Data Modeling

م. مينا مراد

تحليل البيانات باستخدام أدوات ذكاء الأعمال

لمايكروسوفت اكسيل ج2

نبذة عن الكورس وماذا سنتعلم في الكورس

- في الجزء الثاني من كورس تحليل البيانات سنتعمق أكثر وأكثر في أهم الأدوات الخاصة بتحليل البيانات
- سنتعلم إدارة البيانات باحتراف باستخدام الـ Power Query
- سنتعلم ماذا تعني Data modeling وكيف نستطيع دمج المعلومات وننشئ ما بينها الـ Relationships الصحيحة
- سنتعلم كتابة معادلات الـ DAX بكل أنواعها، وكيفية صياغة ما تحتاج معرفته باستخدام المعادلة المناسبة
- سنتعلم أيضاً استخدام أداة الـ KPI وكيفية قياس الأداء العام

تحليل البيانات باستخدام أدوات ذكاء الأعمال

لمايكروسوفت اكسيل ج2

الفصل الأول: معالجة البيانات وتجهيزها

- ماذا تحتاج لاحتراف تحليل الأعمال؟
- ماذا بعد الجزء الأول من الكورس؟
- إعداد الـ Power Query
- الداتا مودلينج Data modeling
- إنشاء العلاقات المختلفة بين جداول البيانات
- أنواع العلاقات بين جداول البيانات

تحليل البيانات باستخدام أدوات ذكاء الأعمال
لمايكروسوفت اكسيل ج2

الفصل الثاني: الوصول إلى أدق الإجابات لجميع الاستعلامات
باستخدام أدوات الـ PowerPivot and DAX

- Pivot table Vs. PowerPivot
- Calculated Columns Vs. Measures
- استخدام الـ Data Analysis Expressions بالتفصيل

تحليل البيانات باستخدام أدوات ذكاء الأعمال

لمايكروسوفت اكسيل ج2

الفصل الثاني: الوصول إلى أدق الإجابات لجميع الاستعلامات

باستخدام أدوات الـ PowerPivot and DAX

- تحليل المبيعات والكميات وأداء المحافظات والفروع باستخدام:
  - Math Functions المعادلات الحسابية
  - Logical Functions المعادلات المنطقية
  - Text functions للتعامل مع النصوص
- التحليل الزمني للمبيعات والأداء باستخدام Date functions
- كيفية إنشاء مفاتيح قياس الأداء لجميع تفاصيل المشروع

تحليل البيانات باستخدام أدوات ذكاء الأعمال

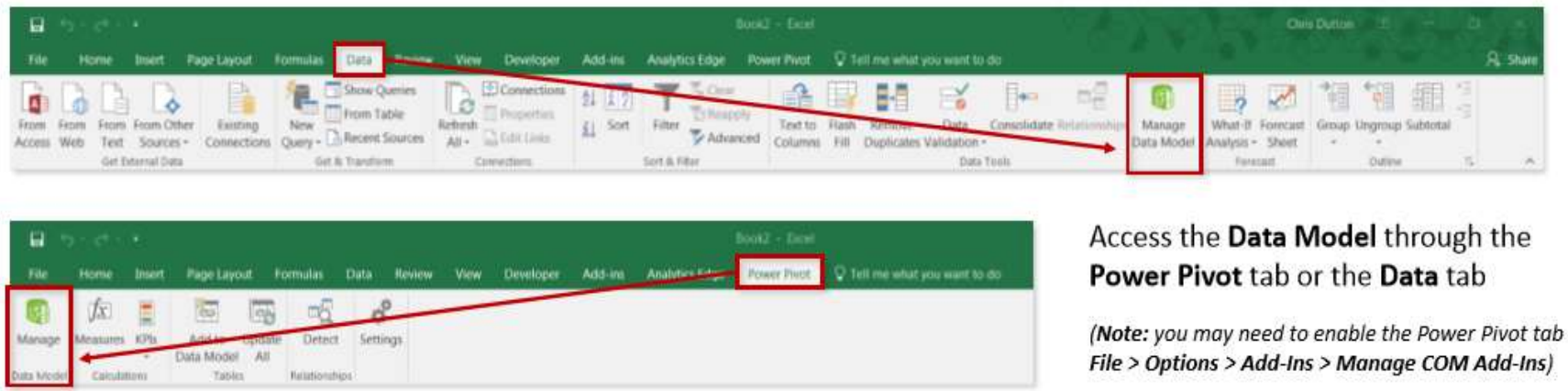
لمايكروسوفت اكسيل ج2

من الذي يستفيد من الكورس؟

- هذا الكورس لكل صاحب أو مدير مشروع أو بيزنس يريد امتلاك مهارة تحليل البيانات، والتي عن طريقها سيستطيع اتخاذ القرار السليم.
- هذا الكورس لأي خريج يرغب في العمل في مجال تحليل البيانات كواحد من المجالات الجديدة في مصر، والتي لها مستقبل عظيم.

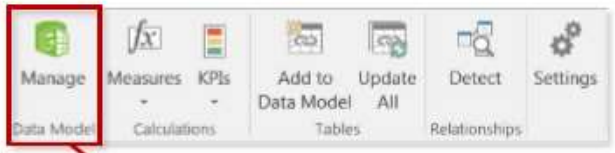
The **Data Model** provides simple and intuitive tools for building relational databases directly in Excel. With the data model you can:

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



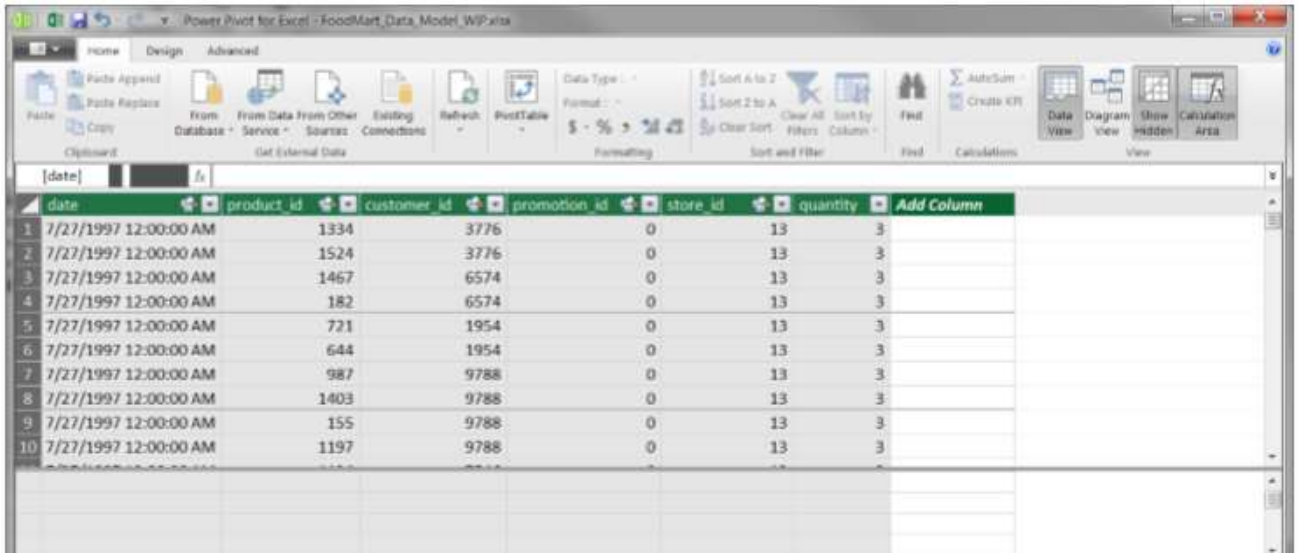
Access the **Data Model** through the **Power Pivot** tab or the **Data** tab

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



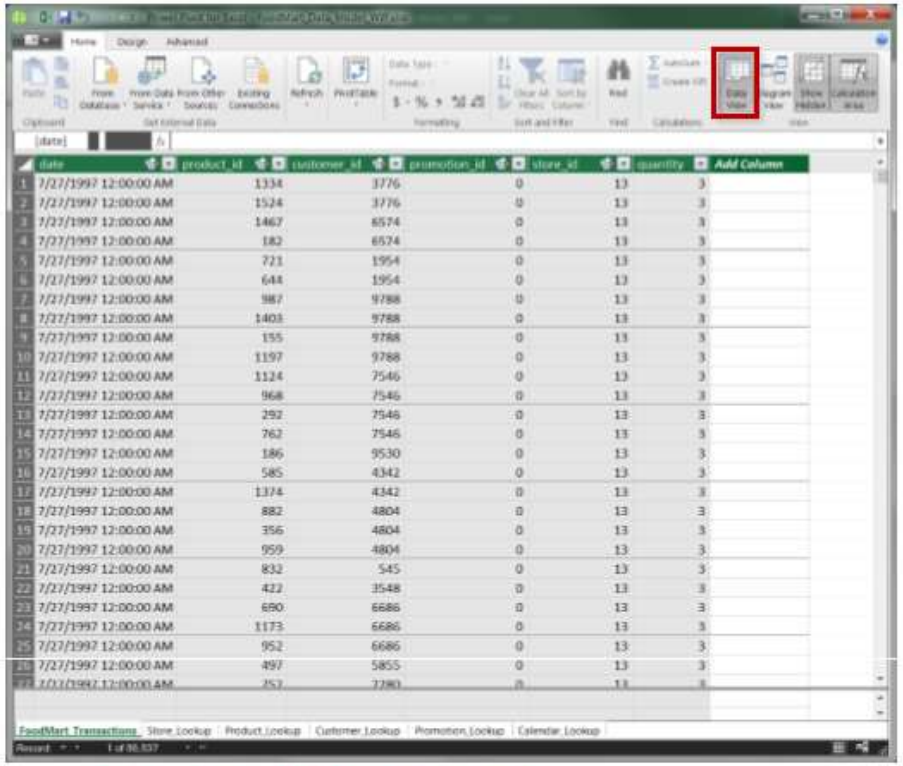
The **Data Model** opens in a separate Excel window, where you can view your data tables, calculate new measures, and define table relationships

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



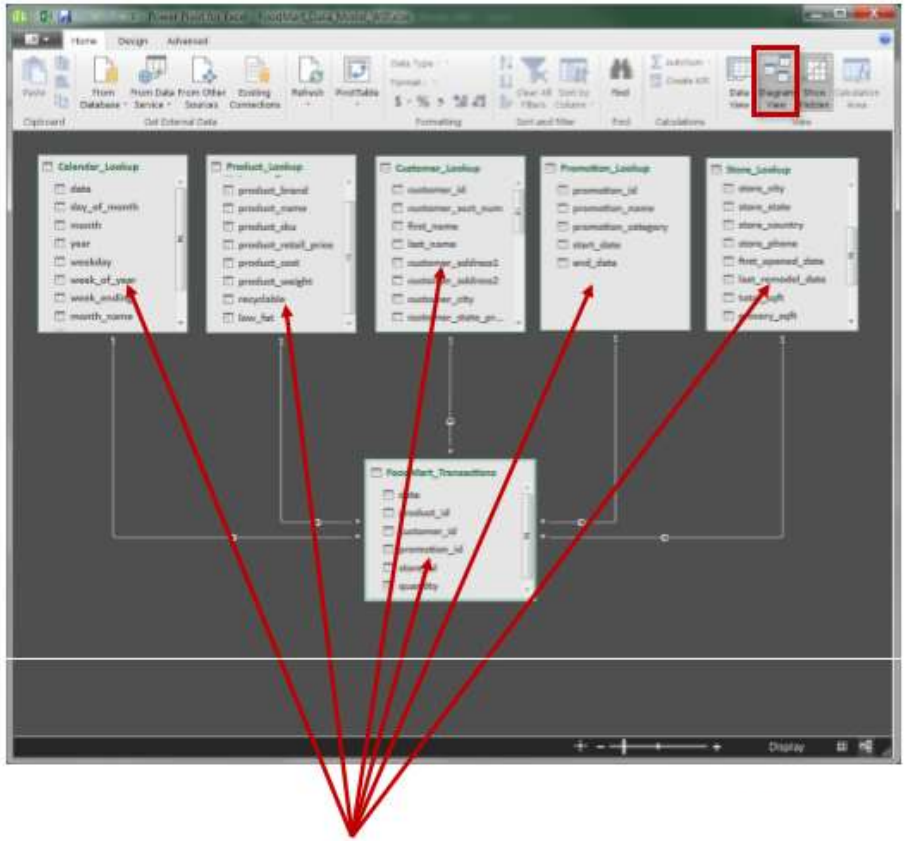
	date	product_id	customer_id	promotion_id	store_id	quantity	Add Column
1	7/27/1997 12:00:00 AM	1334	3776	0	13	3	
2	7/27/1997 12:00:00 AM	1524	3776	0	13	3	
3	7/27/1997 12:00:00 AM	1467	6574	0	13	3	
4	7/27/1997 12:00:00 AM	182	6574	0	13	3	
5	7/27/1997 12:00:00 AM	721	1954	0	13	3	
6	7/27/1997 12:00:00 AM	644	1954	0	13	3	
7	7/27/1997 12:00:00 AM	987	9788	0	13	3	
8	7/27/1997 12:00:00 AM	1403	9788	0	13	3	
9	7/27/1997 12:00:00 AM	155	9788	0	13	3	
10	7/27/1997 12:00:00 AM	1197	9788	0	13	3	

DATA VIEW



*Tables organized in **tabs***

DIAGRAM VIEW



*Tables organized as **objects***

Normalization

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

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

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

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

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

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

Data Table Vs. Lookup Table

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 "**product_id**" columns*

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

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

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

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

Customer Lookup

	customer_id	customer_acct_num	first_name	last_name	customer_address
1	213	89905924797	David	Watson	8157 W. Buchanan
2	227	90065194368	Michael	Suggs	40 Ellis St.
3	246	90350758433	Rita	Santry	207 Barquentine Court
4	255	90442248582	Jacky	Camille	3919 El Pintado Road
5	258	90463679432	John	Minker	1061 Buskrik Avenue
6	273	90642099976	Jennifer	Confetti	2775 Robinson Ave.
7	289	90872421300	Bertha	Jameson	3029 Heather Leaf Ln.
8	294	90917686200	Michelle	Adams	9558 Orchard View A...
9	311	91123164892	Janice	Vrins	1055 Horseshoe Road
10	313	91158838272	Ole	Weldon	5754 Glenhaven Ave
11	331	91323767304	Lanna	Slaven	5118 Boxwood Dr.
12	390	92005987200	Paul	Alcorn	4822 Center Street
13	394	92050951719	Jared	Bustamante	4222 San Jose Dr.

Store Lookup

	store_id	region_id	store_type	store_name	store_street_address
1	1	28	Supermarket	Store 1	2853 Bailey Rd
2	2	78	Small Grocery	Store 2	5203 Catanzaro Way
3	3	76	Supermarket	Store 3	1501 Ramsey Circle
4	4	27	Gourmet Sup...	Store 4	433 St George Dr
5	5	4	Small Grocery	Store 5	1250 Coggins Drive
6	6	47	Gourmet Sup...	Store 6	5495 Mitchell Canyon R...
7	7	3	Supermarket	Store 7	1077 Wharf Drive
8	8	26	Deluxe Supe...	Store 8	3173 Buena Vista Ave
9	9	2	Mid-Size Gro...	Store 9	1872 El Pintado Road
10	10	24	Supermarket	Store 10	7894 Rotherham Dr
11	11	22	Supermarket	Store 11	5371 Holland Circle

Transactions Data Table

	transaction_date	stock_date	product_id	customer_id	store_id	quantity
1	10/23/1998 12:00:0...	10/20/1998 1...	14	1814	19	3
2	10/23/1998 12:00:0...	10/19/1998 1...	46	5326	19	3
3	10/23/1998 12:00:0...	10/22/1998 1...	67	8236	19	3
4	10/23/1998 12:00:0...	10/20/1998 1...	73	10121	19	3
5	10/23/1998 12:00:0...	10/17/1998 1...	77	8236	19	3
6	10/23/1998 12:00:0...	10/17/1998 1...	84	10121	19	3
7	10/23/1998 12:00:0	10/21/1998 1	86	6770	19	3

Primary Vs. Foreign Keys

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

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

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

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

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

Primary key

	customer_id	customer_acct_num	first_name
1	213	89905924797	David
2	227	90065194368	Michael
3	246	90350758433	Rita
4	255	90442248582	Jacky
5	258	90463679432	John
6	273	90642099976	Jennifer
7	289	90872421300	Bertha
8	294	90917686200	Michelle
9	311	91123164892	Janice
10	313	91158838272	Ole
11	331	91323767304	Lanna
12	390	92005987200	Paul
13	394	92050951719	Jared

Primary key

	product_id	Product_Name	Therapy_area
1	1	Rani	Gastrointestinal...
2	2	Depovit B12	Vitamins & Sup...
3	3	Gratisovir	Respiratory
4	4	Dexamethasone	Hormones and ...
5	5	Oplex	Hormones and ...
6	6	Cefaxone	Anti-infectives
7	7	Spasmodigestin	Gastrointestinal...
8	8	Ketolac	Gynaecology

Foreign keys

	transaction_date	stock_date	product_id	customer_id	store_id	quantity
1	10/23/1998 12:00:0...	10/20/1998 1...	14	1814	19	3
2	10/23/1998 12:00:0...	10/19/1998 1...	46	5326	19	3
3	10/23/1998 12:00:0...	10/22/1998 1...	67	8236	19	3
4	10/23/1998 12:00:0...	10/20/1998 1...	73	10121	19	3
5	10/23/1998 12:00:0...	10/17/1998 1...	77	8236	19	3
6	10/23/1998 12:00:0...	10/17/1998 1...	84	10121	19	3
7	10/23/1998 12:00:0	10/21/1998 1	86	6770	19	3

Primary key

	store_id	region
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	

Merge Vs. Relationships

*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

Relationship types

Emp_id	Emp_name
1	Samy
2	Hany
3	Hosam
4	Ahmed

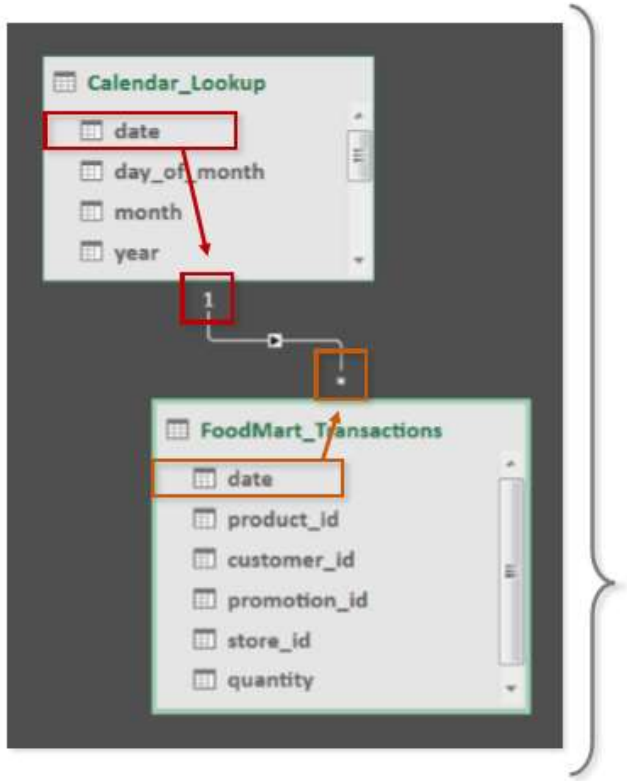
One to One Relationship

1 ⟷ 1

Emp_id	Salary
1	5500
2	4500
3	3455
4	7500

Emp_id	Emp_name	salary
1	Samy	5500
2	Hany	4500
3	Hosam	3455
4	Ahmed	7500

One to many relationships



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)

Primary key

	customer_id	customer_acct_num	first_name
1	213	89905924797	David
2	227	90065194368	Michael
3	246	90350758433	Rita
4	255	90442248582	Jacky
5	258	90463679432	John
6	273	90642099976	Jennifer
7	289	90872421300	Bertha
8	294	90917686200	Michelle
9	311	91123164892	Janice
10	313	91158838272	Ole
11	331	91323767304	Lanna
12	390	92005987200	Paul
13	394	92050951719	Jared

Primary key One to Many Relationship

	product_id	Product_Name	Therapy_area
1	1	Rani	Gastrointestinal...
2	2	Depovit B12	Vitamins & Sup...
3	3	Gratisovir	Respiratory
4	4	Dexamethasone	Hormones and ...
5	5	Oplex	Hormones and ...
6	6	Cefaxone	Anti-infectives
7	7	Spasmodigestin	Gastrointestinal...
8	8	Ketolac	Gynaecology

Foreign keys

	transaction_date	stock_date	product_id	customer_id	store_id	quantity
1	10/23/1998 12:00:0...	10/20/1998 1...	14	1814	19	3
2	10/23/1998 12:00:0...	10/19/1998 1...	46	5326	19	3
3	10/23/1998 12:00:0...	10/22/1998 1...	67	8236	19	3
4	10/23/1998 12:00:0...	10/20/1998 1...	73	10121	19	3
5	10/23/1998 12:00:0...	10/17/1998 1...	77	8236	19	3
6	10/23/1998 12:00:0...	10/17/1998 1...	84	10121	19	3
7	10/23/1998 12:00:0	10/21/1998 1	86	6770	19	3

Primary key

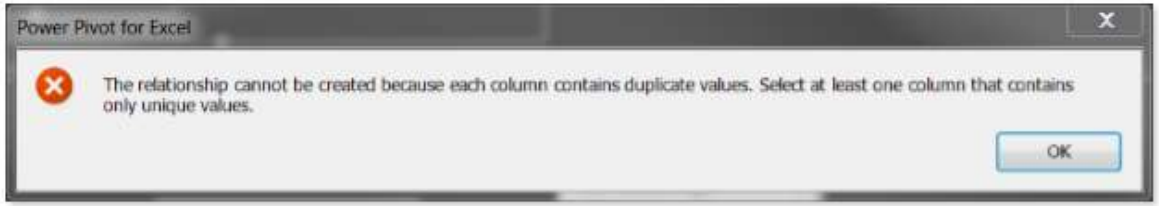
	store_id	region
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	

Project file

Many to many relationship

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

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

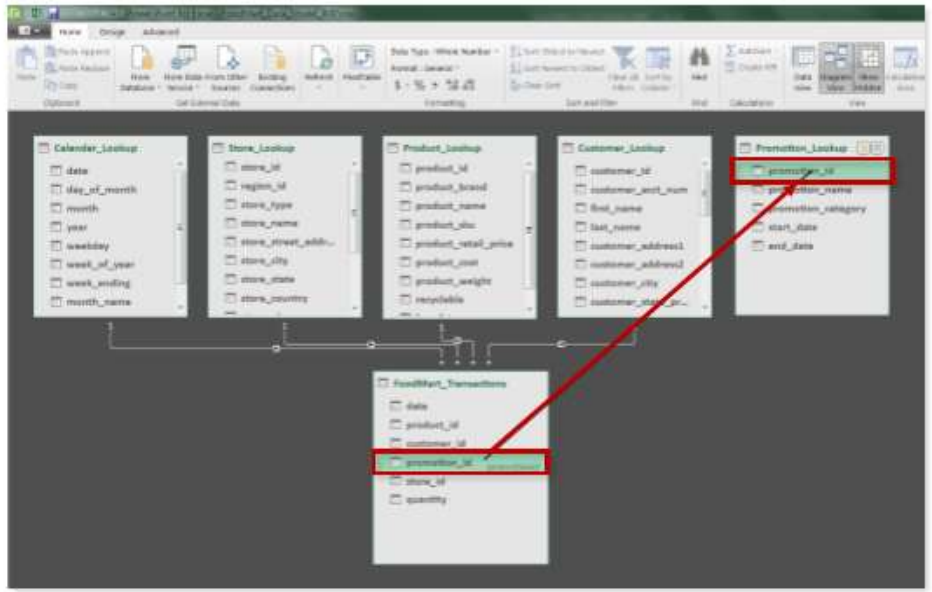


- If we try to connect these tables using the **product_id** field, we'll have a **many-to-many** relationship since there are multiple instances of each ID in both tables

Project file

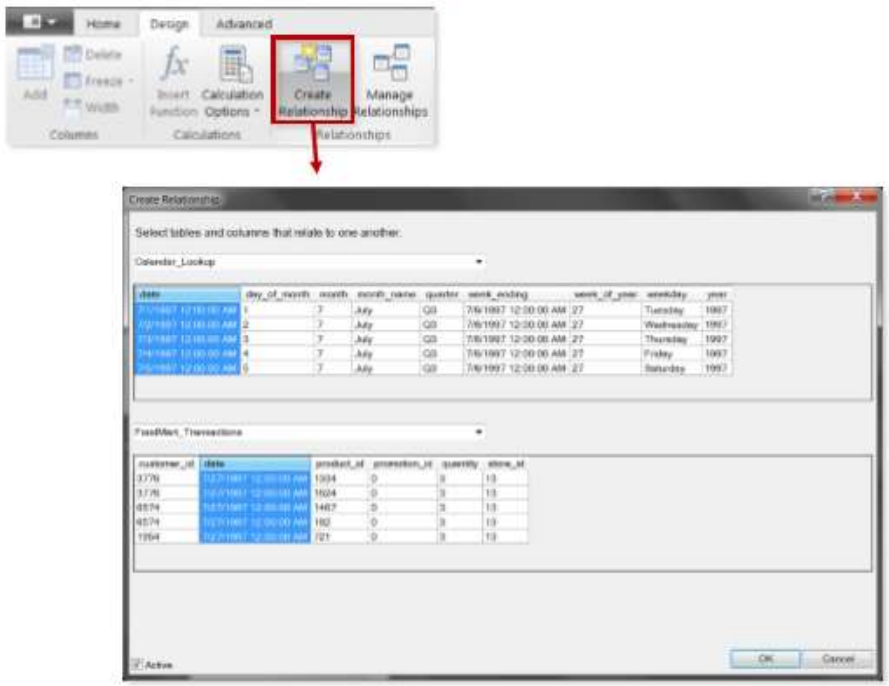
Create Relationships

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

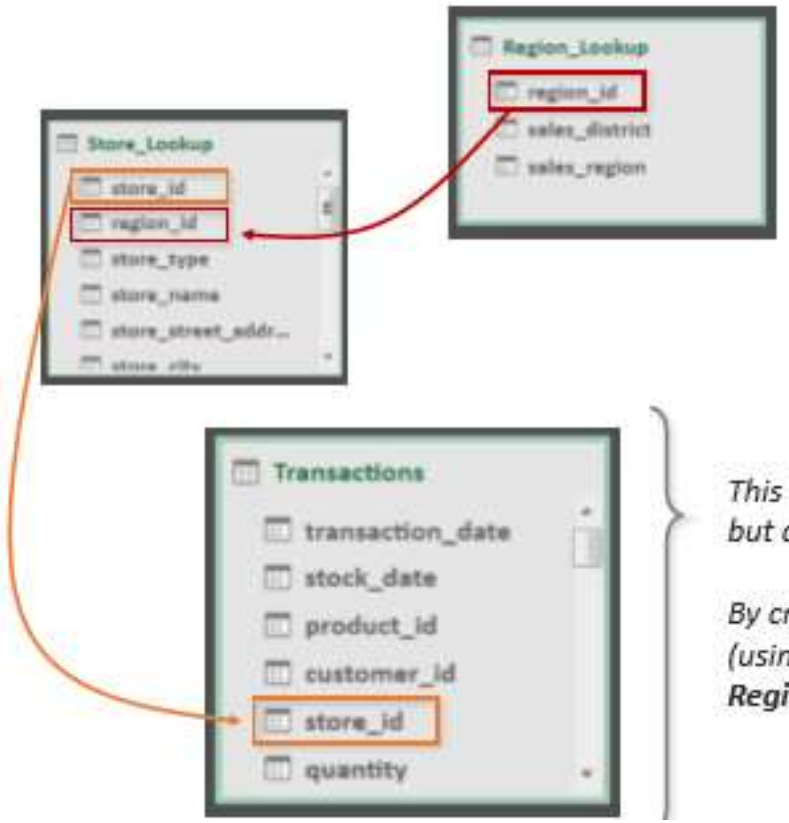


Tip: *Always drag relationships from the* ***Data*** *table to the* ***Lookup*** *tables*

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



Connect Lookup to Lookup Tables



PRO TIP:

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

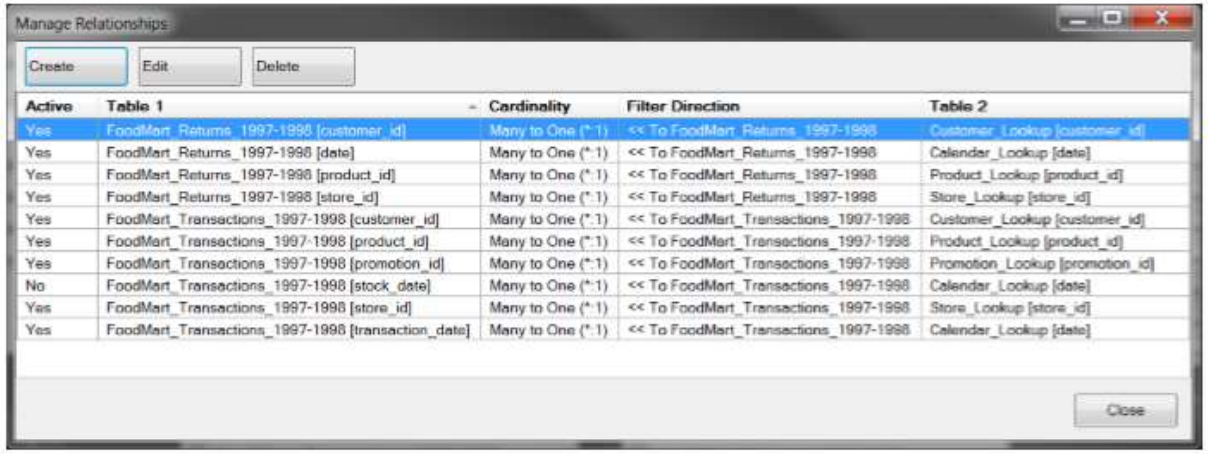
*Models with a single table for each lookup or dimension are called "**star**" schemas*

*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*

Project file

Modifying Relationships

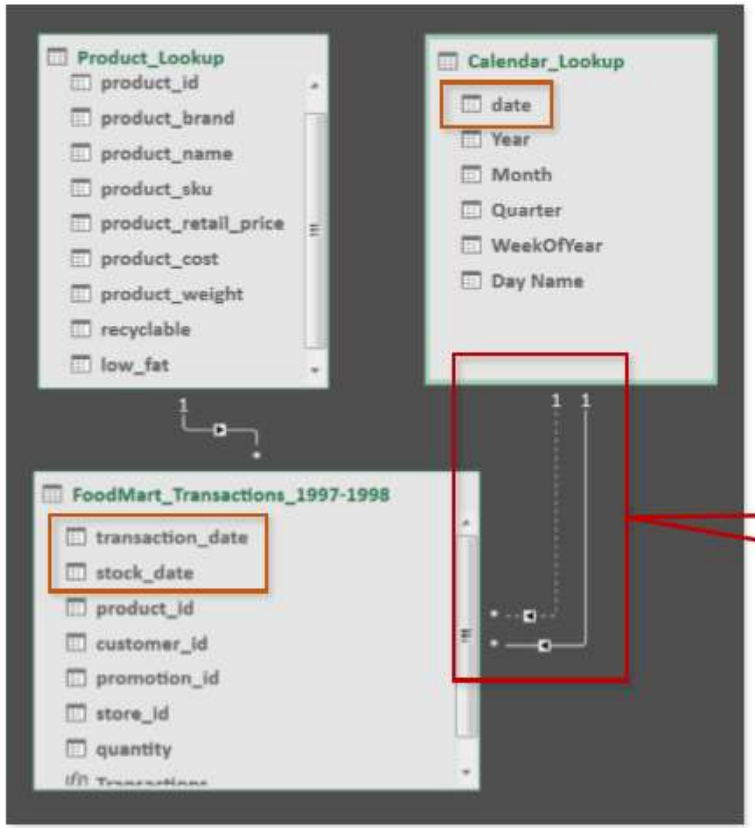


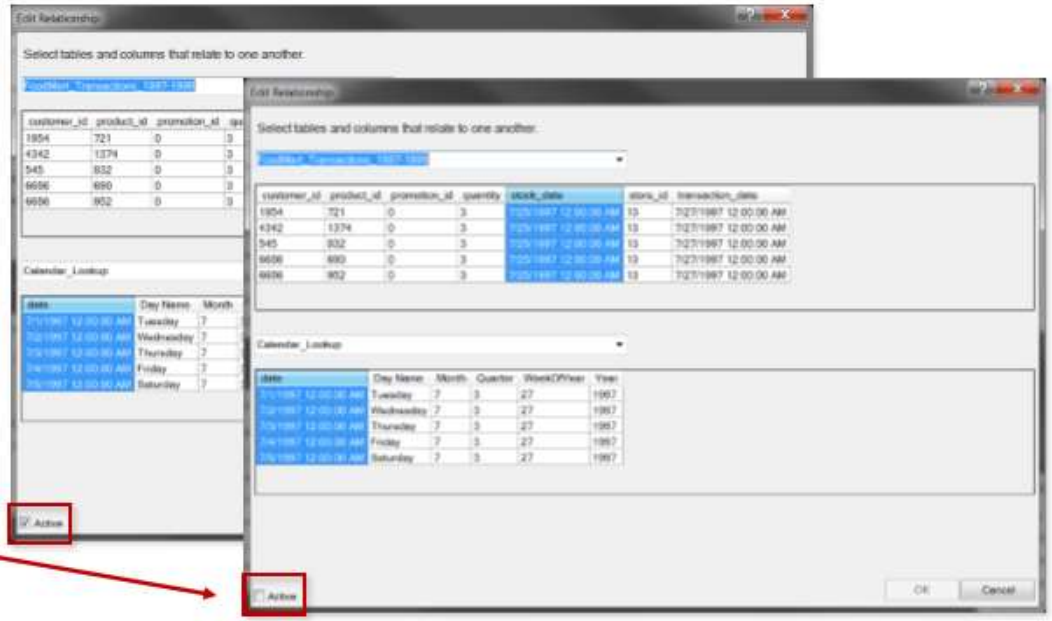
The **Manage Relationships** window allows you to create, edit or delete any connection in the data model

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

Active and inactive relationships

Project file



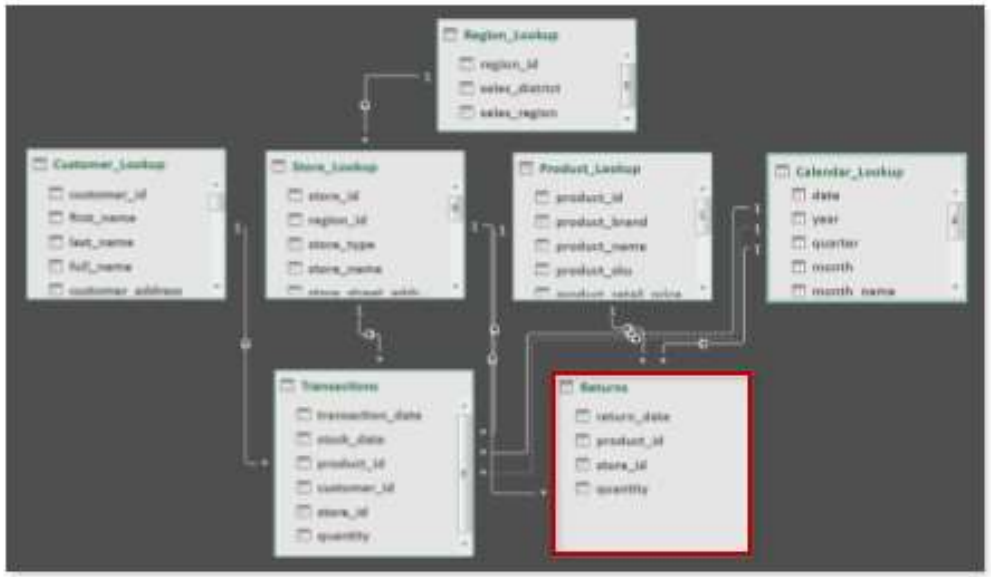


We can connect the **Calendar_Lookup** and **FoodMart_Transactions** tables on both **transaction_date** and **stock_date**; however, only one can be active at a time

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

Connecting multi data table

Project file



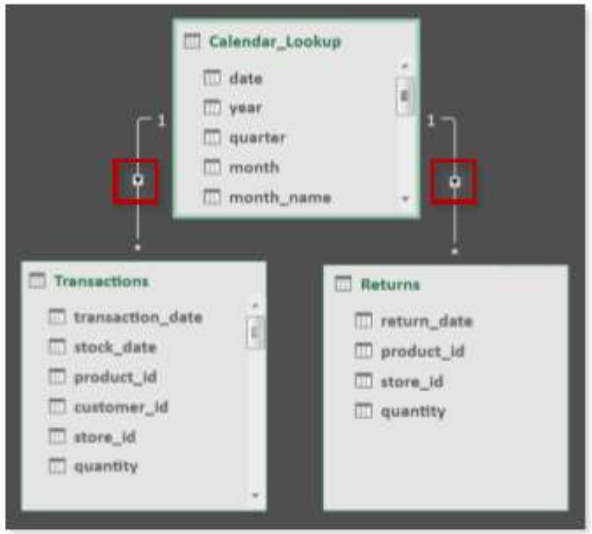
Here we've loaded a second data table named **Returns**, containing records of returns by date, product and store

- 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



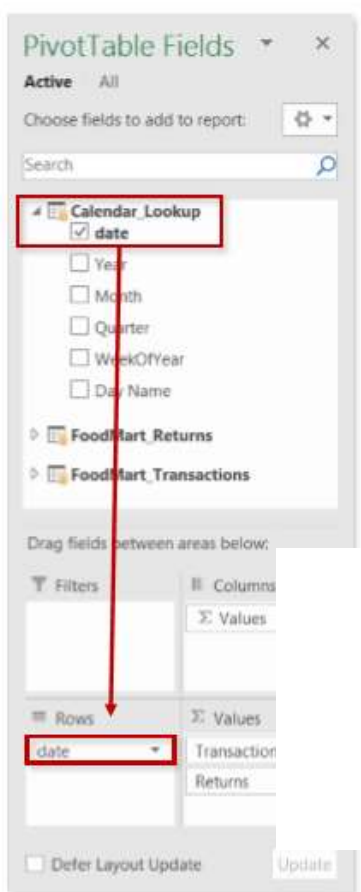
This model includes two data tables (**Transactions** and **Returns**), both connected to the **Calendar_Lookup**

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

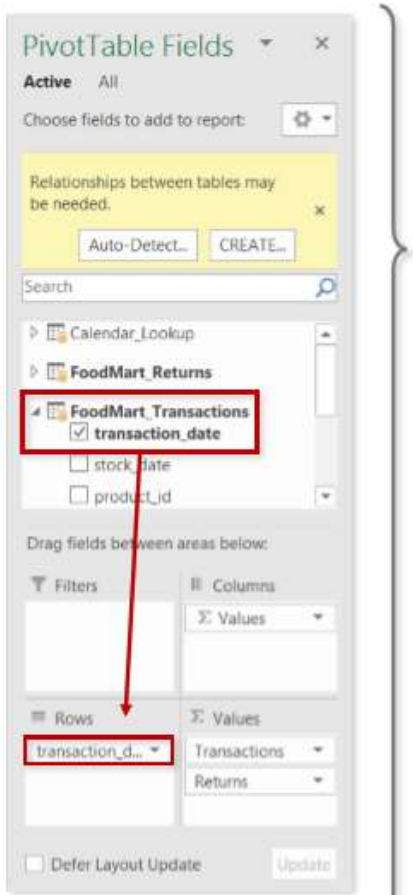
PRO TIP:

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



	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

r
y



	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*

Hiding From the 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!*

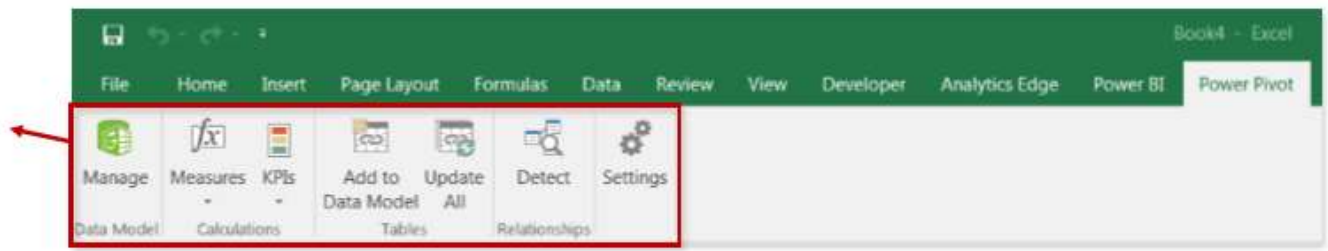
Power Pivot and DAX

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

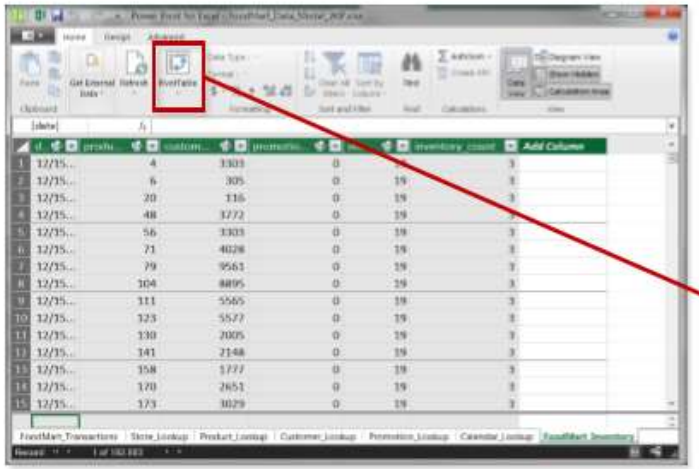
***(Note:** you may need to enable this tab by selecting **File > Options > Add-Ins > Manage COM Add-Ins**)*

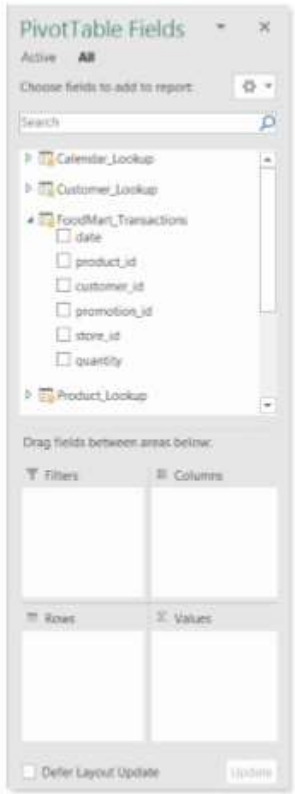


Project file

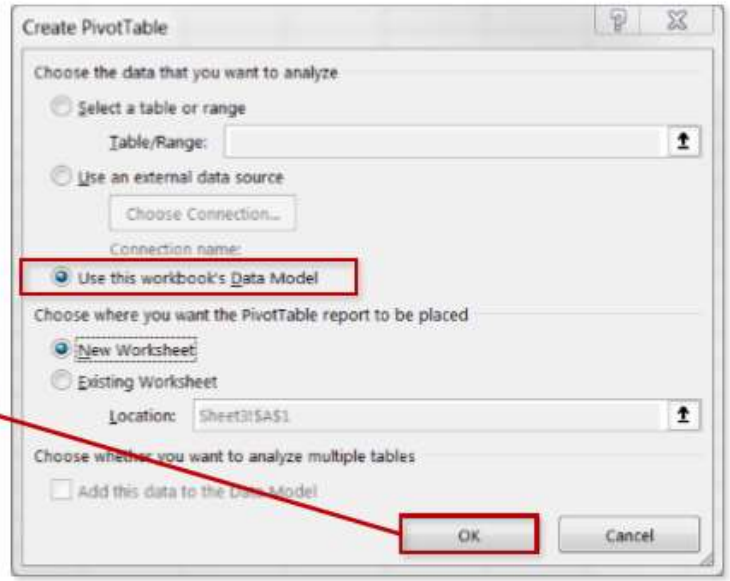
Insert Pivot Table

Option #1: From the Data Model

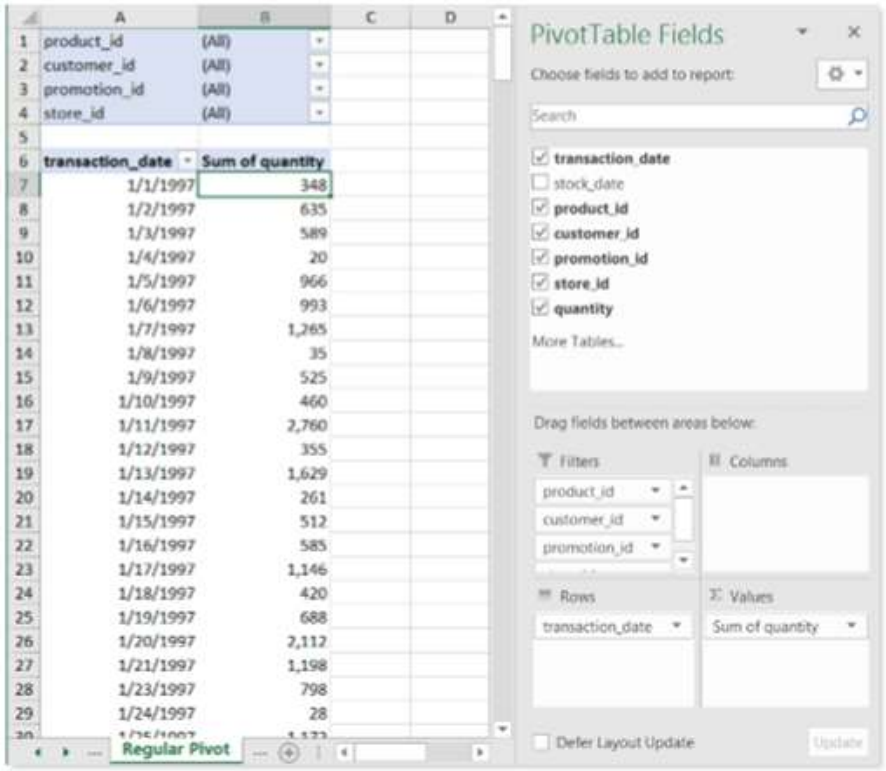




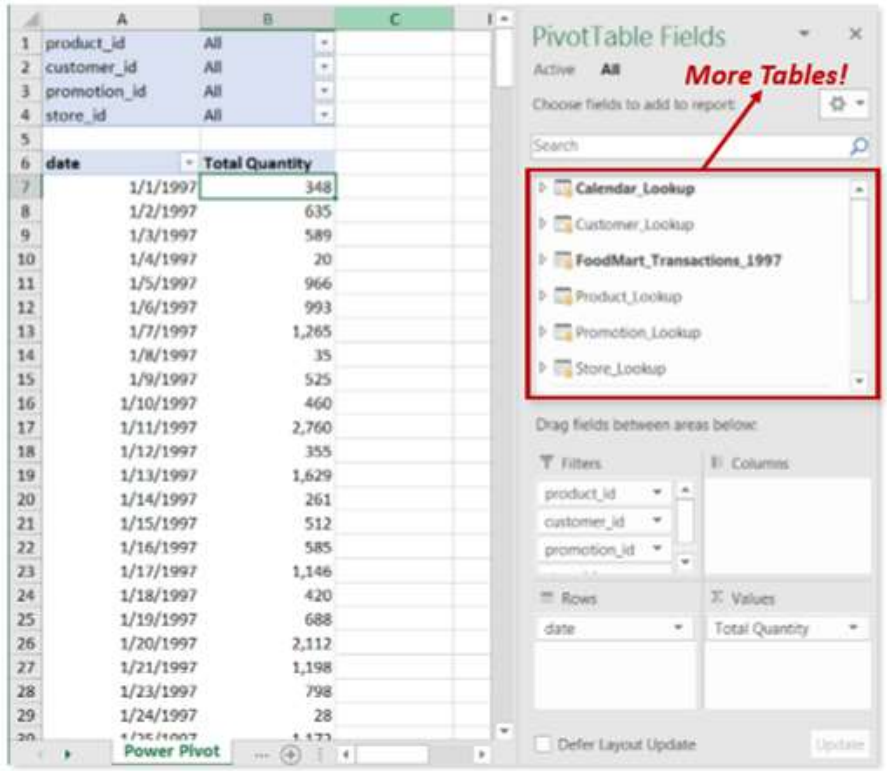
Option #2: From the *Insert* > *PivotTable* dialog box



Normal Pivot



Power Pivot



NORMAL PIVOT

- 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 PIVOT

- 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)

Calculated Columns

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

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

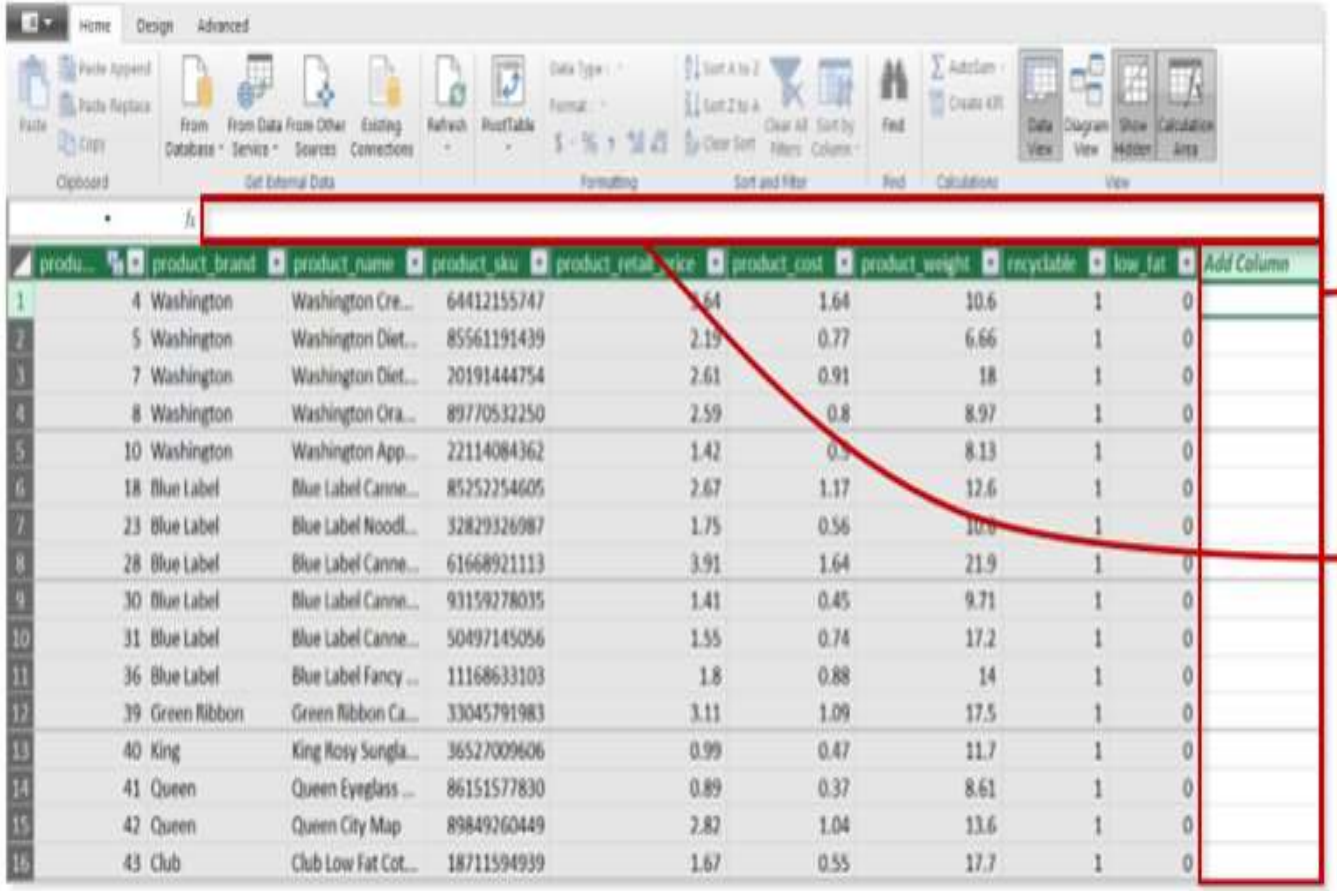
HEY THIS IS IMPORTANT!

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

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

PRO TIP:

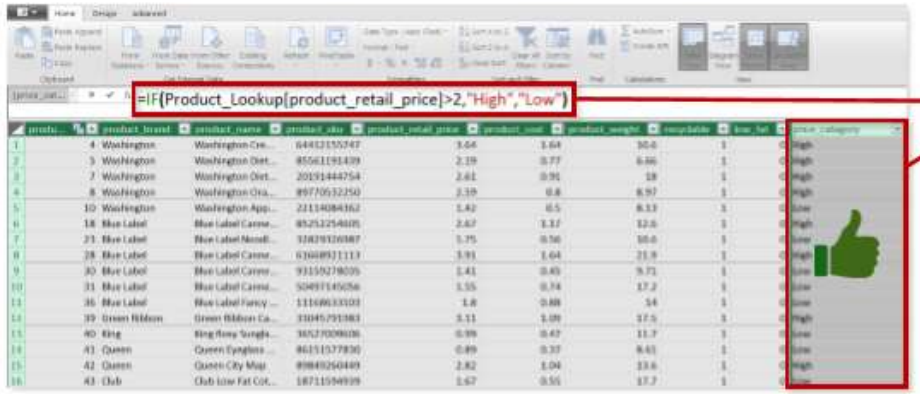
*Calculated columns are typically placed in the **Filters, Slicers, Rows** or **Columns** areas of a pivot*



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

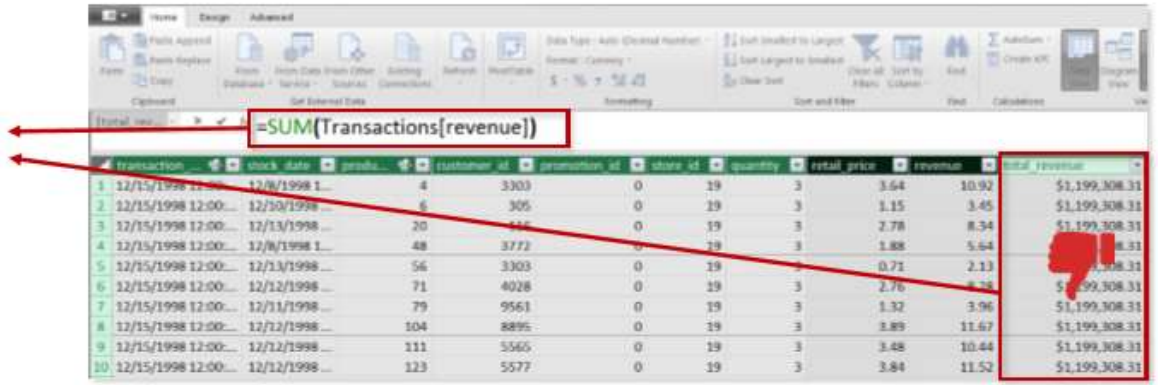


In this case we've added a **calculated column** called **price_category**, which equals "***High***" if the retail price is >$2, and "***Low***" otherwise (*just like you would write in Excel!*)

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

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

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



Measures

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

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

HEY THIS IS IMPORTANT!

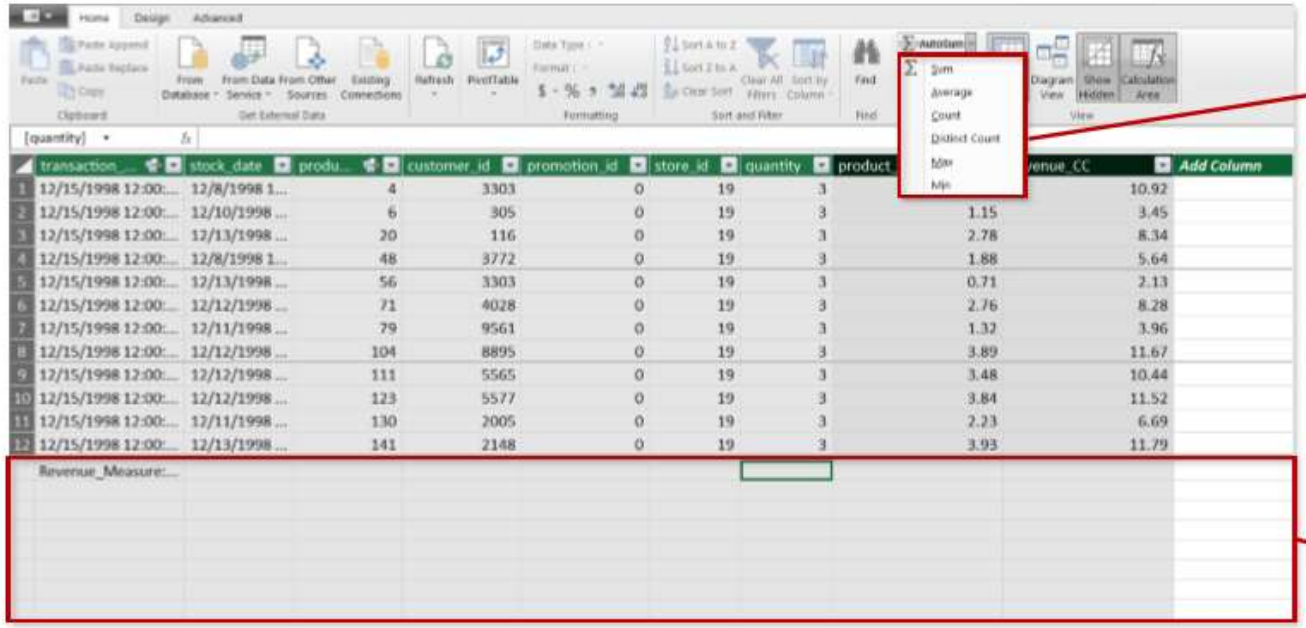
As a rule of thumb, use measures (*vs. calculated columns*) when a single row can't give you the answer (i.e. requires **aggregation**)

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

PRO TIP:

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

Start with the easiest DAX Ever



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!*

Project file – trans

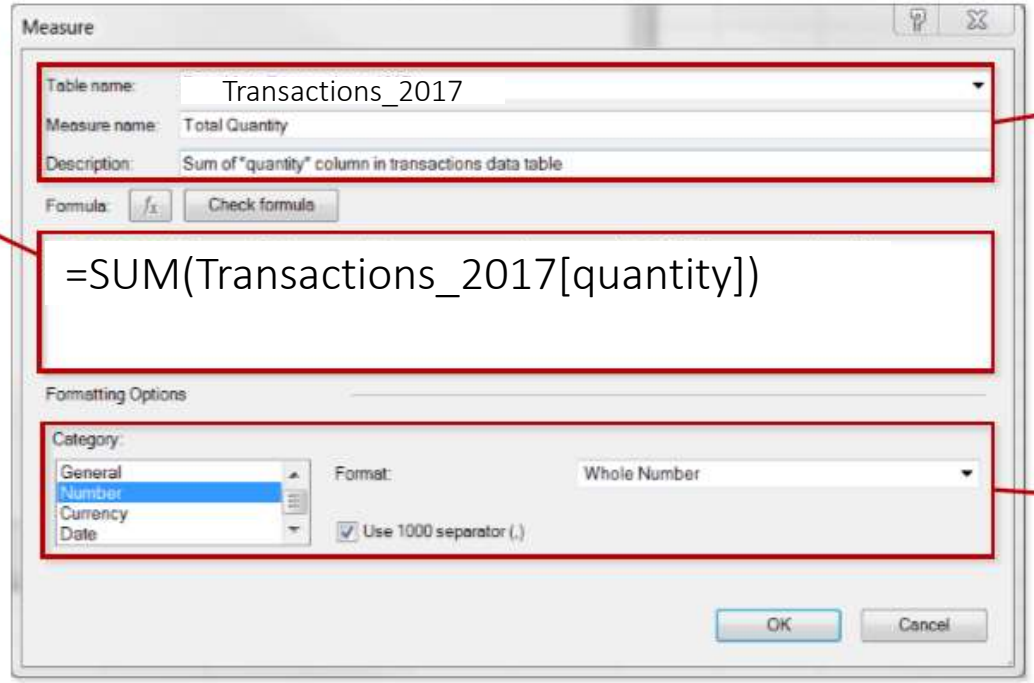
Create Measures

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 auto-populate formula names and tables*

PRO TIP:
Ctrl+scroll *adjusts formula text size*

The Measure Dialog Box



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

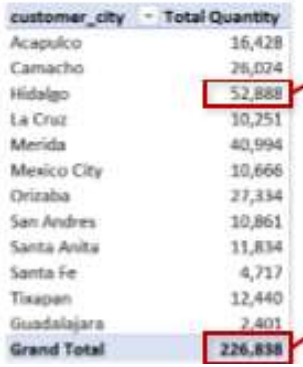
Understand the filter concept

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

HEY THIS IS IMPORTANT!

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

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



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

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

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

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

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

Cell coordinates:

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

Cell coordinates:

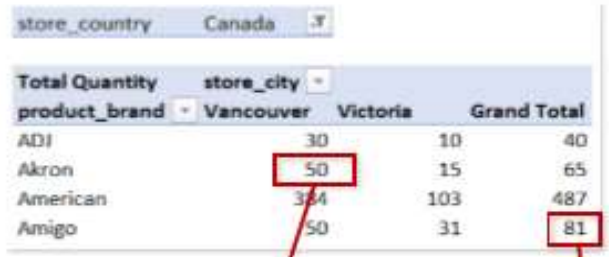
- Calendar_Lookup[Year] = **1997**
- Customer_Lookup[customer_country] = **"USA"**

Cell coordinates:

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

Cell coordinates:

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



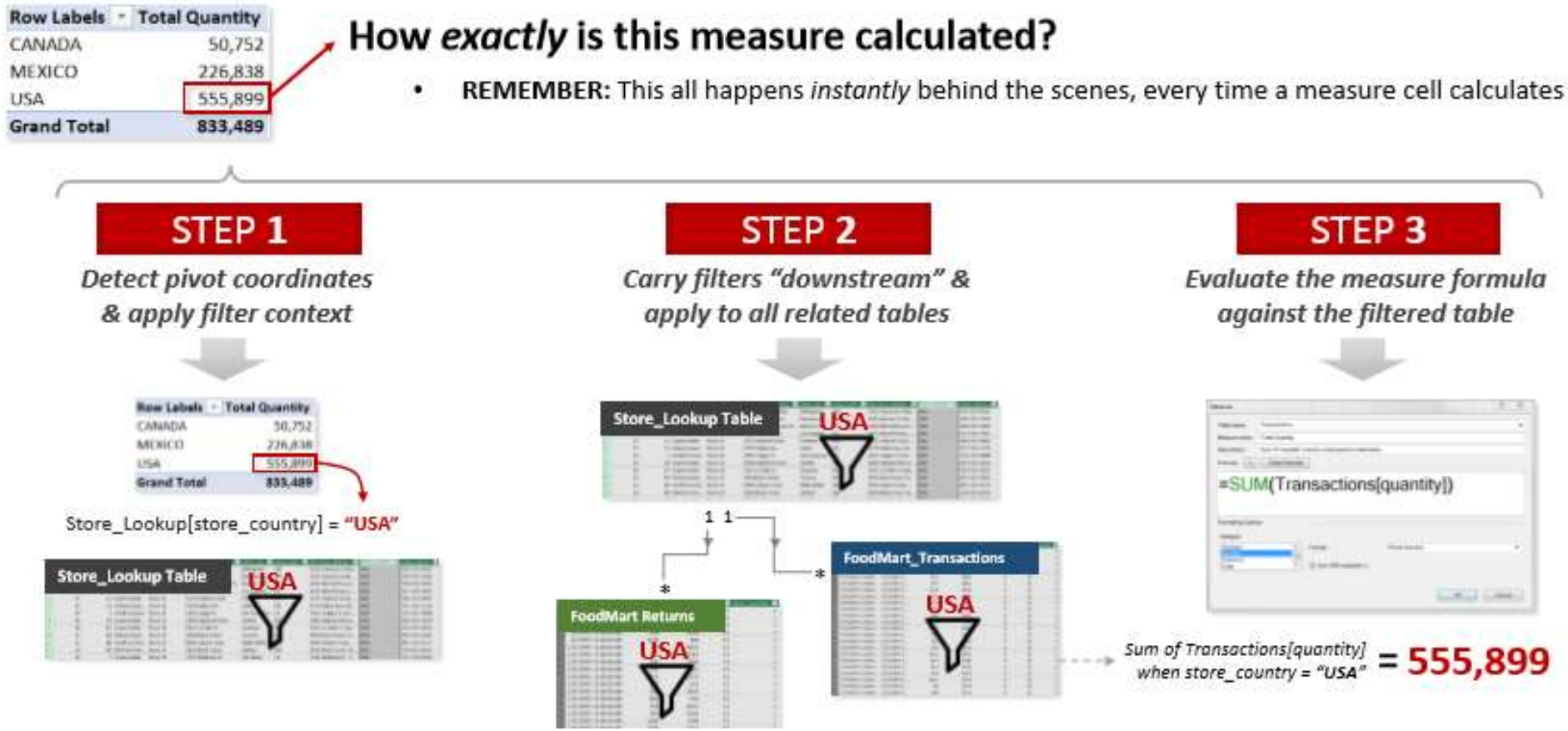
Cell coordinates:

- 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 calculation



CALCULATED COLUMNS

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

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

MEASURES

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

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

DAX Functions

DAX Syntax

MEASURE NAME

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

Referenced **TABLE NAME**

Referenced **COLUMN NAME**

```
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)*

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

Note: *Table names with spaces must be surrounded by* ***single quotes****:*

- *Without a space:* ***Transactions****[quantity]*
- *With a space:* ***'Transactions Table'****[quantity]*

PRO TIP:

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

DAX Operator

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

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

Hey! Pay attention to these!

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

Common Functions

MATH & STATS Functions

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

Common Examples:

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

Iterator Functions:

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

LOGICAL Functions

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

Common Examples:

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

TEXT Functions

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

Common Examples:

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

FILTER Functions

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

Common Examples:

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

DATE & TIME Functions

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

Common Examples:

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

Time Intelligence Functions:

- DATESYTD
- DATESQTD
- DATESMTD
- DATEADD
- DATESINPERIOD

Math

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

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

Logical Functions

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

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

Switch and Switch (true)

SWITCH() *Evaluates an expression against a list of values and returns one of multiple possible result expressions*

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

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

Examples:

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

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

Examples: Values

```
=SWITCH(Calendar_Lookup[month_num],
    1, "January",
    2, "February",
    etc...
```

Conditions

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

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

PRO TIP:

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

Text Functions

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

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

CALCULATE()

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

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

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

Examples:

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

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

Examples:

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

PRO TIP:

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

Using Filter with calculated function

FILTER() *Returns a table that represents a subset of another table or expression*

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

Table to be filtered

Examples:

- *Store_Lookup*
- *Product_Lookup*

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

Examples:

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

HEY THIS IS IMPORTANT!

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

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

PRO TIP:

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

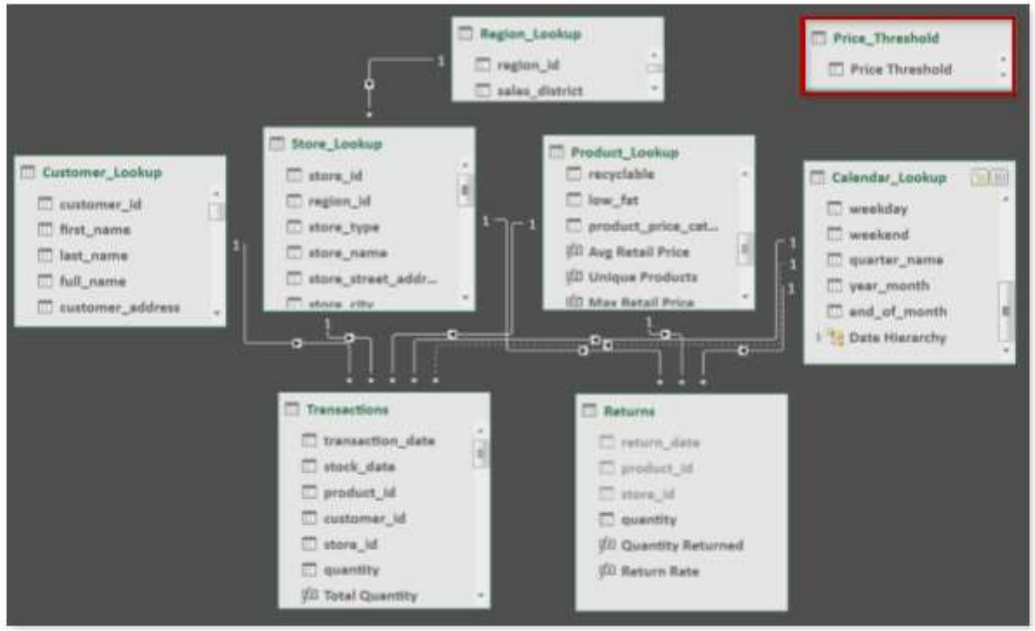
Lets Show the best use of Filters

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

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

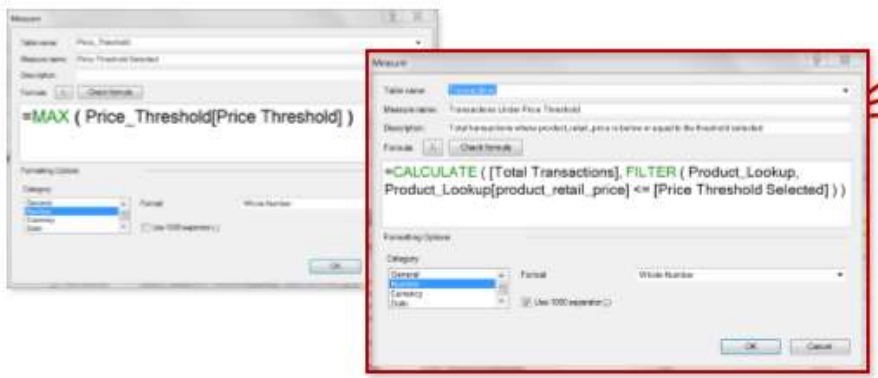
STEP 2: *Add the table to the **Data Model** (from **Power Pivot** tab):*

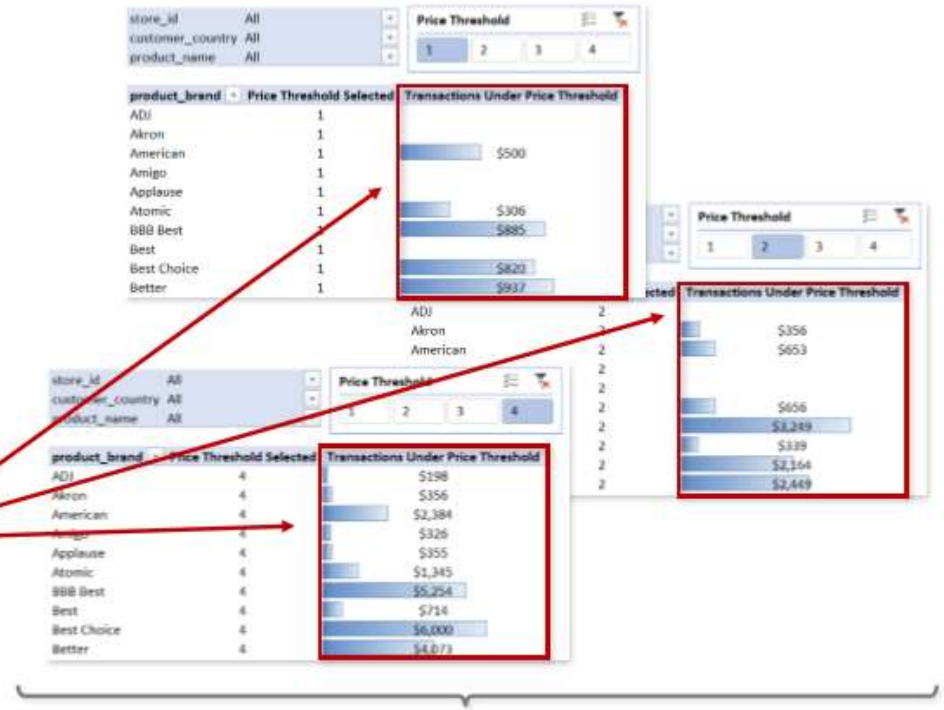
STEP 3: *Make sure that your table loaded, and is NOT connected to any other table in the model:*



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***

All Function
(Filter Opposite, use it with other function)

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

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

The table or column that you want to clear filters on

Examples:

- *Transactions*
- *Product_Lookup[product_brand]*

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

Notes:

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

Examples:

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

PRO TIP:

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

Related function

RELATED() *Returns related values in each row of a table using relationships with other tables*

=RELATED(<column>)

The column that contains the values you want to retrieve

Examples:

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

HEY THIS IS IMPORTANT!

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

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

PRO TIP:

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

X functions

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

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

*Aggregation to apply to calculated rows**

Examples:

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

Table in which the expression will be evaluated

Examples:

- *Transactions*
- *FILTER(Transactions, RELATED(Store_Lookup[country])="USA")*

Expression to be evaluated for each row of the given table

Examples:

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

PRO TIP:

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

- Then apply conditional formatting after sorting with colors
- Apply conditional formatting also for the ray area
- Then show data

Row Labels	total trans	Sum of revenue	total revenue	Total Revenue X	Product Rank (revenue)
Anti-infectives	**27603**	**1,840,148**	**1,840,148**	**1,840,148**	**1**
Ciprofar	4535	462,693	462,693	462,693	10
Sulbin 750 mg	4591	427,830	427,830	427,830	12
Baby Drink	4651	419,601	419,601	419,601	14
Clavimox	4673	273,486	273,486	273,486	24
Cefaxone	4621	157,894	157,894	157,894	31
Sulbin 1500 mg	4532	98,644	98,644	98,644	35
Antipyretics	**4590**	**412,206**	**412,206**	**412,206**	**16**
Alambuphine	4590	412,206	412,206	412,206	16
Central Nervous System	**4657**	**644,985**	**644,985**	**644,985**	**1**
Fawar Fruit	4657	644,985	644,985	644,985	1
Dermatology	**13710**	**997,007**	**997,007**	**997,007**	**1**
Ampiflux	4504	418,620	418,620	418,620	15
Acne Zinc	4486	331,632	331,632	331,632	20
Amrase	4720	246,755	246,755	246,755	26
Gastrointestinal Tract	**31962**	**2,440,160**	**2,440,160**	**2,440,160**	**1**
Alucal Plus	4549	537,548	537,548	537,548	6
Amflux	4614	471,009	471,009	471,009	9
Spasmodigestin	4630	402,052	402,052	402,052	17
Rani	4593	386,235	386,235	386,235	19
Neurovit	4395	245,628	245,628	245,628	27
Declophen	4481	222,944	222,944	222,944	29

Data and Time Functions

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

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

Performance To-Date

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

Ex : Sum of sales from the first of year to the current date

Previous Period

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

Ex : compare between current month and previus month sales

# of intervals to compare (i.e. previous month, rolling 10-day)

Running Total

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

Ex : sum total sales f

PRO TIP:

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

KPI (Key Performance Indicator)

KPI

KPI’s

- What is KPI mean
  - Key Performance Indicator
- Why you should use KPI’s
  - Measuring Performance
- KPI’s Requirement
  - At least one measure
  - Two measure (one for value and other to compared with Ex , total sales Vs Target)
- KPI’s Absolute Value
  - One measure only needed
  - You wrote your second measure (target) your self and set the ranges
- Edit and remove KPI’s if needed

Thank You