Microsoft
Power Platform

Power BI
DAX in a Day

August 2021 BETA release

Brandon Campbell, MCIS, MCT, MCP
Sr. Customer Engineer
Microsoft

BETA version

Instructor introduction

- Brandon Campbell
- Senior Customer Engineer in Manufacturing OU at Microsoft
- Masters in Information Systems and Business Intelligence

Course aim

- Describe Power BI data models
- Introduce Data Analysis Expressions (DAX)
- Learn the calculation basics
- Define common business calculations

Course aim

Prerequisites

- This course has been designed specifically for experienced model developers
- It is recommended that you experience work with Power BI Desktop to create data models

Course modules

01: Introduce Data Models

02: Write DAX Formulas

03: Define Calculated Tables and Columns

04: Define Measures

05: Use DAX Iterator Functions

06: Modify Filter Context

07: Use DAX Time Intelligence Functions

Labs

Scenario

- The labs are based on the sales activities of the fictitious Adventure Works company
- The Adventure Works company:
  - Represents a bicycle manufacturer that sells bicycles and accessories to global markets
  - Accumulates operational data in an Azure SQL Database
  - Needs to explore and discover deeper insight from their data
- In the labs, as their data modeler author, you will enhance an existing model with calculations

Agenda (times are approximate and will be fluid with the class)

Morning	
09:00 AM – 09:15 AM	Introduction
09:15 AM – 09:45 AM	Module 01 Introduction to Data Models
09:45 AM –10:30 AM	Module 02 Write DAX formulas
10:30 AM – 10:50 AM	Lab 02
10:50 AM – 11:00 AM	**Break**
11:00 AM – 11:20 AM	Module 03 Define Calculated Tables and Columns
11:20 AM – 11:30 AM	Lab 03
11:30 AM – 12:00 PM	Module 04 Define Measures
12:00- 12:20	Lab 04
12:20 PM – 01:00 PM	**Break for lunch**
Afternoon	
01:00 PM – 01:30 PM	Module 05
01:30 PM – 01:50 PM	Lab 05
01:50 PM – 02:30 PM	Module 06
02:30 PM – 02:50 PM	Lab 06
02:50 PM – 03:00 PM	**Break**
03:00 PM – 03:30 PM	Module 07
03:30 PM – 03:50 PM	Lab 07
03:50 PM – 04:15 PM	Summary Questions and survey

Labs

2: Write DAX formulas for Power BI

3: Add Calculated table and Columns

4: Add measures to Power BI Desktop models

5: Use DAX iterator functions in Power BI Desktop models

6: Modify DAX filter context in Power BI Desktop Models

7: Use DAX Time Intelligence functions in Power BI Desktop models

Questions?

Microsoft
Power Platform

Power BI
DAX in a Day

Module 01

Introduce Data Models

BETA version

Module outline

01: Introduce Data Models

- Data model fundamentals
- Star schema design
- Define analytic query
- Configure report visuals

Data model fundamentals

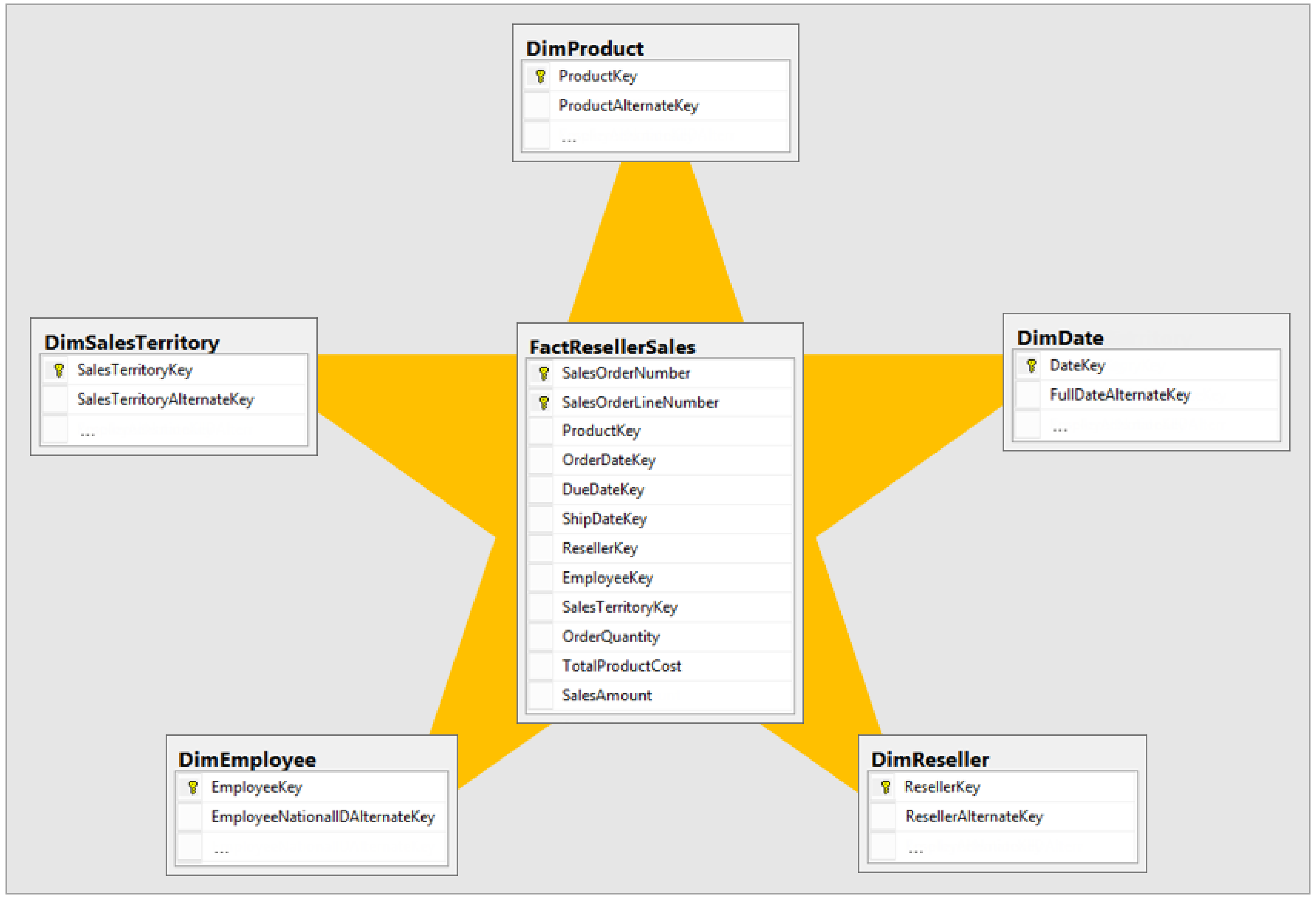
- A **Data model** is a query-able data resource optimized for analytics
  - Microsoft data models can be queried using DAX or MDX
- It can also be described as a *semantic model*, or simply a *model*

Data model fundamentals

Tabular model

- A **Tabular model** comprises one or more tables
- It possibly defines relationships, hierarchies, hierarchy levels, and calculations.
- All Power BI models are tabular

Star schema design



- Fact tables
- Dimension tables

For a detailed explanation of star schema, read the Understand star schema and the importance for Power BI article

BETA version

Star schema design

Fact tables

- A **Fact table** stores an accumulation of rows
- It records a specific business activity, for example:
  - Sales transactions
  - Stock movements
  - Daily currency exchange rates

Star schema design

Dimension tables

- A **Dimension table** describes a business entity
- It is designed to filter and group related fact table data, for example:
  - Date
  - Product
  - Geography
  - People (employees or customers)
- Each dimension table must have a unique column (key) so a relationship can be created to the fact tables

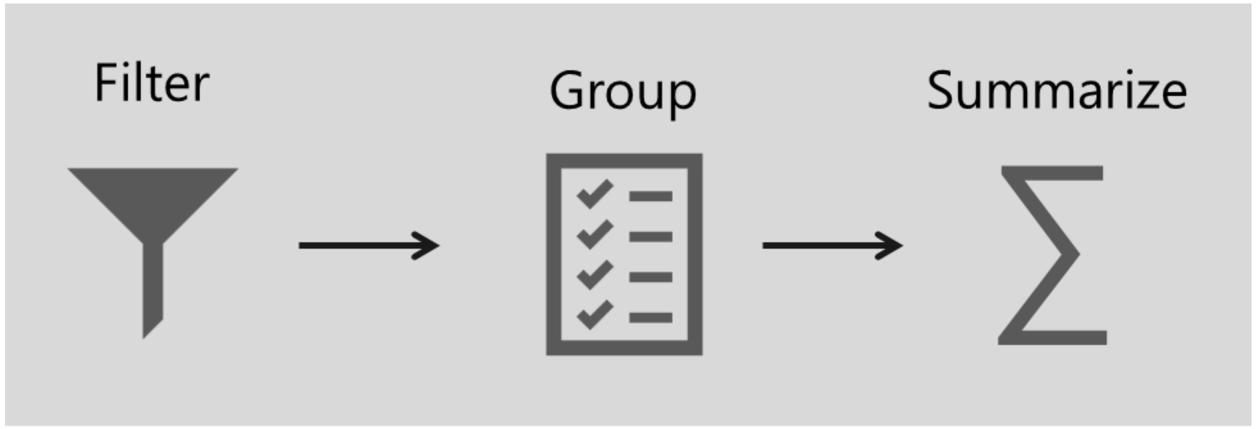
Star schema design

Compare table types

	Dimension table	Fact table
Model purpose	Stores business entities	Stores events or observations
Table structure	Includes a key column and descriptive columns for filtering and grouping	Includes dimension key columns and numeric measure columns that can be summarized
Data volume	Typically, contains fewer rows (relative to fact tables)	Can contain numerous rows
Query purpose	To filter and group	To summarize

Define analytic query

- An **Analytic query** retrieves data from a data model
- It has three distinct phases:
  1. Filter
  2. Group
  3. Summarize



Define analytic query

1: Filter phase

- The **Filter phase** targets data of relevance
- It can be applied to:
  - Report
  - Page
  - Visual
- Also applied using row-level security (RLS)
- Visuals can inherit filters, or they can be applied directly

Define analytic query

2: Group phase

- The **Group phase** divides query results into groups
- It often groups by entity (column) in dimension table
- Common examples of grouping:
  - Year, Month, or Day
  - Product category
  - Country or region

Define analytic query

3: Summarize

- The **Summarize phase** produces a single value
- Typically, numeric columns are summarized
- Common examples of summarizing:
  - Sum
  - Count
  - Average
  - Min and Max

Configure report visuals

- Select a visualization type
- Map dataset fields
- Configure mapped fields (rename etc.)
- Apply format options

Configure report visuals

Fields

- **Fields** is a collective term for columns, hierarchy levels, and measures
- They can be used in the following phases of an analytic query

Field type	Filter	Group	Summarize
Column	✓	✓	✓
Hierarchy level	✓	✓	
Measure	✓		✓

Configure report visuals

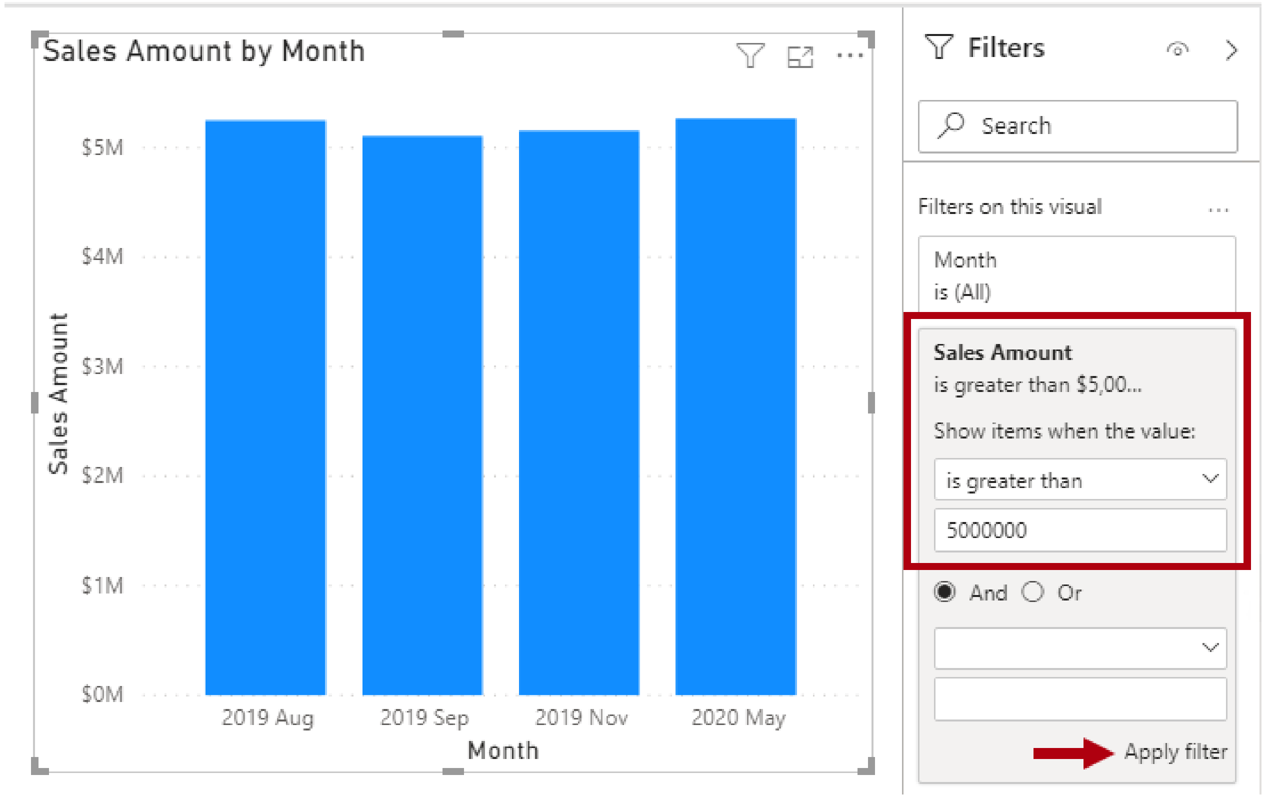
Hierarchy levels

- Hierarchy levels are based on columns
- They can be used to filter and group—but not to summarize

Configure report visuals

Measures

- Measures are designed to summarize
- They can be used to filter
  - However, they filter groups by the measure—for example, show only months where the sales amount (summarization) is greater than $5M
- They cannot be used to group data



Filters on this visual

Month
is (All)

Sales Amount
is greater than $5,00...

Show items when the value:
is greater than
5000000

● And ○ Or

BETA version

Questions?

Microsoft
Power Platform

Power BI
DAX in a Day

Module 02

Write DAX Formulas

BETA version

Module outline

02: Write DAX Formulas

- Describe DAX calculation types
- Write DAX formulas
- Describe DAX data types
- Work with DAX functions
- Use DAX operators

Describe DAX calculation types

- DAX can be used to create three types of calculations:
  - Calculated table
  - Calculated column
  - Measure
- DAX can also be used to define row-level security (RLS) rules

RLS is not in scope
for this course

Describe DAX calculation types

Calculated table

- A **Calculated table** adds a new table to the model
- Examples:
  - Date tables
  - Role-playing dimensions
  - What-if analysis
- Each calculated table will increase the storage size of the model

Describe DAX calculation types

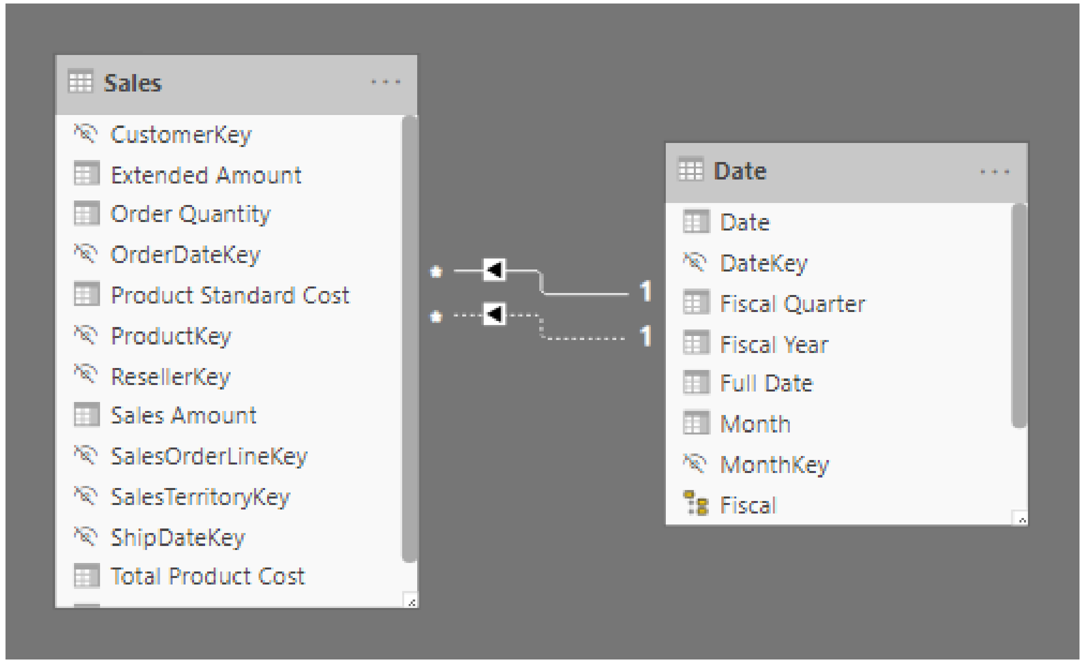
Calculated table » Date table

- A **Date table** is required to work with Time Intelligence functions
- It can be created using the CALENDAR or CALENDARAUTO functions

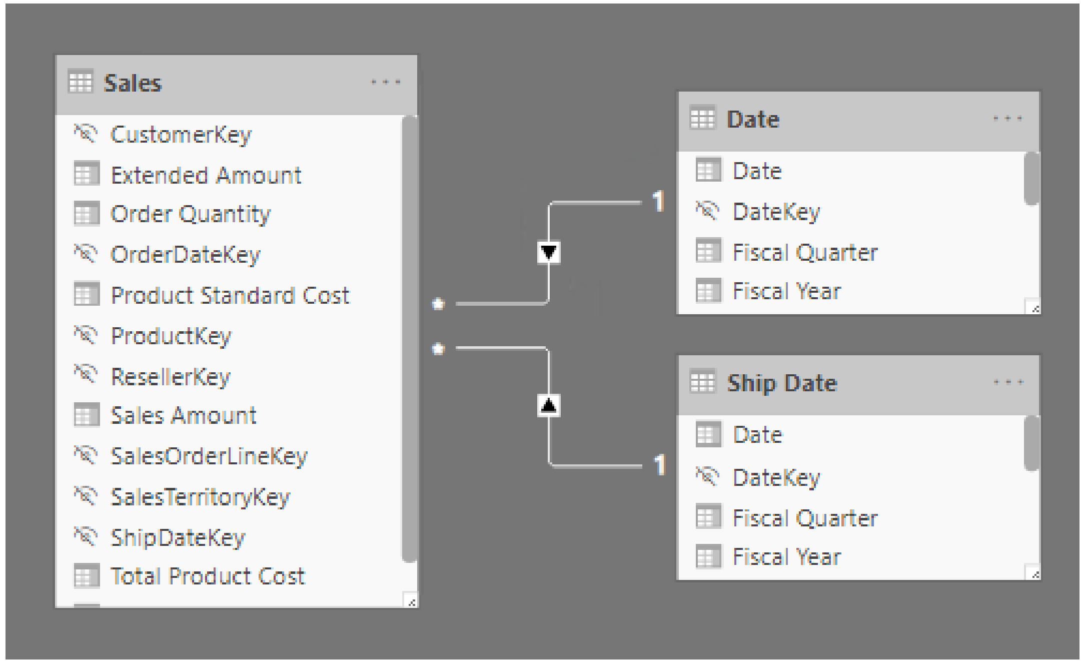
Describe DAX calculation types

Calculated table » Role-playing dimensions

Inactive relationships



Duplicate tables with active relationships



BETA version

Describe DAX calculation types

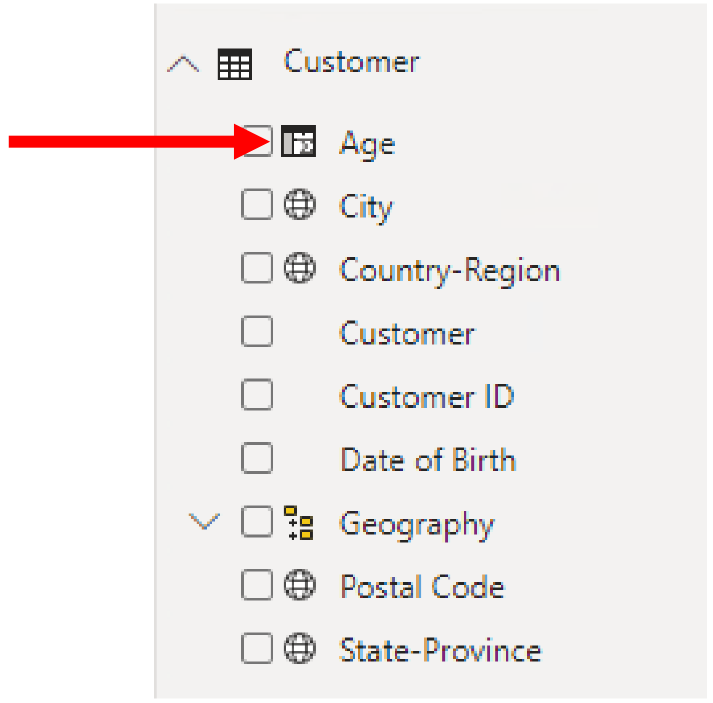
Calculated table » What-if analysis

- Power BI Desktop supports creating **What-if Parameters**
- It automates the creation of a calculated table with a generated series of rows, and a measure to retrieve the currently selected value
- Commonly used in what-if analysis
- Typically, these tables remain disconnected (not related to other tables)
- Examples: Currency conversion, forecasting scenarios

Describe DAX calculation types

Calculated column

- A **Calculated column** adds a new column to a table
- A formula is evaluated for each row in table
- The formula must return a single value
- It is only evaluated when model is refreshed*
- It increase storage size of the model*

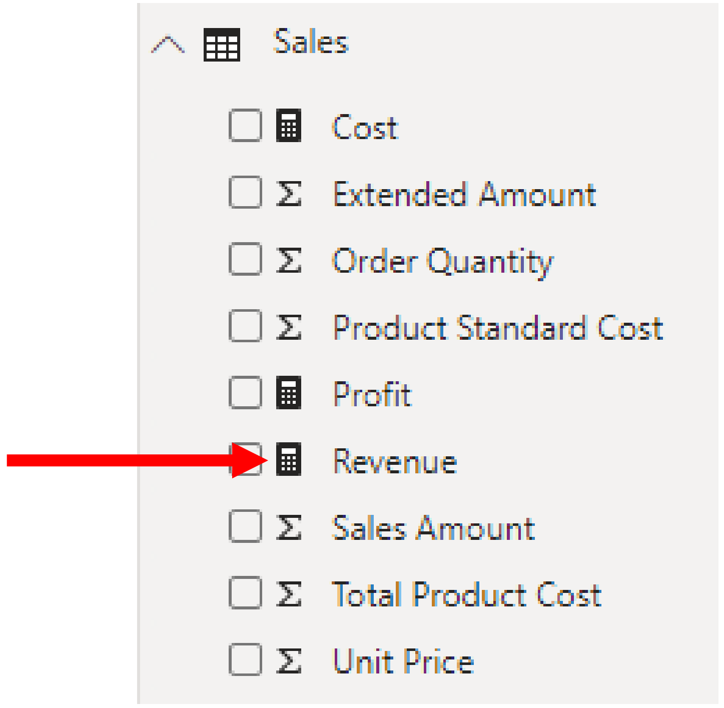


*Except when added to a DirectQuery table

Describe DAX calculation types

Measures

- A **Measure** summarizes model data
- The formula must return a single value
- It is evaluated at query-time
- Results are never stored in the model



Write DAX formulas

- Each model calculation type is defined by its name, followed by the equals symbol (=), which is then followed by a DAX formula

```
<calculation name> = <DAX formula>
```

```
Ship Date = 'Date'
```

Write DAX formulas

- A **DAX formula** consists of expressions that return a result
- The result is either a table object or a scalar value
- Topics:
  - DAX functions
  - DAX operators
  - References to model objects
  - Constant values
  - DAX variables
  - Whitespace

Write DAX formulas

DAX functions

- DAX is a functional language
- Functions can use parameters
- Functions can be nested
- Most functions return an object (scalar value, column, or table)

Many common functions will be introduced in this course

Write DAX formulas

DAX operators

- Formulas also rely on **DAX operators**, which can perform arithmetic calculations, compare values, work with strings, or test conditions
- Operators can:
  - Perform arithmetic calculations
  - Compare values
  - Work with strings
  - Test conditions

DAX operators are described later in this module

Write DAX formulas

Reference model objects

- There are only three types of model objects that are referenced by DAX formulas:
  - Tables
  - Columns
  - Measures
- It is not possible to reference a hierarchy level
  - Instead, use the column that the level is based on

Write DAX formulas

Reference model objects » Table reference

- Table references use the table name
- If the table is a reserved word, it must be enclosed in single quotes

```
Arrival Airport = Airport
```

```
Ship Date = 'Date'
```

Write DAX formulas

Reference model objects » Column reference

- Column references always enclose the column name in square brackets
- To improve readability and to disambiguate, precede the column name with its table name

```
Revenue = SUM([Sales Amount])
```

```
Revenue = SUM(Sales[Sales Amount])
```

Write DAX formulas

Reference model objects » Measure reference

- Measure references always enclose the measure name in square brackets

```
Profit = [Revenue] - [Cost]
```

Write DAX formulas

DAX variables

- Variables can be defined in an expression to make writing DAX easier
- Main benefits:
  - Improves readability of a formula
  - Improves performance when an expression is used multiple times
  - Allows testing portions of a complex formula, by returning only a variable for review

```
VAR <name> = <expression> [VAR <name2>…]
RETURN <expression>
```

Write DAX formulas

DAX variables » Example

```
Revenue YoY % =
VAR RevenuePriorYear =
      CALCULATE(
            [Revenue],
            SAMEPERIODLASTYEAR('Date'[Date])
      )
RETURN
      DIVIDE(
            [Revenue] - RevenuePriorYear,
            RevenuePriorYear
      )
```

BETA version

Write DAX formulas

Whitespace

- Whitespace refers to characters that format formulas in a way that is quick and simple to understand
- Whitespace characters include:
  - Spaces
  - Tabs
  - Carriage returns

Write DAX formulas

Whitespace » Usage

- Use spaces between operators
- Use tabs to indent nested function calls
- Use carriage returns to separate function arguments
- Err on side of more whitespace

Write DAX formulas

Whitespace » Example

```
Revenue YoY % = DIVIDE([Revenue] - CALCULATE([Revenue],
SAMEPERIODLASTYEAR ('Date'[Date])), CALCULATE ([Revenue],
SAMEPERIODLASTYEAR ('Date'[Date])))
```

```
Revenue YoY % =
DIVIDE(
        [Revenue]
                - CALCULATE(
                        [Revenue],
                        SAMEPERIODLASTYEAR ('Date'[Date])
        ),
        CALCULATE (
                [Revenue],
                SAMEPERIODLASTYEAR ('Date'[Date])
        )
)
```

Describe DAX data types

- Columns have a set data type
- Column types are set in Power Query or for calculated columns, they are inferred from the DAX formula
- Measure data types are inferred from the DAX formula

Describe DAX data types

Model data type	DAX data type	Description
Whole number	64-bit integer	-2^{63} through $2^{63}-1$
Decimal number	64-bit real	8 bytes per value (64 bits!!) – Like (SQL) Float
Boolean	Boolean	TRUE or FALSE
Text	String	Unicode character string
Date	Date/time	
Currency	Currency	-9.22×10^{14} to 9.22×10^{14} Limited to four decimal digits of fixed precision
N/A	BLANK	Sometimes equivalent to SQL

BETA version

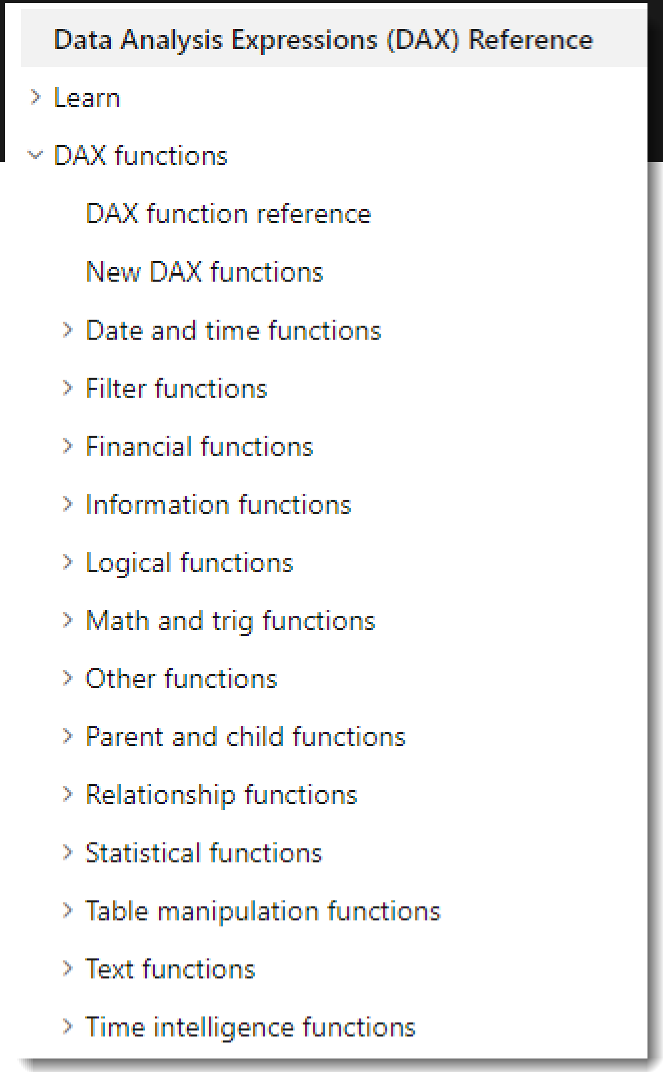
Describe DAX data types

BLANK

- DAX uses **BLANK** for both database NULL and for blank cells in Excel
- BLANK doesn't mean zero
  - Perhaps it is simpler to think of it as the "absence of a value“
- Two DAX functions are related to the BLANK data type:
  - **BLANK** - Returns BLANK
  - **ISBLANK** - Tests whether an expression evaluates as BLANK

Work with DAX functions

- The DAX function library consists of hundreds of functions, each designed to accomplish a specific goal
- Many are similar to Excel functions
- Some are specific to data modeling:
  - Relationship navigation
  - Filter context modification
  - Iterator
  - Time Intelligence
  - Path



Data Analysis Expressions (DAX) Reference

- Learn
- DAX functions
  - DAX function reference
  - New DAX functions
  - Date and time functions
  - Filter functions
  - Financial functions
  - Information functions
  - Logical functions
  - Math and trig functions
  - Other functions
  - Parent and child functions
  - Relationship functions
  - Statistical functions
  - Table manipulation functions
  - Text functions
  - Time intelligence functions

Work with DAX functions

Excel-like functions

- Summarization functions
  - SUM, COUNT, AVERAGE, MIN, MAX ….
- Mathematic functions
  - ABS, ROUND, SQRT, LEN, LEFT, RIGHT, UPPER, DATE, YEAR, MONTH, NOW, ISNUMBER, TRUE, FALSE, AND, OR, NOT, ISERROR
- Conditional logic

```
IF ( <logical_test>, <value_if_true> [, <value_if_false> ] )
```

Work with DAX functions

Not found in Excel

- DISTINCTCOUNT
- DIVIDE

```
DIVIDE ( <numerator>, <denominator> [, <alternate_result>] )
```

Use DAX operators

Arithmetic operators

Operator	Description	Example
+	Addition	2 + 3 = 5
-	Subtraction	5 - 3 = 2
*	Multiplication	2 * 3 = 6
/	Division	6 / 3 = 2
^	Exponentiation	2 ^ 3 = 8

Use DAX operators

Comparison operators

Operator	Description	Example	Result
=	Equal To	2 = 3	FALSE
==	Strict equal to	[Revenue] == 0	*
>	Greater than	2 > 3	FALSE
<	Less than	2 < 3	TRUE
>=	Greater than or equal to	2 >= 3	FALSE
<=	Less than or equal to	2 <= 3	TRUE
<>	Not equal to	2 <> 3	TRUE

Use DAX operators

Text concatenation

- Use the ampersand (&) character to concatenate text values
- DAX functions **CONCATENATEX** and **COMBINEVALUES** are also options for text concatenation

```
Model Color = Product[Model] & "-" & Product[Color]
```

Use DAX operators

Logical operators

Operator	Description	Example	Result
&&	Logical AND	TRUE && FALSE	FALSE
||	Logical OR	TRUE || FALSE	TRUE
IN	Multiline OR	“A” IN {“A”, “B” “C”}	TRUE
NOT	Inverts state	NOT FALSE	TRUE

```
Customer[Country-Region]
     IN {
          "Australia",
          "New Zealand"
     }
```

Use DAX operators

Operator precedence

Operator	Description
^	Exponentiation
-	Sign (as in -1)
* and /	Multiplication and Division
NOT	NOT
+ and -	Addition and subtraction
&	Text concatenation
=, ==, <, >, <=, >=, <>	Comparison

Use DAX operators

Operator precedence » Examples

Operation	Result
2 + 3 * 4	14
(2 + 3) * 4	20

Use DAX operators

Implicit conversion

- Usually, DAX automatically identifies the data types of referenced model objects and performs implicit conversions where necessary to complete the specified operation

Operator	Result	Conversion
TRUE && 1	TRUE	1 to TRUE
TRUE && 0	FALSE	0 to FALSE
“A” & 1	“A1”	1 to text
DATE(2020,1,1) + “2”	3 Jan 2020	“2” to integer
“6” / “2”	3	“6” and “2” to integer

Lab 2

Write DAX formulas for Power BI

BETA version

Microsoft
Power Platform

Power BI
DAX in a Day

Knowledge Check 01

https://aka.ms/DAXIADKnowCheck02

BETA version

Resources

Learn: Write DAX formulas for Power BI Desktop models
https://docs.microsoft.com/learn/modules/dax-power-bi-write-formulas/

Docs: DAX overview
https://docs.microsoft.com/dax/dax-overview

Docs: Column and measure references
https://docs.microsoft.com/power-bi/guidance/dax-column-measure-references

Docs: DIVIDE function vs divide operator (/)
https://docs.microsoft.com/power-bi/guidance/dax-divide-function-operator

Docs: Use variables to improve your formulas
https://docs.microsoft.com/power-bi/guidance/dax-variables

Questions?

Microsoft
Power Platform

Power BI
DAX in a Day

Module 03

Define Calculated Tables and Columns

BETA version

Module outline

03: Define Calculated Tables and Columns

- Create calculated tables
- Create calculated columns
- Describe row context
- Choose technique to add a column

Create calculated tables

- A **Calculated table** is created using a DAX formula
- The formula can duplicate or transform existing table
- It cannot connect to external data
- The formula must return a table object
- It will increase the storage size of the model
- It can increase the data refresh time

Create calculated columns

- A **Calculated column** is added to a table using a DAX formula
- It can be added to any table in your model
- The formula must return a scalar—or single—value
- It will increase the storage size of the model
- It can increase the data refresh time

Describe row context

- The calculated column formula is evaluated for each table row
- Evaluation is within *row context*, which means current row

```
Due Fiscal Year =
"FY"
      & YEAR('Due Date'[Due Date])
      + IF(
            MONTH('Due Date'[Due Date]) <= 6,
            1
      )
```

Choose technique to add a column

- There are three techniques to add a column to a table:
  - Add a column in data source
  - Add a computed column (using M) to a query in Power Query
  - Add a calculated column (using DAX) to a model table
- Regardless of the technique used, it results in the same outcome
  - Report designers or users cannot determine the origin of a column

Lab 3

Add Calculated table and Columns

BETA version

Questions?

Microsoft
Power Platform

Power BI
DAX in a Day

Module 04

Define Measures

BETA version

Module outline

04: Define Measures

- Describe measure types
- Create Quick Measures
- Describe differences between calculated column and measure

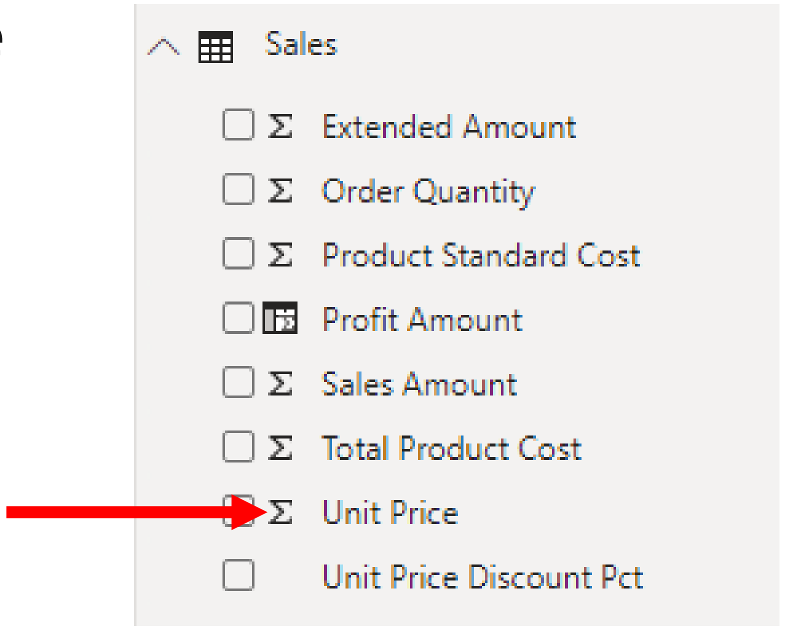
Describe measure types

- Measures produce summarizations of model data
- There are two types of measures:
  - Implicit measure
  - Explicit measure

Describe measure types

Implicit measure

- An **Implicit measure** is a model column that is summarized
- Implicit measures:
  - Appear with the sigma symbol (∑) in the **Fields** pane
  - Are always numeric columns
  - Will summarize values when added to a visual



Describe measure types

Implicit measure (continued)

- Benefits:
  - Simple to use
  - Easy to learn
  - Less work as a data modeler
- Limitations:
  - Easy to choose wrong default
  - Limited scenarios
  - Not visible to MDX

Describe measure types

Explicit measure

- An **Explicit measure** is a measure defined using DAX
- Explicit measures:
  - Define a formula that returns a scalar—or single—value
  - Do not store values in the model
  - Cannot reference tables or columns directly

Describe measure types

Explicit measure » Compound measure

- A **Compound measure** is an explicit measure that refers to other explicit measures

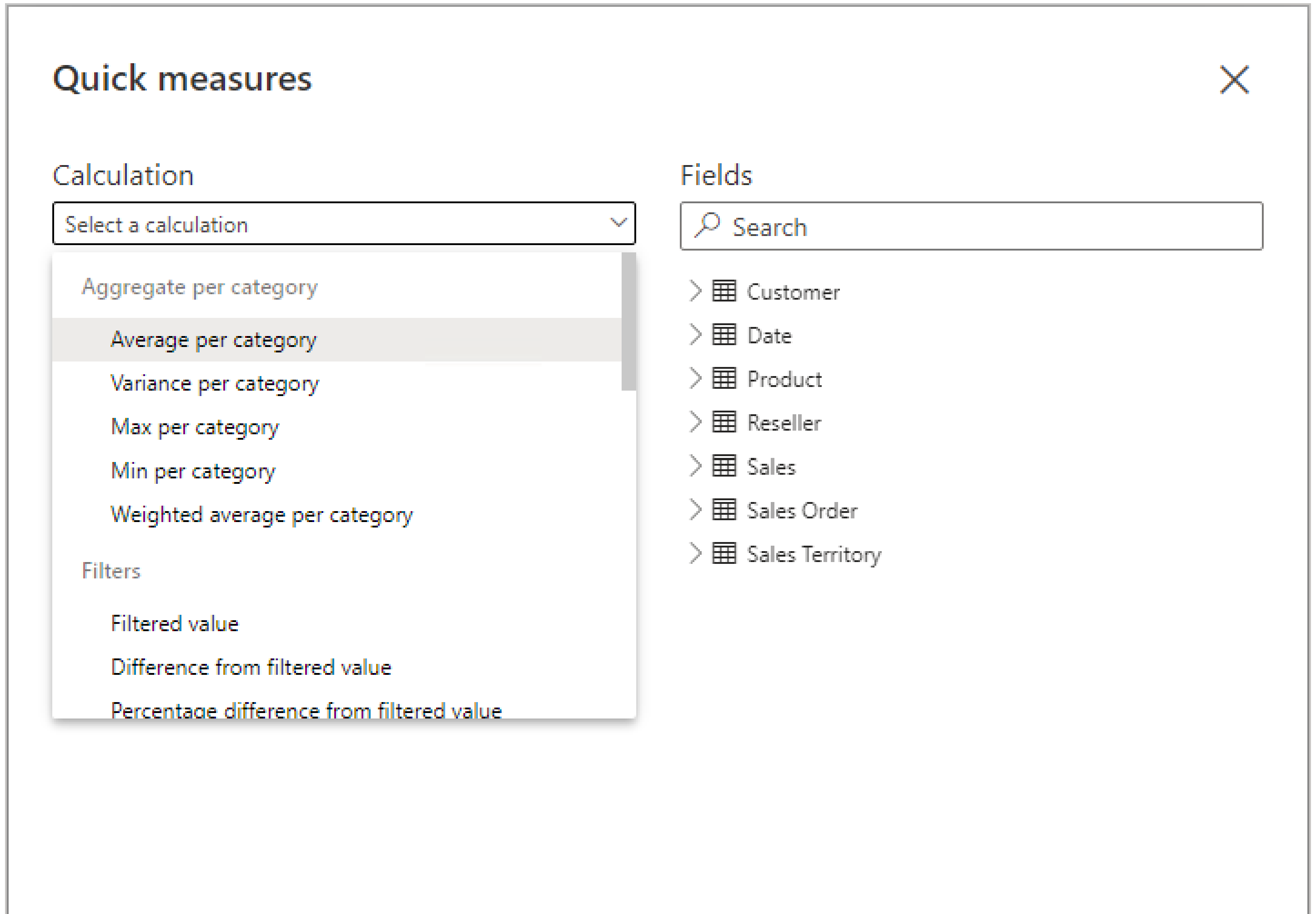
```
Profit = [Revenue] - [Cost]
```

Quick Measures

- Power BI Desktop supports creating **Quick Measures**
- Benefits:
  - Quick and easy
  - Common calculation templates are available
  - Automatically creates the DAX expression
  - No need to understand any DAX
- Once created, they are like any other explicit measure
  - To modify the measure, simply edit the formula

Quick Measures

Example



Calculated columns and measures

- There are differences between calculated column and measure

	Columns	**Measures**
Purpose	Extend table	Summarize model data
Evaluation	Row context at data refresh-time	Filter context at query-time
Storage	Stores value in each table row	Never stores value
Visual use	Filter, group, or summarize	Designed to summarize

Lab 4

Add measures to Power BI Desktop models

BETA version

Questions?

Microsoft
Power Platform

Power BI
DAX in a Day

Module 05

Use DAX Iterator Functions

BETA version

Module outline

05: Integrate Content with the Power BI Client APIs

- Describe iterator functions
- Use aggregation iterator functions
- Calculate ranks

Describe iterator functions

- **Iterator functions** enumerate all rows of given table and evaluate a given expression for each row
- They provide flexibility over how calculations summarize data
- All iterator functions require *Table* and *Expression*
  - The expression must return scalar or single value
- Iterator functions are easily identified by the appended "X"
- For example:
  - Common aggregation: SUMX, AVERAGEX, COUNTX, MINX, and MAXX
  - Special: CONCATENATEX, and RANKX

Describe iterator functions

Example

```
Measure =
SUMX(
     <Table>,
     <Expression>
)
```

BETA version

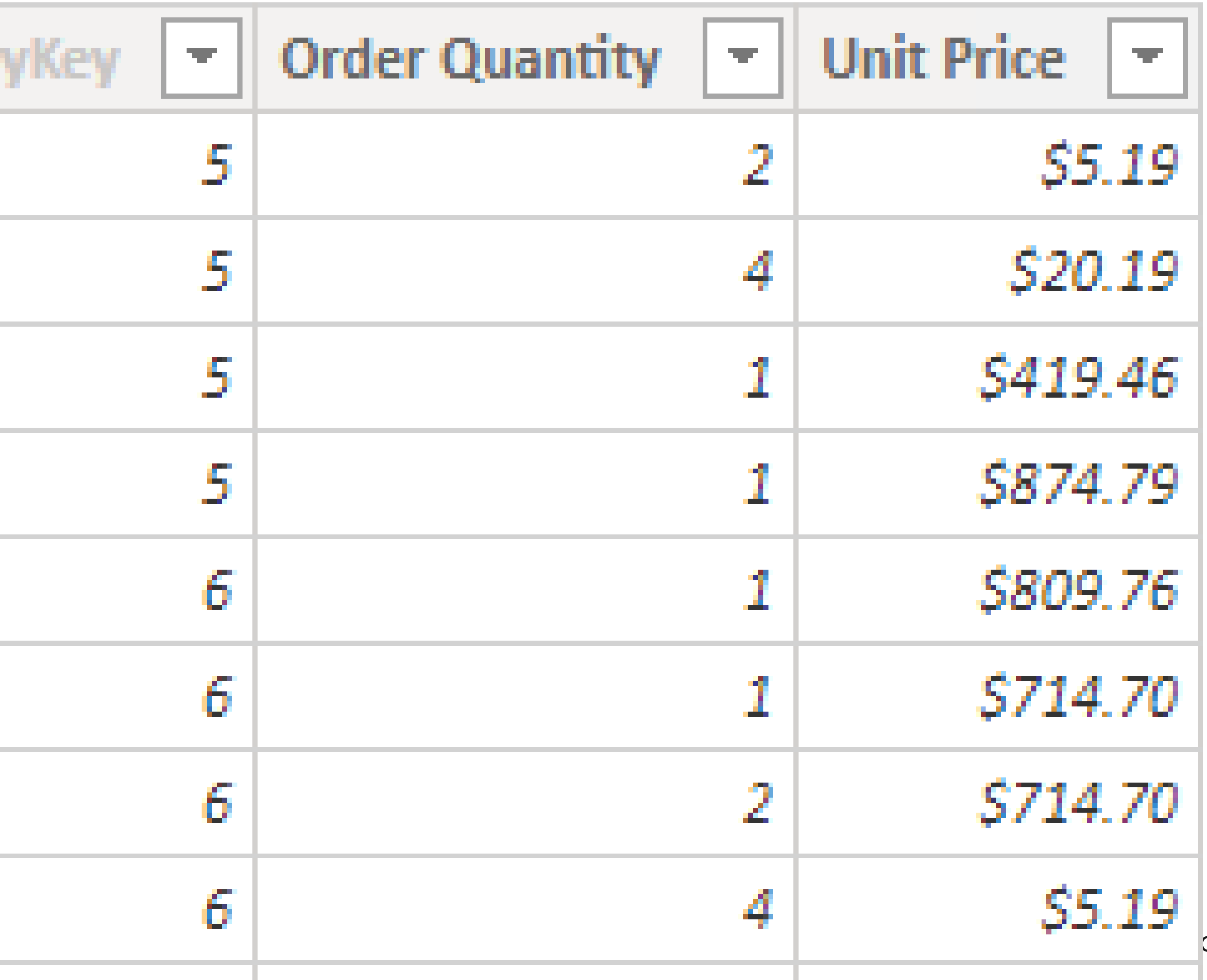
Describe iterator functions

Example

```
Revenue =
SUMX(
     Sales,
     Sales[Order Quantity] * Sales[Unit Price]
)
```

Describe iterator functions

Example



yKey	Order Quantity	Unit Price
5	2	$5.19
5	4	$20.19
5	1	$419.46
5	1	$874.79
6	1	$809.76
6	1	$714.70
6	2	$714.70
6	4	$5.19

10.38
80.76
419.46
874.79
809.76
714.70
1,429.40
…

∑ **208,202.17**

on

Describe iterator functions

Equivalent expressions

```
Revenue = SUM(Sales[Sales Amount])
```

```
Revenue =
SUMX(
      Sales,
      Sales[Sales Amount]
)
```

Describe iterator functions

Evaluation context

- It is important to understand how context works with iterator functions
  - **Table** is evaluated using *filter context*
  - **Expression** is evaluated using *row context*

Calculate ranks

- The **RANKX** function is a special iterator function
- It returns the ranking of a number in a list of numbers for each row in the table argument

```
RANKX ( <table> , <expression> [, <value>[,
<order>[, <ties>]]]])
```

Calculate ranks

Order direction

- Order direction is either ascending or descending
  - Revenue values use descending (favorable)
  - Rank customer complains ascending (unfavorable)
  - Default direction is descending

Calculate ranks

Handle ties

- Either **SKIP** or **DENSE**
- DENSE has no gaps in ranking sequence
  - For example: 1, 2, 2, 3, 4
- SKIP adds gaps to ranking sequence
  - For example: 1, 2, 2, 4, 5
- Default is to SKIP

Lab 5

Use DAX iterator functions in Power BI Desktop models

BETA version

Questions?

Microsoft
Power Platform

Power BI
DAX in a Day

Module 06

Modify Filter Context

BETA version

Module outline

06: Modify Filter Context

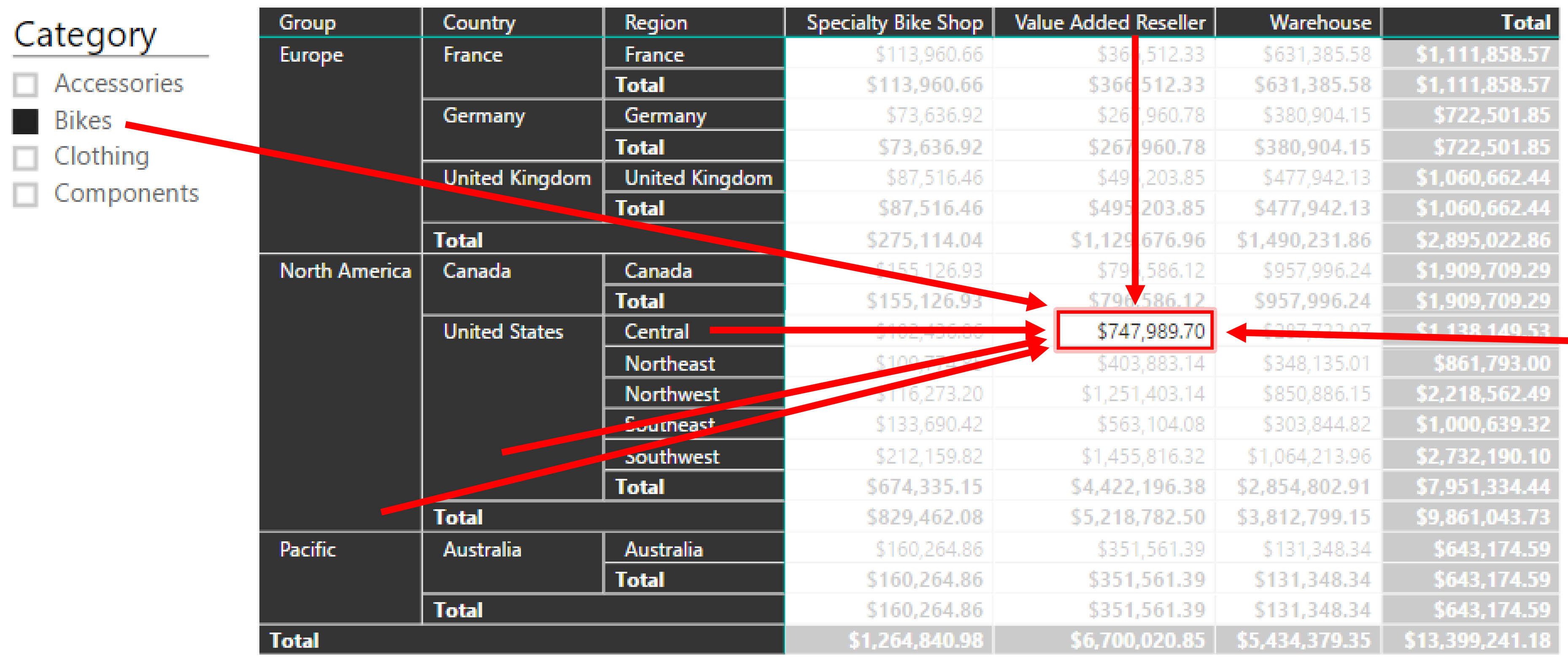
- Describe filter context
- Use the CALCULATE function
- Use filter modifier functions
- Examine filter context
- Perform context transition

Describe filter context

- **Filter context** describes the filters applied during the evaluation of a measure or measure expression
- Filters can be applied *directly* to columns
- Filters can be applied *indirectly* via relationships

Describe filter context

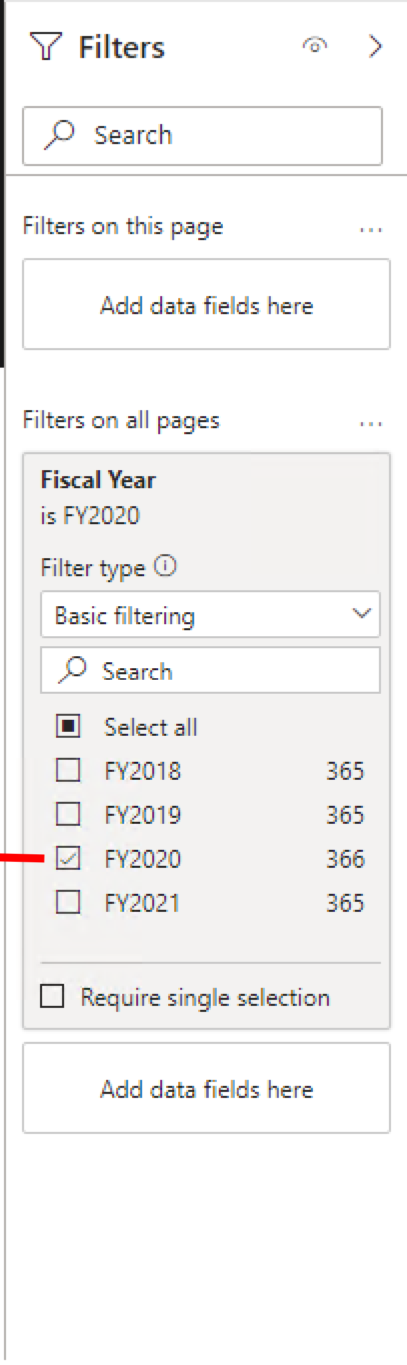
Example



Category

- ☐ Accessories
- ■ Bikes
- ☐ Clothing
- ☐ Components

Group	Country	Region	Specialty Bike Shop	Value Added Reseller	Warehouse	Total
Europe	France	France	$113,960.66	$366,512.33	$631,385.58	$1,111,858.57
		Total	$113,960.66	$366,512.33	$631,385.58	$1,111,858.57
	Germany	Germany	$73,636.92	$267,960.78	$380,904.15	$722,501.85
		Total	$73,636.92	$267,960.78	$380,904.15	$722,501.85
	United Kingdom	United Kingdom	$87,516.46	$495,203.85	$477,942.13	$1,060,662.44
		Total	$87,516.46	$495,203.85	$477,942.13	$1,060,662.44
	Total		$275,114.04	$1,129,676.96	$1,490,231.86	$2,895,022.86
North America	Canada	Canada	$155,126.93	$796,586.12	$957,996.24	$1,909,709.29
		Total	$155,126.93	$796,586.12	$957,996.24	$1,909,709.29
	United States	Central	$102,436.86	$747,989.70	$287,722.97	$1,138,149.53
		Northeast	$109,774.85	$403,883.14	$348,135.01	$861,793.00
		Northwest	$116,273.20	$1,251,403.14	$850,886.15	$2,218,562.49
		Southeast	$133,690.42	$563,104.08	$303,844.82	$1,000,639.32
		Southwest	$212,159.82	$1,455,816.32	$1,064,213.96	$2,732,190.10
		Total	$674,335.15	$4,422,196.38	$2,854,802.91	$7,951,334.44
	Total		$829,462.08	$5,218,782.50	$3,812,799.15	$9,861,043.73
Pacific	Australia	Australia	$160,264.86	$351,561.39	$131,348.34	$643,174.59
		Total	$160,264.86	$351,561.39	$131,348.34	$643,174.59
	Total		$160,264.86	$351,561.39	$131,348.34	$643,174.59
Total			$1,264,840.98	$6,700,020.85	$5,434,379.35	$13,399,241.18



Filters

Search

Filters on this page ...

Add data fields here

Filters on all pages ...

Fiscal Year
is FY2020

Filter type ⓘ
Basic filtering

Search

■	Select all	
☐	FY2018	365
☐	FY2019	365
☑	FY2020	366
☐	FY2021	365

☐ Require single selection

Add data fields here

BETA version

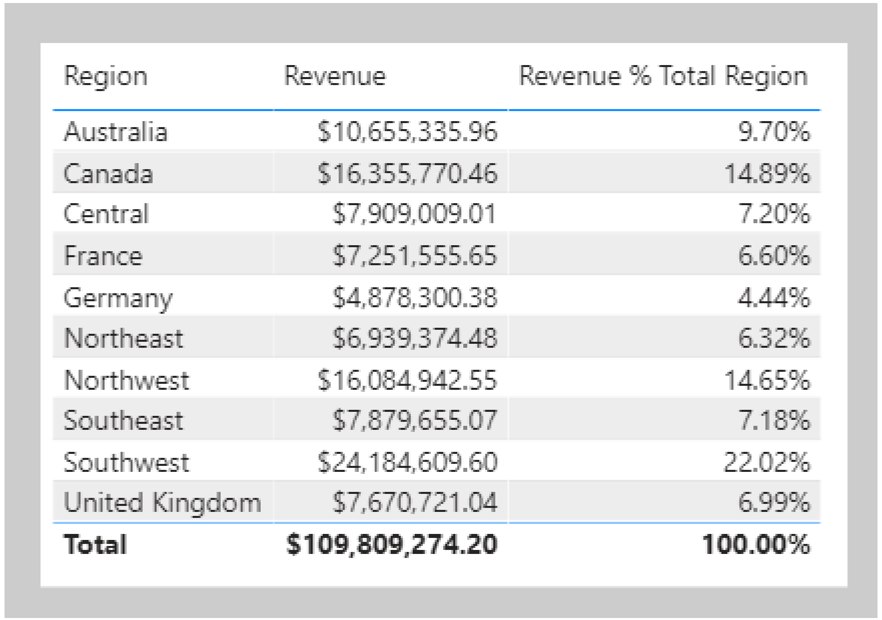
Describe filter context

Filter origin

- Not all filters are applied at report design time
- Report users can add or modify filters when interacting with the report by:
  - Modifying slicer selections
  - Modifying filters in the **Filters** pane
  - Selecting visual elements to cross-filter or cross-highlight other visuals

Describe filter context

Example



Region	Revenue	Revenue % Total Region
Australia	$10,655,335.96	9.70%
Canada	$16,355,770.46	14.89%
Central	$7,909,009.01	7.20%
France	$7,251,555.65	6.60%
Germany	$4,878,300.38	4.44%
Northeast	$6,939,374.48	6.32%
Northwest	$16,084,942.55	14.65%
Southeast	$7,879,655.07	7.18%
Southwest	$24,184,609.60	22.02%
United Kingdom	$7,670,721.04	6.99%
Total	**$109,809,274.20**	**100.00%**

Use the CALCULATE function

```
CALCULATE ( <expression> , [[<filter1>], <filter2>]…)
```

Use the CALCULATE function

Boolean expression filters

- **Boolean expression filters** must evaluate as TRUE or FALSE
- Each filter:
  - Can reference only a single column
  - Can not reference measures
  - Can not use functions that scan or return table

Use the CALCULATE function

Boolean expression filters » Example

```
Revenue Red =
CALCULATE(
     [Revenue],
     'Product'[Color] = "Red"
)
```

Use the CALCULATE function

Boolean expression filters » Example (Continued)

```
Revenue Red =
CALCULATE([Revenue], 'Product'[Color] = "Red")
```

```
Revenue Red =
CALCULATE(
    [Revenue],
    FILTER(
        'Product',
        'Product'[Color] = "Red"
    )
)
```

BETA version

Use the CALCULATE function

Table expression filters

- **Table expression filters** apply a table object as a filter
- Each filter:
  - Can be a reference to model table
  - Can be a DAX function that returns a table object
- It is common to use the FILTER function to create filters

Use the CALCULATE function

Table expression filters » Example

```
Revenue High Margin Products =
CALCULATE(
        [Revenue],
        FILTER(
                'Product',
                'Product'[List Price] > 'Product'[Standard Cost] * 2
        )
)
```

Use the CALCULATE function

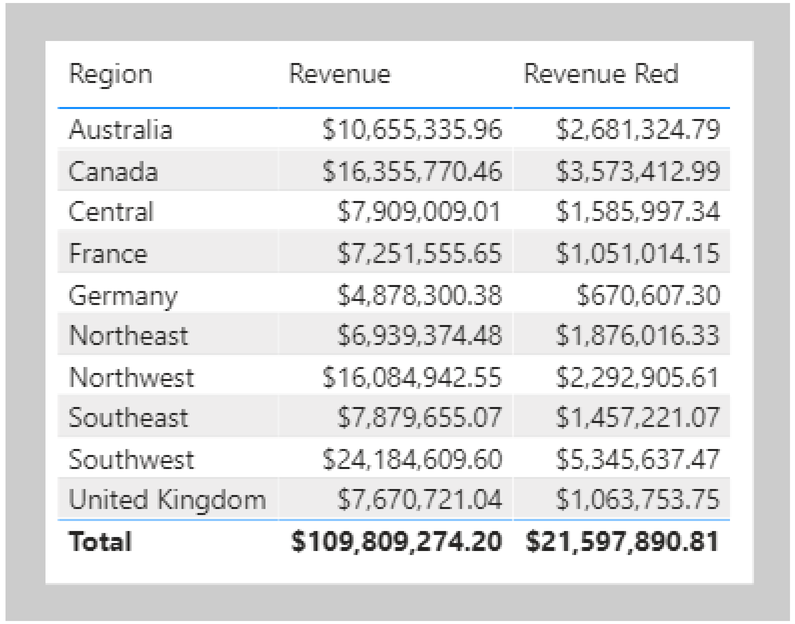
Table filter columns » Filter behavior

- If columns are not in filter context, new filters are **added** to filter context
- If columns already in filter context, existing filters are **overwritten**

Modify filter context

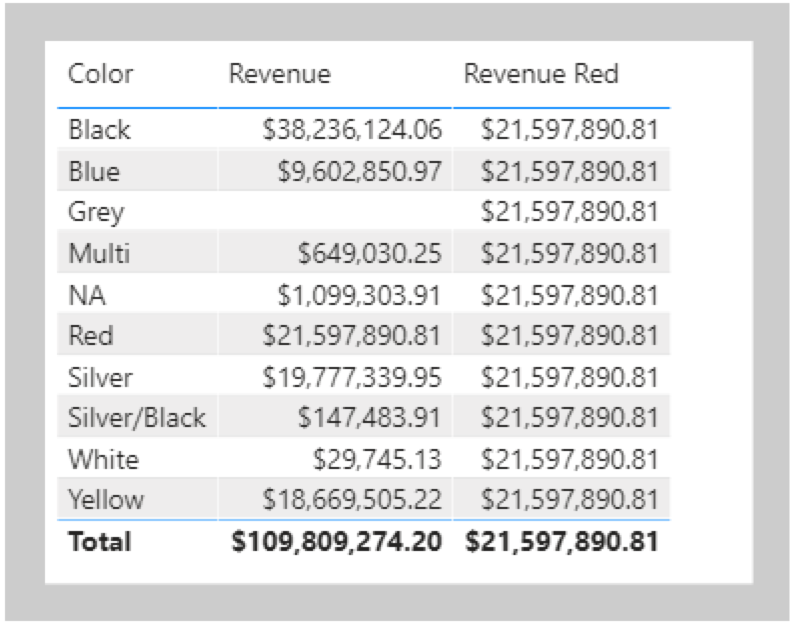
Table filter columns » Filter behavior » Example

New filter added



Region	Revenue	Revenue Red
Australia	$10,655,335.96	$2,681,324.79
Canada	$16,355,770.46	$3,573,412.99
Central	$7,909,009.01	$1,585,997.34
France	$7,251,555.65	$1,051,014.15
Germany	$4,878,300.38	$670,607.30
Northeast	$6,939,374.48	$1,876,016.33
Northwest	$16,084,942.55	$2,292,905.61
Southeast	$7,879,655.07	$1,457,221.07
Southwest	$24,184,609.60	$5,345,637.47
United Kingdom	$7,670,721.04	$1,063,753.75
Total	**$109,809,274.20**	**$21,597,890.81**

Existing filter overwritten



Color	Revenue	Revenue Red
Black	$38,236,124.06	$21,597,890.81
Blue	$9,602,850.97	$21,597,890.81
Grey		$21,597,890.81
Multi	$649,030.25	$21,597,890.81
NA	$1,099,303.91	$21,597,890.81
Red	$21,597,890.81	$21,597,890.81
Silver	$19,777,339.95	$21,597,890.81
Silver/Black	$147,483.91	$21,597,890.81
White	$29,745.13	$21,597,890.81
Yellow	$18,669,505.22	$21,597,890.81
Total	**$109,809,274.20**	**$21,597,890.81**

Use filter modifier functions

- When using CALCULATE, it is possible to pass in filter modification functions
- Filter modifier functions include:
  - **REMOVEFILTERS** – Remove filters from all tables, a single table or column(s)
  - **ALL** – Remove filters from a single table or column(s)
  - **ALLEXCEPT** – Remove filters from all columns of a single table, except those explicitly passed in
  - **ALLNONBLANKROW** – From the parent table of a relationship, returns all rows but the blank row, or all distinct values of a column but the blank row, and disregards any context filters that might exist

Use filter modifier functions

Use inactive relationships

- Use the **USERRELATIONSHIP** function to make an inactive relationship active during the evaluation of the CALCULATE function

```
Revenue Shipped =
CALCULATE(
      [Revenue],
      USERELATIONSHIP(
            'Date'[DateKey],
            'Sales'[ShipDateKey]
      )
)
```

Use filter modifier functions

Modify relationship behavior

- Use **CROSSFILTER** to modify relationship behavior
  - Modify filter direction(s)
  - Disable filters

Examine filter context

- It is possible to examine filter context by using certain DAX functions:
  - VALUES
  - HASONEVALUE
  - SELECTEDVALUE
  - ISFILTERED
  - ISCROSSFILTERED
  - ISINSCOPE

Examine filter context

VALUES function

- Use the **VALUES** function determines what values are in filter context

```
VALUES ( <TableNameOrColumnName> )
```

- Use the **HASONEVALUE** function to determine if the context for a specific column has been filtered down to one distinct value only
- Use the **SELECTEDVALUE** function to return the value when one value is in context

Examine filter context

Other functions

- **ISFILTERED** - Returns TRUE when a passed-in column reference is *directly* filtered
- **ISCROSSFILTERED** - Returns TRUE when a passed-in column reference is *indirectly* filtered
- **ISINSCOPE** - Returns TRUE when a passed-in column reference is the level in a hierarchy of levels

Perform context transition

- When a measure or measure expression is evaluated *within row context*, it undergoes context transition
- In this case, the CALCULATE function applies row context values as filters
- It occurs when:
  - A calculated column formula uses a measure
  - An expression in an iterator function is evaluated

Perform context transition

- Measures used by an expression that is evaluated in row context automatically undergo context transition
- To force context transition, wrap a measure expression inside the CALCULATE function—it does need to pass in any filters

Questions?

Microsoft Power Platform

Power BI
DAX in a Day

Module 07

Use DAX Time Intelligence Functions

BETA version

Module outline

07: Use DAX Time Intelligence Functions

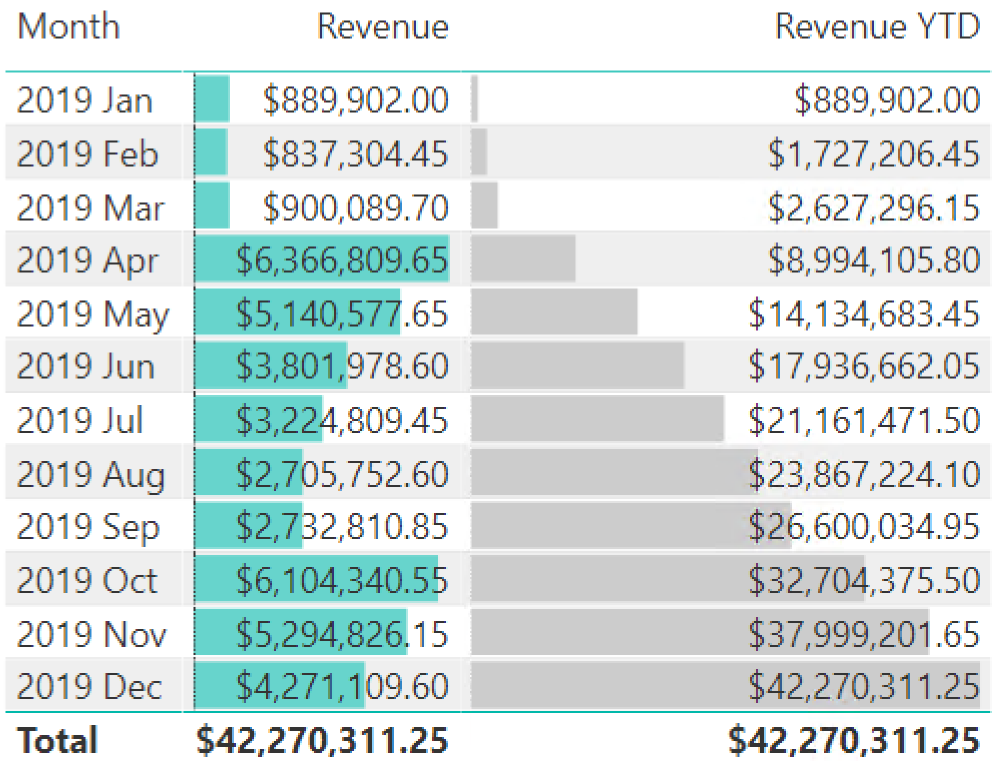
- Define Time Intelligence
- Use Time Intelligence functions

Define Time Intelligence

- **Time Intelligence** relates to calculations over dates, months, quarters, or years
- It involves modifying the filter context for date filters
- The model requires a date table
    - The date table can be created using the CALENDAR or CALENDARAUTO functions

Define Time Intelligence

Example



Month	Revenue	Revenue YTD
2019 Jan	$889,902.00	$889,902.00
2019 Feb	$837,304.45	$1,727,206.45
2019 Mar	$900,089.70	$2,627,296.15
2019 Apr	$6,366,809.65	$8,994,105.80
2019 May	$5,140,577.65	$14,134,683.45
2019 Jun	$3,801,978.60	$17,936,662.05
2019 Jul	$3,224,809.45	$21,161,471.50
2019 Aug	$2,705,752.60	$23,867,224.10
2019 Sep	$2,732,810.85	$26,600,034.95
2019 Oct	$6,104,340.55	$32,704,375.50
2019 Nov	$5,294,826.15	$37,999,201.65
2019 Dec	$4,271,109.60	$42,270,311.25
Total	**$42,270,311.25**	**$42,270,311.25**

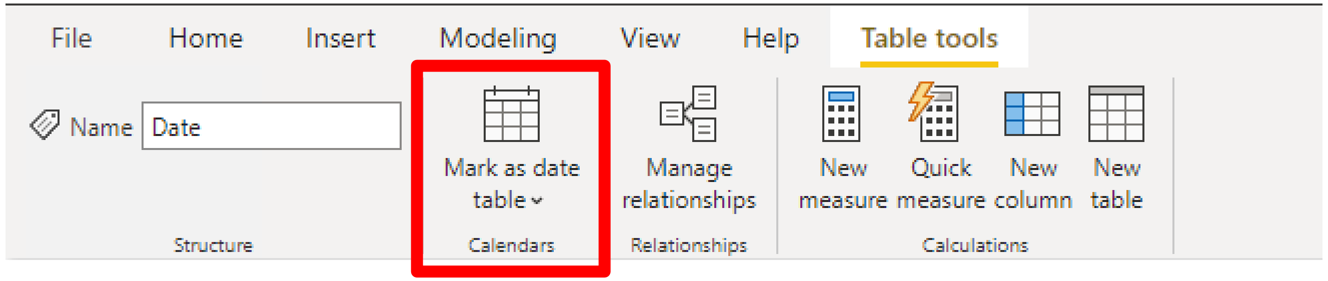
Define Time Intelligence

Date table » Requirements

- The date table must have a Date (or Date/time) column
- The Date column:
  - Must contain unique values
  - Must not contain BLANKS
  - Must not have missing dates (gaps)
  - Must span full years (financial or traditional)
- The date table should be marked as a date table

Define Time Intelligence

Date table » Marking



- To use the Time Intelligence functions, a table must be marked as a date table
  - It also allows the model tables to use a non-date type key
  - The date key can be in ISO format, for example *20210630*
- Multiple model tables can be marked as date tables

Use Time Intelligence functions

- DAX has many inbuilt Time Intelligence functions
  - For example, TOTALYTD and PREVIOUSYEAR
- It is possible to replace them using CALCUALTE
  - However, that requires more work and could be slower for Power BI to evaluate

Use Time Intelligence functions

Common use cases

- Cumulative totals
- Period comparisons
- New customer orders by month
- Stock on-hand value

Use Time Intelligence functions

Summarization over time

- **DATESYTD** – Returns a single-column table of dates
- **TOTALYTD** – Evaluates expression for year-to-date (YTD)
- **DATESBETWEEN** – Returns single-column table with range dates
- **DATESINPERIOD** – Returns single-column table with range of dates

Use Time Intelligence functions

Shifting time periods

- **DATEADD** – Returns single-column table of dates shifted by interval
- **PARALLELPERIOD** – Returns single-column table of parallel dates
- **SAMEPERIODLASTYEAR** – Returns dates shifted back one year
- Many others – NEXTDAY, NEXTMONTH, NEXTQUARTER, NEXTYEAR, PREVDAY, PREVMONTH, PREVQUARTER ,PREVYEAR

Use Time Intelligence functions

Return a single date

- **FIRSTDATE** – Returns first date in current filter context
- **LASTDATE** – Returns last date in current filter context

Use Time Intelligence functions

Snapshot calculations

- Inventory stock levels or account balances
- Useful when tables store snapshot of values
  - For example, inventory stock levels or account balances
- Calculations can summarize across any dimension *except date*
  - For example, stock level counts by product category works, but not by date range

Use Time Intelligence functions

Snapshot calculations » Functions

- **LASTNONBLANK** – Returns the last date that produces a non-BLANK result
- **FIRSTNONBLANK** – Returns the first date that produces a non-BLANK result
- **LASTNONBLANKVALUE** – Returns the value of the last date that produces a non-BLANK result
- **FIRSTNONBLANKVALUE** – Returns the value of the first date that produces a non-BLANK result

Lab 7

Use DAX Time Intelligence functions in Power BI Desktop models

BETA version

Questions?

Resources

Learn: Use DAX in Power BI Desktop

This learning path introduces Data Analysis Expressions (DAX) and provides you with foundational skills required to enhance data models with calculations

https://docs.microsoft.com/learn/paths/dax-power-bi/

Docs: Data Analysis Expressions (DAX) Reference

https://docs.microsoft.com/dax/

DAX videos

https://docs.microsoft.com/dax/dax-learn-videos

Post training survey

- Please complete post training survey to help us gather feedback
  - https://aka.ms/DAXIADSurvey

Microsoft
Power Platform

Microsoft
Power Platform

Power BI
DAX in a Day

Appendix 01

Knowledge Checks by Modules with answers

BETA version

Knowledge check 01A

In a Power BI Desktop model design, which type of object enforces row-level security?

A. Table

B. Column

C. Measure

[✓ D. Role]

Knowledge check 01B

Which of the following statements is correct regarding a star schema design?

A. [✓ Fact tables store accumulations of business events]
B. Fact tables store accumulations of business entities
C. Fact tables must have a unique column

BETA version

Knowledge check 01C

In what order does an analytic query implement its phases?

[✓ A. Filter ► Group ► Summarize]

B. Group ► Filter ► Summarize

C. Summarize ► Filter ► Group

Knowledge check 02A

You're using Power BI Desktop to develop a model. It has a table named **Sales**, which includes a column named **CustomerKey**. In reports, you need a calculation to show the number of different customers who have placed orders. What type of DAX calculation will you add to the **Sales** table?

A. Calculated table

B. Calculated column

C. Computed column

[✓ D. Measure]

Knowledge check 02B

You're using Power BI Desktop to develop a model. It has a table named **Customer**, which includes a column named **DateOfBirth**. In reports, you need to group customers by current age. What type of DAX calculation will you add to the **Customer** table?

A. Calculated table
[✓ B. Calculated column]
C. Computed column
D. Measure

Knowledge check 02C

You're using Power BI Desktop to develop a model. It has a table named **Geography**, which has two relationships to the **Sales** table. One relationship filters by customer region and the other filters by sales region. Report users need to filter or group **Sales** table data by either region. What type of DAX calculation will you add to the model?

[✓ A. Calculated table]

B. Calculated column

C. Computed column

D. Measure

Knowledge check 02D

You write a DAX formula that adds BLANK to the number 20.
What will be the result?

A. The result will be zero (0)

[✓ B. The result will be 20]

C. The result will be BLANK

D. The result will be NULL

Knowledge check 03A

Which statement about calculated tables is true?

A. [✓ Calculated tables increase the storage size of the data model]
B. Calculated tables are evaluated by using row context
C. Calculated tables are created in Power Query
D. Calculated tables cannot include calculated column

Knowledge check 03B

Which statement about calculated columns is true?

A. Calculated columns are created in the Power Query Editor window

[✓ B. Calculated column formulas are evaluated by using row context]

C. Calculated column formulas can only reference columns from within their table

D. Calculated columns can't be related to non-calculated columns

Knowledge check 03C

You're developing a Power BI desktop model that sources data from a Microsoft Azure SQL database. The database has an employee table that stores one row for each employee. Each row has a reference to the employee's manager, which is also a row in the employee table. You need to add several columns to the Employee table in your model to analyze payroll data within the organization hierarchy. Which technique will you use to add the columns?

A. Add persisted columns to a table by using T-SQL

B. Add column expressions in a view by using T-SQL

C. Add computed columns by using M

[✓ D. Add calculated columns by using DAX]

Knowledge check 04A

Which statement about measures is correct?

A. Measures store values in the data model

B. Measures must be added to the data model to achieve summarization

C. Measures can reference columns directly

[✓ D. Measures can reference other measures directly]

Knowledge check 04B

Which DAX function can summarize a table?

A. SUM

B. AVERAGE

[✓ C. COUNTROWS]

D. DISTINCTCOUNT

Knowledge check 04C

Which of the following statements describing similarity of measures and calculated columns in an Import model is true?

A. [✓ They can achieve summarization of model data]
B. They store values in the data model
C. They're evaluated during data refresh
D. They can be created by using quick calculations

Knowledge check 05A

An iterator function always includes at least two arguments.
What are they?

A. Column and expression
[✓ B. Table and expression]
C. Measure and expression
D. Table and ranking

BETA version

Knowledge check 05B

Which statement about DAX iterator functions is true?

A. When used to add values of the same column, the SUM function performs better than the SUMX function

B. When used to add values of the same column, the SUMX function performs better than the SUM function

C. Iterator functions iterate over tables and evaluate an expression for each table row

[✓ D. Iterator functions iterate over expressions and evaluate tables for each row]

Knowledge check 05C

You're developing a Power BI Desktop model. You need to create a measure formula by using the RANKX DAX function to rank students by their test results. The lowest rank should be assigned to the highest test result. Also, if multiple students achieve the same rank, the next student rank should follow on from the tied rank number. Which order and ties arguments should you pass to the RANKX function?

A. Order Ascending | Ties Skip

[✓ B. Order Ascending | Ties Dense]

C. Order Descending | Ties Skip

D. Order Descending | Ties Dense

Knowledge check 06A

Which type of model object is evaluated within a filter context?

A. Calculated column
B. Calculated table
[✓ C. Measure]
D. Security role rule

BETA version

Knowledge check 06B

Which DAX functions allows you to use an inactive relationship when evaluate a measure expression?

A. [✓ USERELATIONSHIP]
B. CROSSFILTER
C. REMOVEFILTERS
D. ISCROSSFILTERED

Knowledge check 06C

Which statement about the CALCULATE function is true?

A. You must pass in at least one filter argument

[✓ B. It modifies filter context to evaluate a given expression]

C. It modifies row context to evaluate a given expression

D. It cannot be used by a calculated column formula

Knowledge check 07A

In the context of data model calculations, which statement best describes time intelligence?

A. Snapshot balance reporting

[✓ B. Filter context modifications involving a date table]

C. Complex calculations involving time

D. Calculations involving hours, minutes, or seconds

Knowledge check 07B

You're developing a data model in Power BI Desktop. You've just added a date table by using the CALENDARAUTO function. You've extended it with calculated columns, and you've related it to other model tables. What else must you do to ensure DAX time intelligence calculations work correctly?

A. Add time intelligence measures to the date table

[✓ B. Mark the date table]

C. Add a fiscal hierarchy

D. Add a date column

Knowledge check 07C

You have a table that stores account balance snapshots for each date, excluding weekends. You need to ensure that your measure formula only filters by a single date. Also, if there’s no record on the last date of a time period, it should use the latest account balance.

Which DAX time intelligence function should you use?

A. FIRST

B. FIRSTNONBLANK

C. LAST

[✓ D. LASTNONBLANK]