

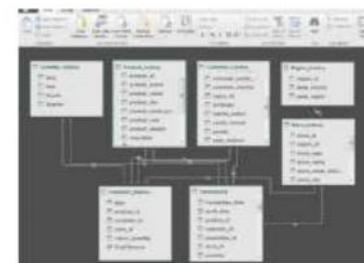
# INTRO TO “POWER

“~~AI~~”



# 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



# “THE BEST THING TO HAPPEN TO EXCEL IN

DATA MANAGEMENT

- 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

\*Quote by Bill Jelen (aka "Mr.

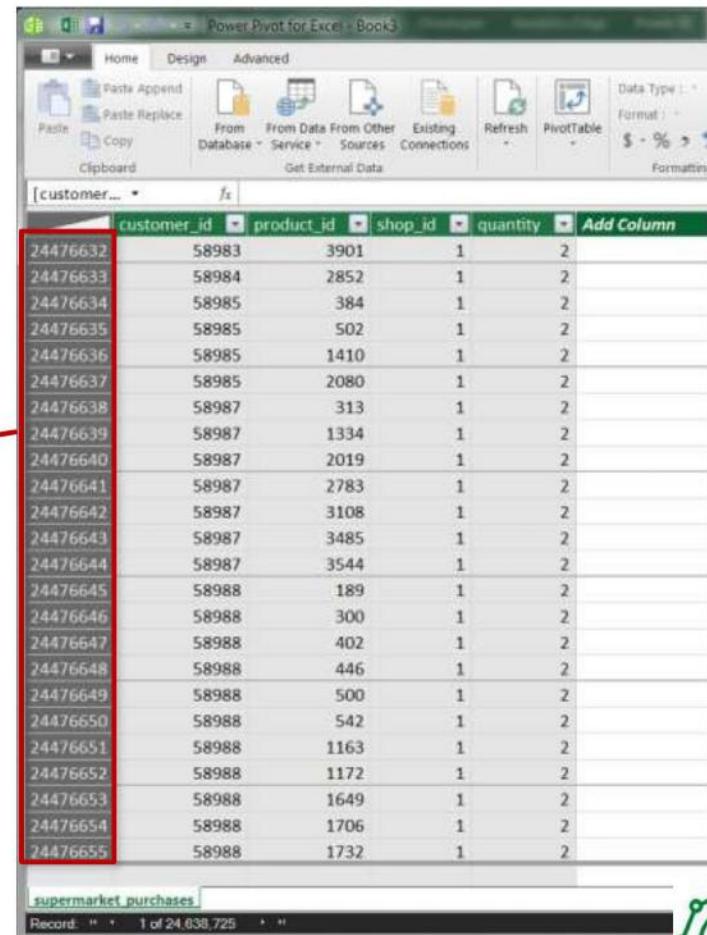


# #1: IMPORT & ANALYZE MILLIONS OF ROWS



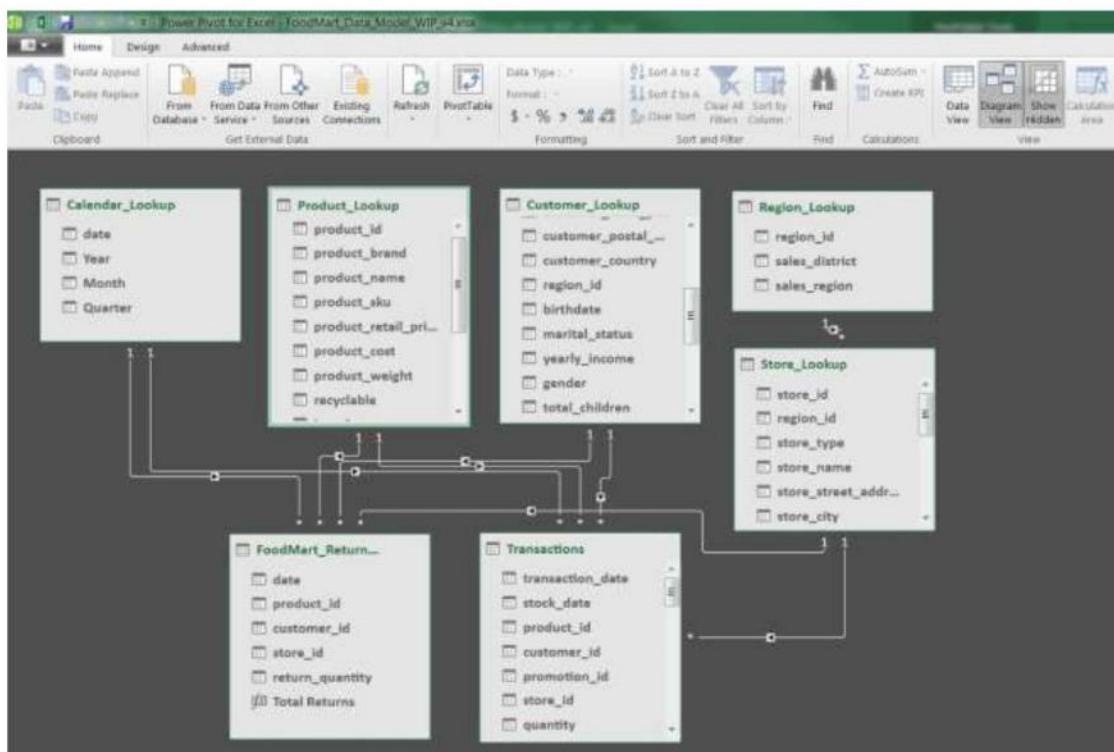
When was the last time you loaded **25,000,000** rows of data into Excel?

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!*)



customer_id	product_id	shop_id	quantity	Add Column
24476632	58983	3901	1	2
24476633	58984	2852	1	2
24476634	58985	384	1	2
24476635	58985	502	1	2
24476636	58985	1410	1	2
24476637	58985	2080	1	2
24476638	58987	313	1	2
24476639	58987	1334	1	2
24476640	58987	2019	1	2
24476641	58987	2783	1	2
24476642	58987	3108	1	2
24476643	58987	3485	1	2
24476644	58987	3544	1	2
24476645	58988	189	1	2
24476646	58988	300	1	2
24476647	58988	402	1	2
24476648	58988	446	1	2
24476649	58988	500	1	2
24476650	58988	542	1	2
24476651	58988	1163	1	2
24476652	58988	1172	1	2
24476653	58988	1649	1	2
24476654	58988	1706	1	2
24476655	58988	1732	1	2

## #2: BUILD DATA MODELS TO BLEND



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

# #3: AUTOMATE YOUR DATA

Supermarket\_Purchase\_Data - Query Editor

Home Transform Add Column View

Close & Load Refresh Properties Advanced Editor Manage Columns Reduce Rows Sort Split Column Group By Replace Values Combine Manage Parameters Data source settings New Source Recent Sources New Query

Queries

= Table.SelectRows(#"Renamed Columns", each not List.IsEmpty

	customer_id	product_id	quantity	Category
1	2	1	2	Low Value Product
2	2	2	5	Low Value Product
3	2	3	3	Low Value Product
4	2	112	4	Low Value Product
5	2	112	14	Low Value Product
6	2	112	5	Low Value Product
7	2	112	35	Low Value Product
8	2	113	3	Low Value Product
9	2	115	1	Low Value Product
10	2	115	5	Low Value Product
11	2	115	2	Low Value Product
12	2	115	26	Low Value Product
13	2	119	2	Low Value Product
14	2	124	1	Low Value Product
15	2	124	4	Low Value Product
16	2	124	5	Low Value Product

4 COLUMNS, 999+ ROWS

PREVIEW DOWNLOADED AT 5:59 PM

Query Settings

PROPERTIES

Name: Supermarket\_Purchase\_Data

All Properties

APPLIED STEPS

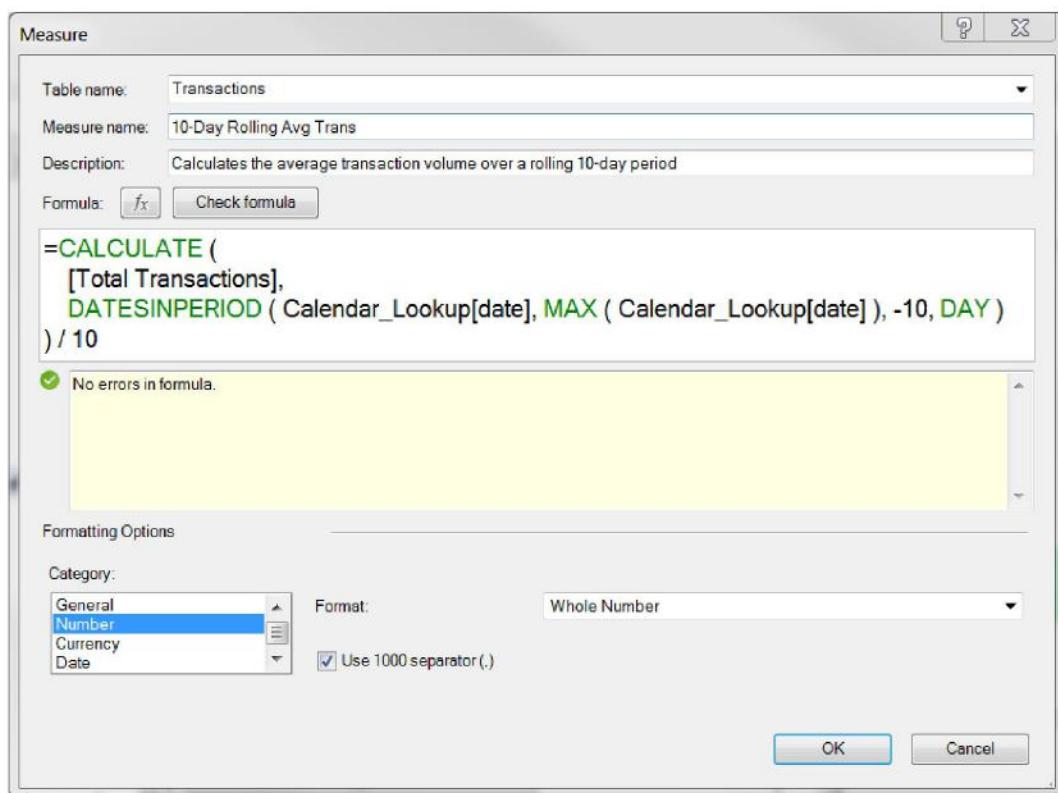
- Source
- Applied Headers
- Changed Column Type
- Removed Columns
- Filtered Rows
- Added Conditional Column
- Renamed Columns
- Removed Blank Rows

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!



## #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**



## WHEN TO USE POWER QUERY &

---

*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**

# POWER

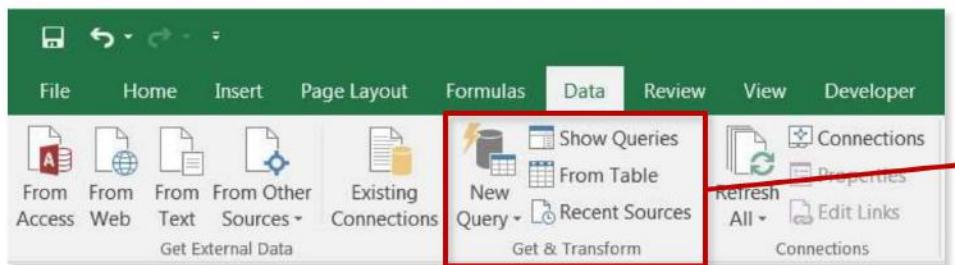
---



# MEET POWER

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

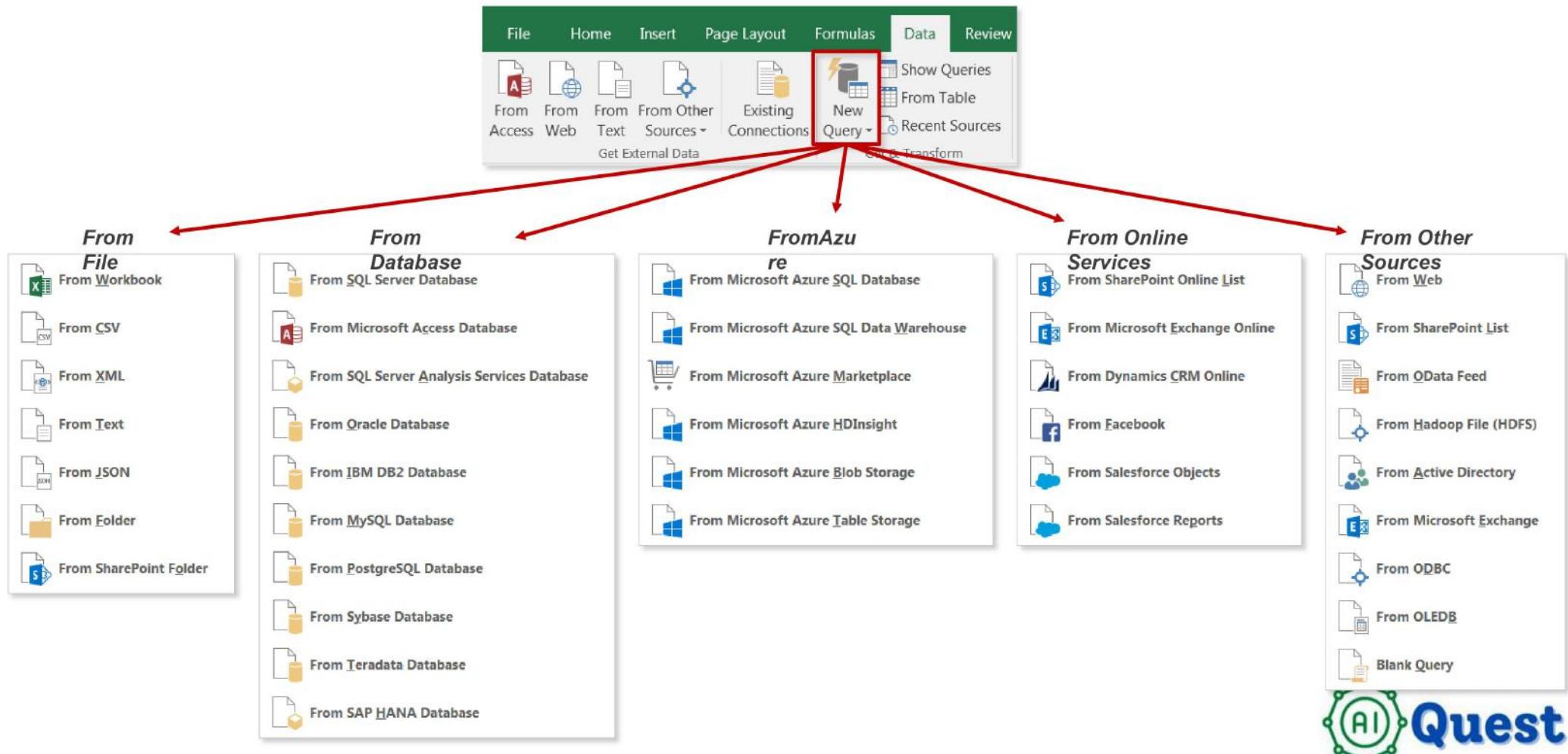
- Connect to data across a wide range of sources
- Filter, shape, append and transform raw data for further analysis and modeling
- Create stored procedures to automate your data prep (*like a macro!*)



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



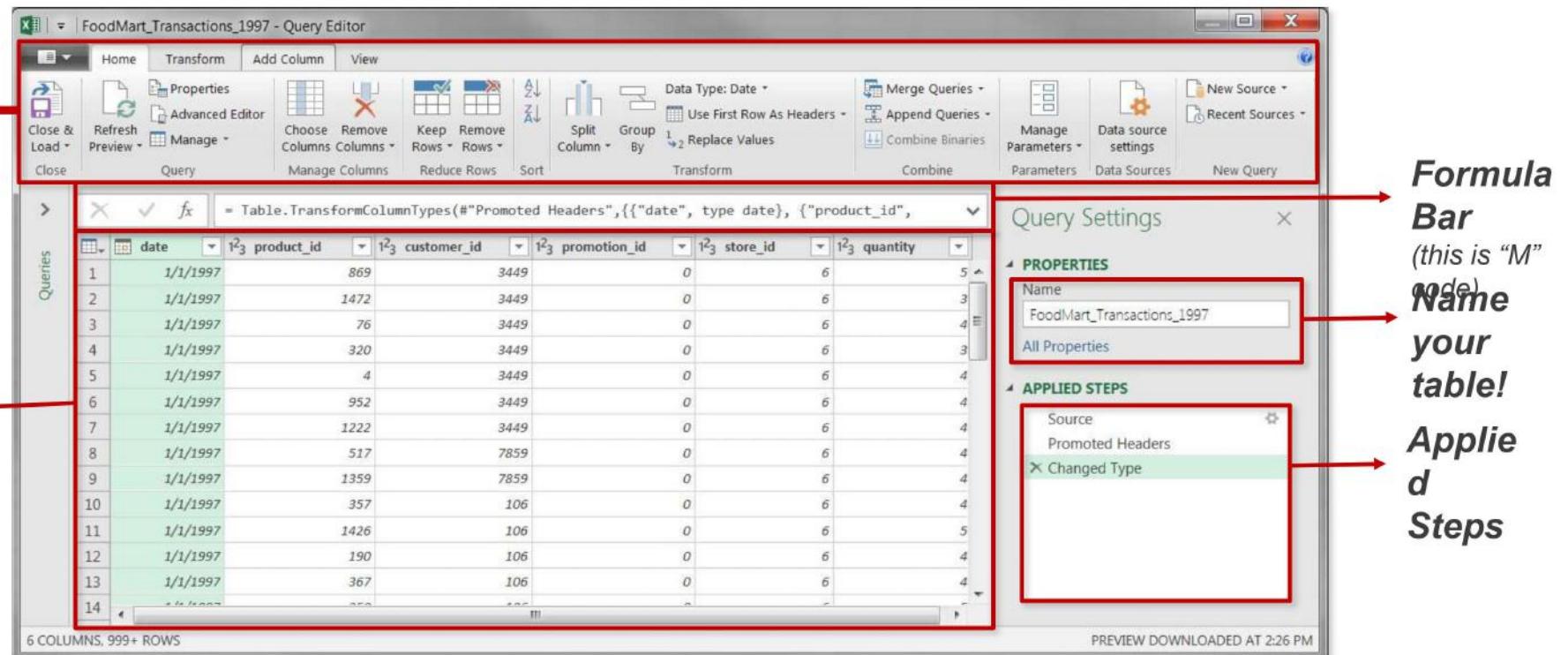
# TYPES OF DATA



# THE QUERY

Query  
Editin  
g Tools

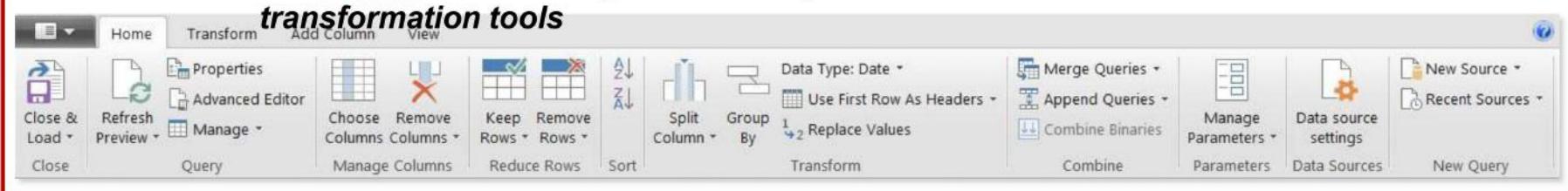
Data  
Previe  
w



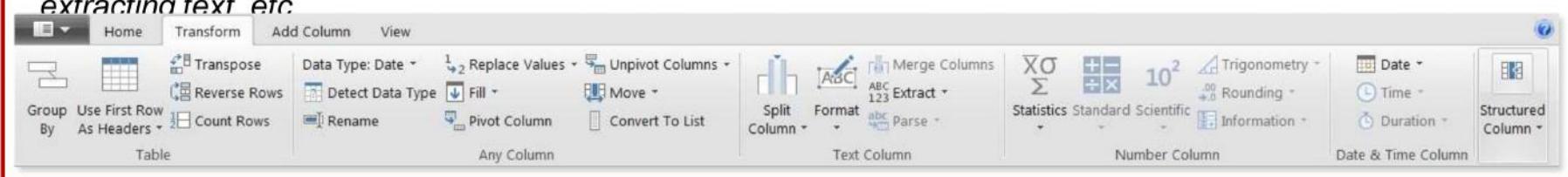
Access the **Query Editor** by creating a new query and choosing the “*Edit*” option, or by launching the Workbook Queries pane (**Data > Show Queries**) and right-clicking an existing query to edit

# QUERY EDITOR

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



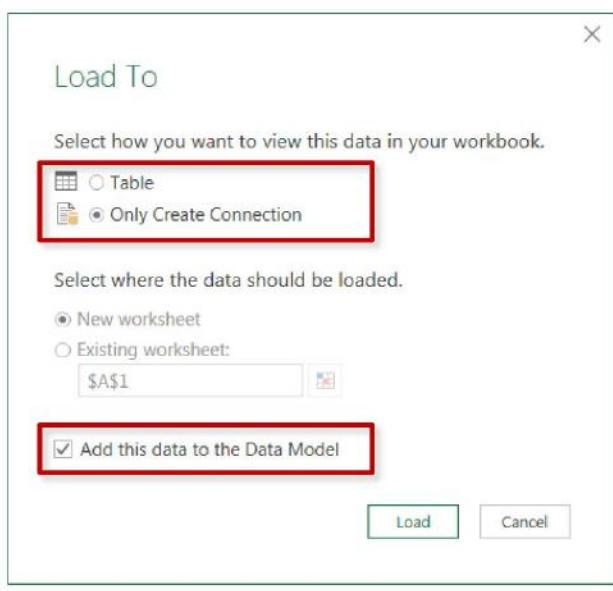
The **TRANSFORM** tab includes tools to **modify existing columns** (splitting/grouping, transposing, extracting text, etc.)



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



# DATA LOADING



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*

# BASIC TABLE

The screenshot shows the Power BI desktop interface with a green header bar. Below it is a ribbon with Home, Transform, Add Column, and View tabs. The Transform tab is selected, revealing a toolbar with icons for Close & Load, Refresh Preview, Advanced Editor, and Query. The main area displays a table with columns labeled 'transaction\_date' and 'date'. A context menu is open over the first column, listing options like Copy, Remove, Remove Other Columns, Duplicate Column, Remove Duplicates, Remove Errors, Change Type, Transform, Replace Values..., Replace Errors..., Group By..., Fill, Unpivot Columns, Unpivot Other Columns, Rename..., Move, Drill Down, and Add as New Query. The menu is highlighted with a red border.

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

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

**Promote header row**

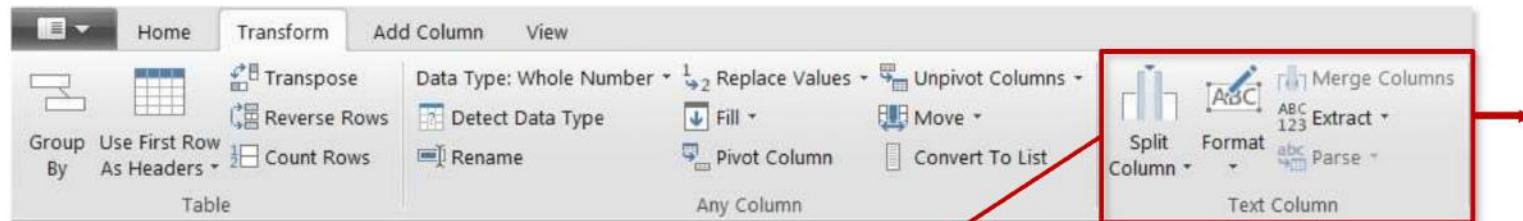
**Keep or remove columns**  
*Tip: use the "Remove Other Columns" option if you always want a specific set*

**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*

transaction_date	date
1	1/1/1997
2	1/1/1997
3	1/1/1997
4	1/1/1997
5	1/1/1997
6	1/1/1997
7	1/1/1997
8	1/1/1997
9	1/1/1997
10	1/1/1997
11	1/1/1997
12	1/1/1997
13	1/1/1997
14	1/1/1997
15	1/1/1997
16	1/1/1997
17	1/1/1997
18	1/1/1997
19	1/1/1997
20	1/1/1997
21	1/1/1997
22	1/1/1997
23	1/1/1997
	12/29/1996
	12/27/1996
	12/31/1996
	12/26/1996

# TEXT-SPECIFIC



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

By Delimiter  
By Number of Characters

lowercase  
UPPERCASE  
Capitalize Each Word  
Trim  
Clean  
Add Prefix  
Add Suffix

Length  
First Characters  
Last Characters  
Range

**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**

**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

# NUMBER-SPECIFIC

The screenshot shows the Power BI ribbon with the 'Transform' tab selected. A red box highlights the 'Number Column' section under the 'Text Column' dropdown. Below the ribbon, three callout boxes point to specific sections: one to 'Statistics' functions, one to 'Standard, Scientific, and Trigonometry' operations, and one to 'Information' tools.

- Statistics:** Sum, Minimum, Maximum, Median, Average, Standard Deviation, Count Values, Count Distinct Values.
- Operations:** Add, Multiply, Subtract, Divide, Integer-Divide, Modulo, Percentage, Percent Of.
- Advanced:** Absolute Value, Power, Square Root, Exponent, Logarithm, Factorial.
- Trigonometry:** Sine, Cosine, Tangent, Arcsine, Arccosine, Arctangent.
- Information:** Is Even, Is Odd, Sign.

**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 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

# 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*

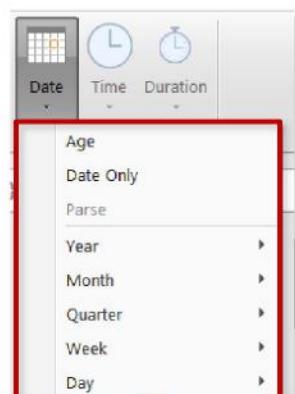


## PRO TIP:

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

# CREATING A BASIC

	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

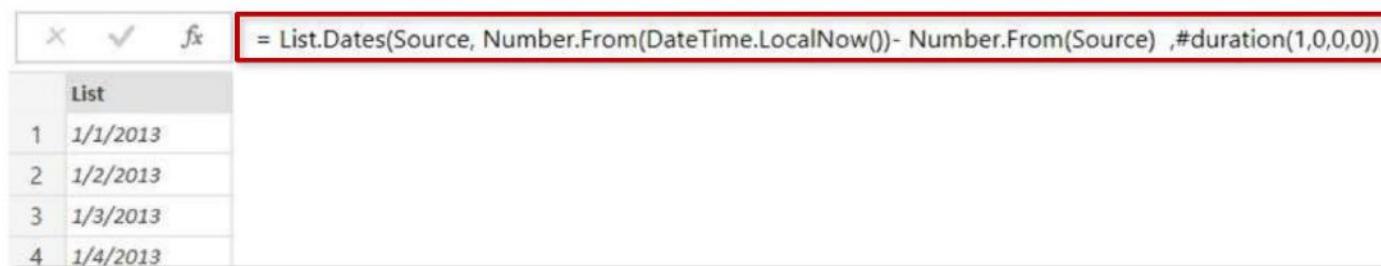
	date	Year	Month	Quarter	WeekOfYear	Day Name
1	1/1/1997	1997	1	1	1	Wednesday
2	1/2/1997	1997	1	1	1	Thursday
3	1/3/1997	1997	1	1	1	Friday
4	1/4/1997	1997	1	1	1	Saturday
5	1/5/1997	1997	1	1	1	Sunday
6	1/6/1997	1997	1	1	2	Monday
7	1/7/1997	1997	1	1	2	Tuesday
8	1/8/1997	1997	1	1	2	Wednesday
9	1/9/1997	1997	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

# PRO TIP: CREATING A ROLLING

- 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):

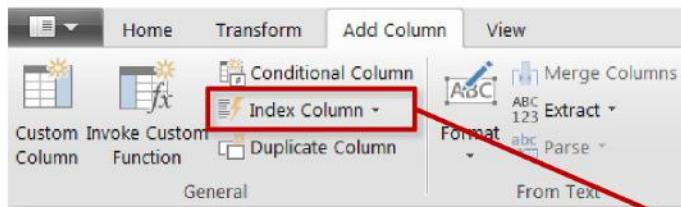


- 3) Click the **fx** icon to add a new custom step, and enter the following formula exactly as shown:



- 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

# ADDING AN INDEX

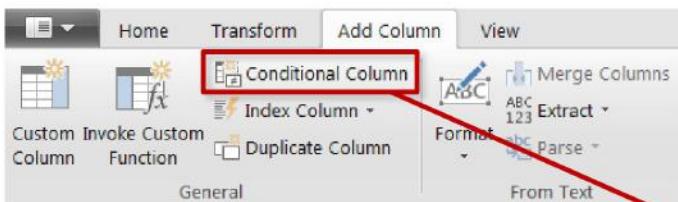


**Index Columns** contain a list of sequential values that can be used to identify each unique row in a table (*typically starting from 0 or 1*)

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

# 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**"

Add Conditional Column

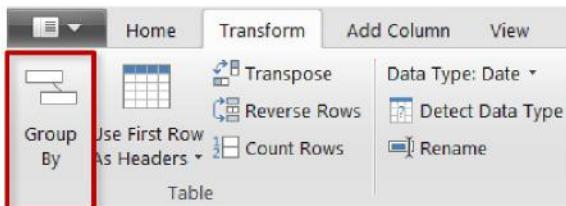
Add a conditional column that is computed from the other columns or values.

New column name  
Order Size

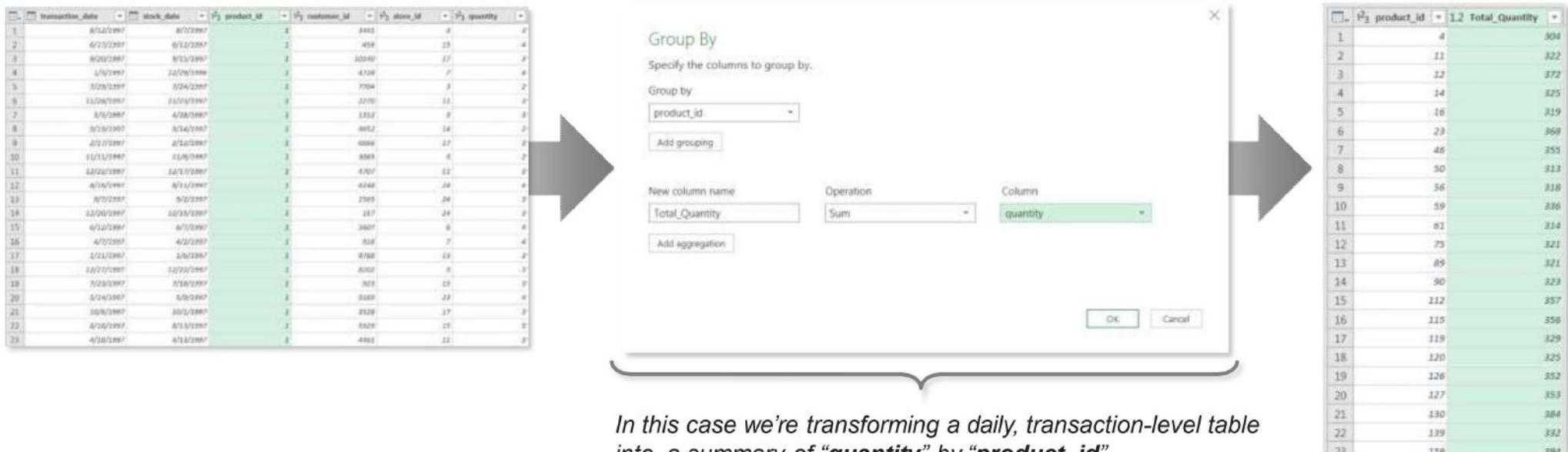
Column Name	Operator	Value ⓘ	Output ⓘ
If quantity	is greater than	ABC 123 5	Then ABC 123 Large
Else If quantity	is greater than or...	ABC 123 2	Then ABC 123 Medium
Else If quantity	equals	ABC 123 1	Then ABC 123 Small
Add Rule			
Otherwise ⓘ ABC 123 Other			

OK Cancel

# 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.)

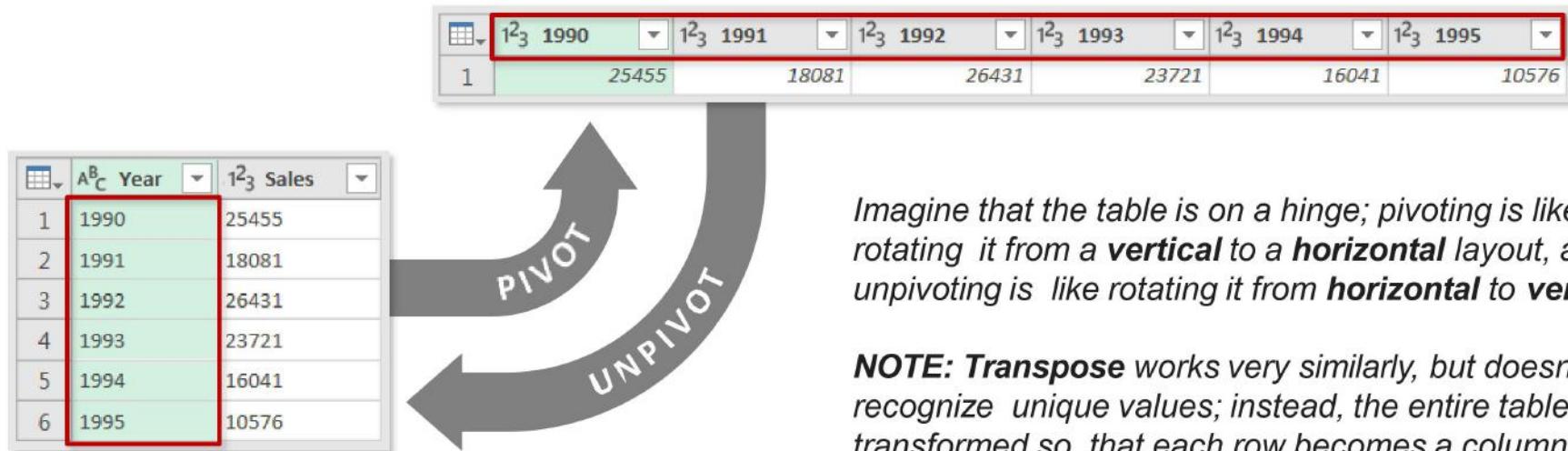


In this case we're transforming a daily, transaction-level table into a summary of “**quantity**” by “**product\_id**”

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

# PIVOTING &

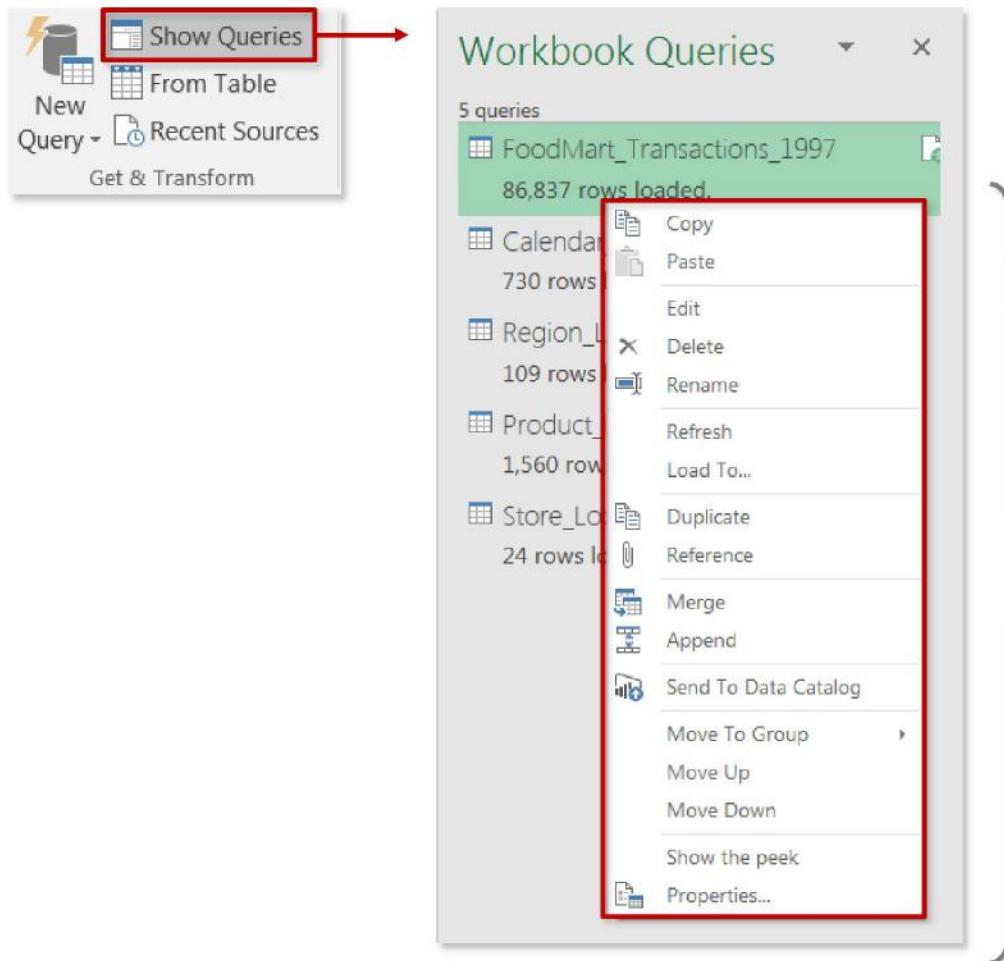
“Pivoting” is a fancy way to describe the process of turning **distinct row values** into **columns** (“*pivoting*”) or turning **columns** into **rows** (“*unpivoting*”)



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

# MODIFYING WORKBOOK



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**

# MERGING

The screenshot shows the 'Merge' dialog box in Microsoft Power BI. On the left, a sidebar lists options: 'Merge Queries' (selected and highlighted with a red box), 'Append Queries', 'Combine Binaries', and 'Combine'. The main area contains two tables: 'FoodMart\_Transactions\_1997' and 'Product\_Lookup'. The 'FoodMart\_Transactions\_1997' table has columns: date, product\_id, customer\_id, promotion\_id, store\_id, quantity, product\_brand, and product\_name. The 'Product\_Lookup' table has columns: product\_id, product\_brand, product\_name, product\_sku, product\_retail\_price, and product\_cost. Below the tables, a 'Join Kind' dropdown is set to 'Left Outer (all from first, matching from second)'. A status message at the bottom says: '✓ The selection has matched 86837 out of the first 86837 rows.' At the bottom right are 'OK' and 'Cancel' buttons.

- 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

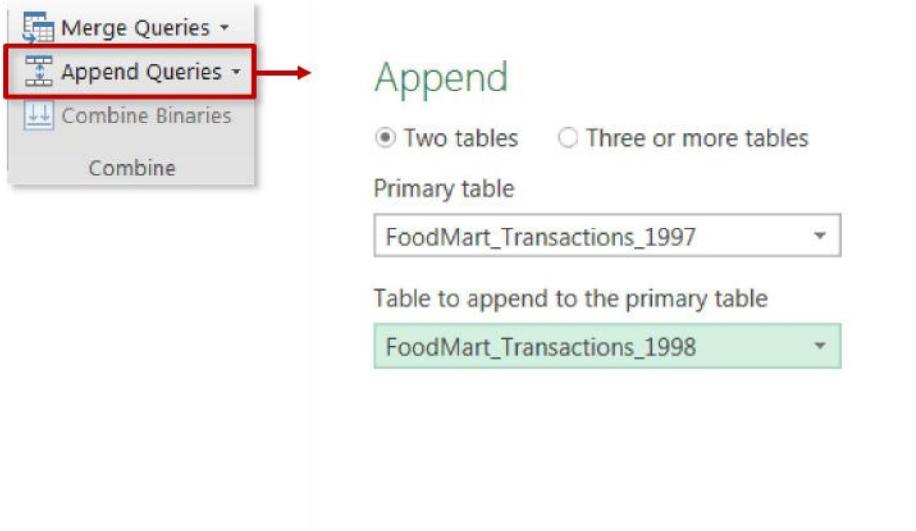
**TIP:** Merging adds columns to an existing table  
**HEY THIS IS IMPORTANT!**



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!)

# 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

## POWER QUERY BEST



**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**

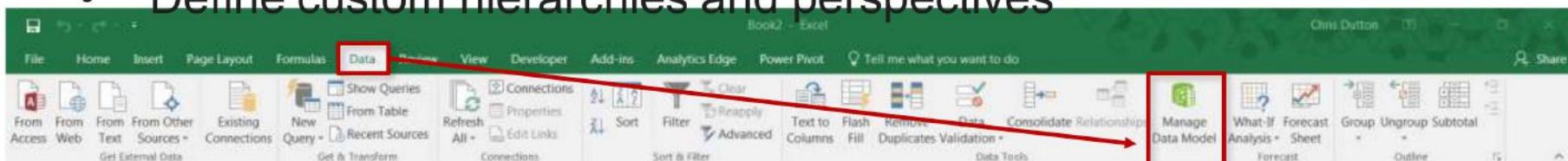


# DATA MODELING

# MEET EXCEL'S DATA

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

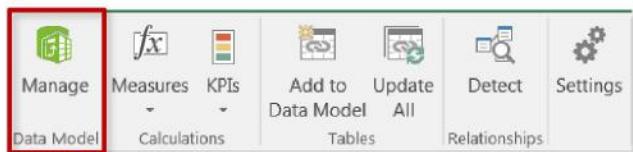
- Manage massive datasets that can't fit into worksheets
- Create table relationships to blend data across multiple sources
- Define custom hierarchies and perspectives



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 > Manage Add-Ins*)

# THE DATA MODEL



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

A screenshot of the Power Pivot window titled 'Power Pivot for Excel - FoodMart\_Data\_Model\_WIP.xlsx'. The window shows a data table with columns: date, product\_id, customer\_id, promotion\_id, store\_id, and quantity. The data consists of 10 rows of transaction records. Below the table, a ribbon bar includes tabs for Home, Design, Advanced, and various data-related functions like Paste Append, Refresh, and PivotTable. At the bottom, there are tabs for FoodMart\_Transactions, Store\_Lookup, Product\_Lookup, Customer\_Lookup, Promotion\_Lookup, and Calendar\_Lookup, along with a record counter showing '1 of 86,837'.

# DATA VIEW VS. DIAGRAM

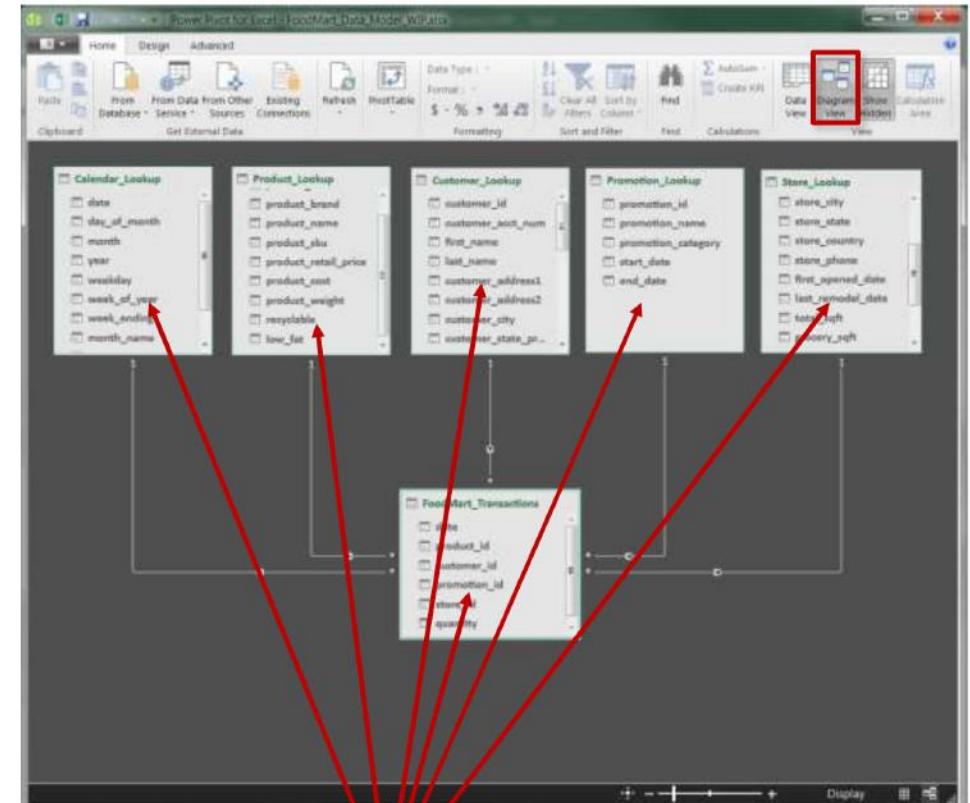
## DATA VIEW

This screenshot shows the Microsoft Access 'Data View' interface. It displays a grid of transaction data with columns for date, product\_id, customer\_id, promotion\_id, store\_id, quantity, and row\_id. The data is organized into 37 rows. At the bottom of the window, there are tabs for 'FoodMart\_Transactions', 'Store\_Lookup', 'Product\_Lookup', 'Customer\_Lookup', 'Promotion\_Lookup', and 'Calendar\_Lookup'. The 'Data View' button in the ribbon is highlighted with a red box.

[date]	product_id	customer_id	promotion_id	store_id	quantity	row_id
7/27/1997 12:00:00 AM	1334	3776	0	13	3	1
7/27/1997 12:00:00 AM	1524	3776	0	13	3	2
7/27/1997 12:00:00 AM	1467	6574	0	13	3	3
7/27/1997 12:00:00 AM	182	6574	0	13	3	4
7/27/1997 12:00:00 AM	721	1954	0	13	3	5
7/27/1997 12:00:00 AM	644	1954	0	13	3	6
7/27/1997 12:00:00 AM	987	9788	0	13	3	7
7/27/1997 12:00:00 AM	1403	9788	0	13	3	8
7/27/1997 12:00:00 AM	155	9788	0	13	3	9
7/27/1997 12:00:00 AM	1197	9788	0	13	3	10
7/27/1997 12:00:00 AM	1124	7546	0	13	3	11
7/27/1997 12:00:00 AM	968	7546	0	13	3	12
7/27/1997 12:00:00 AM	292	7546	0	13	3	13
7/27/1997 12:00:00 AM	762	7546	0	13	3	14
7/27/1997 12:00:00 AM	186	9530	0	13	3	15
7/27/1997 12:00:00 AM	585	4342	0	13	3	16
7/27/1997 12:00:00 AM	1374	4342	0	13	3	17
7/27/1997 12:00:00 AM	882	4804	0	13	3	18
7/27/1997 12:00:00 AM	356	4804	0	13	3	19
7/27/1997 12:00:00 AM	959	4804	0	13	3	20
7/27/1997 12:00:00 AM	832	545	0	13	3	21
7/27/1997 12:00:00 AM	422	3548	0	13	3	22
7/27/1997 12:00:00 AM	690	6686	0	13	3	23
7/27/1997 12:00:00 AM	1173	6686	0	13	3	24
7/27/1997 12:00:00 AM	952	6686	0	13	3	25
7/27/1997 12:00:00 AM	497	5855	0	13	3	26
7/27/1997 12:00:00 AM	753	3280	0	13	3	27

Tables organized in  
tabs

## DIAGRAM



Tables organized as  
objects



# DATABASE

**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



# DATA TABLES VS. LOOKUP

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	58427711925	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.)



# PRIMARY & FOREIGN

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

# RELATIONSHIPS VS. MERGED



*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

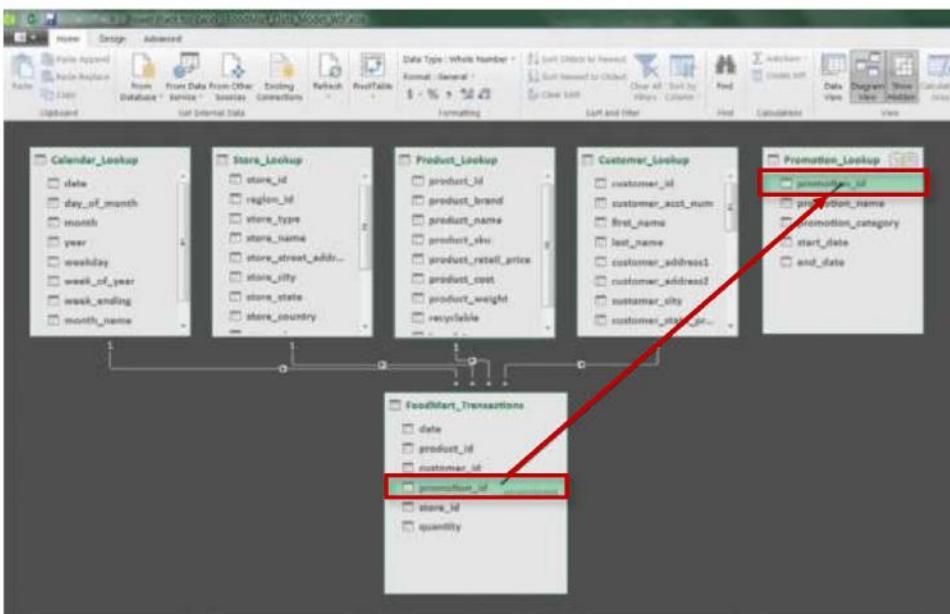
Original Fact Table fields		Attributes from Calendar Lookup table								Attributes from Product Lookup table		
date	product_id	quantity	day_of_month	month	year	weekday	month_name	quarter	product_brand	product_name	product_sku	product_weight
1/1/1997	869	5	1	1	1997	Wednesday	January	Q1	Nationeel	Nationeel 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

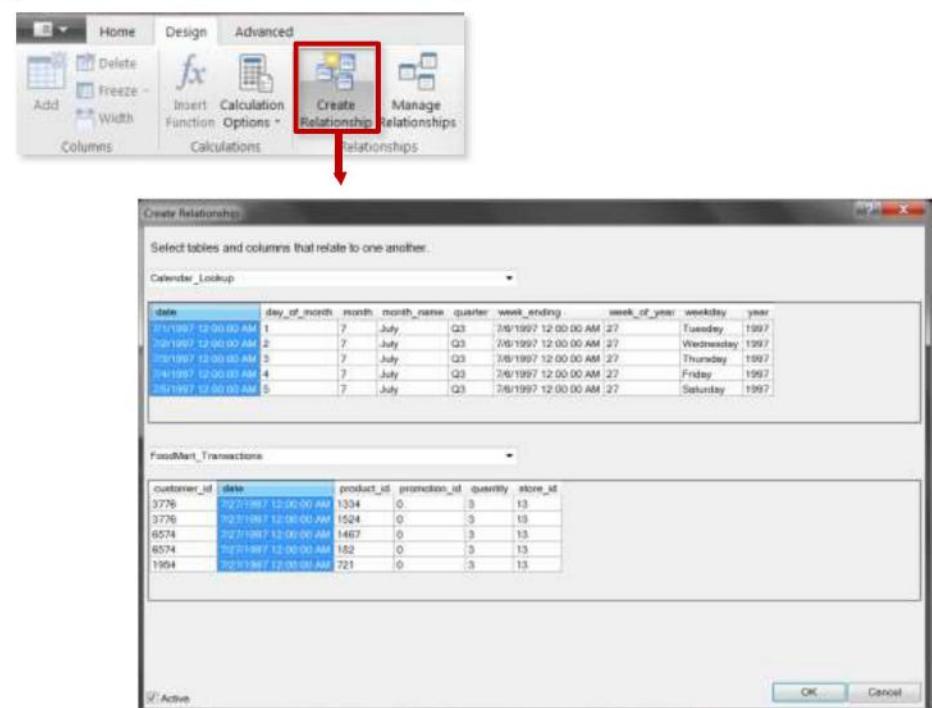
# CREATING TABLE

Option 1: Click and drag relationships in Diagram View



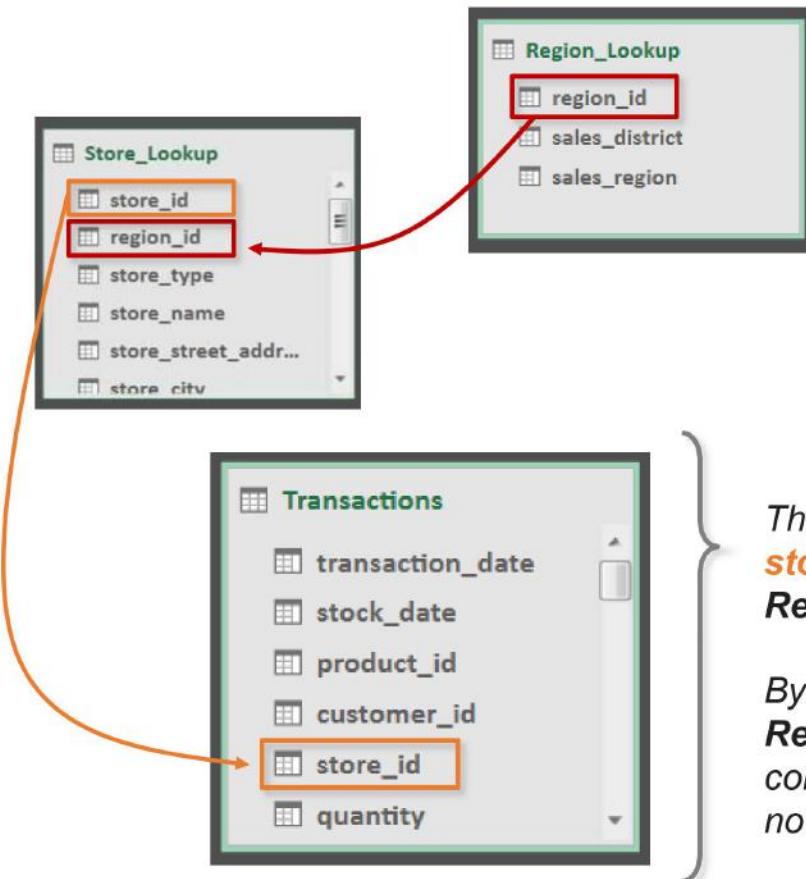
*Tip:* Always drag relationships from the **Data** table to the **Lookup** tables

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

# 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

# MODIFYING TABLE



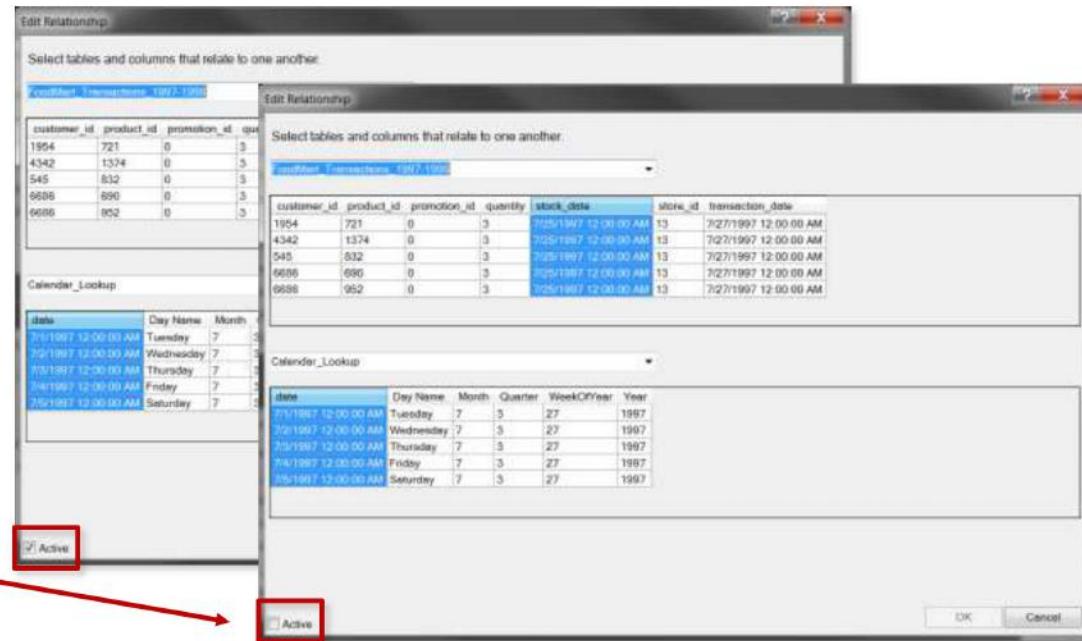
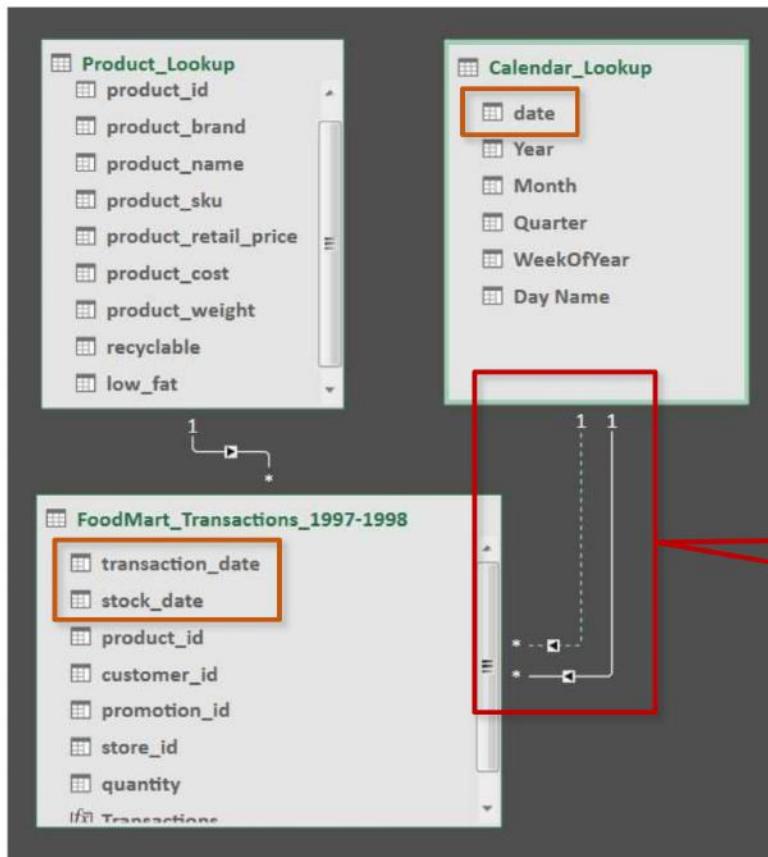
The **Manage Relationships** window allows you to create, edit or delete any connection in the data model

- 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

The 'Manage Relationships' dialog box is open, showing a grid of table relationships. The columns are Active, Table 1, Cardinality, Filter Direction, and Table 2. The grid contains 13 rows of data, each representing a relationship between two tables with specific cardinality and filter directions.

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	Calender_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	Calender_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	Calender_Lookup [date]

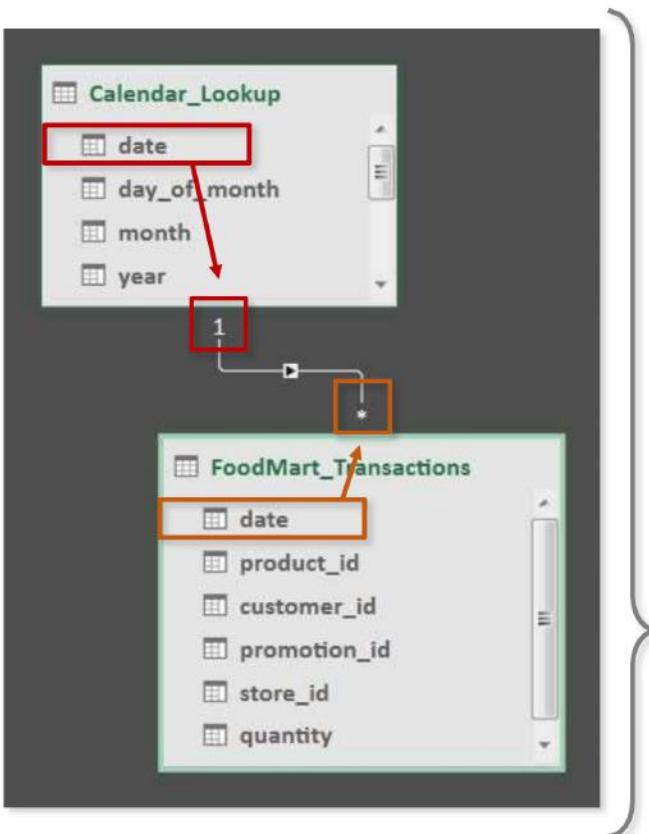
# ACTIVE VS. INACTIVE



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!)

# RELATIONSHIP



**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 defining a one-to-many relationship from 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

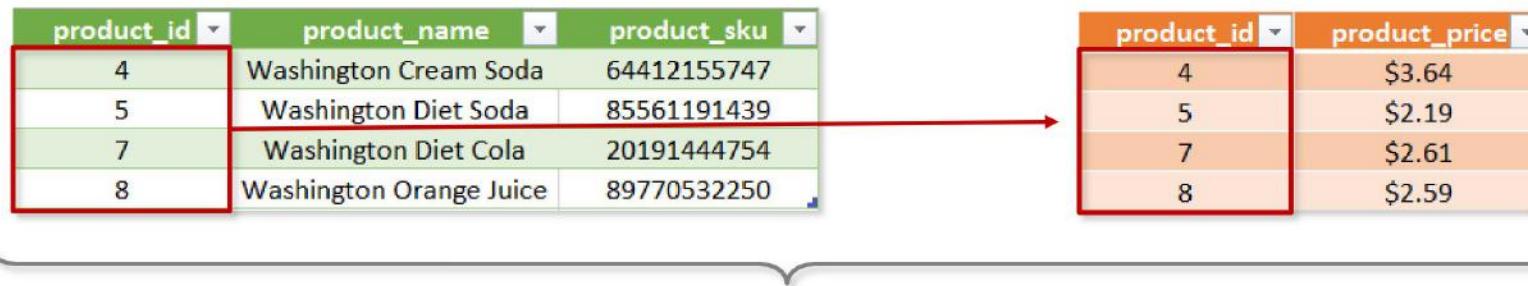
## BAD CARDINALITY: MANY-TO-

product_id	product_name	product_sku	date	product_id	transactions
4	Washington Cream Soda	64412155747	1/1/2017	4	12
4	Washington Diet Cream Soda	81727382373	1/2/2017	4	9
5	Washington Diet Soda	85561191439	1/3/2017	4	11
7	Washington Diet Cola	20191444754	1/1/2017	5	16
8	Washington Orange Juice	89770532250	1/2/2017	5	19
			1/1/2017	7	11

A Power Pivot for Excel dialog box is shown, stating: "The relationship cannot be created because each column contains duplicate values. Select at least one column that contains only unique values." An "OK" button is visible.

- 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**?

# BAD CARDINALITY: ONE-



- 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**

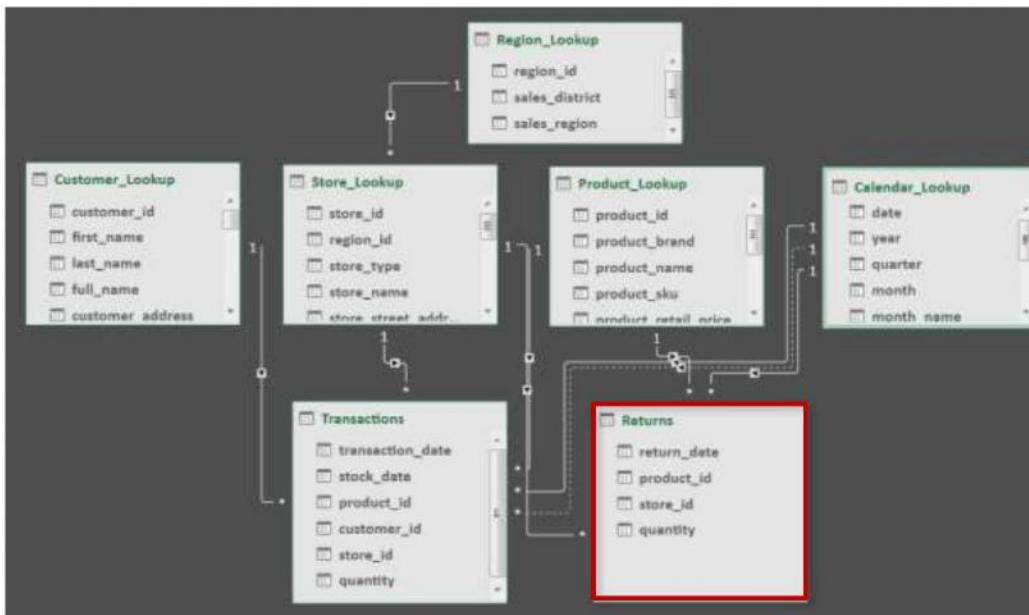
*To eliminate the inefficiency, you could simply merge the two tables into a single, valid lookup*

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

The diagram shows a large brace grouping the two original tables together, indicating they have been merged into a single table. This merged table has four columns: product\_id, product\_name, product\_sku, and product\_price. It contains the same four rows as the original tables, with the row for product\_id 5 highlighted.

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

# CONNECTING MULTIPLE DATA



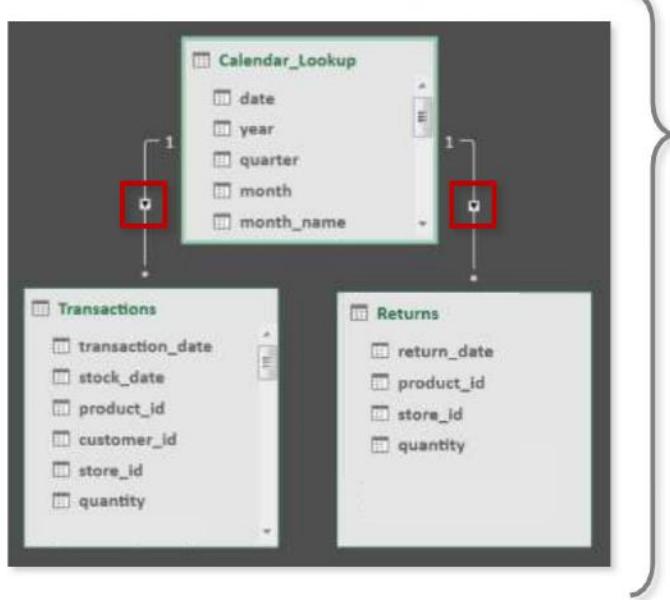
Here we've loaded a second data table named **Returns**, containing records of returns by date, product and store

- 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!

# FILTER DIRECTION IS



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

PivotTable Fields ▾ ×

Active All

Choose fields to add to report:

Search

Calendar\_Lookup   
date   
Year   
Month   
Quarter   
WeekOfYear   
Day Name

FoodMart\_Returns

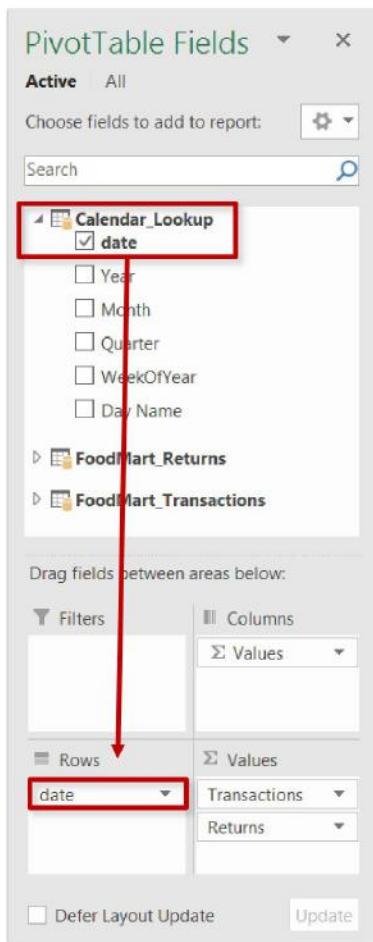
FoodMart\_Transactions

Drag fields between areas below:

Filters   
Rows  date   
Columns   
Values

transaction\_date   
stock\_date   
product\_id

Defer Layout Update  Update



	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



PivotTable Fields ▾ ×

Active All

Choose fields to add to report:

Relationships between tables may be needed.

Auto-Detect... CREATE...

Search

Calendar\_Lookup

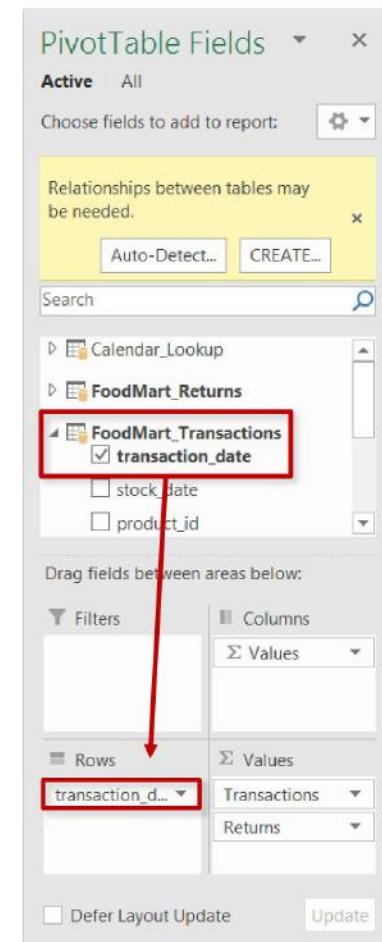
FoodMart\_Returns

FoodMart\_Transactions   
transaction\_date   
stock\_date   
product\_id

Drag fields between areas below:

Filters   
Rows  transaction\_d...   
Columns   
Values

Defer Layout Update  Update

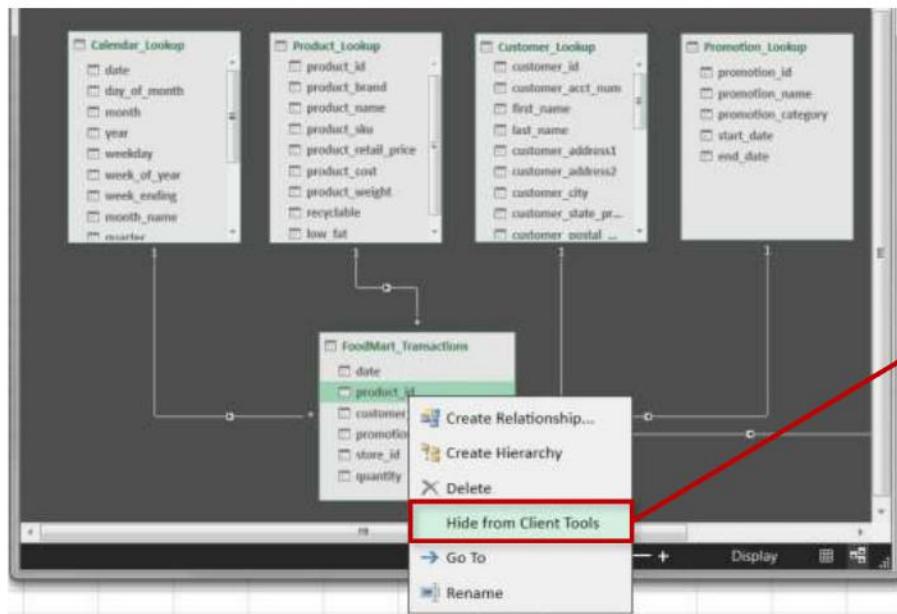


	A	B	C
1	Row Labels	Transactions	Returns
2	1/1/1997	348	8,289
3	1/2/1997	635	8,289
4	1/3/1997	589	8,289
5	1/4/1997	20	8,289
6	1/5/1997	966	8,289
7	1/6/1997	993	8,289
8	1/7/1997	1,265	8,289
9	1/8/1997	35	8,289
10	1/9/1997	525	8,289
11	1/10/1997	460	8,289

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



# 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



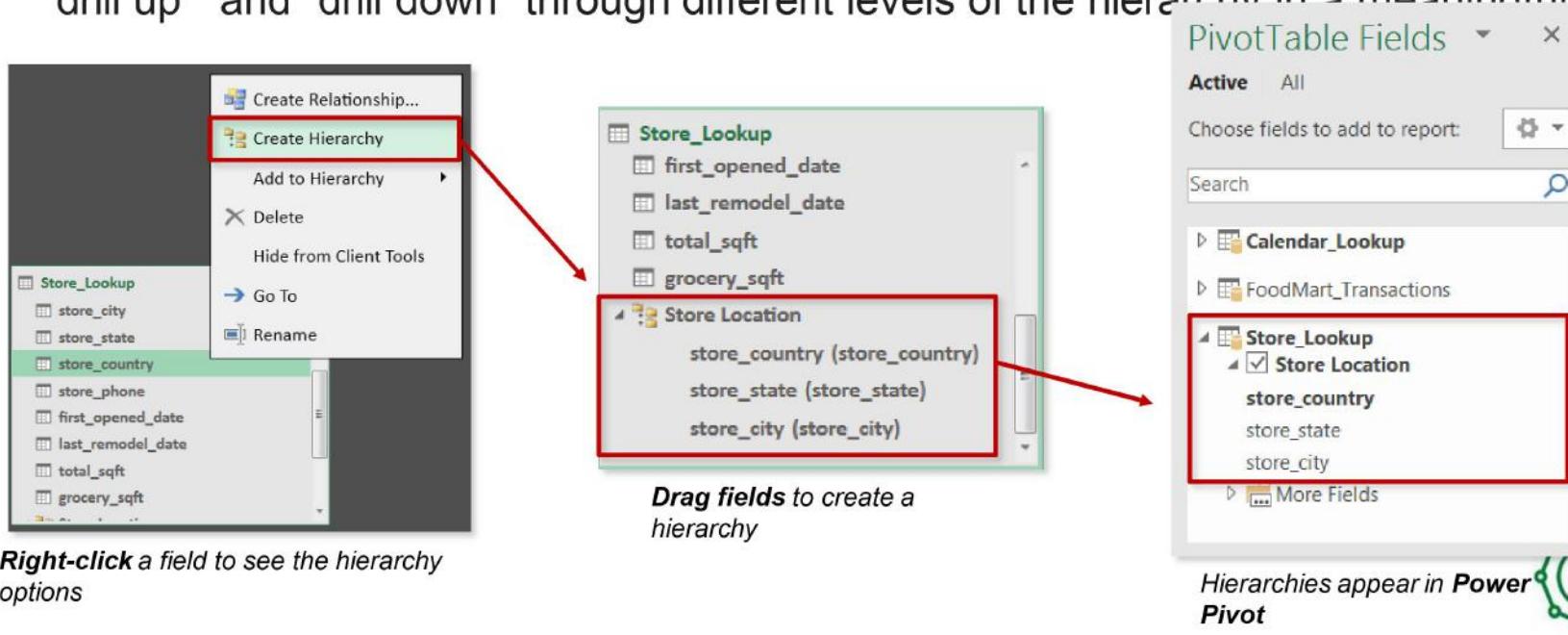
## PRO TIP:

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

# DEFINING

**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



## DATA MODEL BEST



### 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



### Hide fields from client tools to prevent invalid filter context

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

# POWER PIVOT & DAX

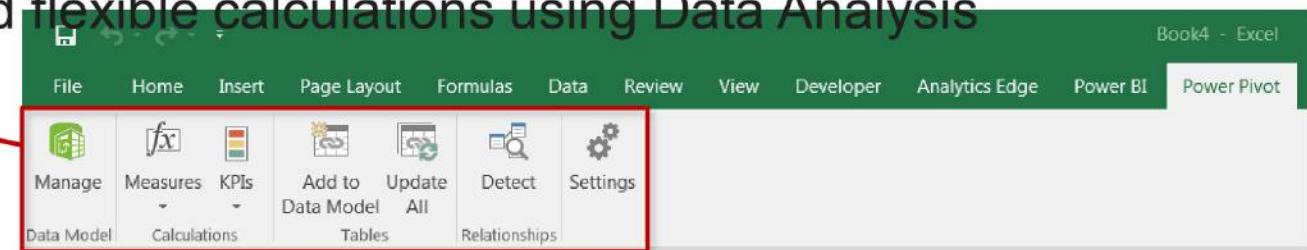


# MEET POWER

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



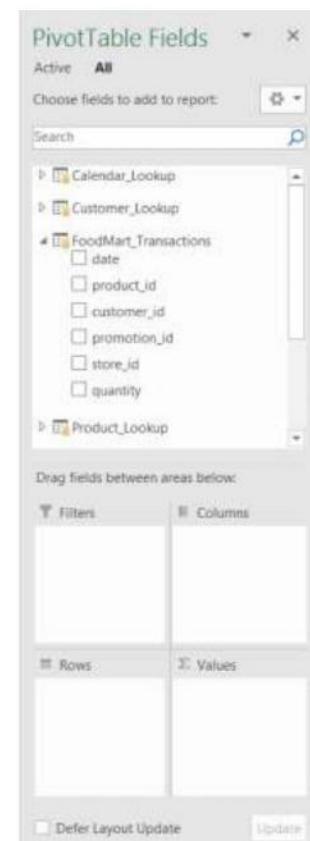
(Note: you may need to enable this tab by selecting  
File > Options > Add-Ins > Manage COM  
Add-Ins)



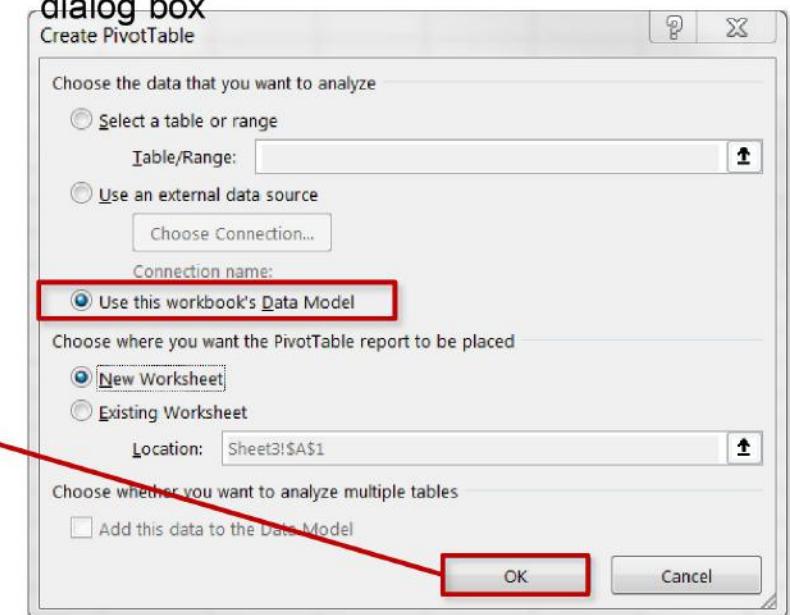
# CREATING A “POWER” PIVOT

## Option #1: From the Data Model

A screenshot of the Microsoft Excel ribbon. The 'Power Pivot' tab is selected, indicated by a red box around its icon. The ribbon tabs include Home, Design, Analyze, Power Pivot, Get External Data, PivotTable, PivotChart, Formulas, Data Tools, Add-ins, and Help.



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



# “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*

# “NORMAL” PIVOTS VS. “POWER”

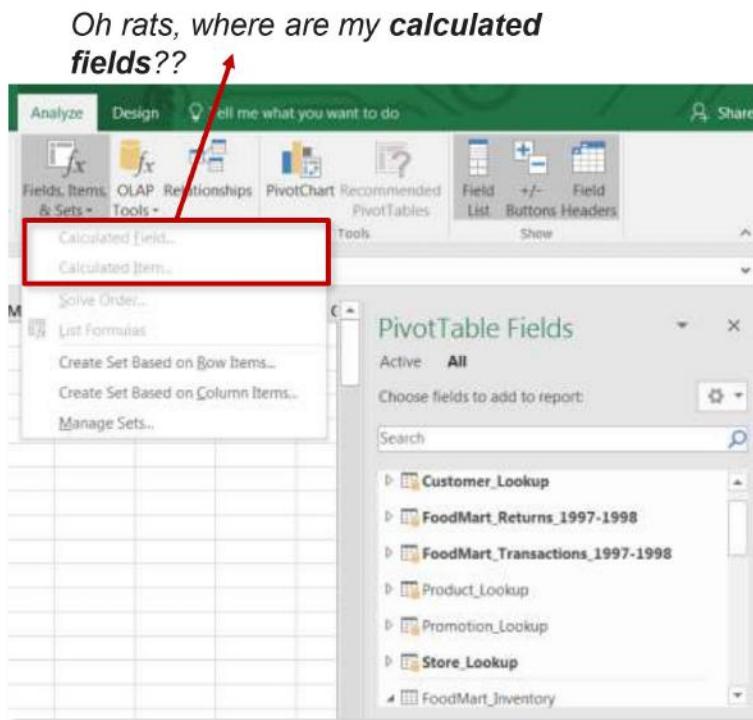
## Normal Pivot Pivot

A screenshot of a Microsoft Excel spreadsheet titled "Regular Pivot". The data is in columns A and B, with rows 1 through 30. Column A contains dates from 1/1/1997 to 1/24/1997. Column B contains the sum of quantity for each date. The PivotTable Fields pane on the right shows fields for transaction\_date, product\_id, customer\_id, promotion\_id, store\_id, and quantity, all checked. The "Sum of quantity" field is selected under Values. The "transaction\_date" field is selected under Rows.

## Power

A screenshot of a Microsoft Excel spreadsheet titled "Power Pivot". The data is in columns A and B, with rows 1 through 30. Column A contains dates from 1/1/1997 to 1/24/1997. Column B contains the total quantity for each date. The PivotTable Fields pane on the right shows fields for date, product\_id, customer\_id, promotion\_id, and store\_id, all checked. The "Total Quantity" field is selected under Values. The "date" field is selected under Rows. A red box highlights the "More Tables!" link in the PivotTable Fields pane, which is expanded to show "Calendar\_Lookup", "Customer\_Lookup", "FoodMart\_Transactions\_1997", "Product\_Lookup", "Promotion\_Lookup", and "Store\_Lookup".

# NO MORE “CALCULATED



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)**

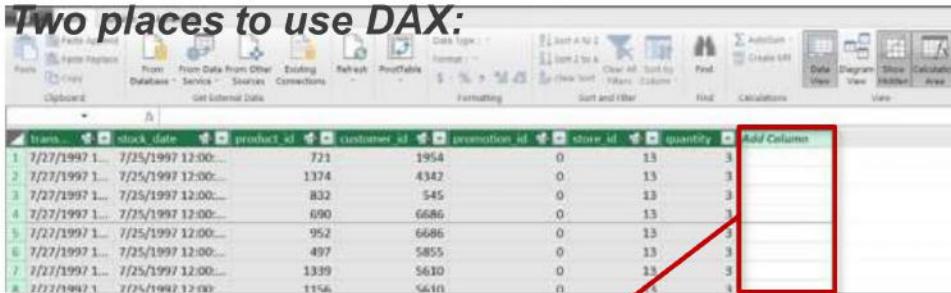
\*Note: Depending on the version of Excel you're using, you might see these referred to as either “**Measures**” (Excel 2010, 2016) or “**Calculated Fields**” (Excel 2013)

# DATA ANALYSIS

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

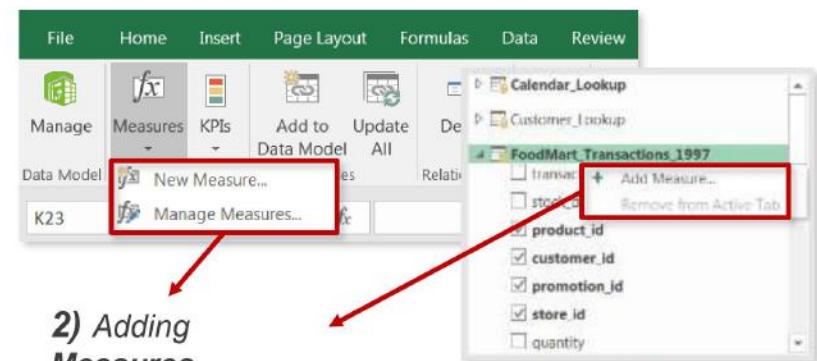
- Add **calculated columns** and **measures** to your model, using intuitive syntax
- Go beyond the capabilities of traditional “grid-style” formulas, with powerful functions built specifically to work with relational data

## Two places to use DAX:



trans	stock_date	product_id	customer_id	promotion_id	store_id	quantity
1	7/27/1997 12:00:00	723	1954	0	13	3
2	7/27/1997 12:00:00	1374	4342	0	13	3
3	7/27/1997 12:00:00	832	545	0	13	3
4	7/27/1997 12:00:00	690	6686	0	13	3
5	7/27/1997 12:00:00	952	6686	0	13	3
6	7/27/1997 12:00:00	497	5855	0	13	3
7	7/27/1997 12:00:00	1339	5610	0	13	3
8	7/27/1997 12:00:00	1156	5610	0	13	3

1) Adding Calculated Columns



File Home Insert Page Layout Formulas Data Review

Manage Measures KPIs Add to Update Data Model All

New Measure... Add Measure... Remove from Active Tab

K23

2) Adding Measures

# CALCULATED

**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)

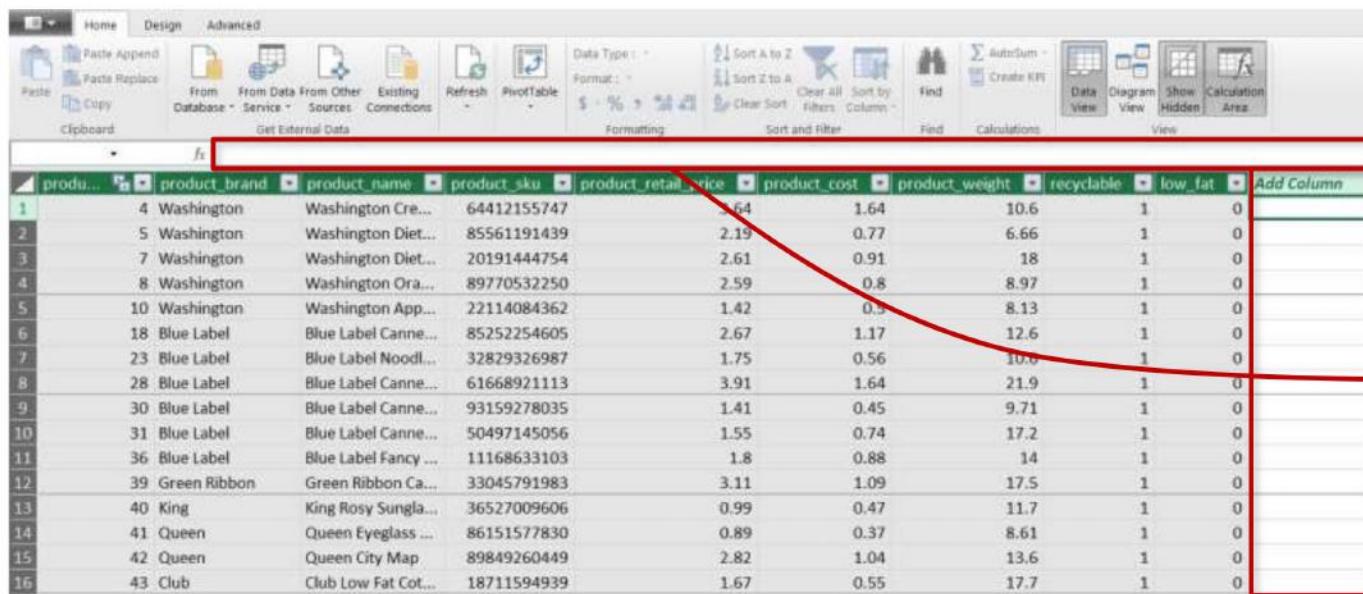


## PRO TIP:

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

# CREATING CALCULATED

## COLUMNS



A screenshot of the Microsoft Power BI Data View interface. The top ribbon shows tabs for Home, Design, Advanced, and various data import options like Paste Append, Paste Replace, From Database, Refresh, and PivotTable. The main area displays a table with 16 rows and 10 columns. The columns are labeled: product\_id, product\_brand, product\_name, product\_sku, product\_retail\_price, product\_cost, product\_weight, recyclable, low\_fat, and Add Column. The 'Add Column' column is highlighted with a red box and an arrow pointing to it from the first step description. The formula bar at the top has an 'fx' icon and is currently empty.

product_id	product_brand	product_name	product_sku	product_retail_price	product_cost	product_weight	recyclable	low_fat	Add Column
1	4 Washington	Washington Cre...	64412155747	6.64	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

# CALCULATED COLUMNS: GOOD

The screenshot shows the Microsoft Power BI Data Editor interface. A calculated column named 'price\_category' has been added to the table. The formula used is =IF(Product\_Lookup[product\_retail\_price]>2,"High","Low"). The column values are displayed as 'High' or 'Low'. A red box highlights the formula bar, and another red box highlights the 'price\_category' column header. A green thumbs-up icon is overlaid on the red box around the column header.

	product_id	product_brand	product_name	product_sku	product_retail_price	product_cost	product_weight	recyclable	new_set	price_category
1	4	Washington	Washington Crea...	6441215347	3.64	1.64	10.6	1	0	High
2	5	Washington	Washington Diet...	85561191439	2.19	0.77	6.66	1	0	High
3	7	Washington	Washington Diet...	20191444754	2.61	0.91	18	1	0	High
4	8	Washington	Washington Oba...	89770532250	2.59	0.8	8.97	1	0	High
5	10	Washington	Washington App...	22114084362	1.42	0.5	8.13	1	0	Low
6	18	Blue Label	Blue Label Canne...	85252254605	2.67	1.17	12.6	1	0	High
7	23	Blue Label	Blue Label Noug...	32829326987	1.75	0.56	10.6	1	0	Low
8	28	Blue Label	Blue Label Canne...	61668921133	3.91	1.64	21.9	1	0	High
9	30	Blue Label	Blue Label Canne...	93139278035	1.41	0.45	9.71	1	0	Low
10	31	Blue Label	Blue Label Canne...	50497145054	1.55	0.74	17.2	1	0	Low
11	36	Blue Label	Blue Label Fancy ...	11168631303	1.8	0.88	18	1	0	Low
12	39	Green Ribbon	Green Ribbon Ca...	33045791983	3.11	1.09	17.5	1	0	High
13	40	King	King Rovy Surgle...	36527009906	0.99	0.47	11.7	1	0	Low
14	41	Queen	Queen Eyepleas...	86153577830	0.89	0.37	8.61	1	0	Low
15	42	Queen	Queen City Map	89849260449	2.82	1.04	13.6	1	0	High
16	43	Club	Club Low Fat Col...	18711594939	1.67	0.55	17.7	1	0	Low

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

The screenshot shows the Microsoft Power BI Data Editor interface. A calculated column named 'total\_revenue' has been added to the table. The formula used is =SUM(Transactions[revenue]). The column values are displayed as monetary amounts. A red box highlights the formula bar, and another red box highlights the 'total\_revenue' column header. A red arrow points from the formula bar to the 'total\_revenue' column header.

	transaction_id	stock_date	product_id	customer_id	promotion_id	store_id	quantity	retail_price	revenue	total_revenue
1	12/15/1998 12:00:00	12/8/1998 1...	4	3303	0	19	3	3.64	10.92	\$1,199,308.31
2	12/15/1998 12:00:00	12/10/1998 ...	6	305	0	19	3	1.15	3.45	\$1,199,308.31
3	12/15/1998 12:00:00	12/13/1998 ...	20	345	0	19	3	2.78	8.34	\$1,199,308.31
4	12/15/1998 12:00:00	12/8/1998 1...	48	3772	0	19	3	1.88	5.64	\$1,199,308.31
5	12/15/1998 12:00:00	12/13/1998 ...	56	3303	0	19	3	0.71	2.13	\$1,199,308.31
6	12/15/1998 12:00:00	12/12/1998 ...	71	4028	0	19	3	2.76	8.28	\$1,199,308.31
7	12/15/1998 12:00:00	12/11/1998 ...	79	9561	0	19	3	1.32	3.96	\$1,199,308.31
8	12/15/1998 12:00:00	12/12/1998 ...	104	8895	0	19	3	3.89	11.67	\$1,199,308.31
9	12/15/1998 12:00:00	12/12/1998 ...	111	5565	0	19	3	3.48	10.44	\$1,199,308.31
10	12/15/1998 12:00:00	12/12/1998 ...	123	5577	0	19	3	3.84	11.52	\$1,199,308.31

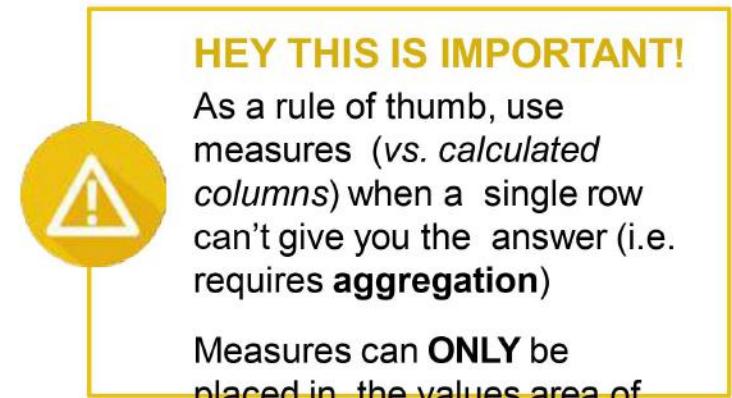
Here we're using an aggregation function (SUM) to calculate a new column named **total\_revenue**

- 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.

# DAX

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

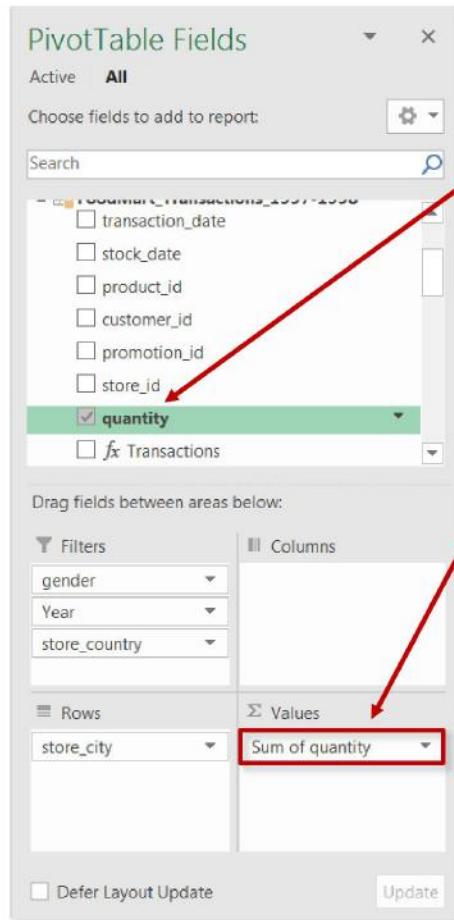
- Like calculated columns, measures reference **entire tables**
- or **columns** (*no A1-style or “grid” references*)
- Unlike calculated columns, measures don’t actually *live* in the table; they get placed in the **values** area of a PivotTable and dynamically calculated in each individual cell
- Measures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



## PRO TIP:

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

# CREATING IMPLICIT



**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**



# CREATING EXPLICIT MEASURES

The screenshot shows the Microsoft Power BI Data View interface. A red box highlights the 'AutoSum' button in the ribbon toolbar. A larger red box highlights the 'Revenue\_CC' column header, which has a red border. A context menu is open over the 'Revenue\_CC' header, listing options: Sum, Average, Count, Distinct Count, Max, and Min. The 'Sum' option is selected. The Data View displays a table of transaction data with columns: transaction\_id, stock\_date, product\_id, customer\_id, promotion\_id, store\_id, quantity, product\_name, and Revenue\_CC.

transaction_id	stock_date	product_id	customer_id	promotion_id	store_id	quantity	product_name	Revenue_CC
1	12/15/1998 12:00:00	12/8/1998 1...	4	3303	0	19	3	10.92
2	12/15/1998 12:00:00	12/10/1998 ...	6	305	0	19	3	3.45
3	12/15/1998 12:00:00	12/13/1998 ...	20	116	0	19	3	2.78
4	12/15/1998 12:00:00	12/8/1998 1...	48	3772	0	19	3	1.88
5	12/15/1998 12:00:00	12/13/1998 ...	56	3303	0	19	3	0.71
6	12/15/1998 12:00:00	12/12/1998 ...	71	4028	0	19	3	2.76
7	12/15/1998 12:00:00	12/11/1998 ...	79	9561	0	19	3	1.32
8	12/15/1998 12:00:00	12/12/1998 ...	104	8895	0	19	3	3.89
9	12/15/1998 12:00:00	12/12/1998 ...	111	5565	0	19	3	3.48
10	12/15/1998 12:00:00	12/12/1998 ...	123	5577	0	19	3	3.84
11	12/15/1998 12:00:00	12/11/1998 ...	130	2005	0	19	3	2.23
12	12/15/1998 12:00:00	12/13/1998 ...	141	2148	0	19	3	3.93

**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!

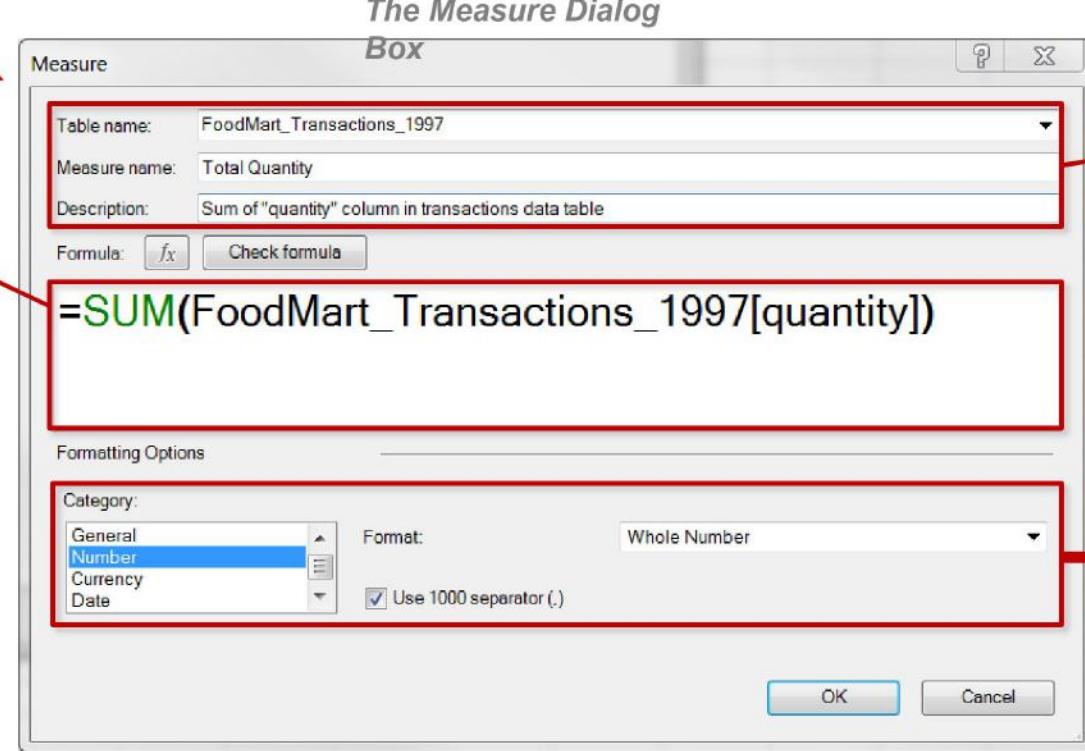
# CREATING EXPLICIT MEASURES



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 automatically 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

# UNDERSTANDING FILTER

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
<b>Grand Total</b>	<b>226,838</b>

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

# FILTER CONTEXT

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	All
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	Vancouver	Victoria	Grand Total
product_brand				
ADJ		30	10	40
Akron		50	15	65
American		34	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"**

# STEP-BY-STEP MEASURE

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

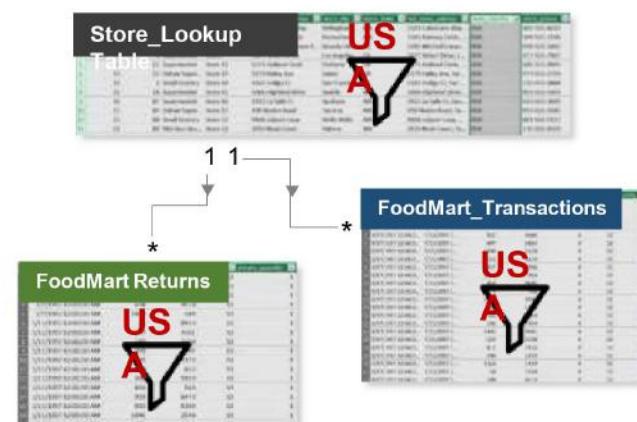
### STEP 1

*Detect pivot coordinates & apply filter context*



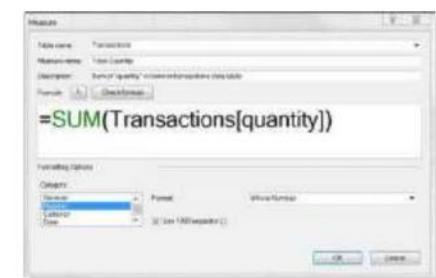
### STEP 2

*Carry filters “downstream” & apply to all related tables*



### STEP 3

*Evaluate the measure formula against the filtered table*



*Sum of Transactions[quantity] when store\_country = "USA"*

=  
**555,899**  
Quest



# RECAP: CALCULATED COLUMNS VS.

## 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



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

- 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	X
Total Quantity	store_city	Grand Total
product_brand	Vancouver	Victoria
ADJ	30	10
Akron	50	15
American	384	103
Amigo	50	31
		487
		81

Measures “live” in PivotTables



# POWER PIVOT BEST



## 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



# DAX

## 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 preceded by the table name

Note: 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])



# DAX

Arithmetic Operator	Meaning	Example
+	Addition	$2 + 7$
-	Subtraction	$5 - 3$
*	Multiplication	$2 * 6$
/	Division	$4 / 2$
$\wedge$	Exponent <i>Hey! Pay attention to these!</i>	$2 \wedge 5$

Comparison Operator	Meaning	Example
=	Equal to	[City]=“Boston”
>	Greater than	[Quantity]>10
<	Less than	[Quantity]<10
$\geq$	Greater than or equal to	[Unit_Price] $\geq$ 2.5
$\leq$	Less than or equal to	[Unit_Price] $\leq$ 2.5
$\neq$	Not equal to	[Country] $\neq$ “Mexico”

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

\*Head to [www.msdn.microsoft.com](http://www.msdn.microsoft.com) for more information about DAX syntax, operators, troubleshooting, etc.



# COMMON FUNCTION

MATH & STATS	LOGIC AL	TEXT Function	FILTER Function	DATE & TIME
<p><b>Basic aggregation functions as well as “iterators” evaluated at the row-level</b></p> <p><b>Common Examples:</b></p> <ul style="list-style-type: none"> <li>SUM</li> <li>AVERAGE</li> <li>MAX/MIN</li> <li>DIVIDE</li> <li>COUNT/COUN TA</li> <li>COUNTROWS</li> <li>DISTINCTCOU NT</li> </ul> <p><b>Iterator Functions:</b></p> <ul style="list-style-type: none"> <li>SUMX</li> <li>AVERAGE X</li> <li>MAXX/MIN X</li> <li>RANKX</li> <li>COUNTX</li> </ul>	<p><b>Functions for returning information about values in a given conditional expression</b></p> <p><b>Common Examples:</b></p> <ul style="list-style-type: none"> <li>IFERR</li> <li>OR</li> <li>AND</li> <li>OR</li> <li>NOT</li> <li>SWITC H</li> <li>TRUE</li> <li>FALSE</li> </ul>	<p><b>Functions to manipulate text strings or control formats for dates, times or numbers</b></p> <p><b>Common Examples:</b></p> <ul style="list-style-type: none"> <li>CONCATENAT E</li> <li>FORMAT</li> <li>LEFT/MID/RIG HT</li> <li>UPPER/LOWE R</li> <li>PROPER</li> <li>LEN</li> <li>SEARCH/FIND</li> <li>REPLACE</li> <li>REPT</li> <li>SUBSTITUTE</li> <li>TRIM</li> <li>UNICHAR</li> </ul>	<p><b>Lookup functions based on related tables and filtering functions for dynamic calculations</b></p> <p><b>Common Examples:</b></p> <ul style="list-style-type: none"> <li>CALCULATE</li> <li>FILTER</li> <li>ALL</li> <li>ALLEXCEPT</li> <li>RELATED</li> <li>RELATEDTABLE</li> <li>DISTINCT</li> <li>VALUES</li> <li>EARLIER/EARLI EST</li> <li>HASONEVALUE</li> <li>HASONEFILTER</li> <li>ISFILTERED</li> <li>USERELATION SHIP</li> </ul>	<p><b>Basic date and time functions as well as advanced time intelligence operations</b></p> <p><b>Common Examples:</b></p> <ul style="list-style-type: none"> <li>DATEDIFF</li> <li>YEARFRAC</li> <li>YEAR/MONTH/DAY</li> <li>HOUR/MINUTE/SEC OND</li> <li>TODAY/NOW</li> <li>WEEKDAY/WEEKNU MBER</li> </ul> <p><b>Time Intelligence Functions:</b></p> <ul style="list-style-type: none"> <li>DATESYTD</li> <li>DATESQTD</li> <li>DATESMTD</li> <li>DATEADD</li> <li>DATESINPERIOD</li> </ul>
<p><b>*Note:</b> This is NOT a comprehensive list (does not include trigonometry functions, parent/child functions, information functions, or other less common functions)</p>				



# BASIC MATH & STATS

**SUM(**

*Evaluates the sum of a column*

=**SUM**(<column>)

**AVERAG**

*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>)



# BASIC MATH & STATS FUNCTIONS

Sum of quantity from the Transactions table

Measure

Table name: Transactions  
Measure name: Total Quantity  
Description: Sum of "quantity" column in transactions data table  
Formula: `=SUM(Transactions[quantity])`

Formatting Options

Category: Number  
Format: Whole Number  
Symbol:  Use 1000 separator (.)

OK Cancel

Average of product\_retail\_price

Measure

Table name: Product\_Lookup  
Measure name: Avg Retail Price  
Description:  
Formula: `=AVERAGE(Product_Lookup[product_retail_price])`

Formatting Options

Category: Number  
Format: Decimal places: 2  
Symbol:  Use 1000 separator (.)

OK Cancel

Quantity Returned divided by Total Quantity

Measure

Table name: Returns  
Measure name: Return Rate  
Description:  
Formula: `=DIVIDE([Quantity Returned], [Total Quantity])`

Formatting Options

Category: Percentage  
Format: Decimal places: 1  
Symbol:  Use 1000 separator (.)

OK Cancel

## 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

# COUNT, COUNTA, DISTINCTCOUNT &

**COUNTROW**  
()

*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>)



# COUNT FUNCTIONS

*Count of all rows in the Transactions table*

Measure

Table name: Transactions  
Measure name: Transactions  
Description: Number of rows in Transactions table  
Formula: `=COUNTROWS(Transactions)`

Formatting Options

Category: Number Format: Whole Number  
 Use 1000 separator (.)

*Count of unique values in the product\_id column*

Measure

Table name: Product\_Lookup  
Measure name: Unique Products  
Description: Distinct count of product IDs  
Formula: `=DISTINCTCOUNT(Product_Lookup[product_id])`

Formatting Options

Category: Number Format: Whole Number  
 Use 1000 separator (.)

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

Measure

Table name: Product\_Lookup  
Measure name: Recyclable Products  
Description: Counts products where "recyclable" field is non-empty  
Formula: `=COUNTA(Product_Lookup[recyclable])`

Formatting Options

Category: Number Format: Whole Number  
 Use 1000 separator (.)



# 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**  
**R()**

*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!



# BASIC LOGICAL FUNCTIONS

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

	name	education	acct_open_date	member_card	occupation	homeowner	full_name	birth_year	has_children	customer_age	education_level	customer_priority	Add Column
1	O High School	11/16/1994 12:00:00	Bronze	Manual	N	Bertha Jam...	1948	Y		60	Non-Grad	Other	
2	O High School	5/5/1992 12:00:00	Bronze	Manual	N	Ole Weldon	1931	Y		86	Non-Grad	Other	
3	O High School	6/26/1994 12:00:00	Bronze	Manual	N	Paul Alcorn	1973	Y		44	Non-Grad	Other	
4	O High School	2/9/1990 12:00:00	Bronze	Manual	N	Jared Busta	1910	Y		107	Non-Grad	Other	
5	O High School	3/15/1992 12:00:00	Bronze	Manual	N	Margaret A...	1979	Y		38	Non-Grad	Other	
6	O High School	3/2/1994 12:00:00	Bronze	Manual	N	Vanessa Ten...	1930	Y		87	Non-Grad	Other	
7	O High School	6/4/1993 12:00:00	Bronze	Manual	N	Catherine ...	1966	Y		53	Non-Grad	Other	
8	O High School	1/9/1992 12:00:00	Bronze	Manual	N	Stacey Carr...	1943	Y		74	Non-Grad	Other	
9	O High School	5/17/1992 12:00:00	Bronze	Manual	N	Marlin Correll	1933	Y					
10	O High School	9/8/1992 12:00:00	Bronze	Manual	N	Drama Sab...	1916	Y					
11	O High School	2/21/1991 12:00:00	Bronze	Manual	N	Joseph Tho...	1968	Y					
12	O High School	6/12/1994 12:00:00	Bronze	Manual	N	Roberta Stu...	1919	Y					

Supermarket\_size equals "Large" if sq ft >30,000, otherwise "Small"

	address	store_city	store_state	full_store_address	store_country	store_phone	area_code	first_opened_date	last_remodel_date	total_sqft	grocery_sqft	supermarket_size	Auto Col
1	Acapulco	Guerrero		2853 Bakery Rd, Acap...	MEXICO	262-555-5124	262	1/9/1982 12:00:00...	12/5/1990 12:00:00...	23593	17475	Small	
2	Way	Bellingham	WA	5209 Catanzaro Way...	USA	605-555-8203	605	4/1/1970 12:00:00...	6/4/1973 12:00:00...	28206	22271	Small	
3	Dr	Bremerton	WA	1501 Ramsey Circle, ...	USA	509-555-1596	509	6/14/1959 12:00:00...	11/19/1967 12:00:00...	39806	24390	Large	
4	Or	Camacho	Zacatecas	433 St George Dr, Ca...	MEXICO	304-555-1474	304	9/27/1994 12:00:00...	12/1/1995 12:00:00...	23759	16844	Small	
5	Servive	Guadalajara	Jalisco	1250 Coggins Drive, ...	MEXICO	805-555-4324	801	9/18/1978 12:00:00...	6/29/1991 12:00:00...	24597	15012	Small	
6	Anyon R.	Beverly Hills	CA	5495 Mitchell Canyon, ...	USA	958-555-5002	958	1/3/1981 12:00:00...	1/1/1981 12:00:00...	23648	15337	Small	
7	ave	Los Angeles	CA	1077 Wharf Drive, L...	USA	477-555-7967	477	5/21/1971 12:00:00...	10/20/1981 12:00:00...	23598	14216	Small	
8	ta Ave	Mérida	Yucatan	3173 Buena Vista Av...	MEXICO	797-555-3417	797	9/23/1958 12:00:00...	12/18/1967 12:00:00...	30797	20341	Large	
9	Road	Mexico City	DF	1872 El Pintado Roa...	MEXICO	439-555-3524	439	3/18/1955 12:00:00...	6/1/1999 12:00:00...	36509	22450	Large	
10	m Dr	Orizaba	Veracruz	7894 Rutherford Dr, ...	MEXICO	212-555-4774	212	4/13/1979 12:00:00...	1/30/1982 12:00:00...	34791	26354	Large	
11	rtle	Portland	OR	5375 Holland Credit, ...	USA	685-555-8995	685	9/7/1976 12:00:00...	5/15/1982 12:00:00...	20119	16232	Small	
12	ter Pl	Ridgely	Zacatecas	1120 Westchester Pl, ...	MEXICO	151-555-1702	151	3/25/1968 12:00:00...	12/18/1993 12:00:00...	30584	21939	Large	
13	Salem	OR		5179 Valley Ave, Sal...	USA	977-555-2724	977	4/13/1957 12:00:00...	11/10/1997 12:00:00...	27694	18670	Small	
14	San Francisco	CA		4365 Indigo Ct, San ...	USA	135-555-4888	135	11/2/1997 12:00:00...	1/7/1998 12:00:00...	22478	15321	Small	
15	Driver	Seattle	WA	5006 Highland Drive...	USA	893-555-1024	893	7/24/1969 12:00:00...	10/29/1973 12:00:00...	21215	13305	Small	
16	Spokane	WA		5922 La Salle Ct, Spo...	USA	643-555-3645	643	8/23/1974 12:00:00...	7/13/1977 12:00:00...	30268	22063	Large	
17	Tacoma	WA		490 Ruxton Road, Ta...	USA	855-555-5581	855	5/30/1970 12:00:00...	6/2/1976 12:00:00...	33868	22123	Large	
18	Hidalgo	Zacatecas		6764 Glen Road, Hid...	MEXICO	528-555-8317	528	6/28/1969 12:00:00...	8/30/1975 12:00:00...	38382	30351	Large	
19	Tekke	Vancouver	BC	6644 Sundance Drive, ...	CANADA	862-555-7395	862	3/27/1977 12:00:00...	10/25/1990 12:00:00...	23112	16418	Small	
20	n	Victoria	BC	3706 Marville Ln, Vi...	CANADA	897-555-1933	897	2/6/1980 12:00:00...	4/9/1987 12:00:00...	34452	27463	Large	

# 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\_name]
- Product\_Lookup[product\_brand]

### PRO TIP:



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

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

Examples:  
=SWITCH(Calendar\_Lookup[month\_number], 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

# SWITCH & SWITCH(TRUE)

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

quarter_n...	=SWITCH(Calendar_Lookup[quarter], 1,"Q1", 2,"Q2", 3,"Q3", "Q4")
1	7/1/1997 12:0... 1997
2	7/2/1997 12:0... 1997
3	7/3/1997 12:0... 1997
4	7/4/1997 12:0... 1997
5	7/5/1997 12:0... 1997
6	7/6/1997 12:0... 1997
7	7/7/1997 12:0... 1997
8	7/8/1997 12:0... 1997
9	7/9/1997 12:0... 1997
10	7/10/1997 12:0... 1997
11	7/11/1997 12:0... 1997
12	7/12/1997 12:0... 1997

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"

product_...	=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")
1	Washington Washington Berr... 90748583674 \$2.85 \$2.28 \$0.94 8.39
2	Washington Washington Ma... 96516502499 \$0.74 \$0.59 \$0.26 7.42
3	Washington Washington Stra... 58427771925 \$0.83 \$0.66 \$0.40 13.1 1
4	Washington Washington Cre... 64412155747 \$3.64 \$2.91 \$1.64 10.6 1
5	Washington Washington Diet... 85561191439 \$2.19 \$1.75 \$0.77 6.66 1
6	Washington Washington Cola 29804642796 \$1.15 \$0.92 \$0.37 15.8
7	Washington Washington Diet... 20191444754 \$2.61 \$2.09 \$0.91 18 1
8	Washington Washington Ora... 89770532250 \$2.59 \$2.07 \$0.80 8.97 1



# TEXT

LEN ()	Returns the number of characters in a string
CONCATENAT E()	Joins two text strings into one
LEFT/MI D/ RIGHT()	Returns a number of characters from the start/middle/end of a text string
UPPER/LOWE R/ PROPER()	Converts letters in a string to upper/lower/proper case
SUBSTITUT E()	Replaces an instance of existing text with new text in a string
SEARC H()	Returns the position where a specified string or character is found, reading left to right

=**LEN**(<text>)

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

=**CONCATENATE**(<text1>, <text2>)

=**LEFT/RIGHT**(<text>, <num\_chars>)  
=**MID**(<text>, <start\_num>, <num\_chars>)

=**UPPER/LOWER/PROPER**(<text>)

=**SUBSTITUTE**(<text>, <old\_text>, <new\_text>, <instance>)

=**SEARCH**(<find\_text>, <within\_text>, <start\_num>, <NotFoundValue>)



# TEXT FUNCTIONS

Extract the **left 3 characters** from each value in the **store\_country** column

store_address	store_country	store_phone	area_code	first_opened_date	last_revised_date	total_sqft	geometry_type	supermarket_size	country_address	store_street
1 Hwy Rd, Acap... MEXICO	262-555-5124	262	1/9/1980 12:00:00...	12/5/1990 12:00:00...	23593	17475 Small	MEX	853		
2 Rancho Way... USA	605-555-8203	605	4/2/1970 12:00:00...	6/4/1973 12:00:00...	28206	22271 Medium	USA	6293		
3 Henry Circle... USA	509-555-1596	509	6/14/1994 12:00:00...	11/19/1997 12:00:00...	39996	24390 Large	USA	5043		
4 George Dr, Ca... MEXICO	304-555-1474	304	9/7/1994 12:00:00...	12/1/1995 12:00:00...	23759	16844 Small	MEX	633		
5 Lagoon Drive... MEXICO	804-555-4324	804	7/18/1978 12:00:00...	6/29/1991 12:00:00...	24597	15012 Small	MEX	6290		
6 Russell Canyon... USA	958-555-5002	958	1/3/1981 12:00:00...	6/3/1981 12:00:00...	23688	15375 Small	USA	6495		
7 Hart Drive, L... USA	477-555-7967	477	5/21/1971 12:00:00...	10/20/1981 12:00:00...	23598	14210 Small	USA	6377		
8 Loma Vista Av... MEXICO	797-555-3417	797	9/3/1958 12:00:00...	11/18/1967 12:00:00...	36797	20141 Large	MEX	6178		
9 Pintado Roa... MEXICO	438-555-9528	439	3/8/1955 12:00:00...	6/7/1958 12:00:00...	36509	22450 Huge	MEX	6872		
10 Betheram Dr... MEXICO	213-555-4778	212	4/13/1979 12:00:00...	1/30/1982 12:00:00...	84793	26354 Large	MEX	7994		
11 Goldend Circle... USA	685-555-8995	685	9/17/1976 12:00:00...	5/15/1982 12:00:00...	20319	16232 Small	USA	6371		
12 Winchester Pl... MEXICO	151-555-1702	151	3/25/1960 12:00:00...	12/18/1993 12:00:00...	30584	21938 Large	MEX	6120		
13 Valley Avn, S... USA	977-555-1724	977	4/13/1957 12:00:00...	11/10/1997 12:00:00...	23694	18670 Medium	USA	6379		
14 Hugo Ct, San... USA	135-555-4888	135	11/24/1957 12:00:00...	1/7/1958 12:00:00...	23478	15321 Small	USA	6365		
15 Highland Drive... USA	899-555-1024	893	7/24/1969 12:00:00...	10/19/1973 12:00:00...	21215	13305 Small	USA	6006		
16 Salle Ct, Spo... USA	643-555-3645	643	8/23/1974 12:00:00...	7/13/1977 12:00:00...	30268	23063 Large	USA	6322		
17 Bon Road, Ta... USA	855-555-5588	855	5/30/1970 12:00:00...	6/23/1976 12:00:00...	3120					
18 Bon Road, Hill... MEXICO	528-555-8317	528	6/28/1969 12:00:00...	8/30/1975 12:00:00...	30					
19 Barnes Drive... CANADA	862-555-7395	862	3/23/1977 12:00:00...	10/25/1990 12:00:00...	21					
20 Carpenter Ln, V... CANADA	807-555-1931	897	2/6/1980 12:00:00...	4/9/1987 12:00:00...	31					

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

year	month	day	yearmonth	monthyear	startofyear	endofyear	startofmonth	endofmonth	quarter_number	yearmonthyear
1 2/1/1997 12:00... 1997	3	2 July	2/1/1997 12:00:00...	2/1/1997 12:00:00...	27/6/1996 12:00:00...	2/28/1997 12:00:00...	1 Tuesday	1/1/1997 12:00:00...	1 Q3	199777
2 2/1/1997 12:00... 1997	3	7 July	2/1/1997 12:00:00...	2/1/1997 12:00:00...	27/6/1996 12:00:00...	2/28/1997 12:00:00...	2 Wednesday	1/1/1997 12:00:00...	1 Q3	199777
3 2/1/1997 12:00... 1997	3	12 July	2/1/1997 12:00:00...	2/1/1997 12:00:00...	27/6/1996 12:00:00...	2/28/1997 12:00:00...	3 Thursday	1/1/1997 12:00:00...	1 Q3	199777
4 2/1/1997 12:00... 1997	3	17 July	2/1/1997 12:00:00...	2/1/1997 12:00:00...	27/6/1996 12:00:00...	2/28/1997 12:00:00...	4 Friday	1/1/1997 12:00:00...	1 Q3	199777
5 2/1/1997 12:00... 1997	3	22 July	2/1/1997 12:00:00...	2/1/1997 12:00:00...	27/6/1996 12:00:00...	2/28/1997 12:00:00...	5 Saturday	1/1/1997 12:00:00...	1 Q3	199777
6 2/1/1997 12:00... 1997	3	27 July	2/1/1997 12:00:00...	2/1/1997 12:00:00...	28/6/1996 12:00:00...	2/28/1997 12:00:00...	6 Sunday	1/1/1997 12:00:00...	1 Q3	199777
7 2/1/1997 12:00... 1997	3	1 August	2/1/1997 12:00:00...	2/1/1997 12:00:00...	28/6/1996 12:00:00...	2/28/1997 12:00:00...	7 Monday	1/1/1997 12:00:00...	1 Q3	199777
8 2/1/1997 12:00... 1997	3	6 August	2/1/1997 12:00:00...	2/1/1997 12:00:00...	28/6/1996 12:00:00...	2/28/1997 12:00:00...	8 Tuesday	1/1/1997 12:00:00...	1 Q3	199777

Extract characters from the left of the **customer\_address** column, up to the space

customer_id	first_name	last_name	address	city	state	zip	phone	email	customer_household_size	customer_address
1 11/16/1994 12:00... Bronze	Bertha Jam...	1948	Y				69	Non-Grad	Other	8029
2 5/5/1992 12:00:00... Bronze	Manual	N	Ole Weldon	1931			86	Non-Grad	Other	5754
3 6/26/1994 12:00:00... Bronze	Manual	N	Paul Alcorn	1973			44	Non-Grad	Other	4822
4 2/9/1990 12:00:00... Bronze	Manual	N	Jared Busta...	1910			107	Non-Grad	Other	4222
5 3/15/1992 12:00:00... Bronze	Manual	N	Margaret A...	1979			38	Non-Grad	Other	8452
6 3/2/1994 12:00:00... Bronze	Manual	N	Vanessa Ten...	1930			87	Non-Grad	Other	6621
7 6/4/1993 12:00:00... Bronze	Manual	N	Catherine ...	1966			51	Non-Grad	Other	1239
8 3/5/1992 12:00:00... Bronze	Manual	N	Stacey Cere...	1943			74	Non-Grad	Other	852
9 5/17/1992 12:00:00... Bronze	Manual	N	Marlin Correll	1933			84	Non-Grad	Other	4824
10 9/8/1992 12:00:00... Bronze	Manual	N	Deanna Sab...	1916			101	Non-Grad	Other	8942
11 2/21/1991 12:00:00... Bronze	Manual	N	Joseph Tho...	1968			49	Non-Grad	Other	2099
12 6/12/1994 12:00:00... Bronze	Manual	N	Roberta Shu...	1919			98	Non-Grad	Other	8086

# 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:  
• SUM(Transactions[quantity])

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

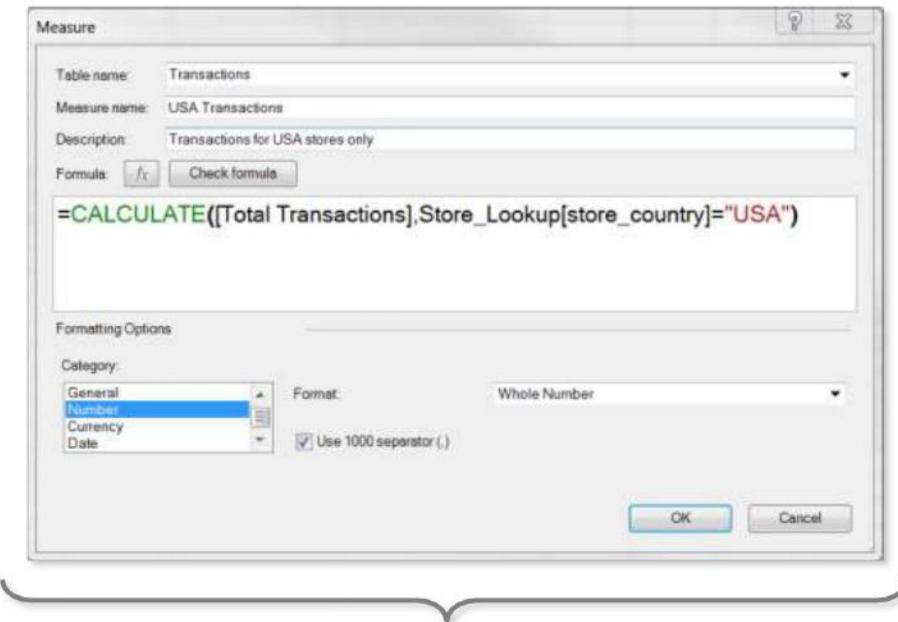
Examples:  
• Store\_Lookup[store\_country] = "USA"  
• Calendar[Year] = 1998  
• Transactions[quantity] >= 5

### 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”

# 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
<b>Grand Total</b>	<b>269,720</b>	<b>180,823</b>

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?

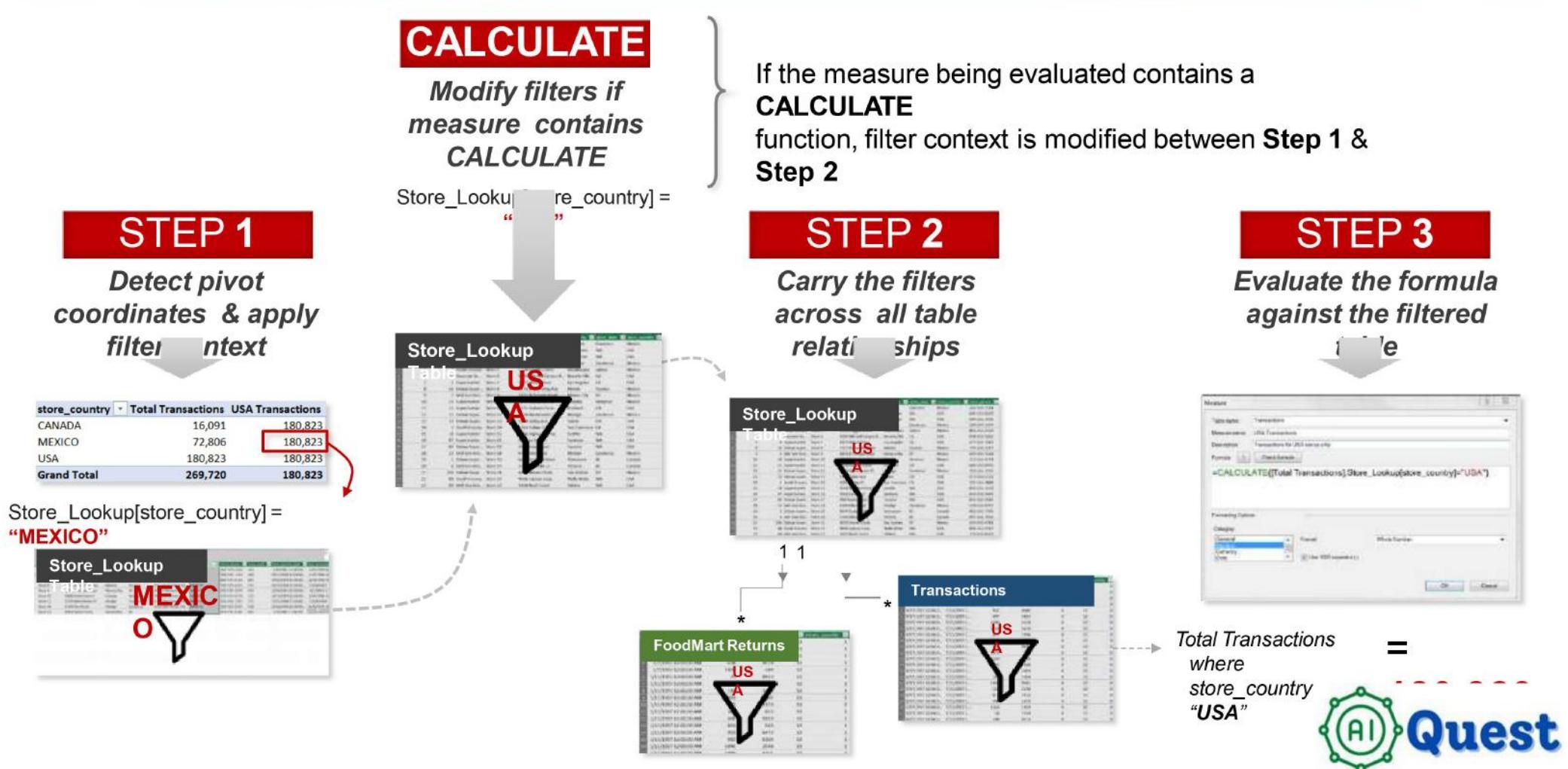
## HEY THIS IS IMPORTANT!

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 **store\_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

# CALCULATE CHANGES THE FILTER



# FILT

## FILTE

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

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

Table to be filtered  
Example

- s:
- Store\_Lookup
- Product\_Lookup

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

Examples:  
• Store\_Lookup[store\_country]=“US”  
• 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!**

# PRO TIP: FILTERING WITH DISCONNECTED

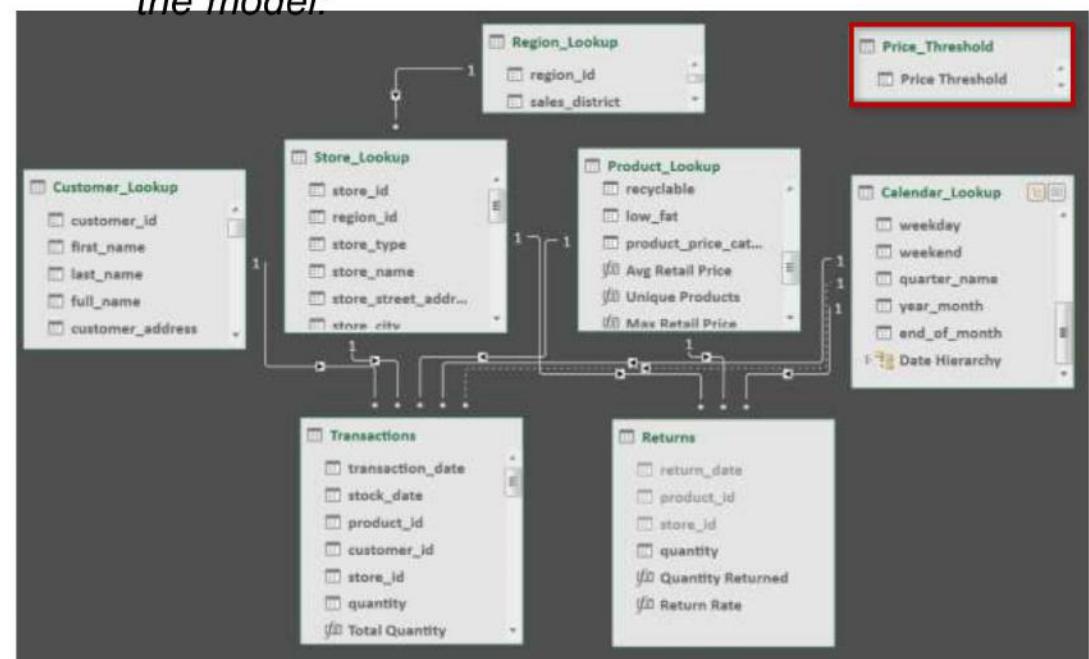
**STEP 1:** Create an Excel table containing a list of values to use as thresholds or parameters:

A	Price Threshold
1	1
2	2
3	3
4	4

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

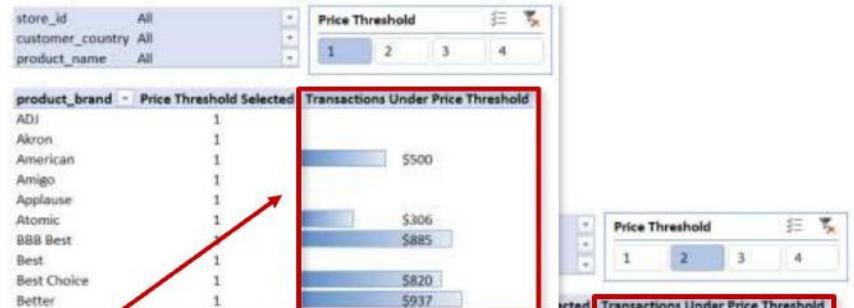


**STEP 3:** Make sure that your table loaded, and is NOT connected to any other table in the model:

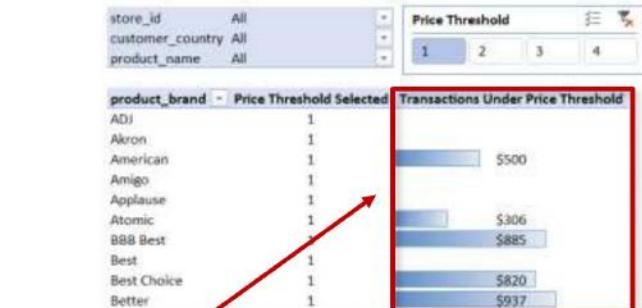
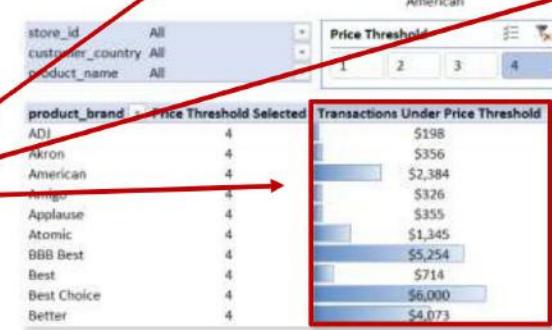
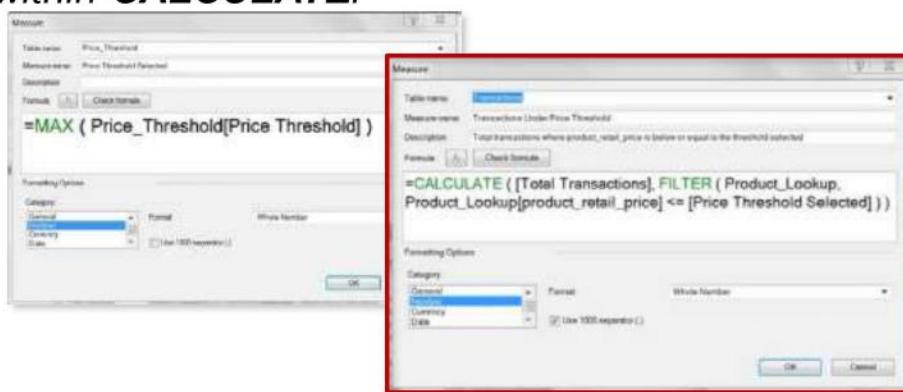


# PRO TIP: FILTERING WITH DISCONNECTED

**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**

# FILTER

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

**Measure**

Table name: **Transactions**

Measure name: **Transactions Under Price Threshold**

Description: Total transactions where product\_retail\_price is below or equal to the threshold selected

Formula: **=CALCULATE ( [Total Transactions], FILTER ( Product\_Lookup, Product\_Lookup[product\_retail\_price] <= [Price Threshold Selected] ) )**

Formatting Options

Category: **Number**

Format: **Whole Number**

Use 1000 separator (.)

**OK** **Cancel**

*Calculate Total Revenue, but only for USA stores*

**Measure**

Table name: **Transactions**

Measure name: **USA Revenue**

Description:

Formula: **=SUMX (**  
**FILTER ( Store\_Lookup, Store\_Lookup[store\_country] = "USA" ), [Total Revenue] )**

Formatting Options

Category: **Currency**

Symbol: **\$**

Decimal places: **0**

Use 1000 separator (.)

**OK** **Cancel**



# 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:

- 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

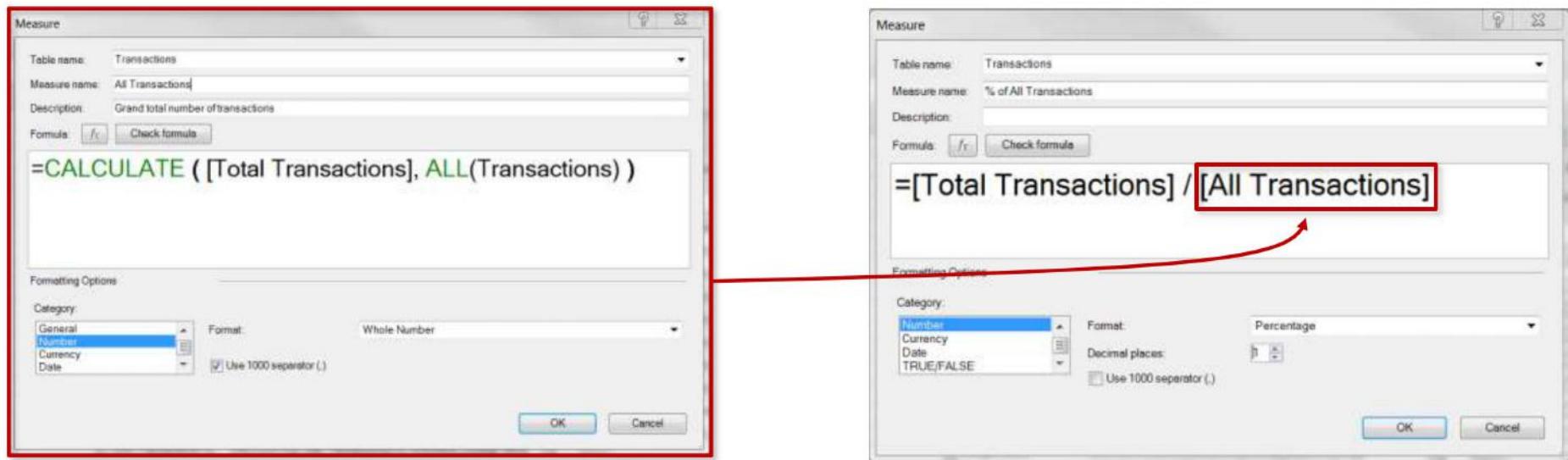
- Example:** All columns must include the table name, and come from the same table
- Customer\_Lookup[customer\_city], Customer\_Lookup[customer\_country]
  - Product\_Lookup[product\_name]

### 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)

# 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

# RELAT

## RELATE

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

=RELATED(<column  
n>)



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

# RELATED

Retrieve the *retail price* from the *Product\_Lookup* table and append it to the *Transactions* table

	Transaction_date	stock	produ	custom	sto	quantity	price	Add Column
1	7/27/1997 12:00:00 AM	7/25/1997 1...	721	1254	13	3	2.26	
2	7/27/1997 12:00:00 AM	7/25/1997 1...	334	4303	13	3	2.89	
3	7/27/1997 12:00:00 AM	7/25/1997 1...	832	545	13	3	3.24	
4	7/27/1997 12:00:00 AM	7/25/1997 1...	690	5680	13	3	1.44	
5	7/27/1997 12:00:00 AM	7/25/1997 1...	952	5680	13	3	2.91	
6	7/27/1997 12:00:00 AM	7/25/1997 1...	497	5625	13	3	1.87	
7	7/27/1997 12:00:00 AM	7/25/1997 1...	1209	5610	13	3	1.31	
8	7/27/1997 12:00:00 AM	7/25/1997 1...	1156	5610	13	3	1.82	
9	7/27/1997 12:00:00 AM	7/25/1997 1...	427	7476	13	3	2.14	
10	7/27/1997 12:00:00 AM	7/25/1997 1...	216	950	13	3	1.87	
11	7/27/1997 12:00:00 AM	7/25/1997 1...	464	5495	13	3	3.56	
12	7/27/1997 12:00:00 AM	7/25/1997 1...	1274	790	13	3	2.89	
13	7/27/1997 12:00:00 AM	7/25/1997 1...	1550	9280	13	3	1.81	
14	7/27/1997 12:00:00 AM	7/25/1997 1...	705	1468	13	3	1.92	
15	7/27/1997 12:00:00 AM	7/25/1997 1...	768	1054	13	3	1.92	
16	7/27/1997 12:00:00 AM	7/25/1997 1...	106	1954	13	3	2.96	
17	7/27/1997 12:00:00 AM	7/25/1997 1...	1435	9511	13	3	2.72	
18	7/27/1997 12:00:00 AM	7/25/1997 1...	623	3548	13	3	1.8	
19	7/27/1997 12:00:00 AM	7/25/1997 1...	912	7436	13	3	3.34	
20	7/27/1997 12:00:00 AM	7/25/1997 1...	294	1429	13	3	1.32	

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

Measure

Table name: Transactions

Measure name: Total Revenue

Description:

Formula:

=SUMX ( Transactions,  
Transactions[quantity] \*  
RELATED ( Product\_Lookup[product\_retail\_price] ) )

Formatting Options

Category:  Currency

Symbol: \$

Decimal places: 0

Use 1000 separator (,)

OK Cancel

# ITERATOR (“X”)

**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.)

Aggregation to apply to calculated rows\*

Examples:

- COUNTX
- AVERAGE
- X
- RANKX
- MAXX/MIN

X



=**SUMX**(*table*, <expression>)  
Table in which the expression will be evaluated

Examples:

- Transactions
- FILTER(Transactions,  
RELATED(Store\_Lookup[country])="SA")

<expression>  
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

# ITERATOR (“X”) FUNCTIONS

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

Measure

Table name: Transactions  
Measure name: Total Revenue (Measure)  
Description:  
Formula: `=SUMX( Transactions, Transactions[quantity] * Transactions[retail_price] )`

Formatting Options

Category: **Currency**      Symbol: \$      Decimal places: 0  
 Use 1000 separator (.)

OK Cancel

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

Measure

Table name: Transactions  
Measure name: Product Brand Rank (By Revenue)  
Description:  
Formula: `=RANKX( ALL( Product_Lookup[product_brand] ), [Total Revenue] )`

Formatting Options

Category: **Number**      Format: Whole Number  
 Use 1000 separator (.)

OK Cancel



# BASIC DATE & TIME

**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>)**

**DATEDIF E()**

Returns the difference between two dates, based on a selected interval

=**DATEDIFF(<start\_date>, <end\_date>, <interval>)**



# BASIC DATE & TIME FUNCTIONS

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

	acct_open_date	member_card	occupation	homeowner	full_name	birth_year	has_children	customer_age	education_level
1	11/16/1994 12:00:00	Bronze	Manual	N	Bertha Jam...	1948	Y	69	Non-Grad
2	5/5/1992 12:00:00	Bronze	Manual	N	Ole Weldon	1931	Y	86	Non-Grad
3	6/26/1994 12:00:00	Bronze	Manual	N	Paul Alcorn	1973	Y	44	Non-Grad
4	2/9/1990 12:00:00	Bronze	Manual	N	Jared Busta...	1910	Y	107	Non-Grad
5	3/15/1992 12:00:00	Bronze	Manual	N	Margaret A...	1979	Y	38	Non-Grad
6	3/2/1994 12:00:00	Bronze	Manual	N	Vanessa Ten...	1930	Y	87	Non-Grad
7	6/4/1993 12:00:00	Bronze	Manual	N	Catherine ...	1966	Y	51	Non-Grad
8	3/5/1992 12:00:00	Bronze	Manual	N				79	Non-Grad
9	5/17/1992 12:00:00	Bronze	Manual	N				69	Non-Grad
10	9/8/1992 12:00:00	Bronze	Manual	N				86	Non-Grad
11	2/21/1991 12:00:00	Bronze	Manual	N				44	Non-Grad
12	6/12/1994 12:00:00	Bronze	Manual	N				107	Non-Grad
13	4/24/1992 12:00:00	Bronze	Manual	N				38	Non-Grad
14	11/8/1991 12:00:00	Bronze	Manual	N				87	Non-Grad
15	6/16/1992 12:00:00	Bronze	Manual	N				51	Non-Grad
16	3/8/1993 12:00:00	Bronze	Manual	N				79	Non-Grad
17	6/19/1994 12:00:00	Bronze	Manual	N				69	Non-Grad
18	8/27/1993 12:00:00	Bronze	Manual	N				86	Non-Grad
19	3/26/1991 12:00:00	Bronze	Manual	N				44	Non-Grad
20	3/28/1994 12:00:00	Bronze	Manual	N				107	Non-Grad

Calculate the end date of the month, for each row in the Calendar\_Lookup table

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

# TIME INTELLIGENCE

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

Performance To-

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

Use DATESQTD for Quarters or DATESMTD for Months

Previous

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

Running

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

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



## PRO TIP:

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

# SPEED & PERFORMANCE



## 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



## 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.)



# DAX BEST

★ **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 (preceded by the table name) make formulas more readable and intuitive and differentiates them from*



# WRAPPING

HR



# DATA VISUALIZATION

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**



# SNEAK PEEK:



**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](http://powerbi.microsoft.com)

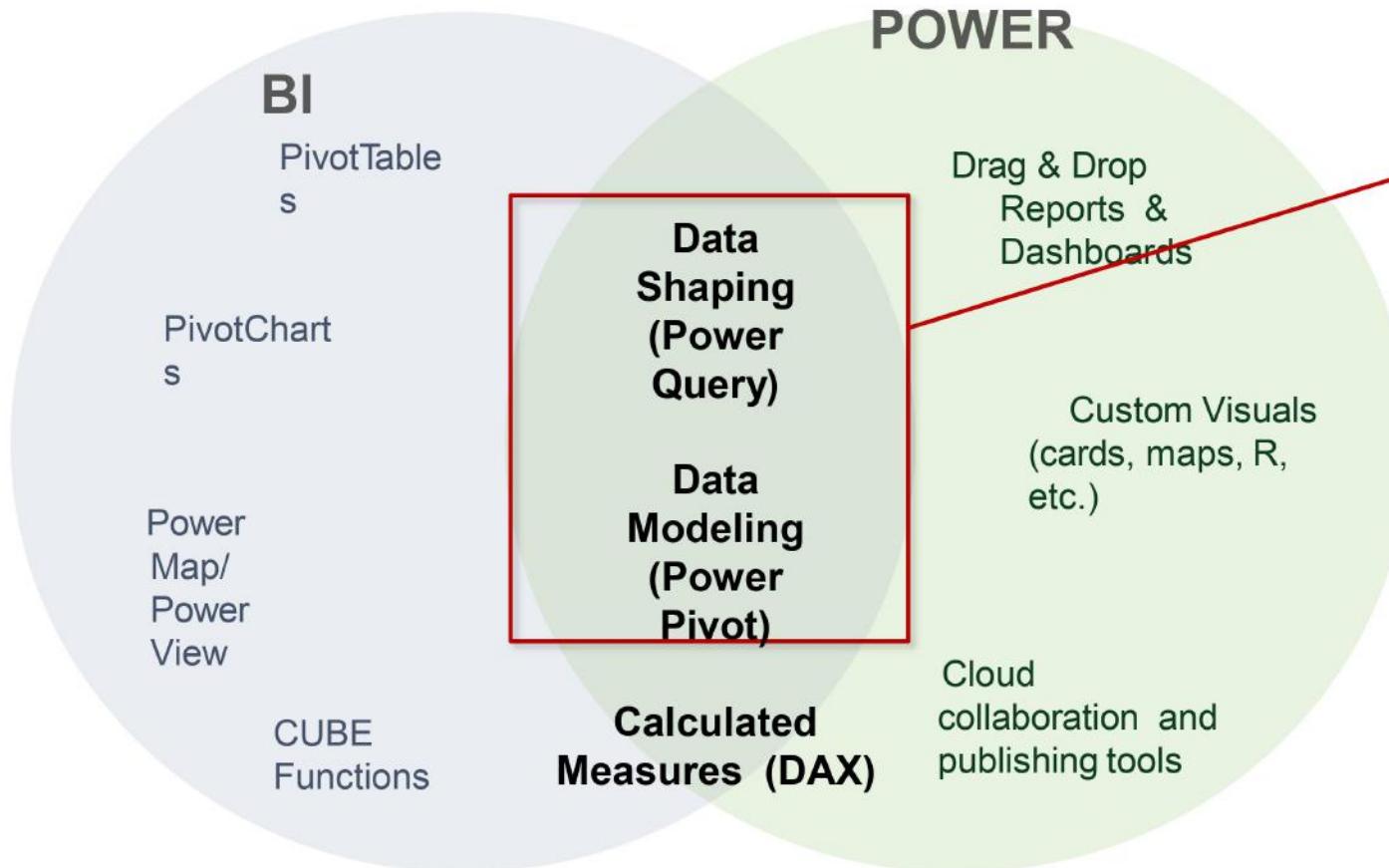


Figure 1. Magic Quadrant for Business Intelligence and Analytics Platforms



# “POWER EXCEL” VS.

## “POWER EXCEL”



“Power Excel”  
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!**





**THANK  
YOU!!**

70%

100%