Microsoft Power Tools for Data Analysis #11

Connecting to an SQL Server Database in Excel or Power BI Desktop

Notes from Video:

Table of Contents:

1. **SQL Database:**
    1) An SQL database is a relational database that uses SQL Code (Structured Query Language) to query or communicate with the database.
    2) Our class will use Power Query to generate the M Code that allows us to connect to an SQL Database and extract data from an SQL Database.
    3) Our class is not about learning how to write SQL Code.
        i. However, the six basic commands in an SQL SELECT Statement:
            1. **S = SELECT** = Select Columns (Fields, Attributes) and/or make Aggregate Calculations
            2. **F = FROM** = What Tables (Relations) & What Relationships
            3. **W = WHERE** = Filters Rows in a Table based on Conditions or Criteria
            4. **G = GROUP BY** = Group By Calculation to create unique list of items ad the make an aggregate calculation for each item
            5. **H = HAVING** = Filter Rows Based on Criteria for Group By Calculation
            6. **O = ORDER** = Sorting
        ii. An example of SQL Code to query two tables in an SQL Database:
            1. Query Goal is to:
                a. Select columns from the Fact Table, fTransactions, and select columns from the Dimension Table, dProduct.
                b. Create the Relationship between the Fact and Dimension Table
                c. Filter Rows in Fact Table Where Quantity >= 100
                d. Group By Product and calculate Total Net Revenue
                e. Filter Rows in the Group By step so that only rows where Net Revenue >= 100,000
                f. Sort Product Names Alphabetically.
            2. SQL Code looks like this:

```
SELECT dProduct.Product, SUM(Quantity*(1-RevenueDiscount)*RetailPrice) AS NetRevenue
FROM fTransactions
JOIN dProduct
ON dProduct.Product = fTransactions.Product
WHERE Quantity >= 100
GROUP BY dProduct.Product
HAVING SUM(Quantity*(1-RevenueDiscount)*RetailPrice) >= 100000
ORDER BY dProduct.Product;
```

4) **Query Folding**
    i. When you connect to an SQL database and you generate M Code using the User Interface, "Query Folding" will take place. Query Folding means that the M Code generated by User Interface will be sent back to the database, where the database optimizer can run the code to extract the data.
    ii. There is no way to tell if it is done more efficiently from the Power Query User Interface.
    iii. You would have to go to source and measure, like with tools like SQL Server Profiler.
    iv. You can prevent Query Folding with functions like Table.Buffer.
    v. Power Query Features like Custom Functions, Table.UnPivot function and some manually generated code will prevent Query Folding from happening.
    vi. Great article about Query Folding: http://blog.pragmaticworks.com/power-bi-checking-query-folding-with-view-native-query

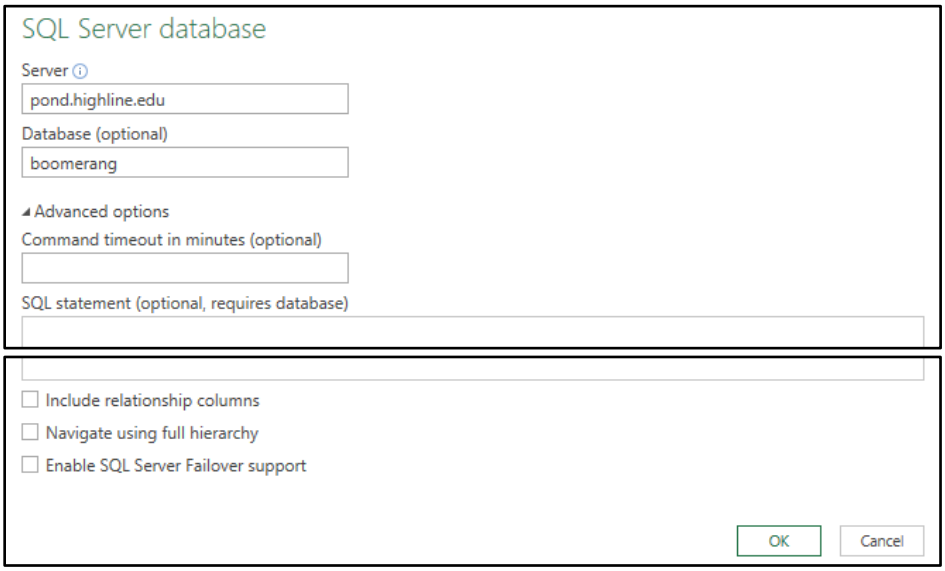
5) **Knowing Which Steps in Applied Steps list are sent back to the SQL Server Database for Query Folding: Use new feature in June 2018 update called "View Native Query"**
    i. View Native Query allows you to see whether a step in the Applied Step list has been sent back to the SQL Database to be performed at the database.
    ii. You can right-click any step in the Applied Steps list of steps for the query and point to "View Native Query". This allows you to see the SQL: Code that is being used by the SQL Server Database in the Query Folding.
    iii. If you right-click a Step in the Applied Steps list and the "View Native Query" is grayed-out, then you know that this step and all remining steps were not sent back to SQL Server Database.
    iv. Because we can use the "View Native Query" feature to determine which steps are sent back for Query Folding and at which step the Query Folding stops, this means the order in which we perform will determine how much of the query is sent back for Query Folding. If we can adjust the order of the steps in the query to send more of the query back for Query Folding, it is likely that the query will have a better overall performance.
    v. Great Article about this new feature (same link as in last paragraph: http://blog.pragmaticworks.com/power-bi-checking-query-folding-with-view-native-query

6) **Rule of Thumb** from both Ken Puls & Chris Webb (Power Query Masters) is that unless you are a masterful SQL Code writer, you should just use the Power Query User Interface and let Query Folding Occur.

2. **Import SQL Data into Excel using Power Query** :
    1) **Get Data From SQL Server Database using Power Query User Interface** :
        i. Credentials for SQL Database used in video:

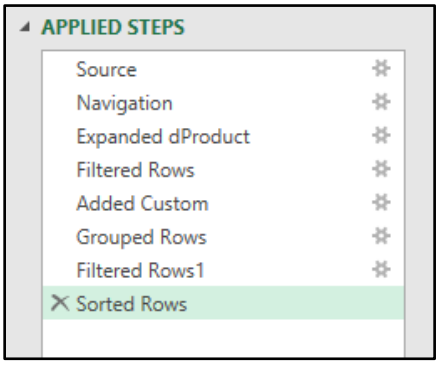
Server:	pond.highline.edu
Database:	boomerang
User:	excelisfun
Password:	ExcelIsFun!

        ii. To access an SQL Database in Power Query Use:
            1. Data Ribbon Tab, Get Data dropdown, From Database, From SQL Database:
        iii. **SQL Server Dialog Box**:
            1. **Server:** Server address.
            2. **Database:** Specific database you want to connect to.
            3. **Command timeout:** how long you want to wait for query to run
            4. **Include relationship columns:** If this is checked you will get extra columns with the related data
            5. **Navigating using full hierarchy:** If you check this, you will see everything from the database, rather than you the single database and its tables. This affects what you see in the next dialog box (Navigation dialog box where you select items to import).
            6. **Enable SQL Server Failover support:** Advanced SQL used to access servers that are part of a "Windows Server Failover Clustering". Links about this topic:
                a. https://www.youtube.com/watch?v=Q8Fimh4lcUg&feature=youtu.be
                https://docs.microsoft.com/en-us/sql/sql-server/failover-clusters/windows/windows-server-failover-clustering-wsfc-with-sql-server?view=sql-server-2017



iv. In the video after we connect to the SQL Database, we created these steps using the Power Query User Interface:

1. Applied Steps:



2. M Code:

Advanced Editor

ProductReportPQUserInterface

```
let
    Source = Sql.Database("pond.highline.edu", "boomerang", [CommandTimeout=#duration(0, 0, 1, 0)]),
    dbo_fTransactions = Source{[Schema="dbo",Item="fTransactions"]}[Data],
    #"Expanded dProduct" = Table.ExpandRecordColumn(dbo_fTransactions, "dProduct", {"RetailPrice"}, {"RetailPrice"}),
    #"Filtered Rows" = Table.SelectRows(#"Expanded dProduct", each [Quantity] >= 100),
    #"Added Custom" = Table.AddColumn(#"Filtered Rows", "Net Revenue", each [RetailPrice]*[Quantity]*(1-[RevenueDiscount])),
    #"Grouped Rows" = Table.Group(#"Added Custom", {"Product"}, {{"Total Net Revenue", each List.Sum([Net Revenue]), type number}}),
    #"Filtered Rows1" = Table.SelectRows(#"Grouped Rows", each [Total Net Revenue] >= 100000),
    #"Sorted Rows" = Table.Sort(#"Filtered Rows1",{{"Product", Order.Ascending}})
in
    #"Sorted Rows"
```

v. This is what the end results looks like:

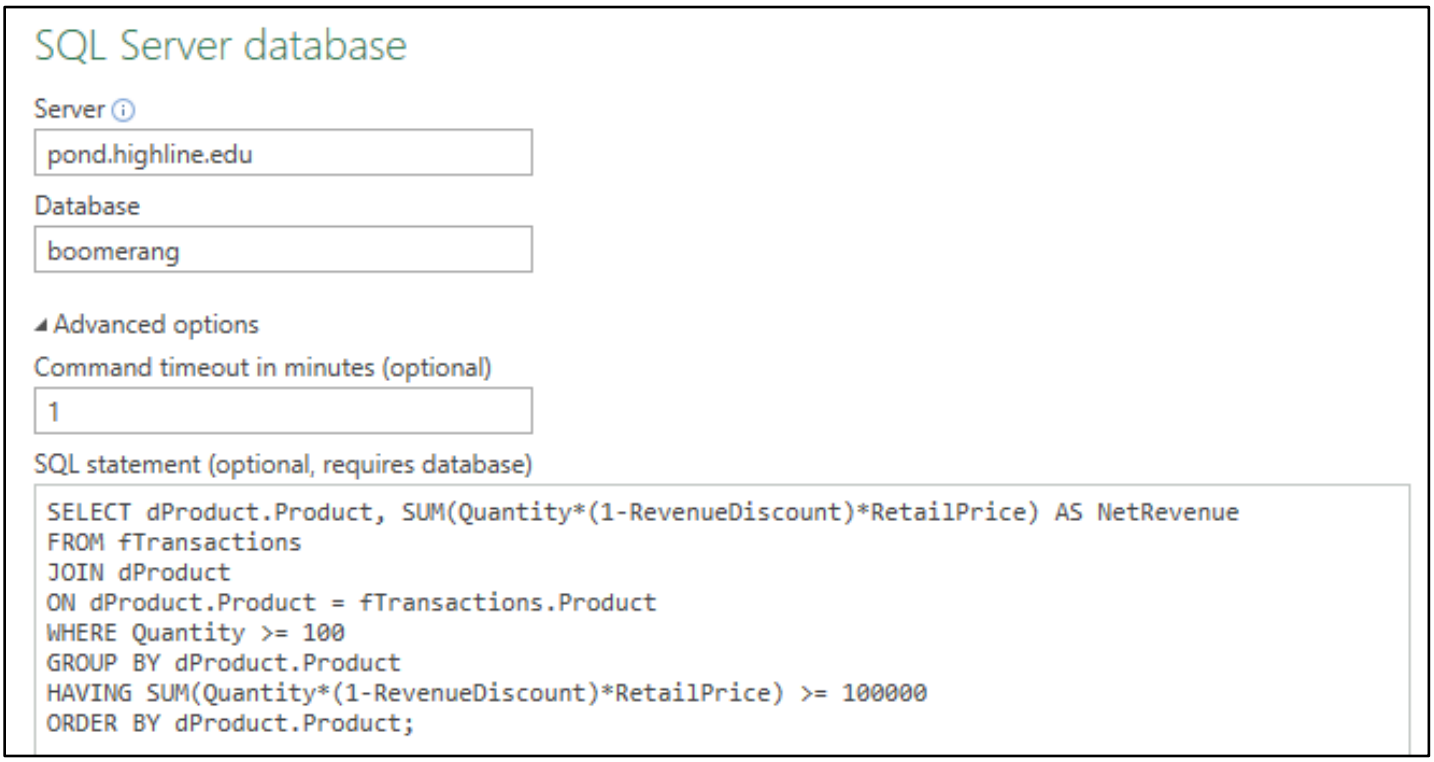
Product	Total Net Revenue
Alpine	11776864.07
Aspen	13958950.15
Bellen	28574273.3
Bower Aussie Round	25786269.36
Carlota	25235273.38
Crested Beaut	20983406.77
Darnell Tri Fly	4524925.493
Eagle	9274236.026
Fire Aspen	6589276.39
Fun Fly	36948046.09
GelFast	7416772.194
Manu LD	298465.776
Mejestic Beaut	16691525.19
Phoenix	9807279.089
Quad	92418009.38
Sunset	12248978.96
Sunshine	12666365.66
Sunspot	4141542.476
Yanaki	15145727.81

2) **Get Data From Microsoft SQL Server Database using SQL Code** :

i. To access an SQL Database in Power Query Use:

1. Data Ribbon Tab, Get Data dropdown, From Database, From SQL Database:

ii. Then we type our code into the SQL server database dialog box, as seen here:

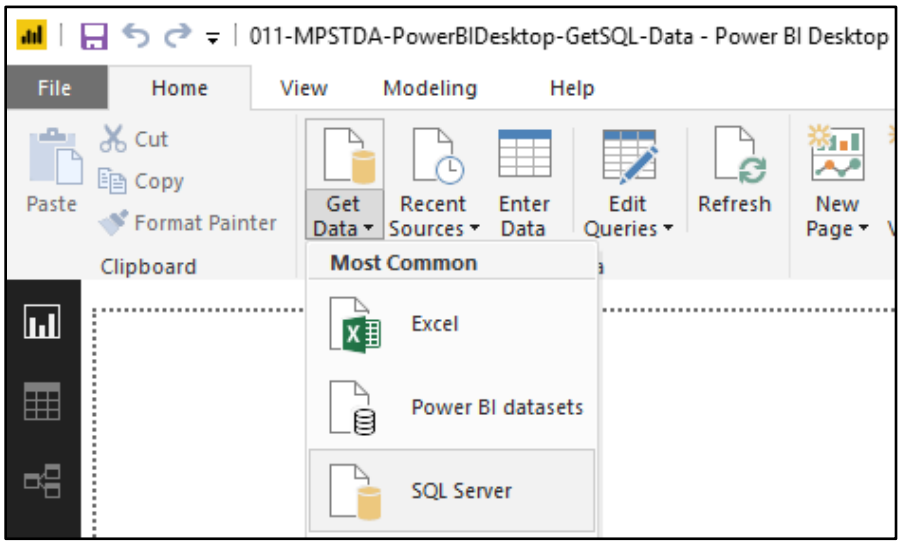


iii. The result from typing the SQL Code looks the same as using the Power Query User Interface, as seen here:

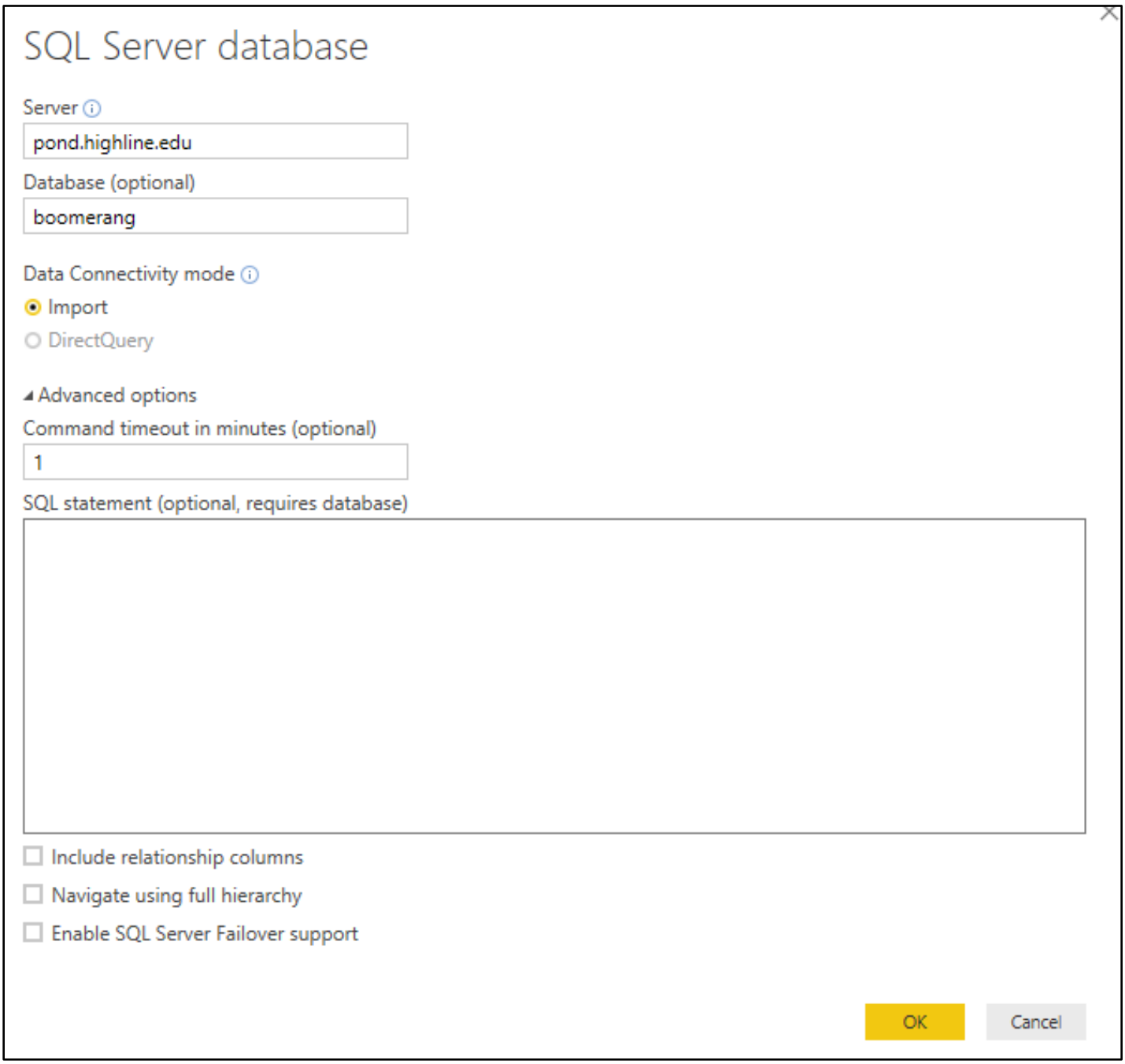
Product	NetRevenue
Alpine	11776864.07
Aspen	13958950.15
Bellen	28574273.3
Bower Aussie Round	25786269.36
Carlota	25235273.38
Crested Beaut	20983406.77
Darnell Tri Fly	4524925.493
Eagle	9274236.026
Fire Aspen	6589276.39
Fun Fly	36948046.09
GelFast	7416772.194
Manu LD	298465.776
Mejestic Beaut	16691525.19
Phoenix	9807279.089
Quad	92418009.38
Sunset	12248978.96
Sunshine	12666365.66
Sunspot	4141542.476
Yanaki	15145727.81

3. **Summary of whether to use Power Query User Interface or Type SQL Code**:
    1) Connecting to SQL Database and using Power Query User Interface:
        i. Steps we create are sent back to SQL Database (called Query Folding)
        ii. SQL Database is often more efficient (performance) than Power Query at importing and making transformations
        iii. Some Power Query Features, like Custom M Code, Table.UnPivot function cannot be understood by SQL Database and performed natively in Power Query
        iv. Table.Buffer Function can prevent Query Folding
    2) Connecting to SQL Database and typing Your Own SQL Code:
        i. Unless you are a master SQL coder, it is usually more efficient to use Power Query User Interface.
        ii. For convenience, some like to just write the code.

4. **Import SQL Data into Power BI Desktop using Power Query**
    1) In Power BI Desktop, we can import data from an SQL Database by opening the SQL server database dialog box using these clicks:
        i. In the Home Ribbon Tab, in the External data group, use the Get Data dropdown arrow, then click the SQL Server option, as seen here:



ii. This is what the SQL server database dialog box looks like:



iii. **Import Data or Direct Query?** The difference in this SQL Server database dialog box, and in Excel, is the option Data Connectivity Mode, as described here:

1. Import Data
    a. Data is imported into Power BI Desktop File.
    b. If data changes, you must refresh to get the new data.
    c. You can use all the features in Power BI Desktop.
    d. There is a data size limit of about 1 GB.
2. Direct Query:
    a. Power Query knows if the source you are connecting to will allow Direct Query and will offer the option to do a Direct Query or Import the data.
    b. Data is not imported and all queries in Power BI Desktop are sent back to Source Database.
    c. Main reason to use Direct Query is if you are connecting to very large databases (over 1 GB) or you want to be connected to the data set, live.
    d. There are significant drawbacks to using Direct Query such as:
        i. Performance may be slow because everything is sent back to source or multiple consumers of a Power BI Report may be querying simultaneously
        ii. Some features such as Time Intelligence Functions can not be used with Direct Query
    e. Direct Query requires trial and error to determine if this is a good option. Microsoft's web site gives guidance for determining if this option is efficient: https://docs.microsoft.com/en-us/power-bi/desktop-use-directquery