ENG

ventum-consulting.com | München | Essen | Wien | Foshan | Beijing | Shanghai

Power BI Desktop Advanced Workshop

Day 1

20/06/2023

Guest WiFi password: 133133_ventum_guests

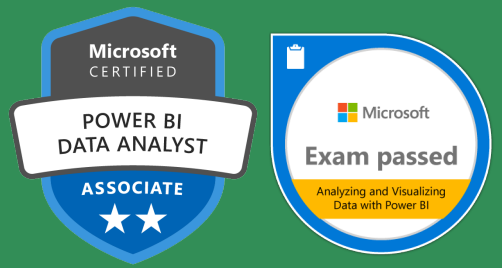
Please make sure you downloaded the Power BI Desktop application from the Company Portal in advance.

TRATON

ventum

About the trainer

Justin Schmidt



Agenda

01 Introduction to Power BI Desktop
Architecture & Environment

02 Data Modeling
Introduction Key Concepts

03 Introduction to DAX
Calculated Columns & Measures

04 Introduction to M
Queries, Columns, Joins Data Types

05 Advanced DAX
Time Intelligence & Iterator Functions

06 Power BI Service
Workspaces, Dashboards & RLS

Key Takeaways

- Basics of Power BI
- Navigation
- Key Concepts

- Basics of Data Modeling
- Data Modeling in Power BI

- Basics of DAX
- Create a calculated column
- Create a measure

Power BI applications

Power BI Service

Cloud-based
Software as a Service (SaaS)

Power BI Desktop

Desktop application for developing and publishing reports

Power BI Report Builder

Desktop application for paginated reports

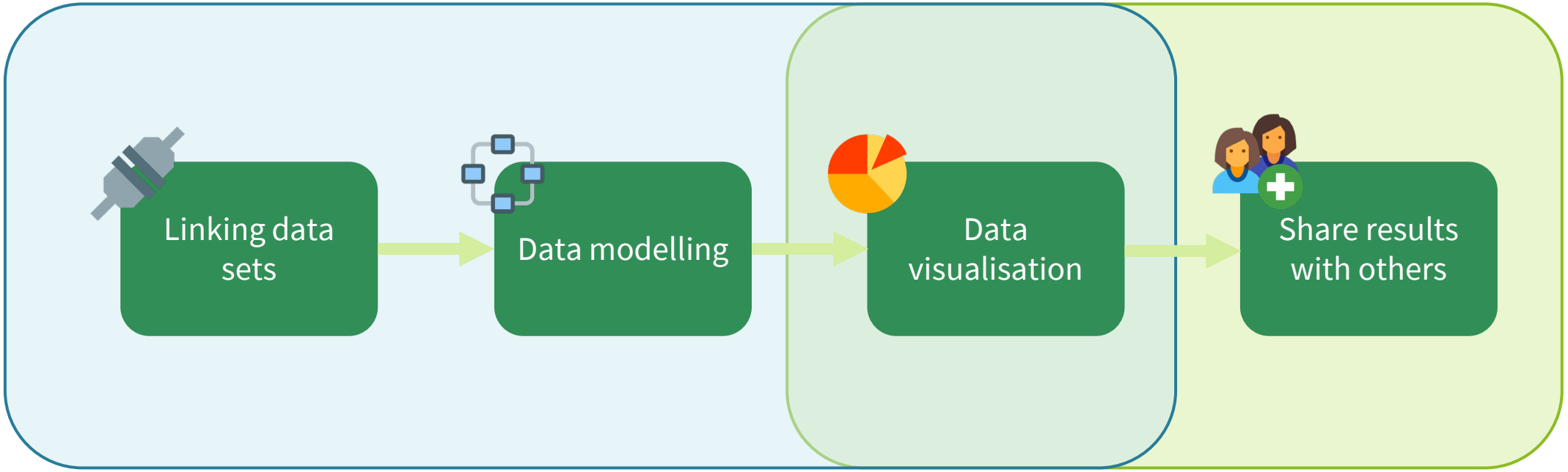
Power BI Mobile

Mobile apps

Power BI Report Server

On-premise Report Server

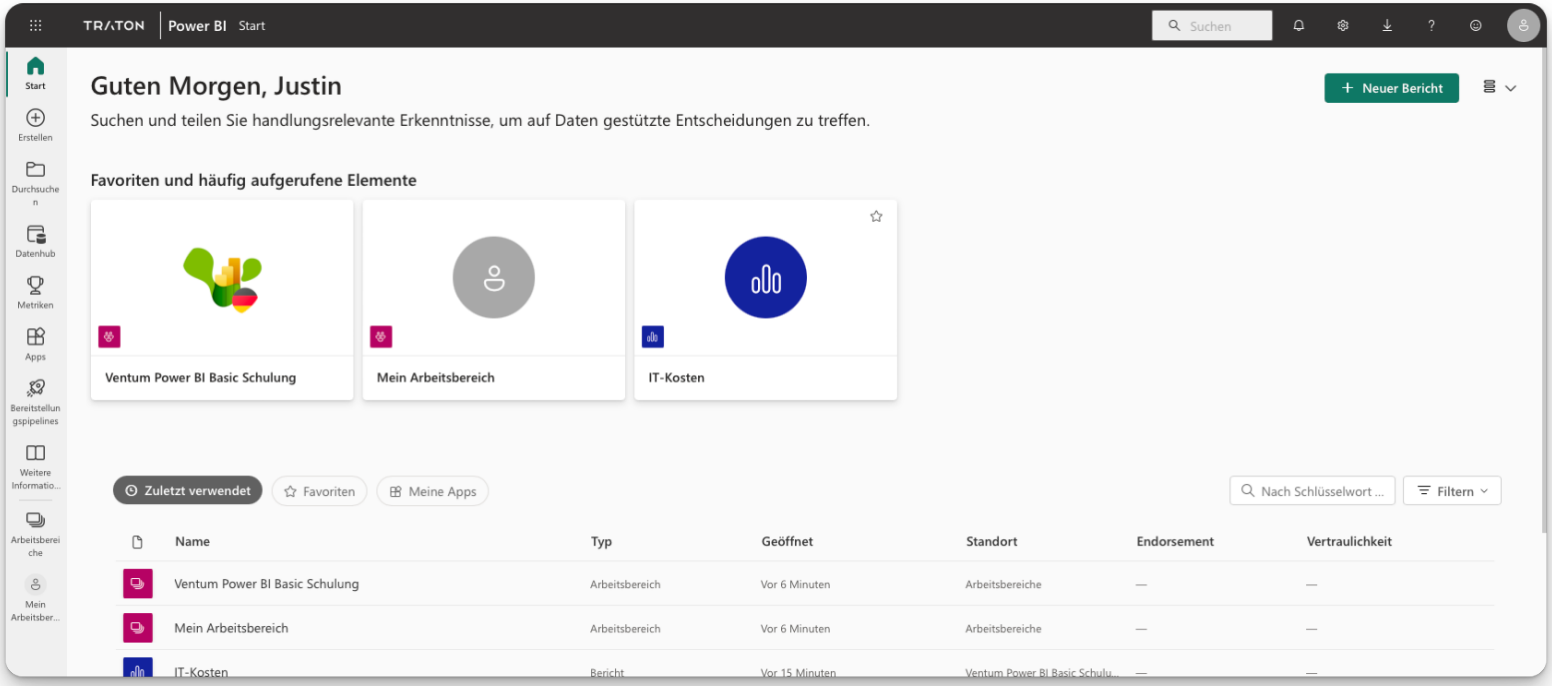
Typical BI workflow



Power BI Desktop
Expert application

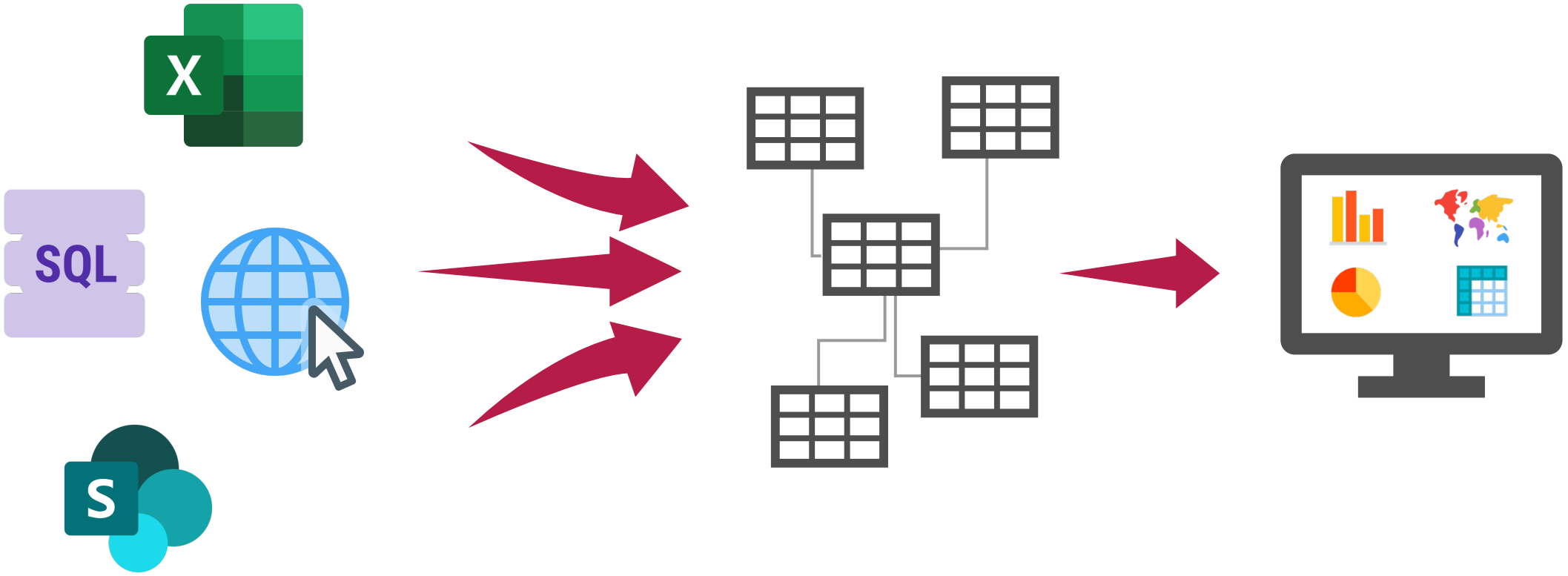
Power BI Service
for Citizen Developer

Power BI Service

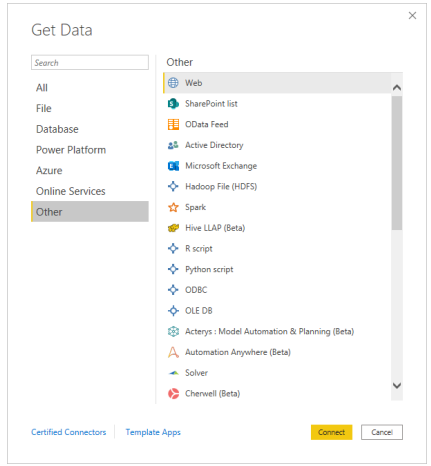


app.powerbi.com

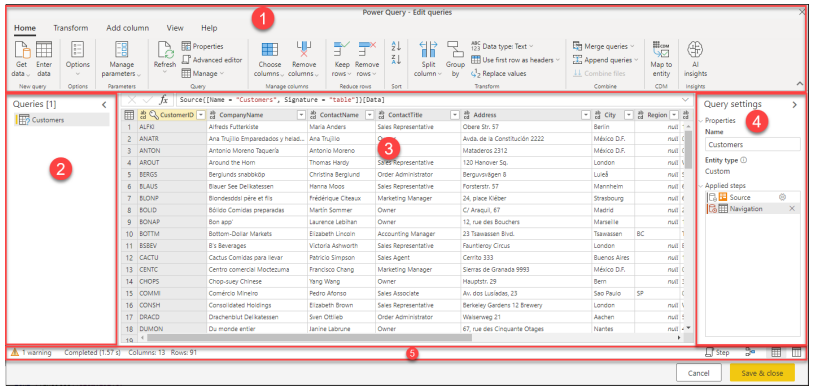
Power BI Desktop: Overview



Power BI Desktop: Step by step

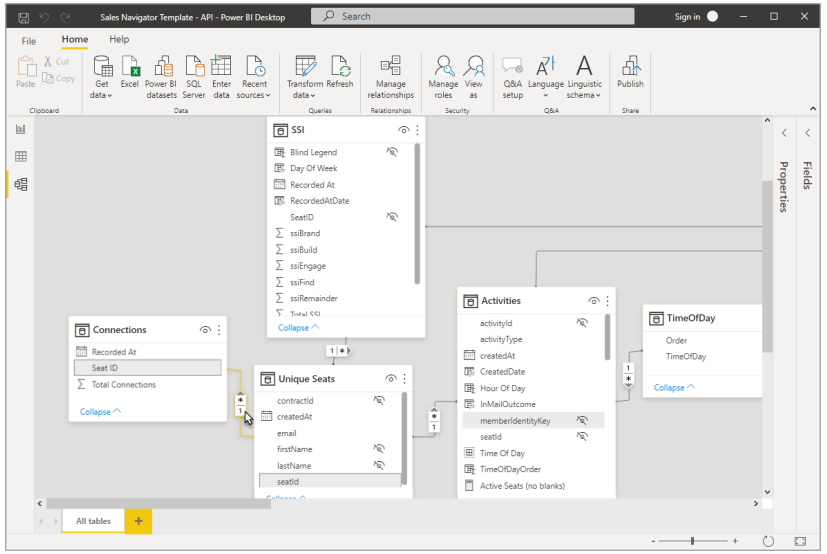


Connect data



PowerQuery Editor

→ Clean data with Power Query M Language



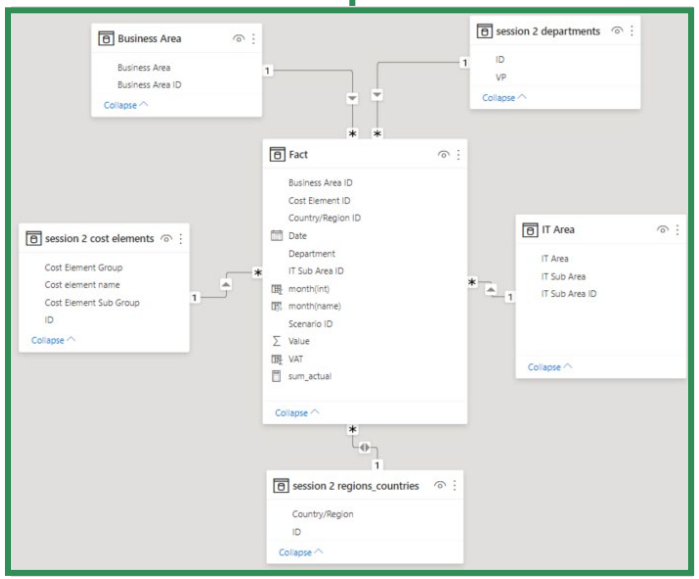
Model View

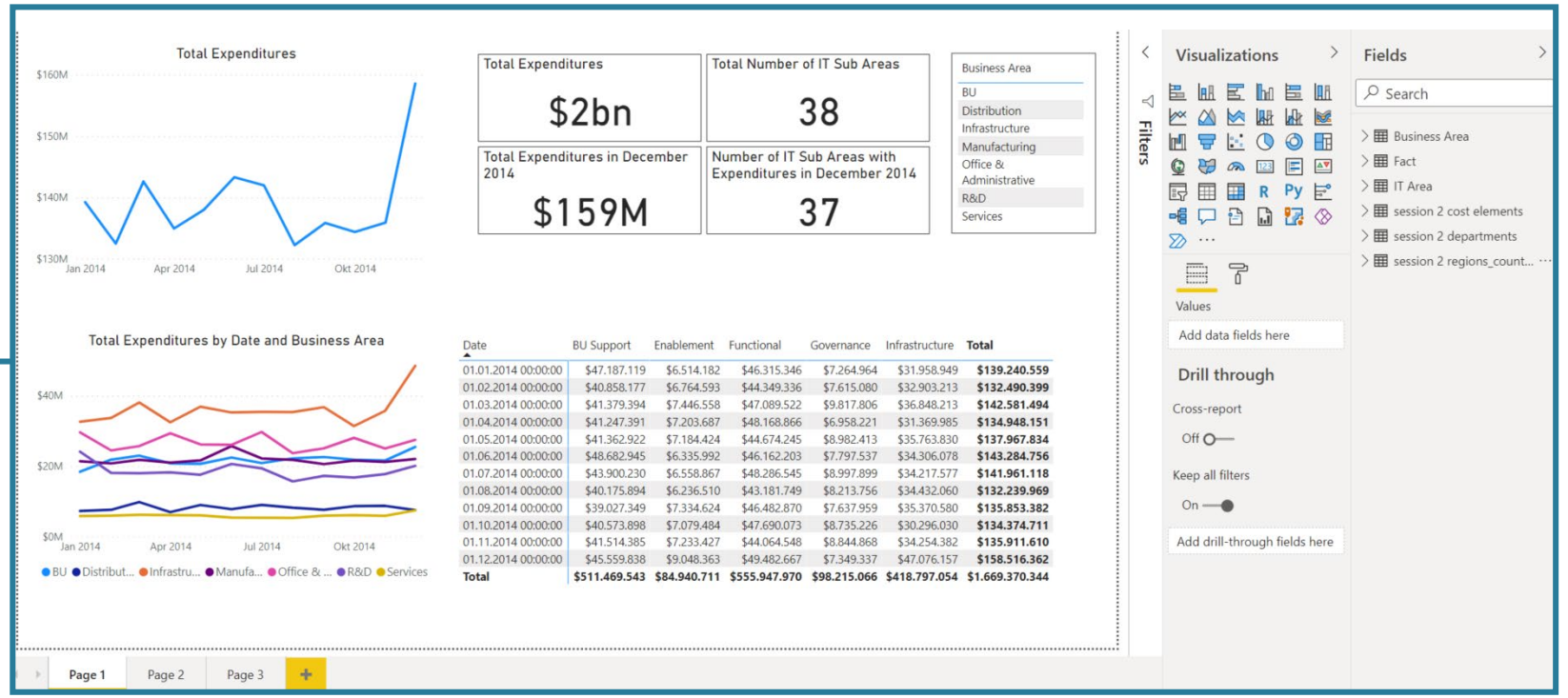
Views in Power BI Desktop

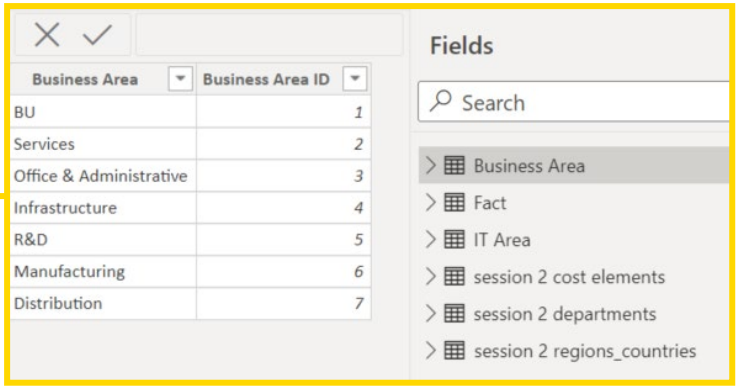
Report View

Data View

Model View







01

Demo I

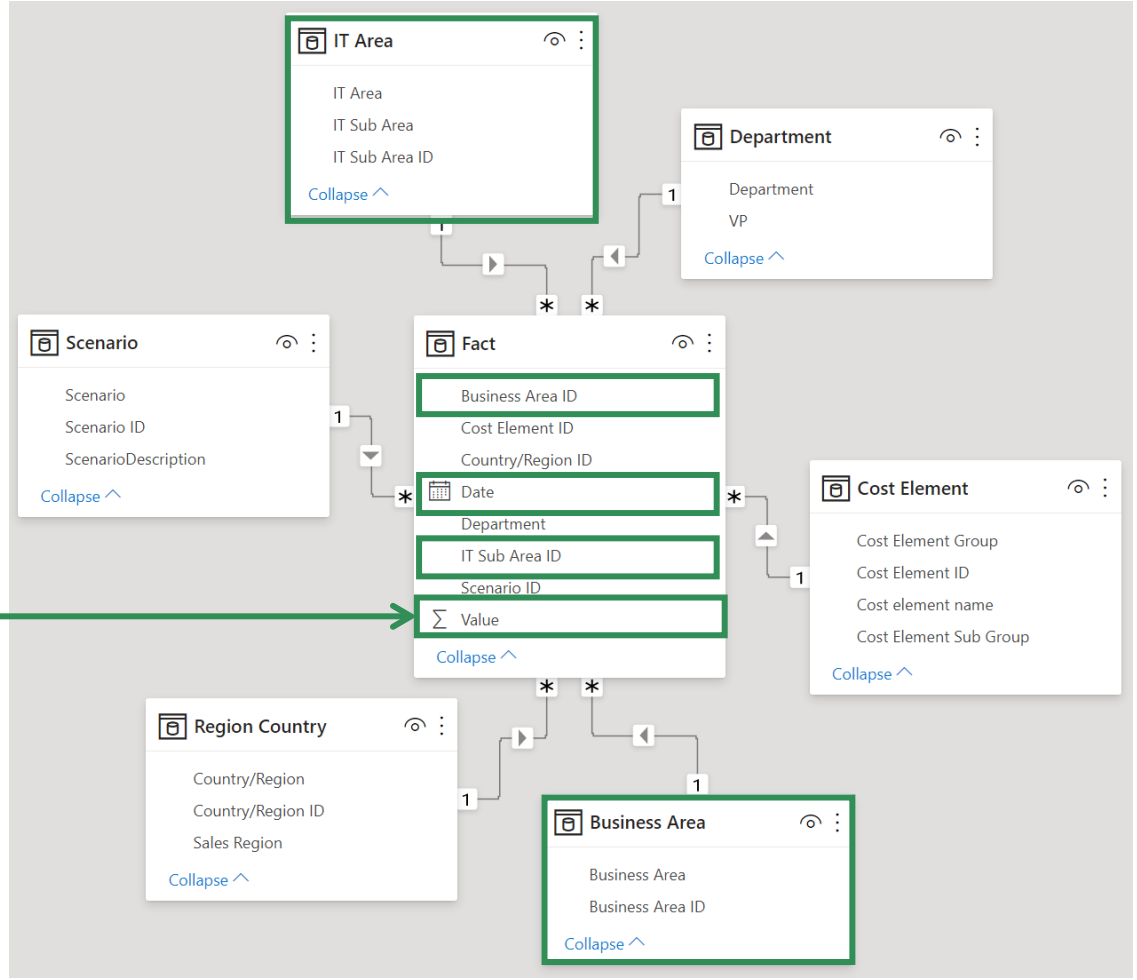
Demo Data

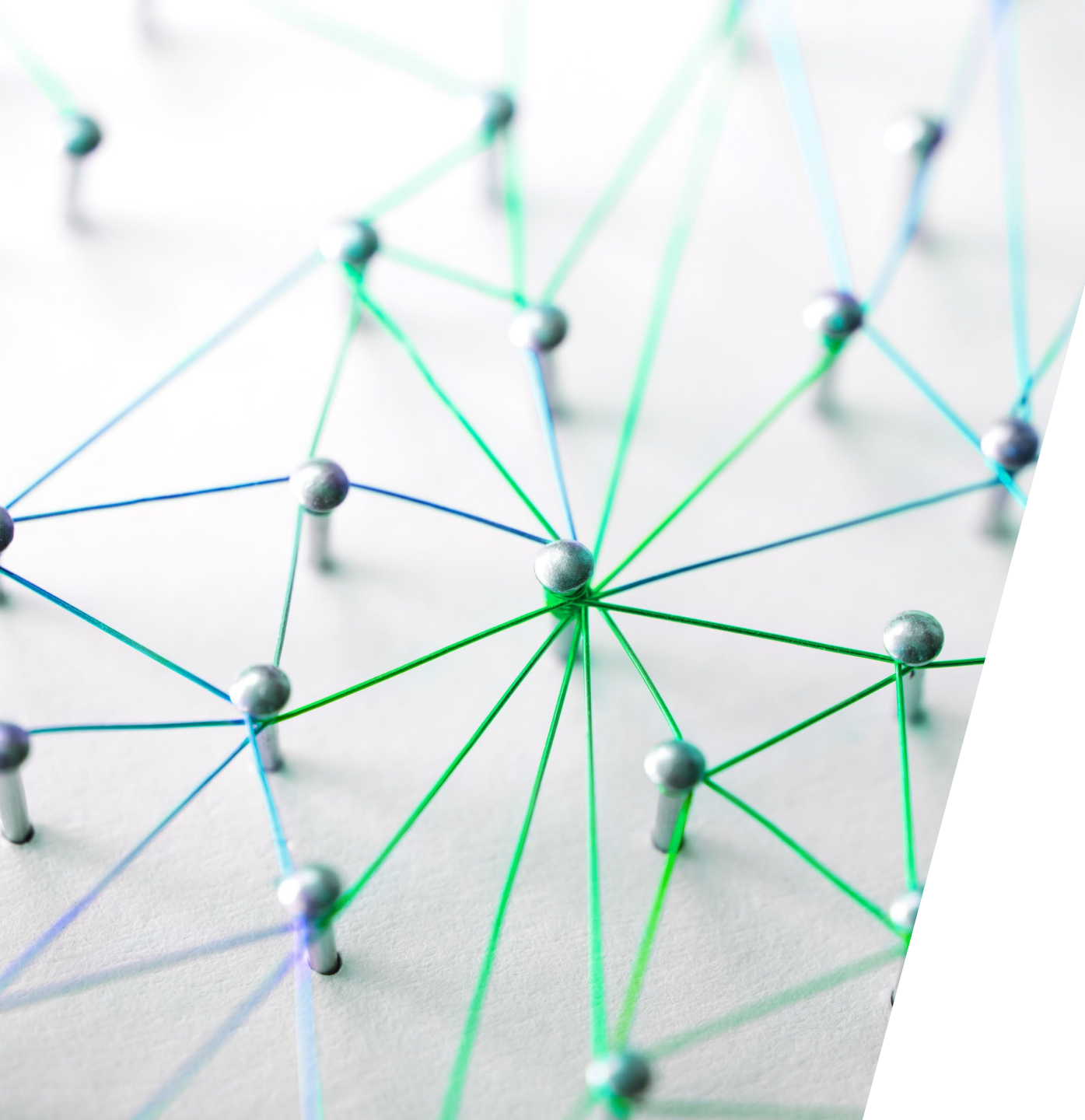
Expenditures/costs of a company from 2014 per

- Month
- IT-Sub-Area
- Business Area

Subsequent exercises:

- Department
- Cost Elements
- Region
- Tax rates
- Scenarios: Plan/Actual





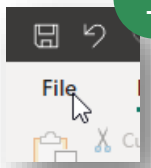
01

Hands-on Exercise I

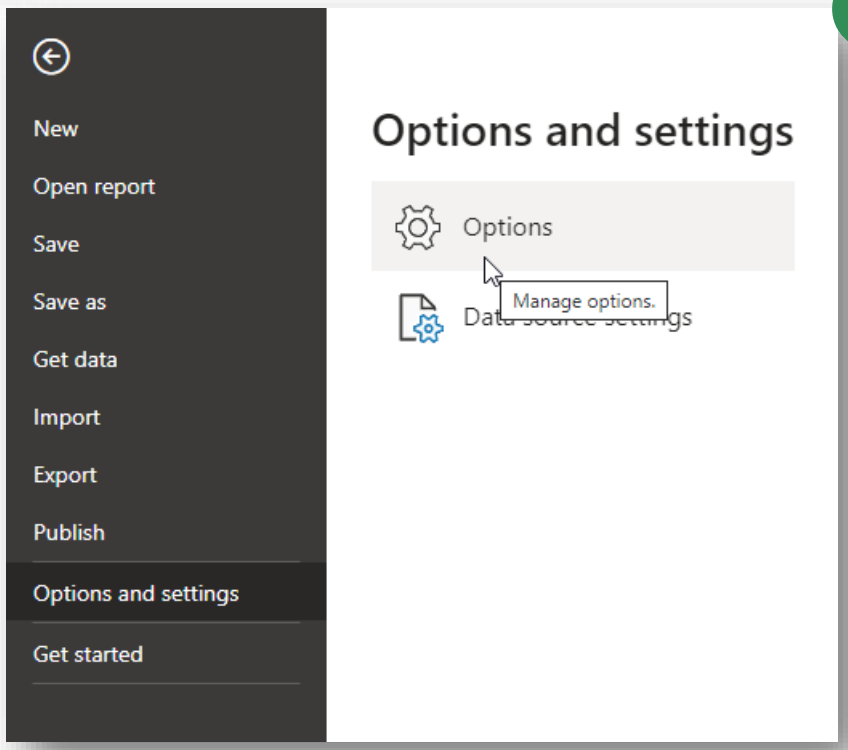
35 Minutes

Changing language preferences in Power BI Desktop

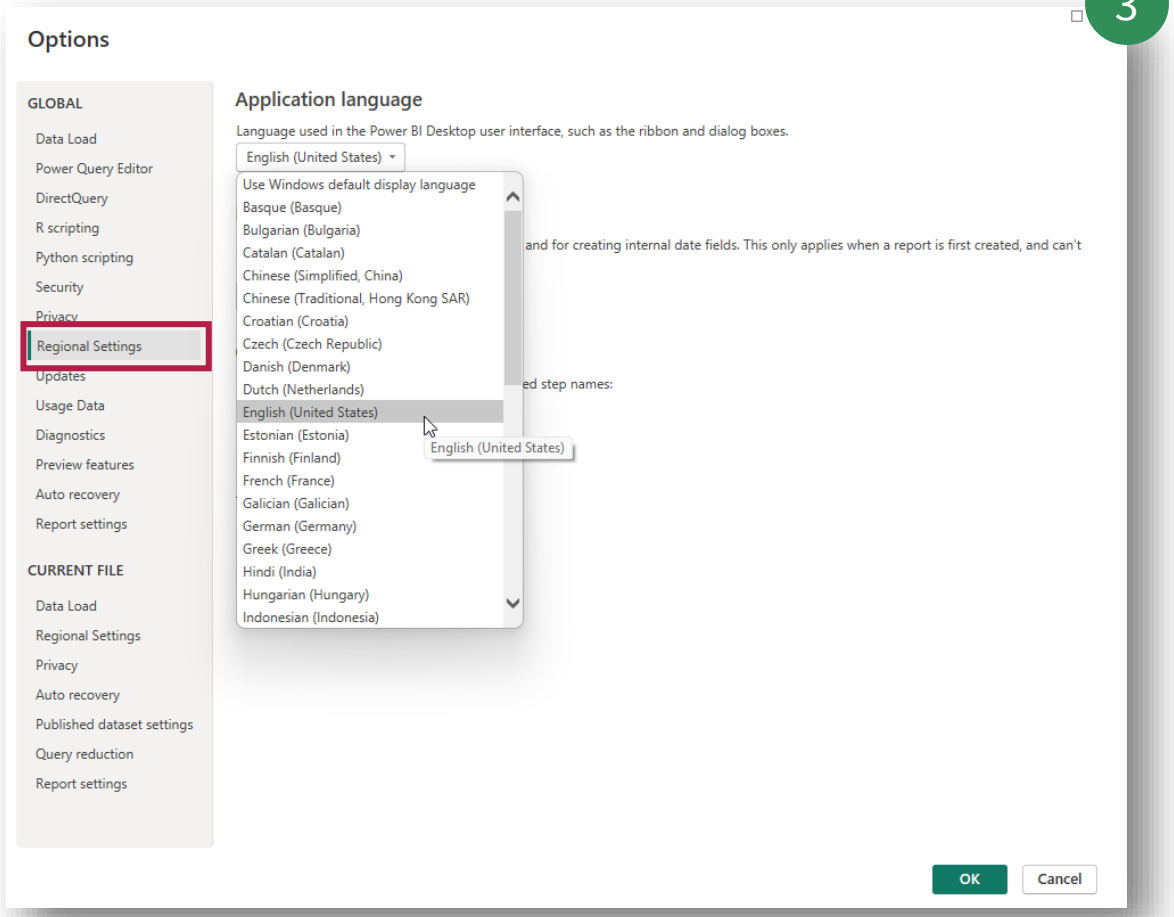
1

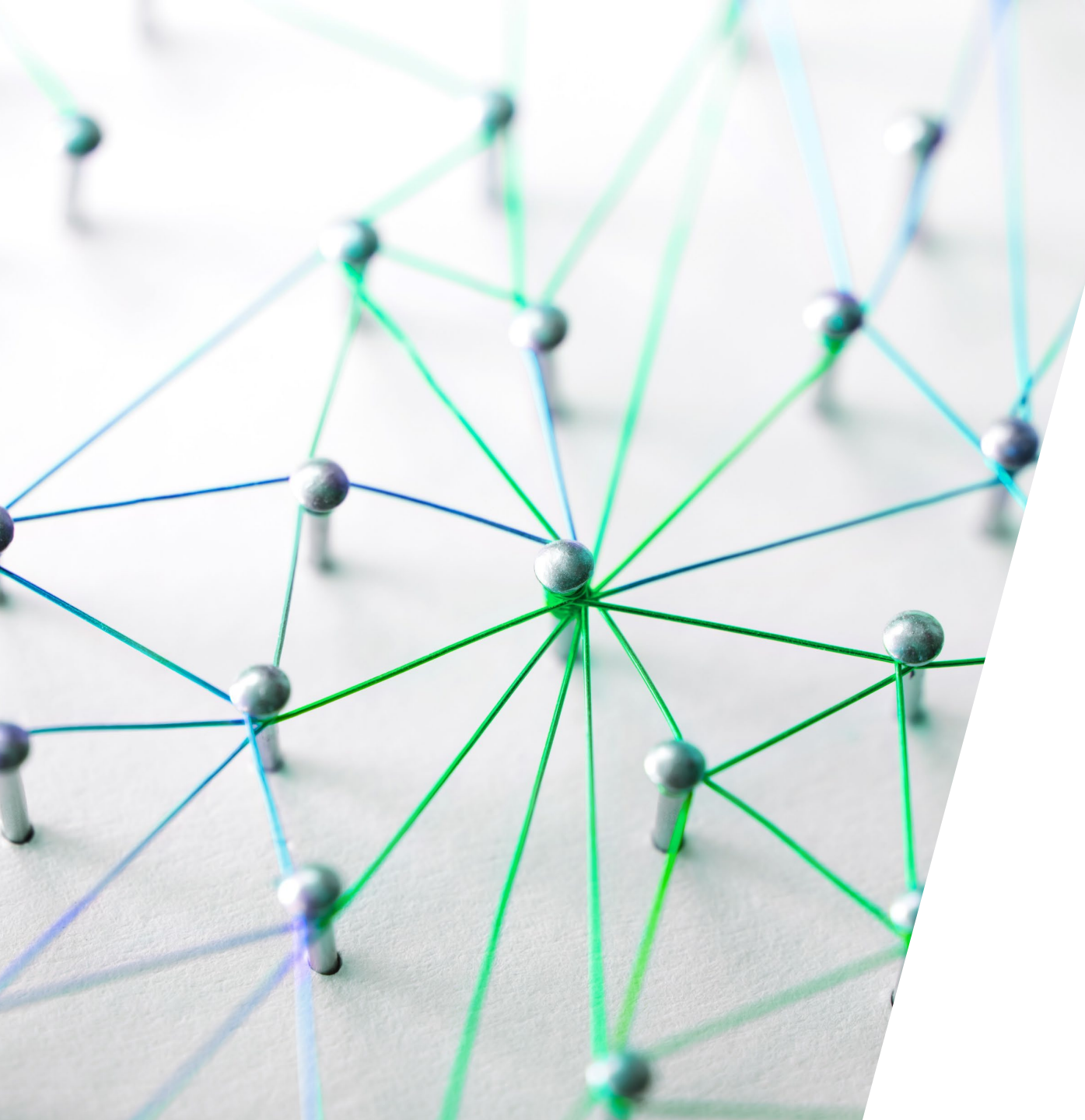


2



3





01

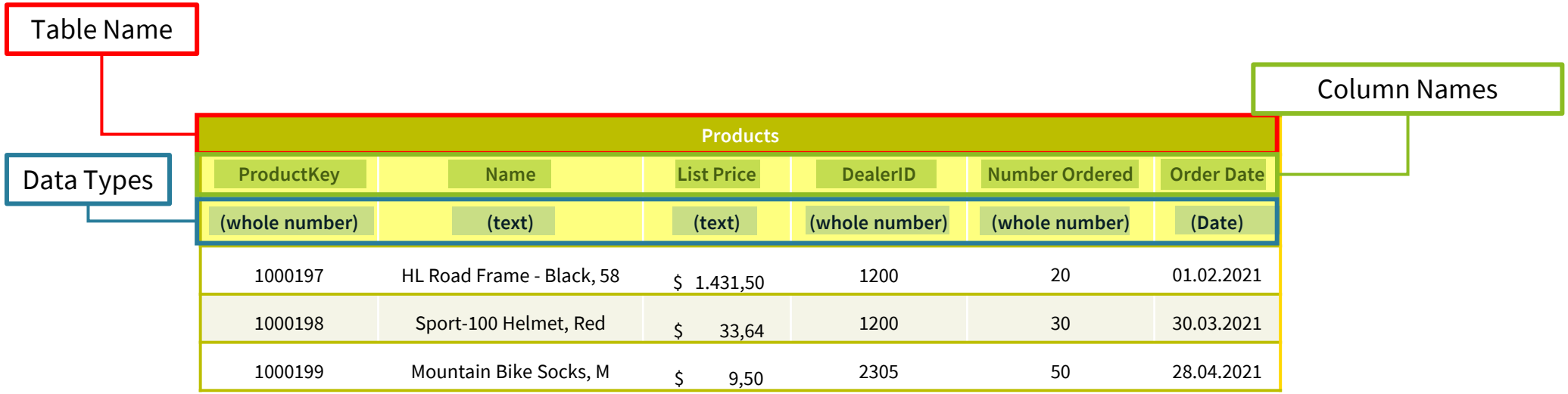
Hands-on Exercise I
Takeaways

- We can filter data in other visuals by selecting a data point in a visual
- Attributes of visuals can be manipulated in the „format“ section that opens when selecting the visual
- Data in visuals can be filtered via the filter panel when visual is selected

02

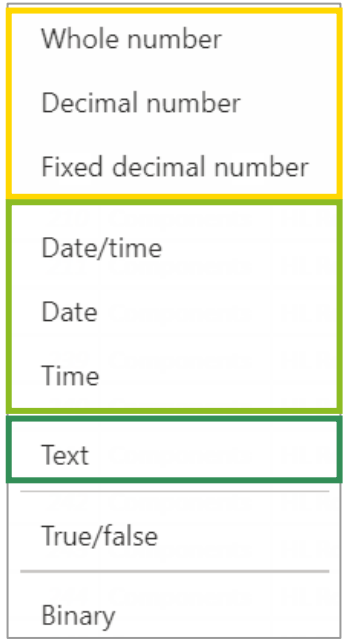
Data Modeling

Introduction to Data Modeling – Tables



Data Types

Products					
ProductKey (whole number)	Name (text)	List Price (text)	DealerID (whole number)	Number Ordered (whole number)	Order Date (Date)
1000197	HL Road Frame - Black, 58	$ 1.431,50	1200	20	01.02.2021
1000198	Sport-100 Helmet, Red	$ 33,64	1200	30	30.03.2021
1000199	Mountain Bike Socks, M	$ 9,50	2305	50	28.04.2021



Numerical Data Types

- Subject to mathematical operations
- Example: Calculate sum over „Number ordered“
- 1+1 = 2

Date and Time Data Types

- Subject to datetime operations
- Example: Add a week to a “Order Date”

Text data types

- Subject to string operations
- Example: combine “Name” and “List Price”
- “hello” & “world” = “hello world”
- “1” & “1” = “11”

Demo 2.1

2.1

Primary Keys & Foreign Keys

Primary Key

Foreign Key

Products

ProductKey (whole number)	Name (text)	List Price (text)	DealerID (whole number)	Number Ordered (whole number)	Order Date (Date)
1000197	HL Road Frame - Black, 58	$ 1.431,50	1200	20	01.02.2021
1000198	Sport-100 Helmet, Red	$ 33,64	1200	30	30.03.2021
1000199	Mountain Bike Socks, M	$ 9,50	2305	50	28.04.2021

A foreign key references a primary key of another table

A primary key uniquely identifies each row in a table.

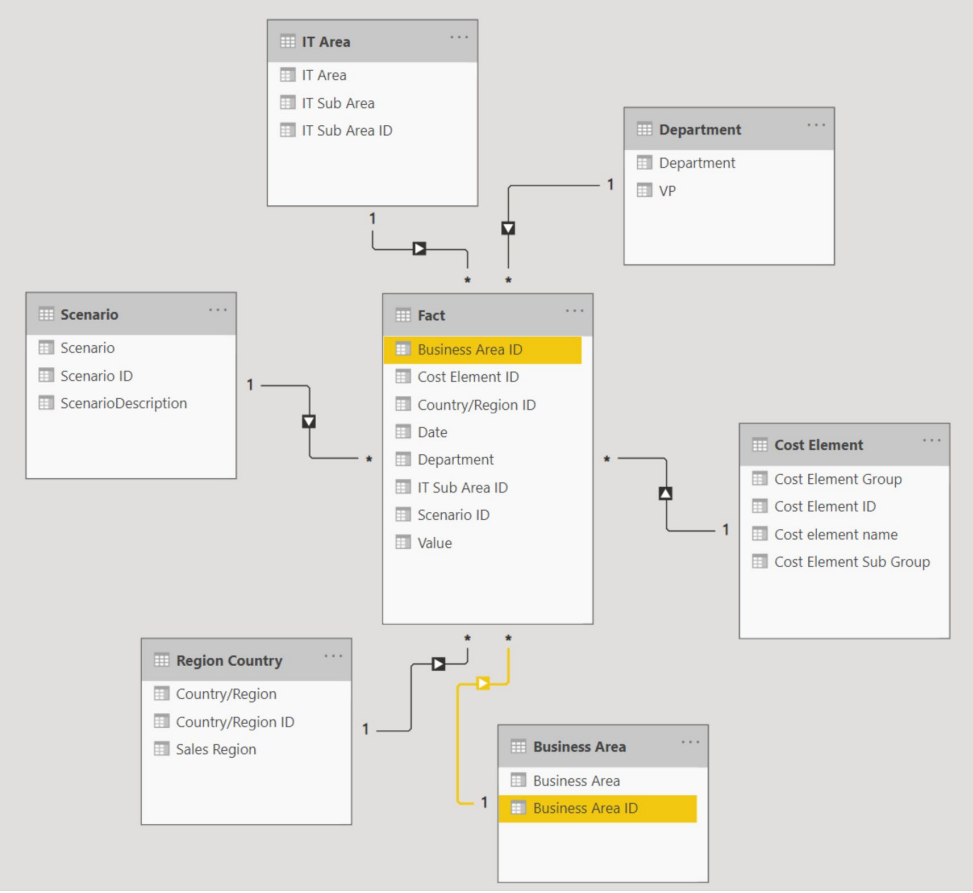
Dealers

Dealer_ID (integer)	Surname (text)	Forename (text)	Location_ID (Zahl)	Status (text)
1200	Roosevelt	Richard	65	active
1447	Nixon	Warren	2013	active

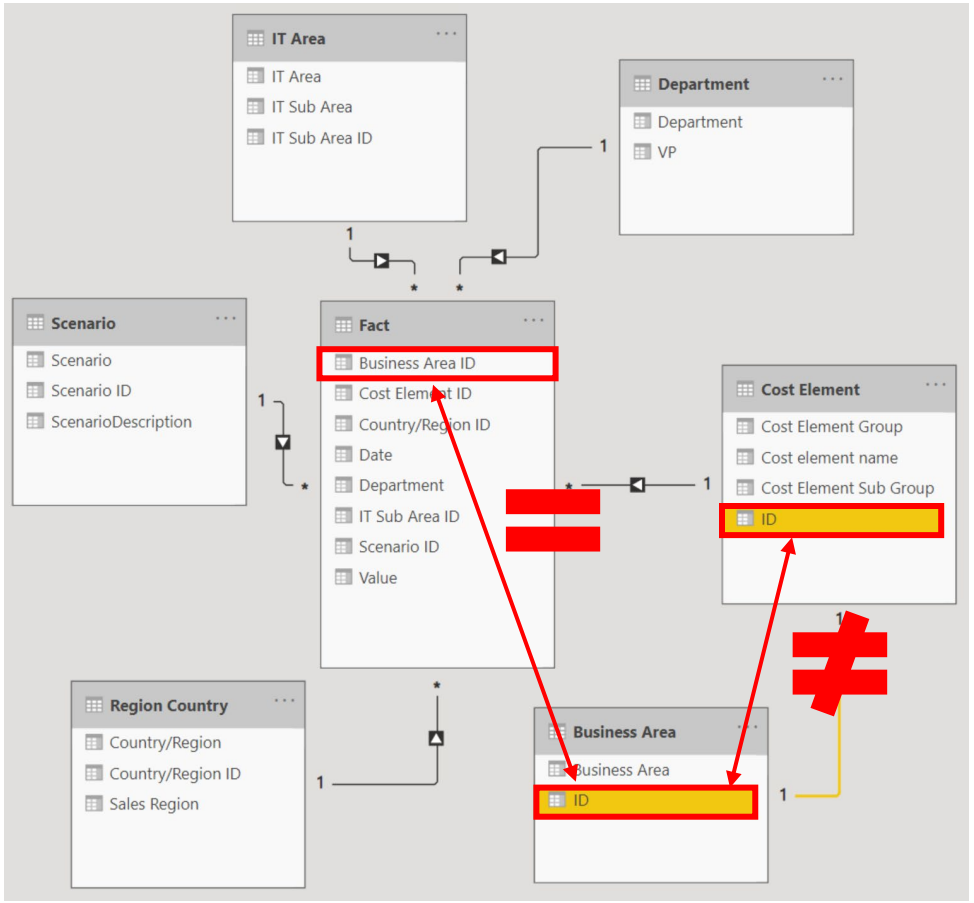
Dealer Richard Roosevelt sells product „HL Road Frame- Black, 58“

Data Modeling in Power BI

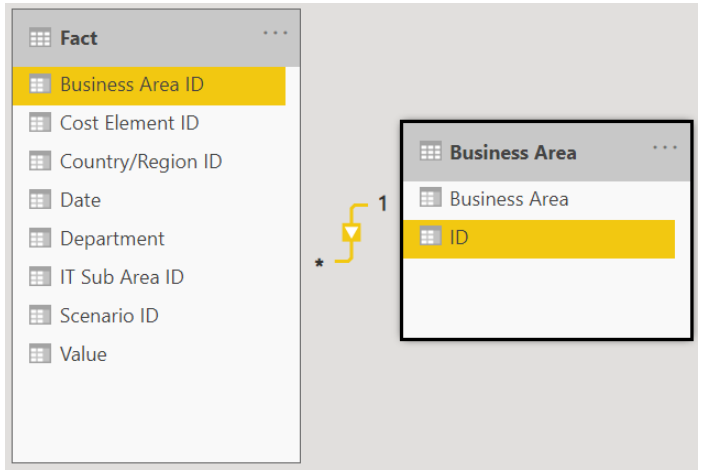
Power BI automatically recognizes foreign keys, if they have the same name and data type



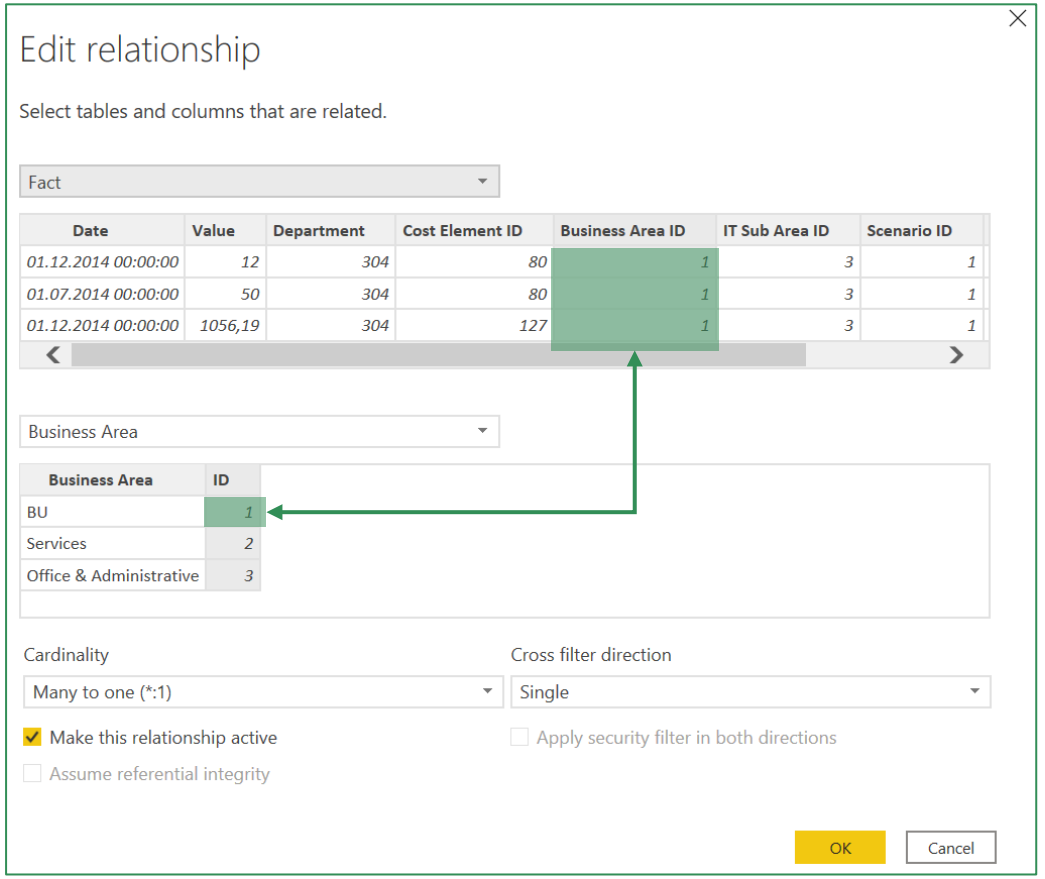
Caveat: if different columns have the same name, Power BI will still connect them



Cardinality



Double click on connection



„Many to one“ (*:1)→ All entries in the upper table connect to no more than one entry in the lower table.

Cross filter direction: Single → lower table can filter upper table

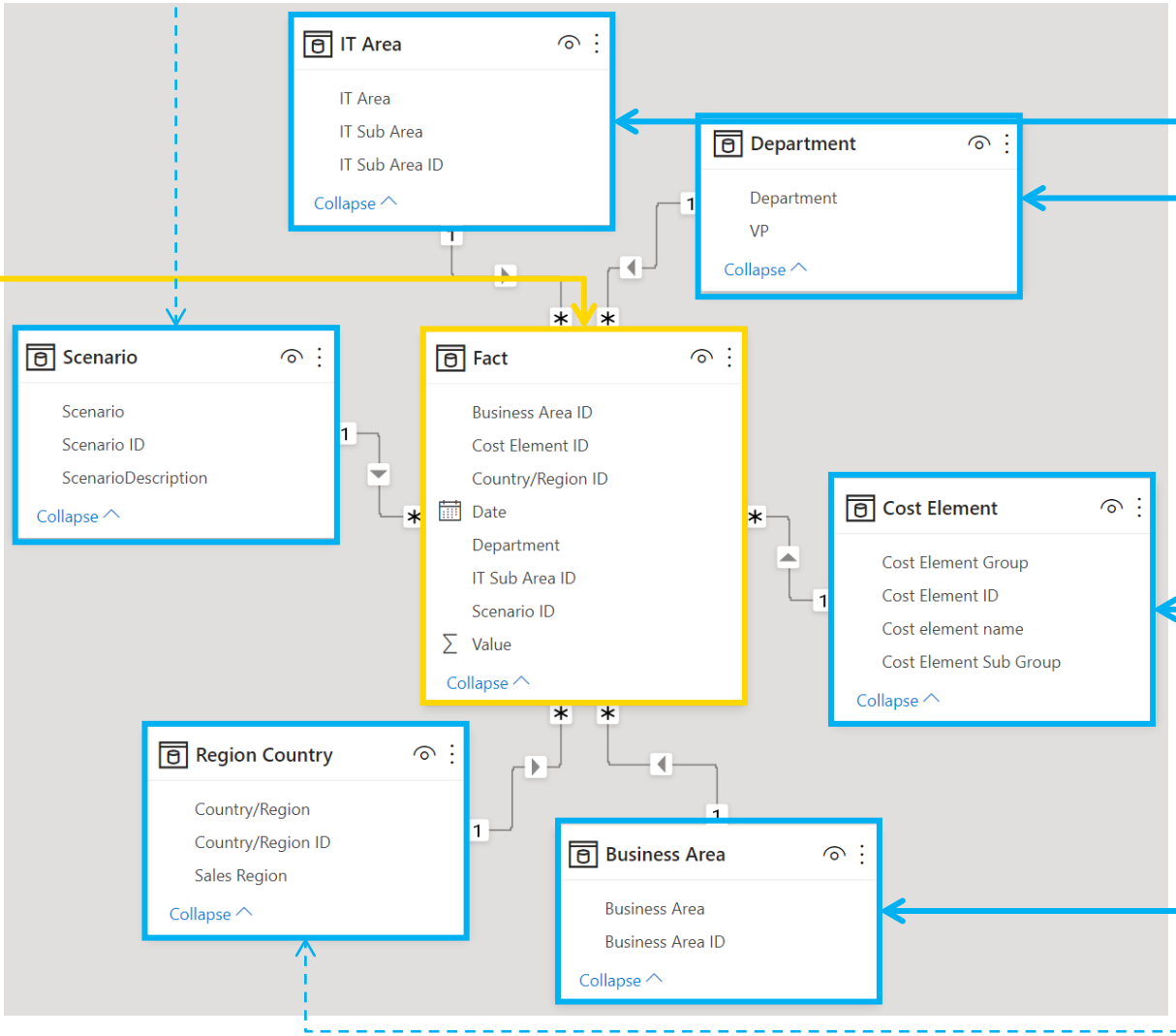
Demo 2.2

2.2

Datamodeling: Star Schema

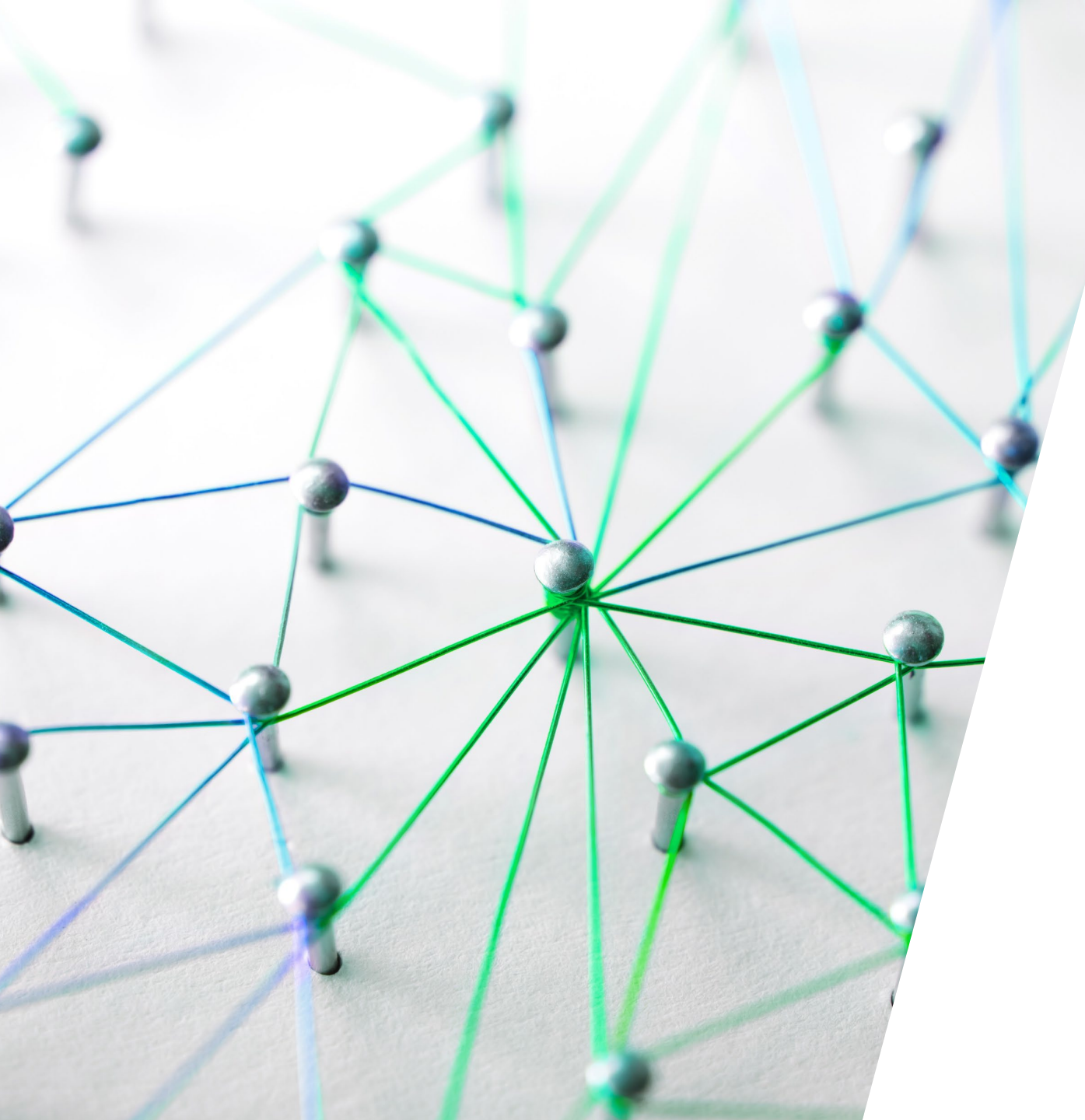
Fact Table

- Combine different sets of dimension key values and store according observations
- Dimension key columns determine ***dimensionality*** of a fact table
- Dimension key values determine ***granularity*** of a fact table



Dimension Table

- Describe business entities
- Key column + attributes
- Used for filtering and grouping



02

Hands-on Exercise 2

40 Minutes + 5 Minutes Break

02

Hands-on Exercise 2 Takeaways

- Source files can be easily replaced
- Automatically recognized relationships should always be verified
- If cross filter directions are set to „both“, slicers based on dimensions can filter each other
- The format of a column can be changed in the data section

03

Introduction to DAX

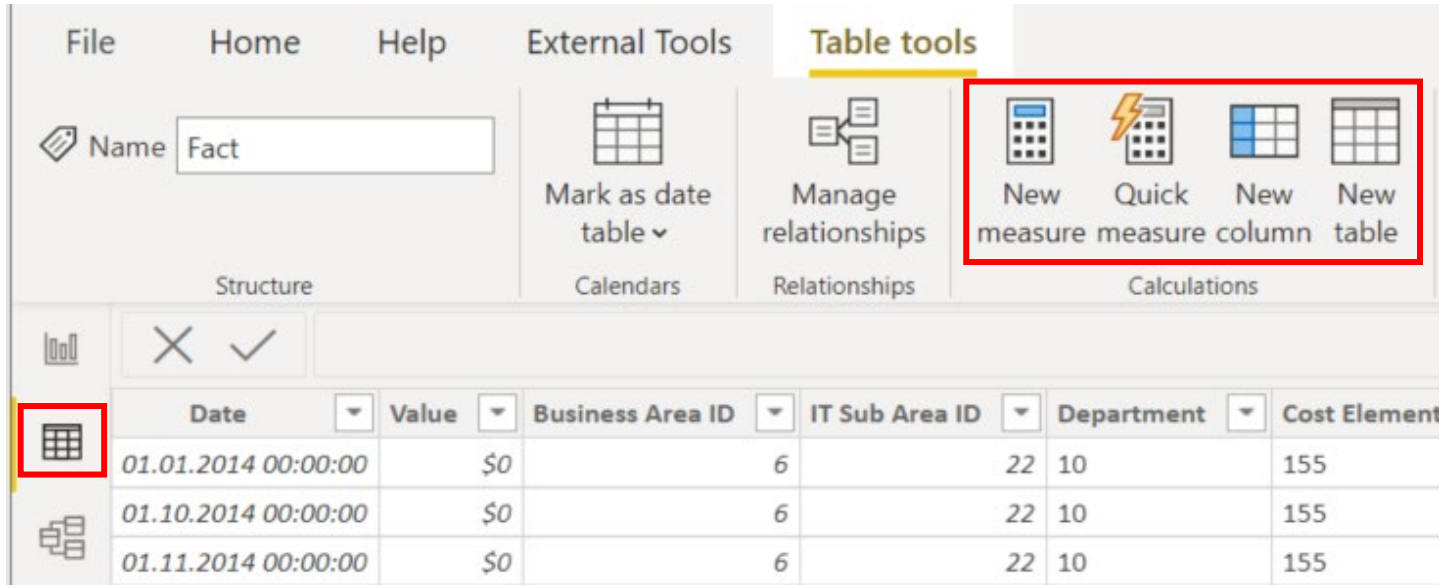
Data Analysis Expressions (DAX): Introduction

Used in

- Power BI
- Power Pivot
- SSAS tabular mode
- Azure Analysis Service

Can be used in Power BI for

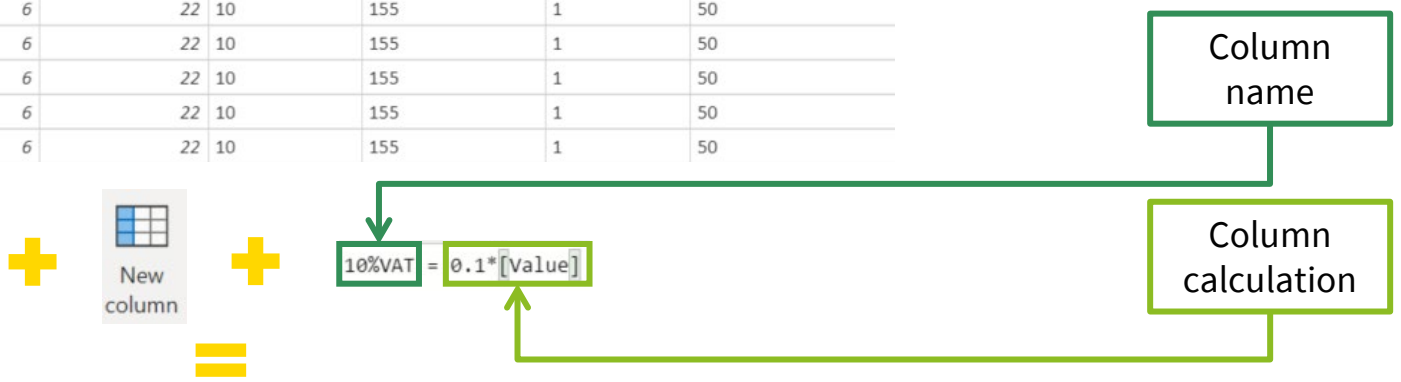
- Measures
- Calculated columns
- Calculated tables
- Row level security (RLS)



Calculated Columns

- Column added to an existing table
- Row based evaluation of expression
- Appear in fields list with table + sum icon:
- Reference on other columns via square brackets []
- Reference on columns in other tables via RELATED() function
- Standard mathematical operators: „+ - * /“

Date	Value	Business Area ID	IT Sub Area ID	Department	Cost Element ID	Scenario ID	Country/Region ID
01.04.2014	$35	6	22	10	155	1	50
01.05.2014	$298	6	22	10	155	1	50
01.06.2014	$12.600	6	22	10	155	1	50
01.07.2014	$12.600	6	22	10	155	1	50
01.08.2014	$12.600	6	22	10	155	1	50



Date	Value	Business Area ID	IT Sub Area ID	Department	Cost Element ID	Scenario ID	Country/Region ID	10%VAT
01.04.2014	$35	6	22	10	155	1	50	$4
01.05.2014	$298	6	22	10	155	1	50	$30
01.06.2014	$12.600	6	22	10	155	1	50	$1.260
01.07.2014	$12.600	6	22	10	155	1	50	$1.260
01.08.2014	$12.600	6	22	10	155	1	50	$1.260

03

Demo 3.1

Common Dax Functions

Aggregation Functions

- SUM
- MIN
- MAX
- AVERAGE
- COUNT
- DISTINCTCOUNT
- …

Date &Time Functions

- DATE
- YEAR
- MONTH
- QUARTER
- DAY
- WEEKDAY
- TODAY
- …

Text Functions

- CONCATENATE
- FORMAT
- LEN
- LEFT
- RIGHT
- MID
- REPLACE
- SUBSTITUTE
- TRIM
- UPPER
- …

Math Functions

Statistical Functions

Logical Functions

Financial Functions

Filter Functions

Information Functions

…

Maths refresher

Calculating percentages

$$Percentage\ change = \frac{New\ value\ - Previous\ value}{Previous\ value} \times 100\%$$

$$Percentage\ Deviation = \frac{Actual\ - Plan}{Plan} = (Actual\ -\ Plan) \div Plan$$

Maths in Power BI

- Addition: +
- Subtraction: -
- Multiplication: *
- Division: /

Pro tip:

Use DIVIDE() instead of to set an exception in case the denominator is 0.

Maths refresher

Calculating with VAT rates

$$Price\ with\ VAT = Price\ without\ VAT \times (1 + VAT\ Rate)$$

$$Price\ without\ VAT = Price\ with\ VAT \div (1 + VAT\ Rate)$$

Maths in Power BI

- Addition: +
- Subtraction: -
- Multiplication: *
- Division: /

Pro tip:

Use DIVIDE() instead of to set an exception in case the denominator is 0.

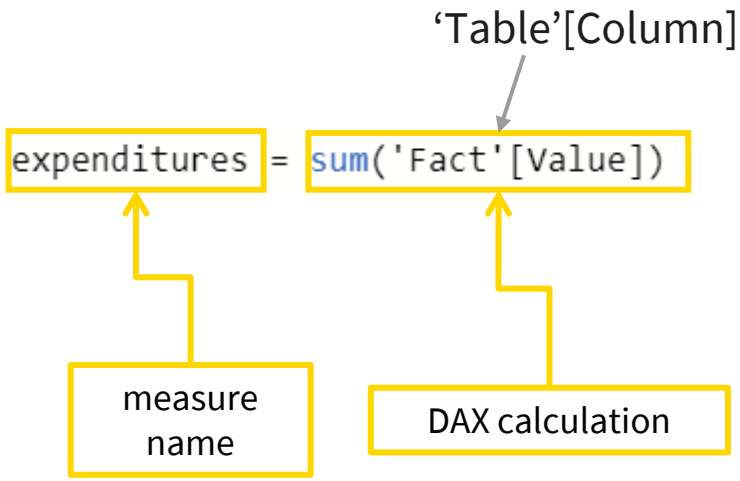
03

Hands-on Exercise 3.1

30 Minutes

Measures

- Calculate result from expression formula
- Aggregate and filter data
- Computed at runtime
- Stored temporarily
- Limited by filter context
- Appear in the fields list with a calculator icon:



03

Demo 3.2

CALCULATE() Function

- Evaluates an expression in a modified filter context.
- Syntax: CALCULATE(<**expression**>, <**filter context**>)

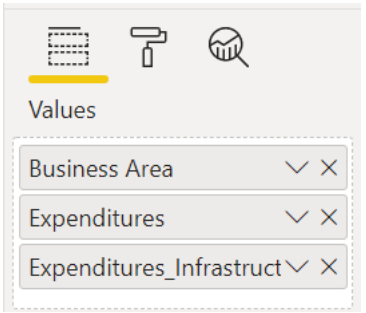
```
Expenditures = sum('Fact'[Value])
```

Expression:
„sum up entries in column ‚Value' from Fact table"

Modified filter context:
„sum up only data for Business Area ID = 4 (‚Infrastructure')"

```
Expenditures_Infrastructure = calculate(sum('Fact'[Value]), 'Fact'[Business Area ID] = 4)
```

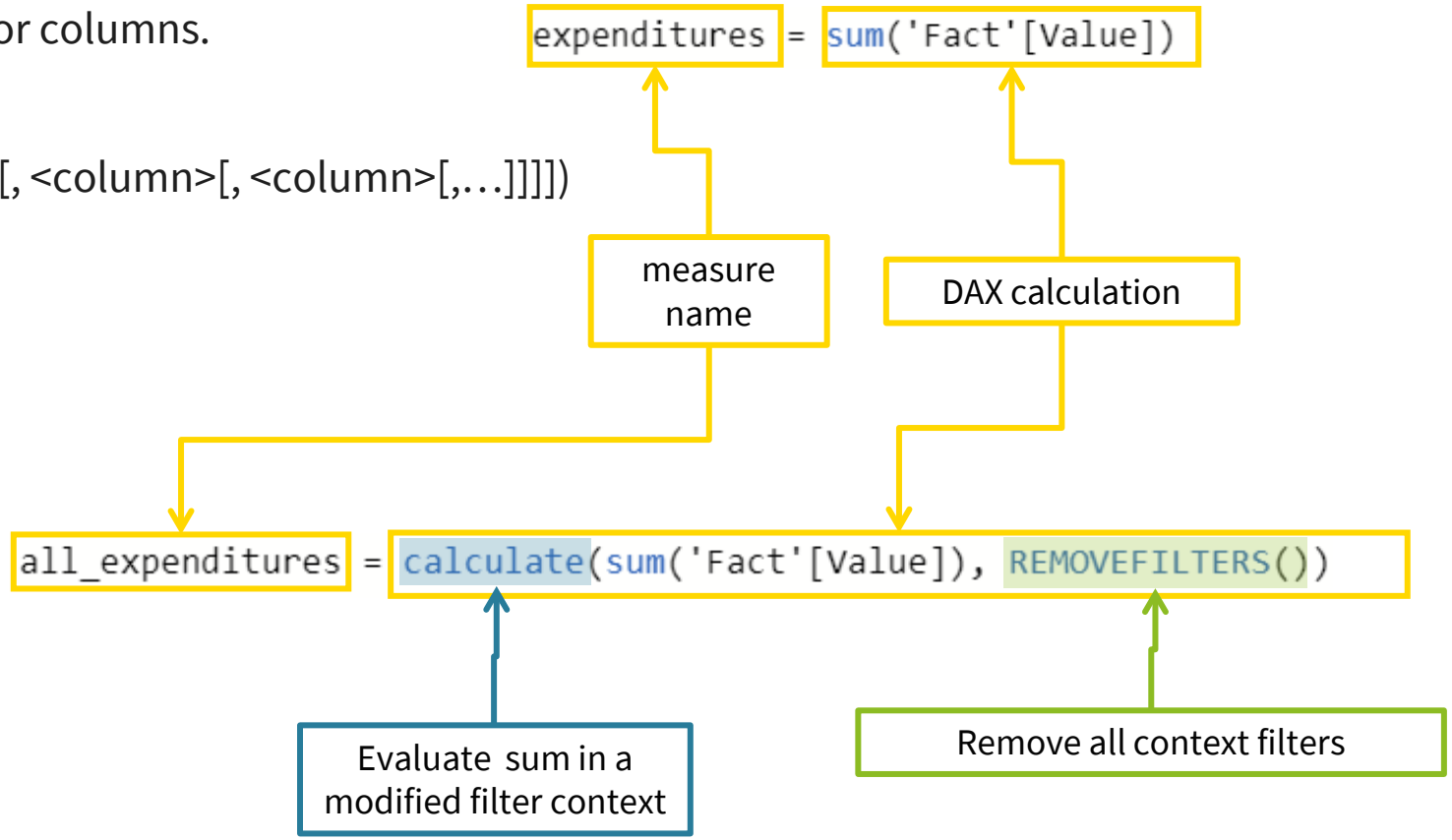
Business Area	Expenditures	Expenditures_Infrastructure
BU	261.777.647,86	
Distribution	98.194.198,98	
Infrastructure	432.082.111,27	$432.082.111,3
Manufacturing	261.682.104,93	
Office & Administrative	320.313.323,57	
R&D	223.591.308,98	
Services	71.729.648,33	
Total	**1.669.370.343,92**	**$432.082.111,3**



REMOVEFILTERS() Function

- Clear filters from the specified tables or columns.
- Syntax:

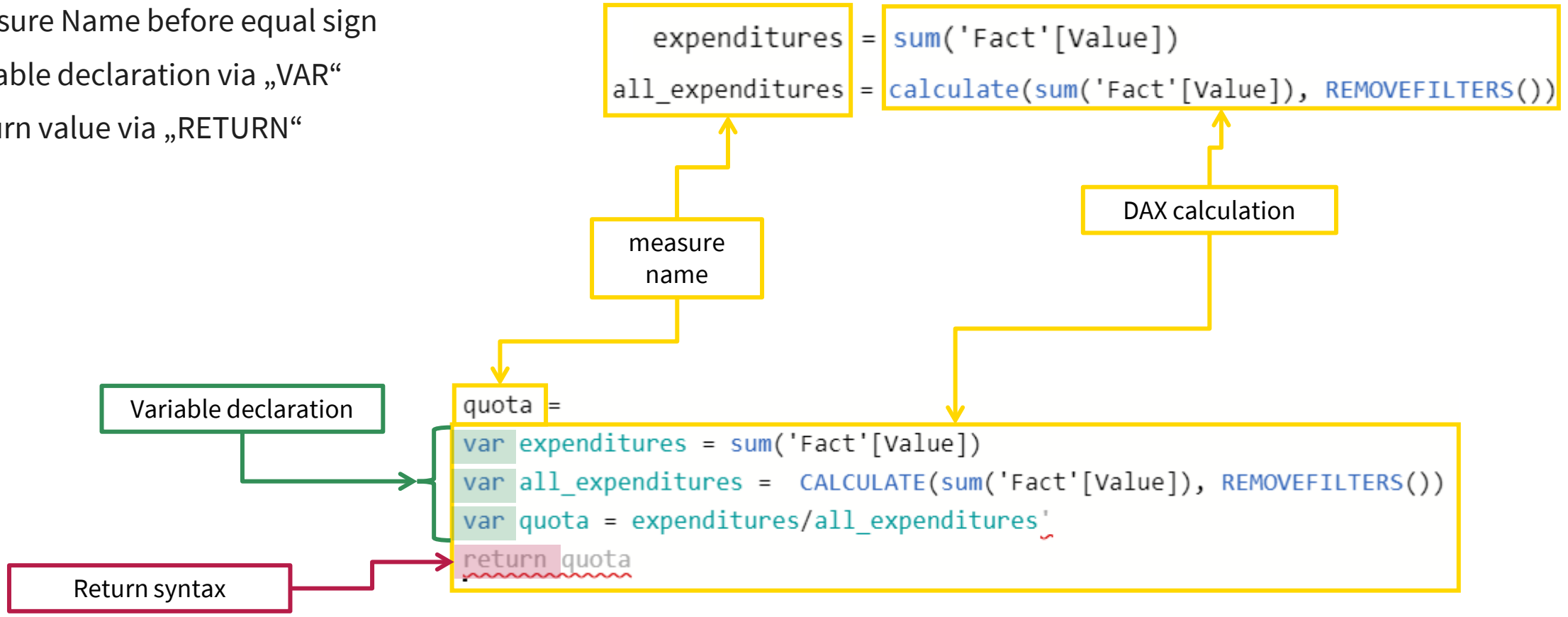
  REMOVEFILTERS([<table> | <column>[, <column>[, <column>[,…]]]])



Business Area	expenditures	all_expenditures
BU	126.144.842,35	858.418.988,84
Distribution	48.161.760,29	858.418.988,84
Infrastructure	255.268.858,10	858.418.988,84
Manufacturing	129.071.223,95	858.418.988,84
Office & Administrative	156.108.319,72	858.418.988,84
R&D	106.845.087,66	858.418.988,84
Services	36.818.896,77	858.418.988,84
Total	**858.418.988,84**	**858.418.988,84**

Measures Syntax

- Measure Name before equal sign
- Variable declaration via „VAR“
- Return value via „RETURN“



03

Demo 3.3

03

Hands-on Exercise 3.2

30 Minutes

03

Hands-on Exercise 3
Takeaways

- Calculated columns look like normal columns
- Related() function can be used to refer to columns from other tables
- Measures appear in field list, but not as a column in data view
- Slicers will filter data before it is evaluated in other visuals

03

Questions?

ENG

ventum-consulting.com | München | Essen | Wien | Foshan | Beijing | Shanghai

Power BI Desktop Advanced Workshop

Day 2

13/06/2023

Guest WiFi password: 133133_ventum_guests

TRATON

ventum

Agenda

01 **Introduction to Power BI**
Architecture & Environment

02 **Data Modeling**
Introduction Key Concepts

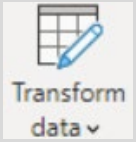
03 **Introduction to DAX**
Calculated Columns & Measures

04 **Introduction to M**
Queries, Columns, Joins Data Types

05 **Advanced DAX**
Time Intelligence & Iterator Functions

06 **Power BI Service**
Workspaces, Dashboards & RLS

Key Takeaways



Navigation through Power Query editor

Basics of M

Basics of ETL in Power BI

Evaluation of measures in different time contexts

Apply iterator functions and control for granularity

Navigation through Power BI Service

Basic Dashboards

RLS

04

Introduction to M

Power Query M syntax in a nutshell

Variables

```
let
    myVariable = 10,
    anotherVariable = "Hello, world!"
in
    myVariable + 5
```

Functions

```
let
    myVariable = Text.Length("Hello, world!")
in
    myVariable
```

Operators

```
let
    additionResult = 5 + 3,
    concatenationResult = "Hello" & " " & "world!"
in
    concatenationResult
```

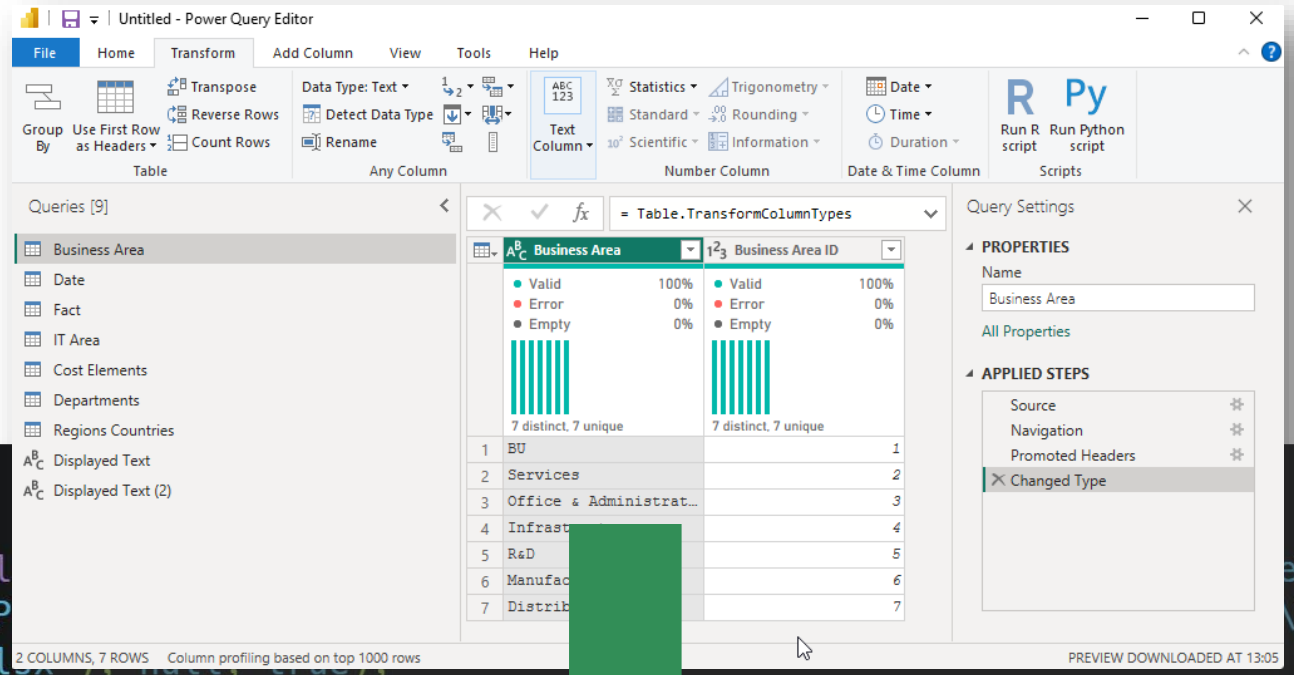
Data types

```
let
    myNumber = 10,
    myText = "Hello, world!",
    myExplicitText = Text.From(123)
in
    myExplicitText
```

Accessing data

```
let
    source = Csv.Document(File.Contents("C:\data.csv")),
    data = Table.PromoteHeaders(source)
in
    data
```

Power Query M code is generated automatically using a WYSIWYG editor

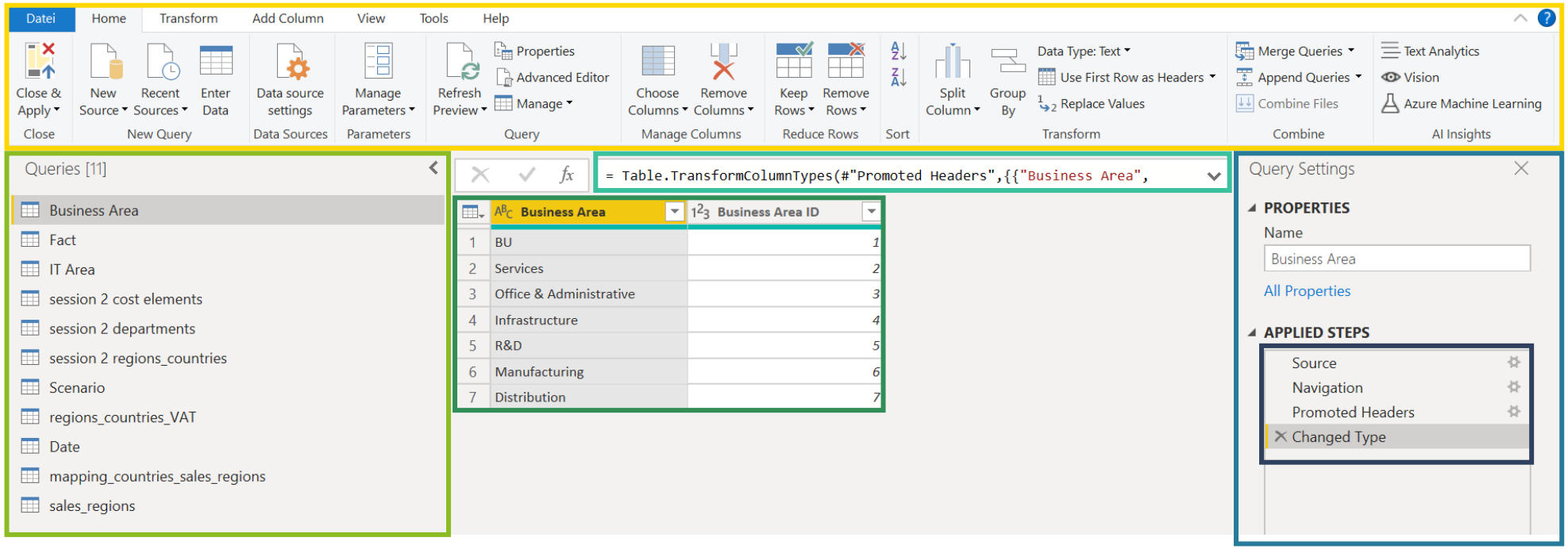



```
let
    Source = Excel
Co. KG\Documents\P                                    entum Consulting GmbH &
Data\Exercise 1.xl                                    \Exercise 1 Sample
    #"Business Area_Sheet" = Source{[Item="                Area",Kind="Sheet"]}[Data],
    #"Promoted Headers" = Table.PromoteHeade                usiness Area_Sheet", [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnType(#"Promoted Headers",{{"Business Area", type text},
{"Business Area ID", Int64.Type}})
in
    #"Changed Type"
```

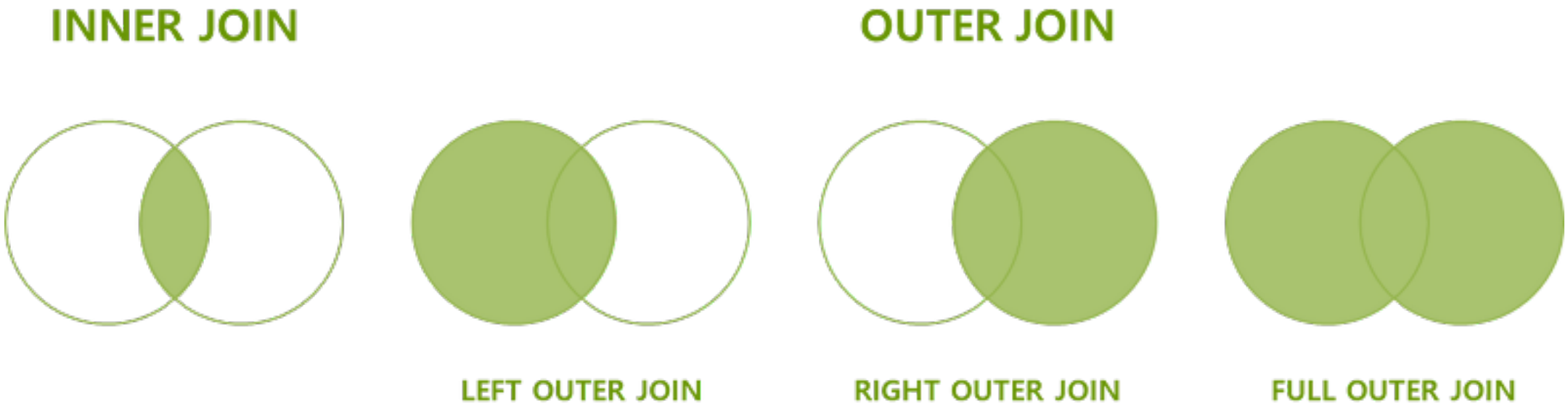
Introduction to M

Power Query formula language

- Renaming queries
- Renaming columns
- Change data types
- Filter rows
- Replacing values
- Create new columns based on existing ones
- Combine queries from multiple sources
- Create new tables
- Split columns
- Enhance queries that were automatically created while loading data
- Python & R integration



Joining tables



Appending tables

Table1

Item	Jan	Feb	Mar
A	78	61	97
B	91	59	81
C	29	98	65

Table2

Item	Jan	Feb	Mar
D	95	41	99
E	31	27	6
F	91	1	36

Table3

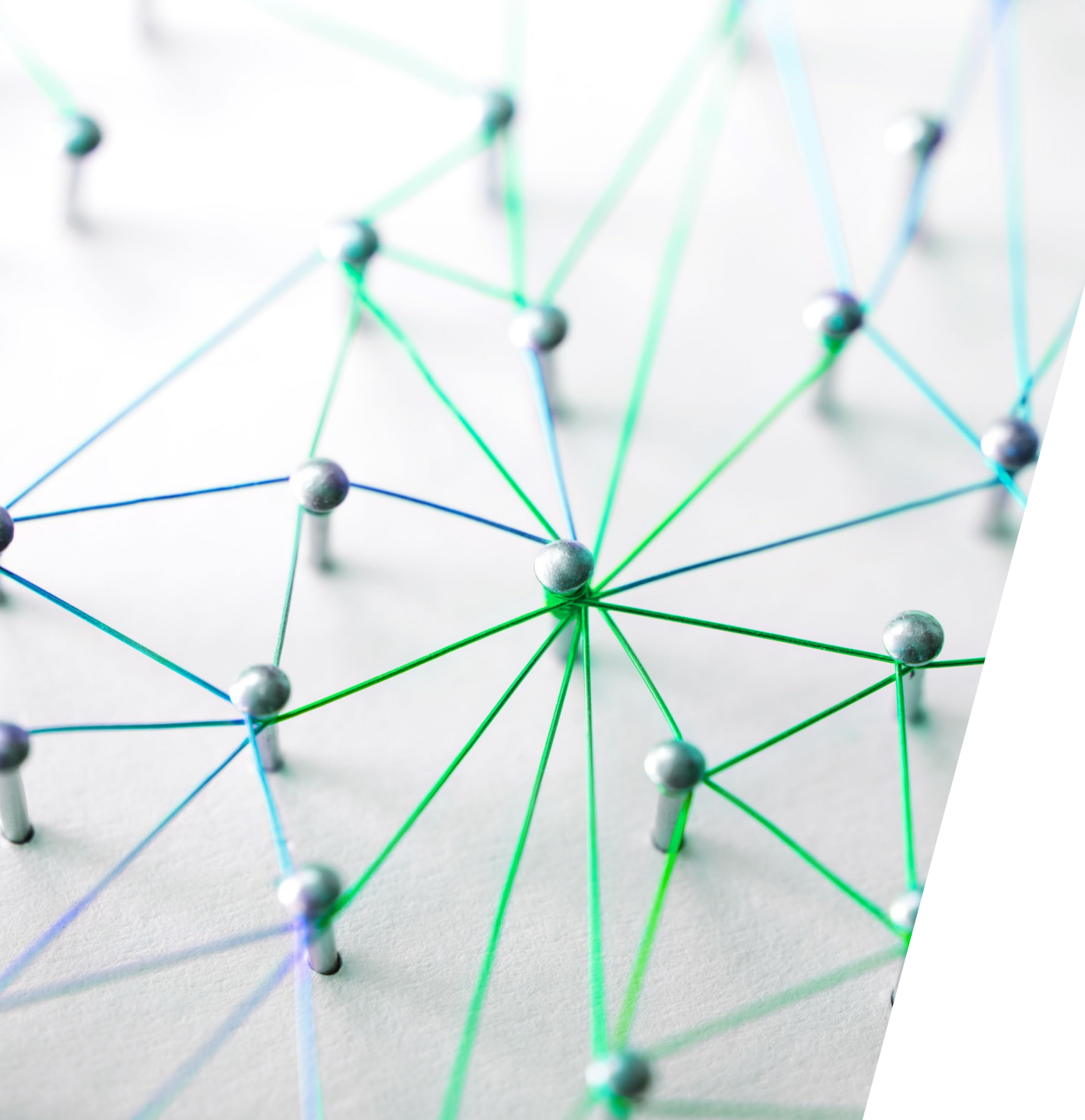
Item	Jan	Feb	Mar
G	22	84	73
H	96	37	54
I	11	31	99

Case1 Result

Item	Jan	Feb	Mar
A	78	61	97
B	91	59	81
C	29	98	65
D	95	41	99
E	31	27	6
F	91	1	36
G	22	84	73
H	96	37	54
I	11	31	99

04

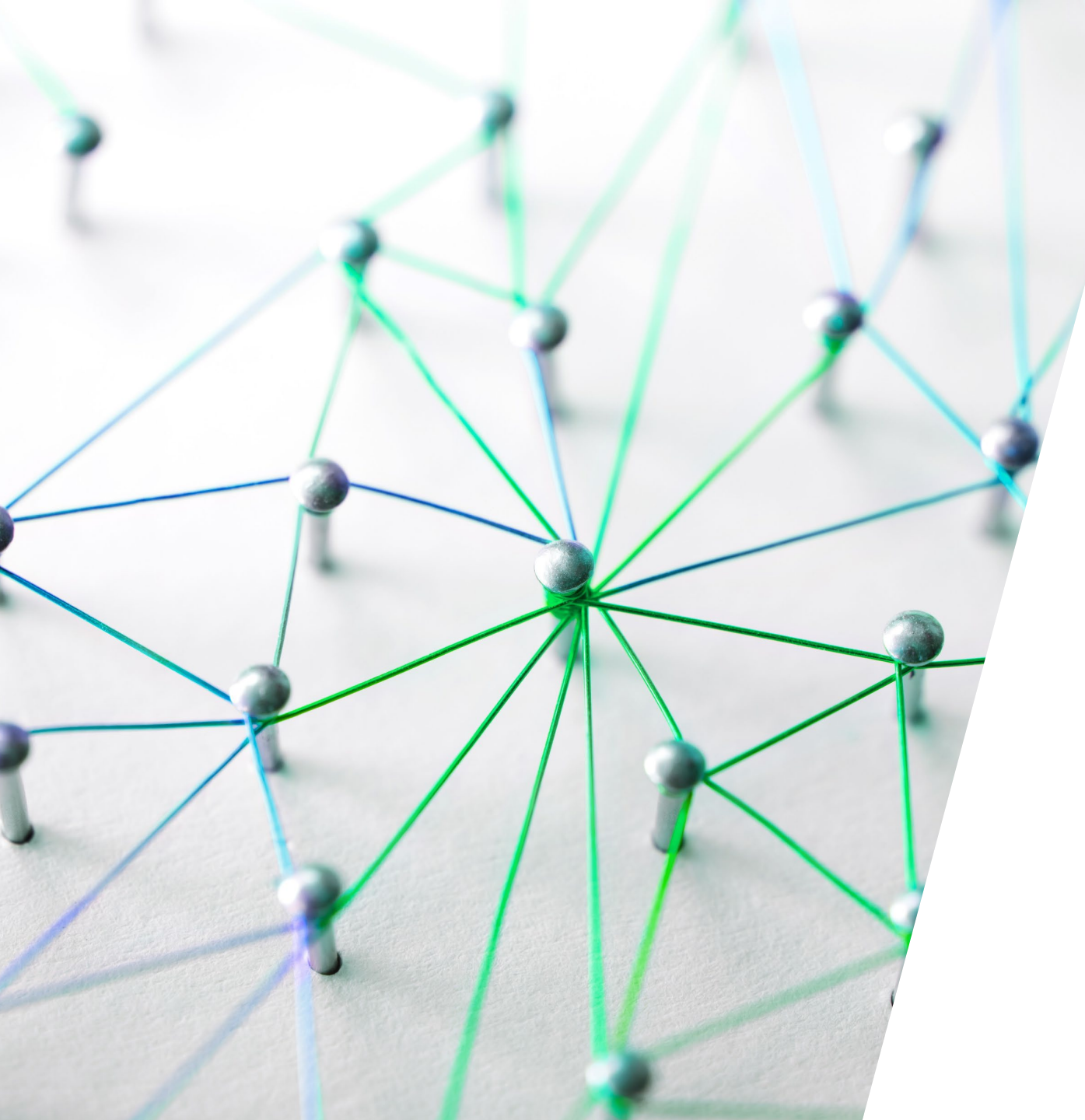
Demo Power Query M



04

Hands-on Exercise 4
Introduction to M

40 Minutes & 5 Minutes Break



04

Hands-on Exercise IV Introduction to M

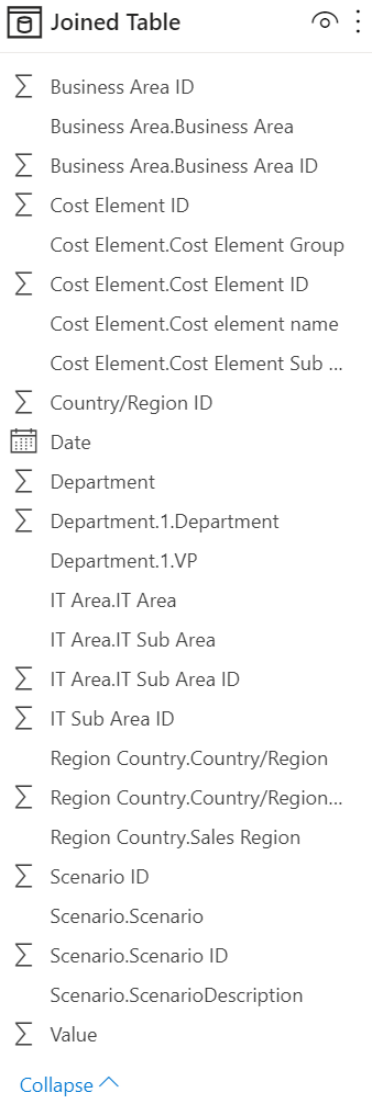
Takeaways

- Renaming queries
- Changing data types
- Adding columns
- Joining queries
- Filtering data according to current date
- Integrating data from SharePoint
- Adding reset button

Joins vs. Relationships

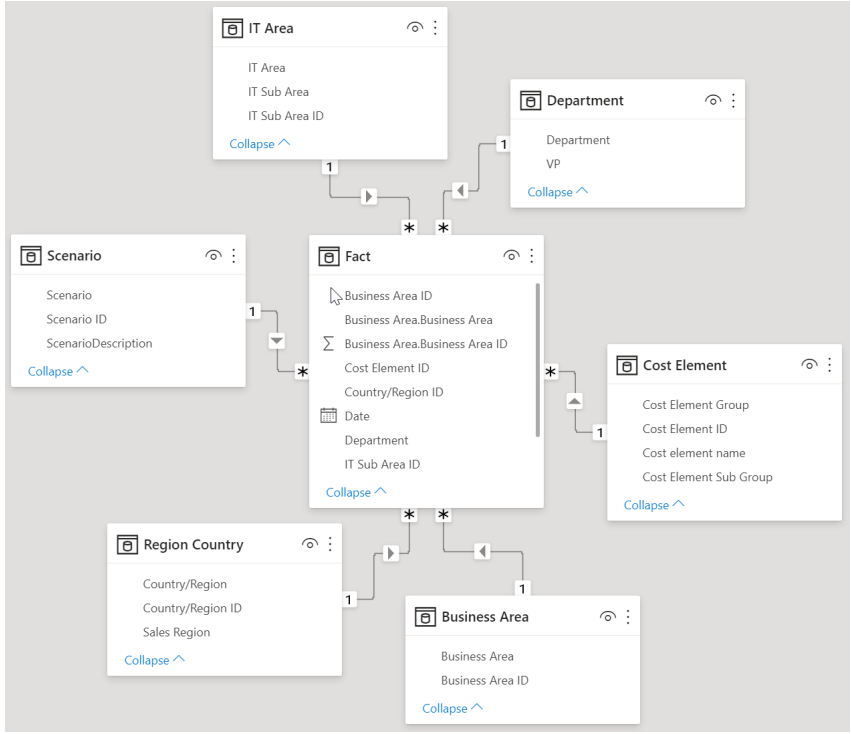
Joins

- Performed in M-Query and DAX-Tables
- Work like SQL joins
- Create one big table
- Danger of duplicate values
- Require more disc space than relationships
- Implementing changes can become very difficult
- Identifying dimensions and granularity can become difficult



Relationships

- Power BI calculates table data as needed
- Closed circular filter directions are not possible
- Many relationships can slow down performance of a report
- A data model with many relationships can become very cluttered and hard to read



DAX vs M

(„data wrangling“)

M is used for data engineering, so:

- **Data Ingestion**:

Power Query allows you to import, combine, and update data from various sources (e.B. from a local folder, SharePoint, Database, API, Azure, and many more).

- **ETL processes**:

Extract – Transform – Load, and Extract – Load – Transform are the two approaches to data processing. For both, Power Query is the tool of the hour.

- **Data Wrangling**:

The data is then cleaned and transformed. Joins are made, filtered, uniform names are assigned, unnecessary columns are removed, etc.

DAX is used for data analysis, so:

- **Aggregations**:

If you need to retrieve aggregated values from a table or dataset, then the best and fastest way to do so is with DAX aggregate functions.

- **Filtering**:

In measures, filtering functions are among the most commonly used. They help to filter only the relevant information and/or edit the data context to create dynamic calculations without changing the data model and its tables.

- Define **relationships**:

Power Query knows no relations. In DAX, on the other hand, you can define and change relationships, but also enforce them using referential integrity.

- **Relationship functions** (**hierarchies**):

DAX contains functions to return e.g. the cardinality of relations and records.

- **Calculated column that references a column from another table**:

With DAX, the user can easily insert columns from different tables into the formula. In Power Query, this maneuver requires complex joining and becomes complicated very quickly.

DAX vs M

Difference in when the code is run

- **M** is run during **data refreshes**
- **DAX measures** are run when **visuals are loaded** that use the measure and respond to **user interactions**
- **DAX calculated columns** are run whenever a report is loaded and increase the size of the data model

... because of this:

- Data sets that we will never need in data analysis are filtered in M
- We do not aggregate records into M
- Columns that we never need are removed in M
- We do not do calculations in M

05

Advanced DAX

DAX Time Intelligence

- Support calculations to compare and aggregate data over time periods
- Wide range of functions, including
  - Start of Month/quarter/year
  - Previous day/month/quarter
  - Dateadd
  - Year/quarter/month to date
  - …

```
perc_deviation_plan_actual = ('Fact'[Plan]-'Fact'[Actual])/'Fact'[Actual]
```

Date	perc_deviation_plan_actual	last_month_perc_deviation
01.01.2014	0,38	NaN
01.02.2014	0,02	0,38
01.03.2014	-0,03	0,02
01.04.2014	0,04	-0,03
01.05.2014	-0,11	0,04
01.06.2014	-0,05	-0,11
01.07.2014	-0,05	-0,05
01.08.2014	-0,07	-0,05
01.09.2014	-0,10	-0,07
01.10.2014	-0,07	-0,10
01.11.2014	-0,13	-0,07
01.12.2014	-0,31	-0,13
Total	**-0,05**	**-0,02**

```
1 last_month_perc_deviation_plan_actual =
2 Var result = CALCULATE (
3     [perc_deviation_plan_actual],
4     DATEADD ( 'Fact'[Date], -1, Month )
5 )
6 return result
```

Evaluate an expression with a different time context

05

Demo 5.1
DAX Time Intelligence

DAX Iterator Functions

- Enumerate rows & evaluate expression for each row
- Require table and expression
- Evaluated in filter context
- Indicated by an x at the end of formula name

SUM	⟷	SUMX
COUNT	⟷	COUNTX
MIN	⟷	MINX
MAX	⟷	MAXX
AVERAGE	⟷	AVERAGEX

Example:

```
Plan = calculate(sum([total_expenditure]), Scenario[ScenarioDescription]= "Plan")+0
Actual = calculate(sum([total_expenditure]), Scenario[ScenarioDescription]= "Actual")+0
```

```
1  sum_plan_minus_actual = SUM( 'Fact'[Plan]- 'Fact'[Actual])
```

! The SUM function only accepts a column reference as an argument.

```
sumx_plan_minus_actual = SUMX( 'Fact', 'Fact'[Plan]- 'Fact'[Actual])
```

DAX Iterator Functions

- Enumerate rows & evaluate expression for each row
- Require table and expression
- Evaluated in filter context
- Indicated by an x at the end of formula name

SUM	⟷	SUMX
COUNT	⟷	COUNTX
MIN	⟷	MINX
MAX	⟷	MAXX
AVERAGE	⟷	AVERAGEX

Example:

```
1  sum_plan_minus_actual = SUM( 'Fact'[Plan]- 'Fact'[Actual])
```

! The SUM function only accepts a column reference as an argument.

```
sumx_plan_minus_actual = SUMX( 'Fact', 'Fact'[Plan]- 'Fact'[Actual])
```

Date	Plan	Actual	sum_plan_minus_actual
01.01.2014	$89.664.416	$65.194.791	$24.469.625
01.02.2014	$74.519.463	$73.155.763	$1.363.700
01.03.2014	$78.377.130	$80.537.287	($2.160.157)
01.04.2014	$76.419.640	$73.636.319	$2.783.322

DAX Iterator Functions

Complex summarization:

- Aggregate data from more than one column
- Refer to data from other columns

Higher grain summarization

- Raise granularity of data over a table or list of values

Example: average actual expenditure per month

```
average_expenditure_per_month = AVERAGEX(values('Fact'[Date]), [Actual])
```

Row based evaluation

Reduction of granularity to monthly data

Target data

05

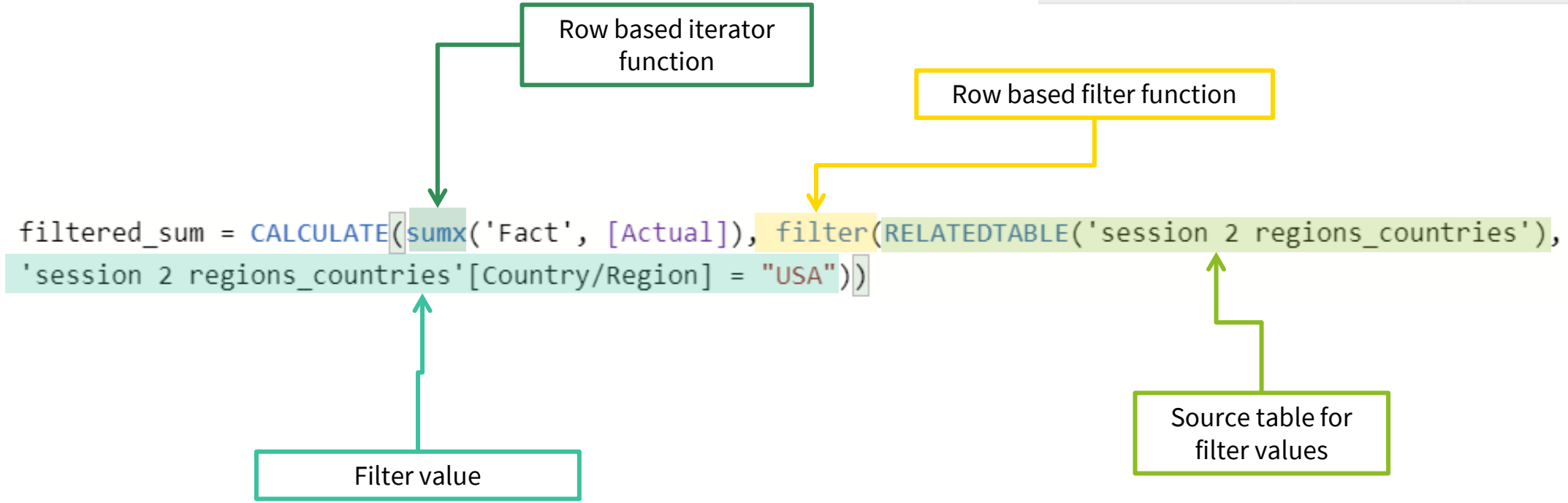
Demo 5.2
DAX Iterator Functions

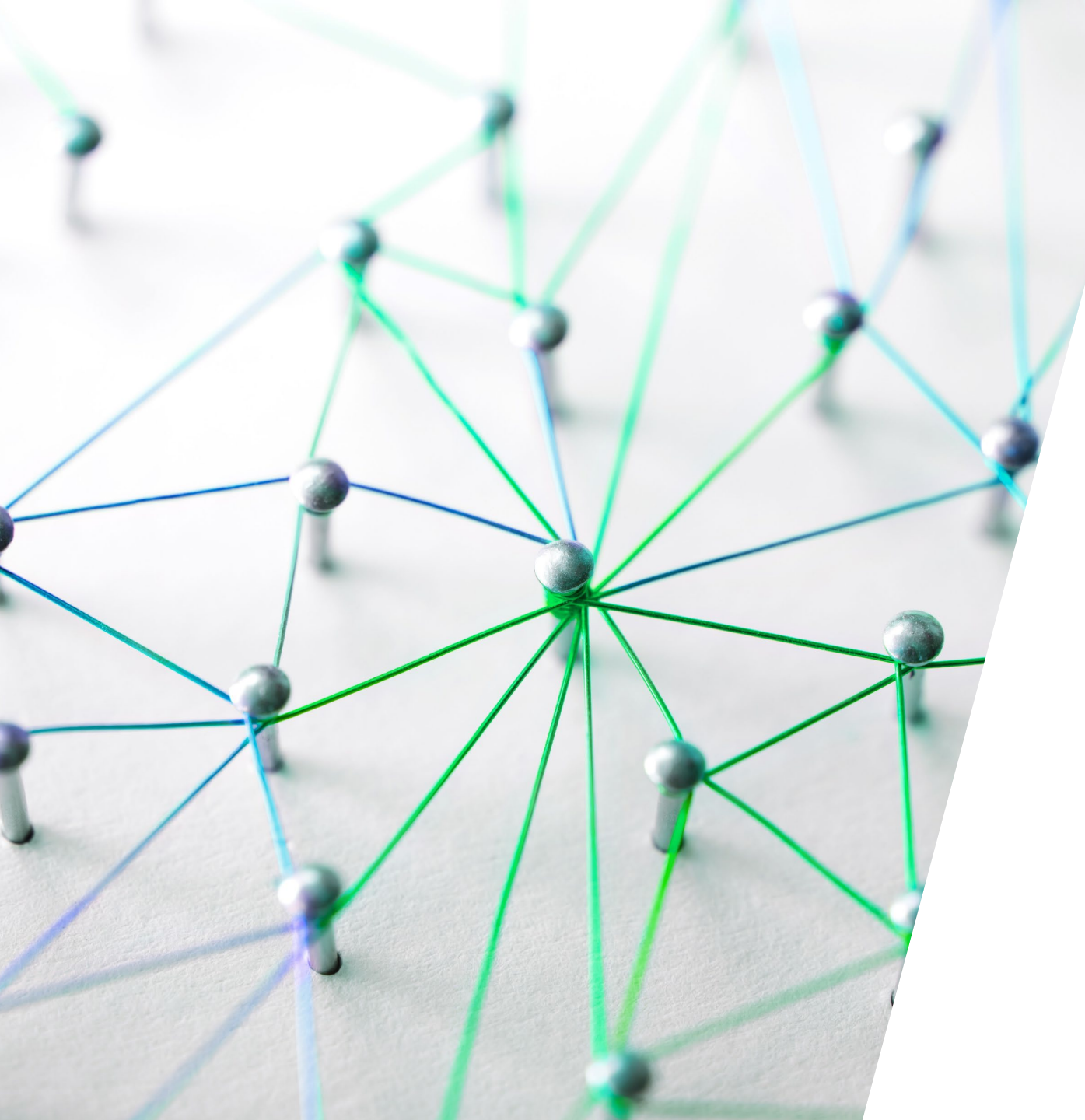
DAX Filter Function

- Returns a table that represents a subset of another table or expression
- Syntax: FILTER(<table name>, <filter condition>)
- Can be applied on iterator functions

Example:

Country/Region	Actual	filtered_sum
Venezuela	$0	
USA	$785.871.295	785.871.295,27
United Kingdom	$53.245.450	
Turkey	$126.618	

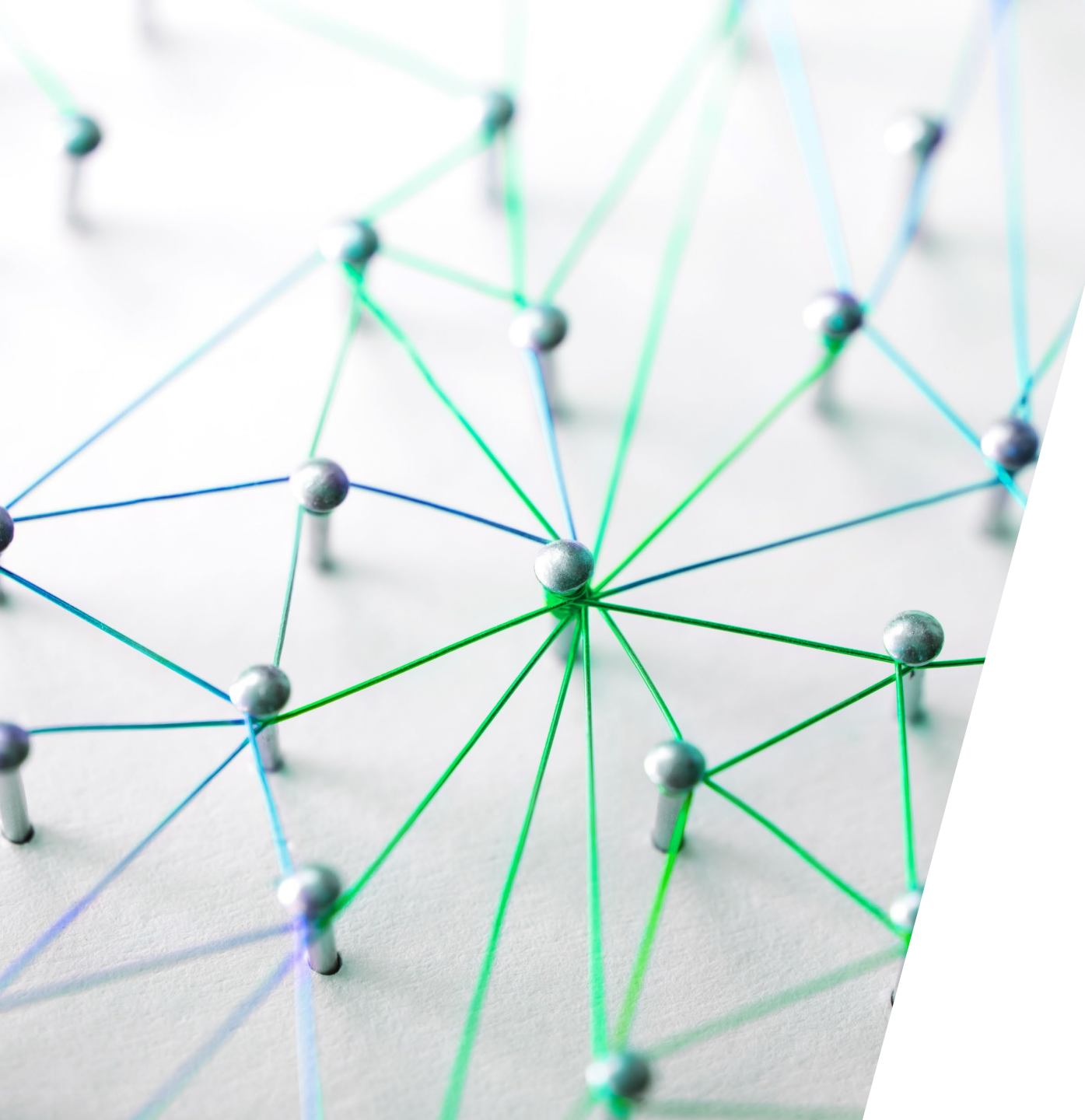




05

Hands-on Ecercise V
DAX Time Intelligence & Iterator Functions

30 Minutes & 10 Minutes Break



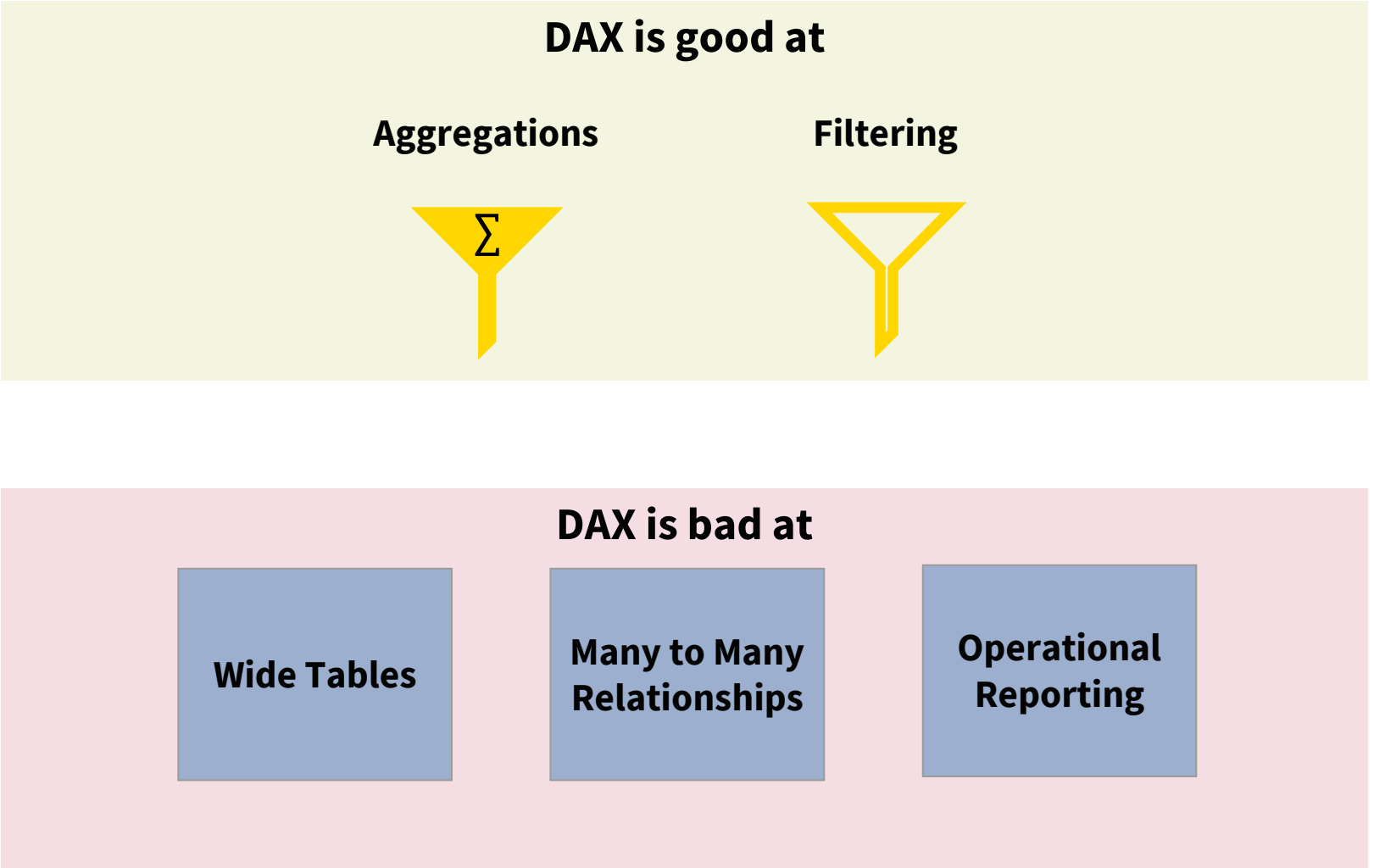
05

Hands-on Ecercise V DAX Time Intelligence & Iterator Functions

Takeaways

- Aggregation level can be changed via iteration funtions
- Rank can be calculated via RANKX-function
- An iterator function, such as SUMX, can be filtered by the filter function

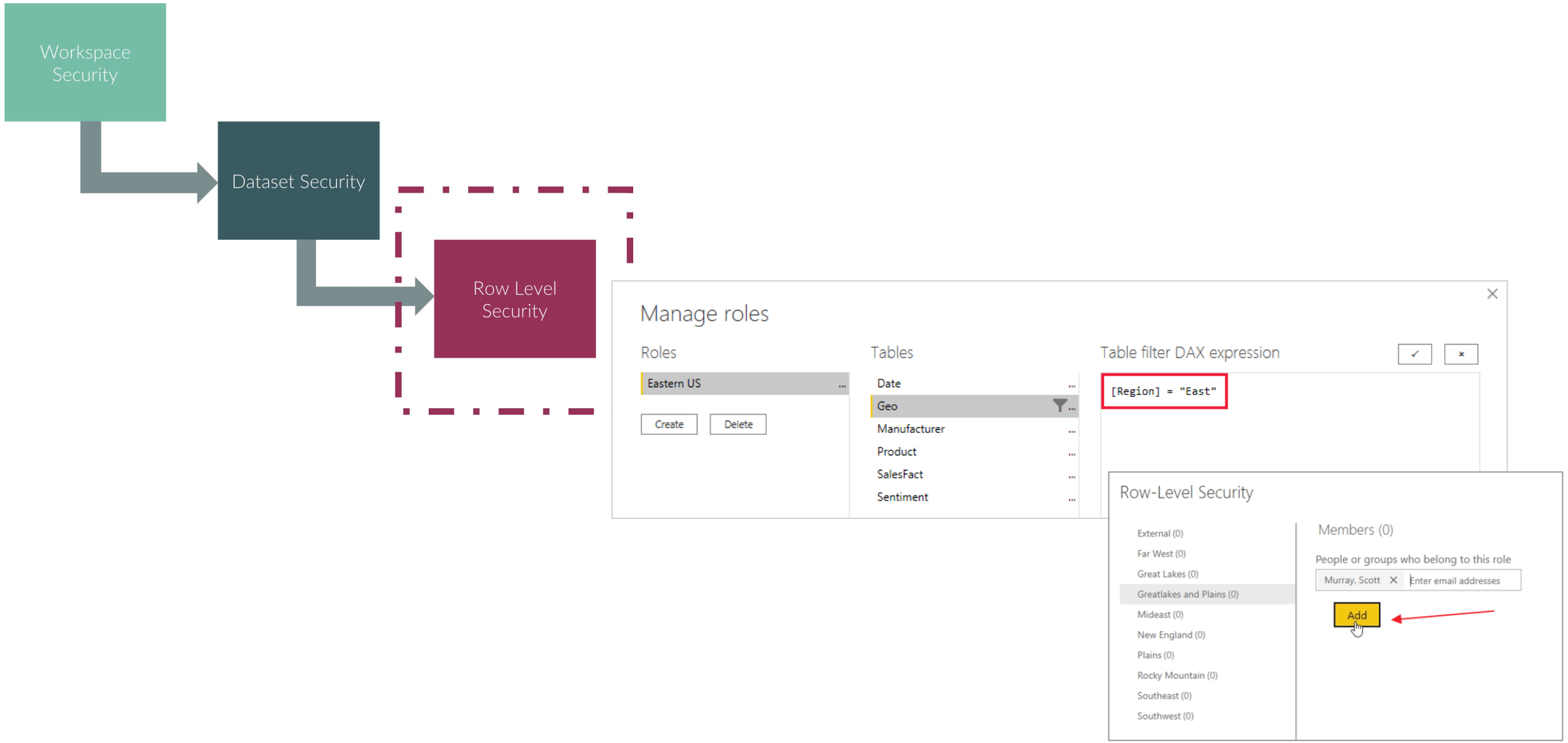
DAX perfomance



06

Power BI Service

Row-Level Security



Basic Features

- **Publishing**
  Publish your report to the web. You can embed it in websites or sharepoint or view it from mobile devices

- **Sharing and Collaboration**
  Share your report with colleagues and give the ability to add comments and questions

- **Role Management**
  Secure your data with row level security (RLS)

- **Scheduling**
  Schedule reloads of your online data

- **Dashboards**
  Create canvasses that contain zero or more tiles and widgets

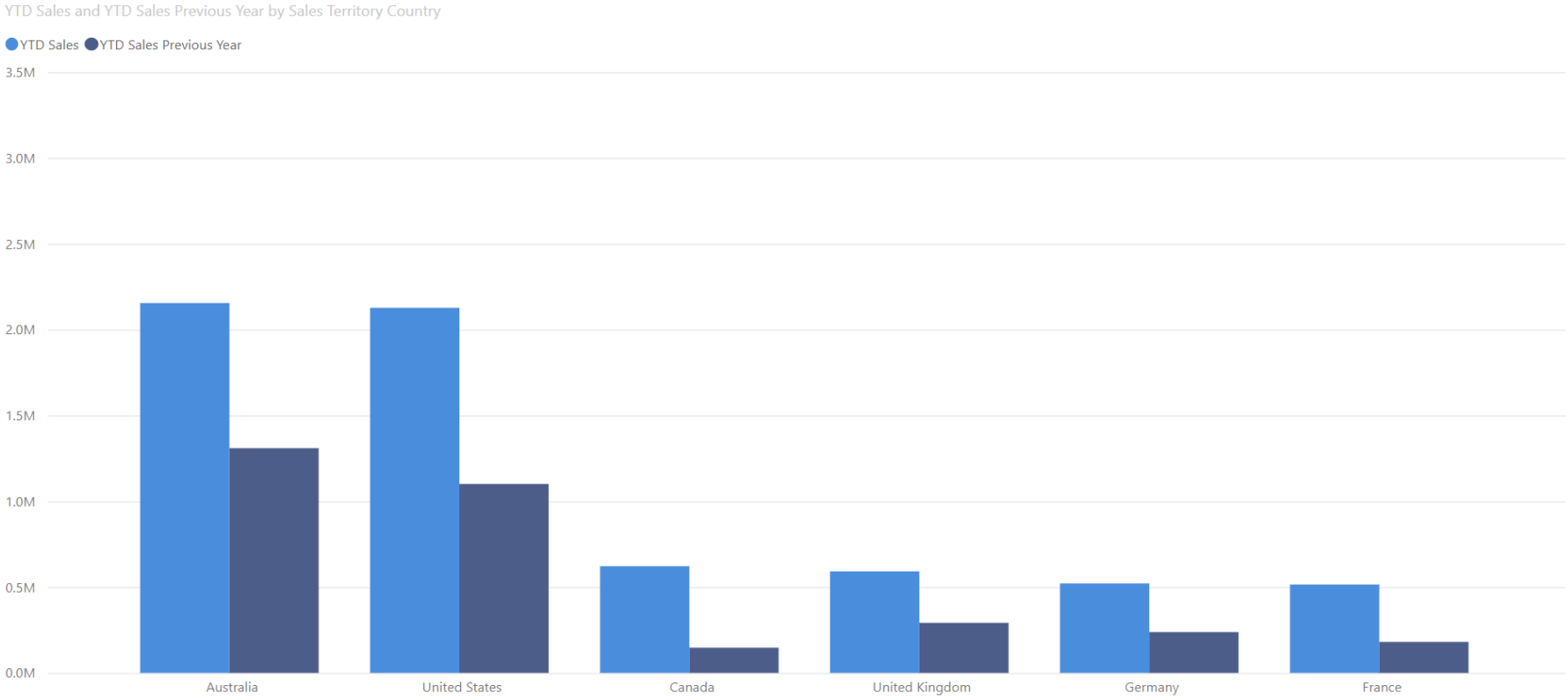
Power BI Workshop

TRATON

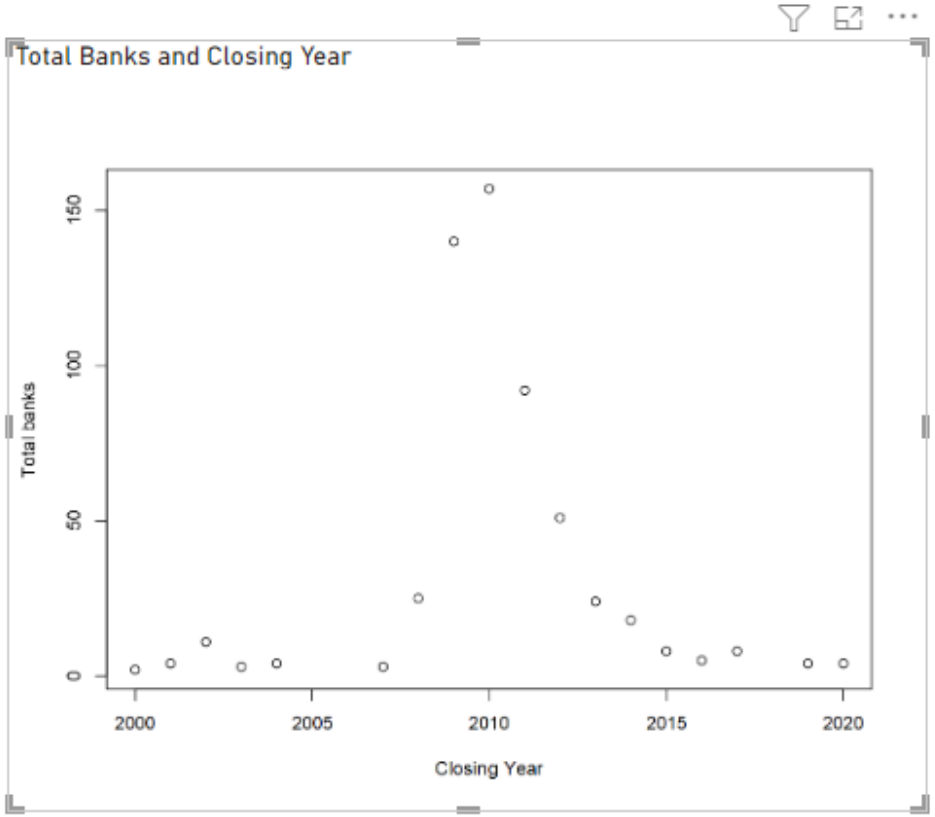
ventum

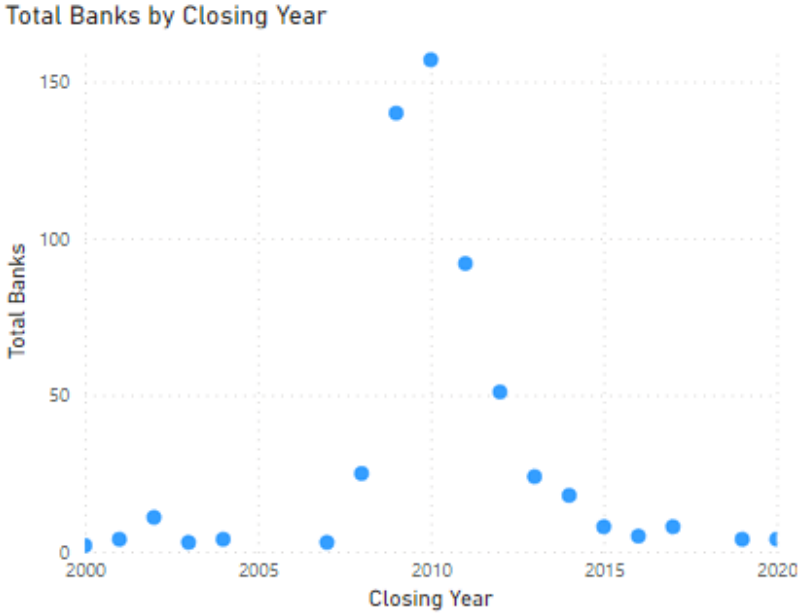
Advanced DAX

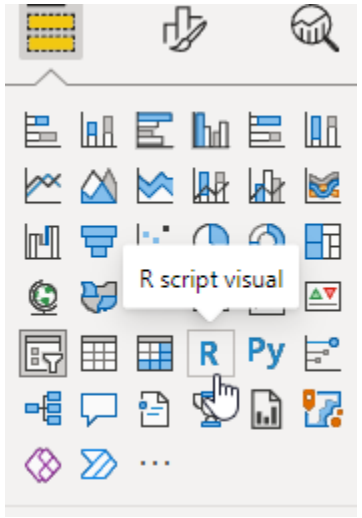
Example DAX: Comparison of cumulative calculations between years



Create custom visualisations with Python and R



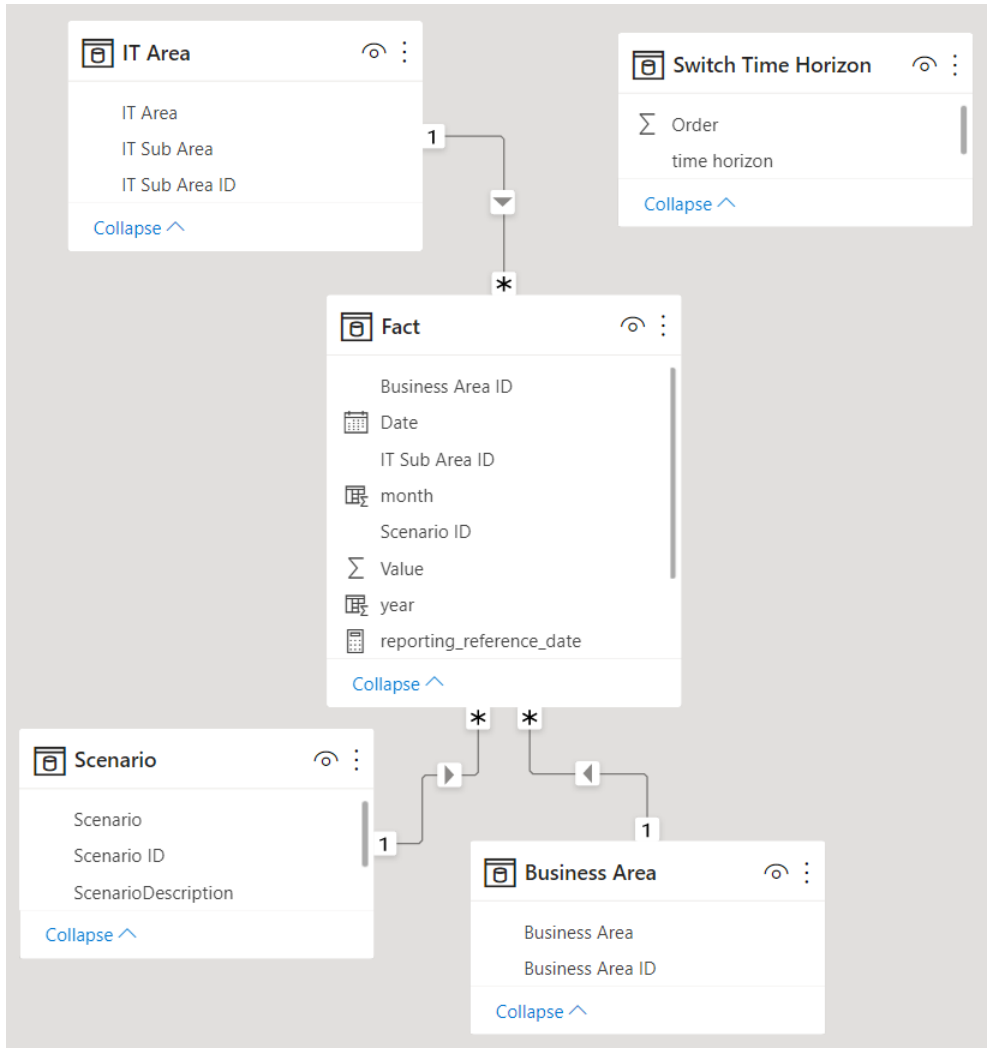




R script editor

```
1
2
3 plot(dataset$"Closing Year",dataset$'Total Banks', ylab = "Total banks", xlab = "Closing Year")
4
```

Data



Scenario	Scenario ID	ScenarioDescription
Actual	1	Actual
LE1	2	Latest Estimate 1
LE2	3	Latest Estimate 2
LE3	4	Latest Estimate 3
Plan	5	Plan

Order	time horizon
1	Default
2	YTD
3	L12M
4	FY

Business Area	Business Area ID
BU	1
Services	2
Office & Administrative	3
Infrastructure	4
R&D	5
Manufacturing	6
Distribution	7

IT Area	IT Sub Area	IT Sub Area ID
BU Support	Distribution	3
BU Support	Development	7
BU Support	Core	8
BU Support	Emerging	9
BU Support	Planning	21

Functions: SELECTEDVALUE

„Returns the value when the context for columnName has been filtered down to one distinct value only. Otherwise returns alternateResult."

```
selected_scenario = SELECTEDVALUE(Scenario[Scenario], "no scenario selected")
```

Scenario	selected_scenario
Actual	Actual
LE1	LE1
LE2	LE2
LE3	LE3
Plan	Plan
Total	**no scenario selected**

Functions: SWITCH

„Evaluates an expression against a list of values and returns one of multiple possible result expressions.”

```
switch_scenario_option = SWITCH(Scenario[selected_scenario],
"Plan", "let's analyze planned values",
"Actual", "let's analyze actual values",
"LE1", "let's analyze LE1 values",
"LE2", "let's analyze LE2 values",
"LE3", "let's analyze LE3 values",
"no scenario selected", "no scenario selected to analyze"
)
```

Functions: SWITCH

```
selected_scenario = SELECTEDVALUE(Scenario[Scenario], "no scenario selected")
```

```
switch_scenario_option = SWITCH(Scenario[selected_scenario],
"Plan", "let's analyze planned values",
"Actual", "let's analyze actual values"
"LE1", "let's analyze LE1 values",
"LE2", "let's analyze LE2 values",
"LE3", "let's analyze LE3 values",
"no scenario selected", "no scenario selected to analyze"
)
```

Scenario	switch_scenario_option
Actual	let's analyze actual values
LE1	let's analyze LE1 values
LE2	let's analyze LE2 values
LE3	let's analyze LE3 values
Plan	let's analyze planned values
Total	**no scenario selected to analyze**

Functions: SWITCH & SELECTEDVALUE

SWITCH and SELECTEDVALUE are very frequently combined!

„Default value“

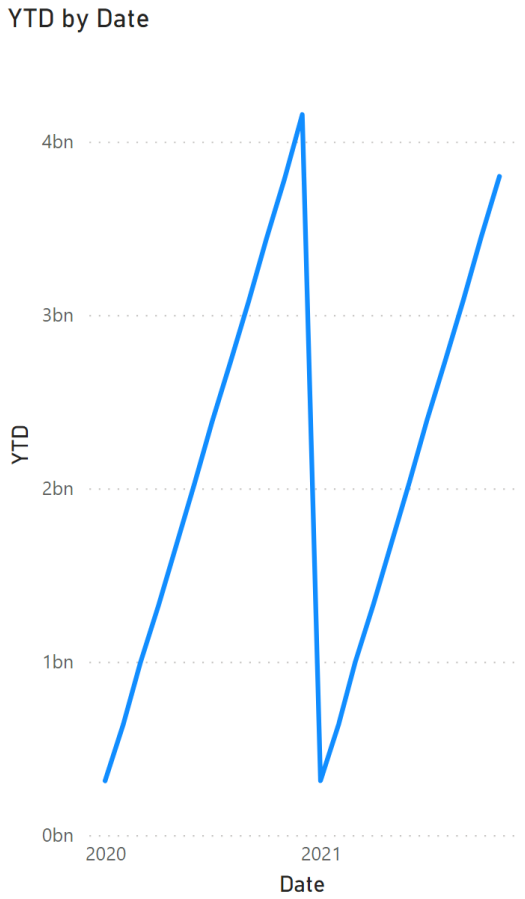
```
switch_scenario_option = SWITCH(SELECTEDVALUE(Scenario[Scenario], "no scenario selected"),
"Plan", "let's analyze planned values",
"Actual", "let's analyze actual values",
"LE1", "let's analyze LE1 values",
"LE2", "let's analyze LE2 values",
"LE3", "let's analyze LE3 values",
"no scenario selected", "no scenario selected to analyze"
)
```

Time Intelligence Functions II

TOTALYTD: Evaluates the year-to-date value of the expression in the current context.

```
YTD = TOTALYTD([sum],'Fact'[Date])
```

Date	YTD
01.01.2020 00:00:00	313.029.231,25
01.02.2020 00:00:00	639.226.971,16
01.03.2020 00:00:00	994.406.880,03
01.04.2020 00:00:00	1.326.543.882,00
01.05.2020 00:00:00	1.671.292.289,60
01.06.2020 00:00:00	2.027.689.179,62
01.07.2020 00:00:00	2.385.809.592,13
01.08.2020 00:00:00	2.727.994.518,58
01.09.2020 00:00:00	3.078.710.422,25
01.10.2020 00:00:00	3.437.927.527,10
01.11.2020 00:00:00	3.786.183.044,66
01.12.2020 00:00:00	4.155.555.872,60
01.01.2021 00:00:00	313.457.981,98
01.02.2021 00:00:00	640.744.400,59
01.03.2021 00:00:00	997.591.940,79
01.04.2021 00:00:00	1.331.051.623,51
01.05.2021 00:00:00	1.677.406.474,72
01.06.2021 00:00:00	2.034.624.344,41
01.07.2021 00:00:00	2.393.765.974,38
01.08.2021 00:00:00	2.736.932.686,35
01.09.2021 00:00:00	3.088.840.270,15
01.10.2021 00:00:00	3.449.291.686,32
01.11.2021 00:00:00	3.798.967.307,91
Total	**3.798.967.307,91**



Time Intelligence Functions I

DATE: Returns the specified date in datetime format.

```
Date_measure = DATE(2021,12,1)
```

Date_measure
01.12.2021 00:00:00

YEAR: Returns the year of a date as a four digit integer in the range 1900-9999

```
year_measure = YEAR([Date_measure])
```

year_measure
2021

MONTH: Returns the month as a number from 1 (January) to 12 (December).

```
month_measure = MONTH([Date_measure])
```

month_measure
12

STARTOFYEAR: Returns the first date of the year in the current context for the specified column of dates.

```
startofyear_measure = STARTOFYEAR('Fact'[Date])
```

Date	startofyear_measure
01.08.2021 00:00:00	01.01.2021 00:00:00
01.09.2021 00:00:00	01.01.2021 00:00:00
01.10.2021 00:00:00	01.01.2021 00:00:00
01.11.2021 00:00:00	01.01.2021 00:00:00
Total	**01.01.2021 00:00:00**

ENDOFYEAR: Returns the last date of the year in the current context for the specified column of dates.

```
endofyear_measure = ENDOFYEAR('Fact'[Date])
```

Date	endofyear_measure
01.01.2020 00:00:00	01.12.2020 00:00:00
01.02.2020 00:00:00	01.12.2020 00:00:00
01.03.2020 00:00:00	01.12.2020 00:00:00
Total	**01.12.2020 00:00:00**

Filter Functions

FILTER: Returns a table that represents a subset of another table or expression.

REMOVEFILTERS: Clear filters from the specified tables or columns.

ALL: „Returns all the rows in a table, or all the values in a column, ignoring any filters that might have been applied”

- Can be used analogous to REMOVEFILTERS when used as filter

```
CALCULATE([Switch time horizon (all | sum)], ALL('Fact'[year]))
=
CALCULATE([Switch time horizon (all | sum)], REMOVEFILTERS('Fact'[year]))
```

Boolean Filter (not a function):

```
all_over_1k = CALCULATE(sum('Fact'[Value]),'Fact'[Value]>1000)
```

Hands-on Exercise
Advanced DAX

Questions?