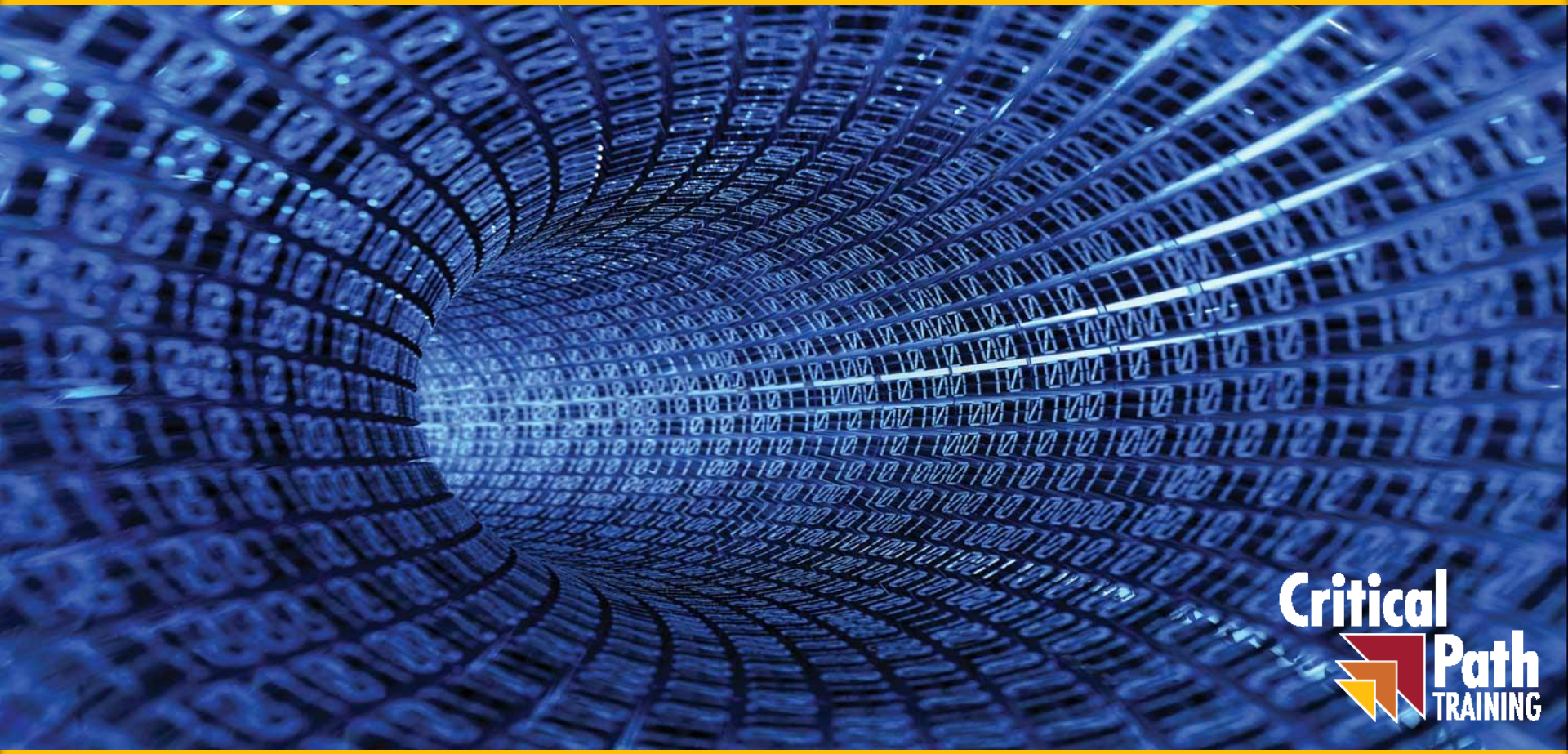
Designing a Data Model with Power BI Desktop



Agenda

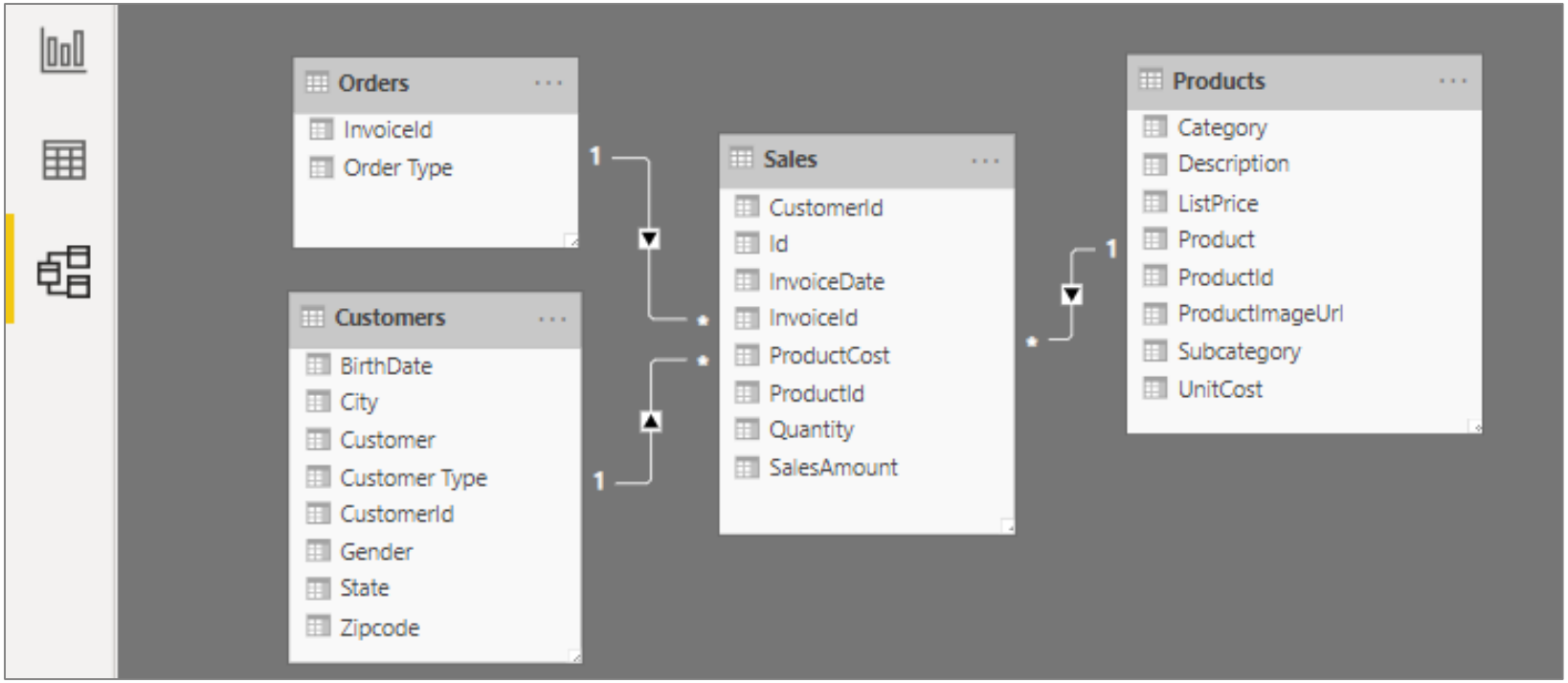
- Creating Table Relationships
- Creating Calculated Columns and Measure
- Creating Tables using DAX Expressions
- Configuring Fields for Geographic Mapping
- Creating Dimensional Hierarchies
- Using the DAX Calculate Function
- Calendar Tables and Time Intelligence

Data Modeling with Power BI Desktop

- Steps to create a data model with Power Pivot
  - Create relationships between tables
  - Modify native columns (e.g. set formatting and data category)
  - Create calculated columns
  - Create measures
  - Create dimensional hierarchies
  - Add Calendar table(s)

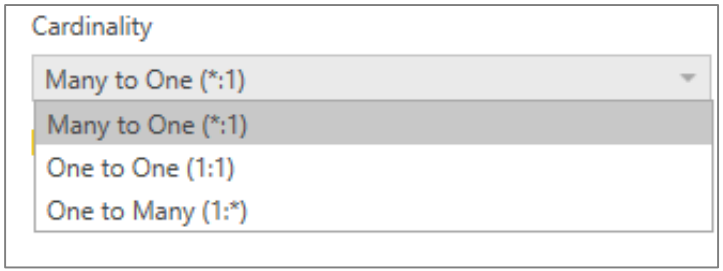
Table Relationships

- Tables in data model associated with relationships
  - Relationships based on single columns
  - Tabular model supports [1-to-1] and [1-to-many] relationships

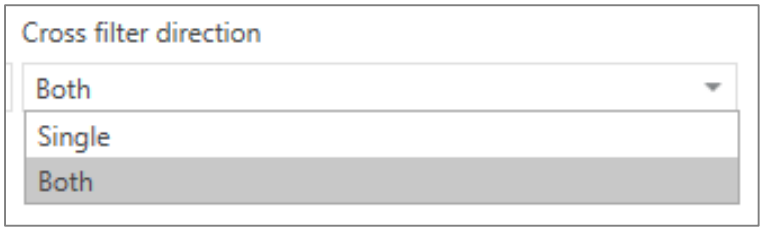


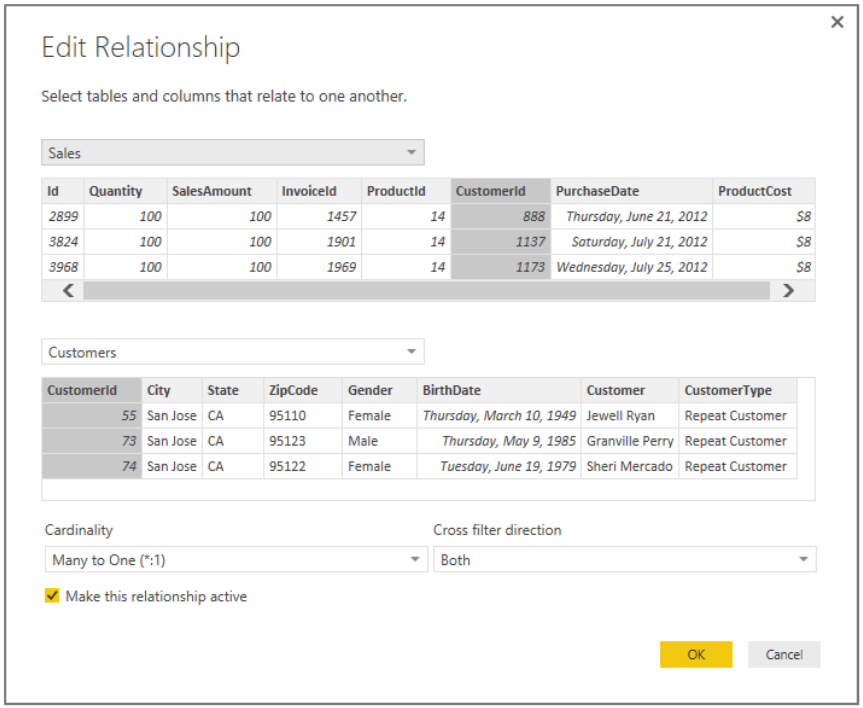
Relationship Properties

- Cardinality



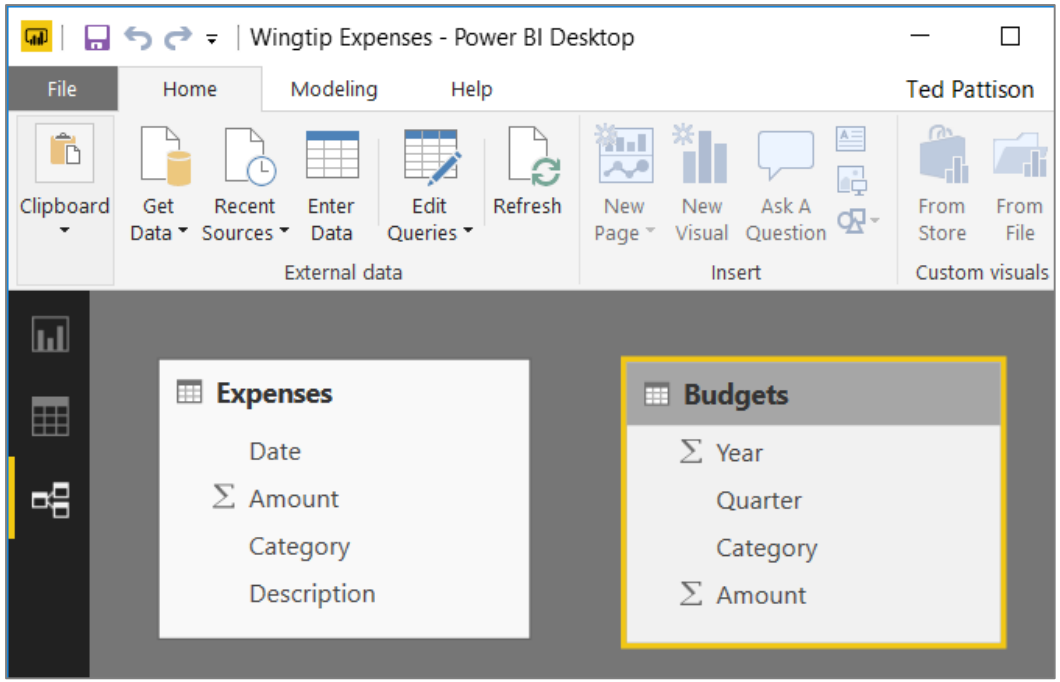
- Cross filter direction





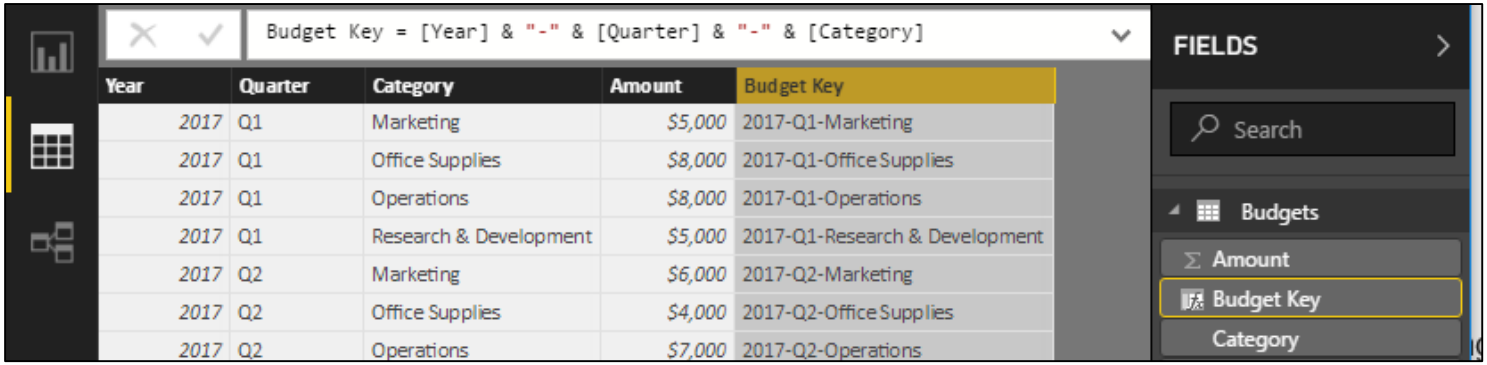
How Do You Create a Relationship Here?

- Two tables don't have fields to create relationship
  - The solution is to create two new calculated columns



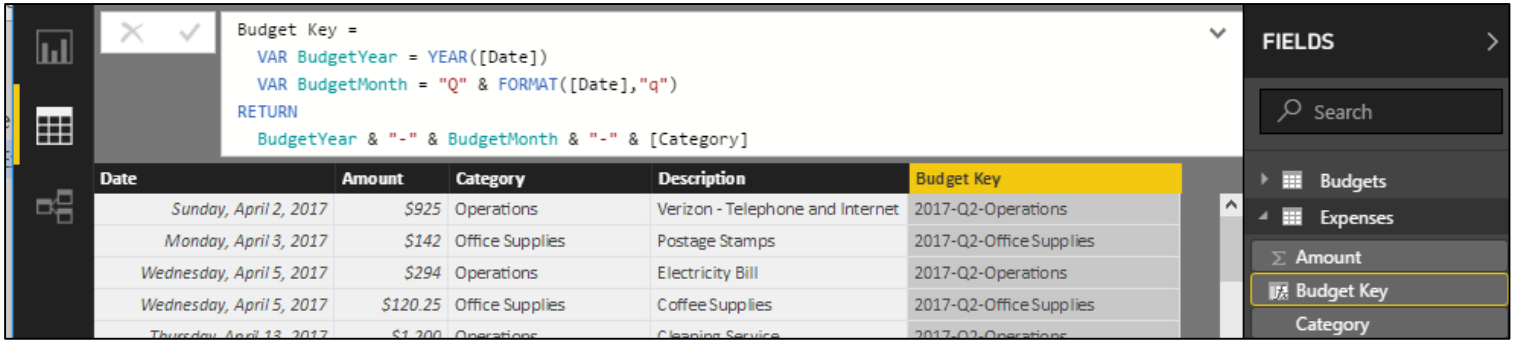
Creating Composite Key Fields

- Create composite key column in Budgets



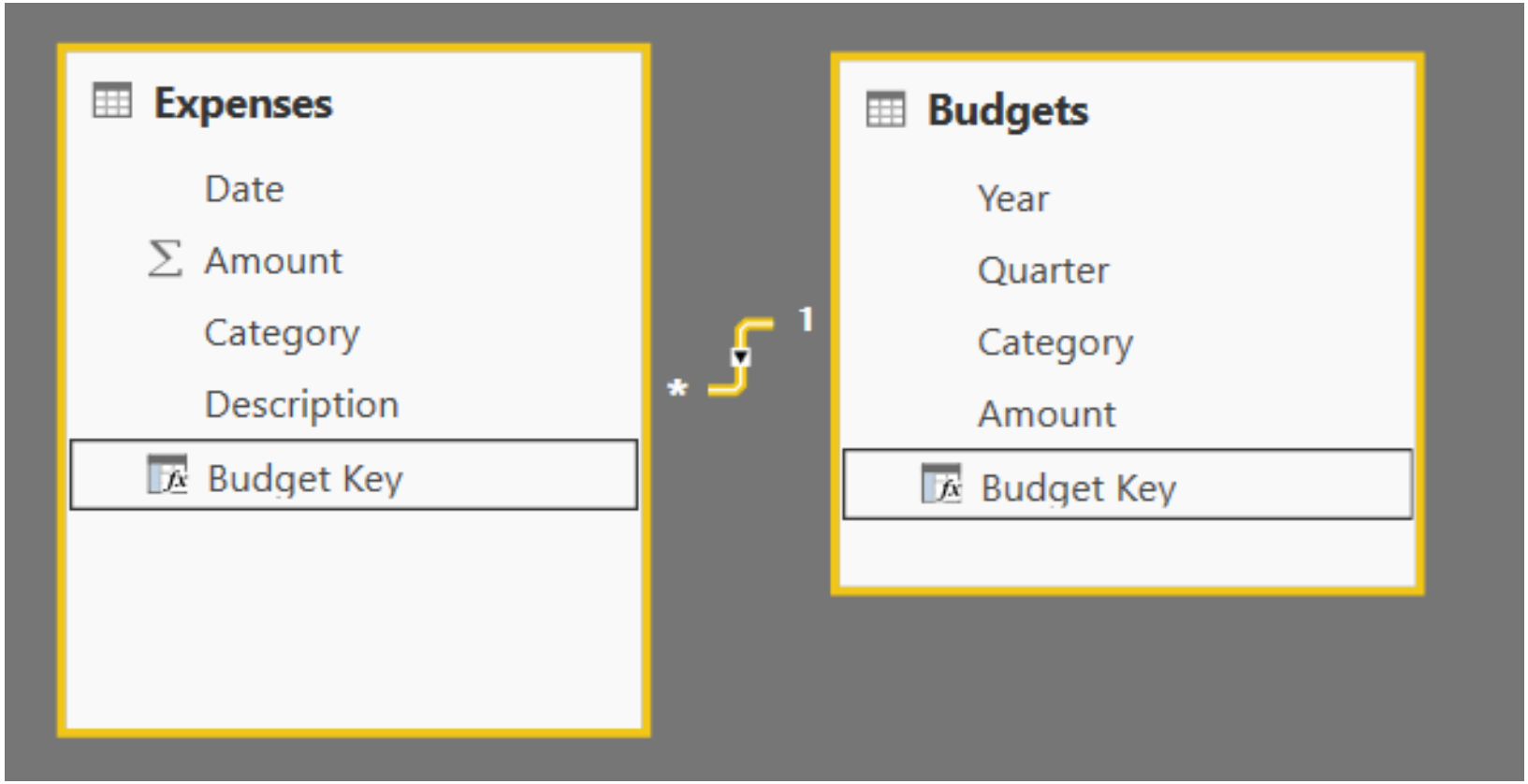
Year	Quarter	Category	Amount	Budget Key
2017	Q1	Marketing	$5,000	2017-Q1-Marketing
2017	Q1	Office Supplies	$8,000	2017-Q1-Office Supplies
2017	Q1	Operations	$8,000	2017-Q1-Operations
2017	Q1	Research & Development	$5,000	2017-Q1-Research & Development
2017	Q2	Marketing	$6,000	2017-Q2-Marketing
2017	Q2	Office Supplies	$4,000	2017-Q2-Office Supplies
2017	Q2	Operations	$7,000	2017-Q2-Operations

- Create composite key column in Expenses



Date	Amount	Category	Description	Budget Key
Sunday, April 2, 2017	$925	Operations	Verizon - Telephone and Internet	2017-Q2-Operations
Monday, April 3, 2017	$142	Office Supplies	Postage Stamps	2017-Q2-Office Supplies
Wednesday, April 5, 2017	$294	Operations	Electricity Bill	2017-Q2-Operations
Wednesday, April 5, 2017	$120.25	Office Supplies	Coffee Supplies	2017-Q2-Office Supplies
Thursday, April 13, 2017	$1,200	Operations	Cleaning Service	2017-Q2-Operations

Create Relationship Using Composite Keys

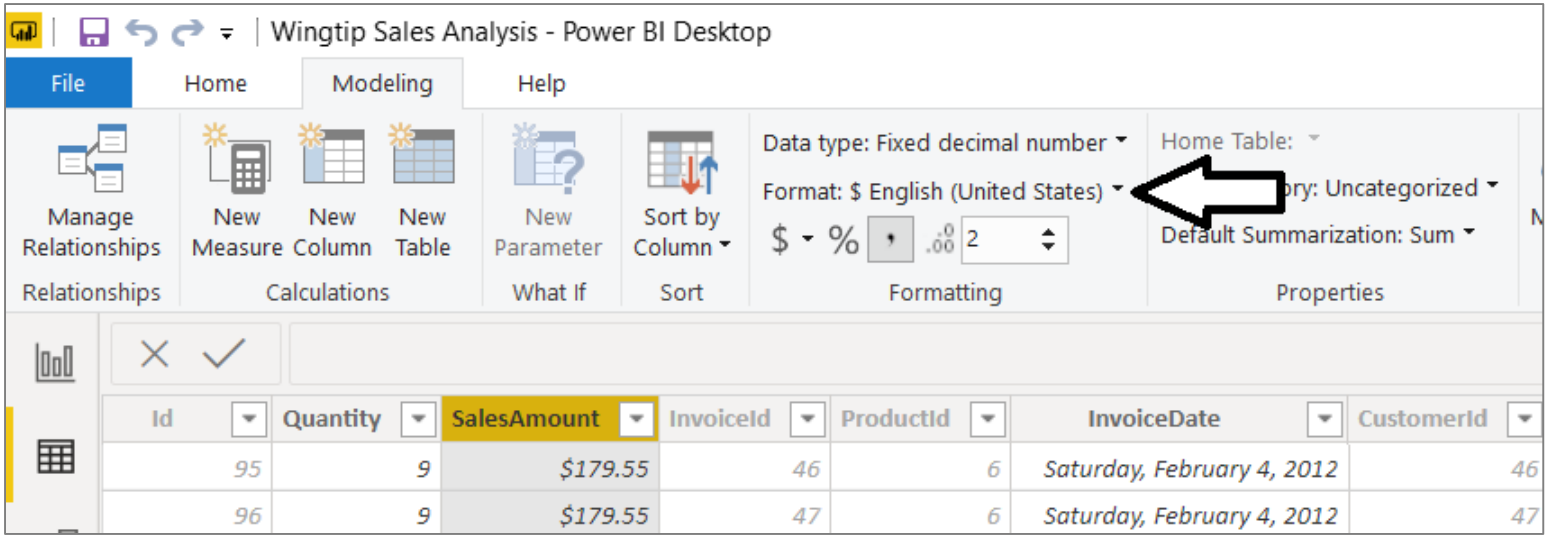


Agenda

- ✓ Creating Table Relationships
- ➢ Creating Calculated Columns and Measure
- Creating Tables using DAX Expressions
- Configuring Fields for Geographic Mapping
- Creating Dimensional Hierarchies
- Using the DAX Calculate Function
- Calendar Tables and Time Intelligence

Formatting Columns

- Each column has its own formatting properties
  - Formatting propagates to reports and visuals
  - Visuals automatically display values using format properties



Working with DAX

- DAX is the language used to create data models
  - DAX stands for "**D**ata **A**nalysis E**x**pression Language"
- DAX expressions are similar to Excel formulas
  - They always start with an equal sign (=)
  - DAX provides many built-in functions similar to Excel
- DAX Expressions are unlike Excel formulas…
  - DAX expressions cannot reference cells (e.g. A1 or C4)
  - Instead DAX expressions reference columns and tables

```
=SUM('Sales'[SalesAmount])
```

Writing DAX Expressions

- Some DAX expressions are simple

```
Sales Revenue = Sum(Sales[SalesAmount])
```

- Some DAX expressions are far more complex

```
Sales Growth PM = IF(
  ( ISFILTERED(Calendar[Month]) && ISFILTERED(Calendar[Date]) = FALSE() ),
  DIVIDE(
    SUM(Sales[SalesAmount]) -
    CALCULATE(
      SUM(Sales[SalesAmount]),
      PREVIOUSMONTH(Calendar[Date])
    ),
    CALCULATE(
      SUM(Sales[SalesAmount]),
      PREVIOUSMONTH(Calendar[Date])
    )
  ),
  BLANK()
)
```

Creating Variables in DAX Expressions

- Variables can be added at start of expression
  - Use **VAR** keyword once for each variable
  - Use **RETURN** keyword to return expression value

```
Budget Key =
  VAR BudgetYear = YEAR([Date])
  VAR BudgetMonth = "Q" & FORMAT([Date],"q")
RETURN
  BudgetYear & "-" & BudgetMonth & "-" & [Category]
```

Calculated Columns vs Measures

- Calculated Columns (aka Columns)
  - Evaluated based on context of a single row
  - Evaluated when data is loaded into memory

```
Column1 = <DAX expression>
```

- Measures
  - Evaluated at query time based on current filter context
  - Commonly used for aggregations (e.g. SUM, AVG, etc.)
  - Used more frequently than calculated columns

```
Measure1 = <DAX expression>
```

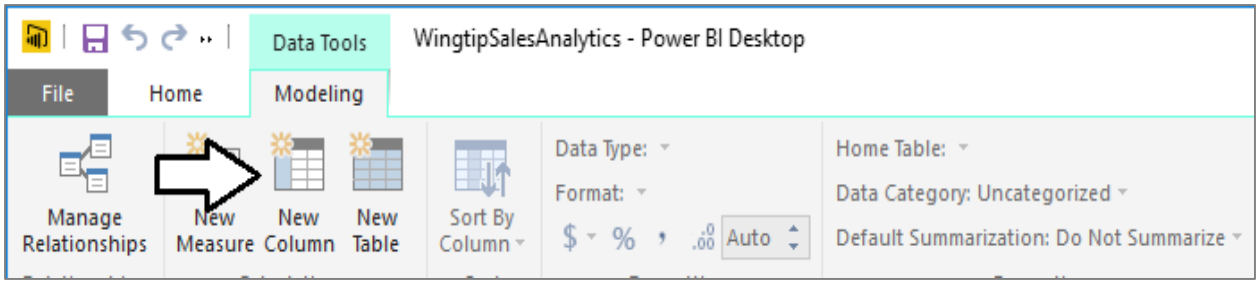
When to Create Calculated Columns

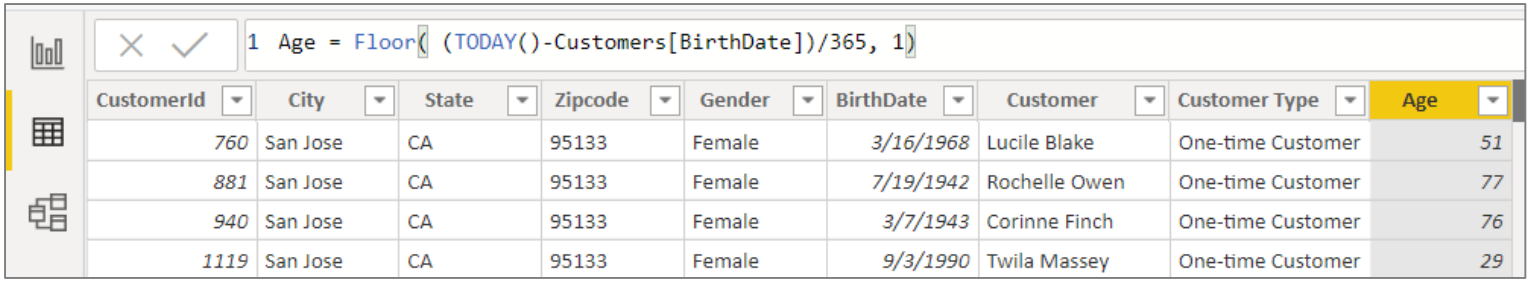
- Measures often better choice than calculate columns
  - Don't create calculated column when you need a measure
  - Prefer to create calculated columns only in specific scenarios

- When should you create calculated columns?
  - To create headers for row labels or column labels
  - To place calculated results in a slicer for filtering
  - Define an expression strictly bound to current row
  - Categories text or numbers (e.g. customer age groups)

Creating Calculated Columns

- Edited in formula bar of Power Pivot data view
  - Start with name and then equals (=) sign
  - Enter a valid DAX expression
  - Clicking on column adds it into expression





CustomerId	City	State	Zipcode	Gender	BirthDate	Customer	Customer Type	Age
760	San Jose	CA	95133	Female	3/16/1968	Lucile Blake	One-time Customer	51
881	San Jose	CA	95133	Female	7/19/1942	Rochelle Owen	One-time Customer	77
940	San Jose	CA	95133	Female	3/7/1943	Corinne Finch	One-time Customer	76
1119	San Jose	CA	95133	Female	9/3/1990	Twila Massey	One-time Customer	29

Calculated Column for Customer Age Group

1. Calculate customer age from birthdate

```
Age = Floor( (TODAY()-Customers[BirthDate])/365, 1)
```

CustomerId	City	State	Zipcode	Gender	BirthDate	Customer	Customer Type	Age
760	San Jose	CA	95133	Female	3/16/1968	Lucile Blake	One-time Customer	51
881	San Jose	CA	95133	Female	7/19/1942	Rochelle Owen	One-time Customer	77
940	San Jose	CA	95133	Female	3/7/1943	Corinne Finch	One-time Customer	76
1119	San Jose	CA	95133	Female	9/3/1990	Twila Massey	One-time Customer	29

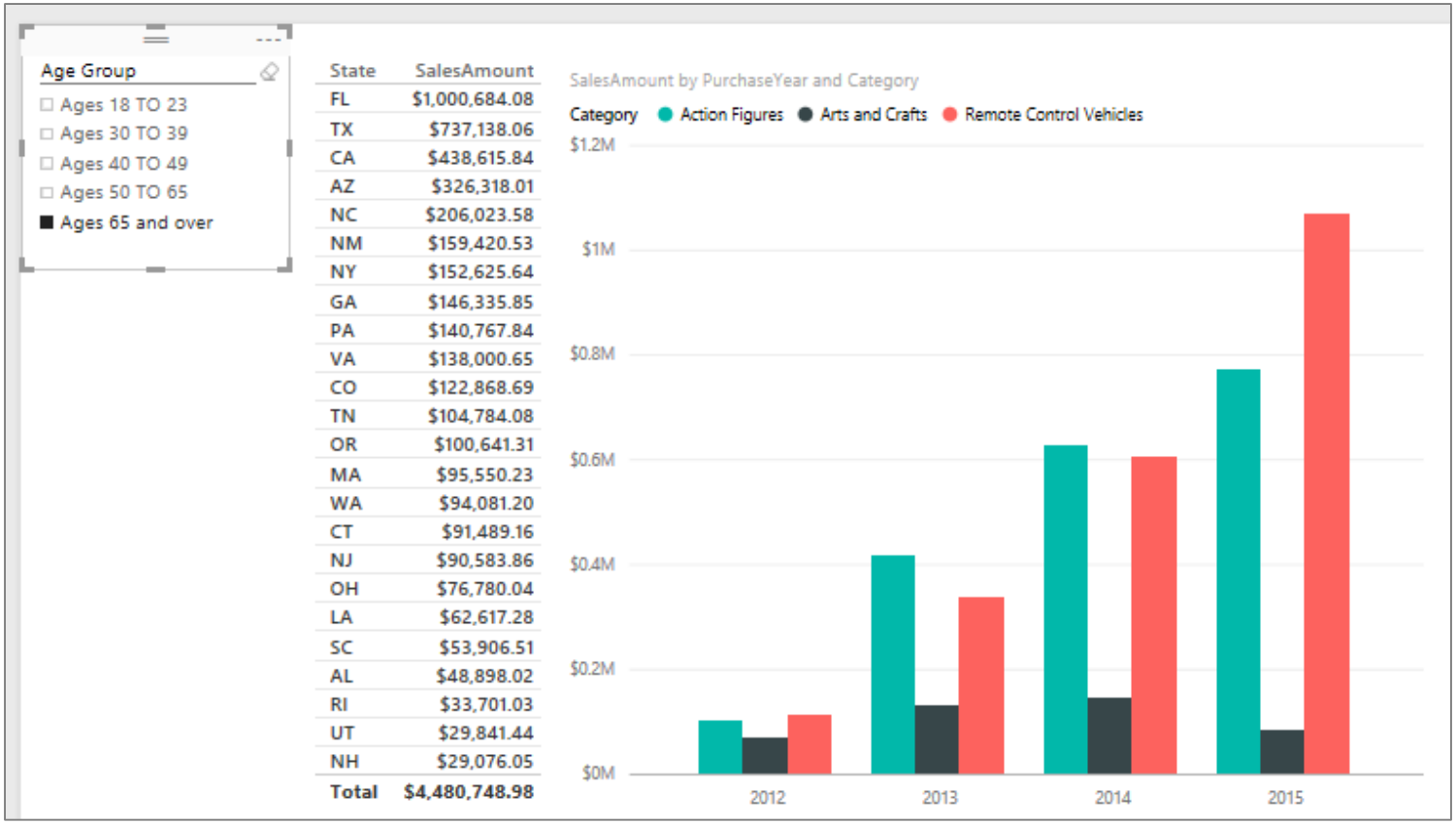
2. Calculate age groups using calculated column

```
Age Group =
SWITCH (
    TRUE(),
    [Age] >= 65, "65 and over",
    [Age] >= 50, "50 to 64",
    [Age] >= 40, "40 to 49",
    [Age] >= 30, "30 to 39",
    [Age] >= 18, "18 to 29",
    [Age] < 18, "Under 18"
)
```

CustomerId	City	State	Zipcode	Gender	BirthDate	Customer	Customer Type	Age	Age Group
760	San Jose	CA	95133	Female	3/16/1968	Lucile Blake	One-time Customer	51	50 to 64
881	San Jose	CA	95133	Female	7/19/1942	Rochelle Owen	One-time Customer	77	65 and over
940	San Jose	CA	95133	Female	3/7/1943	Corinne Finch	One-time Customer	76	65 and over
1119	San Jose	CA	95133	Female	9/3/1990	Twila Massey	One-time Customer	29	18 to 29

Calculated Column used in a Slicer

- Calculated column can populate slicer values

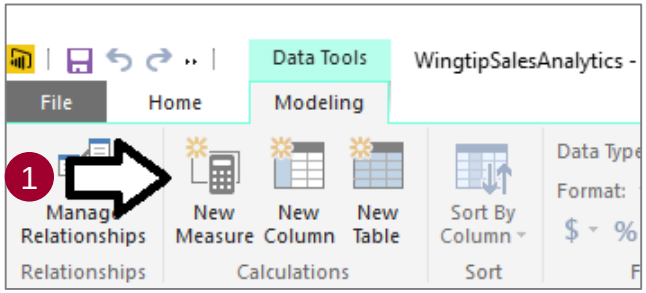


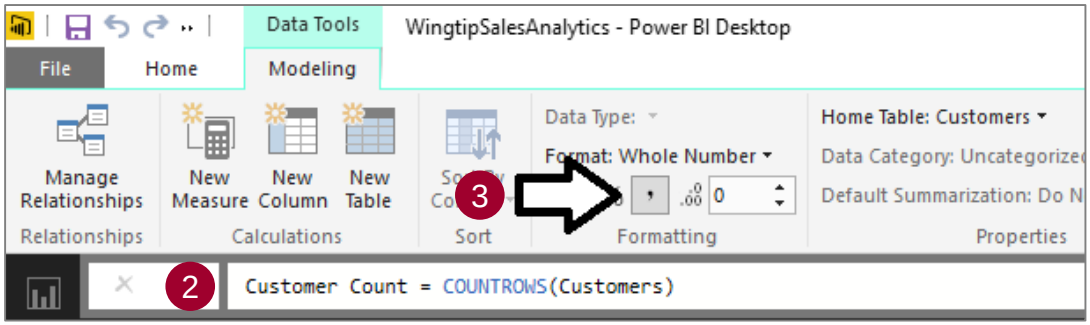
State	SalesAmount
FL	$1,000,684.08
TX	$737,138.06
CA	$438,615.84
AZ	$326,318.01
NC	$206,023.58
NM	$159,420.53
NY	$152,625.64
GA	$146,335.85
PA	$140,767.84
VA	$138,000.65
CO	$122,868.69
TN	$104,784.08
OR	$100,641.31
MA	$95,550.23
WA	$94,081.20
CT	$91,489.16
NJ	$90,583.86
OH	$76,780.04
LA	$62,617.28
SC	$53,906.51
AL	$48,898.02
RI	$33,701.03
UT	$29,841.44
NH	$29,076.05
Total	**$4,480,748.98**

Creating Measures

- Measures have advantage over calculated columns
  - They are evaluated based on the current evaluation context

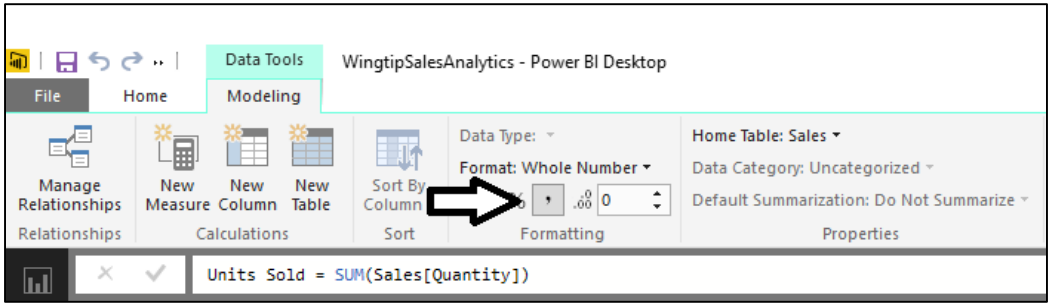
- Creating a measure with Power BI Desktop
  1. Click New Measure button
  2. Give measure a name and write DAX expressions
  3. Configure formatting



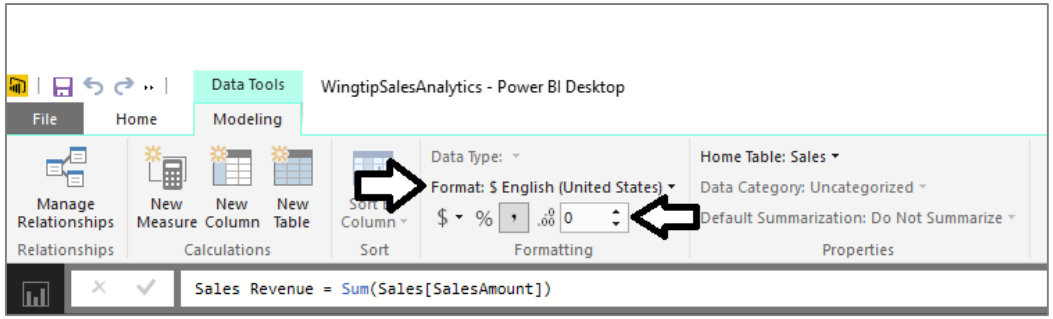


Formatting Measures

- Format as whole number



- Format as currency



Agenda

- ✓ Creating Table Relationships
- ✓ Creating Calculated Columns and Measure
- ➢ Creating Tables using DAX Expressions
- Configuring Fields for Geographic Mapping
- Creating Dimensional Hierarchies
- Using the DAX Calculate Function
- Calendar Tables and Time Intelligence

Creating Tables Dynamically using DAX

Bookmarks - Power BI Desktop

File | Home | Modeling | Help | Ted Pattison

Manage Relationships | New Measure | New Column | New Table | New Parameter | Sort by Column | Data type: | Format: | Auto | Home Table: | Data Category: Uncategorized | Default Summarization: Don't sum

Relationships | Calculations | What If | Sort | Formatting | Properties

```
Sales Regions =
DATATABLE (
    "State", STRING,
    "State Name", STRING,
    "Sales Region", STRING, {
 { "AK", "Alaska", "Western Region" },
 { "AL", "Alabama", "Central Region" },
 { "AR", "Arkansas", "Central Region" },
 { "AZ", "Arizona", "Western Region" },
 { "CA", "California", "Western Region" },
 { "CO", "Colorado", "Western Region" },
```

State	State Name	Sales Region
AK	Alaska	Western Region
AL	Alabama	Central Region
AR	Arkansas	Central Region
AZ	Arizona	Western Region
CA	California	Western Region

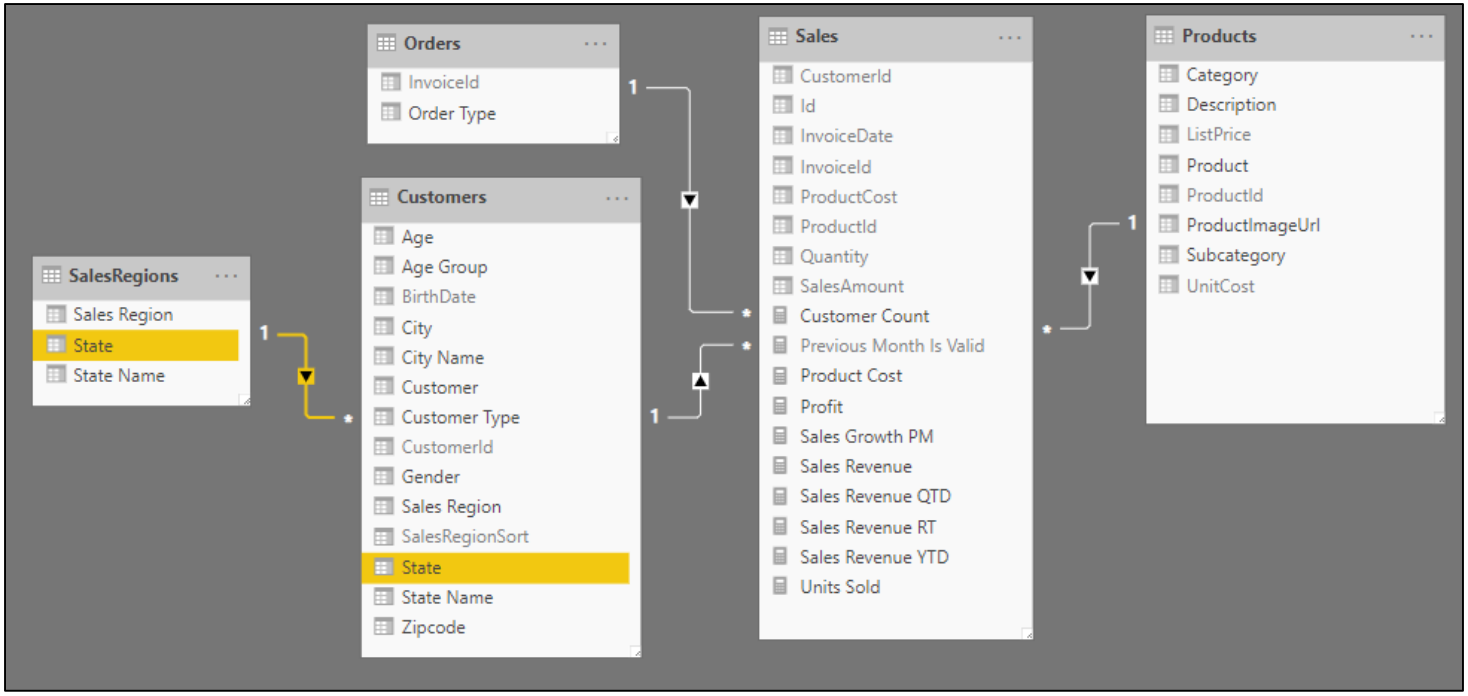
FIELDS

Search

- Sales
- Calendar
- Customers
- Products
- Purchases
- Sales Regions
  - Sales Region
  - State
  - State Name
- SalesRegionsSort

Integrating the Lookup Table into the Data Model

- Lookup table must be integrated into data model
  - Accomplished by creating relationship to one or more tables



The RELATED Function

- RELATED function performs cross-table lookup
  - Effectively replaces older VLOOKUP function
  - Used in many-side table to look up value from one-side
  - Used to pull data from lookup table into primary table

```
1 Sales Region = RELATED(SalesRegions[Sales Region])
```

CustomerId	City	State	Zipcode	Gender	BirthDate	Customer	Customer Type	Age	Age Group	Sales Region
760	San Jose	CA	95133	Female	3/16/1968	Lucile Blake	One-time Customer	51	50 to 64	Western Region
881	San Jose	CA	95133	Female	7/19/1942	Rochelle Owen	One-time Customer	77	65 and over	Western Region

```
1 State Name = RELATED(SalesRegions[State Name])
```

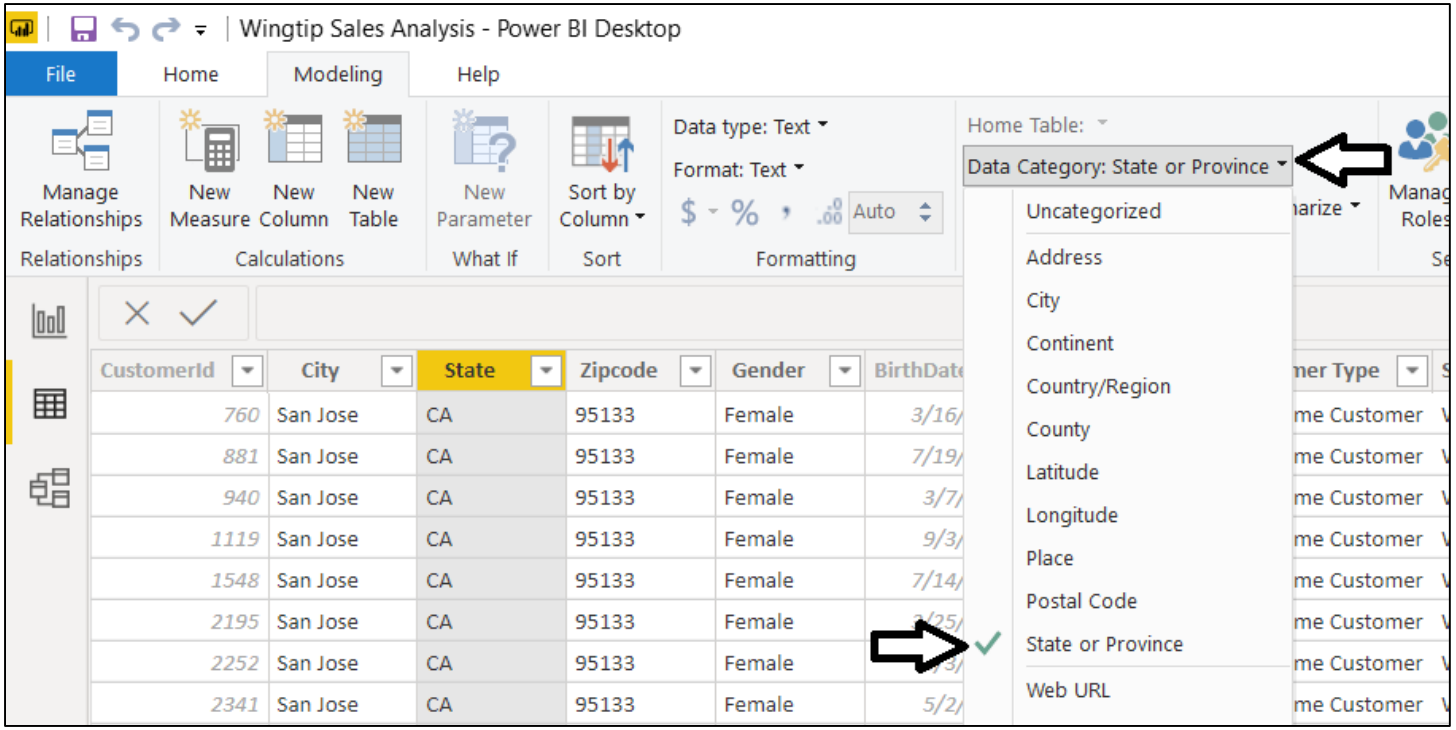
Zipcode	Gender	BirthDate	Customer	Customer Type	Age	Age Group	Sales Region	State Name
95133	Female	3/16/1968	Lucile Blake	One-time Customer	51	50 to 64	Western Region	California
95133	Female	7/19/1942	Rochelle Owen	One-time Customer	77	65 and over	Western Region	California
95133	Female	3/7/1943	Corinne Finch	One-time Customer	76	65 and over	Western Region	California

Agenda

- ✓ Creating Table Relationships
- ✓ Creating Calculated Columns and Measure
- ✓ Creating Tables using DAX Expressions
- ➢ Configuring Fields for Geographic Mapping
- Creating Dimensional Hierarchies
- Using the DAX Calculate Function
- Calendar Tables and Time Intelligence

Geographic Field Metadata

- Fields in data model have metadata properties
  - Metadata used by visuals and reporting tools
  - Used as hints to Bing Mapping service



Eliminate Geographic Ambiguity

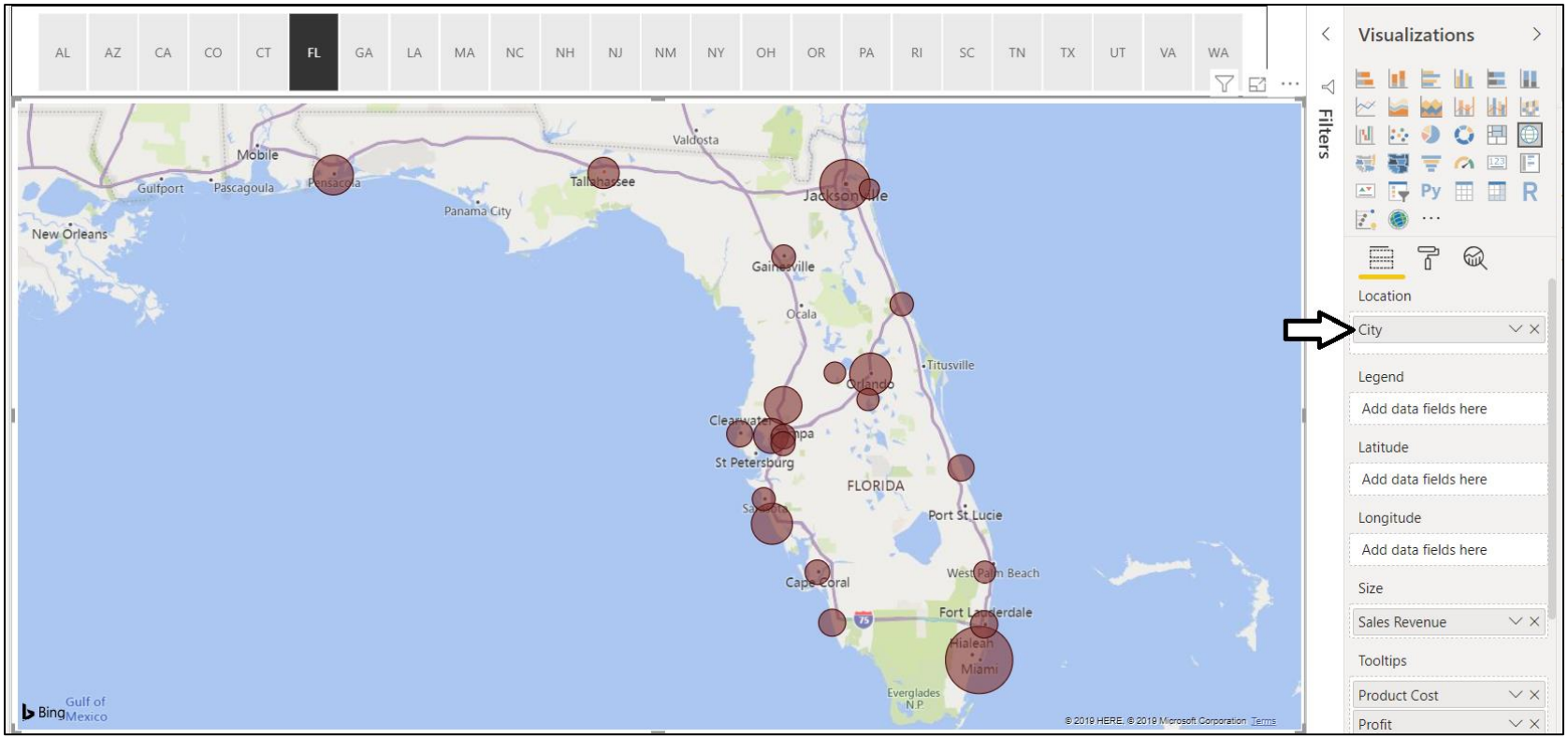
- City name alone is ambiguous
  - "Athens" defaults to Greece not Georgia
  - Concatenate city name with state to disambiguate

```
1 City = [City Name] & ", " & [State]
```

BirthDate	Customer	Customer Type	Age	Age Group	Sales Region	State Name	SalesRegionSort	City
3/16/1968	Lucile Blake	One-time Customer	51	50 to 64	Western Region	California	1	San Jose, CA
7/19/1942	Rochelle Owen	One-time Customer	77	65 and over	Western Region	California	1	San Jose, CA
3/7/1943	Corinne Finch	One-time Customer	76	65 and over	Western Region	California	1	San Jose, CA
9/3/1990	Twila Massey	One-time Customer	29	18 to 29	Western Region	California	1	San Jose, CA
7/14/1955	Kellie Yang	One-time Customer	64	50 to 64	Western Region	California	1	San Jose, CA
3/25/1951	Megan Martin	One-time Customer	68	65 and over	Western Region	California	1	San Jose, CA

Using Map Visual with a Geographic Field

- Map Visual shows distribution over geographic area
  - Visual automatically updates when filtered

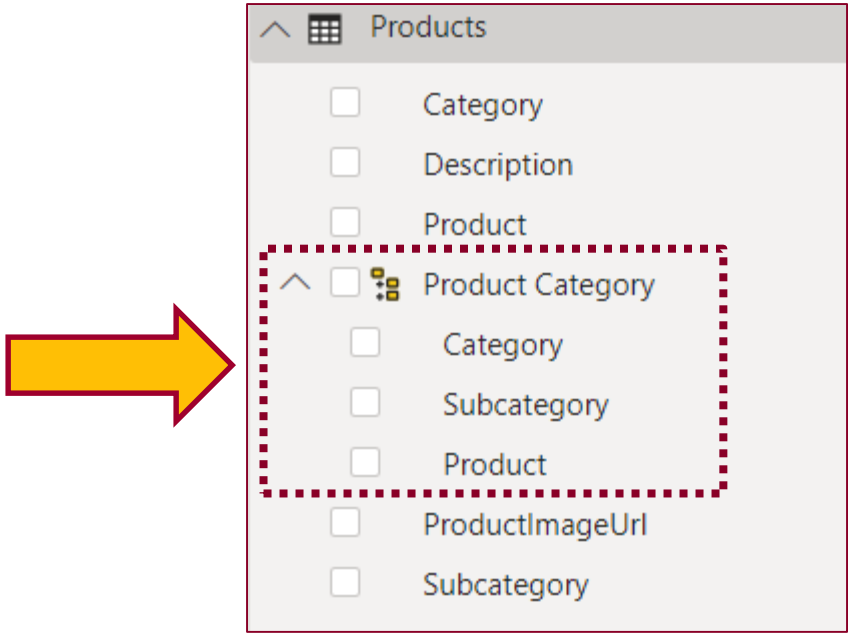


Agenda

- ✓ Creating Table Relationships
- ✓ Creating Calculated Columns and Measure
- ✓ Creating Tables using DAX Expressions
- ✓ Configuring Fields for Geographic Mapping
- ➢ Creating Dimensional Hierarchies
- Using the DAX Calculate Function
- Calendar Tables and Time Intelligence

Dimensional Hierarchies

- Hierarchy created from two or more columns
  - All columns in hierarchy must be from the same table
  - Defines parent-child relationship between columns
  - Provides path to navigate through data
  - Provides path to drill down into greater level of detail



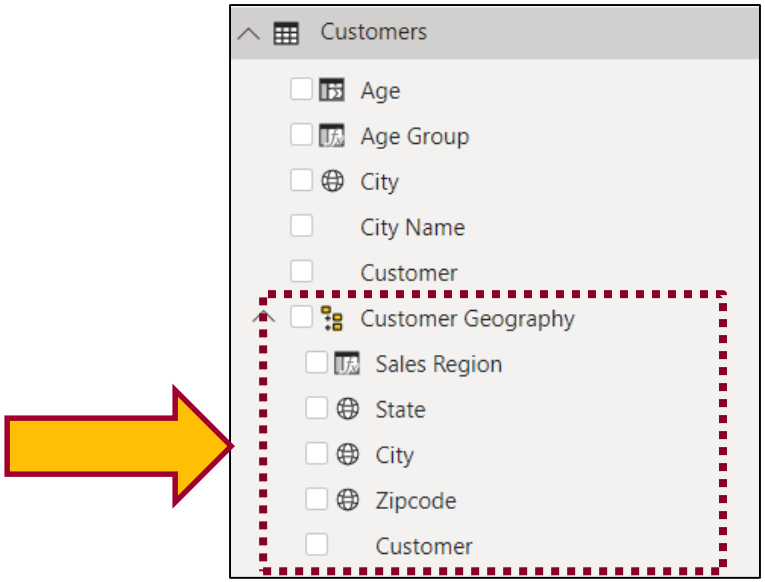
Pulling Columns for Hierarchy into Single Table

- Sometimes hierarchy columns are spread across tables
  - Use RELATED function from DAX to pull columns into single table

```
1 Sales Region = RELATED(SalesRegions[Sales Region])
```

CustomerId	City	State	Zipcode	Gender	BirthDate	Customer	Customer Type	Age	Age Group	Sales Region
760	San Jose	CA	95133	Female	3/16/1968	Lucile Blake	One-time Customer	51	50 to 64	Western Region
881	San Jose	CA	95133	Female	7/19/1942	Rochelle Owen	One-time Customer	77	65 and over	Western Region

  - Then create hierarchy in the table with all the columns



Agenda

- ✓ Creating Table Relationships
- ✓ Creating Calculated Columns and Measure
- ✓ Creating Tables using DAX Expressions
- ✓ Configuring Fields for Geographic Mapping
- ✓ Creating Dimensional Hierarchies
- ➢ Using the DAX Calculate Function
- • Calendar Tables and Time Intelligence

A Tale of Two Evaluation Contexts

- Row Context
  - Context includes all columns in iteration of current row
  - Used to evaluate DAX expression in calculated column
  - Only available in measures with iterator function (e.g. SUMX)

- Filter Context
  - Context includes filter(s) defining current set of rows
  - Used by default to evaluate DAX expressions in measures
  - Can be fully ignored or partially ignored using DAX code
  - Not used to evaluate DAX in calculated columns

Understanding Row Context

- Row context used to evaluate calculated columns

```
City = [City Name] & ", " & [State]
```

	Age Group	Sales Region	State Name	SalesRegionSort	City
48	Ages 40 TO 49	Western Region	California	1	San Jose, CA
74	Ages 65 and over	Western Region	California	1	San Jose, CA
73	Ages 65 and over	Western Region	California	1	San Jose, CA
25	Ages 18 TO 23	Western Region	California	1	San Jose, CA
61	Ages 50 TO 65	Western Region	California	1	San Jose, CA
65	Ages 65 and over	Western Region	California	1	San Jose, CA

```
Age = Floor( (TODAY()-Customers[BirthDate])/365, 1)
```

Customer	Customer Type	Age	Age Group	Sales Region	State Name
Lucile Blake	One-time Customer	48	Ages 40 TO 49	Western Region	California
Rochelle Owen	One-time Customer	74	Ages 65 and over	Western Region	California
Corinne Finch	One-time Customer	73	Ages 65 and over	Western Region	California

Understanding Iterators Like SUMX

- Standard aggregation functions (e.g. SUM) have no row context
  - You can use SUM to sum values of a single column
  - You cannot use SUM to sum results of an expressions

```
Gross Margin = SUM( Sales[SalesAmount]-Sales[ProductCost] )
```

The SUM function only accepts a column reference as an argument.

- Iterator functions (e.g. SUMX) iterate through rows in target table

```
Gross Margin = SUMX(Sales, Sales[SalesAmount]-Sales[ProductCost] )
```

  - First argument accepts expressions that evaluates to table of rows
  - Second argument accepts expression that is evaluated for each row

Understanding Filter Context

- Visuals apply various filters in different evaluation contexts

Month in Year	2012	2013	2014	2015	Total
Jan	$3,063	$307,182	$629,969	$959,863	**$1,900,077**
Feb	$33,218	$291,942	$609,637	$969,330	**$1,904,126**
Mar	$49,213	$346,186	$628,618	$675,533	**$1,699,551**
Apr	$40,434	$380,869	$661,588	$722,456	**$1,805,347**
May	$83,840	$377,376	$748,193	$698,311	**$1,907,720**
Jun	$136,670	$353,586	$814,333	$785,793	**$2,090,382**
Jul	$144,244	$391,202	$788,469	$921,994	**$2,245,908**
Aug	$197,952	$476,884	$869,143	$1,084,189	**$2,628,168**
Sep	$215,097	$504,532	$890,958	$1,088,863	**$2,699,449**
Oct	$239,513	$577,439	$988,789	$1,211,810	**$3,017,551**
Nov	$376,503	$579,507	$999,574	$1,305,029	**$3,260,613**
Dec	$424,240	$769,473	$1,644,980	$1,732,932	**$4,571,625**
Total	**$1,943,986**	**$5,356,177**	**$10,274,251**	**$12,156,103**	**$29,730,517**

Filters on this evaluation

[Year] = 2015

[Month in Year] = "October"

- Filter context also affected by slicers and other filters

Sales Region
- ■ Western Region
- ☐ Central Region
- ☐ Eastern Region

Customer Type
- ☐ One-time Customer
- ■ Repeat Customer

Month in Year	2012	2013	2014	2015	Total
Jan		$117,712	$202,751	$182,616	**$503,079**
Feb	$8,264	$126,522	$181,564	$184,674	**$501,024**
Mar	$22,434	$148,668	$160,857	$169,933	**$501,892**
Apr	$22,235	$178,506	$183,987	$194,197	**$578,925**
May	$36,719	$169,582	$210,150	$173,661	**$590,112**
Jun	$55,119	$158,668	$217,947	$196,431	**$628,166**
Jul	$72,823	$187,093	$233,333	$193,830	**$687,079**
Aug	$90,917	$169,789	$233,101	$209,895	**$703,703**
Sep	$77,898	$155,469	$225,287	$213,017	**$671,672**
Oct	$84,735	$208,700	$197,377	$207,227	**$698,039**
Nov	$130,678	$168,821	$227,856	$190,144	**$717,498**
Dec	$147,043	$203,781	$234,393	$195,796	**$781,013**
Total	**$748,866**	**$1,993,312**	**$2,508,601**	**$2,311,421**	**$7,562,200**

Filters on this evaluation

[Year] = 2015

[Month in Year] = "October"

[Sales Region] = "Western Region"

[Customer Type] = "Repeat Customer"

Using the CALCULATE Function

- CALCULATE function provides greatest amount of control
  - First argument defines expression to evaluate
  - Second argument defines table on which to evaluate expression
  - You can evaluate expressions with or without current filter context

```
Pct of All Products =
DIVIDE(
    SUM( Sales[SalesAmount] ),
    CALCULATE(
        Sum (Sales[SalesAmount] ),
        ALL(Products[Category], Products[Subcategory], Products[Product])
    )
)
```

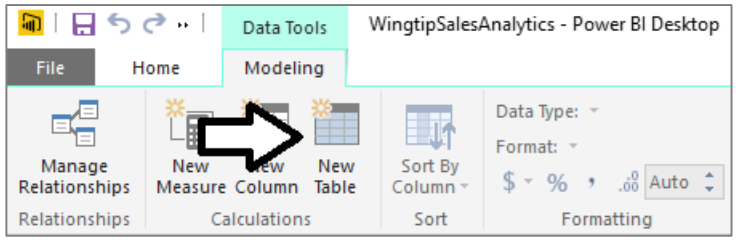
```
Pct of Product Category =
DIVIDE(
    SUM( Sales[SalesAmount] ),
    CALCULATE(
        Sum (Sales[SalesAmount] ),
        ALL( Products[Subcategory], Products[Product] )
    )
)
```

Agenda

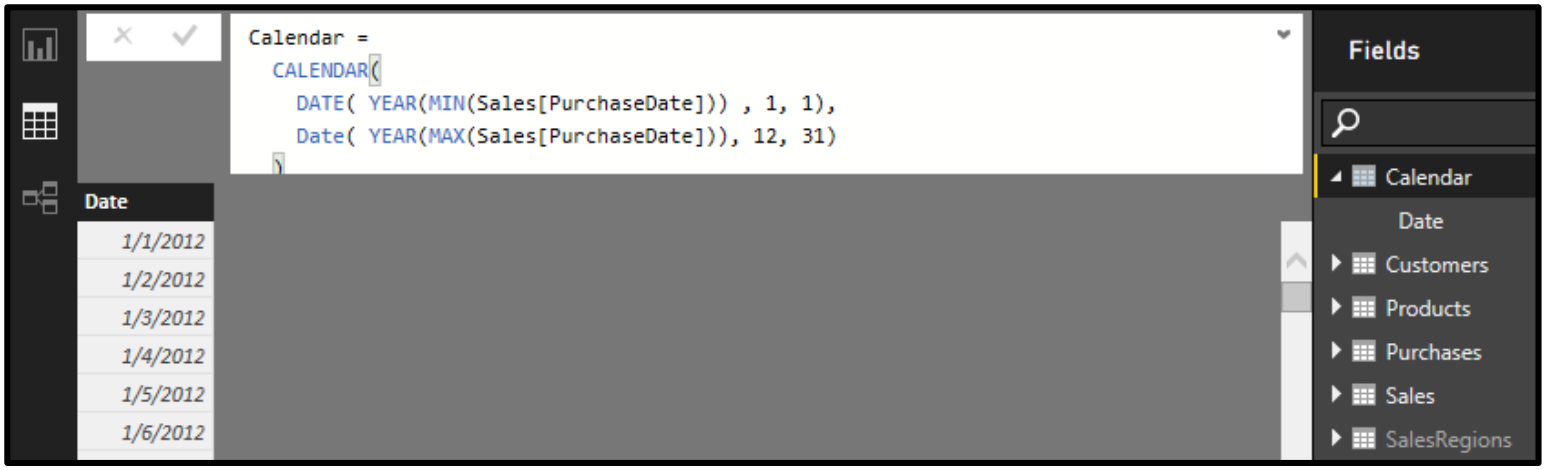
- ✓ Creating Table Relationships
- ✓ Creating Calculated Columns and Measure
- ✓ Creating Tables using DAX Expressions
- ✓ Configuring Fields for Geographic Mapping
- ✓ Creating Dimensional Hierarchies
- ✓ Using the DAX Calculate Function
- ➢ Calendar Tables and Time Intelligence

Creating Calendar Table as Calculated Table

- Use **New Table** command in ribbon



- Create calendar table using DAX **CALENDAR** function



Adding Columns to Calendar Table

- Creating the **Year** column

```
Year = YEAR('Calendar'[Date])
```

Date	Year
1/1/2012	2012
1/2/2012	2012
1/3/2012	2012

- Creating the **Quarter** column

```
Quarter = YEAR('Calendar'[Date]) & "-Q" & FORMAT('Calendar'[Date], "q")
```

Date	Year	Quarter
01/01/2012	2012	2012-Q1
01/02/2012	2012	2012-Q1
01/03/2012	2012	2012-Q1
01/04/2012	2012	2012-Q1
01/05/2012	2012	2012-Q1

- Creating the **Month** column

```
Month = FORMAT('Calendar'[Date], "MMM yyyy")
```

Date	Year	Quarter	Month
1/1/2012	2012	2012-Q1	Jan 2012
1/2/2012	2012	2012-Q1	Jan 2012
1/3/2012	2012	2012-Q1	Jan 2012

Configuring Sort Columns

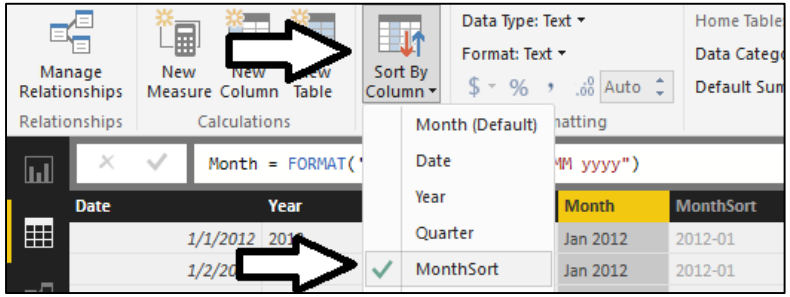
- Month column will not sort in desired fashion by default
  - For example, April will sort before January, February and March

- Creating a sort column for the **Month** column
  - **MonthSort** sorts alphabetically & chronologically at same time

```
MonthSort = FORMAT('Calendar'[Date], "yyyy-MM")
```

Date	Year	Quarter	Month	MonthSort
1/1/2012	2012	2012-Q1	Jan 2012	2012-01
1/2/2012	2012	2012-Q1	Jan 2012	2012-01

  - Configure **Month** column with **MonthSort** as sort column



Columns for Month in Year and Day in week

- Creating the **Month in Year** column

```
Month in Year = FORMAT('Calendar'[Date], "MMMM")
```

Date	Year	Quarter	Month	MonthSort	Month in Year
1/1/2012	2012	2012-Q1	Jan 2012	2012-01	January
1/2/2012	2012	2012-Q1	Jan 2012	2012-01	January

- Creating the **MonthInYearSort** column

```
MonthInYearSort = MONTH('Calendar'[Date])
```

Date	Year	Quarter	Month	MonthSort	Month in Year	MonthInYearSort
1/1/2012	2012	2012-Q1	Jan 2012	2012-01	January	1
1/2/2012	2012	2012-Q1	Jan 2012	2012-01	January	1

- Creating the **Day of Week** column

```
Day of Week = FORMAT('Calendar'[Date], "dddd")
```

Date	Year	Quarter	Month	MonthSort	Month in Year	MonthInYearSort	Day of Week
1/1/2012	2012	2012-Q1	Jan 2012	2012-01	January	1	Sunday
1/2/2012	2012	2012-Q1	Jan 2012	2012-01	January	1	Monday

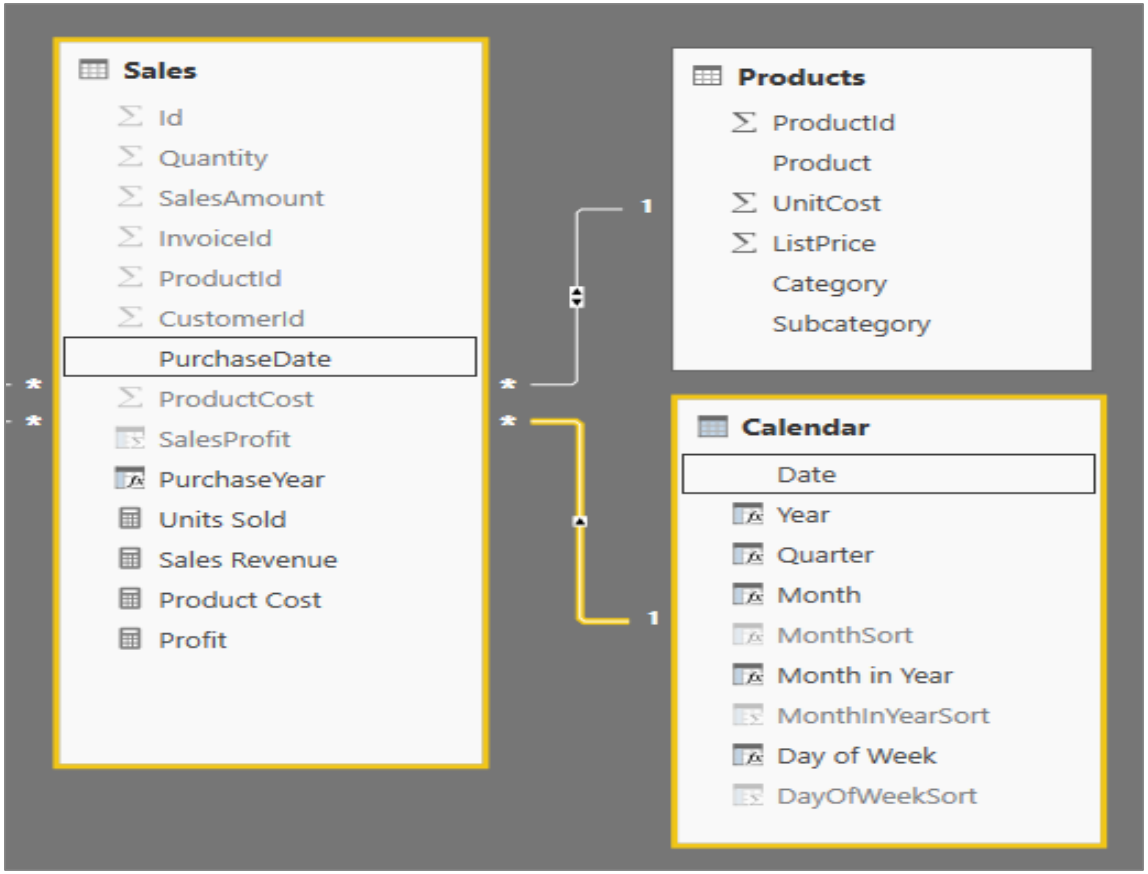
- Creating the **DayOfWeekSort** column

```
DayOfWeekSort = WEEKDAY('Calendar'[Date], 2)
```

Date	Year	Quarter	Month	MonthSort	Month in Year	MonthInYearSort	Day of Week	DayOfWeekSort
1/1/2012	2012	2012-Q1	Jan 2012	2012-01	January	1	Sunday	7
1/2/2012	2012	2012-Q1	Jan 2012	2012-01	January	1	Monday	1
1/3/2012	2012	2012-Q1	Jan 2012	2012-01	January	1	Tuesday	2
1/4/2012	2012	2012-Q1	Jan 2012	2012-01	January	1	Wednesday	3

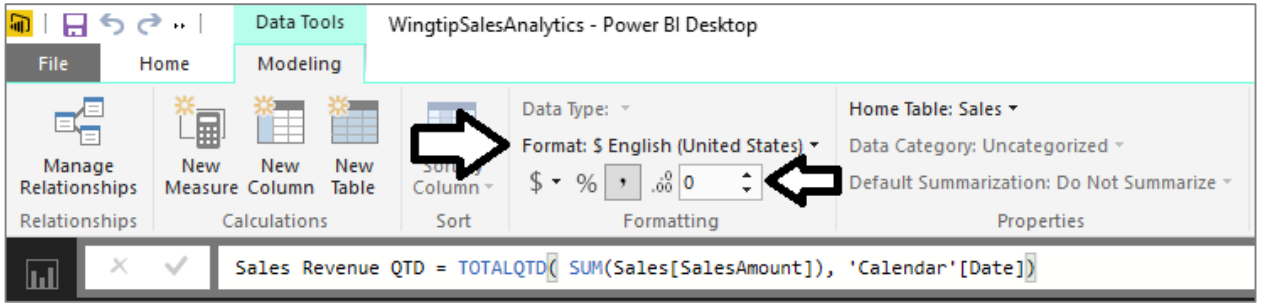
Integrating Calendar Table into Data Model

- Calendar table needs relationship to one or more tables

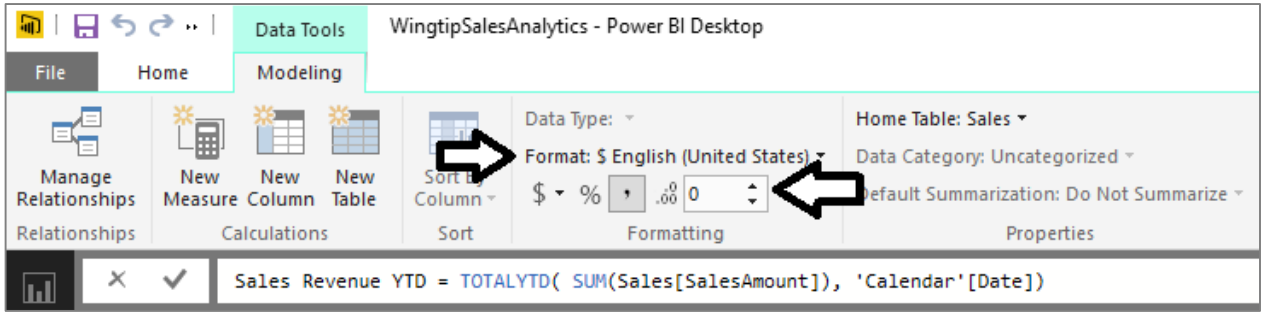


Calculated Fields for QTD and YTD Sales

- TOTALQTD function calculates quarter-to-date totals

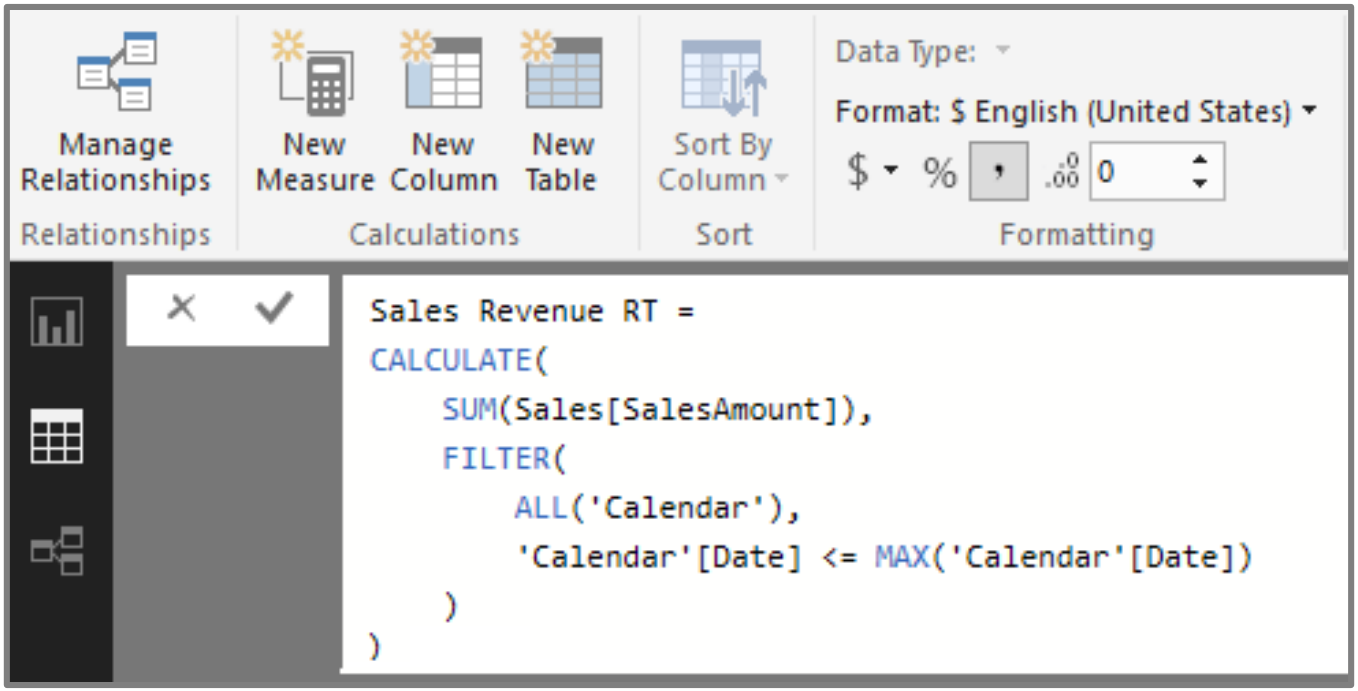


- TOTALYTD function calculates year-to-date totals



Creating Running Total using CALCULATE

- Calculate a running total of sales revenue across years
  - This must be done using **CALCULATE** function



Sales Growth PM Measure - First Attempt

- Create a measure named Sales Growth PM

```
Sales Growth PM =
DIVIDE(
  SUM(Sales[SalesAmount]) -
  CALCULATE(
    SUM(Sales[SalesAmount]),
    PREVIOUSMONTH(Calendar[Date])
  ),
  CALCULATE(
    SUM(Sales[SalesAmount]),
    PREVIOUSMONTH(Calendar[Date])
  )
)
```

- Use measure in matrix evaluating month and quarter
  - Measure returns correct value when filtered by Month
  - Measure returns large, erroneous value when filtered by Quarter

Year	Quarter	Month	Sales Revenue	Sales Growth PM
2014	2014-Q1	Jan 2014	$629,969	-18.13 %
		Feb 2014	$609,637	-3.23 %
		Mar 2014	$628,618	3.11 %
		Total	**$1,868,225**	**142.79 %**
	2014-Q2	Apr 2014	$661,588	5.24 %
		May 2014	$748,193	13.09 %
		Jun 2014	$814,333	8.84 %
		Total	**$2,224,114**	**253.81 %**
	2014-Q3	Jul 2014	$788,469	-3.18 %

Using the ISFILTERED Function

- ISFILTERED function used to determine when perform evaluation

```
Sales Growth PM =
IF(
  ( ISFILTERED(Calendar[Month]) && NOT(ISFILTERED(Calendar[Date])) ),
  DIVIDE(
    SUM(Sales[SalesAmount]) -
    CALCULATE(
      SUM(Sales[SalesAmount]),
      PREVIOUSMONTH(Calendar[Date])
    ),
    CALCULATE(
      SUM(Sales[SalesAmount]),
      PREVIOUSMONTH(Calendar[Date])
    )
  ),
  BLANK()
)
```

- Expression returns Blank value when evaluation context is invalid

Year	Quarter	Month	Sales Revenue	Sales Growth PM
2014	2014-Q1	Jan 2014	$629,969	-18.13 %
		Feb 2014	$609,637	-3.23 %
		Mar 2014	$628,618	3.11 %
		Total	**$1,868,225**	
	2014-Q2	Apr 2014	$661,588	5.24 %
		May 2014	$748,193	13.09 %
		Jun 2014	$814,333	8.84 %
		Total	**$2,224,114**	
	2014-Q3	Jul 2014	$788,469	-3.18 %
		Aug 2014	$869,143	10.23 %

Summary

- ✓ Creating Table Relationships
- ✓ Creating Calculated Columns and Measure
- ✓ Creating Tables using DAX Expressions
- ✓ Configuring Fields for Geographic Mapping
- ✓ Creating Dimensional Hierarchies
- ✓ Using the DAX Calculate Function
- ✓ Calendar Tables and Time Intelligence