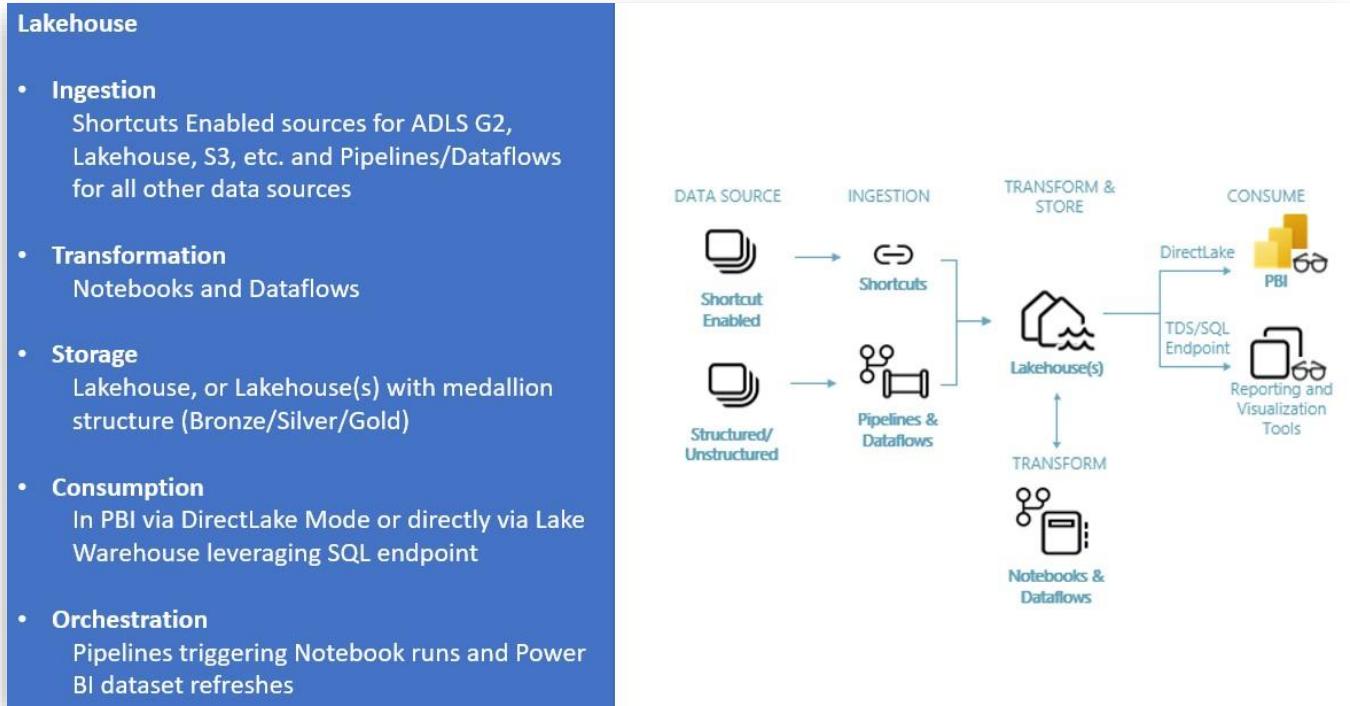


Lakehouse Tutorial

Lakehouse

Published: Aug 2024



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.....	26
Module 3: Ingest, Prep and Analyze	29
Data Ingestion.....	29
Data Preparation	37
Building a report.....	50
Module 4: Clean up resources	62

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.

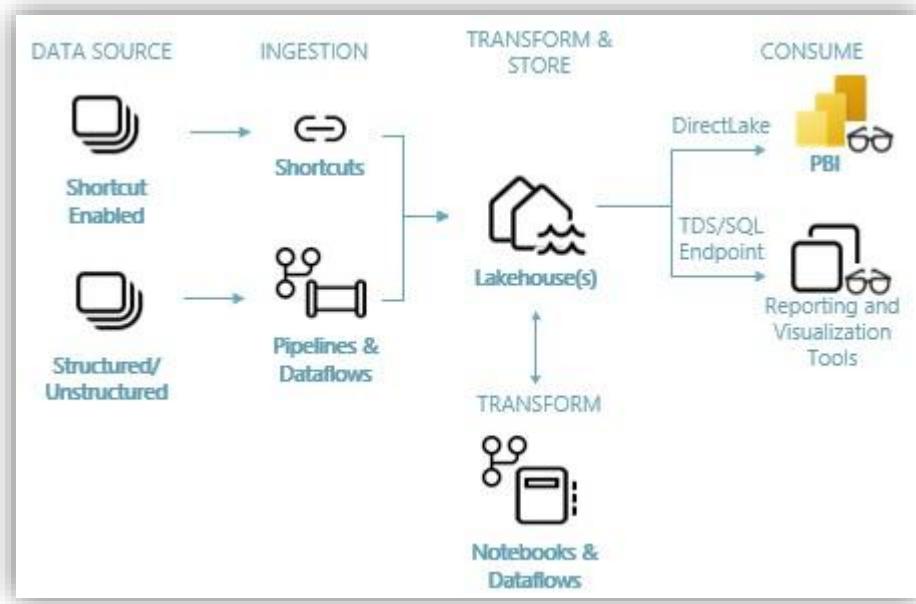
With the flexibility Fabric provides, you can implement either lakehouse or data warehouse architectures or combine these two together to get the best of both the worlds in simple implementation. In this tutorial, you are going to take an example of retail organization and build its lakehouse from start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

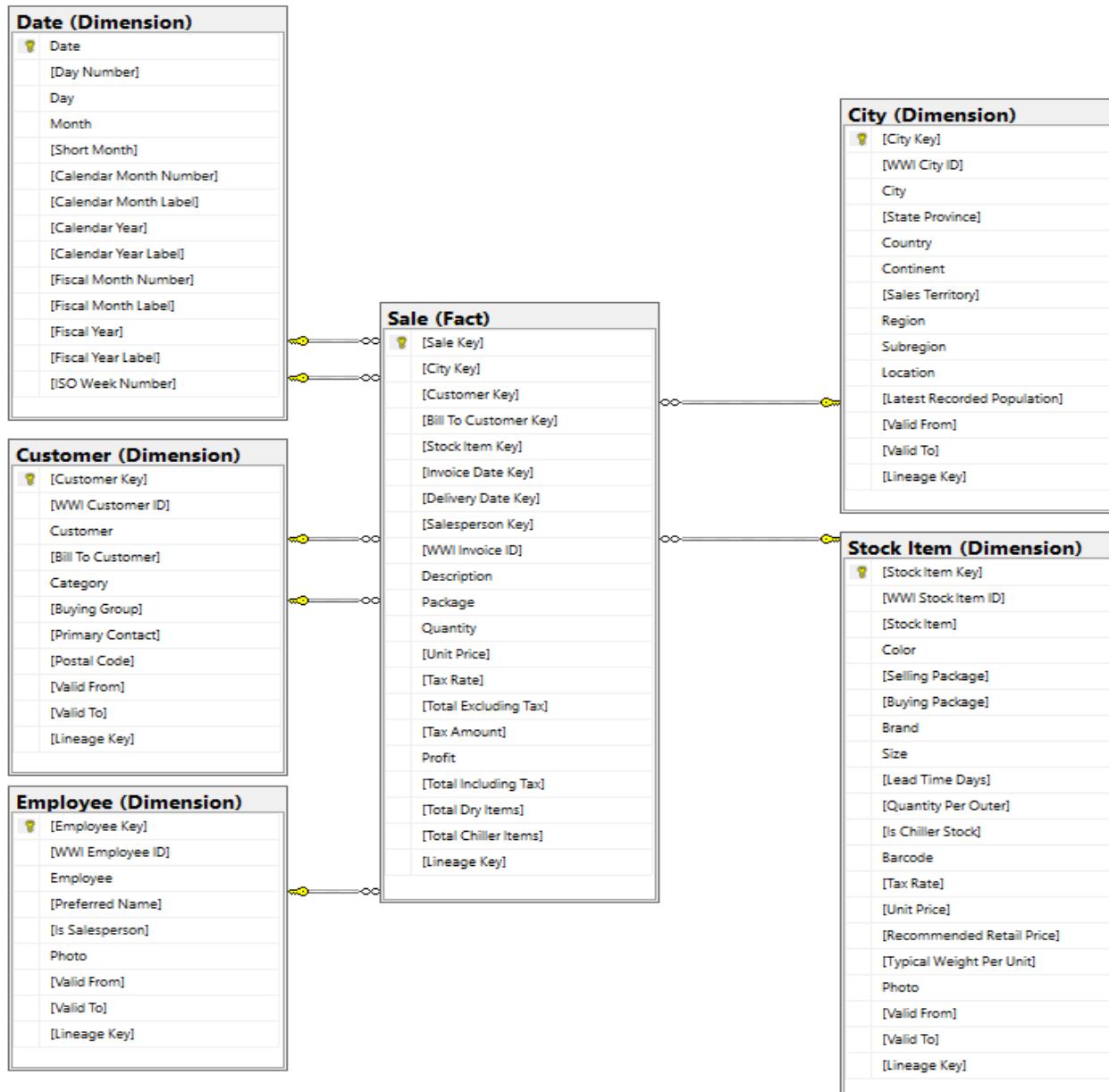
Wide World Importers (WWI) is a wholesale novelty goods importer and distributor operating from the San Francisco Bay area. As a wholesaler, WWI's customers are mostly companies who resell to individuals. WWI sells to retail customers across the United States including specialty stores, supermarkets,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

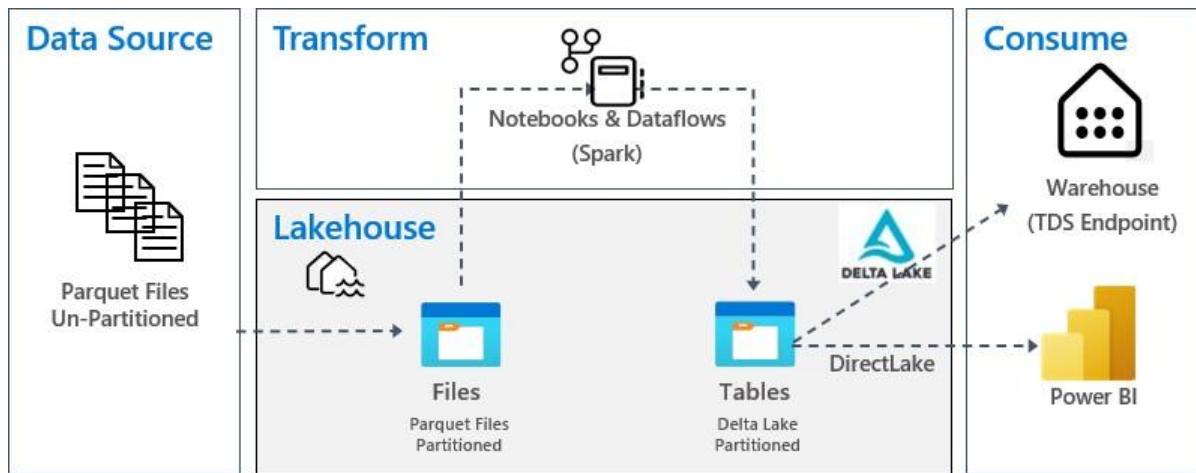
While the WWI dimensional model contains multiple fact tables, for simplicity in explanation we will focus on the Sale Fact table and its related dimensions only, as below, to demonstrate this end-to-end lakehouse scenario:



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

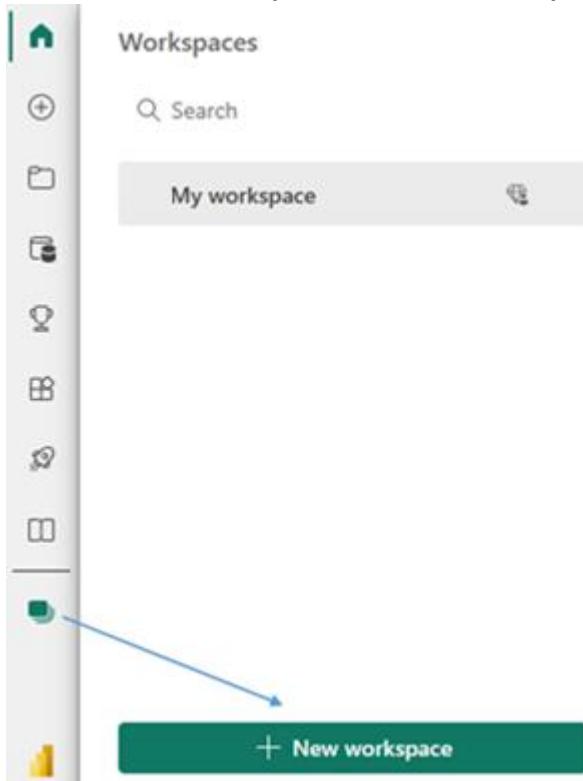
Before you can begin building the lakehouse, you will need to create a workspace where you will build out the remainder of the tutorial.

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.

Create a workspace

X

Name *

Fabric Lakehouse Tutorial

✓ This name is available

Description

This workspace contains all the items for the lakehouse tutorial.

Domain ⓘ

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 

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

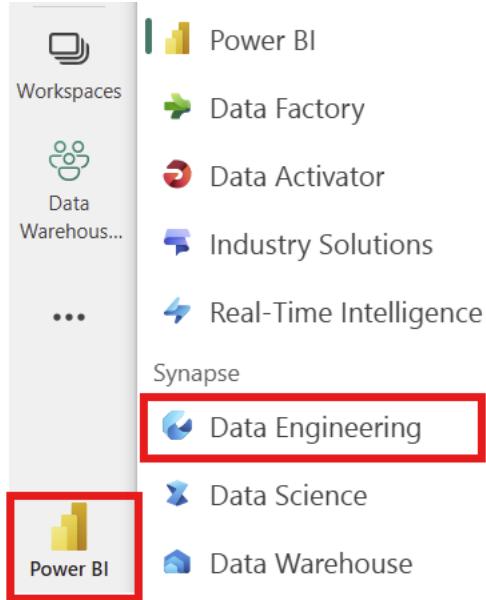
4. Select **Apply**. The workspace will be created and opened.

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

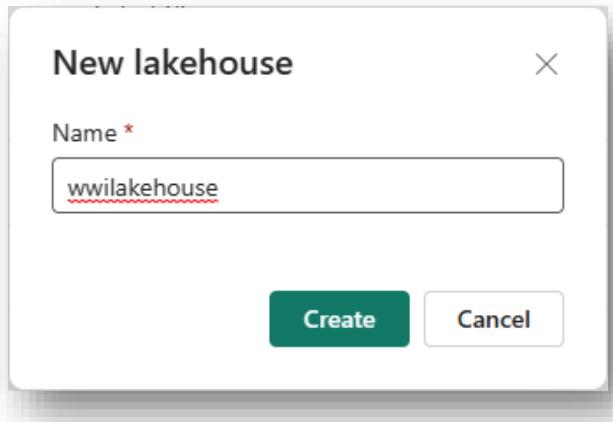
1. In the [Power BI service](#) select **Workspaces** in the left-hand menu.
2. Search for your workspace([Fabric Lakehouse Tutorial](#)) by typing in the search textbox at the top and click on your workspace to open it.
3. From the workload switcher located at the bottom left of the screen, select **Data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



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

The screenshot shows the Microsoft Fabric homepage. On the left is a vertical sidebar with icons for Home, Create, Browse, OneLake data hub, Monitor, Real-Time hub, Workspaces, and Fabric Lakehouse... The main area has a title 'Welcome to Microsoft Fabric' and a sub-section 'Get started with Synapse Data Engineering'. Below this is a row of 'Recommended items to create': 'Lakehouse' (selected and highlighted with a red box), 'Notebook', 'Environment', 'Spark Job Definition', 'Data pipeline', 'Import notebook', and 'Use a sample'. A note below says 'Current workspace: Fabric Lakehouse Tutorial' and 'Items will be saved to this workspace.' At the bottom, there's a section titled 'Learn more about Synapse Data Engineering' with four cards: 'What's a lakehouse?', 'Get data experience in lakehouse', 'Get started with Spark Job Definitions', and 'Develop and execute notebooks'.

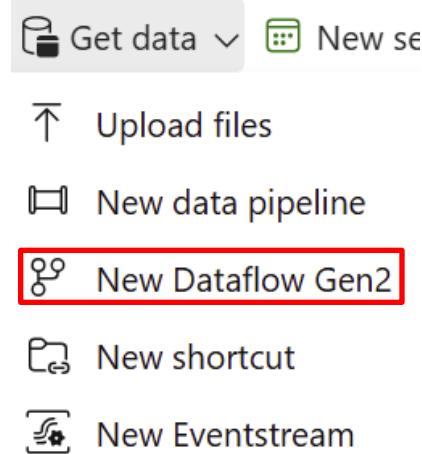
5. Enter **wwilakehouse** in the **Name** box. Please do not check **Lakehouse schemas** checkbox if its enabled for you Lakehouse schemas (Public Preview) ⓘ



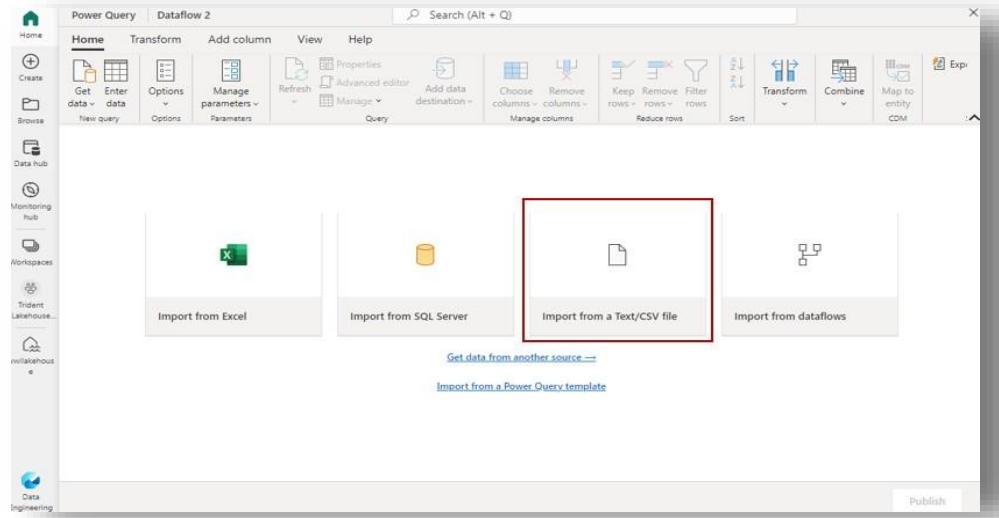
6. Click **Create**. The new lakehouse will be created and automatically opened.

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

- 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.

The screenshot shows the 'Connect to data source' wizard. In the 'Connection settings' section, the 'Upload file' radio button is selected, and a CSV file named 'dimension_customer 1.csv' is shown as uploaded successfully (65.7 KB). In the 'Connection credentials' section, there are fields for 'Connection name' (set to 'Connection'), 'Data gateway' (set to '(none)'), and 'Authentication kind' (set to 'Organizational account').

- 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.

Preview file data

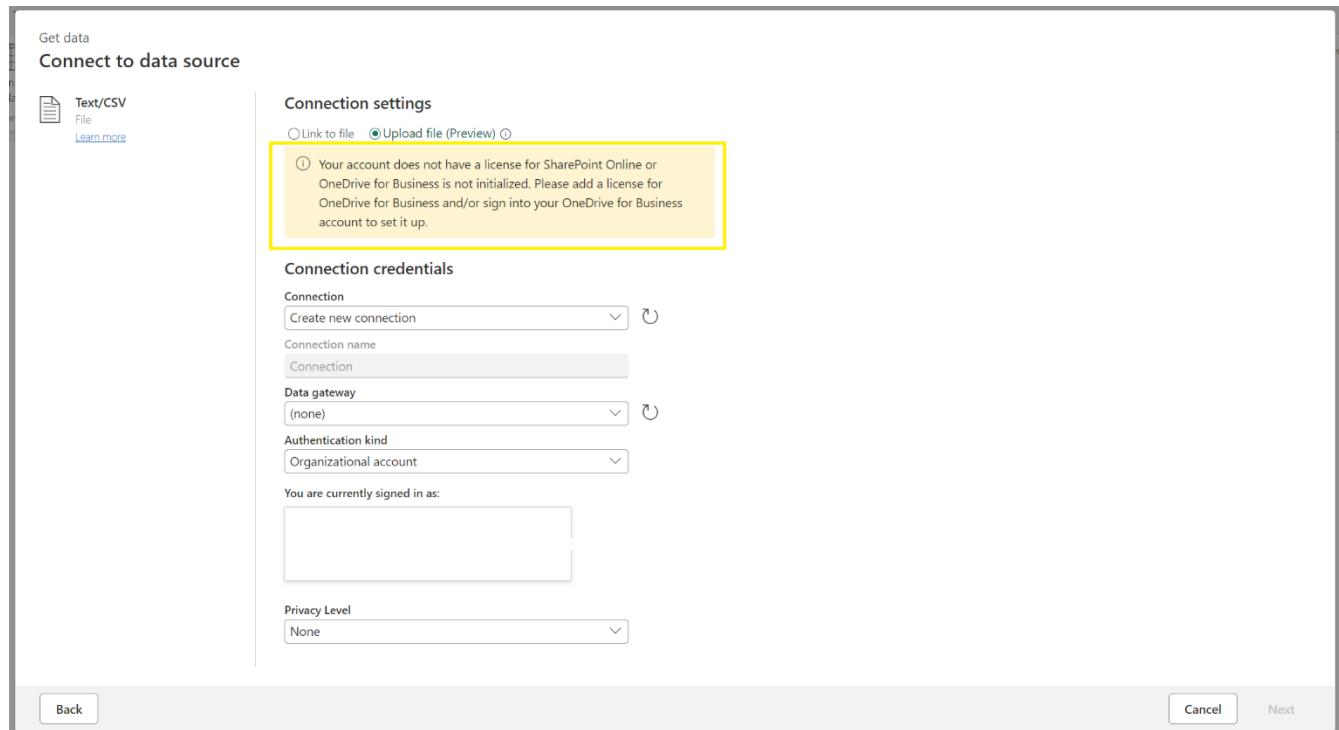
URL: https://microsoft-my.sharepoint.com/personal/arali_microsoft_com/Documents/Apps/Microsoft Power Query/Uploaded Files/dimension_customer 7.csv

File origin	Delimiter	Data type detection
65001: Unicode (UTF-8)	Comma	Based on first 200 rows
1 CustomerKey 2 WWICustomerID 3 Customer		
0 0 Unknown	N/A	N/A N/A N/A
234 433 Wingtip Toys (Baliko, OK)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys Adirake Saenamuang
235 434 Wingtip Toys (Lime Lake, NY)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys Tarja Penttilä
236 435 Wingtip Toys (Teutopolis, IL)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys Devendranath Huq
237 436 Wingtip Toys (Gargatha, VA)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys Enes Olsson
238 437 Wingtip Toys (Cedogian, PA)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys Vanja Grgic
239 438 Wingtip Toys (Lucasville, OH)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys Dayaram Mishra
240 439 Wingtip Toys (Clyon, WI)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys Mina Omidzadeh
241 440 Wingtip Toys (Asher, OK)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys Kadir Useruluy
242 441 Wingtip Toys (Keosauqua, IA)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys Kaisa Jakobsson
243 442 Wingtip Toys (Homer City, PA)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys Rahul Ghate
244 443 Wingtip Toys (Berville, MI)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys Lien Banh
245 444 Wingtip Toys (Tei, SD)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys Divyendu Chakraborty
246 445 Wingtip Toys (Dacono, CO)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys An Dung Ngo
247 446 Wingtip Toys (Saint Landry, LA)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys Chetana Kamath
248 447 Wingtip Toys (Cain, IA)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys Premwadee Saowaluk
249 448 Wingtip Toys (Salt Wells, NV)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys Jae-Hwa Min
250 449 Wingtip Toys (Delray, WV)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys Catalina Nechita
251 450 Wingtip Toys (Mount Summit, IN)	Wingtip Toys (Head Office)	Novelty Shop Wingtip Toys Constanza Laureano



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.



Explorer

wwilakehouse

Tables

Files

Get data in your lakehouse

Upload files
Upload data from your local machine.

Start with sample data
Automatically import tables filled with sample data.

New shortcut
Access data that resides in an external lake.

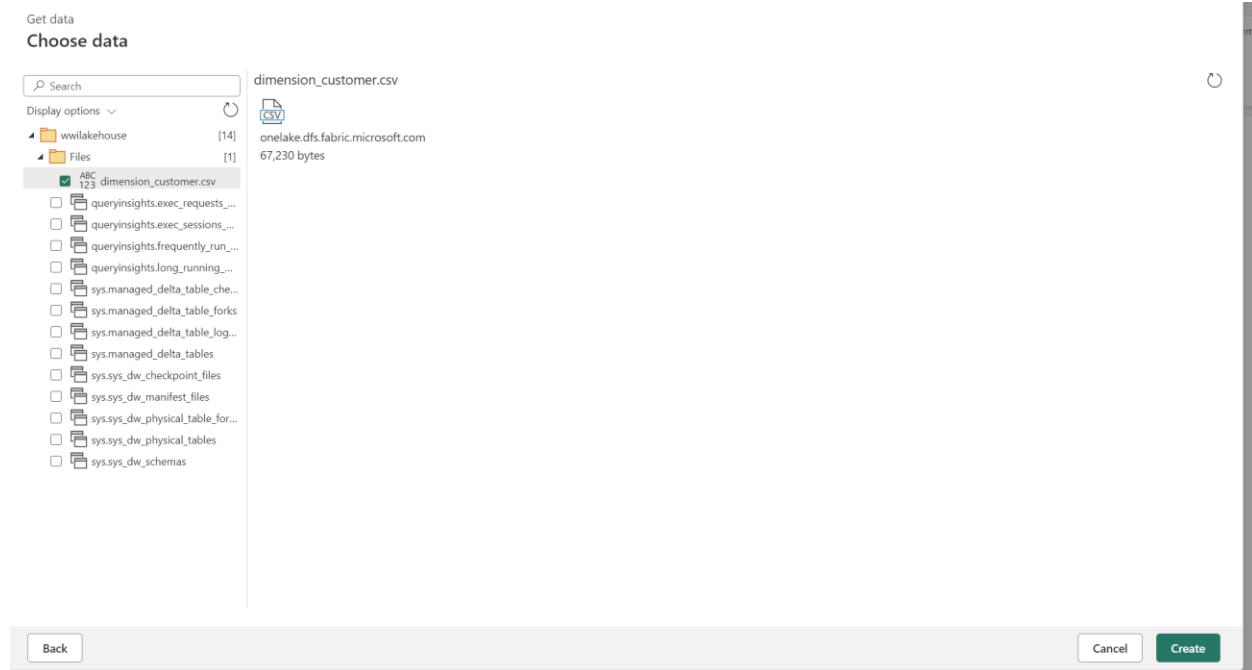
New Dataflow Gen2
Prep, clean, transform, and ingest data.

New data pipeline
Ingest data at scale and schedule data workflows.

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.

Name	Type	Owner	Refreshed	Location	Endorsement	Sensitivity
wwilakehouse	Lakehouse	System Administrator	—	Fabric Lakehouse Tu...	—	—
DataflowsStagingWarehouse	Warehouse	System Administrator	—	Fabric Lakehouse Tu...	—	—

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

Use first row as headers
headers.

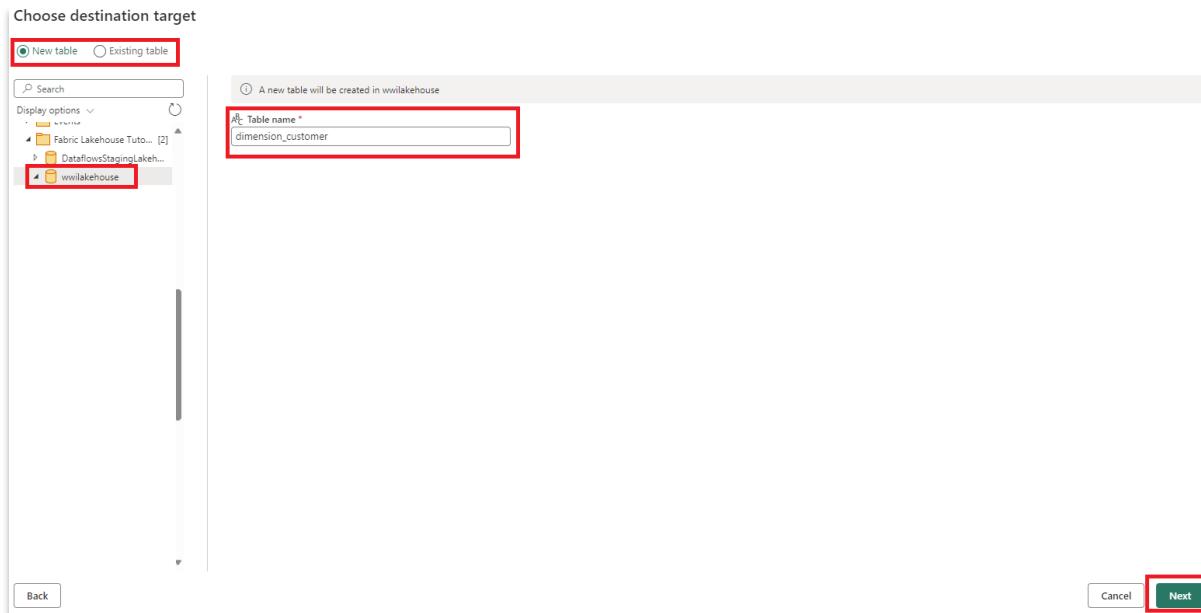
CustomerKey	WWICustomerID	Customer	BillToCustomer	Category	BuyingGroup	PrimaryContact	PostalCode
0	0	Unknown	N/A	N/A	N/A	N/A	N/A
234	433	Wingtip Toys (Balko, OK)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Adrakira Saenamuang	90584
235	434	Wingtip Toys (Lime Lake, NY)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Tarja Penttilä	90667
236	435	Wingtip Toys (Tetropolis, IL)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Devendranath Huq	90269
237	436	Wingtip Toys (Gargatha, VA)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Enes Olsson	90752
238	437	Wingtip Toys (Cadogan, PA)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Vanja Grgic	90357
239	438	Wingtip Toys (Lucasville, OH)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Dayaram Mishra	90391
240	439	Wingtip Toys (Cylon, WI)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Mina Omizadeh	90266
241	440	Wingtip Toys (Asher, OK)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Kadir Usenuly	90683
242	441	Wingtip Toys (Keoaauqua, IA)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Kaja Jacobsson	90281
243	442	Wingtip Toys (Homer City, PA)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Rahul Ghate	90713
244	443	Wingtip Toys (Benville, MI)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Lien Banh	90628
245	444	Wingtip Toys (Tea, SD)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Divyendu Chakraborty	90641
246	445	Wingtip Toys (Dacono, CO)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	An Dung Ngo	90289
247	446	Wingtip Toys (Saint Landry, LA)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Chetana Kamath	90033
248	447	Wingtip Toys (Cohn, IA)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Premvadhee Saevaluk	90458
249	448	Wingtip Toys (Salt Wells, NV)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Jae-Hwa Min	90058
250	449	Wingtip Toys (Delray, WV)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Catalina Nechita	90317
251	450	Wingtip Toys (Mount Summit, IN)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Constanza Laureano	90142
252	451	Wingtip Toys (Smooth, WI)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Aakriti Bhamidipati	90071
253	452	Wingtip Toys (Lake Davis, CA)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Himadri PrabhuPAda	90719
254	453	Wingtip Toys (Standardsville, VA)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Bryan Helms	90739
255	454	Wingtip Toys (Corcovado, PR)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Harri Kask	90498
256	455	Wingtip Toys (Taft Heights, CA)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Ilse Sandell	90304
257	456	Wingtip Toys (West Hempstead, NY)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Subhash Matondkar	90799
258	457	Wingtip Toys (Portales, NM)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Eesvaranu Ankitham	90304
259	458	Wingtip Toys (Coker, AL)	Wingtip Toys (Head Office)	Novelties Shop	Wingtip Toys	Miquel Paez	90364

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.

The screenshot shows the Microsoft Power Query Editor interface. On the left, there's a navigation bar with options like Home, Transform, Add column, View, Help, Get data, Manage connections, Options, Manage parameters, Refresh, Advanced editor, Add data destination, Choose columns, Remove columns, Manage rows, Keep rows, Remove rows, Filter rows, Reduce rows, Suggested transforms, Split, Group, Use first row as headers, Merge queries, Append queries, Map to entity, Combine files, Copilot, and Export template. The main area is titled 'Queries [1]' and contains a table with 31 rows of data. The columns are CustomerKey, CustomerID, Customer, BillToCustomer, Category, BuyingGroup, PrimaryContact, PostalCode, and Validation. The 'Data destination' section on the right is expanded, showing 'Name' set to 'dimension_customer', 'Entity type' as 'Custom', and 'Applied steps' containing 'Promoted' and 'Changed'. The 'Data destination' dropdown is set to 'Lakehouse' with a red box around it. At the bottom, there are 'Step' and 'Publish' buttons.

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" **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 ?

Source	Source type	Destination	Destination type
CustomerKey	1-3 Whole number	CustomerKey	Whole number
WWICustomerID	1-3 Whole number	WWICustomerID	Whole number
Customer	A-B Text	Customer	Text
BillToCustomer	A-B Text	BillToCustomer	Text
Category	A-B Text	Category	Text
BuyingGroup	A-B Text	BuyingGroup	Text
PrimaryContact	A-B Text	PrimaryContact	Text
PostalCode	A-B Text	PostalCode	Text
ValidFrom	Date/Time	ValidFrom	Date/Time
ValidTo	Date/Time	ValidTo	Date/Time
LineageKey	1-3 Whole number	LineageKey	Whole number

Back Cancel Save settings

Choose destination settings

i 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

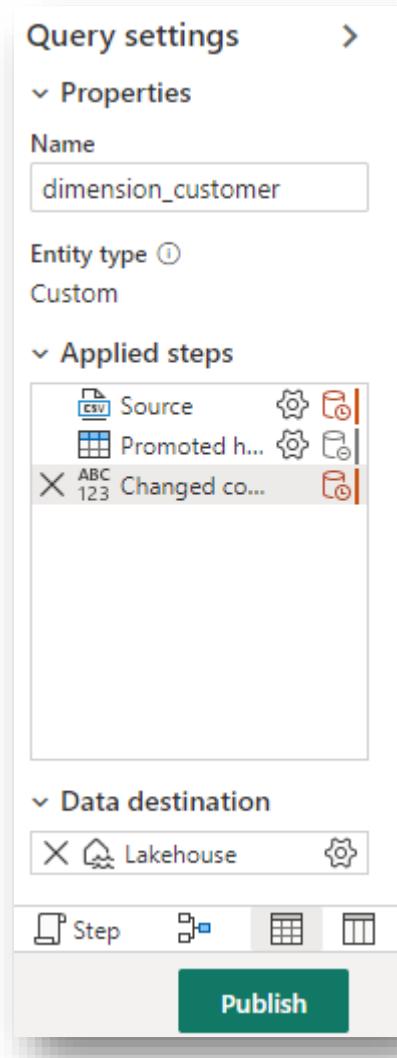
Schema options on publish

Existing schema → Dynamic schema Fixed schema

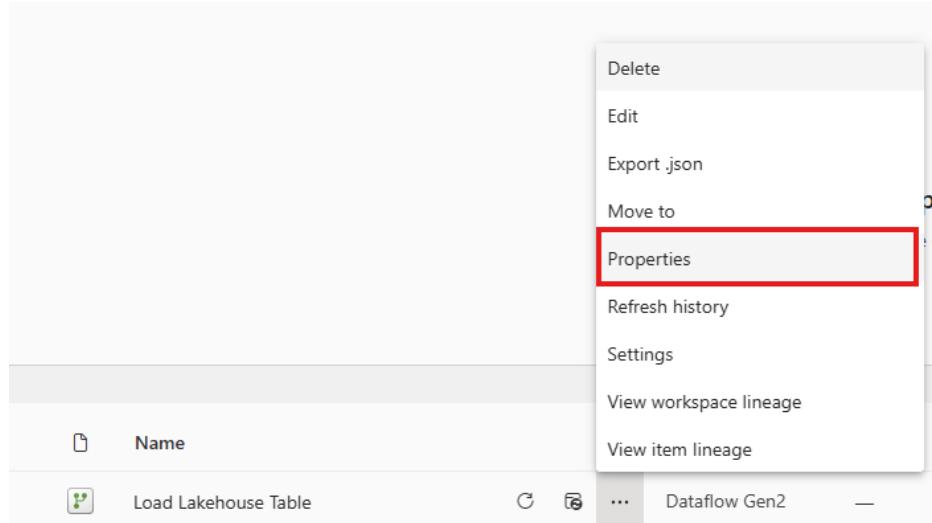
Column mapping

Source	Source type	Destination	Destination type
CustomerKey	1-3 Whole number	CustomerKey	Whole number
WWICustomerID	1-3 Whole number	WWICustomerID	Whole number
Customer	A-B Text	Customer	Text
BillToCustomer	A-B Text	BillToCustomer	Text
Category	A-B Text	Category	Text
BuyingGroup	A-B Text	BuyingGroup	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.

A screenshot of the Azure Data Factory artifact view showing a list of data flows. The columns are Name and Type. The data flows listed are:

Name	Type
Load Lakehouse Table	Dataflow Gen2
wwilakehouse	Lakehouse
wwilakehouse	Semantic model
wwilakehouse	SQL analytics enc

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.

The screenshot shows the Azure Data Lake Storage Explorer interface. On the left, there is a navigation sidebar with icons for Home, Create, Browse, Apps, Metrics, Monitoring hub, and Deployment pipelines. The main area is titled 'Explorer' and shows a tree view of 'wwilakehouse' with 'Tables' and 'Files' nodes. A context menu is open over the 'dimension_customer' table, with a red box highlighting the 'Refresh' button. To the right, a large table view displays the 'dimension_customer' data with 403 rows. At the top right of the table view, there is a 'Lakehouse' card with a 'SQL analytics endpoint' button, which is also highlighted with a red box. The table columns include CustomerKey, WWICustomerID, Customer, BillToCustomerID, Category, BuyingGroup, PrimaryContactName, PostalCode, ValidFrom, ValidTo, and LineageKey.

CustomerKey	WWICustomerID	Customer	BillToCustomerID	Category	BuyingGroup	PrimaryContactName	PostalCode	ValidFrom	ValidTo	LineageKey
1	0	Unknown	N/A	N/A	N/A	N/A	N/A	1/1/2013 12:00...	1/1/2025 12:00...	0
2	102	Tailsin Toys (Fieldbr...	Tailsin Toys (He...	Novelty Shop	Kids Toys	Tea Koppel	90205	1/1/2013 12:00...	1/1/2025 12:00...	2
3	103	Tailsin Toys (Kalevst...	Tailsin Toys (He...	Novelty Shop	Kids Toys	Naseem Radan	90130	1/1/2013 12:00...	1/1/2025 12:00...	2
4	104	Tailsin Toys (Wallagr...	Tailsin Toys (He...	Novelty Shop	Kids Toys	Laboni Deb	90579	1/1/2013 12:00...	1/1/2025 12:00...	2
5	105	Tailsin Toys (Tommol...	Tailsin Toys (He...	Novelty Shop	Kids Toys	Sung-Hwa Hwa...	90400	1/1/2013 12:00...	1/1/2025 12:00...	2
6	106	Tailsin Toys (Tunaca...	Tailsin Toys (He...	Novelty Shop	Kids Toys	Shiva Pipalia	90662	1/1/2013 12:00...	1/1/2025 12:00...	2
7	107	Tailsin Toys (Glen Av...	Tailsin Toys (He...	Novelty Shop	Kids Toys	Karie Mercier	90782	1/1/2013 12:00...	1/1/2025 12:00...	2
8	108	Tailsin Toys (Bernie ...	Tailsin Toys (He...	Novelty Shop	Kids Toys	Bhanu Thota	90045	1/1/2013 12:00...	1/1/2025 12:00...	2
9	109	Tailsin Toys (South L...	Tailsin Toys (He...	Novelty Shop	Kids Toys	Ae-Cha Joo	90247	1/1/2013 12:00...	1/1/2025 12:00...	2
10	110	Tailsin Toys (North C...	Tailsin Toys (He...	Novelty Shop	Kids Toys	Dinara Saparkzy	90792	1/1/2013 12:00...	1/1/2025 12:00...	2
11	111	Tailsin Toys (Oriole ...	Tailsin Toys (He...	Novelty Shop	Kids Toys	Adam Dvorak	90436	1/1/2013 12:00...	1/1/2025 12:00...	2

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.

The screenshot shows the Power BI Data view interface. At the top, there are navigation links: Home, Reporting, New SQL query (which is highlighted with a red box), New visual query, New report, and New measure. A status message below the navigation bar states: "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](#)".

The main area is divided into two sections: "Explorer" on the left and "Data preview" on the right. The "Explorer" section shows a hierarchical tree of database objects. Under "Warehouses", there is "wwilakehouse" which contains "Schemas" (dbo, INFORMATION_SCHEMA, sys, Security), "Views", "Functions", "Stored Procedures", "guest", "queryinsights", and "Queries" (My queries, Shared queries). Under "Tables", the "dimension_customer" table is selected and highlighted with a red box. The "Data preview" section shows a grid of data for the "dimension_customer" table, with columns: CustomerKey, WWICustomerID, Customer, BillToCustomer, and ABC. The data consists of 29 rows, each representing a customer entry with details like customer ID, name, and location.

	CustomerKey	WWICustomerID	Customer	BillToCustomer	ABC
1	0	0	Unknown	N/A	N/A
2	102	102	Tailspin Toys (Fieldbrook, CA)	Tailspin Toys (Head Office)	Nov
3	103	103	Tailspin Toys (Kalvesta, KS)	Tailspin Toys (Head Office)	Nov
4	104	104	Tailspin Toys (Wallgrass, ME)	Tailspin Toys (Head Office)	Nov
5	105	105	Tailspin Toys (Tommolen, MS)	Tailspin Toys (Head Office)	Nov
6	106	106	Tailspin Toys (Tumacacor, AZ)	Tailspin Toys (Head Office)	Nov
7	107	107	Tailspin Toys (Glen Avon, CA)	Tailspin Toys (Head Office)	Nov
8	108	108	Tailspin Toys (Bemie, MO)	Tailspin Toys (Head Office)	Nov
9	109	109	Tailspin Toys (South Laguna, CA)	Tailspin Toys (Head Office)	Nov
10	110	110	Tailspin Toys (North Crows Nest, IN)	Tailspin Toys (Head Office)	Nov
11	111	111	Tailspin Toys (Oriole Beach, FL)	Tailspin Toys (Head Office)	Nov
12	112	112	Tailspin Toys (Sallyards, KS)	Tailspin Toys (Head Office)	Nov
13	113	113	Tailspin Toys (Dahlia, NM)	Tailspin Toys (Head Office)	Nov
14	114	114	Tailspin Toys (Cherry Grove Beach, SC)	Tailspin Toys (Head Office)	Nov
15	115	115	Tailspin Toys (Bethania, NC)	Tailspin Toys (Head Office)	Nov
16	116	116	Tailspin Toys (Rafael Capó, PR)	Tailspin Toys (Head Office)	Nov
17	79	79	Tailspin Toys (Page City, KS)	Tailspin Toys (Head Office)	Nov
18	80	80	Tailspin Toys (Valdese, NC)	Tailspin Toys (Head Office)	Nov
19	81	81	Tailspin Toys (Big Moose, NY)	Tailspin Toys (Head Office)	Nov
20	82	82	Tailspin Toys (La Cueva, NM)	Tailspin Toys (Head Office)	Nov
21	83	83	Tailspin Toys (Absecon, NJ)	Tailspin Toys (Head Office)	Nov
22	84	84	Tailspin Toys (Aeitunas, PR)	Tailspin Toys (Head Office)	Nov
23	85	85	Tailspin Toys (Andrix, CO)	Tailspin Toys (Head Office)	Nov
24	86	86	Tailspin Toys (New Lexington, OH)	Tailspin Toys (Head Office)	Nov
25	87	87	Tailspin Toys (Sauquoit, NY)	Tailspin Toys (Head Office)	Nov
26	88	88	Tailspin Toys (Dracut, MA)	Tailspin Toys (Head Office)	Nov
27	89	89	Tailspin Toys (Victory Gardens, NJ)	Tailspin Toys (Head Office)	Nov
28	90	90	Tailspin Toys (Tolna, ND)	Tailspin Toys (Head Office)	Nov
29	91	91	Tailspin Toys (Alstead, NH)	Tailspin Toys (Head Office)	Nov

At the bottom of the interface, there are tabs: Data, Query, and Model. A success message "Succeeded (4 sec 80 ms)" is displayed at the bottom center.

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
```

The screenshot shows the Power BI Data Editor interface. On the left is the 'Explorer' pane, which lists various databases, schemas, and tables. In the center is the 'Query' editor pane, containing a SQL query. A red box highlights the 'Run' button in the toolbar above the query text. Below the query is a results grid showing data from the 'dimension_customer' table. At the bottom of the editor, there's a context menu for the selected query, with a red box highlighting the 'Run' option. The status bar at the bottom indicates the query succeeded in 1 second.

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

The screenshot shows the Power BI Model view. The top navigation bar has 'Reporting' highlighted in yellow. The main area displays three semantic model objects: 'dimension_customer', 'exec_requests_history', and 'exe'. The 'dimension_customer' object is expanded, showing its columns: BillToCustomer, BuyingGroup, Category, Customer, CustomerKey, LineageKey, PostalCode, PrimaryContact, and ValidFrom. The 'Model' tab is selected at the bottom of the interface. The status bar at the bottom right shows 'Columns: 2 Rows: 5'.

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

The screenshot shows a dialog box titled "Manage default semantic model". It contains a search bar with the placeholder "Search" and a filter icon. Below the search bar is a checkbox labeled "Select all" which is checked. A second checkbox is checked, and its label is "dimension_customer", preceded by a table icon. At the bottom right of the dialog are two buttons: "Confirm" (in green) and "Cancel".

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.

The screenshot shows the item view of a workspace. At the top, there are buttons for "+ New", "Upload", "Create deployment pipeline", "Create app", and a settings icon. Below this is a table with columns "Name" and "Type". There are four items listed:

Name	Type
Load Lakehouse Table	Dataflow Gen2
wwilakehouse	Lakehouse
wwilakehouse	Semantic model (...)
wwilakehouse	SQL analytics end...

The fourth row, which contains the "wwilakehouse" semantic model, has a red box drawn around the "Semantic model (...)" column.

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.

The screenshot shows the Power BI Semantic Model interface. On the left, there's a navigation bar with icons for Home, Create, Browse, OneLake data hub, Monitoring hub, Workspaces, Fabric Lakehouse, and two specific items for 'wwilakehouse'. The main area displays 'Details for wwilakehouse' with fields for Location (Fabric Lakehouse Tuto...), Refreshed (1/26/24, 2:17:58 PM), and Sensitivity (Confidential\Microsoft Extended). Below this are sections for 'Discover business insights' (with a 'Explore this data' button) and 'Share this data' (with a 'Share semantic model' button). A red box highlights the 'Auto-create a report' button in the 'Discover business insights' section. To the right, a 'Tables' pane lists various tables like dimension_customer, CustomerKey, WWICustomerID, etc., with checkboxes next to them. A tooltip above the pane says: 'To select more than one table, and view summarized data, create a paginated report.' A 'Create paginated report' button is also visible.

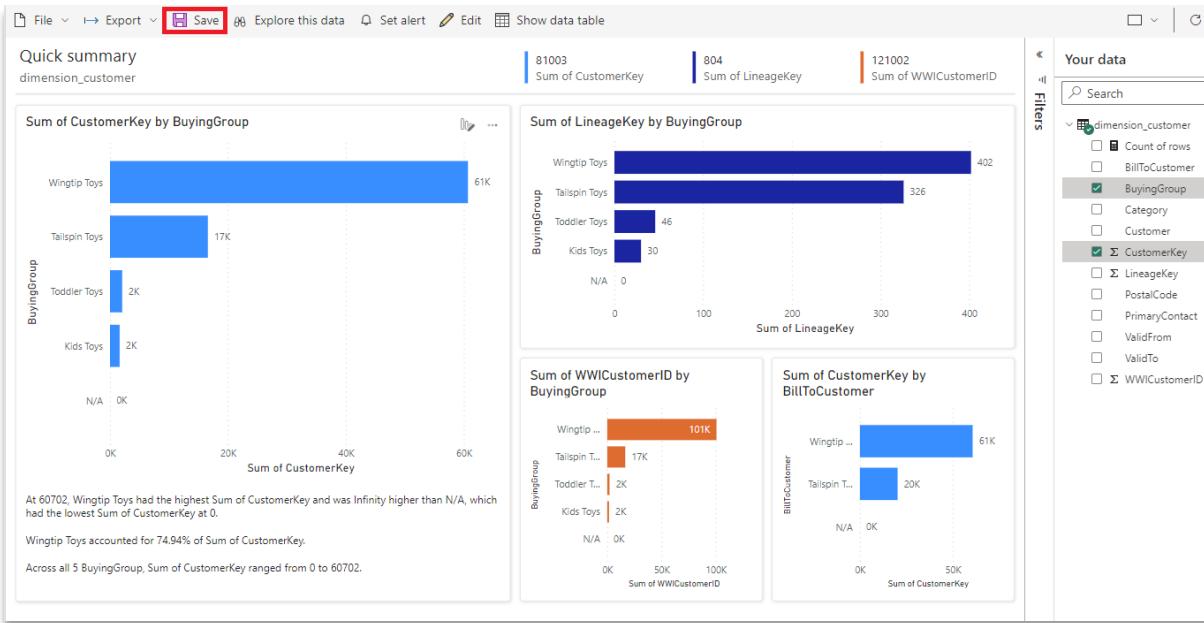
3. Since the table is a dimension and there are no measures in it, Power BI smartly creates a measure for the row sum and aggregates it across different columns and creates different charts as below.

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.

The screenshot shows the Power BI Report view. At the top, there's a ribbon with File, Export, Save, Explore this data, Set alert, Edit, and Show data table. A message box in the top right corner says 'Your report is ready' with options to 'View report now' or 'View it now, or it will auto-load in a few seconds.' Below the ribbon, there's a large yellow bar chart placeholder. At the bottom, there's a note: 'Your report is ready. Hint: To see the best results, try pre-selecting the data you're most interested in.' and a 'Pre-select data' button.



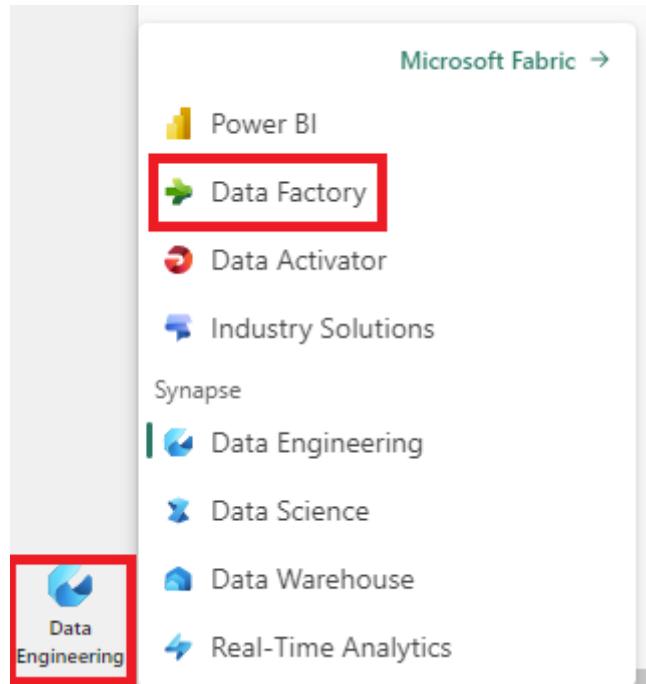
Module 3: Ingest, Prep and Analyze

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.

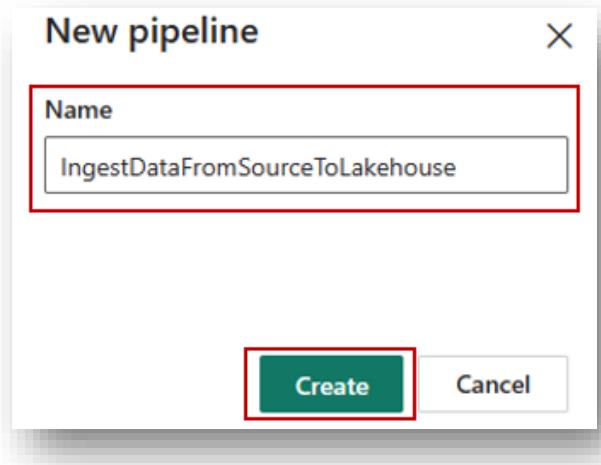
1. On the bottom left of the screen, click on the workload switcher, and then select **Data Factory**.



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

The screenshot shows the Microsoft Fabric Welcome screen. It features a 'Welcome to Microsoft Fabric' header, a 'Get started with Synapse Data Engineering' section, and a 'Recommended items to create' section. This section contains seven items: Lakehouse, Notebook, Environment, Spark Job Definition, Data pipeline, Import notebook, and Use a sample. The 'Data pipeline' item is highlighted with a yellow box.

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

Build a data pipeline to organize and move your data

The screenshot shows the Microsoft Fabric data pipeline builder interface. It features four main options:

- Start with a blank canvas**: Shows a pipeline activity icon and a brief description: "Pipeline activity. Automate data orchestrations using rich no-code activities."
- Start with guidance**: Shows a copy data assistant icon and a brief description: "Copy data assistant. Follow guided steps to copy data into Microsoft Fabric, as well as other data stores." This option is highlighted with a yellow box.
- Practice with sample data**: Shows a practice icon and a brief description: "Practice with sample data. Quickly build a data pipeline with a predefined template to load data into Lakehouse."
- Templates**: Shows a templates icon and a brief description: "Templates. Generate a new data pipeline quickly using a predefined data scenario."

At the bottom, there is a link: "Need help? [Watch a demo](#)".

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

The screenshot shows the 'Copy data into Lakehouse' wizard. The current step is 'Choose data source'. A red box highlights the 'Http' connector in the search results.

- 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/ww-sample-dataset.zip
Connection	Create a new connection
Connection name	wwisampled
Data gateway	None
Authentication kind	Anonymous

The screenshot shows the 'Connect to data source' window. The 'Connection settings' section has 'Url' set to <https://assetsprod.microsoft.com/en-us/ww-sample-datas...>. The 'Connection credentials' section has 'Connection' set to 'Create new connection' and 'Connection name' set to 'wwisampled'. The 'Next' button is highlighted with a red box.

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**.

Copy data into Lakehouse

Connect to data source

Choose data destination

Review + save

Connect to data source

Connection: wwiisampleddata

Base URL

Relative URL:

Request method: GET

Additional headers:

Binary copy **Compression type**: ZipDeflate (.zip)

Compression level:

Request timeout:

Max concurrent connections:

Back **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.

Copy data

Choose data source

Connect to data source

Choose data destination

Define the data store as destination.

Connect to data destination

Review + save

Home **OneLake data hub** **New** **Azure** **New Fabric item** **X**

Search

All **My data** **Endorsed in your org** **Favorites**

Name	Type	Owner	Refreshed	Location
wwilakehouse	Lakehouse	System Administrator	—	Fabric Lakeho
DataflowsStagingLakehouse	Lakehouse	System Administrator	—	Fabric Lakeho

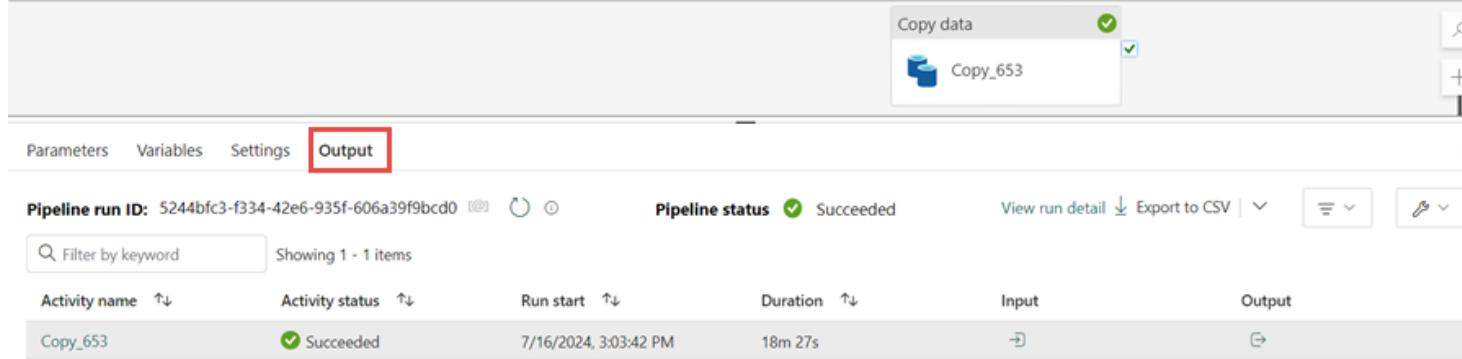
Back

The screenshot shows the 'Copy data into Lakehouse' wizard. On the left, a vertical navigation path indicates the current step: 'Choose data source' (done), 'Connect to data source' (done), 'Connect to data destination' (selected), and 'Review + save' (not yet started). The main panel is titled 'Connect to data destination'. It shows a connection named 'wwilakehouse'. Under 'Root folder', the 'Files' radio button is selected, while 'Tables' is unselected. A note below says: 'If the identity you use to access the data store only has permission to subdirectory instead of the entire account, specify the path to browse.' There are fields for 'File name' and 'Copy behavior'. At the bottom right are 'Back' and 'Next' buttons, with 'Next' being highlighted.

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.

The screenshot shows the 'Copy data into Lakehouse' wizard. The navigation path on the left is identical to the previous screenshot. The main panel is titled 'Connect to data destination'. The 'File format' dropdown is set to 'Binary', which is highlighted with a red box. Below it, the 'Compression type' dropdown is set to 'Select...'. At the bottom right are 'Back' and 'Next' buttons, with 'Next' being highlighted.

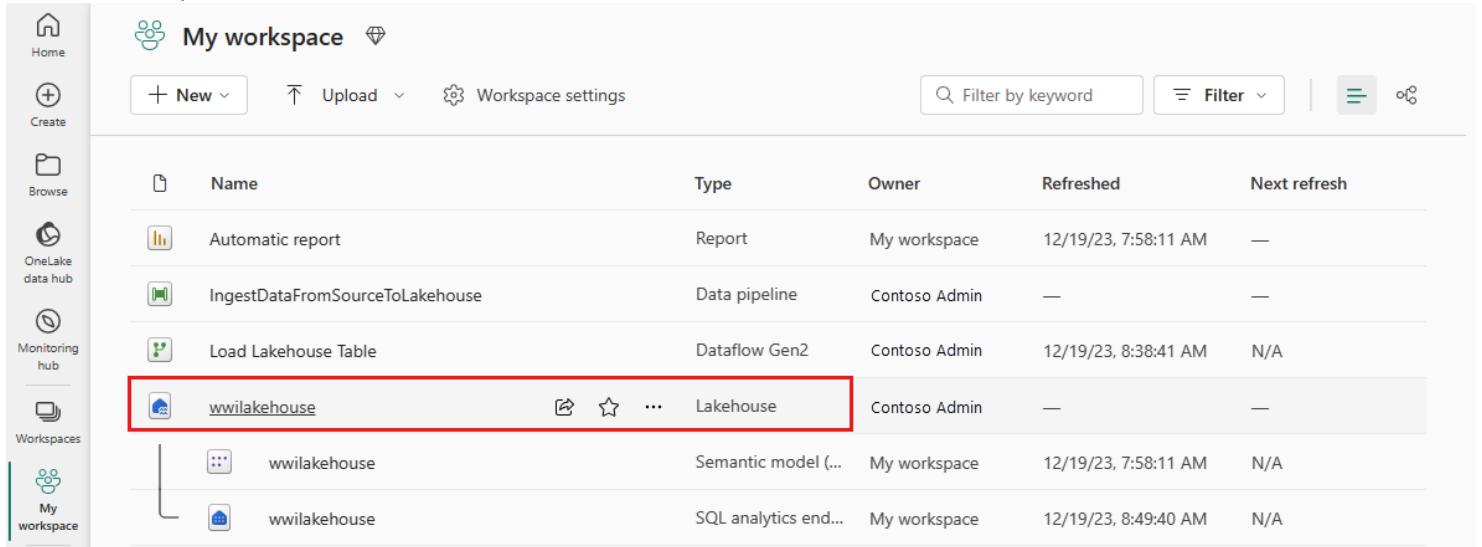
9. You can monitor the pipeline execution and activity in the **Output** tab. You can also view detailed data transfer information by selecting the glasses icon next to the pipeline name, which appears when you hover over the name. **This step may take 15-30 mins depending on fabric region.**



The screenshot shows the 'Output' tab of a pipeline run details page. At the top, there's a summary bar with a 'Copy data' status (green checkmark) and a 'Copy_653' activity. Below it, the 'Output' tab is selected. The main area displays a table of activities:

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

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



The screenshot shows the 'My workspace' section of the Azure Data Explorer interface. On the left, there's a sidebar with navigation links: Home, Create, Browse, OneLake data hub, Monitoring hub, Workspaces, and My workspace (which is currently selected). The main area shows a table of workspaces:

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

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 OneLake

A SQL analytics endpoint for SQL querying and a default Power BI semantic model for reporting were created with this dataset.

Explorer

wwilakehouse

Tables

dimension_customer

Files

...

Files

1ca6dd1e-7bf4-48

New shortcut

New subfolder

Upload >

Properties

Refresh

The screenshot shows the Azure Data Lake Storage Explorer interface. In the main pane, there's a tree view under 'wwilakehouse' with 'Tables' expanded, showing 'dimension_customer'. Below it, 'Files' is selected. A context menu is open over the 'Files' folder, listing options like 'New shortcut', 'New subfolder', 'Upload', 'Properties', and 'Refresh'. The 'Refresh' option is highlighted with a red box.

Explorer

- wwlakehouse
 - Tables
 - Files
- 9952f0a5-eda4-4f22-b4c5-8975cefe147a
 - WideWorldImportersDW
 - csv
 - full

The 'full' folder under 'csv' is highlighted with a red box.

Files > 9952f0a5-eda4-4f22-b4c5-8975cefe147a > WideWorldImportersDW > csv > full

Name	Date modified	Type	Size
dimension_city	7/16/2024 3:04:07 PM	Folder	15 items
dimension_customer	7/16/2024 3:04:12 PM	Folder	7 items
dimension_date	7/16/2024 3:04:13 PM	Folder	15 items
dimension_employee	7/16/2024 3:04:14 PM	Folder	5 items
dimension_stock_item	7/16/2024 3:04:14 PM	Folder	6 items
fact_sale	7/16/2024 3:04:15 PM	Folder	15 items
fact_sale_1y_full	7/16/2024 3:09:59 PM	Folder	15 items

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**

Files

- 1ca6dd1e-7bf4-485f-bfe0-0e477b9b2e04
 - WideWorldImportersDW
 - csv
 - parquet
 - tables

Rename folder

Folder name *

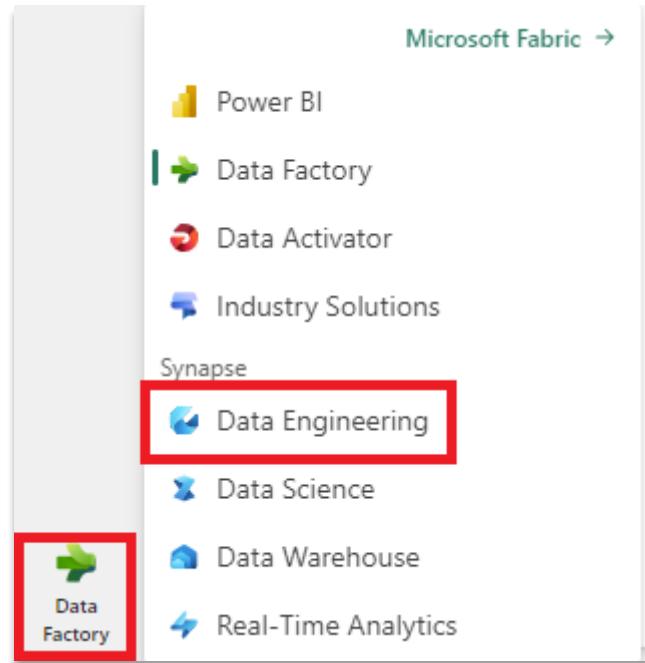
1ca6dd1e-7bf4-485f-bfe0-0e477b9b2e04

Rename Cancel

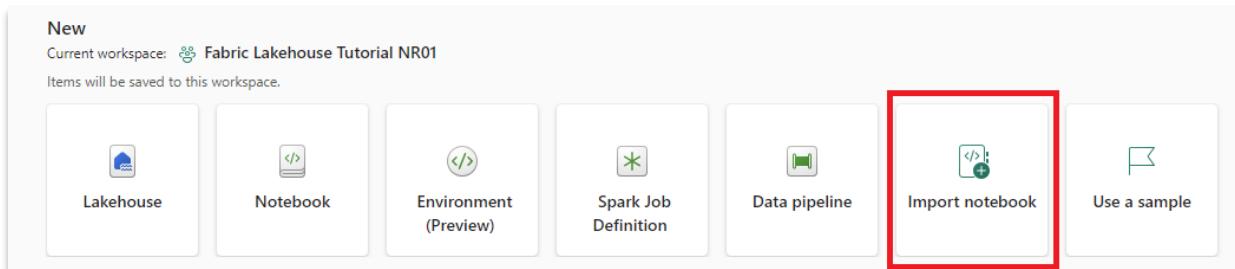
Data Preparation

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.

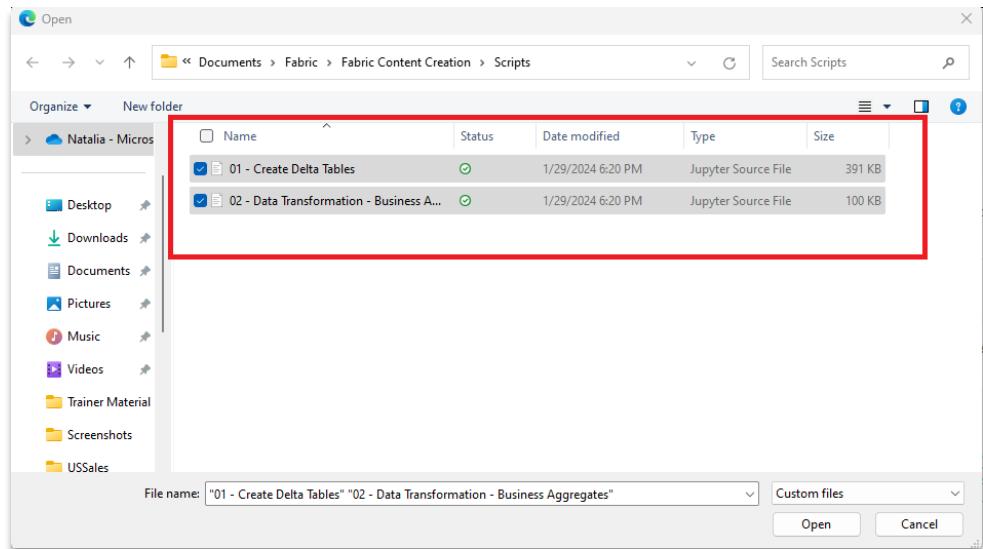
1. Download the set of notebooks found in the Scripts folder
2. From the workload switcher located at the bottom left of the screen, select **Data engineering**.



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.



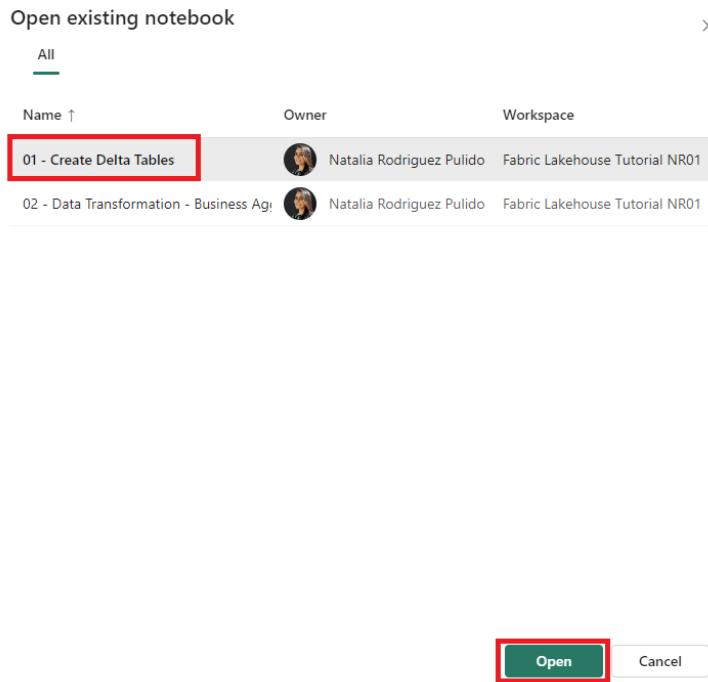
7. After the import of notebooks is successful, you can go to items view of the workspace and see these newly imported notebooks. Click on **wwilakehouse** lakehouse to open it.

	Name	Type	Owner
	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 ...
	report1	Report	Fabric Lakehouse ...
	wwilakehouse	Lakehouse	Natalia Rodriguez ...
	wwilakehouse	Semantic model (...)	Fabric Lakehouse ...
	wwilakehouse	SQL analytics end...	Fabric Lakehouse ...

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

The screenshot shows the Power BI Home ribbon with the 'Existing notebook' option highlighted. The 'Explorer' pane on the left shows a tree structure with 'wwilakehouse' expanded, showing 'Tables' and 'Files'. The 'fact_sa' file is selected in the 'Files' section. The 'Name' column in the preview pane shows '_SUCCESS' and two part files: 'part-00000-ced648ca-e8c8-46e5-8!' and 'part-00001-ced648ca-e8c8-46e5-8!'.

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.

```
# Copyright (c) Microsoft Corporation.  
# Licensed under the MIT License.  
  
spark.conf.set("spark.sql.parquet.vorder.enabled", "true")  
spark.conf.set("spark.microsoft.delta.optimizeWrite.enabled", "true")  
spark.conf.set("spark.microsoft.delta.optimizeWrite.binSize", "1073741824")
```

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

PySpark (Python)

The first thing you would have noticed when executing this cell is that you didn't have to specify the underlying Spark pool or cluster details as Fabric provides it through the concepts of Live Pool i.e., every Fabric workspace comes pre-wired with a default Spark pool, called Live Pool. This 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).

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 delta table.

```
1  from pyspark.sql.functions import col, year, month, quarter
2
3  table_name = 'fact_sale'
4
5  df = spark.read.format("parquet").load('Files/wwi-raw-data/WideWorldImportersDW/parquet/full/fact_sale_1y_full/')
6  df = df.withColumn('Year', year(col("InvoiceDateKey")))
7  df = df.withColumn('Quarter', quarter(col("InvoiceDateKey")))
8  df = df.withColumn('Month', month(col("InvoiceDateKey")))
9
10 df.write.mode("overwrite").format("delta").partitionBy("Year", "Quarter").save("Tables/" + table_name)
✓ 1 min 24 sec - Command executed in 1 min 24 sec 39 ms by System Administrator on 11:00:42 PM, 8/07/24
```

```
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.

Dimensions

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

```
1  from pyspark.sql.types import *
2
3  def loadFullDataFromSource(table_name):
4      df = spark.read.format("parquet").load('Files/wwi-raw-data/WideWorldImportersDW/parquet/full/' + table_name)
5      df = df.select([c for c in df.columns if c != 'Photo'])
6      df.write.mode("overwrite").format("delta").save("Tables/" + table_name)
7
8  full_tables = [
9      'dimension_city',
10     'dimension_date'],
11     'dimension_employee',
12     'dimension_stock_item'
13 ]
14
15 for table in full_tables:
16     loadFullDataFromSource(table)
✓ 12 sec - Command executed in 12 sec 182 ms by System Administrator on 11:02:24 PM, 8/07/24
```

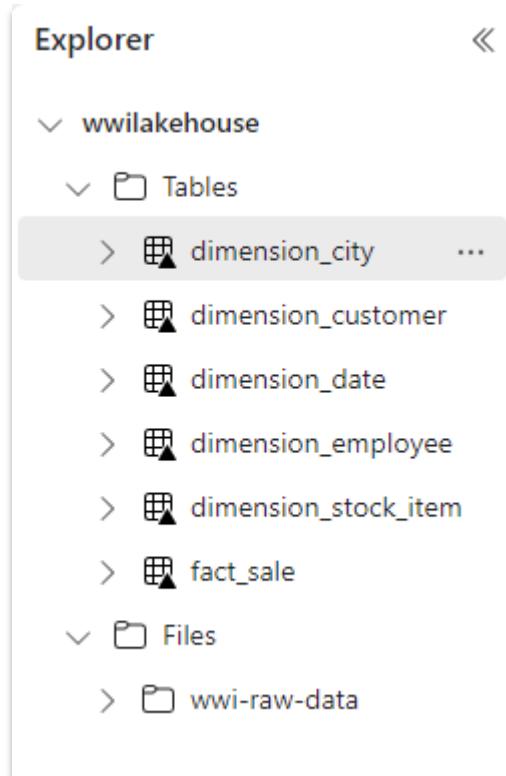
```
from pyspark.sql.types import *

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.

Fabric Lakehouse Tutorial NR01

This workspace contains all the items for the lakehouse tutorial.

Name	Type	Owner	Refreshed
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
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
wwilakehouse	SQL analytics end... SQL analytics end...	Fabric Lakehouse ...	1/29/24, 3:52:33 PM

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

Home

Get data New semantic model Open notebook Existing notebook New notebook

A SQL analytics endpoint for SQL querying and a default P... dimension_city

Explorer

- wwilakehouse
 - Tables
 - dimension_city
 - dimension_customer
 - dimension_date
 - dimension_employee
 - dimension_stock_item
 - fact_sale
 - Files
 - wwi-raw-data

	123 CityKey	123 WWICityID	ABC CityName
1	82085	37232	Willia...
2	90931	25393	Oln...
3	90932	25406	Olym...
4	90933	25417	Omal...
5	90934	25435	Onar...
6	90935	25450	Oneid...
7	90936	25488	Opdy...
8	90937	25506	Oqua...
9	90938	25538	Oran...
10	90939	25567	Orea...
11	90940	25568	Oreg...
12	90941	25587	Orien...

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

Name ↑	Owner	Workspace
01 - Create Delta Tables	Natalia Rodriguez Pulido	Fabric Lakehouse Tutorial NR01
02 - Data Transformation - Business A	Natalia Rodriguez Pulido	Fabric Lakehouse Tutorial NR01

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**.

Home Edit Run View

Run all 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.

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("wwlakehouse.fact_sale")
2 df_dimension_date = spark.read.table("wwlakehouse.dimension_date")
3 df_dimension_city = spark.read.table("wwlakehouse.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 .groupBy("date.Date", "date.CalendarMonthLabel", "date.Day", "date.ShortMonth", "date.CalendarYear", "city.City", "city.StateProvince"),
6 .sum("sale.TotalExcludingTax", "sale.TaxAmount", "sale.TotalIncludingTax", "sale.Profit")\
7 .withColumnRenamed("sum(TotalExcludingTax)", "SumOfTotalExcludingTax")\
8 .withColumnRenamed("sum(TaxAmount)", "SumOfTaxAmount")\
9 .withColumnRenamed("sum(TotalIncludingTax)", "SumOfTotalIncludingTax")\
10 .withColumnRenamed("sum(Profit)", "SumOfProfit")\
11 .orderBy("date.Date", "city.StateProvince", "city.City")
12
13 sale_by_date_city.write.mode("overwrite").format("delta").option("overwriteSchema", "true").save("Tables/aggregate_sale_by_date_city")
```

PySpark (Python) ▾

> Spark jobs (12 of 12 succeeded)

...

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

The screenshot shows a user interface for managing data sources. At the top, there's a back arrow labeled 'All sources' and a double-left arrow icon. Below that, the title 'Lakehouses' is displayed, followed by a 'Data sources' button. A specific lakehouse named 'wwilakehouse' is selected, indicated by a checkmark icon. Under the 'Tables' section, two entries are highlighted with red boxes: 'aggregate_sale_by_date_city' and 'aggregate_sale_by_date_employee'. Other listed tables include 'dimension_city', 'dimension_customer', 'dimension_date', 'dimension_employee', 'dimension_stock_item', and 'fact_sale'. There's also a 'Files' section containing 'wwi-raw-data'.

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	ABC_CalendarMo...	ABC_Day	ABC_ShortMonth	CalendarYear	ABC_City	ABC_StateProvince	ABC_SalesTerritory	SumOfTotalExclu...	SumOfTaxAmount	SumOfTotalInclu...
1/1/2000 12:00...	CY2000-Jan	1	Jan	2000	Bazemore	Alabama	Southeast	82944.00	12441.60	95385.60
1/1/2000 12:00...	CY2000-Jan	1	Jan	2000	Belgreen	Alabama	Southeast	601280.00	90192.00	691472.00
1/1/2000 12:00...	CY2000-Jan	1	Jan	2000	Coker	Alabama	Southeast	257984.00	38697.60	296681.60
1/1/2000 12:00...	CY2000-Jan	1	Jan	2000	Eulaton	Alabama	Southeast	507584.00	76137.60	583721.60
1/1/2000 12:00...	CY2000-Jan	1	Jan	2000	Highland Home	Alabama	Southeast	18480.00	2772.00	21252.00
1/1/2000 12:00...	CY2000-Jan	1	Jan	2000	Jemison	Alabama	Southeast	278816.00	41822.72	320638.72
1/1/2000 12:00...	CY2000-Jan	1	Jan	2000	Marion Junction	Alabama	Southeast	212992.00	31948.80	244940.80
1/1/2000 12:00...	CY2000-Jan	1	Jan	2000	Narafalia	Alabama	Southeast	312256.00	46838.40	359094.40
1/1/2000 12:00...	CY2000-Jan	1	Jan	2000	Robertsdale	Alabama	Southeast	799223.40	119883.36	919105.76
1/1/2000 12:00...	CY2000-Jan	1	Jan	2000	Saks	Alabama	Southeast	4699312.00	70396.64	539708.64
1/1/2000 12:00...	CY2000-Jan	1	Jan	2000	Southside	Alabama	Southeast	20160.00	3034.00	23184.00
1/1/2000 12:00...	CY2000-Jan	1	Jan	2000	Tuscaloosa	Alabama	Southeast	36000.00	5400.00	41400.00
1/1/2000 12:00...	CY2000-Jan	1	Jan	2000	Akhiok	Alaska	Far West	601177.60	90176.64	691354.24
1/1/2000 12:00...	CY2000-Jan	1	Jan	2000	Ekwok	Alaska	Far West	106000.00	15900.00	121900.00
1/1/2000 12:00...	CY2000-Jan	1	Jan	2000	Haycock	Alaska	Far West	284544.00	42661.60	327225.60
1/1/2000 12:00...	CY2000-Jan	1	Jan	2000	Itutan	Alaska	Far West	114400.00	17180.32	131560.32

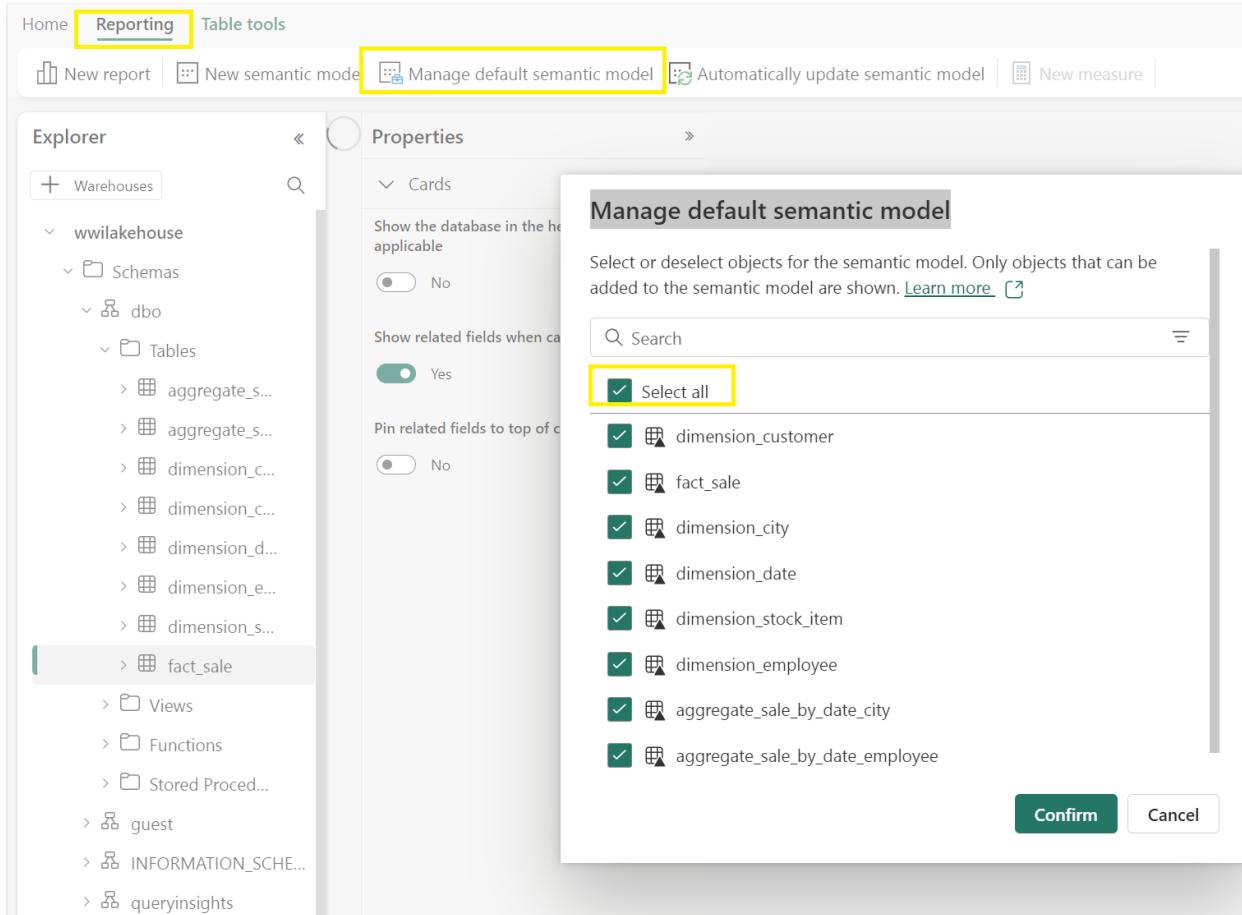
2. Once you are in warehouse mode you should be able to see all the tables you created. If you don't see them yet, please click on the **Refresh** icon at the top. Next, click on **Model** tab at the bottom to open the default Power BI dataset.

The screenshot shows the Power BI Data Engineering interface. The top navigation bar includes Home, Reporting, and a SQL analytics endpoint dropdown. Below the navigation is a message about a lakehouse automatically adding new objects to its semantic model. The left sidebar has sections for Home, Create, Browse, OneLake data hub, Monitoring hub, Workspaces, Fabric Lakehouse..., and wwilakehouse. The wwilakehouse section is expanded, showing Warehouses, Schemas (with dbo selected), Tables (containing aggregate_sa... and other dimension and fact tables), Views, Functions, Stored Procedures, guest, INFORMATION_SCHEMA, queryinsights, sys, Security, Queries, and My queries. The Data preview pane shows a table with 1,000 rows of data from the aggregate_sa table, with columns including Date, ABC_CalendarMonthLabel, ABC_Day, ABC_ShortMonth, ABC_CalendarYear, ABC_City, and ABC_StateProvince. The bottom navigation bar has tabs for Data, Query, and Model, with the Model tab highlighted and a red box around it. A status message at the bottom indicates success (Succeeded) in 7 seconds and 792 ms.

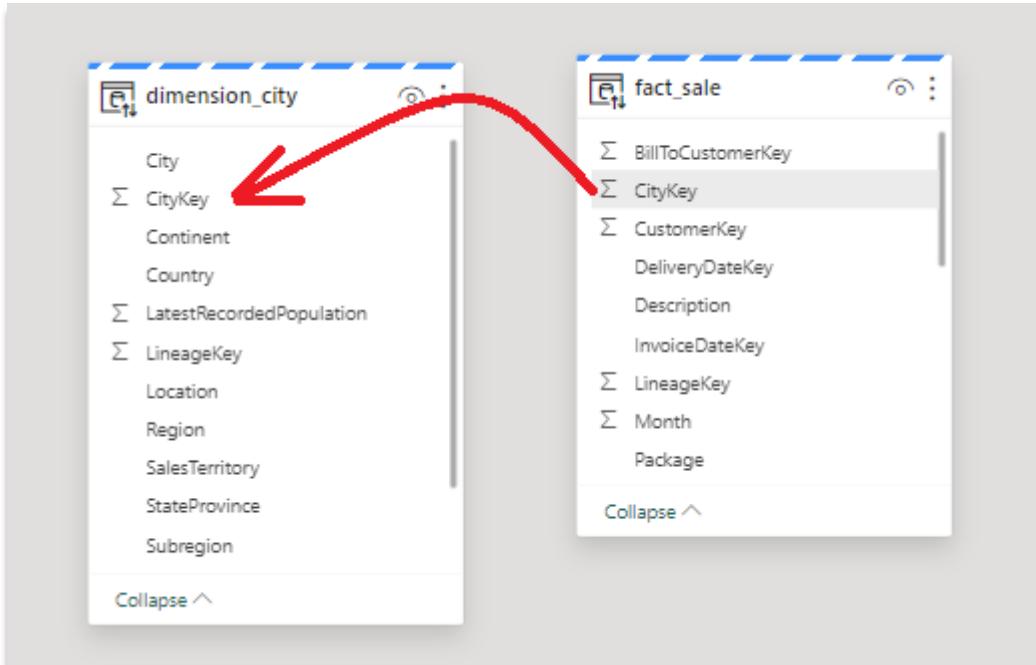
	Date	ABC_CalendarMonthLabel	ABC_Day	ABC_ShortMonth	ABC_CalendarYear	ABC_City	ABC_StateProvince
1	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Bazemore	Alabama
2	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Belgreen	Alabama
3	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Coker	Alabama
4	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	Akhiok	Alaska
14	2000-01-01 00:00:00.0000000	CY2000-Jan	1	Jan	2000	Ekwok	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	Grannis	Arkansas

● Succeeded (7 sec 792 ms) Columns: 12 Rows: 1,000

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.



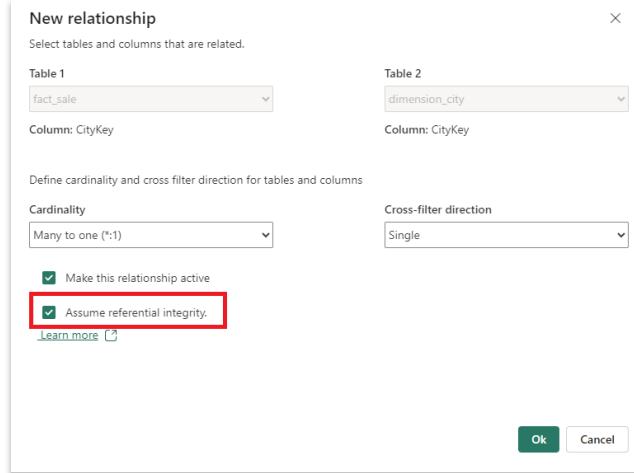
4. For this data model, you need to define the relationship between different tables so that you can create reports and visualizations based on data coming across different tables. From the **fact_sale** table, drag the **CityKey** field and drop it on the **CityKey** field in the **dimension_city**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



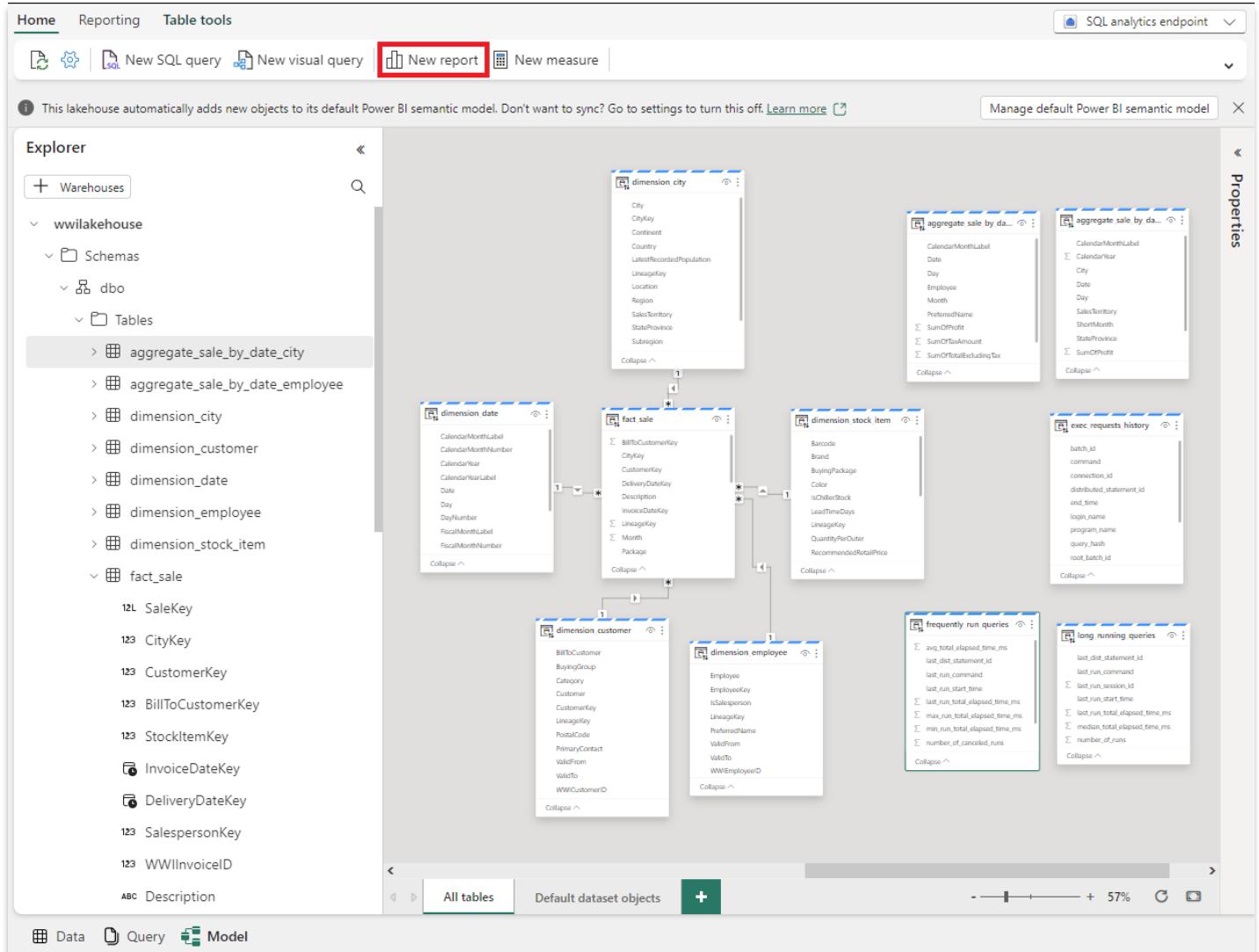
5. On the **New Relationship** settings:

- 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)

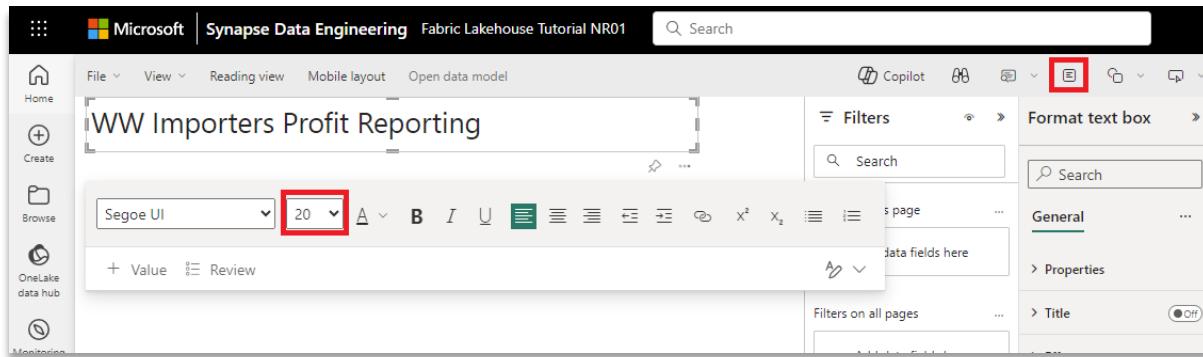


After adding these above relationships, your data model is ready, as above, for reporting. Click on **New report** to start creating reports/dashboards in Power BI.

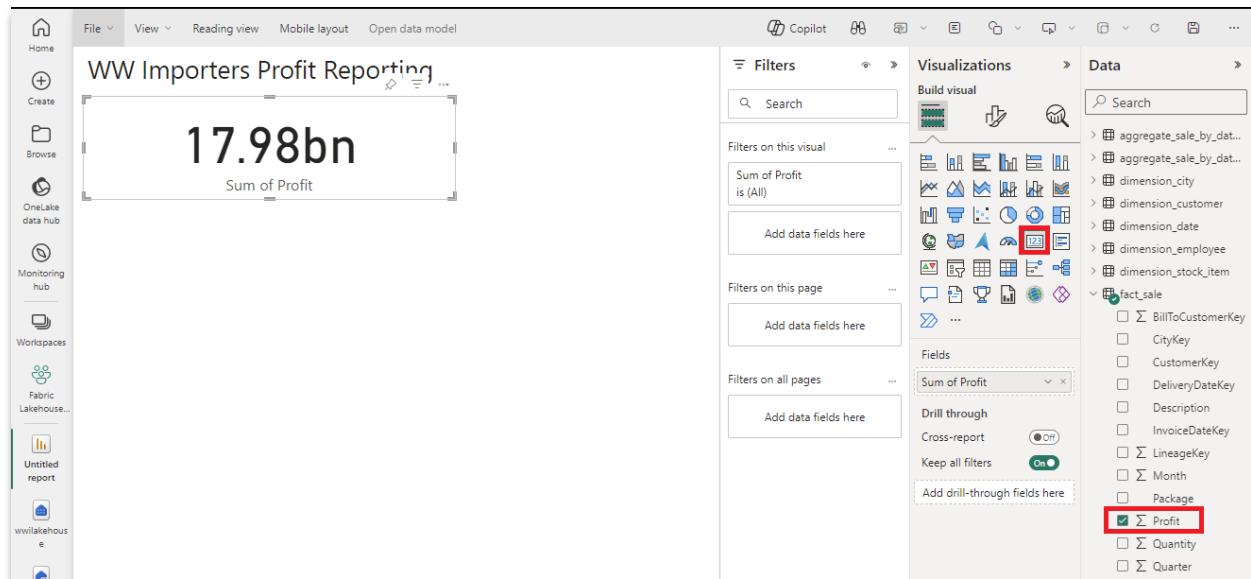
6. On the Power BI report canvas, you can create reports to meet your business requirements by dragging required columns from the **Data** pane to the canvas and using one or more of available visualizations.

The screenshot shows the Microsoft Power BI interface in the 'Data' view. On the left, there's a sidebar with various navigation options like Home, Create, Browse, OneLake data hub, Monitoring hub, Workspaces, Fabric Lakehouse..., Untitled report, and two 'wwlakehouse' items. The main area has a title bar 'Microsoft | Synapse Data Engineering Fabric Lakehouse Tutorial NR01' and a search bar. Below the title bar, there are tabs for File, View, Reading view, Mobile layout, and Open data model. The top right shows a trial status 'Trial: 59 days left' and a user profile icon. The central workspace is titled 'Build visuals with your data' and contains a message 'Select or drag fields from the Data pane onto the report canvas.' A cursor is hovering over a field in a dashed box on the report canvas. To the right is the 'Data' pane, which includes sections for Filters, Visualizations, and Data. The 'Data' section is expanded, showing a large list of fields under 'Values'. A red box highlights this list of fields, which includes: aggregate_sale_by_dat..., aggregate_sale_by_dat..., dimension_city, dimension_customer, dimension_date, dimension_employee, dimension_stock_item, fact_sale, BillToCustomerKey, CityKey, CustomerKey, DeliveryDateKey, Description, InvoiceDateKey, LineageKey, Month, Package, Profit, Quantity, Quarter, SaleKey, SalespersonKey, StockItemKey, TaxAmount, TaxRate, TotalChillerItems, TotalDryItems, TotalExcludingTax, TotalIncludingTax, UnitPrice, WWInvoiceID, and Year.

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

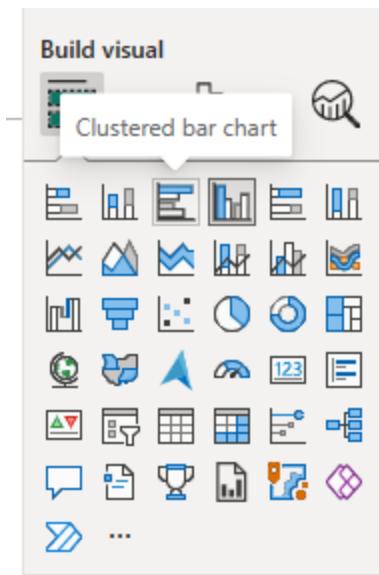


8. Add a Card
- On the **Data** pane, expand **fact_sales** and check the box next to **Profit**. This will create a column chart and add the field to the Y-axis.
 - With the bar chart selected, click the Card visual in the visualization pane. This will convert the visual to a card.
 - Place the card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

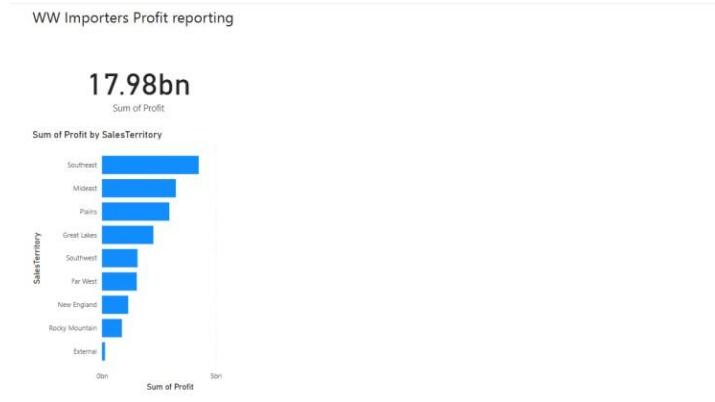


9. Add a Bar chart

- 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.



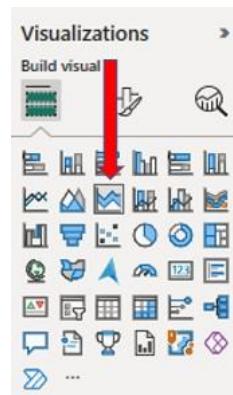
- d. Resize the Bar chart to fill in the area under the title and Card



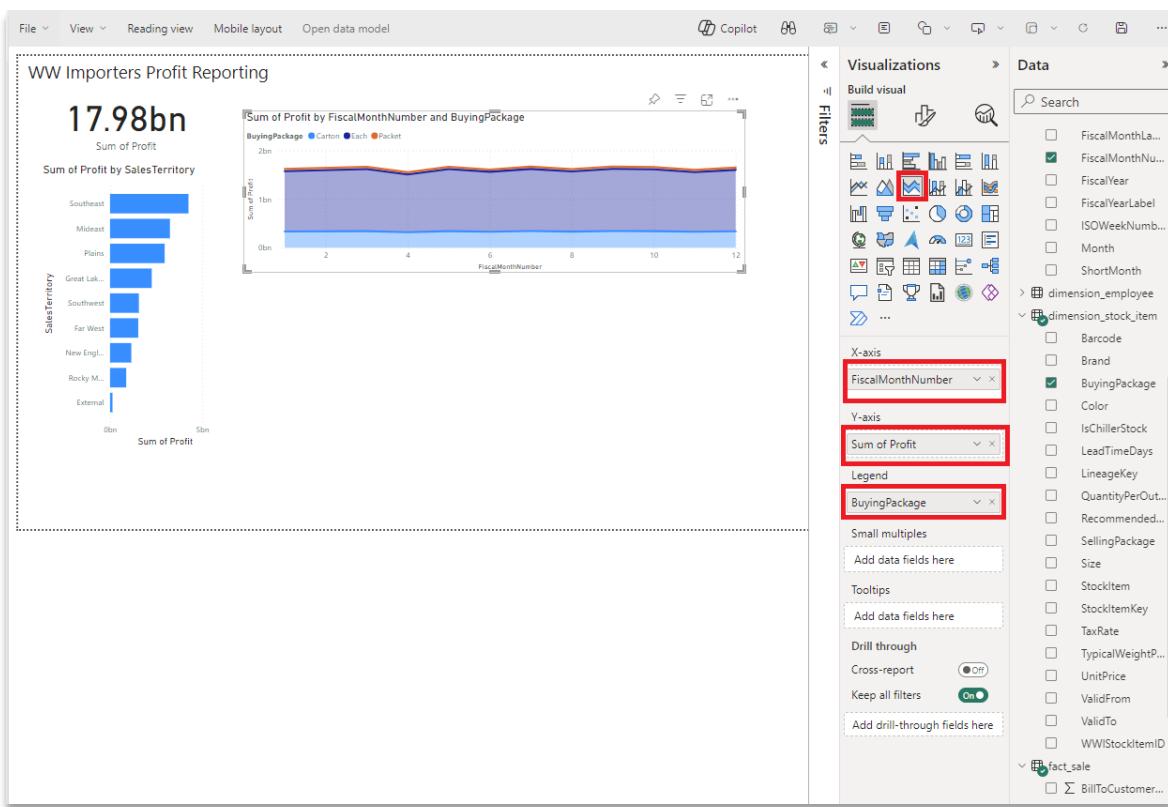
10. Click anywhere on the blank canvas (or press the Esc key) so the bar chart visual is no longer selected.

11. Build a stacked area chart visual:

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



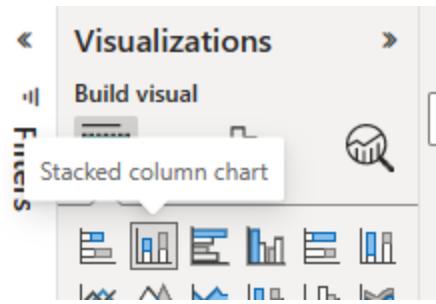
- Reposition and resize the stacked area chart to the right of the card and bar chart visuals created in the previous steps.
- On the **Data** pane, expand **fact_sales** and check the box next to **Profit**. Expand dimension_date and check the box next to **FiscalMonthNumber**. This will create a filled line chart showing profit by fiscal month.
- On the **Data** pane, expand **dimension_stock_item** and drag **BuyingPackage** into the Legend field well. This 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:

- 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.

WW Importers Profit Reporting

Sum of Profit by SalesTerritory

SalesTerritory	Sum of Profit
Southeast	~4.5bn
Midwest	~2.5bn
Plains	~1.5bn
Great Lakes	~1.0bn
Southwest	~0.8bn
Far West	~0.5bn
New England	~0.3bn
Rocky M.	~0.2bn
External	~0.1bn

Sum of Profit by Employee

Employee	Sum of Profit
Sophia Hinton	~2.5bn
Jack Potter	~2.5bn
Kayla Woodcock	~2.0bn
Hudson Omslow	~2.0bn
Amy Treff	~1.8bn
Taj Shand	~1.5bn
Anthony Grosse	~1.5bn
Lily Coide	~1.5bn
Archer Lamble	~1.5bn
Hudson Hollimor	~1.0bn

Visualizations

- Build visual
- Stacked column chart

Data

- Filters
- Build visual
- Stacked column chart

Sum of Profit by Employee

X-axis: Employee

Y-axis: Sum of Profit

Legend

Drill through

Cross-report: Off

Keep all filters: On

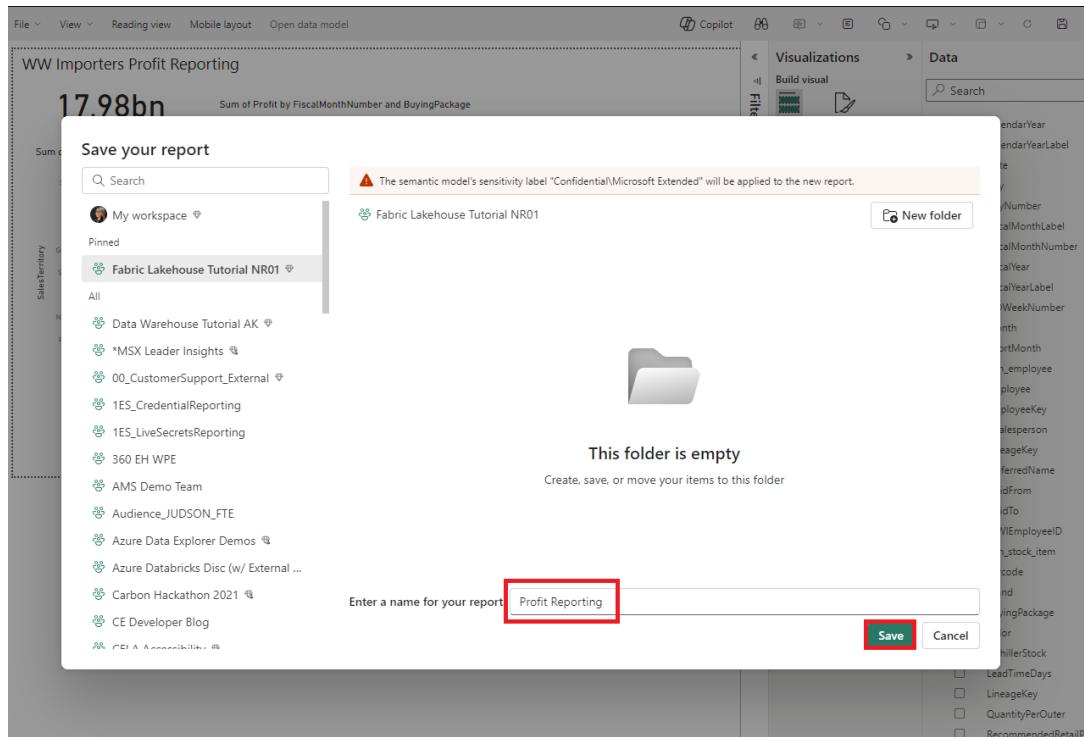
Dimension Fields

- dimension_employee
- Employee
- EmployeeKey
- IsSalesperson
- LineageKey
- PreferredName
- ValidFrom
- ValidTo
- WWIEmployeeID

Dimension Fields

- dimension_stock_item
- Barcode
- Brand
- BuyingPackage
- Color
- IsChillerStock
- LeadTimeDays

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**



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 'Fabric Lakehouse Tutorial NR01' workspace settings page. At the top, there's a header with the workspace name and a 'Manage access' button. Below it is a 'Workspace settings' button, which is highlighted with a red box. A dropdown menu is open over this button, showing options like 'Manage access' and 'Workspace settings'. The main content area lists various workspace items: '01 - Create Delta Tables', '02 - Data Transformation - Business Aggregates', 'IngestDataFromSourceToLakehouse', 'Load Lakehouse Table', 'Profit Reporting', 'report1', 'wwilakehouse', and two entries for 'wwilakehouse'. Each item has columns for Name, Type, Last modified, and Created by.

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

Workspace settings

Search

Delete this workspace and all data and items in it?

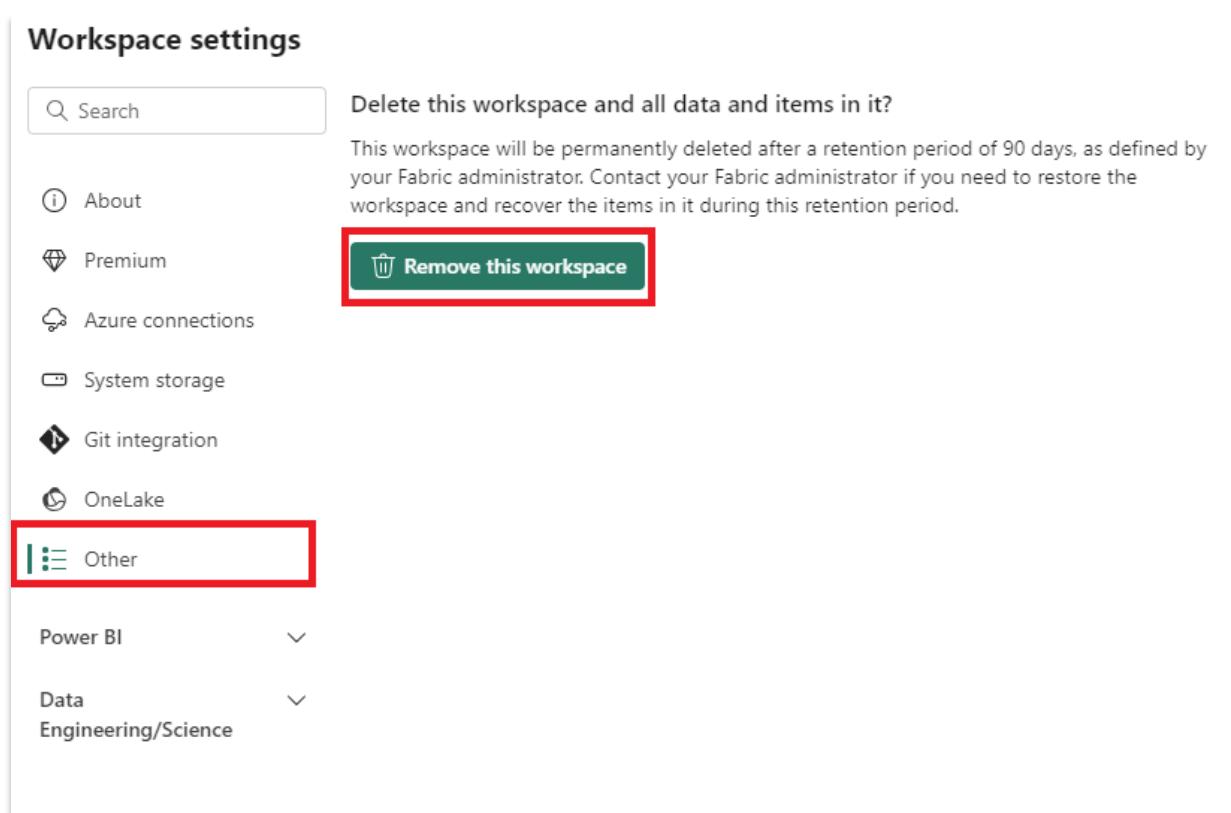
This workspace will be permanently deleted after a retention period of 90 days, as defined by your Fabric administrator. Contact your Fabric administrator if you need to restore the workspace and recover the items in it during this retention period.

Remove this workspace

- About
- Premium
- Azure connections
- System storage
- Git integration
- OneLake
- Other**

Power BI

Data Engineering/Science



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