

The background of the image features a large, faint, stylized head of an animal, likely a beaver or similar rodent, which is the iconic logo for O'Reilly. The head is composed of several overlapping circular shapes in various shades of orange and red, creating a layered, organic effect. The text "O'REILLY" is centered horizontally and partially overlaps the lower portion of the animal head logo.

O'REILLY®



Part of the Power BI
Bootcamp Series

Microsoft Power BI Bootcamp

— — — —
Introduction to Business Intelligence & PBI

Instructors:

Nicolás Lagreste Zucchini

Maria Florencia Hourcouri



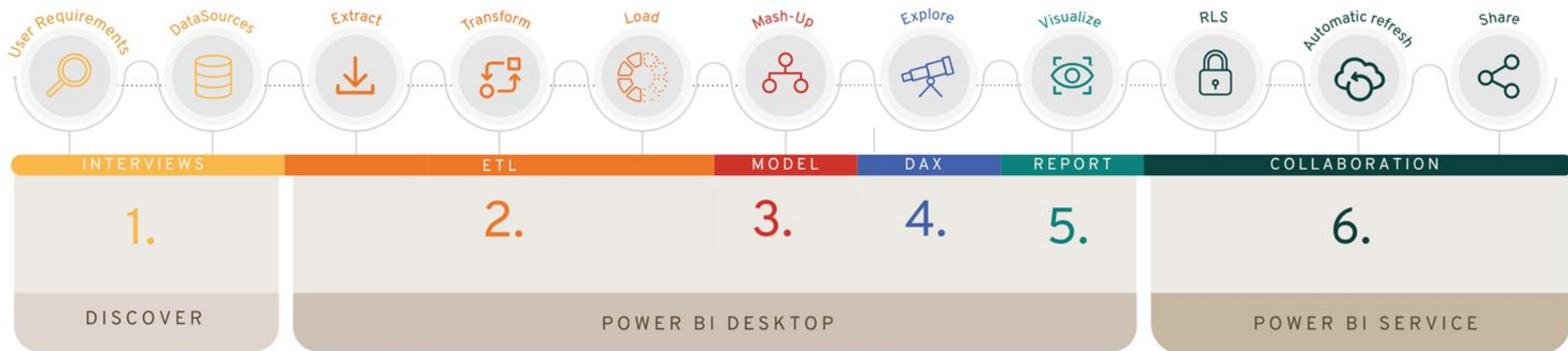


Course Agenda

- **Week 1:** Introduction to Business Intelligence, Power BI and Power Query.
- **Week 2:** Advanced and Interactive Visuals.
- **Week 3:** Calculated measures, columns & custom tables with Power BI Programming Language (DAX).
- **Week 4:** Ways to collaborate and share in Power BI.

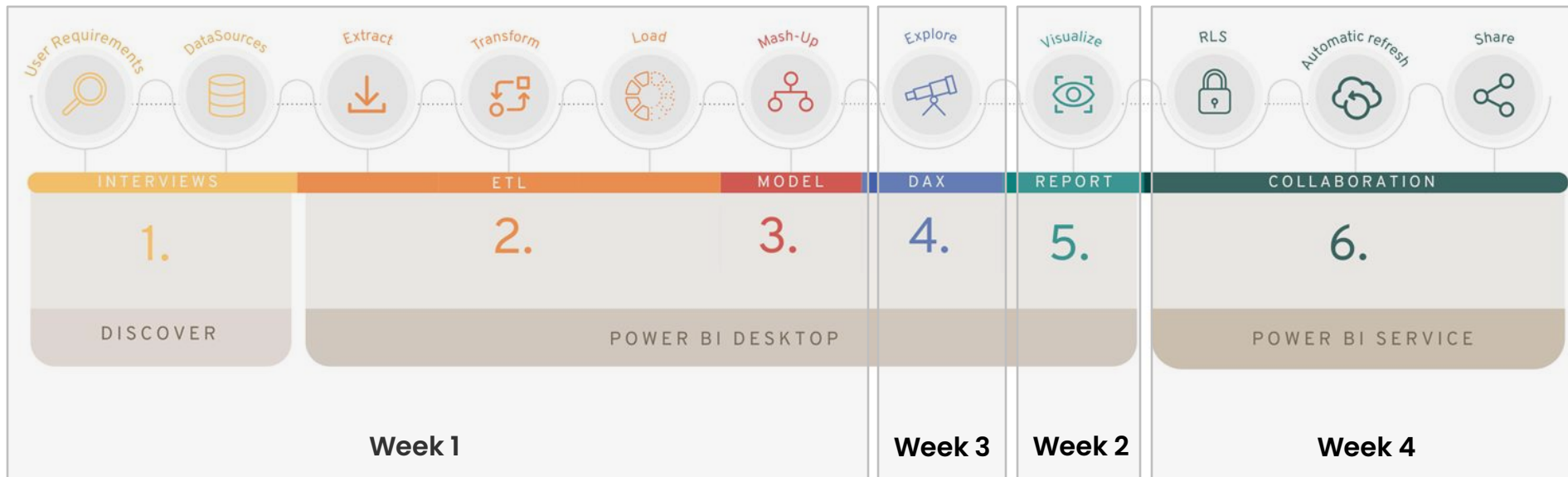


Our Method: Flow of Report Development





Our Method: Flow of Report Development





Instructors

Follow us on LinkedIn:



Nicolás Lagreste Zucchini



María Florencia Hourcouripé

Let's Collaborate: Interested in a personalized mentorship or consulting session? Contact us:

- nico@analyticmood.com
- flor@analyticmood.com

Visit Our Website:

- www.analyticmood.com for resources, upcoming events, and blog articles.

analytic mood

Exercise 0.0 – Initial set up

- ✓ To get started, download all the files from:
www.analyticmood.com/resources
- ✓ Ensure all files are downloaded before proceeding.



NOTE: use the latest version of Power BI Desktop.



O'REILLY®

WEEK 03

Calculated measures,
columns & custom
tables with Power BI
Programming
Language (DAX).



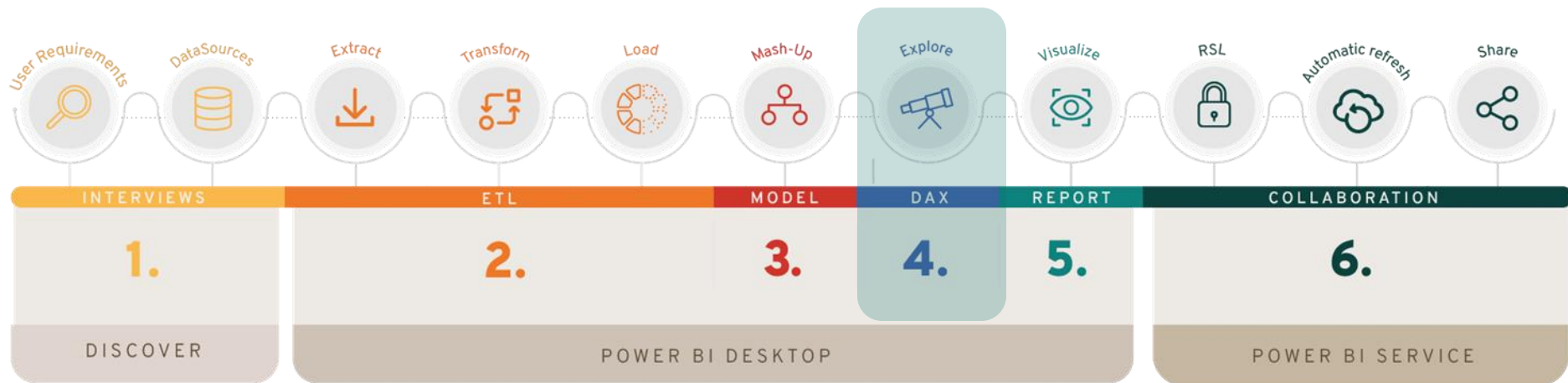
Week 3: Calculated measures, columns & custom tables with Power BI Programming Language (DAX).

- How to solve several basic calculation and data analysis problems by introducing Data Analysis Expressions (DAX).
- How to enhance data models with calculations. It begins by describing the structure of the Power BI Desktop model and how it can be enhanced with DAX calculations.
- Filter context and row context.
- How DAX formulas can be written and the different types of model calculations, including calculated tables, columns, and measures.
- How to write DAX expressions using temporal intelligence functions and iterator functions.





Our Method: Flow of Report Development



INTERVIEWS

ETL

MODEL

DAX

REPORT

COLLABORATION



Exercises



Simple calculated COLUMNS → 3.1



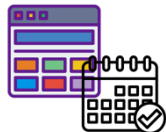
Simple calculated MEASURES → 3.2



More complex calculated COLUMNS → 3.3



More complex calculated MEASURES → 3.4



Time Intelligence MEASURES → 3.5



Introduction to filter context and row context

- Review of Week 2 and Exercise 2.4.
- Presentation: Introduction and examples related to DAX.



DAX

What if you need to analyze:

- Percentage growth of sales between periods of time
- Margin percentage for different:
 - Product categories
 - Customers
 - Date ranges
- Year-over-year growth compared to market trends

Data Analysis Expressions (DAX) is a library of functions and operators that can be combined to build formulas and expressions in Power BI, Analysis Services, and Power Pivot in Excel data models.



DAX

Calculated columns, Measures and calculated tables use DAX expressions.



Columns



Measures



Tables



Functions

Predefined formulas that perform calculations using specific values called arguments, in a particular order or structure. Can be group in:

- ✓ **Filters**
- ✓ **Information**
- ✓ **Logical**
- ✓ **Text**
- ✓ **Math and Trig**



Create a calculate column and use it in a report

- Demo Case 3.1
- A couple of columns are created to see the functionality of the columns.
- Use them in different visuals.
- Exercise Case 3.1
- Q&A



Demo 3.1



Calculated Columns



1 Click on "Data"

2 Select the table to add a new column

3 Click on "New Column"

4 Write the DAX formula to create the new column

1 Kind of purchase =
2 SWITCH(TRUE(),
3 Sales[SalesQuantity] < 20, "Retail",
4 "Wholesale")

SalesKey	DateKey	channelKey	StoreKey	ProductKey	UnitCost	UnitPrice	SalesQuantity	Kind of purchase
3203057	<i>martes, 1 de octubre de 2019</i>	1	108	363	321,44	S/.699	10	Retail
3190730	<i>sábado, 11 de mayo de 2019</i>	1	36	363	321,44	S/.699	10	Retail
3187598	<i>viernes, 28 de junio de 2019</i>	1	212	363	321,44	S/.699	10	Retail
3158687	<i>lunes, 8 de abril de 2019</i>	1	173	363	321,44	S/.699	10	Retail
3379409	<i>domingo, 14 de julio de 2019</i>	1	249	363	321,44	S/.699	10	Retail
3367665	<i>viernes, 25 de octubre de 2019</i>	1	194	363	321,44	S/.699	10	Retail
3346623	<i>miércoles, 10 de abril de 2019</i>	1	63	363	321,44	S/.699	10	Retail
3345574	<i>martes, 22 de octubre de 2019</i>	1	255	363	321,44	S/.699	10	Retail
3340546	<i>jueves, 9 de mayo de 2019</i>	1	101	363	321,44	S/.699	10	Retail
3334588	<i>martes, 29 de octubre de 2019</i>	1	58	363	321,44	S/.699	10	Retail
3311556	<i>viernes, 7 de junio de 2019</i>	1	34	363	321,44	S/.699	10	Retail
3298044	<i>viernes, 26 de abril de 2019</i>	1	291	363	321,44	S/.699	10	Retail



Calculated Columns

- A calculated column is a new column that is created by defining a calculation that transforms or combines two or more existing data elements.
- DAX expression defined for a **calculated column operates in the context of the current row** of the table it belongs to.
- **The results are stored in the model** like any other column.
- They **can be used to define a relationship**.
- Calculated columns are calculated during database processing and then stored in the model, so they **take up in-memory space**.



Let's start with the formulas like Excel

FILTER CONTEXT: The formula is evaluated row by row

- ❑ **SUM:** `TableName[ColumnNameA] + TableName[ColumnNameB]`
- ❑ **SUBTRACTION:** `TableName[ColumnNameA] - TableName[ColumnNameB]`
- ❑ **MULTIPLICATION:** `TableName[ColumnNameA] * TableName[ColumnNameB]`
- ❑ **DIVISION:** `TableName[ColumnNameA] / TableName[ColumnNameB]`
- ❑ **CONDITIONAL:** `IF (TableName[ColumnNameA] > 0; "xx" , "yy")`
- ❑ **BLANK:** `Blank()`
- ❑ **TODAY:** `Today()`
- ❑ **FIRST 2 LETTERS:** `LEFT (TableName[ColumnNameA], 2)`
- ❑ **CONDITIONS:** `&&` (and) `||` (or)



Exercise 3.1



Exercise 3.1: Create new columns in your model. Open the file *Exercise3.1.pbix*

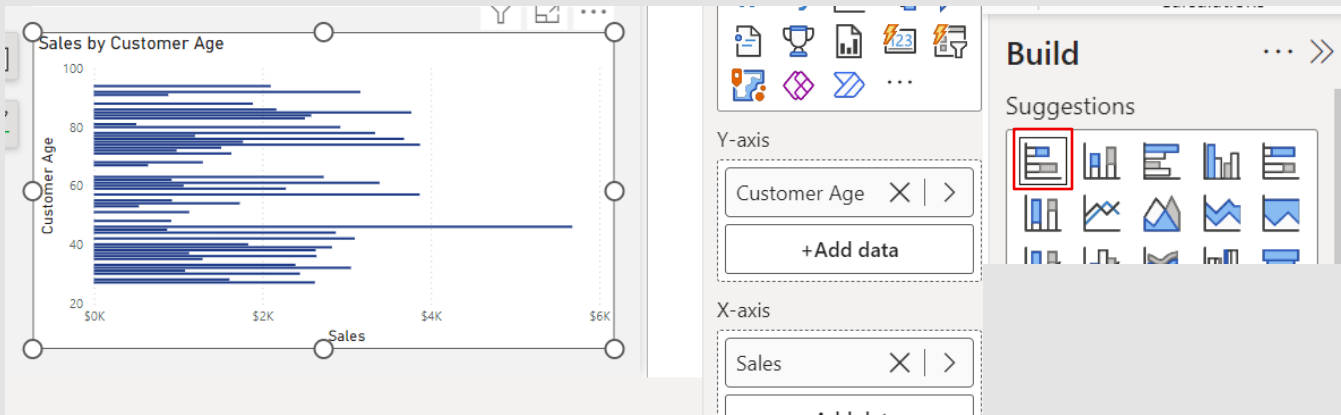
- 1 In the **Date table**, create a calculated column for the Day using the **FORMAT** function.

```
Day = FORMAT('Date'[Date], "dd")
```

- 2 In **Customer Table** create a new column to calculate the customer's age. Use the **DATEDIFF** function.

```
Customer Age = DATEDIFF(Customer[Date of birth], TODAY(), YEAR)
```

- 3 Go to **Customer page** and create a bar chart that displays sales segmented by customer age groups.





Creating basic measures (SUM and AVERAGE)

- Demo Case 3.2: Create some basic measures (SUM and AVERAGE) and use them in the report.
- See the difference with power BI's native aggregated domain functions.
- Exercise Case 3.2
- Q&A

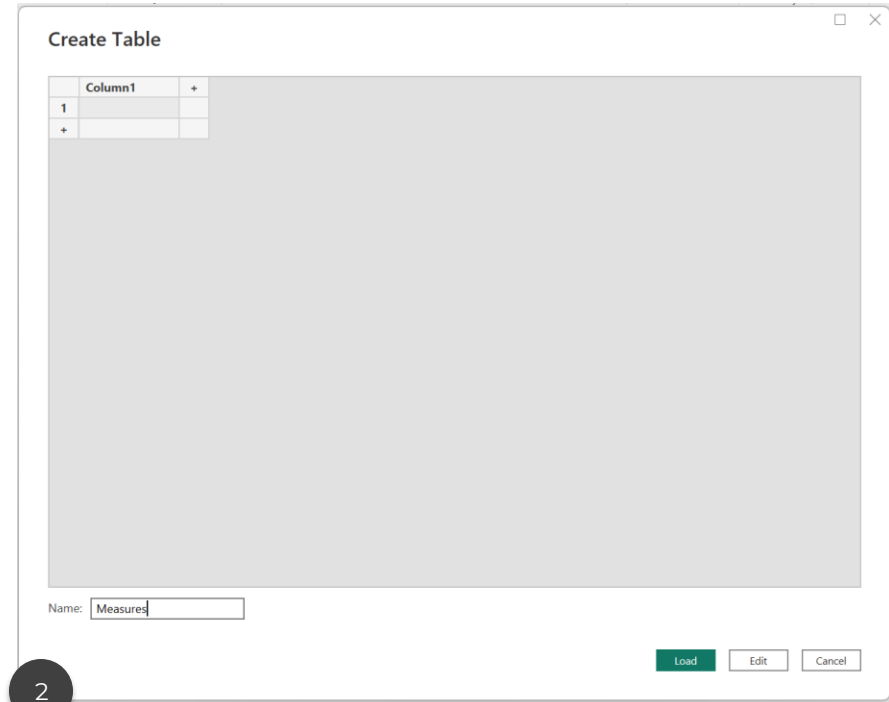
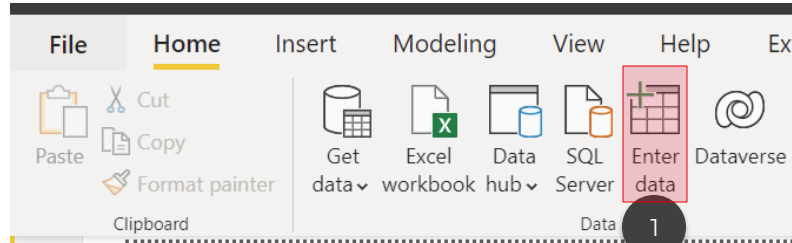


Demo 3.2





Create Measure table



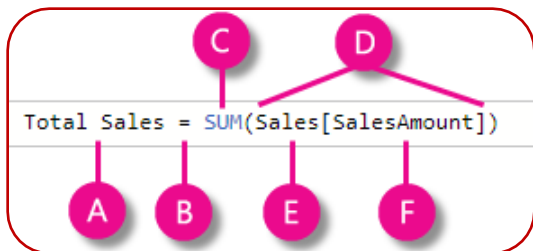


Calculated Measures

- ✓ Some of the most powerful data analysis solutions can be created through the use of measures.
- ✓ They are another way to add values from many rows in a table.
- ✓ They are found in Power BI as “New Measure”, unlike Excel 2013/2016 where they are also known as “Calculated Fields”.
- ✓ How do measurements help us? By performing calculations with our data as we interact with our reports, with the possibility of reusing them.



Syntax



- A. The name of the measure
- B. The operator (=) indicates the beginning of the formula
- C. SUM function with DAX
- D. Parentheses () surround an expression containing one or more arguments
- E. Referenced table
- F. The column it references [SalesAmount] in the sales table



Calculated Measures

- ✓ Calculated measures results are always changing in response to your interaction with your reports, enabling fast and dynamic exploration of ad-hoc data.
- ✓ A measure operates on aggregations of data defined by the current context, which depends on the filter applied to the report, such as slicers, rows, and column selection in a matrix, or axis and filters applied to a chart.

```
1 Sales = sum(Sales[Total Sales])
```

Measures



File Home Insert Modeling View Optimize Help External tools

Paste Cut Copy Format painter Clipboard

Get data Excel workbook Data SQL Server Enter data Datasource Recent sources

Transform Refresh data Queries

New visual

Insert

Text box More visuals

New measure Quick measure Calculations

Sensitivity Sensitivity

Publish Share

Filters

Search

Filters on this page

Add data fields here

Data

Search

Measure

More options

Customer

Date

File Home Insert Modeling View Optimize Help External tools

Paste Cut Copy Format painter Clipboard

Get data Excel workbook Data SQL Server Enter data Datasource Recent sources

Transform Refresh data Queries

New visual

Insert

Text box More visuals

New measure Quick measure Calculations

Sensitivity Sensitivity

Publish Share

Filters

Search

Filters on this page

Add data fields here

Data

Search

Measure

New measure

New column

New quick measure

Refresh data



Measures

File Home Insert Modeling View Optimize Help External tools Table tools **Measure tools**

Name Sales Format Whole number Data category Uncategorized

Home table Measure \$ % 0

Structure Formatting Properties Calculations

1 Sales = SUM(Sales[Total Sales])

3



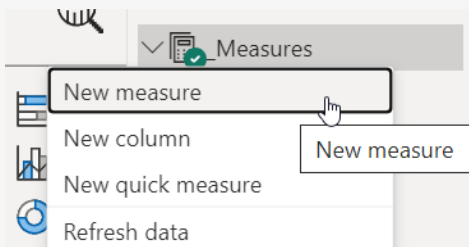
Exercise 3.2



Exercise 3.2: Create basic measures: Open the file *Exercise3.2.pbix*

1

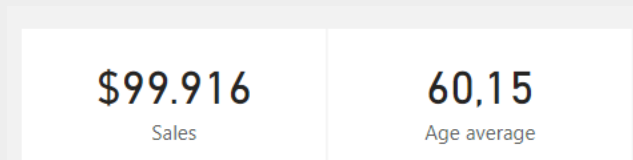
Create a measure that displays the average age of the customers.



```
. Age average = AVERAGE(Customer[Customer Age])
```

2

For testing purposes: Add the Average Age measure in a card visual (**Customer's Page**).





Context in DAX Formulas

Context enables you to perform dynamic analysis, in which the results of a formula can change to reflect the current row and any related data.

Understanding context and using context effectively are very important for building high-performing formulas, dynamic analyses, and for troubleshooting problems in formulas.

This course covers Row and Filter context.



Context in DAX Formulas

Calculated column – Row Context

- When you need the row by row value, to use that field to **filter or as row/column/axis in charts**.
- **Calculated during database processing** and then stored in the model, so they take up in-memory space.
- If you have created a calculated column, **the row context consists of the values in each individual row and values in columns that are related to the current row**.

Measures – Filter Context

- A measure operates on **aggregations of data defined by the current context, which depends on the filter applied to the report, such as slicers, rows, and column selection in a pivot table, or axis and filters applied to a chart**.
- **Evaluated at query time**. Not consume memory and disk space.



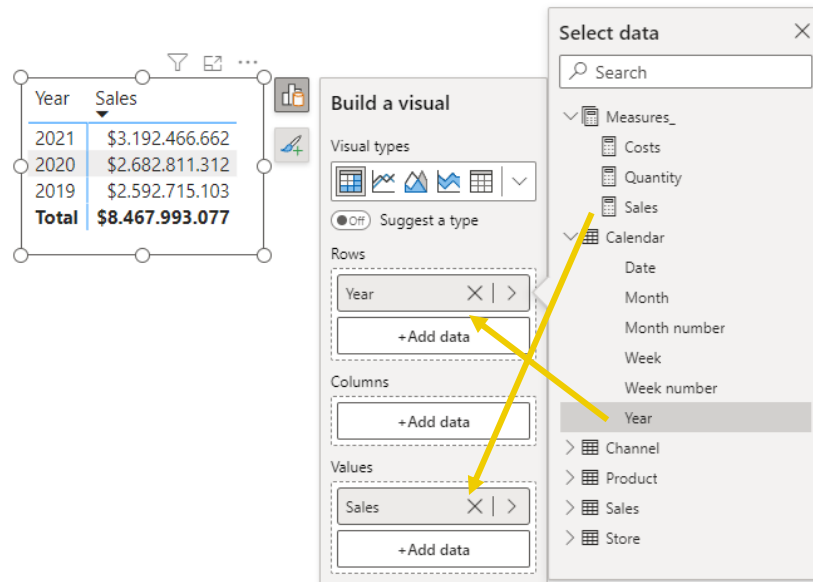
Measures or Calculated Column

Calculated Columns are created to use:

- in a **slicer**
- in **rows or columns in a pivot table** (as opposed to the values area)
- in **axes of a chart**
- as a **filter condition in a DAX query**

Define a **measure** whenever you want to:

- Display resulting calculation values that reflect user selections and see them in the **values area** of a pivot table or in the plot area of a chart.



Example:
Measures in Values
Columns in Rows and Columns
(Pivot Table)



What if...

We need to filter the report with the category of a shop based on its size (large, medium, small shop) we would create a **column** or a calculated **measure**?

Store size

☐ Big Store

☐ Medium Store

☐ Small Store





Poll

If we need to filter the report with the category of a shop based on its size (large, medium, small shop) we would create a **column** or a calculated **measure**?

- Calculated column
- Calculated measure





Adding others calculated columns

- Demo Case 3.3: Calculated columns using SWITCH, &&, ||
- Exercise Case 3.3
- Q&A



Demo 3.3





Adding others calculated columns

```
1 Storesize =  
2 SWITCH(TRUE(),  
3 Store[StoreType] <> "Store", BLANK(),  
4 Store[SellingAreaSize] > 600 && Store[EmployeeCount] > 30, "Big Store",  
5 Store[SellingAreaSize] > 500, "Medium Store", "Small Store")
```

GeoLocation	Geometry	ETLLoadID	LoadDate	UpdateDate	Storesize	Co
00000010CCA54C1A8A4BE47406ABC7493188C5ECO	00000000010C6ABC7493188C5ECOCA54C1A8A4BE4740	1	viernes, 10 de julio de 2009	miércoles, 17 de junio de 2009	Small Store	
00000010CDB8AFD65F75445403FC6DCB584705240	00000000010C3FC6DCB584705240DB8AFD65F7544540	1	miércoles, 10 de diciembre de 2008	martes, 16 de junio de 2009	Big Store	
00000010CDC4603780BD44740D122DBF97E5A5DC0	00000000010CD122DBF97E5A5DC0DC4603780BD44740	1	viernes, 10 de julio de 2009	miércoles, 17 de junio de 2009	Medium Store	
00000010C24287E8CB9D343402FDD2406813D5AC0	00000000010C2FDD2406813D5AC024287E8CB9D34340	1	viernes, 10 de julio de 2009	miércoles, 17 de junio de 2009	Medium Store	
00000010C41F163CC5DAB4E40F6285C8FC2BD62C0	00000000010CF6285C8FC2BD62C041F163CC5DAB4E40	1	domingo, 10 de mayo de 2009	jueves, 18 de junio de 2009	Big Store	
00000010CCC5D4BC8077D4340287E8CB96B2153C0	00000000010C287E8CB96B2153C0CC5D4BC8077D4340	1	lunes, 6 de julio de 2009	martes, 16 de junio de 2009	Small Store	
00000010C787AA52C43344440068195438B2C5AC0	00000000010C068195438B2C5AC0787AA52C43344440	1	viernes, 10 de julio de 2009	miércoles, 17 de junio de 2009	Medium Store	
00000010C8638D6C56D344440B0726891ED2C5AC0	E6100000010CB0726891ED2C5AC08638D6C56D344440	1	domingo, 10 de mayo de 2009	martes, 16 de junio de 2009	Small Store	
00000010C5396218E75214640F5B9DA8AFD1956C0	00000000010CF5B9DA8AFD1956C05396218E75214640	1	viernes, 10 de julio de 2009	jueves, 18 de junio de 2009	Medium Store	
00000010C40A4DFBE0E34444065AA605452B755C0	00000000010C65AA605452B755C040A4DFBE0E344440	1	viernes, 10 de julio de 2009	miércoles, 17 de junio de 2009	Small Store	
00000010CEFC9C342AD714340A69BC420B04653C0	00000000010CA69BC420B04653C0EFC9C342AD714340	1	domingo, 10 de mayo de 2009	jueves, 18 de junio de 2009	Medium Store	
00000010CF0164850FC484440931804560E455AC0	00000000010C931804560E455AC0F0164850FC484440	1	domingo, 10 de mayo de 2009	miércoles, 17 de junio de 2009	Small Store	
00000010CBBB88D06F0AE4340DC4603780B9C52C0	00000000010CDC4603780B9C52C0BBB88D06F0AE4340	1	miércoles, 10 de diciembre de 2008	martes, 16 de junio de 2009	Medium Store	



Exercise 3.3



Exercise 3.3: Add others calculated columns. Open the file *Exercise3.3.pbix*

1

In **Customer table** use the SWITCH function to create a new calculated column for **Customer Age Groups** with the following conditions:

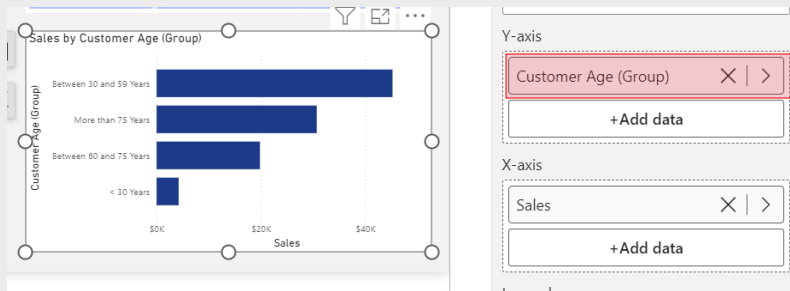
- Age < 30
- Age between 30 and 59
- Age between 60 and 75
- Age > 75

```
Customer Age (Group) =  
    SWITCH(TRUE(),  
        Customer[Customer Age] < 30 , "< 30 Years",  
        Customer[Customer Age] < 60, "Between 30 and 59 Years",  
        Customer[Customer Age] <= 75, "Between 60 and 75 Years",  
        "More than 75 Years")
```

	Customer Age	Customer Age (Group)
2	32	Between 30 and 59 Years
2	32	Between 30 and 59 Years
3	61	Between 60 and 75 Years
4	30	Between 30 and 59 Years
5	48	Between 30 and 59 Years
5	28	< 30 Years
7	84	More than 75 Years

2

Replace the **Customer Age** column with the new calculated **Customer Age (Group)** column in the sales by customer age bar chart.





What if...

We have to calculate the sales margin for our products or channels, would we use a **column** or a calculated **measure**?

% margin	ProductCategoryName
135,35 %	Audio
151,23 %	Cameras and camcorders
126,58 %	Cell phones
132,88 %	Computers
156,82 %	Music, Movies and Audio Books
121,48 %	TV and Video
135,98 %	

% margin	BrandName
138,47 %	Adventure Works
129,73 %	Contoso
135,98 %	

% margin	ChannelName
136,30 %	Catalog
135,97 %	Online
136,14 %	Reseller
135,90 %	Store
135,98 %	





Poll

If we have to calculate the sales margin for our products or channels, would we use a **column** or a calculated **measure**?

- Calculated column
- Calculated measure





Creating more complex measures using DIVIDE

- Demo Case 3.4: delving deeper into the dax language by creating a measure.

Change the data type to view as a percentage

- Exercise Case 3.4
- Q&A



Creating more complex measures using DIVIDE

1 % margin = `DIVIDE` ([Sales] - [Costs] , [Costs])

Year	% margin
2019	133,67 %
2020	136,90 %
2021	137,11 %
Total	135,98 %



Demo 3.4



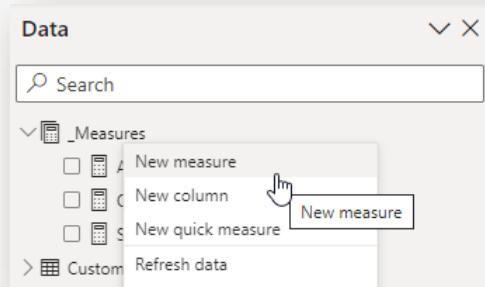


Exercise 3.4



Exercise 3.4: Create advanced measures: DIVIDE. Open the file *Exercise3.4.pbix*

- 1 In the **report view**, navigate to the **measures table** and create a new measure named **Costs**:



```
Costs = sum(Sales[Total Cost])
```

- 2 Create the following calculated measures:

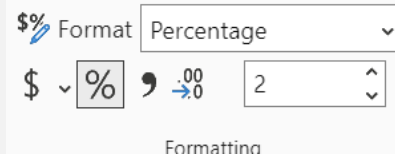
1. **Margin** as the subtraction between **Sales** measure and **Costs** measure.

```
Margin = [Sales] - [Costs]
```

2. **Margin %** as the division of the **Margin** measure (2.1 step) over **Sales** measure. To do so, use DIVIDE function.

```
Margin % = DIVIDE([Margin],[Sales])
```

3. To this last measure change the format to percentage.





Introduction to CALCULATE and TIME INTELLIGENCE

- Demo Case 3.5: presentation of the calculate function, as orchestrator of other functions and modifier of context filters.
- Exercise Case 3.5



Demo 3.5





Calculate

It is the only function in DAX that can change the filter context.

The syntax of calculate extremely simple. You invoke calculate with an expression as its first argument, and a set of filters starting from the second parameter onwards.

For example, the following measure computes the sales amount of Red products, avoiding any filter of the report in the column 'Product'[Color]

Red Sales :=

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

-  **Function**
-  **Measure**
-  **Filter**



Time Intelligence

Time intelligence functions support calculations to compare and aggregate data over time periods, supporting days, months, quarters, and years.

In order to use any time intelligence calculation, you need a well-formed date table in your model.

 <https://dax.guide/functions/time-intelligence/>



Time Intelligence

DateAdd

Moves the given set of dates by a specified interval.

Reference
your Calendar
[Date]
column

Intervals to
move in time.
Last month = -1

Interval can be
day, month,
quarter or year

DATEADD (<Dates>, <NumberOfIntervals>, <Interval>)

Sales LY = **CALCULATE ([Sales], DATEADD ('Calendar'[Date], -1, YEAR))**

CALCULATE
evaluates the
expression modify
by filters

Year	Sales	Sales LY
2019	\$2.592.701.334	
2020	\$2.682.791.755	\$2.592.701.334
2021	\$3.192.445.225	\$2.682.791.755



Time Intelligence

TOTALYTD – TOTALMTD

Evaluates the specified expression over the interval which begins on the first day of the year and ends with the last date in the specified date column after applying specified filters.

```
1 Importe YTD = TOTALYTD([Importe], Calendario[Date])
```

Year	Month	Sales	Sales YTD
2019	Jan	\$187.605.616	\$187.605.616
2019	Feb	\$184.194.371	\$371.799.987
2019	Mar	\$184.284.916	\$556.084.903
2019	Apr	\$213.802.001	\$769.886.904
2019	May	\$235.344.044	\$1.005.230.948
2019	Jun	\$228.848.412	\$1.234.079.360
2019	Jul	\$229.632.802	\$1.463.712.162
2019	Aug	\$224.810.067	\$1.688.522.229
2019	Sep	\$217.612.628	\$1.906.134.857
2019	Oct	\$223.452.206	\$2.129.587.064
2019	Nov	\$228.114.444	\$2.357.701.508
2019	Dec	\$234.999.826	\$2.592.701.334
2020	Jan	\$187.719.290	\$187.719.290
2020	Feb	\$194.404.066	\$382.123.357
2020	Mar	\$186.556.385	\$568.679.742
2020	Apr	\$226.013.480	\$794.693.221
2020	May	\$221.368.197	\$1.016.061.418



Time Intelligence

Advanced functions

- ✓ Accumulated Month: `=DATESMTD(Date[Date]))`
- ✓ Accumulated Year: `=DATESYTD(Date[Date]))`
- ✓ Yesterday Sales: `=PREVIOUSDAY(Date[Date]))`
- ✓ Same Day Previous Year: `=SAMEPERIODLASTYEAR(Date[Date]))`
- ✓ Same Month Previous Year: `=ParallelPeriod(Date[Date];-12;Month))`
- ✓ Full Month: `=DatesBetween(Date[Date];StartOfMonth(Date[Date]);EndOfMonth(Date[Date]))`
- ✓ Previous Period: `=PREVIOUSMONTH(Date[Date]))`
- ✓ Last 30 Days: `=DATESINPERIOD(Date[Date];FIRSTDATE(Date[Date]);-30;Day))`



Exercise 3.5

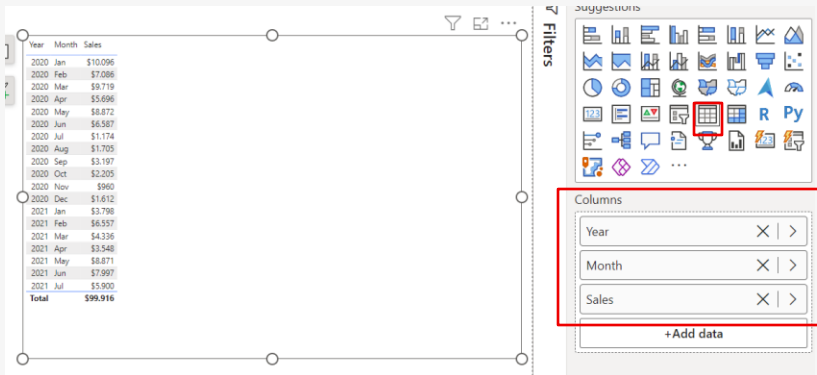


Exercise 3.5 Understand basic time intelligence functions. Open the file *Exercise3.5.pbix*

1

To test measures created and Time Intelligence functions create a new sheet and add the following:

- A table with the **year** and **month** as rows. Add the **Sales** as values



Year	Month	Sales
2020	Jan	\$10,096
2020	Feb	\$7,086
2020	Mar	\$9,719
2020	Apr	\$5,696
2020	May	\$8,872
2020	Jun	\$6,587
2020	Jul	\$1,174
2020	Aug	\$1,705
2020	Sep	\$3,197
2020	Oct	\$2,205
2020	Nov	\$960
2020	Dec	\$1,612
2021	Jan	\$3,798
2021	Feb	\$6,557
2021	Mar	\$4,336
2021	Apr	\$3,548
2021	May	\$8,871
2021	Jun	\$7,997
2021	Jul	\$5,900
Total		\$99,916

2

Create 2 new measures:


- Last month sales, using **DATEADD** function.

```
Last Month Sales = CALCULATE([Sales],DATEADD('Date'[Date],-1,MONTH))
```

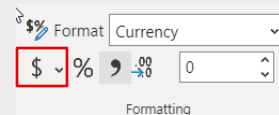
- Year to date sales, using **TOTALYTD** function.

```
Year to date Sales = TOTALYTD([Sales],'Date'[Date])
```

- Add both to the table and change the format to currency.



Year	Month	Sales	Last Month Sales	Year to date Sales
2020	Jan	\$10,096		\$10,096
2020	Feb	\$7,086	\$10,096	\$17,182
2020	Mar	\$9,719	\$7,086	\$26,901
2020	Apr	\$5,696	\$9,719	\$32,597
2020	May	\$8,872	\$5,696	\$41,469
2020	Jun	\$6,587	\$8,872	\$48,056
2020	Jul	\$1,174	\$6,587	\$49,230
2020	Aug	\$1,705	\$1,174	\$50,935
2020	Sep	\$3,197	\$1,705	\$54,132
2020	Oct	\$2,205	\$3,197	\$56,337
2020	Nov	\$960	\$2,205	\$57,297
2020	Dec	\$1,612	\$960	\$58,909
2021	Jan	\$3,798	\$1,612	\$3,798
2021	Feb	\$6,557	\$3,798	\$10,355
2021	Mar	\$4,336	\$6,557	\$14,691
2021	Apr	\$3,548	\$4,336	\$18,239
2021	May	\$8,871	\$3,548	\$27,110





Poll

True or False, removing a measure in Power BI also removes the related source data from the report?

- ☐ True
- ☐ False





Introduction to CALCULATED TABLES

- Demo Case 3.6: presentation of Calculated Tables.

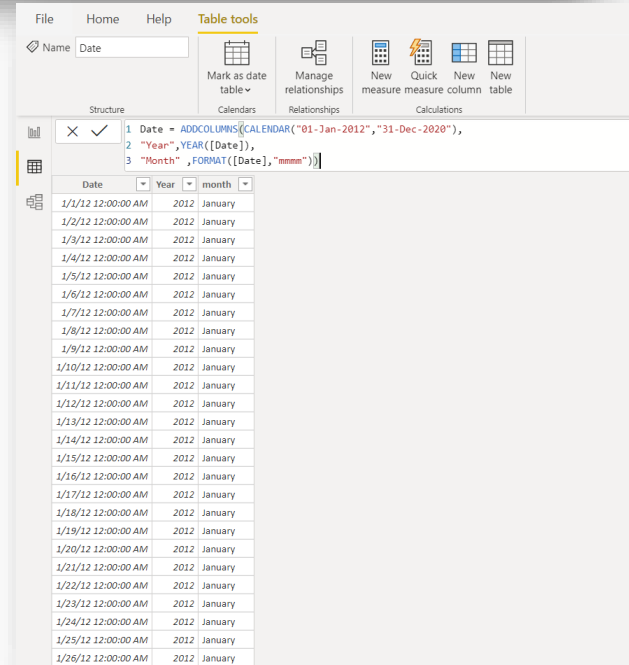
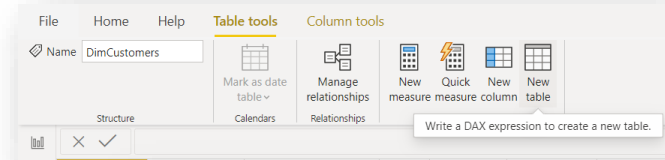


Calculated Tables

- ✓ With calculated tables, a new table can be added to the model.
- ✓ It is defined by a Dax formula.
- ✓ They are generally better for intermediate calculations and the data is stored as part of the model rather than being calculated on the fly or as part of a query.

Calendar =

```
ADDCOLUMNS(  
    CALENDARAUTO(),  
    "Year", year([Date]),  
    "Month num", Month([Date]),  
    "Month", FORMAT([Date], "mmm"),  
    "Day", FORMAT([Date], "dd"))
```





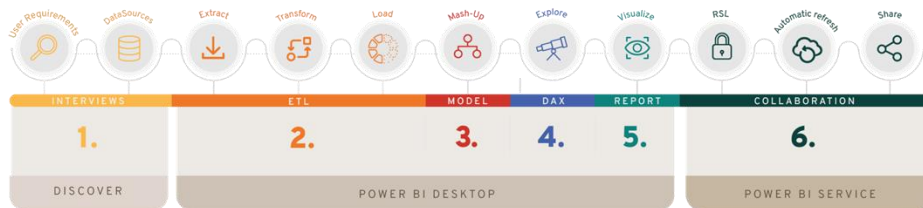
Recommendations for Next Steps

Part of the Power BI
Bootcamp Series

Advanced Learning Path:

- 1) Power BI Bootcamp: Building the Foundations for Power BI
- 2) Power Query: Data Connections and Transformations
- 3) Data Modeling for optimal structure & DAX for calculations
- 4) Visualizations for impactful reporting & Power BI Service

This Bootcamp





Stay Connected

Follow us on LinkedIn:





Nicolás Lagreste Zucchini



María Florencia Hourcouripé

Let's Collaborate: Interested in a personalized mentorship or consulting session? Contact us:

- o  nico@analyticmood.com
- o  flor@analyticmood.com

Visit Our Website:

- o www.analyticmood.com for resources, upcoming events, and blog articles.

Stay Tuned for More: Upcoming courses and events to take your skills to the next level!

analytic mood

Q&A



- Write your questions in the chat so that we can answer and discuss them together.



The background of the image is a gradient from red on the left to yellow on the right. A large, faint, stylized head of an animal, resembling a woodpecker or similar bird, is visible in the background, facing right. The head is composed of several overlapping circular and semi-circular shapes in shades of red and orange.

O'REILLY®