

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.

Create a workspace X

Name *
Fabric Analytics
 This name is available

Description
Describe this workspace

Domain ⓘ
Assign to a domain (optional)

[Learn more about workspace settings](#) ↗

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

Semantic model storage format
 Small semantic model storage format
 Large semantic model storage format
[Learn more about semantic model storage formats](#) ↗

Capacity *
Trial-20251106T060206Z-Wvu71E8KqECSuNRTT-zu7A - Central India ▼

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 created 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.

New Lakehouse

Name *

Location *

Fabric Analytics ▼ ↗

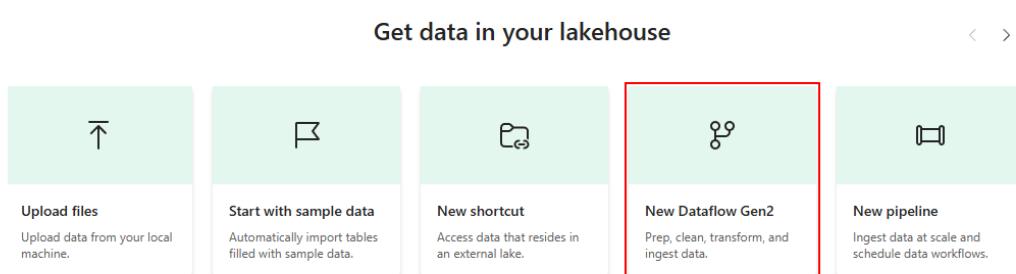
Lakehouse schemas (Public Preview) ⓘ

Create Cancel

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**.

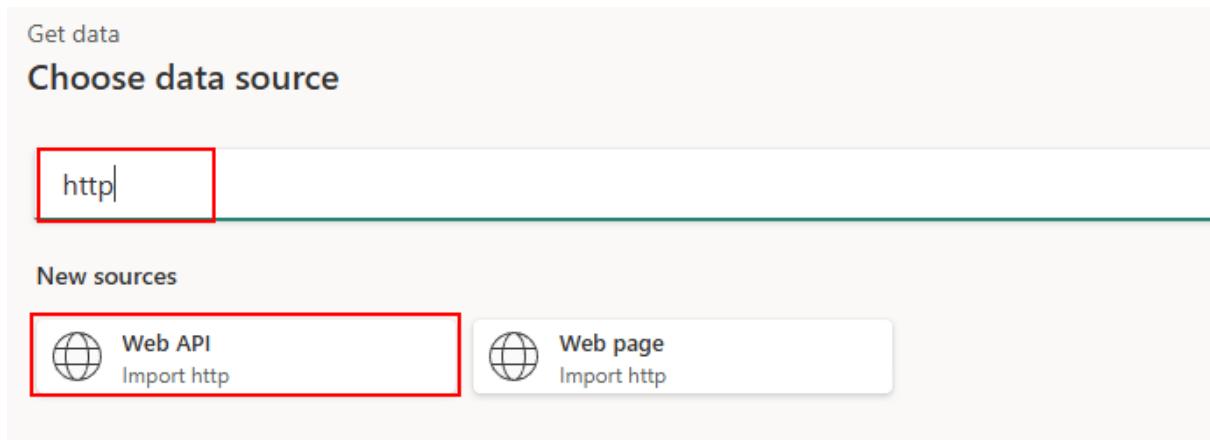
New Dataflow Gen2

Name

Enable Git integration, deployment pipelines and Public API scenarios

Create Cancel

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**.

Connection settings

URL *

Connection credentials

Connection

Connection name

Data gateway

Authentication kind

Privacy Level

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

The screenshot shows the Microsoft Power Query Editor interface. On the left, there's a sidebar with icons for Home, Workspaces, Copilot, OneLake catalog, Monitor, Real-Time, Workloads, and My workspace. The main area shows a list of queries under 'Queries [1]'. The first query is 'dimension_customer'. The 'Default destination' pane is open, with 'Lakehouse' selected. The 'Query settings' pane on the right shows the table name 'dimension_customer'.

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**.

The screenshot shows the 'Choose destination target' pane in the Power BI Data Flow editor. It has three radio button options: 'New table' (selected), 'Existing table', and 'New File (Preview)'. A note says 'A new table will be created in lh_wwi'. Below is a 'Table name' input field with 'dimension_customer' typed in. The 'Display options' section shows a tree view with 'Lakehouse' expanded, showing '[1]' for 'lh_wwi' and '[2]' for 'Fabric Analytics'. The 'lh_wwi' node is highlighted with a red box.

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

Data destination
Choose destination settings

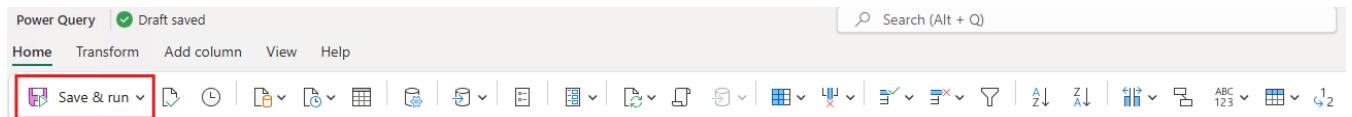
g data is represented by this schema.

Schema options on publish

Source	Source type	Destination	Destination type
<input checked="" type="checkbox"/> CustomerKey	123 Whole number	CustomerKey	Whole number
<input checked="" type="checkbox"/> WWICustomerID	123 Whole number	WWICustomerID	Whole number
<input checked="" type="checkbox"/> Customer	A Text	Customer	Text

Back Cancel Save settings

- 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.



- 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.

Name	Status	Type	Task	Owner	Refreshed
Customer Dimension Data	✓	Dataflow Ge...	—		6/11/2022
Ih_wwi			—		—
Ih_wwi			—		—

Customer Dimension Data

...

Open Delete Settings Favorite View workspace lineage View item lineage View details Move to Save as Refresh now Schedule

- 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.

The screenshot shows the Fabric interface with the following details:

- Top Bar:** Fabric, lh_wwi, Search, Trials activated: 59 days left, Lakehouse (highlighted with a red box), SQL analytics endpoint (highlighted with a red box).
- Home Tab:** Get data, New semantic model, Open notebook, Add to data agent, Manage OneLake security (preview), Update all variables.
- Message Bar:** Materialized lake views (preview) is now available in your lakehouse, try it today! A SQL analytics endpoint for SQL querying was created with this item.
- Explorer:** Add lakehouses, lh_wwi (selected), Tables, dimension_customer (highlighted with a red box).
- Table View:** dimension_customer (selected), showing 8 rows of data.

	CustomerKey	WWICusto...	Customer	BillToCusto...	Category	BuyingGroup	PrimaryCon...	PostalCode	ValidFrom	ValidTo
1	0	0	Unknown	N/A	N/A	N/A	N/A	N/A	2013-01-01T00...	2025-01-01T00...
2	102	102	Tailspin Toys (Fie...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Tea Koppel	90205	2013-01-01T00...	2025-01-01T00...
3	103	103	Tailspin Toys (Kal...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Naseem Radan	90130	2013-01-01T00...	2025-01-01T00...
4	104	104	Tailspin Toys (Wa...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Laboni Deb	90579	2013-01-01T00...	2025-01-01T00...
5	105	105	Tailspin Toys (To...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Sung-Hwan Hwa...	90400	2013-01-01T00...	2025-01-01T00...
6	106	106	Tailspin Toys (Tu...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Shiva Pipalia	90662	2013-01-01T00...	2025-01-01T00...
7	107	107	Tailspin Toys (Gl...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Karie Mercier	90782	2013-01-01T00...	2025-01-01T00...
8	108	108	Tailspin Toys (Be...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Bhanu Thota	90045	2013-01-01T00...	2025-01-01T00...

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

The screenshot shows the Fabric interface with the following details:

- Top Bar:** Fabric, lh_wwi, Search, New SQL query (highlighted with a red box), Query activity, New semantic model, Download SQL database project, Open in, New API for...
- Home Tab:** Warehouses, lh_wwi, Schemas, dbo, dimension_customer (highlighted with a red box).
- Data Preview:** dimension_customer, showing 11 rows of data.

	Customer...	WWICusto...	Customer	BillToCus...	Category	Buying...
1	0	0	Unknown	N/A	N/A	N/A
2	102	102	Tailspin Toys (...)	Tailspin Toys (...)	Novelty Shop	Kids Toys
3	103	103	Tailspin Toys (...)	Tailspin Toys (...)	Novelty Shop	Kids Toys
4	104	104	Tailspin Toys (...)	Tailspin Toys (...)	Novelty Shop	Kids Toys
5	105	105	Tailspin Toys (...)	Tailspin Toys (...)	Novelty Shop	Kids Toys
6	106	106	Tailspin Toys (...)	Tailspin Toys (...)	Novelty Shop	Kids Toys
7	107	107	Tailspin Toys (...)	Tailspin Toys (...)	Novelty Shop	Kids Toys
8	108	108	Tailspin Toys (...)	Tailspin Toys (...)	Novelty Shop	Kids Toys
9	109	109	Tailspin Toys (...)	Tailspin Toys (...)	Novelty Shop	Kids Toys
10	110	110	Tailspin Toys (...)	Tailspin Toys (...)	Novelty Shop	Kids Toys
11	111	111	Tailspin Toys (...)	Tailspin Toys (...)	Novelty Shop	Kids Toys

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.

The screenshot shows the 'New semantic model' dialog box. At the top, there's a 'Direct Lake semantic model name *' field containing 'sm_dim_customer'. Below it is a 'Workspace' dropdown set to 'Fabric Analytics'. A table preview titled 'Data preview - dimension_customer' shows 19 rows of data from the 'dimension_customer' table. On the left, the 'Explorer' sidebar shows the database structure under 'Ih_wwi'. The 'dimension_customer' table is selected in the 'Tables' section. At the bottom right of the dialog, there's a checkbox for 'Select all' and another for 'dimension_customer' which is checked and highlighted with a red box.

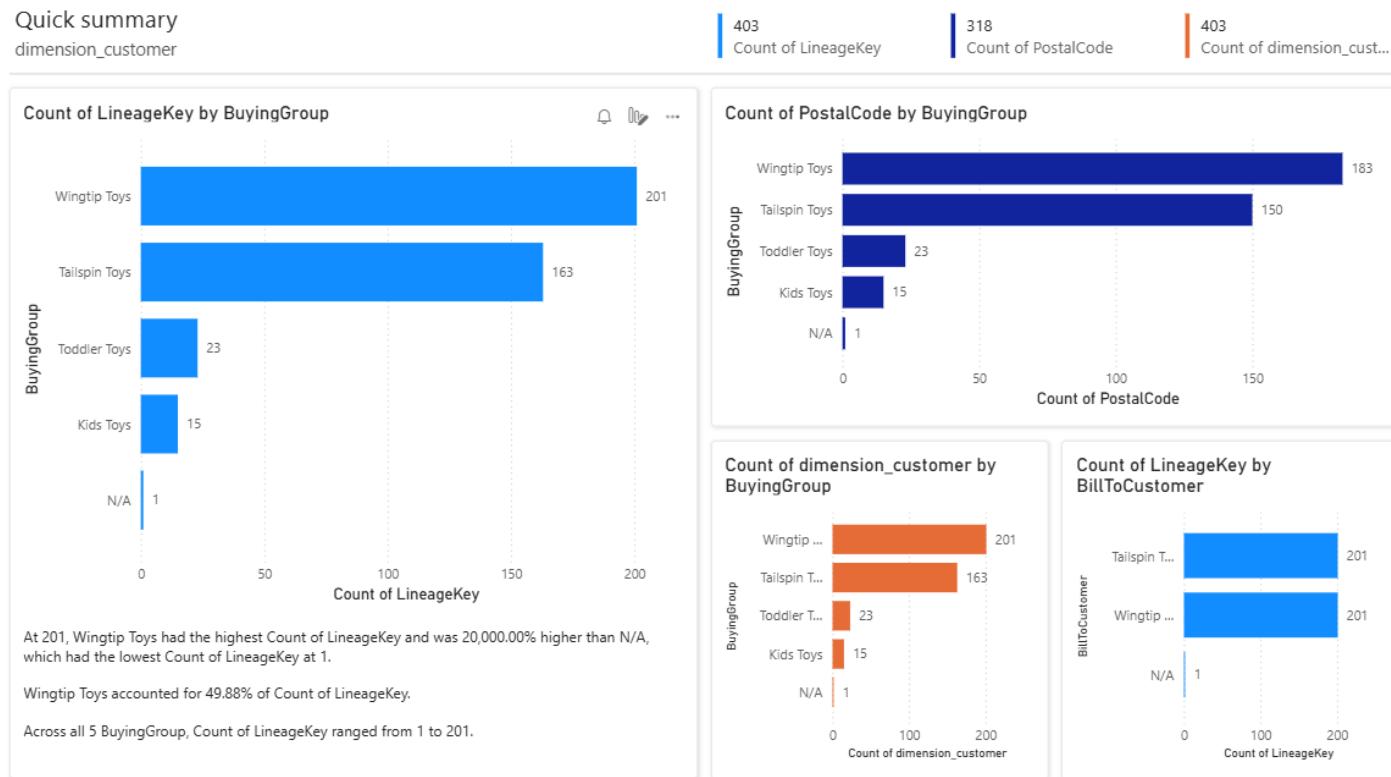
Build a report

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.

The screenshot shows the 'Explore this data' dropdown menu. It includes options like 'Discover business insights', 'Share', and 'Create a blank report'. The 'Auto-create a report' option is highlighted with a red box. To its right, there's another 'Share' button.

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.

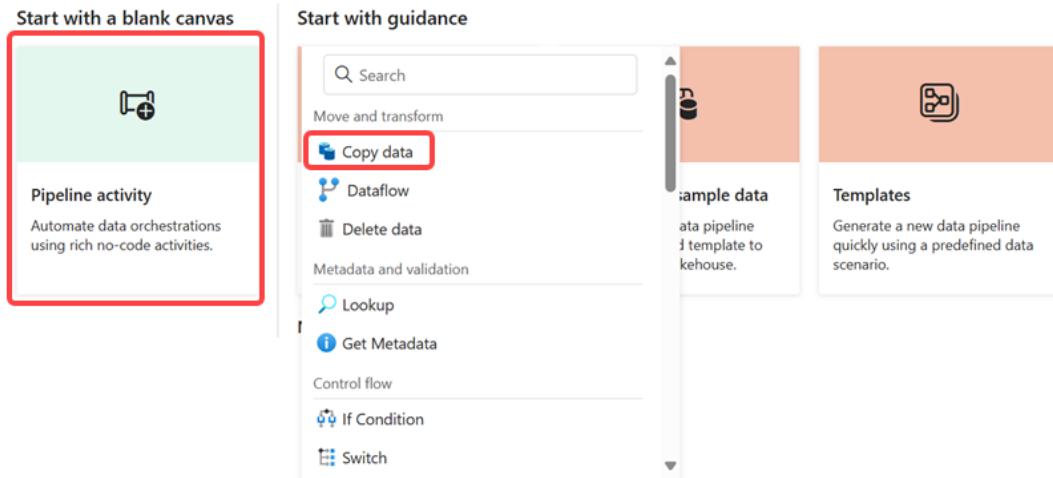
Exercise 2: Ingest data into the lakehouse

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.

Build a data pipeline to organize and move your data



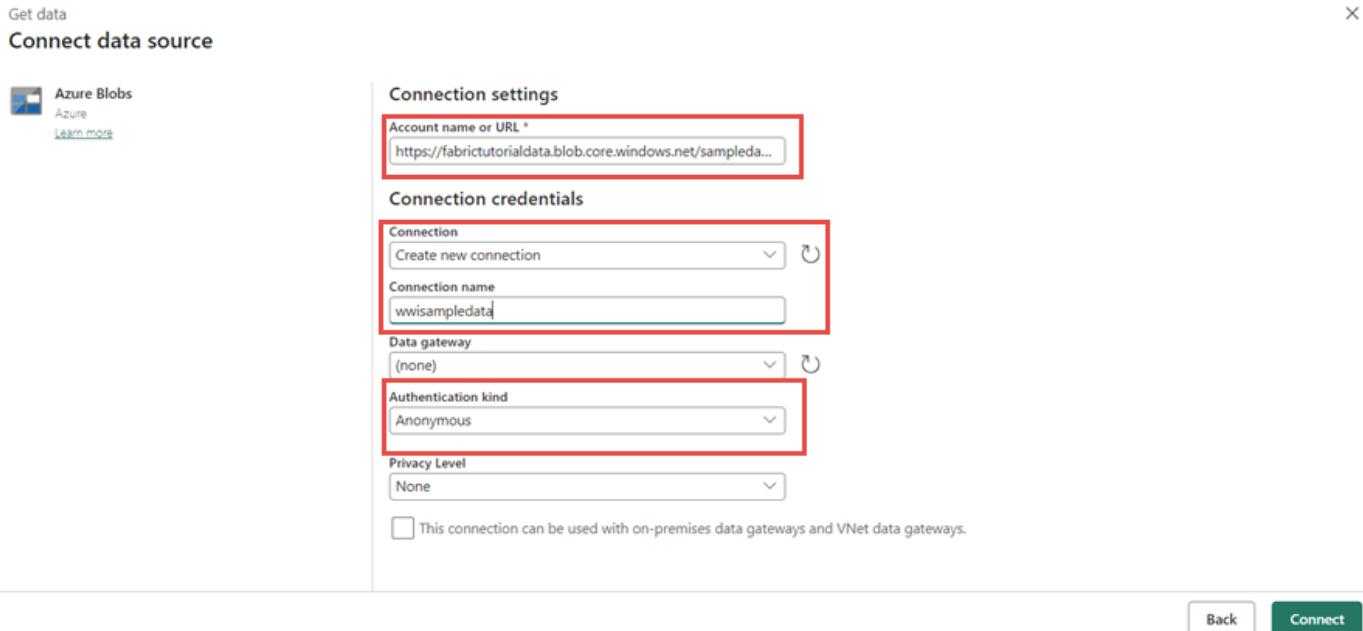
5. Select the newly added copy data activity from the canvas. Activity properties appear in a pane below the canvas (you might need to expand the pane upwards by dragging the top edge). From the **General** tab in the properties pane, type **Data Copy to Lakehouse** in the **Name** field. Leave

remaining properties to their default values.

The screenshot shows the Azure Data Factory studio interface. At the top, there's a navigation bar with 'Home' (underlined), 'Activities', 'Run', and 'View'. Below the navigation bar is a toolbar with icons for Home, New, Settings, Refresh, Validate, Run, Schedule, and View run history. A red box highlights the 'Copy data' activity card, which contains a 'Data Copy to Lakehouse' icon and several small buttons. Below the card, there are tabs: 'General' (underlined), 'Source' (with a red count '1'), 'Destination' (with a red count '1'), 'Mapping', and 'Settings'. The 'General' tab is selected. Under 'General', the 'Name' field is set to 'Data Copy to Lakehouse' (highlighted by a red box) and has a 'Learn more' link. The 'Description' field is empty. Under 'Activity state', the 'Activated' radio button is selected (highlighted by a red circle). Under 'Timeout', the value '0.12:00:00' is entered. Under 'Retry', the value '0' is entered. There's also a 'Advanced' link with a right arrow symbol.

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://fabrichtutorialdata.blob.core.windows.net/sampledata/
Connection	Create new connection
Connection name	wwisampleddata
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	wwisampleddata
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

The screenshot shows the 'Copy data' activity configuration in the Azure Data Factory designer. The 'Source' tab is selected. The 'Connection' dropdown is set to 'azure_blob_wwisampledadmin'. The 'File path type' is 'File path', with the path 'sampleddata / WideWorldImportersDW/parquet' entered. The 'File format' is 'Binary'. The 'Recursively' checkbox is checked.

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

The screenshot shows the 'Copy data' activity configuration in the Azure Data Factory studio. The 'Destination' tab is active. The configuration includes:

- Connection:** Lakehouse admin
- Lakehouse:** lh_wwi
- Root folder:** Files (radio button selected)
- File path:** 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**.

Activity name	Activity status	Run start	Duration	Input	Output
Data Copy to Lakehouse	✓ Succeeded	12/19/2023, 1:16:48 PM	31s	→	↗

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.

The screenshot shows the Azure Data Explorer interface. On the left, the 'Explorer' sidebar displays a tree structure. At the top level is 'lh_wwi'. Under 'lh_wwi', there is a 'Tables' node, which contains a single table named 'dimension_customer'. Below 'Tables' is a 'Files' node, which contains a folder named 'wwi-raw-data'. Inside 'wwi-raw-data' are several sub-folders labeled 'full', followed by five more sub-folders all named 'dimension_customer'. A red box highlights the 'wwi-raw-data' folder and its sub-folders. On the right side of the interface, the 'dimension_customer' table is shown in 'Table view'. The table has one column, 'CustomerKey', with 12 rows of data. The data is as follows:

	CustomerKey
1	0
2	102
3	103
4	104
5	105
6	106
7	107
8	108
9	109
10	110
11	111
12	112

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.

The screenshot shows the Fabric Analytics workspace. At the top, there's a navigation bar with 'New item', 'New folder', 'Import' (which is highlighted with a red box), 'Migrate', and a 'Notebook' dropdown. To the right of the dropdown is a button labeled 'From this computer' (also highlighted with a red box). A tooltip for 'From this computer' says 'Import notebook source code files from your local drive.' Below the navigation bar is a table with columns for Name, Status, and Type. The table contains several items: 'Customer Dimension Data' (Dataflow Ge...), 'IngestDataFromSourceToLakehouse' (Pipeline), 'lh_wwi' (Lakehouse), and 'lh wwi' (SQL analytics...). The 'lh_wwi' row has a red box around its Name column.

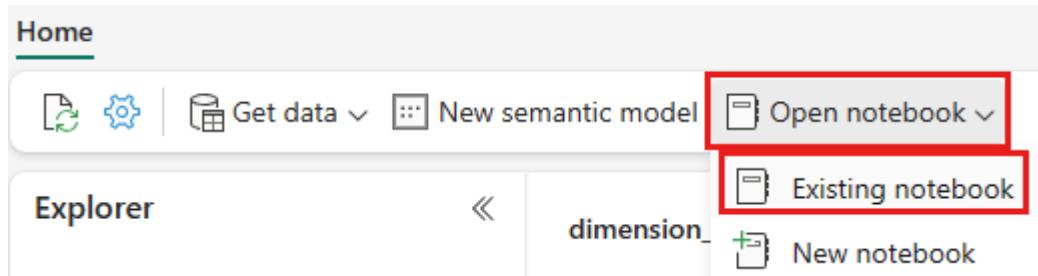
Name	Status	Type
Customer Dimension Data	Dataflow Ge...	—
IngestDataFromSourceToLakehouse	Pipeline	—
lh_wwi	Lakehouse	—
lh wwi	SQL analytics...	—

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.

The screenshot shows the items view of the workspace. It's a table with columns for Name, Status, and Type. Two rows are highlighted with a red box: '01 - Create Delta Tables' (Notebook) and '02 - Data Transformation - Business Aggregate' (Notebook). Below these, there are other items: 'Customer Dimension Data' (Dataflow Ge...), 'IngestDataFromSourceToLakehouse' (Pipeline), 'lh_wwi' (Lakehouse), 'lh wwi' (SQL analytic), 'rpt_auto' (Report), and 'sm_dim_customer' (Semantic m...). The 'lh_wwi' row is also highlighted with a red box.

Name	Status	Type
01 - Create Delta Tables		Notebook
02 - Data Transformation - Business Aggregate		Notebook
Customer Dimension Data	✓	Dataflow Ge...
IngestDataFromSourceToLakehouse		Pipeline
lh_wwi		Lakehouse
lh wwi		SQL analytic
rpt_auto		Report
sm_dim_customer		Semantic m...

8. Once the **Ih_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**.

The screenshot shows the 'Open existing notebook' dialog. It has four filter buttons: 'All' (selected), 'My data', 'Endorsed in your org', and 'Favorites'. Below is a table with columns: Name, Owner, Refreshed, and Location. Two notebooks are listed:

Name	Owner	Refreshed	Location
01 - Create Delta Tables	Pankaj Choudhary	—	Fabric Analytics
02 - Data Transformation - Business...	Pankaj Choudhary	—	Fabric Analytics

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 **Ih_wwi** lakehouse. The tables appear.
15. Go the items view of the workspace again and select the **Ih_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.

Open existing notebook

All

Name ↑

01 - Create Delta Tables

02 - Data Transformation - Business

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 wwlakehouse.fact_sale FS
INNER JOIN wwlakehouse.dimension_date DD ON FS.InvoiceDateKey = DD.Date
INNER JOIN wwlakehouse.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 **wwlakehouse** 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 semantic model

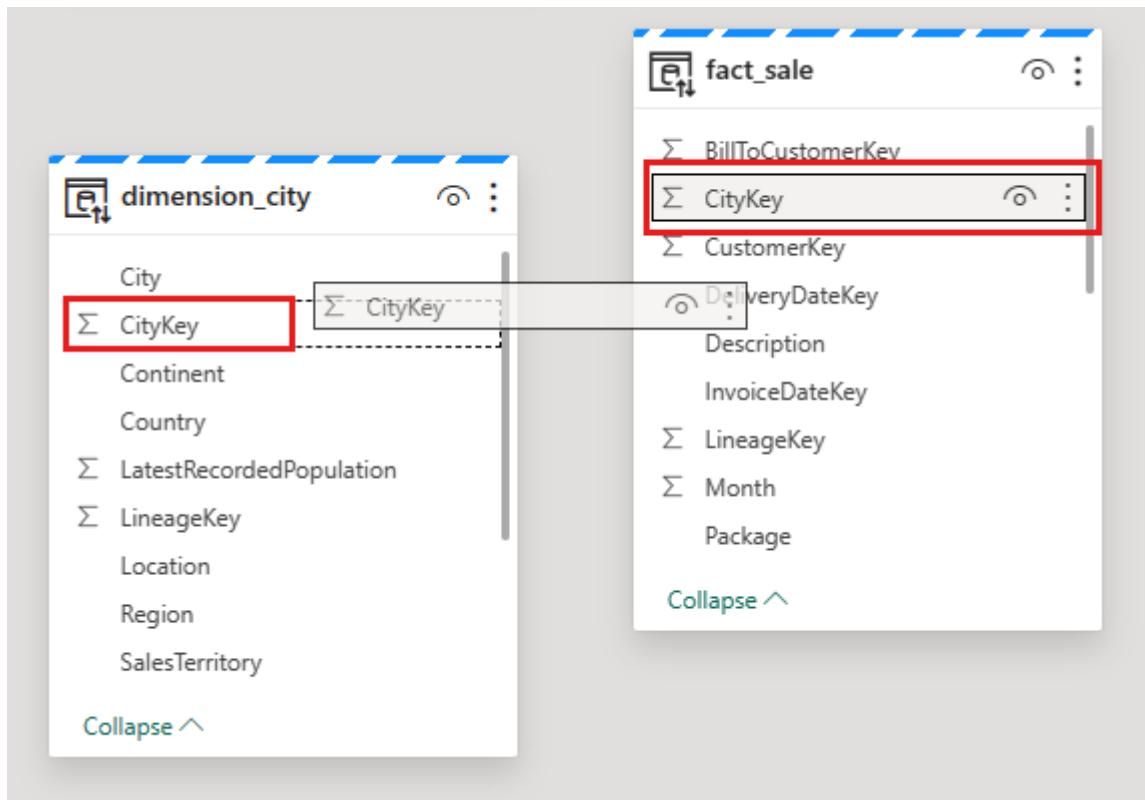
- From your **lh_wwi** lakehouse, select **SQL analytics endpoint** from the **Lakehouse** dropdown menu at the top right of the screen.

The screenshot shows the Microsoft Fabric Home interface. At the top right, there is a dropdown menu labeled "Lakehouse". A red box highlights this menu. Below it, there is a section titled "Explore your data files and folders" with a "SQL analytics endpoint" button, which is also highlighted with a red box. The main area displays a table titled "aggregate_sale_by_date_city" with 14 rows of data. On the left, there is an "Explorer" pane showing the structure of the "wwilakehouse" lakehouse, including tables like "aggregate_sale_b...", "dimension_city", etc., and files like "wwi-raw-data".

- 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.

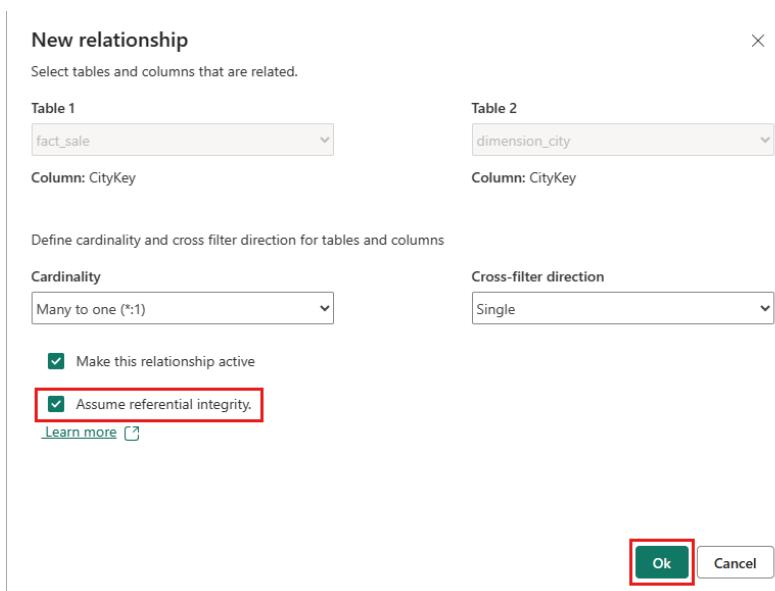
The screenshot shows the "New semantic model" dialog box. In the "Direct Lake semantic model name" field, "sm_wwi" is entered. In the "Workspace" section, "Fabric Analytics" is selected. In the "Select or deselect tables for the semantic model" section, a list of tables is shown, all of which are checked: "dimension_customer", "dimension_date", "dimension_employee", "dimension_stock_item", and "fact_sale". A red box highlights the "Select all" checkbox and the list of checked tables. At the bottom, there are "Confirm" and "Cancel" buttons.

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**.



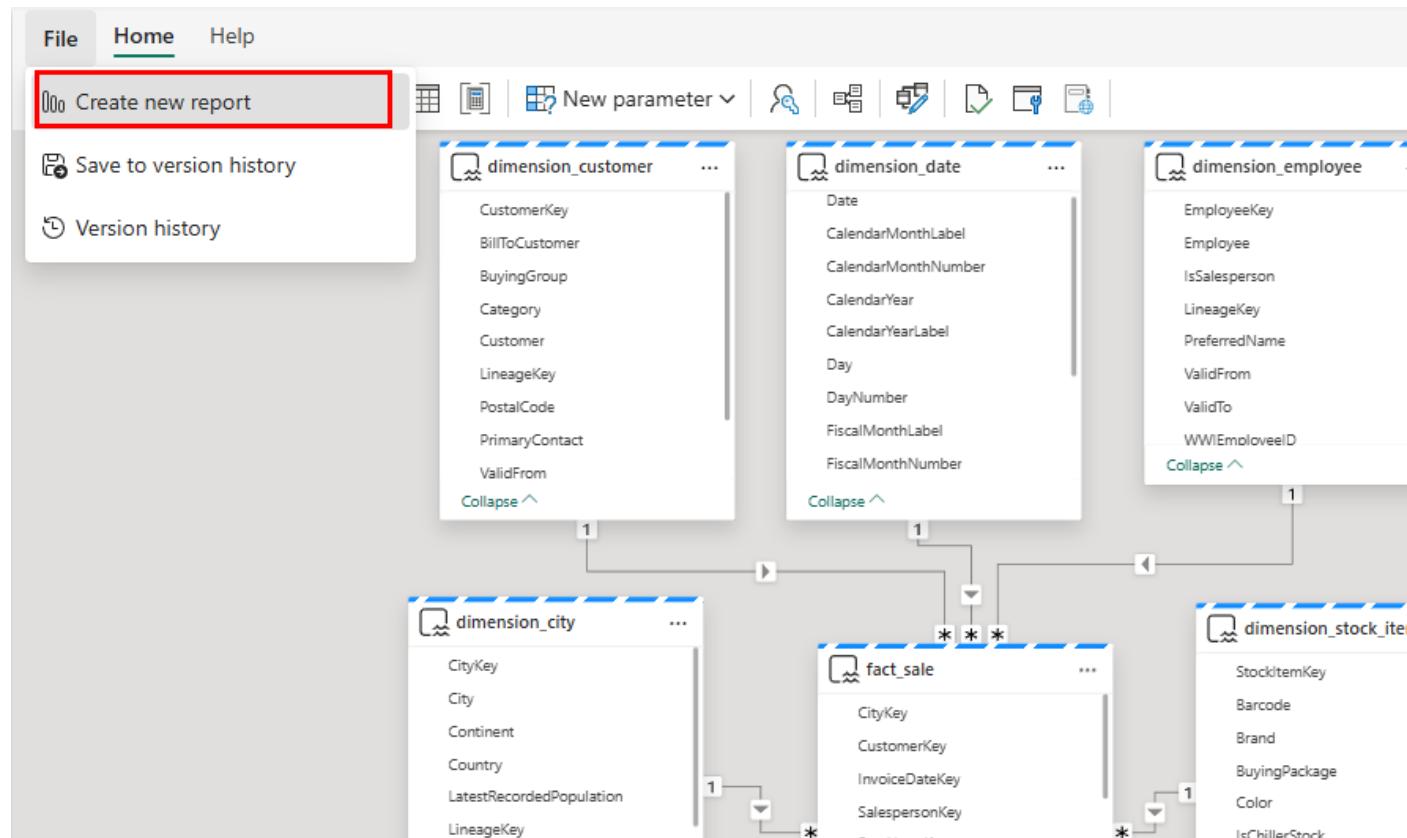
Note:

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.

The screenshot shows the Power BI Desktop interface. The top ribbon has options like File, View, Reading view, Mobile layout, Open data model, Copilot, and various icons. The left side has a 'Build visuals with your data' area with a placeholder icon. The right side has two panes: 'Visualizations' containing a grid of visualization icons, and 'Data' containing a list of fields under 'fact_sale'. A red box highlights the 'fact_sale' section in the Data pane.

Data pane:

- fact_sale
 - aggregate_sale_by_da...
 - aggregate_sale_by_da...
 - dimension_city
 - dimension_customer
 - dimension_date
 - dimension_employee
 - dimension_stock_item
 - fact_sale
 - BillToCustomer...
 - CityKey
 - CustomerKey
 - DeliveryDateKey
 - Description
 - InvoiceDateKey
 - \sum LineageKey
 - \sum Month
 - Package
 - \sum Profit
 - \sum Quantity
 - \sum Quarter
 - \sum SaleKey
 - SalespersonKey

7. Add a title:

- In the Ribbon, select **Text box**.
- Type in **WW Importers Profit Reporting**.
- Highlight the text, increase the size to 20, and move it to the upper left of the report page.

8. 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.

The screenshot shows the Power BI Desktop interface after adding a card visual. The report now includes a title 'WW Importers Profit Reporting' and a card visual below it displaying the value '17.98bn' with the label 'Sum of Profit'. The Data pane on the right shows the 'fact_sale' section expanded, with the 'Profit' checkbox checked, indicating it's part of the current visualization. A red box highlights the 'Card' icon in the Visualizations pane and the checked 'Profit' box in the Data pane.

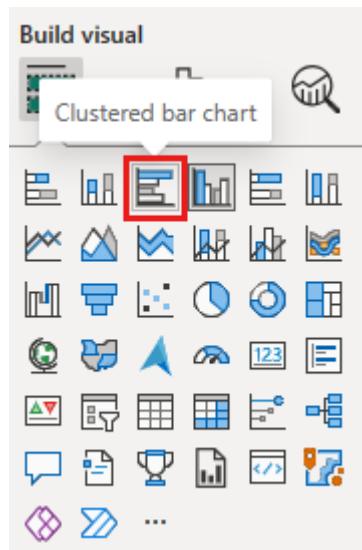
Data pane:

- fact_sale
 - Profit

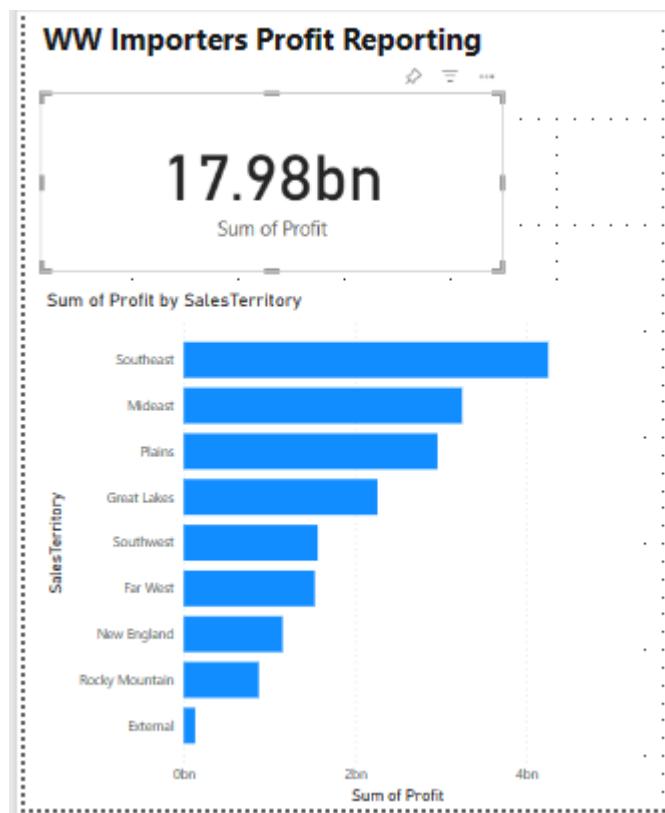
9. Add a Bar chart:

- 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.

- 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.



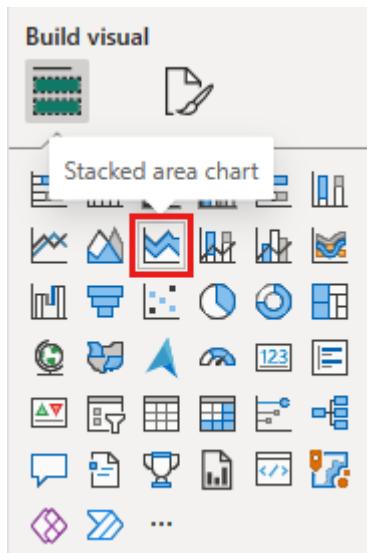
- d. Resize the Bar chart and move it under the title and Card.



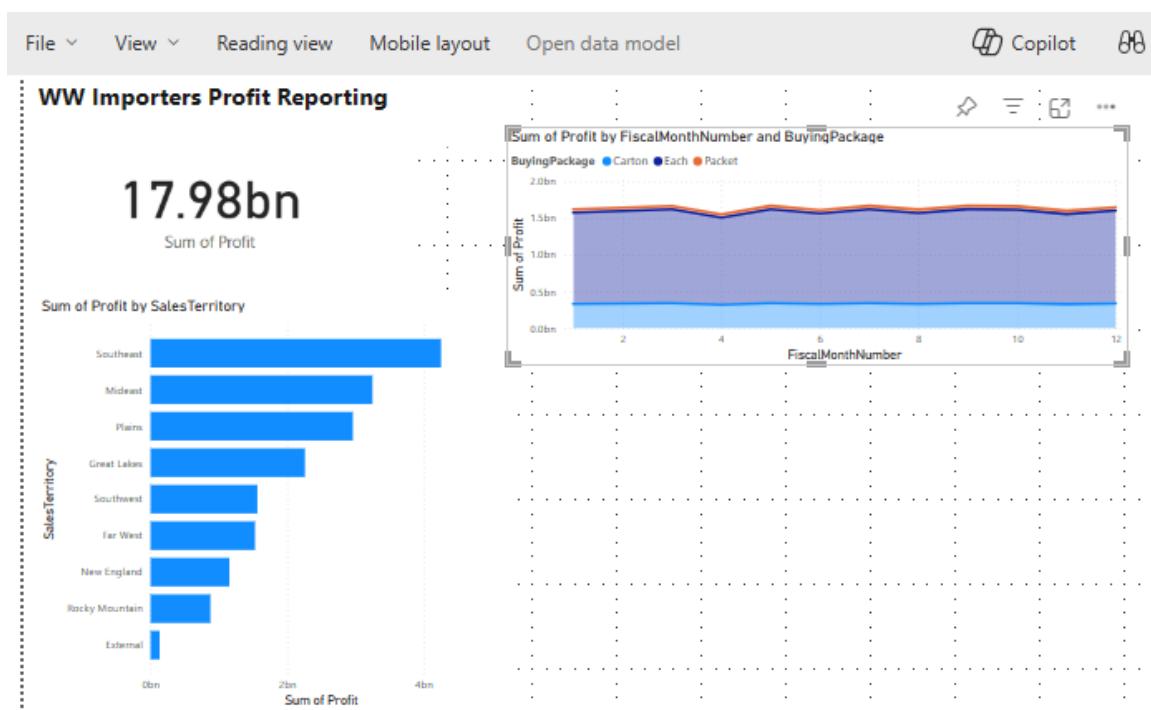
10. Click anywhere on the blank canvas (or press the Esc key) to deselect the bar chart.

11. Build a stacked area chart visual:

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



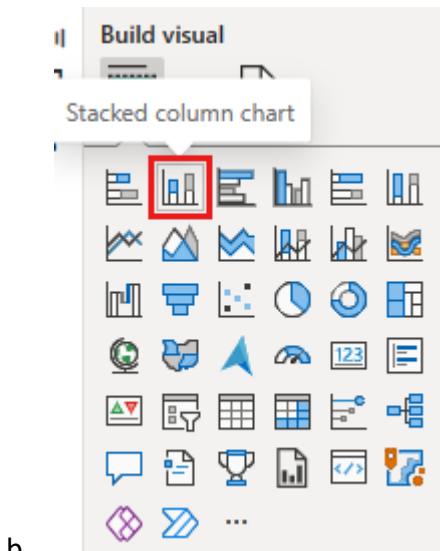
- b. Reposition and resize the stacked area chart to the right of the card and bar chart visuals created in the previous steps.
- c. On the **Data** pane, expand **fact_sales** and check the box next to **Profit**. Expand **dimension_date** and check the box next to **FiscalMonthNumber**. This selection 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.

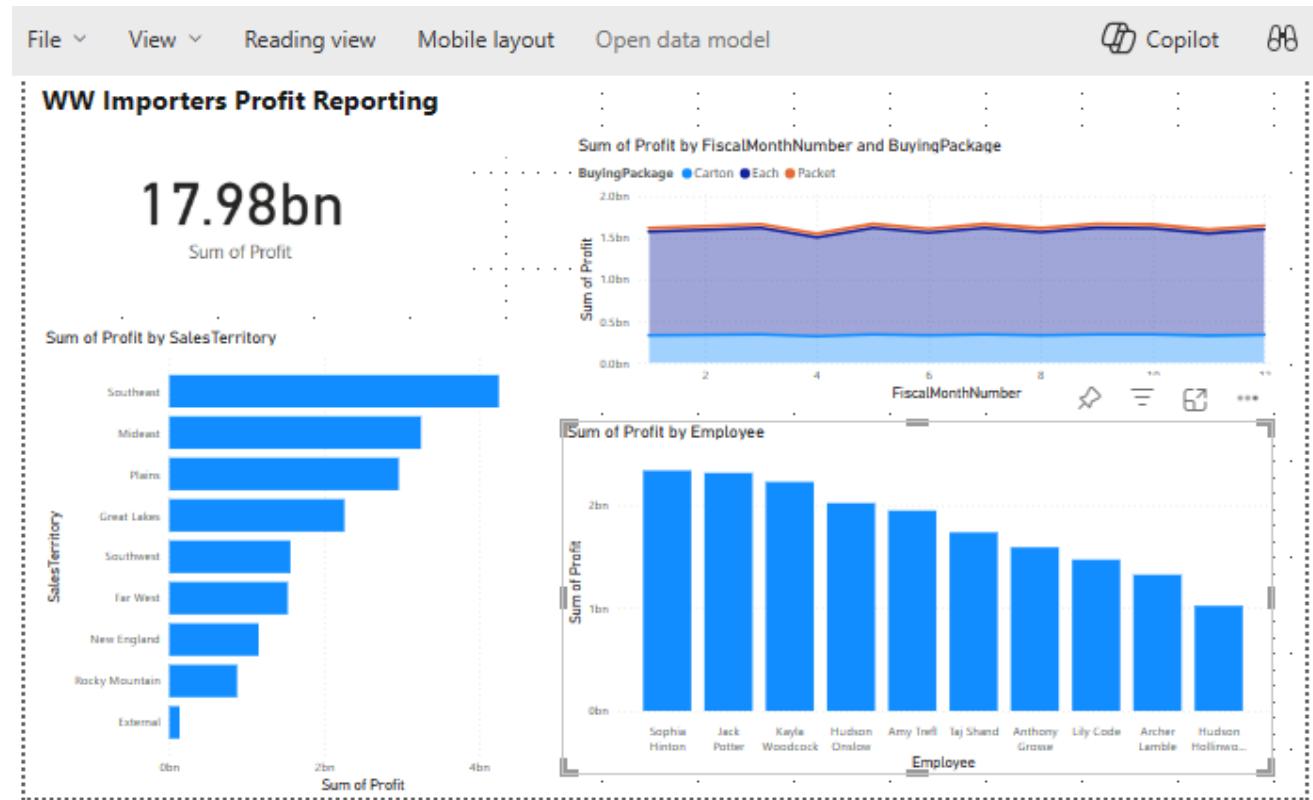
13. Build a column chart:

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



b.

- 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.



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**.