Microsoft Power Tools for Data Analysis #15

Comprehensive Introduction to Power Pivot & DAX

Notes from Video:

Table of Contents:

1) Standard PivotTable or Data Model PivotTable?

Choice between: Standard PivotTable & Data Model PivotTable

Standard PivotTable:

1. Have One Flat Table
2. Don't have Big Data
3. Standard Calculation in PT sufficient
4. Must manually add Number Format for each new Calculation
5. Can NOT re-use a Formula
6. For simple PivotTable Reports on a small data set, Standard PivotTables are great.

Data Model PivotTable:

1. Have Multiple Tables
2. Have Big Data
3. More Varied Calculations with DAX
4. Number Formatting can be added to formula
5. DAX Measures (Formulas) are created once, and can be re-used many times
6. For complex projects or Big Data, Data Model PivotTables are great.

2) Excel Power Pivot & Power BI Desktop?

Choice between: Excel Power Pivot & Power BI Desktop

Excel Power Pivot:

1. Power Query, Columnar Database, Relationships, DAX Formulas are almost identical in both.
2. PivotTable Report is what you want
3. Have Excel Worksheets to compliment Data Model PivotTable Reports that allow you freedom to:

a. Work in cells, not columns and tables
b. Have any of the other Excel features to compliment Data Model PivotTables

4. Familiar with Excel.
5. DAX Formula calculate more slowly in Excel because they are calculated with MDX, which uses only one processor at a time.
6. Hard to share Power Pivot Report.

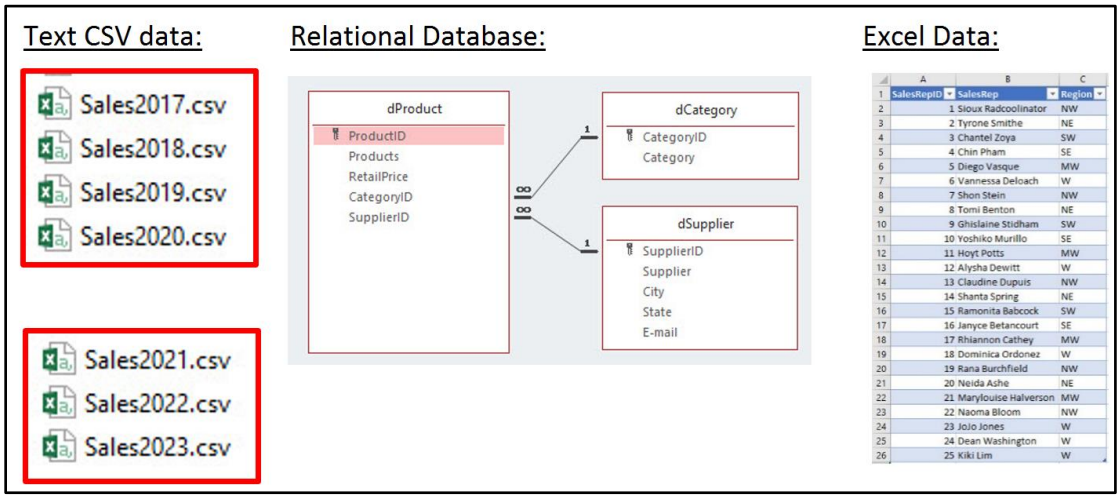
Power BI Desktop:

1. Power Query, Columnar Database, Relationships, DAX Formulas are almost identical in both.
2. More varied Visualizations and Reports
3. Visualizations and Reports are interactive (one can filter the other)
4. You can publish Visualizations and Reports, so they can be consumed on any device.
5. Table DAX Formulas can be part of the Data Model as a Table.
6. DAX Formulas calculate more quickly in Power BI because they are calculated using DAX which allows parallel processors to work on calculations. This matters for big data.
7. PBID gets updates each month & sometimes we get DAX Formulas or other features that aren't in Excel (why we can't create Data Model in PBID & open in Excel, but reverse is possible).

3) Power Query to Extract, Transform and Load Data to Data Model :

i. From what we learned earlier in class we performed these Importing, Transforming and Loading tasks:

1. Picture of Data Sources:



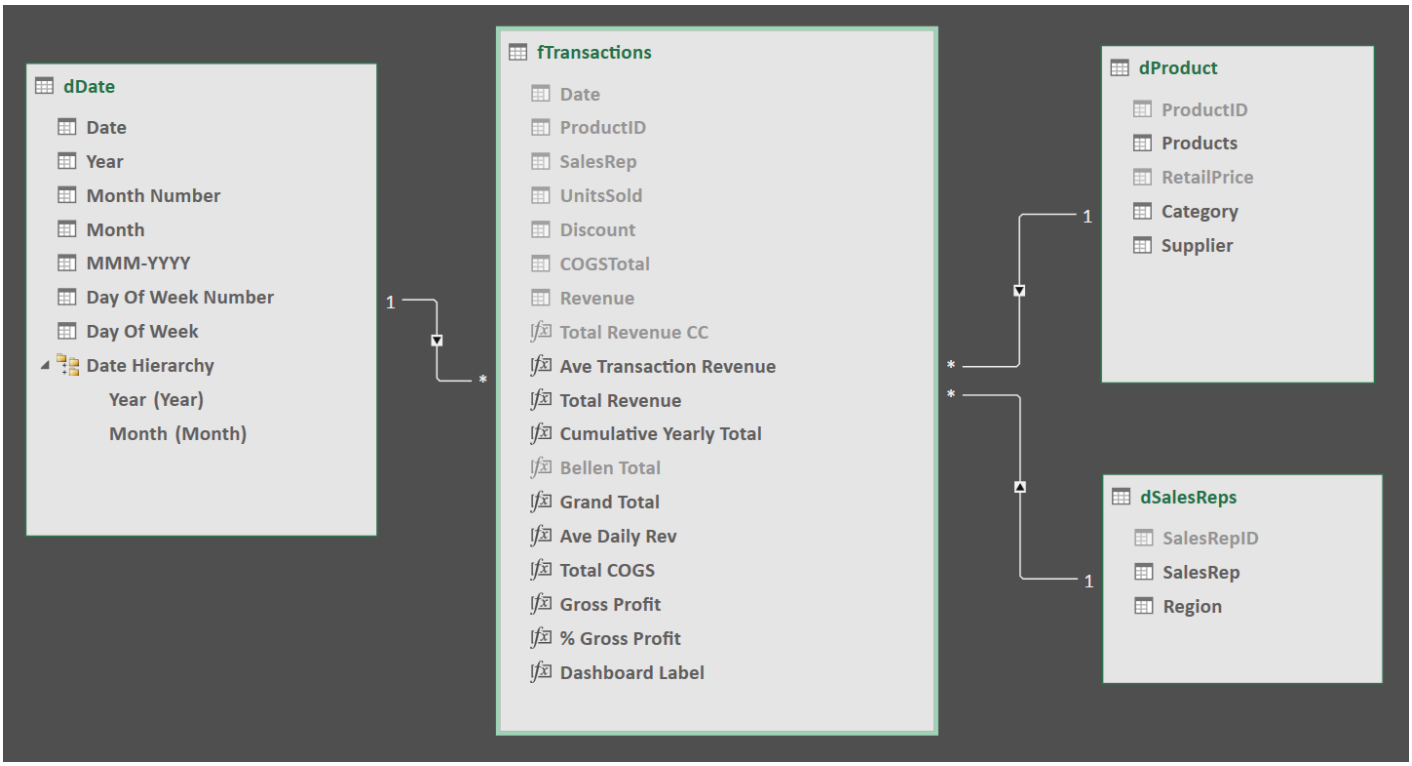
2. Import CSV Fact Table Data From Folder into Data Model
   i. We combined the text files.
   ii. Converted ISO Dates to Actual Dates
   iii. Rounded the COGS Column. By rounding a column with many extraneous decimals, it is likely that we reduced the number of unique values in the column and help reduce the size of the columnar database.
3. Import Access Database Product table into Data Model
   i. We imported the product table and expanded the columns from the related tables so we could have a single dimension table. By expanding the related tables we converted the Snow Flake Model to Star Scheme Model.
4. Import Excel Table into Data Model.

4) Build Relationships :

i. After we imported Tables into the Data Model, we created the Relationships with these steps:

1. In the Power Pivot Ribbon Tab we clicked the Manage Data Model button
2. In the Power Pivot Window, in the Home Ribbon Tab, in the View group, we clicked the Diagram View button.
3. We dragged and dropped Related Fields from the One Side to the Many Side.

ii. The finished Relationships (with Date Table and DAX Formulas) looked like this:



iii. Why Relationships exist in Data Model:

1. If our Data is in an Excel Sheet, Relationships are more efficient than using Helper Columns and the VLOOKUP Function.
2. Relationships allow us to see Multiple Tables in the PivotTable Field List so that we can:
   i. Drag and drop Fields from Dimension Tables as Filters for the Measures into the Row, Column, Filter and Slicer area of a PivotTable.
   ii. Drag and Drop Measures from our Fact tables into the Values area of a PivotTable.
3. Relationships allow DAX Formulas to access a table on the other side of a Relationship, whether we need to retrieve a single item from the One-Side or many items from the Many-Side.
4. A crucial reason that Relationships exist are so they can Transfer a Filter from a Dimension Table to a Fact Table to help facilitate faster DAX Measure Calculation Time.

5) Introduction to DAX Formulas: Measures & Calculated Columns :
    i. DAX stands for **D**ata **A**nalysis E**X**pressions
    ii. Types of DAX Formulas:
        1. Calculated Column (Helper Column with Row-by-Row Calculation) that delivers a single value or scalar value to each row in a table column.
        2. Measures (Formulas we use in Values Area of PivotTable or in other DAX Formulas) that delivers a single value of scalar value to a cell in a PivotTable.
        3. Table Formulas that deliver a table of values. Example: the DAX Table Function CALCULATETABLE.
    iii. Calculated Columns:
        1. Make Row-by-Row Calculations for a Table
        2. Used to Calculate:
            i. Numbers in a Fact Table, like Revenue in a Sales Transaction Table
            ii. Attributes in a Dimension Table, like Fiscal Year in a Date Table
            iii. Helper Columns
    iv. Measures:
        1. Where we use Measures:
            i. Drop into Values Area of PivotTable
            ii. Use in other DAX Formulas
        2. Goal of Measures:
            i. Measuring the health or performance of Entity
            ii. Examples:
                1. Total Revenue
                2. Ave Daily Revenue
                3. Customer Retention Rate
                4. % Crime Type in a Neighborhood
        3. Where we create Measures:
            i. Measure Grid in Power Pivot Window. The Measure Grid is the area below each table.
            ii. Create New Measure Dialog Box, which can be opened in Excel by going to the Power Pivot Ribbon Tab, in the Calculations group, click the Measure dropdown arrow, then click on New Measure.
            iii. Right-click Data Model Table in PivotTable Field List.
        4. Conventions for Measures:
            i. The Convention is to create Measure below the Fact Table. This is because most of the time our Measures are working on Numbers from the Fact Table.
            ii. When we create a Measure, it is assigned to a particular Table and in the PivotTable Field List, the Measure will show up as an item in the Table's Field List.
    v. Conventions for DAX Formulas:
        1. For Calculated Columns and Measures, we must type our formula out in the Formula Bar.
        2. When you refer to a Field from a table (Column from a table), use the Table Name followed by the Field Name in Square Brackets. Example: **Table[Field Name]**
        3. When you use a Measure in a DAX Formula, use Measure Name in Square Brackets. Example: **[Measure]**
            i. Why? So we know when we see a Measure in a formula we know that a hidden CALCULATE Function is present, and may have an impact on whether or not Context Transition occurs (more in later videos)

vi. Introduction to how DAX Formula are Calculated: Row Context & Filter Context :
    1. Row Context
        i. Formulas in DAX Calculated Columns:
            1. Are automatically copied down.
            2. The Column References in the formula are able to "see" the values for each row in the table and make the correct calculation for each row.
    2. Filter Context
        i. DAX Measures & Filter Context:
            1. For a Dimension Table Filter:
                i. Criteria from Row, Column, Filter & Slicer Areas filter the underlying Dimension Tables
                ii. The Filter Flows Across the Relationship
                iii. The Fact Table is filtered down to a smaller size
                iv. DAX Measure calculates over a smaller column
            2. For a Fact Table Filter:
                i. For a Measure like, DISTINCTCOUNT(fSales[ProductID]), the underlying Fact Table is filtered down to a smaller size, and then the Measure calculates over a smaller column.
    3. DAX Formula Calculation Process:
        i. DAX Measures use Filter Context
        ii. DAX Calculated Columns use Row Context

6) DAX Calculated Column using the DAX Functions RELATED and ROUND to calculate Line Revenue :
    i. Steps to create Calculated Column in the video:
        1. Our Goal is to calculate the Line Revenue for each Transaction in the Fact Table.
        2. To create a Calculated Column, double-click the Add Column Header to the right of the last column in the Fact Table to give the Calculated Column a name, then hit Enter. Then type your formula in the Formula Bar.
        3. Final Calculated Column formula is:

```
=ROUND(RELATED(dProduct[RetailPrice])*fTransactions[UnitsSold]*(1-fTransactions[Discount]),2)
```

        4. Row Context in a table = even though we use column references in our formula, the formula sees each row in the table and can pull out the correct number for each row.
        5. RELATED is the function that we use in DAX to look up an item from the Many Side of a Relationship to retrieve an item from the One Side of the Relationship.
        6. ROUND is the same as in Excel.

7) Row Context: How DAX Calculated Columns (and Iterator Functions) are Calculated :
    i. When we create a Calculated Column (or use an Iterator Function), the formula uses full column references and the formula is the same for each row in the table. But internally the formula can see the values in each row and make the correct row-by-row calculation by accessing the values that sit in each record for each row. This process is called "Row Context". It is similar to how Table Formula Nomenclature works in Excel Tables, and how Custom Columns work in Power Query. In all 3 situations, the formula looks the same in each row, but the formula can access the values from each row and make the correct row value calculation.

8) We do not want to use Calculated Column results in PivotTable using Implicit Measures :
    i. After we create a Calculated Column for an amount like Revenue, we do not want to drag and drop the Column (Field) from a PivotTable Field List into the Values Area of PivotTable because it creates an Implicit Measure that is not an efficient (details ahead in the section on Explicit vs. Implicit Measures).

9) DAX Measure to add results from Calculated Column, using DAX SUM Function :
    i. Steps to create Measure to add revenue values created by Calculated Column:
        1. Click in the Measure Grid.
        2. In the Formula Bar type the Measure Name.
        3. Between the Measure Name and formula, you type the assignment operator Colon & Equal Sign.
        4. The Final Measure looks like this:

```
Total Revenue CC:=SUM(fTransactions[Revenue])
```

10) Number Formatting for DAX Measures :
    i. With the cell selected in the Measure Grid, you can apply Number Formatting from the Home Ribbon Tab, Formatting group.
    ii. This Number Formatting is applied any time you use the Measure
11) Data Model PivotTable :
    i. You can create a PivotTable based on the Data Model from two places:
        1. In the Power Pivot Window, in the Home Ribbon Tab, PivotTable button
        2. Create PivotTable dialog box in Excel, with the “Use this Workbook’s Data Model” dialog button selected.
12) Explicit DAX Formulas rather than Implicit DAX Formulas :
    i. Implicit DAX Measures
        1. Implicit = Drag and drop field into Data Model PivotTable/Power BI Values Area and the Data Model makes a Read Only Measure for you
        2. MUCH Less Control when you use Implicit:
            i. Must Re-apply Number Format for each new Calculation
            ii. May get a buildup of unnecessary Implicit Measures
            iii. Can NOT re-use Measure in PivotTable Field List
            iv. Can't Edit Measure Name or Formula
            v. Can't Apply Number Formatting
            vi. In Power BI Desktop, if you use Implicit Measures and Publish Report to powerbi.com, then if you download into Excel, the Implicit Measures do not show up.
        3. Use Implicit Measures ONLY:
            i. For Simple PivotTable Report
    ii. Explicit DAX Measure
        1. Explicit = You create the formula, choose the name, add the Number Formatting and use it over and over.
        2. You create formula in Measured Grid, or Measure dialog box
        3. MUCH More Control when you use Explicit:
            i. You choose what functions go into your Measure.
            ii. You can name the Measure.
            iii. You can apply Number Formatting to the Measure.
            iv. You can use the formula over and over.
            v. Won't have a bunch of extra Measures created by right-click, "Summarize Values By".
            vi. Explicit Measures are present in powerbi.com downloaded Excel Data Models (more later).
13) Show Implicit Measures :
    i. Dragging a Field from a PivotTable Field List into the Values Rea of a PivotTable will automatically create a hidden Implicit Measure in the Data Model. To unhide the Implicit Measures, in the Power Pivot Window, go to the Advanced Ribbon Tab and click the “Show Implicit Measure” button.

14) Filter Context (First Look) How DAX Measures are Calculated :
    i. When you drag a Measure to the Values Area of a Data Model PivotTable, the Measure sees the "Filter Context".
        1. "Filter Context" simply means that the Measure can "see" all the criteria/conditions/filters from the PivotTable Row Area, Column Area, Filter Area and Slicer Area (in Power BI it is the same: Measure sees all the filters that are applied).
        2. When a Measure is dropped into the Values Area of a PivotTable that has the Product Name from a dimension table dropped in the Row Area, the Columnar Database and the Power Pivot Engine will filter the dProduct table down to a single row, and in turn this filter flows across the relationship and filters the Fact Table down to just the records for the given product, in this way the Fact Table is made much smaller before the DAX Formula has to make the calculations.
        3. More about Filter Context later in this class.
15) Drag Columns from Fact Table or Dimension Table? :
    i. If you have a Relationship between a Fact Table and a Dimension Table and there are unmatched items on the Many Side of the Relationship:
        1. When you use a Dimension Table Column as a filter in a PivotTable or visualization, a blank will show up in the PivotTable indicating that there is an unmatched item.
        2. When you use a Fact Table Column as a filter in a PivotTable or visualization, you will get a unique list of items from that column.
    ii. For Efficient Data Model PivotTable Reporting Results, it is best to:
        1. Hide all Columns (Fields) in the Fact Table.
        2. List Only Measures in the Fact Tables.
        3. Use Columns for filtering from the Dimension Tables.
    iii. The Rule is: Use the Dimension Table Fields, NOT the Fact Table Fields.
        1. Why? When Columns from Dimension Tables are used as filters for the Measures:
            i. DAX Formulas work more efficiently on the smaller Dimension Tables
            ii. DAX Formulas are designed to have the Dimension Tables filtered and then have the filter flow through the Relationship to the Fact Table to create a smaller set of values for the DAX Formula to use.
16) Hiding Columns, Tables and Measure from Client Tool :
    i. If you have Columns, Tables or Measures that you do not need in the Reporting Area, hide them so that they do not interfere with an "easy to use" Reporting Area. This makes the Reporting side easier for the user.
    ii. To hide a Data Model element, right-click and point to "Hide From Client Tool".
    iii. The "Show Hidden" button in the View group in the Home Ribbon of the Power Pivot Window will show or hide the hidden elements (Columns, Tables, Measures) in the Data Model.
17) Use Power Query to Refine Data Model :
    i. We can go back to the Power Query Editor at any point and refine the Model.
    ii. In our Model, we did not need the Supplier ID and Category ID Columns in the Product Table so we went back to the Power Query Editor and removed the column.

18) SUMX Function (Iterator Function). DAX Measure for Revenue using the SUMX Function to simulate Calculated Columns in DAX Measures :

i. SUMX DAX Function for calculating Revenue.
    1. The SUMX Function simulates what we do in the two-step process of 1) building a Calculated Column to calculate the Revenue for each row in the Table, and then 2) using the SUM Function in a Measure to add the values from the Calculated Column.
    2. The SUMX Function is an iterator function, where iterator just means that the SUMX can iterate over a table and calculate a value for each row in the table (using Row Context), and then the SUMX takes the calculated values and adds them to get a single total.
    3. You can think of the SUMX Function as a super-charged Array Formula that can create the full helper column in a single cell and then add the result.
    4. Final SUMX Formula looks like this:

```
Total Revenue :=
SUMX(fTransactions,ROUND(RELATED(dProduct[RetailPrice])*fTransactions[UnitsSold]*(1-fTransactions[Discount]),2))
```

ii. SUMX is just one of many “Iterative DAX Functions”, that can create a Row Content for a table, calculate a value for each row, and the make an aggregate calculation.

iii. There are other “X” Iterative functions like that iterate and then aggregate:
    1. AVERAGEX
    2. RANKX
    3. MINX
    4. SUMX
    5. And more…

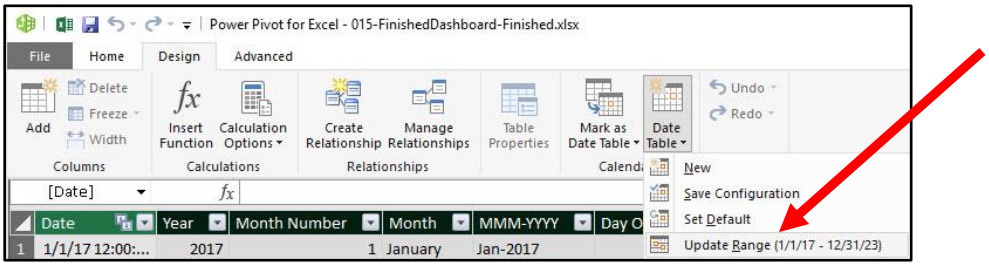
iv. There are other Iterative functions that do not have an “X” in the name like and do not aggregate:
    1. FILTER
    2. ADDCOLUMNS
    3. And more…

19) Compare Calculated Column to Measure for Total Revenue Calculation: :

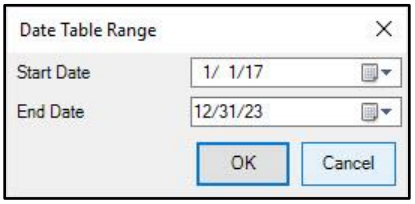
i. We have two options when we have to calculate Total Revenue:
    1. Create Calculated Column and then create a Measure to sum the Calculated Column Numbers
        i. The Calculated Column numbers are stored in the Columnar database as a column for the Fact Table.
        ii. The Calculated Column DAX Formula is calculated or evaluated only when the table is refreshed.
        iii. If we chose to use a Calculated Column the in-RAM Database size is increased.
        iv. If you are concerned with the size of the in-RAM Memory Database, you might want to consider using SUMX and a Measure.
    2. Create a single Measure using SUMX that both calculates the Row-by-Row revenue numbers and then adds the individual revenue numbers to get a total.
        i. The SUMX function calculates the revenue numbers Row-by-Row internally in the formula and then the SUMX adds the number to get a total.
        ii. SUMX has to calculate or generate the numbers every time the Measure in dropped into a PivotTable or when a criterion or condition or filter is changed in the Row, Column, Filter or Slicer area of the PivotTable.
ii. Considerations when choosing between the two:
    1. If the calculation time is too slow when the Measure is dropped into the PivotTable or a condition is changed, then you might want to consider using a Calculated Column to calculate the Row-by-Row Revenue Numbers.
    2. The convention is to Use Measures rather than Calculated Columns. However, according to Marco Russo and Alberto Ferrari, until you reach about 100 million rows of data, either method is fine.

20) Why We Need a Date Table. Why we do NOT use the Automatic Grouping Feature for a Data Model PivotTable :
    i. In a Data Model we need a Date Table (Calendar Table) for these reasons:
        1. Date Table Provides Date Attributes that we can use as Filters for our Reports
        2. We want DAX Formulas to work on (iterate over) a unique list in the first column of the Dimension Table, rather than a full column in a Fact Table, like for a Daily Average calculation.
        3. We need the benefit of having a Dimension Table that can filter the Fact Table (so DAX Formulas can calculate over a smaller set of numbers)
        4. Time Intelligence Functions (like TOTALYTD, SAMEPERIODLASTYEAR and DATEADD) require a Date Table.
    ii. Date Dimension Table must have these characteristics:
        1. Unique List in First Column for every possible day for every given year in the Date Column in the Fact Table (so Time Intelligence Functions work correctly, like for DATEADD function)
        2. Must Mark as Date Table (So Automatic Grouping Feature in Data Model will be disabled).
    iii. Why you do NOT want to use the "Automatic Grouping Feature" in a Data Model, but instead you want a Date Dimension Table:
        1. If you do not have a Date Table, you can not use Data Intelligence Functions (like: TOTALTYD, SAMEPERIODLASTYEAR, DATEADD).
        2. You loose ability to filter a Dimension Table and have the filter flow through the Relationship to the Fact Table for faster DAX Formula calculation times.

21) Build an Automatic Date Table (Calendar Table) in Excel Power Pivot & then build Relationship. :
    i. Click in Date column of Fact Table.
    ii. In the Design Ribbon Tab, in the Calendar group, click Date Table drop-down, then click New.
    iii. IMPORTANT NOTE about Power Pivot Automatic Date Table:
        1. You must manually refresh the Data Table when new data is added with new dates.
        2. The Power Pivot Automatic Date Table does NOT automatically update.
        3. Steps to update:
            i. In the PowerPivot for Excel Window, in the Design Ribbon Tab, in the Calendar group, from the Date Table drop-down arrow, click on Update Range, as seen below:



            ii. In the Date Table Range dialog box, edit the range of dates, as seen below:



22) Introduction to Time Intelligence DAX Functions:
    i. We looked at TOTALYTD function to calculate a Running Total (Cumulative Total) for each year.
    ii. Time Intelligence Functions depend on having a Proper Date Table.
    iii. Other Time Intelligence Functions that we will see later in this class: SAMEPERIODLASTYEAR and DATEADD

23) Introduction to CALCULATE Function: Function that can “see” Data Model and can change the Filter Context. Also see the ALL and DIVIDE DAX Functions. Create formula for “% of Grand Total” Also learn about Context Transition and the Hidden CALCULATE on all Measures. :

1. The CALCULATE Function and the CALCULATETABLE Function Can see the entire Star Schema Data Model and Can Change The Filter Context by applying filters on columns internally in the formula.
2. Examples:
    i. The formula below will calculate the Total Revenue for the Bellen Product in every cell in the PivotTable. The filter of “Bellen” in the formula will override the filter from coming from the Row Area of the PivotTable:

```
Bellen Total:=CALCULATE([Total Revenue],dProduct[Products]="Bellen")
```

    ii. This formula will remove all filters that are applied to the fTransactions Table and return the Grand Overall Total for Revenue. The ALL Function is the function we use in the filter argument of CALCULATE that will “Remove All Filters from the fTransaction Table.

```
CALCULATE([Total Revenue],ALL(fTransactions))
```

    iii. Formula to calculate the “% of Grand Total Revenue”. The Divide Function

```
% of Grand Total:=DIVIDE([Total Revenue],CALCULATE([Total Revenue],ALL(fTransactions)))
```

3. CALCULATE & Context Transition:
    i. Context Transition happens when CALCULATE converts all available Row Contexts into an “Equivalent Filter Context” and then propagates them across the relationships.
        1. Context Transition said differently: Convert Row Context to Filter Context by taking the Row Condition and sending it across the Relationship so that it can filter the Table on the other side of the Relationship.
    ii. Context Transition happens when you wrap CALCULATE around an aggregate formula or when the automatic hidden CALCULATE surrounds a Measures.
    iii. This is important because if you use an aggregate function to add the results for each row in a Calculated Column, you will get the grand overall, rather than the total for each item in the relationship for each row.
        1. For example, if you use the aggregate function, SUM(fTransactions[Revenue]), in a Calculated Column in a Sales Rep Dimension Table with the goal of adding the sales for each Sales Rep, you will not get the total for each Sale Rep, but instead you will get the Grand Overall Total for all Sales Reps.
        2. The reason that the aggregate function, SUM(fTransactions[Revenue]), gives you a Grand Overall Total is because there is no Filter Context for each row in a Calculated Column (or an Iterative Function).
        3. The solution is to bring the Sales Rep Criteria for each row (Row Context) into the Filter Context by wrapping the CALCULATE Function around the aggregate calculation. In this way the SUM can “see” the Sales Rep Row Context, as a Filter Context, and the Sales Rep Dimension Table is filtered down to just the particular Sale Rep for the given row, the filter can flow across Relationship, and the Fact Table will be filtered down to just the numbers for the given row Sales Rep.
4. Hidden CALCULATE Function for all Measures
    i. Every Measure has a Hidden CALCULATE Function wrapped around it so that Measures will automatically converts all available Row Contexts into an “Equivalent Filter Context” and then propagates them across the relationships.

24) DAX Formula Benefits :
    i. DAX Measure can have Number Formatting Applied.
    ii. DAX Measures can be Re-Used.
    iii. DAX Formulas work efficiently on Big Data.
    iv. DAX Formulas can make more varied calculations than in a Standard PivotTable.

25) Example of DAX Formula that is easier to author than if we tried to do it with a Standard Pivot Table or Array Formulas :
    i. In the video we saw that calculating the Average Daily Sales from a Transaction Line Item Fact Table can be harder to do in a Standard PivotTable or with Excel Array Formulas.
26) AVERAGEX Function (Iterator Function) to calculate Average Daily Revenue :
    i. The DAX Measure we used for calculating Average Daily Revenue is listed below.

```
Ave Daily Rev:=AVERAGEX(dDate,[Total Revenue])
```

27) Filter Context (Second Look) AVERAGEX Iterator Function :
    i. This formula uses the Iterator AVERAGEX DAX Function. In the first argument the Date Dimension Table is listed and in the second argument the Measure for Total Revenue is listed. Because the Measure has a hidden CALCULATE Functions wrapped around it, Context Transition occurs and the Row Context of each daily date flows into the Total Revenue Measure, and, with the External PivotTable Filter Context of the Year and Sales Rep criteria flowing into the Measure also, the correct Daily Totals for each Year, Sales Rep and Day the Measure is calculated for each day, and then the AVERAGEX Function calculate the average.
28) Build Dashboard. Create multiple DAX Formulas. Create Multiple Data Model PivotTables and a Data Model Chart :
    i. We used Data Model PivotTables
29) Create Measures for Gross Profit :
    i. Formula as seen in video:

```
Total COGS:=SUM(fTransactions[COGSTotal])

Gross Profit:=[Total Revenue]-[Total COGS]

% Gross Profit:=DIVIDE([Gross Profit],[Total Revenue])
```

30) Continue making more Data Model PivotTables :
    i. We used Data Model PivotTables
31) Make Data Model Pivot Chart :
    i. We are allowed to create Line Charts from Data Model PivotTables
32) Conditional Formatting for Data Model PivotTable :
    i. When you add Conditional Formatting to a Data Model PivotTable that contains a DAX Formula, be sure to use the Smart Tag Icon to make sure the Conditional Formatting is applied to the correct Conditional Calculation.
33) DAX Text Formula for title of Dashboard :
    i. Here is the DAX Formula we created for the label in our Dashboard:

```
Dashboard Label:="Boomerang Inc. Metrics for the Years: "&MIN(dDate[Year])&" to "&MAX(dDate[Year])
```

34) CUBE Function to Convert Data Model PivotTable to Excel Spreadsheet Formulas :
    i. Steps to covert a Data Model PivotTable to CUBE Functions:
        1. In the PivotTable Tools Analyze Ribbon Tab, in the Calculations group click the dropdown for OLAP Tools, then click on Convert to Formulas.

35) Adding New Data and Refreshing :
    i. After we added new CSV files to our Start Folder, we used the Refresh All Button in the Excel data Ribbon Tab to update Power Query and Power Pivot.
    ii. The Refresh All button does not update the Automatic Data Table.
36) Update Excel Power Pivot Automatic Date (Calendar) Table :
    i. In the Design Ribbon Tab, in the Calendar group, click Date Table drop-down, then click Update Range
37) How to Double Check that a DAX Formula is yielding the correct answer? :
    i. To Double check our DAX Formulas, we can pull data from the Power Pivot Data Model into an Excel Sheet using the Existing Connections feature in Excel (as explained three sections ahead), then after you pull the target data, create Excel Spreadsheet formulas to check your work.
38) DAX Table Functions. See CALCULATETABLE DAX Function :
    i. DAX Table formulas and Functions:
        1. Excel Power Pivot DAX Table Formulas can deliver tables to:
            i. Other DAX Formulas
            ii. NOT to the Data Model
            iii. To an Excel Worksheet
        2. Power BI Desktop DAX Table Formulas can deliver tables to:
            i. Other DAX Formulas
            ii. Data Model. Example: Deliver a Date Table
        3. Examples of Table Functions:
            i. CALCULATEDTABLE (as seen in this video)
                1. CALCULATETABLE can “see” the Data Model and filter a Table based on Columns from any table in the Star Schema Data Model.
            ii. FILTER (seen later in class)
            iii. ALL (seen later in class)
            iv. VALUES (seen later in class)
39) DAX Studio to visualize DAX Table Functions, and to efficiently create DAX Formulas :
    i. DAX Studio is a free add-in for Excel and as a stand alone product to work with Power BI.
    ii. Search Google for “DAX Studio”. And then download and install.
    iii. In DAX Studio, we can:
        1. Build Formulas
        2. Visualize DAX Table Functions.
        3. Format our DAX.
        4. Time our Formulas.
40) Existing Connections to import data from Data Model into an Excel Sheet. :
    i. To pull data from the Power Pivot Data Model:
        1. Build your DAX Table Formula using DAX Studio.
        2. Copy the DAX Code from DAX Studio.
        3. In Excel go to the Data Ribbon Tab, then in the Get and Transform group click the Existing Connections button.
        4. In the Existing Connections dialog box, click the Table tab, then select a table from the data model or a query for one of the dimension tables, then click Open
        5. With the table delivered to the Excel Sheet, Right-click the Table, then in the list of options, point to Table, then from the second list of options, click on Edit DAX.
        6. In the Edit DAX dialog box, from the Command Type dropdown, select DAX
        7. In the Expressions textbox, Paste your DAX Table Formula.
        8. Click OK.

41) Overview of Steps to Build Power Pivot Data Model & Dashboard :
    i. Step #1: Power Pivot: Use Power Query to Extract & Transform Data and Load to Power Pivot Columnar Database in the Data Model
    ii. Step #2: Power Pivot: Create Relationships between Tables
    iii. Step #3: Power Pivot: Create DAX Formulas: 1) DAX Calculated Columns & 2) DAX Measures
    iv. Step #4 in Power Pivot: Hiding Fields or Tables from Client Tool
    v. Step #5 Power Pivot: Refine Data Model in Power Query
    vi. Step #6 Power Pivot: Create Date Table
    vii. Step #7 Power Pivot: Create Reports and Dashboard
    viii. Step #8: Power Pivot: Use DAX Studio or Existing Connections to verify DAX Formulas
42) Important Keyboards :
    i. Open Power Pivot Data Model: **Alt, B, M**
    ii. Create PivotTable: **Alt, N, V**
43) BLANK value in Power Pivot :
    i. BLANK in DAX:
        1. BLANK represents Empty Cell, Missing Value, Null (Database "unknown value")
        2. BLANK = Empty Cell, Missing Value
        3. BLANK <> Zero Length Text String
        4. BLANK is not an Error