Power BI

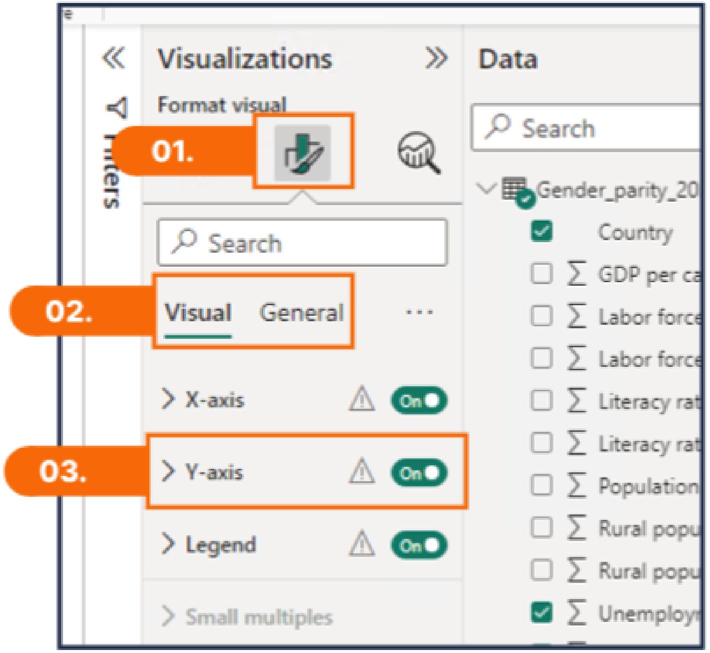
The formatting pane in Power BI

> The **Format your visual pane** in Power BI allows us to modify our visuals in various ways. It offers a **wide range of options and settings that can be adjusted** for each visual element.

How to navigate to the "Format your visual" pane:

01. Select the visual. Go to the **Visualizations** pane and click on the **paintbrush icon**.
02. This will display the formatting pane divided into two sections, **Visual** and **General**. Each section provides a list of formatting options related to the various elements found in a visual.
03. To modify a specific element, go to the appropriate option on the list, then **click on the dropdown** next to it to expand its available settings.

Note: The formatting settings displayed depend on the type of visual selected.



Themes in Power BI

The **formatting pane** allows us to **only format a single visual** at a time.

If we have **multiple visuals** in a report page, we can **use Power BI themes** to format them all at once such that they share the same colours, fonts, and other design elements.

Power BI has **several built-in themes**, each with their own **predefined formatting settings.** Automatically, a new report is given a **default theme.**

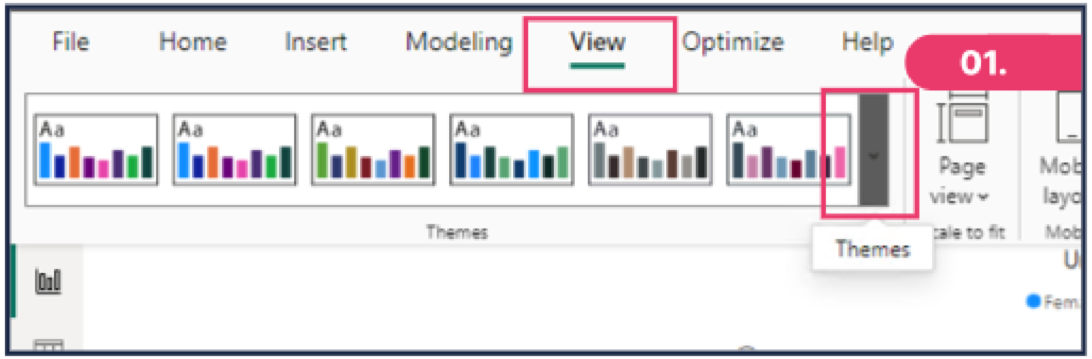
Apply a theme

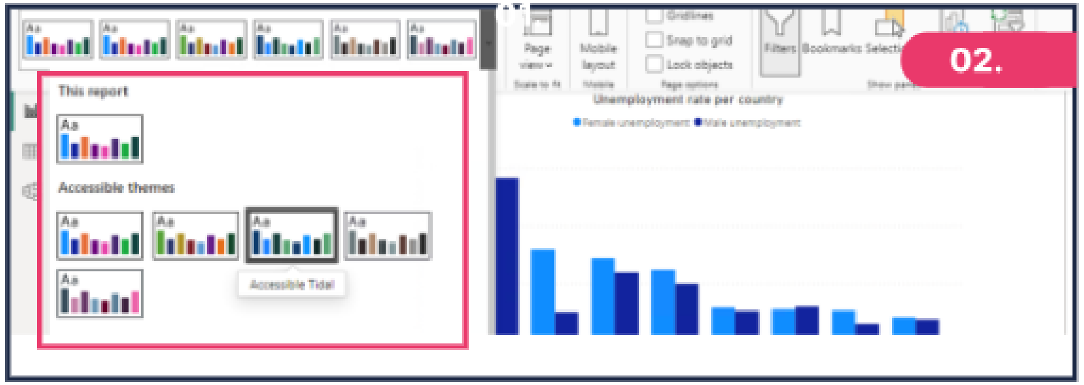
To apply a different theme, we follow these steps:

01. Select **View** at the top. Go to **Themes** and click on the dropdown.
02. **Select** a **theme** from the options provided.

The **theme is applied** and the colours, fonts, and general appearance of our **visuals change accordingly**.

Any **new visual** we create in our report will follow the **same theme**.

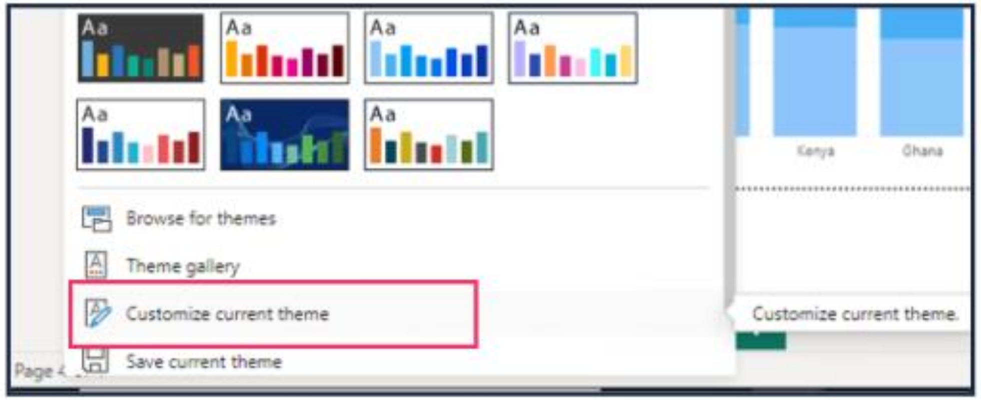




Customize a theme

We can customise a given theme by changing to our preferred formatting settings.

01. Select **View** at the top, and go to **Themes**.
02. Select **Customise current theme**.



A dialogue box with customisable theme settings appears. They are grouped into the following categories:

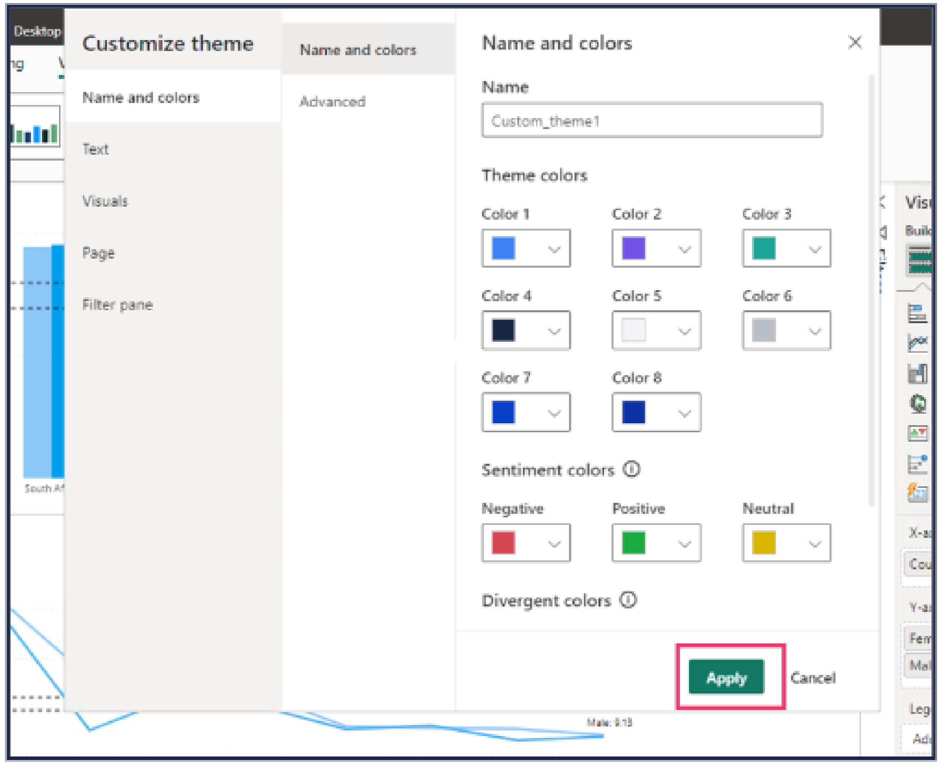
- **Name and colours:** Specify the name and choose a colour palette for the theme.
- **Text:** Change the settings for the text elements in your report such as the font family, size, and colour.
- **Visuals:** Modify visual elements such as background, border, header, and tooltips.
- **Page:** Format and style the report page itself such as the wallpaper and background.
- **Filter pane:** Adjust the appearance and layout of the filter pane.

Customize a theme

03. Click on **Apply** when we have modified all the desired formatting settings.

04. The **changes are saved and applied** to our report.

Note: It's best to choose a theme that is close to what we want so that we only make a few adjustments.



Name: Custom_theme1

Save a theme

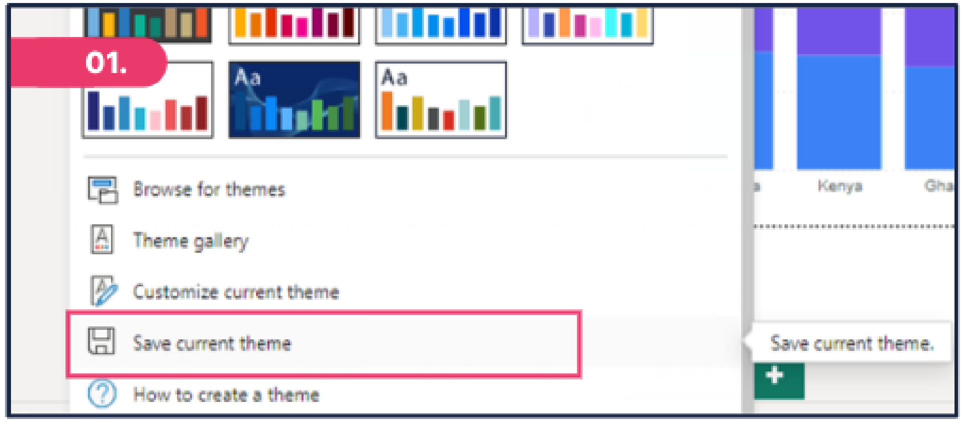
Our customised theme is only **saved to the current Power BI file.** If we close and open this file, the theme will still be available.

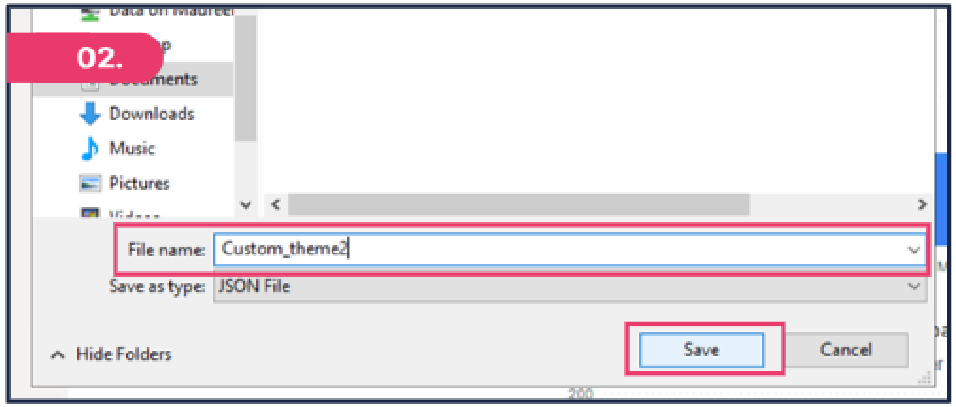
However, if we open a new file, we will not be able to access our customised theme.

To be able **to use the theme in another file**, we need to **save** it as a **JSON file** on our local machine:

01. Go to the **Themes** dropdown and select **Save current theme**.

02. Pick a name and storage location for the file, then click **Save**.





File name: Custom_theme2

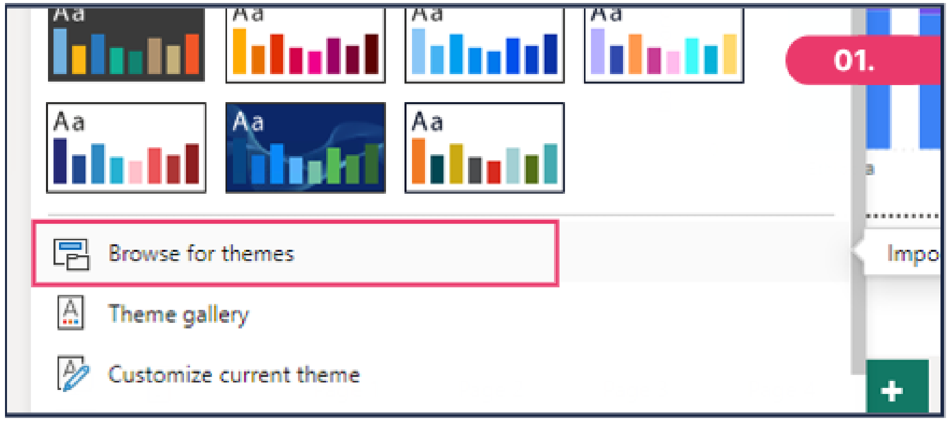
Save as type: JSON File

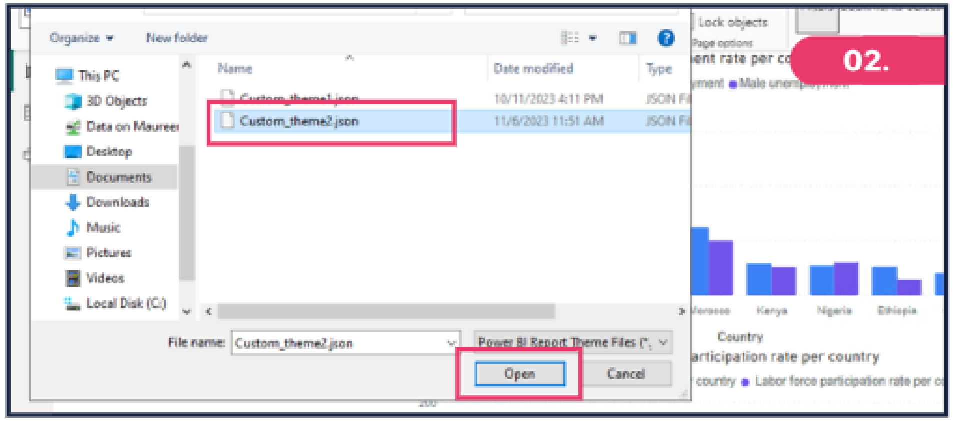
Import a theme

If we have a **saved custom theme file** in our local machine, we can **import it to the current Power BI file** using the following steps:

01. Go to **Themes**, then select **Browse for theme**.

02. **Navigate to** the saved **theme file** and upload it. The theme will then be applied to the current file.



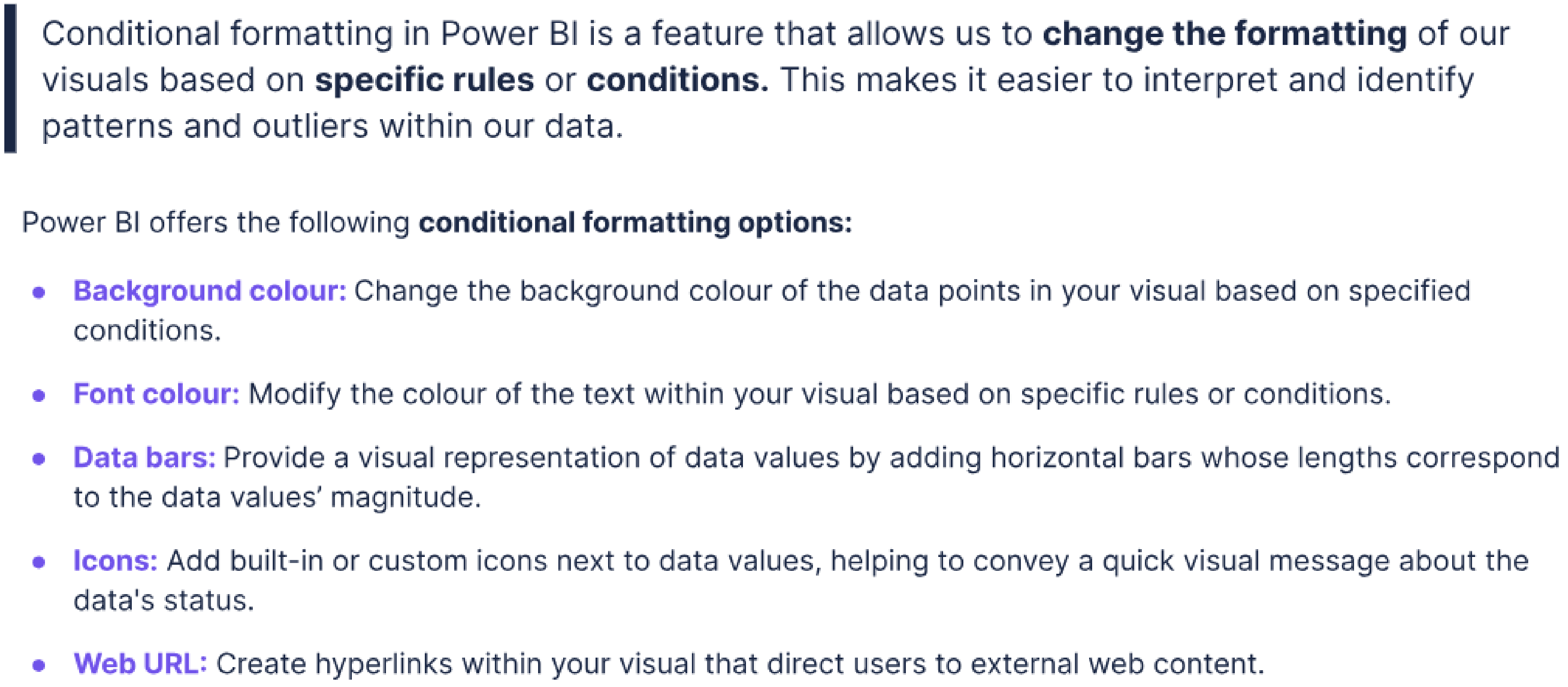


Name	Date modified	Type
Custom_theme1.json	10/11/2023 4:11 PM	JSON Fi
Custom_theme2.json	11/6/2023 11:51 AM	JSON Fi

File name: Custom_theme2.json

Power BI Report Theme Files (*.

Conditional formatting in Power BI



> Conditional formatting in Power BI is a feature that allows us to **change the formatting** of our visuals based on **specific rules** or **conditions.** This makes it easier to interpret and identify patterns and outliers within our data.

Power BI offers the following **conditional formatting options:**

- **Background colour:** Change the background colour of the data points in your visual based on specified conditions.
- **Font colour:** Modify the colour of the text within your visual based on specific rules or conditions.
- **Data bars:** Provide a visual representation of data values by adding horizontal bars whose lengths correspond to the data values' magnitude.
- **Icons:** Add built-in or custom icons next to data values, helping to convey a quick visual message about the data's status.
- **Web URL:** Create hyperlinks within your visual that direct users to external web content.

Conditional formatting in Power BI

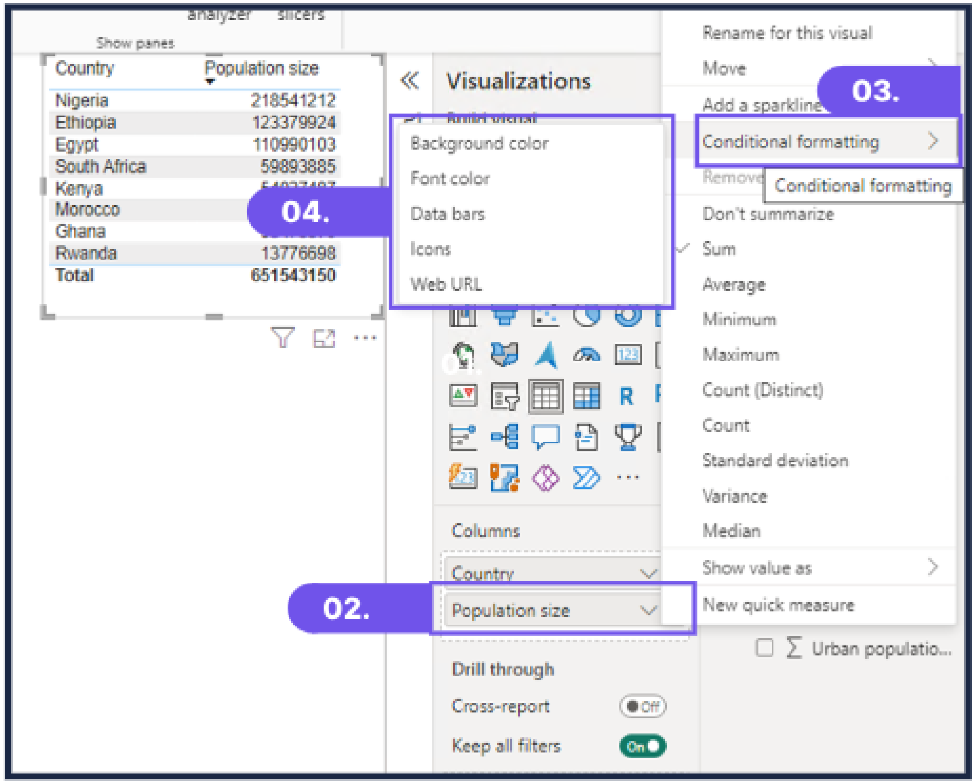
To apply conditional formatting:

01. Go to the **Build visual pane.**

02. **Go to the field** we wish to apply conditional formatting and **click on the dropdown**.

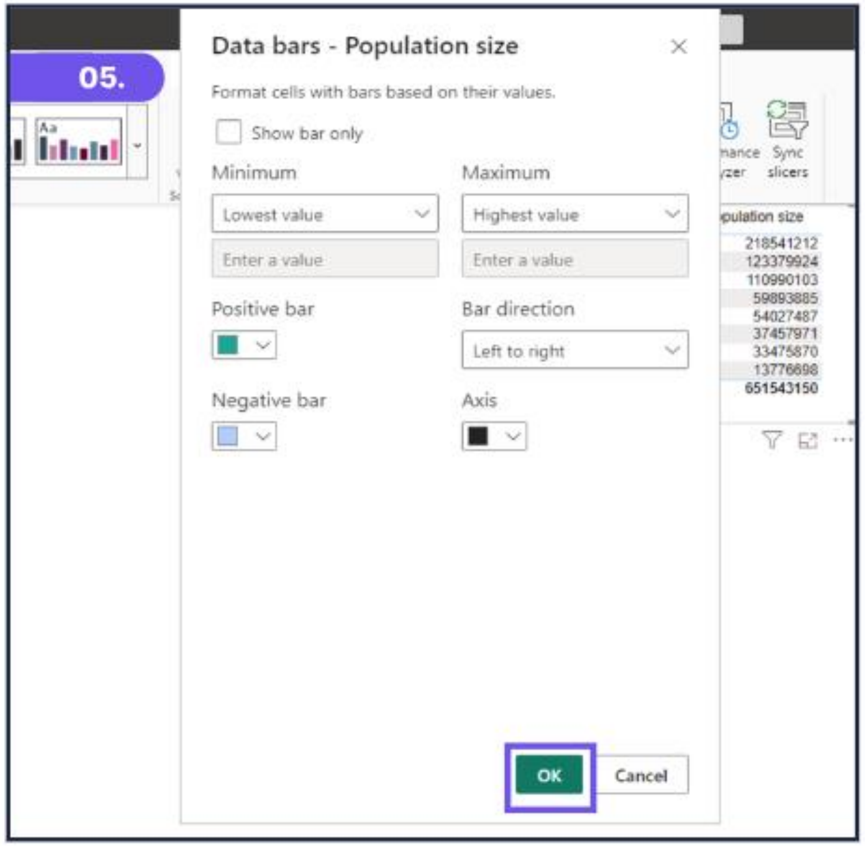
03. Select **conditional formatting.**

04. **Choose** a **formatting option**.



Country	Population size
Nigeria	218541212
Ethiopia	123379924
Egypt	110990103
South Africa	59893885
Kenya	[illegible]
Morocco	[illegible]
Ghana	[illegible]
Rwanda	13776698
Total	651543150

Conditional formatting in Power BI



Data bars - Population size

Format cells with bars based on their values.

□ Show bar only

Minimum: Lowest value

Enter a value

Maximum: Highest value

Enter a value

Positive bar: 🟩

Bar direction: Left to right

Negative bar: 🟦

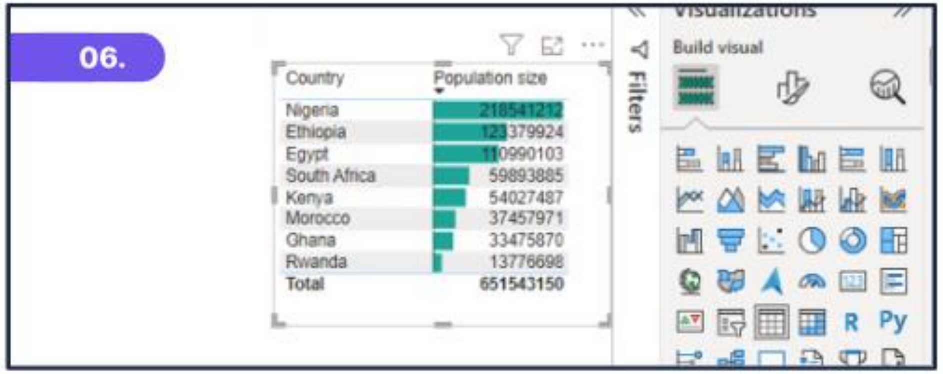
Axis: ⬛

OK   Cancel

05. A dialogue box with **various conditional formatting settings** appears. After we **set all the conditional formatting options**, we click **OK.**

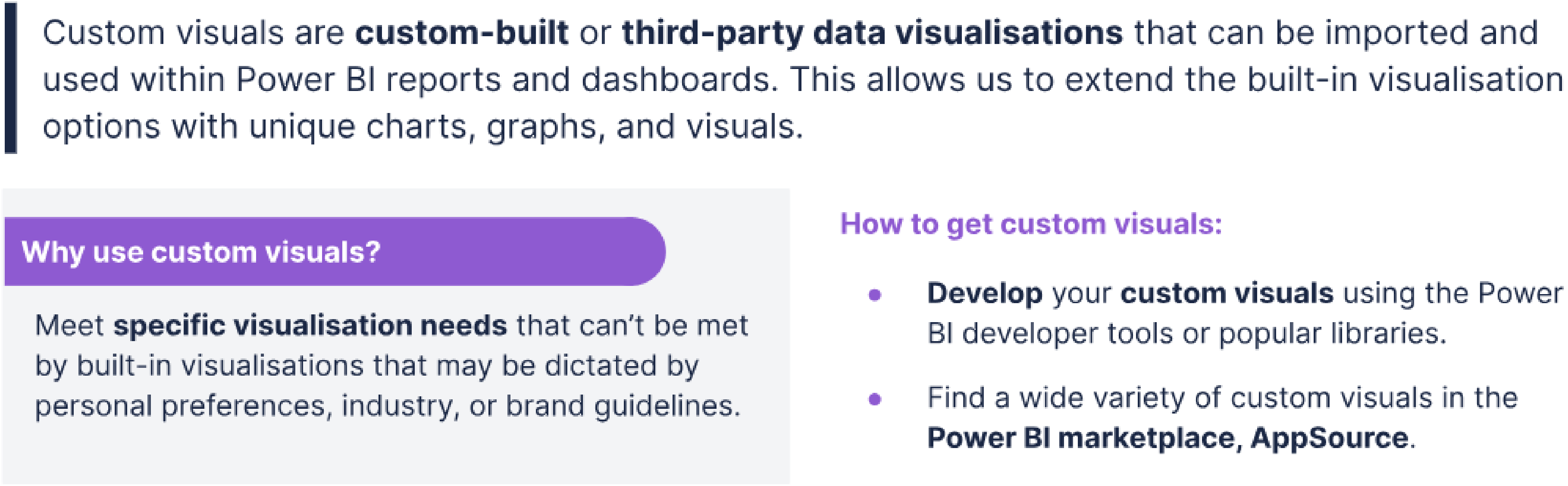
Note: These **settings depend on the formatting option we choose**.

06. The changes are applied.



Country	Population size
Nigeria	218541212
Ethiopia	123379924
Egypt	110990103
South Africa	59893885
Kenya	54027487
Morocco	37457971
Ghana	33475870
Rwanda	13776698
Total	651543150

Custom Visuals in Power BI



Custom visuals are **custom-built** or **third-party data visualisations** that can be imported and used within Power BI reports and dashboards. This allows us to extend the built-in visualisation options with unique charts, graphs, and visuals.

Why use custom visuals?

Meet **specific visualisation needs** that can't be met by built-in visualisations that may be dictated by personal preferences, industry, or brand guidelines.

How to get custom visuals:

- **Develop** your **custom visuals** using the Power BI developer tools or popular libraries.
- Find a wide variety of custom visuals in the **Power BI marketplace, AppSource**.

Custom Visuals in Power BI

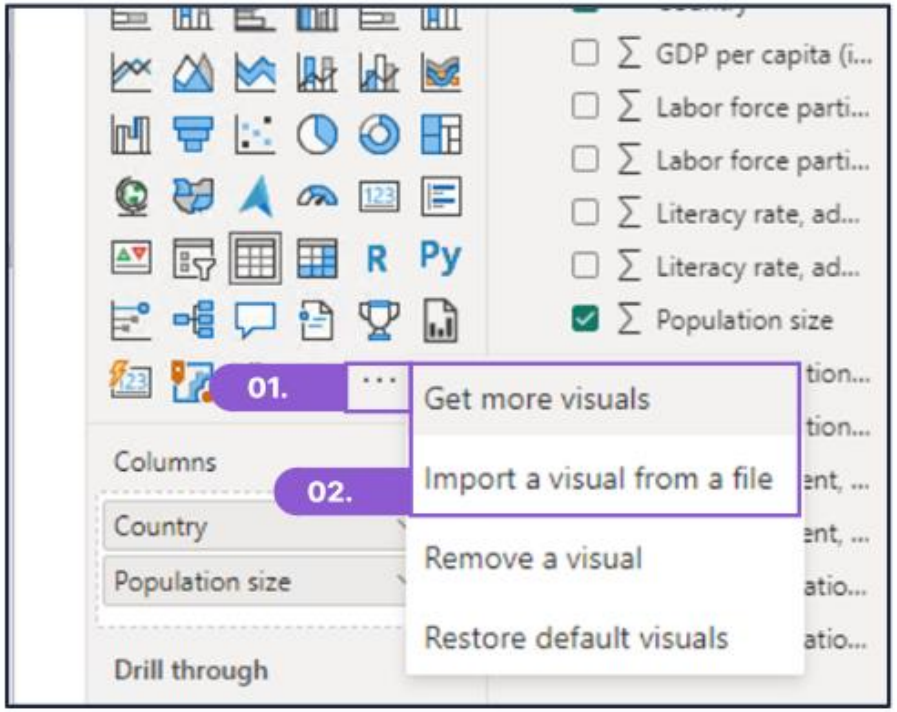
How to import custom visuals:

01. Go to the **Visualizations** pane and select the **ellipsis**.

02. Select **Get more visuals** from the menu. This will direct us to AppSource where we choose among the available visuals and import.

Or select **Import a visual from a file** from the menu. This will direct us to our local machine where we choose a *.pbiviz* file and import it.

03. Once imported, a new icon appears on the Visualizations pane and we can use the custom visual like any built-in visual.



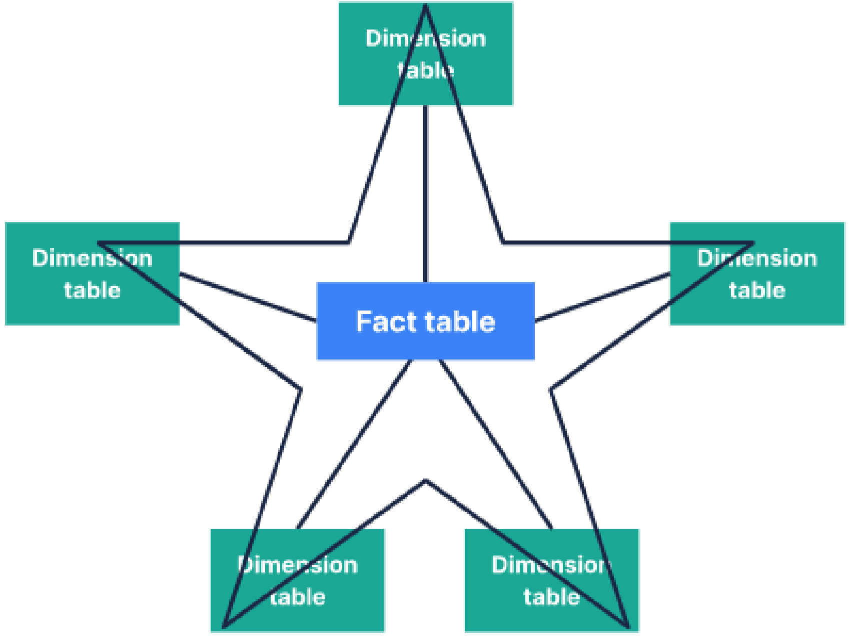
Note: You will **need a Microsoft account** in order to use the Get more visuals option.

The star schema in Power BI

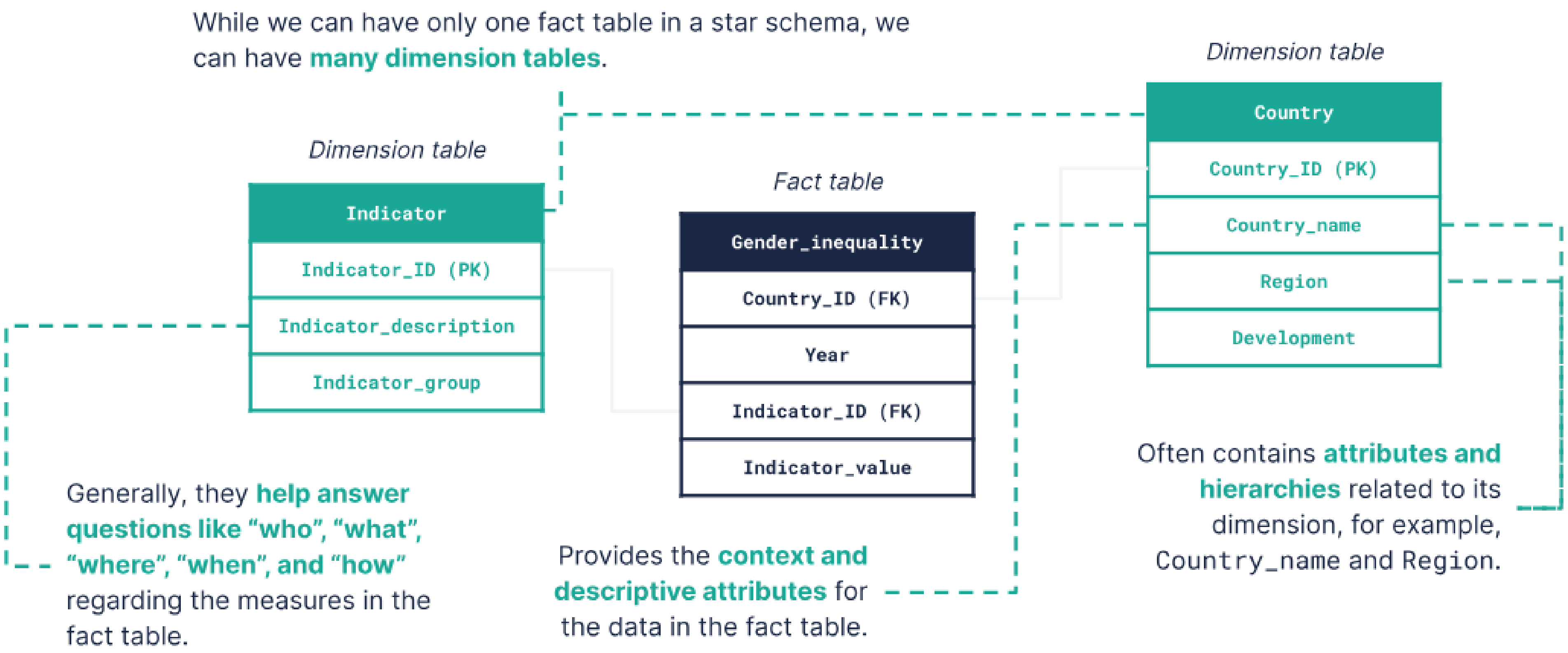
> As in SQL, a **variety of data models** can be implemented in Power BI. However, the most commonly used data modelling approach in Power BI is the **star schema**.

A star schema is **a specific type of data modelling approach** commonly used in Power BI and other data warehouse and business intelligence systems. It is characterised by a **central fact table** connected to multiple **dimension tables**, forming a star-like structure when visualised graphically.

The star schema is popular because it's **simple**, **optimised for query performance**, **scalable**, **easy to maintain**, and **consistent**. In other words, it ticks all of the boxes of being a "good" data model.



The star schema in Power BI



While we can have only one fact table in a star schema, we can have **many dimension tables**.

Dimension table

Indicator
Indicator_ID (PK)
Indicator_description
Indicator_group

Fact table

Gender_inequality
Country_ID (FK)
Year
Indicator_ID (FK)
Indicator_value

Dimension table

Country
Country_ID (PK)
Country_name
Region
Development

Generally, they **help answer questions like "who", "what", "where", "when", and "how"** regarding the measures in the fact table.

Provides the **context and descriptive attributes** for the data in the fact table.

Often contains **attributes and hierarchies** related to its dimension, for example, Country_name and Region.

Defining relationships in Power BI

> As in relational databases, relationships are essential in Power BI for **connecting and associating data from different tables** in a data model.

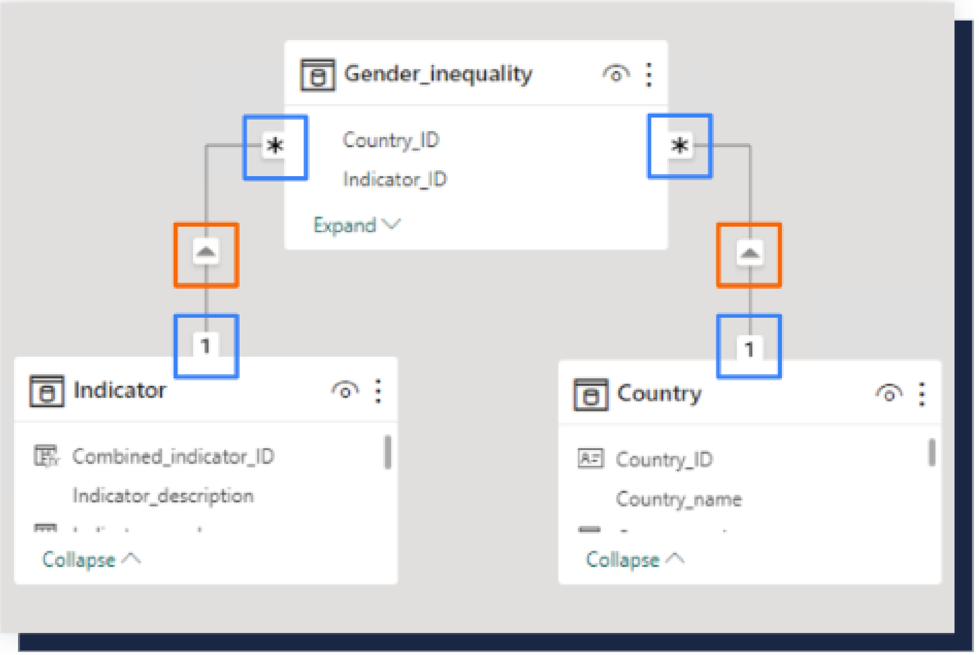
When we define relationships in Power BI, we need to consider the following:

Cardinality

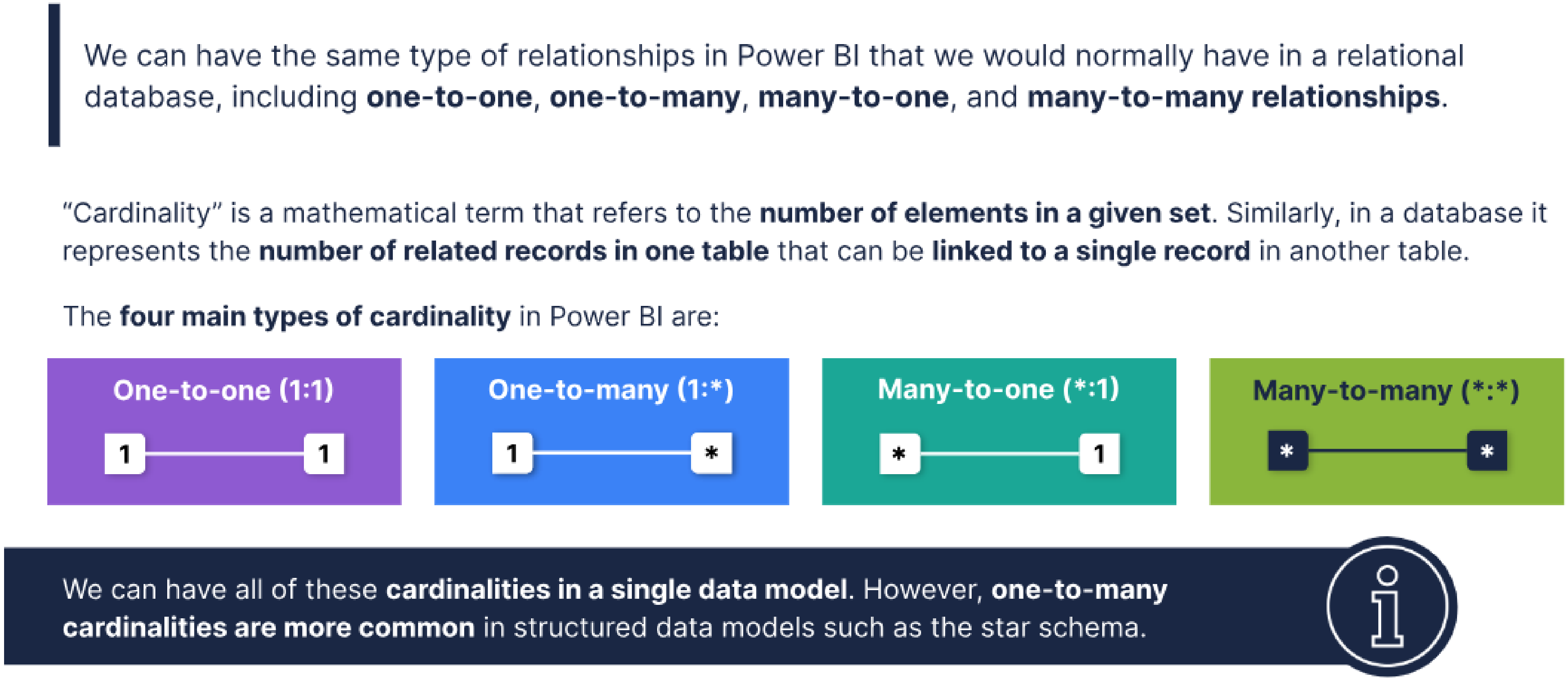
The **number of related records** in one table that can be linked to a single record in another table.

Directionality

Defines **how filtering** in one table affects another table.



Cardinality



> We can have the same type of relationships in Power BI that we would normally have in a relational database, including **one-to-one**, **one-to-many**, **many-to-one**, and **many-to-many relationships**.

"Cardinality" is a mathematical term that refers to the **number of elements in a given set**. Similarly, in a database it represents the **number of related records in one table** that can be **linked to a single record** in another table.

The **four main types of cardinality** in Power BI are:

One-to-one (1:1)

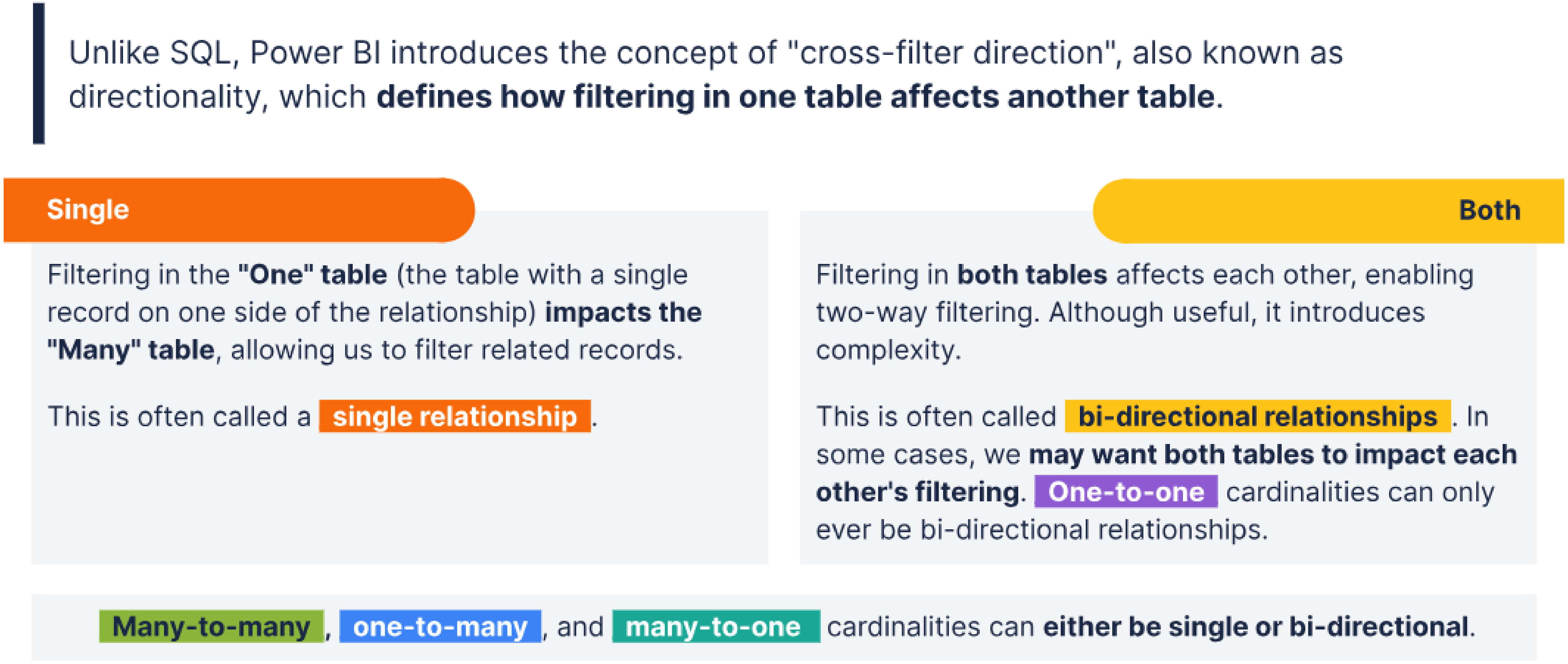
One-to-many (1:*)

Many-to-one (*:1)

Many-to-many (*:*)

We can have all of these **cardinalities in a single data model**. However, **one-to-many cardinalities are more common** in structured data models such as the star schema.

Directionality



> Unlike SQL, Power BI introduces the concept of "cross-filter direction", also known as directionality, which **defines how filtering in one table affects another table.**

Single

Filtering in the **"One" table** (the table with a single record on one side of the relationship) **impacts the "Many" table**, allowing us to filter related records.

This is often called a **single relationship**.

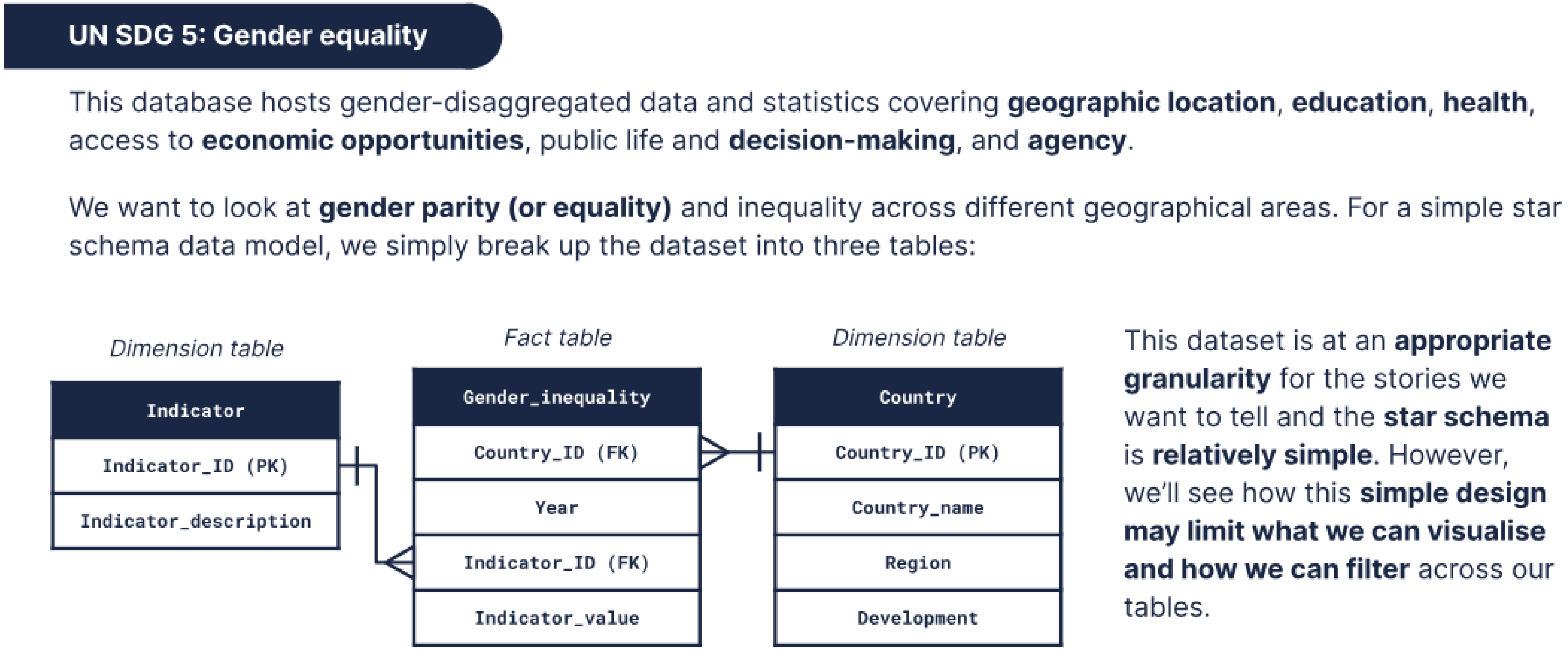
Both

Filtering in **both tables** affects each other, enabling two-way filtering. Although useful, it introduces complexity.

This is often called **bi-directional relationships**. In some cases, we **may want both tables to impact each other's filtering**. **One-to-one** cardinalities can only ever be bi-directional relationships.

Many-to-many, **one-to-many**, and **many-to-one** cardinalities can **either be single or bi-directional**.

Dataset example



UN SDG 5: Gender equality

This database hosts gender-disaggregated data and statistics covering **geographic location**, **education**, **health**, access to **economic opportunities**, public life and **decision-making**, and **agency**.

We want to look at **gender parity (or equality)** and inequality across different geographical areas. For a simple star schema data model, we simply break up the dataset into three tables:

Dimension table

Indicator
Indicator_ID (PK)
Indicator_description

Fact table

Gender_inequality
Country_ID (FK)
Year
Indicator_ID (FK)
Indicator_value

Dimension table

Country
Country_ID (PK)
Country_name
Region
Development

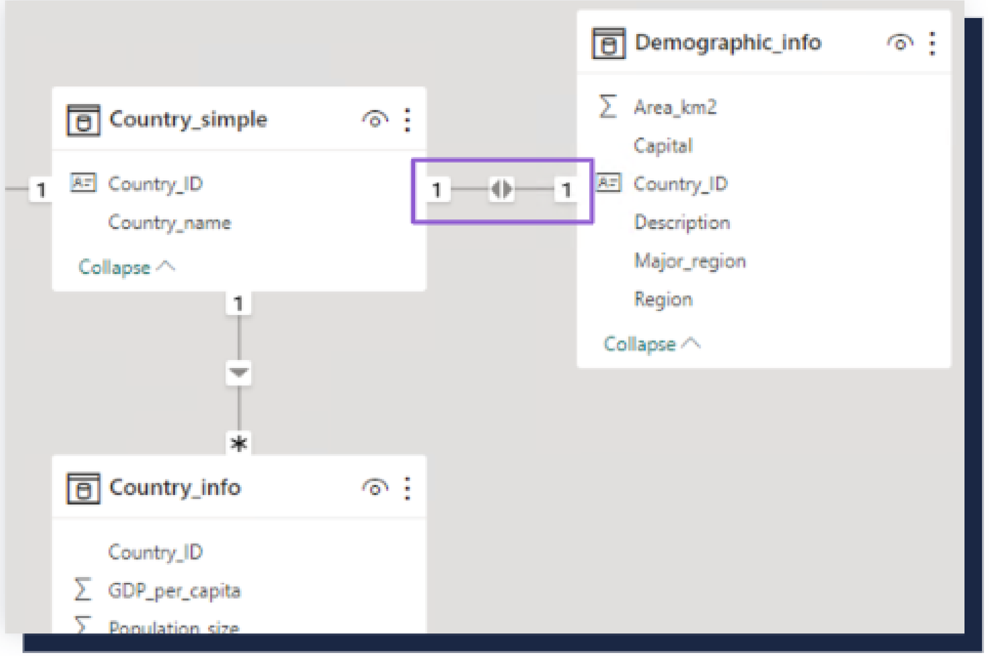
This dataset is at an **appropriate granularity** for the stories we want to tell and the **star schema** is **relatively simple**. However, we'll see how this **simple design may limit what we can visualise and how we can filter** across our tables.

One-to-one(1:1)

> In a one-to-one cardinality, **each record in the first table** is associated with **one and only one record in the second table**, and vice versa.

As an example, we could move all of our geographical information per country (that doesn't change over time) to a **dedicated table called `Demographic_info`** that we can **enrich** with additional information.

This new table would then have a **one-to-one cardinality** with the `Country_simple` table because **one country is related to single description**, **region**, **area**, etc.



One-to-many(1:*)

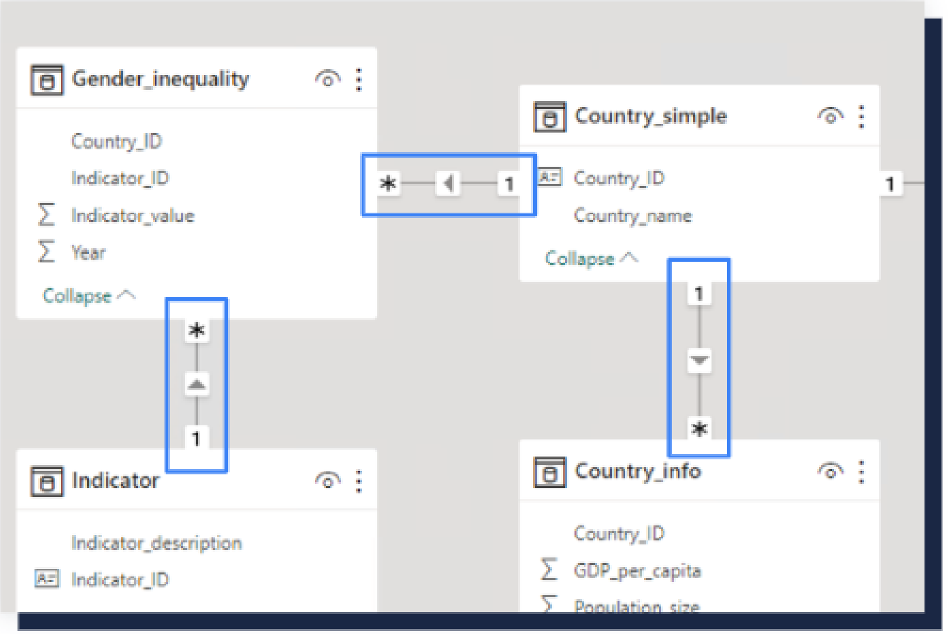
> In a one-to-many cardinality, **each record in the first table can be related to multiple records in the second table**, but each record in the second table is associated with one record in the first table.

Here we have **three** **one-to-many** relationships:

01. A **single country** can have **multiple records for different years and inequality indicators**, but each record in `Gender_inequality` is related to one specific country.

02. A **single country** can have **multiple records for different years and GDP per capita**, **population size**, etc. combinations, but each record in `Country_info` is related to one specific country.

03. A **single indicator** can have **multiple records for different years and countries**, but each record in `Gender_inequality` is related to one specific indicator.

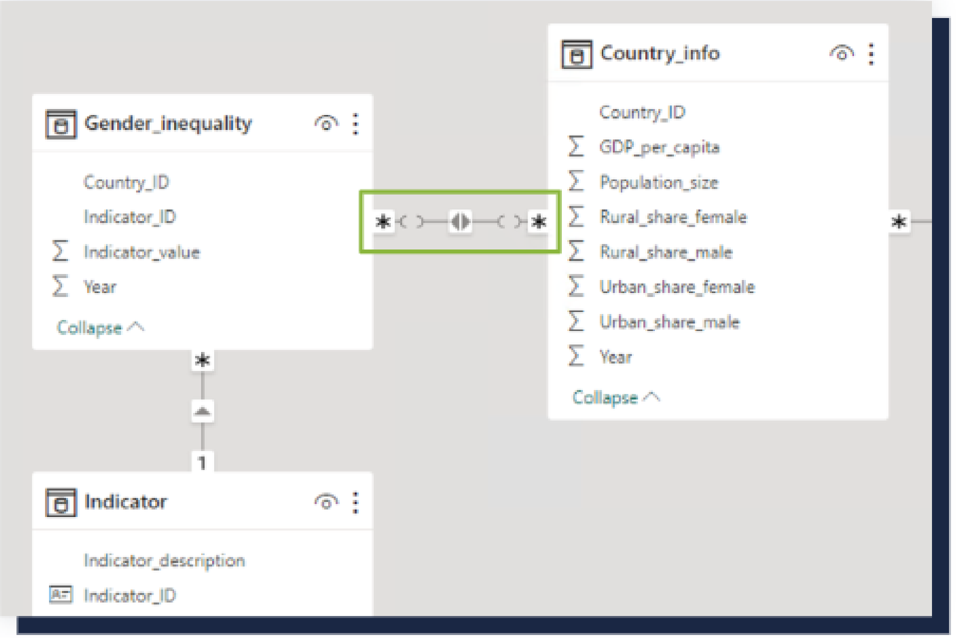


Many-to-many(*:*)

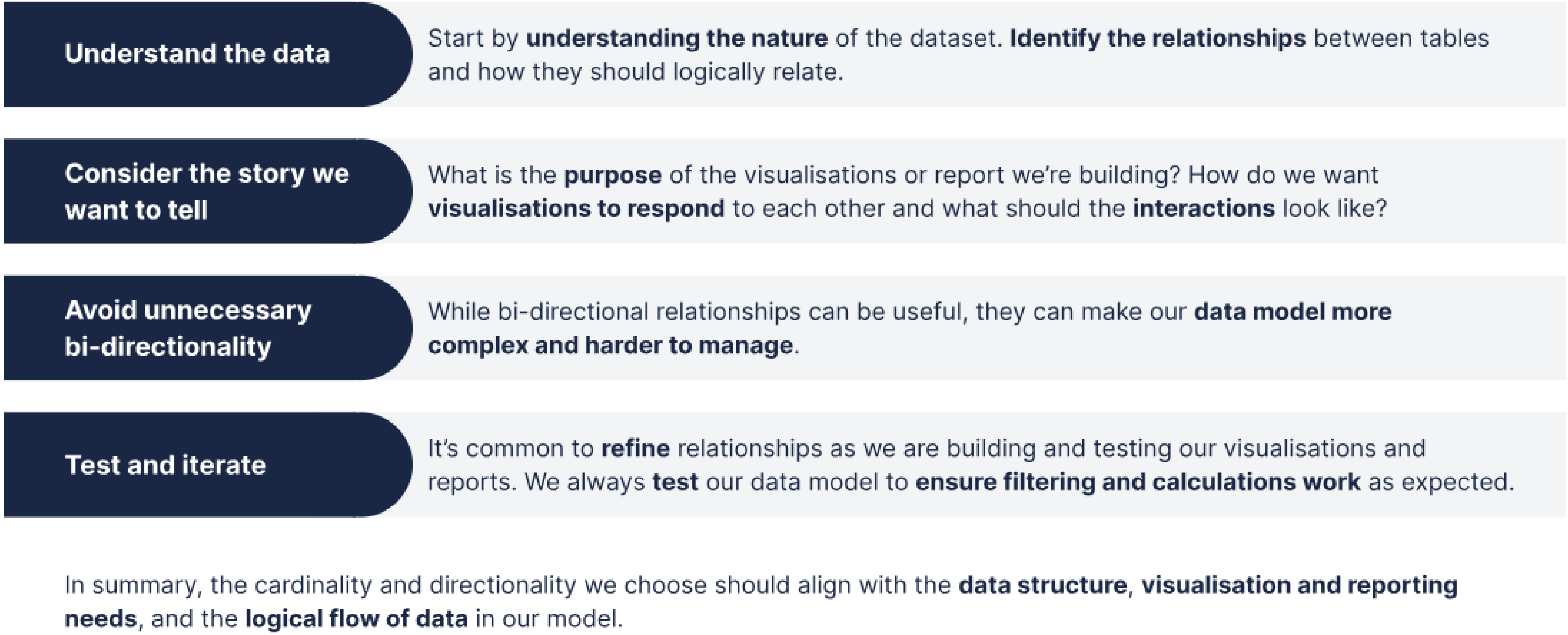
> In a many-to-many cardinality, **multiple records in the first table can be associated with multiple records in the second** table.

Since the Country_simple table **only includes two features**, namely Country_ID and Country_name, we could easily get rid of the table and simply move Country_name to the Country_info table.

Now we'll need a **many-to-many** cardinality since both Gender_inequality and Country_info have **multiple instances of the same country record**.



How to choose the cardinality and directionality



Understand the data

Start by **understanding the nature** of the dataset. **Identify the relationships** between tables and how they should logically relate.

Consider the story we want to tell

What is the **purpose** of the visualisations or report we're building? How do we want **visualisations to respond** to each other and what should the **interactions** look like?

Avoid unnecessary bi-directionality

While bi-directional relationships can be useful, they can make our **data model more complex and harder to manage**.

Test and iterate

It's common to **refine** relationships as we are building and testing our visualisations and reports. We always **test** our data model to **ensure filtering and calculations work** as expected.

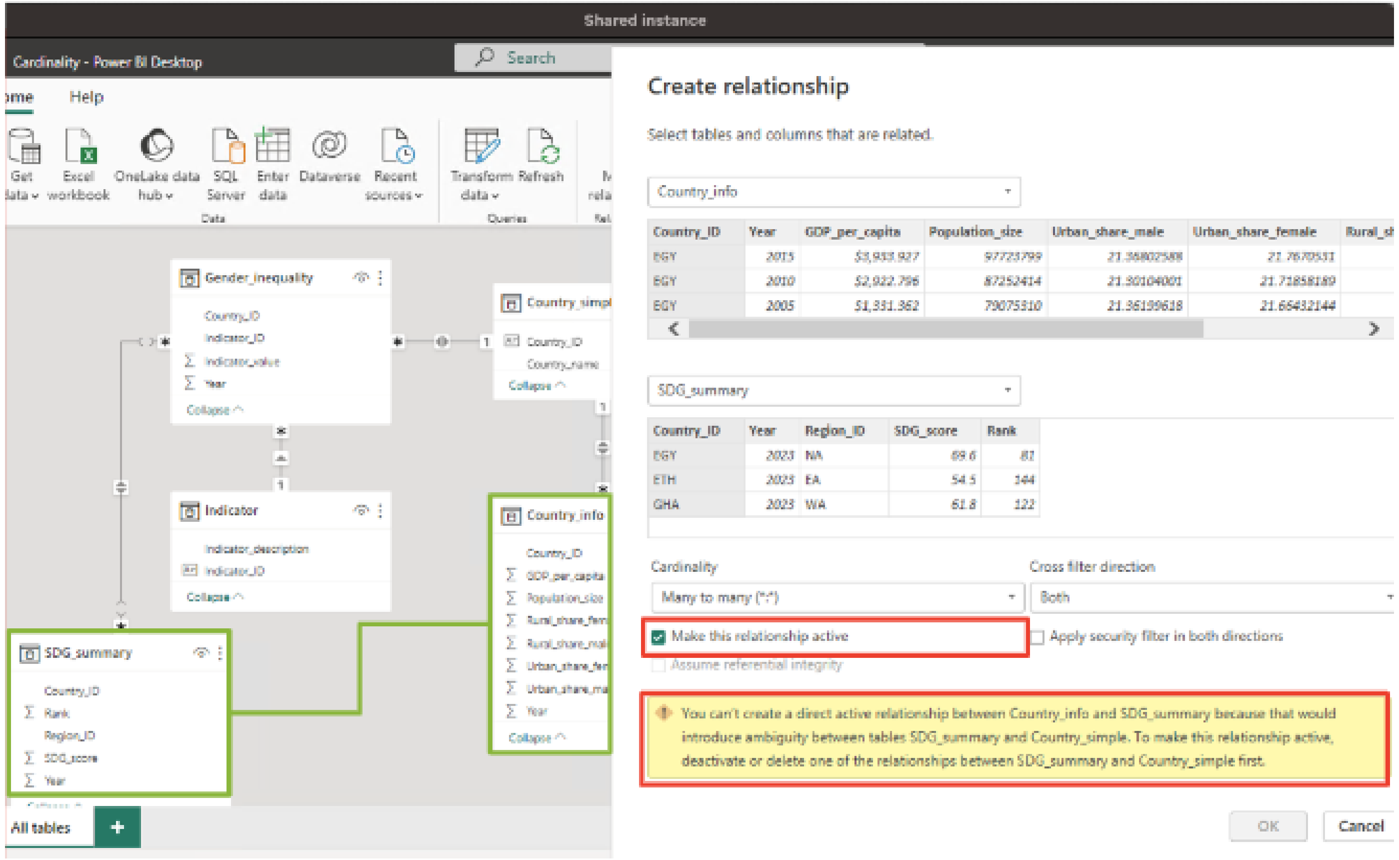
In summary, the cardinality and directionality we choose should align with the **data structure**, **visualisation and reporting needs**, and the **logical flow of data** in our model.

Ambiguous paths

Ambiguous paths often occur when our **data model becomes more complex**.

It also very often occurs when we try to activate unnecessary **many-to-many** relationships. Power BI will **always warn us on many-to-many relationships**.

Luckily, Power BI will most often **not allow us to make a relationship active** that could potentially cause ambiguous paths.



Create relationship

Select tables and columns that are related.

Country_info

Country_ID	Year	GDP_per_capita	Population_size	Urban_share_male	Urban_share_female	Rural_sh
EGY	2015	$3,933.927	97723799	21.36802588	21.7670531	
EGY	2010	$2,922.796	87252414	21.30104001	21.71858189	
EGY	2005	$1,331.362	79075310	21.36199618	21.66432144	

SDG_summary

Country_ID	Year	Region_ID	SDG_score	Rank
EGY	2023	NA	69.6	81
ETH	2023	EA	54.5	144
GHA	2023	WA	61.8	122

Cardinality: Many to many (*:*)

Cross filter direction: Both

☑ Make this relationship active

□ Apply security filter in both directions

□ Assume referential integrity

You can't create a direct active relationship between Country_info and SDG_summary because that would introduce ambiguity between tables SDG_summary and Country_simple. To make this relationship active, deactivate or delete one of the relationships between SDG_summary and Country_simple first.

OK | Cancel

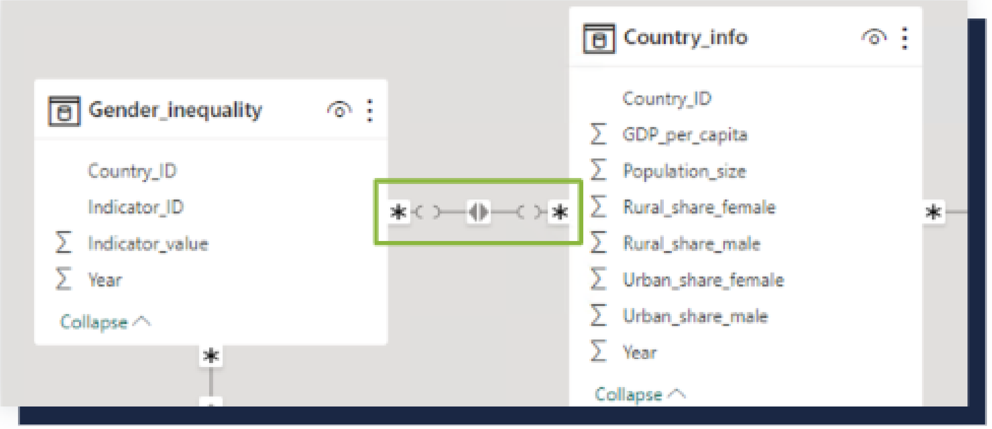
How to avoid challenging relationships?

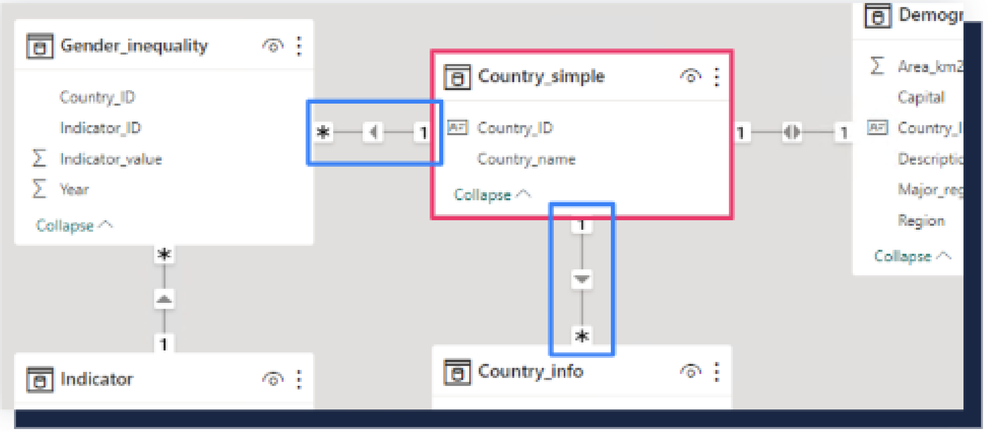
We've seen that many-to-many relationships often contribute to the common challenges we see in Power BI.

A simple **alternative to many-to-many relationships** is to use a **bridge table**, also known as a **junction table** or **association table**.

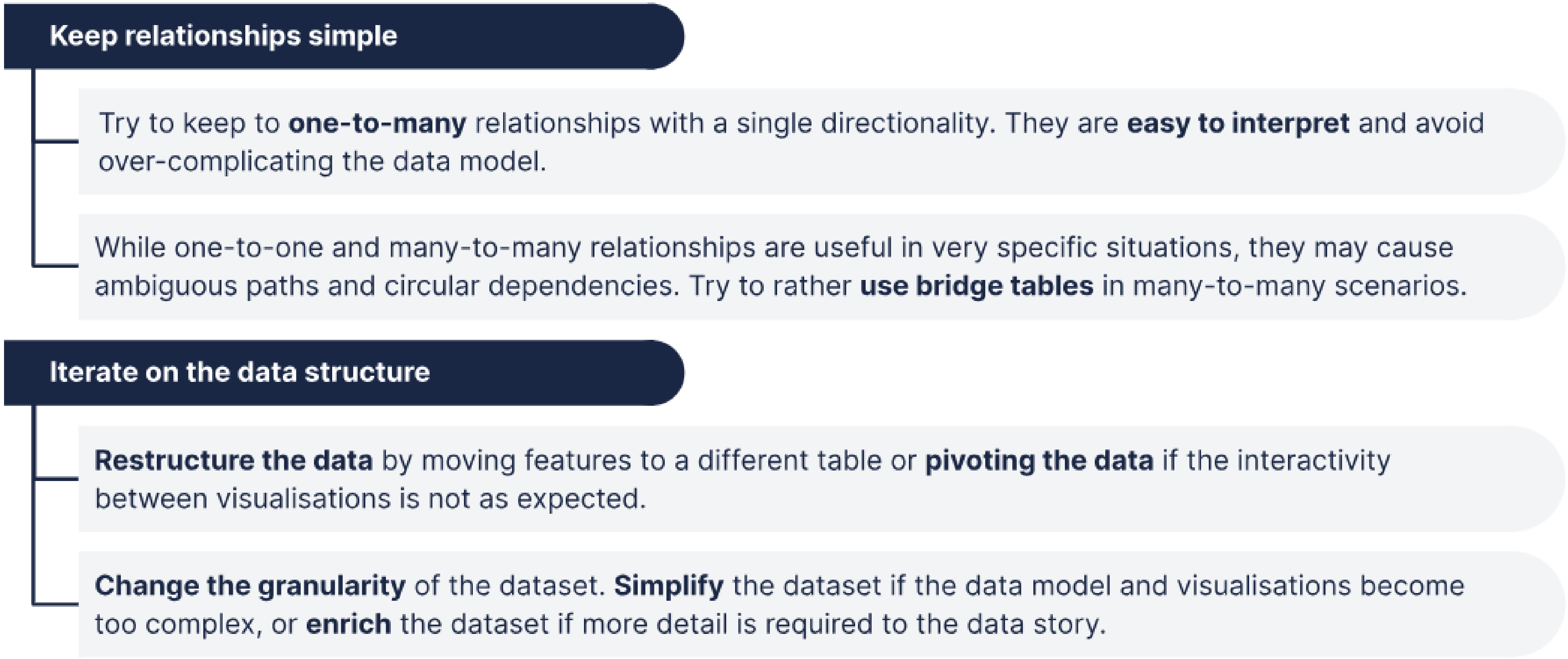
Think back to the **many-to-many** example we had and compare that to the **one-to-many** example.

So, we've technically already seen an example of a **bridge table** in action. We've **replaced the many-to-many relationship with two one-to-many** relationships by introducing the `Country_simple` table.





Tips for relationships in Power BI



Keep relationships simple

- Try to keep to **one-to-many** relationships with a single directionality. They are **easy to interpret** and avoid over-complicating the data model.
- While one-to-one and many-to-many relationships are useful in very specific situations, they may cause ambiguous paths and circular dependencies. Try to rather **use bridge tables** in many-to-many scenarios.

Iterate on the data structure

- **Restructure the data** by moving features to a different table or **pivoting the data** if the interactivity between visualisations is not as expected.
- **Change the granularity** of the dataset. **Simplify** the dataset if the data model and visualisations become too complex, or **enrich** the dataset if more detail is required to the data story.

Naming conventions

User-friendly naming conventions **enhance the readability of a data model**. There are several ways to rename columns and queries in the Power Query Editor. Using the "**Transform**" tab is one method.

01.

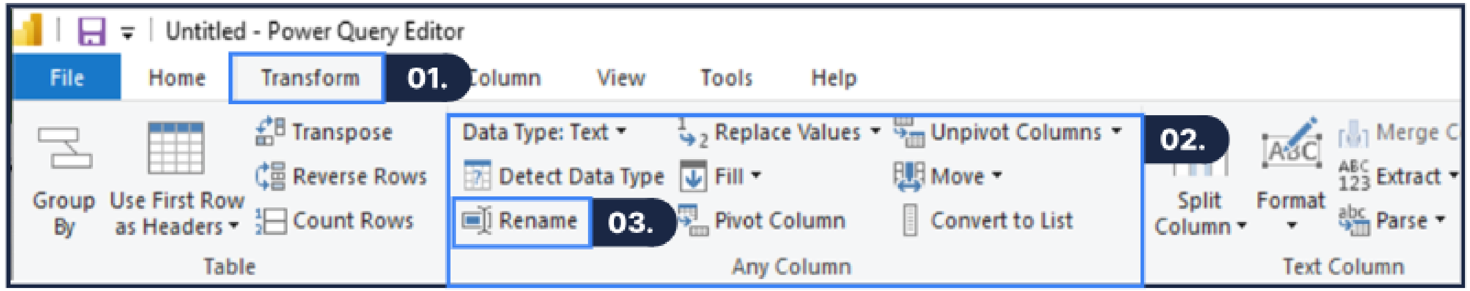
Click on the "**Transform**" tab.

02.

Navigate to the "**Any column**" section which contains transformations at the column level for any data type column.

03.

Select "**Rename**".

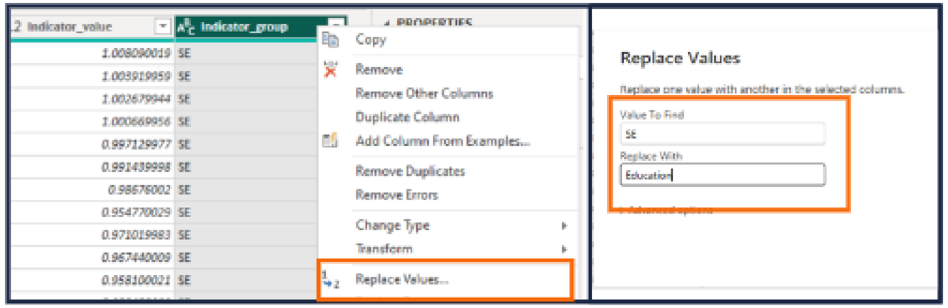


Resolve data quality issues

Power Query Editor enables us to address various data quality issues, such as inconsistencies, unexpected values, and null values by either **replacing** or **deleting** them.

Replace values

- Right-click on the column name.
- Select “**Replace Values**”.
- Enter the “**Value To Find**” and the value to “**Replace With**”.



Indicator_value	Indicator_group
1.008090019	SE
1.003919959	SE
1.002679944	SE
1.000669956	SE
0.997129977	SE
0.991439998	SE
0.98676002	SE
0.954770029	SE
0.971019983	SE
0.967440009	SE
0.958100021	SE

Replace Values

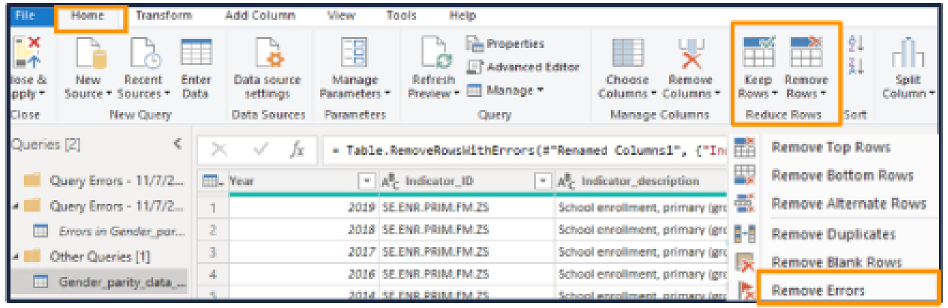
Replace one value with another in the selected columns.

Value To Find: SE

Replace With: Education

Reduce rows

- Select the column name.
- Navigate to the “**Home**” tab.
- Select “**Remove Rows**” under the “**Reduce Rows**” section.
- Select “**Remove Errors**” in the dropdown menu.



```
= Table.RemoveRowsWithErrors(#"Renamed Columns1", {"In
```

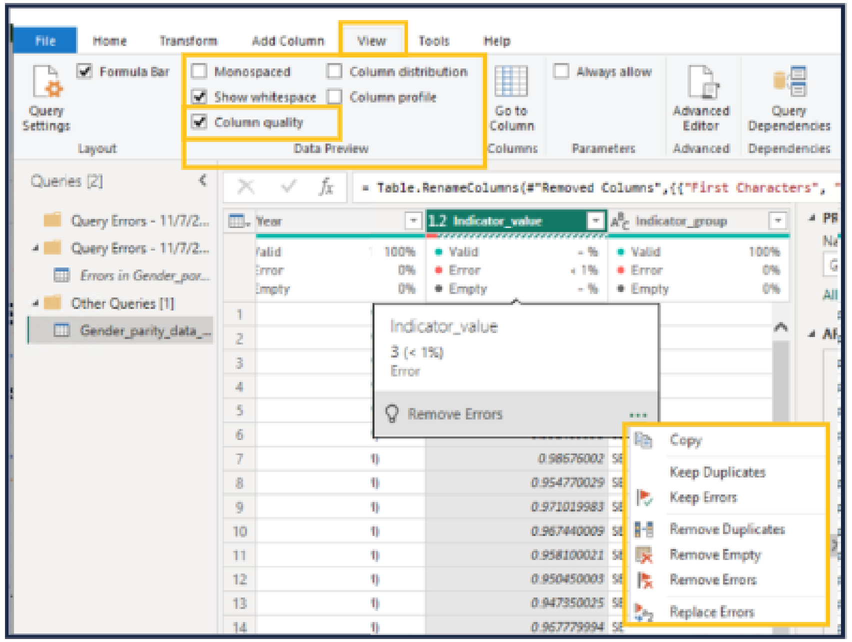
	Year	Indicator_ID	Indicator_description
1	2019	SE.ENR.PRIM.FM.ZS	School enrollment, primary (grc
2	2018	SE.ENR.PRIM.FM.ZS	School enrollment, primary (grc
3	2017	SE.ENR.PRIM.FM.ZS	School enrollment, primary (grc
4	2016	SE.ENR.PRIM.FM.ZS	School enrollment, primary (grc
5	2014	SE.ENR.PRIM.FM.ZS	School enrollment, primary (grc

Resolve data quality issues

> The "**View**" tab in Power Query Editor also allows us to see the **column quality** of each column in the dataset and resolve any data quality issues.

Column quality view

- Right-click on the column name.
- Navigate to the “**View**” tab.
- Select the “**Column quality**” check box in the “**Data Preview**” section.
- Select the appropriate data quality solution.



☑ Formula Bar

□ Monospaced

☑ Show whitespace

☑ Column quality

□ Column distribution

□ Column profile

□ Always allow

```
= Table.RenameColumns(#"Removed Columns",{{"First Characters", "
```

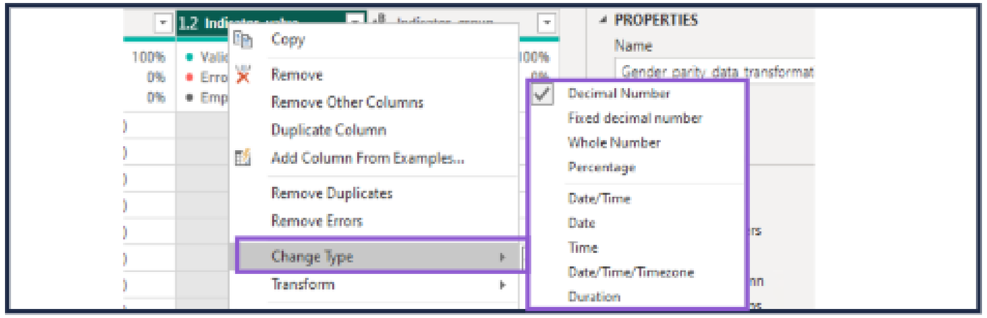
Indicator_value
3 (< 1%)
Error

Evaluate and transform data types

> We can **evaluate and change data types** to ensure data consistency. In the Power Query Editor, there are multiple ways to transform data types.

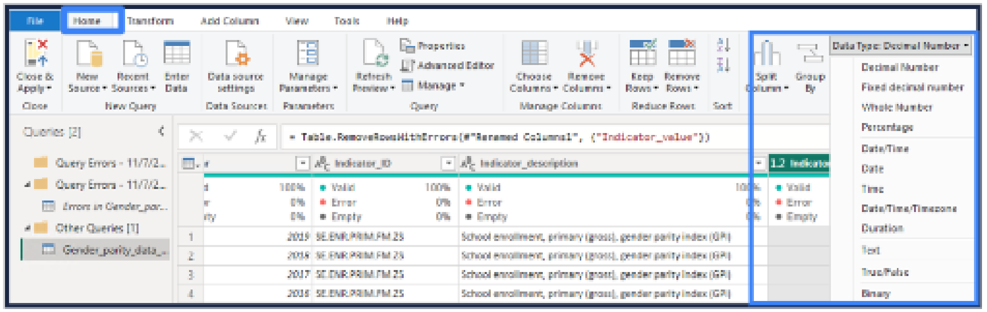
Right-click method

- Right-click on the column name.
- Select "**Change Type**".
- Select the appropriate data type from the dropdown menu.



Transform section method

- Click on the "**Home**" tab.
- Navigate to the "**Transform**" section, which contains general transformations at the column level.
- Select the appropriate data type from the dropdown menu.



```
= Table.RemoveRowsWithErrors(#"Renamed Columns1", {"Indicator_value"})
```

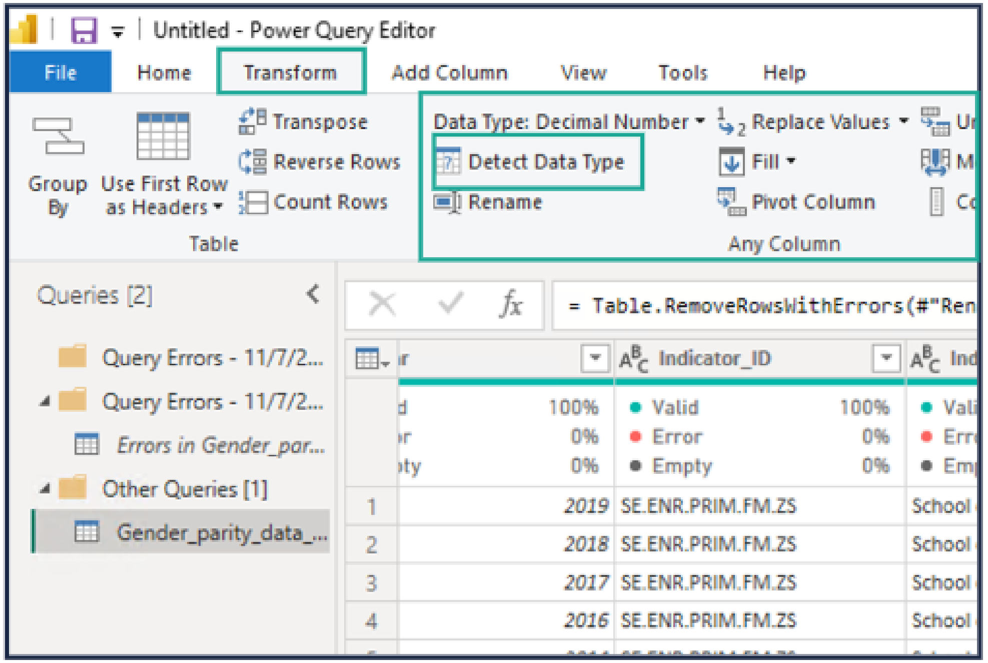
		Indicator_ID	Indicator_description
1	2019	SE.ENR.PRIM.FM.ZS	School enrollment, primary (gross), gender parity index (GPI)
2	2018	SE.ENR.PRIM.FM.ZS	School enrollment, primary (gross), gender parity index (GPI)
3	2017	SE.ENR.PRIM.FM.ZS	School enrollment, primary (gross), gender parity index (GPI)
4	2016	SE.ENR.PRIM.FM.ZS	School enrollment, primary (gross), gender parity index (GPI)

Evaluate and transform data types

> We can also use the Power Query Editor to identify data types automatically based on the values in the column.

Detect the data type

- Select the relevant column.
- Navigate to the "**Transform**" tab.
- Select the "**Detect Data Type**" option in the "**Any Column**" section.



```
= Table.RemoveRowsWithErrors(#"Ren
```

	r	Indicator_ID	Ind
	d 100% r 0% ty 0%	● Valid 100% ● Error 0% ● Empty 0%	● Vali ● Err ● Em
1	2019	SE.ENR.PRIM.FM.ZS	School
2	2018	SE.ENR.PRIM.FM.ZS	School
3	2017	SE.ENR.PRIM.FM.ZS	School
4	2016	SE.ENR.PRIM.FM.ZS	School

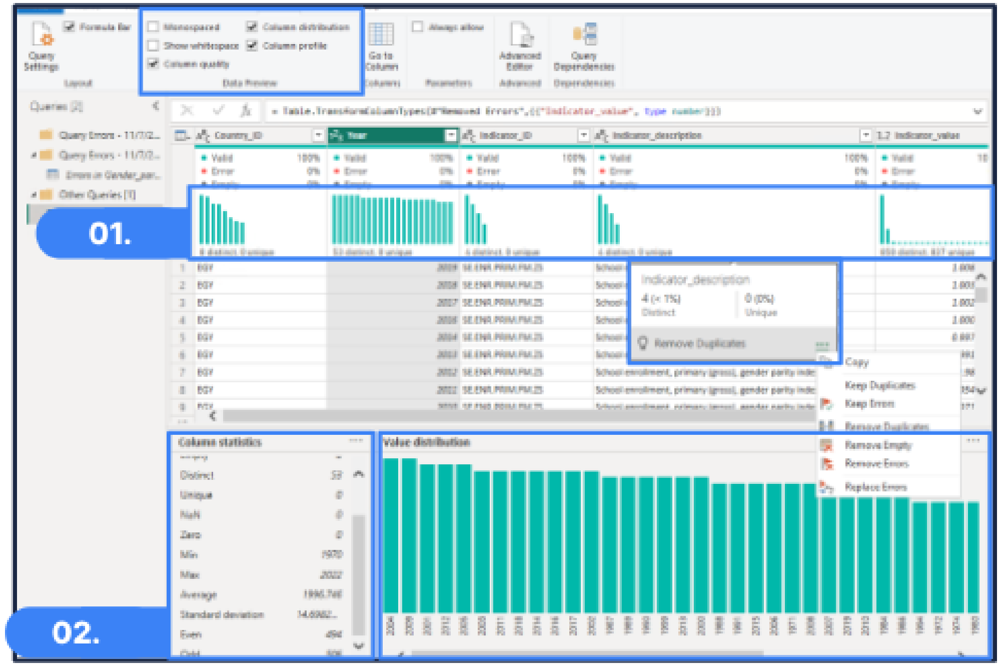
Data profiling

> Profiling data helps us understand its characteristics. We can use profiling to **identify outliers**, **missing values**, and **data distributions**. Power Query Editor provides summary statistics and histograms to help us achieve this.

Navigate to the "**View**" tab. Under the "**Data Preview**" section, select "**Column quality**", "**Column distribution**", and "**Column profile**".

01. Column distribution: Displays a **simple distribution** of the data within the column, the **number of distinct column values,** and how many of those are **unique**. Running our mouse over this graphic will provide us with some options for eliminating duplicate values.

02. Column profile: Based on the selected column, we may zoom into a single column and examine some column summary statistics such as **minimum**, **maximum**, **average**, **standard deviation**, etc. We can also see how the values in that column are **distributed**.



Apply data shape transformation

Transforming data shapes involves grouping, pivoting, unpivoting, or transposing data. The Power Query Editor allows us to achieve this under the “**Transform**” tab.

01.

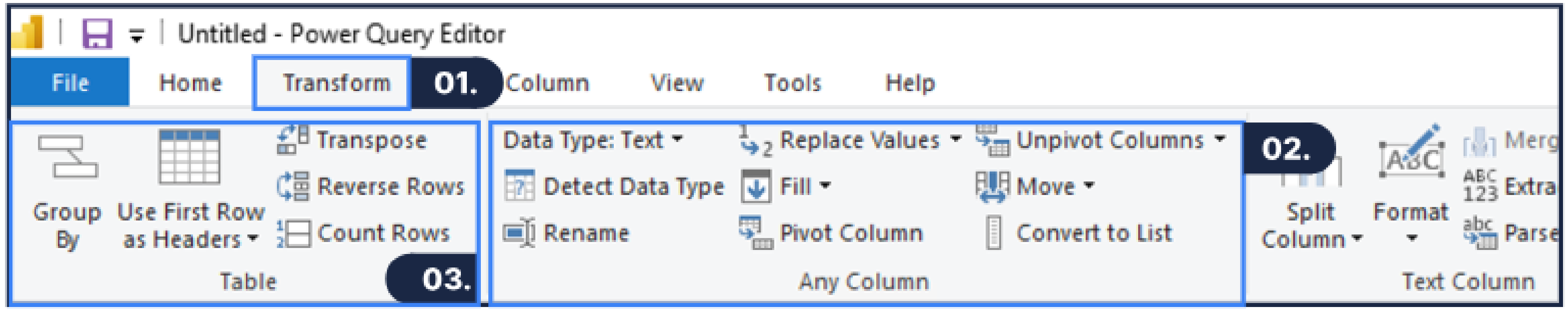
Navigate to the “**Transform**” tab.

02.

The "**Any Column**" feature allows us to pivot or unpivot columns to change the shape of our dataset as well as move columns around.

03.

The "**Table**" section allows us to group data, transpose and rearrange rows.



Group by

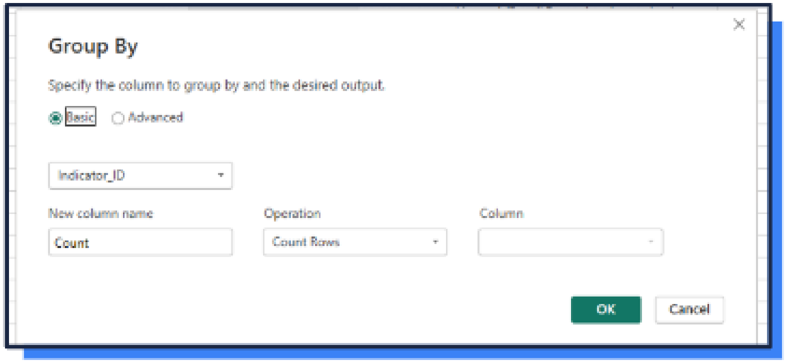
To group data by a specific column, we use the group by option which is either available in the "**Table**" section under the "**Transform**" tab or in the "**Home**" tab under the "**Transform**" section.

Basic:

Under the basic settings, we can **select the column to group by**, as well as the **operation to use to group** the column and the name of the grouped column.

Advanced:

Under the advanced settings, we can **select multiple columns to group by**. Selecting the "Add grouping" button will allow us to choose a new grouping column. We can also use **multiple aggregations for our groupings** by selecting the "Add aggregations" button.



Group By

Specify the column to group by and the desired output.

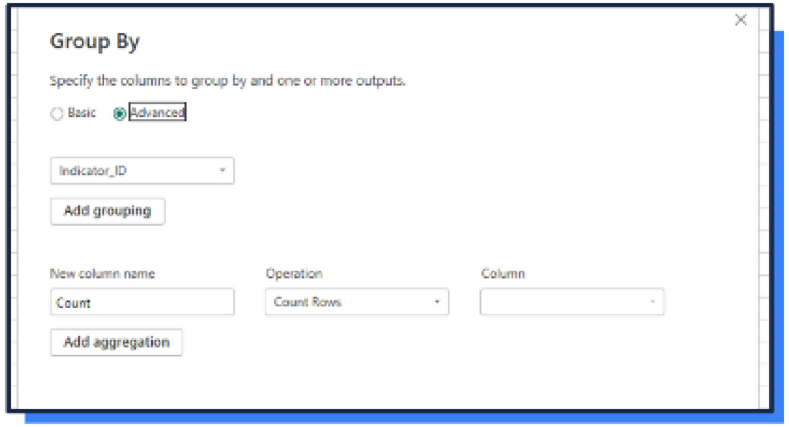
● Basic ○ Advanced

Indicator_ID

New column name: Count

Operation: Count Rows

Column:



Group By

Specify the columns to group by and one or more outputs.

○ Basic ● Advanced

Indicator_ID

New column name: Count

Operation: Count Rows

Column:

Transposing, pivoting, and unpivoting data

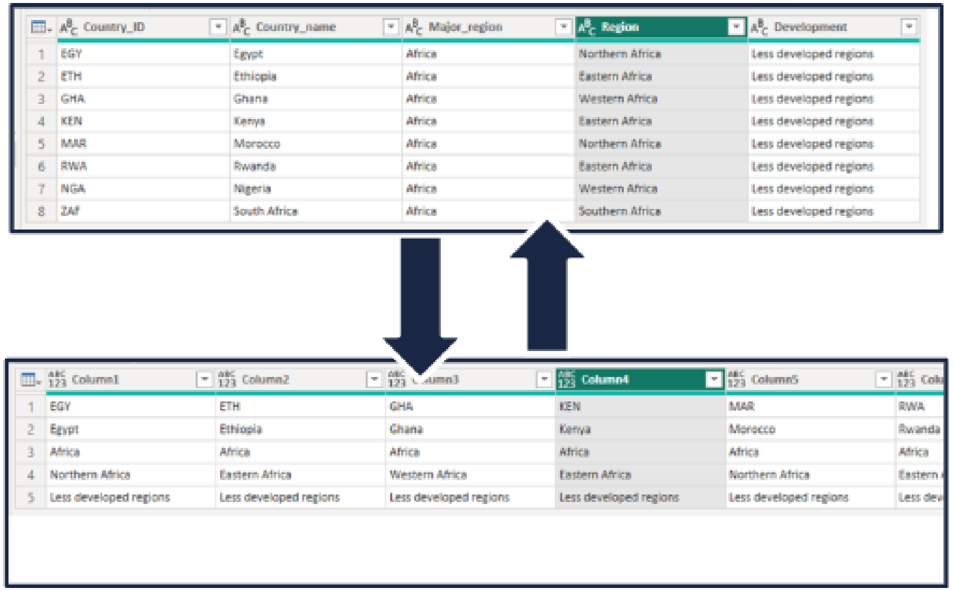
> In the Power Query Editor, there are two distinct **data shape transformation operations**, namely **transposing** and **pivoting** data. Each operation serves a different purpose and can be used to restructure data in various ways.

Transpose

Convert **columns to rows** and **rows to columns** for the entire table.

Allows us to switch from a **wide format** (many columns, few rows) to a **tall format** (few columns, many rows) or vice versa.

Select the table to transpose and click on "**Transpose**" under the "**Table**" section.



	Country_ID	Country_name	Major_region	Region	Development
1	EGY	Egypt	Africa	Northern Africa	Less developed regions
2	ETH	Ethiopia	Africa	Eastern Africa	Less developed regions
3	GHA	Ghana	Africa	Western Africa	Less developed regions
4	KEN	Kenya	Africa	Eastern Africa	Less developed regions
5	MAR	Morocco	Africa	Northern Africa	Less developed regions
6	RWA	Rwanda	Africa	Eastern Africa	Less developed regions
7	NGA	Nigeria	Africa	Western Africa	Less developed regions
8	ZAF	South Africa	Africa	Southern Africa	Less developed regions

	Column1	Column2	Column3	Column4	Column5	Colu
1	EGY	ETH	GHA	KEN	MAR	RWA
2	Egypt	Ethiopia	Ghana	Kenya	Morocco	Rwanda
3	Africa	Africa	Africa	Africa	Africa	Africa
4	Northern Africa	Eastern Africa	Western Africa	Eastern Africa	Northern Africa	Eastern
5	Less developed regions	Less developed regions	Less developed regions	Less developed regions	Less developed regions	Less dev

Transposing, pivoting, and unpivoting data

Pivot

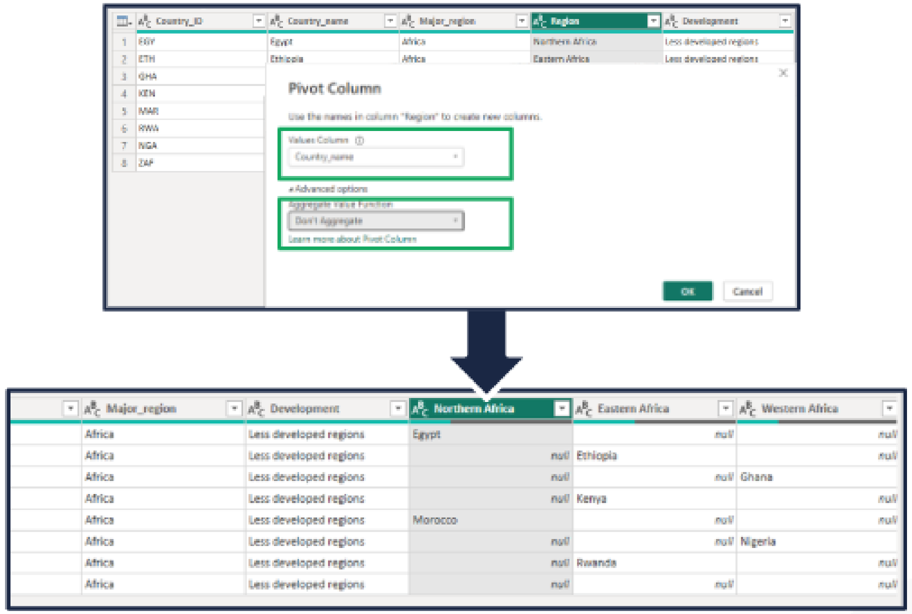
Convert rows into columns for a specific column.

Used to **aggregate data** and create summary tables. It converts unique values in a column into new columns, allowing us to **summarise data** by specific attributes.

Navigate to the "**Transform**" tab and select "**Pivot Column**" under the “**Any Column**” section.

01. Select where the **cell values for the new columns** will come from.

02. We can also specify an **aggregation function**, like sum or average, for the pivoted values.



	Country_ID	Country_name	Major_region	Region	Development
1	EGY	Egypt	Africa	Northern Africa	Less developed regions
2	ETH	Ethiopia	Africa	Eastern Africa	Less developed regions
3	GHA				
4	KEN				
5	MAR				
6	RWA				
7	NGA				
8	ZAF				

Pivot Column

Use the names in column "Region" to create new columns.

Values Column: Country_name

Advanced options

Aggregate Value Function: Don't Aggregate

Learn more about Pivot Column

	Major_region	Development	Northern Africa	Eastern Africa	Western Africa
	Africa	Less developed regions	Egypt	null	null
	Africa	Less developed regions	null	Ethiopia	null
	Africa	Less developed regions	null	null	Ghana
	Africa	Less developed regions	null	Kenya	null
	Africa	Less developed regions	Morocco	null	null
	Africa	Less developed regions	null	null	Nigeria
	Africa	Less developed regions	null	Rwanda	null
	Africa	Less developed regions	null	null	null