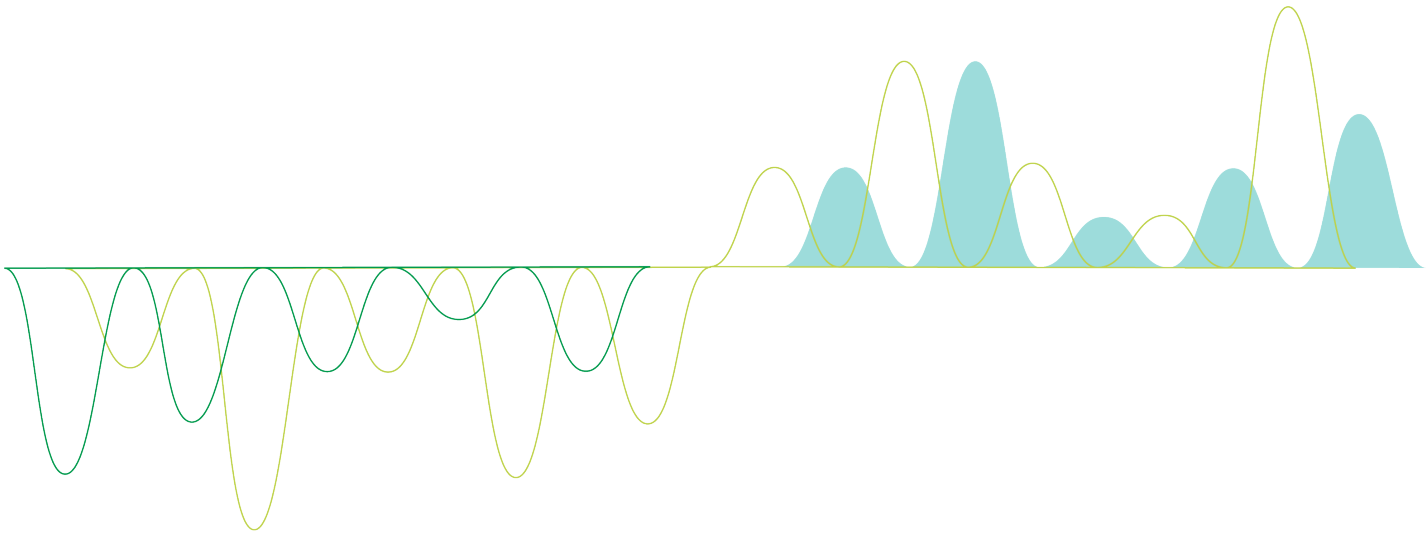
Didacticiel - Création de scripts pour utilisateurs confirmés

Qlik Sense®
November 2023



Contents

1 Bienvenue dans ce didacticiel

Bienvenue dans ce didacticiel, destiné à vous familiariser avec les fonctions de script avancées de Qlik Sense.

Une fois que vous maîtrisez les fonctions de base de création de script, vous pouvez commencer à effectuer des opérations plus sophistiquées sur vos données à mesure que vous les chargez dans Qlik Sense. Ceci peut inclure, par exemple, la transformation de données à l'aide de tableaux croisés, le nettoyage de données, ainsi que la création et le chargement de données à partir de fichiers de données Qlik appelés fichiers QVD.

1.1 Ce que vous allez apprendre

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

1.2 Qui devrait suivre cette formation

Vous devez maîtriser les notions de base de script dans Qlik Sense. C'est-à-dire que vous avez chargé des données et manipulez des données en utilisant des scripts.

Si vous ne l'avez pas déjà fait, il est recommandé de terminer le didacticiel Création de scripts pour débutants.

Vous avez besoin d'un accès à l'éditeur de chargement de données et devez être autorisé à charger des données dans Qlik Sense Enterprise on Windows.

Ces instructions s'appliquent aussi en général à Qlik Sense Cloud Business.

1.3 Contenu du package

Le package zip que vous avez téléchargé contient les fichiers de données suivants dont vous avez besoin pour terminer le didacticiel :

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

Le package contient aussi une copie de l'application *Advanced Scripting Tutorial*. D'autres sections de script dans l'application contiennent les scripts pour les autres applications que vous créez dans ce didacticiel. Vous pouvez charger l'application dans votre hub.

Il est recommandé de créer l'application vous-même comme décrit dans le didacticiel pour optimiser votre apprentissage. De plus, vous devrez télécharger et connecter vos fichiers de données comme décrit dans le didacticiel pour que les chargements de données fonctionnent.

Cependant, si vous rencontrez des problèmes, l'application peut vous aider à les résoudre. Nous avons indiqué quels segments de script sont associés à chaque leçon.

1.4 Leçons dans ce didacticiel

Selon votre expérience avec Qlik Sense, ce didacticiel prend entre 3 et 4 heures pour le terminer. Les rubriques sont conçues pour être terminées en séquence. Cependant, vous pouvez vous arrêter et reprendre à tout moment. Et il n'y a aucun test.

Transformation des données

Utilisation du préfixe Crosstable

Association de tables grâce à Join et Keep

Utilisation des fonctions d'inter-enregistrements Aperçu, Précédent et Existe

Correspondance entre intervalles et chargement itératif

Nettoyage de données

Gestion des données hiérarchiques

Fichiers QVD

1.5 Documentation et ressources supplémentaires

- Qlik tient à votre disposition un large éventail de ressources d'information.
- Une aide en ligne Qlik est disponible.
- Des formations, notamment des cours en ligne gratuits, sont disponibles dans Qlik Continuous Classroom.
- Vous trouverez des forums de discussion, des blogs et bien plus encore dans Qlik Community.

2 Instructions LOAD et SELECT

Vous pouvez charger des données dans Qlik Sense à l'aide des instructions LOAD et SELECT. Chacune de ces instructions génère une table interne. L'instruction LOAD permet de charger des données à partir de fichiers, tandis que l'instruction SELECT procède à partir de bases de données.

Dans ce didacticiel, vous utiliserez des fichiers de données et par conséquent vous utiliserez des instructions LOAD.

Vous pouvez également utiliser un LOAD précédent pour pouvoir manipuler le contenu des données chargées. Par exemple, le renommage des champs doit être effectué dans une instruction LOAD, tandis que l'instruction SELECT n'autorise pas de modification des noms de champ.

Les règles suivantes s'appliquent lors du chargement des données dans Qlik Sense :

- Qlik Sense ne différencie pas les tables générées par les instructions LOAD et SELECT. Autrement dit, si plusieurs tables sont chargées, cela n'a pas d'importance que ce soit l'instruction LOAD, SELECT ou un mélange des deux qui en soit à l'origine.
- L'ordre des champs dans l'instruction ou dans la table d'origine de la base de données n'a pas d'importance pour la logique Qlik Sense.
- Les noms de champs sont sensibles à la casse et utilisés pour établir des associations parmi les tables de données. Par conséquent, il est parfois nécessaire de renommer les champs dans le script de chargement pour atteindre le modèle de données souhaité.

3 Transformation des données

Vous pouvez transformer et manipuler les données dans l'éditeur de chargement de données avant d'utiliser les données dans votre application.

Entre autres avantages, la manipulation de données permet de décider de ne charger qu'un sous-ensemble de données d'un fichier, par exemple quelques colonnes d'une table, afin d'optimiser le traitement des données. Vous pouvez également charger les données plus d'une fois afin de diviser les données brutes en plusieurs nouvelles tables logiques. Il est également possible de charger des données provenant de plusieurs sources et de les fusionner dans une table au sein de Qlik Sense.

Les exercices suivants vous montreront comment charger des données en utilisant le préfixe Crosstable. Vous découvrirez également comment joindre des tables, utiliser des fonctions d'inter-enregistrements telles que Peek et Previous, et charger plusieurs fois la même ligne au moyen de While Load.

3.1 Utilisation du préfixe Crosstable

Les tableaux croisés sont un type de table courant comprenant une matrice de valeurs entre deux listes orthogonales de données d'en-tête. Lorsque vous avez un tableau croisé de données, vous pouvez utiliser le préfixe Crosstable pour transformer les données et créer les champs souhaités.

Préfixe Crosstable

Dans la table *Product* suivante, vous avez une colonne par mois et une ligne par produit.

Table de produit

Produit	Jan 2014	Fév 2014	Mar 2014	Apr 2014	Mai 2014	Juin 2014
A	100	98	100	83	103	82
B	284	279	297	305	294	292
C	50	53	50	54	49	51

Lorsque vous chargez la table, le résultat est une table avec un champ pour *Product* et un champ pour chacun des mois.

Table Product avec le champ Product et un champ pour chacun des mois

Product
Product
Jan 2014
Feb 2014
Mar 2014
Apr 2014
May 2014
Jun 2014

Si vous souhaitez analyser ces données, il est bien plus simple de grouper tous les nombres dans un champ et tous les mois dans l'autre. Dans ce cas, il s'agit d'une table à trois colonnes avec une colonne pour chaque catégorie (*Product, Month, Sales*).

Table Product avec les champs Product, Month et Sales

Product
Product
Month
Sales

Le préfixe Crosstable convertit les données en table comportant une colonne pour le mois (*Month*) et une autre pour les ventes (*Sales*). Autrement dit, il utilise les noms de champ pour les convertir en valeurs de champ.

Procédez comme suit :

1. Créez une nouvelle application et appelez-la *Advanced Scripting Tutorial*.
2. Ajoutez une nouvelle section de script dans l'**éditeur de chargement de données**.
3. Nommez la section *Product*.
4. Sous **AttachedFiles** dans le menu droit, cliquez sur **Sélectionner des données**.
5. Téléchargez, puis sélectionnez *Product.xlsx*.
6. Sélectionnez la table *Product* dans la fenêtre **Sélectionner des données depuis**.

   *Sous **Noms des champs**, assurez-vous que l'option **Noms de champ incorporés** est activée afin d'inclure les noms des champs de table lors du chargement des données.*

7. Cliquez sur **Insérer le script**.

Le script devrait avoir l'aspect suivant :

```
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);
```

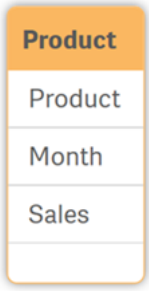
8. Cliquez sur **Charger les données**.
9. Ouvrez le **Visionneur de modèle de données**. Le modèle de données a l'aspect suivant :

   Table Product avec le champ Product et un champ pour chacun des mois

10. Cliquez sur l'onglet *Product* dans l'**éditeur de chargement de données**.
11. Insérez ce qui suit au-dessus de l'instruction LOAD :
    ```
    CrossTable(Month, Sales)
    ```
12. Cliquez sur **Charger les données**.
13. Ouvrez le **Visionneur de modèle de données**. Le modèle de données a l'aspect suivant :

    Table Product avec les champs Product, Month et Sales

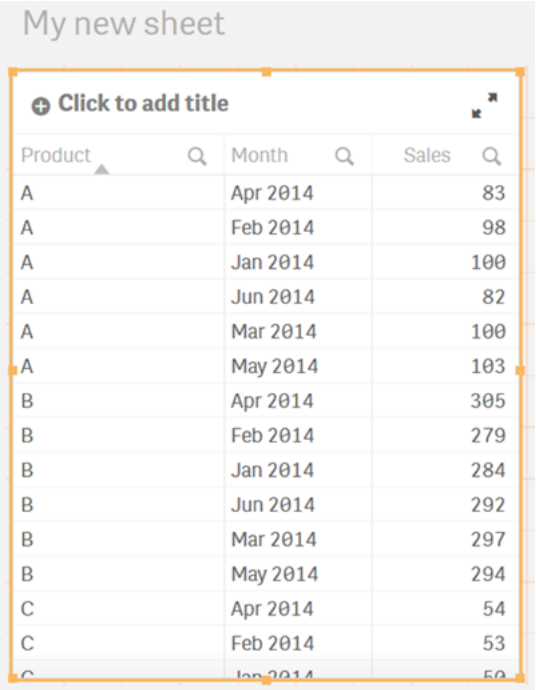
    

Notez que les données d'entrée comprennent généralement une seule colonne comme champ de qualificateur, comme clé interne (*Product* dans l'exemple ci-dessus). Il est toutefois possible d'en utiliser plusieurs. Dans ce cas, tous les champs qualifiants doivent être répertoriés avant les champs

d'attribut dans l'instruction LOAD et le troisième paramètre du préfixe Crosstable doit être utilisé pour définir le nombre de champs qualifiants. Vous ne pouvez pas avoir une instruction LOAD antérieure ou un préfixe devant le mot-clé Crosstable. Cependant, vous pouvez utiliser la concaténation automatique.

Dans une table dans Qlik Sense, vos données ressemblent à ce qui suit :

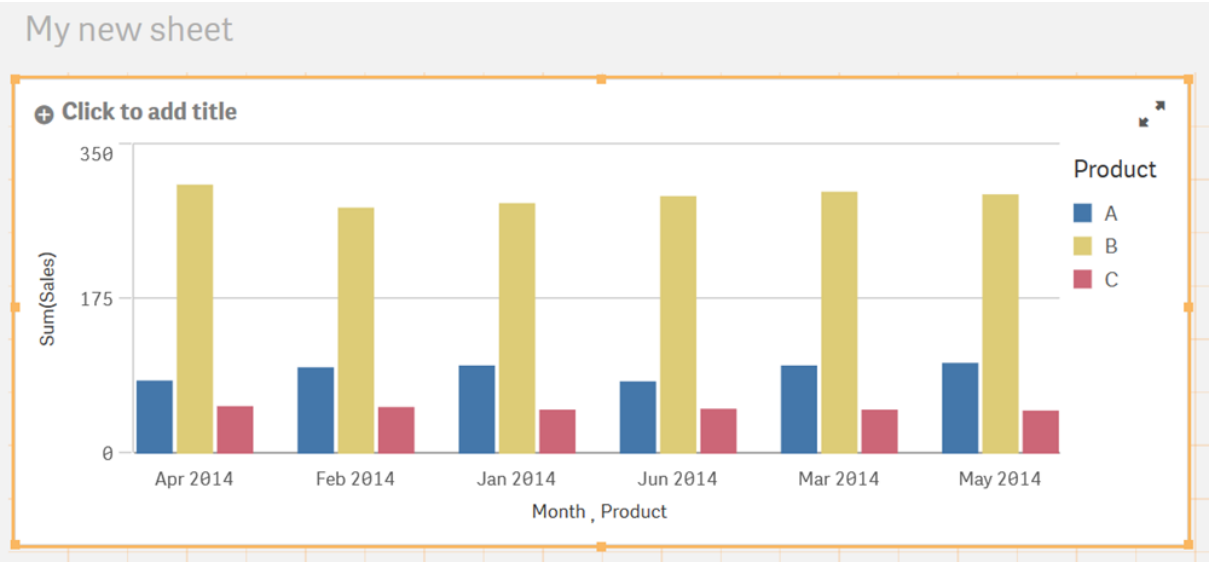
Table affichant des données chargées à l'aide du préfixe Crosstable



Product	Month	Sales
A	Apr 2014	83
A	Feb 2014	98
A	Jan 2014	100
A	Jun 2014	82
A	Mar 2014	100
A	May 2014	103
B	Apr 2014	305
B	Feb 2014	279
B	Jan 2014	284
B	Jun 2014	292
B	Mar 2014	297
B	May 2014	294
C	Apr 2014	54
C	Feb 2014	53

Vous pouvez maintenant, par exemple, créer un graphique à barres en utilisant les données :

graphique à barres affichant des données chargées à l'aide du préfixe Crosstable



Pour en savoir plus sur Crosstable, voir ces articles de blog dans Qlik Community : The Crosstable Load. Les comportements sont abordés dans le contexte de QlikView. Cependant, la logique s'applique également à Qlik Sense.

L'interprétation numérique n'est pas compatible avec les champs d'attribut. En d'autres termes, si les mois sont représentés dans des en-têtes de colonne, ils ne sont pas interprétés automatiquement. La solution consiste à utiliser le préfixe Crosstable en vue de créer une table temporaire et d'exécuter un deuxième passage pour obtenir les interprétations illustrées dans l'exemple suivant :

Notez qu'il s'agit uniquement d'un exemple. Il n'y a aucun exercice correspondant à terminer dans 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;
```

Effacement du cache mémoire

Vous pouvez supprimer les tables que vous créez pour effacer la mémoire cache. Lorsque vous chargez dans une table temporaire, comme à la section précédente, nous vous recommandons de la supprimer lorsque vous n'en avez plus besoin. Par exemple :

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

Vous pouvez aussi abandonner des champs. Par exemple :

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

Comme vous le constatez, les mots-clés TABLE et FIELD peuvent être au singulier ou au pluriel.

3.2 Combinaison de tables grâce à Join et Keep

Une jointure est une opération qui utilise deux tables et les combine en une seule. Les enregistrements de la table résultante sont des combinaisons d'enregistrements des tables d'origine, en général sur la base d'une valeur commune pour un ou plusieurs champs communs aux deux enregistrements contribuant à une combinaison donnée, ce qu'on appelle une jointure naturelle. Dans Qlik Sense, les jointures peuvent être réalisées dans le script, ce qui génère des tables logiques.

Il est possible de joindre des tables figurant déjà dans le script. Dans ce cas, la logique de Qlik Sense ne considère pas les tables séparées, mais plutôt le résultat de la jointure, à savoir une seule table interne. Cette solution s'avère nécessaire dans certaines situations, mais elle présente des inconvénients :

- Les tables chargées deviennent souvent plus volumineuses et Qlik Sense fonctionne plus lentement.
- Certaines informations risquent de se perdre : il se peut que la fréquence (le nombre d'enregistrements) précisée dans la table de départ ne soit plus disponible.

La fonctionnalité Keep, qui a pour effet de réduire l'une ou l'autre table, ou les deux, à l'intersection des données avant que les tables ne soient stockées dans Qlik Sense, a été conçue dans le but de réduire le nombre de cas où l'utilisation de jointures explicites est nécessaire.

Dans cette documentation, le terme jointure désigne habituellement les jointures effectuées avant la création de tables logiques. L'association effectuée après la création des tables internes peut cependant être aussi considérée comme une jointure.

Join

La façon la plus simple de créer une jointure consiste à utiliser le préfixe Join dans le script, qui joint la table interne à une autre table existante ou à la dernière table créée. La jointure est une jointure externe, qui permet de créer toutes les combinaisons possibles de valeurs des deux tables.

Exemple :

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

La table interne résultante comprend les champs a, b, c et d. Le nombre d'enregistrements varie en fonction des valeurs de champ des deux tables.

Les noms des champs de jointure doivent être exactement identiques. Le nombre de champs de jointure est arbitraire. Les tables doivent généralement comporter un ou plusieurs champs en commun. En l'absence de champ commun, la fonction génère le produit cartésien des tables. Que les tables aient tous leurs champs en commun est aussi possible, mais cela n'a habituellement aucun sens. À moins que le nom d'une table déjà chargée soit spécifié dans l'instruction Join, le préfixe Join utilise la dernière table créée. L'ordre des deux instructions n'est donc pas arbitraire.

Utilisation de Join

Le préfixe Join explicite dans le langage de script de Qlik Sense procède à une jointure complète des deux tables. Le résultat en est une seule table. De telles jointures produisent souvent des tables très volumineuses.

Procédez comme suit :

1. Ouvrez l'application *Advanced Scripting Tutorial*.
2. Ajoutez une nouvelle section de script dans l'**éditeur de chargement de données**.

3. Appelez la section *Transactions*.
4. Sous **AttachedFiles** dans le menu droit, cliquez sur **Sélectionner des données**.
5. Téléchargez, puis sélectionnez *Transactions.csv*.

   *Sous **Noms des champs**, assurez-vous que l'option **Noms de champ incorporés** est activée afin d'inclure les noms des champs de table lors du chargement des données.*

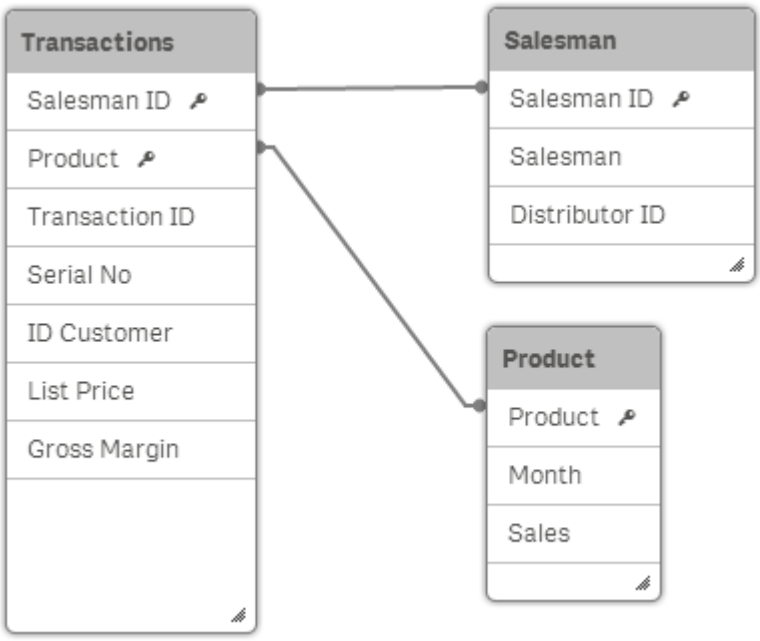
6. Dans la fenêtre **Sélectionner des données depuis**, cliquez sur **Insérer le script**.
7. Téléchargez, puis sélectionnez *Salesman.xlsx*.
8. Dans la fenêtre **Sélectionner des données depuis**, cliquez sur **Insérer le script**.

   Le script devrait avoir l'aspect suivant :

```
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. Cliquez sur **Charger les données**.
10. Ouvrez le **Visionneur de modèle de données**. Le modèle de données a l'aspect suivant :

Modèle de données : Tables Transactions, Salesman et Product



Cependant, l'obtention des tables *Transactions* et *Salesman* séparées ne correspond pas forcément au résultat escompté. Il peut s'avérer préférable de joindre les deux tables.

Procédez comme suit :

1. Pour définir un nom pour la table jointe, ajoutez la ligne suivante au-dessus de la première instruction LOAD :
   ```
   Transactions:
   ```
2. Pour joindre les tables *Transactions* et *Salesman*, ajoutez la ligne suivante au-dessus de la deuxième instruction LOAD :
   ```
   Join(Transactions)
   ```

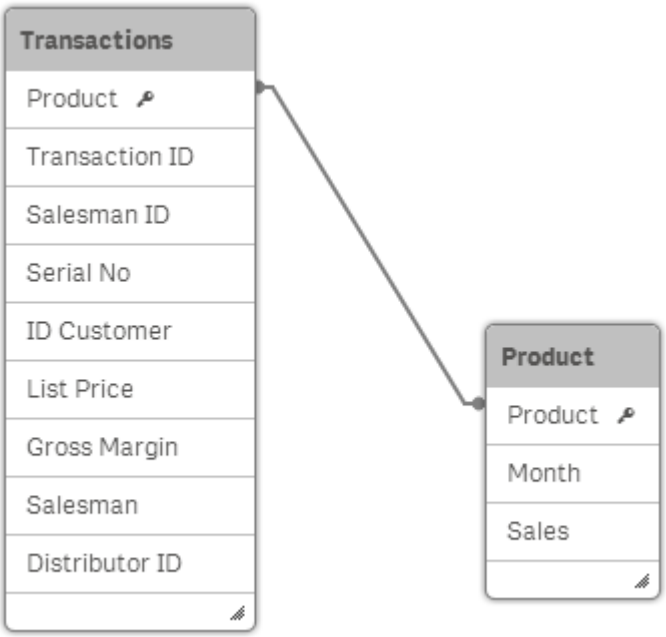
   Le script devrait avoir l'aspect suivant :

   ```
   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. Cliquez sur **Charger les données**.
4. Ouvrez le **Visionneur de modèle de données**. Le modèle de données a l'aspect suivant :

   Modèle de données : Tables Transactions et Product



   Tous les champs des tables *Transactions* et *Salesman* sont à présent réunis dans une seule table *Transactions*.

> *Pour en savoir plus sur le moment de l'utilisation de Join, voir ces articles de blog dans Qlik Community :* To Join or not to Join *(Joindre ou ne pas joindre),* Mapping as an Alternative to Joining *(Mappage comme alternative pour joindre). Les comportements sont abordés dans le contexte de QlikView. Cependant, la logique s'applique également à Qlik Sense.*

Keep

L'une des principales caractéristiques de Qlik Sense est sa capacité à effectuer des associations entre plusieurs tables au lieu de les joindre, ce qui réduit l'espace mémoire utilisé, augmente la vitesse et offre une grande souplesse. La fonctionnalité Keepa été conçue pour réduire le nombre de cas d'utilisation de jointures explicites.

Le préfixe Keep placé entre deux instructions LOAD ou SELECT réduit l'une ou l'autre table, ou les deux, à l'intersection de leurs données avant qu'elles ne soient stockées dans Qlik Sense. Le préfixe Keep doit toujours être précédé d'un des mots clés Inner, Left ou Right. La sélection des enregistrements à partir des tables suit le même principe que la jointure correspondante. Cependant, les deux tables ne sont pas jointes et sont stockées dans Qlik Sense comme deux tables nommées distinctes.

Inner

Dans le script de chargement de données, les préfixes Join et Keep peuvent être précédés du préfixe Inner.

Utilisé avant Join, il indique que la jointure des deux tables doit être une jointure interne. La table obtenue contient ainsi uniquement des combinaisons des deux tables avec un ensemble de données complet des deux côtés.

S'il est utilisé avant Keep, il indique que les deux tables doivent être réduites à leur intersection commune avant d'être stockées dans Qlik Sense.

Exemple :

Ces exemples font appel aux tables source *Table1* et *Table2*.

Notez que ces derniers ne sont que des exemples. Il n'y a aucun exercice correspondant à terminer dans Qlik Sense.

Table 1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Inner Join

Pour commencer, on procède à une jointure Inner Join des tables, ce qui entraîne une table *VTable* ne contenant qu'une seule ligne, le seul enregistrement existant dans les deux tables, les données des deux tables étant combinées.

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

VTable

A	B	C
1	aa	xx

Inner Keep

Si Inner Keep est utilisé à la place, vous disposez toujours de deux tables. Les deux tables sont associées par le champ commun *A*.

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

VTab1

A	B
1	aa

VTab2

A	C
1	xx

Left

Dans le script de chargement de données, les préfixes Join et Keep peuvent être précédés du préfixe left.

Utilisé avant Join, il indique que la jointure des deux tables doit être une jointure gauche. La table résultante contient ainsi uniquement des combinaisons des deux tables avec un ensemble de données complet provenant de la première table.

S'il est utilisé avant Keep, il indique que la seconde table doit être réduite à son intersection commune avec la première table avant d'être stockée dans Qlik Sense.

Exemple :

Ces exemples font appel aux tables source *Table1* et *Table2*.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Pour commencer, on procède à une jointure Left Join des tables, ce qui entraîne une table *VTable* contenant toutes les lignes de la table *Table1* associées aux champs des lignes correspondantes de la table *Table2*.

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

VTable

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

Si Left Keep est utilisé à la place, vous disposez toujours de deux tables. Les deux tables sont associées par le champ commun *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

Les préfixes Join et Keep du langage de script Qlik Sense peuvent être précédés du préfixe right.

Utilisé avant Join, il indique que la jointure des deux tables doit être une jointure droite. La table résultante contient ainsi uniquement des combinaisons des deux tables avec un ensemble de données complet provenant de la seconde table.

S'il est utilisé avant Keep, il indique que la première table doit être réduite à son intersection commune avec la seconde table avant d'être stockée dans Qlik Sense.

Exemple :

Ces exemples font appel aux tables source *Table1* et *Table2*.

Table1

A	**B**
1	aa
2	cc
3	ee

Table2

A	**C**
1	xx
4	yy

Pour commencer, on procède à une jointure Right Join des tables, ce qui entraîne une table *VTable* contenant toutes les lignes de la table *Table2* associées aux champs des lignes correspondantes de la table *Table1*.

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

VTable

A	**B**	**C**
1	aa	xx
4	-	yy

Si Right Keep est utilisé à la place, vous disposez toujours de deux tables. Les deux tables sont associées par le champ commun *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 Utilisation des fonctions d'inter-enregistrements Peek, Previous et Exists

Ces fonctions sont utilisées lorsque l'évaluation de l'enregistrement actif nécessite une valeur provenant d'enregistrements de données déjà chargés.

Dans cette partie du didacticiel, nous allons examiner les fonctions Peek(), Previous() et Exists().

Peek()

Peek() renvoie la valeur d'un champ dans une table pour une ligne qui a déjà été chargée. Il est possible de spécifier le numéro de ligne et la table. Si aucune ligne n'est spécifiée, le dernier enregistrement précédemment chargé sera utilisé.

Syntaxe :

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

La valeur de la ligne doit être un entier. 0 renvoie au premier enregistrement, 1 au deuxième et ainsi de suite. Les nombres négatifs indiquent l'ordre des enregistrements à partir de la fin de la table. -1 renvoie ainsi au dernier enregistrement lu.

Si aucune ligne n'est spécifiée, la fonction utilise -1.

Tablename est une étiquette de table sans les deux-points finaux. Si aucun argument *tablename* n'est spécifié, la table active est utilisée. En cas d'utilisation en dehors de l'instruction **LOAD** ou pour faire référence à une autre table, l'argument *tablename* doit être inclus.

Previous()

Previous() recherche la valeur de l'expression **expr** en utilisant les données de l'enregistrement d'entrée précédent qui n'a pas été ignoré du fait d'une clause **where**. Dans le premier enregistrement d'une table interne, la fonction renvoie NULL.

Syntaxe :

```
Previous(expression)
```

Il est possible d'imbriquer la fonction Previous() afin d'accéder à des enregistrements encore antérieurs. La fonction recherche les données directement dans la source d'entrée, ce qui vous permet de faire aussi référence à des champs qui n'ont pas été chargés dans Qlik Sense, c'est-à-dire même s'ils n'ont pas été stockés dans la base de données associée.

Exists()

Exists() détermine si une valeur de champ donnée a déjà été chargée dans le champ du script de chargement de données. La fonction renvoie TRUE ou FALSE. Elle peut donc être utilisée dans la clause **where** d'une instruction **LOAD** ou d'une instruction **IF**.

Syntaxe :

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

Le champ doit exister dans les données chargées jusqu'ici par le script. *Expression* est une expression qui calcule la valeur de champ à rechercher dans le champ spécifié. Si elle est omise, c'est la valeur de l'enregistrement actif dans le champ spécifié qui est utilisée.

Utilisation de Peek() et Previous()

Dans leur forme la plus simple, Peek() et Previous() permettent d'identifier des valeurs précises dans une table. Voici un échantillon de données dans la table *Employees* que vous chargerez dans cet exercice.

Échantillon de données de la table Employés

Date	Embauche	Cessation d'emploi
1/1/2011	6	0
2/1/2011	4	2
3/1/2011	6	1
4/1/2011	5	2

Pour le moment, cette table collecte seulement les données relatives aux mois, aux embauches et aux cessations d'emploi. Nous allons donc ajouter des champs pour le nombre d'employés (*Employee Count*) et la variance d'employés (*Employee Var*) en utilisant les fonctions Peek() et Previous(), qui nous permettront de voir la différence d'effectifs selon les mois.

Procédez comme suit :

1. Ouvrez l'application *Advanced Scripting Tutorial*.
2. Ajoutez une nouvelle section de script dans l'**éditeur de chargement de données**.
3. Appelez la section *Employees*.
4. Sous **AttachedFiles** dans le menu droit, cliquez sur **Sélectionner des données**.
5. Téléchargez, puis sélectionnez *Employees.xlsx*.

> *Sous Field names, assurez-vous que l'option Embedded field names est sélectionnée afin d'inclure des champs de table lors du chargement des données.*

6. Dans la fenêtre **Sélectionner des données depuis**, cliquez sur **Insérer le script**.

   Le script devrait avoir l'aspect suivant :

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

7. Modifiez le script comme suit :

```
[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);
```

   Les dates indiquées du champ *Date* de la feuille Excel suivent le format MM/JJ/AAAA. Pour vous assurer que les dates sont interprétées correctement via le format provenant des variables système, la fonction Date est appliquée au champ *Date*.

   La fonction Peek() vous permet d'identifier n'importe quelle valeur chargée pour un champ défini. Dans l'expression, nous commençons par examiner le script pour voir si le paramètre rowno() est égal à 1. S'il est égal à 1, aucun champ *Employee Count* ne sera présent. Nous indiquons alors dans le champ la différence des embauches et des cessations d'emploi, soit *Hired* moins *Terminated*.

   Si le paramètre rowno() est supérieur à 1, nous considérons la colonne *Employee Count* du mois précédent et utilisons sa valeur pour l'ajouter à la différence de la colonne *Hired* de ce mois moins la colonne *Terminated*.

   Vous observerez également que nous utilisons (-1) dans la fonction Peek(). Cela permet à Qlik Sense de considérer l'enregistrement situé au-dessus de l'enregistrement actif. Si (-1) n'est pas spécifié, Qlik Sense suppose que vous souhaitez examiner l'enregistrement précédent.

8. Ajoutez ce qui suit à la fin du 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 fonction Previous() vous permet d'identifier la dernière valeur chargée pour un champ défini. Dans cette expression, nous commençons par examiner le script pour voir si le paramètre rowno() est égal à 1. S'il est égal à 1, nous savons qu'il n'y aura pas de paramètre Employee Var, car il n'existe aucun enregistrement pour le nombre d'employés *Employee Count* du mois dernier. Nous entrons donc 0 pour la valeur.

Si le paramètre rowno() est supérieur à 1, nous savons qu'un paramètre *Employee Var* sera présent. Dans ce cas, nous devrons examiner la colonne *Employee Count* du mois dernier et soustraire cette valeur de la colonne *Employee Count* du mois actuel afin d'obtenir la valeur du champ *Employee Var*.

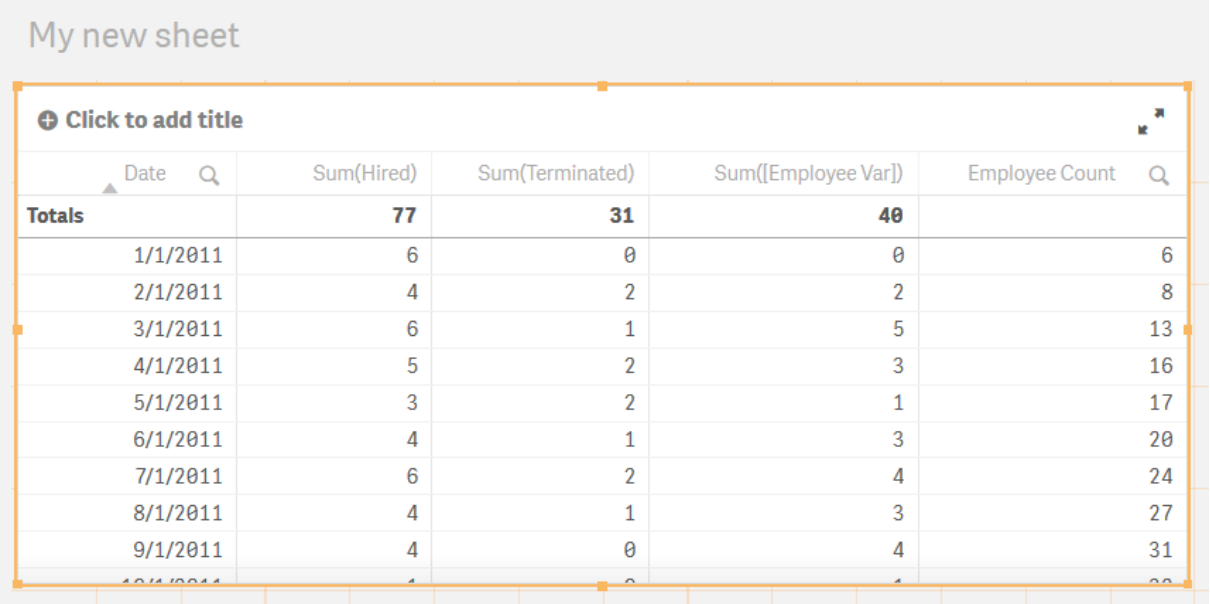
Le script devrait avoir l'aspect suivant :

```
[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. Cliquez sur **Charger les données**.

   Dans une nouvelle feuille de l'aperçu de l'application, créez une table en utilisant *Date, Hired, Terminated, Employee Count* et *Employee Var* comme colonnes de la table. La table résultante doit avoir l'aspect suivant :

Table suivant l'utilisation de Peek et Previous dans le 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() et Previous() vous permettent de cibler des lignes définies dans une table. La principale différence entre les deux réside dans le fait que la fonction Peek() permet à l'utilisateur d'examiner un champ qui n'était pas déjà chargé dans le script tandis que la fonction Previous() peut uniquement considérer un champ déjà chargé. La fonction Previous() fonctionne au niveau de l'entrée de l'instruction LOAD tandis que la fonction Peek() fonctionne au niveau de la sortie de l'instruction LOAD. (La différence entre ces fonctions rappelle celle des fonctions RecNo() et RowNo()). Par conséquent, les deux fonctions se comportent différemment en présence d'une clause Where.

Ainsi, il est préférable d'utiliser la fonction Previous() dans le cas où vous devriez afficher la valeur actuelle par rapport à la valeur précédente. Dans notre exemple, nous avons calculé la variance des employés d'un mois sur l'autre.

L'utilisation de la fonction Peek() est préconisée dans les cas où vous ciblez un champ qui n'a pas encore été chargé dans la table ou lorsque vous avez besoin de cibler une ligne précise. Ce cas est illustré dans l'exemple où nous avons calculé la valeur de la colonne *Employee Count* en examinant la colonne *Employee Count* du mois précédent, puis en lui ajoutant la différence entre le nombre des embauches et le nombre des cessations d'emploi du mois actuel. Rappelez-vous que la colonne *Employee Count* n'était pas un champ dans le fichier d'origine.

Pour en savoir plus sur Peek() et Previous(), voir ces articles de blog dans Qlik Community : Peek() vs Previous() – When to Use Each. Les comportements sont abordés dans le contexte de QlikView. Cependant, la logique s'applique également à Qlik Sense.

Utilisation de Exists()

La fonction Exists() s'utilise souvent avec la clause Where dans le script afin de permettre de charger des données si des données connexes ont déjà été chargées dans le modèle de données.

Dans l'exemple suivant, nous utilisons aussi la fonction Dual() pour attribuer des valeurs numériques à des chaînes.

Procédez comme suit :

1. Créez une application et nommez-la.
2. Ajoutez une nouvelle section de script dans l'**éditeur de chargement de données**.
3. Appelez la section *People*.
4. Saisissez les lignes de script suivantes :

```
//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;
```

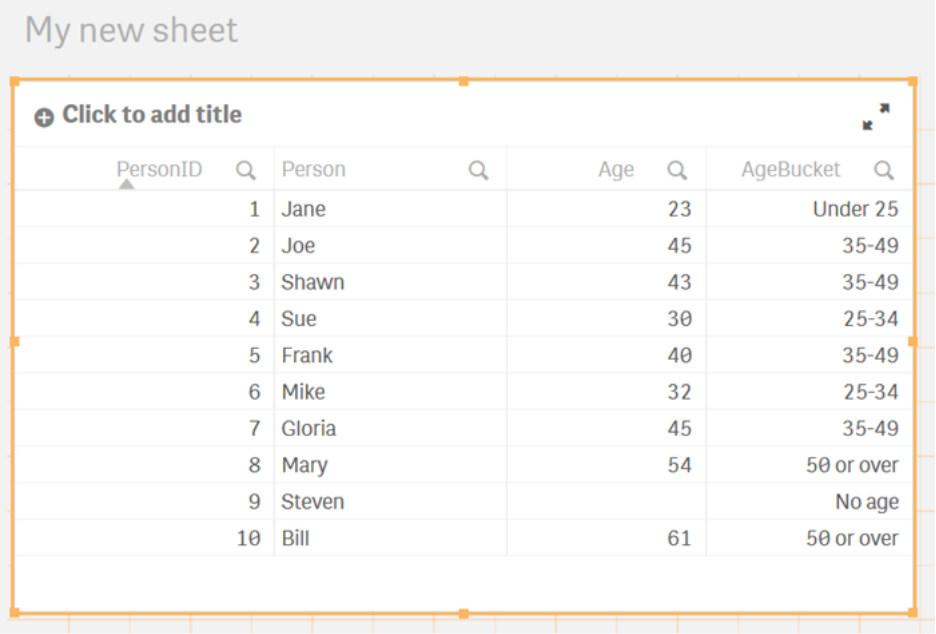
5. Cliquez sur **Charger les données**.

   Dans le script, les champs *Age* et *AgeBucket* sont chargés uniquement si le champ *PersonID* a déjà été chargé dans le modèle de données.

   Dans la table *AgeTemp*, vous observerez que les âges 11 et 12 sont répertoriés pour le champ *PersonID*, mais comme ces ID n'ont pas été chargés dans le modèle de données (dans la table *People*), ils sont exclus par la clause Where Exists(PersonID). Il est également possible d'écrire cette clause de la manière suivante : Where Exists(PersonID, PersonID).

   Le résultat du script ressemble à cela :

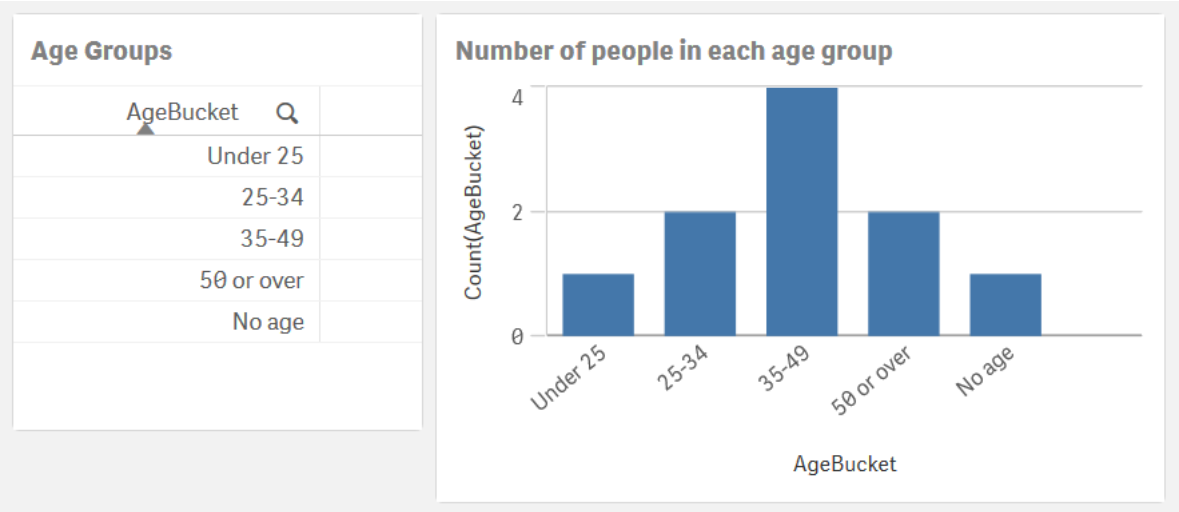
   Table suivant l'utilisation de Exists dans le script



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

Si aucun des ID *PersonID* figurant dans la table *AgeTemp* n'avait été chargé dans le modèle de données, alors les champs *Age* et *AgeBucket* n'auraient pas été joints à la table *People*. L'utilisation de la fonction Exists() permet d'empêcher la présence de données ou d'enregistrements orphelins dans le modèle de données, c.-à-d. de champs *Age* et *AgeBucket* sans aucune personne associée.

6. Créez une feuille et nommez-la.
7. Ouvrez la nouvelle feuille, puis cliquez sur **Éditer la feuille**.
8. Ajoutez à la feuille une table standard comportant la dimension *AgeBucket* et nommez la visualisation *Age Groups*.
9. Ajoutez un graphique à barres à la feuille avec la dimension *AgeBucket* et la mesure *Count ([AgeBucket])*. Nommez la visualisation *Number of people in each age group*.
10. Configurez les propriétés de la table et du graphique à barres à votre convenance, puis cliquez sur **Terminer**.
    Votre feuille devrait à présent ressembler à cela :
    Feuille avec groupements par âge



La fonction Dual() s'avère pratique dans les scripts ou les expressions de graphique, lorsqu'il est nécessaire d'attribuer une valeur numérique à une chaîne.

Dans le script ci-dessus, vous disposez d'une application qui charge les âges. Vous avez décidé de placer ces âges dans des tranches de manière à créer ensuite des visualisations basées sur les tranches d'âges par rapport aux âges réels. Une tranche est définie pour les moins de 25 ans, une autre pour les 25-35 ans et ainsi de suite. Avec la fonction Dual(), vous pouvez attribuer une valeur numérique aux tranches d'âge. Cette valeur permettra ensuite de trier les tranches d'âge dans une liste de sélection ou un graphique. Ainsi, comme sur la feuille de l'application, la mention "No age" (Pas d'âge) figure à la fin de la liste.

Pour en savoir plus sur Exists() et Dual(), voir cet article de blog dans Qlik Community : Dual & Exists – Useful Functions

3.4 Correspondance entre intervalles et chargement itératif

Le préfixe Intervalmatch d'une instruction LOAD ou SELECT permet de lier des valeurs numériques discrètes à un ou plusieurs intervalles numériques. Il s'agit d'une fonction très puissante qui peut s'utiliser, par exemple, dans les environnements de production.

Utilisation du préfixe IntervalMatch()

La correspondance d'intervalle la plus simple est la suivante : vous disposez d'une liste de nombres ou de dates (d'événements) dans une table et d'une liste d'intervalles dans une seconde table. L'objectif est de lier ces deux tables. En général, il s'agit d'une relation de plusieurs à plusieurs. Autrement dit, un intervalle peut comporter plusieurs dates s'y rapportant et une date peut faire partie de plusieurs intervalles. Pour résoudre ce cas, vous devez créer une table de correspondances entre les deux tables initiales. Il existe plusieurs méthodes pour y parvenir.

La méthode la plus simple pour résoudre ce problème dans Qlik Sense est celle qui consiste à placer le préfixe IntervalMatch() devant une instruction LOAD ou SELECT. L'instruction LOAD/SELECT doit seulement contenir deux champs, From et To, qui définissent les intervalles. Le préfixe IntervalMatch() génère ensuite toutes les combinaisons possibles entre les intervalles chargés et un champ numérique précédemment chargé, spécifié sous forme de paramètre du préfixe.

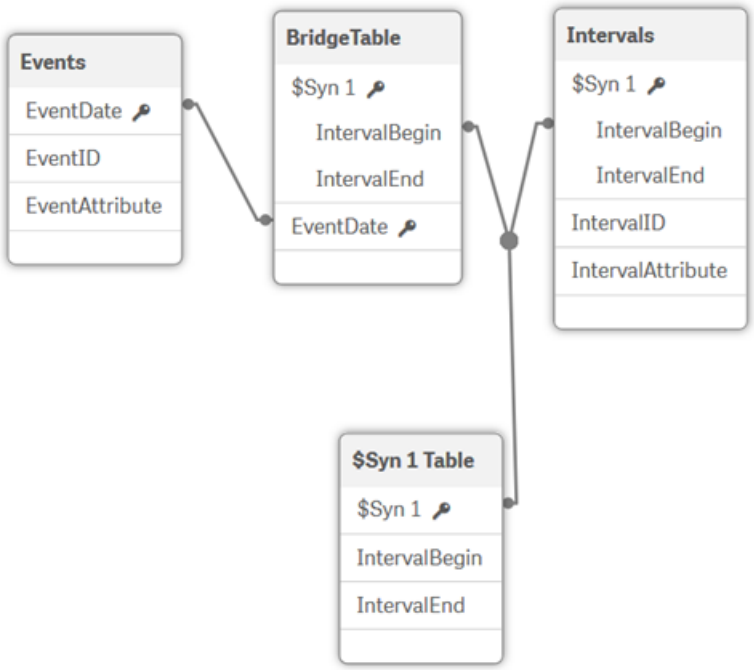
Procédez comme suit :

1. Créez une application et nommez-la.
2. Ajoutez une nouvelle section de script dans l'**éditeur de chargement de données**.
3. Appelez les sections *Events*.
4. Sous **AttachedFiles** dans le menu droit, cliquez sur **Sélectionner des données**.
5. Téléchargez, puis sélectionnez *Events.txt*.
6. Dans la fenêtre **Sélectionner des données depuis**, cliquez sur **Insérer le script**.
7. Téléchargez, puis sélectionnez *Intervals.txt*.
8. Dans la fenêtre **Sélectionner des données depuis**, cliquez sur **Insérer le script**.
9. Dans le script, nommez la première table *Events* et nommez la deuxième table *Intervals*.
10. À la fin du script, ajoutez un préfixe IntervalMatch pour créer une troisième table chargée de relier les deux premières :
    ```
    BridgeTable:
    IntervalMatch (EventDate)
    LOAD distinct IntervalBegin, IntervalEnd
    Resident Intervals;
    ```
11. Le script devrait avoir l'aspect suivant :
    ```
    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. Cliquez sur **Charger les données**.
13. Ouvrez le **Visionneur de modèle de données**. Le modèle de données a l'aspect suivant :

    Modèle de données : Tables Events, BridgeTable, Intervals et $Syn1



Le modèle de données contient une clé composite (les champs *IntervalBegin* et *IntervalEnd*), qui se manifestera sous la forme d'une clé synthétique Qlik Sense.

Les tables de base sont les suivantes :

- La table *Events* qui contient exactement un enregistrement par événement.
- La table *Intervals* qui contient exactement un enregistrement par intervalle.

- La table de correspondances qui contient exactement un enregistrement par combinaison d'événement et d'intervalle, et qui assure la liaison des deux tables précédentes.

Sachez qu'un événement peut faire partie de plusieurs intervalles si ceux-ci se chevauchent. De même, plusieurs événements peuvent appartenir à un même intervalle.

Il s'agit d'un modèle de données optimal, dans le sens où il est normalisé et compact. Les tables *Events* et *Intervals* demeurent toutes deux inchangées et contiennent le nombre d'enregistrements initial. Tous les calculs de Qlik Sense appliqués à ces tables, par exemple, Count(EventID), fonctionneront et aboutiront aux bons résultats.

Pour en savoir plus sur IntervalMatch(), voir ces articles de blog dans Qlik Community : Using IntervalMatch()

Utilisation d'une boucle While et du chargement itératif IterNo()

Vous pouvez parvenir pratiquement au même résultat dans la table de correspondance en utilisant une boucle While et IterNo() qui crée des valeurs énumérables entre les limites supérieure et inférieure de l'intervalle.

Il est possible de créer une boucle au sein de l'instruction LOAD au moyen de la clause While. Par exemple :

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

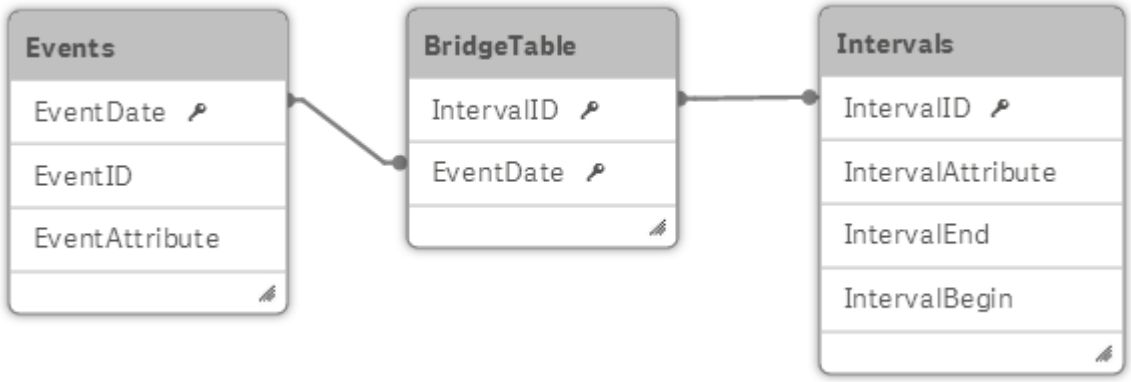
Une telle instruction LOAD créera une boucle sur chaque enregistrement d'entrée et chargera le résultat en permanence tant que l'expression indiquée dans la clause While est vraie. La fonction IterNo() renvoie 1 dans la première itération, 2 dans la deuxième, etc.

Vous disposez d'un identifiant principal pour les intervalles, intitulé IntervalID. Par conséquent, la seule différence notable dans le script est le mode de création de la table de correspondances :

Procédez comme suit :

1. Remplacez les instructions Bridgetable existantes par le script suivant :
   ```
   BridgeTable:
   LOAD distinct * Where Exists(EventDate);
   LOAD IntervalBegin + IterNo() - 1 as EventDate, IntervalID
       Resident Intervals
       While IntervalBegin + IterNo() - 1 <= IntervalEnd;
   ```
2. Cliquez sur **Charger les données**.
3. Ouvrez le **Visionneur de modèle de données**. Le modèle de données a l'aspect suivant :

Modèle de données : Tables Events, BridgeTable et Intervals



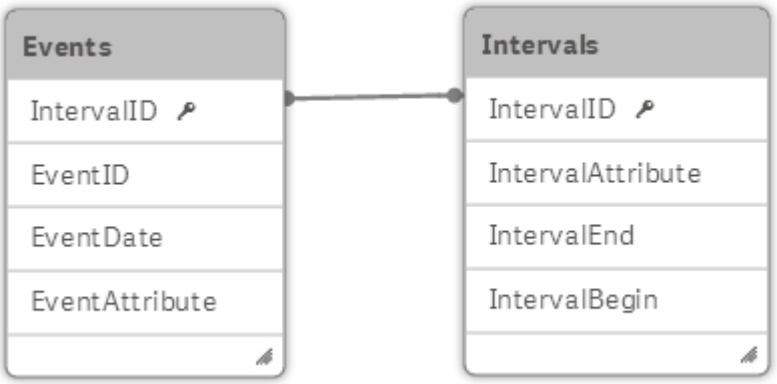
En général, la solution aux trois tables est la meilleure, car elle permet de définir une relation plusieurs à plusieurs entre les intervalles et les événements. Cependant, il arrive souvent que vous sachiez qu'un événement ne peut faire partie que d'un seul intervalle. Dans ce cas, la table de correspondances est vraiment superflue. Vous pouvez stocker directement *IntervalID* dans la table d'événements. Pour ce faire, il existe plusieurs méthodes, mais la plus pertinente consiste à joindre Bridgetable à la table *Events*.

4. Ajoutez le script suivant à la fin de votre script :

```
Join (Events)
LOAD EventDate, IntervalID
Resident BridgeTable;

Drop Table BridgeTable;
```

5. Cliquez sur **Charger les données**.
6. Ouvrez le **Visionneur de modèle de données**. Le modèle de données a l'aspect suivant :

Modèle de données : Tables Events et Intervals



Intervalles ouverts et fermés

L'ouverture et la fermeture d'un intervalle sont déterminées par l'état des extrémités, c.-à-d. si celles-ci sont incluses ou pas dans l'intervalle.

- Si les extrémités font partie de l'intervalle, il s'agit d'un intervalle fermé :
  $[a,b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$
- Si les extrémités ne font pas partie de l'intervalle, il s'agit d'un intervalle ouvert :
  $]a,b[= \{x \in \mathbb{R} \mid a < x < b\}$
- Si une seule extrémité fait partie de l'intervalle, il s'agit d'un intervalle semi-ouvert :
  $[a,b[= \{x \in \mathbb{R} \mid a \leq x < b\}$

Si vous rencontrez un cas où les intervalles se chevauchent et qu'un nombre peut faire partie de plus d'un intervalle, il est généralement préconisé d'utiliser des intervalles fermés.

Cependant, dans certains cas, vous ne souhaitez pas d'intervalles qui se chevauchent, mais un nombre faisant partie d'un seul intervalle. Un problème se pose alors si un point correspond à la fois à la fin d'un intervalle et au début du suivant. Un nombre de ce type est attribué aux deux intervalles. Dans ce cas, il est préférable d'utiliser des intervalles semi-ouverts.

Une solution pratique à ce problème consiste à soustraire une très petite quantité de la valeur de fin de tous les intervalles, afin de créer des intervalles fermés qui ne se chevauchent pas. S'il s'agit de dates, le moyen le plus simple consiste à employer la fonction DayEnd(), qui renvoie la dernière milliseconde du jour :

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

Vous pouvez aussi choisir de soustraire une petite quantité en procédant manuellement. Dans ce cas, veillez à ce que la quantité soustraite ne soit pas trop petite, car l'opération est arrondie à 52 chiffres binaires significatifs (14 chiffres décimaux). Si vous utilisez une quantité trop petite, la différence ne sera pas significative et vous retomberez sur le nombre d'origine.

4 Nettoyage de données

Il arrive que les données sources chargées dans Qlik Sense ne correspondent pas forcément à la manière dont vous souhaitez les utiliser dans l'application Qlik Sense. C'est pourquoi Qlik Sense propose une pléiade de fonctions et d'instructions qui permettent de transformer les données dans un format exploitable.

Vous avez ainsi la possibilité de recourir au mappage dans un script Qlik Sense afin de remplacer ou de modifier des noms ou des valeurs de champ lors de l'exécution du script. Le mappage peut donc s'utiliser pour nettoyer ou uniformiser des données, ou encore pour remplacer tout ou partie d'une valeur de champ.

Lorsque vous chargez des données provenant de différentes tables, les valeurs de champ renvoyant au même élément ne sont pas toujours nommées de façon cohérente. Comme ces incohérences empêchent les associations, elles posent problème. Pour remédier à ce problème de manière simple, il suffit de créer une table de mappage destinée à comparer les valeurs des champs.

4.1 Tables de mappage

Les tables chargées via Mapping load ou Mapping select sont traitées différemment des autres tables. Elles sont stockées dans une zone distincte de la mémoire et sont utilisées uniquement comme tables de mappage au moment de l'exécution du script. Une fois le script exécuté, ces tables sont automatiquement retirées.

Règles :

- Une table de mappage doit comprendre deux colonnes, la première contenant les valeurs de comparaison et la seconde les valeurs de mappage voulues.
- Les deux colonnes doivent être nommées, mais les noms ne sont pas importants en eux-mêmes. Ils sont sans rapport avec les noms des champs dans les tables internes normales.

4.2 Fonctions et instructions Mapping

Les fonctions et instructions de mappage suivantes sont évoquées au cours de ce didacticiel :

• Préfixe Mapping

• ApplyMap()

• MapSubstring()

• Instruction Map … Using

• Instruction Unmap

4.3 Préfixe Mapping

Le préfixe Mapping sert à créer une table de mappage dans un script. La table de mappage peut ensuite s'utiliser avec la fonction ApplyMap(), la fonction MapSubstring() ou l'instruction Map … Using.

Procédez comme suit :

1. Créez une application et nommez-la.
2. Ajoutez une nouvelle section de script dans l'**éditeur de chargement de données**.
3. Appelez la section *Countries*.
4. Saisissez les lignes de script suivantes :

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

La table *CountryMap* stocke deux colonnes : *Country* et *NewCountry*. La colonne *Country* contient les différentes façons dont le pays a été spécifié dans le champ *Country*. La colonne *NewCountry* contient le mode de mappage des valeurs. Cette table de mappage permettra de stocker des valeurs de pays *US* cohérentes dans le champ *Country*. Par exemple, si *U.S.A.* est stocké dans le champ *Country*, mappez-le pour qu'il corresponde à *US*.

4.4 ApplyMap() fonction

Utilisez ApplyMap() pour remplacer des données dans un champ en fonction d'une table de mappage créée précédemment. Veillez à charger la table de mappage avant d'utiliser la fonction ApplyMap(). Les données dans la table *Data.xlsx* que vous chargerez ressemblera à ceci :

Table de données

ID	Nom	Country	Code
1	John Black	U.S.A.	SDFGBS1DI
2	Steve Johnson	U.S.	2ABC
3	Mary White	États-Unis	DJY3DFE34
4	Susan McDaniels	u	DEF5556
5	Dean Smith	US	KSD111DKFJ1

Vous noterez que le pays est indiqué sous différentes formes. Afin d'uniformiser le champ du pays, la table de mappage est chargée, puis la fonction **ApplyMap()** est appliquée.

Procédez comme suit :

1. Sous le script que vous saisissez ci-dessus, sélectionnez et chargez *Data.xlsx*, puis insérez le script.
2. Insérez ce qui suit au-dessus de l'instruction LOAD que vous venez de créer :

```
Data:
```

Le script devrait avoir l'aspect suivant :

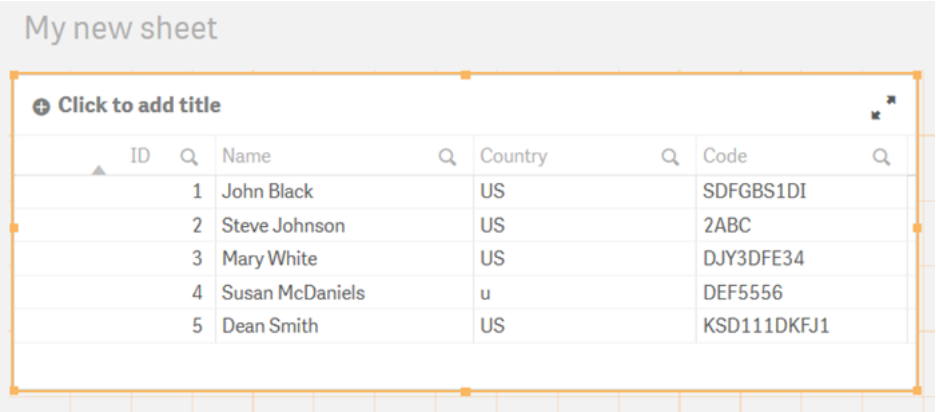
```
CountryMap:
MAPPING LOAD * INLINE [
    Country, NewCountry
    U.S.A., US
    U.S., US
    United States, US
    United States of America, US
];

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

3. Modifiez la ligne contenant `Country`, de la manière suivante :
   `ApplyMap('CountryMap', Country) as Country,`

   Le premier paramètre de la fonction ApplyMap() contient le nom du mappage placé entre guillemets simples. Le deuxième paramètre correspond au champ dont les données doivent être remplacées.

4. Cliquez sur **Charger les données**.
   La table résultante a l'aspect suivant :

   Table affichant des données chargées à l'aide de la fonction ApplyMap()



   Les différentes orthographes du pays *United States* ont toutes été unifiées et remplacées par *US*. Un seul enregistrement était mal orthographié. La fonction ApplyMap() n'a donc pas modifié la valeur de ce champ. La fonction ApplyMap() permet d'utiliser le troisième paramètre pour ajouter une expression par défaut si la table de mappage ne dispose pas de valeur correspondante.

5. Ajoutez `'US'` comme troisième paramètre de la fonction ApplyMap() afin de traiter les cas où le pays peut avoir été mal orthographié :
   `ApplyMap('CountryMap', Country, 'US') as Country,`

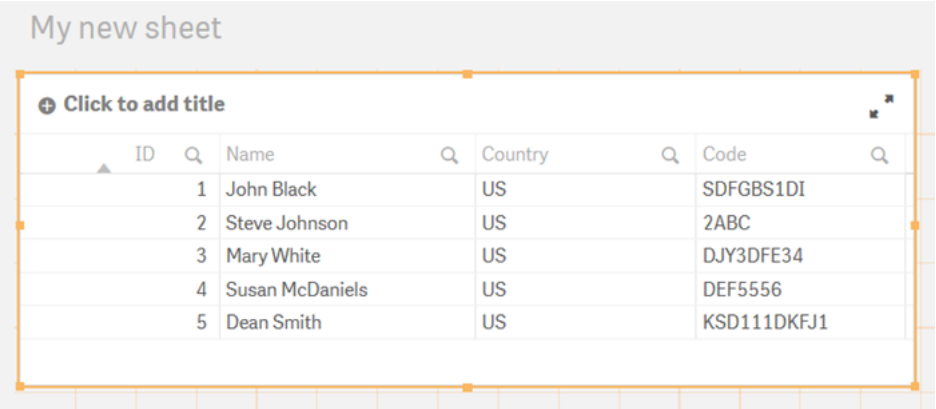
Le script devrait avoir l'aspect suivant :

```
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. Cliquez sur **Charger les données**.

   La table résultante a l'aspect suivant :

   Table affichant des données chargées à l'aide de la fonction ApplyMap



Pour en savoir plus sur ApplyMap(), voir ces articles de blog dans Qlik Community : Don't join - use Applymap instead

4.5 MapSubstring() fonction

La fonction MapSubstring() vous permet de mapper des parties d'un champ.

Dans la table créée par la fonction ApplyMap(), nous souhaitons à présent écrire les nombres sous forme de texte. La fonction MapSubstring() va donc servir à remplacer les données numériques par du texte.

Pour ce faire, il convient tout d'abord de créer une table de mappage.

Procédez comme suit :

1. Ajoutez les lignes de script suivantes à la fin de la section *CountryMap*, mais avant la section *Data*.

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

   Dans la table *CodeMap*, les nombres 1 à 5 et 11 sont mappés.

2. Dans la section *Data* du script, modifiez l'instruction `Code` de la manière suivante :

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

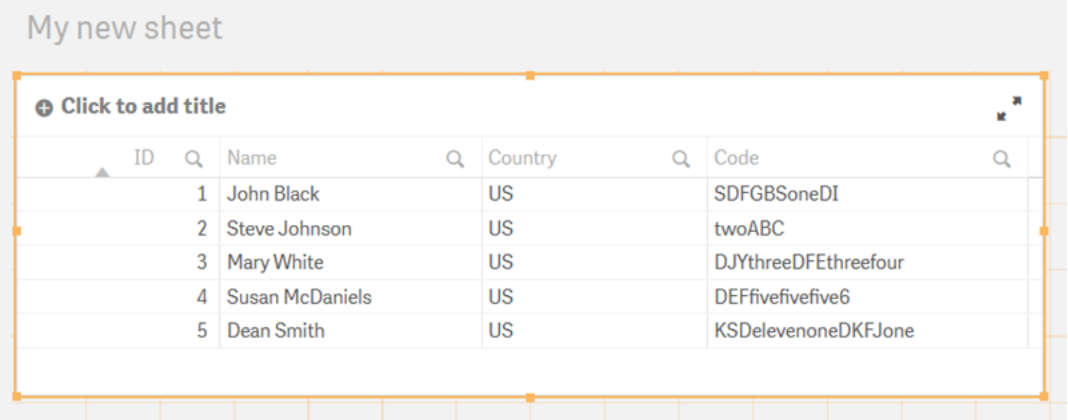
   Le script devrait avoir l'aspect suivant :

   ```
   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. Cliquez sur **Charger les données**.

   La table résultante a l'aspect suivant :

Table affichant des données chargées à l'aide de la fonction MapSubString



Les caractères numériques ont été remplacés par du texte dans le champ *Code*. Si un nombre apparaît plus d'une fois comme dans le cas de ID=3 et ID=4, le texte est également répété. ID=4. *Susan McDaniels* comportait un 6 dans le code. Comme 6 n'était pas mappé dans la table *CodeMap*, il est reste inchangé. ID=5, *Dean Smith*, comportait 111 dans son code. Ce nombre a été mappé en tant que elevenone.

Pour en savoir plus sur MapSubstring(), voir ces articles de blog dans Qlik Community : Mapping … and not the geographical kind *(Mappage... et pas le type géographique)*

4.6 Map … Using

L'instruction Map … Using peut également s'utiliser pour appliquer un mappage à un champ. Cependant, elle fonctionne un peu différemment de l'instruction ApplyMap(). Alors que l'instruction ApplyMap() traite le mappage chaque fois qu'elle rencontre le nom du champ, Map … Using fonctionne lorsque la valeur est stockée sous le nom du champ dans la table interne.

Examinons un exemple. Supposons que nous chargions plusieurs fois dans le script le champ *Country* et que nous souhaitions appliquer un mappage chaque fois que le champ est chargé. Nous pourrions utiliser la fonction ApplyMap() comme illustré précédemment dans ce didacticiel ou opter pour Map … Using.

Si nous choisissons la fonction Map … Using, le mappage est appliqué au champ lorsque celui-ci est stocké dans la table interne. Ainsi, dans l'exemple ci-dessous, le mappage serait appliqué au champ *Country* de la table Data1 mais pas au champ *Country2* de la table *Data2*. Cela s'explique par le fait que l'instruction Map … Using est uniquement appliquée aux champs intitulés *Country*. Lorsque le champ *Country2* est stocké dans la table interne, il n'est plus nommé *Country*. Si vous souhaitez appliquer le mappage à la table *Country2*, utilisez dans ce cas la fonction ApplyMap().

L'instruction Unmap met fin à l'instruction Map … Using. Par conséquent, si *Country* devait être chargé après l'instruction Unmap, *CountryMap* ne serait pas appliqué.

Procédez comme suit :

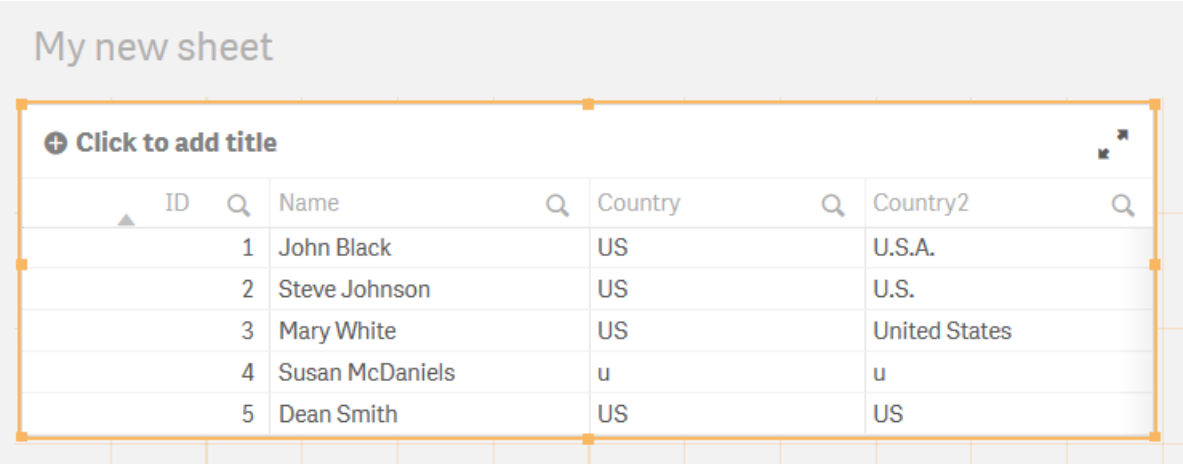
1. Remplacez le script pour la table *Data* avec ce qui suit :

```
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. Cliquez sur **Charger les données**.

   La table résultante a l'aspect suivant :

   Table affichant des données chargées à l'aide de la fonction Map … Using



5 Gestion des données hiérarchiques

Composante essentielle de toutes les solutions d'informatique décisionnelle (BI), les hiérarchies permettent de décrire des dimensions qui contiennent naturellement différents niveaux de granularité. Certaines sont simples et intuitives tandis que d'autres se révèlent complexes et exigent une réflexion approfondie pour être modélisées correctement.

À mesure que l'on descend dans la hiérarchie, ses membres font l'objet d'une description de plus en plus détaillée. Par exemple, dans une dimension comportant les niveaux Market, Country, State et City, le membre Americas figure au premier niveau de la hiérarchie, le membre U.S.A. au deuxième niveau, le membre California au troisième niveau et le membre San Francisco au dernier niveau. California est plus précis que U.S.A., et San Francisco est plus précis que California.

Le stockage de hiérarchies dans un modèle relationnel est une problématique courante aux solutions multiples. Il existe plusieurs approches :

- la hiérarchie horizontale ;
- le modèle de liste de contiguïté ;
- la méthode d'énumération Path ;
- le modèle d'ensembles imbriqués ;
- la liste des ancêtres.

Pour les besoins de ce didacticiel, nous allons créer une liste des ancêtres, car elle présente la hiérarchie sous une forme directement exploitable dans une requête. Vous trouverez des informations complémentaires sur les autres approches sur Qlik Community.

5.1 Préfixe Hierarchy

Le préfixe Hierarchy est une commande de script à placer devant l'instruction LOAD ou SELECT et qui charge une table de nœuds adjacents. L'instruction LOAD doit comporter au moins trois champs : un ID correspondant à une clé unique pour le nœud, une référence au parent et un nom.

Le préfixe permet de transformer une table chargée en table de nœuds étendus ; une table comportant un certain nombre de colonnes supplémentaires ; une par niveau hiérarchique.

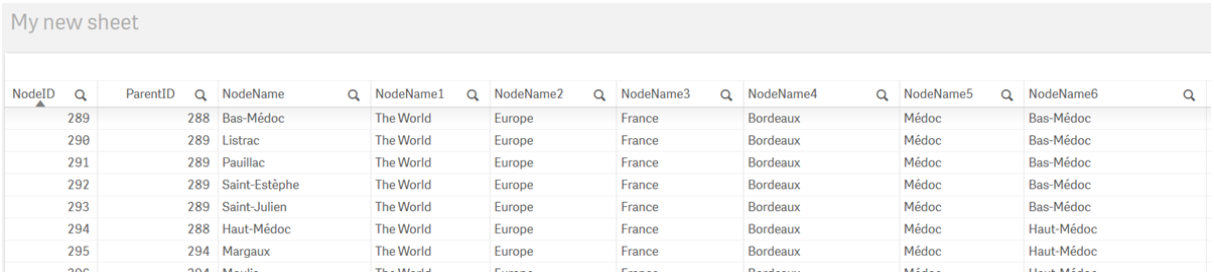
Procédez comme suit :

1. Créez une application et nommez-la.
2. Ajoutez une nouvelle section de script dans l'**éditeur de chargement de données**.
3. Appelez la section *Wine*.
4. Sous **AttachedFiles** dans le menu droit, cliquez sur **Sélectionner des données**.
5. Téléchargez, puis sélectionnez *Winedistricts.txt*.
6. Dans la fenêtre **Sélection de données provenant de**, désactivez les champs *Lbound* et *RBound* pour qu'ils ne soient pas chargés.
7. Cliquez sur **Insérer le script**.

8. Insérez ce qui suit au-dessus de l'instruction LOAD :
   ```
   Hierarchy (NodeID, ParentID, NodeName)
   ```
   Le script devrait avoir l'aspect suivant :
   ```
   Hierarchy (NodeID, ParentID, NodeName)
   LOAD
       NodeID,
       ParentID,
       NodeName
   FROM [lib://AttachedFiles/Winedistricts.txt]
   (txt, utf8, embedded labels, delimiter is '\t', msq);
   ```
9. Cliquez sur **Charger les données**.
10. La section **Aperçu** du **visionneur de modèle de données** vous permet d'afficher la table résultante.
    La table de nœuds étendus résultante comporte exactement le même nombre d'enregistrements que sa table source : un par nœud. La table de nœuds étendus est très pratique, car elle remplit plusieurs exigences relatives à l'analyse d'une hiérarchie dans un modèle relationnel :
    - Tous les noms de nœud figurent dans une seule et même colonne, pour pouvoir être utilisés dans les recherches.
    - De plus, les différents niveaux de nœud ont été étendus en un champ chacun ; ces champs sont utilisables dans des groupes hiérarchiques ou comme dimensions dans les tableaux croisés dynamiques.
    - De plus, les différents niveaux de nœud ont été étendus en un champ chacun ; ces champs sont utilisables dans des groupes hiérarchiques.
    - Elle peut être définie pour contenir un chemin propre au nœud, qui répertorie tous les ancêtres dans le bon ordre.
    - Elle peut être définie pour contenir la profondeur du nœud, c.-à-d. la distance par rapport à la racine.

    La table résultante a l'aspect suivant :

    Table affichant des échantillons de données chargés à l'aide du préfixe Hierarchy



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 Préfixe HierarchyBelongsTo

Comme le préfixe Hierarchy, le préfixe HierarchyBelongsTo est une commande de script à placer devant l'instruction LOAD ou SELECT et qui charge une table de nœuds adjacents.

Dans ce cas également, l'instruction LOAD doit comporter au moins trois champs : un ID correspondant à une clé unique pour le nœud, une référence au parent et un nom. Le préfixe permet de transformer la table chargée en table des ancêtres, une table comportant chacune des combinaisons ancêtre/descendant répertoriées sous forme d'enregistrement distinct. Il est donc très facile de trouver tous les ancêtres ou tous les descendants d'un nœud donné.

Procédez comme suit :

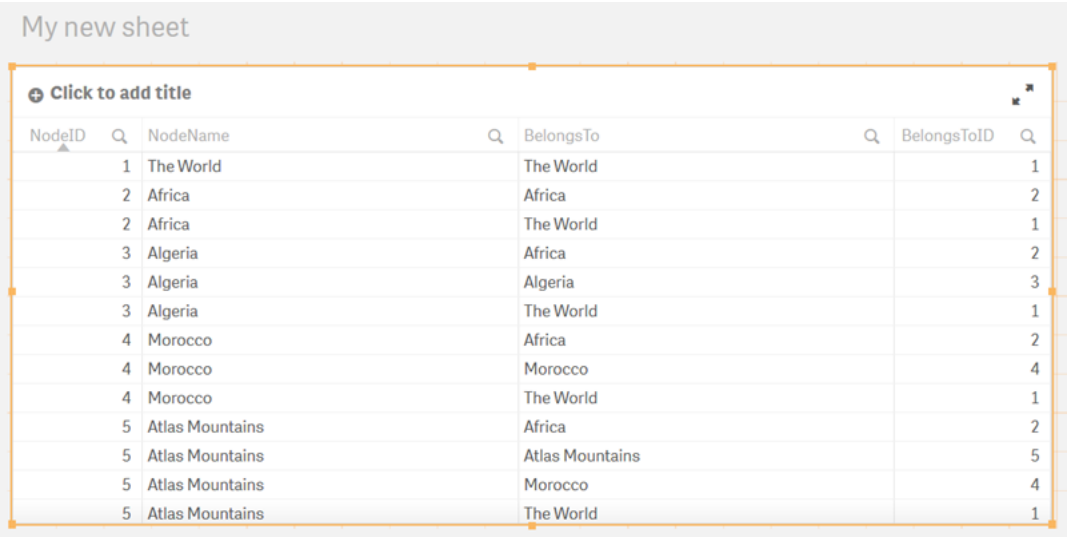
1. Modifiez l'instruction Hierarchy dans l'**éditeur de chargement de données** de la manière suivante : `HierarchyBelongsTo (NodeID, ParentID, NodeName, BelongsToID, BelongsTo)`
2. Cliquez sur **Charger les données**.
3. La section **Aperçu** du **visionneur de modèle de données** vous permet d'afficher la table résultante.

   La table des ancêtres remplit plusieurs exigences relatives à l'analyse d'une hiérarchie dans un modèle relationnel :

   - Si l'ID de nœud représente les nœuds individuels, l'ID de l'ancêtre correspond à des arborescences et sous-arborescences complètes de la hiérarchie.
   - Tous les noms de nœud existent à la fois en tant que nœuds et en tant qu'arborescences, et les deux peuvent être utilisés dans les recherches.
   - Elle peut être définie pour contenir la différence entre la profondeur du nœud et la profondeur de l'ancêtre, c.-à-d. la distance par rapport à la racine de la sous-arborescence.

   La table résultante a l'aspect suivant :

   Table affichant des données chargées à l'aide du préfixe HierarchyBelongsTo



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

Autorisation

Il n'est pas rare qu'une hiérarchie soit utilisée à des fins d'autorisation. Prenons l'exemple d'une hiérarchie organisationnelle. Dans une entreprise, chaque responsable doit être autorisé à accéder à tout ce qui concerne son service, sections sous-jacentes comprises. Cependant, il ne doit pas nécessairement avoir accès aux informations des autres services.

Exemple de hiérarchie organisationnelle

Autrement dit, tous les employés ne sont pas habilités à accéder aux mêmes sous-arborescences de l'entreprise. La table d'autorisations peut revêtir l'aspect suivant :

Table d'autorisation

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	Produit

Dans ce cas, *Carol* est autorisée à voir tout ce qui concerne le niveau *CEO* et les niveaux inférieurs ; *Larry* a accès au niveau *Product* et *James*, au niveau *Engineering* uniquement.

Exemple :

La hiérarchie est souvent stockée dans une table de nœuds adjacents. Dans cet exemple, pour résoudre ceci, vous pouvez charger la table de nœuds adjacents à l'aide de HierarchyBelongsTo et nommer le champ de l'ancêtre *Tree*.

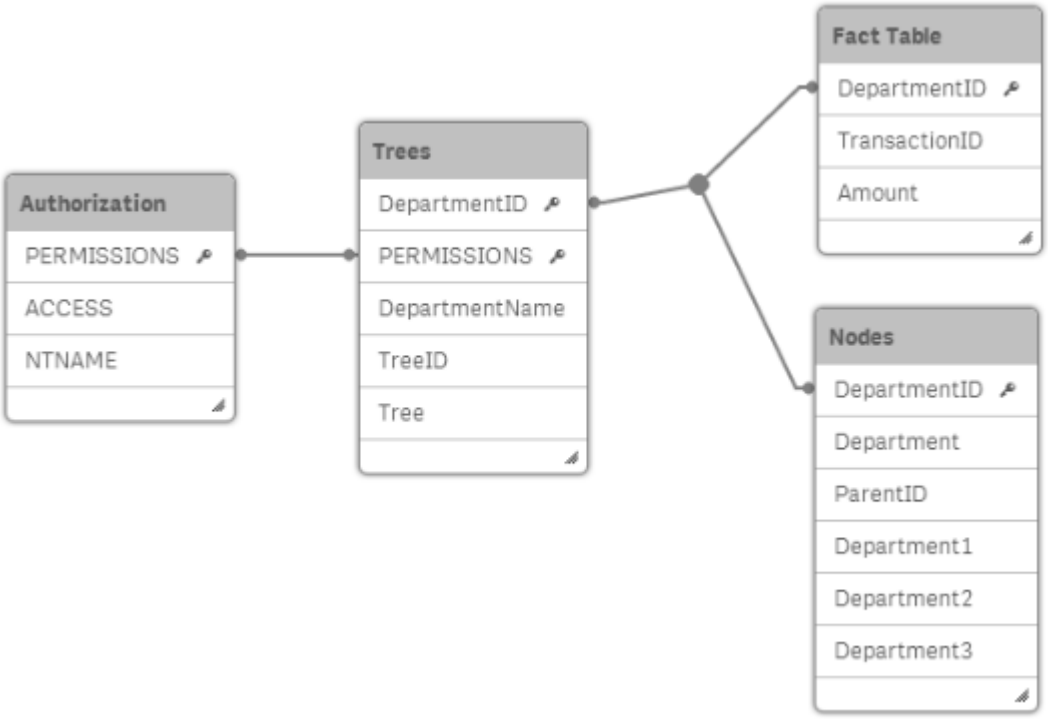
Si vous souhaitez utiliser Section Access, chargez une copie de *Tree* en majuscules et nommez ce nouveau champ *PERMISSIONS*. Enfin, il reste à charger la table d'autorisations. Ces deux dernières étapes peuvent être effectuées à l'aide des lignes de script suivantes : Notez que la table TempTrees correspond à la table créée par l'instruction HierarchyBelongsTo.

Notez qu'il s'agit uniquement d'un exemple. Il n'y a aucun exercice correspondant à terminer dans 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;
```

Cet exemple devrait produire le modèle de données suivant :

Modèle de données : Tables Authorization, Trees, Fact et Nodes



6 Fichiers QVD

Un fichier QVD (QlikView Data) est un fichier qui comprend une table de données exportées à partir de Qlik Sense ou de QlikView. Le format QVD est un format Qlik natif qui ne peut être modifié et lu que par Qlik Sense ou QlikView. Le format de fichier est optimisé pour la vitesse de lecture des données à partir d'un script Qlik Sense tout en demeurant très compact. La lecture de données à partir d'un fichier QVD est généralement 10 à 100 fois plus rapide qu'à partir d'autres sources de données.

Les fichiers QVD peuvent être lus dans deux modes : standard (rapide) et optimisé (plus rapide). Le mode sélectionné est déterminé automatiquement par le moteur de script Qlik Sense. Le mode optimisé s'utilise uniquement lorsque tous les champs chargés sont lus sans aucune transformation (formules agissant sur les champs) ; il est toutefois possible de renommer les champs. Une clause Where entraînant la décompression des enregistrements par Qlik Sense désactive également le chargement optimisé.

Un fichier QVD contient exactement une table de données et se compose de trois parties :

- Un en-tête XML (dans le jeu de caractères UTF-8) qui décrit les champs de la table, la disposition des informations ultérieures et d'autres métadonnées.
- Des tables de symboles dans un format à remplissage de bits.
- Les données de la table dans un format à remplissage de bits.

Les fichiers QVD s'utilisent à de nombreuses fins. Quatre usages principaux sont facilement identifiables. Plusieurs pourront s'appliquer dans n'importe quelle situation :

- Augmentation de la vitesse de chargement des données

  Grâce à la mise en mémoire tampon de blocs de données d'entrée qui ne changent pas ou qui changent lentement dans les fichiers QVD, l'exécution du script devient beaucoup plus rapide pour les grands ensembles de données.

- Augmentation de la vitesse de chargement des données

  Grâce à la mise en mémoire tampon de blocs de données d'entrée qui ne changent pas ou qui changent lentement dans les fichiers QVD, l'exécution du script devient beaucoup plus rapide pour les grands ensembles de données.

- Diminution de la charge sur les serveurs de base de données

  Il est par ailleurs possible de réduire considérablement la quantité de données récupérées à partir de sources de données externes. Conséquence : une diminution de la charge de travail pour les bases de données externes et le trafic réseau. En outre, lorsque les mêmes données sont utilisées dans plusieurs scripts Qlik Sense à la fois, il suffit de les charger une seule fois dans un fichier QVD à partir de la base de données source. Les autres applications peuvent ensuite exploiter les mêmes données par le biais de ce fichier QVD.

- Consolidation de données provenant de plusieurs applications Qlik Sense

L'instruction de script Binary permet de charger les données d'une seule application Qlik Sense dans une autre. Cependant, avec les fichiers QVD, un script Qlik Sense peut combiner les données issues d'un nombre indéfini d'applications Qlik Sense. Cette méthode ouvre des possibilités, par exemple pour des applications destinées à consolider des données similaires issues de différentes unités commerciales, etc.

- Chargement incrémentiel

  Dans de nombreux cas courants, il est possible d'employer la fonctionnalité QVD dans le but de faciliter le chargement incrémentiel, c'est-à-dire en chargeant exclusivement les nouveaux enregistrements d'une base de données en cours d'expansion.

Pour savoir comment la Communauté Qlik utilise Qlik Application Automation pour améliorer les temps de chargement des fichiers QVD, voir Comment diviser des QVD via une automatisation pour améliorer les chargements.

6.1 Création de fichiers QVD

Il existe deux façons de créer un fichier QVD :

- Création et dénomination explicites en utilisant la commande Store dans le script Qlik Sense.

  Il suffit de spécifier dans le script qu'une table déjà lue en totalité ou en partie doit être exportée vers un fichier explicitement nommé à l'emplacement de votre choix.

- Création et maintenance automatiques depuis le script.

  En faisant précéder une instruction load ou select du préfixe Buffer, Qlik Sense crée automatiquement un fichier QVD qui, sous certaines conditions, peut être utilisé à la place de la source de données d'origine lors du rechargement des données.

Les fichiers QVD résultants ne présentent aucune différence en termes de vitesse de lecture.

Store

Cette instruction de script crée un fichier QVD, CSV ou txt avec un nom explicite.

Syntaxe :

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

L'instruction permet uniquement d'exporter des champs provenant d'une table de données. Si vous devez exporter des champs issus de plusieurs tables, définissez au préalable une jointure explicite dans le script afin de créer la table de données à exporter.

Les valeurs de texte sont exportées vers le fichier CSV au format UTF-8. Vous pouvez spécifier un délimiteur (voir **LOAD**). L'instruction store envoyée à un fichier CSV ne prend pas en charge l'exportation BIFF .

Exemples :

```
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';
```

Procédez comme suit :

1. Ouvrez l'application *Advanced Scripting Tutorial*.
2. Cliquez sur la section de script *Product*.
3. Ajoutez ce qui suit à la fin du script :
   ```
   Store * from Product into [lib://AttachedFiles/ProductData.qvd](qvd);
   ```

   Le script devrait avoir l'aspect suivant :

   ```
   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. Cliquez sur **Charger les données**.
   Le fichier *Product.qvd* doit désormais figurer dans la liste des fichiers.

   Ce fichier de données est le résultat du script **Crosstable** ; il s'agit d'une table de trois colonnes, une par catégorie (Product, Month, Sales). Ce fichier de données peut remplacer toute la section de script *Product* à présent.

6.2 Lecture de données à partir de fichiers QVD

Il est possible de lire un fichier QVD ou d'y accéder au moyen de Qlik Sense en recourant aux méthodes suivantes :

- Chargement d'un fichier QVD comme source de données explicite. Les fichiers QVD peuvent être référencés par une instruction load dans le script Qlik Sense comme tout autre type de fichier texte (csv, fix, dif, biff, etc).

Exemples :

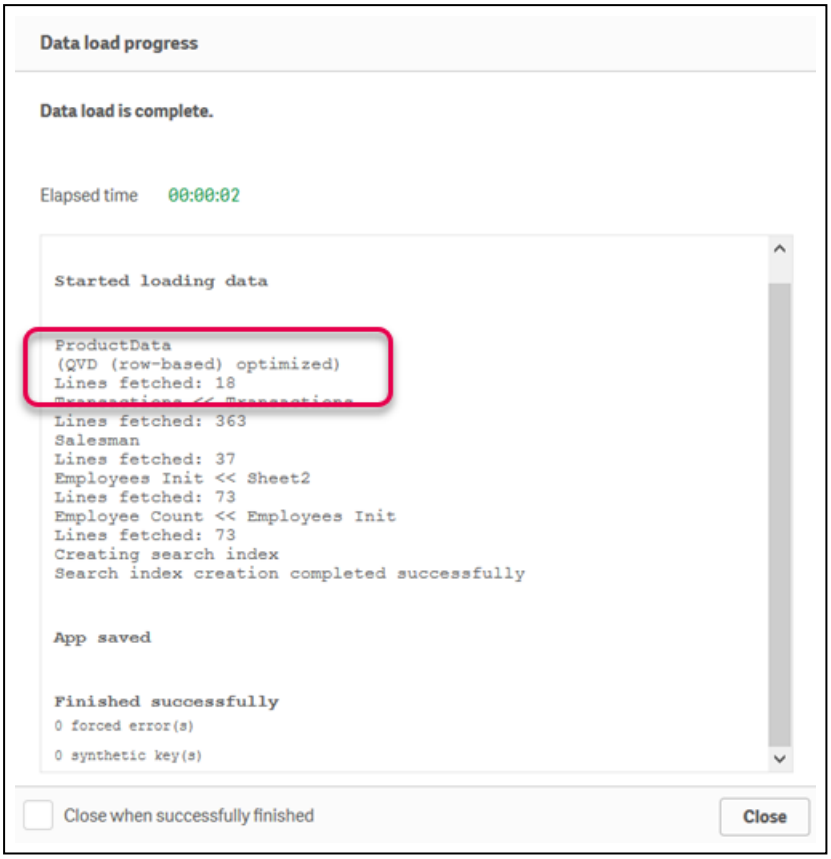
```
LOAD * from 'lib://FolderConnection/xyz.qvd' (qvd);
LOAD fieldname1, fieldname2 from [lib://FolderConnection/xyz.qvd] (qvd);
LOAD fieldname1 as newfieldname1, fieldname2 as newfieldname2 from
[lib://AttachedFiles/xyz.qvd](qvd);
```

- Chargement automatique de fichiers QVD placés en mémoire tampon. Lorsque vous utilisez le préfixe buffer avec des instructions load ou select, aucune instruction explicite n'est nécessaire pour la lecture. Qlik Sense détermine dans quelle mesure il utilisera les données du fichier QVD plutôt que d'acquérir les données au moyen de l'instruction LOAD ou SELECT d'origine.
- Accès aux fichiers QVD à partir du script. Il est possible d'utiliser un certain nombre de fonctions de script (toutes celles commençant par QVD) pour récupérer différentes informations sur les données figurant dans l'en-tête XML d'un fichier QVD.

Procédez comme suit :

1. Mettez des commentaires sur tout le script dans la section de script *Product*.
2. Saisissez les lignes de script suivantes :
   ```
   Load * from [lib://AttachedFiles/ProductData.qvd](qvd);
   ```
3. Cliquez sur **Charger les données**.
   Les données sont chargées à partir du fichier QVD.

Fenêtre de progression de chargement de données



Pour en savoir plus sur l'utilisation de fichiers QVD pour les chargements incrémentiels, consultez cet article de blog dans Qlik Community: Vue d'ensemble du chargement incrémentiel Qlik

Buffer

Il est possible de créer et de gérer automatiquement des fichiers QVD à l'aide du préfixe Buffer. Ce préfixe peut être utilisé dans la plupart des instructions LOAD et SELECT du script. Il indique que des fichiers QVD sont utilisés pour mettre en cache/mémoire tampon le résultat de l'instruction.

Syntaxe :

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

Si aucune option n'est utilisée, le tampon (buffer) QVD créé par la première exécution du script est utilisé indéfiniment.

Exemple :

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

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

Amount est un nombre spécifiant la période. Cet argument admet l'utilisation de décimales (Decimals). Si l'unité n'est pas précisée, les jours sont utilisés par défaut.

L'option stale after est généralement utilisée avec des sources de base de données qui ne comportent pas d'horodatage simple pour les données d'origine. Une clause stale after spécifie simplement une période commençant à la date de création du tampon QVD et après laquelle celui-ci ne sera plus considéré comme valide. Avant l'expiration de ce délai, le tampon QVD est utilisé comme source de données et après ce moment, c'est la source de données d'origine. Le fichier de tampon QVD est alors mis à jour automatiquement et une nouvelle période débute.

Exemple :

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

Incremental

L'option incremental permet de ne lire qu'une partie d'un fichier sous-jacent. La taille précédente du fichier est stockée dans l'en-tête XML du fichier QVD. Cette méthode s'avère particulièrement utile dans le cas des fichiers journaux. Tous les enregistrements chargés précédemment sont lus à partir du fichier QVD tandis que les nouveaux enregistrements ultérieurs sont lus à partir de la source d'origine, avant qu'un fichier QVD à jour ne soit créé.

Notez toutefois que l'option incremental s'emploie exclusivement avec des instructions LOAD et des fichiers texte, et qu'il est impossible d'utiliser le chargement incrémentiel lorsque d'anciennes données ont été modifiées ou supprimées.

Exemple :

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

Les tampons QVD sont normalement supprimés lorsqu'ils ne sont plus référencés nulle part lors d'une exécution de script complète dans l'application qui les a créés ou lorsque l'application qui les a créés n'existe plus. Il est recommandé d'utiliser l'instruction Store pour conserver le contenu du tampon dans un fichier QVD ou CSV.

Procédez comme suit :

1. Créez une application et nommez-la.
2. Ajoutez une nouvelle section de script dans l'**éditeur de chargement de données**.
3. Sous **AttachedFiles** dans le menu droit, cliquez sur **Sélectionner des données**.
4. Téléchargez, puis sélectionnez *Cutlery.xlsx*.
5. Dans la fenêtre **Sélectionner des données depuis**, cliquez sur **Insérer le script**.
6. Mettez des commentaires sur les champs dans l'instruction de chargement, puis changez l'instruction de chargement pour ce qui suit :
   ```
   Buffer LOAD *
   ```

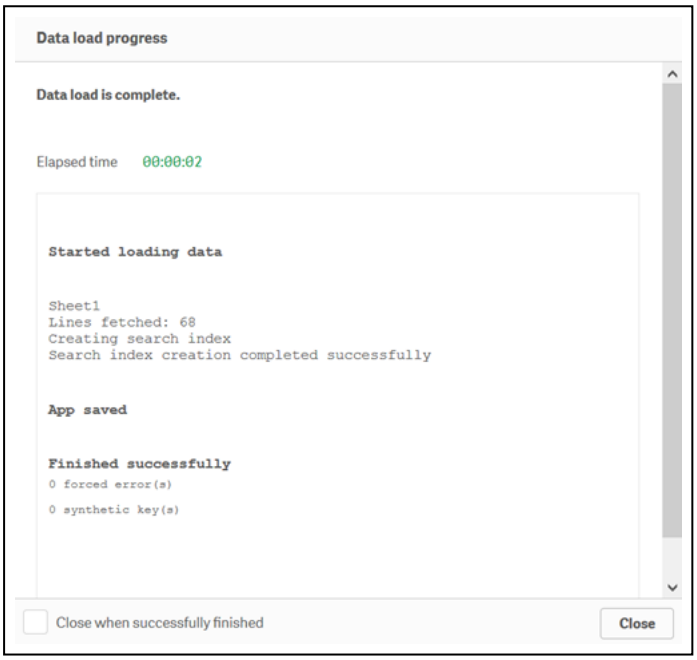
   Le script devrait avoir l'aspect suivant :

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

7. Cliquez sur **Charger les données**.

   La première fois que vous chargez des données, elles sont chargées à partir de *Cutlery.xlsx*.

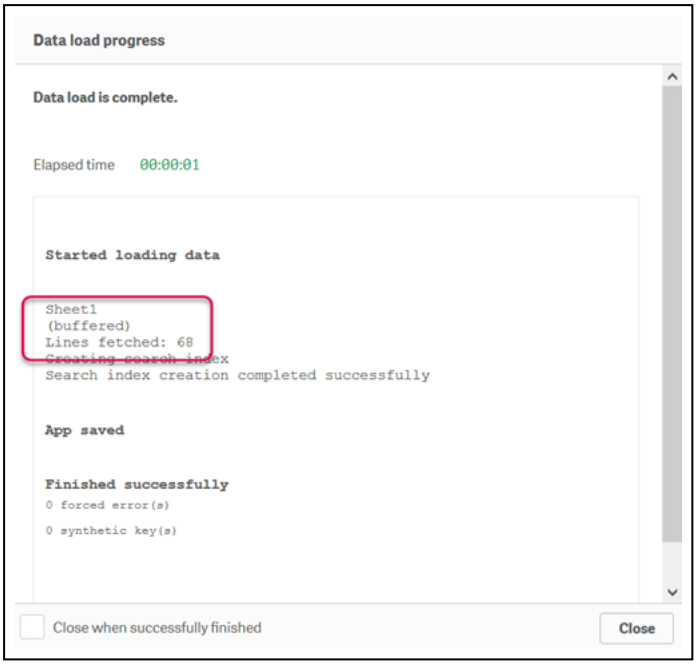
   Fenêtre de progression de chargement de données



   L'instruction Buffer crée également un fichier QVD et le stocke dans Qlik Sense. Dans un déploiement Qlik Sense Enterprise on Windows, il est stocké dans un répertoire sur le serveur Qlik Sense.

8. Cliquez de nouveau sur **Charger les données**.
9. Cette fois-ci, les données sont chargées à partir du fichier QVD créé par l'instruction Buffer lorsque vous avez chargé les données la première fois.

Fenêtre de progression de chargement de données



6.3 Merci !

Maintenant que vous êtes parvenu au terme de ce didacticiel, nous espérons que vous avez acquis de nouvelles connaissances en matière de création de scripts dans Qlik Sense. Pour en savoir plus sur les formations complémentaires disponibles, rendez-vous sur notre site Web.