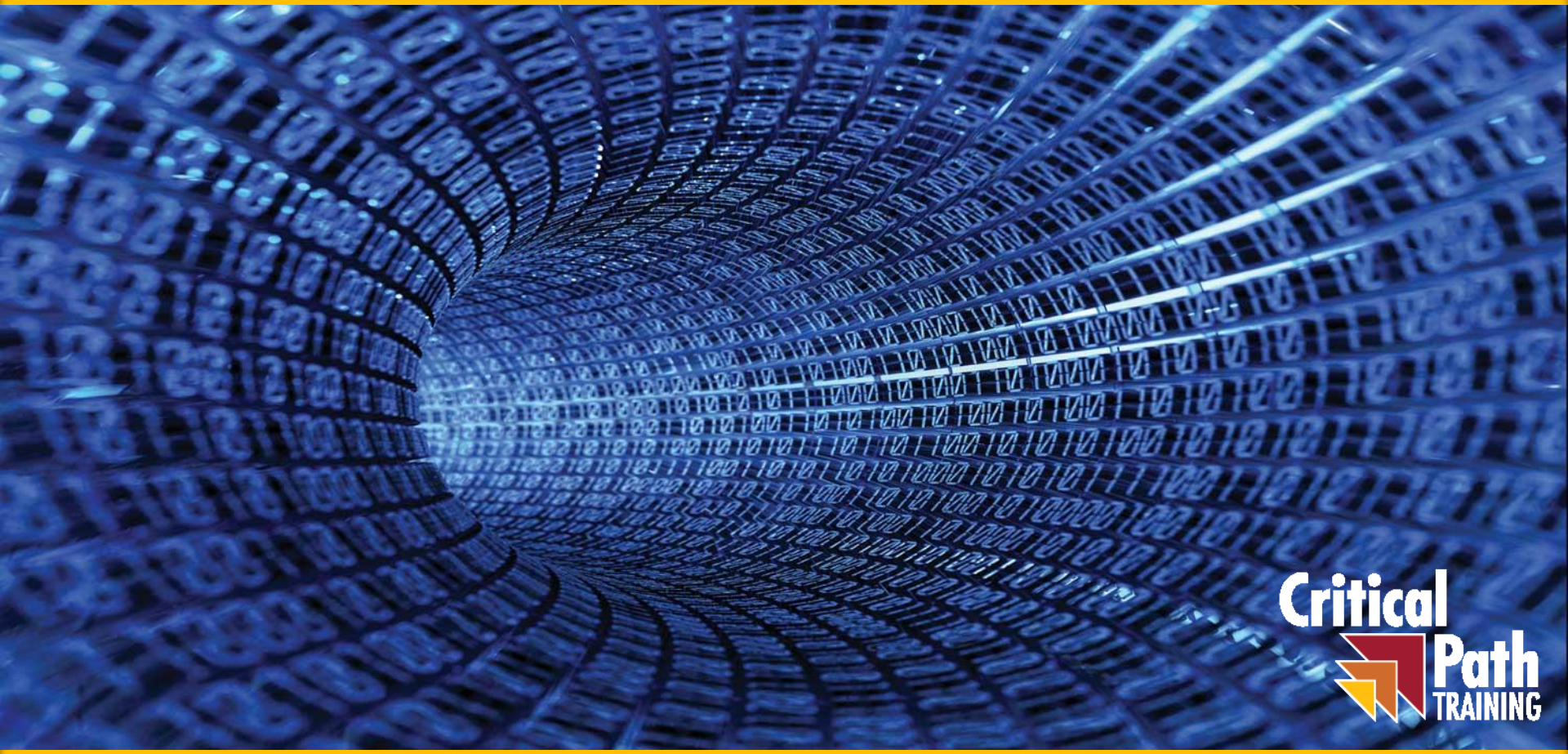
Modeling Data with Hierarchies and Time Intelligence

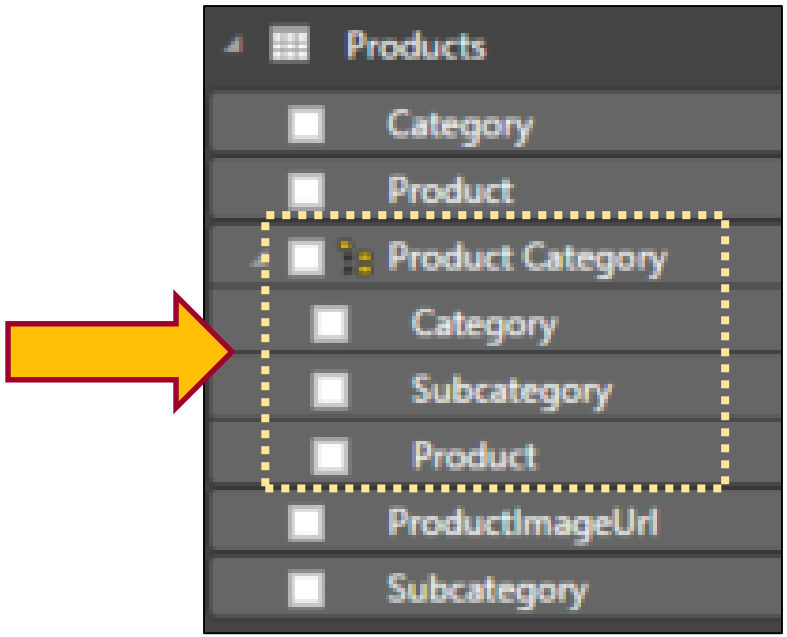


Agenda

- Creating Dimensional Hierarchies
- Understanding the Evaluation Context
- Extending the Data Model using Calendar Tables
- Writing DAX Expressions with Time Intelligence
- Writing DAX Code with Contextual Awareness

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



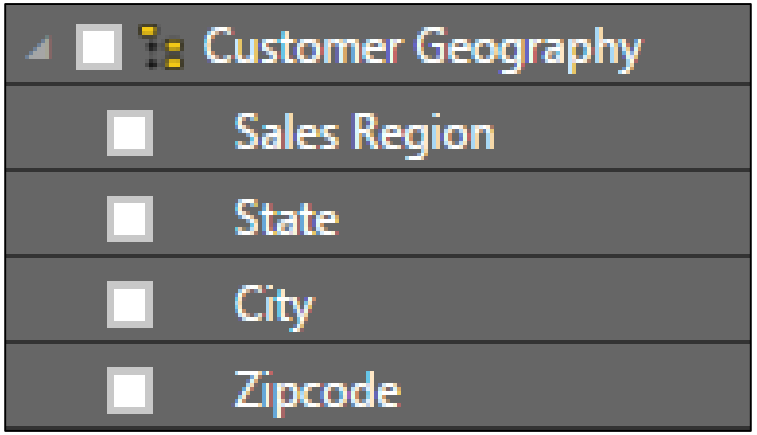
Pulling Columns for Hierarchy into Single Table

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

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

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
Twila Massey	One-time Customer	25	Ages 18 TO 23	Western Region	California

  - Then create hierarchy in the table with all the columns



Agenda

- ✓ Creating Dimensional Hierarchies
- ➢ Understanding the Evaluation Context
- Extending the Data Model using Calendar Tables
- Writing DAX Expressions with Time Intelligence
- Writing DAX Code with Contextual Awareness

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]
```

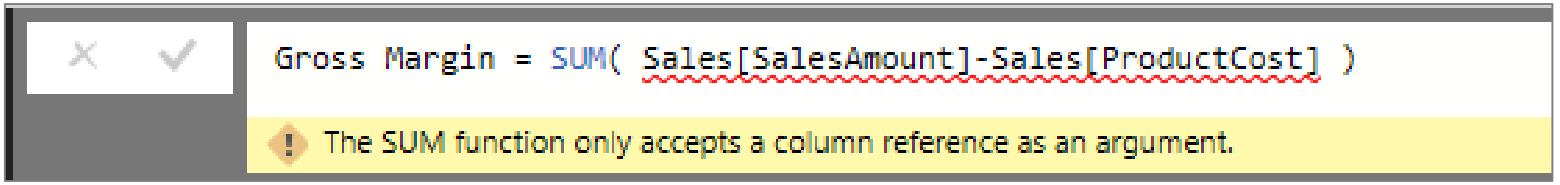
	Age Group	Sales Region	State Name	SalesRegionSort	City
48	Ages 40 TO 49	Western Region	California	1	San Jose, CA
74	Ages 65 and over	Western Region	California	1	San Jose, CA
73	Ages 65 and over	Western Region	California	1	San Jose, CA
25	Ages 18 TO 23	Western Region	California	1	San Jose, CA
61	Ages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

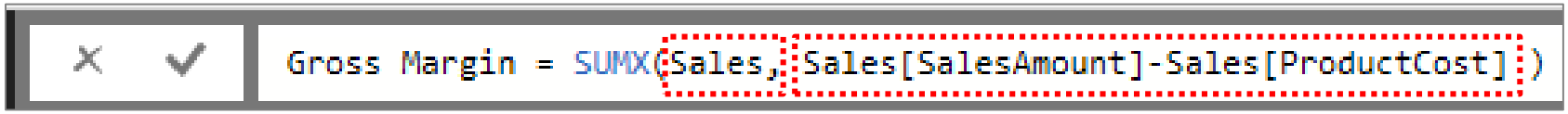
Customer	Customer Type	Age	Age Group	Sales Region	State Name
Lucile Blake	One-time Customer	48	Ages 40 TO 49	Western Region	California
Rochelle Owen	One-time Customer	74	Ages 65 and over	Western Region	California
Corinne Finch	One-time Customer	73	Ages 65 and over	Western Region	California

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



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



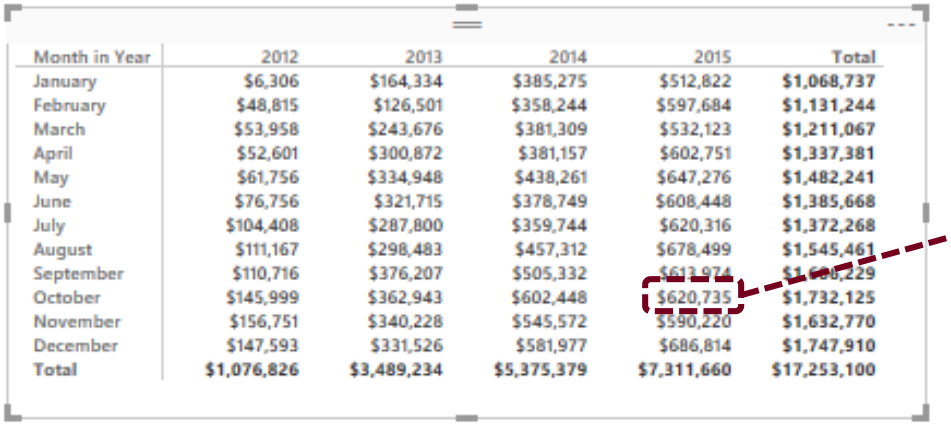
  - First argument accepts expressions that evaluates to table of rows
  - Second argument accepts expression that is evaluated for each row

DAX Table Iterator Functions

- The following DAX functions create row context
  - AVERAGEX
  - COUNTAX
  - COUNTX
  - MAXX
  - MINX
  - SUMX

Understanding Filter Context

- Visuals apply various filters in different evaluation contexts



Month in Year	2012	2013	2014	2015	Total
January	$6,306	$164,334	$385,275	$512,822	$1,068,737
February	$48,815	$126,501	$358,244	$597,684	$1,131,244
March	$53,958	$243,676	$381,309	$532,123	$1,211,067
April	$52,601	$300,872	$381,157	$602,751	$1,337,381
May	$61,756	$334,948	$438,261	$647,276	$1,482,241
June	$76,756	$321,715	$378,749	$608,448	$1,385,668
July	$104,408	$287,800	$359,744	$620,316	$1,372,268
August	$111,167	$298,483	$457,312	$678,499	$1,545,461
September	$110,716	$376,207	$505,332	$613,974	$1,606,229
October	$145,999	$362,943	$602,448	$620,735	$1,732,125
November	$156,751	$340,228	$545,572	$590,220	$1,632,770
December	$147,593	$331,526	$581,977	$686,814	$1,747,910
Total	**$1,076,826**	**$3,489,234**	**$5,375,379**	**$7,311,660**	**$17,253,100**

Filters on this evaluation

[Year] = 2015

[Month in Year] = "October"

- Filter context also affected by slicers and other filters

Sales Region

- Select All
- Central Region
- Eastern Region
- Western Region

Customer Type

- One-time customer
- Repeat Customer

Month in Year	2012	2013	2014	2015	Total
January	$425	$50,169	$61,295	$76,614	$188,503
February	$13,891	$40,133	$63,670	$101,542	$219,236
March	$19,121	$58,411	$73,839	$84,180	$235,551
April	$19,128	$53,711	$67,919	$91,762	$232,520
May	$22,939	$64,259	$78,668	$109,689	$275,555
June	$29,082	$50,564	$73,504	$88,047	$241,197
July	$34,809	$62,971	$69,053	$80,749	$247,582
August	$36,096	$61,217	$76,009	$94,719	$268,041
September	$39,415	$68,653	$82,697	$94,805	$285,570
October	$51,994	$69,122	$99,344	$84,177	$304,637
November	$47,020	$52,548	$85,924	$74,611	$260,102
December	$50,580	$66,260	$102,088	$94,877	$313,804
Total	**$364,500**	**$698,018**	**$934,009**	**$1,075,771**	**$3,072,298**

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] )
    )
)
```

DAX Functions that Return a Table

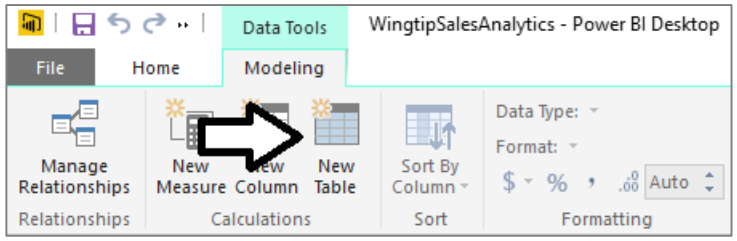
- ALL
- ALLEXCEPT
- CALCULATETABLE
- DISTINCT
- FILTER
- RELATEDTABLE
- VALUES

Agenda

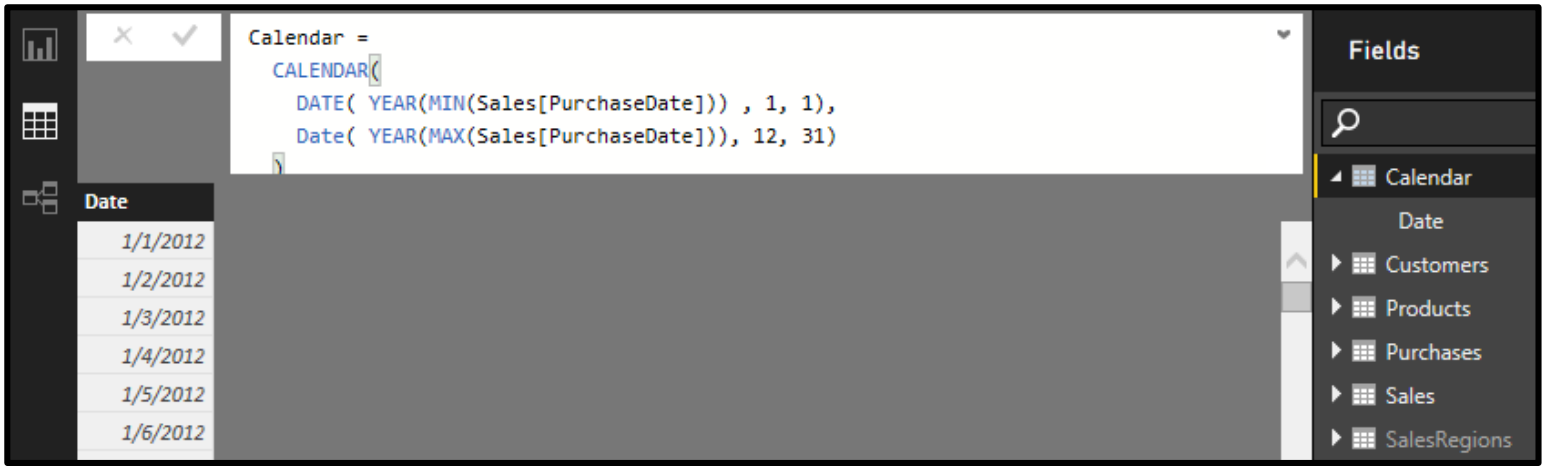
- ✓ Creating Dimensional Hierarchies
- ✓ Understanding the Evaluation Context
- ➢ Extending the Data Model using Calendar Tables
- Writing DAX Expressions with Time Intelligence
- Writing DAX Code with Contextual Awareness

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

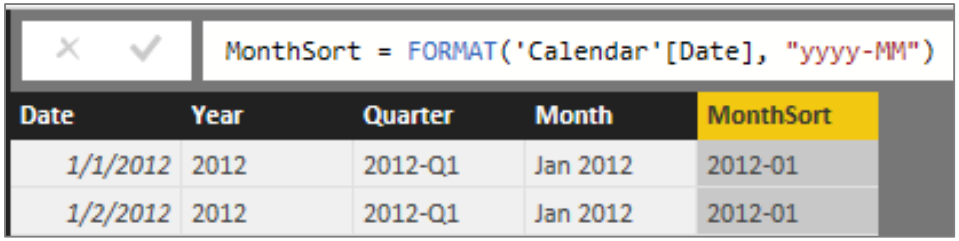
```
Month = FORMAT('Calendar'[Date], "MMM yyyy")
```

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

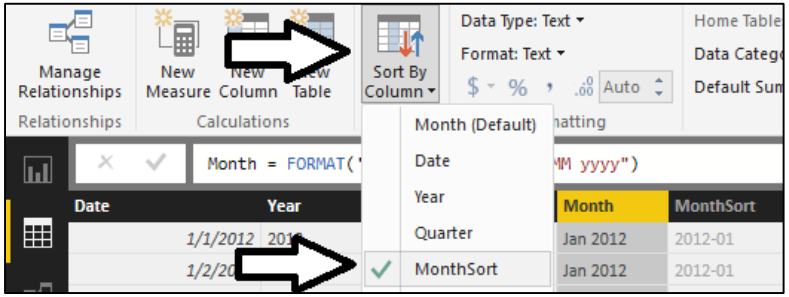
Configuring Sort Columns

- 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



  - 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

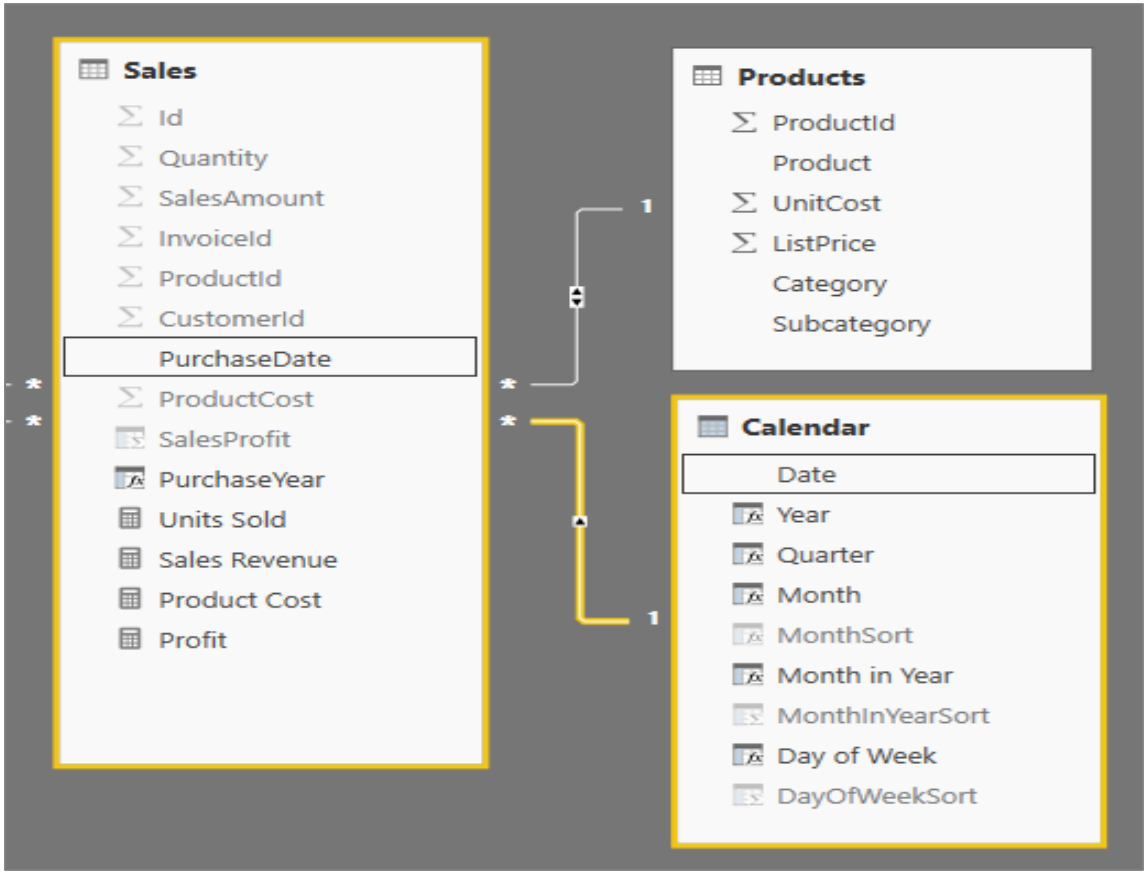
- Creating the **DayOfWeekSort** column

```
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



Creating Visuals with a Calendar Table

- **Year** for row labels and **Month in Year** as column labels

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

- **Month in Year** for row labels and **Year** as column labels

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**

- **Month in Year** for row labels and **Year** as column labels

Day of Week	2012	2013	2014	2015	Total
Mon	$314,471	$801,337	$1,460,373	$1,682,345	**$4,258,527**
Tue	$262,321	$791,863	$1,553,063	$1,726,955	**$4,334,202**
Wed	$269,499	$671,754	$1,525,827	$1,786,688	**$4,253,768**
Thu	$246,499	$777,814	$1,427,989	$1,749,475	**$4,201,776**
Fri	$329,852	$803,028	$1,445,129	$1,790,611	**$4,368,620**
Sat	$289,566	$747,619	$1,447,230	$1,736,439	**$4,220,853**
Sun	$231,779	$762,762	$1,414,640	$1,683,591	**$4,092,772**
Total	**$1,943,986**	**$5,356,177**	**$10,274,251**	**$12,156,103**	**$29,730,517**

Hierarchical Row Labels in a Matrix

- Dimensional hierarchy can be visualized using matrix

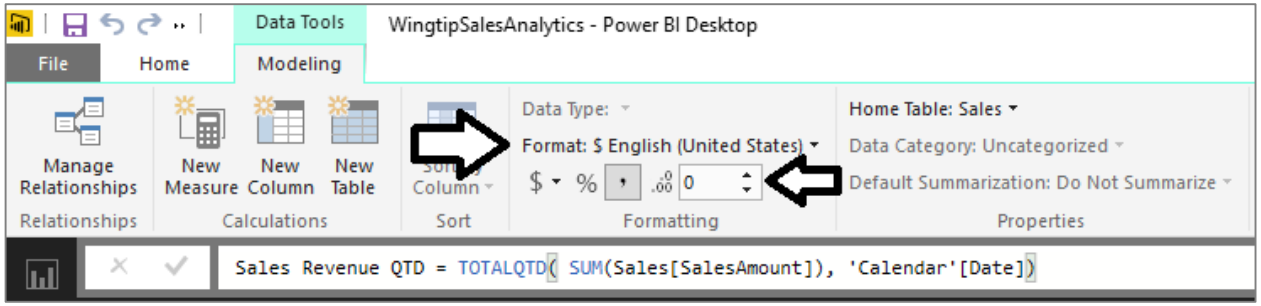
Year	Quarter	Units Sold	Sales Revenue	Product Cost	Profit
2012	2012-Q1	5,023	$85,494	$40,088	$45,406
	2012-Q2	15,845	$260,944	$130,287	$130,657
	2012-Q3	30,979	$557,293	$269,314	$287,979
	2012-Q4	75,386	$1,040,256	$540,222	$500,034
	Total	**127,233**	**$1,943,986**	**$979,909**	**$964,077**
2013	2013-Q1	71,064	$945,310	$517,474	$427,836
	2013-Q2	127,830	$1,111,831	$557,730	$554,101
	2013-Q3	302,557	$1,372,617	$571,187	$801,430
	2013-Q4	494,231	$1,926,420	$864,530	$1,061,889
	Total	**995,682**	**$5,356,177**	**$2,510,921**	**$2,845,256**
2014	2014-Q1	492,123	$1,868,225	$892,244	$975,981
	2014-Q2	542,615	$2,224,114	$1,081,051	$1,143,063
	2014-Q3	417,331	$2,548,569	$1,332,729	$1,215,840
	2014-Q4	642,513	$3,633,343	$1,877,978	$1,755,365
	Total	**2,094,582**	**$10,274,251**	**$5,184,002**	**$5,090,249**
2015	2015-Q1	406,989	$2,604,726	$1,364,369	$1,240,357
	2015-Q2	216,311	$2,206,560	$1,219,892	$986,669
	2015-Q3	308,970	$3,095,046	$1,724,893	$1,370,153
	2015-Q4	402,278	$4,249,771	$2,358,468	$1,891,304
	Total	**1,334,548**	**$12,156,103**	**$6,667,621**	**$5,488,482**
Total		**4,552,045**	**$29,730,517**	**$15,342,453**	**$14,388,064**

Agenda

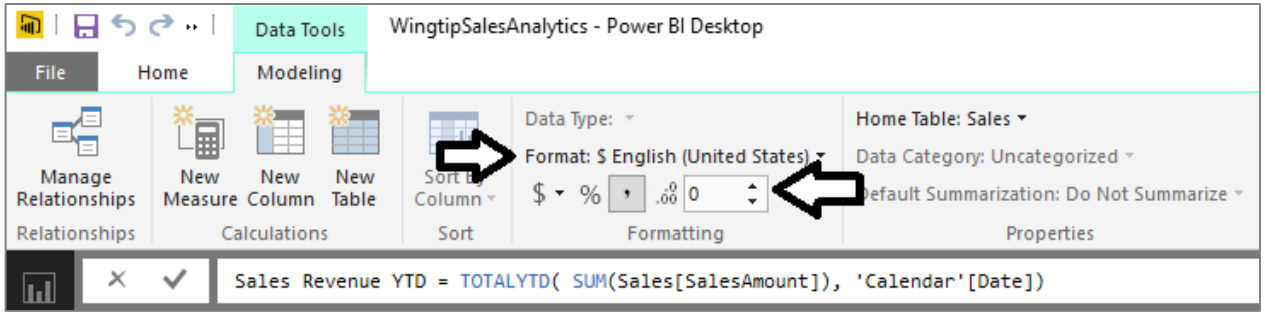
- ✓ Creating Dimensional Hierarchies
- ✓ Understanding the Evaluation Context
- ✓ Extending the Data Model using Calendar Tables
- ➢ Writing DAX Expressions with Time Intelligence
- • Writing DAX Code with Contextual Awareness

Calculated Fields for QTD and YTD Sales

- TOTALQTD function calculates quarter-to-date totals

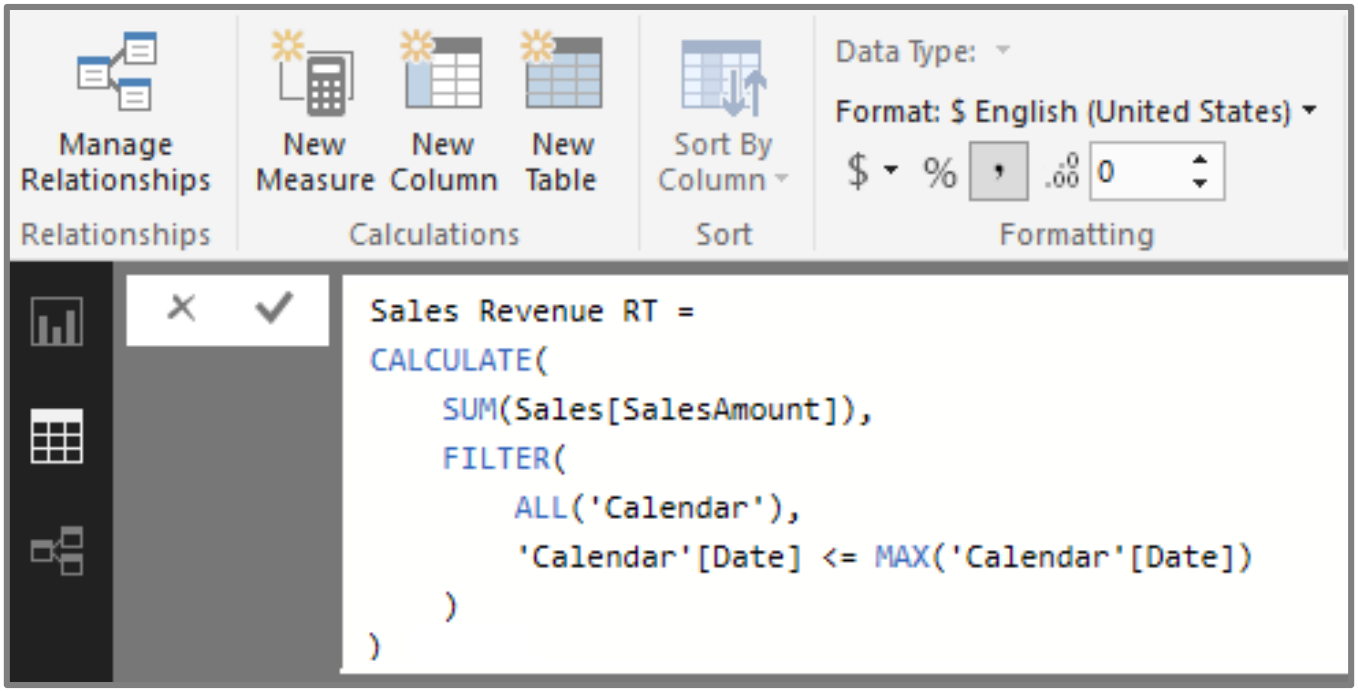


- TOTALYTD function calculates year-to-date totals



Creating Running Total using CALCULATE

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



Matrix Visual with To-Date Running Totals

- Running totals calculated using DAX

Year	Quarter	Month	Sales Revenue	Sales Revenue QTD	Sales Revenue YTD	Sales Revenue RT
2014	2014-Q1	Jan 2014	$629,969	$629,969	$629,969	$7,930,132
		Feb 2014	$609,637	$1,239,606	$1,239,606	$8,539,770
		Mar 2014	$628,618	$1,868,225	$1,868,225	$9,168,388
	2014-Q2	Apr 2014	$661,588	$661,588	$2,529,812	$9,829,976
		May 2014	$748,193	$1,409,780	$3,278,005	$10,578,168
		Jun 2014	$814,333	$2,224,114	$4,092,338	$11,392,502
	2014-Q3	Jul 2014	$788,469	$788,469	$4,880,807	$12,180,970
		Aug 2014	$869,143	$1,657,611	$5,749,950	$13,050,113

- Question: when did Wingtip reach $10,000,000 in sales

Year	Quarter	Month	Sales Revenue	Sales Revenue QTD	Sales Revenue YTD	Sales Revenue RT
2014	2014-Q1	Jan 2014	$629,969	$629,969	$629,969	$7,930,132
		Feb 2014	$609,637	$1,239,606	$1,239,606	$8,539,770
		Mar 2014	$628,618	$1,868,225	$1,868,225	$9,168,388
	2014-Q2	Apr 2014	$661,588	$661,588	$2,529,812	$9,829,976
		May 2014	$748,193	$1,409,780	$3,278,005	$10,578,168
		Jun 2014	$814,333	$2,224,114	$4,092,338	$11,392,502
	2014-Q3	Jul 2014	$788,469	$788,469	$4,880,807	$12,180,970

Agenda

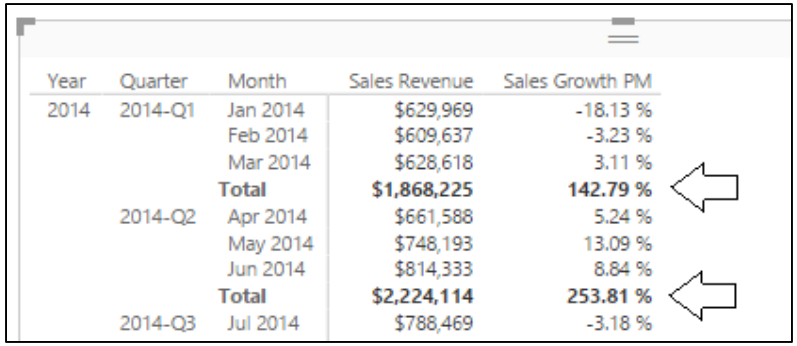
- ✓ Creating Dimensional Hierarchies
- ✓ Understanding the Evaluation Context
- ✓ Extending the Data Model using Calendar Tables
- ✓ Writing DAX Expressions with Time Intelligence
- ➢ Writing DAX Code with Contextual Awareness

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 %

Simulating KPIs with Power BI Desktop

- KPIs are not directly support in data model
  - But you can create something similar using measures

```
Sales Growth PM Eval =
IF( ISNUMBER([Sales Growth PM]),
    SWITCH(TRUE(),
      ([Sales Growth PM] >= 0.2), "EXCELLENT",
      ([Sales Growth PM] >= 0.1), "GOOD",
      ([Sales Growth PM] >= 0), "OK",
      ([Sales Growth PM] <  0), "BAD"
  ))
```

Year	Quarter	Month	Sales Revenue	Sales Growth PM	Sales Growth PM Eval
2014	2014-Q1	Jan 2014	$629,969	-18.13 %	AWFUL
		Feb 2014	$609,637	-3.23 %	BAD
		Mar 2014	$628,618	3.11 %	OK
		Total	**$1,868,225**		
	2014-Q2	Apr 2014	$661,588	5.24 %	OK
		May 2014	$748,193	13.09 %	GOOD
		Jun 2014	$814,333	8.84 %	OK
		Total	**$2,224,114**		
	2014-Q3	Jul 2014	$788,469	-3.18 %	BAD
		Aug 2014	$869,143	10.23 %	GOOD
		Sep 2014	$890,958	2.51 %	OK
		Total	**$2,548,569**		

Summary

- ✓ Creating Dimensional Hierarchies
- ✓ Understanding the Evaluation Context
- ✓ Extending the Data Model using Calendar Tables
- ✓ Writing DAX Expressions with Time Intelligence
- ✓ Writing DAX Code with Contextual Awareness