INTRO TO “POWER EXCEL”

AI Quest

THE "POWER EXCEL"

These are Excel's **Business Intelligence** tools, all of which are available directly in Excel (*provided you have a compatible version*); **no additional software is required!**

RAW DATA

Flat files (csv, txt), Excel tables, databases (SQL, Azure), folders, streaming sources, web data, etc.

POWER QUERY

*(aka "**Get & Transform**")*

Connect to sources, import data, and apply shaping and transformation tools (ETL)

DATA MODEL

Create table relationships, add calculated columns, define hierarchies and perspectives, etc.

POWER PIVOT & DAX

Explore and analyze the entire data model, and create powerful measures using Data Analysis Expressions (DAX)

AI Quest

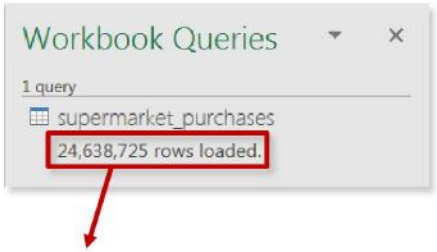
"THE BEST THING TO HAPPEN TO EXCEL IN [illegible]"

- **Import and analyze MILLIONS of rows of data in Excel**
  - *Access data from virtually anywhere (database tables, flat files, cloud services, folders, etc.)*
- **Quickly build models to blend and analyze data across sources**
  - *Instantly connect sources and analyze holistic performance across your entire data model*
- **Create fully automated data shaping and loading procedures**
  - *Connect to databases and watch data flow through your model with the click of a button*
- **Define calculated measures using Data Analysis [illegible]**

**Quote by Bill Jelen (aka "Mr. [illegible]"*

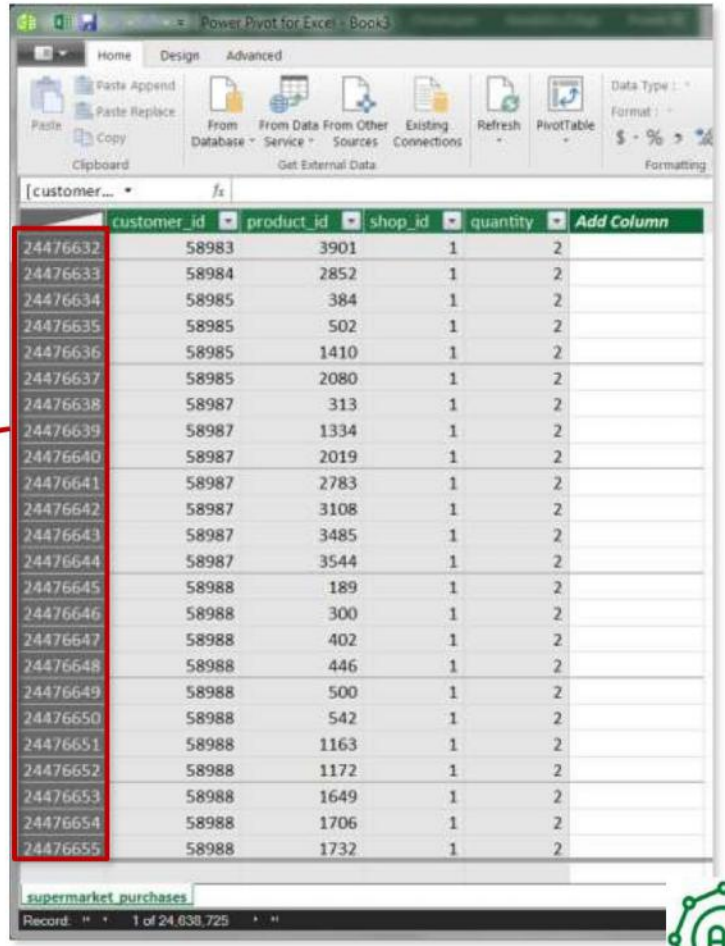
AI Quest

#1: IMPORT & ANALYZE MILLIONS OF ROWS



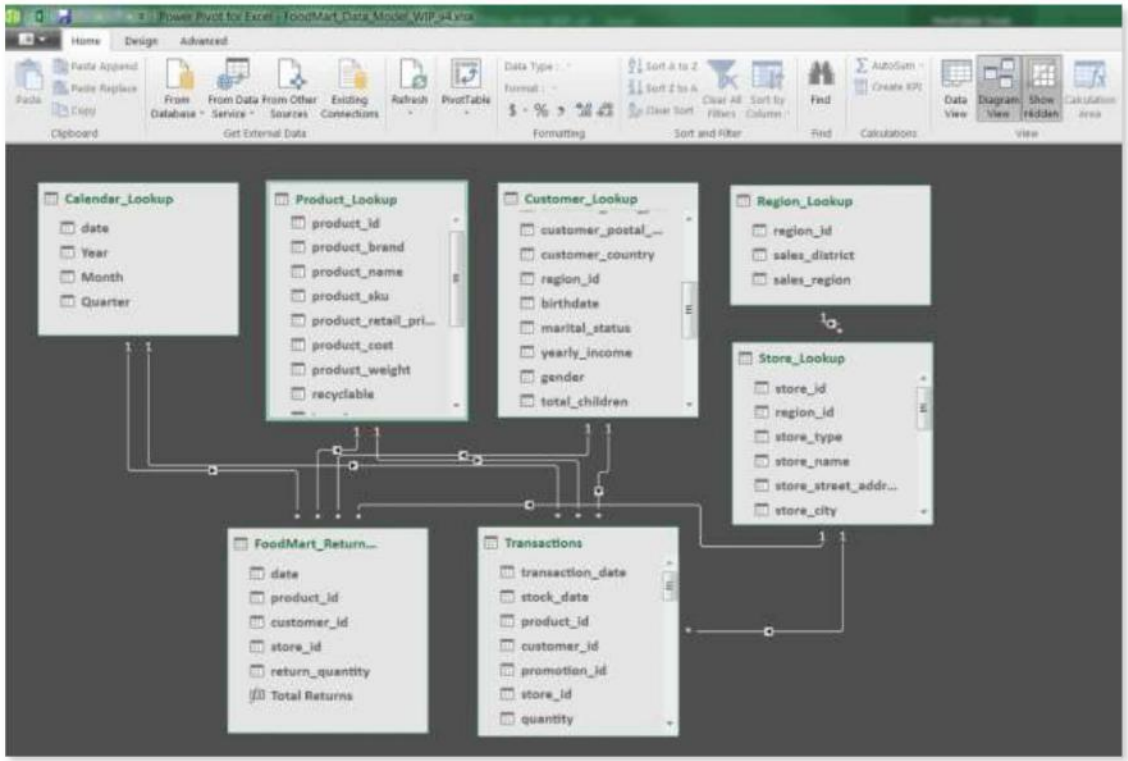
When was the last time you loaded
25,000,000 rows of data into Excel?

When you connect to data with **Power Query** and load it to Excel's **Data Model**, the data is compressed and stored in memory, NOT in worksheets (*no more 1,048,576 row limit!*)



AI Quest

#2: BUILD DATA MODELS TO BLEND SOURCES

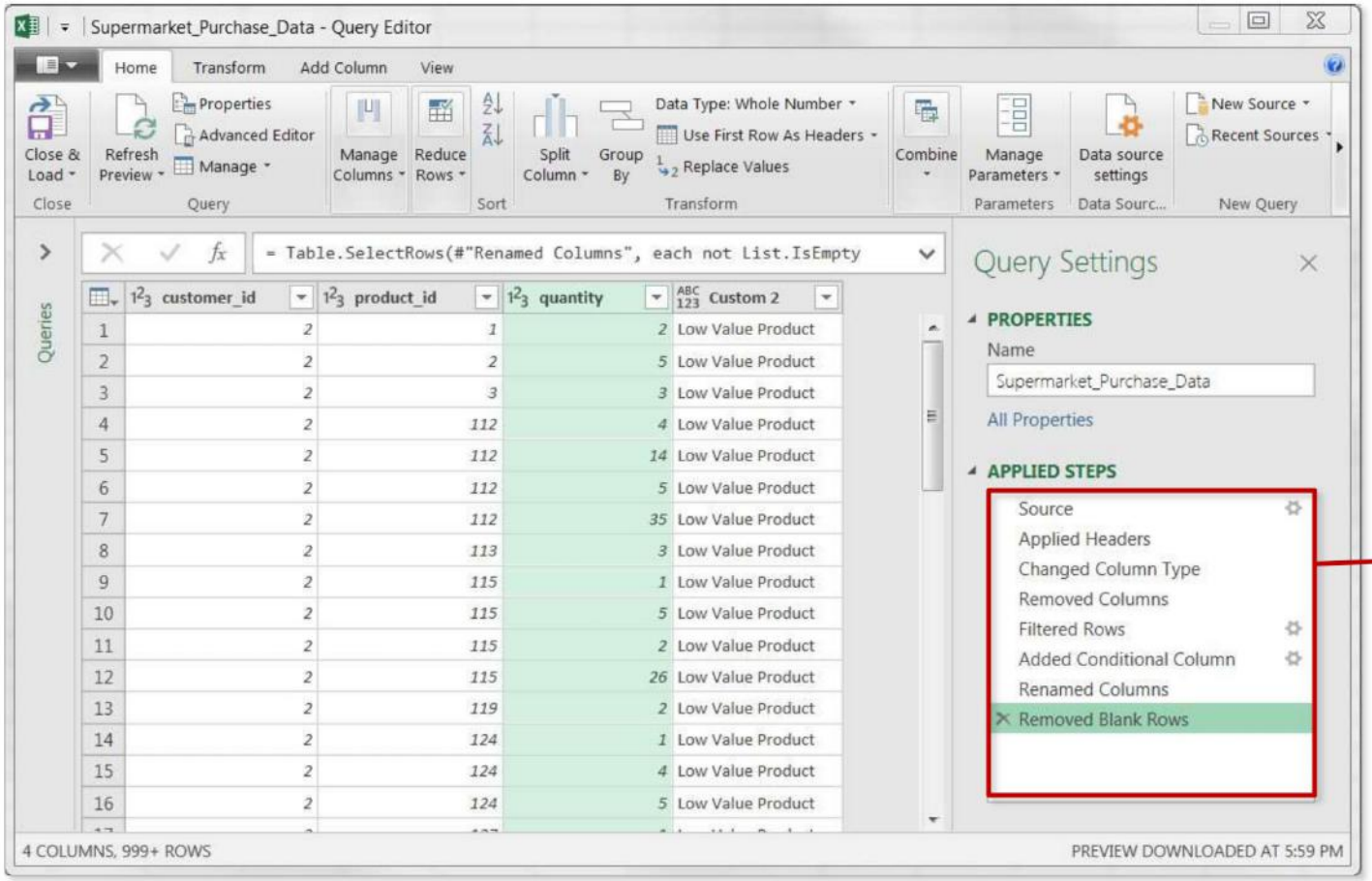


This is an example of a Data Model in "**Diagram View**", which allows you to create connections between tables

Instead of manually stitching tables together with cell formulas, you create ***relationships*** to blend data based on common fields

AI Quest

#3: AUTOMATE YOUR DATA

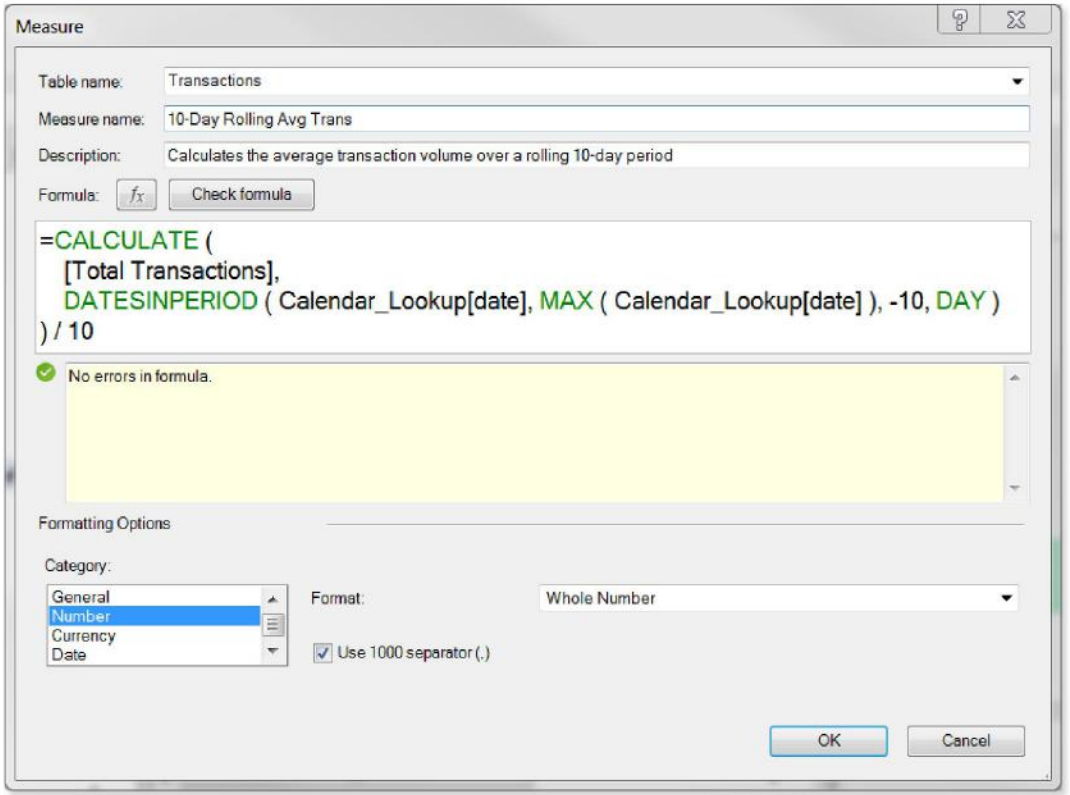


With Power Query, you can **filter**, **shape** and **transform** your raw data before loading it into the data model

Each step is **automatically recorded** and **saved with the query**, and applied whenever the source data is refreshed – like a macro!

AI Quest

#4: CREATE POWERFUL MEASURES



Measures are flexible and powerful calculations defined using **Data Analysis Expressions (DAX)**

In this case we're using a DAX time intelligence formula to calculate a **10-day rolling average**

AI Quest

WHEN TO USE POWER QUERY & POWER PIVOT

*Use **Power Query** and **Power Pivot** when you want to…*

- Analyze more data than can fit into a worksheet
- Create connections to databases or external sources
- Blend data across multiple large tables
- Automate the process of loading and shaping your data
- Unleash the **full business intelligence**

AI Quest

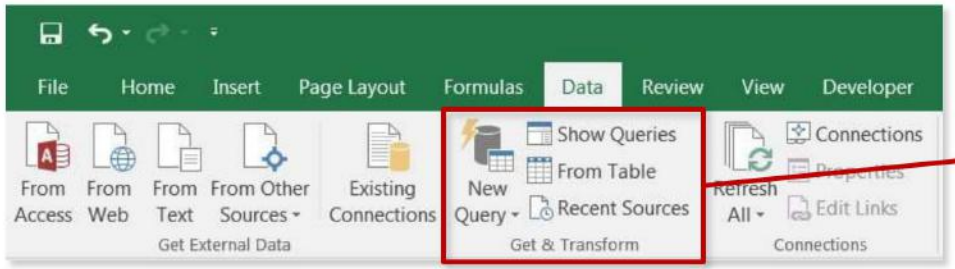
POWER QUERY

AI Quest

MEET POWER QUERY

Power Query (aka "**Get & Transform**") allows you to:

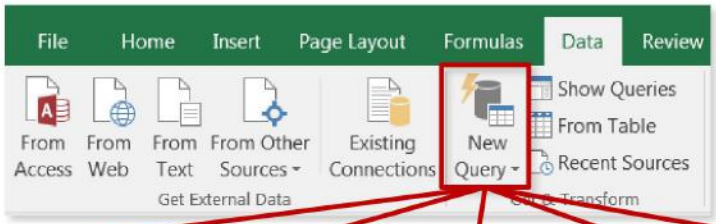
- Connect to data across a wide range of sources
- Filter, shape, append and transform raw data for further analysis and modeling
- Create stored procedures to automate your data prep (*like a macro!)*



The Power Query tools live in the **Data** tab, under the "**Get & Transform**" section (Excel 2016)

AI Quest

TYPES OF DATA CONNECTIONS



From File

- From Workbook
- From CSV
- From XML
- From Text
- From JSON
- From Folder
- From SharePoint Folder

From Database

- From SQL Server Database
- From Microsoft Access Database
- From SQL Server Analysis Services Database
- From Oracle Database
- From IBM DB2 Database
- From MySQL Database
- From PostgreSQL Database
- From Sybase Database
- From Teradata Database
- From SAP HANA Database

FromAzure

- From Microsoft Azure SQL Database
- From Microsoft Azure SQL Data Warehouse
- From Microsoft Azure Marketplace
- From Microsoft Azure HDInsight
- From Microsoft Azure Blob Storage
- From Microsoft Azure Table Storage

From Online Services

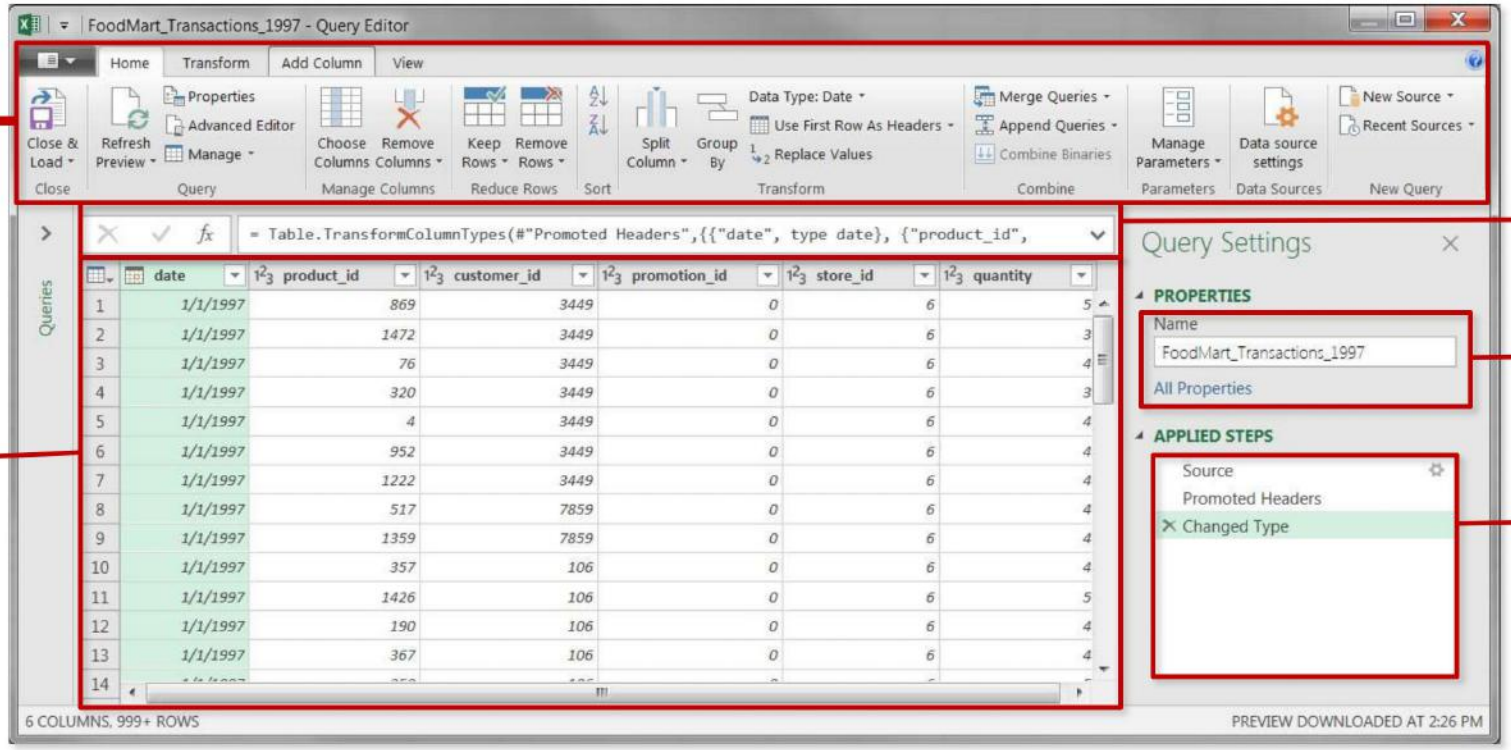
- From SharePoint Online List
- From Microsoft Exchange Online
- From Dynamics CRM Online
- From Facebook
- From Salesforce Objects
- From Salesforce Reports

From Other Sources

- From Web
- From SharePoint List
- From OData Feed
- From Hadoop File (HDFS)
- From Active Directory
- From Microsoft Exchange
- From ODBC
- From OLEDB
- Blank Query

AI Quest

THE QUERY EDITOR



Query Editing Tools

Data Preview

Formula Bar *(this is "M" code)*

Name your table!

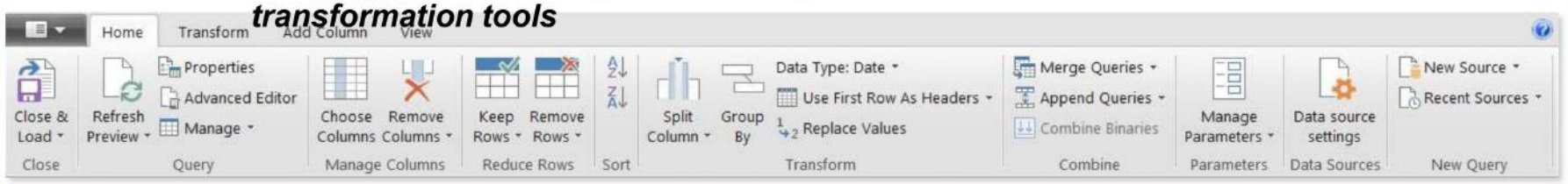
Applied Steps

Access the **Query Editor** by creating a new query and choosing the "*Edit*" option, or by launching the Workbook Queries pane (***Data*** > ***Show Queries***) and right-clicking an existing query to edit

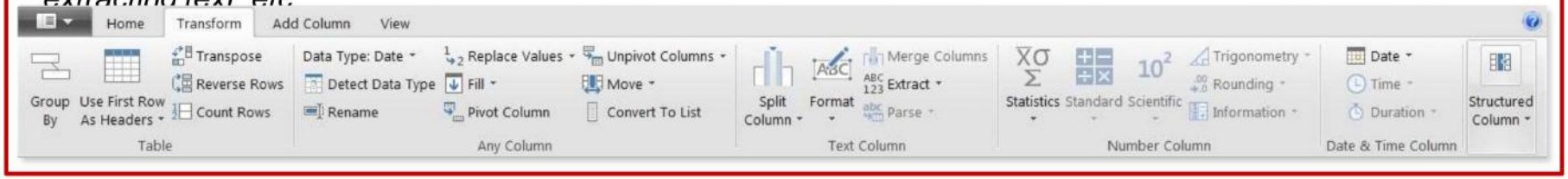
AI Quest

QUERY EDITOR TOOLS

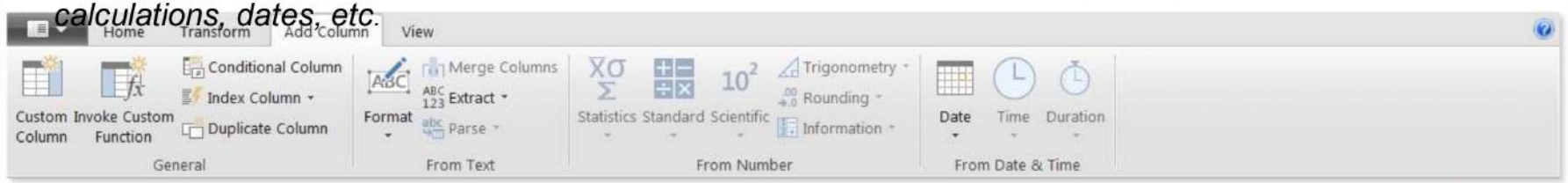
*The **HOME** tab includes **general settings** and **common table transformation tools***



*The **TRANSFORM** tab includes tools to **modify existing columns** (splitting/grouping, transposing, extracting text, etc.*

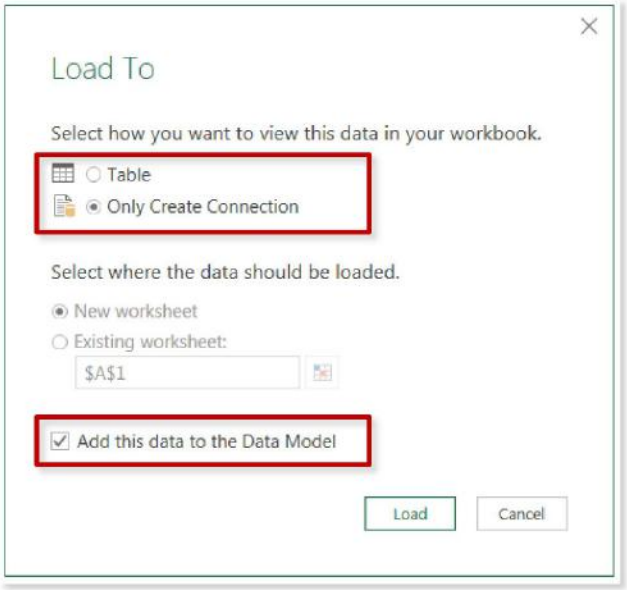


*The **ADD COLUMN** tools **create new columns** based on conditional rules, text operations, calculations, dates, etc.*



AI Quest

DATA LOADING OPTIONS



When you load data from Power Query, you have several options:

- **Table**
  - *Stores the data in a new or existing worksheet*
  - *Requires relatively small data sets (<1mm rows)*
- **Connection Only**
  - *Saves the data connection settings and applied steps*
  - *Data does not load to a worksheet*
- **Add to Data Model**
  - *Compresses and loads data to Excel's Data Model*
  - *Makes data accessible to Power Pivot for further analysis*

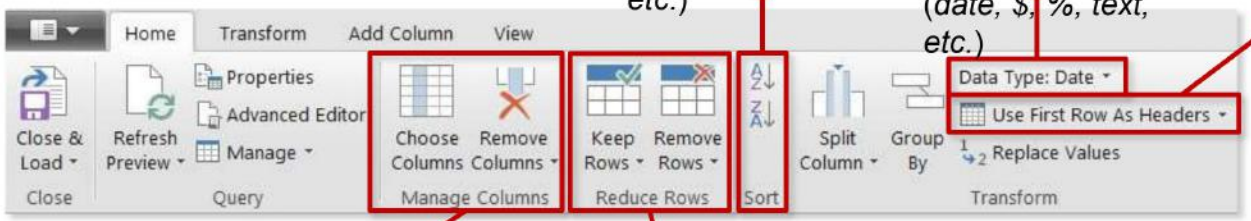
AI Quest

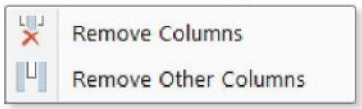
BASIC TABLE

Sort values
(*A-Z, Low-High, etc.*)

Change data types
(*date, $, %, text, etc.*)

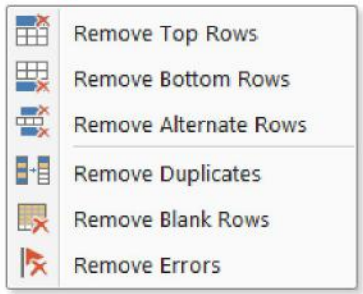
Promote header row





Keep or remove columns

Tip: *use the "Remove Other Columns" option if you always want a specific set*

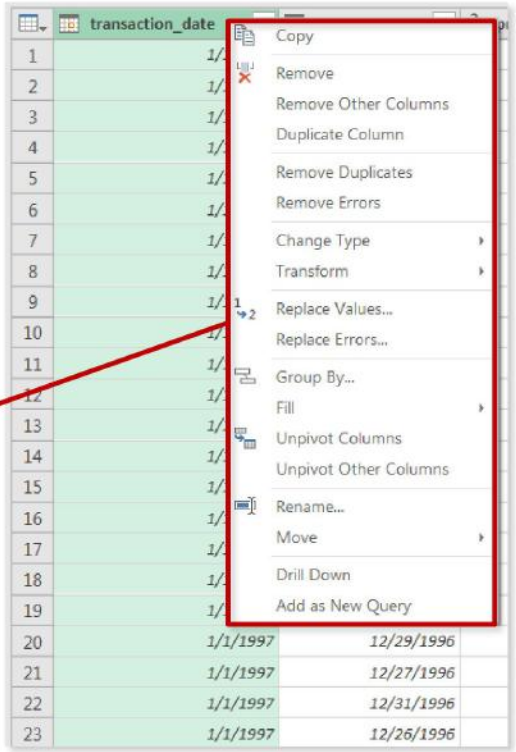


Keep or remove rows

Tip: *use the "Remove Duplicates" option to create a new lookup table from scratch*

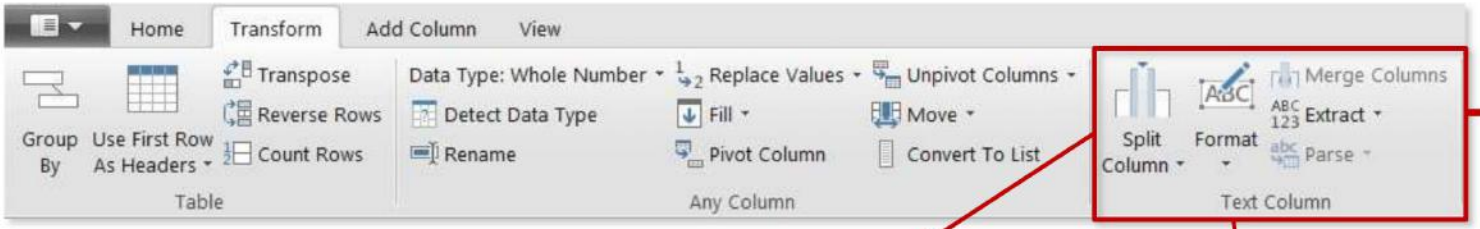
Duplicate, move & rename columns

Tip: *Right-click the column header to access common tools*



AI Quest

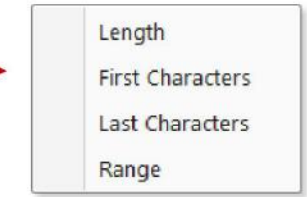
TEXT-SPECIFIC



Split a text column based on either a specific delimiter or a number of characters

Format a text column to upper, lower or proper case, or add a prefix or suffix

Tip: *Use "Trim" to eliminate leading & trailing spaces, or "Clean" to remove non-printable characters*



Extract characters from a text column using a fixed length, first or last, or a defined range

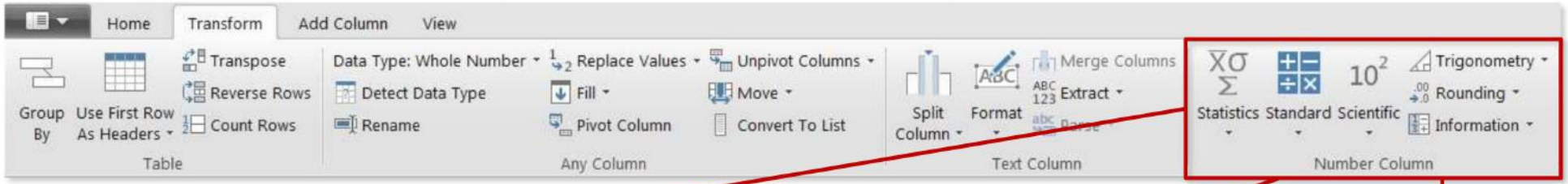
Tip*: Select two or more columns to* ***merge*** *or* ***concatenate*** *fields*

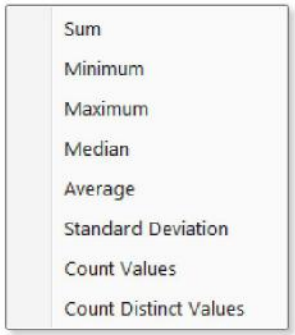
HEY THIS IS IMPORTANT!

You can access many of these tools in both the "**Transform**" and "**Add Column**" menus -- the difference is whether you want to ***add a new column*** or ***modify an existing one***

AI Quest

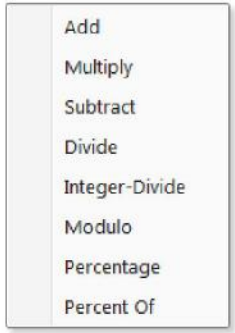
NUMBER-SPECIFIC



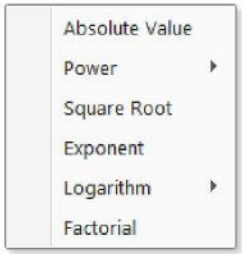


Statistics functions allow you to evaluate basic stats for the selected column (sum, min/max, average, count, countdistinct, etc)

Note: *These tools return a SINGLE value, and are commonly used to explore a table rather than prepare it for loading*



Standard



Scientific

Trigonometry

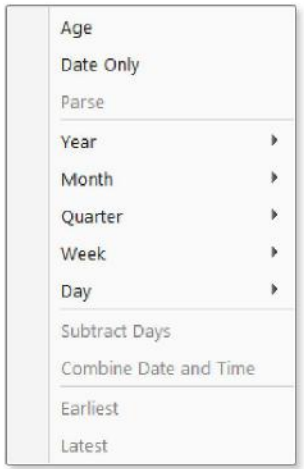
Standard, Scientific and **Trigonometry** tools allow you to apply standard operations (addition, multiplication, division, etc.) or more advanced calculations (power, logarithm, sine, tangent, etc) to each value in a column

Note: *Unlike the Statistics options, these tools are applied to each individual row in the table*

Information tools allow you to define binary flags (*TRUE/FALSE or 1/0*) to mark each row in a column as even, odd, positive or negative

AI Quest

DATE-SPECIFIC



Date & Time tools are relatively straight-forward, and include the following options:

- **Age**: Difference between the current time and the date in each row
- **Date Only:** Removes the time component of a date/time field
- **Year/Month/Quarter/Week/Day**: Extracts individual components from a date field (Time-specific options include Hour, Minute, Second, etc.)
- **Earliest/Latest:** Evaluates the earliest or latest date from a column as a single value (can only be accessed from the "Transform" menu)

Note: *You will almost always want to perform these operations from the "Add Column" menu to build out new fields, rather than transforming an individual date/time column*

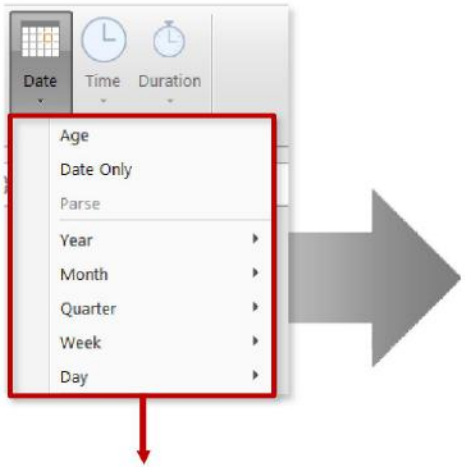
PRO TIP:

*Load up a table containing a **single date column** and use Date tools to build out an **entire calendar table***

HI Quest

CREATING A BASIC CALENDAR TABLE

	date
1	1/1/1997
2	1/2/1997
3	1/3/1997
4	1/4/1997
5	1/5/1997
6	1/6/1997
7	1/7/1997
8	1/8/1997
9	1/9/1997
10	1/10/1997
11	1/11/1997
12	1/12/1997
13	1/13/1997
14	1/14/1997
15	1/15/1997
16	1/16/1997
17	1/17/1997
18	1/18/1997
19	1/19/1997
20	1/20/1997
21	1/21/1997
22	1/22/1997
23	1/23/1997



Use pre-defined **Date** options in the "**Add Column**" menu to quickly build out a calendar table from a list of dates

	date	Year	Month	Quarter	WeekOfYear	Day Name
1	1/1/1997	1997	1	1	1	Wednesday
2	1/2/1997	1997	1	1	1	Thursday
3	1/3/1997	1997	1	1	1	Friday
4	1/4/1997	1997	1	1	1	Saturday
5	1/5/1997	1997	1	1	2	Sunday
6	1/6/1997	1997	1	1	2	Monday
7	1/7/1997	1997	1	1	2	Tuesday
8	1/8/1997	1997	1	1	2	Wednesday
9	1/9/1997	1997	1	1	2	Thursday
10	1/10/1997	1997	1	1	2	Friday
11	1/11/1997	1997	1	1	2	Saturday
12	1/12/1997	1997	1	1	3	Sunday
13	1/13/1997	1997	1	1	3	Monday
14	1/14/1997	1997	1	1	3	Tuesday
15	1/15/1997	1997	1	1	3	Wednesday
16	1/16/1997	1997	1	1	3	Thursday
17	1/17/1997	1997	1	1	3	Friday
18	1/18/1997	1997	1	1	3	Saturday
19	1/19/1997	1997	1	1	4	Sunday
20	1/20/1997	1997	1	1	4	Monday
21	1/21/1997	1997	1	1	4	Tuesday
22	1/22/1997	1997	1	1	4	Wednesday
23	1/23/1997	1997	1	1	4	Thursday

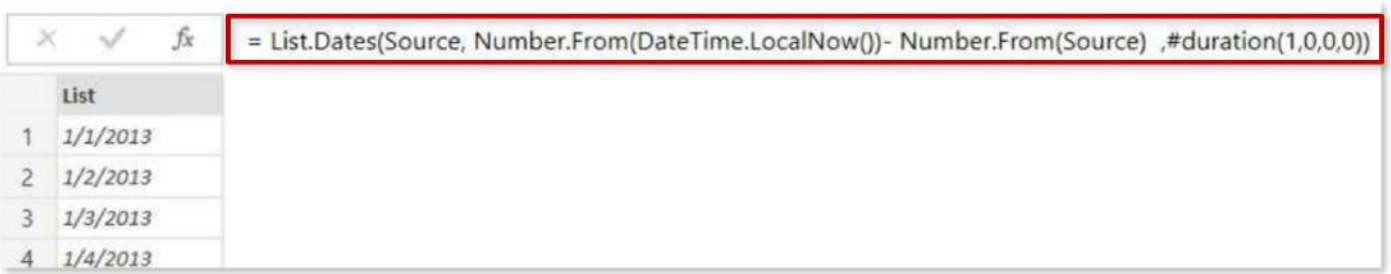
AI Quest

PRO TIP: CREATING A ROLLING CALENDAR

1) Create a new, blank query (***Data*** > ***New Query*** > ***From Other Sources*** > ***Blank Query***)

2) In the formula bar, generate a starting date by entering a "**literal**" (*1/1/2013 shown below*):

3) Click the **fX** icon to add a new custom step, and enter the following formula *exactly* as shown:

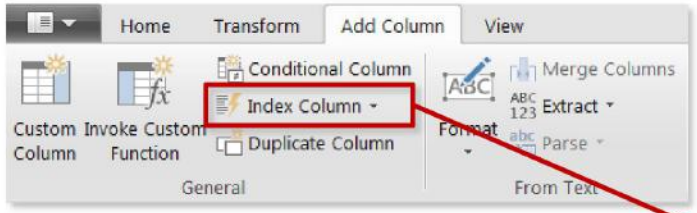


4) Convert the resulting list into a Table (***List Tools*** > ***To Table***) and format the column as a **Date**

5) Add calculated Date columns (*Year, Month, Week, etc.*) as necessary using the **Add Column** tools

AI Quest

ADDING AN INDEX



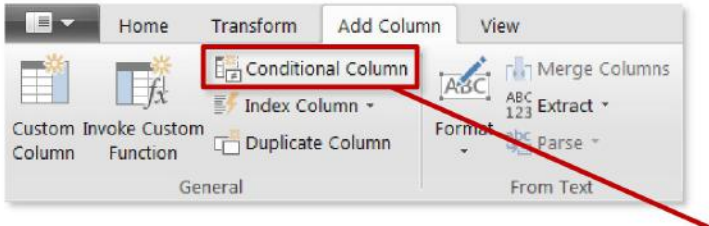
Index Columns contain a list of sequential values that can be used to identify each unique row in a table (*typically starting from 0 or 1*)

These columns are often used to create **unique IDs** that can be used to form relationships between tables (*more on that later!*)

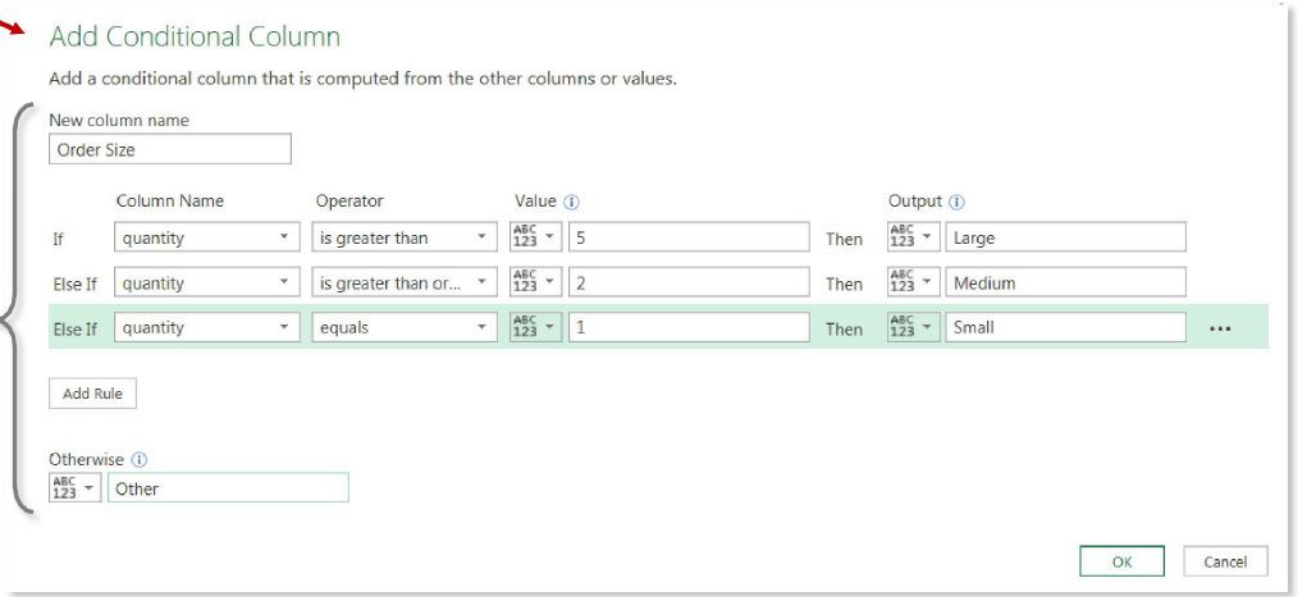
	Index	transaction_date	stock_date	product_id	customer_id	promotion_id	st
1	0	1/1/1997	12/28/1996	761	6613	0	
2	1	1/1/1997	12/30/1996	1435	8830	0	
3	2	1/1/1997	12/29/1996	1175	8830	0	
4	3	1/1/1997	12/30/1996	1152	8830	0	
5	4	1/1/1997	12/31/1996	1245	5005	0	
6	5	1/1/1997	12/27/1996	209	5005	0	
7	6	1/1/1997	12/28/1996	1345	5005	0	
8	7	1/1/1997	12/28/1996	1468	5005	0	
9	8	1/1/1997	12/26/1996	84	7962	0	
10	9	1/1/1997	12/30/1996	966	7962	0	
11	10	1/1/1997	12/27/1996	1022	7962	0	
12	11	1/1/1997	12/29/1996	440	7962	0	
13	12	1/4/1997	12/28/1996	151	2274	1054	
14	13	1/4/1997	12/28/1996	1287	8648	1054	
15	14	1/4/1997	12/30/1996	1264	8648	1054	
16	15	1/4/1997	12/31/1996	188	8648	1054	
17	16	1/4/1997	1/1/1997	1526	8648	1054	
18	17	1/4/1997	12/29/1996	518	8762	1054	
19	18	1/5/1997	12/31/1996	963	4018	0	
20	19	1/5/1997	12/29/1996	154	1418	0	
21							

AI Quest

ADDING A CONDITIONAL



Conditional Columns allow you to define new fields based on logical rules and conditions (*IF/THEN statements*)

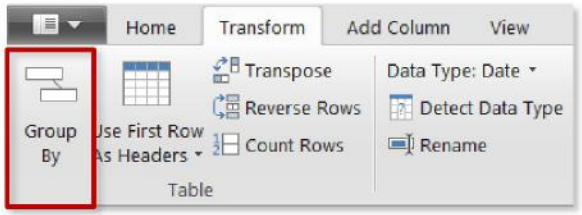


*In this case we're creating a new conditional column called "**Order Size**", which depends on the values in the "**quantity**" column, as follows:*

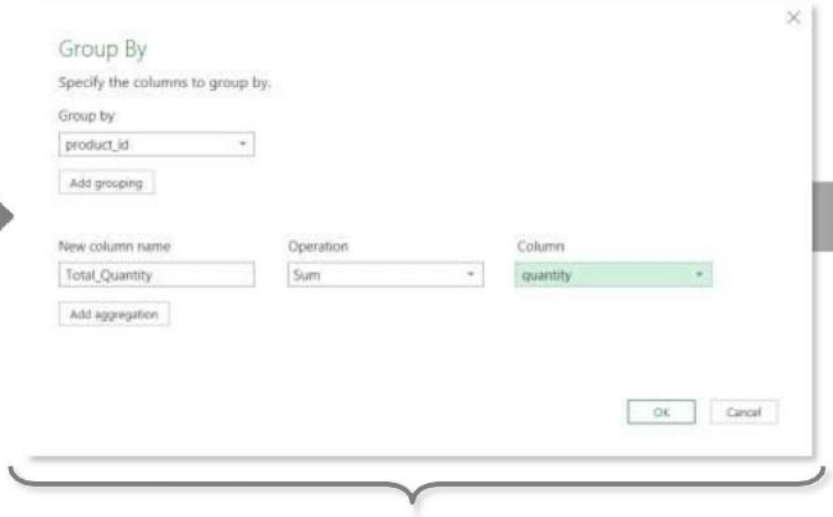
- *If quantity **>5**, Order Size = "**Large**"*
- *If quantity is **from 2-5**, Order Size = "**Medium**"*
- *If quantity **=1**, Order Size = "**Small**"*
- *Otherwise Order Size = "**Other**"*

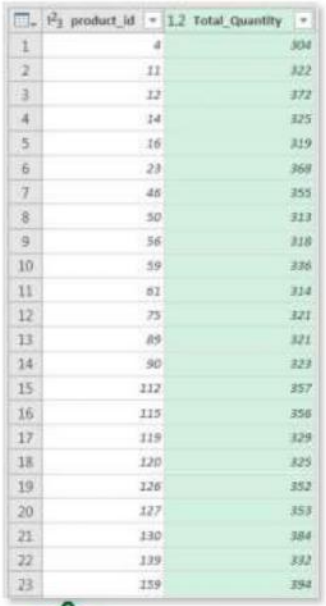
AI Quest

GROUPING & AGGREGATING



Group By allows you to aggregate your data at a different level (*i.e. transform daily data into monthly, roll up transaction-level data by store, etc.*)





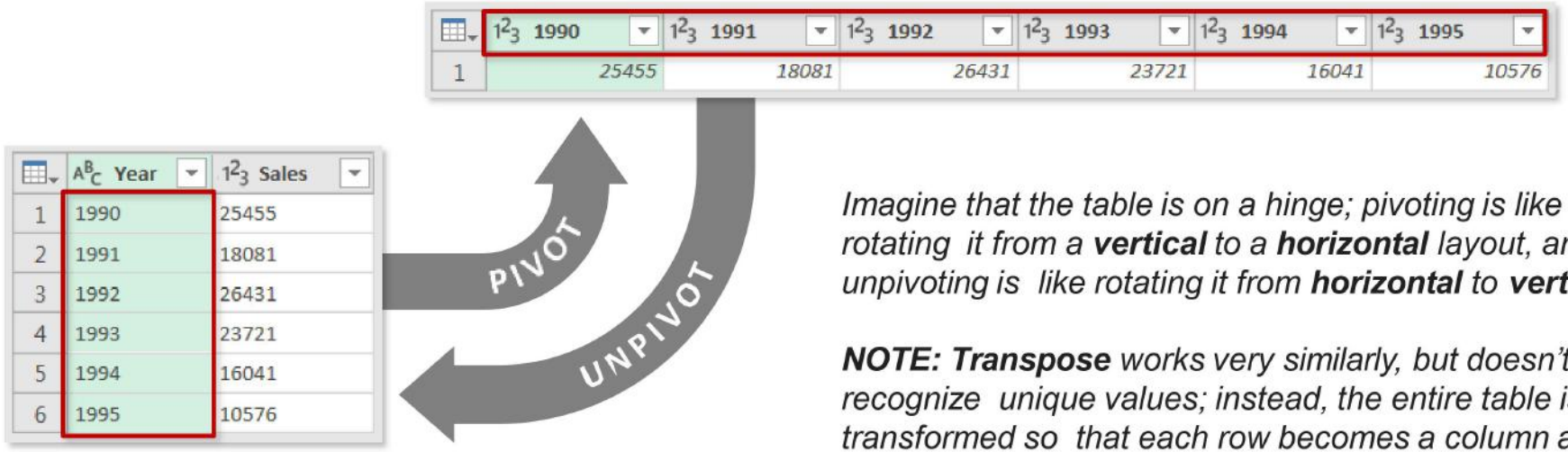
*In this case we're transforming a daily, transaction-level table into a summary of "**quantity**" by "**product_id**"*

Note that we lose any field not specified in the Group By settings

AI Quest

PIVOTING & UNPIVOTING

"Pivoting" is a fancy way to describe the process of turning **distinct row values** into **columns** (*"pivoting"*) or turning **columns** into **rows** (*"unpivoting"*)

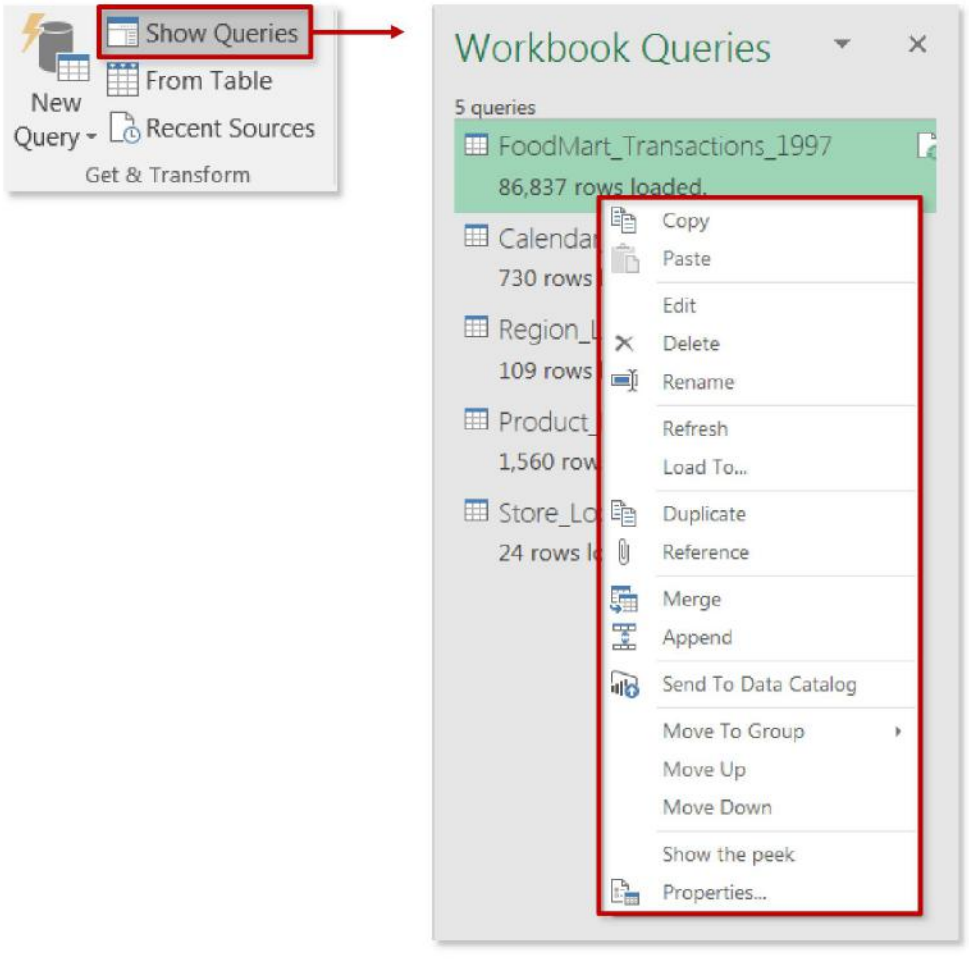


*Imagine that the table is on a hinge; pivoting is like rotating it from a **vertical** to a **horizontal** layout, and unpivoting is like rotating it from **horizontal** to **vertical***

***NOTE: Transpose** works very similarly, but doesn't recognize unique values; instead, the entire table is transformed so that each row becomes a column and vice versa*

AI Quest

MODIFYING WORKBOOK QUERIES



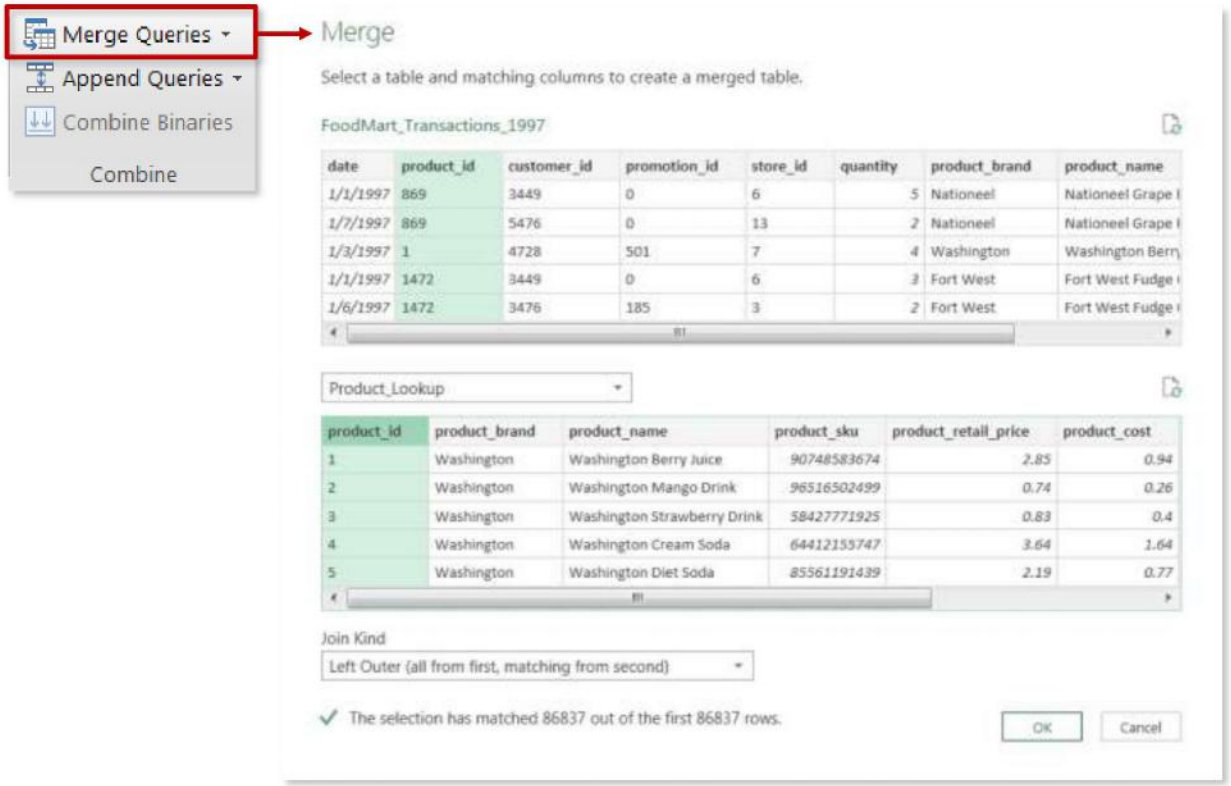
Click on **Show Queries** to launch the **Workbook Queries** pane

Right-click any individual query to access common options and tools:

- **Edit** (launches the Query Editor)
- **Delete**
- **Rename**
- **Refresh**
- **Duplicate**
- **Merge**
- **Append**

AI Quest

MERGING QUERIES



- Merging queries allows you to **join tables** based on a common column (like VLOOKUP)
- In this case we're merging the **FoodMart_Transactions_1997** table with the **Product_Lookup** table, which share a "*product_id*" column

TIP: Merging **adds columns** to an existing table

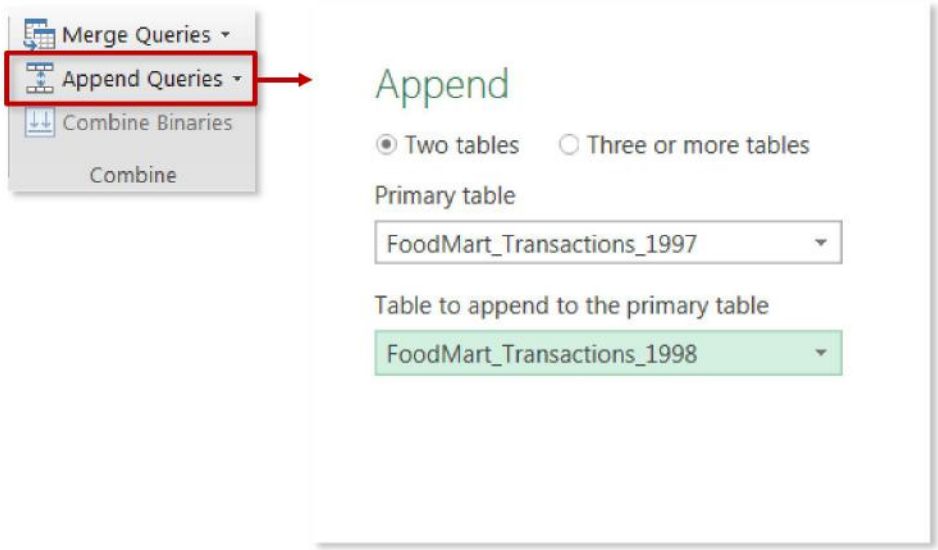
HEY THIS IS IMPORTANT!

Just because you ***can*** merge tables, doesn't mean you ***should***.

In general, it's better to keep tables separate and define **relationships** between them (*more on that later!*)

AI Quest

APPENDING



- Appending queries allows you to **combine** (or ***stack***) tables that share a common structure and set of columns

- In this case we're appending the **FoodMart_Transactions_1998** table to the **FoodMart_Transactions_1997** table, since they contain the same set of columns and data types

TIP: Appending **adds rows** to an existing table

PRO TIP:

*Use the "**From Folder**" query option to automatically append all files from within the same folder*

AI Quest

POWER QUERY BEST PRACTICES

Give your queries clear and intuitive names, *before* loading the data

- *Define names immediately; updating query & table names later can be a headache, especially if you've already referenced them in calculated measures*
- *Don't use spaces in table names (otherwise you have surround them with single quotes)*

Do as much shaping as possible at the source of the data

- *Shaping data at the source (i.e. SQL, Access) minimizes the need for complex procedures in Power Query, and allows you to create new models without replicating the same process*

When working with large tables, only load the data you need

Quest

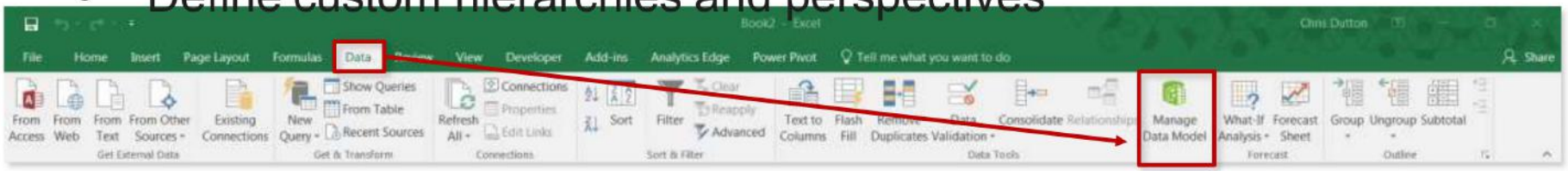
DATA MODELING

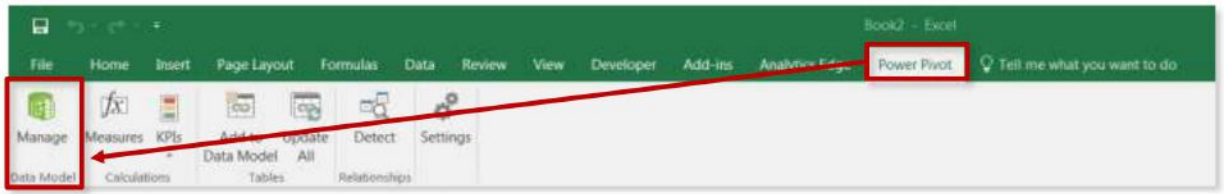
AI Quest

MEET EXCEL'S DATA MODEL

The **Data Model** provides simple and intuitive tools for building relational databases directly in Excel. With the data model you can:

- Manage massive datasets that can't fit into worksheets
- Create table relationships to blend data across multiple sources
- Define custom hierarchies and perspectives



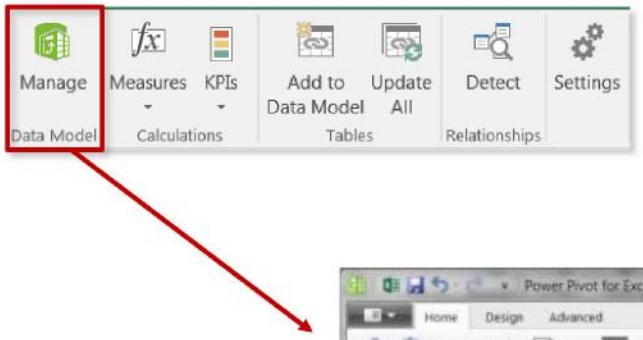


Access the **Data Model** through the **Power Pivot** tab or the **Data** tab

*(**Note:** you may need to enable the Power Pivot tab via **File > Options > Add-Ins > Ma Ins**)*

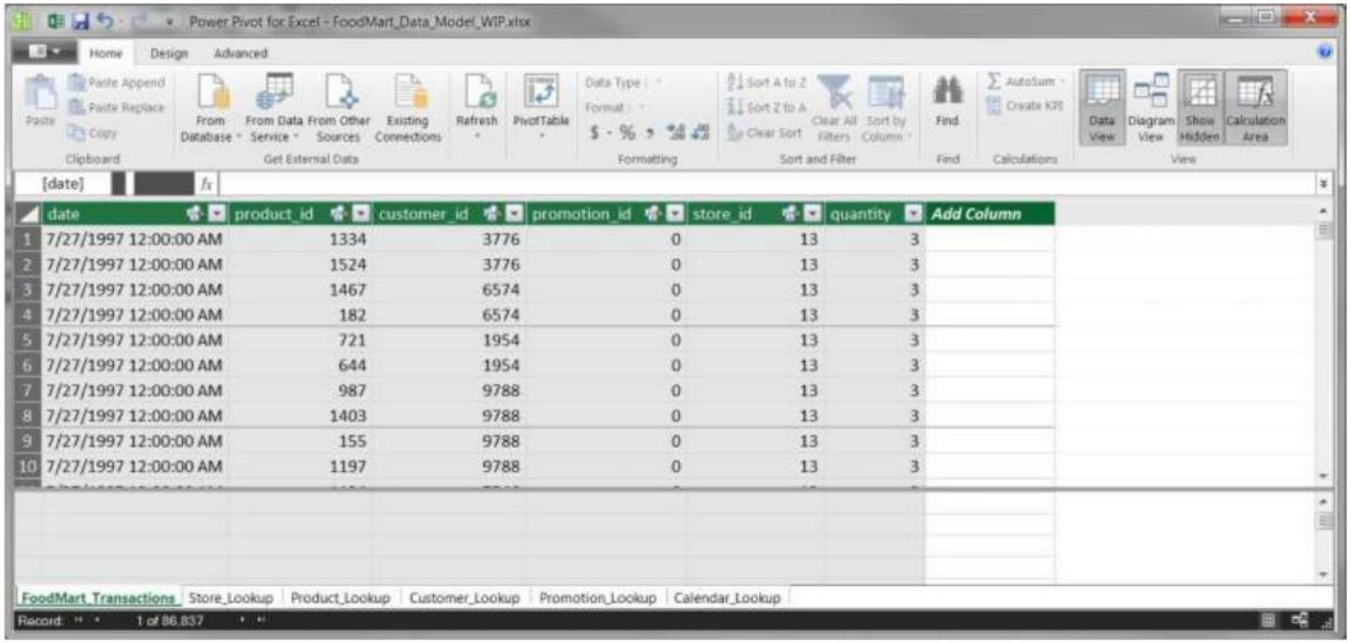
AI Quest

THE DATA MODEL WINDOW



The **Data Model** opens in a separate Excel window, where you can view your data tables, calculate new measures, and define table relationships

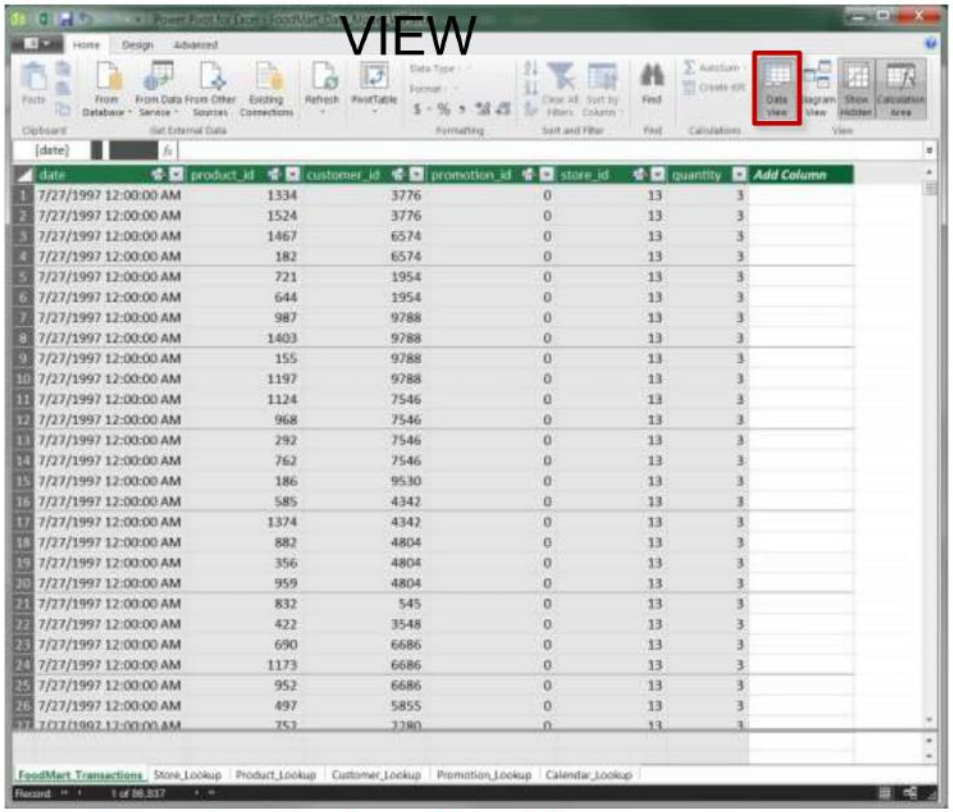
Note: *Closing the Data Model window does NOT close your Excel workbook*



AI Quest

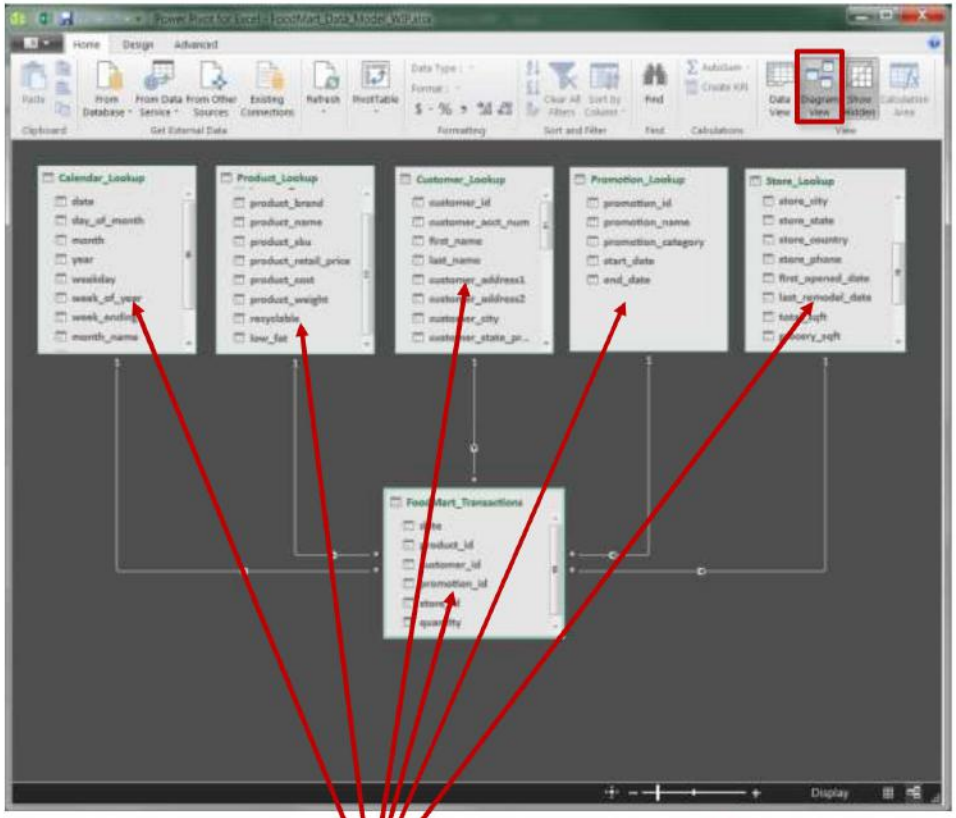
DATA VIEW VS. DIAGRAM VIEW

DATA VIEW



Tables organized in ***tabs***

DIAGRAM



Tables organized as ***objects***

AI Quest

DATABASE NORMALIZATION

Normalization is the process of organizing the tables and columns in a relational database to reduce redundancy and preserve data integrity. It is commonly used to:

- **Eliminate redundant data** to decrease table sizes and improve processing speed & efficiency
- **Minimize errors and anomalies** from data modifications (inserting, updating or deleting records)
- **Simplify queries** and structure the database for meaningful analysis

In a normalized database, each table should serve a ***distinct*** and ***specific*** purpose (*i.e. product information, calendar fields, transaction records, customer attributes, etc.*)

date	product_id	quantity	product_brand	product_name	product_sku	product_weight
1/1/1997	869	5	Nationeel	Nationeel Grape Fruit Roll	52382137179	17
1/7/1997	869	2	Nationeel	Nationeel Grape Fruit Roll	52382137179	17
1/3/1997	1	4	Washington	Washington Berry Juice	90748583674	8.39
1/1/1997	1472	3	Fort West	Fort West Fudge Cookies	37276054024	8.28
1/6/1997	1472	2	Fort West	Fort West Fudge Cookies	37276054024	8.28
1/5/1997	2	4	Washington	Washington Mango Drink	96516502499	7.42
1/1/1997	76	4	Red Spade	Red Spade Sliced Chicken	62054644227	18.1
1/1/1997	76	2	Red Spade	Red Spade Sliced Chicken	62054644227	18.1
1/5/1997	3	2	Washington	Washington Strawberry Drink	58427771925	13.1
1/7/1997	3	2	Washington	Washington Strawberry Drink	58427771925	13.1
1/1/1997	320	3	Excellent	Excellent Cranberry Juice	36570182442	16.4

*When you **don't** normalize, you end up with tables like this; all of the duplicate product records could be eliminated with a lookup table based on **product_id***

This may not seem critical now, but minor inefficiencies can become major problems as databases scale in size

AI Quest

DATA TABLES VS. LOOKUP TABLES

Models generally contain two types of tables: **data** (or "*fact*") tables, and **lookup** (or "*dimension*") tables

- **Data tables** contain numbers or values, typically at the most granular level possible, with ID or "*key*" columns that can be used to connect to each lookup table
- **Lookup tables** provide descriptive, often text-based attributes about each dimension in a table

date	product_id	quantity
1/1/1997	869	5
1/1/1997	1472	3
1/1/1997	76	4
1/1/1997	320	3
1/1/1997	4	4
1/1/1997	952	4
1/1/1997	1222	4
1/1/1997	517	4
1/1/1997	1359	4
1/1/1997	357	4
1/1/1997	1426	5
1/1/1997	190	4
1/1/1997	367	4
1/1/1997	250	5
1/1/1997	600	4
1/1/1997	702	5

*This **Data Table** contains "**quantity**" values, and connects to lookup tables via the "**date**" and*

date	day_of_month	month	year	weekday	week_of_year	week_ending	month_name	quarter
1/1/1997	1	1	1997	Wednesday	1	1/5/1997	January	Q1
1/2/1997	2	1	1997	Thursday	1	1/5/1997	January	Q1
1/3/1997	3	1	1997	Friday	1	1/5/1997	January	Q1
1/4/1997	4	1	1997	Saturday	1	1/5/1997	January	Q1
1/5/1997	5	1	1997	Sunday	2	1/5/1997	January	Q1
1/6/1997	6	1	1997	Monday	2	1/12/1997	January	Q1

*This **Calendar Lookup** table provides additional attributes about each **date** (month, year, weekday, quarter, etc.)*

product_id	product_brand	product_name	product_sku	product_retail_price	product_cost	product_weight
1	Washington	Washington Berry Juice	90748583674	2.85	0.94	8.39
2	Washington	Washington Mango Drink	96516502499	0.74	0.26	7.42
3	Washington	Washington Strawberry Drink	58427771925	0.83	0.4	13.1
4	Washington	Washington Cream Soda	64412155747	3.64	1.64	10.6
5	Washington	Washington Diet Soda	85561191439	2.19	0.77	6.66
6	Washington	Washington Cola	29804642796	1.15	0.37	15.8
7	Washington	Washington Diet Cola	20191444754	2.61	0.91	18
8	Washington	Washington Orange Juice	89770532250	2.59	0.8	8.97

*This **Product Lookup** table provides additional attributes about each **product** (brand, product name, sku, price, etc.)*

AI Quest

PRIMARY & FOREIGN KEYS

date	product_id	quantity
1/1/1997	869	5
1/1/1997	1472	3
1/1/1997	76	4
1/1/1997	320	3
1/1/1997	4	4
1/1/1997	952	4
1/1/1997	1222	4
1/1/1997	517	4
1/1/1997	1359	4
1/1/1997	357	4
1/1/1997	1426	5
1/1/1997	190	4
1/1/1997	367	4
1/1/1997	250	5
1/1/1997	600	4
1/1/1997	702	5

These columns are **foreign keys**; they contain *multiple* instances of each value, and are used to match the **primary keys** in related lookup tables

date	day_of_month	month	year	weekday	week_of_year	week_ending	month_name	quarter
1/1/1997	1	1	1997	Wednesday	1	1/5/1997	January	Q1
1/2/1997	2	1	1997	Thursday	1	1/5/1997	January	Q1
1/3/1997	3	1	1997	Friday	1	1/5/1997	January	Q1
1/4/1997	4	1	1997	Saturday	1	1/5/1997	January	Q1
1/5/1997	5	1	1997	Sunday	2	1/5/1997	January	Q1
1/6/1997	6	1	1997	Monday	2	1/12/1997	January	Q1

product_id	product_brand	product_name	product_sku	product_retail_price	product_cost	product_weight
1	Washington	Washington Berry Juice	90748583674	2.85	0.94	8.39
2	Washington	Washington Mango Drink	96516502499	0.74	0.26	7.42
3	Washington	Washington Strawberry Drink	58427771925	0.83	0.4	13.1
4	Washington	Washington Cream Soda	64412155747	3.64	1.64	10.6
5	Washington	Washington Diet Soda	85561191439	2.19	0.77	6.66
6	Washington	Washington Cola	29804642796	1.15	0.37	15.8
7	Washington	Washington Diet Cola	20191444754	2.61	0.91	18
8	Washington	Washington Orange Juice	89770532250	2.59	0.8	8.97

These columns are **primary keys**; they *uniquely* identify each row of a table, and match the **foreign keys** in related data tables

AI Quest

RELATIONSHIPS VS. MERGED TABLES

*Can't I just **merge queries** or use **LOOKUP** or **RELATED** functions to pull those attributes into the fact table itself, so that I have everything in one place??*

-Anonymous confused man

*Original **Fact Table** fields*

*Attributes from **Calendar Lookup** table*

*Attributes from **Product Lookup** table*

date	product_id	quantity	day_of_month	month	year	weekday	month_name	quarter	product_brand	product_name	product_sku	product_weight
1/1/1997	869	5	1	1	1997	Wednesday	January	Q1	Nationeel	Nationeel Grape Fruit Roll	52382137179	17
1/7/1997	869	2	7	1	1997	Tuesday	January	Q1	Nationeel	Nationeel Grape Fruit Roll	52382137179	17
1/3/1997	1	4	3	1	1997	Friday	January	Q1	Washington	Washington Berry Juice	90748583674	8.39
1/1/1997	1472	3	1	1	1997	Wednesday	January	Q1	Fort West	Fort West Fudge Cookies	37276054024	8.28
1/6/1997	1472	2	6	1	1997	Monday	January	Q1	Fort West	Fort West Fudge Cookies	37276054024	8.28
1/5/1997	2	4	5	1	1997	Sunday	January	Q1	Washington	Washington Mango Drink	96516502499	7.42
1/1/1997	76	4	1	1	1997	Wednesday	January	Q1	Red Spade	Red Spade Sliced Chicken	62054644227	18.1
1/1/1997	76	2	1	1	1997	Wednesday	January	Q1	Red Spade	Red Spade Sliced Chicken	62054644227	18.1
1/5/1997	3	2	5	1	1997	Sunday	January	Q1	Washington	Washington Strawberry Drink	58427771925	13.1
1/7/1997	3	2	7	1	1997	Tuesday	January	Q1	Washington	Washington Strawberry Drink	58427771925	13.1
1/1/1997	320	3	1	1	1997	Wednesday	January	Q1	Excellent	Excellent Cranberry Juice	36570182442	16.4

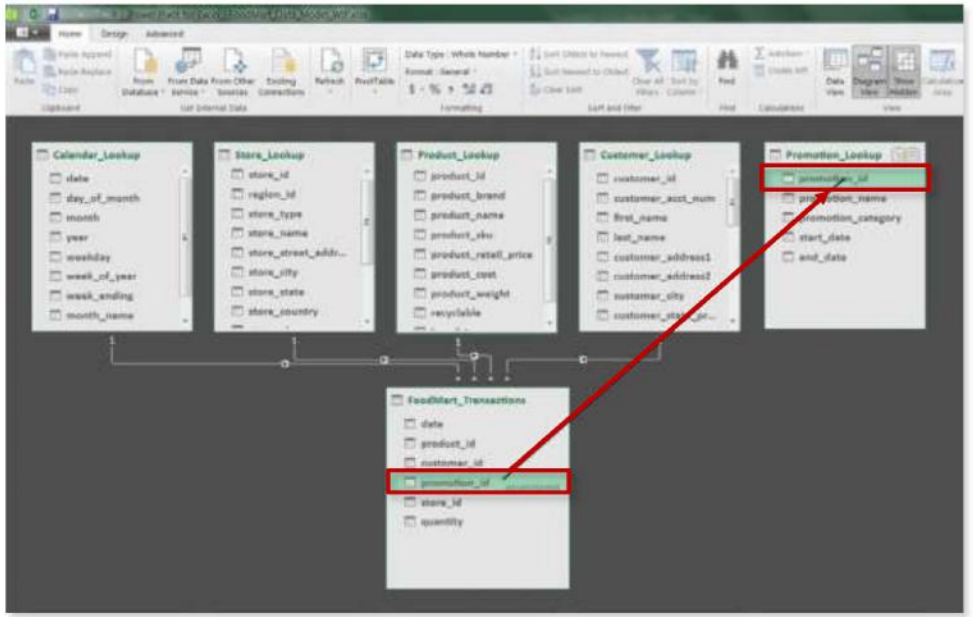
Sure, but **it's extremely inefficient.**

- Merging data in this way creates **redundant data** and utilizes **significantly more memory and processing power** than creating relationships between multiple small tables

AI Quest

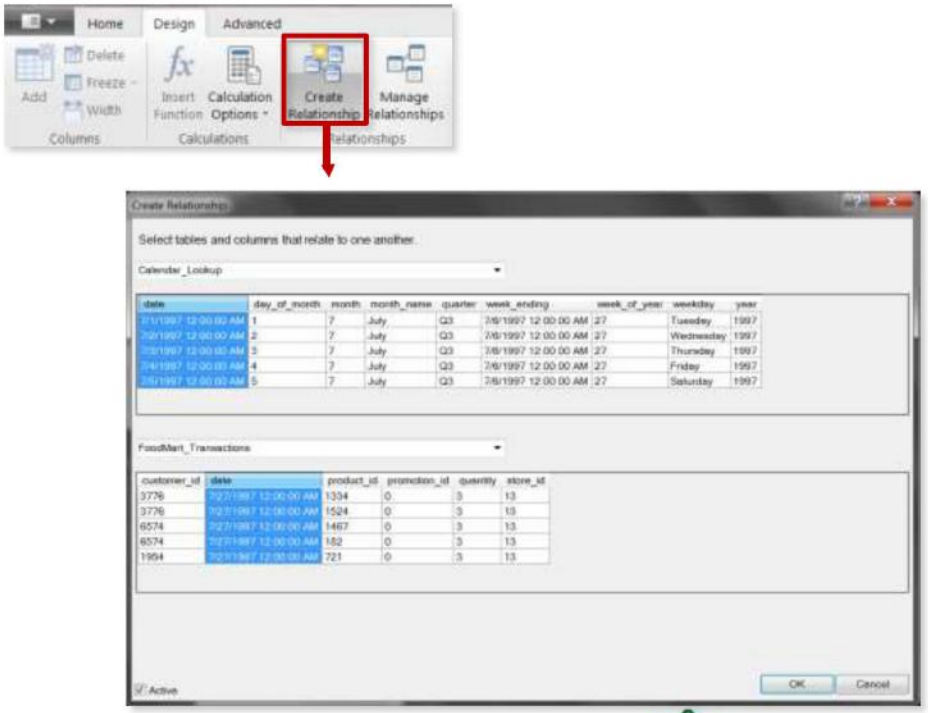
CREATING TABLE RELATIONSHIPS

Option 1: Click and drag relationships in **Diagram View**



Tip: *Always drag relationships from the* ***Data*** *table to the* ***Lookup*** *tables*

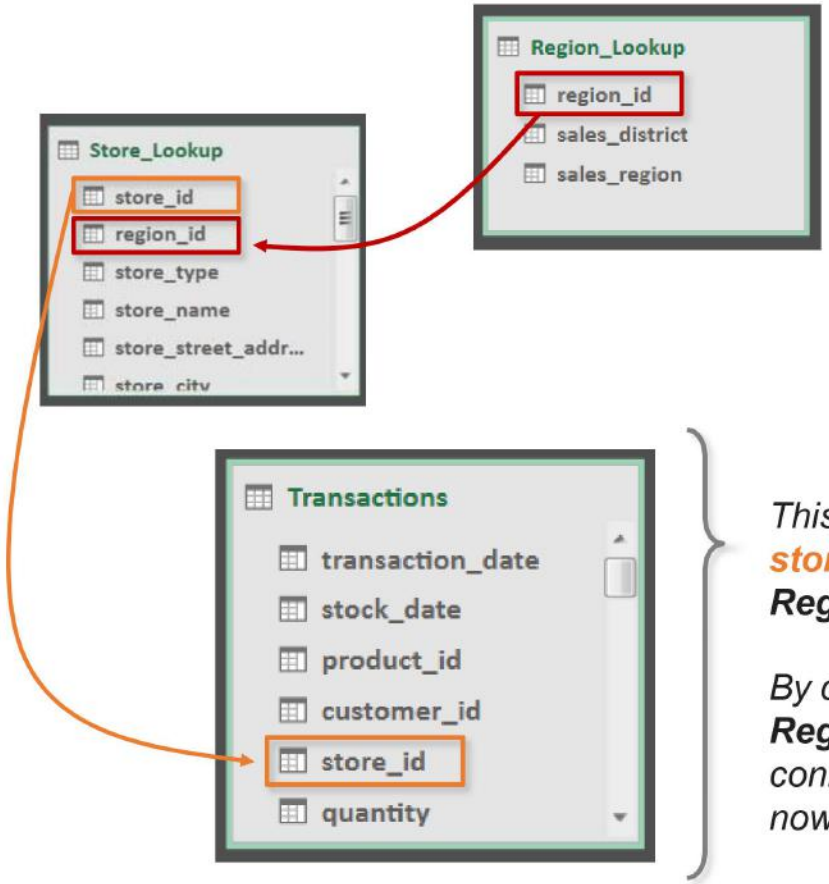
Option 2: Use "**Create Relationship**" in the **Design** tab



***Note:** In Excel 2010/2013 the diagram view looks a bit different, and arrows point in the opposite direction*

AI Quest

CONNECTING LOOKUPS TO



PRO TIP:

*Models with multiple related lookup tables are called "**snowflake**" schemas*

*Models with a single table for each lookup or dimension are called "**star**" schemas*

*This **Transactions** data table can connect to **Store_Lookup** using **store_id**, but does not contain a **region_id** to connect to the **Region_Lookup** table*

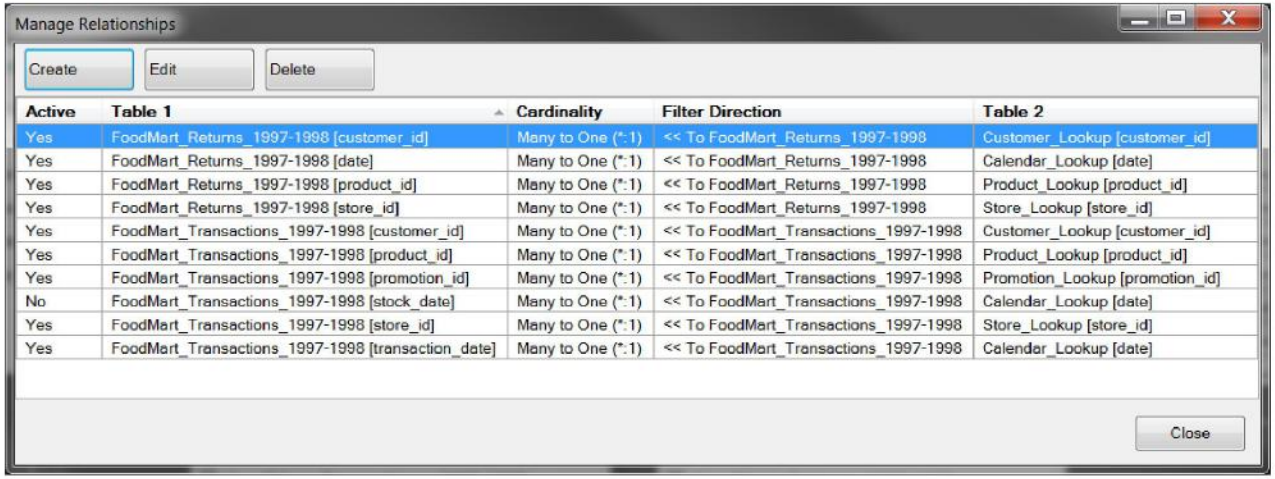
*By creating a relationship between **Store_Lookup** and **Region_Lookup** (using **region_id**), we have essentially connected **Transactions** with **Region_Lookup**; filter context will now flow all the way down the chain*

AI Quest

MODIFYING TABLE RELATIONSHIPS

The **Manage Relationships** window allows you to create, edit or delete any connection in the data model

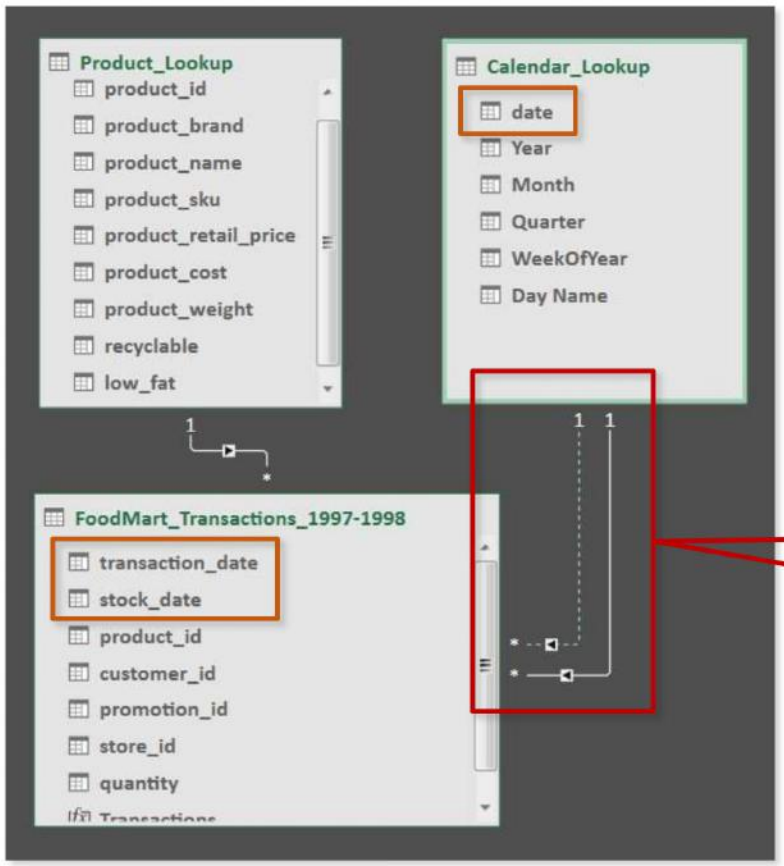
- Use this to see all table relationships, as well as table names, cardinality and filter direction
- ***Note:*** *double-click a single connection in diagram view to edit an individual relationship*

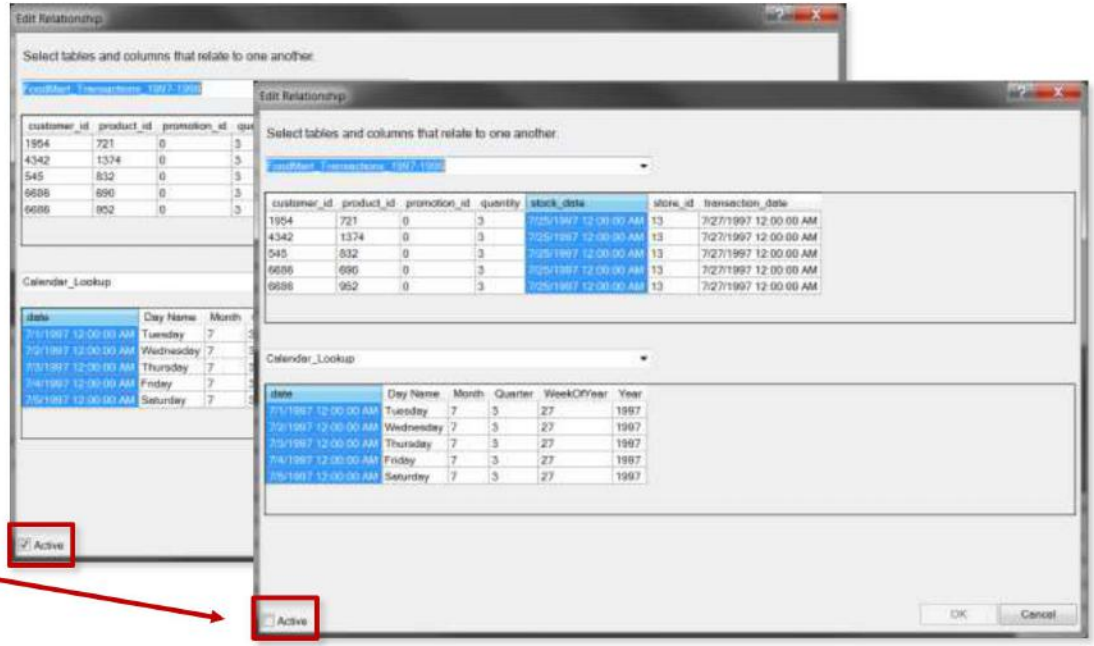


Active	Table 1	Cardinality	Filter Direction	Table 2
Yes	FoodMart_Returns_1997-1998 [customer_id]	Many to One (*:1)	<< To FoodMart_Returns_1997-1998	Customer_Lookup [customer_id]
Yes	FoodMart_Returns_1997-1998 [date]	Many to One (*:1)	<< To FoodMart_Returns_1997-1998	Calendar_Lookup [date]
Yes	FoodMart_Returns_1997-1998 [product_id]	Many to One (*:1)	<< To FoodMart_Returns_1997-1998	Product_Lookup [product_id]
Yes	FoodMart_Returns_1997-1998 [store_id]	Many to One (*:1)	<< To FoodMart_Returns_1997-1998	Store_Lookup [store_id]
Yes	FoodMart_Transactions_1997-1998 [customer_id]	Many to One (*:1)	<< To FoodMart_Transactions_1997-1998	Customer_Lookup [customer_id]
Yes	FoodMart_Transactions_1997-1998 [product_id]	Many to One (*:1)	<< To FoodMart_Transactions_1997-1998	Product_Lookup [product_id]
Yes	FoodMart_Transactions_1997-1998 [promotion_id]	Many to One (*:1)	<< To FoodMart_Transactions_1997-1998	Promotion_Lookup [promotion_id]
No	FoodMart_Transactions_1997-1998 [stock_date]	Many to One (*:1)	<< To FoodMart_Transactions_1997-1998	Calendar_Lookup [date]
Yes	FoodMart_Transactions_1997-1998 [store_id]	Many to One (*:1)	<< To FoodMart_Transactions_1997-1998	Store_Lookup [store_id]
Yes	FoodMart_Transactions_1997-1998 [transaction_date]	Many to One (*:1)	<< To FoodMart_Transactions_1997-1998	Calendar_Lookup [date]

AI Quest

ACTIVE VS. INACTIVE RELATIONSHIPS



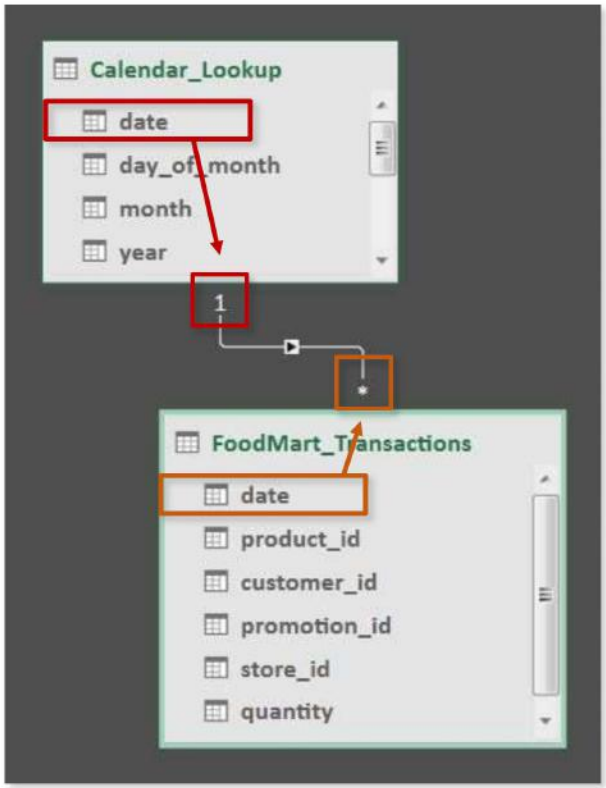


We can connect the **Calendar_Lookup** and **FoodMart_Transactions** tables on both **transaction_date** and **stock_date**; however, only one can be active at a time

To make a connection active or inactive, double-click the connection and check the box, or right-click the relationship line itself (**Note:** must deactivate one before activating another!)

AI Quest

RELATIONSHIP CARDINALITY



Cardinality refers to the uniqueness of values in a column

In Power Pivot, all relationships in a data model should follow a "**one-to-many**" cardinality

- Each column (or "key") used to join tables can only have **one instance** of each unique value in the lookup table (these are the *primary* keys), but may have **many instances** of each unique value in the data table (these are the *foreign keys*)

*In this case we're joining the **Calendar_Lookup** table to the **FoodMart_Transactions** data table using the **date** column as our key*

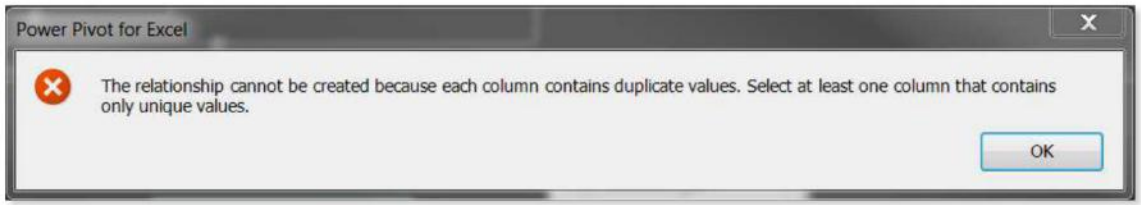
*There is only **one** instance of each date in the lookup table (noted by the "**1**"), but **many** instances of each date in the data table (noted by the asterisk "*****"), since multiple transactions occur each day*

***Note:** In Excel 2010/2013 the diagram view looks a bit different, and arrows point in the opposite direction*

BAD CARDINALITY: MANY-TO-MANY

product_id	product_name	product_sku
4	Washington Cream Soda	64412155747
4	Washington Diet Cream Soda	81727382373
5	Washington Diet Soda	85561191439
7	Washington Diet Cola	20191444754
8	Washington Orange Juice	89770532250

date	product_id	transactions
1/1/2017	4	12
1/2/2017	4	9
1/3/2017	4	11
1/1/2017	5	16
1/2/2017	5	19
1/1/2017	7	11



- If we try to connect these tables using the **product_id** field, we'll have a **many-to-many** relationship since there are multiple instances of each ID in both tables
- Even if we *could* create this relationship in Power Pivot, how would you know which product was actually sold on each date – ***Cream Soda*** or ***Diet Cream Soda***?

AI Quest

BAD CARDINALITY: ONE-TO-ONE

product_id	product_name	product_sku
4	Washington Cream Soda	64412155747
5	Washington Diet Soda	85561191439
7	Washington Diet Cola	20191444754
8	Washington Orange Juice	89770532250

product_id	product_price
4	$3.64
5	$2.19
7	$2.61
8	$2.59

- In this case, connecting the tables above using the **product_id** field creates a **one-to-one** relationship, since each ID only appears once in each table
- Unlike many-to-many, there is nothing *illegal* about this relationship; it's just **inefficient**

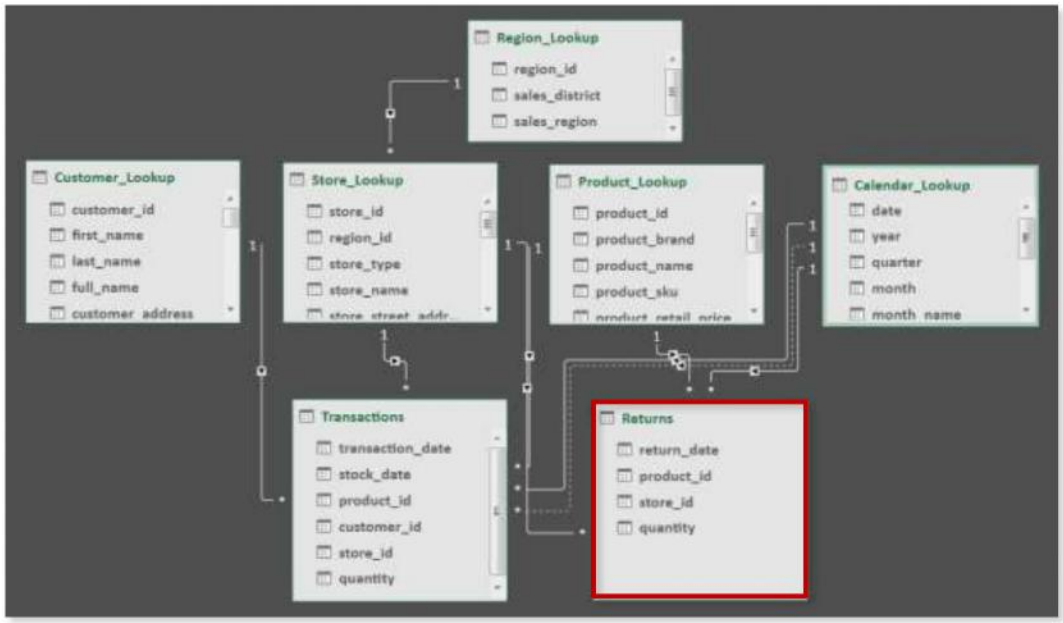
*To eliminate the inefficiency, you could simply **merge the two tables** into a single, valid lookup*

***Note:** this still respects the laws of normalization, since all rows are unique and directly related to the primary key*

product_id	product_name	product_sku	product_price
4	Washington Cream Soda	64412155747	$3.64
5	Washington Diet Soda	85561191439	$2.19
7	Washington Diet Cola	20191444754	$2.61
8	Washington Orange Juice	89770532250	$2.59

AI Quest

CONNECTING MULTIPLE DATA TABLES



Here we've loaded a second data table named **Returns**, containing records of returns by date, product and store

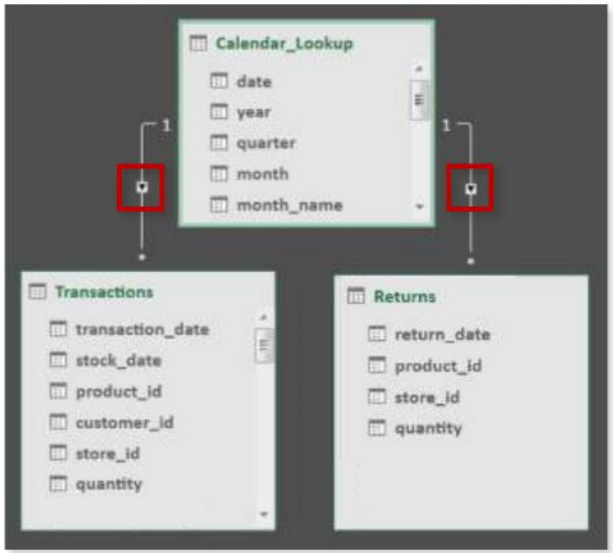
- This table connects to each lookup exactly like the **Transactions** table did, except that there is no way to connect the Returns table to Customer_Lookup
- This allows us to analyze data across both tables in the same pivot, **as long as we only filter or segment the data using lookups that are common to both**
  - In other words, we know which ***product*** was returned, which ***store*** it was returned to, and which ***date*** the return occurred, but NOT which ***customer*** was responsible

HEY THIS IS IMPORTANT!

NEVER try to connect data tables directly to each other;
ALWAYS connect them indirectly via shared lookup tables!

AI Quest

FILTER DIRECTION IS IMPORTANT



This model includes two data tables (**Transactions** and **Returns**), both connected to the **Calendar_Lookup**

Note the filter directions (shown as arrows) in each relationship; **in Power Pivot (2016) these will *always* point from the "one" side of the relationship (lookups) to the "many" side (data tables)***

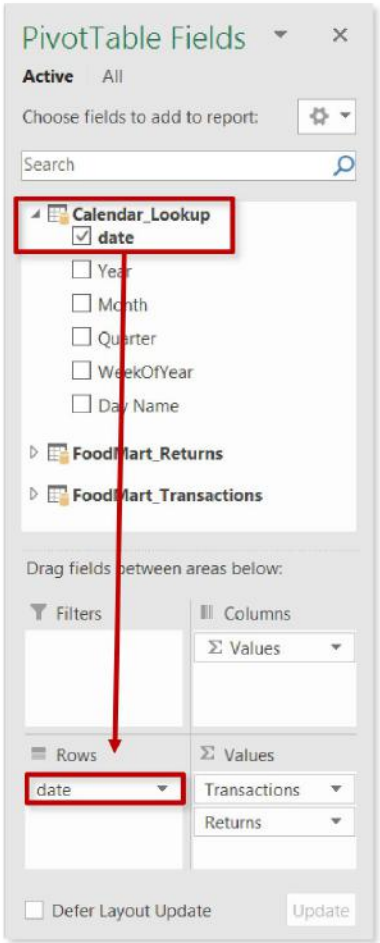
- Filtering a table will impact any tables "downstream" of it, as defined by the filter relationship (i.e the direction of the arrow)
- Let's say we're analyzing both Transactions and Returns in the same PivotTable; filtering by the **Calendar_Lookup** date field will return correctly filtered data from both data tables, but filtering by the **Transactions** date field will yield *unfiltered* Returns values

PRO TIP:

*Arrange your lookup tables **above** your data tables in diagram view to remind you that filters always flow "downstream"*

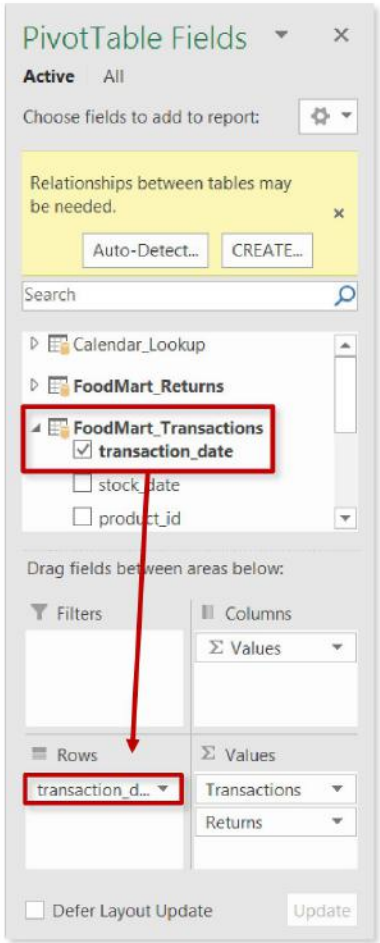
***Note:** In Excel 2010/2013 the diagram view looks a bit different, and arrows point in the opposite direction*

FILTER DIRECTION IS IMPORTANT



	A	B	C
1	Row Labels	Transactions	Returns
2	1/1/1997	348	3
3	1/2/1997	635	6
4	1/3/1997	589	7
5	1/4/1997	20	
6	1/5/1997	966	10
7	1/6/1997	993	11
8	1/7/1997	1,265	8
9	1/8/1997	35	
10	1/9/1997	525	9
11	1/10/1997	460	5

***Calendar_Lookup** filters flow "down" to both the **Transactions** and **Returns** tables, so we can filter or segment those metrics using any field from the Calendar table*

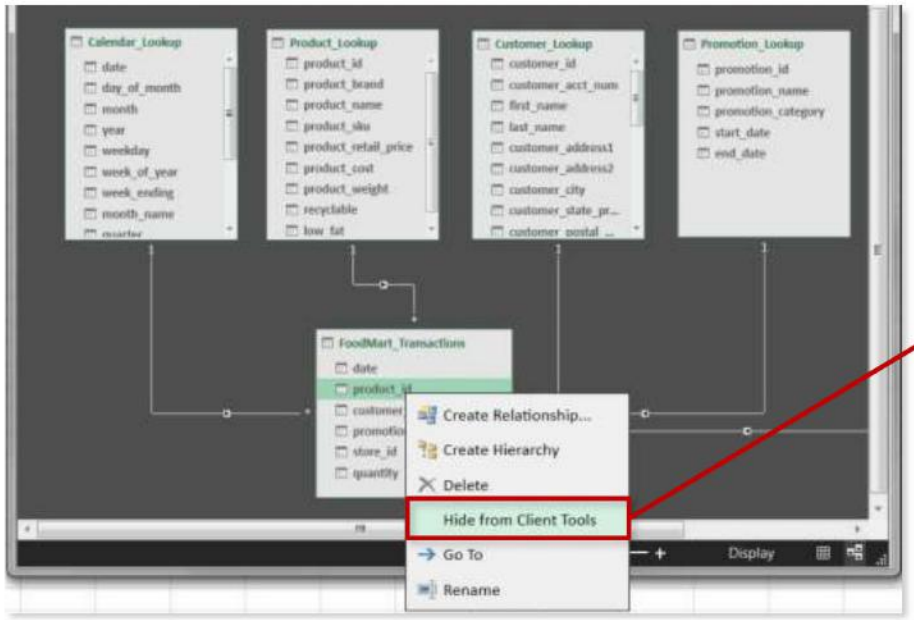


	A	B	C
1	Row Labels	Transactions	Returns
2	1/1/1997	348	8,289
3	1/2/1997	635	8,289
4	1/3/1997	589	8,289
5	1/4/1997	20	8,289
6	1/5/1997	966	8,289
7	1/6/1997	993	8,289
8	1/7/1997	1,265	8,289
9	1/8/1997	35	8,289
10	1/9/1997	525	8,289
11	1/10/1997	460	8,289

*Filtering by date in the **Transactions** table yields incorrect, unfiltered values from the **Returns** table, since filter context cannot flow "upstream" to the Calendar table*

AI Quest

HIDING FIELDS FROM CLIENT



When you **hide a field from Client Tools**, you make it invisible to tools outside of the data model (i.e. Power Pivot)

This can be used to prevent users from filtering or segmenting on invalid fields, or to hide irrelevant metrics from view

PRO TIP:

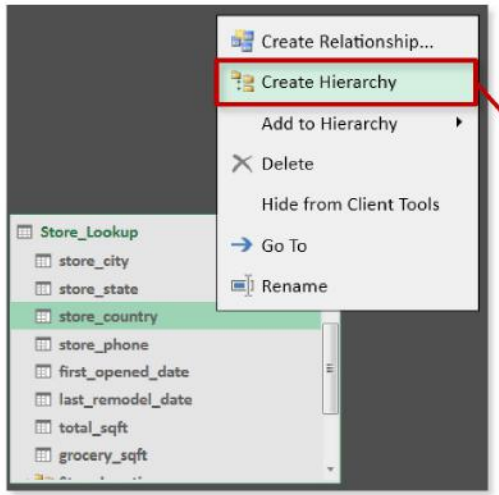
*Always hide the **foreign key columns** in your data tables to prevent users from accidentally filtering on them!*

AI Quest

DEFINING HIERARCHIES

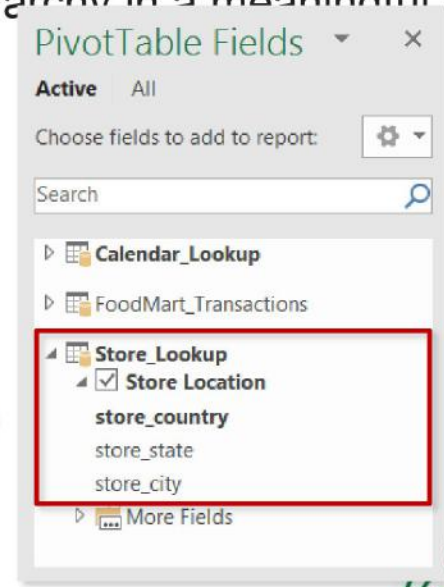
Hierarchies are groups of nested columns that reflect multiple levels of granularity

- For example, a "**Geography**" hierarchy might include **Country**, **State**, and **City** columns
- Each hierarchy is treated as a **single item** in PivotTables and PivotCharts, allowing users to "drill up" and "drill down" through different levels of the hierarchy in a meaningful way



***Right-click** a field to see the hierarchy options*

***Drag fields** to create a hierarchy*



*Hierarchies appear in **Power Pivot***

DATA MODEL BEST PRACTICES

Normalize your data model before you do anything else

- *Make sure that each table in your model serves a single, distinct purpose*
- *Use relationships vs. merged tables; long & narrow tables are better than short & wide*

Organize lookup tables *above* data tables in the diagram view

- *This serves as a visual reminder that filters always flow "**downstream**"*

Hide fields from client tools to prevent invalid filter context

- *All foreign key columns should be hidden from data tables, so that users only able to use valid fields for filtering and segmentation*

AI Quest

POWER PIVOT & DAX

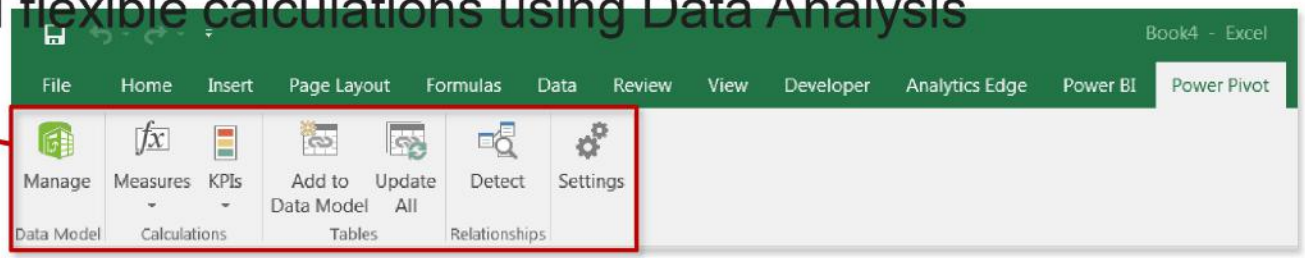
AI Quest

MEET POWER PIVOT

A **"Power" Pivot** is just like a normal PivotTable, except it sits on top of an *entire data model* rather than a single table or range. This allows you to:

- Explore massive datasets consisting of multiple sources and tables, using familiar, user-friendly PivotTable tools and options
- Create powerful and flexible calculations using Data Analysis Expressions (DAX)

The **Power Pivot** tab includes tools to manage the data model and define new measures

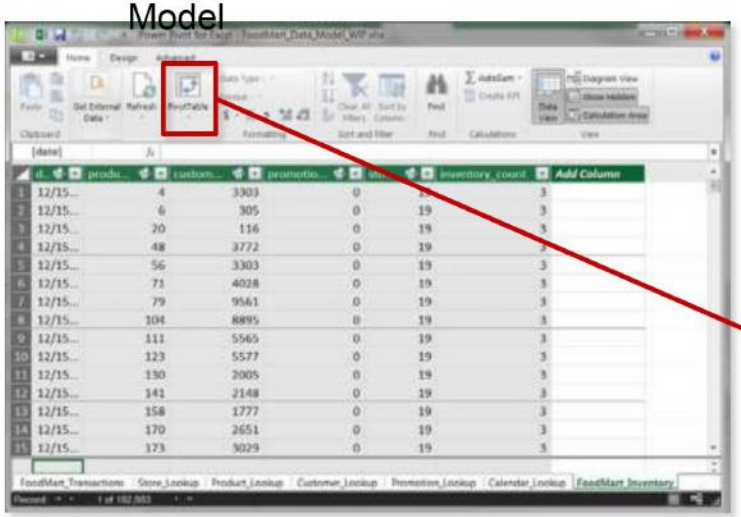


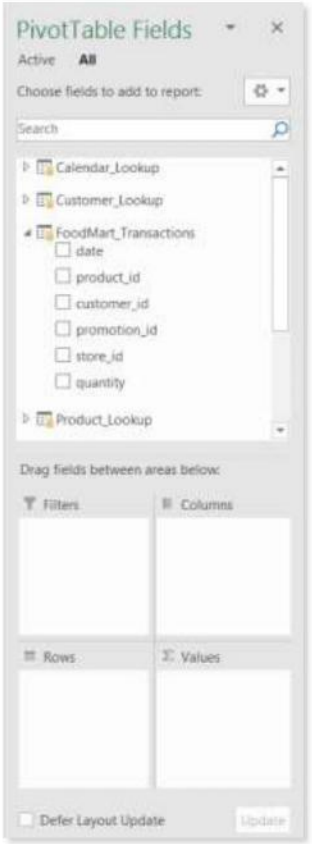
(***Note:*** *you may need to enable this tab by selecting* ***File > Options > Add-Ins > Manage COM Add-Ins***)

AI Quest

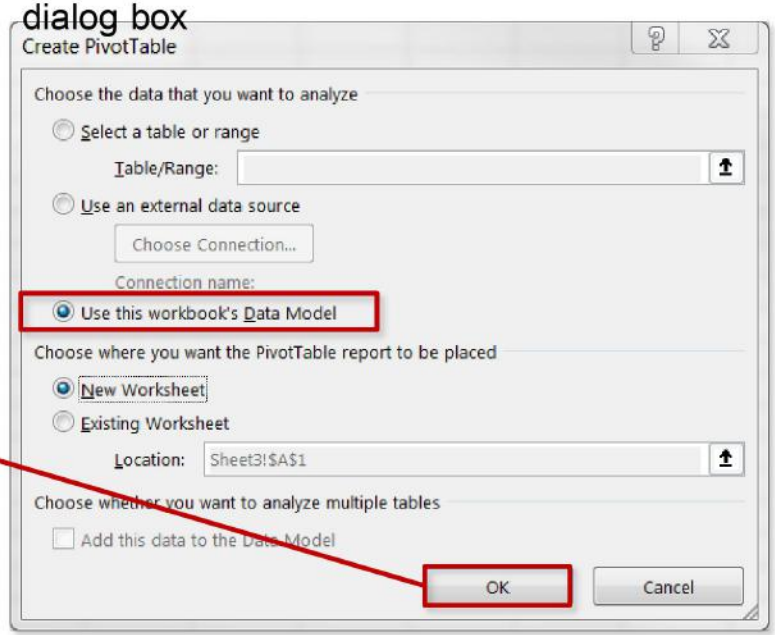
CREATING A "POWER" PIVOT TABLE

Option #1: From the Data Model





Option #2: From the *Insert* > *PivotTable* dialog box



AI Quest

"NORMAL" PIVOTS VS. "POWER"

NORMAL

- Can analyze data from **one table at a time**; multiple tables must be flattened or "stitched" together with cell functions
- Restricted to the data capacity of a **single Excel worksheet** (1,048,576 rows)
- Limited to relatively **basic calculated fields**, using a sub-set of Excel functions

POWER

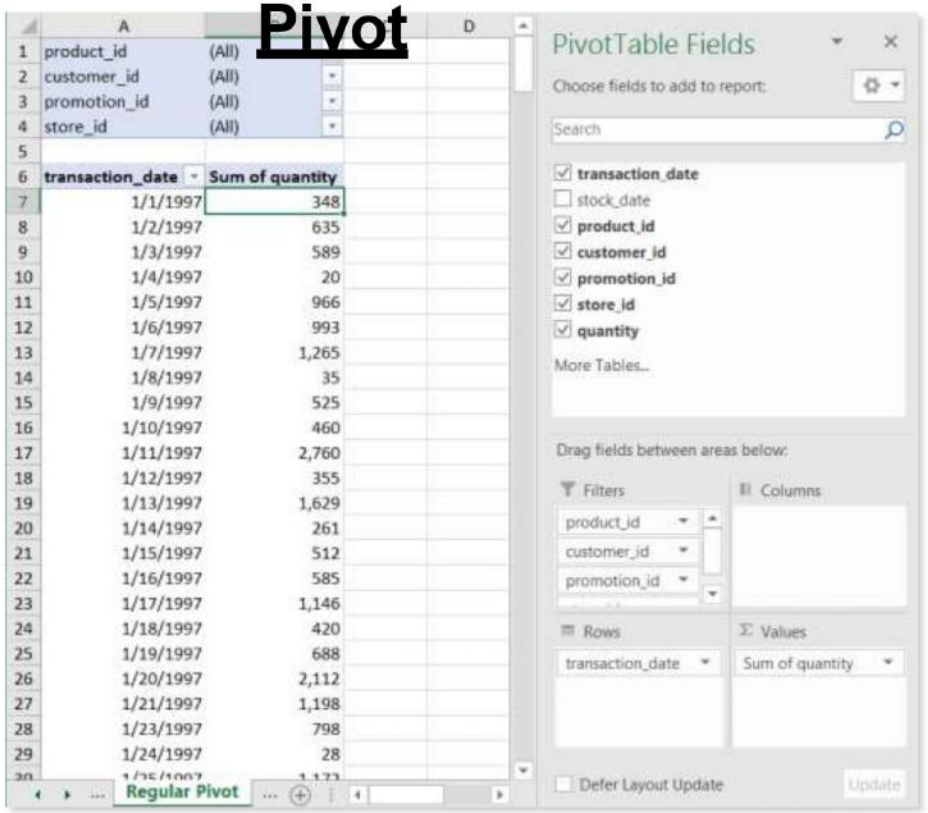
- Can analyze an **entire data model**, consisting of multiple tables connected via relationships rather than cell functions
- Virtually **unlimited data capacity** as tables are compressed outside of normal worksheets
- Performs **complex calculations** using Data Analysis Expressions (DAX)

NOTE: It's not the ***PivotTable*** itself that's different; it's the ***data behind it***

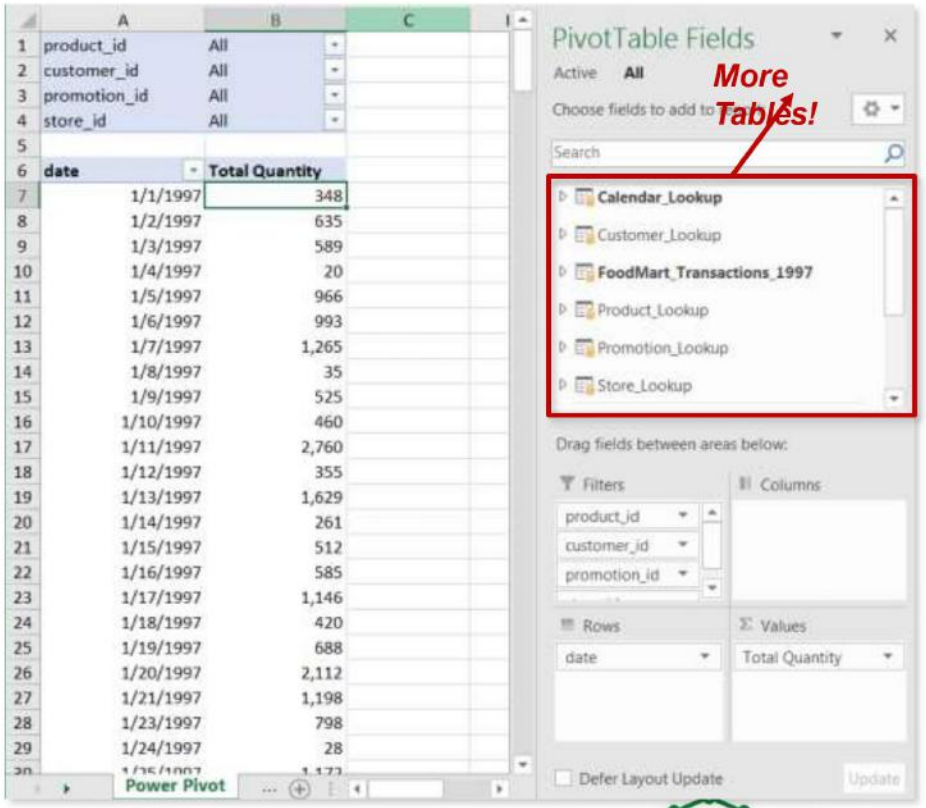
AI Quest

“NORMAL” PIVOTS VS. “POWER” PIVOTS

Normal Pivot



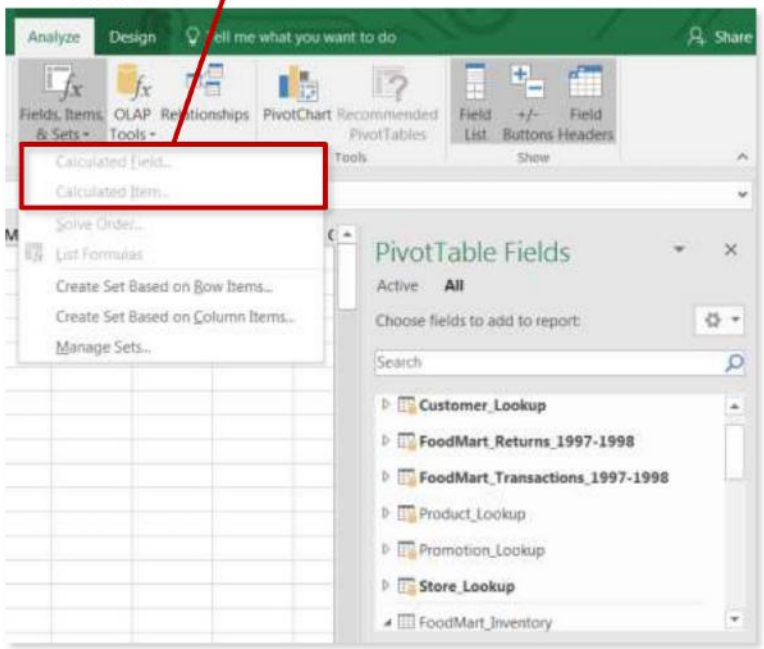
Power



AI Quest

NO MORE "CALCULATED FIELDS"

*Oh rats, where are my **calculated fields**??*



One of the key Power Pivot features is the ability to create *much* more robust calculated fields, known as **measures***

Because these measures interact directly with the data model (including tables stored in memory), traditional cell formulas won't do the trick

- Instead, we'll use a new (but familiar) formula language called **Data Analysis Expressions** (DAX)

***Note:** Depending on the version of Excel you're using, you might see these referred to as either "**Measures**" (Excel 2010, 2016) or "**Calculated Fields**" (Excel 2013)*

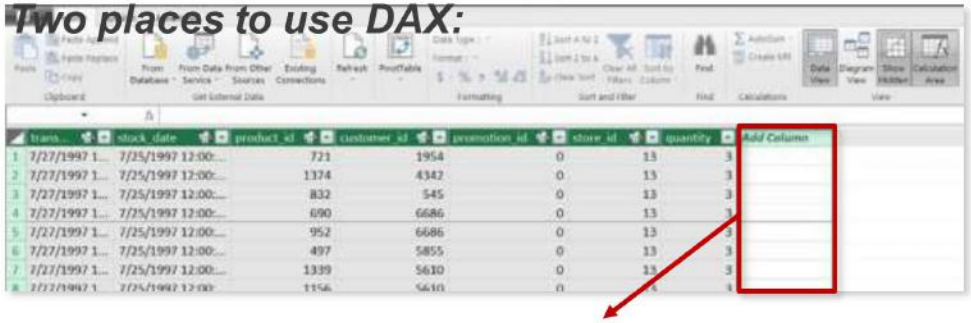
AI Quest

DATA ANALYSIS

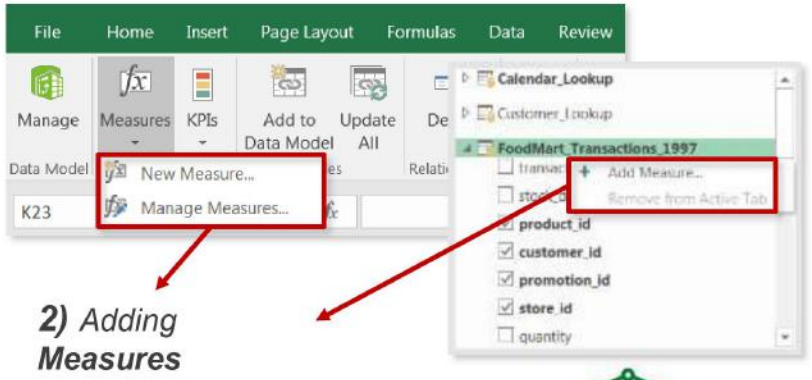
Data Analysis Expressions, commonly known as **DAX**, is the formula language that drives Power Pivot. With DAX, you can:

- Add **calculated columns** and **measures** to your model, using intuitive syntax
- Go beyond the capabilities of traditional “grid-style” formulas, with powerful functions built specifically to work with relational data

Two places to use DAX:



1)** Adding **Calculated Columns



2)** Adding **Measures

AI Quest

CALCULATED COLUMNS

Calculated columns allow you to add new, formula-based columns to tables

- No "A1-style" references; calculated columns refer to **entire tables** or **columns**
- Calculated columns are computed at the row-level, and **values are stored with the table** (*this eats up memory*)
- Calculated columns understand **row context**; they're great for defining new properties based on information in each row, but generally useless for aggregation (*SUM, AVERAGE, COUNT, etc.*)

HEY THIS IS IMPORTANT!

As a rule of thumb, **ONLY** use calculated columns if you want to "stamp" static, fixed values to each row in a table (*or use Power Query!*)

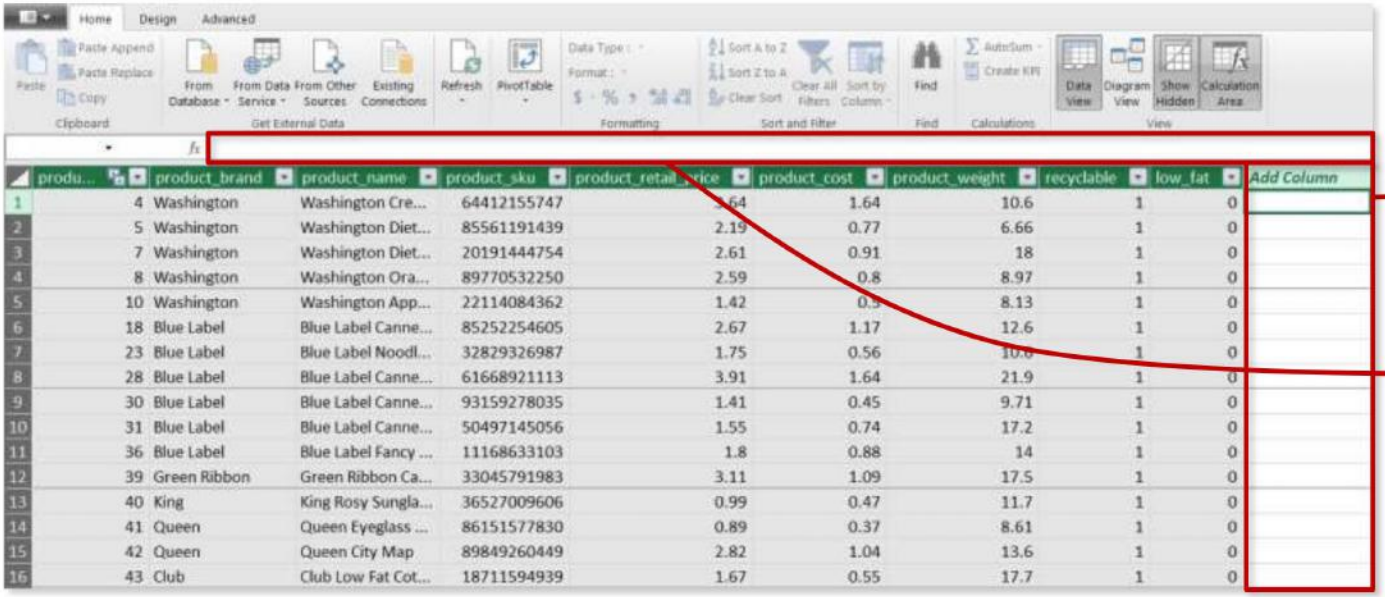
DO NOT use calculated columns for aggregation formulas, or to calculate fields for the "Values" area of a pivot (use **measures** instead)

PRO TIP:

*Calculated columns are typically placed in the **Filters, Slicers, Rows** or **Columns** areas of a pivot*

AI Quest

CREATING CALCULATED COLUMNS



	produ...	product_brand	product_name	product_sku	product_retail_price	product_cost	product_weight	recyclable	low_fat	Add Column
1	4	Washington	Washington Cre...	64412155747	1.64	1.64	10.6	1	0	
2	5	Washington	Washington Diet...	85561191439	2.19	0.77	6.66	1	0	
3	7	Washington	Washington Diet...	20191444754	2.61	0.91	18	1	0	
4	8	Washington	Washington Ora...	89770532250	2.59	0.8	8.97	1	0	
5	10	Washington	Washington App...	22114084362	1.42	0.5	8.13	1	0	
6	18	Blue Label	Blue Label Canne...	85252254605	2.67	1.17	12.6	1	0	
7	23	Blue Label	Blue Label Noodl...	32829326987	1.75	0.56	10.6	1	0	
8	28	Blue Label	Blue Label Canne...	61668921113	3.91	1.64	21.9	1	0	
9	30	Blue Label	Blue Label Canne...	93159278035	1.41	0.45	9.71	1	0	
10	31	Blue Label	Blue Label Canne...	50497145056	1.55	0.74	17.2	1	0	
11	36	Blue Label	Blue Label Fancy ...	11168633103	1.8	0.88	14	1	0	
12	39	Green Ribbon	Green Ribbon Ca...	33045791983	3.11	1.09	17.5	1	0	
13	40	King	King Rosy Sungla...	36527009606	0.99	0.47	11.7	1	0	
14	41	Queen	Queen Eyeglass ...	86151577830	0.89	0.37	8.61	1	0	
15	42	Queen	Queen City Map	89849260449	2.82	1.04	13.6	1	0	
16	43	Club	Club Low Fat Cot...	18711594939	1.67	0.55	17.7	1	0	

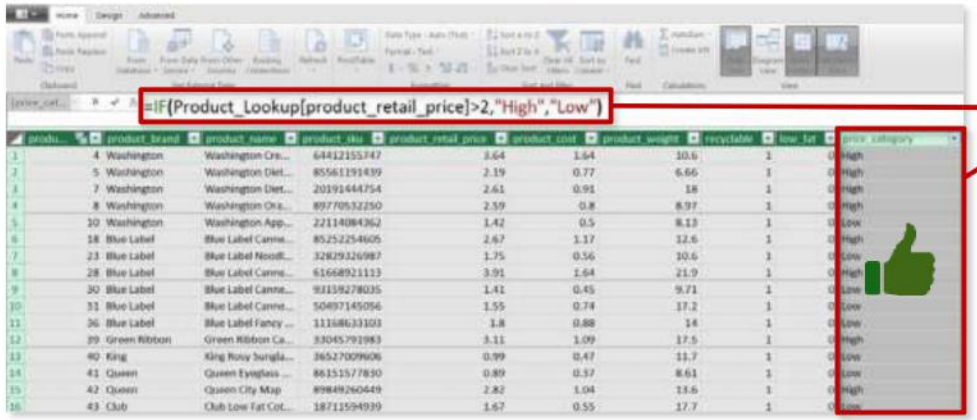
Step 1: In the data model "Data View", choose a table and then select any cell in the "**Add Column**" section

Step 2: Enter a DAX function in the formula bar (*we'll cover specific functions in the next section*)

Step 3: Press "Enter", and all cells in the column will update

AI Quest

CALCULATED COLUMNS: GOOD & BAD

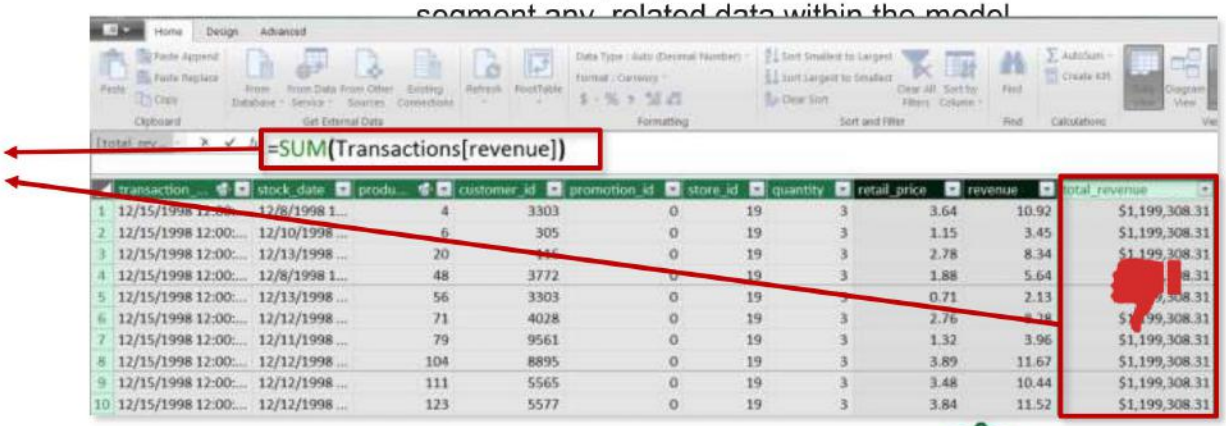


In this case we've added a **calculated column** called **price_category**, which equals "***High***" if the retail price is >$2, and "***Low***" otherwise (*just like you would write in Excel!*)

- Since calculated columns understand **row context**, a new value is calculated in each row based on that row's price
- This is a **valid use** of calculated columns; it creates a new row "property" that we can now use to filter or segment any related data within the model

Here we're using an aggregation function (SUM) to calculate a new column named **total_revenue**

- Since calculated columns do not understand **filter context**, the same grand total is returned in *every single row* of the table
- This is **not a valid use** of calculated columns; these values are statically "stamped" onto the table and can't be filtered, sliced, subdivided, etc.



AI Quest

DAX

MEASURES

Measures are DAX formulas used to generate dynamic values within a PivotTable

- Like calculated columns, measures reference **entire tables** or **columns** (*no A1-style or "grid" references*)
- Unlike calculated columns, measures don't actually *live* in the table; they get placed in the **values** area of a PivotTable and dynamically calculated in each individual cell
- Measures are evaluated based on the **filter context** of each cell, which is determined by the PivotTable layout (filters, slicers, rows and columns)

HEY THIS IS IMPORTANT!

As a rule of thumb, use measures (*vs. calculated columns*) when a single row can't give you the answer (i.e. requires **aggregation**)

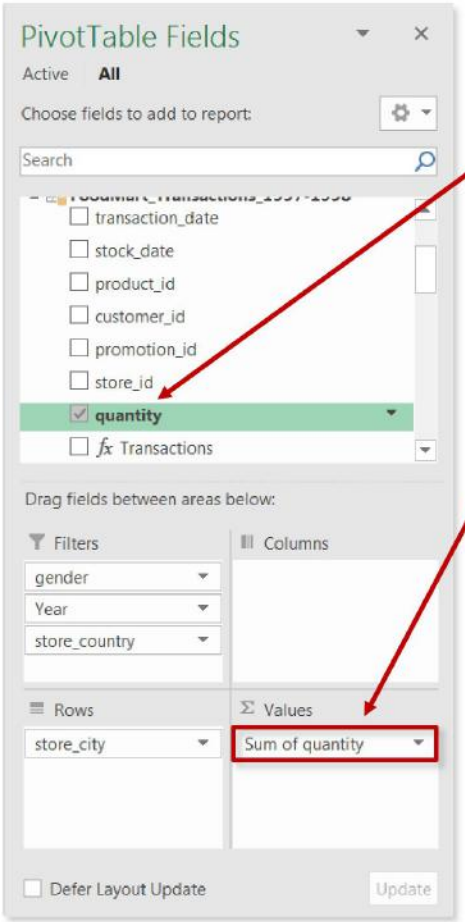
Measures can **ONLY** be placed in the values area of a PivotTable

PRO TIP:

*Use measures to create values **that users can explore with a pivot** (Power Pivot version of a "Calculated Field")*

AI Quest

CREATING IMPLICIT MEASURES



STEP 1: Check the box next to a value field in a data table, or manually drag it into the "Values" box

STEP 2: Pat yourself on the back, you just created a measure!

HEY THIS IS IMPORTANT!

Before you pop the champagne, there's a catch. When you drag a raw data field into the values section of a pivot, you create what's called an **implicit measure**. While there's nothing *wrong* with implicit measures, they are extremely limited.

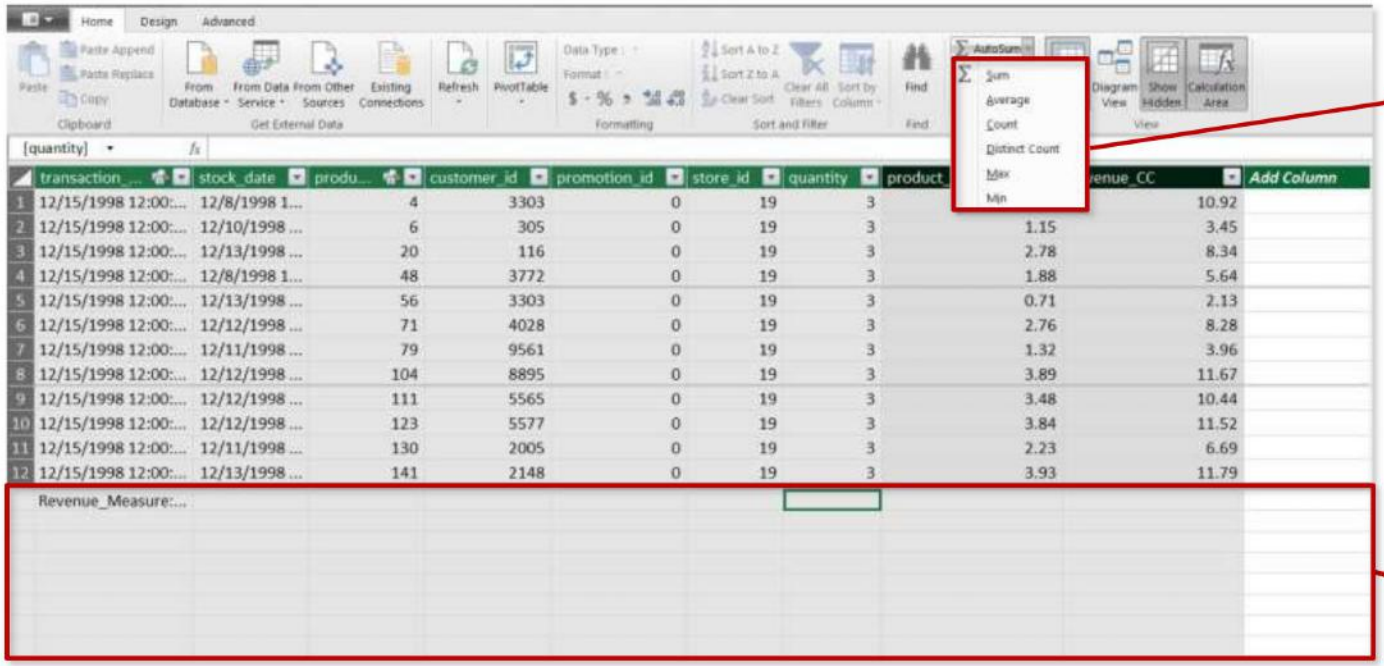
Explicit measures (defined using DAX) will give us *much* more flexibility, as well as the ability to reuse measures in multiple places (measure trees!)

FROM NOW ON, JUST SAY "NO" TO IMPLICIT MEASURES

AI Quest

CREATING EXPLICIT MEASURES

AUTOSUM



AutoSum is a shortcut for creating simple DAX formulas (*Sum, Average, Count, Distinct Count, Max and Min*)

To use AutoSum:

- Click on a cell in the Measures Pane (*see below*), within the column you want to evaluate
- Select the **AutoSum** menu and choose an option from the list

*The **Measures Pane** sits beneath the data in the "Data View" of the model*

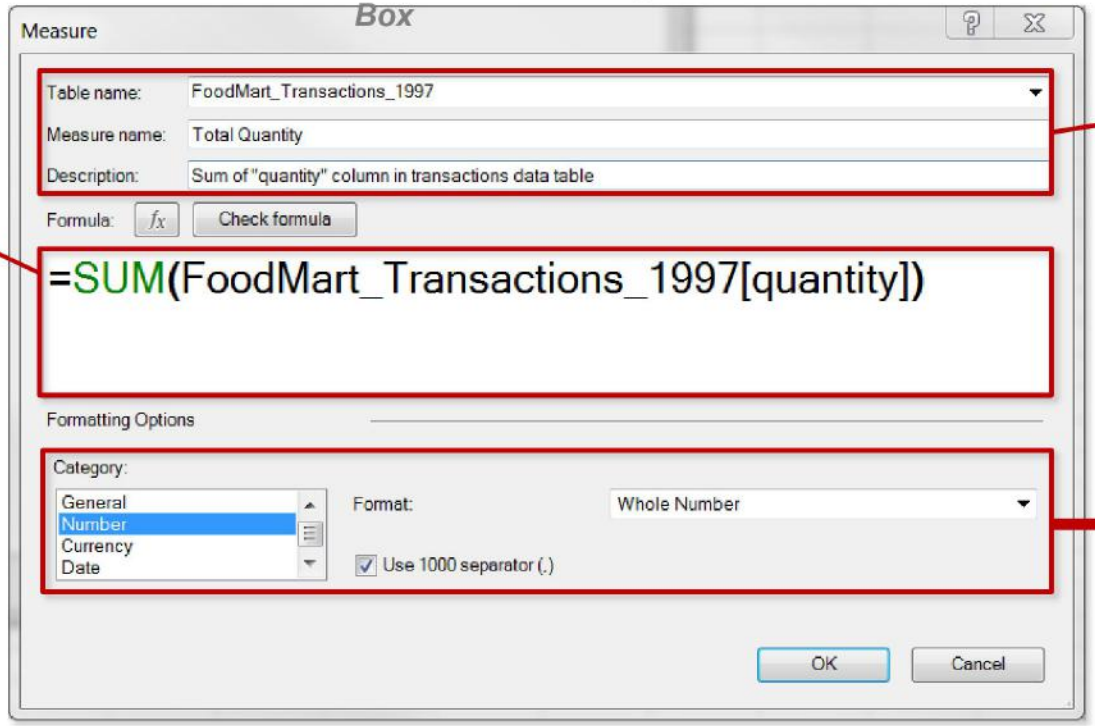
PRO TIP:

***AutoSum** is a nice way to get comfortable with basic DAX and quickly add measures; just don't rely on them when things start to get more complicated!*

AI Quest

CREATING EXPLICIT MEASURES

The Measure Dialog Box



The **Formula** pane contains the actual DAX code, as well as options to browse the formula library or check syntax

Note: *just start typing, and "Intellisense" will kick in to help you auto-populate formula names and tables*

PRO TIP:

Ctrl+scroll *adjusts formula text size*

Each measure is **assigned to a table** and given a **measure name** (as well as an optional description)

Use the **Formatting Options** to specify a format for each measure

AI Quest

UNDERSTANDING FILTER CONTEXT

Measures are calculated based on **filter context**, which is the set of filters (or "*coordinates*") determined by the PivotTable layout (filters, slicers, row labels and column labels)

HEY THIS IS IMPORTANT!

Each measure cell in the pivot **calculates independently**, based on its coordinates (*think of each cell as an island*)

When you change the pivot layout (by updating filters/slicers, row labels or column labels), each measure cell **detects its new coordinates** and then **recalculates its value**

customer_city	Total Quantity
Acapulco	16,428
Camacho	26,024
Hidalgo	52,888
La Cruz	10,251
Merida	40,994
Mexico City	10,666
Orizaba	27,334
San Andres	10,861
Santa Anita	11,834
Santa Fe	4,717
Tixapan	12,440
Guadalajara	2,401
Grand Total	**226,838**

The coordinate for this measure cell is **Customer_Lookup[customer_city] = "Hidalgo"**

- Given this coordinate, Excel filters down to the "Hidalgo" rows in the **Customer_Lookup** table, filters all *related* tables (based on the relationships in data model), then evaluates the arithmetic in the table defined by the measure (*in this case **Total Quantity** equals the **sum of quantity** from the transactions data table*)

This cell does NOT add up the values above it (*it's an island, remember?*)

- Total rows represent a **lack of filters**; since this cell does *not* have a customer_city coordinate, it evaluates the Total Quantity measure across the entire, unfiltered

Quest

FILTER CONTEXT

EXAMPLES

customer_id	All
Year	1997
Month	All
customer_country	USA

customer_city	Total Quantity
Albany	6,806
Altadena	2,574
Anacortes	766

Cell coordinates:

- Calendar_Lookup[Year] = **1997**
- Customer_Lookup[customer_country] = **"USA"**
- Customer_Lookup[customer_city] = **"Altadena"**

customer_country

- Canada
- Mexico
- USA

Year	Quarter	Total Quantity
⊟1997		266,773
	1	66,291
	2	62,610
	3	65,848
	4	72,024
⊟1998		289,126
	1	69,785
	2	68,855
	3	68,574
	4	81,912
Grand Total		555,899

Cell coordinates:

- Calendar_Lookup[Year] = **1997**
- Customer_Lookup[customer_country] = **"USA"**

Cell coordinates:

- Calendar_Lookup[Year] = **1998**
- Calendar_Lookup[Quarter] = **1**
- Customer_Lookup[customer_country] = **"USA"**

Cell coordinates:

- Customer_Lookup[customer_country] = **"USA"**

store_country	Canada

Total Quantity	store_city		
product_brand	Vancouver	Victoria	Grand Total
ADJ	30	10	40
Akron	50	15	65
American	334	103	487
Amigo	50	31	81

Cell coordinates:

- Store_Lookup[store_country] = **"Canada"**
- Store_Lookup[store_city] = **"Vancouver"**
- Product_Lookup[product_brand] = **"Akron"**

Cell coordinates:

- Store_Lookup[store_country] = **"Canada"**
- Product_Lookup[product_brand] = **"Amigo"**

AI Quest

STEP-BY-STEP MEASURE CALCULATION

Row Labels	Total Quantity
CANADA	50,752
MEXICO	226,838
USA	555,899
Grand Total	**833,489**

How *exactly* is this measure calculated?

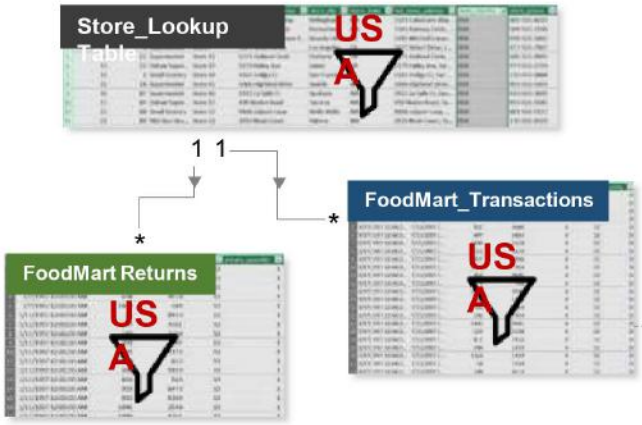
- **REMEMBER:** This all happens *instantly* behind the scenes, every time a measure cell calculates

STEP 1

Detect pivot coordinates & apply filter context

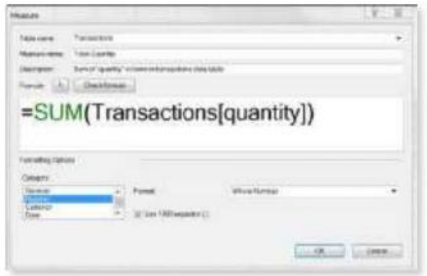
STEP 2

Carry filters "downstream" & apply to all related tables



STEP 3

Evaluate the measure formula against the filter context table



Sum of Transactions[quantity] when store_country = "USA"

=

555,899

AI Quest

RECAP: CALCULATED COLUMNS VS. MEASURES

CALCULATED COLUMNS

- Evaluated in the context of each row of the table to which it belongs (has **row context**)
- Appends static values to each row in a table and stores them in the model, increasing file size
- Only recalculated on data source refresh or changes to component columns
- Primarily used as **rows**, **columns**, **slicers** or **filters**

*Calculated columns "live" in **tables***

MEASURES

- Evaluated in the context of each cell of the PivotTable in which it is displayed (has **filter context**)
- Does not create new data in the tables themselves, and does not increase file size
- Recalculated in response to any change in the PivotTable view
- Can *only* be used as PivotTable **values**

store_country	Canada		
Total Quantity	store_city		
product_brand	Vancouver	Victoria	Grand Total
ADJ	30	10	40
Akron	50	15	65
American	384	103	487
Amigo	50	31	81

*Measures "live" in **PivotTables***

***Note:** Calculated columns CAN be placed in the values area of a pivot, but you can (and should) use a measure instead*

AI Quest

POWER PIVOT BEST PRACTICES

Avoid using implicit measures whenever possible

- *Implicit measures are limited in functionality and restricted to the pivot in which they were created; explicit measures are more **portable** and **powerful***

Don't use a calculated column when a measure will do the trick

- *Only use calculated columns to "stamp" static, fixed values to each row in a table*
- *Use measures when aggregation is necessary, or to create dynamic values in a pivot*

Know your data model inside and out!

- *It's easy to produce incorrect results in Power Pivot if you don't respect the model's table relationships, and errors are often difficult to spot without a*

COMMON DAX FUNCTIONS

AI Quest

DAX SYNTAX

MEASURE NAME

- ***Note:*** *Measures are always surrounded in brackets (i.e.* ***[Total Quantity]****) when referenced in formulas, so spaces are OK*

Total Quantity:

=SUM(Transactions[quantity])

FUNCTION NAME

- *Calculated columns don't always use functions, but measures do:*
  - *In a calculated column,* ***=Transactions[quantity]*** *returns the value from the quantity column in each row (since it evaluates for each row)*
  - *In a measure,* ***=Transactions[quantity]*** *will return an* ***error*** *since Excel doesn't know how to evaluate that as a single value in a pivot (you need some sort of aggregation)*

Referenced **TABLE NAME**

Referenced **COLUMN NAME**

This is a ***"fully qualified"*** *column, since it's preceeded by the table name*

- ***Note:*** *Table names with spaces must be surrounded by* ***single quotes****:*
  - *Without a space:* ***Transactions****[quantity]*
  - *With a space:* ***'Transactions Table'****[quantity]*

PRO TIP:

For ***column*** *references, use the fully qualified name (i.e.* ***Table[Column]****) For* ***measure*** *references, just use the measure name (i.e.* ***[Measure]****)*

AI Quest

DAX

OPERATORS

Arithmetic Operator	Meaning	*Example*
+	Addition	*2 + 7*
-	Subtraction	*5 – 3*
*	Multiplication	*2 * 6*
/	Division	*4 / 2*
^	Exponent	*2 ^ 5*

Hey! Pay attention to these!

Comparison Operator	Meaning	*Example*
=	Equal to	*[City]="Boston"*
>	Greater than	*[Quantity]>10*
<	Less than	*[Quantity]<10*
>=	Greater than or equal to	*[Unit_Price]>=2.5*
<=	Less than or equal to	*[Unit_Price]<=2.5*
<>	Not equal to	*[Country]<>"Mexico"*

Text/Logical Operator	Meaning	*Example*
&	Concatenates two values to produce one text string	*[City] & " " & [State]*
&&	Create an **AND** condition between two logical expressions	*([State]="MA") && ([Quantity]>10)*
|| *(double pipe)*	Create an **OR** condition between two logical expressions	*([State]="MA") || ([State]="CT")*
IN	Creates a logical **OR** condition based on a given list (using curly brackets)	*'Store Lookup'[State] **IN** { "MA", "CT", "NY" }*

Head to **www.msdn.microsoft.com for more information about DAX syntax, operators, troubleshooting, etc.*

AI Quest

COMMON FUNCTION CATEGORIES

MATH & STATS

*Basic **aggregation** functions as well as "**iterators**" evaluated at the row-level*

Common Examples:

- SUM
- AVERAGE
- MAX/MIN
- DIVIDE
- COUNT/COUNTA
- COUNTROWS
- DISTINCTCOUNT

Iterator Functions:

- SUMX
- AVERAGEX
- MAXX/MINX
- RANKX
- COUNTX

LOGICAL

*Functions for returning information about values in a given **conditional expression***

Common Examples:

- IF
- IFERROR
- AND
- OR
- NOT
- SWITCH
- TRUE
- FALSE

TEXT Function

*Functions to manipulate **text strings** or **control formats** for dates, times or numbers*

Common Examples:

- CONCATENATE
- FORMAT
- LEFT/MID/RIGHT
- UPPER/LOWER
- PROPER
- LEN
- SEARCH/FIND
- REPLACE
- REPT
- SUBSTITUTE
- TRIM
- UNICHAR

FILTER Function

***Lookup** functions based on related tables and **filtering** functions for dynamic calculations*

Common Examples:

- CALCULATE
- FILTER
- ALL
- ALLEXCEPT
- RELATED
- RELATEDTABLE
- DISTINCT
- VALUES
- EARLIER/EARLIEST
- HASONEVALUE
- HASONEFILTER
- ISFILTERED
- USERELATIONSHIP

DATE & TIME

*Basic **date and time** functions as well as advanced **time intelligence** operations*

Common Examples:

- DATEDIFF
- YEARFRAC
- YEAR/MONTH/DAY
- HOUR/MINUTE/SECOND
- TODAY/NOW
- WEEKDAY/WEEKNUM

Time Intelligence Functions:

- DATESYTD
- DATESQTD
- DATESMTD
- DATEADD
- DATESINPERIOD

***Note:** This is NOT a comprehensive list (does not include trigonometry functions, parent/child functions, information functions, or other less common functions)*

AI Quest

BASIC MATH & STATS FUNCTIONS

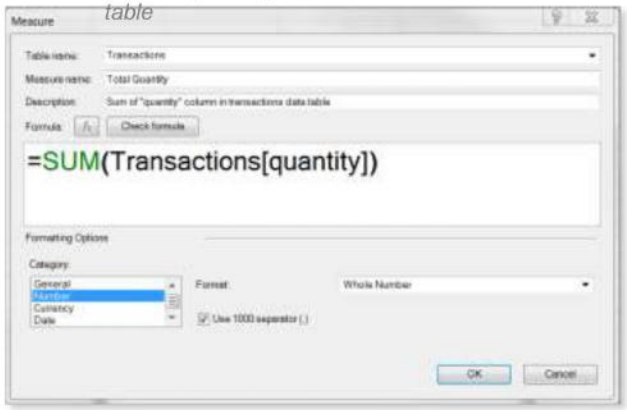
SUM( )	*Evaluates the sum of a column*	=**SUM**(<column>)
AVERAGE( )	*Returns the average (arithmetic mean) of all the numbers in a column*	=**AVERAGE**(<column>)
MAX( )	*Returns the largest value in a column or between two scalar expressions*	=**MAX**(<column>) *or* =**MAX**(<exp1>, <exp2>)
MIN( )	*Returns the smallest value in a column or between two scalar expressions*	=**MIN**(<column>) *or* =**MIN**(<exp1>, <exp2>)
DIVIDE ( )	*Performs division and returns the alternate result (or blank) if div/0*	=**DIVIDE**(<numerator>, <denominator>, <other>)

AI Quest

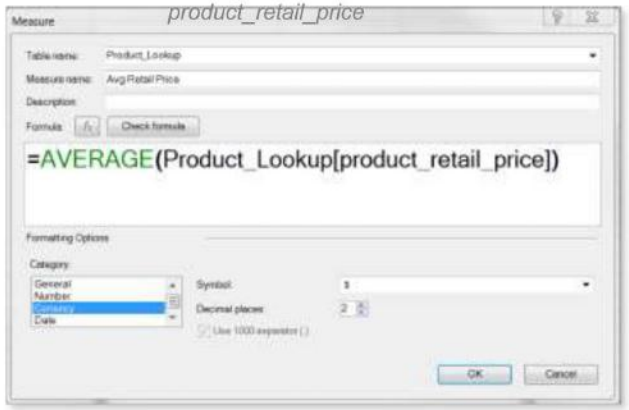
BASIC MATH & STATS FUNCTIONS

EXAMPLES

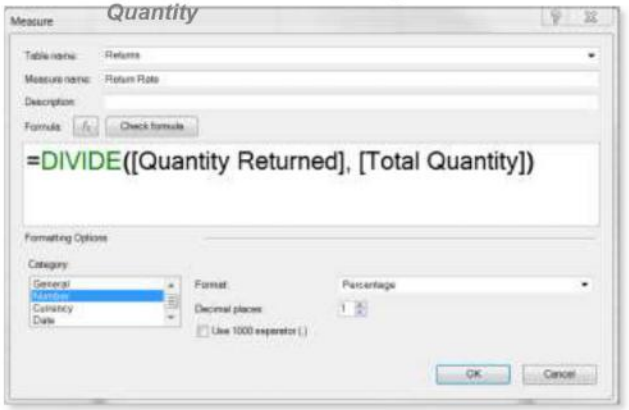
Sum of quantity *from the Transactions table*



Average *of product_retail_price*



Quantity Returned *divided by* ***Total Quantity***



PRO TIP:

Even though it might seem unnecessary, ***creating measures for even simple calculations*** *(like the sum of a column) allows you to use those measures within other calculations, anywhere in the workbook*

AI Quest

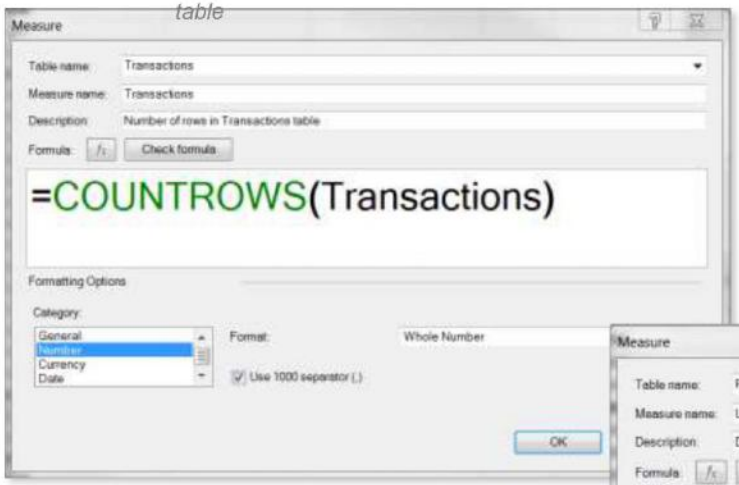
COUNT, COUNTA, DISTINCTCOUNT & COUNTROWS

COUNTROWS()	*Counts the number of rows in the specified table, or a table defined by an expression*	**=COUNTROWS(**<table>**)**
COUNT()	*Counts the number of cells in a column that contain numbers*	**=COUNT(**<column>**)**
COUNTA()	*Counts the number of non-empty cells in a column (numerical and non-numerical)*	**=COUNTA(**<column>**)**
DISTINCTCOUNT()	*Counts the number of different cells in a column of numbers*	**=DISTINCTCOUNT(**<column>**)**

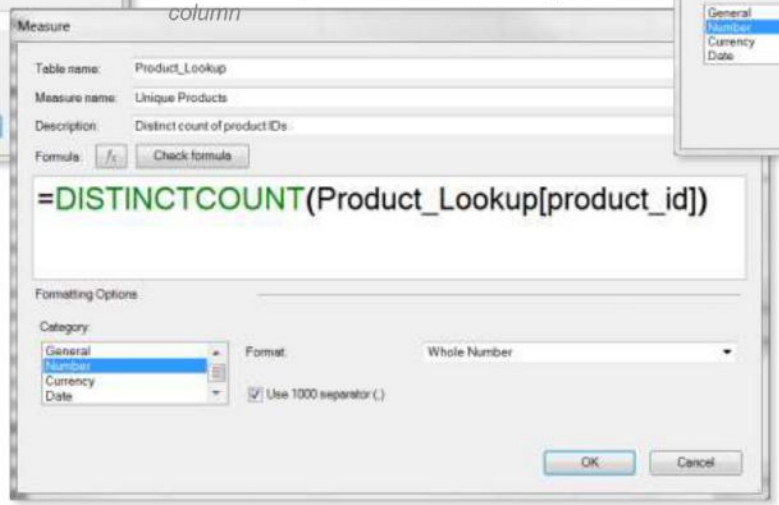
AI Quest

COUNT FUNCTIONS

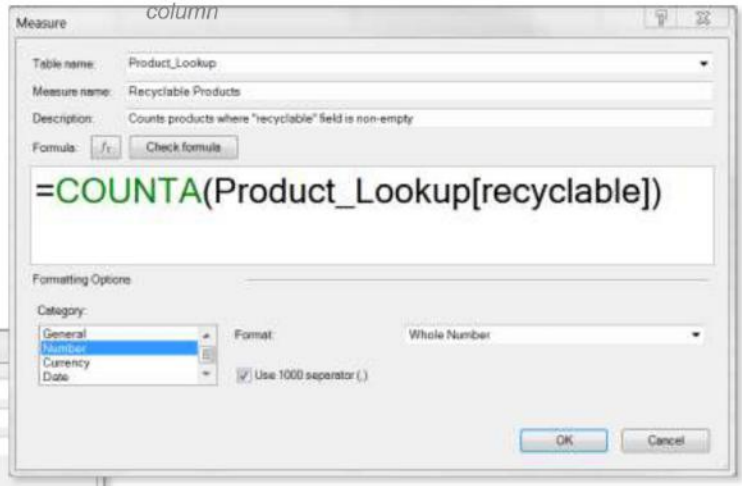
*Count of **all rows** in the Transactions table*



*Count of **unique values** in the product_id column*



*Count of **non-empty cells** in the recyclable column*



AI Quest

BASIC LOGICAL FUNCTIONS

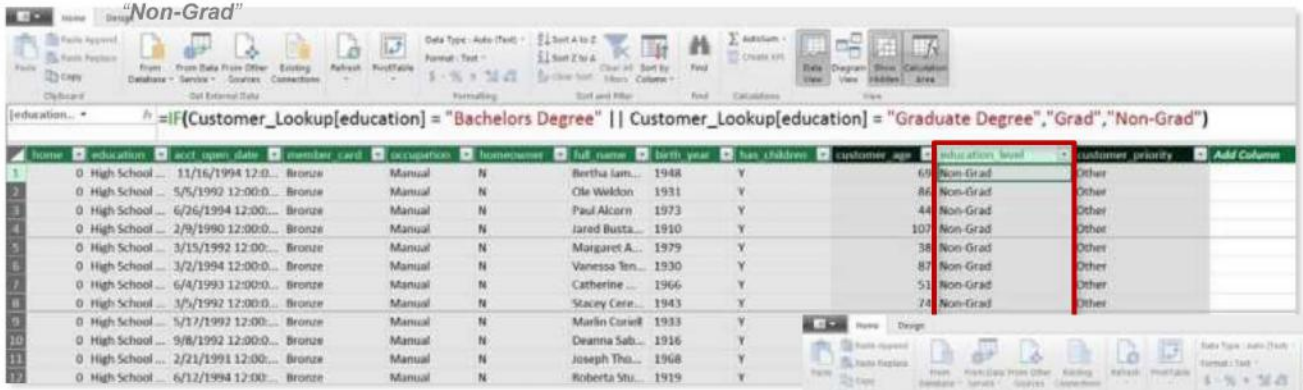
IF( )	*Checks if a given condition is met, and returns one value if the condition is TRUE, and another if the condition is FALSE*	=**IF**(<logical test>, <value_if_true>, <value_if_false>)
IFERROR()	*Evaluates an expression and returns a specified value if the expression returns an error, otherwise returns the expression itself*	=**IFERROR**(value, value_if_error)
AND( )	*Checks whether both arguments are TRUE, and returns TRUE if both arguments are TRUE, otherwise returns FALSE*	=**AND**(<logical1>, <logical2>)
OR( )	*Checks whether one of the arguments is TRUE to return TRUE, and returns FALSE if both arguments are FALSE*	=**OR** (<logical1>, <logical2>)

***Note:** Use the **&&** and || operators if you want to include more than two conditions!*

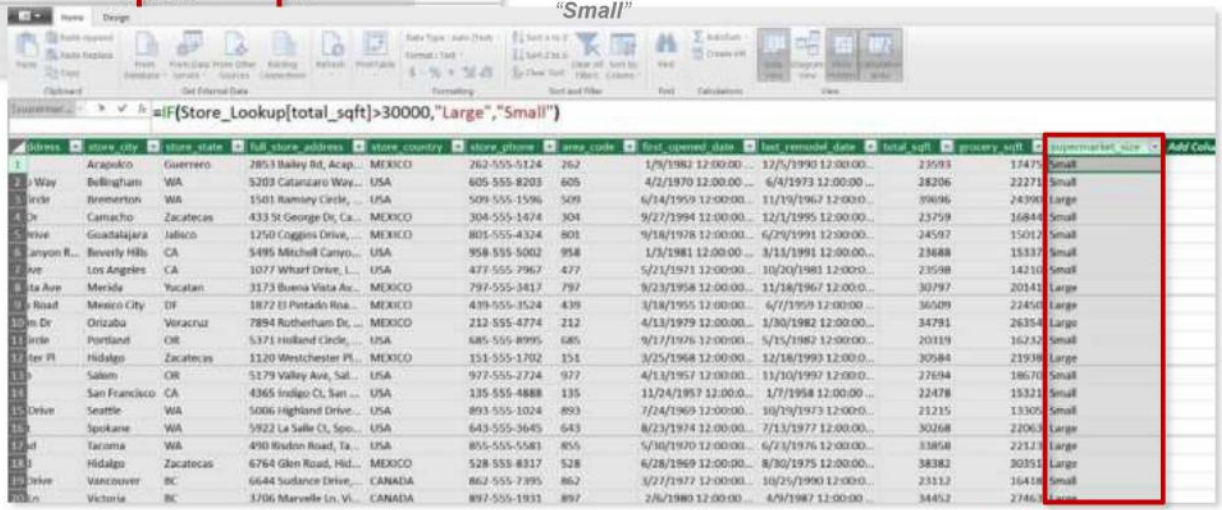
AI Quest

BASIC LOGICAL FUNCTIONS

*Education level equals "**Grad**" if customer has a bachelors degree or a graduate degree, otherwise "**Non-Grad**"*



*Supermarket_size equals "**Large**" if sq ft >30,000, otherwise "**Small**"*



AI Quest

SWITCH &

SWITCH

Evaluates an expression against a list of values and returns one of multiple possible result expressions

=**SWITCH**(<expression>, <value1>, <result1>, <value2>, <result2>, … <else>)

Any DAX expression that returns a single scalar value, evaluated multiple times (for each row/constant)

Examples:

- *Calendar_Lookup[month_num]*
- *Product_Lookup[product_brand]*

*List of **values** produced by the expression, each paired with a **result** to return for rows/cases that match*

Examples:

=SWITCH(Calendar_Lookup[month_num], 1, "**January**",
2,
"**February**",
etc…)

=SWITCH(TRUE(),
[retail_price]<5, "**Low Price**",
AND([retail_price>=5, [retail_price]<20), "**Med Price**", AND([retail_price>=20, [retail_price]<50),
"**High Price**" "**Premium Price**")

Value returned if the expression doesn't match any value argument

PRO TIP:

*Use the **SWITCH(TRUE()** combo to generate results based on Boolean (True/False) expressions (instead of those pesky nested IF statements!)*

AI Quest

SWITCH & SWITCH(TRUE)

*Switch quarter 1 with "**Q1**", quarter 2 with "**Q2**", quarter 3 = "**Q3**", else "**Q4**"*

```
=SWITCH(Calendar_Lookup[quarter],
1,"Q1",
2,"Q2",
3,"Q3",
"Q4")
```

	date	year	quarter	month	month_name	week_of_year	start_of_week	day	weekday	weekend	quarter_name
1	7/1/1997 12:0...	1997	3	7	July	27	6/30/1997 12:0...	1	Tuesday	0	Q3
2	7/2/1997 12:0...	1997	3	7	July	27	6/30/1997 12:0...	2	Wednesday	0	Q3
3	7/3/1997 12:0...	1997	3	7	July	27	6/30/1997 12:0...	3	Thursday	0	Q3
4	7/4/1997 12:0...	1997	3	7	July	27	6/30/1997 12:0...	4	Friday	0	Q3
5	7/5/1997 12:0...	1997	3	7	July	27	6/30/1997 12:0...	5	Saturday	1	Q3
6	7/6/1997 12:0...	1997	3	7	July	28	6/30/1997 12:0...	6	Sunday	1	Q3
7	7/7/1997 12:0...	1997	3	7	July						
8	7/8/1997 12:0...	1997	3	7	July						
9	7/9/1997 12:0...	1997	3	7	July						
10	7/10/1997 12:...	1997	3	7	July						
11	7/11/1997 12:...	1997	3	7	July						
12	7/12/1997 12:...	1997	3	7	July						

*Set product_price_category to "**High**" if retail price > $3, "**Medium**" if price is between $2 and $3, "**Low**" if price is <=$2, else "**Other**"*

```
=SWITCH(TRUE(),
Product_Lookup[product_retail_price]>3, "High",
Product_Lookup[product_retail_price]>2 && Product_Lookup[product_retail_price] <=3, "Medium",
Product_Lookup[product_retail_price]<=2, "Low",
"Other")
```

	produ...	product_brand	product_name	product_sku	product_retail_price	discount_retail_price	product_cost	product_weight	recyclable	low_fat	product_price_category
1	1	Washington	Washington Berr...	90748583674	$2.85	$2.28	$0.94	8.39			Medium
2	2	Washington	Washington Ma...	96516502499	$0.74	$0.59	$0.26	7.42		1	Low
3	3	Washington	Washington Stra...	58427771925	$0.83	$0.66	$0.40	13.1	1	1	Low
4	4	Washington	Washington Cre...	64412155747	$3.64	$2.91	$1.64	10.6	1		High
5	5	Washington	Washington Diet...	85561191439	$2.19	$1.75	$0.77	6.66	1		Medium
6	6	Washington	Washington Cola	29804642796	$1.15	$0.92	$0.37	15.8			Low
7	7	Washington	Washington Diet...	20191444754	$2.61	$2.09	$0.91	18	1		Medium
8	8	Washington	Washington Ora...	89770532250	$2.59	$2.07	$0.80	8.97	1		Medium

AI Quest

TEXT FUNCTIONS

LEN()	Returns the number of characters in a string	=LEN(<text>)
CONCATENATE()	Joins two text strings into one	=CONCATENATE(<text1>, <text2>)
LEFT/MID/RIGHT()	Returns a number of characters from the start/middle/end of a text string	=LEFT/RIGHT(<text>, <num_chars>) =MID(<text>, <start_num>, <num_chars>)
UPPER/LOWER/PROPER()	Converts letters in a string to upper/lower/proper case	=UPPER/LOWER/PROPER(<text>)
SUBSTITUTE()	Replaces an instance of existing text with new text in a string	=SUBSTITUTE(<text>, <old_text>, <new_text>, <instance>)
SEARCH()	Returns the position where a specified string or character is found, reading left to right	=SEARCH(<find_text>, <within_text>, <start_num>, <NotFoundValue>)

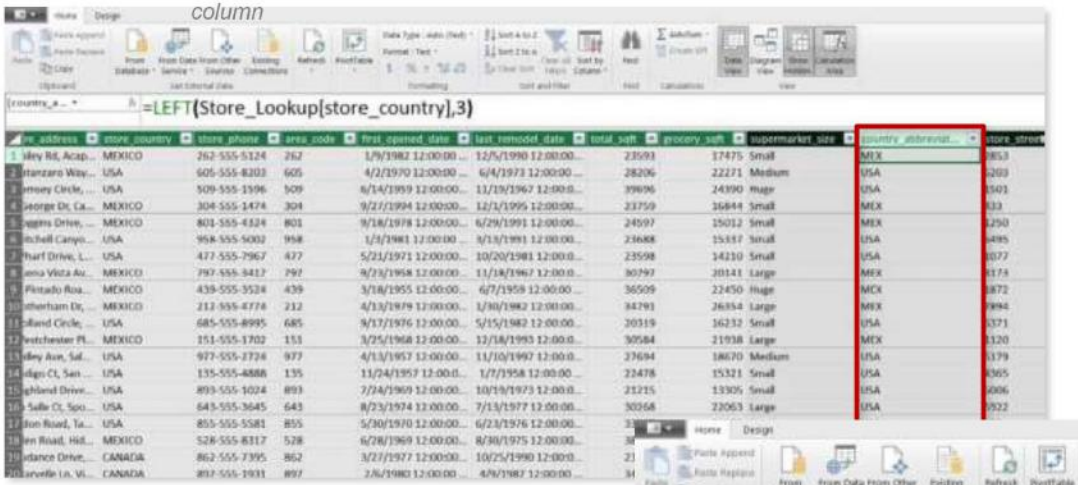
Note: *Use the* ***&*** *operator as a shortcut, or to combine more than two strings!*

Quest

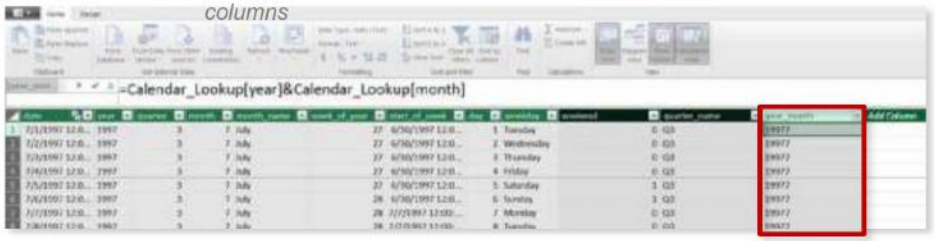
TEXT FUNCTIONS

EXAMPLES

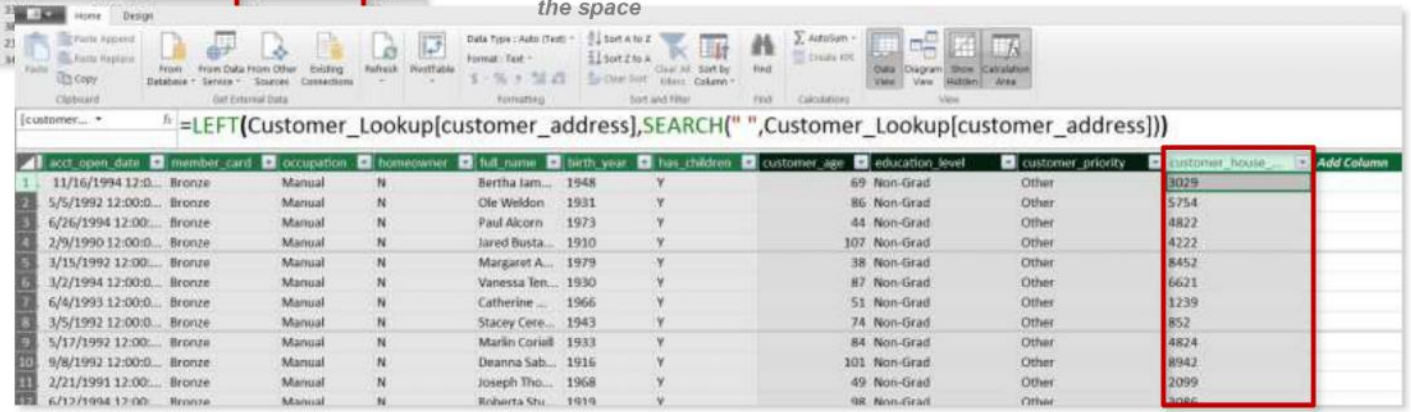
*Extract the **left 3 characters** from each value in the store_country column*



*Concatenate the values from the **year** and **month** columns*



Extract characters** from the left of the customer_address column, **up to the space



	acct_open_date	member_card	occupation	homeowner	full_name	birth_year	has_children	customer_age	education_level	customer_priority	customer_house_...	Add Column
1	11/16/1994 12:0...	Bronze	Manual	N	Bertha Iam...	1948	Y	69	Non-Grad	Other	3029	
2	5/5/1992 12:00:0...	Bronze	Manual	N	Ole Weldon	1931	Y	86	Non-Grad	Other	5754	
3	6/26/1994 12:00:...	Bronze	Manual	N	Paul Alcorn	1973	Y	44	Non-Grad	Other	4822	
4	2/9/1990 12:00:0...	Bronze	Manual	N	Jared Busta...	1910	Y	107	Non-Grad	Other	4222	
5	3/15/1992 12:00:...	Bronze	Manual	N	Margaret A...	1979	Y	38	Non-Grad	Other	8452	
6	3/2/1994 12:00:0...	Bronze	Manual	N	Vanessa Ten...	1930	Y	87	Non-Grad	Other	6621	
7	6/4/1993 12:00:0...	Bronze	Manual	N	Catherine ...	1966	Y	51	Non-Grad	Other	1239	
8	3/5/1992 12:00:0...	Bronze	Manual	N	Stacey Cere...	1943	Y	74	Non-Grad	Other	852	
9	5/17/1992 12:00:...	Bronze	Manual	N	Marlin Coriell	1933	Y	84	Non-Grad	Other	4824	
10	9/8/1992 12:00:0...	Bronze	Manual	N	Deanna Sab...	1916	Y	101	Non-Grad	Other	8942	
11	2/21/1991 12:00:...	Bronze	Manual	N	Joseph Tho...	1968	Y	49	Non-Grad	Other	2099	
12	6/12/1994 12:00:...	Bronze	Manual	N	Roberta Stu...	1919	Y	98	Non-Grad	Other	3086	

AI Quest

CALCUL

CALCULAT

Evaluates a given expression or formula under a set of defined filters

=**CALCULATE**(<expression>, <filter1>, <filter2>,…)

Name of an existing measure or a formula for a valid measure

Examples:

- *[Total Transactions]*
- *SUM(Transactions[quantity])*

*List of simple Boolean (True/False) filter expressions (**note:** these require simple, fixed values; you cannot create filters based on measures)*

Examples:

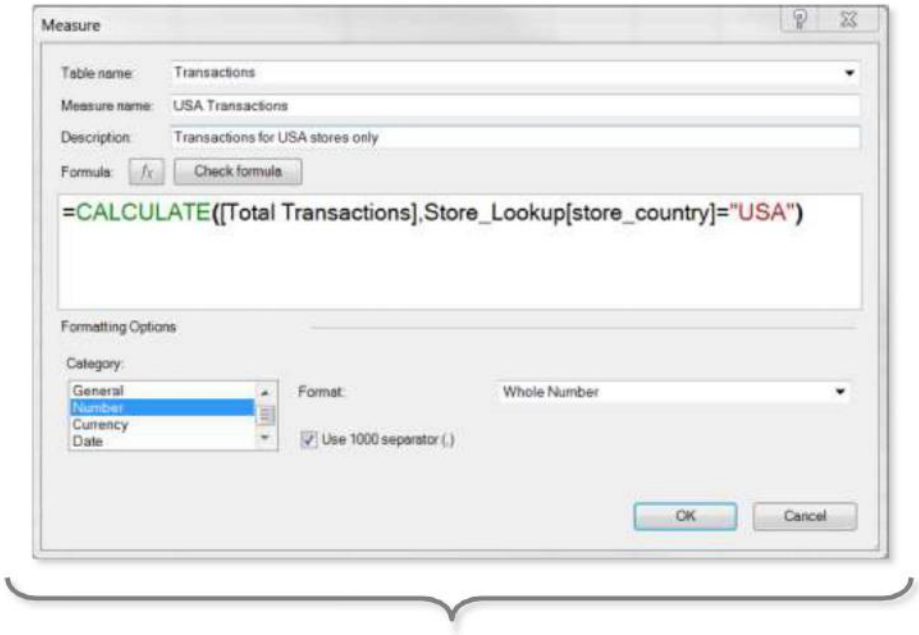
- *Store_Lookup[store_country]=**"USA"***
- *Calendar[Year]=**1998***
- *Transactions[quantity]>=**5***

PRO TIP:

*CALCULATE works just like **SUMIF** or **COUNTIF**, except it can evaluate measures based on ANY sort of calculation (not just a sum, count, etc); it may help to think of it like **"CALCULATEIF"***

AI Quest

CALCULATE



In this case we've defined a new measure named "**USA Transactions**", which evaluates the "**Total Transactions**" measure when the store country equals "**USA**"

store_country	Total Transactions	USA Transactions
CANADA	16,091	180,823
MEXICO	72,806	180,823
USA	180,823	180,823
Grand Total	**269,720**	**180,823**

*Why do we see the same repeating value when we add **store_country** to rows? Shouldn't these cells have filter contexts for Canada and Mexico?*

HEY THIS IS IMPORTANT!

The CALCULATE function **modifies filters** and **overrules** any competing ones defined by the PivotTable coordinates!

In this example, the MEXICO cell has a filter context of store_country= "**MEXICO**" (*defined by the row label*) AND story_country= "**USA**" (*defined by the CALCULATE function*)

Both cannot be true at the same time, so the MEXICO filter is overwritten and CALCULATE takes priority

AI Quest

CALCULATE *CHANGES* THE FILTER CONTEXT

CALCULATE

Modify filters if measure contains CALCULATE

If the measure being evaluated contains a **CALCULATE** function, filter context is modified between **Step 1** & **Step 2**

Store_Lookup[store_country] = "USA"

STEP 1

Detect pivot coordinates & apply filter context

store_country	Total Transactions	USA Transactions
CANADA	16,091	180,823
MEXICO	72,806	180,823
USA	180,823	180,823
Grand Total	**269,720**	**180,823**

Store_Lookup[store_country] = "MEXICO"

STEP 2

Carry the filters across all table relationships

1 1

* *

STEP 3

Evaluate the formula against the filtered table

Total Transactions where store_country "USA" =

AI Quest

FILT

FILTE

Returns a table that represents a subset of another table or expression

=FILTER(<table>, <filter expression>)

Table to be filtered

Examples:

- *Store_Lookup*
- *Product_Lookup*

A Boolean (True/False) filter expression to be evaluated for each row of the table

Examples:

- *Store_Lookup[store_country]=**"USA"***
- *Calendar[Year]=**1998***
- *[retail_price]>**AVERAGE[retail_price]***

HEY THIS IS IMPORTANT!

FILTER is used to **add filter context** on top of what's already defined by the PivotTable layout.

Since FILTER returns a table (as opposed to a scalar), it's almost always used as an *input* to other functions, **like enabling more complex filtering options within a CALCULATE function** (or passing a filtered table to an iterator like SUMX)

PRO TIP:

*Since FILTER iterates through each row in a table, it can be slow and processor-intensive; **never use FILTER when a normal CALCULATE function will accomplish the same thing!***

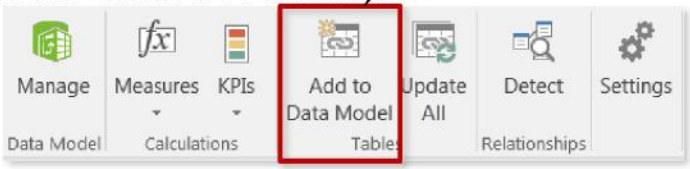
AI Quest

PRO TIP: FILTERING WITH DISCONNECTED

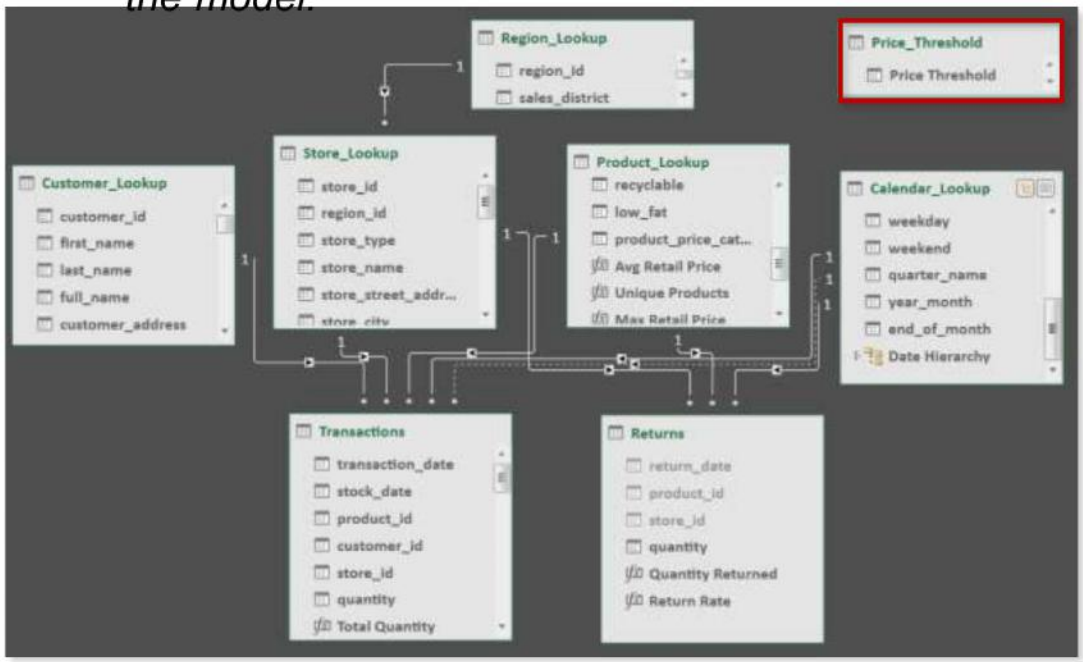
STEP 1: *Create an Excel table containing a list of values to use as thresholds or parameters:*

	A
1	**Price Threshold**
2	1
3	2
4	3
5	4

STEP 2: *Add the table to the* ***Data Model*** *(from* ***Power Pivot*** *tab):*



STEP 3: *Make sure that your table loaded, and is NOT connected to any other table in the model:*

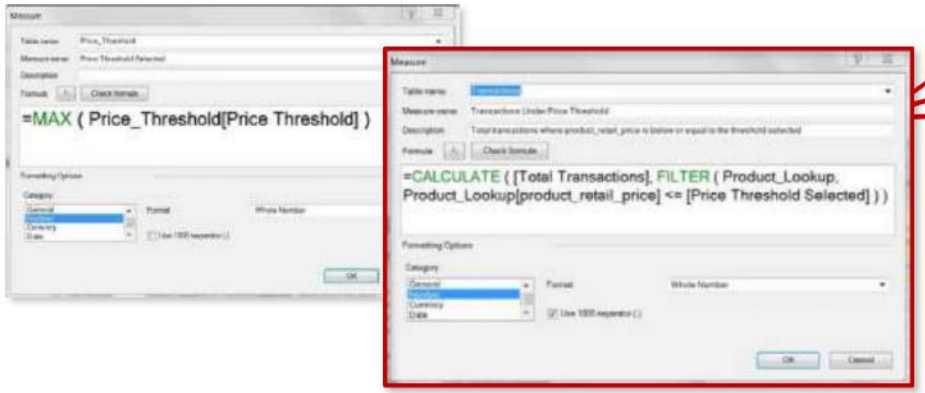


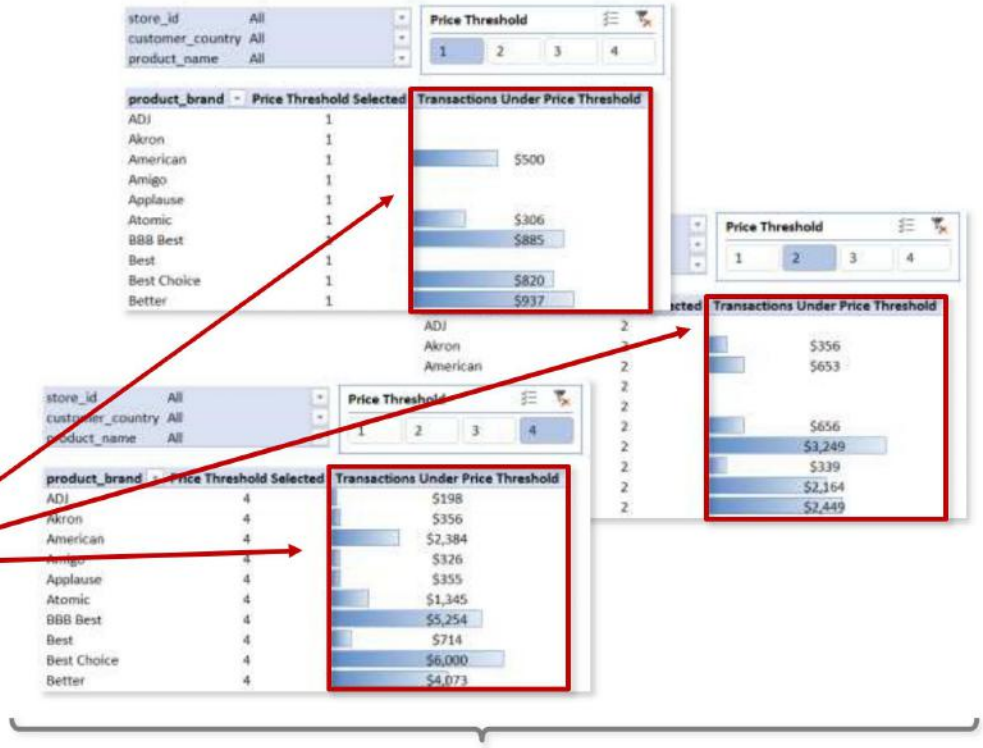
AI Quest

PRO TIP: FILTERING WITH DISCONNECTED

STEP 4: *Place your new table on the pivot as a slicer:*

STEP 5: *Create a measure to capture the slicer selection, then reference it in a* ***FILTER*** *statement within* ***CALCULATE****:*



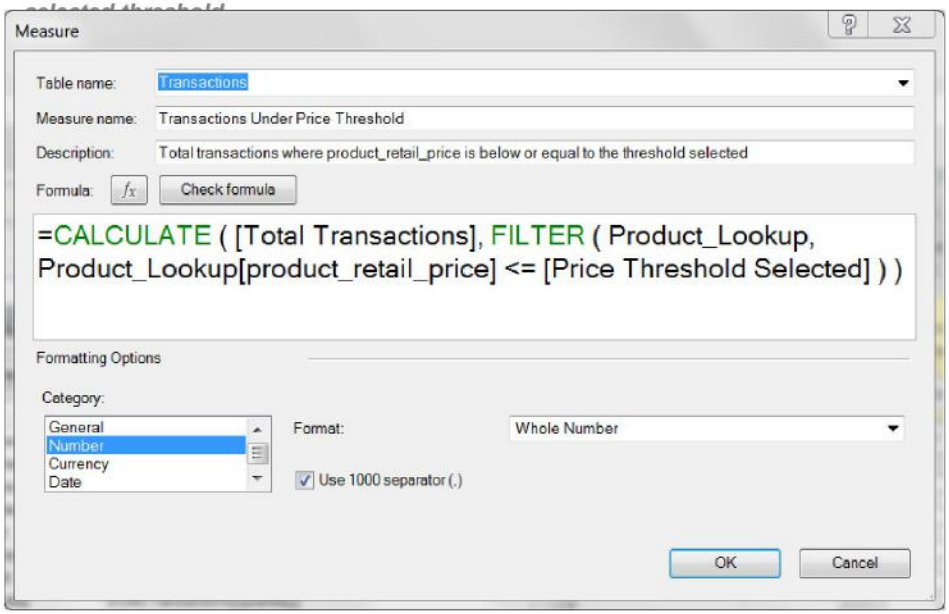


The ***Transactions Under Price Threshold*** *measure calculates Total Transactions* ***when the product price is below the selected threshold***

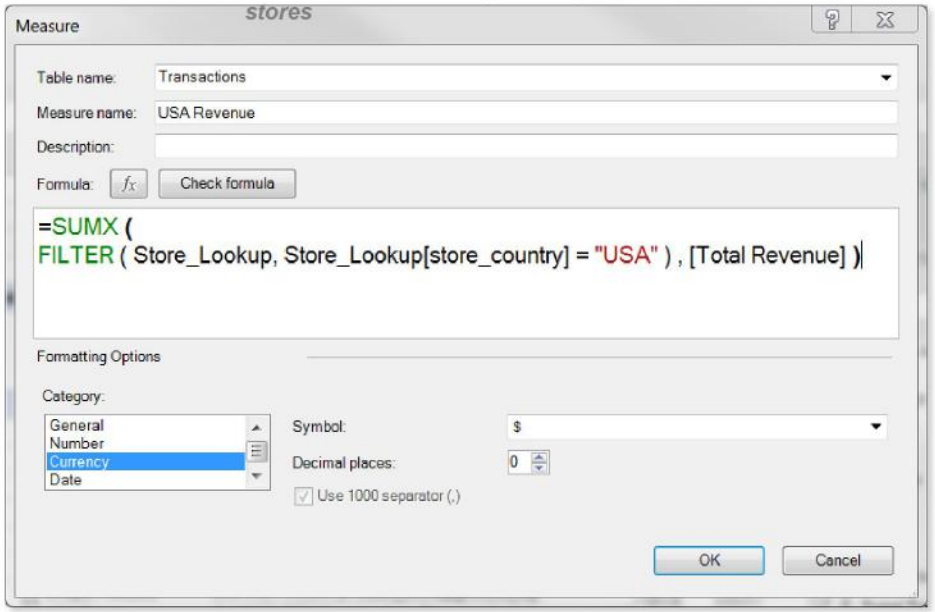
AI Quest

FILTER

Calculate Total Transactions ***only for cases where the product price is below a selected threshold***



Calculate Total Revenue, ***but only for USA stores***



AI Quest

AL

ALL

Returns all rows in a table, or all values in a column, ignoring any filters that have been applied

=ALL(<table> or <column>, [column1], [column2],…)

The table or column that you want to clear filters on

Examples:

- *Transactions*
- *Product_Lookup[product_brand]*

List of columns that you want to clear filters on (optional)

Notes:

- *If your first parameter is a table, you can't specify additional columns*
- *All columns must include the table name, and come from the same table*

Examples:

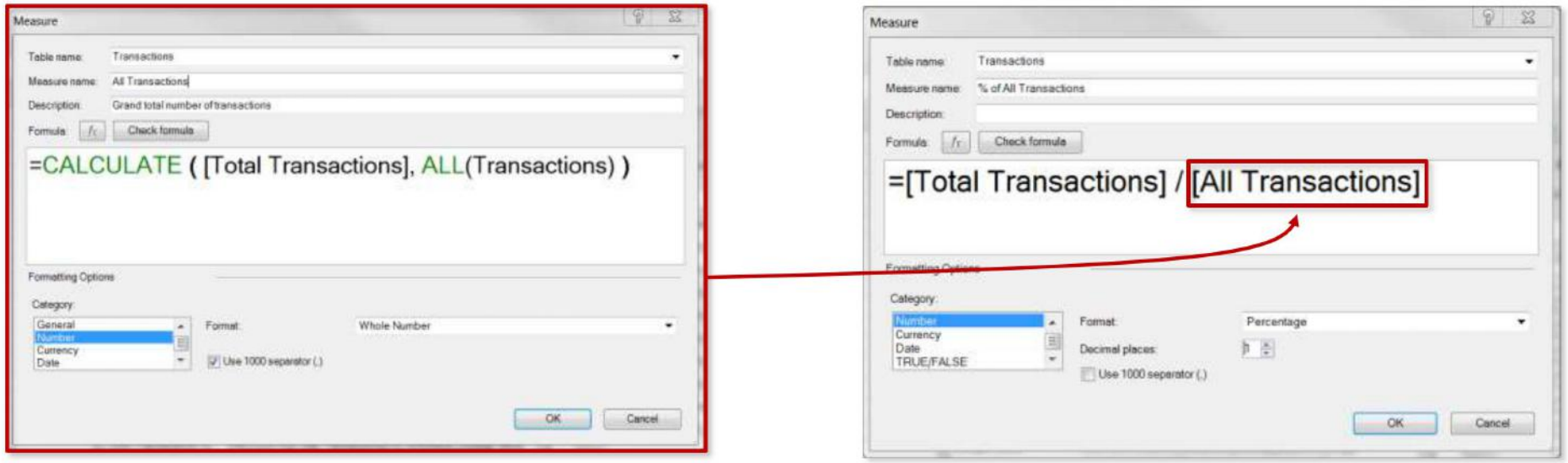
- *Customer_Lookup[customer_city], Customer_Lookup[customer_country]*
- *Product_Lookup[product_name]*

PRO TIP:

*ALL is like the opposite of FILTER; instead of adding filter context, ALL **removes filter context**. This is often used when you need unfiltered values that won't be skewed by the PivotTable layout (i.e. Category sales as % of Total)*

AI Quest

ALL



- In this example, we use **ALL** to calculate total transactions across *all rows* in the Transactions table, **ignoring any filter context from the PivotTable**
  - By dividing the original **[Total Transaction]** measure (which responds to PivotTable filter context as expected) by the new **[All Transactions]** measure, we can correctly calculate the percentage of the total no matter how the PivotTable is filtered

AI Quest

RELAT
ED

RELATE

Returns related values in each row of a table using relationships with other tables

=RELATED(<column>**)**

The column that contains the values you want to retrieve

Examples:

- *Product_Lookup[product_brand]*
- *Store_Lookup[store_country]*

HEY THIS IS IMPORTANT!

RELATED works almost *exactly* like a **VLOOKUP** function – it uses the relationship between tables (*defined by primary and foreign keys*) to pull values from one table into a new column of another.

Since this function requires row context, it can only be used as a **calculated column** or as part of an **iterator function** that cycles through all rows in a table (FILTER, SUMX, MAXX, etc.)

PRO TIP:

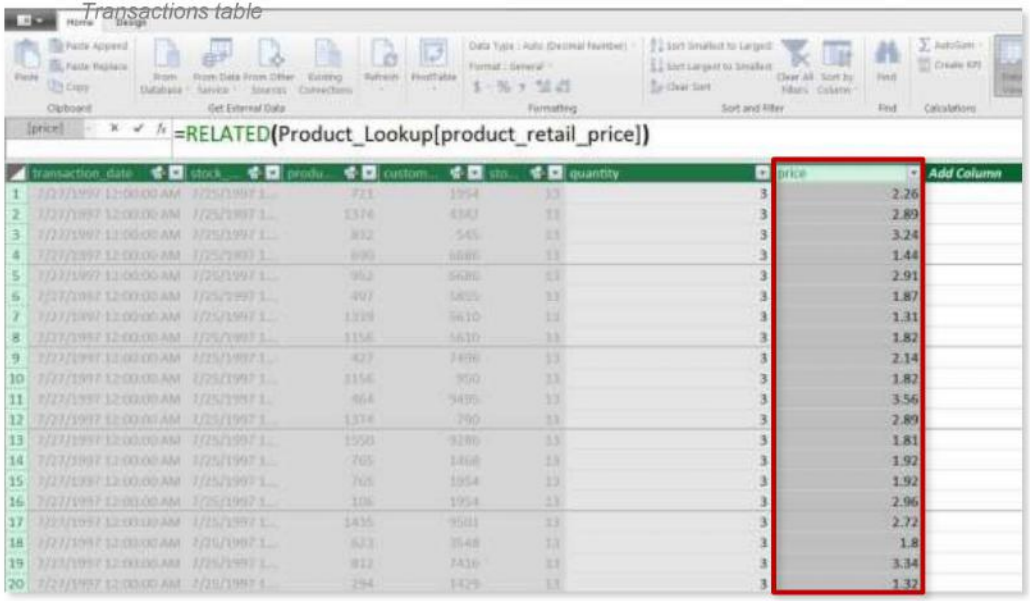
Avoid using RELATED to create redundant calculated columns unless you absolutely need them, since those extra columns increase file size; instead, use RELATED within a measure like FILTER or SUMX

AI Quest

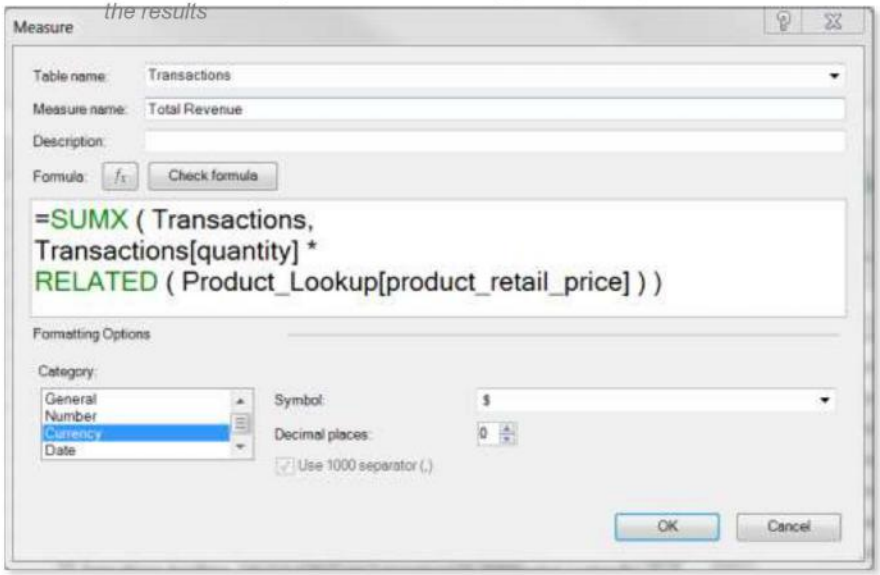
RELATED

EXAMPLES

*Retrieve the **retail price** from the Product_Lookup table and append it to the Transactions table*



*Multiply the **quantity** in each row of the Transactions table with the related **retail price** from the Product_Lookup table, and sum the results*



AI Quest

ITERATOR (“X”) FUNCTIONS

Iterator (or “**X**”) **functions** allow you to loop through the same calculation or expression on
each row of a table, and then apply some sort of aggregation to the results (SUM, MAX, etc.)

=**SUMX**(<table>, <expression>)

*Aggregation to apply to calculated rows**

Examples:
- *SUMX*
- *COUNTX*
- *AVERAGEX*
- *RANKX*
- *MAXX/MINX*

Table in which the expression will be evaluated

Examples:
- *Transactions*
- *FILTER(Transactions, RELATED(Store_Lookup[country])=“USA”)*

Expression to be evaluated for each row of the given table

Examples:
- *[Total Transactions]*
- *Transactions[price] * Transactions[quantity]*

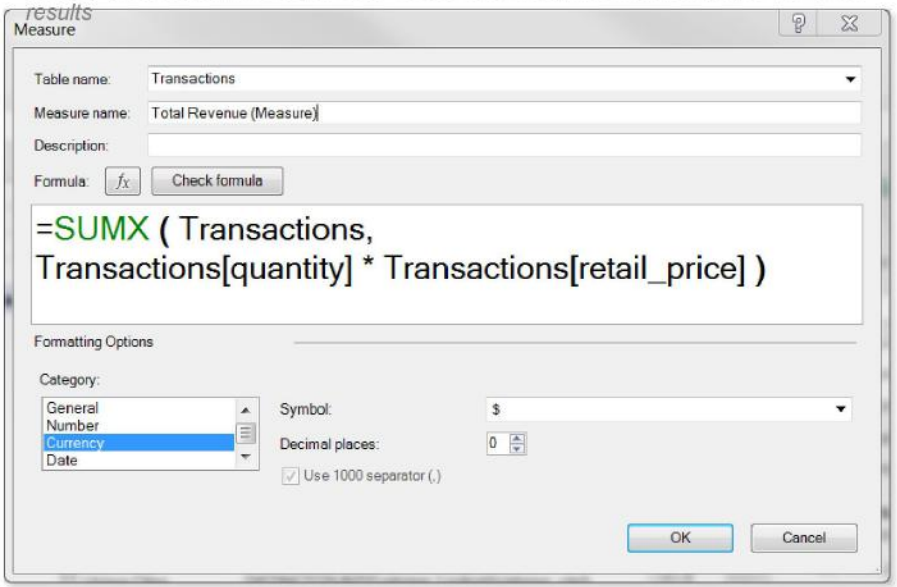
PRO TIP:
*Imagine the function **adding a temporary new column** to the table, calculating the value in each row*
(based on the expression) and then applying the aggregation to that new column (like SUMPRODUCT)

In this example we’re looking at **SUMX, but all “X” functions follow a similar syntax*

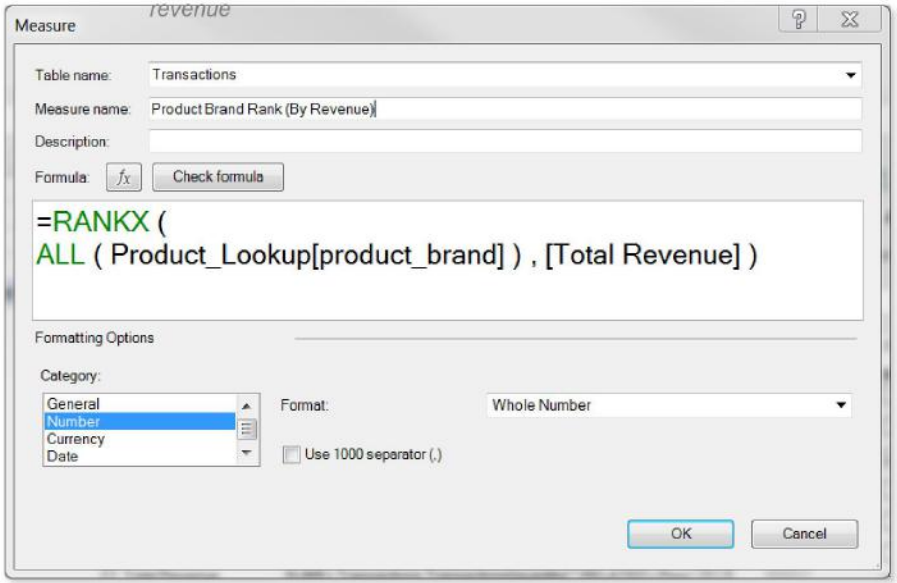
AI Quest

ITERATOR ("X") FUNCTIONS

*Multiply **quantity** and **retail price** for each row in the Transactions table, and sum the results*



*Calculate the **rank of each product brand**, based on total revenue*



AI Quest

BASIC DATE & TIME FUNCTIONS

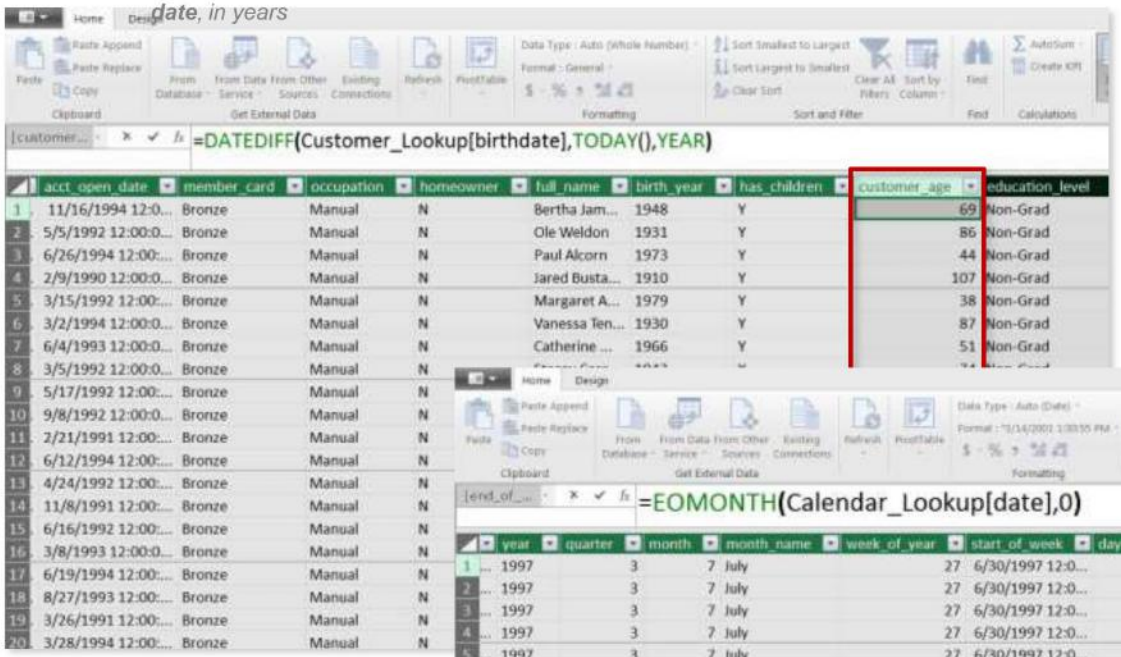
DAY/MONTH / YEAR()	*Returns the day of the month (1-31), month of the year (1-12), or year of a given date*	**=DAY/MONTH/YEAR**(<date>)
HOUR/MINUTE/ SECOND()	*Returns the hour (0-23), minute (0-59), or second (0-59) of a given datetime value*	**=HOUR/MINUTE/SECOND**(<datetime>)
TODAY/NOW()	*Returns the current date or exact time*	**=TODAY/NOW**()
WEEKDAY / WEEKNUM()	*Returns a weekday number from 1 (Sunday) to 7 (Sunday), or the week # of the year*	**=WEEKDAY/WEEKNUM**(<date>, *<type>*)
EOMONTH()	*Returns the date of the last day of the month, +/- a specified number of months*	**=EOMONTH**(<start_date>, <months>)
DATEDIFF()	*Returns the difference between two dates, based on a selected interval*	**=DATEDIFF**(<start_date>, <end_date>, <interval>)

AI Quest

BASIC DATE & TIME FUNCTIONS

EXAMPLES

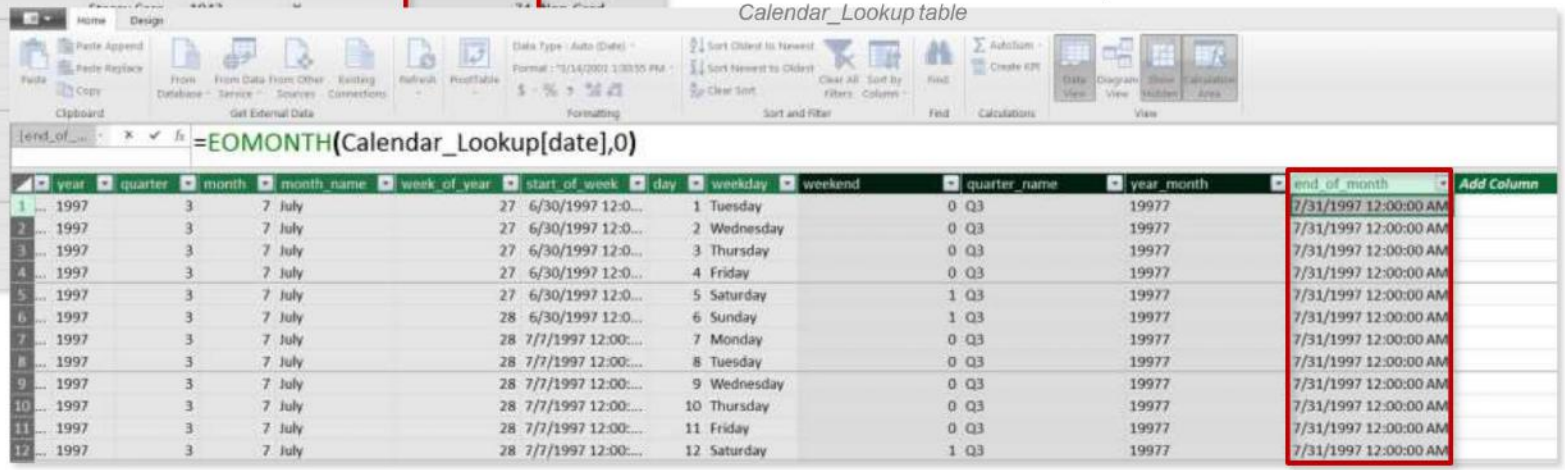
*Calculate the **time difference between the customer birthdate and current date**, in years*



=DATEDIFF(Customer_Lookup[birthdate],TODAY(),YEAR)

	acct_open_date	member_card	occupation	homeowner	full_name	birth_year	has_children	customer_age	education_level
1	11/16/1994 12:0...	Bronze	Manual	N	Bertha Jam...	1948	Y	69	Non-Grad
2	5/5/1992 12:00:0...	Bronze	Manual	N	Ole Weldon	1931	Y	86	Non-Grad
3	6/26/1994 12:00:...	Bronze	Manual	N	Paul Alcorn	1973	Y	44	Non-Grad
4	2/9/1990 12:00:0...	Bronze	Manual	N	Jared Busta...	1910	Y	107	Non-Grad
5	3/15/1992 12:00:...	Bronze	Manual	N	Margaret A...	1979	Y	38	Non-Grad
6	3/2/1994 12:00:0...	Bronze	Manual	N	Vanessa Ten...	1930	Y	87	Non-Grad
7	6/4/1993 12:00:0...	Bronze	Manual	N	Catherine ...	1966	Y	51	Non-Grad
8	3/5/1992 12:00:0...	Bronze	Manual	N					
9	5/17/1992 12:00:...	Bronze	Manual	N					
10	9/8/1992 12:00:0...	Bronze	Manual	N					
11	2/21/1991 12:00:...	Bronze	Manual	N					
12	6/12/1994 12:00:...	Bronze	Manual	N					
13	4/24/1992 12:00:...	Bronze	Manual	N					
14	11/8/1991 12:00:...	Bronze	Manual	N					
15	6/16/1992 12:00:...	Bronze	Manual	N					
16	3/8/1993 12:00:0...	Bronze	Manual	N					
17	6/19/1994 12:00:...	Bronze	Manual	N					
18	8/27/1993 12:00:...	Bronze	Manual	N					
19	3/26/1991 12:00:...	Bronze	Manual	N					
20	3/28/1994 12:00:...	Bronze	Manual	N					

*Calculate the **end date of the month**, for each row in the Calendar_Lookup table*



=EOMONTH(Calendar_Lookup[date],0)

	year	quarter	month	month_name	week_of_year	start_of_week	day	weekday	weekend	quarter_name	year_month	end_of_month	Add Column
1	1997	3	7	July	27	6/30/1997 12:0...	1	Tuesday	0	Q3	19977	7/31/1997 12:00:00 AM	
2	1997	3	7	July	27	6/30/1997 12:0...	2	Wednesday	0	Q3	19977	7/31/1997 12:00:00 AM	
3	1997	3	7	July	27	6/30/1997 12:0...	3	Thursday	0	Q3	19977	7/31/1997 12:00:00 AM	
4	1997	3	7	July	27	6/30/1997 12:0...	4	Friday	0	Q3	19977	7/31/1997 12:00:00 AM	
5	1997	3	7	July	27	6/30/1997 12:0...	5	Saturday	1	Q3	19977	7/31/1997 12:00:00 AM	
6	1997	3	7	July	28	6/30/1997 12:0...	6	Sunday	1	Q3	19977	7/31/1997 12:00:00 AM	
7	1997	3	7	July	28	7/7/1997 12:00:...	7	Monday	0	Q3	19977	7/31/1997 12:00:00 AM	
8	1997	3	7	July	28	7/7/1997 12:00:...	8	Tuesday	0	Q3	19977	7/31/1997 12:00:00 AM	
9	1997	3	7	July	28	7/7/1997 12:00:...	9	Wednesday	0	Q3	19977	7/31/1997 12:00:00 AM	
10	1997	3	7	July	28	7/7/1997 12:00:...	10	Thursday	0	Q3	19977	7/31/1997 12:00:00 AM	
11	1997	3	7	July	28	7/7/1997 12:00:...	11	Friday	0	Q3	19977	7/31/1997 12:00:00 AM	
12	1997	3	7	July	28	7/7/1997 12:00:...	12	Saturday	1	Q3	19977	7/31/1997 12:00:00 AM	

AI Quest

TIME INTELLIGENCE FORMULAS

Time Intelligence functions allow you to easily calculate common time comparisons:

Performance To-Date

```
=CALCULATE(<measure>, DATESYTD(Calendar[Date]))
```

Use **DATESQTD** *for Quarters or* **DATESMTD** *for Months*

Previous

```
=CALCULATE(<measure>, DATEADD(Calendar[Date],-1,MONTH))
```

*Select an interval (***DAY***,* **MONTH***,* **QUARTER***, or* **YEAR***) and the # of intervals to compare (i.e. previous month, rolling 10-day)*

Running

```
=CALCULATE(<measure>,
        DATESINPERIOD(Calendar[Date], MAX(Calendar[Date]),-10,DAY))
```

PRO TIP:

To calculate a ***moving average****, use the running total calculation above and divide by the # of intervals!*

AI Quest

SPEED & PERFORMANCE CONSIDERATIONS

Avoid using unnecessary slicers, or consider disabling cross-filtering

- *When you use multiple slicers, they "cross-filter" by default; in other words, options in **Slicer B** are automatically grayed out if they aren't relevant given a selected value in **Slicer A***
- *To disable, select **Slicer Tools** > **Slicer Settings** and uncheck "**Visually indicate items with no data**"*

Eliminate redundant columns; keep data tables narrow

- *Data tables should ideally only contain quantitative values and foreign keys; any extra descriptive columns should live in a related lookup table*

Imported columns are better than calculated columns

- *When possible, create calculated columns at the source (i.e. in your raw database) or using Power Query; this is more efficient than processing those calculations in the Data Model/Power Pivot*

Minimize iterator functions (FILTER, SUMX, etc.)

- *Functions that cycle through each row in a table are "expensive", meaning that*

DAX BEST PRACTICES

- **Write measures for even the simplest calculations** *(i.e. Sum of Sales)*
  - *Once you create a measure it can be used anywhere in the workbook and as an input to other, more complex calculations*
- **Break measures down into simple, component parts**
  - *DAX is a difficult language to master; focus on practicing and understanding simple components at first, then assemble them into more advanced formulas*
- **Reference columns with the table name, and measures alone**
  - *Using "fully qualified" column references (preceeded by the table name) make formulas more readable and intuitive, and differentiates them from*

AI Quest

WRAPPING

DATA VISUALIZATION OPTIONS

There are several options for building **visuals** and **reports** from a data model:

*Available **within Excel***

1. **PivotCharts & Conditional Formatting**
   - *Check out my **Data Analysis with Excel PivotTables** course for a deep dive*
2. **Spreadsheet-based dashboards built with CUBE functions**
   - *Use CUBE functions to pull values from the data model for custom Excel reports (no pivots)*
3. **Power View, Power Map, etc.**
   - *Excel plug-in with Power Pivot and other BI tools; recommend PowerBI as a better option*

Standalone product (desktop + online)

4. **Microsoft PowerBI**
   - *Brand new (**free!**) self-service BI product for **loading, shaping, modeling, and visualizing data***

AI Quest

SNEAK PEEK:

Power BI

PowerBI is a standalone Microsoft business intelligence product, which includes both desktop and web-based applications for loading, modeling, and visualizing data

For info about plans & pricing:
powerbi.microsoft.com

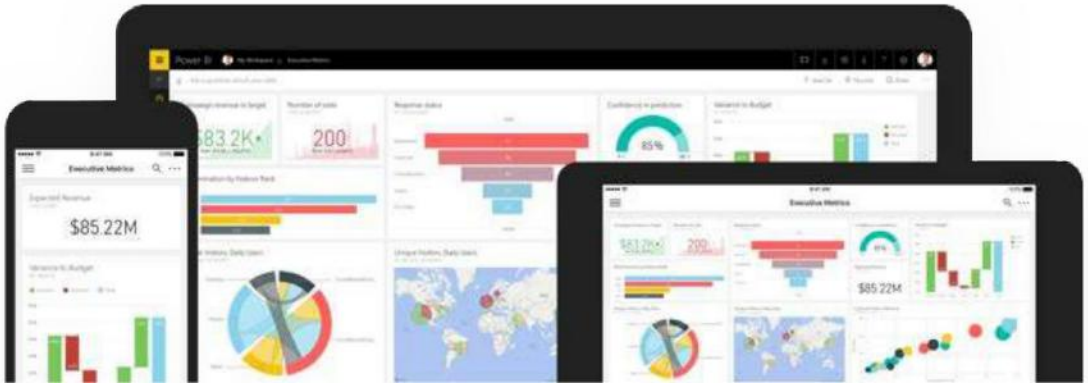
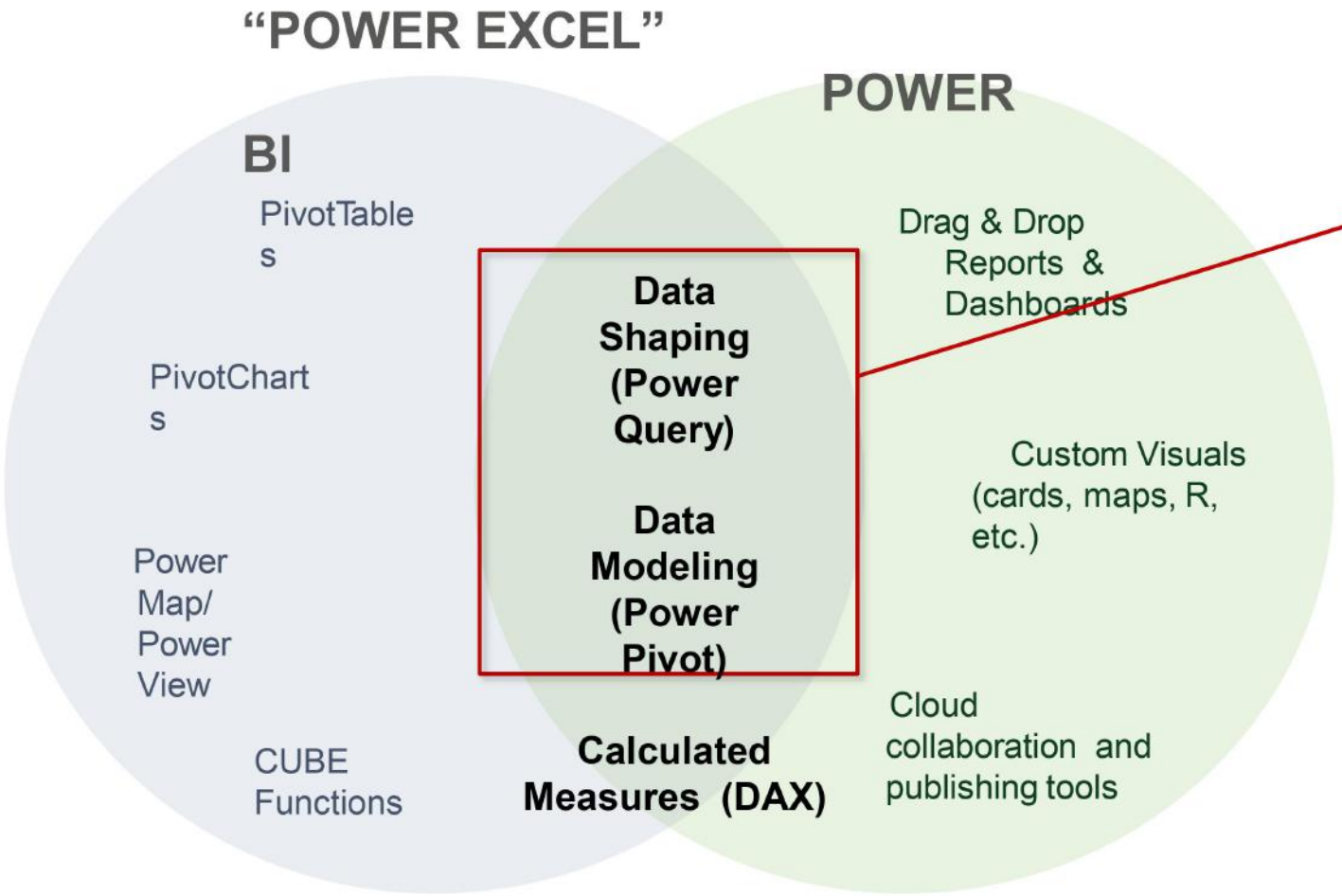


Figure 1. Magic Quadrant for Business Intelligence and Analytics Platforms

Source: Gartner (February 2017)

AI Quest

“POWER EXCEL” VS. POWER BI



“Power Excel” and PowerBI are built on top of the ***exact same engine*!**
PowerBI takes the same data shaping, modeling and analytics capabilities and adds new **reporting and visualization tools**; you can import an entire data model directly from Excel!

AI Quest


Jul
Aug
Sep
Oct
Nov
8/9
8/16
8/23
8/30
8/30
Chris Burd
Courtney Lowe
Phil Escott
70%
100%
THANK
YOU!