

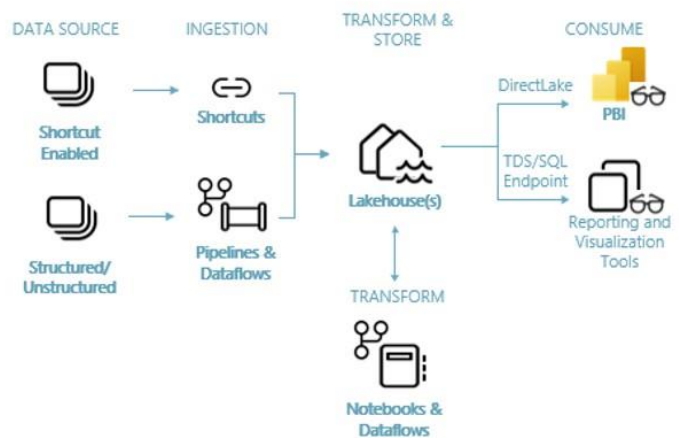
Lakehouse Tutorial

Lakehouse

Published: Jan 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	11
Create a lakehouse	11
Data Ingestion.....	13
Building a report.....	23
Module 3: Ingest, Prep and Analyze	25
Data Ingestion.....	25
Data Preparation	33
Building a report.....	46
Module 4: Clean up resources	58

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.

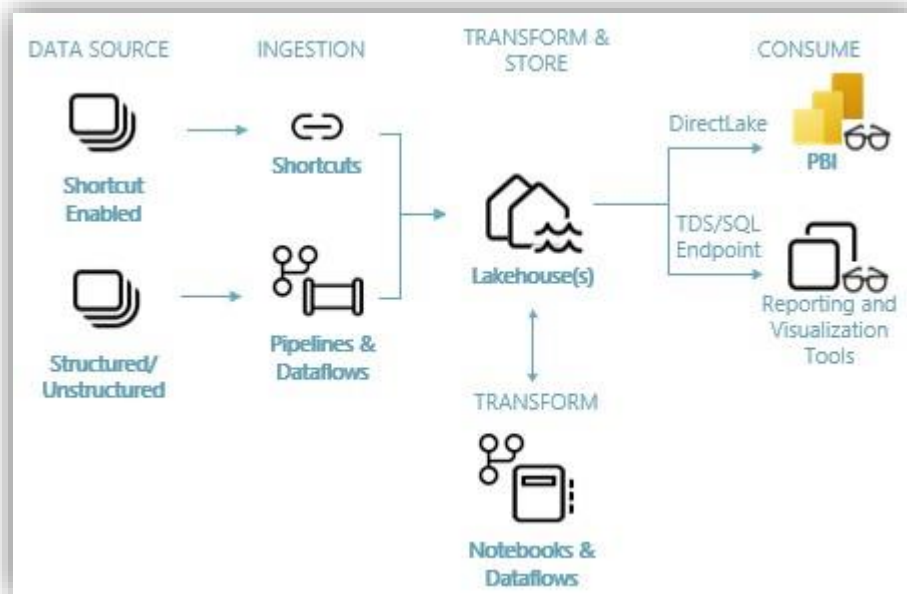
With the flexibility Fabric provides, you can implement either lakehouse or data warehouse architectures or combine these two together to get the best of both the worlds in simple implementation. In this tutorial, you are going to take an example of retail organization and build its lakehouse from start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

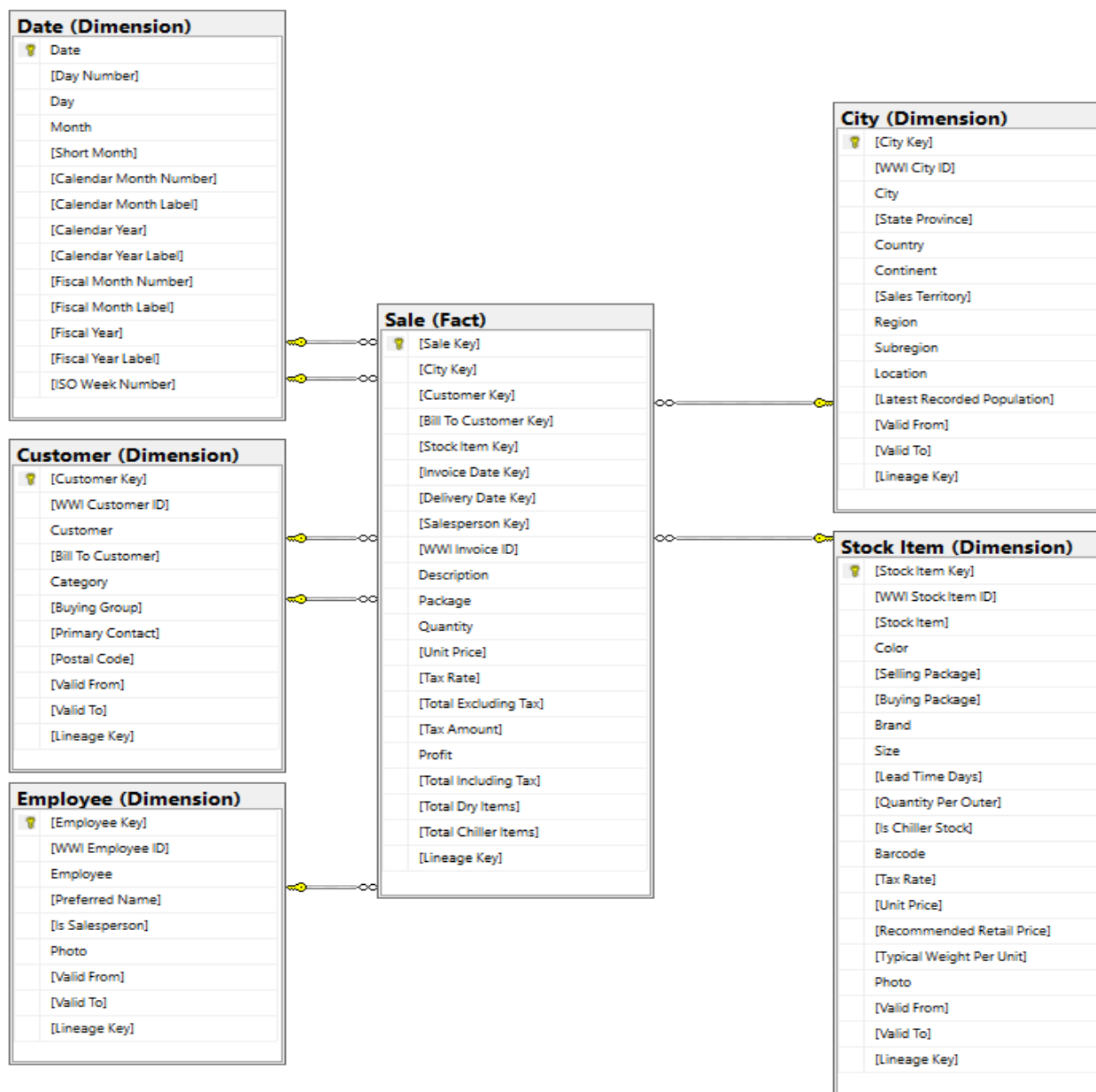
Wide World Importers (WWI) is a wholesale novelty goods importer and distributor operating from the San Francisco Bay area. As a wholesaler, WWI's customers are mostly companies who resell to individuals. WWI sells to retail customers across the United States including specialty stores, supermarkets,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

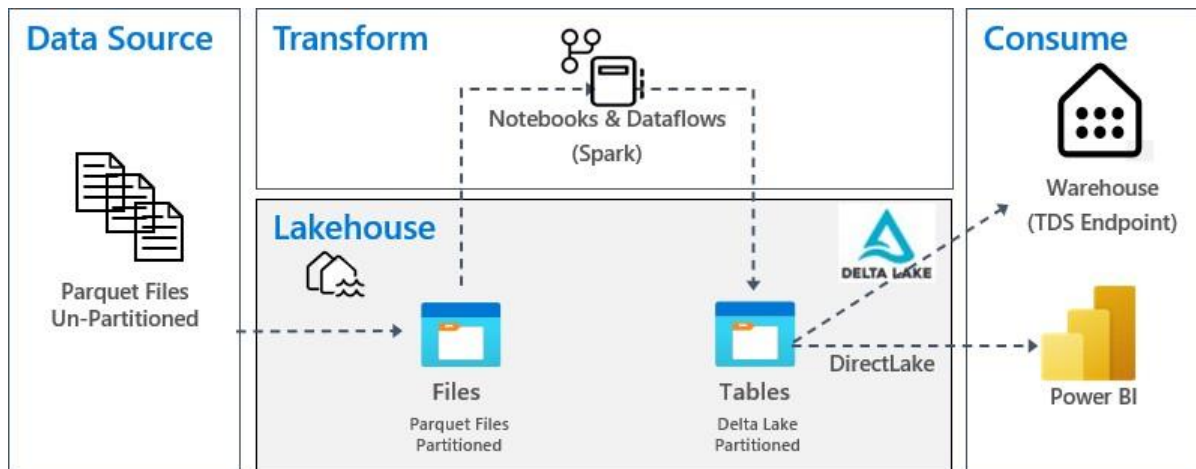
While the WWI dimensional model contains multiple fact tables, for simplicity in explanation we will focus on the Sale Fact table and its related dimensions only, as below, to demonstrate this end-to-end lakehouse scenario:



End-to-end data and transformation flow

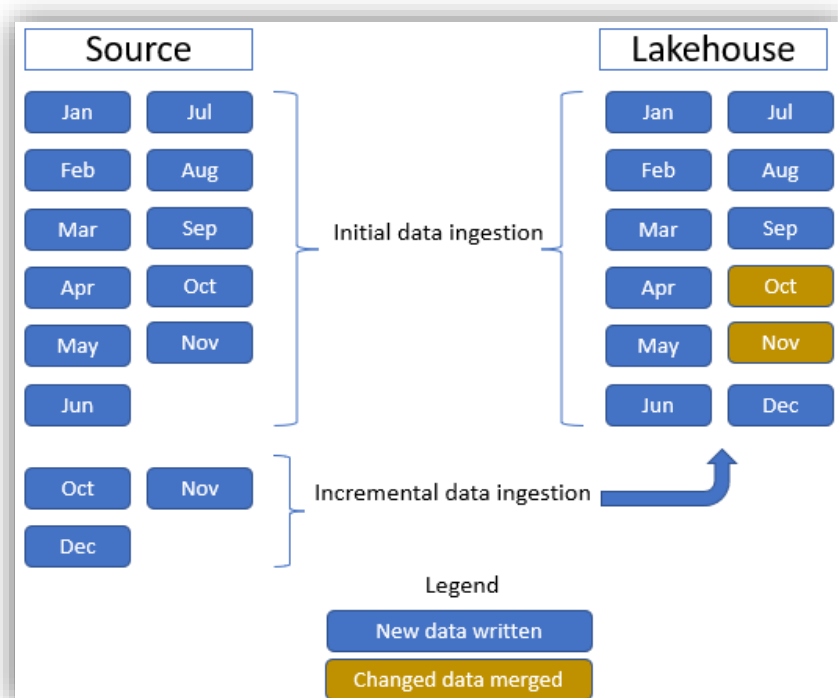
Earlier, we described the [Wide World Importers \(WWI\) sample data](#) which we are going to leverage in building this end to end lakehouse. For this implementation, this sample data is in an Azure Data storage account in Parquet file format for

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.

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.

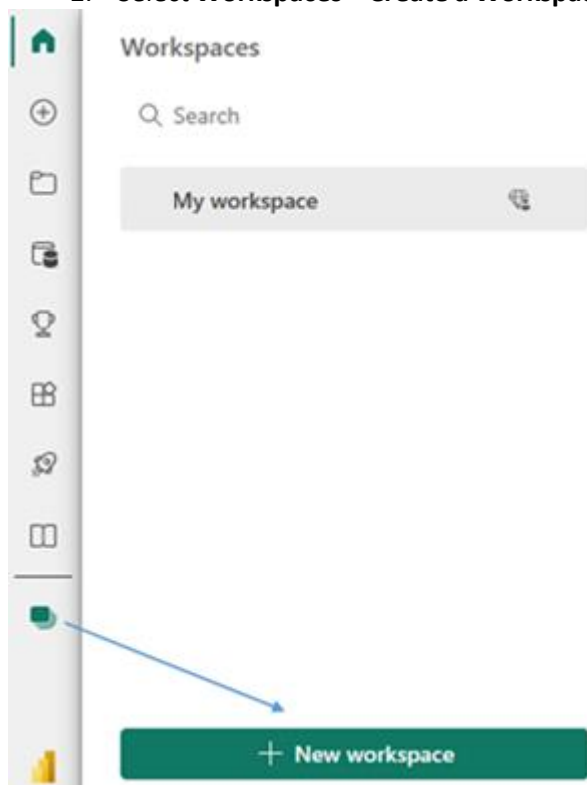
Module 1: Getting Started

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

Fabric Lakehouse Tutorial - AA

Available

Description

This workspace contains all the items for the lakehouse tutorial.

Domain (preview) ⓘ

Assign to a domain (optional)



[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 x Enter users and groups

License mode ⓘ


☐ Pro

Select Pro to use basic Power BI features and collaborate on reports, dashboards, and scorecards. To access a Pro workspace, users need Pro per-user licenses. [Learn more](#) 


☒ 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

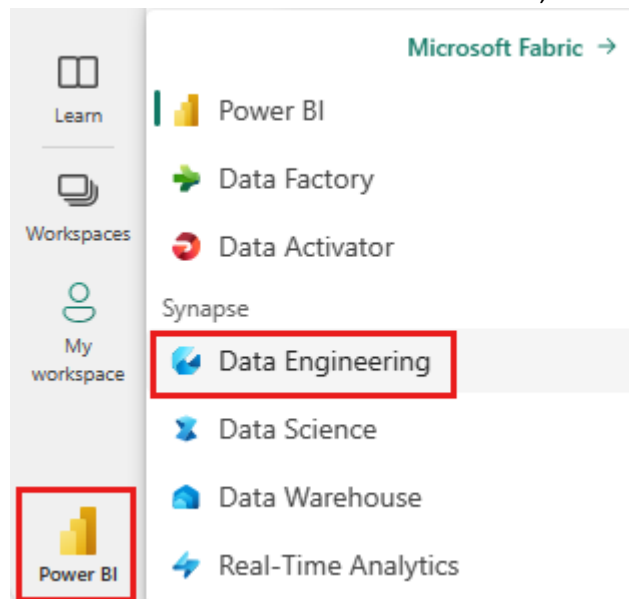
4. Select **Apply**. The workspace will be created and opened.

Module 2: Build your first Lakehouse in Fabric

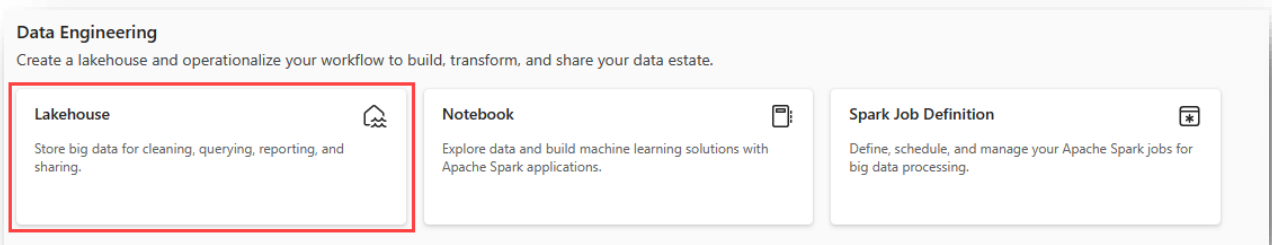
The intent of this module is to quickly build end to end journey of building a lakehouse, ingesting data for a table, applying transformation whenever required and then using ingested data into the lakehouse delta table for creating reports.

Create a lakehouse

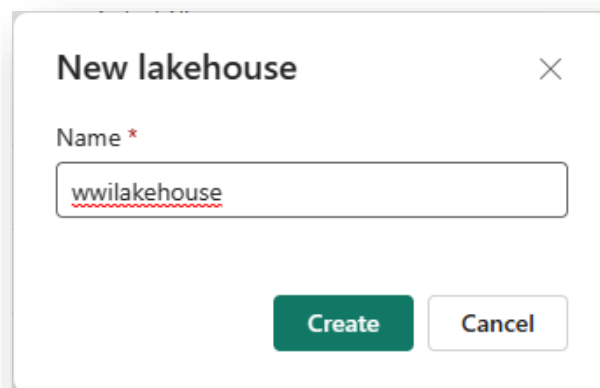
1. In the [Power BI service](#) select **Workspaces** in the left-hand menu.
2. Search for your workspace by typing in the search textbox at the top and click on your workspace to open it.
3. From the workload switcher located at the bottom left of the screen, select **Data engineering**.



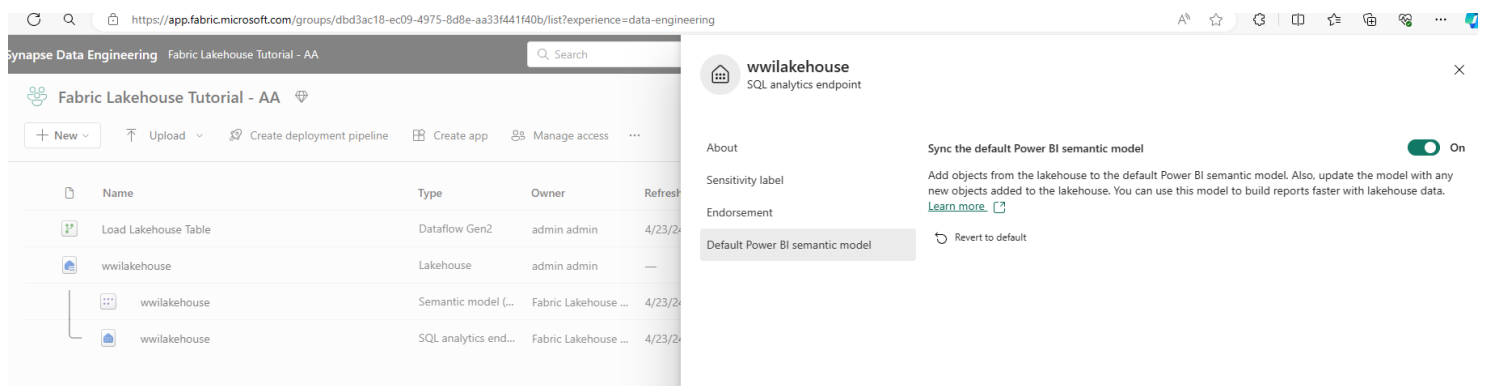
4. In the **Data Engineering** section, select **Lakehouse** to create a lakehouse.



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

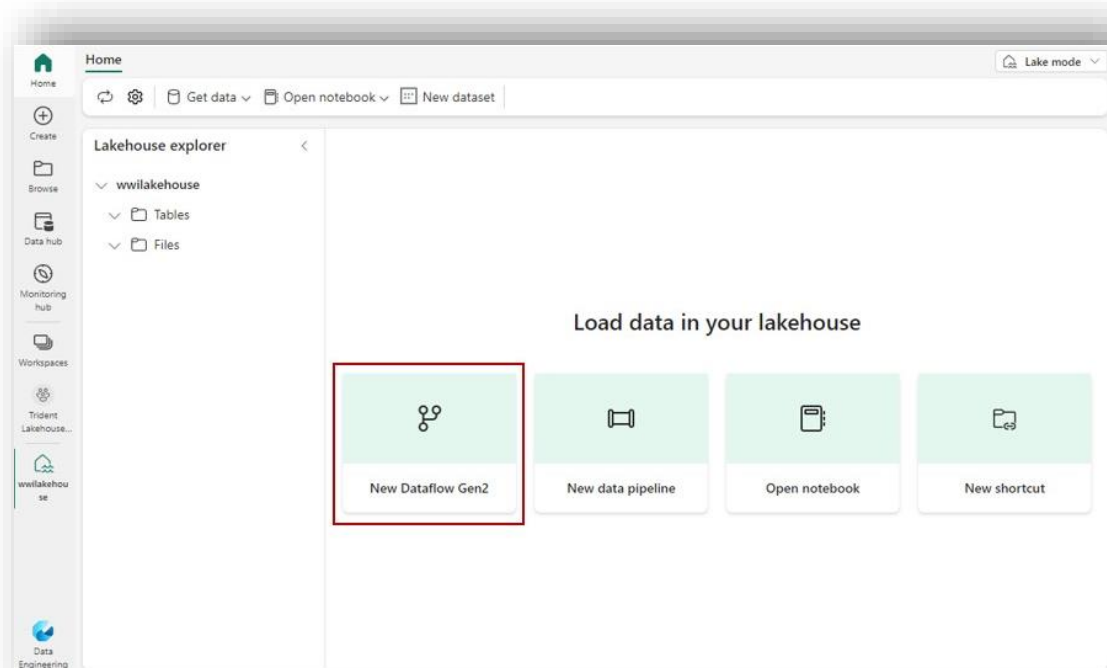


6. Click **Create**. The new lakehouse will be created and automatically opened.
7. On **wwilakehouse** SQL analytics endpoint, enable 'Sync' the default Power BI semantics model.



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, connect **Azure Data Storage Gen2**

Get data

Connect to data source

Azure Data Lake Storage Gen2
Azure
[Learn more](#)

Connection settings

URL *

Data view

☒ File system view
☐ CDM folder view

Connection credentials

Connection

Connection name

Data gateway

Authentication kind

SAS token

Privacy Level

Back Cancel Next

Enter into URL: <https://fabricvbdadlspublic.dfs.core.windows.net/fabricfoundation>
https://fabricvbdadlspublic.dfs.core.windows.net/fabricfoundation/lakehouse/dimension_customer.csv

Enter SAS token: instructor will share

Get data

Preview folder data

https://fabricvbdadlspublic.dfs.core.windows.net/fabricfoundation

Content	Name	Extension	Date accessed	Date modified	Date created	Attributes	Folder Path
[Binary]	dimension_customer.csv	.csv	null	4/23/2024, 10:40:05 PM	null	[Record]	https://fabricvbdadlspublic.dfs.core.windows.net/fabricfoundation/lakehouse/

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

Preview file data

URL: https://microsoft-my.sharepoint.com/personal/arali_microsoft_com/Documents/Apps/Microsoft Power Query/Uploaded Files/dimension_customer 7.csv

File origin: 65001: Unicode (UTF-8) Delimiter: Comma Data type detection: Based on first 200 rows

CustomerKey	WWCustomerID	Customer	BillToCustomer	Category	BuyingGroup	PrimaryContact	PostalCode	ValidFrom	ValidTo
0	0	Unknown	N/A	N/A	N/A	N/A	N/A	1/1/2013, 12:00:00 AM	1/1/2025
234	433	Wingtip Toys (Baiko, OK)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Adirake Saenamuang	90584	1/1/2013, 12:00:00 AM	1/1/2025
235	434	Wingtip Toys (Lime Lake, NY)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Tarja Penttila	90667	1/1/2013, 12:00:00 AM	1/1/2025
236	435	Wingtip Toys (Teutopolis, IL)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Devendranath Huq	90269	1/1/2013, 12:00:00 AM	1/1/2025
237	436	Wingtip Toys (Gargatha, VA)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Enes Olsson	90752	1/1/2013, 12:00:00 AM	1/1/2025
238	437	Wingtip Toys (Cadogan, PA)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Vanja Grigic	90357	1/1/2013, 12:00:00 AM	1/1/2025
239	438	Wingtip Toys (Lucasville, OH)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Dayaram Mishra	90391	1/1/2013, 12:00:00 AM	1/1/2025
240	439	Wingtip Toys (Cylon, WI)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Mina Omidzadeh	90266	1/1/2013, 12:00:00 AM	1/1/2025
241	440	Wingtip Toys (Asher, OK)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Kadir Usenuly	90683	1/1/2013, 12:00:00 AM	1/1/2025
242	441	Wingtip Toys (Keosauqua, IA)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Kajsa Jakobsson	90281	1/1/2013, 12:00:00 AM	1/1/2025
243	442	Wingtip Toys (Homer City, PA)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Rahul Ghate	90713	1/1/2013, 12:00:00 AM	1/1/2025
244	443	Wingtip Toys (Benville, MI)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Lien Binh	90628	1/1/2013, 12:00:00 AM	1/1/2025
245	444	Wingtip Toys (Tea, SD)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Divyendu Chakraborty	90641	1/1/2013, 12:00:00 AM	1/1/2025
246	445	Wingtip Toys (Dacono, CO)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	An Dung Ngo	90289	1/1/2013, 12:00:00 AM	1/1/2025
247	446	Wingtip Toys (Saint Landry, LA)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Chetana Kamath	90033	1/1/2013, 12:00:00 AM	1/1/2025
248	447	Wingtip Toys (Coin, IA)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Premwadee Saowaluk	90458	1/1/2013, 12:00:00 AM	1/1/2025
249	448	Wingtip Toys (Salt Wells, NV)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Jae-Hwa Min	90058	1/1/2013, 12:00:00 AM	1/1/2025
250	449	Wingtip Toys (Delray, WV)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Catalina Nechita	90317	1/1/2013, 12:00:00 AM	1/1/2025
251	450	Wingtip Toys (Mount Summit, IN)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Constanza Laureano	90142	1/1/2013, 12:00:00 AM	1/1/2025

Back Cancel Add table using examples **Create**

- 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.

Power Query

Home Transform Add column View Help

Get Enter data Manage connections Options Refresh Advanced editor Manage Query Add data Choose Remove Keep Remove Filter Suggested Split Group Data type Whole number Merge queries Append queries Map to entity Copilot Export template

Queries [1] dimension_customer

Table.TransformColumnTypes(*Promoted headers*, {"CustomerKey", Int64.Type}, {"BillToCustomer", Int64.Type}, {"Customer", type text}, {"Category", type text})

CustomerKey	WWCustomerID	Customer	BillToCustomer	Category	BuyingGroup	PrimaryContact	PostalCode	ValidFrom	ValidTo
0	0	Unknown	N/A	N/A	N/A	N/A	N/A	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
234	433	Wingtip Toys (Baiko, OK)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Adirake Saenamuang	90584	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
235	434	Wingtip Toys (Lime Lake, NY)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Tarja Penttila	90667	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
236	435	Wingtip Toys (Teutopolis, IL)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Devendranath Huq	90269	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
237	436	Wingtip Toys (Gargatha, VA)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Enes Olsson	90752	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
238	437	Wingtip Toys (Cadogan, PA)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Vanja Grigic	90357	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
239	438	Wingtip Toys (Lucasville, OH)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Dayaram Mishra	90391	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
240	439	Wingtip Toys (Cylon, WI)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Mina Omidzadeh	90266	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
241	440	Wingtip Toys (Asher, OK)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Kadir Usenuly	90683	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
242	441	Wingtip Toys (Keosauqua, IA)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Kajsa Jakobsson	90281	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
243	442	Wingtip Toys (Homer City, PA)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Rahul Ghate	90713	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
244	443	Wingtip Toys (Benville, MI)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Lien Binh	90628	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
245	444	Wingtip Toys (Tea, SD)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Divyendu Chakraborty	90641	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
246	445	Wingtip Toys (Dacono, CO)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	An Dung Ngo	90289	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
247	446	Wingtip Toys (Saint Landry, LA)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Chetana Kamath	90033	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
248	447	Wingtip Toys (Coin, IA)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Premwadee Saowaluk	90458	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
249	448	Wingtip Toys (Salt Wells, NV)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Jae-Hwa Min	90058	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
250	449	Wingtip Toys (Delray, WV)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Catalina Nechita	90317	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
251	450	Wingtip Toys (Mount Summit, IN)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Constanza Laureano	90142	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
252	451	Wingtip Toys (Smart, WI)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Aakriti Bhandari	90071	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
253	452	Wingtip Toys (Lake Davis, CA)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Himani Prabhugadga	90719	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
254	453	Wingtip Toys (Staradovila, IL)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Bryan Helms	90739	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
255	454	Wingtip Toys (Concordia, PR)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Hari Kask	90486	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
256	455	Wingtip Toys (Tift Heights, CA)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Ise Saneli	90304	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
257	456	Wingtip Toys (West Hempstead, NY)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Subash Matondkar	90799	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
258	457	Wingtip Toys (Portales, NM)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Besavenu Anantham	90304	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
259	458	Wingtip Toys (Coker, AK)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Miguel Peaz	90364	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
260	459	Wingtip Toys (Haydenville, OH)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Tobias Gruber	90519	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
261	460	Wingtip Toys (Jalmyri, NM)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Danica Dumina	90741	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM
262	461	Wingtip Toys (Oak, TX)	Wingtip Toys (Head Office)	Novelty Shop	Wingtip Toys	Garen Mulveyee	90707	1/1/2013, 12:00:00 AM	1/1/2025, 12:00:00 AM

Completed (7.11 s) Columns: 11 Rows: 99+

Query settings

Name: **dimension_customer**

Entity type: Custom

Applied steps: Promoted, Changed

Data destination: Lakehouse

Publish

6. If necessary, on the **Connect to data destination** screen, sign into your account. Click **Next**.
7. Navigate to the **wwilakehouse** in your workspace.
8. 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.

The screenshot shows a 'Choose destination target' dialog box. At the top, there are two radio buttons: 'New table' (selected) and 'Existing table'. Below this is a search bar and a list of destinations. The 'wwilakehouse' destination is selected and highlighted. To the right, a message states 'A new table will be created in wwilakehouse'. Below this message is a text input field labeled 'Table name *' containing the text 'dimension_customer'. At the bottom right, there are 'Cancel' and 'Next' buttons, with the 'Next' button being highlighted.

9. 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

<input checked="" type="checkbox"/>	Source		Destination	Type
<input checked="" type="checkbox"/>	CustomerKey	→	CustomerKey	Whole number
<input checked="" type="checkbox"/>	WWICustomerID	→	WWICustomerID	Whole number
<input checked="" type="checkbox"/>	Customer	→	Customer	Text
<input checked="" type="checkbox"/>	BillToCustomer	→	BillToCustomer	Text
<input checked="" type="checkbox"/>	Category	→	Category	Text
<input checked="" type="checkbox"/>	BuyingGroup	→	BuyingGroup	Text

10. 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.

Query settings >

▼ Properties

Name

dimension_customer

Entity type ⓘ
Custom

▼ Applied steps

Source

Promoted h...

ABC
123 Changed co...









▼ Data destination

Lakehouse

Step

Publish

11. 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**.
12. 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.

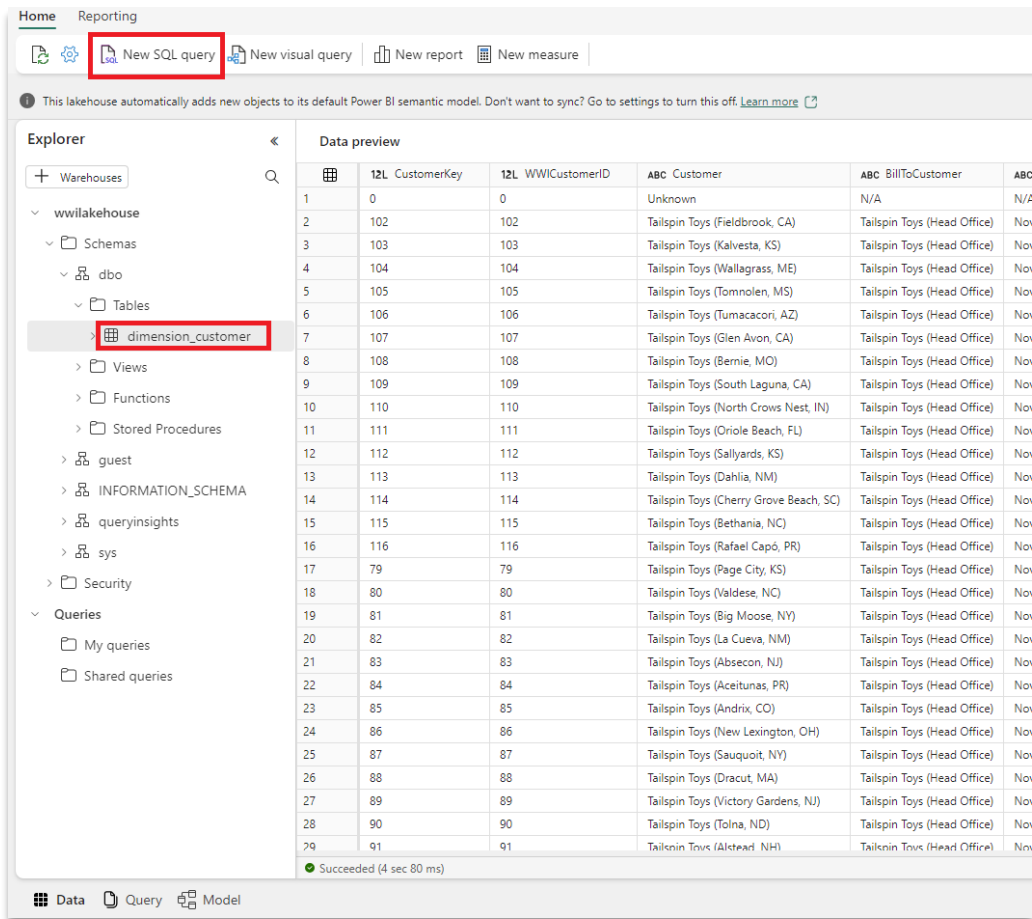
+ New ▾		↑ Upload ▾	🚀 Create deployment pipeline	📦 Create app
		Name		Type
<input type="checkbox"/>		Load Lakehouse Table	  ...	Dataflow Gen2
		wwilakehouse		Lakehouse
		wwilakehouse		Semantic model
		wwilakehouse		SQL analytics enc

13. 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.

The screenshot displays the Databricks Lakehouse interface. On the left, the 'Explorer' sidebar shows the 'wwilakehouse' containing a 'Tables' folder with the 'dimension_customer' table highlighted. The main area shows a preview of the 'dimension_customer' table with 403 rows. A red box highlights the 'SQL analytics endpoint' in the top right corner, which is used to query data using SQL.

id	CustomerKey	WWCustomerID	ABC Customer	ABC BillToCusto...	ABC Category	ABC BuyingGroup	ABC PrimaryCont...	ABC PostalCode	ValidFrom	ValidTo	LineageKey
1	0	0	Unknown	N/A	N/A	N/A	N/A	N/A	1/1/2013 12:00:0...	1/1/2025 12:00:0...	0
2	102	102	Tailspin Toys (Fieldbr...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Tea Koppel	90205	1/1/2013 12:00:0...	1/1/2025 12:00:0...	2
3	103	103	Tailspin Toys (Kilvest...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Naseem Radan	90130	1/1/2013 12:00:0...	1/1/2025 12:00:0...	2
4	104	104	Tailspin Toys (Wallagr...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Laboni Deb	90579	1/1/2013 12:00:0...	1/1/2025 12:00:0...	2
5	105	105	Tailspin Toys (Tomnol...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Sung-Hwan Hwa...	90400	1/1/2013 12:00:0...	1/1/2025 12:00:0...	2
6	106	106	Tailspin Toys (Tumaca...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Shiva Pipalia	90662	1/1/2013 12:00:0...	1/1/2025 12:00:0...	2
7	107	107	Tailspin Toys (Glen Av...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Karie Mercier	90782	1/1/2013 12:00:0...	1/1/2025 12:00:0...	2
8	108	108	Tailspin Toys (Bernie, ...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Bhanu Thota	90045	1/1/2013 12:00:0...	1/1/2025 12:00:0...	2
9	109	109	Tailspin Toys (South L...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Ae-Cha Joo	90247	1/1/2013 12:00:0...	1/1/2025 12:00:0...	2
10	110	110	Tailspin Toys (North C...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Dinara Saparkyzy	90792	1/1/2013 12:00:0...	1/1/2025 12:00:0...	2
11	111	111	Tailspin Toys (Oniole ...	Tailspin Toys (He...	Novelty Shop	Kids Toys	Adam Dvorak	90436	1/1/2013 12:00:0...	1/1/2025 12:00:0...	2

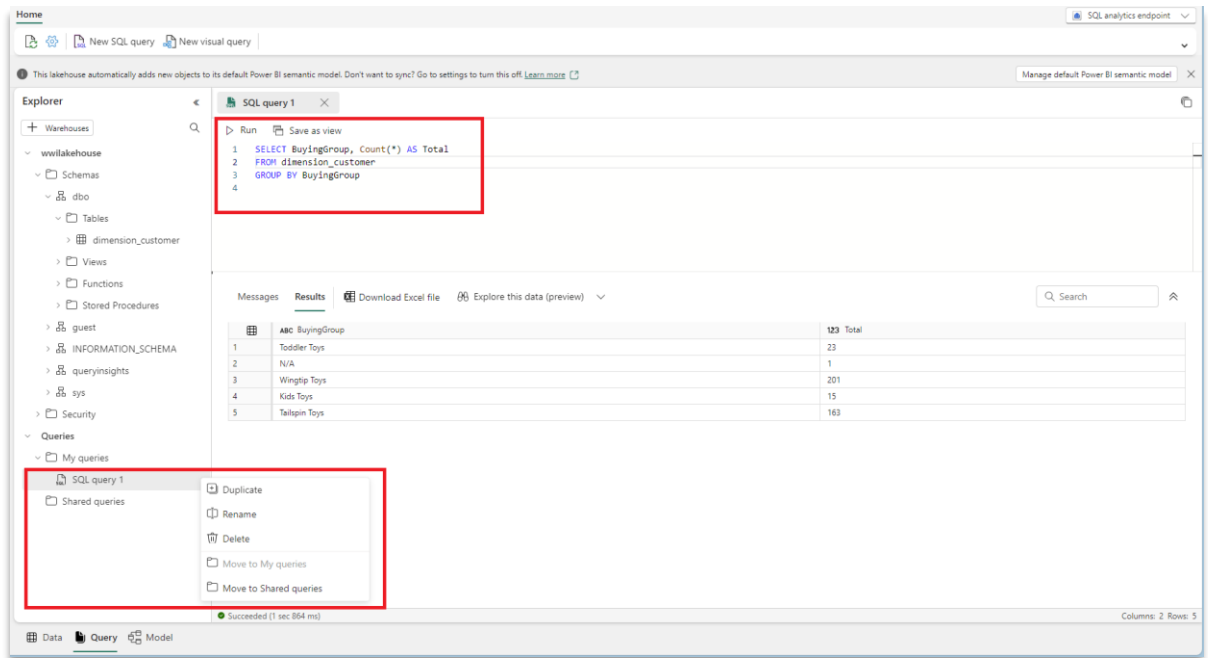
14. In the warehouse mode, you can click on **dimension_customer** table to preview its data and/or click on **New SQL query** to write your SQL statements.



15. 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.

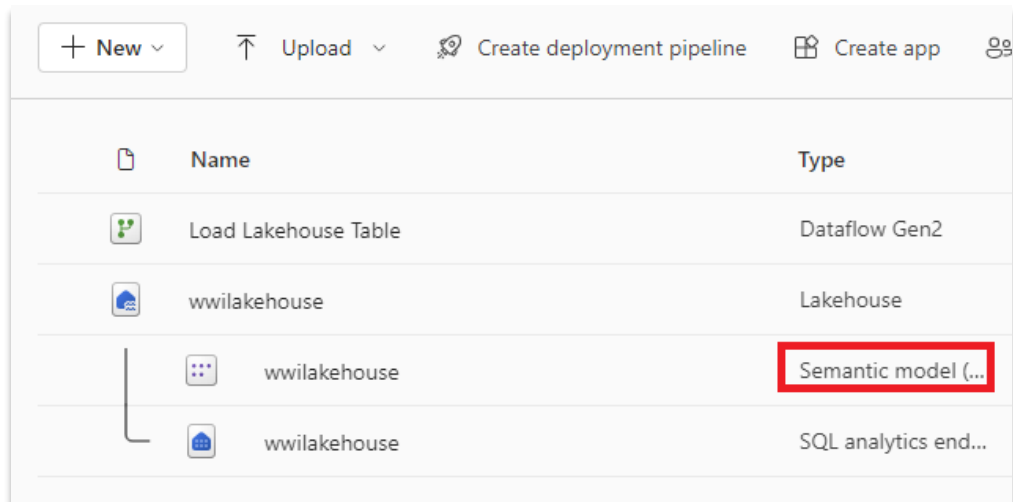
To run the script, click on the **Run** icon at the top of the script file.

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

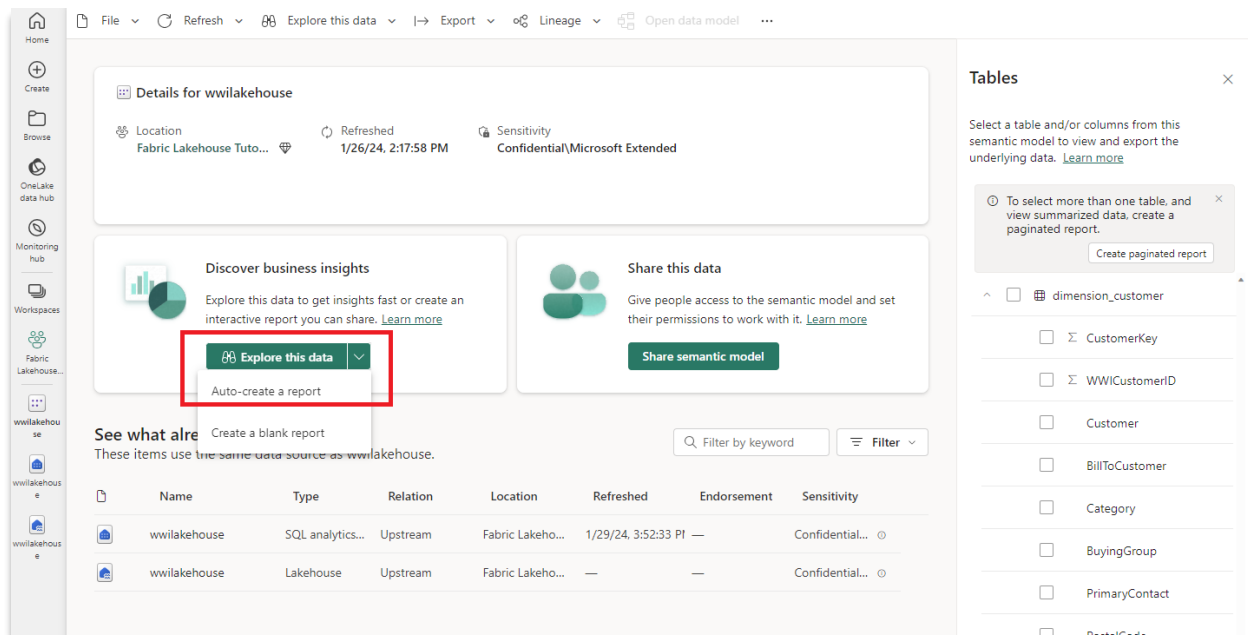


Building a report

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 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.



- 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.



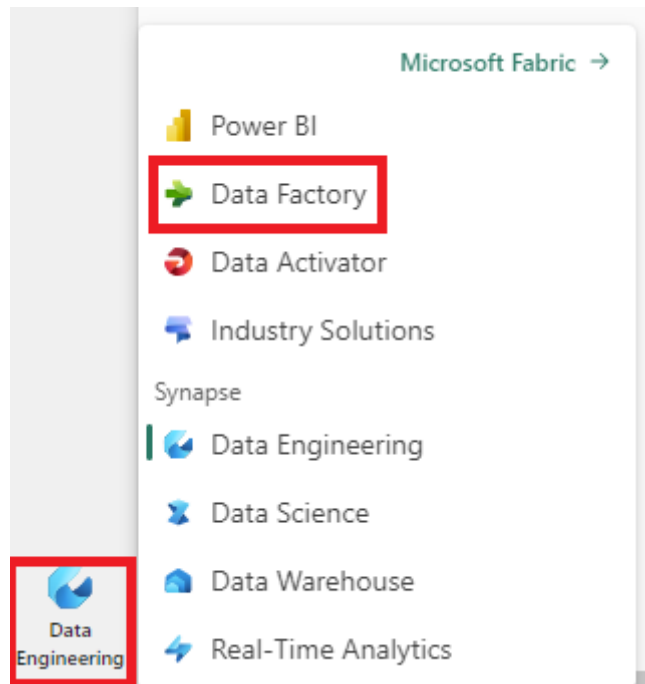
Module 3: Ingest, Prep and Analyze

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.

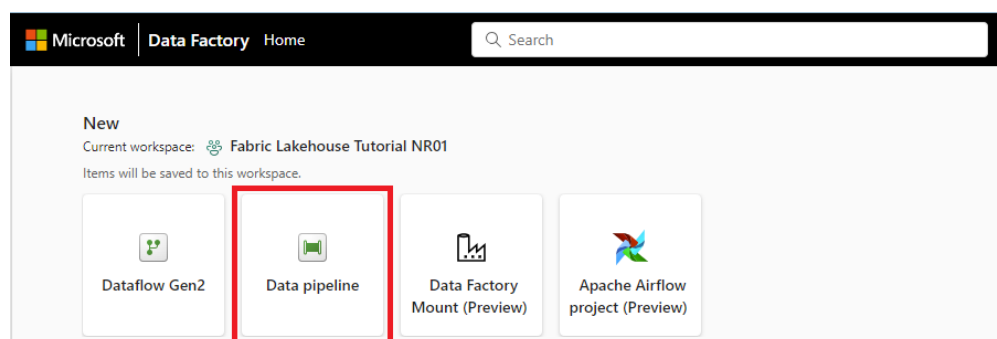
Data Ingestion

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.

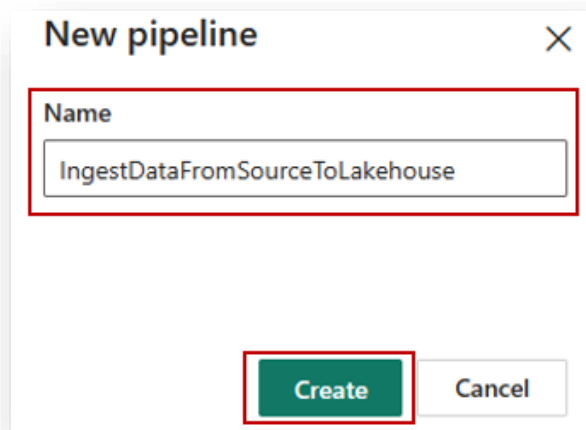
1. On the bottom left of the screen, click on the workload switcher, and then select **Data Factory**.



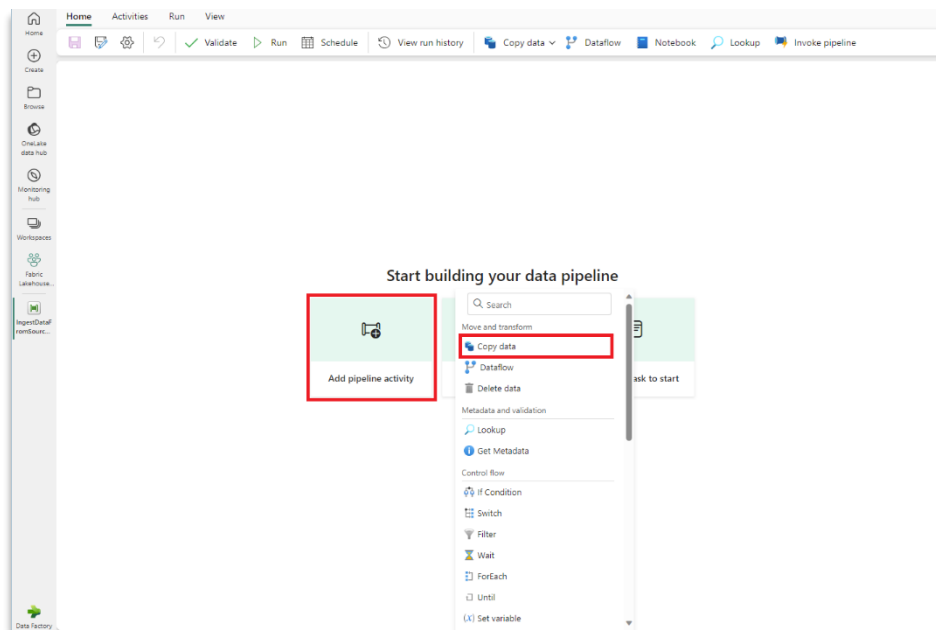
2. Click on **Data pipeline** under **New** to create a new data pipeline.



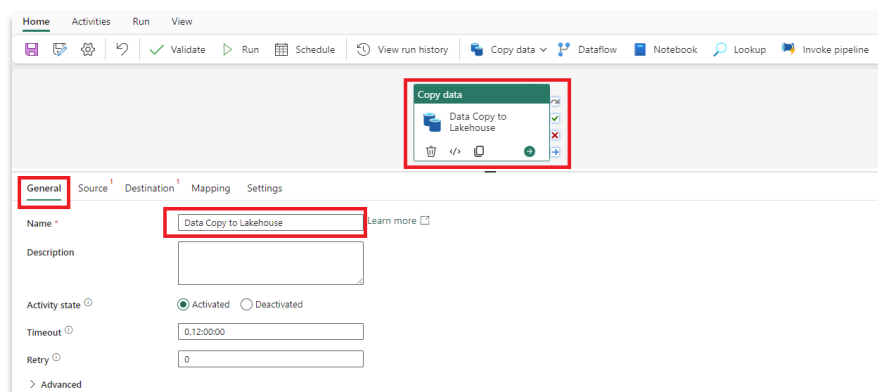
3. For the **New pipeline**, specify the name as **IngestDataFromSourceToLakehouse** and click on **Create**. This will create a new data factory pipeline and open it on the screen to work on it.



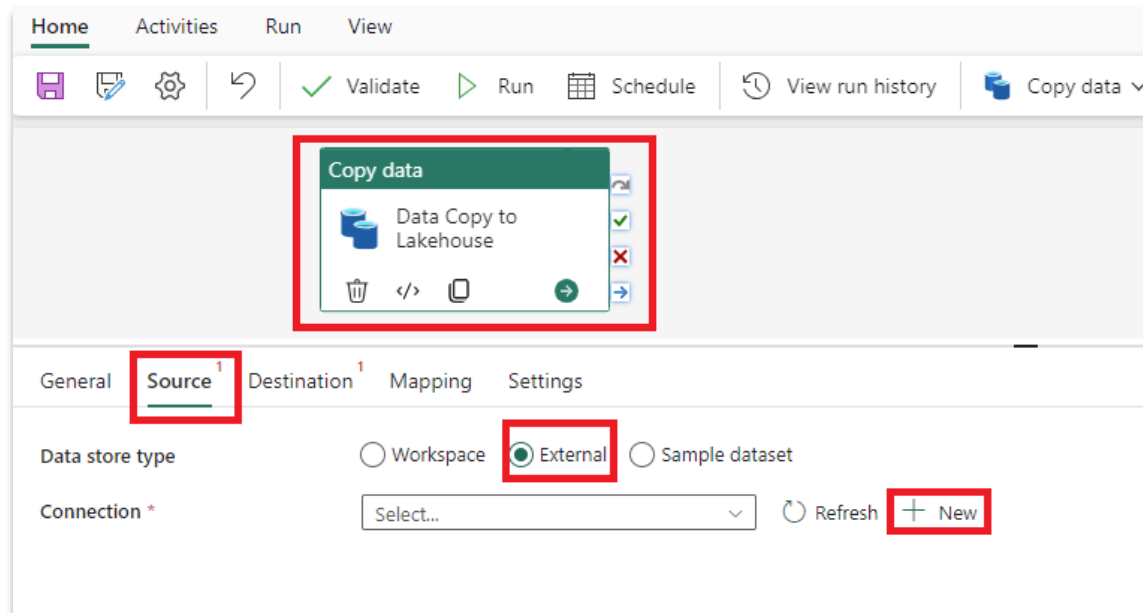
- On the newly created data factory pipeline, click **Add pipeline activity** to add an activity to the pipeline and click on **Copy data**. This will add copy data activity to the pipeline canvas.



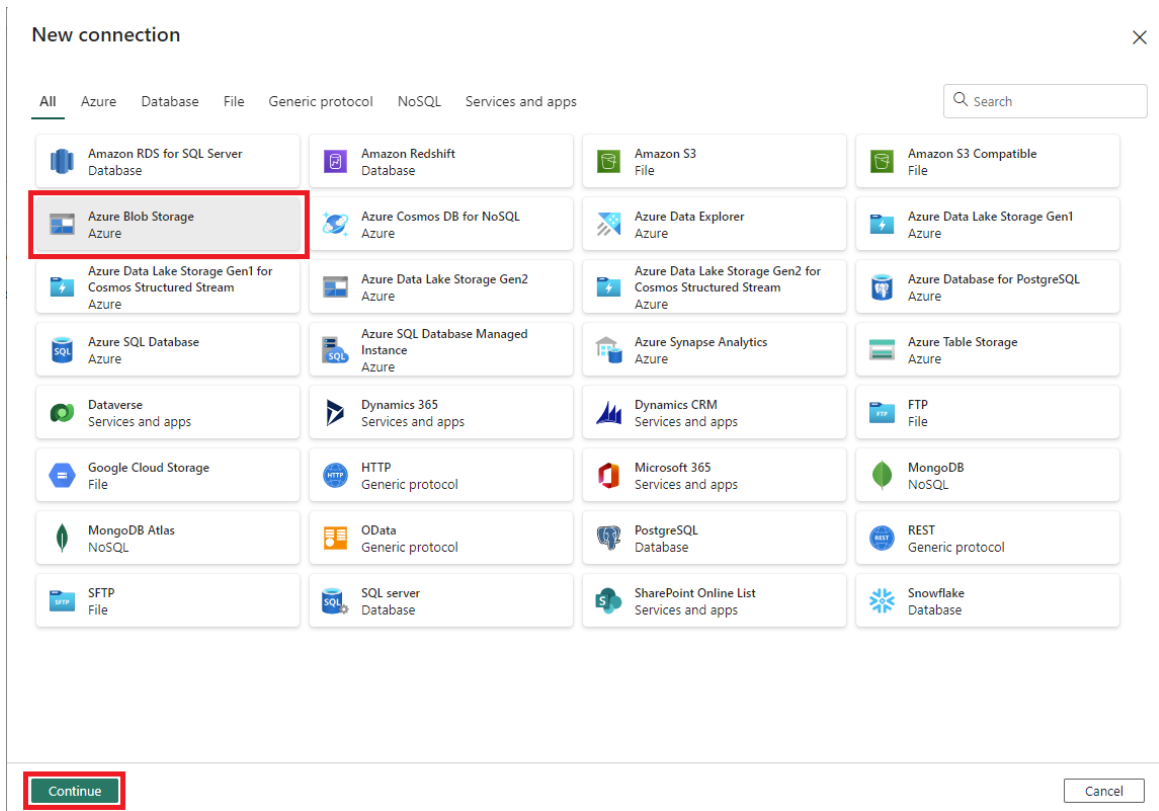
- Select the newly added copy data activity to the canvas, which will show activity properties at the bottom. Under **General** tab, specify the name for the copy data activity **Data Copy to Lakehouse**.



6. Under **Source** tab of the selected copy data activity, select **External** as **Data store type** and then click on **+ New** to create a new connection to data source.



7. For this tutorial, all the sample data is available in a public container of Azure blob storage so you will connect to this container to copy data from. On the **New connection wizard**, select **Azure Blob Storage** and then click on **Continue**.



8. On the next screen of the **New connection wizard**, please specify these details and click on **Create** to create the connection to the data source.

Account name or URI	https://azuresynapsestorage.blob.core.windows.net/sampledata
Connection	Create new connection
Connection name	wwisampledta
Authentication kind	Anonymous

New connection

Azure Blob Storage
Learn more

Connection settings

Account name or URL *
https://azuresynapsestorage.blob.core.windows.net/sampl...

Connection credentials

Connection
Create new connection

Connection name
wwisampledta

Authentication kind
Anonymous

Create Back Cancel

9. Once the new connection is created, you will return to **Source** tab of the copy data activity and newly created connection will be selected by default. Please specify these properties before moving to the destination settings.

Data store type	External
Connection	wwisampledta
File path type	File path
File path	Container name (first text box) – sampledata Directory name (second text box) – WideWorldImportersDW/parquet
Recursively	Checked
File Format	Binary

Home Activities Run View

Validate Run Schedule View run history Copy data Dataflow Notebook

Copy data

Data Copy to Lakehouse

General **Source** Destination Mapping Settings

Data store type
☐ Workspace ☒ External ☐ Sample dataset

Connection *
wwisampledta

File path type
☒ File path ☐ Prefix ☐ Wildcard file path ☐ List of files

File path *
sampledata / WideWorldImporters...

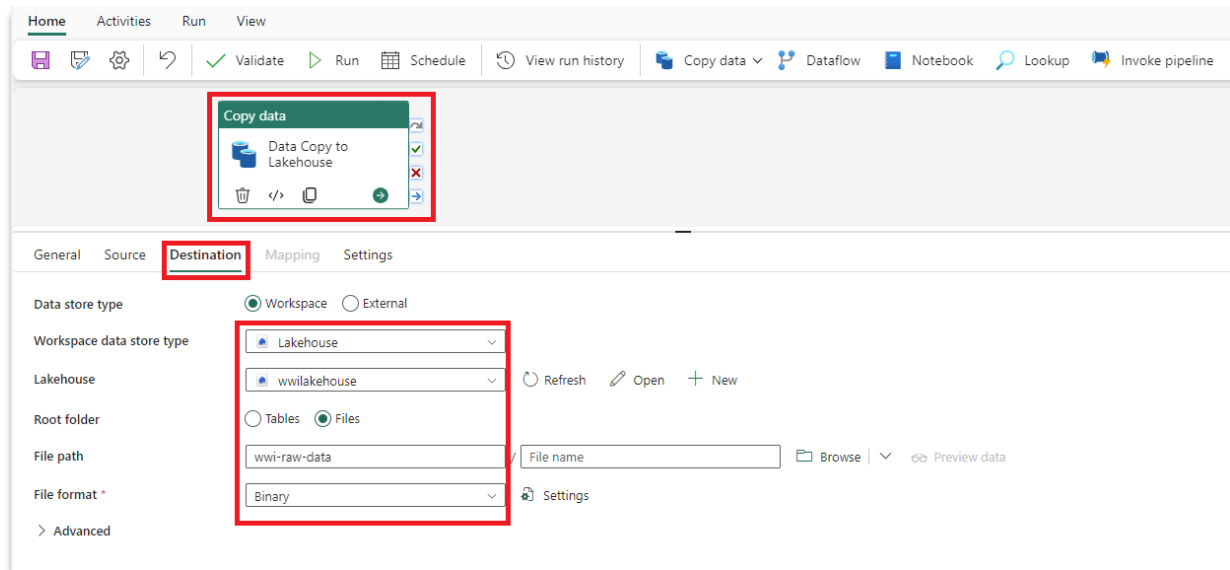
Recursively ☒

File format *
Binary

> Advanced

10. Under **Destination** tab of the selected copy data activity, please specify these properties:

Data store type	Workspace
Workspace data store type	Lakehouse
Lakehouse	Wwilakehouse
Root Folder	Files
File path	Directory name (first text box) – wwi-raw-data
File Format	Binary

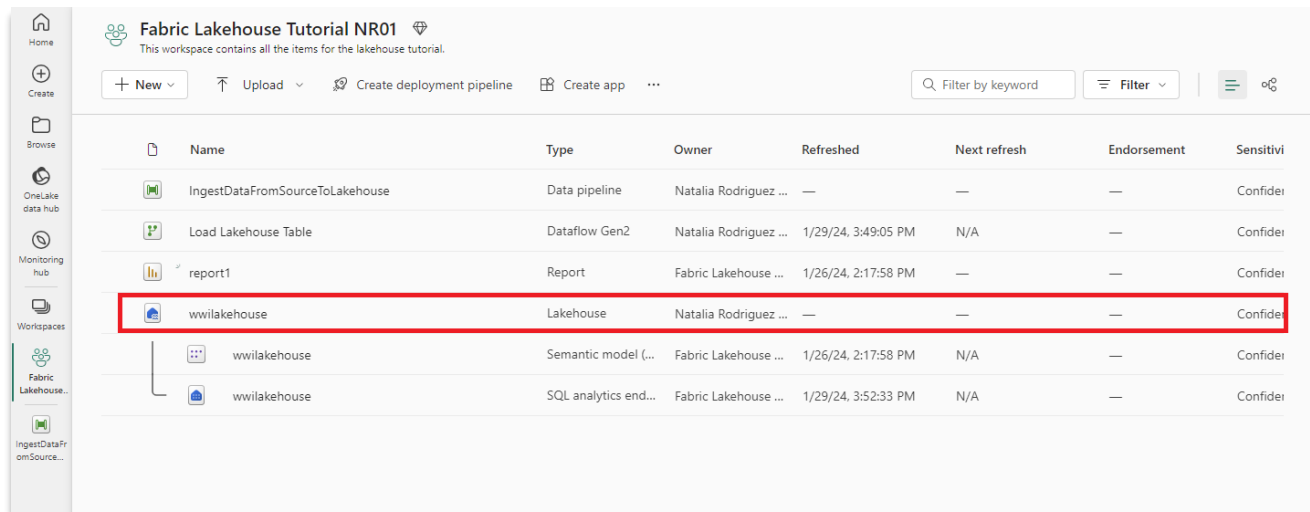


11. At this time, you have completed configuring copy data activity. Click on **Save** button under **Home** to save all these details and click on **Run** to kick off execution of this pipeline and its activity. You can also schedule pipelines to refresh data at defined intervals to meet your business requirements, however for this module, we will run the pipeline once by clicking on **Run** button. This will trigger data copy from the underlying data source to the specified lakehouse and might take up to a minute to complete. You can monitor the execution of the pipeline and its activity under **Output** tab which appears when you click anywhere on the canvas. Optionally, you can click on the activity name “**Data Copy to Lakehouse**” to look at the details of data transfer.

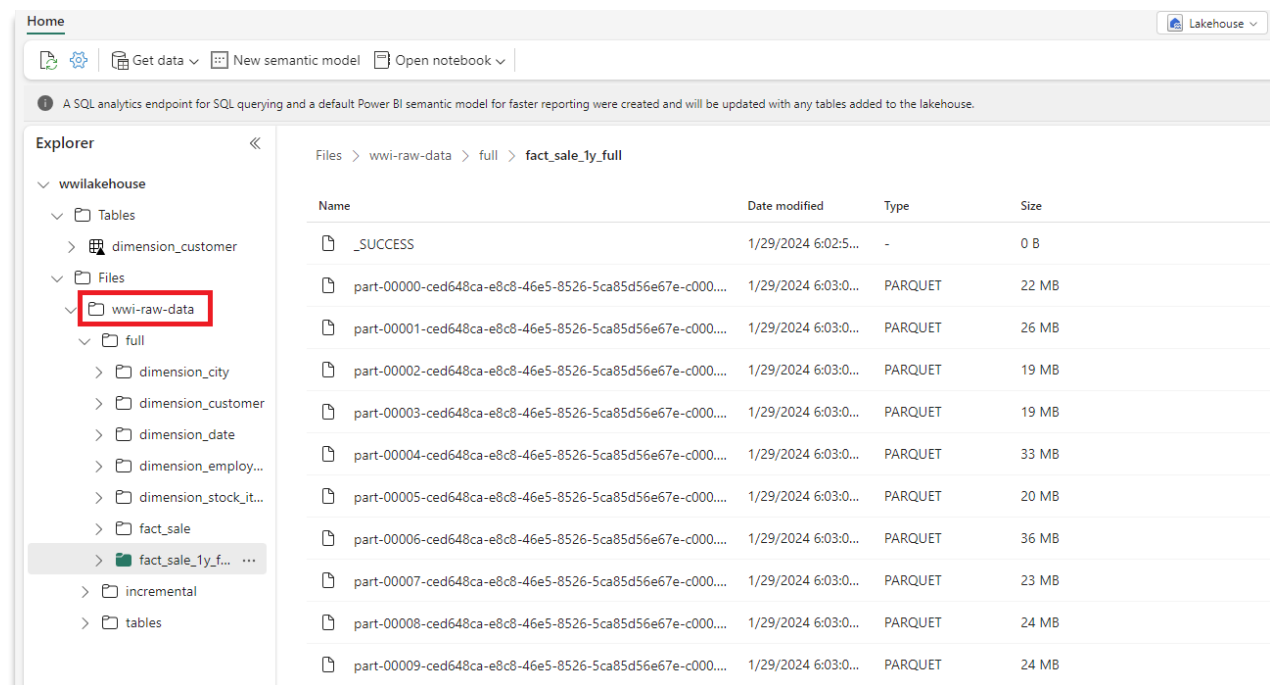
The screenshot shows the Databricks workspace interface. At the top, the 'Run' button is highlighted with a red box. Below the canvas, the 'Output' tab is selected and highlighted with a red box. The pipeline run ID is 4af1614a-0c73-40e4-9a56-716d853e0b63, and the pipeline status is 'Succeeded'. The output table shows one item: 'Data Copy to Lakehouse' with a status of 'Succeeded', a run start time of 1/29/2024, 6:02:35 PM, and a duration of 30s. The 'Data Copy to Lakehouse' activity is also visible in the canvas above the output table.

Activity name	Activity status	Run start	Duration	Input	Output
Data Copy to Lakehouse	Succeeded	1/29/2024, 6:02:35 PM	30s		

12. Once the data copy is completed, go to items view of the workspace and click on **wwilakehouse** to launch **Lakehouse explorer** of this selected lakehouse.



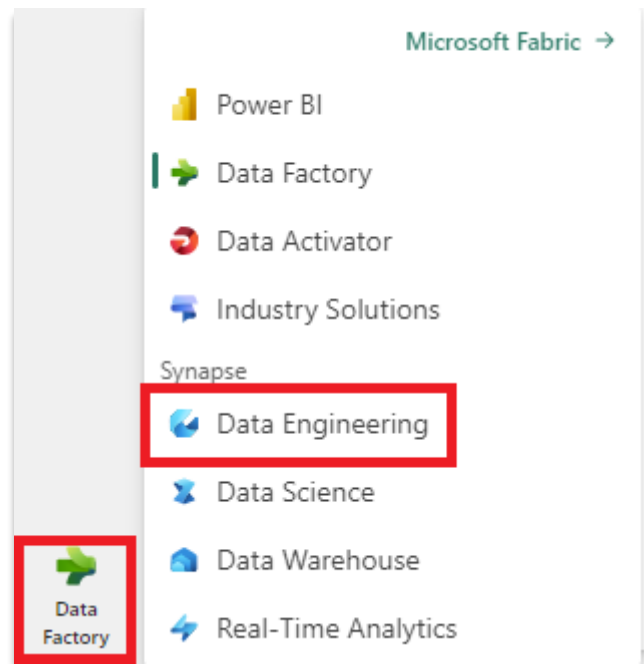
13. In the **Lakehouse explorer** view, you would notice a new folder **wwi-raw-data** has been created and data for all the tables have been copied here.



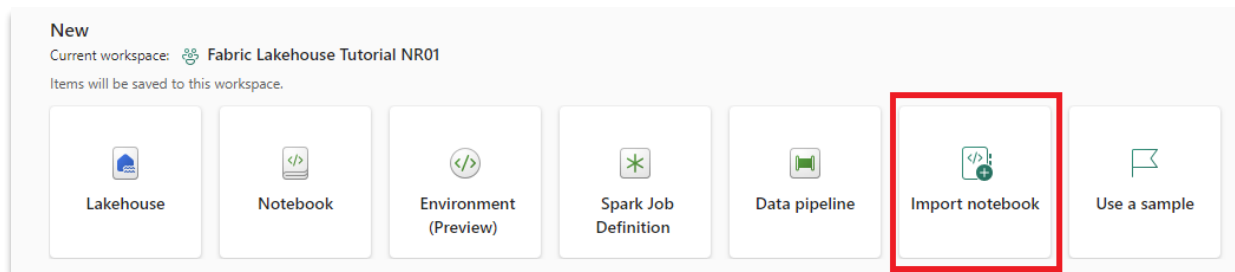
Data Preparation

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.

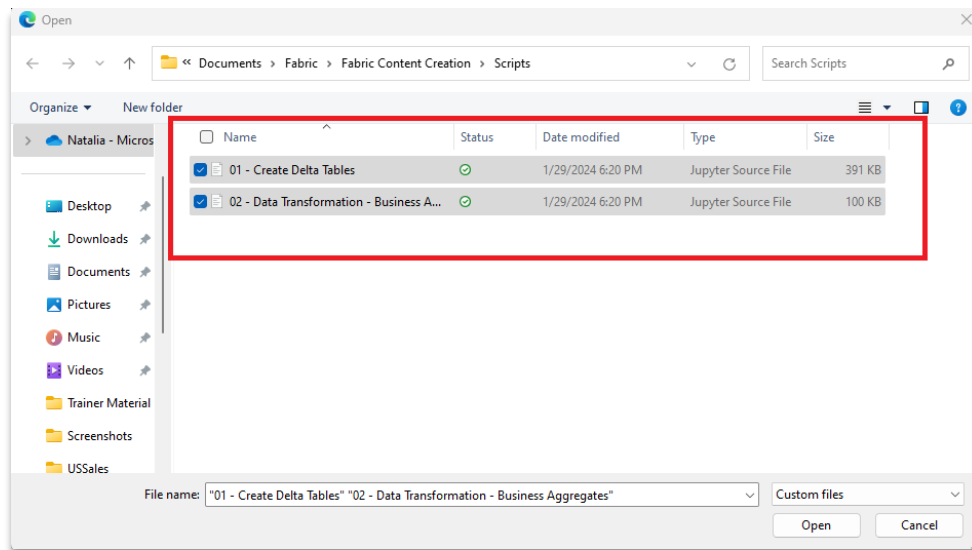
1. Download the set of notebooks found in the Scripts folder
2. From the workload switcher located at the bottom left of the screen, select **Data engineering**.



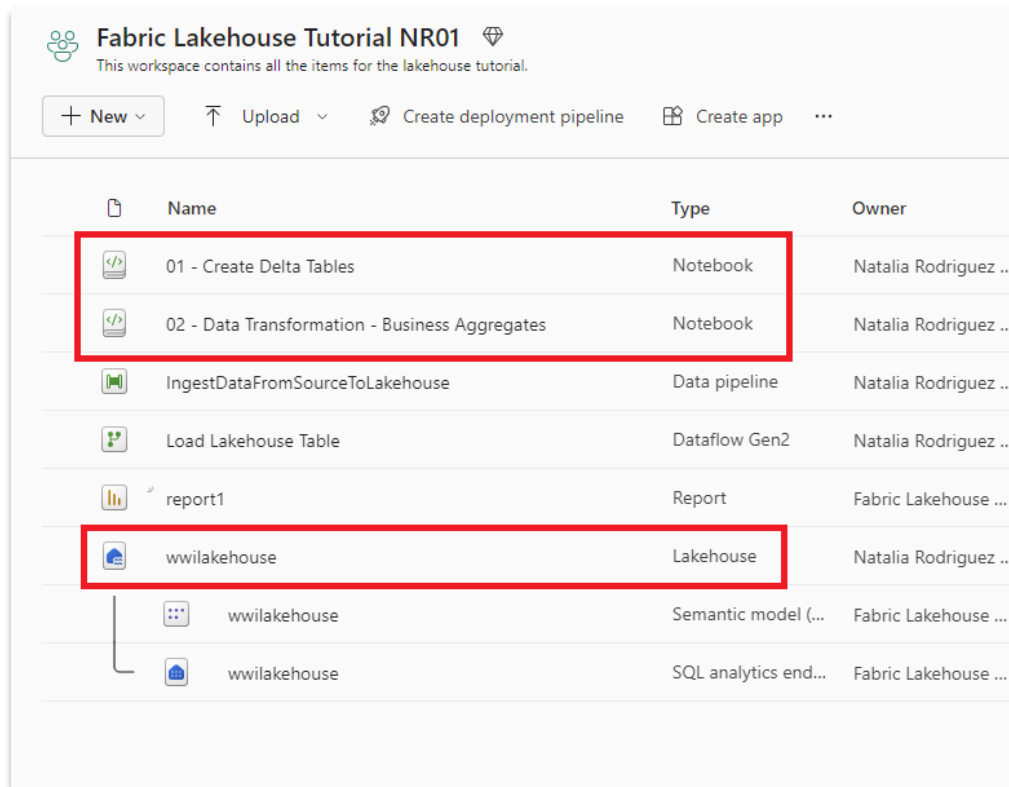
3. Select **Import notebook** from the **New** section at the top of the landing page.



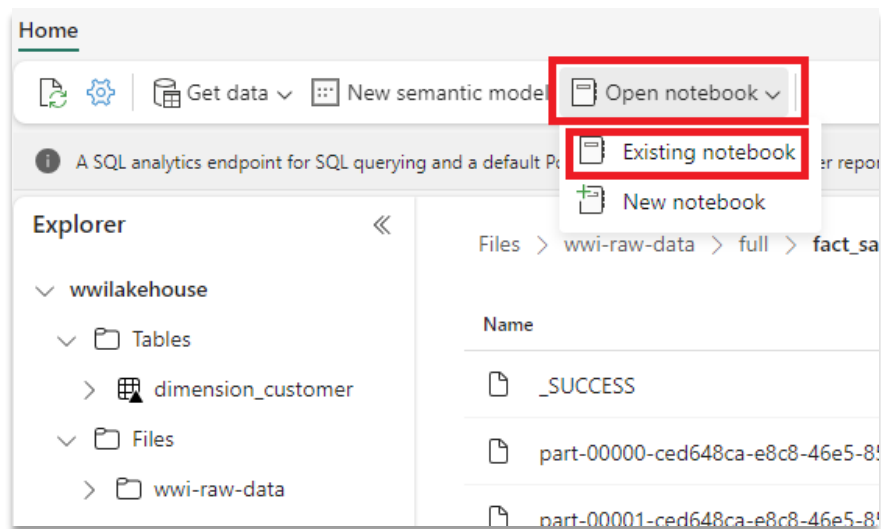
4. Select **Upload** from the **Import status** pane that opens on the right side of the screen.
5. Select all the notebooks that were downloaded and/or unzipped in step 1 of this section.
6. Select **Open**. A notification indicating the status of the import will appear in the top right corner of the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



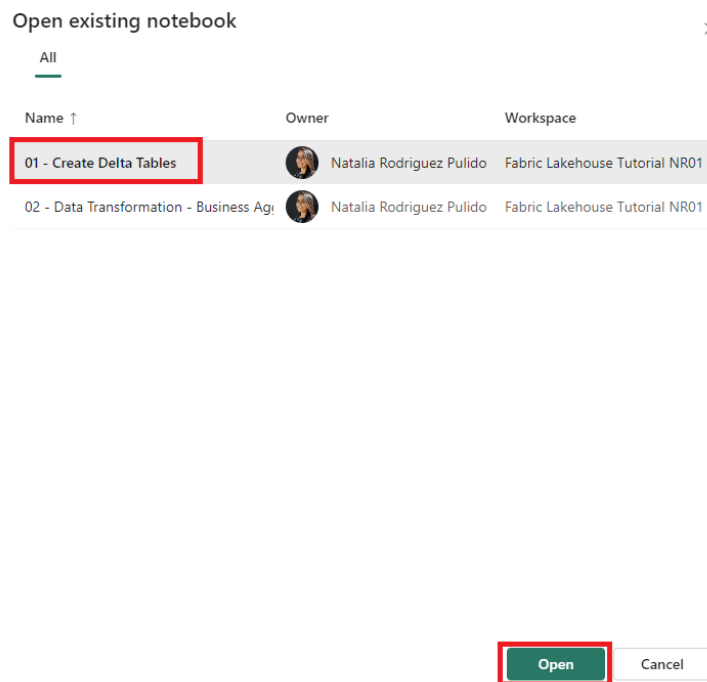
7. After the import of notebooks is successful, you can go to items view of the workspace and see these newly imported notebooks. Click on **wwilakehouse** lakehouse to open it.



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 date 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/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 55 sec 644 ms by Arshad Ali on 11:03:21 AM, 4/14/23

PySpark (Python) ▾

```
from pyspark.sql.functions import col, year, month, quarter
```

```
table_name = 'fact_sale'
```

```
df = spark.read.format("parquet").load('Files/wwi-raw-data/full/fact_sale_1y_full')
df = df.withColumn('Year', year(col("InvoiceDateKey")))
df = df.withColumn('Quarter', quarter(col("InvoiceDateKey")))
df = df.withColumn('Month', month(col("InvoiceDateKey")))
```

```
df.write.mode("overwrite").format("delta").partitionBy("Year","Quarter").save("Tables/" + table_name)
```

13. After 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.

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/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)
```

```

from pyspark.sql.types import *

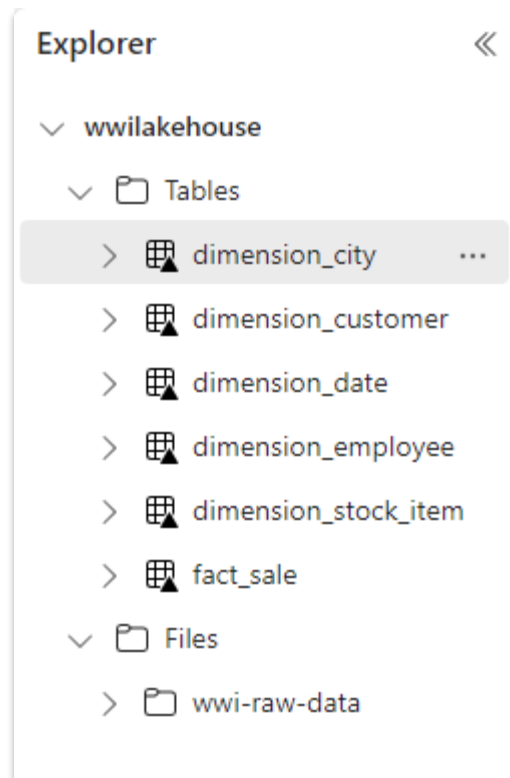
def loadFullDataFromSource(table_name):
    df = spark.read.format("parquet").load('Files/wwi-raw-data/full/' +
        table_name)
    df = df.select([c for c in df.columns if c != 'Photo'])
    df.write.mode("overwrite").format("delta").save("Tables/" + table_name)

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

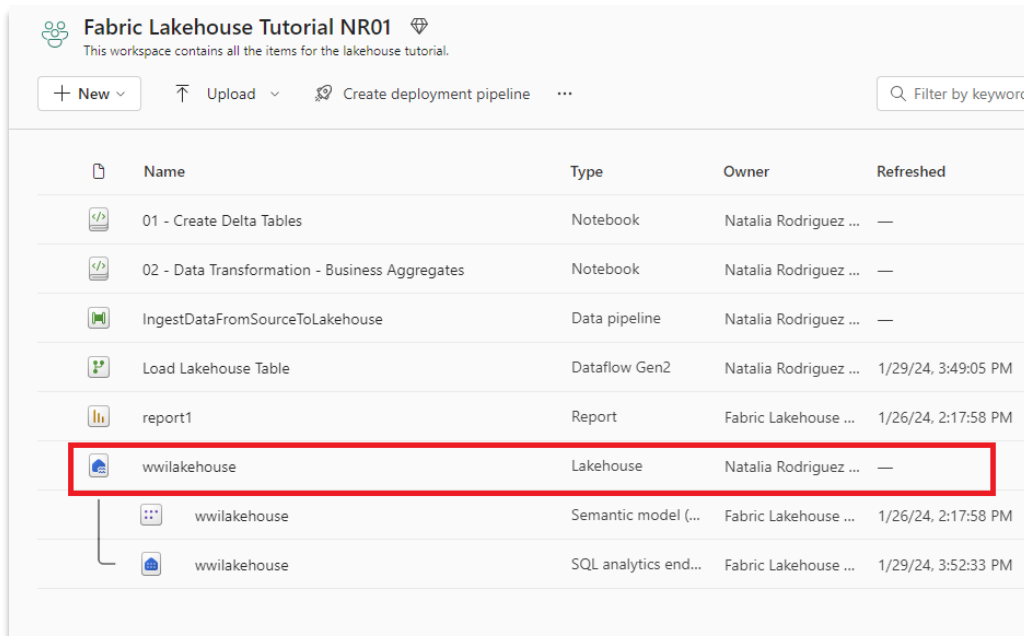
for table in full_tables:
    loadFullDataFromSource(table)

```

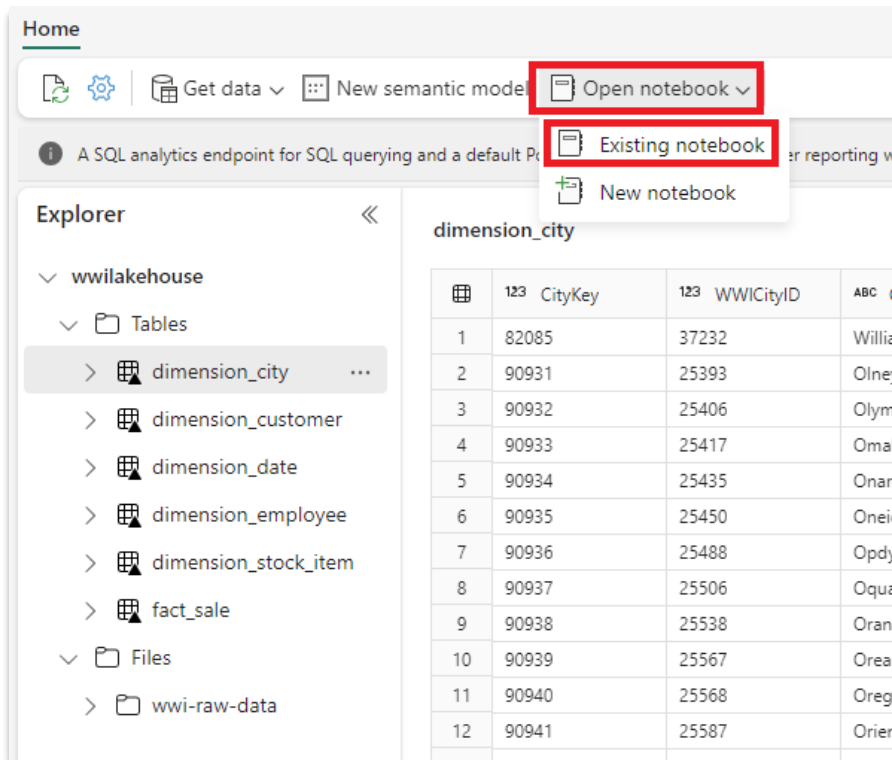
14. You can validate created tables by right clicking and selecting refresh on **wwilakehouse** lakehouse and you will notice these created tables.



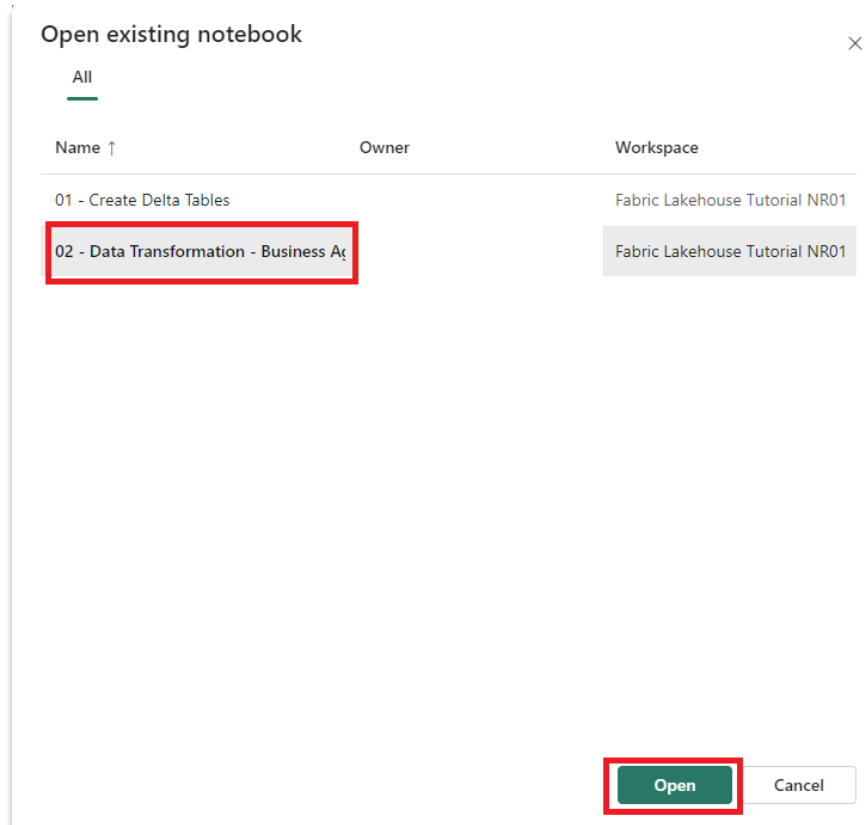
15. Please go the items view of the workspace again and click on **wwilakehouse** lakehouse to open it.



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.
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")
```

✓ 59 sec - Command executed in 59 sec 175 ms by Arshad Ali on 4:15:57 PM, 4/14/23

PySpark (Python) ▾

> Spark jobs (9 of 9 succeeded) ...

+ Code + Markdown

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 in the Tables section of the lakehouse.

+ Code + Markdown

```
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 "city.SalesTerritory", "sale.TotalExcludingTax", "sale.TaxAmount", "sale.TotalIncludingTax", "sale.Profit") \
6 .groupBy("date.Date", "date.CalendarMonthLabel", "date.Day", "date.ShortMonth", "date.CalendarYear", "city.City", "city.StateProvince",
7 "city.SalesTerritory") \
8 .sum("sale.TotalExcludingTax", "sale.TaxAmount", "sale.TotalIncludingTax", "sale.Profit") \
9 .withColumnRenamed("sum(TotalExcludingTax)", "SumOfTotalExcludingTax") \
10 .withColumnRenamed("sum(TaxAmount)", "SumOfTaxAmount") \
11 .withColumnRenamed("sum(TotalIncludingTax)", "SumOfTotalIncludingTax") \
12 .withColumnRenamed("sum(Profit)", "SumOfProfit") \
13 .orderBy("date.Date", "city.StateProvince", "city.City")
14 sale_by_date_city.write.mode("overwrite").format("delta").option("overwriteSchema", "true").save("Tables/aggregate_sale_by_date_city")
```

✓ 19 sec - Command executed in 19 sec 85 ms by Arshad Ali on 4:16:16 PM, 4/14/23

PySpark (Python) ▾

> Spark jobs (12 of 12 succeeded) ...

```
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.

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 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")
```

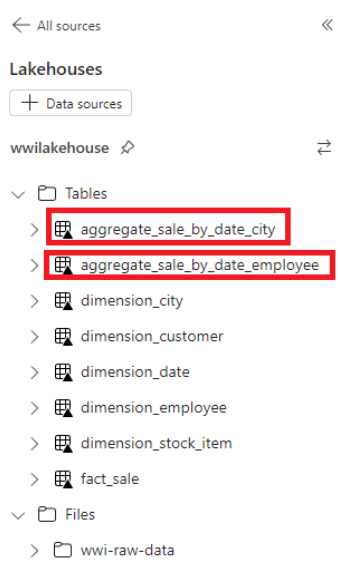
✓ 19 sec - Command executed in 19 sec 74 ms by Arshad Ali on 4:16:46 PM, 4/14/23

PySpark (Python)

```
%%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.

	Date	ARB	CalendarMonth	ARB	Day	ARB	ShortMonth	123	CalendarYear	ARB	City	ARB	StateProvince	ARB	SalesTerritory	SumOfTotalExclu...	SumOfTotalAmount	SumOfTotalInclu...
1	1/1/2000 12:00...	CY2000-Jan	1		Jan	2000	Bazemore	Alabama	Southeast	82944.00	12441.60	95385.60						
2	1/1/2000 12:00...	CY2000-Jan	1		Jan	2000	Belgreen	Alabama	Southeast	601380.00	90192.00	691472.00						
3	1/1/2000 12:00...	CY2000-Jan	1		Jan	2000	Coker	Alabama	Southeast	257984.00	38697.60	296681.60						
4	1/1/2000 12:00...	CY2000-Jan	1		Jan	2000	Eulaton	Alabama	Southeast	507584.00	76137.60	583721.60						
5	1/1/2000 12:00...	CY2000-Jan	1		Jan	2000	Highland Home	Alabama	Southeast	18480.00	2772.00	21252.00						
6	1/1/2000 12:00...	CY2000-Jan	1		Jan	2000	Jemison	Alabama	Southeast	278816.00	41822.72	320638.72						
7	1/1/2000 12:00...	CY2000-Jan	1		Jan	2000	Marion Junction	Alabama	Southeast	212992.00	31948.80	244940.80						
8	1/1/2000 12:00...	CY2000-Jan	1		Jan	2000	Nanafalia	Alabama	Southeast	312256.00	46638.40	359094.40						
9	1/1/2000 12:00...	CY2000-Jan	1		Jan	2000	Robertsdale	Alabama	Southeast	799222.40	119883.36	919105.76						
10	1/1/2000 12:00...	CY2000-Jan	1		Jan	2000	Saks	Alabama	Southeast	469312.00	70396.64	539708.64						
11	1/1/2000 12:00...	CY2000-Jan	1		Jan	2000	Southside	Alabama	Southeast	20160.00	3024.00	23184.00						
12	1/1/2000 12:00...	CY2000-Jan	1		Jan	2000	Tuscaloosa	Alabama	Southeast	36000.00	5400.00	41400.00						
13	1/1/2000 12:00...	CY2000-Jan	1		Jan	2000	Akhvok	Alaska	Far West	601177.60	90176.64	691354.24						
14	1/1/2000 12:00...	CY2000-Jan	1		Jan	2000	Elvok	Alaska	Far West	106000.00	15900.00	121900.00						
15	1/1/2000 12:00...	CY2000-Jan	1		Jan	2000	Haycock	Alaska	Far West	284544.00	42681.60	327225.60						
16	1/1/2000 12:00...	CY2000-Jan	1		Jan	2000	Ikatan	Alaska	Far West	114400.00	17160.32	131560.32						

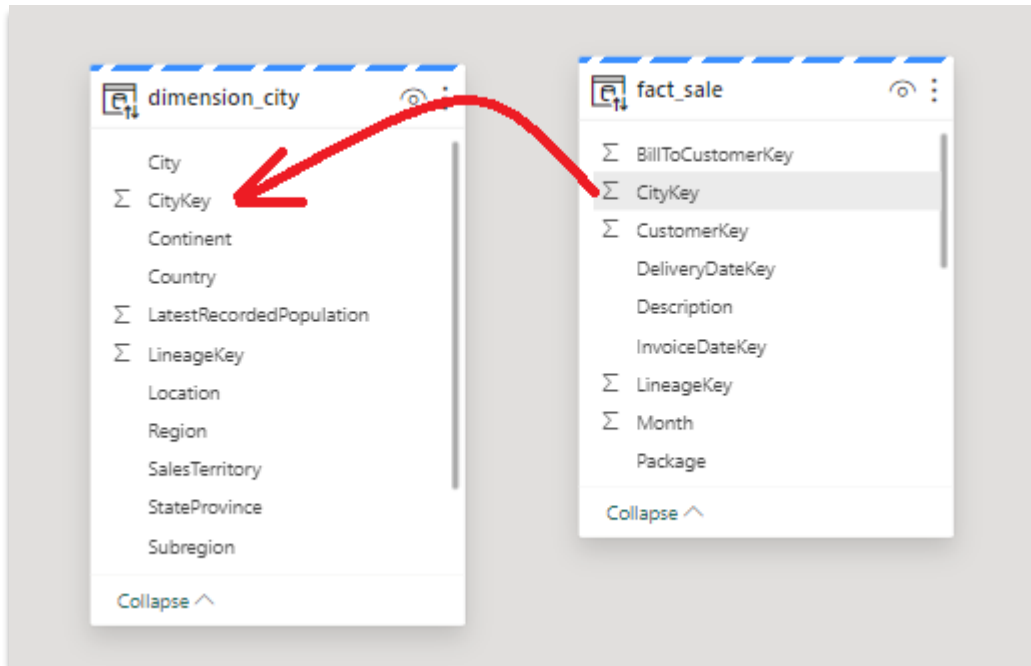
2. Once you are in warehouse mode you should be able to see all the tables you created. If you don't see them yet, please click on the **Refresh** icon at the top. Next, click on **Model** tab at the bottom to open the default Power BI dataset.

The screenshot displays the Microsoft Fabric Lakehouse interface. On the left, the 'Explorer' pane shows a tree view of the 'wwilakehouse' with folders for 'Schemas', 'Tables', 'Views', 'Functions', 'Stored Procedures', 'Queries', and 'My queries'. The 'Tables' folder is expanded, showing a list of tables including 'aggregate_sa...', 'dimension_city', 'dimension_cu...', 'dimension_d...', 'dimension_e...', 'dimension_st...', 'fact_sale', and 'guest'. The 'Data preview' pane on the right shows a table with 1000 rows and 10 columns. The 'Model' tab is selected at the bottom of the interface.

	Date	ABC_CalendarMonthLabel	ABC_Day	ABC_ShortMonth	123_CalendarYear	ABC_City	ABC_StateProvinc
1	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Bazemore	Alabama
2	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Belgreen	Alabama
3	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Coker	Alabama
4	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Eulaton	Alabama
5	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Highland Home	Alabama
6	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Jemison	Alabama
7	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Marion Junction	Alabama
8	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Nanafalia	Alabama
9	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Robertsdale	Alabama
10	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Saks	Alabama
11	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Southside	Alabama
12	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Tuscaloosa	Alabama
13	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Alkiok	Alaska
14	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Ekwook	Alaska
15	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Haycock	Alaska
16	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Ikatan	Alaska
17	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Iliamna	Alaska
18	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	King Cove	Alaska
19	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Kwethluk	Alaska
20	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Lemeta	Alaska
21	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Port Chilkoot	Alaska
22	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Cortaro	Arizona
23	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Peoples Valley	Arizona
24	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Queen Valley	Arizona
25	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Tempe	Arizona
26	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Tumacacori	Arizona
27	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Walapai	Arizona
28	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Cale	Arkansas
29	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Grannick	Arkansas

Select 'Default semantic model objects' tab at the bottom of the screen.

3. 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.

- 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**.

New relationship

Select tables and columns that are related.

Table 1	Table 2
fact_sale	dimension_city
Column: CityKey	Column: CityKey

Define cardinality and cross filter direction for tables and columns

Cardinality	Cross-filter direction
Many to one (*:1)	Single

☒ Make this relationship active

☒ Assume referential integrity. [Learn more](#)

Ok **Cancel**

- 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
- WWInvoiceID

ABC Description

dimension_city

- City
- CityKey
- Continent
- Country
- LatestRecordedPopulation
- LineageKey
- Location
- Region
- SalesTerritory
- StateProvince
- SubRegion

dimension_date

- CalendarMonthLabel
- CalendarMonthNumber
- CalendarYearLabel
- CalendarYear
- Date
- Day
- DayNumber
- FiscalMonthLabel
- FiscalMonthNumber

fact_sale

- BillToCustomerKey
- CityKey
- CustomerKey
- DeliveryDateKey
- Description
- InvoiceDateKey
- LineageKey
- Month
- Package

dimension_stock_item

- Barcode
- Brand
- BuyingPackage
- Color
- IsChillerStock
- LeadTimeDays
- LineageKey
- QuantityPerOuter
- RecommendedRetailPrice

dimension_customer

- BillToCustomer
- BuyingGroup
- Category
- Customer
- CustomerKey
- LineageKey
- PostalCode
- PrimaryContact
- ValueFrom
- ValueTo
- WWCustomerID

dimension_employee

- Employee
- EmployeeKey
- IsSalesperson
- LineageKey
- PreferredName
- ValueFrom
- ValueTo
- WWEmployeeID

aggregate sale by da...

- CalendarMonthLabel
- Date
- Day
- Employee
- Month
- PreferredName
- SumOfProfit
- SumOfTaxAmount
- SumOfTotalExcludingTax

aggregate sale by da...

- CalendarMonthLabel
- CalendarYear
- City
- Date
- Day
- SalesTerritory
- ShortMonth
- StateProvince
- SumOfProfit

exec_requests_history

- batch_id
- command
- connection_id
- distributed_statement_id
- end_time
- login_name
- program_name
- query_hash
- root_batch_id

frequently run queries

- avg_total_elapsed_time_ms
- last_statement_id
- last_run_command
- last_run_start_time
- last_run_total_elapsed_time_ms
- max_run_total_elapsed_time_ms
- min_run_total_elapsed_time_ms
- number_of_cancelled_runs

long running queries

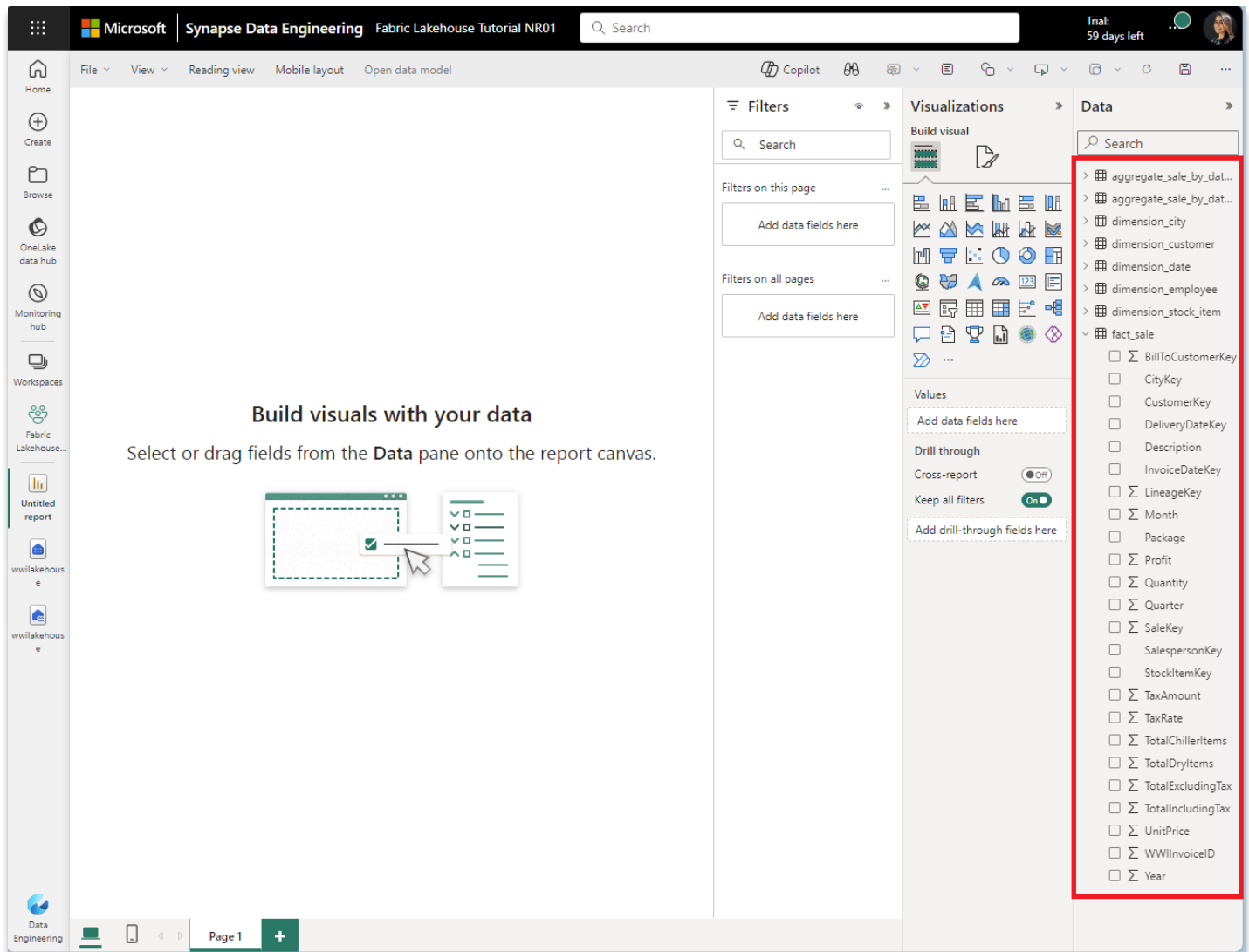
- last_statement_id
- last_run_session_id
- last_run_start_time
- last_run_total_elapsed_time_ms
- median_total_elapsed_time_ms
- number_of_runs

All tables Default dataset objects + 57%

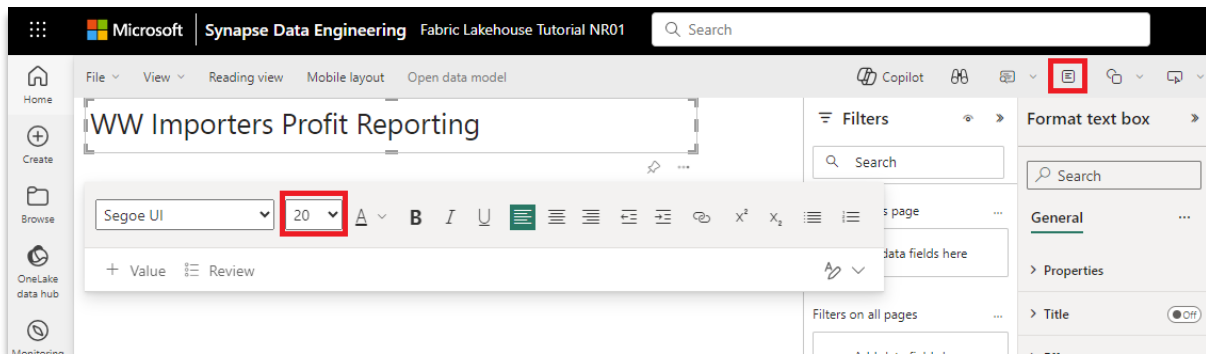
Data Query Model

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.

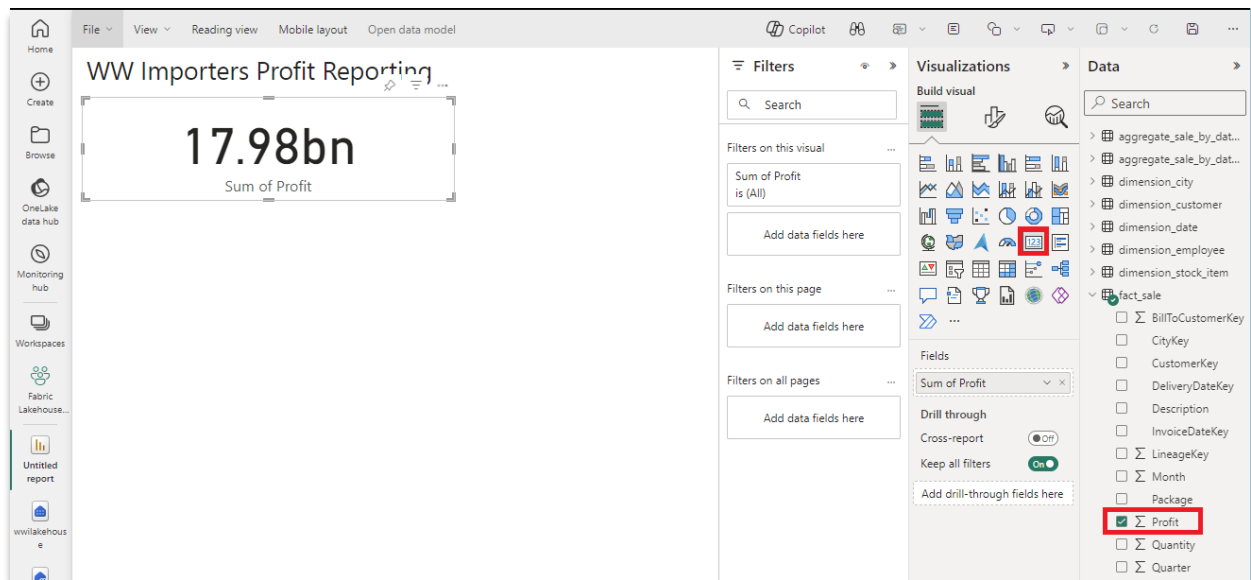
- On the Power BI report canvas, you can create reports to meet your business requirements by dragging required columns from the **Data** pane to the canvas and using one or more of available visualizations.



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

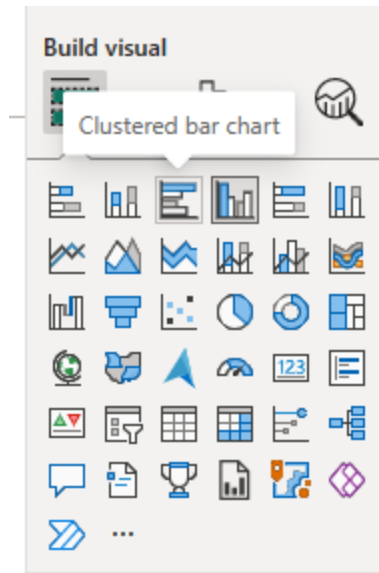


7. Add a Card
 - a. On the **Data** pane, expand **fact_sales** and check the box next to **Profit**. This will create a column chart and add the field to the Y-axis.
 - b. With the bar chart selected, click the Card visual in the visualization pane. This will convert the visual to a card.
 - c. Place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

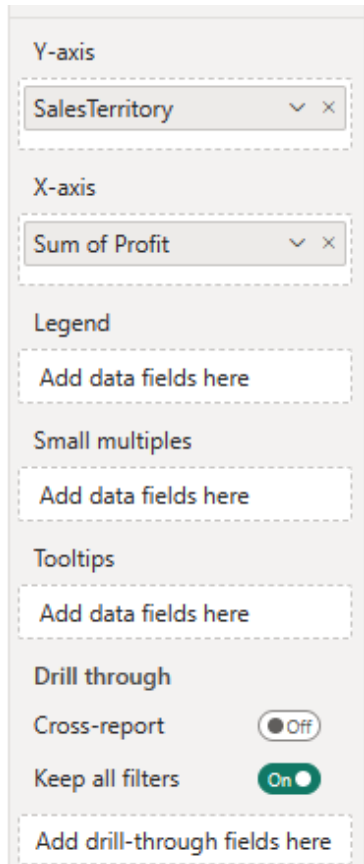


8. Add a Bar chart

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



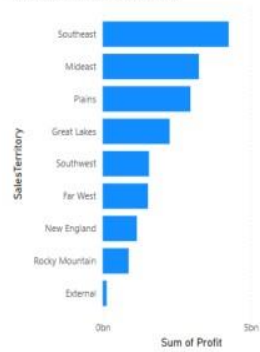
- d. Resize the Bar chart to fill in the area under the title and Card



17.98bn

Sum of Profit

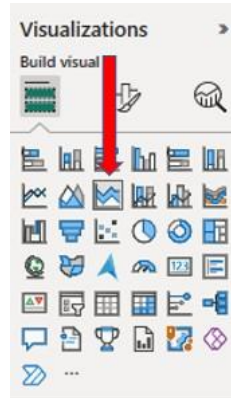
Sum of Profit by SalesTerritory



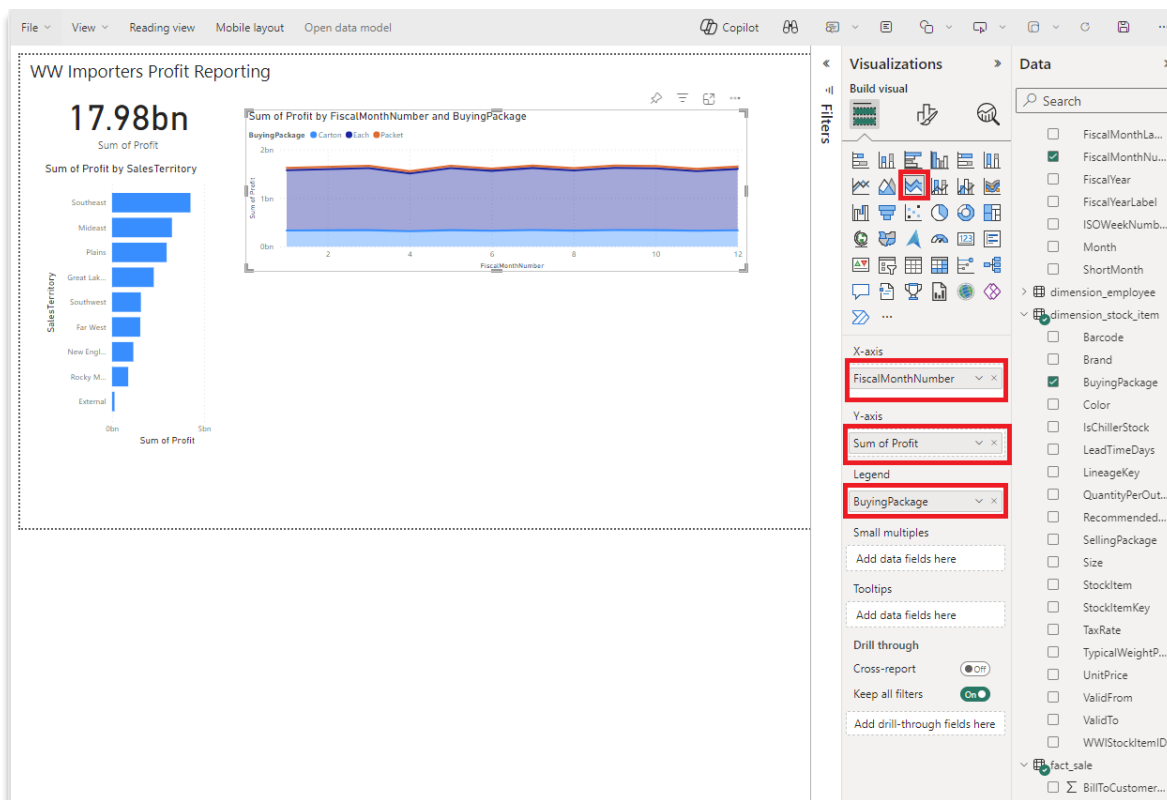
9. Click anywhere on the blank canvas (or press the Esc key) so the bar chart visual is no longer selected.

10. Build a stacked area chart visual:

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



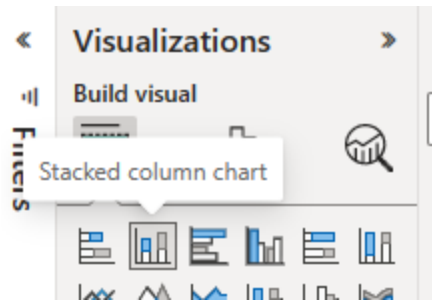
- b. Reposition and resize the stacked area chart to the right of the card and bar chart visuals created in the previous steps.
- c. On the **Data** pane, expand **fact_sales** and check the box next to **Profit**. Expand dimension_date and check the box next to **FiscalMonthNumber**. This will create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



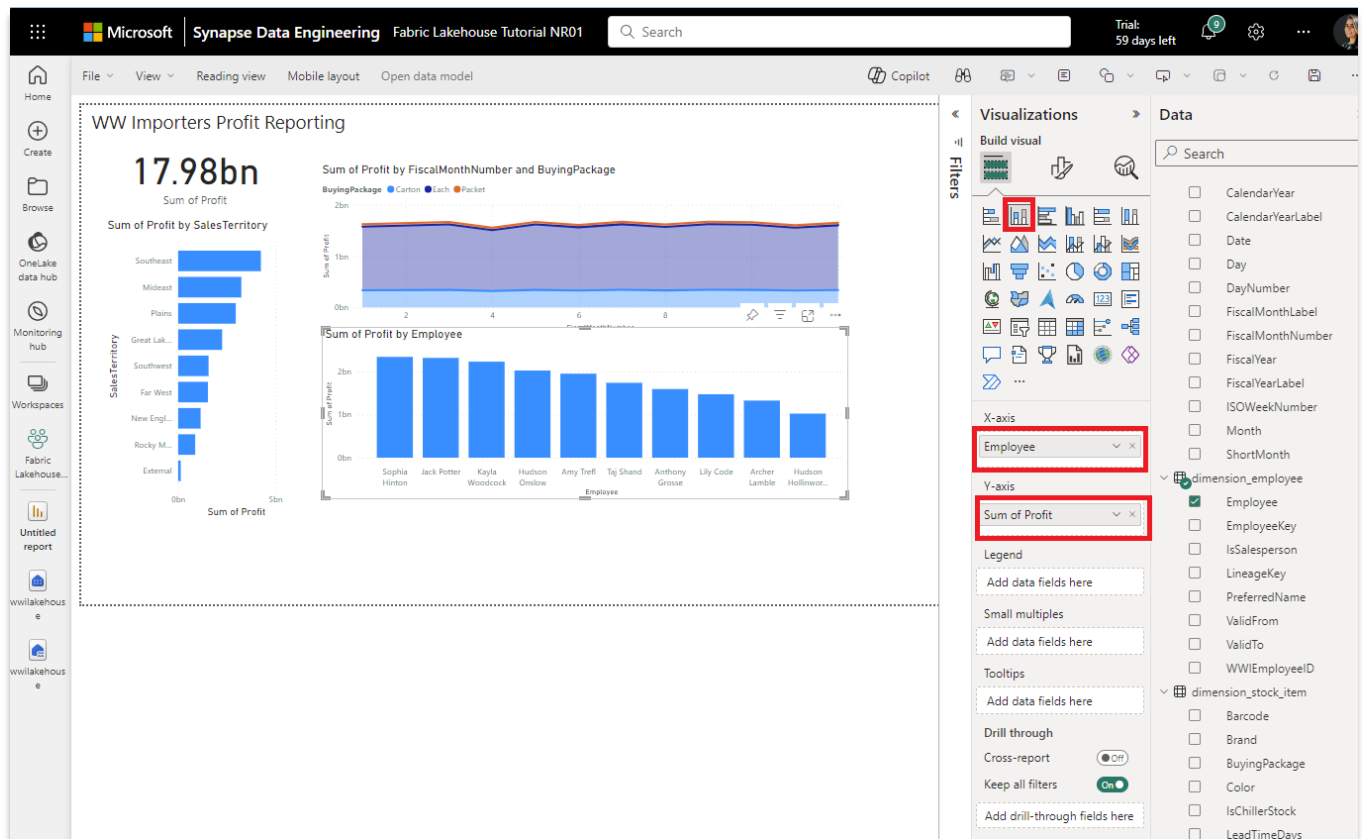
11. Click anywhere on the blank canvas (or press the Esc key) so the stacked area chart visual is no longer selected.

12. Build a column chart:

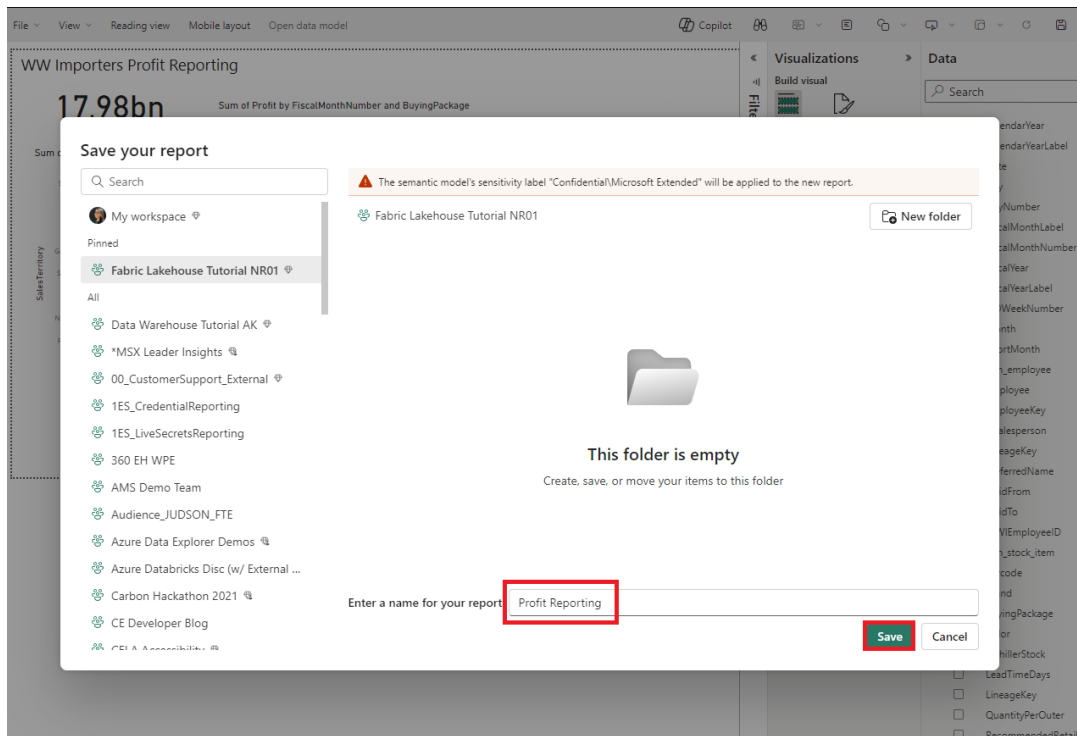
- a. On the **Visualizations** pane, select the **Stacked column chart** visual.



- 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.



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

The screenshot shows the Microsoft Synapse Data Engineering interface. The top header includes the Microsoft logo, 'Synapse Data Engineering', and the workspace name 'Fabric Lakehouse Tutorial NR01'. Below the header, the workspace name 'Fabric Lakehouse Tutorial NR01' is displayed with a description 'This workspace contains all the items for the lakehouse tutorial.' To the right of the workspace name is a three-dot menu icon (highlighted with a red rectangle). A dropdown menu is open, showing 'Manage access' and 'Workspace settings' (highlighted with a red rectangle). The main area contains a table of workspace items.

	Name	Type			Next refr
	01 - Create Delta Tables	Notebook	Natalia Rodriguez ...	—	—
	02 - Data Transformation - Business Aggregates	Notebook	Natalia Rodriguez ...	—	—
	IngestDataFromSourceToLakehouse	Data pipeline	Natalia Rodriguez ...	—	—
	Load Lakehouse Table	Dataflow Gen2	Natalia Rodriguez ...	1/29/24, 3:49:05 PM	N/A
	Profit Reporting	Report	Fabric Lakehouse ...	1/26/24, 2:17:58 PM	—
	report1	Report	Fabric Lakehouse ...	1/26/24, 2:17:58 PM	—
	wwilakehouse	Lakehouse	Natalia Rodriguez ...	—	—
	wwilakehouse	Semantic model (...)	Fabric Lakehouse ...	1/26/24, 2:17:58 PM	N/A
	wwilakehouse	SQL analytics end...	Fabric Lakehouse ...	1/30/24, 2:41:09 PM	N/A

