PL-300

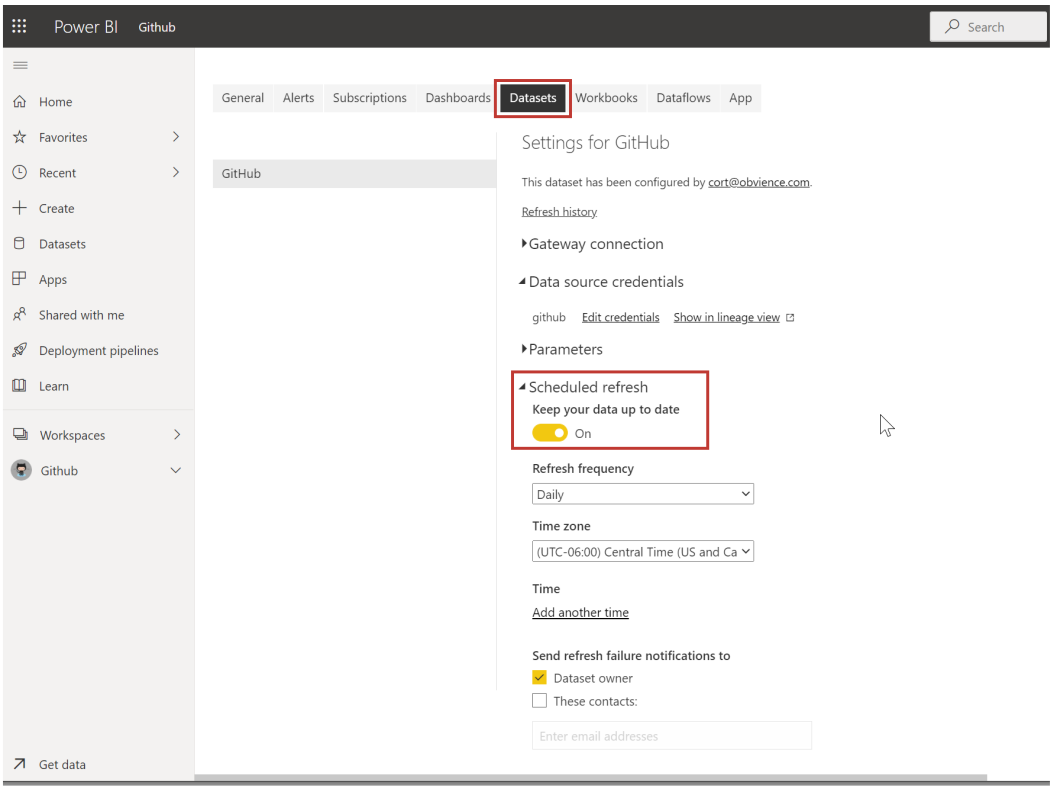
Created	@November 5, 2023 5:47 PM
Class	[MS Learn] Power BI
Note	**Cardinality** describes the uniqueness of the values in a column Cardinality (relationships context) describes the direction of the relationship
Note2	**Aggregating data** → summarize data and present it in at a higher grain (level) Ex: Summarize all sales data and group it at the month level
Note3	**Incremental refresh:** If you have five years' worth of data, and you only need to refresh the last 10 days because that is the only data that has changed, the incremental refresh will refresh only those 10 days of data.

💡 **Making report manifesto**

- What purpose is this report for?
- Who is using the report?
- How can I help people do a better job?
- What is the most important information and how can I highlight it?
- Is this report readable?
- Can people change the elements that they need to if their questions change?
- Do I have visuals that are distracting from the core message of the report?
- Is this report staying focused in a single topic or only a few topics?
- Am I providing all information that the user expects to see in the report?

▼ Get started in PBI Service

- **Power BI Desktop** for authoring reports made up of datasets and visualizations.
- Power BI service for creating dashboards from published reports and distributing content with apps.
- **Power BI Mobile** for on-the-go access to the Power BI service content, designed for mobile.
- **Refresh data**



- **What is a collection of reports and dashboards called in Power BI?**

  An app is a collection of ready-made visuals, pre-arranged in dashboards and reports. You can get apps that connect to many online services from the AppSource.

- **What is the common flow of activity in Power BI?**

Bring data into Power BI Desktop and create a report, share it to the Power BI service, view and interact with reports and dashboards in the service and Power BI mobile.

▼ Get data in PBI

Create dynamic reports with parameters

- Dynamic reports: a single report can be used for multiple purposes
- Use parameters by determining the values that you want to see data for in the report → the report updates accordingly by filtering the data for you
- Need to write SQL query
- Creat dynamic reports for INDIVIDUAL or MULTIPLE values

Get data from relational data sources

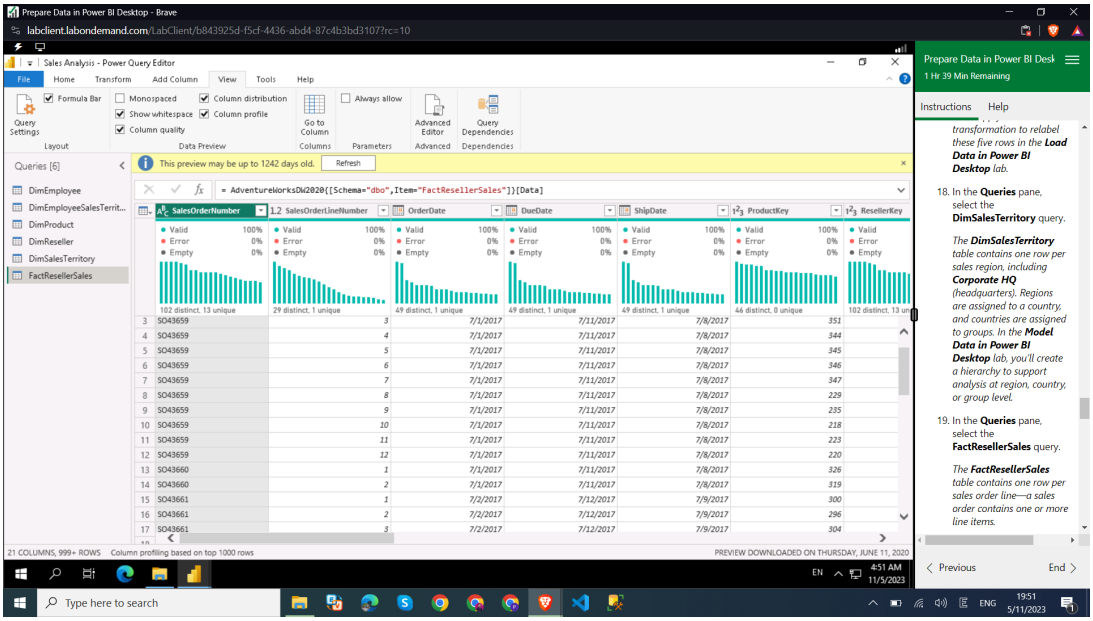
- **Windows** - Use your Windows account (Azure Active Directory credentials).
- **Database** - Use your database credentials.
- **Microsoft account** - Use your Microsoft account credentials. This option is often used for Azure services.

Lab: Prepare data in PowerBI

▼ Get data from SQL Server and Preview Data in Power Query Editor

- ***DimEmployee*** *table in the SQL Server database stores one row for each employee*
  - Review the column distribution for the **EmployeeKey** column
    - *When the distinct and unique counts are the same, it means the column contains unique values.*
    - *When modeling, it's important that some model tables have unique columns. These unique columns can be used to create one-to-many relationships, which you'll do in the* ***Model Data in Power BI Desktop*** *lab.*

- ***DimEmployeeSalesTerritory*** *table stores one row for each employee and the sales territory regions they manage. The table supports relating many regions to a single employee. Some employees manage one, two, or possibly more regions. When you model this data, you'll need to define a many-to-many relationship.*
- **DimProduct** table contains one row per product sold by the company.
- ***DimReseller*** *table contains one row per reseller. Resellers sell, distribute, or value add to the Adventure Works products.*
- ***DimSalesTerritory*** *table contains one row per sales region, including* ***Corporate HQ*** *(headquarters). Regions are assigned to a country, and countries are assigned to groups.*
- ***FactResellerSales*** *table contains one row per sales order line—a sales order contains one or more line items.*



  - Review the column quality for the **TotalProductCost** column → 8% of the rows are empty.

    Missing ***TotalProductCost*** *column values is a data quality issue. To address the issue, in the* ***Load Data in Power BI Desktop*** *lab, you'll apply transformations to fill in missing values by using the product standard cost, which is stored in the related* ***DimProduct*** *table.*

▼ **Get data from CSV Files**

- ***ResellerSalesTargets*** *CSV file contains one row per salesperson, per year. Each row records 12 monthly sales targets (expressed in thousands). The business year for the Adventure Works company commences on July 1.*
- *Hyphen character was used in the source CSV file to represent zero (0)*

Select a storage mode: Import or DirectQuery?

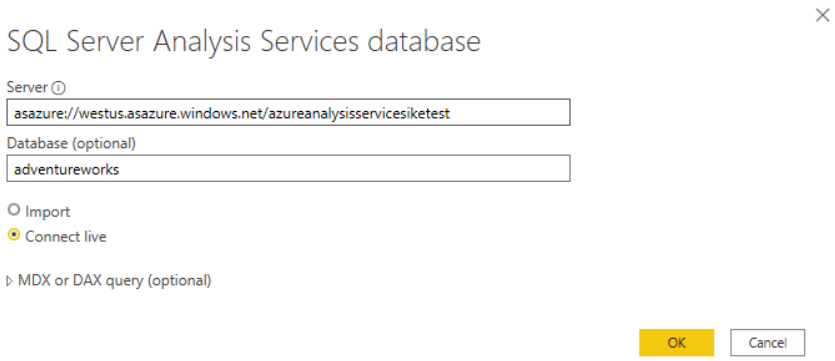
- **Importing** the data means that the data is stored (or cached) in the Power BI file and gets published along with the Power BI reports
  - Data refreshes can be scheduled or on-demand

But let’s say, for security reasons, you aren't allowed to import local copies of the data into your reports or your datasets may simply be too large

⇒ **DirectQuery** allows you to query the data in the data source directly and not import a copy into Power BI ⇒ ensures always viewing the most recent version of the data.

- **Dual** you can identify some data to be directly imported and other data that must be queried

Get data from Azure Analysis Services

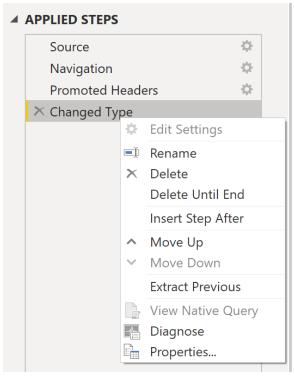


- Similar to getting data from SQL Server:
  - Authenticate to the server.
  - Pick the model you want to use.
  - Select which tables you need.
- Differences
  - Analysis Services models have calculations already created.
  - Instead of using Transact-SQL (T-SQL) to query the data, you can use multi-dimensional expressions (MDX) or data analysis expressions (DAX).
- **Connect live**
  - Helps keep the data and DAX calculations in their original location (without having to import them all into Power BI)
  - When data is refreshed in the service → Power BI reports immediately updated
- You'll likely import the data directly into Power BI. An acceptable alternative is to import all other data that you want (from Excel, SQL Server, and so on) into the Azure Analysis Services model and then use a live connection → This approach keeps the data modeling and DAX measures in one place.

Fix performance issues

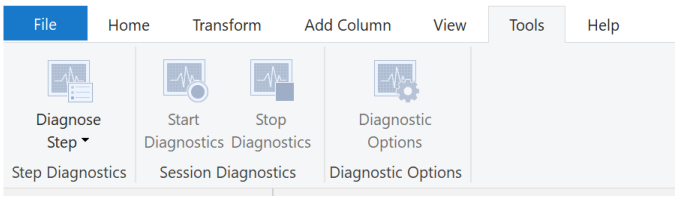
- **Optimize performance in Power Query**

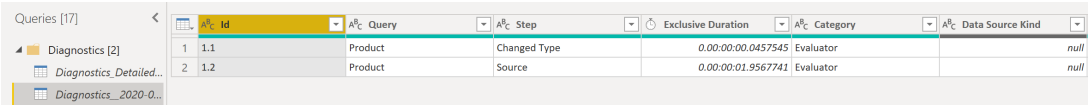
- *Query folding* is the process by which the transformations and edits that you make in Power Query Editor are simultaneously tracked as native queries, or simple **Select** SQL statements, while you're actively making transformations.
- If the **View Native Query** option isn't available → query folding isn't possible for this step



- Tips: If you can translate a transformation into a **Select** statement, which includes operators and clauses such as GROUP BY, SORT BY, WHERE, UNION ALL, and JOIN → you can use query folding.

- **Query diagnostics**
  - Useful when you want to analyze performance on the Power Query side: loading datasets, running data refreshes, or running other transformative tasks, etc.





- **Other techniques to optimize performance**
  - **Process as much data as possible in the original data source.**

    - **Use native SQL queries.** When using DirectQuery for SQL databases, make sure that you aren't pulling data from stored procedures or common table expressions (CTEs).
        - In Advanced Options: when customizing query, avoid using functions or CTE (*If you use a transformation that is too complex, then you will receive an error that either it must be deleted, or the model switched to Import mode*)
    - **Separate date and time** into distinct columns before importing into Power BI.
- **Resolve data import errors**
    - **Query timeout expired** many people concurrently use the same data in the same database
    - **Power BI Query Error: Timeout expired** you've pulled too much data according to your organization's policies
    - **We couldn't find any data formatted as a table**
    - **Couldn't find file**
    - **Data type errors**
        - Sometimes, when you import data into Power BI, the columns appear blank → error in interpreting the data type in Power BI.
        - The resolution to this error is unique to the data source.

          Ex: If you're importing data from SQL Server and see blank columns, you could try to convert to the correct data type in the query.

          Instead of using this query: `SELECT CustomerPostalCode FROM Sales.Customers`

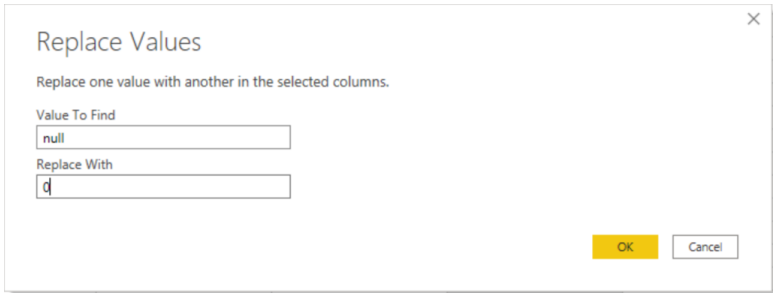
          Use this query: `SELECT CAST(CustomerPostalCode as varchar(10)) FROM Sales.Customers`

▼ Clean, transform, and load data

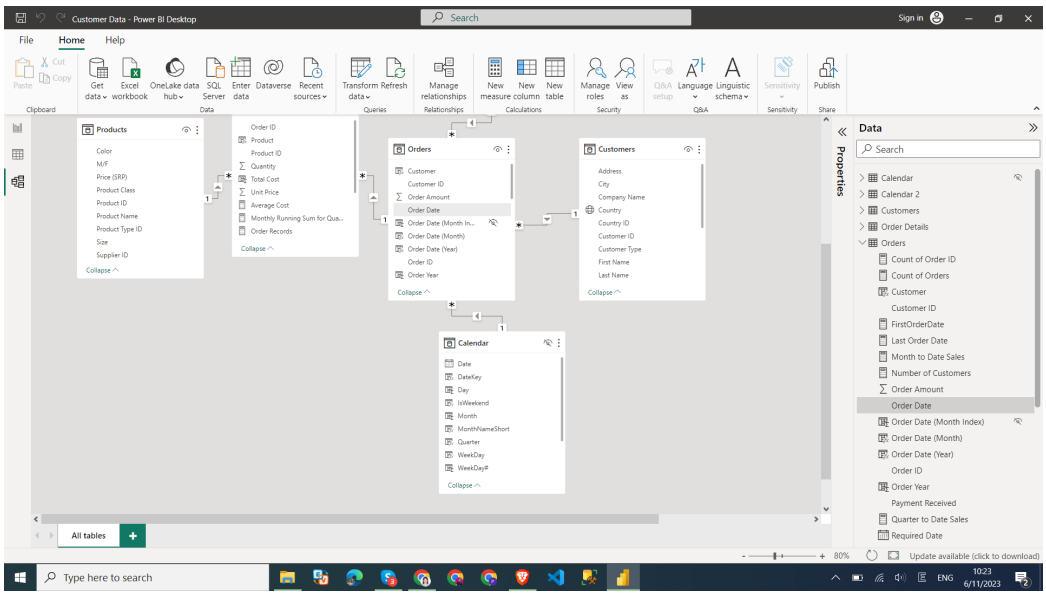
Shaping data

- Promote headers; Rename cols; Remove rows/cols
- Unpivot/Pivot columns

- Replace nulls



- Best practices for naming tables, columns, and values
  - Descriptive business terms
  - Replace underscores ("_") with spaces.
  - Be consistent with abbreviations, prefixes, and words like "number" and "ID"
  - Avoiding acronyms in values (provided that the text will fit on the visual)
- Data types
  - If date as type Text → Date ⇒ auto generated date hierarchy
  - But best practice: use a date table (`CALENDAR` / `CALENDARAUTO`) & create relationship ⇒ get rid of the auto generated hierarchy



Combine multiple tables into a single table

- Append queries: add rows of data to another table or query
  - **Append Queries as New** output of appending will result in a new query or table
  - **Append Queries** add the rows from an existing table into another
- Merge queries
  - Merging queries allows integrating data (even from different data sources such as SQL Server and a CSV file)

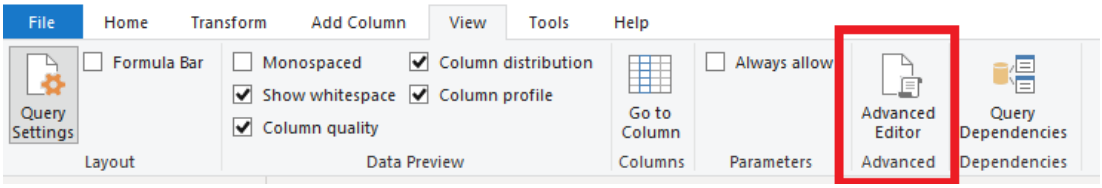
Profile data

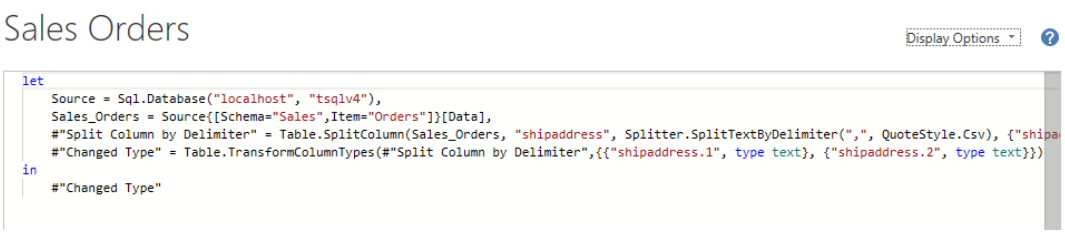
- Examine data structures: using ***Model*** view
- Data anomalies and statistics:
  - By default, Power Query examines the first 1000 rows of your data set. To change this, select the profiling status in the status bar and select **Column profiling based on entire data set**.
  - **Column profile:** Starting point to determine outliers

Column statistics	Value distribution
Ex: Draw attention if you have a maximum value that is beyond what your business identifies as a "maximum"	Ex: If a value appears far more than other values in a column

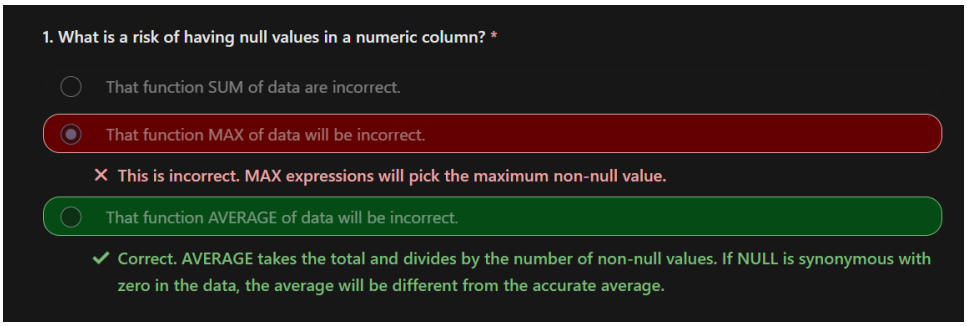
Advanced Editor (M code)

- Each time you shape data in Power Query, you create a step → The combined steps are available in Power Query Advanced Editor





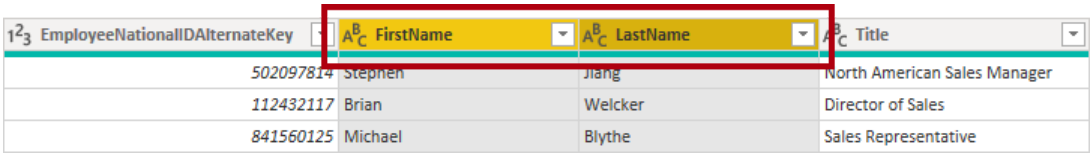
Check knowledge



Lab: Transform and load data

- ***Go to Column*** *is a useful feature with many columns. Otherwise, you can horizontally scroll find columns.*

- **Merge Columns**
  - To create a single name column, first select the **FirstName** column header. While pressing the **Ctrl** key, select the **LastName** column.



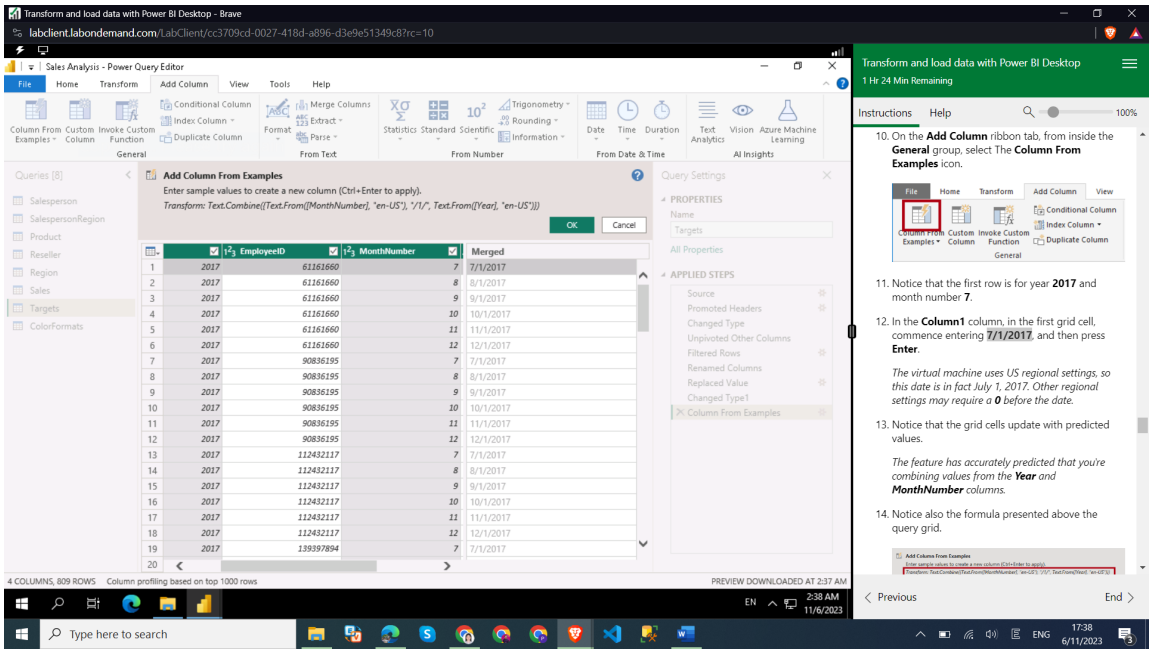
EmployeeNationalIDAlternateKey	FirstName	LastName	Title
502097814	Stephen	Jiang	North American Sales Manager
112432117	Brian	Welcker	Director of Sales
841560125	Michael	Blythe	Sales Representative

  - Right-click either of the select column headers, and then in the context menu, select **Merge Columns** *(available in the ribbon as well)*

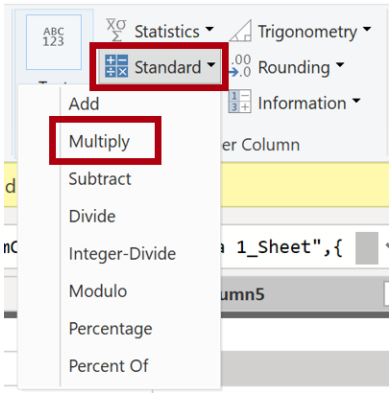
- **Custom Column**
  - *You may recall in the **Prepare Data in Power BI Desktop** lab that a small percentage of **FactResellerSales** rows had missing **TotalProductCost** values.*
  - *The **DimProduct** column has been included to retrieve the product standard cost column to assist fixing the missing values.*
    - Expand the **DimProduct** column → include only the **StandardCost** column
    - `if [TotalProductCost] = null then [OrderQuantity] * [StandardCost] else [TotalProductCost]`

- **Fixed Decimal Number ($ symbol)**
  - *The fixed decimal number data type allows for 19 digits, and allows for more precision to avoid rounding errors.*
  - *It's important to use the fixed decimal number type for **financial values**, or rates (like exchange rates).*
    - Unit Price
    - Sales
    - Cost

- **Custom From Examples**

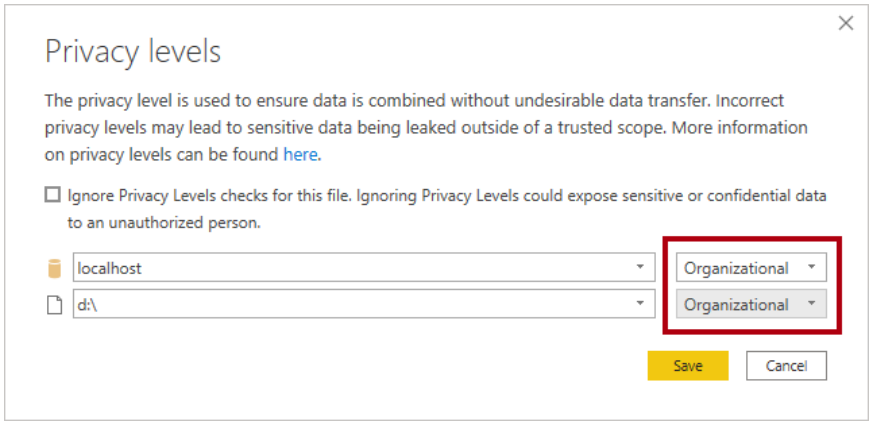


- Transform > Number Column > Standard > Multiply
  - To multiply the **Target** values by 1000

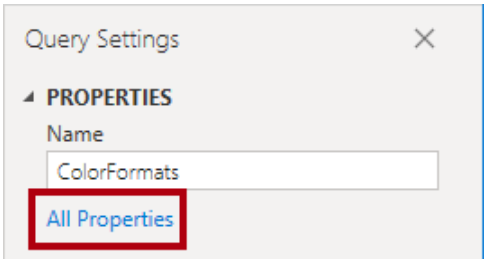


- **Merging queries: Privacy Levels**
  - Privacy levels can be configured for data source to determine whether data can be shared between sources.
  - Setting each data source as **Organizational** allows them to share data, if necessary.

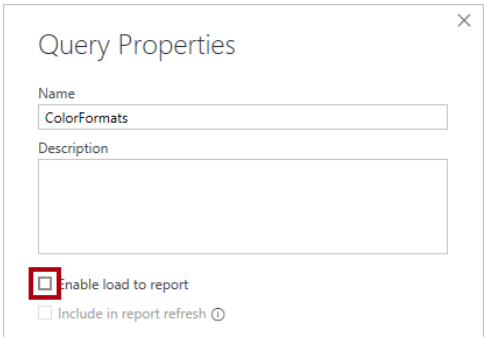
- Private data sources can never be shared with other data sources. It doesn't mean that Private data can't be shared; it means that the Power Query engine can't share data between the sources.



- **Not load a table to data model**
  - In the **Query Settings** pane, select the **All Properties** link.



  - In the **Query Properties** window, UNCHECK the **Enable Load To Report** checkbox.



▼ Design a data model

▼ Develop data model

- **Developing the model involves the following tasks:**
  - Connect to data
  - Transform and prepare data
  - Define business logic by adding DAX calculations
  - Enforce data permissions with row-level security by adding roles
  - Publish the model to Power BI

  Optimal models are important for delivering good query performance and for minimizing data refresh times and the use of service resources, including memory and CPU.

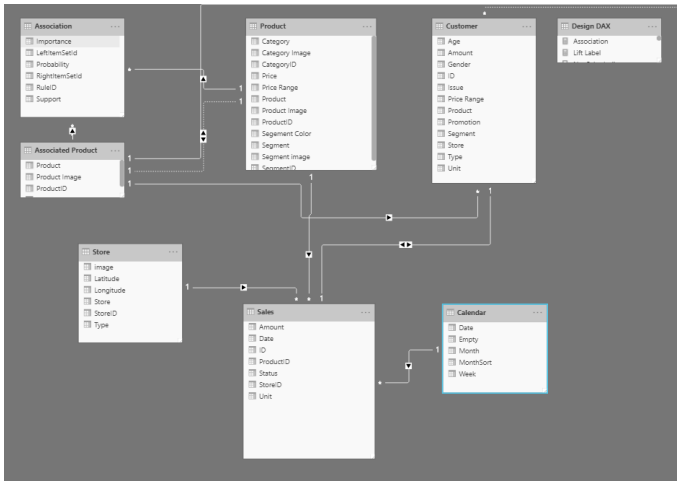
- **Star schema:**
  - **Fact tables** store an accumulation of rows that represent ***observations*** or ***events*** that record a specific business activity

    Ex: sales orders and the order lines; record stock movements, stock balances, or daily currency exchange rates.
  - **Dimension tables** describe your business entities, which commonly represent people, places, products, or concepts.

- **Filtering:** or slicing
  - Filters can be applied to three different scopes: the entire report, a specific page, or a specific visual.
  - Filtering is also applied in the background when row-level security (RLS) is enforced.
    - Ex: When doing Learnit's PBI course: apply RLS so that a specific role can only see the data of the West region.
  - Each report visual can inherit filters or have filters directly applied to it.

▼ Choose a PBI model framework: import model, DirectQuery model, or composite model?

- **PBI model fundamentals:**
  - **Data model** (or model):
    - comprises of tables
    - It can also include relationships, hierarchies, and calculations.



  - **PBI dataset:** a model published to a workspace in PBI Service ⇒ dataset
    - Some datasets represent connections to external-hosted models in AAS or SSAS
    - Others can represent real-time data structures (push datasets, streaming datasets, or hybrid datasets)
  - **Analytic query:**
    - When PBI visualizes dataset data, it prepares and sends an analytic query.
    - An analytic query has three phases executed in order: Filter → Group → Summarize
  - **Star schema design**
  - **Table storage mode**

        - **Import** – Queries retrieve data that's stored, or cached, in the model.
        - **DirectQuery** – Queries pass through to the data source.
        - **Dual** – Queries retrieve stored data or pass through to the data source. Power BI determines the most efficient plan, striving to use cached data whenever possible.
    - **Model framework:**
        - An import model: all tables' storage mode set to **Import**
        - A DirectQuery model: all tables' storage mode set to **DirectQuery**
        - A composite model: more than one source group.

- **Import model**

  Benefits:

    - Support all Power BI data source types: databases, files, feeds, web pages, dataflows, and more.
    - Can integrate source data. For example, one table sources its data from a relational database while a related table sources its data from a web page.
    - Support all DAX and Power Query (M) functionality.
    - Support calculated tables.
    - Deliver the best query performance. That's because the data cached in the model is optimized for analytic queries (filter, group, and summarize) and the model is stored entirely in memory.

  Limitations:

    - **Model size:** 1-GB per dataset limit (refers to the compressed size of the Power BI model, not the volume of data being collected from the source system)

- **Data refresh:** Imposes limits on how often scheduled refresh operations can occur (up to 8 times per day in a shared capacity, and up to 48 times per day in a dedicated capacity)
  - By default, to refresh a table, Power BI removes all data and reloads it again.
  - For large fact tables, to reduce burden → incremental refresh ⇒ automates the creation and management of time-period partitions, and intelligently update only those partitions that require refresh.

- **DirectQuery model**

  Benefits:

  - Source data is large (e.g., an entire data WH) and changes rapidly + users need to see current data
  - **Enforce source RLS:**
    - Useful when the source database already enforces row-level security (RLS). Instead of replicating those RLS rules in PBI model, the source data base can enforce its rules.
    - Works only for some relational databases (see Azure SQL Database with DirectQuery)
  - **Data sovereignty restrictions:**
    - If your organization has security policies that restrict data leaving their premises, then it isn't possible to import data. A DirectQuery model that connects to an on-premises data source may be appropriate. (Power BI Report Server for on-premises reporting)
  - **Create specialized datasets:**
    - You can connect to a **PBI dataset** (or Azure Analysis Services model) and convert it to a DirectQuery local model.
    - The original dataset is a remote model, and the new dataset is the local model.
    - Can personalize and/or extend a remote model:

- Rename objects (tables or columns), add measures, calculated columns or calculated tables
- Can add new Import or DirectQuery tables → might become a composite model

Limitations:

- **Not all data sources are supported:** Only major relational database systems are supported.
- All Power Query (M) transformations are not possible
  - Ex: If you are connecting to a MySQL database, you would write a native query in MySQL flavor.

    → Native queries are used to translate Power Query (M) transformations into a language that the underlying data source can understand.

    → This means that some transformations, such as pivot and unpivot, cannot be translated into a native query and are therefore not supported in DirectQuery.
- Analytic query performance can be slow, especially if source systems aren't optimized (with indexes or materialized views), or database has insufficient resources for the analytic workload.
- Analytic queries can impact on source system performance → could result in slower experience for all workloads, including OLTP operations.

Boost DirectQuery model performance:

- **Data source optimizations:** create indexes and materialized views; ensure the database has sufficient resources for all workloads
- **Aggregation tables using DirectQuery storage mode:**
  - User-defined aggregation

        - When you set the aggregation table to DirectQuery storage mode, it can query a materialized view in the data source.

- **Composite model:**

  Benefits:

  - Boost performance (when some analytic queries can be performed from imported data)
  - Can extend your model with new calculated columns and tables

  Limitations:

  - Import (or dual, as described later) storage mode tables requires periodic refresh as imported data can become out of sync with DirectQuery sourced data
  - **When an analytic query must combine imported and DirectQuery data, Power BI must consolidate source group query results, which can impact performance.
  - When chaining models (DirectQuery to Power BI datasets), modifications made to upstream models can break downstream models (dataset impact analysis)
  - Relationships between tables from different source groups are known as limited relationships. A model relationship is limited when the Power BI can't determine a "one" side of a relationship. Limited relationships may result in different evaluations of model queries and calculations. (see Relationship evaluation)

  Boost DirectQuery model performance with import data:

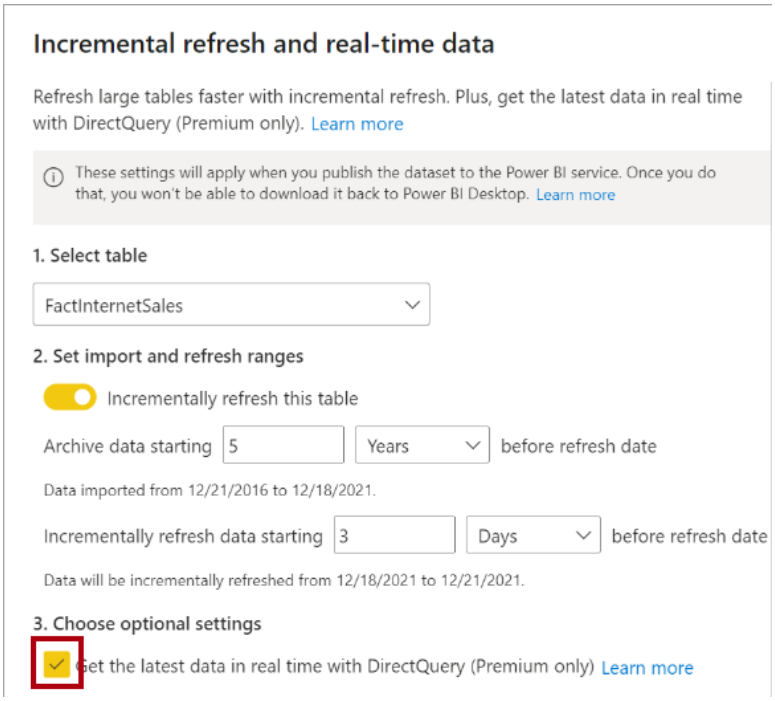
  - **To help avoid this situation for higher-grain queries → Can add aggregation tables using Import storage mode to your model (or enable automatic aggregations) and set related dimension tables to dual storage mode.

⇒ This way, Power BI directs higher-grain fact queries to a cached aggregation.

⇒ Slicer visuals and filter card lists (often based on dimension table columns) render more quickly because they're queried from cached data.

Deliver real-time data from an import model: Premium license

- When you set up an import table with incremental refresh, you can enable the **Get the latest data in real-time with DirectQuery** option.



- By enabling this option, Power BI automatically creates a table partition that uses DirectQuery storage mode.
- In this case, the table becomes a **hybrid table**, meaning it has import partitions to store older data, and a single DirectQuery partition for current data.
- When Power BI queries a hybrid table, the query uses the cache for older data, and passes through to the data source to retrieve current data.

Choose the **import** model framework whenever possible. This framework offers you the most options, design flexibility, and delivers fast performance. Be sure to apply data reduction techniques to ensure that Power BI loads the least amount of data possible.

Choose the **DirectQuery** model framework when your data source stores large volumes of data and/or your report needs to deliver near real-time data.

Choose the **composite** model framework to:

- Boost the query performance of a DirectQuery model.
- Deliver near real-time query results from an import model.
- Extend a Power BI dataset (or Azure Analysis Services model)

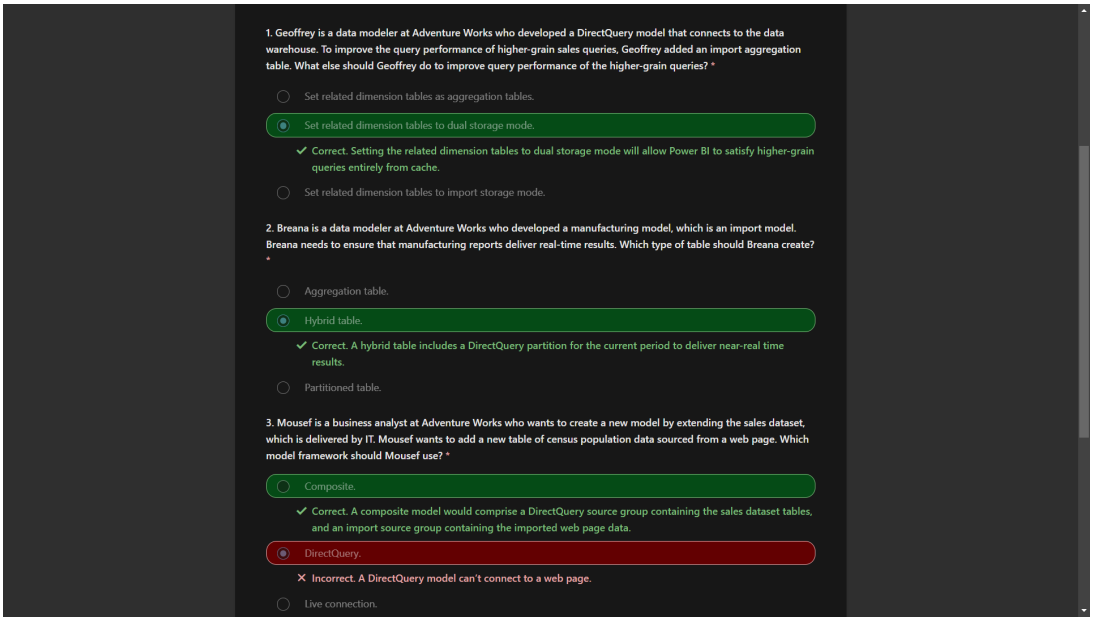
You can boost the query performance of a DirectQuery model by using aggregation tables that use either import or DirectQuery storage mode:

- When using import aggregation tables, be sure to set related dimension tables to use dual storage mode. That way, Power BI can satisfy higher-grain queries entirely from cache.
- When you set the aggregation table to DirectQuery storage mode, it can query a materialized view in the data source.

You can deliver near real-time query results from in import model by creating a hybrid table. In this case, Power BI adds a DirectQuery partition for the current period.

Lastly, you can create specialized models by chaining to a core model by using DirectQuery. This type of development is typically done by a business analyst who extends core models, which IT delivers and supports.

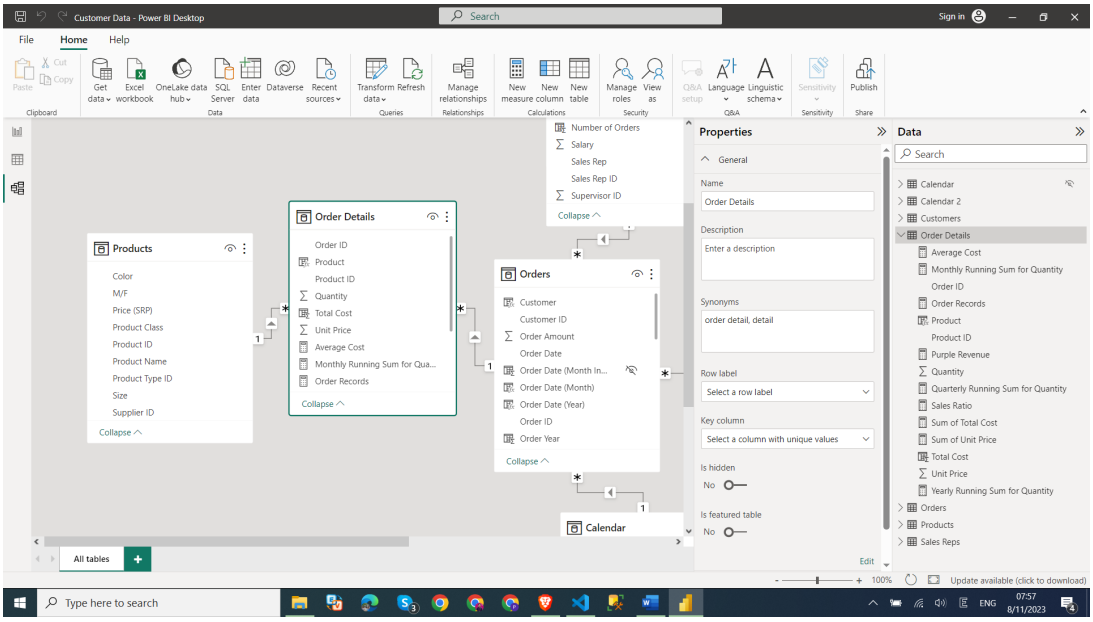
Check knowledge



▼ Design a data model in Power BI

Work with tables

- Build relationships between tables
- Configure table and column properties: **Properties** pane



General tab	**Formatting** tab (when click on column)	Advanced tab
• Edit the **name** and **description** of the column • Add **synonyms** that can be used to identify the column when you are using the Q&A feature • Add columns into a **folder** to further organize the table structure • **Hide** or show the column	• Change the **data type** • **Format date**	• **Sort** by a specific column • Assign a specific **category** to the data (ex: 🌐 icon displayed if choose category “Country/Region for a column) • Summarize the data • Determine if the column or table contains **nulls**

Create a date table

- **Create a common date table**
  - **Source data:** Occasionally, source databases and data warehouses already have their own date tables.
    - Identify company holidays
    - Separate calendar and fiscal year
    - Identify weekends versus weekdays

- **DAX:** CALENDARAUTO() or CALENDAR()

```
Dates  = CALENDAR(DATE(2011, 5, 31), DATE(2022, 12, 31))
```

You also want to see columns for just the year, the month number, the week of the year, and the day of the week: select **New Columns**

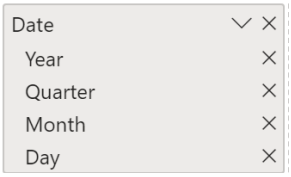
```
Year = YEAR(Dates[Date])
MonthNum = MONTH(Dates[Date])
WeekNum = WEEKNUM(Dates[Date])
DayoftheWeek = FORMAT(Dates[Date], "DDDD")
```

- **Power Query:** link

- **Mark as the official date table**
- **Build your visual**
  - Need to establish a relationship btw common date table and the Sales and Orders tables.

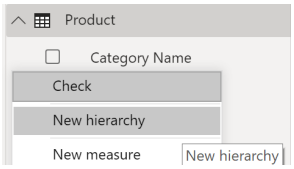
Work with dimensions

- **Hierarchies**
  - Values of the date type auto entered as a hierarchy (if the table has not been marked as a date table)

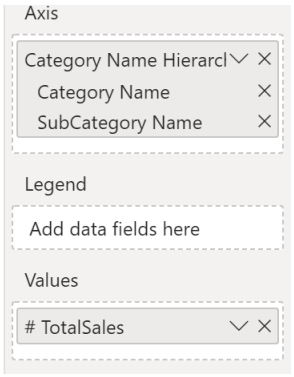


  - Consider that you want to create a stacked bar chart of **Total Sales by Category and Subcategory** → create a hierarchy in the **Product** table for categories and subcategories.

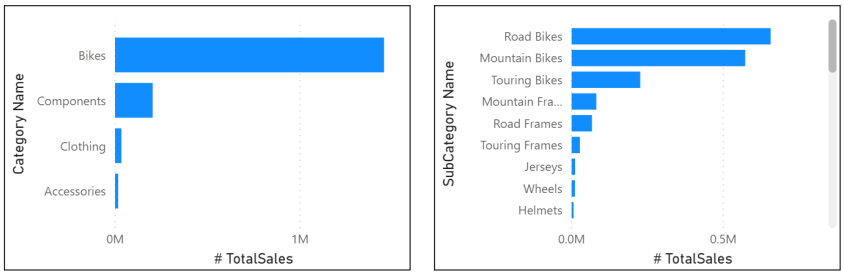
- Right-click **Category Name** > **New hierarchy**



- Drag and drop the **Subcategory** column into this new hierarchy
- In **Properties** pane > scroll down > select **Apply Level Changes** for hierarchy
- Visualize



- Drill down on the visual to view both **Category** and **Subcategory**



- **Parent-child hierarchy:** multiple employees can have the same manager, manager is “child” of another manager

	Employee ID	Employee	Manager ID	Manager
1	1010	Roy F	null	
2	1011	Pam H	1010	Roy F
3	1012	Guy L	1010	Roy F
4	1013	Roger M	1011	Pam H
5	1014	Kaylie S	1011	Pam H
6	1015	Mike O	1012	Guy L
7	1016	Rudy Q	1012	Guy L

→ Your responsibility to "flatten" it so that you can see more data granularity.

Flatten parent-child hierarchy

- DAX to retrieve managerial path for each employee

```
Path = PATH(Employee[Employee ID], Employee[Manager ID])
```

Employee ID	Manager ID	Employee	Manager	Path
1010		Roy F		1010
1011	1010	Pam H	Roy F	1010|1011
1012	1010	Guy L	Roy F	1010|1012
1013	1011	Roger M	Pam H	1010|1011|1013
1014	1011	Kaylie S	Pam H	1010|1011|1014
1015	1012	Mike O	Guy L	1010|1012|1015
1016	1012	Rudy Q	Guy L	1010|1012|1016

Roger M (ID 1013) ← Pam H (ID 1011) is his manager ← Roy F (ID 1010) is her manager

- To flatten the hierarchy → PATHITEM function

```
Level 1 = PATHITEM(Employee[Path],1)
```

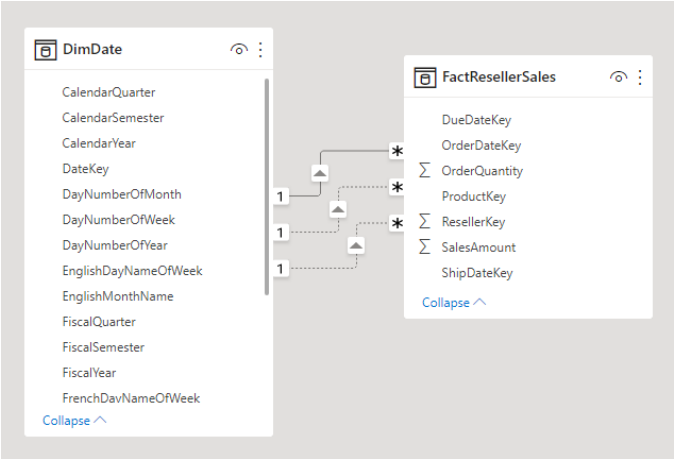
```
Level 2 = PATHITEM(Employee[Path],2)
```

```
Level 3 = PATHITEM(Employee[Path],3)
```

Employee ID	Name	Manager	Manager ID	Path	Level 1	Level 2	Level 3	Level 4
1000	Quincy Howard			1000	1000			
1001	Mallory Yang	Quincy Howard	1000	1000|1001	1000	1001		
1002	Donovan Maynard	Quincy Howard	1000	1000|1002	1000	1002		
1003	Giselle Mcclain	Mallory Yang	1001	1000|1001|1003	1000	1001	1003	
1004	Melvin Marsh	Mallory Yang	1001	1000|1001|1004	1000	1001	1004	
1005	Ria Snow	Giselle Mcclain	1003	1000|1001|1003|1005	1000	1001	1003	1005
1006	Callie Savage	Giselle Mcclain	1003	1000|1001|1003|1006	1000	1001	1003	1006

- Create a hierarchy as did previously

- **Role-playing dimensions:**
  - A **role-playing dimension** is a dimension that can filter related facts differently.
  - For example, at Adventure Works, the date dimension table has three relationships to the **FactResellerSales** table.



→ You have a table named **FactResellerSales** that includes two date columns, **OrderDateKey** and **ShipDateKey.**

→ Both columns are related to the **Date** column in the **DimDate** table.

→ In this case, the **Date** table is described as a role-playing dimension because it could play the role of *order date* or *ship date*.

⇒ While this design is possible, it's important to understand that there can only be one active relationship between two tables. All remaining relationships must be set to inactive.

⇒ In this instance, the active relationship is set to the most common filter used by reports at Adventure Works: *order date* relationship.

- To use inactive relationships: `USERELATIONSHIP`
- Mitchell's course: avoid rebuild measures for role-playing dimension (from 1:53:58) **Tabular Editor add-ins → Calculation groups**

Define data granularity

- **Data granularity:** The ***level of detail*** in your data (more granularity → greater level of detail)

  Scenario: You company manages 1,000 refrigerated semi-trucks. Every few minutes, each truck uses a MS Azure IoT application to record its current temperature → TOO MANY records each day to find interesting ones.

  - You might want to import the data by using a daily average for each truck → reduce the records in the database to one record for each truck for each day.

    → If acceptable enough for tracking costs and errors, then use that data granularity.

  - Alternatively, you could select the last recorded temperature, or you could only import records that are above or below a normal range of temperatures.

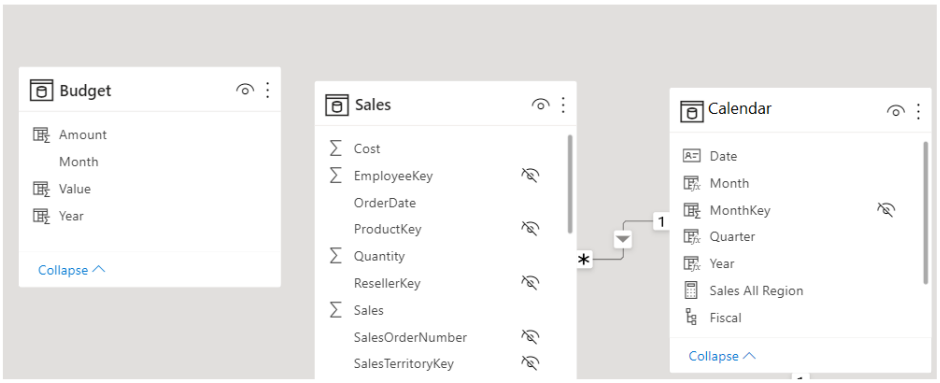
  ⇒ Any of these methods will reduce the total records that you import, while still bringing in data that is comprehensive and valuable.

  For different scenarios, you could settle on data granularity that is defined weekly, monthly, or quarterly.

- **Change data granularity to build a relationship between two tables**

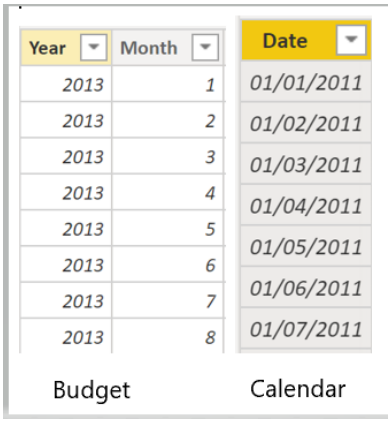
Asked to build a matrix of total sales and budget over time by using the **Calendar**, **Sales**, and **Budget** tables.

- Lowest level of time-based detail that the **Sales** table goes into is by day, for instance 5/1/2020, 6/7/2020, and 6/18/2020.
- The **Budget** table only goes to the monthly level, for instance, the budget data is 5/2020 and 6/2020.



As shown in the preceding figure, a relationship between Budget and Calendar is missing.

Current columns of 2 tables:



Year	Month
2013	1
2013	2
2013	3
2013	4
2013	5
2013	6
2013	7
2013	8

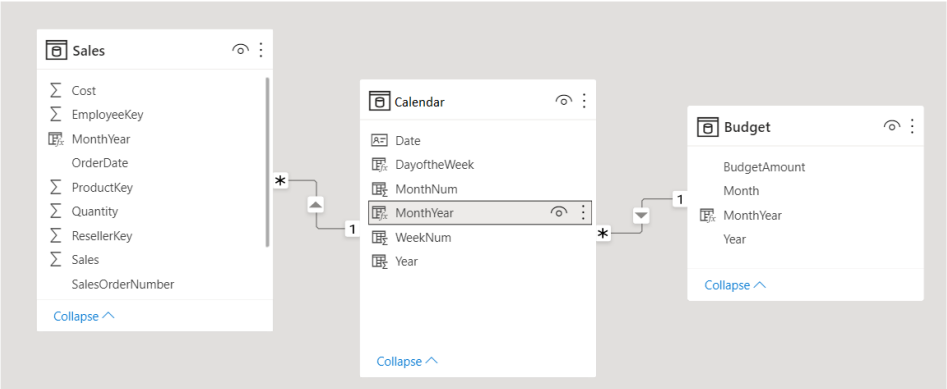
Budget

Date
01/01/2011
01/02/2011
01/03/2011
01/04/2011
01/05/2011
01/06/2011
01/07/2011

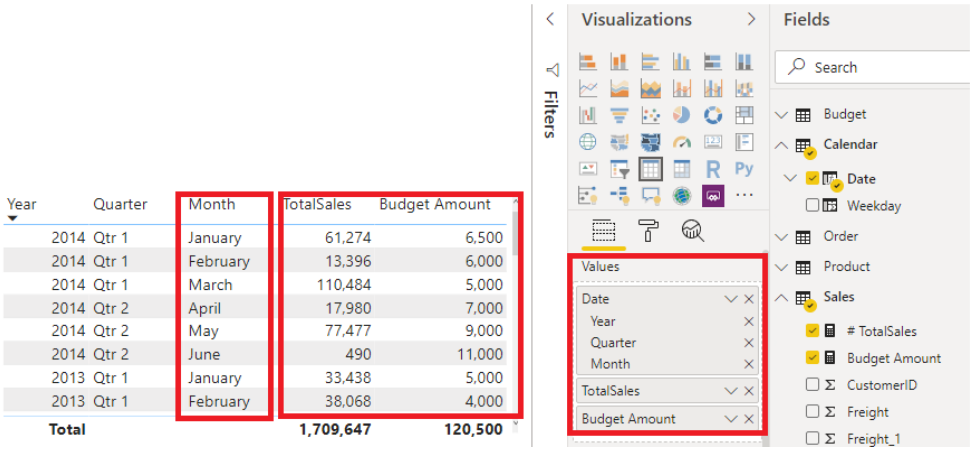
Calendar

- Concatenate the **Year** and **Month** columns of the **Budget** table
- Change **MonthYear** column's data type to Date type (auto create hierarchy)

- Change format to dd/mm/yyyy (auto generates 1st of each month)
- Create the relationship: **MonthYear** column of **Budget** table - **Date** column of **Calendar** table



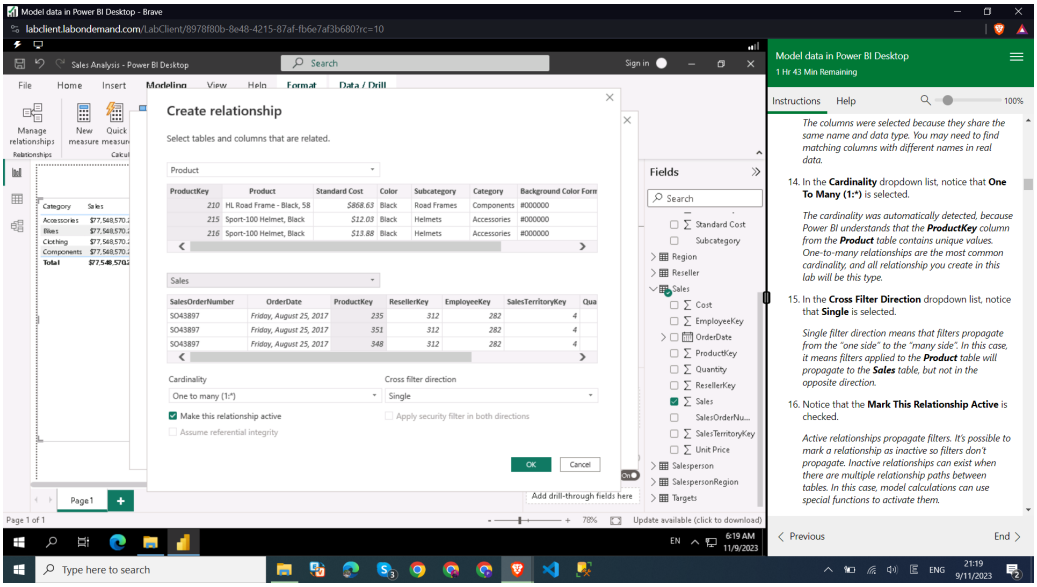
- Visualize



▼ Lab: Model data in Power BI Desktop

Cross-filter direction:

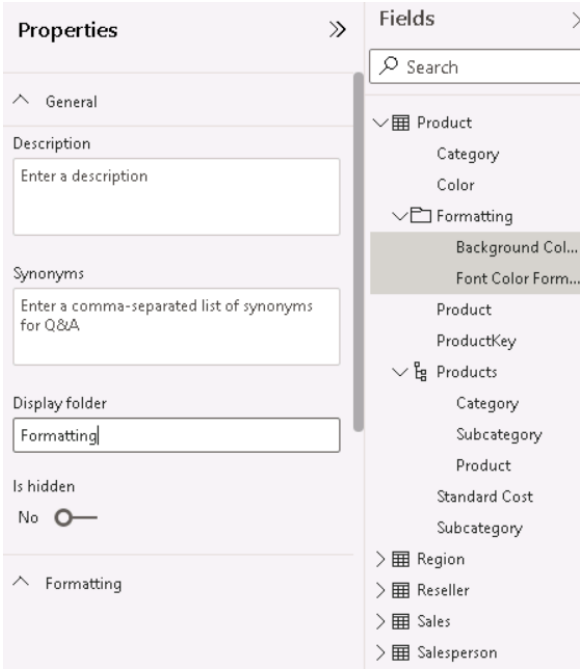
- **Cross Filter Direction** dropdown list > **Single**
- *Single filter direction means that filters propagate from the "one side" to the "many side". In this case, it means filters applied to the **Product** table will propagate to the **Sales** table, but not in the opposite direction.*



Configure tables:

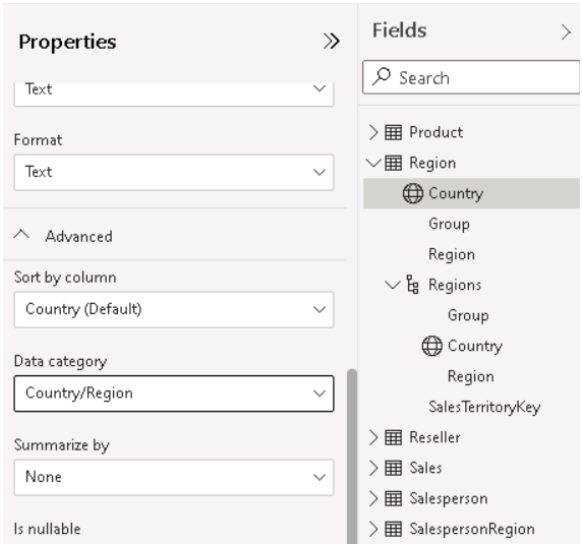
- **Properties** pane > **Display Folder** box

  *2 columns are now inside a folder **Formatting***



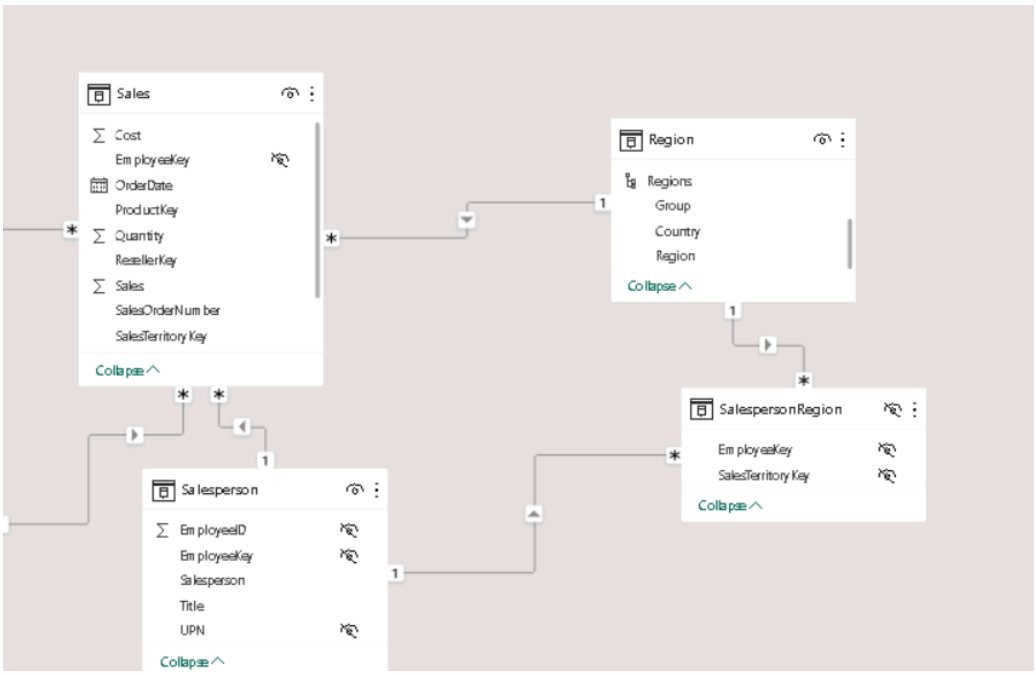
- **Category**

Data categorization can provide hints to the report designer. In this case, categorizing the column as country or region (🌐) provides more accurate information to Power BI when it renders a map visualization.

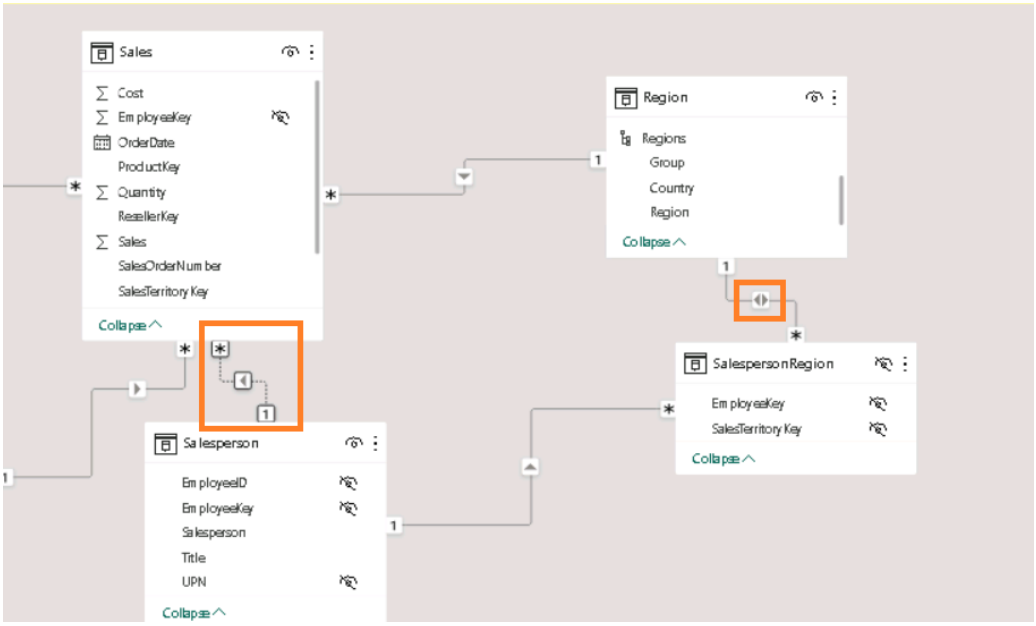


- **Create a many-to-many relationship:** Read this section carefully

  *Consider that the **Salesperson** table filters the **Sales** table. It also filters the **SalespersonRegion** table, but it doesn't continue by propagating filters to the **Region** table (the arrowhead is pointing the wrong direction btw SalespersonRegion and Region).*



After configuring, relationship has double-arrow head and the other relationship is inactive.



Understanding the lab then come back to this module can help understanding section *Cross-filter direction & Cardinality and cross-*

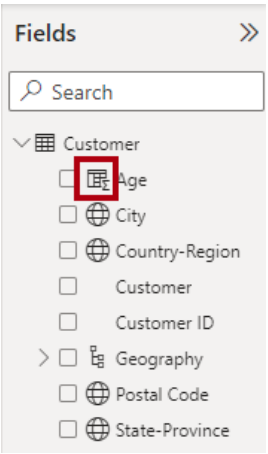
***filter direction* better.**

▼ DAX

▼ Write DAX formula

DAX formula

- Calculated tables
  - Date tables
  - Role-playing dimensions
  - What-if analysis
    - A calculated table is automatically added to your model.
    - What-if parameters allow report users to select or filter by values stored in the calculated table.
    - What-if calculated tables aren't related to other model tables.
- Calculated columns: Add New Column > include calculations



- Measures
  - Measures don't store values in the model.
  - Measures are only evaluated during query time to return summarizations of model data.
  - Measures can reference other measures → compound measure.

- *Implicit measures* only use a specific aggregation function (e.g., either SUM or AVG etc.)
- (*Explicit) measures* are more complex calculations (e.g., calculate the ratio of each month's sales amount over the yearly sales amount)

> 💡 While **calculated columns** are evaluated during data **refresh** time, **measures** are only evaluated during **query** time.

> 💡 Add **custom columns (using M)** to Power Query queries.
>
> Add **calculated columns (using DAX)** to model tables.

DAX data types

Model data type	DAX data type	Related function
N/A	BLANK	`BLANK` DAX function returns BLANK `ISBLANK` DAX function tests whether an expression evaluates to BLANK

DAX operators

- **Comparison operators**

Operator	Description
==	Strict equal to

  All comparison operators, **EXCEPT strict equal to** (**==**), treat BLANK as equal to the number zero, an empty string (""), the date December 30, 1899, or FALSE.

  - `[Revenue] = 0` will be TRUE when the value of `[Revenue]` is either zero or BLANK.
  - `[Revenue] == 0` is TRUE only when the value of `[Revenue]` is zero.

- **Text concatenation operator**

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

- **Logical operators**

Operator	Description
&&	AND
|| (double pipe)	OR
IN	
NOT	

```
ANZ Revenue =
CALCULATE(
    [Revenue],
    Customer[Country-Region] IN {
        "Australia",
        "New Zealand"
    }
)
```

▼ Add measures (can skip)

Add measures to Power BI Desktop models - Training

In this module, you'll learn how to work with implicit and explicit measures. You'll start by creating simple measures, which summarize a single column or table. Then, you'll

https://learn.microsoft.com/en-us/training/modules/dax-power-bi-add-measures/

▼ Add calculated tables and columns

- Ex: When the formula is evaluated for each row, the `'Due Date'[Due Date]` column reference returns the column value for *that row* (like how Excel works)

- However, row context doesn't extend beyond the table. If your formula needs to reference columns in other tables:
  - If the tables are related, directly or indirectly:
    - `RELATED` function retrieves the value at the one-side of the relationship,
    - `RELATEDTABLE` retrieves values on the many-side (returns a table object)
  - When the tables aren't related, you can use the `LOOKUPVALUE` DAX
  - Ex: Add to the **Sales** table:

```
Discount Amount =
(
    Sales[Order Quantity]
        * RELATED('Product'[List Price])
) - Sales[Sales Amount]
```

▼ DAX time intelligence + Lab

Time intelligence calculations modify the filter context for ***date filters*** (involving a date table).

Useful for cumulative calculations like yearly/quarterly/monthly running total/average

- `TOTALYTD` Evaluates an expression for YTD in the current filter context (can only pass 1 filter expression)
  - `TOTALQTD` , `TOTALMTD`

> 💡 If need to apply multiple filter expressions, use `CALCULATE` + `DATESYTD` functions.

- `DATESYTD` Returns a 1-column table that contains dates for the year-to-date
  - `DATESQTD` , `DATESMTD`

- `DATESBETWEEN` Returns a 1-column table that contains dates (given start date, ***given end date***)
- `DATESINPERIOD` Returns a 1-column table that contains dates (given start date, ***number of intervals***)

Comparisons over time

- `DATEADD` Returns a 1-column table that contains dates
- `PARALLELPERIOD`

> 💡 Dates starts at June 10 and finishes at June 21 of the same year → want to shift that selection forward by one month:
> + `DATEADD` returns only dates from July 10 to July 21
> + `PARALLELPERIOD` returns all dates from the next month (July 1 to July 31)

- `SAMEPERIODLASTYEAR` Returns a 1-column table that contains dates shifted 1 year back in time from the dates specified.
- Others: `NEXTDAY` and `PREVIOUS` `DAY` (MONTH, QUARTER, YEAR)

📌 **See example in each section and in lab**

📌 **Additional time intelligence calculations:** Read examples

- **Calculate new occurrences** (life-to-date (LTD) calculations)

  Ex: Calculate number of new customers up to Aug 2017

  Calculate number of new customers within Aug 2017

- **Snapshot calculations**

  Ex: Last recorded units balance for each month

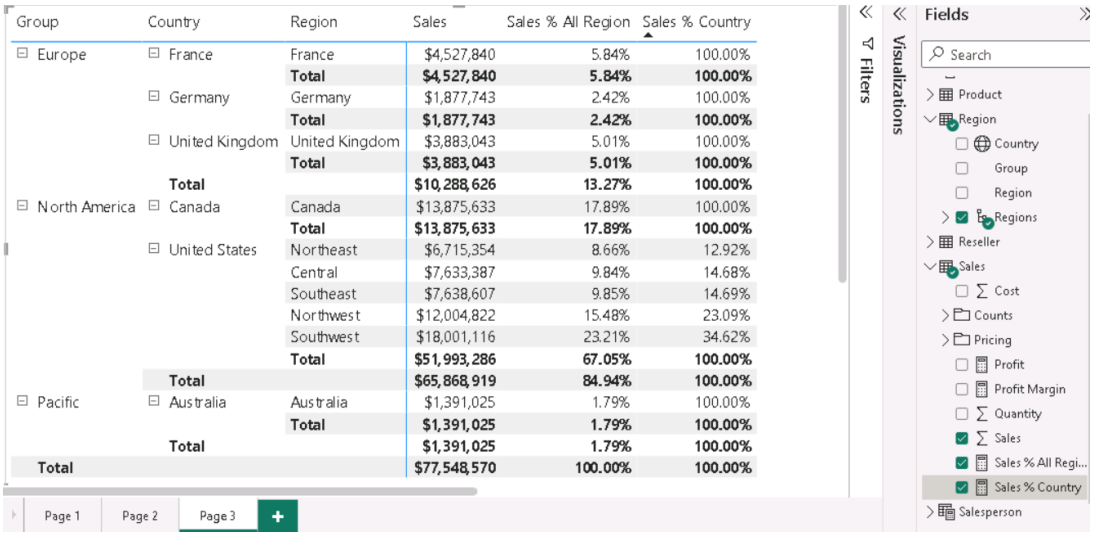
  - `LASTDATE`

- `LASTNONBLANK`

Term	Definition
column	A column expression
expression	An expression evaluated for blanks (for each value of **column**)

▼ **Lab:** `REMOVEFILTERS`

Work with Filter Context: Matrix visual



- Sales % All Region

```
Sales % All Region =
DIVIDE(
 SUM(Sales[Sales]),
 CALCULATE(
 SUM(Sales[Sales]),
 REMOVEFILTERS(Region)
 )
)
```

*Removes any filters applied to the **Region** table*

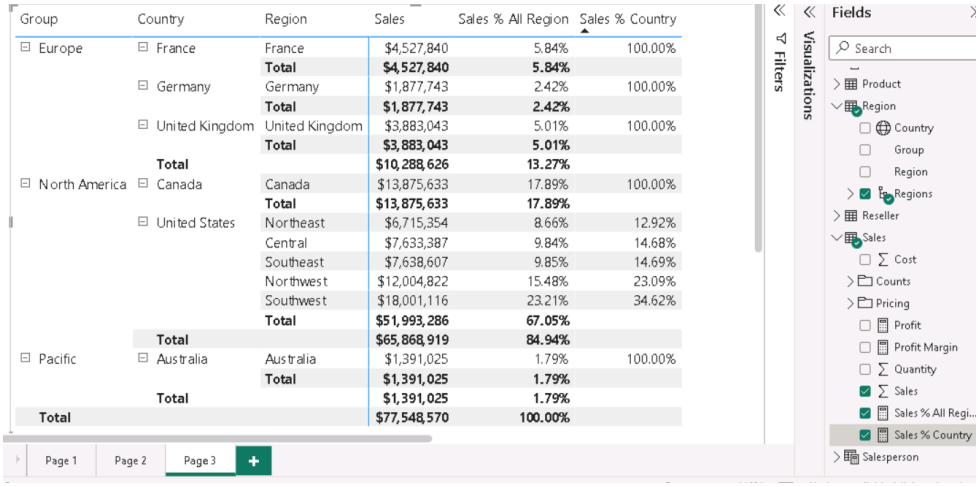
- Sales % Country

```
Sales % Country =
DIVIDE(
 SUM(Sales[Sales]),
 CALCULATE(
 SUM(Sales[Sales]),
 REMOVEFILTERS(Region[Region])
 )
)
```

*Removing filters on the **Region** column of the table, not all columns of the table. It means that any filters applied to the group or country columns are preserved. It will achieve a result that represents the sales as a percentage of country.*

  - To improve the readability of measure **Sales % Country**

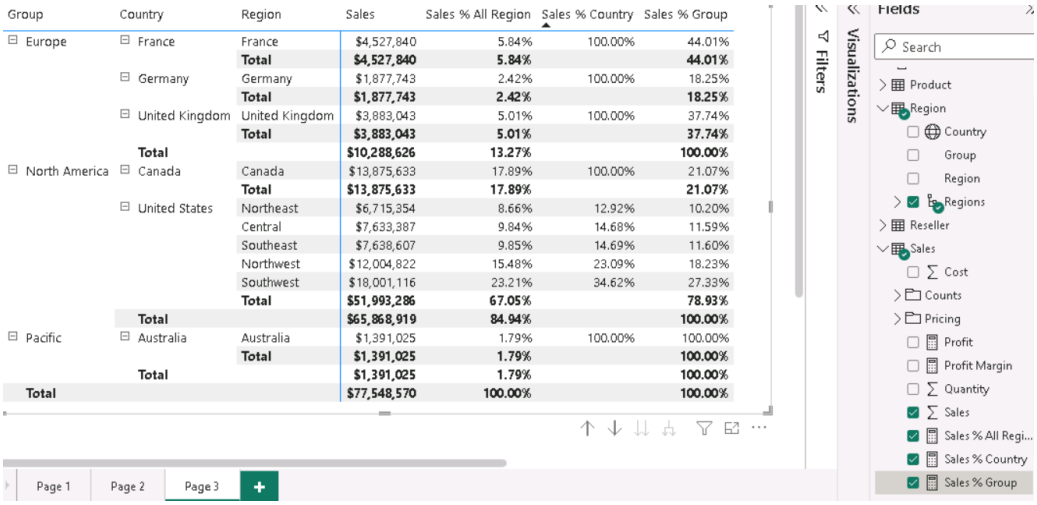
```
Sales % Country =
IF(
 ISINSCOPE(Region[Region]),
 DIVIDE(
 SUM(Sales[Sales]),
 CALCULATE(
 SUM(Sales[Sales]),
 REMOVEFILTERS(Region[Region])
 )
 )
)
```



*The IF() function uses the ISINSCOPE() function to test **whether the region column is the level in a hierarchy of levels**. When true, the DIVIDE() function is evaluated.*

- Sales % Group

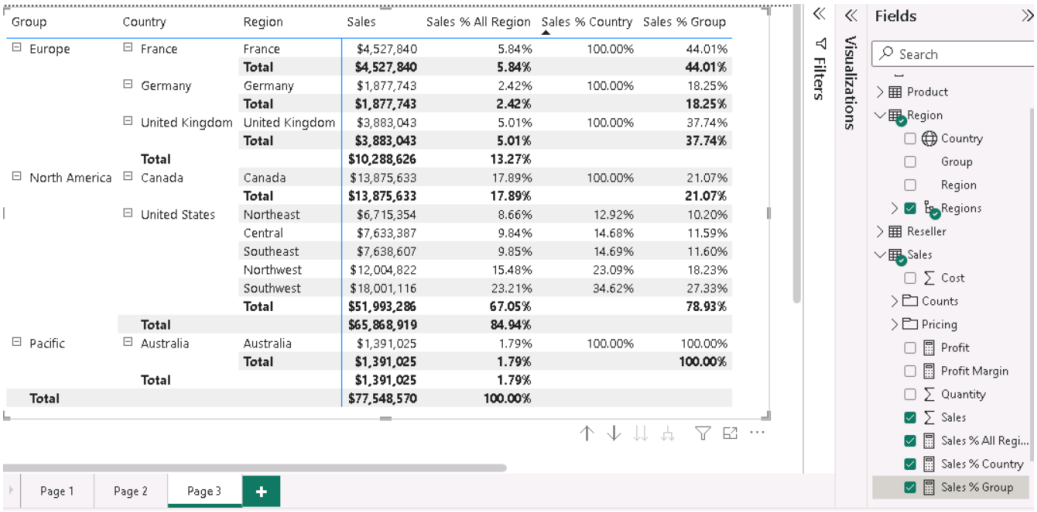
```
Sales % Group =
DIVIDE(
 SUM(Sales[Sales]),
 CALCULATE(
 SUM(Sales[Sales]),
 REMOVEFILTERS(
 Region[Region],
 Region[Country]
 )
 )
)
```



Group	Country	Region	Sales	Sales % All Region	Sales % Country	Sales % Group
Europe	France	France	$4,527,840	5.84%	100.00%	44.01%
		Total	**$4,527,840**	**5.84%**		**44.01%**
	Germany	Germany	$1,877,743	2.42%	100.00%	18.25%
		Total	**$1,877,743**	**2.42%**		**18.25%**
	United Kingdom	United Kingdom	$3,883,043	5.01%	100.00%	37.74%
		Total	**$3,883,043**	**5.01%**		**37.74%**
	Total		**$10,288,626**	**13.27%**		**100.00%**
North America	Canada	Canada	$13,875,633	17.89%	100.00%	21.07%
		Total	**$13,875,633**	**17.89%**		**21.07%**
	United States	Northeast	$6,715,354	8.66%	12.92%	10.20%
		Central	$7,633,387	9.84%	14.68%	11.59%
		Southeast	$7,638,607	9.85%	14.69%	11.60%
		Northwest	$12,004,822	15.48%	23.09%	18.23%
		Southwest	$18,001,116	23.21%	34.62%	27.33%
		Total	**$51,993,286**	**67.05%**		**78.93%**
	Total		**$65,868,919**	**84.94%**		**100.00%**
Pacific	Australia	Australia	$1,391,025	1.79%	100.00%	100.00%
		Total	**$1,391,025**	**1.79%**		**100.00%**
	Total		**$1,391,025**	**1.79%**		**100.00%**
Total			**$77,548,570**	**100.00%**		**100.00%**

- To improve the readability of measure **Sales % Group**

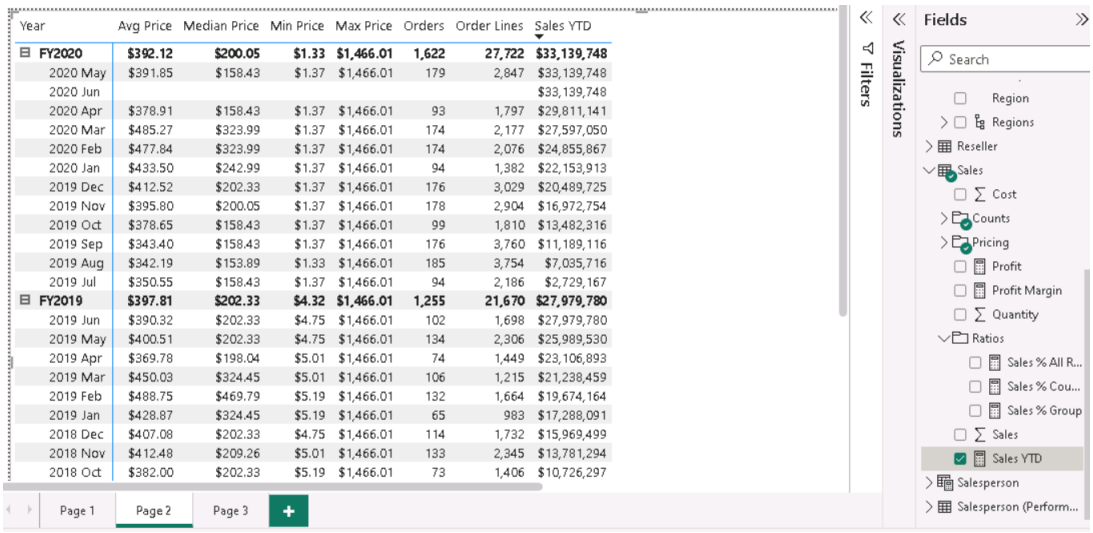
```
Sales % Group =
IF(
 ISINSCOPE(Region[Region])
 || ISINSCOPE(Region[Country]),
 DIVIDE(
 SUM(Sales[Sales]),
 CALCULATE(
 SUM(Sales[Sales]),
 REMOVEFILTERS(
 Region[Region],
 Region[Country]
 )
 )
 )
)
```



Work with Time Intelligence

Create a YTD measure

```
Sales YTD =
TOTALYTD(SUM(Sales[Sales]), 'Date'[Date], "6-30")
```



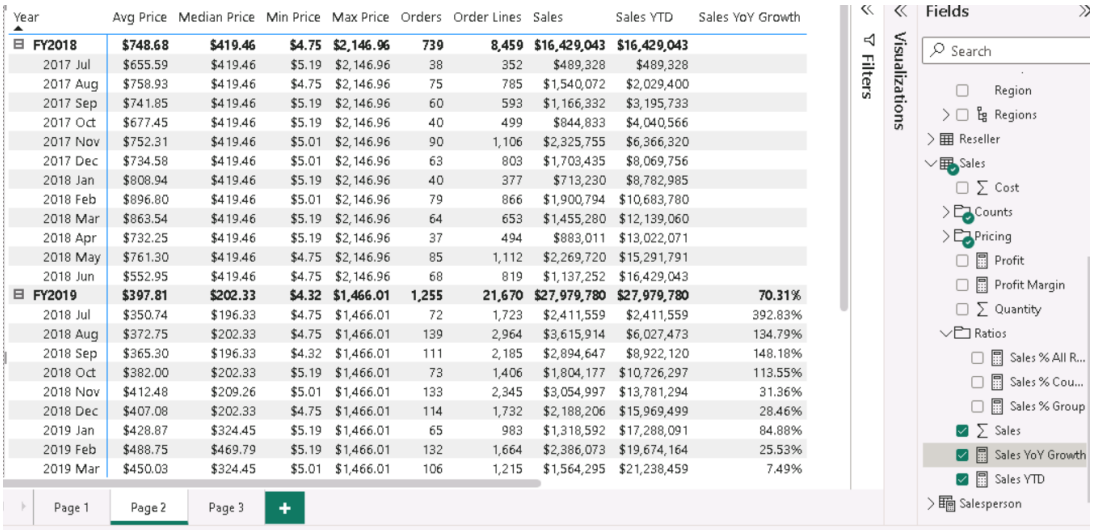
*The TOTALYTD() function evaluates an expression—in this case the sum of the **Sales** column—over a given date column. The date column must belong to a date table marked as a date table, as was done in the **Create DAX Calculations in Power BI Desktop** lab.*

The function can also take a third optional argument representing the last date of a year. By default, December 31 is the last date of the year. For Adventure Works, June in the last month of their year, and so "6-30" is used.

*For example, to compute YTD sales for September 2019 (the third month of the fiscal year), all filters on the **Date** table are removed and replaced with a new filter of dates commencing at the beginning of the fiscal year (July 1, 2019) and extending through to the last date of the in-context date period (September 30, 2019).*

Create a YoY growth measure (compare a year to the previous year)

```
Sales YoY Growth =
VAR SalesPriorYear =
 CALCULATE(
 SUM(Sales[Sales]),
 PARALLELPERIOD(
 'Date'[Date],
 -12,
 MONTH
 )
 )
RETURN
 DIVIDE(
 (SUM(Sales[Sales]) - SalesPriorYear),
 SalesPriorYear
 )
```



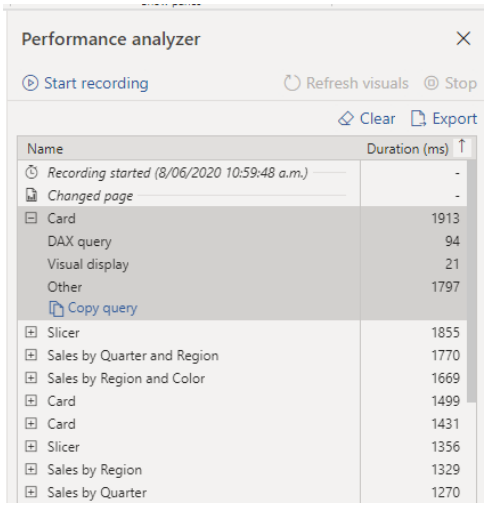
Year	Avg Price	Median Price	Min Price	Max Price	Orders	Order Lines	Sales	Sales YTD	Sales YoY Growth
FY2018	**$748.68**	**$419.46**	**$4.75**	**$2,146.96**	**739**	**8,459**	**$16,429,043**	**$16,429,043**	
2017 Jul	$655.59	$419.46	$5.19	$2,146.96	38	352	$489,328	$489,328	
2017 Aug	$758.93	$419.46	$4.75	$2,146.96	75	785	$1,540,072	$2,029,400	
2017 Sep	$741.85	$419.46	$5.19	$2,146.96	60	593	$1,166,332	$3,195,733	
2017 Oct	$677.45	$419.46	$5.19	$2,146.96	40	499	$844,833	$4,040,566	
2017 Nov	$752.31	$419.46	$5.01	$2,146.96	90	1,106	$2,325,755	$6,366,320	
2017 Dec	$734.58	$419.46	$5.01	$2,146.96	63	803	$1,703,435	$8,069,756	
2018 Jan	$808.94	$419.46	$5.19	$2,146.96	40	377	$713,230	$8,782,985	
2018 Feb	$896.80	$419.46	$5.01	$2,146.96	79	866	$1,900,794	$10,683,780	
2018 Mar	$863.54	$419.46	$5.19	$2,146.96	64	653	$1,455,280	$12,139,060	
2018 Apr	$732.25	$419.46	$5.19	$2,146.96	37	494	$883,011	$13,022,071	
2018 May	$761.30	$419.46	$4.75	$2,146.96	85	1,112	$2,269,720	$15,291,791	
2018 Jun	$552.95	$419.46	$4.75	$2,146.96	68	819	$1,137,252	$16,429,043	
FY2019	**$397.81**	**$202.33**	**$4.32**	**$1,466.01**	**1,255**	**21,670**	**$27,979,780**	**$27,979,780**	**70.31%**
2018 Jul	$350.74	$196.33	$4.75	$1,466.01	72	1,723	$2,411,559	$2,411,559	392.83%
2018 Aug	$372.75	$202.33	$4.75	$1,466.01	139	2,964	$3,615,914	$6,027,473	134.79%
2018 Sep	$365.30	$196.33	$4.32	$1,466.01	111	2,185	$2,894,647	$8,922,120	148.18%
2018 Oct	$382.00	$202.33	$5.19	$1,466.01	73	1,406	$1,804,177	$10,726,297	113.55%
2018 Nov	$412.48	$209.26	$5.01	$1,466.01	133	2,345	$3,054,997	$13,781,294	31.36%
2018 Dec	$407.08	$202.33	$4.75	$1,466.01	114	1,732	$2,188,206	$15,969,499	28.46%
2019 Jan	$428.87	$324.45	$5.19	$1,466.01	65	983	$1,318,592	$17,288,091	84.88%
2019 Feb	$488.75	$469.79	$5.19	$1,466.01	132	1,664	$2,386,073	$19,674,164	25.53%
2019 Mar	$450.03	$324.45	$5.01	$1,466.01	106	1,215	$1,564,295	$21,238,459	7.49%

PARALLELPERIOD() function to shift 12 months back from each date in filter context.

▼ Optimize model performance

Review performance

- Identify the slowest query in the semantic model → work through
- **View** tab > **Performance Analyzer** > **Start recording**
  - Make sure to start with a clear visual cache and a clear data cache
  - Review result:
    - **DAX query** - The time it took for the visual to send the query, along with the time it took Analysis Services to return the results.
    - **Visual display** - The time it took for the visual to render on the screen, including the time required to retrieve web images or geocoding.
    - **Other** - The time it took the visual to prepare queries, wait for other visuals to complete, or perform other background processing tasks. (If displays a long duration → optimize DAX queries for other visuals, or reduce the number of visuals in the report).



  - To analyze your queries in more detail → DAX Studio (open-source tool)

Resolve issues and optimize performance

- **Visuals:** fewer visuals → better performance
- **DAX query:** take ≤ 120 milliseconds
- **Relationships:** established correct relationships
- **Columns:** don't import columns not needed
- **Metadata:** Identify unnecessary columns, data errors, incorrect data types, volume of data loaded, etc.
    - Remove unnecessary columns/rows
    - Ensure data type is correct
    - Column quality, Column distribution, Column profile
    - File size of data loaded into PBI
    - Data refresh rate

  Benefit of analyzing the metadata → identify data inconsistences with your semantic model.
- **Auto date/time feature**
    - If your data source already defines a date dimension table, that table should be used to consistently define time within your organization → should disable **Auto date/time** → lower the size of your semantic model & reduce the refresh time
    - Enable/disable this **Auto date/time** option globally → applies to all PBI files
    - Enable/disable the option for the current file only
    - **File > Options and settings > Options (**either the **Global** or **Current File** section) > **Data Load** > **Time Intelligence**

Use variables (VAR) in DAX

Reduce cardinality

- **Cardinality levels in columns:** A column with lots of unique values → high level of cardinality. Lower cardinality = optimized performance.

- **Reduce relationship cardinality**
- **Reduce cardinality levels**

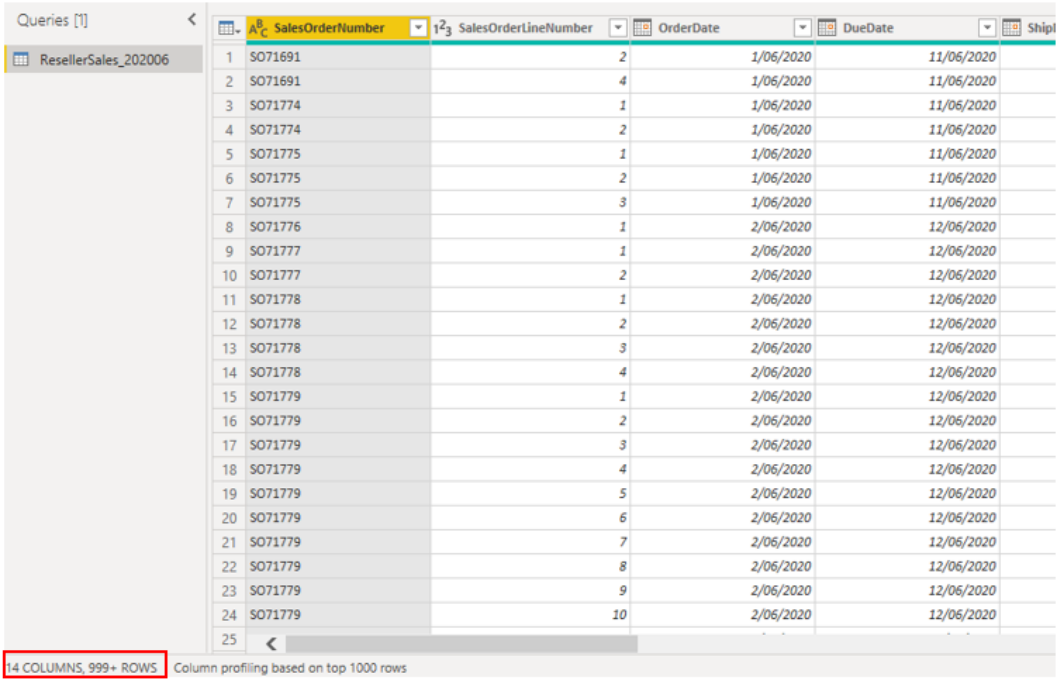
  For example, a source sales fact table stores one row for each individual transaction order. Significant data reduction could be achieved by grouping by date at month level. However, reporting at day level or an individual order level is no longer possible → lose the ability to drill through. This tradeoff could be mitigated by using a **mixed model design**.

  - The summarized sales data could be used to achieve high performance "summary" reporting.
  - A drill-through page could be created to display granular sales for specific (and narrow) filter context. It would include visuals based on a **DirectQuery** table to retrieve the sales order details.

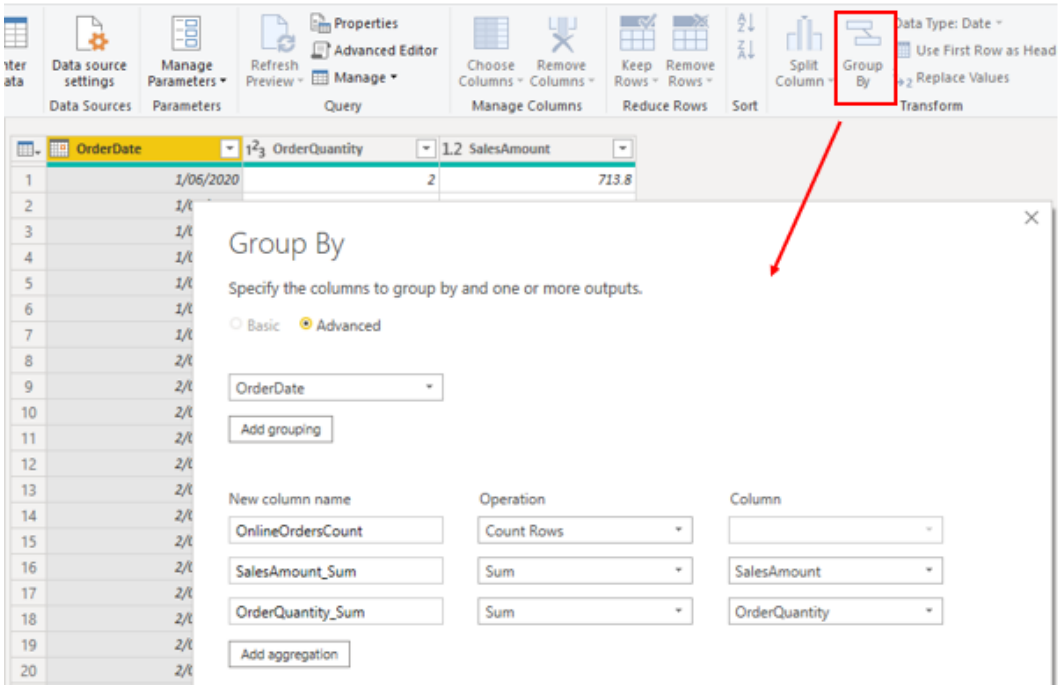
Optimize DirectQuery models with table level storage: link

Create and manage aggregations

Ex: Aggregate this sales data at the day level



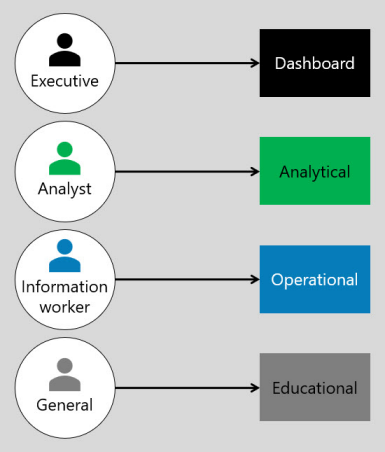
- If you have access to the database, you could create a table/view with the aggregation and then import that table/view into Power BI Desktop.
- In PBI Desktop, you can use Power Query Editor to create the aggregations step-by-step.
    - Choose Columns > **OrderDate, OrderQuantity, SalesAmount** columns
    - Group By



	OrderDate	OnlineOrdersCount	SalesAmount_Sum	OrderQuantity_Sum
1	1/06/2020	7	3221.28	14
2	2/06/2020	78	68495.62	225
3	3/06/2020	151	213860.64	738
4	4/06/2020	89	87484.01	218
5	5/06/2020	224	292106.92	978
6	6/06/2020	163	145808.57	572
7	7/06/2020	125	110115.85	557
8	8/06/2020	156	177334.37	441

▼ Build PBI visuals and reports

▼ Scope



Dashboard

- "How are we doing?" or "Are we there yet?"
- Analytics values, targets, statuses, and trends

Analytical reports

- "How are we doing?"
- "Why did that happen?" or "What might happen next?"
- Common: drill down, drill through, and tooltips (e.g., sales revenue from year, down to quarter, month, and day.)

Operational reports

- Monitor current or real-time data, make decisions, and act on those decisions
- Minimize the number of analytical features
- A good example of an operational report → allows monitoring of a manufacturing production line.

  When an unexpected event arises, such as equipment malfunction, a button could allow workers to start a maintenance request.

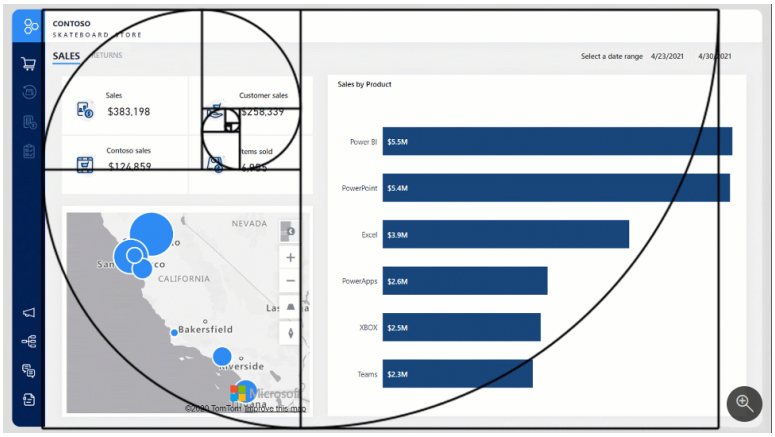
Educational reports

- Often used in journalism and by governments to disseminate information to large audiences that have varying levels of understanding of the subject
- Ex: COVID-19 vaccination progress

▼ Design

Layout

- **Placement** left to right and top to bottom
- **Balance** a page to have one large visual to draw initial attention → then supported by smaller visuals that provide context



- **Proximity** related visuals are placed near one another
- **Contrast** emphasize important objects
- **Repetition** Many key metrics are presented in single-value cards → this design allows report consumers to quickly understand and interpret the metrics.

Margin & Alignment

- **Margins**

- **Alignment**
  - Alignment of titles and legends *within visuals* should be consistent
  - When multiple visuals are on the report page → spacings between visuals should be consistent

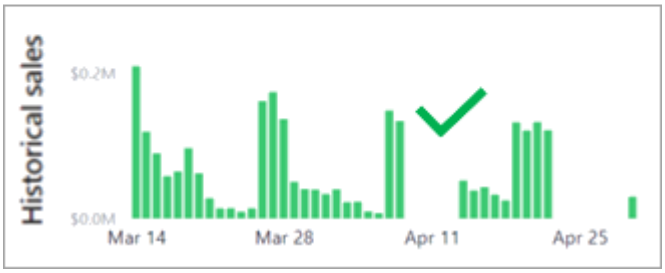
- **Implied sections:** aligning groups of visuals in close proximity

- **Explicit sections:** colored background shapes and spacing (highlighted with shading) separate the visuals into three sections

Select report visuals: link

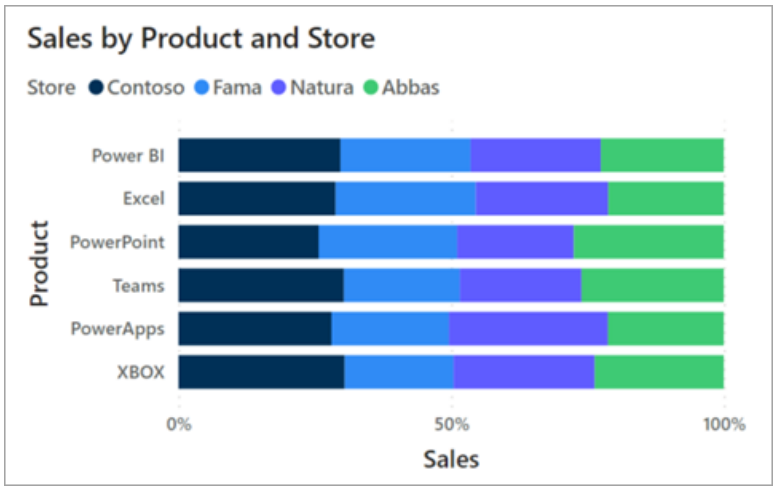
- **Time series visuals**

  If missing values are a possibility, a column chart might be a better visual choice because it will help to avoid the interpretation of a non-existent trend.

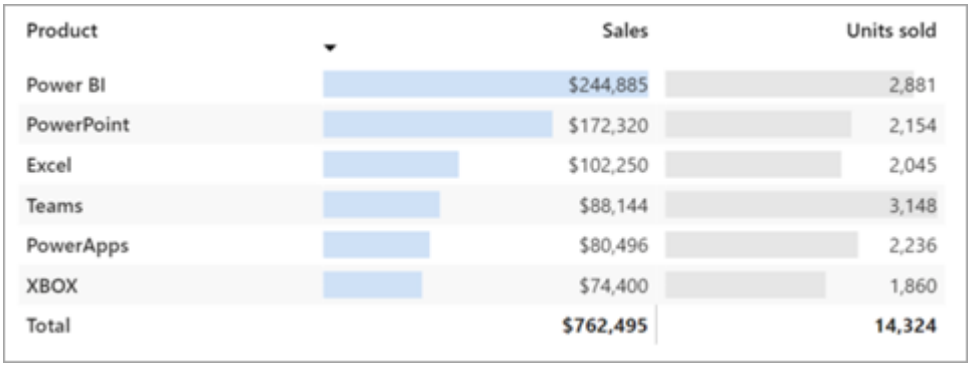


- **Proportional visuals**

  If necessary, you can reveal the actual values in a tooltip

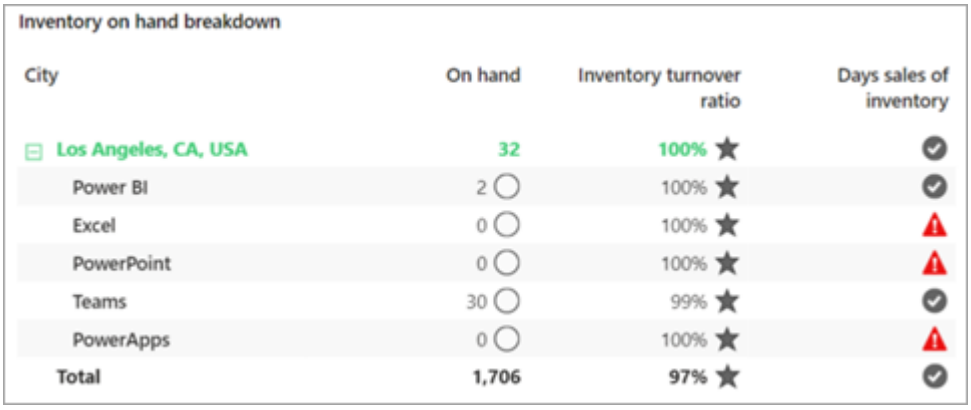


- **Grid visuals**
  - Table: A table doesn't support drill down to reveal detailed values



Product	Sales	Units sold
Power BI	$244,885	2,881
PowerPoint	$172,320	2,154
Excel	$102,250	2,045
Teams	$88,144	3,148
PowerApps	$80,496	2,236
XBOX	$74,400	1,860
Total	**$762,495**	**14,324**

- Matrix: A matrix visual allows drill down on the columns and rows to reveal detailed values
  - Can turn off stepped layout



Inventory on hand breakdown

City	On hand	Inventory turnover ratio	Days sales of inventory
Los Angeles, CA, USA	32	100%	
Power BI	2	100%	
Excel	0	100%	
PowerPoint	0	100%	
Teams	30	99%	
PowerApps	0	100%	
Total	**1,706**	**97%**	

- Conditional formatting for grid visuals:
  - Background color
  - Font color
  - Data bars
  - Icons

- **Performance visuals:**

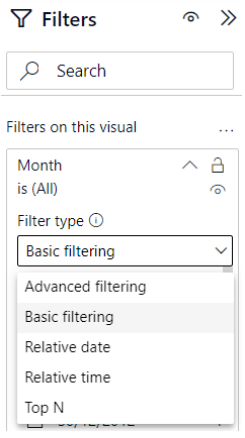
    - Gauge
    - KPI
    - Table, with conditional formatting
    - Matrix, with conditional formatting

- **Q&A visuals**
    - Allows for the entry of natural language questions that are answered by data visuals

▼ **Report filters**

Filters pane

- If the **Filters** pane isn't hidden, a report consumer can:
    - Clear the filter
    - Apply a new filter
    - Change the filter type: from basic to advanced (unless disabled for the report)



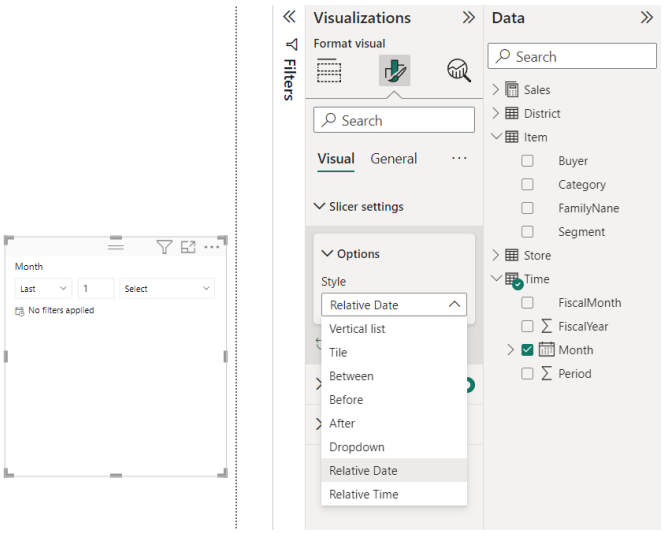
    - Search for values to filter by (unless disabled for the report)
- **Top N** filtering can only be achieved in the Filters pane

Slicer

✅ A visual that propagates filters to other visuals on the same page or (when synced) across other pages.

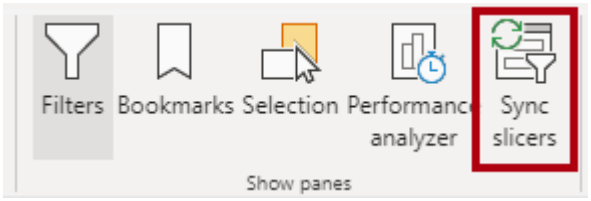
❌ Slicers apply page-level filters

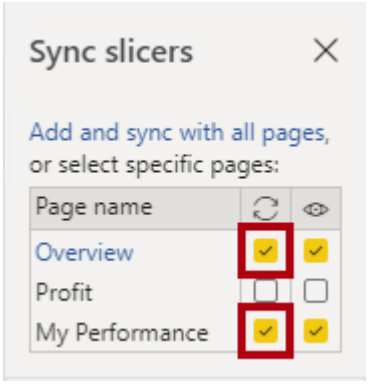
Relative Date/Time can also be used for slicer



Sync slicers

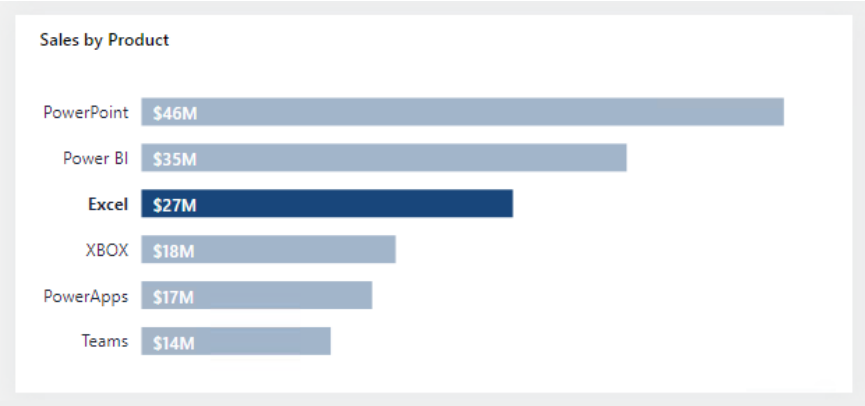
- Click on a slicer visual > **View** ribbon tab > **Show Panes** group > **Sync Slicers**





Cross filter (or cross highlighting)

- Report consumers: can temporarily cross filter visuals on the report page by selecting one or more elements in a visual
- Ex: select a single or multiple bars



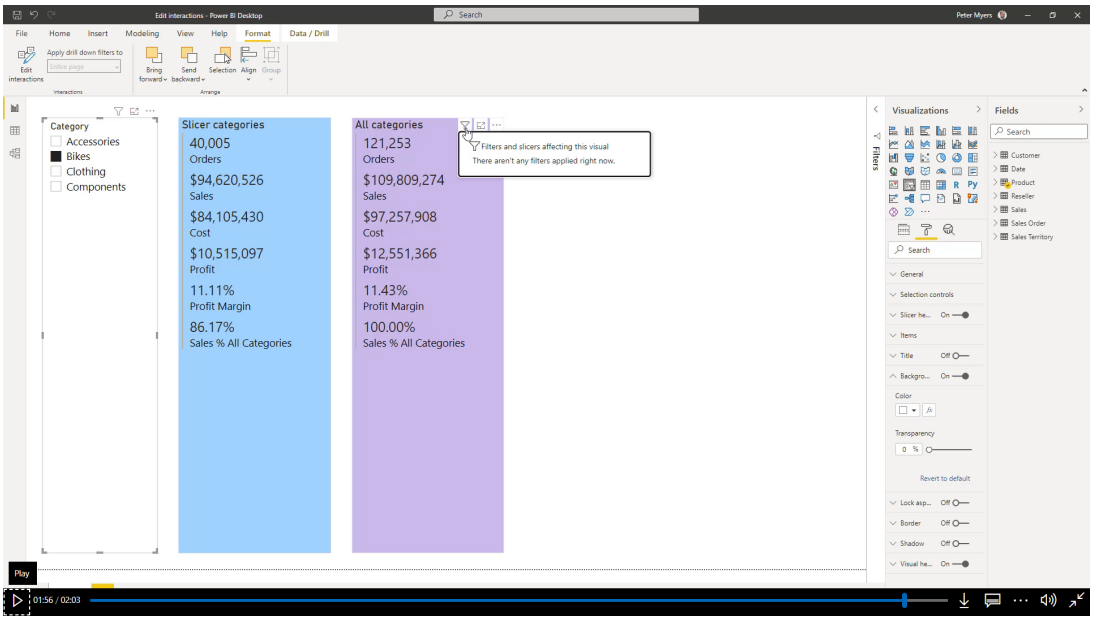
- Report author:
  - By default, cross filtering (or cross highlighting) works between any two visuals

    → Can edit visual interactions in either direction
  - Can disable cross filtering
  - Can modify to use cross highlighting instead

Advanced filtering techniques

- **Visual interactions:**

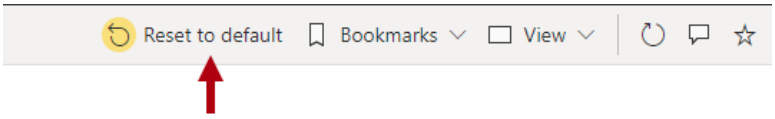
  Stop the filters from a slicer propagating to a visual: **Format** ribbon > **Edit interactions**



- **Drill-through**
- **Report tooltip**
- **Bookmarks**

Persistent filters

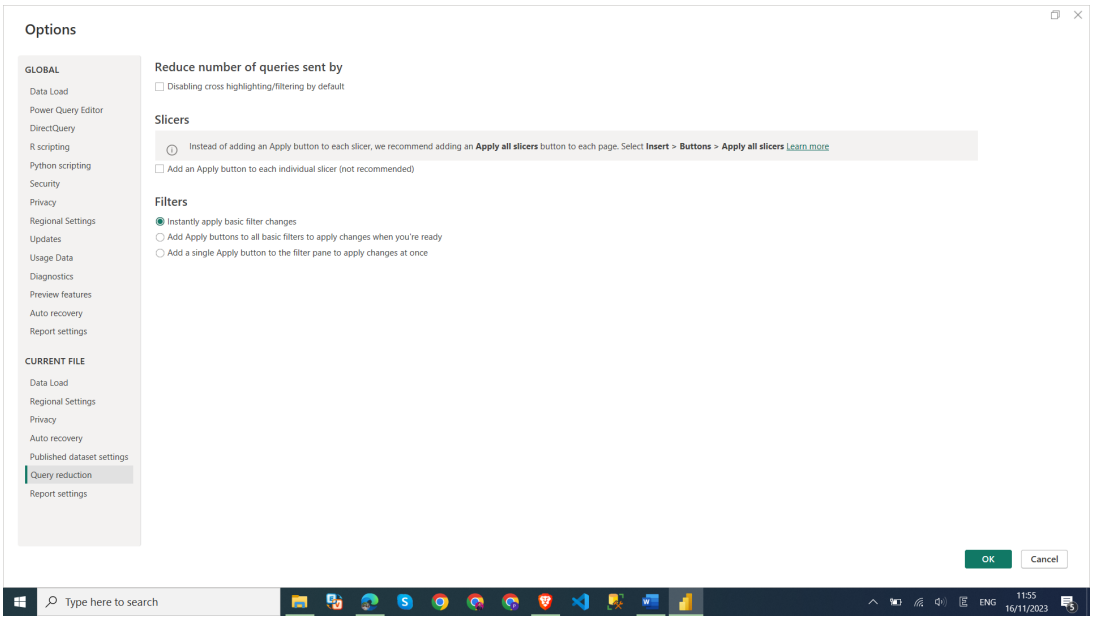
- Automatically applies the settings when the report consumer reopens the report
- Revert to default filters:



If the **Reset to default** button is not available → persistent filters is disabled for the report.

Query reduction: Create *Apply* button

- In slicer
- In Filters pane



Tips:

- Use either filters or slicers. Avoid using both filter techniques because it can create confusion.
- In the **Filters** pane, consider locking or hiding visual-level filters to avoid confusing report consumers. (Often, report consumer shouldn't modify or see visual-level filters.)
- Create a bookmark to reset all slicers to default values. Then, add a button to the page to invoke the bookmark. For example, the button could be captioned as **Reset slicers**.
- When a requirement is in place to lay out many slicers, consider creating a page that is dedicated to showing all slicers. For example, the page could be named **Slicers**. Sync the slicers to other pages and then set the slicers as hidden on those pages. This design technique will require that report consumers should always go to the **Slicers** page to modify slicer settings. To help them, you can add a page navigation button at a consistent location on each page so that they can easily return to the **Slicers** page.
- Consider using other visuals in place of slicers. Be sure to teach report consumers how to cross filter by using these visuals.

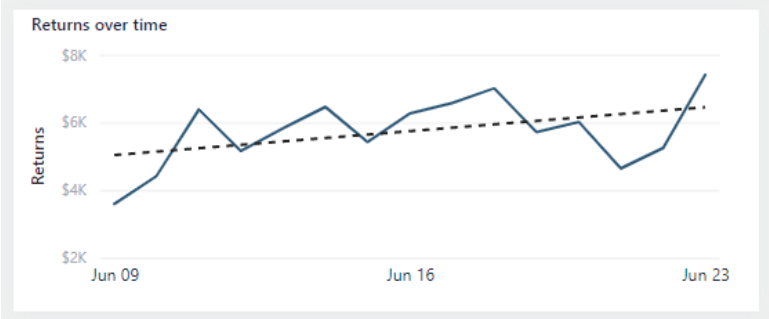
For more information, see Create buttons in Power BI reports.

For more information, see Create page navigation.

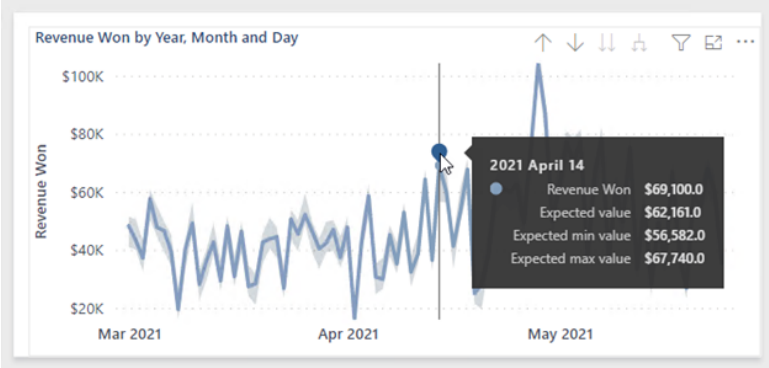
▼ Enhance reports

Highlight values

- Conditional formatting
- Overlaid analytics (trend lines)



- Anomaly detection link



- Specialized visuals

  Highlight values; specifically, the **Key influencers** visual and the **Decomposition Tree** visual.

Design reports that behave like apps: Button action

- **Q&A** action
  - Q&A for Power BI business users.
  - Unlike the **Q&A** visual, a button that is assigned the action doesn't occupy significant space on the report page.
- **Web URL** action
  - Opens the URL by using the default web browser
  - Can produce a URL that appends filter context to the query string. For example, if the report consumer filters the page by a single customer, the

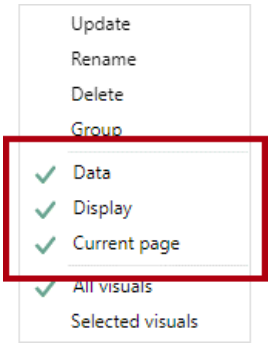
measure can return a URL to an external system that includes the customer key in the query string.

- **Work with bookmarks:** link 📌

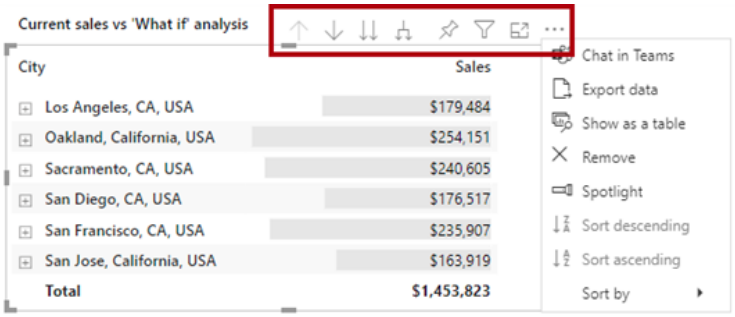
  Lab instructions

  → How to create a bookmark to swap visuals

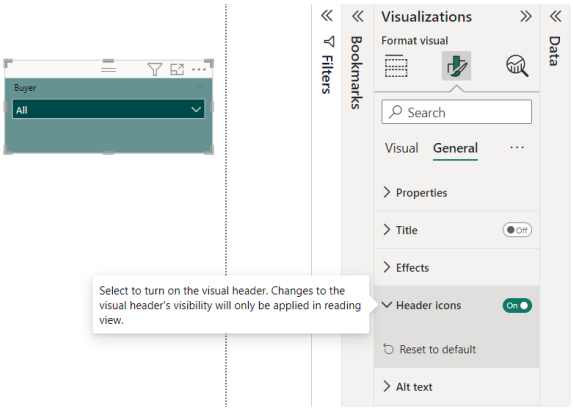
  → Understand “Data” bookmark state



Visual headers



- Always leave sufficient space for the visual headers to appear in the upper-right section of objects.
- Turns off visual headers for objects that don't need them

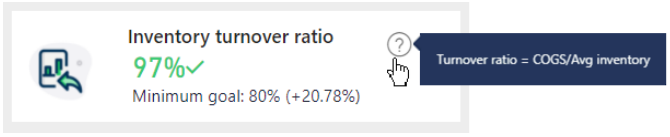


- Disable all visual header icons would include the **More options** (**...**) menu

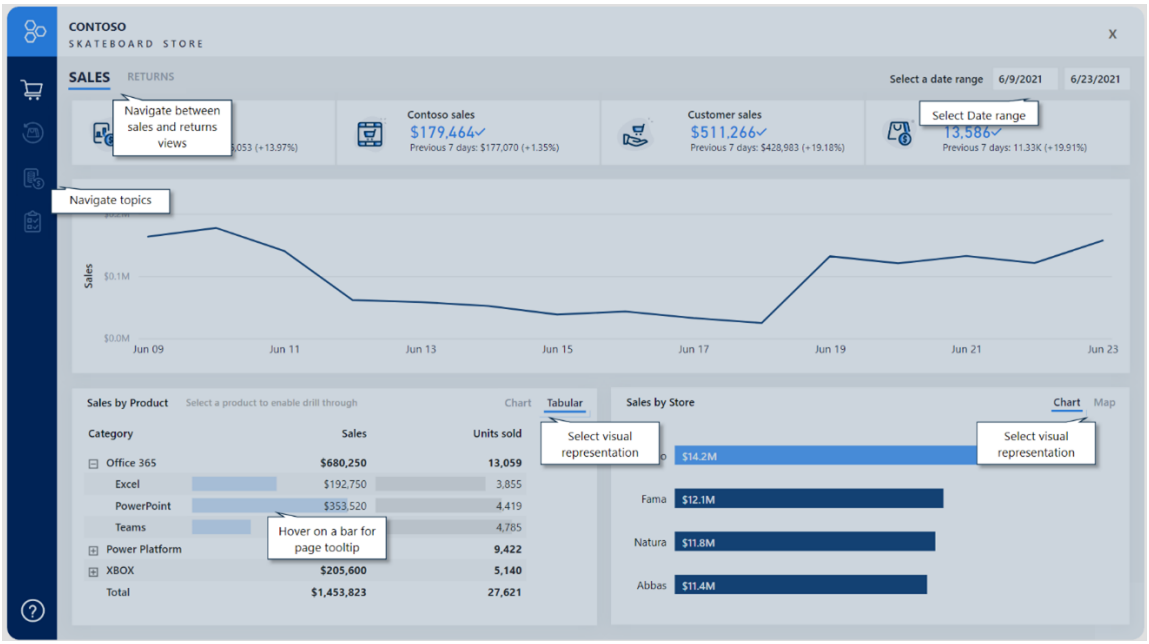
  Example use case: When report contains some sensitive data that shouldn't be exported → disabling the More options (…) icon → report consumers can't export data

Built-in assistance

- Information page
  - Dedicate an entire report page that includes instructions and definitions.
  - Configure these buttons to use the **Information** or **Help** icon.
- Visual header tooltip icon



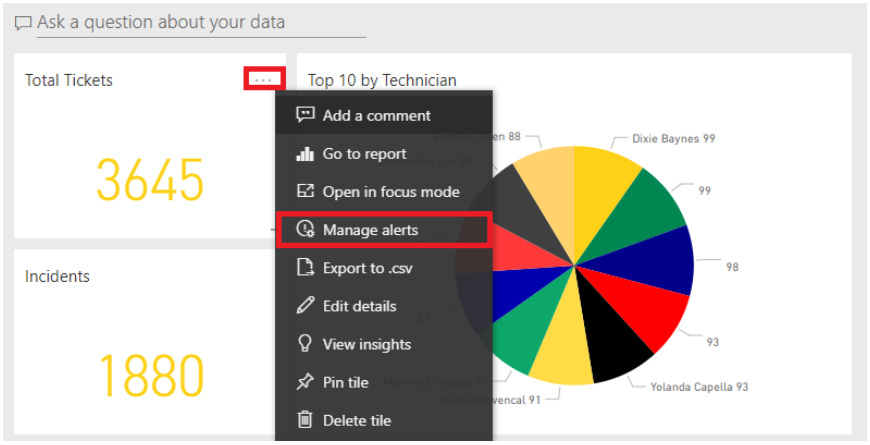
- Button overlay
  - Use buttons and bookmarks to overlay an image, shape, or text box that provides built-in assistance.
  - Ex: Create a bookmark to swap visuals



▼ Create dashboards in PBI

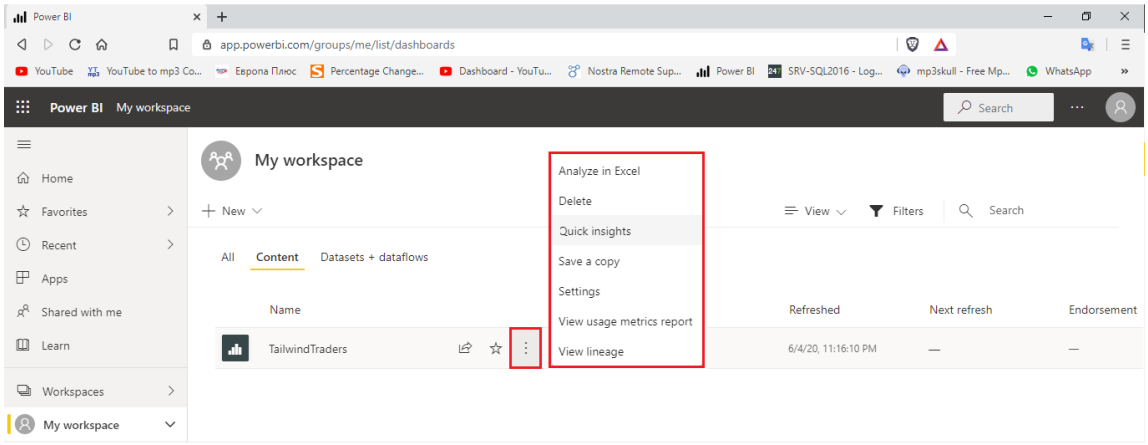
Configure data alerts

- PBI service only
- Can only be set on specific visuals such as KPI cards, gauges, and cards.

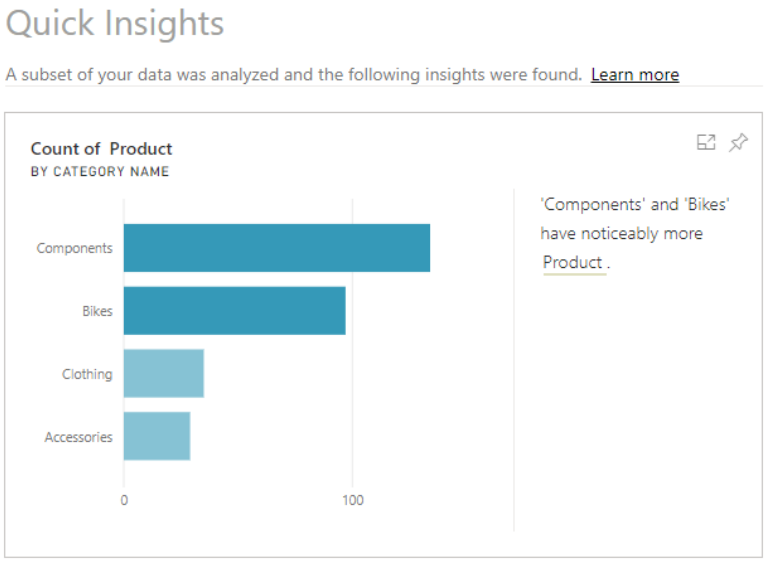


Review Quick insights

- PBI service only
- Only works with data that is imported to Power BI



- Can add Quick insights result to your dashboard by selecting **Pin visual**



Dashboard theme

Go to a dashboard > Edit dropdown arrow > **Dashboard theme**

Pin an entre report page: link

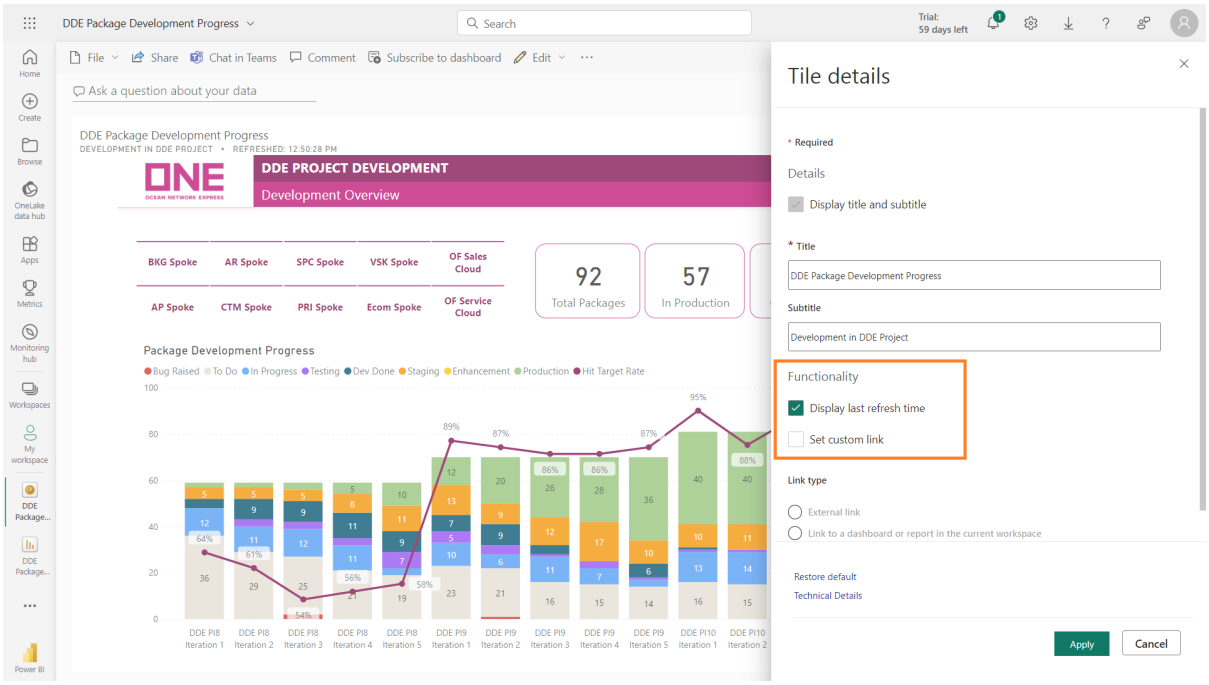
- When you pin an entire page, the tiles are *live* → can interact with them on the dashboard.
- Any changes made to any of the visualizations back in the report editor are reflected on the dashboard when the page is refreshed

Configure a real-time dashboard: link

Configure data classification: link

- PBI service only → admin role
- Go to dashboard > Hover over the ellipsis (**...**) by the name of the dashboard and then select **Settings**
- If contains highly sensitive information → mark as **High Impact**

Display refresh time on a dashboard tile

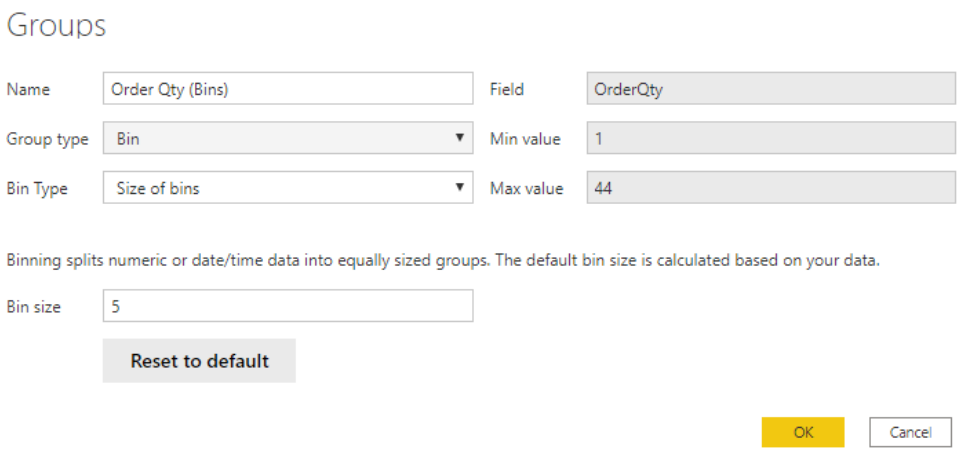


▼ Perform analytics in PBI

Statistical summary: link

- **Top N analysis:** using a Q&A visual, Top N filter, or writing DAX
- **Histogram**

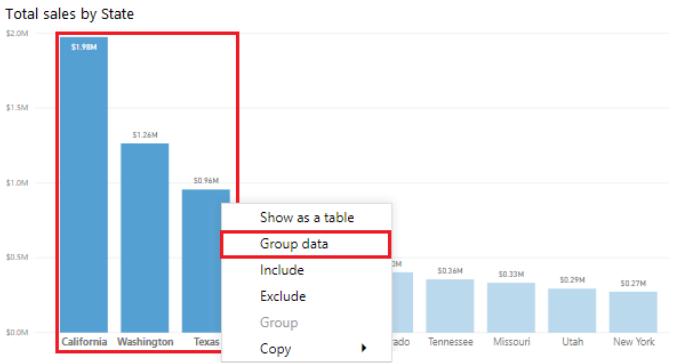
- From a bar chart > Group your numerical and time field data into "bins" of equal size

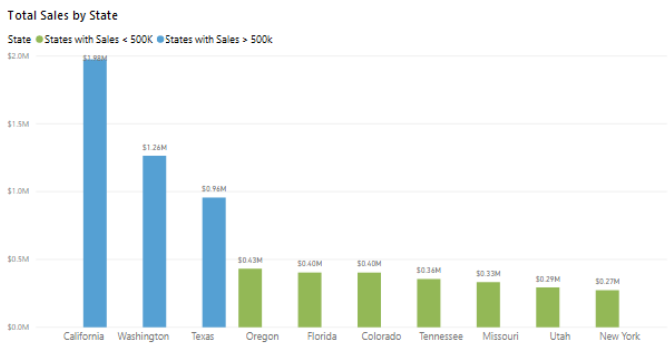


- **Identify outliers:** link
  - Scatter plot
  - DAX

Grouping & Clustering

- **Clustering:** link
- **Group data:**



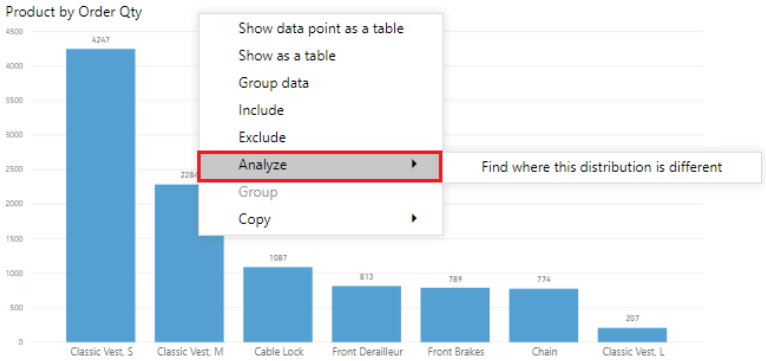


Time series analysis:

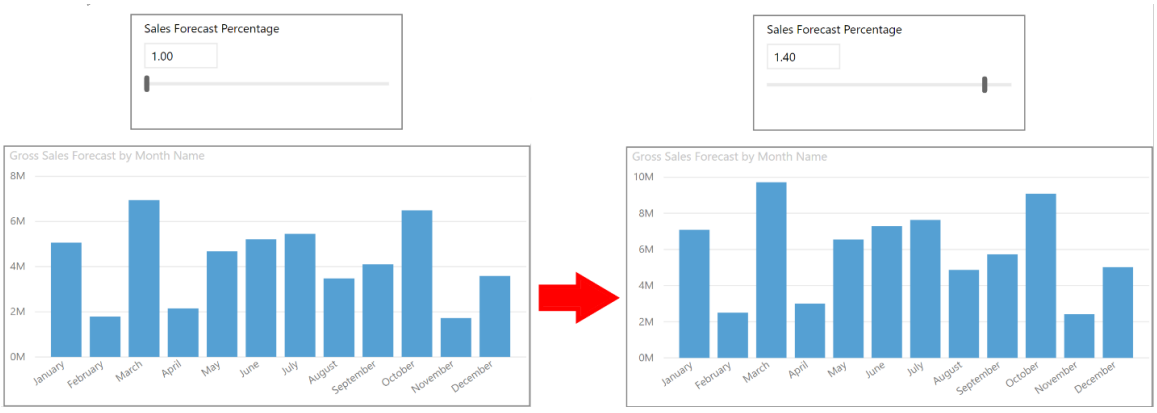
- Microsoft AppSource: Play Axis (Dynamic Slicer)

Use the Analyze feature

- Options that are available will depend on the data point selected.
- (**+**) icon to add it to your report

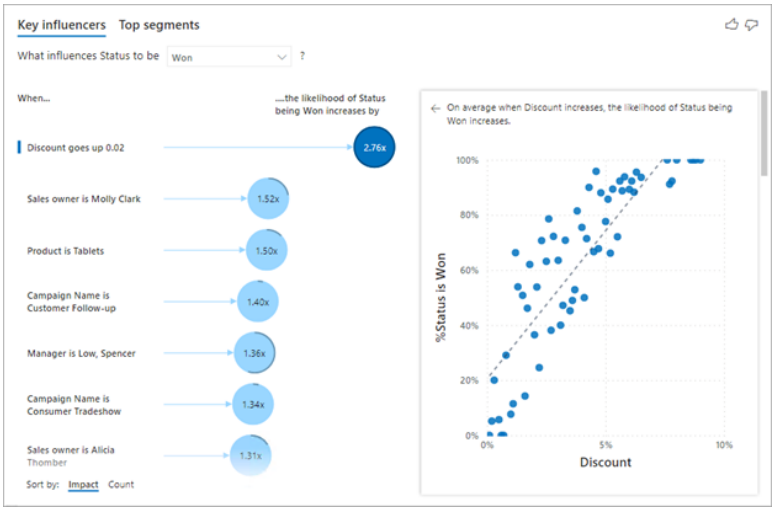


Create what-if parameters: link

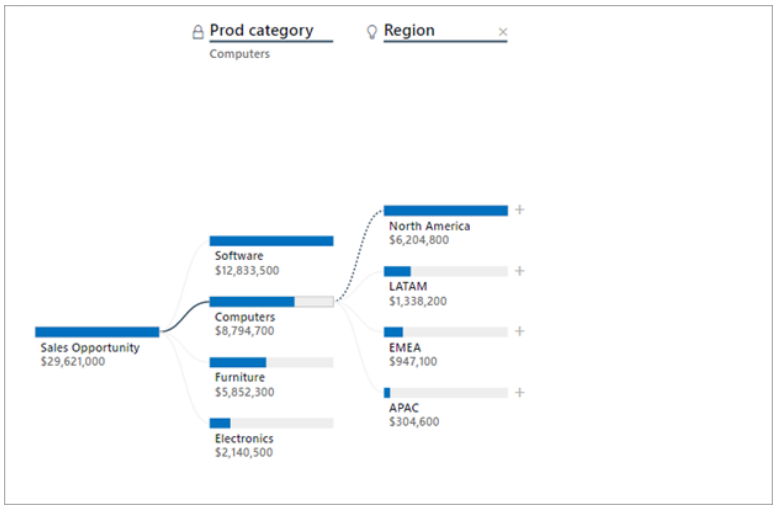


Specialized visuals: link

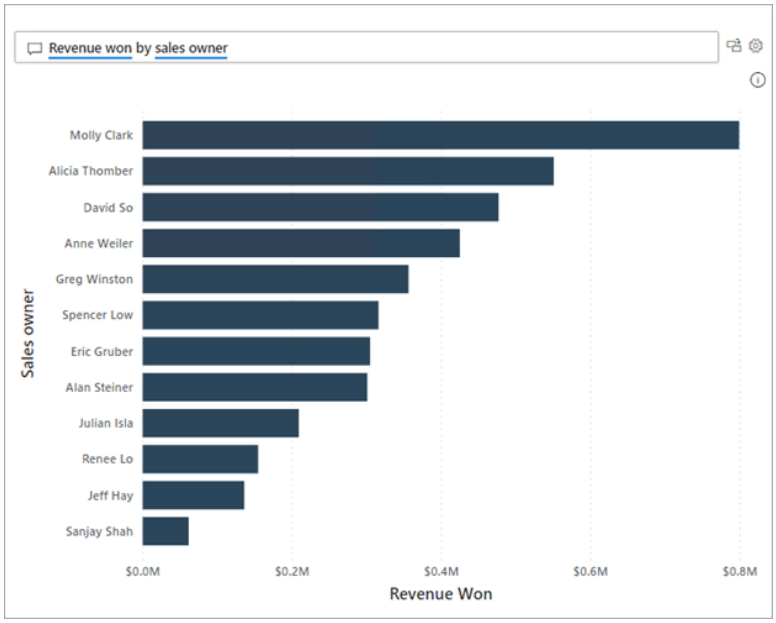
- Key influencers



- Decomposition tree
  - Lock icon: split added by report author
  - Light bulb icon: split suggested by AI



- Q&A visual
  - To optimize the Q&A → enhance the semantic model with synonyms and terms.
  - Hide fields, such as fields that are used in model relationships → restrict their use in Q&A.
  - Add suggested questions that become prompts in the **Q&A** visual.



▼ Paginated report

- Link
- PBI Report Builder
- All of the material in the SQL Server Reporting Service (SSRS) documentation will apply to PBI Report Builder

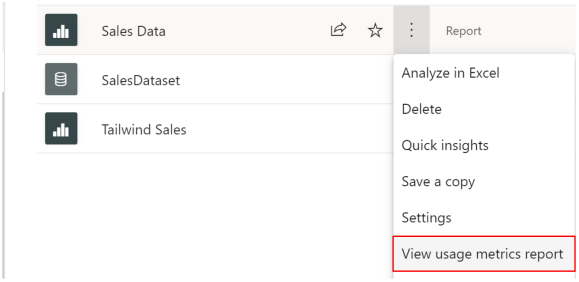
▼ Manage workspace and semantic model

Workspace

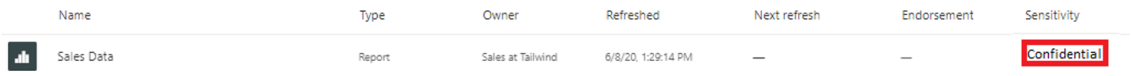
- A workspace → share reports, build dashboards, and collaborate with your teams.
- Workspaces appropriate security requirements → easier to share content

---

- Monitor usage and performance



- Deployment pipeline (Premium): Dev - Test - Prd
- View data lineage
- Sensitivity labels



Semantic model

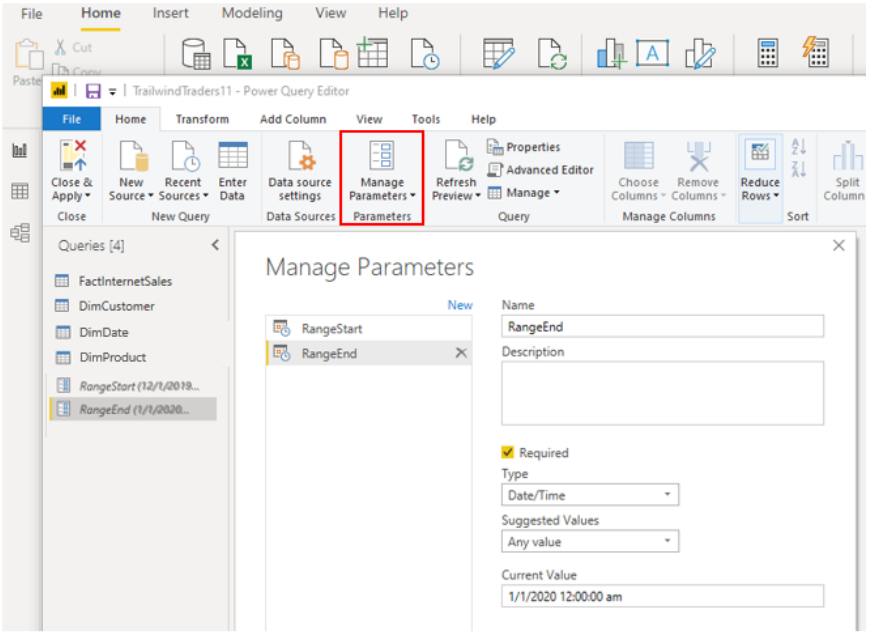
- Published to organization's workspace → collaborate

- One semantic model can be used for different business reasons → create multiple reports
- Reuse semantic models → productivity boost

- Set up and maintain a gateway to access on-premises data source from the cloud
- Refresh semantic models:
  - Schedule refresh → need to have created a gateway connection
  - Manual refresh
  - ▼ Manual incremental refresh

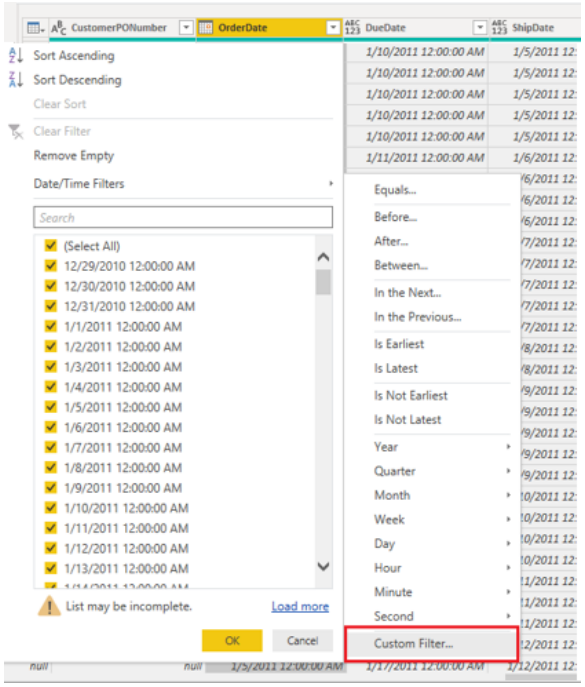
    Define filter parameters

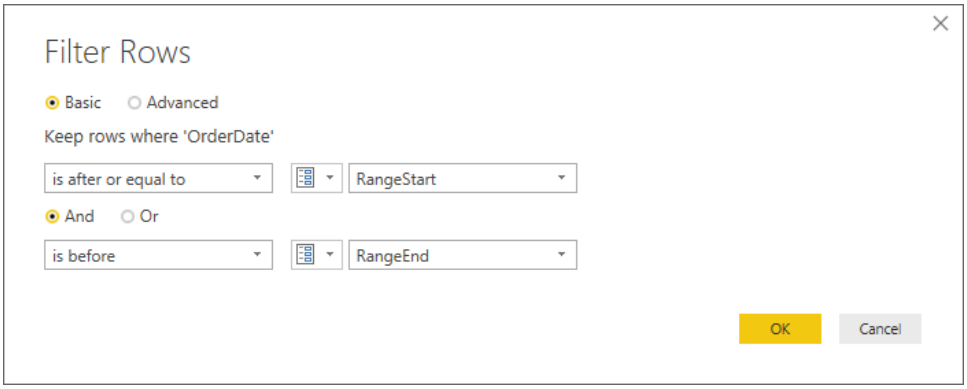
    1. Power Query > **Home** tab > **Manage Parameters**.
    2. Add two new parameters: **RangeStart** and **RangeEnd**
    3. **Current Value** → configure the start and end of where incremental refresh should occur



Apply the filter

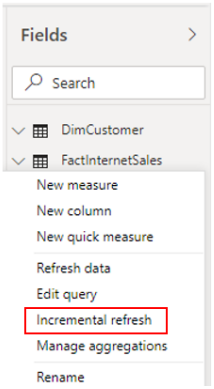
Go to the applicable **Date** column > select **Custom Filter**

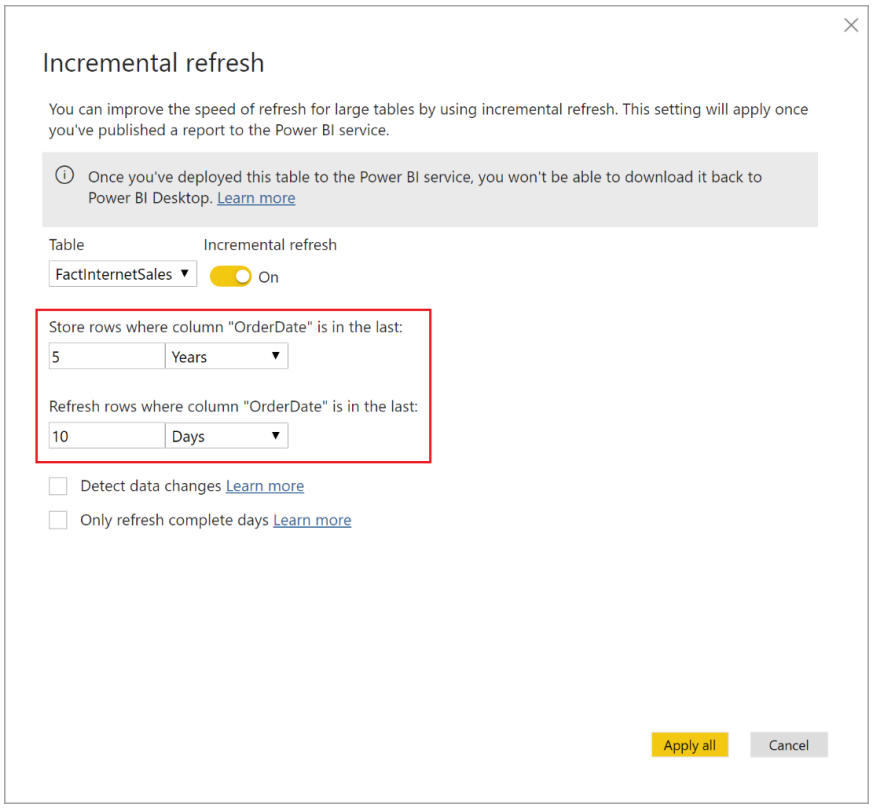




Define the incremental refresh policy

Right-click the applicable table > select **Incremental refresh**





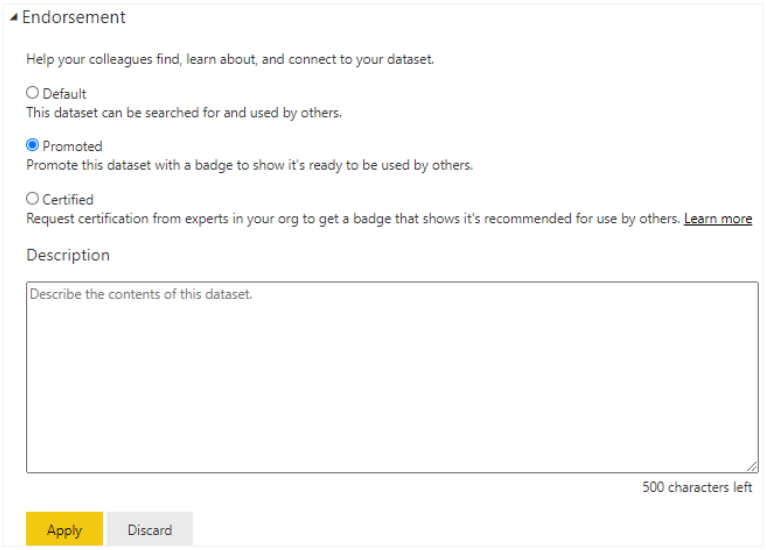
Publish to Power BI service

Defined the incremental refresh policy in PBI Desktop → Need to publish the report to PBI Service to apply that refresh policy

---

- Direct users to the most up-to-date and highest-quality semantic models in your workspaces (link)
  - Promotion does not need specific permissions

- Certification requires permission from the semantic model owner to access to the semantic model



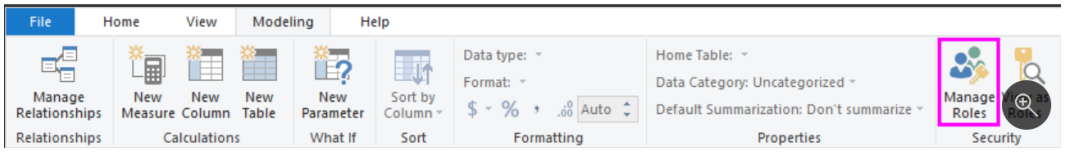
- Query caching (link)
  - Use the local caching services of PBI to process query results (aka cloud resources on PBI Premium)

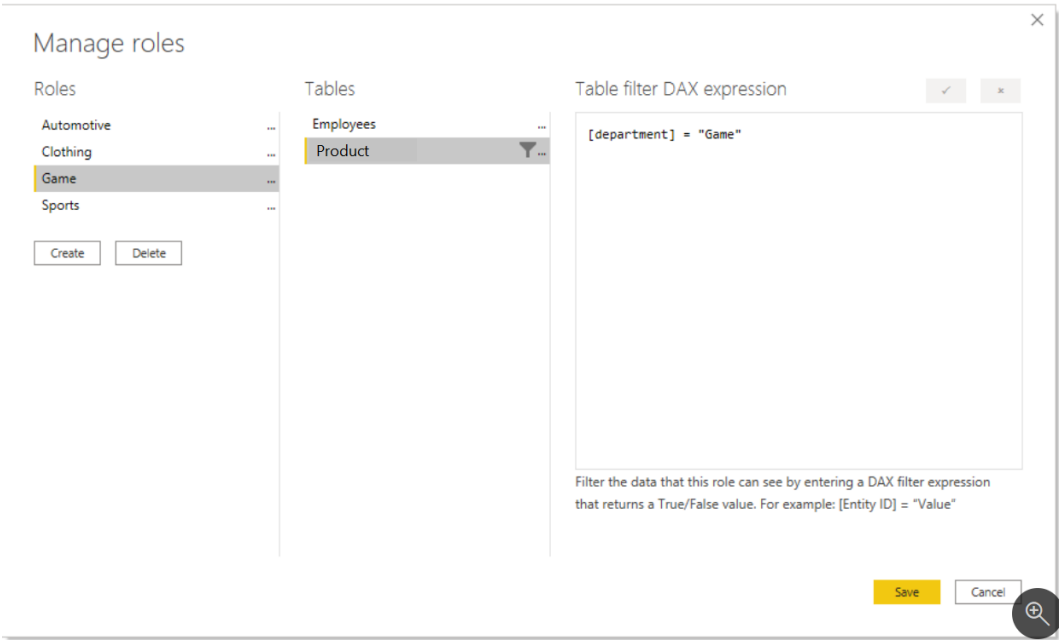
▼ Model security

RLS → control access to specific data row

▼ Configure RLS (static method)

- **Create RLS roles**

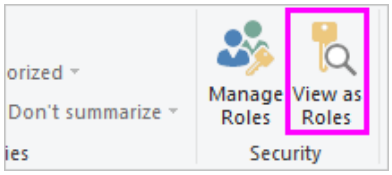




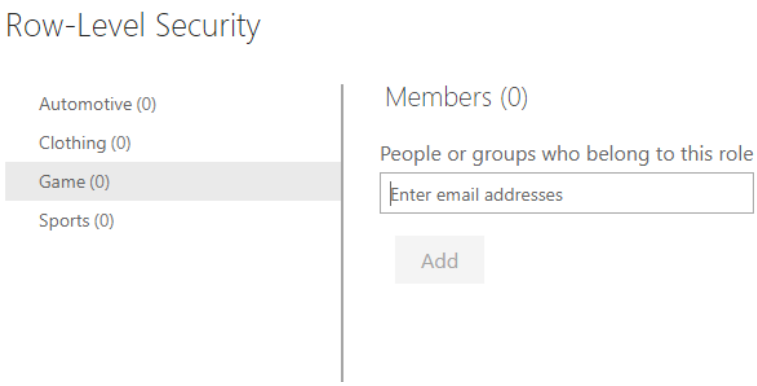
→ Apply the DAX filter to a **dimension table** (as was done with the Products table)

→ Keep the DAX filter as simple as possible

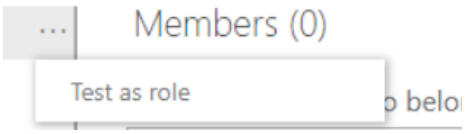
- **Test roles**



- **Deploy the report to PBI service**
- **Add members to the role in PBI service:** Go to your workspace in PBI service > Find the semantic model with the same name as your report > Select (**...**) button > **Security**



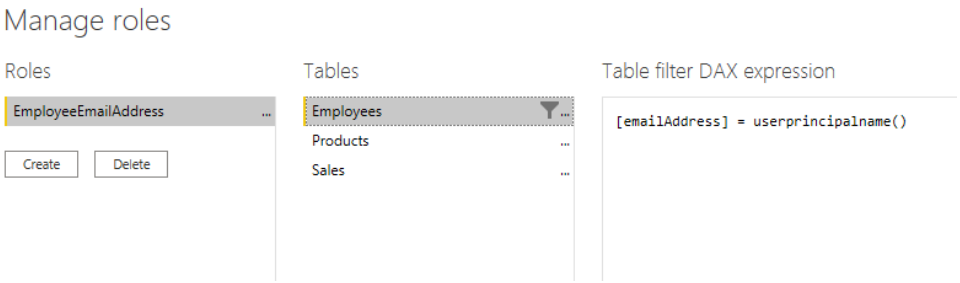
- **Test roles in PBI Service**



▼ Configure RLS (dynamic method)

- **Dynamic rules**
  - `USERPRINCIPALNAME` – Returns the Power BI authenticated user as a text value.
  - `CUSTOMDATA` – Returns the **CustomData** property passed in the connection string. Non-Power BI reporting tools that connect to the dataset by using a connection string can set this property, like Microsoft Excel.

- **Create roles**



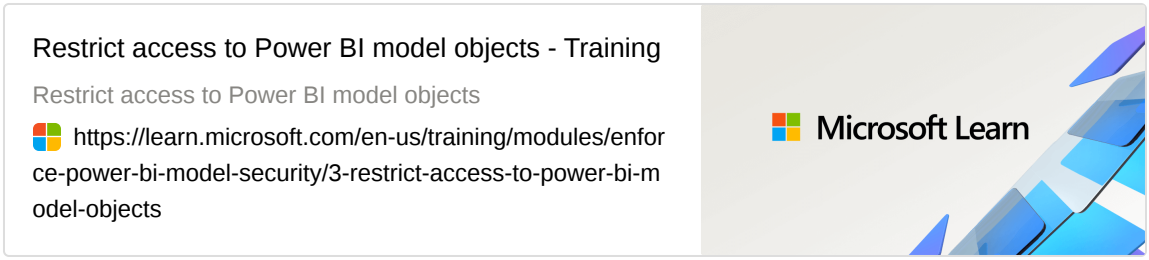
→ The `USERPRINCIPALNAME()` function will compare the email address from the Employees table with the email that the user entered when signing in to Power BI service.

→ If Russel King uses the email address `russel@tailwindtraders.com` to sign in to Power BI service, the system will compare that value to the email address in the Employees table.

→ Assuming that a relationship has been created between dimension Employees and fact Sales, Russel will only see his four sales.

Object-level security (OLS) → control access to entire tables or columns; secure metadata

▼ Set up OLS



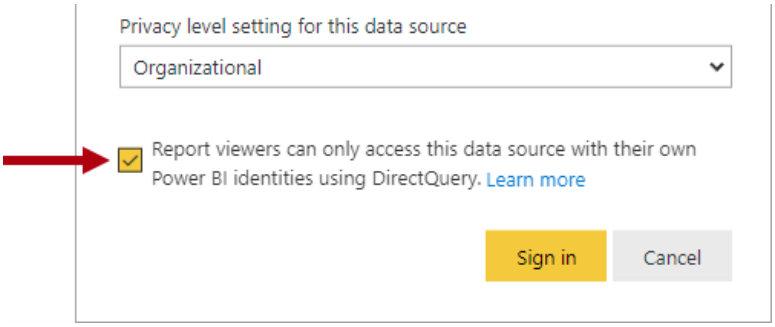
Set up role mappings

- Role mapping involves assigning Microsoft Entra security objects to roles.
- Security objects can be user accounts or security groups.
- For more information about setting up RLS, see:
  - Row-level security (RLS) with Power BI
  - Row-level security (RLS) guidance in Power BI Desktop

Use single sign-on (SSO) for DirectQuery sources

- When your data model has DirectQuery tables and their data source supports SSO, the data source can enforce data permissions.

- Consider that Adventure Works has an Azure SQL Database for their sales operations
    - The database enforces RLS in various database tables.
    - You can:
        1. Create a DirectQuery model that connects to this database without roles
        2. Publish it to PBI service
        3. When set the data source credentials in the PBI service → you enable SSO



    - When report consumers open PBI reports, PBI passes their identity to the data source. The data source then enforces RLS based on the identity of the report consumer.