

# Designing Dataflows to Extract and Transform Data

**Lab Time:** 40 minutes

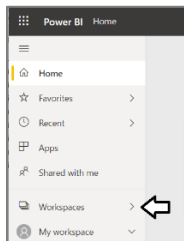
**Lab Folder:** C:\Student\Modules\02\_Dataflows\Lab\

**Lab Overview:** In this lab you will begin by creating a new app workspace and a new dataflow. Next, you will learn to work with the Power Query features in the browser to extract data from a SQL Azure database and to transform the data as it is loaded into dataflow storage in the app workspace. After you create the dataflow, you will create a Power BI Desktop project in which you will import the dataflow entities to create a starting point for a data model.

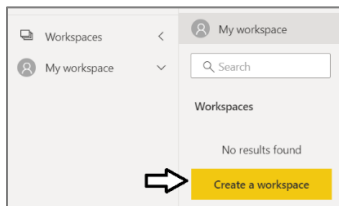
## Exercise 1: Use Power Query to Create a New Dataflow

In this exercise you will create a new workspace named **Wingtip Sales Model** and a new dataflow named **Wingtip Sales Dataflow**. This will give you a chance to learn how Power Query is used to create new dataflow entities. You will create four entities by extracting data a SQL Server database running in the Microsoft Azure cloud.

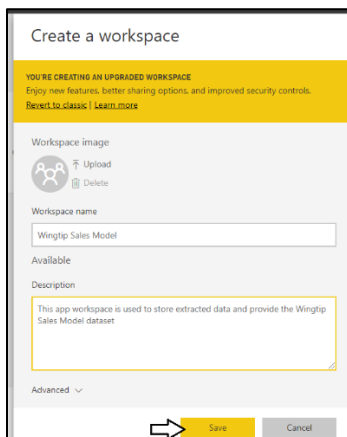
1. Log into the Power BI Service with your new organizational account.
  - a) Navigate the Power BI portal at <https://app.powerbi.com> and if prompted, log in using your organizational account.
2. Create a new app workspace named **Wingtip Sales Model**.
  - a) Click the **Workspace** flyout menu in the left navigation.



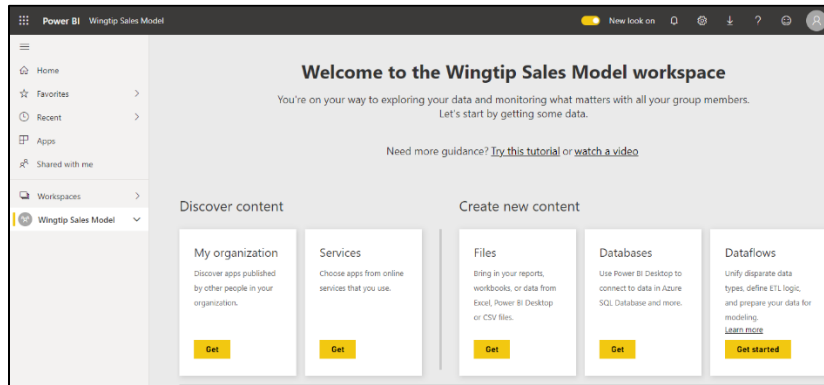
- b) Click the **Create app workspace** button to display the **Create an app workspace** dialog.



- c) In the **Create an app workspace** pane, enter a workspace name of **Wingtip Sales Model**.
- d) Click the **Save** button to create the new app workspace named **Wingtip Sales Model**.



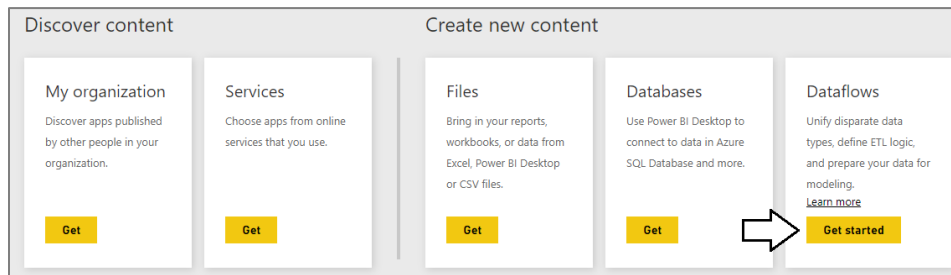
- e) When you click **Save**, the Power BI service should create the new app workspace and then switch your current Power BI session to be running within the context of the new **Wingtip Sales Model** workspace.



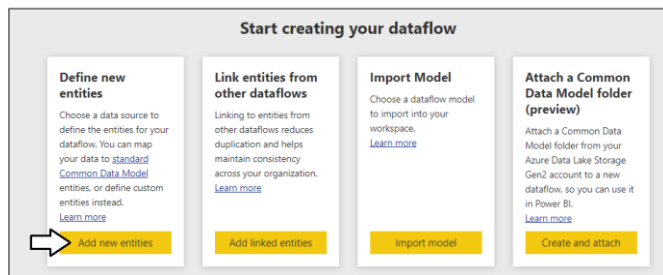
You should now see the welcome page for the **Wingtip Sales Model** workspace. The welcome page is shown by default when an app workspace is empty and contains no content.

### 3. Create a new dataflow.

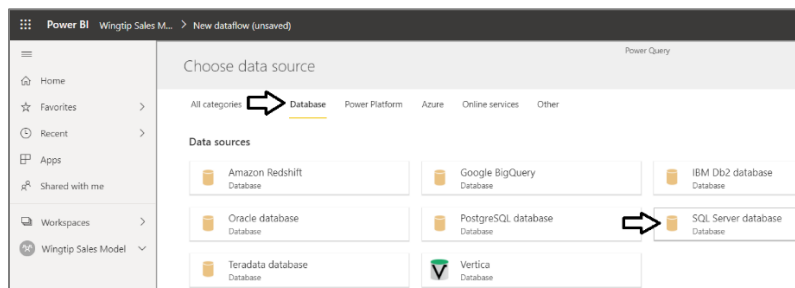
- a) Click the **Get started** button in the **Dataflows** section.



- b) Click the **Add new entities** button in the **Define new entities** section.



- c) On the **Choose data source** page, select the **Database** tab and then select **SQL Server database**.



- d) Enter a **Server** value of **cpt.database.windows.net**.
- e) Enter a **Database** value of **WingtipSalesDB**.

Connect to data source

SQL Server database Database

Connection settings

Server  
cpt.database.windows.net

Database  
WingtipSalesDB

Connection credentials

- f) Move down the **Connection credentials** section.
- g) Set the **Authentication kind** setting to **Basic**.
- h) Enter a **Username** of **CptStudent**.
- i) Enter a **Password** of **pass@word1**
- j) Click the **Next** button to continue.

Connection credentials

On-premises data gateway  
(none)

Authentication kind  
Basic

Username  
CptStudent

Password  
pass@word1

Cancel Next

Once you establish a connection, you should be promoted with the **Choose data** screen to select one or more tables.

- 4. Select the **Customers** table
  - a) In the **Choose data** section, select the **Customers** table and then click **Transform data**.

Choose data

Search

Display options

SQL Server database [11]

- sys.database\_firewall\_rules
- view\_CustomerDimension
- view\_ProductDimension
- view\_ProductSalesfacts
- view\_PurchaseDimension
- Customers**
- InvoiceDetails
- Invoices
- Products
- sysdiagrams
- fn\_diagramobjects

Select related tables

Back

Customers

CustomerId	FirstName	LastName	Company	EmailAddress
1	Nina	Diaz	Dunder Mifflin	Nina.Diaz@DunderMifflin
2	Melinda	Carter	Vandelay Industries	Melinda.Carter@Vandelay
3	Pam	Miller	Brown Streak Railroad	Pam.Miller@BrownStreak
4	Merle	Blackwell	Krusty Burger	Merle.Blackwell@KrustyB
5	Ariel	Hale	ComTron	Ariel.Hale@ComTron.cor
6	Randy	Carter	ComTron	Randy.Carter@ComTron
7	Lillie	Hinton	Blaidd Dwrg Nuclear Power Plant	Lillie.Hinton@BlaiddDwn
8	Ladonna	Moody	Porter Automobiles	Ladonna.Moody@Porter
9	Buddy	McKay	InGen	Buddy.McKay@InGen.co
10	Warren	Sykes	Cyberdyne Systems	Warren.Sykes@Cyberdyr
11	Jan	Rutledge	Umbrella Corporation	Jan.Rutledge@Umbrella
12	Dallas	Lester	Strickland Propane	Dallas.Lester@Stricklan
13	Matthew	Zimmerman	Groovy Smoothie	Matthew.Zimmerman@C
14	Sheryl	Hernandez	Scavo's Pizzeria	Sheryl.Hernandez@Scav
15	Bradley	Cannon	Trade Federation	Bradley.Cannon@TradeF
16	Markus	Soto	Groovy Smoothie	Markus.Soto@GroovySrr
17	Roderick	Owens	Global Dynamics	Roderick.Owens@Global

Cancel Transform data

- b) You should now see the **Edit queries** screen displaying query results and the M formula bar for the current step.

Edit queries					
Get data Manage parameters Refresh Options Manage columns Transform table Reduce rows Add column					
Queries <					
Customers					
= Source([Schema = "dbo", Item = "Customers"])[Data]					
Customerid	FirstName	LastName	Company	EmailAddress	
1	Nina	Diaz	Dunder Mifflin	Nina.Diaz@DunderMifflin.com	
2	Melinda	Carter	Vandelay Industries	Melinda.Carter@VandelayIndustries.com	
3	Pam	Miller	Brown Streak Railroad	Pam.Miller@BrownStreakRailroad.com	

5. Use Power Query to clean and transform the data from the **Customers** table.

- a) Drop down the **Manage columns** menu button and select the **Choose columns** command.

Edit queries					
Get data Manage parameters Refresh Options Manage columns Transform table Reduce rows					
Queries <					
Customers					
= Source([Schema = "dbo", Item = "Customers"])[Data]					
Customerid	FirstName	LastName	Company	EmailAddress	
1	Nina	Diaz	Dunder Mifflin	Nina.Diaz@DunderMifflin.com	

- b) In the **Choose columns** dialog, begin by clicking on the **(Select all)** checkbox at the top to unselect all column. Next, select the checkboxes for the following columns.
- Customerid**
  - FirstName**
  - LastName**
  - City**
  - State**
  - Zipcode**
  - Gender**
  - BirthDate**
  - FirstPurchaseDate**
  - LastPurchaseDate**
- c) Once you have the columns selected as shown in the following screenshot, click **OK** to close the **Choose columns** dialog.

Choose columns
Choose the columns to keep.

☒ (Select all)
☒ Customerid
☒ FirstName
☒ LastName
☐ Company
☐ EmailAddress
☐ WorkPhone
☐ HomePhone
☐ Address
☒ City
☒ State
☒ Zipcode
☒ Gender
☒ BirthDate
☒ FirstPurchaseDate
☒ LastPurchaseDate
☐ Invoices

- d) You should be able to verify that the Power Query editor now only shows the columns that you selected.

= Table.SelectColumns(Navigation, {"CustomerId", "FirstName", "LastName", "City", "State", "Zipcode", "Gender", "BirthDate"})								
1	2	3	4	5	6	7	8	9
CustomerId	FirstName	LastName	City	State	Zipcode	Gender	BirthDate	
1	Nina	Diaz	Eureka	CA	95501	F	4/11/1966, 12	
2	Melinda	Carter	Napa	CA	94558	F	6/6/1976, 12	
3	Pam	Miller	Napa	CA	94558	F	9/8/1952, 12	

6. In this step you will merge the **FirstName** column and the **LastName** column together into a single column named **Customer**.

- Select the **FirstName** column by clicking on its column header.
- Next, hold down the **SHIFT** key and select the **LastName** column by clicking on its column header.
- Right-click on the selected columns and click the **Merge columns** menu command.

= Table.SelectColumns(Navigation, {"CustomerId", "FirstName", "LastName", "City", "State", "Zipcode", "Gender", "BirthDate"})								
1	2	3	4	5	6	7	8	9
CustomerId	FirstName	LastName	City	State	Zipcode	Gender	BirthDate	
1	Nina	Diaz	Eureka	CA	95501	F	4/11/1966, 12	
2	Melinda	Carter	Napa	CA	94558	F	6/6/1976, 12	
3	Pam	Miller	Napa	CA	94558	F	9/8/1952, 12	
4	Merle	Blackwell	Napa	CA	94558	F	9/8/1952, 12	
5	Ariel	Hale	Napa	CA	94558	F	9/8/1952, 12	
6	Randy	Carter	Napa	CA	94558	F	9/8/1952, 12	
7	Lillie	Hinton	Napa	CA	94558	F	9/8/1952, 12	
8	Ladonna	McKay	Napa	CA	94558	F	9/8/1952, 12	
9	Buddy	McKay	Napa	CA	94558	F	9/8/1952, 12	

- In the **Merge Column** dialog, drop down the **Separator** control and select a value of **Space**.
- Add a **New column name** value of **Customer** and click the **OK** button to modify the underlying query with your changes.

Merge columns  
Choose how to merge the selected columns.

Separator  
Space

New column name (optional)  
Customer

OK Cancel

- f) You should now be able to see that the **FirstName** column and the **LastName** column have been replaced with a single merged column named **Customer**.

1	2	3	4	5
CustomerId	Customer	City	State	
1	Nina Diaz	Eureka	CA	
2	Melinda Carter	Napa	CA	
3	Pam Miller	Napa	CA	

7. Modify the query so that the **Gender** column returns values of **Male** and **Female** instead of **M** and **F**.

- Locate the **Gender** column in the **Customers** table.
- Right-click the header for the **Gender** column and select the **Replace values** command to display the **Replace values** dialog.

1	2	3	4	5	6	7	8	9
CustomerId	Customer	City	State	Zipcode	Gender	BirthDate		
1	Nina Diaz	Eureka	CA	95501	F	4/11/1966, 12		
2	Melinda Carter	Napa	CA	94558	F	6/6/1976, 12		
3	Pam Miller	Napa	CA	94558	F	9/8/1952, 12		
4	Merle Blackwell	Napa	CA	94558	F	9/8/1952, 12		
5	Ariel Hale	Napa	CA	94558	F	9/8/1952, 12		
6	Randy Carter	Napa	CA	94558	F	9/8/1952, 12		
7	Lillie Hinton	Napa	CA	94558	F	9/8/1952, 12		
8	Ladonna McKay	Napa	CA	94558	F	9/8/1952, 12		
9	Buddy McKay	Napa	CA	94558	F	9/8/1952, 12		

- c) In the **Replace values** dialog, enter a value of **F** in the **Value to find** textbox and enter a value of **Female** in the **Replace with** textbox. Click to **OK** button add your changes to the underlying query.

Replace values

Replace one value with another in the selected columns.

☒ Basic ☐ Advanced

Value to find  
F

Replace with  
Female

OK Cancel

- d) You should be able to see that all values of **F** in the **Gender** column have been replaced with a value of **Female**.

Gender
Female
Female
Female
Female
M
M
Female

- e) Right-click the header for the **Gender** column and select the **Replace values** command a second time.
- f) In the **Replace values** dialog, enter a value of **M** in the **Value to find** textbox and enter a value of **Male** in the **Replace with** textbox. Click to **OK** button add your changes to the underlying query.

Replace values

Replace one value with another in the selected columns.

☒ Basic ☐ Advanced

Value to find  
M

Replace with  
Male

OK Cancel

- g) You should be able to confirm that all values in the **Gender** column have been replaced with a value of either **Male** or **Female**.

8. Change the name of query steps.

- a) Inspect the **Applied Steps** list in the **Query settings** pane. You should be able to see that there are two steps at the end that have been given the generic names of **Replaced value** and **Replaced value 1**.

Replacer.ReplaceText,

Gender	BirthDate
Female	4/11/1966, 12:00:00 A
Female	6/6/1976, 12:00:00 A
Female	9/8/1952, 12:00:00 A
Female	9/12/1939, 12:00:00 A
Male	9/15/1965, 12:00:00 A
Male	7/14/1953, 12:00:00 A
Female	2/3/1992, 12:00:00 A
Female	4/5/1949, 12:00:00 A
Male	5/10/1989, 12:00:00 A
Male	6/17/1960, 12:00:00 A
Female	11/26/1981, 12:00:00 A
Male	3/26/1973, 12:00:00 A

Query settings

Name  
Customers

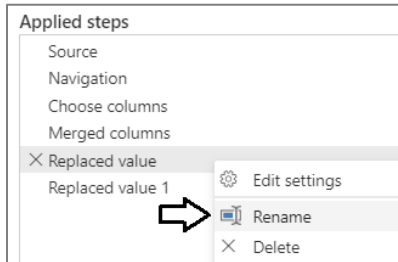
Entity type ⓘ  
Custom

Applied steps

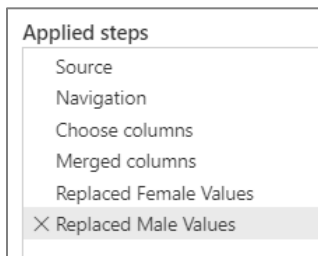
- Source
- Navigation
- Choose columns
- Merged columns
- Replaced value
- Replaced value 1

In order to promote higher levels of maintainability, it's often a good idea to rename steps that are given generic names such as **Replaced value** and **Replaced value 1**.

- b) Rename the **Replaced Values** step by right-clicking it and clicking the **Rename** command to place the step name in edit mode. Modify the name of this step to **Replace Female Values**.



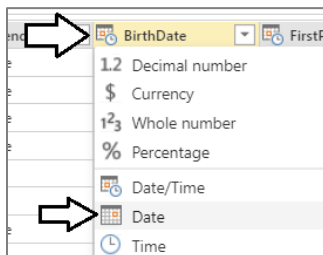
- c) Using the same technique, rename the **Replaced Value 1** step to **Replaced Male Values**.



You have now learned how to rename a query step. Note that this lab exercise will not continue to ask you to change the name of every step due to time constraints. However, when you are creating queries in larger, real-world projects that involve multiple team members, it's a good practice to rename query steps to make your query logic easier for others to read, understand and extend.

9. Change the column type of **BirthDate**, **FirstPurchasedDate** and **LastPurchasedDate** from **Date/Time** to **Date**.

- a) Use the column type drop down on the left-hand side of the **BirthDate** column to configure the column using the **Date** type.

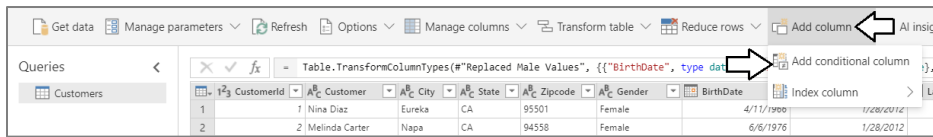


- b) Use the datatype drop down menu of the **FirstPurchaseDate** column to configure the column using the **Date** type.  
c) Use the datatype drop down menu of the **LastPurchaseDate** column to configure the column using the **Date** type.  
d) You should see that the three columns now show values with a date but without a time.

BirthDate	FirstPurchaseDate	LastPurchaseDate
4/11/1966	1/28/2012	1/28/2012
6/6/1976	1/28/2012	1/28/2012
9/8/1952	1/28/2012	1/28/2012
9/12/1939	1/28/2012	1/28/2012
9/15/1965	1/28/2012	1/28/2012
7/14/1953	1/28/2012	1/28/2012

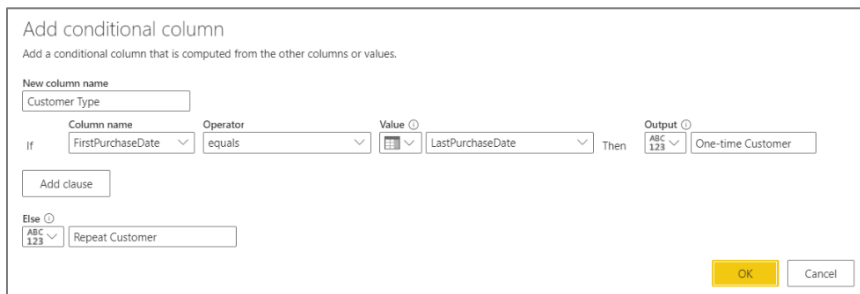
10. Add a new conditional column named **Customer Type** to indicates whether the customer is a repeat customer or not.

- a) Drop down the **Add column** menu button and select the **Add conditional column** command.



In this particular scenario, you are working under the assumption that the customer is a repeat customer when the **FirstPurchaseDate** column and the **LastPurchaseDate** column are not equal indicating the customer has made two or more purchases.

- b) In the **Add conditional column** dialog, enter a **New column name** value of **Customer Type**.  
c) Configure a rule to return a string value of "One-time Customer" if **FirstPurchaseDate** equals **LastPurchaseDate**.  
d) For the **Else** evaluation, return a string value of "Repeat Customer".  
e) When the **Add conditional column** dialog matches the screenshot below, click the **OK** button to add the new column.



- f) You should be able to verify that the new **Customer Type** column has a value of **Repeat Customer** when the current customer has a **FirstPurchaseDate** column value that is not equal to the **LastPurchaseDate** column value. When these column values are equal, the **Customer Type** column has a value of **One-time Customer**.

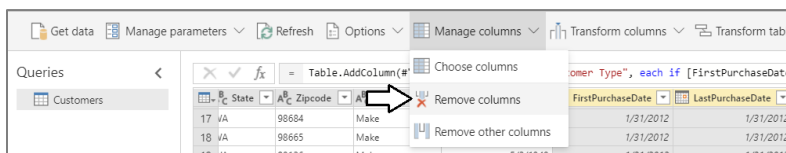
You might have to scroll down several pages of records in the **Customers** query results before you begin to see repeat customers.

FirstPurchaseDate	LastPurchaseDate	Customer Type
2/1/2012	2/1/2012	One-time Customer
2/1/2012	2/1/2012	One-time Customer
2/1/2012	2/1/2012	One-time Customer
2/1/2012	10/21/2015	Repeat Customer
2/1/2012	2/1/2012	One-time Customer
2/1/2012	2/1/2012	One-time Customer

Now, that you are done using the **FirstPurchaseDate** column and the **LastPurchaseDate** column to calculate the value of the **Customer Type** column, you can delete them because they are no longer needed.

11. Remove the **FirstPurchaseDate** column and the **LastPurchaseDate** column.

- a) Select the **FirstPurchaseDate** column by clicking its column header.  
b) Hold down the **SHIFT** key and click the column header for **LastPurchaseDate** so that both columns are selected.  
c) Right click the one of the selected columns and click the **Remove Columns**.



- d) You should be able to confirm that the **FirstPurchaseDate** column and the **LastPurchaseDate** columns have been removed from the query results. However, the **Customer Type** column is still there.



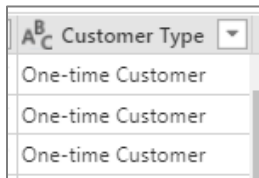
12. Set the column type for the **Customer Type** column to **Text**.

- a) You might notice that column type menu for the **Customer Type** column is not set to a specific type.

ABC Gender	BirthDate	ABC 123 Customer Type
Female	4/11/1966	One-time Customer
Female	6/6/1976	One-time Customer
Female	9/8/1952	One-time Customer

When you see the type as **ABC** above and **123** below, that the column is being assigned the generic **Any** type. In general, it is bad to leave a column configured to the type of **Any**. Instead, you should configure each column to return a specific type.

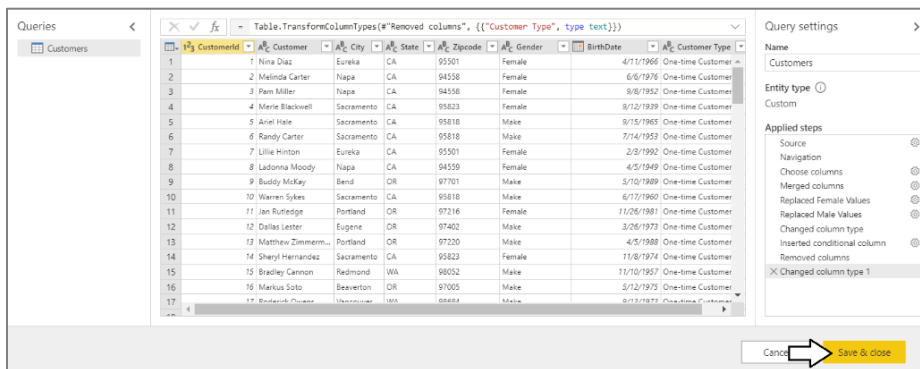
- b) Drop down the Type menu for the **Customer Type** column and set its value to **Text**.



You have now completed the work with Power Query to create a new dataflow entity named **Customers**. You will now save the new dataflow and give it a name of **Wingtip Sales Data**.

13. Save the dataflow with a name of **Wingtip Sales Data**.

- a) Click the **Save & close** button at the bottom right of the **Queries** window.



- b) When prompted by the **Save your dataflow** dialog, enter a name of **Wingtip Sales Data** and then click **Save**.

### Save your dataflow

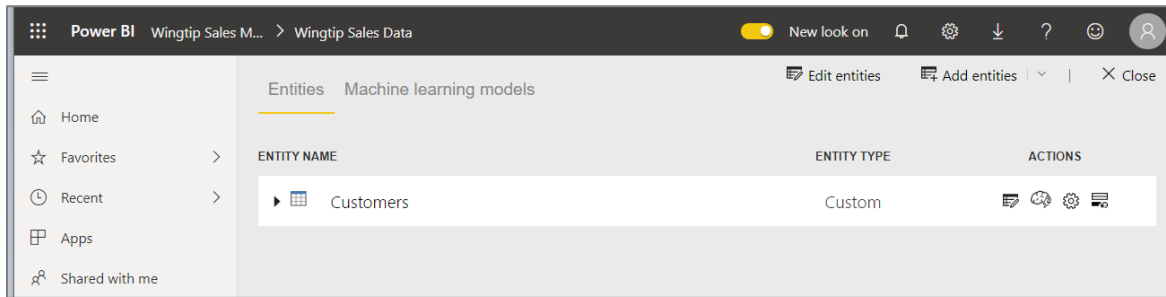
Name \*

Description

A dataflow to extract, clean and prepare data to analyze Wingtip sales performance.

Save Cancel

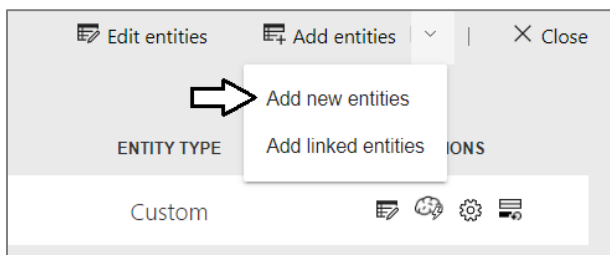
- c) You should now see the summary page for the **Wingtip Sales Data** dataflow.



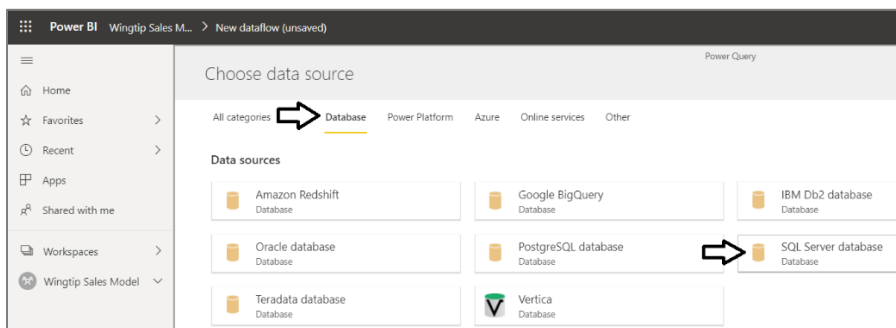
## Exercise 2: Extend the Dataflow by Adding Entities for Products and Sales

In the following exercise, you will use Power Query to add additional entities to the **Wingtip Sales Data** dataflow.

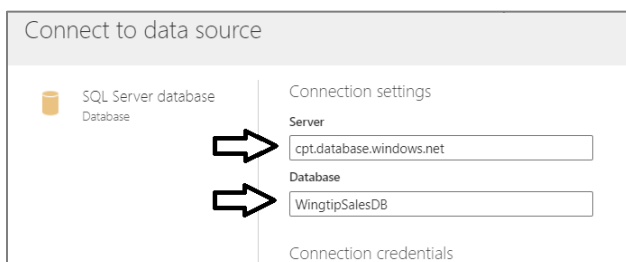
1. Add three new entities to the **Wingtip Sales Data** dataflow.
  - a) Locate the **Add entities** menu button in the top right corner of the **Wingtip Sales Data** dataflow summary page.
  - b) Drop down the **Add entities** menu button and select the **Add new entities** command.



- c) On the **Choose data source** page, select the **Database** tab and then select **SQL Server database**.



- d) Enter a **Server** value of **cpt.database.windows.net**.
    - e) Enter a **Database** value of **WingtipSalesDB**.



You should **NOT** be required to enter database credentials. That's because you already entered the credentials for this database in exercise 1 and the credentials are now stored in the Microsoft cloud and will be used transparently.

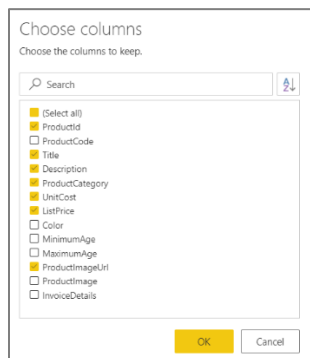
- f) Verify that values for **Username** and **Password** were automatically populated when you entered **Server** and **Database**.

- g) Click the **Next** button to continue.  
h) When prompted to **Choose data**, select the three tables named **InvoiceDetails**, **Invoices** and **Products**.  
i) Click the **OK** button to add three new entity queries to the dataflow.

- j) You should now see three new queries on the **Edit queries** page named **InvoiceDetails**, **Invoices** and **Products**.

- k) Select the **Products** query from the **Queries** list on the left so you can begin to modify its query logic with Power Query.

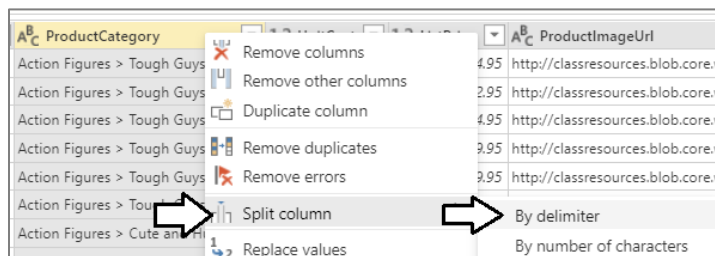
2. Remove the columns that are not required in the **Products** query results.
  - a) Click the **Choose Columns** button in the ribbon to display the **Choose Columns** dialog.
  - b) In the **Choose Columns** dialog, begin by clicking on the **(Select all)** checkbox at the top to unselect all columns.
  - c) Select the checkboxes for **ProductId**, **Title**, **Description**, **ProductCategory**, **UnitCost**, **ListPrice** and **ProductImageUrl** as shown in the following screenshot.



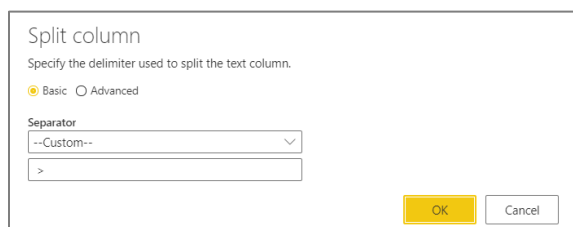
- d) Click the **OK** button to close the **Choose Columns** dialog.
3. Rename the **Title** column to **Product**.
  - a) Right-click on the **Title** column and click **Rename**.
  - b) Update the column name to **Product**.

	1 2 3 ProductId	A B C Product	A B C Description
1	1	Batman Action Figure	A super hero who sometimes plays the role of a dark knight.
2	2	Captain America Action Figure	A super action figure that protects freedom and the American way.
3	3	GI Joe Action Figure	A classic action figure from the 1970s.

4. Split the **ProductCategory** column up into two separate columns named **Category** and **Subcategory**.
  - a) Right-click the **ProductCategory** column and then click the **Split column > By delimiter** command.



- b) In the **Split column** dialog, drop down the **Separator** combo box and select **--Custom--**.
  - c) In the textbox enter a three-character text value which includes a space follow by the **>** character followed by another space.
  - d) When the **Split column** dialog matches the following screenshot, click the **OK** button.



- e) You should be able to confirm that Power BI Desktop has split the **ProductCategory** column into two separate columns named **ProductCategory.1** and **ProductCategory.2**.

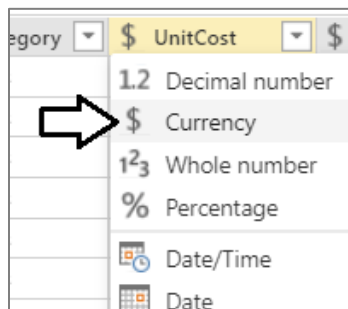
ProductCategory.1	ProductCategory.2
Action Figures	Tough Guys
Action Figures	Tough Guys
Action Figures	Tough Guys
Action Figures	Tough Guys
Action Figures	Tough Guys
Action Figures	Tough Guys
Action Figures	Cute and Huggable
Action Figures	Cute and Huggable

- f) Rename the **ProductCategory.1** column to **Category** and rename **ProductCategory.2** to **Subcategory**.

Category	Subcategory
Action Figures	Tough Guys
Action Figures	Tough Guys
Action Figures	Tough Guys
Action Figures	Tough Guys
Action Figures	Tough Guys
Action Figures	Tough Guys
Action Figures	Cute and Huggable
Action Figures	Cute and Huggable

5. Modify the column type of the **UnitCost** column and the **ListPrice** column to the **Currency** type.

- a) Use the dropdown column type menu to set the type of the **UnitCost** to **Currency**.



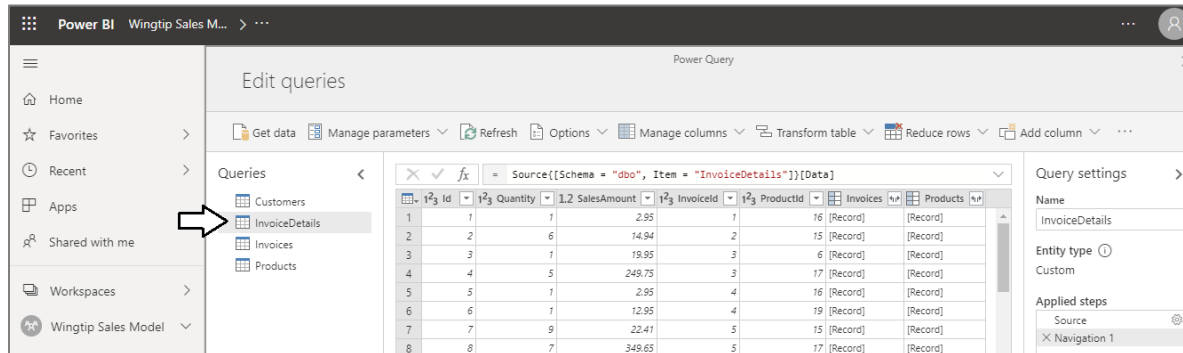
- b) Use the dropdown column type menu to set the type of the **ListPrice** to **Currency**.

UnitCost	ListPrice
6.85	14.95
7.05	12.95
6.1	14.95
2.85	9.95
2.85	9.95
14.25	19.95
12	21.95

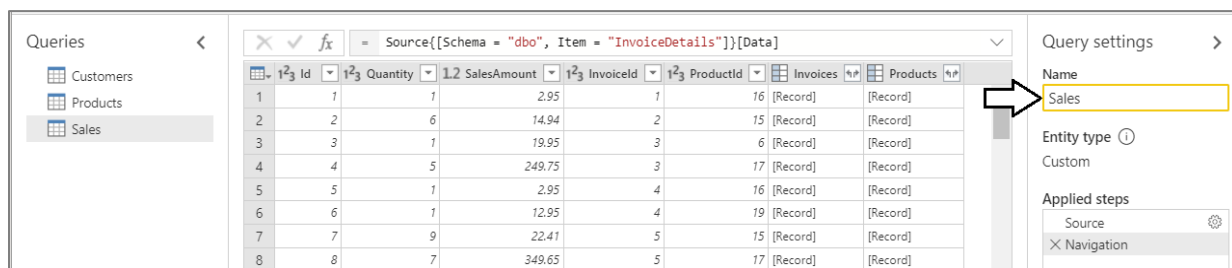
- c) You have now completed your work on the **Products** query.

One important point of flexibility in the import process is that you can change the name of a query, and therefore the name of the resulting entity in the dataflow. Try to use entity names that are more intuitive and easier to understand. In the next step you will rename the **InvoiceDetails** query to **Sales** to indicate that it is a fact table containing sales data.

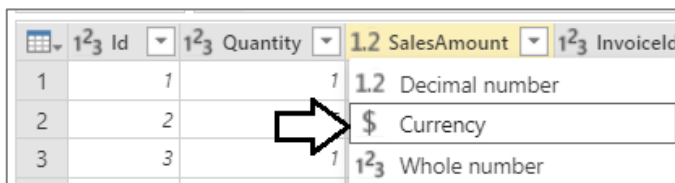
6. Rename the **InvoiceDetails** query to **Sales**.
  - a) Select the **InvoiceDetails** query from the **Queries** list on the left.



- b) Update the name of the **InvoiceDetails** query to **Sales** by replacing the text in the **Name** textbox in the **Query Settings** pane.

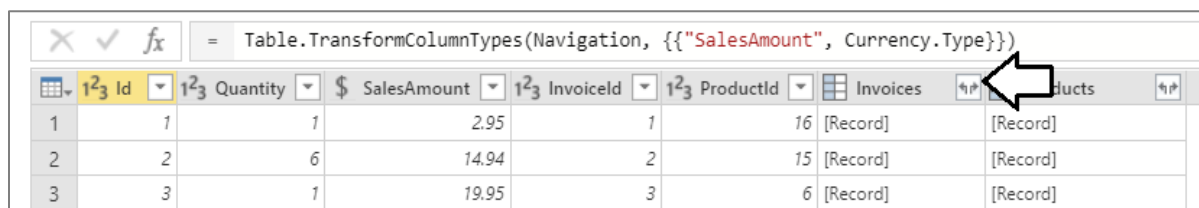


7. Modify the columns of the **Sales** query.
  - a) Modify the column type of the **SalesAmount** column to the **Currency** type.

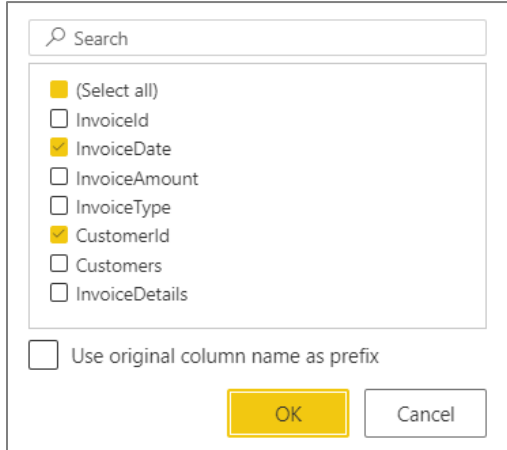


In order for the **Sales** table to play the role of a fact table, the **Sales** query must merge in additional columns from the **Invoices** table. You must pull in the **CustomerId** column so that you can create a direct relationship between the **Sales** table and the **Customers** table. You will also pull in the **InvoiceDate** so that you can analyze data in the **Sales** table over time.

8. Expand the **Invoices** column to add the **InvoiceDate** column and the **CustomerId** column to the **Sales** query.
  - a) Click the **Expand** button inside the column header of the **Invoices** column to display the **Columns to Expand** dialog.



- b) In the **Columns to Expand** dialog, begin by clicking on the **(Select all)** checkbox at the top to unselect all columns.
- c) Select the checkboxes for the **InvoiceDate** column and the **CustomerId** column.
- d) Make sure to uncheck the checkbox with the caption **Use original column name as prefix**.
- e) Click the **OK** button to close the dialog and to modify the underlying query.



The dialog box titled 'Columns to Expand' contains a search bar at the top. Below it is a list of columns with checkboxes: (Select all) [unchecked], Invoiceld [unchecked], InvoiceDate [checked], InvoiceAmount [unchecked], InvoiceType [unchecked], CustomerId [checked], Customers [unchecked], and InvoiceDetails [unchecked]. At the bottom, there is a checkbox for 'Use original column name as prefix' which is unchecked, and two buttons: 'OK' and 'Cancel'.

- f) You should see that the **InvoiceDate** column and the **CustomerId** column have now been added to the **Sales** query results.

123 ProductId	123 InvoiceDate	123 CustomerId	Products
16	1/28/2012, 12:00:00 ...	1	[Record]
15	1/28/2012, 12:00:00 ...	2	[Record]
6	1/28/2012, 12:00:00 ...	3	[Record]
17	1/28/2012, 12:00:00 ...	3	[Record]

- g) Change the column type of the **InvoiceDate** to the **Date** type.

es("#Expanded Invoices", {{"InvoiceDate", type date}})

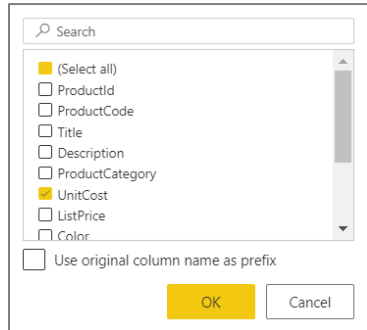
123 Invoiceld	123 ProductId	123 InvoiceDate	123 CustomerId	Products
5	1	16	1/28/2012	1 [Record]
4	2	15	1/28/2012	2 [Record]
5	3	6	1/28/2012	3 [Record]

9. Expand the **Products** column to add the **UnitCost** column to the **Sales** query.

- a) Click the Expand button inside the column header of the **Products** column to display the **Columns to Expand** dialog.

123 InvoiceDate	123 CustomerId	Products
1/28/2012	1	[Record]
1/28/2012	2	[Record]
1/28/2012	3	[Record]

- b) In the **Columns to Expand** dialog, begin by clicking on the **(Select all)** checkbox at the top to unselect all columns.
- c) Select the checkbox for the **UnitCost** column.
- d) Uncheck the checkbox with the caption **Use original column name as prefix**.
- e) Once the **Columns to Expand** dialog matches the following screenshot, click the **OK** button to close the dialog.



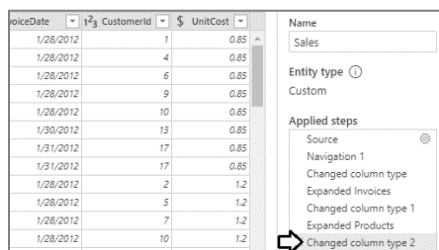
- f) You should see that the **UnitCost** column has now been added to the **Sales** query results.
- g) Modify the column type of the **UnitCost** column to the **Currency** type.

	InvoiceDate	CustomerId	UnitCost
5	1/28/2012	1	0.85
5	1/28/2012	4	0.85
5	1/28/2012	6	0.85
5	1/28/2012	9	0.85
5	1/28/2012	10	0.85

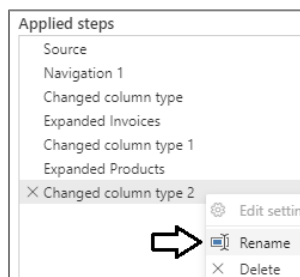
In the next step you will add a custom column which requires modifying the query's underlying M code in the Advanced editor window. Before opening the Advanced editor window, you will first rename the last step in the query to remove spaces from its name. The reason for doing this is that it will make the M code easier to read and write.

10. Rename the last step in the **Sales** query to remove spaces.

- a) Locate the final step in the **Applied steps** list in **Sales** query which has a name such as **Changed column type 2**.

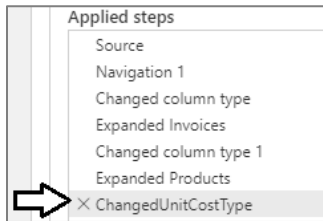


- b) Right-click on the last step the select the **Rename menu** command.



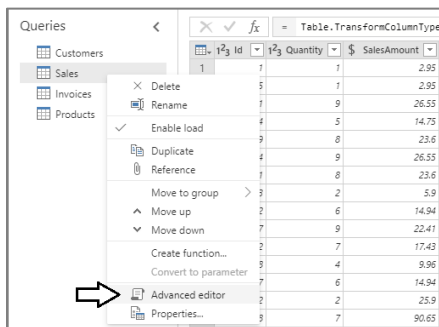


- c) Change the name of the step to **ChangedUnitCostType** so the step name contains no spaces.



11. Add a new custom column named **ProductCost** to calculate the product of the **Quantity** field multiplied by the **UnitCost** field

- a) Right-click on the **Sales** query in the **Queries** list on the right and select the **Advanced editor** menu command.



- b) Inspect the M expression in the Advanced editor window.



- c) Locate the following line of code for the **ChangedUnitCostType** step at the end of the **let** block just before the **in** block.

```
ChangedUnitCostType = Table.TransformColumnTypes("#Expanded Products", {"UnitCost", Currency.Type})
```

- d) Add a comma at the end of that line and then add a line break so you can add a new line of M code at the end of the **let** block.

```
ChangedUnitCostType = Table.TransformColumnTypes("#Expanded Products", {"UnitCost", Currency.Type}),
```

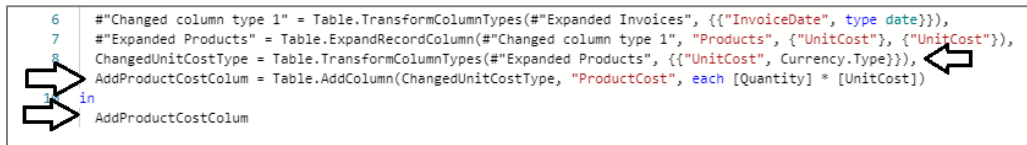
- e) Add the following line of M code as the last line of the **let** block.

```
AddProductCostColumn = Table.AddColumn(ChangedUnitCostType, "ProductCost", each [Quantity] * [UnitCost])
```

- f) Modify the line of code at the bottom in the **in** block to replace **ChangedUnitCostType** with **AddProductCostColumn**.

```
ChangedUnitCostType = Table.TransformColumnTypes("#Expanded Products", {"UnitCost", Currency.Type}),
AddProductCostColumn = Table.AddColumn(ChangedUnitCostType, "ProductCost", each [Quantity] * [UnitCost])
in
AddProductCostColumn
```

- g) The M code at the end of the query should match the following screenshot.



```

6  #"Changed column type 1" = Table.TransformColumnTypes(#"Expanded Invoices", {"InvoiceDate", type date})),
7  #"Expanded Products" = Table.ExpandRecordColumn(#"Changed column type 1", "Products", {"UnitCost"}, {"UnitCost"}),
8  ChangedUnitCostType = Table.TransformColumnTypes(#"Expanded Products", {"UnitCost", Currency.Type})),
9  AddProductCostColumn = Table.AddColumn(ChangedUnitCostType, "ProductCost", each [Quantity] * [UnitCost])
10 in
11 AddProductCostColumn
    
```

- h) The code in the Advanced editor window should now match the following code listing.

```

let
    Source = Sql.Database("cpt.database.windows.net", "wingtipsalesDB"),
    #"Navigation 1" = Source[Schema = "dbo", Item = "InvoiceDetails"][Data],
    #"Changed column type" = Table.TransformColumnTypes(#"Navigation 1", {"SalesAmount", Currency.Type})),
    #"Expanded Invoices" = Table.ExpandRecordColumn(#"Changed column type", "Invoices", {"InvoiceDate", "CustomerId"}, {"InvoiceDate", "CustomerId"}),
    #"Changed column type 1" = Table.TransformColumnTypes(#"Expanded Invoices", {"InvoiceDate", type date})),
    #"Expanded Products" = Table.ExpandRecordColumn(#"Changed column type 1", "Products", {"UnitCost"}, {"UnitCost"}),
    ChangedUnitCostType = Table.TransformColumnTypes(#"Expanded Products", {"UnitCost", Currency.Type})),
    AddProductCostColumn = Table.AddColumn(ChangedUnitCostType, "ProductCost", each [Quantity] * [UnitCost])
in
    AddProductCostColumn
    
```

- i) Click **OK** to close the **Advanced editor** dialog.



Advanced editor

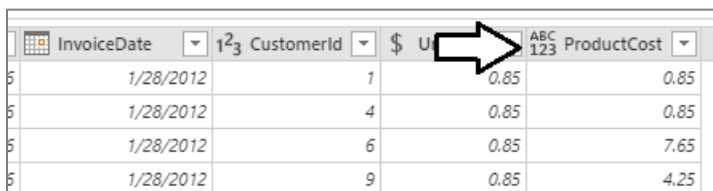
```

1 let
2   Source = Sql.Database("cpt.database.windows.net", "WingtipSalesDB"),
3   #"Navigation 1" = Source[Schema = "dbo", Item = "InvoiceDetails"][Data],
4   #"Changed column type" = Table.TransformColumnTypes(#"Navigation 1", {"SalesAmount", Currency.Type})),
5   #"Expanded Invoices" = Table.ExpandRecordColumn(#"Changed column type", "Invoices", {"InvoiceDate", "CustomerId"}, {"InvoiceDate", "CustomerId"}),
6   #"Changed column type 1" = Table.TransformColumnTypes(#"Expanded Invoices", {"InvoiceDate", type date})),
7   #"Expanded Products" = Table.ExpandRecordColumn(#"Changed column type 1", "Products", {"UnitCost"}, {"UnitCost"}),
8   ChangedUnitCostType = Table.TransformColumnTypes(#"Expanded Products", {"UnitCost", Currency.Type})),
9   AddProductCostColumn = Table.AddColumn(ChangedUnitCostType, "ProductCost", each [Quantity] * [UnitCost])
10 in
11   AddProductCostColumn
    
```

OK Cancel

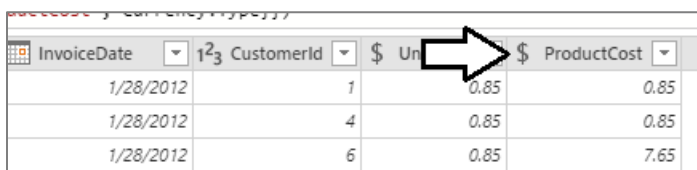
This M code can be copied from the file named **AddProductCostColumn.m.txt** in the **Lab** folder for this module.

- j) You should be able to verify that the new **ProductCost** column has a value which is the product **Quantity** times **UnitCost**.



	InvoiceDate	CustomerId	\$ Unit	ProductCost
5	1/28/2012	1	0.85	0.85
5	1/28/2012	4	0.85	0.85
5	1/28/2012	6	0.85	7.65
5	1/28/2012	9	0.85	4.25

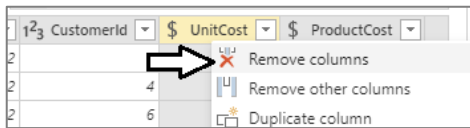
- k) Modify the column type of the **Product Cost** column to the **Currency** type.



	InvoiceDate	CustomerId	\$ Unit	\$ ProductCost
	1/28/2012	1	0.85	0.85
	1/28/2012	4	0.85	0.85
	1/28/2012	6	0.85	7.65

Once the **UnitCost** column has been used to calculate the **ProductCost** value, the column is no longer needed and can be removed.

12. Remove the **UnitCost** column by right-clicking its column header and selecting the **Remove columns** command.



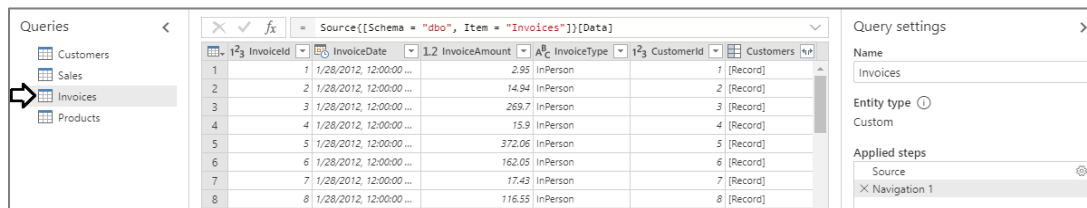
13. Now you should see the **ProductCost** column in the query output, but not the **UnitCost** column.

InvoiceDate	Customerid	ProductCost
1/28/2012	1	0.85
1/28/2012	4	0.85
1/28/2012	6	7.65
1/28/2012	9	4.35

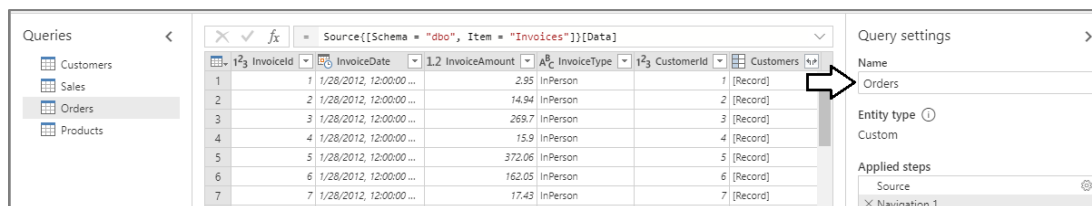
You are now done working with the **Sales** query.

14. Rename the **Invoices** query to **Orders**.

a) Select the **Invoices** query from the **Queries** list on the left.

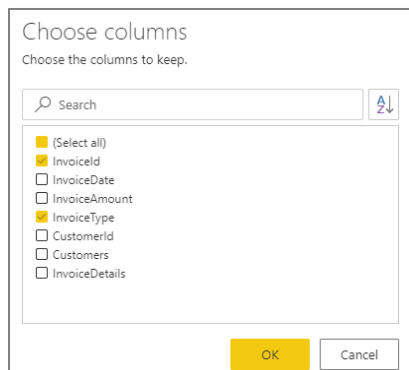


b) Update the name of the **Invoices** query to **Orders** by replacing the text in the **Name** textbox in the **Query Settings** pane.



15. Remove the unneeded columns from the **Orders** query.

- Click the **Choose columns** button to display the **Choose columns** dialog.
- In the **Choose columns** dialog, begin by clicking on the **(Select all)** checkbox at the top to unselect all columns.
- Select the checkboxes for **Invoiceld** and **InvoiceType** as shown in the following screenshot and then click **OK**.



- d) You should be able to see that the **Orders** query now only shows the columns that you selected.

Queries		fx = Table.SelectColumns	
Customers		1 InvoiceId	A InvoiceType
Sales		1 1	1 InPerson
Orders		2 2	2 InPerson
Products		3 3	3 InPerson
		4 4	4 InPerson
		5 5	5 InPerson
		6 6	6 InPerson

16. Modify the query so that the **InvoiceType** column returns values that are more human-readable.

- a) Right-click the header for the **InvoiceType** column and select **Replace values**.

1 2 3 InvoiceId	A B C InvoiceType
1	1 InPerson
2	2 InPerson
3	3 InPerson
4	4 InPerson
5	5 InPerson
6	6 InPerson
7	7 InPerson
8	8 InPerson
9	9 InPerson

- b) In the **Replace Values** dialog, enter a value of **InPerson** in the **Value to find** textbox.  
c) Enter a value of **Store Purchases** in the **Replace with** textbox and click **OK** to add your changes to the underlying query.

Replace values

Replace one value with another in the selected columns.

Basic ☒ Advanced ☐

Value to find  
InPerson

Replace with  
Store Purchases

OK Cancel

- d) Add a second **Replace values** step to replace **MailOrder** with a value of **Mail Order Purchases**.

Replace values

Replace one value with another in the selected columns.

Basic ☒ Advanced ☐

Value to find  
MailOrder

Replace with  
Mail Order Purchases

OK Cancel

- e) Add a third **Replace values** step to replace **Online** with a value of **Online Purchases**.

Replace values

Replace one value with another in the selected columns.

Basic ☒ Advanced ☐

Value to find  
Online

Replace with  
Online Purchases

OK Cancel

If you scroll down and look at all the rows within the **Orders** query results, you should be able to see that each row has an **InvoiceType** column value of either **Store Purchases**, **Mail Order Purchases** or **Online Purchases**.

- f) Change the name of the **InvoiceType** column to **Order Type**.

	1 <sup>2</sup> InvoiceId	A <sup>B</sup> Order Type
1	1	Store Purchases
2	2	Store Purchases
3	3	Store Purchases

The next step is only necessary because of a bug in the November 2019 edition of Power BI Desktop. The bug involves Power BI Desktop not being able to import a dataflow entity if it has one or more columns based in the **Date** datatype. While the bug prevents importing dataflow entities with columns based on the **Date** datatype, it does support columns based on the **Date/Time** datatype. Therefore, you will update the columns currently based on the **Date** datatype to use the **Date/Time** datatype.

17. Update all **Date** columns in the **Wingtip Sales Data** dataflow to the datatype of **Date/Time**.

- a) Select the **Customers** query in the **Queries** list and locate the **BirthDate** column.

Queries		Table.TransformColumnTypes(#"Removed columns", {"Customer Type", type text})
Customers		
Sales		
Orders		

1 <sup>2</sup> CustomerId	A <sup>B</sup> Customer	A <sup>B</sup> City	A <sup>B</sup> State	A <sup>B</sup> Zipcode	A <sup>B</sup> Gender	BirthDate	A <sup>B</sup> Customer Type
1	Nina Diaz	Eureka	CA	95501	Female	4/11/1966	One-time Customer
2	Melinda Carter	Napa	CA	94558	Female	6/6/1976	One-time Customer

- b) Update the datatype of the **BirthDate** column to **Date/Time**.

A <sup>B</sup> Zipcode	A <sup>B</sup> Gender	1 <sup>2</sup> Decimal number
95501	Female	4/1 \$ Currency
94558	Female	6/6 1 <sup>2</sup> Whole number
94558	Female	9/8 % Percentage
95823	Female	8/1 Date/Time
95818	Male	

- c) Verify the **BirthDate** column is now configured with the **Date/Time** datatype.

A <sup>B</sup> Zipcode	A <sup>B</sup> Gender	BirthDate
95501	Female	4/11/1966, 12:00:00...
94558	Female	6/6/1976, 12:00:00...

- d) Select the **Sales** query in the **Queries** list and locate the **InvoiceDate** column.

Queries		Table.RemoveColumns(#"Changed column type 2", {"UnitCost"})
Customers		
Sales		
Orders		

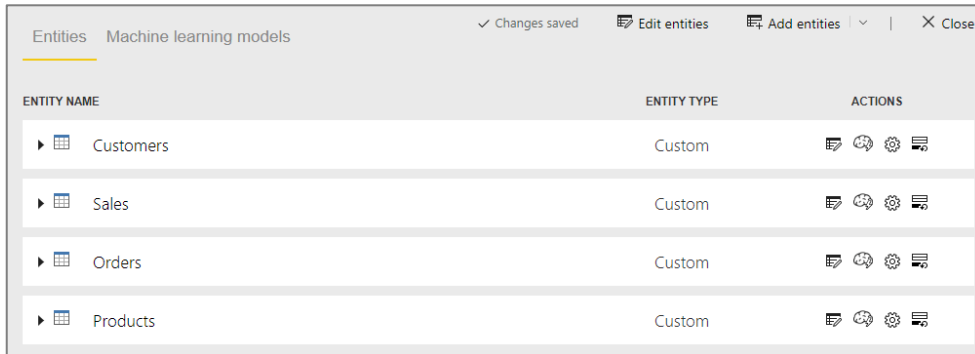
1 <sup>2</sup> Id	1 <sup>2</sup> Quantity	\$ SalesAmount	1 <sup>2</sup> InvoiceId	1 <sup>2</sup> ProductId	InvoiceDate	1 <sup>2</sup> CustomerId
1	1	2.95	1	16	1/28/2012	
2	5	2.95	4	16	1/28/2012	

- e) Update the datatype of the **InvoiceDate** column to **Date/Time**.

\$ SalesAmount	1 <sup>2</sup> InvoiceId	1 <sup>2</sup> ProductId	InvoiceDate
2.95	1	16	1/28/2012, 12:00:00...
2.95	4	16	1/28/2012, 12:00:00...

18. Save your work on the **Wingtip Sales Data** workflow.

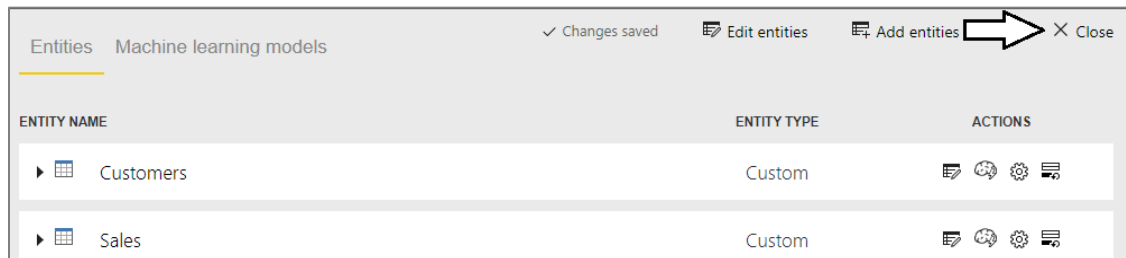
- Click the **Save & close** button at the bottom right of the page to save your work and return to the dataflow summary page.
- The **Entities** list should now display four entities named **Customers**, **Sales**, **Orders** and **Products**.



ENTITY NAME	ENTITY TYPE	ACTIONS
Customers	Custom	[Icons]
Sales	Custom	[Icons]
Orders	Custom	[Icons]
Products	Custom	[Icons]

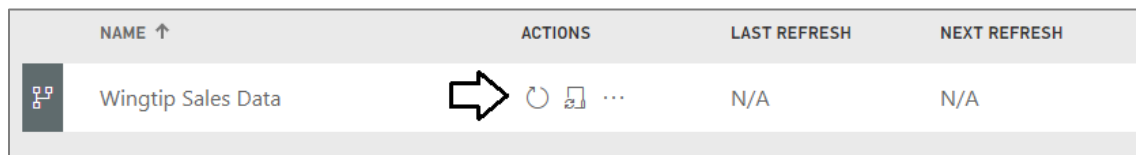
19. Refresh the **Wingtip Sales Data** dataflow to populate it with data.

- Click the **Close** button in the upper right corner of the dataflow summary page.



ENTITY NAME	ENTITY TYPE	ACTIONS
Customers	Custom	[Icons]
Sales	Custom	[Icons]

- You should now see the **Wingtip Sales Data** dataflow in the app workspace summary page.
- Click the Refresh button to begin a refresh operation on the **Wingtip Sales Data** dataflow.

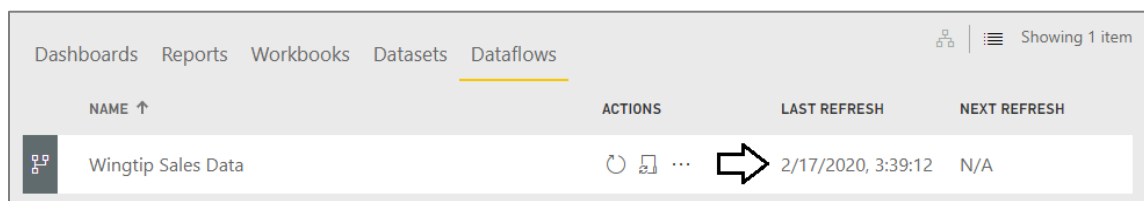


NAME ↑	ACTIONS	LAST REFRESH	NEXT REFRESH
Wingtip Sales Data	[Icons]	N/A	N/A

- Wait for the refresh operation to complete. It might take one or two minutes to complete.



- Once the refresh operation completes, you should see the **LAST REFRESH** time has been updated.



NAME ↑	ACTIONS	LAST REFRESH	NEXT REFRESH
Wingtip Sales Data	[Icons]	2/17/2020, 3:39:12	N/A

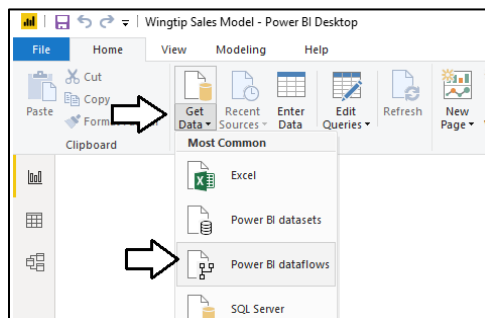
### Exercise 3: Importing Dataflow Entity Data with Power BI Desktop

In the following exercise, you will create a new Power BI Desktop project named **Wingtip Sales Model.pbix** into which you will import the entities from the **Wingtip Sales Data** dataflow.

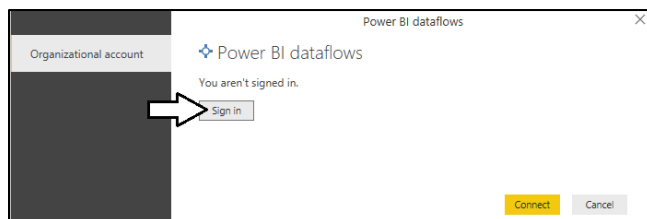
1. Launch Power BI Desktop to start a new project.
2. Save the new project as **Wingtip Sales Model.pbix** using the following path.

**C:\Student\Projects\wingtip Sales Model.pbix**

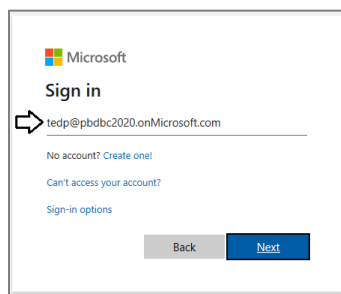
3. Import all four entities from the **Wingtip sales Data** dataflow.
  - a) Drop down the **Get Data** menu button on the ribbon and click **Power BI dataflows**.



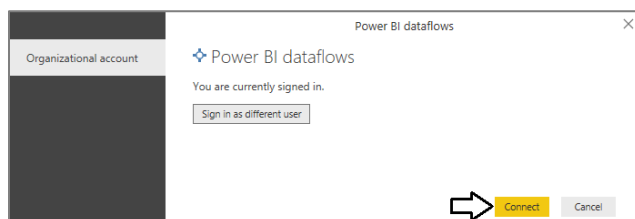
- b) When prompted by the **Power BI dataflows** connect dialog, click **Sign in**.



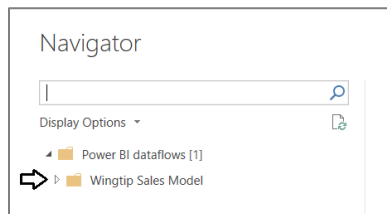
- c) In the **Sign in** dialog, enter your account name and then click **Next**.



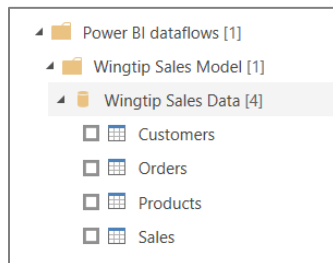
- d) Once you have signed in, click the **Connect** button.



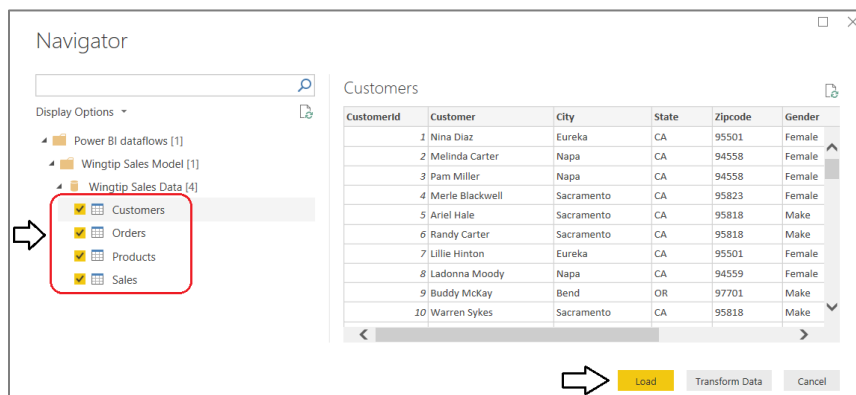
- e) In the **Navigator** dialog, expand the **Wingtip Sales Model** workspace



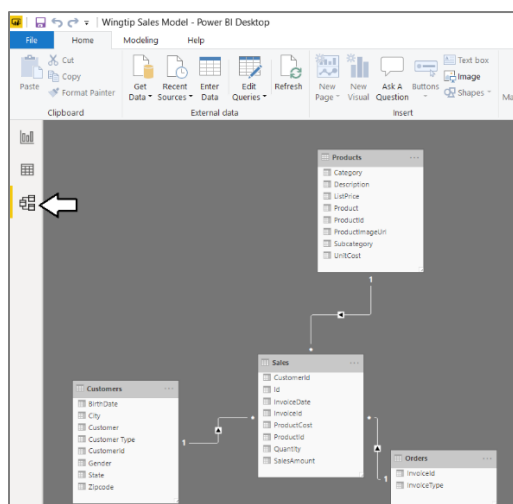
- f) Expand the **Wingtip Sales Data** workflow so you can see the entities inside this dataflow.



- g) Select all four of the entities in the **Wingtip Sales Data** workflow and then click the **Load** button.

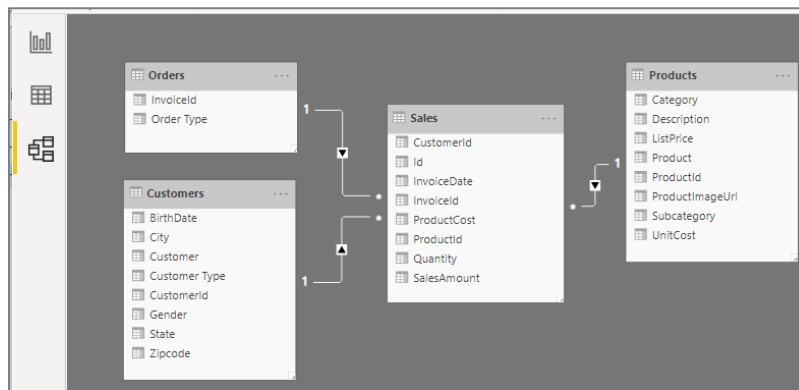


- h) Once the queries execute, click the **Model View** button to see the four tables imported from the dataflow.





- i) Using the mouse, rearrange the four tables in Model View to match the following screenshot.

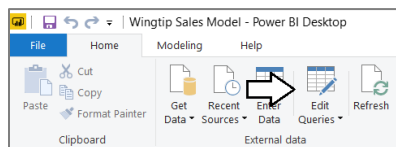


- j) Save your work to **Wingtip Sales Model.pbix**.

The final steps will provide a work around for the Power BI Desktop bug in the November 2019 edition which prevents importing **Date** columns from a dataflow. You will update the two columns named **BirthDate** and **InvoiceDate** back to **Date** columns.

4. Update the **Date/Time** columns in the **Wingtip Sales Data** dataflow to the datatype of **Date**.

- a) Click the **Edit Queries** button on the **Home** tab of the ribbon to open the Power Query Editor window.



- b) Select the **Customers** query in the **Queries** list and locate the **BirthDate** column.

The screenshot shows the Power Query Editor with the 'Customers' query selected in the 'Queries' list. The 'BirthDate' column is highlighted in the table view.

	AB_C Gender	BirthDate	AB_C Customer Type
1	Female	4/11/1966 12:00:00 AM	One-time Customer
2	Female	6/6/1976 12:00:00 AM	One-time Customer

- c) Update the datatype of the **BirthDate** column to **Date**.

The screenshot shows the Power Query Editor with the 'Customers' query selected. The 'BirthDate' column is highlighted, and the datatype dropdown menu is open, showing 'Date' as the selected option.

	AB_C Gender	BirthDate	AB_C Customer Type
1	Female	4/11/1966 12:00:00 AM	One-time Customer
2	Female	6/6/1976 12:00:00 AM	One-time Customer
3	Female		
4	Female		
5	Male		
6	Male		

- d) Select the **Sales** query in the **Queries** list and locate the **InvoiceDate** column.

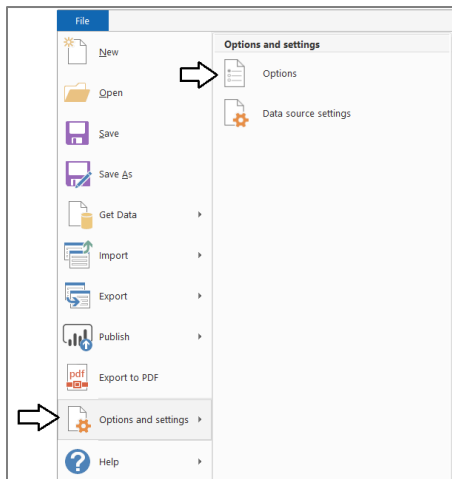
- e) Update the datatype of the **InvoiceDate** column to **Date**.

The screenshot shows the Power Query Editor with the 'Sales' query selected in the 'Queries' list. The 'InvoiceDate' column is highlighted in the table view.

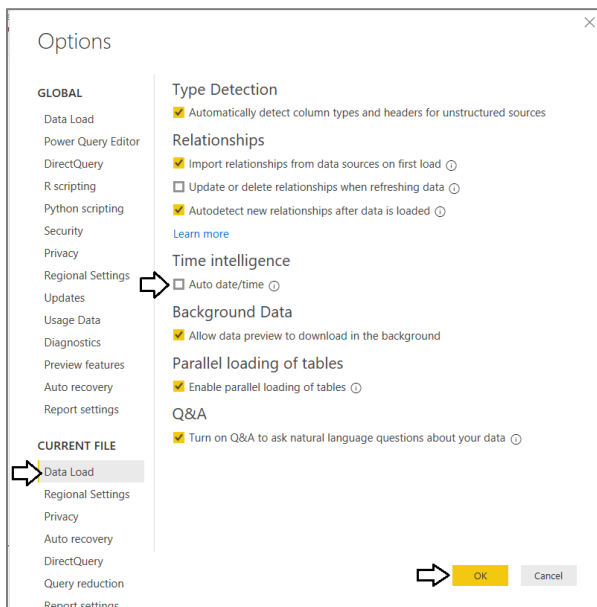
	1^2_3 InvoiceId	1^2_3 ProductId	InvoiceDate
1	35	1	1/28/2012
2	35	4	1/28/2012
3	55	6	1/28/2012
4	75	9	1/28/2012

5. Disable the Auto date/time feature the **Wingtip Sales Model.pbix** project.

- a) Drop down the Power BI Desktop **File** menu and select the **Options and settings > Options** menu command.



- b) In the **Options** dialog, click the **Data Load** link in the **CURRENT FILE** section in the left navigation.  
c) Unselect the **Auto date/time** option.  
d) Click the **OK** button to close the **Options** dialog and save your changes.



6. Save all your changes and close **Wingtip Sales Model.pbix**.

You have now completed all the work for this lab. While you haven't yet begun the data modeling phase yet, you have refactored the database tables from **WingtipSalesDB** into a **star schema** which is a best practice in query design for data analysis projects.

**NOTE:** You will continue to work on this PBIX file named **Wingtip Sales Model.pbix** in the next lab and the ones that follow.