



# **END TO END DATA ANALYTICS SOLUTION USING MICROSOFT FABRIC WITH WWI DATASET.**

*A step-by-step guide to ingest, transform, model, and visualize  
business data using the Lakehouse paradigm.*

Gosala Kasi Raju  
[kasirajugosala@outlook.com](mailto:kasirajugosala@outlook.com)

# LAKEHOUSE END TO END SCENARIO

## PROJECT OVERVIEW

Implement a full end-to-end data analytics solution using Microsoft Fabric Lakehouse to ingest, transform, store, and visualize data from the **Wide World Importers (WWI) dataset**.

## PROJECT PURPOSE

To demonstrate how organizations can use Microsoft Fabric to build scalable, collaborative analytics solutions with minimal infrastructure setup, integrating raw data ingestion, Delta Lake storage, semantic modeling, and Power BI reporting.

## PREREQUISITES

### **1. Environment:**

- Power BI account with Microsoft Fabric Trial enabled.
- A Fabric-enabled workspace created in Power BI.

### **2. Tools Required:**

- Lakehouse – for storing raw and transformed data.
- **Dataflows Gen2** – to ingest .csv files.
- **Pipelines** – to automate ZIP data ingestion.
- **Notebooks** – for data transformation using PySpark or SQL.
- **Power BI Reports** – for visualizing the data.
- **SQL Analytics Endpoint** – to run SQL queries.

### **3. Sample Data:**

- dimension\_customer.csv:

[Download CSV](#)

- wwi-sample-dataset.zip:

[Download ZIP](#)

## PROJECT STEPS

### ➤ **STEP-1: Environment Setup**

- Create a Fabric – enabled workspace
- Create a Workspace named as Project WS1
- Create a new Lakehouse (wwiLakehouse) in **Project WS1**

### ➤ **STEP-2: Ingest Initial Data (CSV)**

- Use Dataflow Gen2 to load dimension\_customer.csv
- Transform : Promote headers, save, and load as table.

### ➤ STEP-3: Ingest Full Dataset ( ZIP)

- Create a Pipeline to ingest wwi-sample-dataset.zip via HTTP.
- Enable binary copy and extract ZIP into the Lakehouse Files.
- Rename the imported folder to wwi-raw-data.

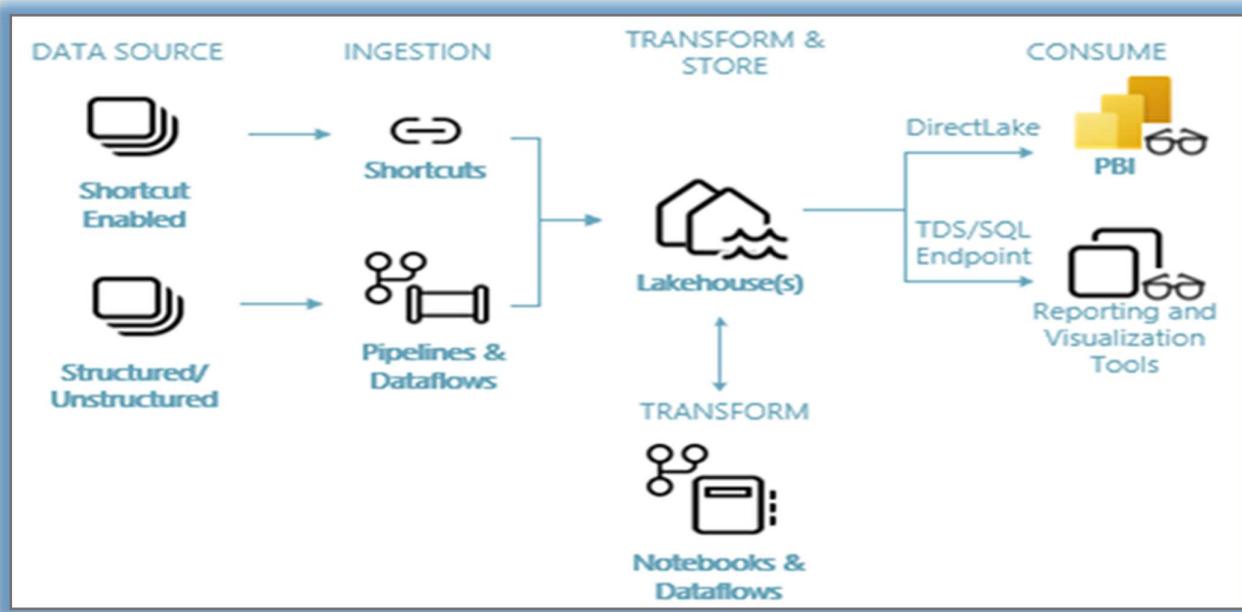
### ➤ STEP-4: Data Transformation

- Import and run Notebook 01 – Create Delta Tables:
  - Read raw files, transform, write to Delta tables.
- Run Notebook 02 – Transformation – Business:
  - Create business aggregates using PySpark or SQL

### ➤ STEP-5: Build Report

- Add tables to semantic model via SQL endpoint.
- Enable sync with power BI semantic model.
- Use Auto-Create Report in Power BI and save.

## ARCHITECTURE DIAGRAM

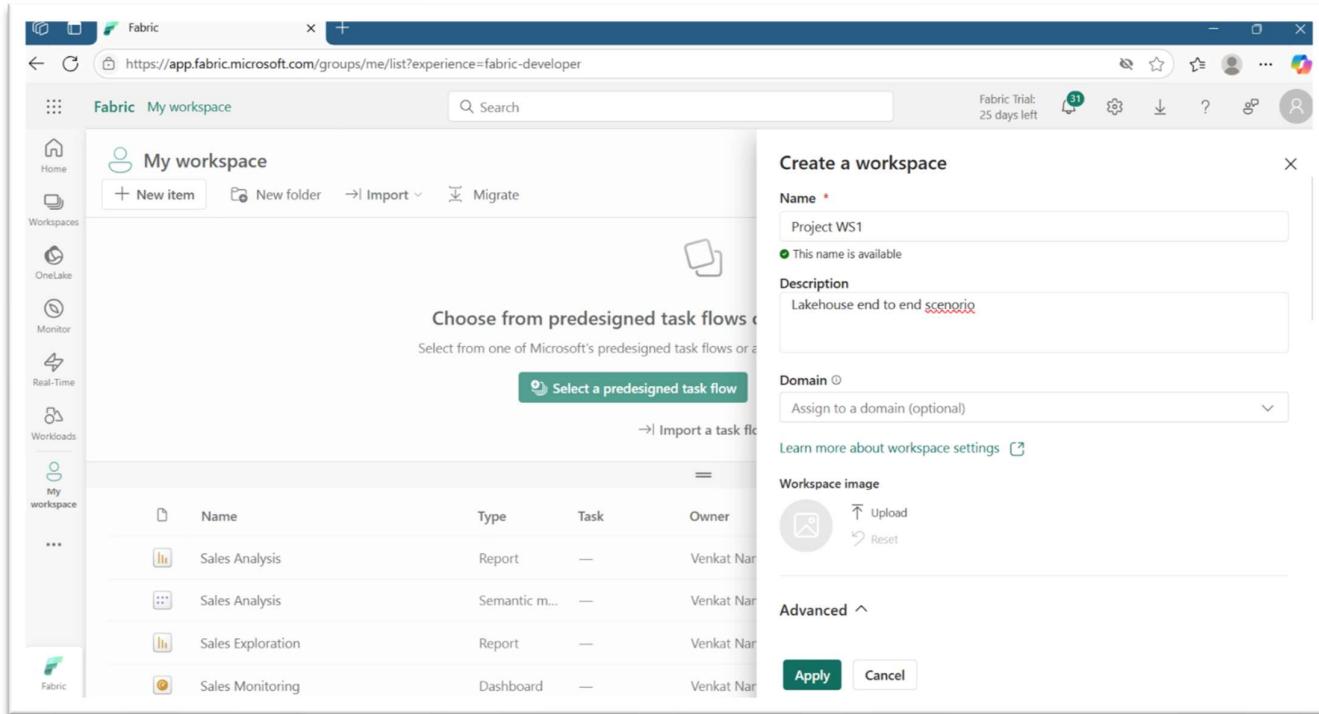


## THE PROCESS OF AN ENTIRE IMPLEMENTATION

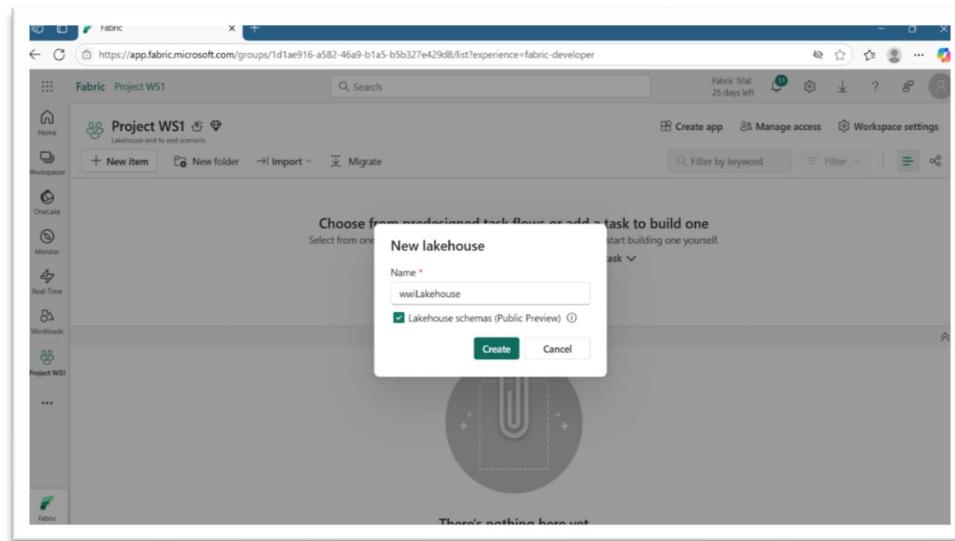
### STEP – 1

- ✓ Create a workspace named **Project WS1**.

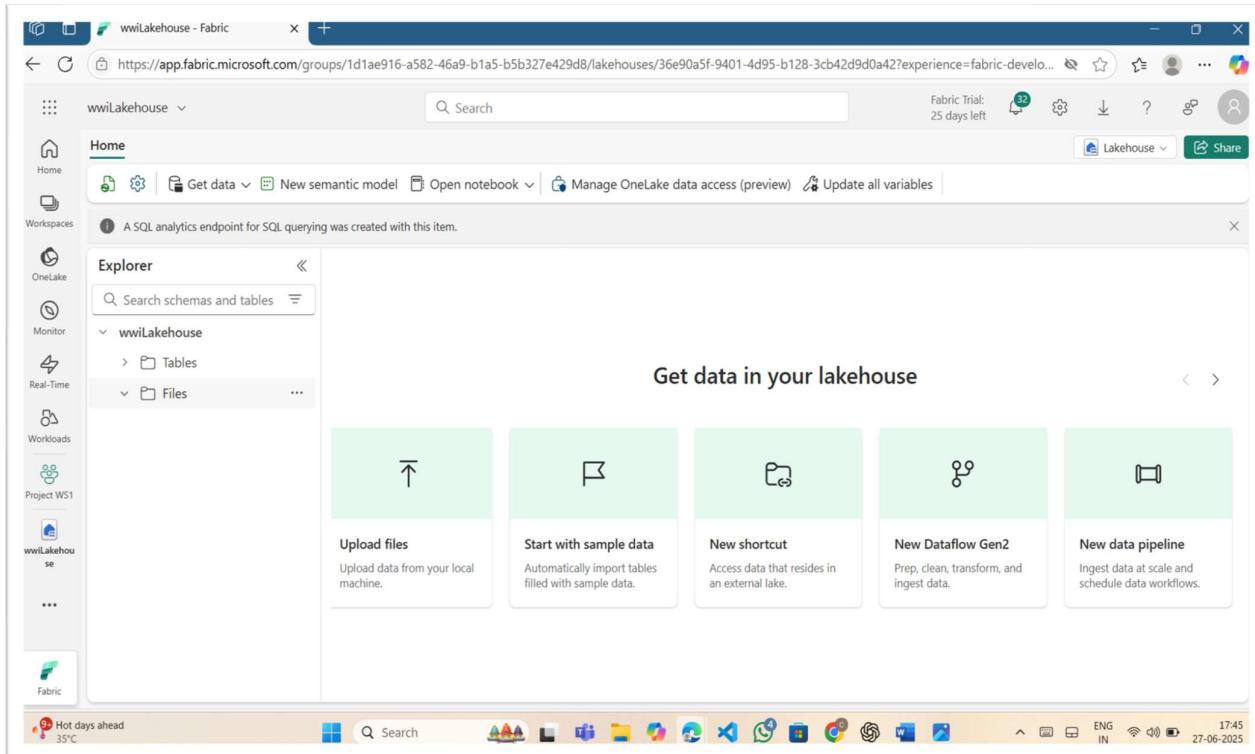
- ✓ Select License Mode – Trail
- ✓ Select Semantic Model Storage Format – Small SMSF
- ✓ Click Apply.



- ✓ Open the **Project WS1** workspace. From the workspace, select **New item**, then choose **Lakehouse**.
- ✓ In the **New lakehouse** dialog box, enter **wwiLakehouse** in the name field.
- ✓ Click **Create** to open the Lakehouse.

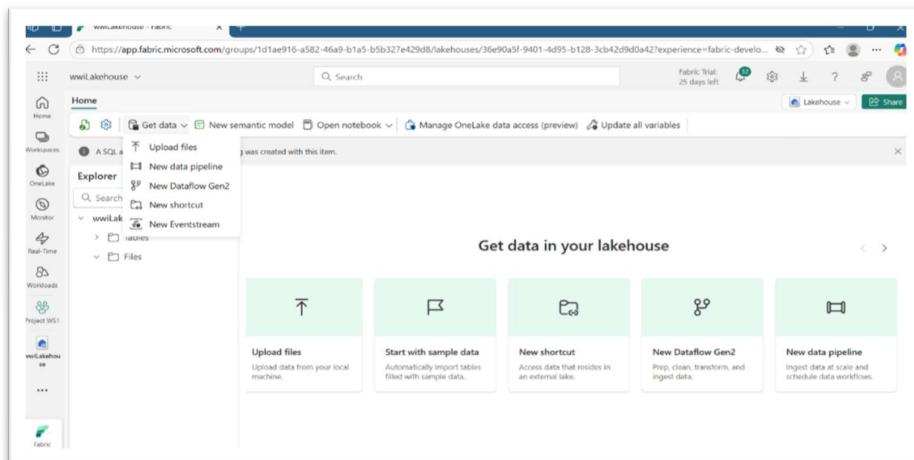


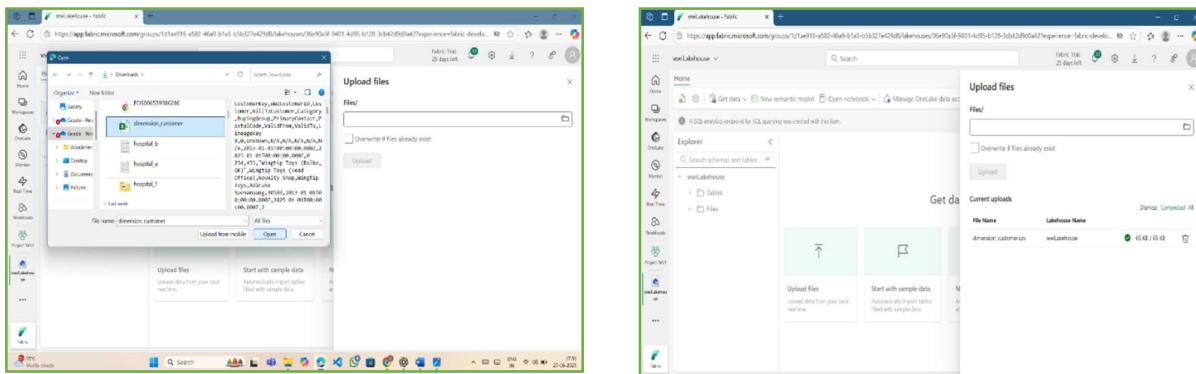
- ✓ The Lakehouse has already been created, as shown in the figure below. It includes the setup to ingest sample data and build a report.



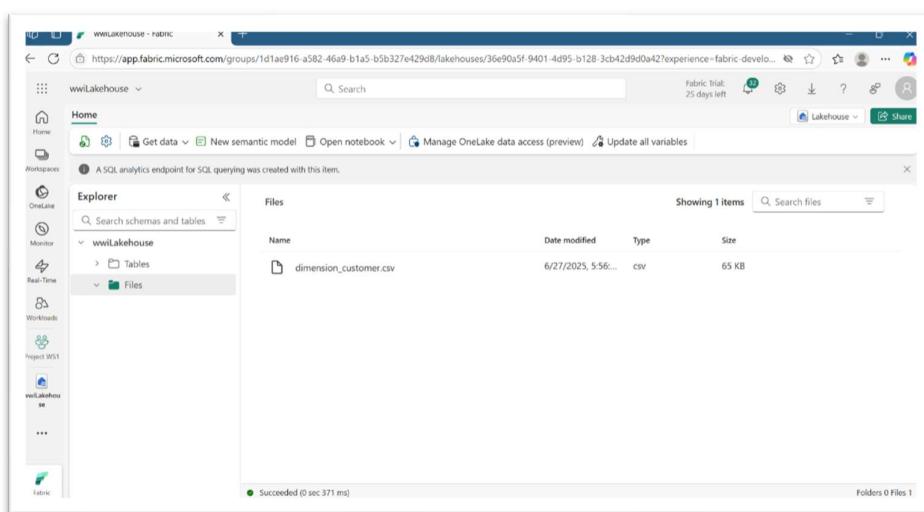
## ➤ INGEST SAMPLE DATA

- Download the *dimension\_customer.csv* file from the Fabric sample repository.
- In the Lakehouse Home panel Select the **GET DATA** and Explore pane. Click **Upload files**, then browse and select the *dimension\_customer.csv* file from the local system folder.

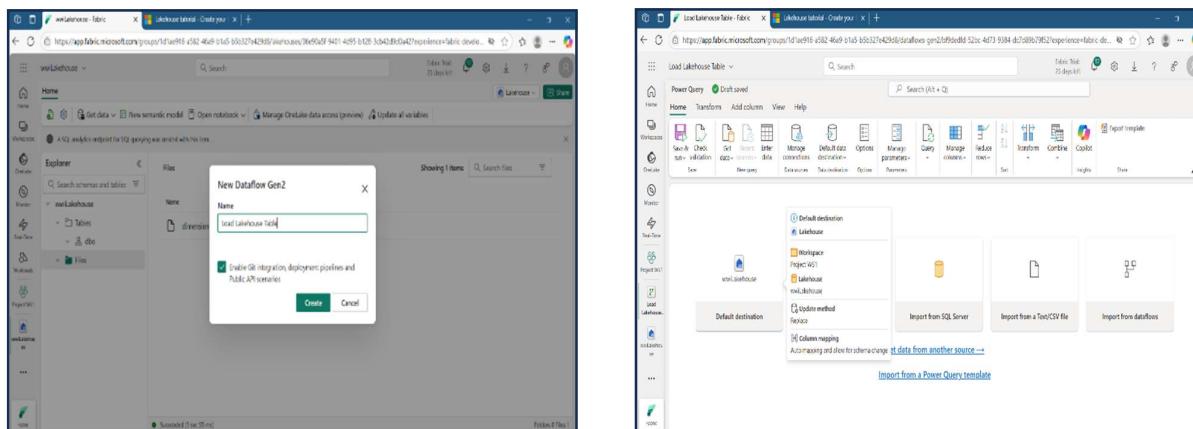




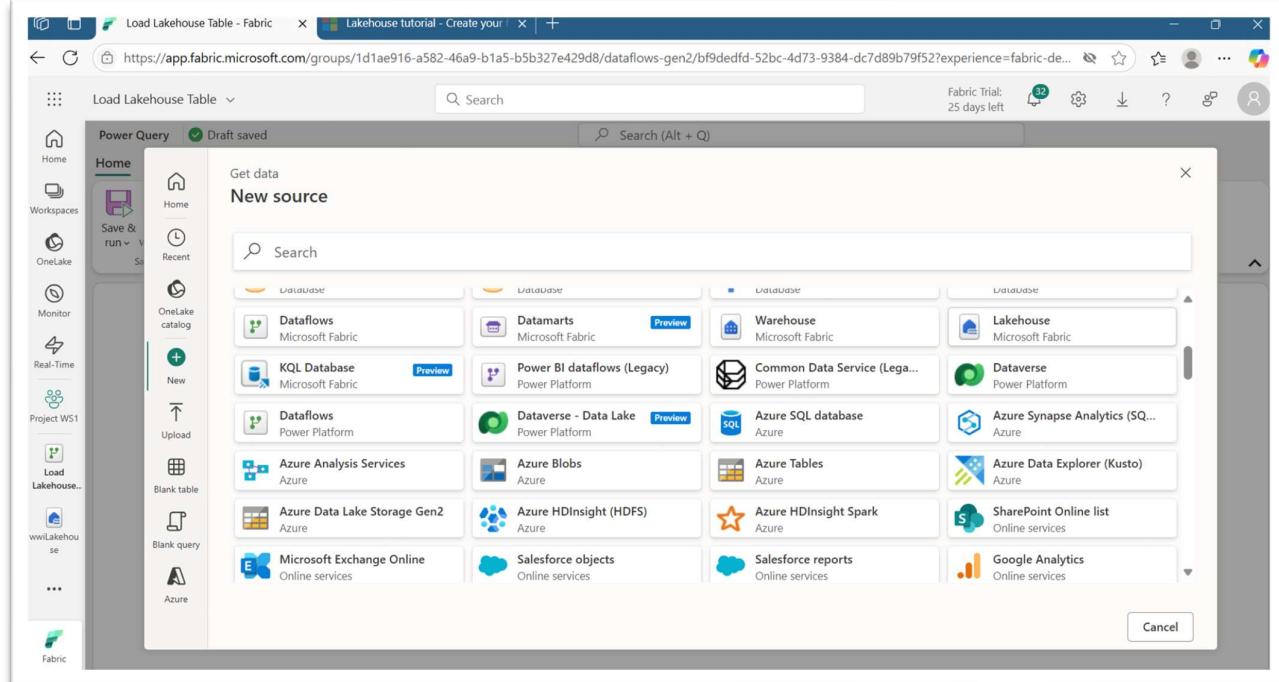
- After the file is successfully uploaded to the lakehouse, it will appear as shown below.



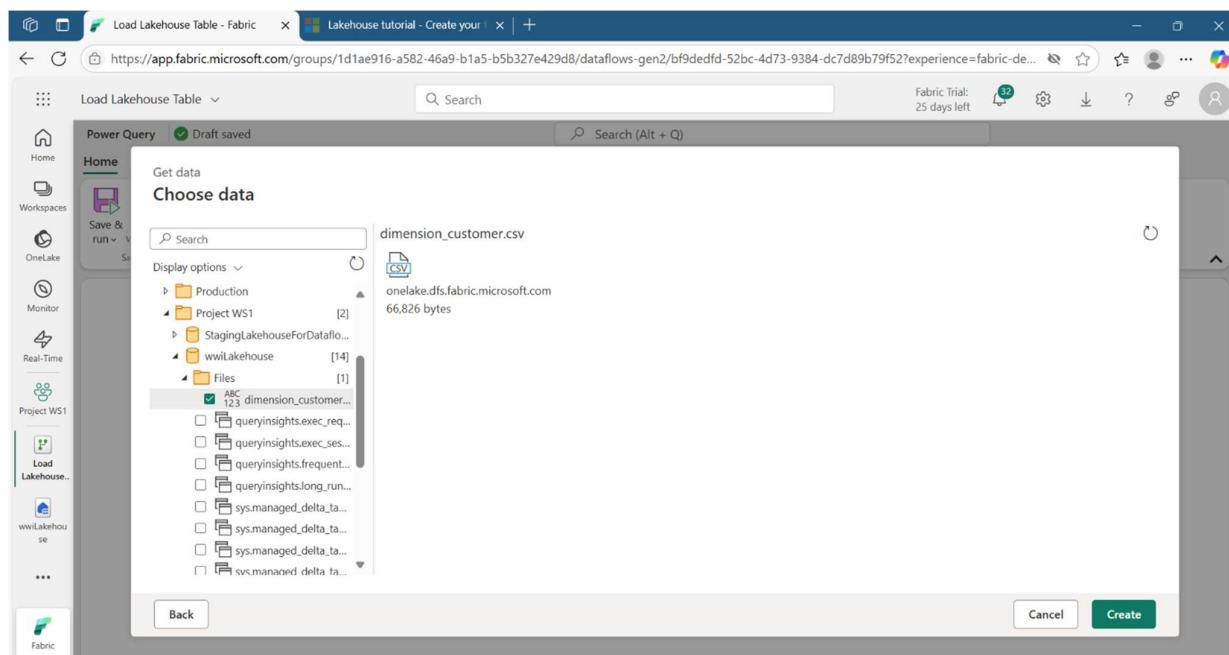
- Now Select **GET DATA** again and Explore the pane. Then select the **NEW DATAFLOW GEN2** and Create it with name **Load Lakehouse Table**.



- The Dataflow Gen2 Opens with the Default Destination (**wwiLakehouse**).
- Now Select **GET DATA** from the pane , expand it , and click on View more. Then, from the New source options, select **Lakehouse**.



- After selecting the Lakehouse and setting the connection credentials, click Next to proceed.
- After that, we need choose the data from the display options , Select **Project WS1 > wwiLakehouse > Files > dimension\_customer.csv** , then click **Create**.



- After Creating it, open the preview of the table.. On the left side of table, check the **Query Settings** and **Properties**, and ensure the name is set to **dimension\_customer**.

- Now select the Transform from the Home Pane. Choose **Use first row as headers**. This transforms the dataset into a proper table with correct headers. Then, save and run it.

The screenshot shows the Microsoft Fabric Power Query Editor interface. The 'Transform' tab is selected in the ribbon. A tooltip for the 'Use first row as headers' option is visible. The main area displays a table with 11 columns and 99+ rows, with the first row serving as the header. The 'Query settings' pane on the right indicates the source is 'Imported CSV' and the destination is 'Lakehouse'.

- Open the Lakehouse (wwiLakehouse) and refresh it. Then, select and expand the *dimension\_customer* table to view its structure.

The screenshot shows the Microsoft Fabric Lakehouse Explorer interface. The 'dimension\_customer' table is selected and expanded. The table has 11 columns and 403 rows. The first few rows of data are visible, showing columns like CustomerKey, WWICustomerID, Customer, BillToCustomer, Category, BuyingGroup, PrimaryContact, PostalCode, and Val.

	CustomerKey	WWICustomerID	Customer	BillToCustomer	Category	BuyingGroup	PrimaryContact	PostalCode	Val
1	0	0	Unknown	N/A	N/A	N/A	N/A	N/A	201
2	102	102	Tailspin Toys (Fie...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Tea Koppel	90205	
3	103	103	Tailspin Toys (Kal...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Naseem Radan	90130	
4	104	104	Tailspin Toys (Wa...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Laboni Deb	90579	
5	105	105	Tailspin Toys (To...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Sung-Hwan Hwa...	90400	
6	106	106	Tailspin Toys (Tu...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Shiva Pipali...	90662	
7	107	107	Tailspin Toys (Gle...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Karie Mercier	90782	
8	108	108	Tailspin Toys (Ber...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Bhanu Thota	90045	
9	109	109	Tailspin Toys (So...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Ae-Cha Joo	90247	
10	110	110	Tailspin Toys (No...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Dinara Saparkzy...	90792	
11	111	111	Tailspin Toys (Cri...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Adam Dvorak	90436	

- Select the **SQL analytics endpoint** from the lakehouse, then click **New SQL Query** from the Home pane.

The left screenshot displays the 'dimension\_customer' table in the 'dimension' schema of the 'wwiLakehouse' warehouse. The table contains 403 rows of data, including columns such as ID, CustomerKey, DimCustomer, DimCustomerKey, Category, HouseholdSize, MaritalStatus, and more. The right screenshot shows the 'New SQL query in notebook' dialog box, which is part of the 'Home' tab's interface. It includes a preview area showing a simple query and its results.

- Run the SQL query to validate the table and confirm successful ingestion.

The screenshot shows the Microsoft Fabric interface with the 'wwiLakehouse' workspace selected. In the 'Explorer' pane, a 'SQL query 1' is open, displaying the following SQL code:

```

1 SELECT BuyingGroup, COUNT(*) AS Total
2 FROM dimension_customer
3 GROUP BY BuyingGroup

```

The 'Results' tab shows the output of the query:

BuyingGroup	Total
Toddler Toys	23
N/A	1
Wingtip Toys	201
Kids Toys	15

The status bar at the bottom indicates 'Succeeded (4 sec 881 ms)'.

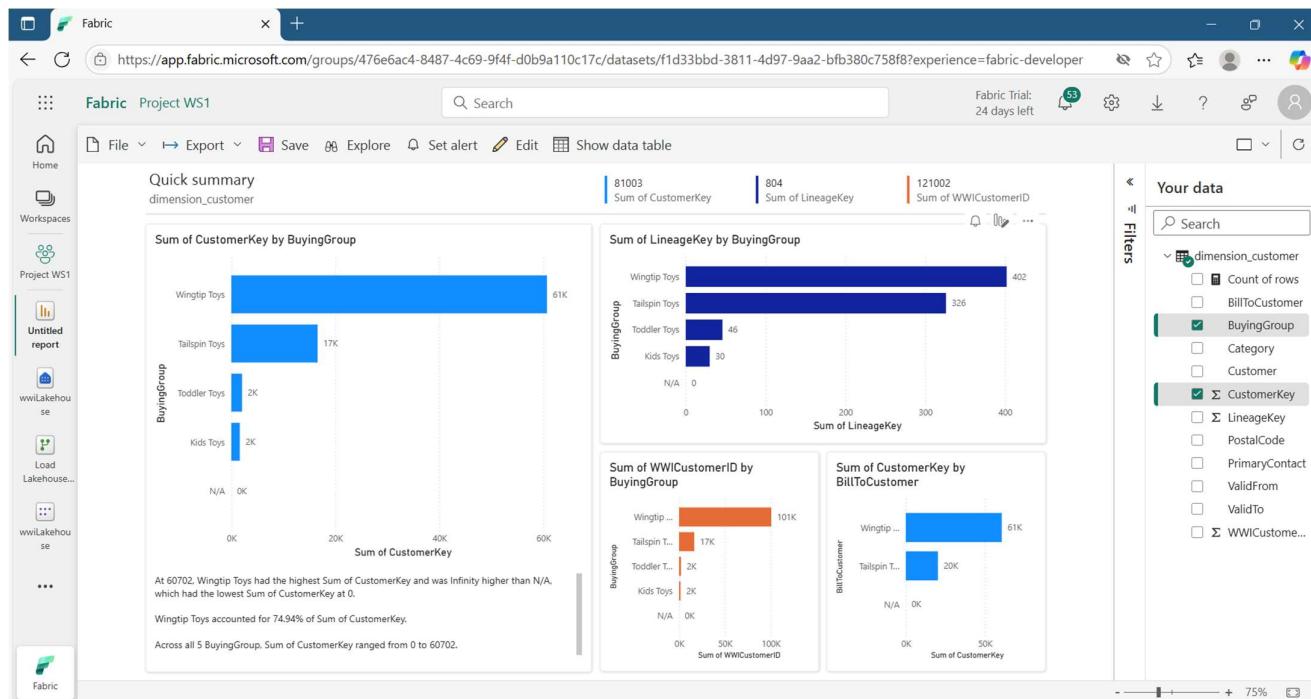
- Open the Lakehouse and switch to the **SQL analytics endpoint** view. From the **Reporting** tab, select **Manage default semantic model** and choose the tables that wants to include. In this case, select the ***dimension\_customer*** table.

The screenshot shows the Microsoft Fabric web interface. On the left, there's a sidebar with 'Home', 'Workspaces', and 'Project WS1'. Under 'Project WS1', there are several lakehouse entries, with 'wwiLakehouse' selected. The main area has tabs for 'Home', 'Reporting', and 'Help'. Below these are buttons for 'New report', 'New semantic model', 'Manage default semantic model', and 'Automatically update semantic model'. A message states: 'This warehouse has a default Power BI semantic model. To automatically add objects, go to warehouse settings. To manually add objects, use Manage default semantic model. Learn more...'. The central part of the screen shows an 'Explorer' pane with 'Warehouses' expanded, showing 'wwiLakehouse' with 'Schemas' (dbo, INFORMATION..., sys, Security) and 'Queries' (My queries). An open 'SQL query 1' shows a simple SELECT statement. To the right, a modal dialog titled 'Manage default semantic model' lists 'Tables' under 'dbo', with 'dimension\_customer' checked. Buttons for 'Confirm' and 'Cancel' are at the bottom.

- To ensure the tables in the semantic model stay in sync, switch to the **SQL analytics endpoint** view and open the Lakehouse **settings** pane. Select **Default Power BI semantic model**, then enable the **Sync the default Power BI semantic model** option.

The screenshot shows the Microsoft Fabric web interface with the 'wwiLakehouse' workspace selected. The left sidebar includes 'Home', 'Workspaces', and 'Project WS1'. The main area has tabs for 'Home', 'Reporting', and 'Help'. A message about a default Power BI semantic model is present. The 'Settings' pane is open for 'wwiLakehouse', showing sections for 'About', 'Endorsement', 'SQL endpoint', and 'Copilot'. Under 'Default Power BI Semantic Model', a toggle switch labeled 'Sync the default Power BI semantic model' is turned 'On'. A descriptive text explains: 'Add objects from the SQL analytics endpoint to the default Power BI semantic model. Also, update the model with any new objects added to the SQL analytics endpoint. You can use this model to build reports faster with SQL analytics endpoint data.' A 'Revert to default' link is also visible.

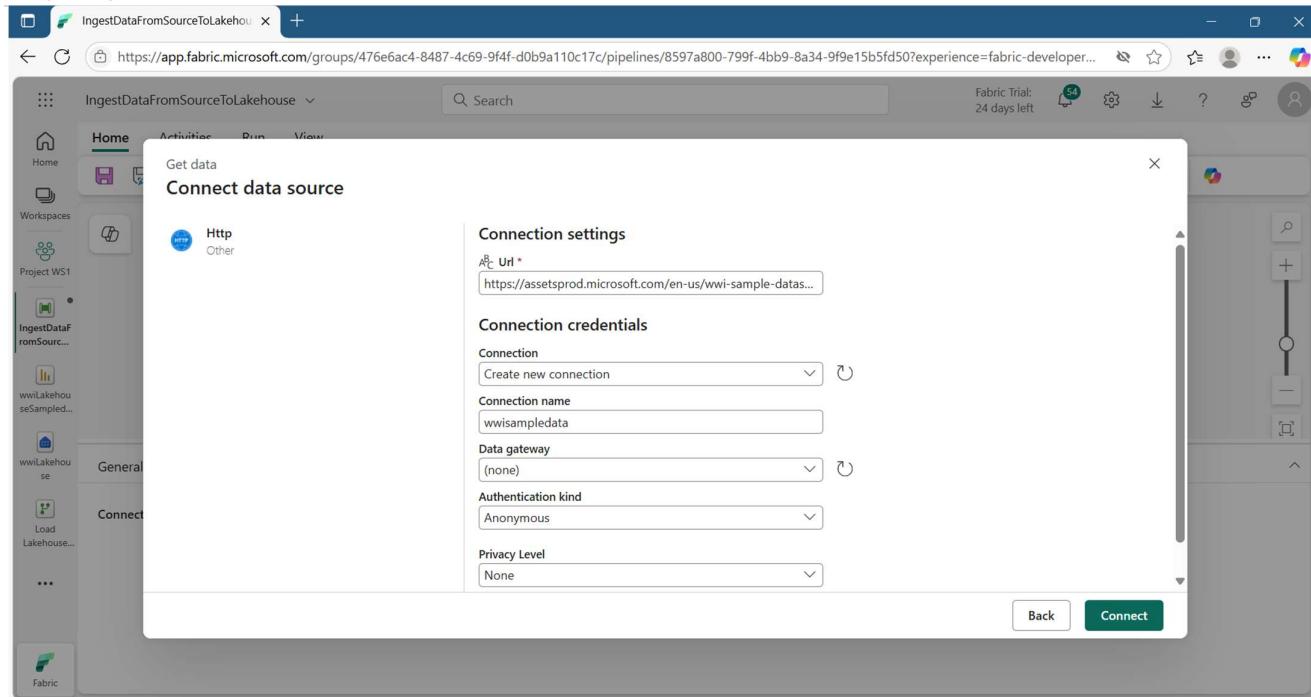
- After the table is added, Fabric creates a semantic model with the same name as the lakehouse.
- Let Power BI automatically create a report based on your data. Under **Explore this data**, select **Auto-create a report**.
- Because the table is a dimension and contains no measures, Power BI creates a row count measure. It then uses this to generate various charts, as shown in the figure below. **Save the Report**.



## INGEST DATA INTO THE LAKEHOUSE:

- Open the workspace (Project WS1), select the **DATA PIPELINE** from the **New item**.
- Specify the name as **IngestDataFromSourceToLakehouse** and select create. A new data factory pipeline is created and opened.

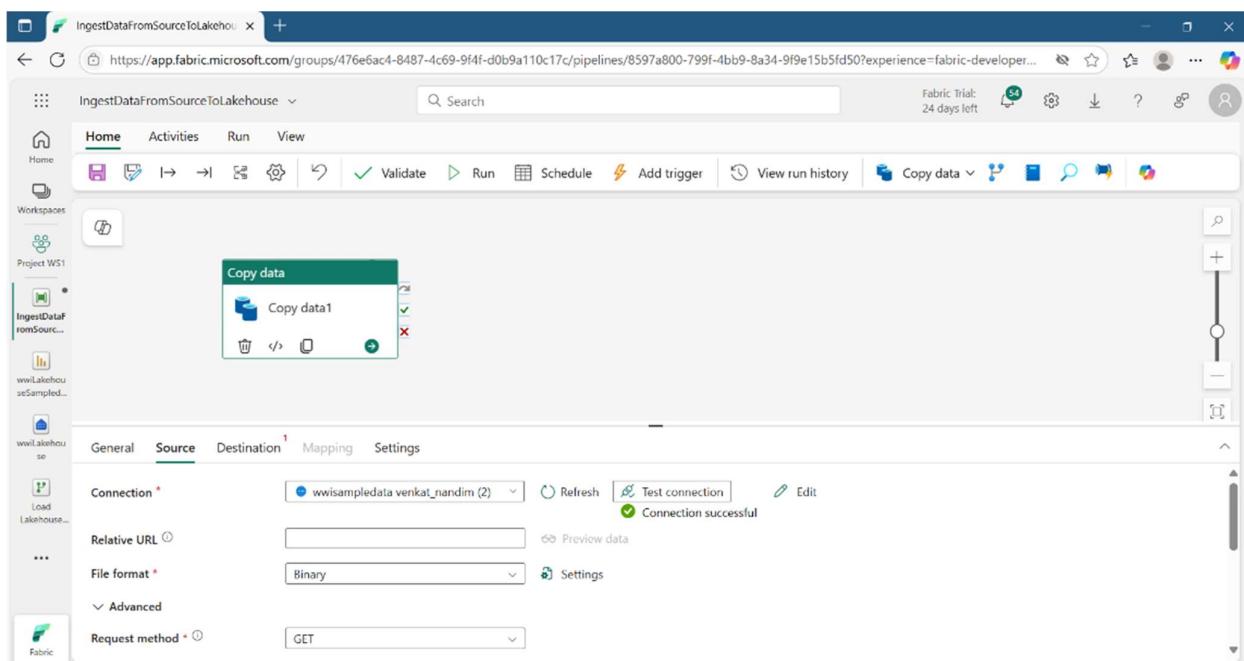
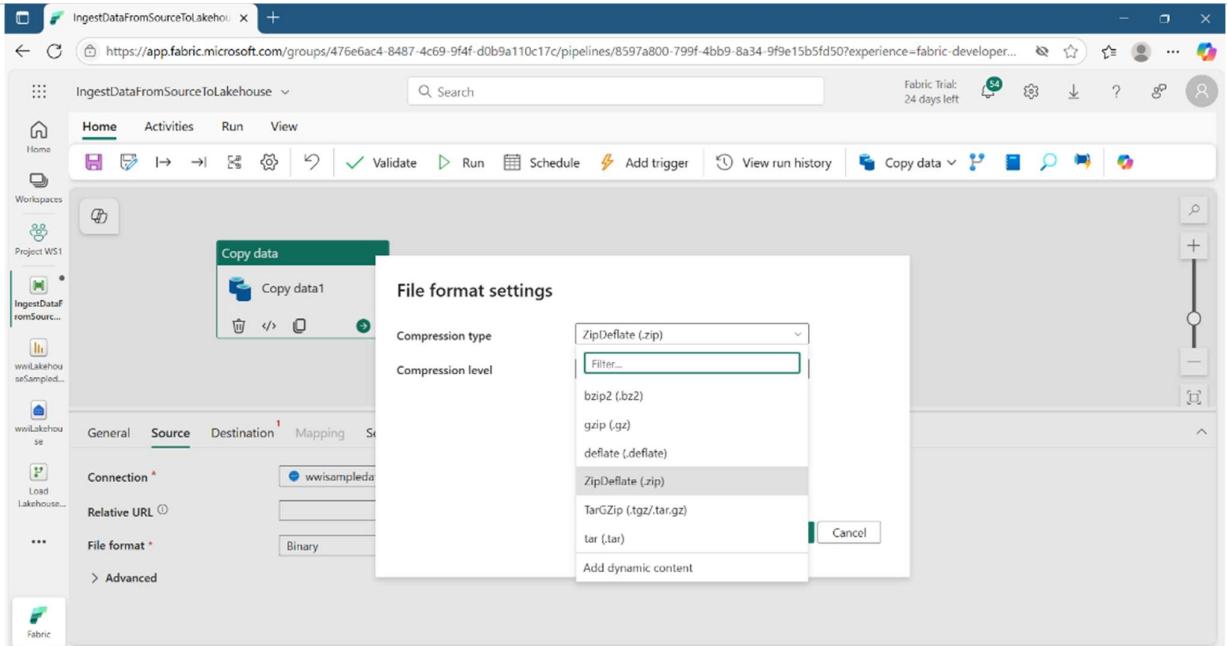
- Next, set up an HTTP connection to import the sample World Wide Importers data into the Lakehouse. From the **Copy data** list , select **Add to canvas**. Then open **Source and Destination**, choose **Source**, click **View more**, search for **Http** and select it.



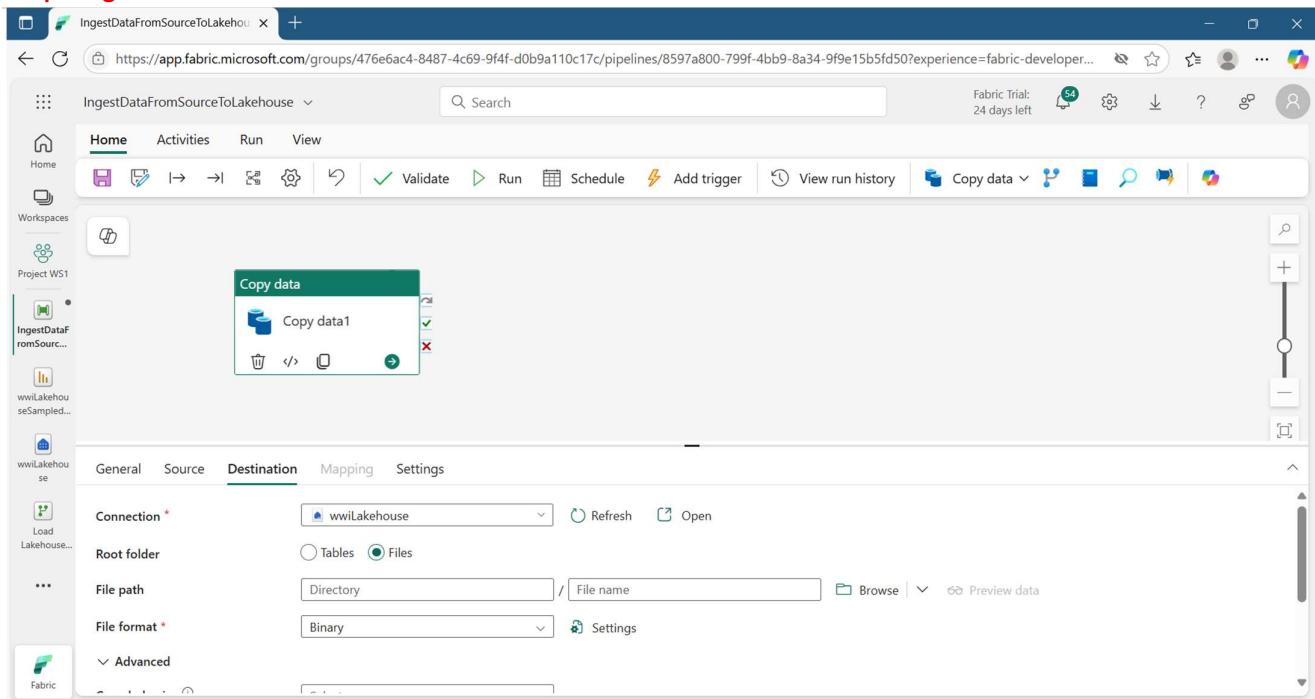
- In the **Connect to Data Source** window, enter the following details.

Property	Value
URL	<code>https://assetsprod.microsoft.com/en-us/wwi-sample-dataset.zip</code>
Connection	Create a new connection
Connection name	wwisampleddata
Data gateway	None
Authentication kind	Anonymous

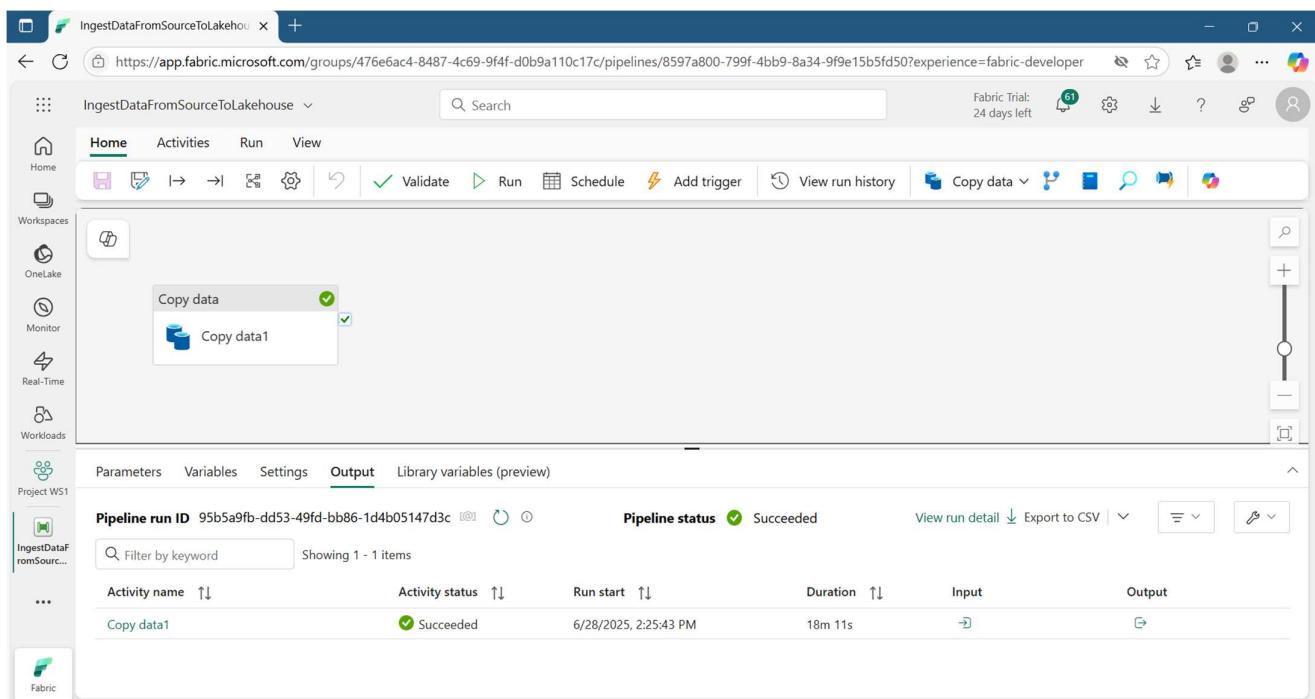
- In the next step, enable **Binary copy** and choose **ZipDeflate (.zip)** as the **Compression type**, since the source is a ZIP file. Keep the other fields at their default values, then click **Next**. Finally, click **Test Connection** to ensure it is successful.



- In the **Connect to data destination** window, specify **Files** as the root folder and click Next. This will write the data to the Files section of the Lakehouse.



- Next, click **Save and Run**. After approximately 20 minutes, the output will be displayed.



- After the successful execution of the pipeline, go to your Lakehouse (**wwiLakehouse**) and open the Explorer to view the imported data.

- Verify that the folder **WideWorldImportersDW** is present on the Explorer view and contains data for all tables.

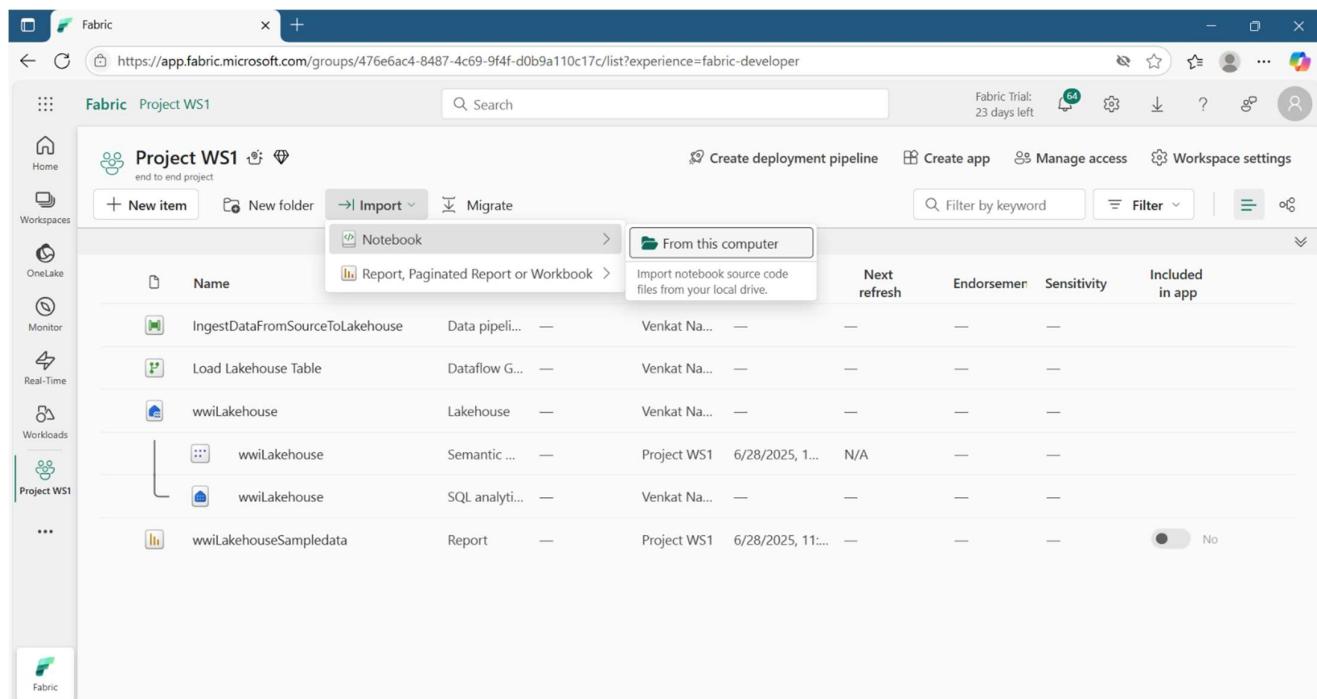
The screenshot shows the Azure Fabric Lakehouse Explorer interface. On the left, the 'Explorer' sidebar lists 'wwiLakehouse' with its 'Tables' and 'Files' sections. Under 'Files', a folder with a GUID name ('b305806e-6366-482f-a866-0489f54aa3db') is expanded, revealing sub-folders 'full', 'incremental', 'parquet', and 'tables'. Each of these sub-folders contains CSV and Parquet files for dimensions like 'dimension\_city', 'dimension\_customer', 'dimension\_date', 'dimension\_employee', and 'dimension\_stock\_item', as well as fact tables like 'fact\_sale' and 'fact\_sale\_1y\_full'. The main pane displays the contents of the 'WideWorldImporters...' folder, showing one item: 'WideWorldImporters...' (Type: Folder, Date modified: 6/28/20...).

- The data is created under the **Files** section of the Lakehouse Explorer. A new folder with a **GUID** is generated, which contains all the required data. Rename this folder to **wwi-raw-data**.

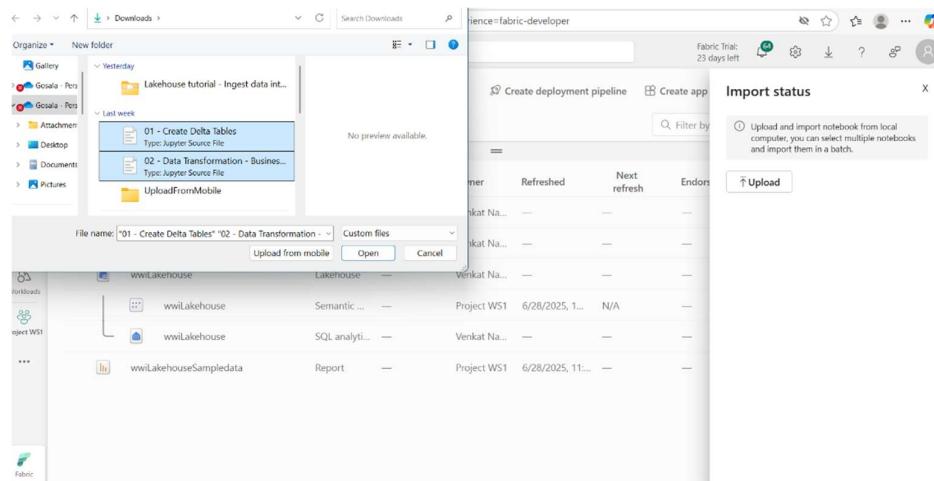
The screenshot shows the same Azure Fabric Lakehouse Explorer interface as before, but with a 'Rename folder' dialog box overlaid. The folder 'WideWorldImporters...' is selected in the list. The dialog box has a 'Folder name' input field containing 'wwi-raw-data'. Below the input field are 'Rename' and 'Cancel' buttons. The 'Rename' button is highlighted with a green background and white text.

## PREPARE AND TRANSFORM DATA IN THE LAKEHOUSE

- Download the notebooks from the Lakehouse Tutorial Source Code folder.
- From the workspace, select **Import > Notebook > From this computer**.



- Select **Import Notebook** from the **New** section at the top of the landing page.
- Select **Upload** from the **Import status** pane that opens on the right side of the screen.



- Select both notebooks and click **Open**.

- After the import is successful, go to the **Items** view of the workspace to see the newly imported notebooks. Then select **wwiLakehouse** to open it.
- Once the **wwiLakehouse** lakehouse is opened, select **Open notebook > Existing notebook** from the top menu.

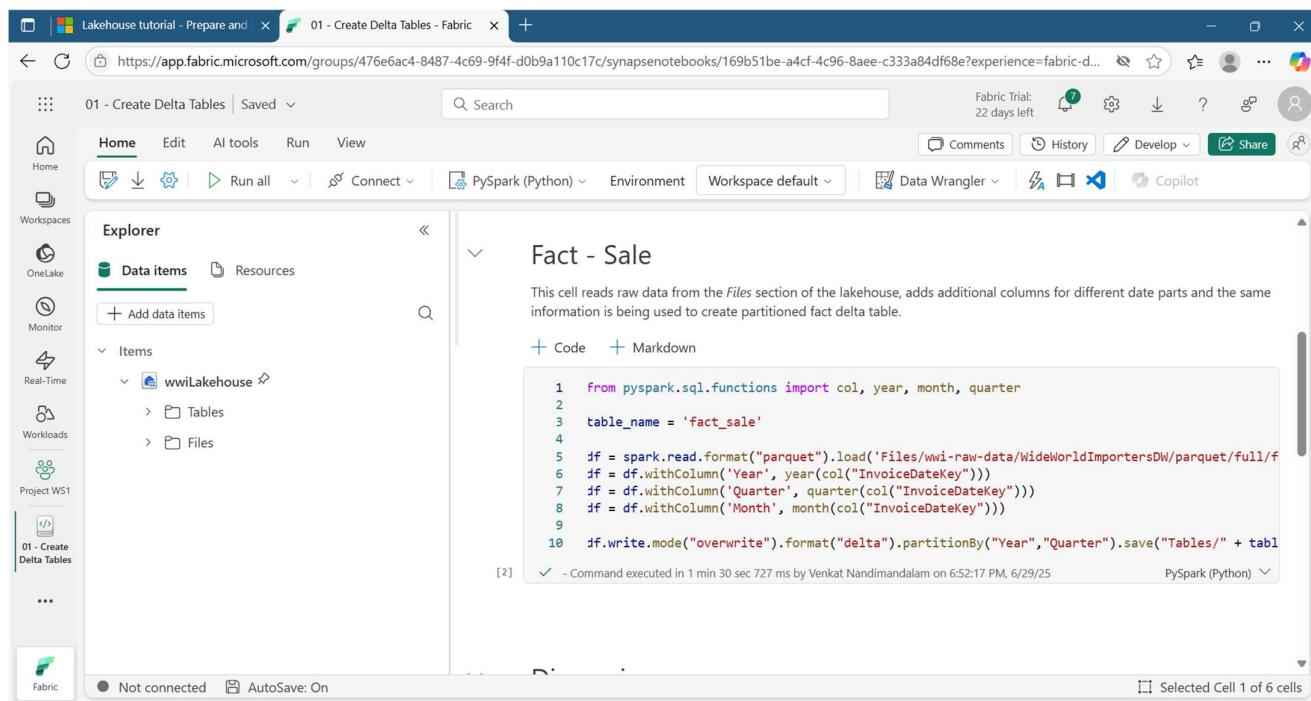
The screenshot shows the Microsoft Fabric interface for the 'wwiLakehouse' workspace. On the left, the 'Explorer' sidebar lists 'wwiLakehouse' under 'Tables' and 'Files'. In the main area, a table named 'dimension\_customer' is displayed in 'Table view' mode, showing 403 rows of data. The columns include CustomerKey, WWICusto..., Customer, BillToCusto..., Category, and Buyin. The data shows various customer entries, such as Tailspin Toys and Novelty Shop. At the bottom, a message indicates 'Succeeded (23 sec 852 ms)'.

- From the **Existing Notebook** section, select the **01- Create Delta Tables**, then click **Open**.
- In the list of existing notebooks in the Lakehouse Explorer, you'll see the notebook is already linked to the opened Lakehouse.

The screenshot shows the Microsoft Fabric Notebook interface for the '01 - Create Delta Tables' notebook. The left sidebar shows the 'Items' view with 'wwiLakehouse' selected. The main area displays two code cells. The first cell is titled 'Spark session configuration' and contains Python code for setting up a Spark session with specific configurations like VERTI-PARQUET and OPTIMIZE ON WRITE. The second cell is titled 'Fact - Sale' and contains code for reading raw data from files and adding date parts. The status bar at the bottom indicates 'Not connected' and 'AutoSave: On'.

- Before writing data as Delta Lake tables in the **Tables** section of the Lakehouse, we enable two Fabric features : **V-order** and **Optimize Write**.
- To start the notebook and execute all the cells in sequence, click **Run All** from the **Home** ribbon.
- Next, read raw data from the **Files** section of the Lakehouse, and add additional columns for various date parts as part of the transformation. Finally, use Spark API to partition the data before writing it in Delta table format, based on the newly created data columns (Year and Quarter).

### **Fact\_Sale Table.**



The screenshot shows the Microsoft Fabric Data Wrangler interface. The left sidebar displays 'Workspaces' with 'OneLake' selected. The main area has a title 'Fact - Sale'. Below it, a note states: 'This cell reads raw data from the Files section of the lakehouse, adds additional columns for different date parts and the same information is being used to create partitioned fact delta table.' A code editor window contains the following Python code:

```

1  from pyspark.sql.functions import col, year, month, quarter
2
3  table_name = 'fact_sale'
4
5  df = spark.read.format("parquet").load('Files/wwi-raw-data/WideWorldImportersDW/parquet/full/f')
6  df = df.withColumn('Year', year(col("InvoiceDateKey")))
7  df = df.withColumn('Quarter', quarter(col("InvoiceDateKey")))
8  df = df.withColumn('Month', month(col("InvoiceDateKey")))
9
10 df.write.mode("overwrite").format("delta").partitionBy("Year", "Quarter").save("Tables/" + tabl

```

The status bar at the bottom indicates: 'Not connected' and 'AutoSave: On'.

- After the **fact table loaded**, you can proceed to load data for the rest of the dimensions. The following cell creates a function that reads raw data from the **Files** section of the Lakehouse for each table names passed as a parameter.

The screenshot shows the Microsoft Fabric Data Wrangler interface. On the left, the 'Explorer' sidebar lists 'Data items' and 'Resources'. Under 'Items', it shows a folder 'wwiLakehouse' containing a 'Tables' folder with several tables: dimension\_city, dimension\_customer, dimension\_date, dimension\_employee, fact\_sale, and sale\_by\_date\_employee. A 'Date' folder is also present. The main panel is titled 'Dimensions' and contains a code editor with the following Python script:

```

1  from pyspark.sql.types import *
2
3  def loadFullDataFromSource(table_name):
4      df = spark.read.format("parquet").load('Files/wwi-raw-data/WideWorldImportersDW/parquet/fu')
5      df = df.select([c for c in df.columns if c != 'Photo'])
6      df.write.mode("overwrite").format("delta").save("Tables/" + table_name)
7
8  full_tables = [
9      'dimension_city',
10     'dimension_customer',
11     'dimension_date',
12     'dimension_employee',
13     'dimension_stock_item'
14 ]
15
16 for table in full_tables:
17     loadFullDataFromSource(table)
    
```

The status bar at the bottom indicates 'Not connected' and 'AutoSave: On'.

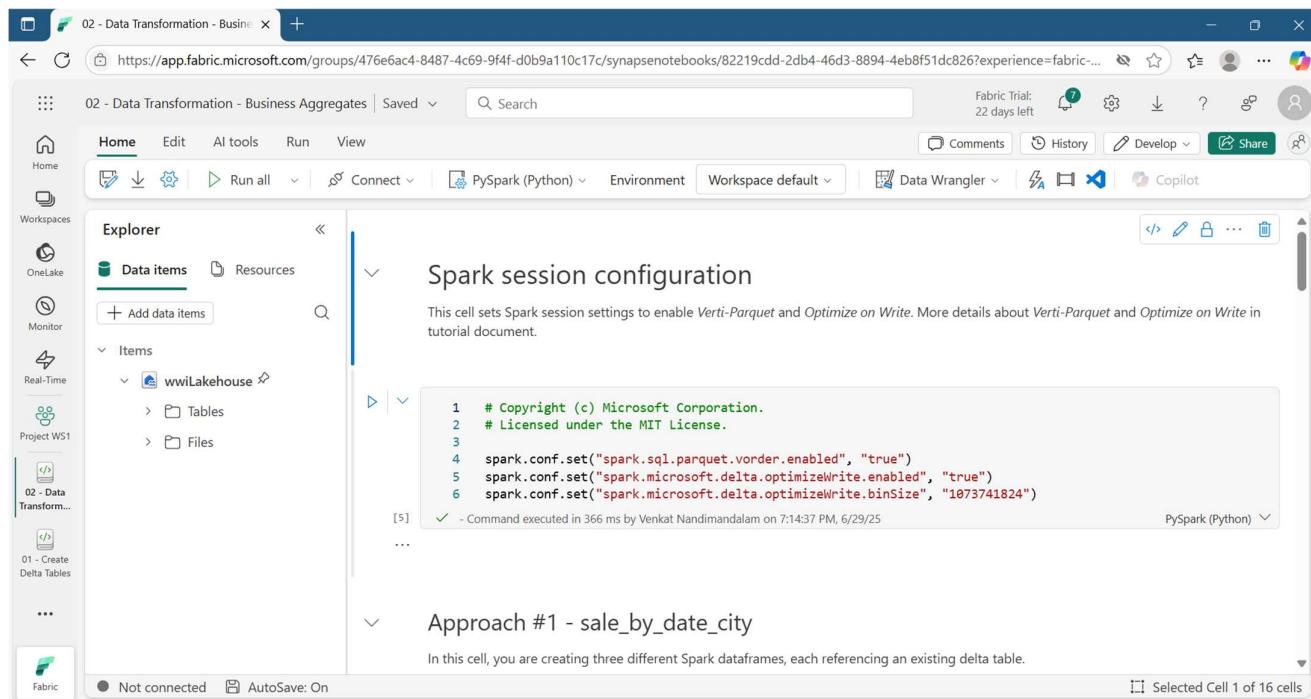
- To validate the created tables, right-click on **wwiLakehouse** and select **Refresh**. The tables appear in the above figure.
- Open the Lakehouse again, then launch the second notebook. In the Lakehouse view, select **Open notebook > Existing notebook** from the **Home** menu.
- Select the **02 – Data Transformation – Business** notebook to open it.

The screenshot shows the Microsoft Fabric OneLake catalog. The left sidebar lists 'OneLake catalog', 'Home', 'Workspaces', 'OneLake', 'Monitor', 'Real-Time', 'Project WS1', and '01 - Create Delta Tables'. The main area displays a table titled 'Open existing notebook' with the following data:

Name	Owner	Refreshed	Location	Endorsement	Sensitivity
01 - Create Delta Tables	Venkat Nandimanda...	—	Project WS1	—	—
Sales Orders Data Explore	Venkat Nandimanda...	—	venkat-dp700	—	—
Notebook 1	Venkat Nandimanda...	—	venkat-dp700	—	—
Load Sales	Venkat Nandimanda...	—	venkat-dp700	—	—
Query Products	Venkat Nandimanda...	—	dp700 practice	—	—
<b>02 - Data Transformation - B...</b>	<b>Venkat Nandimanda...</b>	<b>—</b>	<b>Project WS1</b>	<b>—</b>	<b>—</b>

The row for '02 - Data Transformation - B...' is highlighted. At the bottom right are 'Open' and 'Cancel' buttons.

- There are two different approaches to transforming and generating business aggregates:
  - Approach #1 – Use **PySpark** to join and aggregates data for generating business – level summaries. This approach is ideal if you have a background in programming, such as Python or PySpark.
  - Approach #2 – Use **Spark SQL** to join and aggregates data using SQL syntax. This method is useful for those more comfortable with SQL-based development.
  - First, set the Spark session configuration before proceeding with either method.



The screenshot shows the Microsoft Fabric Data Transformation workspace. The left sidebar lists workspaces: OneLake, Monitor, Real-Time, Project WS1, 01 - Create Delta Tables, and 02 - Data Transform... The main area has tabs: Home, Edit, AI tools, Run, View, PySpark (Python), Environment, Workspace default, Data Wrangler, and Copilot. The Data Wrangler tab is selected. The Explorer panel shows a project structure under 'Items': 'wwILakehouse' containing 'Tables' and 'Files'. A code cell titled 'Spark session configuration' contains the following Python code:

```

1  # Copyright (c) Microsoft Corporation.
2  # Licensed under the MIT License.
3
4  spark.conf.set("spark.sql.parquet.vorder.enabled", "true")
5  spark.conf.set("spark.microsoft.delta.optimizeWrite.enabled", "true")
6  spark.conf.set("spark.microsoft.delta.optimizeWrite.binsize", "1073741824")

```

The code cell status bar indicates it was executed in 366 ms by Venkat Nandimandalam on 7:14:37 PM, 6/29/25. The bottom status bar shows 'Not connected' and 'AutoSave: On'.

- **Approach #1 (sale\_by\_data\_city)** – Use **PySpark** to join and aggregate data for generating business aggregates. Using the following code, three different Spark DataFrames are created, each referencing an existing Delta table. A few columns are renamed, and the results are written as a Delta table to the **Tables** section of the Lakehouse to persist the data.
  - In this step, you create three Spark DataFrames, each referencing an existing Delta table.

The screenshot shows a Microsoft Fabric Data Transformation workspace titled "02 - Data Transformation - Business Aggregates". The left sidebar lists workspaces like OneLake, Monitor, Real-Time, and Project WS1. The main area has tabs for Home, Edit, AI tools, Run, View, and PySpark (Python). A search bar and a "Fabric Trial: 22 days left" message are at the top. The Explorer sidebar shows "Data items" and "Resources", with "Items" expanded to show "wwiLakehouse" containing "Tables" and "Files". The central pane displays code for creating three Spark dataframes:

```

1 df = spark.read.format("delta").load("Tables/fact_sale")
2 df.write.format("delta").saveAsTable("dbo факт_продажи")
3
[9] ✓ - Command executed in 1 min 13 sec 270 ms by Venkat Nandimandalam on 7:22:03 PM, 6/29/25
PySpark (Python) ▾
...
1 # Load from existing Delta folder
2 df = spark.read.format("delta").load("Tables/dimension_city")
3
4 # Save under dbo schema
5 df.write.format("delta").saveAsTable("dbo.dimension_city")
6
[13] ✓ - Command executed in 7 sec 965 ms by Venkat Nandimandalam on 7:27:25 PM, 6/29/25
PySpark (Python) ▾

```

At the bottom, status bars indicate "Not connected" and "AutoSave: On".

- > Add the following code to the same cell to join the tables using the dataframes created earlier. **group by** generate aggregation, rename a few of the columns, and finally write the results as a Delta table in the Tables section of the Lakehouse.

The screenshot shows the same Microsoft Fabric Data Transformation workspace. The code cell now includes joining the created dataframes:

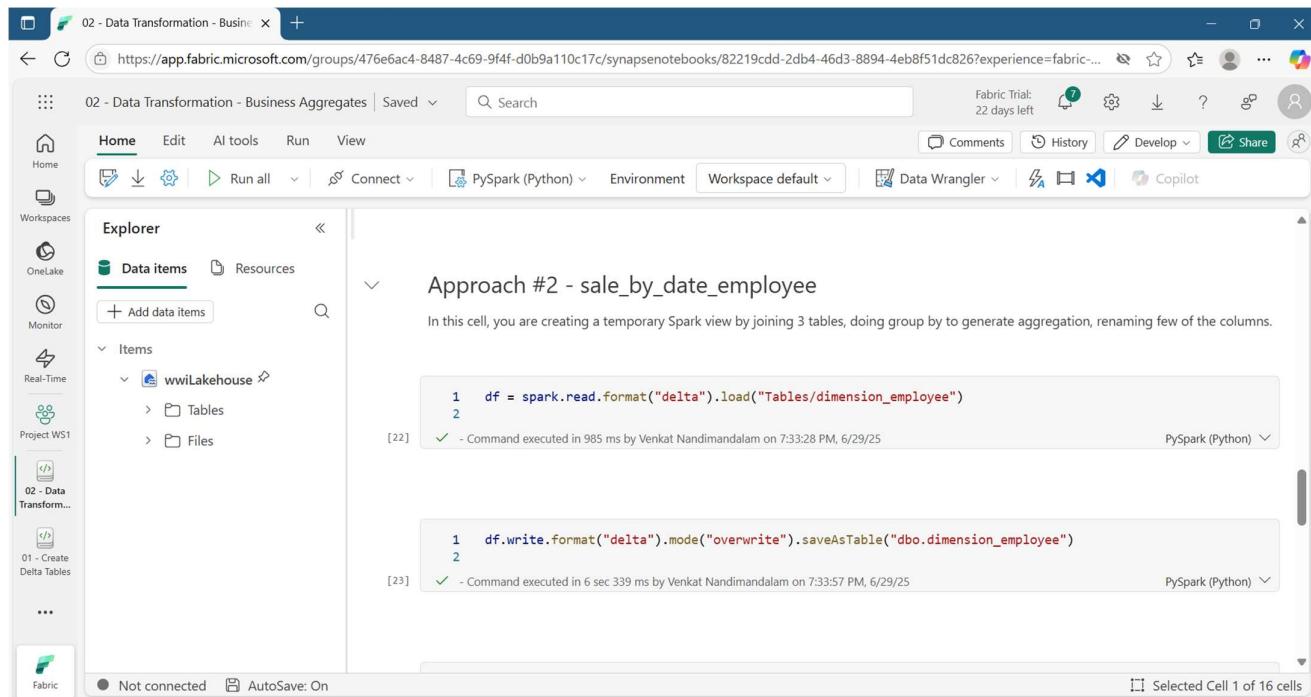
```

1 df_fact_sale = spark.read.table("dbo факт_продажи")
2 df_dimension_date = spark.read.table("dbo dimension_date")
3 df_dimension_city = spark.read.table("dbo dimension_city")
4
[15] ✓ - Command executed in 3 sec 146 ms by Venkat Nandimandalam on 7:28:51 PM, 6/29/25
PySpark (Python) ▾
...
1 sale_by_date_city = df_fact_sale.alias("sale") \
2 .join(df_dimension_date.alias("date"), df_fact_sale.InvoiceDateKey == df_dimension_date.Date, "inner") \
3 .join(df_dimension_city.alias("city"), df_fact_sale.CityKey == df_dimension_city.CityKey, "inner") \
4 .select("date.Date", "date.CalendarMonthLabel", "date.Day", "date.ShortMonth", "date.CalendarYear", "city.Ci
5 .groupBy("date.Date", "date.CalendarMonthLabel", "date.Day", "date.ShortMonth", "date.CalendarYear", "city.C
6 .sum("sale.TotalExcludingTax", "sale.TaxAmount", "sale.TotalIncludingTax", "sale.Profit") \
7 .withColumnRenamed("sum(TotalExcludingTax)", "SumOfTotalExcludingTax") \
8 .withColumnRenamed("sum(TaxAmount)", "SumOfTaxAmount") \
9 .withColumnRenamed("sum(TotalIncludingTax)", "SumOfTotalIncludingTax") \
10 .withColumnRenamed("sum(Profit)", "SumOfProfit") \
11 .orderBy("date.Date", "city.StateProvince", "city.City")
12
13 sale_by_date_city.write.mode("overwrite").format("delta").option("overwriteSchema", "true").save("Tables/agg
[16] ✓ - Command executed in 20 sec 855 ms by Venkat Nandimandalam on 7:29:28 PM, 6/29/25
PySpark (Python) ▾

```

At the bottom, status bars indicate "Not connected" and "AutoSave: On".

➤ **Approach #2 ( sale\_by\_date\_employee ) – Use Spark SQL to join and aggregate data for generating business aggregates.** Using the following code, I create a temporary Spark view by joining three tables, perform a **group by** to generate aggregation, and rename a few of the columns. Finally, I read from the temporary Spark view and write the results as a Delta table in the **Tables** section of the Lakehouse to persist the data.

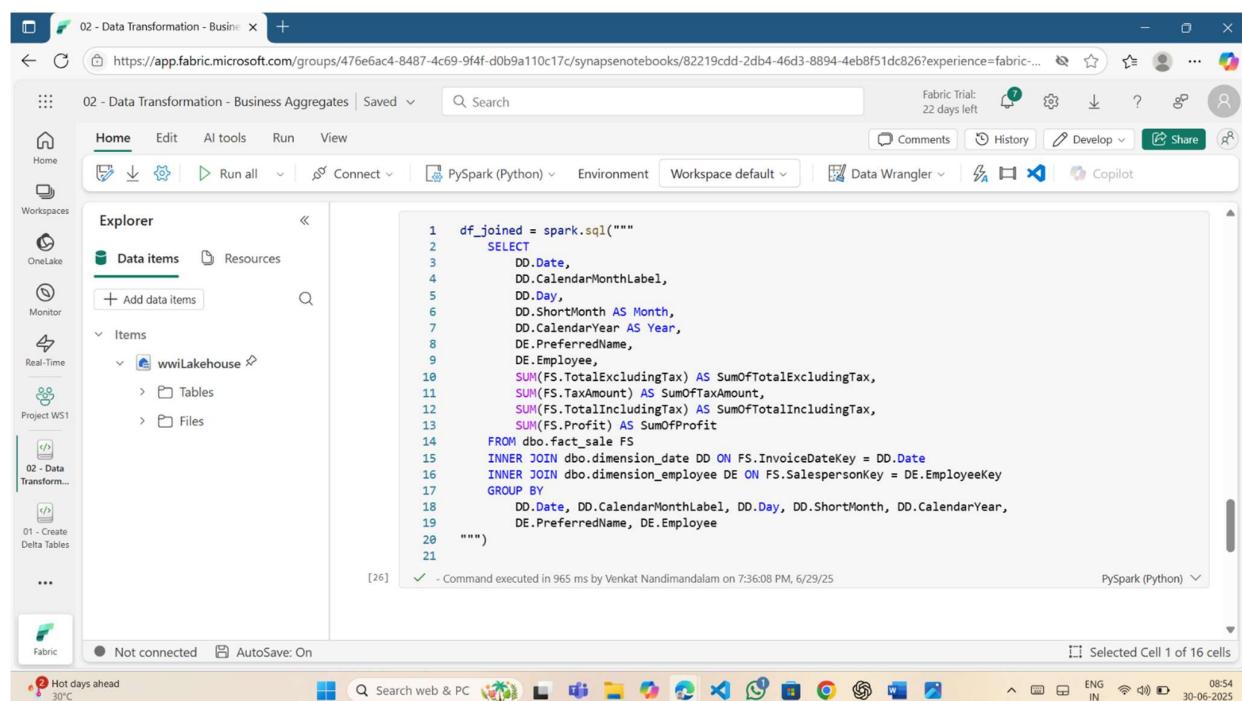


```

1 df = spark.read.format("delta").load("Tables/dimension_employee")
2
[22] ✓ - Command executed in 985 ms by Venkat Nandimandalam on 7:33:28 PM, 6/29/25
PySpark (Python) ▾

1 df.write.format("delta").mode("overwrite").saveAsTable("dbo.dimension_employee")
2
[23] ✓ - Command executed in 6 sec 339 ms by Venkat Nandimandalam on 7:33:57 PM, 6/29/25
PySpark (Python) ▾

```

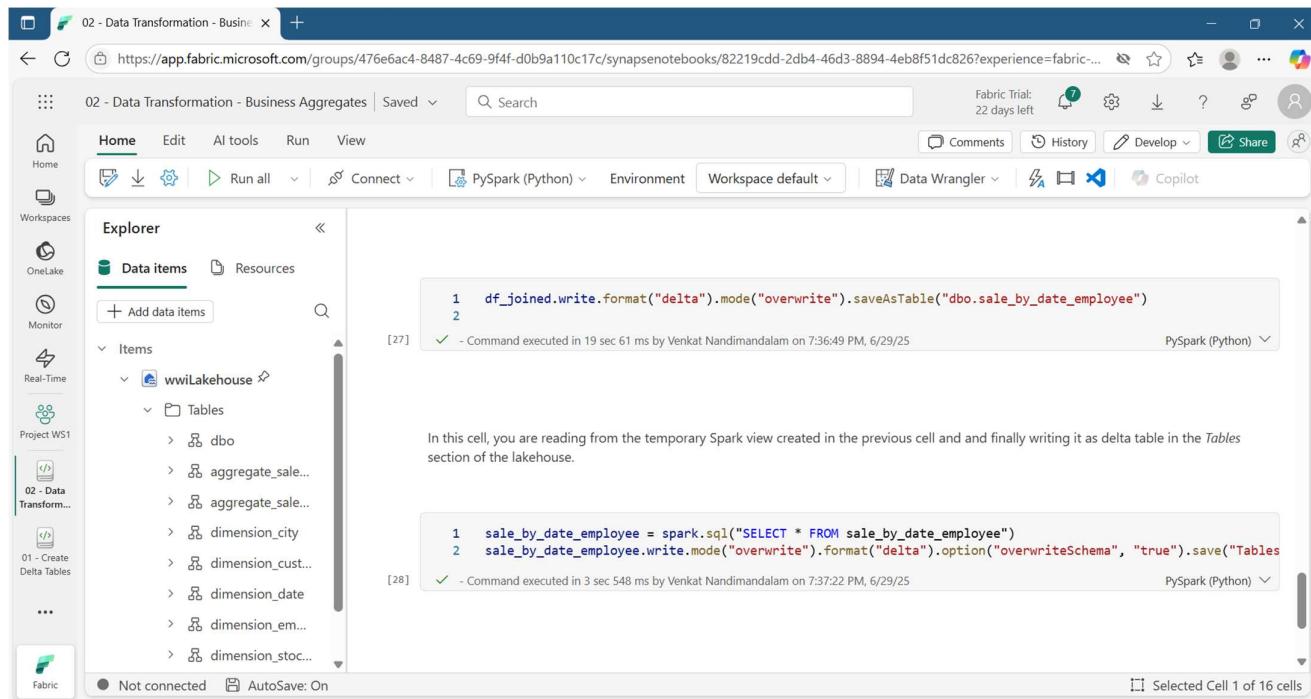


```

1 df_joined = spark.sql"""
2   SELECT
3     DD.Date,
4     DD.CalendarMonthLabel,
5     DD.Day,
6     DD.ShortMonth AS Month,
7     DD.CalendarYear AS Year,
8     DE.PreferredName,
9     DE.Employee,
10    SUM(FS.TotalExcludingTax) AS SumOfTotalExcludingTax,
11    SUM(FS.TaxAmount) AS SumOfTaxAmount,
12    SUM(FS.TotalIncludingTax) AS SumOfTotalIncludingTax,
13    SUM(FS.Profit) AS SumOfProfit
14   FROM dbo.fact_sale FS
15   INNER JOIN dbo.dimension_date DD ON FS.InvoiceDateKey = DD.Date
16   INNER JOIN dbo.dimension_employee DE ON FS.SalespersonKey = DE.EmployeeKey
17   GROUP BY
18     DD.Date, DD.CalendarMonthLabel, DD.Day, DD.ShortMonth, DD.CalendarYear,
19     DE.PreferredName, DE.Employee
20   """
21
[26] ✓ - Command executed in 965 ms by Venkat Nandimandalam on 7:36:08 PM, 6/29/25
PySpark (Python) ▾

```

- In this step, the data is read from the temporary Spark view created earlier and written as a Delta table to the **Tables** section of the lakehouse.



The screenshot shows a Microsoft Fabric Data Transformation workspace titled "02 - Data Transformation - Business Aggregates". The left sidebar lists workspaces: OneLake, Monitor, Real-Time, Project WS1, 02 - Data Transformation, and 01 - Create Delta Tables. The main area has tabs for Home, Edit, AI tools, Run, View, PySpark (Python), Environment, Workspace default, Data Wrangler, and Copilot. The Explorer pane shows a tree structure under "wwILakehouse" with "Tables" expanded, showing "dbo", "aggregate\_sale...", "dimension\_city", "dimension\_cust", "dimension\_date", "dimension\_em...", and "dimension\_stoc...". The code editor contains two cells:

```
[27] 1 df_joined.write.format("delta").mode("overwrite").saveAsTable("dbo.sale_by_date_employee")
      2
      ✓ - Command executed in 19 sec 61 ms by Venkat Nandimandalam on 7:36:49 PM, 6/29/25
      PySpark (Python) ▾
```

In this cell, you are reading from the temporary Spark view created in the previous cell and finally writing it as delta table in the *Tables* section of the lakehouse.

```
[28] 1 sale_by_date_employee = spark.sql("SELECT * FROM sale_by_date_employee")
      2 sale_by_date_employee.write.mode("overwrite").format("delta").option("overwriteSchema", "true").save("Tables"
      ✓ - Command executed in 3 sec 548 ms by Venkat Nandimandalam on 7:37:22 PM, 6/29/25
      PySpark (Python) ▾
```

Selected Cell 1 of 16 cells

- The both approaches – using PySpark or Spark SQL produce a similar outcome. To avoid the need to learn a new technology or compromise on performance, choose the method that best suits for background and personal preferences.
- you may notice that the data writing as **Delta lake** files. Microsoft Fabric's automatic table discovery and registration feature detects and register these files in the metastore. Therefore you don't need to explicitly run *CREATE TABLE* statements to make them available for SQL queries.

## BUILDING REPORTS IN MICROSOFT FABRIC.

### Build a report

- ❖ From the **wwILakehouse**, select **SQL Analytics Endpoint** from the Lakehouse dropdown menu at the top.
- ❖ In the SQL Analytics Endpoint pane, you should be able to see all the created tables. Click **Refresh** at the top, then select the **Model** at the bottom to open the default Power BI semantic model.

- ❖ To build meaningful reports, you need to define the relationship between different tables so that can reports and visualization based on the data coming across different tables. From the **fact\_sale** table, drag the **CityKey** field and drop it on the **CityKey** in the **dimension\_city** table .
- ❖ The **New relationship** dialog box appears.

The screenshot shows the Microsoft Fabric Model layouts interface. On the left, the Explorer sidebar lists 'Warehouses' (wwiLakehouse), 'Schemas' (dbo, INFORMATIO...), 'Tables' (dimension\_city, fact\_sale), and 'Columns'. The 'dimension\_city' table is selected. The main area displays the table schema with columns: Continent, Country, LatestRecordedPopulation, LineageKey, Location, Region, SalesTerritory, StateProvince, Subregion, and Month. To the right, the 'Properties' panel shows the table's name as 'dimension\_city', a description field, and a 'Row label' dropdown set to 'Select a row label'. A note at the top states: 'This warehouse has a default Power BI semantic model. To automatically add objects, go to warehouse settings. To manually add objects, use Manage default semantic model. Learn more'.

#### ❖ In the New relationship dialog box:

- Table 1 is populated with **fact\_sale** and 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**.

The screenshot shows the Microsoft Fabric Model layouts interface with the 'New relationship' dialog box open. The 'From table' dropdown is set to 'fact\_sale' and the 'To table' dropdown is set to 'dimension\_city'. Both dropdowns show their respective schemas. The dialog box contains the message: 'Select tables and columns that are related.' Below the dropdowns, there is a preview area with the text 'A preview of this table isn't available'. The background shows the same Explorer and Model layouts interface as the previous screenshot.

The screenshot shows the Microsoft Fabric Model Layouts interface. On the left, the Explorer sidebar lists 'fact\_sale' and 'dimension\_city' tables under the 'wwiLakehouse' warehouse. The 'Model layouts' tab is selected, displaying a diagram where 'fact\_sale' is connected to 'dimension\_city'. The 'Properties' pane on the right shows the relationship settings: 'Table' is 'fact\_sale' and 'Column' is 'CityKey', with 'Cardinality' set to 'Many to one (\*:1)'. A green checkmark indicates the relationship is active.

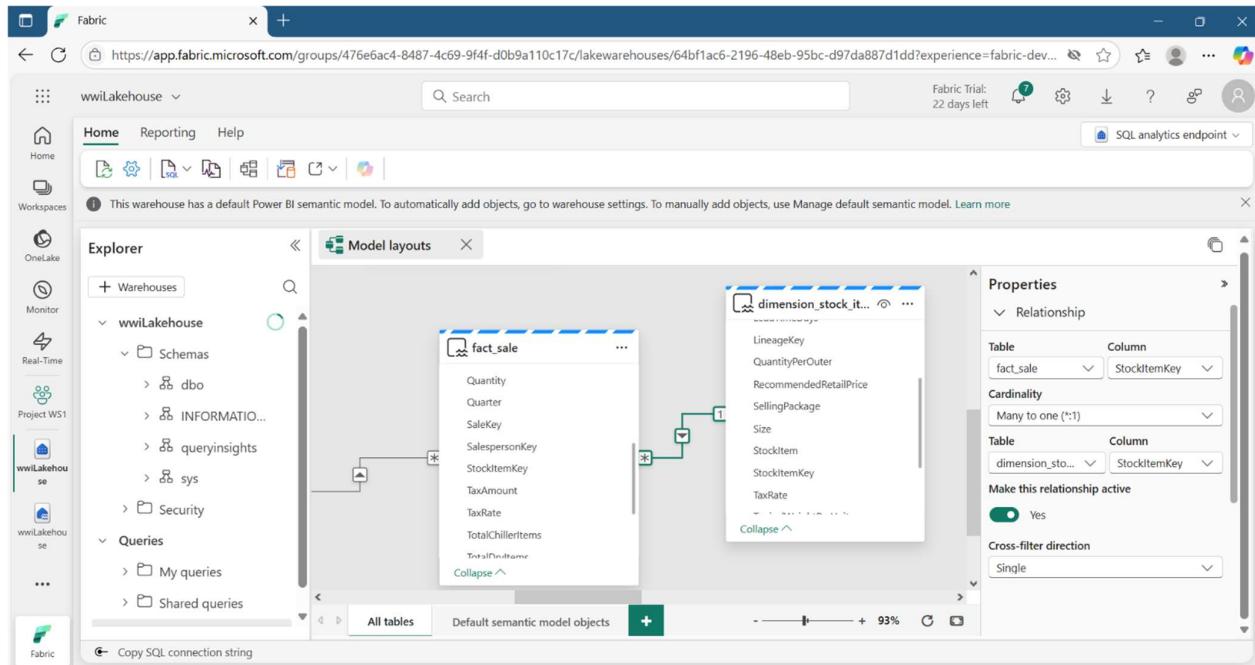
❖ The relationship between these two tables ( **fact\_sale** )-  
**(dimension\_stock\_item)**:

The screenshot shows the Microsoft Fabric Model Layouts interface. On the left, the Explorer sidebar lists 'fact\_sale' and 'dimension\_stock\_item' tables under the 'wwiLakehouse' warehouse. The 'Model layouts' tab is selected, displaying a diagram where 'fact\_sale' is connected to 'dimension\_stock\_item'. The 'Properties' pane on the right shows the relationship settings: 'Table' is 'dimension\_stock\_item' and 'Column' is 'StockItemKey', with 'Cardinality' set to 'One to many (1:\*). A green checkmark indicates the relationship is active.

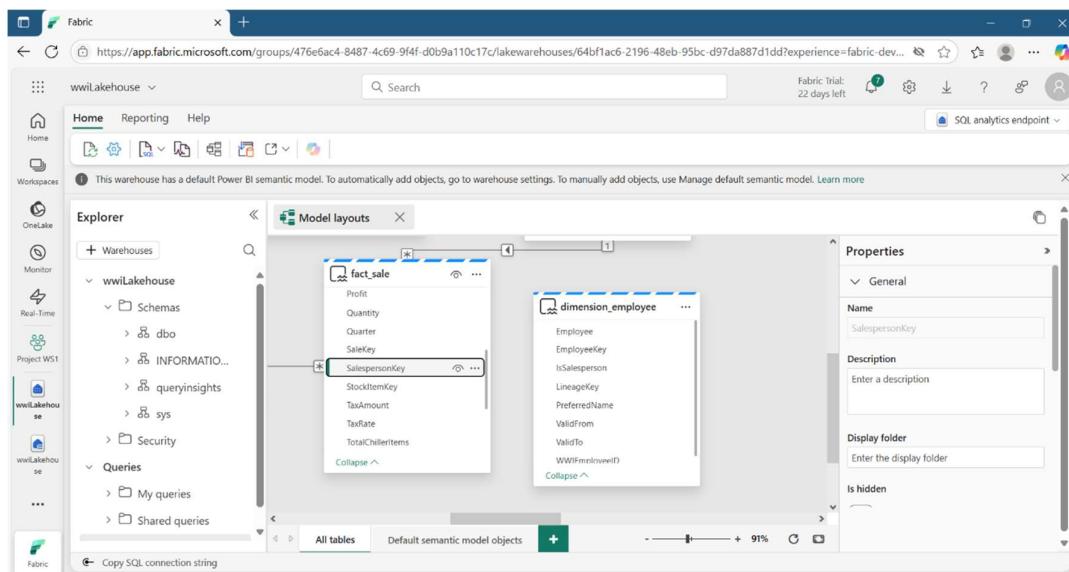
❖ In the New relationship dialog box:

- Table 1 is populated with **fact\_sale** and column of **StockItemKey**.
- Table 2 is populated with **dimension\_stock\_item** and the column of **StockItemKey**.
- 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**.



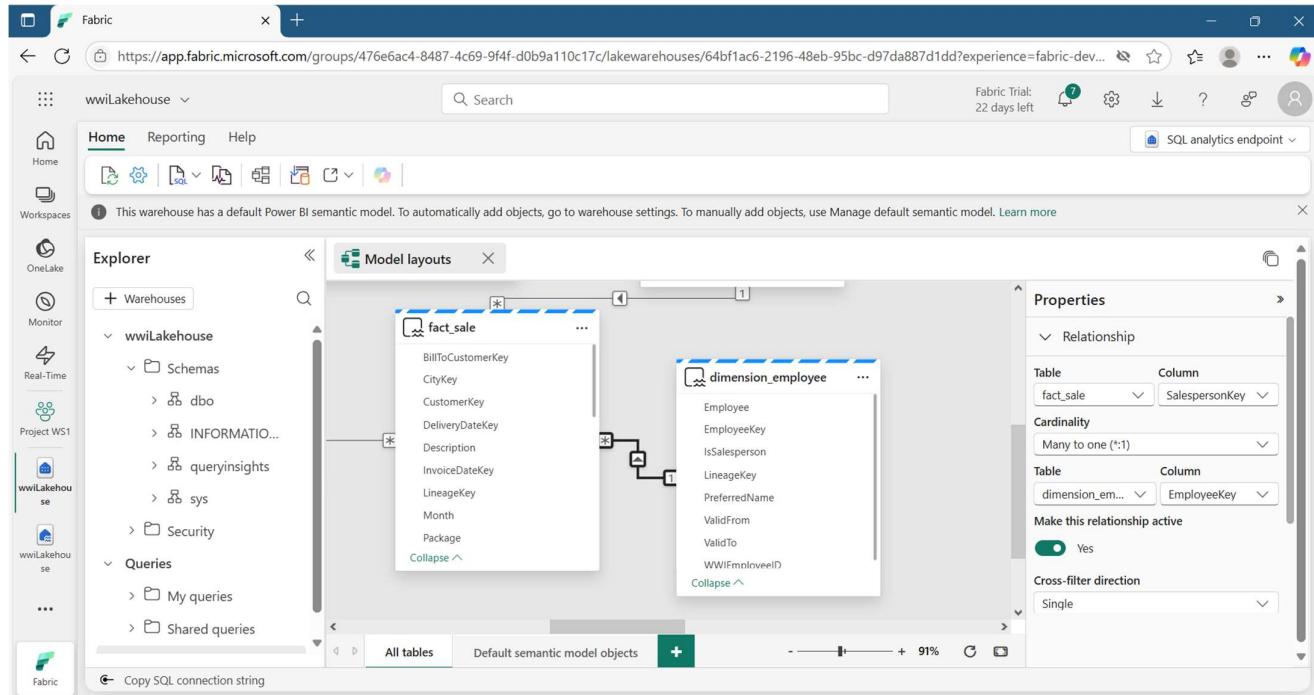
❖ The relationship between these two tables (**fact\_sale**) - (**dimension\_employee**):



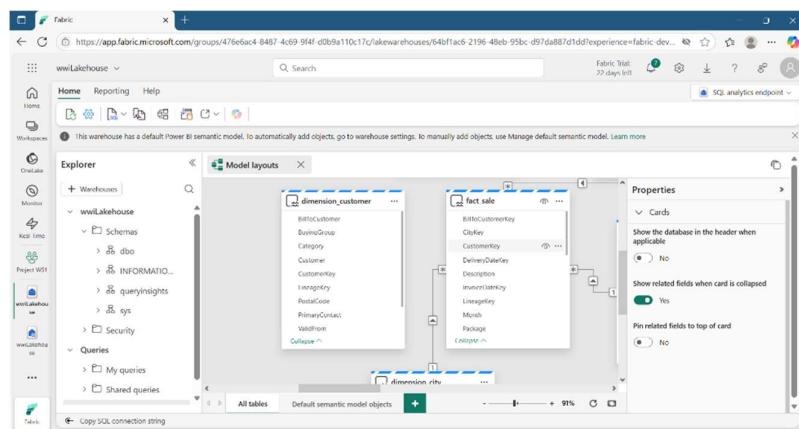
❖ In the New relationship dialog box:

- Table 1 is populated with **fact\_sale** and column of **SalespersonKey**.
- Table 2 is populated with **dimension\_employee** and the column of **EmployeeKey**.
- 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**.



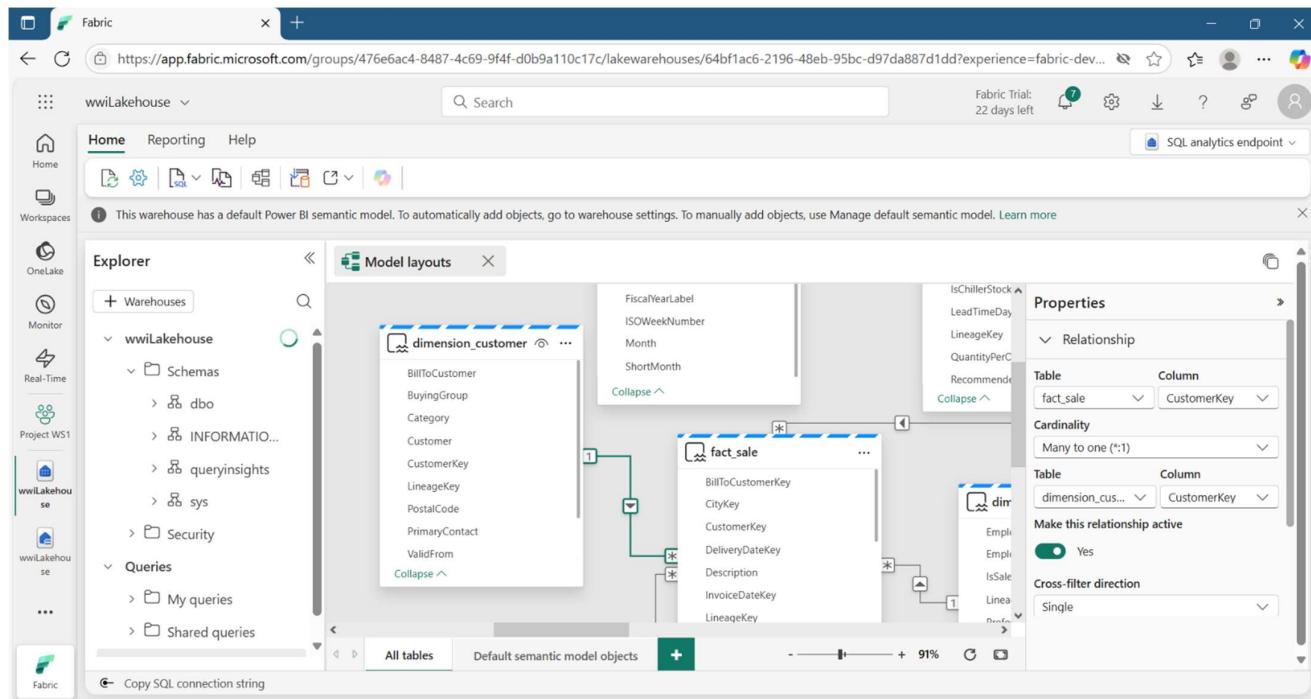
❖ The relationship between these two tables (**fact\_sale**)-  
(**dimension\_customer**):



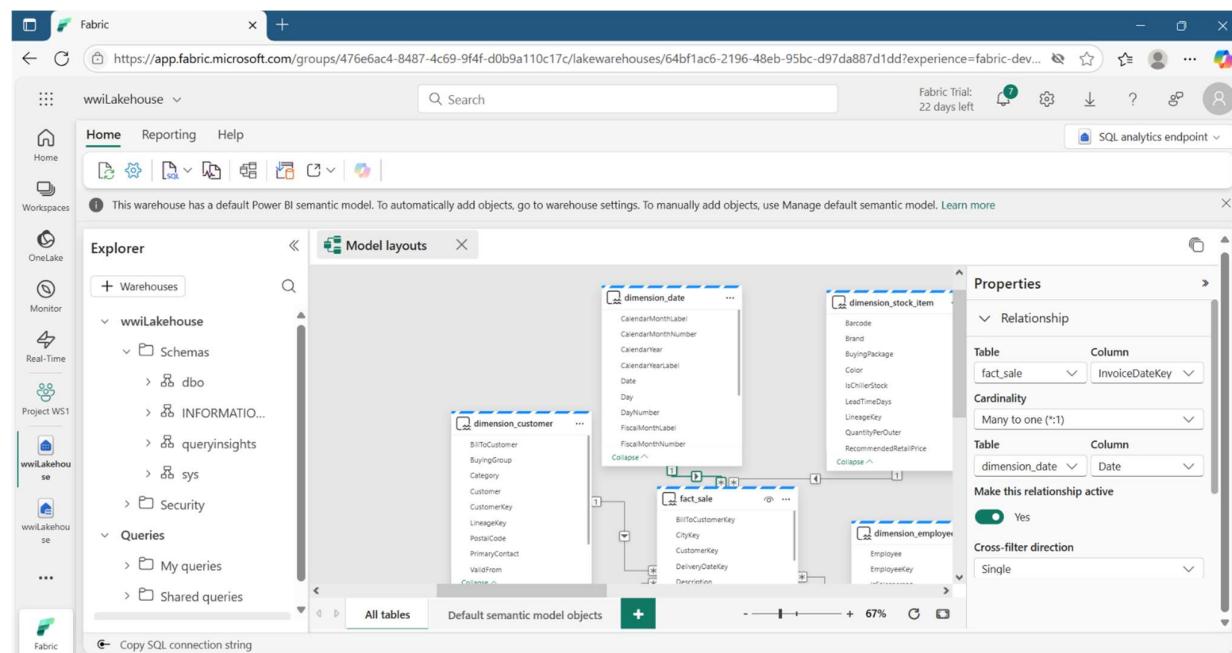
❖ In the New relationship dialog box:

- Table 1 is populated with **fact\_sale** and column of **CustomerKey**.
- Table 2 is populated with **dimension\_customer** and the column of **CustomerKey**.
- 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.

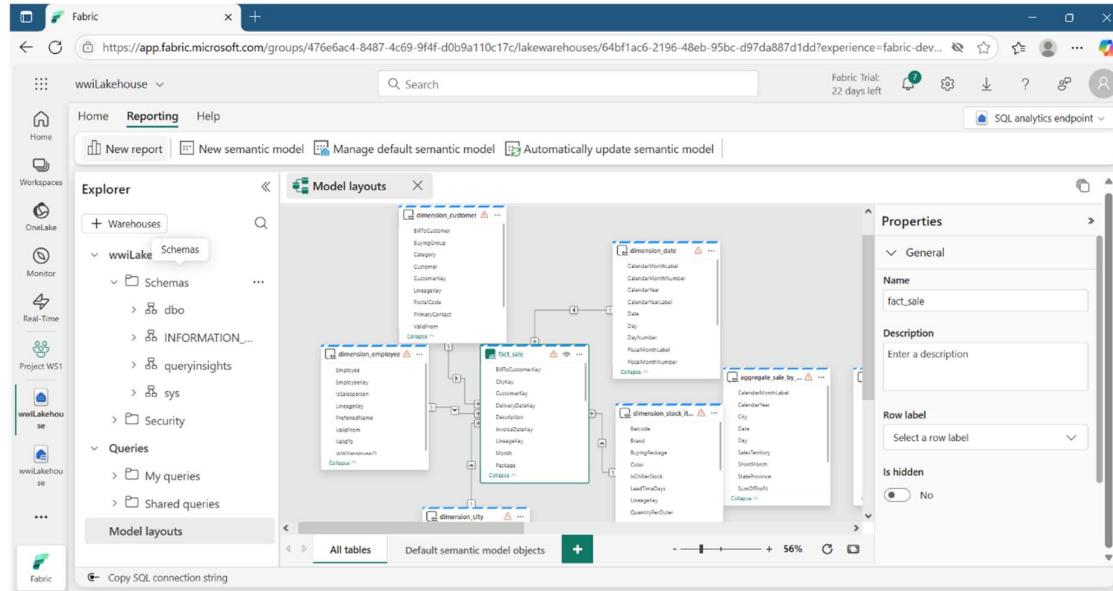


## ❖ The relationship between these two tables (fact\_sale)-(dimension\_date):

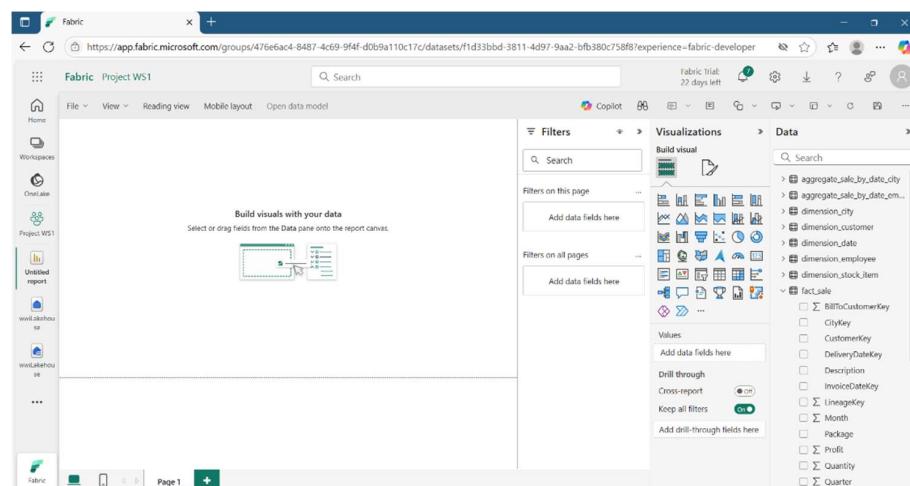


- ❖ In the New relationship dialog box:
  - Table 1 is populated with **fact\_sale** and column of **invoiceDateKey**.
  - Table 2 is populated with **dimension\_date** and the column of **Date**.

- 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**



- ❖ From the Reporting tab , select New Report to start building reports/dashboards in Power BI. On the Power BI report canvas, you can create reports to meet business requirements by dragging required columns from the **Data** pane to the canvas and using one or more of available visualizations.



- ❖ **Add a Title :**

- i) In the right side of ribbon, select Text box.
- ii) Type in **WW Importers Profit Reporting**.

- iii) Highlight the text, increase the size to 20, and move it to the upper left of the report page.

The screenshot shows the Microsoft Fabric Data Explorer interface. On the left, there's a sidebar with 'Home', 'Workspaces', 'OneLake', and 'Project WS1' sections. Under 'Project WS1', there are two reports: 'Untitled report' and 'wwiLakehouse'. The main area displays a report titled 'WWI Importers Profit Reporting'. The top navigation bar includes 'File', 'View', 'Reading view', 'Mobile layout', 'Open data model', 'Copilot', and various icons for search, filters, and data selection. On the right side, there are three main panes: 'Filters', 'Visualizations', and 'Data'. The 'Filters' pane contains sections for 'Filters on this page' and 'Filters on all pages', each with a 'Add data fields here' button. The 'Visualizations' pane shows a grid of visualization icons, with a 'Card' icon selected. The 'Data' pane lists various data models and tables, such as 'aggregate\_sale\_by\_da...', 'dimension\_city', 'dimension\_customer', etc. At the bottom, there are buttons for 'Page 1' and a '+' sign.

#### ❖ Add a Card :

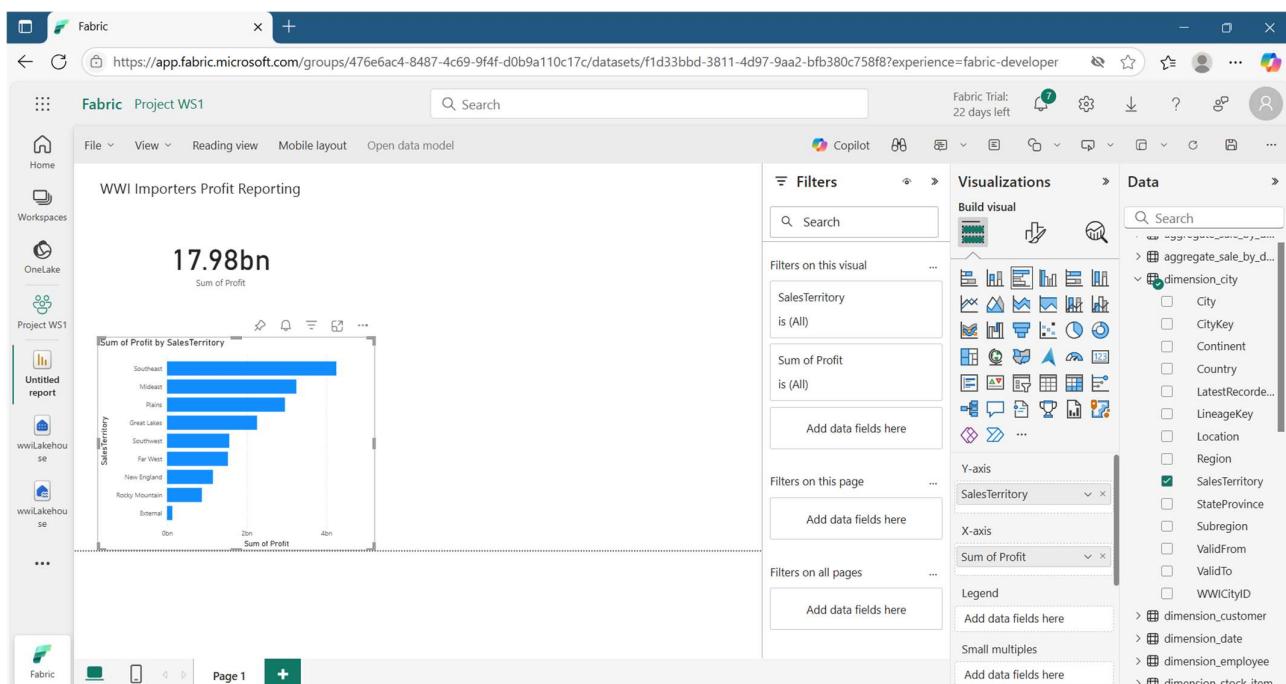
- On the Data pane, expand **fact\_sale**, and check the box next to **Profit**. This selection creates a column chart and adds the field to the Y-axis.
- With the chart selected, select the **Card** visual in the visualization pane. This selection converts the visual to a card.
- Place the card under the Title like this.

This screenshot shows the Microsoft Fabric Data Explorer interface after following the steps in the previous section. The report now features a large 'Card' visual at the top, displaying the value '17.98bn' and the subtitle 'Sum of Profit'. The 'Visualizations' pane on the right has the 'Card' icon highlighted. The 'Data' pane on the far right shows the 'fact\_sale' table expanded, with the 'Profit' field checked, indicating it is the current Y-axis field for the chart.

#### ❖ Add a Bar chart :

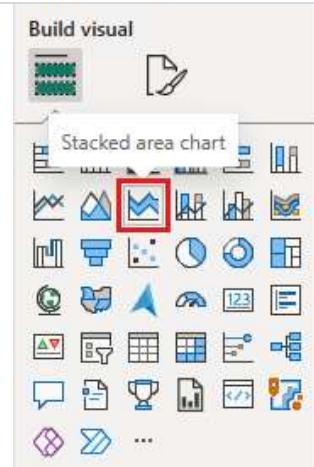
- On the **Data** pane, expand **fact\_sales** and check the box next to **Profit**. This selection creates a column chart and adds the field to the X-axis.

- ii) On the **Data** pane, expand **dimension\_city** and check the box for **SalesTerritory**. This selection adds the field to the Y-axis.
- iii) 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. Resize the Bar chart and move it under the **Title and Card**.

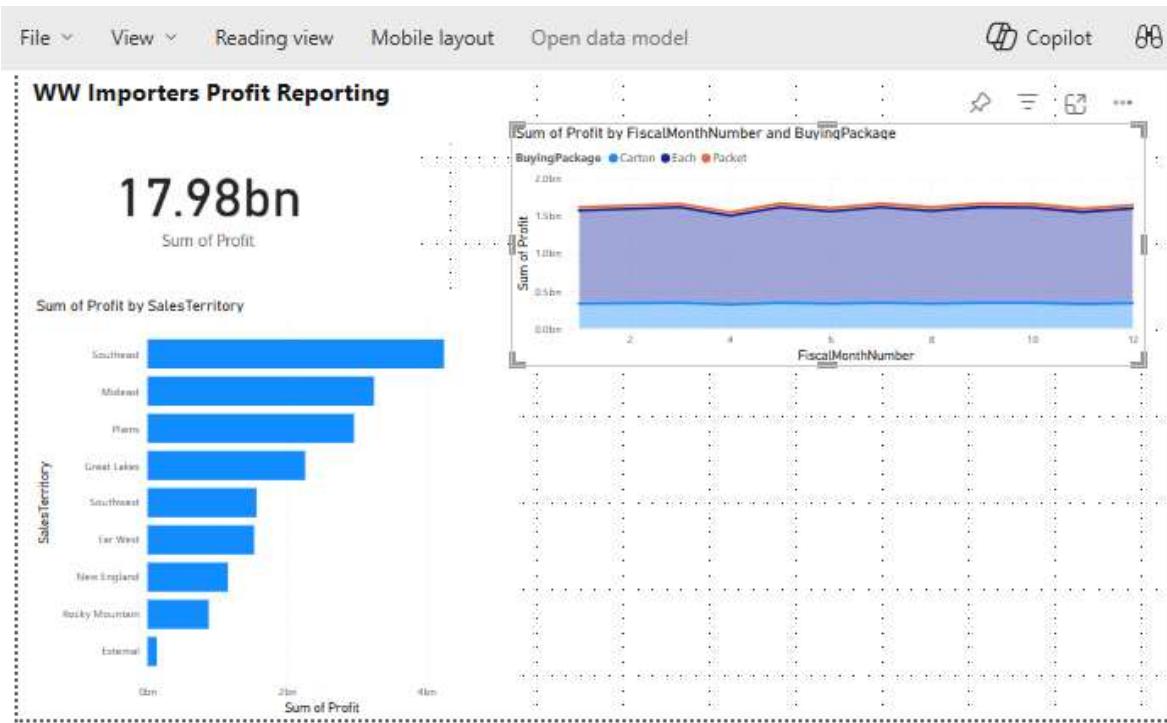


#### ❖ Build a Stacked area chart visual :

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



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



The screenshot shows the Microsoft Fabric interface with the following details:

- Left Sidebar:** Home, Workspaces, OneLake, Project WS1 (selected), Untitled report, wwlakehouse\_se, wwlakehouse.
- Top Bar:** Fabric, https://app.fabric.microsoft.com/groups/476e6ac4-8487-4c69-9f4f-d0b9a110c17c/datasets/f1d33bbd-3811-4d97-9aa2-bfb380c758f8?experience=fabric-developer, Fabric Trial: 22 days left.
- Visualizations:**
  - Sum of Profit by FiscalMonthNumber and BuyingPackage:** A stacked area chart showing profit over 12 months for Carton, Each, and Packet categories.
  - Sum of Profit by SalesTerritory:** A horizontal bar chart showing profit by Sales Territory: Southeast, Midwest, Plains, Great Lakes, Southwest, Far West, New England, Rocky Mountain, External.
  - Sum of Profit by Employee:** A line chart showing profit over 12 months for various employees.
- Data Pane:** Shows filters for BuyingPackage (Carton, Each, Packet), FiscalMonthNumber (All), and Sum of Profit (All). It also lists dimensions like dimension\_employee (Employee, EmployeeID) and dimension\_stock\_item (Barcode, Brand, Color, IsChillerStock, LeadTimeDays, LineageKey, QuantityPerOrder, RecommendedProduct).

### ❖ Build a Column chart :

- On the **Data** pane, expand **fact\_sales** and check the box next to **Profit**. This selection adds the field to the Y-axis.
- On the **Data** pane, expand **dimension\_employee** and check the box next to **Employee**. This selection adds the field to the X-axis.

The screenshot shows the Microsoft Fabric interface after building a column chart:

- Visualizations:**
  - Sum of Profit by FiscalMonthNumber and BuyingPackage:** Stacked area chart.
  - Sum of Profit by SalesTerritory:** Horizontal bar chart.
  - Sum of Profit by Employee:** A column chart showing profit for employees: Sophia, Harlan, Jack, Putter, Kayla, Woodsack, Hudson, Amy, Trelf, Taj, Shand, Anthony, Grosser, Lily, Code, Archer, Lambe, Hudson, Holloman.

The screenshot shows a Microsoft Fabric project titled "Fabric Project WS1". On the left, there's a navigation bar with Home, Workspaces, OneLake, and Project WS1 sections. The main area displays three visualizations: a Stacked column chart titled "Sum of Profit by FiscalMonthNumber and BuyingPackage", a Bar chart titled "Sum of Profit by SalesTerritory", and another Bar chart titled "Sum of Profit by Employee". The right side of the screen features a "Filters" pane, a "Visualizations" pane with a "Build visual" button, and a "Data" pane containing a search bar and a list of data fields categorized under dimension\_employee and dimension\_stock\_item.

- ❖ From the ribbon, select **File > Save**.
- ❖ Enter the name of your report as **Profit Reporting**.
- ❖ Select **Save**.

This screenshot shows the same Microsoft Fabric project as above, but with the "File" menu open. The "Save" menu is highlighted, showing options: "Save this report", "Save as", "Print", and "Download this file". The "Download this file" option is currently selected, as indicated by its blue background. The rest of the interface remains consistent with the first screenshot, showing the three visualizations and the Data pane on the right.

Fabric Project WS1

Save your report

Project WS1

Name Type Owner

- 01 - Create Delta Tables Notebook Venkat Nandimandalam
- 02 - Data Transformation - Business Aggre... Notebook Venkat Nandimandalam
- IngestDataFromSourceToLakehouse Data pipeline Venkat Nandimandalam
- Load Lakehouse Table Dataflow Gen2 (CI/CD) Venkat Nandimandalam
- StagingLakehouseForDataflows\_20250628... Lakehouse Venkat Nandimandalam
- StagingLakehouseForDataflows\_20250... Semantic model (default) Project WS1

Enter a name for your report Profit Reporting

Assign to task Select a task

Save Cancel

Profit Reporting - Fabric

Sum of Profit by FiscalMonthNumber and BuyingPackage

Sum of Profit by SalesTerritory

SalesTerritory	Sum of Profit
Southeast	~4.5bn
Midwest	~3.5bn
Plains	~3.0bn
Great Lakes	~2.5bn
Southwest	~2.0bn
Far West	~1.8bn
New England	~1.5bn
Rocky Mountain	~1.2bn
External	~0.5bn

Sum of Profit by Employee

Employee	Sum of Profit
Sophia Hinton	~2.2bn
Jack Potter	~2.1bn
Kayla Woodcock	~2.0bn
Hudson Onslow	~1.9bn
Amy Treff	~1.8bn
Taj Shand	~1.7bn
Anthony Gross	~1.6bn
Lily Code	~1.5bn
Archer Lamble	~1.4bn
Hudson Hollingsworth	~1.2bn

Report saved

This report has been saved.