

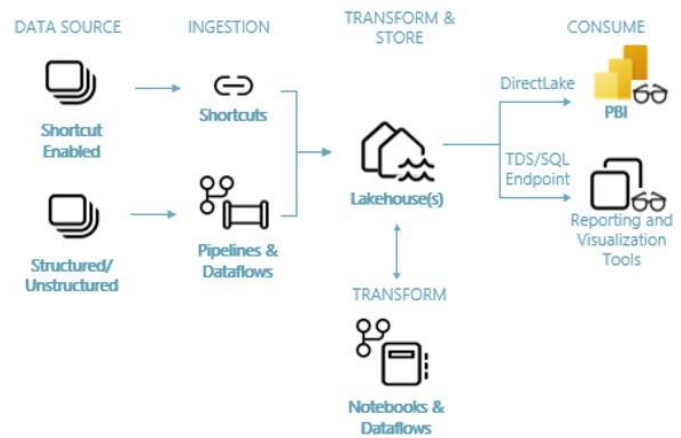
Lakehouse Tutorial

Lakehouse

Published: July 2024

Lakehouse

- **Ingestion**
Shortcuts Enabled sources for ADLS G2, Lakehouse, S3, etc. and Pipelines/Dataflows for all other data sources
- **Transformation**
Notebooks and Dataflows
- **Storage**
Lakehouse, or Lakehouse(s) with medallion structure (Bronze/Silver/Gold)
- **Consumption**
In PBI via DirectLake Mode or directly via Lake Warehouse leveraging SQL endpoint
- **Orchestration**
Pipelines triggering Notebook runs and Power BI dataset refreshes



Contents

Introduction..... 3

Module 1: Getting Started 8

 Create a Fabric workspace 8

Module 2: Build your first Lakehouse in Fabric 10

 Create a lakehouse 10

 Data Ingestion..... 11

 Building a report..... 16

Module 3: Ingest, Prep and Analyze 18

 Data Ingestion..... 18

 Data Preparation 23

 Building a report..... 31

Module 4: Clean up resources 39

Introduction

What is Fabric?

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

To get an overview of the components and concepts of Fabric read [Fabric - Overview and Concepts](#).

The purpose of this tutorial

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

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

The lakehouse end-to-end scenario

Traditionally, organizations have been building modern data warehouses for their transactional and structured data processing/analytics needs and data lakehouses for big data (semi/unstructured data) processing/analytics needs. These two systems ran in parallel, creating silos, data duplication and increased total cost of ownership.

Fabric with its unification of data store and standardization on Delta Lake format allows you to eliminate silos, remove data duplication, and drastically reduce total cost of ownership.

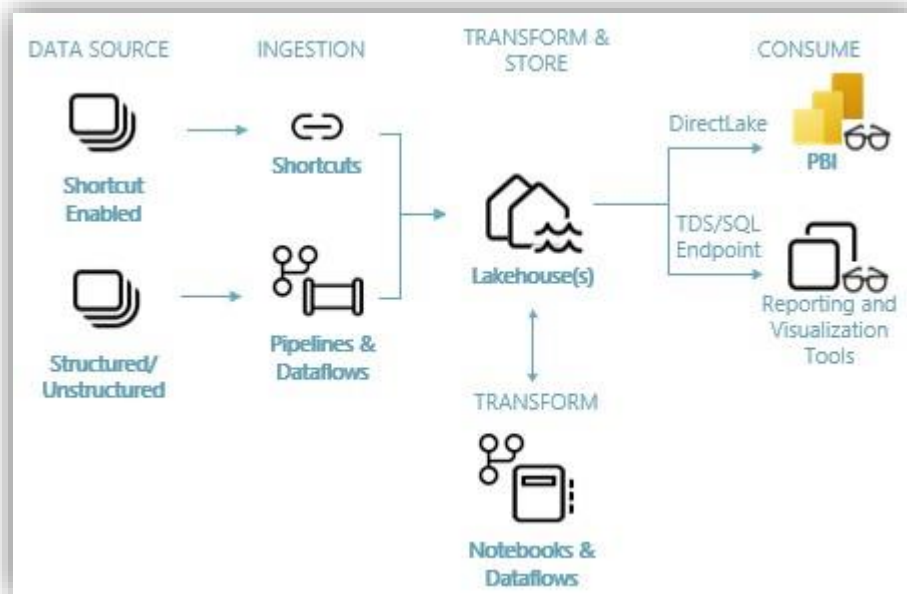
With the flexibility Fabric provides, you can implement either lakehouse or data warehouse architectures or combine these two together to get the best of both the worlds in simple implementation. In this tutorial, you are going to take an example of retail organization and build its lakehouse from start to finish. The same approach can be taken to implement a lakehouse for any organization from any industry.

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

- Sign into your Power BI online account, or if you don't have an account yet, sign up for a free trial.
- Build and implement an end-to-end lakehouse for your organization:
 - Create a Fabric workspace
 - Quickly create a lakehouse – an optional module to implement medallion architecture (Bronze, Silver, and Gold)
 - Ingest, Transform and load data into the lakehouse – bronze, silver and gold zones as delta lake tables for medallion architecture
 - Explore OneLake, OneCopy of your data across lake mode and warehouse mode
 - Connect to your lakehouse using TDS/SQL endpoint
 - Create Power BI report using DirectLake – to analyze sales data across different dimensions
 - Orchestrate and schedule data ingestion and transformation flow with Pipeline

- Cleanup resources by deleting the workspace and other items

The lakehouse end-to-end architecture



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

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

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

Consume – Data from Lakehouse can be consumed by Power BI, industry leading business intelligence tool, for reporting and visualization. Each Lakehouse comes with a built-in TDS/SQL endpoint for easily connecting to and querying data in the Lakehouse tables from other reporting tools, when needed. When a Lakehouse is created a secondary item, called a Warehouse, will be automatically generated at the same time with the same name as the Lakehouse and this Warehouse item provides you with the TDS/SQL endpoint.

The sample data

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

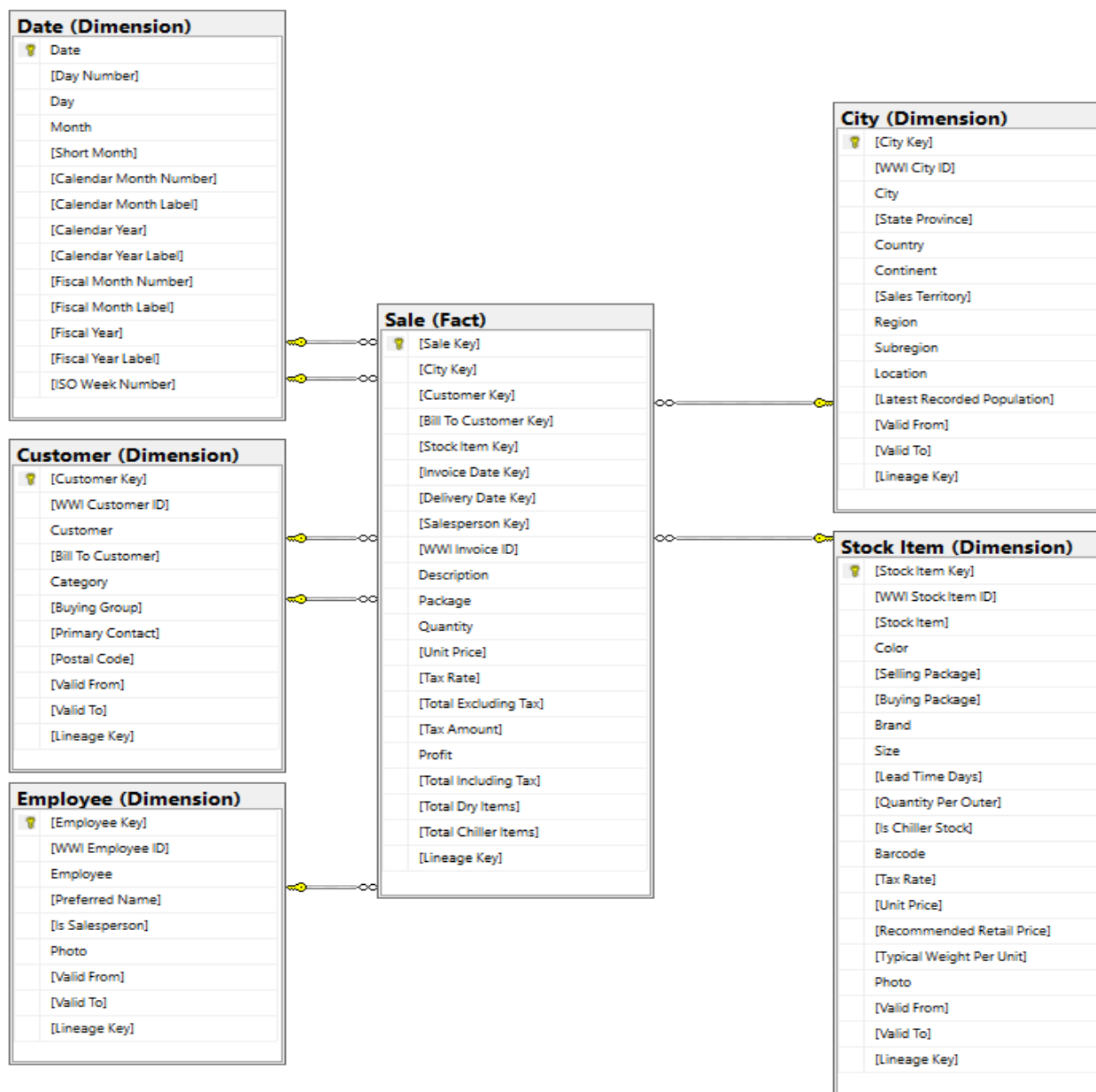
Wide World Importers (WWI) is a wholesale novelty goods importer and distributor operating from the San Francisco Bay area. As a wholesaler, WWI's customers are mostly companies who resell to individuals. WWI sells to retail customers across the United States including specialty stores, supermarkets, computing stores, tourist attraction shops, and some

individuals. WWI also sells to other wholesalers via a network of agents who promote the products on WWI's behalf. You can learn more about their company profile and operation [here](#).

Typically, you would bring data from transactional systems (or line of business applications) into a lakehouse, however for simplicity of this tutorial, we are going to use the dimensional model provided by WWI as our initial data source. We are going to use it as the source to ingest the data into a lakehouse and transform it through different stages (Bronze, Silver, and Gold) of a medallion architecture.

The data model

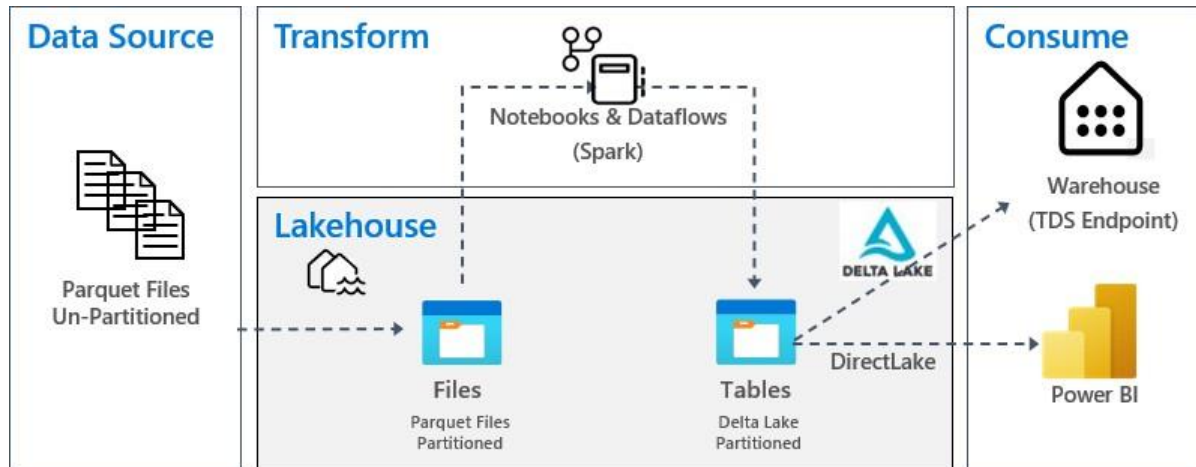
While the WWI dimensional model contains multiple fact tables, for simplicity in explanation we will focus on the Sale Fact table and its related dimensions only, as below, to demonstrate this end-to-end lakehouse scenario:



End-to-end data and transformation flow

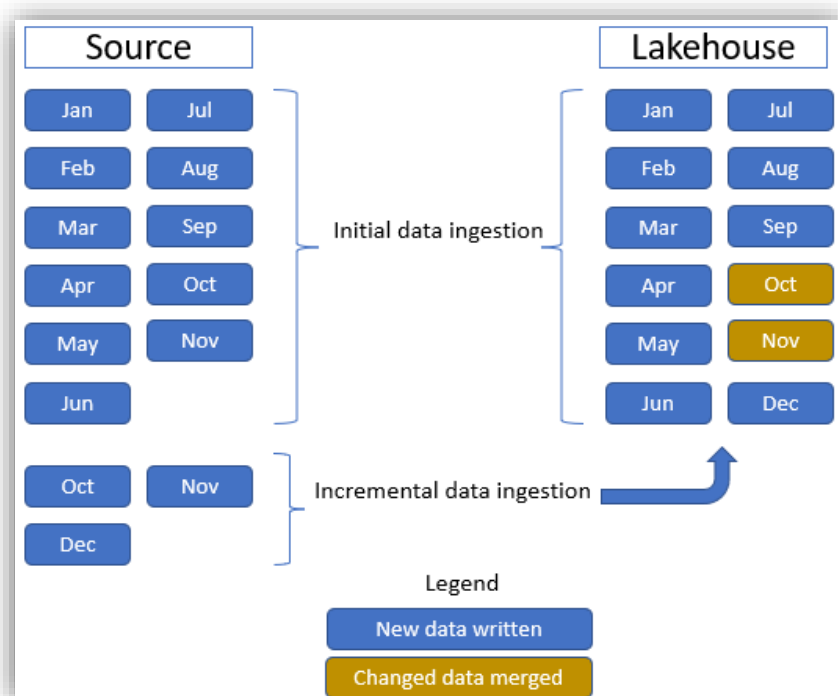
Earlier, we described the [Wide World Importers \(WWI\) sample data](#) which we are going to leverage in building this end to end lakehouse. For this implementation, this sample data is in an Azure Data storage account in Parquet file format for

all the tables, however in your real-world implementation data would likely come from varieties of sources and in a variety of formats.



Data Source – source data is in Parquet file format in an un-partitioned structure, stored in a folder for each table. For the purpose of this tutorial, we will set up pipeline to copy/ingest the complete historical or onetime data to the lakehouse.

To demonstrate process and capabilities for subsequent incremental data load, we will have an optional module at the end of this tutorial. In that module, we use a fact table (Sale) which has one parent folder with all the historical data for 11 months (one subfolder for each month) and another folder for just incremental data for 3 months (one subfolder for each month). During initial data ingestion, 11 months of data is ingested or written into the lakehouse table. However, when the incremental data arrives, it has updated data for Oct and Nov and new data for Dec, so Oct and Nov data are merged and new data for Dec is written into lakehouse table as depicted in figure below.



Lakehouse – For this implementation and for the simplicity's sake, we are going to create one lakehouse, ingest data into Files section of the lakehouse and then create delta lake tables in the Tables section of the lakehouse.

You will find an optional module at the end of this tutorial which covers creating the lakehouse with medallion architecture (Bronze, Silver, and Gold) and talks about its recommended approach.

Transform – For data preparation and transformation, we are going to demonstrate the use of Notebooks/Spark for code first users as well as demonstrate Pipelines/Dataflow for low-code/no-code users.

Consume – To demonstrate data consumption you will learn how you can use the DirectLake feature of Power BI to create reports/dashboards and directly query data from the lakehouse. Further, to demonstrate how you can make your data available to third party reporting tools, you can use TDS/SQL endpoint to connect to Warehouse and run SQL-based queries for analytics.

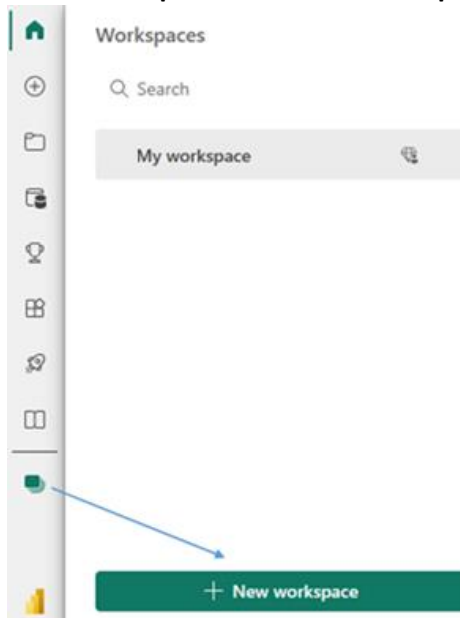
Module 1: Getting Started

Before you can begin building the lakehouse, you will need to create a workspace where you will build out the remainder of the tutorial.

Create a Fabric workspace

In this step, you create a Fabric workspace in the Power BI service. The workspace will contain all the artifacts needed for lakehouse tutorial including Lakehouse, Dataflows, Data Factory pipelines, the Notebooks, Power BI datasets, and reports.

1. Sign in to [Power BI](#).
2. Select **Workspaces** > **Create a Workspace**.



3. Fill out the **Create a workspace** form as follows:
 - a. **Name:** Enter *Fabric Lakehouse Tutorial*, and some characters for uniqueness.
 - b. **Description:** Optionally, enter a description for the workspace.

Create a workspace


Name

Available

Description

Domain (preview) ⓘ

[Learn more about workspace settings](#) ⓘ

Workspace image
 [Upload](#)
[Reset](#)

- c. **Advanced:** Under **License mode**, select **Trial** capacity. You can also choose **Fabric capacity** with F64 SKU or a Power BI **Premium capacity** with P1 SKU if you have access to them. These SKUs provide you access to all the Fabric capabilities.

Advanced ^

Contact list * ⓘ

Contoso ⓘ Enter users and groups

License mode ⓘ

☒ **Trial**

Select the free trial per-user license to try all the new features and experiences in Microsoft Fabric for 60 days. A Microsoft Fabric trial license allows users to create Microsoft Fabric items and collaborate with others in a Microsoft Fabric trial capacity. Explore new capabilities in Power BI, Data Factory, Data Engineering, and Real-Time Analytics, among others. [Learn more](#) ⓘ

☐ Premium per-user

Select Premium per-user to collaborate using Power BI Premium features, including paginated reports, dataflows, and datamarts. To collaborate and share content in a Premium per-user workspace, users need Premium per-user licenses. [Learn more](#) ⓘ

☐ Premium capacity

Select premium capacity if the workspace will be hosted in a premium capacity. When you share, collaborate on, and distribute Power BI and Microsoft Fabric content, users in the viewer role can access this content without needing a Pro or Premium per-user license. [Learn more](#) ⓘ

☐ Embedded

Select embedded if the workspace will be hosted in an Azure embedded capacity. ISVs and developers use Power BI Embedded to embed visuals and analytics in their applications. [Learn more](#) ⓘ

☐ Fabric capacity

Select Fabric capacity if the workspace will be hosted in a Microsoft Fabric capacity. With Fabric capacities, users can create Microsoft Fabric items and collaborate with others using Fabric features and experiences. Explore new capabilities in Power BI, Data Factory, Data Engineering, and Real-Time Analytics, among others. [Learn more](#) ⓘ

Default storage format

☒ Small semantic model storage format

☐ Large semantic model storage format

Apply Cancel

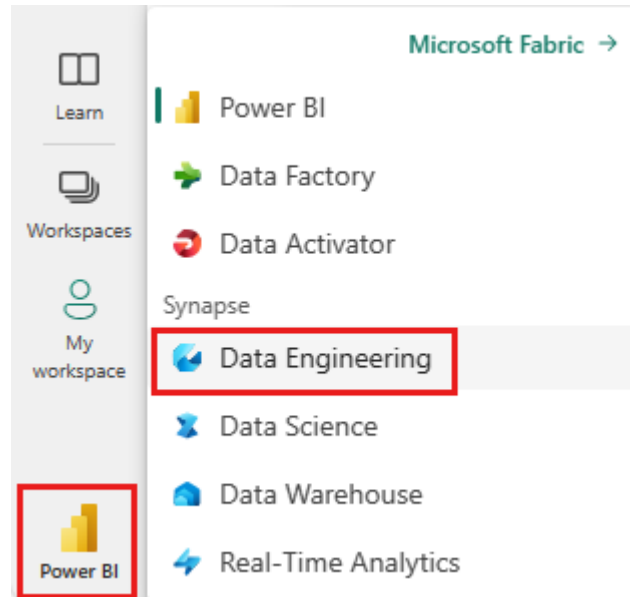
4. Select **Apply**. The workspace will be created and opened.

Module 2: Build your first Lakehouse in Fabric

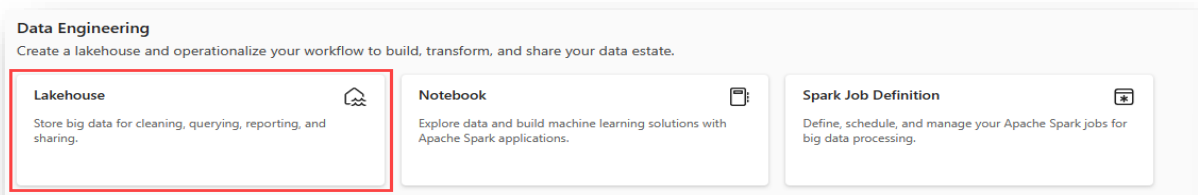
The intent of this module is to quickly build end to end journey of building a lakehouse, ingesting data for a table, applying transformation whenever required and then using ingested data into the lakehouse delta table for creating reports.

Create a lakehouse

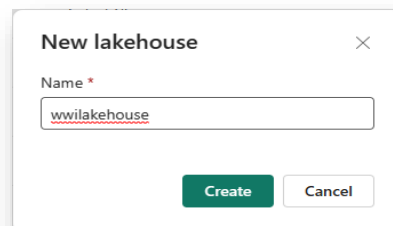
1. In the [Power BI service](#), select **Workspaces** in the left-hand menu.
2. Search for your workspace by typing in the search textbox at the top and click on your workspace to open it.
3. From the workload switcher located at the bottom left of the screen, select **Data engineering**.



4. In the **Data Engineering** section, select **Lakehouse** to create a lakehouse.



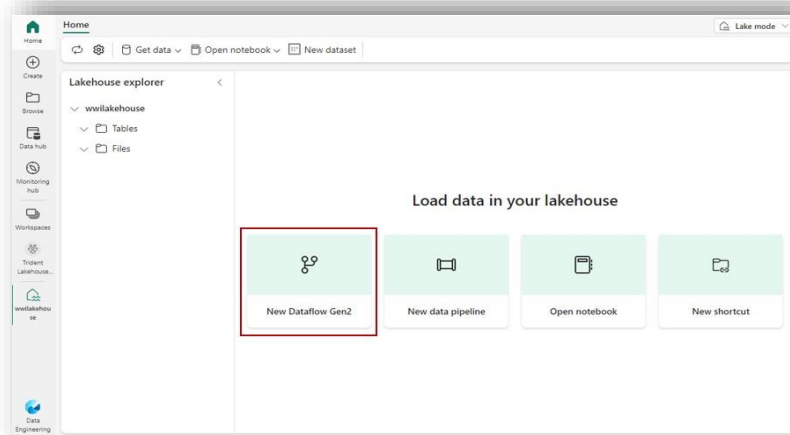
5. Enter **wwilakehouse** in the **Name** box.



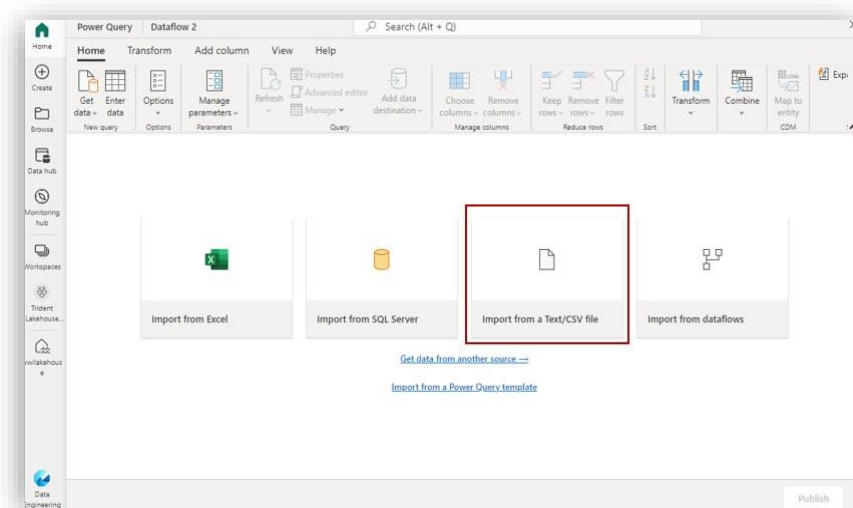
6. Click **Create**. The new lakehouse will be created and automatically opened.

Data Ingestion

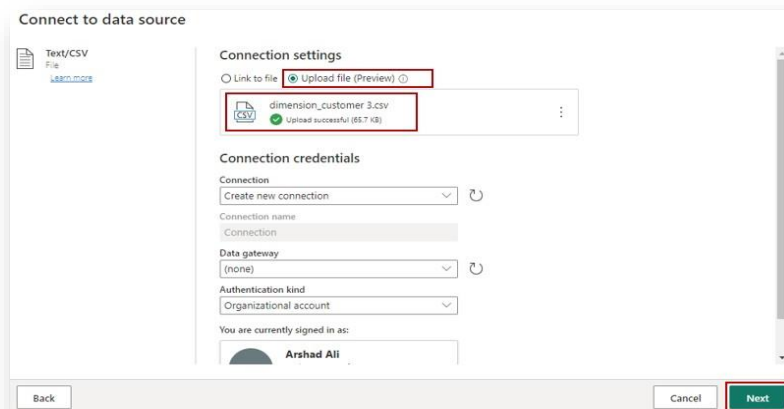
1. Download the 'dimension_customer.csv' file found in Data folder.
2. In the lakehouse view, you will see options to load data into lakehouse. Click on **New Dataflow Gen2**.



3. On the new dataflow page, click on **Import from a Text/CSV file**.

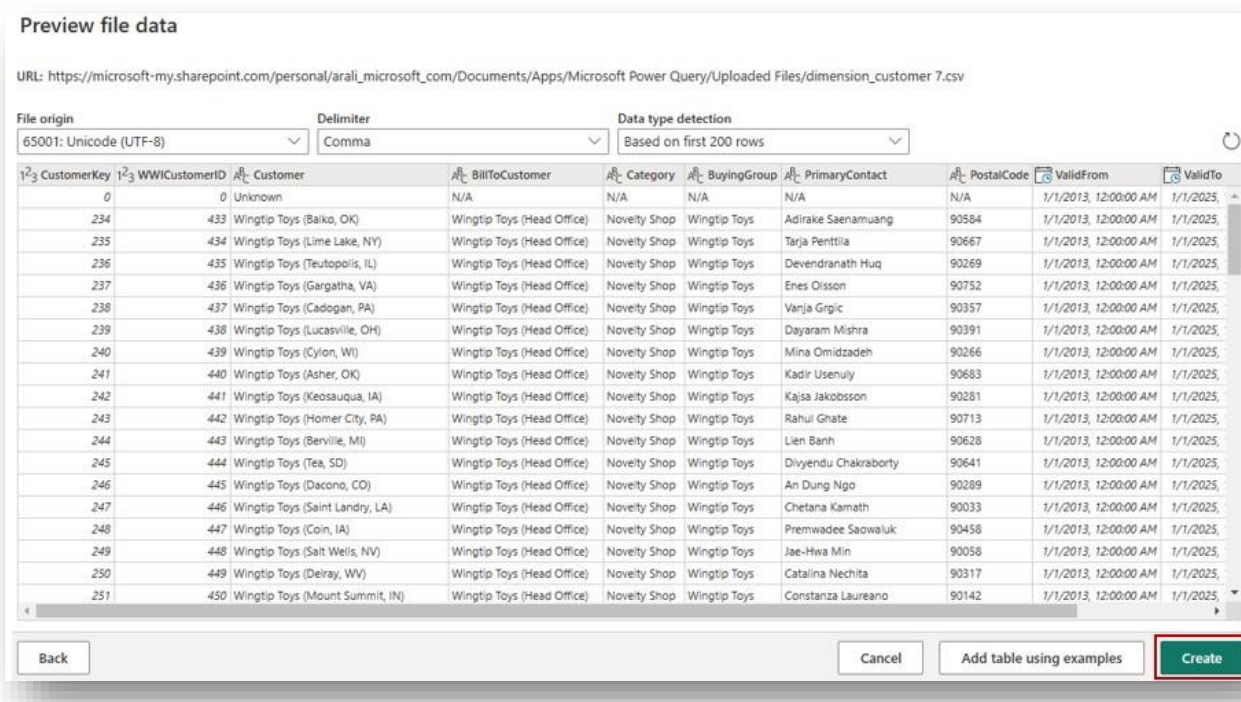


4. On the **Connect to data source** wizard, click on **Upload file** radio button and then drag and drop the data file that

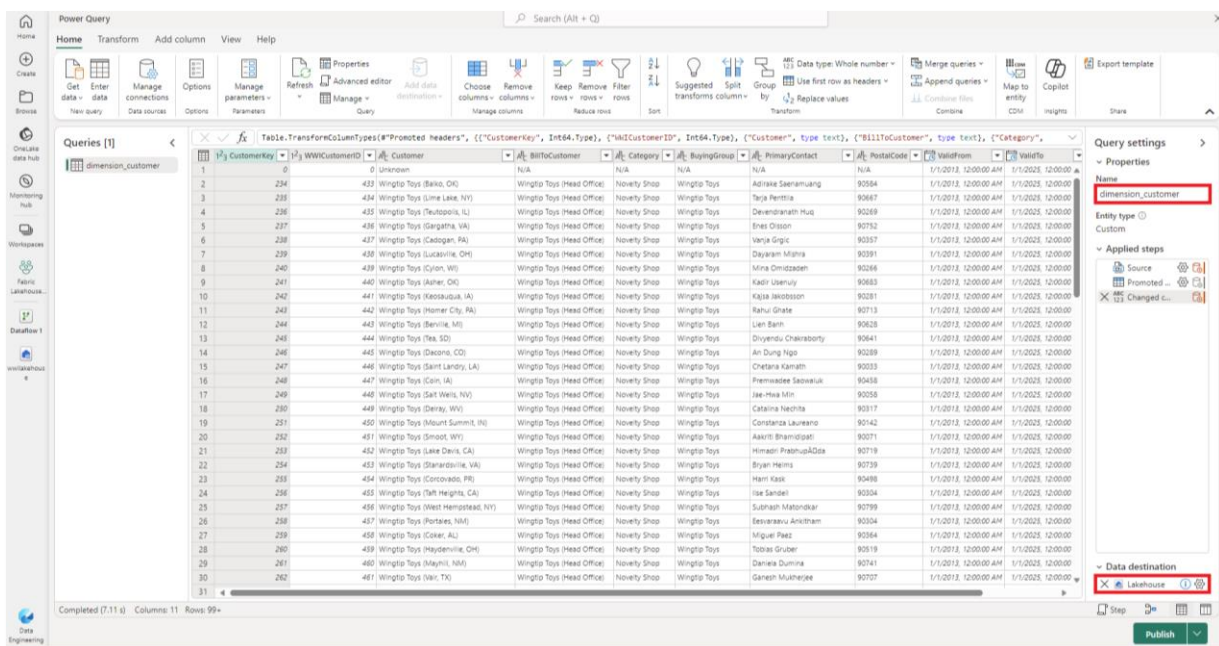


you downloaded in step 1 of this module. Once the file is uploaded, click **Next**.

- Clicking **Next** on the previous screen will open the **Preview file data** page, click on **Create** to proceed and return back to dataflow canvas.



- In the **Name** property of the query settings pane, enter **dimension_customer**. Next, select the gear in the **Data destination** property at the bottom of the query settings pane.



7. If necessary, on the **Connect to data destination** screen, sign into your account. Click **Next**.
8. Navigate to the **wwilakehouse** in your workspace.
9. If the **dimension_customer** table does not exist, select the **New table** setting and enter the Table name of **dimension_customer**. If the table already exists, select the **Existing table** setting and select **dimension_customer** from the table list in the object explorer. Select **Next**.

Note – UI adds <space> Number at the end of the table name by default. Table names must be lower case and must not contain space. Please name it appropriately and remove any space from the table name.

Choose destination target

☒ New table ☐ Existing table

Search

Display options

Fabric Lakehouse Tuto... (2)

DataflowStagingLakeh...

wwilakehouse

A new table will be created in wwilakehouse

Table name *

dimension_customer

Back

Cancel

Next

10. On the **Choose destination settings**, select **Replace** as **Update method** and click on **Save Settings**.

Choose destination settings

Update method

Existing data New data → Append Replace

Column mapping

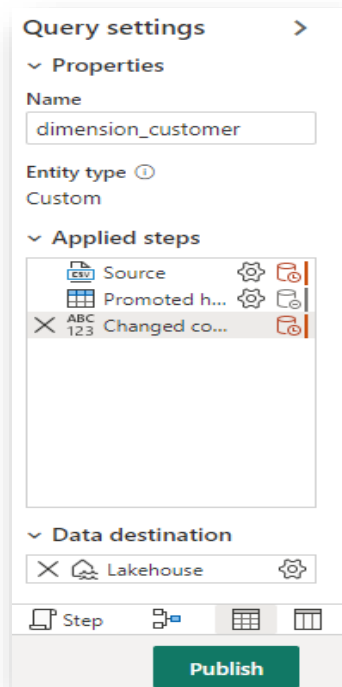
| Source | Destination | Type |
|----------------|----------------|--------------|
| CustomerKey | CustomerKey | Whole number |
| WWICustomerID | WWICustomerID | Whole number |
| Customer | Customer | Text |
| BillToCustomer | BillToCustomer | Text |
| Category | Category | Text |
| BuyingGroup | BuyingGroup | Text |

Back

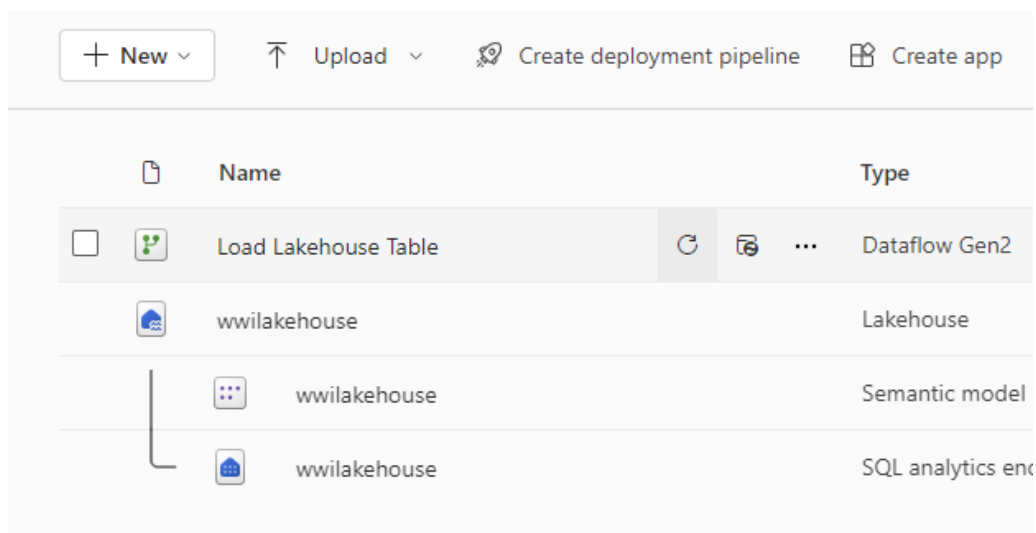
Cancel

Save settings

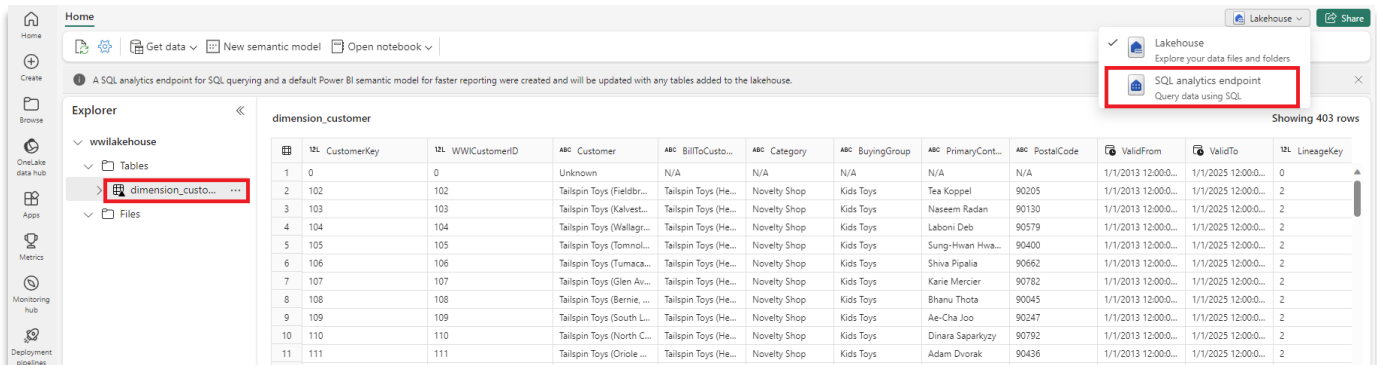
- This will return you to the canvas of the dataflow. You can easily and quickly transform the data based on your business requirements using this nice and intuitive graphical user interface. For the purpose of this module, we will not make any changes here. To proceed, click on **Publish** at bottom right of the screen.



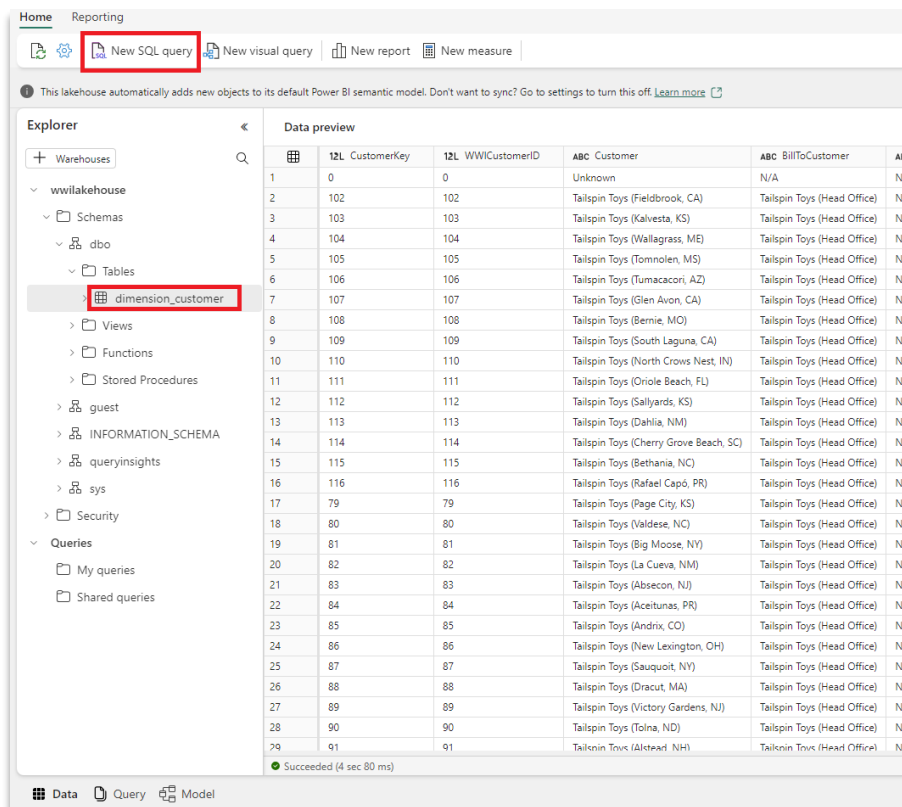
- A spinning circle next to the dataflow's name will indicate publishing is in progress on the artifact view. When this is completed, click on the ellipsis and select **Properties** to rename the dataflow. For the purpose of this module, Change the name to **Load Lakehouse Table** and select **Save**.
- Next to the name of the data flow in the artifact view, there is an icon (**Refresh now**) to refresh the dataflow. Click on it to kick off execution of dataflow and to move the data from the source file to lakehouse table. While it's in progress, you will see a spinning circle under **Refreshed** column in the artifact view.



14. Once the dataflow's refresh is completed, you can go to the lakehouse, and you will notice **dimension_customer** delta table has been created. When you click on it, you should be able to preview its data. Further, you can use the SQL endpoint of the lakehouse to query the data with SQL statements in warehouse mode. Click on **SQL endpoint** under **Lake mode** on top right of the screen.



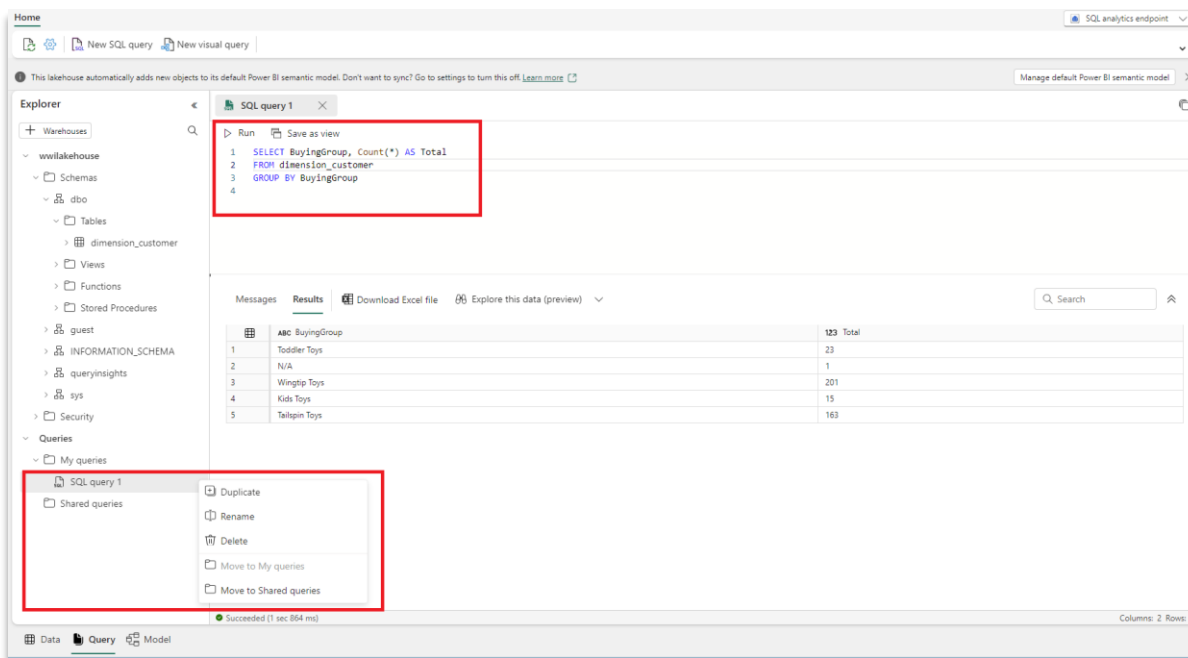
15. In the warehouse mode, you can click on **dimension_customer** table to preview its data and/or click on **New SQL query** to write your SQL statements.



16. Below is a sample query to aggregate the row count based on buyinggroup column of the **dimension_customer** table and its output. SQL query files are saved automatically for future references, and you can rename or delete these files appropriately based on your need.

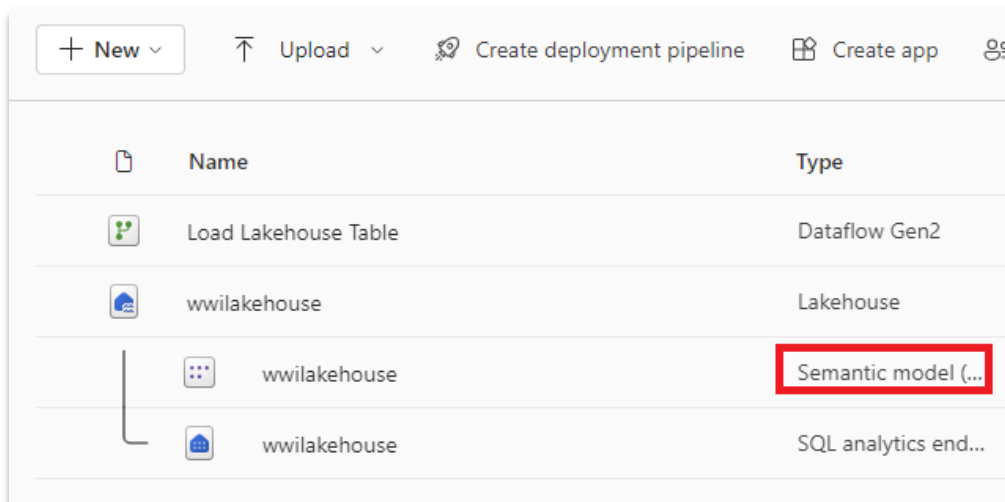
To run the script, click on the **Run** icon at the top of the script file.

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

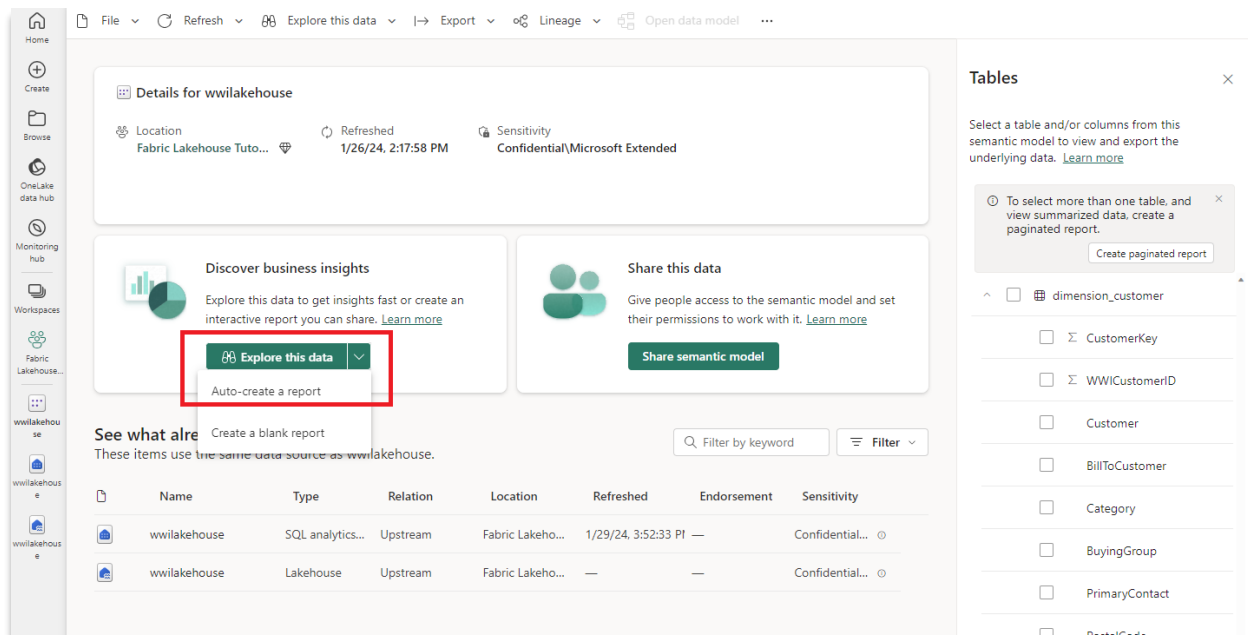


Building a report

1. In the artifact view of the workspace, click on **wwilakehouse** default semantic model, which gets created automatically with the same name of the lakehouse when you create a lakehouse.

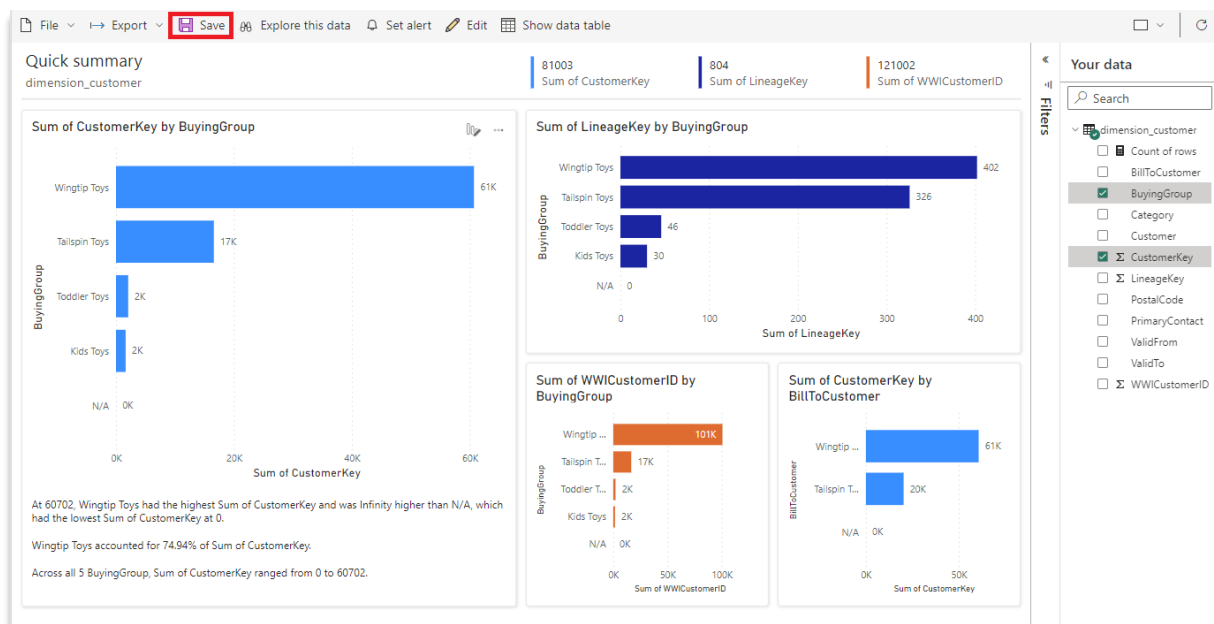


- On the semantic model screen, you can view all the tables. You will have options to create reports either from scratch, paginated reports or let Power BI do magic for you by automatically creating a report based on your data. For the purpose of this module, click on **Auto-create a report** under **Explore this data**. In the next module, we will create a report from scratch.



- Since the table is a dimension and there are no measures in it, Power BI smartly creates a measure for the row sum and aggregates it across different columns and creates different charts as below.

You can save this report for the future by clicking on **Save** button at the top ribbon and giving it a name. You can further make changes to this report to meet your requirement by including or excluding additional tables or columns.



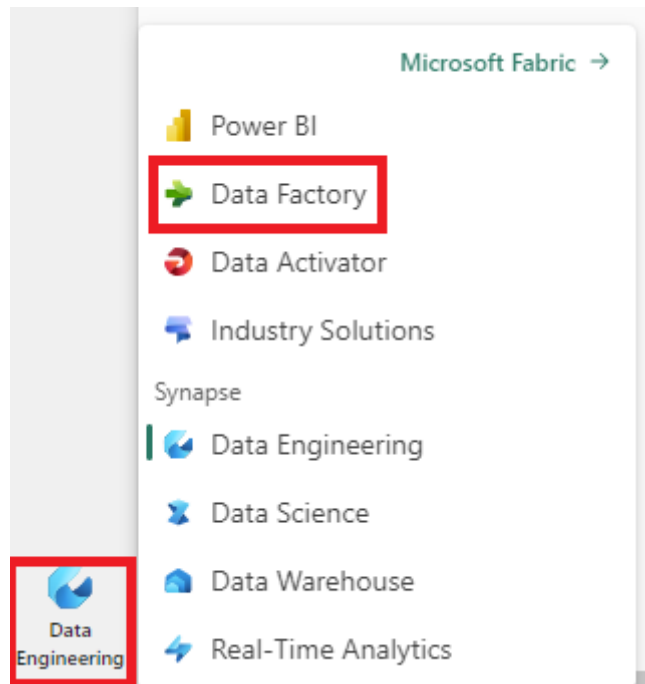
Module 3: Ingest, Prep and Analyze

This module is going to build on the work you completed in the previous module and ingest additional tables (dimensions and fact) of the dimensional model of Wide World Importers (WWI) as mentioned in the Introduction section of this document. Next, you will use notebooks with Spark runtime to transform and prepare the data. Finally, you will create Power BI data model and create a report from scratch.

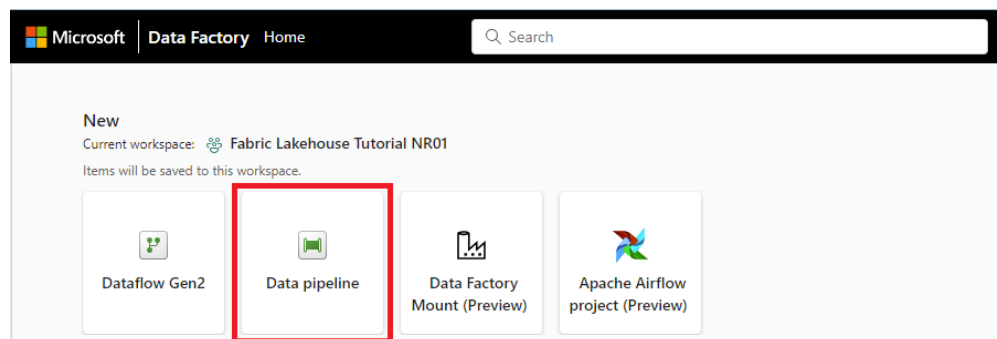
Data Ingestion

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

1. On the bottom left of the screen, click on the workload switcher, and then select **Data Factory**.



2. Click on **Data pipeline** under **New** to create a new data pipeline.



3. For the **New pipeline**, specify the name as **IngestDataFromSourceToLakehouse** and click on **Create**. This will create a new data factory pipeline and after creation, a canvas is displayed, allowing you to add tasks to the pipeline.

New pipeline

Name


IngestDataFromSourceToLakehouse

Create Cancel

Validate Run Schedule Trigger (preview) View run history Copy data Dataflow Notebook Lookup Invoke Pipeline Copilot

Build a data pipeline to organize and move your data


Start with a blank canvas



Pipeline activity


Automate data orchestrations using rich no-code activities.

Start with guidance




Copy data assistant

Follow guided steps to copy data into Microsoft Fabric, as well as other data stores.



Practice with sample data

Quickly build a data pipeline with a predefined template to load data into Lakehouse.



Templates


Generate a new data pipeline quickly using a predefined data scenario.

Need help? [Watch a demo](#)

- In the middle of the working canvas, click on 'Copy data assistant'. This guides you in the process of copying data into your lakehouse.

Build a data pipeline to organize and move your data


Start with a blank canvas



Pipeline activity


Automate data orchestrations using rich no-code activities.

Start with guidance




Copy data assistant

Follow guided steps to copy data into Microsoft Fabric, as well as other data stores.



Practice with sample data

Quickly build a data pipeline with a predefined template to load data into Lakehouse.

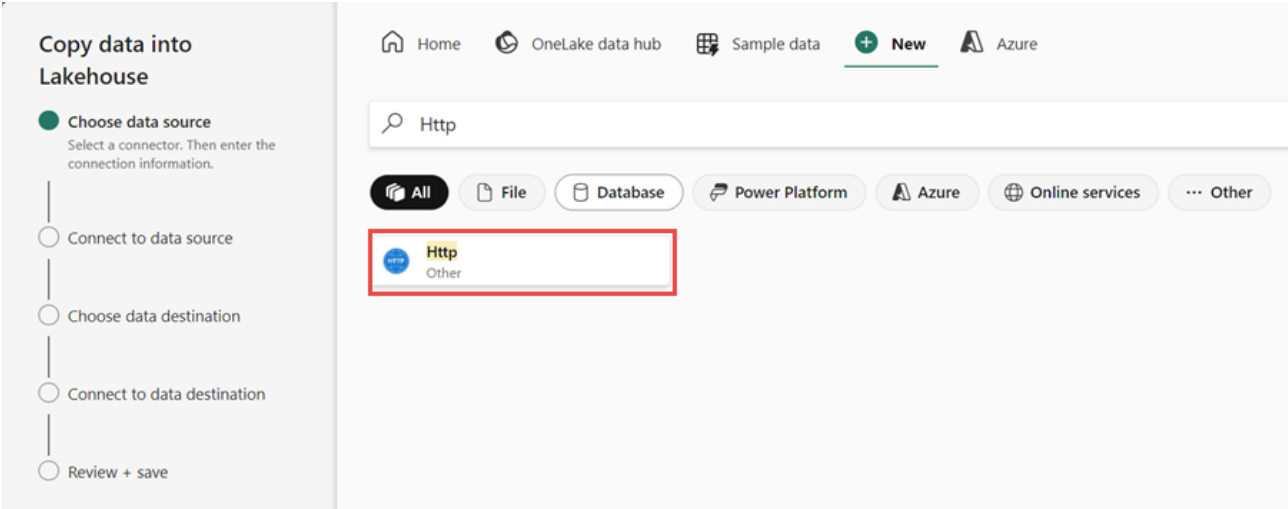


Templates

Generate a new data pipeline quickly using a predefined data scenario.

Need help? [Watch a demo](#)

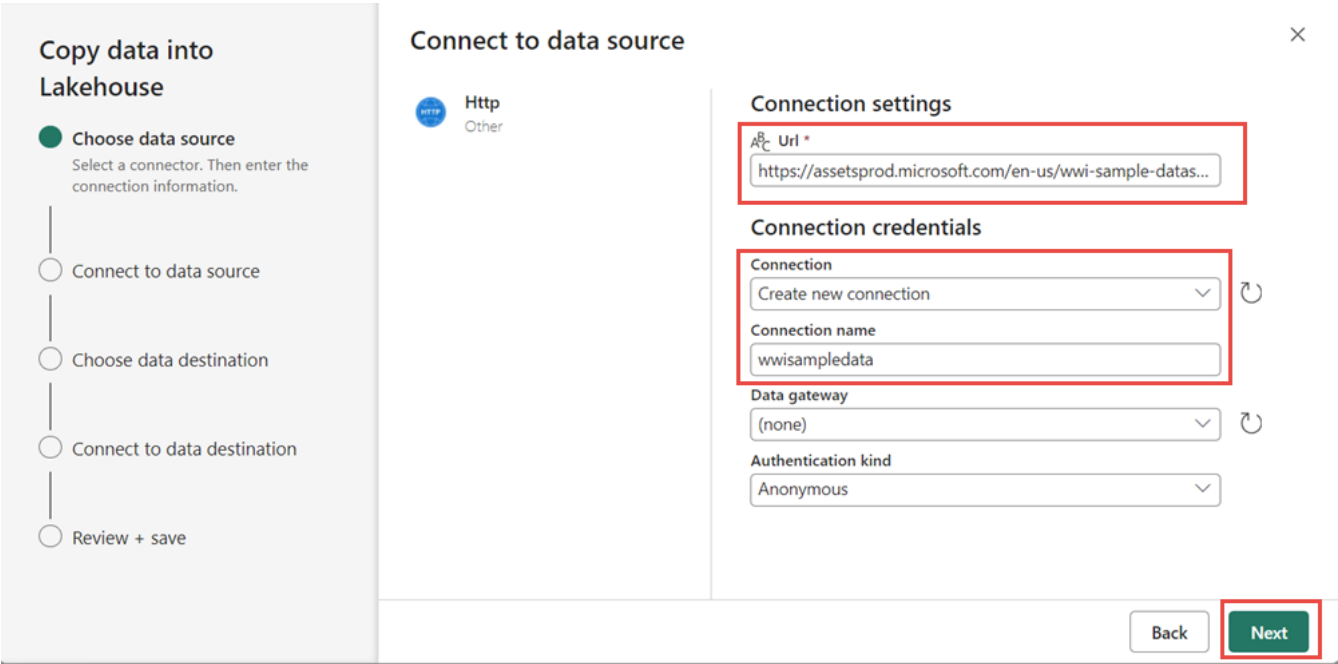
5. Next, set up an HTTP connection to import the sample World Wide Importers data into the Lakehouse. From the list of **New sources**, select **View more**, search for **Http** and select it.



6. In the **Connect to data source** window, enter the details from the table below and select **Next**.

Expand table

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



7. In the next step, enable the **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 and click **Next**.

The screenshot shows the 'Connect to data source' window. On the left, a progress bar indicates the current step is 'Connect to data source'. The main area contains fields for 'Connection' (wwisampleddata), 'Base URL', 'Relative URL', 'Request method' (GET), 'Additional headers', 'Binary copy' (checked), 'Compression type' (ZipDeflate (.zip)), 'Compression level', 'Request timeout', and 'Max concurrent connections'. The 'Binary copy' checkbox and the 'Compression type' dropdown are highlighted with a red box. At the bottom right, there are 'Back' and 'Next' buttons.

8. In the **Connect to data destination** window, specify the **Root folder** as **Files** and click **Next**. This will write the data to the *Files* section of the lakehouse.

The screenshot shows the 'Connect to data destination' window. On the left, the progress bar indicates the current step is 'Connect to data destination'. The main area contains fields for 'Connection' (wwilakehouse), 'Root folder' (Files selected), 'Folder path', 'File name', and 'Copy behavior'. The 'Root folder' section, which includes radio buttons for 'Tables' and 'Files', is highlighted with a red box. At the bottom right, there are 'Back' and 'Next' buttons, with the 'Next' button also highlighted by a red box.

9. Choose the **File format** as **Binary** for the destination. Click **Next** and then **Save+Run**. You can schedule pipelines to refresh data periodically. In this tutorial, we only run the pipeline once. The data copy process takes approximately 10-15 minutes to complete.

Copy data into Lakehouse

- ✓ Choose data source
- ✓ Connect to data source
- **Connect to data destination**
Select and map to folder path or table.
- Review + save

Connect to data destination

File format

Binary

Compression type

Select...

Back

Next

10. You can monitor the pipeline execution and activity in the **Output** tab. You can also view detailed data transfer information by selecting the glasses icon next to the pipeline name, which appears when you hover over the name.

Copy data

Copy_653

Parameters

Variables

Settings

Output

Pipeline run ID: 5244bfc3-f334-42e6-935f-606a39f9bcd0

Pipeline status: ✓ Succeeded

View run detail

Export to CSV

Filter by keyword

Showing 1 - 1 items

| Activity name | Activity status | Run start | Duration | Input | Output |
|---------------|--------------------------|-----------------------|----------|-------|--------|
| Copy_653 | ✓ Succeeded | 7/16/2024, 3:03:42 PM | 18m 27s | | |

11. After the successful execution of the pipeline, go to your lakehouse (**wwilakehouse**) and open the explorer to see the imported data.

Home

Create

Browse

OneLake data hub

Monitoring hub

Workspaces

My workspace

My workspace

+ New

Upload

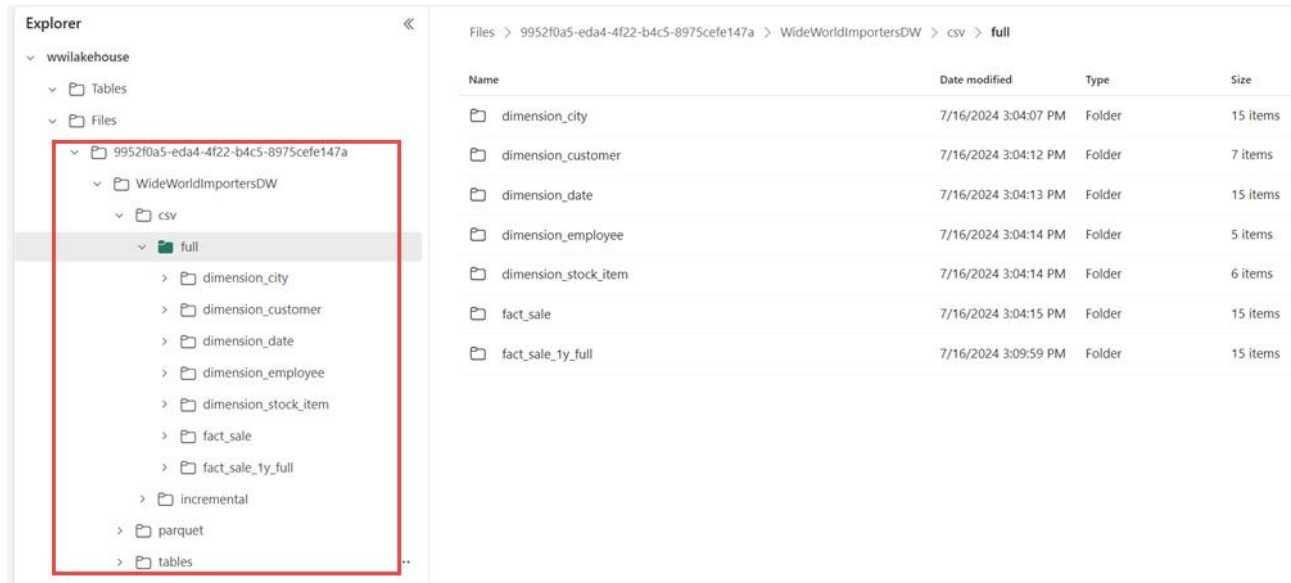
Workspace settings

Filter by keyword

Filter

| Name | Type | Owner | Refreshed | Next refresh |
|---------------------------------|----------------------|---------------|----------------------|--------------|
| Automatic report | Report | My workspace | 12/19/23, 7:58:11 AM | — |
| IngestDataFromSourceToLakehouse | Data pipeline | Contoso Admin | — | — |
| Load Lakehouse Table | Dataflow Gen2 | Contoso Admin | 12/19/23, 8:38:41 AM | N/A |
| wwilakehouse | Lakehouse | Contoso Admin | — | — |
| wwilakehouse | Semantic model (...) | My workspace | 12/19/23, 7:58:11 AM | N/A |
| wwilakehouse | SQL analytics end... | My workspace | 12/19/23, 8:49:40 AM | N/A |

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

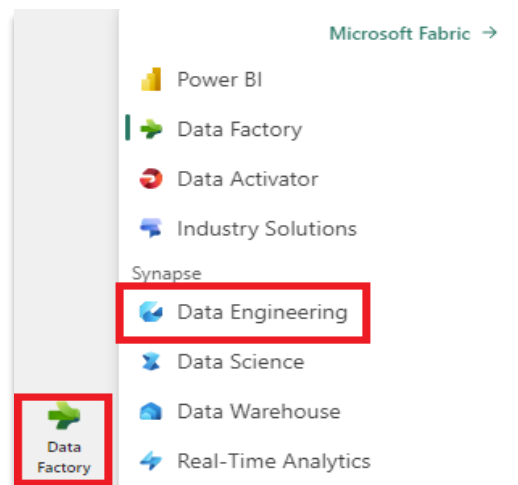


13. The data is created under the **Files** section of the lakehouse explorer. A new folder with GUID contains all the needed data. Rename the GUID to **wwi-raw-data**.

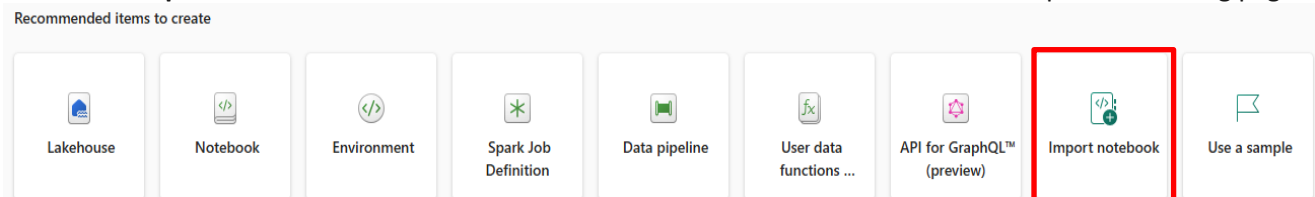
Data Preparation

Now that we have the raw data already ingested from source to the **Files** section of the lakehouse, you can take this data, transform it and prepare it to create delta tables.

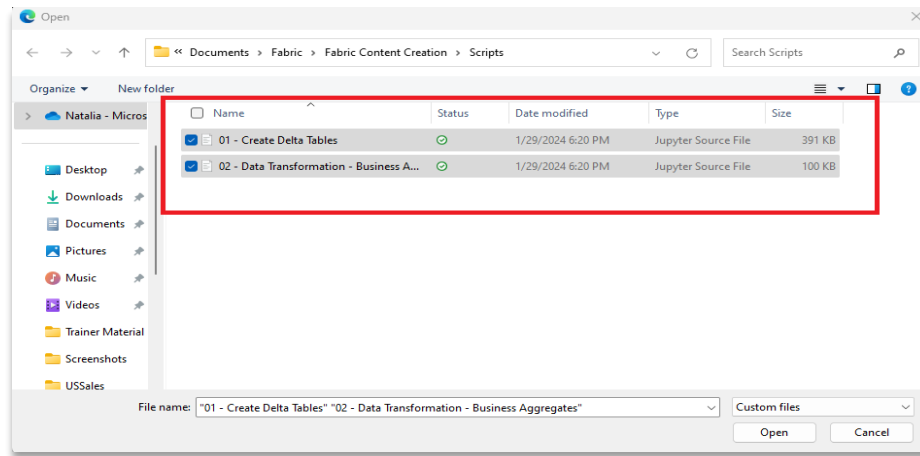
1. Download the set of notebooks found in the Scripts folder.'
2. From the workload switcher located at the bottom left of the screen, select **Data engineering**.



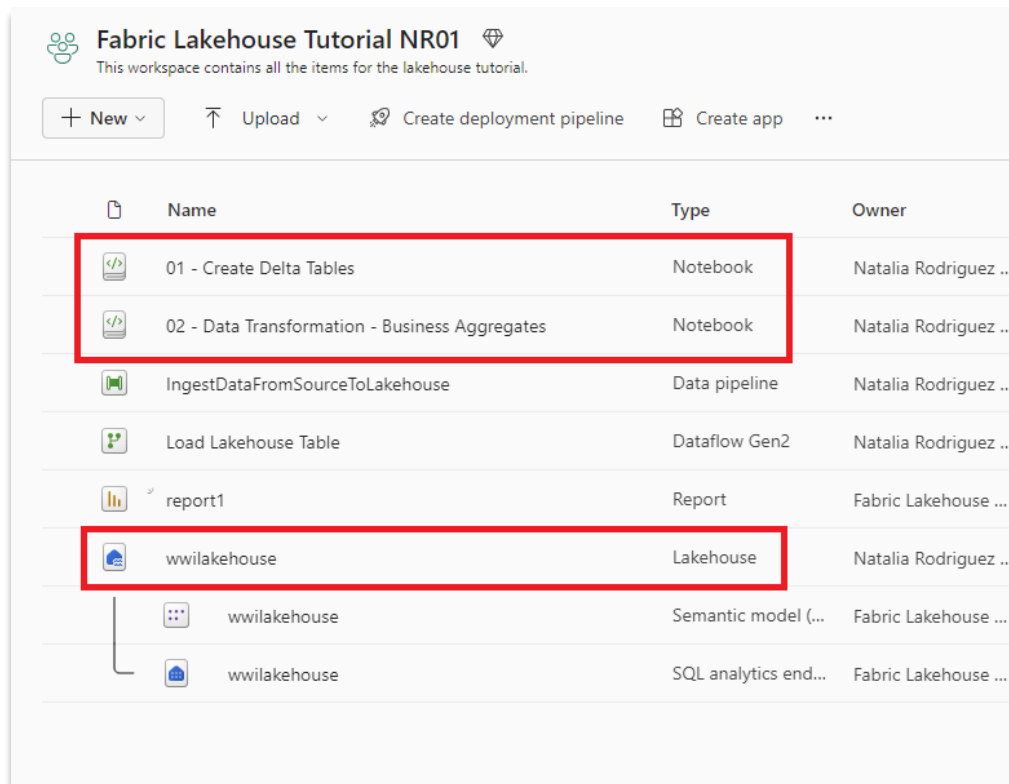
3. Select **Import notebook** from the '**Recommended items to create**' section at the top of the landing page.



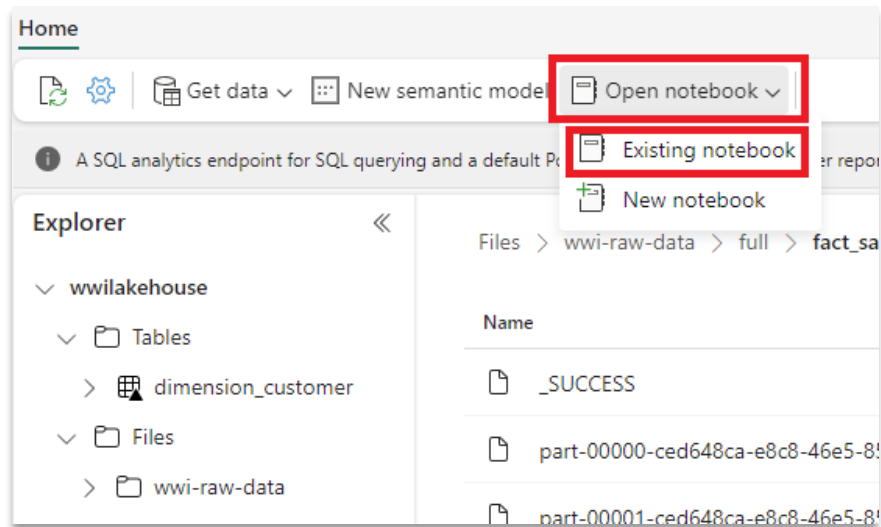
4. Select **Upload** from the **Import status** pane that opens on the right side of the screen.
5. Select all the notebooks that were downloaded and/or unzipped in step 1 of this section.
6. Select **Open**. A notification indicating the status of the import will appear in the top right corner of the browser window.



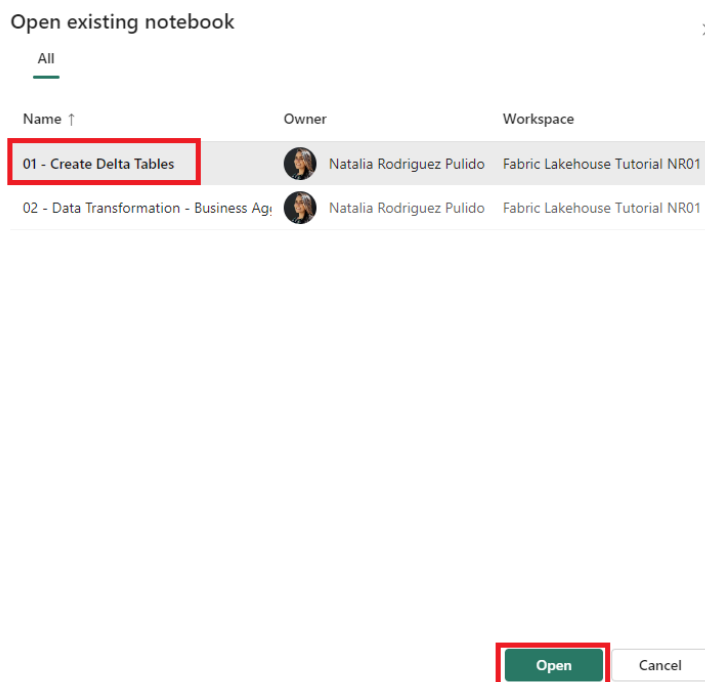
7. After the import of notebooks is successful, you can go to items view of the workspace and see these newly imported notebooks. To open a notebook, first, click on **wwilakehouse** lakehouse to open your lakehouse.



8. Once the **wwilakehouse** lakehouse is opened, click on **Open notebook** > **Existing notebook** on the ribbon at the top.



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



10. Once the notebook is opened, in the **Lakehouse explorer** you will notice the notebook is already linked to your opened lakehouse.

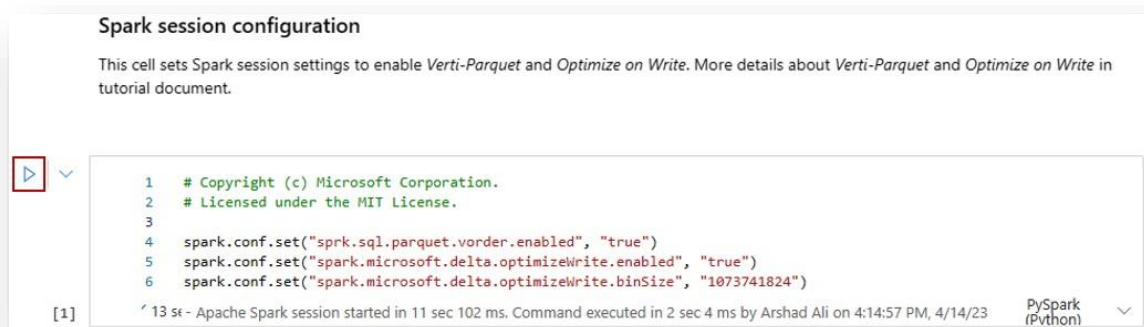
Note

Fabric provides these unique capabilities for writing optimized delta lake files:

- **Verti-Parquet** – Fabric includes Microsoft's unique VertiParquet IP. VertiParquet transparently optimizes the Delta Lake files in a way that is highly optimized by Fabric compute engines, often resulting in 3x-4x compression improvement and up to 10x performance acceleration over Delta Lake files not optimized using VertiParquet while still maintaining full Delta Lake format compliance.
- **Optimize write** – Apache Spark performs most efficiently when using standardized larger file sizes. The relation between the file size, the number of files, the number of Spark workers and Spark's configurations play a critical role in performance. Ingestion workloads into Delta Lake tables may have the inherited characteristic of constantly writing lots of small files; this scenario is commonly known as the "small files problem". To overcome this problem, Spark in Fabric includes an Optimize Write feature that reduces the number of files written and aims to increase individual file size of the written data. It dynamically optimizes partitions while generating files with a default 128 MB size. The target file size may be changed per workload requirements using configurations.

11. Before you write data as delta lake tables in the **Tables** section of the lakehouse, you are going to use two features (**Verti-Parquet** and **Optimize Write**) of Fabric for optimized data writing and for subsequent better reading performance. To enable these features in your session you can set these configurations in the first cell of your notebook - eventually these features will be by default enabled for the Spark session.

To start execution, click **Run All** under **Home** at the top to start execution of the notebook and all its cells in the sequence or click **Run** icon on the left of the cell or hit **SHIFT+ENTER** while control is in the cell to execute code from that specific cell.



```
Spark session configuration

This cell sets Spark session settings to enable Verti-Parquet and Optimize on Write. More details about Verti-Parquet and Optimize on Write in tutorial document.

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')

[1] 13 s - Apache Spark session started in 11 sec 102 ms. Command executed in 2 sec 4 ms by Arshad Ali on 4:14:57 PM, 4/14/23 PySpark (Python)
```

The first thing you would have noticed when executing this cell is that you didn't have to specify the underlying Spark pool or cluster details as Fabric provides it through the concepts of Live Pool i.e., every Fabric workspace comes pre-wired with a default Spark pool, called Live Pool. This implies that when users create notebooks, they are relieved from the task of specifying any Spark configurations or cluster details. Upon executing their first command, the live pool activates and establishes the Spark session within seconds, subsequently executing the code in the cell. As long as the Spark session remains active, subsequent code executions in this notebook proceed almost instantaneously.

12. Next, you will execute the 'Fact-Sale' cell which reads raw data from the **Files** section of the lakehouse, add

additional columns for the purpose of saving different date parts as an action of the transformation process. Finally, you will use the `partitionBy` Spark API method to partition the data before writing it as a delta table based on the newly created data part columns (Year and Quarter).

Fact - Sale

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.

```
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/fact_sale_1y_full')
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/" + table_name)
```

✓ - Command executed in 1 min 17 sec 153 ms by Marcos Yzquierdo on 9:43:30 AM, 8/21/24

13. After the fact data load is complete, move on to the 'Dimensions' cell to finalize the data loading data process for the notebook. The cell creates a function to read raw data from the **Files** section of the lakehouse for each of table names passed as a parameter. Next, it creates a list of dimension tables. Finally, the code executes a 'for loop' to iterate through the list of tables and calls the newly created function with each table name passed as a parameter to read data for that specific table and creates a delta table respectively.

Dimensions

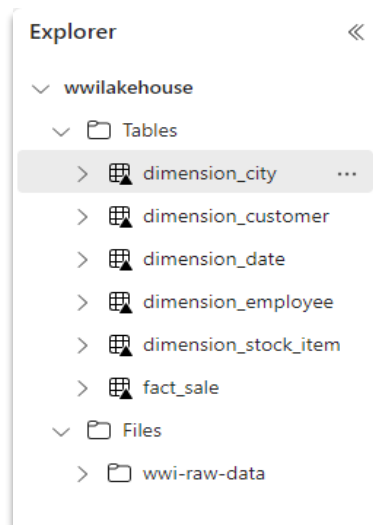
This cell creates a function to read raw data from the *Files* section of the lakehouse for the table name passed as a parameter. Next, it creates a list of dimension tables. Finally, it has a *for loop* to loop through the list of tables and call above function with each table name as parameter to read data for that specific table and create delta table.

```
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/full/' + table_name)
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_date',
11    'dimension_employee',
12    'dimension_stock_item'
13 ]
14
15 for table in full_tables:
16     loadFullDataFromSource(table)
```

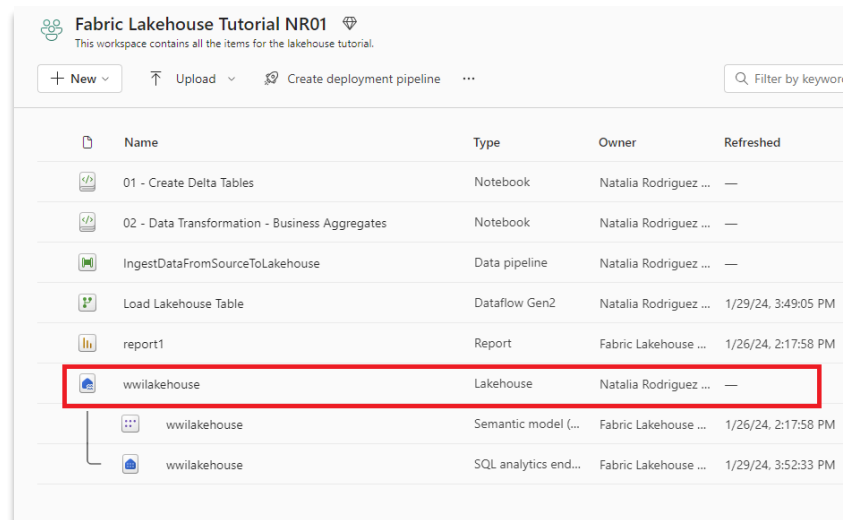
✓ - Command executed in 34 sec 445 ms by Marcos Yzquierdo on 9:47:47 AM, 8/21/24

PySpark (Python) ▾

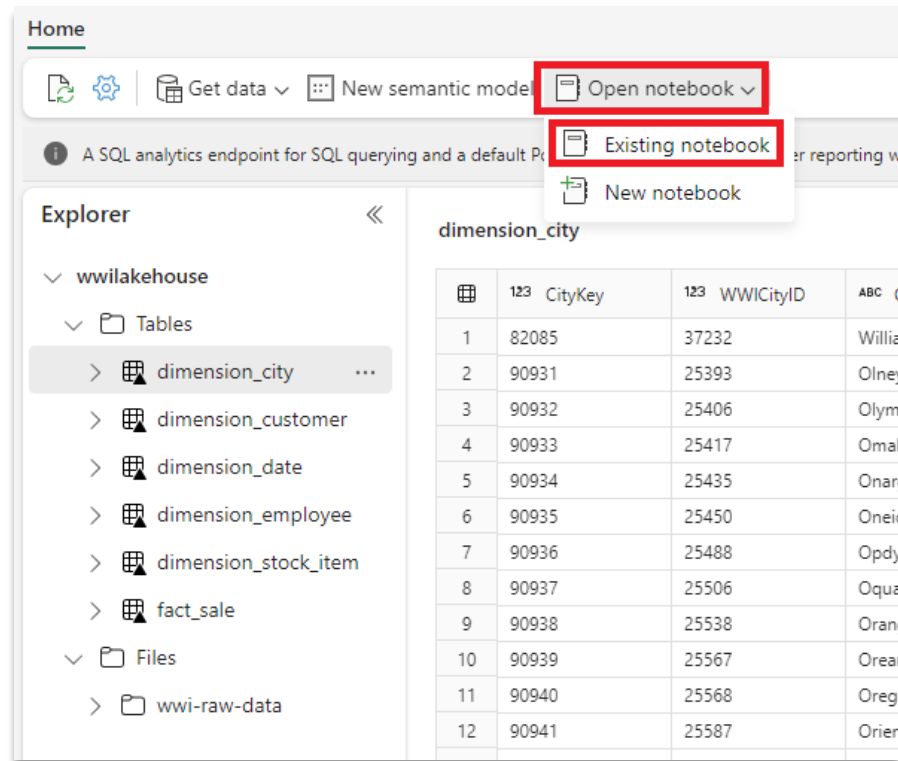
14. You can validate created tables by right clicking and selecting refresh on **wwilakehouse** lakehouse and you will notice these created tables.



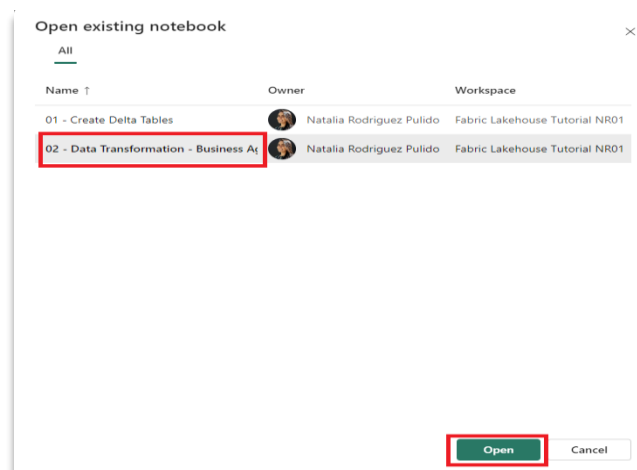
15. Please go the items view of the workspace again and click on **wwilakehouse** lakehouse to open it.



16. Now, open the second notebook. In the lakehouse view, click on **Open notebook** > **Existing notebook** on the ribbon at the top.



17. From the list of existing notebooks, select the **02 - Data Transformation - Business Ar** notebook to open it.



18. Once the notebook is opened, in the **Lakehouse explorer** you will notice the notebook is already linked to your opened lakehouse.

19. An organization might have a group of data engineers working with Scala/Python while there might be other groups of data engineers preferring to work with SQL (Spark SQL or T-SQL) and all these while 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.

You are going to learn two different approaches, as below, to transform and generate business aggregates, and as a developer 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 would be 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 would be preferable to someone with SQL background, transitioning to Spark.

20. **Approach #1** – Use PySpark to join and aggregates data for generating business aggregates. In the below code, you are creating three different Spark dataframes, each referencing an existing delta table. Then, you are joining these tables using the dataframes, doing group by to generate aggregation, renaming few of the columns and finally writing it as delta table in the **Tables** section of the lakehouse to persist with the data.

Approach #1 - sale_by_date_city

In this cell, you are creating three different Spark dataframes, each referencing an existing delta table.

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

In this cell, you are joining these tables using the dataframes created earlier, doing group by to generate aggregation, renaming few of the columns and finally writing it as delta table.

```
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.City", "city.StateProvince",
5 .groupby("date.Date", "date.CalendarMonthLabel", "date.Day", "date.ShortMonth", "date.CalendarYear", "city.City", "city.StateProvince",
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/aggregate_sale_by_date_city")
```

21. **Approach #2** – Use Spark SQL to join and aggregates data for generating business aggregates. In the below code, you are creating a temporary Spark view by joining 3 tables, doing group by to generate aggregation, renaming few of the columns. Finally, you are reading from the temporary Spark view and finally writing it as delta table in the Tables section of the lakehouse to persist with the data.

Approach #2 - sale_by_date_employee

In this cell, you are creating a temporary Spark view by joining 3 tables, doing group by to generate aggregation, renaming few of the columns.

```
1 %sql
2 CREATE OR REPLACE TEMPORARY VIEW sale_by_date_employee
3 AS
4 SELECT
5     DD.Date, DD.CalendarMonthLabel
6     , DD.Day, DD.ShortMonth, Month, CalendarYear Year
7     , DE.PreferredName, DE.Employee
8     , SUM(FS.TotalExcludingTax) SumOfTotalExcludingTax
9     , SUM(FS.TaxAmount) SumOfTaxAmount
10    , SUM(FS.TotalIncludingTax) SumOfTotalIncludingTax
11    , SUM(Profit) SumOfProfit
12 FROM wwilakehouse.fact_sale FS
13 INNER JOIN wwilakehouse.dimension_date DD ON FS.InvoiceDateKey = DD.Date
14 INNER JOIN wwilakehouse.dimension_employee DE ON FS.SalespersonKey = DE.EmployeeKey
15 GROUP BY DD.Date, DD.CalendarMonthLabel, DD.Day, DD.ShortMonth, DD.CalendarYear, DE.PreferredName, DE.Employee
16 ORDER BY DD.Date ASC, DE.PreferredName ASC, DE.Employee ASC
```

✓ 10 sec - Command executed in 10 sec 136 ms by Arshad Ali on 4:16:26 PM, 4/14/23

Spark SQL

> Spark jobs (3 of 3 succeeded)

No data available

+ Code + Markdown

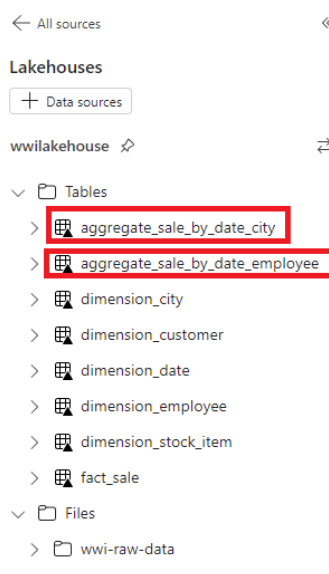
In this cell, you are reading from the temporary Spark view created in the previous cell and finally writing it as delta table in the *Tables* section of the lakehouse.

```
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/aggregate_sale_by_date_em")
```

✓ 19 sec - Command executed in 19 sec 74 ms by Arshad Ali on 4:16:46 PM, 4/14/23

PySpark (Python)

22. You can validate created tables by right clicking and selecting refresh on **wwilakehouse** lakehouse and you will notice these aggregate tables appear.



If you notice both these above approaches (1 and 2) produce a similar outcome, however based on developer background and preference, one approach can be picked up over the other without any need for the developer to learn a new technology and compromising on the performance.

Further, you would have noticed that you are writing data as delta lake files and then the automatic table discovery and registration feature of Fabric picks it up and registers it in the metastore. That means, you don't need to explicitly call CREATE TABLE statement to create tables to use with SQL.

Building a report

Power BI is natively integrated in the whole Fabric experience and this native integration brings unique mode of accessing the data, called DirectLake, from the lakehouse to provide the most performant query and reporting experience. DirectLake mode is a groundbreaking new engine capability to analyze very large datasets in Power BI. The

technology is based on the idea of loading parquet-formatted files directly from a data lake without having to query a data warehouse or lakehouse endpoint and without having to import or duplicate data into a Power BI dataset. DirectLake is a fast path to load the data from the data lake straight into the Power BI engine, ready for analysis.

In traditional DirectQuery mode, the Power BI engine queries the data directly from the data source every time it is queried and hence query performance depends on the speed data can be retrieved from the data source. It avoids having to copy the data i.e., any changes at the source are immediately reflected in the query results while in the import mode, on the other hand, performance is much better because the data is readily available in memory without having to query the data source each time, but the Power BI engine must first copy the data into the dataset at refresh time. Any changes at the source are only picked up during the next data refresh.

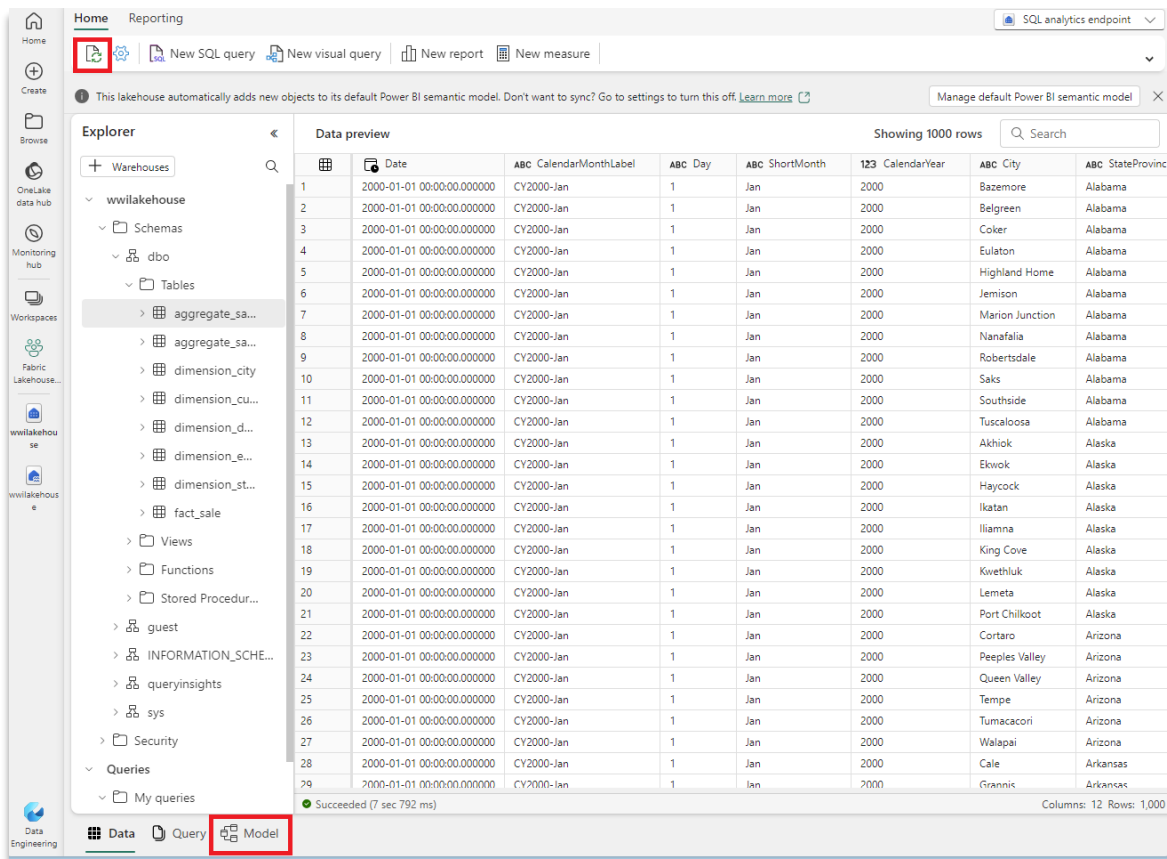
DirectLake mode now eliminates this import requirement by loading the data files directly into memory. Because there is no explicit import process, it is possible to pick up any changes at the source as they occur, thus combining the advantages of DirectQuery and import mode while avoiding their disadvantages. DirectLake mode is therefore the ideal choice for analyzing very large datasets as well as datasets with frequent updates at the source.

1. Please go the **wwilakehouse** lakehouse, click **SQL analytics endpoint** under **Lakehouse** on top right of the screen to open warehouse mode of the selected lakehouse.

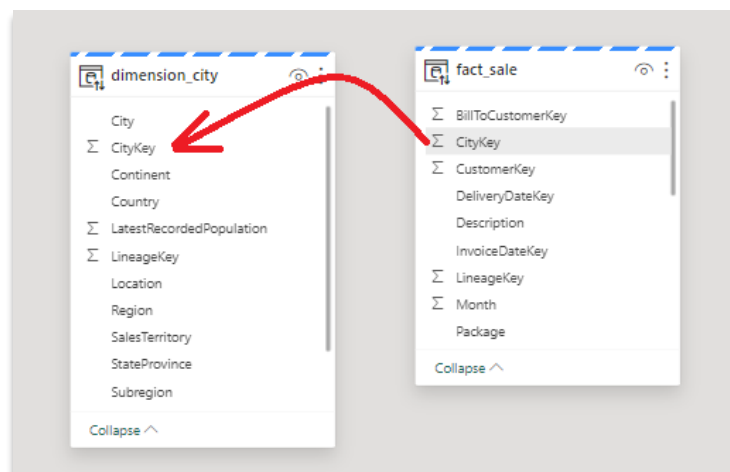
| | Date | CalendarMonth | Day | ShortMonth | CalendarYear | City | StateProvince | SalesTerritory | SumOfTotalExclu... | SumOfTotalInclu... | SumOfTotalInclu... |
|----|----------------------|---------------|-----|------------|--------------|-----------------|---------------|----------------|--------------------|--------------------|--------------------|
| 1 | 1/1/2000 12:00:00... | CY2000-Jan | 1 | Jan | 2000 | Bazemore | Alabama | Southeast | 82944.00 | 12441.60 | 95385.60 |
| 2 | 1/1/2000 12:00:00... | CY2000-Jan | 1 | Jan | 2000 | Belgreen | Alabama | Southeast | 601180.00 | 90192.00 | 691472.00 |
| 3 | 1/1/2000 12:00:00... | CY2000-Jan | 1 | Jan | 2000 | Coker | Alabama | Southeast | 257984.00 | 38697.60 | 296681.60 |
| 4 | 1/1/2000 12:00:00... | CY2000-Jan | 1 | Jan | 2000 | Eulaton | Alabama | Southeast | 507584.00 | 76137.60 | 583721.60 |
| 5 | 1/1/2000 12:00:00... | CY2000-Jan | 1 | Jan | 2000 | Highland Home | Alabama | Southeast | 18480.00 | 2772.00 | 21252.00 |
| 6 | 1/1/2000 12:00:00... | CY2000-Jan | 1 | Jan | 2000 | Jemison | Alabama | Southeast | 278816.00 | 41822.72 | 320638.72 |
| 7 | 1/1/2000 12:00:00... | CY2000-Jan | 1 | Jan | 2000 | Marion Junction | Alabama | Southeast | 212992.00 | 31948.80 | 244940.80 |
| 8 | 1/1/2000 12:00:00... | CY2000-Jan | 1 | Jan | 2000 | Nanafalia | Alabama | Southeast | 312256.00 | 46638.40 | 359094.40 |
| 9 | 1/1/2000 12:00:00... | CY2000-Jan | 1 | Jan | 2000 | Robertsdale | Alabama | Southeast | 799222.40 | 119883.36 | 919105.76 |
| 10 | 1/1/2000 12:00:00... | CY2000-Jan | 1 | Jan | 2000 | Saks | Alabama | Southeast | 469312.00 | 70396.64 | 539708.64 |
| 11 | 1/1/2000 12:00:00... | CY2000-Jan | 1 | Jan | 2000 | Southside | Alabama | Southeast | 20160.00 | 3024.00 | 23184.00 |
| 12 | 1/1/2000 12:00:00... | CY2000-Jan | 1 | Jan | 2000 | Tuscaloosa | Alabama | Southwest | 36000.00 | 5400.00 | 41400.00 |
| 13 | 1/1/2000 12:00:00... | CY2000-Jan | 1 | Jan | 2000 | Aktchik | Alaska | Far West | 601177.60 | 90176.64 | 691354.24 |
| 14 | 1/1/2000 12:00:00... | CY2000-Jan | 1 | Jan | 2000 | Elvick | Alaska | Far West | 106000.00 | 15900.00 | 121900.00 |
| 15 | 1/1/2000 12:00:00... | CY2000-Jan | 1 | Jan | 2000 | Haycock | Alaska | Far West | 284544.00 | 42681.60 | 327225.60 |
| 16 | 1/1/2000 12:00:00... | CY2000-Jan | 1 | Jan | 2000 | Ikatan | Alaska | Far West | 114400.00 | 17160.32 | 131560.32 |

Note: If necessary, hover over a table and to the right are three bullets for 'more options'. Select 'more options' and choose 'add to default semantic model'.

- Once you are in warehouse mode you should be able to see all the tables you created. If you don't see them yet, please click on the **Refresh** icon at the top. Next, click on **Model** tab at the bottom to open the default Power BI dataset.



- 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. Check the "Assume referential integrity" and click on **Confirm** to establish the relationship. **Note** – When defining relationships, please make sure you have many to one relationship from the fact to the dimension and not vice versa.

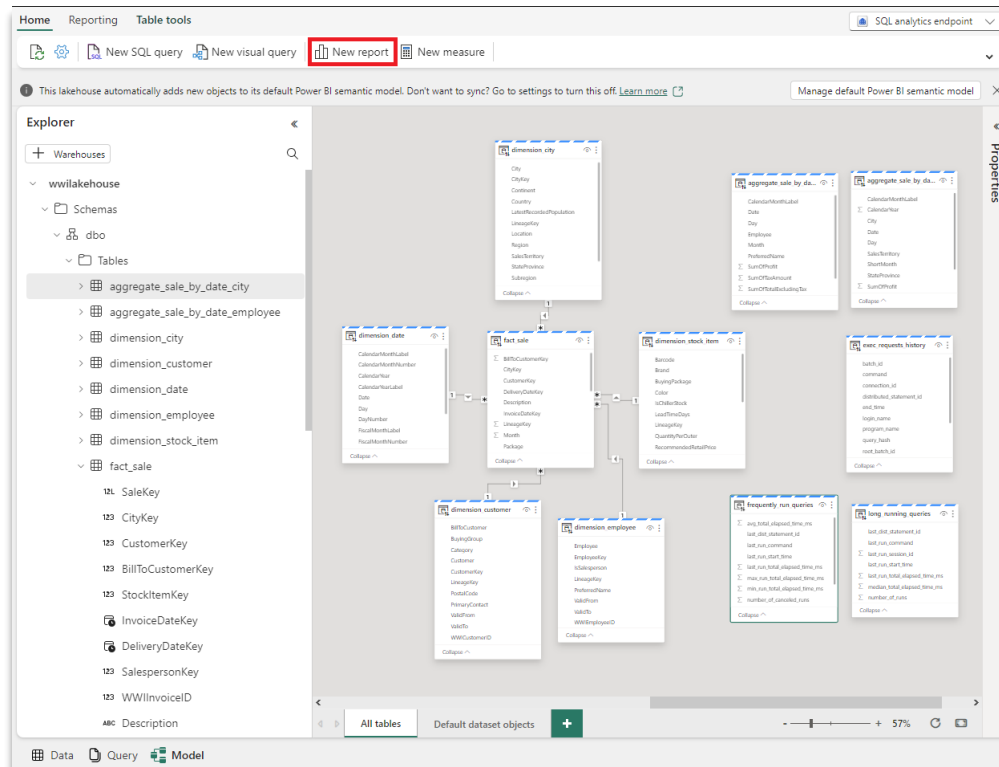


4. On the **New Relationship** settings:

- Table 1 will be populated with fact_sale and the column of CityKey.
- Table 2 will be 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** checked.
- Check the box next to **Assume referential integrity**.
- Select **Confirm**.

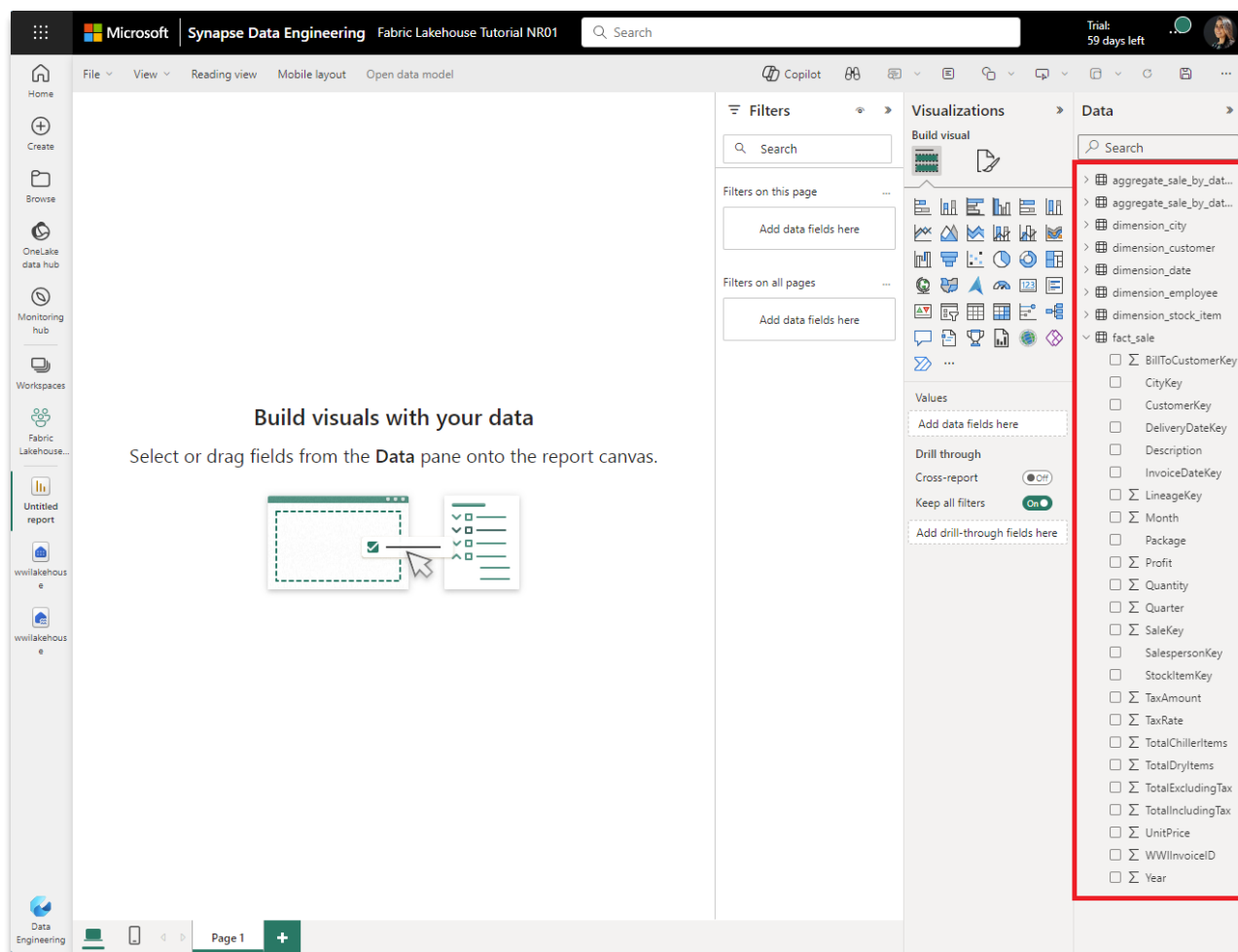
h. Similarly, you need to add these relationships as well:

- 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)

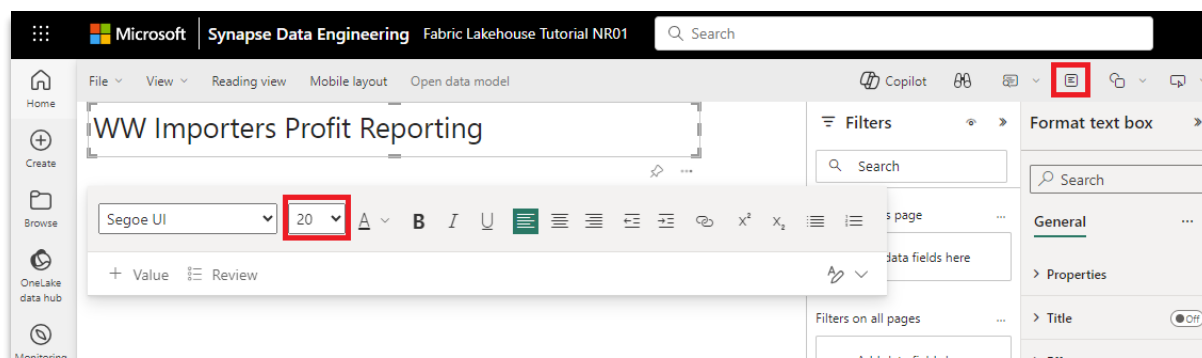


After adding these above relationships, your data model is ready, as above, for reporting. Click on **New report** to start creating reports/dashboards in Power BI.

5. 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.

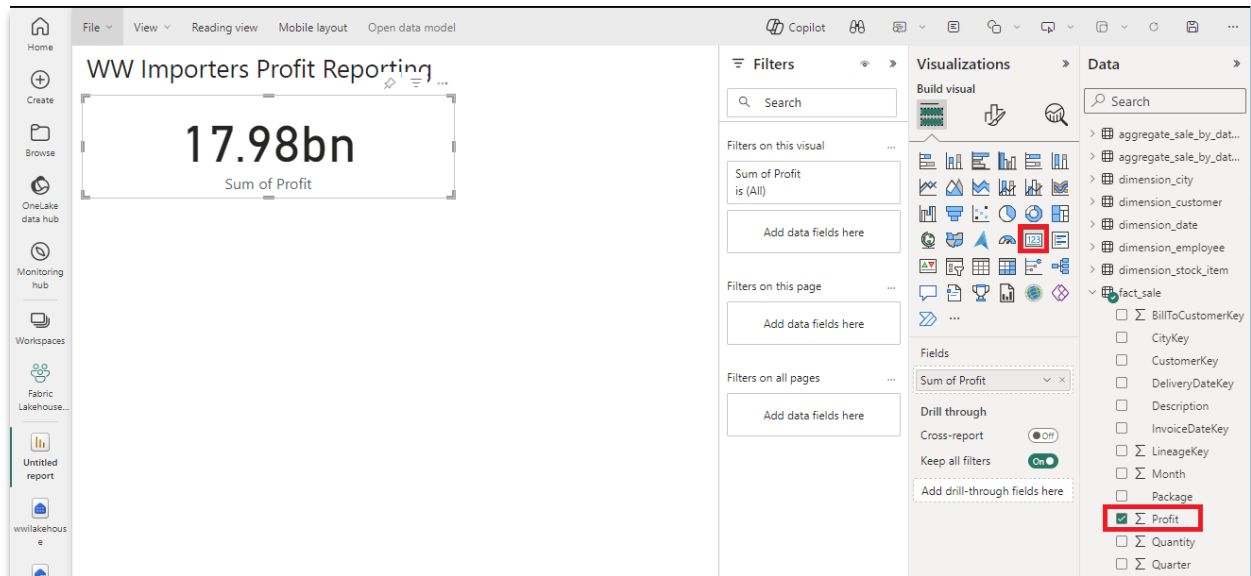


6. Add a title:
 - a. In the Ribbon, click the Text Box icon
 - b. Type in **WW Importers Profit Reporting**
 - c. Highlight the text and increase size to 20 and place in the upper left of the report page



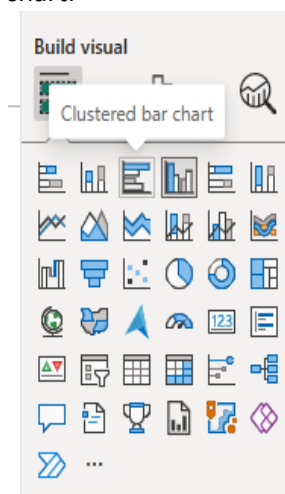
7. Add a Card

- On the **Data** pane, expand **fact_sales** and check the box next to **Profit**. This will create a column chart and add the field to the Y-axis.
- With the bar chart selected, click the Card visual in the visualization pane. This will convert the visual to a card.
- Place the card under the title

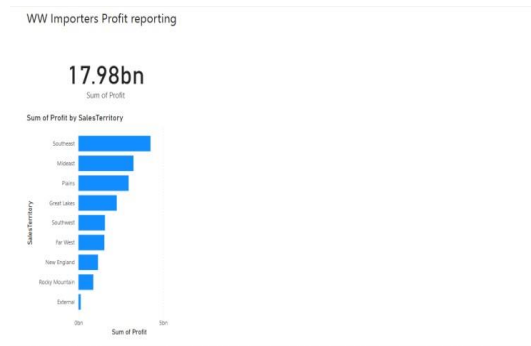


8. Add a Bar chart

- On the **Data** pane, expand **fact_sales** and check the box next to **Profit**. This will create a column chart and add the field to the Y-axis.
- On the **Data** pane, expand **dimension_city** and check the box for **SalesTerritory**. This will add the field to the X-axis.
- With the bar chart selected, click on the Clustered Bar Chart visual in the visualization pane. This will convert the column chart into a bar chart.



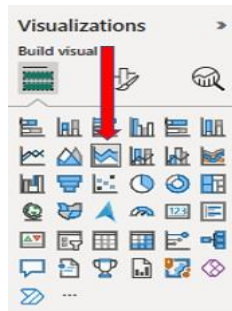
- d. Resize the Bar chart to fill in the area under the title and Card



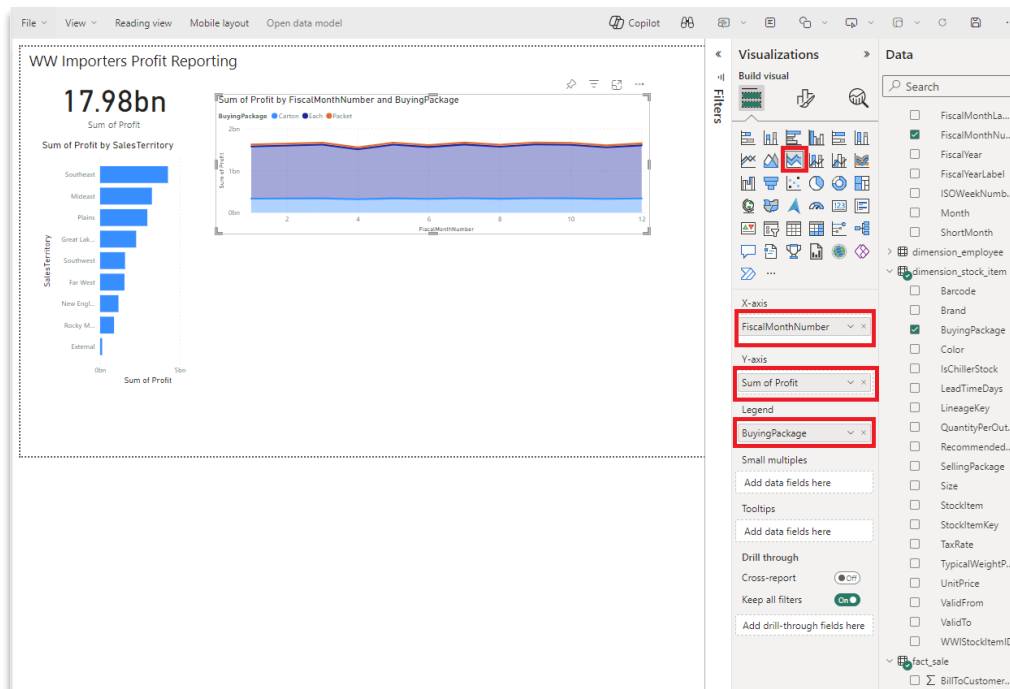
9. Click anywhere on the blank canvas (or press the Esc key) so the bar chart visual is no longer selected.

10. Build a stacked area chart visual:

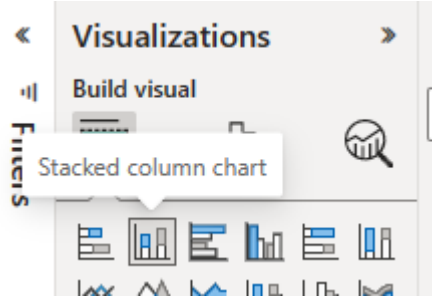
- a. On the **Visualizations** pane, select the **Stacked area chart** visual.



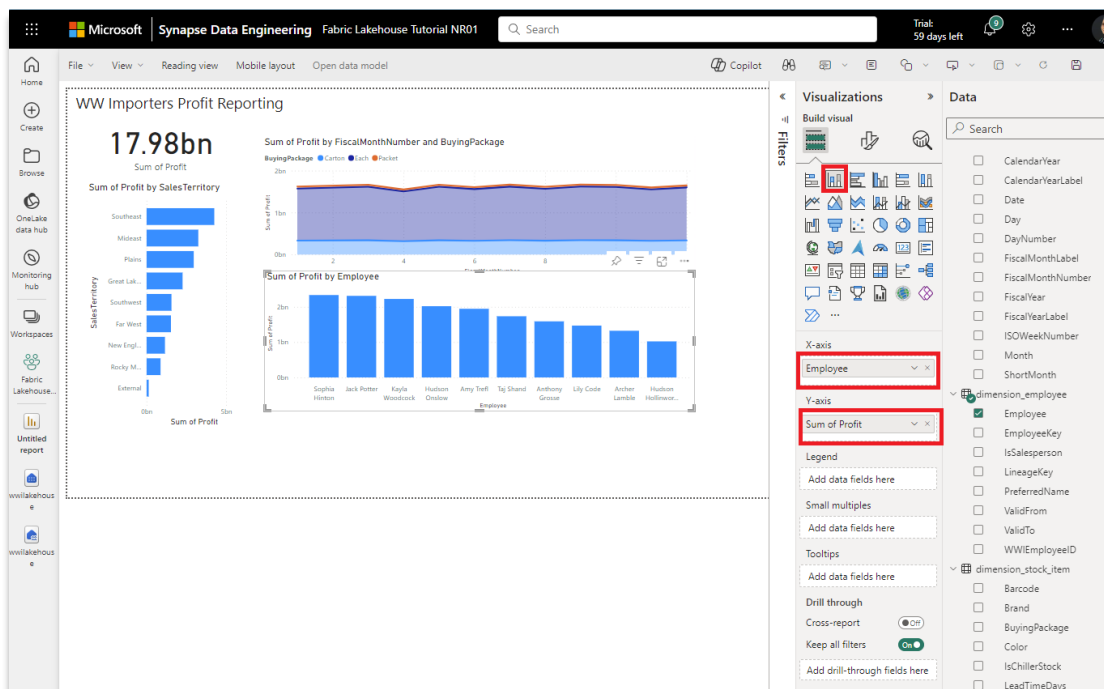
- b. Reposition and resize the stacked area chart to the right of the card and bar chart visuals created in the previous steps.
- c. On the **Data** pane, expand **fact_sales** and check the box next to **Profit**. Expand **dimension_date** and check the box next to **FiscalMonthNumber**. This will create 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 will add a line for each of the Buying Packages.



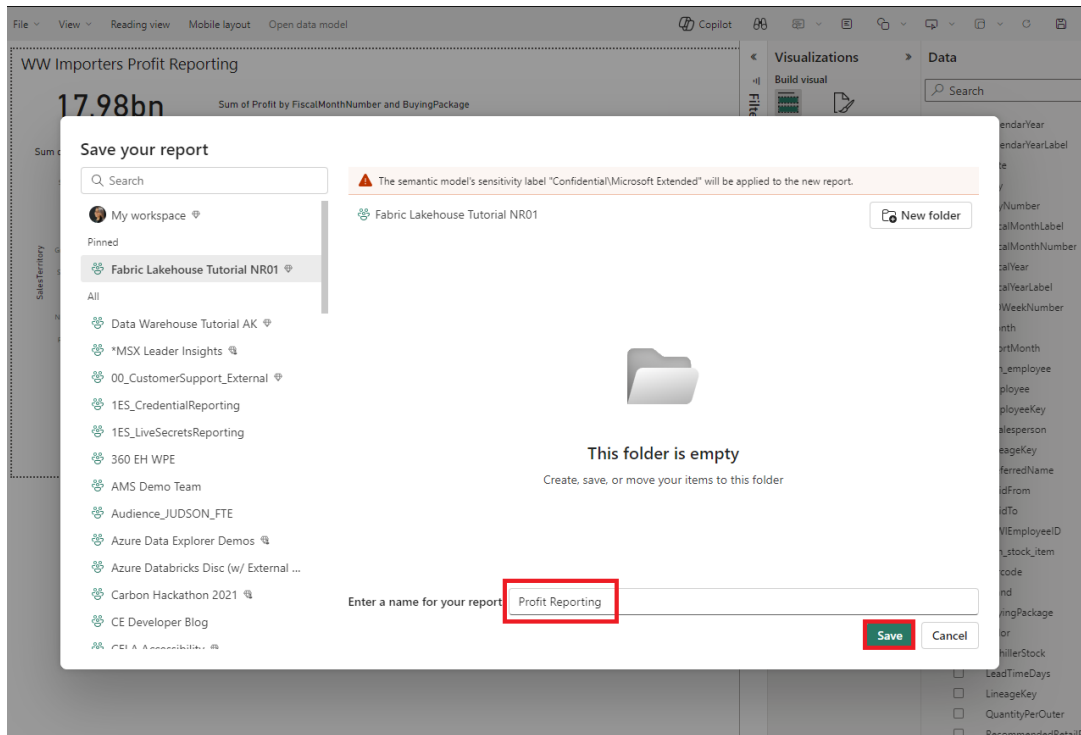
11. Click anywhere on the blank canvas (or press the Esc key) so the stacked area chart visual is no longer selected.
12. Build a column chart:
 - a. On the **Visualizations** pane, select the **Stacked column chart** visual.



- b. On the **Data** pane, expand **fact_sales** and check the box next to **Profit**. This will add the field to the Y-axis.
- c. On the **Data** pane, expand **dimension_employee** and check the box next to **Employee**. This will add the field to the X-axis.



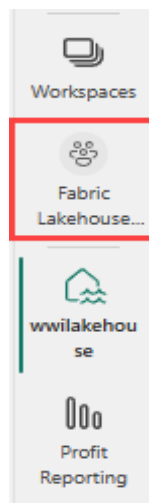
13. Click anywhere on the blank canvas (or press the Esc key) so the LINE chart visual is no longer selected.
14. From the ribbon, select **File > Save**.
15. Enter the name of your report as **Profit Reporting**.
16. Select **Save**



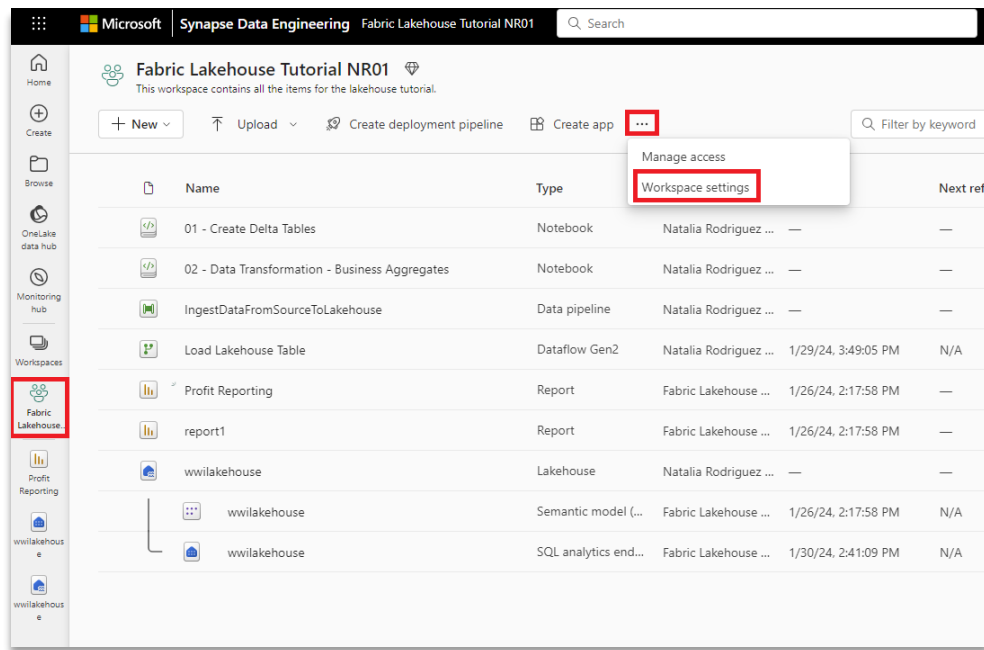
Module 4: Clean up resources

You can delete individual reports, pipelines, warehouses, and other items or remove the entire workspace.

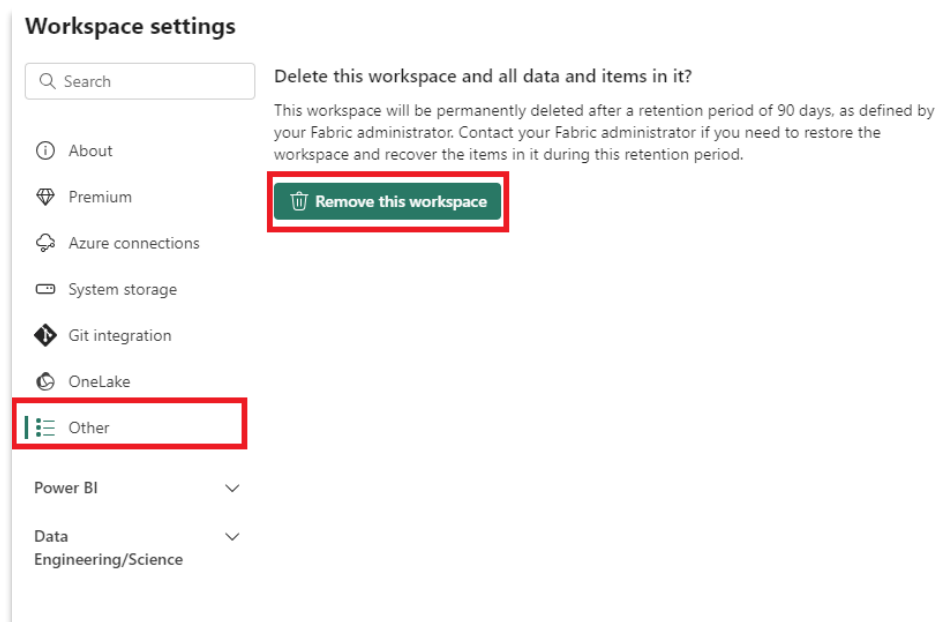
1. Select **Fabric Lakehouse Tutorial <suffix you added to make it unique>** in the left-hand navigation menu to return to the workspace item view.



- Below the workspace name and description at the top of the workspace header, select **Workspace settings**.



- Select **Other > Delete this workspace**.



- Select **Delete** on the warning.