

# Microsoft Fabric

Fabric Analyst in a Day

Lab 3

Version: August 2024



Contents

Introduction ..... 3

Shortcut to ADLS Gen2..... 3

    Task 1: Create Shortcut ..... 3

Transform data using Visual Query ..... 8

    Task 2: Create Geo view using Visual Query ..... 8

    Task 3: Create Reseller view using Visual Query ..... 17

    Task 4: Create Sales view using Visual query ..... 23

    Task 5: Create Product view using Visual query..... 30

References..... 34

# Introduction

In our scenario, Sales Data comes from the ERP system and is stored in an ADLS Gen2. It gets updated at noon / 12 PM every day. We need to transform and ingest this data into Lakehouse and use it in our model.

There are multiple ways to ingest this data.

- **Shortcuts:** This creates a link to the data, and we can use Visual query views to transform it. We are going to use Shortcuts in this lab.
- **Notebooks:** This requires us to write code. It is a developer friendly approach.
- **Dataflow Gen2:** You are probably familiar with Power Query or Dataflow Gen1. Dataflow Gen2, as the name indicates, is the newer version of Dataflow. It provides all the capabilities of Power Query / Dataflow Gen1 with the added ability to transform and ingest data into multiple data sources. We are going to introduce this in the next couple of labs.
- **Data Pipeline:** This is an orchestration tool. Activities can be orchestrated to extract, transform, and ingest data. We will be using Data Pipeline to execute Dataflow Gen2 activity which in turn will perform extraction, transformation, and ingestion.

We will start by creating a Shortcut to ingest data into Lakehouse from ADLS Gen2 data source. Once ingested, we are going to use Visual query views to transform it.

By the end of this lab, you will have learned:

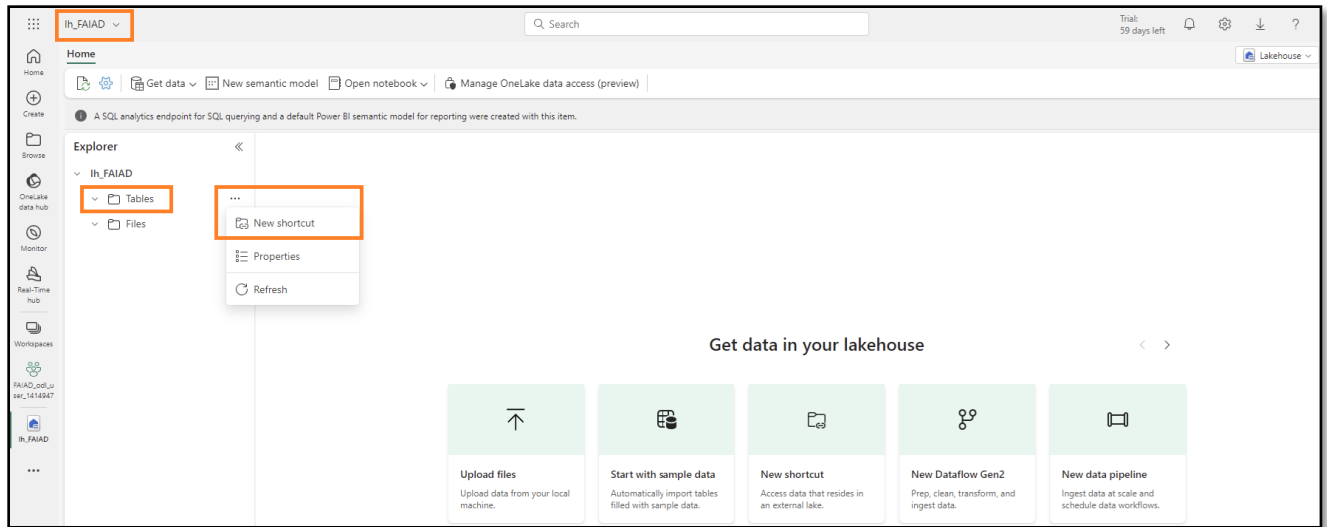
- How to create Shortcut to Lakehouse
- How to transform data using Visual query

## Shortcut to ADLS Gen2

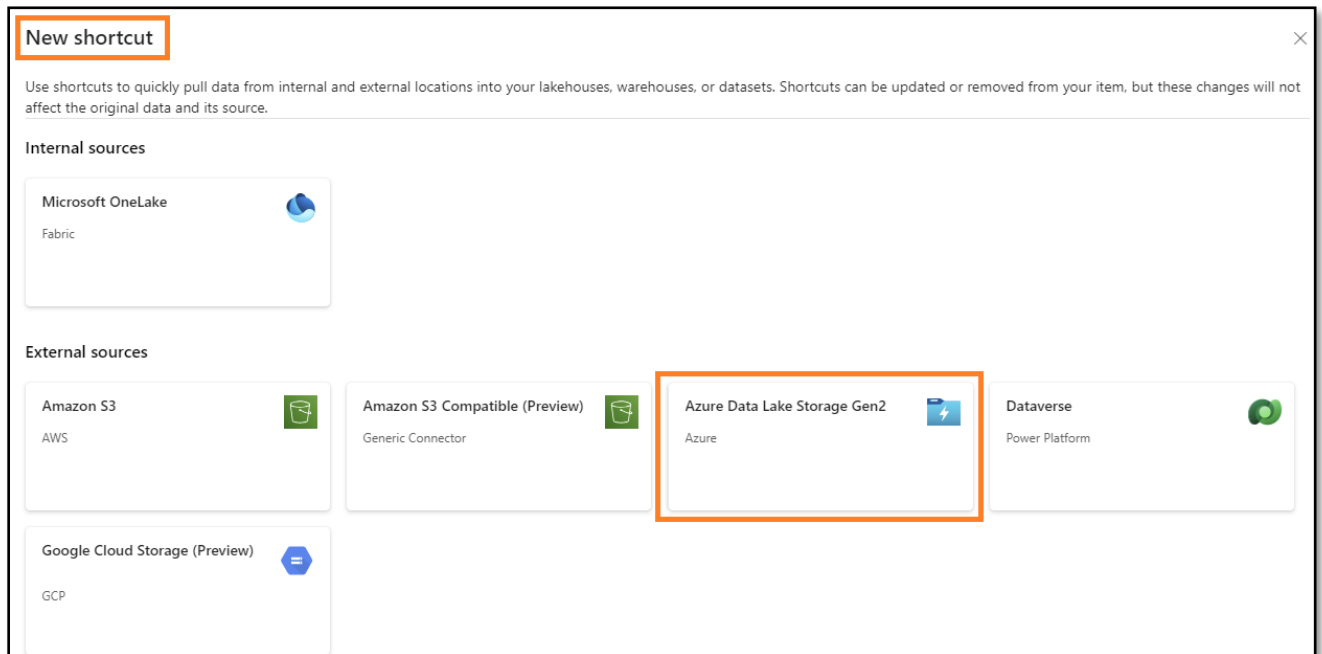
### Task 1: Create Shortcut

Shortcut is used to create a link to the target location. This is like creating shortcuts in Windows desktop.

1. Let's navigate back to the **Fabric workspace** you created in the Lab 2, Task 9.
2. If you have not navigated away after the previous lab, you will be in the Lakehouse screen. If you have navigated away that is fine. Select **lh\_FAIID** to navigate to the Lakehouse.
3. In the **Explorer** panel, select the **ellipsis** next to **Tables**.
4. Select **New Shortcut**.



5. **New Shortcut** dialog opens. Under **External sources**, select **Azure Data Lake Storage Gen2**.



6. You need to create a connection to the ADLS Gen2 data source. Under **Connection Settings** -> **URL** enter this link <https://stvnexblobstorage.dfs.core.windows.net/fabrikam-sales>

7. Select **Account key** from the Authentication kind dropdown.

8. Copy the **Adls storage account Access Key** from the **Environment Variables tab** (next to the Lab Guide tab) and paste it in the **Account key text box**.

9. Select **Next** on the bottom right of the screen.

**New shortcut**

① lh\_FAIAID is located in the region **West US 3**. Any data sourced through this shortcut will be processed in the same region.

Azure Data Lake Storage

Gen2

Azure

[Learn more](#)

**Connection settings**

URL \* ⓘ  
https://stvnexblobstorage.dfs.core.windows.net/fabrikam-...

**Connection credentials**

Connection  
Create new connection

Connection name  
https://stvnexblobstorage.dfs.core.windows.net/fabrikam-...

Authentication kind  
Account key

Account key  
.....

Previous **Next** Cancel

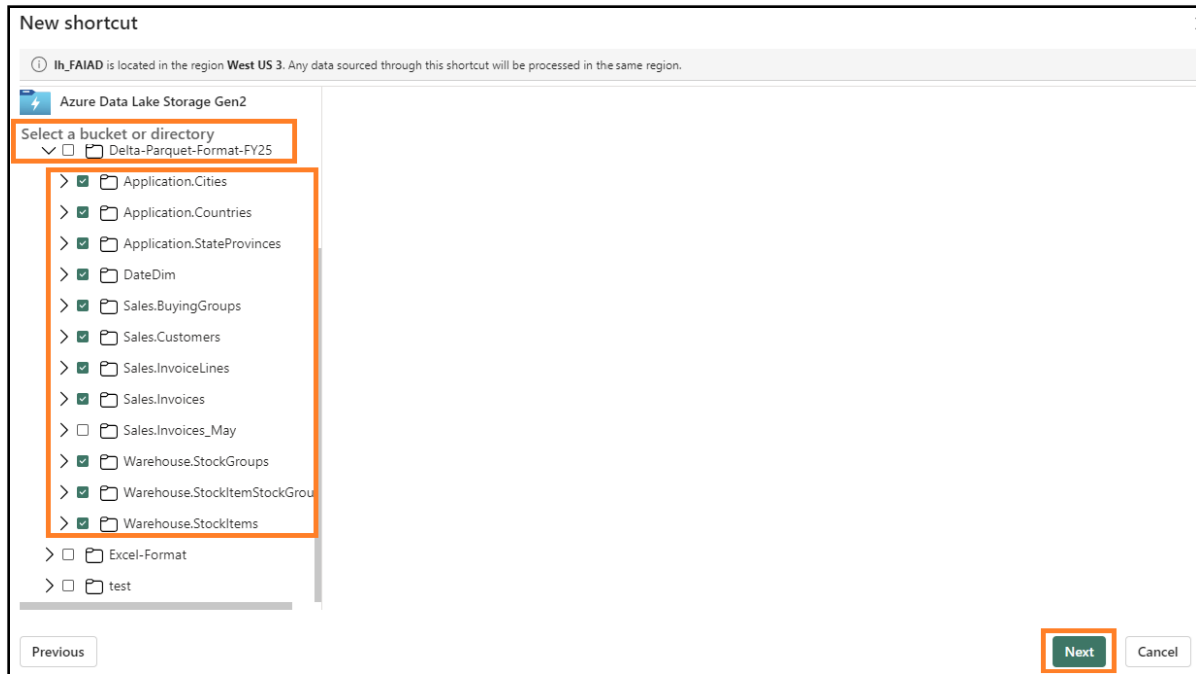
10. You will be connected to ADLS Gen2 with the directory structure displayed in the left panel. Expand **Delta-Parquet-Format-FY25**.

11. **Select** the following directories:

- Application.Cities
- Application.Countries
- Application.StateProvinces
- DateDim
- Sales.BuyingGroups
- Sales.Customers
- Sales.Invoices
- Sales.InvoiceLines
- Warehouse.StockItems
- Warehouse.StockGroups
- Warehouse.StockItemStockGroups

**Note:** Sales.Invoices\_May is the only directory that is **not** selected.

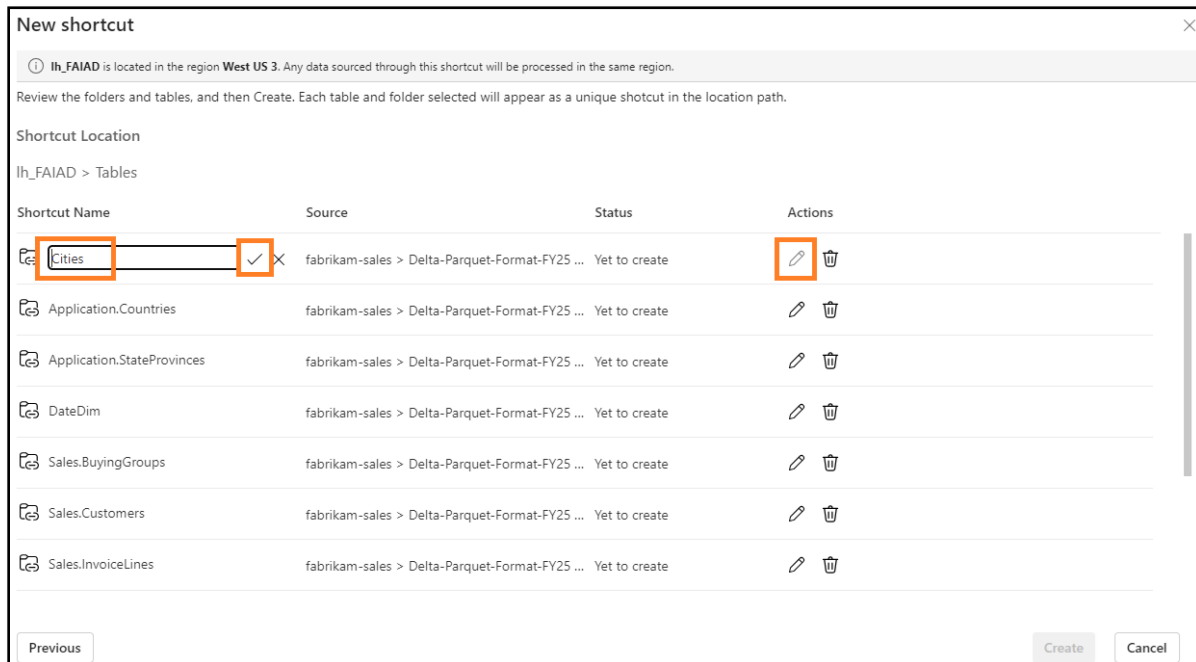
12. Select **Next**.



13. You will be navigated to the next dialog where we can edit the names. Select the **Edit icon** under Actions for **Application.Cities**.

14. Rename **Application.Cities** to **Cities**.

15. Select the check mark next to the name to save the change.



16. Similarly, rename the Shortcut Names as below:

- Application.Countries to **Countries**
- Application.StateProvinces to **States**

- c. DateDim to **Date**
- d. Sales.BuyingGroups to **BuyingGroups**
- e. Sales.Customers to **Customers**
- f. Sales.InvoiceLines to **InvoiceLineItems**
- g. Sales.Invoices to **Invoices**
- h. Warehouse.StockGroups to **ProductGroups**
- i. Warehouse.StockItemStockGroups to **ProductItemGroup**
- j. Warehouse.StockItems to **ProductItem**

**Note:** Double check the names. A typo may cause errors during the lab.

17. Select **Create** to create the Shortcut.

New shortcut






















1

Ih\_FAIAD is located in the region **West US 3**. Any data sourced through this shortcut will be processed in the same region.

Review the folders and tables, and then Create. Each table and folder selected will appear as a unique shortcut in the location path.

Shortcut Location

Ih\_FAIAD > Tables

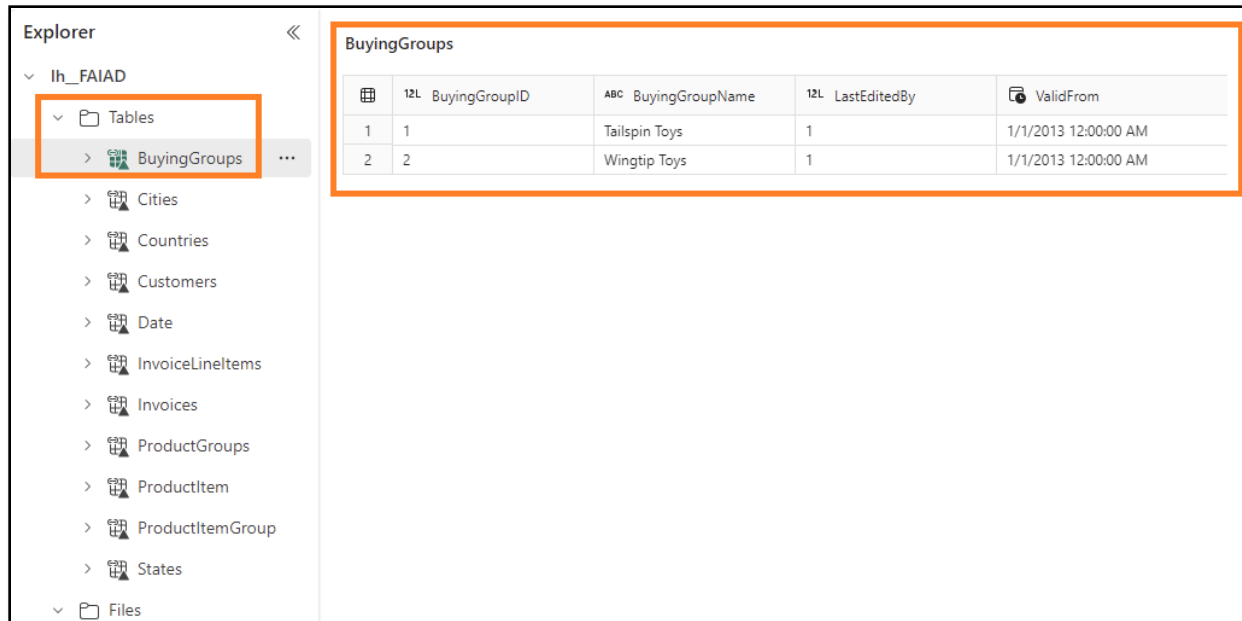
Shortcut Name	Source	Status	Actions
 Cities	fabrikam-sales > Delta-Parquet-Format-FY25 ...	Yet to create	 
 Countries	fabrikam-sales > Delta-Parquet-Format-FY25 ...	Yet to create	 
 States	fabrikam-sales > Delta-Parquet-Format-FY25 ...	Yet to create	 
 Date	fabrikam-sales > Delta-Parquet-Format-FY25 ...	Yet to create	 
 BuyingGroups	fabrikam-sales > Delta-Parquet-Format-FY25 ...	Yet to create	 
 Customers	fabrikam-sales > Delta-Parquet-Format-FY25 ...	Yet to create	 
 InvoiceLineItems	fabrikam-sales > Delta-Parquet-Format-FY25 ...	Yet to create	 

Previous

Create

Cancel

18. Notice all the Shortcuts are created as Tables. Select **BuyingGroups** table and notice we can see a preview of the data in the data panel.

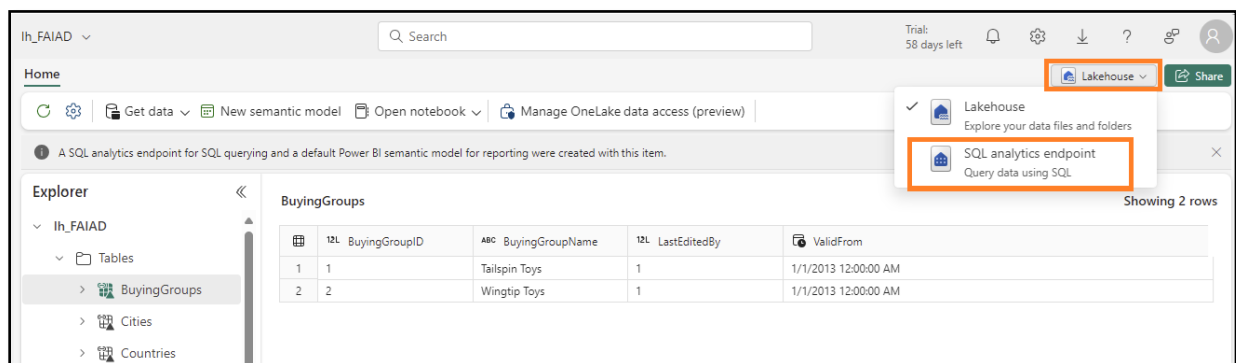


The next step is to transform the data, so we can create a semantic model. We are going to create views to transform the data.

## Transform data using Visual Query

### Task 2: Create Geo view using Visual Query

1. We can access the lakehouse using a SQL endpoint. This provides the ability to query the data and create views. On the **top right** of the screen, select **Lakehouse -> SQL analytics endpoint**.

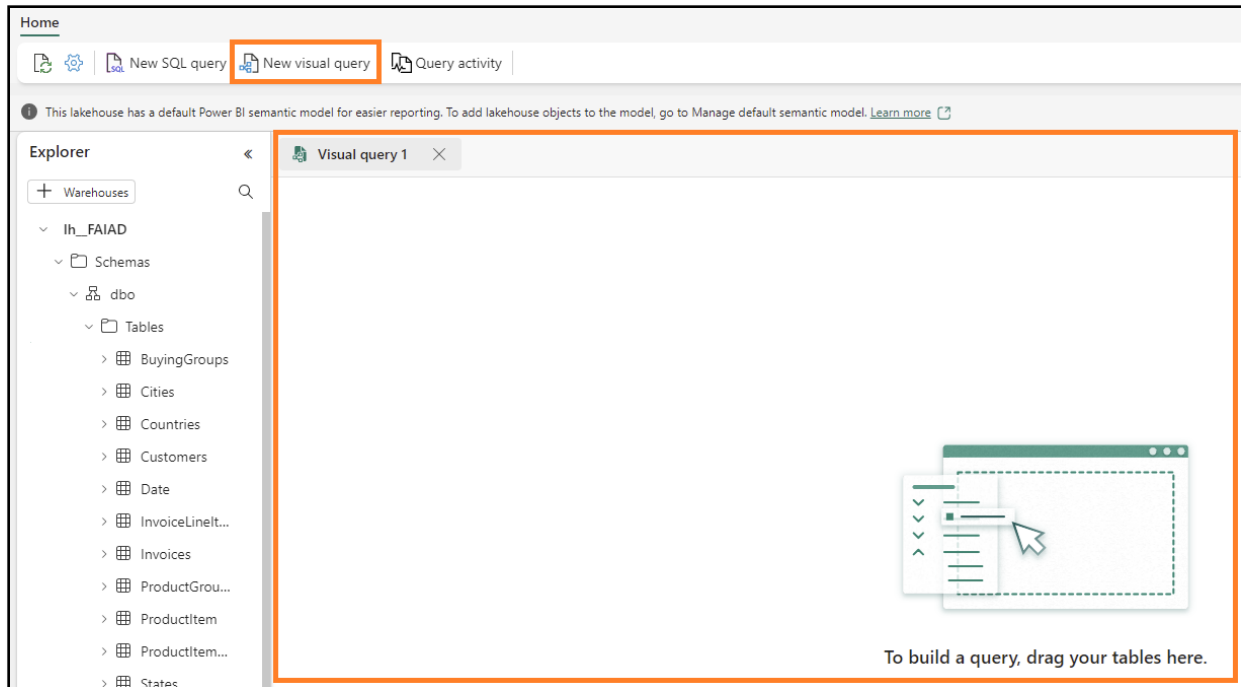


You will be navigated to SQL analytics endpoint. Notice the Explorer panel has changed. You now can create views, stored procedures, queries and more. We are going to create a visual query as it provides a Power Query like interface and save this as a view.

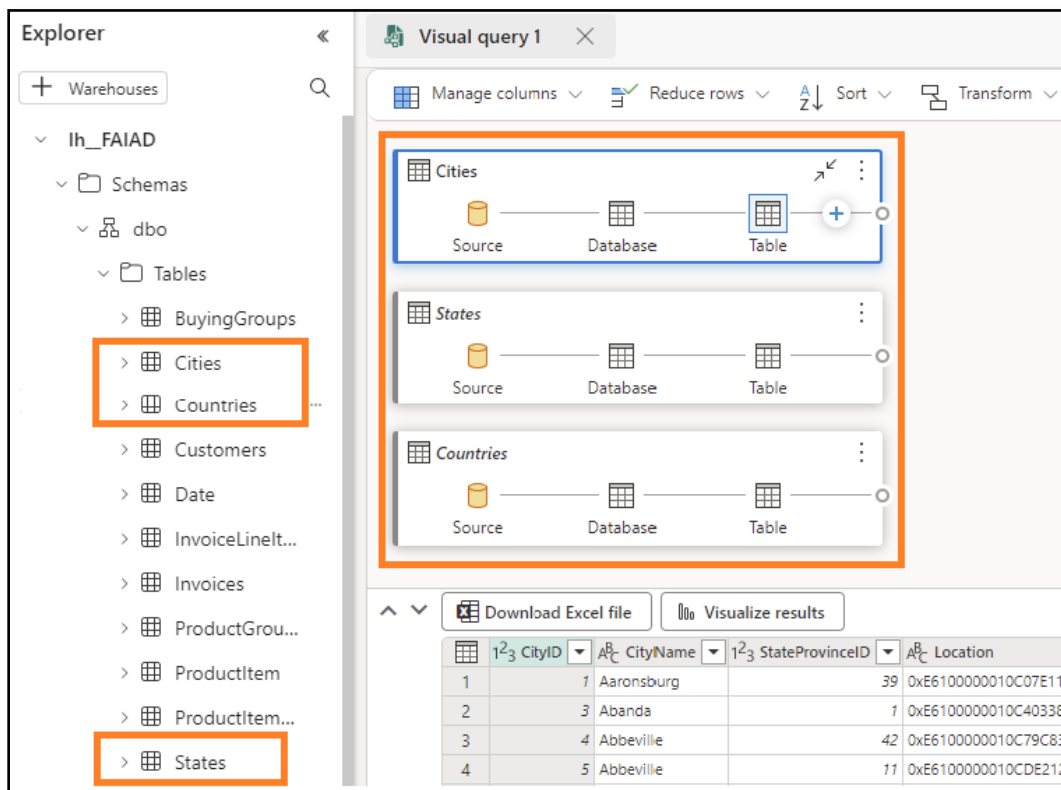
We will start by creating Geo view. We need to merge data from Cities, States and Countries query to create Geo.

2. From the top menu, select **New visual query**.



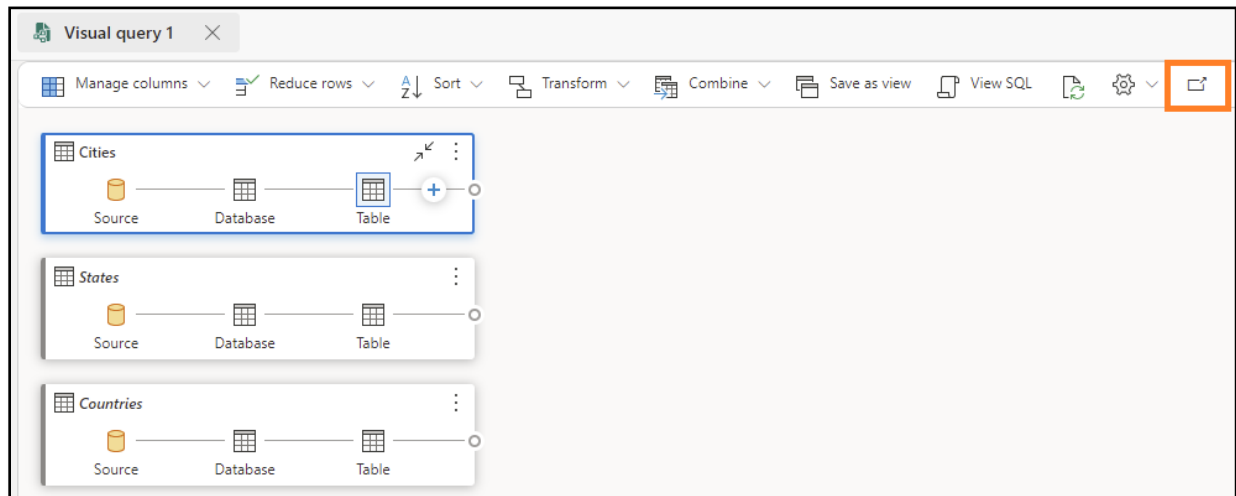


3. We need to drag tables to the Visual query panel to build a query. Let's drag, Cities, States and Countries query into the visual query panel.

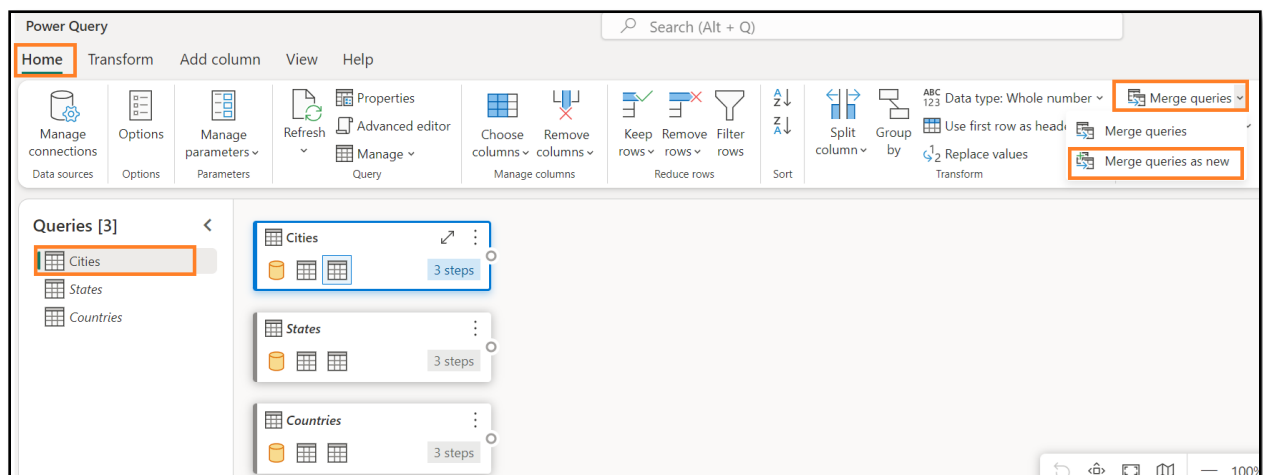


We need to merge these queries. And visual query comes with the option to use Power Query editor. Let's use this, since we are familiar with this.

4. From the menu in Visual query editor, select the **Focus mode** icon (towards the right). You will be navigated to Power Query editor.



5. With Cities query selected, from the Power Query editor ribbon, select **Home -> Merge queries -> Merge queries as new**. Merge queries dialog opens.



6. In the **left table to merge**, select **Cities**.
7. In the **right table to merge**, select **States**.
8. Select **StateProvinceID** columns from both the tables. We are going to join using this column.
9. Select **Inner** as the **Join kind**.
10. Select **OK**.

**Merge** ?

Select tables and matching columns to create a merged table.

**Left table for merge \***

Cities

1^2_3 CityID	A^C CityName	1^2_3 StateProvinceID	A^C Location	1,2 LatestRe
1	Aaronsburg	39	DxE6100000010C07E11B542C73444087C09140...	
3	Abanda	1	DxE6100000010C4033880FEC8C4040EFBF3A33E...	
4	Abbeville	42	DxE6100000010C79C83956CE164140CCAC4AC7...	
5	Abbeville	11	DxE6100000010CDE2120BAFBFD3F408F876302...	

**Right table for merge \***

States

1^2_3 StateProvinceID	A^C StateProvinceCode	A^C StateProvinceName	1^2_3 CountryID	A^C SalesTerritory
1	AL	Alabama	230	Southeast
2	AK	Alaska	230	Far West
3	AZ	Arizona	230	Southwest
4	AR	Arkansas	230	Southeast

**Join kind**

Left outer Right outer Full outer **Inner** Left anti Right anti

☐ Use fuzzy matching to perform the merge

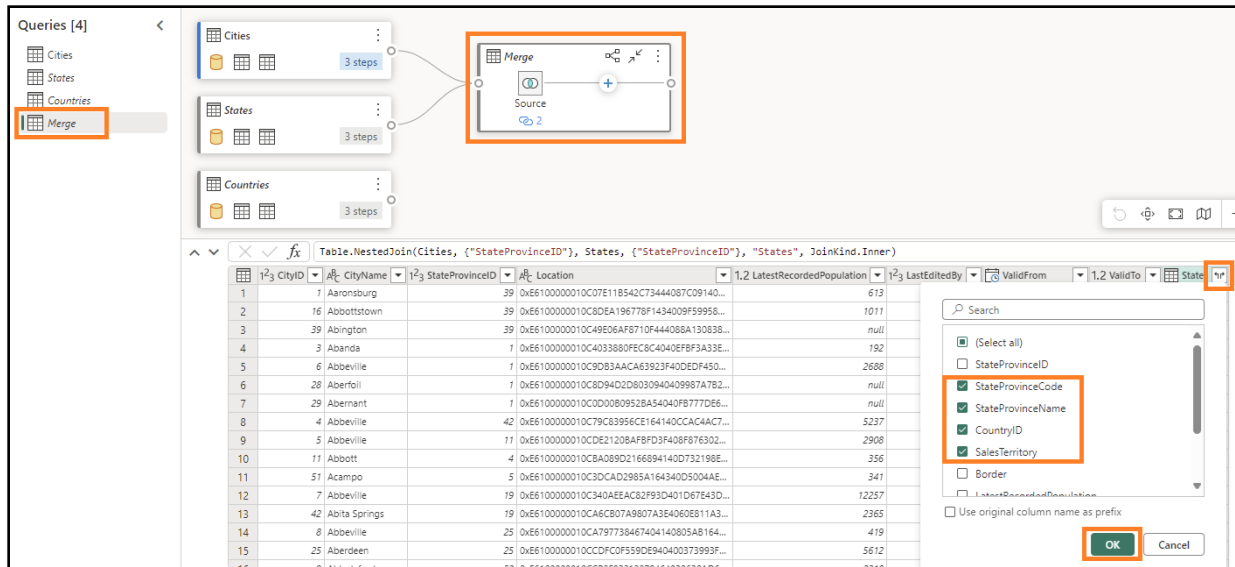
> Fuzzy matching options

✓ The selection matches 37,940 rows from both the tables

**OK** Cancel

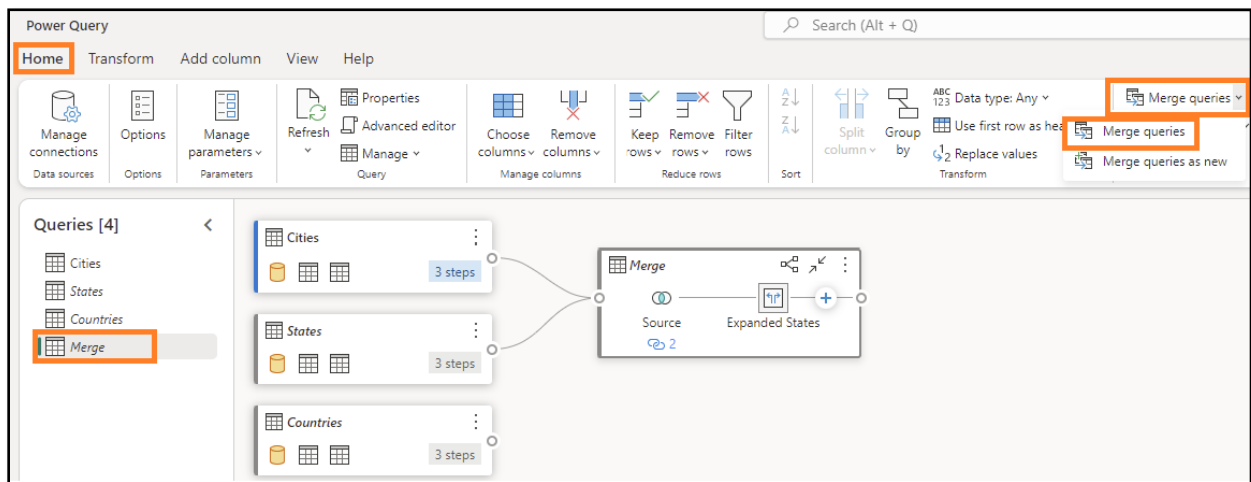
Notice a new query called Merge has been created. We need a few columns from States.

11. In the **Data view** (bottom panel), click on the **double arrow** next to the **States** column (last column to the right).
12. A panel opens. **Select** following columns:
  - a. StateProvinceCode
  - b. StateProvinceName
  - c. CountryID
  - d. SalesTerritory
13. Select **OK**.



We need to merge Countries query now.

14. With Merge query selected, select **Home -> Merge queries -> Merge queries** from the ribbon.



15. Merge query dialog opens. In the **right table to merge**, select **Countries**.

16. Select **CountryID** columns from both the tables. We are going to join using this column.

17. Select **Inner** as the **Join kind**.

18. Select **OK**.

**Merge** ?

Select a table and matching columns to create a merged table.

Merge

1,2 ValidTo	1,2 StateProvinceCode	1,2 StateProvinceName	1,2 CountryID	1,2 SalesTerritory
1/1/2013	PA	Pennsylvania	230	Mideast
1/1/2013	AL	Alabama	230	Southeast
1/1/2013	SC	South Carolina	230	Southeast
1/1/2013	GA	Georgia	230	Southeast

Right table for merge \*

Countries

1,2 CountryID	1,2 CountryName	1,2 FormalName	1,2 IsoAlpha3Code	1,2 IsoNumericCode
1	Afghanistan	Islamic State of Afghanistan	AFG	
3	Albania	Republic of Albania	ALB	
4	Algeria	People's Democratic Republic of Algeria	DZA	
6	Andorra	Principality of Andorra	AND	

Join kind

Left outer Right outer Full outer Inner Left anti Right anti

☐ Use fuzzy matching to perform the merge

> Fuzzy matching options

☒ The selection matches 37,940 rows from both the tables

OK Cancel

We need a few columns from Countries.

19. In the **Data view** (bottom panel), click on the **double arrow** next to the **Countries** column.

20. A panel opens. **Select** following columns:

- CountryName
- FormalName
- IsoAlpha3Code
- IsoNumericCode
- CountryType
- Continent
- Region
- Subregion

21. Select **OK**.

Table.NestedJoin("#Expanded States", {"CountryID"}, Countries, {"CountryID"}, "Countries", JoinKind.Inner)

1,2 LatestRecordedPopulation	1,2 LastEditedBy	1,2 ValidFrom	1,2 ValidTo	1,2 StateProvinceCode	1,2 StateProvinceName	1,2 CountryID	1,2 SalesTerritory	1,2 CountryName
613	1	1/1/2013, 12:00:00 AM	1/1/2013, 12:00:00 AM	PA	Pennsylvania	230	Mideast	Afghanistan
192	1	1/1/2013, 12:00:00 AM	1/1/2013, 12:00:00 AM	AL	Alabama	230	Southeast	Albania
5237	1	1/1/2013, 12:00:00 AM	1/1/2013, 12:00:00 AM	SC	South Carolina	230	Southeast	Algeria
2908	1	1/1/2013, 12:00:00 AM	1/1/2013, 12:00:00 AM	GA	Georgia	230	Southeast	Andorra
2688	1	1/1/2013, 12:00:00 AM	1/1/2013, 12:00:00 AM	AL	Alabama	230	Southeast	Afghanistan
12257	1	1/1/2013, 12:00:00 AM	1/1/2013, 12:00:00 AM	LA	Louisiana	230	Southeast	Albania
419	1	1/1/2013, 12:00:00 AM	1/1/2013, 12:00:00 AM	MS	Mississippi	230	Southeast	Algeria
2310	1	1/1/2013, 12:00:00 AM	1/1/2013, 12:00:00 AM	WI	Wisconsin	230	Southeast	Andorra
356	1	1/1/2013, 12:00:00 AM	1/1/2013, 12:00:00 AM	TX	Texas	230	Southeast	Afghanistan
356	1	1/1/2013, 12:00:00 AM	1/1/2013, 12:00:00 AM	AR	Arkansas	230	Southeast	Albania
356	1	1/1/2013, 12:00:00 AM	1/1/2013, 12:00:00 AM	NM	New Mexico	230	Southeast	Algeria
356	1	1/1/2013, 12:00:00 AM	1/1/2013, 12:00:00 AM	VA	Virginia	230	Southeast	Andorra
356	1	1/1/2013, 12:00:00 AM	1/1/2013, 12:00:00 AM	WV	West Virginia	230	Southeast	Afghanistan
356	1	1/1/2013, 12:00:00 AM	1/1/2013, 12:00:00 AM	NC	North Carolina	230	Southeast	Albania
1011	1	1/1/2013, 12:00:00 AM	1/1/2013, 12:00:00 AM	PA	Pennsylvania	230	Southeast	Algeria

Search

☒ (Select all)

☐ CountryID

☒ CountryName

☒ FormalName

☒ IsoAlpha3Code

☒ IsoNumericCode

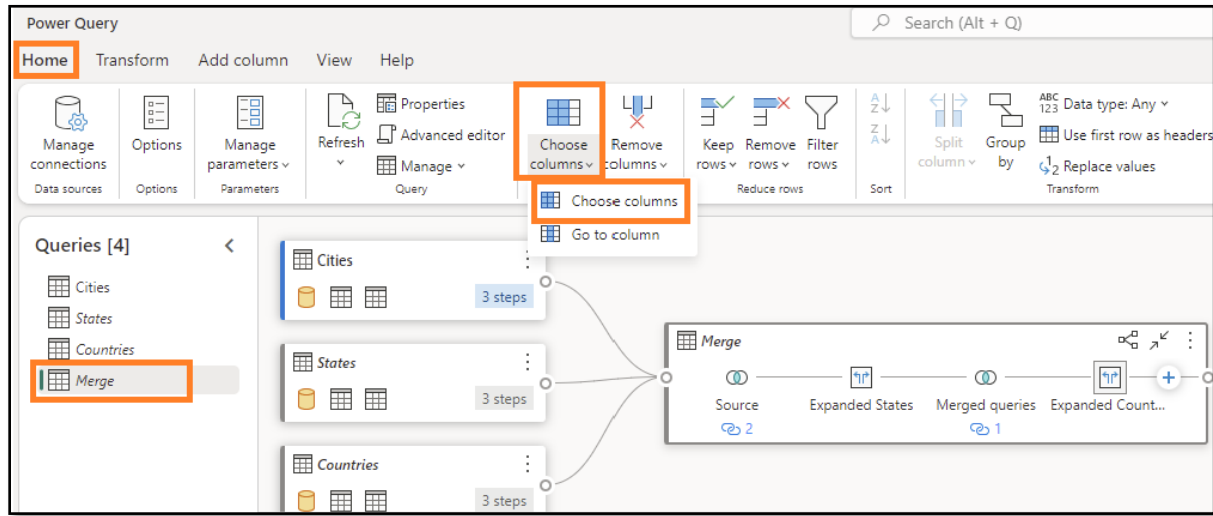
☒ CountryType

☐ Use original column name as prefix

OK Cancel

We do not need all the columns. Let's select only those we need.

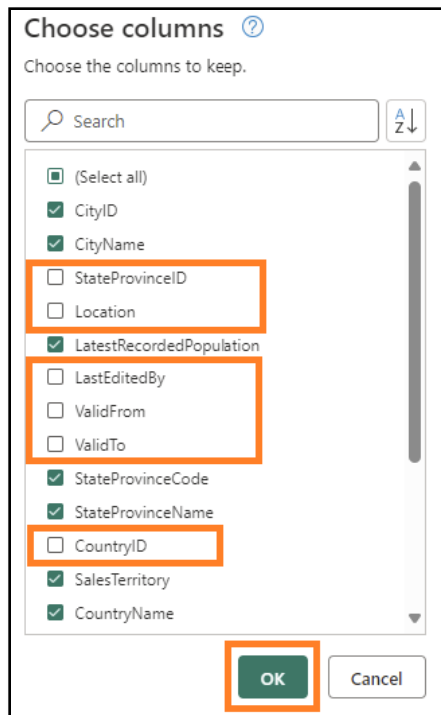
22. With Merge query selected, from the ribbon select **Home** -> **Choose columns** -> **Choose columns**.



23. Choose columns dialog opens. **Uncheck** the following columns.

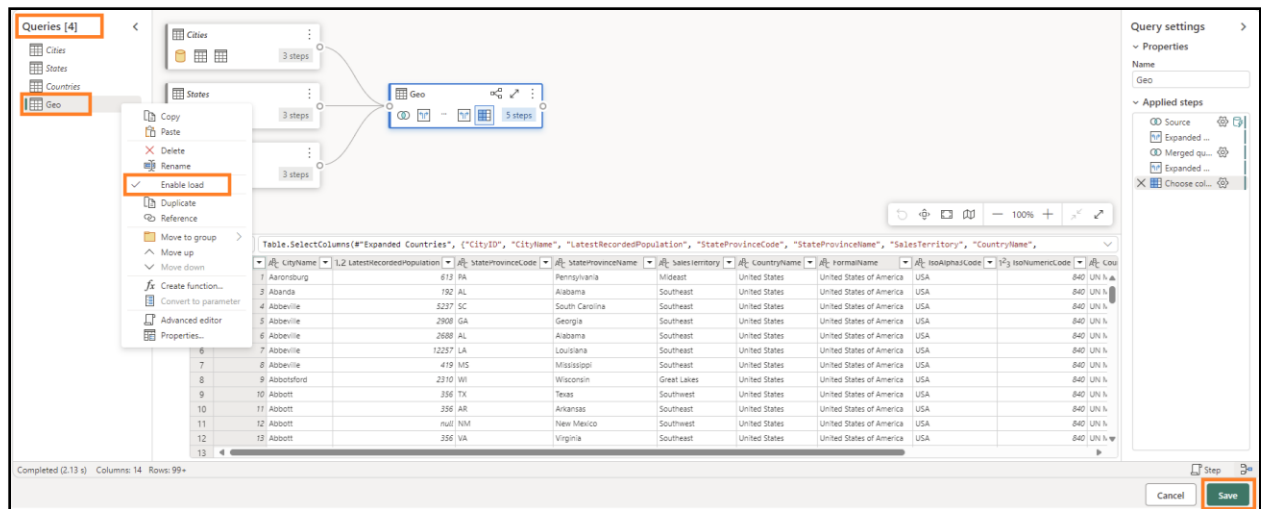
- a. StateProvinceID
- b. Location
- c. LastEditedBy
- d. ValidFrom
- e. ValidTo
- f. CountryID

24. Select **OK**.



Notice the process is like Power Query, we have all the steps recorded both in the Applied Steps panel on the right and the visual view. Let's rename Merge query and Enable load, so that the data is loaded from this query.

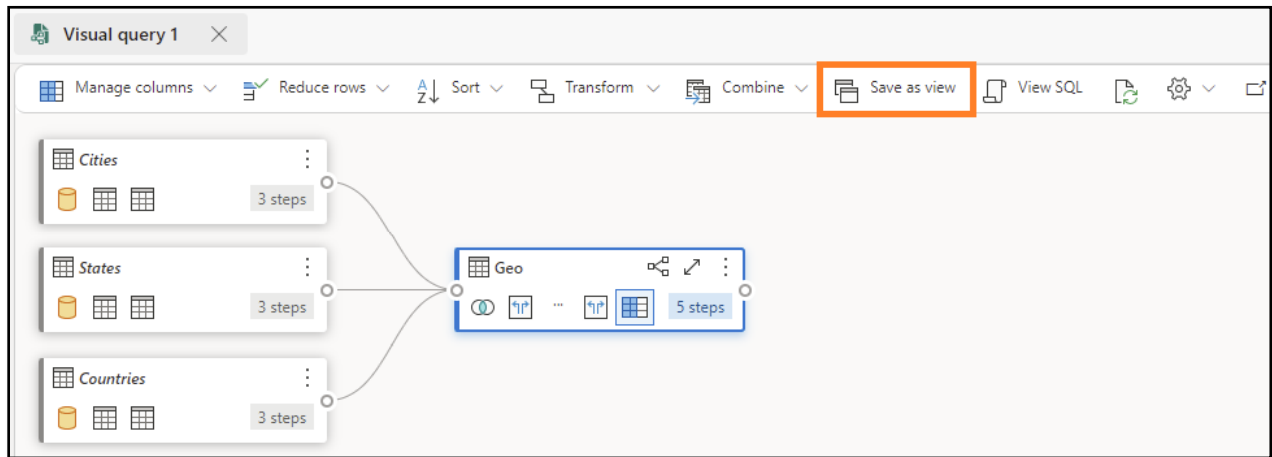
25. **Right click on Merge** query in the Queries (left) panel. Select **Rename** and rename the query to **Geo**.
26. **Right click on Geo** query in the Queries (left) panel. Select **Enable Load** to enable this query.
27. Make sure that Cities, States and Countries queries are **disabled**.
28. Select **Save**.



We will be navigated to Visual query editor. Let's now save this query as a view.

**Note:** All the steps we performed using Power Query editor can be performed using Visual query editor as well.

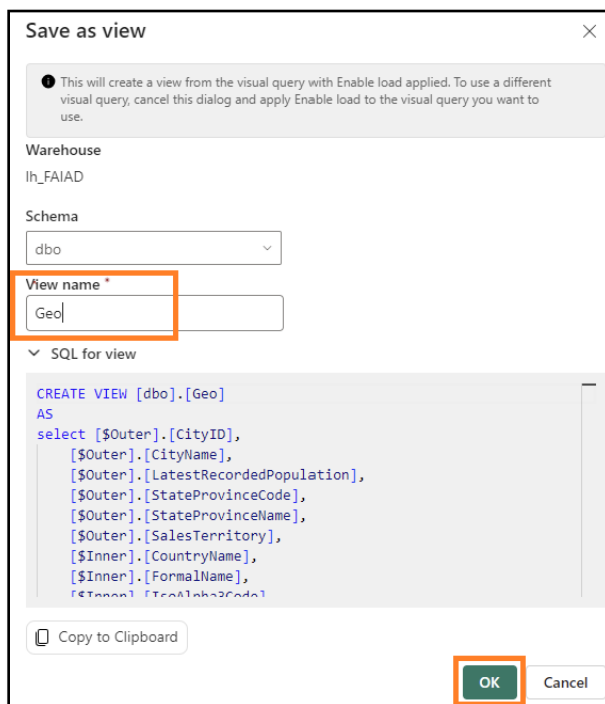
29. From the Visual query editor menu select **Save as view**.



Save as view dialog opens. Notice the SQL query is available. You can review it, if you choose it.

30. Enter **Geo** as **View name**.

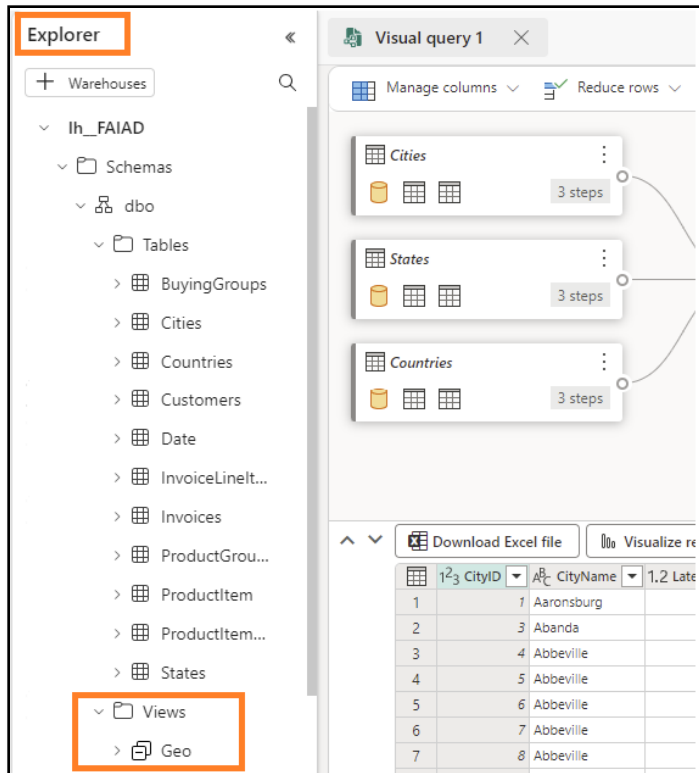
31. Select **OK** to save the view.



You will get an alert once the view is saved.

32. In the Explorer (left) panel, expand **Views**. We have the newly created Geo view.

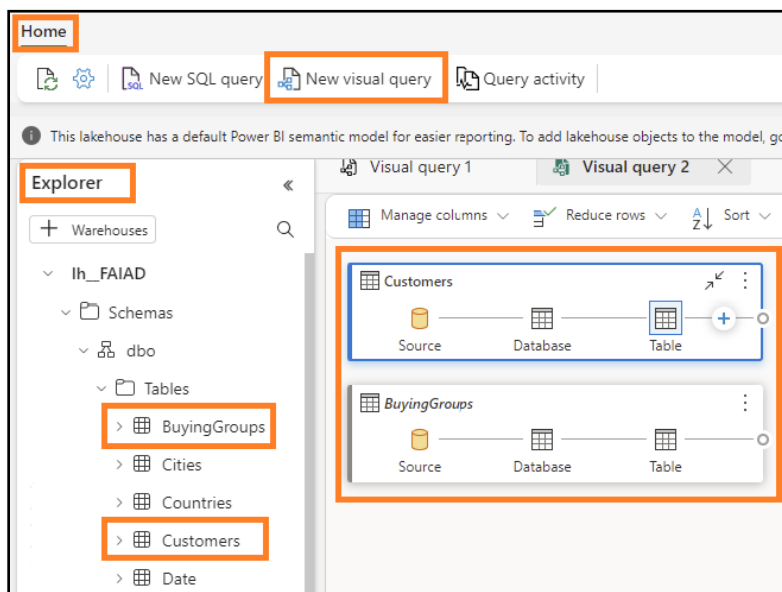




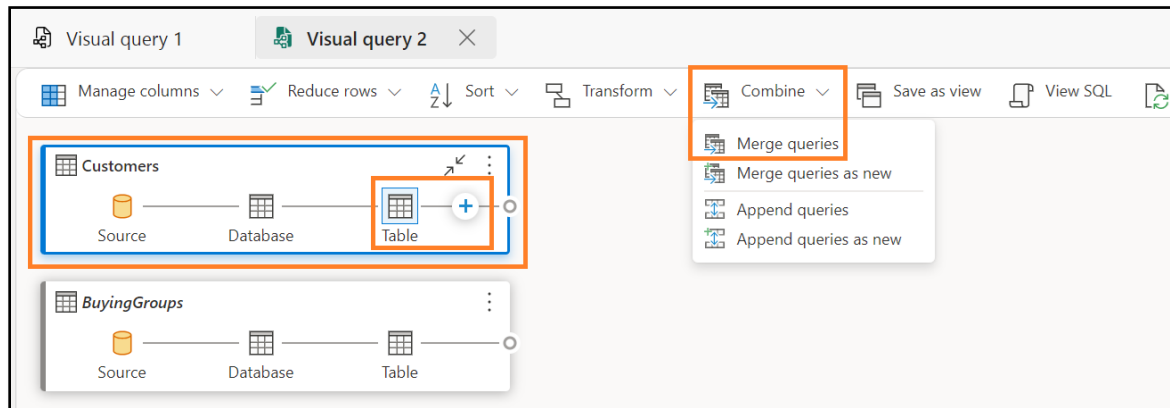
### Task 3: Create Reseller view using Visual Query

Let's create Reseller view which is created by merging Customers table with BuyingGroups table. This time around we will create the view using Visual query.

1. From the Lakehouse menu bar, select **Home -> New visual query**. A new visual query opens.
2. From Explorer section, drag Customers and BuyingGroups tables to the visual query section.

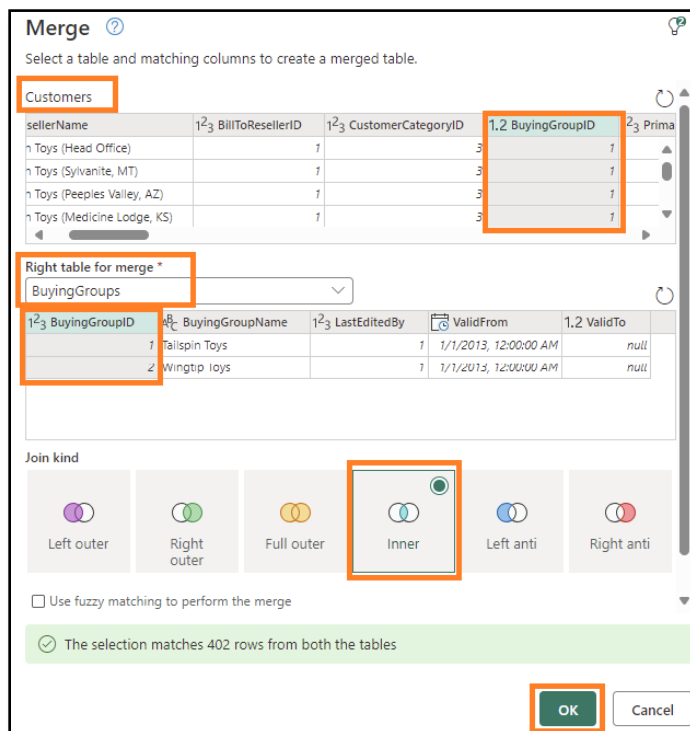


3. **Select Customers** query. When selected, Customers will have a blue border and there is a “+” sign after Table (this indicates we are adding a step after Table. If you do not see the “+” sign after table, you may have selected a different step. Select Table and you will be good to go).
4. From the Visual query menu, select **Combine -> Merge queries**.



Merge dialog opens with Customers selected as the top table.

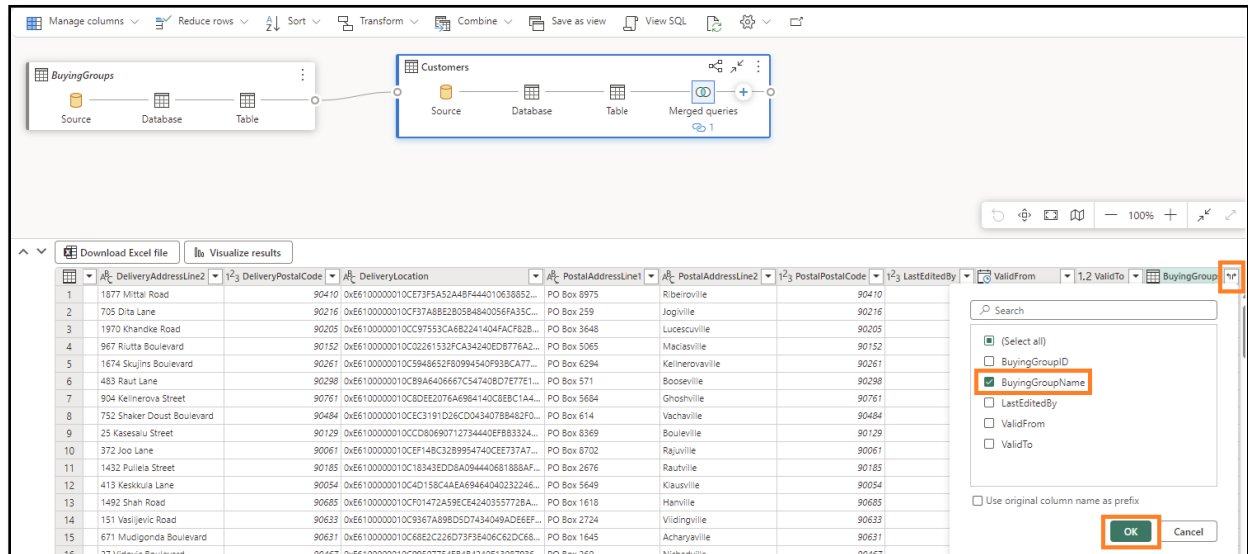
5. In the **right table to merge**, select **BuyingGroups**.
6. Select **BuyingGroupID** columns from both the tables. We are going to join using this column.
7. Select **Inner** as the **Join kind**.
8. Select **OK**.



9. In the **Data view** (bottom panel), click on the **double arrow** next to the **BuyingGroups** column (last column to the right) to select the columns we need from BuyingGroups.

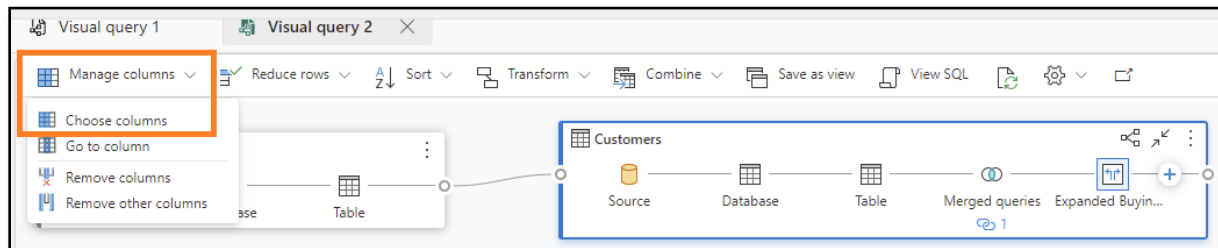
10. A panel opens. **Select BuyingGroupName** column.

11. Select **OK**.



We do not need all the columns. Let's select only those we need.

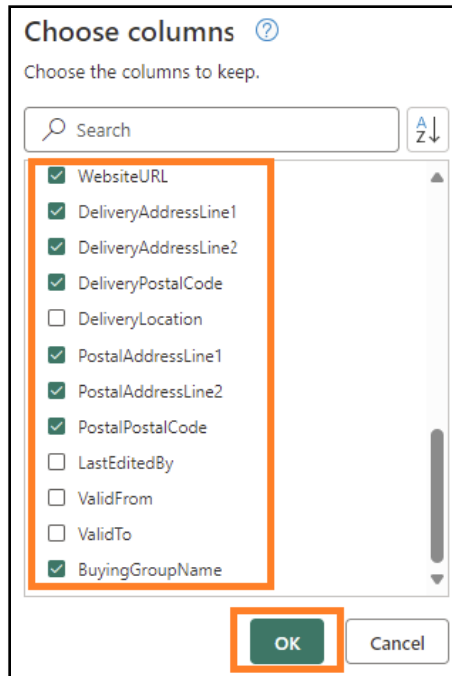
12. From the Visual query menu, select **Manage columns -> Choose columns**.



13. Choose columns dialog opens. **Select** the following columns.

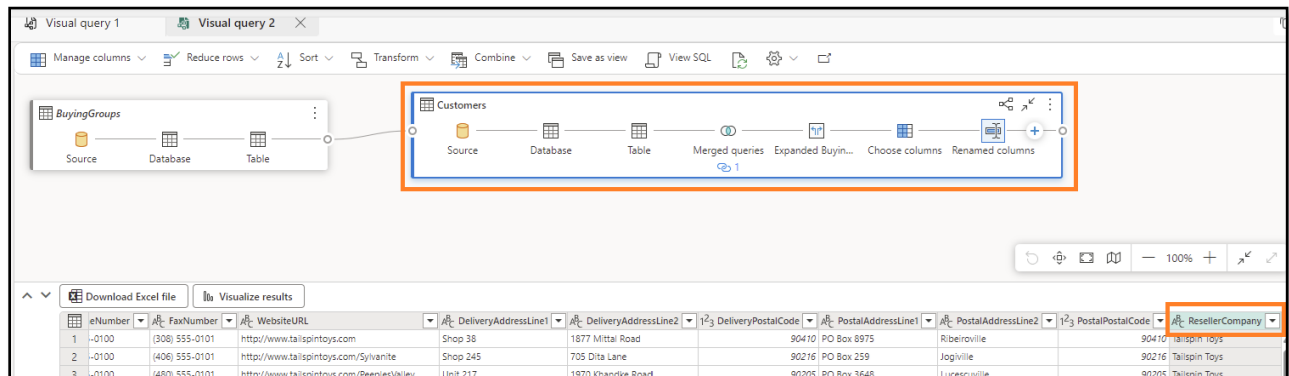
- ResellerID
- ResellerName
- PostalCityID
- PhoneNumber
- FaxNumber
- WebsiteURL
- DeliveryAddressLine1
- DeliveryAddressLine2
- DeliveryPostalCode
- PostalAddressLine1
- PostalAddressLine2
- PostalPostalCode
- BuyingGroupName

14. Select **OK**.



15. Let's rename BuyingGroupName column. In the **Data view**, double click on BuyingGroupName column header to make it editable.

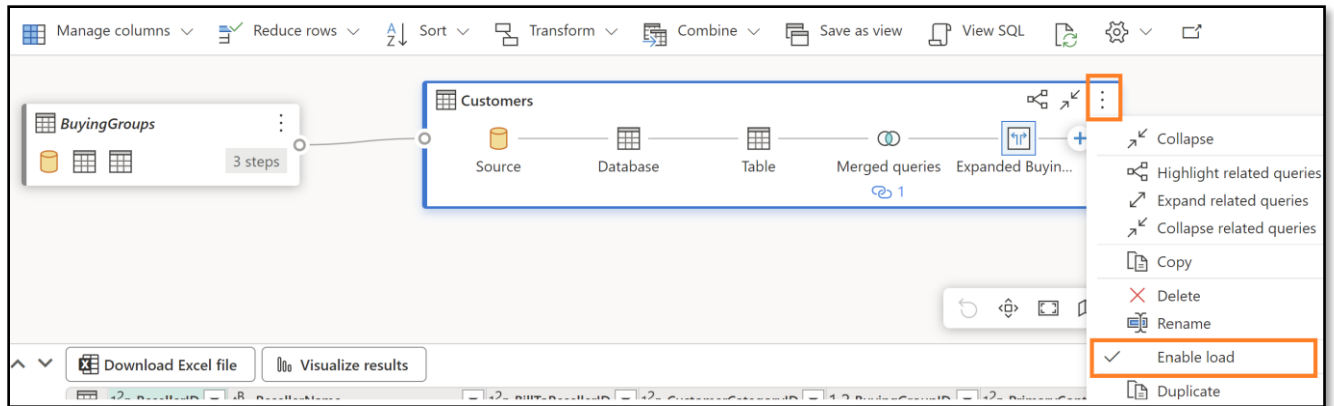
16. Rename the column to **ResellerCompany**.



Notice the Customer table has all the steps documented. Now let's save this view.

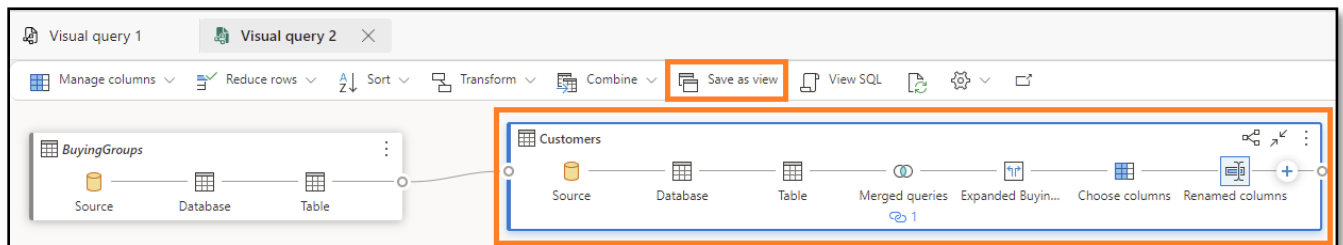
17. We need to save the Customer query as it has all the steps. We need to Enable load. Select the **ellipsis** in the **Customer** query box.

18. Make sure **Enable load** is checked.



**Note:** Customer box should have a blue border if enable load is checked.

19. From the Visual query menu, select **Save as view**.



Save as view dialog opens. Notice the SQL query is available. You can review it, if you choose it.

20. Enter **Reseller** as **View name**.

21. Select **OK** to save the view.

Save as view

This will create a view from the visual query with Enable load applied. To use a different visual query, cancel this dialog and apply Enable load to the visual query you want to use.

Warehouse

lh\_FAIAD

Schema

dbo

View name \*

Reseller

SQL for view

```

CREATE VIEW [dbo].[Reseller]
AS
select [$Outer].[ResellerID] as [ResellerID],
[$Outer].[ResellerName] as [ResellerName],
[$Outer].[PostalCityID] as [PostalCityID],
[$Outer].[PhoneNumber] as [PhoneNumber],
[$Outer].[FaxNumber] as [FaxNumber],
[$Outer].[WebsiteURL] as [WebsiteURL],
[$Outer].[DeliveryAddressLine1] as [DeliveryAddressLine1],
[$Outer].[DeliveryAddressLine2] as [DeliveryAddressLine2],
[$Outer].[DeliveryPostalCode] as [DeliveryPostalCode]

```

Copy to Clipboard

OK Cancel

You will get an alert once the view is saved.

22. In the Explorer (left) panel, expand **Views**. We have the newly created Reseller view.

Explorer

Warehouses

lh\_FAIAD

Schemas

dbo

Tables

BuyingGroups

Cities

Countries

Customers

Date

InvoiceLine1...

Invoices

ProductGrou...

ProductItem

ProductItem...

States

Views

Geo

Reseller

Visual query 1

Visual

Manage columns

Reduce row

BuyingGroups

Source

Database

Download Excel file

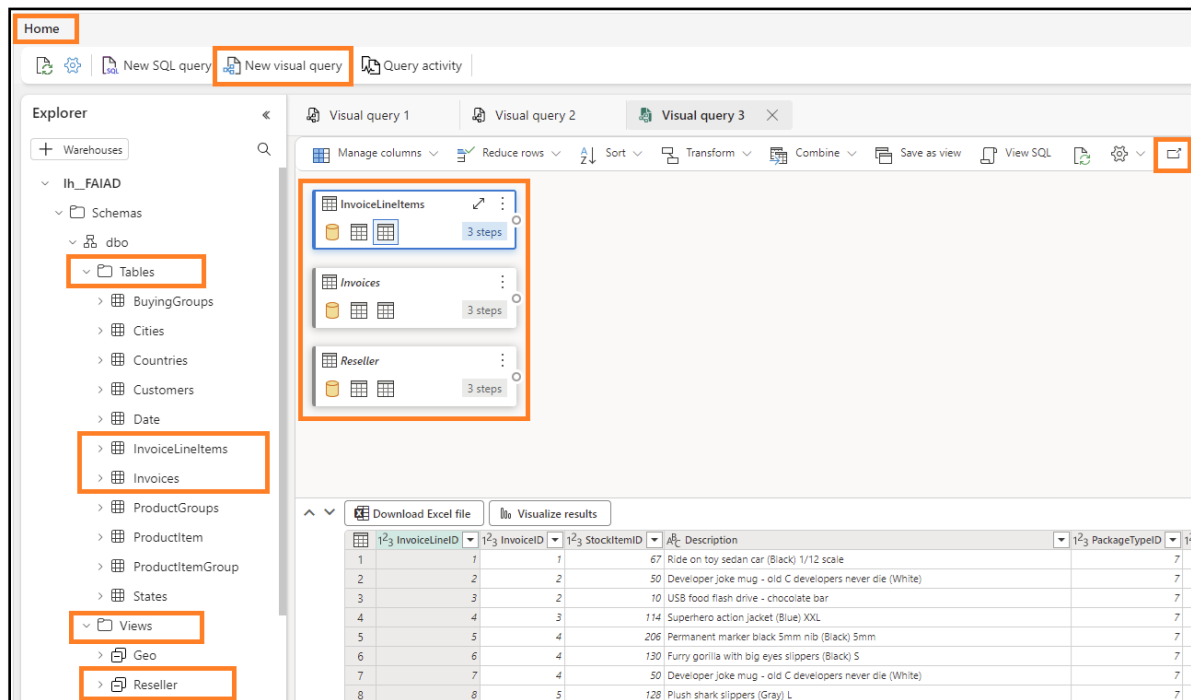
Vis

	eNumber	FaxNumber
1	-0100	(308) 555-0101
2	-0100	(406) 555-0101
3	-0100	(480) 555-0101
4	-0100	(316) 555-0101
5	-0100	(212) 555-0101
6	-0100	(701) 555-0101
7	-0100	(423) 555-0101
8	-0100	(303) 555-0101
9	-0100	(201) 555-0101
10	-0100	(701) 555-0101
11	-0100	(215) 555-0101
12	-0100	(218) 555-0101
13	-0100	(217) 555-0101

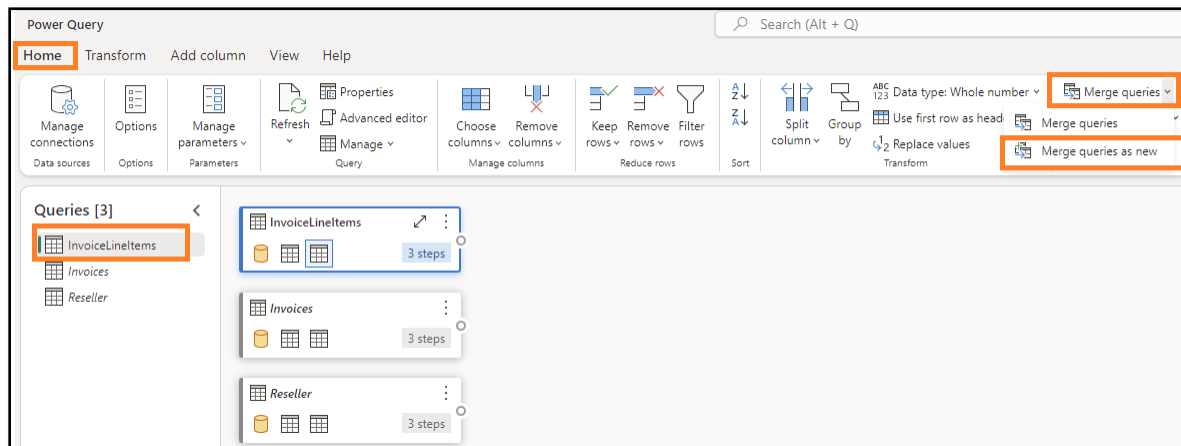
## Task 4: Create Sales view using Visual query

Let's create Sales view, which is created by merging InvoiceLineItems and Invoices table and Reseller view. We have this query in Power BI Desktop. We will copy the code from Advanced Editor. But before copying the code, we need to create a merge table using Visual query as creating a blank query is not possible in Visual query. Let's give this method a try.

1. From the Lakehouse menu bar, select **Home** -> **New visual query**. A new visual query opens.
2. From **Explorer** -> **Table** section, drag **InvoiceLineItems**, **Invoices** tables to the visual query section
3. From **Explorer** -> **Views** section, drag **Reseller** view to the visual query section
4. From the Visual query editor, select the **Focus mode icon** to open Power Query editor.



5. With InvoiceLineItems query selected, from the ribbon select **Home** -> **Merge queries** -> **Merge queries as new**.



Merge dialog opens.

6. In the **left table to merge**, select **InvoiceLineItems**.
7. In the **right table to merge**, select **Invoices**.
8. Select **InvoiceID** columns from both the tables. We are going to join using this column.
9. Select **Inner** as the **Join kind**.
10. Select **OK**.

Merge ?

Select tables and matching columns to create a merged table.

Left table for merge \*

InvoiceLineItems

1 <sup>2</sup> 3 InvoiceLineID	1 <sup>2</sup> 3 InvoiceID	1 <sup>2</sup> 3 StockItemID	1 <sup>2</sup> 3 Description
1	1	67	Ride on toy sedan car (Black) 1/12 scale
2	2	50	Developer joke mug - old C developers never die (White)
3	2	10	USB food flash drive - chocolate bar
4	3	114	Superhero action jacket (Blue) XXL

Right table for merge \*

Invoices

1 <sup>2</sup> 3 InvoiceID	1 <sup>2</sup> 3 CustomerID	1 <sup>2</sup> 3 BillToResellerID	1 <sup>2</sup> 3 OrderID	1 <sup>2</sup> 3 DeliveryMethodID	1 <sup>2</sup> 3 ContactP
68	972	972	68	3	
1499	893	899	1525	3	
3371	981	981	3022	3	
4832	988	988	3801	3	

Join kind

Left outer Right outer Full outer Inner Left anti Right anti

☐ Use fuzzy matching to perform the merge

> Fuzzy matching options

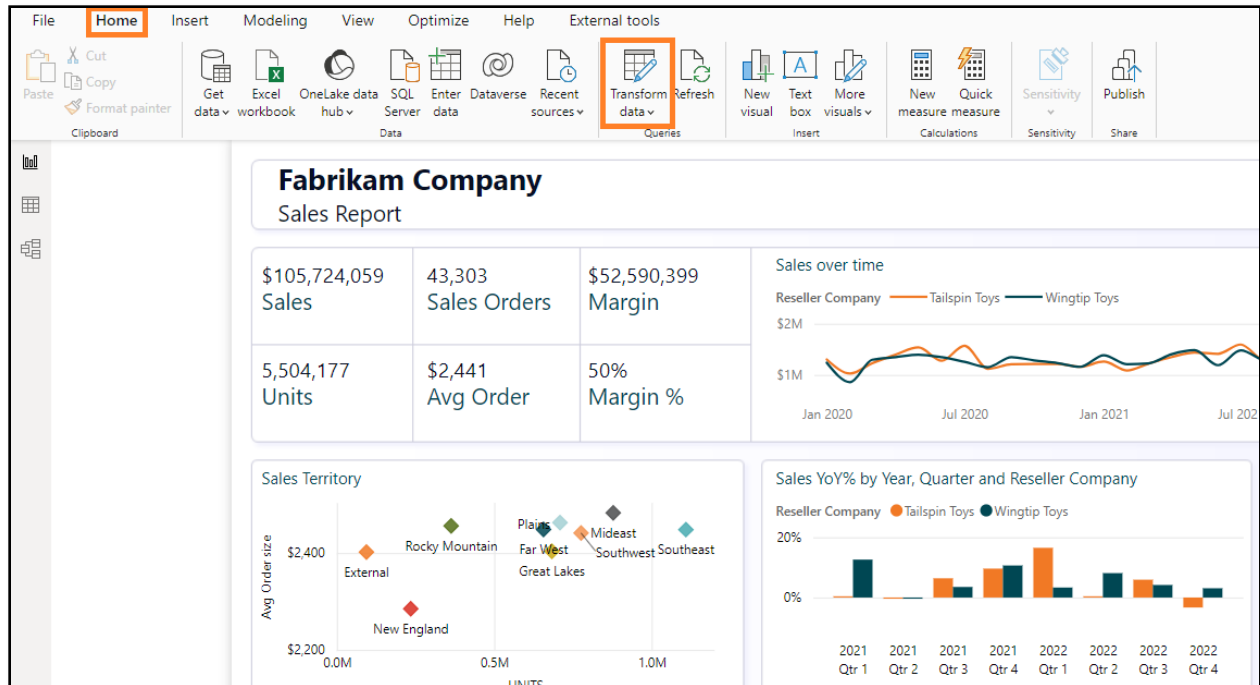
✓ The selection matches 221,914 rows from both the tables

OK Cancel

We are going to copy code from Power BI Desktop and paste it using Advanced Editor.

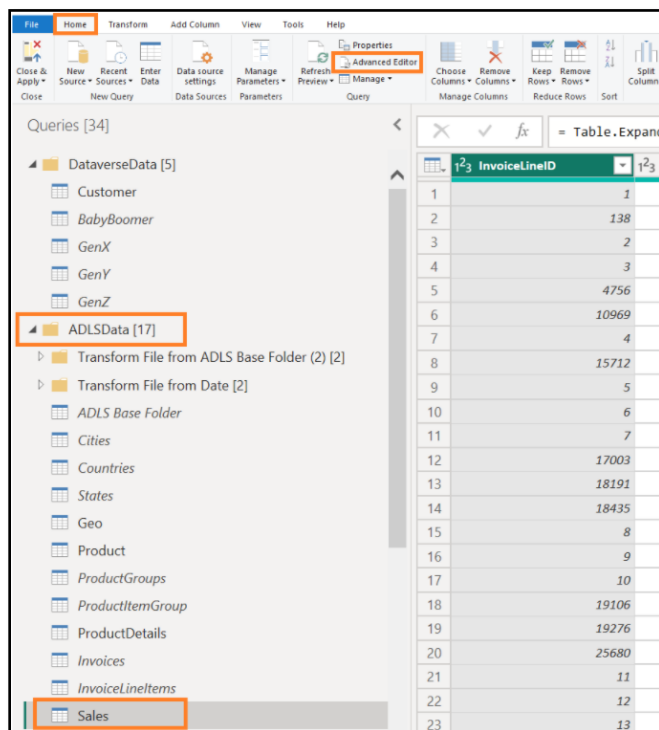
11. If you have not already opened it, open **FAIAD.pbix** located in the **Reports** folder on the desktop of your lab environment.
12. From the ribbon select **Home -> Transform data**. Power Query window opens. As you have noticed in the earlier lab, queries in the left panel are organized by data source.





13. From the left panel, under the ADLSData folder, select **Sales** query.

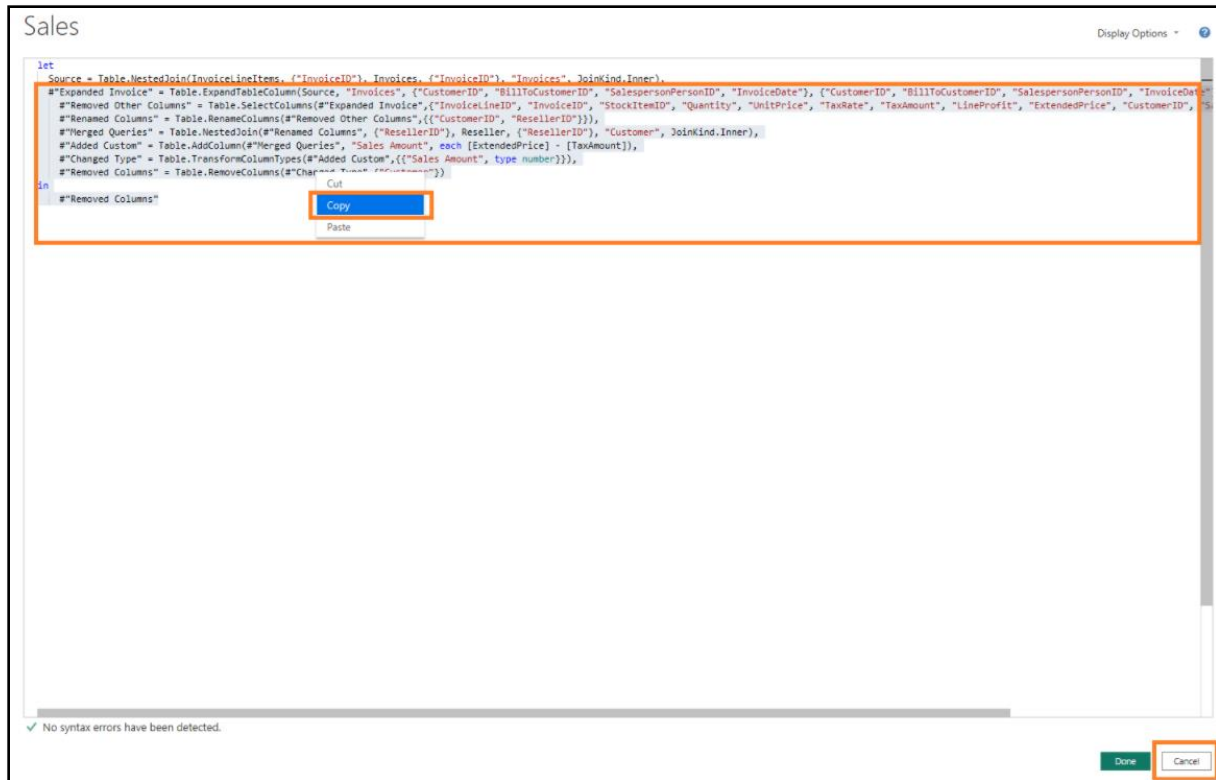
14. From the ribbon select **Home - > Advanced Editor**. Advanced Editor dialog opens.



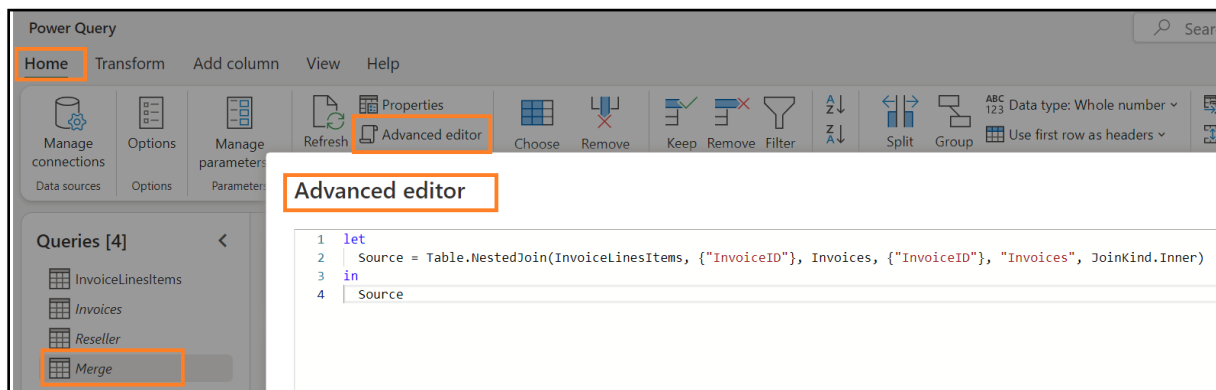
15. Select code from Line 3 ("Expanded Invoice" ...) all the way through to the last line of code.

16. Right click and select **Copy**.

17. Select **Cancel** to close Advanced Editor.

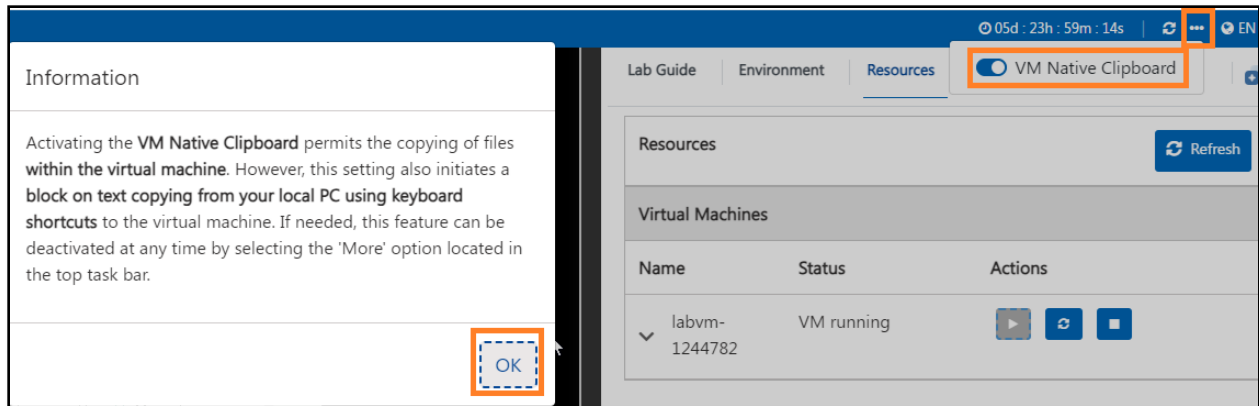


18. **Navigate back to the browser** where you have the Power Query Editor open.
19. Make sure you have **Merge** query selected.
20. From the ribbon select **Home -> Advanced Editor**. Advanced Editor dialog opens.

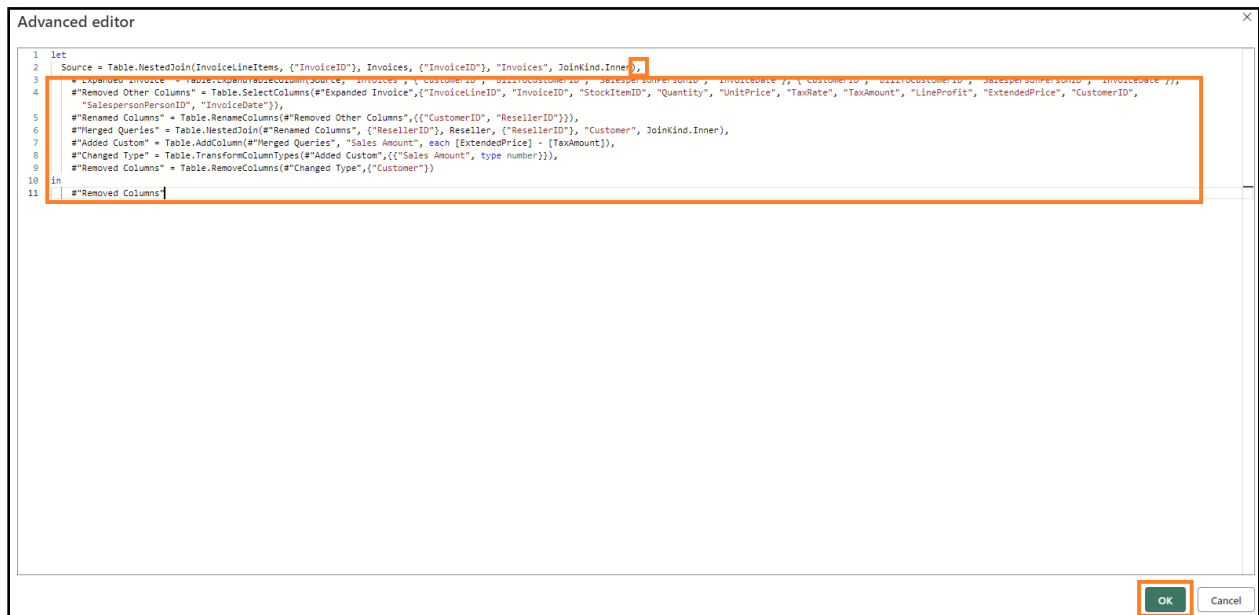


21. At the **end of line 2 add a comma** (Source = Table.NestedJoin(InvoiceLineItems, {"InvoiceID"}, Invoices, {"InvoiceID"}, "Invoices", JoinKind.Inner)
22. Click **Enter** to start a new line.
23. Enter **Ctrl+V** on your keyboard to paste the code you copied from Power BI Desktop.

**Note:** If you are working in the lab environment, please select the ellipsis on the top right of the screen. Use the slider to **enable VM Native Clipboard**. Select OK in the dialog. Once done pasting the queries you can disable this option.



24. Highlight the last two lines of code ( in Source) and **delete** it.
25. Select **OK** to save the changes.



If it is easier, delete all the code in the Advanced Editor and paste the below code into Advanced Editor.

```

let
    Source = Table.NestedJoin(InvoiceLineItems, {"InvoiceID"}, Invoices, {"InvoiceID"}, "Invoices", JoinKind.Inner),
    #"Expanded Invoice" = Table.ExpandTableColumn(Source, "Invoices", {"CustomerID", "BillToCustomerID", "SalespersonPersonID", "InvoiceDate"}, {"CustomerID", "BillToCustomerID", "SalespersonPersonID", "InvoiceDate"}),
    #"Removed Other Columns" = Table.SelectColumns(#"Expanded Invoice", {"InvoiceLineID", "InvoiceID", "StockItemID", "Quantity", "UnitPrice", "TaxRate", "TaxAmount", "LineProfit", "ExtendedPrice", "CustomerID", "SalespersonPersonID", "InvoiceDate"}),
    #"Renamed Columns" = Table.RenameColumns(#"Removed Other Columns", {"CustomerID", "ResellerID"}),

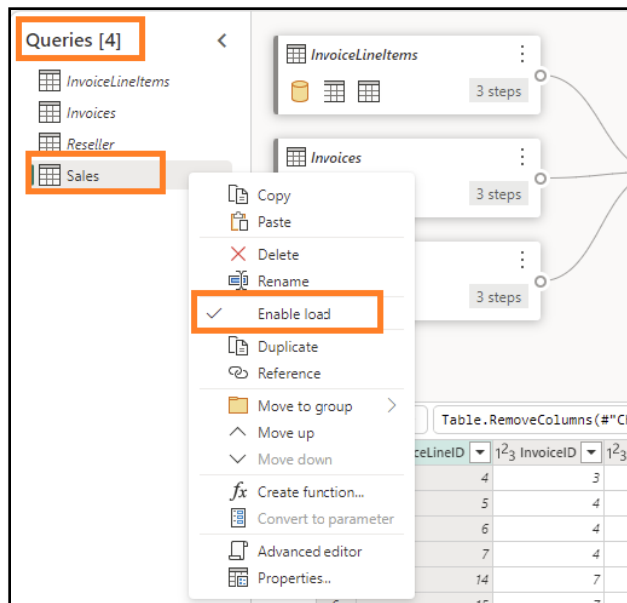
```

```

#"Merged Queries" = Table.NestedJoin(#"Renamed Columns", {"ResellerID"}, Reseller,
{"ResellerID"}, "Customer", JoinKind.Inner),
#"Added Custom" = Table.AddColumn(#"Merged Queries", "Sales Amount", each
[ExtendedPrice] - [TaxAmount]),
#"Changed Type" = Table.TransformColumnTypes(#"Added Custom",{{"Sales Amount", type
number}}),
#"Removed Columns" = Table.RemoveColumns(#"Changed Type",{"Customer"})
in
#"Removed Columns"

```

26. You will be navigated back to Power Query Editor. In the left, Queries panel, **double click on Merge** query to rename it.
27. **Rename** Merge query to **Sales**.
28. Right click on Sales query and select **Enable load** to enable the query to be loaded.



29. Select **Save** to Save and Close the Power Query dialog. You will be navigated to Visual query.
30. From the Visual query menu, select **Save as view**. Save as view dialog opens. Notice the SQL query is available. You can review it, if you choose it.
31. Enter **Sales** as **View name**.
32. Select **OK** to save the view.

Save as view

● This will create a view from the visual query with Enable load applied. To use a different visual query, cancel this dialog and apply Enable load to the visual query you want to use.

Warehouse  
lh\_FAIAD

Schema  
dbo

View name \*  
Sales

SQL for view

```

CREATE VIEW [dbo].[Sales]
AS
select [$Outer].[InvoiceLineID] as [InvoiceLineID],
[$Outer].[InvoiceID] as [InvoiceID],
[$Outer].[StockItemID] as [StockItemID],
[$Outer].[Quantity] as [Quantity],
[$Outer].[UnitPrice] as [UnitPrice],
[$Outer].[TaxRate] as [TaxRate],
[$Outer].[TaxAmount] as [TaxAmount],
[$Outer].[LineProfit] as [LineProfit],
[$Outer].[ExtendedPrice] as [ExtendedPrice]

```

Copy to Clipboard

OK Cancel

You will get an alert once the view is saved.

33. In the Explorer (left) panel, expand **Views**. We have the newly created Sales view.

Home

New SQL query
New visual query
Query activity

Explorer

Warehouses

lh\_FAIAD

Schemas

dbo

Tables

BuyingGroups

Cities

Countries

Customers

Date

InvoiceLineItems

Invoices

ProductGroups

ProductItem

ProductItemGroup

States

Views

Geo

Reseller

Sales

Visual query 1

Visual query 2

Visual query 3

Manage columns
Reduce rows
Sort
Transform
Combine

InvoiceLineItems

Invoices

Reseller

Sales

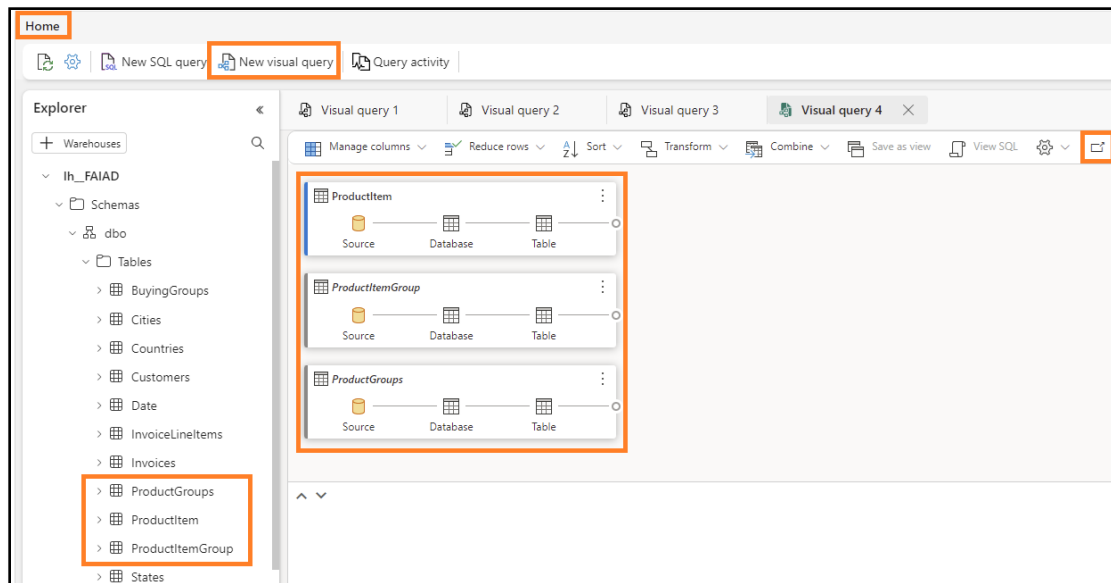
Download Excel file
Visualize results

	123 InvoiceLineID	123 InvoiceID	123 StockItemID	123 Quantity	1,2 UnitPrice	123 TaxRate
1	4	3	114	3	30	
2	5	4	206	96	2,7	
3	6	4	130	5	32	
4	7	4	50	2	13	
5	14	7	39	9	13	
6	15	7	184	75	3,5	
7	16	7	0	0	2,1	

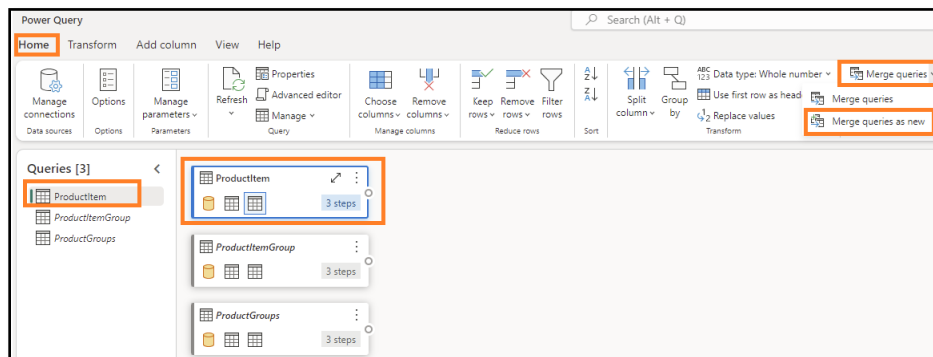
## Task 5: Create Product view using Visual query

Let's create Product view, which is created by merging ProductItem, ProductItemGroup and ProductGroups tables. To move things along, we are going to copy code into Advanced Editor.

1. From the Lakehouse menu bar, select **Home -> New visual query**. A new visual query opens.
2. From Explorer section, drag **ProductItem**, **ProductItemGroup** and **ProductGroups** tables to the visual query section
3. From the Visual query editor, select the **Focus mode icon** to open Power Query editor.



4. With **ProductItem** query selected, from the ribbon select **Home -> Merge queries -> Merge queries as new**. Merge dialog opens.



5. In the **left table to merge**, select **ProductItem**.
6. In the **right table to merge**, select **ProductItemGroup**.
7. Select **StockItemID** columns from both the tables. We are going to join using this column.
8. Select **Left outer** as the **Join kind**.
9. Select **OK**. New Merge query is created.

**Merge** ⓘ

Select tables and matching columns to create a merged table.

Left table for merge: ProductItem

1^2_3 StockItemID	1^2_3 StockItemName	1^2_3 SupplierID	1,2 ColorID	1^2_3 Uni
1	USB missile launcher (Green)		12	null
2	USB rocket launcher (Gray)		12	12
3	Office cube periscope (Black)		12	3
4	USB food flash drive - sushi roll		12	null

Right table for merge: ProductItemGroup

1^2_3 StockItemStockGroupID	1^2_3 StockItemID	1^2_3 StockGroupID	1^2_3 LastEditedBy	1^2_3 LastEditedWhen
1	1	6	1	1/1/2013, 12:00:00
2	1	1	1	1/1/2013, 12:00:00
3	1	7	1	1/1/2013, 12:00:00
4	2	6	1	1/1/2013, 12:00:00

Join kind: Left outer

☐ Use fuzzy matching to perform the merge

> Fuzzy matching options

✓ The selection matches 227 of 227 rows from the first table

OK Cancel

10. With Merge query selected, from the ribbon, select **Home** -> **Advanced editor**. Advanced editor dialog opens.

Power Query

Home Transform Add column View Help

Manage connections Options Manage parameters Refresh Properties Advanced editor Choose Remove Keep Remove Filter Split Group Data type: Whole number v Merge queries v Append queries v

Queries [4]

- ProductItem
- ProductItemGroup
- ProductGroups
- Merge

Advanced editor

```

1 let
2   Source = Table.NestedJoin(ProductItem, {"StockItemID"}, ProductItemGroup, {"StockItemID"}, "ProductItemGroup", JoinKind.LeftOuter),
3   in
4   Source

```

11. **Select all the code** in Advanced editor and **delete** it.

12. **Paste** the below code into Advanced editor.

```

let
    Source = Table.NestedJoin(ProductItem, {"StockItemID"}, ProductItemGroup, {"StockItemID"}, "ProductItemGroup", JoinKind.LeftOuter),
    #"Expanded ProductItemGroup" = Table.ExpandTableColumn(Source, "ProductItemGroup", {"StockGroupID"}, {"StockGroupID"}),
    #"Merged queries" = Table.NestedJoin(#"Expanded ProductItemGroup", {"StockGroupID"}, ProductGroups, {"StockGroupID"}, "ProductGroups", JoinKind.LeftOuter),
    #"Expanded ProductGroups" = Table.ExpandTableColumn(#"Merged queries", "ProductGroups", {"StockGroupName"}, {"StockGroupName"}),

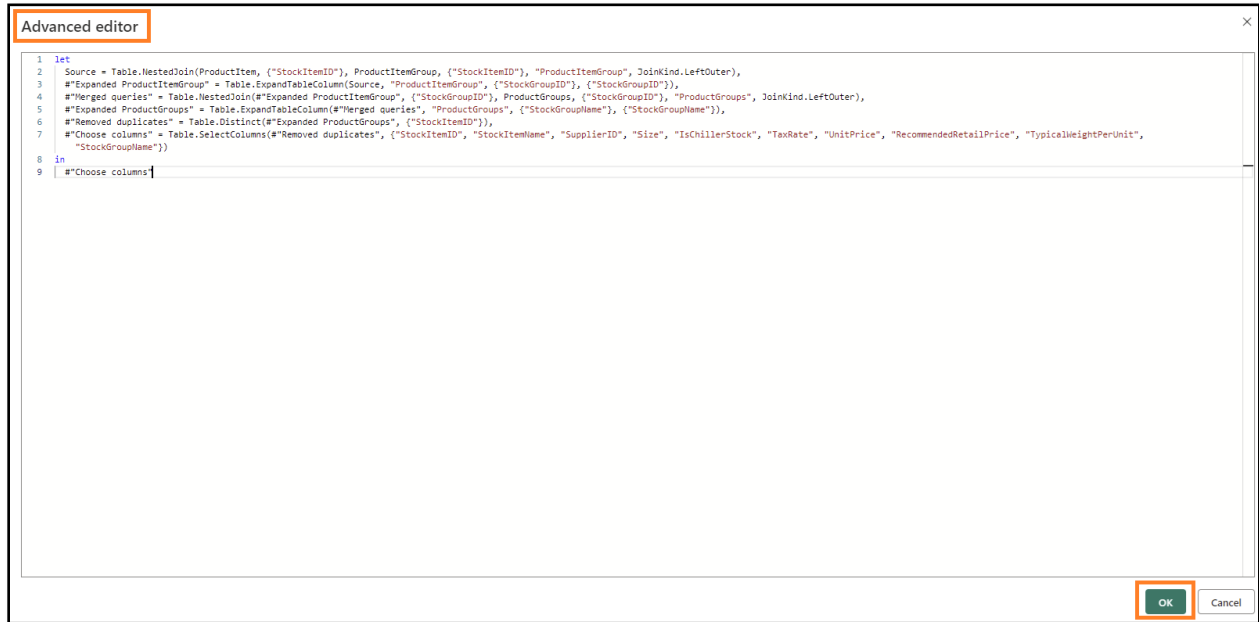
```

```

#"Choose columns" = Table.SelectColumns("#Expanded ProductGroups", {"StockItemID",
"StockItemName", "SupplierID", "Size", "IsChillerStock", "TaxRate", "UnitPrice",
"RecommendedRetailPrice", "TypicalWeightPerUnit", "StockGroupName"})
in
#"Choose columns"

```

13. Select **OK** to close Advanced Editor. You will be navigated back to Power Query editor.

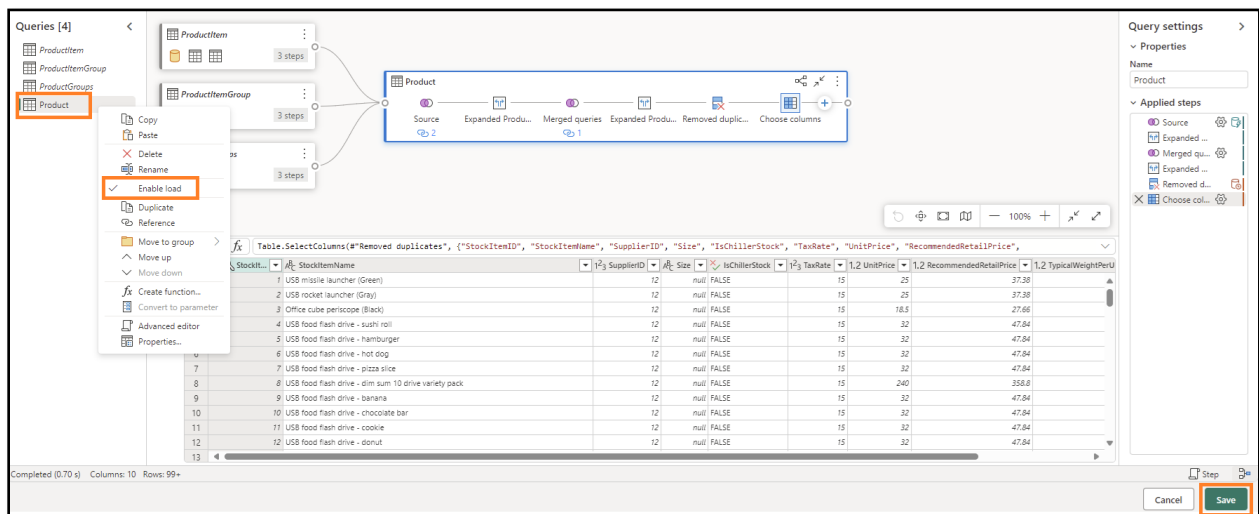


14. In the left, Queries panel, **double click on Merge** query to rename it.

15. **Rename Merge** query to **Product**.

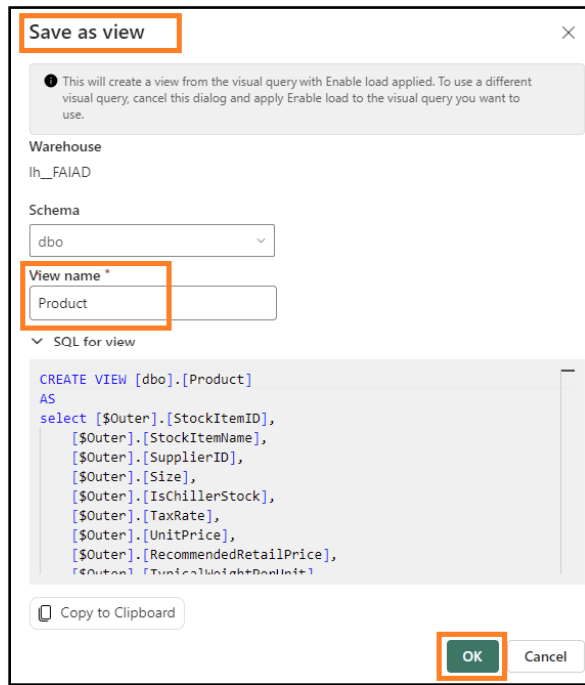
16. Right click on Product query and select **Enable load** to enable the query to be loaded.

17. Select **Save** to Save and Close the Power Query dialog. You will be navigated to Visual query.



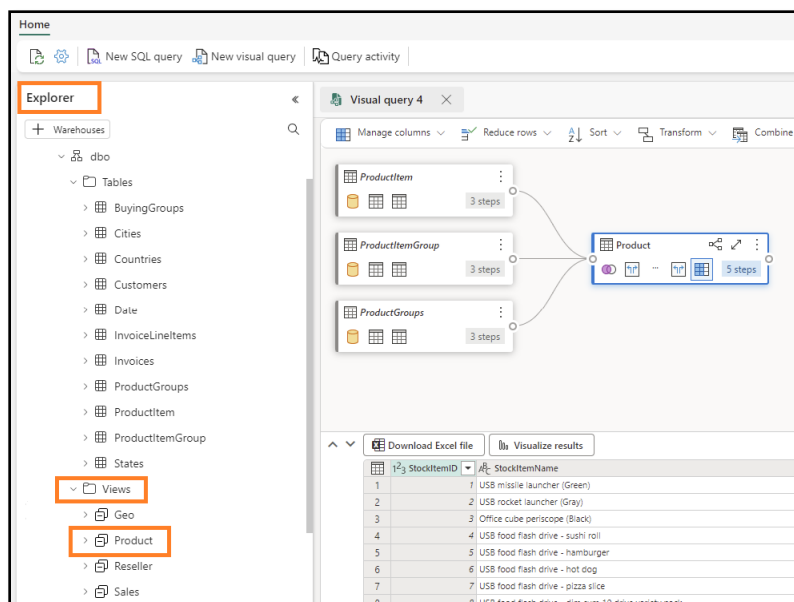


18. From the Visual query menu, select **Save as view**. Save as view dialog opens. Notice the SQL query is available. You can review it, if you choose it.
19. Enter **Product** as **View name**.
20. Select **OK** to save the view.



You will get an alert once the view is saved.

21. In the Explorer (left) panel, expand **Views**. We have the newly created Product view.

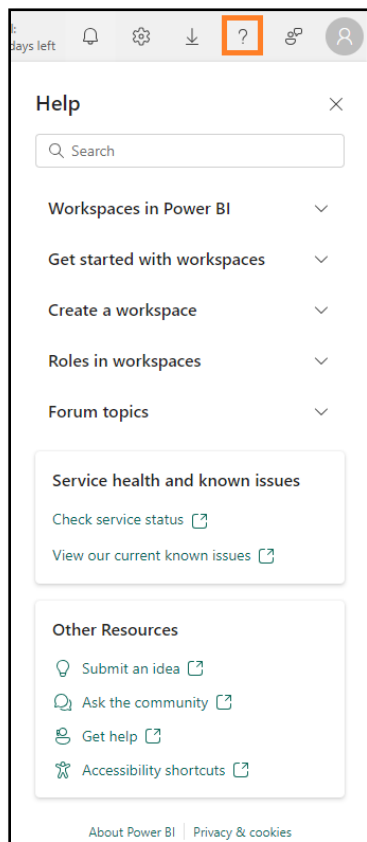


We have transformed the data from ADLS Gen2 data source. In this lab, we learned how to create shortcuts and explored various options for using visual query views to transform data.

In the next lab, we will learn how to use Dataflow Gen2 and create Shortcut to another Lakehouse.

## References

Fabric Analyst in a Day (FAIAD) introduces you to some of the key functions available in Microsoft Fabric. In the menu of the service, the Help (?) section has links to some great resources.



Here are a few more resources that will help you with your next steps with Microsoft Fabric.

- See blog post to read the full [Microsoft Fabric GA announcement](#)
- Explore Fabric through the [Guided Tour](#)
- Sign up for the [Microsoft Fabric free trial](#)
- Visit the [Microsoft Fabric website](#)
- Learn new skills by exploring the [Fabric Learning modules](#)
- Explore the [Fabric technical documentation](#)
- Read the [free e-book on getting started with Fabric](#)
- Join the [Fabric community](#) to post your questions, share your feedback, and learn from others

Read the more in-depth Fabric experience announcement blogs:

- [Data Factory experience in Fabric blog](#)
- [Synapse Data Engineering experience in Fabric blog](#)
- [Synapse Data Science experience in Fabric blog](#)
- [Synapse Data Warehousing experience in Fabric blog](#)
- [Synapse Real-Time Analytics experience in Fabric blog](#)
- [Power BI announcement blog](#)
- [Data Activator experience in Fabric blog](#)
- [Administration and governance in Fabric blog](#)
- [OneLake in Fabric blog](#)
- [Dataverse and Microsoft Fabric integration blog](#)

© 2023 Microsoft Corporation. All rights reserved.

By using this demo/lab, you agree to the following terms:

The technology/functionality described in this demo/lab is provided by Microsoft Corporation for purposes of obtaining your feedback and to provide you with a learning experience. You may only use the demo/lab to evaluate such technology features and functionality and provide feedback to Microsoft. You may not use it for any other purpose. You may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, or sell this demo/lab or any portion thereof.

COPYING OR REPRODUCTION OF THE DEMO/LAB (OR ANY PORTION OF IT) TO ANY OTHER SERVER OR LOCATION FOR FURTHER REPRODUCTION OR REDISTRIBUTION IS EXPRESSLY PROHIBITED.

THIS DEMO/LAB PROVIDES CERTAIN SOFTWARE TECHNOLOGY/PRODUCT FEATURES AND FUNCTIONALITY, INCLUDING POTENTIAL NEW FEATURES AND CONCEPTS, IN A SIMULATED ENVIRONMENT WITHOUT COMPLEX SET-UP OR INSTALLATION FOR THE PURPOSE DESCRIBED ABOVE. THE TECHNOLOGY/CONCEPTS REPRESENTED IN THIS DEMO/LAB MAY NOT REPRESENT FULL FEATURE FUNCTIONALITY AND MAY NOT WORK THE WAY A FINAL VERSION MAY WORK. WE ALSO MAY NOT RELEASE A FINAL VERSION OF SUCH FEATURES OR CONCEPTS. YOUR EXPERIENCE WITH USING SUCH FEATURES AND FUNCTIONALITY IN A PHYSICAL ENVIRONMENT MAY ALSO BE DIFFERENT.

**FEEDBACK.** If you give feedback about the technology features, functionality and/or concepts described in this demo/lab to Microsoft, you give to Microsoft, without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft software or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement.

MICROSOFT CORPORATION HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE DEMO/LAB, INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. MICROSOFT DOES NOT MAKE ANY ASSURANCES OR REPRESENTATIONS WITH REGARD TO THE ACCURACY OF THE RESULTS, OUTPUT THAT DERIVES FROM USE OF DEMO/ LAB, OR SUITABILITY OF THE INFORMATION CONTAINED IN THE DEMO/LAB FOR ANY PURPOSE.

#### **DISCLAIMER**

This demo/lab contains only a portion of new features and enhancements in Microsoft Power BI. Some of the features might change in future releases of the product. In this demo/lab, you will learn about some, but not all, new features.