CHAPTER 13

Creating Reports with SQL Queries

I’m convinced you can combine this with reporting integrity and accuracy.

—Jack Brickhouse

With the data warehouse completed and filled with data, it is time to realize some of the fruits of your efforts by making your first preliminary reports.

In this chapter, we take a look at the process of creating report code using the SQL programming language and some of the decisions that you have to make regarding this code. We walk you through the process of creating report queries from start to finish.

A data warehouse can have thousands of reports against it, and all reports have aspects in common:

- Reports are viewed with user applications such as Excel, PowerPivot, or Reporting Server Reports.
- The underlying code is most often written in SQL or MDX (query languages used for relational and OLAP databases).
- All report code should be consistent and well formed.
- Abstraction layers are used to keep code maintenance costs low.

Reporting queries can become complex very quickly, and complex queries can be overwhelming for developers who do not do a lot SQL programming; therefore, we do what we can to keep them simple. And, to make things more understandable, we break the SQL code into bite-size chunks and break down each of the features that typically make up a standard SQL report query. Using the methods in this chapter, you will soon be writing polished, professional, and accurate queries.

Note Examples in this chapter are SQL-based. MDX is the other common reporting language in use. We cover MDX next, in Chapter 14.
Identifying the Data

The first step is to determine the type of report your client needs. This falls into the category of “identifying the data.” After having been through the interview process with your client, you should have at least some idea of what they are looking for.

At this point, it is common for the client to be somewhat unsure of what they need. But after they have seen an initial report, it can help them be more specific. This process becomes more exact after you have presented your preliminary versions and examples to them. After your first reports, you will likely be refining them in versions 2, 3, or even 4. This is part of the process of learning what questions to ask before diving into creating the first version of the report.

We recommend getting started by creating a commented header section at the beginning of each SQL script. The comment might look something like the one shown in Listing 13-1.

Listing 13-1. A Script Header

```sql
/*****************************/
Title:SalesByTitlesByDates
Description: Which titles were sold on which dates
Developer:RRoot
Date: 6/1/2012

Change Log: Who, When, What
CMason,6/2/2013,fixed numerous grammatical errors
/*****************************/
```

The script header is similar to what we have used in the past, but most companies have their own standards of what information they require to be inserted into this header. If the company does not already have a standard for this, now would be a good time to establish one. Once you have a header outlining what you want to accomplish, you need to locate the data within the data warehouse.

Listing 13-2 is a very simple SELECT statement against the fact table from which most reports will originate. Notice that we have formatted the SELECT statement to be more legible by stacking the column listings in a vertical fashion. Over the years Microsoft seems to have settled on this being a best practice, and we agree that it does make things easier to read when you have to go through a large amount of code.

Listing 13-2. A Basic Starter Query

```sql
SELECT
  OrderNumber
, OrderDateKey
, TitleKey
, StoreKey
, SalesQuantity
FROM DWPubsSales.dbo.FactSales
```

One simple addition that makes reporting easier long-term is using fully qualified names for objects. This example includes not only the name of the table but its database name of DWPubsSales and schema name of DBO as well. Maintenance on reports includes tracking which reports are connected to which databases and database objects. Using fully qualified names in your SQL queries can help with this process and is a simple addition that takes little time to implement. Besides, you also get a small gain in performance, because the database engine does not have to resolve the object name implicitly. As shown in Figure 13-1, the results you get back are not particularly pleasing to the eye.