BIG DATA
EXPERIENCE

Data Science

Data Analytic Expression (DAX)

Asst. Prof. Dr. Santitham Prom-on

Department of Computer Engineering, Faculty of Engineering
King Mongkut's University of Technology Thonburi

BIG DATA
EXPERIENCE
CENTER

KMUTT g-able

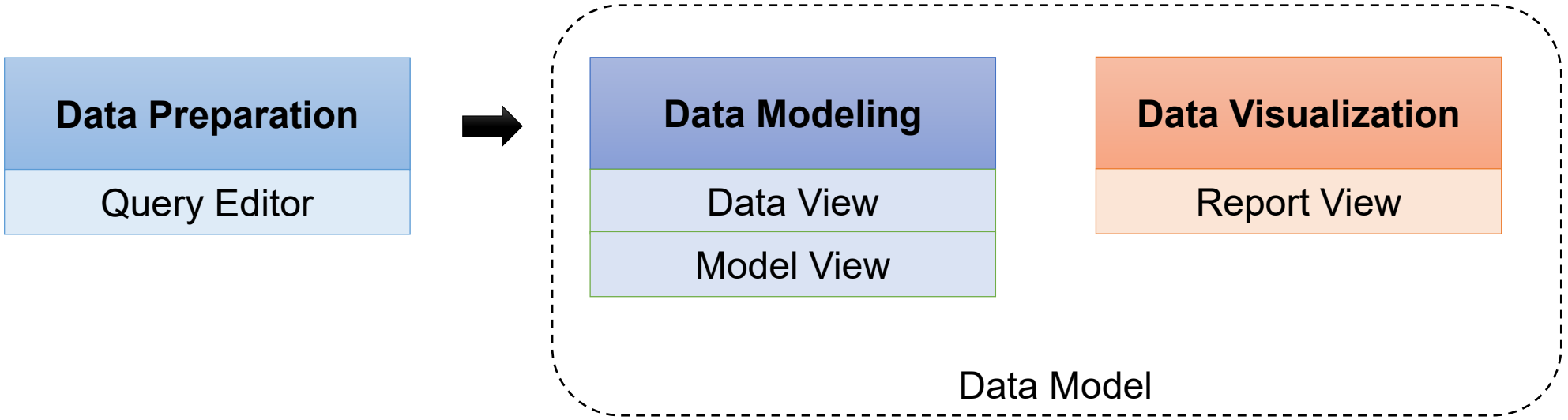
BIG DATA EXPERIENCE

Part 1

- Power BI workflow
- DAX engine and evaluations
- Basic DAX functions
  - DAX Aggregation

KMUTT g-able

Power BI desktop Workflow



Microsoft Adventure Works Database

AdventureWorks is a worldwide chain of bicycle stores that has retail stores in countries including Australia, the United States, Canada, France, and the United Kingdom.

It consists of the following tables

- Calendar
- Products
- Customers
- Territory
- Sales
- dimProductSubCategory
- dimProductCategory
- BudgetPeriod
- Budget

Tax Invoice / Receipt
Adventure Works
Australia

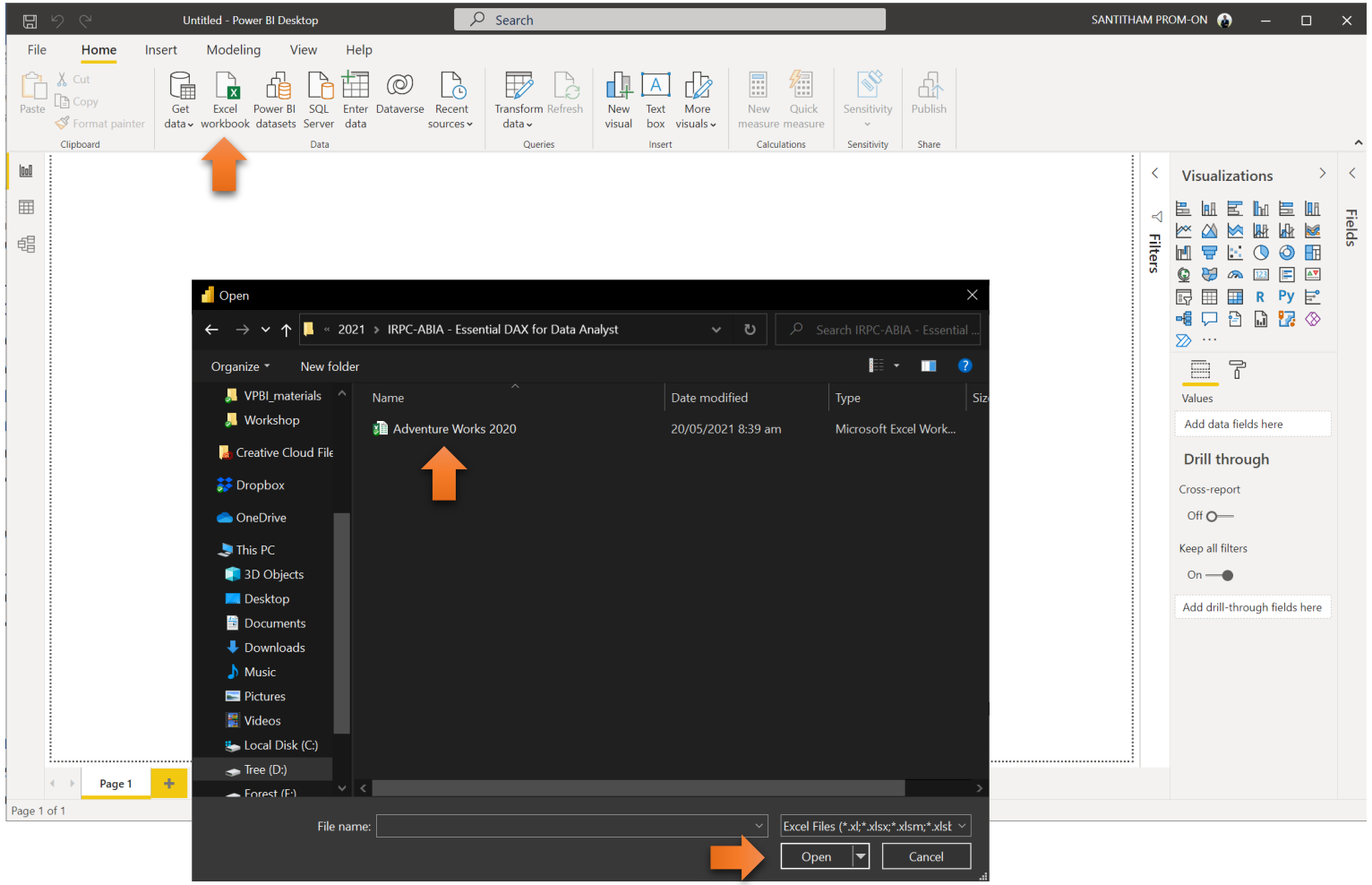
OrderNo: **SO57418** 4/11/2018
Bill To: Jon Yang
Cust No.: 11000

Item	Description	Amount
573	Touring-1000 Blue, 46	$ 2,384.07
541	Touring Tire	$ 28.99
530	Touring Tire Tube	$ 4.99
214	Sport-100 Helmet, Red	$ 34.99
488	Short-Sleeve Classic Jersey, S	$ 53.99

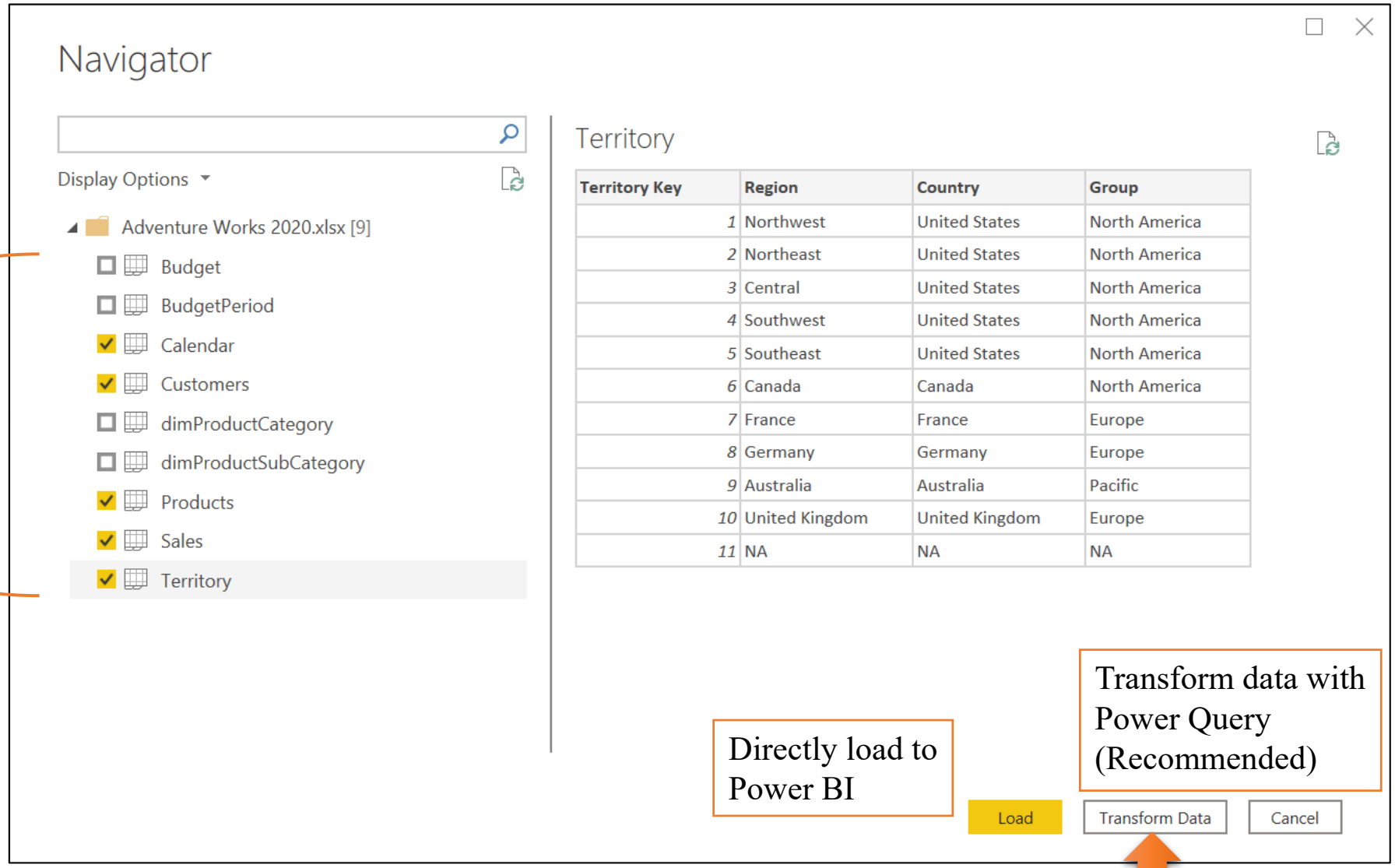
Subtotal:	$ 2,507.03
Total ex tax:	$ 2,507.03
Tax (8%):	$ 200.56
Total Inc tax:	$ 2,707.59

http://fastdata.in.th/IRPC_EDDA/Adventure%20Works%202020.xlsx

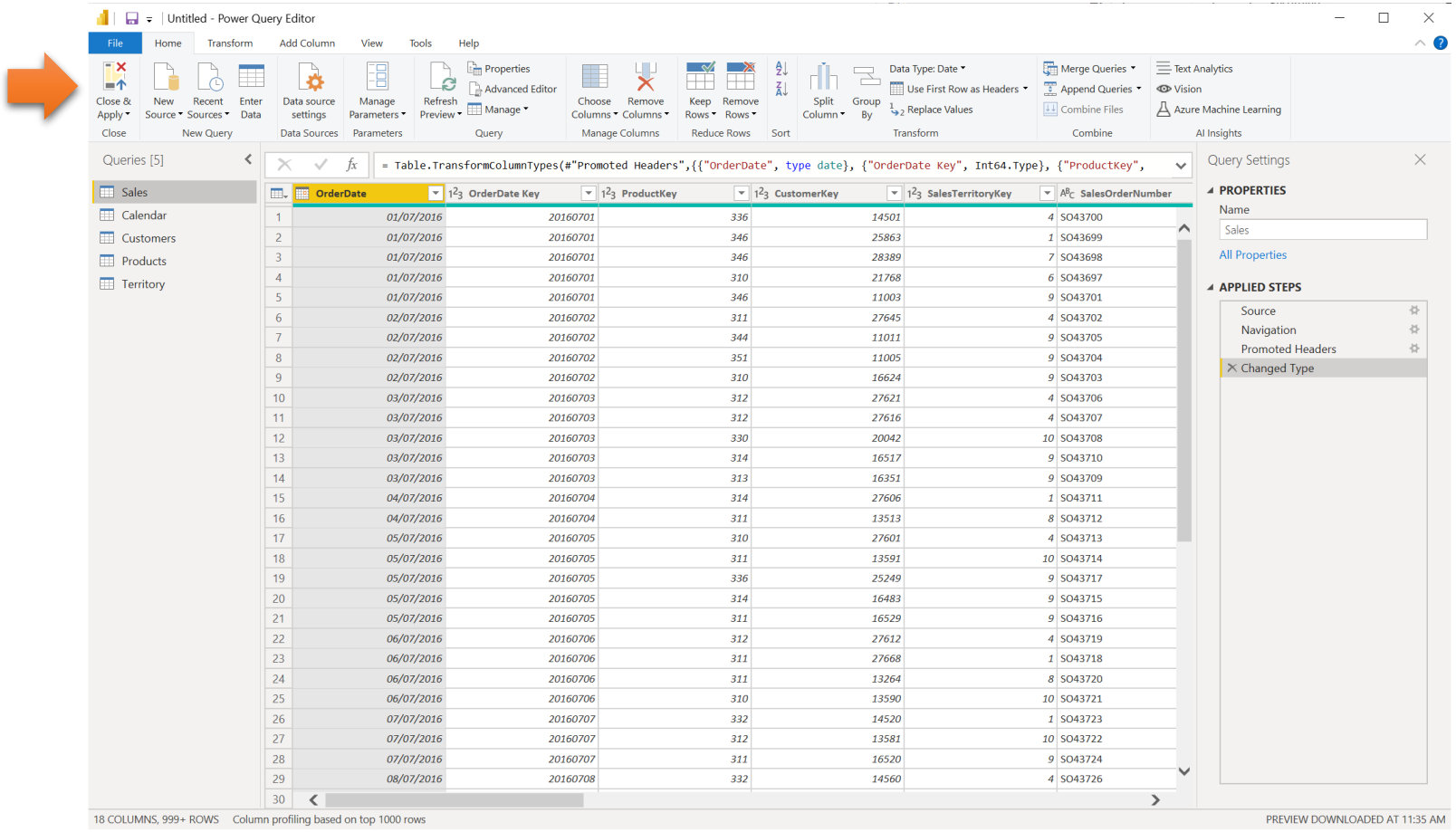
Import data


Untitled - Power BI Desktop
Search
SANTITHAM PROM-ON
File
Home
Insert
Modeling
View
Help
Paste
Cut
Copy
Format painter
Clipboard
Get data
Excel workbook
Power BI datasets
SQL Server
Enter data
Dataverse
Recent sources
Data
Transform data
Refresh
Queries
New visual
Text box
More visuals
Insert
New measure
Quick measure
Calculations
Sensitivity
Sensitivity
Publish
Share
Visualizations
Filters
Fields
Values
Add data fields here
Drill through
Cross-report
Off
Keep all filters
On
Add drill-through fields here
Open
2021 › IRPC-ABIA - Essential DAX for Data Analyst
Search IRPC-ABIA - Essential ...
Organize
New folder
VPBI_materials
Workshop
Creative Cloud File
Dropbox
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Tree (D:)
Name
Date modified
Type
Adventure Works 2020
20/05/2021 8:39 am
Microsoft Excel Work...
File name:
Excel Files (*.xl;*.xlsx;*.xlsm;*.xlst
Open
Cancel
Page 1
Page 1 of 1

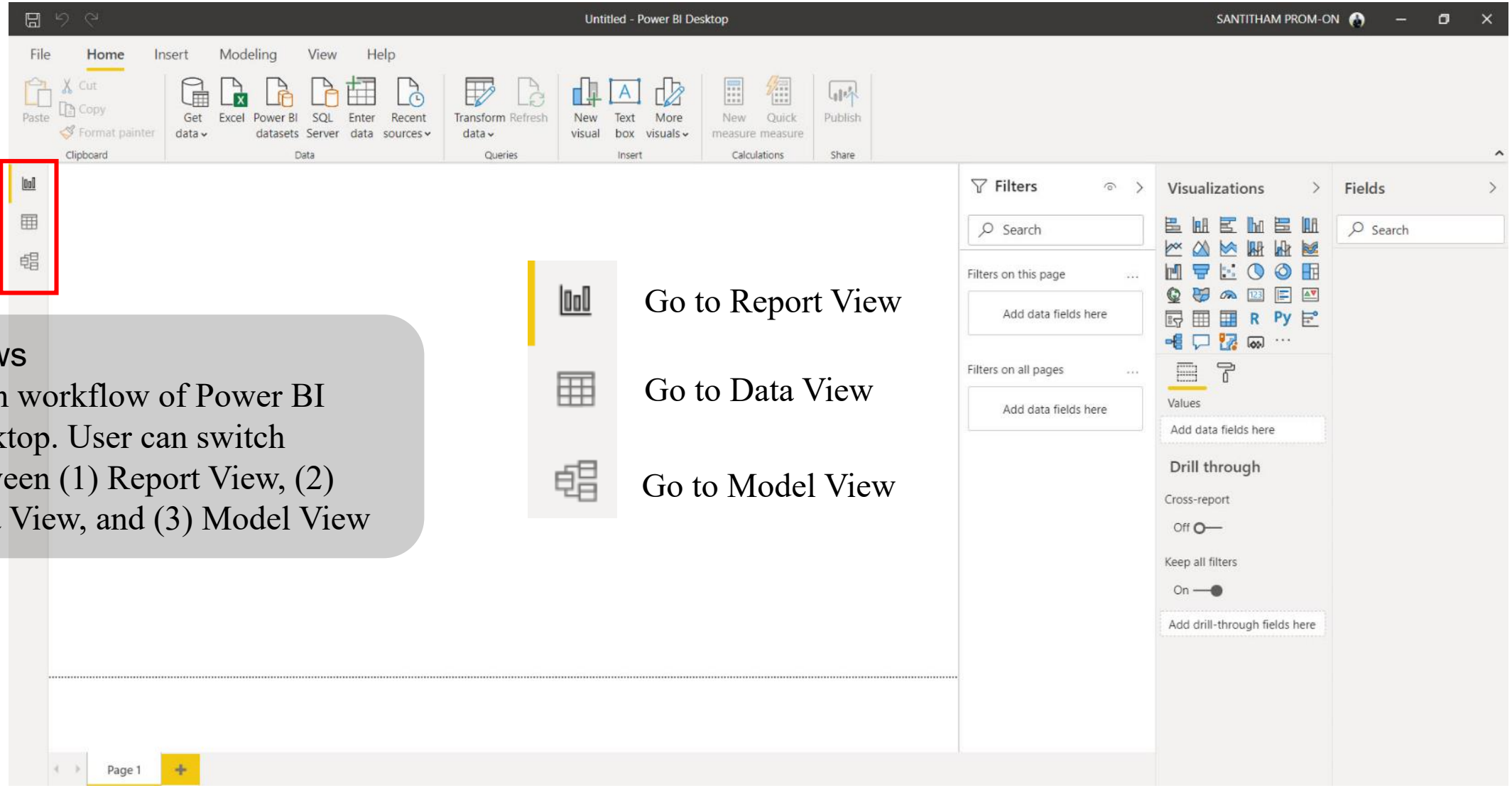
Navigator


Navigator
Display Options
Adventure Works 2020.xlsx [9]
Budget
BudgetPeriod
Calendar
Customers
dimProductCategory
dimProductSubCategory
Products
Sales
Territory
Territory
Territory Key | Region | Country | Group
1 | Northwest | United States | North America
2 | Northeast | United States | North America
3 | Central | United States | North America
4 | Southwest | United States | North America
5 | Southeast | United States | North America
6 | Canada | Canada | North America
7 | France | France | Europe
8 | Germany | Germany | Europe
9 | Australia | Australia | Pacific
10 | United Kingdom | United Kingdom | Europe
11 | NA | NA | NA
Directly load to Power BI
Transform data with Power Query (Recommended)
Load
Transform Data
Cancel

Power Query


Untitled - Power Query Editor
File
Home
Transform
Add Column
View
Tools
Help
Close & Apply
New Source
Recent Sources
Enter Data
Data source settings
Manage Parameters
Refresh Preview
Properties
Advanced Editor
Manage
Choose Columns
Remove Columns
Keep Rows
Remove Rows
Split Column
Group By
Data Type: Date
Use First Row as Headers
Replace Values
Merge Queries
Append Queries
Combine Files
Text Analytics
Vision
Azure Machine Learning
Close
New Query
Data Sources
Parameters
Query
Manage Columns
Reduce Rows
Sort
Transform
Combine
AI Insights
Queries [5]
Sales
Calendar
Customers
Products
Territory
= Table.TransformColumnTypes(#"Promoted Headers",{{"OrderDate", type date}, {"OrderDate Key", Int64.Type}, {"ProductKey",
OrderDate
OrderDate Key
ProductKey
CustomerKey
SalesTerritoryKey
SalesOrderNumber
1 01/07/2016 20160701 336 14501 4 SO43700
2 01/07/2016 20160701 346 25863 1 SO43699
3 01/07/2016 20160701 346 28389 7 SO43698
4 01/07/2016 20160701 310 21768 6 SO43697
5 01/07/2016 20160701 346 11003 9 SO43701
6 02/07/2016 20160702 311 27645 4 SO43702
7 02/07/2016 20160702 344 11011 9 SO43705
8 02/07/2016 20160702 351 11005 9 SO43704
9 02/07/2016 20160702 310 16624 9 SO43703
10 03/07/2016 20160703 312 27621 4 SO43706
11 03/07/2016 20160703 312 27616 4 SO43707
12 03/07/2016 20160703 330 20042 10 SO43708
13 03/07/2016 20160703 314 16517 9 SO43710
14 03/07/2016 20160703 313 16351 9 SO43709
15 04/07/2016 20160704 314 27606 1 SO43711
16 04/07/2016 20160704 311 13513 8 SO43712
17 05/07/2016 20160705 310 27601 4 SO43713
18 05/07/2016 20160705 311 13591 10 SO43714
19 05/07/2016 20160705 336 25249 9 SO43717
20 05/07/2016 20160705 314 16483 9 SO43715
21 05/07/2016 20160705 311 16529 9 SO43716
22 06/07/2016 20160706 312 27612 4 SO43719
23 06/07/2016 20160706 311 27668 1 SO43718
24 06/07/2016 20160706 311 13264 8 SO43720
25 06/07/2016 20160706 310 13590 10 SO43721
26 07/07/2016 20160707 332 14520 1 SO43723
27 07/07/2016 20160707 312 13581 10 SO43722
28 07/07/2016 20160707 311 16520 9 SO43724
29 08/07/2016 20160708 332 14560 4 SO43726
30
Query Settings
PROPERTIES
Name
Sales
All Properties
APPLIED STEPS
Source
Navigation
Promoted Headers
Changed Type
18 COLUMNS, 999+ ROWS
Column profiling based on top 1000 rows
PREVIEW DOWNLOADED AT 11:35 AM

BIG DATA EXPERIENCE



Views
Main workflow of Power BI Desktop. User can switch between (1) Report View, (2) Data View, and (3) Model View

Go to Report View

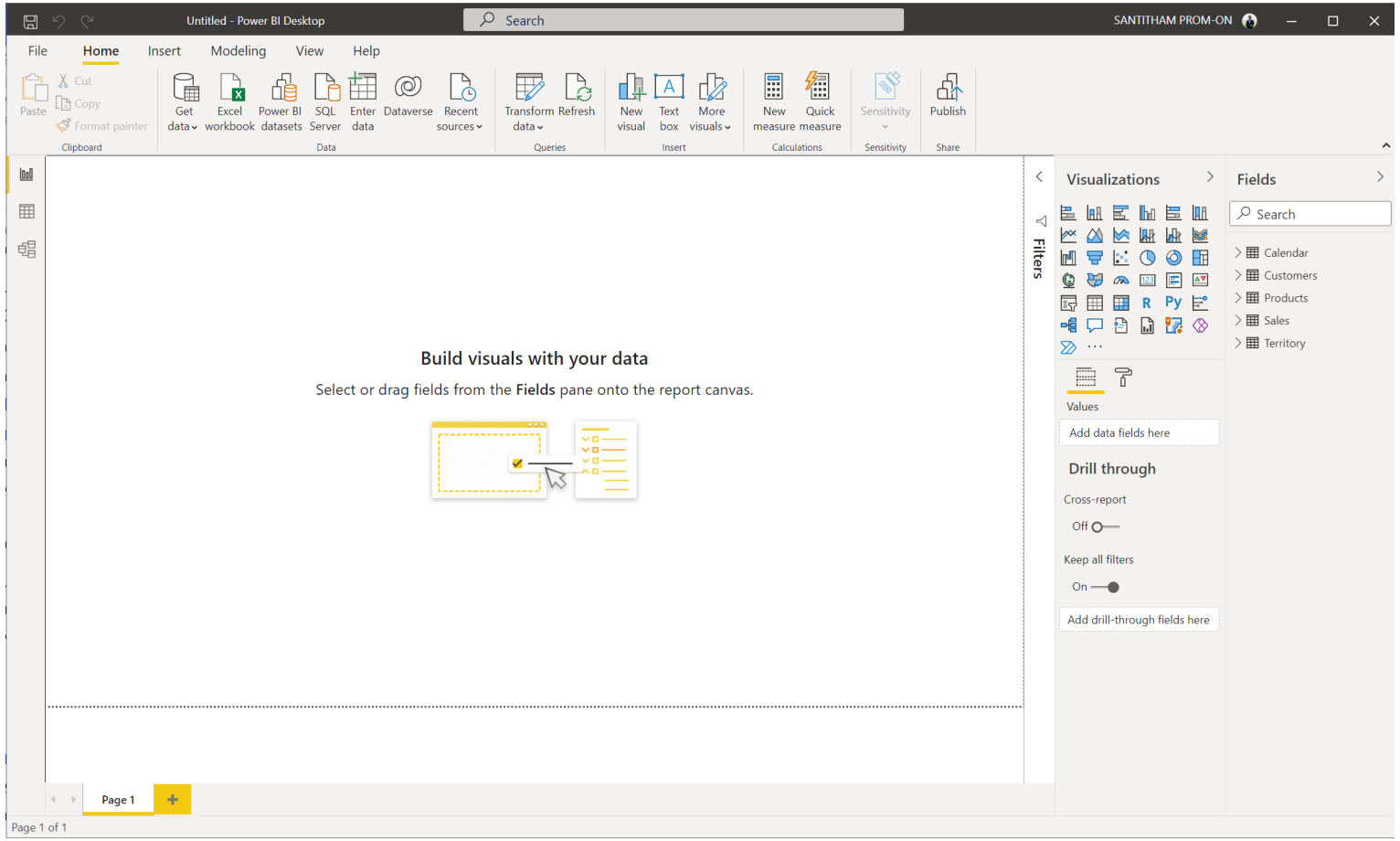
Go to Data View

Go to Model View

KMUTT g-able

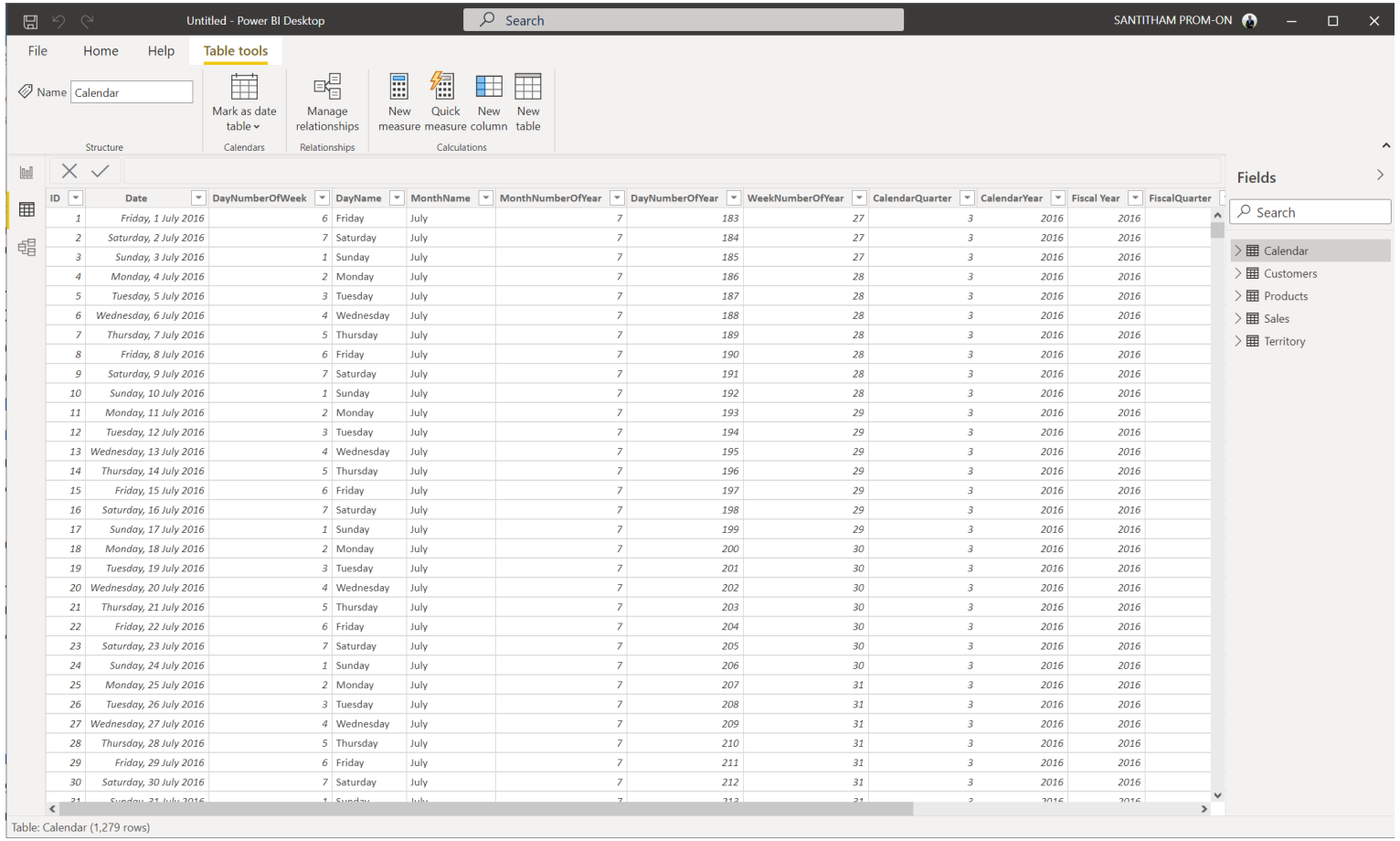
Report View

Report View
This View is used for creating visualization.
Data and Model have to be defined first.



Data View

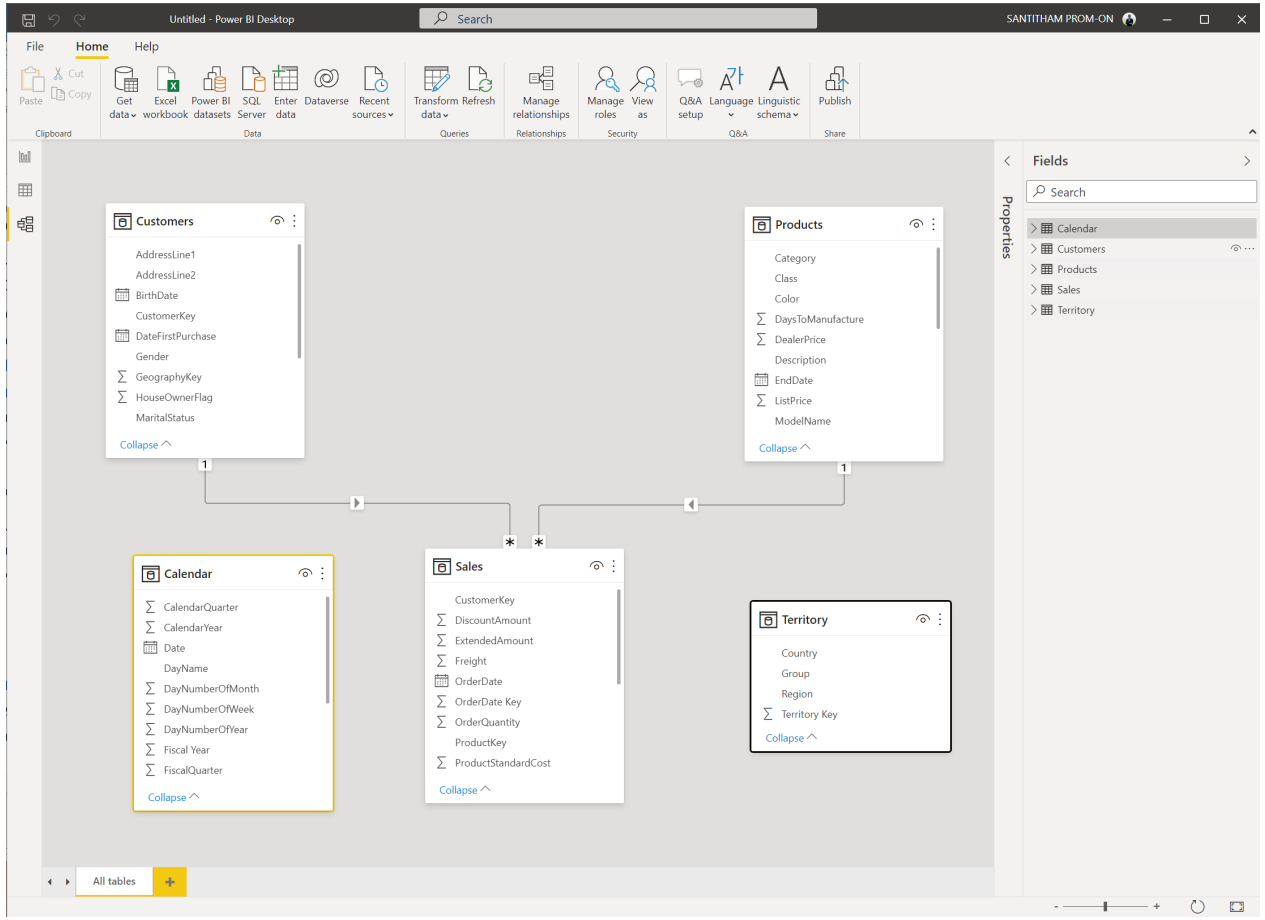
Data View
This View shows the detail data for each table.
It also has tools to process the data.



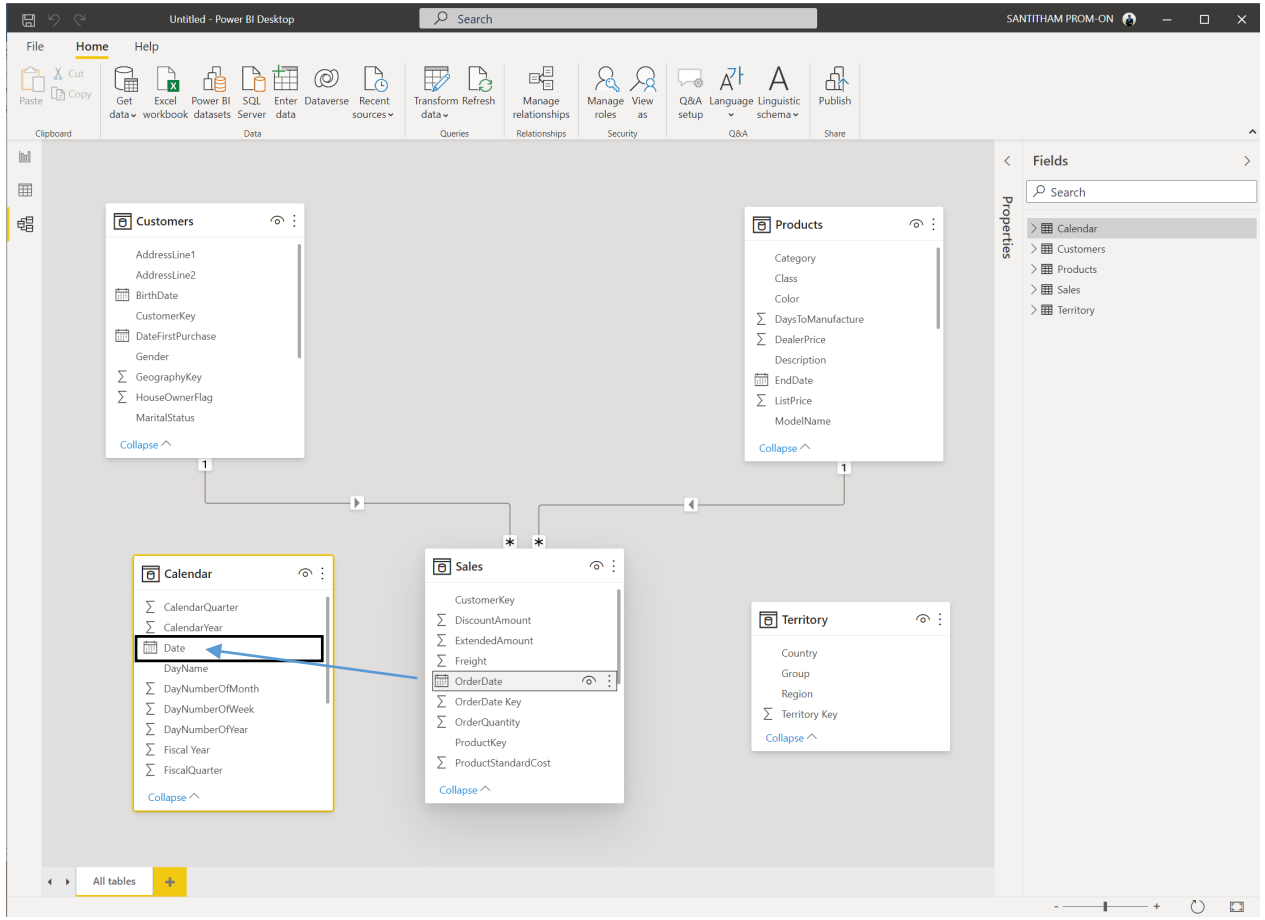
ID	Date	DayNumberOfWeek	DayName	MonthName	MonthNumberOfYear	DayNumberOfYear	WeekNumberOfYear	CalendarQuarter	CalendarYear	Fiscal Year	FiscalQuarter
1	Friday, 1 July 2016	6	Friday	July	7	183	27	3	2016	2016	
2	Saturday, 2 July 2016	7	Saturday	July	7	184	27	3	2016	2016	
3	Sunday, 3 July 2016	1	Sunday	July	7	185	27	3	2016	2016	
4	Monday, 4 July 2016	2	Monday	July	7	186	28	3	2016	2016	
5	Tuesday, 5 July 2016	3	Tuesday	July	7	187	28	3	2016	2016	
6	Wednesday, 6 July 2016	4	Wednesday	July	7	188	28	3	2016	2016	
7	Thursday, 7 July 2016	5	Thursday	July	7	189	28	3	2016	2016	
8	Friday, 8 July 2016	6	Friday	July	7	190	28	3	2016	2016	
9	Saturday, 9 July 2016	7	Saturday	July	7	191	28	3	2016	2016	
10	Sunday, 10 July 2016	1	Sunday	July	7	192	28	3	2016	2016	
11	Monday, 11 July 2016	2	Monday	July	7	193	29	3	2016	2016	
12	Tuesday, 12 July 2016	3	Tuesday	July	7	194	29	3	2016	2016	
13	Wednesday, 13 July 2016	4	Wednesday	July	7	195	29	3	2016	2016	
14	Thursday, 14 July 2016	5	Thursday	July	7	196	29	3	2016	2016	
15	Friday, 15 July 2016	6	Friday	July	7	197	29	3	2016	2016	
16	Saturday, 16 July 2016	7	Saturday	July	7	198	29	3	2016	2016	
17	Sunday, 17 July 2016	1	Sunday	July	7	199	29	3	2016	2016	
18	Monday, 18 July 2016	2	Monday	July	7	200	30	3	2016	2016	
19	Tuesday, 19 July 2016	3	Tuesday	July	7	201	30	3	2016	2016	
20	Wednesday, 20 July 2016	4	Wednesday	July	7	202	30	3	2016	2016	
21	Thursday, 21 July 2016	5	Thursday	July	7	203	30	3	2016	2016	
22	Friday, 22 July 2016	6	Friday	July	7	204	30	3	2016	2016	
23	Saturday, 23 July 2016	7	Saturday	July	7	205	30	3	2016	2016	
24	Sunday, 24 July 2016	1	Sunday	July	7	206	30	3	2016	2016	
25	Monday, 25 July 2016	2	Monday	July	7	207	31	3	2016	2016	
26	Tuesday, 26 July 2016	3	Tuesday	July	7	208	31	3	2016	2016	
27	Wednesday, 27 July 2016	4	Wednesday	July	7	209	31	3	2016	2016	
28	Thursday, 28 July 2016	5	Thursday	July	7	210	31	3	2016	2016	
29	Friday, 29 July 2016	6	Friday	July	7	211	31	3	2016	2016	
30	Saturday, 30 July 2016	7	Saturday	July	7	212	31	3	2016	2016	

Model View

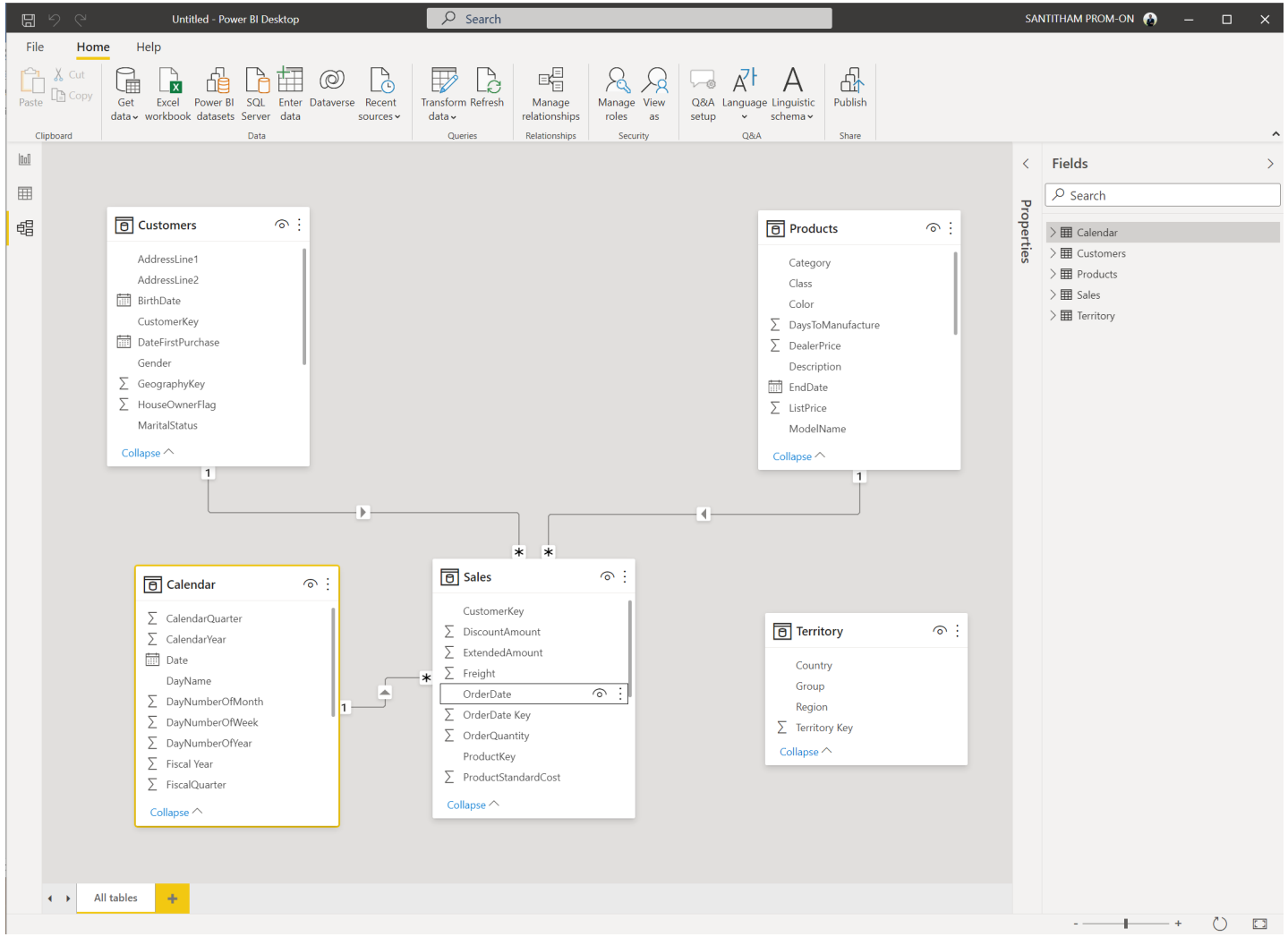
Model View
This view shows the relationships between each tables



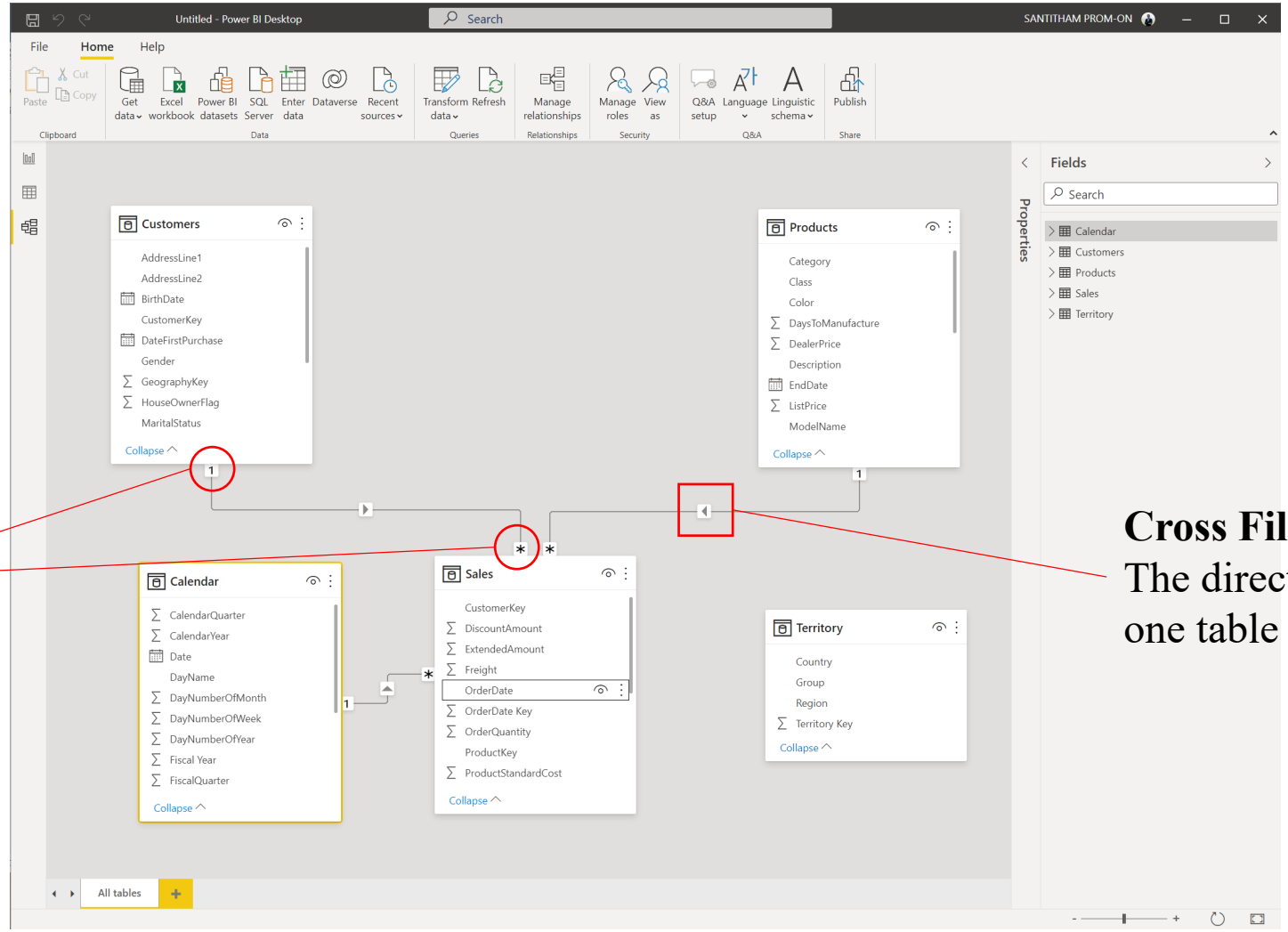
Create relationship



Relationships
(More explanation in Module 2)



Relationships
(More explanation in Module 2)



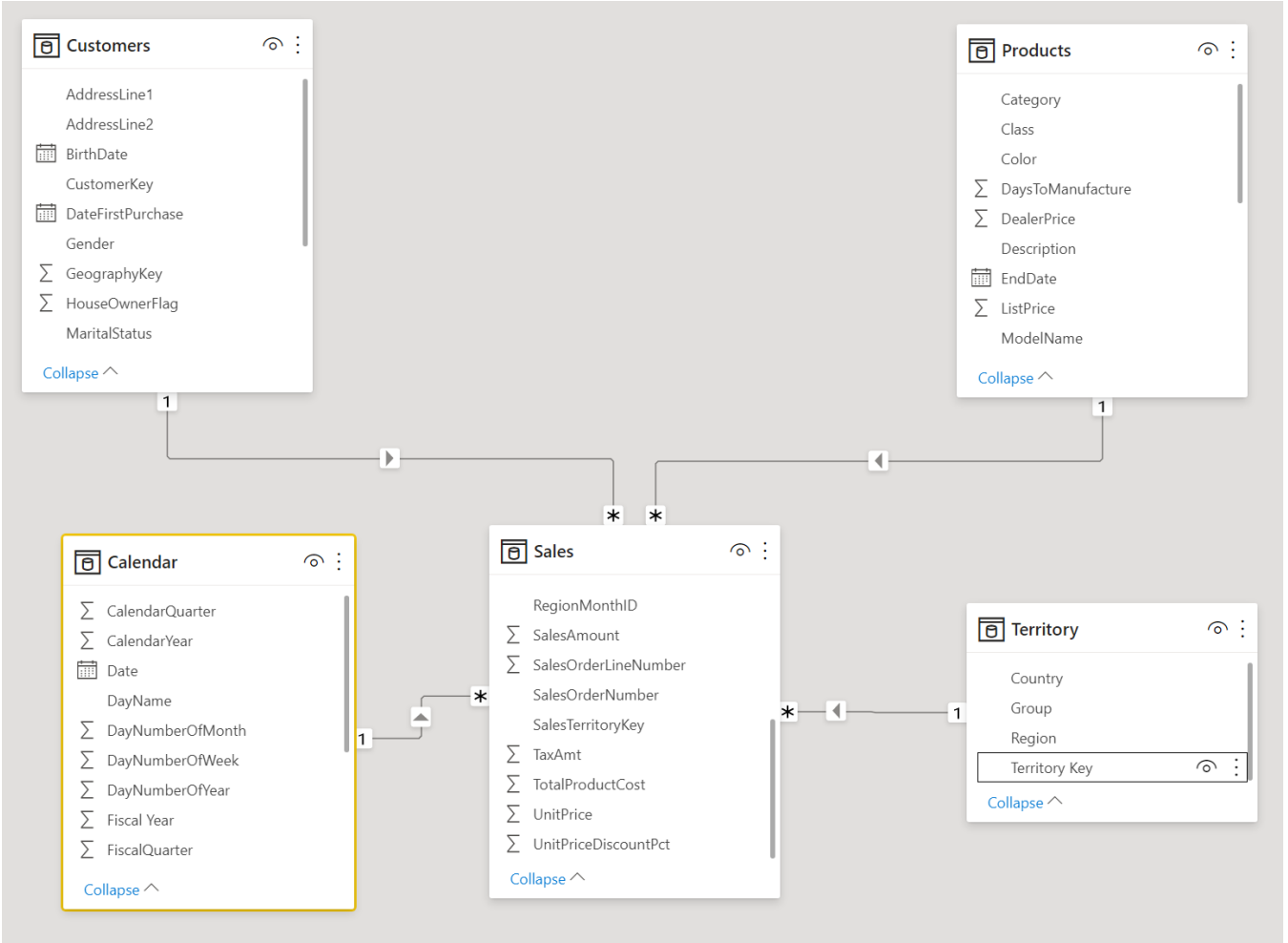
Cardinality
Types of relation

Cross Filter Direction
The direction that the data from one table filter another table.

Activity – Adding relationship

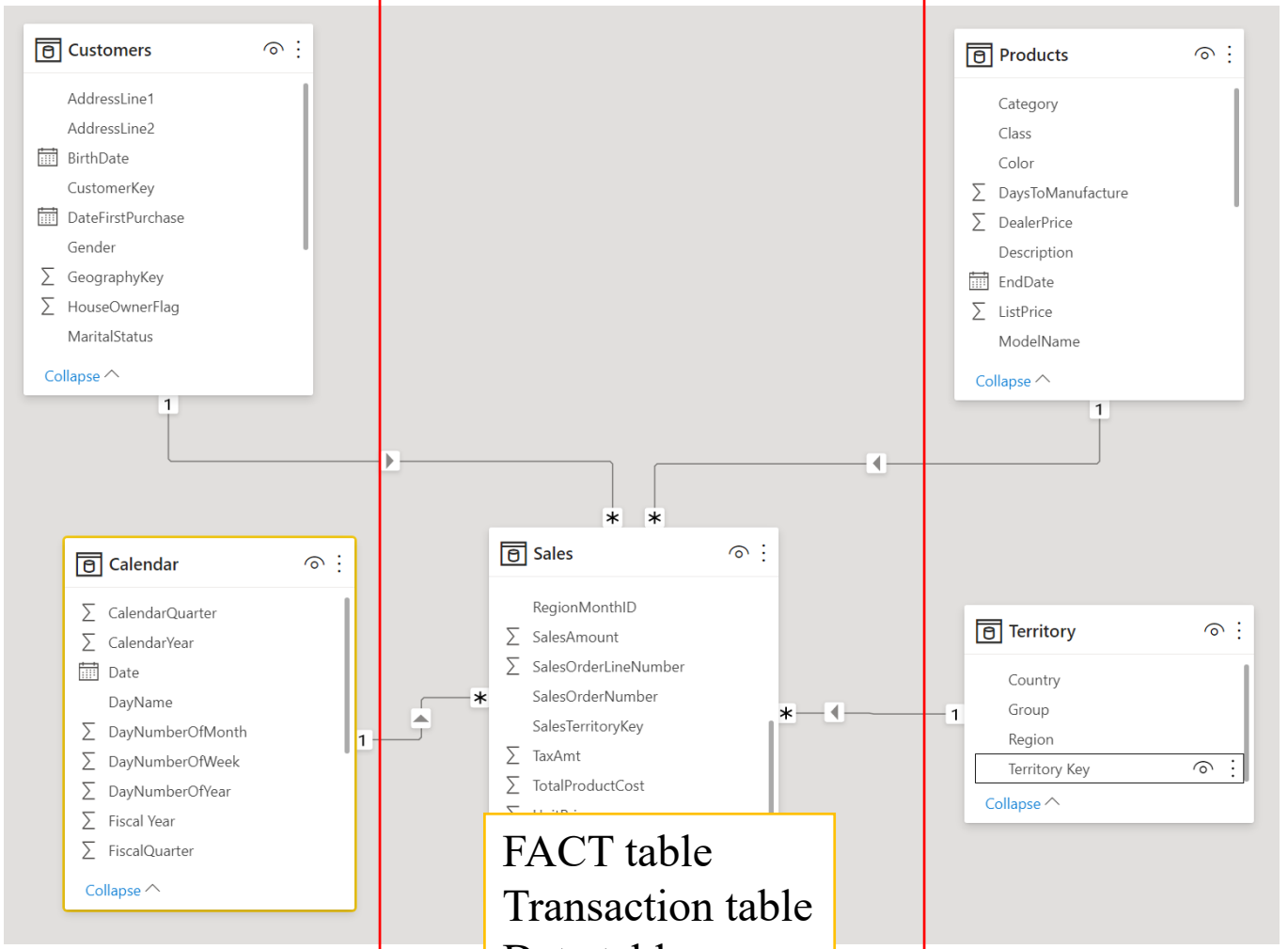
Add relationship between “Territories” and “Sales” using

- “Territory Key” in Table “Territories”
- “SalesTerritoryKey” in Sales



Type of Tables

DIM table
Master table
Lookup table



DIM table
Master table
Lookup table

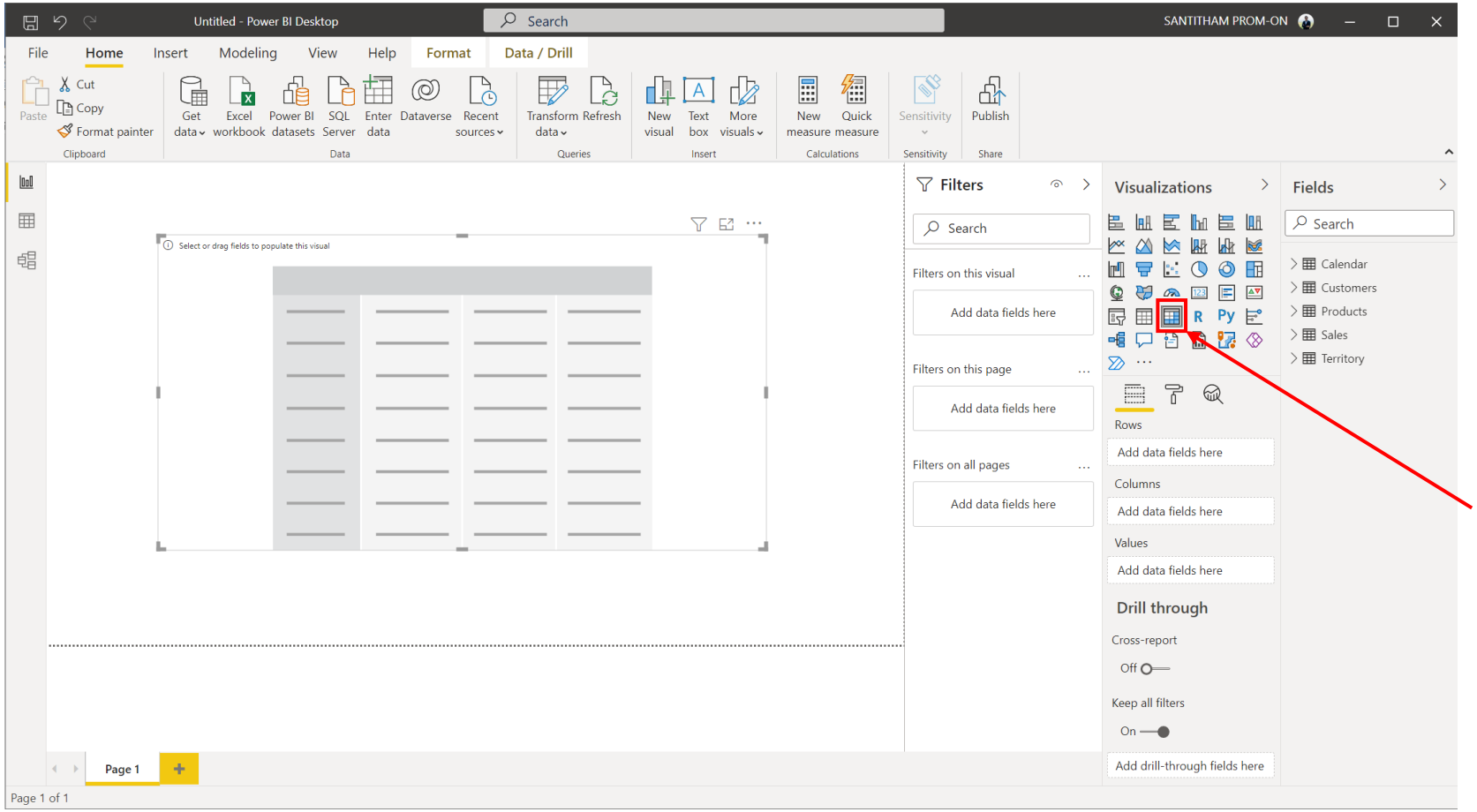
FACT table
Transaction table
Data table

M language and DAX (Data Analysis eXpression)

	Description	Where to Use
M	Power Query Formula Language	Data Preparation
	Data Transformation	Before Data Model
★ DAX	Data Analysis Expression Language	Create Insights
	Analytical Data Calculations	In Data Model
	Comparable to Excel Function	

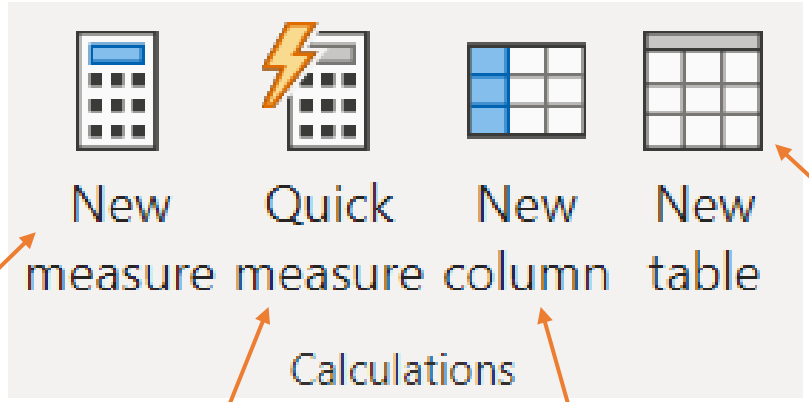
Best visual for developing DAX Matrix

Report View



Matrix Icon

Types of DAX based on output



Return a value

Build measure using Wizard

Return a column

Return a table

BIG DATA EXPERIENCE

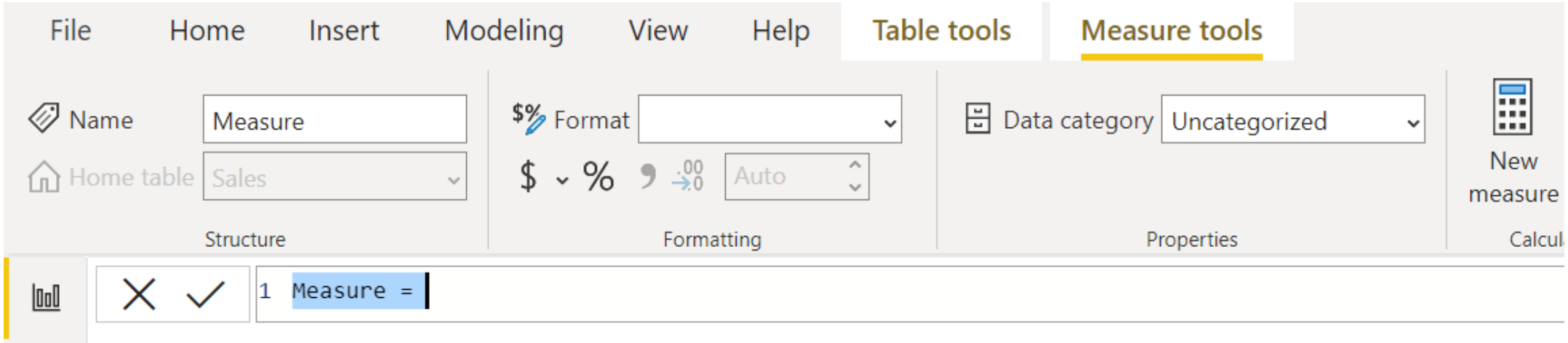
Measure

- Measures have been around for many years in the enterprise versions of Microsoft BI tools, such as SQL Server Analysis Services.
- Measures have now made it into the world of business users who want to create Power BI reports.
- There is nothing confusing or hard to learn about measures.
- A measure is simply a DAX formula that instructs Power BI to do a calculation on data.

KMUTT g-able

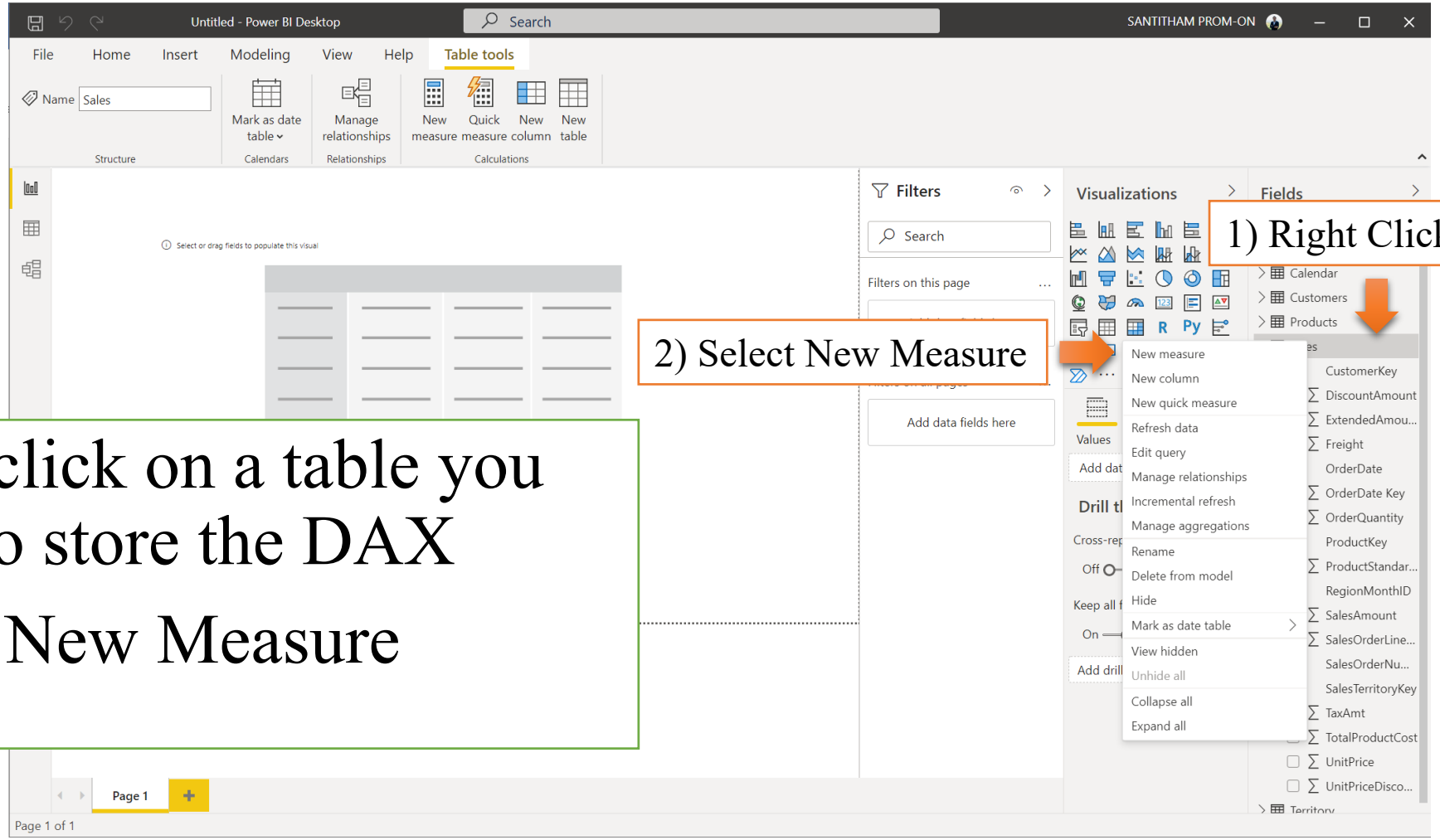
Technique for writing DAX measure

You can write measures in the formula bar



The formula bar is not visible unless you have a measure selected in the Fields pane on the right-hand side of Power BI.

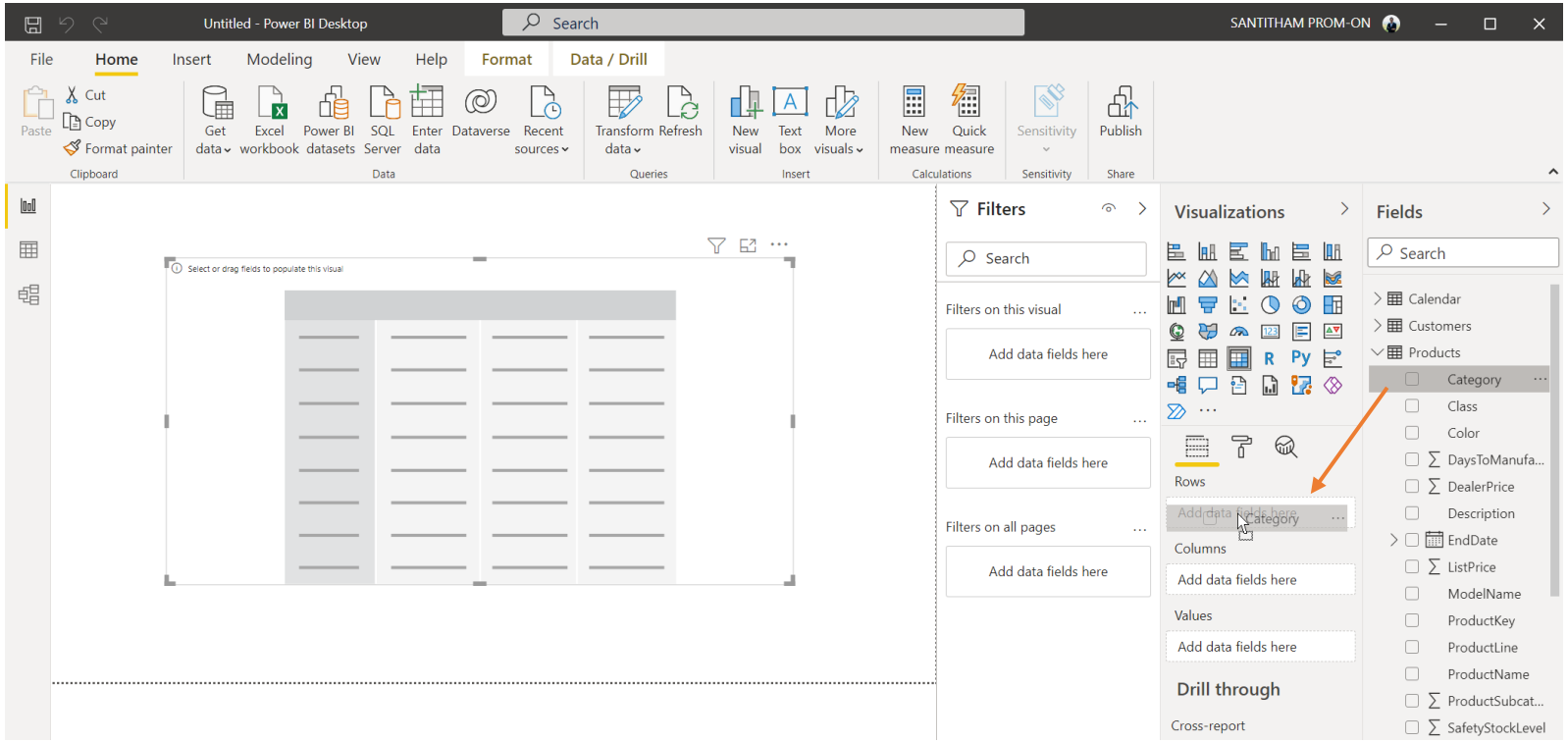
Best practice in creating DAX



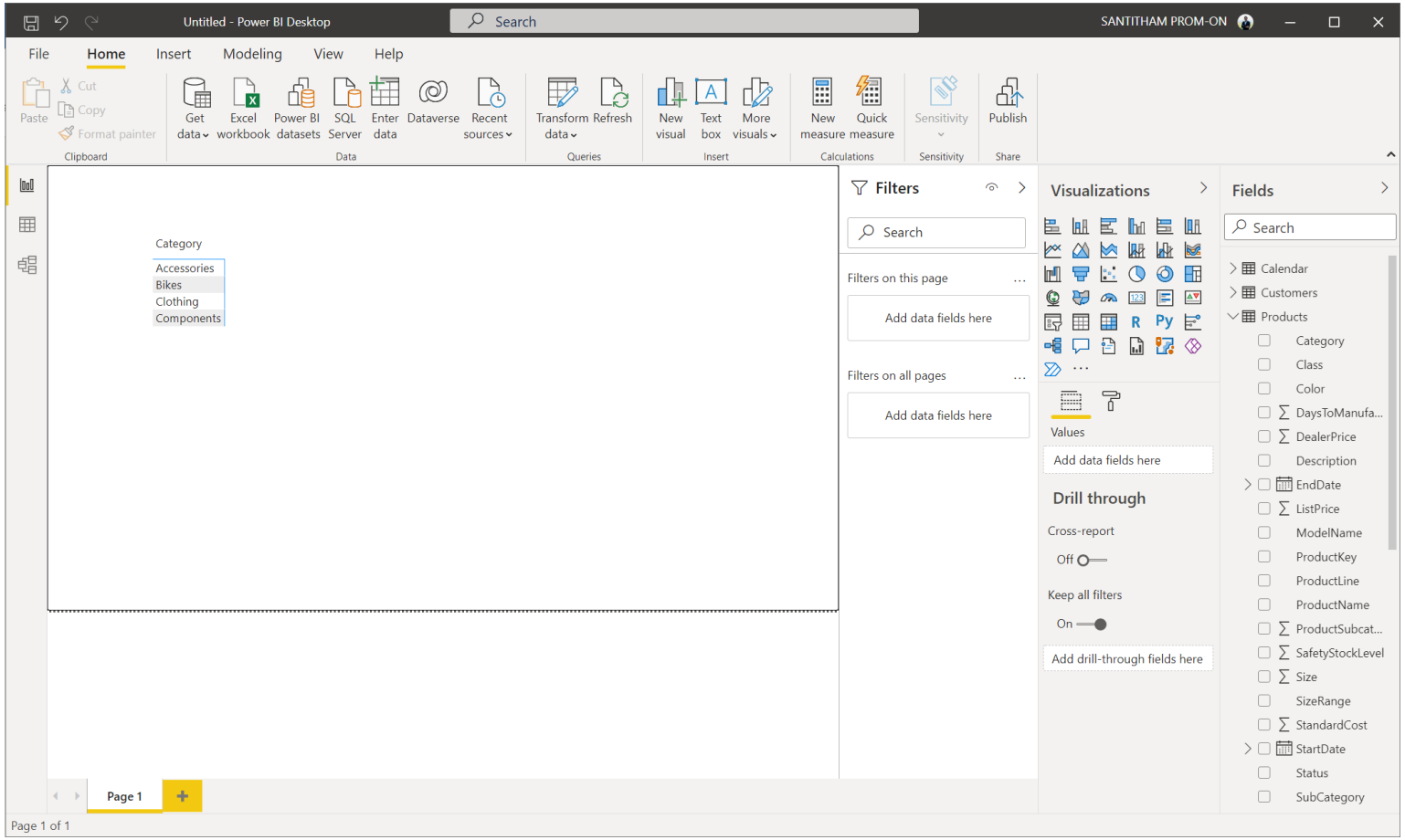
- Right click on a table you want to store the DAX
- Select New Measure

Let’s write our first DAX

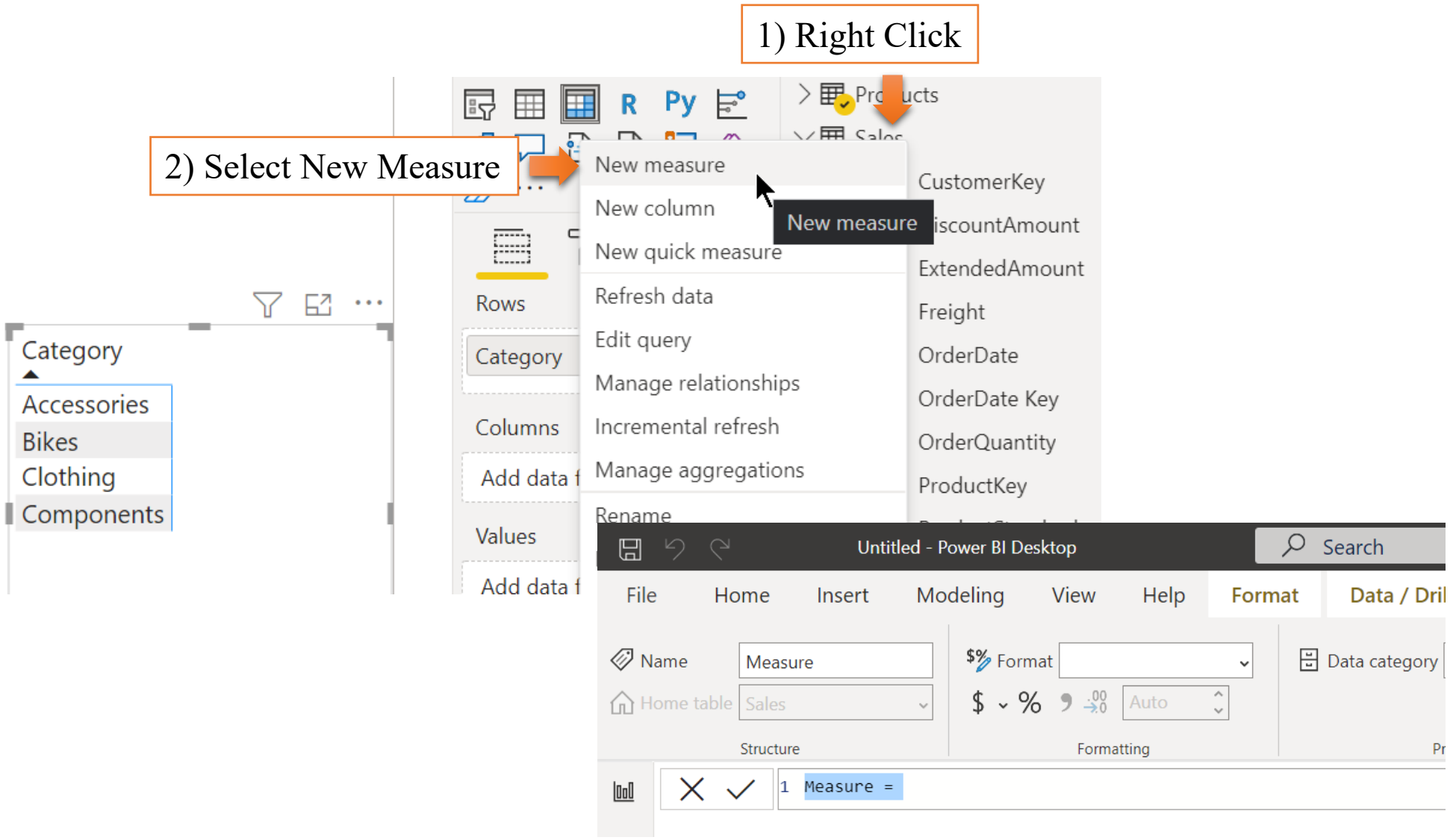
Build a matrix with Product[Category] at Rows



Matrix


Untitled - Power BI Desktop
Search
SANTITHAM PROM-ON
File
Home
Insert
Modeling
View
Help
Paste
Cut
Copy
Format painter
Clipboard
Get data
Excel workbook
Power BI datasets
SQL Server
Enter data
Dataverse
Recent sources
Data
Transform data
Refresh
Queries
New visual
Text box
More visuals
Insert
New measure
Quick measure
Calculations
Sensitivity
Sensitivity
Publish
Share
Category
Accessories
Bikes
Clothing
Components
Filters
Search
Filters on this page
Add data fields here
Filters on all pages
Add data fields here
Visualizations
Values
Add data fields here
Drill through
Cross-report
Off
Keep all filters
On
Add drill-through fields here
Fields
Search
Calendar
Customers
Products
Category
Class
Color
DaysToManufa...
DealerPrice
Description
EndDate
ListPrice
ModelName
ProductKey
ProductLine
ProductName
ProductSubcat...
SafetyStockLevel
Size
SizeRange
StandardCost
StartDate
Status
SubCategory
Page 1
Page 1 of 1

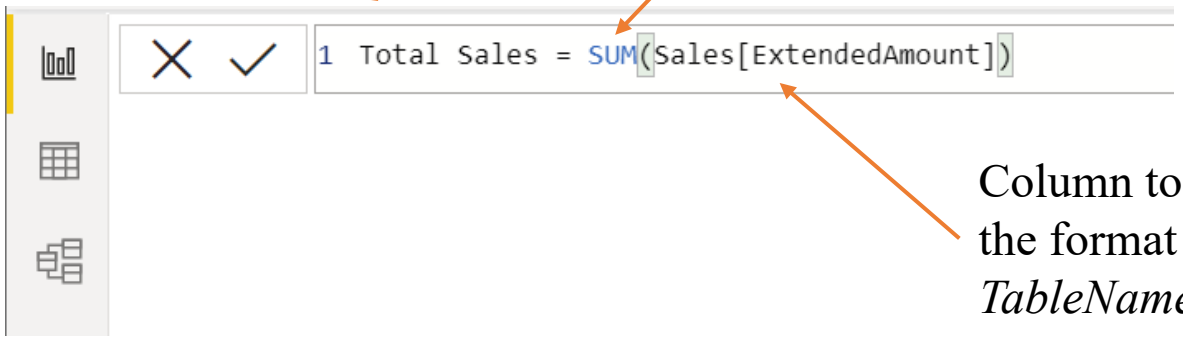
Add New measure



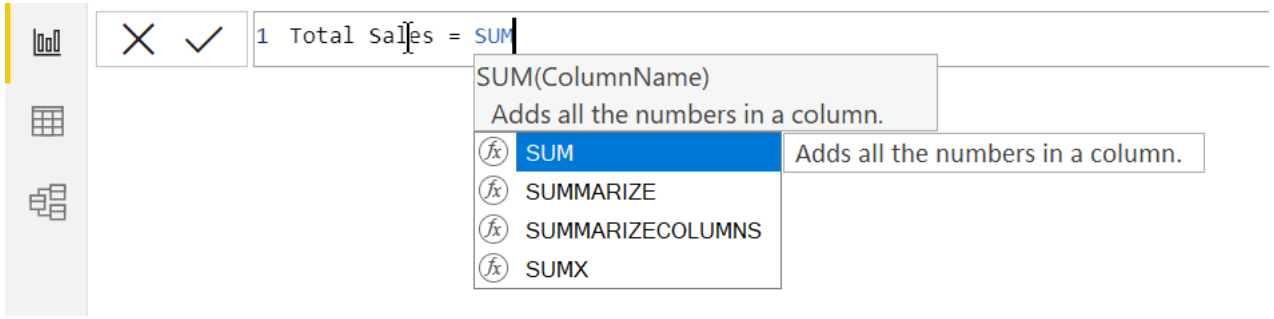
Add Total Sales

Measure Name

Aggregate Function

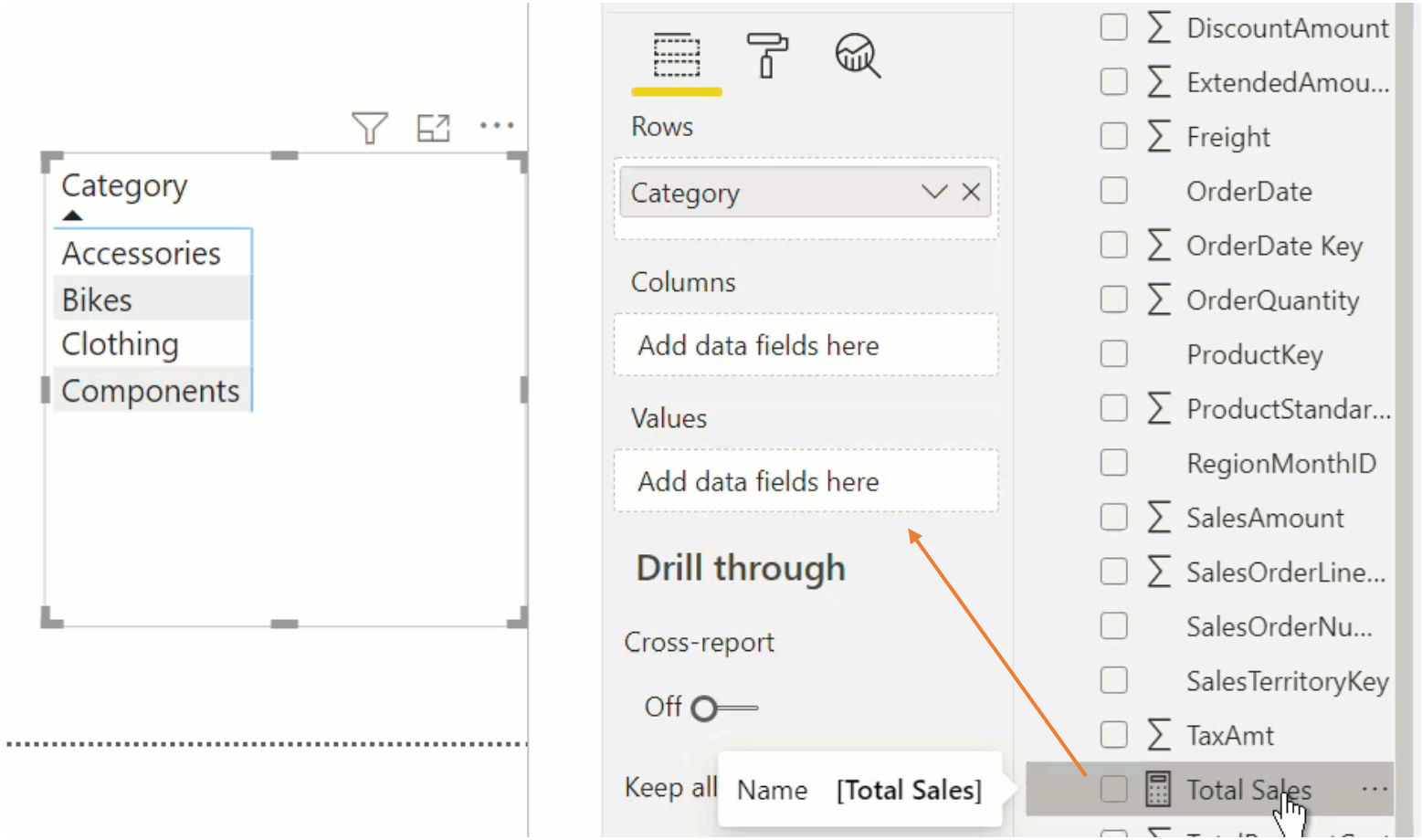


Column to be aggregated in the format *TableName*[*ColumnName*]

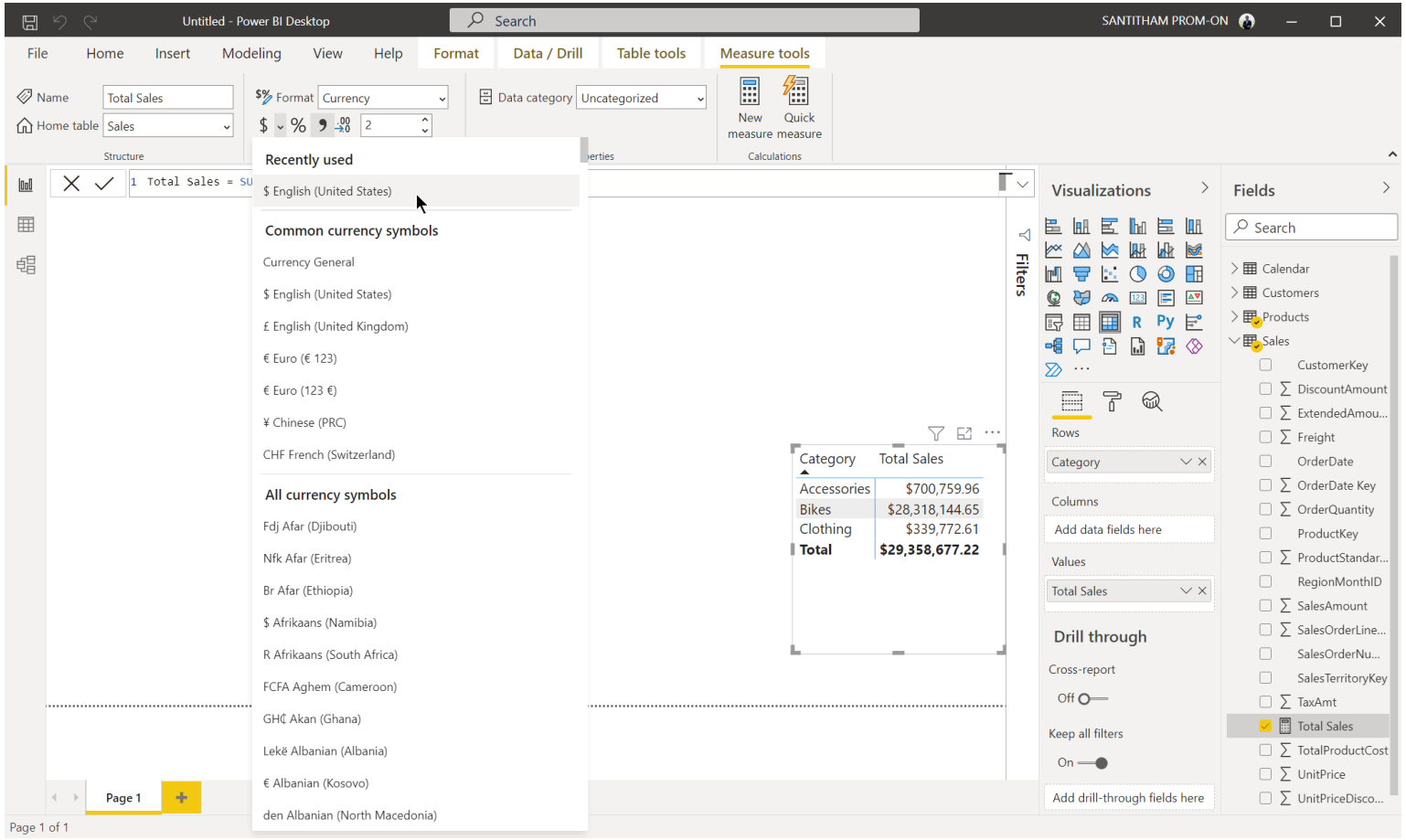


IntelliSense

Test your measure with Matrix

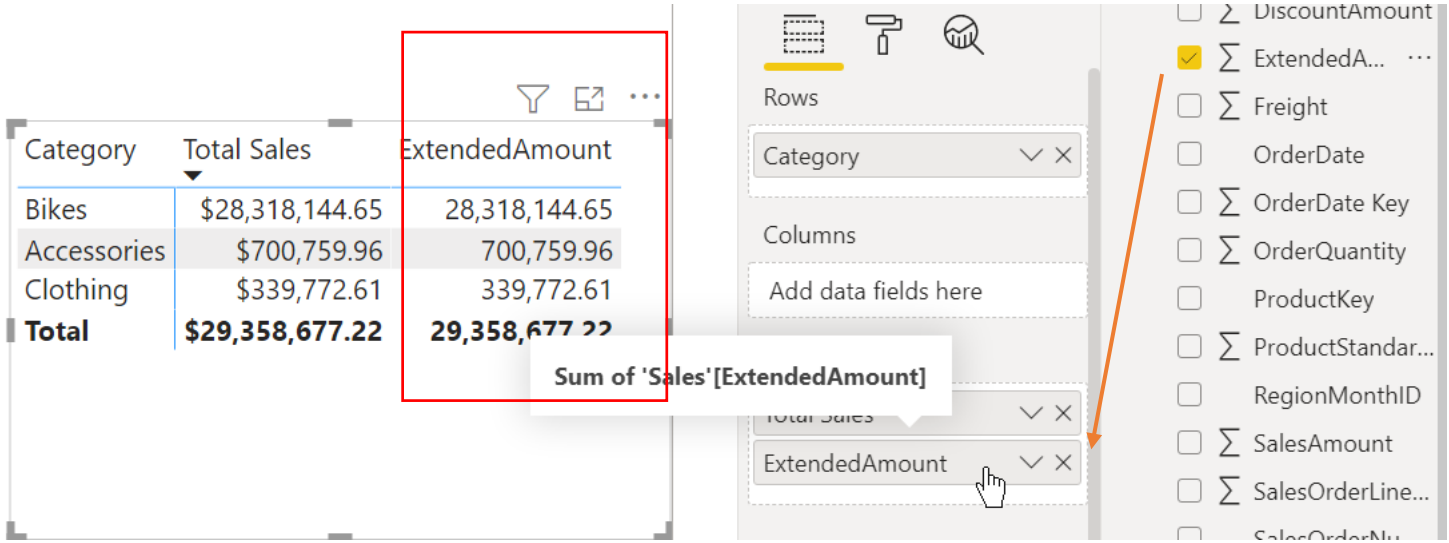


Format your DAX

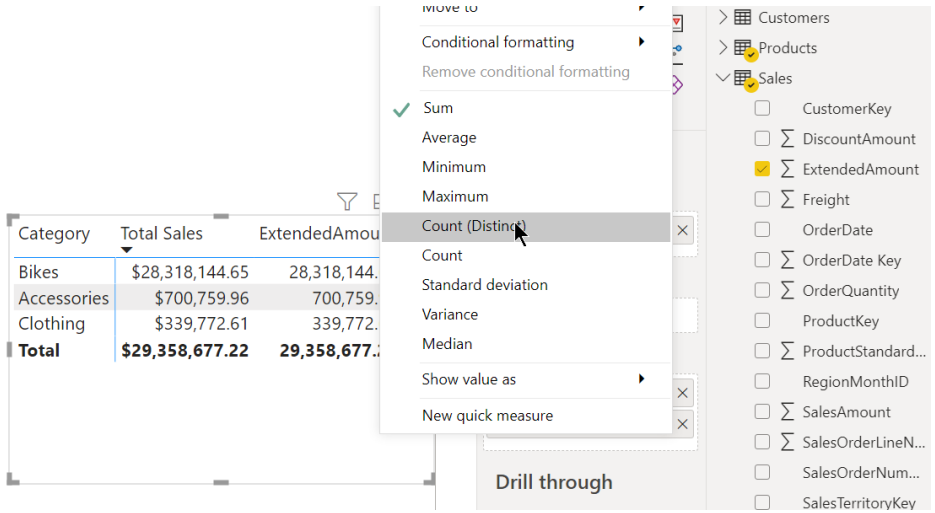


Avoid implicit measure

If you directly drag the Field to the Values, it will be automatically aggregated. (SUM as default)



Aggregation behavior can be changed by right clicking the Field



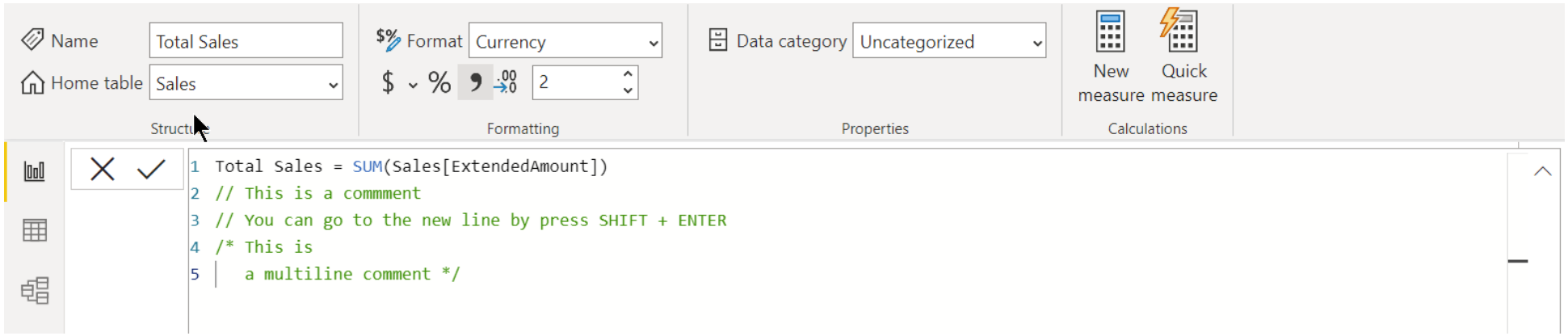
BIG DATA EXPERIENCE

Why implicit measure should be avoided?

- Name is not helpful. Need additional steps in renaming.
- No formatting is applied. While this can be done in the source field, you cannot have different formatting for different visuals.
- You cannot reference implicit measures in other measures.
- You will not learn how to write DAX

KMUTT g-able

Comments



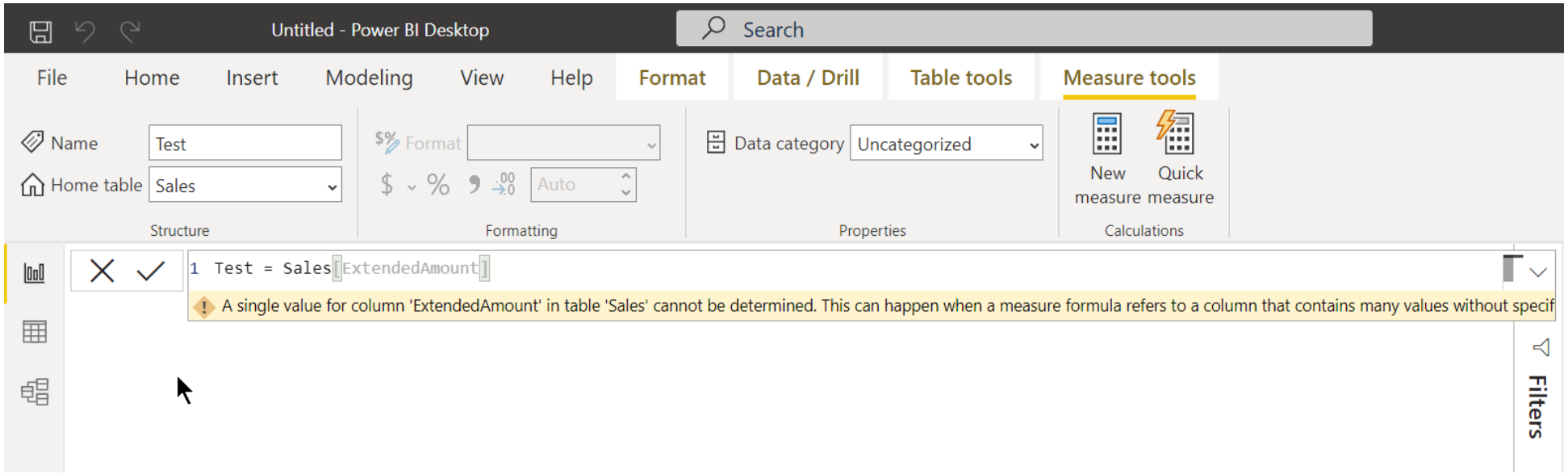
To write go to the new line

SHIFT + ENTER

To add comment

Use either // or /* */

Measure with Naked columns do not work



Expect a single value ≠ Return a column

DAX, Aggregation: SUM Practice

- [Total Sales], you can use the ExtendedAmount in the Sales table
- [Total Cost], you can use any product cost in Sales table
- [Total Tax Paid], you can use the tax column in Sales table
- [Total Order Quantity], you can use the quantities in Sales table

BIG DATA EXPERIENCE

Check Point – Modeler Habits

- Did you create a matrix and put Product[Category] on Rows in your matrix?
- Did you right-click the Sales table in the Fields pane and select New Measure to start the process?
- Did you reference all columns in your measures in the format *TableName*[*ColumnName*]
- Did you apply formatting to your measure immediately after your write it?
- Did you use the keyboard and look at the IntelliSense and you typed the measures?

KMUTT g-able

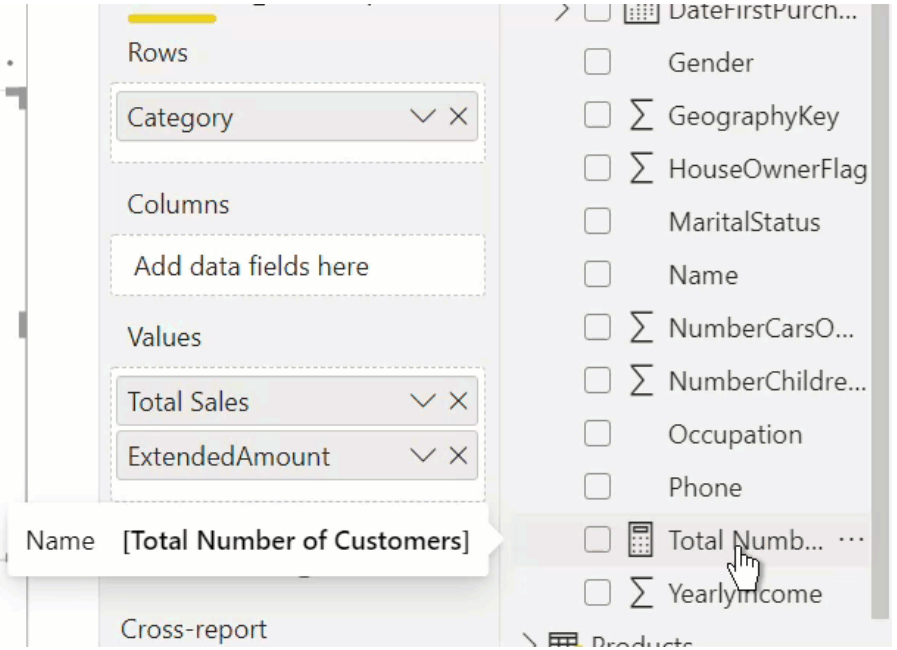
DAX, Aggregation: COUNT() Practice

- [Total Number of Products], use the product table to count how many products there are
- [Total Number of Customers], use the customer name instead of customer key

Total Number of Customers

```
1  Total Number of Customers = COUNT(Customers[CustomerKey])
```

Category	Total Sales	ExtendedAmount
Bikes	$28,318,144.65	28,318,144.65
Accessories	$700,759.96	700,759.96
Clothing	$339,772.61	339,772.61
Total	**$29,358,677.22**	**29,358,677.22**



BIG DATA EXPERIENCE

Why?

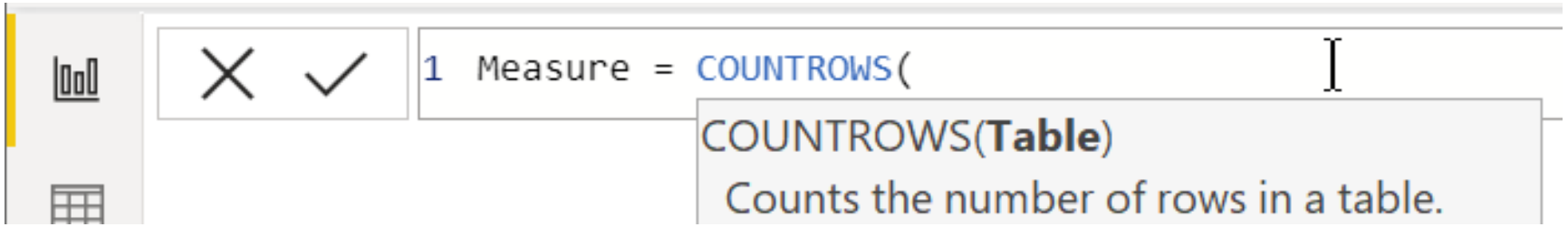
Category	mber of Customers	Total Sales	ExtendedAmount
Bikes	18484	$28,318,144.65	28,318,144.65
Accessories	18484	$700,759.96	700,759.96
Clothing	18484	$339,772.61	339,772.61
Components	18484		
Total	**18484**	**$29,358,677.22**	**29,358,677.22**

Total number of customers

Total sales of each Category

DAX, Aggregation: COUNTROWS()

- COUNTROWS[Table] counts the number of rows in a table

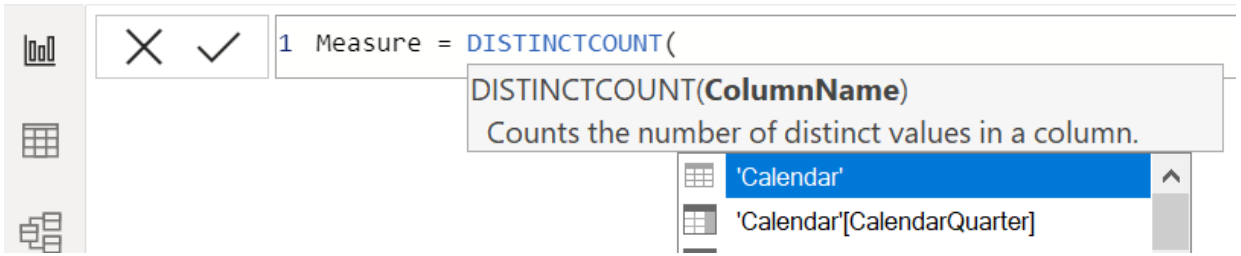


Practice

- [Total Number of Products COUNTROW]
- [Total Number of Customers COUNTROW]

DAX, Aggregation: DISTINCTCOUNT()

- Count the number of distinct values in a column.



Practice

- Use occupation instead of category
- [Total Customers DISTINCTCOUNT], should we use customer key or name?
- [Count of Country], use Territories table and count the number of countries in each Group
- [Total Customers That Have Purchased], use subcategories as Row Header, count the distinct customers purchased products in each subcategories

DAX, Aggregation: MAX(), MIN(), AVERAGE()

Use Product[Category] as Row header

- [Maximum Tax Paid on a Product] use a suitable column from the Sales table along with MAX() function
- [Minimum Price Paid for a Product] use a suitable column from the Sales table along with MIN() function
- [Average Price Paid for a Product] use a suitable column from the Sales table along with AVERAGE()

BIG DATA EXPERIENCE

Expected output

Category	Maximum Tax Paid on a Product	Minimum Price Paid for a Product	Average Price Paid for a Product
Accessories	$12.72	$2.29	$19.42
Bikes	$286.26	$539.99	$1,862.42
Clothing	$5.60	$8.99	$37.33
Total	**$286.26**	**$2.29**	**$486.09**

DAX, Aggregation: COUNTBLANK()

- Count the number of blanks in a column
- Format

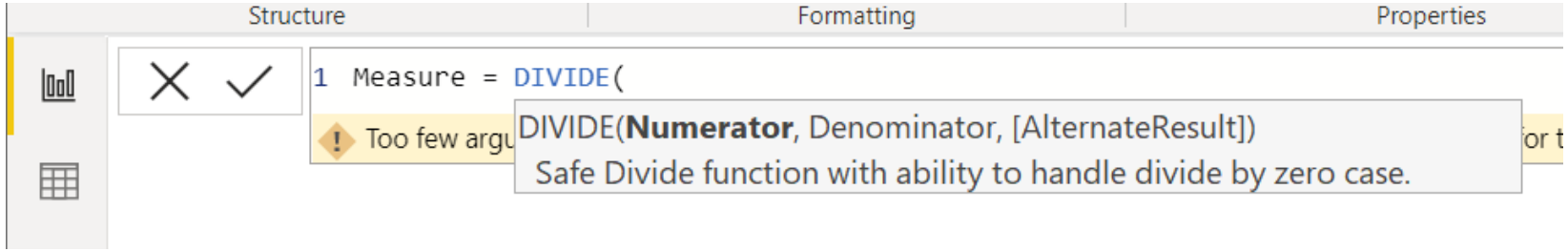
    COUNTBLANK(ColumnName)

Practice

- [Total Customers Without Address Line 2]
- [Total Products Without Weight Values]

DAX, Operator: DIVIDE()

- Safe divide



Practice

- [Margin %], [Total Margin $] divided by [Total Sales]
  - Total Margin $ is the profit
- [Tax %], [Total Tax] divided by [Total Sales]

Query Evaluation

- There are 2 distinct engines that work together to process every DAX query: Formula Engine and Storage Engine

Formula Engine

Receives, interpret and executes all DAX request

Process DAX query and generate **query plan**

Work with **datacache** to evaluate DAX query and return it

Storage Engine

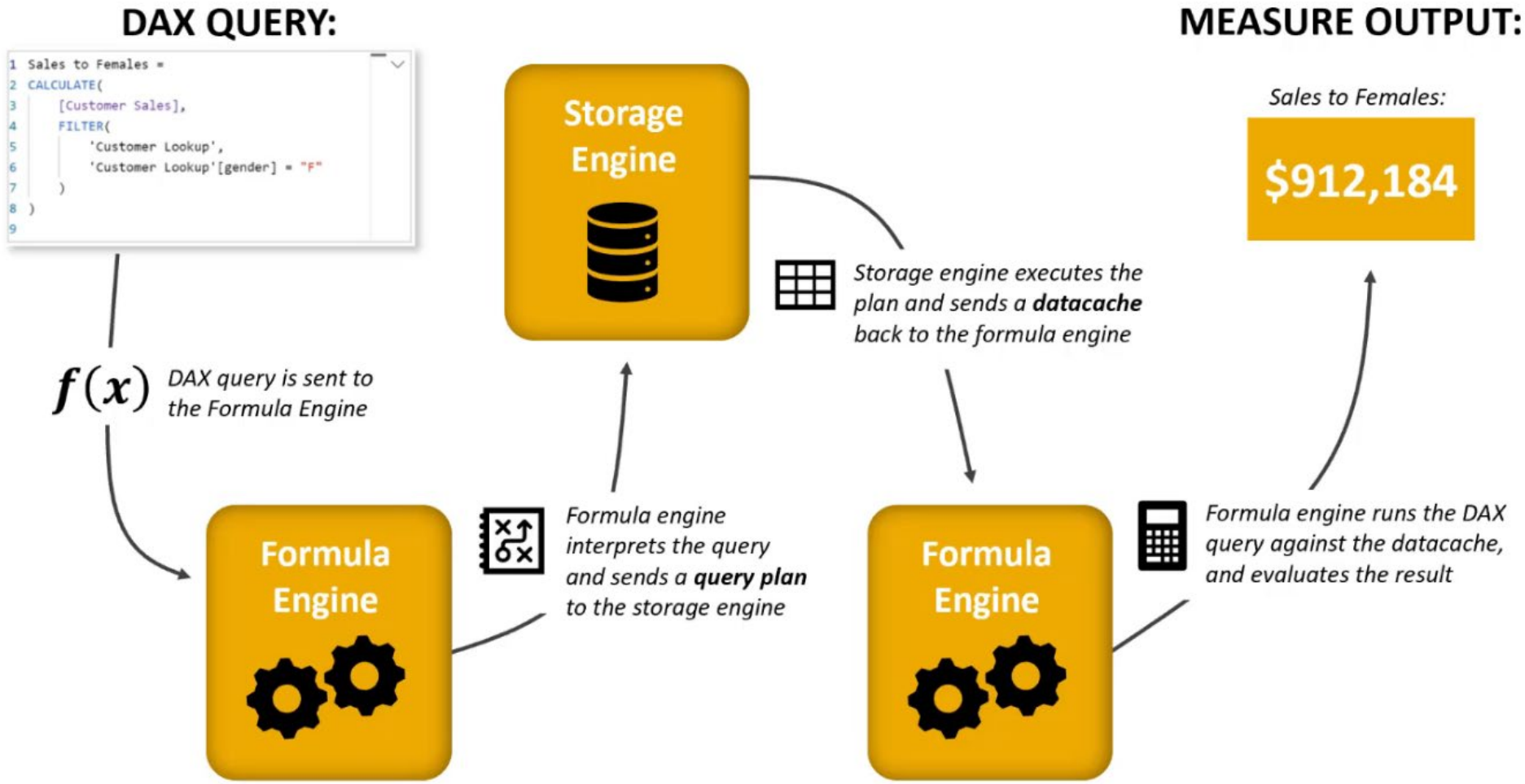
Compresses and encodes raw data and only communicate with Formula engine

Receive a query plan from Formula Engine, executes it and return **datacache**

There are two types of Storage Engine

- **VertiPaq** is used for in-memory data (import mode)
- **DirectQuery** is used for data read directly from the source (e.g. MSSQL)

DAX Query Evaluation



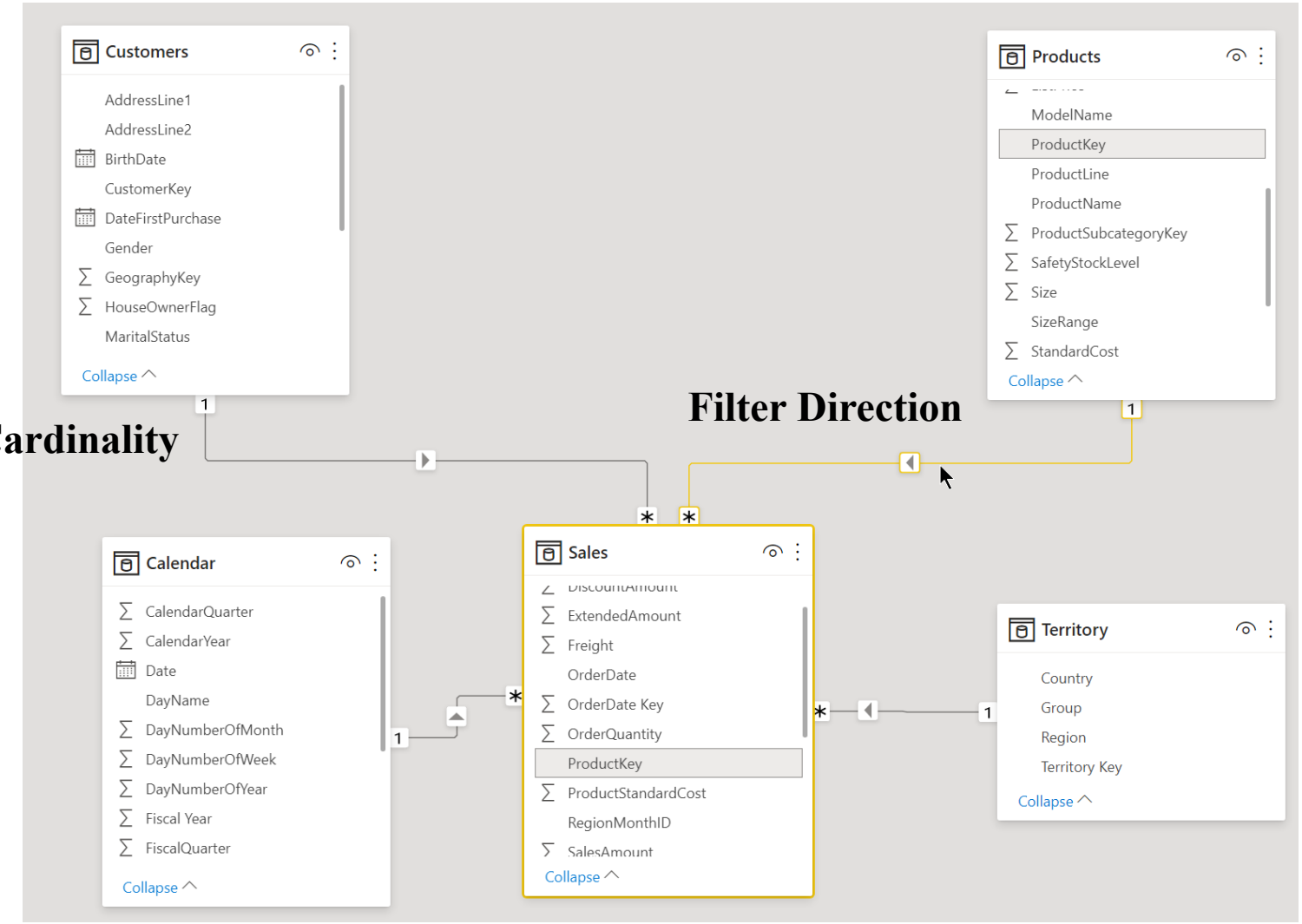
BIG DATA EXPERIENCE

Part 2

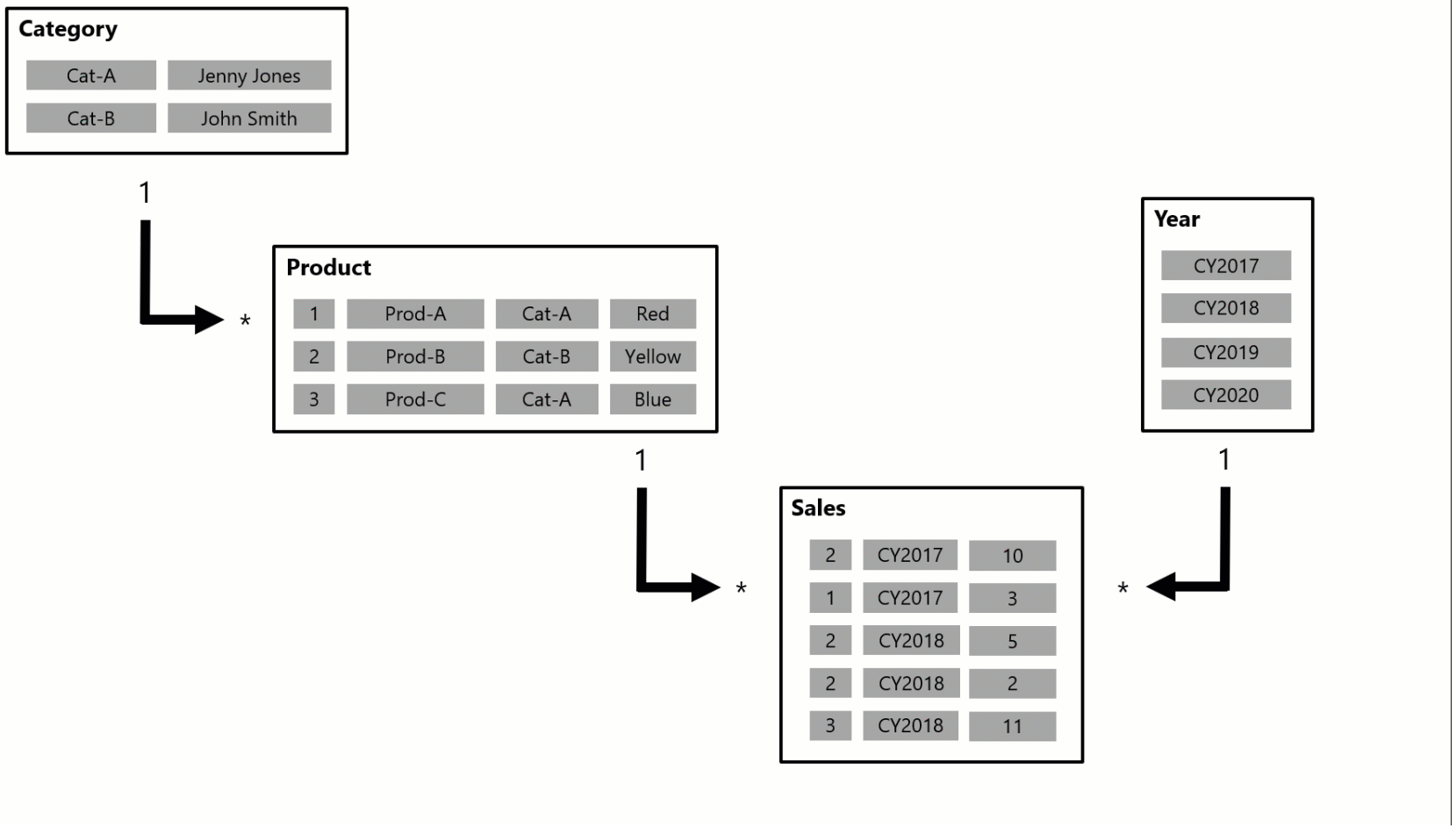
- Relationship and cardinality
- Filter propagation
- DAX: Calculated columns
- DAX: RELATED()
- DAX: Iterator
- DAX: Calculate

KM UIT g-able

Model View


Customers
AddressLine1
AddressLine2
BirthDate
CustomerKey
DateFirstPurchase
Gender
GeographyKey
HouseOwnerFlag
MaritalStatus
Collapse
Products
ModelName
ProductKey
ProductLine
ProductName
ProductSubcategoryKey
SafetyStockLevel
Size
SizeRange
StandardCost
Collapse
Cardinality
Filter Direction
1
1
Calendar
CalendarQuarter
CalendarYear
Date
DayName
DayNumberOfMonth
DayNumberOfWeek
DayNumberOfYear
Fiscal Year
FiscalQuarter
Collapse
Sales
ExtendedAmount
Freight
OrderDate
OrderDate Key
OrderQuantity
ProductKey
ProductStandardCost
RegionMonthID
SalesAmount
Collapse
Territory
Country
Group
Region
Territory Key
Collapse

Filter propagation



Cross Filter Direction

- One-to-many (1:*)
- Many-to-one (*:1)
- One-to-one (1:1)
- Many-to-many (*:*)

Cardinality type	Cross filter options
One-to-many (or Many-to-one)	Single Both
One-to-one	Both
Many-to-many	Single (Table1 to Table2) Single (Table2 to Table1) Both

- *Single* cross filter direction means "single direction“
- *Both* means "both directions". aka *bi-directional*.
- For One-to-many relationships, the cross filter direction is always from the "one" side, and optionally from the "many" side (bi-directional).
- For One-to-one relationships, the cross filter direction is always from both tables.
- For the Many-to-many relationships, cross filter direction can be from either one of the tables, or from both tables.

Initial Filter

Category	Total Number of Customers	Total Number of Products
Accessories	18,484	35
Bikes	18,484	125
Clothing	18,484	48
Components	18,484	189
Total	**18,484**	**397**

The matrix first “filters” the data model (in this case, it is the Products[Category] column), and then Power BI evaluates the measures for each row in the matrix with the filters applied

Let’s call it “Initial Filter”

Initial Filters

Rows (#1)

Columns (#2)

Slicers (#3)

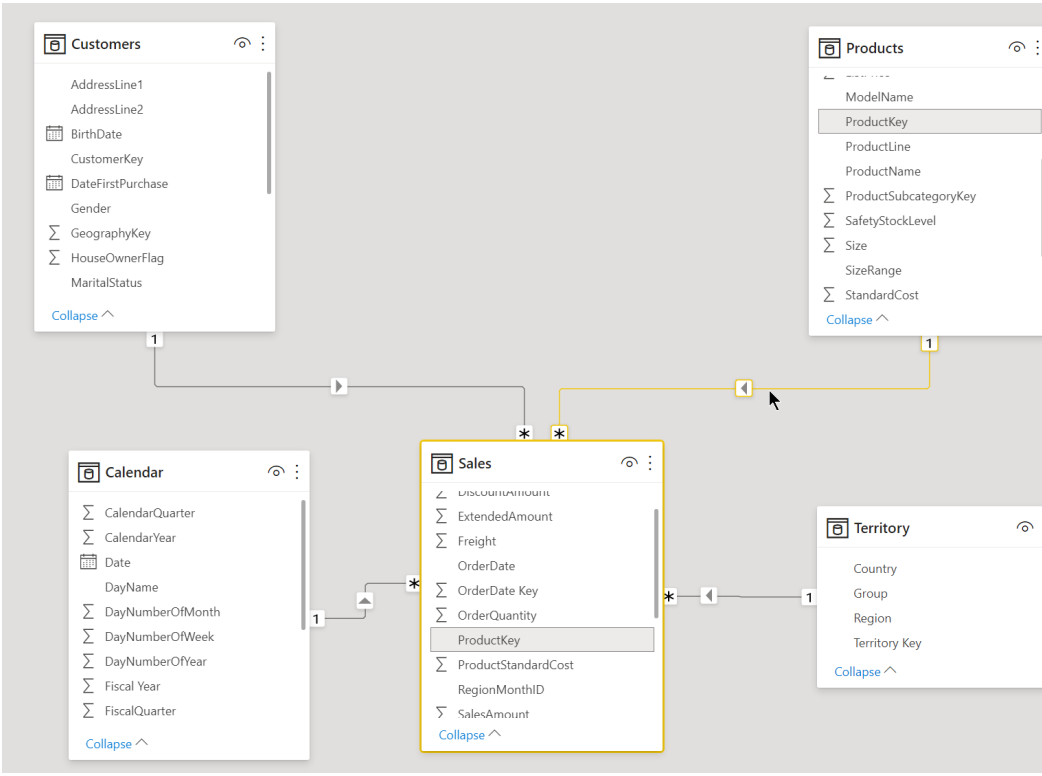
Filters (#4)

Let's add [Total Number of Products] to table Product

```
1 Total Number of Product = DISTINCTCOUNT(Products[ProductKey])
```

Category	Total Number of Customers	Total Number of Product	Total Sales
Accessories	18484	35	$700,759.96
Bikes	18484	125	$28,318,144.65
Clothing	18484	48	$339,772.61
Components	18484	189	
Total	**18484**	**397**	**$29,358,677.22**

Filter Propagation



Not related to initial filter

Initial Filter

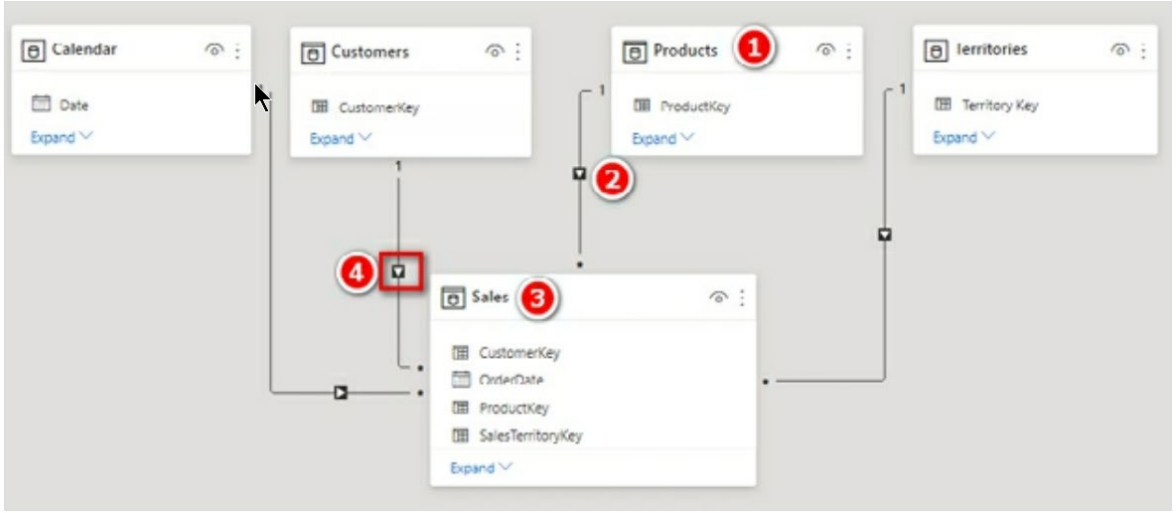
Category	Total Number of Customers	Total Number of Product	Total Sales
Accessories	18484	35	$700,759.96
Bikes	18484	125	$28,318,144.65
Clothing	18484	48	$339,772.61
Components	18484	189	
Total	**18484**	**397**	**$29,358,677.22**

In the same table as initial filer

Cross-filter direction from Product to Sales

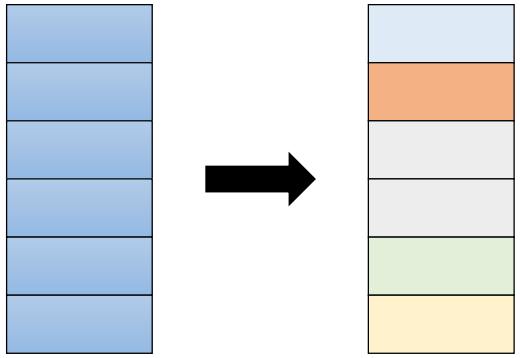
1. The initial filter is applied to the table(s). In this example, Products[Category] = "Clothing".
2. This filter automatically propagates through the relationships that exist between the tables, flowing in the direction of the arrow to the connected tables.
3. The connected table(s) (Sales in this example) is then also filtered so that the same products in the Products table will remain in the Sales table.
4. The filter applied to the Sales table does not automatically flow back uphill to the Customers table

Category	Total Number of Customers	Total Number of Products
Accessories	18,484	35
Bikes	18,484	125
Clothing	18,484	48
Components	18,484	189
Total	**18,484**	**397**

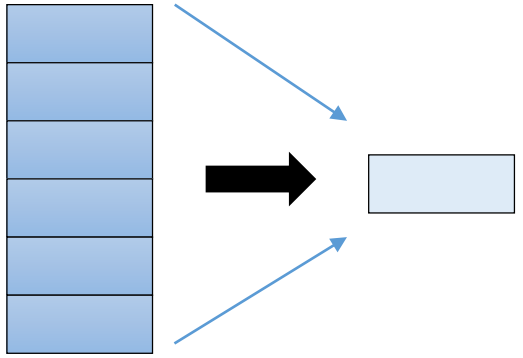


BIG DATA EXPERIENCE

New Column



New Measure



KMUTT g-able

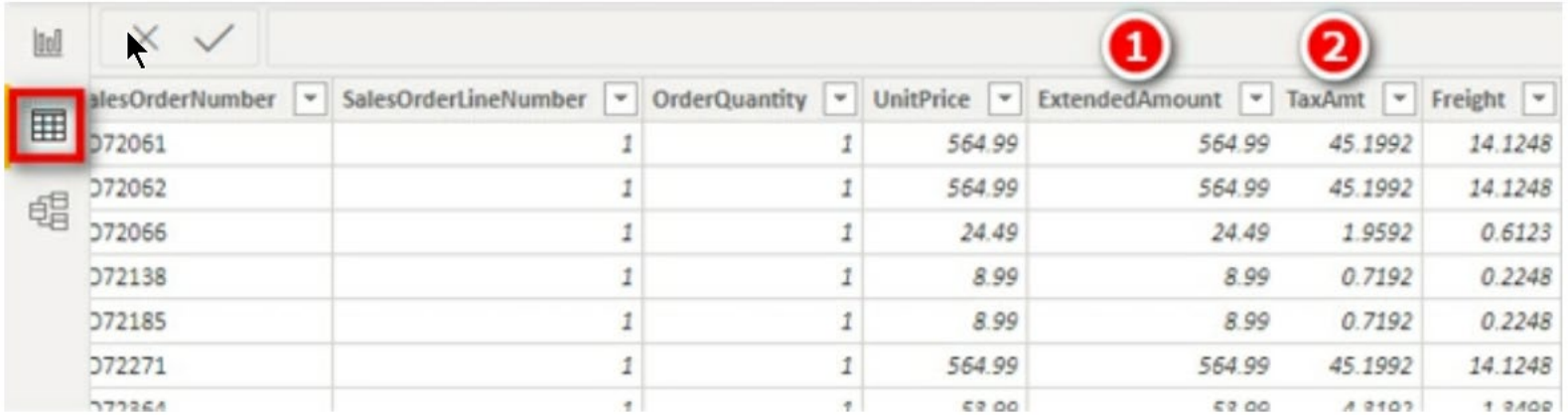
BIG DATA EXPERIENCE

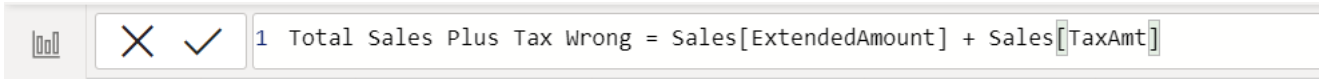
Calculated columns

- Row-by-row evaluation
- Add column to data model (consume more memory)

KMUTT g-able

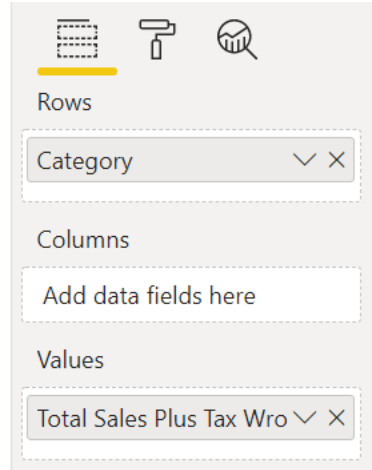
Basic (but wrong) calculated columns





ExtendedAmount	TaxAmt	Total Sales Plus Tax Column
539.99	43.1992	$583.19
159	12.72	$171.72
120	9.6	$129.60
1 742.35	2 59.388	$801.74
769.49	61.5592	$831.05
564.99	45.1992	$610.19

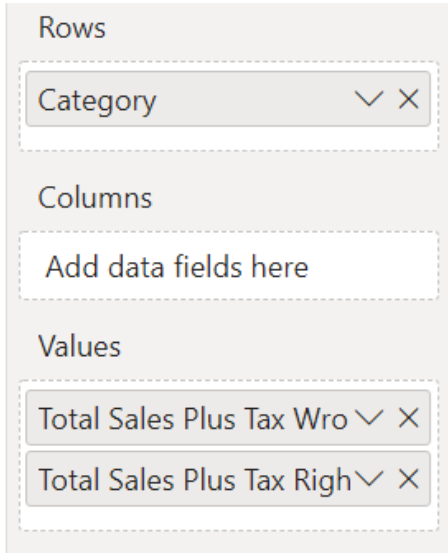
Category	Total Sales Plus Tax Wrong
Bikes	30,583,596.28
Accessories	756,820.76
Clothing	366,954.42
Total	**31,707,371.45**



Use New Measure Instead

```
1  Total Sales Plus Tax Right = SUM(Sales[ExtendedAmount]) + SUM(Sales[TaxAmt])
```

Category	Total Sales Plus Tax Wrong	Total Sales Plus Tax Right
Bikes	30,583,596.28	30,583,596.28
Accessories	756,820.76	756,820.76
Clothing	366,954.42	366,954.42
Total	**31,707,371.45**	**31,707,371.45**



BIG DATA EXPERIENCE

Big Problem with Calculated Columns

- Calculated columns make your tables and workbooks bigger.
- Mostly (though not always) you can get the same outcome by using a measure, which is a better way.

You should not use calculated columns if you can

- Use a measure instead
- Bring the data into a table directly from the source data
- Create the column during data load by using Power Query

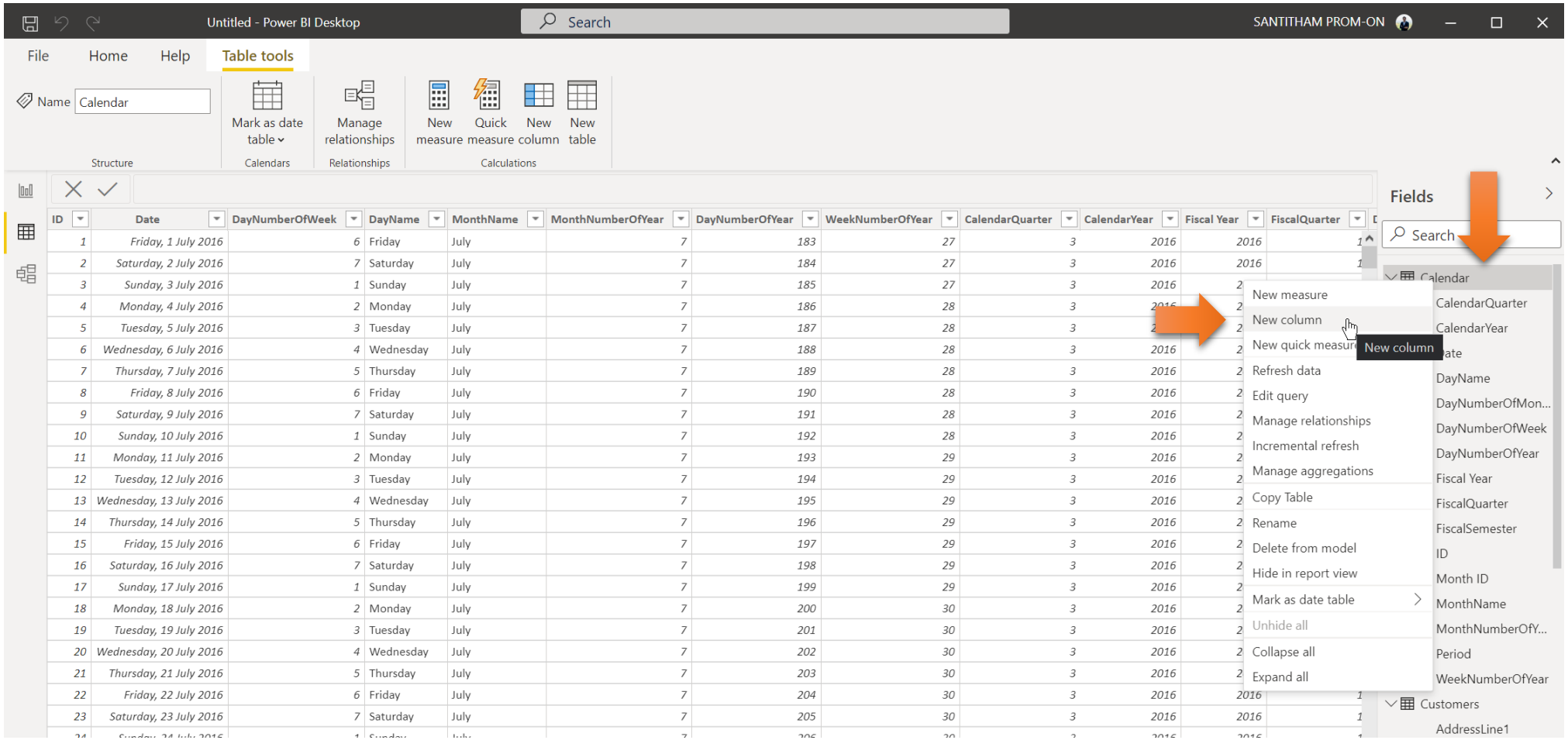
KM UTT g·able

BIG DATA EXPERIENCE

When to use calculated columns

- You need to filter/slice a visual based on the results of a column (i.e., you want to use the column on Filters, Slicer, Rows, or Columns). Measures don't work in this case.
- You can't bring the column of data you need in from your source data or by using Power Query (for whatever reason).

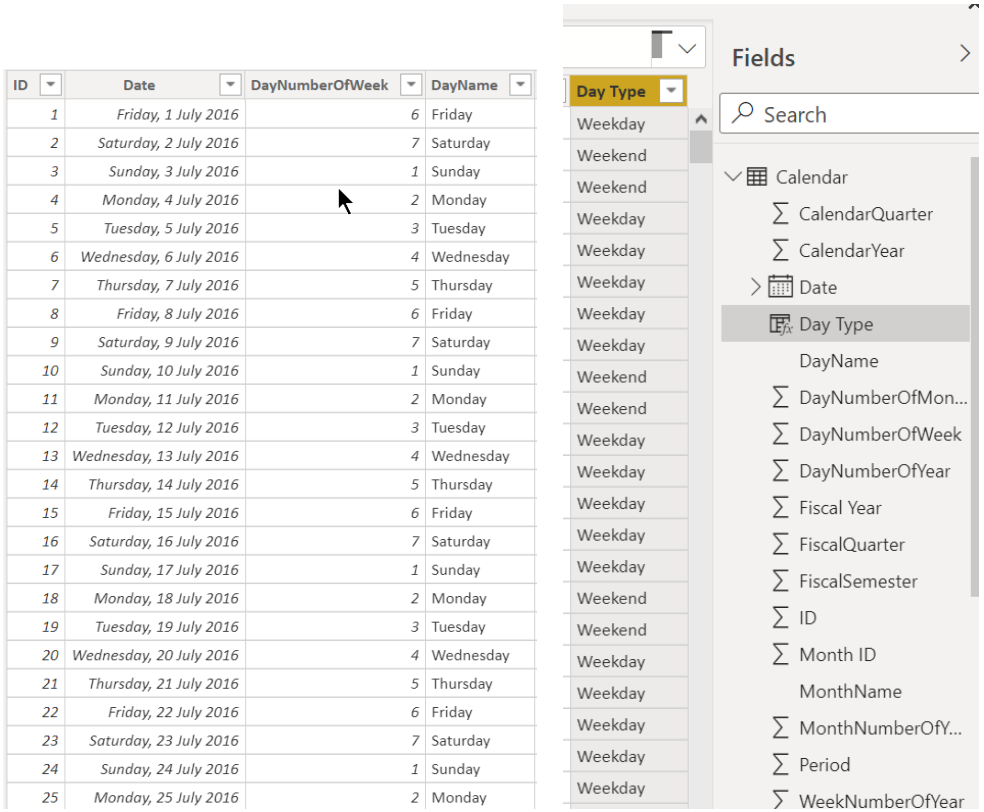
Example: Day Type Calculated Column (1)



ID	Date	DayNumberOfWeek	DayName	MonthName	MonthNumberOfYear	DayNumberOfYear	WeekNumberOfYear	CalendarQuarter	CalendarYear	Fiscal Year	FiscalQuarter
1	Friday, 1 July 2016	6	Friday	July	7	183	27	3	2016	2016	1
2	Saturday, 2 July 2016	7	Saturday	July	7	184	27	3	2016	2016	1
3	Sunday, 3 July 2016	1	Sunday	July	7	185	27	3	2016	2	
4	Monday, 4 July 2016	2	Monday	July	7	186	28	3	2016	2	
5	Tuesday, 5 July 2016	3	Tuesday	July	7	187	28	3	2	2	
6	Wednesday, 6 July 2016	4	Wednesday	July	7	188	28	3	2016	2	
7	Thursday, 7 July 2016	5	Thursday	July	7	189	28	3	2016	2	
8	Friday, 8 July 2016	6	Friday	July	7	190	28	3	2016	2	
9	Saturday, 9 July 2016	7	Saturday	July	7	191	28	3	2016	2	
10	Sunday, 10 July 2016	1	Sunday	July	7	192	28	3	2016	2	
11	Monday, 11 July 2016	2	Monday	July	7	193	29	3	2016	2	
12	Tuesday, 12 July 2016	3	Tuesday	July	7	194	29	3	2016	2	
13	Wednesday, 13 July 2016	4	Wednesday	July	7	195	29	3	2016	2	
14	Thursday, 14 July 2016	5	Thursday	July	7	196	29	3	2016	2	
15	Friday, 15 July 2016	6	Friday	July	7	197	29	3	2016	2	
16	Saturday, 16 July 2016	7	Saturday	July	7	198	29	3	2016	2	
17	Sunday, 17 July 2016	1	Sunday	July	7	199	29	3	2016	2	
18	Monday, 18 July 2016	2	Monday	July	7	200	30	3	2016	2	
19	Tuesday, 19 July 2016	3	Tuesday	July	7	201	30	3	2016	2	
20	Wednesday, 20 July 2016	4	Wednesday	July	7	202	30	3	2016	2	
21	Thursday, 21 July 2016	5	Thursday	July	7	203	30	3	2016	2	
22	Friday, 22 July 2016	6	Friday	July	7	204	30	3	2016	2016	1
23	Saturday, 23 July 2016	7	Saturday	July	7	205	30	3	2016	2016	1

Example: Day Type Calculated Column (1)

```
Day Type = IF( 'Calendar'[DayNumberOfWeek] = 1 ||
'Calendar'[DayNumberOfWeek]=7, "Weekend","Weekday" )
```



ID	Date	DayNumberOfWeek	DayName	Day Type
1	Friday, 1 July 2016	6	Friday	Weekday
2	Saturday, 2 July 2016	7	Saturday	Weekend
3	Sunday, 3 July 2016	1	Sunday	Weekend
4	Monday, 4 July 2016	2	Monday	Weekday
5	Tuesday, 5 July 2016	3	Tuesday	Weekday
6	Wednesday, 6 July 2016	4	Wednesday	Weekday
7	Thursday, 7 July 2016	5	Thursday	Weekday
8	Friday, 8 July 2016	6	Friday	Weekday
9	Saturday, 9 July 2016	7	Saturday	Weekend
10	Sunday, 10 July 2016	1	Sunday	Weekend
11	Monday, 11 July 2016	2	Monday	Weekday
12	Tuesday, 12 July 2016	3	Tuesday	Weekday
13	Wednesday, 13 July 2016	4	Wednesday	Weekday
14	Thursday, 14 July 2016	5	Thursday	Weekday
15	Friday, 15 July 2016	6	Friday	Weekday
16	Saturday, 16 July 2016	7	Saturday	Weekend
17	Sunday, 17 July 2016	1	Sunday	Weekend
18	Monday, 18 July 2016	2	Monday	Weekday
19	Tuesday, 19 July 2016	3	Tuesday	Weekday
20	Wednesday, 20 July 2016	4	Wednesday	Weekday
21	Thursday, 21 July 2016	5	Thursday	Weekday
22	Friday, 22 July 2016	6	Friday	Weekday
23	Saturday, 23 July 2016	7	Saturday	Weekend
24	Sunday, 24 July 2016	1	Sunday	Weekend
25	Monday, 25 July 2016	2	Monday	Weekday

BIG DATA EXPERIENCE

Different forms of logical expression

- Day Type = IF('Calendar'[DayNumberOfWeek] = 1 || 'Calendar'[DayNumberOfWeek]=7, "Weekend","Weekday")
- IF(OR('Calendar'[DayNumberOfWeek] = 1, 'Calendar'[DayNumberOfWeek] = 7) "Weekend","Weekday")
- IF('Calendar'[DayNumberOfWeek] IN {1,7}, "Weekend","Weekday")

BIG DATA EXPERIENCE

Power BI arithmetic operator

Arithmetic operator	Meaning	Example
+ (plus sign)	Addition	3+3
– (minus sign)	Subtraction or sign	3–1–1
* (asterisk)	Multiplication	3*3
/ (forward slash)	Division	3/3
^ (caret)	Exponentiation	16^4

BIG DATA EXPERIENCE

Power BI comparison operators

Comparison operator	Meaning	Example
=	Equal to	[Region] = "USA"
==	Strict equal to	[Region] == "USA"
>	Greater than	[Sales Date] > "Jan 2009"
<	Less than	[Sales Date] < "Jan 1 2009"
>=	Greater than or equal to	[Amount] >= 20000
<=	Less than or equal to	[Amount] <= 100
<>	Not equal to	[Region] <> "USA"

BIG DATA EXPERIENCE

Power BI logical operators

Text operator	Meaning	Examples
&& (double ampersand)	Creates an AND condition between two expressions that each have a Boolean result. If both expressions return TRUE, the combination of the expressions also returns TRUE; otherwise the combination returns FALSE.	([Region] = "France") && ([BikeBuyer] = "yes"))
|| (double pipe symbol)	Creates an OR condition between two logical expressions. If either expression returns TRUE, the result is TRUE; only when both expressions are FALSE is the result FALSE.	(([Region] = "France") || ([BikeBuyer] = "yes"))
IN	Creates a logical OR condition between each row being compared to a table. Note: the table constructor syntax uses curly braces.	'Product'[Color] IN { "Red", "Blue", "Black" }

KM UTT g·able

Operator Precedence

Higher Precedence
Execute First

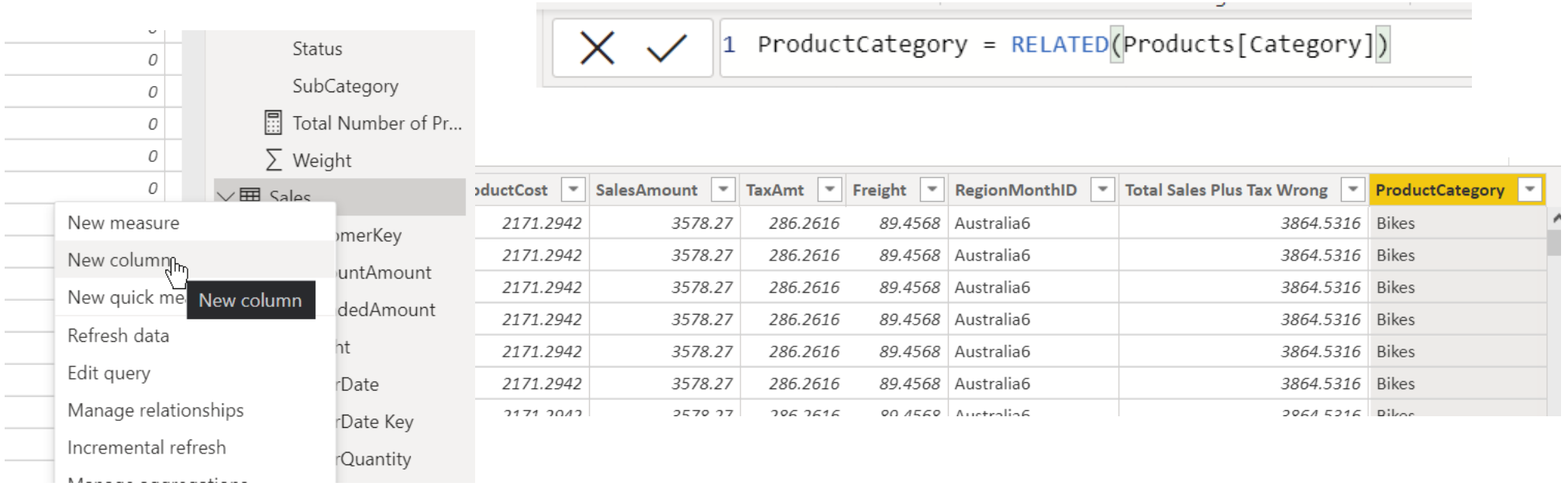
Operator	Description
^	Exponentiation
–	Sign (as in –1)
* and /	Multiplication and division
!	NOT (unary operator)
+ and –	Addition and subtraction
&	Connects two strings of text (concatenation)
=,==,<,>,<=,>=,<>	Comparison

DAX, Calculated Columns Practices

- Creating a Half-Year column
  - In the Calendar table, write a calculated column that returns the value H1 for the first half of each year (January through June) and returns H2 for the second half of each year (July through December). Hint: You might want to use an IF statement to do this.

DAX, RELATED

- The functions RELATED() is typically used in calculated columns to reference relevant records in other tables

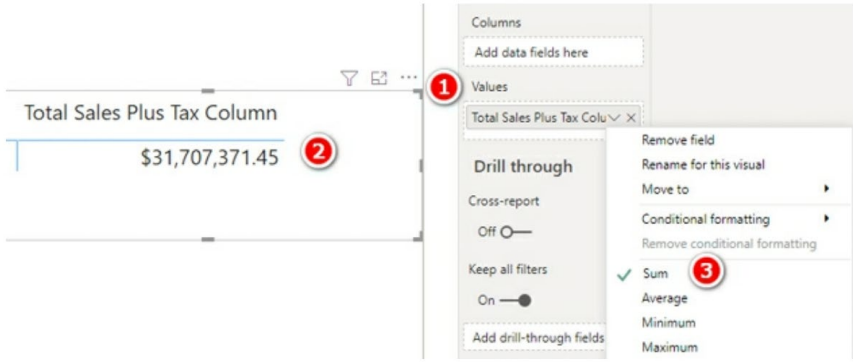


DAX, Iterators

- SUM(), COUNT(), etc. are called “aggregation functions”
- There is another class of functions that perform aggregation after expression. These X-functions are part of the iterator family.
- These iterators are measures.

Iterating function vs calculated columns

Total Sales Plus Tax Column =
Sales[ExtendedAmount] + Sales[TaxAmt]



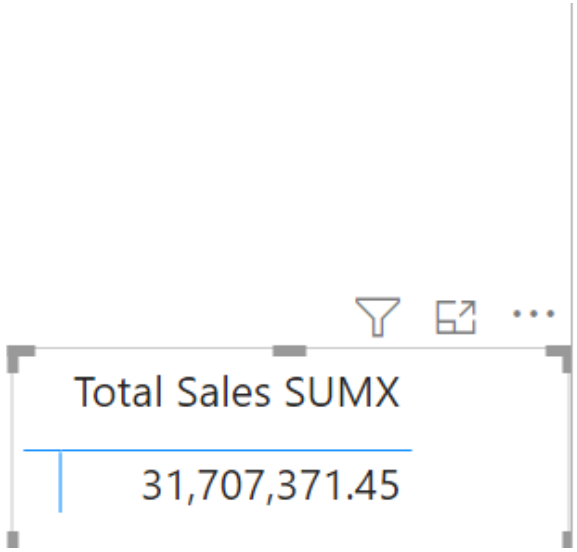
VS

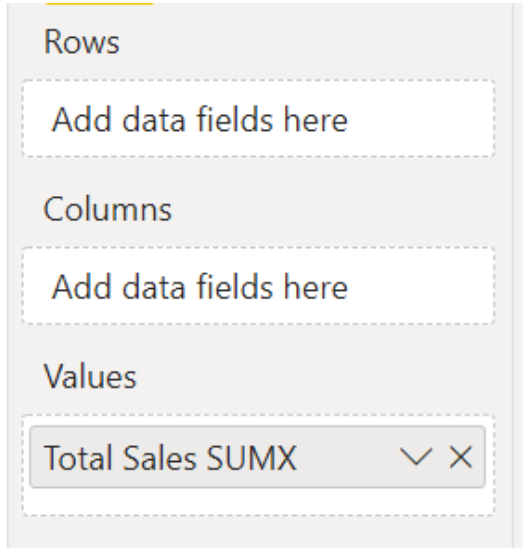
```
Total Sales Including Tax SUMX Version =
SUMX ( Sales, Sales[ExtendedAmount] + Sales[TaxAmt] )
```

The big difference between iterating functions like SUMX() and calculated columns is that a calculated column permanently stores the results in the new column.

DAX, iterator: SUMX()

```
1  Total Sales SUMX = SUMX(Sales, Sales[ExtendedAmount] + Sales[TaxAmt])
```





BIG DATA EXPERIENCE

DAX, iterator: SUMX() Practices

- [Total Sales SUMX version]
  - Multiply quantity by unit price
- [Total Sales Including Freight]
  - Add the ExtendedAmount column to the freight cost

KMUTT g-able

DAX, iterator: AVERAGEX Practices

- [Average Selling Price Including Tax Per Item]
  - Selling price + Tax
- [Average profit per item sold]
  - Profit is selling price – cost price

BIG DATA EXPERIENCE

Other iterators

- MAXX()
- MINX()
- COUNTAX()
- COUNTX()

KMUTT g-able

CALCULATE()

- CALCULATE() is the most important and powerful function in the DAX language.
- It is important because it is the only function that has the ability to modify the natural filtering behavior of visuals.
- CALCULATE() alters the natural filtering behaviour of visuals.
- It modifies an expression (which can be a measure or another DAX formula) by applying, removing, or modifying filters.
- The syntax of CALCULATE() is:

= CALCULATE(expression, filter 1, filter 2, ..., filter n)

Simple filter

- Suppose we want only the total sales of blue product

```
Total Sales of Blue Products = CALCULATE([Total Sales], Products[Color]="Blue")
```

Category	Total Sales
Accessories	$700,760
Bikes	$28,318,145
Clothing	$339,773
Total	**$29,358,677**

Category	Total Sales	Total Sales of Blue Products
Accessories	$700,760	$74,354
Bikes	$28,318,145	$2,169,056
Clothing	$339,773	$35,687
Total	**$29,358,677**	**$2,279,096**

DAX: CALCULATE()
Practices, one table

- [Total Male Customers], for each occupation
- [Total Customers Born Before 1970], use DATE() function for creating date reference, for each occupation
- [Customers earning at least $100,000 per year], for each occupation

BIG DATA EXPERIENCE

DAX: CALCULATE()
Practices, multiple table

- Use Region as Row Header
- [Total Sales] as a reference, not use CALCULATE
- [Total Sales of Clothing], for Region
- [Sales to Female Customers]

KMUTT g-able

BIG DATA
EXPERIENCE

Thank you

Question?

KMUTT g-able