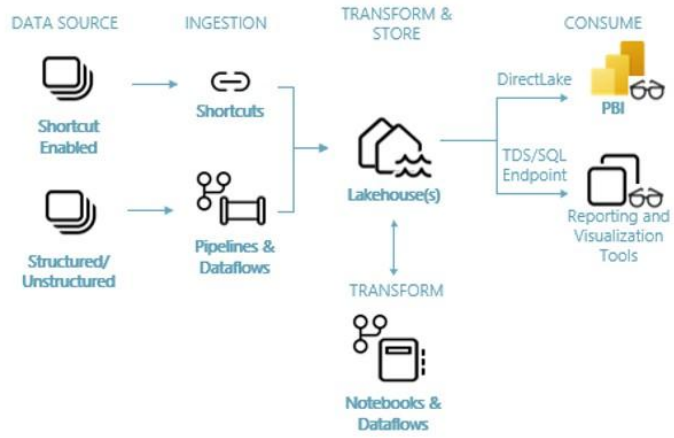
Lakehouse Tutorial

Lakehouse

Published: Aug 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

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 duplicity 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 duplicity, and drastically reduce total cost of ownership.

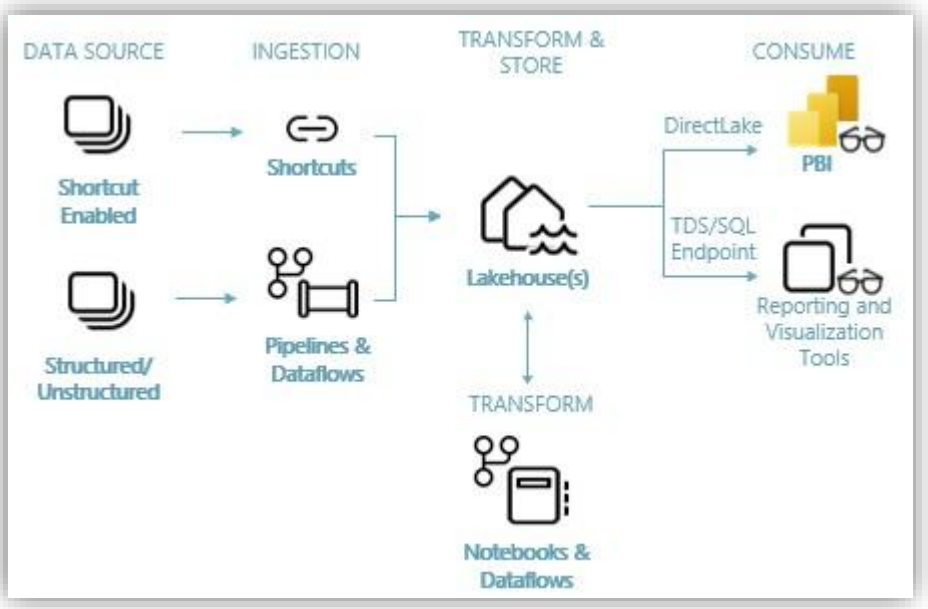
With the flexibility Fabric provides, you can implement either lakehouse or data warehouse architectures or combine these two together to get the best of both the worlds in simple implementation. In this tutorial, you are going to take an example of retail organization and build its lakehouse from start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

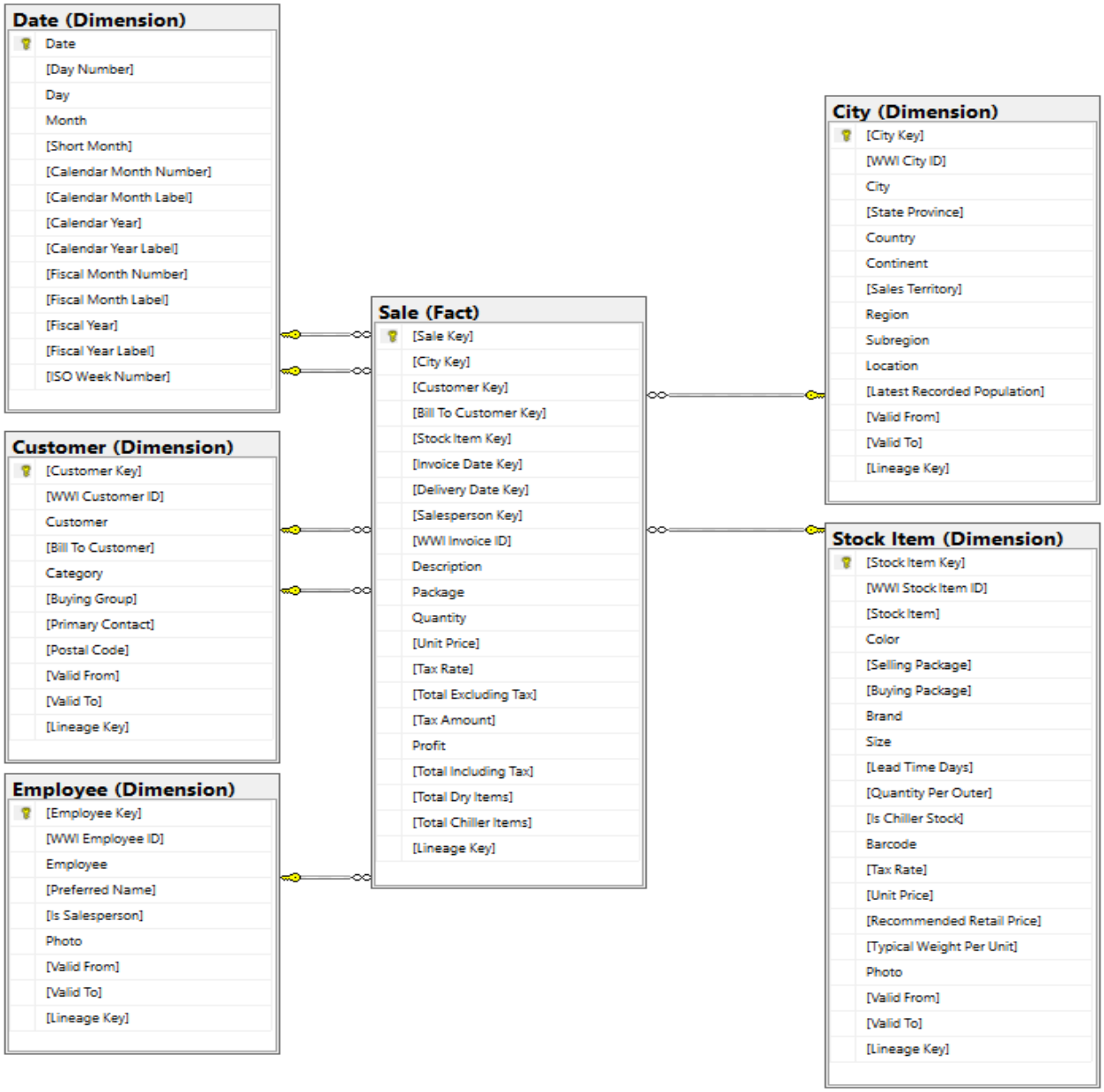
Wide World Importers (WWI) is a wholesale novelty goods importer and distributor operating from the San Francisco Bay area. As a wholesaler, WWI's customers are mostly companies who resell to individuals. WWI sells to retail customers across the United States including specialty stores, supermarkets,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

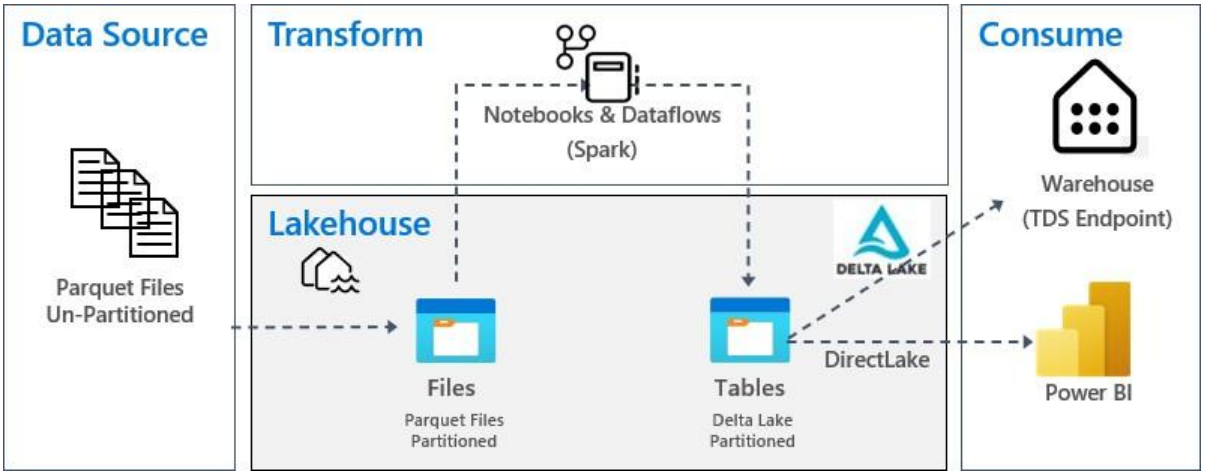
While the WWI dimensional model contains multiple fact tables, for simplicity in explanation we will focus on the Sale Fact table and its related dimensions only, as below, to demonstrate this end-to-end lakehouse scenario:



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.

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

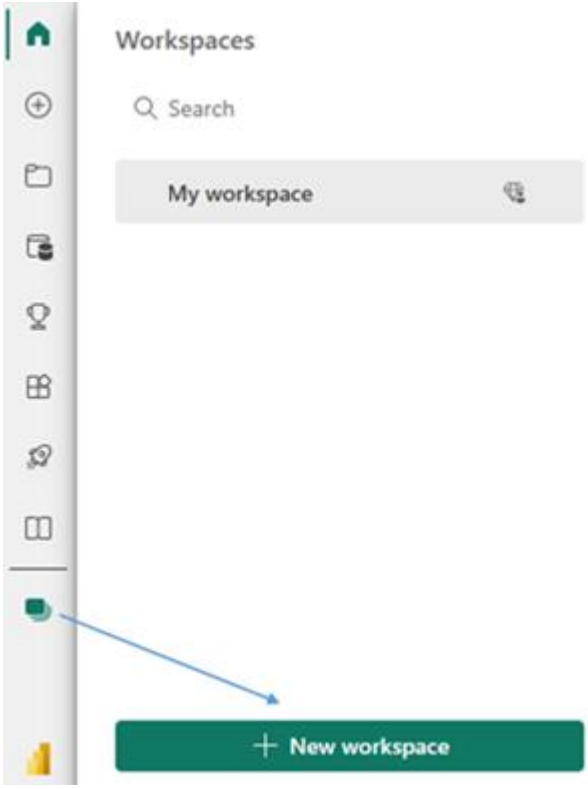
Before you can begin building the lakehouse, you will need to create a workspace where you will build out the remainder of the tutorial.

Imp: You can skip this Module if the workspace has already been setup for you as part of lab setup

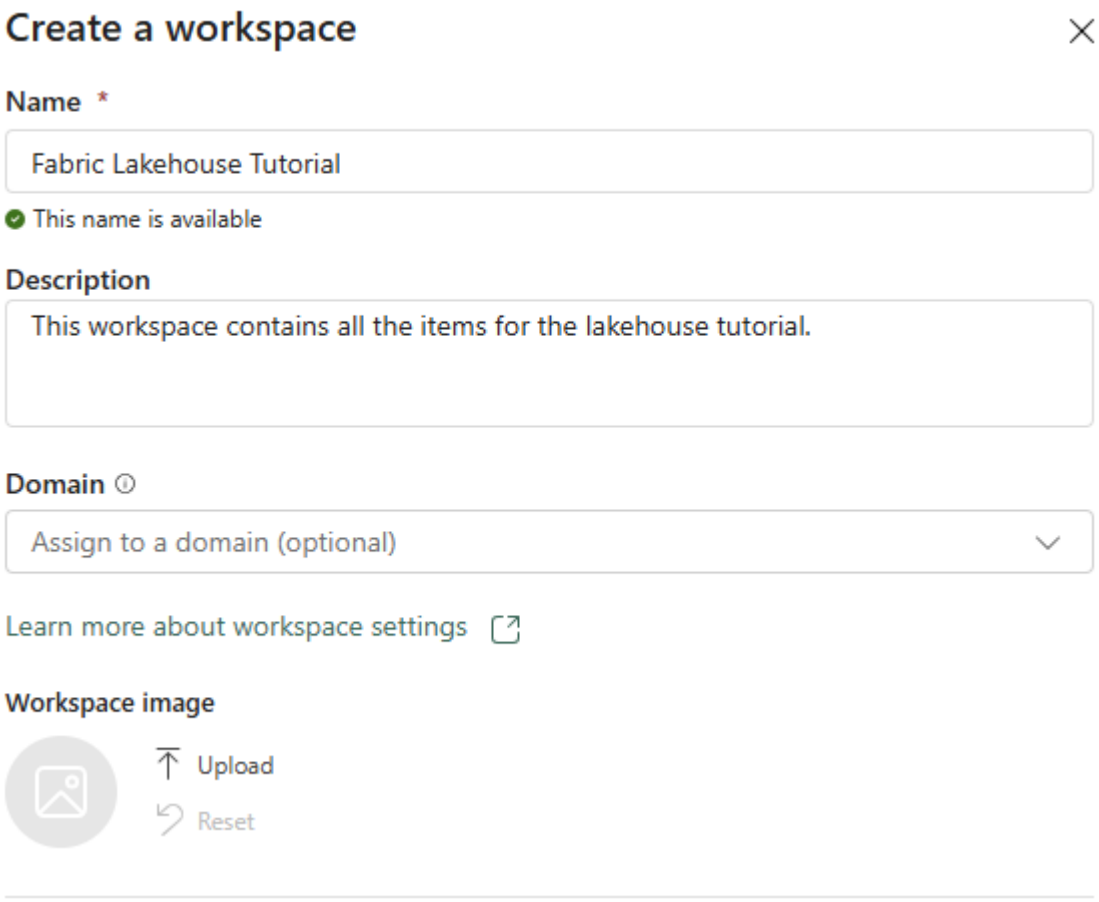
Create a Fabric workspace

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.

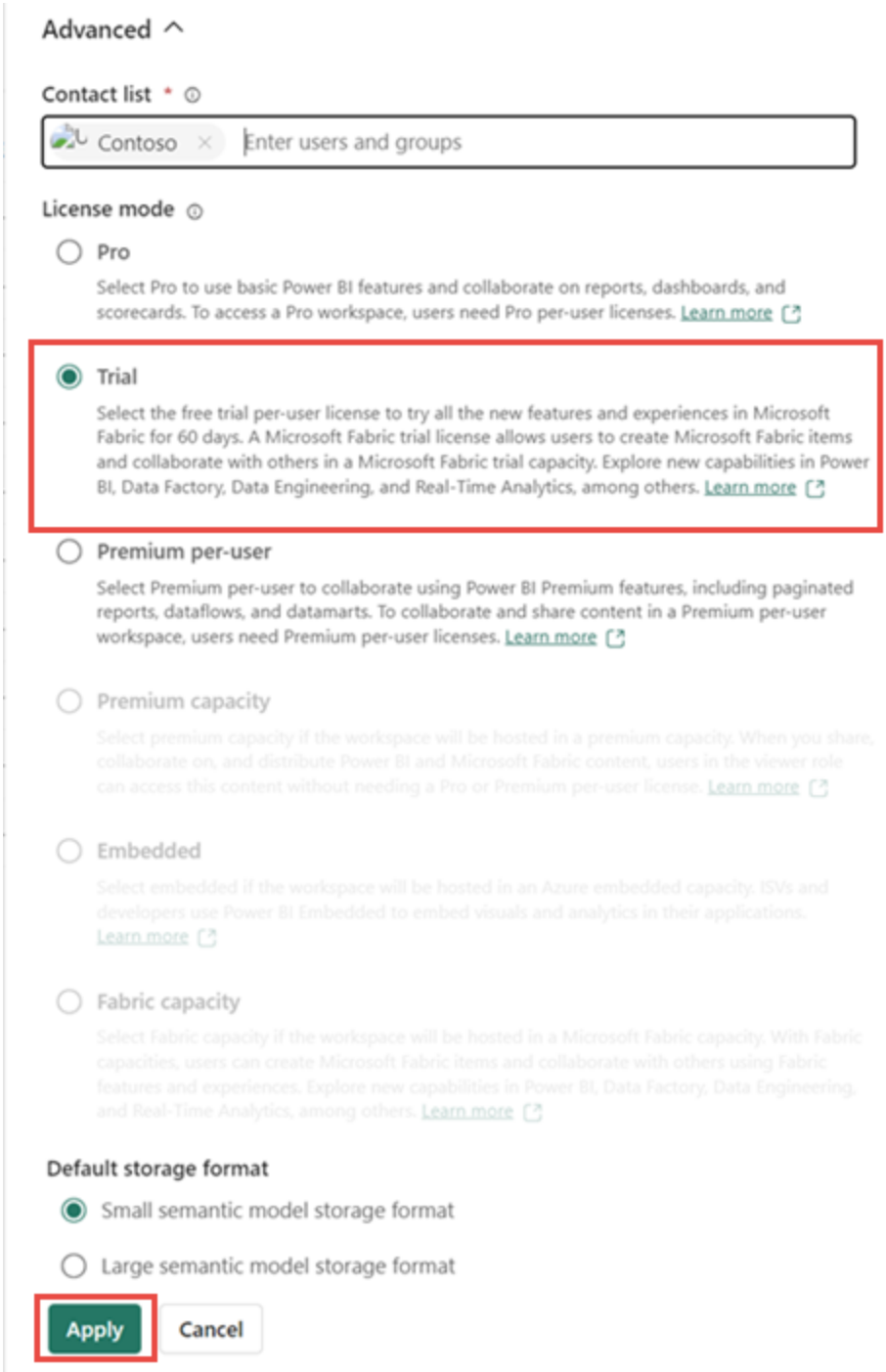
1. Sign in to Power BI.
2. Select **Workspaces > Create a Workspace**.



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.



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..



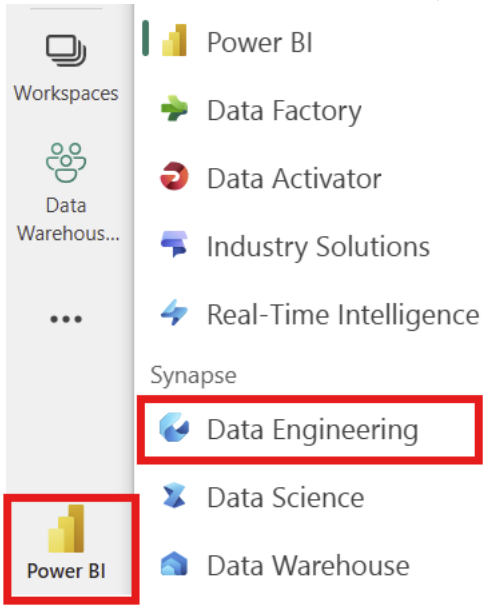
4. Select **Apply.** The workspace will be created and opened.

Module 2: Build your first Lakehouse in Fabric

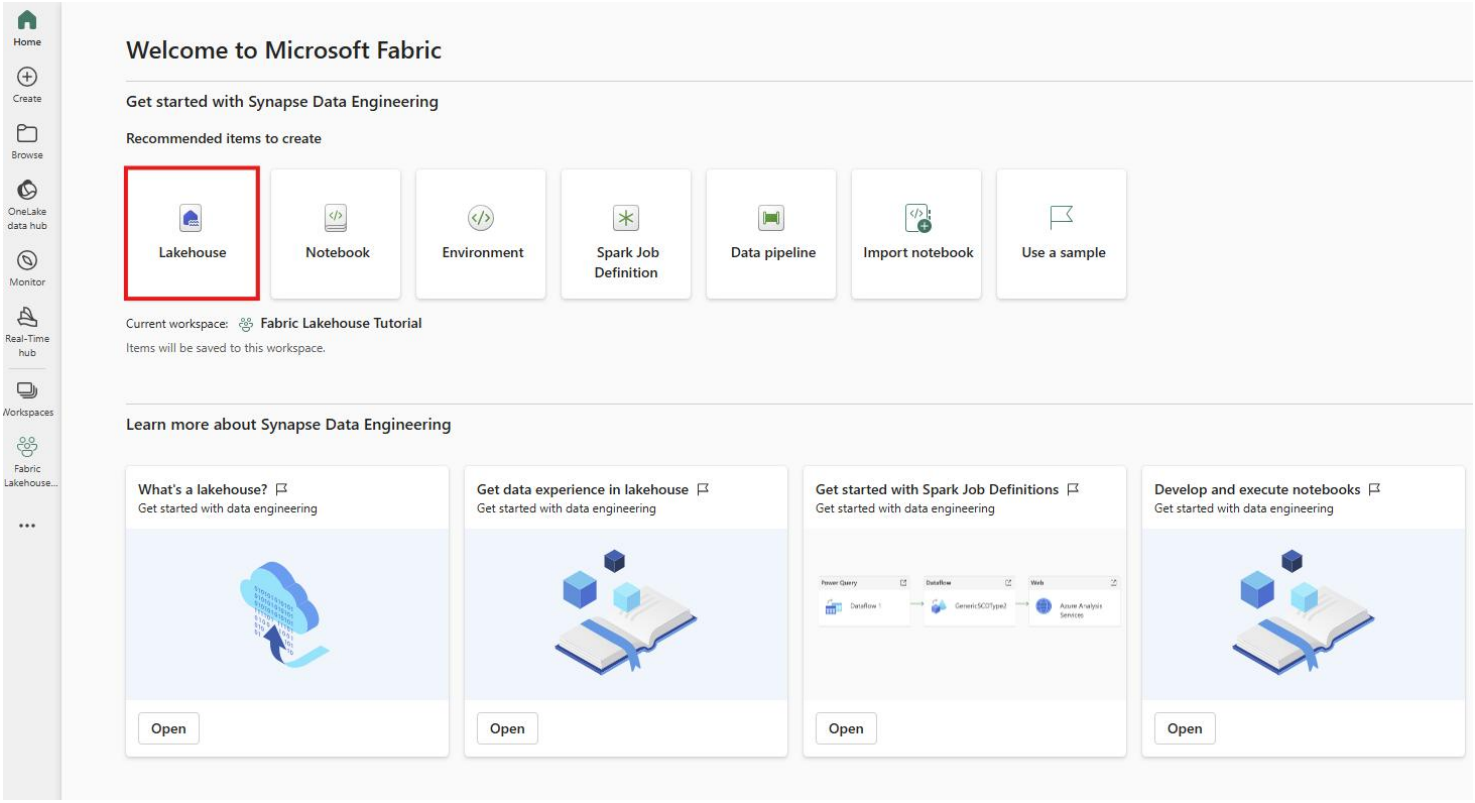
The intent of this module is to quickly build end to end journey of building a lakehouse, ingesting data for a table, applying transformation whenever required and then using ingested data into the lakehouse delta table for creating reports.

Create a lakehouse

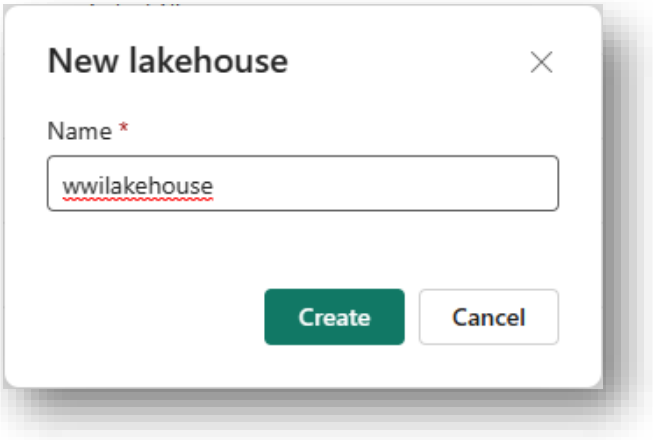
1. In the Power BI service select **Workspaces** in the left-hand menu.
2. Search for your workspace(**Fabric Lakehouse Tutorial**) by typing in the search textbox at the top and click on your workspace to open it.
3. From the workload switcher located at the bottom left of the screen, select **Data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



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



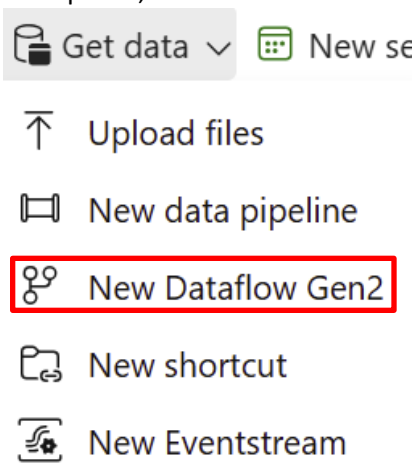
5. Enter **wwilakehouse** in the **Name** box. Please do not check **Lakehouse schemas** checkbox if its enabled for you ☐ Lakehouse schemas (Public Preview) ⓘ



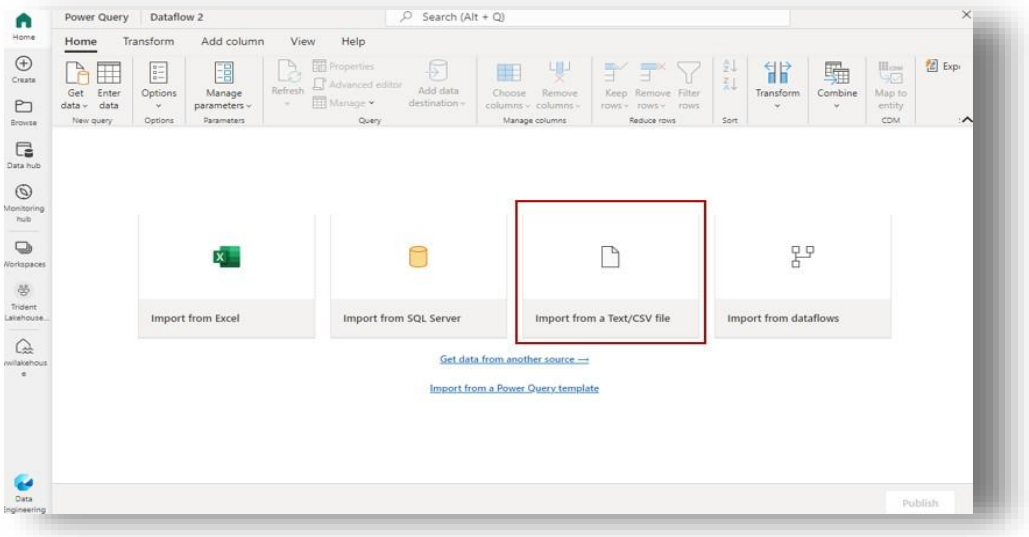
6. Click **Create**. The new lakehouse will be created and automatically opened.

Data Ingestion

1. Download the 'dimension_customer.csv' file found in **Data** folder. The File will provided you as part of the lab content. Please check with instructor.
2. In the lakehouse Click on Get data menu option , Click on **New Dataflow Gen2**.

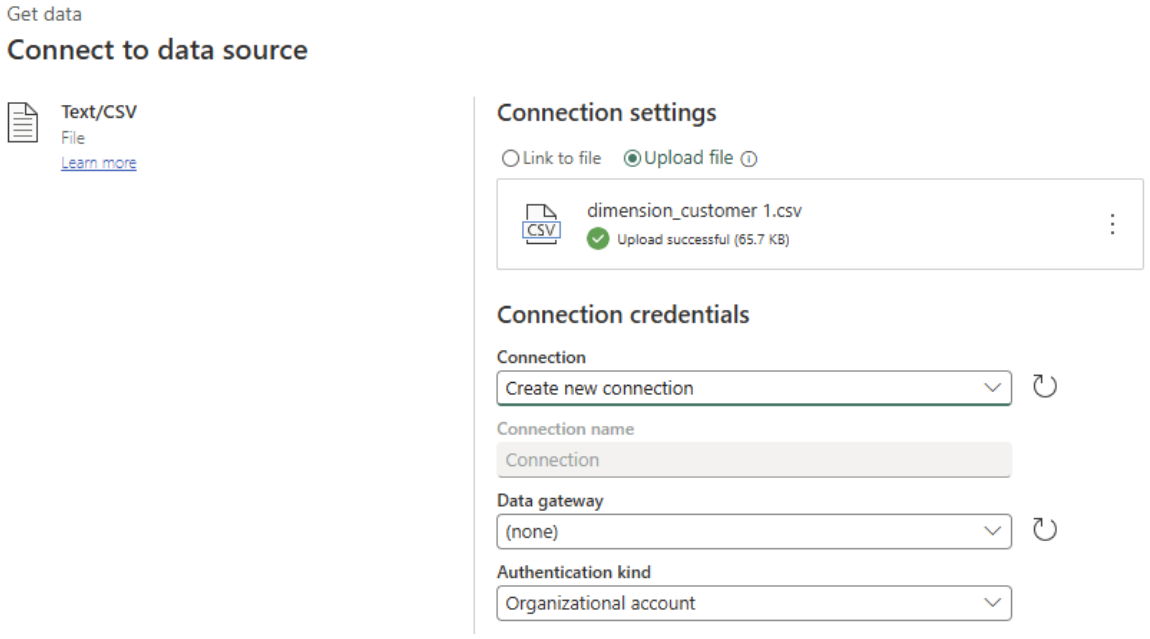


3. On the new dataflow page, click on **Import from a Text/CSV file**.

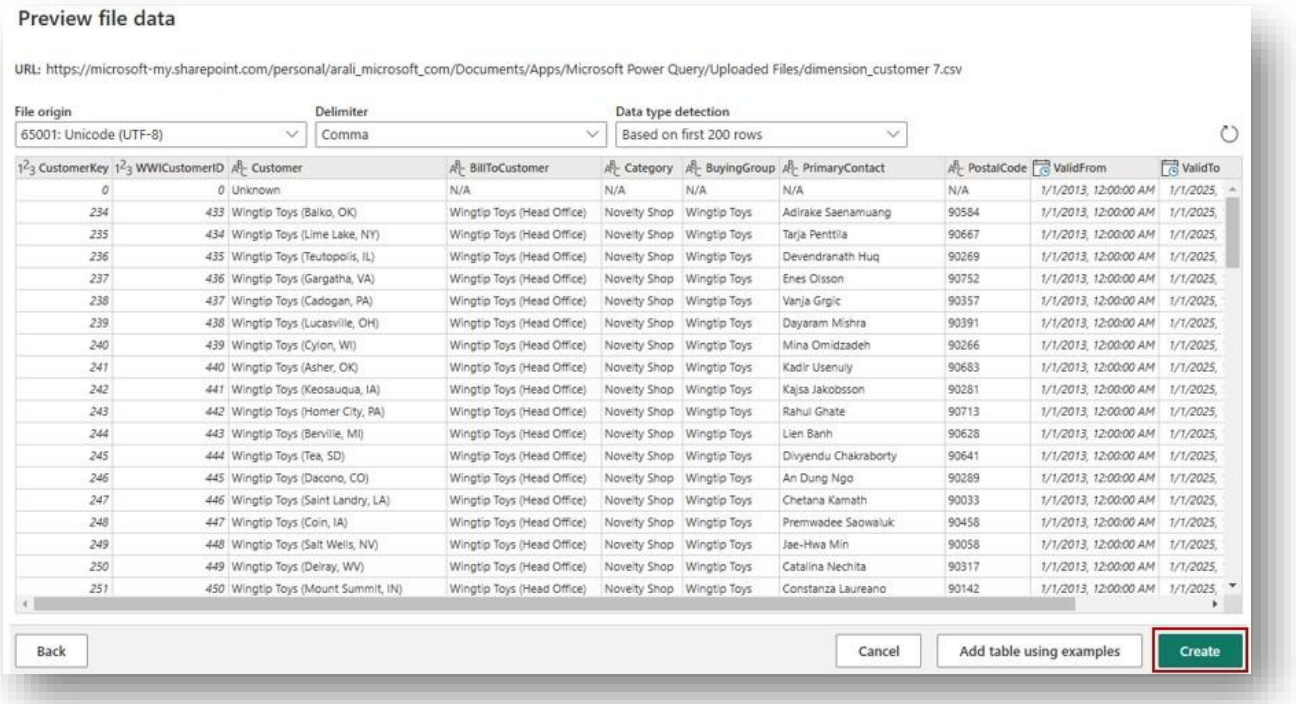


⚠ ***If you encounter problems with file uploads, like lacking a SharePoint Online license or uninitialized OneDrive for Business, skip 4&5 and follow steps 6-9. If upload options is available, proceed to Step 4 & 5 and ignore steps 6-9***

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.

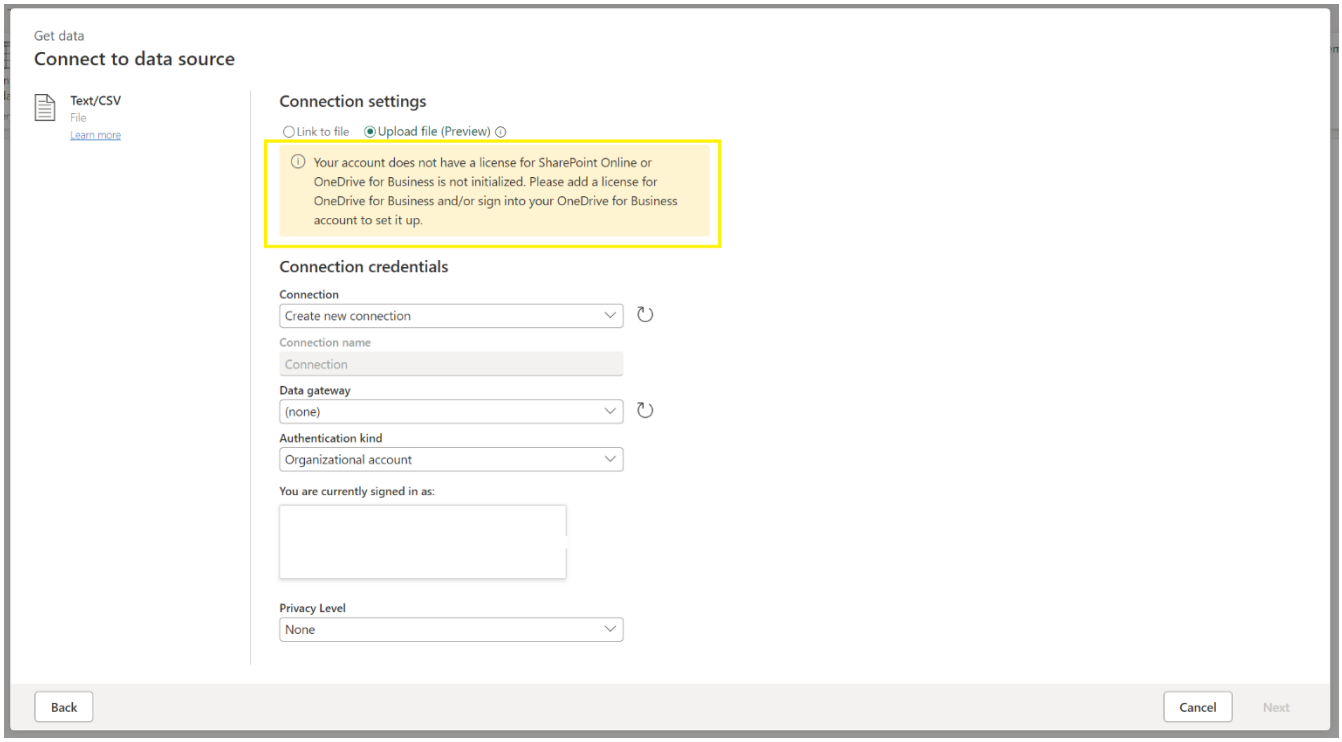


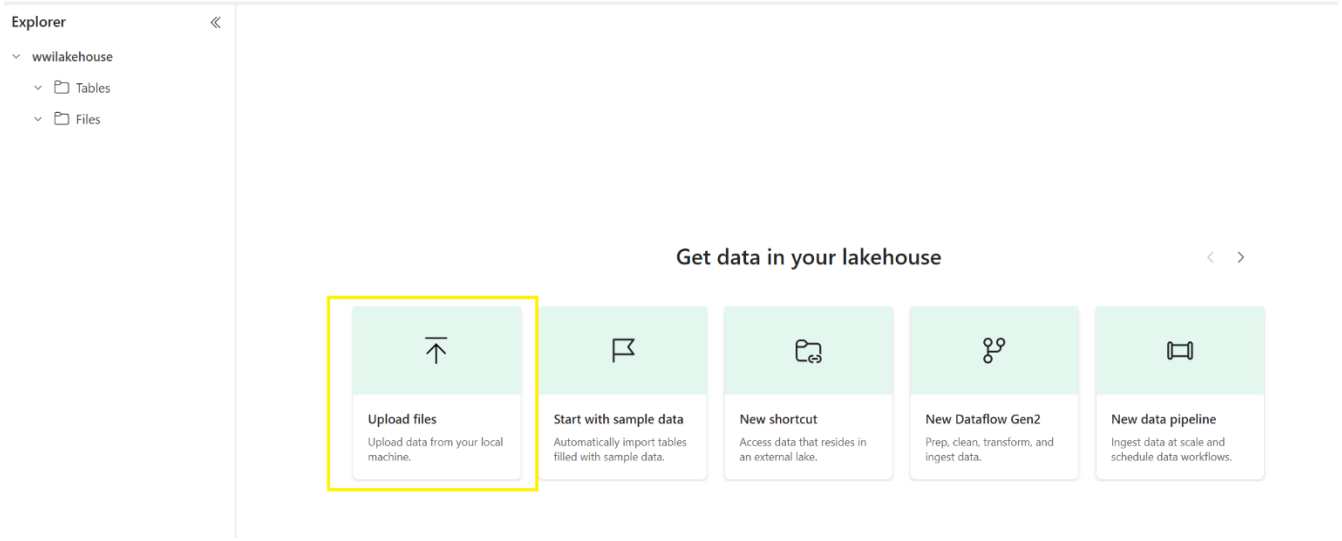
5. Clicking **Next** Once the file is uploaded, click **Next** on the previous screen will open the **Preview file data** page, click on **Create** to proceed and return back to dataflow canvas.

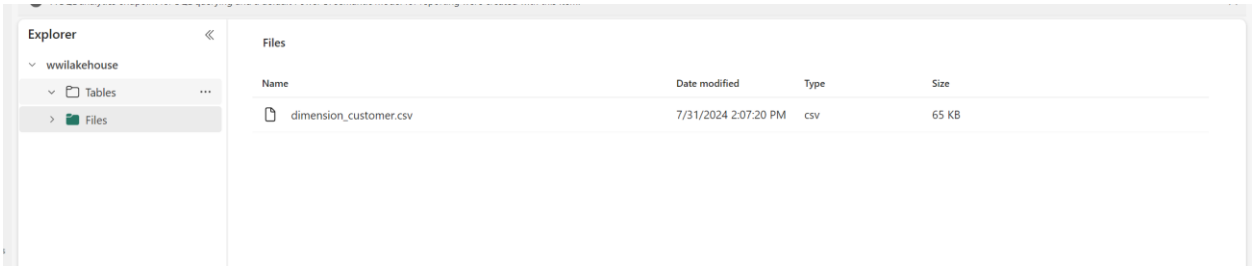


Ignore 6-9 step if you that the upload file option working

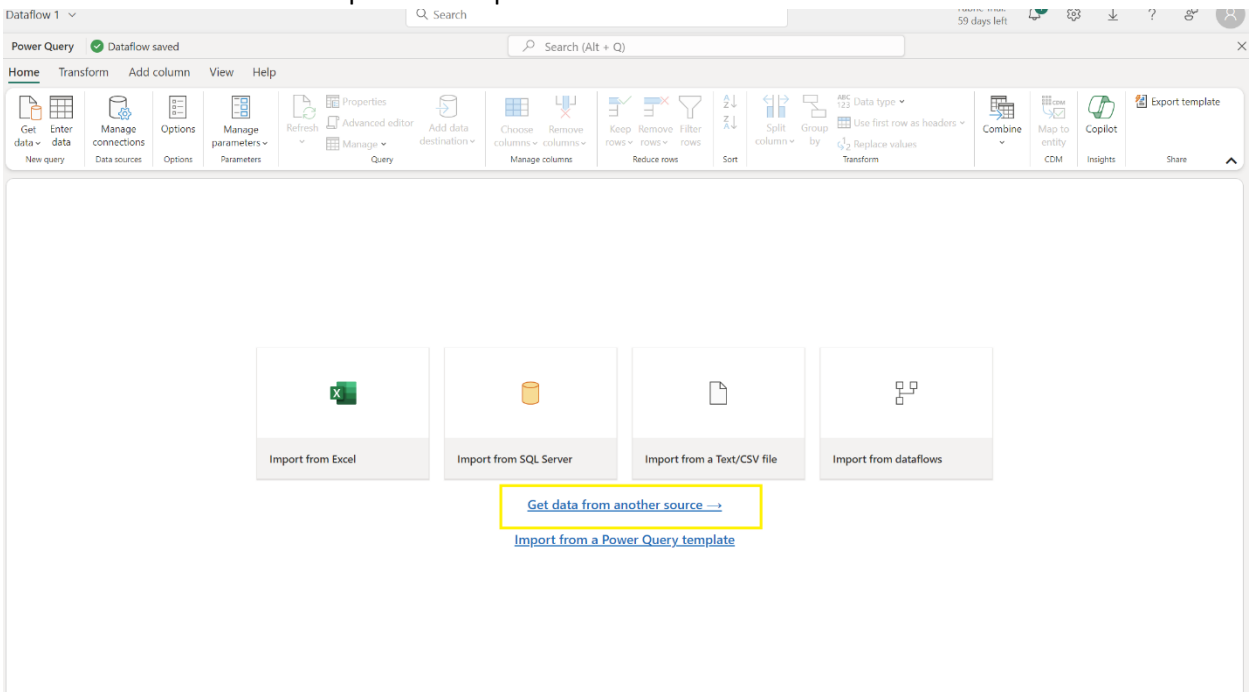
6. Cancel the “Connect to data source” window without proceeding, Close the dataflow, navigate to the lakehouse within your workspace and choose the “upload file” feature to upload the data file you acquired in the first step of this module.

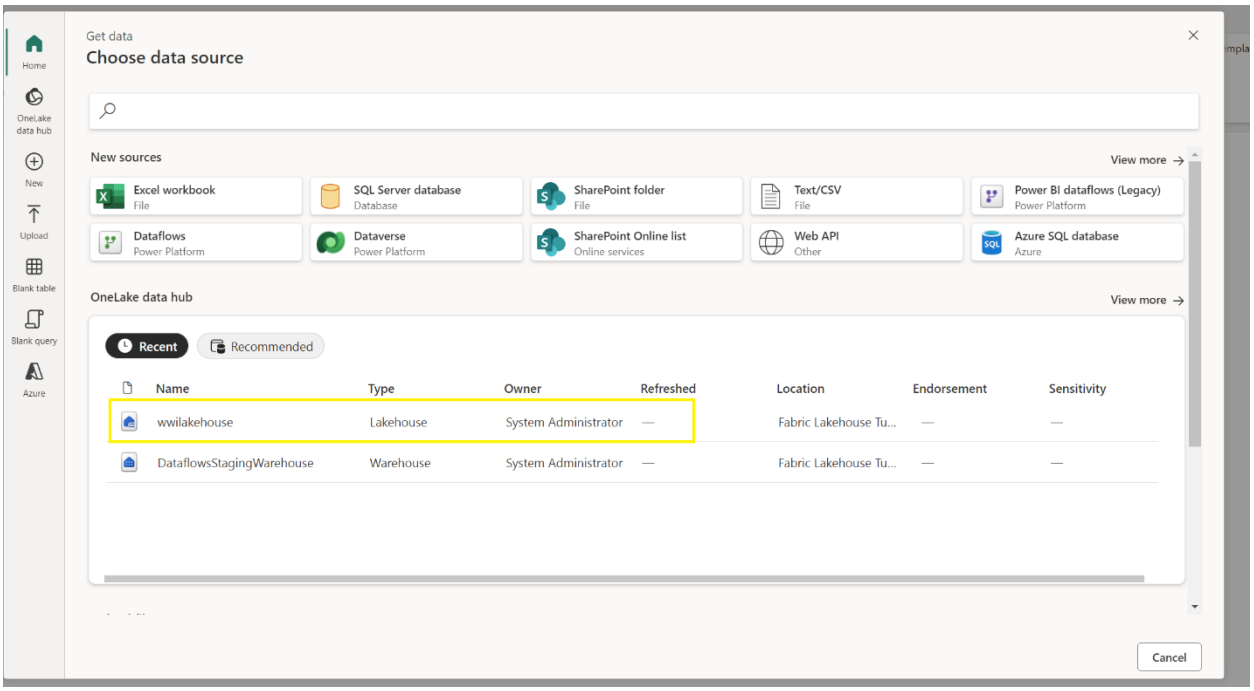




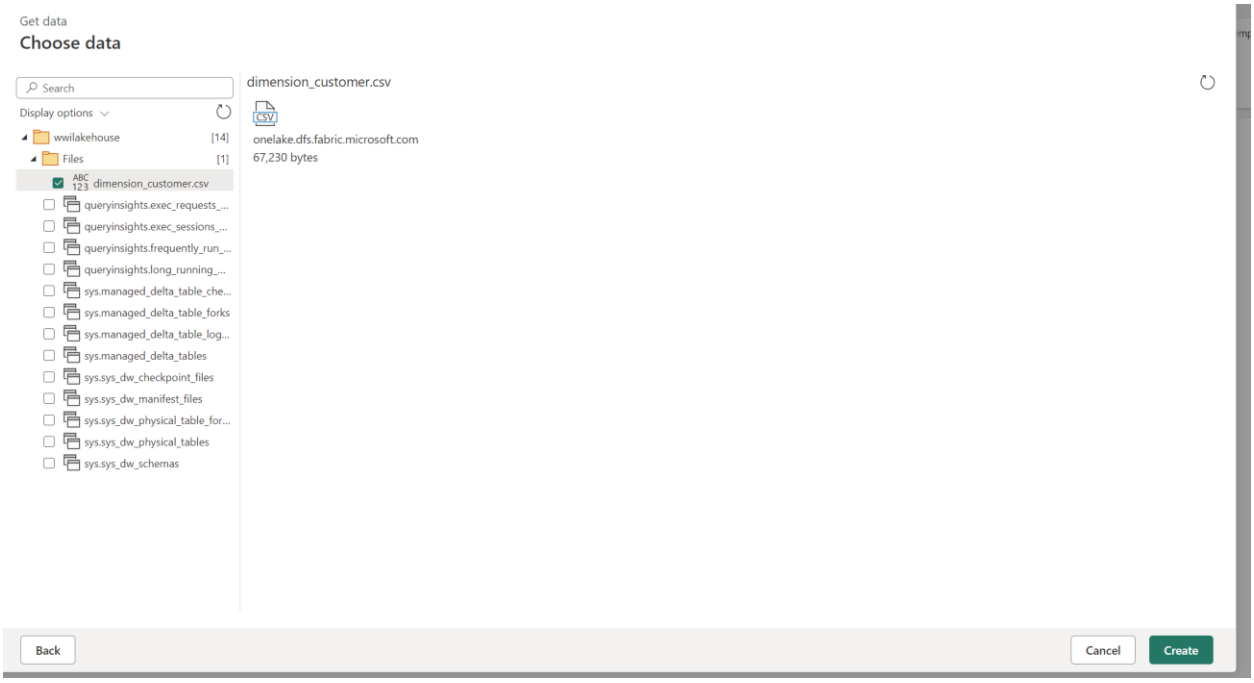


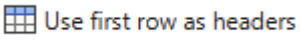
7. After uploading the file, open the dataflow, select the "Get data from another Source" option, and pick the lakehouse established in the previous steps.

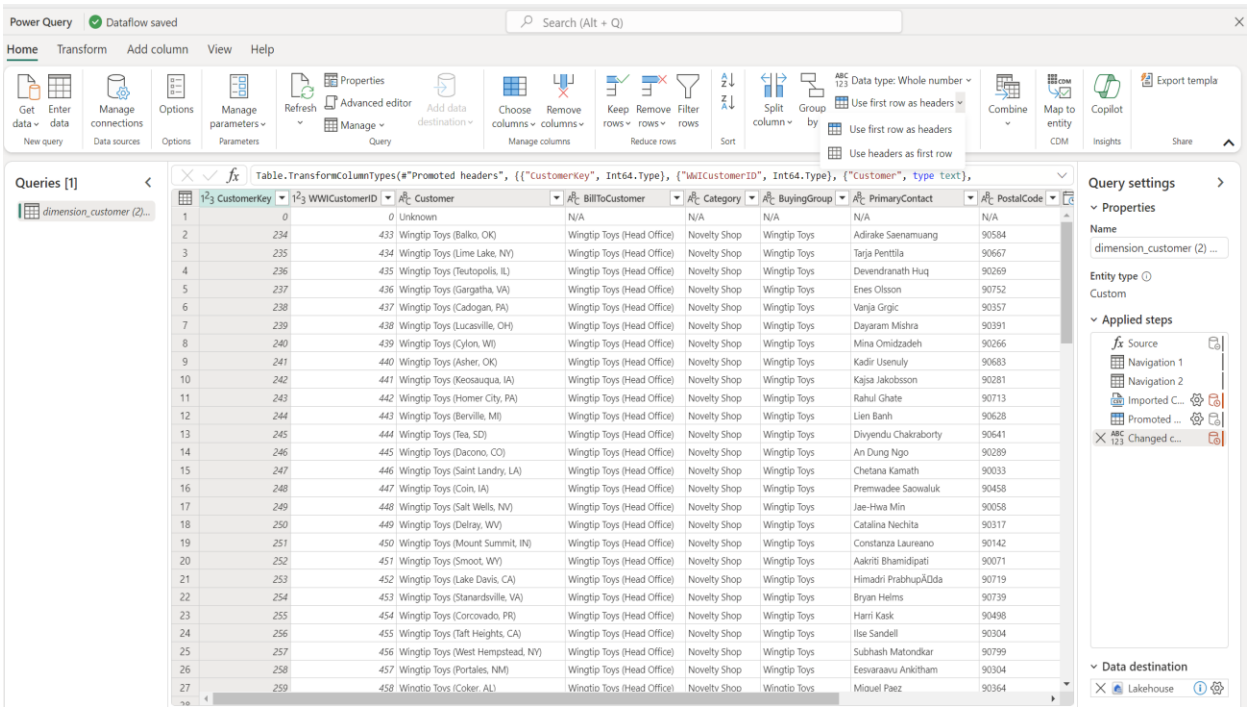




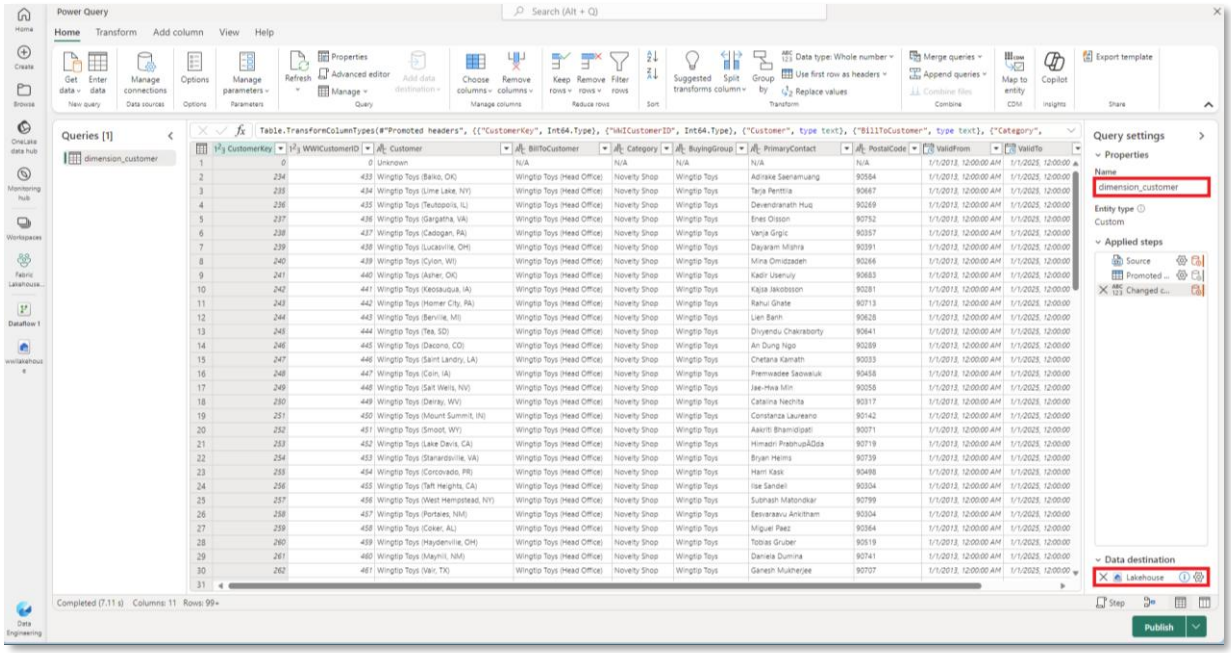
8. Open the file section, select the data file from the lakehouse you've uploaded, and click on create.



9. Select "**Use the first row as headers**" from the Home section in the Power Query home tab to set your column headers. 

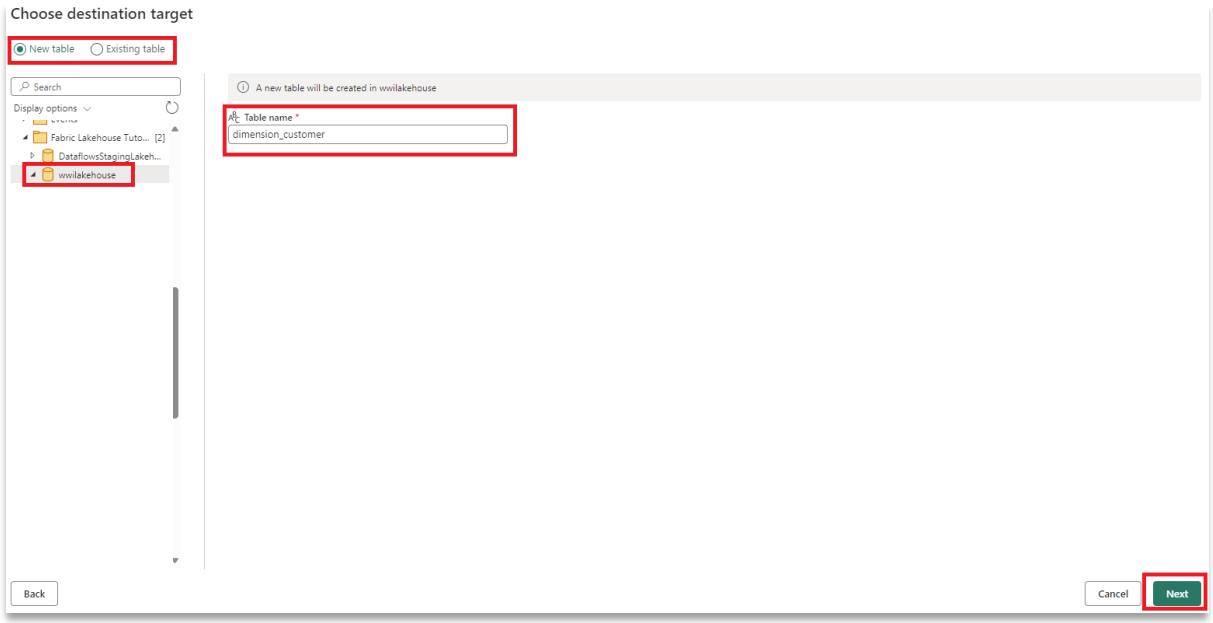


10. In the Name field of the query settings pane, type **dimension_customer**. Then, click on the gear icon in the Data destination field at the bottom of the query settings pane.If the upload file option worked for you the data destination will be automatically selected else No Data destination will be selected, click the plus sign and choose lakehouse.

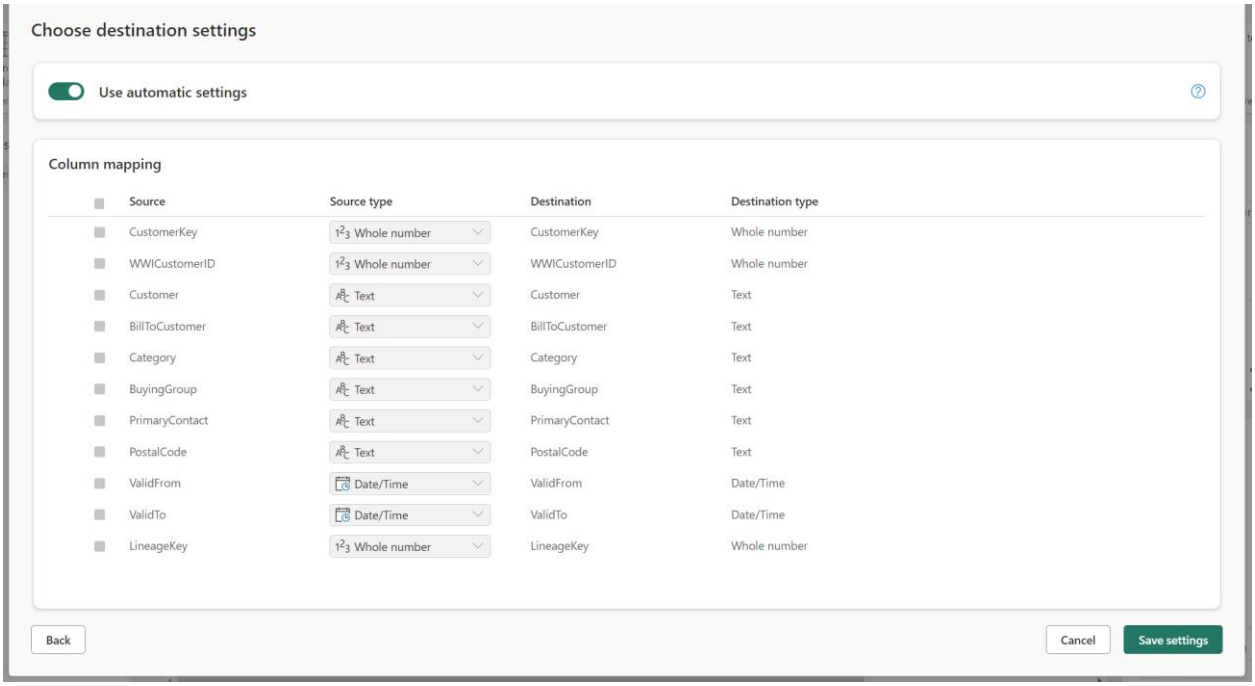


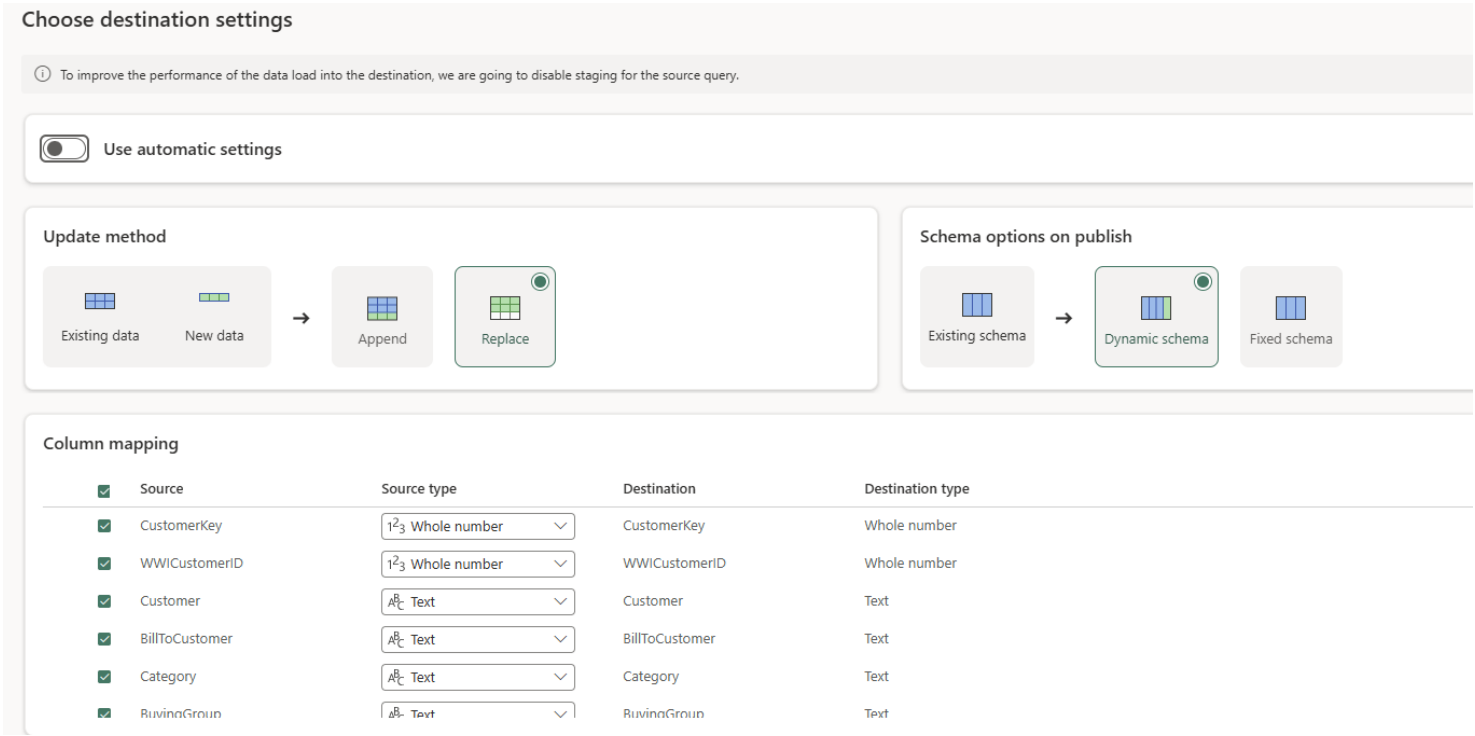
11. If necessary, on the **Connect to data destination** screen, sign into your account. Click **Next**.
12. Navigate to the **wwilakehouse** in your workspace.
13. If the **dimension_customer** table does not exist, select the **New table** setting and enter the Table name of **dimension_customer**. 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.

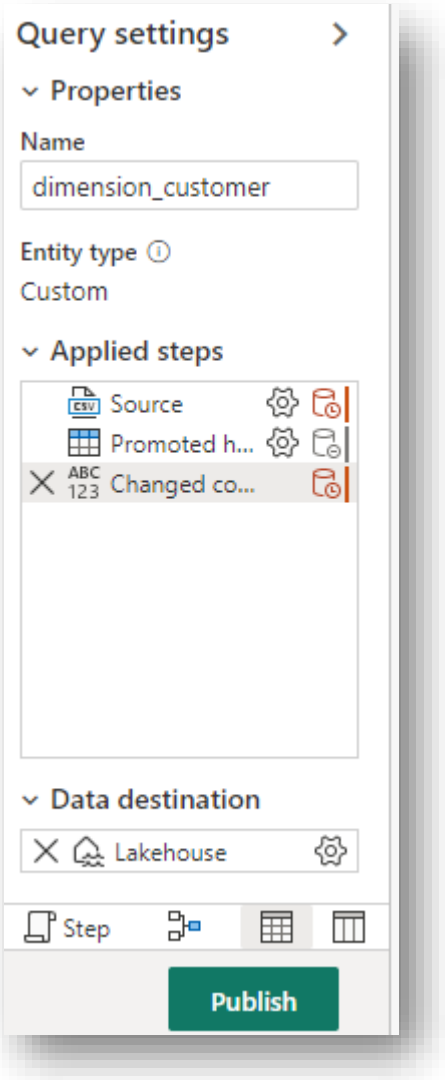


14. For initial dataflow creation, turn off "Use automatic settings"  in the Destination Settings Tab to select the '**Replace**' update option and **Dynamic schema** in the schema options on publish and Click **Save settings**

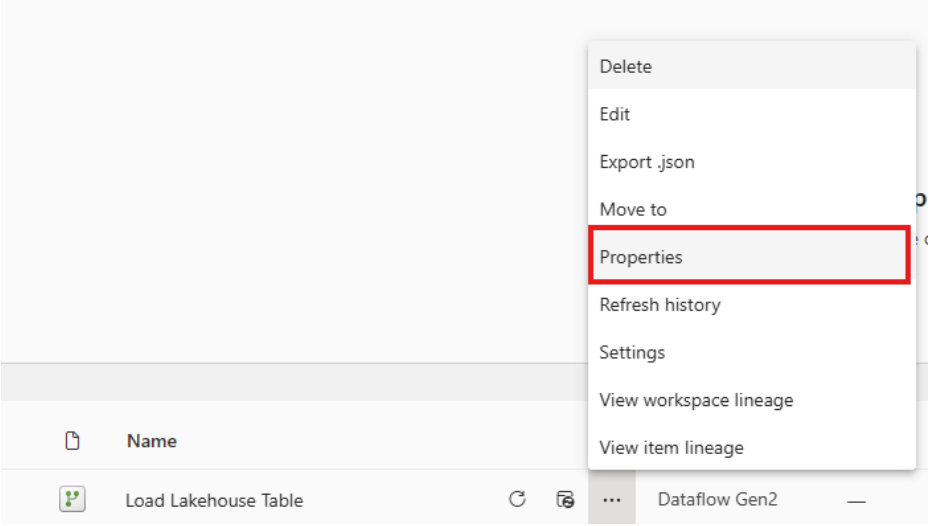

Choose destination settings
Use automatic settings
Column mapping
Source
Source type
Destination
Destination type
CustomerKey
Whole number
CustomerKey
Whole number
WWICustomerID
Whole number
WWICustomerID
Whole number
Customer
Text
Customer
Text
BillToCustomer
Text
BillToCustomer
Text
Category
Text
Category
Text
BuyingGroup
Text
BuyingGroup
Text
PrimaryContact
Text
PrimaryContact
Text
PostalCode
Text
PostalCode
Text
ValidFrom
Date/Time
ValidFrom
Date/Time
ValidTo
Date/Time
ValidTo
Date/Time
LineageKey
Whole number
LineageKey
Whole number
Back
Cancel
Save settings


Choose destination settings
To improve the performance of the data load into the destination, we are going to disable staging for the source query.
Use automatic settings
Update method
Existing data
New data
Append
Replace
Schema options on publish
Existing schema
Dynamic schema
Fixed schema
Column mapping
Source
Source type
Destination
Destination type
CustomerKey
Whole number
CustomerKey
Whole number
WWICustomerID
Whole number
WWICustomerID
Whole number
Customer
Text
Customer
Text
BillToCustomer
Text
BillToCustomer
Text
Category
Text
Category
Text

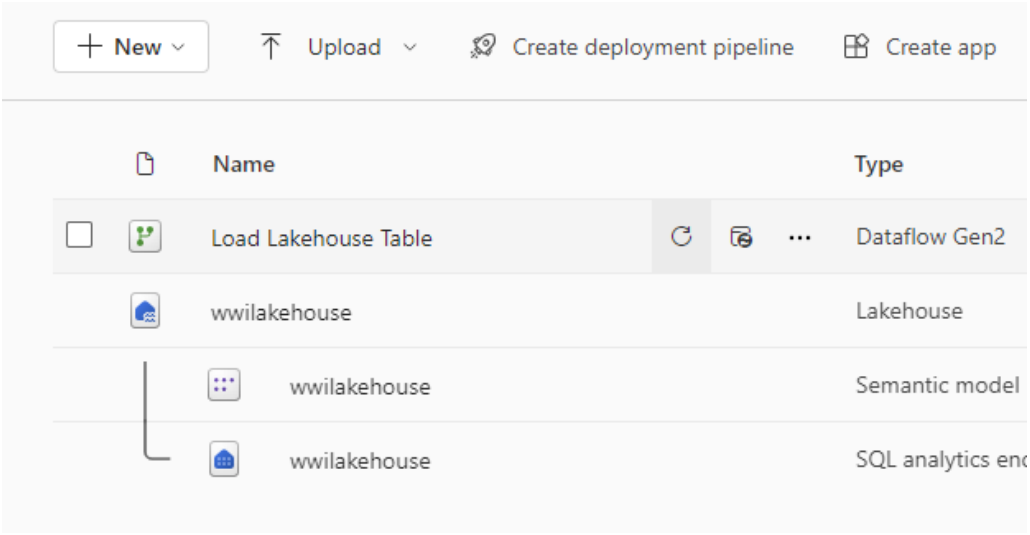
15. This will return you to the canvas of the dataflow. You can easily and quickly transform the data based on your business requirements using this nice and intuitive graphical user interface. For the purpose of this module, we will not make any changes here. To proceed, click on **Publish** at bottom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



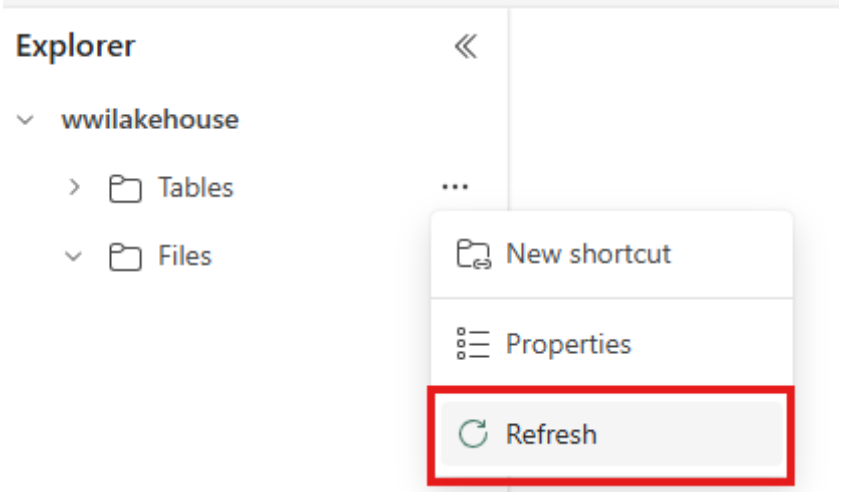
16. A spinning circle next to the dataflow's name will indicate publishing is in progress on the artifact view. When this is completed, click on the ellipsis and select **Properties** to rename the dataflow. For the purpose of this module, Change the name to **Load Lakehouse Table** and select **Save**.

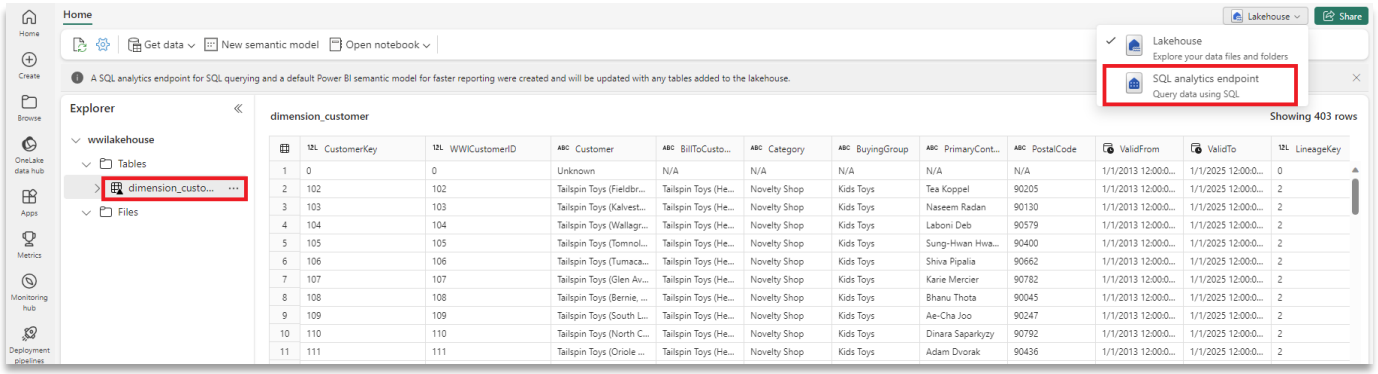


17. Next to the name of the data flow in the artifact view, there is an icon (**Refresh now**) to refresh the dataflow. The data was already refreshed when the dataflow was refreshed. For future refreshes ,
Click on it to kick off execution of dataflow and to move the data from the source file to lakehouse table. While it's in progress, you will see a spinning circle under **Refreshed** column in the artifact view.

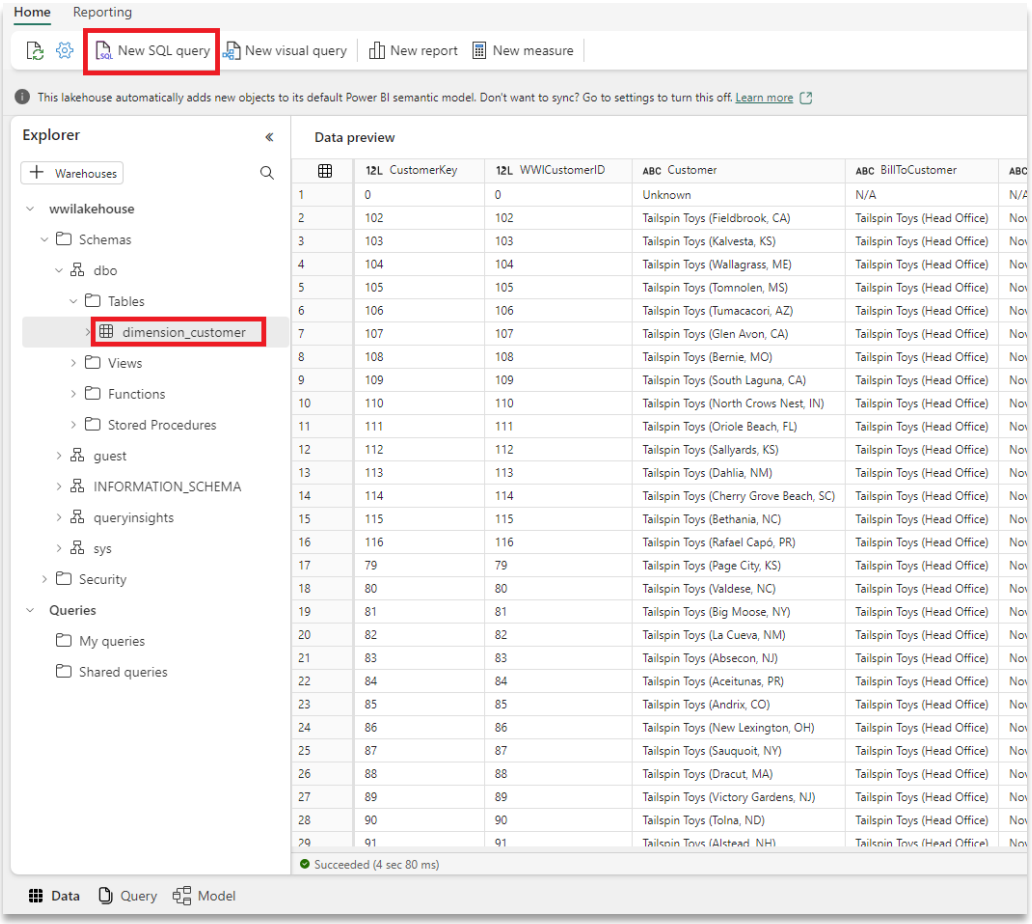


18. Once the dataflow's refresh is completed, you can go to the lakehouse, refresh the explorer view in case you don't see the table(**refresh it twice in case you see an unidentified folder**) and you will notice **dimension_customer** delta table has been created. When you click on it, you should be 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 **Lakehouse** on top right of the screen.





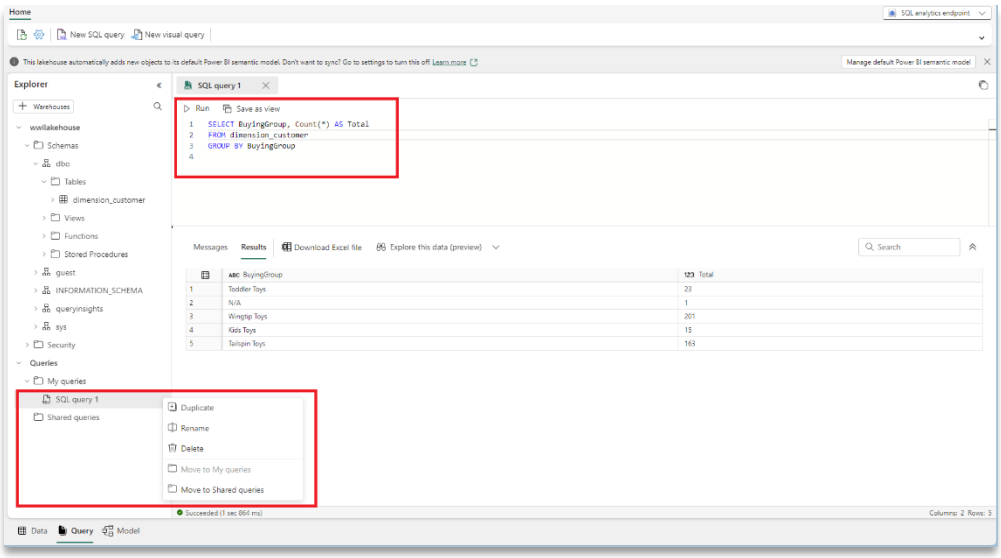
19. In the warehouse mode, you can click on **dimension_customer** table to preview its data and/or click on **New SQL query** to write your SQL statements.



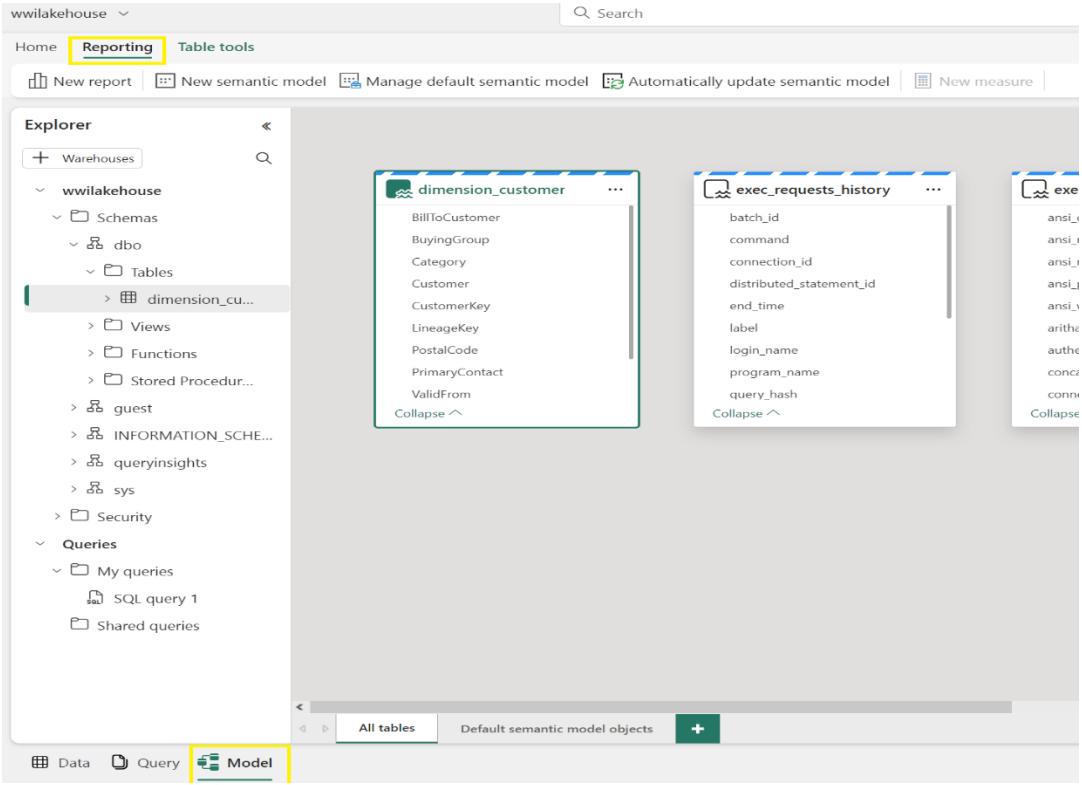
20. Here is a sample query to aggregate the row count based on Buyinggroup column of the **dimension_customer** table and its output. SQL query files are saved automatically for future references, and you can rename or delete these files appropriately based on your need.

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

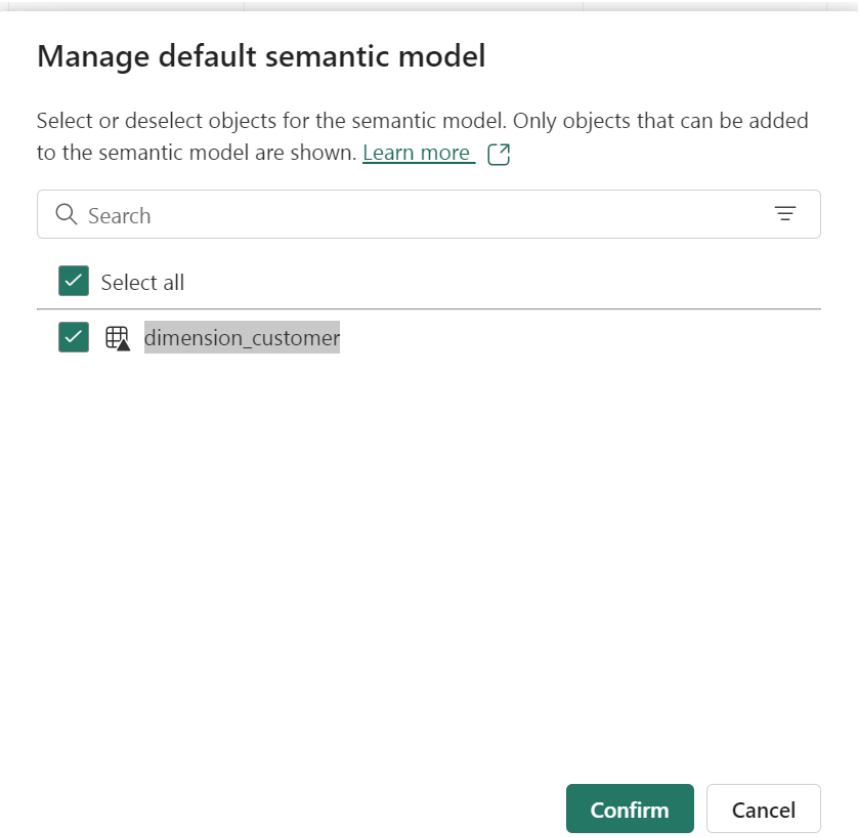
```
SELECT BuyingGroup, Count(*) AS Total
FROM dimension_customer
GROUP BY BuyingGroup
```



21. Click the bottom tab to move to "**Model**" and then choose "**Reporting**" from the upper menu.

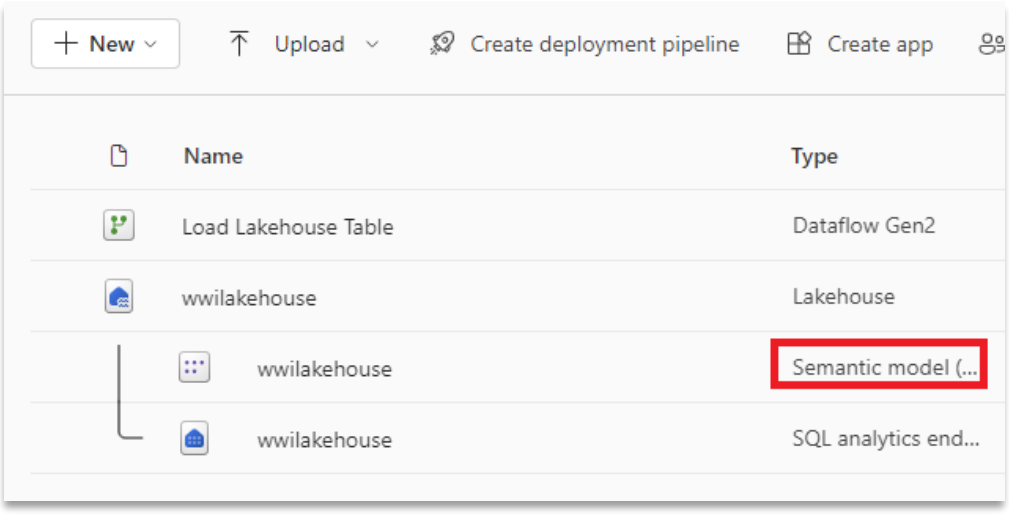


22. Click "Manage default semantic model," choose the `dimension_customer` table, and confirm the selection.

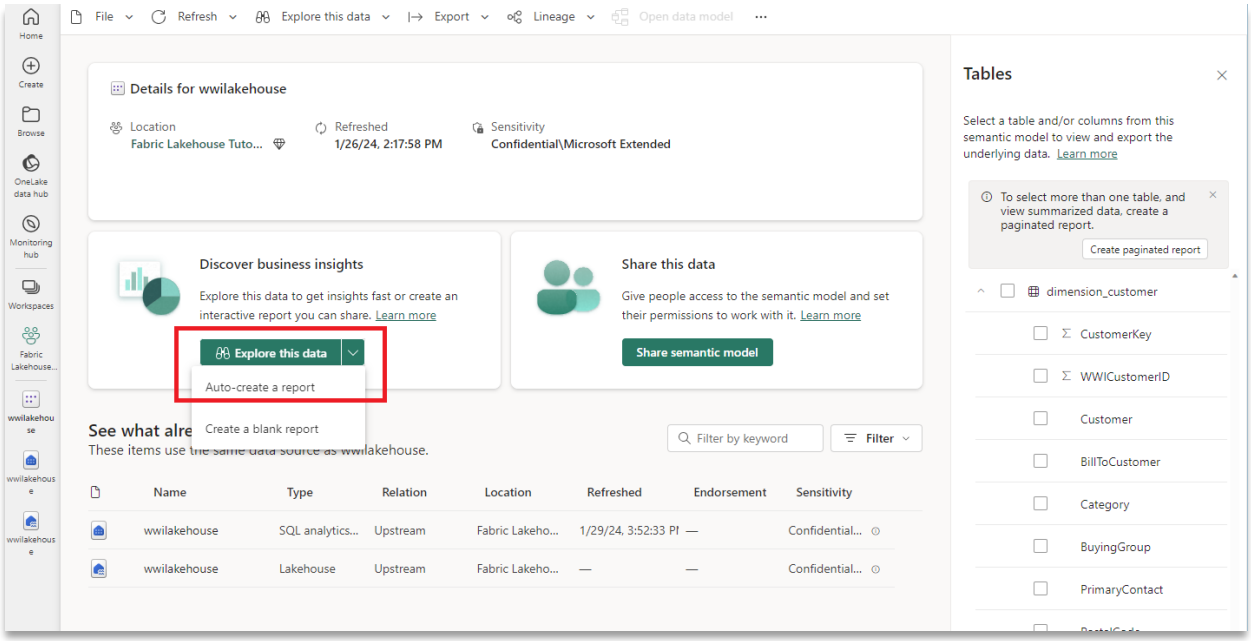


Building a report

1. Click on the workspace name on the left to get to the item 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.



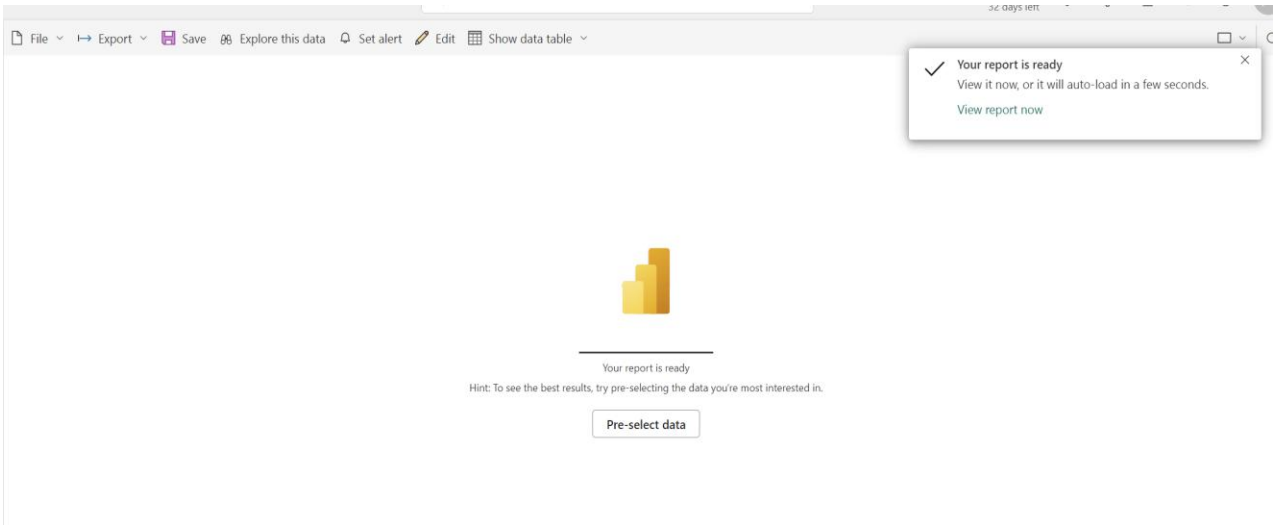
2. On the semantic model screen, you can view all the tables. You will have options to create reports either from scratch, paginated report or let Power BI do magic for you by automatically creating a report based on your data. For the purpose of this module, click on **Auto-create a report** under **Explore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

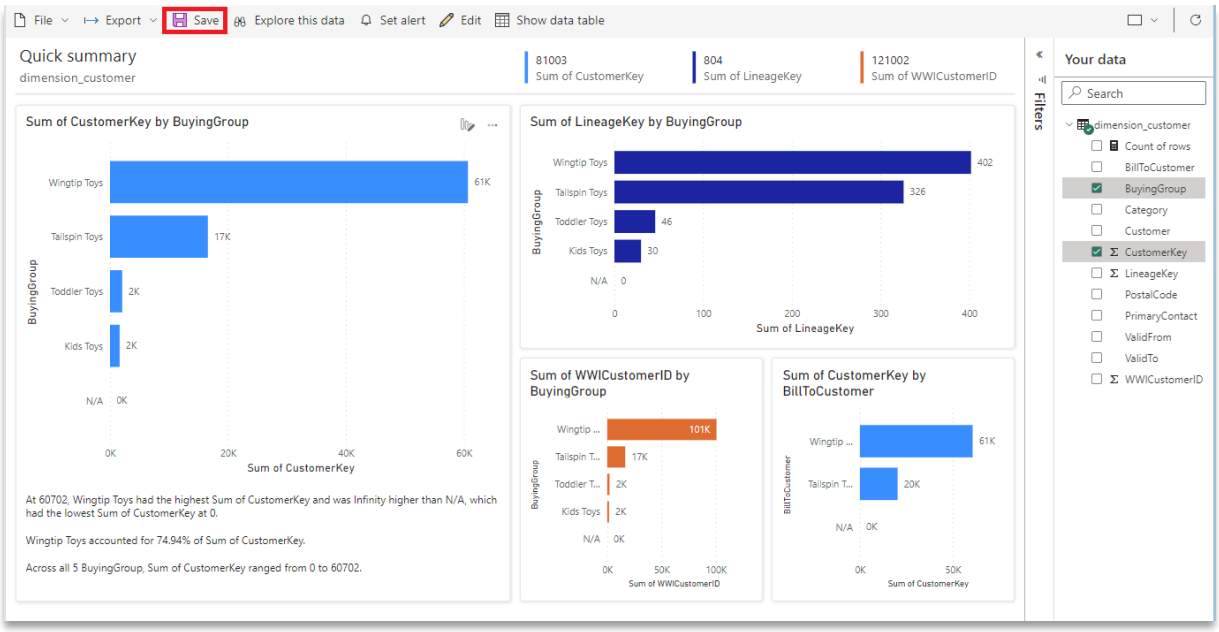


3. Since the table is a dimension and there are no measures in it, Power BI smartly creates a measure for the row sum and aggregates it across different columns and creates different charts as below.

   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.

> ***Generally, report visuals load quickly. If they are slow or not rendering, save the report with a new name to resolve the issue.***




File
Export
Save
Explore this data
Set alert
Edit
Show data table
Quick summary
dimension_customer
81003
Sum of CustomerKey
804
Sum of LineageKey
121002
Sum of WWICustomerID
Sum of CustomerKey by BuyingGroup
Wingtip Toys
61K
Tailspin Toys
17K
Toddler Toys
2K
Kids Toys
2K
N/A
0K
BuyingGroup
0K
20K
40K
60K
Sum of CustomerKey
At 60702, Wingtip Toys had the highest Sum of CustomerKey and was Infinity higher than N/A, which had the lowest Sum of CustomerKey at 0.
Wingtip Toys accounted for 74.94% of Sum of CustomerKey.
Across all 5 BuyingGroup, Sum of CustomerKey ranged from 0 to 60702.
Sum of LineageKey by BuyingGroup
Wingtip Toys
402
Tailspin Toys
326
Toddler Toys
46
Kids Toys
30
N/A
0
BuyingGroup
0
100
200
300
400
Sum of LineageKey
Sum of WWICustomerID by BuyingGroup
Wingtip ...
101K
Tailspin T...
17K
Toddler T...
2K
Kids Toys
2K
N/A
0K
BuyingGroup
0K
50K
100K
Sum of WWICustomerID
Sum of CustomerKey by BillToCustomer
Wingtip ...
61K
Tailspin T...
20K
N/A
0K
BillToCustomer
0K
50K
Sum of CustomerKey
Filters
Your data
Search
dimension_customer
Count of rows
BillToCustomer
BuyingGroup
Category
Customer
CustomerKey
LineageKey
PostalCode
PrimaryContact
ValidFrom
ValidTo
WWICustomerID

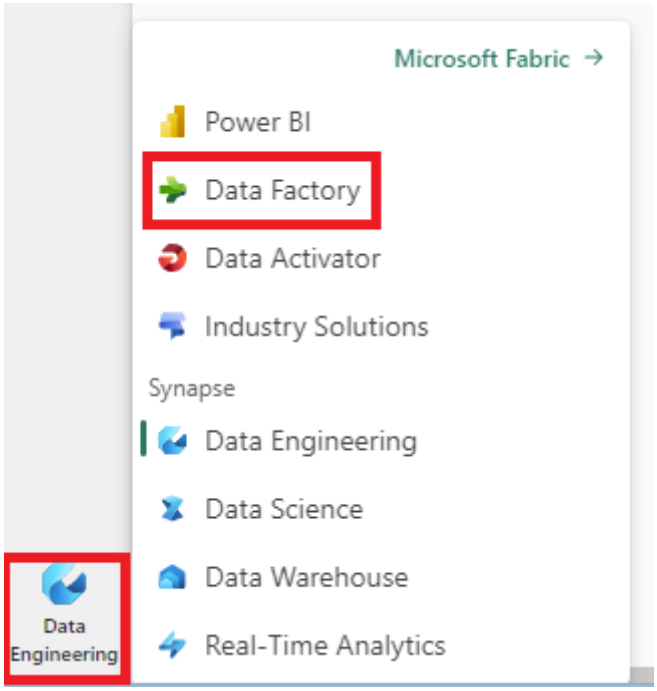
Module 3: Ingest, Prep and Analyze

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.

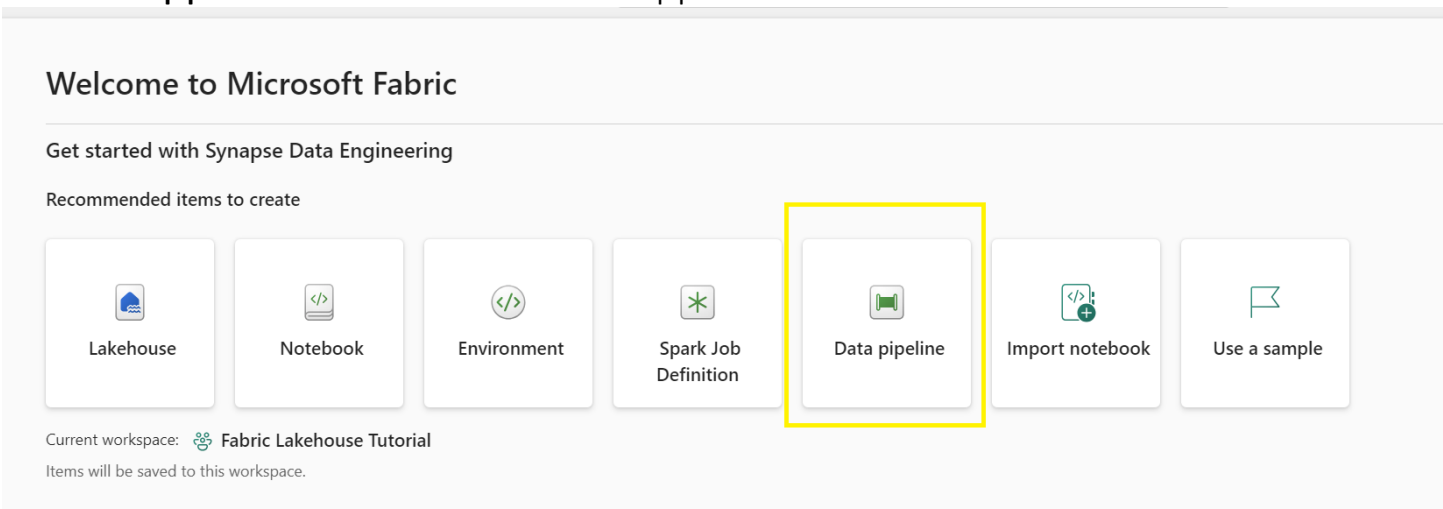
Data Ingestion

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.

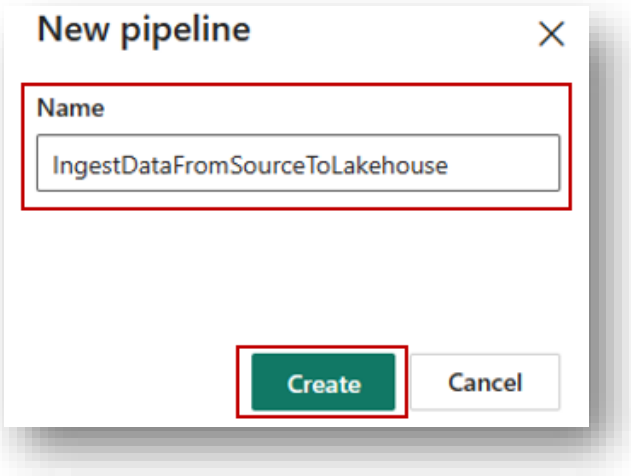
1. On the bottom left of the screen, click on the workload switcher, and then select **Data Factory**.



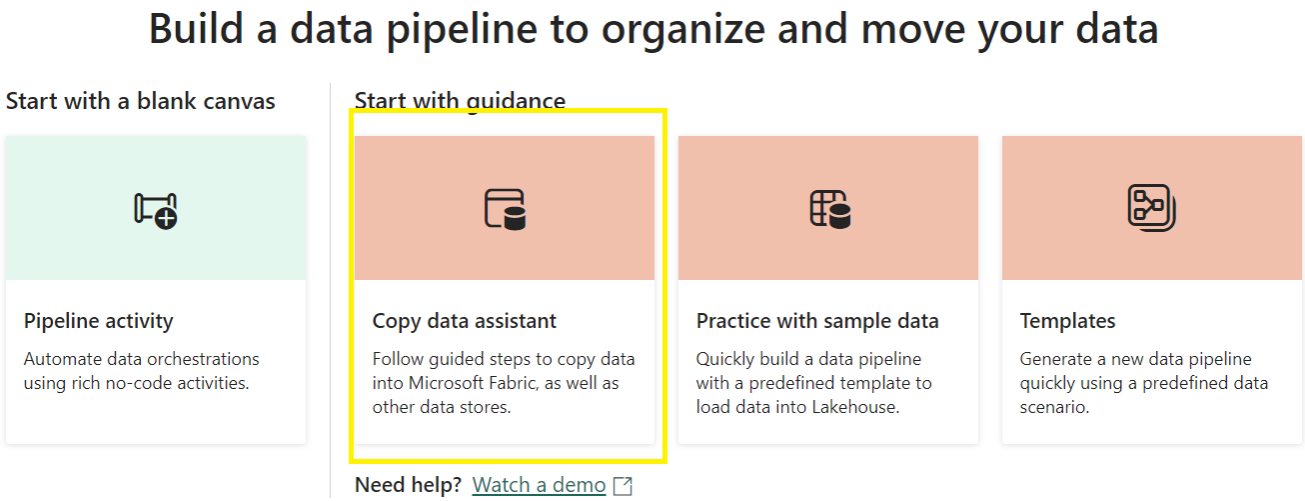
2. Click on **Data pipeline** under **New** to create a new data pipeline.



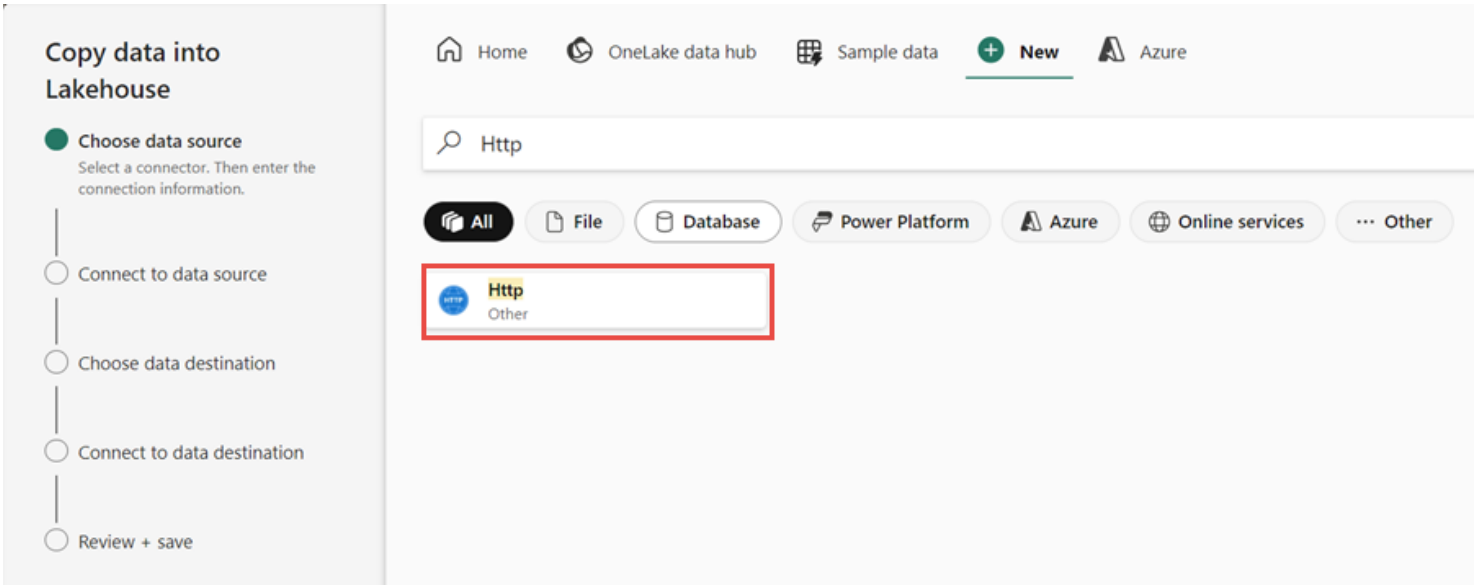
3. For the **New pipeline**, specify the name as **IngestDataFromSourceToLakehouse** and click on **Create**. This will create a new data factory pipeline and open it on the screen to work on it.



4. Select the Copy Data Assistant



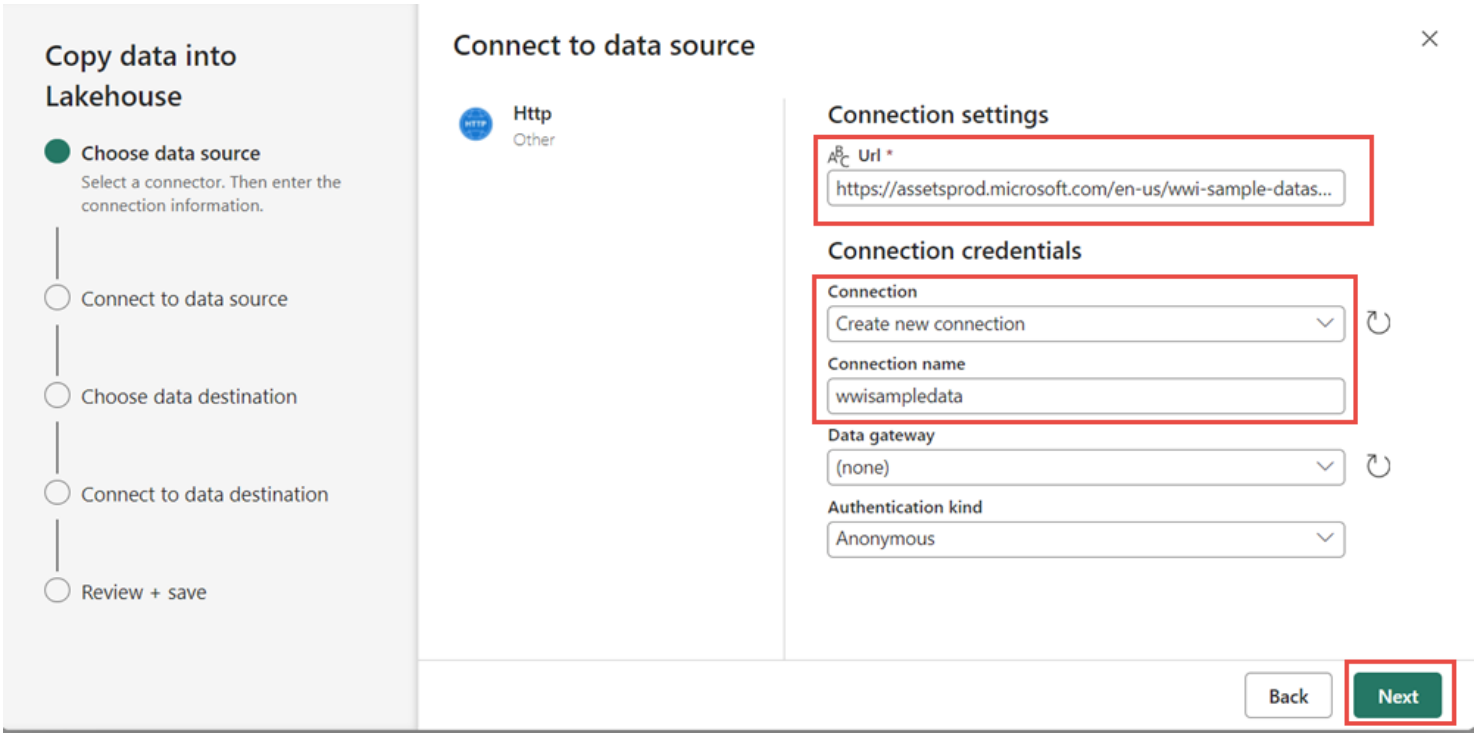
5. Next, set up an HTTP connection to import the sample World Wide Importers data into the Lakehouse. From the list of **New sources**, Type **Http** in the search bar and select it.



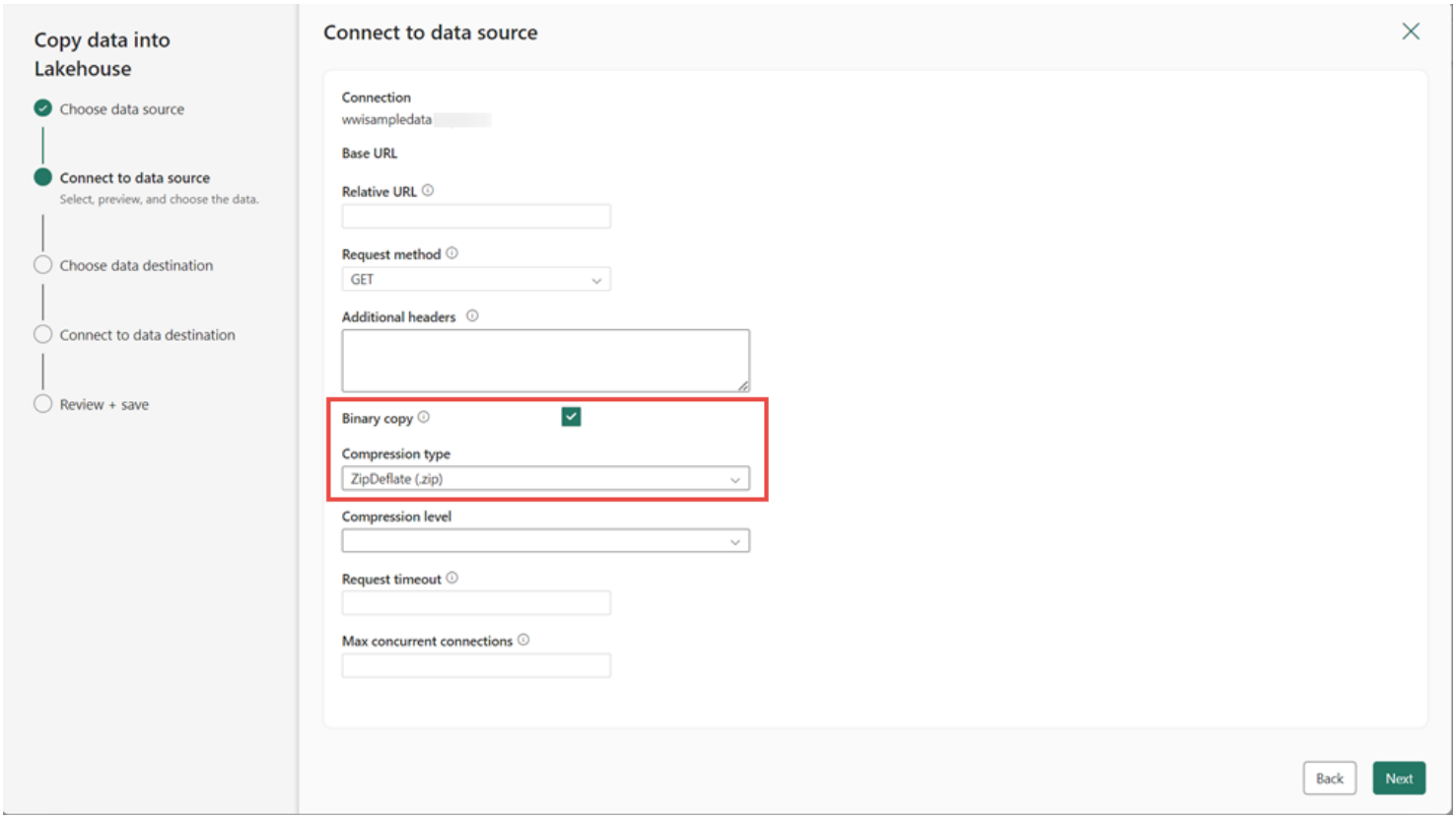
6. In the **Connect to data source** window, enter the details from the table below and select **Next**.

Expand table

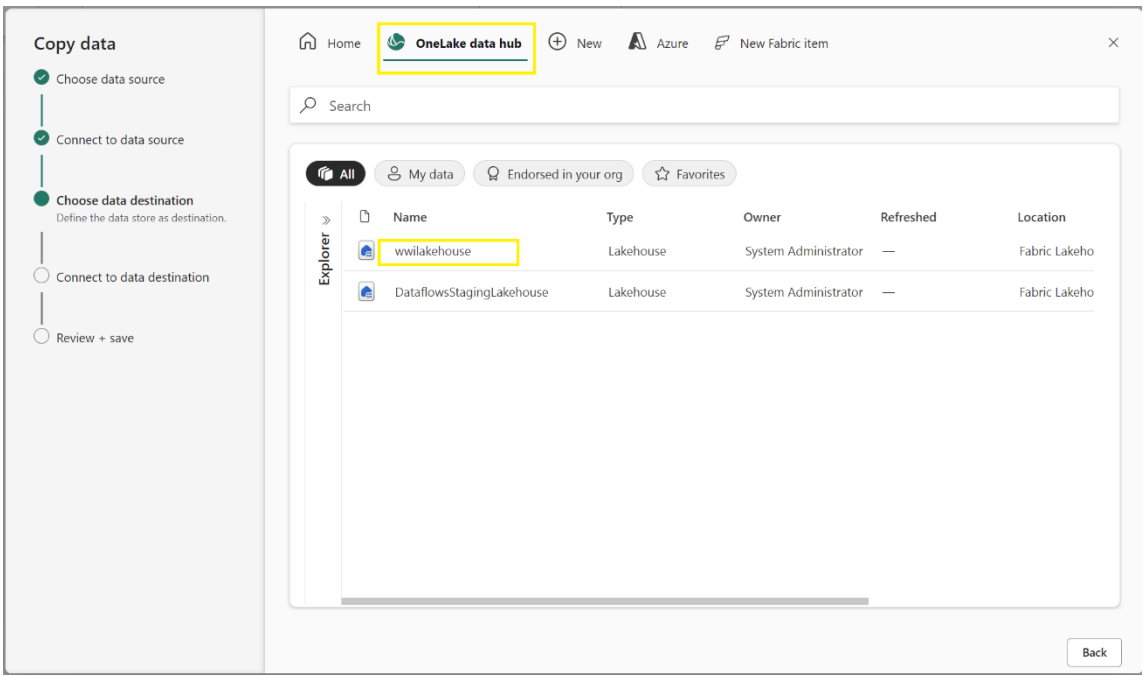
Property	**Value**
URL	https://assetsprod.microsoft.com/en-us/wwi-sample-dataset.zip
Connection	Create a new connection
Connection name	wwisampledata
Data gateway	None
Authentication kind	Anonymous

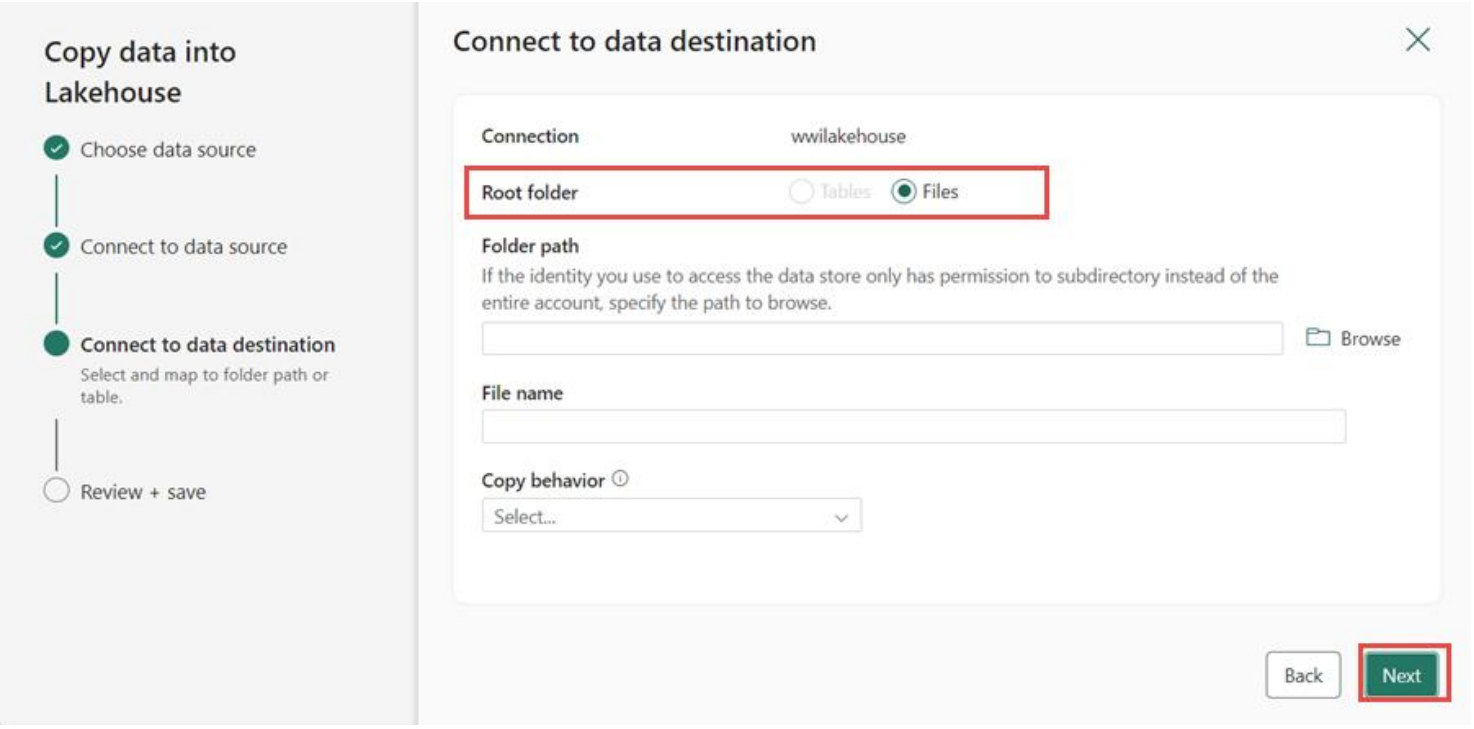


7. In the next step, enable the **Binary copy** and choose **ZipDeflate (.zip)** as the **Compression type** since the source is a .zip file. Keep the other fields at their default values and click **Next**.

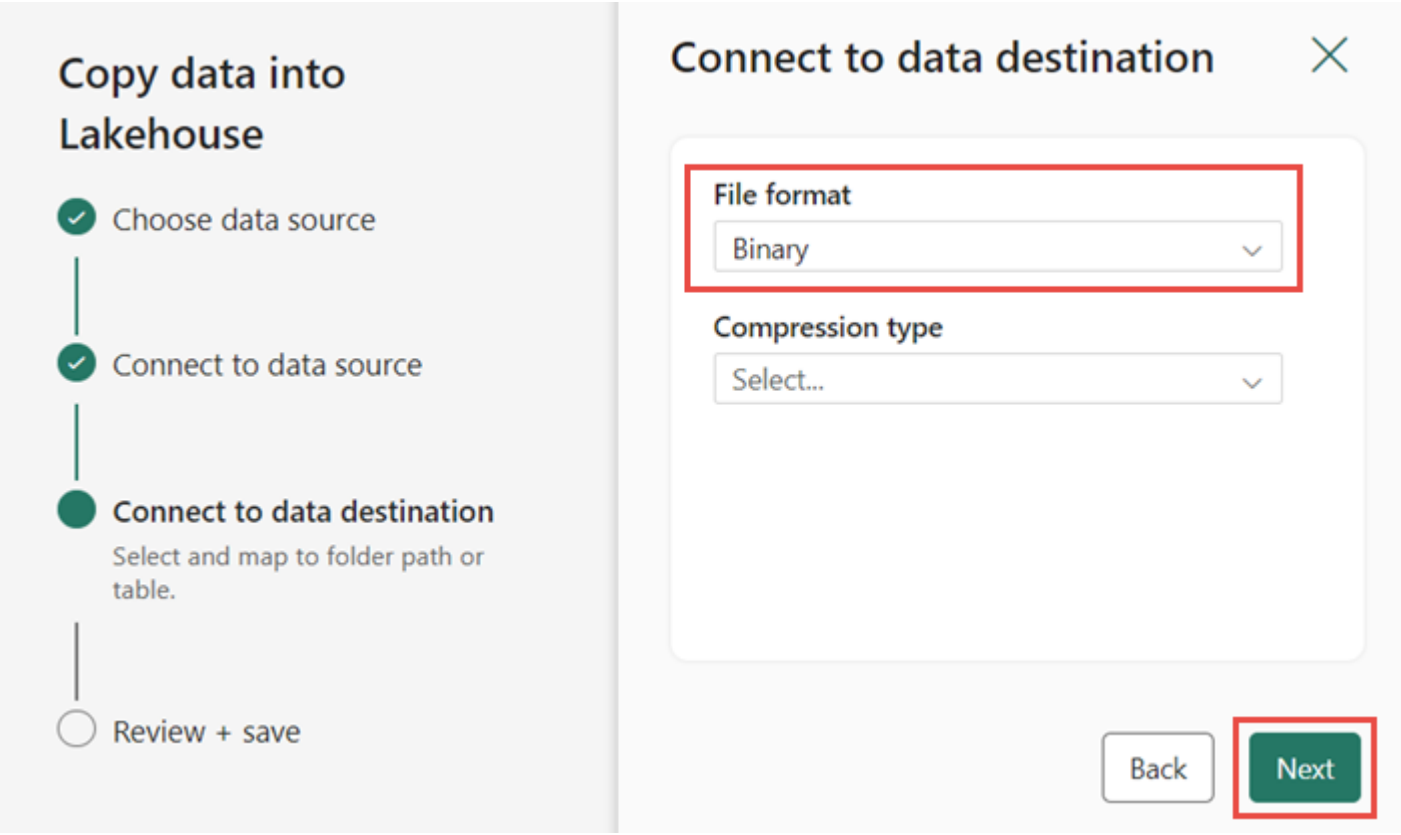


8. In the **Connect to data destination** window, choose the lakehouse from OneLake data hub tab and specify the **Root folder** as **Files** and click **Next**. This will write the data to the *Files* section of the lakehouse.

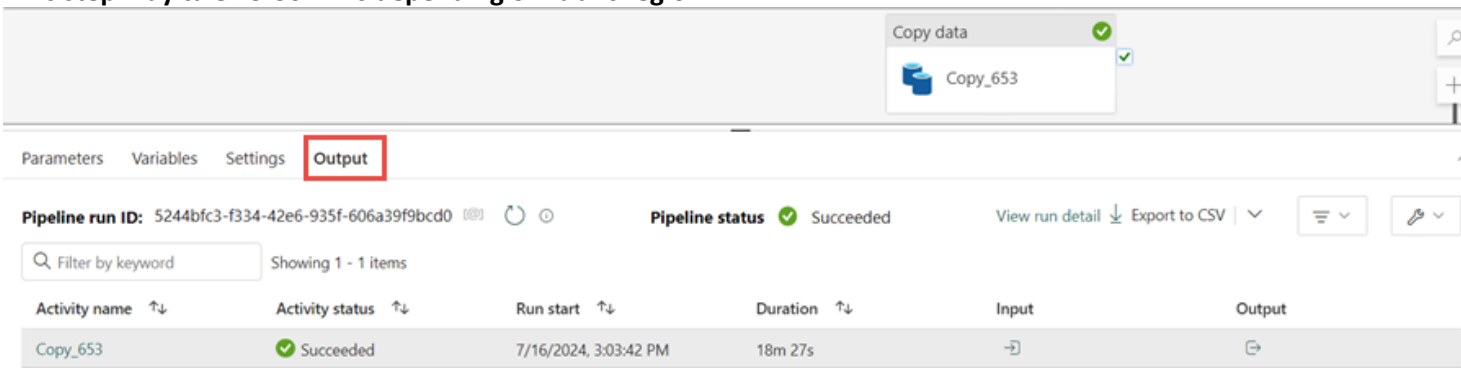




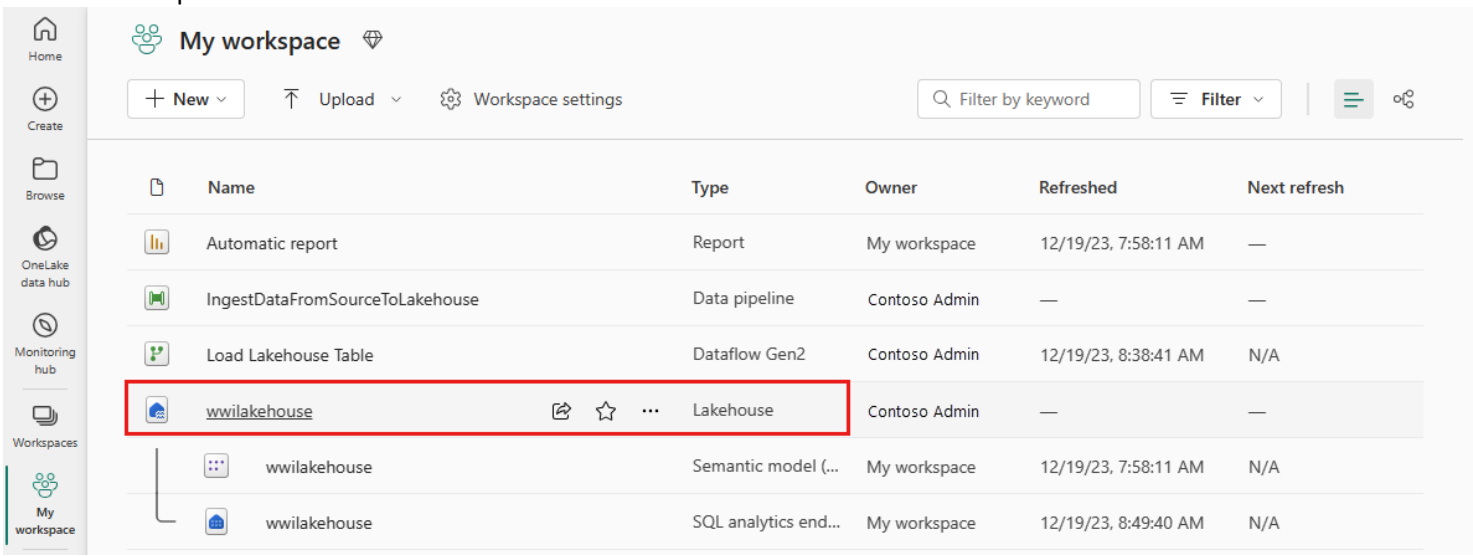
9. Choose the **File format** as **Binary** for the destination. Click **Next** and then **Save+Run**. You can schedule pipelines to refresh data periodically. In this tutorial, we only run the pipeline once. The data copy process takes approximately 10-15 minutes to complete.



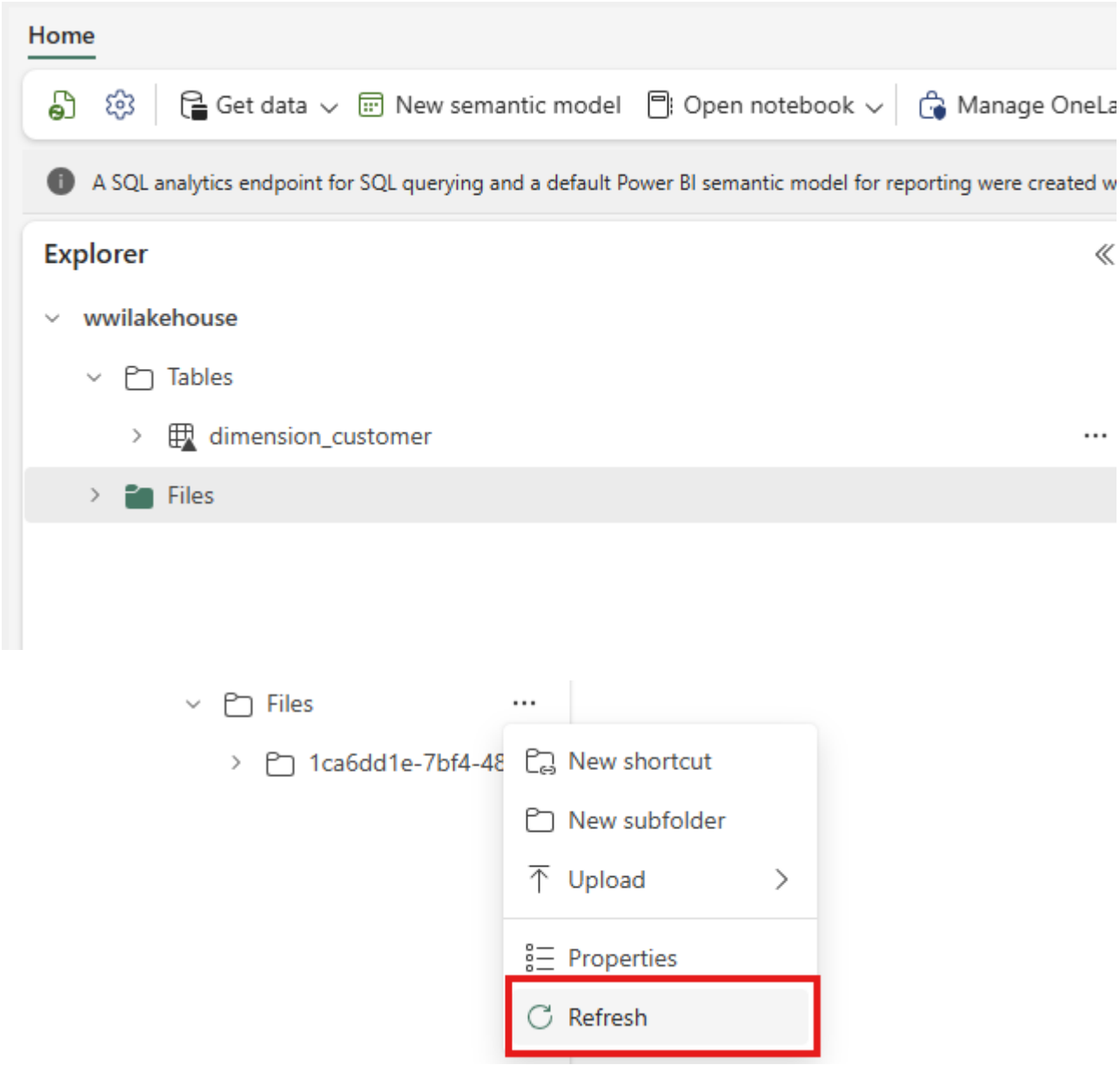
9. You can monitor the pipeline execution and activity in the **Output** tab. You can also view detailed data transfer information by selecting the glasses icon next to the pipeline name, which appears when you hover over the name. **This step may tale 15-30 mins depending on fabric region.**

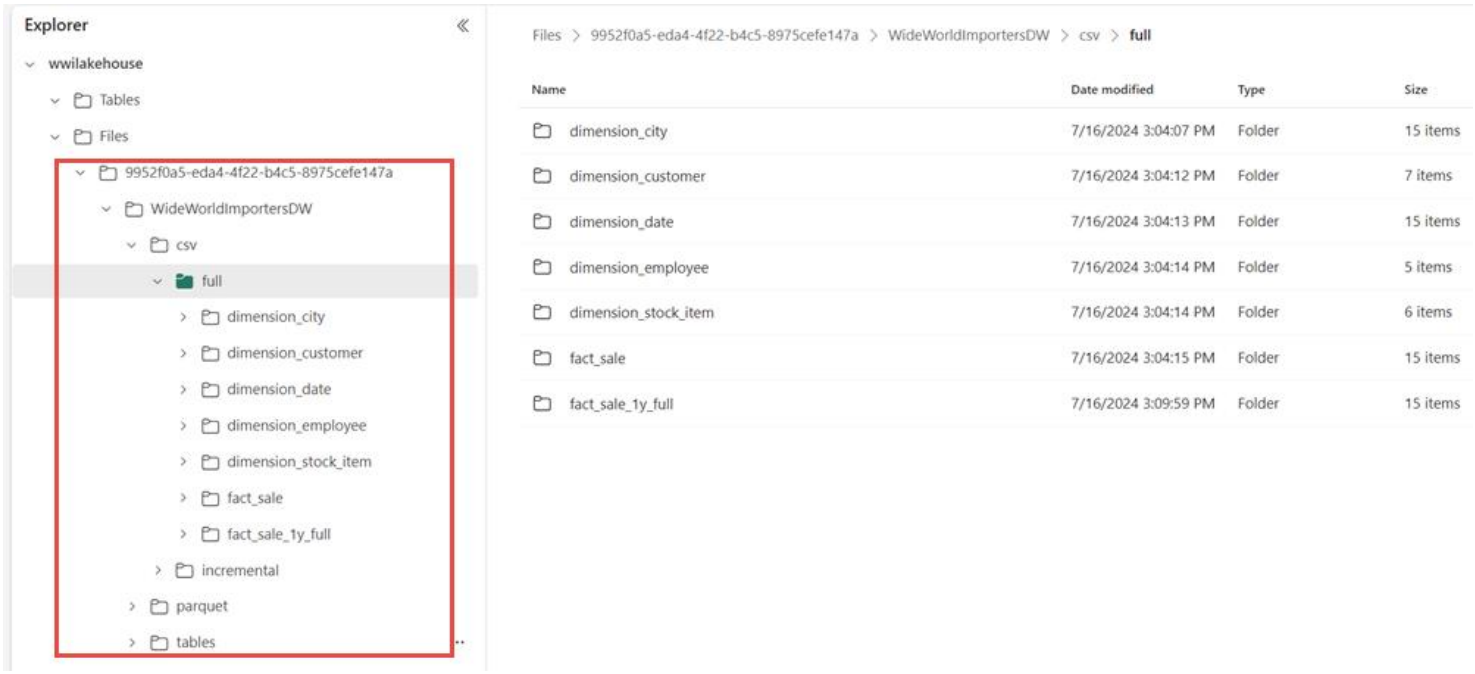


10. After the successful execution of the pipeline, go to your lakehouse (**wwilakehouse**) and open the explorer to see the imported data.

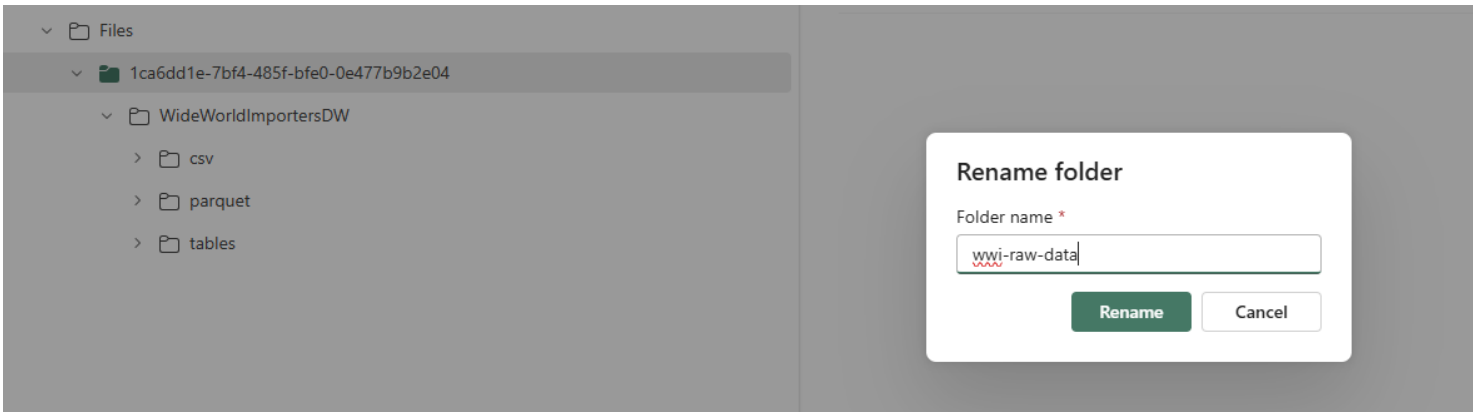


11. Ensure the folder named WideWorldImportersDW is visible in the Explorer view and includes data for every table. If the files do not appear, please click on the refresh icon located under the home tab.


Home
Get data
New semantic model
Open notebook
Manage OneLa
A SQL analytics endpoint for SQL querying and a default Power BI semantic model for reporting were created w
Explorer
wwilakehouse
Tables
dimension_customer
Files
Files
1ca6dd1e-7bf4-48
New shortcut
New subfolder
Upload
Properties
Refresh



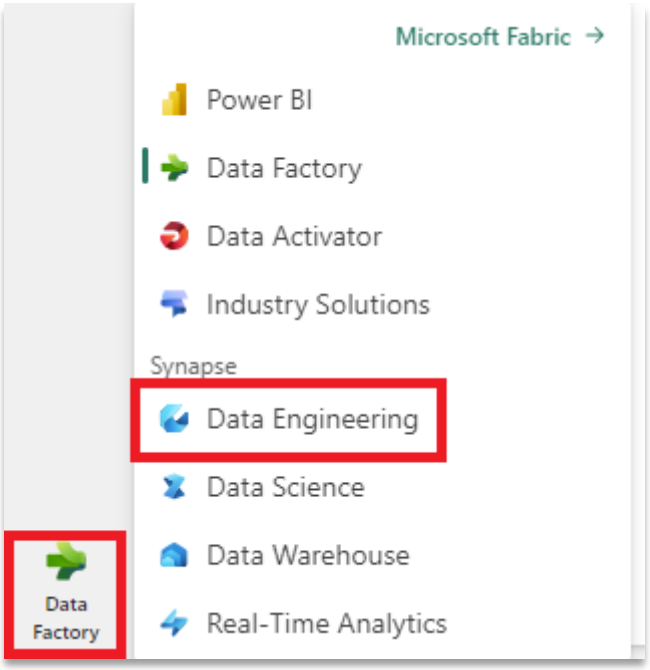
12. The data is created under the **Files** section of the lakehouse explorer. A new folder with GUID contains all the needed data. Rename the GUID to **wwi-raw-data**



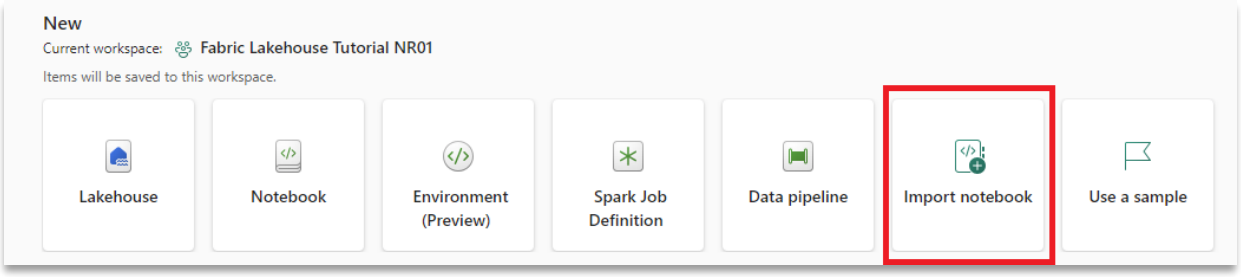
Data Preparation

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

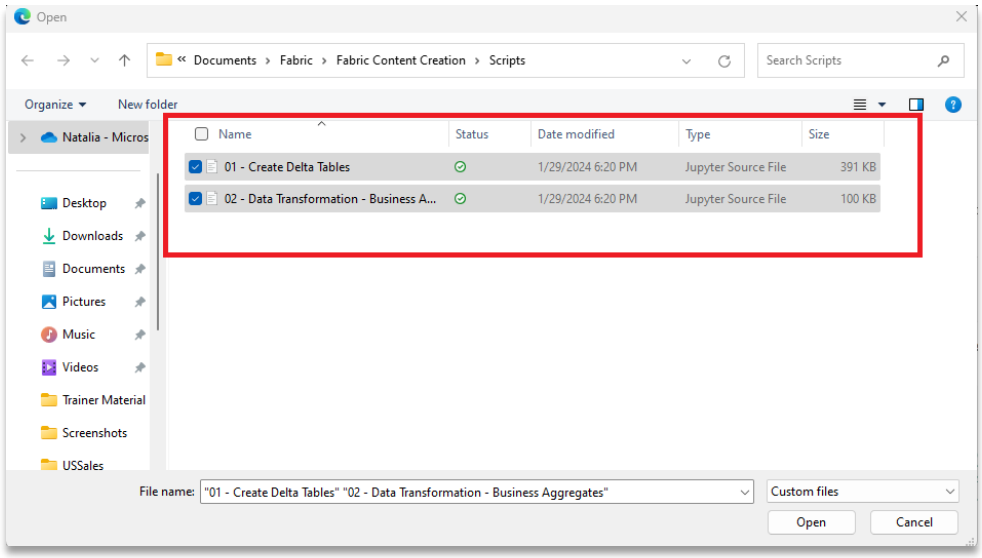
1. Download the set of notebooks found in the Scripts folder
2. From the workload switcher located at the bottom left of the screen, select **Data engineering**.



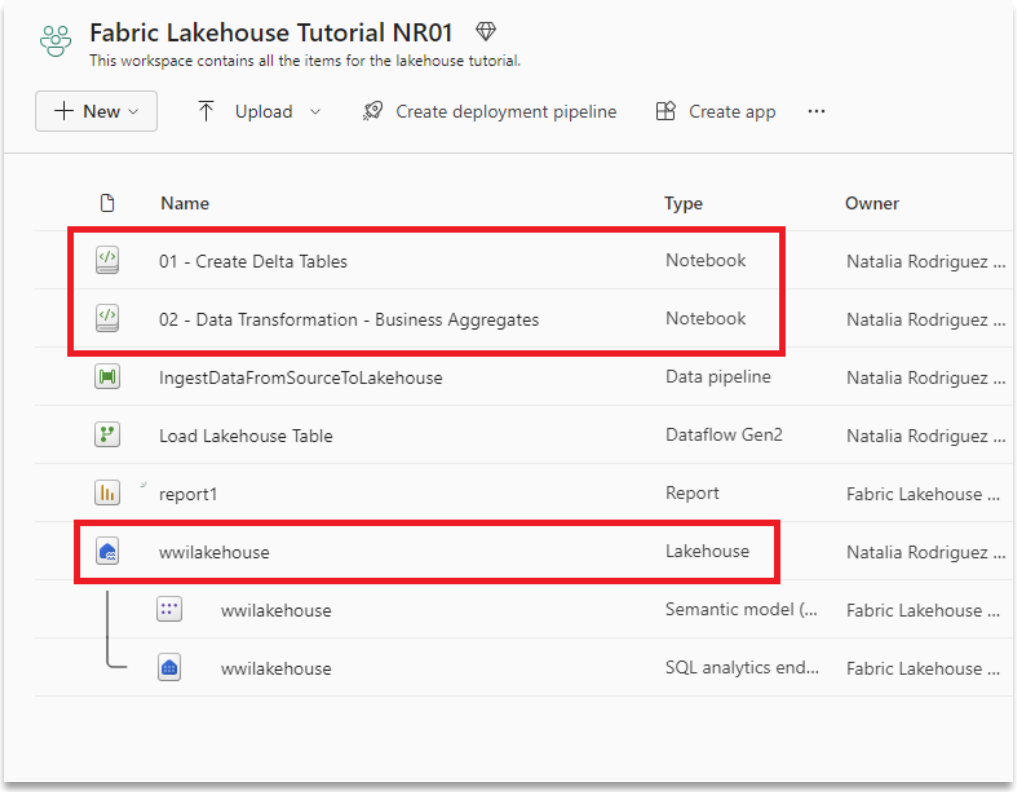
3. Select **Import notebook** from the **New** section at the top of the landing page.



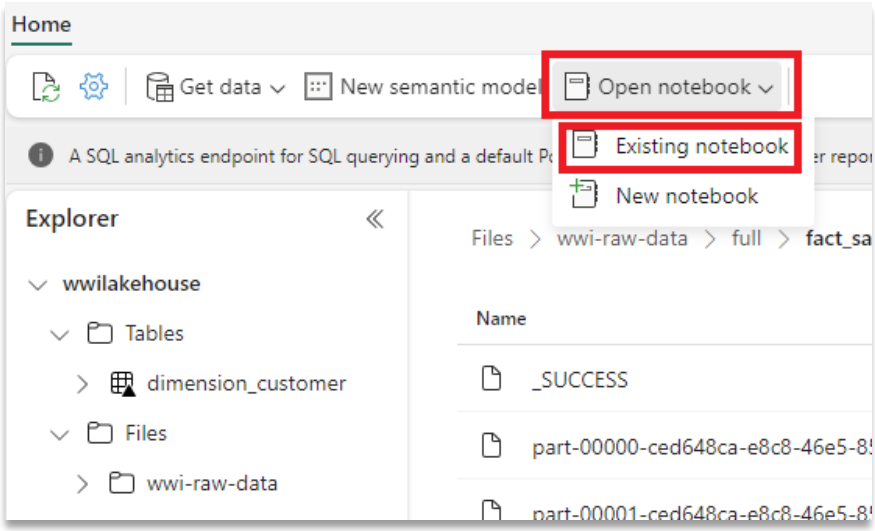
4. Select **Upload** from the **Import status** pane that opens on the right side of the screen.
5. Select all the notebooks that were downloaded and/or unzipped in step 1 of this section.
6. Select **Open**. A notification indicating the status of the import will appear in the top right corner of the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


Open
Documents › Fabric › Fabric Content Creation › Scripts
Search Scripts
Organize
New folder
Natalia - Micros
Desktop
Downloads
Documents
Pictures
Music
Videos
Trainer Material
Screenshots
USSales
Name
Status
Date modified
Type
Size
01 - Create Delta Tables
1/29/2024 6:20 PM
Jupyter Source File
391 KB
02 - Data Transformation - Business A...
1/29/2024 6:20 PM
Jupyter Source File
100 KB
File name:
"01 - Create Delta Tables" "02 - Data Transformation - Business Aggregates"
Custom files
Open
Cancel

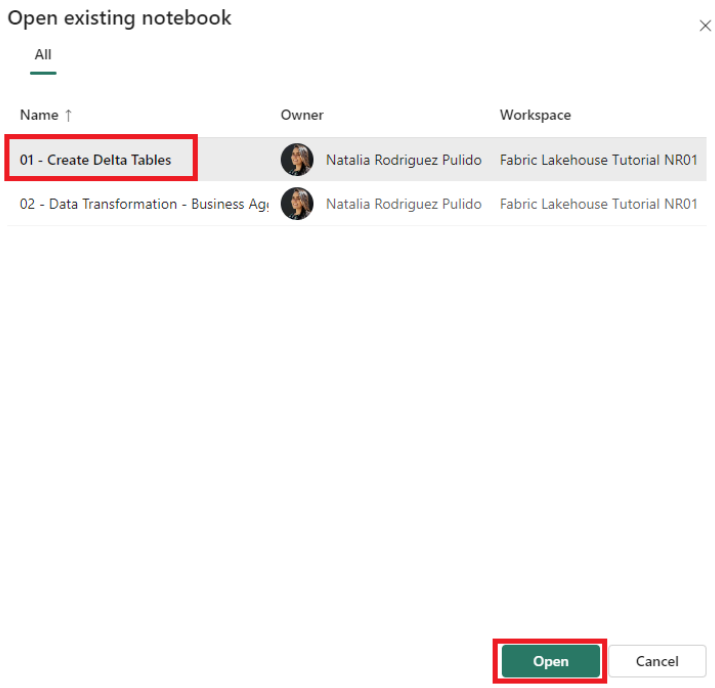
7. After the import of notebooks is successful, you can go to items view of the workspace and see these newly imported notebooks. Click on **wwilakehouse** lakehouse to open it.



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.

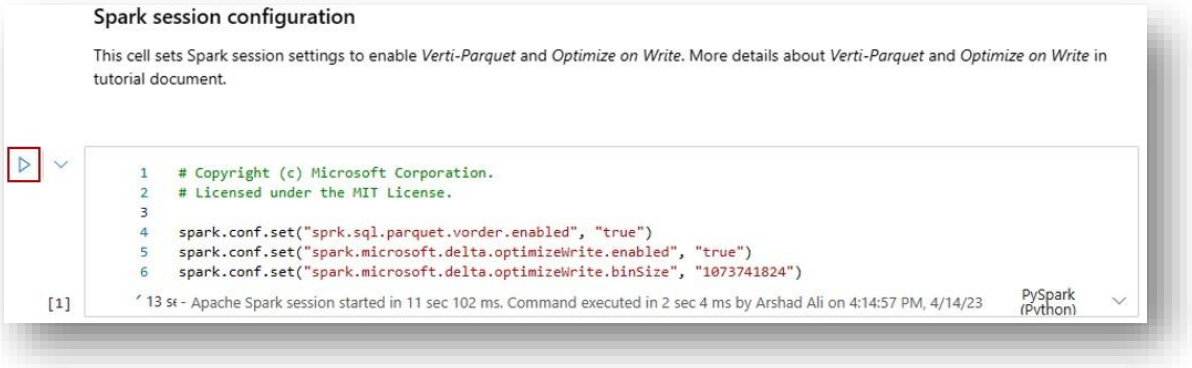
> **Note**
> Fabric provides these unique capabilities for writing optimized delta lake files:
> - Verti-Parquet – Fabric includes Microsoft 's unique VertiParquet IP. VertiParquet transparently optimizes the Delta Lake files in a way that is highly optimized by Fabric

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.

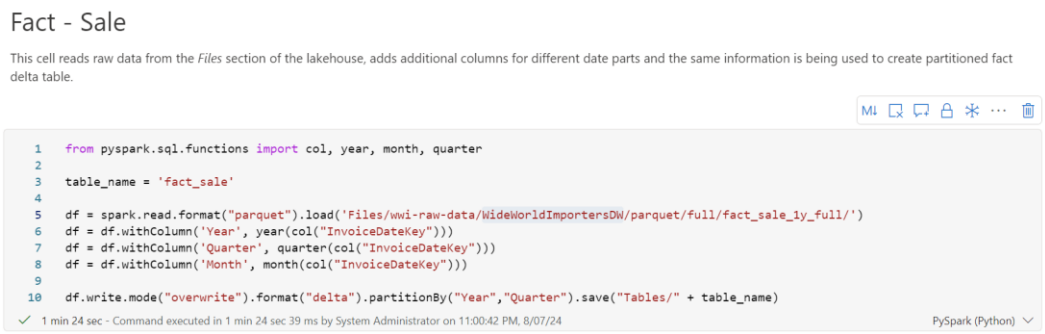
11. Before you write data as delta lake tables in the **Tables** section of the lakehouse, you are going to use two features (**Verti-Parquet** and **Optimize Write**) of Fabric for optimized data writing and for subsequent better reading performance. To enable these features in your session you can 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.



    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 means that when users create notebooks, they don't have to worry about specifying any Spark configurations or cluster details (or something like that) and when they execute their first command in the notebooks the live pool kicks in and is up in a few seconds after establishing their Spark session and starts executing the code in the cell. Subsequent code execution is almost instantaneous in this notebook as long as the Spark session is active.

12. Next, you will read raw data from the **Files** section of the lakehouse, add additional columns for different date parts as part of the transformation. Finally, you will use partitionBy Spark API to partition the data before writing it as delta table 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/WideWorldImportersDW/parquet/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("Ta
bles/" + table_name)
```

13. After fact data 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 table names 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 created function with each table name as parameter to read data for that specific table and create delta table respectively.

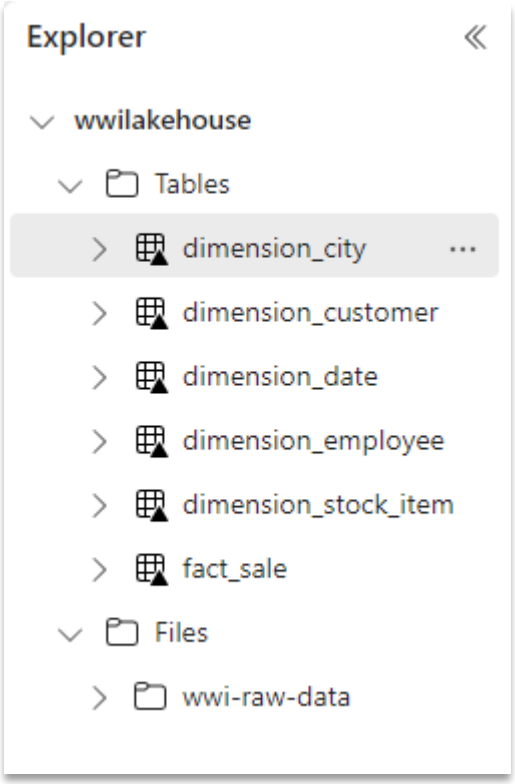

```
from pyspark.sql.types import *

def loadFullDataFromSource(table_name):
    df = spark.read.format("parquet").load('Files/wwi-raw-
data/WideWorldImportersDW/parquet/full/' + table_name)
    df = df.select([c for c in df.columns if c != 'Photo'])
    df.write.mode("overwrite").format("delta").save("Tables/" + table_name)
```

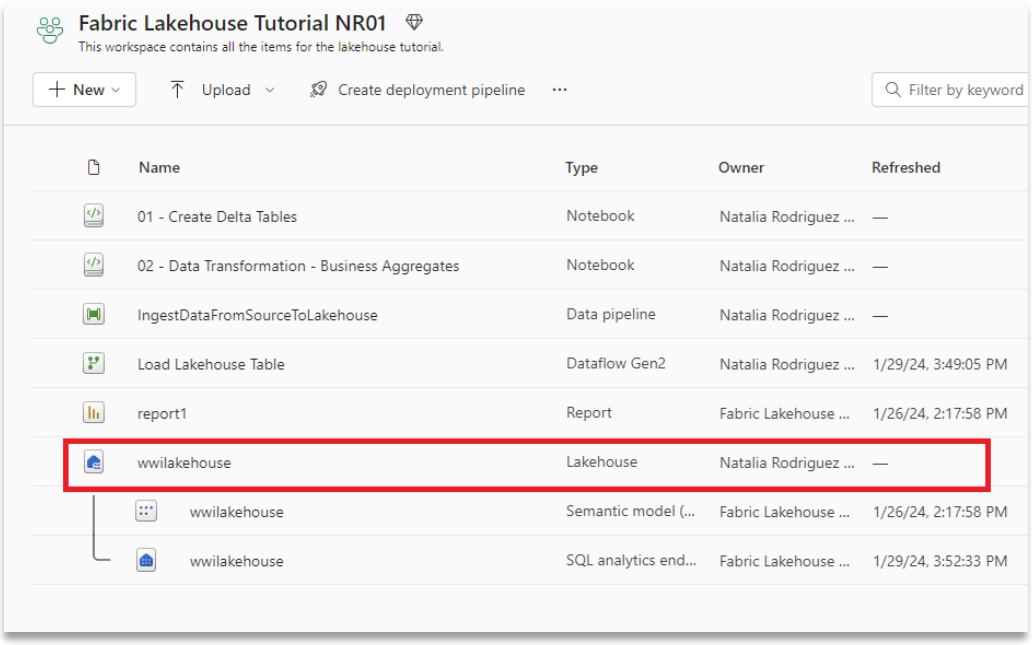
```
full_tables = [
    'dimension_city',
    'dimension_date',
    'dimension_employee',
    'dimension_stock_item'
    ]

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

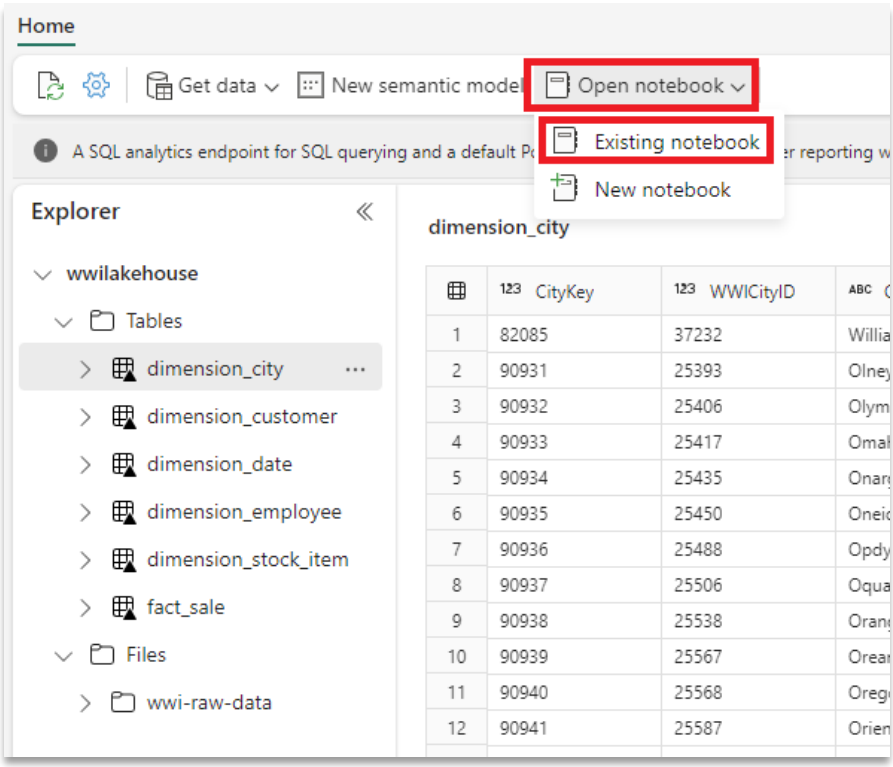
14. You can validate created tables by right clicking and selecting refresh on **wwilakehouse** lakehouse and you will notice these created tables. **Please refresh the explorer view in case you don't see it.**



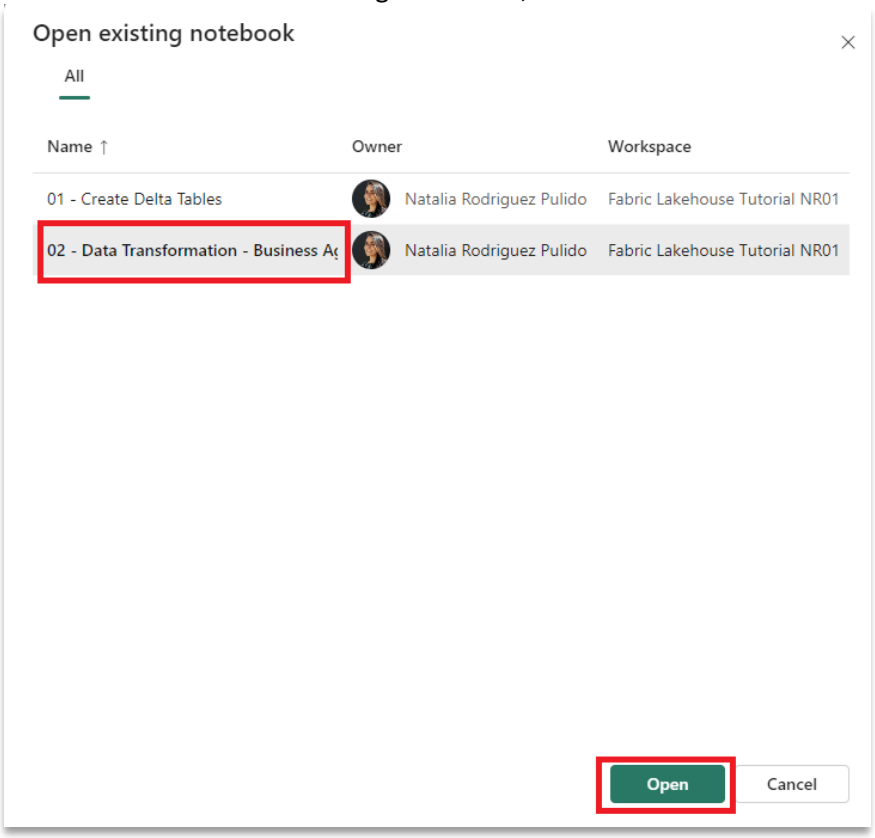
15. Please go the items view of the workspace again and click on **wwilakehouse** lakehouse to open it.



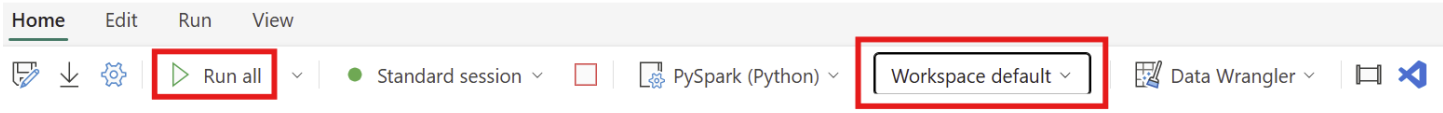
16. Now, open the second notebook. In the lakehouse view, click on **Open notebook** > **Existing notebook** on the ribbon at the top.



17. From the list of existing notebooks, select the **02 - Data Transformation - Business** 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. Click **Run all** to execute the notebook . In case you are getting a session error make sure the environment is set to **Workspace default**.



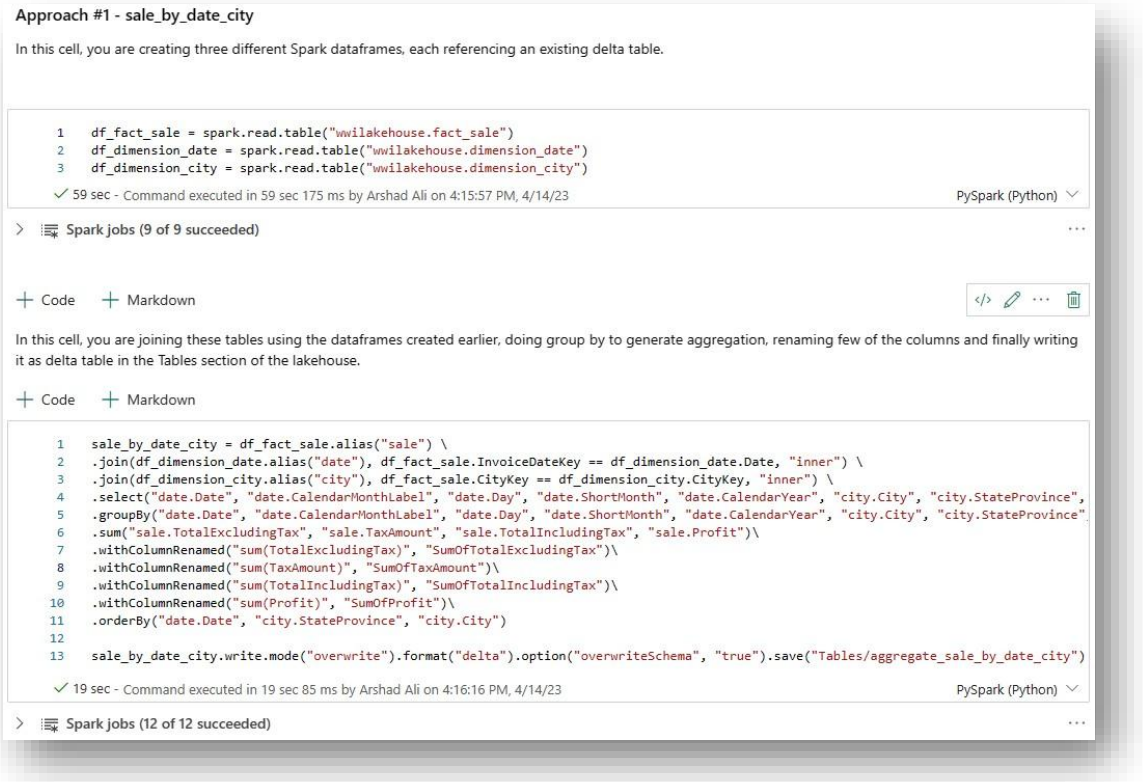
19. An organization might have a group of data engineers working with Scala/Python while there might be other groups of data engineers preferring to work with SQL (Spark SQL or T-SQL) and all these while working on the

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.



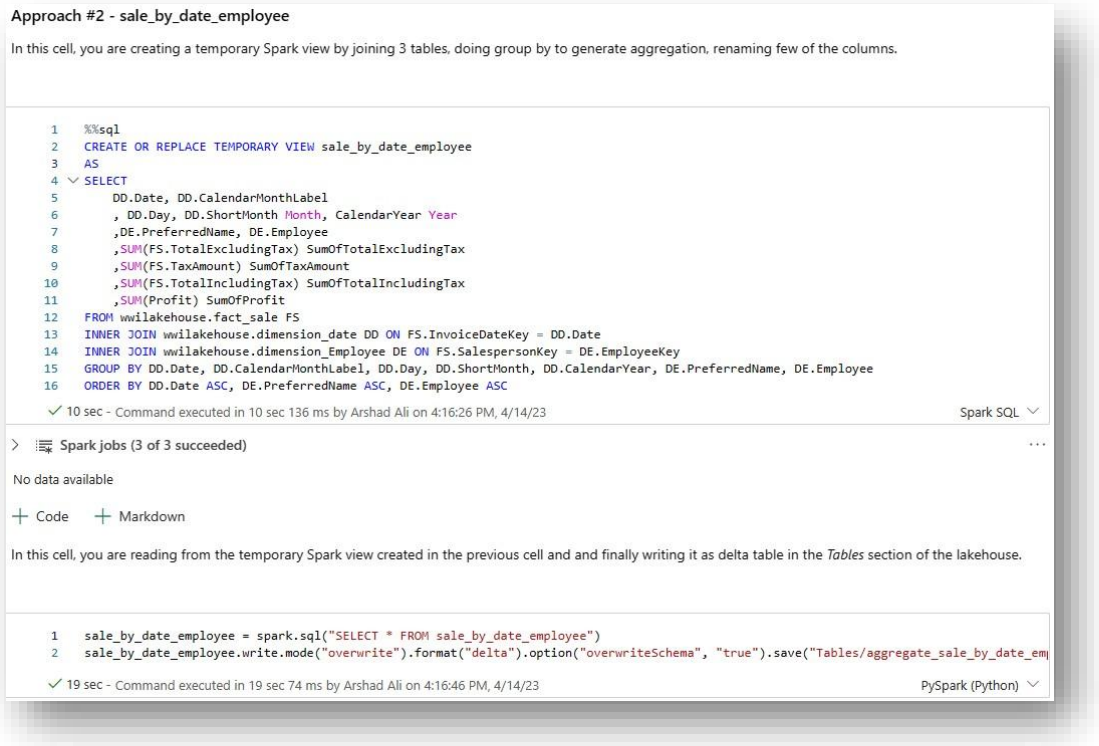

```
df_fact_sale = spark.read.table("wwilakehouse.fact_sale")
df_dimension_date = spark.read.table("wwilakehouse.dimension_date")
df_dimension_city = spark.read.table("wwilakehouse.dimension_city")

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")
```

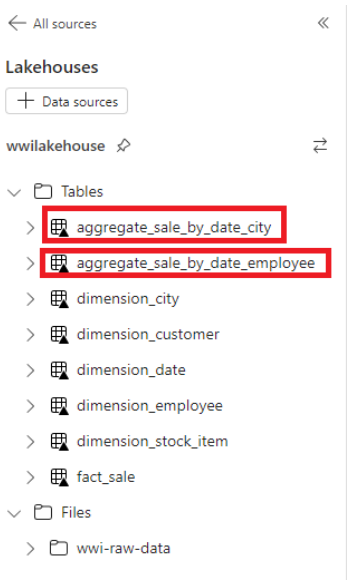
21. **Approach #2** – Use Spark SQL to join and aggregates data for generating business aggregates. In the below code, you are creating a temporary Spark view by joining 3 tables, doing group by to generate aggregation, renaming few of the columns. Finally, you are reading from the temporary Spark view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




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


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. 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.

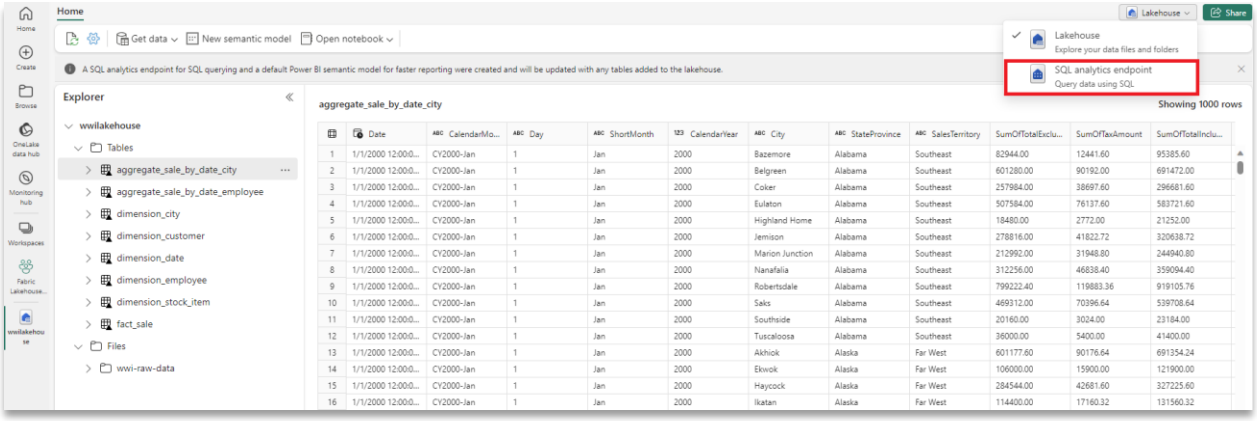
Building a report

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.

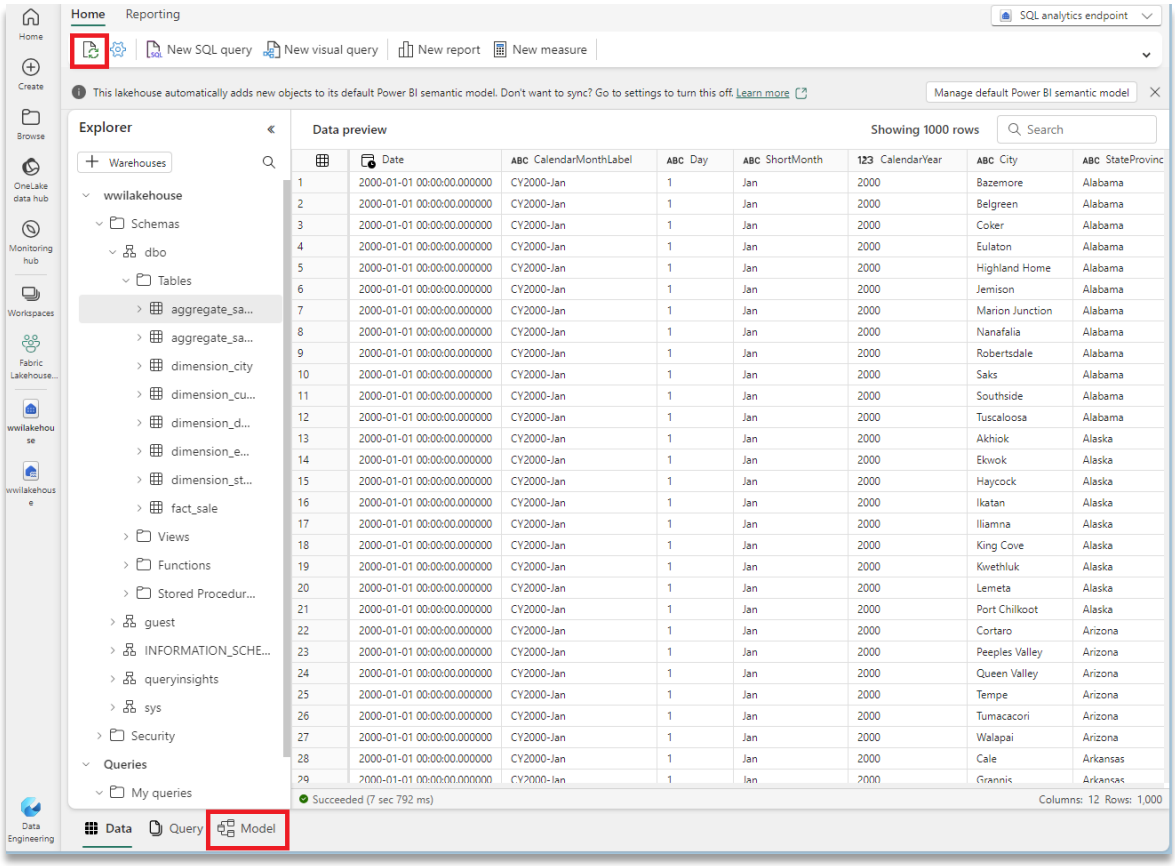
In traditional DirectQuery mode, the Power BI engine queries the data directly from the data source every time it is queried and hence query performance depends on the speed data can be retrieved from the data source. It avoids having to copy the data i.e., any changes at the source are immediately reflected 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.

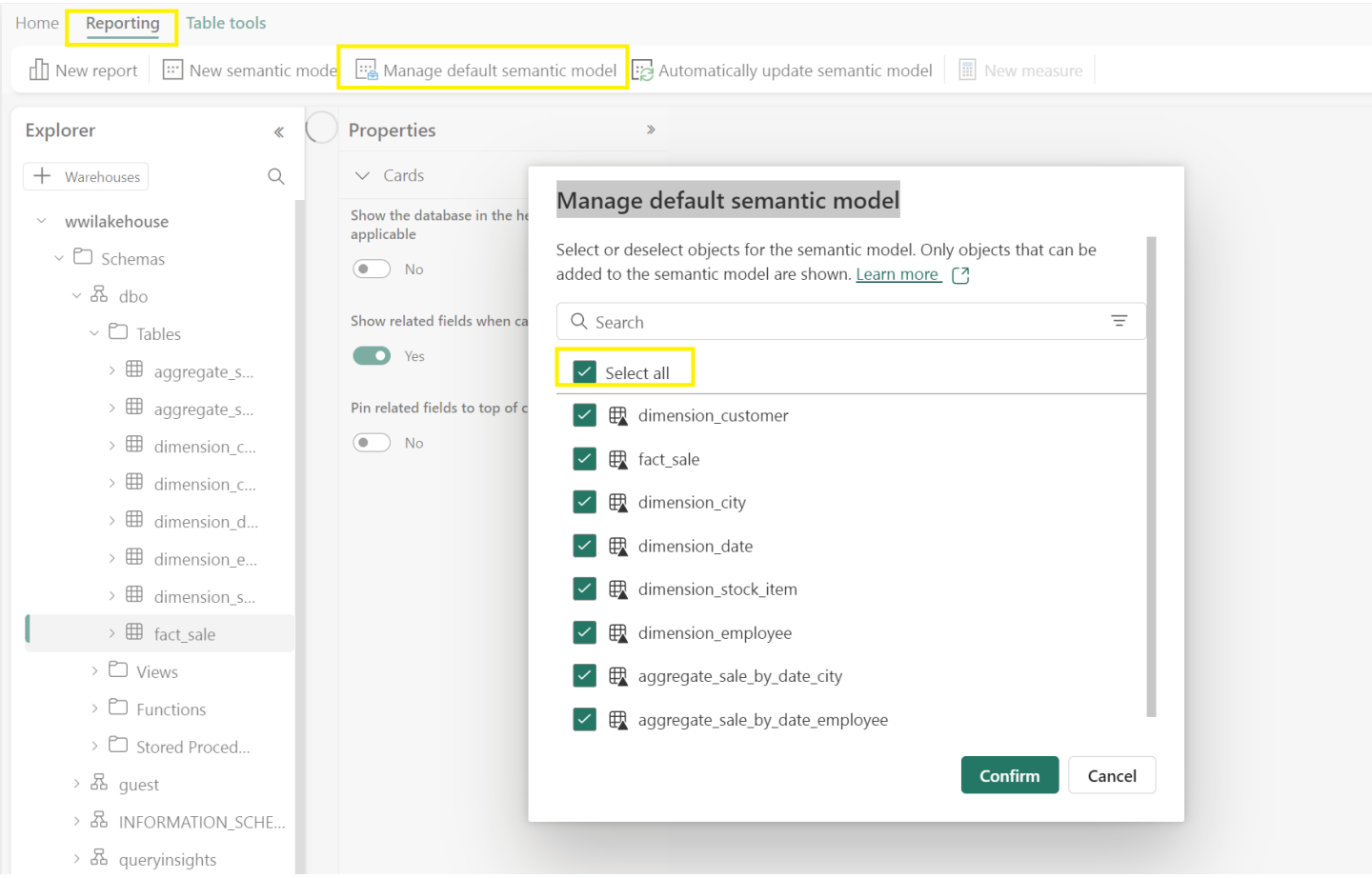
1. Please go the **wwilakehouse** lakehouse, click **SQL analytics endpoint** under **Lakehouse** on top right of the screen to open warehouse mode of the selected lakehouse.



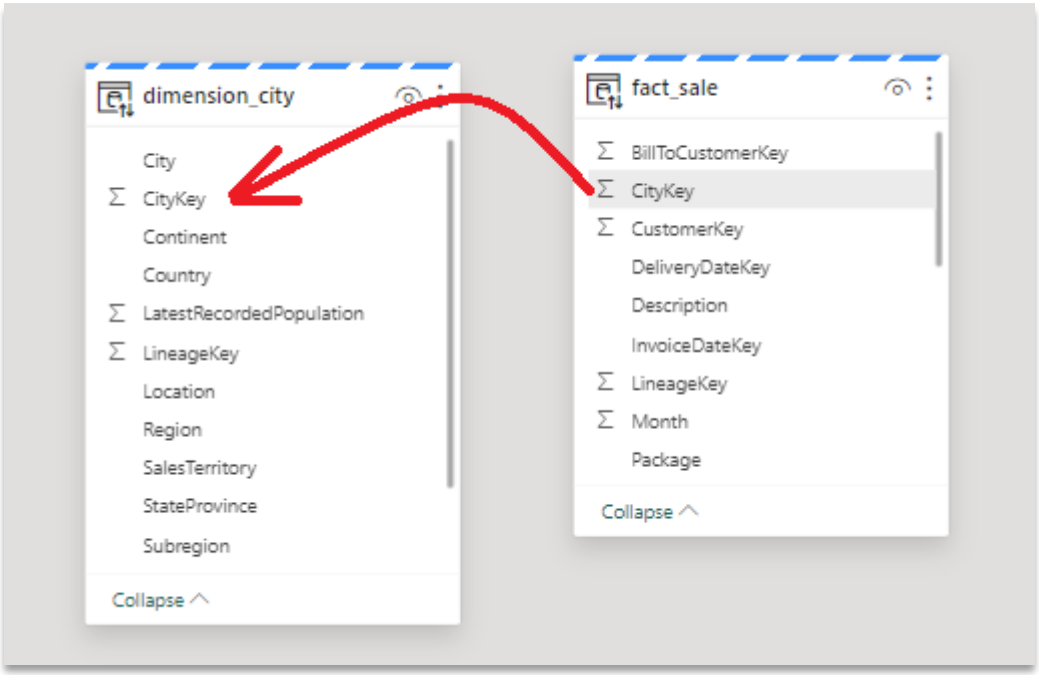
2. Once you are in warehouse mode you should be able to see all the tables you created. If you don't see them yet, please click on the **Refresh** icon at the top. Next, click on **Model** tab at the bottom to open the default Power BI dataset.



3. While in the Model Tab, go to the Reporting tab within the upper ribbon, choose “Manage default semantic model”, select every table, and click confirm.

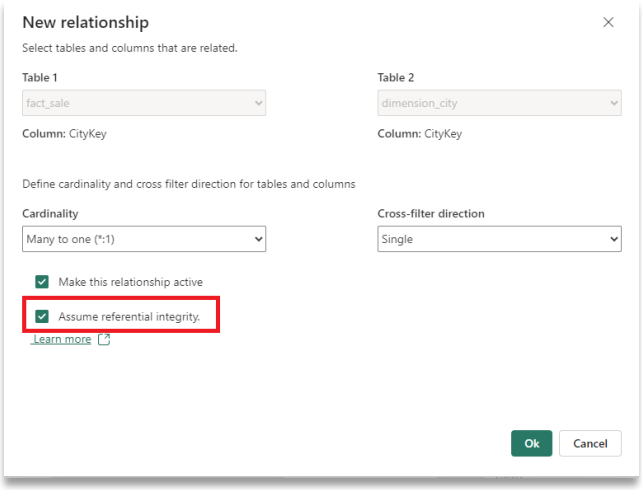


4. For this data model, you need to define the relationship between different tables so that you can create reports and visualizations based on data coming across different tables. From the **fact_sale** table, drag the **CityKey** field and drop it on the **CityKey** field in the **dimension_city** 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.

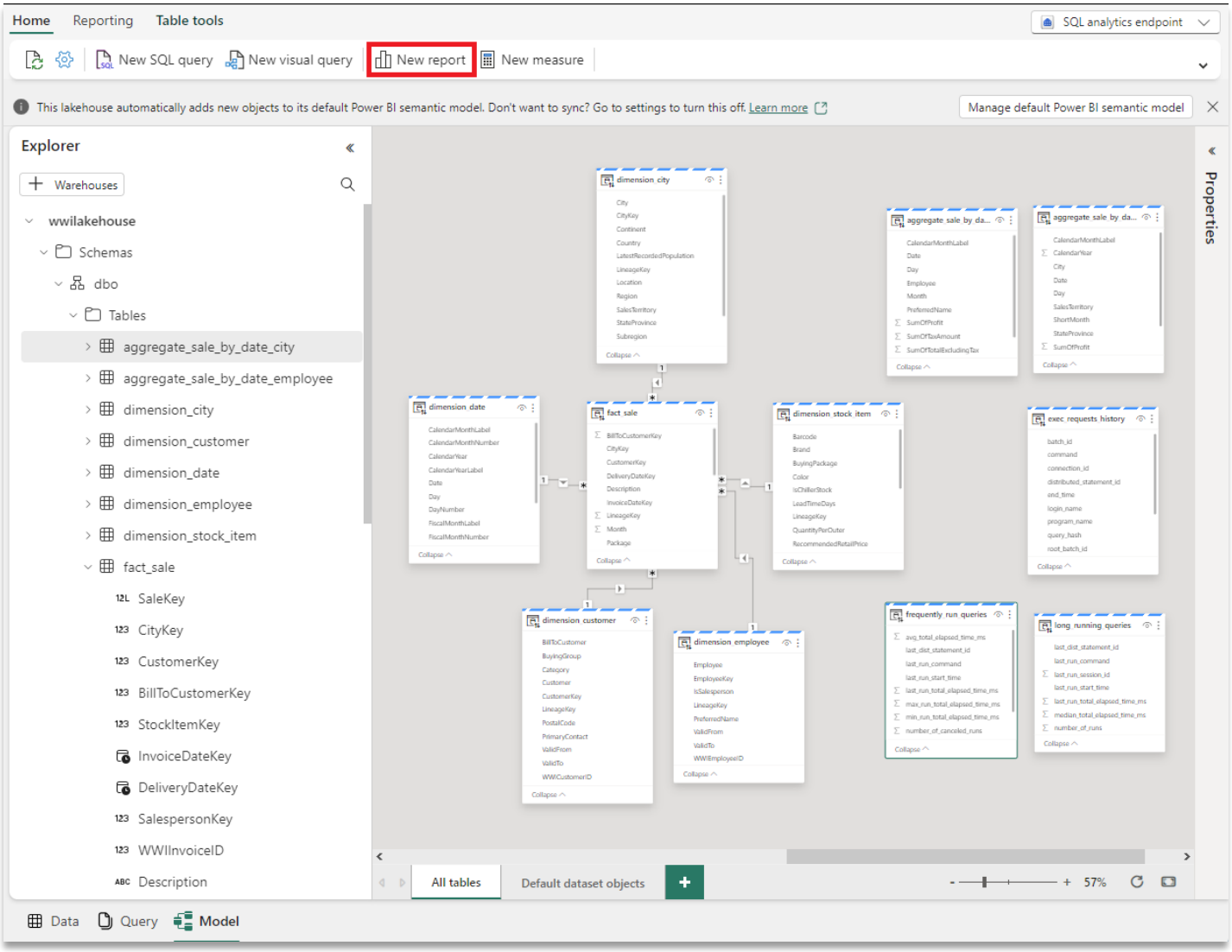


5. On the **New Relationship** settings:
    a. Table 1 will be populated with fact_sale and the column of CityKey.

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

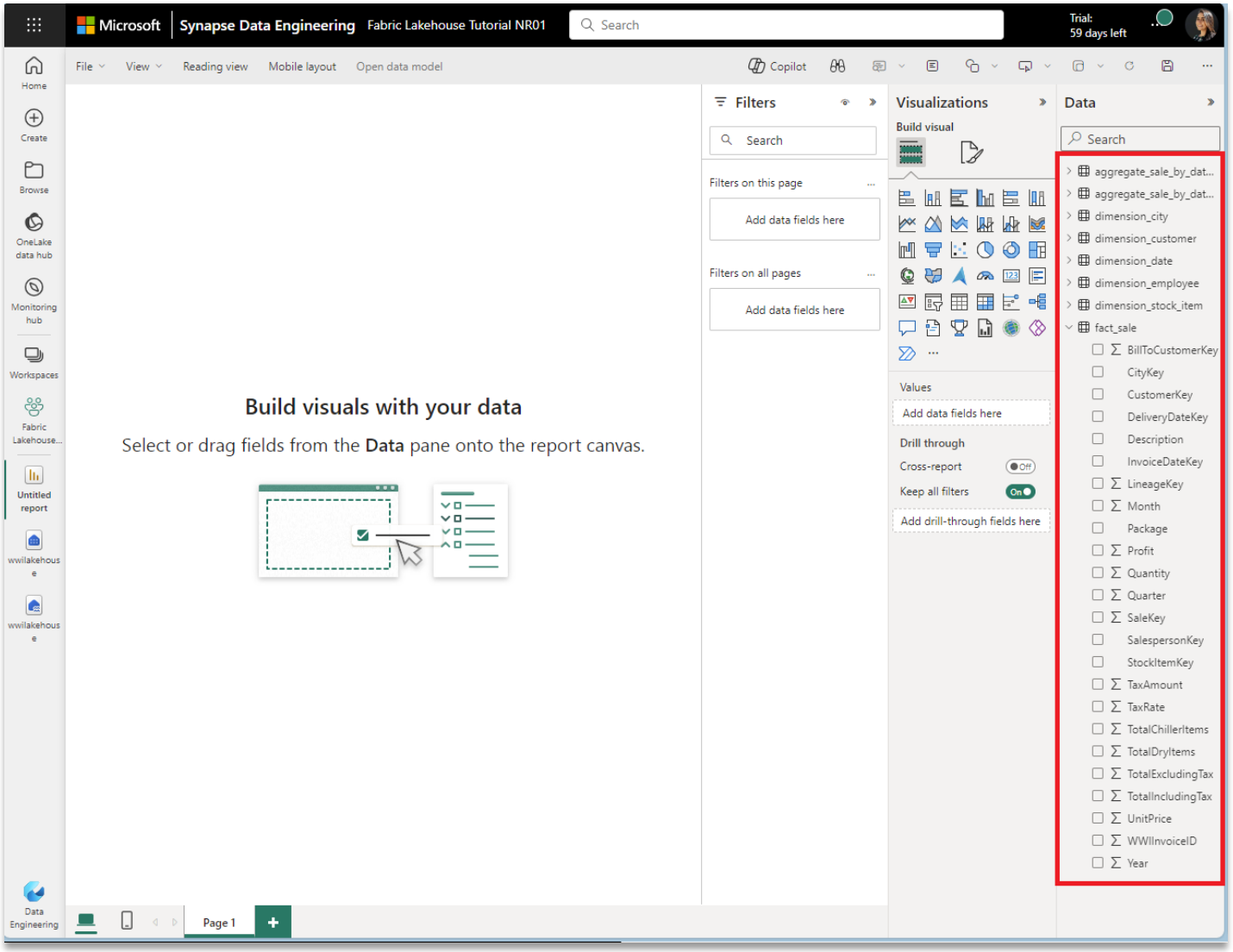


h. Similarly, you need to add these relationships as well:
   - StockItemKey(fact_sale) – StockItemKey(dimension_stock_item)
   - Salespersonkey(fact_sale) – EmployeeKey(dimension_employee)
   - CustomerKey(fact_sale) – CustomerKey(dimension_customer)
   - InvoiceDateKey(fact_sale) – Date(dimension_date)

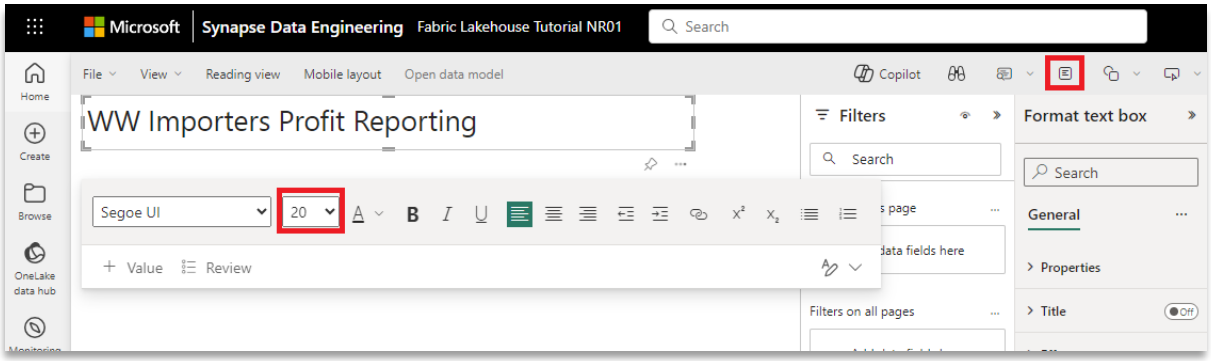

Home
Reporting
Table tools
SQL analytics endpoint
New SQL query
New visual query
New report
New measure
This lakehouse automatically adds new objects to its default Power BI semantic model. Don't want to sync? Go to settings to turn this off. Learn more
Manage default Power BI semantic model
Explorer
Warehouses
wwilakehouse
Schemas
dbo
Tables
aggregate_sale_by_date_city
aggregate_sale_by_date_employee
dimension_city
dimension_customer
dimension_date
dimension_employee
dimension_stock_item
fact_sale
SaleKey
CityKey
CustomerKey
BillToCustomerKey
StockItemKey
InvoiceDateKey
DeliveryDateKey
SalespersonKey
WWIInvoiceID
Description
Properties
All tables
Default dataset objects
57%
Data
Query
Model

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.

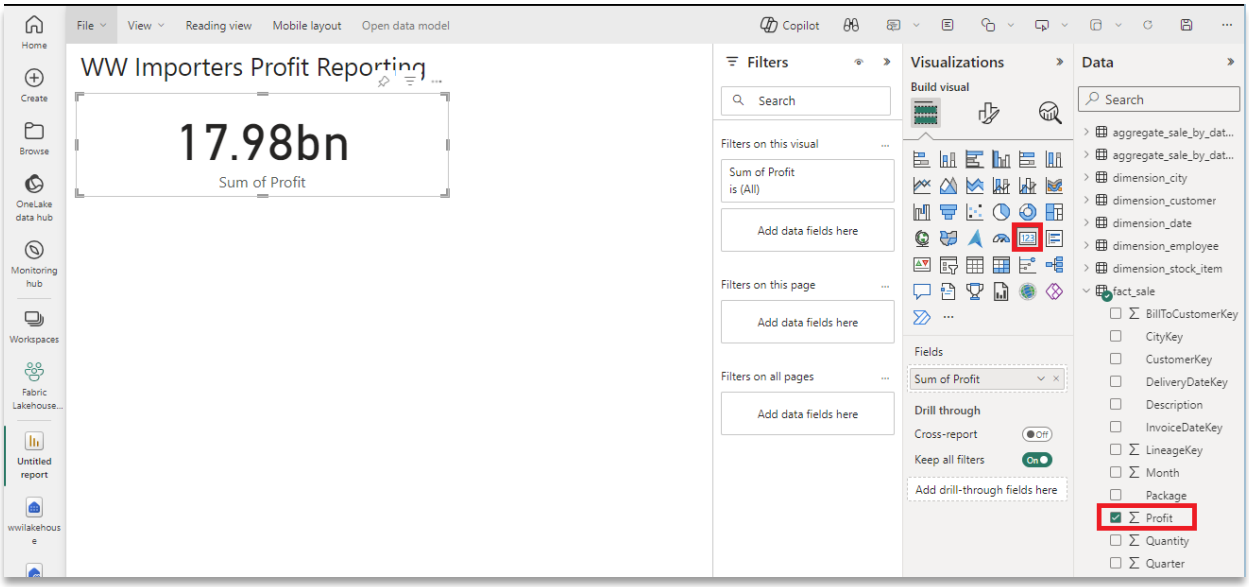
6. On the Power BI report canvas, you can create reports to meet your business requirements by dragging required columns from the **Data** pane to the canvas and using one or more of available visualizations.



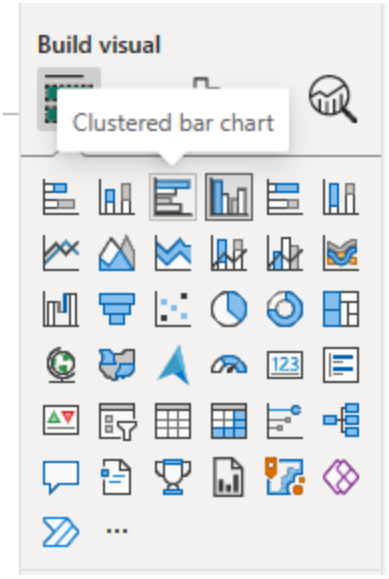
7. Add a title:
    a. In the Ribbon, click the Text Box icon
    b. Type in **WW Importers Profit Reporting**
    c. Highlight the text and increase size to 20 and place in the upper left of the report page



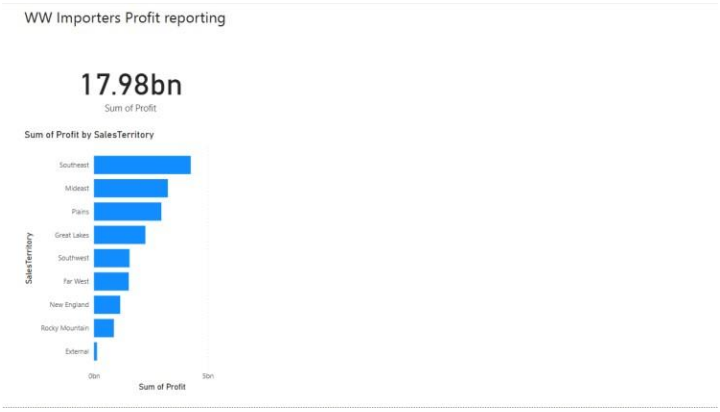
8. Add a Card
    a. On the **Data** pane, expand **fact_sales** and check the box next to **Profit**. This will create a column chart and add the field to the Y-axis.
    b. With the bar chart selected, click the Card visual in the visualization pane. This will convert the visual to a card.
    c. Place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



9. Add a Bar chart
    a. On the **Data** pane, expand **fact_sales** and check the box next to **Profit**. This will create a column chart and add the field to the Y-axis.
    b. On the **Data** pane, expand **dimension_city** and check the box for **SalesTerritory** This will add the field to the X-axis.
    c. 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.

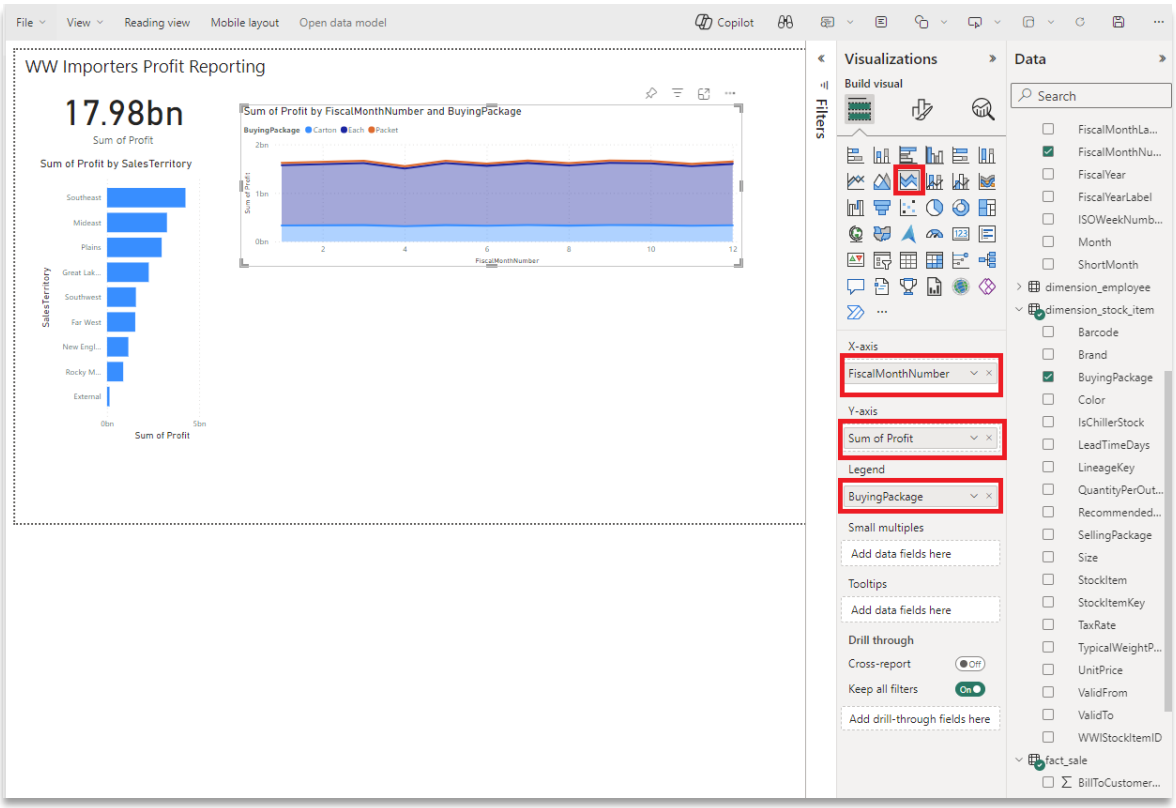


    d. Resize the Bar chart to fill in the area under the title and Card

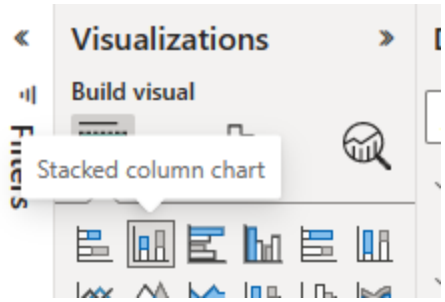


10. Click anywhere on the blank canvas (or press the Esc key) so the bar chart visual is no longer selected.
11. Build a stacked area chart visual:
    a. On the **Visualizations** pane, select the **Stacked area chart** visual.

    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 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.

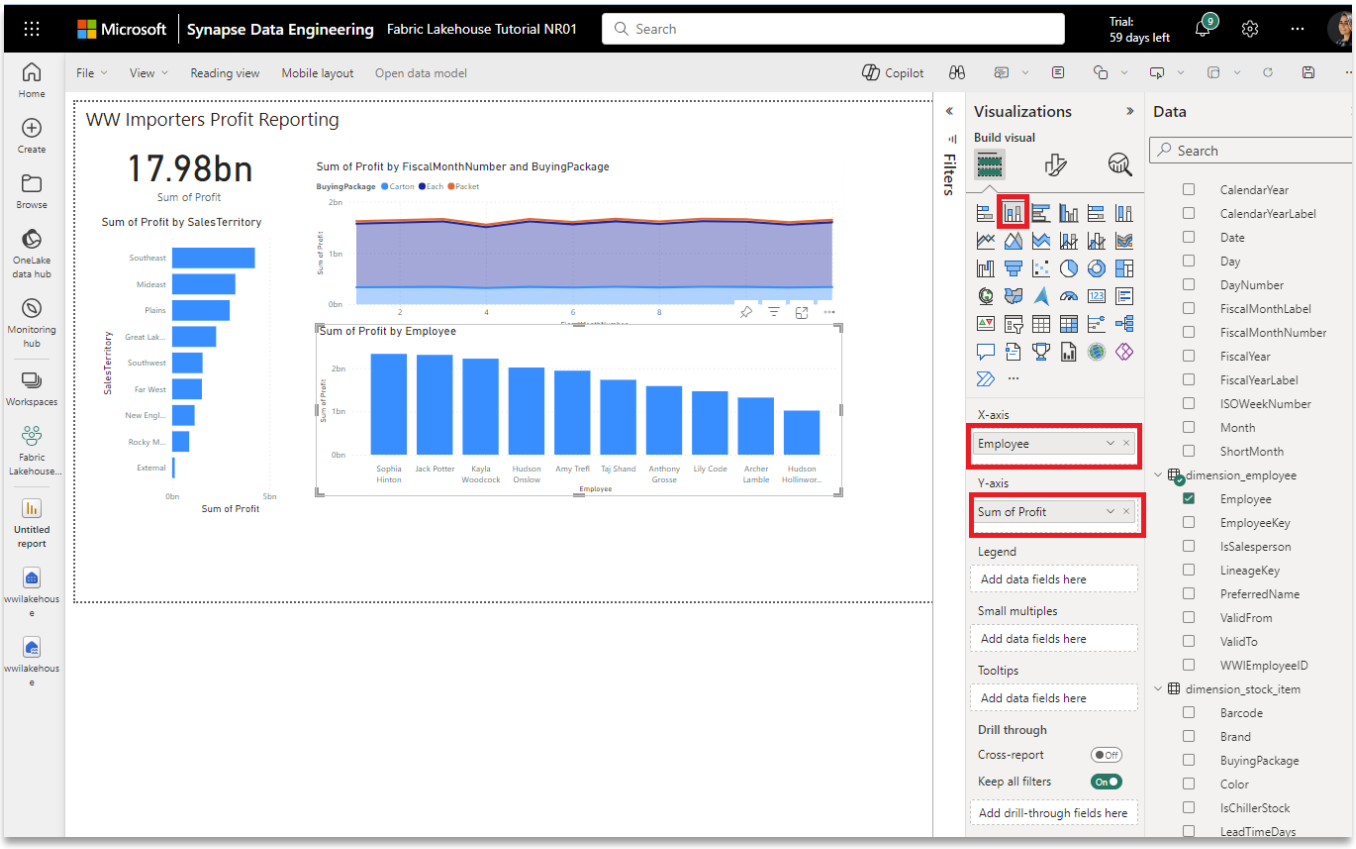


12. Click anywhere on the blank canvas (or press the Esc key) so the stacked area chart visual is no longer selected.
13. Build a column chart:
    a. On the **Visualizations** pane, select the **Stacked column chart** visual.

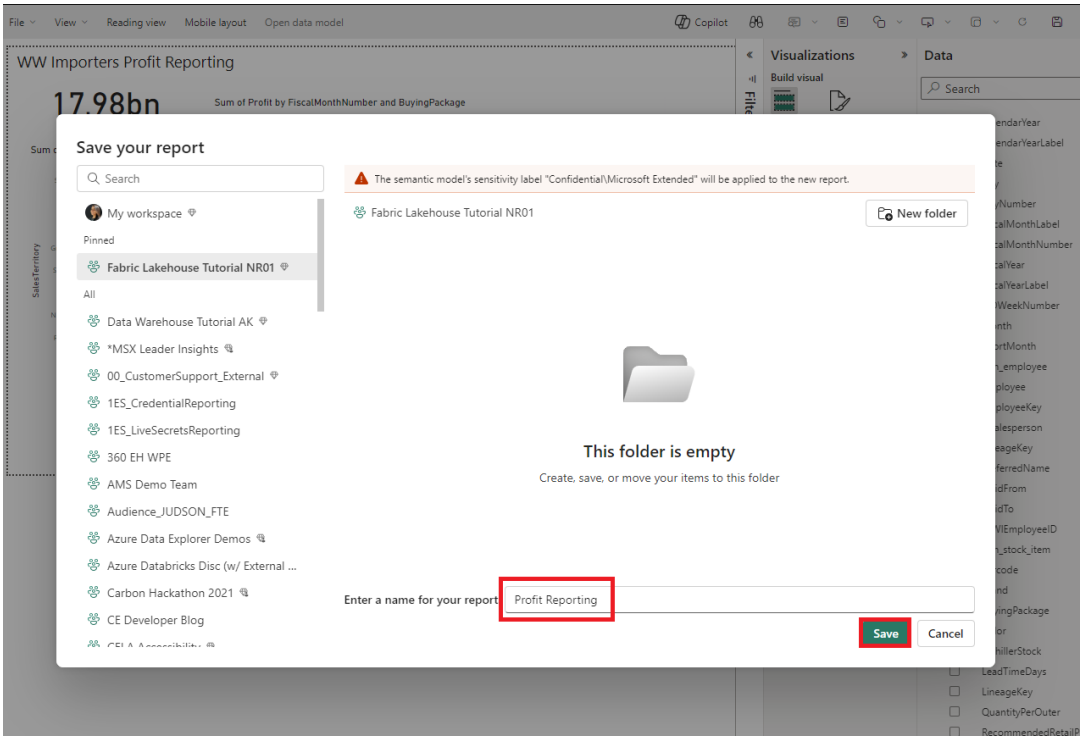


b. On the **Data** pane, expand **fact_sales** and check the box next to **Profit**. This will add the field to the Y-axis.

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



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

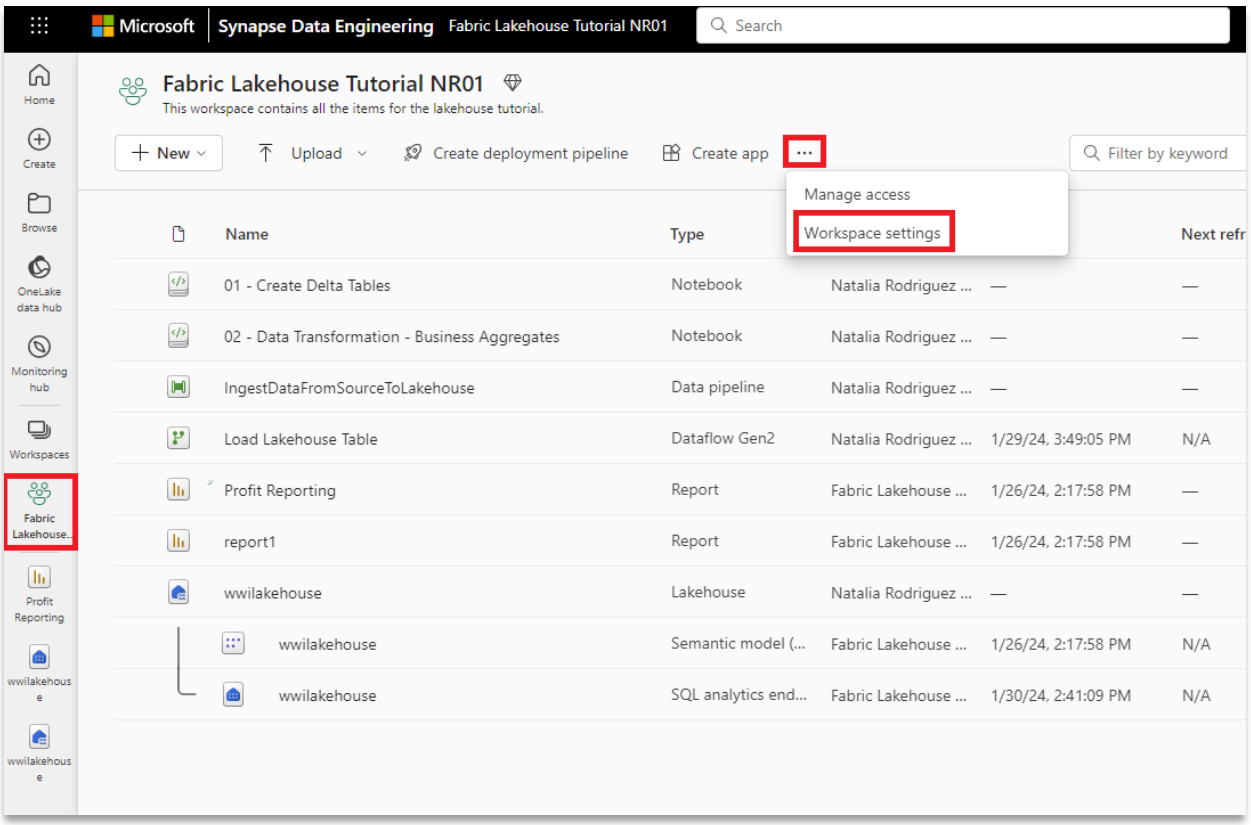

File
View
Reading view
Mobile layout
Open data model
Copilot
WW Importers Profit Reporting
17.98bn
Sum of Profit by FiscalMonthNumber and BuyingPackage
Visualizations
Build visual
Data
Search
Save your report
Search
My workspace
Pinned
Fabric Lakehouse Tutorial NR01
All
Data Warehouse Tutorial AK
*MSX Leader Insights
00_CustomerSupport_External
1ES_CredentialReporting
1ES_LiveSecretsReporting
360 EH WPE
AMS Demo Team
Audience_JUDSON_FTE
Azure Data Explorer Demos
Azure Databricks Disc (w/ External ...
Carbon Hackathon 2021
CE Developer Blog
The semantic model's sensitivity label "Confidential\Microsoft Extended" will be applied to the new report.
Fabric Lakehouse Tutorial NR01
New folder
This folder is empty
Create, save, or move your items to this folder
Enter a name for your report
Profit Reporting
Save
Cancel

Module 4: Clean up resources

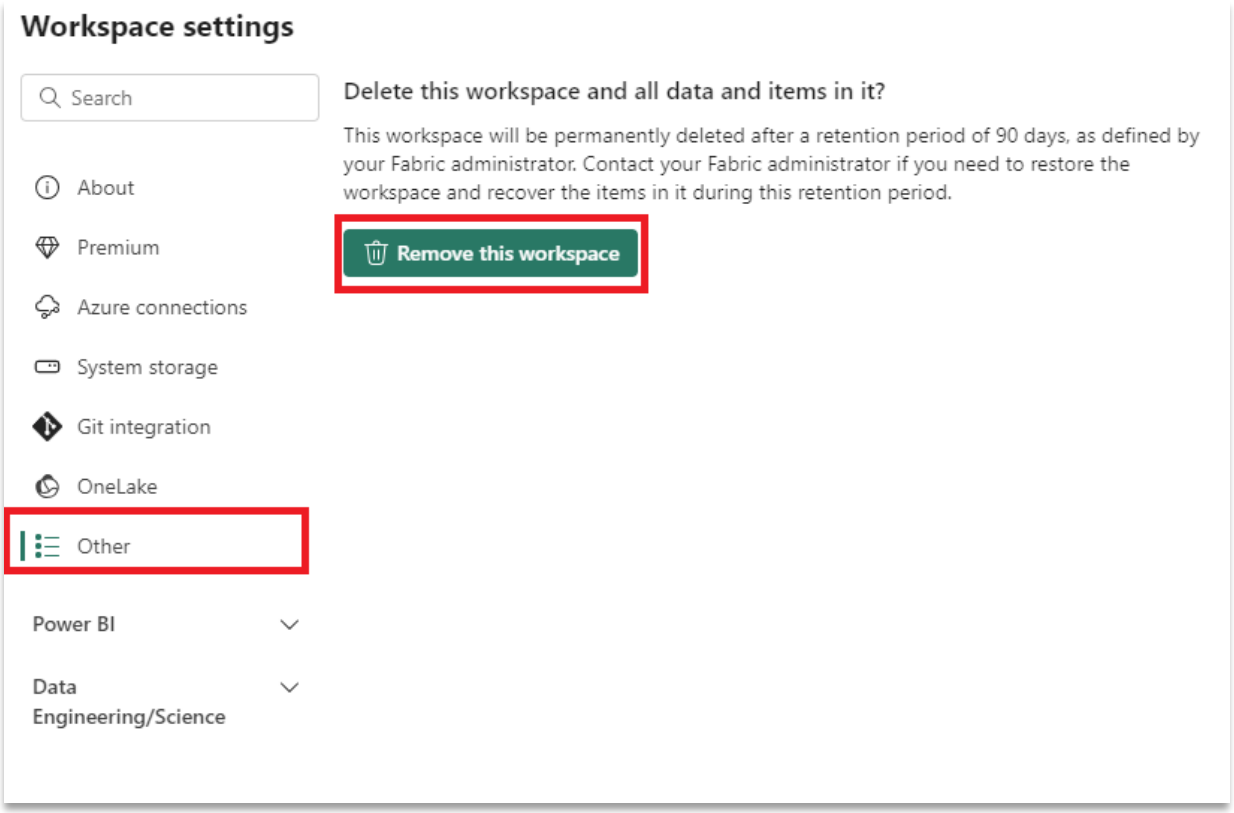
You can delete individual reports, pipelines, warehouses, and other items or remove the entire workspace.

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.

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



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



4. Select **Delete** on the warning.