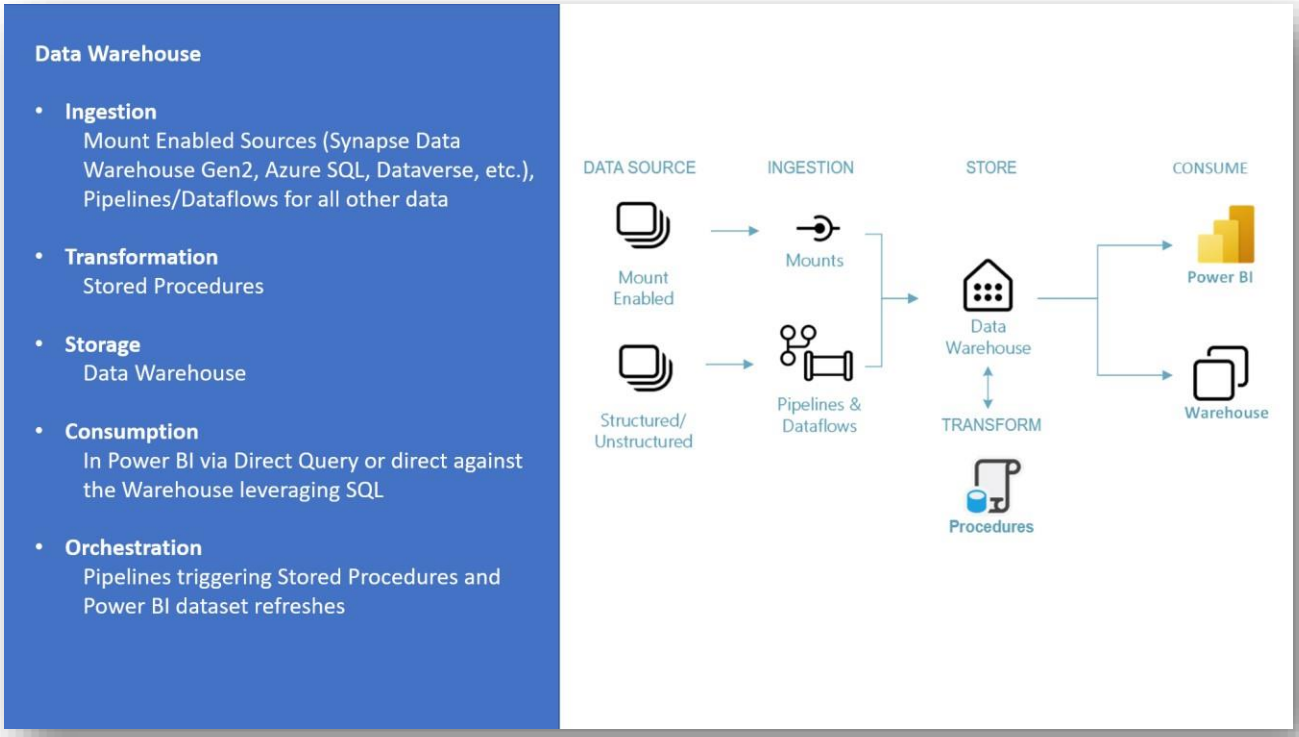
# Tutorial

## Data Warehouse

Published: June 2023



# Contents

# Introduction

**What is Fabric?**

Fabric provides a one-stop shop for all the analytical needs for every enterprise. It covers the complete spectrum of services including data movement, data lake, data engineering, data integration and data science, real time analytics, and business intelligence. With Fabric, there is no need to stitch together different services from multiple vendors. Instead, the customer enjoys an end-to-end, highly integrated, single comprehensive product that is easy to understand, onboard, create and operate. There is no other product on the market that offers the breadth, depth, and level of integration that Fabric offers. Additionally, Microsoft Purview is included by default in every tenant to meet compliance and governance needs.

To get an overview over the components and concepts of Fabric read Fabric - Overview and Concepts.

**Purpose of this tutorial**

While many concepts in Fabric may be familiar to data and analytics professionals it can be challenging to apply those concepts in a new environment. This tutorial has been designed to walk step-by-step through an end-to-end scenario from data acquisition to data consumption to build a basic understanding of the Fabric UX, the various workloads and their integration points, and the Fabric professional and citizen developer experiences.

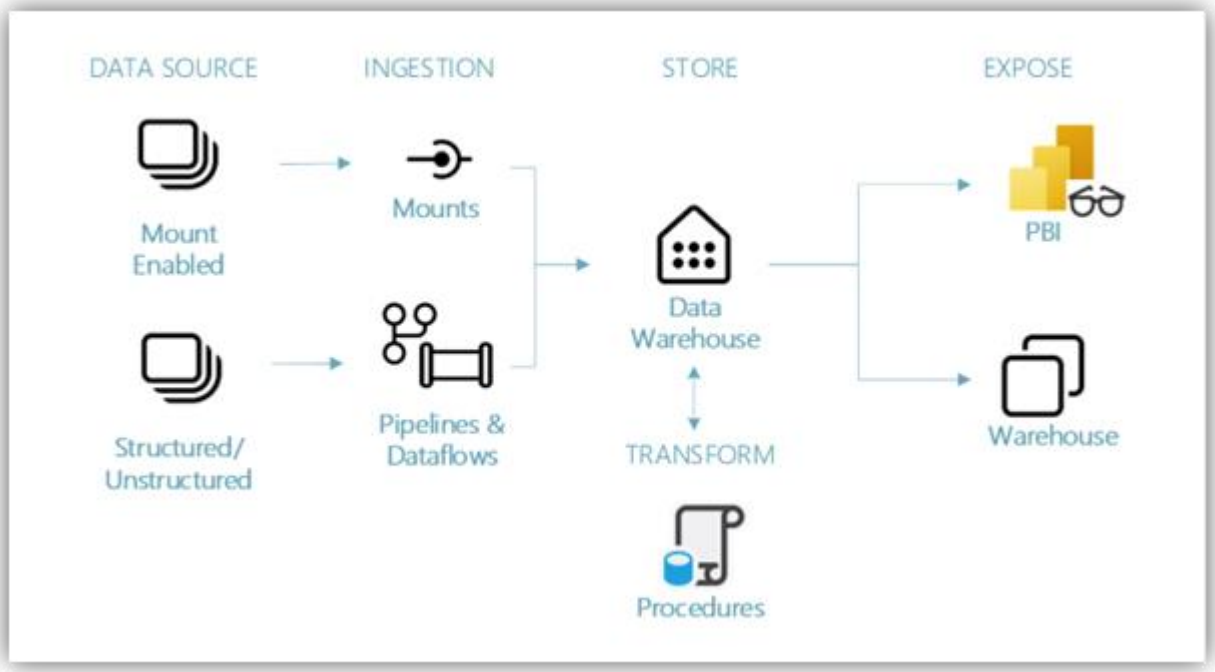
The tutorials are not intended to be a reference architecture, an exhaustive list of features and functionality, or a recommendation of specific best practices.

**The data warehouse tutorial**

In this tutorial, you will take on the role of a data warehouse developer at the fictional Wide World Importers company and complete the following steps:

- Sign into your Power BI online account , or if you don't have an account yet, sign up for a free trial.
- Build and implement an end-to-end data warehouse for your organization: ○ Enable Fabric in your tenant ○ Create a Fabric workspace ○ Quickly create a data warehouse
    - Ingest data from source to the data warehouse dimensional model ○ Transform the data to create aggregated datasets using T-SQL
    - Perform orchestration, data ingestion, and data transformation with pipelines ○ Query the data warehouse using T-SQL and a visual query editor ○ Create Power BI report using DirectLake mode to analyze the data in place
- Cleanup resources by deleting the workspace and other items

**The data warehouse end-to-end architecture**




**Data Sources** – Fabric makes it easy and quick to connect to Azure Data Services, other cloud platforms, and on-premises data sources to ingest data from.

**Ingestion** – With 200+ native connectors as part of the Fabric pipeline and with drag and drop data transformation with dataflow, you can quickly build insights for your organization. Shortcut is a new feature in Fabric that provides a way to connect to existing data without having to copy or move it – more details about Shortcut later in this tutorial.

**Transform and Store** – Fabric standardizes on Delta Lake format, that means all the engines of Fabric can read and work on the same dataset stored in OneLake – no need for data duplicity. This storage allows you to build a data warehouse or data mesh based on your organizational need. For transformation, you can choose either low-code or nocode experience with pipelines/dataflows or use T-SQL for a code first experience.

**Consume** – Data from the data warehouse can be consumed by Power BI, industry leading business intelligence tool, for reporting and visualization. Each data warehouse comes with a built-in TDS/SQL endpoint for easily connecting to and querying data from other reporting tools, when needed. When a data warehouse is created a secondary item, called a default dataset, will be automatically generated at the same time with the same name of the data warehouse to start visualizing data with just a couple of mouse clicks.

**The sample data**

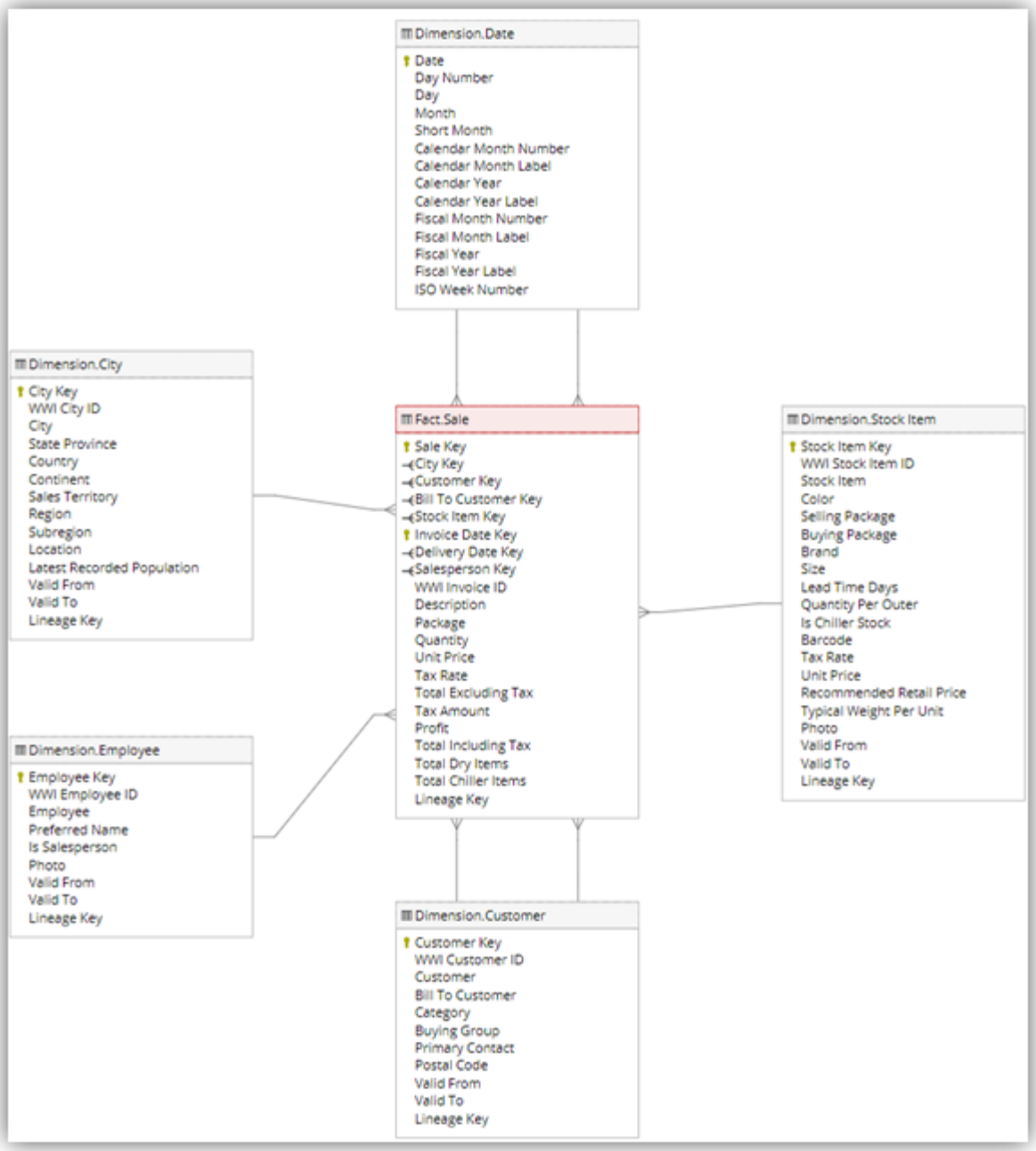
For sample data, we are going to use Wide World Importers (WWI) sample database. For our data warehouse end-to-end scenario, we have generated sufficient data for a sneak peek into the scale and performance capabilities of the Fabric platform.

Wide World Importers (WWI) is a wholesale novelty goods importer and distributor operating from the San Francisco Bay area. As a wholesaler, WWI's customers are mostly companies who resell to individuals. WWI sells to retail customers across the United States including specialty stores, supermarkets, computing stores, tourist attraction shops, and some individuals. WWI also sells to other wholesalers via a network of agents who promote the products on WWI's behalf. You can learn more about their company profile and operation here.

Typically, you would bring data from transactional systems (or line of business applications) into a data lake or data warehouse staging area, however for simplicity of this tutorial, we are going to use the dimensional model provided by WWI as our initial data source. We are going to use it as the source to ingest the data into a data warehouse and transform it through T-SQL.

**The data model**

While the WWI dimensional model contains multiple fact tables, for simplicity in explanation we will focus on the Sale Fact table and its related dimensions only, as below, to demonstrate this end-to-end data warehouse scenario:

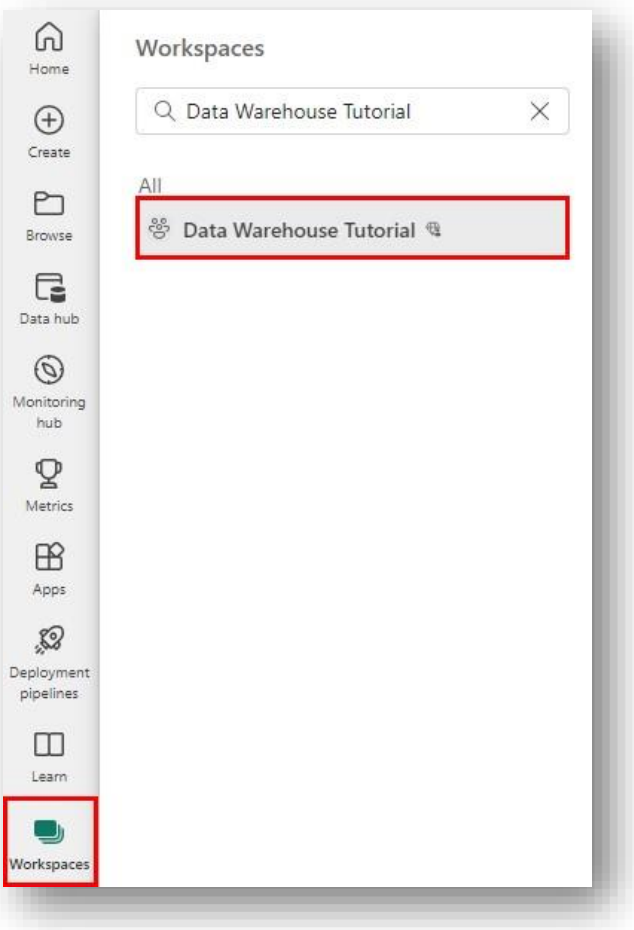


# Module 1: Build your first data warehouse

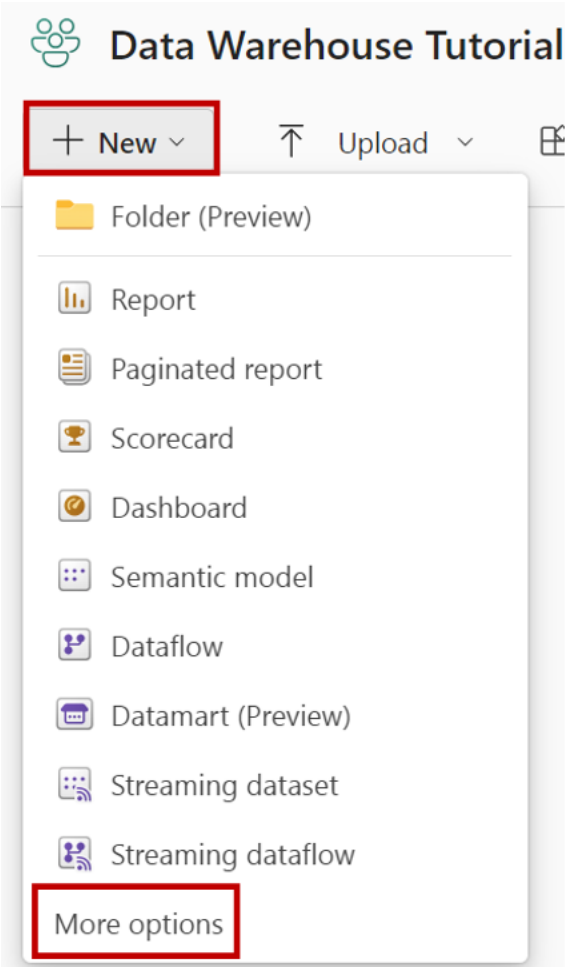
The intent of this module is to quickly build end to end journey of building a data warehouse, ingesting data for a table and then using the data warehouse for creating a report.

## Create a data warehouse

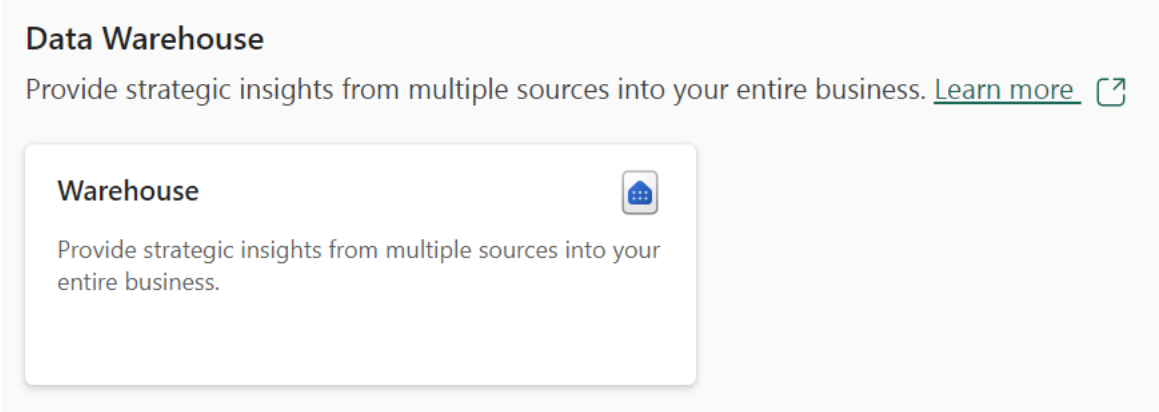
1. In the Power BI service select **Workspaces** in the left-hand menu.
2. Search for your workspace by typing in the search textbox at the top and click on your workspace to open it.



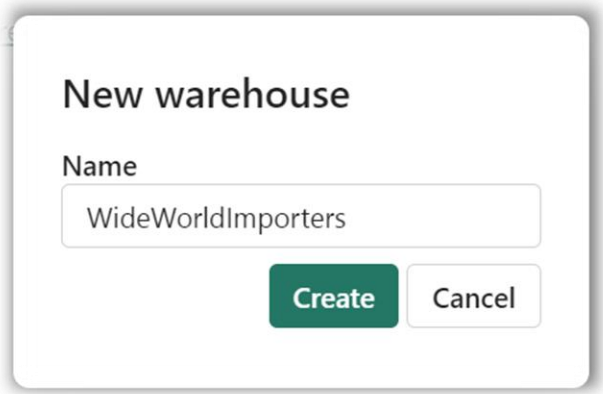
3. In the upper left corner, select **New > More Options** to display a full list of available items.



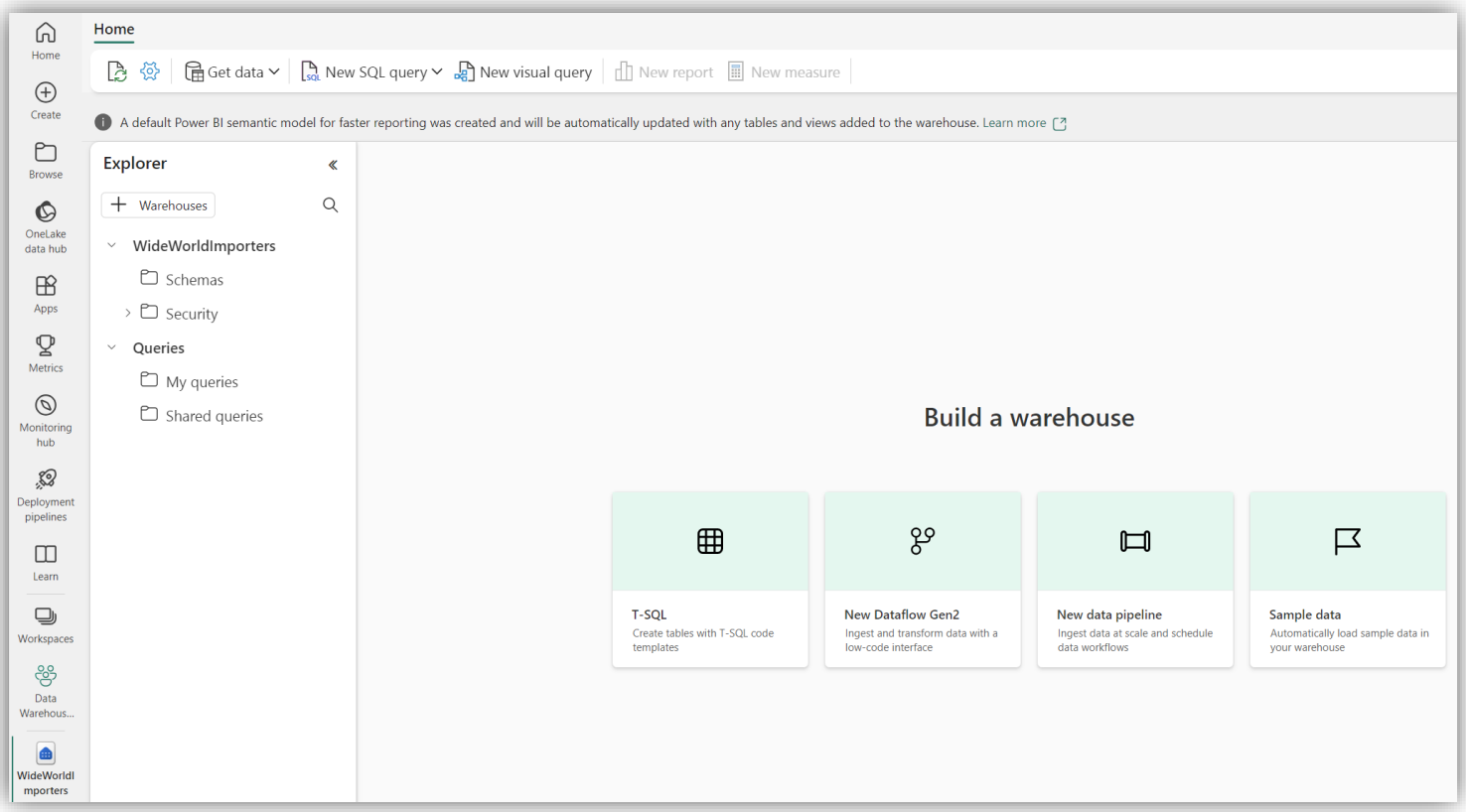

5. In the **Data warehouse** section, select **Warehouse**.




6. On the **New warehouse** dialog, enter **WideWorldImporters** as the name.
7. Select **Create**.

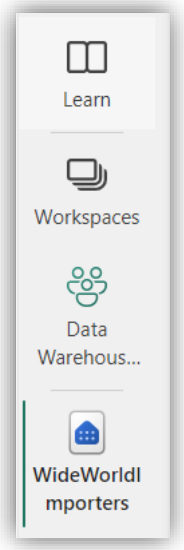



When provisioning is complete the **Build a warehouse** landing page will be shown.

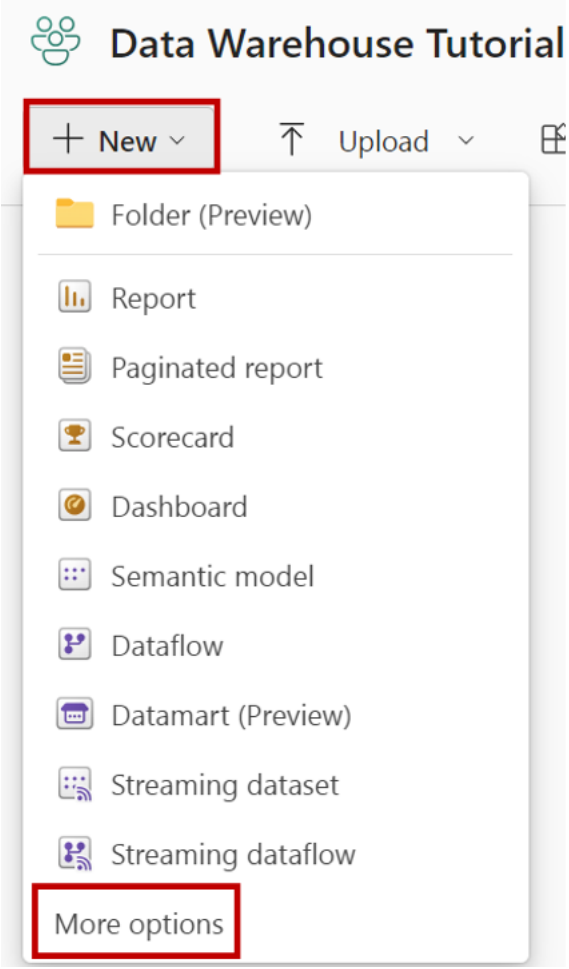



## Data ingestion

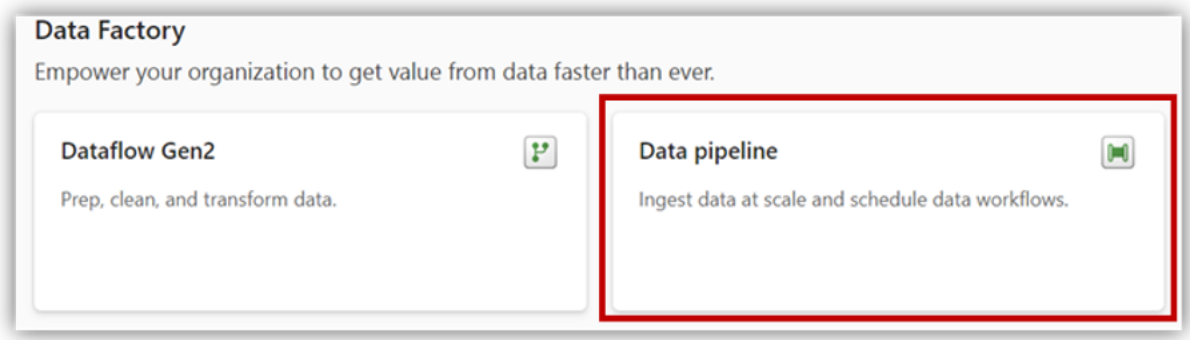
1. Select **Data Warehouse Tutorial** in the left-hand navigation menu to return to the workspace artifact view.



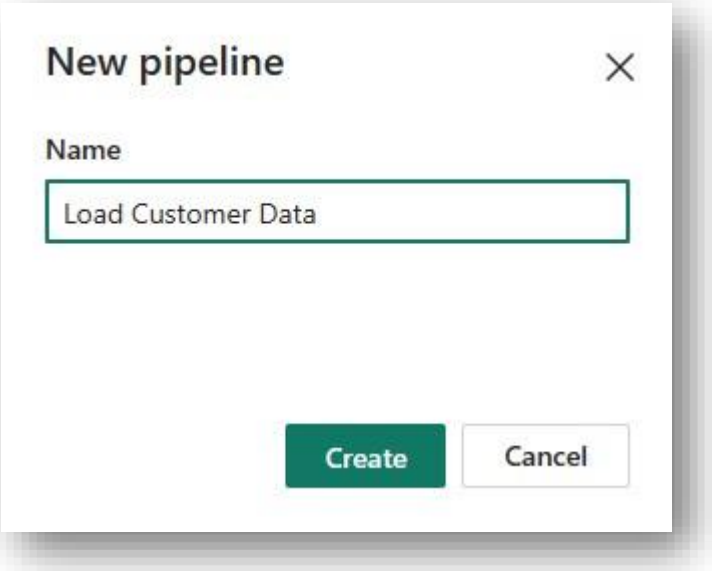

2. In the upper left corner, select **New > More Options** to display a full list of available items.



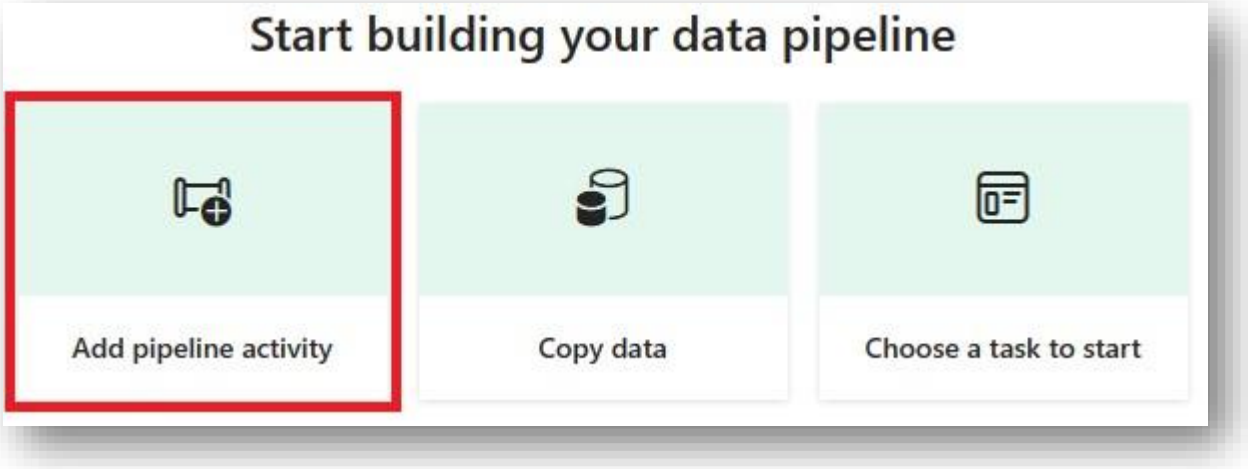

9. In the **Data Factory** section, select **Data pipeline**.



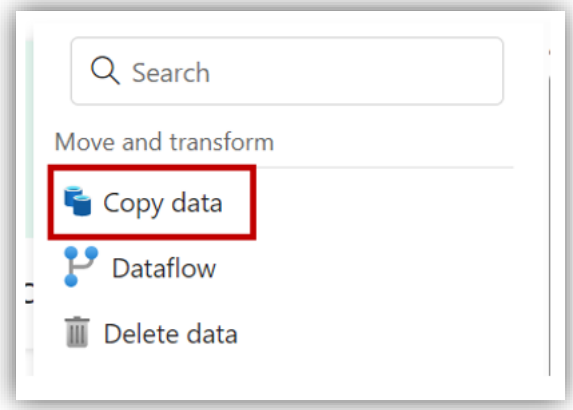

10. On the **New pipeline** dialog, enter **Load Customer Data** as the name.



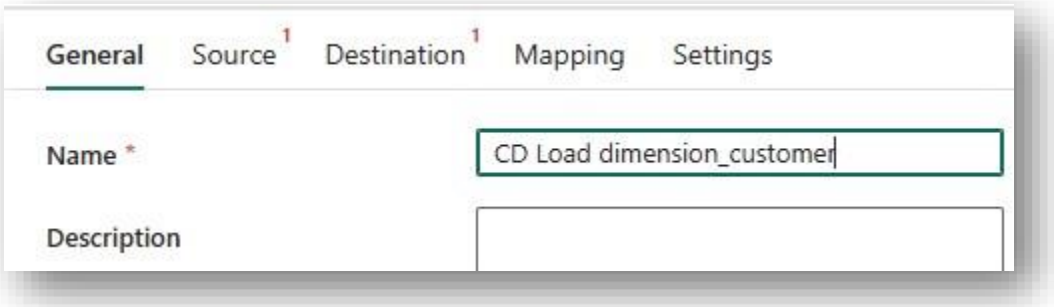

11. Select **Create**.
12. Select **Add pipeline activity** from the **Start building your data pipeline** landing page.



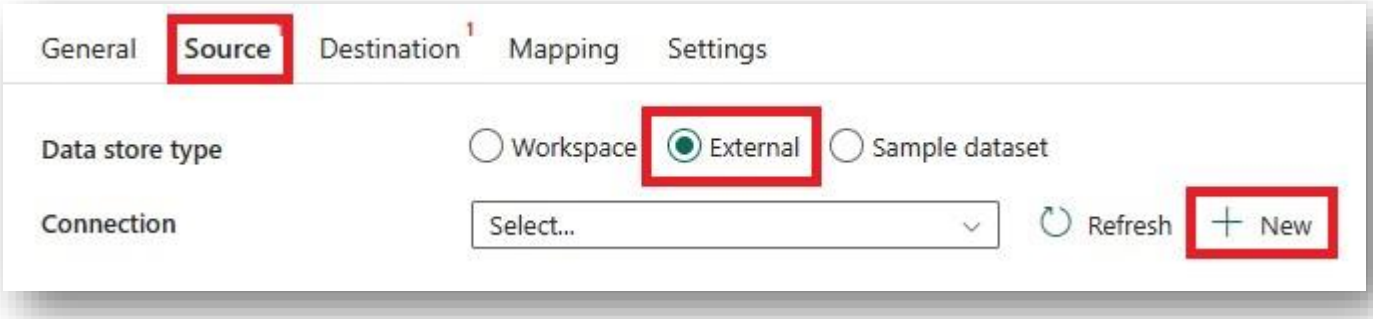

13. Select **Copy data** from the **Move & transform** section.



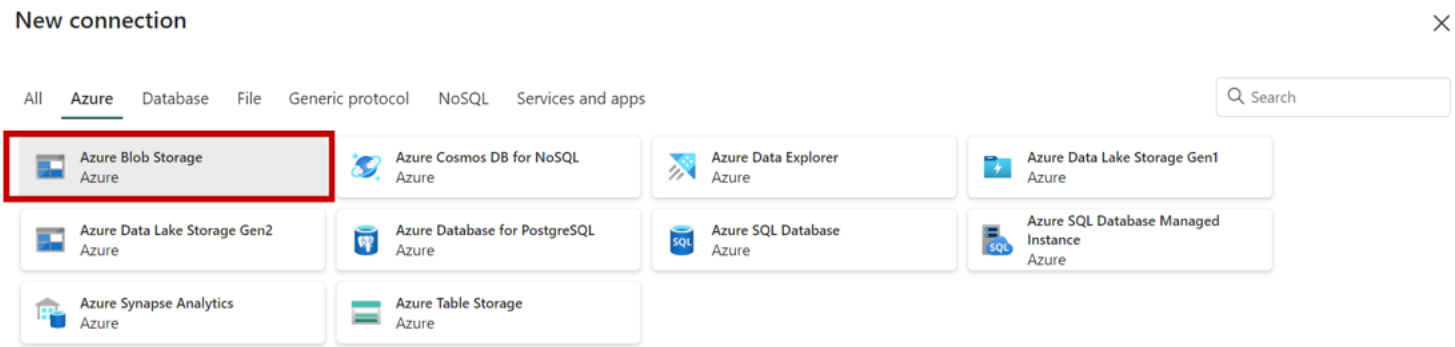

14. If necessary, select the newly created Copy data activity from the design canvas and follow the steps below to configure it.
15. On the **General** page, enter **CD Load dimension_customer** as the **Name**.



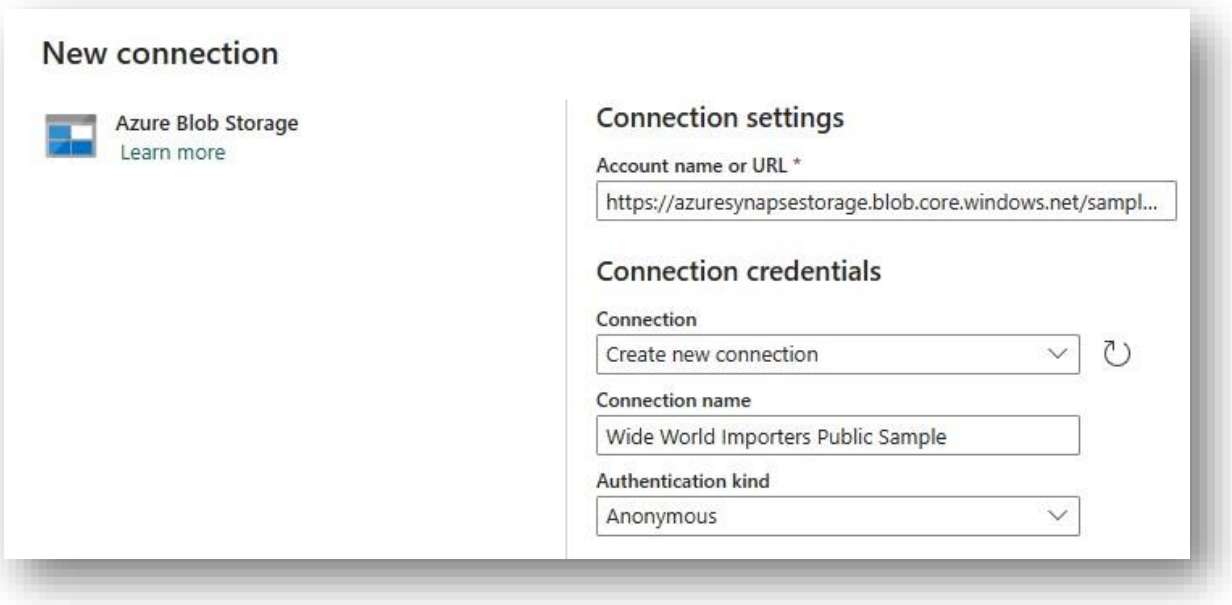

16. On the **Source** page select **External** for the **Data store type**.
17. Next to the **Connection** box, select **New** to create a new connection.



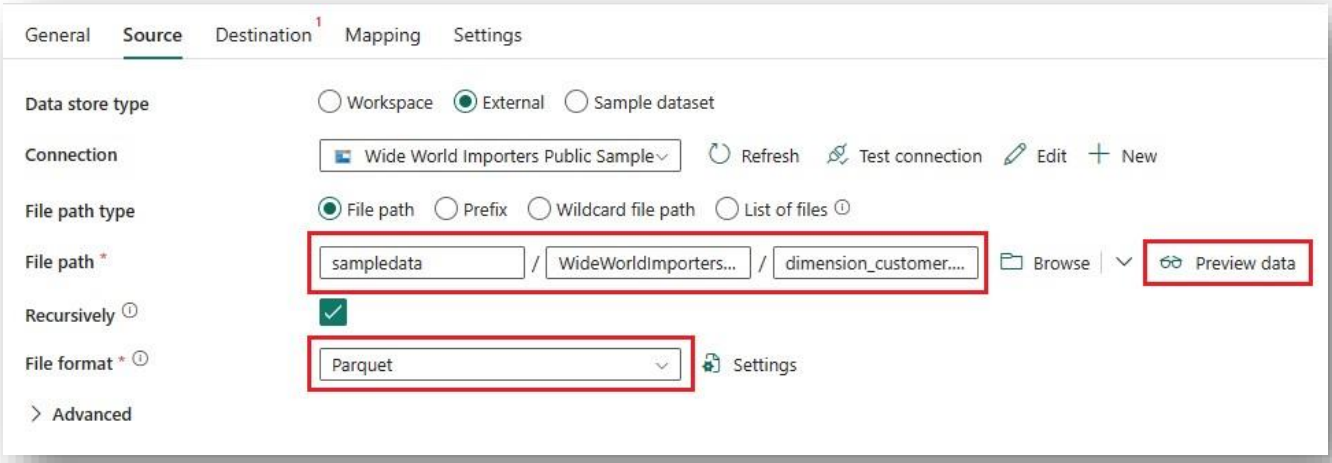

18. On the **New connection** page, select **Azure Blob Storage** from the list of connection options.



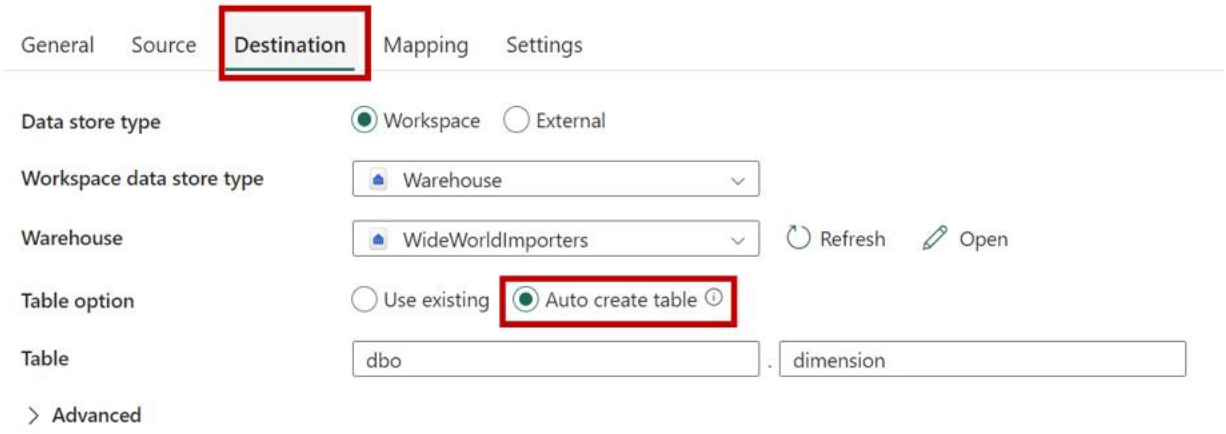

19. Select **Continue**.
20. On the **Connection settings** page, configure the settings as follows:
    1. Enter **https://azuresynapsestorage.blob.core.windows.net/sampledata** in the **Account name or URL**.
    2. In the **Connection credentials** section, select **Create new connection** in the dropdown for the **Connection**.
    3. Enter **Wide World Importers Public Sample** for the **Connection name**.
    4. Set the **Authentication kind** to **Anonymous**.



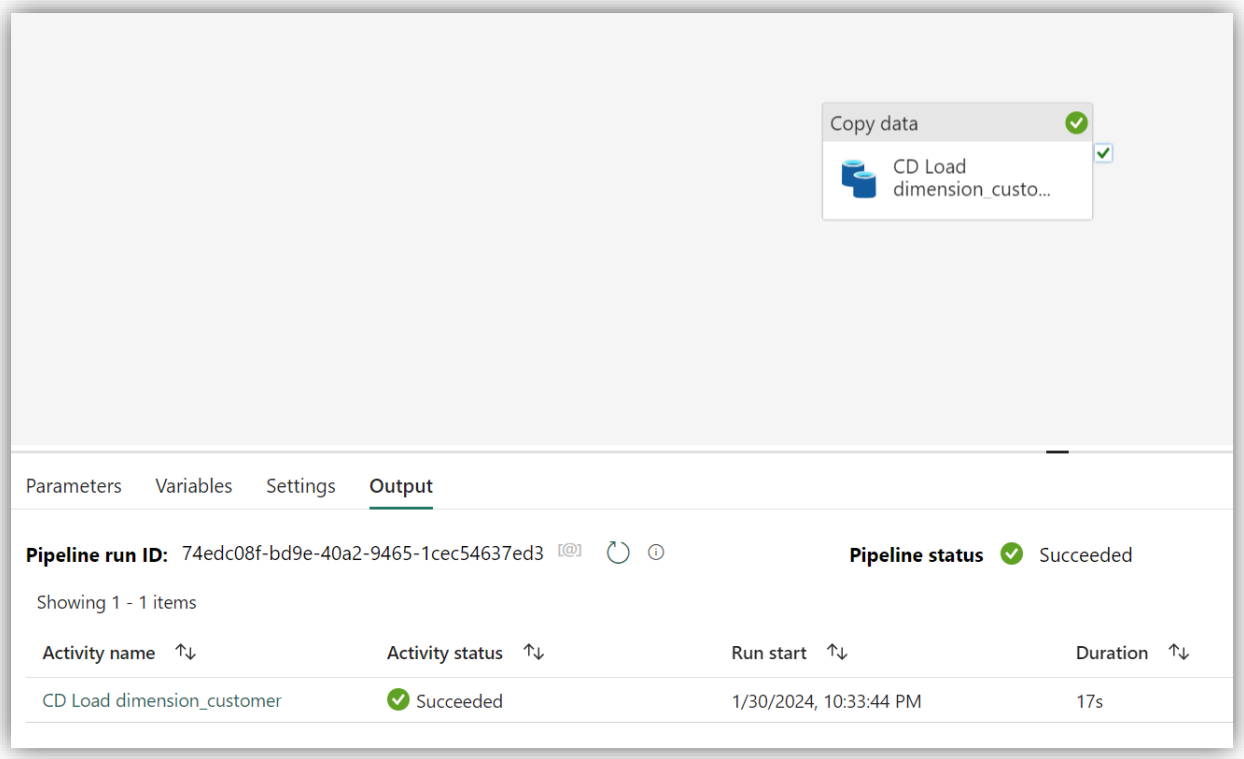

21. Click **Create**.
22. Change the remaining settings on the **Source** page of the copy activity as follows:
    1. **File path – Container:** sampledata
    2. **File path – Directory:** WideWorldImportersDW/tables
    3. **File path – File name:** dimension_customer.parquet
    4. **File format:** Parquet
23. Select **Preview data** next to the **File path** setting to ensure there are no errors.




24. On the **Destination** page, select **Workspace** for the **Data store type**.
25. Select **Warehouse** for the **Workspace data store type**.
26. In the **Warehouse** drop down, select **WideWorldImporters** from the list.
27. Next to the **Table option** setting, select **Auto create table**
28. In the first box (schema name) next to the **Table** setting, enter **dbo**.
29. In the second box(table name) next to the **Table** setting, enter **dimension_customer**.




30. From the ribbon, select **Run**.
31. Select **Save and run** from the dialog box. The pipeline to load the dimension_customer table with start.
32. Monitor the copy activity's progress on the **Output** page and wait for it to complete.

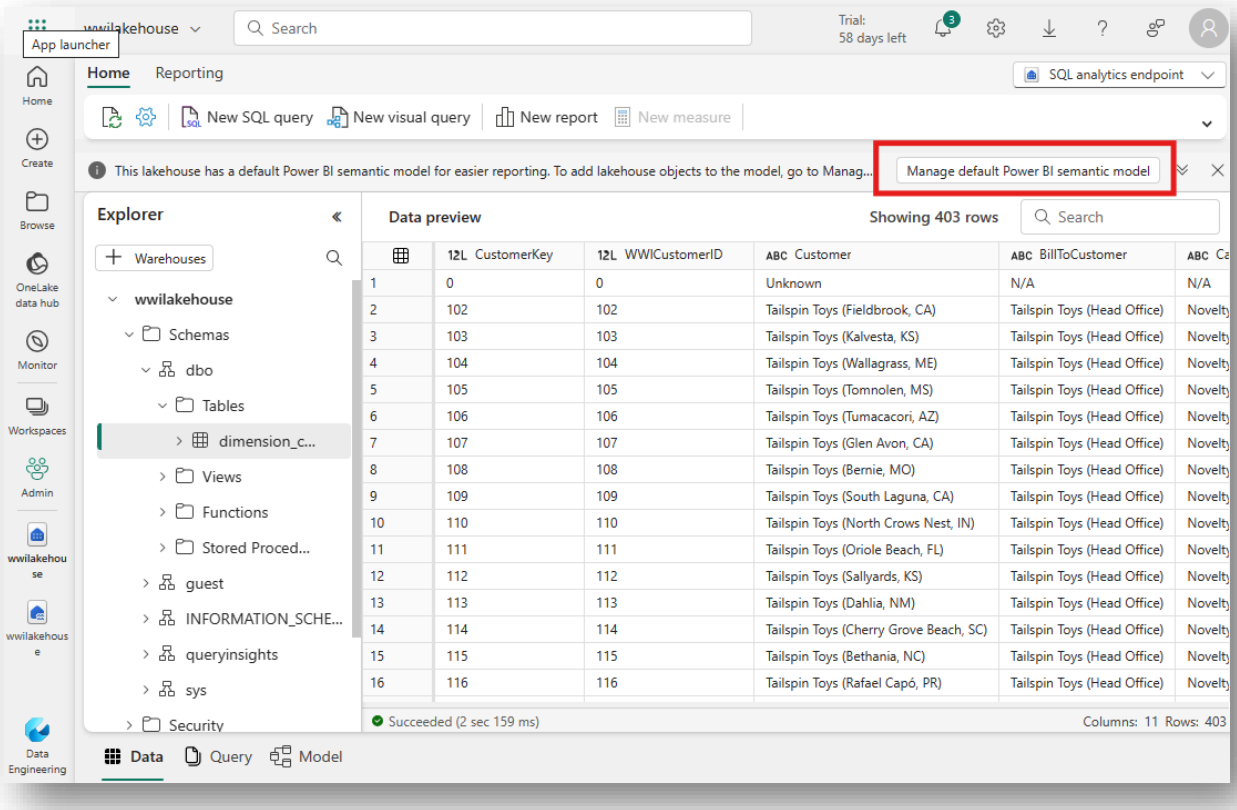



## Add table to the semantic model

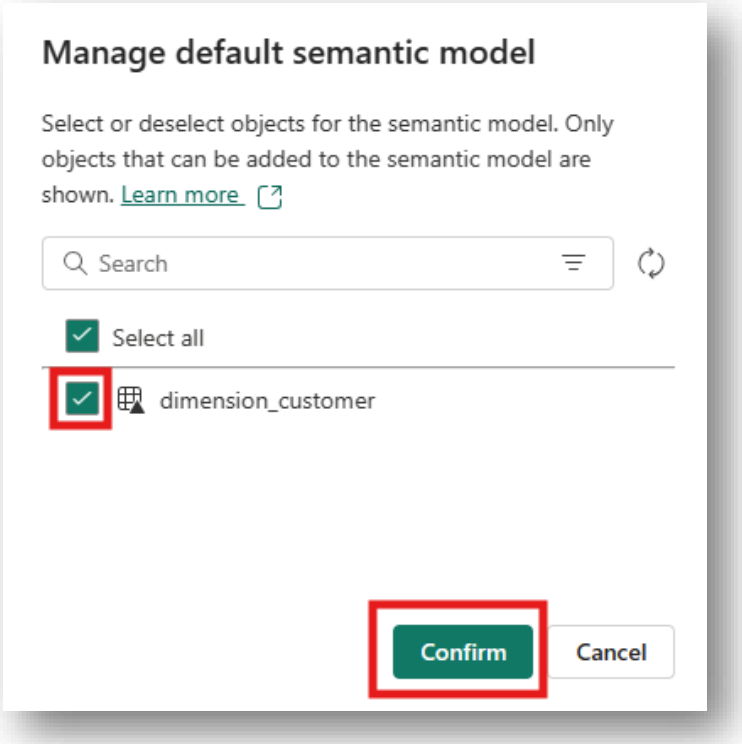
In Microsoft Fabric, Power BI semantic models are a logical description of an analytical domain, with metrics, business friendly terminology, and representation, to enable deeper analysis. This semantic model is typically a star schema with facts that represent a domain, and dimensions that allow you to analyze, or slice and dice the domain to drill down, filter, and calculate different analyses.

Now that you've ingested data into a lakehouse table, we'll add that table to the **default semantic model** for the lakehouse.

1. From the SQL analytics Data view of the dimension_customer table, click **Manage default Power BI semantic model** button on the upper right.



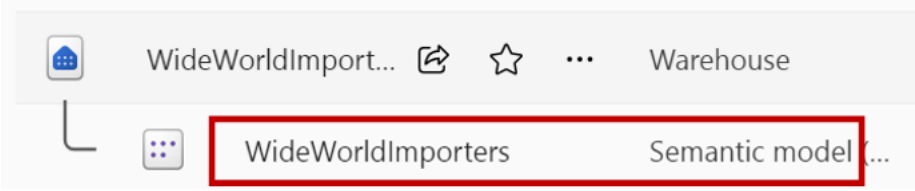

2. The **Manage default semantic model** dialog will appear. Select the dimension_customer table and click **Confirm**.



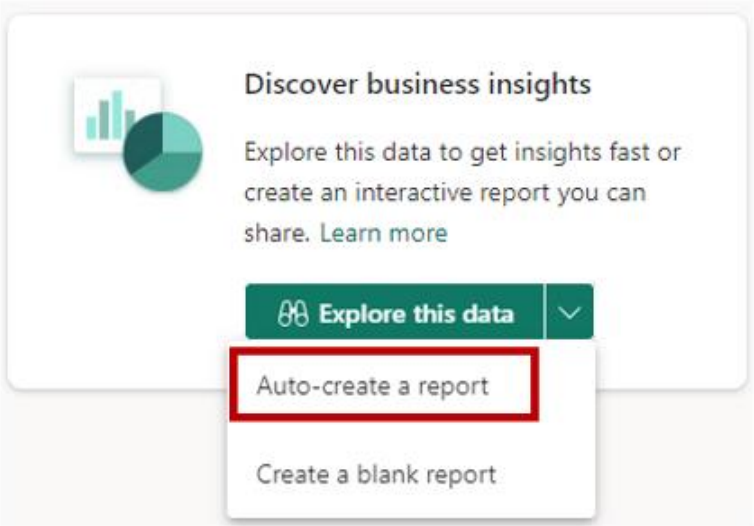

## Building a report

1. Select **Data Warehouse Tutorial** in the left-hand navigation menu to return to the workspace artifact view.

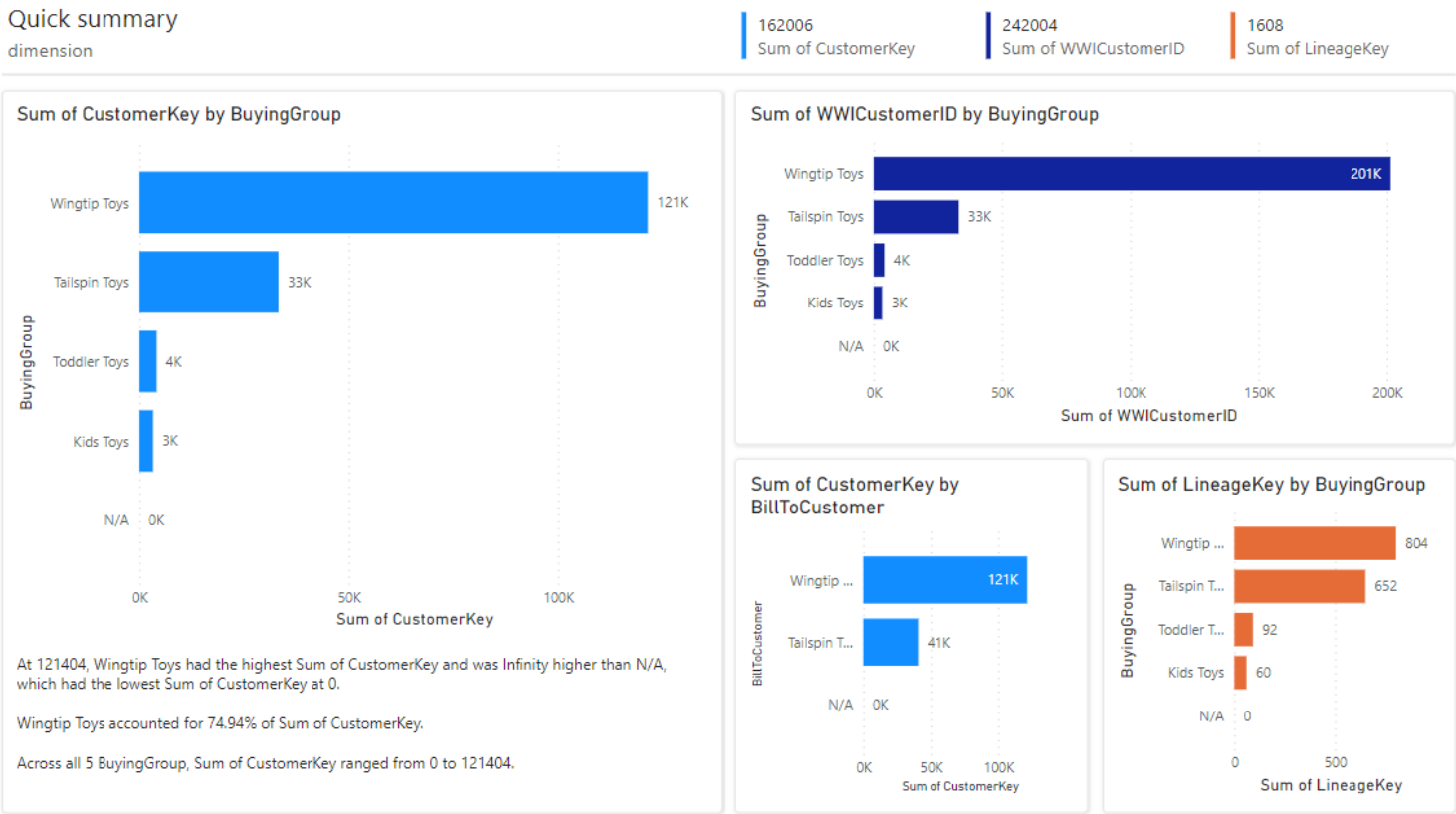

2. From the artifact list, select **WideWroldImporters** with the type of **Semantic Model (default)**.




3. In the **Discover business insights** section, select **Explore this data > Auto-create a report**. A report will be generated from the dimension_customer table that was loaded in the previous section.

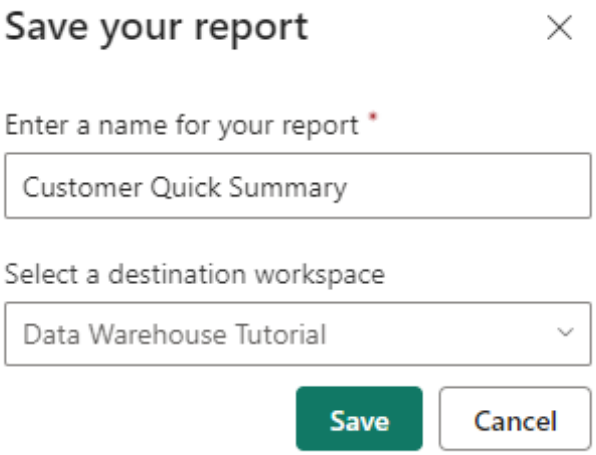



4. A report similar to one shown below will be generated.




5. From the ribbon, select **Save.**


6.
7. Enter **Customer Quick Summary** in the name box.
8. Select **Save**.

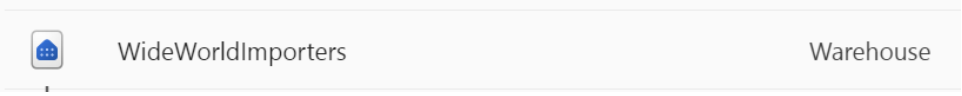


# Module 2: Extending the solution

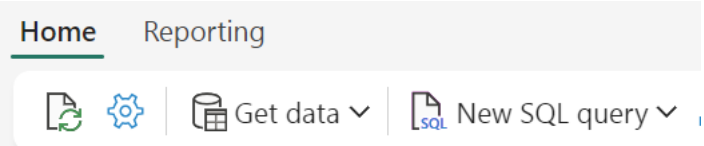
Now that you have see how to build a data warehouse, load a table, and generate a report it is time to extend the solution by exploring additional methods for loading data, querying data, and building reports.

## Creating tables in the data warehouse

1. Select **Workspaces** in the left-hand menu of the Power BI service.
2. Select the workspace created in **Module 1: Getting started**, such as **Data Warehouse Tutorial**.
3. From the artifact list, select **WideWorldImporters** with the type of **Warehouse**.




4. From the ribbon, select **New SQL query**.



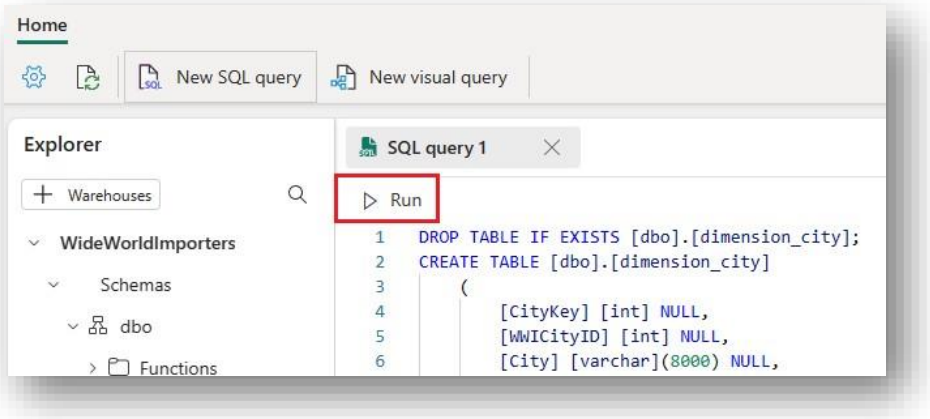

5. In the query editor, paste the code below.

   **Note:** In case of issues with copy/paste formatting, a text file containing the script called **Create Tables.txt** can be accessed from the Scripts folder.

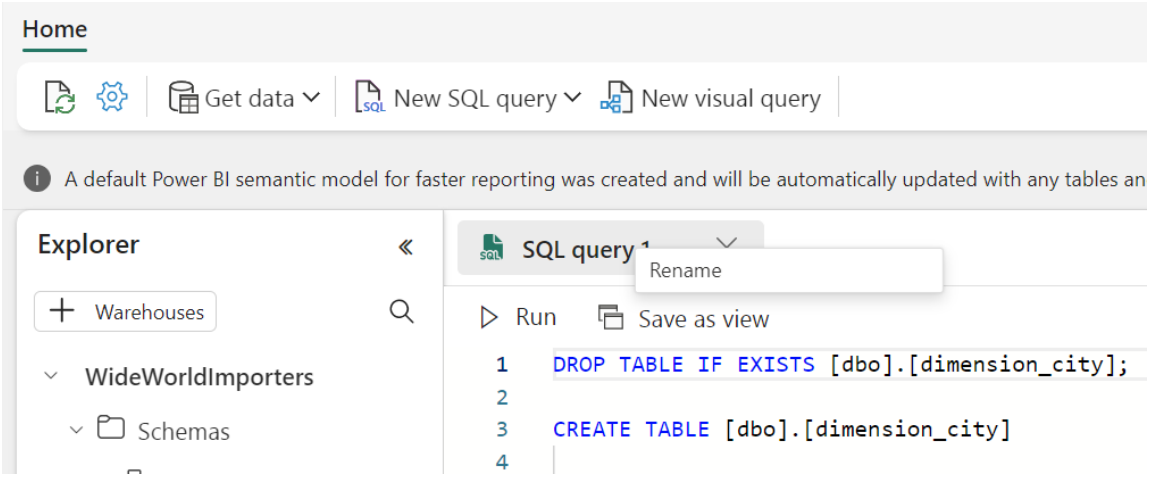
```
/*
1. Drop the dimension_city table if it already exists.
2. Create the dimension_city table.
3. Drop the fact_sale table if it already exists.
4. Create the fact_sale table.
*/


--dimension_city
DROP TABLE IF EXISTS [dbo].[dimension_city];

CREATE TABLE [dbo].[dimension_city]

    (

        [CityKey] [int] NULL,

        [WWICityID] [int] NULL,

        [City] [varchar](8000) NULL,

        [StateProvince] [varchar](8000) NULL,

        [Country] [varchar](8000) NULL,

        [Continent] [varchar](8000) NULL,

        [SalesTerritory] [varchar](8000) NULL,

        [Region] [varchar](8000) NULL,

        [Subregion] [varchar](8000) NULL,

        [Location] [varchar](8000) NULL,

        [LatestRecordedPopulation] [bigint] NULL,
```

```
        [ValidFrom] [datetime2](6) NULL,
        [ValidTo] [datetime2](6) NULL,
        [LineageKey] [int] NULL
    );


--fact_sale
DROP TABLE IF EXISTS [dbo].[fact_sale];
CREATE TABLE [dbo].[fact_sale]
    (
        [SaleKey] [bigint] NULL,
        [CityKey] [int] NULL,
        [CustomerKey] [int] NULL,
        [BillToCustomerKey] [int] NULL,
        [StockItemKey] [int] NULL,
        [InvoiceDateKey] [datetime2](6) NULL,
        [DeliveryDateKey] [datetime2](6) NULL,
        [SalespersonKey] [int] NULL,
        [WWIInvoiceID] [int] NULL,
        [Description] [varchar](8000) NULL,
        [Package] [varchar](8000) NULL,
        [Quantity] [int] NULL,
        [UnitPrice] [decimal](18, 2) NULL,
        [TaxRate] [decimal](18, 3) NULL,
        [TotalExcludingTax] [decimal](29, 2) NULL,
        [TaxAmount] [decimal](38, 6) NULL,
        [Profit] [decimal](18, 2) NULL,
        [TotalIncludingTax] [decimal](38, 6) NULL,
        [TotalDryItems] [int] NULL,
        [TotalChillerItems] [int] NULL,
        [LineageKey] [int] NULL,
        [Month] [int] NULL,
        [Year] [int] NULL,
        [Quarter] [int] NULL
    );
```

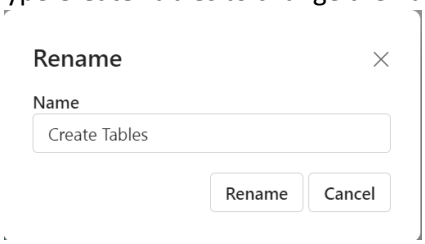
6. Select **Run** to execute the query.



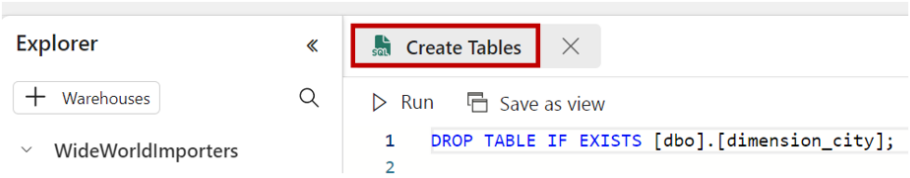

7. To save this query for reference later, right-click on the query tab just above the editor and select **Rename**.



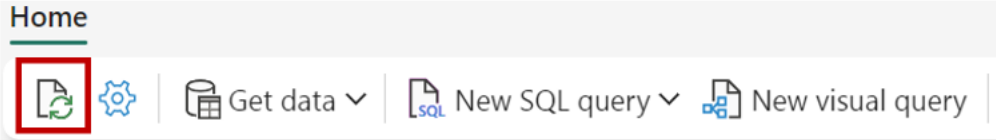

8. Type **Create Tables** to change the name of the query.




9. Click **Rename** to save the query with a given name




10. Validate the table was created successfully by clicking the **refresh** button on the ribbon.




11. In the **Object explorer** verify that you can see the newly created **Create Tables** query, **fact_sale** table, and **dimension_city** table.

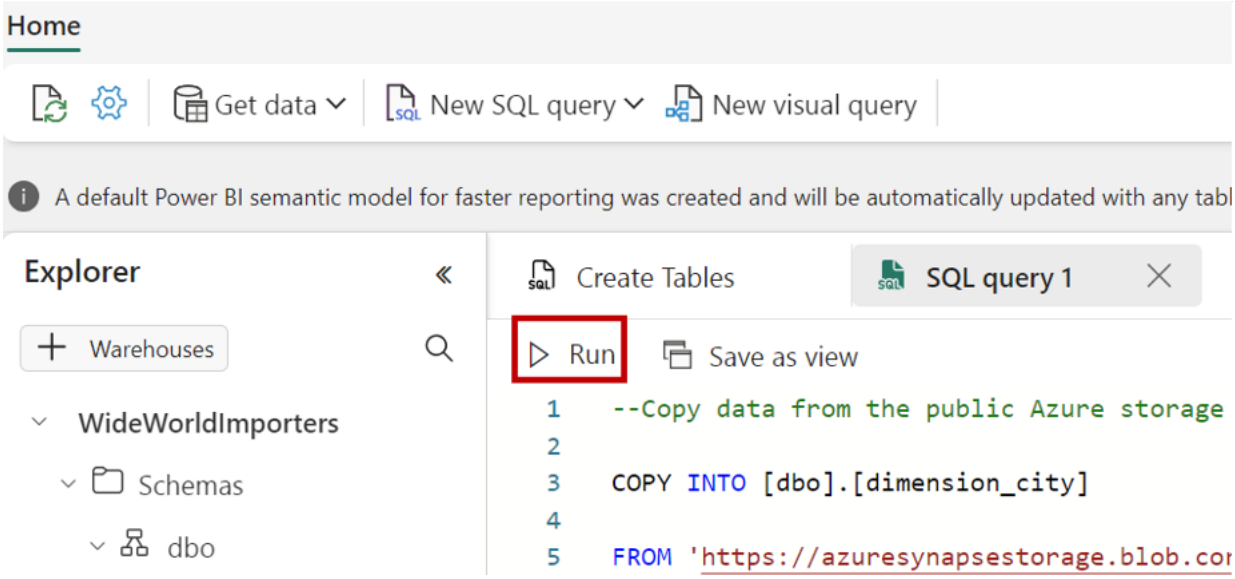

## Loading data using T-SQL

1. From the ribbon, select **New SQL query**.


2. In the query editor, paste the code below.
   **Note:** In case of issues with copy/paste formatting, a text file containing the script called **Load Tables.txt** can be accessed from Scripts folder .

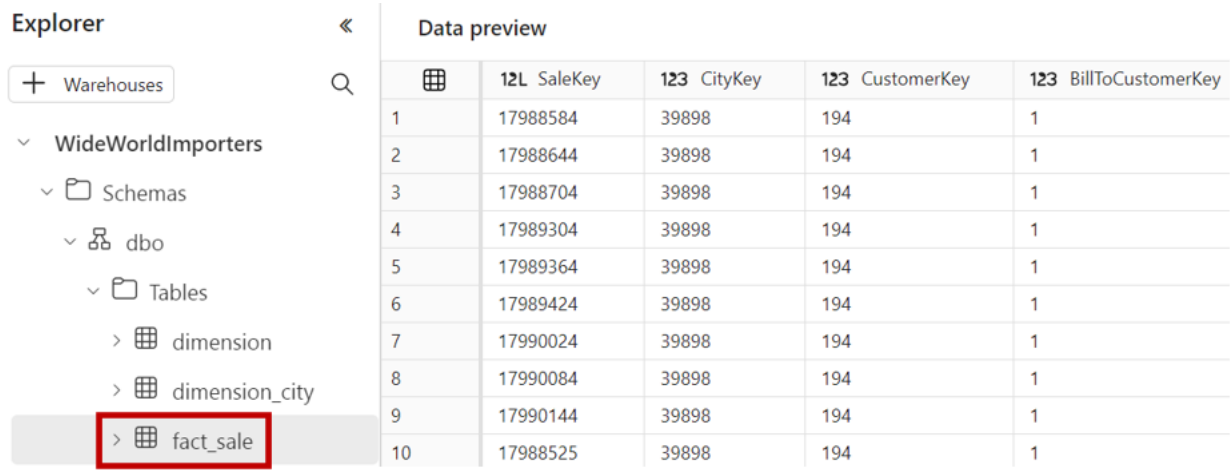
```
--Copy data from the public Azure storage account to the dbo.dimension_city table.
COPY INTO [dbo].[dimension_city]
FROM 'https://azuresynapsestorage.blob.core.windows.net/sampledata/WideWorldImportersDW/tables/dimension_city.parquet'
WITH (FILE_TYPE = 'PARQUET');


--Copy data from the public Azure storage account to the dbo.fact_sale table.
COPY INTO [dbo].[fact_sale]
FROM 'https://azuresynapsestorage.blob.core.windows.net/sampledata/WideWorldImportersDW/tables/fact_sale.parquet'
WITH (FILE_TYPE = 'PARQUET');
```

3. Select **Run** to execute query. The query will take between 1 and 4 minutes to execute.

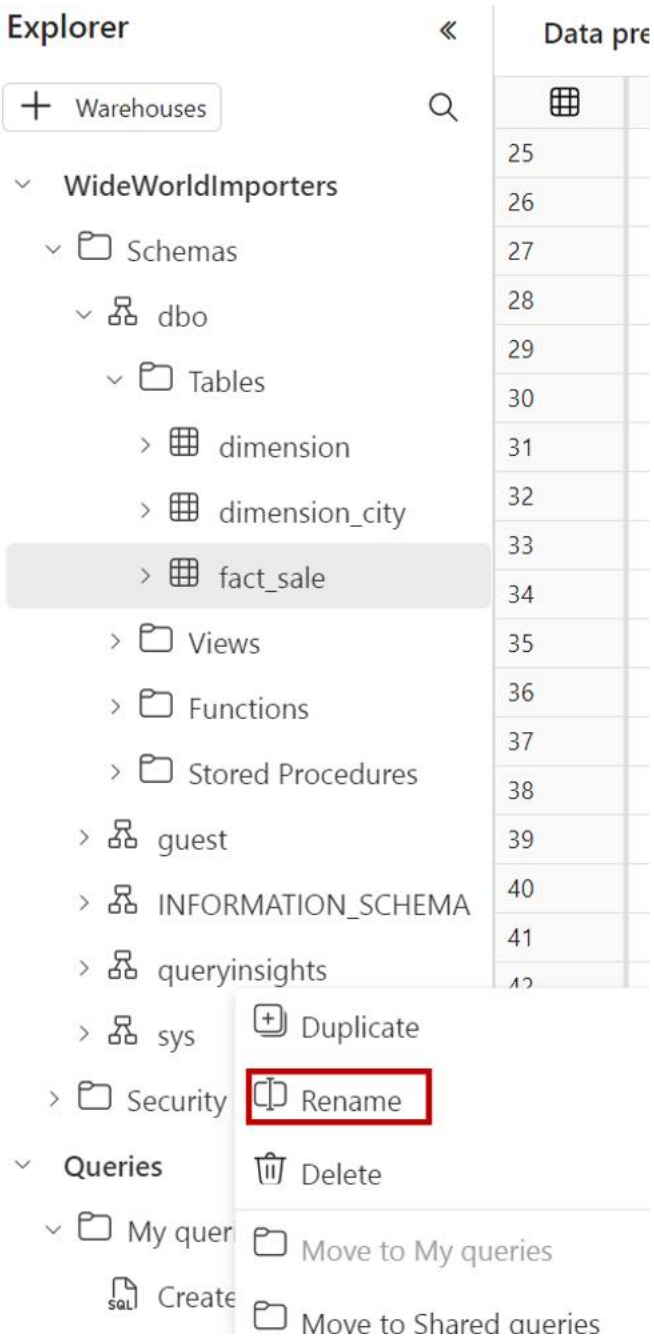



4. After the query is completed, review the messages to see the rows affected which indicated the number of rows that were loaded into the dimension_city and fact_sale tables respectively.

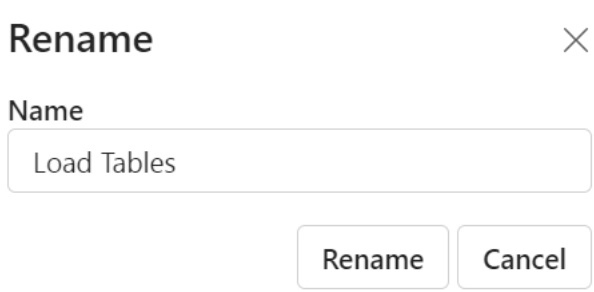

5. Load the data preview to validate the data loaded successfully by clicking on the **fact_sale** table in the **Explorer**.




6. Rename the query for reference later. Right-click on **SQL query 1** in the **Explorer** and select **Rename**.



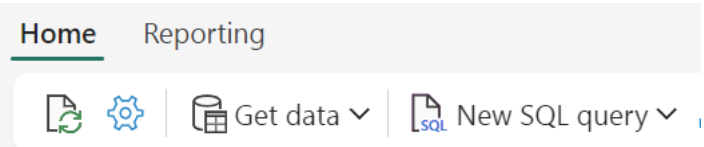

7. Type **Load Tables** to change the name of the query.



8. Click **Rename** to change the name of the query

## Data transformation using a stored procedure

1. From the **Home** tab of the ribbon, select **New SQL query**.

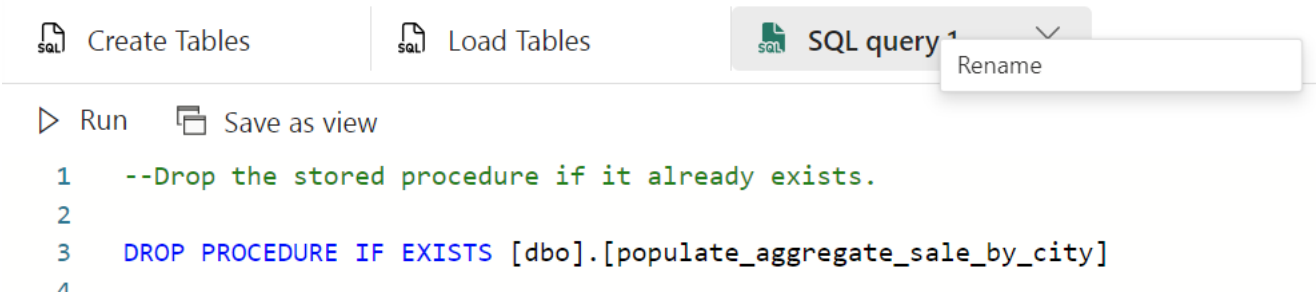



2. In the query editor, paste the code below.
   **Note:** In case of issues with copy/paste formatting, a text file containing the script called **Create Aggregate Procedure.txt** from the Scripts folder .

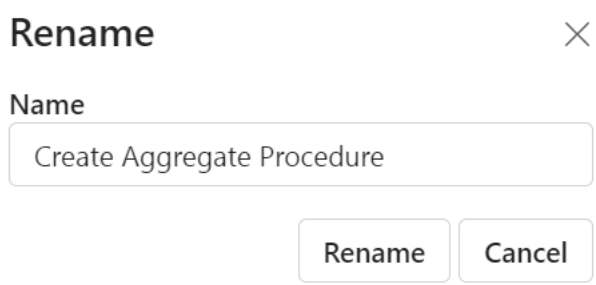
```
--Drop the stored procedure if it already exists.
DROP PROCEDURE IF EXISTS [dbo].[populate_aggregate_sale_by_city]
GO


--Create the populate_aggregate_sale_by_city stored procedure.
CREATE PROCEDURE [dbo].[populate_aggregate_sale_by_city]
AS
BEGIN
    --If the aggregate table already exists, drop it. Then create the table.
    DROP TABLE IF EXISTS [dbo].[aggregate_sale_by_date_city];
    CREATE TABLE [dbo].[aggregate_sale_by_date_city]
        (
            [Date] [DATETIME2](6),
            [City] [VARCHAR](8000),
            [StateProvince] [VARCHAR](8000),
            [SalesTerritory] [VARCHAR](8000),
            [SumOfTotalExcludingTax] [DECIMAL](38,2),
            [SumOfTaxAmount] [DECIMAL](38,6),
            [SumOfTotalIncludingTax] [DECIMAL](38,6),
            [SumOfProfit] [DECIMAL](38,2)
        );


    --Reload the aggregated dataset to the table.
    INSERT INTO [dbo].[aggregate_sale_by_date_city]
    SELECT
        FS.[InvoiceDateKey] AS [Date],
        DC.[City],
        DC.[StateProvince],
        DC.[SalesTerritory],
        SUM(FS.[TotalExcludingTax]) AS [SumOfTotalExcludingTax],
SUM(FS.[TaxAmount]) AS [SumOfTaxAmount],
        SUM(FS.[TotalIncludingTax]) AS [SumOfTotalIncludingTax],
        SUM(FS.[Profit]) AS [SumOfProfit]
```

```
    FROM [dbo].[fact_sale] AS FS
    INNER JOIN [dbo].[dimension_city] AS DC
        ON FS.[CityKey] = DC.[CityKey]
    GROUP BY
        FS.[InvoiceDateKey],
DC.[City],
        DC.[StateProvince],
        DC.[SalesTerritory]
    ORDER BY
        FS.[InvoiceDateKey],
        DC.[StateProvince],
        DC.[City];
END
```

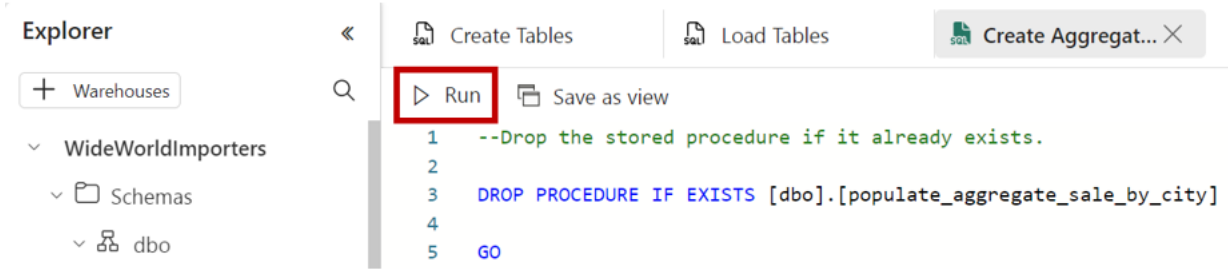
1. To save this query for reference later, right-click on the query tab just above the editor and select **Rename**.



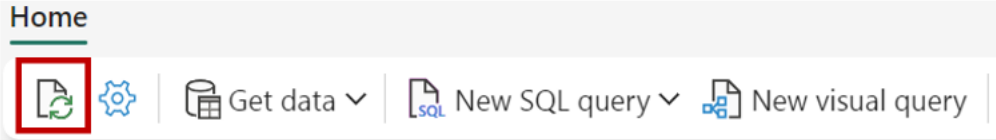

2. Type **Create Aggregate Procedure** to change the name of the query.



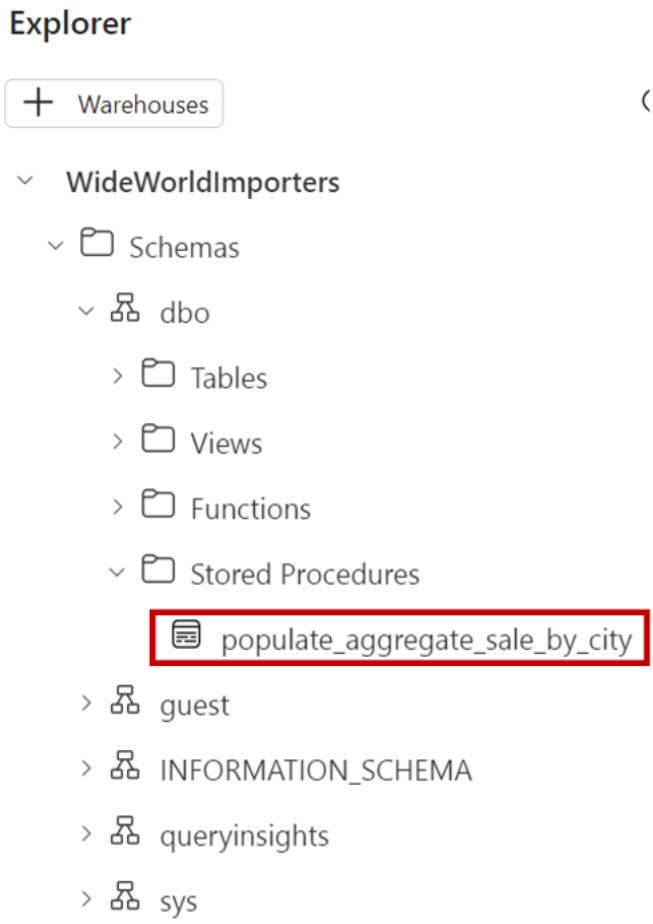

3. Click on **Rename** to save the query with given name
4. Select **Run** to execute the query.




5. Click the **refresh** button on the ribbon.



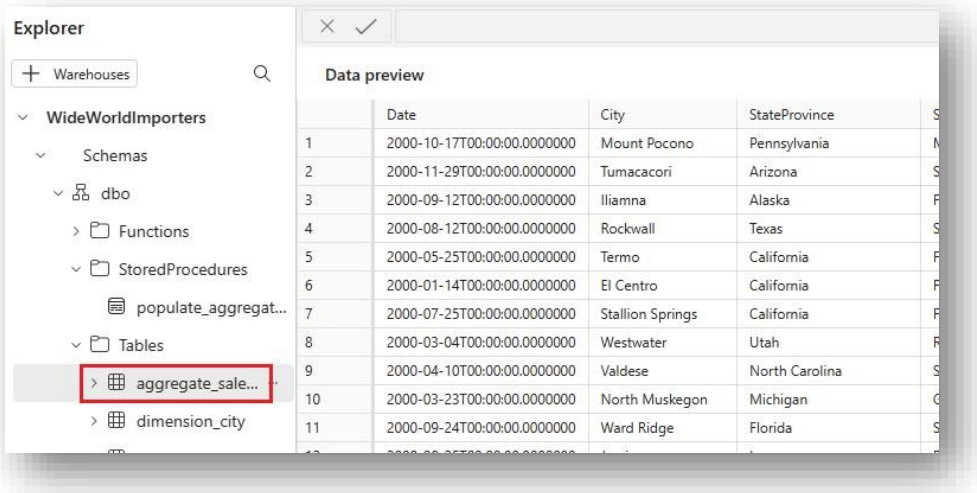

6. In the **Object explorer** verify that you can see the newly created stored procedure by expanding the **StoredProcedures** node under the **dbo** schema.




7. From the **Home** tab of the ribbon, select **New SQL query**.
8. In the query editor, paste the code below.
   **Note:** In case of issues with copy/paste formatting, a text file containing the script called **Run Aggregate Procedure.txt** can be accessed from the Scripts folder .

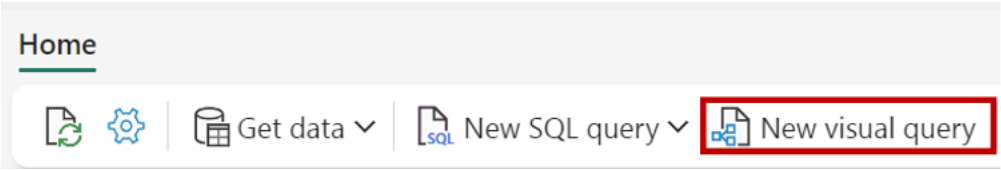
```
--Execute the stored procedure to create the aggregate table.
EXEC [dbo].[populate_aggregate_sale_by_city];
```

9. To save this query for reference later, right-click on the query tab just above the editor and select **Rename**.
10. Type **Run Create Aggregate Procedure** to change the name of the query.
11. Select **Run** to execute the query.
12. Click the **refresh** button on the ribbon. The query will take between 2 and 3 minutes to execute.
13. In the **Object explorer**, load the data preview to validate the data loaded successfully by clicking on the **aggregate_sale_by_city** table in the **Explorer**.

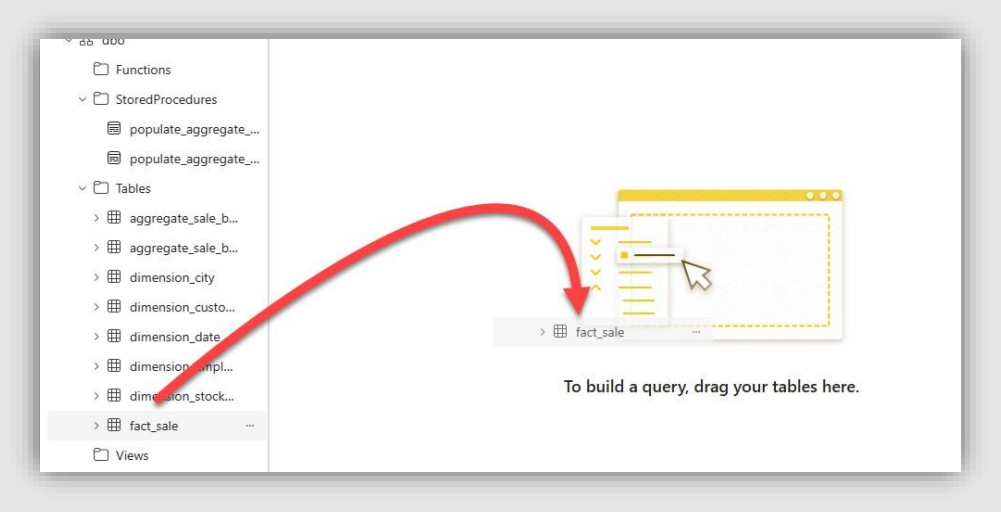



## Using the visual query builder

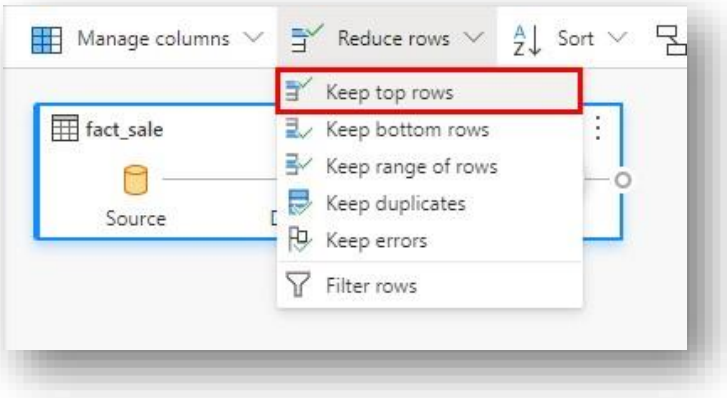
1. From the **Home** tab of the ribbon, select **New visual query**.



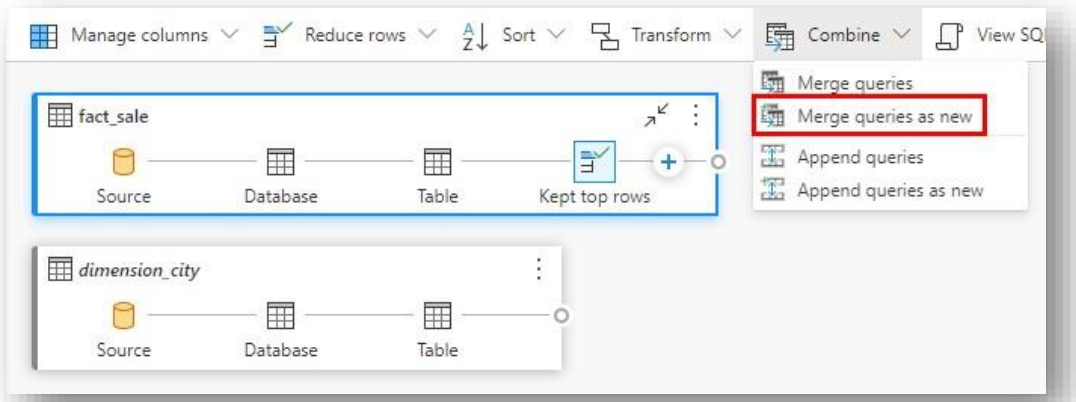

2. Drag the **fact_sale** table from the explorer to the query design pane.



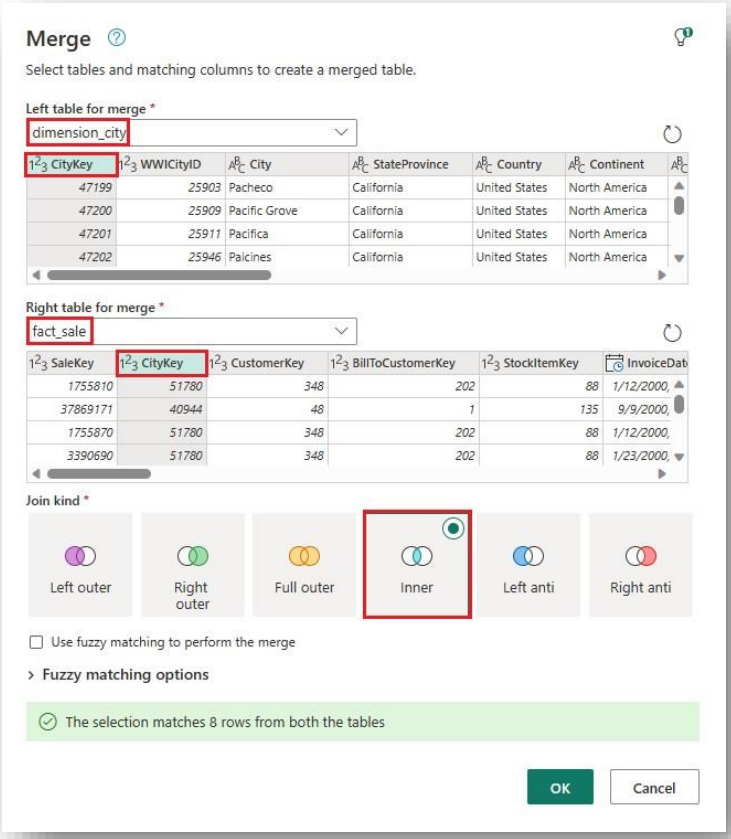

3. Limit the dataset size by selecting **Reduce rows > Keep top rows** from the transformations ribbon.




4. In the **Keep top rows** dialog enter **10,000**.
5. Select **OK**.
6. Drag the **dimension_city** table from the explorer to the query design pane.
7. From the transformations ribbon, select the dropdown next to **Combine** and select **Merge queries as new**.

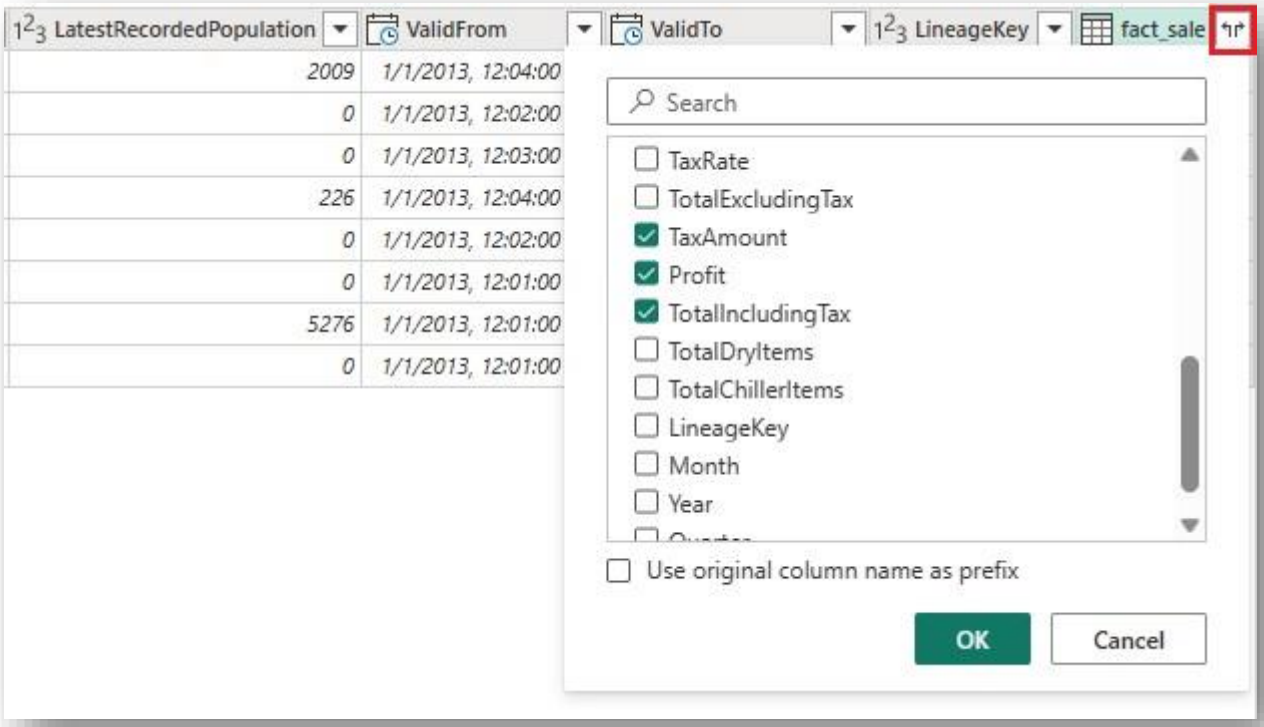



8. On the **Merge** settings page:
    a. **Left table for merge:** dimension_city
    b. **Right table for merge:** fact_sale
    c. Select the **CityKey** field in the **dimension_city** table by clicking on the column name in the header row to indicate the join column.
    d. Select the **CityKey** field in the **fact_sale** table by clicking on the column name in the header row to indicate the join column.
    e. **Join kind:** Inner



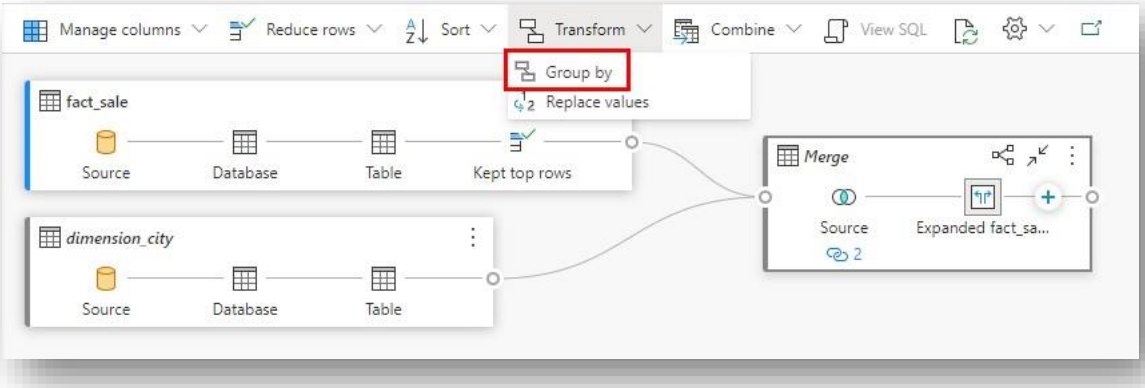

Select **OK**.

9. With the **Merge** step selected, select the **Expand** button next to **fact_sale** on the header of the data grid then select only **TaxAmount, Profit,** and **TotalIncludingTax**.



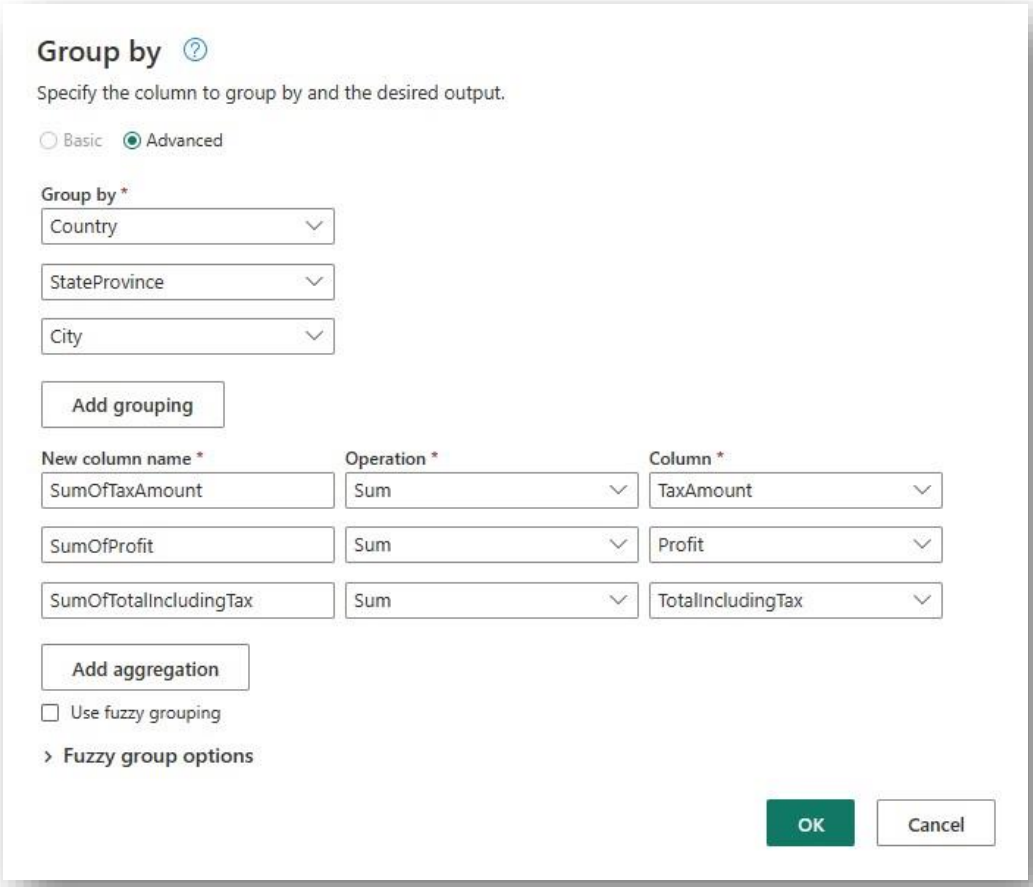

Select **OK**.

10. Select **Transform > Group by** from the transformations ribbon.



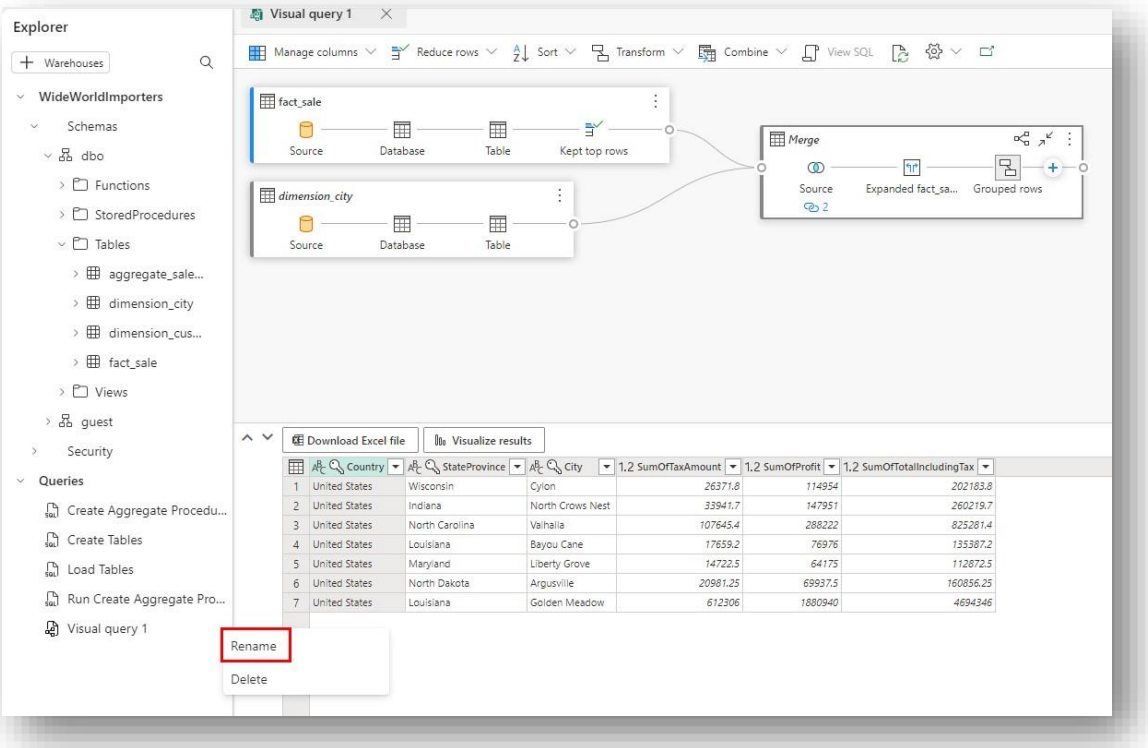

11. On the **Group by** settings page:
    a. Change to **Advanced**.
    b. **Group by** (if necessary, select **Add grouping** to add additional group by columns)**:** i. Country
        ii. StateProvince
        iii. City
    c. **New column name** (if necessary, select **Add aggregation** to add additional aggregate columns and operations)**:**

i. **SumOfTaxAmount** with **Operation** of **Sum** and **Column** of **TaxAmount**
ii. **SumOfProfit** with **Operation** of **Sum** and **Column** of **Profit** iii. **SumOfTotalIncludingTax** with **Operation** of **Sum** and **Column** of **TotalIncludingTax**




Select **OK**.

12. Right-click on **Visual query 1** in the explorer and select **Rename**.

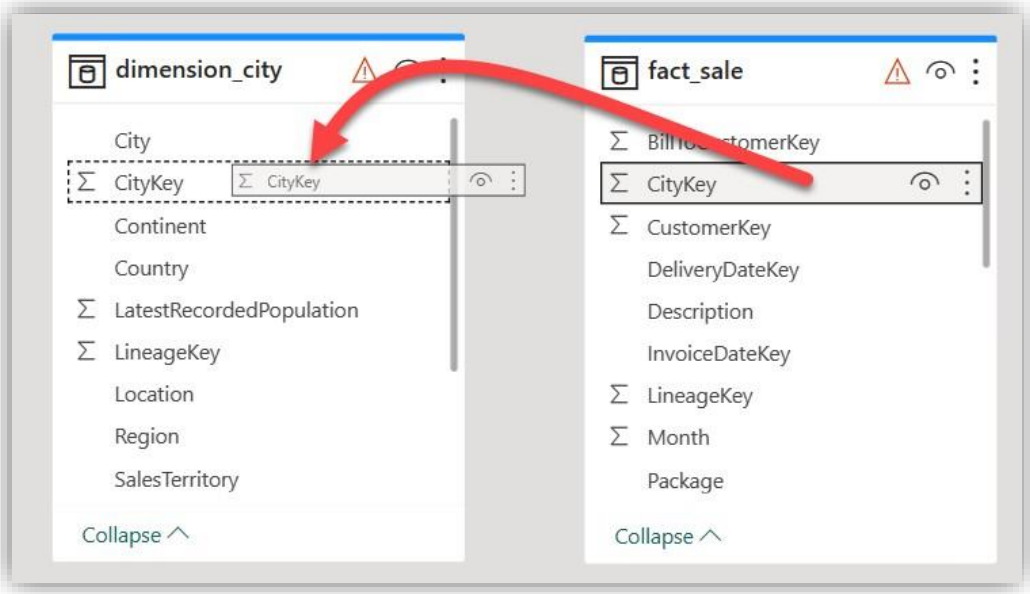



13. Type **Sales Summary** to change the name of the query.
14. Press **Enter** on the keyboard or click off anywhere outside the tab to save the change.

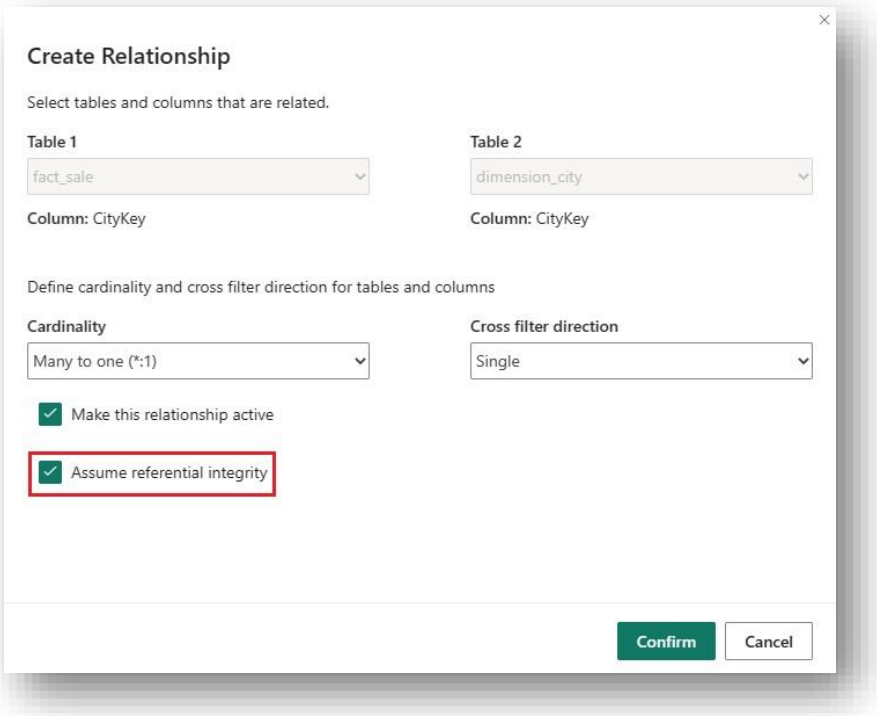
## Create a Power BI report

1. Select the **Model** view from the options in the bottom left corner, just outside the canvas.


2. From the **fact_sale** table, drag the **CityKey** field and drop it on the **CityKey** field in the **dimension_city** table to create a relationship.

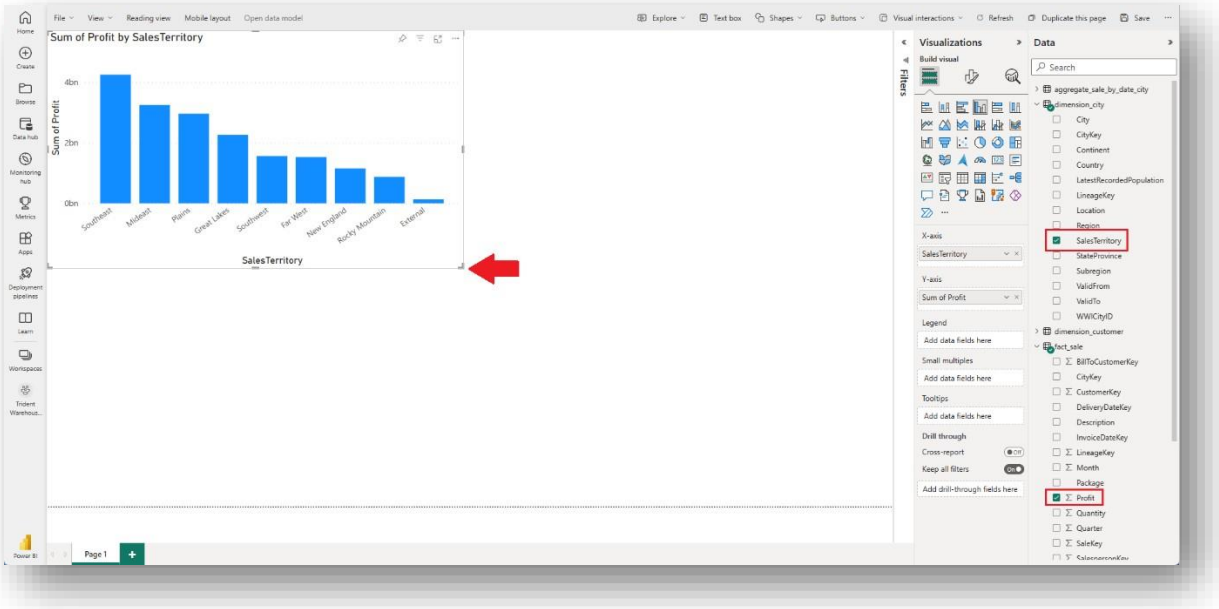



3. On the **Create Relationship** settings:
    a. Table 1 will be populated with fact_sale and the column of CityKey.
    b. Table 2 will be populated with dimension_city and the column of CityKey.
    c. Cardinality: **Many to one (*:1)**
    d. Cross filter direction: **Single**
    e. Leave the box next to **Make this relationship active** checked.
    f. Check the box next to **Assume referential integrity.**

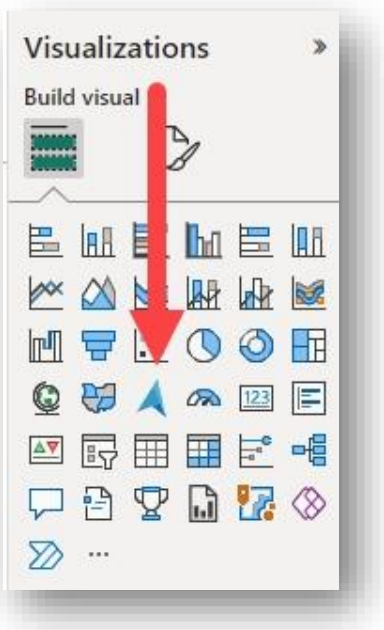



Select **Confirm**.

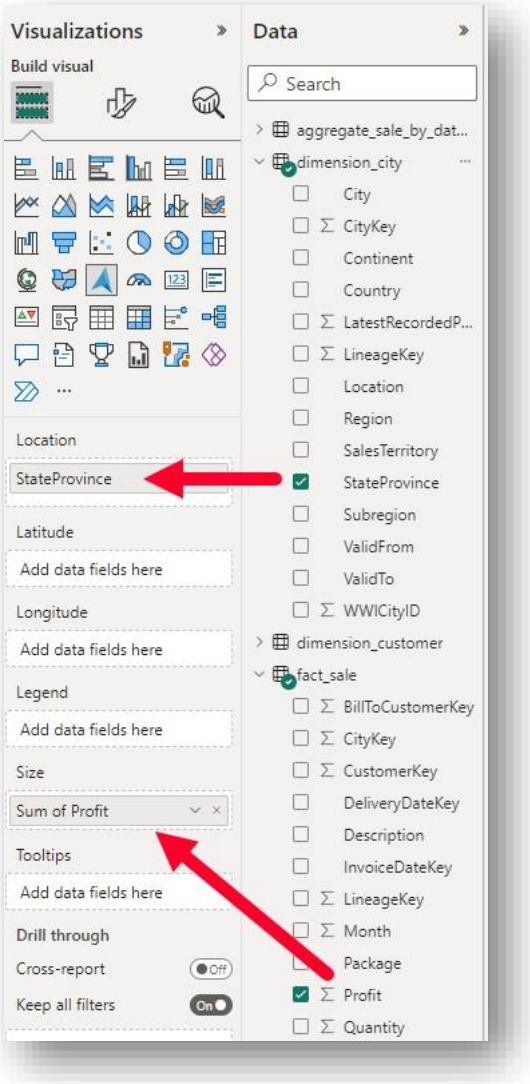
4. From the **Home** tab of the ribbon, select **New report**.
5. Build a column chart visual:
    a. On the **Data** pane, expand **fact_sales** and check the box next to **Profit**. This will create a column chart and add the field to the Y-axis.
    b. On the **Data** pane, expand **dimension_city** and check the box next to **SalesTerritory**. This will add the field to the X-axis.
    c. Reposition and resize the column chart to take up the top left quarter of the canvas by dragging the anchor points on the corners of the visual.



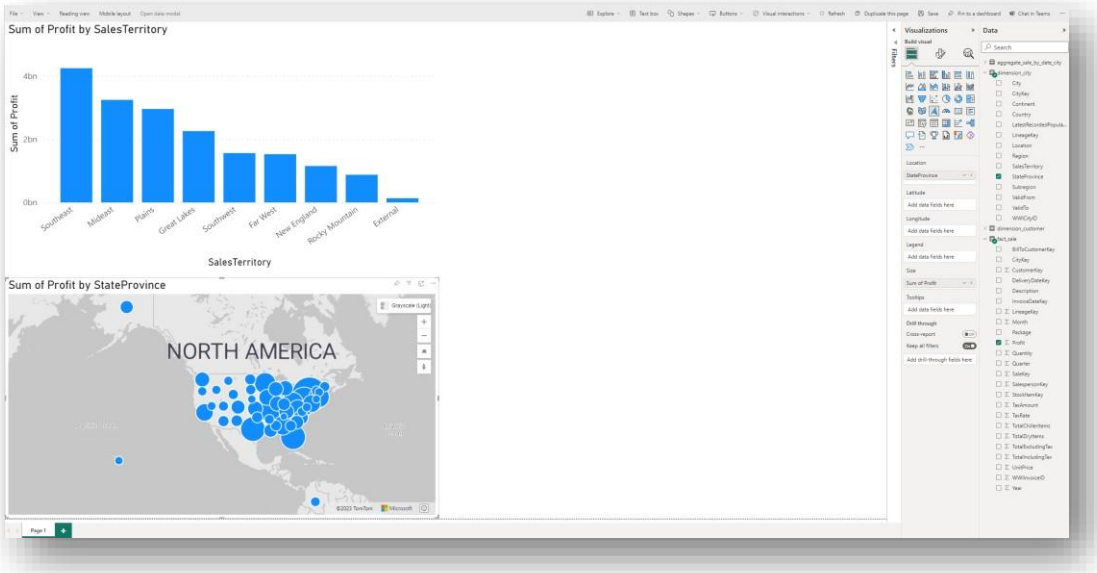

6. Click anywhere on the blank canvas (or press the Esc key) so the column chart visual is no longer selected.
7. Build a map visual:
    a. On the **Visualizations** pane, select the **Azure Map for Power BI** visual. **Azure Map** visual needs to be enabled by PowerBI admin



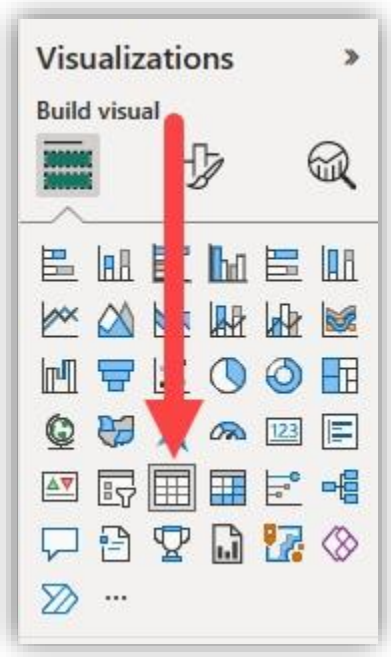

    b. From the **Data** pane, drag **StateProvince** from the **dimension_city** table to the **Location** bucket on the **Visualizations** pane.
    c. From the **Data** pane, drag **Profit** from the **fact_sale** table to the **Size** bucket on the **Visualizations** pane.




d. If necessary, reposition and resize the map to take up the bottom left quarter of the canvas by dragging the anchor points on the corners of the visual.

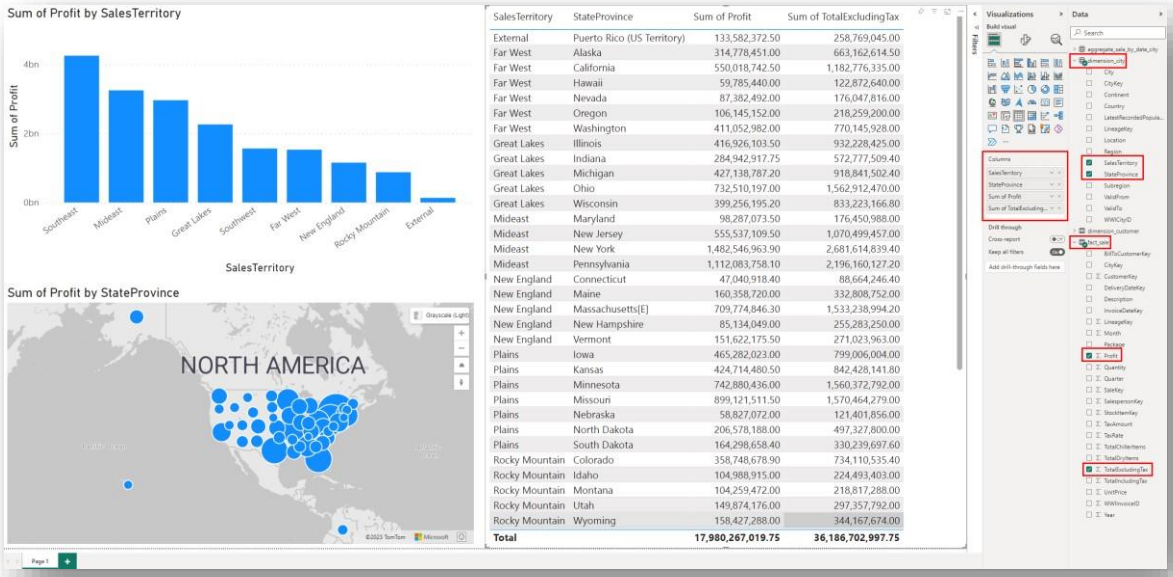



8. Click anywhere on the blank canvas (or press the Esc key) so the map visual is no longer selected.
9. Build a table visual:
    a. On the **Visualizations** pane, select the **Table** visual.




    b. From the **Data** pane, check the box next to **SalesTerritory** on the **dimension_city** table.
    c. From the **Data** pane, check the box next to **StateProvince** on the **dimension_city** table.
    d. From the **Data** pane, check the box next to **Profit** on the **fact_sale** table.
    e. From the **Data** pane, check the box next to **TotalExcludingTax** on the **fact_sale** table.

f. Reposition and resize the column chart to take up the right half of the canvas by dragging the anchor points on the corners of the visual.




10. From the ribbon, select **File > Save**.
11. Enter the name of your report as **Sales Analysis**.
12. Select **Save**.