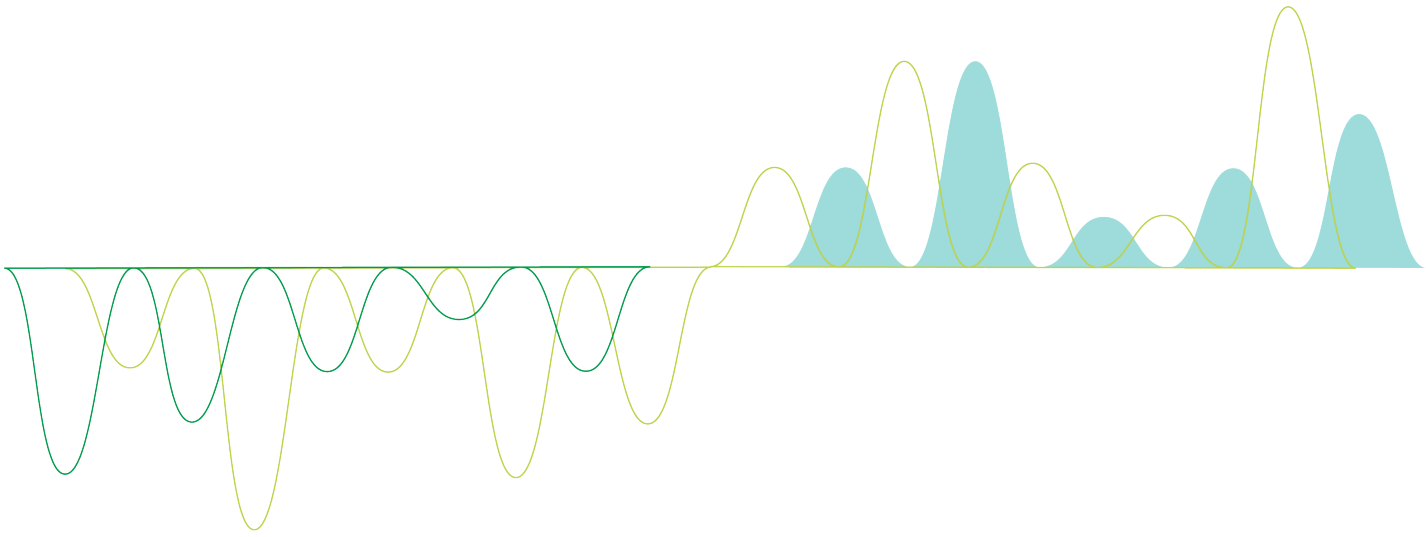
Tutorial - Passaggi successivi nella creazione degli script

Qlik Sense®
August 2023



Contents

1 Benvenuti a questo tutorial

Benvenuti a questo tutorial che introdurrà le procedure avanzate di creazione degli script in Qlik Sense.

Dopo aver acquisito familiarità con le nozioni di base di creazione degli script, sarà possibile iniziare a eseguire operazioni più sofisticate con i dati al momento del loro caricamento in Qlik Sense. Ad esempio, sarà possibile trasformare i dati utilizzando tabelle incrociate, ripulire i dati e creare e caricare dati da file Qlik denominati file QVD .

1.1 Cosa si apprenderà

After completing this tutorial, you should be comfortable with loading data using some of the more advanced scripting functions in Qlik Sense.

1.2 Chi deve seguire questo corso

È consigliabile conoscere le nozioni di base della creazione degli script in Qlik Sense. In altre parole, è consigliabile aver caricato e manipolato i dati utilizzando gli script.

Se ancora non è stato fatto, consigliamo di completare il tutorial sulla creazione degli script per principianti.

È necessario l'accesso all'editor caricamento dati e l'autorizzazione a caricare i dati in Qlik Sense Enterprise on Windows.

In genere le instruzioni si applicano anche a Qlik Sense Cloud Business.

1.3 Contenuto del pacchetto

Il pacchetto zip scaricato contiene i seguenti file di dati, necessari per completare il tutorial:

- *Cutlery.xlsx*
- *Data.xlsx*
- *Events.txt*
- *Employees.xlsx*
- *Intervals.txt*
- *Product.xlsx*
- *Salesman.xlsx*
- *Transactions.csv*
- *Winedistricts.txt*

Il pacchetto contiene anche una copia dell'app *Advanced Scripting Tutorial*. Le sezioni aggiuntive dello script nell'app contengono gli script per le altre app che verranno create in questo tutorial. È possibile caricare l'app nel proprio hub.

Si consiglia di creare l'app in prima persona come descritto nel tutorial, in modo da massimizzare l'apprendimento. Inoltre, per il corretto funzionamento dei caricamenti di dati è necessario caricare e connettersi ai file di dati nel modo descritto nel tutorial.

Tuttavia, se si verificano problemi, l'app può aiutare a risolverli. Abbiamo indicato quali segmenti dello script sono associati a ciascuna lezione.

1.4 Lezioni in questo tutorial

A seconda della propria esperienza con Qlik Sense, questo tutorial può essere completato in 3 o 4 ore. Gli argomenti sono stati progettati per essere completati in sequenza. Tuttavia, è possibile uscire e tornare in qualsiasi momento. Non sono previsti test da eseguire.

Trasformazione dei dati
Utilizzo del prefisso Crosstable
Combinazione di tabelle con Join e Keep
Utilizzo delle funzioni intra-record: Peek, Previous ed Exists
Corrispondenza degli intervalli e caricamento iterativo
Pulizia dei dati
Gestione dei dati gerarchici
File QVD

1.5 Ulteriori letture e risorse

- Per chi desidera approfondire le proprie conoscenze, Qlik offre una vasta gamma di risorse.
- La guida online di Qlik è disponibile.
- Il materiale formativo, compresi corsi online gratuiti, è disponibile in Qlik Continuous Classroom.
- In Qlik Community è possibile trovare forum di discussione, blog e altro ancora.

2 Istruzioni LOAD e SELECT

È possibile caricare dati in Qlik Sense utilizzando le istruzioni LOAD e SELECT. Ciascuna di queste istruzioni genera una tabella interna. L'istruzione LOAD viene utilizzata per caricare i dati dai file, mentre l'istruzione SELECT viene utilizzata per caricare i dati dai database.

In questo tutorial verranno utilizzati dati da file, pertanto verranno utilizzate istruzioni LOAD.

È anche possibile utilizzare un LOAD precedente per essere in grado di gestire il contenuto dei dati caricati. Ad esempio, la ridenominazione dei campi deve essere effettuata in un'istruzione LOAD, mentre l'istruzione SELECT non permette alcuna modifica ai nomi di campo.

Durante il caricamento dei dati in Qlik Sense, vengono applicate le seguenti regole:

- Qlik Sense non opera alcuna distinzione tra le tabelle generate da un'istruzione LOAD o da un'istruzione SELECT. Quindi, se vengono caricate più tabelle, non ha alcuna importanza se il caricamento viene eseguito da un'istruzione LOAD, da un'istruzione SELECT o da una combinazione delle due.
- L'ordine dei campi nell'istruzione o nella tabella originale nel database non è importante per la logica di Qlik Sense.
- I nomi di campo distinguono tra maiuscole e minuscole e vengono utilizzati per stabilire associazioni tra tabelle di dati. Per questo motivo, a volte è necessario ridenominare i campi nello script di caricamento per ottenere un modello dati desiderato.

3 Trasformazione dei dati

È possibile trasformare e manipolare i dati nell'editor caricamento dati prima di utilizzarli nell'app.

Uno dei vantaggi offerti dalla manipolazione dei dati è che è possibile scegliere di caricare solo un sottogruppo di dati da un file, ad esempio alcune colonne di una tabella, per rendere la gestione dei dati più efficiente. È inoltre possibile caricare i dati più volte per suddividere i dati non elaborati in molte nuove tabelle logiche. È infine possibile caricare i dati da più sorgenti e unirli in una tabella in Qlik Sense.

Negli esercizi seguenti verrà spiegato come caricare i dati utilizzando il prefisso Crosstable. Si apprenderà inoltre come unire le tabelle, utilizzare le funzioni intra-record, come Peek e Previous, e come caricare la stessa riga più volte utilizzando While Load.

3.1 Utilizzo del prefisso Crosstable

Le tabelle incrociate sono un tipo comune di tabella contenente una matrice di valori tra due elenchi ortogonali di dati di intestazione. Quando si hanno dei dati in tabelle incrociate, è possibile utilizzare il prefisso Crosstable per trasformare i dati e creare i campi desiderati.

Prefisso Crosstable

La tabella *Product* seguente presenta una colonna per mese e una riga per prodotto.

Tabella Product

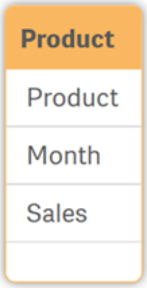
Prodotto	Jan 2014	Feb 2014	Mar 2014	Apr 2014	May 2014	Jun 2014
A	100	98	100	83	103	82
B	284	279	297	305	294	292
C	50	53	50	54	49	51

Quando si carica la tabella, l'output sarà una tabella con un campo per *Product* e un campo per ciascun mese.

Tabella Product con il campo Product e un campo per ogni mese

L'analisi di questi dati risulterà più facile se tutti i numeri si trovano in un campo e tutti i mesi in un altro. In questo caso, si avrà una tabella a tre colonne con una colonna per ogni categoria (*Product, Month, Sales*).

Tabella Product con i campi Product, Month e Sales



Il prefisso Crosstable converte i dati in una tabella con una colonna per *Month* e una colonna per *Sales*. Altrimenti detto, il prefisso converte i nomi di campo in valori di campo.

Procedere come indicato di seguito:

1. Creare una nuova app e denominarla *Advanced Scripting Tutorial*.
2. Aggiungere una nuova sezione dello script nell'**editor caricamento dati**.
3. Denominare la sezione *Product*.
4. Nel menu a destra, in **AttachedFiles**, fare clic su **Seleziona dati**.
5. Caricare e selezionare *Product.xlsx*.
6. Selezionare la tabella *Product* nella finestra **Seleziona dati da**.

   > *In **Nomi di campo** assicurarsi che l'opzione **Nomi di campo incorporati** sia selezionata per includere i nomi dei campi delle tabelle quando si caricano i dati.*

7. Fare clic su **Inserisci script**.

   Lo script avrà questo aspetto:

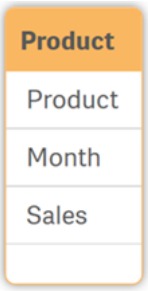
   ```
   LOAD
       Product,
       "Jan 2014",
       "Feb 2014",
       "Mar 2014",
       "Apr 2014",
       "May 2014",
       "Jun 2014"
   FROM [lib://AttachedFiles/Product.xlsx]
   (ooxml, embedded labels, table is Product);
   ```

8. Fare clic su **Carica dati**.
9. Aprire il **sistema di visualizzazione modello dati**. Il modello dati ha l'aspetto seguente:

Tabella Product con il campo Product e un campo per ogni mese

10. Fare clic sulla scheda *Product* nell'**Editor caricamento dati**.
11. Sopra l'istruzione LOAD immettere quanto segue:
    `CrossTable(Month, Sales)`
12. Fare clic su **Carica dati**.
13. Aprire il **sistema di visualizzazione modello dati**. Il modello dati ha l'aspetto seguente:

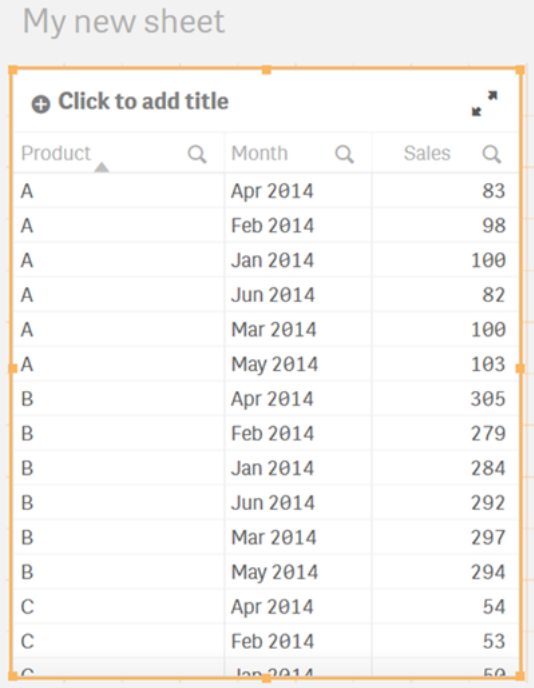
    Tabella Product con i campi Product, Month e Sales

    Si noti che in generale, i dati di input presentano una sola colonna come campo qualificatore; come chiave interna (*Product* nell'esempio precedente). Tuttavia, è possibile inserire più colonne. In questo caso, tutti i campi di qualificazione devono essere inseriti nell'istruzione LOAD prima dei campi di attributo e il terzo parametro del prefisso Crosstable deve essere utilizzato per definire il numero di campi di qualificazione. Non è possibile inserire un'istruzione LOAD precedente o un prefisso davanti alla parola chiave Crosstable. È tuttavia possibile utilizzare la concatenazione automatica.

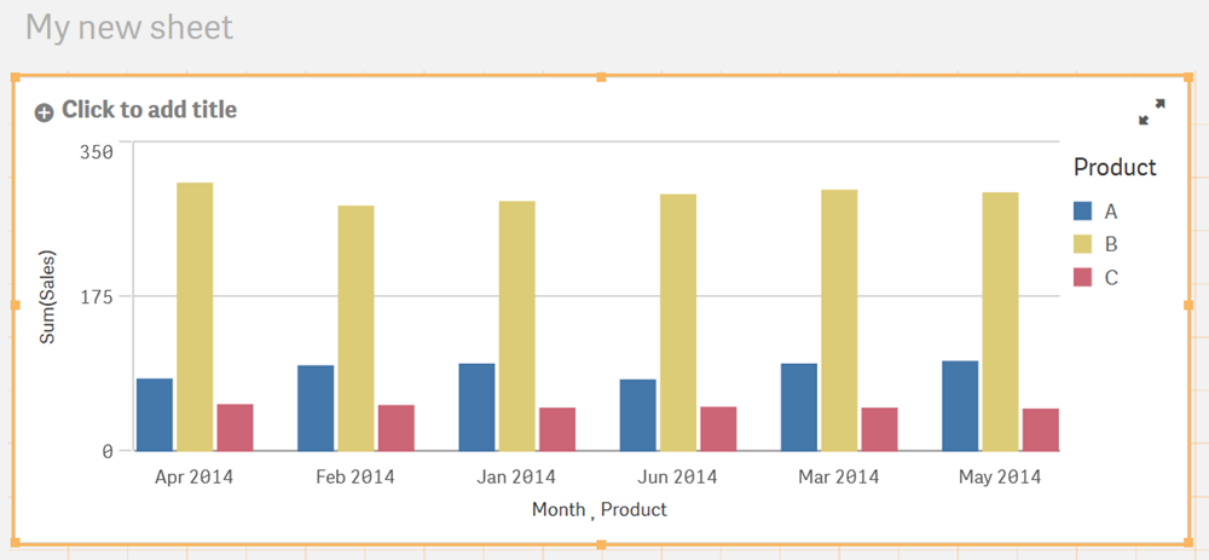
    In una tabella di Qlik Sense, i dati avranno l'aspetto seguente:

In questa tabella vengono mostrati dati caricati utilizzando il prefisso Crosstable



Ad esempio, ora è possibile creare un grafico a barre utilizzando i dati:

Grafico a barre che mostra i dati caricati utilizzando il prefisso Crosstable



Per ulteriori informazioni su Crosstable, vedere questo post di blog in Qlik Community: The Crosstable Load. I comportamenti sono trattati nel contesto di QlikView. Tuttavia, la logica si applica anche a Qlik Sense.

Interpretazione numerica che non funzionerà per i campi di attributo. Ciò significa che se nelle intestazioni delle colonne sono stati inseriti i mesi, l'interpretazione non verrà eseguita automaticamente. La soluzione consiste nell'utilizzare il prefisso Crosstable per creare una tabella temporanea ed eseguire un secondo passaggio sui dati per creare le interpretazioni come mostrato nell'esempio seguente.

Questo rappresenta solo un esempio. Non sono disponibili esercizi accompagnatori da completare in Qlik Sense.

```
tmpData:
Crosstable (MonthText, Sales)
LOAD Product, [Jan 2014], [Feb 2014], [Mar 2014], [Apr 2014], [May 2014], [Jun 2014]
FROM ...

Final:
LOAD Product,
Date(Date#(MonthText,'MMM YYYY'),'MMM YYYY') as Month,
Sales
Resident tmpData;
Drop Table tmpData;
```

Cancellazione della cache della memoria

È possibile eliminare le tabelle create per cancellare la cache della memoria. Quando il caricamento viene eseguito in una tabella temporanea, come nella sezione precedente, si consiglia di eliminarla se non è più necessaria. Ad esempio:

```
DROP TABLE Table1, Table2, Table3, Table4;
DROP TABLES Table1, Table2, Table3, Table4;
```

È anche possibile eliminare campi. Ad esempio:

```
DROP FIELD Field1, Field2, Field3, Field4;
DROP FIELDS Field1, Field2, Field3, Field4;
DROP FIELD Field1 from Table1;
DROP FIELDS Field1 from Table1;
```

Come si può notare, le parole chiave TABLE e FIELD possono essere singolari o plurali.

3.2 Combinazione di tabelle con Join e Keep

L'operazione di unione utilizza due tabelle e le unisce in una. I record della tabella risultante sono combinazioni dei record presenti nelle tabelle originali, in modo che i due record che contribuiscono a qualsiasi combinazione data nella tabella risultante presentino un valore comune per uno o più campi altrettanto comuni, formando quella che può essere definita un'unione naturale. In Qlik Sense le unioni possono essere eseguite nello script, producendo tabelle logiche.

È possibile unire le tabelle già presenti nello script. La logica di Qlik Sense non vede quindi le tabelle separate, ma il risultato dell'unione, ossia una sola tabella interna. In alcune circostanze, ciò si rivela necessario, anche se può presentare svantaggi:

- Le tabelle caricate diventano spesso più grandi, rallentando Qlik Sense.
- Alcune informazioni potrebbero andare perse: la frequenza (numero di record) all'interno della tabella originale potrebbe non essere più disponibile.

La funzionalità Keep, che ha l'effetto di ridurre una o entrambe le due tabelle all'intersezione dei dati della tabella prima della loro memorizzazione in Qlik Sense, è stata progettata per ridurre il numero dei casi in cui è necessario utilizzare unioni esplicite.

In questa documentazione il termine unione si riferisce di solito alle unioni precedenti alla creazione delle tabelle interne. Anche l'associazione, effettuata dopo la creazione delle tabelle interne, è tuttavia essenzialmente un'unione.

Join

Il modo più semplice per creare un'unione è con il prefisso Join nello script, che unisce la tabella interna con un'altra tabella con nome o con l'ultima tabella creata in precedenza. L'unione sarà un'unione esterna, che crea tutte le possibili combinazioni di valori dalle due tabelle.

Esempio:

```
LOAD  a, b, c from table1.csv;
join LOAD a, d from table2.csv;
```

La tabella interna risultante presenta i campi a, b, c e d. Il numero di record differisce a seconda dei valori di campo delle due tabelle.

I nomi dei campi da unire devono essere identici. Il numero di campi da unire è arbitrario. Di solito, le tabelle devono avere uno o più campi in comune. Nessun campo in comune rende il prodotto cartesiano delle tabelle. Sono possibili anche tutti i campi in comune, ma di solito non ha senso. A meno che nell'istruzione Join non sia specificato il nome di una tabella caricata in precedenza, il prefisso Join utilizza l'ultima tabella creata. L'ordine delle due istruzioni, pertanto, non è arbitrario.

Utilizzo di Join

Il prefisso Join esplicito nel linguaggio dello script di Qlik Sense consente di eseguire un'unione completa delle due tabelle. Il risultato è una sola tabella. Queste unioni possono spesso comportare tabelle molto grandi.

Procedere come indicato di seguito:

1. Aprire l'app *Advanced Scripting Tutorial*.
2. Aggiungere una nuova sezione dello script nell'**editor caricamento dati**.
3. Richiamare la sezione *Transactions*.
4. Nel menu a destra, in **AttachedFiles**, fare clic su **Seleziona dati**.
5. Caricare e selezionare *Transactions.csv*.

> *In **Nomi di campo** assicurarsi che l'opzione **Nomi di campo incorporati** sia selezionata per includere i nomi dei campi delle tabelle quando si caricano i dati.*

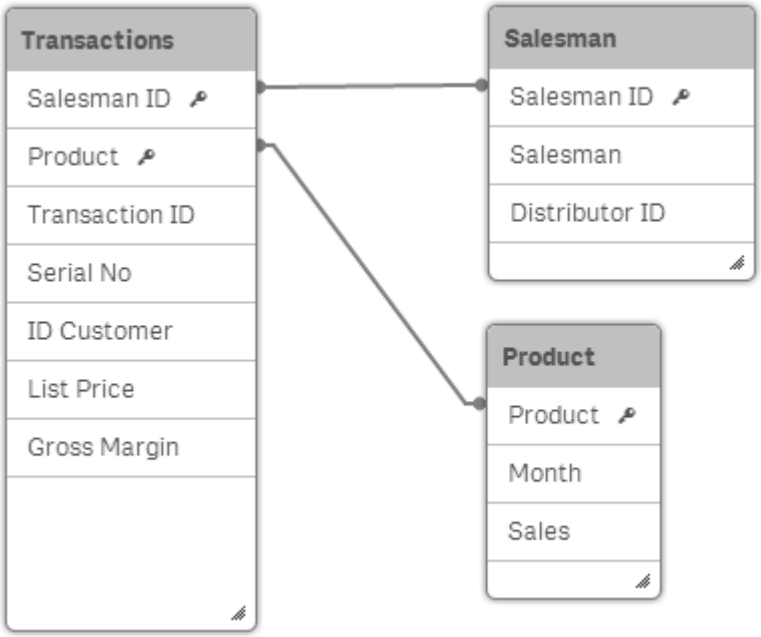
6. Nella finestra **Seleziona dati da**, fare clic su **Inserisci script**.
7. Caricare e selezionare *Salesman.xlsx*.
8. Nella finestra **Seleziona dati da**, fare clic su **Inserisci script**.

   Lo script avrà questo aspetto:

```
LOAD
    "Transaction ID",
    "Salesman ID",
    Product,
    "Serial No",
    "ID Customer",
    "List Price",
    "Gross Margin"
FROM [lib://AttachedFiles/Transactions.csv]
(txt, codepage is 28591, embedded labels, delimiter is ',', msq);

LOAD
    "Salesman ID",
    Salesman,
    "Distributor ID"
FROM [lib://AttachedFiles/Salesman.xlsx]
(ooxml, embedded labels, table is Salesman);
```

9. Fare clic su **Carica dati**.
10. Aprire il **sistema di visualizzazione modello dati**. Il modello dati ha l'aspetto seguente:

    Modello dati: Tabelle Transactions, Salesman e Product



Tuttavia la separazione delle tabelle *Transactions* e *Salesman* potrebbe non essere il risultato richiesto. Una soluzione migliore potrebbe essere quella di unire le due tabelle.

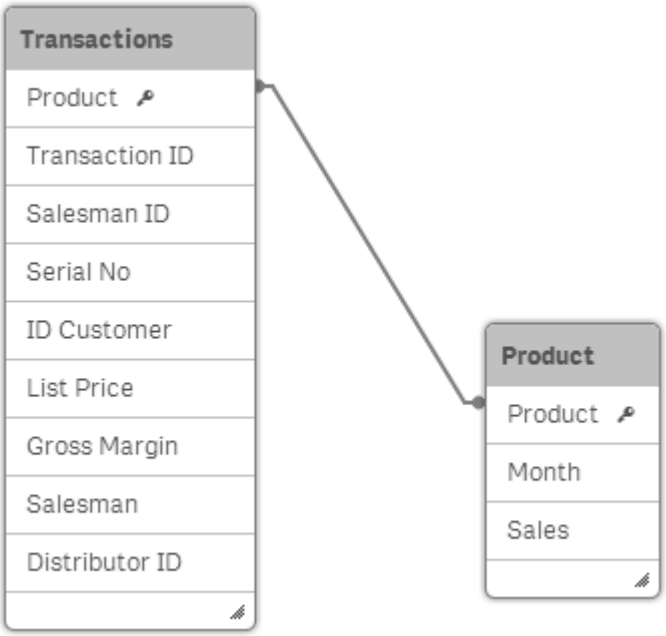
Procedere come indicato di seguito:

1. Per impostare un nome per la tabella unita, aggiungere la riga seguente sopra la prima istruzione LOAD:
   ```
   Transactions:
   ```
2. Per unire le tabelle *Transactions* e *Salesman*, sopra la seconda istruzione LOAD aggiungere la riga seguente:
   ```
   Join(Transactions)
   ```

   Lo script avrà questo aspetto:

   ```
   Transactions:
   LOAD
       "Transaction ID",
       "Salesman ID",
       Product,
       "Serial No",
       "ID Customer",
       "List Price",
       "Gross Margin"
   FROM [lib://AttachedFiles/Transactions.csv]
   (txt, codepage is 28591, embedded labels, delimiter is ',', msq);

   Join(Transactions)
   LOAD
       "Salesman ID",
       Salesman,
       "Distributor ID"
   FROM [lib://AttachedFiles/Salesman.xlsx]
   (ooxml, embedded labels, table is Salesman);
   ```
3. Fare clic su **Carica dati**.
4. Aprire il **sistema di visualizzazione modello dati**. Il modello dati ha l'aspetto seguente:

Modello dati: Tabelle Transactions e Product



Tutti i campi delle tabelle *Transactions* e *Salesman* risultano ora uniti in un'unica tabella *Transactions*.

Per ulteriori informazioni su quando utilizzare Join, vedere questi post di blog in Qlik Community: To join or not to join (Unire oppure no), Mapping as an alternative to joining (Mappare come alternativa all'unione). I comportamenti sono trattati nel contesto di QlikView. Tuttavia, la logica si applica anche a Qlik Sense.

Keep

Una delle principali funzioni di Qlik Sense è la sua capacità di generare associazioni tra tabelle invece di unirle, il che riduce lo spazio nella memoria, aumenta la velocità e offre grande flessibilità. La funzionalità Keep è stata progettata per ridurre le situazioni in cui occorre utilizzare operazioni di unione esplicite.

Il prefisso Keep compreso tra due istruzioni LOAD o SELECT riduce una o tutte e due le tabelle all'intersezione dei dati della tabella prima che vengano memorizzate in Qlik Sense. Il prefisso Keep deve sempre essere preceduto da una delle seguenti parole chiave: Inner, Left o Right. La selezione dei record dalle tabelle viene effettuata nello stesso modo di un'unione corrispondente. In ogni modo, le due tabelle non vengono unite e verranno memorizzate in . Qlik Sense come due tabelle separate.

Inner

I prefissi Join e Keep nello script di caricamento dei dati possono essere preceduti dal prefisso Inner.

Se viene utilizzato prima di Join, specifica che l'unione tra le due tabelle deve essere un'unione interna. La tabella risultante conterrà solo combinazioni fra le due tabelle con una serie di dati completa da tutte e due le parti.

Se viene utilizzato prima di Keep, specifica che le due tabelle devono essere ridotte alla loro intersezione comune prima di essere memorizzate in Qlik Sense.

Esempio:

In questi esempi vengono utilizzate le tabelle di origine *Table1* e *Table2*.

Notare che questi sono solo esempi. Non sono disponibili esercizi accompagnatori da completare in Qlik Sense.

Table 1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Inner Join

Prima, si esegue Inner Join nelle tabelle, determinando *VTable*, contenente solo una riga, l'unico record esistente in entrambe le tabelle, con i dati combinati da entrambe le tabelle.

```
VTable:
SELECT * from Table1;
inner join SELECT * from Table2;
```

VTable

A	B	C
1	aa	xx

Inner Keep

Se invece si esegue Inner Keep, avremo ancora due tabelle. Le due tabelle vengono associate mediante il campo comune *A*.

```
VTab1:
SELECT * from Table1;
VTab2:
inner keep SELECT * from Table2;
```

VTab1

A	B
1	aa

VTab2

A	C
1	xx

Left

I prefissi Join e Keep nello script di caricamento dei dati possono essere preceduti dal prefisso left.

Se viene utilizzato prima di Join, specifica che l'unione tra le due tabelle deve essere un'unione a sinistra. La tabella risultante conterrà solo combinazioni fra le due tabelle con una serie di dati completa proveniente dalla prima tabella.

Se viene utilizzato prima di Keep, specifica che la seconda tabella deve essere ridotta alla sua intersezione comune con la prima tabella prima di essere memorizzata in Qlik Sense.

Esempio:

In questi esempi vengono utilizzate le tabelle di origine *Table1* e *Table2*.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Prima, si esegue Left Join nelle tabelle, determinando *VTable*, contenente tutte le righe di *Table1*, combinata con i campi delle righe corrispondenti in *Table2*.

```
VTable:
SELECT * from Table1;
left join SELECT * from Table2;
```

VTable

A	B	C
1	aa	xx
2	cc	-
3	ee	-

Se invece si esegue Left Keep, avremo ancora due tabelle. Le due tabelle vengono associate mediante il campo comune *A*.

```
VTab1:
SELECT * from Table1;
VTab2:
left keep SELECT * from Table2;
```

VTab1

A	B
1	aa
2	cc
3	ee

VTab2

A	C
1	xx

Right

I prefissi Join e Keep nel linguaggio dello script di Qlik Sense possono essere preceduti dal prefisso right.

Se viene utilizzato prima di Join, specifica che l'unione tra le due tabelle deve essere un Right Join. La tabella risultante contiene combinazioni fra le due tabelle con una serie di dati completa dalla seconda tabella.

Se viene utilizzato prima di Keep, specifica che la prima tabella deve essere ridotta alla sua intersezione comune con la seconda tabella prima che venga memorizzata in Qlik Sense.

Esempio:

In questi esempi vengono utilizzate le tabelle di origine *Table1* e *Table2*.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Prima, si esegue Right Join nelle tabelle, determinando *VTable*, contenente tutte le righe di *Table2*, combinata con i campi delle righe corrispondenti in *Table1*.

```
VTable:
SELECT * from Table1;
right join SELECT * from Table2;
```

VTable

A	B	C
1	aa	xx
4	-	yy

Se invece si esegue Right Keep, avremo ancora due tabelle. Le due tabelle vengono associate mediante il campo comune *A*.

```
VTab1:
SELECT * from Table1;
VTab2:
right keep SELECT * from Table2;
```

VTab1

A	B
1	aa

VTab2

A	C
1	xx
4	yy

3.3 Utilizzo delle funzioni intra-record: Peek, Previous e Exists

Queste funzioni vengono utilizzate quando il valore di record di dati precedentemente caricati è necessario per valutare il record attuale.

In questa parte del tutorial si prenderanno in esame le funzioni Peek(), Previous() e Exists().

Peek()

Peek() restituisce il valore di un campo in una tabella per una riga che è già stata caricata. È possibile specificare il numero di riga così come la tabella. Se non viene specificato alcun numero di riga, verrà utilizzato l'ultimo record precedentemente caricato.

Sintassi:
```
Peek(fieldname [ , row [ , tablename ] ] )
```

Row deve essere un numero intero. 0 indica il primo record, 1 il secondo e così via. I numeri negativi indicano l'ordine dalla fine della tabella. -1 indica l'ultimo record letto.

Se non viene dichiarata alcuna riga, viene utilizzato -1.

Tablename è un'etichetta di tabella senza i due punti finali. Se non è dichiarato *tablename*, viene utilizzata la tabella attuale. Se utilizzato al di fuori dell'istruzione **LOAD** o se fa riferimento a un'altra tabella, è necessario includere *tablename*.

Previous()

Previous() restituisce il valore dell'espressione **expr** utilizzando i dati del record di input precedente che non è stato eliminato a causa di una clausola **where**. Nel primo record di una tabella interna, la funzione restituirà NULL.

Sintassi:
```
Previous(expression)
```

La funzione Previous() può essere nidificata per accedere a record ancora più precedenti. I dati vengono caricati direttamente dalla sorgente di input, rendendo possibile fare riferimento anche a campi che non sono stati caricati in Qlik Sense, vale a dire anche se non sono stati memorizzati nel database associato.

Exists()

Exists() determina se un valore di campo specifico è già stato caricato nel campo nello script di caricamento dei dati. La funzione restituisce TRUE o FALSE, quindi può essere utilizzata nella clausola **where** di un'istruzione **LOAD** o un'istruzione **IF**.

Sintassi:
```
Exists(field [, expression ] )
```

Il campo deve essere presente nei dati caricati finora dallo script. *Expression* è un'espressione che restituisce un valore da ricercare nel campo specificato. Se questa espressione viene omessa, viene utilizzato il valore del record attuale nel campo specificato.

Utilizzo di Peek() e Previous()

Nella loro forma più semplice, le funzioni Peek() e Previous() vengono utilizzate per identificare valori specifici all'interno di una tabella. Ecco un campione di dati della tabella *Employees* che verranno caricati in questo esercizio.

Campione di dati della tabella Dipendenti

Date	Hired	Terminated
1/1/2011	6	0
2/1/2011	4	2
3/1/2011	6	1
4/1/2011	5	2

Al momento questa tabella raccoglie esclusivamente dati per mese, assunzioni e cessazioni, quindi si aggiungeranno campi per *Employee Count* e *Employee Var*, utilizzando le funzioni Peek() e Previous() per visualizzare la differenza mensile nel numero totale di dipendenti.

Procedere come indicato di seguito:

1. Aprire l'app *Advanced Scripting Tutorial*.
2. Aggiungere una nuova sezione dello script nell'**editor caricamento dati**.
3. Richiamare la sezione *Employees*.
4. Nel menu a destra, in **AttachedFiles**, fare clic su **Seleziona dati**.
5. Caricare e selezionare *Employees.xlsx*.

   In Field names, assicurarsi che l'opzione Embedded field names sia selezionata per includere i nomi dei campi delle tabelle quando si caricano i dati.

6. Nella finestra **Seleziona dati da**, fare clic su **Inserisci script**.

   Lo script avrà questo aspetto:

   ```
   LOAD
       "Date",
       Hired,
       Terminated
   FROM [lib://AttachedFiles/Employees.xlsx]
   (ooxml, embedded labels, table is Sheet1);
   ```

7. Modificare lo script nel modo seguente:

   ```
   [Employees Init]:
   LOAD
       rowno() as Row,
       Date(Date) as Date,
       Hired,
       Terminated,
       If(rowno()=1, Hired-Terminated,  peek([Employee Count], -1)+(Hired-Terminated)) as
   [Employee Count]
   FROM [lib://AttachedFiles/Employees.xlsx]
   (ooxml, embedded labels, table is Sheet1);
   ```

   Le date nel campo *Date* del foglio Excel sono nel formato MM/GG/AAAA. Per assicurare la corretta interpretazione delle date che utilizzano il formato specificato dalle variabili di sistema, al campo *Date* viene applicata la funzione Date.

La funzione Peek() consente di identificare qualsiasi valore caricato per un campo definito. Nell'espressione si osserverà prima di tutto se rowno() è uguale a 1. Se è uguale a 1, *Employee Count* non sarà presente, quindi si compilerà il campo con la differenza tra *Hired* e *Terminated*.

Se rowno() è maggiore di 1, si farà riferimento al campo *Employee Count* dell'ultimo mese e si utilizzerà tale numero per aggiungere la differenza tra i dipendenti *Hired* e i dipendenti *Terminated* del mese.

Notare che nella funzione Peek() si utilizza (-1). In questo modo Qlik Sense cercherà il record sopra il record corrente. Se (-1) non è specificato, Qlik Sense presuppone che si desideri utilizzare il record precedente.

8. Aggiungere quanto segue alla fine dello script:

```
[Employee Count]:
LOAD
       Row,
       Date,
       Hired,
       Terminated,
       [Employee Count],
       If(rowno()=1,0,[Employee Count]-Previous([Employee Count])) as [Employee Var]
Resident [Employees Init] Order By Row asc;
Drop Table [Employees Init];
```

La funzione Previous() consente di identificare l'ultimo valore caricato per un campo definito. Nell'espressione si osserverà prima di tutto se rowno() è uguale a 1. Se è uguale a 1, Employee Var non sarà presente poiché non vi sono record relativi al campo *Employee Count* del mese precedente. Pertanto si immette semplicemente il valore 0.

Se rowno() è maggiore di 1, sarà presente un campo *Employee Var*, quindi si osserverà il campo *Employee Count* dell'ultimo mese e si sottrarrà tale numero dal campo *Employee Count* del mese attuale per creare il valore nel campo *Employee Var*.

Lo script avrà questo aspetto:

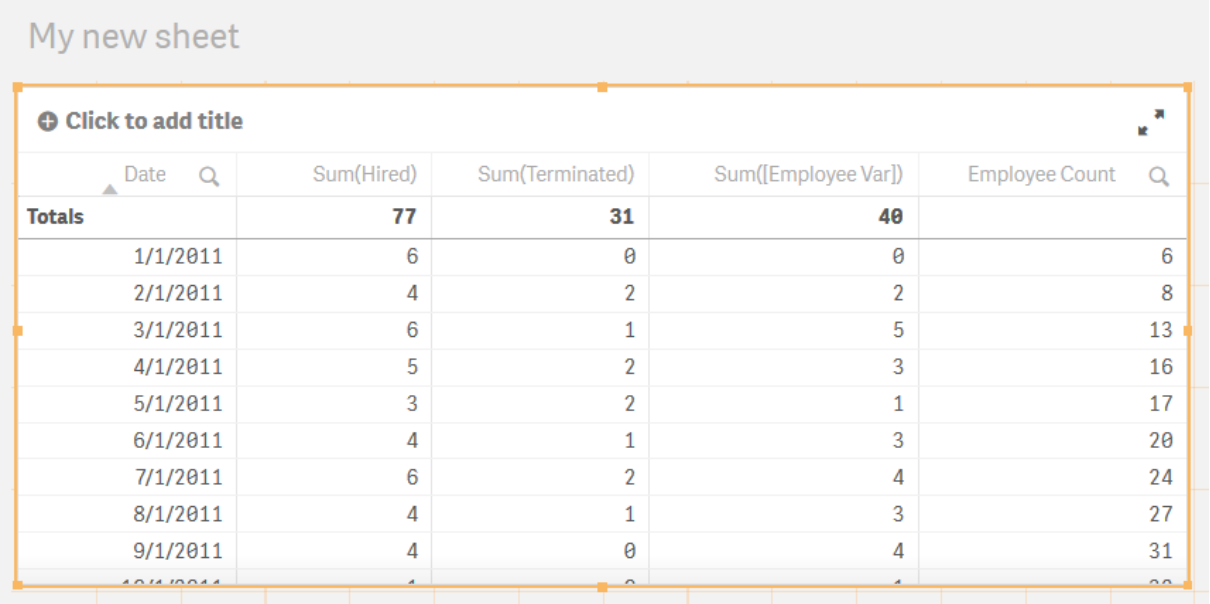
```
[Employees Init]:
LOAD
    rowno() as Row,
    Date(Date) as Date,
    Hired,
    Terminated,
    If(rowno()=1, Hired-Terminated,  peek([Employee Count], -1)+(Hired-Terminated)) as
[Employee Count]
FROM [lib://AttachedFiles/Employees.xlsx]
(ooxml, embedded labels, table is Sheet1);

[Employee Count]:
LOAD
    Row,
    Date,
    Hired,
    Terminated,
    [Employee Count],
```

```
    If(rowno()=1,0,[Employee Count]-Previous([Employee Count])) as [Employee Var]
Resident [Employees Init] Order By Row asc;
Drop Table [Employees Init];
```

9. Fare clic su **Carica dati**.

   In un nuovo foglio della panoramica App, creare una tabella utilizzando *Date, Hired, Terminated, Employee Count* e *Employee Var* come colonne. La tabella risultante avrà l'aspetto seguente:

   Tabella dopo l'utilizzo di Peek e Previous nello script



Date	Sum(Hired)	Sum(Terminated)	Sum([Employee Var])	Employee Count
Totals	**77**	**31**	**40**	
1/1/2011	6	0	0	6
2/1/2011	4	2	2	8
3/1/2011	6	1	5	13
4/1/2011	5	2	3	16
5/1/2011	3	2	1	17
6/1/2011	4	1	3	20
7/1/2011	6	2	4	24
8/1/2011	4	1	3	27
9/1/2011	4	0	4	31

Peek() e Previous() consentono di puntare a righe definite all'interno di una tabella. La più grande differenza tra le due funzioni è che la funzione Peek() consente all'utente di osservare un campo che non era stato caricato nello script in precedenza, mentre la funzione Previous() consente di osservare esclusivamente un campo caricato in precedenza. Previous() agisce sull'input dell'istruzione LOAD, mentre Peek() agisce sull'output dell'istruzione LOAD. Uguale alla differenza tra RecNo() e RowNo(). Ciò significa che il comportamento delle due funzioni sarà diverso in presenza di una clausola Where.

Quindi sarebbe meglio utilizzare la funzione Previous() quando è necessario mostrare il valore attuale invece del valore precedente. Nell'esempio la varianza dei dipendenti viene calcolata mese per mese.

È preferibile utilizzare la funzione Peek() quando si desidera puntare a un campo non caricato in precedenza nella tabella o quando è necessario puntare a una riga specifica. Ciò è mostrato nell'esempio in cui il campo *Employee Count* è stato calcolato mediante l'osservazione di *Employee Count* per il mese precedente ed è stata aggiunta la differenza tra le assunzioni e le cessazioni dei dipendenti per il mese attuale. Tenere presente che *Employee Count* non era un campo del file originale.

Per ulteriori informazioni su quando utilizzare Peek() e Previous(), vedere questo post di blog in Qlik Community: Peek() vs Previous() – When to Use Each. I comportamenti sono trattati nel contesto di QlikView. Tuttavia, la logica si applica anche a Qlik Sense.

Utilizzo di Exists()

La funzione Exists() viene spesso utilizzata unitamente alla clausola Where nello script per caricare i dati se i dati correlati sono già stati caricati nel modello dati.

Nell'esempio seguente verrà utilizzata anche la funzione Dual() per assegnare valori numerici alle stringhe.

Procedere come indicato di seguito:

1. Creare una nuova app e assegnarle un nome.
2. Aggiungere una nuova sezione dello script nell'**editor caricamento dati**.
3. Richiamare la sezione *People*.
4. Immettere lo script seguente:

```
//Add dummy people data
PeopleTemp:
LOAD * INLINE [
PersonID, Person
1, Jane
2, Joe
3, Shawn
4, Sue
5, Frank
6, Mike
7, Gloria
8, Mary
9, Steven,
10, Bill
];

//Add dummy age data
AgeTemp:
LOAD * INLINE [
PersonID, Age
1, 23
2, 45
3, 43
4, 30
5, 40
6, 32
7, 45
8, 54
9,
10, 61
11, 21
12, 39
];

//LOAD new table with people
People:
NoConcatenate LOAD
    PersonID,
```

```
    Person
Resident PeopleTemp;

Drop Table PeopleTemp;

//Add age and age bucket fields to the People table
Left Join (People)
LOAD
    PersonID,
    Age,
       If(IsNull(Age) or Age='', Dual('No age', 5),
        If(Age<25, Dual('Under 25', 1),
         If(Age>=25 and Age <35, Dual('25-34', 2),
          If(Age>=35 and Age<50, Dual('35-49' , 3),
           If(Age>=50, Dual('50 or over', 4)
            ))))) as AgeBucket
Resident AgeTemp
Where Exists(PersonID);

DROP Table AgeTemp;
```

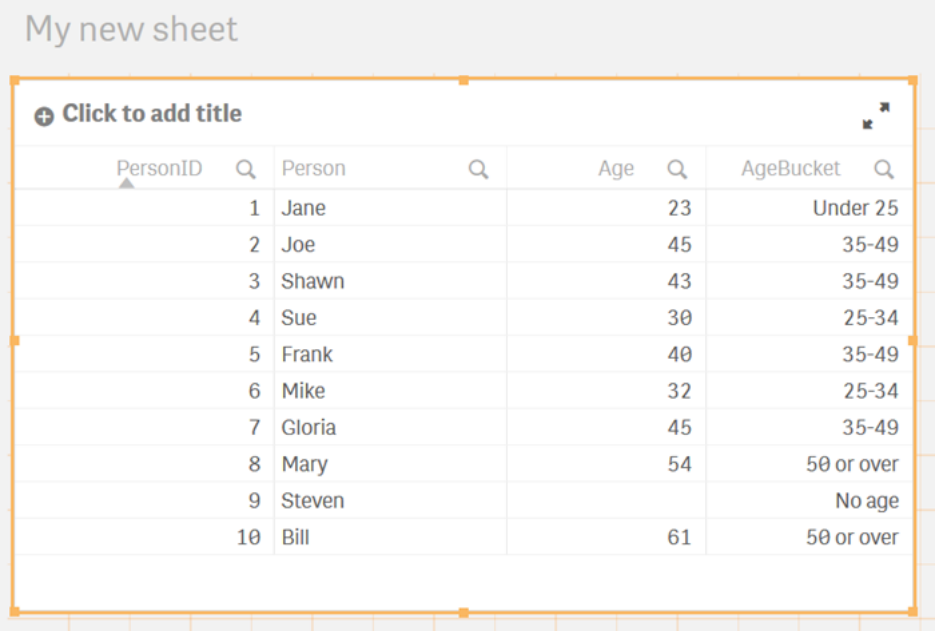
5. Fare clic su **Carica dati**.

   Nello script i campi *Age* e *AgeBucket* vengono caricati solo se il campo *PersonID* è già stato caricato nel modello dati.

   Si prega di notare che nella tabella *AgeTemp* sono presenti le età inserite per *PersonID* 11 e 12, tuttavia, visto che tali ID non sono stati caricati nel modello dati (nella tabella *People*), vengono esclusi dalla clausola Where Exists(PersonID). Questa clausola può venire scritta anche nel modo seguente: Where Exists(PersonID, PersonID).

   L'output dello script avrà il seguente aspetto:

Tabella dopo l'utilizzo di Exists nello script

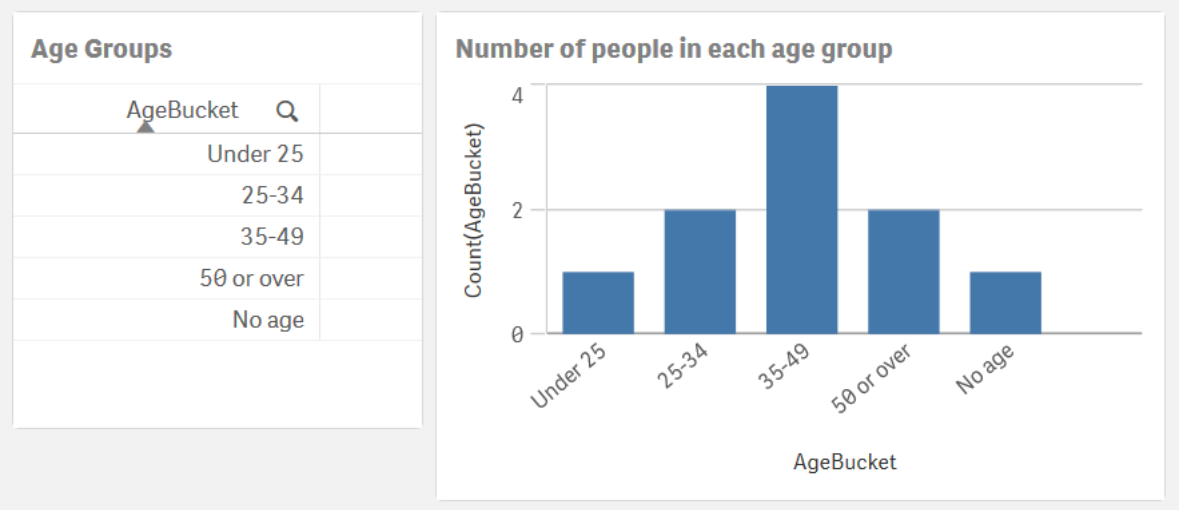


PersonID	Person	Age	AgeBucket
1	Jane	23	Under 25
2	Joe	45	35-49
3	Shawn	43	35-49
4	Sue	30	25-34
5	Frank	40	35-49
6	Mike	32	25-34
7	Gloria	45	35-49
8	Mary	54	50 or over
9	Steven		No age
10	Bill	61	50 or over

Se nessuno dei campi *PersonID* nella tabella *AgeTemp* viene caricato nel modello dati, i campi *Age* e *AgeBucket* non verranno uniti alla tabella *People*. L'utilizzo della funzione Exists() può evitare la presenza di record/dati orfani nel modello dati, vale a dire campi *Age* e *AgeBucket* a cui non sono state associate delle persone.

6. Creare un nuovo foglio e assegnargli un nome.
7. Aprire il nuovo foglio e fare clic su **Modifica foglio**.
8. Aggiungere una tabella standard al foglio con la dimensione *AgeBucket* e assegnare alla visualizzazione il nome *Age Groups*.
9. Aggiungere un grafico a barre al foglio con la dimensione *AgeBucket* e la misura *Count([AgeBucket])*. Denominare la visualizzazione *Number of people in each age group*.
10. Regolare le proprietà della tabella e del grafico a barre in base alle preferenze, quindi fare clic su **Fine**. Il foglio dovrebbe ora avere l'aspetto seguente:

Foglio con raggruppamenti per età



La funzione Dual() risulta utile nello script o in un'espressione del grafico quando occorre assegnare un valore numerico a una stringa.

All'interno dello script è presente un'applicazione che carica le età e si è deciso di suddividerle in fasce in modo da poter creare visualizzazioni basate sulle fasce d'età invece che sulle età effettive. È stata creata una fascia per le persone minori di 25 anni, tra i 25 anni e i 35 anni e così via. Grazie all'utilizzo della funzione Dual () alle fasce di età è possibile assegnare un valore numerico che può essere in seguito utilizzato per ordinare le fasce d'età in una casella di elenco o in un grafico. Quindi, come nel foglio dell'app, l'ordinamento inserisce "No age" alla fine dell'elenco.

Per ulteriori informazioni su Exists() e Dual(), vedere questo post di blog in Qlik Community: Dual & Exists – Useful Functions (Dual & Exists - Funzioni utili)

3.4 Corrispondenza degli intervalli e caricamento iterativo

Il prefisso Intervalmatch in un'istruzione LOAD o SELECT viene utilizzato per collegare valori numerici discreti a uno o più intervalli numerici. Si tratta di una funzione molto avanzata che può essere utilizzata, ad esempio, negli ambienti di produzione.

Utilizzo del prefisso IntervalMatch()

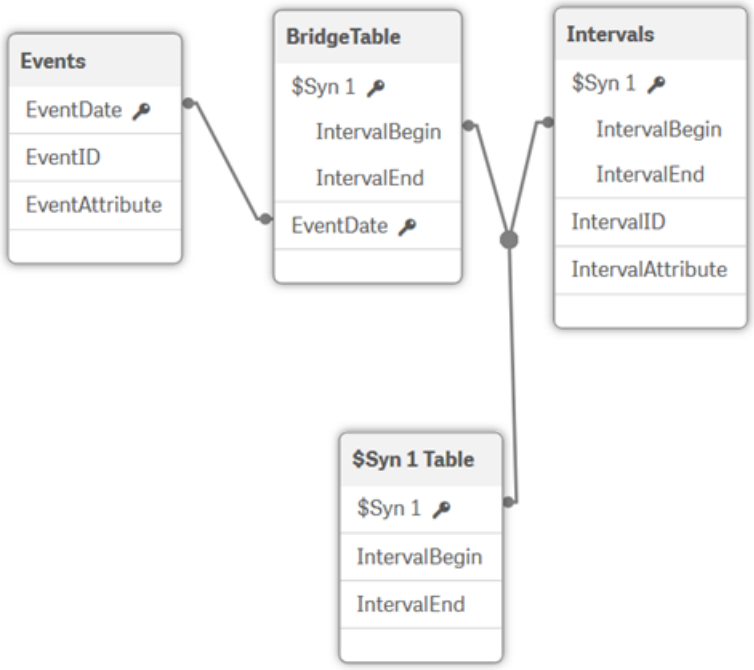
La corrispondenza dell'intervallo più semplice consiste nel disporre di numeri o date (eventi) in una tabella e di un elenco di intervalli in una seconda tabella. L'obiettivo è quello di collegare le due tabelle. In generale, si tratta di una relazione molti a molti, vale a dire che un intervallo può presentare diverse date che gli appartengono e una data può appartenere a diversi intervalli. Per eseguire questo collegamento, occorre creare una tabella ponte tra le due tabelle originali. Sono disponibili diversi modi per eseguire questa operazione.

Il modo più semplice per risolvere questo problema in Qlik Sense è quello di utilizzare il prefisso IntervalMatch () davanti a un'istruzione LOAD o SELECT. L'istruzione LOAD/SELECT deve contenere solo due campi, i campi From e To che definiscono gli intervalli. Il prefisso IntervalMatch() genererà quindi tutte le combinazioni tra gli intervalli caricati e un campo numerico caricato in precedenza, specificato come parametro per il prefisso.

Procedere come indicato di seguito:

1. Creare una nuova app e assegnarle un nome.
2. Aggiungere una nuova sezione dello script nell'**editor caricamento dati**.
3. Richiamare le sezioni *Events*.
4. Nel menu a destra, in **AttachedFiles**, fare clic su **Seleziona dati**.
5. Caricare e selezionare *Events.txt*.
6. Nella finestra **Seleziona dati da**, fare clic su **Inserisci script**.
7. Caricare e selezionare *Intervals.txt*.
8. Nella finestra **Seleziona dati da**, fare clic su **Inserisci script**.
9. Nello script, denominare la prima tabella *Eventi* e la seconda *Intervals*.
10. Alla fine dello script aggiungere IntervalMatch per creare una terza tabella che serva da ponte tra le due prime tabelle:
    ```
    BridgeTable:
    IntervalMatch (EventDate)
    LOAD distinct IntervalBegin, IntervalEnd
    Resident Intervals;
    ```
11. Lo script avrà questo aspetto:
    ```
    Events:
    LOAD
        EventID,
        EventDate,
        EventAttribute
    FROM [lib://AttachedFiles/Events.txt]
    (txt, utf8, embedded labels, delimiter is '\t', msq);

    Intervals:
    LOAD
        IntervalID,
        IntervalAttribute,
        IntervalBegin,
        IntervalEnd
    FROM [lib://AttachedFiles/Intervals.txt]
    (txt, utf8, embedded labels, delimiter is '\t', msq);

    BridgeTable:
    IntervalMatch (EventDate)
    LOAD distinct IntervalBegin, IntervalEnd
    Resident Intervals;
    ```
12. Fare clic su **Carica dati**.
13. Aprire il **sistema di visualizzazione modello dati**. Il modello dati ha l'aspetto seguente:

Modello dati: tabelle Events, BridgeTable, Intervals e $Syn1



Il modello dati contiene una chiave composita (i campi *IntervalBegin* e *IntervalEnd*) che, in Qlik Sense, diventerà una chiave sintetica:

Le tabelle di base sono:

- La tabella *Events* che contiene esattamente un record per evento.
- La tabella *Intervals* che contiene esattamente un record per intervallo.
- La tabella ponte che contiene esattamente un record per combinazione di evento e intervallo e che collega le due tabelle precedenti.

Tenere presente che un evento può appartenere a diversi intervalli se gli intervalli si sovrappongono. Inoltre un intervallo può presentare diversi eventi che gli appartengono.

Questo modello dati è ideale poiché è normalizzato e compatto. La tabella *Events* e la tabella *Intervals* sono entrambe inalterate e contengono il numero di record originale. Tutti i calcoli di Qlik Sense applicati a queste tabelle, ad esempio Count(EventID), funzioneranno e saranno valutati correttamente.

Per ulteriori informazioni su IntervalMatch(), vedere questo post di blog in Qlik Community: Using IntervalMatch() (Utilizzo di IntervalMatch())

Utilizzo di un ciclo While e del caricamento iterativo IterNo()

È possibile ottenere quasi la stessa tabella ponte utilizzando il ciclo While e IterNo() che crea valori enumerabili tra il limite inferiore e superiore dell'intervallo.

È possibile creare un ciclo all'interno dell'istruzione LOAD mediante la clausola While. Ad esempio:

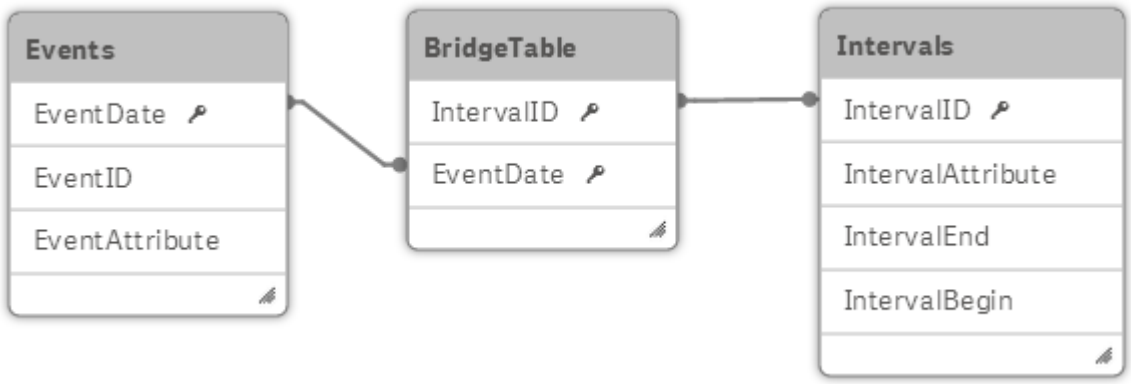
```
LOAD Date, IterNo() as Iteration From … While IterNo() <= 4;
```

Un'istruzione LOAD di questo tipo eseguirà il ciclo su ogni record di input ed eseguirà il caricamento ripetutamente purché l'espressione nella clausola While sia vera. La funzione IterNo() restituisce "1" nella prima iterazione, "2" nella seconda e così via.

Per gli intervalli si dispone di una chiave primaria, IntervalID, quindi l'unica differenza nello script sarà come viene creata la tabella ponte:

Procedere come indicato di seguito:

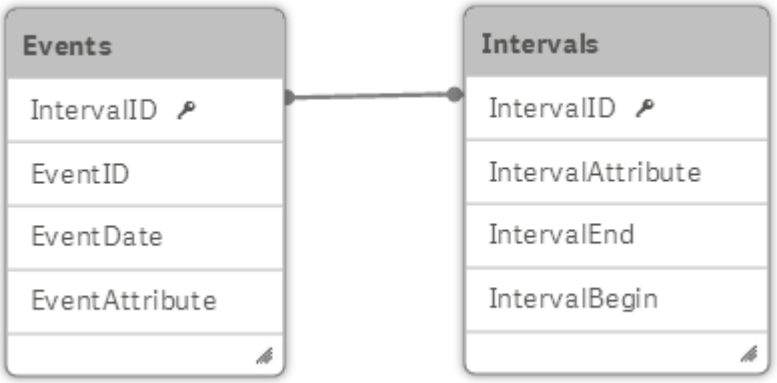
1. Sostituire le istruzioni Bridgetable esistenti con lo script seguente:
   ```
   BridgeTable:
   LOAD distinct * Where Exists(EventDate);
   LOAD IntervalBegin + IterNo() - 1 as EventDate, IntervalID
       Resident Intervals
       While IntervalBegin + IterNo() - 1 <= IntervalEnd;
   ```
2. Fare clic su **Carica dati**.
3. Aprire il **sistema di visualizzazione modello dati**. Il modello dati ha l'aspetto seguente:
   Modello dati: Tabelle Events, BridgeTable e Intervals

   In generale, la soluzione con le tre tabelle è la migliore poiché consente di instaurare una relazione molti a molti tra gli intervalli e gli eventi. Tuttavia, una situazione molto comune è quella di un evento che può appartenere solo a un unico intervallo. In questo caso, la tabella ponte non è realmente necessaria: *IntervalID* può essere memorizzato direttamente nella tabella degli eventi. Sono disponibili diversi modi per eseguire questa operazione, tuttavia quello più utile è unire Bridgetable alla tabella *Events*.
4. Aggiungere lo script seguente alla fine dello script:
   ```
   Join (Events)
   LOAD EventDate, IntervalID
   Resident BridgeTable;
   ```

```
Drop Table BridgeTable;
```

5. Fare clic su **Carica dati**.
6. Aprire il **sistema di visualizzazione modello dati**. Il modello dati ha l'aspetto seguente:
   Modello dati: Tabelle Events e Intervals



Intervalli aperti e chiusi

Lo stato aperto o chiuso di un intervallo viene determinato dai punti finali, indipendentemente dal fatto che siano inclusi o meno nell'intervallo.

- Se vengono inclusi i punti finali, si parla di intervallo chiuso:
  $[a,b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$
- Se non vengono inclusi i punti finali, si parla di intervallo aperto:
  $]a,b[= \{x \in \mathbb{R} \mid a < x < b\}$
- Se viene incluso un punto finale, si parla di intervallo semi aperto:
  $[a,b[= \{x \in \mathbb{R} \mid a \leq x < b\}$

Nel caso in cui gli intervalli si sovrappongano e un numero possa appartenere a più di un intervallo, in generale, occorre utilizzare intervalli chiusi.

Tuttavia, in alcuni casi, non si desidera la sovrapposizione degli intervalli, ma che un numero appartenga a un unico intervallo. Di conseguenza, il problema sorge nel caso in cui un punto corrisponde alla fine di un intervallo e, contemporaneamente, all'inizio di quello successivo. Un numero con questo valore verrà attribuito a entrambi gli intervalli. Gli intervalli semi aperti rappresentano quindi la soluzione ideale.

Una soluzione pratica al problema è quella di sottrarre una cifra minima dal valore finale di tutti gli intervalli, creando quindi intervalli chiusi ma che non si sovrappongono. Se i numeri disponibili sono date, il modo più semplice per eseguire questa operazione è utilizzare la funzione DayEnd(), che restituisce l'ultimo millisecondo del giorno:

```
Intervals:
LOAD…, DayEnd(IntervalEnd – 1) as IntervalEnd From Intervals;
```

È anche possibile sottrarre una piccola cifra manualmente. In questo caso, assicurarsi che la cifra sottratta non sia eccessivamente ridotta dato che l'operazione verrà arrotondata in base a 52 cifre binarie significative (14 cifre decimali). Se si utilizza una cifra eccessivamente ridotta, la differenza non sarà significativa e si dovrà riutilizzare il numero originale.

4 Pulizia dei dati

Talvolta i dati sorgente caricati in Qlik Sense non corrispondono necessariamente ai dati che si desidera avere nell'applicazione Qlik Sense. Qlik Sense offre una gamma di funzioni e istruzioni che consentono di convertire i dati nel formato desiderato.

È possibile utilizzare il mapping in uno script di Qlik Sense per sostituire o modificare i valori o i nomi di campo durante l'esecuzione dello script, quindi il mapping consente di pulire i dati e renderli più uniformi o sostituire in parte o completamente un valore di campo.

Quando si caricano i dati da tabelle differenti, i valori di campo che indicano lo stesso elemento non sono sempre associati a nomi coerenti. Poiché questa mancanza di coerenza ostacola le associazioni, è necessario risolvere il problema. Per risolvere il problema in modo elegante, creare una tabella di mapping che consenta di confrontare i valori di campo.

4.1 Tabelle di mapping

Le tabelle caricate con Mapping load o Mapping select vengono trattate in modo diverso da altre tabelle. Vengono salvate in un'area di memoria separata e utilizzate solo come tabelle di mapping durante l'esecuzione dello script. Una volta eseguito lo script, queste tabelle verranno eliminate automaticamente.

Regole:

- Una tabella di mapping deve avere due colonne: una con i valori di confronto e una con i valori di mapping desiderati.
- Le due colonne devono avere un nome, ma i nomi non hanno rilevanza in se stessi. I nomi delle colonne non hanno alcuna connessione con i nomi dei campi delle tabelle interne standard.

4.2 Funzioni e istruzioni Mapping

In questo tutorial verranno trattate le funzioni/istruzioni di mapping seguenti:

• Prefisso Mapping

• ApplyMap()

• MapSubstring()

• Istruzione Map … Using

• Istruzione Unmap

4.3 Prefisso Mapping

Il prefisso Mapping viene utilizzato in uno script per creare una tabella di mapping. È quindi possibile utilizzare la tabella di mapping con la funzione ApplyMap(), la funzione MapSubstring() o l'istruzione Map … Using.

Procedere come indicato di seguito:

1. Creare una nuova app e assegnarle un nome.
2. Aggiungere una nuova sezione dello script nell'**editor caricamento dati**.
3. Richiamare la sezione *Countries*.
4. Immettere lo script seguente:

```
CountryMap:
MAPPING LOAD * INLINE [
Country, NewCountry
U.S.A., US
U.S., US
United States, US
United States of America, US
];
```

   La tabella *CountryMap* memorizza due colonne: *Country* e *NewCountry*. La colonna *Country* memorizza i vari modi in cui il nome del paese è stato immesso nel campo *Country*. La colonna *NewCountry* memorizza il metodo di mapping dei valori. Questa tabella di mapping verrà utilizzata per memorizzare i valori uniformi del paese *US* nel campo *Country*. Ad esempio, se *U.S.A.* viene memorizzato nel campo *Country*, eseguire il mapping a *US*.

4.4 Funzione ApplyMap()

Utilizzare ApplyMap() per sostituire i dati in un campo in base a una tabella di mapping creata in precedenza La tabella di mapping deve essere caricata prima di poter utilizzare la funzione ApplyMap(). I dati nella tabella *Data.xlsx* che verrà caricata avranno l'aspetto seguente:

Tabella dati

ID	Nome	Paese	Codice
1	John Black	USA	SDFGBS1DI
2	Steve Johnson	U.S.	2ABC
3	Mary White	Stati Uniti	DJY3DFE34
4	Susan McDaniels	u	DEF5556
5	Dean Smith	US	KSD111DKFJ1

Notare come il nome del paese immesso è scritto in modi diversi. Per poter uniformare il campo relativo al paese, la tabella di mapping viene caricata, quindi viene utilizzata la funzione **ApplyMap()**.

Procedere come indicato di seguito:

1. Sotto lo script immesso sopra, selezionare e caricare *Data.xlsx*, quindi inserire lo script.
2. Sopra l'istruzione LOAD appena creata immettere quanto segue:

```
Data:
```

Lo script avrà questo aspetto:

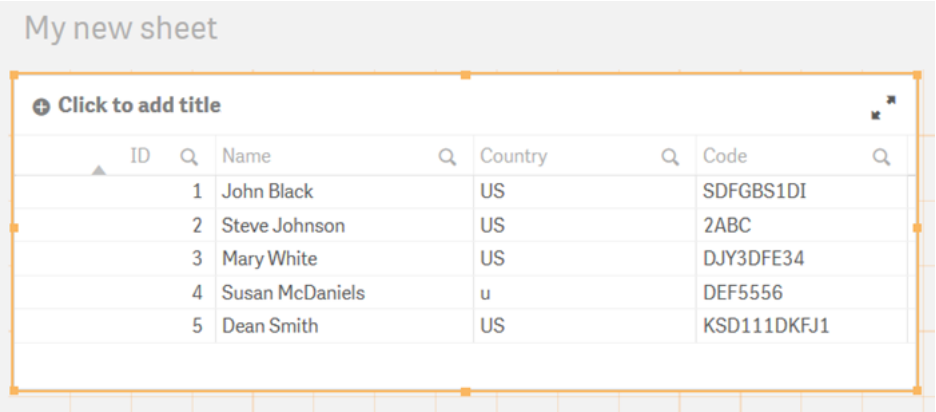
```
CountryMap:
MAPPING LOAD * INLINE [
    Country, NewCountry
    U.S.A., US
    U.S., US
    United States, US
    United States of America, US
];

Data:
LOAD
    ID,
    Name,
    Country,
    Code
FROM [lib://AttachedFiles/Data.xlsx]
(ooxml, embedded labels, table is Sheet1);
```

3. Modificare la riga contenente `Country,` nel modo seguente:
   `ApplyMap('CountryMap', Country) as Country,`

   Nel primo parametro della funzione ApplyMap() il nome della mappa è racchiuso tra virgolette singole. Il secondo parametro corrisponde al campo con i dati da sostituire.

4. Fare clic su **Carica dati**.
   La tabella risultante avrà l'aspetto seguente:

   Tabella in cui sono presenti dati caricati utilizzando la funzione ApplyMap()



ID	Name	Country	Code
1	John Black	US	SDFGBS1DI
2	Steve Johnson	US	2ABC
3	Mary White	US	DJY3DFE34
4	Susan McDaniels	u	DEF5556
5	Dean Smith	US	KSD111DKFJ1

   Le diverse grafie di *United States* sono state tutte modificate in *US*. Un record non era stato scritto correttamente, quindi la funzione ApplyMap() non ha modificato quel valore di campo. La funzione ApplyMap() consente di utilizzare il terzo parametro per aggiungere un'espressione predefinita se la tabella di mapping non presenta un valore corrispondente.

5. Aggiungere `'US'` come terzo parametro della funzione ApplyMap() per gestire quei casi in cui il nome del paese non viene immesso in modo corretto:
   `ApplyMap('CountryMap', Country, 'US') as Country,`

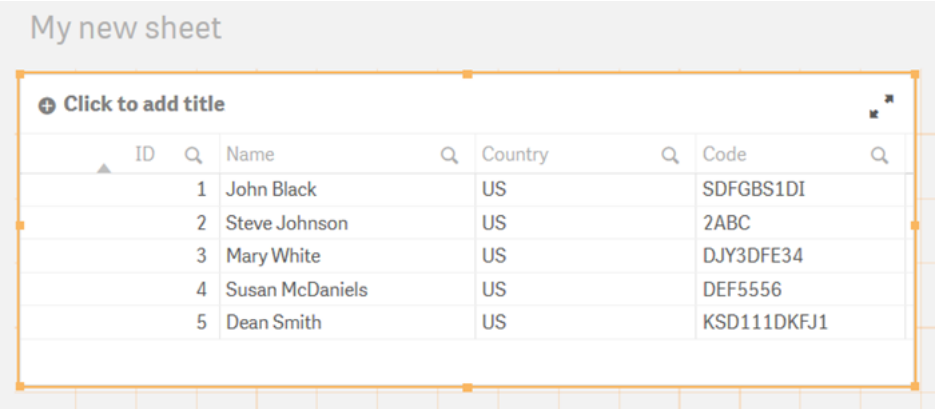
Lo script avrà questo aspetto:

```
CountryMap:
MAPPING LOAD * INLINE [
    Country, NewCountry
    U.S.A., US
    U.S., US
    United States, US
    United States of America, US
];

Data:
LOAD
    ID,
    Name,
    ApplyMap('CountryMap', Country, 'US') as Country,
    Code
FROM [lib://AttachedFiles/Data.xlsx]
(ooxml, embedded labels, table is Sheet1);
```

6. Fare clic su **Carica dati**.

   La tabella risultante avrà l'aspetto seguente:

   Tabella in cui sono presenti dati caricati utilizzando la funzione ApplyMap



Per ulteriori informazioni su ApplyMap(), vedere questo post di blog in Qlik Community: Don't join - use Applymap instead (Non unire e usare invece Applymap)

4.5 Funzione MapSubstring()

La funzione MapSubstring() consente di eseguire il mapping di alcune parti di un campo.

Nella tabella creata da ApplyMap() si desidera scrivere i numeri in caratteri alfabetici e si utilizzerà quindi la funzione MapSubstring() per sostituire i dati numerici con il testo corrispondente.

A tale scopo occorre creare innanzitutto una tabella di mapping.

Procedere come indicato di seguito:

1. Aggiungere le righe di script seguenti alla fine della sezione *CountryMap*, ma prima della sezione *Data*.

   ```
   CodeMap:
   MAPPING LOAD * INLINE [
   F1, F2
   1, one
   2, two
   3, three
   4, four
   5, five
   11, eleven
   ];
   ```

   Nella tabella *CodeMap* vengono mappati i numeri da 1 a 5 e 11.

2. Nella sezione *Data* dello script, modificare l'istruzione `Code` nel modo seguente:

   ```
   MapSubString('CodeMap', Code) as Code
   ```

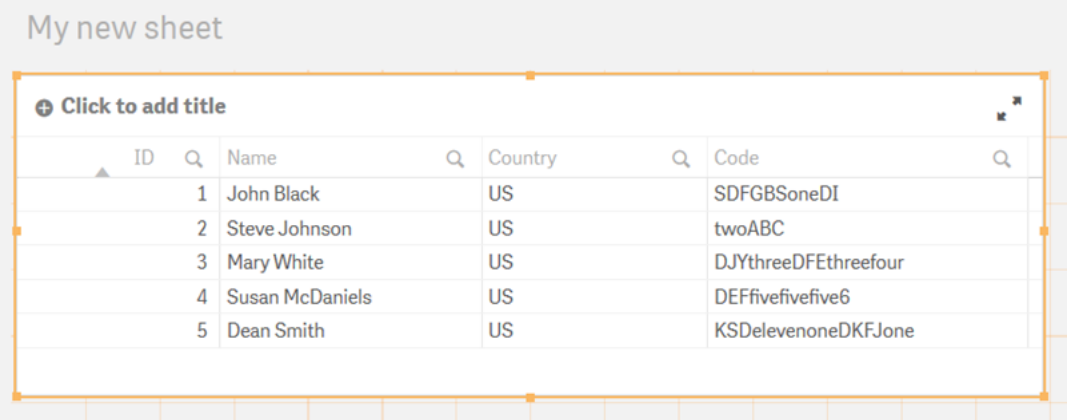
   Lo script avrà questo aspetto:

   ```
   CountryMap:
   MAPPING LOAD * INLINE [
       Country, NewCountry
       U.S.A., US
       U.S., US
       United States, US
       United States of America, US
   ];

   CodeMap:
   MAPPING LOAD * INLINE [
   F1, F2
   1, one
   2, two
   3, three
   4, four
   5, five
   11, eleven
   ];

   Data:
   LOAD
       ID,
       Name,
       ApplyMap('CountryMap', Country, 'US') as Country,
       MapSubString('CodeMap', Code) as Code
   FROM [lib://AttachedFiles/Data.xlsx]
   (ooxml, embedded labels, table is Sheet1);
   ```

3. Fare clic su **Carica dati**.

   La tabella risultante avrà l'aspetto seguente:

Tabella in cui sono presenti dati caricati utilizzando la funzione MapSubString



ID	Name	Country	Code
1	John Black	US	SDFGBSoneDI
2	Steve Johnson	US	twoABC
3	Mary White	US	DJYthreeDFEthreefour
4	Susan McDaniels	US	DEFfivefivefive6
5	Dean Smith	US	KSDelevenoneDKFJone

I caratteri numerici sono stati sostituiti dal testo nel campo *Code*. Se un numero compare più di una volta come in ID=3 e ID=4, anche il testo corrispondente verrà ripetuto. ID=4. Il codice di *Susan McDaniels* conteneva un 6. Dato che non era stato eseguito il mapping del 6 nella tabella *CodeMap*, non viene apportata alcuna modifica. ID=5, il codice di *Dean Smith* conteneva il numero 111. Il relativo mapping è 'elevenone'.

Per ulteriori informazioni su MapSubstring(), vedere questo post di blog in Qlik Community: Mapping... and not the geographical kind (Mappare … ma non nel senso geografico)

4.6 Map … Using

L'istruzione Map … Using può anche essere utilizzata per applicare una mappa a un campo. Tuttavia, funziona in modo un po' diverso rispetto a ApplyMap(). Mentre ApplyMap() gestisce il mapping ogni volta che si rileva il nome di campo, Map … Using gestisce il mapping quando il valore viene memorizzato nel nome di campo nella tabella interna.

Ecco di seguito un esempio. Si supponga di dover caricare più volte il campo *Country* nello script e di voler applicare una mappa a ogni caricamento del campo. È possibile utilizzare la funzione ApplyMap() come illustrato in precedenza in questo tutorial oppure è possibile utilizzare Map … Using.

Se viene utilizzata la funzione Map … Using, la mappa viene applicata al campo quando quest'ultimo viene memorizzato nella tabella interna. Quindi, nell'esempio seguente, la mappa viene applicata al campo *Country* nella tabella Data1, tuttavia non verrà applicata al campo *Country2* nella tabella *Data2*. Ciò si verifica perché l'istruzione Map … Using viene applicata solo ai campi denominati *Country*. Quando il campo *Country2* viene memorizzato nella tabella interna, il suo nome non sarà più *Country*. Per applicare la mappa alla tabella *Country2*, occorrerà utilizzare la funzione ApplyMap().

L'istruzione Unmap termina l'istruzione Map … Using, quindi se si dovesse caricare *Country* dopo l'istruzione Unmap, *CountryMap* non verrebbe applicata.

Procedere come indicato di seguito:

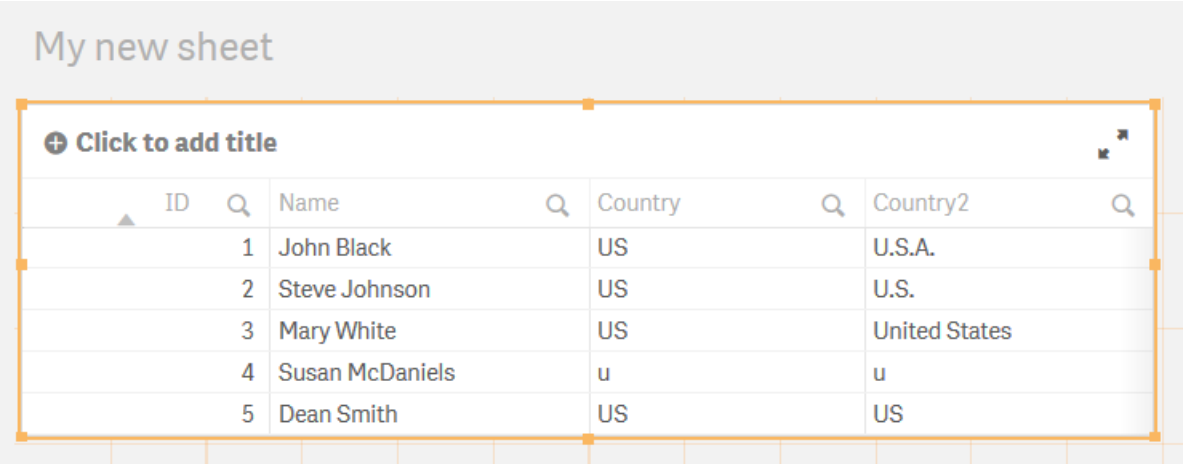
1. Sostituire lo script per la tabella *Data* con quanto segue:

```
Map Country Using CountryMap;
Data1:
    LOAD
    ID,
    Name,
    Country
FROM [lib://AttachedFiles/Data.xlsx]
(ooxml, embedded labels, table is Sheet1);

Data2:
LOAD
    ID,
    Country as Country2
FROM [lib://AttachedFiles/Data.xlsx]
(ooxml, embedded labels, table is Sheet1);
UNMAP;
```

2. Fare clic su **Carica dati**.

   La tabella risultante avrà l'aspetto seguente:

   Tabella in cui sono presenti dati caricati utilizzando la funzione Map … Using



5 Gestione dei dati gerarchici

Le gerarchie sono una parte importante di tutte le soluzioni di Business Intelligence offrendo una descrizione delle dimensioni che contengono naturalmente livelli di granularità diversi. Alcune di queste gerarchie sono semplici e intuitive, mentre altre sono complesse e richiedono impegno per poter essere modellate correttamente.

I membri risultano sempre più dettagliati man mano che dalla parte alta della gerarchia si scende verso quella bassa. Ad esempio, in una dimensione con i livelli Market, Country, State e City, il membro Americas compare nel livello superiore della gerarchia, il membro U.S.A. compare nel secondo livello, il livello California appare nel terzo livello e San Francisco nell'ultimo livello. California è più specifico rispetto a U.S.A., mentre San Francisco è più specifico rispetto a California.

La memorizzazione delle gerarchie in un modello relazionale è un'operazione impegnativa che può essere eseguita in diversi modi. Sono disponibili diversi approcci:

- La gerarchia Orizzontale
- Il modello di elenco Adiacenza
- Il metodo di enumerazione Percorso
- Il modello delle serie Nidificate
- L'elenco Padre

Ai fini di questo tutorial, si creerà un elenco Padre poiché presenta la gerarchia in una forma direttamente utilizzabile in una query. Ulteriori informazioni su altre soluzioni sono disponibili in Qlik Community.

5.1 Prefisso Hierarchy

Il prefisso Hierarchy è un comando di script da inserire davanti all'istruzione LOAD o SELECT per il caricamento di una tabella di nodi adiacenti. L'istruzione LOAD deve avere almeno tre campi: un ID che rappresenta una chiave univoca per il nodo, un riferimento al nodo padre e un nome.

Il prefisso trasformerà una tabella caricata in una tabella di nodi espansi; una tabella con alcune colonne aggiuntive, una per ogni livello della gerarchia.

Procedere come indicato di seguito:

1. Creare una nuova app e assegnarle un nome.
2. Aggiungere una nuova sezione dello script nell'**editor caricamento dati**.
3. Richiamare la sezione *Wine*.
4. Nel menu a destra, in **AttachedFiles**, fare clic su **Seleziona dati**.
5. Caricare e selezionare *Winedistricts.txt*.
6. Nella finestra **Seleziona dati da**, deselezionare i campi *Lbound* e *RBound* in modo che non vengano caricati.
7. Fare clic su **Inserisci script**.
8. Sopra l'istruzione LOAD immettere quanto segue:

```
Hierarchy (NodeID, ParentID, NodeName)
```

Lo script avrà questo aspetto:

```
Hierarchy (NodeID, ParentID, NodeName)
LOAD
    NodeID,
    ParentID,
    NodeName
FROM [lib://AttachedFiles/Winedistricts.txt]
(txt, utf8, embedded labels, delimiter is '\t', msq);
```

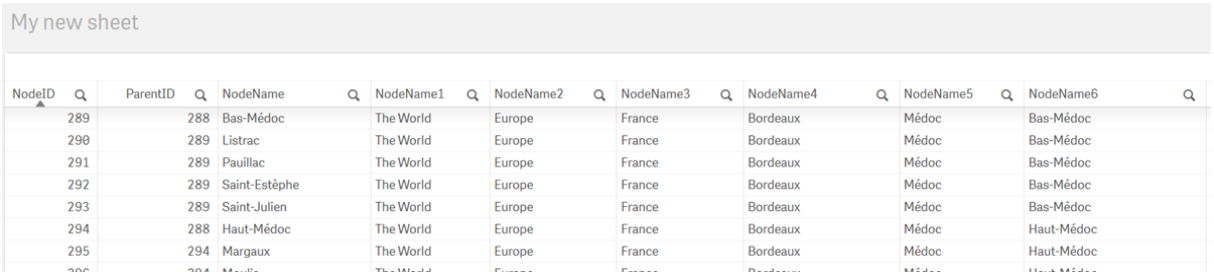
9. Fare clic su **Carica dati**.
10. Utilizzare la sezione **Anteprima** del **sistema di visualizzazione modello dati** per visualizzare la tabella risultante.

    La tabella risultante di nodi espansi presenta esattamente lo stesso numero di record della tabella sorgente: uno per nodo. La tabella di nodi espansi si rivela molto pratica poiché soddisfa alcuni requisiti di analisi della gerarchia in un modello relazionale:

    - Tutti i nomi dei nodi vengono inseriti nella stessa colonna in modo che possa essere utilizzata per le ricerche.
    - Inoltre, ognuno dei livelli dei nodi è stato espanso in un campo. Questi campi possono essere utilizzati nei gruppi di drill-down o come dimensioni nelle tabelle pivot.
    - Inoltre, ognuno dei livelli dei nodi è stato espanso in un campo. Questi campi possono essere utilizzati nei gruppi di drill-down.
    - È possibile fare in modo che il percorso del nodo sia univoco, inserendo tutti i nodi padre nell'ordine corretto.
    - È possibile fare in modo che contenga la profondità del nodo, ad esempio la distanza dal nodo radice.

    La tabella risultante avrà l'aspetto seguente:

    In questa tabella vengono mostrati dati campione caricati utilizzando il prefisso Hierarchy


My new sheet

NodeID	ParentID	NodeName	NodeName1	NodeName2	NodeName3	NodeName4	NodeName5	NodeName6
289	288	Bas-Médoc	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
290	289	Listrac	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
291	289	Pauillac	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
292	289	Saint-Estèphe	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
293	289	Saint-Julien	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
294	288	Haut-Médoc	The World	Europe	France	Bordeaux	Médoc	Haut-Médoc
295	294	Margaux	The World	Europe	France	Bordeaux	Médoc	Haut-Médoc

5.2 Prefisso HierarchyBelongsTo

Allo stesso modo del prefisso Hierarchy, il prefisso HierarchyBelongsTo è un comando di script da inserire davanti all'istruzione LOAD o SELECT per il caricamento di una tabella di nodi adiacenti.

Anche in questo caso, l'istruzione LOAD deve avere almeno tre campi: un ID che rappresenta una chiave univoca per il nodo, un riferimento al nodo padre e un nome. Il prefisso trasformerà la tabella caricata in una tabella padre, vale a dire in una tabella in cui qualsiasi combinazione di nodi padre e nodi decrescenti viene inserita come record separato. Pertanto, è molto facile recuperare tutti i nodi padre o i nodi decrescenti di un nodo specifico.

Procedere come indicato di seguito:

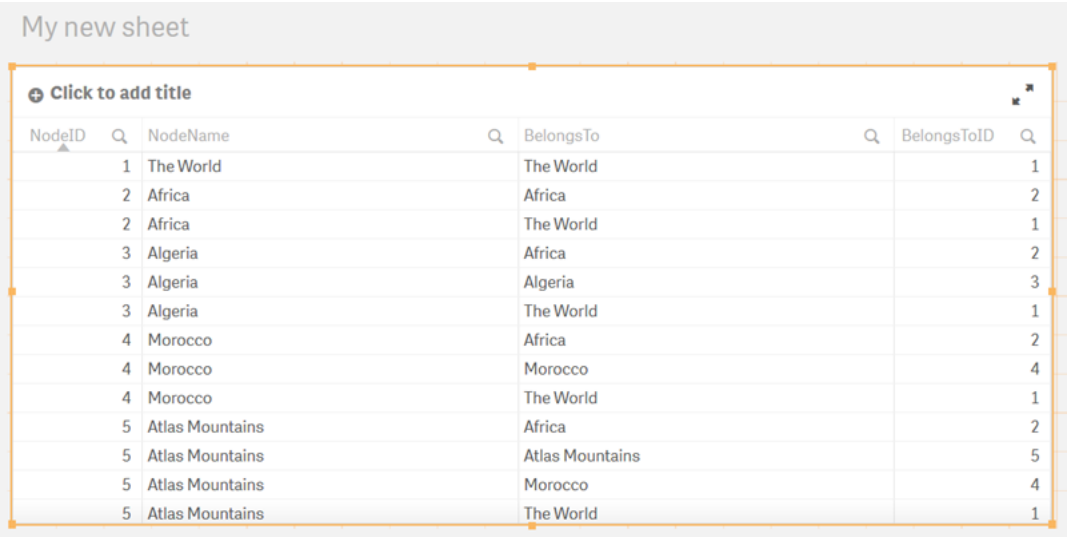
1. Modificare l'istruzione Hierarchy nell'**editor caricamento dati** in modo che riporti quanto segue:
   `HierarchyBelongsTo (NodeID, ParentID, NodeName, BelongsToID, BelongsTo)`
2. Fare clic su **Carica dati**.
3. Utilizzare la sezione **Anteprima** del **sistema di visualizzazione modello dati** per visualizzare la tabella risultante.

   La tabella padre soddisfa alcuni requisiti di analisi della gerarchia in un modello relazionale:

   - Se l'ID del nodo rappresenta i singoli nodi, l'ID padre rappresenta le intere sezioni o le sezioni secondarie della gerarchia.
   - Tutti i nomi dei nodi esistono sia nel ruolo di nodi che nel ruolo di sezioni ed entrambi possono essere utilizzati per le ricerche.
   - È possibile fare in modo che contenga la differenza di profondità tra la profondità del nodo e la profondità del nodo padre, vale a dire la distanza dal nodo radice della sezione secondaria.

   La tabella risultante avrà l'aspetto seguente:

   In questa tabella vengono mostrati dati caricati utilizzando il prefisso HierarchyBelongsTo


My new sheet

Click to add title

NodeID	NodeName	BelongsTo	BelongsToID
1	The World	The World	1
2	Africa	Africa	2
2	Africa	The World	1
3	Algeria	Africa	2
3	Algeria	Algeria	3
3	Algeria	The World	1
4	Morocco	Africa	2
4	Morocco	Morocco	4
4	Morocco	The World	1
5	Atlas Mountains	Africa	2
5	Atlas Mountains	Atlas Mountains	5
5	Atlas Mountains	Morocco	4
5	Atlas Mountains	The World	1

Autorizzazione

Le gerarchie vengono spesso utilizzate per l'autorizzazione, come ad esempio in una gerarchia organizzativa. Ogni responsabile deve disporre del diritto di visualizzare tutto ciò che riguarda il reparto, compresi i reparti secondari. Tuttavia, non deve necessariamente disporre del diritto di accedere alle informazioni di altri reparti.

Esempio in una gerarchia organizzativa

Ciò significa che l'autorizzazione alla visualizzazione di differenti sezioni secondarie dell'organizzazione verrà concessa a persone diverse. La tabella delle autorizzazioni potrebbe avere il seguente aspetto:

Tabella delle autorizzazioni

ACCESS	NTNAME	PERSON	POSITION	PERMISSIONS
USER	ACME\JRL	John	CPO	HR
USER	ACME\CAH	Carol	CEO	CEO
USER	ACME\JER	James	Director Engineering	Engineering
USER	ACME\DBK	Diana	CFO	Finance
USER	ACME\RNL	Bob	COO	Sales
USER	ACME\LFD	Larry	CTO	Product

In questo caso a *Carol* è consentito vedere tutti gli elementi appartenenti a *CEO* e quelli sottostanti; a *Larry* è consentito vedere l'organizzazione *Product* è a *James* è consentito vedere solo l'organizzazione *Engineering*.

Esempio:

Spesso la gerarchia è memorizzata in una tabella di nodi adiacenti. In questo esempio, per risolvere ciò è sufficiente caricare la tabella di nodi adiacenti usando il prefisso HierarchyBelongsTo e assegnare al campo padre il nome *Tree*.

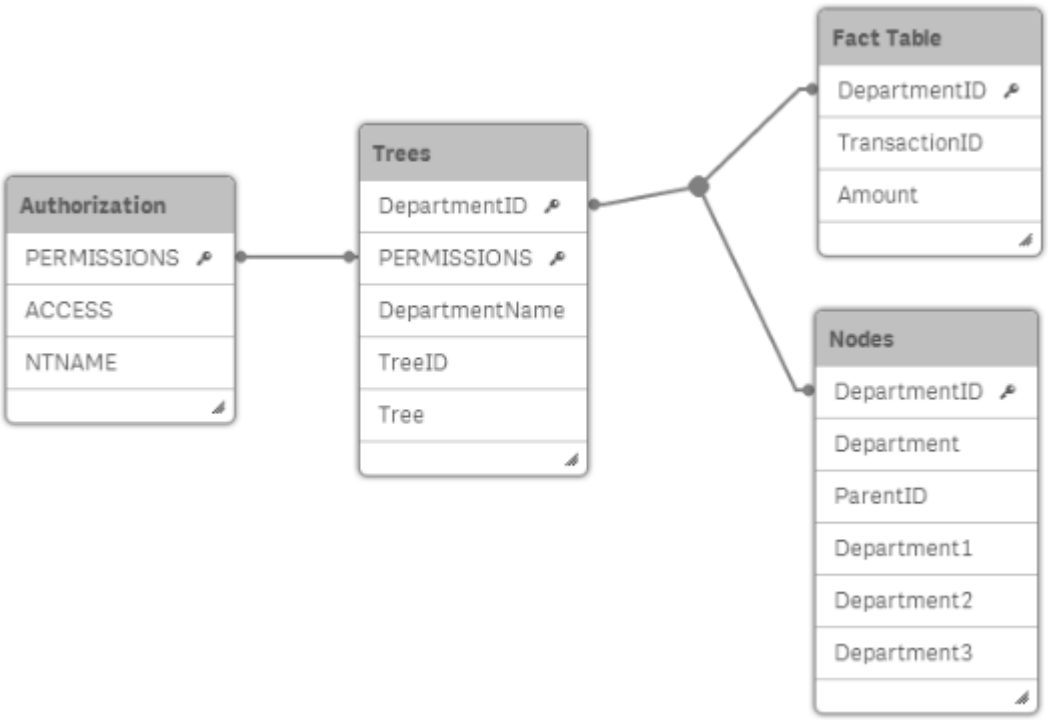
Se si desidera utilizzare Section Access, caricare una copia in maiuscolo di *Tree* e denominare questo nuovo campo *PERMISSIONS*. Infine, è necessario caricare la tabella delle autorizzazioni. Questi ultimi due passaggi possono essere eseguiti utilizzando le seguenti righe di script. Tenere presente che la tabella TempTrees è quella creata dall'istruzione HierarchyBelongsTo.

Questo rappresenta solo un esempio. Non sono disponibili esercizi accompagnatori da completare in Qlik Sense.

```
Trees:
LOAD *,
    Upper(Tree) as PERMISSIONS
    Resident TempTrees;
Drop Table TempTrees;

Section Access;
Authorization:
LOAD   ACCESS,
       NTNAME,
       UPPER(Permissions) as PERMISSIONS
From Organization;
Section Application;
```

Questo esempio produce il seguente modello dati:

Modello dati: tabelle Authorization, Trees, Fact e Nodes



6 File QVD

Un file QVD (QlikView Data) è un file contenente una tabella di dati esportati da Qlik Sense o QlikView. QVD è un formato di Qlik nativo che può essere scritto e letto esclusivamente da Qlik Sense o QlikView. Il formato di file è ottimizzato per la velocità di lettura dei dati da uno script di Qlik Sense rimanendo comunque molto compatto. La lettura di dati da un file QVD è in genere 10-100 volte più veloce rispetto alla lettura da altre sorgenti dati.

I file QVD possono essere letti in due modalità: standard (veloce) e ottimizzata (più veloce). La modalità selezionata viene determinata automaticamente dall'engine degli script di Qlik Sense. La modalità ottimizzata può essere utilizzata solo quando tutti i campi caricati vengono letti senza alcuna trasformazione (formule applicate ai campi), sebbene sia consentito modificare i nomi dei campi. La clausola Where per la decompressione dei record da parte di Qlik Sense disabiliterà anche il caricamento ottimizzato.

Un file QVD conserva esattamente una tabella dati ed è composto da tre parti:

- Un'intestazione XML (con set di caratteri UTF-8) che descrive i campi nella tabella, la presentazione delle informazioni successive e altri metadati.
- Tabelle di simboli in un formato a byte compressi.
- Dati effettivi della tabella in un formato a bit compressi.

I file QVD possono essere utilizzati a fini diversi. È possibile identificare facilmente quattro utilizzi principali. A una qualsiasi situazione, è possibile applicare diversi utilizzi:

- Incremento della velocità di caricamento dei dati

  Eseguendo il buffer in memoria di blocchi di dati di input che non cambiano oppure che cambiano lentamente nei file QVD, l'esecuzione dello script diventa notevolmente più veloce per le serie di dati di grandi dimensioni.

- Incremento della velocità di caricamento dei dati

  Eseguendo il buffer in memoria di blocchi di dati di input che non cambiano oppure che cambiano lentamente nei file QVD, l'esecuzione dello script diventa notevolmente più veloce per le serie di dati di grandi dimensioni.

- Diminuzione del carico sui server di database

  Anche la quantità di dati trasferiti da sorgenti dati esterne può essere ridotta considerevolmente. Ciò consente di ridurre il carico di lavoro sui database esterni e il traffico di rete. Inoltre, quando diversi script Qlik Sense condividono gli stessi dati, è sufficiente caricarli una volta dal database sorgente in un file QVD. Le altre applicazioni possono utilizzare gli stessi dati tramite questo file QVD.

- Consolidamento dei dati provenienti da più applicazioni Qlik Sense

  L'istruzione dello script Binary consente di caricare i dati da una singola applicazione Qlik Sense a un'altra, mentre con i file QVD uno script di Qlik Sense è in grado di combinare i dati provenienti da un numero qualsiasi di applicazioni Qlik Sense. Questa possibilità amplia le prospettive per le applicazioni che consolidano dati simili da unità aziendali diverse e così via.

- Carico incrementale

In molti casi comuni, la funzionalità QVD può essere utilizzata per facilitare il carico incrementale, ad esempio scaricando esclusivamente nuovi record da un database in crescita.

6.1 Creazione di file QVDome dividere

Un file QVD può essere creato in due modi:

- Creazione e denominazione esplicite utilizzando il comando Store nello script Qlik Sense.

  Dichiarare nello script che si desidera esportare una tabella letta in precedenza, o una sua parte, in un file con un nome esplicito nella posizione desiderata.

- Creazione automatica e manutenzione dallo script.

  Facendo precedere all'istruzione load o select il prefisso Buffer, Qlik Sense creerà automaticamente un file QVD che, in determinate condizioni, può essere utilizzato al posto della sorgente dati originale quando si ricaricano i dati.

Non esiste differenza fra i file QVD risultanti relativamente alla velocità di lettura.

Store

Questa istruzione dello script crea un file QVD, CSV o txt con nome esplicito.

Sintassi:

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

L'istruzione può esportare campi solo da una tabella dati. Se occorre esportare i campi da più tabelle, un'unione esplicita deve essere eseguita precedentemente nello script per la creazione della tabella dati da esportare.

I valori di testo sono esportati nel file CSV in formato UTF-8. È possibile specificare un delimitatore, vedere **LOAD**. L'istruzione store associata a un file CSV non supporta l'esportazione BIFF .

Esempi:

```
Store mytable into [lib://AttachedFiles/xyz.qvd];
Store * from mytable into [lib://FolderConnection/xyz.qvd];
Store myfield from mytable into 'lib://FolderConnection/xyz.qvd';
Store myfield as renamedfield, myfield2 as renamedfield2 from mytable into
[lib://AttachedFiles/xyz.qvd];
Store mytable into 'lib://FolderConnection/myfile.txt';
Store * from mytable into 'lib://FolderConnection/myfile.csv';
```

Procedere come indicato di seguito:

1. Aprire l'app *Advanced Scripting Tutorial*.
2. Fare clic sulla sezione dello script *Product*.
3. Aggiungere quanto segue alla fine dello script:

```
Store * from Product into [lib://AttachedFiles/ProductData.qvd](qvd);
```

Lo script avrà questo aspetto:

```
CrossTable(Month, Sales)
LOAD
    Product,
    "Jan 2014",
    "Feb 2014",
    "Mar 2014",
    "Apr 2014",
    "May 2014"
FROM [lib://AttachedFiles/Product.xlsx]
(ooxml, embedded labels, table is Product);

Store * from Product into [lib://AttachedFiles/ProductData.qvd](qvd);
```

4. Fare clic su **Carica dati**.
   Il file *Product.qvd* dovrebbe ora trovarsi nell'elenco dei file.

   Questo file di dati è il risultato dello script **Crosstable** e corrisponde a una tabella con tre colonne, una colonna per ogni categoria (Product, Month, Sales). Questo file di dati può ora essere utilizzato per sostituire l'intera sezione dello script *Product* .

6.2 Lettura dei dati dai file QVD

Un file QVD può essere letto o modificato da Qlik Sense utilizzando i metodi seguenti:

- Caricando un file QVD come sorgente dati esplicita. È possibile fare riferimento ai file QVD con un'istruzione load nello script di Qlik Sense proprio come qualsiasi altro tipo di file di testo (csv, fix, dif, biff e così via).

  Esempi:

  ```
  LOAD * from 'lib://FolderConnection/xyz.qvd' (qvd);
  LOAD fieldname1, fieldname2 from [lib://FolderConnection/xyz.qvd] (qvd);
  LOAD fieldname1 as newfieldname1, fieldname2 as newfieldname2 from
  [lib://AttachedFiles/xyz.qvd](qvd);
  ```

- Caricamento automatico di file QVD inseriti nel buffer. Quando si utilizza il prefisso buffer nelle istruzioni load o select, non è necessaria alcuna istruzione esplicita per la lettura. Qlik Sense determina in quale misura utilizzare i dati dal file QVD o acquisire i dati tramite l'istruzione LOAD o SELECT originale.
- Accesso ai file QVD dallo script. È possibile utilizzare numerose funzioni di script (tutte che iniziano con QVD) per recuperare diverse informazioni sui dati individuati nell'intestazione XML di un file QVD.

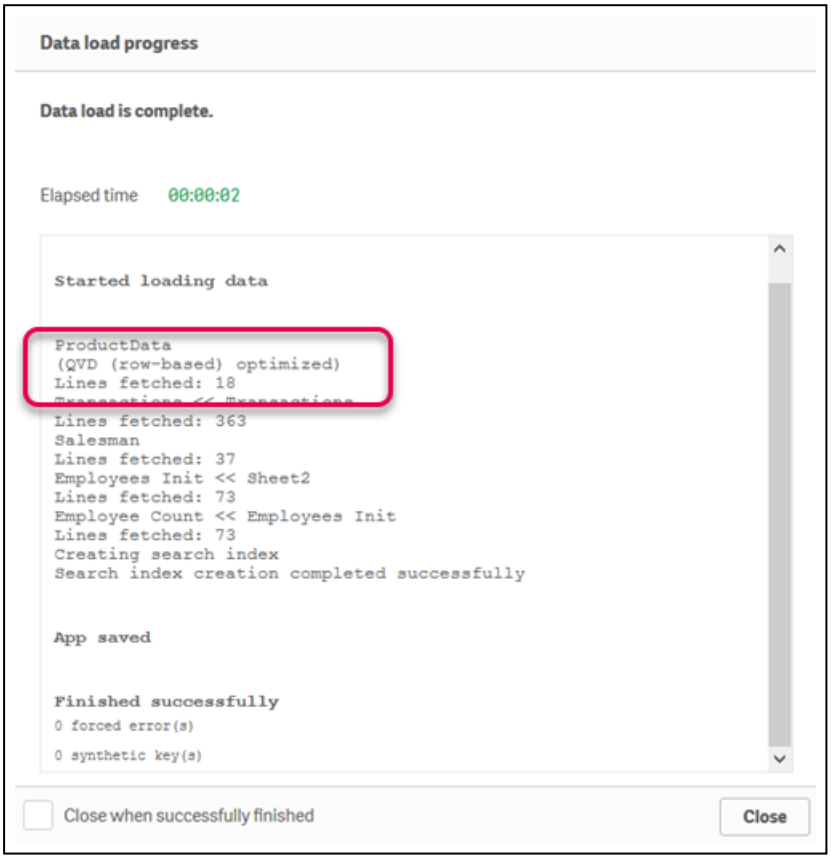
Procedere come indicato di seguito:

1. Impostare come commento l'intero script nella sezione *Product*.
2. Immettere lo script seguente:

```
Load * from [lib://AttachedFiles/ProductData.qvd](qvd);
```

3. Fare clic su **Carica dati**.
   I dati vengono caricati dal file QVD.

   Finestra di avanzamento del caricamento dati



Per informazioni sull'utilizzo di file QVD per caricamenti incrementali, vedere questo post di blog in Qlik Community: Panoramica del caricamento incrementale di Qlik

Buffer

È possibile creare e gestire automaticamente i file QVD mediante il prefisso Buffer. Questo prefisso può essere utilizzato in quasi tutte le istruzioni LOAD e SELECT di uno script e indica che i file QVD vengono utilizzati per memorizzare nella cache/nel buffer il risultato dell'istruzione.

Sintassi:

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

Se nessuna opzione viene utilizzata, il buffer QVD creato alla prima esecuzione dello script verrà utilizzato indefinitamente.

Esempio:

```
Buffer load * from MyTable;
```

stale [after] amount [(days | hours)]

Amount è un numero che specifica l'intervallo di tempo. È possibile utilizzare Decimals. Se omessa, verrà utilizzata l'unità di misura giorni.

In genere, l'opzione stale after viene utilizzata con sorgenti del database i cui dati originali non dispongono di alcun indicatore temporale semplice. Una clausola stale after dichiara semplicemente l'intervallo di tempo a partire dalla creazione del buffer QVD, trascorso il quale non verrà più considerato valido. Prima di quel tempo, il buffer QVD verrà utilizzato come sorgente dei dati e, trascorso l'intervallo specificato, verrà utilizzata la sorgente dati iniziale. Il file del buffer QVD verrà aggiornato automaticamente, quindi avrà inizio un nuovo intervallo.

Esempio:

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

Incremental

L'opzione incremental consente di leggere solo parte di un file sottostante. La dimensione precedente del file viene salvata nell'intestazione XML del file QVD. Queste informazioni risultano particolarmente utili con i file di registro. Tutti i record caricati durante la sessione precedente vengono letti dal file QVD, mentre i nuovi record seguenti vengono letti dalla sorgente originale, quindi viene creato un file QVD aggiornato.

È bene ricordare che l'opzione incremental può essere utilizzata solo con istruzioni LOAD e con file di testo e che non è possibile utilizzare il carico incrementale quando sono stati modificati o eliminati dati precedenti.

Esempio:

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

I buffer QVD vengono normalmente rimossi quando non esistono più riferimenti a essi durante l'intera esecuzione di uno script nell'app che li ha creati oppure quando l'app che li ha creati non esiste più. L'istruzione Store deve essere utilizzata se si desidera memorizzare il contenuto del buffer in un file QVD o CSV.

Procedere come indicato di seguito:

1. Creare una nuova app e assegnarle un nome.
2. Aggiungere una nuova sezione dello script nell'**editor caricamento dati**.
3. Nel menu a destra, in **AttachedFiles**, fare clic su **Seleziona dati**.
4. Caricare e selezionare *Cutlery.xlsx*.
5. Nella finestra **Seleziona dati da**, fare clic su **Inserisci script**.

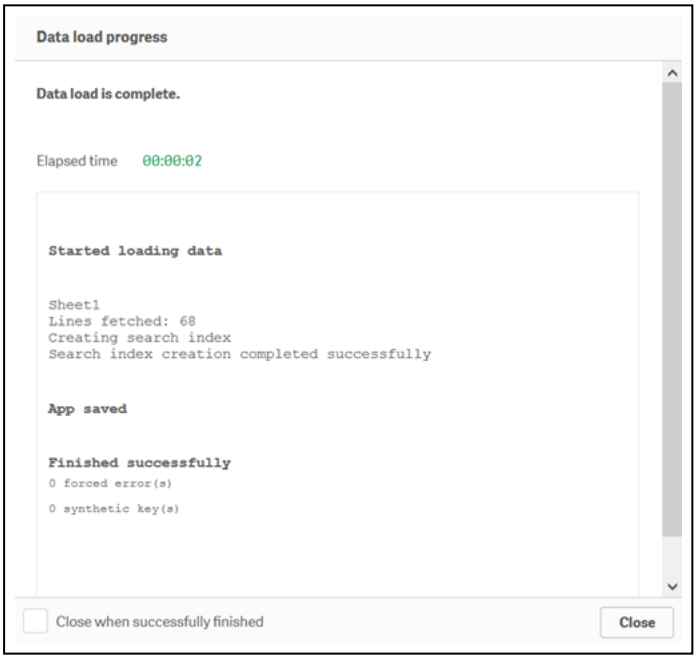
6. Impostare come commenti i campi nell'istruzione LOAD e modificare l'istruzione LOAD nel modo seguente:
   ```
   Buffer LOAD *
   ```

   Lo script avrà questo aspetto:

   ```
   Buffer LOAD *
          //     "date",
          //     item,
          //     quantity
          FROM [lib://AttachedFiles/Cutlery.xlsx]
          (ooxml, embedded labels, table is Sheet1);
   ```

7. Fare clic su **Carica dati**.
   La prima volta che vengono caricati i dati, il caricamento avverrà da *Cutlery.xlsx*.

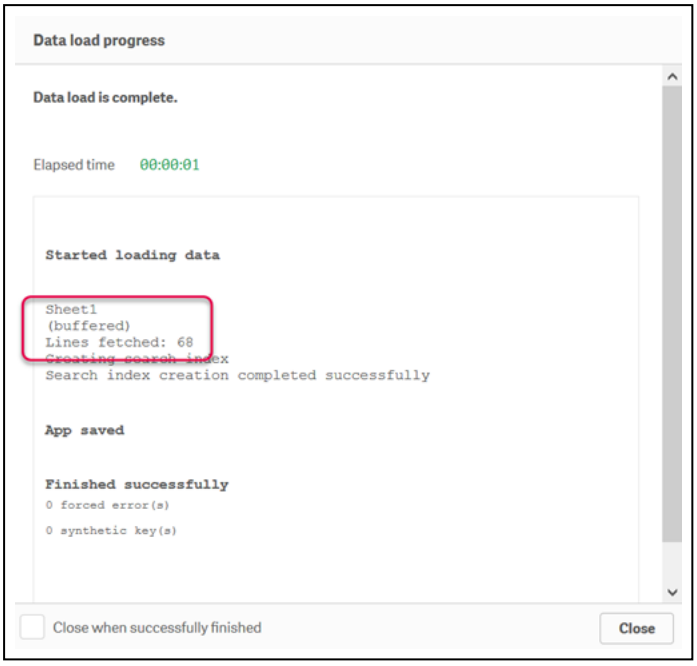
   Finestra di avanzamento del caricamento dati

   L'istruzione Buffer crea anche un file QVD e lo memorizza in Qlik Sense. In una distribuzione Qlik Sense Enterprise on Windows, viene memorizzato in una directory sul server Qlik Sense.

8. Fare di nuovo clic su **Carica dati**.
9. Questa volta i dati vengono caricati dal file QVD creato dall'istruzione Buffer quando i dati sono stati caricati per la prima volta.

Finestra di avanzamento del caricamento dati



6.3 Grazie!

Questo tutorial è stato completato. Ci auguriamo che l'utente abbia acquisito ulteriori informazioni sugli script in Qlik Sense. Visitare il sito Web del programma per ulteriori informazioni sul materiale formativo disponibile.