算的列取決於 **offset** 的值，如果呈現，預設情況下是正上方的列。對於表格以外的圖表，**Above()** 會評估圖表連續表同等表格中目前列上方的列。

```
Above - 圖表函數([TOTAL [<fld{,fld}>]] expr [ , offset [,count]])
```

Below

Below() 會在表格中某個資料行區段內的目前列下方，在某個列中評估運算式。用來計算的列取決於 **offset** 的值，如果呈現，預設情況下是正下方的列。對於表格以外的圖表，**Below()** 會評估圖表連續表同等表格中目前資料行下方的列。

```
Below - 圖表函數([TOTAL[<fld{,fld}>]] expression [ , offset [,count ]])
```

Bottom

Bottom() 會在表格中某個資料行區段內的最後一 (底端) 列，評估運算式。用來計算的列取決於 **offset** 的值，如果呈現，預設情況下是底端的列。對於表格以外的圖表，該運算式會評估圖表連續表同等表格中目前資料行的最後一列。

```
Bottom - 圖表函數([TOTAL[<fld{,fld}>]] expr [ , offset [,count ]])
```

Top

Top() 會在表格中某個資料行區段內的第一 (頂端) 列，評估運算式。用來計算的列取決於 **offset** 的值，如果呈現，預設情況下是頂端的列。對於表格以外的圖表，**Top()** 會評估圖表連續表同等表格中目前資料行的第一列。

```
Top - 圖表函數([TOTAL [<fld{,fld}>]] expr [ , offset [,count ]])
```

NoOfRows

NoOfRows() 傳回表格中目前資料行區段中的列數。對於點陣圖圖表，**NoOfRows()** 傳回圖表的連續表同等表格中的列數。

```
NoOfRows - 圖表函數([TOTAL])
```

資料行函數

這些函數僅可用於圖表運算式。

Column

Column() 會忽略維度，傳回在對應於連續表中 **ColumnNo** 的資料行中找到的值。例如，**Column(2)** 傳回第二個量值資料行的值。

```
Column - 圖表函數(ColumnNo)
```

Dimensionality

Dimensionality() 傳回目前列的維度數目。如果是樞紐分析表格，則該函數傳回有非彙總內容 (亦即不包含部分加總或摺疊彙總) 之維度資料行的總數目。

```
Dimensionality - 圖表函數 ( )
```

Secondarydimensionality

SecondaryDimensionality() 傳回有非彙總內容 (亦即不包含部分加總或摺疊彙總) 之維度樞紐分析表列的數目。此函數相當於針對水平樞紐分析表維度的 **dimensionality()** 函數。

```
SecondaryDimensionality- 圖表函數 ( )
```

欄位函數

FieldIndex

FieldIndex() 傳回在欄位 **field_name** 中欄位值 **value** 的位置 (依據載入順序)。

```
FieldIndex (field_name , value)
```

FieldValue

FieldValue() 傳回在欄位 **field_name** 的 **elem_no** 位置中找到的欄位值 (依據載入順序)。

```
FieldValue (field_name , elem_no)
```

FieldValueCount

FieldValueCount() 是一個 **integer** 函數，傳回欄位中相異值的數目。

```
FieldValueCount (field_name)
```

樞紐分析表函數

這些函數僅可用於圖表運算式。

After

After() 傳回以樞紐分析表的維度值評估而得的運算式值，這些值會顯示在樞紐分析表的列區段內，目前資料行之後的資料行中。

```
After - 圖表函數([TOTAL] expression [ , offset [,n]])
```

Before

Before() 傳回以樞紐分析表的維度值評估而得的運算式值，這些值會顯示在樞紐分析表的列區段內，目前資料行之前的資料行中。

```
Before - 圖表函數([TOTAL] expression [ , offset [,n]])
```

First

First() 傳回以樞紐分析表的維度值評估而得的運算式值，這些值會顯示在樞紐分析表中目前列區段的第一個資料行中。此函數在所有圖表類型中都會傳回 NULL，除了樞紐分析表以外。

```
First - 圖表函數([TOTAL] expression [ , offset [,n]])
```

Last

Last() 傳回以樞紐分析表的維度值評估而得的運算式值，這些值會顯示在樞紐分析表中目前列區段的最後一個資料行中。此函數在所有圖表類型中都會傳回 NULL，除了樞紐分析表以外。

```
Last - 圖表函數([TOTAL] expression [ , offset [,n]])
```

ColumnNo

ColumnNo() 傳回樞紐分析表中目前列區段內，目前資料行的編號。第一行的編號為 1。

```
ColumnNo - 圖表函數([TOTAL])
```

NoOfColumns

NoOfColumns() 傳回樞紐分析表中目前列區段中的資料行數。

```
NoOfColumns - 圖表函數([TOTAL])
```

資料載入指令碼中的記錄間函數

Exists

Exists() 會判定特定欄位值是否已載入資料載入指令碼的欄位中。該函數會傳回 TRUE 或 FALSE，因此可以在 **LOAD** 陳述式或 **IF** 陳述式的 **where** 子句中使用。

```
Exists (field_name [, expr])
```

LookUp

Lookup() 會查看已載入的表格，並且傳回的 **field_name** 值相當於欄位 **match_field_name** 中出現的第一個值 **match_field_value**。該表格可以是目前表格，或者先前載入的另一個表格。

```
LookUp (field_name, match_field_name, match_field_value [, table_name])
```

Peek

Peek() 會傳回表格中已載入的列的欄位值。可以指定列號，也可以指定表格。若未指定列數，將會使用上一個載入的記錄。

```
Peek (field_name[, row_no[, table_name ] ])
```

Previous

Previous() 會使用由於 **where** 子句而尚未捨棄的先前輸入記錄中的資料，來尋找 **expr** 運算式的值。在內部表格的第一筆記錄中，此函數會傳回 NULL。

```
Previous (page 1236)(expr)
```

另請參見：

- *範圍函數 (page 1255)*

Above - 圖表函數

Above() 會在表格中某個資料行區段內的目前列上方，在某個列中評估運算式。用來計算的列取決於 **offset** 的值，如果呈現，預設情況下是正上方的列。對於表格以外的圖表，**Above()** 會評估圖表連續表同等表格中目前列上方的列。

語法:

```
Above([TOTAL] expr [ , offset [,count]])
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
offset	將一個 offsetn 指定為大於 0, 則可將運算式的評估從目前列往上移動 n 列。 若將位移指定為 0, 則會根據目前列來評估運算式。 若指定負值的位移, 會讓 Above 函數的作用相當於包含對應正值位移的 Below 函數。
count	將第三個引數 **count** 指定為大於 1, 函數會傳回 **count** 的範圍值, 從原始儲存格往上算 **count** 個表格列的每列一個值。 在這種形式下, 該函數可作為任何特殊範圍函數的引數。 *範圍函數 (page 1255)*
TOTAL	如果表格為單一維度或 **TOTAL** 限定詞作為引數, 則目前資料行區段一律等於整個資料行。

系統會對資料行區段的第一列傳回 NULL 值, 因為該列之上沒有任何列。

資料行區段定義為連續的儲存格子集, 針對目前排序順序中的維度具有相同的值。內部記錄圖表函數在資料行區段中進行計算, 並排除同等連續表圖表中的最右側維度。如果圖表中只有一個維度, 或者如果指定 TOTAL 限定詞, 則該運算式會對完整表格進行評估。

如果表格或同等表格有多個垂直維度, 則目前資料行區段僅會包含在所有維度資料行中與目前列有相同值的列, 除了顯示在欄位間排序順序中為最後一個維度的資料行之外。

限制:

- 遞迴呼叫會傳回 NULL。
- 當任何圖表運算式使用了此圖表函數時, 就不允許在圖表中依據 Y 值排序, 或在表格中依據運算式資料行排序。因此, 這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數, 視覺化的排序將還原為對此函數的排序輸入。

範例與結果:

Example 1:

範例 1 的表格視覺化

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

本範例展示的表格螢幕擷取畫面中，表格視覺化是從維度 **Customer** 和量值 `Sum(Sales)` 與 `Above(Sum(Sales))` 建立而成。

資料行 Above(Sum(Sales)) 會針對包含 **Astrida** 的 **Customer** 列傳回 NULL，因為該列之上沒有任何列。**Betacab** 列的結果顯示 **Astrida** 的 Sum(Sales) 值；**Canutility** 的結果顯示 **Betacab** 的 **Sum(Sales)** 值，以此類推。

針對標記為 Sum(Sales)+Above(Sum(Sales)) 的資料行，**Betacab** 列顯示 **Betacab** + **Astrida** 列加上 **Sum(Sales)** 值的結果 (539+587)。**Betacab** 列的結果顯示 **Canutility** + **Canutility** 列加上 **Sum(Sales)** 值的結果 (683+539)。

使用 `Sum(Sales)+Above(Sum(Sales), 3)` 運算式所建立且標記為 Above offset 3 的量值，擁有設為 3 的引數 **offset**，且具有使用目前列上方第三列值的效果。將目前 **Customer** 的 **Sum(Sales)** 值加到上方三列的 **Customer** 值。前三個 **Customer** 列傳回的值為 NULL。

該表格也顯示更複雜的量值：一個是從 `Sum(Sales)+Above(Sum(Sales))` 建立，另一個標記為 **Higher?**，是從 `IF(Sum(Sales)>Above(Sum(Sales)), 'Higher')` 建立而成。

此函數還可以用於表格之外的圖表中，例如長條圖。

對於其他圖表類型，將圖表轉換為連續表同等表格，以便您可以輕鬆解譯函數相關的列。

Example 2:

本範例展示的表格螢幕擷取畫面中，視覺化中新增了更多維度：**Month** 和 **Product**。對於有多個維度的圖表，包含 **Above**、**Below**、**Top** 及 **Bottom** 函數的運算式結果取決於按 Qlik Sense 排序的資料行維度的順序。Qlik Sense 會根據上一次排序維度所產生的資料行區段來評估功能。資料行排序順序在屬性面板的**排序**下進行控制，而不需要按照資料行在表格中呈現的順序。

以下範例 2 的表格視覺化螢幕擷取畫面中，最後一個排序的維度是 **Month**，因此 **Above** 函數會根據月份進行評估。每個月 (**Jan** 到 **Aug**) 的每個 **Product** 值都有一組結果 - 一個資料行區段。其後接著下一個資料行區段的序列：針對下個月 **Month** 的下個 **Product**。每個 **Product** 的每個 **Customer** 值都有一個資料行區段。

範例 2 的表格視覺化

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:

範例 3 的表格視覺化螢幕擷取畫面中，最後一個排序的維度是 **Product**。此程序是透過將維度 Product 移動到屬性面板上「排序」標籤中的位置 3 而完成。系統會針對每個 **Product** 評估 **Above** 函數，而因為只有兩個產品 **AA** 和 **BB**，每個序列中只會有一個非 NULL 結果。在 **Jan** 月份的 **BB** 列中，**Above(Sum(Sales))** 的值是 46。而 **AA** 列的值為 NULL。每月的每個 **AA** 列的值一律為 NULL，因為 AA 上方沒有任何 **Product** 值。第二個序列是對 **Feb** 月份的 **AA** 和 **BB** 進行評估，針對 **Customer** 值、**Astrida**。針對 **Astrida** 完成所有月份的評估後，該順序會針對第二個 **Customer**Betacab 重複執行一次，以此類推。

範例 3 的表格視覺化

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

範例 4

Example 4:	結果
Above 函數可做為範圍函數的輸入使用。例如：`RangeAvg (Above(Sum(Sales),1,3))`。	在 Above() 函數的引數中，offset 設定為 1，而 count 設定為 3。函數得到資料行區段 (其中有一列) 中目前列正上方三列的 Sum(Sales) 運算式結果。這三個值可用做 RangeAvg() 函數的輸入，該函數進而得出所提供數值範圍中的平均值。 含有 Customer 作為維度的表格可提供 RangeAvg() 運算式的以下結果。 Astrida - Betacab 587 Canutility 563 Divadip: 603

範例中使用的資料：

```
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 '|');
```

另請參見：

- *Below - 圖表函數 (page 1207)*
- *Bottom - 圖表函數 (page 1210)*

- *Top - 圖表函數 (page 1238)*
- *RangeAvg (page 1258)*

Below - 圖表函數

Below() 會在表格中某個資料行區段內的目前列下方，在某個列中評估運算式。用來計算的列取決於 **offset** 的值，如果呈現，預設情況下是正下方的列。對於表格以外的圖表，**Below()** 會評估圖表連續表同等表格中目前資料行下方的列。

語法:

```
Below([TOTAL] expr [ , offset [,count ]])
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
offset	將一個 **offset**n 指定為大於 1，則可將運算式的評估從目前列往下移動 n 列。 若將位移指定為 0，則會根據目前列來評估運算式。 若指定負值的位移，會讓 **Below** 函數的作用相當於包含對應正值位移的 **Above** 函數。
count	將第三個參數 **count** 指定為大於 1，函數會傳回 **count** 的範圍值，從原始儲存格往下算 **count** 個表格列的每列一個值。在這種形式下，該函數可作為任何特殊範圍函數的引數。 *範圍函數 (page 1255)*
TOTAL	如果表格為單一維度或 **TOTAL** 限定詞作為引數，則目前資料行區段一律等於整個資料行。

會對資料行區段的最後一列傳回 NULL 值，因為在該列之下沒有任何列。

資料行區段定義為連續的儲存格子集，針對目前排序順序中的維度具有相同的值。內部記錄圖表函數在資料行區段中進行計算，並排除同等連續表圖表中的最右側維度。如果圖表中只有一個維度，或者如果指定 TOTAL 限定詞，則該運算式會對完整表格進行評估。

如果表格或同等表格有多個垂直維度，則目前資料行區段僅會包含在所有維度資料行中與目前列有相同值的列，除了顯示在欄位間排序順序中為最後一個維度的資料行之外。

限制：

- 遞迴呼叫會傳回 NULL。
- 當任何圖表運算式使用了此圖表函數時，就不允許在圖表中依據 Y 值排序，或在表格中依據運算式資料行排序。因此，這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數，視覺化的排序將還原為對此函數的排序輸入。

範例與結果：

Example 1:

範例 1 的表格視覺化

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	-	-	-	-

範例 1 螢幕擷取畫面展示的表格中，表格視覺化是從維度 **Customer** 和量值 `Sum(Sales)` 與 `Below(Sum(Sales))` 建立而成。

資料行 **Below(Sum(Sales))** 會針對包含 **Divadip** 的 **Customer** 列傳回 NULL，因為該列之下沒有任何列。**Canutility** 列的結果顯示 **Divadip** 的 Sum(Sales) 值；**Betacab** 的結果顯示 **Canutility** 的 **Sum(Sales)** 值，以此類推。

該表格也顯示更複雜的量值，會顯示在具有下列標記的資料行中：`Sum(Sales)+Below(Sum(Sales))`、**Below +Offset 3** 與 **Higher?**。這些運算式的運作方式如以下段落所述。

針對標記為 **Sum(Sales)+Below(Sum(Sales))** 的資料行，**Astrida** 列顯示 **Betacab** + **Astrida** 列加上 **Sum (Sales)** 值的結果 (539+587)。**Betacab** 列的結果顯示 **Canutility** + **Betacab** 列加上 **Sum(Sales)** 值的結果 (539+683)。

使用 `Sum(Sales)+Below(Sum(Sales), 3)` 運算式所建立且標記為 **Below +Offset 3** 的量值，擁有設為 3 的引數 **offset**，且具有使用目前列下方第三列值的效果。將目前 **Customer** 的 **Sum(Sales)** 值加到下方三列的 **Customer** 值。最下面三個 **Customer** 列傳回的值為 NULL。

標記為 **Higher?** 的量值是從運算式：`IF(Sum(Sales)>Below(Sum(Sales)), 'Higher')` 建立而成。這會比較量值 **Sum(Sales)** 中目前列的值與其下方列的值。若目前列的值較大，輸出則為文字「Higher」。

此函數還可以用於表格之外的圖表中，例如長條圖。

對於其他圖表類型，將圖表轉換為連續表同等表格，以便您可以輕鬆解譯函數相關的列。

對於有多個維度的圖表，包含 **Above**、**Below**、**Top** 及 **Bottom** 函數的運算式結果取決於按 Qlik Sense 排序的資料行維度的順序。Qlik Sense 會根據上一次排序維度所產生的資料行區段來評估功能。資料行排序順序在屬性面板的**排序**下進行控制，而不需要按照資料行在表格中呈現的順序。請參閱 **Above** 函數中的範例：2 以取得進一步詳細資料。

範例 2

Example 2:	**結果**
Below 函數可做為範圍函數的輸入使用。例如：`RangeAvg (Below(Sum(Sales),1,3))`。	在 **Below()** 函數的引數中，offset 設定為 1，而 count 設定為 3。函數得到資料行區段 (其中有一列) 中目前列正下方三列的 **Sum(Sales)** 運算式結果。這三個值可用做 RangeAvg() 函數的輸入，該函數進而得出所提供數值範圍中的平均值。 含有 **Customer** 作為維度的表格可提供 RangeAvg() 運算式的以下結果。
	Astrida 659.67 Betacab 720 Canutility 757 Divadip: -

範例中使用的資料：

```
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 '|');
```

另請參見:

- *Above - 圖表函數 (page 1202)*
- *Bottom - 圖表函數 (page 1210)*
- *Top - 圖表函數 (page 1238)*
- *RangeAvg (page 1258)*

Bottom - 圖表函數

Bottom() 會在表格中某個資料行區段內的最後一 (底端) 列, 評估運算式。用來計算的列取決於 **offset** 的值, 如果呈現, 預設情況下是底端的列。對於表格以外的圖表, 該運算式會評估圖表連續表同等表格中目前資料行的最後一列。

語法:

```
Bottom([TOTAL] expr [ , offset [,count ]])
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
offset	將一個 **offset**n 指定為大於 1, 則可將運算式的評估從底端列往上移動 n 列。 若指定負值的位移, 會讓 **Bottom** 函數的作用相當於包含對應正值位移的 **Top** 函數。
count	將第三個參數 **count** 指定為大於 1, 函數會傳回不是一個, 而是 **count** 的範圍值, 目前資料行區段的最後 **count** 列的每列一個值。在這種形式下, 該函數可作為任何特殊範圍函數的引數。 *範圍函數 (page 1255)*
TOTAL	如果表格為單一維度或 **TOTAL** 限定詞作為引數, 則目前資料行區段一律等於整個資料行。

資料行區段定義為連續的儲存格子集, 針對目前排序順序中的維度具有相同的值。內部記錄圖表函數在資料行區段中進行計算, 並排除同等連續表圖表中的最右側維度。如果圖表中只有一個維度, 或者如果指定 TOTAL 限定詞, 則該運算式會對完整表格進行評估。

如果表格或同等表格有多個垂直維度, 則目前資料行區段僅會包含在所有維度資料行中與目前列有相同值的列, 除了顯示在欄位間排序順序中為最後一個維度的資料行之外。

限制:

- 遞迴呼叫會傳回 NULL。
- 當任何圖表運算式使用了此圖表函數時,就不允許在圖表中依據 Y 值排序,或在表格中依據運算式資料行排序。因此,這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數,視覺化的排序將還原為對此函數的排序輸入。

範例與結果:

範例 1 的表格視覺化

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

本範例展示的表格螢幕擷取畫面中,表格視覺化是從維度 **Customer** 和量值 `Sum(Sales)` 與 `Bottom(Sum(Sales))` 建立而成。

資料行 **Bottom(Sum(Sales))** 針對所有列傳回 757,因為這是底端列:**Divadip** 的值。

該表格也顯示更複雜的量值:一個是從 `Sum(Sales)+Bottom(Sum(Sales))` 建立,另一個標記為 **Bottom offset 3**,是使用運算式 `Sum(Sales)+Bottom(Sum(Sales), 3)` 建立且具有設為 3 的 **offset** 引數。將目前列的 **Sum(Sales)** 值加上從底端列算起第三列的值,也就是目前列加上 **Betacab** 的值。

範例:2

本範例展示的表格螢幕擷取畫面中,視覺化中新增了更多維度:**Month** 和 **Product**。對於有多個維度的圖表,包含 **Above**、**Below**、**Top** 及 **Bottom** 函數的運算式結果取決於按 Qlik Sense 排序的資料行維度的順序。Qlik Sense 會根據上一次排序維度所產生的資料行區段來評估功能。資料行排序順序在屬性面板的**排序**下進行控制,而不需要按照資料行在表格中呈現的順序。

在第一個表格中,系統是根據 **Month** 評估運算式,而第二個表格中則是根據 **Product** 加以評估。量值 **End value** 包含運算式 `Bottom(Sum(Sales))`。**Month** 的底端列是 Dec,而螢幕擷取畫面中顯示的 Dec 的值和 **Product** 值是 22。(部分列已在螢幕擷取畫面之外進行編輯以節省空間。)

範例 2 的第一個表格。End value 量值根據 Month (Dec) 的 Bottom 值。

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

範例 2 的第二個表格。End value 量值根據 Product (Astrida 的 BB) 的 Bottom 值。

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

請參閱 **Above** 函數中的範例:2 以取得進一步詳細資料。

範例 3

範例:3	**結果**
Bottom 函數可做為範圍函數的輸入使用。例如:`RangeAvg (Bottom(Sum(Sales),1,3))`。	在 **Bottom()** 函數的引數中, offset 設定為 1, 而 count 設定為 3。函數得到資料行區段中頂端列上面一列算起的三列 (因為 offset=1) 和其上方兩列 (其中有一列) 的 **Sum(Sales)** 運算式結果。這三個值可用做 RangeAvg() 函數的輸入, 該函數進而得出所提供數值範圍中的平均值。 含有 **Customer** 作為維度的表格可提供 RangeAvg() 運算式的以下結果。

範例：3	結果	
	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 '|');
```

另請參見：

Top - 圖表函數 (page 1238)

Column - 圖表函數

Column() 會忽略維度，傳回在對應於連續表中 **ColumnNo** 的資料行中找到的值。例如，**Column(2)** 傳回第二個量值資料行的值。

語法：

```
Column(ColumnNo)
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
ColumnNo	包含量值的表格中資料行的資料行數。 *Column() 函數會忽略維度資料行。*

限制:

- 遞迴呼叫會傳回 NULL。
- 若 **ColumnNo** 參考無量值的資料行, 系統就會傳回 NULL 值。
- 當任何圖表運算式使用了此圖表函數時, 就不允許在圖表中依據 Y 值排序, 或在表格中依據運算式資料行排序。因此, 這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數, 視覺化的排序將還原為對此函數的排序輸入。

範例與結果:

範例: 總銷售百分比

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
A	AA	15	10	150	505	29.70
A	AA	16	4	64	505	12.67
A	BB	9	9	81	505	16.04
B	BB	10	5	50	505	9.90
B	CC	20	2	40	505	7.92
B	DD	25	-	0	505	0.00
C	AA	15	8	120	505	23.76
C	CC	19	-	0	505	0.00

範例: 所選客戶的銷售百分比

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
A	AA	15	10	150	295	50.85

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
A	AA	16	4	64	295	21.69
A	BB	9	9	81	295	27.46

範例與結果

範例	結果
使用運算式：`Sum(UnitPrice*UnitSales)` 將 Order Value 作為量值新增到表格。	Column(1) 的結果取自資料行 Order Value，因為這是第一個量值資料行。
使用運算式將 Total Sales Value 作為量值新增：`Sum(TOTAL UnitPrice*UnitSales)`	Column(2) 的結果取自 Total Sales Value，因為這是第二個量值資料行。
使用運算式 `100*Column(1)/Column(2)` 將 % Sales 作為量值新增。	查看範例 *總銷售百分比 (page 1214)* 中的 % Sales 資料行結果。
選取 Customer A。	選項變更了 Total Sales Value，因此 %Sales 也改變。請參閱範例 *所選客戶的銷售百分比 (page 1214)*。

範例中使用的資料：

```
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 - 圖表函數

Dimensionality() 傳回目前列的維度數目。如果是樞紐分析表格，則該函數傳回有非彙總內容 (亦即不包含部分加總或摺疊彙總) 之維度資料行的總數目。

語法：

```
Dimensionality ( )
```

傳回的資料類型： 整數

限制：

此函數僅適用於圖表。對於所有圖表類型 (除了樞紐分析表以外)，此函數會傳回所有列中的維度數目 (總計列除外，該列的維度數目為 0)。

當任何圖表運算式使用了此圖表函數時，就不允許在圖表中依據 Y 值排序，或在表格中依據運算式資料行排序。因此，這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數，視覺化的排序將還原為對此函數的排序輸入。

範例：使用維度的圖表運算式

範例：圖表運算式

Dimensionality() 函數可搭配樞紐分析表使用作為圖表運算式，且您想在此根據具有非彙總資料的列中的維度數量套用不同儲存格格式。此範例使用 Dimensionality() 函數將背景色彩套用至符合給定條件的表格儲存格。

載入指令碼

在資料載入編輯器中載入下列資料作為內嵌載入，以建立以下的圖表運算式範例。

```
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 ',');
```

圖表運算式

在 Qlik Sense 工作表中建立具有**國家**和**產品**作為維度的樞紐分析表視覺化。新增 **Sum(Sales)**、**Sum (Budget)** 和 **Dimensionality()** 作為量值。

在**屬性**面板中，輸入下列運算式作為 **Sum(Sales)** 量值的**背景色彩運算式**。

```
If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156),
If(Dimensionality()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)
))
```

結果：

Country / Product	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**

解釋

運算式 `If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156), If(Dimensionality()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)))` 包含條件陳述式，這會檢查維度值以及每個產品的 Sum(Sales) 和 Sum(Budget)。若符合條件，背景色彩會套用至 Sum(Sales) 值。

Exists

Exists() 會判定特定欄位值是否已載入資料載入指令碼的欄位中。該函數會傳回 TRUE 或 FALSE，因此可以在 **LOAD** 陳述式或 **IF** 陳述式的 **where** 子句中使用。

*您也可以使用 **Not Exists()** 以判定欄位值是否未載入，但建議您謹慎使用 where 子句中的 **Not Exists()**。**Exists()** 函數可在目前表格中測試先前載入的表格和先前載入的值。因此，只有首次出現時才會載入。第二次出現時，該值已經載入。請參閱範例瞭解更多資訊。*

語法：

```
Exists(field_name [, expr])
```

傳回的資料類型: 布林

引數:

引數

引數	描述
field_ name	想要搜尋值的欄位名稱。您可以使用不含引號的明確欄位名稱, 欄位必須已由指令碼載入。這代表,您無法參考在指令碼中進一步向下的子句中載入的欄位。
expr	您想要檢查是否存在的值。您可以使用明確的值或參考目前 LOAD 陳述式中一個或數個欄位的運算式。 *您無法參考未納入目前 LOAD 陳述式的欄位。* 此引數為選擇性。若您省略,函數將會檢查目前記錄中 **field_name** 的值是否已存在。

範例與結果:

範例 1

```
Exists (Employee)
```

如果目前記錄中欄位 **Employee** 的值已經存在於任何先前讀取的記錄 (包含該欄位) 中,則會傳回 -1 (True)。

陳述式 `Exists (Employee, Employee)` 與 `Exists (Employee)` 相當。

範例 2

```
Exists(Employee, 'Bill')
```

如果欄位值 **'Bill'** 被發現位於欄位 **Employee** 的目前內容中,則會傳回 -1 (True)。

範例 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;
```

此結果位於表格中，您可以使用維度 Employee 和 Address 在表格視覺化中使用該表格。

where 子句 `where Exists (Employee)` 僅表示來自表格 Citizens 的名稱，這些名稱也位於載入至新表格的 Employees 中。Drop 陳述式會移除表格 Employees，以避免混淆。

結果

Employee	Address
Bill	New York
John	Miami
Steve	Chicago

範例 4

```
Employees:
Load * inline [
Employee|ID|Salary
Bill|001|20000
John|002|30000
Steve|003|35000
] (delimiter is '|');

Citizens:
Load * inline [
Employee|Address
Bill|New York
Mary|London
Steve|Chicago
Lucy|Madrid
Lucy|Paris
John|Miami
] (delimiter is '|') where not Exists (Employee);

Drop Tables Employees;
```

where 子句包含 not: `where not Exists (Employee)`。

這表示只有來自表格 Citizens 且不在 Employees 中的名稱才會被載入新表格中。

請注意，Citizens 表格中有兩個 Lucy 的值，但只有一個包含在結果表格中。透過值 Lucy 載入第一列時，這會包含在 Employee 欄位中。因此，檢查第二行時，值已經存在。

結果

員工	地址
Mary	London
Lucy	Madrid

範例 5

此範例顯示載入所有值的方式。

```
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;
```

為了能夠取得 Lucy 的所有值，已變更兩件事：

- 插入了前置載入到 Employees 表格，其中 Employee 已重新命名為 Name。
  `Load Employee As Name;`
- Citizens 中的 Where 條件已變更為：
  `not Exists (Name, Employee).`

這會為 Name 和 Employee 建立欄位。檢查含有 Lucy 的第二列時，這仍然不會存在於 Name。

結果

員工	地址
Mary	London
Lucy	Madrid
Lucy	Paris

FieldIndex

FieldIndex() 傳回在欄位 **field_name** 中欄位值 **value** 的位置 (依據載入順序)。

語法:

```
FieldIndex(field_name , value)
```

傳回的資料類型: 整數

引數:

引數

引數	描述
field_name	索引需要的欄位名稱。例如表格中的資料行。必須作為字串值提供。這表示欄位名稱必須含括在單引號中。
value	欄位 **field_name** 的值。

限制:

- 若無法在欄位 **field_name** 的欄位值內找到 **value**, 則會傳回 0。
- 當任何圖表運算式使用了此圖表函數時, 就不允許在圖表中依據 Y 值排序, 或在表格中依據運算式資料行排序。因此, 這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數, 視覺化的排序將還原為對此函數的排序輸入。此限制不適用於對等指令碼函數。

範例與結果:

以下範例使用來自表格 **Names** 的欄位:**First name**。

範例與結果

範例	結果
新增範例資料到應用程式中並加以執行。	表格 **Names** 已載入, 如同樣本資料。
圖表函數:在包含維度 First name 的表格中, 新增為量值。	
`FieldIndex ('First name','John')`	1, 因為「John」是 **First name** 欄位載入順序中第一個出現者。請注意, 在篩選窗格中 **John** 會出現在頂端第二項, 因為是依字母順序而非載入順序。
`FieldIndex ('First name','Peter')`	4, 因為 **FieldIndex()** 只傳回一個值, 即為載入順序中第一個出現者。
指令碼函數:假定表格 **Names** 已載入, 如同範例資料:	

範例	結果
John1: Load FieldIndex('First name','John') as MyJohnPos Resident Names;	MyJohnPos=1，因為「John」是 **First name** 欄位載入順序中第一個出現者。請注意，在篩選窗格中 **John** 會出現在頂端第二項，因為是依字母順序而非載入順序。
Peter1: Load FieldIndex('First name','Peter') as MyPeterPos Resident Names;	MyPeterPos=4，因為 **FieldIndex()** 只傳回一個值，即為載入順序中第一個出現者。

範例中使用的資料：

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');

John1:
Load FieldIndex('First name','John') as MyJohnPos
Resident Names;

Peter1:
Load FieldIndex('First name','Peter') as MyPeterPos
Resident Names;
```

FieldValue

FieldValue() 傳回在欄位 **field_name** 的 **elem_no** 位置中找到的欄位值 (依據載入順序)。

語法：

```
FieldValue(field_name , elem_no)
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
field_name	值需要的欄位名稱。例如表格中的資料行。必須作為字串值提供。這表示欄位名稱必須含括在單引號中。
elem_no	緊接著載入順序，傳回的值所針對的欄位位置 (元素) 編號。可對應至表格中的列，不過須視載入哪些元素 (列) 而定。

限制：

- 若 **elem_no** 大於欄位值的數目，則會傳回 NULL。
- 當任何圖表運算式使用了此圖表函數時，就不允許在圖表中依據 Y 值排序，或在表格中依據運算式資料行排序。因此，這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數，視覺化的排序將還原為對此函數的排序輸入。此限制不適用於對等指令碼函數。

範例

載入指令碼

在資料載入編輯器中載入下列資料作為內嵌載入，以建立以下範例。

```
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;
```

建立視覺化

在 Qlik Sense 工作表中建立表格視覺化。將欄位 **First name**、**MyPos1** 和 **MyPos2** 新增至表格。

結果

First name	**MyPos1**	**MyPos2**
Jane	John	Jane
John	John	Jane
Mark	John	Jane
Peter	John	Jane
Sue	John	Jane

解釋

FieldValue('First name','1') 產生 John 作為所有名字的 **MyPos1** 值，因為 John 以**名字**欄位的載入順序第一個顯示。請注意，在篩選窗格中 John 會出現在頂端第二項，Jane 的後面，因為是依字母順序而非載入順序。

FieldValue('First name','5') 產生 Jane 作為所有名字的 **MyPos2** 值，因為 Jane 以 **First name** 欄位的載入順序第一個顯示。

FieldValueCount

FieldValueCount() 是一個 **integer** 函數，傳回欄位中相異值的數目。

部分載入可以從資料移除值，這不會反映在傳回的數字中。傳回的數字將對應至透過初始載入或任何後續部分載入所載入的所有相異值。

> *當任何圖表運算式使用了此圖表函數時，就不允許在圖表中依據 Y 值排序，或在表格中依據運算式資料行排序。因此，這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數，視覺化的排序將還原為對此函數的排序輸入。此限制不適用於對等指令碼函數。*

語法：

```
FieldValueCount(field_name)
```

傳回的資料類型：整數

引數：

引數

引數	描述
field_name	值需要的欄位名稱。例如表格中的資料行。必須作為字串值提供。這表示欄位名稱必須含括在單引號中。

範例與結果：

以下範例使用來自表格 **Names** 的欄位 **First name**。

範例與結果

範例	結果
新增範例資料到應用程式中並加以執行。	表格 **Names** 已載入，如同樣本資料。
圖表函數：在包含維度 First name 的表格中，新增為量值。	
`FieldValueCount('First name')`	5，因為 **Peter** 出現兩次。

範例	結果
`FieldValueCount('Initials')`	6, 因為 **Initials** 只有相異值。
指令碼函數:假定表格 **Names** 已載入, 如同範例資料:	

`FieldCount1:`
`Load FieldValueCount('First name') as MyFieldCount1`
`Resident Names;`
	`MyFieldCount1=5`, 因為「Peter」出現兩次。

`FieldCount2:`
`Load FieldValueCount('Initials') as MyInitialsCount1`
`Resident Names;`
	`MyFieldCount1=6`, 因為「Initials」只有相異值。

範例中使用的資料:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');

FieldCount1:
Load FieldValueCount('First name') as MyFieldCount1
Resident Names;

FieldCount2:
Load FieldValueCount('Initials') as MyInitialsCount1
Resident Names;
```

LookUp

Lookup() 會查看已載入的表格, 並且傳回的 **field_name** 值相當於欄位 **match_field_name** 中出現的第一個值 **match_field_value**。該表格可以是目前表格, 或者先前載入的另一個表格。

語法:

```
lookup(field_name, match_field_name, match_field_value [, table_name])
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
field_name	需要傳回值的欄位的名稱。輸入值必須指定為字串 (如引號中的常值)。

引數	描述
match_field_name	要在其中查閱 **match_field_value** 之欄位的名稱。輸入值必須指定為字串 (如引號中的常值)。
match_field_value	要在 **match_field_name** 欄位中查閱的值。
table_name	要在其中查閱值的表格名稱。輸入值必須指定為字串 (如引號中的常值)。 如果省略 **table_name**, 則會採用目前表格。

不含引號的引數會參考目前表格。若要參考其他表格, 請將引數含括在單引號中。

限制:

搜尋順序為載入順序, 除非表格是複雜運算 (如 join) 的結果, 若是如此, 順序無法明確定義。**field_name** 和 **match_field_name** 必須是相同表格 (以 **table_name** 指定) 中的欄位。

如果找不到符合的值, 則會傳回 NULL。

範例

載入指令碼

在資料載入編輯器中載入下列資料作為內嵌載入, 以建立以下範例。

```
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;
```

建立視覺化

在 Qlik Sense 工作表中建立表格視覺化。將欄位 **ProductID**、**InvoiceID**、**CustomerID**、**Units** 和 **CategoryID** 新增至表格。

結果

結果表格

ProductID (產品 ID)	InvoiceID	CustomerID	單位:	CategoryID
1	1	Astrida	8	1
2	1	Astrida	6	1
3	2	Betacab	10	2
3	3	Divadip	5	2
4	4	Divadip	10	3

解釋

範例資料使用下列形式的 **Lookup()** 函數:

```
Lookup('Category', 'ProductID', ProductID, 'ProductList')
```

首先載入 **ProductList** 表格。

Lookup() 函數用來建置 **OrderData** 表格。如果將第三個引數指定為 **ProductID**。這是將在 **ProductList** 的第二個引數 **'ProductID'** 中查閱值的欄位,如含括的單引號所指示。

函數傳回 '**Category**' (在 **ProductList** 表格中) 的值,作為 **CategoryID** 載入。

drop 陳述式會從資料模型中刪除 **ProductList** 表格,因為不需要該項目,這會留下產生的 **OrderData** 表格。

Lookup() 函數是彈性的,可以存取任何先前載入的表格。然而,它與 Applymap() 函數相比較速度較慢。

另請參見:

ApplyMap (page 1248)

NoOfRows - 圖表函數

NoOfRows() 傳回表格中目前資料行區段中的列數。對於點陣圖圖表,**NoOfRows()** 傳回圖表的連續表同等表格中的列數。

如果表格或同等表格有多個垂直維度,則目前資料行區段僅會包含在所有維度資料行中與目前列有相同值的列,除了顯示在欄位間排序順序中為最後一個維度的資料行之外。

> *當任何圖表運算式使用了此圖表函數時，就不允許在圖表中依據 Y 值排序，或在表格中依據運算式資料行排序。因此，這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數，視覺化的排序將還原為對此函數的排序輸入。*

語法：

```
NoOfRows([TOTAL])
```

傳回的資料類型： 整數

引數：

引數

引數	描述
TOTAL	如果表格為單一維度或 **TOTAL** 限定詞作為引數，則目前資料行區段一律等於整個資料行。

範例：使用 NoOfRows 的圖表運算式

範例 - 圖表運算式

載入指令碼

在資料載入編輯器中載入下列資料作為內嵌載入，以建立以下的圖表運算式範例。

```
        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 '|');
```

圖表運算式

在 Qlik Sense 工作表中建立具有 **Region** 和 **SubRegion** 作為維度的表格視覺化。將 `RowNo( )`、`NoOfRows()` 和 `NoOfRows(Total)` 新增為量值。

結果

Region	SubRegion	RowNo()	NoOfRows()	NoOfRows (Total)
Africa	Eastern	1	2	9
Africa	Western	2	2	9
Americas	Central	1	2	9
Americas	Northern	2	2	9
Asia	Eastern	1	1	9
Europe	Eastern	1	3	9
Europe	Northern	2	3	9
Eurrope	Western	3	3	9
Oceania	Australia	1	1	9

解釋

在此範例中，排序順序依據第一個維度 Region。因此，每個資料行區段由具有相同值的區域群組組成，例如非洲。

RowNo() 資料行顯示每個資料行區段的列數，例如非洲區域有兩列。接著下一個資料行區段 Americas 的列編號會再度從 1 開始。

NoOfRows() 資料行計算每個資料行區段中的列數，例如歐洲在資料行區段中有三列。

由於 `NoOfRows()` 的 `TOTAL` 引數，**NoOfRows(Total)** 資料行會忽略維度，並計入表格中的列。

若表格排序在第二個維度 SubRegion，資料行區段會以該維度為基礎，因此每個 SubRegion 的列數計算會變更。

另請參見：

RowNo - 圖表函數 (page 547)

Peek

Peek() 會傳回表格中已載入的列的欄位值。可以指定列號，也可以指定表格。若未指定列數，將會使用上一個載入的記錄。

peek() 函數通常用來在先前載入的表格中尋找相關邊界，亦即特定欄位的第一個值或最後一個值。在大部分情況下，此值儲存在變數中以供之後使用，例如作為 do-while 迴圈中的條件。

語法：

```
Peek(
```

```
field_name
```

```
[, row_no[, table_name ] ])
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
field_ name	需要傳回值的欄位的名稱。輸入值必須指定為字串 (如引號中的常值)。
row_no	指定所需欄位的表格中的列。可以是運算式,但是必須解析為整數。0 代表第一筆記錄,1 代表第二筆記錄,依此類推。負數表示從表格結尾算起的順序。-1 代表上次記錄讀取。 如未指定 **row_no**,則會採用 -1。
table_ name	沒有結束分號的表格標籤。如未指定 **table_name**,則會採用目前表格。若用於 **LOAD** 陳述式之外,或參考其他表格,則必須包括 **table_name**。

限制:

函數只能傳回來自已載入記錄的值。這表示,在第一個表格記錄中,使用 -1 作為 row_no 的呼叫將會傳回 NULL。

範例與結果:

範例 1

將範例指令碼新增至您的應用程式並予以執行。若要查看結果,將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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;
```

結果表格

員工代碼	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，因為 `Peek('EmployeeCode',0, 'EmployeeDates')` 傳回表格 EmployeeDates 中 EmployeeCode 的第一個值。

LastCode = 106，因為 `Peek('EmployeeCode',-1, 'EmployeeDates')` 傳回表格 EmployeeDates 中 EmployeeCode 的最後一個值。

取代引數 **row_no** 的值會傳回表格中其他列的值，如下所示：

`Peek('EmployeeCode',2, 'EmployeeDates')` 傳回表格中的第三個值 103 作為 FirstCode。

然而，請注意，在這些範例中，如果未將表格指定為第三個引數 **table_name**，則函數會參考目前 (在此情況下，內部) 表格。

範例 2

若您想要進一步存取表格中的資料，需要以兩個步驟進行：首先，將整個表格載入到暫存資料表，然後在使用 **Peek()** 時重新排序。

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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.

結果表格

ID	清單	值
1	3,4	4
1	3,4,6	6
1	3	3
2	1,11	11
2	1	1
3	7,8	8
3	7	7
5	2,78	78
5	2,78,13	13
5	2	2

IF() 陳述式從臨時表格 T1 建置。
`Peek('ID')` 會參考目前表格 T2 中先前列內的欄位 ID。
`Peek('List')` 會參考表格 T2 中先前列內的欄位 List，目前建置為評估運算式。

評估陳述式如下：
如果 ID 的目前值與 ID 的前一個值相同，則寫入與 Value 的目前值串連的 Peek('List') 的值。否則，僅寫入 Value 的目前值。

如果 Peek('List') 已包含串連的結果，則 Peek('List') 的新結果將串連至其中。

*請注意 **Order by** 子句。這會指定如何排序表格 (按照 ID 遞增順序)。沒有此項，Peek() 函數將使用內部表格具有的任意順序，這可能產生非預期的結果。*

範例 3

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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];
```

結果表格

金額	AmountMonthBefore	月
1	4	2022-06
2	-	2022-01
3	2	2022-02
4	9	2022-05
7	3	2022-03
9	7	2022-04

欄位 AmountMonthBefore 將會保留來自上個月的金額。

在此會省略 row_no 和 table_name 參數，以便使用預設值。在此範例中，下列三個函數呼叫等同：

- Peek(Amount)
- Peek(Amount,-1)
- Peek(Amount,-1,'Amounts')

使用 -1 作為 row_no 表示將會使用來自上一列的值。透過替代此值，可以擷取表格中其他列的值：

Peek(Amount,2) 傳回表格中的第三個值：7。

範例 4

需要正確排序資料，才能取得正確結果，但是很遺憾，不一定能夠如此。此外，Peek() 函數無法用於參考尚未載入的資料。使用暫存資料表並透過資料執行多次傳遞，可避免這類問題。

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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;
```

解釋

會根據月份排序初始表格，這表示在許多情況下，peek() 函數會傳回錯誤產品的金額。因此，需要重新排序此表格。進行方式是透過建立新表格 tmp2Amounts 的資料執行第二次傳遞。請注意 Order By 子句。這先按產品排序記錄，然後以遞增順序按月份排序。

需要 If() 函數，因為若上一列包含相同產品 (而非上個月) 的資料，則只能計算 AmountMonthBefore。藉由比較目前列的產品與上一列的產品，可驗證此條件。

建立第二個表格後，會使用 Drop Table 陳述式卸除第一個表格 tmp1Amounts。

最後，會透過資料進行第三次傳遞，但此時會以相反順序排序月份。以此方式，也能計算 AmountMonthAfter。

Order by 子句指定表格的排序方式;沒有此項，Peek() 函數將使用內部表格具有的任意順序，這可能產生非預期的結果。

結果

結果表格

月	產品	金額	AmountMonthBefore	AmountMonthAfter
2022-01	A	8	-	6
2022-02	B	3	-	4
2022-03	A	6	8	6
2022-04	B	4	3	1
2022-05	A	6	6	5
2022-06	B	1	4	5

月	產品	金額	AmountMonthBefore	AmountMonthAfter
2022-01	A	5	6	7
2022-02	B	5	1	6
2022-03	A	7	5	4
2022-04	B	6	5	8
2022-05	A	4	7	-
2022-06	B	8	6	-

範例 5

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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;
```

結果

結果表格

季度	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

解釋

LOAD 陳述式 **Load *, rangesum(SumVal,peek('AccSumVal')) as AccSumVal** 包括遞迴呼叫，其中先前的值會新增至目前的值。此操作用來計算指令碼中的累積值。

另請參見：

Previous

Previous() 會使用由於 **where** 子句而尚未捨棄的先前輸入記錄中的資料，來尋找 **expr** 運算式的值。在內部表格的第一筆記錄中，此函數會傳回 NULL。

語法：

```
Previous(expr)
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。 運算式可以包含巢狀 **previous()** 函數，以存取更早之前的記錄。資料是直接擷取自輸入來源，因此也可參考尚未載入 Qlik Sense 的欄位，即尚未儲存在關聯資料庫中的欄位。

限制：

在內部表格的第一筆記錄中，此函數會傳回 NULL。

範例：

在載入指令碼中輸入下列內容：

```
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 '|');
```

透過使用 **Load** 陳述式中的 **Previous()** 函數，我們可以比較 Sales 的目前值與前置值，並將它用在第三個欄位 Increase 中。

結果表格

月	銷售額	增加
1	12	-
2	13	1
3	15	2
4	17	2
5	21	4

月	銷售額	增加
6	21	0
7	22	1
8	23	1
9	32	9
10	35	3
11	40	5
12	41	1

Top - 圖表函數

Top() 會在表格中某個資料行區段內的第一 (頂端) 列，評估運算式。用來計算的列取決於 **offset** 的值，如果呈現，預設情況下是頂端的列。對於表格以外的圖表，**Top()** 會評估圖表連續表同等表格中目前資料行的第一列。

語法：

```
Top([TOTAL] expr [ , offset [,count ]])
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
offset	將 **offset** 的 n 指定為大於 1，則可將運算式的評估從頂端列往下移動 n 列。 若指定負值的位移，會讓 **Top** 函數的作用相當於包含對應正值位移的 **Bottom** 函數。
count	將第三個參數 **count** 指定為大於 1，函數會傳回 **count** 的範圍值，目前資料行區段的最後 **count** 列的每列一個值。在這種形式下，該函數可作為任何特殊範圍函數的引數。*範圍函數 (page 1255)*
TOTAL	如果表格為單一維度或 **TOTAL** 限定詞作為引數，則目前資料行區段一律等於整個資料行。

資料行區段定義為連續的儲存格子集，針對目前排序順序中的維度具有相同的值。內部記錄圖表函數在資料行區段中進行計算，並排除同等連續表圖表中的最右側維度。如果圖表中只有一個維度，或者如果指定 TOTAL 限定詞，則該運算式會對完整表格進行評估。

如果表格或同等表格有多個垂直維度，則目前資料行區段僅會包含在所有維度資料行中與目前列有相同值的列，除了顯示在欄位間排序順序中為最後一個維度的資料行之外。

限制：

- 遞迴呼叫會傳回 NULL。
- 當任何圖表運算式使用了此圖表函數時，就不允許在圖表中依據 Y 值排序，或在表格中依據運算式資料行排序。因此，這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數，視覺化的排序將還原為對此函數的排序輸入。

範例與結果：

範例：1

本範例展示的表格螢幕擷取畫面中，表格視覺化是從維度 **Customer** 和量值 `Sum(Sales)` 與 `Top(Sum(Sales))` 建立而成。

資料行 **Top(Sum(Sales))** 針對所有列傳回 587，因為這是頂端列：**Astrida** 的值。

該表格也顯示更複雜的量值：一個是從 `Sum(Sales)+Top(Sum(Sales))` 建立，另一個標記為 **Top offset 3**，是使用運算式 `Sum(Sales)+Top(Sum(Sales), 3)` 建立且具有設為 3 的 **offset** 引數。將目前列的 **Sum(Sales)** 值加上從頂端列算起第三列的值，也就是目前列加上 **Canutility** 的值。

範例 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

範例：2

本範例展示的表格螢幕擷取畫面中，視覺化中新增了更多維度：**Month** 和 **Product**。對於有多個維度的圖表，包含 **Above**、**Below**、**Top** 及 **Bottom** 函數的運算式結果取決於按 Qlik Sense 排序的資料行維度的順序。Qlik Sense 會根據上一次排序維度所產生的資料行區段來評估功能。資料行排序順序在屬性面板的**排序**下進行控制，而不需要按照資料行在表格中呈現的順序。

範例 2 的第一個表格。First value 量值根據 Month (Jan) 的 Top 值。

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

範例 2 的第二個表格。First value 量值根據 Product (Astrida 的 AA) 的 Top 值。

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

請參閱 **Above** 函數中的範例:2 以取得進一步詳細資料。

範例 3

範例：3	結果
Top 函數可做為範圍函數的輸入使用。例如：`RangeAvg (Top(Sum(Sales),1,3))`。	在 **Top()** 函數的引數中，offset 設定為 `1`，而 count 設定為 `3`。函數得到資料行區段中頂端列下面一列算起的三列 (因為 offset=1) 和其下方兩列 (其中有一列) 的 **Sum(Sales)** 運算式結果。這三個值可用做 RangeAvg() 函數的輸入，該函數進而得出所提供數值範圍中的平均值。 含有 **Customer** 作為維度的表格可提供 RangeAvg() 運算式的以下結果。
	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 '|');
```

另請參見：

- *Bottom - 圖表函數 (page 1210)*
- *Above - 圖表函數 (page 1202)*

- *Sum - 圖表函數 (page 328)*
- *RangeAvg (page 1258)*
- *範圍函數 (page 1255)*

SecondaryDimensionality- 圖表函數

SecondaryDimensionality() 傳回有非彙總內容 (亦即不包含部分加總或摺疊彙總) 之維度樞紐分析表列的數目。此函數相當於針對水平樞紐分析表維度的 **dimensionality()** 函數。

語法:

```
SecondaryDimensionality( )
```

傳回的資料類型: 整數

限制:

- 除非在樞紐分析表中使用, 否則 **SecondaryDimensionality** 函數總是傳回 0。
- 當任何圖表運算式使用了此圖表函數時, 就不允許在圖表中依據 Y 值排序, 或在表格中依據運算式資料行排序。因此, 這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數, 視覺化的排序將還原為對此函數的排序輸入。

After - 圖表函數

After() 傳回以樞紐分析表的維度值評估而得的運算式值, 這些值會顯示在樞紐分析表的列區段內, 目前資料行之後的資料行中。

語法:

```
after([TOTAL] expr [, offset [, count ]])
```

當任何圖表運算式使用了此圖表函數時, 就不允許在圖表中依據 Y 值排序, 或在表格中依據運算式資料行排序。因此, 這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數, 視覺化的排序將還原為對此函數的排序輸入。

此函數在所有圖表類型中都會傳回 NULL, 除了樞紐分析表以外。

引數:

引數	描述
expr	包含待測量之資料的運算式或欄位。
offset	將一個 **offset** n, 指定為大於 1, 則可將運算式的評估從目前列往右移動 n 列。 若將位移指定為 0, 則會根據目前列來評估運算式。 若指定負值的位移, 會讓 **After** 函數的作用相當於包含對應正值位移的 **Before** 函數。

引數	描述
count	將第三個參數 **count** 指定為大於 1, 函數會傳回一系列值, 從原始儲存格往右算, 每個表格列的值最多為 **count** 個。
TOTAL	如果表格為單一維度或 **TOTAL** 限定詞作為引數, 則目前資料行區段一律等於整個資料行。

會對列區段的最後一個資料行傳回 NULL, 因為在此資料行之後沒有任何資料行。

如果樞紐分析表有多個水平維度, 則目前列區段僅會包括在所有維度列中與目前資料行有相同值的資料行 (顯示欄位間排序順序之最後一個水平維度的列除外)。樞紐分析表中水平維度的欄位間排序順序, 可簡單透過維度從上至下的順序來定義。.

範例:

```
after( sum( Sales ))
after( sum( Sales ), 2 )
after( total sum( Sales ))
```

rangeavg (after(sum(x),1,3)) 會根據緊接在目前資料行右方的三個資料行, 傳回所評估 **sum(x)** 函數之三個結果的平均值。

Before - 圖表函數

Before() 傳回以樞紐分析表的維度值評估而得的運算式值, 這些值會顯示在樞紐分析表的列區段內, 目前資料行之前的資料行中。

語法:

```
before([TOTAL] expr [, offset [, count]])
```

此函數在所有圖表類型中都會傳回 NULL, 除了樞紐分析表以外。

當任何圖表運算式使用了此圖表函數時, 就不允許在圖表中依據 Y 值排序, 或在表格中依據運算式資料行排序。因此, 這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數, 視覺化的排序將還原為對此函數的排序輸入。

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
offset	將一個 **offset** n, 指定為大於 1, 則可將運算式的評估從目前列往左移動 n 列。 若將位移指定為 0, 則會根據目前列來評估運算式。 若指定負值的位移, 會讓 **Before** 函數的作用相當於包含對應正值位移的 **After** 函數。

引數	描述
count	將第三個參數 **count** 指定為大於 1，函數會傳回一系列值，從原始儲存格往左算，每個表格列的值最多為 **count** 個。
TOTAL	如果表格為單一維度或 **TOTAL** 限定詞作為引數，則目前資料行區段一律等於整個資料行。

會對列區段的第一個資料行傳回 NULL 值，因為在此資料行之前沒有任何資料行。

如果樞紐分析表有多個水平維度，則目前列區段僅會包括在所有維度列中與目前資料行有相同值的資料行 (顯示欄位間排序順序之最後一個水平維度的列除外)。樞紐分析表中水平維度的欄位間排序順序，可簡單透過維度從上至下的順序來定義。.

範例：

```
before( sum( Sales ))
before( sum( Sales ), 2 )
before( total sum( Sales ))
```

`rangeavg (before(sum(x),1,3))` 會根據緊接在目前資料行左方的三個資料行，傳回所評估 **sum(x)** 函數之三個結果的平均值。

First - 圖表函數

First() 傳回以樞紐分析表的維度值評估而得的運算式值，這些值會顯示在樞紐分析表中目前列區段的第一個資料行中。此函數在所有圖表類型中都會傳回 NULL，除了樞紐分析表以外。

當任何圖表運算式使用了此圖表函數時，就不允許在圖表中依據 Y 值排序，或在表格中依據運算式資料行排序。因此，這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數，視覺化的排序將還原為對此函數的排序輸入。

語法：

```
first([TOTAL] expr [, offset [, count]])
```

引數：

引數

引數	描述
expression	包含待測量之資料的運算式或欄位。
offset	將一個 **offset** n，指定為大於 1，則可將運算式的評估從目前列往右移動 n 列。 若將位移指定為 0，則會根據目前列來評估運算式。 若指定負值的位移，會讓 **First** 函數的作用相當於包含對應正值位移的 **Last** 函數。
count	將第三個參數 **count** 指定為大於 1，函數會傳回一系列值，從原始儲存格往右算，每個表格列的值最多為 **count** 個。

引數	描述
TOTAL	如果表格為單一維度或 **TOTAL** 限定詞作為引數，則目前資料行區段一律等於整個資料行。

如果樞紐分析表有多個水平維度，則目前列區段僅會包括在所有維度列中與目前資料行有相同值的資料行 (顯示欄位間排序順序之最後一個水平維度的列除外)。樞紐分析表中水平維度的欄位間排序順序，可簡單透過維度從上至下的順序來定義。.

範例：

```
first( sum( Sales ))
first( sum( Sales ), 2 )
first( total sum( Sales )
```

`rangeavg (first(sum(x),1,5))` 會根據目前列區段的最左方五個資料行，傳回所評估 **sum(x)** 函數之結果的平均值。

Last - 圖表函數

Last() 傳回以樞紐分析表的維度值評估而得的運算式值，這些值會顯示在樞紐分析表中目前列區段的最後一個資料行中。此函數在所有圖表類型中都會傳回 NULL，除了樞紐分析表以外。

當任何圖表運算式使用了此圖表函數時，就不允許在圖表中依據 Y 值排序，或在表格中依據運算式資料行排序。因此，這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數，視覺化的排序將還原為對此函數的排序輸入。

語法：

```
last([TOTAL] expr [, offset [, count]])
```

引數：

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
offset	將一個 **offset** n，指定為大於 1，則可將運算式的評估從目前列往左移動 n 列。 若將位移指定為 0，則會根據目前列來評估運算式。 若指定負值的位移，會讓 **First** 函數的作用相當於包含對應正值位移的 **Last** 函數。
count	將第三個參數 **count** 指定為大於 1，函數會傳回一系列值，從原始儲存格往左算，每個表格列的值最多為 **count** 個。
TOTAL	如果表格為單一維度或 **TOTAL** 限定詞作為引數，則目前資料行區段一律等於整個資料行。

如果樞紐分析表有多個水平維度, 則目前列區段僅會包括在所有維度列中與目前資料行有相同值的資料行 (顯示欄位間排序順序之最後一個水平維度的列除外)。樞紐分析表中水平維度的欄位間排序順序, 可簡單透過維度從上至下的順序來定義。.

範例:

```
last( sum( Sales ))
last( sum( Sales ), 2 )
last( total sum( Sales )
```

`rangeavg (last(sum(x),1,5))` 會根據目前列區段的最右方五個資料行, 傳回所評估 **sum(x)** 函數之結果的平均值。

ColumnNo - 圖表函數

ColumnNo() 傳回樞紐分析表中目前列區段內, 目前資料行的編號。第一行的編號為 1。

語法:

```
ColumnNo([total])
```

引數:

引數

引數	描述
TOTAL	如果表格為單一維度或 **TOTAL** 限定詞作為引數, 則目前資料行區段一律等於整個資料行。

如果樞紐分析表有多個水平維度, 則目前列區段僅會包括在所有維度列中與目前資料行有相同值的資料行 (顯示欄位間排序順序之最後一個水平維度的列除外)。樞紐分析表中水平維度的欄位間排序順序, 可簡單透過維度從上至下的順序來定義。.

當任何圖表運算式使用了此圖表函數時, 就不允許在圖表中依據 Y 值排序, 或在表格中依據運算式資料行排序。因此, 這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數, 視覺化的排序將還原為對此函數的排序輸入。

範例:

```
if( ColumnNo( )=1, 0, sum( Sales ) / before( sum( Sales )))
```

NoOfColumns - 圖表函數

NoOfColumns() 傳回樞紐分析表中目前列區段中的資料行數。

當任何圖表運算式使用了此圖表函數時, 就不允許在圖表中依據 Y 值排序, 或在表格中依據運算式資料行排序。因此, 這些排序替代選項會自動停用。若您在視覺化或表格中使用此圖表函數, 視覺化的排序將還原為對此函數的排序輸入。

語法:

```
NoOfColumns([total])
```

引數:

引數

引數	描述
TOTAL	如果表格為單一維度或 **TOTAL** 限定詞作為引數,則目前資料行區段一律等於整個資料行。

如果樞紐分析表有多個水平維度,則目前列區段僅會包含在所有維度列中與目前資料行有相同值的資料行,顯示欄位間排序順序中最後一個維度的列除外。樞紐分析表中水平維度的欄位間排序順序,可簡單透過維度從上至下的順序來定義。.

範例:

```
if( ColumnNo( )=NoOfColumns( ), 0, after( sum( Sales )))
```

8.17 邏輯函數

本節說明處理邏輯操作的函數。所有函數皆可用於資料載入指令碼和圖表運算式。

IsNum

如果運算式可解譯為數字,會傳回 -1 (True),否則會傳回 0 (False)。

```
IsNum( expr )
```

IsText

如果運算式有文字表示法,會傳回 -1 (True),否則會傳回 0 (False)。

```
IsText( expr )
```

*如果運算式為 NULL,**IsNum** 和 **IsText** 都會傳回 0。*

範例:

下列範例載入具有混合文字和數值的內嵌表格,並新增兩個欄位以檢查值是否分別為數值和文字值。

```
Load *, IsNum(Value), IsText(Value)
Inline [
Value
23
Green
Blue
12
33Red];
```

產生的表格如下所示:

Resulting table

Value	IsNum(Value)	IsText(Value)
23	-1	0
Green	0	-1
Blue	0	-1
12	-1	0
33Red	0	-1

8.18 對應函數

本節說明用來處理對應表格的函數。對應表格可以用來在執行指令碼期間取代欄位值或欄位名稱。

對應函數只能在資料載入指令碼中使用。

對應函數概述

概述之後,會進一步描述每個函數。您還可以在語法中按一下函數名稱,以立即存取該特定函數的詳細資料。

ApplyMap

ApplyMap 指令碼函數用於將某個運算式的輸出對應到先前載入的對應表。

```
ApplyMap ('mapname', expr [ , defaultexpr ] )
```

MapSubstring

MapSubstring 指令碼函數是用於將任何運算式的部分對應到先前載入的對應表。對應會區分大小寫且不反覆運算,並會從左至右對應子字串。

```
MapSubstring ('mapname', expr)
```

ApplyMap

ApplyMap 指令碼函數用於將某個運算式的輸出對應到先前載入的對應表。

語法:

```
ApplyMap('map_name', expression [ , default_mapping ] )
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
map_name	先前已透過 **mapping load** 或 **mapping select** 陳述式建立之對應表格的名稱。其名稱必須以一般單引號括住。 *若您在巨集擴展變數中使用此函數,並參考不存在的對應表格,則函數叫用會失敗,且不會建立欄位。*
expression	應該對應其結果的運算式。
default_ mapping	如果指定,則當對應表格未包含 expression 的相符值時,此值將用作預設值。如果未指定,則將按原樣傳回 expression 的值。

ApplyMap 輸出欄位的名稱不可與其中一個輸出欄位的名稱相同。這可能會造成意外結果。不使用的範例:`ApplyMap('Map', A) as A`。

範例:

在此範例中,我們載入銷售人員清單,並使用國家/地區代碼表示其居住國家/地區。我們使用表格將國家/地區代碼對應至國家/地區,將國家/地區代碼取代為國家/地區名稱。對應表格中僅定義三個國家/地區,其他國家/地區代碼對應至 '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';
```

產生的表格 (銷售人員) 如下所示:

Resulting table

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

MapSubstring

MapSubstring 指令碼函數是用於將任何運算式的部分對應到先前載入的對應表。對應會區分大小寫且不反覆運算, 並會從左至右對應子字串。

語法:

```
MapSubstring('map_name', expression)
```

傳回的資料類型: 字串

引數:

引數

引數	描述
map_name	之前在 **mapping load** 或 **mapping select** 陳述式中讀取的對應表格名稱。該名稱必須以一般單引號括住。 *若您在巨集擴展變數中使用此函數, 並參考不存在的對應表格, 則函數叫用會失敗, 且不會建立欄位。*
expression	應由子字串對應其結果的運算式。

範例:

在此範例中,我們載入產品型號清單。每一個型號都具有一組由複合代碼描述的屬性。搭配使用對應表格與 MapSubstring,我們可以將屬性代碼展開為描述。

```
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';
```

產生的表格如下所示:

Resulting table

Model	**Description**
Twixie	Red Cotton Small
Boomer	Blue Polyester Large
Raven	Yellow Polyester Medium
Seedling	Red Cotton Large
SeedlingPlus	Red Cotton Large with hood
Younger	Blue Cotton with patch
MultiStripe	Red Yellow Blue Cotton Small/Medium/Large

8.19 數學函數

本節說明數學常數和布林值函數。這些函數沒有任何參數，但是仍需要括弧。

所有函數皆可用於資料載入指令碼和圖表運算式。

e

該函數傳回自然對數 **e** (2.71828...) 的基數。

```
e( )
```

false

該函數傳回由文字值 'False' 與數值 0 組成的雙值，這可用來表示運算式的邏輯為 false。

```
false( )
```

pi

該函數傳回值 π (3.14159...)。

```
pi( )
```

rand

該函數傳回介於 0 與 1 之間的隨機數字。這可用來建立樣本資料。

```
rand( )
```

範例：

此範例指令碼會建立包含 1000 筆記錄的表格，並隨機選取大寫字元，即在範圍 65 至 91 (65+26) 中的字元。

```
Load
    Chr( Floor(rand() * 26) + 65) as UCaseChar,
    RecNo() as ID
    Autogenerate 1000;
```

true

該函數傳回由文字值 'True' 與數值 -1 組成的雙值，這可用來表示運算式的邏輯為 true。

```
true( )
```

8.20 NULL 函數

本節描述用來傳回或偵測 NULL 值的函數。

所有函數皆可用於資料載入指令碼和圖表運算式。

NULL 函數概述

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

EmptyIsNull

EmptyIsNull 函數將空的字串轉換為 NULL。因此，若參數是空的字串，這會傳回 NULL，否則會傳回參數。

```
EmptyIsNull (expr )
```

IsNull

IsNull 函數會檢測運算式的值是否為 NULL，如果是的話，會傳回 -1 (True)，否則會傳回 0 (False)。

```
IsNull (expr )
```

Null

Null 函數會傳回 NULL 值。

```
NULL( )
```

EmptyIsNull

EmptyIsNull 函數將空的字串轉換為 NULL。因此，若參數是空的字串，這會傳回 NULL，否則會傳回參數。

語法：

```
EmptyIsNull(exp )
```

範例與結果：

指令碼處理範例

範例	結果
`EmptyIsNull(AdditionalComments)`	此運算式將會以 NULL 傳回 *AdditionalComments* 欄位中任何空的字串值，而非傳回空的字串。會傳回非空字串和數字。
`EmptyIsNull(PurgeChar (PhoneNumber, ' -()'))`	此運算式將會從 *PhoneNumber* 欄位移除任何破折號、空格和括弧。若沒有剩下任何字元，EmptyIsNull 函數會以 NULL 傳回空的字串；空的電話號碼與無電話號碼相同。

IsNull

IsNull 函數會檢測運算式的值是否為 NULL，如果是的話，會傳回 -1 (True)，否則會傳回 0 (False)。

語法：

```
IsNull(expr )
```

*長度為 0 的字串不會視為 NULL，並將導致 **IsNull** 傳回 False。*

範例: 資料載入指令碼

在此範例中, 會載入含有四列的內嵌表格, 其中前三行不包含任何內容, 或是在 Value 資料行中包含 - 或 'NULL'。我們使用 **Null** 函數, 將這些值轉換為具有中間前置 **LOAD** 的真正NULL 值表示法。

第一個前置 **LOAD** 新增一個欄位, 使用 **IsNull** 函數檢查值是否為 NULL。

```
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];
```

這是產生的表格。在 ValueNullConv 資料列中, NULL 值由 - 代表。

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

NULL

Null 函數會傳回 NULL 值。

語法:

```
Null( )
```

範例: 資料載入指令碼

在此範例中, 會載入含有四列的內嵌表格, 其中前三行不包含任何內容, 或是在 Value 資料行中包含 - 或 'NULL'。您想要將這些值轉換為真正NULL 值表示法。

這個中間前置 **LOAD** 使用 **Null** 函數執行轉換。

第一個前置 **LOAD** 會新增一個欄位, 僅針對此範例中的圖例, 檢查值是否為 NULL。

```
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];
```

這是產生的表格。在 ValueNullConv 資料列中，NULL 值由 - 代表。

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

8.21 範圍函數

範圍函數是一種使用值陣列，產生單一值作為結果的函數。所有範圍函數皆可用於資料載入指令碼和圖表運算式。

例如，在視覺化中，範圍函數可從記錄間陣列計算出單一值。在資料載入指令碼中，範圍函數可從內部表格中的值陣列計算出單一值。

*範圍函數可取代以下一般數值函數：**numsum**、**numavg**、**numcount**、**nummin** 和 **nummax**，這些函數現應被視為已過時。*

基本範圍函數

RangeMax

RangeMax() 傳回運算式或欄位中找到的最高數值。

```
RangeMax (first_expr[, Expression])
```

RangeMaxString

RangeMaxString() 傳回運算式或欄位中找到的文字排序順序中的最後一個值。

```
RangeMaxString (first_expr[, Expression])
```

RangeMin

RangeMin() 傳回運算式或欄位內找到的最低數值。

```
RangeMin (first_expr[, Expression])
```

RangeMinString

RangeMinString() 傳回運算式或欄位中找到的文字排序順序中的第一個值。

```
RangeMinString (first_expr[, Expression])
```

RangeMode

RangeMode() 尋找運算式或欄位中最常發生的值 (模式值)。

```
RangeMode (first_expr[, Expression])
```

RangeOnly

RangeOnly() 是一種 dual 函數，如果運算式評估為一個唯一的值，則傳回某個值。如果不是，則傳回 **NULL**。

```
RangeOnly (first_expr[, Expression])
```

RangeSum

RangeSum() 傳回值的範圍總和。所有非數值作為 0 處理。

```
RangeSum (first_expr[, Expression])
```

計數器範圍函數

RangeCount

RangeCount() 傳回運算式或欄位中值 (文字與數值) 的數目。

```
RangeCount (first_expr[, Expression])
```

RangeMissingCount

RangeMissingCount() 傳回運算式或欄位中的非數值 (包括 NULL) 的數目。

```
RangeMissingCount (first_expr[, Expression])
```

RangeNullCount

RangeNullCount() 尋找運算式或欄位中的 NULL 值數目。

```
RangeNullCount (first_expr[, Expression])
```

RangeNumericCount

RangeNumericCount() 尋找運算式或欄位中的數值數目。

```
RangeNumericCount (first_expr[, Expression])
```

RangeTextCount

RangeTextCount() 傳回運算式或欄位中的文字值數目。

```
RangeTextCount (first_expr[, Expression])
```

統計範圍函數

RangeAvg

RangeAvg() 傳回範圍的平均值。函數的輸入可能是一定範圍內的值或運算式。

```
RangeAvg (first_expr[, Expression])
```

RangeCorrel

RangeCorrel() 傳回兩組資料的相關係數。相關係數是資料集之間關係的量值。

```
RangeCorrel (x_values , y_values[, Expression])
```

RangeFractile

RangeFractile() 傳回對應於一定範圍數字中第 n 個 **fractile** (分位數) 的值。

```
RangeFractile (fractile, first_expr[ ,Expression])
```

RangeKurtosis

RangeKurtosis() 傳回對應於一定範圍數字中 kurtosis 的值。

```
RangeKurtosis (first_expr[, Expression])
```

RangeSkew

RangeSkew() 傳回對應於一定範圍數字中 skewness 的值。

```
RangeSkew (first_expr[, Expression])
```

RangeStdev

RangeStdev() 尋找一定範圍數字的標準差。

```
RangeStdev (expr1[, Expression])
```

財務範圍函數

RangeIRR

RangeIRR() 會針對輸入值所代表的一組現金流量，傳回內部報酬率。

```
RangeIRR (value[, value][, Expression])
```

RangeNPV

RangeNPV() 會根據折扣率以及一系列未來定期支出 (負值) 和收入 (正值)，傳回投資的淨現值。結果具有**money**的預設數字格式。

```
RangeNPV (discount_rate, value[, value][, Expression])
```

RangeXIRR

RangeXIRR() 會針對不一定為週期性的現金流量表，傳回內部報酬率 (每年)。若要計算一系列週期性現金流量的內部報酬率，請使用 **RangeIRR** 函數。

```
RangeXIRR (values, dates[, Expression])
```

RangeXNPV

RangeXNPV() 會針對運算式的配對數字 (以 **pmt** 和 **date** 表示) 所代表的現金流量排程 (不一定是定期) 傳回淨現值。所有支出均按照 1 年 365 天攤算。

```
RangeXNPV (discount_rate, values, dates[, Expression])
```

另請參見:

記錄間函數 (page 1199)

RangeAvg

RangeAvg() 傳回範圍的平均值。函數的輸入可能是一定範圍內的值或運算式。

語法:

```
RangeAvg(first_expr[, Expression])
```

傳回的資料類型: 數值

引數:

此函數的引數可能包含記錄間函數, 它們會自行傳回值的清單。

引數

引數	描述
first_expr	包含待測量之資料的運算式或欄位。
Expression	包含待測量之資料範圍的選用運算式或欄位。

限制:

如果找不到數值, 則會傳回 NULL。

範例與結果:

指令碼處理範例

範例	結果
`RangeAvg (1,2,4)`	傳回 2.33333333
`RangeAvg (1,'xyz')`	傳回 1
`RangeAvg (null( ), 'abc')`	傳回 NULL

範例:

將範例指令碼新增至您的應用程式並予以執行。若要查看結果, 將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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
];
```

產生的表格顯示表格中各記錄的 MyRangeAvg 傳回值。

結果表格

RangeID	MyRangeAvg
1	7
2	4
3	6
4	12.666
5	6.333
6	5

運算式範例:

```
RangeAvg (Above(MyField),0,3))
```

傳回 **MyField** 針對目前列和目前列上方兩列, 計算出三個值之範圍結果的滑動平均。若將第三個引數指定為 3, 則 **Above()** 函數會傳回三個值 (上方需有足夠的列), 並做為 **RangeAvg()** 函數的輸入。

範例中使用的資料:

*停用 **MyField** 的排序, 確保範例如預期運作。*

樣本資料

MyField	RangeAvg (Above(MyField,0,3))	Comments
10	10	因為這是頂端列, 故範圍只包含一個值。
2	6	此列上方只有一列, 故範圍是: 10,2.
8	6.6666666667	相當於 RangeAvg(10,2,8)
18	9.333333333	-
5	10. 333333333	-
9	10.6666666667	-

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

另請參見：

- Avg - 圖表函數 (page 378)
- Count - 圖表函數 (page 332)

RangeCorrel

RangeCorrel() 傳回兩組資料的相關係數。相關係數是資料集之間關係的量值。

語法：

```
RangeCorrel(x_value , y_value[, Expression])
```

傳回的資料類型： 數值

資料序列應成對輸入 (x,y)。例如，若要評估陣列 1 和陣列 2 這兩個資料序列，其中陣列 1 = 2,6,9 而陣列 2 = 3,8,4，您可編寫為 `RangeCorrel (2,3,6,8,9,4)`，它會傳回 0.269。

引數：

引數

引數	描述
x-value, y-value	每個值代表第三個選用參數的記錄間函數所傳回之單一值或一個範圍的值。每個值或每個範圍的值都必須對應於一個 **x-value** 或一個範圍的 **y-values**。
Expression	包含待測量之資料範圍的選用運算式或欄位。

限制：

此函數需要至少兩對座標才能計算。

文字值、NULL 值和遺漏值會傳回 NULL。

範例與結果：

函數範例

範例	結果
`RangeCorrel (2,3,6,8,9,4,8,5)`	傳回 0.2492。此函數可在指令碼中載入，或在運算式編輯器中新增至視覺化。

範例：

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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 '|');
```

在含有 ID1 作為維度和量值 RangeCorrel(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6)) 的表格中，**RangeCorrel()** 函數可針對每個 ID1 值在六對 x,y 的範圍中找到 **Correl** 值。

結果表格

ID1	MyRangeCorrel
01	-0.9517
02	-0.5209
03	-0.5209
04	-0.1599

範例：

```
XY:
LOAD recno() as RangeID, * Inline [
X|Y
2|3
6|8
9|4
8|5
](delimiter is '|');
```

在具有 RangeID 作為維度和量值的表格中：RangeCorrel(Below(X,0,4,BelowY,0,4))，**RangeCorrel()** 函數使用 **Below()** 函數的結果，由於第三個引數 (count) 設為 4，因此該結果從載入表格 XY 中產生一系列的四個 x-y 值。

結果表格

RangeID	MyRangeCorrel2
01	0.2492
02	-0.9959
03	-1.0000
04	-

RangeID 01 的值與手動輸入 RangeCorrel(2,3,6,8,9,4,8,5) 的值相同。對於 RangeID 的其他值，Below() 函數產生的序列為：(6,8,9,4,8,5)、(9,4,8,5) 和 (8,5)，最後一個產生 Null 值結果。

另請參見：

Correl - 圖表函數 (page 381)

RangeCount

RangeCount() 傳回運算式或欄位中值 (文字與數值) 的數目。

語法：

```
RangeCount(first_expr[, Expression])
```

傳回的資料類型： 整數

引數：

此函數的引數可能包含記錄間函數，它們會自行傳回值的清單。

引數

引數	描述
first_expr	包含待計算資料的運算式或欄位。
Expression	包含待計算資料範圍的選用運算式或欄位。

限制：

NULL 值不在計算範圍內。

範例與結果：

函數範例

範例	結果
`RangeCount (1,2,4)`	傳回 3
`RangeCount (2,'xyz')`	傳回 2
`RangeCount (null( ))`	傳回 0
`RangeCount (2,'xyz', null())`	傳回 2

範例：

將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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
];
```

產生的表格顯示表格中各記錄的 MyRangeCount 傳回值。

結果表格

RangeID	MyRangeCount
1	3
2	3
3	3
4	3
5	3
6	3

運算式範例:

```
RangeCount (Above(MyField,1,3))
```

傳回 **MyField** 的三個結果中包含的值數目。若將 **Above()** 函數的第一個引數指定為 1, 並將第二個引數指定為 3, 該函數會從目前列上方的前三個欄位傳回值 (上方需有足夠的列), 並作為 **RangeCount ()** 函數的輸入。

範例中使用的資料:

樣本資料

MyField	RangeCount(Above(MyField,1,3))
10	0
2	1
8	2
18	3
5	3
9	3

範例中使用的資料:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
```

```
5
9
] ;
```

另請參見:

Count - 圖表函數 (page 332)

RangeFractile

RangeFractile() 傳回對應於一定範圍數字中第 n 個 **fractile** (分位數) 的值。

計算分位數時, RangeFractile() 使用最接近排名的線性解譯。

語法:

```
RangeFractile(fractile, first_expr[, Expression])
```

傳回的資料類型: 數值

引數:

此函數的引數可能包含記錄間函數, 它們會自行傳回值的清單。

引數

引數	描述
fractile	介於 0 與 1 之間的數字對應於要計算的分位數 (以分數表示的分位數)。
first_expr	包含待測量之資料的運算式或欄位。
Expression	包含待測量之資料範圍的選用運算式或欄位。

範例與結果:

函數範例

範例	結果
`RangeFractile (0.24,1,2,4,6)`	傳回 1.72
`RangeFractile(0.5,1,2,3,4,6)`	傳回 3
`RangeFractile (0.5,1,2,5,6)`	傳回 3.5

範例:

將範例指令碼新增至您的應用程式並予以執行。若要查看結果, 將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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
];
```

產生的表格顯示表格中各記錄的 MyRangeFrac 傳回值。

結果表格

RangeID	MyRangeFrac
1	6
2	3
3	8
4	11
5	5
6	4

運算式範例:

```
RangeFractile (0.5, Above(Sum(MyField),0,3))
```

在此範例中,記錄間函數 **Above()** 包含選用的 offset 和 count 引數。這會產生一個結果範圍,可作為任何範圍函數的輸入使用。在這種情況下,`Above(Sum(MyField),0,3)` 會傳回目前列和目前列上方兩列的 `MyField` 值。這些值提供 **RangeFractile()** 函數的輸入。因此,對於以下表格的底端列,這是 `RangeFractile(0.5, 3,4,6)` 的對應,亦即計算數列 3、4 和 6 的 0.5 分位數。在以下表格的前兩列,範圍內數值會相應減少,其中不會有任何列位於目前列之上。其他記錄間函數會產生類似結果。

樣本資料

MyField	RangeFractile(0.5, Above(Sum(MyField),0,3))
1	1
2	1.5
3	2
4	3
5	4
6	5

範例中使用的資料:

```
RangeTab:
LOAD * INLINE [
MyField
1
```

```
2
3
4
5
6
] ;
```

另請參見:

- *Above - 圖表函數 (page 1202)*
- *Fractile - 圖表函數 (page 384)*

RangeIRR

RangeIRR() 會針對輸入值所代表的一組現金流量，傳回內部報酬率。

內部報酬率是對於定期支出 (負值) 和收入 (正值) 的投資所收到的利率。

此函數使用 Newton 方法的簡化版本來計算內部報酬率 (IRR)。

語法:

```
RangeIRR(value[, value][, Expression])
```

傳回的資料類型: 數值

引數

引數	描述
value	帶第三個選用參數的記錄間函數所傳回之單一值或一個範圍的值。此函數需要至少一個正值和一個負值才能計算。
Expression	包含待測量之資料範圍的選用運算式或欄位。

限制:

文字值、NULL 值和遺漏值將予以忽略。

範例表格

範例	結果
`RangeIRR(-70000,12000,15000,18000,21000,26000)`	傳回 0.0866

<table>
<tr><th>範例</th><th>結果</th></tr>
<tr><td>將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

<pre>
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 '|');
</pre></td>
<td>產生的表格顯示表格中各記錄的 RangeIRR 傳回值。

<table>
<tr><th>RangeID</th><th>RangeIRR</th></tr>
<tr><td>1</td><td>0.0639</td></tr>
<tr><td>2</td><td>0.8708</td></tr>
<tr><td>3</td><td>-</td></tr>
<tr><td>4</td><td>5.8419</td></tr>
<tr><td>5</td><td>0.9318</td></tr>
<tr><td>6</td><td>-0.2566</td></tr>
</table></td></tr>
</table>

另請參見：

記錄間函數 (page 1199)

RangeKurtosis

RangeKurtosis() 傳回對應於一定範圍數字中 kurtosis 的值。

語法：

```
RangeKurtosis(first_expr[, Expression])
```

傳回的資料類型： 數值

引數：

此函數的引數可能包含記錄間函數，它們會自行傳回值的清單。

引數

引數	描述
first_expr	包含待測量之資料的運算式或欄位。
Expression	包含待測量之資料範圍的選用運算式或欄位。

限制:

如果找不到數值，則會傳回 NULL。

範例與結果:

函數範例

範例	結果
`RangeKurtosis (1,2,4,7)`	傳回 -0.28571428571429

另請參見:

Kurtosis - 圖表函數 (page 391)

RangeMax

RangeMax() 傳回運算式或欄位中找到的最高數值。

語法:

```
RangeMax(first_expr[, Expression])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
first_expr	包含待測量之資料的運算式或欄位。
Expression	包含待測量之資料範圍的選用運算式或欄位。

限制:

如果找不到數值，則會傳回 NULL。

範例與結果:

函數範例

範例	結果
RangeMax (1,2,4)	傳回 4
RangeMax (1,'xyz')	傳回 1
RangeMax (null( ), 'abc')	傳回 NULL

範例:

將範例指令碼新增至您的應用程式並予以執行。若要查看結果,將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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
];
```

產生的表格顯示表格中各記錄的 MyRangeMax 傳回值。

結果表格

RangeID	MyRangeMax
1	10
2	7
3	8
4	18
5	9
6	9

運算式範例:

```
RangeMax (Above(MyField,0,3))
```

傳回 **MyField** 針對目前列和目前列上方兩列,所計算出的三個值中最大的值。若將第三個引數指定為 3,則 **Above()** 函數會傳回三個值 (上方需有足夠的列),並做為 **RangeMax()** 函數的輸入。

範例中使用的資料:

*停用 **MyField** 的排序，確保範例如預期運作。*

樣本資料

MyField	RangeMax (Above(Sum(MyField),1,3))
10	10
2	10
8	10
18	18
5	18
9	18

範例中使用的資料：

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

RangeMaxString

RangeMaxString() 傳回運算式或欄位中找到的文字排序順序中的最後一個值。

語法：

```
RangeMaxString(first_expr[, Expression])
```

傳回的資料類型： 字串

引數：

此函數的引數可能包含記錄間函數，它們會自行傳回值的清單。

引數

引數	描述
first_expr	包含待測量之資料的運算式或欄位。
Expression	包含待測量之資料範圍的選用運算式或欄位。

範例與結果:

函數範例

範例	結果
`RangeMaxString (1,2,4)`	傳回 4
`RangeMaxString ('xyz','abc')`	傳回 'xyz'
`RangeMaxString (5,'abc')`	傳回 'abc'
`RangeMaxString (null( ))`	傳回 NULL

運算式範例:

```
RangeMaxString (Above(MaxString(MyField),0,3))
```

傳回 **MaxString(MyField)** 函數針對目前列和目前列上方兩列，所評估的三個結果中最後一項 (依文字排列順序)。

範例中使用的資料:

*停用 **MyField** 的排序，確保範例如預期運作。*

樣本資料

MyField	RangeMaxString(Above(MaxString(MyField),0,3))
10	10
abc	abc
8	abc
def	def
xyz	xyz
9	xyz

範例中使用的資料:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

另請參見:

- *MaxString - 圖表函數 (page 501)*

RangeMin

RangeMin() 傳回運算式或欄位內找到的最低數值。

語法:

```
RangeMin(first_expr[, Expression])
```

傳回的資料類型: 數值

引數:

引數

引數	描述
first_expr	包含待測量之資料的運算式或欄位。
Expression	包含待測量之資料範圍的選用運算式或欄位。

限制:

如果找不到數值,則會傳回 NULL。

範例與結果:

函數範例

範例	結果
RangeMin (1,2,4)	傳回 1
RangeMin (1,'xyz')	傳回 1
RangeMin (null( ), 'abc')	傳回 NULL

範例:

將範例指令碼新增至您的應用程式並予以執行。若要查看結果,將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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
];
```

產生的表格顯示表格中各記錄的 MyRangeMin 傳回值。

結果表格

RangeID	**MyRangeMin**
1	5
2	2
3	2
4	9
5	5
6	2

運算式範例:

```
RangeMin (Above(MyField,0,3)
```

傳回 **MyField** 針對目前列和目前列上方兩列, 所計算出的三個值中最小的值。若將第三個引數指定為 3, 則 **Above()** 函數會傳回三個值 (上方需有足夠的列), 並做為 **RangeMin()** 函數的輸入。

範例中使用的資料:

樣本資料

MyField	**RangeMin(Above(MyField,0,3))**
10	10
2	2
8	2
18	2
5	5
9	5

範例中使用的資料:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

另請參見:

Min - 圖表函數 (page 320)

RangeMinString

RangeMinString() 傳回運算式或欄位中找到的文字排序順序中的第一個值。

語法:

```
RangeMinString(first_expr[, Expression])
```

傳回的資料類型: 字串

引數:

此函數的引數可能包含記錄間函數，它們會自行傳回值的清單。

引數

引數	描述
first_expr	包含待測量之資料的運算式或欄位。
Expression	包含待測量之資料範圍的選用運算式或欄位。

範例與結果:

函數範例

範例	結果
`RangeMinString (1,2,4)`	傳回 1
`RangeMinString ('xyz','abc')`	傳回 'abc'
`RangeMinString (5,'abc')`	傳回 5
`RangeMinString (null( ))`	傳回 NULL

運算式範例:

```
RangeMinString (Above(MinString(MyField),0,3))
```

傳回 **MinString(MyField)** 函數針對目前列和目前列上方兩列，所評估的三個結果中第一項 (依文字排列順序)。

範例中使用的資料:

*停用 **MyField** 的排序，確保範例如預期運作。*

樣本資料

MyField	RangeMinString(Above(MinString(MyField),0,3))
10	10
abc	10

MyField	RangeMinString(Above(MinString(MyField),0,3))
8	8
def	8
xyz	8
9	9

範例中使用的資料:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

另請參見:

MinString - 圖表函數 (page 504)

RangeMissingCount

RangeMissingCount() 傳回運算式或欄位中的非數值 (包括 NULL) 的數目。

語法:

```
RangeMissingCount(first_expr[, Expression])
```

傳回的資料類型: 整數

引數:

此函數的引數可能包含記錄間函數, 它們會自行傳回值的清單。

引數

引數	描述
first_expr	包含待計算資料的運算式或欄位。
Expression	包含待計算資料範圍的選用運算式或欄位。

範例與結果:

函數範例

範例	結果
`RangeMissingCount (1,2,4)`	傳回 0
`RangeMissingCount (5,'abc')`	傳回 1
`RangeMissingCount (null( ))`	傳回 1

運算式範例:

`RangeMissingCount (Above(MinString(MyField),0,3))`

傳回 **MinString(MyField)** 函數針對目前列和目前列上方兩列, 所評估的三個結果中非數值的數量。

*停用 **MyField** 的排序, 確保範例如預期運作。*

樣本資料

MyField	RangeMissingCount (Above(MinString (MyField),0,3))	Explanation
10	2	傳回 2, 因為此列上方沒有任何列, 所以 3 個值中的 2 個值遺失。
abc	2	傳回 2, 因為目前列上方只有一列, 且目前列非數值 ('abc')。
8	1	傳回 1, 因為 3 列中有 1 列包含非數值 ('abc')。
def	2	傳回 2, 因為 3 列中有 2 列包含非數值 ('def' 和 'abc')。
xyz	2	傳回 2, 因為 3 列中有 2 列包含非數值 (' xyz' 和 'def')。
9	2	傳回 2, 因為 3 列中有 2 列包含非數值 (' xyz' 和 'def')。

範例中使用的資料:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

另請參見:

- *MissingCount - 圖表函數 (page 335)*

RangeMode

RangeMode() 尋找運算式或欄位中最常發生的值 (模式值)。

語法:

```
RangeMode(first_expr {, Expression})
```

傳回的資料類型: 數值

引數:

此函數的引數可能包含記錄間函數, 它們會自行傳回值的清單。

引數

引數	描述
first_expr	包含待測量之資料的運算式或欄位。
Expression	包含待測量之資料範圍的選用運算式或欄位。

限制:

如果不只一個值有最高的出現頻率, 則會傳回 NULL。

範例與結果:

函數範例

範例	結果
`RangeMode (1,2,9,2,4)`	傳回 2
`RangeMode ('a',4,'a',4)`	傳回 NULL
`RangeMode (null( ))`	傳回 NULL

範例:

將範例指令碼新增至您的應用程式並予以執行。若要查看結果, 將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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
];
```

產生的表格顯示表格中各記錄的 **MyRangeMode** 傳回值。

結果表格

RangeID	MyRangMode
1	-
2	-
3	8
4	-
5	5
6	-

運算式範例：

```
RangeMode (Above(MyField,0,3))
```

傳回 **MyField** 針對目前列和目前列上方兩列，所評估的三個結果中最常出現的值。若將第三個引數指定為 3，則 **Above()** 函數會傳回三個值 (上方需有足夠的列)，並做為 **RangeMode()** 函數的輸入。

範例中使用的資料：

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

*停用 **MyField** 的排序，確保範例如預期運作。*

樣本資料

MyField	RangeMode(Above(MyField,0,3))
10	傳回 10，因為上方沒有任何列，所以單一值是最常發生的值。
2	-
8	-
18	-
5	-
9	-

另請參見:

Mode - 圖表函數 (page 323)

RangeNPV

RangeNPV() 會根據折扣率以及一系列未來定期支出 (負值) 和收入 (正值), 傳回投資的淨現值。結果具有**money**的預設數字格式。

對於不一定是定期的現金流量表, 請參見 *RangeXNPV (page 1291)*。

語法:

```
RangeNPV(discount_rate, value[,value][, Expression])
```

傳回的資料類型: 數值

引數

引數	描述
discount_rate	各個期間的利率。
value	每期期末時發生的支出或所得。每個值可能是第三個選用參數的記錄間函數所傳回之單一值或一個範圍的值。
Expression	包含待測量之資料範圍的選用運算式或欄位。

限制:

文字值、NULL 值和遺漏值將予以忽略。

範例	結果
RangeNPV(0.1,-10000,3000,4200,6800)	傳回 1188.44

<table>
<tr><th>範例</th><th>結果</th></tr>
<tr><td>將範例指令碼新增至您的應用程式並予以執行。若要查看結果，將結果資料行中列出的欄位新增至您應用程式中的工作表。

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 '|');</td><td>產生的表格顯示表格中各記錄的 RangeNPV 傳回值。

RangeID RangeNPV
1 $-49.13
2 $777.78
3 $98.77
4 $25.51
5 $250.83
6 $20.40</td></tr>
</table>

另請參見：

記錄間函數 (page 1199)

RangeNullCount

RangeNullCount() 尋找運算式或欄位中的 NULL 值數目。

語法：

```
RangeNullCount(first_expr [, Expression])
```

傳回的資料類型： 整數

引數：

此函數的引數可能包含記錄間函數，它們會自行傳回值的清單。

引數

引數	描述
first_expr	包含待測量之資料的運算式或欄位。
Expression	包含待測量之資料範圍的選用運算式或欄位。

範例與結果:

函數範例

範例	結果
`RangeNullCount (1,2,4)`	傳回 0
`RangeNullCount (5,'abc')`	傳回 0
`RangeNullCount (null( ), null( ))`	傳回 2

運算式範例:

`RangeNullCount (Above(Sum(MyField),0,3))`

傳回 **Sum(MyField)** 函數針對目前列和目前列上方兩列,所評估的三個結果中 NULL 值的數量。

*在下方範例中複製 **MyField** 不會產生 NULL 值。*

樣本資料

MyField	RangeNullCount(Above(Sum(MyField),0,3))
10	傳回 2,因為此列上方沒有任何列,所以 3 個值中的 2 個值遺失 (=NULL)。
'abc'	傳回 1,因為此列上方只有一列,所以 3 個值中的 1 個值遺失 (=NULL)。
8	傳回 0,因為三列都不是 NULL 值。

範例中使用的資料:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
] ;
```

另請參見:

NullCount - 圖表函數 (page 338)

RangeNumericCount

RangeNumericCount() 尋找運算式或欄位中的數值數目。

語法:

```
RangeNumericCount(first_expr[, Expression])
```

傳回的資料類型: 整數

引數:

此函數的引數可能包含記錄間函數,它們會自行傳回值的清單。

引數

引數	描述
first_expr	包含待測量之資料的運算式或欄位。
Expression	包含待測量之資料範圍的選用運算式或欄位。

範例與結果:

函數範例

範例	結果
`RangeNumericCount (1,2,4)`	傳回 3
`RangeNumericCount (5,'abc')`	傳回 1
`RangeNumericCount (null( ))`	傳回 0

運算式範例:

```
RangeNumericCount (Above(MaxString(MyField),0,3))
```

傳回 **MaxString(MyField)** 函數針對目前列和目前列上方兩列,所評估的三個結果中數值的數量。

*停用 **MyField** 的排序,確保範例如預期運作。*

樣本資料

MyField	RangeNumericCount(Above(MaxString(MyField),0,3))
10	1
abc	1
8	2
def	1
xyz	1
9	1

範例中使用的資料:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
def
xyz
9
] ;
```

另請參見:

NumericCount - 圖表函數 (page 341)

RangeOnly

RangeOnly() 是一種 dual 函數,如果運算式評估為一個唯一的值,則傳回某個值。如果不是,則傳回 **NULL**。

語法:

```
RangeOnly(first_expr[, Expression])
```

傳回的資料類型: 雙值

引數:

此函數的引數可能包含記錄間函數,它們會自行傳回值的清單。

引數	描述
first_expr	包含待測量之資料的運算式或欄位。
Expression	包含待測量之資料範圍的選用運算式或欄位。

範例與結果:

範例	結果
`RangeOnly (1,2,4)`	傳回 NULL
`RangeOnly (5,'abc')`	傳回 NULL
`RangeOnly (null( ), 'abc')`	傳回 'abc'
`RangeOnly(10,10,10)`	傳回 10

另請參見:

Only - 圖表函數 (page 325)

RangeSkew

RangeSkew() 傳回對應於一定範圍數字中 skewness 的值。

語法:

```
RangeSkew(first_expr[, Expression])
```

傳回的資料類型: 數值

引數:

此函數的引數可能包含記錄間函數, 它們會自行傳回值的清單。

引數

引數	描述
first_expr	包含待測量之資料的運算式或欄位。
Expression	包含待測量之資料範圍的選用運算式或欄位。

限制:

如果找不到數值, 則會傳回 NULL。

範例與結果:

函數範例

範例	結果
`rangeskew (1,2,4)`	傳回 0.93521952958283
`rangeskew (above (SalesValue,0,3))`	傳回 above() 函數針對目前列和目前列上方兩列計算, 所傳回之三個值範圍的滑動偏態。

範例中使用的資料:

樣本資料

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
] ;
```

另請參見:

Skew - 圖表函數 (page 420)

RangeStdev

RangeStdev() 尋找一定範圍數字的標準差。

語法:

```
RangeStdev(first_expr[, Expression])
```

傳回的資料類型: 數值

引數:

此函數的引數可能包含記錄間函數, 它們會自行傳回值的清單。

引數

引數	描述
first_expr	包含待測量之資料的運算式或欄位。
Expression	包含待測量之資料範圍的選用運算式或欄位。

限制:

如果找不到數值, 則會傳回 NULL。

範例與結果:

函數範例

範例	結果
`RangeStdev (1,2,4)`	傳回 1.5275252316519
`RangeStdev (null( ))`	傳回 NULL

範例	結果
`RangeStdev (above (SalesValue),0,3))`	傳回 above() 函數針對目前列和目前列上方兩列計算, 所傳回的三個值範圍之滑動標準差。

範例中使用的資料:

樣本資料

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
] ;
```

另請參見:

Stdev - 圖表函數 (page 423)

RangeSum

RangeSum() 傳回值的範圍總和。所有非數值作為 0 處理。

語法:

```
RangeSum(first_expr[, Expression])
```

傳回的資料類型: 數值

引數:

此函數的引數可能包含記錄間函數, 它們會自行傳回值的清單。

引數

引數	描述
first_expr	包含待測量之資料的運算式或欄位。
Expression	包含待測量之資料範圍的選用運算式或欄位。

限制:

RangeSum 函數會將所有非數值視為 0。

範例與結果:

範例

範例	結果
`RangeSum (1,2,4)`	傳回 7
`RangeSum (5,'abc')`	傳回 5
`RangeSum (null( ))`	傳回 0

範例:

將範例指令碼新增至您的應用程式並予以執行。若要查看結果, 將結果資料行中列出的欄位新增至您應用程式中的工作表。

```
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
];
```

產生的表格顯示表格中各記錄的 MyRangeSum 傳回值。

結果表格

RangeID	MyRangeSum
1	21
2	12
3	18
4	38
5	19
6	15

運算式範例:

```
RangeSum (Above(MyField,0,3))
```

傳回 **MyField)** 三個值的加總:從目前列和目前列上方兩列。若將第三個引數指定為 3,則 **Above()** 函數會傳回三個值 (上方需有足夠的列),並做為 **RangeSum()** 函數的輸入。

範例中使用的資料:

*停用 **MyField** 的排序,確保範例如預期運作。*

樣本資料

MyField	RangeSum(Above(MyField,0,3))
10	10
2	12
8	20
18	28
5	31
9	32

範例中使用的資料:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

另請參見:

- *Sum - 圖表函數 (page 328)*
- *Above - 圖表函數 (page 1202)*

RangeTextCount

RangeTextCount() 傳回運算式或欄位中的文字值數目。

語法:

```
RangeTextCount(first_expr[, Expression])
```

傳回的資料類型: 整數

引數:

此函數的引數可能包含記錄間函數,它們會自行傳回值的清單。

引數

引數	描述
first_expr	包含待測量之資料的運算式或欄位。
Expression	包含待測量之資料範圍的選用運算式或欄位。

範例與結果:

函數範例

範例	結果
`RangeTextCount (1,2,4)`	傳回 0
`RangeTextCount (5,'abc')`	傳回 1
`RangeTextCount (null( ))`	傳回 0

運算式範例:

```
RangeTextCount (Above(MaxString(MyField),0,3))
```

傳回 **MaxString(MyField)** 函數針對目前列和目前列上方兩列,所評估的三個結果中文字值的數量。

範例中使用的資料:

*停用 **MyField** 的排序,確保範例如預期運作。*

範例資料

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

範例中使用的資料：

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
null()
'xyz'
9
] ;
```

另請參見：

TextCount - 圖表函數 (page 344)

RangeXIRR

RangeXIRR() 會針對不一定為週期性的現金流量表，傳回內部報酬率 (每年)。若要計算一系列週期性現金流量的內部報酬率，請使用 **RangeIRR** 函數。

Qlik 的 XIRR 功能 (**XIRR()** 和 **RangeXIRR()** 函數) 使用解開 `Rate` 值的下列方程式，以判定正確的 XIRR 值：

```
XNPV(Rate, pmt, date) = 0
```

會使用 Newton 方法的簡化版本解開方程式。

語法：

```
RangeXIRR(value, date{, value, date})
```

傳回的資料類型: 數值

引數

引數	描述
value	對應於多個日期支出排程的一筆現金流量或一系列現金流量。一組值中必須包含至少一個正值和一個負值。
date	對應於現金流量付款的支付日期或支付日期排程。

使用此函數時, 適用下列限制:

- 文字值、NULL 值和遺漏值將予以忽略。
- 所有支出均按照 1 年 365 天攤算。
- 此函數需要至少一個有效負值和至少一個有效正值付款 (連同對應的有效日期)。若未提供這些付款, 會傳回 NULL 值。

下列主題可能可協助您使用此函數:

- *RangeXNPV (page 1291)*: 使用此函數針對不一定為週期性的現金流排程, 計算淨現值。
- *XIRR (page 357)*:**XIRR()** 函數會針對現金流排程 (不一定為週期性), 計算彙總內部報酬率 (每年)。

在 Qlik Sense 用戶端管理 的不同版本, 此函數使用的基礎演算法有變化。如需關於演算法最近更新的資訊, 請參閱支援文章 [XIRR 函數修正和更新]。

範例與結果:

範例與結果

範例	結果
`RangeXIRR(-2500,'2008-01-01',2750,'2008-09-01')`	傳回 0.1532

另請參見:

- *RangeIRR (page 1266)*
- *RangeXNPV (page 1291)*
- *XIRR (page 357)*
- XIRR 函數修正和更新

RangeXNPV

RangeXNPV() 會針對運算式的配對數字 (以 **pmt** 和 **date** 表示) 所代表的現金流量排程 (不一定是定期) 傳回淨現值。所有支出均按照 1 年 365 天攤算。

語法:

```
RangeXNPV(discount_rate, value, date{, value, date})
```

傳回的資料類型: 數值

引數

引數	描述
discount_rate	**discount_rate** 是付款應據此折扣的年折扣率。
value	對應於多個日期支出排程的一筆現金流量或一系列現金流量。每個值可能是第三個選用參數的記錄間函數所傳回之單一值或一個範圍的值。一組值中必須包含至少一個正值和一個負值。
date	對應於現金流量付款的支付日期或支付日期排程。

使用此函數時, 適用下列限制:

- 文字值、NULL 值和遺漏值將予以忽略。
- 所有支出均按照 1 年 365 天攤算。

範例 - 指令碼

載入指令碼和結果

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 稱為 RangeTab3 的表格中包含的財務資料。
- 使用 **RangeXNPV()** 函數運算淨現值。

載入指令碼

```
RangeTab3:
LOAD *,
recno() as RangeID,
RangeXNPV(DiscountRate,Value1,Date1,Value2,Date2) as RangeXNPV;
LOAD * INLINE [
DiscountRate|Value1|Date1|Value2|Date2
0.1|-100|2021-01-01|100|2022-01-01|
0.1|-100|2021-01-01|110|2022-01-01|
0.1|-100|2021-01-01|125|2022-01-01|
] (delimiter is '|');
```

結果

載入資料並開啟工作表。建立新的表格並將這些欄位新增為維度:

- RangeID
- RangeXNPV

結果表格

RangeID	RangeXNPV
1	-$9.09
2	-$0.00
3	$13.64

範例 - 圖表運算式

載入指令碼和圖表運算式

概覽

開啟資料載入編輯器並將下面的載入指令碼新增至新的索引標籤。

載入指令碼包含:

- 稱為 RangeTab3 的表格中包含的財務資料。
- 使用 **RangeXNPV()** 函數運算淨現值。

載入指令碼

```
RangeTab3:
LOAD *,
recno() as RangeID,
RangeXNPV(DiscountRate,Value1,Date1,Value2,Date2) as RangeXNPV;
LOAD * INLINE [
DiscountRate|Value1|Date1|Value2|Date2
0.1|-100|2021-01-01|100|2022-01-01|
0.1|-100|2021-01-01|110|2022-01-01|
0.1|-100|2021-01-01|125|2022-01-01|
] (delimiter is '|');
```

結果

請執行下列動作:

載入資料並開啟工作表。建立新的表格並新增下列計算作為量值。

```
=RangeXNPV(0.1, -2500,'2008-01-01',2750,'2008-09-01')
```

結果表格

=XIRR(Payments, Date)
$80.25

另請參見：

XNPV (page 363)

8.22 關係函數

這是計算圖表中個別維度值屬性的函數群組，使用已彙總的數字。

這些函數在 Sense 中有關聯，該函數不僅根據資料點本身的值輸出，也根據值與其他資料點的關係來輸出。例如，若沒有與其他維度值比較，無法計算排名。

這些函數僅可用於圖表運算式。這無法用於載入指令碼。

圖表中需要維度，因為這定義比較所需的其他資料點。因此，關係函數在沒有維度的圖表 (例如 KPI 物件) 中不具意義。

排名函數

當使用這些函數時，會自動停用 [隱藏零值]。NULL 值將予以忽略。

Rank

Rank() 評估運算式中圖表的列，並且對每一列，顯示在運算式中評估之維度的值的相對位置。評估運算式時，該函數會比較該結果與包含目前資料行區段的其他列所產生的結果，並傳回區段內目前列的排名。

Rank - 圖表函數(**[TOTAL** [<fld {, fld}>]] expr[, mode[, fmt]])

HRank

HRank() 評估運算式，並將結果與包含樞紐分析表之目前列區段的其他資料行結果進行比較。然後，該函數傳回區段內目前資料行的排名。

HRank- 圖表函數(**[TOTAL**] expr[, mode[, fmt]])

叢集函數

KMeans2D

屬性群組**網站授權**包含與 Qlik Sense 系統授權相關的屬性。所有欄位皆為必填且不可空白。

網站授權屬性

屬性名稱	**描述**
所有者姓名	Qlik Sense 產品擁有者的使用者名稱。
所有者組織	Qlik Sense 產品擁有者所屬組織的名稱。
序號	指派到 Qlik Sense 軟體的序號。
控制編號	指派到 Qlik Sense 軟體的控制編號。
LEF 存取	指派到 Qlik Sense 軟體的 授權啟用器檔案 (LEF)。

KMeans2D() 會套用 K 平均演算法叢集以評估圖表的列，而對於每個圖表列，會針對此資料點指派到的叢集顯示叢集 ID。叢集演算法使用的欄由參數 coordinate_1 和 coordinate_2 分別判定。這些都是彙總。建立的叢集數量由 num_clusters 參數判定。可以選擇透過規範參數來正規化資料。

KMeans2D - 圖表函數(num_clusters, coordinate_1, coordinate_2 [, norm]**)**

KMeansND

KMeansND() 會套用 K 平均演算法叢集以評估圖表的列，而對於每個圖表列，會針對此資料點指派到的叢集顯示叢集 ID。叢集演算法使用的欄由參數 coordinate_1 和 coordinate_2, 等來判定，最多 n 欄。這些都是彙總。建立的叢集數量由 num_clusters 參數判定。

KMeansND - 圖表函數(num_clusters, num_iter, coordinate_1, coordinate_2 [,coordinate_3 [, ...]]**)**

KMeansCentroid2D

KMeansCentroid2D() 會套用 K 平均演算法叢集以評估圖表的列，而對於每個圖表列，會針對此資料點指派到的叢集顯示該叢集的所需座標。叢集演算法使用的欄由參數 coordinate_1 和 coordinate_2 分別判定。這些都是彙總。建立的叢集數量由 num_clusters 參數判定。可以選擇透過規範參數來正規化資料。

KMeansCentroid2D - 圖表函數(num_clusters, coordinate_no, coordinate_1, coordinate_2 [, norm]**)**

KMeansCentroidND

KMeansCentroidND() 會套用 k-means 叢集以評估圖表的列，而對於每個圖表列，會針對此資料點指派到的叢集顯示該叢集的所需座標。叢集演算法使用的欄由參數 coordinate_1、coordinate_2 等來判定，最多 n 欄。這些都是彙總。建立的叢集數量由 num_clusters 參數判定。

KMeansCentroidND - 圖表函數(num_clusters, num_iter, coordinate_no, coordinate_1, coordinate_2 [,coordinate_3 [, ...]]**)**

時間序列分解函數

STL_Trend

STL_Trend 是時間序列分解函數。連同 **STL_Seasonal** 和 **STL_Residual**，此函數用來將時間序列分解為季節、趨勢和剩餘元件。在 STL 演算法的脈絡下，時間序列分解用來識別輸入指標和其他參數下的週期性季節模式和一般趨勢。**STL_Trend** 函數將會從時間序列資料中識別一般趨勢，獨立於季節模式和週期。

STL_Trend - 圖表函數(target_measure, period_int [,seasonal_smoother [,trend_smoother]])

STL_Seasonal

STL_Seasonal 是時間序列分解函數。連同 **STL_Trend** 和 **STL_Residual**，此函數用來將時間序列分解為季節、趨勢和剩餘元件。在 STL 演算法的脈絡下，時間序列分解用來識別輸入指標和其他參數下的週期性季節模式和一般趨勢。**STL_Seasonal** 函數可以識別時間序列內的季節模式，從依資料顯示的一般趨勢中將此分出。

STL_Seasonal - 圖表函數(target_measure, period_int [,seasonal_smoother [,trend_smoother]]**)**

STL_Residual

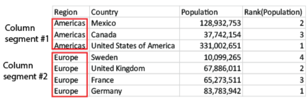
STL_Residual 是時間序列分解函數。連同 **STL_Seasonal** 和 **STL_Trend**，此函數用來將時間序列分解為季節、趨勢和剩餘元件。在 STL 演算法的脈絡下，時間序列分解用來識別輸入指標和其他參數下的週期性季節模式和一般趨勢。執行此操作時，輸入指標中的部分變化不適合用於季節和趨勢元件中，將會定義為剩餘元件。**STL_Residual** 圖表函數擷取此計算部分。

STL_Residual - 圖表函數(`target_measure, period_int [,seasonal_smoother [,trend_smoother]]`**)**

Rank - 圖表函數

Rank() 評估運算式中圖表的列，並且對每一列，顯示在運算式中評估之維度的值的相對位置。評估運算式時，該函數會比較該結果與包含目前資料行區段的其他列所產生的結果，並傳回區段內目前列的排名。

資料行區段



	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

針對圖表而非表格，目前的資料行區段會在其出現於圖表的連續表同等表格時加以定義。

語法：

Rank([**TOTAL**] `expr[, mode[, fmt]]`**)**

傳回的資料類型： 雙值

引數：

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
mode	指定函數結果的數字表示法。
fmt	指定函數結果的文字表示法。
TOTAL	如果圖表為單一維度或運算式前面加上 **TOTAL** 限定詞，則沿著整個資料行計算該函數。如果表格或同等表格有多個垂直維度，則目前資料行區段僅會包含在所有維度資料行中與目前列有相同值的列 (顯示欄位間排序順序之最後一個維度的資料行除外)。

傳回的排名都是雙值，在這種情況下，各個列都會有唯一的排名，這是介於 1 與目前資料行區段中的列數兩者之間的整數。

多個列有相同的排名時，可以使用 **mode** 和 **fmt** 參數控制文字和數字表示法。

mode

第二個引數 **mode** 可控制下列值：

mode 範例

值	描述
0 (預設)	如果共用群組中所有的排名都落在整個排名中間值下半部，所有列將得到共用群組的最低排名。 如果共用群組中所有的排名都落在整個排名中間值上半部，所有列將得到共用群組的最高排名。 如果共用群組中的排名落在整個排名的中間值，所有列的值將相當於整個資料行區段中最高與最低排名兩者的平均。
1	所有列的最低排名。
2	所有列的平均排名。
3	所有列的最高排名。
4	第一列的最低排名，然後每列遞增 1。

fmt

第三個引數 **fmt** 可使用下列值：

fmt 範例

值	描述
0 (預設)	低值 **-** 所有列的高值 (例如 3 - 4)。
1	所有列的低值。
2	第一列的低值，在後續列上則空白。

mode 4 與 **fmt** 2 的列順序取決於圖表維度的排序順序。

範例與結果：

從維度 Product 和 Sales 建立兩個視覺化，再從 Product 和 UnitSales 建立另一個視覺化。如下表所示新增量值。

排名範例

範例	結果
範例 1。建立一個具有維度 `Customer` 和 `Sales` 及量值 `Rank(Sales)` 的表格	結果取決於維度的排序順序。如果表格針對 Customer 排序，該表格會列出 Astrida 的所有 Sales 值，然後列出 Betacab，以此類推。Rank(Sales) 的結果會顯示 Sales 值 12 為 10、Sales 值 13 為 9，以此類推，而針對 Sales 值 78 傳回的排名值為 1。下一個資料行區段是從 Betacab 開始，而區段中的第一個 Sales 值是 12。其 Rank(Sales) 排名值指定為 11。 如果表格針對 Sales 排序，資料行區段會包含 Sales 值和對應的 Customer。因為有兩個 Sales 值都是 12 (Astrida 和 Betacab)，針對該資料行區段，Customer 每個值的 Rank (Sales) 值是 1-2。這是因為 Sales 值 12 有兩個 Customer 值。若有 4 個值，所有列的結果就會是 1-4。這顯示了引數 fmt 預設值 (0) 結果的外觀。
範例 2。以維度 [Customer] 取代 [Product]，並新增量值 `Rank (Sales,1,2)`	這樣系統會針對每個資料行區段的第一列傳回 1，並讓所有其他列保留空白，因為引數 **mode** 和 **fmt** 分別設為 1 和 2。

範例 1 的結果，表格針對 Customer 排序：

結果表格

Customer	Sales	Rank(Sales)
Astrida	12	10
Astrida	13	9
Astrida	20	8
Astrida	22	7
Astrida	45	6
Astrida	46	5
Astrida	60	4
Astrida	65	3
Astrida	70	2
Astrida	78	1
Betcab	12	11

範例 1 的結果，表格針對 Sales 排序：

結果表格

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

範例中使用的資料:

```
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 '|');
```

另請參見:

Sum - 圖表函數 (page 328)

HRank- 圖表函數

HRank() 評估運算式, 並將結果與包含樞紐分析表之目前列區段的其他資料行結果進行比較。然後, 該函數傳回區段內目前資料行的排名。

語法:

```
HRank([ TOTAL ] expr [ , mode [, fmt ] ])
```

傳回的資料類型: 雙值

此函數僅適用於樞紐分析表。在所有其他圖表類型中, 則會傳回 NULL。

引數:

引數

引數	描述
expr	包含待測量之資料的運算式或欄位。
mode	指定函數結果的數字表示法。
fmt	指定函數結果的文字表示法。
TOTAL	如果圖表為單一維度或運算式前面加上 **TOTAL** 限定詞, 則沿著整個資料行計算該函數。如果表格或同等表格有多個垂直維度, 則目前資料行區段僅會包含在所有維度資料行中與目前列有相同值的列 (顯示欄位間排序順序之最後一個維度的資料行除外)。

如果樞紐分析表單一維度或運算式前面加上 **total** 限定詞, 則目前列區段一律等於整個列。如果樞紐分析表有多個水平維度, 則目前列區段僅會包括在所有維度列中與目前資料行有相同值的資料行 (顯示欄位間排序順序之最後一個水平維度的列除外)。

傳回的排名都是雙值, 在這種情況下, 各個資料行都會有唯一的排名, 這是介於 1 與目前列區段中的資料行數兩者之間的整數。

多個資料行有相同的排名時, 可以使用 **mode** 和 **format** 引數控制文字和數字表示法。

第二個引數 **mode** 會指定函數結果的數字表示法:

mode 範例

值	描述
0 (預設)	如果共用群組中所有的排名都落在整個排名中間值下半部，所有資料行將得到共用群組的最低排名。 如果共用群組中所有的排名都落在整個排名中間值上半部，所有資料行將得到共用群組的最高排名。 如果共用群組中的排名落在整個排名的中間值，所有列的值將相當於整個資料行區段中最高與最低排名兩者的平均。
1	群組中所有資料行的最低排名。
2	群組中所有資料行的平均排名。
3	群組中所有資料行的最高排名。
4	第一個資料行的最低排名，然後群組中每一資料行遞增 1。

第三個引數 **format** 會指定函數結果的文字表示法：

format 範例

值	描述
0 (預設)	群組中所有資料行的低值 **&' - '&** 高值 (例如 3 - 4)。
1	群組中所有資料行的低值。
2	第一個資料行的低值，在群組的後續資料行上則空白。

mode 4 與 **format** 2 的資料行順序取決於圖表維度的排序順序。

範例：

```
HRank( sum( Sales ))
```

```
HRank( sum( Sales ), 2 )
```

```
HRank( sum( Sales ), 0, 1 )
```

透過 k-means 最佳化：實際範例

下列範例說明實際案例，其中 KMeans 叢集和 Centroid 函數套用至資料集。KMeans 函數將資料點區分為共用相似性的叢集。透過可設定的反覆項目數量套用 KMeans 演算法時，叢集變得更加精簡且不同。

Kmeans 在各種使用案例中用於許多欄位；一些叢集使用案例的範例包括客戶分段、詐騙偵測、預測帳戶消耗、設定用戶端獎勵目標、網路犯罪識別以及傳遞路徑最佳化。KMeans 叢集演算法的使用逐漸增加，企業嘗試藉此推斷模式和最佳化服務提供。

Qlik Sense KMeans 和 Centroid 函數

Qlik Sense 提供兩個根據相似性將資料點組成叢集的 KMeans 函數。請參閱 *KMeans2D - 圖表函數 (page 1310)* 和 *KMeansND - 圖表函數 (page 1325)*。**KMeans2D** 函數接受兩個維度，透過**散佈圖**圖表視覺化結果時，也運作良好。**KMeansND** 函數接受超過兩個維度。由於在標準圖表概念化 2D 結果很容易，下列示範使用兩個維度在**散佈圖**圖表套用 KMeans。可以透過運算式來著色，以視覺化 KMeans 叢集；或者按照此範例的說明透過維度來進行。

Qlik Sense centroid 函數決定叢集中所有資料點的算數平均值位置，並識別該叢集的中央點或 centroid。對於每個圖表列 (或記錄)，centroid 函數會顯示此資料點已指派到的叢集座標。請參閱 *KMeansCentroid2D - 圖表函數 (page 1339)* 和 *KMeansCentroidND - 圖表函數 (page 1340)*。

使用案例和範例概述

下列範例透過模擬實際情境來分段說明。美國紐約州的一間紡織公司必須盡量降低運費以減少支出。一個方式是重新設置更靠近經銷商的倉庫。該公司在紐約州各地採用 118 個經銷商。下列示範模擬營運經理可以如何使用 KMeans 函數，將經銷商分為五個叢集的地理區域，然後使用 centroid 函數，識別位於這些叢集中央的五個最佳倉庫位置。目標是探索可用來識別五個中央倉庫位置的對應座標。

資料集

資料集根據隨機產生的名稱和紐約州地址，並具有真實的緯度和經度座標。資料集包含以下十欄：id、first_name、last_name、telephone、address、city、state、zip、latitude、longitude。資料集可在下面作為檔案使用，您可以在本機下載，然後上傳至 Qlik Sense 或在資料載入編輯器進行內嵌。建立的應用程式命名為 *Distributors KMeans and Centroid* (經銷商 KMeans 和 Centroid)，而應用程式中的第一個工作表命名為 *stribution cluster analysis* (經銷叢集分析)。

選取下列連結以下載樣本資料檔案：DistributorData.csv

Distributor 資料集：Qlik Sense 中資料載入編輯器的內嵌載入 (page 1308)

標題：DistributorData

記錄總數：118

套用 KMeans2D 函數

在此範例中，使用 *DistributorData* 資料集示範**散佈圖**圖表的設定、套用 **KMeans2D** 函數，並且依照維度為圖表著色。

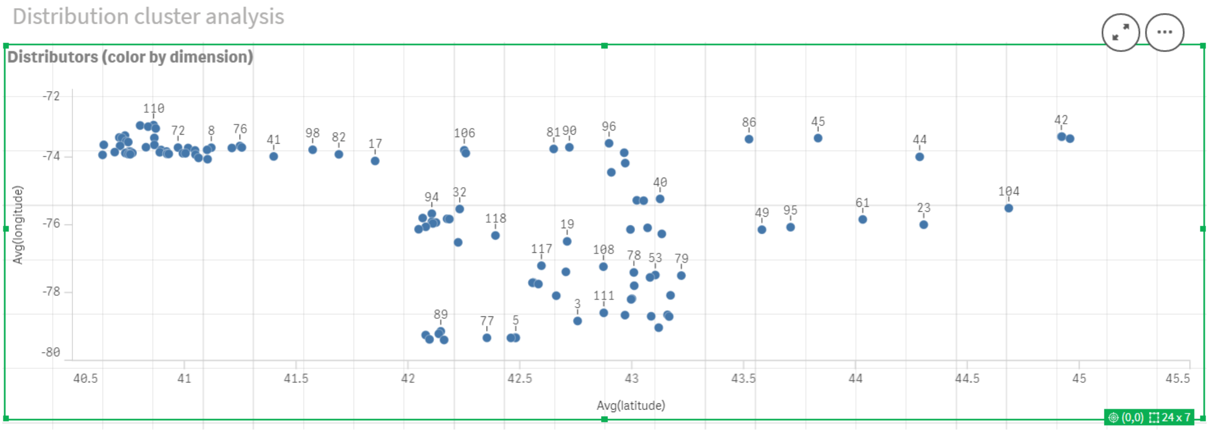
請注意，Qlik Sense KMeans 函數使用稱為深度差 (DeD) 的方法支援自動叢集。若使用者對叢集數量設定 0，會判定該資料集的最佳叢集數量。不過，對於此範例，會為 **num_clusters** 引數建立一個變數 (參閱 *KMeans2D - 圖表函數 (page 1310)* 瞭解語法)。因此，會依照變數指定所需叢集數量 (k=5)。

1. 將**散佈圖**圖表拖放到工作表並命名為 *Distributors (by dimension)* (經銷商 (依維度))。
2. 建立**變數**以指定叢集的數量。該**變數**命名為 *vDistClusters*。對於變數**定義**，輸入 *5*。
3. 圖表的**資料**設定：
   a. 在**維度**之下，在**泡泡球**選取 *id* 欄位。在**標籤**輸入*Cluster id*。
   b. 在**量值**之下，*Avg([latitude])* 是 **X 軸**的運算式。

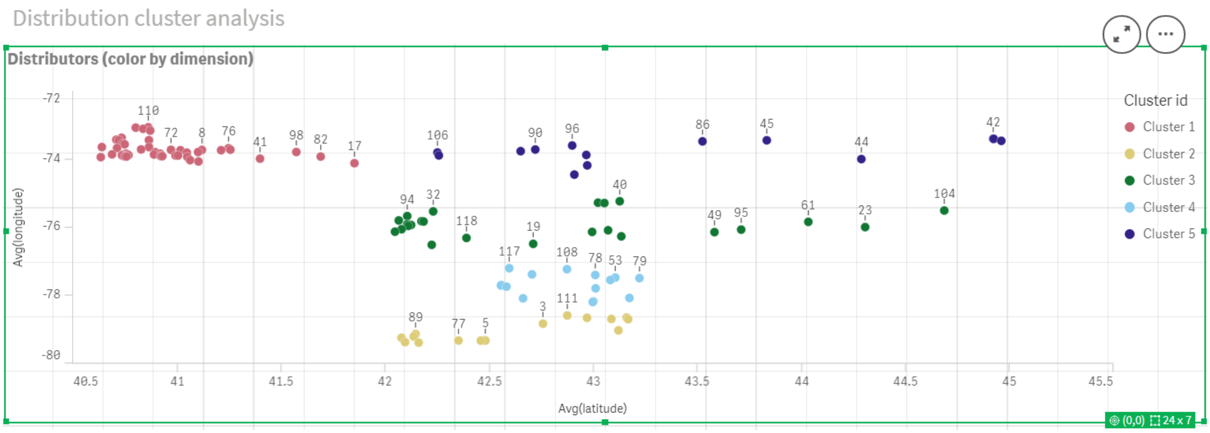
c. 在**量值**之下，*Avg([longitude])* 是 **Y 軸**的運算式。

4. **外觀**設定：
   a. 在**色彩和圖例**之下，為**色彩**選擇**自訂**。
   b. 選取**依維度**以著色圖表。
   c. 輸入下列運算式：*=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
   d. 選取**持續性色彩**的核取方塊。

依維度著色的 KMeans 之前的散佈圖已套用



依維度著色的 KMeans 之後的散佈圖已套用



新增**表格**：*經銷商*

手邊有個表格以便快速存取相關資料，會很有幫助。**散佈圖**圖表透過表格顯示 *ids*，並新增對應經銷商名稱以供參考。

1. 將命令為*經銷商*的**表格**拖放至工作表，其中新增了以下的**欄** (維度)：*id*、*first_name* 和 *last_name*。

表格:經銷商名稱

Distributors

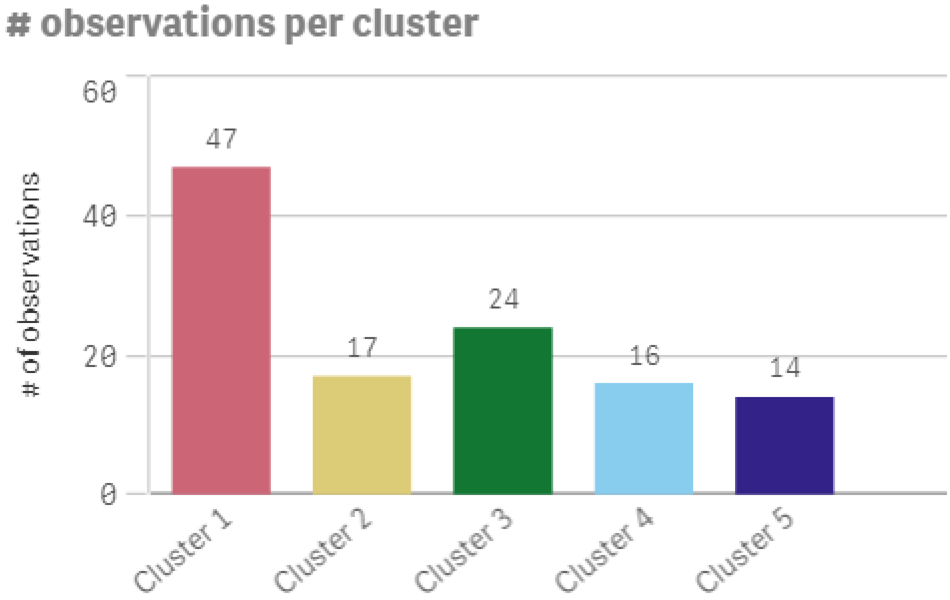
id	first_name	last_name
1	Kaiya	Snow
2	Dean	Roy
3	Eden	Paul
4	Bryanna	Higgins
5	Elisabeth	Lee
6	Skylar	Robinson
7	Cody	Bailey
8	Dario	Sims
9	Deacon	Hood

新增**長條圖**:*# observations per cluster*

對於倉庫分佈情境,瞭解每個倉庫將會服務多少經銷商很有幫助。因此,會建立**長條圖**,衡量向每個叢集指派多少經銷商。

1. 將**長條圖**拖放到工作表。圖表命名為:*# observations per cluster*。
2. **長條圖**的**資料**設定:
   a. 新增標籤為 *Clusters* 的**維度** (可在套用運算式之後新增標籤)。輸入下列運算式:*=pick (aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
   b. 新增標籤為 *# of observations* 的**量值**。輸入下列運算式:*=count(aggr(KMeans2D (vDistClusters,only(latitude),only(longitude)),id))*
3. **外觀**設定:
   a. 在**色彩和圖例**之下,為**色彩**選擇**自訂**。
   b. 選取**依維度**以著色圖表。
   c. 輸入下列運算式: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
   d. 選取**持續性色彩**的核取方塊。
   e. **顯示圖例**已關閉。
   f. 在**呈現方式**之下,**值標籤**切換為**自動**。
   g. 在 **X 軸**之下:選取**叢集**、**僅標籤**。

長條圖 : # observations per cluster



套用 **Centroid2D** 函數

已為 **Centroid2D** 函數新增第二個表格，這將識別潛在倉庫位置的座標。此表格顯示五個已識別經銷商群組的的中心位置 (centroid 值)。

1. 將**表格**拖放到工作表並命名為 *Cluster centroids* (叢集 centroids)，並新增以下的欄：
   a. 新增標籤為*叢集*的**維度**。輸入下列運算式：*=pick(aggr(KMeans2D(vDistClusters,only (latitude),only(longitude)),id)+1,'Warehouse 1','Warehouse 2','Warehouse 3','Warehouse 4','Warehouse 5')*
   b. 新增標籤為 *latitude (D1)* 的**量值**。輸入下列運算式：*=only(aggr(KMeansCentroid2D (vDistClusters,0,only(latitude),only(longitude)),id))*
   請注意，參數 **coordinate_no** 對應至第一個維度(0)。在此情況下，會根據 X 軸繪製維度 *latitude* (緯度)。若我們處理 **CentroidND** 函數且最多有六個維度，這些參數項目可能是六個值中的任一項：0、1、2、3、4 或 5。
   c. 新增標籤為 *longitude (D2)* 的**量值**。輸入下列運算式：*=only(aggr(KMeansCentroid2D (vDistClusters,1,only(latitude),only(longitude)),id))*
   運算式中的參數 **coordinate_no** 對應至第二個維度(1)。在此情況下，會根據 Y 軸繪製維度 *longitude* (經度)。

表格：叢集 centroid 計算

Cluster centroids

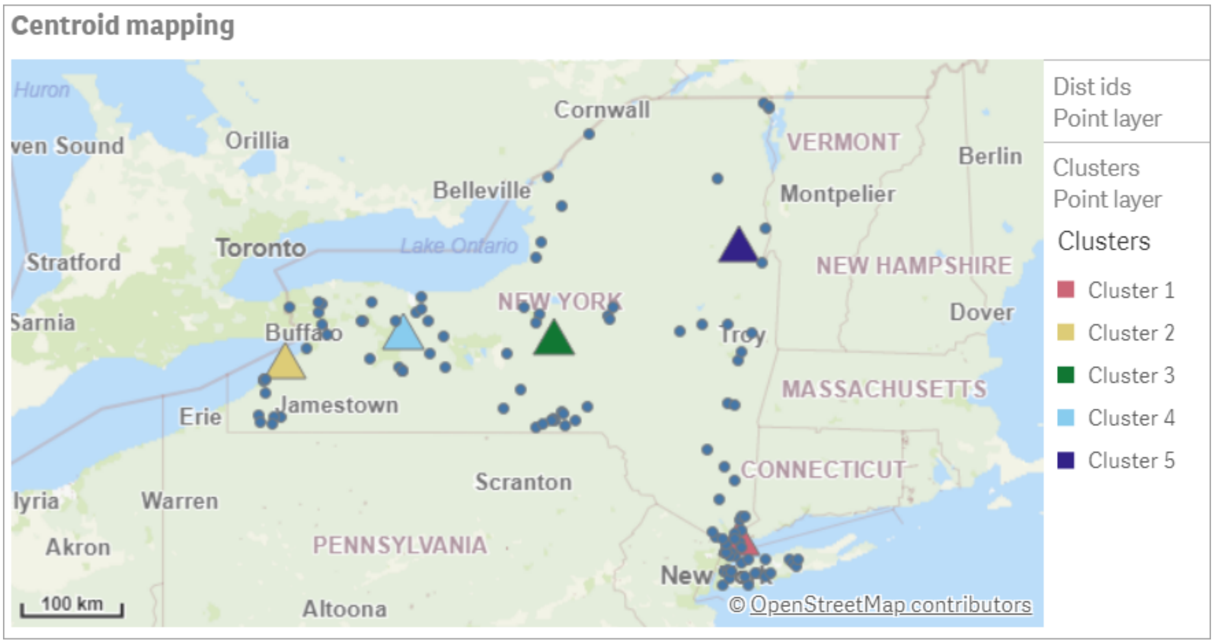
Clusters	latitude (D1)	longitude (D2)
Totals	-	-
Warehouse 1	40.945422240426	-73.719966482979
Warehouse 2	42.590538729412	-79.067889217647
Warehouse 3	42.805089516667	-75.901621883333
Warehouse 4	42.8581692625	-77.6800485875
Warehouse 5	43.436770771429	-73.734622635714

Centroid 對應

下一步是對應 centroids。由應用程式開發人員決定是否偏好在獨立的工作表置放視覺化。

1. 將命名為 *Centroid mapping* 的**對應**拖放到工作表。
2. 在**圖層**區段中。選取**新增圖層**，然後選取**點圖層**。
   a. 選取**欄位** *id* 並新增 *Dist ids* **標籤**。
   b. 在**位置** 區段中，選取**緯度和經度欄位**的核取方塊。
   c. 對於**緯度**，選取 *latitude* 欄位。
   d. 對於**經度**，選取 *longitude* 欄位。
   e. 在**大小 & 形狀** 區段中，為**形狀**選取**泡泡**，並在滑桿依照偏好減少**大小**。
   f. 在**色彩**區段，**單一色彩**並為**色彩**選取藍色，為**輪廓**色彩選取灰色 (這些選項也依照偏好選取)。
3. 在**圖層**區段，選取**新增圖層**，然後選取**點圖層**，以新增第二個**點圖層**。
   a. 輸入下列運算式：*=aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)*
   b. 新增 **標籤** *Clusters*。
   c. 在**位置** 區段中，選取**緯度和經度欄位**的核取方塊。
   d. 對於在此情況下沿著 X 軸繪製的**緯度**，新增下列運算式：*=aggr(KMeansCentroid2D (vDistClusters,0,only(latitude),only(longitude)),id)*
   e. 對於在此情況下沿著 Y 軸繪製的**經度**，新增下列運算式：*=aggr(KMeansCentroid2D (vDistClusters,1,only(latitude),only(longitude)),id)*
   f. 在**大小 & 形狀** 區段中，為**形狀**選取**三角形**，並在滑桿依照偏好減少**大小**。
   g. 在**色彩和圖例**之下，為**色彩**選取**自訂**。
   h. 選取**依維度**以著色圖表。輸入下列運算式：*=pick(aggr(KMeans2D(vDistClusters,only (latitude),only(longitude)),id)+1,'Cluster 1','Cluster 2','Cluster 3','Cluster 4','Cluster 5')*
   i. 維度標籤為 *Clusters*。
4. 在**地圖設定**中，為**投影**選取**調適型**。為**測量單位**選取**公制**。

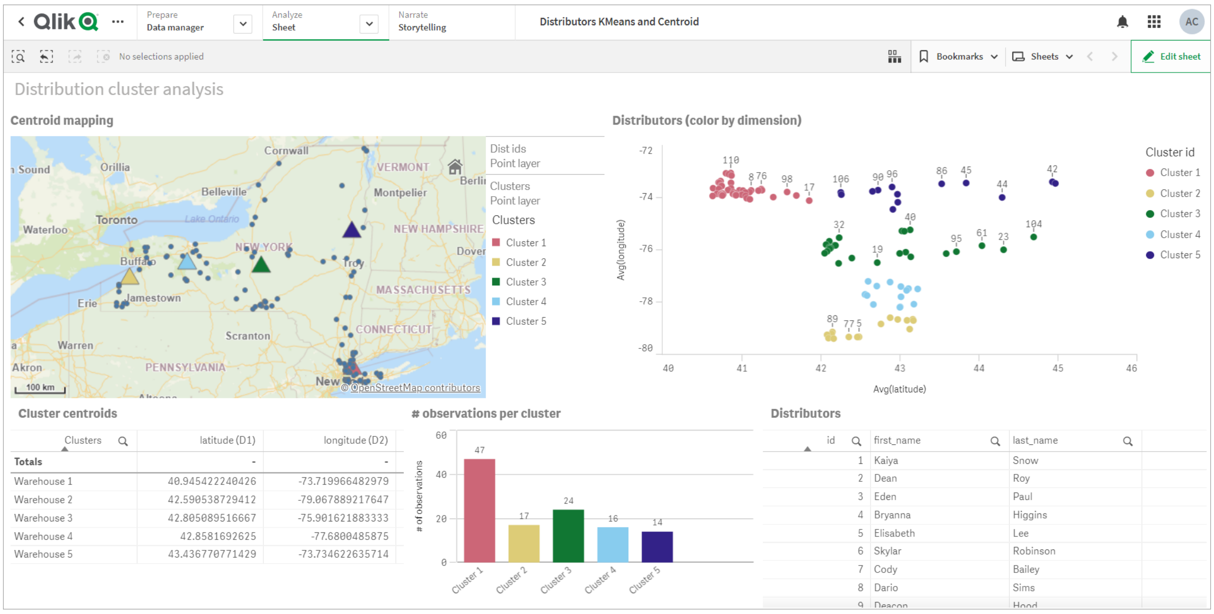
地圖:依叢集對應的 Centroids



結論

對此實際情境使用 KMeans 函數時,經銷商已根據相似性分為類似群組或叢集,在此情況下,亦即彼此的鄰近性。Centroid 函數已套用至這些叢集,以識別五個對應座標。這些座標提供建立或定位倉庫的初始中心位置。centroid 函數已套用至**地圖**圖表,因此應用程式使用者可以看見 centroids 相對於周遭叢集資料點的位置。產生的座標代表可讓紐約州的經銷商盡量降低運費成本的潛在倉庫位置。

應用程式:KMeans 和 centroid 分析範例



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

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

Distributor 資料集：Qlik Sense 中資料載入編輯器的內嵌載入

```
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 - 圖表函數

KMeans2D() 會套用 K 平均演算法叢集以評估圖表的列，而對於每個圖表列，會針對此資料點指派到的叢集顯示叢集 ID。叢集演算法使用的欄由參數 coordinate_1 和 coordinate_2 分別判定。這些都是彙總。建立的叢集數量由 num_clusters 參數判定。可以選擇透過規範參數來正規化資料。

KMeans2D 每個資料點傳回一個值。傳回的值是雙值，也是對應於每個資料點指派到的叢集的整數值。

語法:

```
KMeans2D(num_clusters, coordinate_1, coordinate_2 [, norm])
```

傳回的資料類型: 雙值

引數:

引數

引數	描述
num_clusters	指定叢集數量的整數。
coordinate_1	計算第一個座標的彙總，通常是 x 軸 (屬於可從圖表製作的散佈圖)。其他參數 coordinate_2 會計算第二個座標。

引數	描述
norm	選用的正規化方法已在 KMeans 叢集之前套用至資料集。 可能值: 0 或「無」代表沒有正規化 1 或「zscore」代表 Z 分數正規化 2 或「minmax」代表最小-最大正規化 若沒有套用參數或者若套用的參數不正確,則不會套用任何正規化。 Z 分數根據功能平均值和標準差正規化資料。Z 分數無法確保每個功能都有相同的比例,但這在處理異常值時是優於最小-最大的方法。 最小-最大正規化可取用每一項的最小和最大值並重新計算每個資料點,以確保功能具有相同比例。

範例: 圖表運算式
在此範例中,我們使用 *Iris* 資料集建立散佈圖,然後使用 KMeans 依運算式為資料著色。

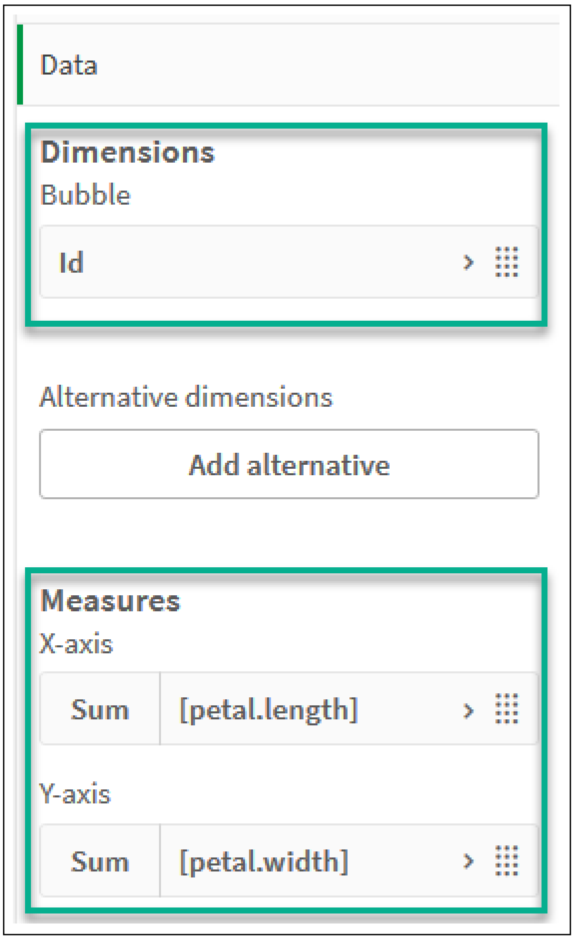
我們也會建立 *num_clusters* 引數的變數,然後使用變數輸入方塊來變更叢集的數量。

Iris 資料集公開用於各種格式。我們已提供資料作為要在 Qlik Sense 中使用資料載入編輯器載入的內嵌表格。請注意,在此範圍中,我們已將 *Id* 欄新增至資料表格。

在 Qlik Sense 中載入資料後,我們進行下列事項:

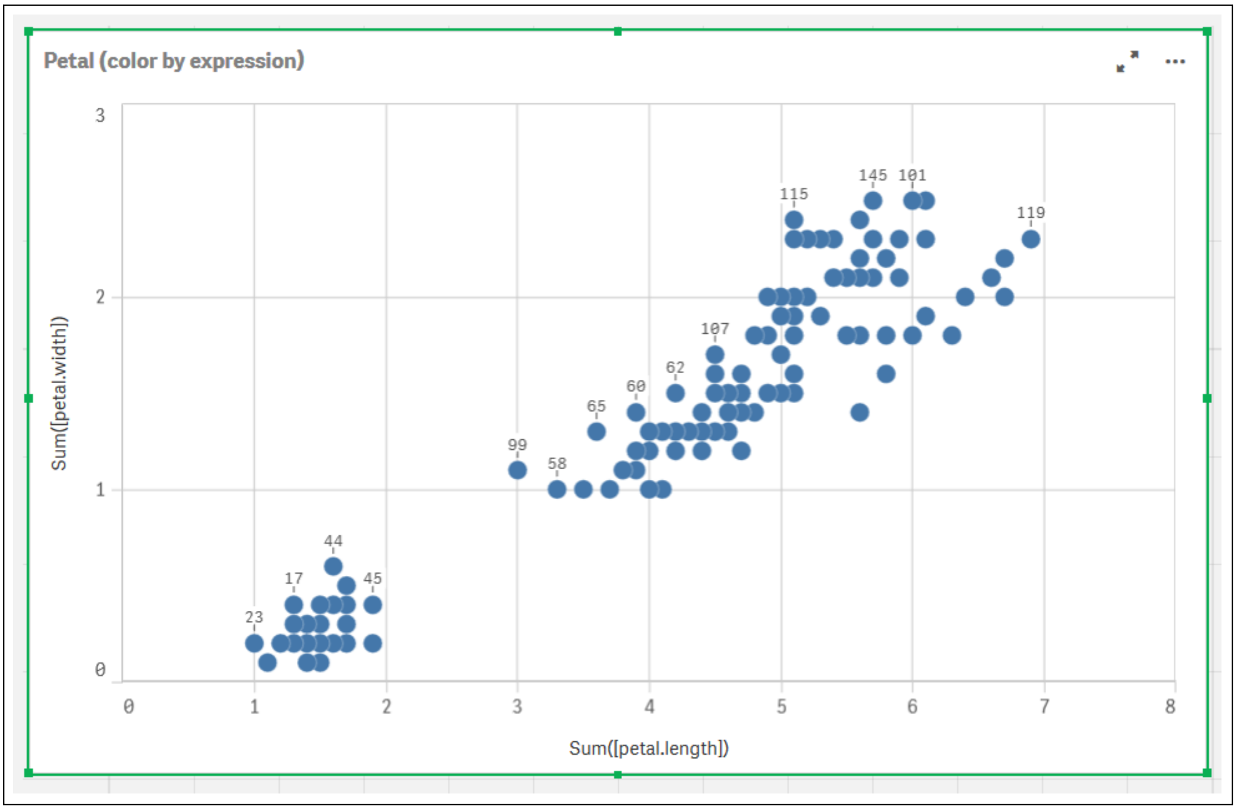
1. 將**散佈圖**圖表拖曳至新的工作表。將圖表命名為*花瓣 (依運算式著色)*。
2. 建立變數以指定叢集的數量。對於變數**名稱**,輸入 *KmeansPetalClusters*。對於變數**定義**,輸入 *=2*。
3. 設定圖表的**資料**:
   i. 在**維度**之下,在**泡泡球**選擇欄位的 *id*。輸入標籤的叢集 ID。
   ii. 在**量值**之下,為 **X 軸**的運算式選擇 *Sum([petal.length])*。
   iii. 在**量值**之下,為 **Y 軸**的運算式選擇 *Sum([petal.width])*。

花瓣 (依運算式著色) 圖表的資料設定



資料點繪製於圖表上。

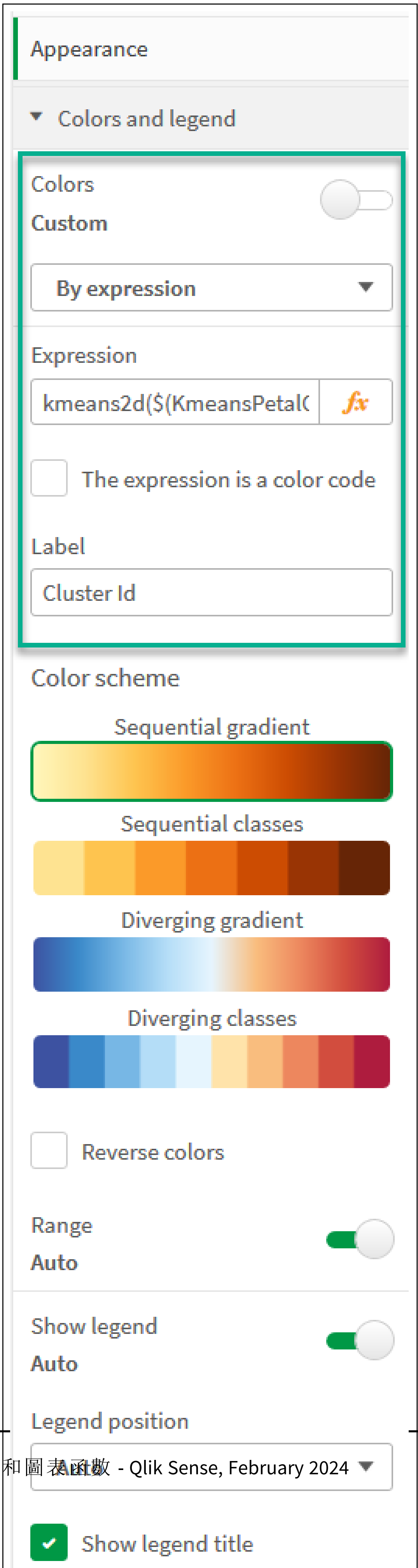
花瓣 (依運算式著色) 圖表的資料點



4. 設定圖表的**外觀**:
   i. 在**色彩和圖例**之下，為**色彩**選擇 **自訂**。
   ii. 選擇**依運算式**為圖表著色。
   iii. 在**運算式**輸入下列內容：*kmeans2d($(KmeansPetalClusters), Sum([petal.length]), Sum ([petal.width]))*
   請注意，*KmeansPetalClusters* 是我們設定為 *2* 的變數。
   或者，請輸入下列內容：*kmeans2d(2, Sum([petal.length]), Sum([petal.width]))*
   iv. 取消選取**運算式為色彩代碼**核取方塊。

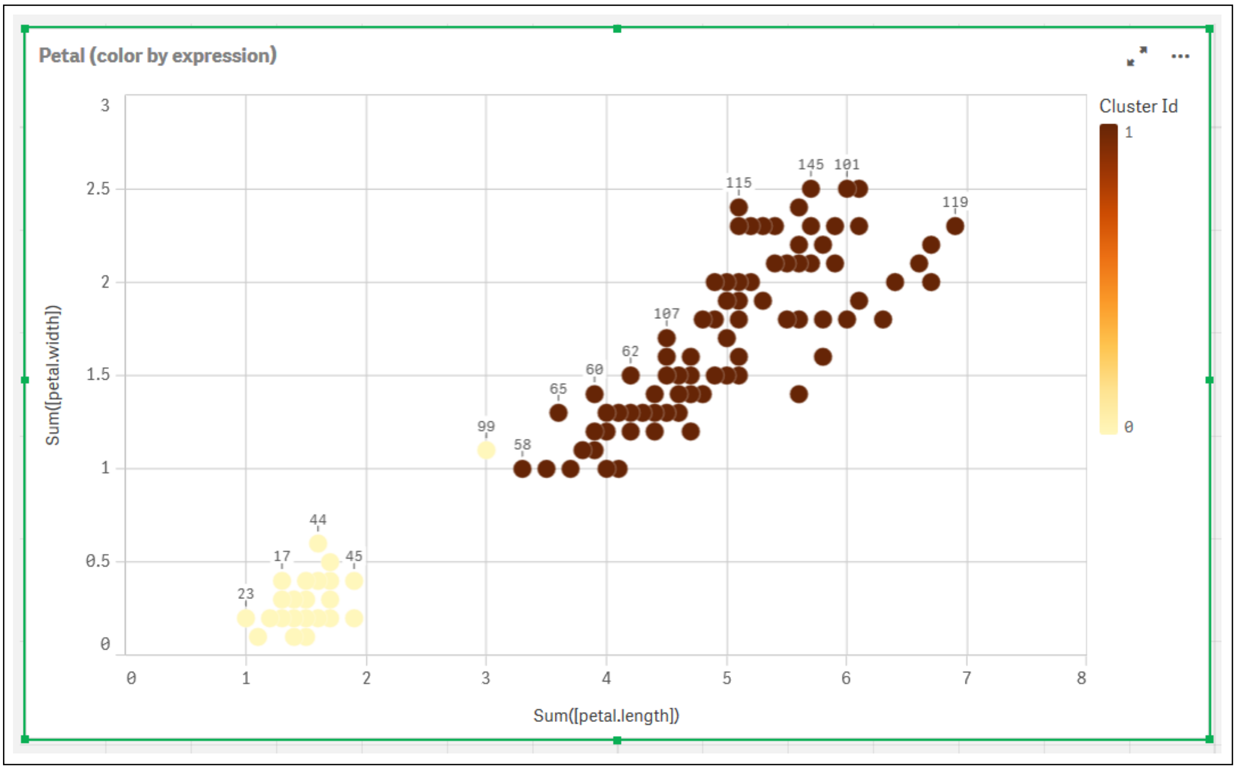
v. 在**標籤**輸入下列內容: *叢集 Id*

花瓣 (依運算式著色) 圖表的外觀設定


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

圖表上的兩個叢集依 KMeans 運算式著色。

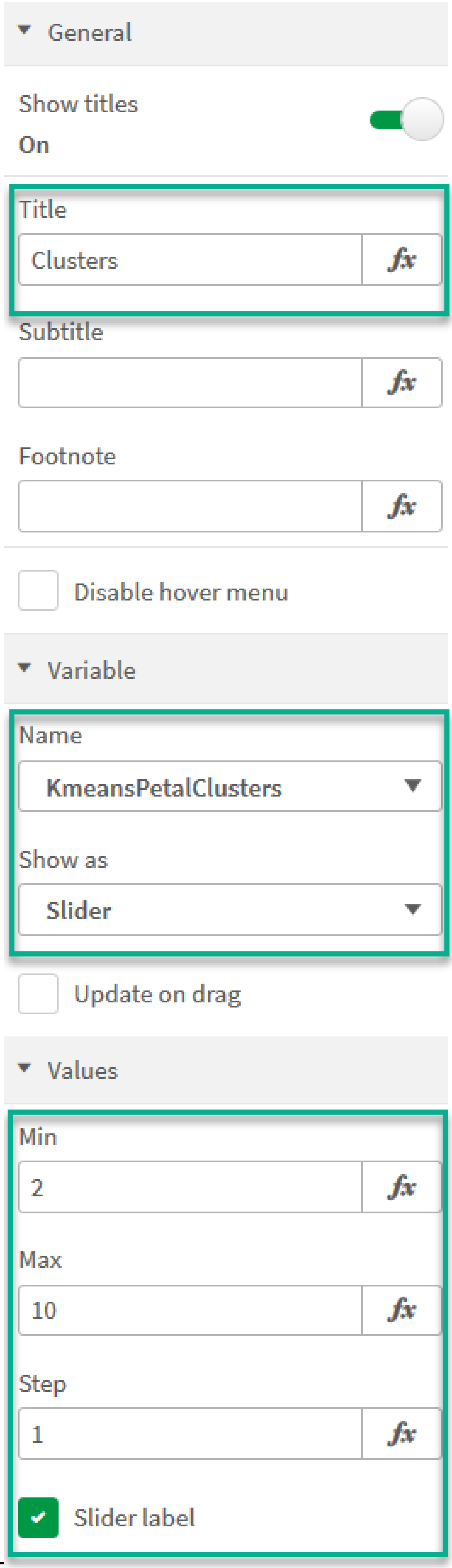
叢集依花瓣 (依運算式著色) 圖表上的運算式著色



5. 為叢集數量新增**變數輸入**方塊。
   i. 在**資產**面板的**自訂物件**之下，選擇 **Qlik 儀表板搭售**。若我們無法存取儀表板搭售，仍可使用我們建立的變數變更叢集數量，或直接作為運算式中的整數。
   ii. 將**變數輸入**方塊拖曳到工作表。
   iii. 在**外觀**之下，按一下**一般**。
   iv. 在**標題**輸入下列內容：*叢集*
   v. 按一下**變數**。
   vi. 在**名稱**選擇下列變數：*KmeansPetalClusters*。
   vii. 在**顯示為**選擇**滑桿**。

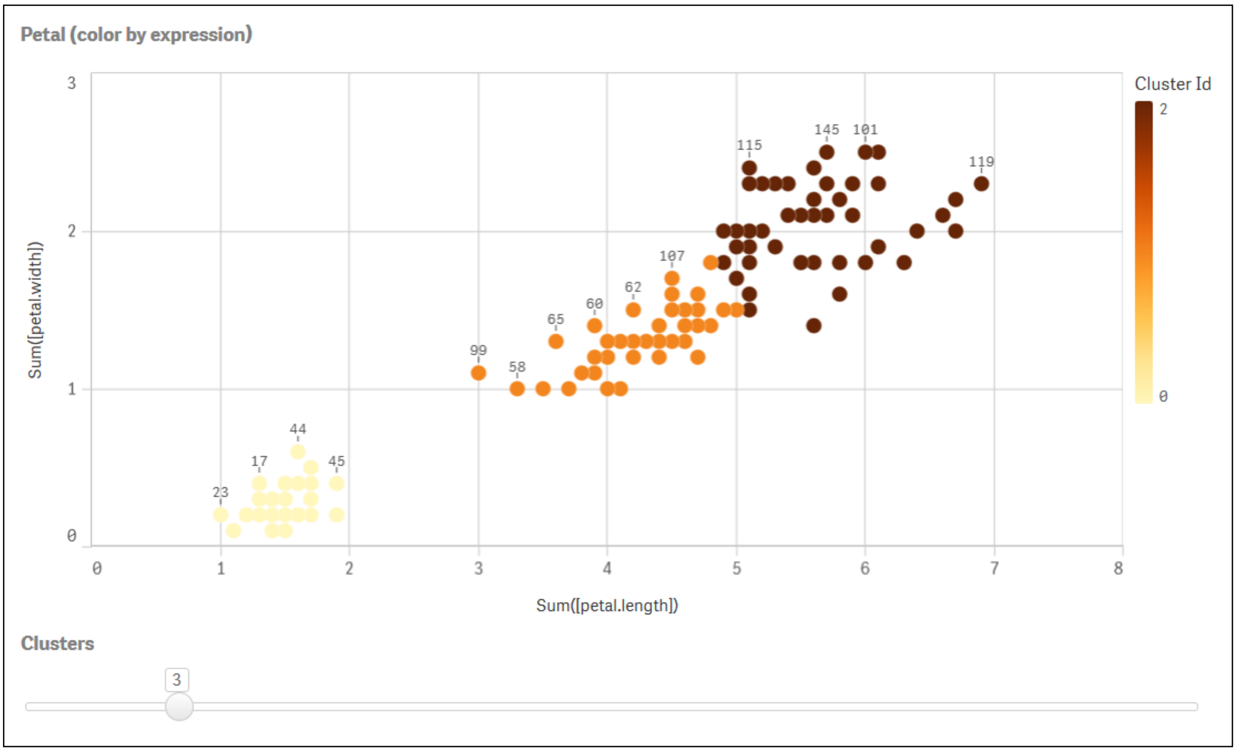
viii. 選擇**值**, 並按需要進行設定。

叢集變數輸入方塊的外觀


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

完成編輯後，可以使用*叢集變數輸入方塊*中的滑桿來變更叢集的數量。

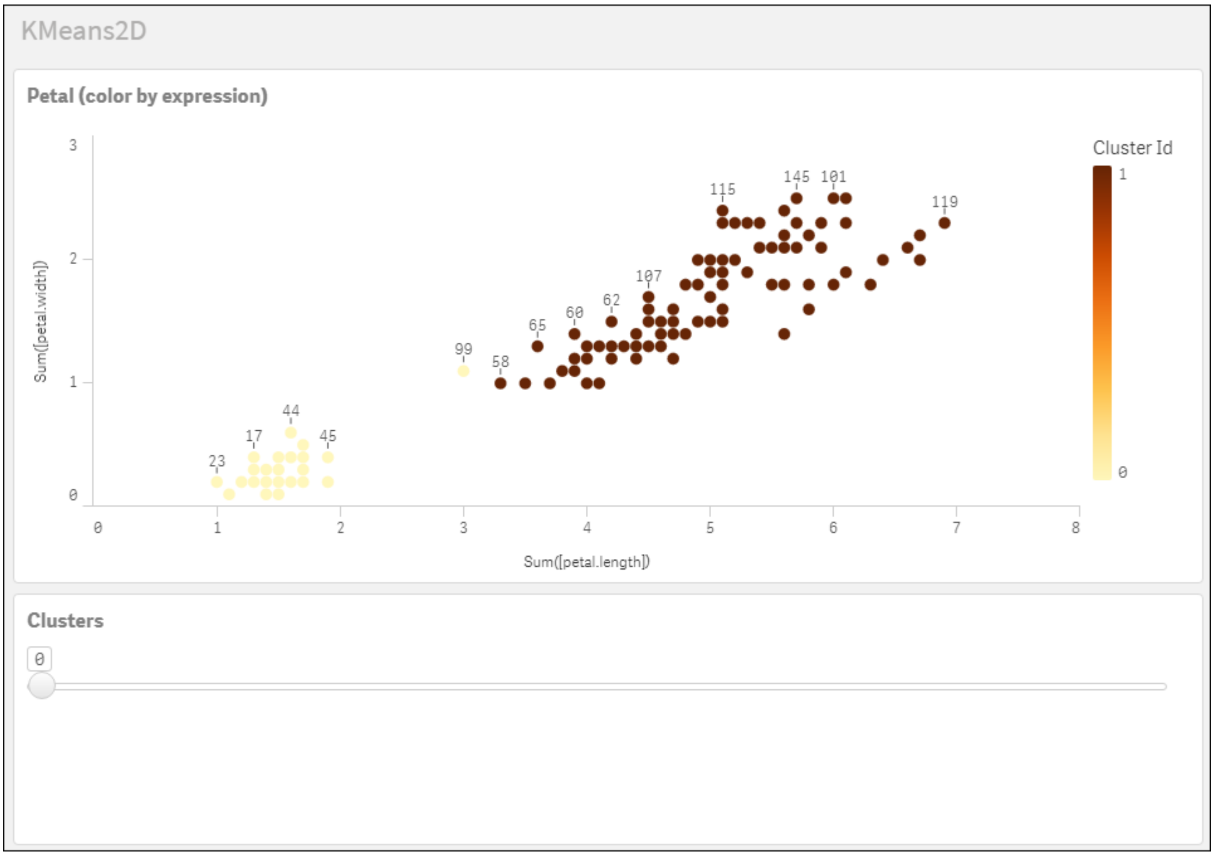
叢集依花瓣 (依運算式著色) 圖表上的運算式著色



自動叢集

KMeans 函數使用稱為深度差 (DeD) 的方法支援自動叢集。若使用者對叢集數量設定 0，會判定該資料集的最佳叢集數量。請注意，若沒有明確以整數傳回叢集數量 (*k*)，則會在 KMeans 演算法內計算。例如，若對 *KmeansPetalClusters* 的值在函數中指定 0，或透過變數輸入方塊設定 0，則會根據最佳叢集數量為資料集自動計算叢集指派。

KMeans 深度差方法可在 (k) 設定為 0 時判定最佳叢集數量



Iris 資料集：Qlik Sense 中資料載入編輯器的內嵌載入

```
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 - 圖表函數

KMeansND() 會套用 K 平均演算法叢集以評估圖表的列，而對於每個圖表列，會針對此資料點指派到的叢集顯示叢集 ID。叢集演算法使用的欄由參數 coordinate_1 和 coordinate_2, 等來判定，最多 n 欄。這些都是彙總。建立的叢集數量由 num_clusters 參數判定。

KMeansND 每個資料點傳回一個值。傳回的值是雙值，也是對應於每個資料點指派到的叢集的整數值。

語法：

```
KMeansND(num_clusters, num_iter, coordinate_1, coordinate_2 [,coordinate_3 [, ...]])
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
num_clusters	指定叢集數量的整數。
num_iter	叢集反覆項目的數量與重新初始化的叢集中心。
coordinate_1	計算第一個座標的彙總，通常是 x 軸 (屬於可從圖表製作的散佈圖)。其他參數會計算第二、第三和第四個座標等。

範例：圖表運算式

在此範例中，我們使用 *Iris* 資料集建立散佈圖，然後使用 KMeans 依運算式為資料著色。

我們也會建立 *num_clusters* 引數的變數，然後使用變數輸入方塊來變更叢集的數量。

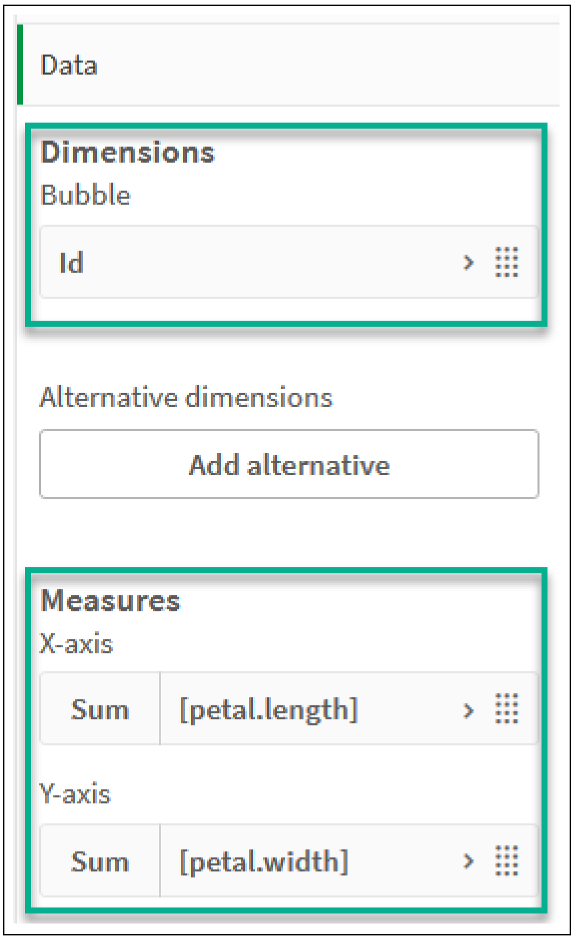
此外，我們會建立 *num_iter* 引數的變數，然後使用第二變數輸入方塊來變更反覆項目的數量。

Iris 資料集公開用於各種格式。我們已提供資料作為要在 Qlik Sense 中使用資料載入編輯器載入的內嵌表格。請注意，在此範圍中，我們已將 *Id* 欄新增至資料表格。

在 Qlik Sense 中載入資料後，我們進行下列事項：

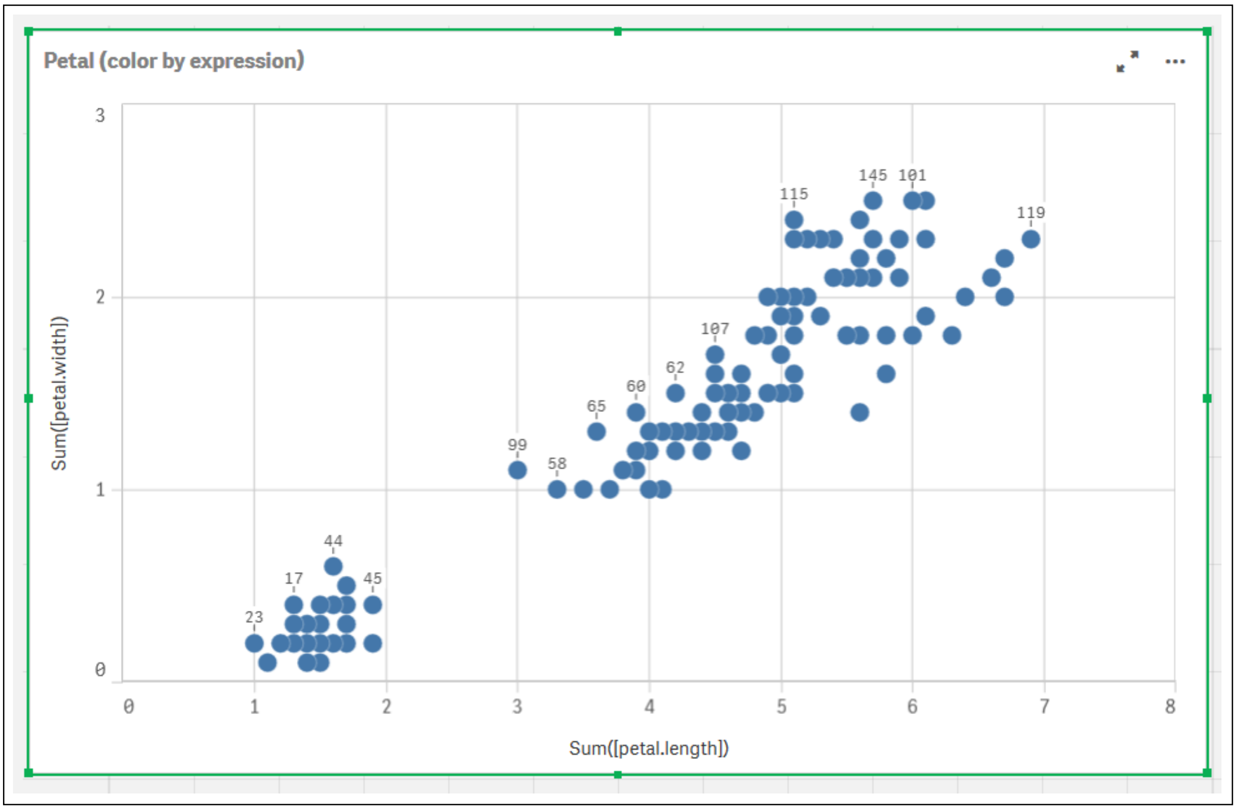
1. 將**散佈圖**圖表拖曳至新的工作表。將圖表命名為*花瓣 (依運算式著色)*。
2. 建立變數以指定叢集的數量。對於變數**名稱**，輸入 *KmeansPetalClusters*。對於變數**定義**，輸入 *=2*。
3. 建立變數以指定反覆項目的數量。對於變數**名稱**，輸入 *KmeansNumberIterations*。對於變數**定義**，輸入 *=1*。
4. 設定圖表的**資料**：
   i. 在**維度**之下，在**泡泡球**選擇欄位的 *id*。輸入標籤的叢集 ID。
   ii. 在**量值**之下，為 **X 軸**的運算式選擇 *Sum([petal.length])*。
   iii. 在**量值**之下，為 **Y 軸**的運算式選擇 *Sum([petal.width])*。

   花瓣 (依運算式著色) 圖表的資料設定

   資料點繪製於圖表上。

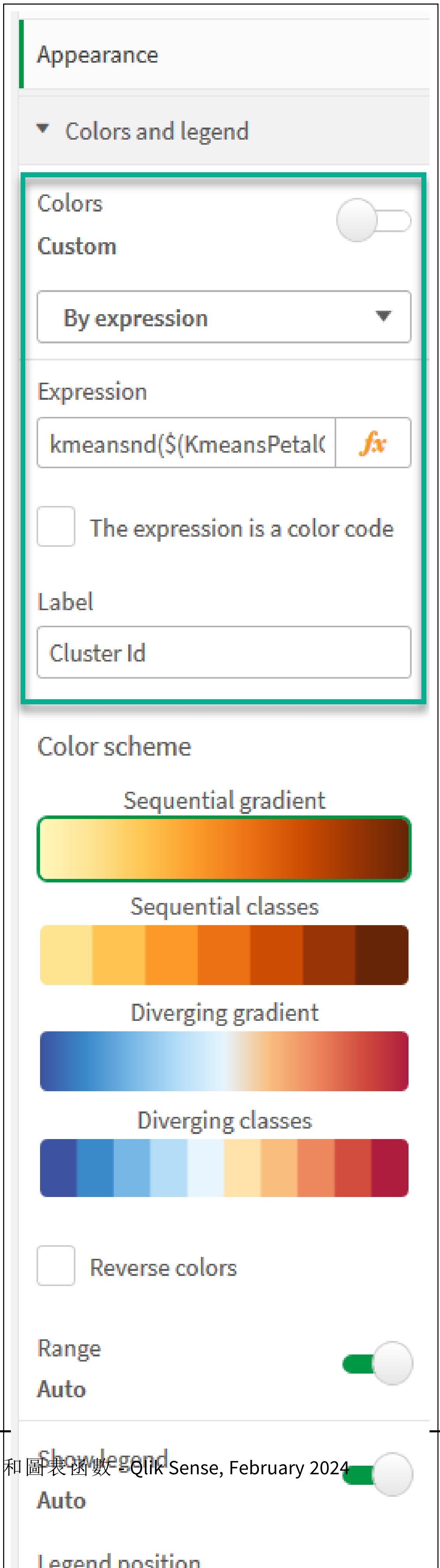
花瓣 (依運算式著色) 圖表的資料點



5. 設定圖表的**外觀**：
   i. 在**色彩和圖例**之下，為**色彩**選擇 **自訂**。
   ii. 選擇**依運算式**為圖表著色。
   iii. 在**運算式**輸入下列內容：*kmeansnd($(KmeansPetalClusters),$(KmeansNumberIterations), Sum([petal.length]), Sum([petal.width]),Sum([sepal.length]), Sum([sepal.width]))*
   請注意，*KmeansPetalClusters* 是我們設定為 *2* 的變數。*KmeansNumberIterations* 是我們設定為 *1* 的變數。
   或者，請輸入下列內容：*kmeansnd(2, 2, Sum([petal.length]), Sum([petal.width]),Sum([sepal.length]), Sum([sepal.width]))*
   iv. 取消選取**運算式為色彩代碼**核取方塊。

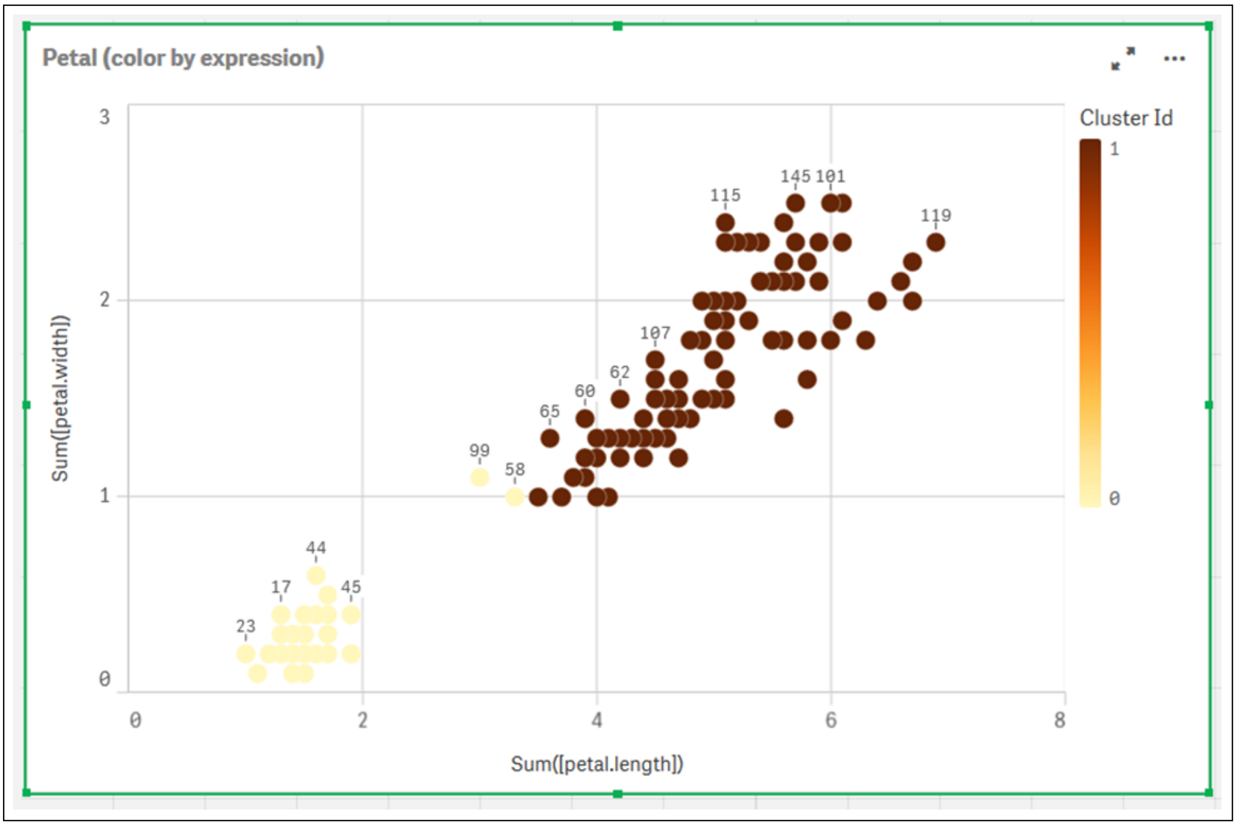
v. 在**標籤**輸入下列內容：*叢集 Id*

花瓣 (依運算式著色) 圖表的外觀設定


Appearance
Colors and legend
Colors
Custom
By expression
Expression
kmeansnd($(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

圖表上的兩個叢集依 KMeans 運算式著色。

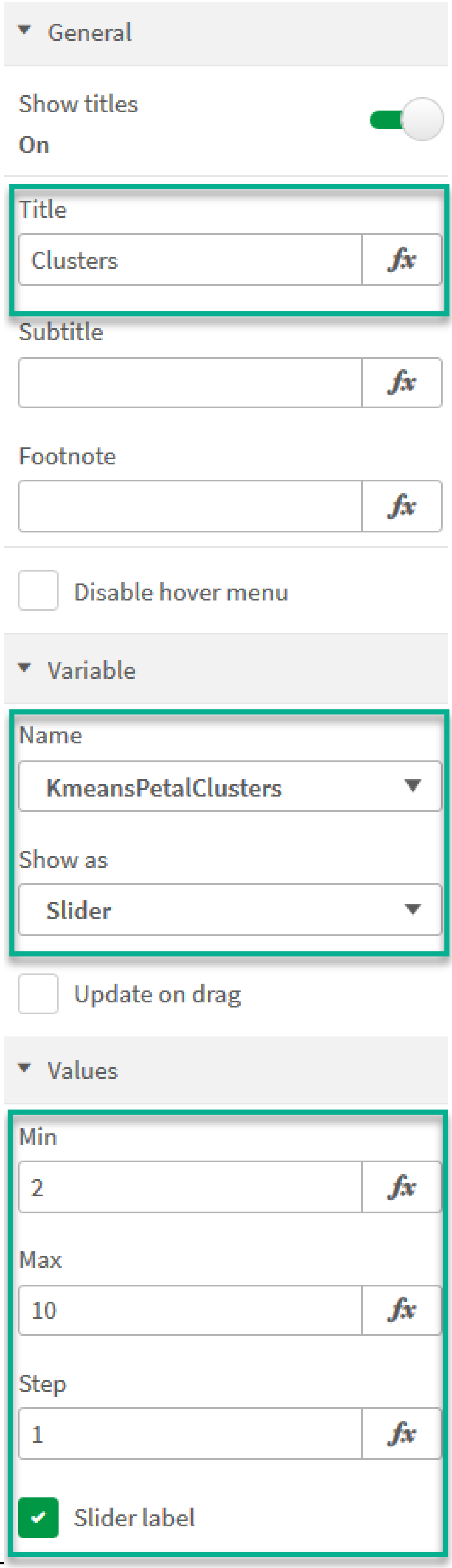
叢集依花瓣 (依運算式著色) 圖表上的運算式著色



6. 為叢集數量新增**變數輸入**方塊。
   i. 在**資產**面板的**自訂物件**之下，選擇 **Qlik 儀表板搭售**。若我們無法存取儀表板搭售，仍可使用我們建立的變數變更叢集數量，或直接作為運算式中的整數。
   ii. 將**變數輸入**方塊拖曳到工作表。
   iii. 在**外觀**之下，按一下**一般**。
   iv. 在**標題**輸入下列內容：*叢集*
   v. 按一下**變數**。
   vi. 在**名稱**選擇下列變數：*KmeansPetalClusters*。
   vii. 在**顯示為**選擇**滑桿**。

viii. 選擇**值**, 並按需要進行設定。

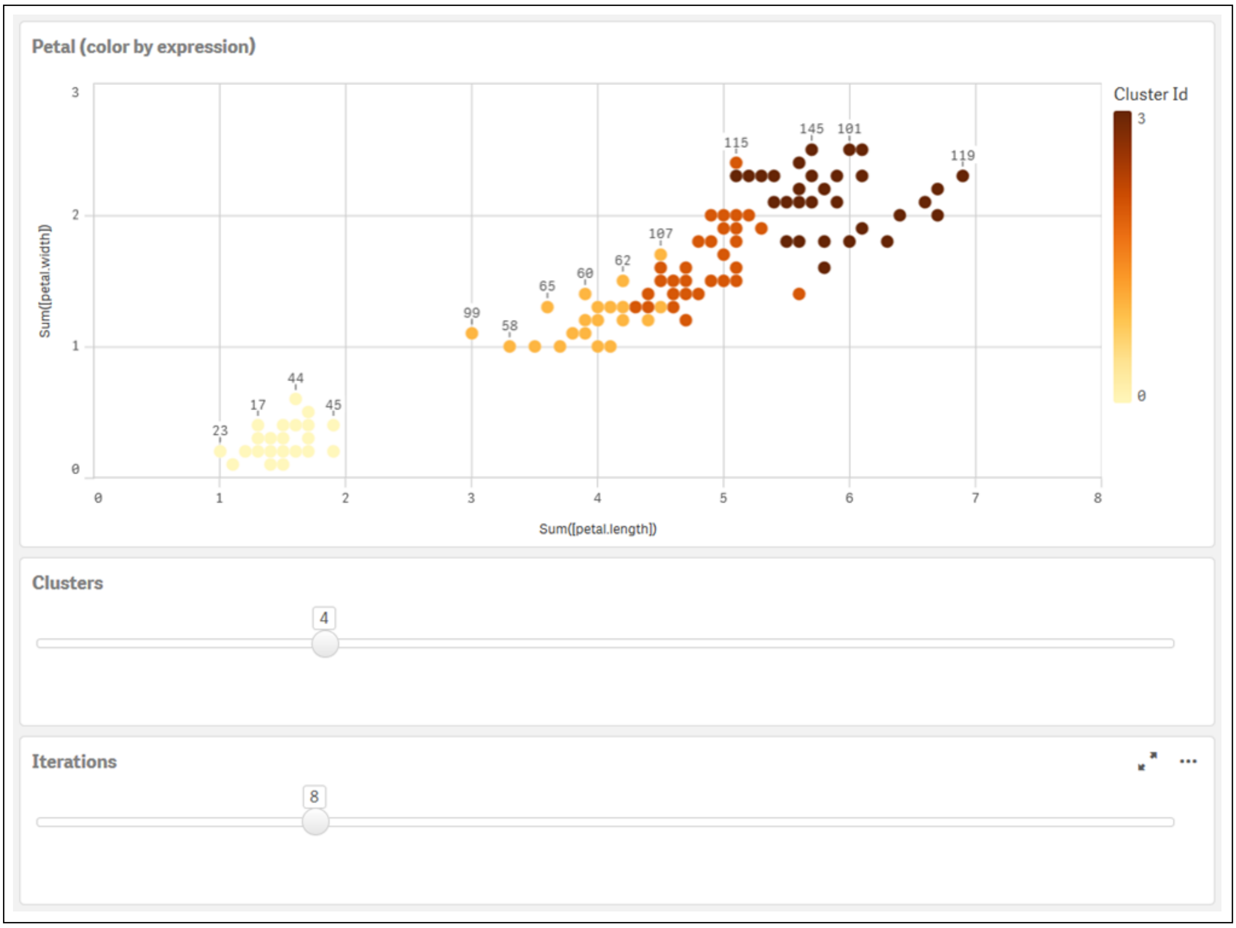
叢集變數輸入方塊的外觀


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. 為反覆項目數量新增**變數輸入**方塊。
    i. 將**變數輸入**方塊拖曳到工作表。
    ii. 在**外觀**之下，選擇**一般**。
    iii. 在**標題**輸入下列內容：*反覆項目*
    iv. 在**外觀**之下，選擇**變數**。
    v. 在**名稱**之下選擇下列變數：*KmeansNumberIterations*。
    vi. 按需求進行其他設定。

    我們現在可以使用變數輸入方塊中的滑桿來變更叢集和反覆項目的數量。

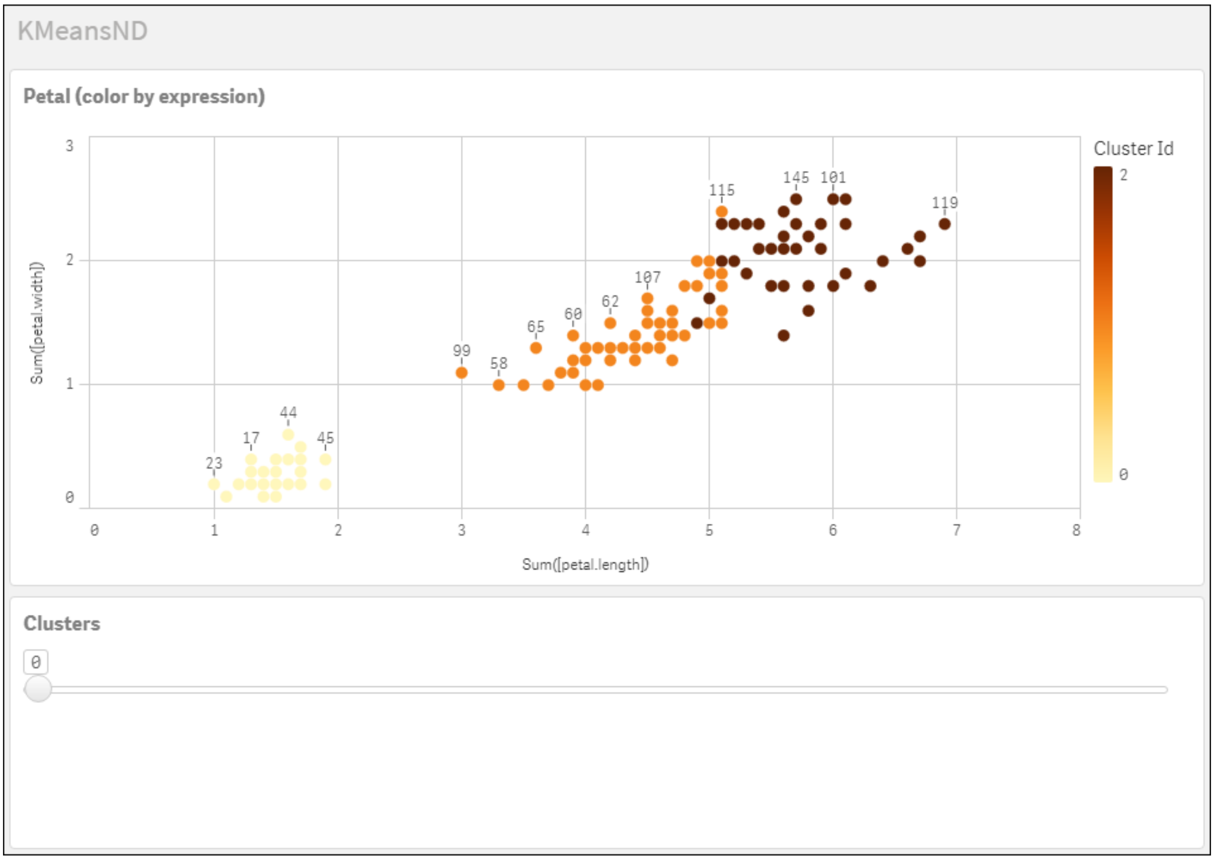
叢集依花瓣 (依運算式著色) 圖表上的運算式著色



自動叢集

KMeans 函數使用稱為深度差 (DeD) 的方法支援自動叢集。若使用者對叢集數量設定 0，會判定該資料集的最佳叢集數量。請注意，若沒有明確以整數傳回叢集數量 (*k*)，則會在 KMeans 演算法內計算。例如，若對 *KmeansPetalClusters* 的值在函數中指定 0，或透過變數輸入方塊設定 0，則會根據最佳叢集數量為資料集自動計算叢集指派。就 Iris 資料集而言，若對叢集數量選取 0，則演算法會為此資料集判定 (自動叢集) 最佳叢集數量 (3)。

KMeans 深度差方法可在 (k) 設定為 0 時判定最佳叢集數量。



Iris 資料集:Qlik Sense 中資料載入編輯器的內嵌載入

```
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 - 圖表函數

KMeansCentroid2D() 會套用 K 平均演算法叢集以評估圖表的列，而對於每個圖表列，會針對此資料點指派到的叢集顯示該叢集的所需座標。叢集演算法使用的欄由參數 coordinate_1 和 coordinate_2 分別判定。這些都是彙總。建立的叢集數量由 num_clusters 參數判定。可以選擇透過規範參數來正規化資料。

KMeansCentroid2D 每個資料點傳回一個值。傳回的值是雙值，也是對應於資料點指派到的叢集中心的位置座標之一。

語法：

```
KMeansCentroid2D(num_clusters, coordinate_no, coordinate_1, coordinate_2 [, norm])
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
num_clusters	指定叢集數量的整數。
coordinate_no	所需的形心座標數 (例如對應至 x、y 或 z 軸)。
coordinate_1	計算第一個座標的彙總，通常是 x 軸 (屬於可從圖表製作的散佈圖)。其他參數 coordinate_2 會計算第二個座標。

引數	描述
norm	選用的正規化方法已在 KMeans 叢集之前套用至資料集。 可能值： 0 或「無」代表沒有正規化 1 或「zscore」代表 Z 分數正規化 2 或「minmax」代表最小-最大正規化 若沒有套用參數或者若套用的參數不正確，則不會套用任何正規化。 Z 分數根據功能平均值和標準差正規化資料。Z 分數無法確保每個功能都有相同的比例，但這在處理異常值時是優於最小-最大的方法。 最小-最大正規化可取用每一項的最小和最大值並重新計算每個資料點，以確保功能具有相同比例。

自動叢集

KMeans 函數使用稱為深度差 (DeD) 的方法支援自動叢集。若使用者對叢集數量設定 0，會判定該資料集的最佳叢集數量。請注意，若沒有明確以整數傳回叢集數量 (*k*)，則會在 KMeans 演算法內計算。例如，若對 *KmeansPetalClusters* 的值在函數中指定 0，或透過變數輸入方塊設定 0，則會根據最佳叢集數量為資料集自動計算叢集指派。

KMeansCentroidND - 圖表函數

KMeansCentroidND() 會套用 k-means 叢集以評估圖表的列，而對於每個圖表列，會針對此資料點指派到的叢集顯示該叢集的所需座標。叢集演算法使用的欄由參數 coordinate_1、coordinate_2 等來判定，最多 n 欄。這些都是彙總。建立的叢集數量由 num_clusters 參數判定。

KMeansCentroidND 每列傳回一個值。傳回的值是雙值，也是對應於資料點指派到的叢集中心的位置座標之一。

語法：

```
KMeansCentroidND((num_clusters, num_iter, coordinate_no, coordinate_1,
coordinate_2 [,coordinate_3 [, ...]])
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
num_clusters	指定叢集數量的整數。

引數	描述
num_iter	叢集反覆項目的數量與重新初始化的叢集中心。
coordinate_no	所需的形心座標數 (例如對應至 x、y 或 z 軸)。
coordinate_1	計算第一個座標的彙總，通常是 x 軸 (屬於可從圖表製作的散佈圖)。其他參數會計算第二、第三和第四個座標等。

自動叢集

KMeans 函數使用稱為深度差 (DeD) 的方法支援自動叢集。若使用者對叢集數量設定 0，會判定該資料集的最佳叢集數量。請注意，若沒有明確以整數傳回叢集數量 (*k*)，則會在 KMeans 演算法內計算。例如，若對 *KmeansPetalClusters* 的值在函數中指定 0，或透過變數輸入方塊設定 0，則會根據最佳叢集數量為資料集自動計算叢集指派。

STL_Trend - 圖表函數

STL_Trend 是時間序列分解函數。連同 **STL_Seasonal** 和 **STL_Residual**，此函數用來將時間序列分解為季節、趨勢和剩餘元件。在 STL 演算法的脈絡下，時間序列分解用來識別輸入指標和其他參數下的週期性季節模式和一般趨勢。**STL_Trend** 函數將會從時間序列資料中識別一般趨勢，獨立於季節模式和週期。

三個 STL 函數與透過簡單加總的輸入指標相關：

STL_Trend + **STL_Seasonal** + **STL_Residual** = 輸入指標

STL (使用 Loess 的季節和趨勢分解) 採用資料平滑技術，並透過其輸入參數，讓使用者調整所執行的計算週期性。此週期性決定如何在分析中分段輸入指標的時間維度。

至少，**STL_Trend** 為其 `period_int` 採用輸入指標 (`target_measure`) 和整數值，傳回浮動點值。輸入指標將為隨時間維度而異的彙總形式。或者，您也可以納入 `seasonal_smoother` 和 `trend_smoother` 的值，以調整平滑演算法。

您可以直接在圖表的運算式編輯器中輸入，或在量值新增**時間序列分解**修飾詞，以使用此函數。這也可以作為分析類型用於 Insights。

語法：

```
STL_Trend(target_measure, period_int [,seasonal_smoother [,trend_smoother]])
```

傳回的資料類型： 雙值

引數

引數	描述
target_measure	分解為「季節」和「趨勢」元件的量值。這應為隨著時間維度而異的量值，例如 Sum (Sales) 或 Sum(Passengers)。 這不可是常數值。

引數	描述
period_int	資料集的週期性。此參數是整數值，代表組成訊號的一個期間或季節週期的離散步驟數量。 例如，若時間序列針對該年的每季分為一個區段，您必須將 **period_int** 值設定為 4，以將週期性定義為年。
seasonal_ smoother	季節平滑器長度。這必須是奇數。季節平滑器透過期間數量使用季節變化中特定階段的資料。會從每個期間使用時間維度的一個離散步驟。季節平滑器指示用於平滑處理的期間數量。 例如，若以月份分隔時間維度，而期間為年 (12)，則會運算季節元件，會在該年和相鄰年份從相同月份的資料計算每年的每個特定月份。**seasonal_smoother** 值是用於平滑處理的年數。
trend_ smoother	趨勢平滑器長度。這必須是奇數。趨勢平滑器使用與 **period_int** 參數相同的時間刻度，其值是用於平滑處理的精度數量。 例如，若以月份分隔時間序列，則趨勢平滑器將會是用於平滑處理的月數。

STL_Trend 圖表函數通常用於組合下列函數：

相關函數

函數	互動
STL_Seasonal - 圖表函數 (page 1342)	這是用來運算時間序列季節元件的函數。
STL_Residual - 圖表函數 (page 1344)	將輸入指標分為季節和趨勢元件時，部分量值的變化將不適合兩個主元件的任一項。**STL_ Residual** 函數會運算此分解部分。

如需以完整範例顯示如何使用此功能的教學課程，請參閱 *教學課程 - Qlik Sense 中的時間序列分解 (page 1345)*。

STL_Seasonal - 圖表函數

STL_Seasonal 是時間序列分解函數。連同 **STL_Trend** 和 **STL_Residual**，此函數用來將時間序列分解為季節、趨勢和剩餘元件。在 STL 演算法的脈絡下，時間序列分解用來識別輸入指標和其他參數下的週期性季節模式和一般趨勢。**STL_Seasonal** 函數可以識別時

間序列內的季節模式，從依資料顯示的一般趨勢中將此分出。

三個 STL 函數與透過簡單加總的輸入指標相關：

STL_Trend + **STL_Seasonal** + **STL_Residual** = 輸入指標

STL (使用 Loess 的季節和趨勢分解) 採用資料平滑技術，並透過其輸入參數，讓使用者調整所執行的計算週期性。此週期性決定如何在分析中分段輸入指標的時間維度。

至少，**STL_Seasonal** 為其 `period_int` 採用輸入指標 (`target_measure`) 和整數值，傳回浮動點值。輸入指標將為隨時間維度而異的彙總形式。或者，您也可以納入 `seasonal_smoother` 和 `trend_smoother` 的值，以調整平滑演算法。

您可以直接在圖表的運算式編輯器中輸入，或在量值新增**時間序列分解**修飾詞，以使用此函數。這也可以作為分析類型用於 Insights。

語法：

```
STL_Seasonal(target_measure, period_int [,seasonal_smoother [,trend_
smoother]])
```

傳回的資料類型：雙值

引數

引數	描述
target_measure	分解為「季節」和「趨勢」元件的量值。這應為隨著時間維度而異的量值，例如 Sum (Sales) 或 Sum(Passengers)。 這不可是常數值。
period_int	資料集的週期性。此參數是整數值，代表組成訊號的一個期間或季節週期的離散步驟數量。 例如，若時間序列針對該年的每季分為一個區段，您必須將 **period_int** 值設定為 4，以將週期性定義為年。
seasonal_smoother	季節平滑器長度。這必須是奇數。季節平滑器透過期間數量使用季節變化中特定階段的資料。會從每個期間使用時間維度的一個離散步驟。季節平滑器指示用於平滑處理的期間數量。 例如，若以月份分隔時間維度，而期間為年 (12)，則會運算季節元件，會在該年和相鄰年份從相同月份的資料計算每年的每個特定月份。**seasonal_smoother** 值是用於平滑處理的年數。
trend_smoother	趨勢平滑器長度。這必須是奇數。趨勢平滑器使用與 **period_int** 參數相同的時間刻度，其值是用於平滑處理的精度數量。 例如，若以月份分隔時間序列，則趨勢平滑器將會是用於平滑處理的月數。

STL_Seasonal 圖表函數通常用於組合下列函數：

相關函數

函數	互動
STL_Trend - 圖表函數 (page 1341)	這是用來運算時間序列趨勢元件的函數。
STL_Residual - 圖表函數 (page 1344)	將輸入指標分為季節和趨勢元件時，部分量值的變化將不適合兩個主元件的任一項。**STL_Residual** 函數會運算此分解部分。

如需以完整範例顯示如何使用此功能的教學課程，請參閱 *教學課程 - Qlik Sense 中的時間序列分解 (page 1345)*。

STL_Residual - 圖表函數

STL_Residual 是時間序列分解函數。連同 **STL_Seasonal** 和 **STL_Trend**，此函數用來將時間序列分解為季節、趨勢和剩餘元件。在 STL 演算法的脈絡下，時間序列分解用來識別輸入指標和其他參數下的週期性季節模式和一般趨勢。執行此操作時，輸入指標中的部分變化不適合用於季節和趨勢元件中，將會定義為剩餘元件。**STL_Residual** 圖表函數擷取此計算部分。

三個 STL 函數與透過簡單加總的輸入指標相關：

STL_Trend + **STL_Seasonal** + **STL_Residual** = 輸入指標

STL (使用 Loess 的季節和趨勢分解) 採用資料平滑技術，並透過其輸入參數，讓使用者調整所執行的計算週期性。此週期性決定如何在分析中分段輸入指標的時間維度。

由於時間序列分解主要尋找資料中的季節和一般變化，剩餘的資訊會被視為三個元件中最不重要的元件。不過，不準確或週期性剩餘元件可協助識別計算中的問題，例如不正確的週期性設定。

至少，**STL_Residual** 為其 `period_int` 採用輸入指標 (`target_measure`) 和整數值，傳回浮動點值。輸入指標將為隨時間維度而異的彙總形式。或者，您也可以納入 `seasonal_smoother` 和 `trend_smoother` 的值，以調整平滑演算法。

您可以直接在圖表的運算式編輯器中輸入，或在量值新增**時間序列分解**修飾詞，以使用此函數。這也可以作為分析類型用於 Insights。

語法：

```
STL_Residual(target_measure, period_int [,seasonal_smoother [,trend_
smoother]])
```

傳回的資料類型: 雙值

引數

引數	描述
target_ measure	分解為「季節」和「趨勢」元件的量值。這應為隨著時間維度而異的量值，例如 Sum (Sales) 或 Sum(Passengers)。 這不可是常數值。
period_int	資料集的週期性。此參數是整數值，代表組成訊號的一個期間或季節週期的離散步驟數量。 例如，若時間序列針對該年的每季分為一個區段，您必須將 **period_int** 值設定為 4，以將週期性定義為年。
seasonal_ smoother	季節平滑器長度。這必須是奇數。季節平滑器透過期間數量使用季節變化中特定階段的資料。會從每個期間使用時間維度的一個離散步驟。季節平滑器指示用於平滑處理的期間數量。 例如，若以月份分隔時間維度，而期間為年 (12)，則會運算季節元件，會在該年和相鄰年份從相同月份的資料計算每年的每個特定月份。**seasonal_smoother** 值是用於平滑處理的年數。
trend_ smoother	趨勢平滑器長度。這必須是奇數。趨勢平滑器使用與 **period_int** 參數相同的時間刻度，其值是用於平滑處理的精度數量。 例如，若以月份分隔時間序列，則趨勢平滑器將會是用於平滑處理的月數。

STL_Residual 圖表函數通常用於組合下列函數：

相關函數

函數	互動
STL_Seasonal - 圖表函數 (page 1342)	這是用來運算時間序列季節元件的函數。
STL_Trend - 圖表函數 (page 1341)	這是用來運算時間序列趨勢元件的函數。

如需以完整範例顯示如何使用此功能的教學課程，請參閱 *教學課程 - Qlik Sense 中的時間序列分解 (page 1345)*。

教學課程 - Qlik Sense 中的時間序列分解

本教學課程呈現使用三個圖表函數以透過 STL 演算法分解時間序列。

本教學課程為每月使用航空公司的乘客數量，使用時間序列資料，以呈現 STL 演算法的功能。**STL_Trend**、**STL_Seasonal** 和 **STL_Residual** 圖表函數將用來建立視覺化。如需更多關於 Qlik Sense 中時間序列分解的資訊，請參閱 *時間序列分解函數 (page 1295)*。

建立應用程式

從建立新的應用程式並匯入資料集開始。

下載此資料集：

教學課程 - 時間序列分解

此檔案包含關於航空公司每月乘客數量的資料。

請執行下列動作：

1. 從中心按一下**建立新應用程式**。
2. 開啟應用程式並將 *Tutorial - Time series decomposition.csv* 拖曳到此。

準備並載入資料

為了讓 Qlik Sense 正確解譯 YearMonth 欄位，您可能需要使用資料管理員，以將欄位辨識為日期欄位，而非具有字串值的欄位。通常會自動處理此步驟，但在此案例中，日期以較不常見的 *YYYY-MM* 格式呈現。

1. 在資料管理員中，選取表格並按一下 。
2. 選取 *YearMonth* 欄位後，按一下 並將**欄位類型**設定為**日期**。
3. 在**輸入格式**之下，輸入 *YYYY-MM*。
4. 在**顯示格式**之下，輸入 *YYYY-MM* 並按一下**確定**。
   欄位現在應顯示行事曆圖示。
5. 按一下**載入資料**。

現在您已準備好開始使用 STL 函數，以視覺化方式呈現您的資料。

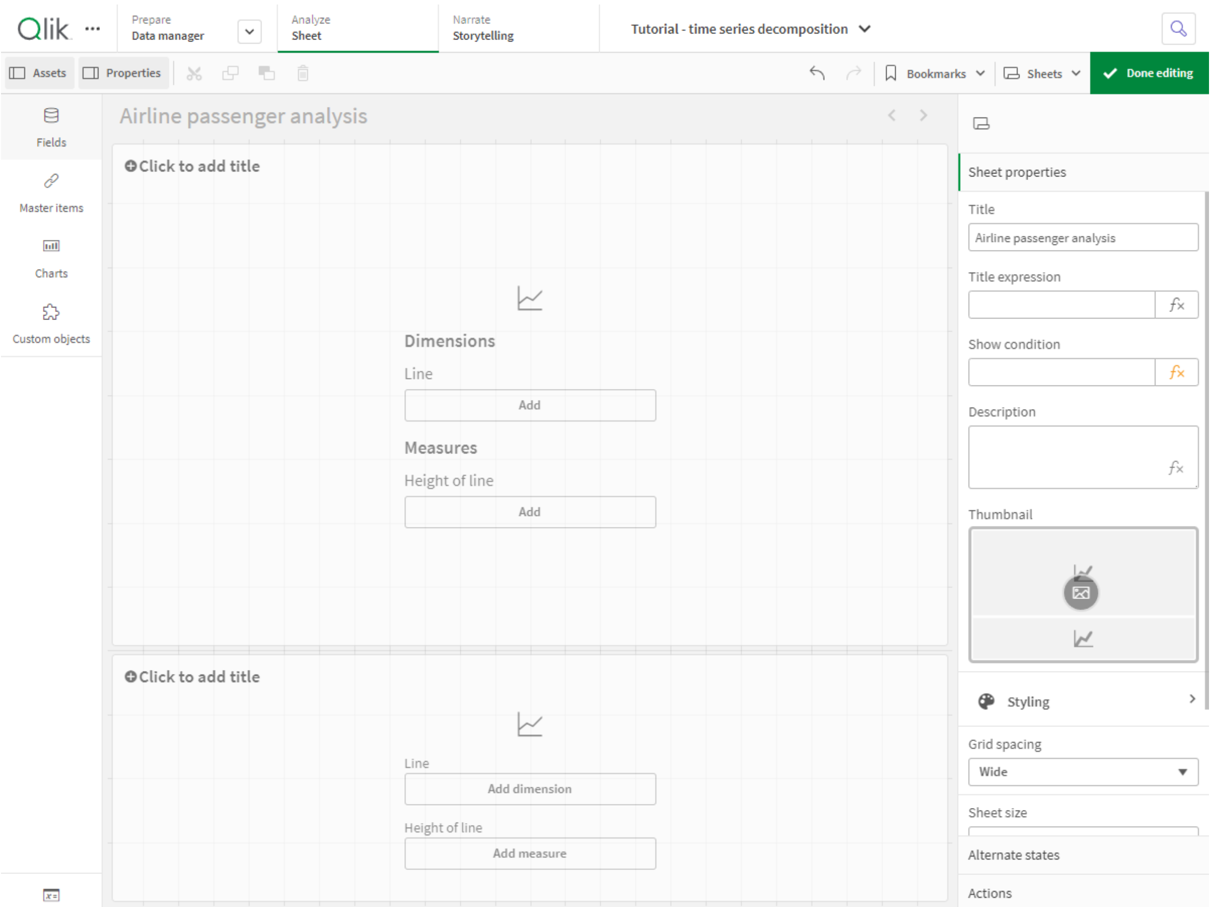
建立視覺化

接下來，您將會建立兩個折線圖，以呈現 **STL_Trend**、**STL_Seasonal** 和 **STL_Residual** 圖表函數的功能。

開啟新的工作表並提供標題。

將兩個折線圖新增至工作表。調整大小並重新放置圖表，以符合下圖。

Qlik Sense 空白應用程式工作表的格線外框



第一個折線圖：趨勢和季節元件

請執行下列動作：

1. 將標題*季節和趨勢*新增至第一個折線圖。
2. 新增 *YearMonth* 作為維度，並標示*日期*。
3. 新增下列量值並標示*每月乘客*：
   =Sum(Passengers)
4. 在**資料**之下，展開*每月乘客*量值並按一下**新增趨勢線**。
5. 將**類型**設定為**線性**。
   您將會比較此趨勢線與平滑的趨勢元件輸出。
6. 新增下列量值以繪製趨勢元件並標示*趨勢*：
   =STL_Trend(SUM(Passengers), 12)
7. 接下來，新增下列量值以繪製季節元件並標示*季節*：
   =STL_Seasonal(SUM(Passengers), 12)
8. 在**外觀** > **呈現**之下，將**捲軸**設定為**無**。
9. 保留預設色彩，或根據您的偏好變更。

第二個折線圖：剩餘元件

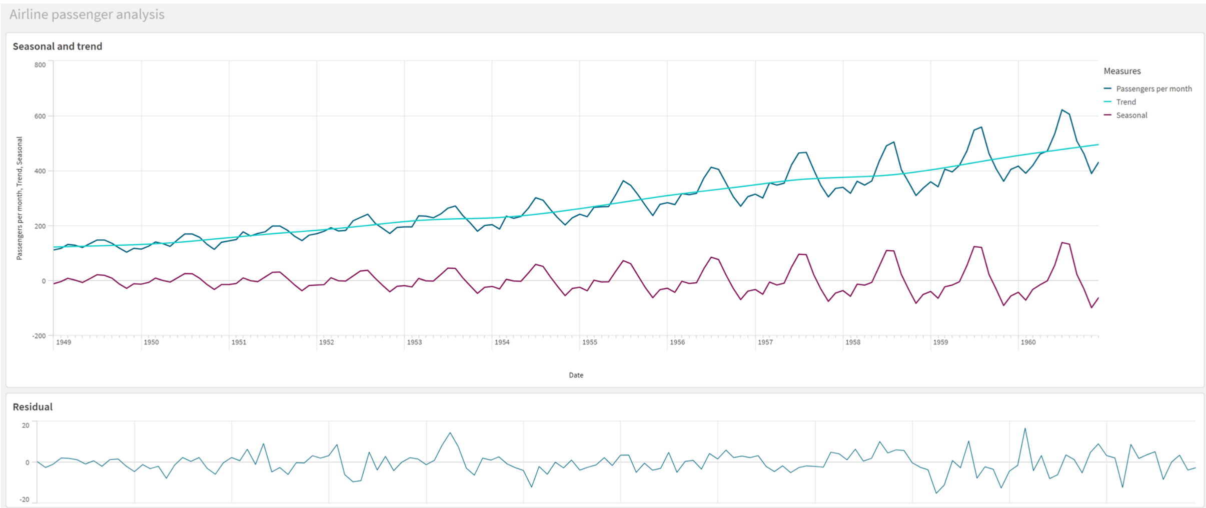
接下來，設定第二個折線圖。此視覺化將會顯示時間序列的剩餘元件。

請執行下列動作：

1. 將折線圖拖曳至工作表。新增標題*剩餘*。
2. 新增*日期*作為維度。
3. 新增下列量值並標示*剩餘*：
   =STL_Residual(SUM(Passengers), 12)
4. 在**外觀** > **呈現**之下，將**捲軸**設定為**無**。

您的工作表現在應如下所示。

用於航空公司乘客分析的 Qlik Sense 工作表



解譯並解釋資料

透過 STL 圖表函數，我們可以從時間序列資料取得一些深入資訊。

趨勢元件

趨勢元件中的統計資訊會去季節化。這會更容易查看隨時間變化的一般非重複變動。相較於*每月乘客*的直線趨勢線，STL 趨勢元件不會擷取變更趨勢。這顯示一些清楚的偏差，同時仍以可讀方式呈現資訊。STL 演算法中的平滑行為有助於擷取此內容。

STL 趨勢圖表中顯示的航空公司乘客數量下降可解釋為 1950 年代經濟衰退影響的一部分。

季節元件

去趨勢化的季節元件隔離整個時間序列的週期性變動，並從該分析部分移除一般趨勢資訊。我們從包括年月彙總的資料集開始。藉由此資料，顯然我們將資料分為一個月的細微程度。定義期間值 12 後，代表我們在一年 (十二個月) 週期將圖表設定為模型季節模式。

在資料中，航空公司的乘客在夏季月份有重複季節陡增模式，接著在冬季月份減少。這吻合夏季通常是熱門度假和旅遊時間的想法。我們也看見，隨著時間序列，這些季節週期大幅增加。

剩餘元件

剩餘元件的圖表顯示趨勢和季節分解中未擷取的所有資訊。剩餘元件包括統計雜訊，但也可以指示不正確的 STL 趨勢和季節函數引數設定。一般而言，若訊號的剩餘元件中有週期性振盪，或顯示的資訊明顯不是隨機，通常是時間序列中有資訊目前未在季節或趨勢元件擷取的徵兆。在此情況下，您需要重新造訪每個函數引數的定義，可能也要變更週期性。

平滑器值

由於我們沒有為趨勢和季節平滑器指定任何值，函數將會使用這些參數的預設值。在 Qlik Sense 中，STL 演算法中的預設平滑器值會產生有效結果。因此，在大部分的情況下，這些引數可以留在運算式外。

在三個 STL 函數的任一個將季節或趨勢平滑器引數設定為 0，會讓演算法使用預設值，而非值 0。

趨勢平滑器值使用圖表中指定的維度。由於 *YearMonth* 欄位依月份呈現資料，趨勢平滑器值將會是月數。季節平滑器將會反映定義的週期性。在此案例中，由於我們將一個期間定義為持續十二個月 (一年)，季節平滑器值是年數。雖然可能聽起來很混亂，但這其實代表若要尋找季節性，我們需要總覽一些季節。此數字是季節平滑器。

其他實用資訊

有鑒於季節週期幅度隨著時間增加，更進階的分析方法可以使用對數函數建立乘法分解。實際上，可以透過趨勢元件分割季節，以在 Qlik Sense 中建立相對幅度的簡單量值。完成後，我們注意到，隨著時間，每個週期的夏季高峰在相對幅度方面越來越大。不過，冬季低點的幅度沒有隨著時間增加。

8.23 統計分布函數

統計分佈函數會傳回既定輸入變數出現可能的不同結果的機率。您可以使用這些函數來計算資料點的可能值。

以下所述的三個統計分佈函數群組皆使用 Cephes 函數物件庫實作於 Qlik Sense。如需所使用演算法、準確性等的參考和詳細資料，請參考：Cephes library。Cephes 函數物件庫已獲授權使用。

- 機率函數可計算由所提供的值提出的分佈中的點的機率。
  - 頻率函數可用於離散分佈。
  - 密度函數可用於連續函數。
- Dist 函數可計算由所提供的值提出的分佈中的點的分佈的累積機率。
- Inv 函數可計算相反值 (既定的分佈累績機率)。

所有函數皆可用於資料載入指令碼和圖表運算式。

統計分布函數概述

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

BetaDensity

`BetaDensity()` 傳回 Beta 機率分佈。

```
BetaDensity (value, alpha, beta)
```

BetaDist

`BetaDist()` 傳回累積的 Beta 機率分佈。

```
BetaDist (value, alpha, beta)
```

BetaInv

`BetaINV()` 傳回累積的 Beta 逆機率分佈。

```
BetaInv (prob, alpha, beta)
```

BinomDist

`BinomDist()` 傳回累積的二項式機率分佈。

```
BinomDist (value, trials, trial_probability)
```

BinomFrequency

`BinomFrequency()` 傳回二項式機率分佈。

```
BinomFrequency (value, trials, trial_probability)
```

BinomInv

`BinomInv()` 傳回累積的二項式逆機率分佈。

```
BinomInv (prob, trials, trial_probability)
```

ChiDensity

`ChiDensity()` 會傳回 chi^2 分佈的單尾機率。chi^2 密度函數與 chi^2 測試相關聯。

```
ChiDensity (value, degrees_freedom)
```

ChiDist

`ChiDist()` 會傳回 chi^2 分佈的單尾機率。chi^2 分佈與 chi^2 測試相關聯。

```
ChiDist (value, degrees_freedom)
```

ChiInv

`ChiInv()` 會傳回 chi^2 分佈的逆單尾機率。

```
ChiInv (prob, degrees_freedom)
```

FDensity

`FDensity()` 傳回 `F` 機率分佈。

```
FDensity (value, degrees_freedom1, degrees_freedom2)
```

FDist

`FDist()` 傳回累積的 `F` 機率分佈。

```
FDist (value, degrees_freedom1, degrees_freedom2)
```

FInv

`FInv()` 傳回累積的 F 逆機率分佈。

```
FInv (prob, degrees_freedom1, degrees_freedom2)
```

GammaDensity

`GammaDensity()`傳回 Gamma 機率分佈。

```
GammaDensity (value, k, θ)
```

GammaDist

`GammaDist()` 傳回累積的 Gamma 機率分佈。

```
GammaDist (value, k, θ)
```

GammaInv

`GammaInv()` 傳回累積的 Gamma 逆機率分佈。

```
GammaInv (prob, k, θ)
```

NormDist

`NormDist()` 傳回指定平均數和標準差的累積常態分佈。如果 mean = 0 且 standard_dev = 1，則此函數會傳回標準常態分佈。

```
NormDist (value, mean, standard_dev)
```

NormInv

`NormInv()` 傳回指定平均數和標準差的逆累積常態分佈。

```
NormInv (prob, mean, standard_dev)
```

PoissonDist

`PoissonDist()` 傳回累積的 Poisson 機率分佈。

```
PoissonDist (value, mean)
```

PoissonFrequency

`PoissonFrequency()` 傳回 Poisson 機率分佈。

```
PoissonFrequency (value, mean)
```

PoissonInv

`PoissonInv()` 傳回累積的 Poisson 逆機率分佈。

```
PoissonInv (prob, mean)
```

TDensity

`TDensity()` 傳回學生 `t` 密度函數的值，其中的數值是要運算機率的 `t` 計算值。

```
TDensity (value, degrees_freedom, tails)
```

TDist

`TDist()` 傳回學生 t 分佈的機率，其中的數值是要計算機率的 t 計算值。

```
TDist (value, degrees_freedom, tails)
```

TInv

`TInv()` 以機率和自由度的函數傳回學生 t 分佈的 t 值。

```
TInv (prob, degrees_freedom)
```

另請參見：

統計彙總函數 (page 371)

BetaDensity

`BetaDensity()` 傳回 Beta 機率分佈。

語法：

```
BetaDensity(value, alpha, beta)
```

傳回的資料類型： 數字

引數

引數	描述
value	要評估分佈的值。值必須介於 0 與 1 之間。
alpha	第一個形狀參數會以一個正數來定義。該值為隨機變數的指數
beta	第二個形狀參數會以一個正數來定義。該值表示分母自由度的數字。

BetaDist

`BetaDist()` 傳回累積的 Beta 機率分佈。

語法：

```
BetaDist(value, alpha, beta)
```

傳回的資料類型： 數字

引數

引數	描述
value	要評估分佈的值。值必須介於 0 與 1 之間。
alpha	第一個形狀參數會以一個正數來定義。該值為隨機變數的指數
beta	第二個形狀參數會以一個正數來定義。該值為控制分佈形狀的指數。

此函數與 `BetaInv` 函數的關係如下：
`If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value`

BetaInv

`BetaINV()` 傳回累積的 Beta 逆機率分佈。

語法：

```
BetaInv(prob, alpha, beta)
```

傳回的資料類型： 數字

引數

引數	描述
prob	與 Beta 機率分佈相關聯的機率。必須是介於 0 和 1 之間的數字。
alpha	第一個形狀參數會以一個正數來定義。該值為隨機變數的指數
beta	第二個形狀參數會以一個正數來定義。該值為控制分佈形狀的指數。

此函數與 `BetaDist` 函數的關係如下：
`If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value`

BinomDist

`BinomDist()` 傳回累積的二項式機率分佈。

語法：

```
BinomDist(value, trials, trial_probability)
```

傳回的資料類型： 數字

引數

引數	描述
value	要評估分佈的值。該值必須為不小於零且不超過試驗次數的整數。
trials	表示試驗次數的一個正整數。
trial_probability	每項試驗的成功機率。必須為介於 0 與 1 之間的數字。

此函數與 `BinomInv` 函數的關係如下：
`If prob = BinomDIST(value, trials, trial_probability), then BinomInv(prob, trials, trial_probability) = value`

BinomFrequency

`BinomFrequency()` 傳回二項式機率分佈。

語法：

```
BinomFrequency(value, trials, trial_probability)
```

傳回的資料類型: 數字

引數

引數	描述
value	要評估分佈的值。該值必須為不小於零且不超過試驗次數的整數。
trials	表示試驗次數的一個正整數。
trial_probability	每項試驗的成功機率。必須為介於 0 與 1 之間的數字。

BinomInv

BinomInv() 傳回累積的二項式逆機率分佈。

語法:

```
BinomInv(prob, trials, trial_probability)
```

傳回的資料類型: 數字

引數

引數	描述
prob	與二項式機率分佈相關聯的機率。必須是介於 0 和 1 之間的數字。
trials	表示試驗次數的一個正整數。
trial_probability	每項試驗的成功機率。必須為介於 0 與 1 之間的數字。

此函數與 BinomDist 函數的關係如下:
If prob = BinomDist(value, trials, trial_probability), then BinomInv(prob, trials, trial_probability) = value

ChiDensity

ChiDensity() 會傳回 chi^2 分佈的單尾機率。chi^2 密度函數與 chi^2 測試相關聯。

語法:

```
ChiDensity(value, degrees_freedom)
```

傳回的資料類型: 數字

引數

引數	描述
value	要評估分佈的值。該值不得為負值。
degrees_freedom	表示分子自由度數字的正整數。

ChiDist

`ChiDist()` 會傳回 chi^2 分佈的單尾機率。chi^2 分佈與 chi^2 測試相關聯。

語法:

```
CHIDIST(value, degrees_freedom)
```

傳回的資料類型: 數字

引數:

引數

引數	描述
value	要評估分佈的值。該值不得為負值。
degrees_freedom	表示自由度數字的正整數。

此函數與 **ChiInv** 函數的關係如下:

```
If prob = CHIDIST(value,df), then CHIINV(prob, df) = value
```

限制:

所有的引數必須都是數值,否則會傳回 NULL。

範例與結果:

範例	結果
CHIDIST(8, 15)	傳回 0.9238

ChiInv

`ChiInv()` 會傳回 chi^2 分佈的逆單尾機率。

語法:

```
CHIINV(prob, degrees_freedom)
```

傳回的資料類型: 數字

引數:

引數

引數	描述
prob	與 chi^2 分佈關聯的機率。必須是介於 0 和 1 之間的數字。
degrees_freedom	表示自由度數字的整數。

此函數與 **ChiDist** 函數的關係如下:

```
If prob = CHIDIST(value,df), then CHIINV(prob, df) = value
```

限制：

所有的引數必須都是數值，否則會傳回 NULL。

範例與結果：

範例	**結果**
CHIINV(0.9237827, 15)	傳回 8.0000

FDensity

`FDensity()` 傳回 `F` 機率分佈。

語法：

```
FDensity(value, degrees_freedom1, degrees_freedom2)
```

傳回的資料類型： 數字

引數

引數	**描述**
value	要評估分佈的值。該值不得為負值。
degrees_freedom1	表示分子自由度數字的正整數。
degrees_freedom2	表示分母自由度數字的正整數。

FDist

`FDist()` 傳回累積的 `F` 機率分佈。

語法：

```
FDist(value, degrees_freedom1, degrees_freedom2)
```

傳回的資料類型： 數字

引數：

引數

引數	**描述**
value	要評估分佈的值。該值不得為負值。
degrees_freedom1	表示分子自由度數字的正整數。
degrees_freedom2	表示分母自由度數字的正整數。

此函數與 **FInv** 函數的關係如下：

`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

限制：

所有的引數必須都是數值，否則會傳回 NULL。

範例與結果：

範例	結果
FDIST(15, 8, 6)	傳回 0.0019

FInv

`FInv()` 傳回累積的 `F` 逆機率分佈。

語法：

```
FInv(prob, degrees_freedom1, degrees_freedom2)
```

傳回的資料類型： 數字

引數：

引數

引數	描述
prob	與 F-機率分佈相關聯的機率，且必須是介於 0 和 1 之間的數字。
degrees_freedom	表示自由度數字的整數。

此函數與 **FDist** 函數的關係如下：
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

限制：

所有的引數必須都是數值，否則會傳回 NULL。

範例與結果：

範例	結果
FINV(0.0019369, 8, 6)	傳回 15.0000

GammaDensity

`GammaDensity()`傳回 Gamma 機率分佈。

語法：

```
GammaDensity(value, k, θ)
```

傳回的資料類型: 數字

引數

引數	描述
value	要評估分佈的值。該值不得為負值。
k	形狀參數會以一個正數來定義。
θ	尺度參數會以一個正數來定義。

GammaDist

GammaDist() 傳回累積的 Gamma 機率分佈。

語法:

```
GammaDist(value, k, θ)
```

傳回的資料類型: 數字

引數

引數	描述
value	要評估分佈的值。該值不得為負值。
k	形狀參數會以一個正數來定義。
θ	尺度參數會以一個正數來定義。

此函數與 GammaINV 函數的關係如下:
`If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value`

GammaInv

GammaInv() 傳回累積的 Gamma 逆機率分佈。

語法:

```
GammaInv(prob, k, θ)
```

傳回的資料類型: 數字

引數

引數	描述
prob	與 Gamma 機率分佈相關聯的機率。必須是介於 0 和 1 之間的數字。
k	形狀參數會以一個正數來定義。
θ	尺度參數會以一個正數來定義。

此函數與 `GammaDist` 函數的關係如下:
`If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value`

NormDist

`NormDist()` 傳回指定平均數和標準差的累積常態分佈。如果 mean = 0 且 standard_dev = 1, 則此函數會傳回標準常態分佈。

語法:

```
NORMDIST(value, [mean], [standard_dev], [cumulative])
```

傳回的資料類型: 數字

引數:

引數

引數	描述
value	要評估分佈的值。
mean	表示分佈之算術平均數的選用值。 若您沒有陳述此引數, 則預設值為 0。
standard_dev	表示分佈之標準差的選用正值。 若您沒有陳述此引數, 則預設值為 1。
cumulative	您可以選擇性選取以使用標準正常分佈或累積分佈。 0 = 標準正常分佈 1 = 累積分佈 (預設)

此函數與 **NormInv** 函數的關係如下:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

限制:

所有的引數必須都是數值, 否則會傳回 NULL。

範例與結果:

範例	結果
NORMDIST(0.5, 0, 1)	傳回 0.6915

NormInv

`NormInv()` 傳回指定平均數和標準差的逆累積常態分佈。

語法:

```
NORMINV(prob, mean, standard_dev)
```

傳回的資料類型: 數字

引數:

引數

引數	描述
prob	與常態分佈關聯的機率，必須是介於 0 和 1 之間的數字。
mean	表示分佈之算術平均數的值。
standard_dev	表示分佈之標準差的正值。

此函數與 **NormDist** 函數的關係如下:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

限制:

所有的引數必須都是數值，否則會傳回 NULL。

範例與結果:

範例	結果
NORMINV(0.6914625, 0, 1)	傳回 0.5000

PoissonDist

`PoissonDist()` 傳回累積的 Poisson 機率分佈。

語法:

```
PoissonDist(value, mean)
```

傳回的資料類型: 數字

引數

引數	描述
`value`	要評估分佈的值。該值不得為負值。
`mean`	平均結果會以一個正數來定義。

此函數與 `PoissonInv` 函數的關係如下:
`If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value`

PoissonFrequency

`PoissonFrequency()` 傳回 Poisson 機率分佈。

語法:

```
PoissonFrequency(value, mean)
```

傳回的資料類型: 數字

引數

引數	描述
value	要評估分佈的值。該值不得為負值。
mean	平均結果會以一個正數來定義。

PoissonInv

`PoissonInv()` 傳回累積的 Poisson 逆機率分佈。

語法:

```
PoissonInv(prob, mean)
```

傳回的資料類型: 數字

引數

引數	描述
prob	與 Poisson 機率分佈相關聯的機率。必須是介於 0 和 1 之間的數字。
mean	平均結果會以一個正數來定義。

此函數與 `PoissonDIST` 函數的關係如下:
`If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value`

TDensity

`TDensity()` 傳回學生 `t` 密度函數的值，其中的數值是要運算機率的 `t` 計算值。

語法:

```
TDensity(value, degrees_freedom)
```

傳回的資料類型: 數字

引數

引數	描述
value	要評估分佈的值。該值不得為負值。
degrees_freedom	表示自由度數字的正整數。

TDist

`TDist()` 傳回學生 `t` 分佈的機率，其中的數值是要計算機率的 `t` 計算值。

語法:

```
TDist(value, degrees_freedom, tails)
```

傳回的資料類型: 數字

引數:

引數

引數	描述
value	要評估分佈的值。該值不得為負值。
degrees_freedom	表示自由度數字的正整數。
tails	必須是 1 (單尾分佈) 或 2 (雙尾分佈)。

此函數與 **TInv** 函數的關係如下:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value`

限制:

所有的引數必須都是數值, 否則會傳回 NULL。

範例與結果:

範例	結果
TDIST(1, 30, 2)	傳回 0.3253

TInv

`TInv()` 以機率和自由度的函數傳回學生 `t` 分佈的 `t` 值。

語法:

```
TINV(prob, degrees_freedom)
```

傳回的資料類型: 數字

引數:

引數

引數	描述
prob	與 t 分佈相關聯的雙尾機率。必須是介於 0 和 1 之間的數字。
degrees_freedom	表示自由度數字的整數。

限制:

所有的引數必須都是數值, 否則會傳回 NULL。

此函數與 **TDist** 函數的關係如下:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value。`

範例與結果：

範例	結果
TINV(0.3253086, 30)	傳回 1.0000

8.24 字串函數

本節描述用來處理及操縱字串的函數。

所有函數都可同時用於資料載入指令碼和圖表運算式中，**Evaluate** 除外，它只能用於資料載入指令碼中。

字串函數概述

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

Capitalize

Capitalize() 會傳回所有單字都使用首字母大寫的字串。

```
Capitalize (text)
```

Chr

Chr() 會傳回相當於輸入整數的 Unicode 字元。

```
Chr (int)
```

Evaluate

Evaluate() 會尋找輸入文字字串是否能夠評估為有效的 Qlik Sense 運算式，如果可以，則以字串形式傳回運算式的值。如果輸入字串不是有效的運算式，則會傳回 NULL。

```
Evaluate (expression_text)
```

FindOneOf

FindOneOf() 會搜尋字串，以尋找出現的所提供字元集中任何字元的位置。會傳回搜尋集合中第一個出現的任何字元的位置，除非提供第三個引數 (使用大於 1 的值)。如果找不到符合的值，則會傳回 **0**。

```
FindOneOf (text, char_set[, count])
```

Hash128

Hash128() 會傳回組合輸入運算式值的 128 位元雜湊。結果為包含 22 個字元的字串。

```
Hash128 (expr{, expression})
```

Hash160

Hash160() 會傳回組合輸入運算式值的 160 位元雜湊。結果為包含 27 個字元的字串。

```
Hash160 (expr{, expression})
```

Hash256

Hash256() 會傳回組合輸入運算式值的 256 位元雜湊。結果為包含 43 個字元的字串。

```
Hash256 (expr{, expression})
```

Index

Index() 會搜尋字串，以尋找第 n 次出現所提供子字串的開始位置。可選的第三個引數會提供 n 的值，如果省略則為 1。負數值會從字串的結尾開始搜尋。字串中的位置編號是從 **1** 算起。

```
Index  (text, substring[, count])
```

IsJson

IsJson() 測試指定的字串是否包含有效的 JSON (JavaScript 物件標記法) 資料。您也可以驗證特定的 JSON 資料類型。

```
IsJson (json [, type])
```

JsonGet

JsonGet() 傳回 JSON (JavaScript 物件標記法) 資料字串的路徑。資料必須為有效的 JSON，但可以包含額外的空格或新行。

```
JsonGet (json, path)
```

JsonSet

JsonSet() 修改包含 JSON (JavaScript 物件標記法) 資料的字串。這可以透過路徑指定的新位置設定或插入 JSON 值。資料必須為有效的 JSON，但可以包含額外的空格或新行。

```
JsonSet(json, path, value)
```

KeepChar

KeepChar() 會傳回包含第一個字串 'text' 的字串，減去第二個字串 "keep_chars" 中未包含的字元。

```
KeepChar (text, keep_chars)
```

Left

Left() 會傳回包含輸入字串第一個 (最左側) 字元的字串，其中字元數由第二個引數決定。

```
Left (text, count)
```

Len

Len() 會傳回輸入字串的長度。

```
Len (text)
```

LevenshteinDist

LevenshteinDist() 傳回兩個字串之間的 Levenshtein 距離。這定義為將一個字串變更為另一個字串所需的最小單一字元編輯數量 (插入、刪除或替代)。該函數對於模糊字串比較很實用。

```
LevenshteinDist (text1, text2)
```

Lower

Lower() 會將輸入字串中的所有字元轉換為小寫形式。

```
Lower (text)
```

LTrim

LTrim() 會傳回修剪掉所有前置空格的輸入字串。

```
LTrim (text)
```

Mid

Mid() 會傳回輸入字串的部分，從第二個引數 'start' 定義的字元位置開始，並傳回第三個引數 'count' 定義的字元數目。如省略 'count'，則會傳回剩餘的輸入字串。如省略 'count'，則會傳回剩餘的輸入字串。輸入字串中的第一個字元編號為 1。

```
Mid (text, start[, count])
```

Ord

Ord() 會傳回輸入字串中第一個字元的 Unicode 字碼指標編號。

```
Ord (text)
```

PurgeChar

PurgeChar() 會傳回包含輸入字串 ('text') 中包括之字元的字串，第二個引數 ('remove_chars') 中出現的任何字元除外。

```
PurgeChar (text, remove_chars)
```

Repeat

Repeat() 會形成一個字串，包含的輸入字串重複第二個引數所定義的次數。

```
Repeat (text[, repeat_count])
```

Replace

將輸入字串內所有出現的指定子字串取代為另一個子字串之後，**Replace()** 會傳回該字串。此函數不遞迴並會從左至右運作。

```
Replace (text, from_str, to_str)
```

Right

Right() 會傳回包含輸入字串最後一個 (最右側) 字元的字串，其中字元數由第二個引數決定。

```
Right (text, count)
```

RTrim

RTrim() 會傳回修剪掉所有尾端空格的輸入字串。

```
RTrim (text)
```

SubField

SubField() 用來從上層字串欄位中擷取子字串元件，其中原始記錄欄位包含兩個或多個部分，以分隔符號分隔。

```
SubField (text, delimiter[, field_no ])
```

SubStringCount

SubStringCount() 會傳回輸入字串文字中指定子字串的出現次數。如果沒有任何相符項，則會傳回0。

```
SubStringCount (text, substring)
```

TextBetween

TextBetween() 會傳回輸入字串中在指定為分隔符號之字元之間出現的文字。

```
TextBetween (text, delimiter1, delimiter2[, n])
```

Trim

Trim() 會傳回修剪掉所有前置和尾端空格的輸入字串。

```
Trim (text)
```

Upper

Upper() 會將輸入字串中的所有字元轉換為運算式中所有文字字元的大寫形式。數字和符號會被忽略。

```
Upper (text)
```

Capitalize

Capitalize() 會傳回所有單字都使用首字母大寫的字串。

語法:

```
Capitalize(text)
```

傳回的資料類型: 字串

範例:圖表運算式

範例	**結果**
`Capitalize ( 'star trek' )`	傳回 'Star Trek'
`Capitalize ( 'AA bb cC Dd')`	傳回 'Aa Bb Cc Dd'

範例:載入指令碼

```
Load  String, Capitalize(String) Inline [String rHode iSland washingTon d.C. new york];
```

結果

字串	**Capitalize(String)**
rHode iSland	Rhode Island
washingTon d.C.	Washington D.C.
new york	New York

Chr

Chr() 會傳回相當於輸入整數的 Unicode 字元。

語法:

```
Chr(int)
```

傳回的資料類型: 字串

範例與結果:

範例	**結果**
Chr(65)	傳回字串 'A'
Chr(163)	傳回字串 '£'
Chr(35)	傳回字串 '#'

Evaluate

Evaluate() 會尋找輸入文字字串是否能夠評估為有效的 Qlik Sense 運算式，如果可以，則以字串形式傳回運算式的值。如果輸入字串不是有效的運算式，則會傳回 NULL。

語法:

```
Evaluate(expression_text)
```

傳回的資料類型: 雙值

此字串函數不可用於圖表運算式。

範例與結果:

函數範例	**結果**
`Evaluate ( 5 * 8 )`	傳回 '40'

載入指令碼範例

```
Load  Evaluate(String) as Evaluated, String Inline [String 4 5+3 0123456789012345678 Today()
];
```

結果

字串	**已評估**
4	4

字串	已評估
5+3	8
0123456789012345678	0123456789012345678
Today()	2022-02-02

FindOneOf

FindOneOf() 會搜尋字串，以尋找出現的所提供字元集中任何字元的位置。會傳回搜尋集合中第一個出現的任何字元的位置，除非提供第三個引數 (使用大於 1 的值)。如果找不到符合的值，則會傳回 **0**。

語法：

```
FindOneOf(text, char_set[, count])
```

傳回的資料類型： 整數

引數：

引數

引數	描述
text	原始字串。
char_set	要在 text 中搜尋的字元集。
count	定義要搜尋的任何字元的出現項目。例如，值 2 會搜尋第二個出現項目。

範例：圖表運算式

範例	結果
`FindOneOf( 'my example text string', 'et%s')`	傳回 '4'，因為 'e' 是範例字串中的第四個字元。
`FindOneOf( 'my example text string', 'et%s', 3)`	傳回 '12'，因為搜尋的是 e、t、% 或 s 中的任何字元，而 "t" 是第三個出現項目，位於範例字串的位置 12。
`FindOneOf( 'my example text string', '¤%&')`	傳回 '0'，因為字元 ¤、% 或 & 中沒有任何字元存在於範例字串中。

範例：載入指令碼

```
Load * Inline [SearchFor, Occurrence et%s,1 et%s,3 ¤%&,1]
```

結果

SearchFor	**Occurrence**	**FindOneOf('my example text string', SearchFor, Occurrence)**
et%s	1	4
et%s	3	12
¤%&	1	0

Hash128

Hash128() 會傳回組合輸入運算式值的 128 位元雜湊。結果為包含 22 個字元的字串。

語法:

```
Hash128(expr{, expression})
```

傳回的資料類型: 字串

範例:圖表運算式

範例	**結果**
Hash128 ('abc', 'xyz', '123')	傳回 'MA&5]6+3=:>:>G%S<U*S2+'。
Hash128 (Region, Year, Month) Note: Region, Year, and Month are table fields.	傳回 'G7*=6GKPJ(Z+)^KM?<$'A+'。

範例:載入指令碼

```
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 ];
```

結果

區域	**年**	**月**	**Hash128**
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2+
EU	2022	01	B40^K&[T@!;VB'XR]<5=/$
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!
US	2022	02	C6@#]4#_G-(]J7EQY#KRW0

Hash160

Hash160() 會傳回組合輸入運算式值的 160 位元雜湊。結果為包含 27 個字元的字串。

語法:

```
Hash160(expr{, expression})
```

傳回的資料類型: 字串

範例:圖表運算式

範例	**結果**
Hash160 ('abc', 'xyz', '123')	傳回 'MA&5]6+3=:>;>G%S<U*S2I:`=X*'。
Hash160 (Region, Year, Month) Note: Region, Year, and Month are table fields.	傳回 'G7*=6GKPJ (Z+)^KM?<$'AI.)?U$'。

範例:載入指令碼

```
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 ];
```

結果

區域	**年**	**月**	**Hash160**
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2I:`=X*
EU	2022	01	B40^K&[T@!;VB'XR]<5=//_F853
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!T"FWZ
US	2022	02	C6@#]4#_G-(]J7EQY#KRW`@KF+W

Hash256

Hash256() 會傳回組合輸入運算式值的 256 位元雜湊。結果為包含 43 個字元的字串。

語法:

```
Hash256(expr{, expression})
```

傳回的資料類型: 字串

範例:圖表運算式

範例	**結果**
Hash256 ('abc', 'xyz', '123')	傳回 '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.	傳回 'G7*=6GKPJ(Z+)^KM?<$'AI.)?U$#X2RB [:0ZP=+Z`F:'。

範例:載入指令碼

```
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 ];
```

結果

區域	年	月	Hash256
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2I:`=X*A.IO*8N\%Y7Q;YEJ
EU	2022	01	B40^K&[T@!;VB'XR]<5=//_F853?BE6'G&,YH*T'MF)
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!T"FWZT=4\#V`M%6_\0C>4
US	2022	02	C6@#]4#_G-(]J7EQY#KRW`@KF+W-0]`[Z8R+#'")=+0

Index

Index() 會搜尋字串，以尋找第 n 次出現所提供子字串的開始位置。可選的第三個引數會提供 n 的值，如果省略則為 1。負數值會從字串的結尾開始搜尋。字串中的位置編號是從 **1** 算起。

語法：

```
Index(text, substring[, count])
```

傳回的資料類型： 整數

引數：

引數

引數	描述
text	原始字串。
substring	要在 text 中進行搜尋的字元字串。
count	定義要搜尋的 **substring** 出現項目。例如，值 2 會搜尋第二個出現項目。

範例與結果：

範例	結果
Index('abcdefg', 'cd')	傳回 3
Index('abcdabcd', 'b', 2)	傳回 6 ('b' 的第二個出現項目)
Index('abcdabcd', 'b',-2)	傳回 2 (從結尾開始 'b' 的第二個出現項目)
`Left( Date, Index( Date,'-') -1 )` where **Date** = 1997-07-14	傳回 1997
`Mid( Date, Index( Date, '-', 2 ) -2, 2 )` where **Date** = 1997-07-14	傳回 07

範例：指令碼

```
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() 測試指定的字串是否包含有效的 JSON (JavaScript 物件標記法) 資料。您也可以驗證特定的 JSON 資料類型。

語法：

```
value IsJson(json [, type])
```

傳回的資料類型： 雙值

引數

引數	描述
json	要測試的字串。這可以包含額外的空間或新行。
type	指定要測試之 JSON 資料類型的選用引數。 • '值' (預設) • '物件' • '陣列' • '字串' • '數字' • '布林' • 'null'

範例：有效的 JSON 和類型

範例	結果
`IsJson('null')`	傳回 -1 (true)
`IsJson('"abc"', 'value')`	傳回 -1 (true)
`IsJson('"abc"', 'string')`	傳回 -1 (true)
`IsJson(123, 'number')`	傳回 -1 (true)

範例：無效的 JSON 或類型

範例	結果	描述
`IsJson('text')`	傳回 0 (false)	'text' 不是有效的 JSON 值

範例	結果	描述
`IsJson('"text"', 'number')`	傳回 0 (false)	'"text"' 不是有效的 JSON 數字
`IsJson('"text"', 'text')`	傳回 0 (false)	'text' 不是有效的 JSON 類型

JsonGet

JsonGet() 傳回 JSON (JavaScript 物件標記法) 資料字串的路徑。資料必須為有效的 JSON，但可以包含額外的空格或新行。

語法：

```
value JsonGet(json, path)
```

傳回的資料類型： 雙值

引數

引數	描述
json	字串包含 JSON 資料。
path	必須根據 RFC 6901 指定路徑。這將允許查閱 JSON 資料的內部屬性，而不必使用複雜的子字串或索引函數。

範例：有效的 JSON 和路徑

範例	結果
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '')`	傳回 '{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/a')`	傳回 '{"foo":"bar"}'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/a/foo')`	傳回 '"bar"'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b')`	傳回 '[123,"abc","ABC"]'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/0')`	傳回 '123'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/1')`	傳回 '"abc"'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/2')`	傳回 '"ABC"'

範例:無效的 JSON 或路徑

範例	結果	描述
`JsonGet('{"a":"b"}','/b')`	傳回 null	路徑沒有指向 JSON 資料的有效部分。
`JsonGet('{"a"}','/a')`	傳回 null	JSON 資料是無效的 JSON (成員「a」沒有值)。

JsonSet

JsonSet() 修改包含 JSON (JavaScript 物件標記法) 資料的字串。這可以透過路徑指定的新位置設定或插入 JSON 值。資料必須為有效的 JSON,但可以包含額外的空格或新行。

語法:

```
value JsonSet(json, path, value)
```

傳回的資料類型: 雙值

引數

引數	描述
json	字串包含 JSON 資料。
path	必須根據 RFC 6901 指定路徑。這允許建置 JSON 資料的內部屬性,而不必使用複雜的子字串或索引函數和串連。
value	使用 JSON 格式的新字串值。

範例:有效的 JSON、路徑和值

範例	結果
`JsonSet('{}','/a','"b"')`	傳回 '{"a":"b"}'
`JsonSet('[]','/0','"x"')`	傳回 '["x"]'
`JsonSet('"abc"','','123')`	傳回 123

範例:無效的 JSON、路徑或值

範例	結果	描述
`JsonSet('"abc"','/x','123')`	傳回 null	路徑沒有指向 JSON 資料的有效部分。
`JsonSet('{"a":{"b":"c"}}','a/b','"x"')`	傳回 null	路徑無效。
`JsonSet('{"a":"b"}','/a','abc')`	傳回 null	值不是有效的 JSON。必須以引號括住字串。

KeepChar

KeepChar() 會傳回包含第一個字串 'text' 的字串，減去第二個字串 "keep_chars" 中未包含的字元。

語法:

```
KeepChar(text, keep_chars)
```

傳回的資料類型: 字串

引數:

引數

引數	描述
text	原始字串。
keep_chars	包含 text 中要保存之字元的字串。

範例:圖表運算式

範例	結果
`KeepChar ( 'a1b2c3','123' )`	傳回 '123'。
`KeepChar ( 'a1b2c3','1234' )`	傳回 '123'。
`KeepChar ( 'a1b22c3','1234' )`	傳回 '1223'。
`KeepChar ( 'a1b2c3','312' )`	傳回 '123'。

範例:載入指令碼

```
T1:
Load
*,
keepchar(String1, String2) as KeepChar;
Load * inline [
String1, String2
'a1b2c3', '123'
];
```

結果

使用在載入指令碼中的 *KeepChar* 函數來顯示輸出的 Qlik Sense 表格。

String1	String2	KeepChar
a1b2c3	123	123

另請參見:

PurgeChar (page 1381)

Left

Left() 會傳回包含輸入字串第一個 (最左側) 字元的字串, 其中字元數由第二個引數決定。

語法:

```
Left(text, count)
```

傳回的資料類型: 字串

引數:

引數	描述
text	原始字串。
count	定義要在字串 **text** 左側部分中包括的字元數目。

範例:圖表運算式

範例	結果
`Left('abcdef', 3)`	傳回 'abc'

範例:載入指令碼

```
T1: Load  *,  left(Text,Start) as Left;        Load * inline [ Text, Start 'abcdef', 3 '2021-07-14', 4 '2021-07-14', 2 ];
```

結果

使用在載入指令碼中的 *Left* 函數來顯示輸出的 Qlik Sense 表格。

文字	開始	向左
abcdef	3	abc
2021-07-14	4	2021
2021-07-14	2	20

亦請參閱 *Index (page 1371)*, 允許更複雜的字串分析。

Len

Len() 會傳回輸入字串的長度。

語法:

```
Len(text)
```

傳回的資料類型: 整數

範例:圖表運算式

範例	**結果**
`Len('Peter')`	傳回 '5'

範例:載入指令碼

```
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 ];
```

結果

字串	**NewString**
this is a sample text string	This is a sample text string
capitalize first letter only	Capitalize first letter only

LevenshteinDist

LevenshteinDist() 傳回兩個字串之間的 Levenshtein 距離。這定義為將一個字串變更為另一個字串所需的最小單一字元編輯數量 (插入、刪除或替代)。該函數對於模糊字串比較很實用。

語法:

```
LevenshteinDist(text1, text2)
```

傳回的資料類型: 整數

範例:圖表運算式

範例	**結果**
`LevenshteinDist('Kitten','Sitting')`	傳回「3」

範例:載入指令碼

載入指令碼

```
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;
```

結果

ID	String_1	String_2	LevenshteinDistance
1	Silver	Sliver	2
2	Silver	SSiver	2
3	Silver	SSiveer	3
4	Gold	粗體	1
5	Gold	Bool	3
6	Gold	Bond	2
7	Ove	Ove	0
8	Ove	Uve	1
9	Ove	Üve	1
10	ABC	DEFG	4
11	ABC	abc	3
12	ABC	ビビビ	3
13	X		1
14	X	-	1
15	X	1	1

Lower

Lower() 會將輸入字串中的所有字元轉換為小寫形式。

語法:

```
Lower(text)
```

傳回的資料類型: 字串

範例:圖表運算式

範例	**結果**
`Lower('abcD')`	傳回 'abcd'

範例:載入指令碼

```
Load  String, Lower(String) Inline [String rHode iSland washingTon d.C. new york];
```

結果

字串	Lower(String)
rHode iSland	rhode island
washingTon d.C.	washington d.c.
new york	new york

LTrim

LTrim() 會傳回修剪掉所有前置空格的輸入字串。

語法:

```
LTrim(text)
```

傳回的資料類型: 字串

範例:圖表運算式

範例	結果
`LTrim( ' abc' )`	傳回 'abc'
`LTrim( 'abc ' )`	傳回 'abc '

範例:載入指令碼

```
Set verbatim=1;  T1:  Load *, len(LtrimString) as LtrimStringLength;      Load *, ltrim
(String) as LtrimString; Load *, len(String) as StringLength;              Load * Inline [
String '   abc    ' ' def  '];
```

範例中包括了 "Set verbatim=1" 陳述式,以確保在展示 ltrim 函數之前不會自動削減空間。如需更多資訊,請參閱Verbatim (page 194)。

結果

字串	StringLength	LtrimStringLength
def	6	5
abc	10	7

另請參見:

RTrim (page 1384)

Mid

Mid() 會傳回輸入字串的部分，從第二個引數 'start' 定義的字元位置開始，並傳回第三個引數 'count' 定義的字元數目。如省略 'count'，則會傳回剩餘的輸入字串。如省略 'count'，則會傳回剩餘的輸入字串。輸入字串中的第一個字元編號為 1。

語法：

```
Mid(text, start[, count])
```

傳回的資料類型： 字串

引數：

引數

引數	描述
text	原始字串。
start	定義要包括的 text 第一個字元位置的整數。
count	定義輸出字串的字串長度。如果省略，會包括 **start** 定義位置的所有字元。

範例：圖表運算式

範例	結果
`Mid('abcdef',3 )`	傳回 'cdef'
`Mid('abcdef',3, 2 )`	傳回 'cd'

範例：載入指令碼

```
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 ];
```

結果

使用在載入指令碼中的 *Mid* 函數來顯示輸出的 Qlik Sense 表格。

文字	開始	Mid1	計數	Mid2
abcdef	2	bcdef	3	bcd
abcdef	3	cdef	2	cd
210714	2	10714	3	107
210714	3	0714	2	07

另請參見：

Index (page 1371)

Ord

Ord() 會傳回輸入字串中第一個字元的 Unicode 字碼指標編號。

語法:

```
Ord(text)
```

傳回的資料類型: 整數

範例與結果:

範例: 圖表運算式

範例	結果
Ord('A')	傳回整數 65。
Ord('Ab')	傳回整數 65。

範例: 載入指令碼

```
//Guqin (Chinese: 古琴) - 7-stringed zithers T2: Load *,  ord(Chinese) as OrdUnicode,
        ord(Western) as OrdASCII;                    Load * inline [ Chinese, Western 古琴,
Guqin ];
```

結果:

中文	西方	OrdASCII	OrdUnicode
古琴	Guqin	71	21476

PurgeChar

PurgeChar() 會傳回包含輸入字串 ('text') 中包括之字元的字串，第二個引數 ('remove_chars') 中出現的任何字元除外。

語法:

```
PurgeChar(text, remove_chars)
```

傳回的資料類型: 字串

引數:

引數

引數	描述
text	原始字串。
remove_chars	包含 text 中要移除之字元的字串。

傳回的資料類型: 字串

範例:圖表運算式

範例	**結果**
`PurgeChar ( 'a1b2c3','123' )`	傳回 'abc'。
`PurgeChar ( 'a1b2c3','312' )`	傳回 'abc'。

範例:載入指令碼

```
T1:
Load
*,
purgechar(String1, String2) as PurgeChar;
Load * inline [
String1, String2
'a1b2c3', '123'
];
```

結果

使用在載入指令碼中的 *PurgeChar* 函數來顯示輸出的 Qlik Sense 表格。

String1	**String2**	**PurgeChar**
a1b2c3	123	abc

另請參見:

KeepChar (page 1375)

Repeat

Repeat() 會形成一個字串，包含的輸入字串重複第二個引數所定義的次數。

語法:

```
Repeat(text[, repeat_count])
```

傳回的資料類型: 字串

引數:

引數

引數	**描述**
text	原始字串。
repeat_count	定義字串 **text** 中要在輸出字串中重複的字元的次數。

範例:圖表運算式

範例	結果
`Repeat( ' * ', rating )` when **rating** = 4	傳回 '****'

範例:載入指令碼

```
T1: Load *, repeat(String,2) as Repeat; Load * inline [ String hello world! hOw aRe you? ];
```

結果

字串	重複
hello world!	hello world!hello world!
hOw aRe you?	hOw aRe you?hOw aRe you?

Replace

將輸入字串內所有出現的指定子字串取代為另一個子字串之後, **Replace()** 會傳回該字串。此函數不遞迴並會從左至右運作。

語法:

```
Replace(text, from_str, to_str)
```

傳回的資料類型: 字串

引數:

引數

引數	描述
text	原始字串。
from_str	輸入字串 **text** 內可能出現一次或多次的字串。
to_str	要取代字串 **text** 內所有出現的 **from_str** 的字串。

範例與結果:

範例	結果
`Replace('abccde','cc','xyz')`	傳回 'abxyzde'

另請參見:

Right

Right() 會傳回包含輸入字串最後一個 (最右側) 字元的字串, 其中字元數由第二個引數決定。

語法:

```
Right(text, count)
```

傳回的資料類型: 字串

引數:

引數

引數	描述
text	原始字串。
count	定義字串 **text** 最右側部分中要包括的字元數目。

範例:圖表運算式

範例	結果
`Right('abcdef', 3)`	傳回 'def'

範例:載入指令碼

```
T1:
Load
*,
right(Text,Start) as Right;
Load * inline [
Text, Start
'abcdef', 3
'2021-07-14', 4
'2021-07-14', 2
];
```

結果

使用在載入指令碼中的 *Right* 函數來顯示輸出的 Qlik Sense 表格。

文字	開始	向右
abcdef	3	def
2021-07-14	4	7-14
2021-07-14	2	14

RTrim

RTrim() 會傳回修剪掉所有尾端空格的輸入字串。

語法:

```
RTrim(text)
```

傳回的資料類型: 字串

範例:圖表運算式

範例	結果
`RTrim( ' abc' )`	傳回 ' abc'
`RTrim( 'abc ' )`	傳回 'abc'

範例:載入指令碼

```
Set verbatim=1;    T1:  Load *, len(RtrimString) as RtrimStringLength;         Load *, rtrim
(String) as RtrimString; Load *, len(String) as StringLength;            Load * Inline [
String '   abc    ' ' def  '];
```

範例中包括了 "Set verbatim=1" 陳述式,以確保在展示 rtrim 函數之前不會自動削減空間。如需更多資訊,請參閱Verbatim (page 194)。

結果

字串	StringLength	RtrimStringLength
def	6	4
abc	10	6

另請參見:

LTrim (page 1379)

SubField

SubField() 用來從上層字串欄位中擷取子字串元件,其中原始記錄欄位包含兩個或多個部分,以分隔符號分隔。

可以使用 **Subfield()** 函數,例如,為了從包含完整名稱的記錄清單中擷取名字和暱稱、路徑名稱的元件部分,或者從逗點分隔的表格中擷取資料。

如果您在 **LOAD** 陳述式中使用 **Subfield()** 函數,並且省略選用 field_no 參數,將針對每一個子字串產生一筆完整記錄。如果使用 **Subfield()** 載入數個欄位,則會建立所有組合的 Cartesian 產品。

語法:

```
SubField(text, delimiter[, field_no ])
```

傳回的資料類型： 字串

引數：

引數

引數	描述
text	原始字串。這可以是硬式編碼文字、變數、貨幣符號展開或其他運算式。
delimiter	輸入 **text** 內將字串分為元件部分的字元。
field_no	選用第三引數是一個整數，指定要傳回上層字串 **text** 中的哪個子字串。使用值 1 可傳回第一個子字串，使用值 2 可傳回第二個子字串，以此類推。 • 若 **field_no** 為正值，會從左至右擷取子字串。 • 若 **field_no** 為負值，會從右至左擷取子字串。

可以使用 SubField()，來取代函數的複雜組合 (例如 Len()、Right()、Left()、Mid() 及其他字串函數)。

範例：使用 SubField 的指令碼和圖表運算式

範例 - 指令碼和圖表運算式

基本範例

範例	結果
`SubField(S, ';' ,2)`	如果 **S** 為 'abc;cde;efg'，則傳回 'cde'。
`SubField(S, ';' ,1)`	如果 **S** 為空字串，則會傳回空字串 。
`SubField(S, ';' ,1)`	如果 **S** 為 ';'，則傳回空字串。
假定您有一個包含路徑名稱 vMyPath， `Set vMyPath=\Users\ext_jrb\Documents\Qlik\Sense\Apps;`。	在文字與影像圖表中，您可以新增量值，例如：`SubField(vMyPath, '\',-3)`，這將產生 'Qlik'，因為它是從變數 vMyPath 右端起的第三個子字串。

指令碼範例 1

載入指令碼

在資料載入編輯器中載入下列指令碼運算式和資料。

```
FullName:

LOAD * inline [
Name
```

```
'Dave Owen'
'Joe Tem'
];

SepNames:

Load Name,
SubField(Name, ' ',1) as FirstName,
SubField(Name, ' ',-1) as Surname
Resident FullName;
Drop Table FullName;
```

建立視覺化

在 Qlik Sense 工作表中建立具有 **Name**、**FirstName** 和 **SurName** 作為維度的表格視覺化。

結果

Name	FirstName	SurName
Dave Owen	Dave	Owen
Joe Tem	Joe	Tem

解釋

SubField() 函數將 **field_no** 引數設定為 1，以擷取 **Name** 的第一個子字串。由於 **field_no** 的值為正值，擷取字串時會按照由左至右的順序。第二個函數呼叫將 **field _no** 引數設定為 -1，以擷取第二個子字串，這按照由右至左的順序擷取子字串。

指令碼範例 2

載入指令碼

在資料載入編輯器中載入下列指令碼運算式和資料。

```
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 '|');
```

建立視覺化

在 Qlik Sense 工作表中建立具有 **Instrument**、**Player** 和 **Project** 作為維度的表格視覺化。

結果

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

解釋

此範例展示如何從相同的 **LOAD** 陳述式中使用 **Subfield()** 函數的多個執行個體 (每個執行個體均忽略 field_no 參數) 建立所有組合的笛卡兒乘積。將使用 **DISTINCT** 選項，以免產生重複記錄。

SubStringCount

SubStringCount() 會傳回輸入字串文字中指定子字串的出現次數。如果沒有任何相符項，則會傳回 0。

語法：

```
SubStringCount(text, sub_string)
```

傳回的資料類型： 整數

引數：

引數	描述
text	原始字串。
sub_string	輸入字串 **text** 內可能出現一次或多次的字串。

範例:圖表運算式

範例	結果
SubStringCount ('abcdefgcdxyz', 'cd')	傳回 '2'
SubStringCount ('abcdefgcdxyz', 'dc')	傳回 '0'

範例:載入指令碼

```
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 ];
```

結果

字串	SubStringCount_AB
aB/cd/ef/gh/Abc/abandoned	3
ABC:DEF:GHI:AB:CD:EF:GH	2

TextBetween

TextBetween() 會傳回輸入字串中在指定為分隔符號之字元之間出現的文字。

語法:

```
TextBetween(text, delimiter1, delimiter2[, n])
```

傳回的資料類型: 字串

引數:

引數	描述
text	原始字串。
delimiter1	指定要在 **text** 中搜尋的第一個分隔字元 (或字串)。
delimiter2	指定要在 **text** 中搜尋的第二個分隔字元 (或字串)。
n	定義要在其中進行搜尋的分隔符號配對。例如, 值 2 傳回 delimiter1 第二次出現與 delimiter2 第二次出現之間的字元。

範例:圖表運算式

範例	結果
TextBetween('<abc>', '<', '>')	傳回 'abc'
TextBetween('<abc><de>', '<', '>',2)	傳回 'de'

範例	結果
TextBetween('abc', '<', '>') TextBetween('<a<b', '<', '>')	兩個範例皆傳回 NULL。 若在字串中找不到任何分隔符號，則會傳回 NULL。
TextBetween('<>', '<', '>')	傳回零長度字串。
TextBetween('<abc>', '<', '>', 2)	傳回 NULL，因為 n 大於分隔符號的出現次數。

範例：載入指令碼

```
Load *, textbetween(Text,'<','>') as TextBetween, textbetween(Text,'<','>',2) as
SecondTextBetween; Load * inline [ Text <abc><de> <def><ghi><jkl> ];
```

結果

文字	TextBetween	SecondTextBetween
<abc><de>	abc	de
<def><ghi><jkl>	def	ghi

Trim

Trim() 會傳回修剪掉所有前置和尾端空格的輸入字串。

語法：

```
Trim(text)
```

傳回的資料類型： 字串

範例與結果：

範例：圖表運算式

範例	結果
Trim(' abc')	傳回 'abc'
Trim('abc ')	傳回 'abc'
Trim(' abc ')	傳回 'abc'

範例：載入指令碼

```
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');
```

範例中包括了 "Set verbatim=1" 陳述式，以確保在展示 trim 函數之前不會自動削減空間。如需更多資訊，請參閱Verbatim (page 194)。

結果：

字串	StringLength	TrimStringLength
def	6	3
abc	10	3

Upper

Upper() 會將輸入字串中的所有字元轉換為運算式中所有文字字元的大寫形式。數字和符號會被忽略。

語法：

```
Upper(text)
```

傳回的資料類型： 字串

範例：圖表運算式

範例	結果
`Upper(' abcD')`	傳回 'ABCD'

範例：載入指令碼

```
Load  String,Upper(String) Inline [String rHode iSland washingTon d.C. new york];
```

結果

字串	Upper(String)
rHode iSland	RHODE ISLAND
washingTon d.C.	WASHINGTON D.C.
new york	NEW YORK

8.25 系統函數

系統函數提供用來存取系統、裝置和 Qlik Sense 應用程式屬性的函數。

系統函數概述

概述之後，會進一步描述部分函數。對於那些函數，您可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

Author()

此函數傳回包含目前應用程式的作者屬性的字串。它可以用於資料載入指令碼與圖表運算式中。

不能在 Qlik Sense 的目前版本中設定作者屬性。如果您遷移一個 QlikView 文件，將會保留作者屬性。

ClientPlatform()

此函數傳回用戶端瀏覽器的使用者代理程式字串。它可以用於資料載入指令碼與圖表運算式中。

範例：

```
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/35.0.1916.114 Safari/537.36
```

ComputerName

此函數傳回包含電腦名稱 (由作業系統所傳回) 的字串。 它可以用於資料載入指令碼與圖表運算式中。

若電腦名稱超過 15 個字元，字串只會包含前 15 個字元。

```
ComputerName( )
```

DocumentName

此函數會傳回一個字串，其中包含目前 Qlik Sense 應用程式的名稱，不含路徑，但包含副檔名。它可以用於資料載入指令碼與圖表運算式中。

```
DocumentName( )
```

DocumentPath

此函數會傳回一個字串，其中包含目前 Qlik Sense 應用程式的完整路徑。它可以用於資料載入指令碼與圖表運算式中。

```
DocumentPath( )
```

標準模式下不支援這項功能。。

DocumentTitle

此函數會傳回一個字串，其中包含目前 Qlik Sense 應用程式的標題。它可以用於資料載入指令碼與圖表運算式中。

```
DocumentTitle( )
```

EngineVersion

此函數會以字串形式傳回完整的 Qlik Sense 引擎版本。

```
EngineVersion ()
```

GetCollationLocale

此指令碼函數傳回所使用定序地區設定的文化名稱。如果尚未設定變數 CollationLocale，則會傳回實際使用者機器地區設定。

```
GetCollationLocale( )
```

GetObjectField

GetObjectField() 會傳回維度名稱。**Index** 為選用整數，代表應傳回的維度。

```
GetObjectField - 圖表函數([index])
```

GetRegistryString

此函數會傳回 Windows 登錄中索引鍵的值。它可以用於資料載入指令碼與圖表運算式中。

```
GetRegistryString(path, key)
```

標準模式下不支援這項功能。。

GetSysAttr

此函數傳回租用戶和所選應用程式的空間網域屬性。它可以用於資料載入指令碼與圖表運算式中。

```
GetSysAttr (name)
```

若您在 Qlik Sense 用戶端管理 中使用此函數，只會傳回空的資料值。

IsPartialReload

如果目前的重新載入為部分，則此函數會傳回 - 1 (True)，否則會傳回 0 (False)。

```
IsPartialReload ()
```

InObject

InObject() 圖表函數評估另一個具有函數引數中指定 ID 的物件內是否包含目前物件。物件可以是工作表或視覺化。

```
InObject - 圖表函數(id_str)
```

ObjectId

ObjectId() 圖表函數傳回用以評估運算式的物件 ID。該函數採用選用引數，指定該函數顧慮哪個物件類型。物件可以是工作表或視覺化。此函數僅可用於圖表運算式。

```
ObjectId - 圖表函數([object_type_str])
```

OSUser

此函數傳回一個字串，該字串包含目前已連線的使用者名稱。它可以用於資料載入指令碼與圖表運算式中。

```
OSUser( )
```

> *在 Qlik Sense Desktop 與 Qlik Sense Client-Managed 行動版 中，此函數總是傳回 'Personal\Me'。*

ProductVersion
此函數會傳回完整的 Qlik Sense 版本，並以字串形式建置數字。

此函數會由 **EngineVersion()** 取代及替換。

```
ProductVersion ()
```

ReloadTime
此函數傳回完成最後一筆資料載入時的時間戳記。它可以用於資料載入指令碼與圖表運算式中。

```
ReloadTime( )
```

StateName
StateName() 會傳回使用此變數之視覺化的替代狀態名稱。例如，可使用 StateName 建立含動態文字與顏色的視覺化，以在視覺化狀態發生變更時進行反映。可在圖表運算式中使用此函數，但無法將其用於決定運算式所指的狀態。

```
StateName - 圖表函數()
```

EngineVersion

此函數會以字串形式傳回完整的 Qlik Sense 引擎版本。

語法：

```
EngineVersion()
```

GetSysAttr

此函數傳回租用戶和所選應用程式的空間網域屬性。它可以用於資料載入指令碼與圖表運算式中。

若您在 Qlik Sense 用戶端管理 中使用此函數，會傳回空的資料值。因此，您可以使用函數在 Qlik Sense 用戶端管理 中開發載入指令碼，而不會遇到錯誤，之後再將應用程式上傳至 Qlik Cloud。

若要存取 Qlik Cloud 函數的完整文件，請參閱 GetSysAttr - 指令碼和圖表函數。

InObject - 圖表函數

InObject() 圖表函數評估另一個具有函數引數中指定 ID 的物件內是否包含目前物件。物件可以是工作表或視覺化。

此函數可用來顯示工作表中的物件階層，從上層工作表物件到嵌於其他視覺化內部的視覺化。此函數可以隨 **if** 和 **ObjectId** 函數使用，以在應用程式中建立自訂導覽。

語法：

```
InObject(id_str)
```

傳回的資料類型： 布林

在 Qlik Sense 中，布林值 true 值以 -1 代表，而 false 值以 0 代表。

引數

引數	描述
id_str	呈現評估中物件 ID 的字串值。

可以從應用程式 URL 取得工作表 ID。對於視覺化，使用**開發人員**選項，以識別物件 ID 和物件類型的文字字串。

請執行下列動作：

1. 在分析模式中，將下列文字新增至 URL：
   /options/developer
2. 用滑鼠右鍵按一下視覺化，然後按一下 **開發人員**。
3. 在**屬性**之下，從對話方塊標頭取得物件 ID，然後從 **"qType"** 屬性取得物件類型。

限制：

在作為主項目之容器內部的物件 (例如按鈕) 中叫用時，此函數可以提供意外結果。此限制也適用於篩選窗格主項目，這是一些清單方塊的容器。其原因是主項目使用物件階層的方式。

InObject() 通常用於組合下列函數：

相關函數

函數	互動
if (page 524)	**if** 和 **ObjectId** 函數可以一起使用，以建立條件運算式。例如，視覺化可能會透過使用這些函數的運算式實現條件式著色。
ObjectId - 圖表函數 (page 1398)	與 **if** 類似，**ObjectId** 也能搭配 **InObject** 使用，以建立條件運算式。

範例 1 – 基本功能

圖表運算式和結果

下列基本範例呈現如何判定物件是否包含在另一個物件內部。在此案例中，我們將會使用工作表 ID 作為引數，檢查**文字與影像**物件是否位於工作表物件中。

請執行下列動作：

1. 開啟新的工作表並將**文字與影像**圖表拖曳到工作表。
2. 在屬性面板中，按一下**新增量值**。
3. 按一下 *fx* 可開啟運算式編輯器。
4. 在對話方塊中貼上以下運算式：
   `=InObject()`
5. 修改運算式，以納入工作表 ID 作為括弧之間的字串。
   例如，對於具有 ID 1234-5678 的工作表，您會使用以下內容：
   `=InObject('1234-5678')`
6. 按一下**套用**。

值 `-1` 會顯示在圖表中，指示運算式評估為真。

範例 2 – 具有條件式色彩的物件

圖表運算式和結果

概覽

下列範例呈現如何建立顯示不同著色的自訂導覽按鈕，以指示目前開啟的工作表。

開始方式是建立新的應用程式並開啟資料載入編輯器。在新的索引標籤中貼上以下載入指令碼。請注意，資料本身是預留位置，不會用於範例內容中。

載入指令碼

```
Transactions:

Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'4/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'7/26/2022',45.89
```

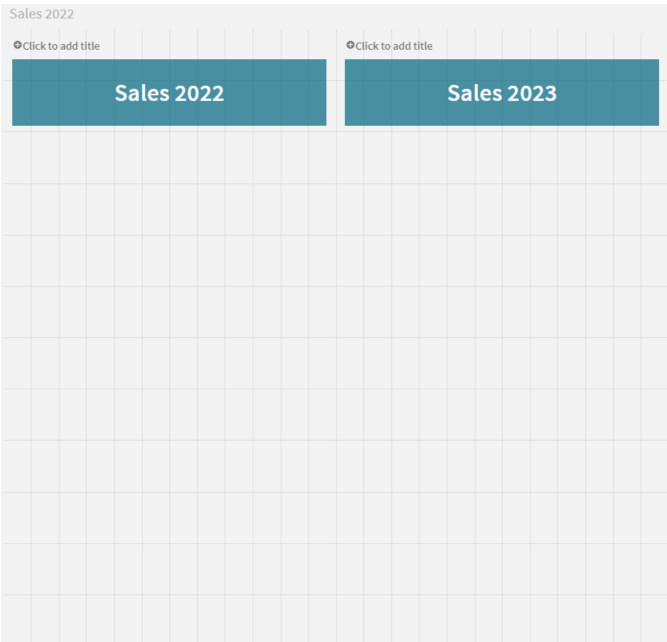
```
8198,'8/9/2022',36.23
8199,'9/22/2022',25.66
8200,'11/23/2022',82.77
8201,'12/27/2022',69.98
8202,'1/1/2023',76.11
8203,'2/8/2022',25.12
8204,'3/19/2022',46.23
8205,'6/26/2022',84.21
8206,'9/14/2022',96.24
8207,'11/29/2022',67.67
];
```

建立視覺化

載入資料並建立兩個新的工作表。分別將標題設為 *Sales 2022* 和 *Sales 2023*。

接下來，建置兩個將用來在兩個工作表之間導覽的按鈕物件。

請執行下列動作：

1. 將兩個**按鈕**物件新增至工作表。
2. 在**外觀 > 一般**之下，分別將每個按鈕的**標籤**設定為 *Sales 2022* 和 *Sales 2023*。
3. 排列按鈕以符合下圖。
   Sales 2022 工作表排列與兩個導覽按鈕
   

4. 選取 *Sales 2022* 按鈕，並在屬性面板展開**動作和導覽**。
5. 按一下**新增動作**，並在**導覽**之下選取**前往工作表**。
6. 在**工作表**之下，選取 *Sales 2022*。
7. 重複此按鈕動作設定，以將 **Sales 2023** 按鈕連結至 *Sales 2023* 工作表。
8. 用滑鼠右鍵按一下按鈕並選取 **新增至主項目**，以將按鈕轉換為主項目。

您現在可以在工作表上使用相同大小和排列方式，複製每個按鈕並在 *Sales 2023* 工作表中貼上。

建立條件式色彩

接下來，設定按鈕，若按鈕連結至目前開啟的工作表，就會是藍色；若連結至未開啟的工作表，就會是淺灰色。

請執行下列動作：

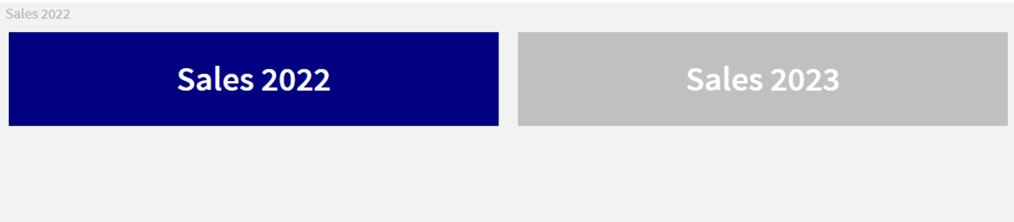
1. 開啟 *Sales 2022* 工作表並從 URL 取得工作表 ID。保持 *Sales 2022* 工作表開啟。
2. 按一下 **Sales 2022** 按鈕主項目並在屬性面板中選取**編輯**。
3. 在**外觀** > **背景**之下，選取以**依運算式**著色按鈕。
4. 在**運算式**中，貼上以下文字：
   =if(InObject(''), Blue(), LightGray())
5. 在上述運算式的括弧之間，貼上 *Sales 2022* 工作表的工作表 ID。

按鈕現在設定為在 *Sales 2022* 工作表開啟時變成藍色，並在未開啟時變成淺灰色。

為 *Sales 2023* 工作表重複上述說明，將 **Sales 2023** 按鈕主項目連結至 *Sales 2023* 工作表 ID。

每個工作表現在應有兩個按鈕，以藍色指示目前開啟的工作表。

目前顯示指示 2022 年銷售額的藍色 Sales 2022 工作表



IsPartialReload

如果目前的重新載入為部分，則此函數會傳回 - 1 (True)，否則會傳回 0 (False)。

語法：

```
IsPartialReload()
```

ObjectId - 圖表函數

ObjectId() 圖表函數傳回用以評估運算式的物件 ID。該函數採用選用引數，指定該函數顧慮哪個物件類型。物件可以是工作表或視覺化。此函數僅可用於圖表運算式。

語法：

```
ObjectId([object_type_str])
```

傳回的資料類型： 字串

函數的唯一引數 **object_type_str** 是選用性質，意指呈現物件類型的字串值。

引數

引數	描述
object_type_str	呈現評估中物件類型的字串值。

若在函數運算式中未指定任何引數, **ObjectId()** 會傳回使用運算式的物件 ID 。若要傳回其中顯示視覺化的工作表物件 ID, 使用 *ObjectId('sheet')*。

若視覺化物件內嵌於其他視覺化物件內部, 請在函數引數中指定所需物件類型以獲得不同結果。例如, 對於容器內部的**文字與 影像**圖表, 使用 *'text-image'* 以傳回**文字與 影像**物件, 並使用 *'container'* 以傳回容器 ID 。

請執行下列動作:

1. 在分析模式中, 將下列文字新增至 URL:
   /options/developer
2. 用滑鼠右鍵按一下視覺化, 然後按一下 **開發人員**。
3. 在**屬性**之下, 從對話方塊標頭取得物件 ID, 然後從 **"qType"** 屬性取得物件類型。

限制:

在作為主項目之容器內部的物件 (例如按鈕) 中叫用時, 此函數可以提供意外結果。此限制也適用於篩選窗格主項目, 這是一些清單方塊的容器。其原因是主項目使用物件階層的方式。

圖表運算式 *ObjectId('sheet')* 將會在這些情況下傳回空的字串, 其中 *ObjectId('masterobject')* 將會顯示擁有主項目的識別碼。

ObjectId() 通常用於組合下列函數:

相關函數

函數	互動
if (page 524)	**if** 和 **ObjectId** 函數可以一起使用, 以建立條件運算式。例如, 視覺化可能會透過使用這些函數的運算式實現條件式著色。
InObject - 圖表函數 (page 1394)	與 **if** 類似, **InObject** 也能搭配 **ObjectId** 使用, 以建立條件運算式。

範例 1 – 傳回圖表物件 ID

圖表運算式和結果
下列基本範例呈現如何傳回視覺化的 ID。

請執行下列動作：

1. 開啟新的工作表並將**文字與影像**圖表拖曳到工作表。
2. 在屬性面板中，按一下**新增量值**。
3. 按一下 *fx* 可開啟運算式編輯器。
4. 在對話方塊中貼上以下運算式：
   `=ObjectId()`
5. 按一下**套用**。

文字與影像物件的 ID 顯示在視覺化中。

可以使用下列運算式達到相同結果：

```
=ObjectId('text-image')
```

範例 2 – 傳回工作表 ID

圖表運算式和結果
下列基本範例呈現如何傳回其中顯示視覺化的工作表 ID。

請執行下列動作：

1. 開啟新的工作表並將**文字與影像**圖表拖曳到工作表。
2. 在屬性面板中，按一下**新增量值**。
3. 按一下 *fx* 可開啟運算式編輯器。
4. 在對話方塊中貼上以下運算式：
   `=ObjectId('sheet')`
5. 按一下**套用**。

工作表的 ID 顯示在視覺化中。

範例 3 – 巢狀運算式

圖表運算式和結果
下列範例顯示 **ObjectId()** 函數可以如何內嵌於其他運算式內部。

請執行下列動作：

1. 開啟新的工作表並將**文字與影像**圖表拖曳到工作表。
2. 在屬性面板中，按一下**新增量值**。
3. 按一下 *fx* 可開啟運算式編輯器。

4. 在對話方塊中貼上以下運算式：
   =if(InObject(ObjectId('text-image')), 'In Text & image', 'Not in Text & image')
5. 按一下**套用**。

文字 *In Text & image* (在文字與影像中) 會顯示在圖表中，指示運算式中參考的物件是**文字與影像**圖表。

如需更詳細的條件式著色使用範例，請參閱 *InObject - 圖表函數 (page 1394)* 的範例。

ProductVersion

此函數會傳回完整的 Qlik Sense 版本，並以字串形式建置數字。此函數會由 **EngineVersion()** 取代及替換。

語法：

```
ProductVersion()
```

StateName - 圖表函數

StateName() 會傳回使用此變數之視覺化的替代狀態名稱。例如，可使用 StateName 建立含動態文字與顏色的視覺化，以在視覺化狀態發生變更時進行反映。可在圖表運算式中使用此函數，但無法將其用於決定運算式所指的狀態。

語法：

```
StateName ()
```

Example 1:

動態文字

```
='Region - ' & if(StateName() = '$', 'Default', StateName())
```

Example 2:

動態色彩

```
if(StateName() = 'Group 1', rgb(152, 171, 206),

      if(StateName() = 'Group 2', rgb(187, 200, 179),

            rgb(210, 210, 210)

      )

)
```

8.26 表格函數

表格函數傳回目前讀取之資料表格的相關資訊。如果未指定任何表格名稱，並且在 **LOAD** 陳述式內使用函數，則會採用目前表格。

所有函數都可以用於資料載入指令碼中，而只有 **NoOfRows** 可用於圖表運算式中。

表格函數概述

概述之後，會進一步描述部分函數。對於那些函數，您可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

FieldName

FieldName 指令碼函數會傳回具有先前載入表格內指定編號之欄位的名稱。如果此函數用於 **LOAD** 陳述式內，則不得參考目前正在載入的表格。

```
FieldName (field_number ,table_name)
```

FieldNumber

FieldNumber 指令碼函數會傳回先前載入表格內指定欄位的編號。如果此函數用於 **LOAD** 陳述式內，則不得參考目前正在載入的表格。

```
FieldNumber (field_name ,table_name)
```

NoOfFields

NoOfFields 指令碼函數會傳回先前載入表格內的欄位數目。如果此函數用於 **LOAD** 陳述式內，則不得參考目前正在載入的表格。

```
NoOfFields (table_name)
```

NoOfRows

NoOfRows 函數會傳回先前載入表格中的列數 (記錄筆數)。如果此函數用於 **LOAD** 陳述式內，則不得參考目前正在載入的表格。

```
NoOfRows (table_name)
```

NoOfTables

此指令碼函數傳回先前載入表格的編號。

```
NoOfTables()
```

TableName

此指令碼函數傳回指定編號之表格的名稱。

```
TableName(table_number)
```

TableNumber

此指令碼函數傳回指定表格的編號。第一個表格編號為 0。

如果 table_name 不存在，則會傳回 NULL。

```
TableNumber(table_name)
```

範例：

在此範例中，我們想要使用已載入表格和欄位的相關資訊建立表格。

首先，我們載入樣本資料。這會建立將用來說明本節中描述之表格函數的兩個表格。

```
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 ;
```

接下來，我們使用 **NoOfTables** 函數，反覆運算已載入的表格，然後使用 **NoOfFields** 函數，反覆運算每一個表格的欄位，最後使用表格函數載入資訊。

```
//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;
```

產生的表格 Tables 將如下所示：

Load table

Table	**TableNo**	**TableRows**	**Field**	**FieldNo**
Characters	0	26	Alpha	1
Characters	0	26	Num	2
ASCII	1	191	Num	1
ASCII	1	191	AsciiAlpha	2
ASCII	1	191	AsciiNum	3

FieldName

FieldName 指令碼函數會傳回具有先前載入表格內指定編號之欄位的名稱。如果此函數用於 **LOAD** 陳述式內，則不得參考目前正在載入的表格。

語法：

```
FieldName(field_number ,table_name)
```

引數：

引數

引數	描述
field_number	您要參考之欄位的欄位編號。
table_name	包含您要參考之欄位的表格。

範例：

```
LET a = FieldName(4,'tab1');
```

FieldNumber

FieldNumber 指令碼函數會傳回先前載入表格內指定欄位的編號。如果此函數用於 **LOAD** 陳述式內，則不得參考目前正在載入的表格。

語法：

```
FieldNumber(field_name ,table_name)
```

引數：

引數

引數	描述
field_name	欄位的名稱。
table_name	包含欄位之表格的名稱。

如果欄位 field_name 不存在於 table_name 中，或者 table_name 不存在，則函數會傳回 0。

範例：

```
LET a = FieldNumber('Customer','tab1');
```

NoOfFields

NoOfFields 指令碼函數會傳回先前載入表格內的欄位數目。如果此函數用於 **LOAD** 陳述式內，則不得參考目前正在載入的表格。

語法：

```
NoOfFields(table_name)
```

引數：

引數

引數	描述
table_name	表格的名稱。

範例:

```
LET a = NoOfFields('tab1');
```

NoOfRows

NoOfRows 函數會傳回先前載入表格中的列數 (記錄筆數)。如果此函數用於 **LOAD** 陳述式內, 則不得參考目前正在載入的表格。

語法:

```
NoOfRows(table_name)
```

引數:

引數

引數	描述
table_name	表格的名稱。

範例:

```
LET a = NoOfRows('tab1');
```

8.27 三角與雙曲函數

本節描述用來執行三角與雙曲作業的函數。在所有函數中, 引數是解析為以弧度為測量單位之角度的運算式, 其中 **x** 應該解譯為實數。

所有角度都以弧度為測量單位。

所有函數皆可用於資料載入指令碼和圖表運算式。

cos
x 的餘弦函數 (Cosine)。結果是介於 -1 到 1 間的數字。

```
cos( x )
```

acos
x 的反餘弦函數。只有 -1≤**x**≤1 時才定義此函數。結果是介於 0 到 π 間的數字。

```
acos( x )
```

sin
x 的正弦函數 (Sine)。結果是介於 -1 到 1 間的數字。

```
sin( x )
```

asin
x 的反正弦函數。只有 -1≤**x**≤1 時才定義此函數。結果是介於 - π/2 到 π/2 間的數字。

```
asin( x )
```

tan

x 的正切函數 (Tangent)。結果是實數。

```
tan( x )
```

atan

x 的反正切函數。結果是介於 - π/2 到 π/2 間的數字。

```
atan( x )
```

atan2

反正切函數的二維一般化。傳回起點與座標 **x** 與 **y** 表示的點之間的角度。結果是介於 - π 到 + π 間的數字。

```
atan2( y,x )
```

cosh

x 的雙曲餘弦函數。結果是正實數。

```
cosh( x )
```

sinh

x 的雙曲正弦函數。結果是實數。

```
sinh( x )
```

tanh

x 的雙曲正切函數。結果是實數。

```
tanh( x )
```

acosh

x 的反雙曲餘弦函數。結果是正實數。

```
acosh( x )
```

asinh

x 的反雙曲正弦函數。結果是實數。

```
asinh( x )
```

atanh

x 的反雙曲正切函數。結果是實數。

```
atanh( x )
```

範例:

下列指令碼載入樣本表格, 然後載入值中包含已計算三角與雙曲作業的表格。

```
SampleData:
LOAD * Inline
```

```
[Value
-1
0
1];

Results:
Load *,
cos(Value),
acos(Value),
sin(Value),
asin(Value),
tan(Value),
atan(Value),
atan2(Value, Value),
cosh(Value),
sinh(Value),
tanh(Value)
RESIDENT SampleData;

Drop Table SampleData;
```

8.28 Window 函數

Window 函數使用多個列中的值執行計算，為每列分別產生值。讀取整個表格後，只能計算一次 Window 函數。

您可以使用 window 函數執行操作，例如：

- 比較列中的個別數值與欄內的平均、最大或最小值。
- 計算個別值的排名，欄內或整個表格內皆可。

window 函數不會變更表格中的記錄數量，但可以執行與彙總函數或關係函數和範圍函數類似的任務。

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

Window

Window 函數從多列執行計算，為每列分別產生值。

Window - 指令碼函數(input_expr, [partition1, partition2, ...], [sort_type, [sort_expr]],[filter_expr], [start_expr,end_expr]**)**[row_window_size]**)**

WRank

WRank 函數在 **Window** 內部執行排名計算。

WRank - 指令碼函數([**TOTAL**] expr[, mode[, fmt]]**)**

Window - 指令碼函數

Window() 從多列執行計算，為每列分別產生值。

您可以使用 **Window** 函數執行操作，例如：

- 比較列中的個別數值與欄內的平均、最大或最小值。
- 計算個別值的排名，欄內或整個表格內皆可。

Window 函數不會變更表格中的記錄數量，但仍然可以執行與彙總、關係和範圍函數類似的任務。

Window 函數在您所用表格的 LOAD 陳述式內必須有快取，以新增至表格。例如：

```
[Transactions]:
Load
	*,
	Window(avg(Expression1),[Num]);
LOAD
	TransLineID,
	TransID,
	"Num",
	Dim1,
	Dim2,
	Dim3,
	Expression1,
	Expression2,
	Expression3
FROM [lib://AttachedFiles/transactions.qvd] (qvd);
```

Window 支援一般函數，例如捨入或基本數字操作。例如：

```
Load *, Round(Window(Sum(Salary),Department)) as SumSalary
Load *, Window(Sum(Salary),Department) + 5 as SumSalary
```

您可以定義 **Window** 函數的滑動視窗。這設定在目前的列套用 **Window** 函數時使用的列數。例如，您可以將視窗設定為之前 3 列和之後 3 列。

語法：

Window (input_expr, [partition1, partition2, ...], [sort_type, [sort_expr]], [filter_expr], [start_expr,end_expr])

傳回的資料類型：新欄位新增至透過 LOAD 陳述式建立的結果表格。

引數：

引數

<table>
<tr><th>引數</th><th>描述</th></tr>
<tr><td>input_expr</td><td>透過函數計算並傳回輸入運算式。這必須是根據彙總的任何運算式，例如 <code>Median (Salary)</code>。例如：

<code>Window(Median(Salary)) as MedianSalary</code>
輸入也可以是沒有套用彙總的欄位名稱。在此情況下，Window 對此的看待方式如同 Only() 函數套用至該欄位。例如：

<code>Window(Salary,Department) as WSalary</code>

您也可以選擇定義透過輸入運算式分割。分割與透過 group by 子句達到的分組相同，差異為結果作為新欄新增至輸入表格。分割不會減少輸入表格的記錄數量。可以定義多個分割區欄位。

範例：

<pre>LOAD
Window(Max(Sales), City, 'ASC', OrderDate, Sales > 300)
+ AddMonths(OrderDate,-6) as MAX_Sales_City_Last_6_Mos,
Window(Avg(Sales), City, 'ASC', OrderDate, City = 'Portland')
 + AddMonths(OrderDate,-6) as Avg_Sales_Portland_Last_6_Mos,
Window(Max(Sales), City, 'ASC', OrderDate, Sales > 300)
 + AddMonths(OrderDate,-12) as MAX_Sales_City_Last_12_Mos;
LOAD
       City,
       Sales,
       OrderDate
       FROM [lib://AttachedFiles/Sales Data.xlsx]
(ooxml, embedded labels, table is [Sales Data]);</pre></td></tr>
<tr><td>partition1, partition2</td><td>在 input_expr 之後，您可以定任何分割區數量。分割區是定義哪些組合要用來套用彙總的欄位。每個分割區會獨立套件彙總。例如：

<code>Window(Avg(Salary), Unit, Department, Country) as AvgSalary</code>
總之，分割區為 Unit、Department 和 Country。

分割區不是強制性質，但為適當的欄位視窗化所必需。</td></tr>
</table>

引數	描述
sort_type, [sort_expr]]	也可以選擇指定排序類型和排序運算式。sort_type 可以有兩個值之一: • ASC: 遞增排序。 • DESC: 遞減排序。 若您定義 sort_type, 您需要定義排序運算式。此為決定分割區內列順序的運算式。 例如： `Window(RecNo(), Department, 'ASC', Year)` 在以上範例中, 分割區內的結果透過 *Year* 欄位遞增排序。 *主要是* ***RecNo*** *和* ***WRank*** *函數才需要排序類型和排序運算式。*

filter_expr	也可以新增篩選運算式。此為布林值運算式, 這決定是否應將記錄納入計算中。
	此參數可以完全省略, 結果應為沒有篩選。
	例如：
	`Window(avg(Salary), Department, 'ASC', Age, EmployeeID=3 Or EmployeeID=7)`
	`as wAvgSalary) as wAvgSalaryIfEmpIs3or7`

[start_ expr,end_ expr]	也可以選擇為滑動視窗功能設定引數。滑動視窗需要兩個引數:
	• 開始運算式: 要在視窗中納入的目前列之前的列數。
	• 結束運算式: 要在視窗中納入的目前列之後的列數。
	例如, 若您想要納入之前 3 列、目前列和下一列:
	`Window(concat(Text(Salary),'-'), Department, 'ASC', Age, Year>0,  -3, 1)`
	` as WSalaryDepartment`
	若要指示之前所有列或之後所有列, 您可以使用 **Unbounded()** 函數。例如, 若要納入之前所有列、目前列和下一列:
	`Window(concat(Text(Salary),'-'), Department, 'ASC', Age, Year>0,  UNBOUNDED(), 1)`
	` as WSlidingSalaryDepartment`
	例如, 若要納入從目前列算起第三列和之後所有列:
	`Window(concat(Text(Salary),'-'), Department, 'ASC', Age, Year>0,  3, UNBOUNDED())`
	` as WSlidingSalaryDepartment`

範例 - 新增包含彙總的欄位

範例: 新增包含彙總的欄位

載入指令碼

在資料載入編輯器中建立新的索引標籤, 然後載入下列資料作為內嵌載入。在 Qlik Sense 中建立以下表格以查看結果。

```
Transactions:
Load
*,
Window(Avg(transaction_amount),customer_id) as AvgCustTransaction;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

結果

新增包含彙總之欄位的結果

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	大小	color_code	AvgCustTransaction
3750	20180830	23.56	2	2038593	L	紅色	103.43
3751	20180907	556.31	6	203521	M	橘色	266.775
3752	20180916	5.75	1	5646471	S	藍色	42.935
3753	20180922	125.00	7	3036491	L	黑色	64.21
3754	20180922	484.21	13	049681	XS	紅色	273.88
3756	20180922	59.18	2	2038593	M	藍色	103.43
3757	20180923	177.42	21	203521	XL	黑色	266.775

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	大小	color_code	AvgCustTransaction
3758	20180924	153.42	14	2038593	L	紅色	103.43
3759	20180925	7.42	5	203521	M	橘色	266.775
3760	20180925	80.12	18	5646471	M	藍色	42.935
3761	20180926	3.42	7	3036491	XS	黑色	64.21
3763	20180926	63.55	12	049681	S	紅色	273.88
3763	20180927	177.56	10	2038593	L	藍色	103.43
3764	20180927	325.95	8	203521	XL	黑色	266.775

範例 - 新增包含為特定值篩選之彙總的欄位

範例:新增包含為特定值篩選之彙總的欄位

載入指令碼

在資料載入編輯器中建立新的索引標籤，然後載入下列資料作為內嵌載入。在 Qlik Sense 中建立以下表格以查看結果。

```
Transactions:
Load
*,
Window(Avg(transaction_amount),customer_id, color_code = 'Blue') as AvgCustTransaction;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

結果

新增包含為特定值篩選之彙總的欄位的結果

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	大小	color_code	AvgCustTransaction
3750	20180830	23.56	2	2038593	L	紅色	-
3751	20180907	556.31	6	203521	M	橘色	-
3752	20180916	5.75	1	5646471	S	藍色	42.94
3753	20180922	125.00	7	3036491	L	黑色	-
3754	20180922	484.21	13	049681	XS	紅色	-
3756	20180922	59.18	2	2038593	M	藍色	118.4
3757	20180923	177.42	21	203521	XL	黑色	-
3758	20180924	153.42	14	2038593	L	紅色	-
3759	20180925	7.42	5	203521	M	橘色	-
3760	20180925	80.12	18	5646471	M	藍色	42.94
3761	20180926	3.42	7	3036491	XS	黑色	-
3763	20180926	63.55	12	049681	S	紅色	-
3763	20180927	177.56	10	2038593	L	藍色	118.4
3764	20180927	325.95	8	203521	XL	黑色	-

範例 - 新增具有滑動視窗的欄位

範例:新增具有滑動視窗的欄位

載入指令碼

在資料載入編輯器中建立新的索引標籤，然後載入下列資料作為內嵌載入。在 Qlik Sense 中建立以下表格以查看結果。

```
Transactions:
Load
*,
Window(Avg(transaction_amount),customer_id, 'ASC', -1, 1, 0, 1) as AvgCustTransaction;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
```

```
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

結果

新增包含為特定值篩選之彙總的欄位的結果

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	大小	color_code	AvgCustTransaction
3750	20180830	23.56	2	2038593	L	紅色	41.37
3751	20180907	556.31	6	203521	M	橘色	366.865
3752	20180916	5.75	1	5646471	S	藍色	42.935
3753	20180922	125.00	7	3036491	L	黑色	64.21
3754	20180922	484.21	13	049681	XS	紅色	273.88
3756	20180922	59.18	2	2038593	M	藍色	106.3
3757	20180923	177.42	21	203521	XL	黑色	92.42
3758	20180924	153.42	14	2038593	L	紅色	165.49
3759	20180925	7.42	5	203521	M	橘色	166.685
3760	20180925	80.12	18	5646471	M	藍色	80.12
3761	20180926	3.42	7	3036491	XS	黑色	3.42
3763	20180926	63.55	12	049681	S	紅色	177.56
3763	20180927	177.56	10	2038593	L	藍色	63.55
3764	20180927	325.95	8	203521	XL	黑色	325.95

限制

Window 具有下列限制：

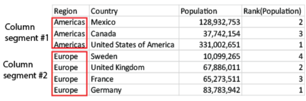
- **Window** 是資源密集函數，尤其是在記憶體取用方面。
- Qlik Sense Mobile 中不支援 **Window**。
- 圖表運算式不支援 **Window**。
- 您無法在 **Window** 函數內部內嵌 **Window** 函數。
- **Window** 無法在彙總函數內部使用。

- **Window** 需要能夠掃描整個表格。
- **WRank()**、**RecNo()** 和 **RowNo()** 無法在使用滑動視窗功能時用於 **Window**。

WRank - 指令碼函數

WRank() 評估載入指令碼中表格的列，並且對每一列，顯示在載入指令碼中評估之欄位的值的相對位置。評估表格時，該函數會比較該結果與包含目前分割的其他列所產生的結果，並傳回區段內目前列的排名。

表格中的分割



	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

WRank 只能用於 **Window** 函數。**Window** 函數必須包括排序類型和排序運算式。排名套用於排序運算式。

語法：

```
WRank ([mode[, fmt]])
```

傳回的資料類型： 雙值

引數：

引數

引數	描述
mode	也可以指定函數結果的數字表示法。
fmt	也可以指定函數結果的文字表示法。
TOTAL	如果表格為單一維度或指令碼前面加上 **TOTAL** 限定詞，則沿著整個資料行計算該函數。如果表格或同等表格有多個垂直維度，則目前分割僅會包含在所有維度資料行中與目前列有相同值的列 (顯示欄位間排序順序之最後一個維度的資料行除外)。

傳回的排名都是雙值，在這種情況下，各個列都會有唯一的排名，這是介於 1 與目前分割中的列數兩者之間的整數。

多個列有相同的排名時，可以使用 **mode** 和 **fmt** 參數控制文字和數字表示法。

mode
第一個引數 **mode** 可使用下列值：

mode 值

值	描述
0 (預設)	如果共用群組中所有的排名都落在整個排名中間值下半部，所有列將得到共用群組的最低排名。 如果共用群組中所有的排名都落在整個排名中間值上半部，所有列將得到共用群組的最高排名。 如果共用群組中的排名落在整個排名的中間值，所有列的值將相當於整個分割中最高與最低排名兩者的平均。
1	所有列的最低排名。
2	所有列的平均排名。
3	所有列的最高排名。
4	第一列的最低排名，然後每列遞增 1。

fmt

第二個引數 **fmt** 可控制下列值：

fmt 值

值	描述
0 (預設)	低值 **-** 所有列的高值 (例如 3 - 4)。
1	所有列的低值。
2	第一列的低值，在後續列上則空白。

mode 4 與 **fmt** 2 的列順序取決於表格欄位的載入順序。

範例 - 新增排名欄位

範例：新增排名欄位

載入指令碼

在資料載入編輯器中建立新的索引標籤，然後載入下列資料作為內嵌載入。在 Qlik Sense 中建立以下表格以查看結果。

```
Transactions:
Load
*,
Window(WRank(0),customer_id, 'Desc', transaction_amount) as TransactionRanking;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
```

```
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

結果

新增排名欄位的結果

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	大小	color_code	TransactionRanking
3750	20180830	23.56	2	2038593	L	紅色	4-4
3751	20180907	556.31	6	203521	M	橘色	1-1
3752	20180916	5.75	1	5646471	S	藍色	2-2
3754	20180922	484.21	13	049681	XS	紅色	1-1
3756	20180922	59.18	2	2038593	M	藍色	3-3
3753	20180922	125.00	7	3036491	L	黑色	1-1
3757	20180923	177.42	21	203521	XL	黑色	3-3
3758	20180924	153.42	14	2038593	L	紅色	2-2
3759	20180925	7.42	5	203521	M	橘色	4-4
3760	20180925	80.12	18	5646471	M	藍色	1-1
3763	20180926	63.55	12	049681	S	紅色	2-2
3761	20180926	3.42	7	3036491	XS	黑色	2-2
3764	20180927	325.95	8	203521	XL	黑色	2-2
3763	20180927	177.56	10	2038593	L	藍色	1-1

範例 - 對單一數字結果使用 fmt 新增排名欄位

範例 : 對單一數字結果使用 fmt 新增排名欄位

載入指令碼

在資料載入編輯器中建立新的索引標籤 , 然後載入下列資料作為內嵌載入。在 Qlik Sense 中建立以下表格以查看結果。

```
Transactions:
Load
*,Window(WRank(0,1),customer_id, 'Desc', transaction_amount) as TransactionRanking;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

結果

對單一數字結果使用 fmt 新增排名欄位的結果

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	大小	color_code	TransactionRanking
3750	20180830	23.56	2	2038593	L	紅色	4
3751	20180907	556.31	6	203521	M	橘色	1
3752	20180916	5.75	1	5646471	S	藍色	2
3754	20180922	484.21	13	049681	XS	紅色	1
3756	20180922	59.18	2	2038593	M	藍色	3
3753	20180922	125.00	7	3036491	L	黑色	1
3757	20180923	177.42	21	203521	XL	黑色	3
3758	20180924	153.42	14	2038593	L	紅色	2
3759	20180925	7.42	5	203521	M	橘色	4
3760	20180925	80.12	18	5646471	M	藍色	1
3763	20180926	63.55	12	049681	S	紅色	2
3761	20180926	3.42	7	3036491	XS	黑色	2
3764	20180927	325.95	8	203521	XL	黑色	2
3763	20180927	177.56	10	2038593	L	藍色	1

範例 - 透過多個分割區新增排名欄位

範例: 透過多個分割區新增排名欄位

載入指令碼

在資料載入編輯器中建立新的索引標籤, 然後載入下列資料作為內嵌載入。在 Qlik Sense 中建立以下表格以查看結果。

```
Transactions:
Load
*,Window(WRank(0,1),customer_id, size, color_code, 'Desc', transaction_amount) as
TransactionRanking;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

結果

對單一數字結果使用 fmt 新增排名欄位的結果

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	大小	color_code	TransactionRanking
3750	20180830	23.56	2	2038593	L	紅色	2
3751	20180907	556.31	6	203521	M	橘色	1
3752	20180916	5.75	1	5646471	S	藍色	1
3754	20180922	484.21	13	049681	XS	紅色	1
3756	20180922	59.18	2	2038593	M	藍色	1
3753	20180922	125.00	7	3036491	L	黑色	1
3757	20180923	177.42	21	203521	XL	黑色	2

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	大小	color_code	TransactionRanking
3758	20180924	153.42	14	2038593	L	紅色	1
3759	20180925	7.42	5	203521	M	橘色	2
3760	20180925	80.12	18	5646471	M	藍色	1
3763	20180926	63.55	12	049681	S	紅色	1
3761	20180926	3.42	7	3036491	XS	黑色	1
3764	20180927	325.95	8	203521	XL	黑色	1
3763	20180927	177.56	10	2038593	L	藍色	1

限制

WRank 具有下列限制：

- 若 fmt 值是 `0`，且您想要對 **WRank** 使用雙重結果的文字部分，則您必須使用 **Text()** 搭配 **Window(WRank)**。例如：`Text(Window(WRank(0), Unit, 'DESC', Age)) as UnitWRankedByAgeText`。

9 檔案系統存取限制

出於安全原因，標準模式下的 Qlik Sense 在資料載入指令碼或公開檔案系統的函數與變數中不支援路徑。

然而，由於 QlikView 中支援檔案系統路徑，所以可以停用標準模式並使用舊模式，以便重複使用 QlikView 載入指令碼。

停用標準模式可以透過公開檔案系統來建立安全性風險。

停用標準模式 (page 1425)

9.1 連接至基於檔案的 ODBC 與 OLE DB 資料連線時的安全方面

使用檔案型驅動程式的 ODBC 與 OLE DB 資料連線，將公開連線字串中已連接資料檔案的路徑。在資料選擇對話方塊或者在某些 SQL 查詢中編輯連線時，可以公開路徑。這是標準模式與舊模式中的案例。

如果您對公開資料檔案的路徑有疑慮，則建議在可能的情況下，使用資料夾資料連線連接至資料檔案。

9.2 標準模式下的限制

數個陳述式、變數及函數不能在標準模式下使用，或者有使用限制。在資料載入指令碼中使用不受支援的陳述式會在載入指令碼執行時產生錯誤。可以在指令碼記錄檔中找到錯誤訊息。使用不受支援的變數及函數不會產生錯誤訊息或記錄檔項目。該函數會改為傳回 NULL 值。

在您編輯資料載入指令碼時，不會指明變數、陳述式或函數不受支援。

系統變數

系統變數

變數	標準模式	舊模式	定義
Floppy	不支援	受支援	傳回所找到第一個軟碟機的磁碟機代號，通常是 *a:*。
CD	不支援	受支援	傳回所找到第一個光碟機的磁碟機代號。如果找不到 CD-ROM，則會傳回 *c:*。

變數	標準模式	舊模式	定義
QvPath	不支援	受支援	會傳回 Qlik Sense 執行檔的瀏覽字串。
QvRoot	不支援	受支援	會傳回 Qlik Sense 執行檔的根目錄。
QvWorkPath	不支援	受支援	將瀏覽字串傳回目前 Qlik Sense 應用程式。
QvWorkRoot	不支援	受支援	傳回目前 Qlik Sense 應用程式的根目錄。
WinPath	不支援	受支援	將瀏覽字串傳回 Windows。
WinRoot	不支援	受支援	傳回 Windows 的根目錄。
$(include=...)	受支援的輸入:使用物件庫連線的路徑	受支援的輸入:使用物件庫連線或檔案系統的路徑	**Include/Must_Include** 變數會指定包含應該納入指令碼並且評估為指令碼之文字的檔案。這不是用來新增資料。您可以將指令碼的各部分儲存在獨立文字檔中,並在數個應用程式中重複使用。這是使用者定義的變數。

一般指令碼陳述式

一般指令碼陳述式

陳述式	標準模式	舊模式	定義
Binary	受支援的輸入:使用物件庫連線的路徑	受支援的輸入:使用物件庫連線或檔案系統的路徑	**binary** 陳述式用來載入另一個應用程式的資料。
Connect	受支援的輸入:使用物件庫連線的路徑	受支援的輸入:使用物件庫連線或檔案系統的路徑	**CONNECT** 陳述式用來定義透過 OLE DB/ODBC 介面對一般資料庫的 Qlik Sense 存取。若是 ODBC,首先必須使用 ODBC 管理員指定資料來源。

陳述式	標準模式	舊模式	定義
Directory	受支援的輸入:使用物件庫連線的路徑	受支援的輸入:使用物件庫連線或檔案系統的路徑	**Directory** 陳述式會在後續 **LOAD** 陳述式中定義要在其中尋找資料檔的目錄,直到發出新的 **Directory** 陳述式為止。
Execute	不支援	受支援的輸入:使用物件庫連線或檔案系統的路徑	**Execute** 陳述式用來在 Qlik Sense 載入資料時,執行其他程式。例如,進行必要的轉換。
LOAD from ...	受支援的輸入:使用物件庫連線的路徑	受支援的輸入:使用物件庫連線或檔案系統的路徑	**LOAD** 陳述式可以從檔案、指令碼中定義的資料、先前載入的表格、網頁、後續 **SELECT** 陳述式的結果或自動產生的資料來載入欄位。
Store into ...	受支援的輸入:使用物件庫連線的路徑	受支援的輸入:使用物件庫連線或檔案系統的路徑	**Store** 陳述式建立 QVD、Parquet、CSV 或 TXT 檔案。

指令碼控制陳述式

指令碼控制陳述式

陳述式	標準模式	舊模式	定義
For each... filelist mask/dirlist mask	受支援的輸入:使用物件庫連線的路徑 傳回的輸出:物件庫連線	受支援的輸入:使用物件庫連線或檔案系統的路徑 傳回的輸出:物件庫連線或檔案系統路徑,視輸入而定	filelist mask 語法會對於符合 **filelist mask** 的目前目錄中的所有檔案產生逗號分隔清單。**dirlist mask** 語法會對於符合資料夾目錄名稱遮罩的目前資料夾中的所有資料夾產生逗號分隔清單。

檔案函數

檔案函數

函數	標準模式	舊模式	定義
Attribute()	受支援的輸入:使用物件庫連線的路徑	受支援的輸入:使用物件庫連線或檔案系統的路徑	以文字形式傳回各種媒體檔案的中繼標記值。
ConnectString()	傳回的輸出:物件庫連線名稱	物件庫連線名稱或實際連線,取決於輸入	傳回 ODBC 或 OLE DB 連線的作用中連線字串。
FileDir()	傳回的輸出:物件庫連線	傳回的輸出:物件庫連線或檔案系統路徑,視輸入而定	**FileDir** 函數會傳回一個字串,其中包含目前正在讀取之表格檔案的目錄路徑。
FilePath()	傳回的輸出:物件庫連線	傳回的輸出:物件庫連線或檔案系統路徑,視輸入而定	**FilePath** 函數會傳回一個字串,其中包含目前正在讀取之表格檔案的完整路徑。
FileSize()	受支援的輸入:使用物件庫連線的路徑	受支援的輸入:使用物件庫連線或檔案系統的路徑	**FileSize** 函數會傳回一個整數,其中包含檔案 filename 或 (若未指定 filename) 目前正在讀取之表格檔案的大小 (以位元組為單位)。
FileTime()	受支援的輸入:使用物件庫連線的路徑	受支援的輸入:使用物件庫連線或檔案系統的路徑	**FileTime** 函數會以指定檔案上次修改的 UTC 格式傳回時間戳記。若未指定檔案,函數會以目前讀取表格檔案上次修改的 UTC 傳回時間戳記。
GetFolderPath()	不支援	傳回的輸出:絕對路徑	**GetFolderPath** 函數會傳回 Microsoft Windows *SHGetFolderPath* 函數的值。此函數會輸入 Microsoft Windows 資料夾的名稱,並傳回資料夾的完整路徑。

函數	標準模式	舊模式	定義
QvdCreateTime()	受支援的輸入:使用物件庫連線的路徑	受支援的輸入:使用物件庫連線或檔案系統的路徑	此指令碼函數會從 QVD 檔案傳回 XML 標頭時間戳記 (如果呈現的話),否則會傳回 NULL。在時間戳記中,所提供時間為協調世界時。
QvdFieldName()	受支援的輸入:使用物件庫連線的路徑	受支援的輸入:使用物件庫連線或檔案系統的路徑	此指令碼函數會傳回 QVD 檔案中的欄位編號 **fieldno** 名稱。如果該欄位不存在,則會傳回 NULL。
QvdNoOfFields()	受支援的輸入:使用物件庫連線的路徑	受支援的輸入:使用物件庫連線或檔案系統的路徑	此指令碼函數會傳回 QVD 檔案中的欄位數。
QvdNoOfRecords()	受支援的輸入:使用物件庫連線的路徑	受支援的輸入:使用物件庫連線或檔案系統的路徑	此指令碼函數會傳回 QVD 檔案中的目前記錄數。
QvdTableName()	受支援的輸入:使用物件庫連線的路徑	受支援的輸入:使用物件庫連線或檔案系統的路徑	此指令碼函數會傳回 QVD 檔案中儲存的表格名稱。

系統函數

系統函數

函數	標準模式	舊模式	定義
DocumentPath()	不支援	傳回的輸出:絕對路徑	此函數會傳回一個字串,其中包含目前 Qlik Sense 應用程式的完整路徑。
GetRegistryString()	不支援	受支援	傳回含指定登錄路徑的具名登錄機碼值。此函數可用於類似圖表與指令碼中。

9.3 停用標準模式

您可停用標準模式,或換言之,即設定舊模式,以重複使用參考絕對或相對檔案路徑以及物件庫連線的 QlikView 載入指令碼。

停用標準模式可以透過公開檔案系統來建立安全性風險。

Qlik Sense

對於 Qlik Sense 而言，標準模式可使用**標準模式**屬性在 QMC 中停用。

Qlik Sense Desktop

在 Qlik Sense Desktop 中，您可在 *Settings.ini* 設定標準/舊模式。

若您 Qlik Sense Desktop 使用預設安裝位置進行安裝，則 *Settings.ini* 會位於 *C:\Users\{user}\Documents\Qlik\Sense\Settings.ini*。若您安裝 Qlik Sense Desktop 到所選資料夾，則 *Settings.ini* 會位於安裝路徑的 *Engine* 資料夾。

請執行下列動作：

1. 在文字編輯器中開啟 *Settings.ini*。
2. 變更 *StandardReload=1* 為 *StandardReload=0*。
3. 儲存檔案並開始 Qlik Sense Desktop。

Qlik Sense Desktop 目前以舊版模式執行。

設定

標準重新載入的可用設定為：

- 1 (標準模式)
- 0 (舊模式)

10 圖表層級指令碼

修改圖表資料時，您可使用 Qlik Sense 指令碼的子集，這由許多陳述式組成。陳述式可以是一般的指令碼陳述式或是指令碼控制陳述式。某些陳述式的前面可加上前置詞。

一般陳述式通常用於以某種方式操縱資料。這些陳述式可在指令碼中撰寫為任意行數，但是一定必須以分號 (";") 終止。

控制陳述式通常用於控制指令碼執行的流程。控制陳述式的每個子句都必須保持在一個指令碼行內，並可以分號或行結尾終止。

前置詞可套用於適用的一般陳述式，但絕不可套用於控制陳述式。

所有的指令碼關鍵字皆可以小寫和大寫字元的任意組合輸入。但用於陳述式中的欄位和變數名稱則會區分大小寫。

在此區段中，您可以找到按字母順序排列的所有指令碼陳述式、控制陳述式和前置詞清單，這可用於修改圖表資料時使用的指令碼子集。

10.1 控制陳述式

修改圖表資料時，您可使用 Qlik Sense 指令碼的子集，這由許多陳述式組成。陳述式可以是一般的指令碼陳述式或是指令碼控制陳述式。

控制陳述式通常用於控制指令碼執行的流程。控制陳述式的每個子句都必須保持在一個指令碼行內，並可以分號或行結尾終止。

前置詞從未套用至控制陳述式。

所有的指令碼關鍵字皆可以小寫和大寫字元的任意組合輸入。

圖表修飾詞控制陳述式概述

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

Call

call 控制陳述式會呼叫必須由先前的 **sub** 陳述式定義的副程式。

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

Do..loop

do..loop 控制陳述式是指令碼反覆運算建構，這會執行一或數個陳述式，直到符合邏輯條件為止。

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

End

End 指令碼關鍵字用來結束 **If**、**Sub** 和 **Switch** 子句。

Exit

Exit 指令碼關鍵字屬於 **Exit Script** 陳述式，但是也可以用來結束 **Do**、**For** 或 **Sub** 子句。

Exit script

此控制陳述式可停止指令碼執行。它可以插入指令碼的任一處。

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

For..next

for..next 控制陳述式是包含計數器的指令碼反覆運算建構。將對於所指定上下限之間的各個 counter 變數值，執行 **for** 與 **next** 所括住迴圈之內的陳述式。

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

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

For each ..next

for each..next 控制陳述式是指令碼反覆運算建構，這會對於逗號分隔清單中的各個值執行一或數個陳述式。將會為清單的各個值，執行由 **for** 與 **next** 括住之迴圈內的陳述式。

```
For each..next var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

If..then

if..then 控制陳述式是一種陳述式選項建構，會根據一或數個邏輯條件強制指令碼執行遵照不同的路徑。

> *由於 **if..then** 陳述式是控制陳述式，因而會以分號或行尾來結束，所以這四個可能的子句 (**if..then**、**elseif..then**、**else** 和 **end if**) 都不能超過行邊界。*

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

  [ statements ]

{ elseif condition then
```

```
  [ statements ] }
[ else
  [ statements ] ]
end if
```

Next

Next 指令碼用來結束 **For** 迴圈。

Sub

sub..end sub 控制陳述式會定義可從 **call** 陳述式呼叫的副程式。

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

Switch

switch 控制陳述式是一個指令碼選項建構，會根據運算式的值強制指令碼執行遵照不同的路徑。

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

To

To 指令碼關鍵字用於數個指令碼陳述式中。

Call

call 控制陳述式會呼叫必須由先前的 **sub** 陳述式定義的副程式。

語法：

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

引數：

引數

引數	描述
name	副程式的名稱。
paramlist	要傳送到副程式的實際參數的逗號分隔清單。清單中的各個項目可能是欄位名稱、變數或任意的運算式。

call 陳述式呼叫的副程式必須由指令碼執行期間較早出現的 **sub** 加以定義。

參數會複製到副程式中，而且如果 **call** 陳述式中的參數是變數而不是運算式，將在結束副程式時再次複製回參數。

限制：

- 由於 **call** 陳述式是控制陳述式，而且結尾是分號或行結尾，因此不可超出行邊界。
- 透過控制陳述式內部的 `Sub..end sub` 定義副程式時，例如 `if..then`，您只能從相同的控制陳述式內部叫用副程式。

Do..loop

do..loop 控制陳述式是指令碼反覆運算建構，這會執行一或數個陳述式，直到符合邏輯條件為止。

語法：

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

*由於 **do..loop** 陳述式是控制陳述式，因而會以分號或行結尾來結束，所以這三個可能的子句 (**do**、**exit do**、與 **loop**) 都不能超過行邊界。*

引數：

引數

引數	描述
condition	評估為 True 或 False 的邏輯運算式。
statements	是一或多個 Qlik Sense 指令碼陳述式的任何群組。
while / until	**while** 或 **until** 條件子句必須只在任何 **do..loop** 陳述式 (也就是在 **do** 之後或 **loop** 之後) 中出現一次。只有在各個條件第一次出現時，才會予以解譯，不過每次在迴圈中出現時，都會予以評估。
exit do	如果 **exit do** 子句出現在迴圈中，指令碼的執行將傳輸到 **loop** 子句後的第一個陳述式，代表迴圈結束。選用 **when** 或 **unless** 後置詞，即可將 **exit do** 子句設定為條件子句。

End

End 指令碼關鍵字用來結束 **If**、**Sub** 和 **Switch** 子句。

Exit

Exit 指令碼關鍵字屬於 **Exit Script** 陳述式，但是也可以用來結束 **Do**、**For** 或 **Sub** 子句。

Exit script

此控制陳述式可停止指令碼執行。它可以插入指令碼的任一處。

語法:

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

由於 **exit script** 陳述式是控制陳述式，而且結尾是分號或行結尾，因此不可超出行邊界。

引數:

引數

引數	描述
condition	評估為 True 或 False 的邏輯運算式。
when / unless	選用 **when** 或 **unless** 子句可將 **exit script** 陳述式設定為條件式。

範例:

```
//Exit script
Exit Script;

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

For..next

for..next 控制陳述式是包含計數器的指令碼反覆運算建構。將對於所指定上下限之間的各個 counter 變數值，執行 **for** 與 **next** 所括住迴圈之內的陳述式。

語法:

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

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

只有在第一次進入迴圈時，才會評估 *expr1*、*expr2* 和 *expr3* 運算式。迴圈內的陳述式可變更 counter 變數的值，不過在程式設計上這不見得是好方法。

如果 **exit for** 子句出現在迴圈中，指令碼的執行將傳輸到 **next** 子句後的第一個陳述式，代表迴圈結束。選用 **when** 或 **unless** 後置詞，即可將 **exit for** 子句設定為條件子句。

> *由於 **for..next** 陳述式是控制陳述式，因而會以分號或行結尾來結束，所以這三個可能的子句 (**for..to..step**、**exit for**、與 **next**) 都不能超過行邊界。*

引數:

引數

引數	描述
counter	變數名稱。如果在 **next** 之後指定 *counter*, 它的變數名稱必須與對應的 **for** 之後出現的變數名稱相同。
expr1	決定應對其執行迴圈的第一個 *counter* 變數值所用的運算式。
expr2	決定應對其執行迴圈的最後一個 *counter* 變數值所用的運算式。
expr3	決定每次執行迴圈時, 表示 *counter* 變數遞增的值所用的運算式。
condition	評估為 True 或 False 的邏輯運算式。
statements	是一或多個 Qlik Sense 指令碼陳述式的任何群組。

For each..next

for each..next 控制陳述式是指令碼反覆運算建構, 這會對於逗號分隔清單中的各個值執行一或數個陳述式。將會為清單的各個值, 執行由 **for** 與 **next** 括住之迴圈內的陳述式。

語法:

特殊語法能夠以目前目錄中的檔案和目錄名稱產生清單。

```
for each var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

引數:

引數

引數	描述
var	為各個迴圈執行從清單取得新值的指令碼變數名稱。如果在 **next** 之後指定 **var**, 它的變數名稱必須與對應的 **for each** 之後出現的變數名稱相同。

迴圈內的陳述式可變更 **var** 變數的值, 不過在程式設計上這不見得是好方法。

如果 **exit for** 子句出現在迴圈中, 指令碼的執行將傳輸到 **next** 子句後的第一個陳述式, 代表迴圈結束。選用 **when** 或 **unless** 後置詞, 即可將 **exit for** 子句設定為條件子句。

*由於 **for each..next** 陳述式是控制陳述式，因而會以分號或行結尾來結束，所以這三個可能的子句 (**for each**、**exit for**、與 **next**) 都不能超過行邊界。*

語法：

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

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

引數

引數	描述
constant	任何數字或字串。請注意，直接在指令碼中寫入的字串必須以單引號括住。沒有單引號的字串將被解譯為變數，因而將使用變數的值。數字不需要以單引號括住。
expression	任意運算式。
mask	檔案名稱或目錄名稱遮罩，其中可能包含任何有效的檔案名稱字元，以及標準的萬用字元 ***** 和 **?**。 您可以使用絕對檔案路徑或 lib:// 路徑。
condition	評估為 True 或 False 的邏輯運算式。
statements	是一或多個 Qlik Sense 指令碼陳述式的任何群組。
filelist mask	此語法會對於符合檔案名稱遮罩的目前目錄中所有的檔案產生逗號分隔清單。 *此引數僅在標準模式下支援物件庫連線。*
dirlist mask	此語法會對於符合資料夾名稱遮罩的目前資料夾中所有的資料夾產生逗號分隔清單。 *此引數僅在標準模式下支援物件庫連線。*
fieldvaluelist mask	此語法透過已載入至 Qlik Sense 的欄位值反覆運算。

*Qlik 網頁儲存空間提供者連接器 和其他 DataFiles 連線不支援使用萬用字元 (***** 和 **?**) 的篩選遮罩。*

Example 1: **載入檔案清單**

```
// 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: **在磁碟上建立檔案清單**

此範例會載入資料夾中所有 Qlik Sense 相關檔案的清單。

```
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: **透過欄位值反覆運算**

此範例會透過 FIELD 的已載入值清單反覆運算，並產生新的欄位 NEWFIELD。對於 FIELD 的每一個值，將建立兩筆 NEWFIELD 記錄。

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

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

產生的表格如下所示：

Example table

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

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

if..then 控制陳述式是一種陳述式選項建構，會根據一或數個邏輯條件強制指令碼執行遵照不同的路徑。

控制陳述式通常用於控制指令碼執行的流程。在圖表運算式中，請改用 **if** 條件式函數。

語法：

```
If  condition then

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

由於 **if..then** 陳述式是控制陳述式，因而會以分號或行尾來結束，所以這四個可能的子句 (**if..then**、**elseif..then**、**else** 和 **end if**) 都不能超過行邊界。

引數：

引數

引數	描述
condition	可評估為 True 或 False 的邏輯運算式。
statements	是一或多個 Qlik Sense 指令碼陳述式的任何群組。

Example 1:

```
if a=1 then
```

```
        LOAD * from abc.csv;

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

Example 2:

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

Example 3:

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

Next

Next 指令碼用來結束 **For** 迴圈。

Sub..end sub

sub..end sub 控制陳述式會定義可從 **call** 陳述式呼叫的副程式。

語法：

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

引數會複製到副程式中，而且如果 **call** 陳述式中的對應實際參數是變數名稱，將在結束副程式時再次複製回引數。

如果副程式擁有的型式參數比 **call** 陳述式傳遞的實際參數多，則會將額外的參數初始化為 NULL，且可在副程式內做為本機變數。

引數：

引數

引數	描述
name	副程式的名稱。
paramlist	以逗號分隔副程式型式參數之變數名稱的清單。其可在副程式內做為任何變數。
statements	是一或多個 Qlik Sense 指令碼陳述式的任何群組。

限制:

- 由於 **sub** 陳述式是控制陳述式, 因而會以分號或行尾來結束, 所以這兩個子句 (**sub** 與 **end sub**) 都不能超過行邊界。
- 透過控制陳述式內部的 `Sub..end sub` 定義副程式時, 例如 `if..then`, 您只能從相同的控制陳述式內部叫用副程式。

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: - 參數傳送

```
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)
```

以上範例的結果會是在本機的副程式內, A 將初始化為 1、B 將初始化為 4, 而 C 將初始化為 NULL。

結束副程式後, 全域變數 A 會取得 2 做為值 (從副程式複製回來)。第二個實際參數 “(X+1)*2” 因為不是變數, 所以不會被複製回來。最後, 全域變數 C 不會受到副程式呼叫的影響。

Switch..case..default..end switch

switch 控制陳述式是一個指令碼選項建構, 會根據運算式的值強制指令碼執行遵照不同的路徑。

語法：

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

*由於 **switch** 陳述式是控制陳述式，因而會以分號或行尾來結束，所以這四個可能的子句 (**switch**、**case**、**default** 和 **end switch**) 都不能超過行邊界。*

引數：

引數

引數	描述
expression	任意運算式。
valuelist	是以逗號分隔的值清單，其會與運算式的值進行比較。指令碼將繼續執行，且遇到 valuelist 值的第一個群組中的陳述式等於運算式中的值。valuelist 中的每個值可以是任意運算式。如果在所有 **case** 子句中都找不到任何相符項目，將會執行 **default** 子句下的陳述式 (若已指定)。
statements	是一或多個 Qlik Sense 指令碼陳述式的任何群組。

範例：

```
Switch I

Case 1

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

Case 2

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

Default

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

End Switch
```

To

To 指令碼關鍵字用於數個指令碼陳述式中。

10.2 前置詞

前置詞可套用於適用的一般陳述式，但絕不可套用於控制陳述式。

所有的指令碼關鍵字皆可以小寫和大寫字元的任意組合輸入。但用於陳述式中的欄位和變數名稱則會區分大小寫。

圖表修飾詞前置詞概述

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

Add

Add 前置詞可新增至指令碼中的任何 **LOAD** 或 **SELECT** 陳述式，以指定這應將記錄新增至另一個表格。這也能指定應在部分載入中執行此陳述式。**Add** 前置詞也能用於 **Map** 陳述式。

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

Replace

Replace 前置詞 可新增至指令碼中的任何 **LOAD** 或 **SELECT** 陳述式，以指定載入的表格應取代另一個表格。這也能指定應在部分載入中執行此陳述式。**Replace** 前置詞也能用於 **Map** 陳述式。

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

Add

在圖表修改內容中，**Add** 前置詞搭配 **LOAD** 使用，以將值附加至 *HC1* 表格，代表由 Qlik 關聯引擎 運算的超立方體。您可以指定一個或數個欄。遺失的值會由 Qlik 關聯引擎 自動填入。

語法：

```
Add loadstatement
```

範例：

此範例將兩列從內嵌陳述式新增至欄*日期*和*銷售*

```
Add Load
x as Dates,
y as Sales
Inline
[
Dates,Sales
2001/09/1,1000
2001/09/10,-300
]
```

Replace

在圖表修改內容中，**Replace** 前置詞可透過由指令碼定義的運算值變更 *HC1* 表格的所有值。

語法：

```
Replace loadstatement
```

範例：

此範例可透過 `x` 和 `y` 的總和，覆寫欄 `z` 中的所有值。

```
Replace Load
x+y as z
Resident HC1;
```

10.3 一般陳述式

一般陳述式通常用於以某種方式操縱資料。這些陳述式可在指令碼中撰寫為任意行數，但是一定必須以分號 (";") 終止。

所有的指令碼關鍵字皆可以小寫和大寫字元的任意組合輸入。但用於陳述式中的欄位和變數名稱則會區分大小寫。

圖表修飾詞一般陳述式概述

概述之後，會進一步描述每個函數。您還可以在語法中按一下函數名稱，以立即存取該特定函數的詳細資料。

LOAD

在圖表修改內容中，**LOAD** 陳述式可將其他資料從指令碼中定義的資料或從先前載入的表格載入到超立方體。這也可以從分析連線載入資料。

> ***LOAD*** *陳述式必須有* ***Replace*** *或* ***Add*** *前置詞，否則將會被拒絕。*

```
Add | Replace Load [ distinct ] fieldlist
(
inline data [ format-spec ] |
resident table-label
) | extension pluginname.functionname([script] tabledescription)]
[ where criterion | while criterion ]
[ group by groupbyfieldlist ]
[order by orderbyfieldlist ]
```

Let

let 陳述式是 **set** 陳述式的補集，用來定義指令碼變數。相對於 **set** 陳述式，**let** 陳述式會先在指令碼執行時間評估 '=' 右邊的運算式，然後才將運算式指派給變數。

```
Let variablename=expression
```

Set

set 陳述式用來定義指令碼變數。這些可用來取代字串、路徑、磁碟機等。

```
Set variablename=string
```

Put

Put 陳述式用來設定超立方體中的某些數值。

HCValue

HCValue 陳述式可用於在指定的欄中擷取列中的值。

Load

在圖表修改內容中，**LOAD** 陳述式可將其他資料從指令碼中定義的資料或從先前載入的表格載入到超立方體。這也可以從分析連線載入資料。

> *__LOAD__ 陳述式必須有 __Replace__ 或 __Add__ 前置詞，否則將會被拒絕。*

語法:

```
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 ]
```

引數：

引數

引數	描述
fieldlist	*fieldlist ::= (* | field{**,** * | field })* 要載入的欄位清單。使用 * 做為欄位清單表示表格中的所有欄位。 *field ::= (fieldref | expression) [***as** *aliasname]* 欄位定義必須一律包含常值、現有欄位的參考或運算式。 *fieldref ::= (fieldname |@fieldnumber |@startpos:endpos [**I** | **U** | **R** | **B** |* **T**]) *fieldname* 是與表格中欄位名稱相同的文字。請注意，欄位名稱如果包含空格則必須以一般雙引號或方括弧括住。有時候不一定會有檔案名稱。這時請用不同的標記法： *@fieldnumber* 代表分隔表格檔案中的欄位編號。其必須是正整數，前面加上 "@"。編號一律會從 1 開始，一直編號到欄位的數目為止。 *@startpos*:*endpos* 代表欄位在固定長度記錄的檔案中開始和結束的位置。這些位置必須是正整數。這兩個編號前面必須加上 "@"，並且以冒號分隔。編號一律會從 1 開始，一直編號到位置的數目為止。在最後一個欄位中，**n** 用作結束位置。 • 如果 *@startpos*:*endpos* 後面緊接著 **I** 或 **U** 字元，會將讀取的位元組解譯為二元帶正負號 (**I**) 或不帶正負號 (**U**) 的整數 (Intel 位元組順序)。讀取的位置編號必須是 1、2 或 4。 • 如果 *@startpos*:*endpos* 後面緊接著 **R** 字元，會將讀取的位元組解譯為二進位實數 (IEEE 32 位元或 64 位元浮點)。讀取的位置編號必須是 4 或 8。 • 如果 *@startpos*:*endpos* 後面緊接著 **B** 字元，會按照 COMP-3 標準將讀取的位元組解譯為 BCD (Binary Coded Decimal) 編號。可指定任何數目的位元組。 *expression* 可以是以同一個表格中的其他一或數個欄位為基礎的數值函數或字串函數。如需進一步資訊，請參閱運算式的語法。 **as** 用來指派欄位的新名稱。
inline	如果應該在指令碼中輸入資料，而不是從檔案載入資料，可使用 **inline**。 *data ::= [text]* 透過 **inline** 子句輸入的資料必須以雙引號或方括弧括住。這些之間的文字將以檔案內容的相同方式加以解譯。因此，在文字檔中要插入新行的位置，也應該在 **inline** 子句的文字中插入新行，也就是在輸入指令碼時按下 Enter 鍵。資料行數量以第一行定義。 *format-spec ::= (fspec-item {, fspec-item })* 格式規格包含括弧內數個格式規格項目的清單。如需詳細資訊，請參閱 *格式規格項目 (page 157)*。

引數	描述
resident	如果應該從先前載入的表格載入資料，可使用 **resident**。 *table label* 是建立原始表格的 **LOAD** 陳述式之前的標籤。標籤結尾應該加上冒號。
extension	您可以從分析連線載入資料。您需要使用**延伸**子句以叫用伺服器端延伸 (SSE) 外掛程式中定義的函數，或評估指令碼。 您可以將單一表格傳送至 SSE 外掛程式，就會傳回單一資料表格。若外掛程式沒有指定傳回的欄位名稱，則欄位必須命名為 Field1, Field2 等。 `Extension pluginname.functionname( tabledescription );` • 使用 SSE 外掛程式中的函數載入資料 *tabledescription ::= (table { ,tablefield})* 若您沒有說明表格欄位，則必須以載入順序使用欄位。 • 評估 SSE 外掛程式中的指令碼以載入資料 *tabledescription ::= (script, table { ,tablefield})* **表格欄位定義中的資料類型處理** 分析連線中會自動偵測資料類型。若資料沒有數值，且有至少一個非 NULL 文字字串，則欄位可被視為文字。在任何其他情況下，這被視為數字。 您可以 **String()** 或 **Mixed()** 括起欄位名稱，以強制資料類型。 • **String()** 可強制欄位為文字。若欄位為數字，則會擷取雙值的文字部分，不會執行任何轉換。 • **Mixed()** 可強制欄位為雙值。 **String()** 或 **Mixed()** 無法在**延伸**表格欄位定義之外使用，您無法在表格欄位定義中使用其他 Qlik Sense 函數。
where	**where** 是指明選項中是否應該包含記錄的子句。如果 *criterion* 為 True，將包含選項。 *criterion* 是邏輯運算式。
while	**while** 是指明是否應該重複讀取記錄所用的子句。只要 *criterion* 為 True，就會讀取相同的記錄。若要使用，**while** 子句一般必須包含 **IterNo( )** 函數。 *criterion* 是邏輯運算式。
group by	**group by** 是定義應該對哪些欄位彙總 (分組) 資料所用的子句。彙總欄位應該以某些方式包含在載入的運算式中。只有彙總欄位才能在載入的運算式中的彙總函數之外使用。 *groupbyfieldlist ::= (fieldname { ,fieldname })*

引數	描述
order by	**order by** 是 **load** 陳述式處理常駐表格記錄前將這些記錄排序所用的子句。常駐表格可按照一或多個欄位以遞增或遞減順序排序。排序主要以數值進行，其次以國家排序順序進行。只有在資料來源是常駐表格時，才能使用此子句。 排序欄位可指定常駐表格按照哪些欄位排序。可以按照常駐表格的名稱或編號 (第一個欄位的編號是 1) 指定欄位。 orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } *sortorder* 以 *asc* 表示遞增，以 *desc* 表示遞減。如果未指定 *sortorder*，將會採用 *asc*。 *fieldname*、*path*、*filename* 和 *aliasname* 是指出個別名稱代表什麼意義的文字字串。來源表格的任何欄位均可做為 *fieldname*。不過，透過 as 子句 (*aliasname*) 建立的欄位若不在範圍內，無法在同一個 **load** 陳述式內使用。

Let

let 陳述式是 **set** 陳述式的補集，用來定義指令碼變數。相對於 **set** 陳述式，**let** 陳述式會先在指令碼執行時間評估 '=' 右邊的運算式，然後才將運算式指派給變數。

語法：

```
Let variablename=expression
```

範例與結果：

範例	結果

`Set x=3+4;`
 	
`Let y=3+4;`
 	
`z=$(y)+1;`
	`$(x)` 將評估為 '3+4'
	
	`$(y)` 將評估為 ' 7'
	
	`$(z)` 將評估為 ' 8'
	
	注意 **Set** 和 **Let** 陳述式之間的差異。**Set** 陳述式將字串 '3+4' 指派至變數，而 **Let** 陳述式則評估字串並將 7 指派至變數。

Let T=now( );	`$(T)` 將得到目前時間的值。

Set

set 陳述式用來定義指令碼變數。這些可用來取代字串、路徑、磁碟機等。

語法：

```
Set variablename=string
```

Example 1:

```
Set FileToUse=Data1.csv;
```

Example 2:

```
Set Constant="My string";
```

Example 3:

```
Set BudgetYear=2012;
```

Put

put 陳述式用來設定超立方體中的某些數值。

存取欄可以透過標籤進行。您也可以依宣告順序存取欄和列。請參閱以下範例瞭解更多詳細資訊。

語法:

```
put column(position)=value
```

Example 1:

存取欄可以透過標籤進行。

此範例將會在有 *Sales* 標籤之欄的第一個位置中設定 `1` 的值。

```
Put Sales(1) = 1;
```

Example 2:

您可以為量值使用 `#hc1.measure` 格式依宣告順序存取量值欄。

此範例將會設定最終排序超立方體的第十個位置的值 1000。

```
Put #hc1.measure.2(10) = 1000;
```

Example 3:

您可以為維度使用 `#hc1.dimension` 格式依宣告順序存取維度列。

此範例將常數 Pi 的值置於第三個宣告維度的第五列。

```
Put #hc1.dimension.3(5) = Pi();
```

若沒有這類維度或運算式，則在值或標籤中，會傳回錯誤，指示找不到欄。若欄的索引超出範圍，不會顯示錯誤。

HCValue

HCValue 函數用來在指定的欄中擷取列中的值。

語法:

```
HCValue(column,position)
```

Example 1:

此範例傳回欄中第一個位置具有標籤 'Sales' 的值。

```
HCValue(Sales,1)
```

Example 2:

此範例傳回排序超立方體的第十個位置的值。

```
HCValue(#hc1.measure2,10)
```

Example 3:

此範例傳回第三維度中第五列的值。

```
HCValue(#hc1.dimension.3,5)
```

若沒有這類維度或運算式, 則在值或標籤中, 會傳回錯誤, 指示找不到欄。若欄的索引超出範圍, 則會傳回 NULL。

11 Qlik Sense 中不支援的 QlikView 函數與陳述式

在 QlikView 載入指令碼和圖表運算式可以使用的大部份函數和陳述式在 Qlik Sense 中也同樣受支援，但有一些例外，如此處所述。

11.1 Qlik Sense 不支援的指令碼陳述式

QlikViewQlik Sense 不支援的指令碼陳述式

陳述式	註解
Command	改為使用 **SQL**。
InputField	

11.2 Qlik Sense 中不支援的函數

此清單列出 Qlik Sense 中不支援的 QlikView 指令碼和圖表函數。

- **GetCurrentField**
- **GetExtendedProperty**
- **Input**
- **InputAvg**
- **InputSum**
- **MsgBox**
- **NoOfReports**
- **ReportComment**
- **ReportId**
- **ReportName**
- **ReportNumber**

11.3 Qlik Sense 不支援的前置詞

此清單列出 Qlik Sense 中不支援的 QlikView 前置詞。

- **Bundle**
- **Image_Size**
- **Info**

12 不建議在 Qlik Sense 中使用的函數和陳述式

大多數可以在 QlikView 載入指令碼和圖表運算式中使用的函數和陳述式在 Qlik Sense 中也同樣受支援，但它們當中有一些不建議在 Qlik Sense 中使用。也有函數和陳述式可用於已被取代的先前版本 Qlik Sense。

儘管出於相容性目的，它們仍可以如期運作，但最好按照本節的建議更新程式碼，因為它們在以後的版本中可能會被移除。

12.1 不建議在 Qlik Sense 中使用的指令碼陳述式

此表格包含不建議在 Qlik Sense 中使用的指令碼陳述式。

不建議使用的指令碼陳述式

陳述式	建議
Command	改為使用 **SQL**。
CustomConnect	改為使用 **Custom Connect**。

12.2 不建議在 Qlik Sense 中使用的指令碼陳述式參數

此表格列出不建議在 Qlik Sense 中使用的指令碼陳述式參數。

不建議使用的指令碼陳述式參數

陳述式	參數
Buffer	使用 **Incremental**，而不是： • **Inc** (不建議) • **Incr** (不建議)

陳述式	參數
LOAD	以下參數關鍵字由 QlikView 檔案轉換精靈產生。當重新載入資料時，會保留相關功能，但 Qlik Sense 不針對如何使用這些參數產生陳述式提供指引支援/精靈。 • **Bottom** • **Cellvalue** • **Col** • **Colmatch** • **Colsplit** • **Colxtr** • **Compound** • **Contain** • **Equal** • **Every** • **Expand** • **Filters** • **Intarray** • **Interpret** • **Length** • **Longer** • **Numerical** • **Pos** • **Remove** • **Rotate** • **Row** • **Rowcnd** • **Shorter** • **Start**

12.3 不建議在 Qlik Sense 中使用的函數

此表格列出不建議在 Qlik Sense 中使用的指令碼和圖表函數。

不建議使用的函數

函數	建議
NumAvg **NumCount** **NumMax** **NumMin** **NumSum**	改為使用範圍函數。 *範圍函數 (page 1255)*
Color() **QliktechBlue** **QliktechGray**	改用其他色彩函數。要得到相同的色彩，**QliktechBlue()** 可被取代為 **RGB(8, 18, 90)**，**QliktechGray** 可被取代為 **RGB(158, 148, 137)**。 *色彩函數 (page 515)*
QlikViewVersion	改為使用 **EngineVersion**。 *EngineVersion (page 1394)*
ProductVersion	改為使用 **EngineVersion**。 *EngineVersion (page 1394)*
QVUser	
Year2Date	改為使用 **YearToDate**。
Vrank	改為使用 **Rank**。
WildMatch5	改為使用 **WildMatch**。

ALL 限定詞

在 QlikView 中，**ALL** 限定詞可能出現在運算式之前。這相當於使用 **{1} TOTAL**。在這種情況下，將對文件中的所有欄位值進行計算，忽略圖表維度和當前的選項。總是傳回相同的值，而不論在文件中的邏輯狀態為何。如果使用 **ALL** 限定詞，則不能使用集合運算式，因為 **ALL** 限定詞自身已定義了集合。出於舊版原因，**ALL** 限定詞在 Qlik Sense 的此版本中仍然生效，但在將來的版本中可能會被移除。