Designing Dataflows to Extract and Transform Data

Lab Time: 40 minutes

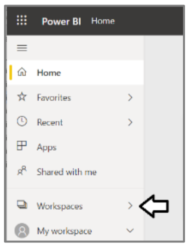
Lab Folder: C:\Student\Modules\10_Dataflows\Lab\

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.

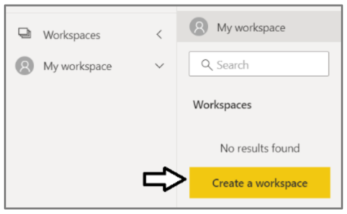
Exercise 1: Use Power Query to Create a New Dataflow

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 an Azure SQL database running in the Microsoft cloud.

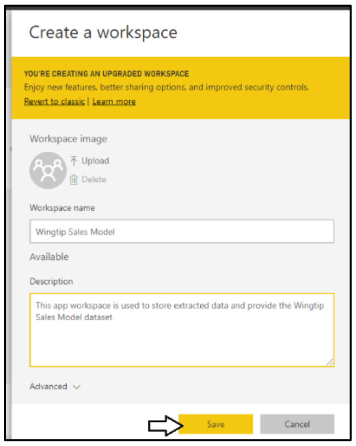
1. Log into the Power BI Service with your new organizational account.
   a) Navigate the Power BI portal at https://app.powerbi.com and if prompted, log in using your organizational account.
2. Create a new app workspace named **Wingtip Sales Model**.
   a) Click the **Workspace** flyout menu in the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



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

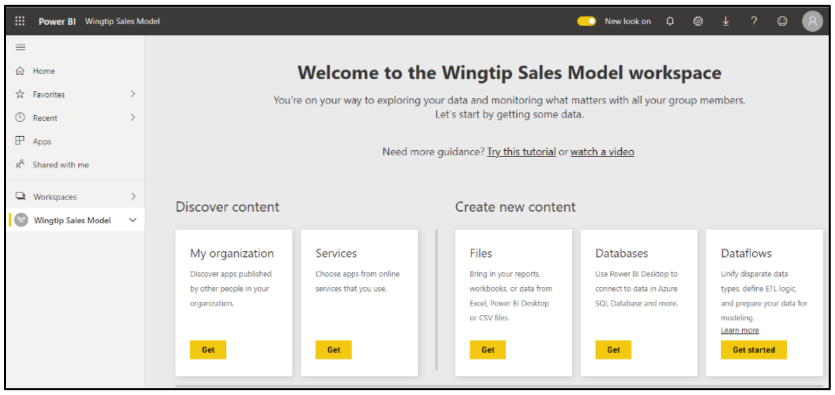


   c) In the **Create an app workspace** pane, enter a workspace name of **Wingtip Sales Model**.
   d) Click the **Save** button to create the new app workspace named **Wingtip Sales Model**.


Create a workspace
YOU'RE CREATING AN UPGRADED WORKSPACE
Enjoy new features, better sharing options, and improved security controls.
Revert to classic | Learn more
Workspace image
Upload
Delete
Advanced
Save
Cancel

Workspace name: Wingtip Sales Model
Available
Description: This app workspace is used to store extracted data and provide the Wingtip Sales Model dataset

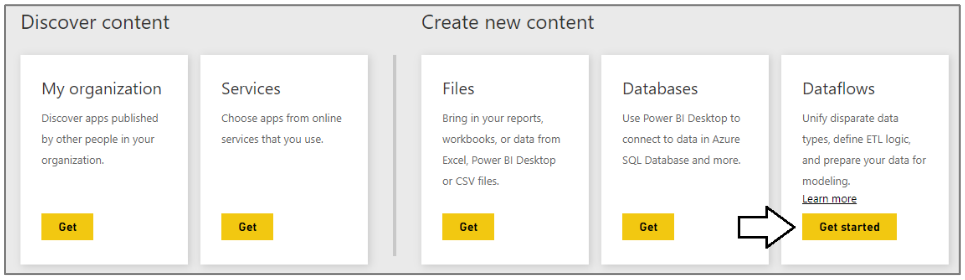
e) When you click **Save**, the Power BI service should create the new app workspace and then switch your current Power BI session to be running within the context of the new **Wingtip Sales Model** app workspace.


Power BI Wingtip Sales Model
New look on
Home
Favorites
Recent
Apps
Shared with me
Workspaces
Wingtip Sales Model
Welcome to the Wingtip Sales Model workspace
You're on your way to exploring your data and monitoring what matters with all your group members. Let's start by getting some data.
Need more guidance? Try this tutorial or watch a video
Discover content
Create new content
My organization
Services
Files
Databases
Dataflows
Get
Get started

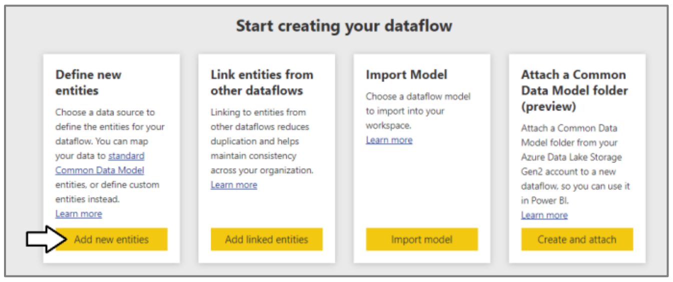
You should now see the welcome page for the app workspace. The welcome page is shown by default when an app workspace is empty and contains no content.

3. Create a new dataflow.

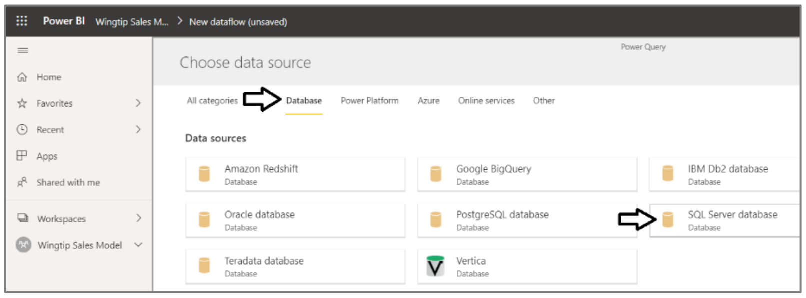
a) Click the **Get started** button in the **Dataflows** section.


Discover content
Create new content
My organization
Discover apps published by other people in your organization.
Services
Choose apps from online services that you use.
Files
Bring in your reports, workbooks, or data from Excel, Power BI Desktop or CSV files.
Databases
Use Power BI Desktop to connect to data in Azure SQL Database and more.
Dataflows
Unify disparate data types, define ETL logic, and prepare your data for modeling.
Learn more
Get
Get started

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

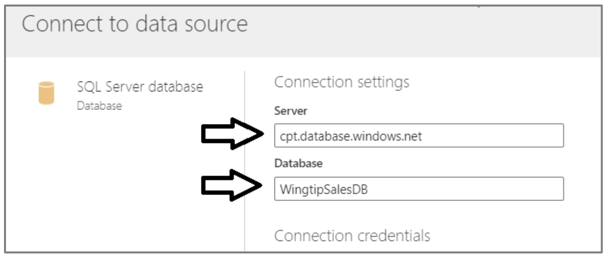

Start creating your dataflow
Define new entities
Choose a data source to define the entities for your dataflow. You can map your data to standard Common Data Model entities, or define custom entities instead.
Learn more
Add new entities
Link entities from other dataflows
Linking to entities from other dataflows reduces duplication and helps maintain consistency across your organization.
Learn more
Add linked entities
Import Model
Choose a dataflow model to import into your workspace.
Learn more
Import model
Attach a Common Data Model folder (preview)
Attach a Common Data Model folder from your Azure Data Lake Storage Gen2 account to a new dataflow, so you can use it in Power BI.
Learn more
Create and attach

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**.



Server: cpt.database.windows.net
Database: WingtipSalesDB

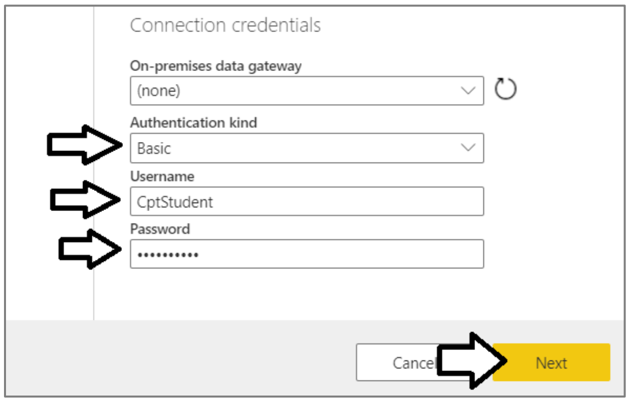
f) Move down the **Connection credentials** section.

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.

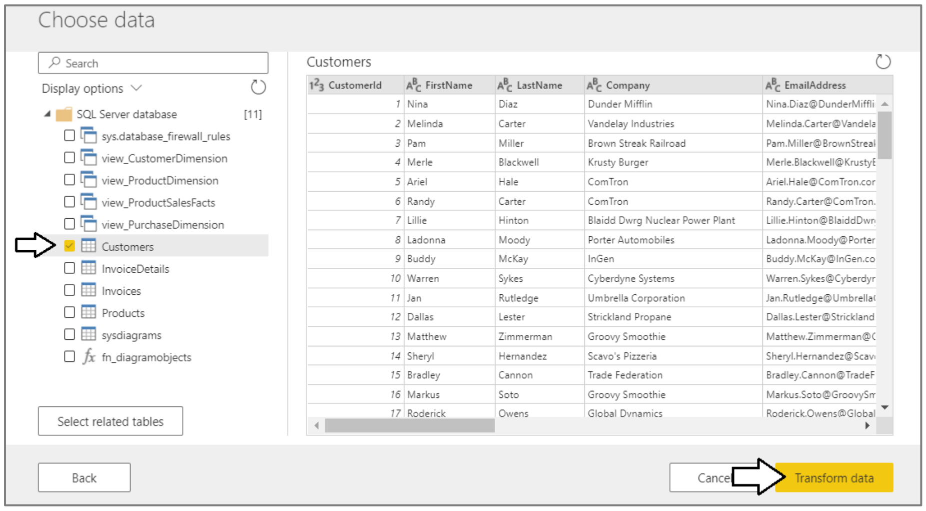


On-premises data gateway: (none)
Authentication kind: Basic
Username: CptStudent
Password: ••••••••••

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

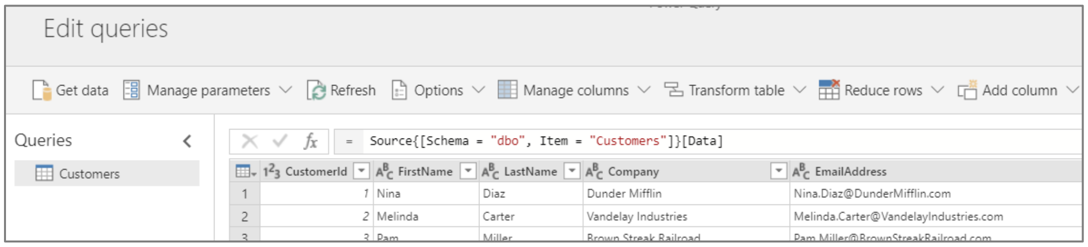
a) In the **Choose data** section, select the **Customers** table and then click **Transform data**.



Customers

CustomerId	FirstName	LastName	Company	EmailAddress
1	Nina	Diaz	Dunder Mifflin	Nina.Diaz@DunderMiffli
2	Melinda	Carter	Vandelay Industries	Melinda.Carter@Vandela
3	Pam	Miller	Brown Streak Railroad	Pam.Miller@BrownStreal
4	Merle	Blackwell	Krusty Burger	Merle.Blackwell@KrustyE
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@BlaiddDwr
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@Strickland
13	Matthew	Zimmerman	Groovy Smoothie	Matthew.Zimmerman@
14	Sheryl	Hernandez	Scavo's Pizzeria	Sheryl.Hernandez@Scav
15	Bradley	Cannon	Trade Federation	Bradley.Cannon@TradeF
16	Markus	Soto	Groovy Smoothie	Markus.Soto@GroovySm
17	Roderick	Owens	Global Dynamics	Roderick.Owens@Global

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

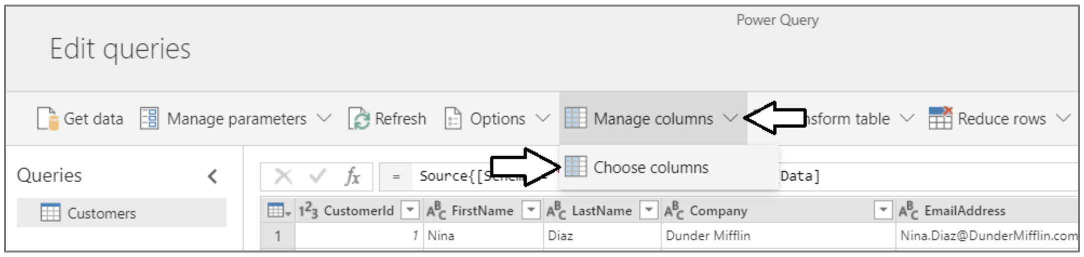


```
= Source{[Schema = "dbo", Item = "Customers"]}[Data]
```

	CustomerId	FirstName	LastName	Company	EmailAddress
1	1	Nina	Diaz	Dunder Mifflin	Nina.Diaz@DunderMifflin.com
2	2	Melinda	Carter	Vandelay Industries	Melinda.Carter@VandelayIndustries.com
3	3	Pam	Miller	Brown Streak Railroad	Pam.Miller@BrownStreakRailroad.com

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

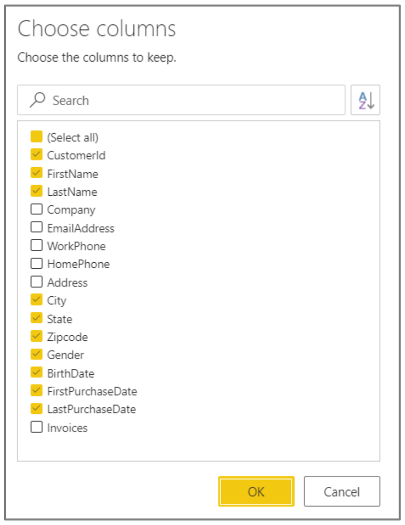
a) Drop down the **Manage columns** menu button and select the **Choose columns** command.



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.

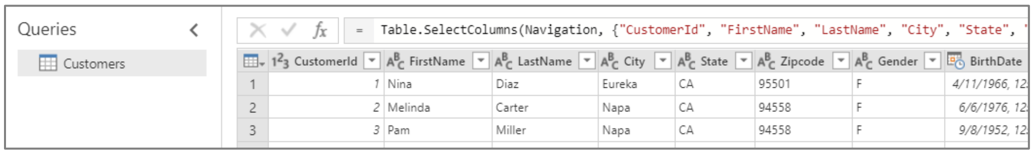
i) **CustomerId**
ii) **FirstName**
iii) **LastName**
iv) **City**
v) **State**
vi) **Zipcode**
vii) **Gender**
viii) **BirthDate**
ix) **FirstPurchaseDate**
x) **LastPurchaseDate**

c) Once you have the columns selected as shown in the following screenshot, click **OK** to close the **Choose columns** dialog.



- ■ (Select all)
- ✓ CustomerId
- ✓ FirstName
- ✓ LastName
- ☐ Company
- ☐ EmailAddress
- ☐ WorkPhone
- ☐ HomePhone
- ☐ Address
- ✓ City
- ✓ State
- ✓ Zipcode
- ✓ Gender
- ✓ BirthDate
- ✓ FirstPurchaseDate
- ✓ LastPurchaseDate
- ☐ Invoices

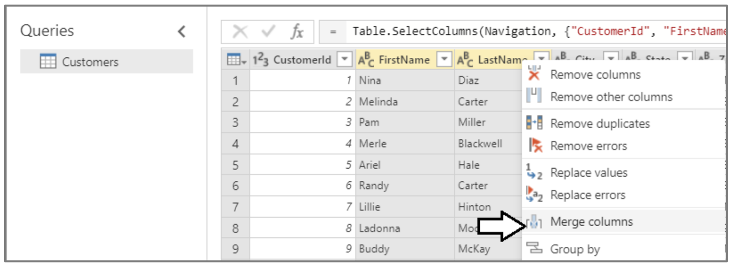
d) You should be able to verify that the Power Query editor now only shows the columns that you selected.



```
= Table.SelectColumns(Navigation, {"CustomerId", "FirstName", "LastName", "City", "State", '
```

	CustomerId	FirstName	LastName	City	State	Zipcode	Gender	BirthDate
1	1	Nina	Diaz	Eureka	CA	95501	F	4/11/1966, 12
2	2	Melinda	Carter	Napa	CA	94558	F	6/6/1976, 12
3	3	Pam	Miller	Napa	CA	94558	F	9/8/1952, 12

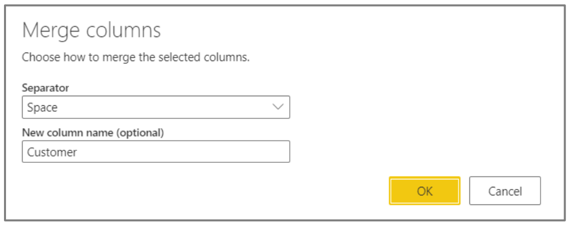
6. In this step you will merge the **FirstName** column and the **LastName** column together into a single column named **Customer**.
   a) Select the **FirstName** column by clicking on its column header.
   b) Next, hold down the **SHIFT** key and select the **LastName** column by clicking on its column header.
   c) Right-click on the selected columns and click the **Merge columns** menu command.



```
= Table.SelectColumns(Navigation, {"CustomerId", "FirstName
```

	CustomerId	FirstName	LastName
1	1	Nina	Diaz
2	2	Melinda	Carter
3	3	Pam	Miller
4	4	Merle	Blackwell
5	5	Ariel	Hale
6	6	Randy	Carter
7	7	Lillie	Hinton
8	8	Ladonna	Mo
9	9	Buddy	McKay

d) 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.

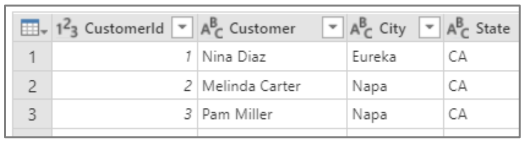

Merge columns

Choose how to merge the selected columns.

Separator: Space

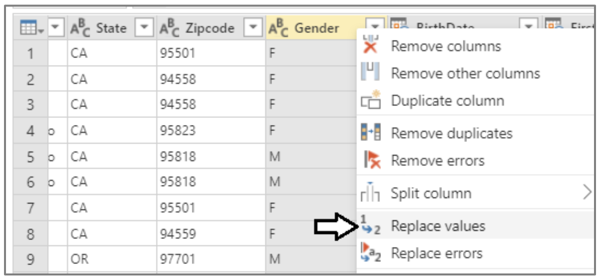
New column name (optional): Customer

e) 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**.



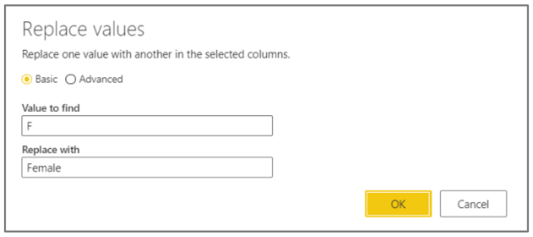
	CustomerId	Customer	City	State
1	1	Nina Diaz	Eureka	CA
2	2	Melinda Carter	Napa	CA
3	3	Pam Miller	Napa	CA

7. Modify the query so that the **Gender** column returns values of **Male** and **Female** instead of **M** and **F**.
   a) Locate the **Gender** column in the **Customers** table.
   b) Right-click the header for the **Gender** column and select the **Replace values** command to display the **Replace values** dialog.



	State	Zipcode	Gender
1	CA	95501	F
2	CA	94558	F
3	CA	94558	F
4	CA	95823	F
5	CA	95818	M
6	CA	95818	M
7	CA	95501	F
8	CA	94559	F
9	OR	97701	M

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

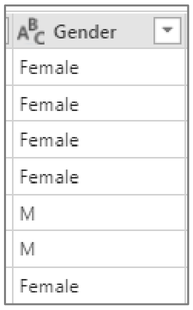
Replace one value with another in the selected columns.

[✓ Basic] Advanced

Value to find: F

Replace with: Female

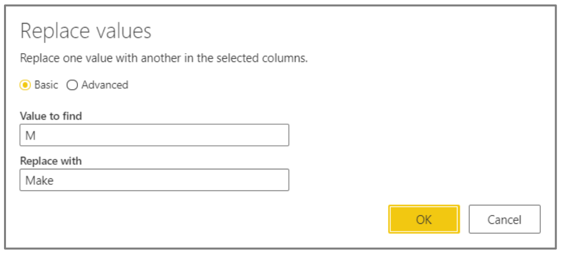
d) You should be able to see that all values of **F** in the **Gender** column have been replaced with a value of **Female**.



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

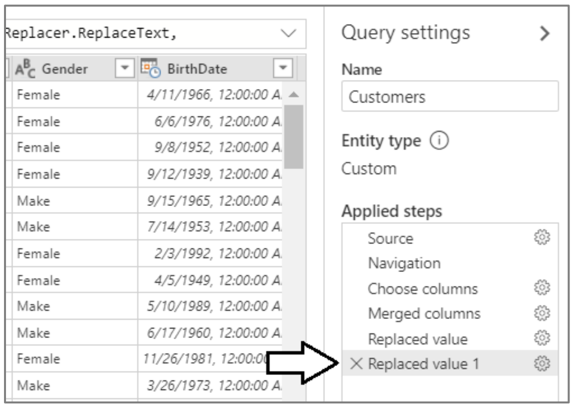
Value to find: M

Replace with: Make

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
Make	9/15/1965, 12:00:00 A
Make	7/14/1953, 12:00:00 A
Female	2/3/1992, 12:00:00 A
Female	4/5/1949, 12:00:00 A
Make	5/10/1989, 12:00:00 A
Make	6/17/1960, 12:00:00 A
Female	11/26/1981, 12:00:0
Make	3/26/1973, 12:00:00 A

Query settings

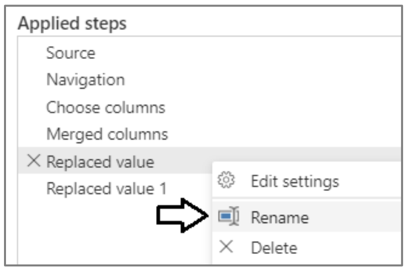
Name: Customers

Entity type: Custom

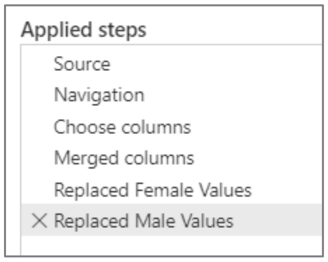
Applied steps
- Source
- Navigation
- Choose columns
- Merged columns
- Replaced value
- Replaced value 1

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**.



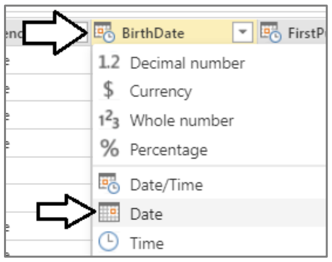
c) Using the same technique, rename the **Replaced Value 1** step to **Replaced Male Values**.



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**.

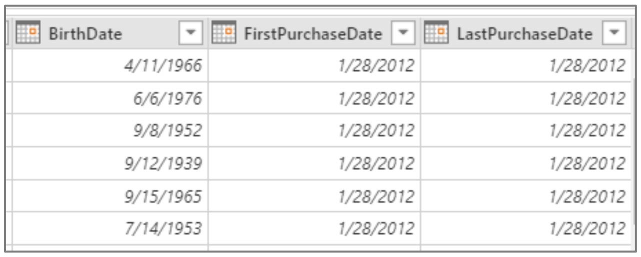
a) Use the column type drop down on the left-hand side of the **BirthDate** column to configure the column using the **Date** type.



b) Use the column type drop down of the **FirstPurchaseDate** column to configure the column using the **Date** type.

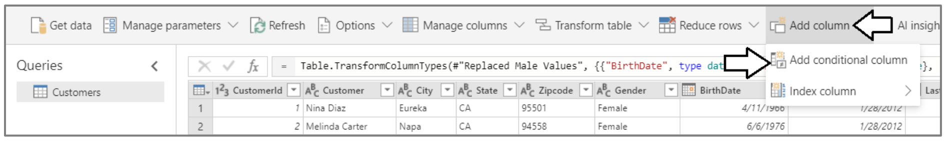
c) Use the column type drop down of the **LastPurchaseDate** column to configure the column using the **Date** type.

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.

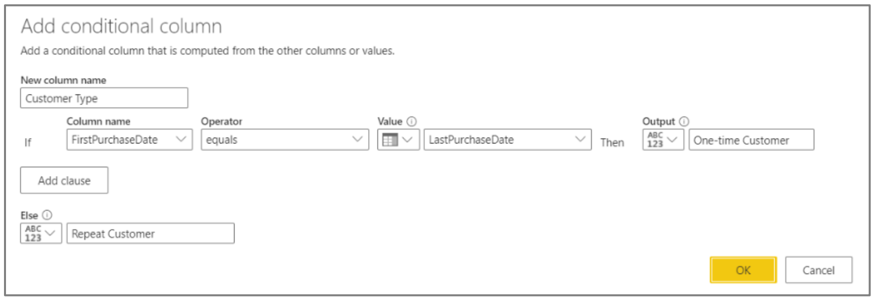


```
= Table.TransformColumnTypes(#"Replaced Male Values", {{"BirthDate", type dat
```

	CustomerId	Customer	City	State	Zipcode	Gender	BirthDate	
1	1	Nina Diaz	Eureka	CA	95501	Female	4/11/1966	1/28/2012
2	2	Melinda Carter	Napa	CA	94558	Female	6/6/1976	1/28/2012

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.

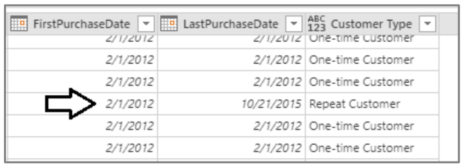
   b) In the **Add conditional column** dialog, enter a **New column name** value of **Customer Type**.
   c) Configure a rule to return a string value of “One-time Customer” if **FirstPurchaseDate** equals **LastPurchaseDate**.
   d) For the **Else** evaluation, return a string value of “Repeat Customer”.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



Add conditional column
Add a conditional column that is computed from the other columns or values.
New column name: Customer Type
If | Column name: FirstPurchaseDate | Operator: equals | Value: LastPurchaseDate | Then | Output: One-time Customer
Else: Repeat Customer

   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 **CustomerType** column has a value of **One-time Customer**.

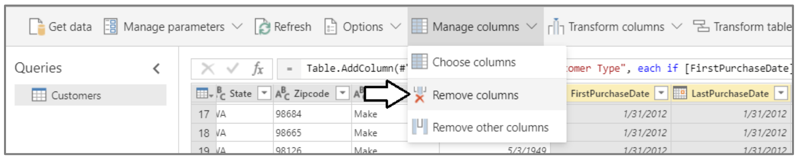
You might have to scroll down several pages of records in the **Customers** query results before you begin to see repeat customers.



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 have used 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**.



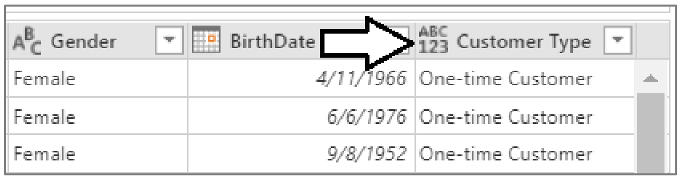
```
= Table.AddColumn(#"...omer Type", each if [FirstPurchaseDate]
```

	State	Zipcode			FirstPurchaseDate	LastPurchaseDate
17	VA	98684	Make		1/31/2012	1/31/2012
18	VA	98665	Make		1/31/2012	1/31/2012
19	VA	98126	Make	5/3/1949	1/31/2012	1/31/2012

   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.

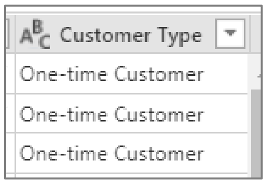
12. Set the column type for the **Customer Type** column to **Text**.

a) You might notice that column type menu for the **Customer Type** column is not set to a specific type. When you see the type as ABC above and 123 below, that the column is being assigned the generic **Any** type.



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

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

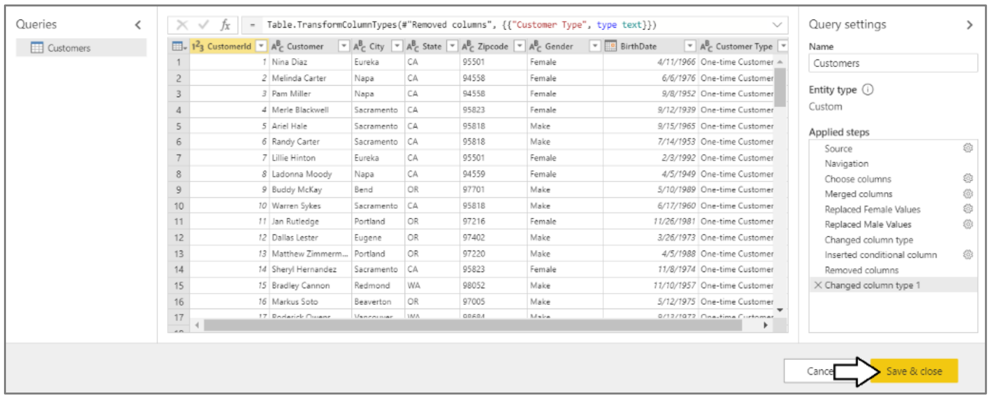


Customer Type
One-time Customer
One-time Customer
One-time Customer

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.



```
= Table.TransformColumnTypes(#"Removed columns", {{"Customer Type", type text}})
```

	CustomerId	Customer	City	State	Zipcode	Gender	BirthDate	Customer Type
1	1	Nina Diaz	Eureka	CA	95501	Female	4/11/1966	One-time Customer
2	2	Melinda Carter	Napa	CA	94558	Female	6/6/1976	One-time Customer
3	3	Pam Miller	Napa	CA	94558	Female	9/8/1952	One-time Customer
4	4	Merle Blackwell	Sacramento	CA	95823	Female	9/12/1939	One-time Customer
5	5	Ariel Hale	Sacramento	CA	95818	Make	9/15/1965	One-time Customer
6	6	Randy Carter	Sacramento	CA	95818	Make	7/14/1953	One-time Customer
7	7	Lillie Hinton	Eureka	CA	95501	Female	2/3/1992	One-time Customer
8	8	Ladonna Moody	Napa	CA	94559	Female	4/5/1949	One-time Customer
9	9	Buddy McKay	Bend	OR	97701	Make	5/10/1989	One-time Customer
10	10	Warren Sykes	Sacramento	CA	95818	Make	6/17/1960	One-time Customer
11	11	Jan Rutledge	Portland	OR	97216	Female	11/26/1981	One-time Customer
12	12	Dallas Lester	Eugene	OR	97402	Make	3/26/1973	One-time Customer
13	13	Matthew Zimmerm...	Portland	OR	97220	Make	4/5/1988	One-time Customer
14	14	Sheryl Hernandez	Sacramento	CA	95823	Female	11/8/1974	One-time Customer
15	15	Bradley Cannon	Redmond	WA	98052	Make	11/10/1957	One-time Customer
16	16	Markus Soto	Beaverton	OR	97005	Make	5/12/1975	One-time Customer

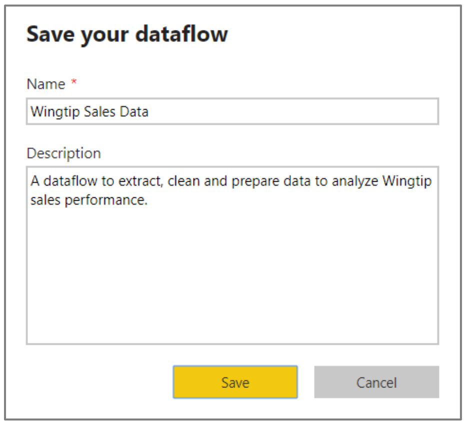
Query settings

Name: Customers

Entity type: Custom

Applied steps:
- Source
- Navigation
- Choose columns
- Merged columns
- Replaced Female Values
- Replaced Male Values
- Changed column type
- Inserted conditional column
- Removed columns
- Changed column type 1

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

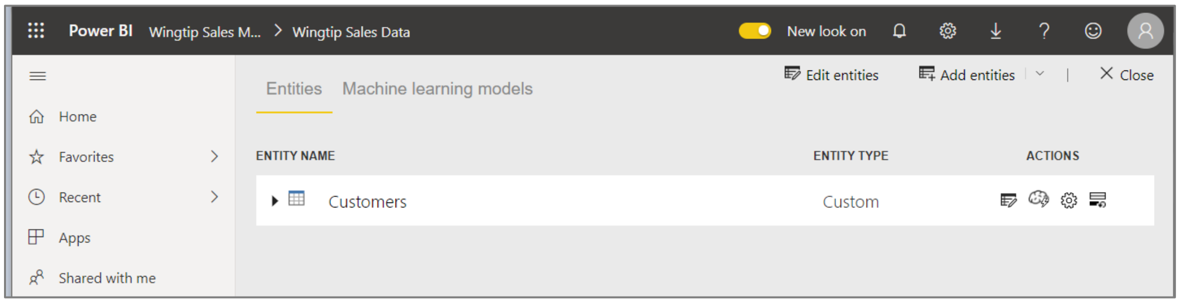


Save your dataflow

Name *: Wingtip Sales Data

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

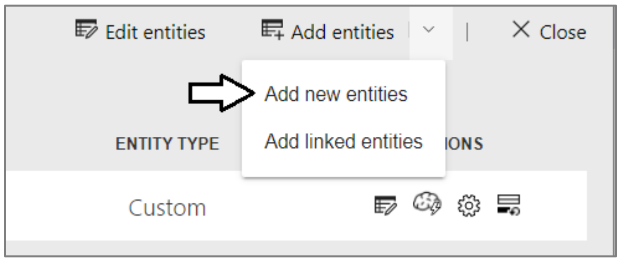
c) You should now see the summary page for the **Wingtip Sales Data** dataflow.



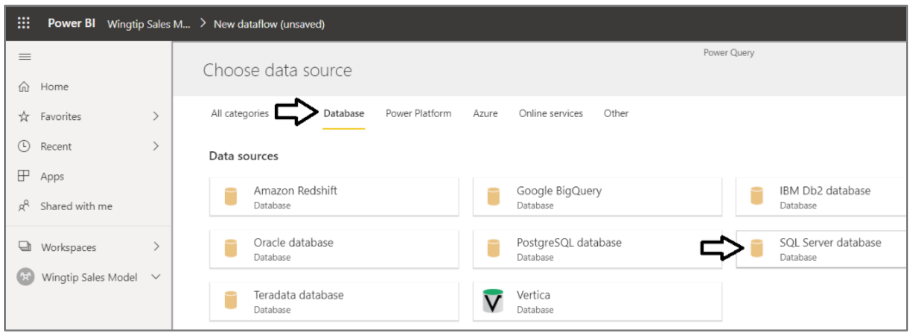
Exercise 2: Extend the Dataflow by Adding Entities for Products and Sales

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

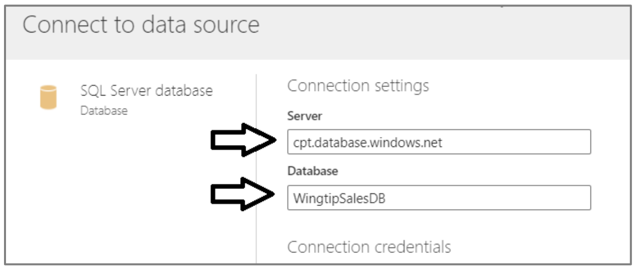
1. Add three new entities to the **Wingtip Sales Data** dataflow.
   a) Locate the **Add entities** menu button in the top right corner of the **Wingtip Sales Data** dataflow summary page.
   b) Drop down the **Add entities** menu button and select the **Add new entities** command.



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



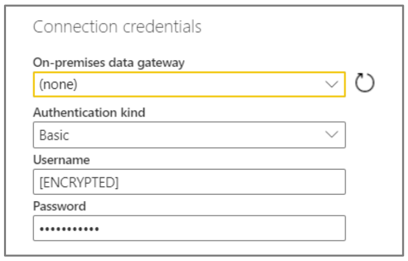
   d) Enter a **Server** value of **cpt.database.windows.net**.
   e) Enter a **Database** value of **WingtipSalesDB**.



Server: cpt.database.windows.net
Database: 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 in an encrypted fashion.

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



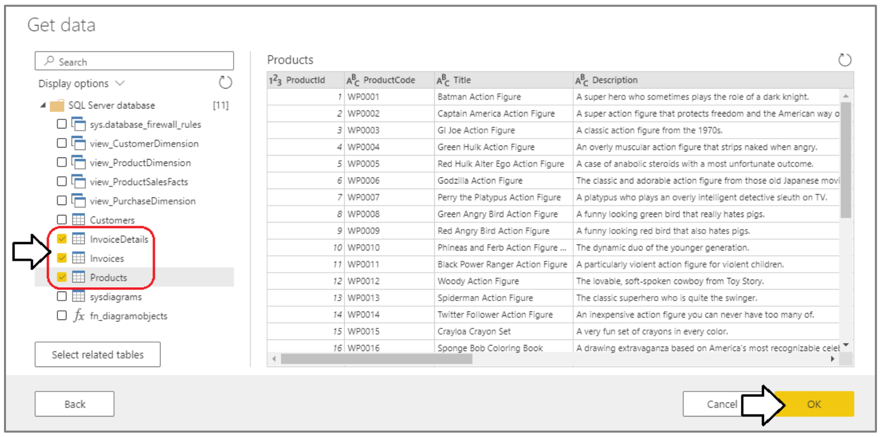
Connection credentials

On-premises data gateway	(none)
Authentication kind	Basic
Username	[ENCRYPTED]
Password	•••••••••••

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.



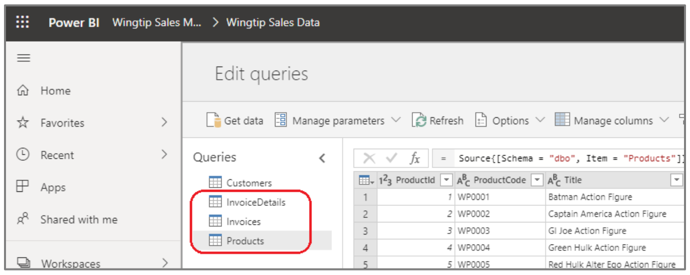
Get data

SQL Server database [11]
- ☐ sys.database_firewall_rules
- ☐ view_CustomerDimension
- ☐ view_ProductDimension
- ☐ view_ProductSalesFacts
- ☐ view_PurchaseDimension
- ☐ Customers
- ☑ InvoiceDetails
- ☑ Invoices
- ☑ Products
- ☐ sysdiagrams
- ☐ fn_diagramobjects

Products

ProductId	ProductCode	Title	Description
1	WP0001	Batman Action Figure	A super hero who sometimes plays the role of a dark knight.
2	WP0002	Captain America Action Figure	A super action figure that protects freedom and the American way o
3	WP0003	GI Joe Action Figure	A classic action figure from the 1970s.
4	WP0004	Green Hulk Action Figure	An overly muscular action figure that strips naked when angry.
5	WP0005	Red Hulk Alter Ego Action Figure	A case of anabolic steroids with a most unfortunate outcome.
6	WP0006	Godzilla Action Figure	The classic and adorable action figure from those old Japanese movi
7	WP0007	Perry the Platypus Action Figure	A platypus who plays an overly intelligent detective sleuth on TV.
8	WP0008	Green Angry Bird Action Figure	A funny looking green bird that really hates pigs.
9	WP0009	Red Angry Bird Action Figure	A funny looking red bird that also hates pigs.
10	WP0010	Phineas and Ferb Action Figure ...	The dynamic duo of the younger generation.
11	WP0011	Black Power Ranger Action Figure	A particularly violent action figure for violent children.
12	WP0012	Woody Action Figure	The lovable, soft-spoken cowboy from Toy Story.
13	WP0013	Spiderman Action Figure	The classic superhero who is quite the swinger.
14	WP0014	Twitter Follower Action Figure	An inexpensive action figure you can never have too many of.
15	WP0015	Crayloa Crayon Set	A very fun set of crayons in every color.
16	WP0016	Sponge Bob Coloring Book	A drawing extravaganza based on America's most recognizable celel

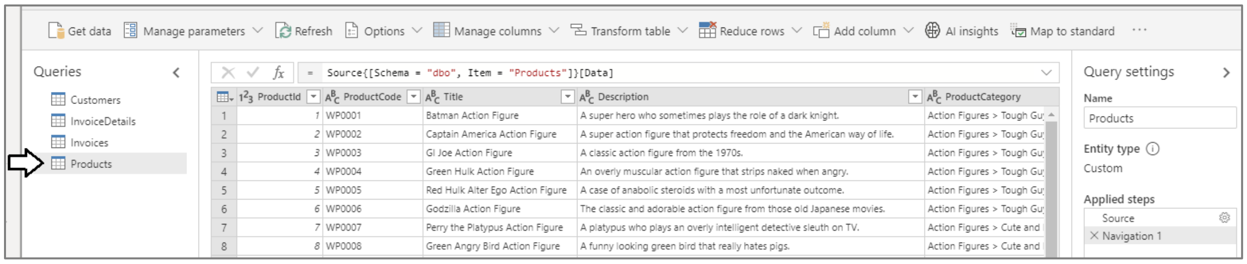
j) You should now see three new queries on the **Edit queries** page named **InvoiceDetails**, **Invoices** and **Products**.



```
= Source{[Schema = "dbo", Item = "Products"]}
```

	ProductId	ProductCode	Title
1	1	WP0001	Batman Action Figure
2	2	WP0002	Captain America Action Figure
3	3	WP0003	GI Joe Action Figure
4	4	WP0004	Green Hulk Action Figure
5	5	WP0005	Red Hulk Alter Ego Action Figure

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

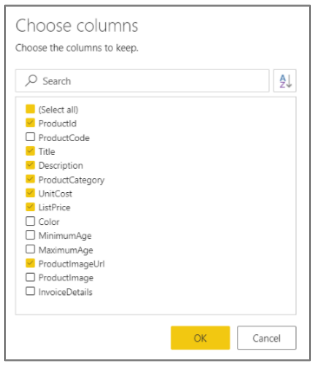


```
= Source{[Schema = "dbo", Item = "Products"]}[Data]
```

	ProductId	ProductCode	Title	Description	ProductCategory
1	1	WP0001	Batman Action Figure	A super hero who sometimes plays the role of a dark knight.	Action Figures > Tough Gu
2	2	WP0002	Captain America Action Figure	A super action figure that protects freedom and the American way of life.	Action Figures > Tough Gu
3	3	WP0003	GI Joe Action Figure	A classic action figure from the 1970s.	Action Figures > Tough Gu
4	4	WP0004	Green Hulk Action Figure	An overly muscular action figure that strips naked when angry.	Action Figures > Tough Gu
5	5	WP0005	Red Hulk Alter Ego Action Figure	A case of anabolic steroids with a most unfortunate outcome.	Action Figures > Tough Gu
6	6	WP0006	Godzilla Action Figure	The classic and adorable action figure from those old Japanese movies.	Action Figures > Tough Gu
7	7	WP0007	Perry the Platypus Action Figure	A platypus who plays an overly intelligent detective sleuth on TV.	Action Figures > Cute and
8	8	WP0008	Green Angry Bird Action Figure	A funny looking green bird that really hates pigs.	Action Figures > Cute and

Name	Products
Entity type	Custom
Applied steps	Source; Navigation 1

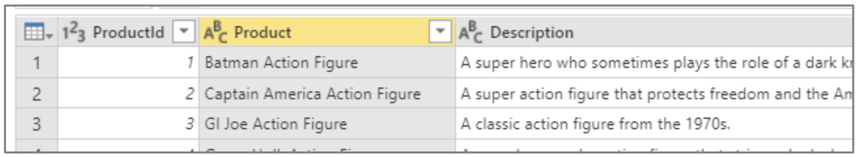
2. Remove the columns that are not required in the **Products** query results.
   a) Click the **Choose Columns** button in the ribbon to display the **Choose Columns** dialog.
   b) In the **Choose Columns** dialog, begin by clicking on the **(Select all)** checkbox at the top to unselect all columns.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


Choose columns
Choose the columns to keep.
Search

- ☐ (Select all)
- ✓ ProductId
- ☐ ProductCode
- ✓ Title
- ✓ Description
- ✓ ProductCategory
- ✓ UnitCost
- ✓ ListPrice
- ☐ Color
- ☐ MinimumAge
- ☐ MaximumAge
- ✓ ProductImageUrl
- ☐ ProductImage
- ☐ InvoiceDetails

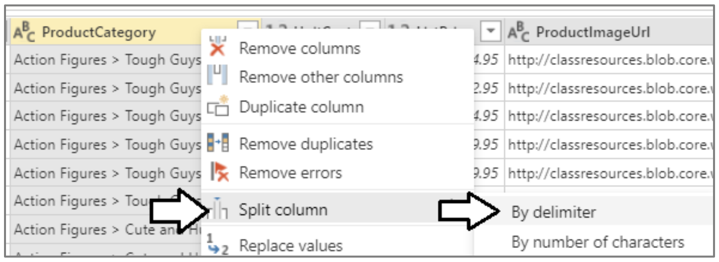
   d) Click the **OK** button to close the **Choose Columns** dialog.

3. Rename the **Title** column to **Product**.
   a) Right-click on the **Title** column and click **Rename**.
   b) Update the column name to **Product**.

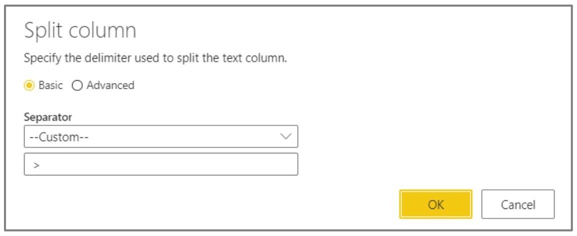


	ProductId	Product	Description
1	1	Batman Action Figure	A super hero who sometimes plays the role of a dark k
2	2	Captain America Action Figure	A super action figure that protects freedom and the Am
3	3	GI Joe Action Figure	A classic action figure from the 1970s.

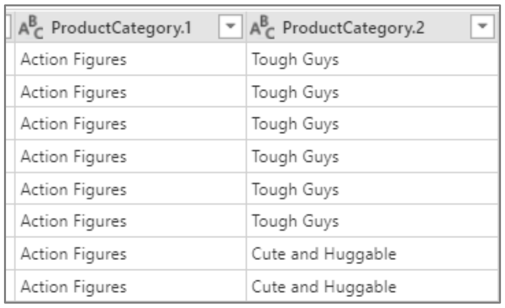
4. Split the **ProductCategory** column up into two separate columns named **Category** and **Subcategory**.
   a) Right-click the **ProductCategory** column and then click the **Split column** > **By delimiter** command.



   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.

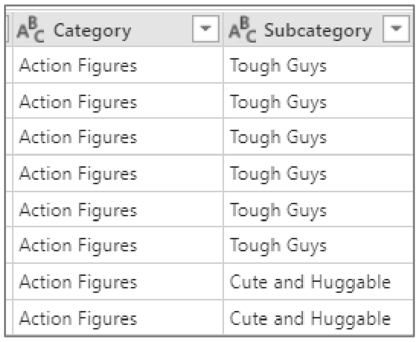

Split column
Specify the delimiter used to split the text column.
[✓ Basic] Advanced
Separator: --Custom--
>

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

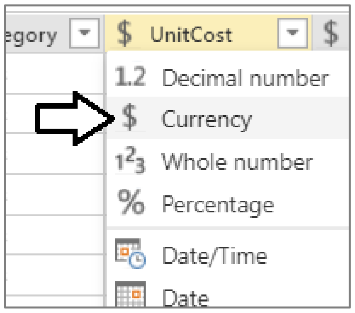
f) Rename the **ProductCategory.1** column to **Category** and rename **ProductCategory.2** to **Subcategory**.



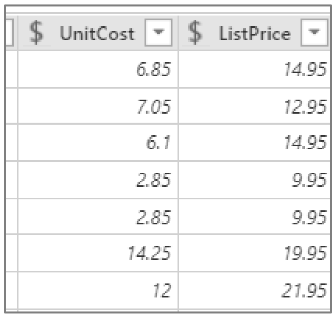
Category	Subcategory
Action Figures	Tough Guys
Action Figures	Tough Guys
Action Figures	Tough Guys
Action Figures	Tough Guys
Action Figures	Tough Guys
Action Figures	Tough Guys
Action Figures	Cute and Huggable
Action Figures	Cute and Huggable

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**.

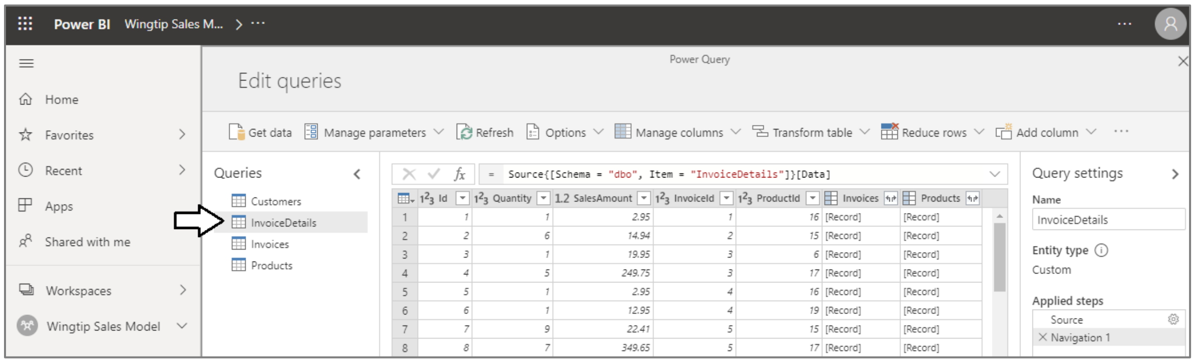


UnitCost	ListPrice
6.85	14.95
7.05	12.95
6.1	14.95
2.85	9.95
2.85	9.95
14.25	19.95
12	21.95

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.

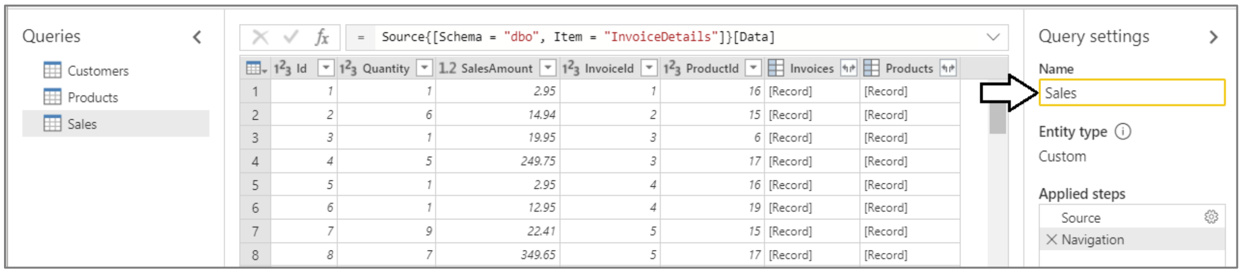


```
= Source{[Schema = "dbo", Item = "InvoiceDetails"]}[Data]
```

	Id	Quantity	SalesAmount	InvoiceId	ProductId	Invoices	Products
1	1	1	2.95	1	16	[Record]	[Record]
2	2	6	14.94	2	15	[Record]	[Record]
3	3	1	19.95	3	6	[Record]	[Record]
4	4	5	249.75	3	17	[Record]	[Record]
5	5	1	2.95	4	16	[Record]	[Record]
6	6	1	12.95	4	19	[Record]	[Record]
7	7	9	22.41	5	15	[Record]	[Record]
8	8	7	349.65	5	17	[Record]	[Record]

Name: InvoiceDetails
Entity type: Custom
Applied steps: Source, Navigation 1

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

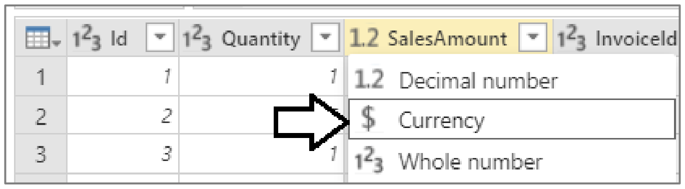


```
= Source{[Schema = "dbo", Item = "InvoiceDetails"]}[Data]
```

	Id	Quantity	SalesAmount	InvoiceId	ProductId	Invoices	Products
1	1	1	2.95	1	16	[Record]	[Record]
2	2	6	14.94	2	15	[Record]	[Record]
3	3	1	19.95	3	6	[Record]	[Record]
4	4	5	249.75	3	17	[Record]	[Record]
5	5	1	2.95	4	16	[Record]	[Record]
6	6	1	12.95	4	19	[Record]	[Record]
7	7	9	22.41	5	15	[Record]	[Record]
8	8	7	349.65	5	17	[Record]	[Record]

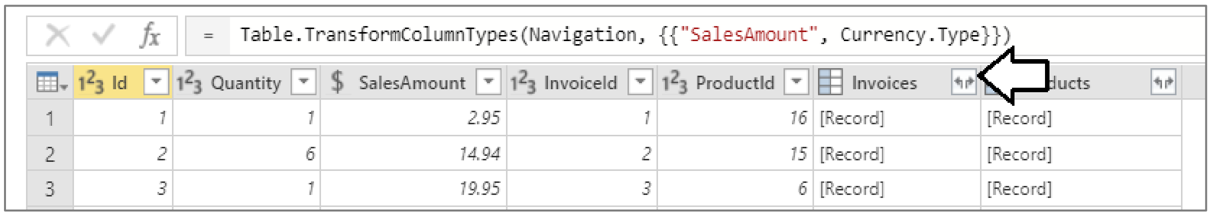
Name: Sales
Entity type: Custom
Applied steps: Source, Navigation

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 to 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.



```
= Table.TransformColumnTypes(Navigation, {{"SalesAmount", Currency.Type}})
```

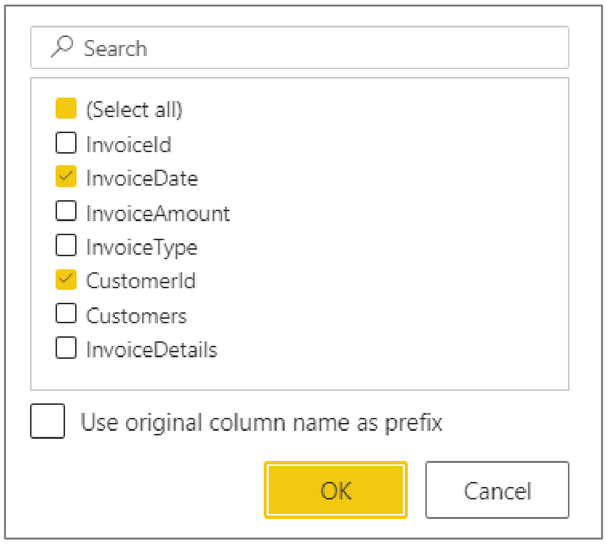
	Id	Quantity	SalesAmount	InvoiceId	ProductId	Invoices	ducts
1	1	1	2.95	1	16	[Record]	[Record]
2	2	6	14.94	2	15	[Record]	[Record]
3	3	1	19.95	3	6	[Record]	[Record]

b) In the **Columns to Expand** dialog, begin by clicking on the **(Select all)** checkbox at the top to unselect all columns.

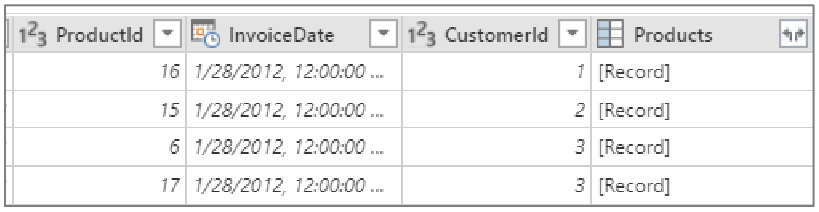
c) Select the checkboxes for the **InvoiceDate** column and the **CustomerId** column.

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.

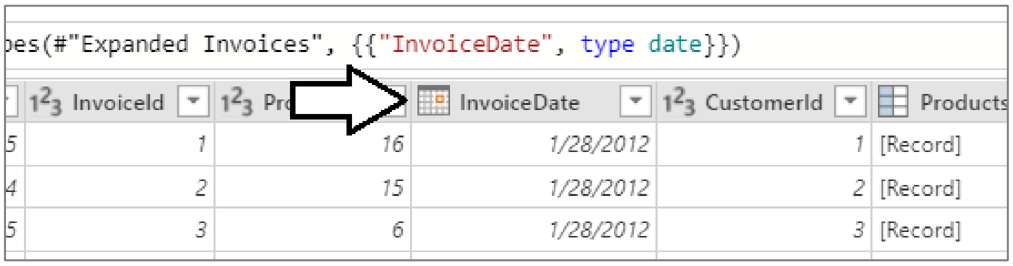


f) You should see that the **InvoiceDate** column and the **CustomerId** column have now been added to the **Sales** query results.



ProductId	InvoiceDate	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.

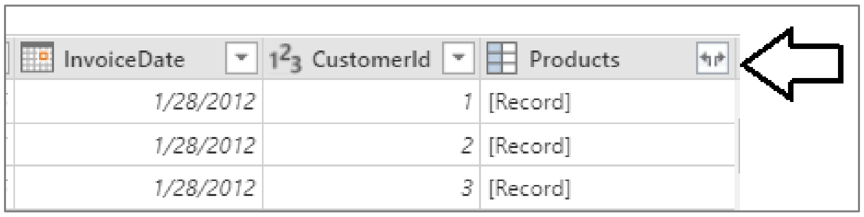


```
pes(#"Expanded Invoices", {{"InvoiceDate", type date}})
```

	InvoiceId	Pr	InvoiceDate	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.



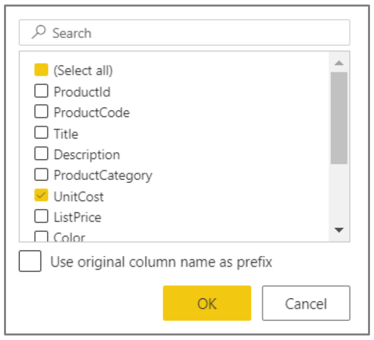
InvoiceDate	CustomerId	Products
1/28/2012	1	[Record]
1/28/2012	2	[Record]
1/28/2012	3	[Record]

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.



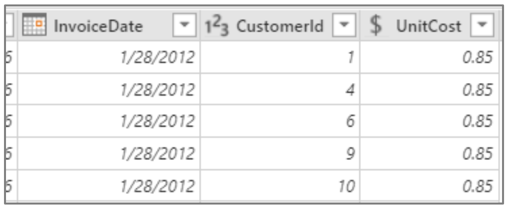
Search

- ▪ (Select all)
- ☐ ProductId
- ☐ ProductCode
- ☐ Title
- ☐ Description
- ☐ ProductCategory
- ☑ UnitCost
- ☐ ListPrice
- ☐ Color

☐ Use original column name as prefix

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.

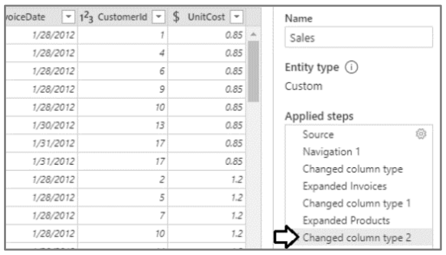


	InvoiceDate	1²3 CustomerId	$ UnitCost
5	1/28/2012	1	0.85
5	1/28/2012	4	0.85
5	1/28/2012	6	0.85
5	1/28/2012	9	0.85
5	1/28/2012	10	0.85

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**.



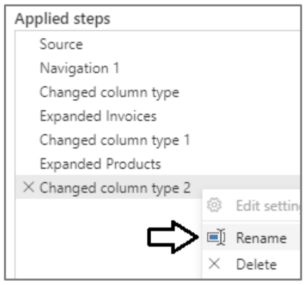
voiceDate	1²3 CustomerId	$ UnitCost
1/28/2012	1	0.85
1/28/2012	4	0.85
1/28/2012	6	0.85
1/28/2012	9	0.85
1/28/2012	10	0.85
1/30/2012	13	0.85
1/31/2012	17	0.85
1/31/2012	17	0.85
1/28/2012	2	1.2
1/28/2012	5	1.2
1/28/2012	7	1.2
1/28/2012	10	1.2

Name: Sales

Entity type: Custom

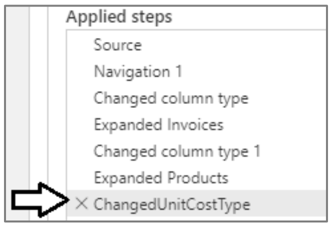
Applied steps
- Source
- Navigation 1
- Changed column type
- Expanded Invoices
- Changed column type 1
- Expanded Products
- Changed column type 2

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



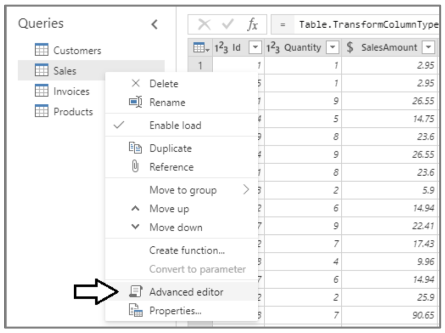
Applied steps
- Source
- Navigation 1
- Changed column type
- Expanded Invoices
- Changed column type 1
- Expanded Products
- Changed column type 2

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

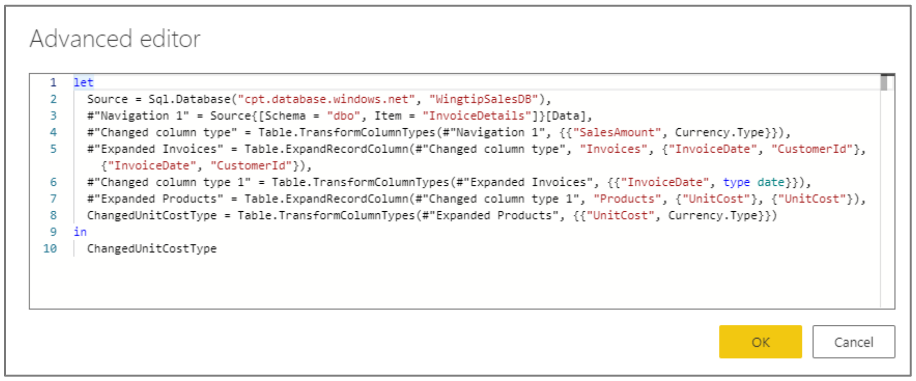


11. Add a new custom column named **ProductCost** to calculate the product of the **Quantity** field multiplied by the **UnitCost** field

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 code in the Advanced editor window.



```
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}})
in
  ChangedUnitCostType
```

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}}),

```

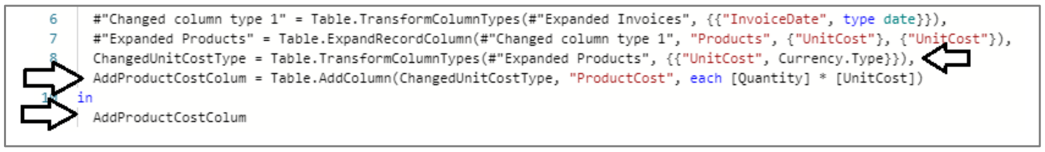
e) Add the following line of M code as the last line of the **let** block.

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

f) Modify the line of code in the **in** block to replace **ChangedUnitCostType** with **AddProductCostColum**.

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

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

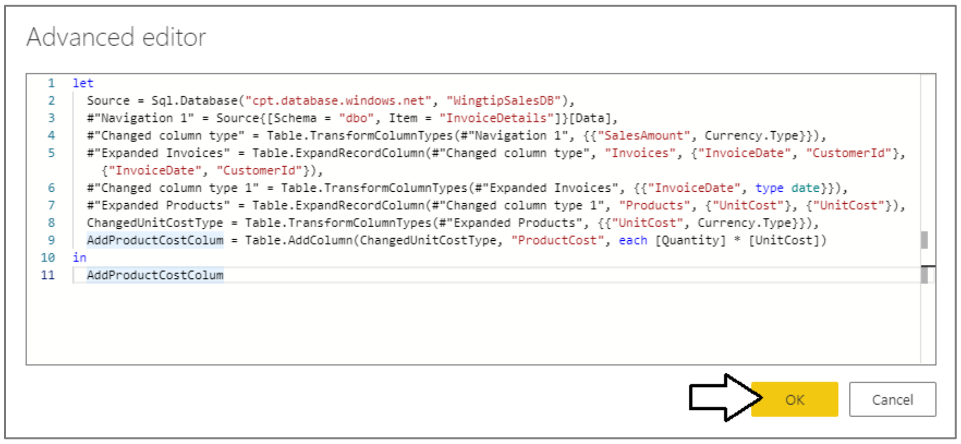


```
    #"Changed column type 1" = Table.TransformColumnTypes(#"Expanded Invoices", {{"InvoiceDate", type date}}),
    #"Expanded Products" = Table.ExpandRecordColumn(#"Changed column type 1", "Products", {"UnitCost"}, {"UnitCost"}),
    ChangedUnitCostType = Table.TransformColumnTypes(#"Expanded Products",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

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

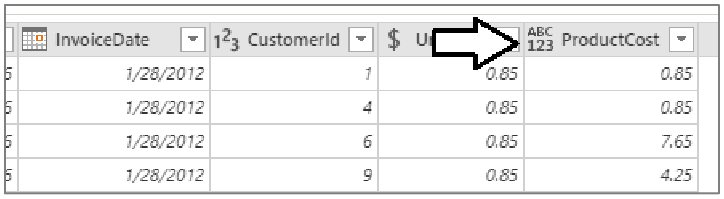
i) Click **OK** to close the **Advanced editor** dialog.



Advanced editor

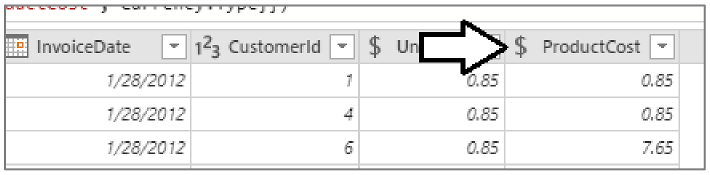
```
let
    Source = Sql.Database("cpt.database.windows.net", "WingtipSalesDB"),
    #"Navigation 1" = Source{[Schema = "dbo", Item = "InvoiceDetails"]}[Data],
    #"Changed column type" = Table.TransformColumnTypes(#"Navigation 1", {{"SalesAmount", Currency.Type}}),
    #"Expanded Invoices" = Table.ExpandRecordColumn(#"Changed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

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



	InvoiceDate	CustomerId	Ur	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

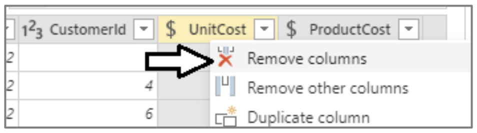
k) Modify the column type of the **Product Cost** column to the **Currency** type.



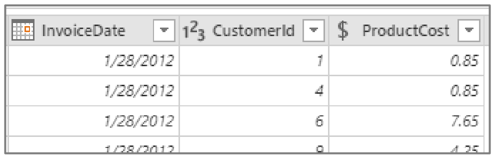
InvoiceDate	CustomerId	Un	ProductCost
1/28/2012	1	0.85	0.85
1/28/2012	4	0.85	0.85
1/28/2012	6	0.85	7.65

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.



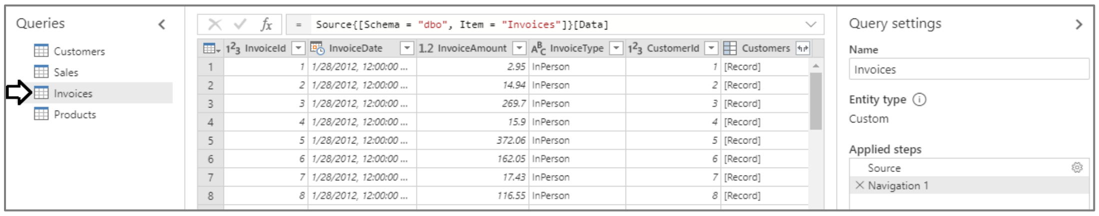
13. Now you should see the **ProductCost** column in the query output, but not the **UnitCost** column.



InvoiceDate	CustomerId	ProductCost
1/28/2012	1	0.85
1/28/2012	4	0.85
1/28/2012	6	7.65

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.



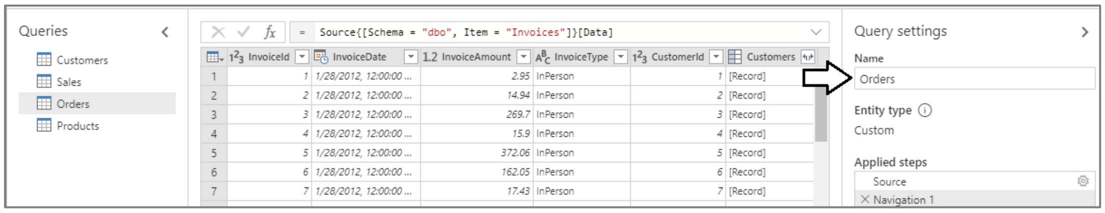
Queries: Customers, Sales, Invoices, Products

```
= Source{[Schema = "dbo", Item = "Invoices"]}[Data]
```

	InvoiceId	InvoiceDate	InvoiceAmount	InvoiceType	CustomerId	Customers
1	1	1/28/2012, 12:00:00 ...	2.95	InPerson	1	[Record]
2	2	1/28/2012, 12:00:00 ...	14.94	InPerson	2	[Record]
3	3	1/28/2012, 12:00:00 ...	269.7	InPerson	3	[Record]
4	4	1/28/2012, 12:00:00 ...	15.9	InPerson	4	[Record]
5	5	1/28/2012, 12:00:00 ...	372.06	InPerson	5	[Record]
6	6	1/28/2012, 12:00:00 ...	162.05	InPerson	6	[Record]
7	7	1/28/2012, 12:00:00 ...	17.43	InPerson	7	[Record]
8	8	1/28/2012, 12:00:00 ...	116.55	InPerson	8	[Record]

Query settings
Name: Invoices
Entity type: Custom
Applied steps: Source, Navigation 1

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



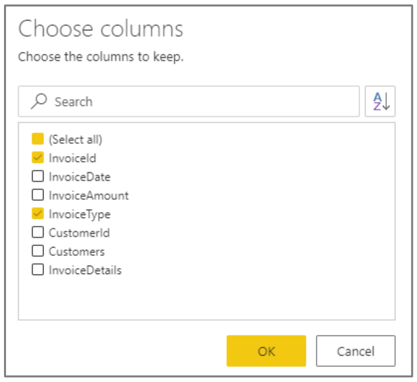
Queries: Customers, Sales, Orders, Products

```
= Source{[Schema = "dbo", Item = "Invoices"]}[Data]
```

	InvoiceId	InvoiceDate	InvoiceAmount	InvoiceType	CustomerId	Customers
1	1	1/28/2012, 12:00:00 ...	2.95	InPerson	1	[Record]
2	2	1/28/2012, 12:00:00 ...	14.94	InPerson	2	[Record]
3	3	1/28/2012, 12:00:00 ...	269.7	InPerson	3	[Record]
4	4	1/28/2012, 12:00:00 ...	15.9	InPerson	4	[Record]
5	5	1/28/2012, 12:00:00 ...	372.06	InPerson	5	[Record]
6	6	1/28/2012, 12:00:00 ...	162.05	InPerson	6	[Record]
7	7	1/28/2012, 12:00:00 ...	17.43	InPerson	7	[Record]

Query settings
Name: Orders
Entity type: Custom
Applied steps: Source, Navigation 1

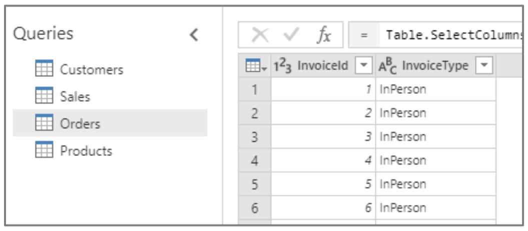
15. Remove the unneeded columns from the **Orders** query.
    a) Click the **Choose columns** button to display the **Choose columns** dialog.
    b) In the **Choose columns** dialog, begin by clicking on the **(Select all)** checkbox at the top to unselect all columns.
    c) Select the checkboxes for **InvoiceId** and **InvoiceType** as shown in the following screenshot and then click **OK**.



Choose columns
Choose the columns to keep.

- ▨ (Select all)
- ✓ InvoiceId
- ☐ InvoiceDate
- ☐ InvoiceAmount
- ✓ InvoiceType
- ☐ CustomerId
- ☐ Customers
- ☐ InvoiceDetails

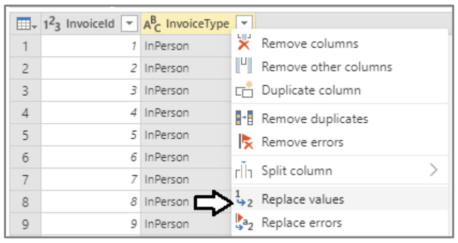
d) You should be able to see that the **Orders** query now only shows the columns that you selected.



	InvoiceId	InvoiceType
1	1	InPerson
2	2	InPerson
3	3	InPerson
4	4	InPerson
5	5	InPerson
6	6	InPerson

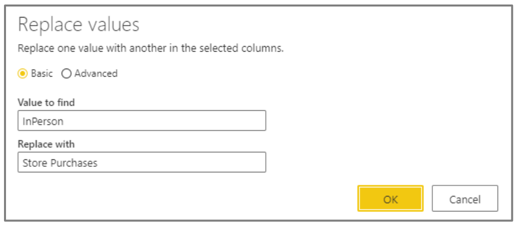
16. Modify the query so that the **InvoiceType** column returns values that are more human-readable.

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



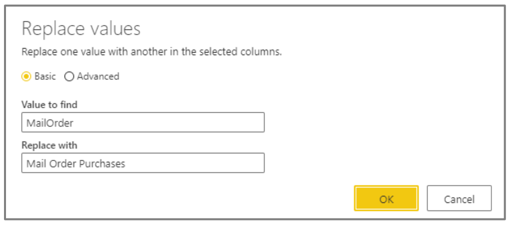
b) In the **Replace Values** dialog, enter a value of **InPerson** in the **Value to find** textbox.

c) Enter a value of **Store Purchases** in the **Replace with** textbox and click **OK** to add your changes to the underlying query.



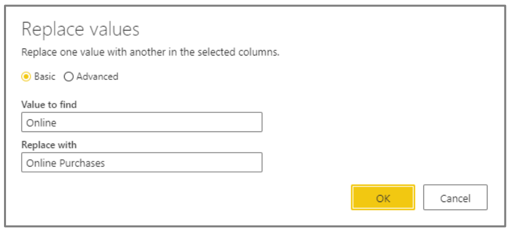
Value to find: InPerson
Replace with: Store Purchases

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



Value to find: MailOrder
Replace with: Mail Order Purchases

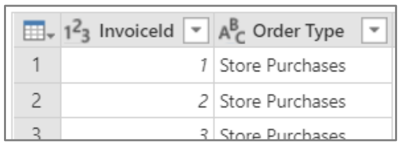
e) Add a third **Replace values** step to replace **Online** with a value of **Online Purchases**.



Value to find: Online
Replace with: Online Purchases

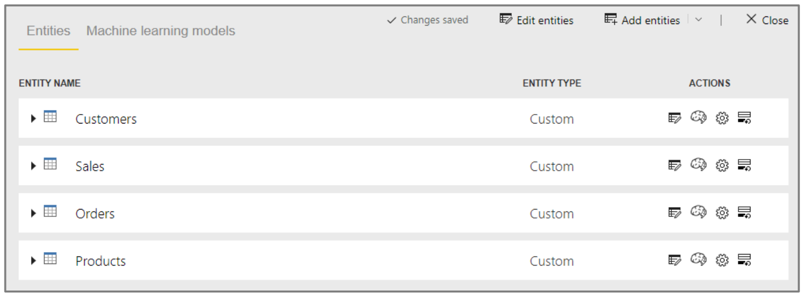
If you scroll down and look at all the rows within the **Orders** query results, you should be able to see that each row has an **InvoiceType** column value of either **Store Purchases**, **Mail Order Purchases** or **Online Purchases**.

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



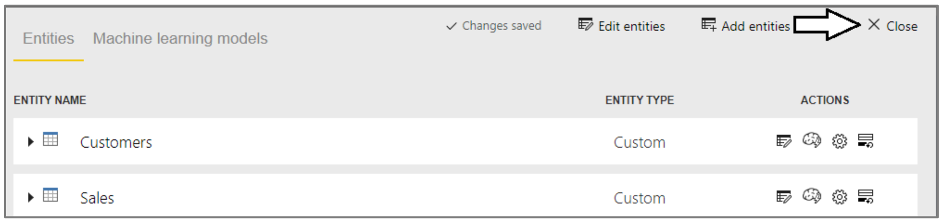
	InvoiceId	Order Type
1	1	Store Purchases
2	2	Store Purchases
3	3	Store Purchases

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



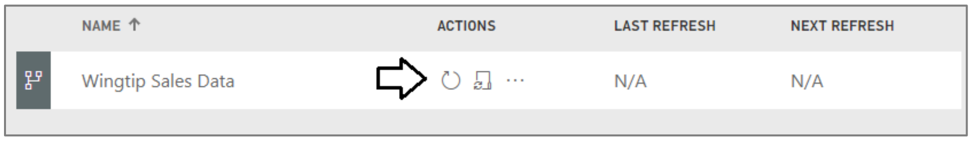
ENTITY NAME	ENTITY TYPE	ACTIONS
Customers	Custom	
Sales	Custom	
Orders	Custom	
Products	Custom	

18. Refresh the **Wingtip Sales Data** dataflow to populate it with data.
    a) Click the **Close** button in the upper right corner of the dataflow summary page.



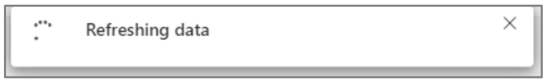
ENTITY NAME	ENTITY TYPE	ACTIONS
Customers	Custom	
Sales	Custom	

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



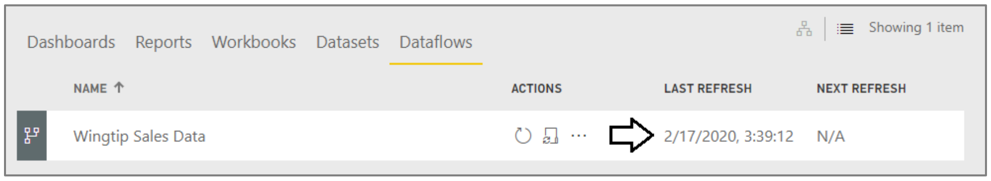
NAME	ACTIONS	LAST REFRESH	NEXT REFRESH
Wingtip Sales Data	···	N/A	N/A

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



Refreshing data

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



NAME	ACTIONS	LAST REFRESH	NEXT REFRESH
Wingtip Sales Data	···	2/17/2020, 3:39:12	N/A

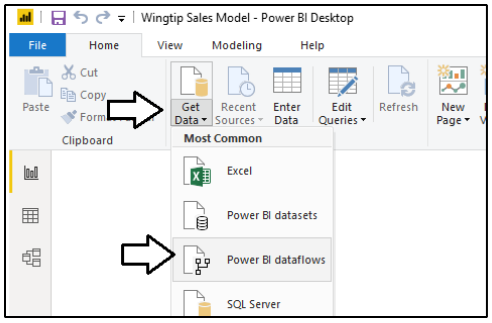
Exercise 3: Importing Dataflow Entity Data with Power BI Desktop

In the following exercise, you will use the **Query Editor** window to modify the **Invoices** query to transform invoice data as it is being loaded into the data model.

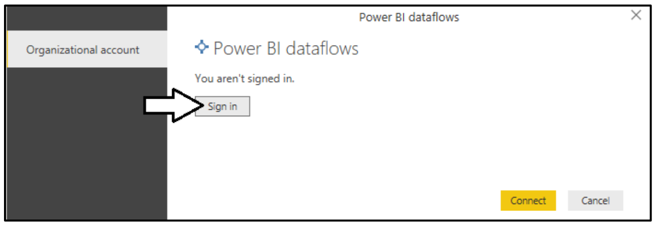
1. Launch Power BI Desktop to start a new project.
2. Save the new project as **Wingtip Sales Model.pbix** using the following path.

```
C:\Student\Projects\Wingtip Sales Model.pbix
```

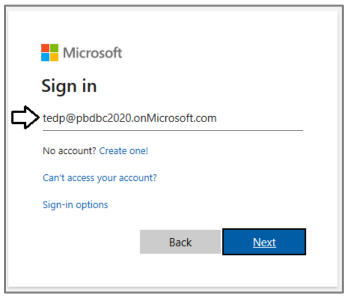
3. Import all four entities from the **Wingtip sales Data** dataflow.
   a) Drop down the **Get Data** menu button on the ribbon and click **Power BI dataflows**.



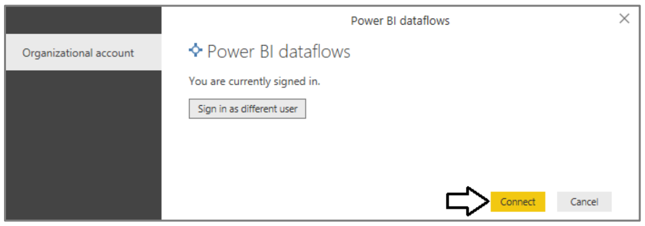
   b) When prompted by the **Power BI dataflows** connect dialog, click **Sign in**.



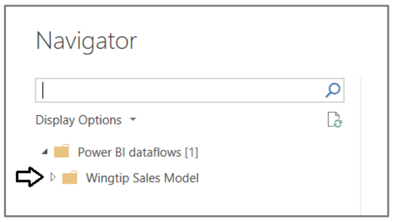
   c) In the **Sign in** dialog, enter your account name and then click **Next**.



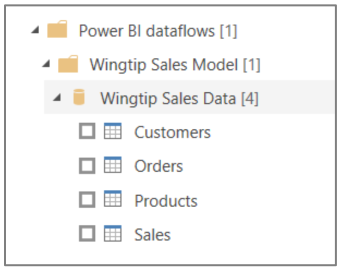
   d) Once you have signed in, click the **Connect** button.



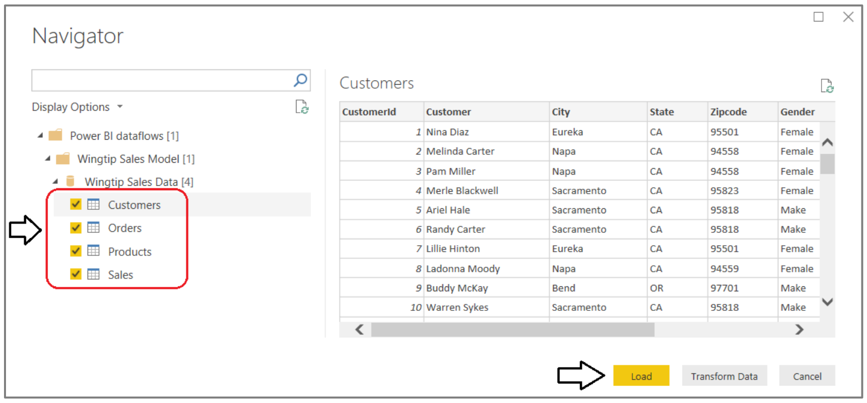
e) In the **Navigator** dialog, expand the **Wingtip Sales Model** workspace



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



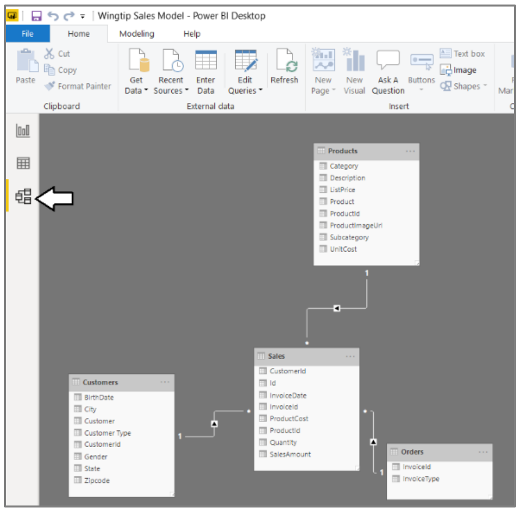
g) Select all four of the entities in the **Wingtip Sales Data** workflow and then click the **Load** button.



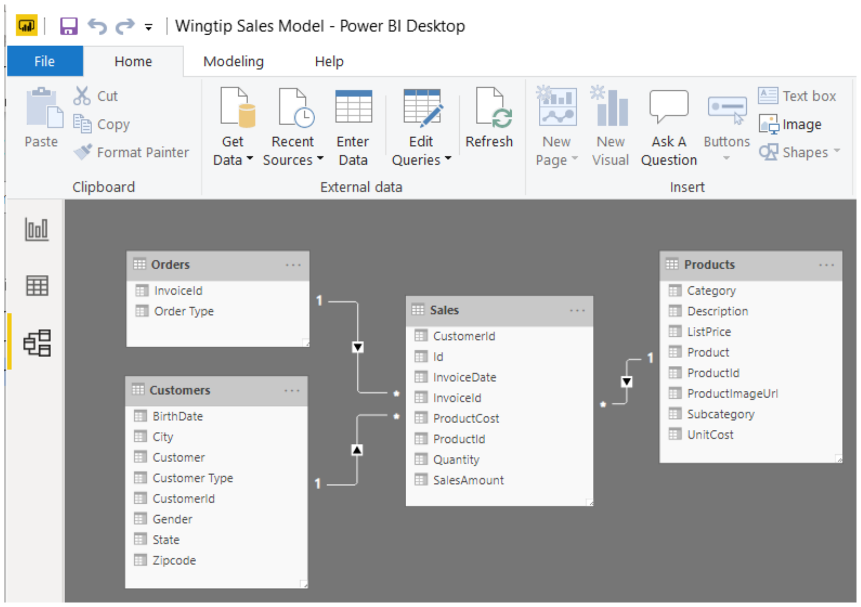
Customers

CustomerId	Customer	City	State	Zipcode	Gender
1	Nina Diaz	Eureka	CA	95501	Female
2	Melinda Carter	Napa	CA	94558	Female
3	Pam Miller	Napa	CA	94558	Female
4	Merle Blackwell	Sacramento	CA	95823	Female
5	Ariel Hale	Sacramento	CA	95818	Make
6	Randy Carter	Sacramento	CA	95818	Make
7	Lillie Hinton	Eureka	CA	95501	Female
8	Ladonna Moody	Napa	CA	94559	Female
9	Buddy McKay	Bend	OR	97701	Make
10	Warren Sykes	Sacramento	CA	95818	Make

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**.

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.