Tasks

The report file **Lesson 4 - Building the Data Model.pbix** is stll used in this exercise.

In the previous task, you learned how to create a relationship between two tables that had a one-to-many relationship. However, the analytical value achieved through many-to-many relationships does not happen automatically and requires an extra step.

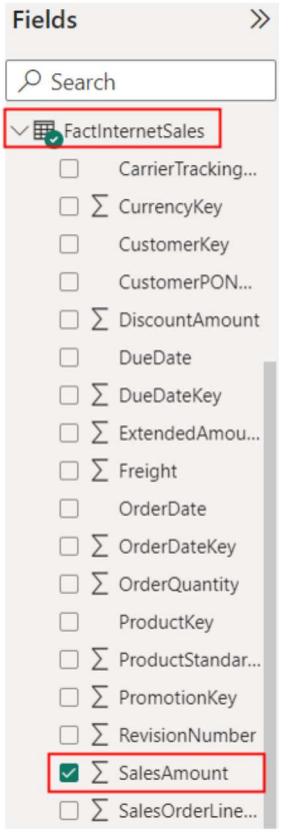
A many-to-many relationship is when multiple rows in one table are associated with multiple rows in another table. An example of a many-to-many relationship can be observed in the relationship between products and customers. A product can be sold to many customers; likewise, a customer can purchase many products.

Task 1 - Filtering behavior

Step 1: Create a report to show the total **SalesAmount** of all transactions:

- In *Report view* select **Table** in **Visualizations**.

- In **Fields** select **FactInternetSales > Sales Amount:**.



The visual you see is simply the sum of the column *SalesAmount* from the *FactInternetSales* table:

SalesAmount
29 358 677,22

Step 2: Create a report to show the total **SalesAmount for all transactions** broken down **by country**

1. In *Report view* select **Table** in **Visualizations**
2. In **Fields** select **DimSalesTerritory > SalesTerritory**
3. In **Fields** select **FactInternetSales > Sales Amount**

SalesTerritoryRegion	SalesAmount
Australia	9 061 000,58
Canada	1 977 844,86
Central	3 000,83
France	2 644 017,71
Germany	2 894 312,34
Northeast	6 532,47
Northwest	3 649 866,55
Southeast	12 238,85
Southwest	5 718 150,81
United Kingdom	3 391 712,21
Total	**29 358 677,22**

Note! This only works because a valid relationship exists between the *FactInternetSales* and *DimSalesTerritory* tables.

Step 3: Create a report to show the **Temperature range** and total **SalesAmount**:

1. In *Report view* create a new page by clicking **+**

2. In *Report view* select **Table** in **Visualizations**

3. In **Fields** select **5 Regions 2008 > Temperature Range**
4. In **Fields** select **FactInternetSales > Sales Amount**

Temperature Range	SalesAmount
Cold	29 358 677,22
Cool	29 358 677,22
Hot	29 358 677,22
Warm	29 358 677,22
Total	**29 358 677,22**

Notice how the total sales amount is repeated for each temperature range. This behavior indicates that the *5 Regions 2008* table is unable to filter the *FactInternetSales* table. This inability to filter can happen for a number of different reasons:

- Because a relationship does not exist between the tables
- Because an existing relationship is invalid
- Because an existing relationship does not allow the filtering to pass through an intermediate table.

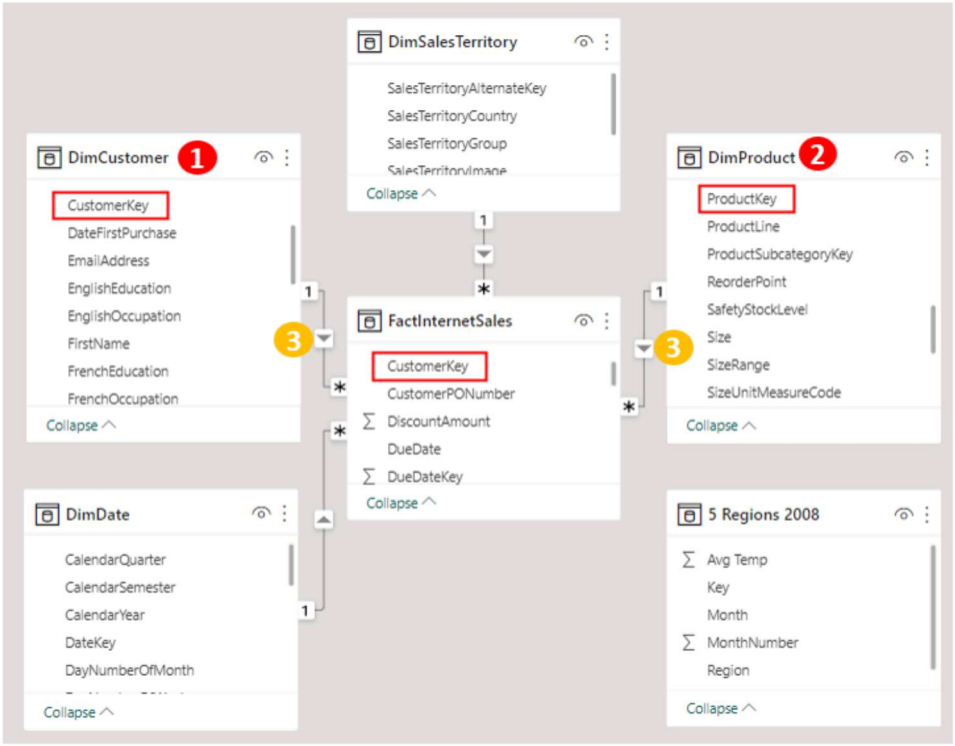
Note! If you see the repeated value behavior demonstrated above, then go back to the relationship view and verify that all relationships have been created and are valid.

Task 2 - Enable filtering from the many side of a relationship

In this data model, **DimProduct** and **DimCustomer** have a *many-to-many relationship*. A product can be sold to many customers. For example, bread can be sold to Jessica, Kim, and Tyrone. A customer can purchase many products. Kim could purchase bread, milk, and cheese.

A **bridge table** can be used to store the relationship between two tables that have a many-to-many relationship. The *FactInternetSales* table in the figure is a large, many-to-many bridge table:

- The relationship between *DimCustomer* and *FactInternetSales (1 > 2)*
- The relationship between *DimProduct* and *FactInternetSales (2 > 1)*
- The cross-filter direction is set to single (3).



Step 1: Create a report to display the *total sales*, *total transactions*, and *customer count* for each product:

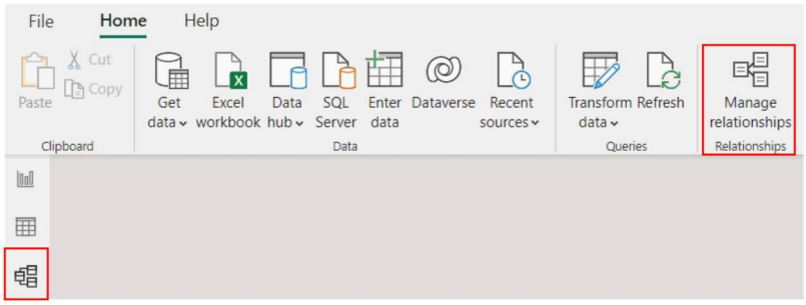
1. In *Report view* create a new page by clicking **+**

2. In *Report view* select **Table** in **Visualizations**
3. Select **EnglishProductName** from the *DimProduct* table
4. Select *T***otal Sales** (the SUM of the *SalesAmount* column) from the *FactInternetSales* table
5. Select **Total Transactions** (the COUNT of associated rows) from the *FactInternetSales* table
6. Select **Customer Count** (the COUNT of the *CustomerKey* column) from the *DimCustomer* table.

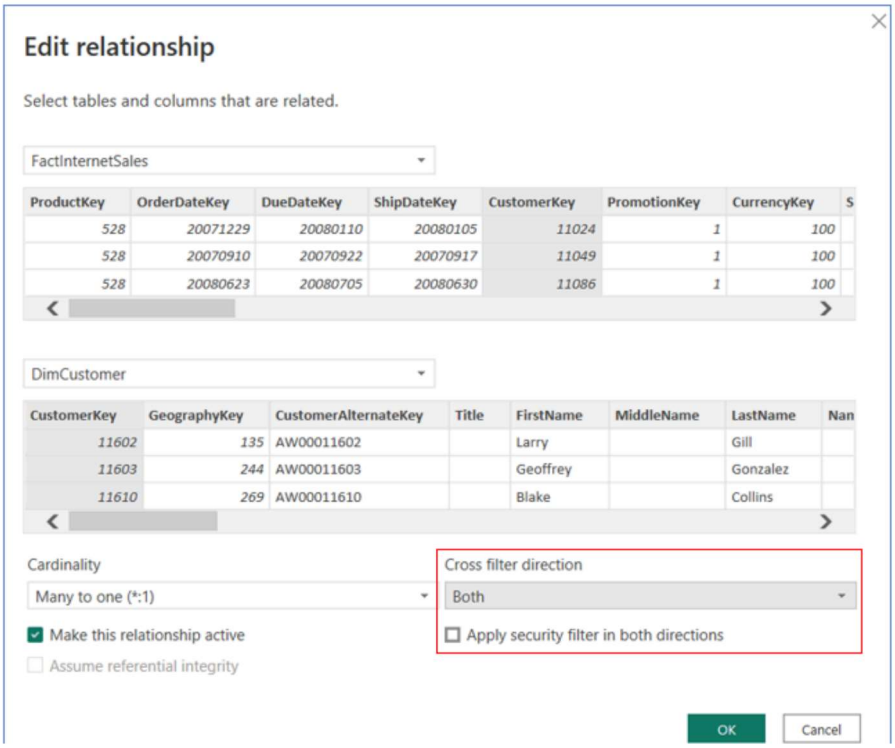
EnglishProductName	SalesAmount	OrderQuantity	CustomerKey
Adjustable Race			18484
All-Purpose Bike Stand	39 591,00	249	18484
AWC Logo Cap	19 688,10	2190	18484
BB Ball Bearing			18484
Bearing Ball			18484
Bike Wash - Dissolver	7 218,60	908	18484
Blade			18484
Cable Lock			18484
Chain			18484
Chain Stays			18484
Chainring			18484
Total	**29 358 677,22**	**60398**	**18484**

Total Sales and *Total Transactions* are returning the correct results for each product. But *Customer Count* is returning the same value for all products (18484). This is due to the way that filtering works. By simply enabling cross-filtering in both directions, the *FactInternetSales* table will be able to filter the customer table and the customer count will work.

Step 2: Open the **Manage Relationships** editor, and click **Edit...**



Step 3: In **Edit relationships** select the relationship between *FactInternetSales* and *DimCustomer*, and then click **Edit**. Change the **Cross filter direction** from *Single* to *Both.*



Step 4: Back in the Report view, you will now see the correct *Customer Count* for each product:

EnglishProductName	SalesAmount	OrderQuantity	CustomerKey
All-Purpose Bike Stand	39 591,00	249	243
AWC Logo Cap	19 688,10	2190	2132
Bike Wash - Dissolver	7 218,60	908	875
Classic Vest, L	12 382,50	195	195
Classic Vest, M	12 636,50	199	199
Classic Vest, S	10 668,00	168	168
Fender Set - Mountain	46 619,58	2121	2110
Half-Finger Gloves, L	10 849,07	443	437
Half-Finger Gloves, M	12 220,51	499	488
Half-Finger Gloves, S	11 951,12	488	479
Hitch Rack - 4-Bike	39 360,00	328	325
Total	**29 358 677,22**	**60398**	**18484**

As a best practice, it is not recommended to enable cross-filtering in yBack in the Report view, you will now see the correct *Customer Count* for each product:our data model. Cross-filtering can cause ambiguity in your results and can cause some time intelligence functions to not function properly; the date table must have a contiguous range of dates and therefore cannot be filtered by other tables. .

Task 3 - Role-playing tables

A **role-playing table** is a table that can play multiple roles, which helps to reduce data redundancy. Most often, a date table is a role-playing table. For example, the **FactInternetSales** table has three dates to track the processing of an order. There is the *Order Date*, *Ship Date*, and *Due Date* and, without role-playing tables, you would need to have three separate date tables instead of just one. The additional tables take up valuable resources, such as memory, as well as adding an extra layer of administrative upkeep.

Each of these dates is very important to different people and different departments within an organization. For example, the finance department may wish to see total sales and profit by the date that a product was purchased, the order date. However, your shipping department may wish to see product quantity based on the ship date. How do you accommodate requests from different departments in a single data model?

There are generally two ways you can handle role-playing tables in Power BI:

- Import the table multiple times and create an active relationship for each table

- Use DAX and inactive relationships to create calculations that show calculations by different dates

The first way, and the method we will show here, is importing the table multiple times. Yes, this means that it will take up more resources. The data model will have three date tables, one table to support each date in the *FactInternetSales* table. Each date table will have a single active relationship to the *FactInternetSales* table.

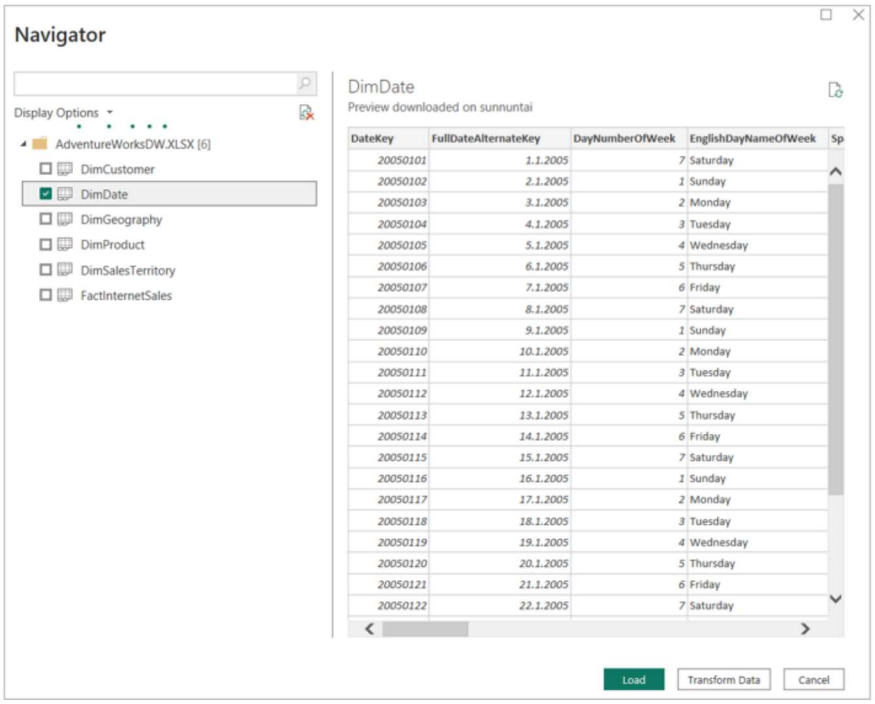
Some of the benefits of importing the table multiple times are as follows:

- It is easier to train and acclimate end users with the data model. For example, if you want to see sales and profit by the ship date, then you would simply use the date attributes from the ship date table in your reports.
- Most of your DAX measures will work across all date tables, so there is no need to create new measures. The exception here is your time intelligence calculations; they will need to be rebuilt for each date table.
- The analytical value of putting different dates in a matrix. For example, sales ordered and sales shipped by date. For clarification.

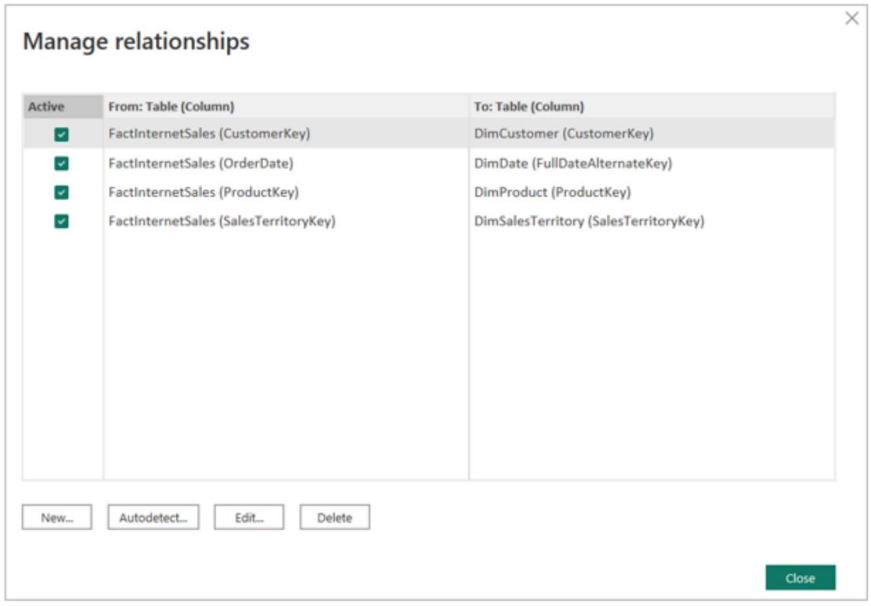
Create a report to show *sales ordered* and *sales shipped* by date

Step 1: From the **Get data** option, select **Excel**, and open the **AdventureWorksDW Excel** file.

Step 2: Select **DimDate** from the list of tables, and then click **Load**.

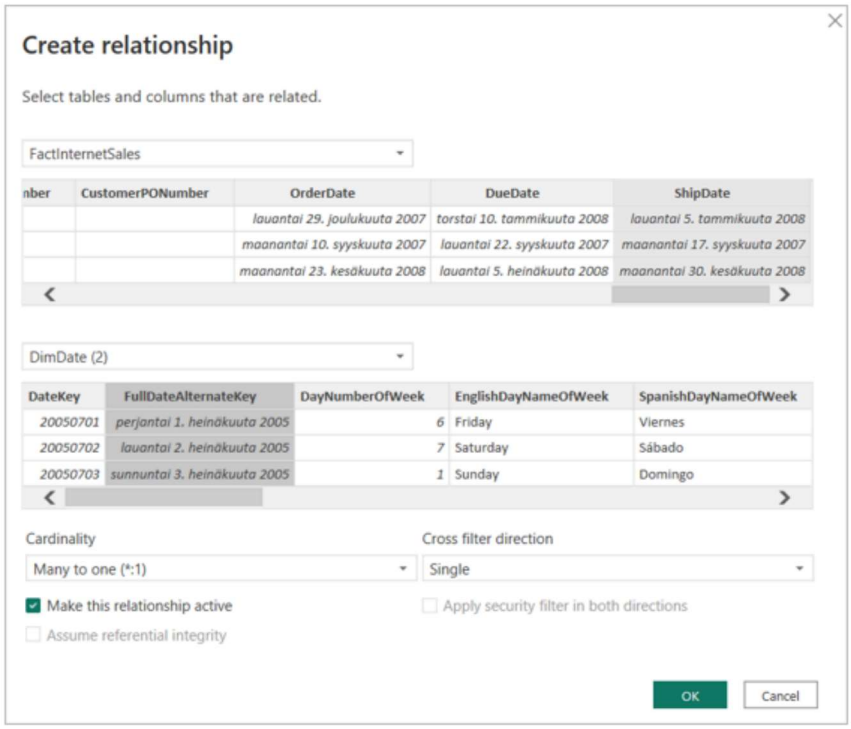


Step 3: . Open **Manage relationships.**

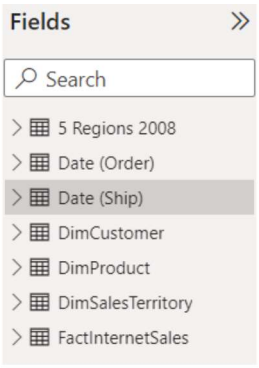


Step 4: From the relationship editor, click **New** to create a new relationship, and complete the following steps:

1. Select **FactInternetSales** from the drop-down list
2. Select the **ShipDate** column; use the scroll bar to scroll all the way to the right
3. Select **DimDate (2)** from the drop-down list
4. Select the **FullDateAlternateKey** column
5. Click **OK,** and **Close** to close the **Create relationship** window.



Step 5: Rename colums *DimDate* to *Date (Order)*, and *DimDate (2)* to *Date (Ship).*



1. In *Report view* create a new page by clicking **+**
2. In *Report view* select **Table** in **Visualizations**
3. In **Fields** select *5 Regions 2008* > **Temperature Range**
4. In **Fields** select *FactInternetSales* > **Sales Amount**

You can observe the analytical benefit of having a shipping date table and an order date table in the same data model. In this example, the total sales are being displayed in a matrix visual with the year from the order date table on the rows and the year from the shipping date table on the columns:

1. The value of $3,266,374 is the number of total sales that were made in the year 2005.
2. The value of $3,105,587 is the number of total sales that were shipped in the year 2005.
3. If you take a look at the column 2006 (ShipDate), you will notice that $6,576,979 of sales shipped in 2006. Upon closer inspection, $160,786 of what shipped in 2006 was actually ordered in 2005 and the remaining $6,416,193 was ordered in 2006.

Some of the cons of importing the table multiples times are:

- Resources: Additional memory and space will be used.
- Administrative changes: Any modifications made to one table will need to be repeated for all tables, as these tables are not linked. For example, if you create a hierarchy in one table, then you would need to create a hierarchy in all date tables.
- Time Intelligence: Time intelligence calculations will need to be rewritten for each date table.

The report in *Figure* shows total sales and total transactions by year, but which year? Is this the year that a product was purchased or the year a product was shipped? The active relationship is on the order date, so the report is displaying the results based on when the product was purchased:

CalendarYear	Total Sales	Total Transactions
2005	$3,266,373.66	1013
2006	$6,530,343.53	2677
2007	$9,791,060.30	24443
2008	$9,770,899.74	32265
Total	**$29,358,677.22**	**60398**

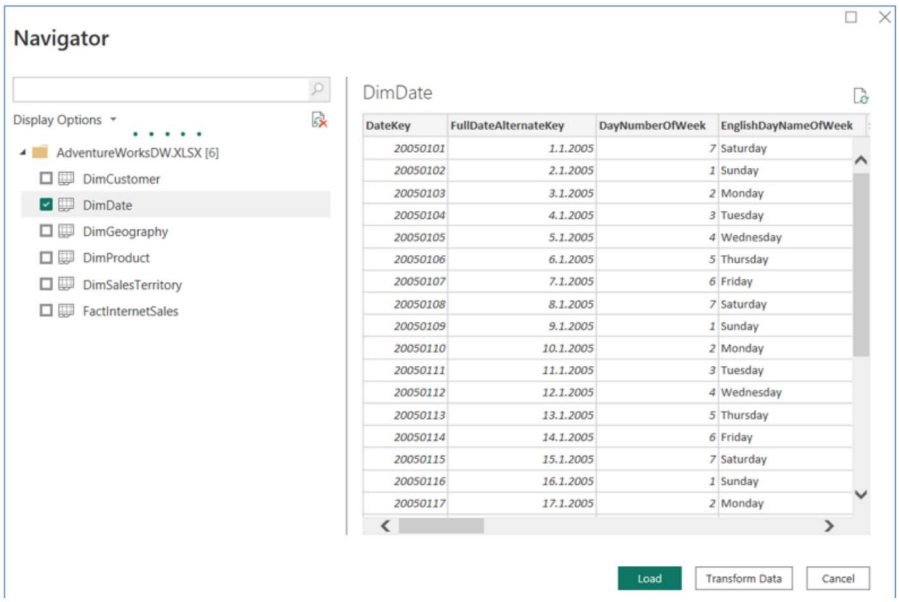
The previous visualization is correct but it is ambiguous. To remove any uncertainty from our reports, the data model can be further improved by renaming columns. In the next section, you will learn how to make small changes in your data model so that the visuals are more specific.

Task 4 - Import the date table

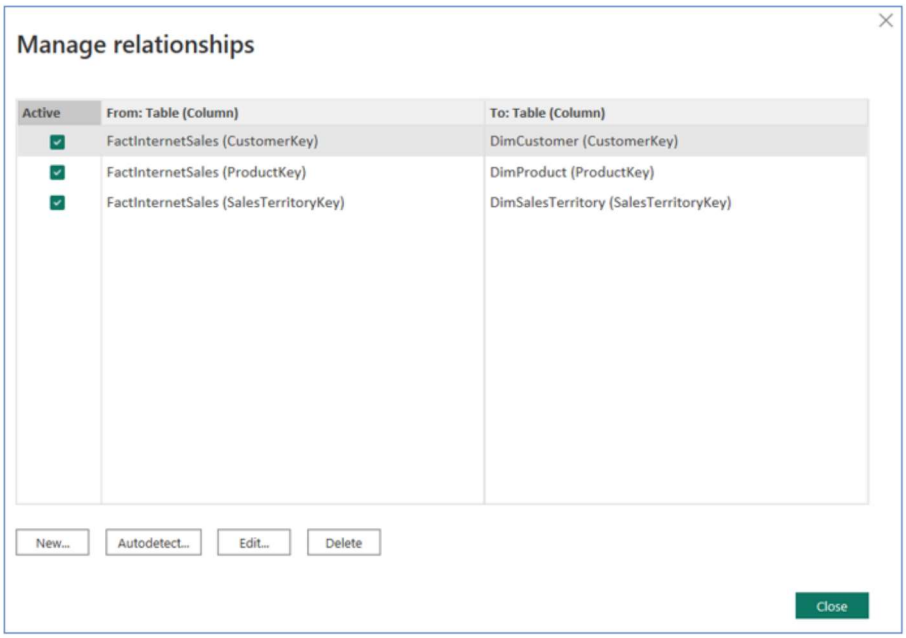
In this section, we are going to import a date table to support the analysis of data based on when an order shipped.

Step 1: From the Get data option, select **Excel** and open the **AdventureWorksDW** Excel file from *C:\PBExams*.

Step 2: Select DimDate from the list of tables, and then click **Load**:



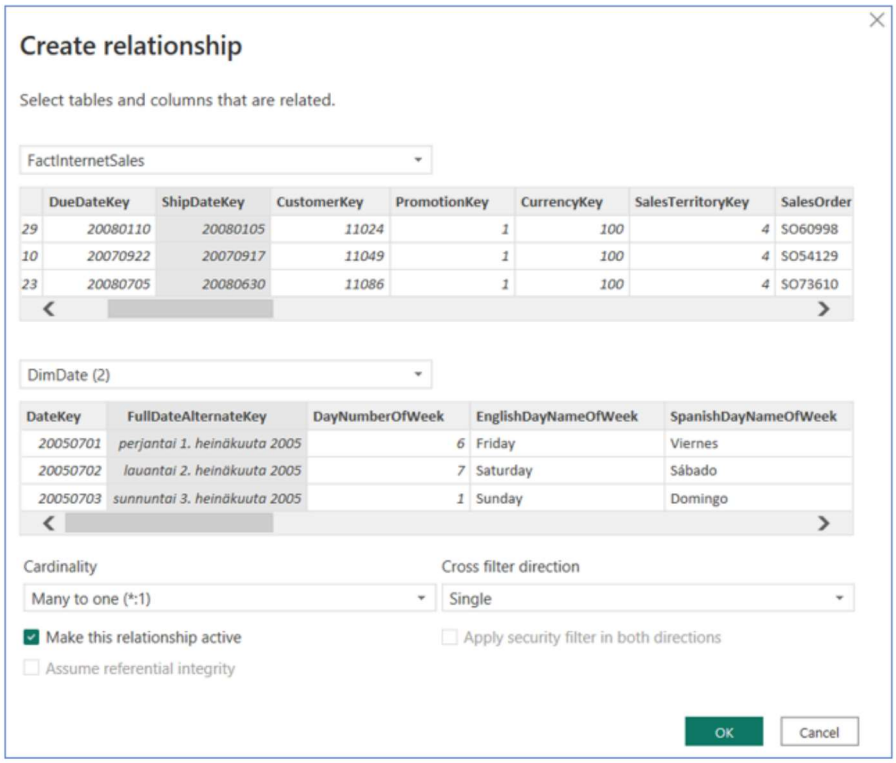
Step 3: Now that the data has been imported, the next step is creating a valid relationship. Open **Manage relationships** from either the **Home** or **Modeling** ribbon, depending on which view you are currently in.



From the relationship editor, click **New** to create a new relationship.

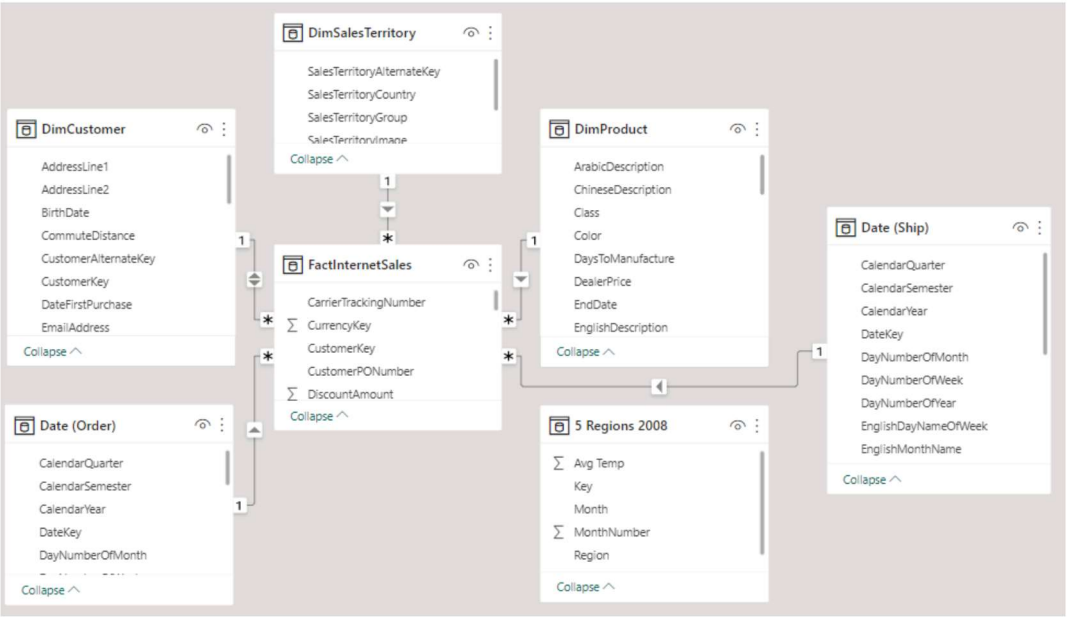
Step 4: Complete the following steps:

1. Select *FactInternetSales* from the drop-down list
2. Select the *ShipDate* column; use the scroll bar to scroll all the way to the right
3. Select *DimDate (2)* from the drop-down list
4. Select the *FullDateAlternateKey* column
5. Click **OK,** and **Close** to close the *Create relationship window*.



The table and column names here were changed, for clarity. You will learn how to rename tables and columns in the following *Usability enhancements* task. *DimDate* has been renamed *Date (Order)*. *DimDate (2)* has been renamed *Date (Ship)*.

The data model now has two date tables, each with an active relationship to the *FactInternetSales* table.



Step 5: Create a report to show sales by *Order Year:*

Order Year	SalesAmount
2005	3 266 373,66
2006	6 530 343,53
2007	9 791 060,30
2008	9 770 899,74
Total	**29 358 677,22**

Step 6: Create a report to show sales by *Ship Year:*

Ship Year	SalesAmount
2005	3 105 587,33
2006	6 576 978,98
2007	9 517 548,53
2008	10 158 562,38
Total	**29 358 677,22**

Importing the same table multiple times is the easiest solution to implement for new users to Power BI because this method doesn't require writing any DAX. This method is easy to explain to end users and allows you to reuse most of your existing DAX calculations.

The alternative method is to create inactive relationships and then create new calculations (measures) using the DAX language. This method of leveraging inactive relationships can become overwhelming from an administrative point of view. Imagine having to create copies of the existing measures in the data model for each relationship between two tables. In the current data model, *FactInternetSales* stores three dates, and this would possibly mean having to create and maintain three copies of each measure, one to support each date.

End-of-Exercise

Olet suorittanut 100 % oppitunnista

100%

◄ Exercise 8 - Building relationships

Siirry...

Exercise 10 - Usability enhancements ►