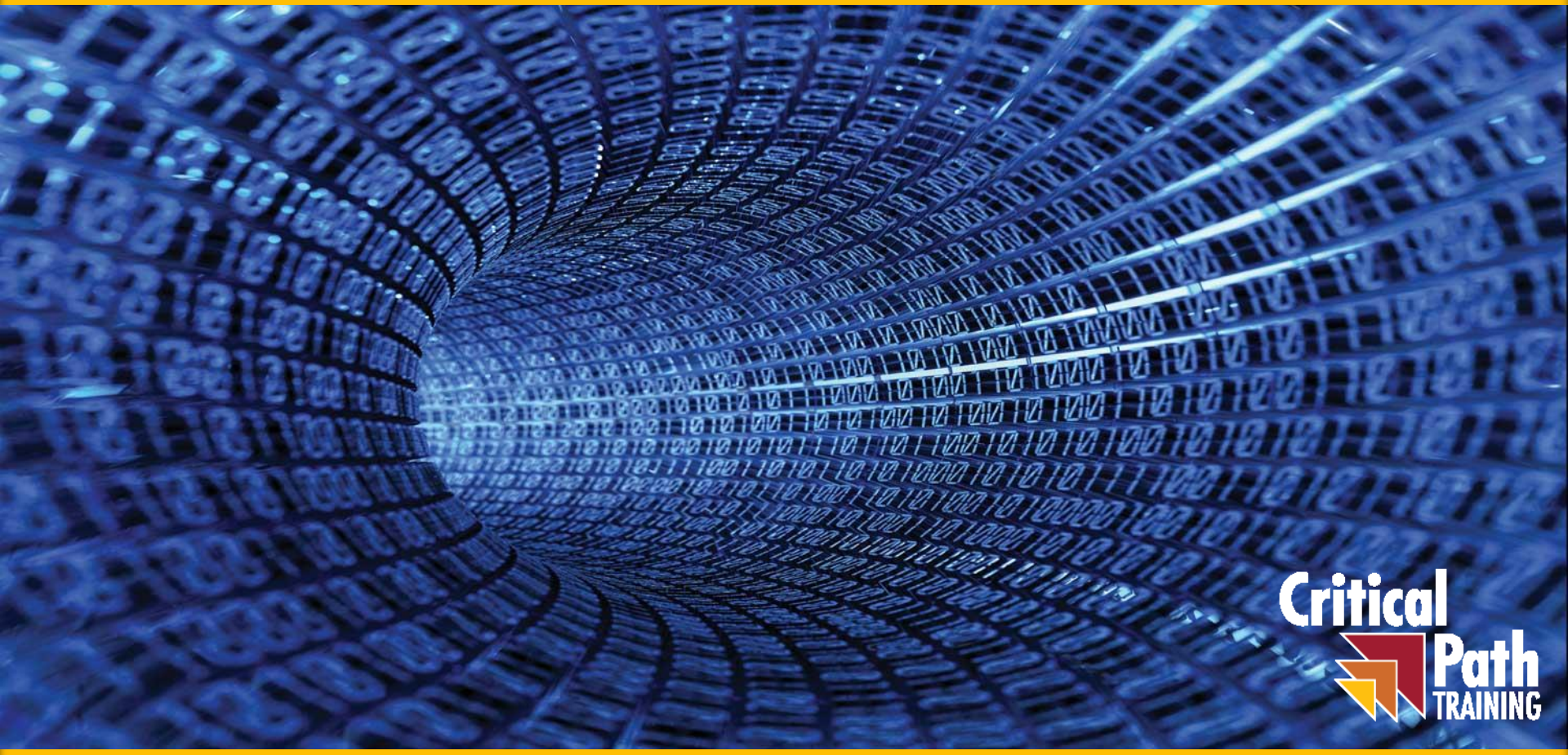
# Designing a Data Model with Power BI Desktop



# Agenda

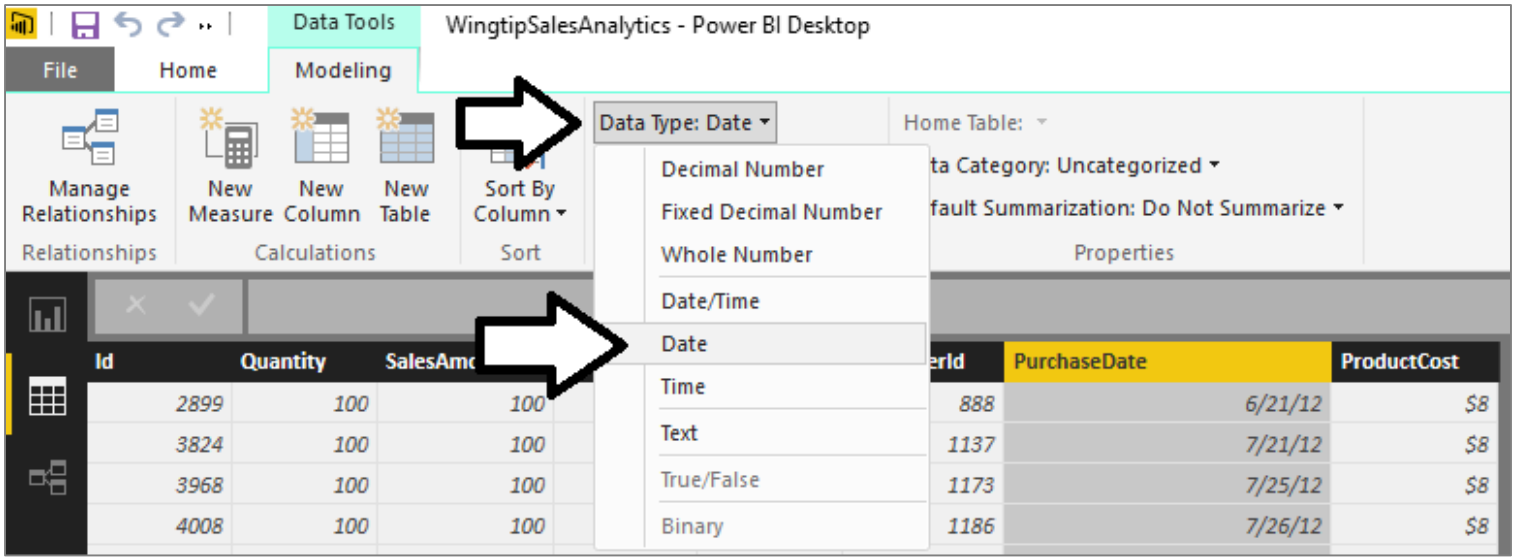
- Modeling Data in Power BI Desktop
- Understanding Table Relationships
- Writing DAX Expressions
- Creating Calculated Columns
- Creating Measures to Aggregate Data
- Adding Geographic Fields to a Data Model

# Data Modeling with Power BI Desktop

- Steps to create a data model with Power Pivot
  - Create relationships between tables
  - Modify columns (rename, set formatting, convert type)
  - Create calculated columns
  - Create measures
  - Add column metadata
  - Create dimensional hierarchies
  - Add Calendar table(s)

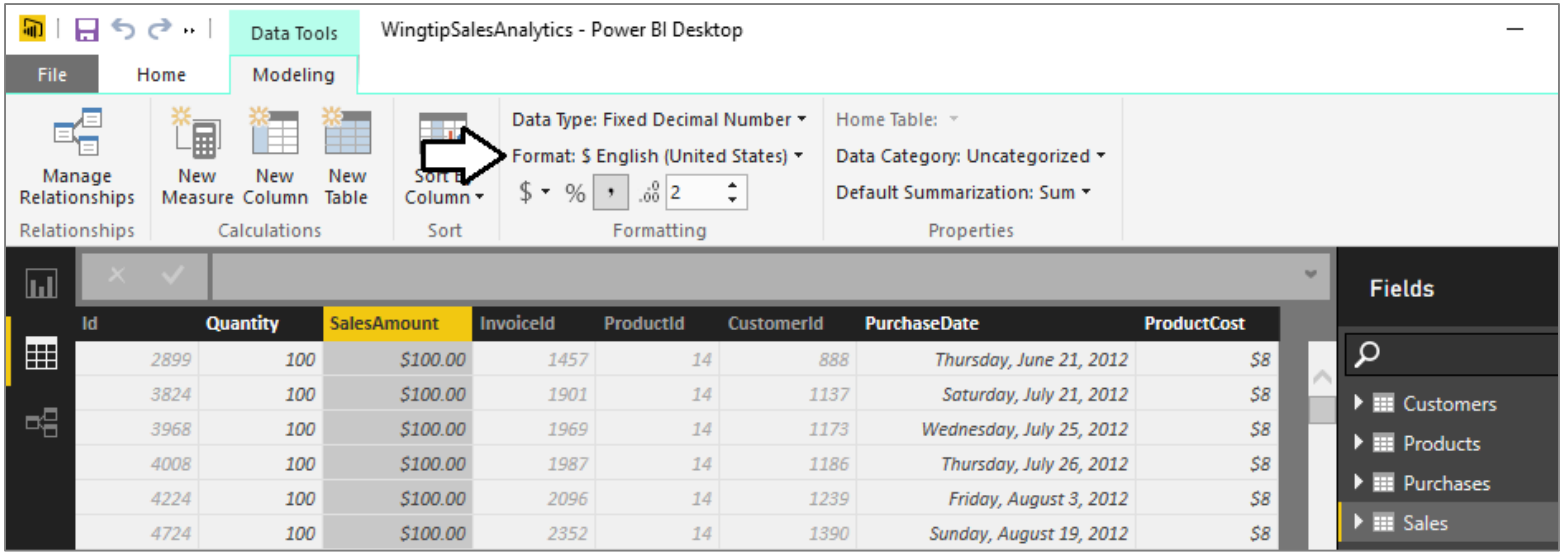
# Converting Column Types

- Power Pivot allows you to convert columns
  - Alternative to converting column with Power Query



# Formatting Columns

- Each column has its own formatting properties
  - Formatting propagated to reports and visuals
  - Makes it easier on data model consumers



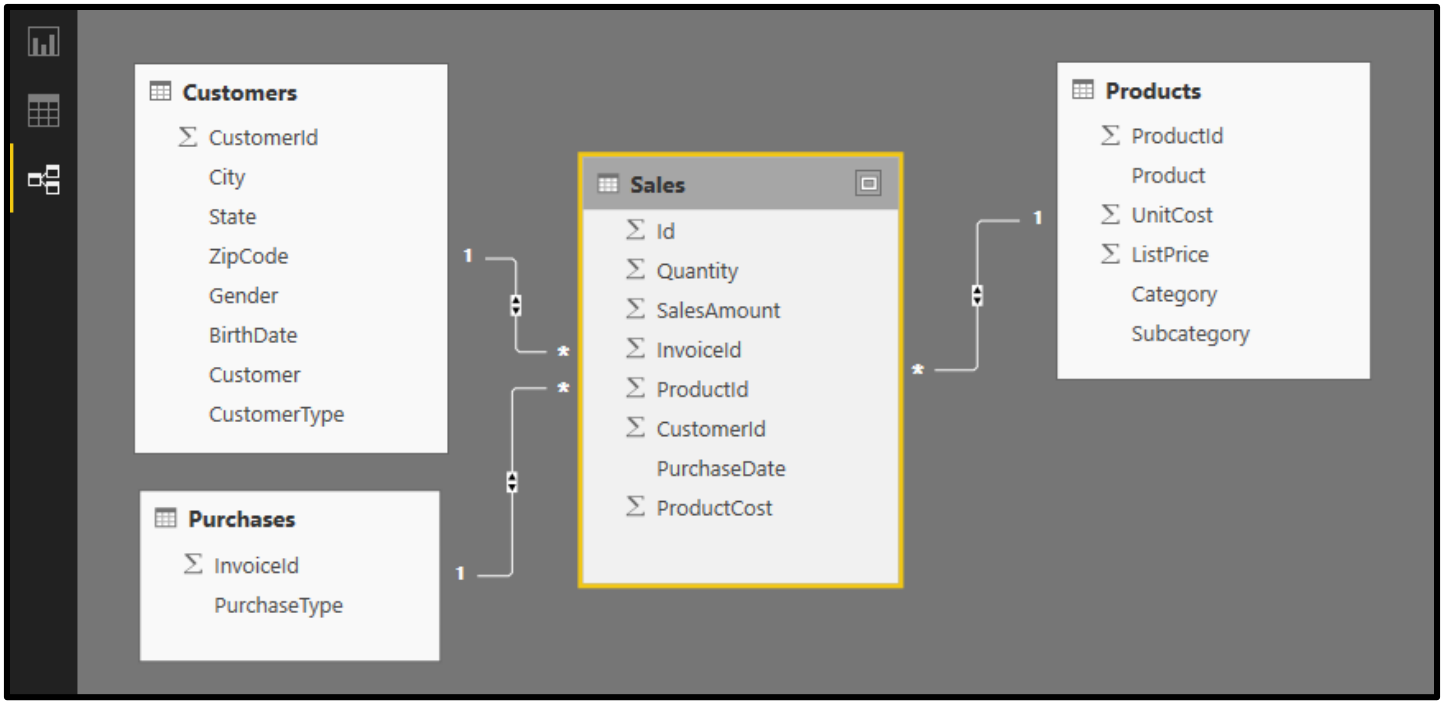

| Id | Quantity | SalesAmount | InvoiceId | ProductId | CustomerId | PurchaseDate | ProductCost |
|---|---|---|---|---|---|---|---|
| 2899 | 100 | $100.00 | 1457 | 14 | 888 | Thursday, June 21, 2012 | $8 |
| 3824 | 100 | $100.00 | 1901 | 14 | 1137 | Saturday, July 21, 2012 | $8 |
| 3968 | 100 | $100.00 | 1969 | 14 | 1173 | Wednesday, July 25, 2012 | $8 |
| 4008 | 100 | $100.00 | 1987 | 14 | 1186 | Thursday, July 26, 2012 | $8 |
| 4224 | 100 | $100.00 | 2096 | 14 | 1239 | Friday, August 3, 2012 | $8 |
| 4724 | 100 | $100.00 | 2352 | 14 | 1390 | Sunday, August 19, 2012 | $8 |

# Agenda

- ✓ Modeling Data in Power BI Desktop
- ➢ Understanding Table Relationships
- Writing DAX Expressions
- Creating Calculated Columns
- Creating Measures to Aggregate Data
- Adding Geographic Fields to a Data Model

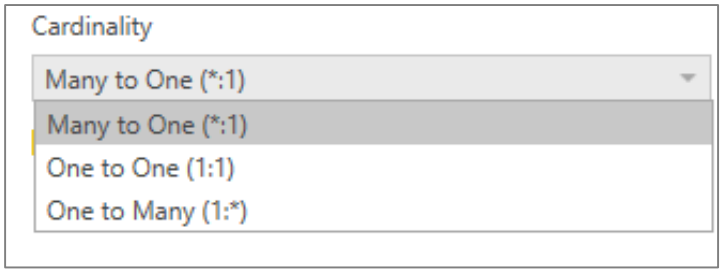
# Table Relationships

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

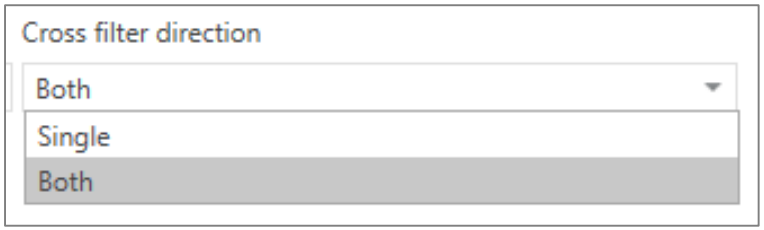


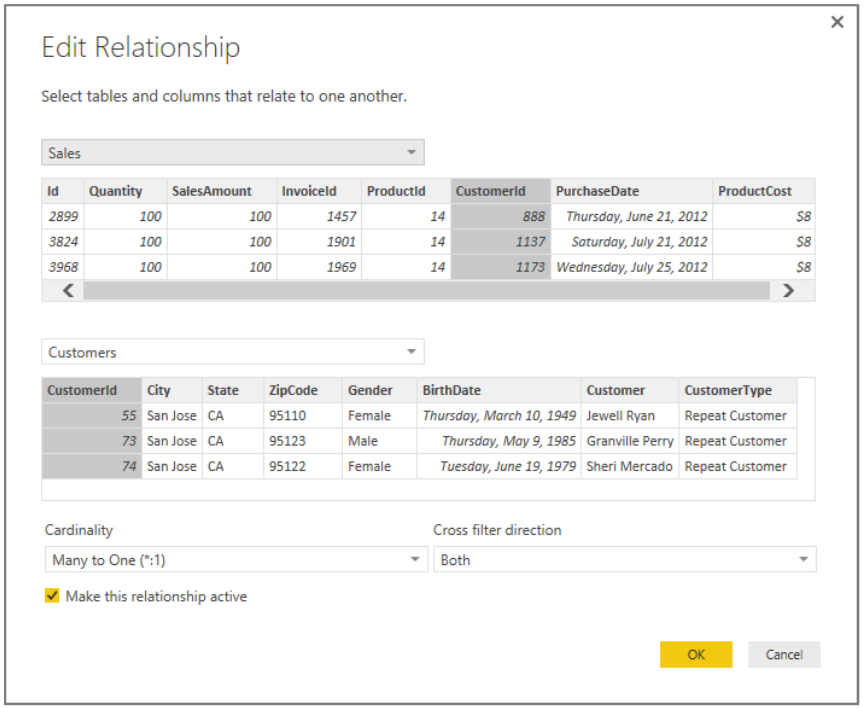
# Relationship Properties

- Cardinality




- Cross filter direction



# Agenda

- ✓ Modeling Data in Power BI Desktop
- ✓ Understanding Table Relationships
- ➢ Writing DAX Expressions
- Creating Calculated Columns
- Creating Measures to Aggregate Data
- Adding Geographic Fields to a Data Model

# Working with DAX

- DAX is the language used to create data model
  - 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()
)
```

# Types of DAX Functions

- Date and Time Functions
- Information Functions
- Logical Functions
- Mathematical and Trigonometric Functions
- Statistical Functions
- Filter Functions
- Text Functions
- Time Intelligence Functions

# 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>
```

# Agenda

- ✓ Modeling Data in Power BI Desktop
- ✓ Understanding Table Relationships
- ✓ Writing DAX Expressions
- ➢ Creating Calculated Columns
- Creating Measures to Aggregate Data
- Adding Geographic Fields to a Data Model

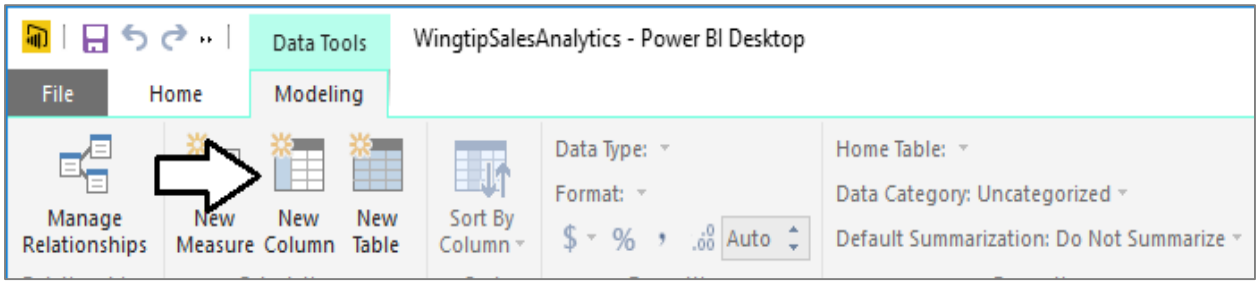
# When to Create Calculated Columns

- 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

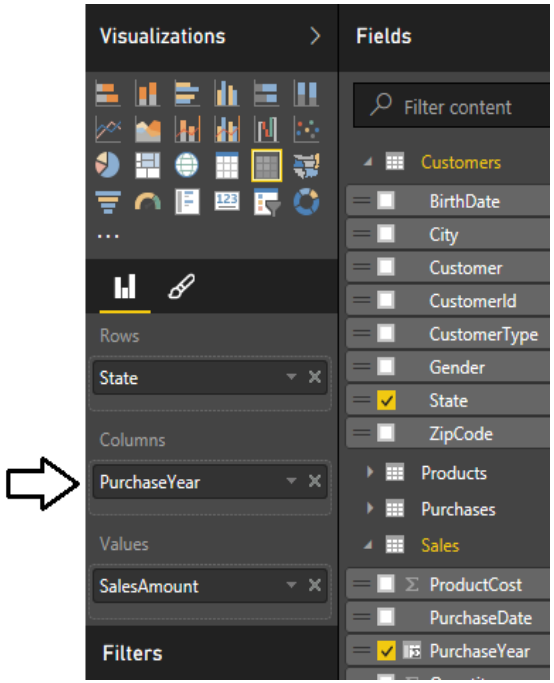



```
PurchaseYear = YEAR(Sales[PurchaseDate])
```

| Id | Quantity | SalesAmount | InvoiceId | ProductId | CustomerId | PurchaseDate | ProductCost | SalesProfit | PurchaseYear |
|---|---|---|---|---|---|---|---|---|---|
| 2899 | 100 | $100.00 | 1457 | 14 | 888 | 6/21/12 | $8.00 | $92.00 | 2012 |
| 3824 | 100 | $100.00 | 1901 | 14 | 1137 | 7/21/12 | $8.00 | $92.00 | 2012 |
| 3968 | 100 | $100.00 | 1969 | 14 | 1173 | 7/25/12 | $8.00 | $92.00 | 2012 |
| 4008 | 100 | $100.00 | 1987 | 14 | 1186 | 7/26/12 | $8.00 | $92.00 | 2012 |
| 4224 | 100 | $100.00 | 2096 | 14 | 1239 | 8/3/12 | $8.00 | $92.00 | 2012 |
| 4724 | 100 | $100.00 | 2352 | 14 | 1390 | 8/19/12 | $8.00 | $92.00 | 2012 |

# Calculated Column as a Column Label

- Calculate column can serve as...
  - Row labels
  - Column labels




| State | 2012 | 2013 | 2014 | 2015 | Total |
|---|---|---|---|---|---|
| CA | $270,926.32 | $550,160.02 | $737,878.53 | $770,402.11 | $2,329,366.98 |
| TX | $212,085.08 | $490,643.98 | $683,079.11 | $919,030.36 | $2,304,838.53 |
| FL | $51,730.85 | $300,866.87 | $535,693.94 | $891,344.92 | $1,779,636.58 |
| NC | $11,018.02 | $164,804.24 | $315,139.92 | $448,638.72 | $939,600.90 |
| NY | $24,207.43 | $165,046.23 | $256,294.27 | $430,971.24 | $876,519.17 |
| GA | $40,305.80 | $152,807.51 | $239,451.05 | $417,037.28 | $849,601.64 |

# 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 | CustomerType | Age |
|---|---|---|---|---|---|---|---|---|
| 55 | San Jose | CA | 95110 | Female | 3/10/49 | Jewell Ryan | Repeat Customer | 66 |
| 73 | San Jose | CA | 95123 | Male | 5/9/85 | Granville Perry | Repeat Customer | 30 |
| 74 | San Jose | CA | 95122 | Female | 6/19/79 | Sheri Mercado | Repeat Customer | 36 |
| 78 | San Jose | CA | 95110 | Male | 6/16/78 | Raleigh Olson | Repeat Customer | 37 |
| 136 | San Jose | CA | 95124 | Female | 1/2/45 | Carrie Foreman | Repeat Customer | 70 |
| 150 | San Jose | CA | 95134 | Female | 8/11/84 | Renee McMillan | Repeat Customer | 31 |

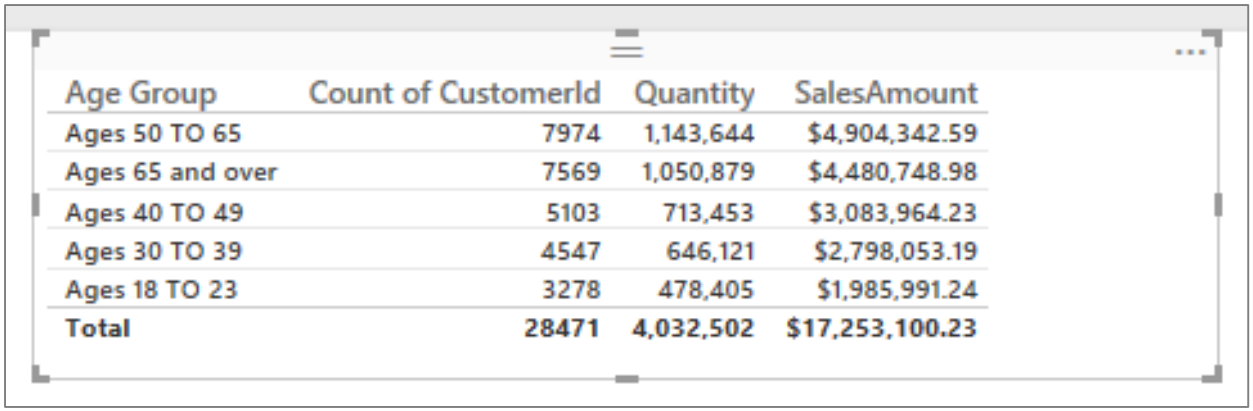
2. Calculate age groups using calculated column

```
Age Group =
  SWITCH( TRUE(),
    [Age] >= 65, "Ages 65 and over",
    [Age] >= 50, "Ages 50 TO 65",
    [Age] >= 40, "Ages 40 TO 49",
    [Age] >= 30, "Ages 30 TO 39",
    [Age] >= 18, "Ages 18 TO 23",
    [Age] < 18, "Ages under 18"
  )
```

| CustomerId | City | State | ZipCode | Gender | BirthDate | Customer | CustomerType | Age | Age Group |
|---|---|---|---|---|---|---|---|---|---|
| 55 | San Jose | CA | 95110 | Female | 3/10/49 | Jewell Ryan | Repeat Customer | 66 | Ages 65 and over |
| 73 | San Jose | CA | 95123 | Male | 5/9/85 | Granville Perry | Repeat Customer | 30 | Ages 30 TO 39 |
| 74 | San Jose | CA | 95122 | Female | 6/19/79 | Sheri Mercado | Repeat Customer | 36 | Ages 30 TO 39 |
| 78 | San Jose | CA | 95110 | Male | 6/16/78 | Raleigh Olson | Repeat Customer | 37 | Ages 30 TO 39 |
| 136 | San Jose | CA | 95124 | Female | 1/2/45 | Carrie Foreman | Repeat Customer | 70 | Ages 65 and over |
| 150 | San Jose | CA | 95134 | Female | 8/11/84 | Renee McMillan | Repeat Customer | 31 | Ages 30 TO 39 |
| 178 | San Jose | CA | 95123 | Male | 9/7/89 | Wayne Gordon | Repeat Customer | 26 | Ages 18 TO 23 |
| 183 | San Jose | CA | 95110 | Female | 5/26/85 | Luella Vinson | Repeat Customer | 30 | Ages 30 TO 39 |
| 213 | San Jose | CA | 95123 | Male | 8/13/78 | Rosario Knight | Repeat Customer | 37 | Ages 30 TO 39 |

# Calculated Column as a Row Label

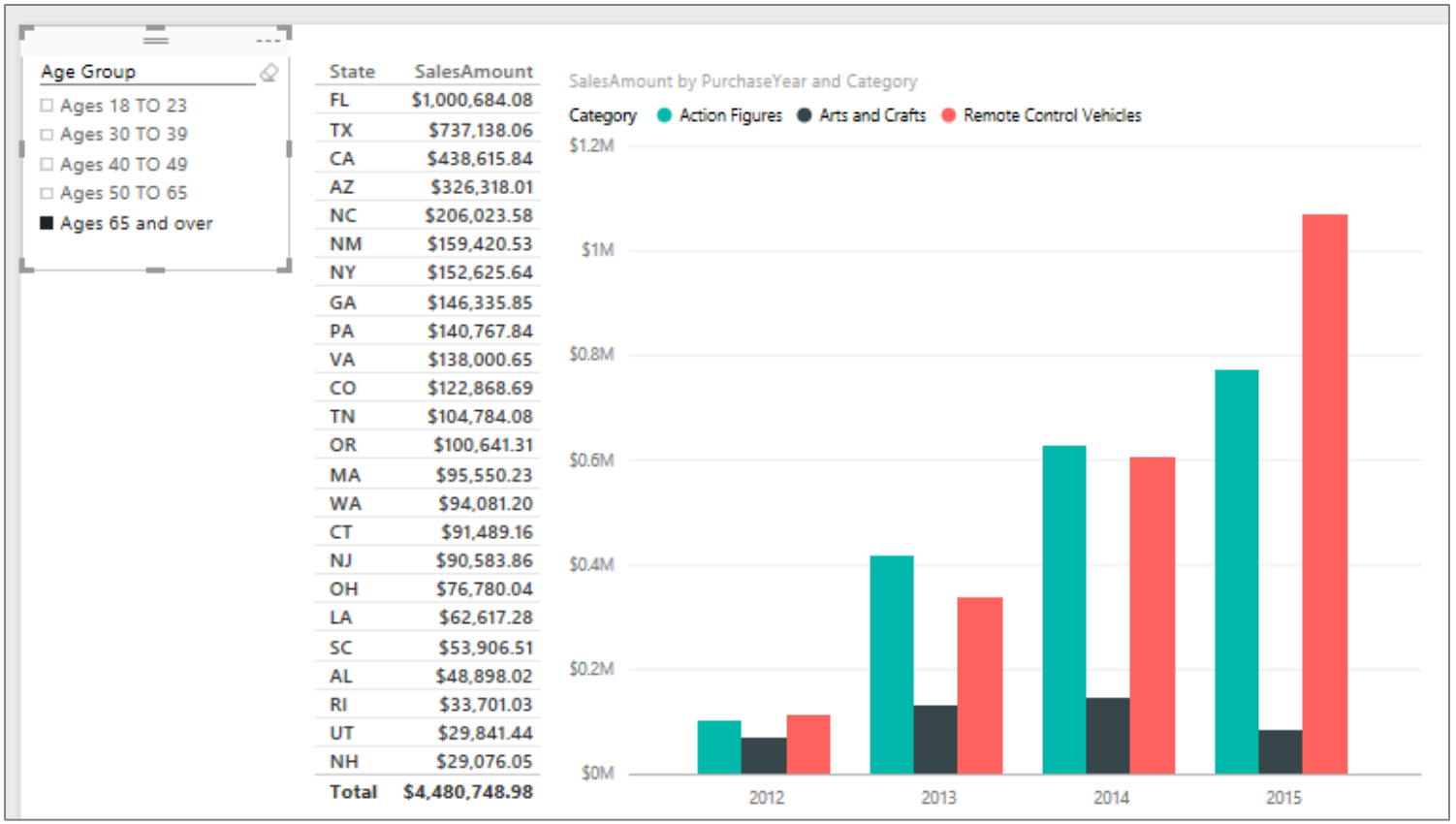
- Age Group can now be used as row label



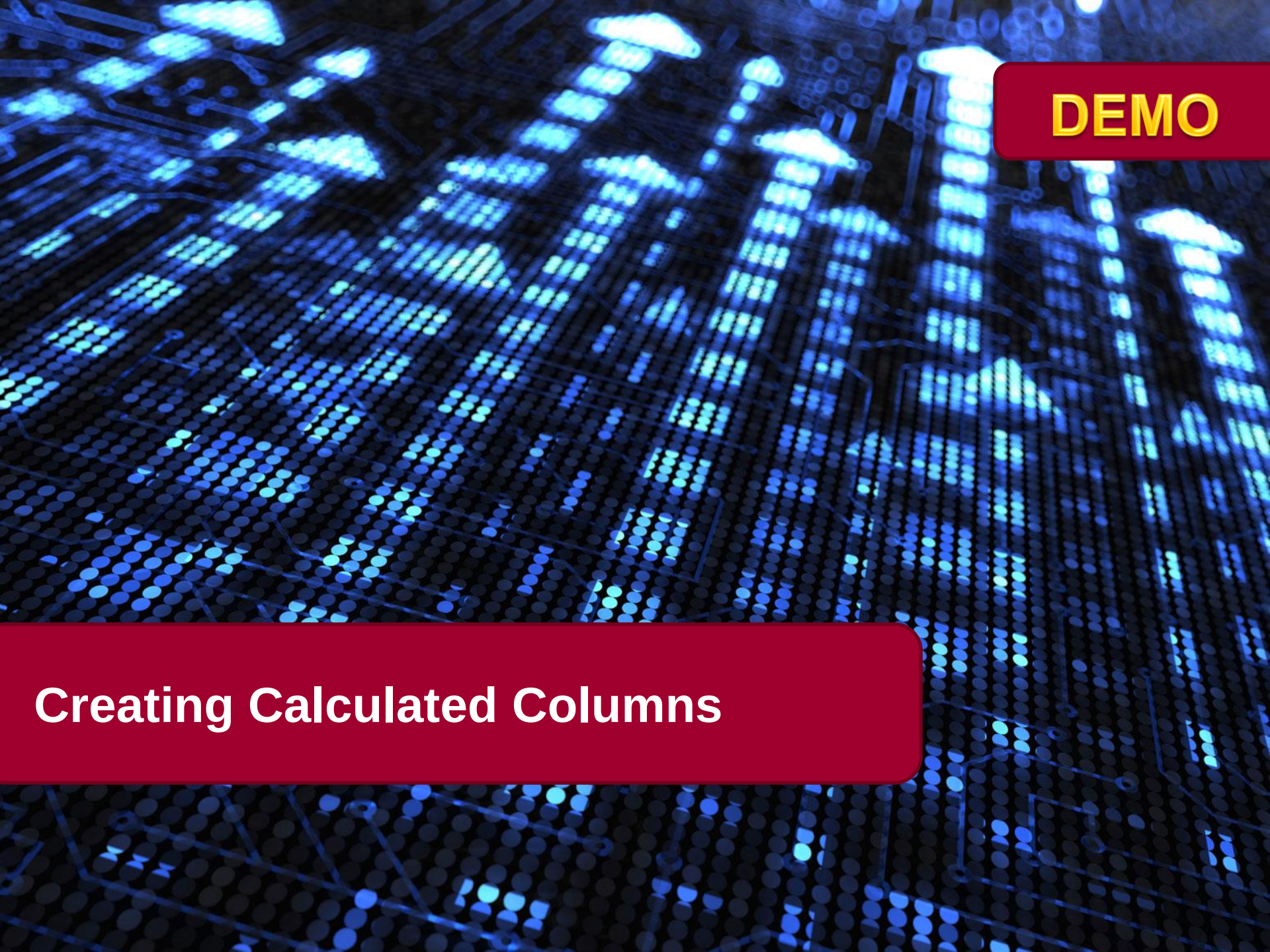
| Age Group | Count of CustomerId | Quantity | SalesAmount |
|---|---|---|---|
| Ages 50 TO 65 | 7974 | 1,143,644 | $4,904,342.59 |
| Ages 65 and over | 7569 | 1,050,879 | $4,480,748.98 |
| Ages 40 TO 49 | 5103 | 713,453 | $3,083,964.23 |
| Ages 30 TO 39 | 4547 | 646,121 | $2,798,053.19 |
| Ages 18 TO 23 | 3278 | 478,405 | $1,985,991.24 |
| **Total** | **28471** | **4,032,502** | **$17,253,100.23** |

# Calculated Column used in a Slicer

- Calculated column can populate slicer values

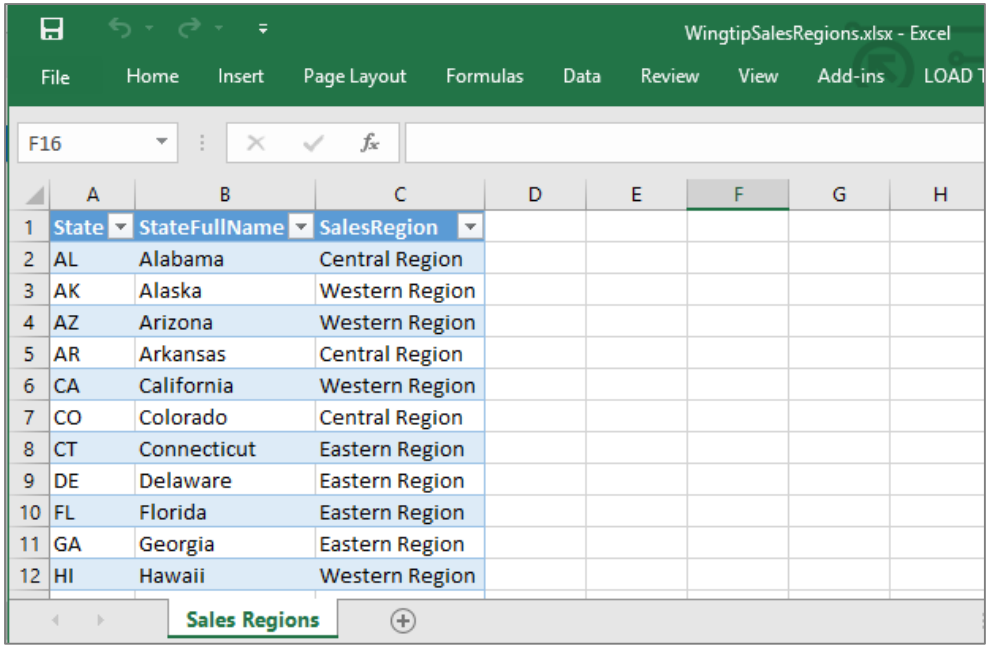



| State | SalesAmount |
|---|---|
| FL | $1,000,684.08 |
| TX | $737,138.06 |
| CA | $438,615.84 |
| AZ | $326,318.01 |
| NC | $206,023.58 |
| NM | $159,420.53 |
| NY | $152,625.64 |
| GA | $146,335.85 |
| PA | $140,767.84 |
| VA | $138,000.65 |
| CO | $122,868.69 |
| TN | $104,784.08 |
| OR | $100,641.31 |
| 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** |


DEMO
Creating Calculated Columns

# Adding Lookup Tables to the Data Model

- Data modeling might required adding lookup tables
  - Lookup tables inject extra related data into data model
- Example: Sales Regions table
  - Assign each state to specific sales region
  - Include full state name it required in reporting




| | State | StateFullName | SalesRegion |
|---|---|---|---|
| 2 | AL | Alabama | Central Region |
| 3 | AK | Alaska | Western Region |
| 4 | AZ | Arizona | Western Region |
| 5 | AR | Arkansas | Central Region |
| 6 | CA | California | Western Region |
| 7 | CO | Colorado | Central Region |
| 8 | CT | Connecticut | Eastern Region |
| 9 | DE | Delaware | Eastern Region |
| 10 | FL | Florida | Eastern Region |
| 11 | GA | Georgia | Eastern Region |
| 12 | HI | Hawaii | Western Region |

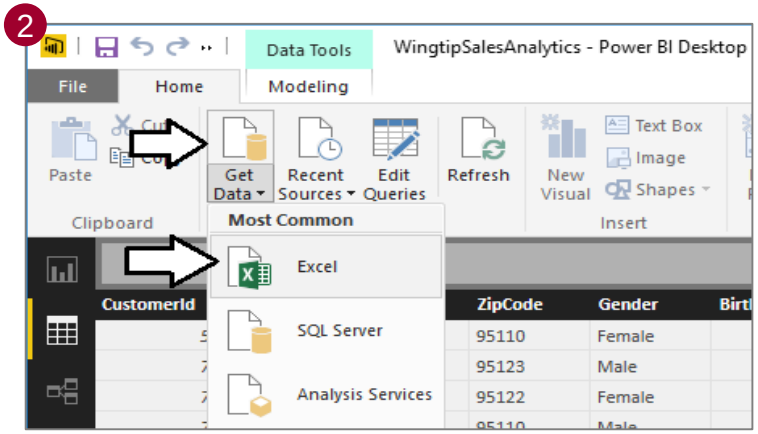
# Importing the SalesRegions Table from Excel

- Import table from Excel using Power Query

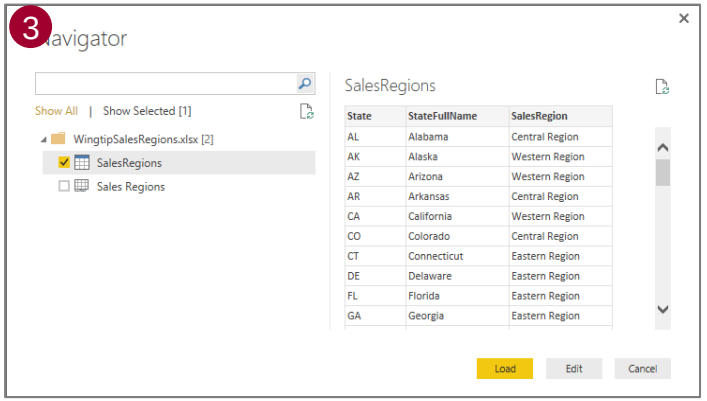
1

| | A | B | C |
|---|---|---|---|
| 1 | State | StateFullName | SalesRegion |
| 2 | AL | Alabama | Central Region |
| 3 | AK | Alaska | Western Region |
| 4 | AZ | Arizona | Western Region |
| 5 | AR | Arkansas | Central Region |
| 6 | CA | California | Western Region |
| 7 | CO | Colorado | Central Region |
| 8 | CT | Connecticut | Eastern Region |
| 9 | DE | Delaware | Eastern Region |
| 10 | FL | Florida | Eastern Region |
| 11 | GA | Georgia | Eastern Region |
| 12 | HI | Hawaii | Western Region |

2

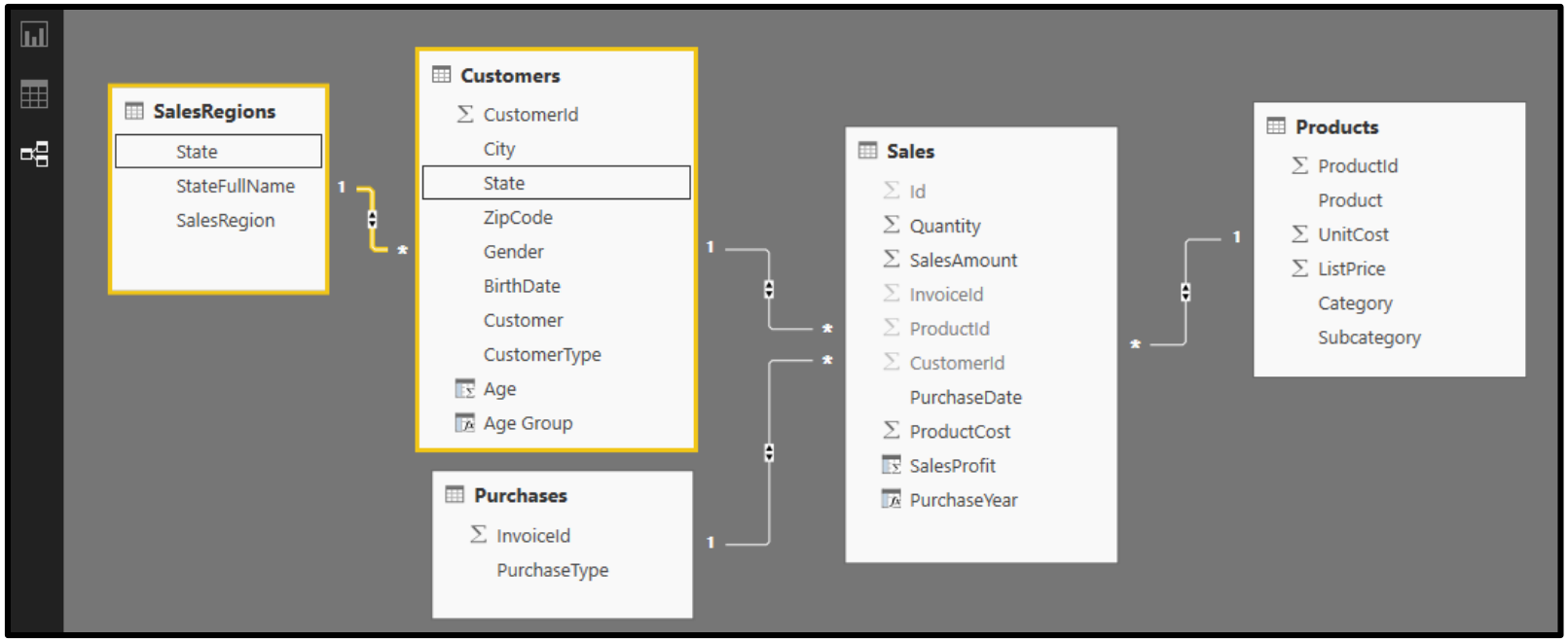



3



# 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

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

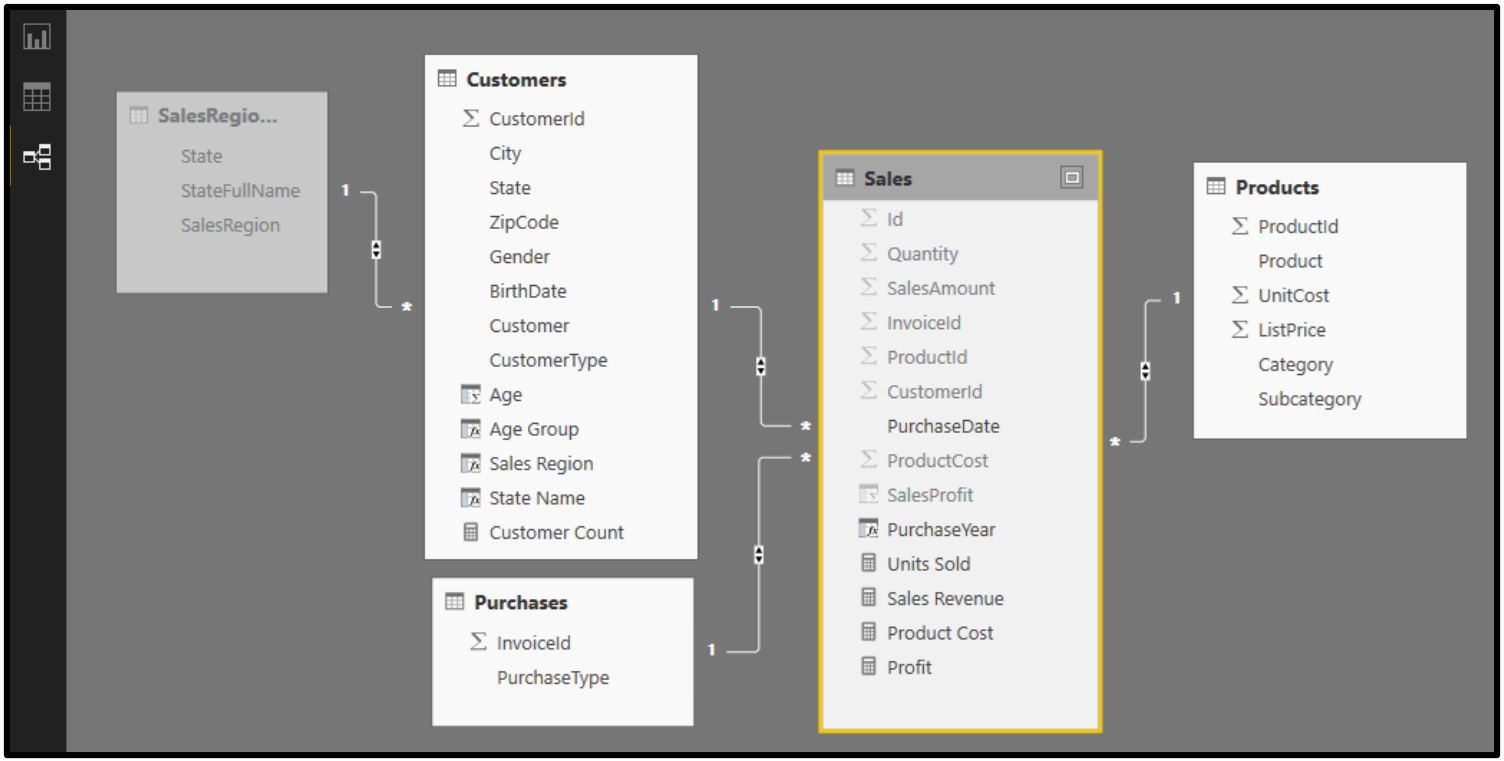
| CustomerId | City | State | ZipCode | Gender | BirthDate | Customer | CustomerType | Age | Age Group | Sales Region |
|---|---|---|---|---|---|---|---|---|---|---|
| 55 | San Jose | CA | 95110 | Female | 3/10/49 | Jewell Ryan | Repeat Customer | 66 | Ages 65 and over | Western Region |
| 73 | San Jose | CA | 95123 | Male | 5/9/85 | Granville Perry | Repeat Customer | 30 | Ages 30 TO 39 | Western Region |
| 74 | San Jose | CA | 95122 | Female | 6/19/79 | Sheri Mercado | Repeat Customer | 36 | Ages 30 TO 39 | Western Region |
| 78 | San Jose | CA | 95110 | Male | 6/16/78 | Raleigh Olson | Repeat Customer | 37 | Ages 30 TO 39 | Western Region |
| 136 | San Jose | CA | 95124 | Female | 1/2/45 | Carrie Foreman | Repeat Customer | 70 | Ages 65 and over | Western Region |
| 150 | San Jose | CA | 95134 | Female | 8/11/84 | Renee McMillan | Repeat Customer | 31 | Ages 30 TO 39 | Western Region |

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

| State | ZipCode | Gender | BirthDate | Customer | CustomerType | Age | Age Group | Sales Region | State Name |
|---|---|---|---|---|---|---|---|---|---|
| CA | 95110 | Female | 3/10/49 | Jewell Ryan | Repeat Customer | 66 | Ages 65 and over | Western Region | California |
| CA | 95123 | Male | 5/9/85 | Granville Perry | Repeat Customer | 30 | Ages 30 TO 39 | Western Region | California |
| CA | 95122 | Female | 6/19/79 | Sheri Mercado | Repeat Customer | 36 | Ages 30 TO 39 | Western Region | California |
| CA | 95110 | Male | 6/16/78 | Raleigh Olson | Repeat Customer | 37 | Ages 30 TO 39 | Western Region | California |
| CA | 95124 | Female | 1/2/45 | Carrie Foreman | Repeat Customer | 70 | Ages 65 and over | Western Region | California |
| CA | 95134 | Female | 8/11/84 | Renee McMillan | Repeat Customer | 31 | Ages 30 TO 39 | Western Region | California |

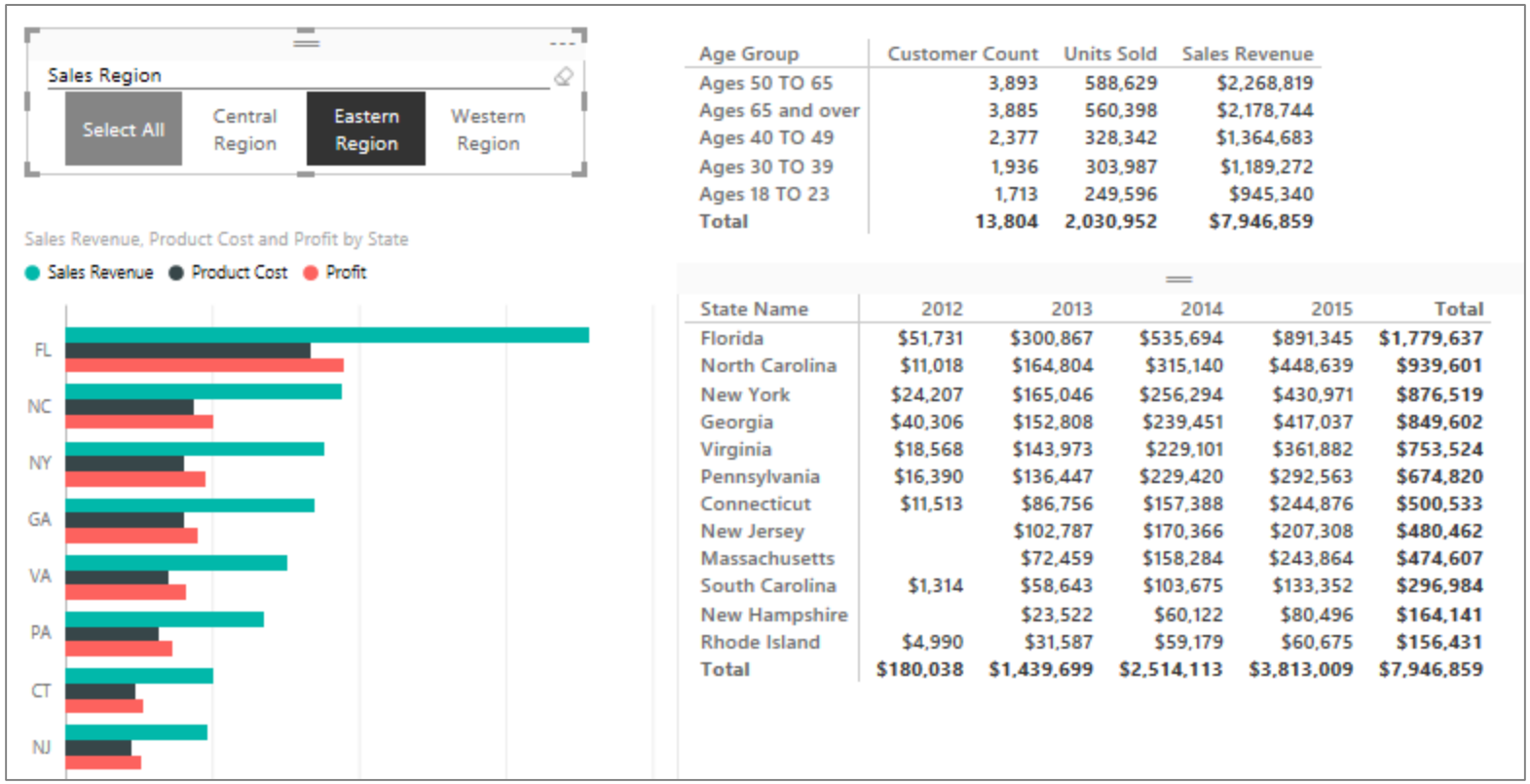
# Hiding the Lookup Table

- Lookup table can often be hidden
  - simplifies reporting for data model consumers



# Filtering on Sales Region

- Calculated column used to calculate slicer values




| Age Group | Customer Count | Units Sold | Sales Revenue |
|---|---|---|---|
| Ages 50 TO 65 | 3,893 | 588,629 | $2,268,819 |
| Ages 65 and over | 3,885 | 560,398 | $2,178,744 |
| Ages 40 TO 49 | 2,377 | 328,342 | $1,364,683 |
| Ages 30 TO 39 | 1,936 | 303,987 | $1,189,272 |
| Ages 18 TO 23 | 1,713 | 249,596 | $945,340 |
| **Total** | **13,804** | **2,030,952** | **$7,946,859** |

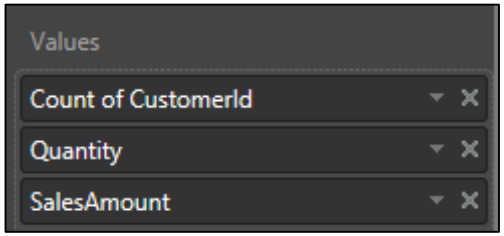
| State Name | 2012 | 2013 | 2014 | 2015 | Total |
|---|---|---|---|---|---|
| Florida | $51,731 | $300,867 | $535,694 | $891,345 | **$1,779,637** |
| North Carolina | $11,018 | $164,804 | $315,140 | $448,639 | **$939,601** |
| New York | $24,207 | $165,046 | $256,294 | $430,971 | **$876,519** |
| Georgia | $40,306 | $152,808 | $239,451 | $417,037 | **$849,602** |
| Virginia | $18,568 | $143,973 | $229,101 | $361,882 | **$753,524** |
| Pennsylvania | $16,390 | $136,447 | $229,420 | $292,563 | **$674,820** |
| Connecticut | $11,513 | $86,756 | $157,388 | $244,876 | **$500,533** |
| New Jersey | | $102,787 | $170,366 | $207,308 | **$480,462** |
| Massachusetts | | $72,459 | $158,284 | $243,864 | **$474,607** |
| South Carolina | $1,314 | $58,643 | $103,675 | $133,352 | **$296,984** |
| New Hampshire | | $23,522 | $60,122 | $80,496 | **$164,141** |
| Rhode Island | $4,990 | $31,587 | $59,179 | $60,675 | **$156,431** |
| **Total** | **$180,038** | **$1,439,699** | **$2,514,113** | **$3,813,009** | **$7,946,859** |

# Agenda

- ✓ Modeling Data in Power BI Desktop
- ✓ Understanding Table Relationships
- ✓ Writing DAX Expressions
- ✓ Creating Calculated Columns
- ➢ Creating Measures to Aggregate Data
- Adding Geographic Fields to a Data Model

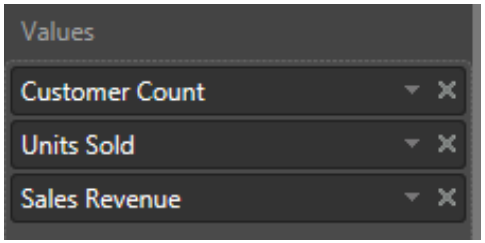
# Benefits of Measures over Calculated Columns

- Calculated columns can be aggregated in visual
  - However, aggregation details are stored in visual
  - Visual doesn't offer control over name and formatting




| Age Group | Count of CustomerId | Quantity | SalesAmount |
|---|---|---|---|
| Ages 50 TO 65 | 7974 | 1,143,644 | $4,904,342.59 |
| Ages 65 and over | 7569 | 1,050,879 | $4,480,748.98 |
| Ages 40 TO 49 | 5103 | 713,453 | $3,083,964.23 |
| Ages 30 TO 39 | 4547 | 646,121 | $2,798,053.19 |
| Ages 18 TO 23 | 3278 | 478,405 | $1,985,991.24 |
| **Total** | **28471** | **4,032,502** | **$17,253,100.23** |

- Measure defines name, aggregation and formatting
  - Work is done once and reused across many visuals
  - Makes data model more fool-proof for report designers

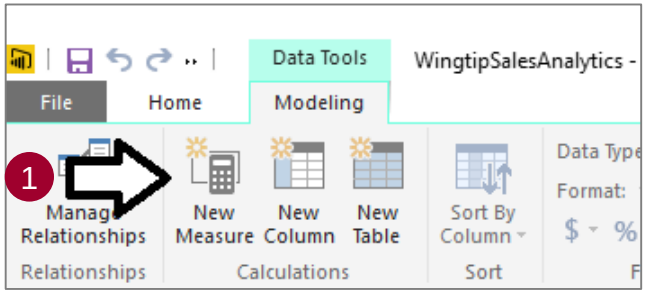


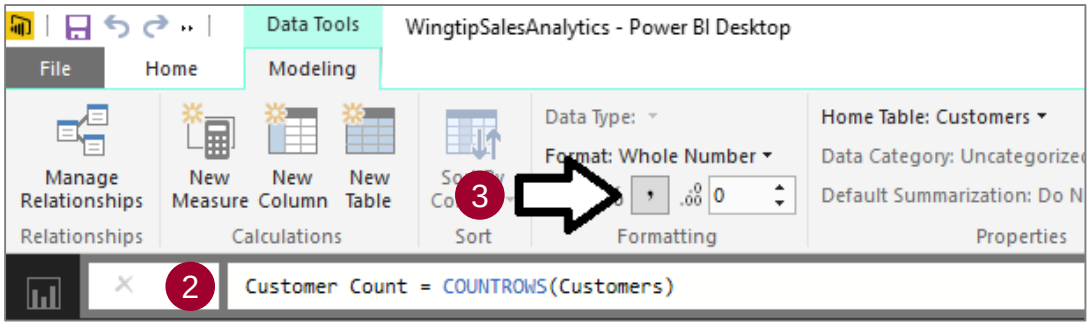

| Age Group | Customer Count | Units Sold | Sales Revenue |
|---|---|---|---|
| Ages 50 TO 65 | 7,974 | 1,143,644 | $4,904,343 |
| Ages 65 and over | 7,569 | 1,050,879 | $4,480,749 |
| Ages 40 TO 49 | 5,103 | 713,453 | $3,083,964 |
| Ages 30 TO 39 | 4,547 | 646,121 | $2,798,053 |
| Ages 18 TO 23 | 3,278 | 478,405 | $1,985,991 |
| **Total** | **28,471** | **4,032,502** | **$17,253,100** |

# Creating Measures

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

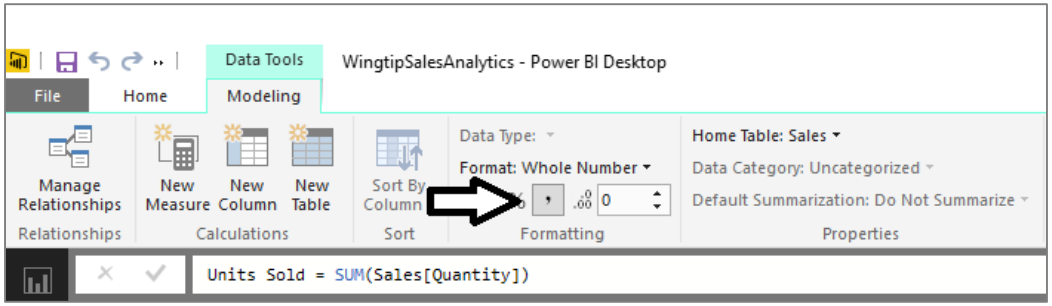
- Creating a measure with Power BI Desktop
  1. Click New Measure button
  2. Give measure a name and write DAX expressions
  3. Configure formatting



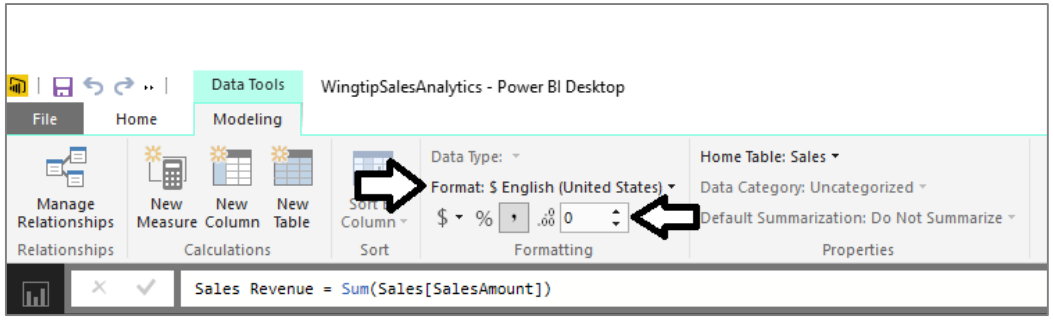



# Formatting Measures

- Format as whole number




- Format as currency



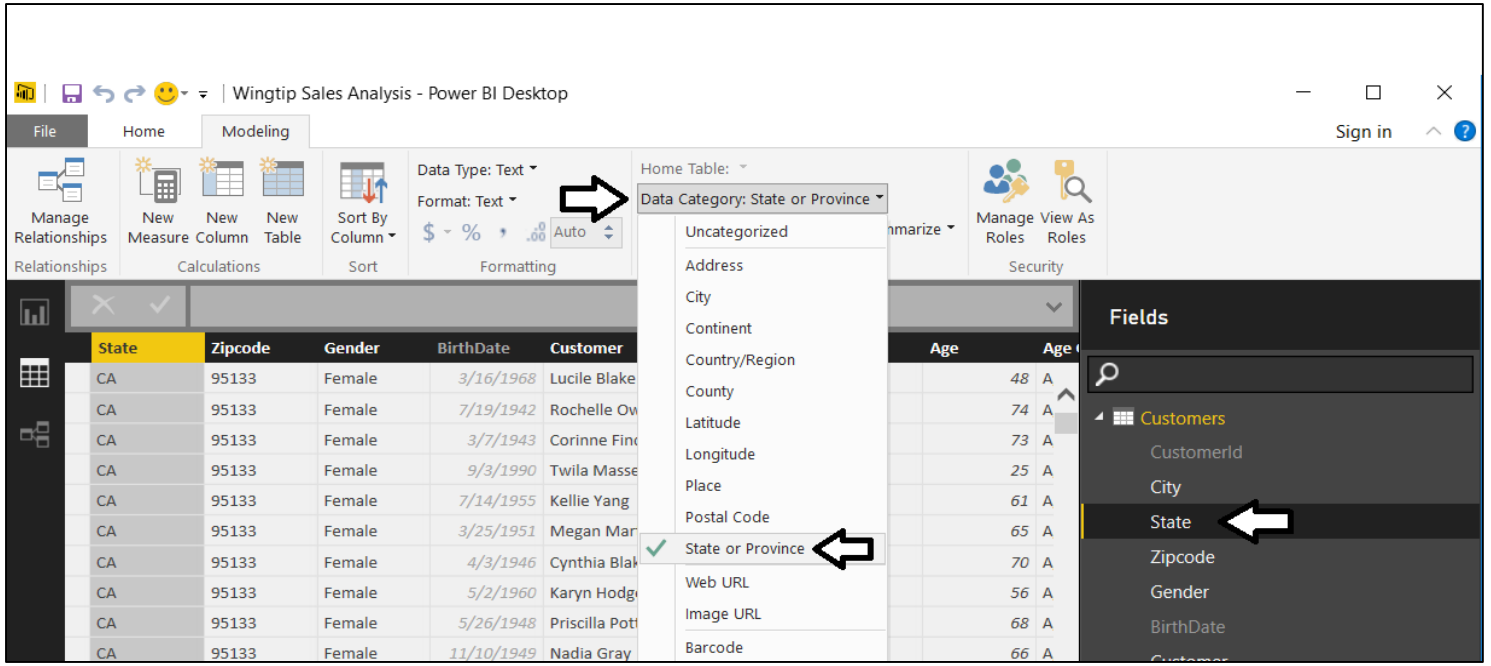

DEMO
Creating Measures

# Agenda

- ✓ Modeling Data in Power BI Desktop
- ✓ Understanding Table Relationships
- ✓ Writing DAX Expressions
- ✓ Creating Calculated Columns
- ✓ Creating Measures to Aggregate Data
- ➢ Adding Geographic Fields to a Data Model

# 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

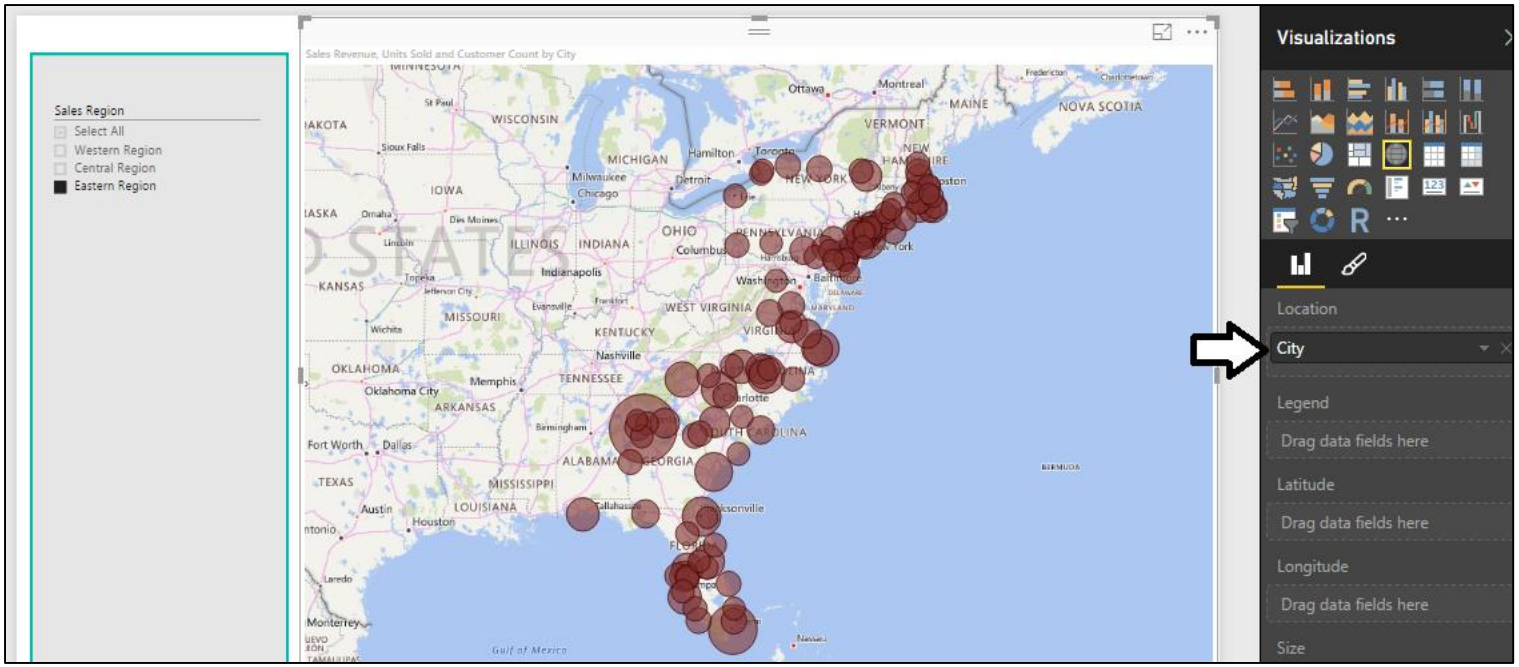
- City name alone is ambiguous
  - "Athens" defaults to Greece not Georgia
  - Concatenate city name with state to disambiguate

```
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 |

# Using Map Visual with a Geographic Field

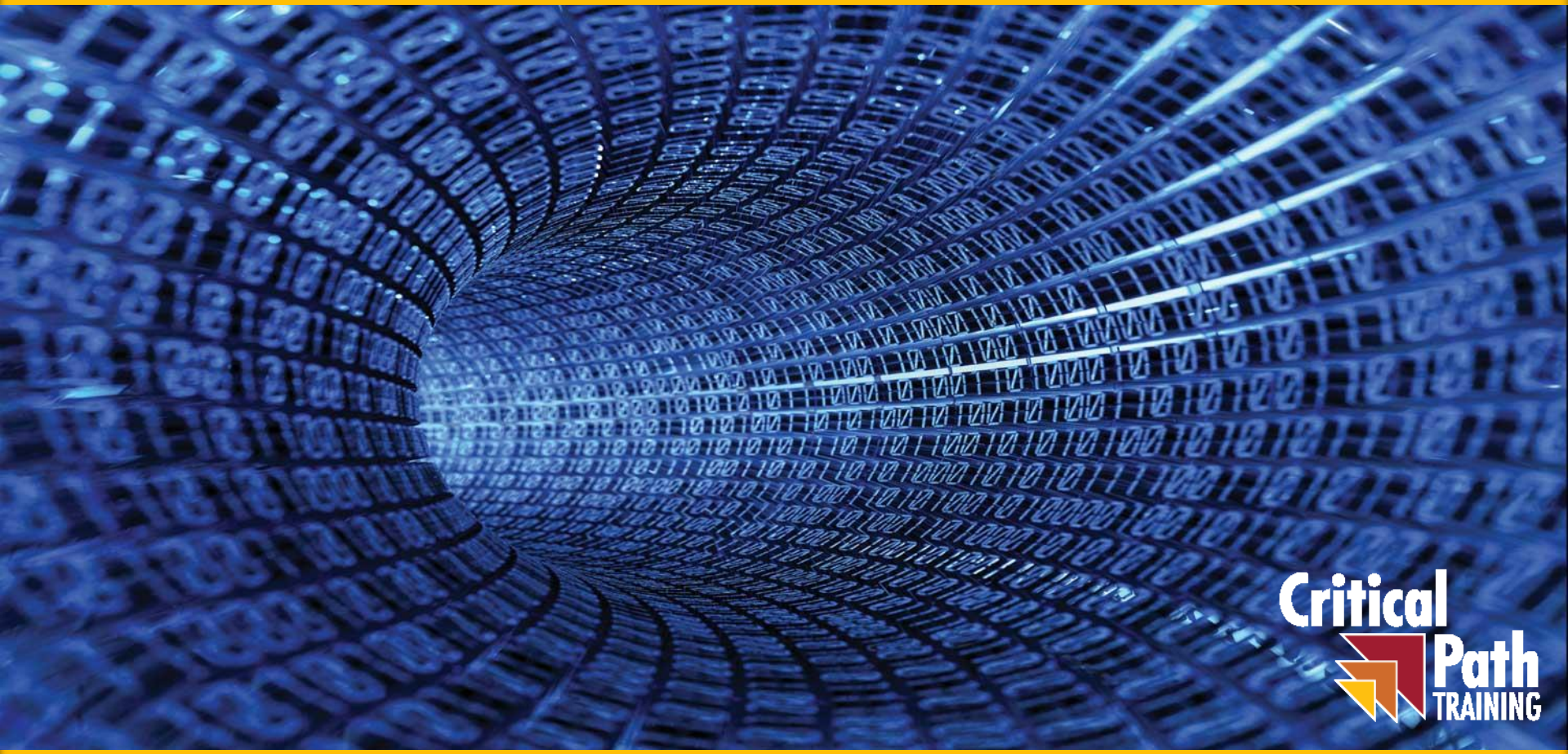
- Map Visual shows distribution over geographic area
  - Visual automatically updates when filtered



# Summary

- ✓ Modeling Data in Power BI Desktop
- ✓ Understanding Table Relationships
- ✓ Writing DAX Expressions
- ✓ Creating Calculated Columns
- ✓ Creating Measures to Aggregate Data
- ✓ Adding Geographic Fields to a Data Model

# Modeling Data with Hierarchies and Time Intelligence

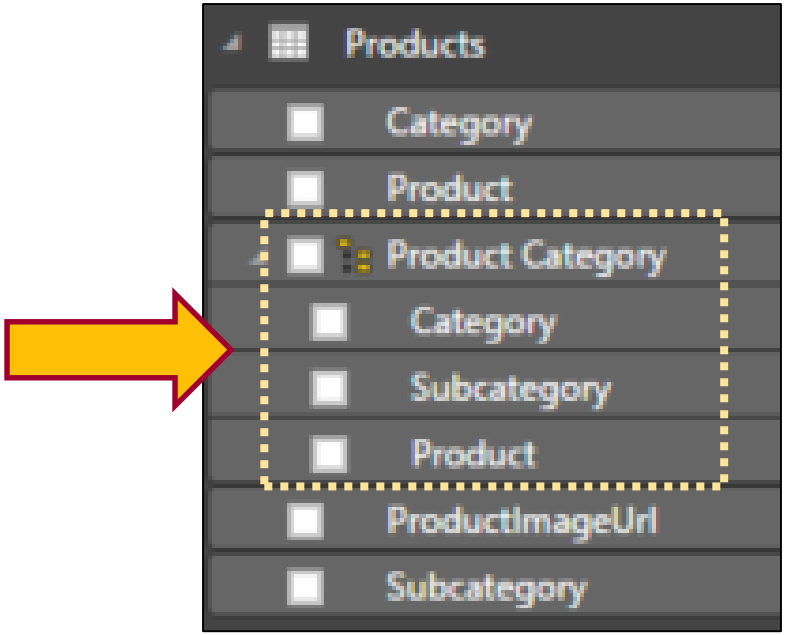


# Agenda

- 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



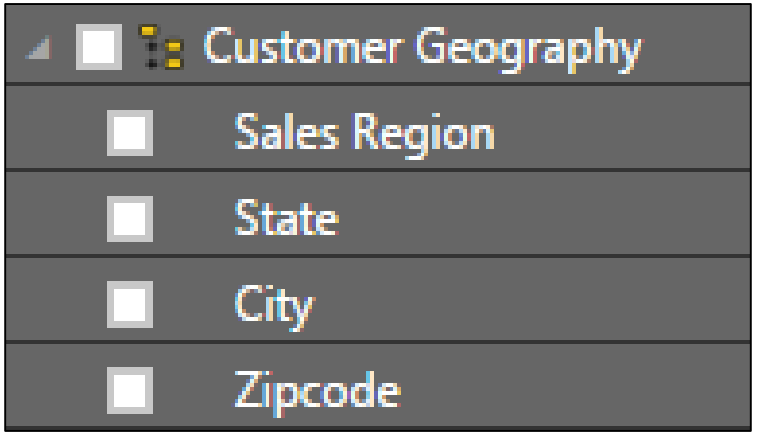
# Pulling Columns for Hierarchy into Single Table

- 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]
```

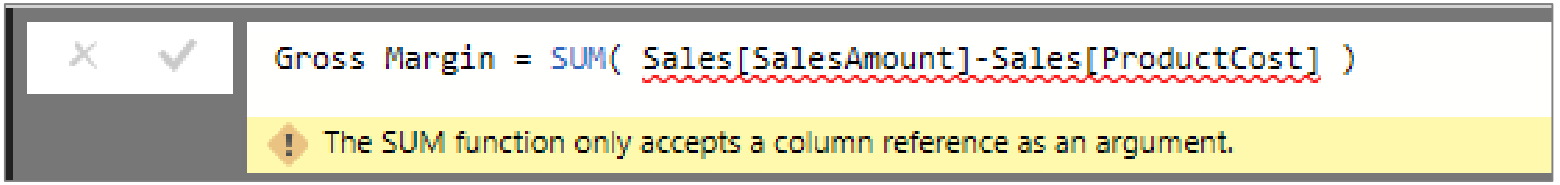
| | Age Group | Sales Region | State Name | SalesRegionSort | City |
|---|---|---|---|---|---|
| 48 | Ages 40 TO 49 | Western Region | California | 1 | San Jose, CA |
| 74 | Ages 65 and over | Western Region | California | 1 | San Jose, CA |
| 73 | Ages 65 and over | Western Region | California | 1 |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

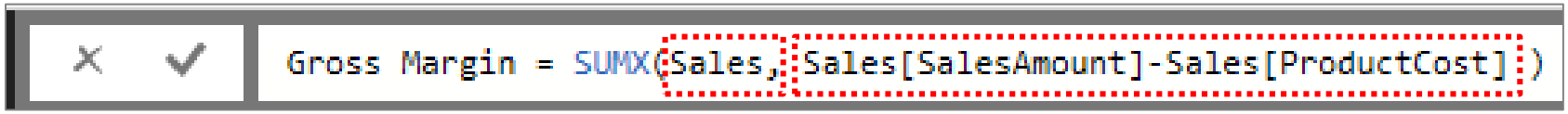
| Customer | Customer Type | Age | Age Group | Sales Region | State Name |
|---|---|---|---|---|---|
| Lucile Blake | One-time Customer | 48 | Ages 40 TO 49 | Western Region | California |
| Rochelle Owen | One-time Customer | 74 | Ages 65 and over | Western Region | California |
| Corinne Finch | One-time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

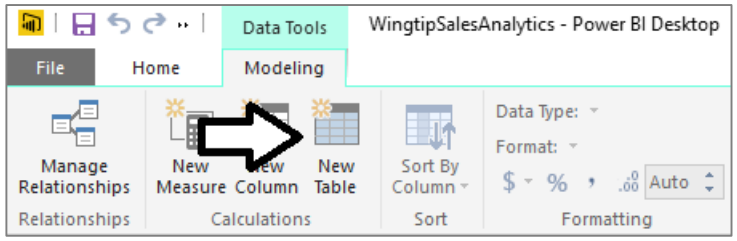
- ALL
- ALLEXCEPT
- CALCULATETABLE
- DISTINCT
- FILTER
- RELATEDTABLE
- VALUES

# Agenda

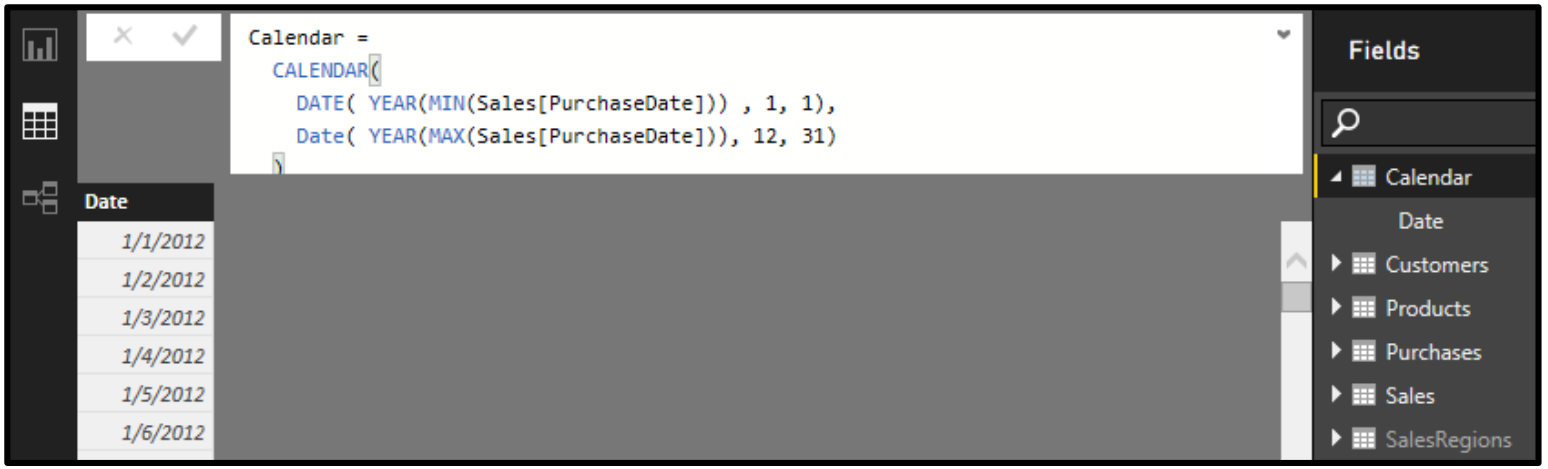
- ✓ Creating Dimensional Hierarchies
- ✓ Understanding the Evaluation Context
- ➢ Extending the Data Model using Calendar Tables
- Writing DAX Expressions with Time Intelligence
- Writing DAX Code with Contextual Awareness

# 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

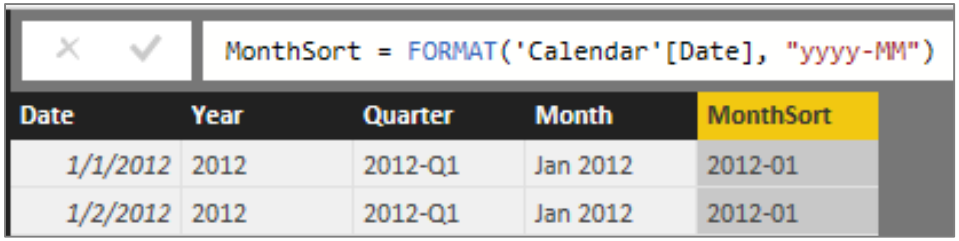
```
Month = FORMAT('Calendar'[Date], "MMM yyyy")
```

| 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 |

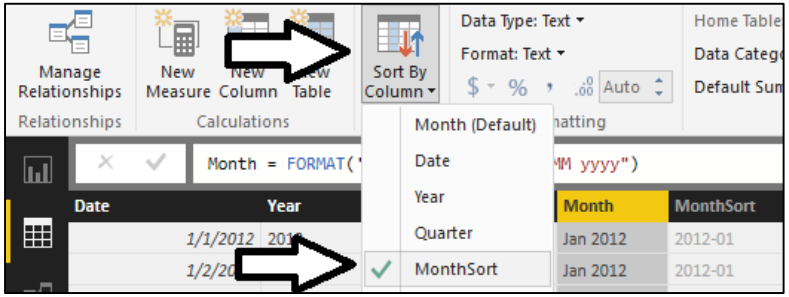
# Configuring Sort Columns

- 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 |

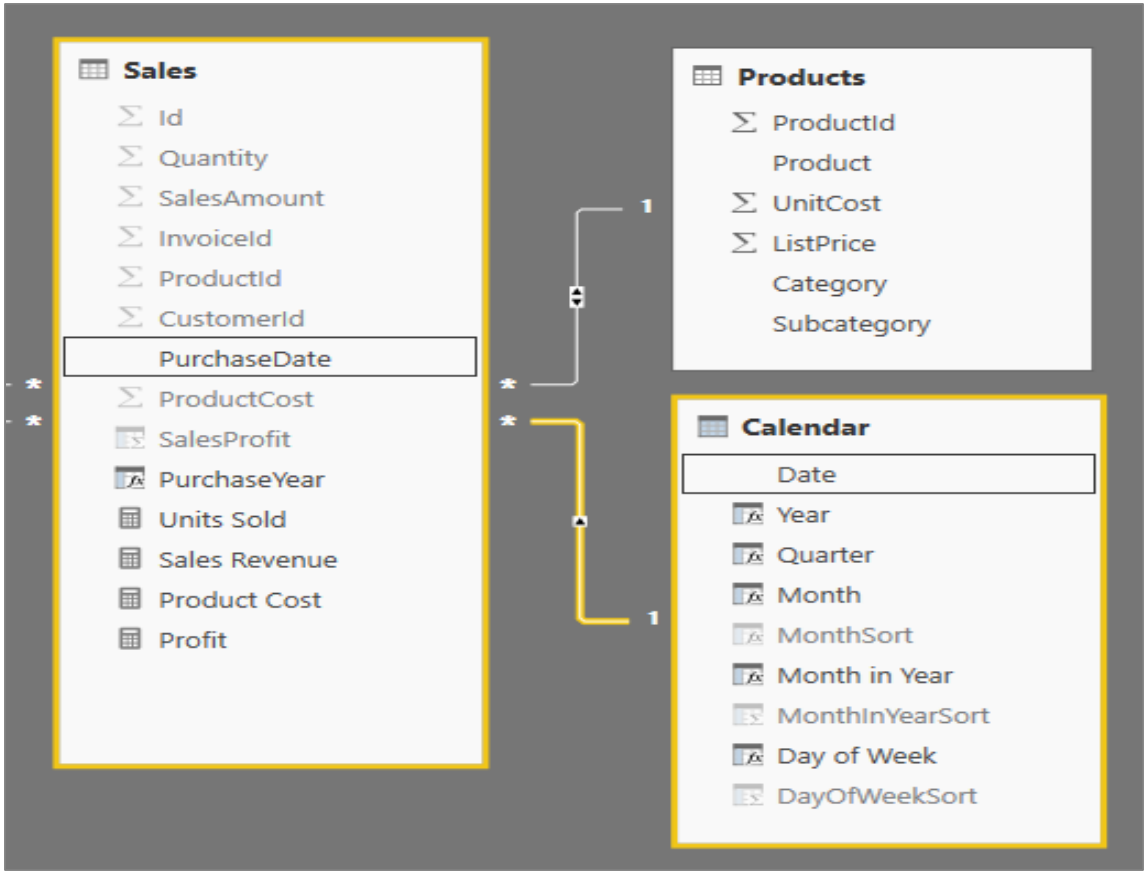
- Creating the **DayOfWeekSort** column

```
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

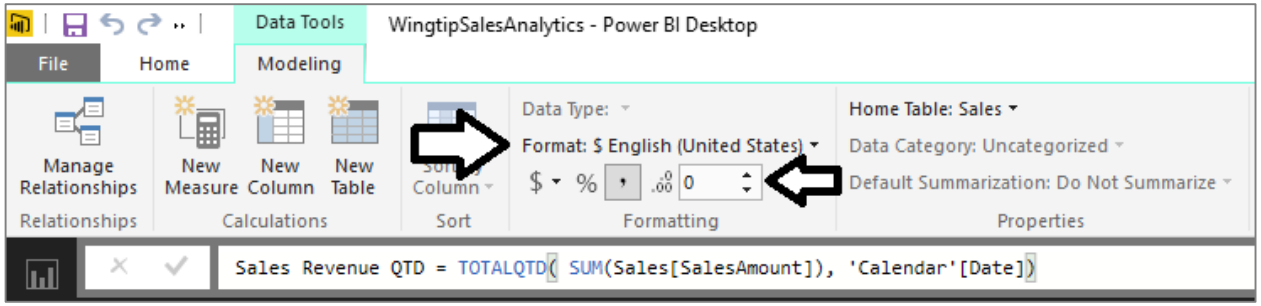
| Year | Quarter | Units Sold | Sales Revenue | Product Cost | Profit |
|---|---|---|---|---|---|
| 2012 | 2012-Q1 | 5,023 | $85,494 | $40,088 | $45,406 |
| | 2012-Q2 | 15,845 | $260,944 | $130,287 | $130,657 |
| | 2012-Q3 | 30,979 | $557,293 | $269,314 | $287,979 |
| | 2012-Q4 | 75,386 | $1,040,256 | $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

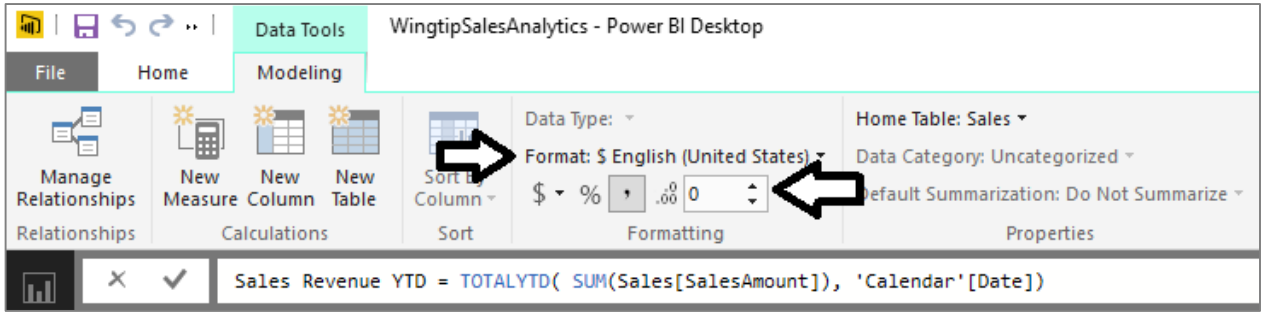
- ✓ Creating Dimensional Hierarchies
- ✓ Understanding the Evaluation Context
- ✓ Extending the Data Model using Calendar Tables
- ➢ Writing DAX Expressions with Time Intelligence
- • Writing DAX Code with Contextual Awareness

# Calculated Fields for QTD and YTD Sales

- TOTALQTD function calculates quarter-to-date totals

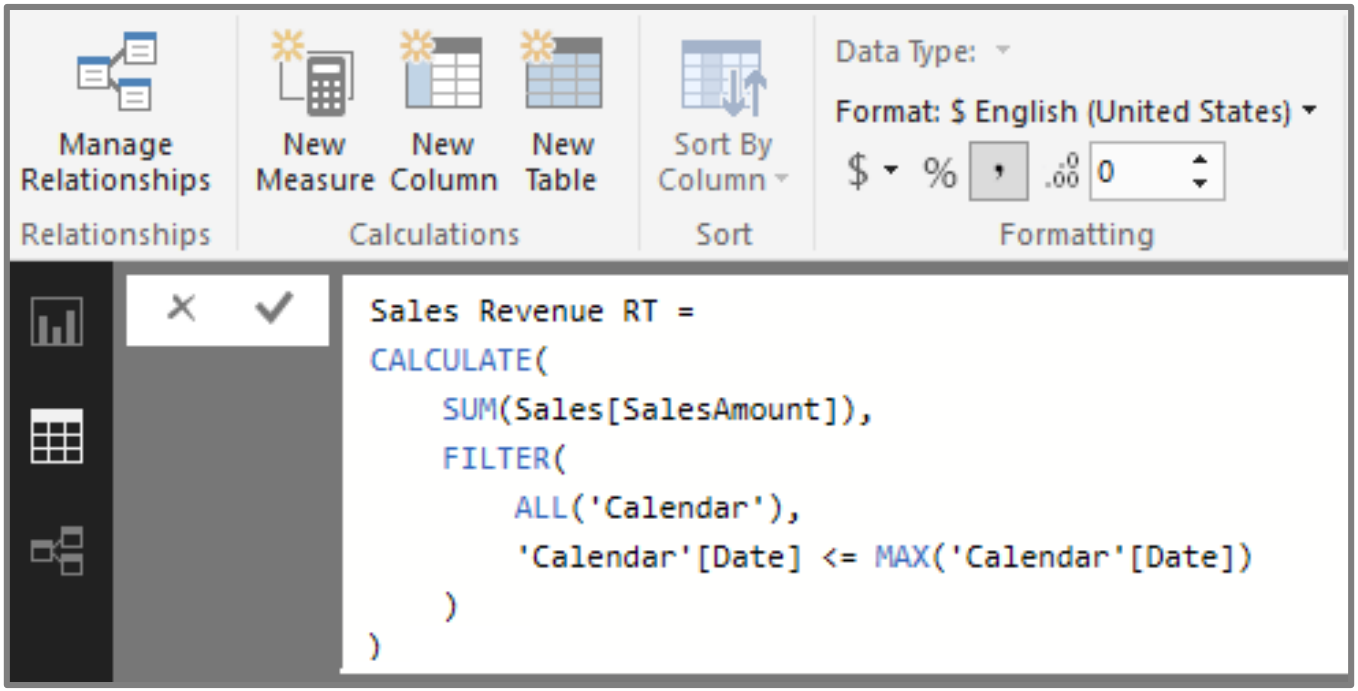



- TOTALYTD function calculates year-to-date totals



# 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

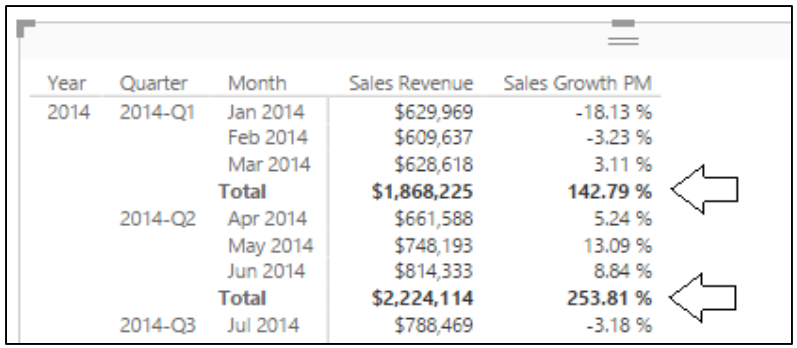
- ✓ Creating Dimensional Hierarchies
- ✓ Understanding the Evaluation Context
- ✓ Extending the Data Model using Calendar Tables
- ✓ Writing DAX Expressions with Time Intelligence
- ➢ Writing DAX Code with Contextual Awareness

# 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