Create a Lakehouse, ingest sample data and build a report

This lab provides an end-to-end walkthrough of building a retail lakehouse in Microsoft Fabric—from data acquisition to consumption. It introduces the core Fabric experiences, showcasing how they integrate for both professional and citizen developers. By unifying data storage through the Delta Lake format, Fabric eliminates silos, reduces data duplication, and lowers total cost of ownership. Using the medallion architecture (bronze for raw, silver for validated, and gold for refined data), the lab demonstrates how organizations across industries can easily build scalable lakehouse or data warehouse solutions within a single, unified platform.

Objectives:

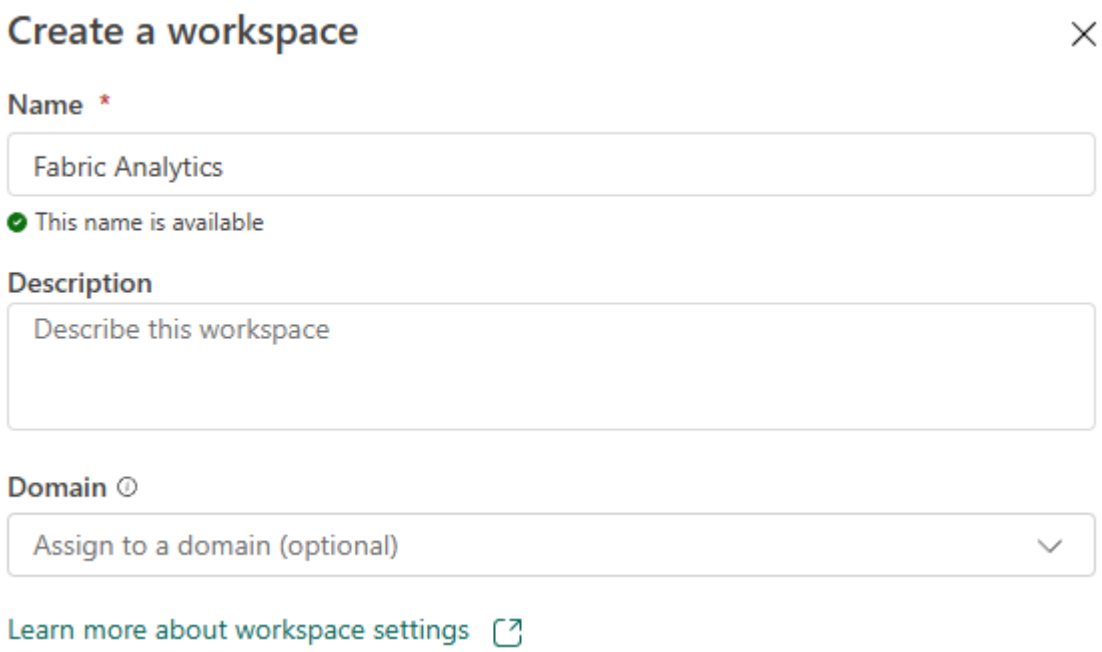
1. Build and implement an end-to-end lakehouse for the organization, including creating a Fabric workspace and a lakehouse.
2. Ingest sample data into the lakehouse and prepare it for further processing.
3. Transform and prepare the data using Python/PySpark and SQL notebooks.
4. Create business aggregate tables using different approaches.
5. Establish relationships between tables for seamless reporting.
6. Build a Power BI report with visualizations based on the prepared data.
7. Save and store the created report for future reference and analysis.

Exercise 1: Build and implement an end-to-end lakehouse for the organization

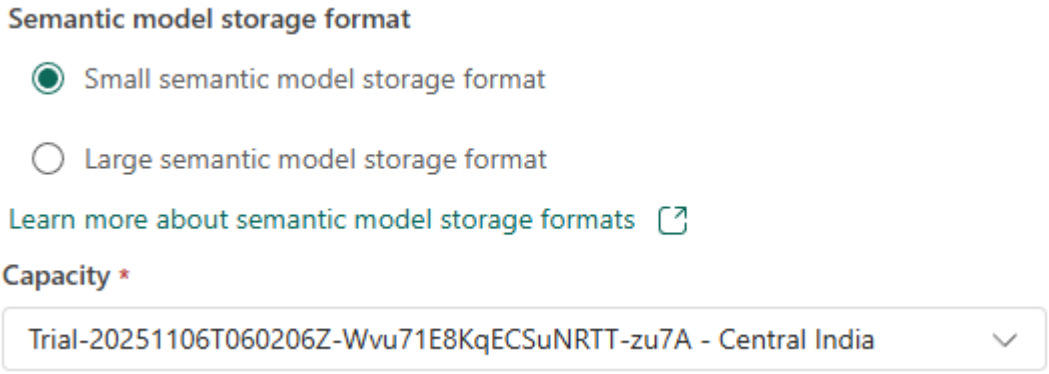
Create a Workspace (If not assigned)

In this step, you create a Fabric workspace. The workspace contains all the items needed for this lakehouse tutorial, which includes lakehouse, dataflows, Data Factory pipelines, the notebooks, Power BI semantic models, and reports.

1. Sign in to the Microsoft Fabric portal.
2. Select **Workspaces** and **New workspace**.
3. Fill out the **Create a workspace** form with the following details:
    - **Name**: Enter *Fabric Lakehouse Tutorial*, and any extra characters to make the name unique.
    - **Description**: Enter an optional description for your workspace.



    - **Advanced**: Under **License mode**, select **Trial** capacity or **Fabric capacity**.

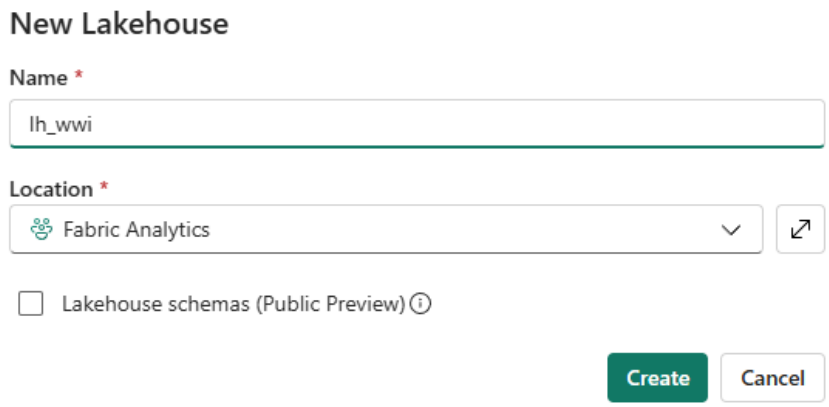


4. Select **Apply** to create and open the workspace.

Create a lakehouse

In this section, you create a lakehouse in Fabric.

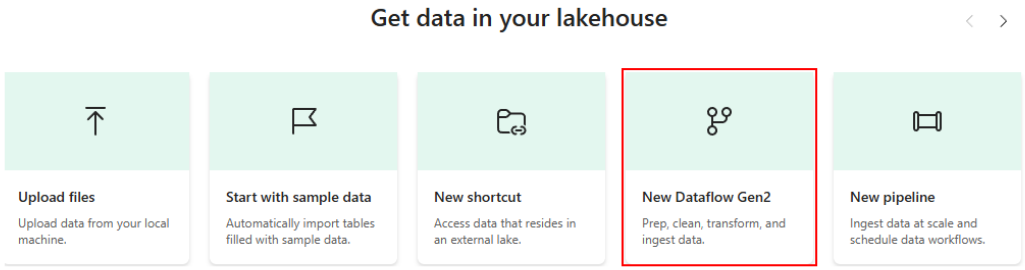
1. In Fabric, select **Workspaces** from the navigation bar.
2. Open your workspace create in previous step.
3. From the workspace, select **New item**, enter **Lakehouse** in the search box, then select **Lakehouse**.
4. In the **New lakehouse** dialog box, enter **lh_wwi** in the **Name** field.



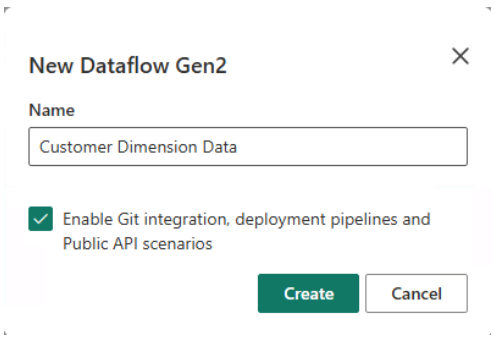
5. Select **Create** to create and open the new lakehouse.

Ingest sample data using Data Flow Gen 2

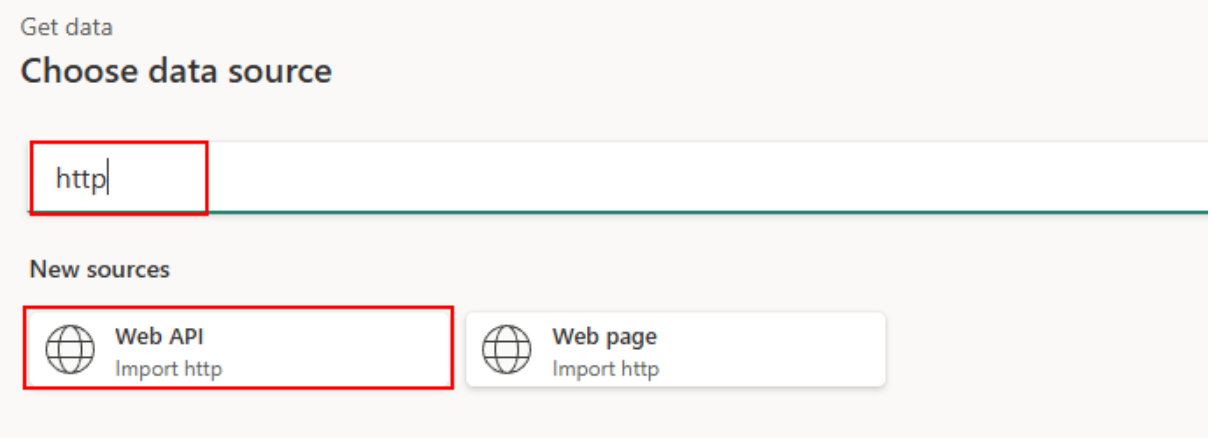
1. From Lakehouse, In the **Home** tab, under **Get data in your lakehouse**, you see options to load data into the lakehouse. Select **New Dataflow Gen2**.



2. In the **Create a dataflow** pane, enter **Customer Dimension Data** in the **Name** field and select **Next**.



3. On the new dataflow screen, select **Get data from another source ->**.
4. On the **Choose data source** screen, type **http** in the search box and select **Web API**.

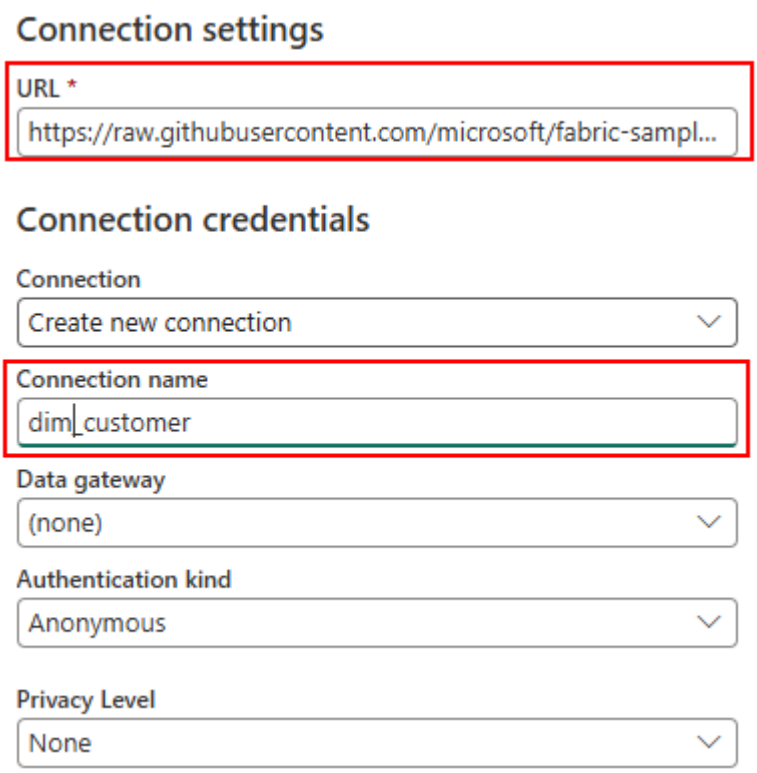


5. On the **Connect to data source** screen, enter the

   URL: https://raw.githubusercontent.com/microsoft/fabric-samples/refs/heads/main/docs-samples/data-engineering/dimension_customer.csv

   Connection name: dim_customer

   Leave rest of the field as it is and select **Next**.

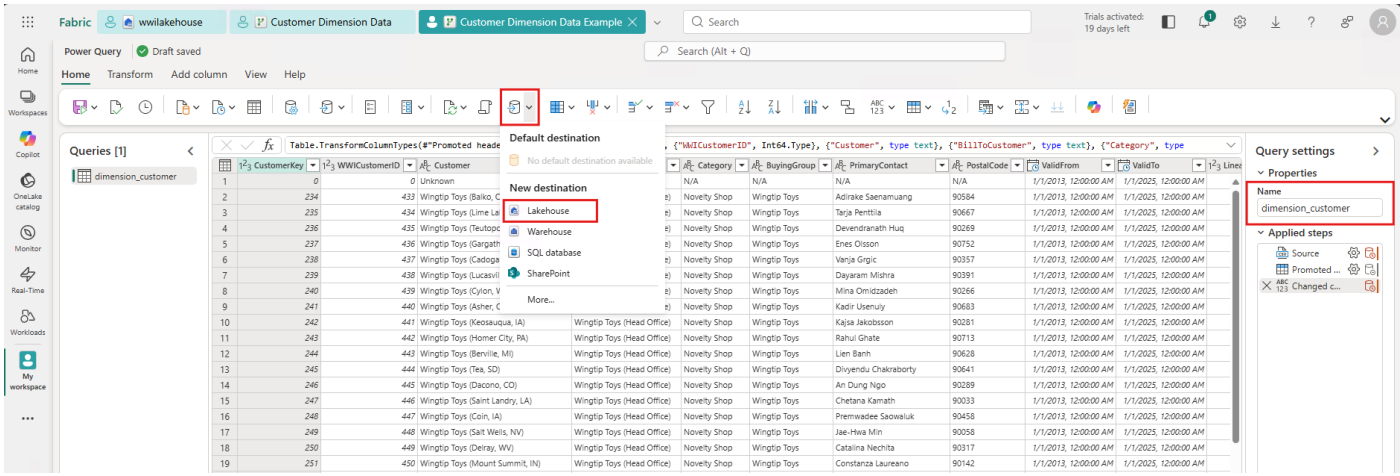


6. From the **Preview file data** page, preview the data and select **Create** to proceed and return back to the dataflow canvas.

Transform and load data into the lakehouse

1. In the **Query settings** pane, update the **Name** field to **dimension_customer**.

   Note: Fabric adds a space and number at the end of the table name by default. Table names must be lowercase and must not contain spaces. Rename it appropriately and remove any spaces from the table name

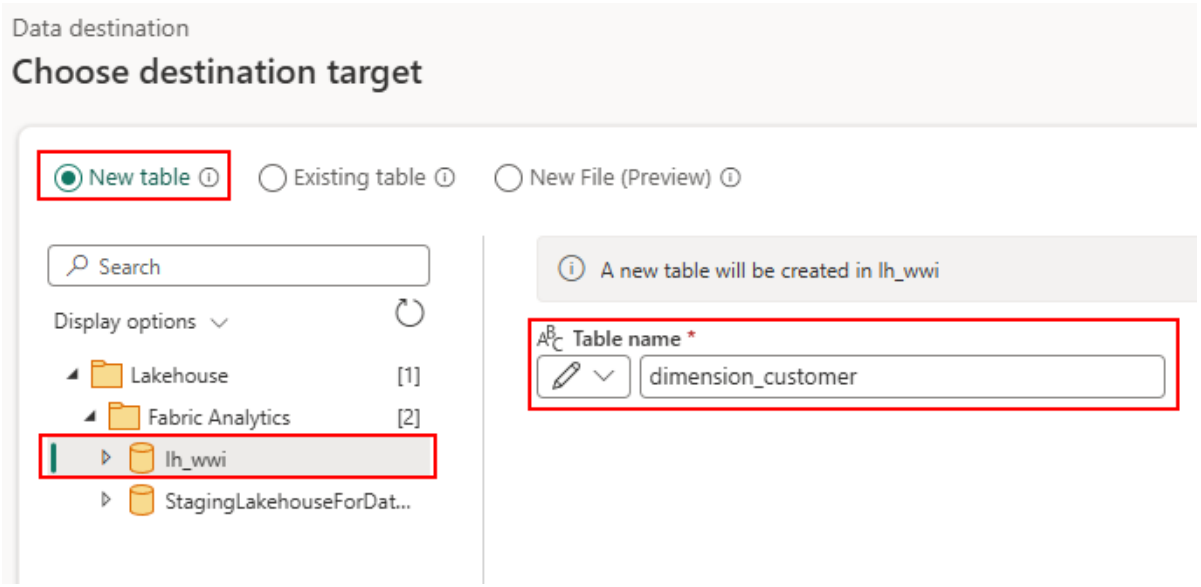


2. You associated the customer data with a lakehouse. **If you create a dataflow from the lakehouse, the uploaded data is automatically linked to the default lakehouse.** If you're creating the dataflow separately, you can optionally associate it with a lakehouse by following these steps:

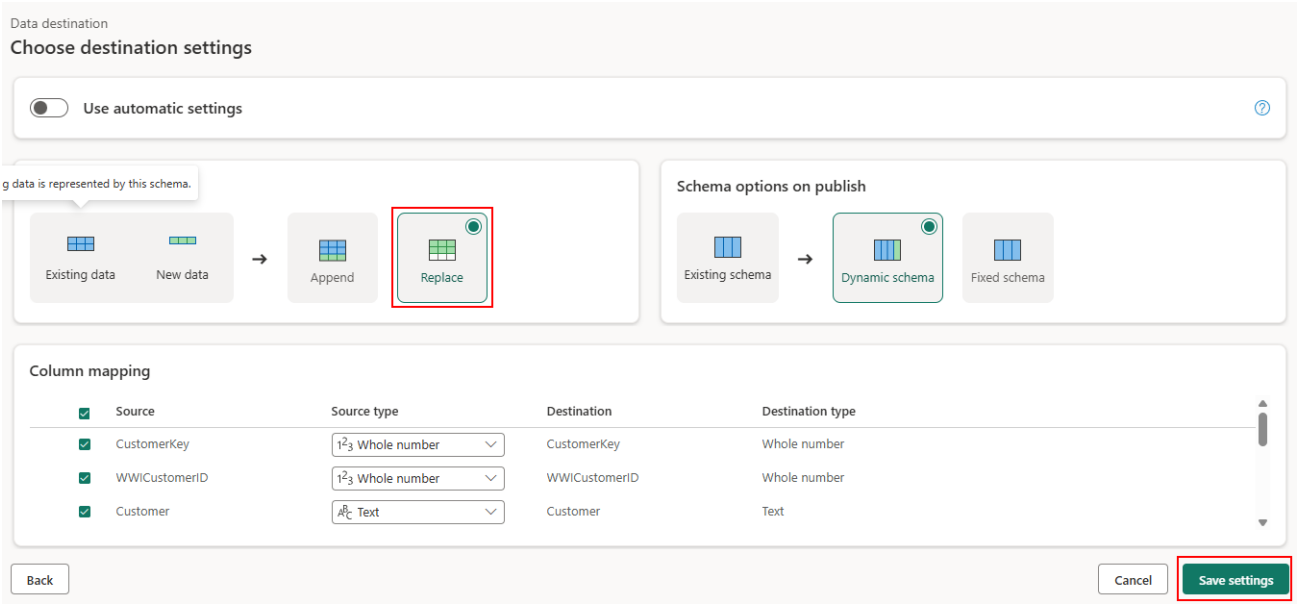
   a. From the menu items, select **Add data destination** and select **Lakehouse**. From the **Connect to data destination** screen, sign in to your account if necessary and select **Next**.

   b. Navigate to the **lh_wwi** in your workspace.

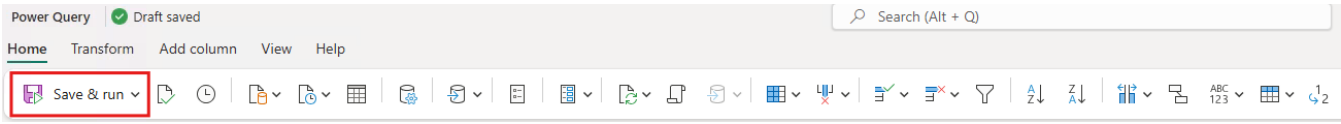
   c. If the **dimension_customer** table doesn't exist, select the **New table** setting and enter the table name **dimension_customer**. If the table already exists, select the **Existing table** setting and choose **dimension_customer** from the list of tables in the object explorer. Select **Next**.



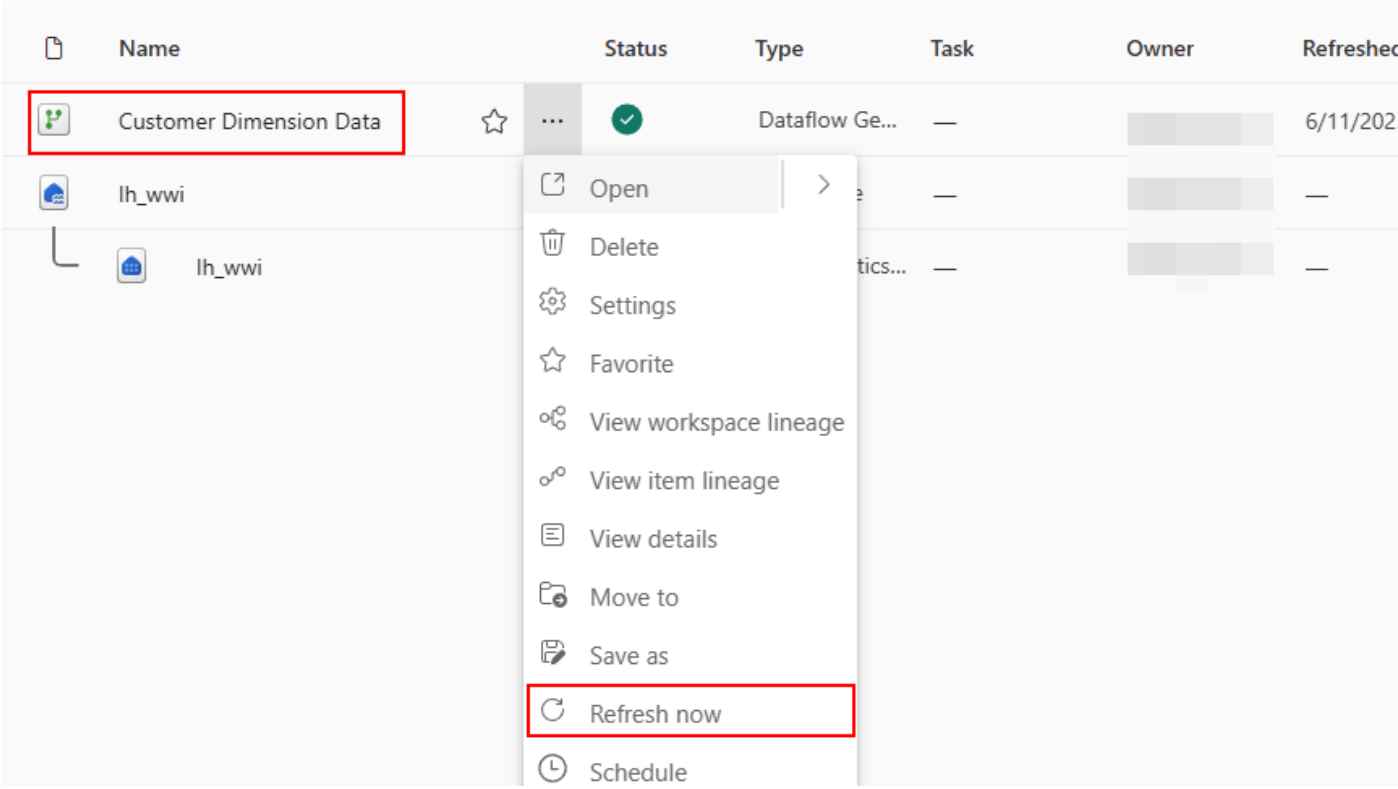
   d. On the **Choose destination settings** pane, select **Replace** as **Update method**. Select **Save settings** to return to the dataflow canvas.



3. From the dataflow canvas, you can easily transform the data based on your business requirements. For simplicity, we aren't making any changes in this tutorial. To proceed, select **Save and Run** in the tool bar.

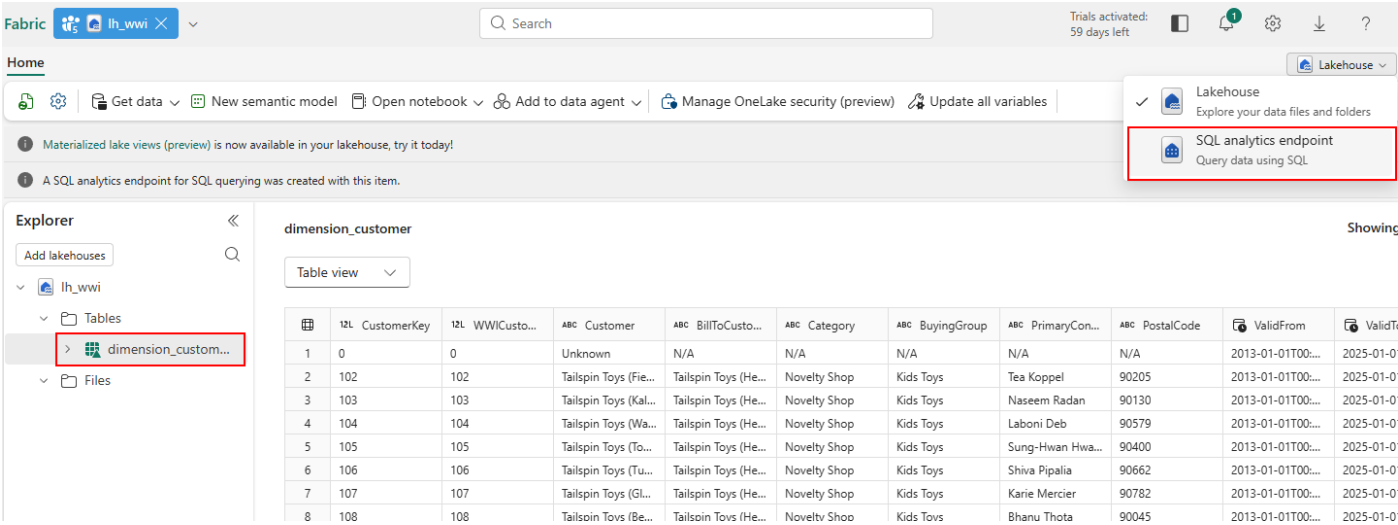


4. Return to your workspace and hover over the **Customer Dimension Data** dataflow, select the **...** menu, and then select **Refresh now**. This option runs the data flow and moves data from the source file to lakehouse table. While it's in progress, you see a spinning circle next to the dataflow's name.

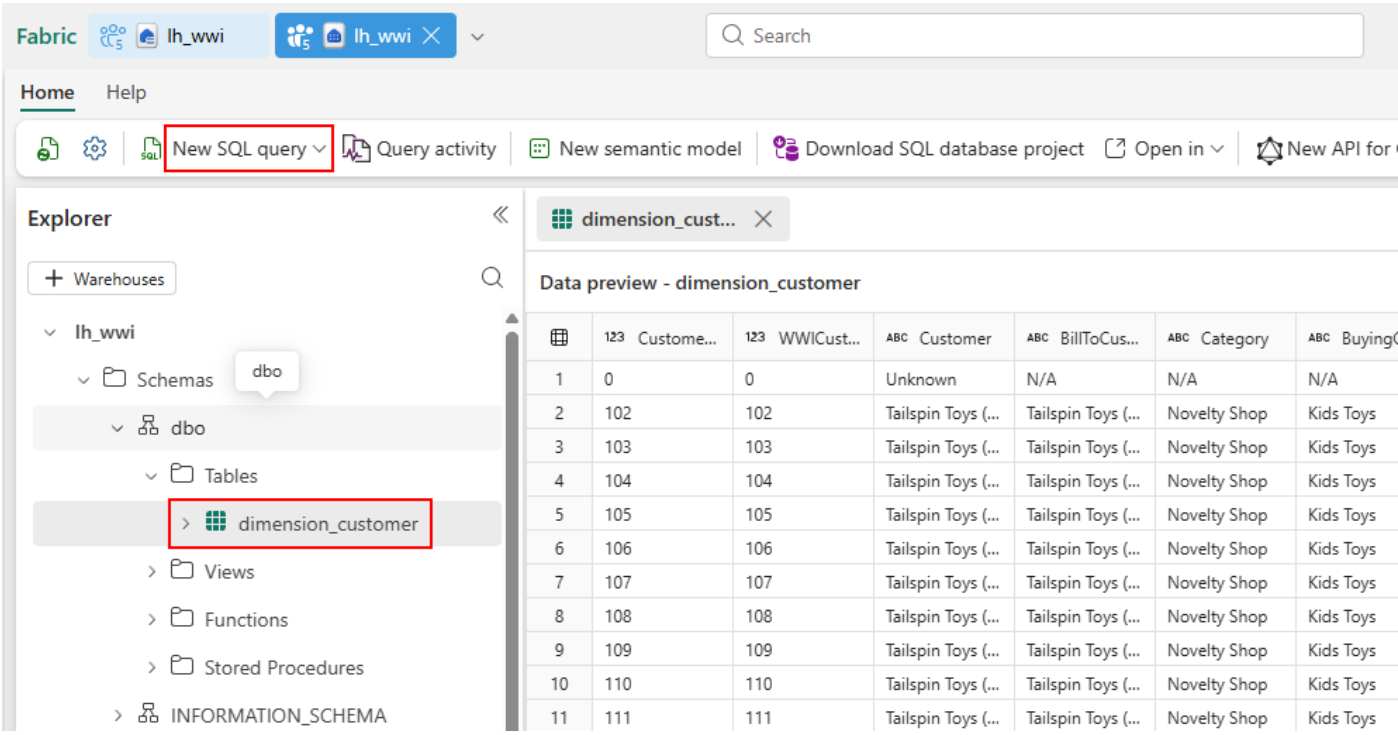


5. Once the dataflow is refreshed, select your lakehouse in the top menu bar to view the **dimension_customer** Delta table.

6. Select the table to preview its data. You can also use the SQL analytics endpoint of the lakehouse to query the data with SQL statements. Select **SQL analytics endpoint** from the **Lakehouse** dropdown menu at the top right of the screen.



7. Select the **dimension_customer** table to preview its data or select **New SQL query** to write your SQL statements.



8. The following sample query aggregates the row count based on the *BuyingGroup* column of the *dimension_customer* table. SQL query files are saved automatically for future reference, and you can rename or delete these files based on your need.

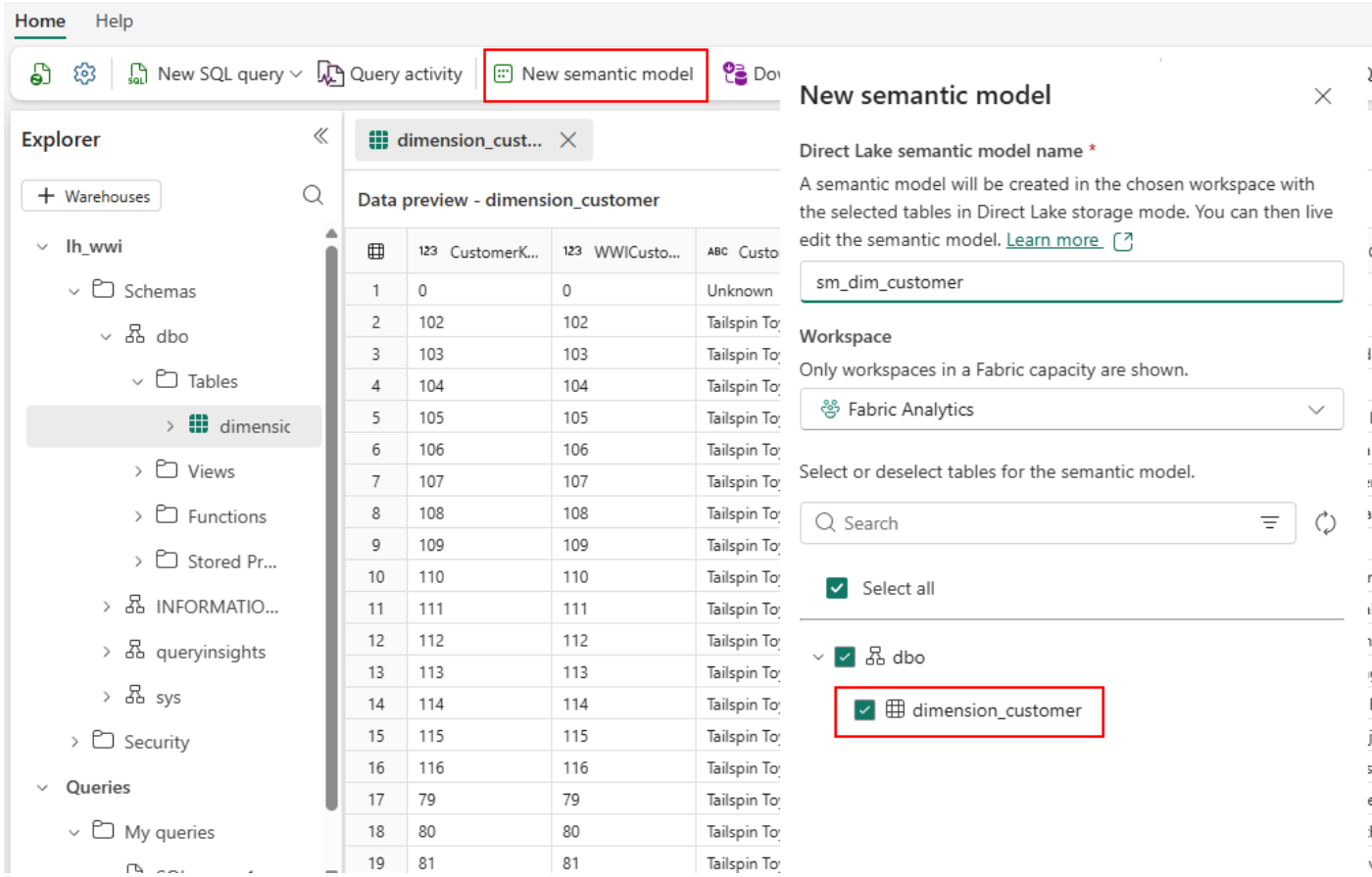
   To run the script, select the **Run** icon at the top of the script file.

```
SELECT BuyingGroup, Count(*) AS Total
FROM dimension_customer
GROUP BY BuyingGroup
```

Create a new semantic model

In this section, you add the tables to the semantic model so that you can use them to create reports.

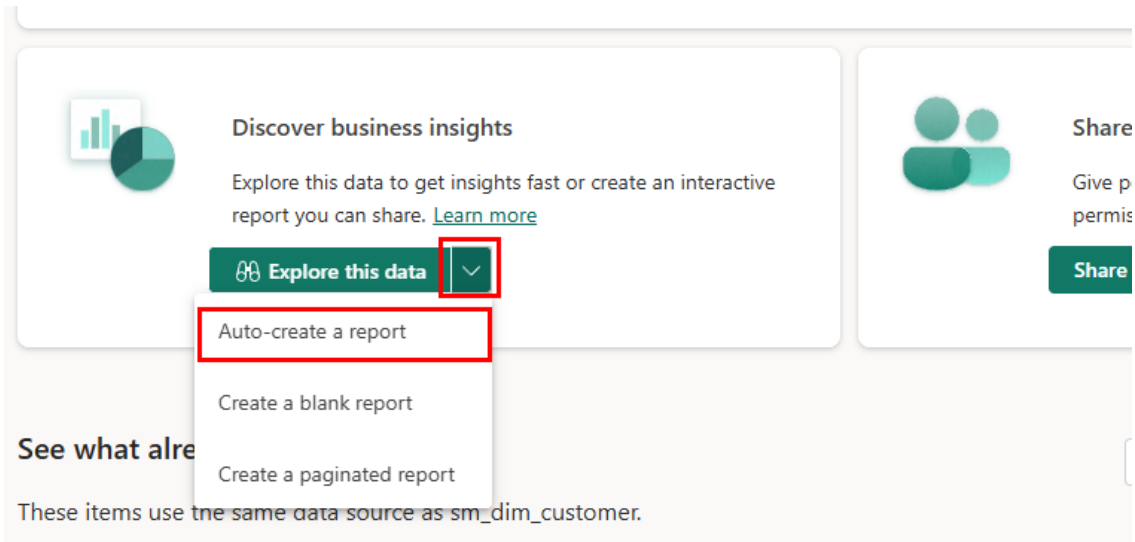
1. Open your lakehouse and switch to the **SQL analytics endpoint** view, select **New semantic model**, name the semantic model, assign a workspace, and select the tables that you want to add to the semantic model. In this case, select the **dimension_customer** table.



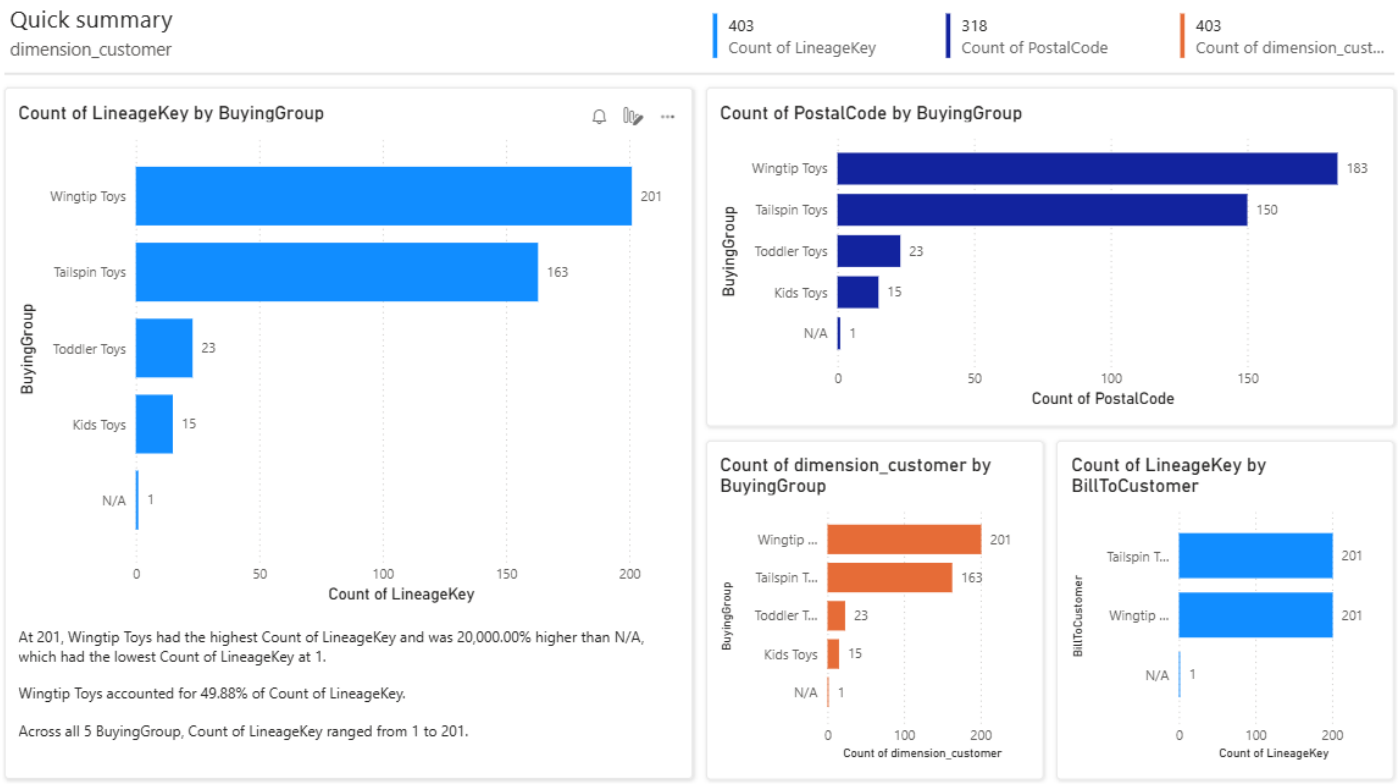
Build a report

In this section, you build a report from the ingested data.

1. Select the semantic model in your workspace, select the dropdown **Explore this data**, and then select **Auto-create a report**. In the next tutorial, we create a report from scratch.



2. The table is a dimension and there are no measures in it. Power BI creates a measure for the row count, aggregates it across different columns, and creates different charts as shown in the following image.



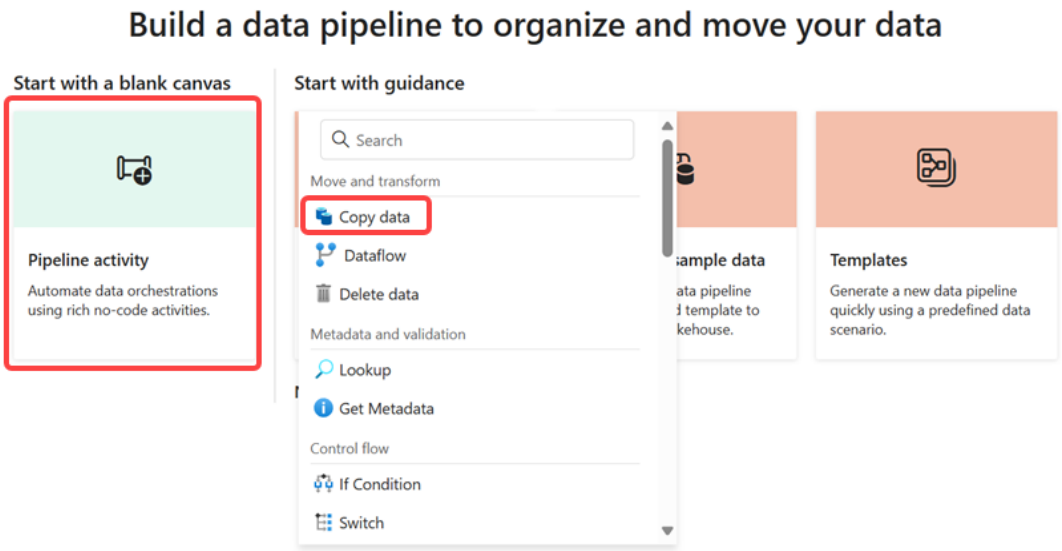
3. You can save this report for the future by selecting **Save** from the top ribbon. You can make more changes to this report to meet your requirements by including or excluding other tables or columns.

Exercise 2: Ingest data into the lakehouse

Ingest data

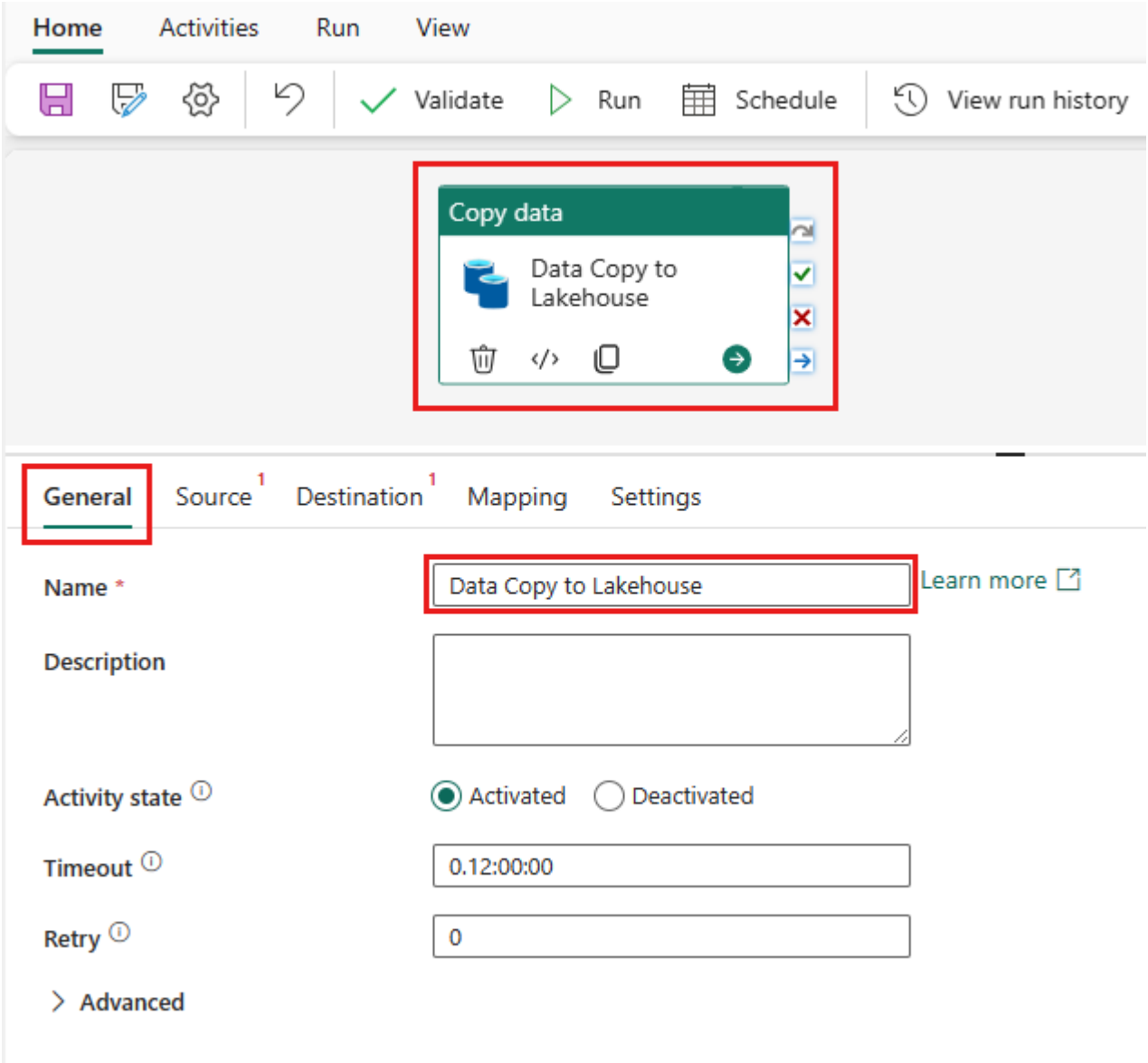
In this section, you use the **Copy data activity** of the Data Factory pipeline to ingest sample data from an Azure storage account to the **Files** section of the lakehouse you created earlier.

1. Select **Workspaces** in the left navigation pane, and then select your new workspace from the **Workspaces** menu. The items view of your workspace appears.
2. From the **New item** option in the workspace ribbon, select **Pipeline**.
3. In the **New pipeline** dialog box, specify the name as **IngestDataFromSourceToLakehouse** and select **Create**.
4. From your newly created pipeline, select **Pipeline activity** to add an activity to the pipeline and select **Copy data**. This action adds copy data activity to the pipeline canvas.



5. Select the newly added copy data activity from the canvas. Activity properties appear in a pane below the canvas (you might need to expand the pane upwards by dragging the top edge). From the **General** tab in the properties pane, type **Data Copy to Lakehouse** in the **Name** field. Leave

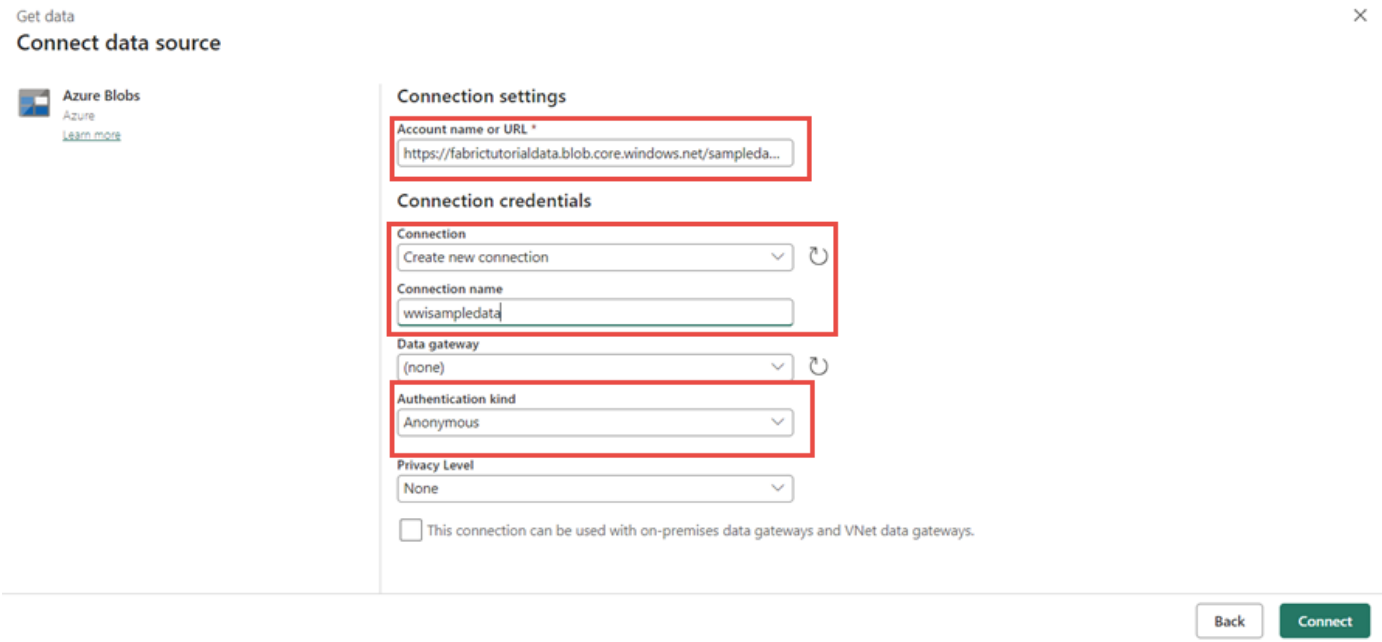
remaining properties to their default values.



6. From the **Source** tab of the selected copy data activity, open the **Connection** field and select **Browse all**. Choose data source window pops up, search and select **Azure blobs**. For this tutorial, all the sample data is available in a public container of Azure blob storage. You connect to this container to copy data from it.

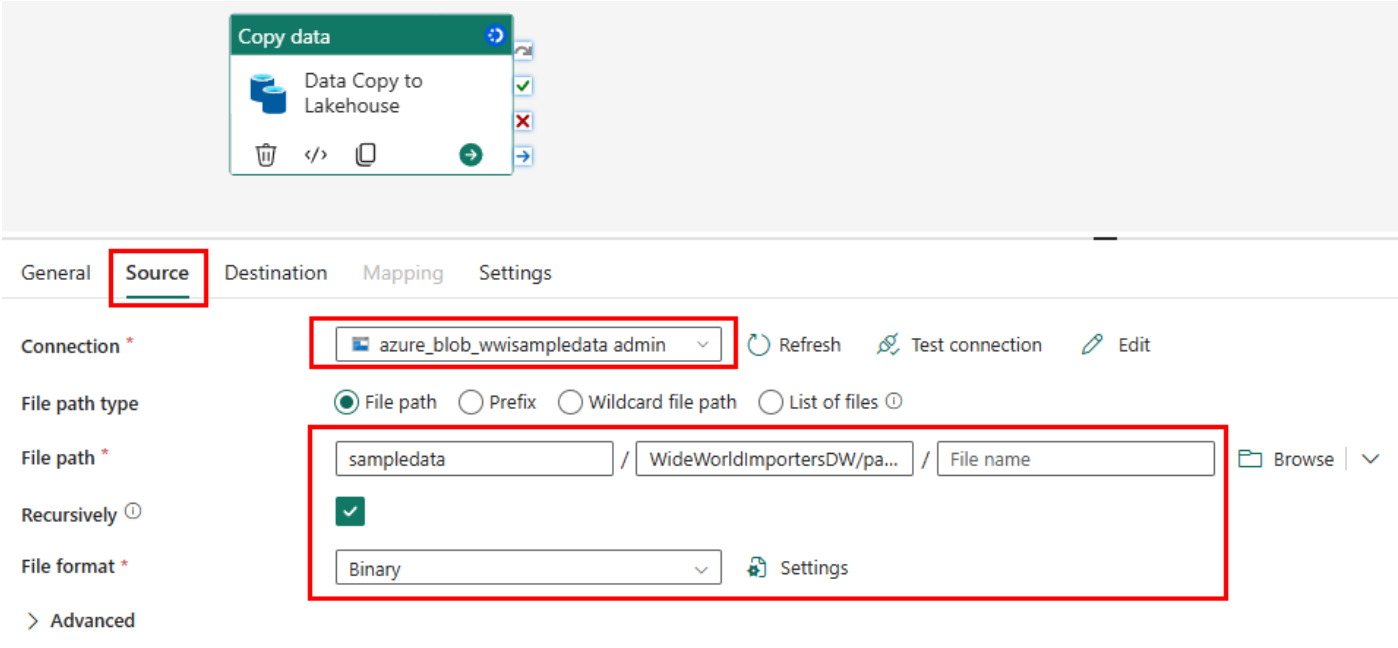
7. Enter the following details in the **Connection settings** window, and select **Connect** to create the connection to the data source.

Property	**Value**
Account name or URL	https://fabrictutorialdata.blob.core.windows.net/sampledata/
Connection	Create new connection
Connection name	wwisampledata
Authentication kind	Anonymous



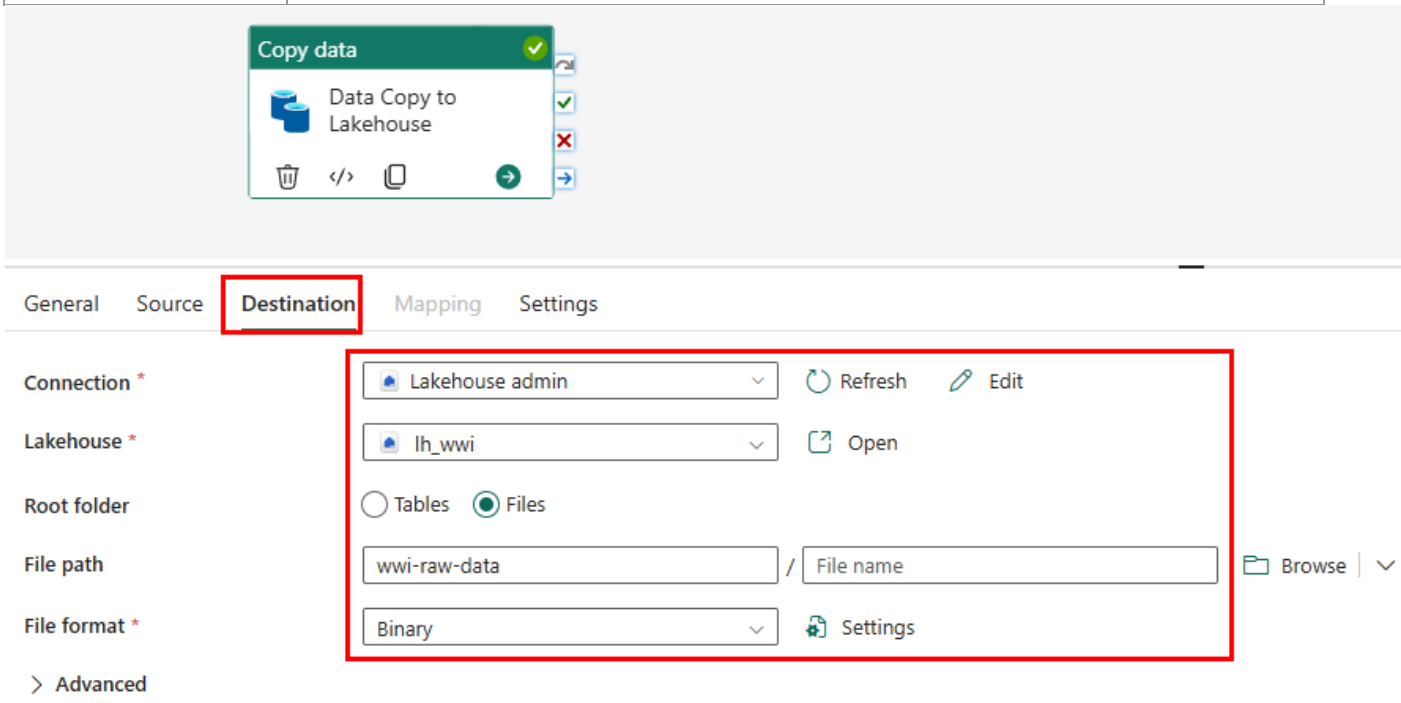
8. Once the new connection is created, return to the **Source** tab of the copy data activity, and the newly created connection is selected by default. Specify the following properties before moving to the destination settings.

Property	Value
Connection	wwisampledata
File path type	File path
File path	Container name (first text box): sampledata Directory name (second text box): WideWorldImportersDW/parquet
Recursively	Checked
File format	Binary



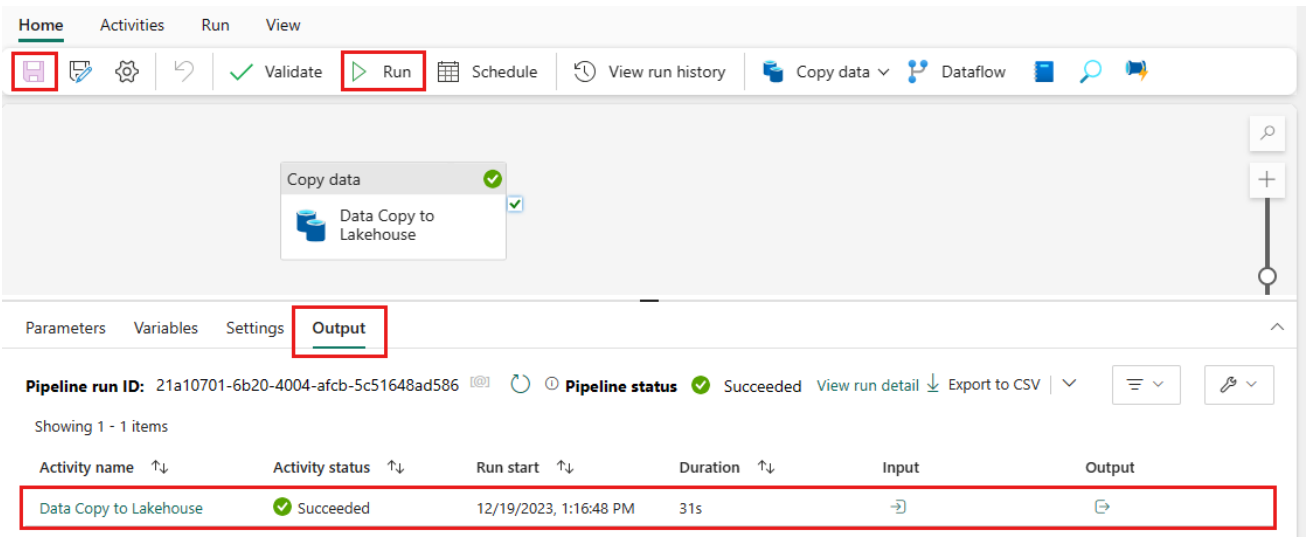
9. From the **Destination** tab of the selected copy data activity, specify the following properties:

Property	Value
Connection	wwilakehouse (choose your lakehouse if you named it differently)
Root folder	Files
File path	Directory name (first text box): wwi-raw-data
File format	Binary

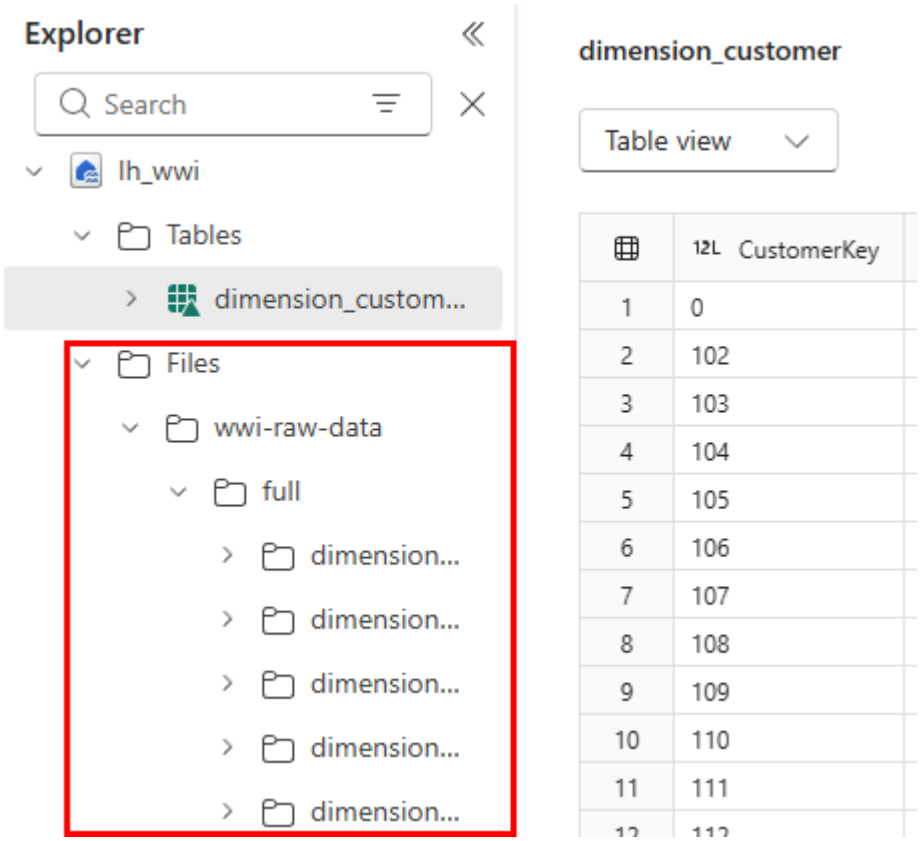


10. You have configured the copy data activity. Select the **Save** icon on the top ribbon (below Home) to save your changes, and select **Run** to execute your pipeline and its activity. You can also schedule pipelines to refresh data at defined intervals to meet your business requirements. For this tutorial, we run the pipeline only once by selecting **Run**.

11. This action triggers data copy from the underlying data source to the specified lakehouse and might take up to a minute to complete. You can monitor the execution of the pipeline and its activity under the **Output** tab. The activity status changes from **Queued** > **In progress** > **Succeeded**.



12. After the copy activity is successful, open your lakehouse (wwilakehouse) to view the data. Refresh the **Files** section to see the ingested data. A new folder **wwi-raw-data** appears in the files section, and data from Azure Blob tables is copied there.

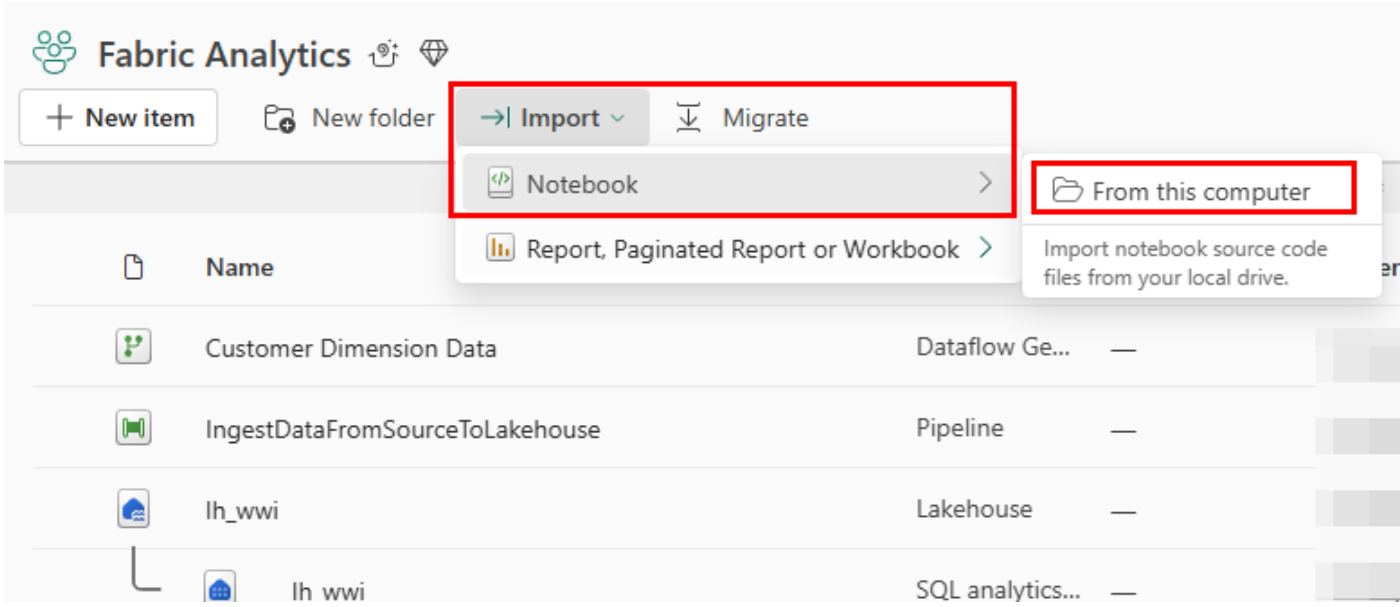


Exercise 4: Prepare and transform data in the lakehouse

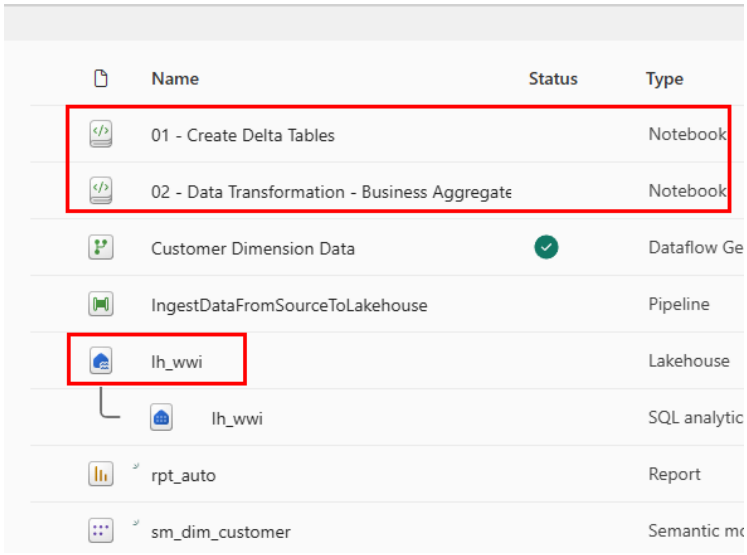
Prepare data

From the previous tutorial steps, we have raw data ingested from the source to the **Files** section of the lakehouse. Now you can transform that data and prepare it for creating Delta tables.

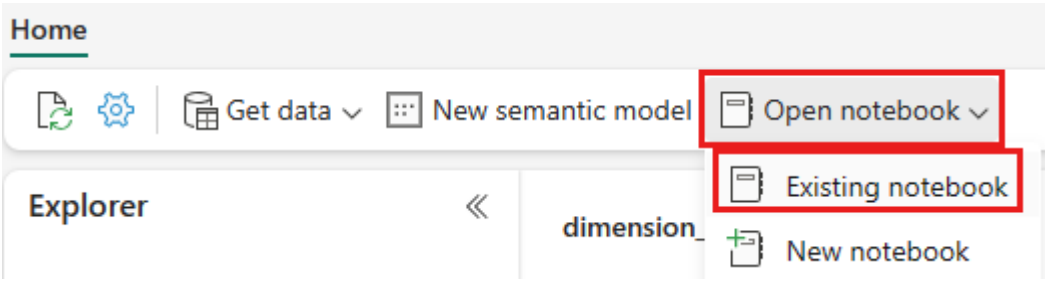
1. Download the notebooks from the Lakehouse Tutorial Source Code folder.
2. Open your workspace, select **Import** > **Notebook** > **From this computer**.
3. Select **Import notebook** from the **New** section at the top of the landing page.
4. Select **Upload** from the **Import status** pane that opens on the right side of the screen.
5. Select all the notebooks that you downloaded in first step of this section.



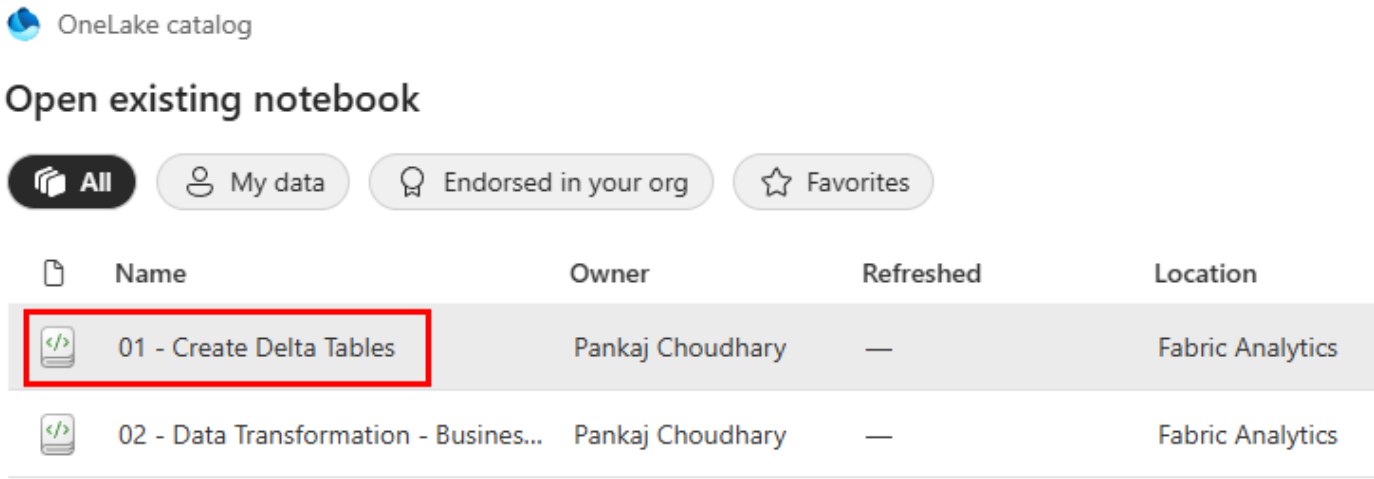
6. Select **Open**. A notification indicating the status of the import appears in the top right corner of the browser window.
7. After the import is successful, go to items view of the workspace and see the newly imported notebooks. Select **lh_wwi** lakehouse to open it.



8. Once the **lh_wwi** lakehouse is opened, select **Open notebook** > **Existing notebook** from the top navigation menu.



9. From the list of existing notebooks, select the **01 - Create Delta Tables** notebook and select **Open**.



10. In the open notebook in the lakehouse **Explorer**, you see the notebook is already linked to your opened lakehouse.

    Note:

    Fabric provides the V-order capability to write optimized Delta lake files. V-order often improves compression by three to four times, and up to 10 times, performance acceleration over the Delta Lake files that aren't optimized. Spark in Fabric dynamically optimizes partitions while generating files with a default 128 MB size. The target file size may be changed per workload requirements using configurations.

    With the optimize write capability, the Apache Spark engine reduces the number of files written and aims to increase individual file size of the written data.

11. Before you write data as Delta lake tables in the **Tables** section of the lakehouse, you use two Fabric features (**V-order** and **Optimize Write**) for optimized data writing and for improved reading performance. To enable these features in your session, set these configurations in the first cell of your notebook. Create a new cell and past the following code:

```
spark.conf.set("spark.sql.parquet.vorder.enabled", "true")
spark.conf.set("spark.microsoft.delta.optimizeWrite.enabled", "true")
spark.conf.set("spark.microsoft.delta.optimizeWrite.binSize", "1073741824")
```

When running a cell, you didn't have to specify the underlying Spark pool or cluster details because Fabric provides them through Live Pool. Every Fabric workspace comes with a default Spark pool, called Live Pool. This means when you create notebooks, you don't have to worry about specifying any Spark configurations or cluster details. When you execute the first notebook command, the live pool is up and running in a few seconds. And the Spark session is established and it starts executing the code. Subsequent code execution is almost instantaneous in this notebook while the Spark session is active.

12. Next, you read raw data from the **Files** section of the lakehouse, and add more columns for different date parts as part of the transformation. Finally, you use partition By Spark API to partition the data before writing it as Delta table format based on the newly created data part columns (**Year** and **Quarter**).

```
from pyspark.sql.functions import col, year, month, quarter
table_name = 'fact_sale'
df = spark.read.format("parquet").load('Files/wwi-raw-data/full/fact_sale_1y_full')
df = df.withColumn('Year', year(col("InvoiceDateKey")))
df = df.withColumn('Quarter', quarter(col("InvoiceDateKey")))
df = df.withColumn('Month', month(col("InvoiceDateKey")))
df.write.mode(“overwrite”).format(“delta”).partitionBy(“Year”,”Quarter”).save(“Tables/” + table_name)
```

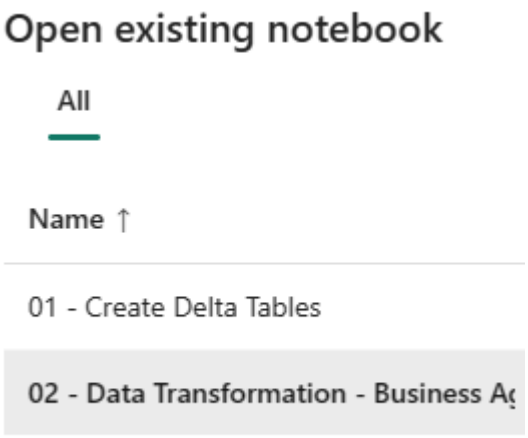
13. After the fact tables load, you can move on to loading data for the rest of the dimensions. The following cell creates a function to read raw data from the **Files** section of the lakehouse for each of the table names passed as a parameter. Next, it creates a list of dimension tables. Finally, it loops through the list of tables and creates a Delta table for each table name that's read from the input parameter. Note that the script drops the column named Photo in this example because the column isn't used.

```
from pyspark.sql.types import *
def loadFullDataFromSource(table_name):
    df = spark.read.format("parquet").load('Files/wwi-raw-data/full/' + table_name)
    df = df.drop("Photo")
    df.write.mode("overwrite").format("delta").save("Tables/" + table_name)

full_tables = [
    'dimension_city',
    'dimension_date',
    'dimension_employee',
    'dimension_stock_item'
    ]

for table in full_tables:
    loadFullDataFromSource(table)
```

14. To validate the created tables, right-click and select refresh on the **lh_wwi** lakehouse. The tables appear.

15. Go the items view of the workspace again and select the **lh_wwi** lakehouse to open it.

16. Now, open the second notebook. In the lakehouse view, select **Open notebook** > **Existing notebook** from the ribbon.
17. From the list of existing notebooks, select the **02 - Data Transformation - Business** notebook to open it.



18. In the open notebook in the lakehouse **Explorer**, you see the notebook is already linked to your opened lakehouse.
19. An organization might have data engineers working with Scala/Python and other data engineers working with SQL (Spark SQL or T-SQL), all working on the same copy of the data. Fabric makes it possible for these different groups, with varied experience and preference, to work and collaborate. The two different approaches transform and generate business aggregates. You can pick the one suitable for you or mix and match these approaches based on your preference without compromising on the performance:
    - **Approach #1** - Use PySpark to join and aggregates data for generating business aggregates. This approach is preferable to someone with a programming (Python or PySpark) background.
    - **Approach #2** - Use Spark SQL to join and aggregates data for generating business aggregates. This approach is preferable to someone with SQL background, transitioning to Spark.
20. **Approach #1 (sale_by_date_city)** - Use PySpark to join and aggregate data for generating business aggregates. With the following code, you create three different Spark dataframes, each referencing an existing Delta table. Then you join these tables using the dataframes, do group by to generate aggregation, rename a few of the columns, and finally write it as a Delta table in the **Tables** section of the lakehouse to persist with the data.

    ```
    df_fact_sale = spark.read.table("wwilakehouse.fact_sale")
    df_dimension_date = spark.read.table("wwilakehouse.dimension_date")
    df_dimension_city = spark.read.table("wwilakehouse.dimension_city")
    ```

    Add the following code to the same cell to join these tables using the dataframes created earlier. Group by to generate aggregation, rename a few of the columns, and finally write it as a Delta table in the **Tables** section of the lakehouse.

```
sale_by_date_city = df_fact_sale.alias("sale") \
.join(df_dimension_date.alias("date"), df_fact_sale.InvoiceDateKey == df_dimension_date.Date, "inner") \
.join(df_dimension_city.alias("city"), df_fact_sale.CityKey == df_dimension_city.CityKey, "inner") \
.select("date.Date", "date.CalendarMonthLabel", "date.Day", "date.ShortMonth", "date.CalendarYear",
"city.City", "city.StateProvince", "city.SalesTerritory", "sale.TotalExcludingTax", "sale.TaxAmount",
"sale.TotalIncludingTax", "sale.Profit")\
.groupBy("date.Date", "date.CalendarMonthLabel", "date.Day", "date.ShortMonth", "date.CalendarYear",
"city.City", "city.StateProvince", "city.SalesTerritory")\
.sum("sale.TotalExcludingTax", "sale.TaxAmount", "sale.TotalIncludingTax", "sale.Profit")\
.withColumnRenamed("sum(TotalExcludingTax)", "SumOfTotalExcludingTax")\
.withColumnRenamed("sum(TaxAmount)", "SumOfTaxAmount")\
.withColumnRenamed("sum(TotalIncludingTax)", "SumOfTotalIncludingTax")\
.withColumnRenamed("sum(Profit)", "SumOfProfit")\
.orderBy("date.Date", "city.StateProvince", "city.City")

sale_by_date_city.write.mode("overwrite").format("delta").option("overwriteSchema",
"true").save("Tables/aggregate_sale_by_date_city")
```

21. **Approach #2 (sale_by_date_employee)** - Use Spark SQL to join and aggregate data for generating business aggregates. With the following code, you create a temporary Spark view by joining three tables, do group by to generate aggregation, and rename a few of the columns. Finally, you read from the temporary Spark view and finally write it as a Delta table in the **Tables** section of the lakehouse to persist with the data.

    You create a temporary Spark view by joining three tables, do group by to generate aggregation, and rename a few of the columns.

```
%%sql
CREATE OR REPLACE TEMPORARY VIEW sale_by_date_employee
AS
SELECT
     DD.Date, DD.CalendarMonthLabel
 , DD.Day, DD.ShortMonth Month, CalendarYear Year
    ,DE.PreferredName, DE.Employee
    ,SUM(FS.TotalExcludingTax) SumOfTotalExcludingTax
    ,SUM(FS.TaxAmount) SumOfTaxAmount
    ,SUM(FS.TotalIncludingTax) SumOfTotalIncludingTax
    ,SUM(Profit) SumOfProfit
FROM wwilakehouse.fact_sale FS
INNER JOIN wwilakehouse.dimension_date DD ON FS.InvoiceDateKey = DD.Date
INNER JOIN wwilakehouse.dimension_Employee DE ON FS.SalespersonKey = DE.EmployeeKey
GROUP BY DD.Date, DD.CalendarMonthLabel, DD.Day, DD.ShortMonth, DD.CalendarYear, DE.PreferredName,
DE.Employee
ORDER BY DD.Date ASC, DE.PreferredName ASC, DE.Employee ASC
```

    You read from the temporary Spark view created in the previous cell and finally write it as a Delta table in the **Tables** section of the lakehouse.

```
sale_by_date_employee = spark.sql("SELECT * FROM sale_by_date_employee")
sale_by_date_employee.write.mode("overwrite").format("delta").option("overwriteSchema",
"true").save("Tables/aggregate_sale_by_date_employee")
```

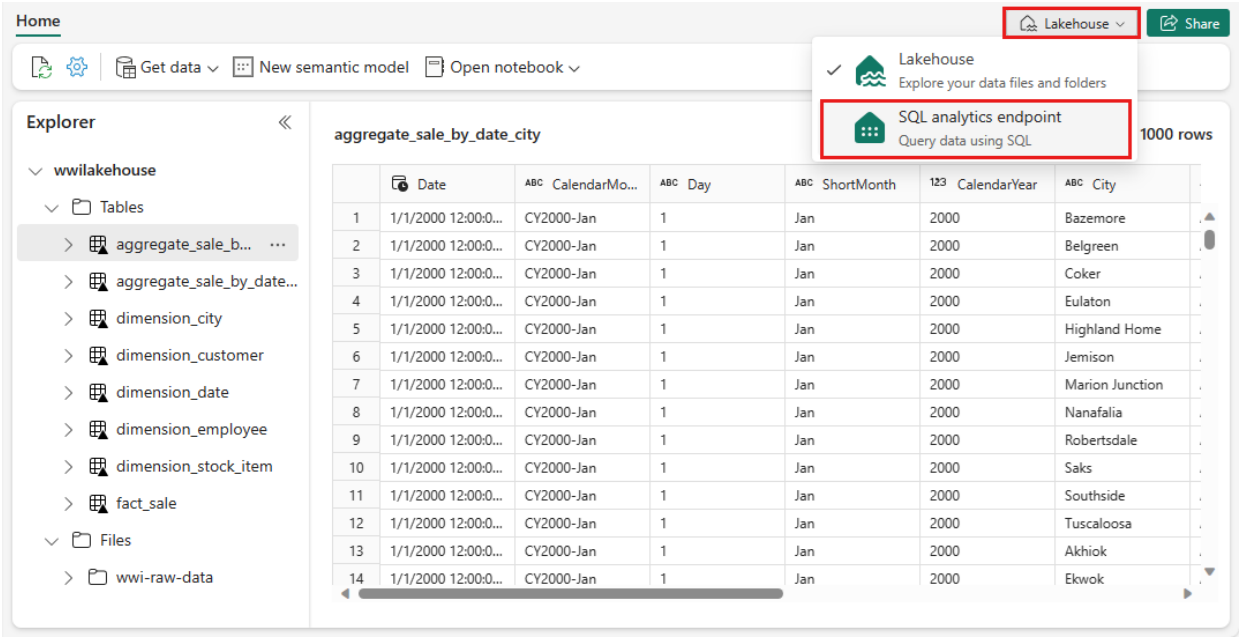
22. To validate the created tables, right-click and select **Refresh** on the **wwilakehouse** lakehouse. The aggregate tables appear.

The two approaches produce a similar outcome. To minimize the need for you to learn a new technology or compromise on performance, choose the approach that best suits your background and preference.

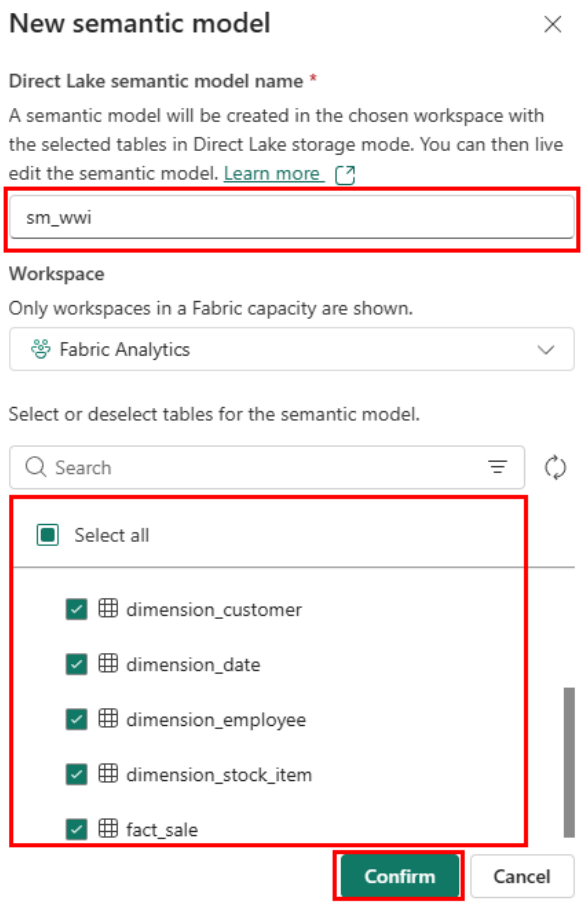
Exercise 5: Building reports in Microsoft Fabric

Create sematic model

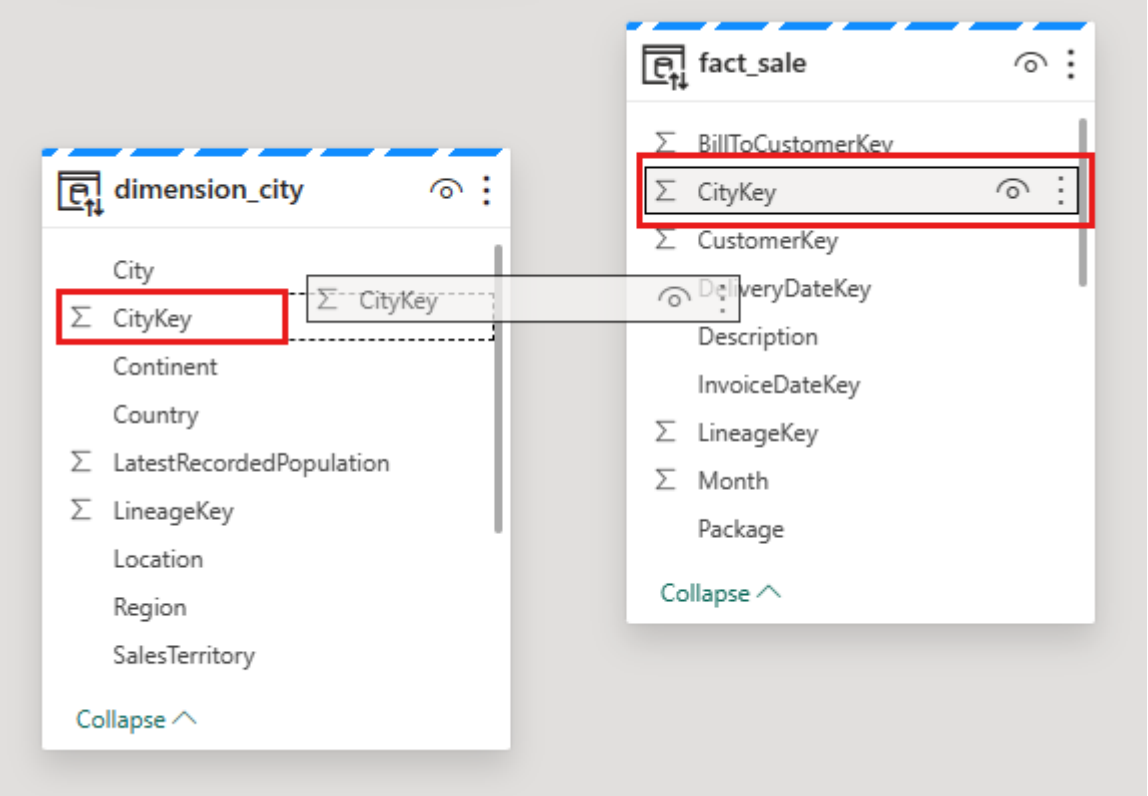
1. From your **lh_wwi** lakehouse, select **SQL analytics endpoint** from the **Lakehouse** dropdown menu at the top right of the screen.



2. From the SQL analytics endpoint pane, you should be able to see all the tables you created. If you don't see them yet, select the **Refresh** icon at the top. Next, select the **New semantic model** and select all the table dimension and fact table.

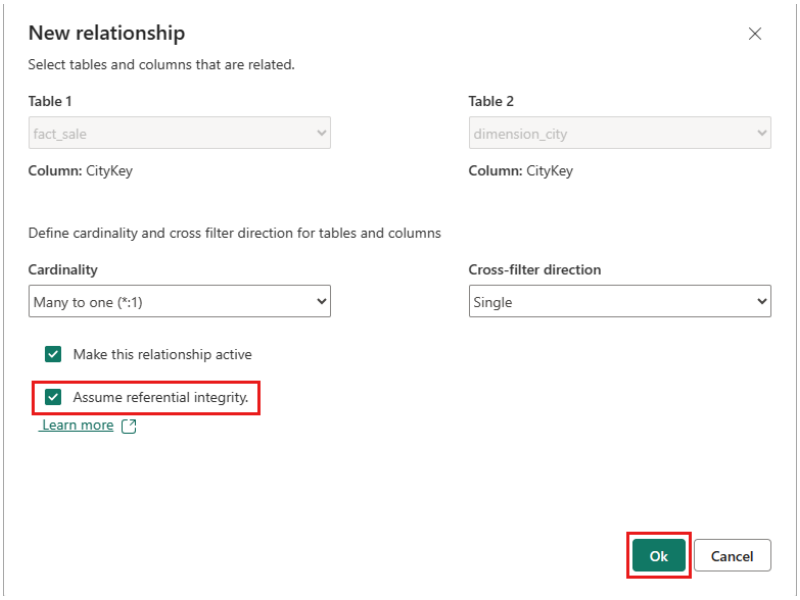


3. From top right corner switch to **Editing.** For this data model, you need to define the relationship between different tables so that you can create reports and visualizations based on data coming across different tables. From the **fact_sale** table, drag the **CityKey** field and drop it on the **CityKey** field in the **dimension_city** table to create a relationship. The **New relationship** dialog box appears.



4. In the **New relationship** dialog box:

   - Table 1 is populated with **fact_sale** and the column of CityKey.
   - Table 2 is populated with **dimension_city** and the column of CityKey.
   - Cardinality: **Many to one (*:1)**.
   - Cross filter direction: **Single**.
   - Leave the box next to **Make this relationship active** selected.
   - Select the box next to **Assume referential integrity**.
   - Select **Save**.

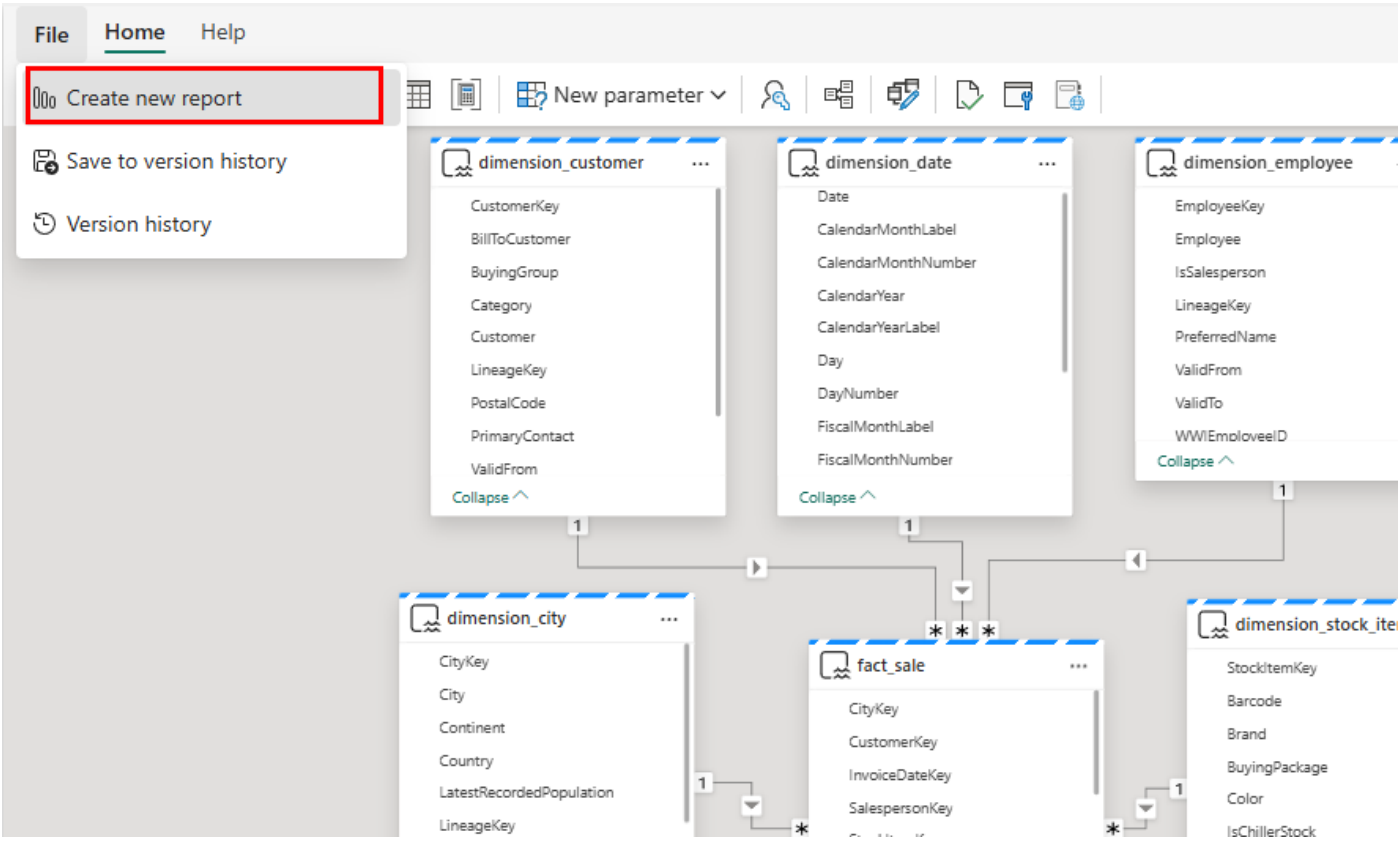


Note:

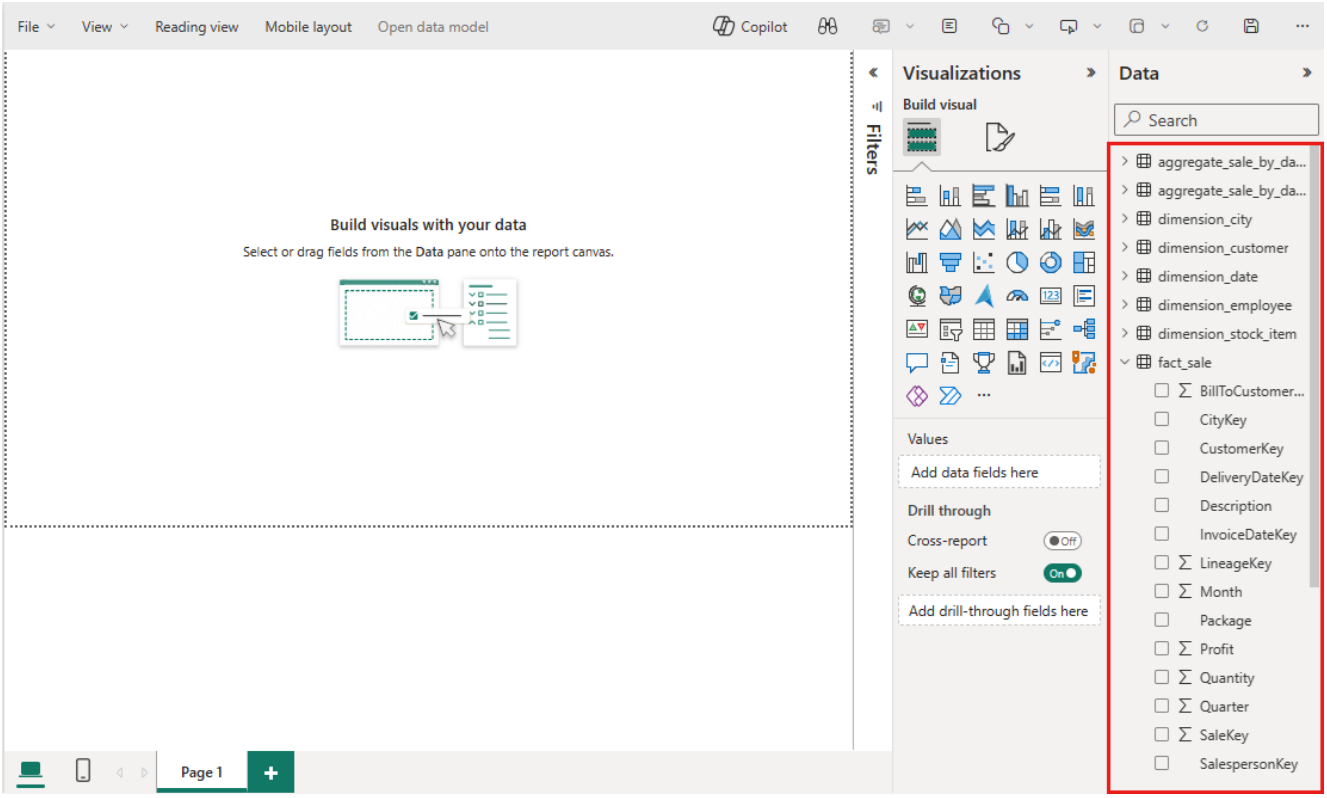
When defining relationships for this report, make sure you have a many to one relationship from the **fact_sale** table (Table 1) to the **dimension_*** tables (Table 2) and not vice versa.

5. Next, add these relationships with the same **New relationship** settings shown in the previous step, but with the following tables and columns:
    - StockItemKey(fact_sale) - StockItemKey(dimension_stock_item)
    - Salespersonkey(fact_sale) - EmployeeKey(dimension_employee)
    - CustomerKey(fact_sale) - CustomerKey(dimension_customer)
    - InvoiceDateKey(fact_sale) - Date(dimension_date)

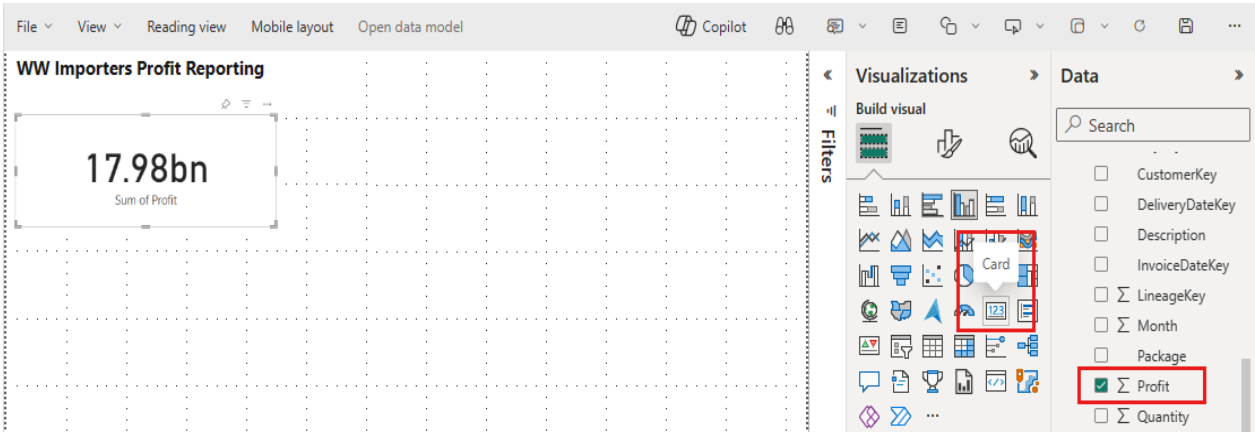
After you add these relationships, your data model is ready for reporting as shown in the following image:



6. Select **New report** to start creating reports/dashboards in Power BI. On the Power BI report canvas, you can create reports to meet your business requirements by dragging required columns from the **Data** pane to the canvas and using one or more of available visualizations.



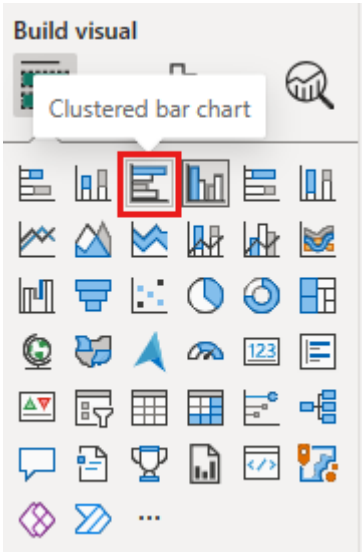
7. Add a title:
    a. In the Ribbon, select **Text box**.
    b. Type in **WW Importers Profit Reporting**.
    c. Highlight the text, increase the size to 20, and move it to the upper left of the report page.
8. Add a Card:
    a. On the **Data** pane, expand **fact_sale**, and check the box next to **Profit**. This selection creates a column chart and adds the field to the Y-axis.
    b. With the chart selected, select the **Card** visual in the visualization pane. This selection converts the visual to a card.
    c. Place the card under the title.



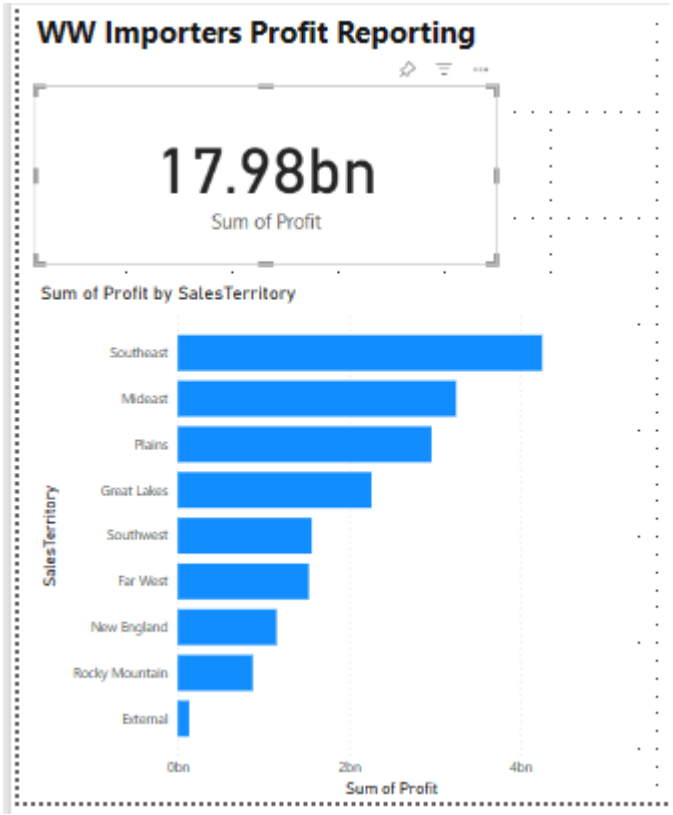
9. Add a Bar chart:
    a. On the **Data** pane, expand **fact_sales** and check the box next to **Profit**. This selection creates a column chart and adds the field to the X-axis.

b. On the **Data** pane, expand **dimension_city** and check the box for **SalesTerritory**. This selection adds the field to the Y-axis.

c. With the bar chart selected, select the **Clustered bar chart** visual in the visualization pane. This selection converts the column chart into a bar chart.



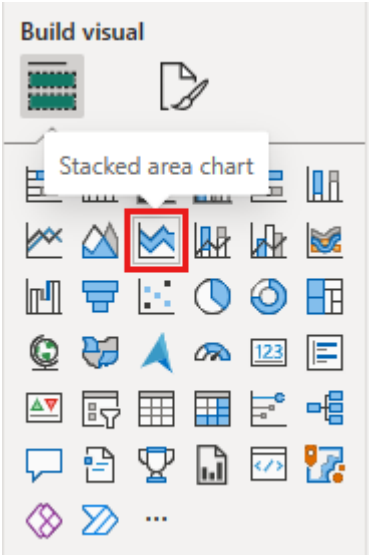
d. Resize the Bar chart and move it under the title and Card.



10. Click anywhere on the blank canvas (or press the Esc key) to deselect the bar chart.

11. Build a stacked area chart visual:

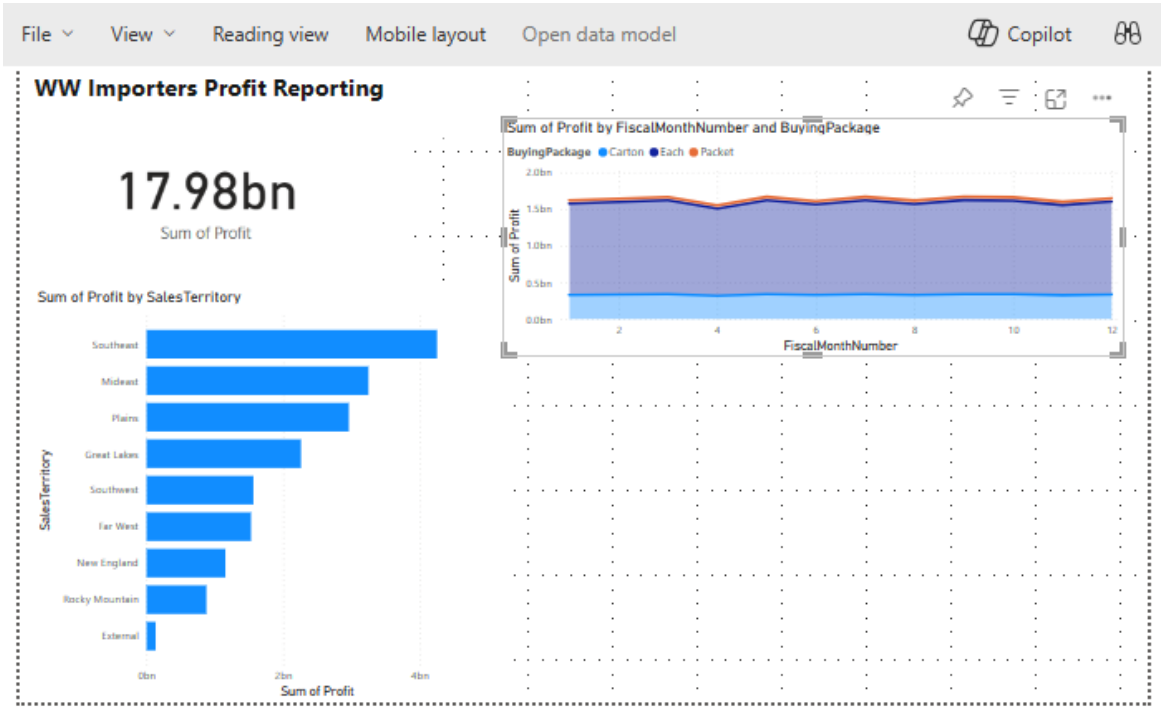
a. On the **Visualizations** pane, select the **Stacked area chart** visual.



b. Reposition and resize the stacked area chart to the right of the card and bar chart visuals created in the previous steps.

c. On the **Data** pane, expand **fact_sales** and check the box next to **Profit**. Expand **dimension_date** and check the box next to **FiscalMonthNumber**. This selection creates a filled line chart showing profit by fiscal month.

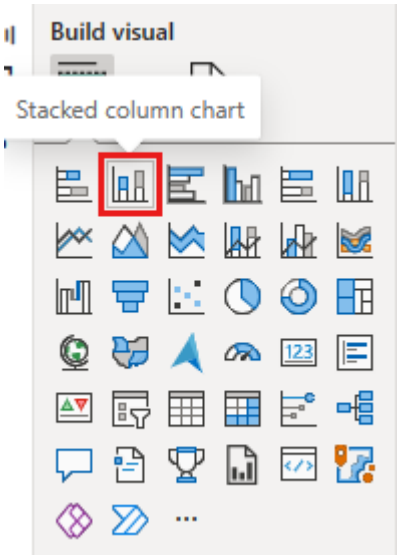
d. On the **Data** pane, expand **dimension_stock_item** and drag **BuyingPackage** into the Legend field well. This selection adds a line for each of the Buying Packages.



12. Click anywhere on the blank canvas (or press the Esc key) to deselect the stacked area chart.

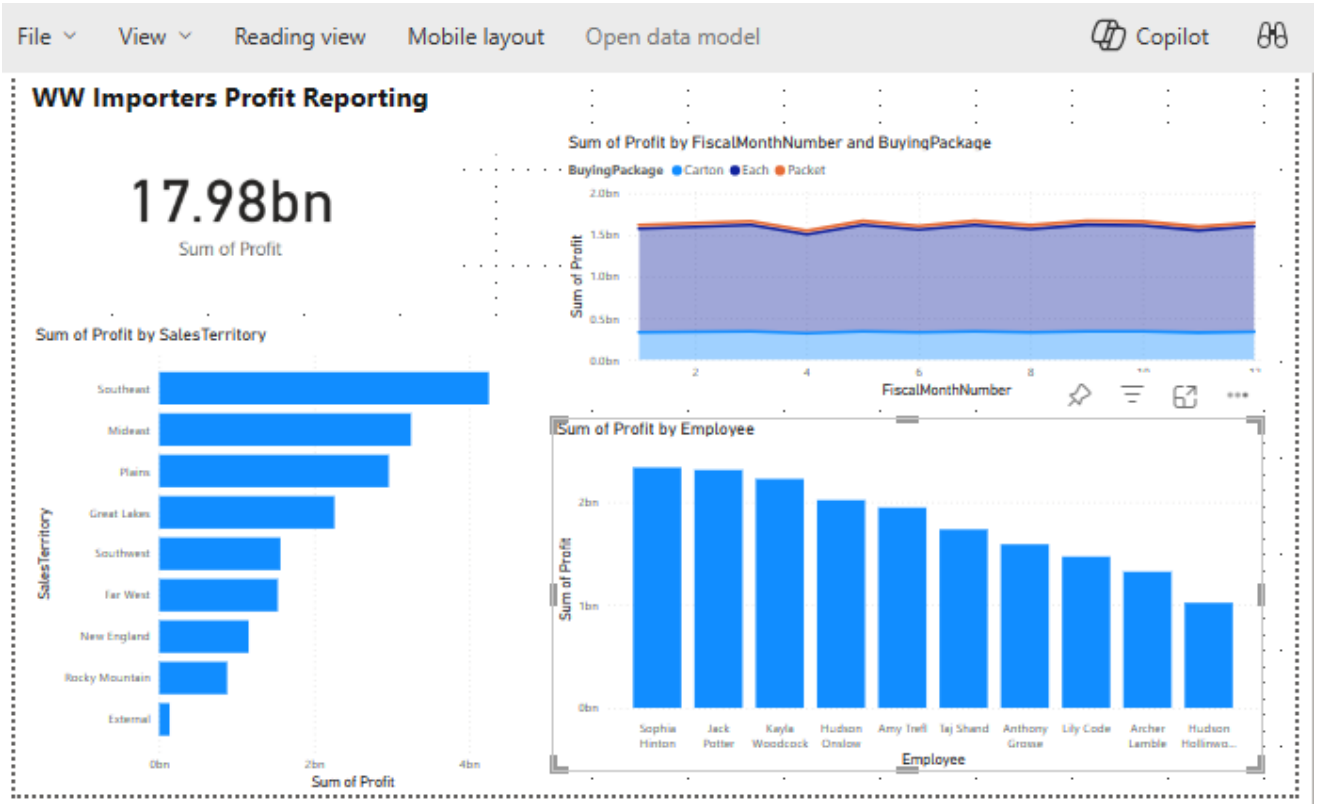
13. Build a column chart:

a. On the **Visualizations** pane, select the **Stacked column chart** visual.

b. 

a. On the **Data** pane, expand **fact_sales** and check the box next to **Profit**. This selection adds the field to the Y-axis.

b. On the **Data** pane, expand **dimension_employee** and check the box next to **Employee**. This selection adds the field to the X-axis.



14. Click anywhere on the blank canvas (or press the Esc key) to deselect the chart.

15. From the ribbon, select **File** > **Save**.

16. Enter the name of your report as **Profit Reporting**.

17. Select **Save**.