TEAM ACADEMY

Model Data in Power BI

Module 3

Introduction

TEAM ACADEMY

- Creating a great data model is one of the most important tasks that a data analyst can perform in Microsoft Power BI.
- By doing this job well, you help make it easier for people to understand your data, which will make building valuable Power BI reports easier for them and for you.

A good data model offers the following benefits:

- Data exploration is faster.
- Aggregations are simpler to build.
- Reports are more accurate.
- Writing reports takes less time.
- Reports are easier to maintain in the future.

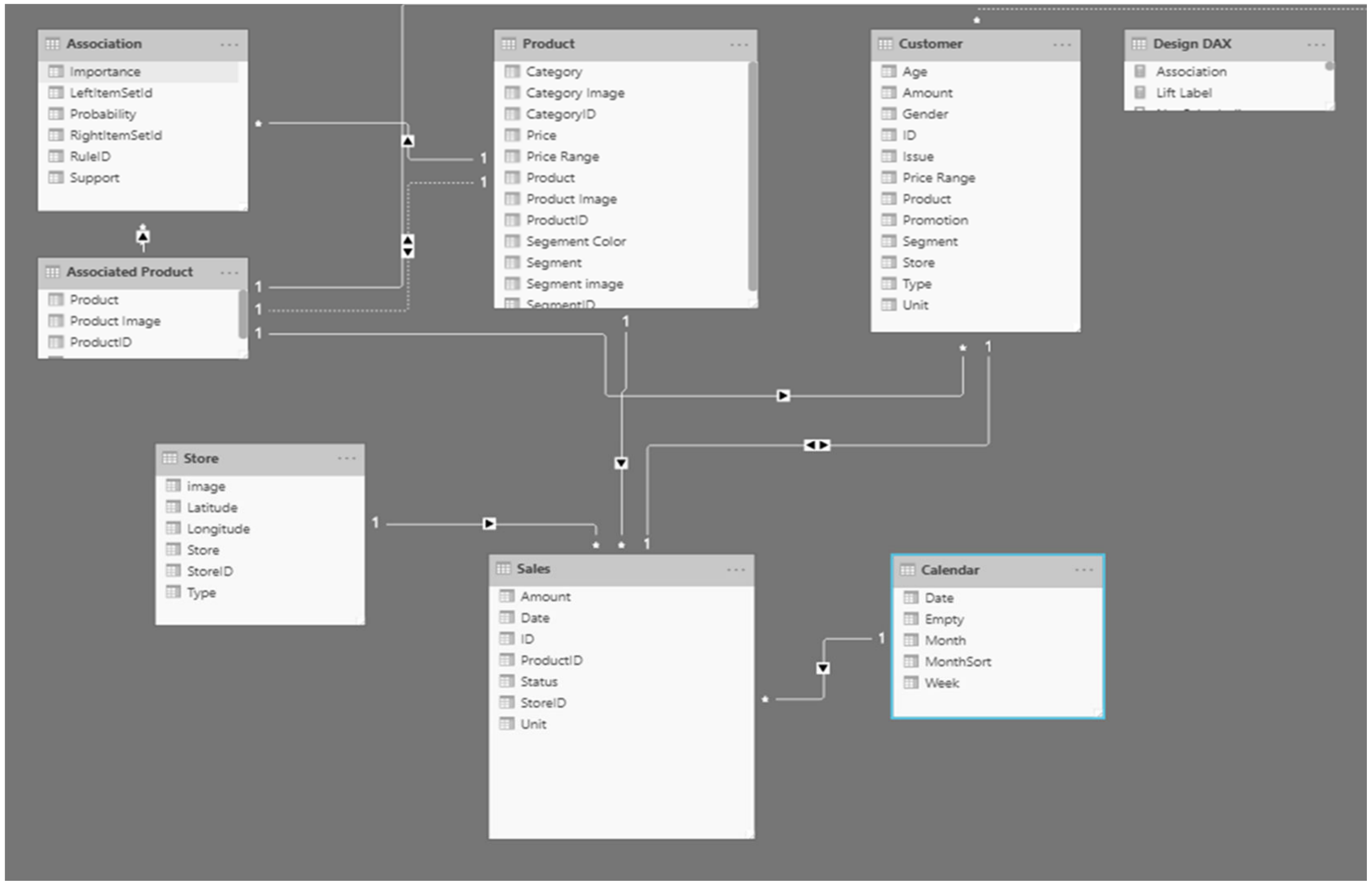
Continue …

TEAM ACADEMY

- Providing set rules for what makes a good data model is difficult because all data is different, and the usage of that data varies.
- Generally, a smaller data model is better because it will perform faster and will be simpler to use.
- However, defining what a smaller data model entails is equally as problematic because it's a heuristic and subjective concept.
- Typically, a smaller data model is comprised of fewer tables and fewer columns in each table that the user can see.

Example of Data Model

TEAM ACADEMY



Continue …

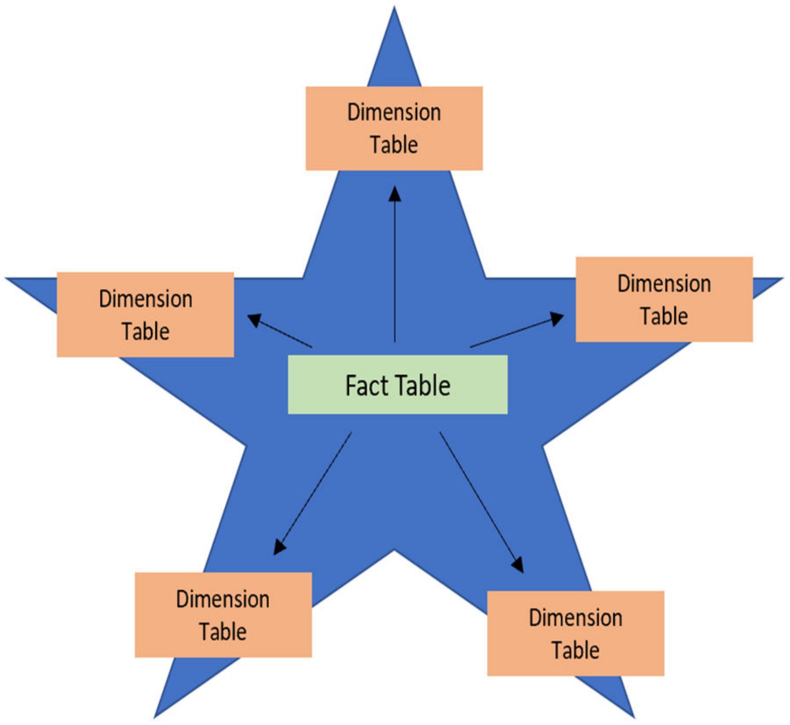
TEAM ACADEMY

- that enables you to pull one table from Microsoft Excel and another from Power BI allows relationships to be built from tables with different data sources, a powerful function a relational database.
- You would then create the relationship between those two tables and treat them as a unified dataset.
- Now that you have learned about the relationships that make up the data schema, you'll be able to explore a specific type of schema design, the star schema, which is optimized for high performance and usability.

Star Schemas

TEAM ACADEMY

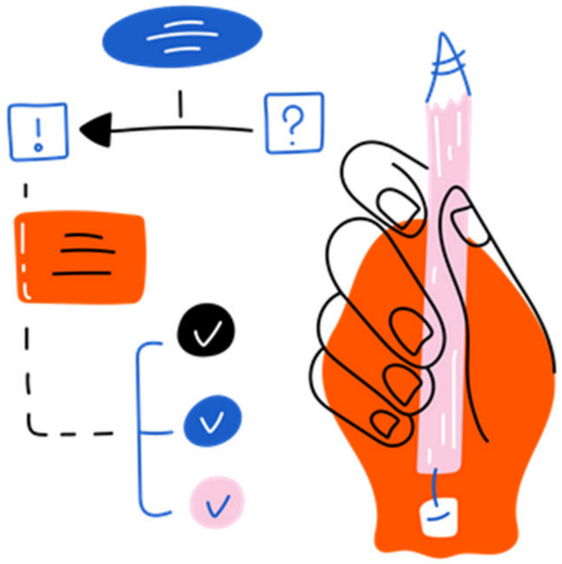
- You can design a star schema to simplify your data.
- It's not the only way to simplify your data, but it is a popular method; therefore, every Power BI data analyst should understand it.
- In a star schema, each table within your dataset is defined as a dimension or a fact table, as shown in the following visual.
- **Fact tables** contain observational or event data values: sales orders, product counts, prices, transactional dates and times, and quantities. Fact tables can contain several repeated values.
- **Dimension tables** contain the details about the data in fact tables: products, locations, employees, and order types. These tables are connected to the fact table through key columns. Dimension tables are used to filter and group the data in fact tables.



Work with Tables

A simple table structure will:

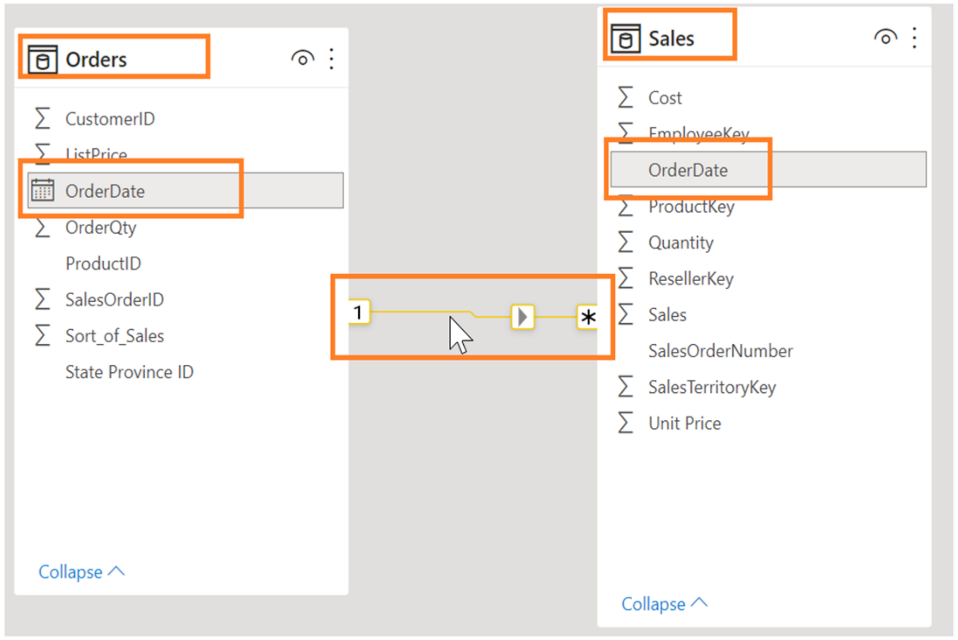
- Be simple to navigate because of column and table properties that are specific and user-friendly.
- Have merged or appended tables to simplify the tables within your data structure.
- Have good-quality relationships between tables that make sense.



Configure Data Model And Build Relationships Between Tables

TEAM ACADEMY

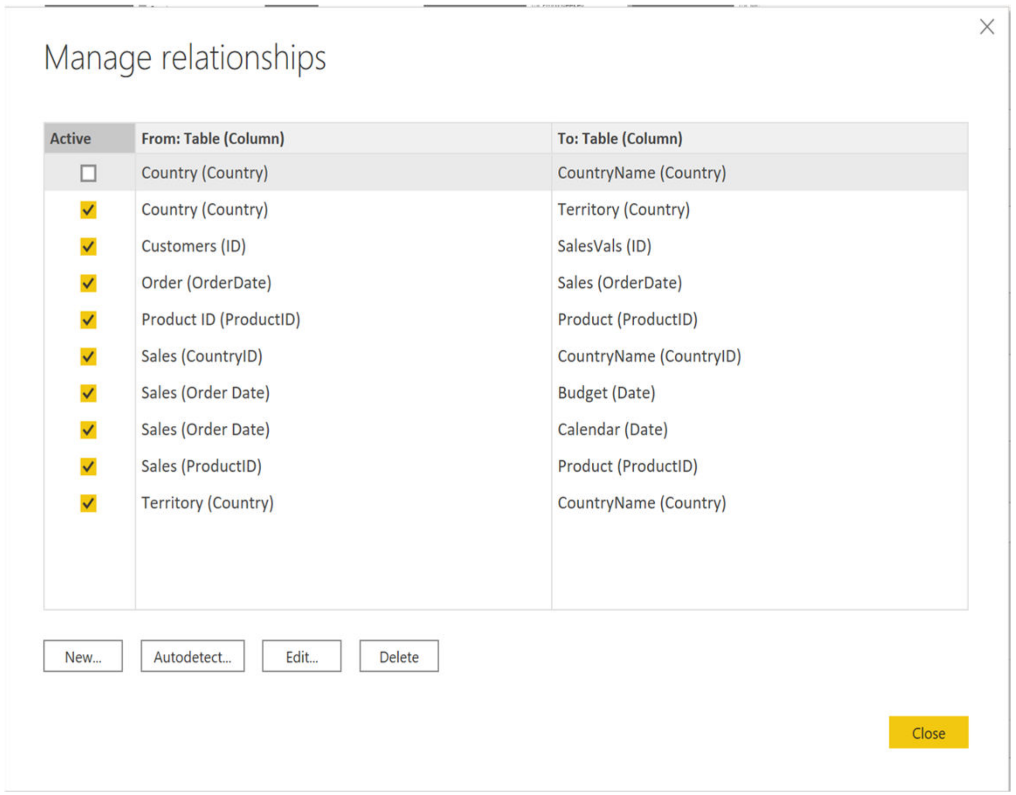
- Assuming that you've already retrieved your data and cleaned it in Power Query, you can then go to the **Model** tab, where the data model is located.
- The following image shows how the relationship between the **Order** and **Sales** tables can be seen through the **Order Date** column.



Continue …

TEAM ACADEMY

While the **Manage Relationships** feature allows you to configure relationships between tables, you can also configure table and column properties to ensure organization in your table structure.



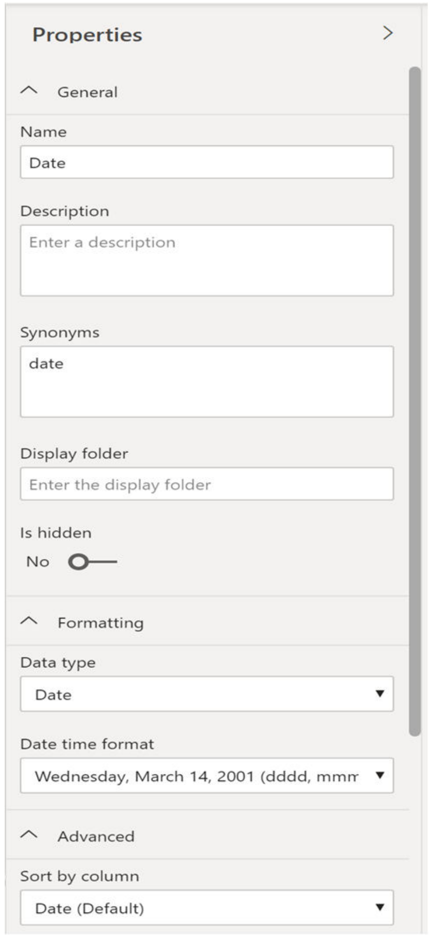
Manage relationships

Active	From: Table (Column)	To: Table (Column)
☐	Country (Country)	CountryName (Country)
✓	Country (Country)	Territory (Country)
✓	Customers (ID)	SalesVals (ID)
✓	Order (OrderDate)	Sales (OrderDate)
✓	Product ID (ProductID)	Product (ProductID)
✓	Sales (CountryID)	CountryName (CountryID)
✓	Sales (Order Date)	Budget (Date)
✓	Sales (Order Date)	Calendar (Date)
✓	Sales (ProductID)	Product (ProductID)
✓	Territory (Country)	CountryName (Country)

Configure Table and Column Properties

TEAM ACADEMY

- The **Model** view in Power BI desktop provides many options within the column properties that you can view or update.
- A simple method to get to this menu to update the tables and fields is by Ctrl+clicking or Shift+clicking items on this page.
- Under the **General** tab, you can:
- 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 a column into a folder to further organize the table structure.
- Hide or show the column.


Properties

General

Name: Date

Description: Enter a description

Synonyms: date

Display folder: Enter the display folder

Is hidden: No

Formatting

Data type: Date

Date time format: Wednesday, March 14, 2001 (dddd, mmm

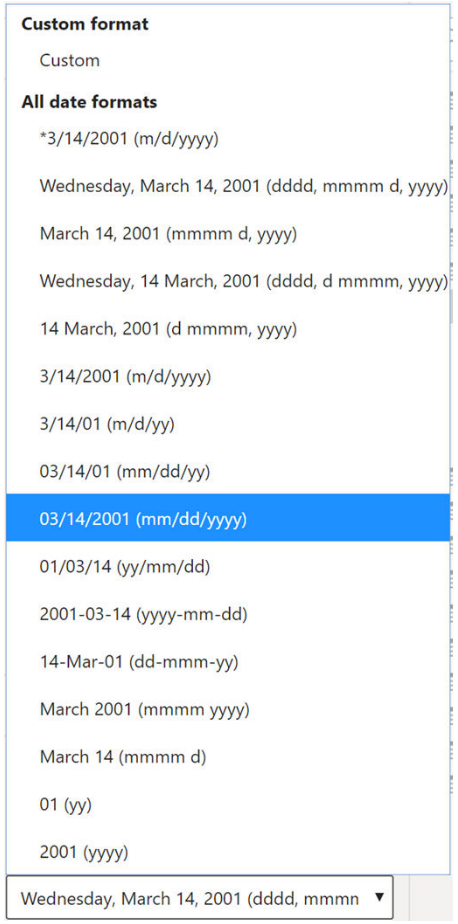
Advanced

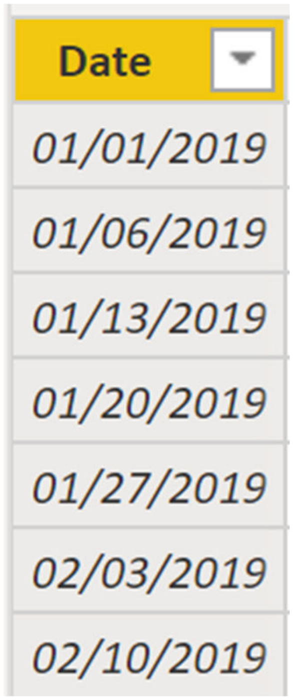
Sort by column: Date (Default)

Continue ...

TEAM ACADEMY

- Under the **Formatting** tab, you can:
- Change the data type.
- Format the date.
- For instance, suppose that the dates in your column are formatted, as seen in the previous screenshot, in the form of "Wednesday, March 14, 2001".
- If you want to change the format so that the date was in the "mm/dd/yyyy" format, you would select the drop-down menu under **All date time formats** and then choose the appropriate date format, as shown in the following figure.
- After selecting the appropriate date format, return to the **Date** column, where you should see that the format has indeed changed, as shown in the following figure.





Date
01/01/2019
01/06/2019
01/13/2019
01/20/2019
01/27/2019
02/03/2019
02/10/2019

Continue ...

TEAM ACADEMY

Under the **Advanced** tab, you can:

- Sort by a specific column.
- Assign a specific category to the data.
- Summarize the data.
- Determine if the column or table contains null values.

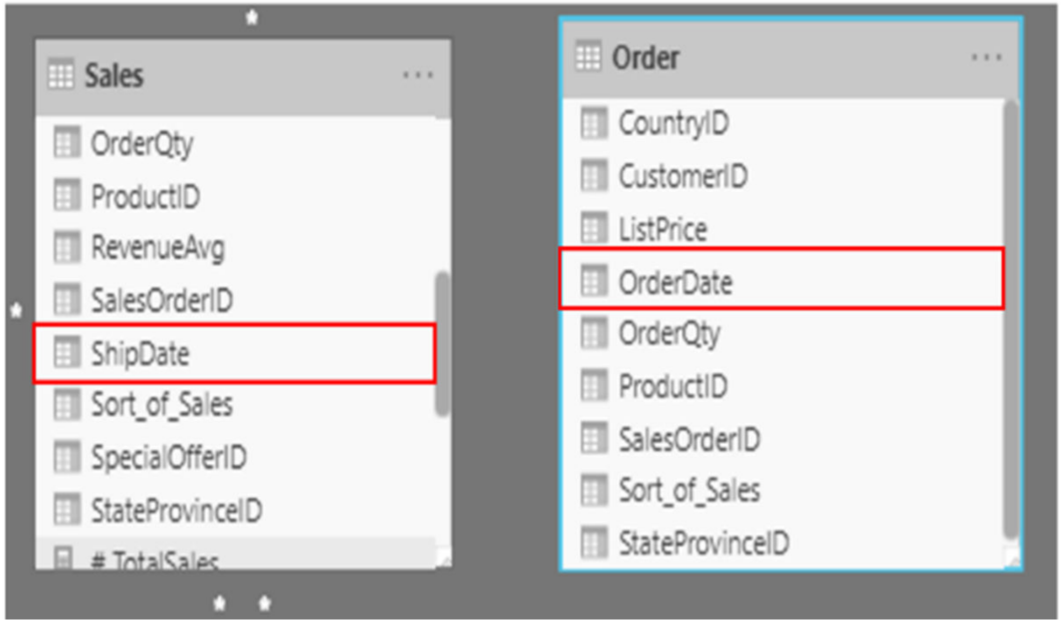
Additionally, Power BI has a new functionality to update these properties on many tables and fields by Ctrl+clicking or Shift+clicking items.

- These examples are only some of the many types of transformations that you can make to simplify the table structure. This step is important to take before you begin making your visuals so that you don't have to go back and forth when making formatting changes. This process of formatting and configuring tables can also be done in Power Query.

Create a Data Table

TEAM ACADEMY

- During report creation in Power BI, a common business requirement is to make calculations based on date and time.
- Organizations want to know how their business is doing over months, quarters, fiscal years, and so on.
- For this reason, it is crucial that these time-oriented values are formatted correctly.
- Power BI autodetects for date columns and tables; however, situations can occur where you will need to take extra steps to get the dates in the format that your organization requires.



Create a Common Data Table

TEAM ACADEMY

- Ways that you can build a common date table are:
- Source data
- DAX
- Power Query

Source Data

TEAM ACADEMY

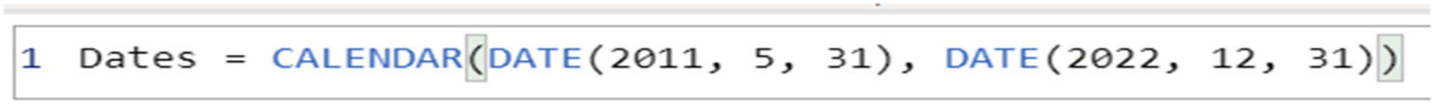
- Occasionally, source databases and data warehouses already have their own date tables. If the administrator who designed the database did a thorough job, these tables can be used to perform the following tasks:
- Identify company holidays
- Separate calendar and fiscal year

DAX

TEAM ACADEMY

- You can use the Data Analysis Expression (DAX) functions CALENDARAUTO() or CALENDAR() to build your common date table. The CALENDAR() function returns a contiguous range of dates based on a start and end date that are entered as arguments in the function.
- Alternatively, the CALENDARAUTO() function returns a contiguous, complete range of dates that are automatically determined from your dataset.
- The starting date is chosen as the earliest date that exists in your dataset, and the ending date is the latest date that exists in your dataset plus data that has been populated to the fiscal month that you can choose to include as an argument in the CALENDARAUTO() function.
- In Power BI Desktop, go to the **Table** tab on the ribbon. Select **New Table**, and then enter in the following DAX

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



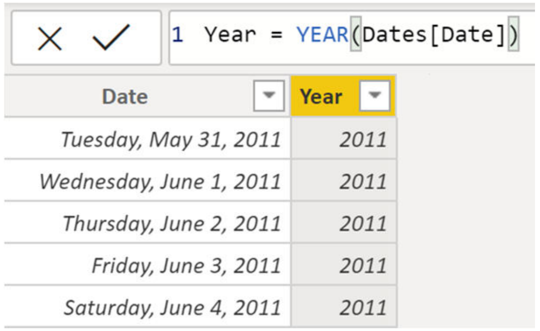
```
Dates = CALENDAR(DATE(2011, 5, 31), DATE(2022, 12, 31))
```

Continue …

TEAM ACADEMY

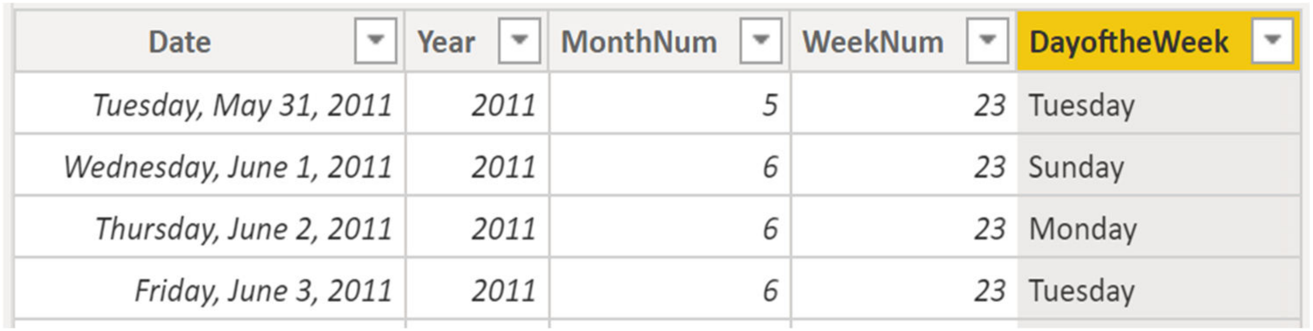
- Year = YEAR(Dates[Date])

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



```
1 Year = YEAR(Dates[Date])
```

Date	Year
Tuesday, May 31, 2011	2011
Wednesday, June 1, 2011	2011
Thursday, June 2, 2011	2011
Friday, June 3, 2011	2011
Saturday, June 4, 2011	2011

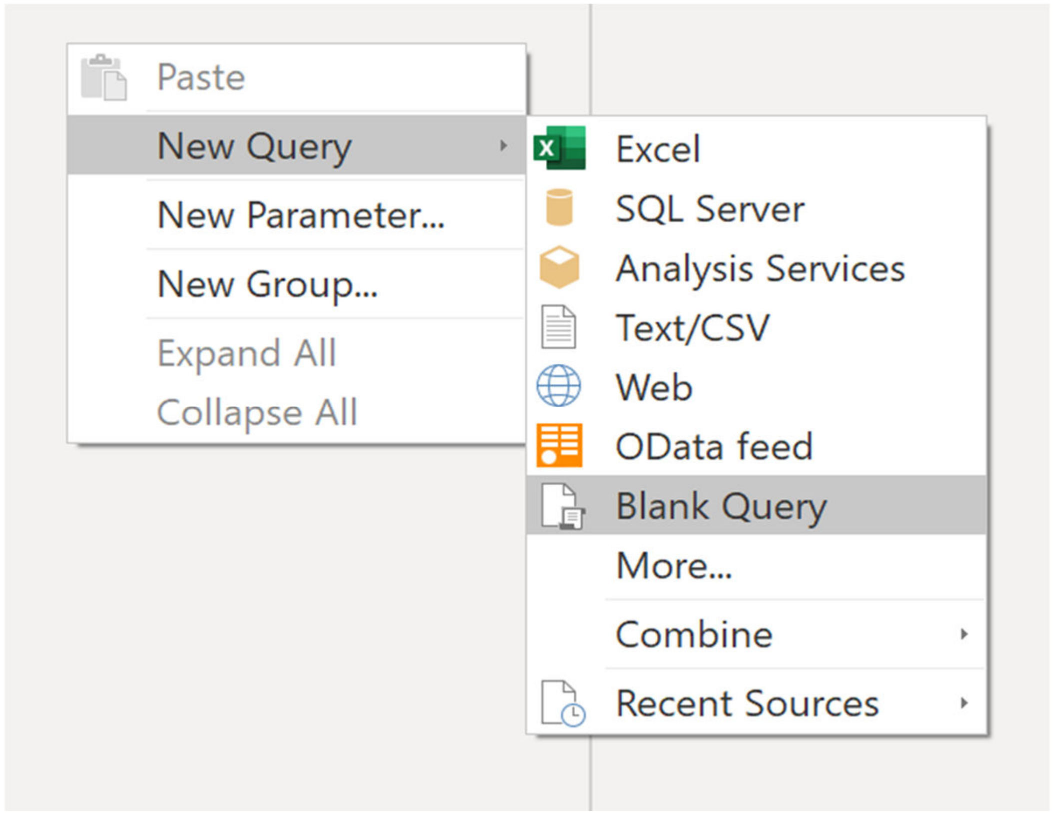


Date	Year	MonthNum	WeekNum	DayoftheWeek
Tuesday, May 31, 2011	2011	5	23	Tuesday
Wednesday, June 1, 2011	2011	6	23	Sunday
Thursday, June 2, 2011	2011	6	23	Monday
Friday, June 3, 2011	2011	6	23	Tuesday

Power Query

TEAM ACADEMY

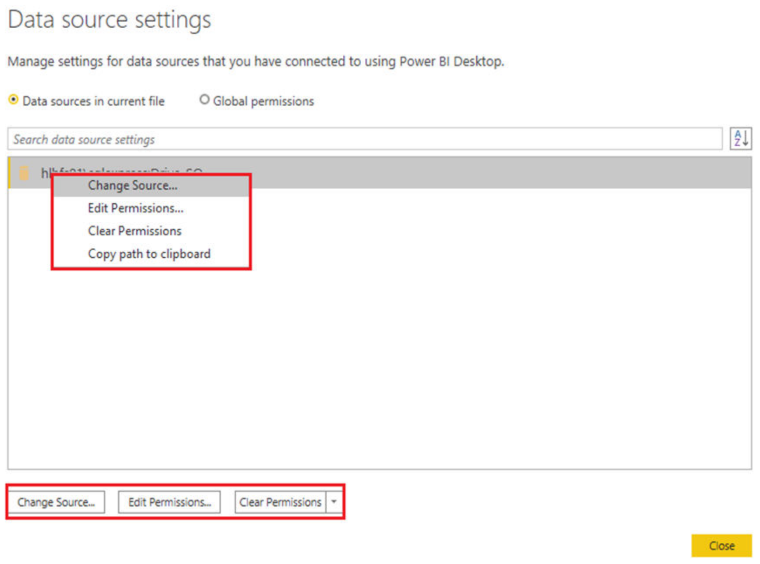
- You can use M-language, the development language that is used to build queries in Power Query, to define a common date table.
- Select **Transform Data** in Power BI Desktop, which will direct you to Power Query. In the blank space of the left **Queries** pane, right-click to open the following drop-down menu, where you will select **New Query > Blank Query**.



Continue …

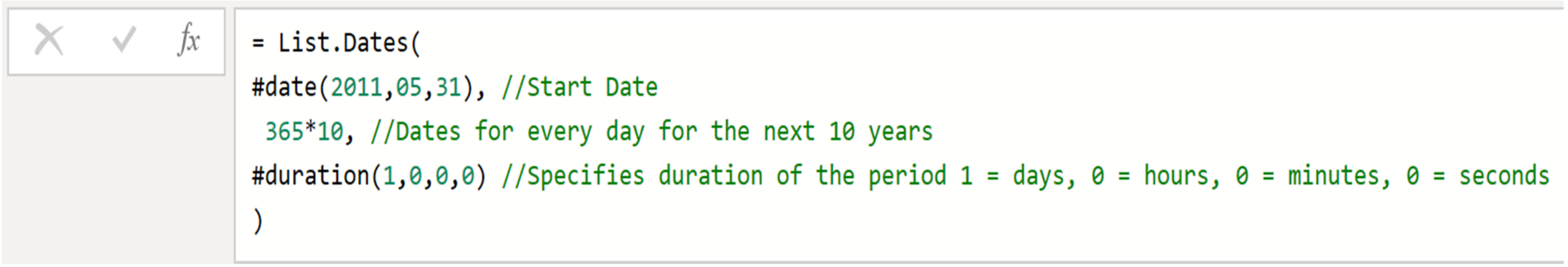
TEAM ACADEMY

- From the list of data sources that displays, select the data source that you want to update.
- Then, you can right-click that data source to view the available update options or you can use the update option buttons on the lower left of the window.
- Select the update option that you need, change the settings as required, and then apply your changes.
- = List.Dates(#date(2011,05,31), 365*10, #duration(1,0,0,0))


Data source settings

Manage settings for data sources that you have connected to using Power BI Desktop.

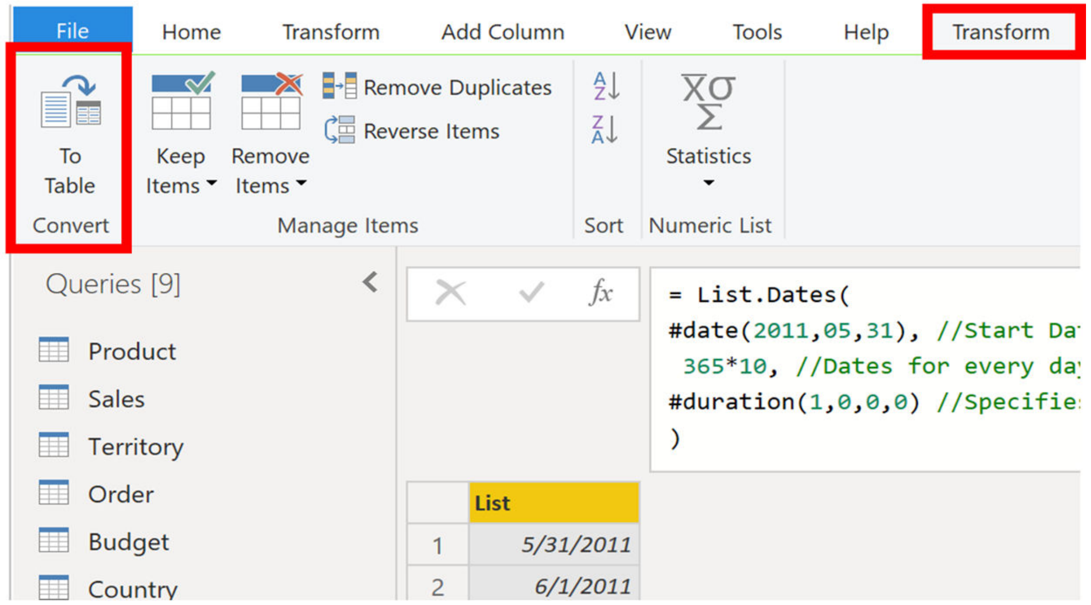
[✓ Data sources in current file] Global permissions


```
= List.Dates(
#date(2011,05,31), //Start Date
 365*10, //Dates for every day for the next 10 years
#duration(1,0,0,0) //Specifies duration of the period 1 = days, 0 = hours, 0 = minutes, 0 = seconds
)
```

Continue …

TEAM ACADEMY

- After you have realized success in the process, you notice that you have a list of dates instead of a table of dates.
- To correct this error, go to the **Transform** tab on the ribbon and select **Convert > To Table**. As the name suggests, this feature will convert your list into a table.
- You can also rename the column to **DateCol**.



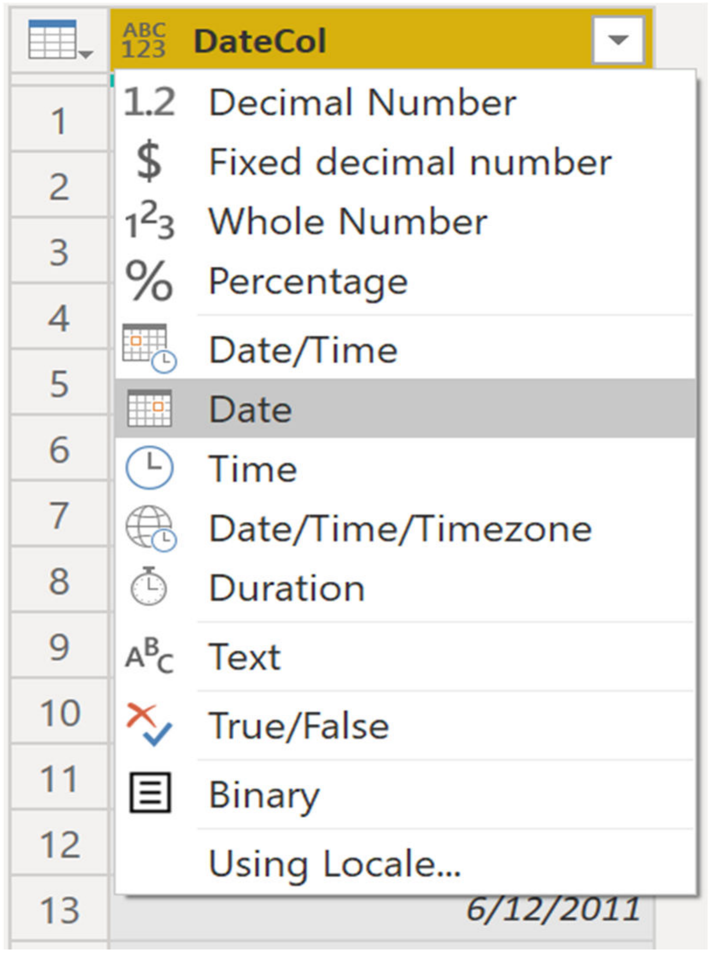
```
= List.Dates(
#date(2011,05,31), //Start Da
 365*10, //Dates for every da
#duration(1,0,0,0) //Specifie
)
```

	List
1	5/31/2011
2	6/1/2011

Continue ...

TEAM ACADEMY

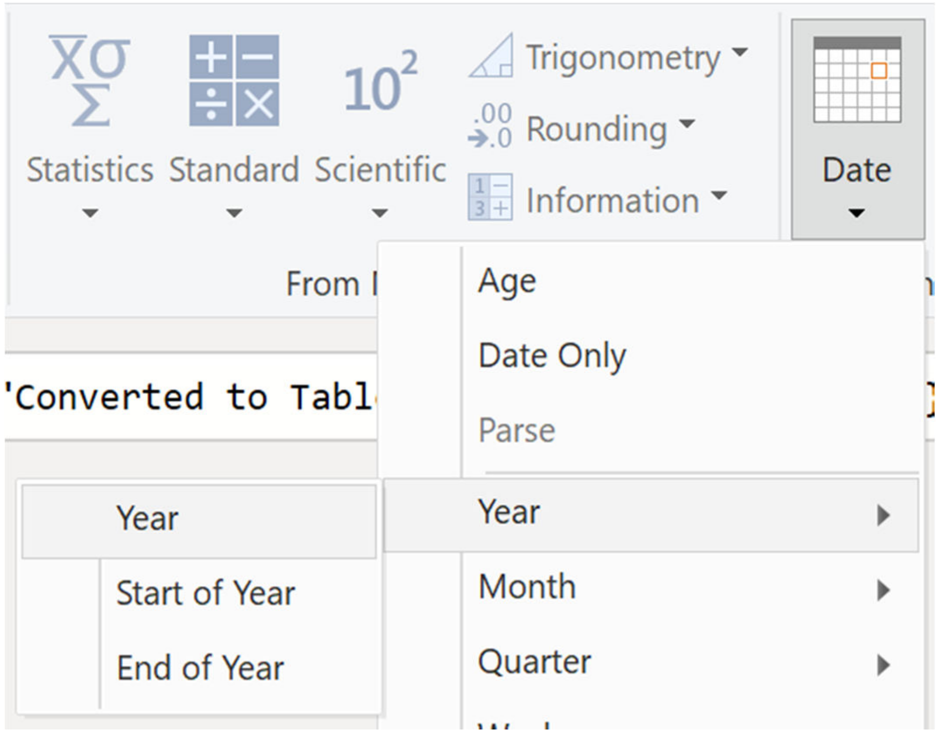
- Next, you want to add columns to your new table to see dates in terms of year, month, week, and day so that you can build a hierarchy in your visual.
- Your first task is to change the column type by selecting the icon next to the name of the column and, in the resulting drop-down menu, selecting the **Date** type.



Continue …

TEAM ACADEMY

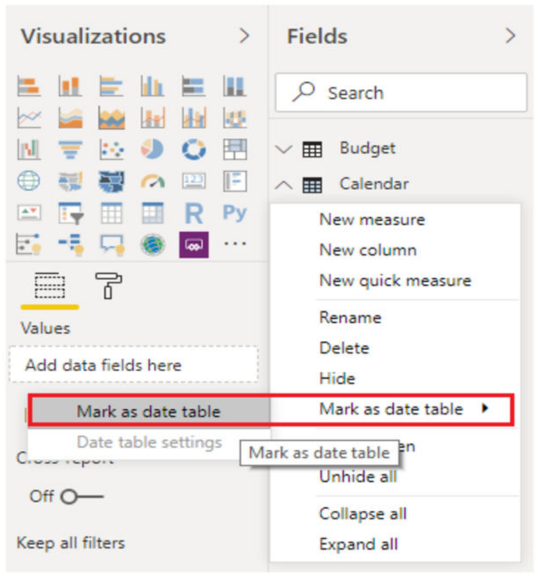
- After you have finished selecting the **Date** type, you can add columns for year, months, weeks, and days.
- Go to **Add Column**, select the drop-down menu under **Date**, and then select **Year**, as shown in the following figure.

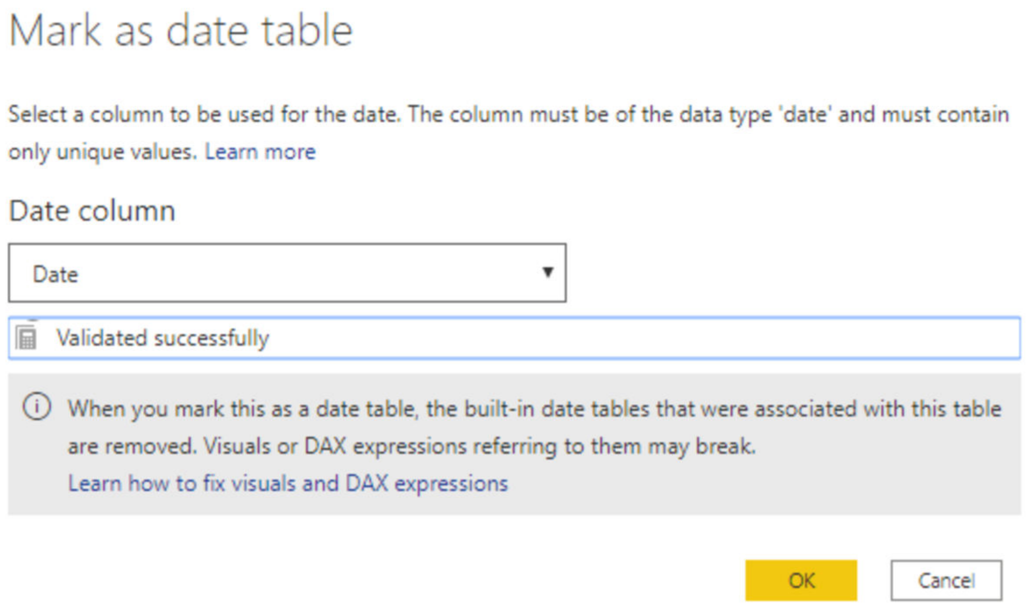


Mark as The Official Data Table

TEAM ACADEMY

- Your first task in marking your table as the official date table is to find the new table on the **Fields** pane. Right-click the name of the table and then select **Mark as date table**, as shown in the following figure.




Mark as date table

Select a column to be used for the date. The column must be of the data type 'date' and must contain only unique values. Learn more

Date column

Date

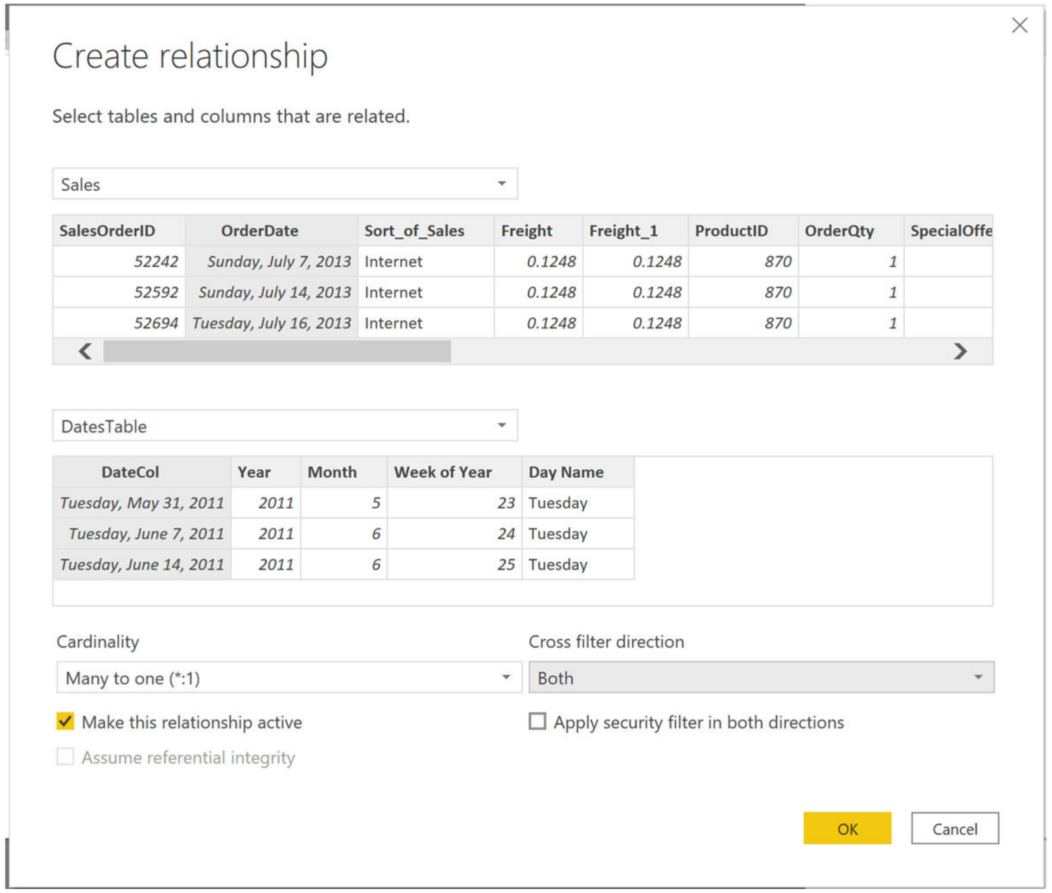
Validated successfully

When you mark this as a date table, the built-in date tables that were associated with this table are removed. Visuals or DAX expressions referring to them may break.
Learn how to fix visuals and DAX expressions

Build Your Visual

TEAM ACADEMY

- To build your visual between the Sales and Orders tables, you will need to establish a relationship between this new common date table and the Sales and Orders tables. As a result, you will be able to build visuals by using the new date table. To complete this task, go to **Model** tab **> Manage Relationships**, where you can create relationships between the common date table and the Orders and Sales tables by using the **OrderDate** column. The following screenshot shows an example of one such relationship.



Create relationship

Select tables and columns that are related.

Sales

SalesOrderID	OrderDate	Sort_of_Sales	Freight	Freight_1	ProductID	OrderQty	SpecialOffe
52242	Sunday, July 7, 2013	Internet	0.1248	0.1248	870	1	
52592	Sunday, July 14, 2013	Internet	0.1248	0.1248	870	1	
52694	Tuesday, July 16, 2013	Internet	0.1248	0.1248	870	1	

DatesTable

DateCol	Year	Month	Week of Year	Day Name
Tuesday, May 31, 2011	2011	5	23	Tuesday
Tuesday, June 7, 2011	2011	6	24	Tuesday
Tuesday, June 14, 2011	2011	6	25	Tuesday

Cardinality: Many to one (*:1)

Cross filter direction: Both

[✓] Make this relationship active

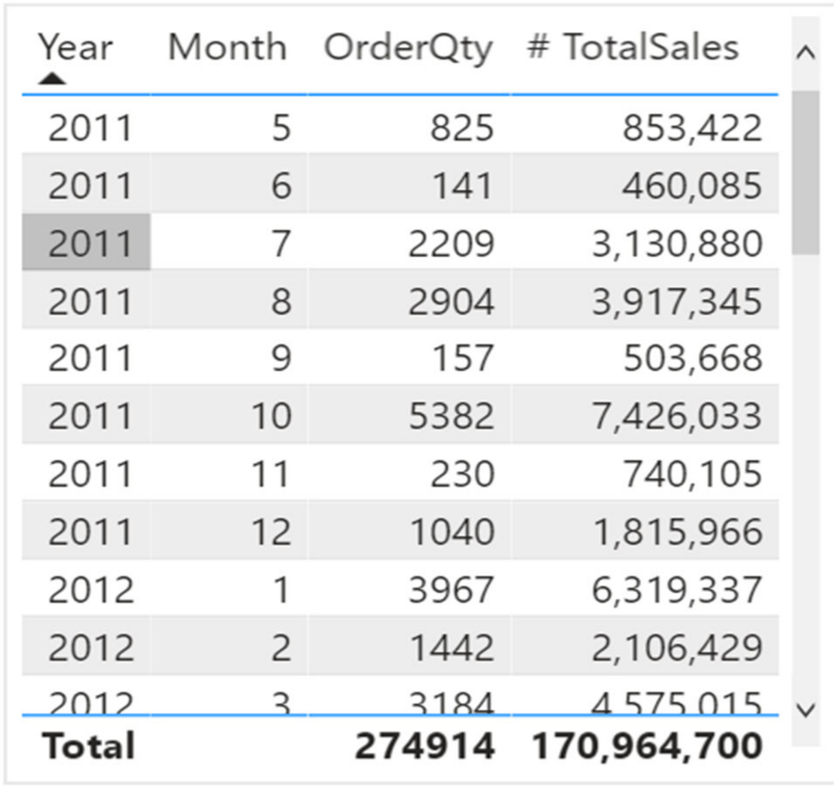
[ ] Apply security filter in both directions

[ ] Assume referential integrity

Continue ...

TEAM ACADEMY

- After you have finished, you can create a table by returning to the **Visualizations** tab and selecting the **Table** visual.
- You want to see the total orders and sales by year and month, so you only want to include the Year and Month columns from your date table, the **OrderQty** column, and the **#TotalSales** measure.
- When you learn about hierarchies, you can also build a hierarchy that will allow you drill down from years to months.
- For this example, you can view them side-by-side. You have now successfully created a visual with a common date table.

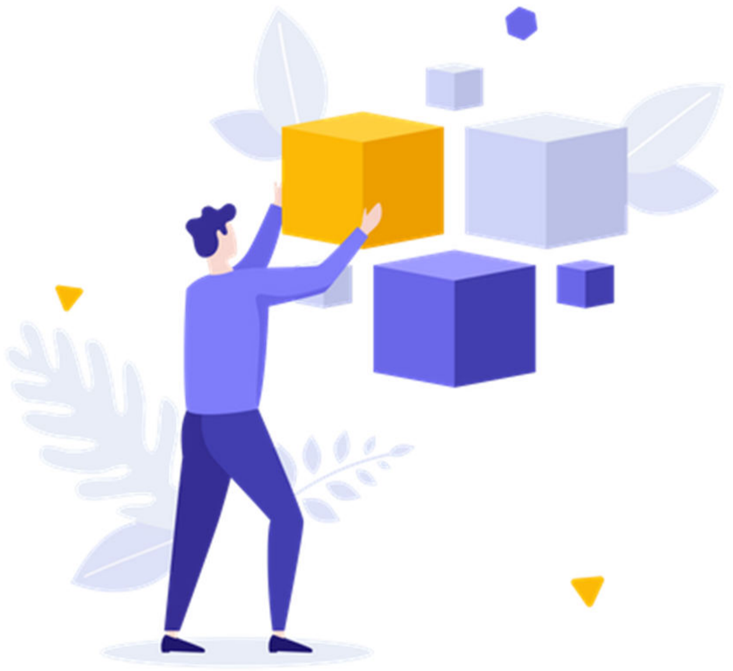


Year	Month	OrderQty	# TotalSales
2011	5	825	853,422
2011	6	141	460,085
2011	7	2209	3,130,880
2011	8	2904	3,917,345
2011	9	157	503,668
2011	10	5382	7,426,033
2011	11	230	740,105
2011	12	1040	1,815,966
2012	1	3967	6,319,337
2012	2	1442	2,106,429
2012	3	3184	4,575,015
Total		**274914**	**170,964,700**

Work with Dimensions

TEAM ACADEMY

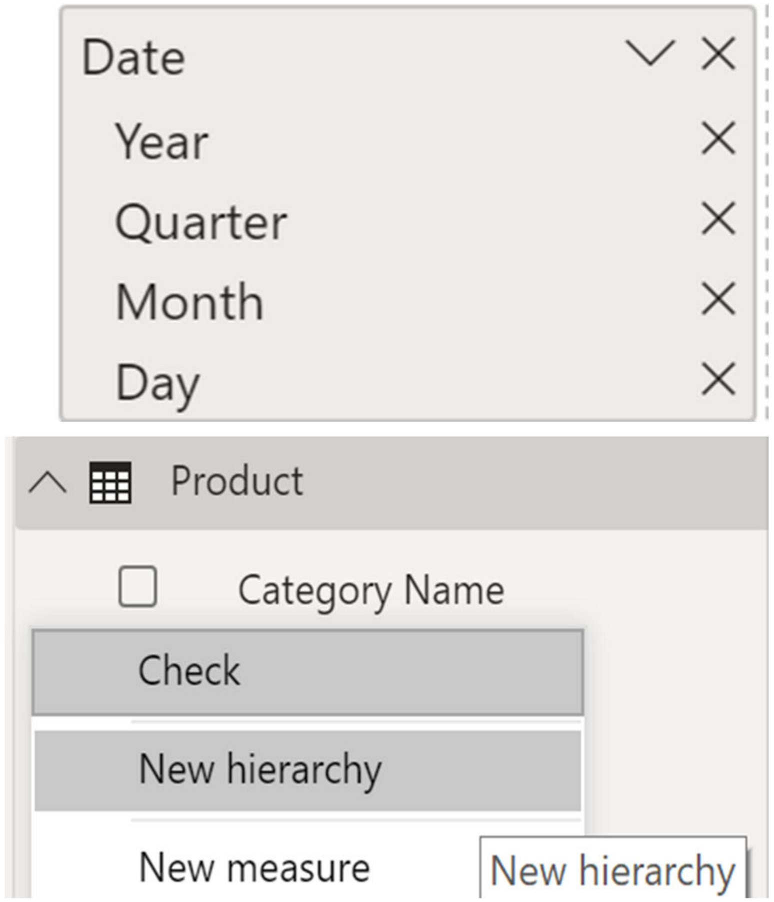
- You can use hierarchies as one source to help you find detail in dimension tables.
- These hierarchies form through natural segments in your data. For instance, you can have a hierarchy of dates in which your dates can be segmented into years, months, weeks, and days.
- Hierarchies are useful because they allow you to drill down into the specifics of your data instead of only seeing the data at a high level.



Hierarchies

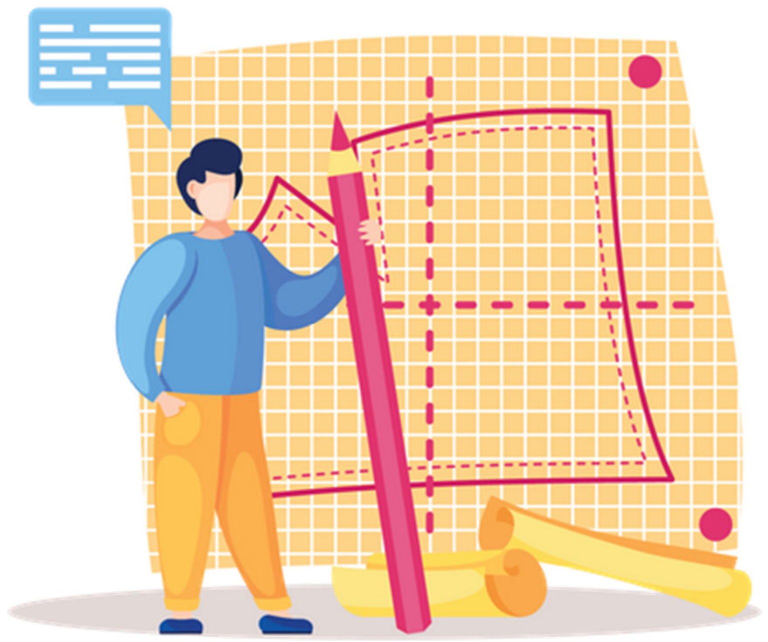
TEAM ACADEMY

- When you are building visuals, Power BI automatically enters values of the date type as a hierarchy (if the table has not been marked as a date table).
- In the preceding **Date** column, the date is shown in increasingly finer detail through year, quarters, months, and days. You can also manually create hierarchies.
- For example, consider a situation where you want to create a stacked bar chart of **Total Sales by Category and Subcategory**. You can accomplish this task by creating a hierarchy in the **Product** table for categories and subcategories. To create a hierarchy, go to the **Fields** pane on Power BI and then right-click the column that you want the hierarchy for. Select **New hierarchy**, as shown in the following figure.



Role – Playing Dimensions

- Role-playing dimensions have multiple valid relationships with fact tables, meaning that the same dimension can be used to filter multiple columns or tables of data.
- As a result, you can filter data differently depending on what information you need to retrieve.
- This topic is complex, so it is only introduced in this section. Working with role-playing dimensions requires complex DAX functions that will be discussed in later sections.



Define Data Granularity

TEAM ACADEMY

- Data granularity is the detail that is represented within your data, meaning that the more granularity your data has, the greater the level of detail within your data.
- Data granularity is an important topic for all data analysts, regardless of the Power BI tools that you are using. Defining the correct data granularity can have a big impact on the performance and usability of your Power BI reports and visuals.

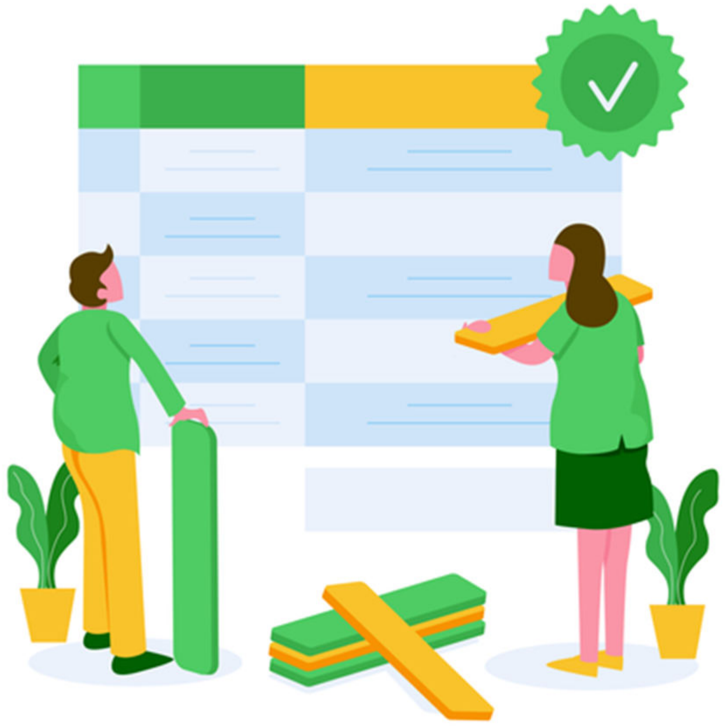
Data Granularity Defined

- Every few minutes, each truck uses a Microsoft Azure IoT application to record its current temperature.
- This temperature is important to your organization because, if the refrigeration were to malfunction, it could spoil the entire load, costing thousands of dollars.
- With so many trucks and so many sensors, extensive data is generated every day. Your report users don't want to sift through numerous records to find the ones that they are particularly interested in.

Change Data Granularity to Build a Relationship Between Two Tables

TEAM ACADEMY

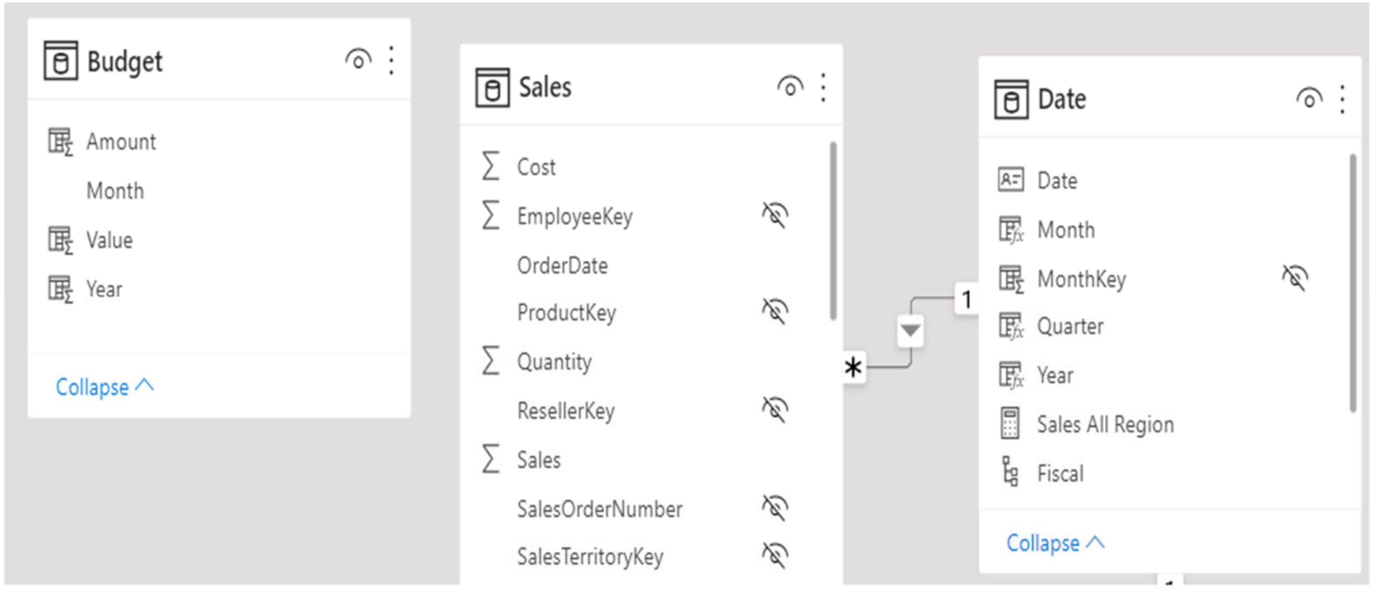
- Data granularity can also have an impact when you are building relationships between tables in Power BI.
- For example, consider that you are building reports for the Sales team at Tailwind Traders.
- You have been asked to build a matrix of total sales and budget over time by using the Calendar, Sales, and Budget tables. You notice that the 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.



Continue ...

TEAM ACADEMY

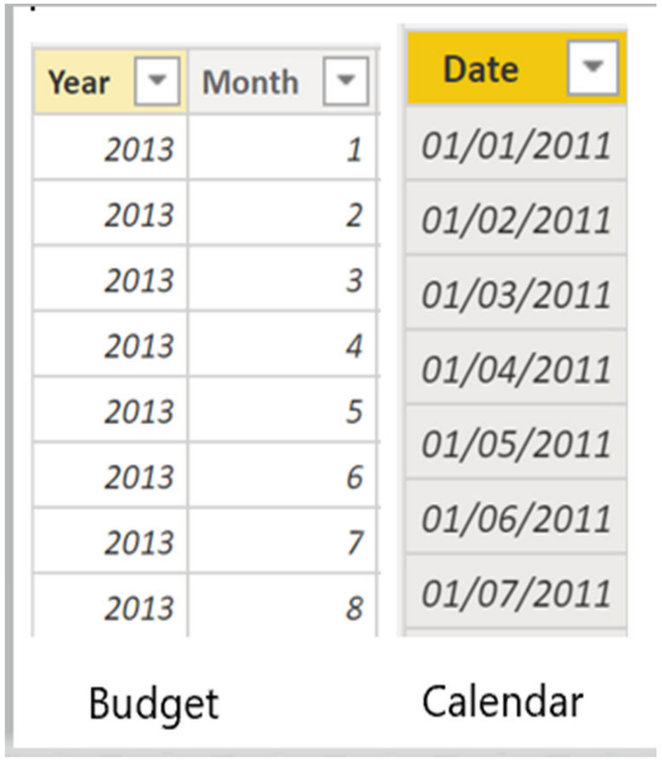
- The Budget table only goes to the monthly level, for instance, the budget data is 5/2020 and 6/2020.These tables have different granularities that need to be reconciled before you can build a relationship between tables.



Continue …

TEAM ACADEMY

- As shown in the preceding figure, a relationship between Budget and Calendar is missing.
- Therefore, you need to create this relationship before you can build your visual.
- Notice that if you transform the **Year** and **Month** columns in the Calendar table into a new column, and do the same transformation in the Budget table, you can match the format of the **Date** column in the Calendar table.
- Then, you can establish a relationship between the two columns. To complete this task, you will concatenate the **Year** and **Month** columns and then change the format.



Budget

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

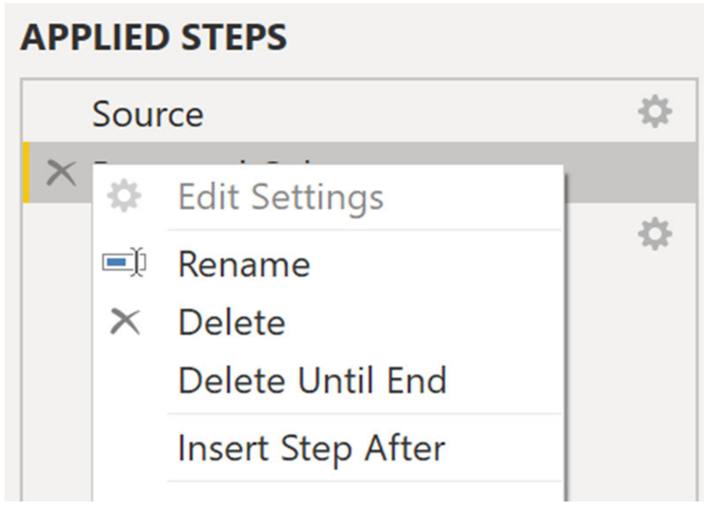
Calendar

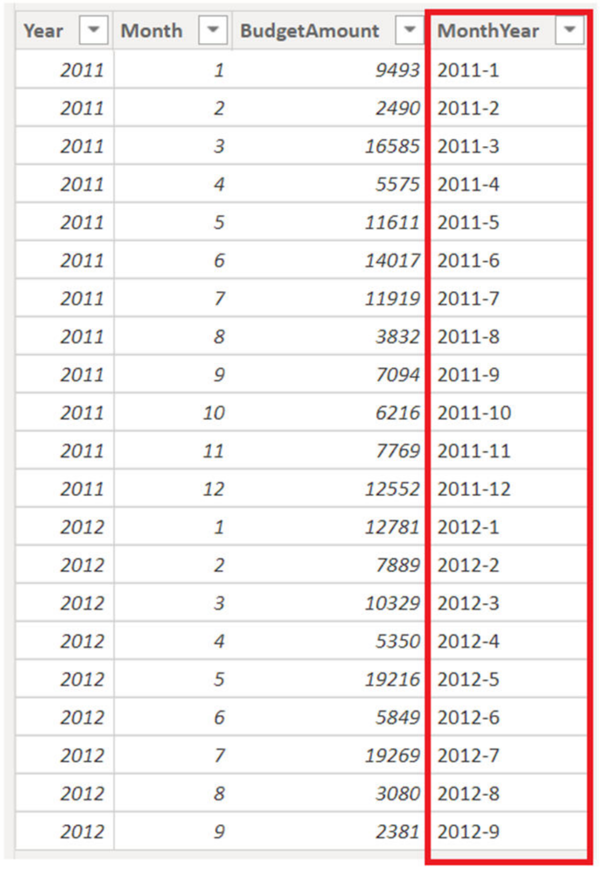
Date
01/01/2011
01/02/2011
01/03/2011
01/04/2011
01/05/2011
01/06/2011
01/07/2011

Continue …

TEAM ACADEMY

- Select **Transform Data** on the ribbon. On **Applied Steps**, on the right pane, right-click the last step and then select **Insert Step After**.
- Change the data type to **Date** and then rename the column. Your Budget table should resemble the following figure.





Year	Month	BudgetAmount	MonthYear
2011	1	9493	2011-1
2011	2	2490	2011-2
2011	3	16585	2011-3
2011	4	5575	2011-4
2011	5	11611	2011-5
2011	6	14017	2011-6
2011	7	11919	2011-7
2011	8	3832	2011-8
2011	9	7094	2011-9
2011	10	6216	2011-10
2011	11	7769	2011-11
2011	12	12552	2011-12
2012	1	12781	2012-1
2012	2	7889	2012-2
2012	3	10329	2012-3
2012	4	5350	2012-4
2012	5	19216	2012-5
2012	6	5849	2012-6
2012	7	19269	2012-7
2012	8	3080	2012-8
2012	9	2381	2012-9

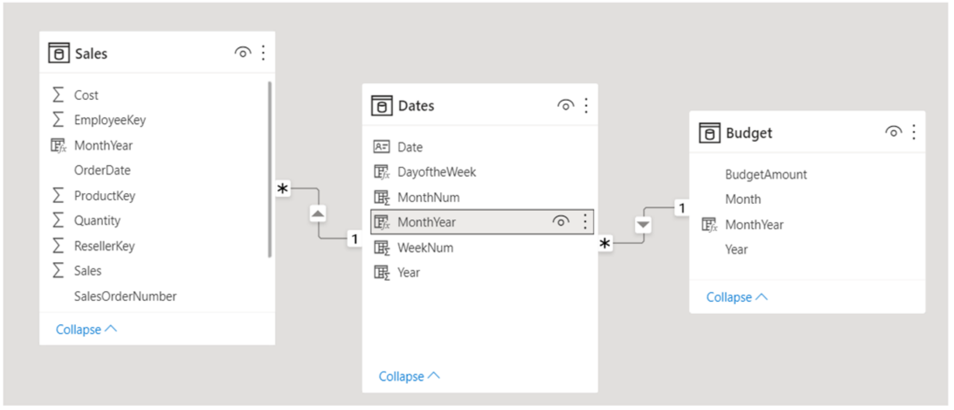
Create a Relationship Between Tables

TEAM ACADEMY

- Power BI automatically detects relationships, but you can also go to **Manage Relationships > New** and create the relationship on the **Date** column.
- The relationship should resemble the following figure.
- By completing this task, you have ensured that the granularity is the same between your different tables.

Now, you need to create DAX measures to calculate **Total Sales** and **Budget Amount**. Go to the **Data** pane on Power BI Desktop, select **New Measure**, and then create two measures with the following equations:

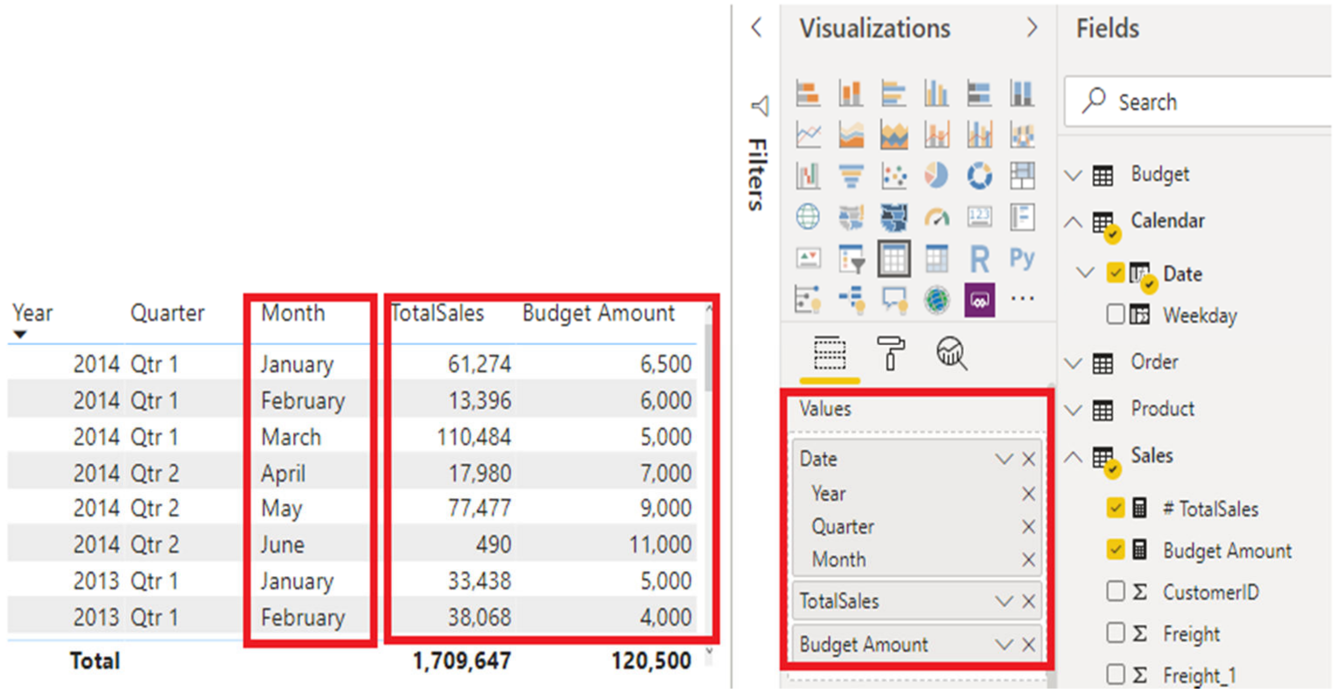
Total Sales = SUM(Sales[Total Sales])
Budget Amount = SUM (Budget [Budget Amount])



Continue ...

TEAM ACADEMY

- Select the table visual on the **Visualization** pane, and then enter these measures and the **Date** into the **Values** field. You have now accomplished the goal of building a matrix of the total sales and budgets over time.



Year	Quarter	Month	TotalSales	Budget Amount
2014	Qtr 1	January	61,274	6,500
2014	Qtr 1	February	13,396	6,000
2014	Qtr 1	March	110,484	5,000
2014	Qtr 2	April	17,980	7,000
2014	Qtr 2	May	77,477	9,000
2014	Qtr 2	June	490	11,000
2013	Qtr 1	January	33,438	5,000
2013	Qtr 1	February	38,068	4,000
Total			**1,709,647**	**120,500**

Work with Relationships and Cardinality

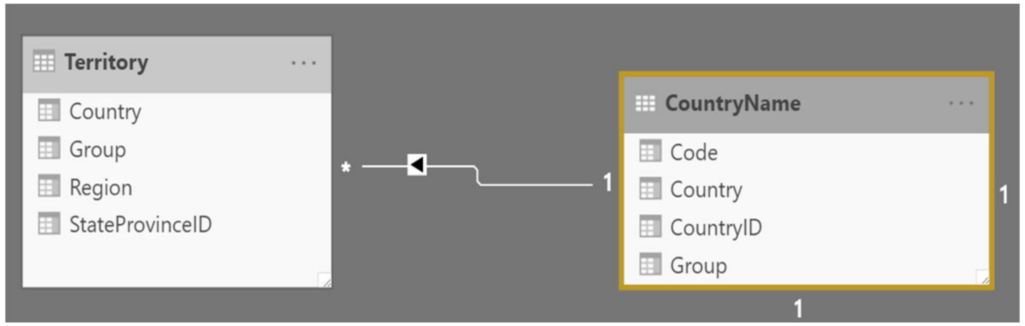
TEAM ACADEMY

- Unlike other database management systems, Power BI has the concept of *directionality* to a relationship.
- This directionality plays an important role in filtering data between multiple tables. When you load data, Power BI automatically looks for relationships that exist within the data by matching column names.
- You can also use **Manage Relationships** to edit these options manually.

Relationships

TEAM ACADEMY

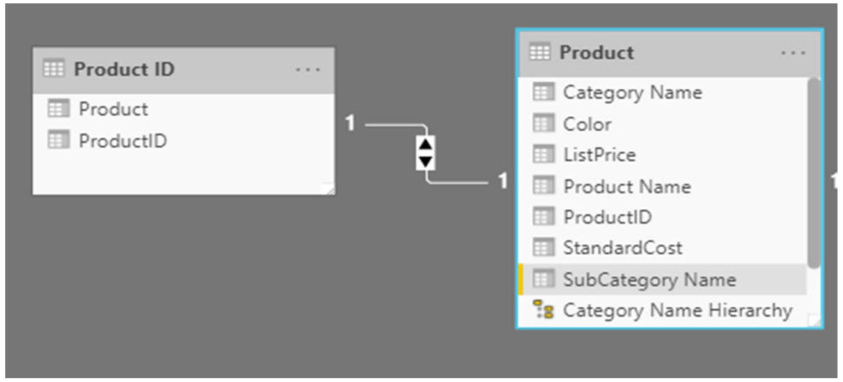
- The following are different types of relationships that you'll find in Power BI.
- Many-to-one (*:1) or one-to-many (1: *) relationship
- Describes a relationship in which you have many instances of a value in one column that are related to only one unique corresponding instance in another column.
- Describes the directionality between fact and dimension tables.
- Is the most common type of directionality and is the Power BI default when you are automatically creating relationships.
- An example of a one-to-many relationship would be between the CountryName and Territory tables, where you can have many territories that are associated with one unique country.



Continue …

TEAM ACADEMY

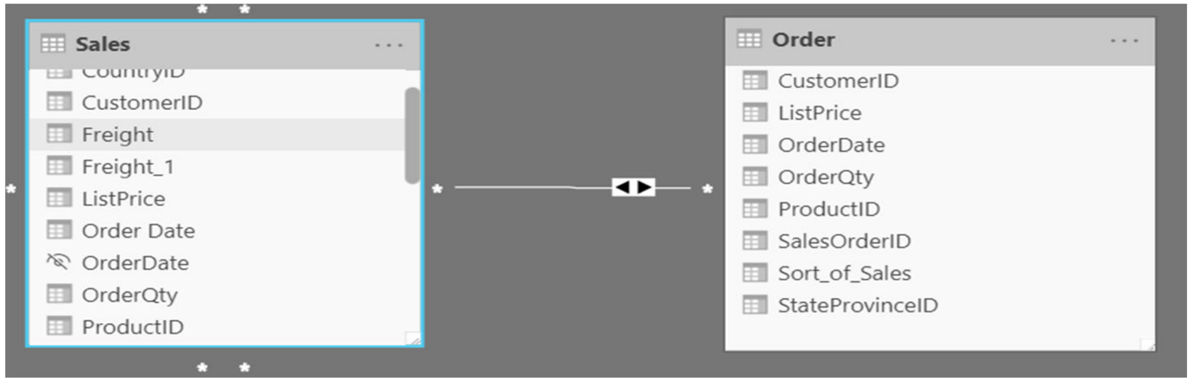
- One-to-one (1:1) relationship:
- Describes a relationship in which only one instance of a value is common between two tables.
- Requires unique values in both tables.
- Is not recommended because this relationship stores redundant information and suggests that the model is not designed correctly. It is better practice to combine the tables.
- An example of a one-to-one relationship would be if you had products and product IDs in two different tables.
- Creating a one-to-one relationship is redundant and these two tables should be combined.



Continue …

TEAM ACADEMY

- Many-to-many (.) relationship:
- Describes a relationship where many values are in common between two tables.
- Does not require unique values in either table in a relationship.
- Is not recommended; a lack of unique values introduces ambiguity and your users might not know which column of values is referring to what.
- For instance, the following figure shows a many-to-many relationship between the Sales and Order tables on the **Order Date** column because multiple sales can have multiple orders associated with them.
- Ambiguity is introduced because both tables can have the same order date.



Cross Filter Direction

- Data can be filtered on one or both sides of a relationship.

With a **single cross-filter direction**:

- Only one table in a relationship can be used to filter the data. For instance, Table 1 can be filtered by Table 2, but Table 2 cannot be filtered by Table 1.
- For a one-to-many or many-to-one relationship, the cross-filter direction will be from the "one" side, meaning that the filtering will occur in the table that has many values.

With **both cross-filter directions** or **bi-directional cross-filtering**:

- One table in a relationship can be used to filter the other. For instance, a dimension table can be filtered through the fact table, and the fact tables can be filtered through the dimension table.
- You might have lower performance when using bi-directional cross-filtering with many-to-many relationships.

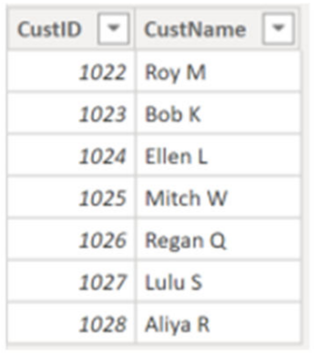
Cardinality and Cross Filter Direction

- For one-to-one relationships, the only option that is available is bi-directional cross-filtering.
- Data can be filtered on either side of this relationship and result in one distinct, unambiguous value.
- For instance, you can filter on one Product ID and be returned a single Product, and you can filter on a Product and be returned
- For many-to-many relationships, you can choose to filter in a single direction or in both directions by using bi-directional cross-filtering.
- The ambiguity that is associated with bi-directional cross-filtering is amplified in a many-to-many relationship because multiple paths will exist between different tables.a single Product ID.

Create a Many-To-Many Relationships

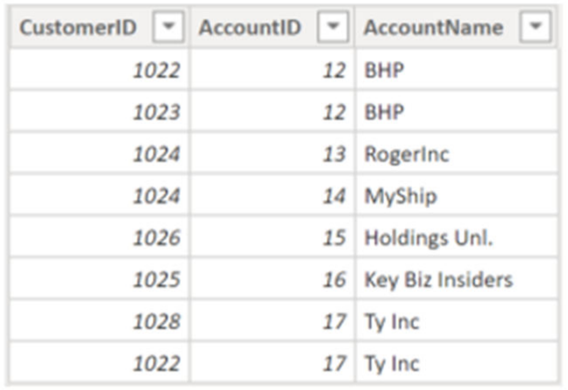
TEAM ACADEMY

- Consider the scenario where you are tasked with building a visual that examines budgets for customers and accounts.
- You can have multiple customers on the same account and multiple accounts with the same customer, so you know that you need to create a many-to-many relationship.



CustID	CustName
1022	Roy M
1023	Bob K
1024	Ellen L
1025	Mitch W
1026	Regan Q
1027	Lulu S
1028	Aliya R

CustomerTable



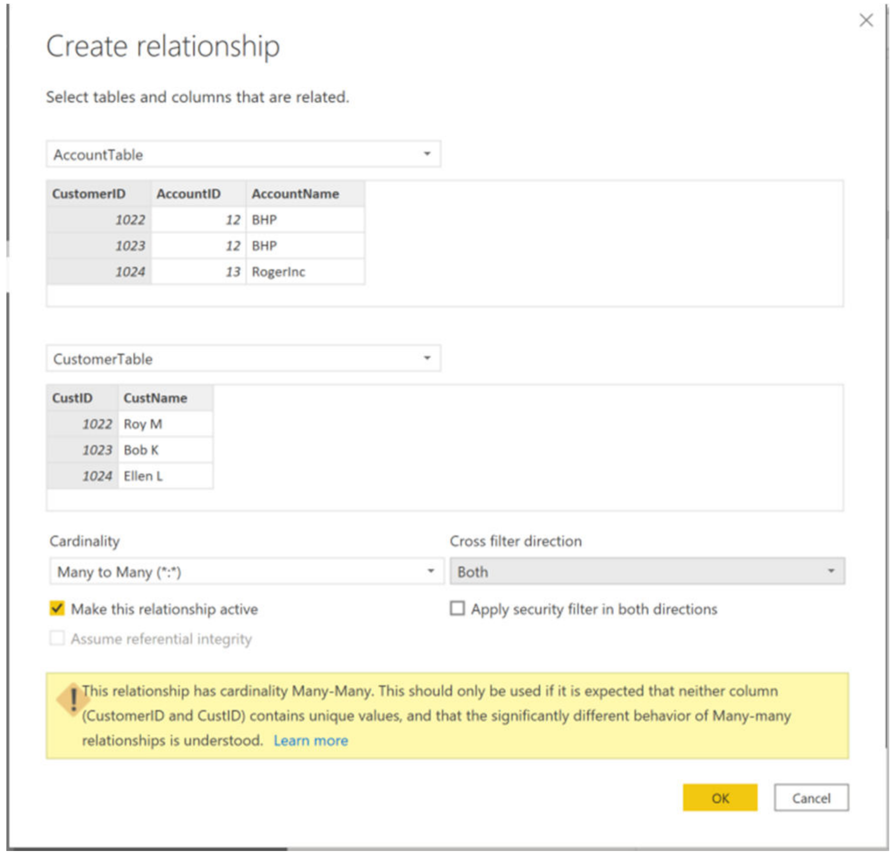
CustomerID	AccountID	AccountName
1022	12	BHP
1023	12	BHP
1024	13	RogerInc
1024	14	MyShip
1026	15	Holdings Unl.
1025	16	Key Biz Insiders
1028	17	Ty Inc
1022	17	Ty Inc

AccountTable

Continue …

TEAM ACADEMY

- To create this relationship, go to **Manage Relationships > New**.
- In the resulting window, create a relationship between the **Customer ID** column in CustomerTable and AccountTable. The relationship is set to many-to-many, and the filter type is in both directions.
- Immediately, you will be warned that you should only use this type of relationship if it is expected that neither column will have unique values because you might get unexpected values. Because you want to filter in both directions, choose **bi-directional cross-filtering**.
- Select **OK**. You have now successfully created a many-to-many relationship.



Create relationship

Select tables and columns that are related.

AccountTable

CustomerID	AccountID	AccountName
1022	12	BHP
1023	12	BHP
1024	13	RogerInc

CustomerTable

CustID	CustName
1022	Roy M
1023	Bob K
1024	Ellen L

Cardinality: Many to Many (*:*)

Cross filter direction: Both

☑ Make this relationship active

☐ Apply security filter in both directions

☐ Assume referential integrity

This relationship has cardinality Many-Many. This should only be used if it is expected that neither column (CustomerID and CustID) contains unique values, and that the significantly different behavior of Many-many relationships is understood. Learn more

Resolve Modeling Challenges

TEAM ACADEMY

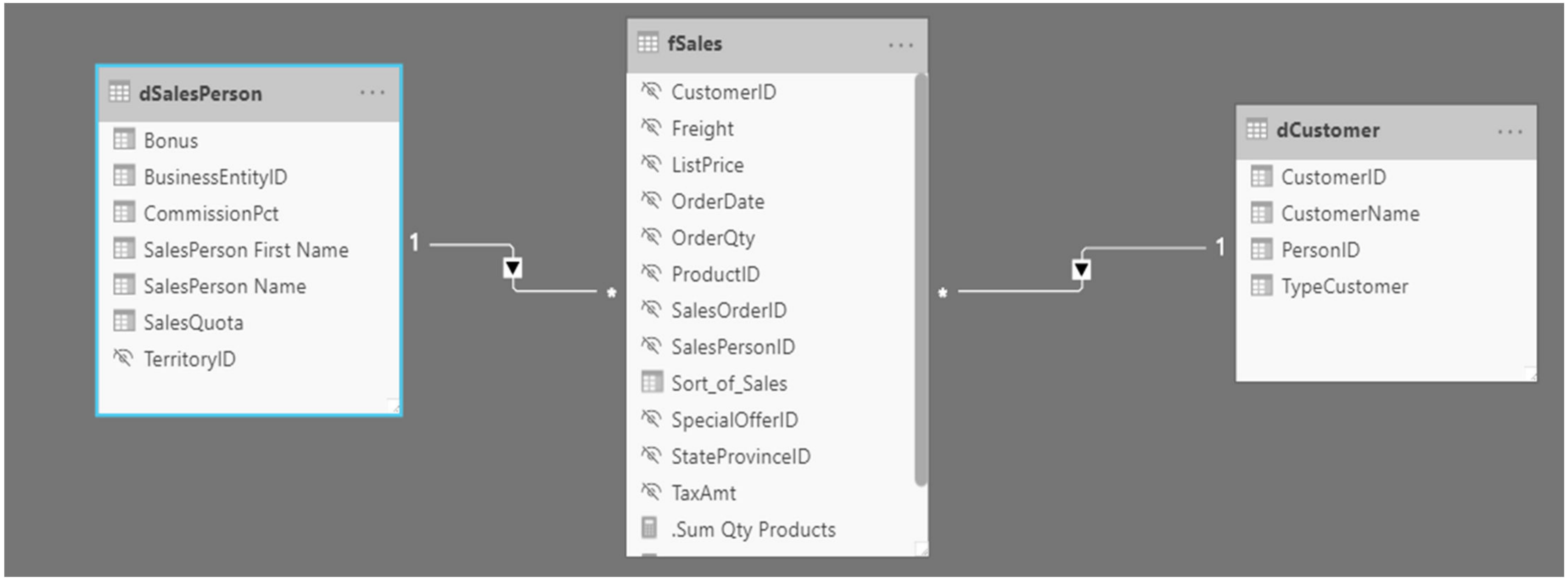
- Modeling data is about establishing and maintaining relationships so that you can effectively visualize the data in the form that your business requires.
- When you are creating these relationships, a common pitfall that you might encounter are circular relationships.

Relationships Dependencies

- To understand circular relationships, you first need to understand dependencies.
- For example, consider that you have the following calculated column **Total** in the Sales table.
- Sales['TotalCost'] = Sales['Quantity'] * Sales['Price']
- **TotalCost** depends on **Quantity** and **Price**, so if a change occurs in either quantity or price, a change will occur in **TotalCost** as well.
- This example outlines a dependency of a column on other columns, but you can also have dependencies between measures, tables, and relationships.
- Consider the following relationships between **SalesPerson**, **Sales**, and **Customer**.
- A change in **Customer** will result in a change in **Sales**, which results in changes in **SalesPerson.** These types of dependencies can exist within relationships.

Continue ...

TEAM ACADEMY



Visit

teamacademy.qa

Contact us

info@teamacademy.net

+974 6623 0468

Follow us on

Team Academy Global

Team Academy Global

Team Academy Global