DAX Functions in Power BI

Table of Content

1. Setting up Data Models with DAX

DAX stands for data analysis expressions

- DAX is a formula expression language used in multiple Microsoft analytics tools

- DAX formulas include functions, operators and values to perform advanced calculations
- DAX formulas are used in:
  - **Measures**
  - **Calculated**
  - **columns**
  - **Calculated tables**

    Row-level security

The power of DAX

- It opens up new capabilities:
  - Joins, filters, measures and calculated fields become part of your toolbox
- DAX + Power Query = a powerful data analysis tool:
  - Dive deeper into the data and extract key insights
  - Use DAX for rapid prototyping

Measures vs calculated columns

Calculated Columns:

- Calculated on data import
- Visible in data & report Pane

```
COST = Orders[Sales] - Orders[Profit]
```

Order_ID	Sales	Pofit	Cost
3151	$77.88	$3.89	$73.99
3152	$6.63	$1.79	$4.84
3153	$22.72	$10.22	$12.50
3154	45.36	$21.77	$23.59

Measures vs calculated columns

Calculated Columns:

- Calculated on data import
- Visible in data & report Pane

```
COST = Orders[Sales] - Orders[Profit]
```

Order_ID	Sales	Pofit	Cost
3151	$77.88	$3.89	$73.99
3152	$6.63	$1.79	$4.84
3153	$22.72	$10.22	$12.50
3154	45.36	$21.77	$23.59

Measures:

- Calculated at query run-time
- Visible only in report pane

```
Total Sales = SUM(Orders[Sales])
```

Region	Total Sales
Central	$501,239.89
East	$678,781.24
West	$391,721.91
South	$725.457.82
Total	**$2,297,200.86**

Context allows you to perform dynamic analysis

There are three types of context: row, query and filter context

- Row context: (1)
  - "The current row"
  - DAX calculated columns

```
COST = Orders[Sales] - Orders[Profit]
```

Context allows you to perform dynamic analysis

There are three types of context: row, query and filter context

- Row context: (1)
  - "The current row"
  - DAX calculated columns

```
COST = Orders[Sales] - Orders[Profit]
```

Order_ID	Sales	Pofit	Cost
3151	$77.88	$3.89	$73.99
3152	$6.63	$1.79	$4.84
3153	$22.72	$10.22	$12.50
3154	45.36	$21.77	$23.59

Context allows you to perform dynamic analysis

There are three types of context: row, query and filter context

- Query context: (2)
  - Refers to the subset of data that is implicitly retrieved for a formula
  - Controlled by slicers, page filters, table columns and row headers
  - Controlled by chart/visual filters
  - Applies after row context

Context allows you to perform dynamic analysis

- Query context: (2)
  - *Example:* Filter data by Region.

Region	Total Sales
Central	$501,239
East	$678,781
West	$391,721
South	$725.457

- Query context: (2)
  - *Example:* Filter data by State.

State	Total Sales
Alabama	$13,724
Arizona	$38,710
Arkansas	$7,669
California	$381,306

Context allows you to perform dynamic analysis

There are three types of context: row, query and filter context

- Filter Context: (3)
  - The set of values allowed in each column, or in the values retrieved from a related table
  - By using arguments to a formula or by using report filters on row and column headings
  - Applies after query context

Context allows you to perform dynamic analysis

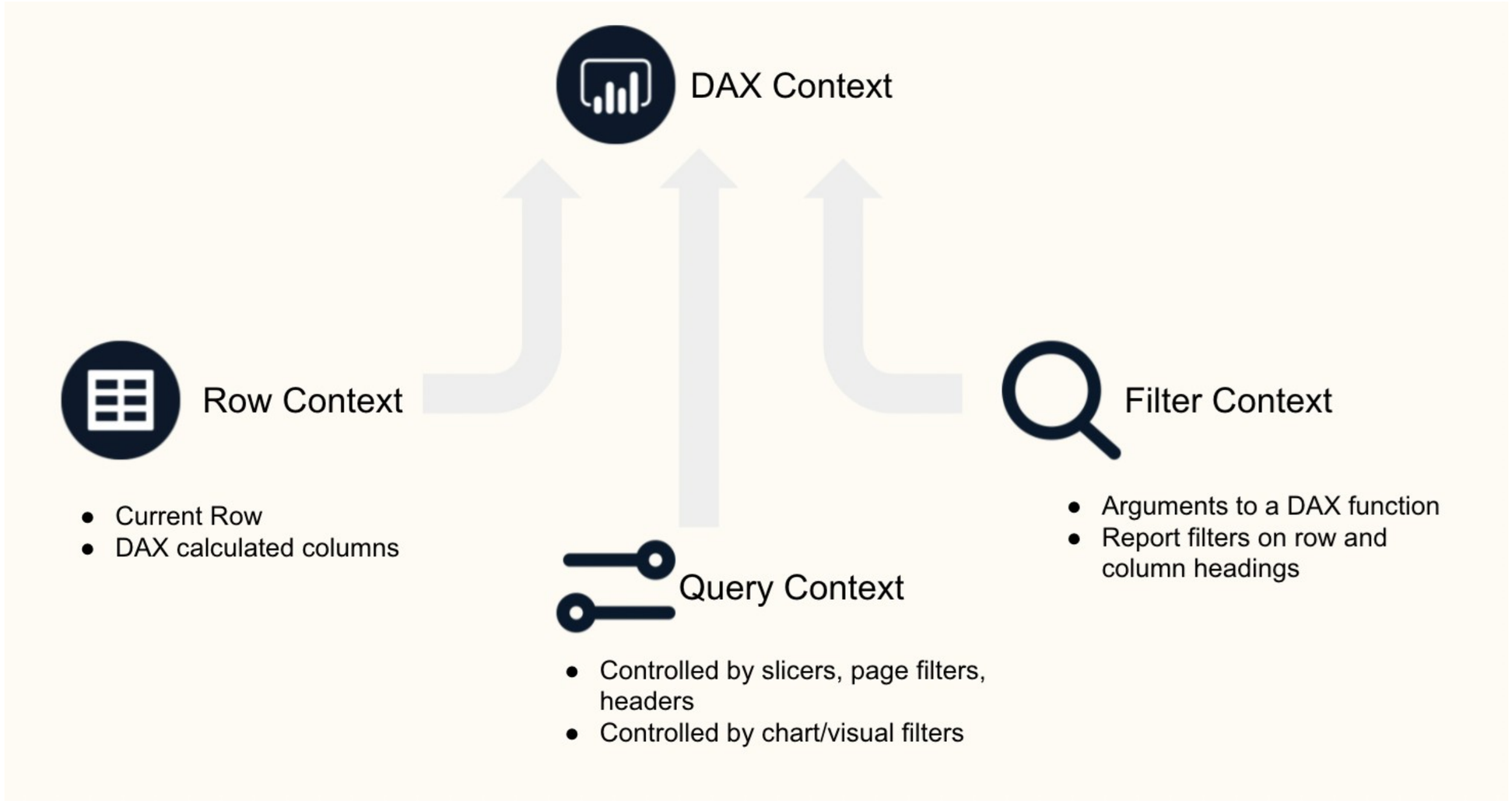
There are three types of context: row, query and filter context.

- Filter Context (3)

```
Total Costs East = CALCULATE([Total Costs], Orders[Region] = 'East')
```

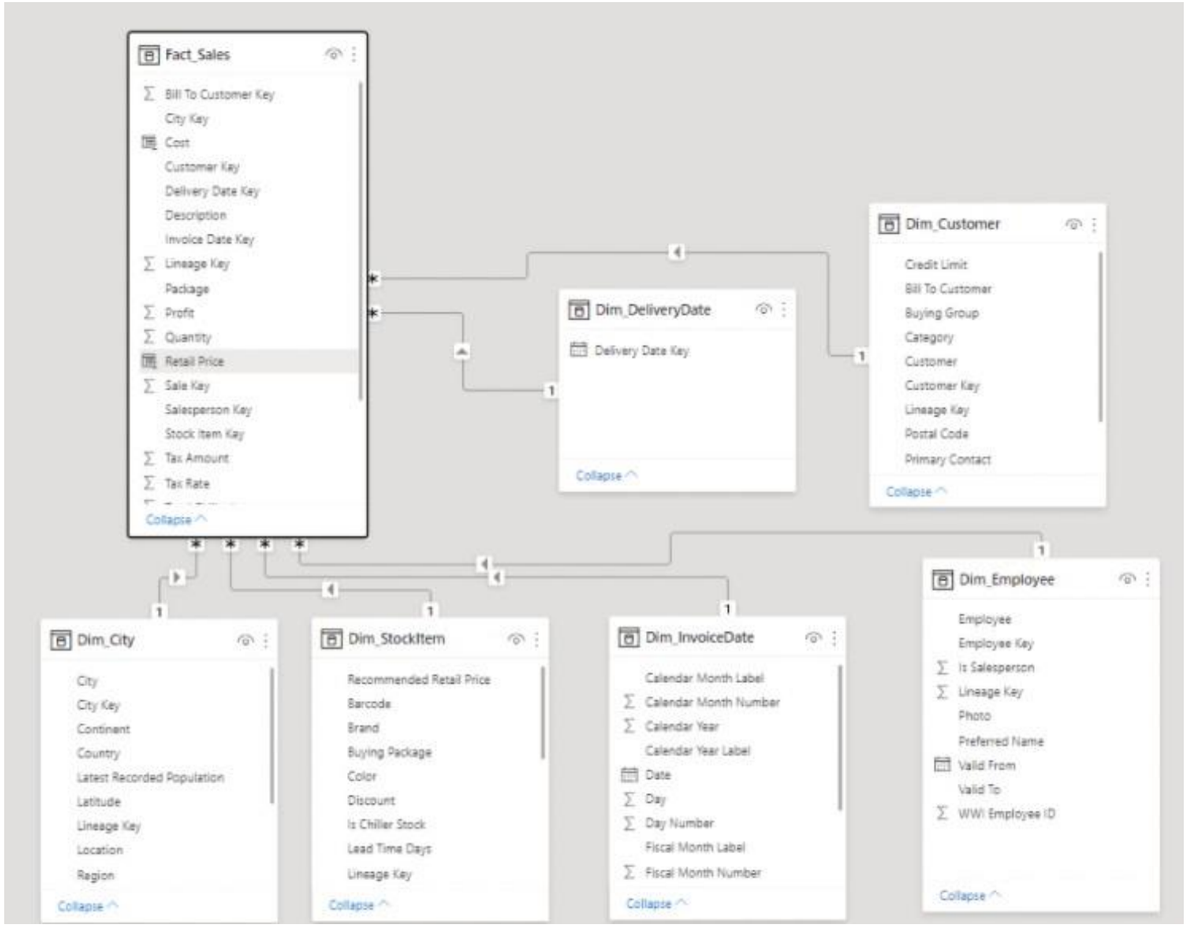
Region	Total costs	Total costs East
Central	$617,039	
East	$587,258	$587,258
West	$461,534	
South	$344,972	
Total	**$2,010,804**	**$587,258**

Context in a nutshell



World wide importers dataset

- A fictitious wholesaler who imports and distributes novelty goods
- The dataset consists of:
  - A fact table that detailing sales transactions
  - Multiple other dimension tables:
    - Dates
    - Customers
    - Cities
    - Employees
    - Stock Items



Demo

2. DAX and Measures

Implicit vs explicit measures

Implicit

- Automatically created by Power BI
- Comes directly from the Database
- E.g.: If we drag `Sales` to values of a table, Power BI will automatically sum it
- Using a dropdown menu we can define the aggregation: sum, average, count, ...

Explicit

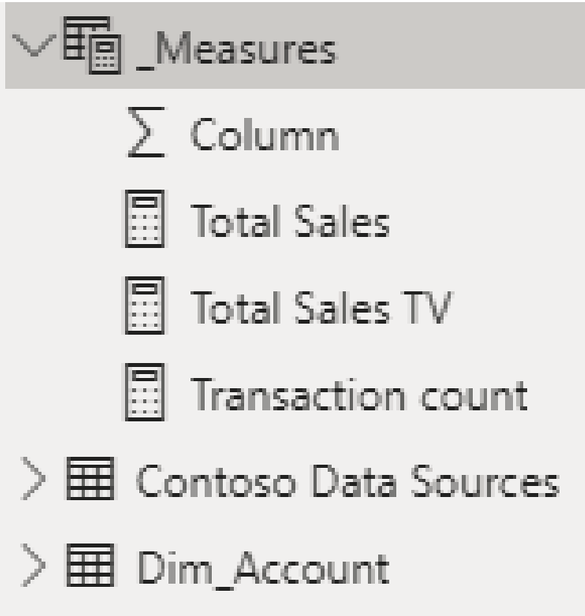
- Writing measures in an explicit way
- E.g.: `Total Sales = SUM(Orders[Sales])`
- Offer flexibility

Why explicit measures are preferred

- Reduces confusion of what a measure is or does
  - `Total Sales = SUM(Orders[Sales])`
  - `Total Sales` is more clear than `Sales` (SUM, AVG, MIN, ... ?)
- Reusable within other measures
  - `Total Sales East = CALCULATE([Total Sales],Orders[Region] = 'East')`
- Can be given a custom name to explain its functionality
- Makes maintenance of complex models more sustainable

Best practices

- **Keep DAX measures grouped together:**
  - Measures are free to move to any table
  - This is in contrast with calculated columns, which belong to a specific table



- **Format and comment with**
  - **DAX:** Use indentations to increase understanding
    - Shift Enter to start a new line
    - Tab to indent
  - Add comments after a //

Use variables to improve your formulas

- Stores the result of an expression as a named variable
- Can be used as an argument to other measure expressions
- Four main advantages:
  - Improve performance
  - Improve readability
  - Simplify debugging
  - Reduce complexity

Syntax:

- `VAR <name> = <expression>`
  - Name = The name of the variable
  - A DAX expression which returns a scalar or table value
  - Followed by a `RETURN` statement

Use variables to improve your formulas - example

- Calculate the sales from last year and store it as a variable

```
VAR
SALESPRIORYEAR = CALCULATE([SALES],SAMEPERIODLASTYEAR('DATE'))
RETURN
```

- Use the variable in a formula

```
Sales growth = [Sales] - SALESPRIORYEAR
```

Use variables to improve your formulas - example

- All together it would look like this:

```
Sales growth =


VAR
SALESPRIORYEAR = CALCULATE([SALES],SAMEPERIODLASTYEAR('DATE'))
RETURN


Sales growth = [Sales] - SALESPRIORYEAR
```

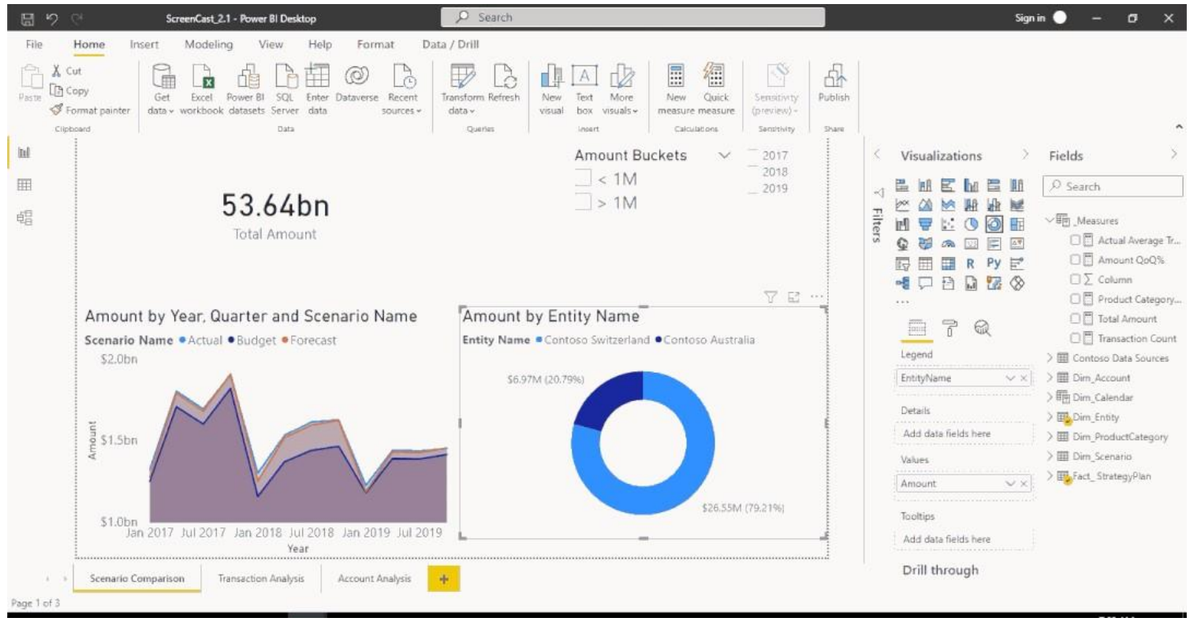
Demo

3. Filtering and counting with DAX

Filter functions

- Filters are applied on the filter context
- Filters take precedence over any visual

```
Total Sales = SUM(Orders[Sales])
```



Filter functions

- Filters are applied on the filter context
- Filters take precedence over any visual

```
Total Sales = SUM(Orders[Sales])
```

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

- Used with intermediate functions

```
Total Sales ALL = CALCULATE(
                    [Total Sales],
                    ALL(Orders))
```

Region	Total Sales
Central	$501,239.89
East	$678,781.24
South	$391,721.91
West	$725,457.82
TOTAL	**$2,297,200.86**

Filter functions

- Filters are applied on the filter context
- Filters take precedence over any visual

```
Total Sales = SUM(Orders[Sales])
```

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

- Used with intermediate functions

```
Total Sales ALL = CALCULATE(
                    [Total Sales],
                    ALL(Orders))
```

Region	Total Sales	Total Sales ALL
Central	$501,239.89	$2,297,200.86
East	$678,781.24	$2,297,200.86
South	$391,721.91	$2,297,200.86
West	$725,457.82	$2,297,200.86
TOTAL	**$2,297,200.86**	**$2,297,200.86**

More filter options

- `FILTER(<table>, <filter>)`
  - *Returns a filtered table*

```
Total Sales Chuck =
CALCULATE(
    [Total Sales],
    FILTER(Fact_Orders,
        RELATED(Dim_Sales[Salesperson]) = "Chuck"))
```

More filter options

- `FILTER(<table>, <filter>)`
  - *Returns a filtered table*

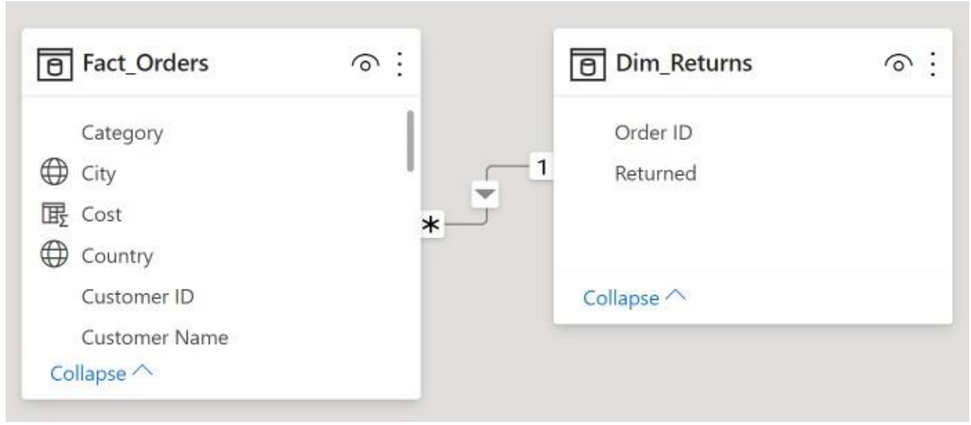
```
Total Sales Chuck =
CALCULATE(
    [Total Sales],
    FILTER(Fact_Orders,
        RELATED(Dim_Sales[Salesperson]) = "Chuck"))
```

Total Sales	Total Sales Chuck
$2,297,200.86	$235,856.05

- `RELATED()` is used to return values from another table

More filter options

- `CROSSFILTER(<col1>, <col2>, <direction>)`
  - *Specifies the cross-filtering direction between two columns*




```
CROSSFILTER(Dim_Returns[Order ID],
            Fact_Orders[Order ID],
            Both)
```

- Overrides relationship direction of data model

The benefits of filtering in DAX

- Improves performance
  - Filter out unnecessary data
  - Define specific relationships between tables
- Reusability
  - Refer to other calculated measures
- More complex computations
  - Concise syntax

Counting

- `COUNT(<column>)`
  - *Returns the amount of rows with numbers, dates, or strings in a column*
- `COUNTA(<column>)`
  - *Returns the amount of rows with numbers, dates, strings, or booleans in a column*
- `COUNTBLANKS(<column>)`
  - *Returns the amount of blank rows*
- `DISTINCTCOUNT(<column>)`
  - *Returns the amount of distinct values in a column*
- `COUNTROWS(<table>)`
  - *Returns the amount of rows with numbers, dates, and strings in a table*

Demo

4. Iterating functions

Iterating functions

- Iterate over each row of a given table to perform an expression

`SUMX(<table>, <expression>)` `AVERAGEX(<table>, <expression>)`

- X stands for eXpression
- Allow for advanced calculations specified at each row

Iterating functions: SUMX()

Calculated column example

```
Cost = Fact_Orders[Sales] - Fact_Orders[Profit]
```

```
Total Costs = SUM(Fact_Orders[Cost])
```

Sales	Profit	Cost
$77.88	$3.89	$73.99
$22.72	$10.22	$12.50
...	...	...

Total Costs
$2,569

Iterating functions: SUMX()

Calculated column example

```
Cost = Fact_Orders[Sales] - Fact_Orders[Profit]
```

```
Total Costs = SUM(Fact_Orders[Cost])
```

Sales	Profit	Cost
$77.88	$3.89	$73.99
$22.72	$10.22	$12.50
...	...	...

Total Costs
$2,569

Iterating function example

```
Total Costs SUMX =
SUMX(Fact_Orders,
     Fact_Orders[Sales] - Fact_Orders[Profit])
```

Total Costs SUMX
$2,569

Filtering iterating functions

- Use filter functions, such as FILTER(), to return a filtered table

```
SUMX(
    FILTER(
        <table>,
        <filter>),
    <expression>)
```

```
Total Costs East SUMX =
SUMX(
    FILTER(
        Fact_Orders,
        Fact_Orders[Region] = "East"),
    Fact_Orders[Sales] - Fact_Orders[Profit])
```

Filtering iterating functions

- Use filter functions, such as FILTER(), to return a filtered table

```
SUMX(
    FILTER(
        <table>,
        <filter>),
    <expression>)
```

```
Total Costs East SUMX =
SUMX(
    FILTER(
        Fact_Orders,
        Fact_Orders[Region] = "East"),
    Fact_Orders[Sales] - Fact_Orders[Profit])
```

Region	Total Costs	Total Costs East SUMX
Central	$501,239.89	
East	$678,781.24	$678,781.24
South	$391,721.91	
West	$725,457.82	
TOTAL	**$2,297,200.86**	**$678,781.24**

Iterating functions: RANKX()

```
RANKX(
    <table>,
    <expression>)
```

- Rank regions by total costs

```
Total Costs RANKX =
RANKX(
    ALL(Dim_Sales[Region]),
    [Total Costs])
```

- Use `ALL()` to evaluate all rows from the dimension table

Iterating functions: RANKX()

```
RANKX(
    <table>,
    <expression>)
```

- Rank regions by total costs

```
Total Costs RANKX =
RANKX(
    ALL(Dim_Sales[Region]),
    [Total Costs])
```

- Use `ALL()` to evaluate all rows from the dimension table

Region	Total Costs	Total Costs RANKX
Central	$725,457.82	1
East	$678,781.24	2
South	$501,239.89	3
West	$391,721.91	4

Operators in DAX

COMPARISON OPERATORS

Operator	Meaning
=	Equal to
==	Strict equal to
>	Greater than
<	Smaller than
>=	Greater than or equal to
<=	Smaller than or equal to
<>	Not equal to

Operators in DAX

COMPARISON OPERATORS

Operator	Meaning
=	Equal to
==	Strict equal to
>	Greater than
<	Smaller than
>=	Greater than or equal to
<=	Smaller than or equal to
<>	Not equal to

TEXT OPERATOR

Operator	Meaning	Example
&	Concatenates text values	[City]&", "&[State]

Operators in DAX

COMPARISON OPERATORS

Operator	Meaning
=	Equal to
==	Strict equal to
>	Greater than
<	Smaller than
>=	Greater than or equal to
<=	Smaller than or equal to
<>	Not equal to

TEXT OPERATOR

Operator	Meaning	Example
&	Concatenates text values	[City]&", "& [State]

LOGICAL

Operator	Meaning	Example
&&	`AND` condition	([City] = "Bru") && ([Return] = "Yes"))
||	`OR` condition	([City] = "Bru") || ([Return] = "Yes"))
IN { }	`OR` condition for each row	Product[Color] IN {"Red", "Blue", "Gold"}

Demo

Congratulatios!

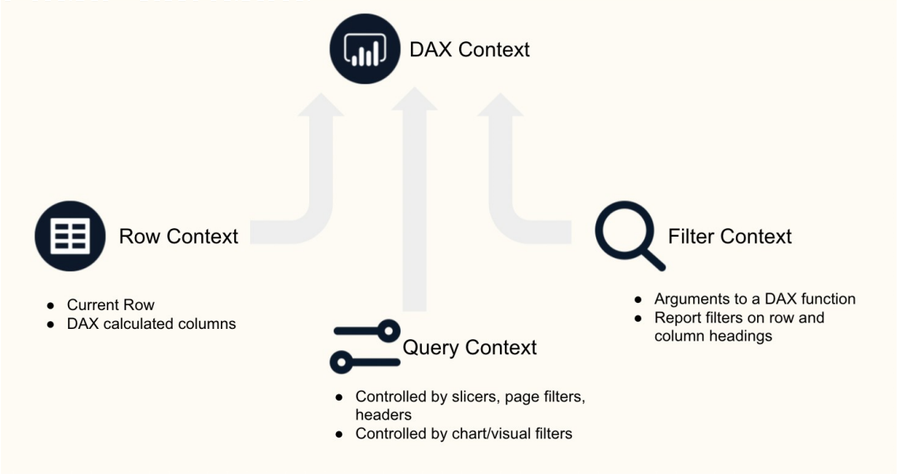
DAX stands for Data Analysis eXpressions

DAX formulas are used in:

- Measures
- Calculated columns
- Calculated tables

Context in DAX Formulas:

- Row context
- Query context
- Filter context



DAX Toolbox

General:

- Implicit vs explicit measures
- Quick measures
- Variables: `VAR`

Data Modeling:

- `CALENDAR()`
- `CALCULATE()`
- `RELATED()`
- `FILTER()`
- `CROSSFILTER()`