QlikView Essentials

This guide demonstrates just how easy it is to get started with QlikView and create your own BI application. Featuring an introduction to its core features before exploring how to load data and model it, it will help you become more confident that you can take full advantage of QlikView’s capabilities. You will also learn how to use QVD files with QlikView and how they offer a simpler way of handling data.

After digging deeper into data handling, as you learn how to use mapping tables and create a master calendar, you’ll then find out how to get the most from QlikView’s visualization features — vital if you are to use your data insights effectively.

With details on how to finally secure your application and deploy it for a successful integration into your organization, QlikView Essentials underlines exactly why QlikView is becoming more and more popular for businesses that understand the value of data.

Who this book is written for

If you recognize the challenges of harnessing data for a modern business, this book is for you. Maybe you already know a little about QlikView — if you want to learn more, QlikView Essentials is a great way to develop your knowledge and skills.

What you will learn from this book

- Learn the complete QlikView workflow — from loading data to visualization and analytics
- Learn how to load data from different sources, including QVD files, and how to optimize data models for accuracy and precision
- Discover solutions to common data modeling problems so you can respond quickly to changing situations
- Create accessible dashboards and quality data visualizations to share insights effectively
- Learn how to deploy your BI application for optimal availability

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 3 'Optimizing Your Data Model'
- A synopsis of the book’s content
- More information on QlikView Essentials
Chandraish Sinha is a Business Intelligence enthusiast from Ohio. He brings 17 years of experience in providing cost-effective BI solutions. He is responsible for many QlikView implementations in various industries, namely financial, insurance, pharmaceuticals, and event management. He currently holds QlikView Designer and QlikView Developer certifications. He shares his knowledge and passion through his QlikView blog (http://www.learnallbi.com/). He currently works as an independent BI consultant and helps organizations in implementing BI solutions.
Preface

The data is growing at a higher pace so does the need to understand data. There are many applications that perform data analysis and design, but QlikView takes Business Intelligence to the next level. The ability of QlikView to extract and present the data in a way that the human mind thinks, has made QlikView hugely popular. The associative nature of the QlikView data model has made business discovery fairly simple.

This book is being designed in a way that provides equal value to a novice BI developer and a seasoned practitioner. This book starts with the basics of QlikView, data warehousing and works through creating data models and visualizations. This book covers all the topics for the QlikView designer and developer and can be used as a reference guide in new or ongoing implementations.

Each chapter in the book follows a structure:

- Each chapter will cover the essentials of the topic in the chapter.
- This book covers both QlikView developer/Data model and QlikView designer/visualization topics.
- Each topic is explained first and then followed by a step-by-step exercise. Readers can follow these exercises to create their own data model and dash boarding application.
- This book uses the Adventure Works database; most readers will have familiarity with this database. There is tons of information available online on this database, so users will easily understand.
- This book also comes with data in MS Access, Excel, and text files. It is also accompanied with QlikView solutions/qvw's that the reader can download and follow.
What this book covers

Chapter 1, QlikView Essentials, provides the basics of QlikView. It gives an overview of the QlikView architecture. It also provides instructions on how to download QlikView. You get to know about the star schema and learn about the underlining data model used in the book.

Chapter 2, Extract, Transform, and Load, as the name suggests, dives into building a data model in QlikView by extracting, transforming, and loading data. In this chapter, readers will learn about using scripts to load data from different sources and data transformation.

Chapter 3, Optimizing Your Data Model, deals with techniques to optimize a data model. This involves different ways to join data and data aggregation.

Chapter 4, Data Modeling Challenges, helps you in understanding and resolving different data modeling challenges. You learn about loading some special table types. You will also learn about the best practices of data modeling.

Chapter 5, Creating Dashboards, gives you the opportunity to learn about different dashboarding practices and create different visualization objects.

Chapter 6, Comparative Analysis, enforces the importance of data comparison in the dashboards using Set Analysis and Alternate State. You also learn about implementing what-if analysis.

Chapter 7, Securing Your Application, teaches how to secure your dashboard application.

Chapter 8, Application Deployment, provides an overview of servers and how applications can be deployed on the server.
Optimizing Your Data Model

Your data model should always be simple and easy to understand. An expert data modeler will always clean up the data model to remove any unnecessary tables.

QlikView provides different techniques for data model cleanup. In this chapter we will learn ways to clean up the data model and create a star schema.

In this chapter we shall:

- Learn about mapping tables and mapping load
- Clean up the data model using "mapping load" and "apply map"
- Learn about different kinds of joins
- Aggregate data
- Learn to use "concatenate"

Let's review the data model that we created in the last chapter:
One of our requirements is to create a robust data model that is clean, easy to understand, and helps in data analysis. This chapter describes techniques to achieve this requirement. By understanding and applying the concepts of mapping load, concatenate, and joins, we can create a more useful data model.

**Mapping table essentials**

Mapping tables are very useful in QlikView data modeling. Some of the uses of mapping tables include data cleansing, renaming the data source-specific column names with business-friendly names, and providing comments in the script. The following points discuss some of the components and properties of mapping tables:

- Mapping load is used to reduce the number of tables in a data model. Mapping prefix is used to create a mapping table. Mapping tables are used only for field mapping and are automatically dropped after script execution.
- These are usually the look-up tables.
- Mapping tables must have two columns. Even if the table has more columns, you can use only two columns at a time to achieve mapping table functionality. The same table can be used multiple times in case other columns need to be mapped.
- The first column is always a key (ID) column. The second column contains the desired mapping value. The first column field name can differ between the mapping table and the mapped table.
- Use the `ApplyMap` function in the table to compare the key/ID field to get the desired mapping value.
- It provides a third optional parameter when no comparison is found.

Looking at the previous data model, you can identify the tables that are good candidates for mapping load. These tables will be:

- Territory
- ProductSubcategory
- Product Category
- CountryRegion_Inline
Mapping load – Territory table
The objective here is to add TerritoryDesc to the Orders table and remove the Territory table from the data model. This will help in data model cleanup as one table will be reduced:

1. As a best practice, all mapping loads should be in one tab. This tab should be the very first tab in the script after the Main tab.
2. Open the QlikView file QlikViewEssentials.qvw that you have been using so far. Save as QlikViewEssentials_Chop3.qvw.
3. Invoke the script editor by pressing Ctrl + E.
4. Create a new tab and name it Mapping. Move this tab all the way to the left by using the Promote Tab icon as highlighted in the following screenshot. It should be your tab after the Main tab.
5. Navigate to the Dimensions tab and cut the load script of SalesTerritory and paste it in the Mapping tab.
6. Navigate to the Mapping tab, change the comment, and name the table as Territory_Map.
7. Prefix the load statement with the Mapping keyword. This will treat the SalesTerritory table as a mapping table. This table will not be present in the memory and will get dropped after the script execution.
8. Make the first column as the ID column and name it TerriID, and name the second column as TerriDesc. Names are optional.
9. The mapping table load script should look like the following:

```
//*********Sales Territory Map*********
Territory_Map:
Mapping LOAD
    TerritoryID  As TerriID,
    TerritoryDesc  As TerriDesc;
SQL SELECT *
FROM SalesTerritory;
```
10. Navigate to the **Orders** tab, and go to the last line in the **Orders** table load statement after **TotalDue**. Add "," after **TotalDue**. Use the **ApplyMap** function to add a **TerritoryDesc** column in the **Orders** table. Name this column as [Order Territory].

```sql
ApplyMap('Territory_Map', TerritoryID, 'Territory Not Found') As [Order Territory]
```

This is the **ApplyMap** syntax. Here, **Territory_Map** is the name of the mapping table. **TerritoryID** is the lookup column. 'Territory Not Found' is an optional parameter in case the match is not found.

11. Your **Orders** table script should look like the following:

```sql
/***********Orders***********/
LOAD
CustomerID,
DueDate,
EmployeeID,
EmployeeID As SalesEmployeeID,
Freight,
OnlineOrderFlag,
OnlineOrderFlag As OnlineOrderFlagCount,
OrderDate,
Year(OrderDate) As Year,
Month(OrderDate) As Month,
OrderID,
ShipDate,
ShipperID,
Status,
SubTotal,
TaxAmt,
TerritoryID,
TotalDue,
ApplyMap('Territory_Map', TerritoryID, 'Territory Not Found') As [Order Territory]
SQL SELECT * FROM OrderHeader;
```

12. Reload your script.

13. Navigate to Table Viewer (Ctrl + T). You will notice that the **Territory** table has gone and the **Order Territory** column has been added to the **Orders** table.

14. Create a list box for **Order Territory**.

15. To create a list box, close the script editor by clicking **OK**. Anywhere in the empty space on the sheet, right click and choose **New Sheet Object** and select **List Box**. In the properties of the list box, select **Order Territory** under the **Field** section.
You will see a new field is being created. Notice Territory Not Found because in some cases no match is found between the two tables.

<table>
<thead>
<tr>
<th>Order Territory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Australia</td>
</tr>
<tr>
<td>Canada</td>
</tr>
<tr>
<td>Central</td>
</tr>
<tr>
<td>France</td>
</tr>
<tr>
<td>Germany</td>
</tr>
<tr>
<td>Northeast</td>
</tr>
<tr>
<td>Northwest</td>
</tr>
<tr>
<td>Southeast</td>
</tr>
<tr>
<td>Southwest</td>
</tr>
<tr>
<td>Territory Not Found</td>
</tr>
<tr>
<td>United Kingdom</td>
</tr>
</tbody>
</table>

### Mapping load – Product Category and ProductSubcategory tables

Here the objective is to add the [Category Desc] to the ProductSubcategory table and then we will add Category desc and subcategory name from the ProductSubcategory table to the Product table:

1. Open QlikViewEssentials_Chap3.qvw.
2. Invoke the script editor by pressing Ctrl + E.
3. Navigate to the Dimensions tab and cut the Product Category load script and paste it in the Mapping tab.
4. The mapping load of Product Category will appear as follows:

```sql
){//**********Category Map**********
Category_Map:
Mapping LOAD
   ProductCategoryID As CategoryID,
   CategoryName As CategoryName
FROM
   [\Data\ProductCategory.txt]
(txt, codepage is 1252, embedded labels, delimiter is "t", msq):
```
5. Navigate to the **Dimensions** tab. Go to the last line of the `ProductSubCategory` load script and use `ApplyMap` to add `[Category Desc]` to the `ProductSubcategory`.

```sql
/**********ProductSubCategory**********/
ProductSubCategory:
  LOAD
    ProductCategoryID,
    ProductSubcategoryID,
    SubCategoryName,
    ApplyMap('Category_Map',ProductCategoryID,'N/A') As [Category Desc],
  SQL SELECT *
FROM ProductSubcategory;
```

6. Save and execute your script. Review your data model in Table Viewer to see that the `Product Category` table is removed and `Category Desc` is added to the `ProductSubcategory` table.

**Mapping load – ProductSubcategory table**

Use mapping load with the `ProductSubcategory` table to get the subcategory name:

1. Navigate to the **Dimensions** tab and cut the `ProductSubcategory` load script and paste it in the **Mapping** tab.

2. The mapping load of `ProductSubcategory` for `SubCategoryName` and `[Category Desc]`, will appear as follows:

```sql
/**********SubCategory_Name_Map**********/
SubCategory_Name_Map:
Mapping LOAD
  ProductSubcategoryID,
  SubCategoryName;
SQL SELECT *
FROM ProductSubcategory;

/**********SubCategory_Category_Desc_Map**********/
SubCategory_CategoryMap:
Mapping LOAD
  ProductSubcategoryID,
  ApplyMap('Category_Map',ProductSubcategoryID,'No Category')
SQL SELECT *
FROM ProductSubcategory;
```
3. Navigate to the **Dimensions** tab and go to the last line of the *Product* load script. Use *ApplyMap* to add *SubCategoryName* and *CategoryName* to the *Product* table. *ApplyMap* statements in the *Product* table will appear as follows:

```
ApplyMap(SubCategory_Map, ProductSubcategory(D, No SubCategory)) As [SubCategory Desc],
ApplyMap(SubCategory_CategoryMap, ProductSubcategory(D, No Category)) As [Category Desc].
```

4. Save and load the script. View your data model in Table Viewer. The *ProductSubcategory* table is removed and *SubCategoryName* and *CategoryName* is added to the *Product* table.

**Mapping load – CountryRegion_Inline**

Use mapping load to get the *CountryRegion_Desc* from *CountryRegion_Inline* and include it in the *Customers* table:

1. The *CountryRegion_Inline* table can also be converted to the mapping table.
2. Navigate to the **Inline** tab and cut the *CountryRegion_Inline* load script and paste it in the **Mapping** tab.
3. The mapping load script for *CountryRegion* will appear as follows:

```csharp
//***********CountryRegion Inline Map***********
CountryRegion_Inline_Map:
Mapping LOAD * INLINE [CountryRegionName, Region
Australia, Oceania
Canada, North America
France, Europe
Germany, Europe
United Kingdom, Europe
United States, North America
];
```

4. Navigate to the **Dimensions** tab. Go to the load script of the *Customers* table and, in the last line, use *ApplyMap* as follows:

```
ApplyMap(CountryRegion_Inline_Map, CountryRegionName,'N/A') As CountryRegionDesc;
```

5. Save and load the script.
6. Review the data model in Table Viewer. The data model so far will appear as follows. It is much cleaner with fewer tables.

**Concatenation**

Another way to optimize your data model in QlikView is through concatenation.

Concatenation is a way to combine or merge tables. It is similar to `Union All` in SQL. It appends rows from one table to another. The result of concatenation between two tables (Table1 and Table2, for example) is that a new table contains the sum of the numbers of records in Table1 and Table2. Concatenate is used when you have two fact tables in your data model. Two fact tables are not good for your data model. You can combine them using concatenate or link tables. Link tables are explained in the next chapter.

Concatenate is also useful when you have to combine two tables with similar structures. Say, for example, your organization has an `Employee` table that stores employee data. Your organization buys another company and now wants to merge the details of the employees of this new company with the existing `Employee` table.
Let's explore different concatenation options:

- **Automatic concatenation**: If the field names and the number of fields in two or more tables are exactly the same, QlikView will automatically concatenate the output of the different load statements into one table.

- **Forced concatenation**: If the field names and the number of fields in two or more tables are NOT exactly the same, QlikView will allow you to force concatenation by the use of `Concatenate` keyword.

- **No concatenate**: If the field names and the number of fields in two or more tables are exactly the same, QlikView will automatically concatenate them but you can avoid such automatic concatenation by using the `NoConcatenate` keyword.

### Concatenation example

Create an Inline table in QlikView to understand the concept of concatenation:

**Automatic Concatenation**

1. Navigate to the Inline tab.
2. Create two Inline tables as shown in the following:

```plaintext
//Concatenate example
Table1: Load * Inline [
col1,col2
a,b
c,d
e,f
];

Table2: Load * Inline [   
col1,col2
a,b
c,d
e,f
];
```

3. Save and reload the script.
4. View the data model in Table Viewer. You will see only one table with six rows. This is because `Table2` automatically concatenated with `Table1`. 
Optimizing Your Data Model

**NoConcatenate**

Use the same script and add `NoConcatenate` between the two load scripts:

In Table View you will see two tables, `Table1` and `Table2`. You will see that the synthetic key between them, as column names in the tables, are the same.

**Forced concatenation**

To test force concatenation, comment the previous `Table1` and `Table2` load scripts.

1. Now create two tables with different columns and use the `Concatenate` keyword to force concatenation.

   ```sql
   //Force Concatenate example
   Table1:
   Load * Inline [  
   col1,col2  
   e,b  
   c,d  
   e,f  
   ];  
   Concatenate  
   Table2:
   Load * Inline [  
   col3,col4  
   1,2  
   3,4  
   5,6  
   ];
   ```

2. After executing the script, notice that Table Viewer shows one single table, `Table1`. Even though the column names in the two tables are different, QlikView merged them.
Joins

Joins in QlikView are similar to joins in SQL. Joins between two tables always result in one table. With joins, the columns of the two tables are affected. Joins are explicitly made by using inner join, left join, right join, and outer join.

In QlikView, joins work in the following ways:

- **Inner join**: Only the matching records in the two tables are stored.
- **Left join**: All the records from the first/left table are kept, and only those records from the second table that match a record in the first table are kept.
- **Right join**: All the records from the second/right table are kept, and only those records from the first table that match a record in the second table are kept.
- **Outer join or join**: Records from both the tables will be stored and, where possible, records will be matched.

Use Joins.qvw to practice different kinds of joins and concatenate options. It is located in your Apps folder.

Join Orders and Order Detail

The Orders and Order Detail tables will be joined because an optimized data model should have only one fact table.

We will use left join between the tables:

1. Use the QlikViewEssentials_Chp3.qvw. Navigate to the Orders tab in your script editor.
2. After the load scripts of the Orders table, type left join (Orders).
3. Comment the [Order Detail] table name as this table name will be irrelevant now.
4. Your updated script will appear as follows:

```sql
/*Orders*/
ORDERS
LOAD
CustomerID,
DueDate,
EmployeeID,
EmployeeID As SalesEmployeeID,
Freight,
OrderID,
OrderDate,
Year(OrderDate) As Year,
Month(OrderDate) As Month,
OrderID,
ShipDate,
ShipperID,
Status,
SubTotal,
TaxAmt,
TerritoryID,
TotalDue,
ApplyMap(Territory,Map,TerritoryID,Territory Not Found) As [Order Territory]
FROM OrderHeader;

left join (Orders)
   // This will join Order Detail with Orders. There will be one Orders table in the data model
   // This will join Order Detail with Orders. There will be one Orders table in the data model
   // Order detail name is relevant now as it will be combined with the Orders table
   // This will ensure that all the columns from the previous load are loaded
   // This will ensure that all the columns from the previous load are loaded
   // Preceding load. Takes LineTotal from the previous load statement
LOAD *
   + Discount
   + OrderID
   + OrderQty As Quantity,
   + ProductID,
   + UnitPrice,
   + UnitPrice As OrderQty As LineTotal ;
   + SQL SELECT *
   FROM OrderDetail;
```

5. Observe the data model in Table Viewer and you will see the Orders and Order Detail tables have combined into one table, Orders.

**Left join SalesPerson**

To further optimize your data model, left join SalesPerson with the Orders table:

1. Navigate to the Employees Tab. Go to the SalesPerson load script.
2. In the empty space before the load statement, type `left join (Orders)`.
3. This will left join Orders and SalesPerson. It is good practice to type the table name with left join otherwise it will left join with the previously loaded table script.
4. Save and reload.
5. Observe the Orders table in the Table Viewer. SalesPerson Name and SalesPersonTitle have been added.

Aggregating data
We have combined the Orders and Order Detail tables; now it is time to find the sales amount for each of these orders by aggregating the data by OrderID:

1. Use QlikViewEssentials_Chap3.qvw. Navigate to the Orders tab in your script editor.
2. Navigate to the empty space after the Order Detail load statement.
3. Perform "resident load" from the Orders table and aggregate using the Group By function. Resident load is used here as the Orders table is already loaded in the script and we can reuse the same table using resident load.
4. It will be better to keep [Order Sales Amt] in the Orders table as it is at the Order level. To achieve this, go to the empty space above the load statement and type left join(Orders).
5. Your script should appear as the following:

```qlik
left Join(Orders)
[Order Aggregate]:
Load
distinct
OrderID,
Sum(LineSalesAmount) As [Order Sales Amt]
resident Orders
Group By OrderID;
```

Concatenating the new Employees table
As the QlikView development was progressing, Adventure Works Inc. acquired a new company. Now they want to merge the new company's employees into the Adventure Works' Employees table.

This can be best achieved by using Concatenate, since we know that two tables with the same structure get automatically concatenated. Even if the structure of NewEmployees is not the same, we should make it the same so that it concatenates with Employees.
Optimizing Your Data Model

The new employee file structure is similar to the Employees table loaded in QlikView.

This new employee file NewEmployees.xlsx is also located in the same location:

1. Navigate to the Employees tab in QlikViewEssentials_Chap3.qvw.
2. Go to the empty space after the SalesPerson load script.
3. Click on Table Files and browse to NewEmployees.xlsx, which is located under C:\QlikViewEssentials\Data. Make sure to select Embedded labels under Labels while selecting the file.
4. Remove the directory. Save and reload your script. You will observe just one table, Employees, in the data model and it will have records of both Employees and NewEmployees.
5. The script of the NewEmployees table will appear as the following:

```sql
/***********New Employees Load ***********/
LOAD EmployeeID,
[Emp FirstName],
[Emp MiddleName],
[Emp LastName],
JobTitle,
StartDate,
EndDate,
[Emp PhoneNumber],
[Emp Email Address],
[Emp Address],
[Emp City],
[Emp State],
[Emp PostalCode],
[Emp Country]
FROM
[\Data\NewEmployees.xlsx]
{\xml, embedded labels, table is Sheet1}),```
Reviewing the final data model

By applying the previous techniques, the data model is being converted into a star schema.

Summary

QlikView always desires a star schema as it is simple to understand and efficient for reporting. In this chapter, we learned different techniques of creating a star schema and converted our previously created data model into a star schema by using mapping loads, joins, and concatenation.

In the next chapter, we will learn about data modeling challenges, which are resolved by using cross tables, link tables, and the master calendar. You will also learn about handling slowly changing dimensions using the Interval Match function. Finally we will use QVDs to read and write our data model tables.
Where to buy this book

You can buy QlikView Essentials from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.