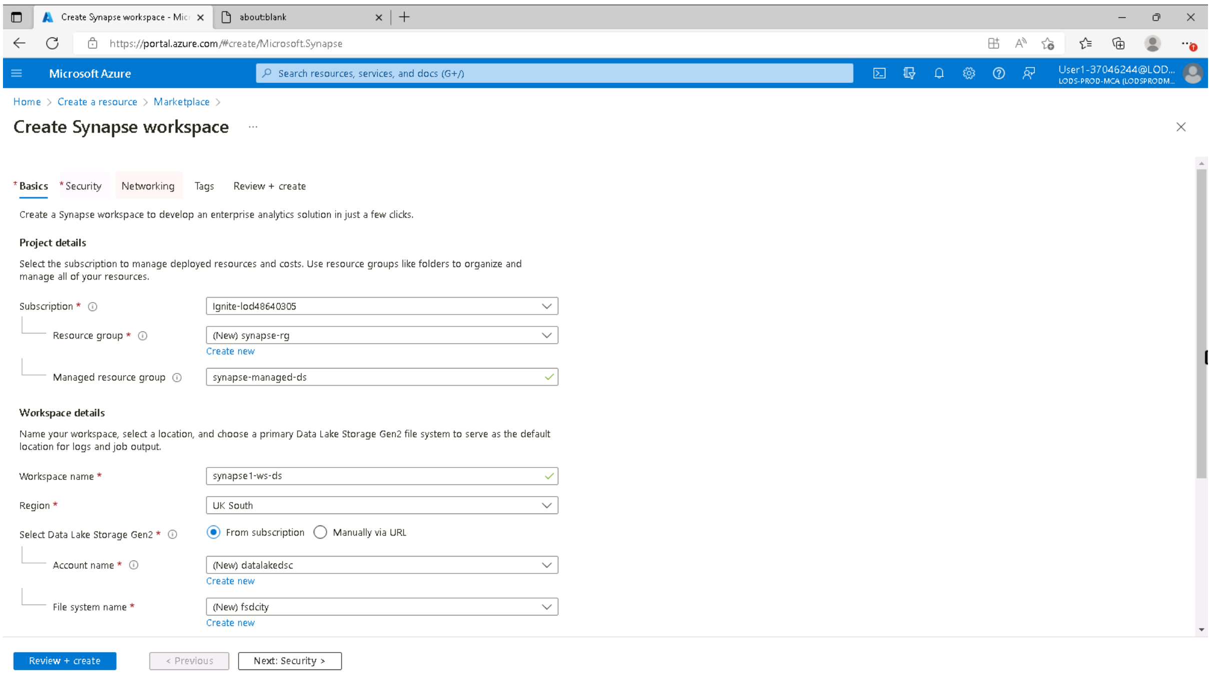
Explore Data Analytics in Azure

Create a Synapse workspace



Create Synapse workspace - Mic
about:blank
https://portal.azure.com/#create/Microsoft.Synapse
Microsoft Azure
Search resources, services, and docs (G+/)
User1-37046244@LOD...
LODS-PROD-MCA (LODSPRODM...
Home > Create a resource > Marketplace >

Create Synapse workspace

*Basics *Security Networking Tags Review + create

Create a Synapse workspace to develop an enterprise analytics solution in just a few clicks.

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all of your resources.

Subscription * : Ignite-lod48640305

Resource group * : (New) synapse-rg
Create new

Managed resource group : synapse-managed-ds

Workspace details

Name your workspace, select a location, and choose a primary Data Lake Storage Gen2 file system to serve as the default location for logs and job output.

Workspace name * : synapse1-ws-ds

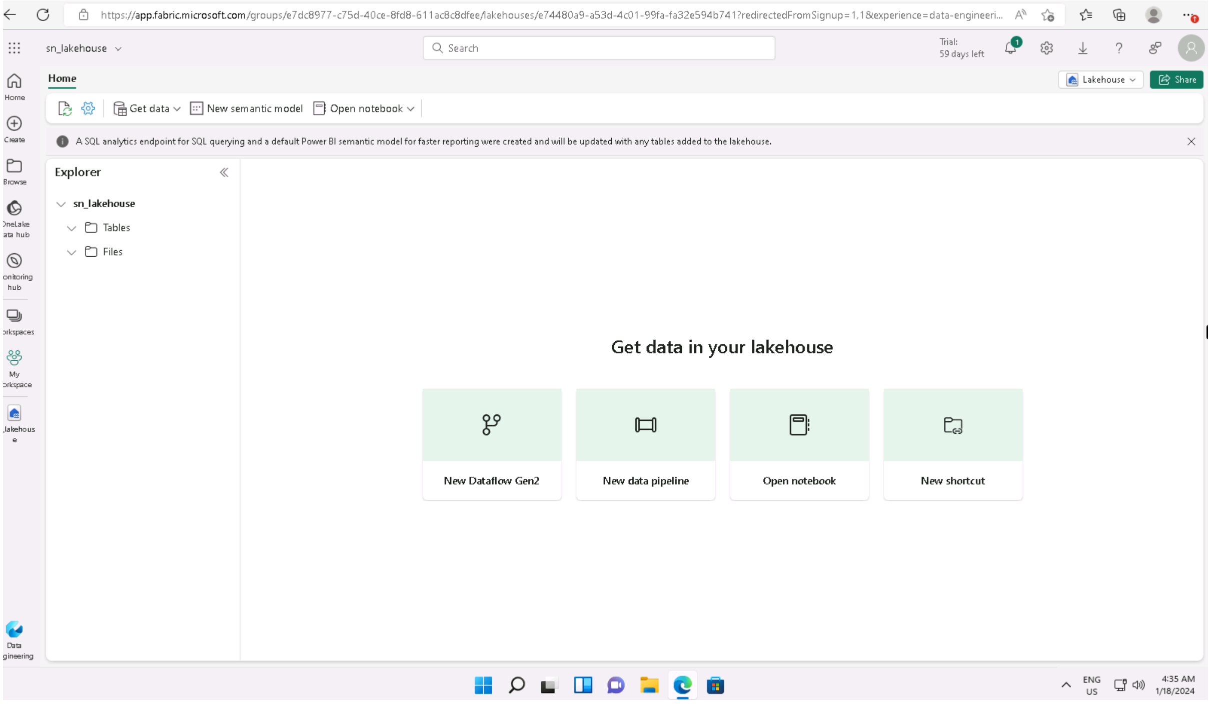
Region * : UK South

Select Data Lake Storage Gen2 * : ● From subscription ○ Manually via URL

Account name * : (New) datalakedsc
Create new

File system name * : (New) fsdcity
Create new

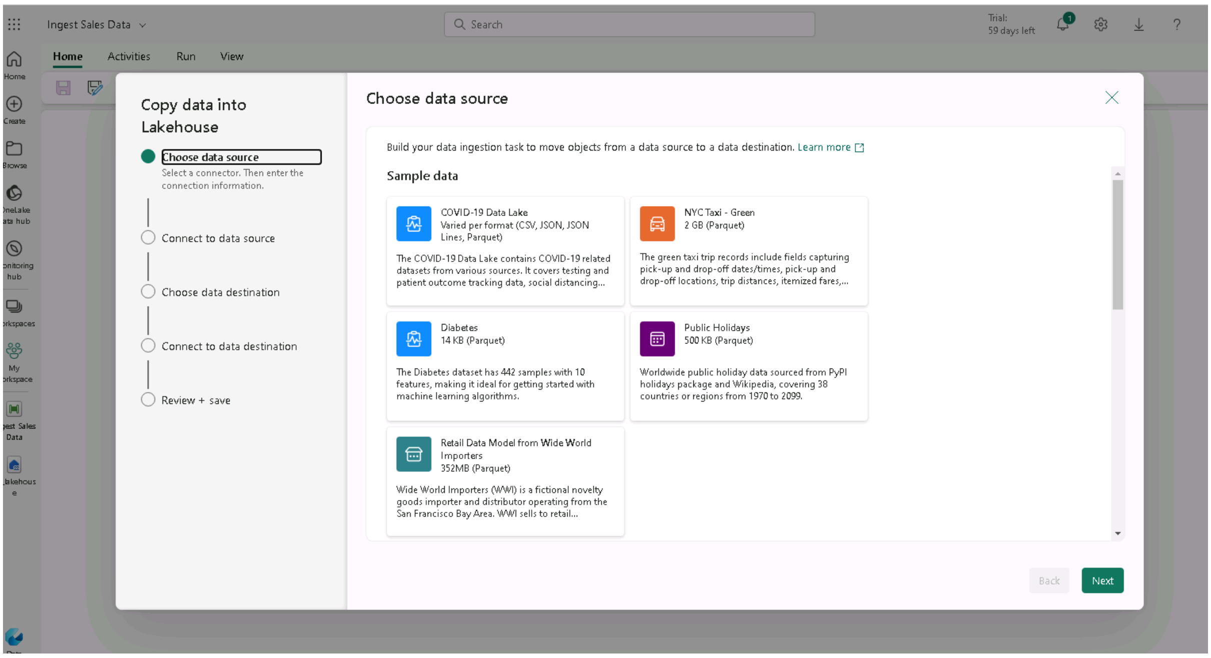
Create a Lakehouse



A SQL analytics endpoint for SQL querying and a default Power BI semantic model for faster reporting were created and will be updated with any tables added to the lakehouse.

Ingest Data

- A simple way to ingest data is to use a **Copy Data** activity in a pipeline to extract the data from a source and copy it to a file in the lakehouse.



Copy data into Lakehouse

- Choose data source
  Select a connector. Then enter the connection information.
- Connect to data source
- Choose data destination
- Connect to data destination
- Review + save

Choose data source

Build your data ingestion task to move objects from a data source to a data destination. Learn more

Sample data

COVID-19 Data Lake
Varied per format (CSV, JSON, JSON Lines, Parquet)
The COVID-19 Data Lake contains COVID-19 related datasets from various sources. It covers testing and patient outcome tracking data, social distancing...

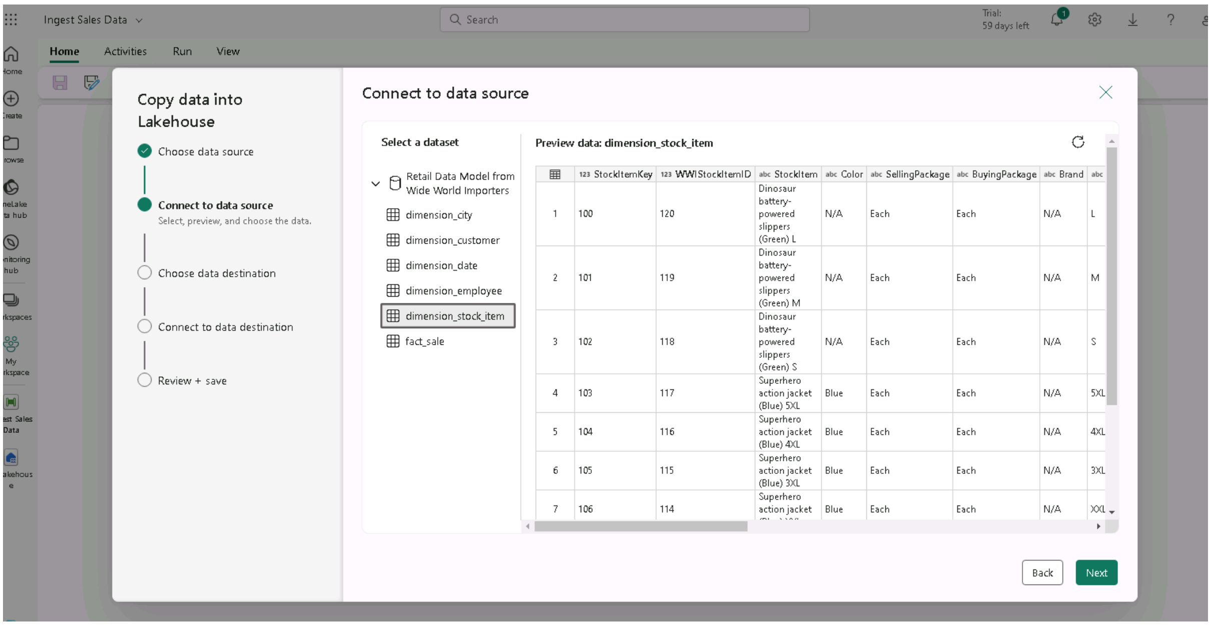
NYC Taxi - Green
2 GB (Parquet)
The green taxi trip records include fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares,...

Diabetes
14 KB (Parquet)
The Diabetes dataset has 442 samples with 10 features, making it ideal for getting started with machine learning algorithms.

Public Holidays
500 KB (Parquet)
Worldwide public holiday data sourced from PyPI holidays package and Wikipedia, covering 38 countries or regions from 1970 to 2099.

Retail Data Model from Wide World Importers
352MB (Parquet)
Wide World Importers (WWI) is a fictional novelty goods importer and distributor operating from the San Francisco Bay Area. WWI sells to retail...

1. Connecting to a data source



Copy data into Lakehouse

- Choose data source
- Connect to data source
  Select, preview, and choose the data.
- Choose data destination
- Connect to data destination
- Review + save

Connect to data source

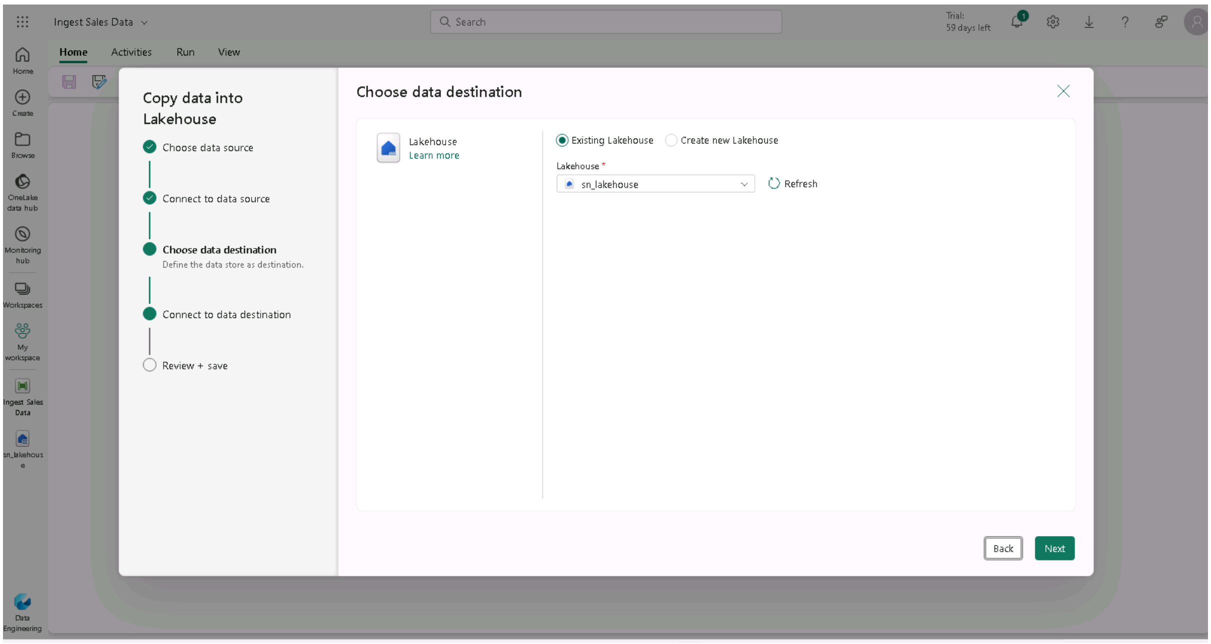
Select a dataset

- Retail Data Model from Wide World Importers
  - dimension_city
  - dimension_customer
  - dimension_date
  - dimension_employee
  - dimension_stock_item
  - fact_sale

Preview data: dimension_stock_item

	123 StockItemKey	123 WWIStockItemID	abc StockItem	abc Color	abc SellingPackage	abc BuyingPackage	abc Brand	abc
1	100	120	Dinosaur battery-powered slippers (Green) L	N/A	Each	Each	N/A	L
2	101	119	Dinosaur battery-powered slippers (Green) M	N/A	Each	Each	N/A	M
3	102	118	Dinosaur battery-powered slippers (Green) S	N/A	Each	Each	N/A	S
4	103	117	Superhero action jacket (Blue) 5XL	Blue	Each	Each	N/A	5XL
5	104	116	Superhero action jacket (Blue) 4XL	Blue	Each	Each	N/A	4XL
6	105	115	Superhero action jacket (Blue) 3XL	Blue	Each	Each	N/A	3XL
7	106	114	Superhero action jacket	Blue	Each	Each	N/A	XXL

2. Choosing an existing data destination



Copy data into Lakehouse

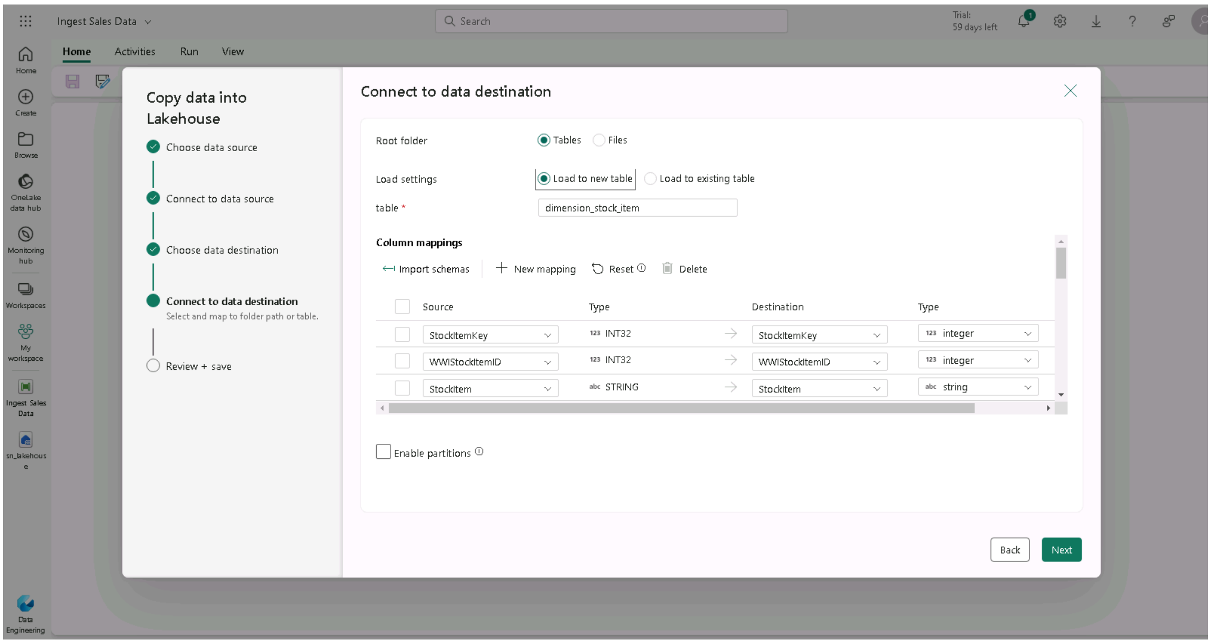
- Choose data source
- Connect to data source
- **Choose data destination**
  Define the data store as destination.
- Connect to data destination
- Review + save

Choose data destination

Lakehouse
Learn more

● Existing Lakehouse ○ Create new Lakehouse

Lakehouse *
sn_lakehouse



Copy data into Lakehouse

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

Connect to data destination

Root folder ● Tables ○ Files

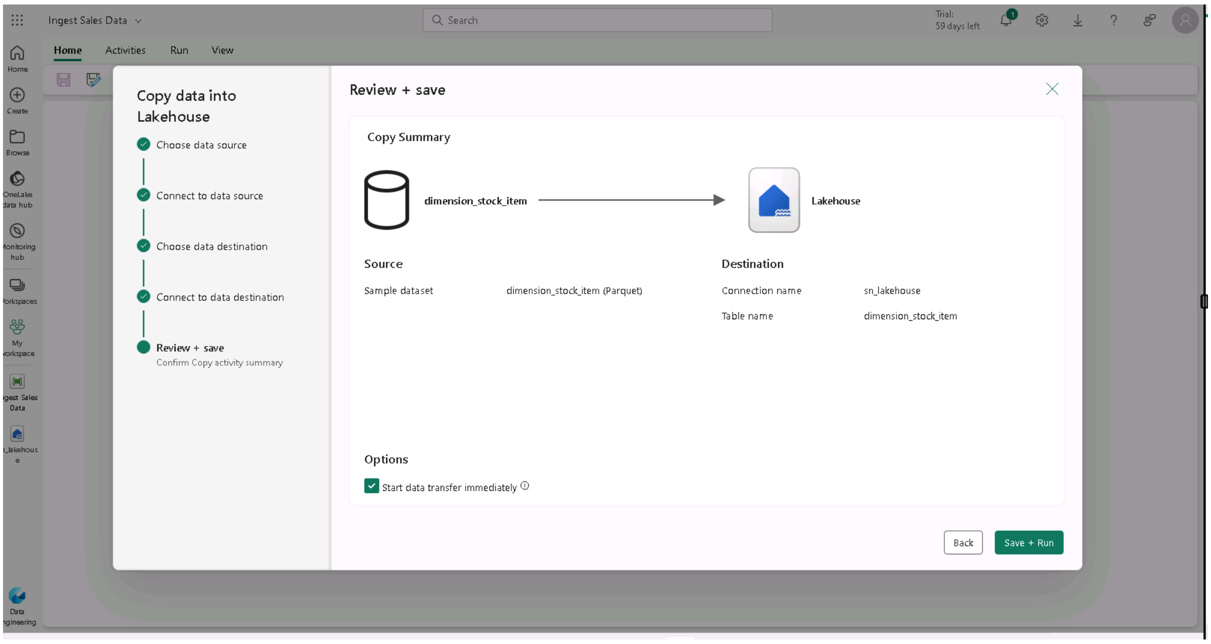
Load settings ● Load to new table ○ Load to existing table

table * dimension_stock_item

Column mappings

□	Source	Type		Destination	Type
□	StockItemKey	INT32	→	StockItemKey	integer
□	WWIStockItemID	INT32	→	WWIStockItemID	integer
□	StockItem	STRING	→	StockItem	string

□ Enable partitions



Copy data into Lakehouse

- Choose data source
- Connect to data source
- Choose data destination
- Connect to data destination
- Review + save
  Confirm Copy activity summary

Review + save

Copy Summary

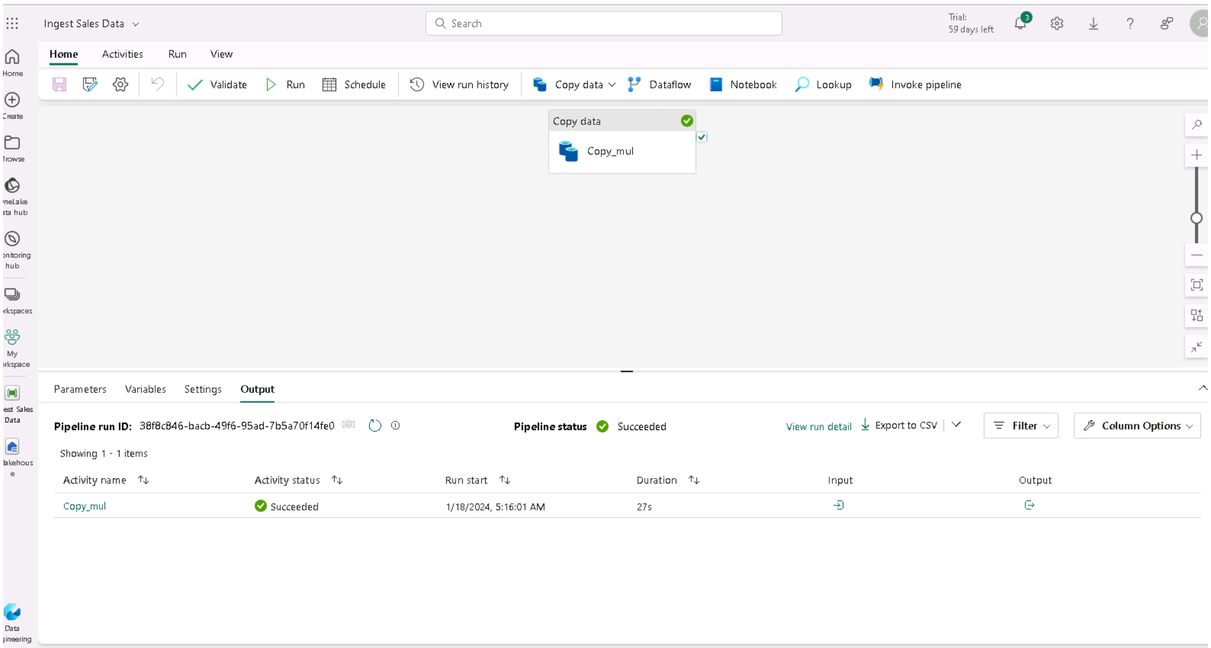
dimension_stock_item → Lakehouse

Source		Destination	
Sample dataset	dimension_stock_item (Parquet)	Connection name	sn_lakehouse
		Table name	dimension_stock_item

Options

☑ Start data transfer immediately

A new pipeline containing a **Copy Data** activity is created, as shown here



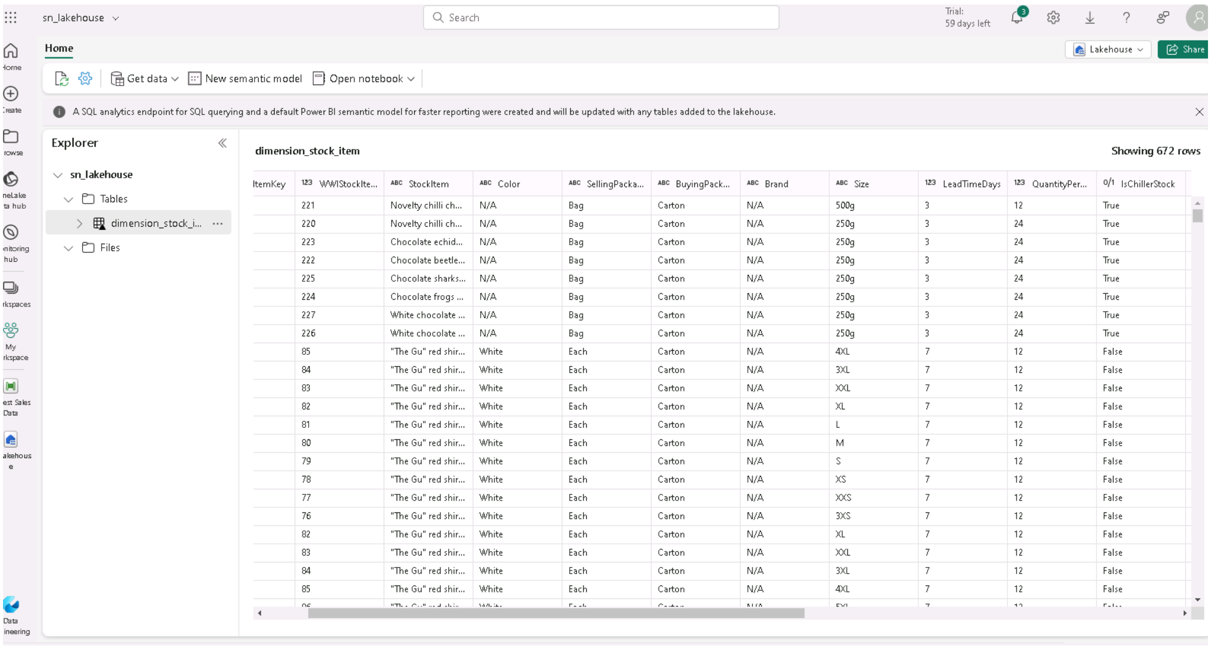
Pipeline run ID: 38f8c846-bacb-49f6-95ad-7b5a70f14fe0

Pipeline status Succeeded

Showing 1 - 1 items

Activity name	Activity status	Run start	Duration	Input	Output
Copy_mul	Succeeded	1/18/2024, 5:16:01 AM	27s		

Results:



A SQL analytics endpoint for SQL querying and a default Power BI semantic model for faster reporting were created and will be updated with any tables added to the lakehouse.

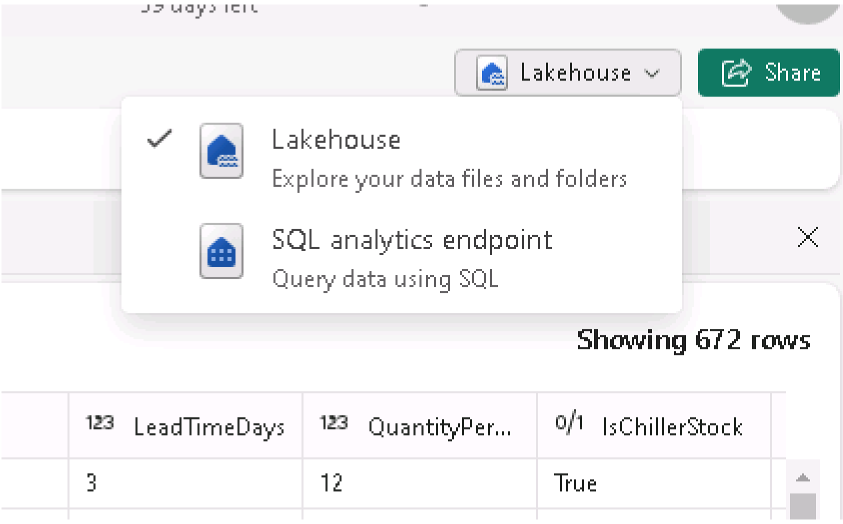
dimension_stock_item

Showing 672 rows

ItemKey	123 WWIStockIte...	ABC StockItem	ABC Color	ABC SellingPacka...	ABC BuyingPack...	ABC Brand	ABC Size	123 LeadTimeDays	123 QuantityPer...	0/1 IsChillerStock
	221	Novelty chilli ch...	N/A	Bag	Carton	N/A	500g	3	12	True
	220	Novelty chilli ch...	N/A	Bag	Carton	N/A	250g	3	24	True
	223	Chocolate echid...	N/A	Bag	Carton	N/A	250g	3	24	True
	222	Chocolate beetle...	N/A	Bag	Carton	N/A	250g	3	24	True
	225	Chocolate sharks...	N/A	Bag	Carton	N/A	250g	3	24	True
	224	Chocolate frogs ...	N/A	Bag	Carton	N/A	250g	3	24	True
	227	White chocolate ...	N/A	Bag	Carton	N/A	250g	3	24	True
	226	White chocolate ...	N/A	Bag	Carton	N/A	250g	3	24	True
	85	"The Gu" red shir...	White	Each	Carton	N/A	4XL	7	12	False
	84	"The Gu" red shir...	White	Each	Carton	N/A	3XL	7	12	False
	83	"The Gu" red shir...	White	Each	Carton	N/A	XXL	7	12	False
	82	"The Gu" red shir...	White	Each	Carton	N/A	XL	7	12	False
	81	"The Gu" red shir...	White	Each	Carton	N/A	L	7	12	False
	80	"The Gu" red shir...	White	Each	Carton	N/A	M	7	12	False
	79	"The Gu" red shir...	White	Each	Carton	N/A	S	7	12	False
	78	"The Gu" red shir...	White	Each	Carton	N/A	XS	7	12	False
	77	"The Gu" red shir...	White	Each	Carton	N/A	XXS	7	12	False
	76	"The Gu" red shir...	White	Each	Carton	N/A	3XS	7	12	False
	82	"The Gu" red shir...	White	Each	Carton	N/A	XL	7	12	False
	83	"The Gu" red shir...	White	Each	Carton	N/A	XXL	7	12	False
	84	"The Gu" red shir...	White	Each	Carton	N/A	3XL	7	12	False
	85	"The Gu" red shir...	White	Each	Carton	N/A	4XL	7	12	False

Query data in a lakehouse:

1. Select SQL analytical endpoint

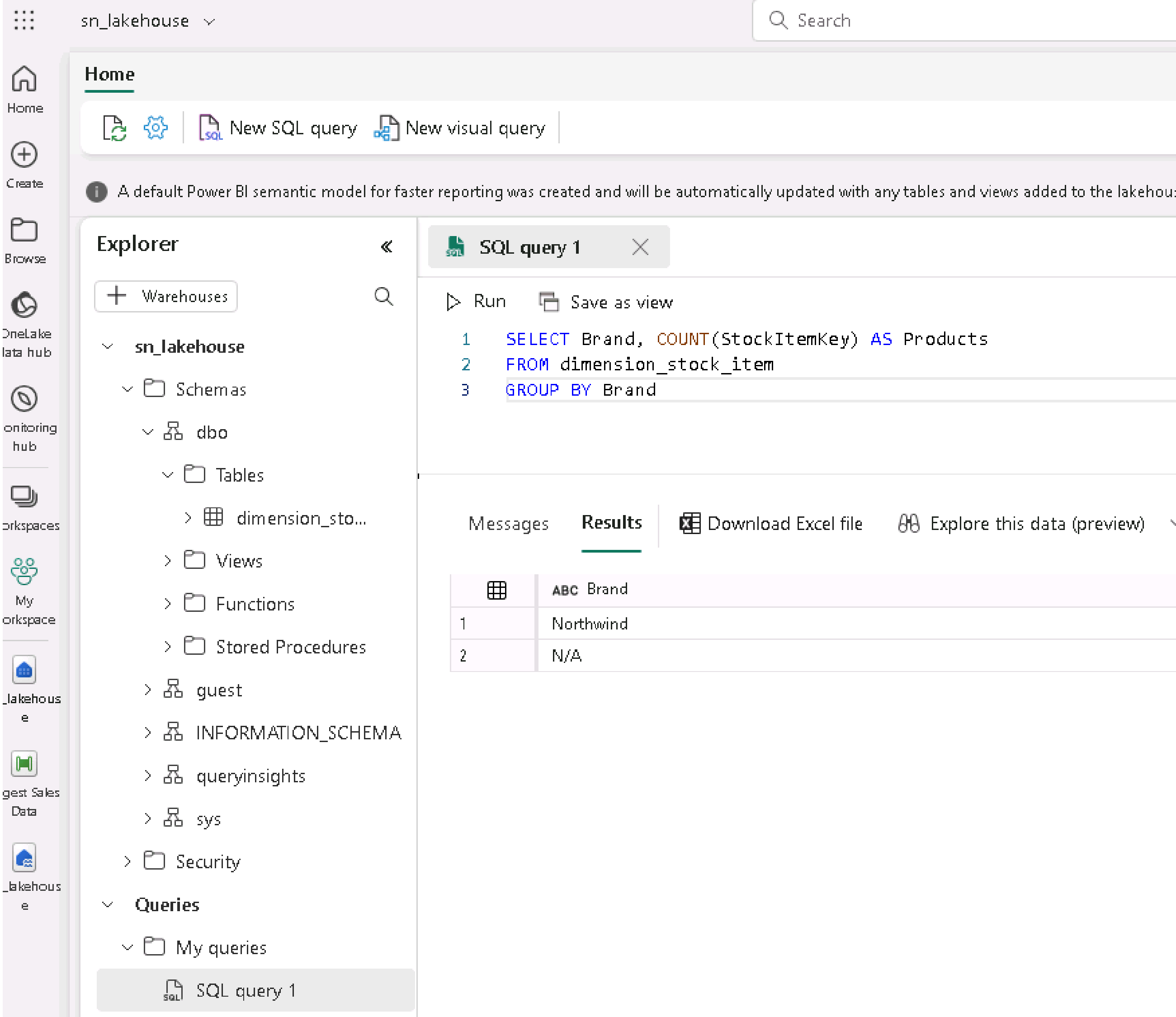


Showing 672 rows

123 LeadTimeDays	123 QuantityPer...	0/1 IsChillerStock
3	12	True

2. Select new sql query and entered the sql code into the query editor. Ran the query and it revealed that there are two brand values - N/A and Northwide

and shows the number of products in each



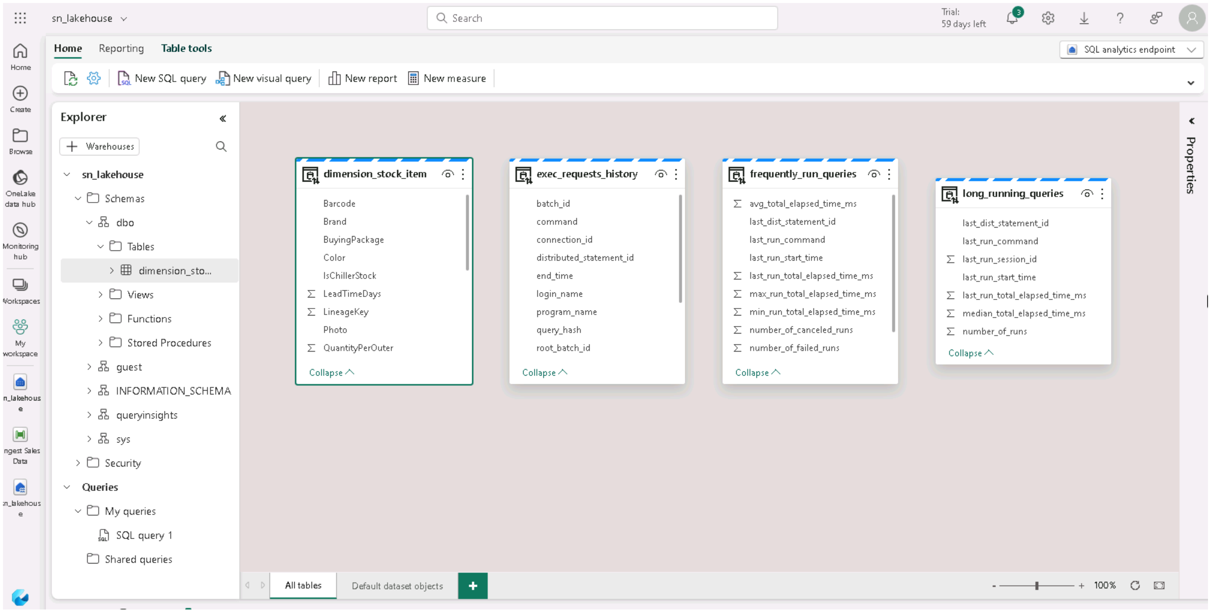
A default Power BI semantic model for faster reporting was created and will be automatically updated with any tables and views added to the lakehou

```
SELECT Brand, COUNT(StockItemKey) AS Products
FROM dimension_stock_item
GROUP BY Brand
```

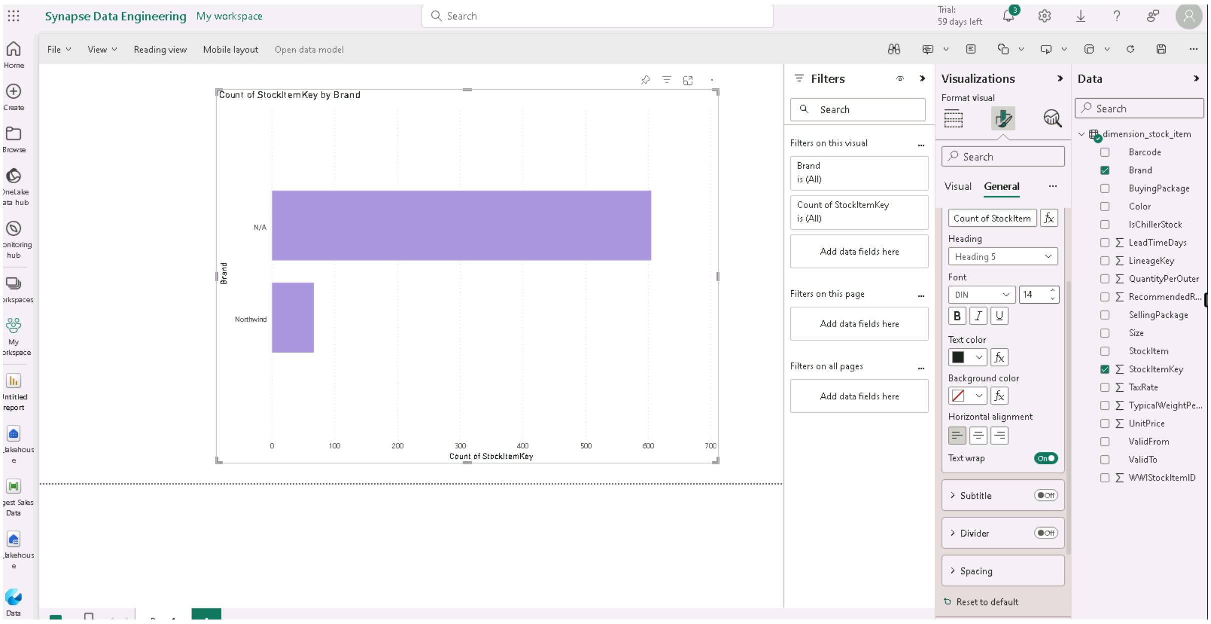
	Brand
1	Northwind
2	N/A

Visualise you data in a lakehouse:

1. Select model tab. This will let you see the data for the tables in the lakehouse.

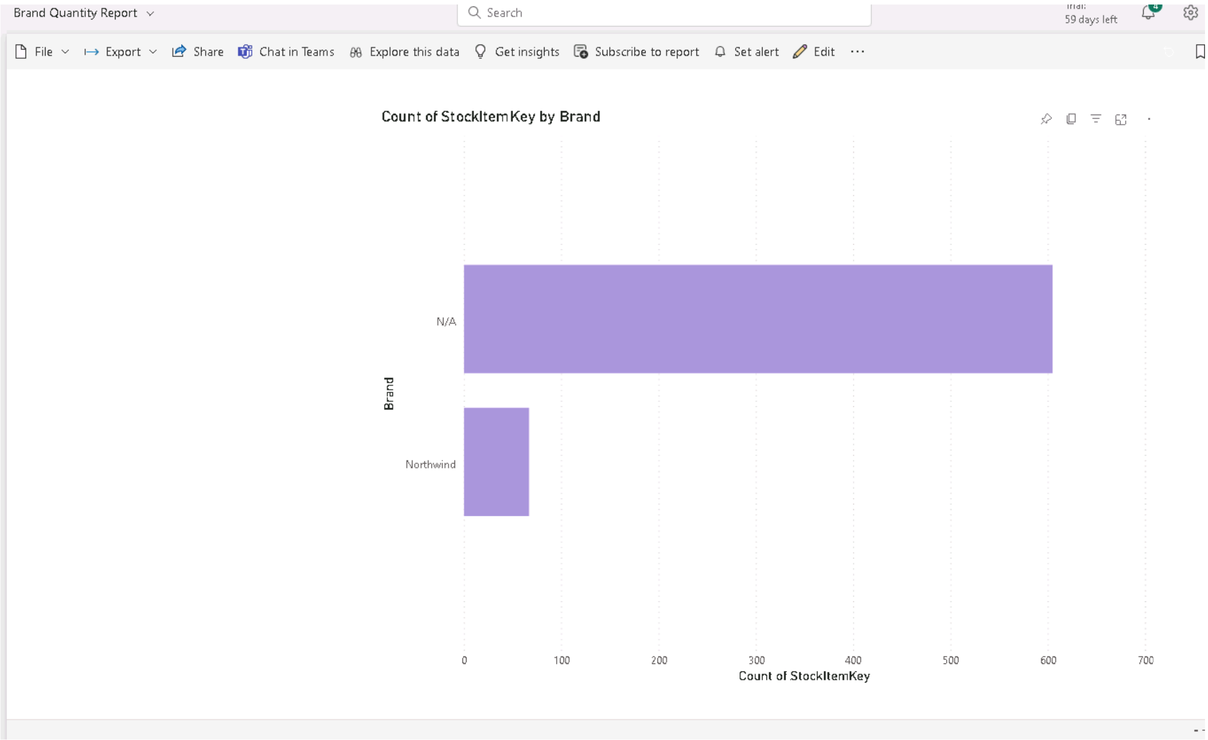


2. Selected stacked barchart and input brand and stockitemkey fields. Then the aggregation in the y-axis and x-axis was changed to count. Next, added colours and resized the chart to make it more visualsing.



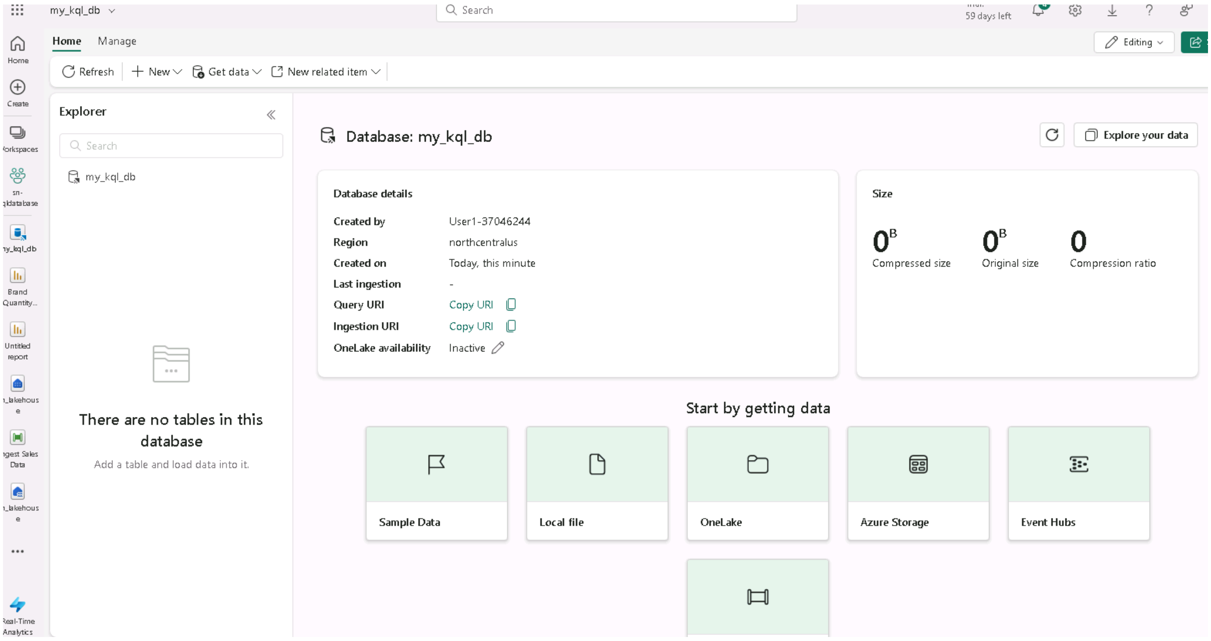
3.

4. Saved it as brand quantity report



Create a KQL Database

1. Switched to Real Time Analytics. The real time analytics home page includes tiles to create commonly used assets for real-time data analyis. Named the database ive created ‛ my_kql_database’.



There are no tables in this database

Add a table and load data into it.

Database details

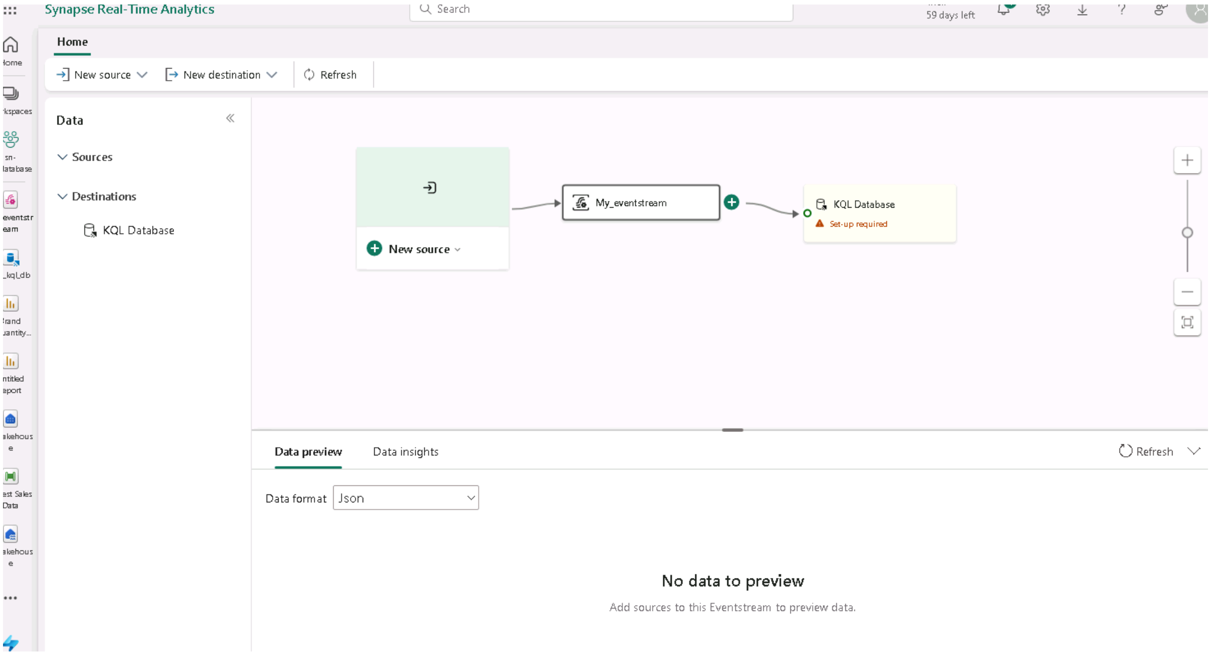
Created by	User1-37046244
Region	northcentralus
Created on	Today, this minute
Last ingestion	-
Query URI	Copy URI
Ingestion URI	Copy URI
OneLake availability	Inactive

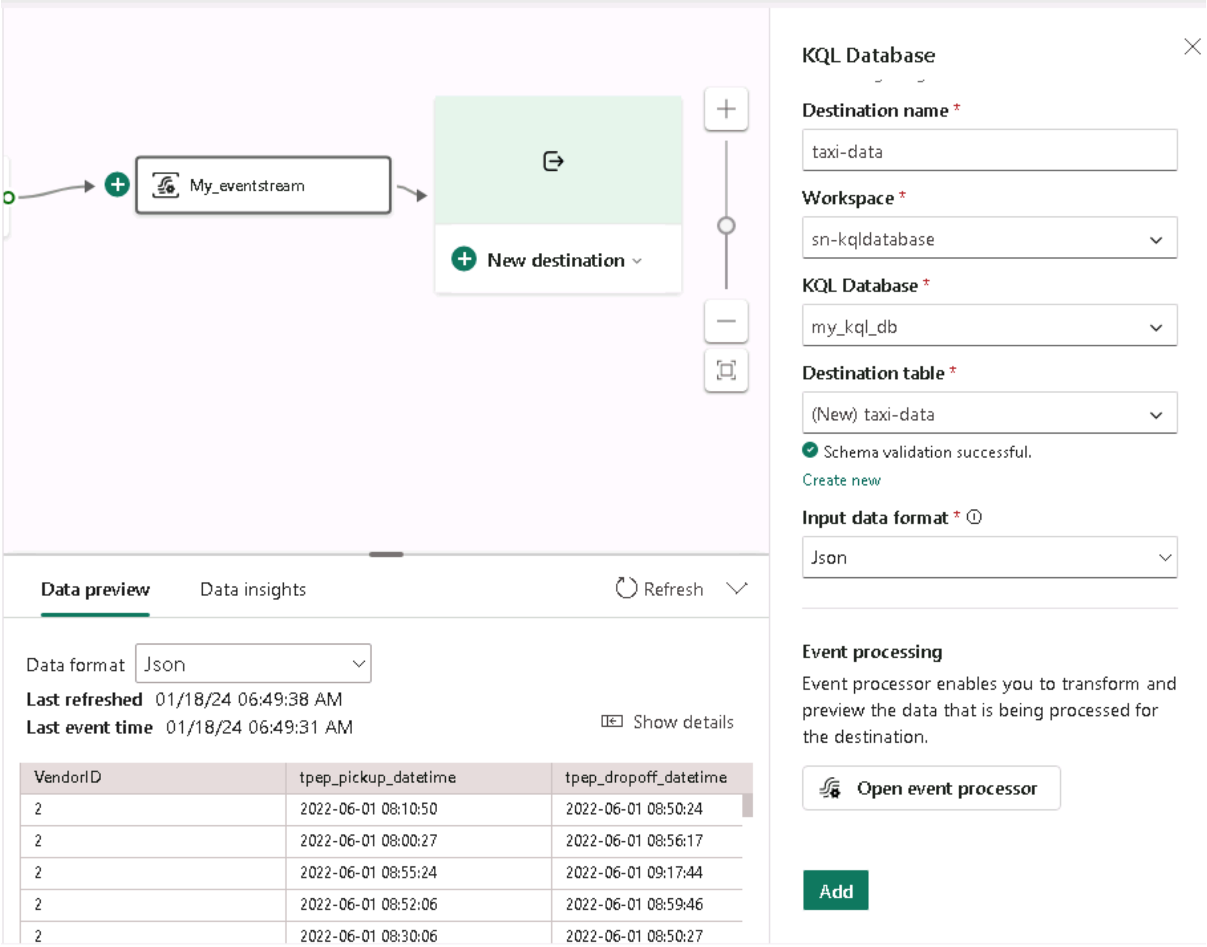
Size

0B	0B	0
Compressed size	Original size	Compression ratio

Create an eventstream

1. Eventstreams provide a scalable and flexible way to ingest real-time data from a streaming source.





KQL Database

Destination name * taxi-data

Workspace * sn-kqldatabase

KQL Database * my_kql_db

Destination table * (New) taxi-data

Schema validation successful.

Create new

Input data format * Json

Event processing

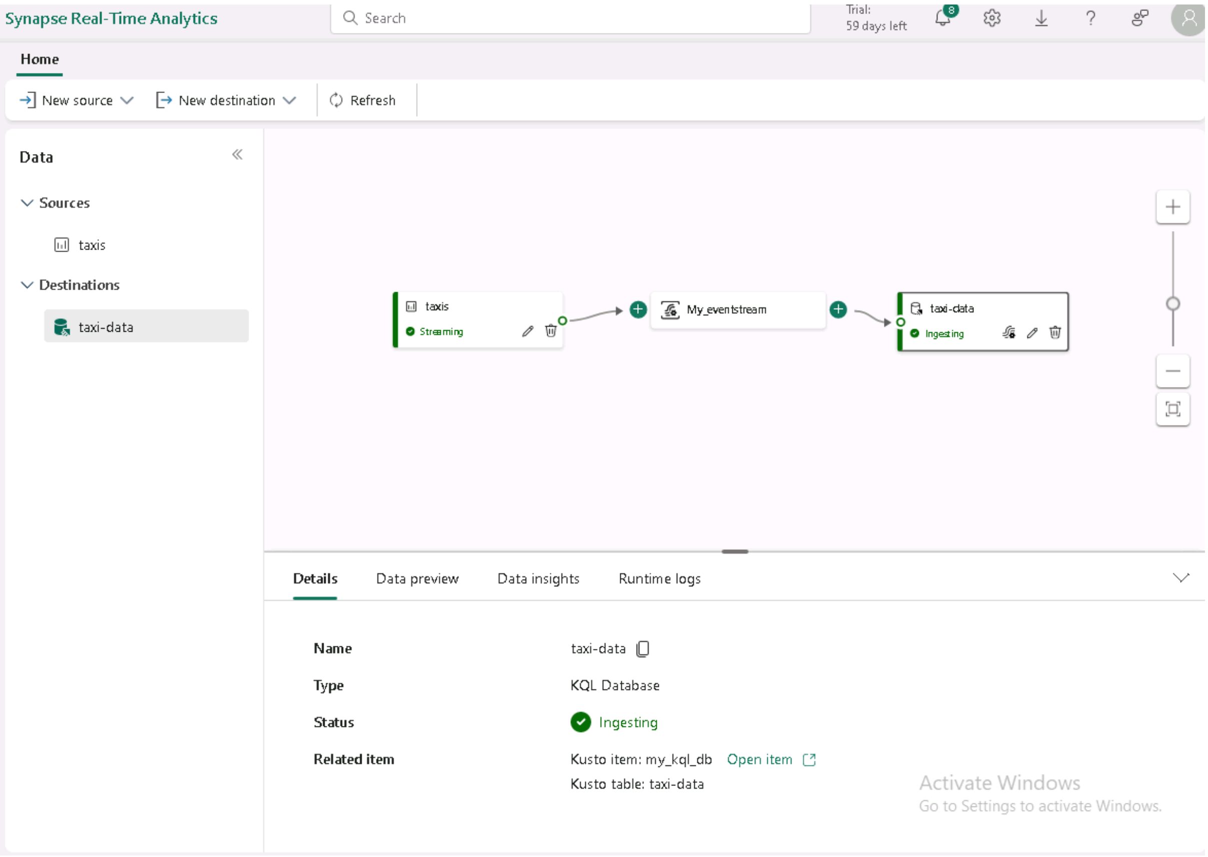
Event processor enables you to transform and preview the data that is being processed for the destination.

Data format Json

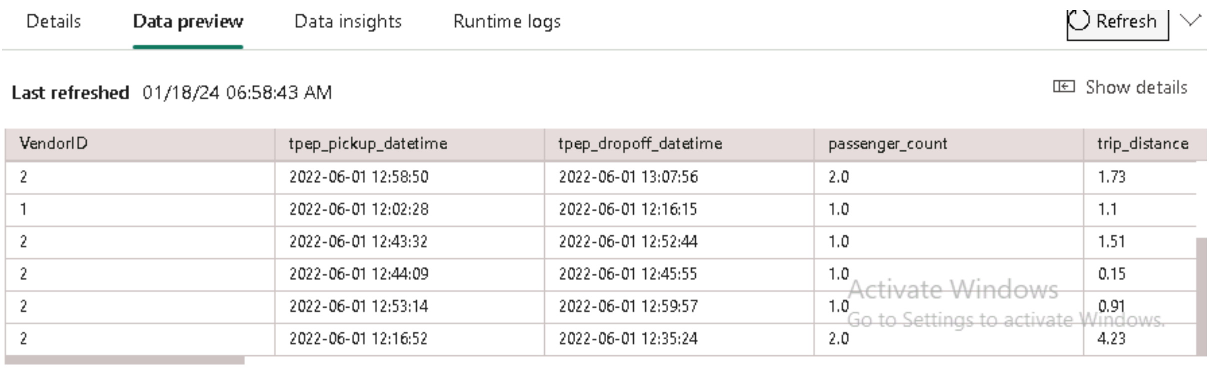
Last refreshed 01/18/24 06:49:38 AM

Last event time 01/18/24 06:49:31 AM

VendorID	tpep_pickup_datetime	tpep_dropoff_datetime
2	2022-06-01 08:10:50	2022-06-01 08:50:24
2	2022-06-01 08:00:27	2022-06-01 08:56:17
2	2022-06-01 08:55:24	2022-06-01 09:17:44
2	2022-06-01 08:52:06	2022-06-01 08:59:46
2	2022-06-01 08:30:06	2022-06-01 08:50:27



Name	taxi-data
Type	KQL Database
Status	Ingesting
Related item	Kusto item: my_kql_db Open item Kusto table: taxi-data

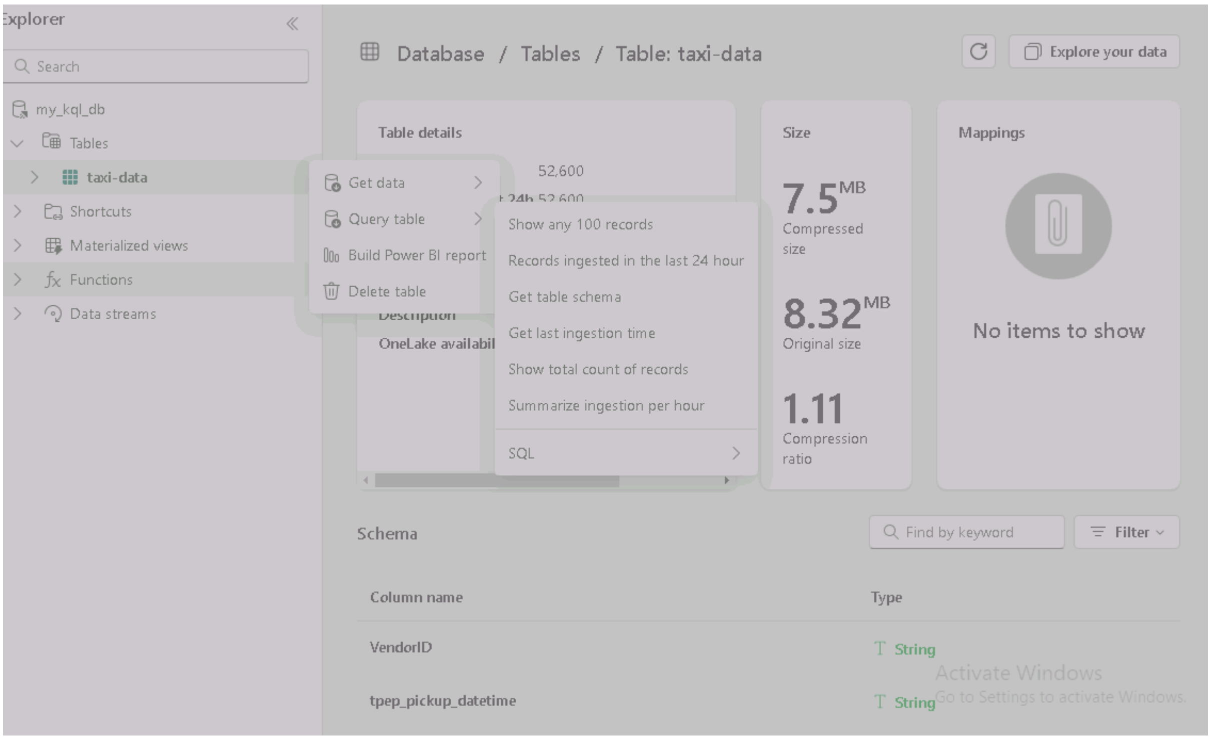


Last refreshed 01/18/24 06:58:43 AM

VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance
2	2022-06-01 12:58:50	2022-06-01 13:07:56	2.0	1.73
1	2022-06-01 12:02:28	2022-06-01 12:16:15	1.0	1.1
2	2022-06-01 12:43:32	2022-06-01 12:52:44	1.0	1.51
2	2022-06-01 12:44:09	2022-06-01 12:45:55	1.0	0.15
2	2022-06-01 12:53:14	2022-06-01 12:59:57	1.0	0.91
2	2022-06-01 12:16:52	2022-06-01 12:35:24	2.0	4.23

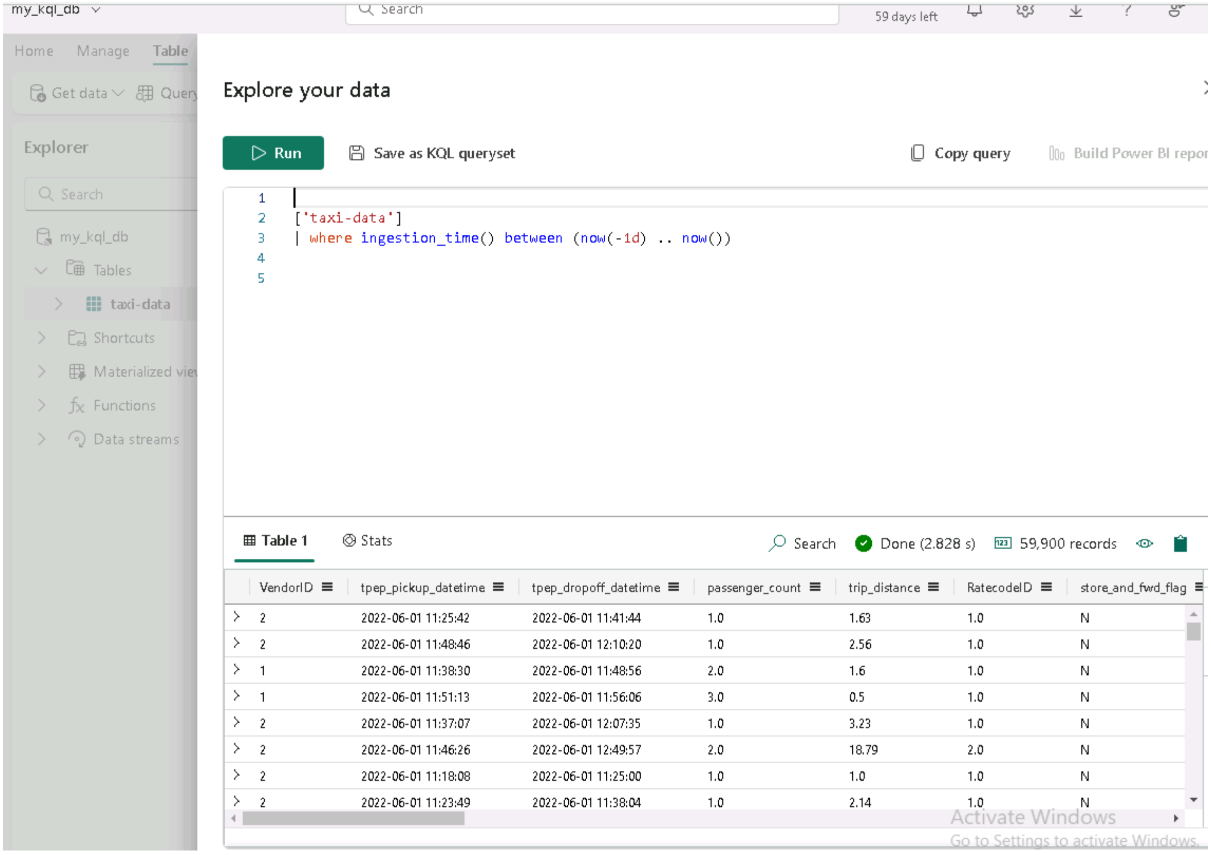
Query real time data in in a KQL data base:

1. select **Query table > Records ingested in the last 24 hours**.



Column name	Type
VendorID	String
tpep_pickup_datetime	String

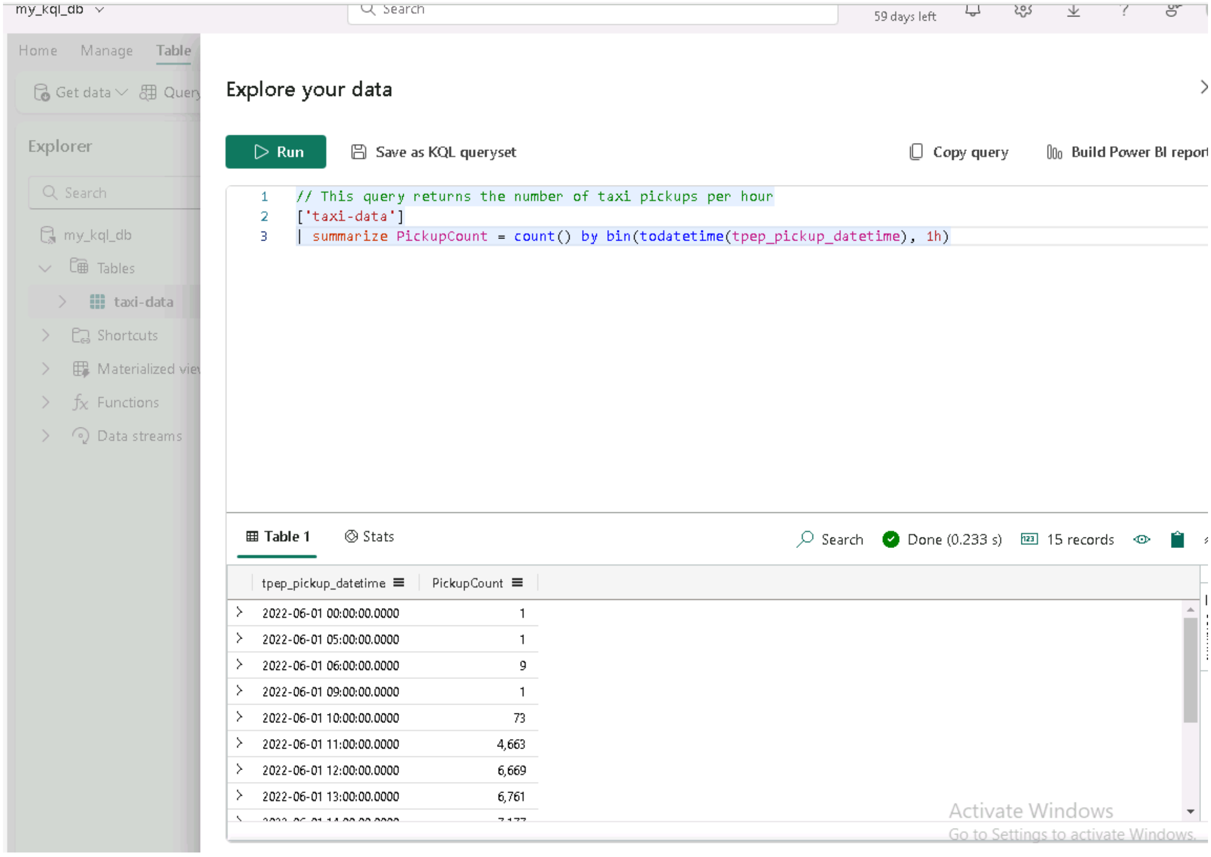
2. This KQL query shows all taxi records ingested from the streaming source in the last 24 hours.



```
['taxi-data']
| where ingestion_time() between (now(-1d) .. now())
```

VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag
2	2022-06-01 11:25:42	2022-06-01 11:41:44	1.0	1.63	1.0	N
2	2022-06-01 11:48:46	2022-06-01 12:10:20	1.0	2.56	1.0	N
1	2022-06-01 11:38:30	2022-06-01 11:48:56	2.0	1.6	1.0	N
1	2022-06-01 11:51:13	2022-06-01 11:56:06	3.0	0.5	1.0	N
2	2022-06-01 11:37:07	2022-06-01 12:07:35	1.0	3.23	1.0	N
2	2022-06-01 11:46:26	2022-06-01 12:49:57	2.0	18.79	2.0	N
2	2022-06-01 11:18:08	2022-06-01 11:25:00	1.0	1.0	1.0	N
2	2022-06-01 11:23:49	2022-06-01 11:38:04	1.0	2.14	1.0	N

3. This KQL query code shows the number of taxi pickups for each hour.



```
// This query returns the number of taxi pickups per hour
['taxi-data']
| summarize PickupCount = count() by bin(todatetime(tpep_pickup_datetime), 1h)
```

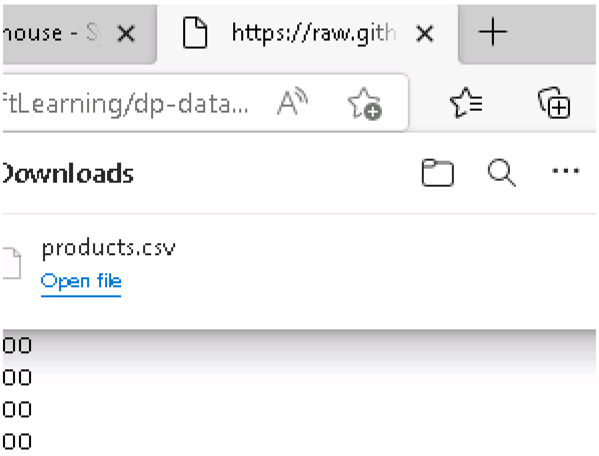
tpep_pickup_datetime	PickupCount
2022-06-01 00:00:00.0000	1
2022-06-01 05:00:00.0000	1
2022-06-01 06:00:00.0000	9
2022-06-01 09:00:00.0000	1
2022-06-01 10:00:00.0000	73
2022-06-01 11:00:00.0000	4,663
2022-06-01 12:00:00.0000	6,669
2022-06-01 13:00:00.0000	6,761
2022-06-01 14:00:00.0000	7,177

Extension: Fabric Exercise

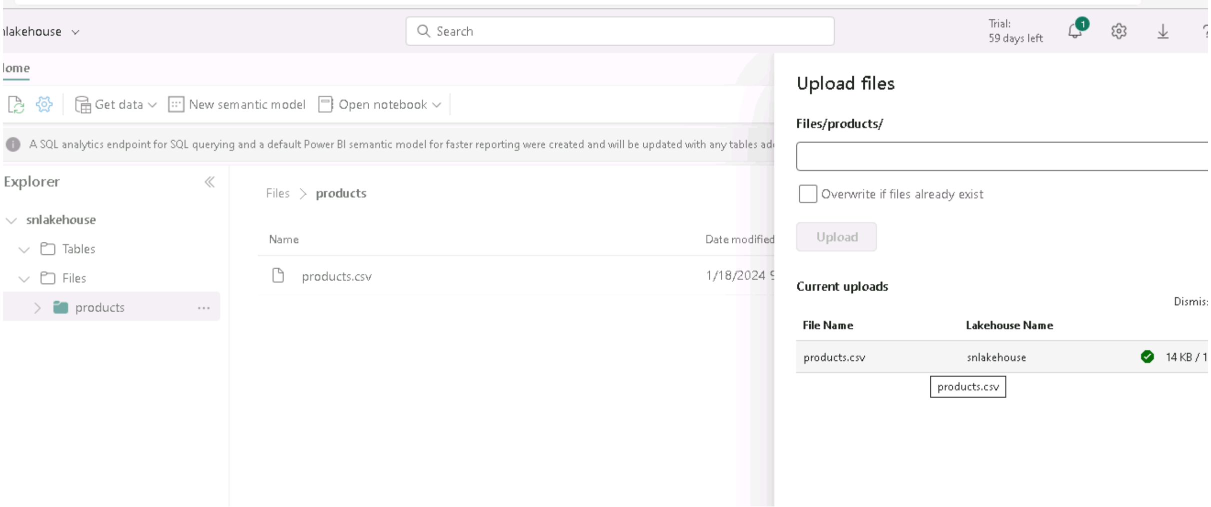
Use delta tables in Apache Spark

Module: Work with Delta Lake tables in Microsoft Fabric

1. Create a lakehouse and upload data
   a. Named new lakehouse
   b. Download the data file for this exercise from `https://github.com/MicrosoftLearning/dp-data/raw/main/products.csv`, saving it as products.csv on your local compute



   c. select New subfolder and create a folder named products.
   d. Upload saved file from downloads.

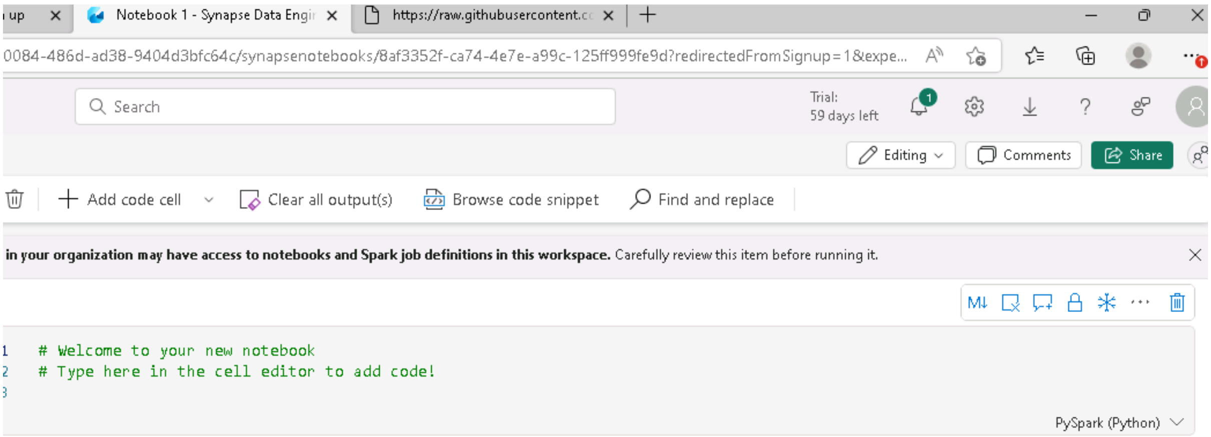


2. Explore data in a dataframe

   a. On the Home page while viewing the contents of the products folder in your datalake, in the Open notebook menu, select New notebook.
   After a few seconds, a new notebook containing a single *cell* will open.

Notebooks are made up of one or more cells that can contain *code* or *markdown* (formatted text).

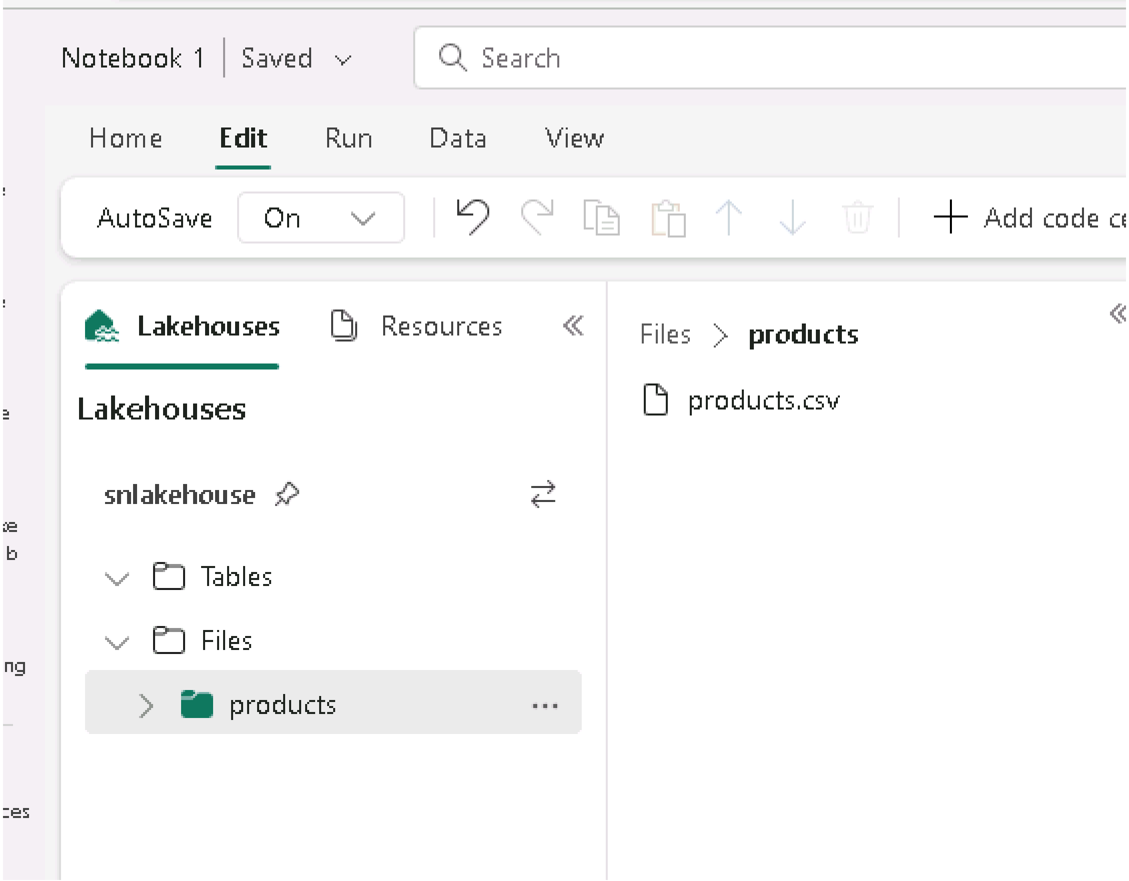
b. Select the existing cell in the notebook, which contains some simple code, and then use its 🗑️ (*Delete*) icon at its top-right to remove it - you will not need this code.


Notebook 1 - Synapse Data Engi
https://raw.githubusercontent.c
0084-486d-ad38-9404d3bfc64c/synapsenotebooks/8af3352f-ca74-4e7e-a99c-125ff999fe9d?redirectedFromSignup=1&expe...
Search
Trial:
59 days left
Editing
Comments
Share
Add code cell
Clear all output(s)
Browse code snippet
Find and replace
PySpark (Python)

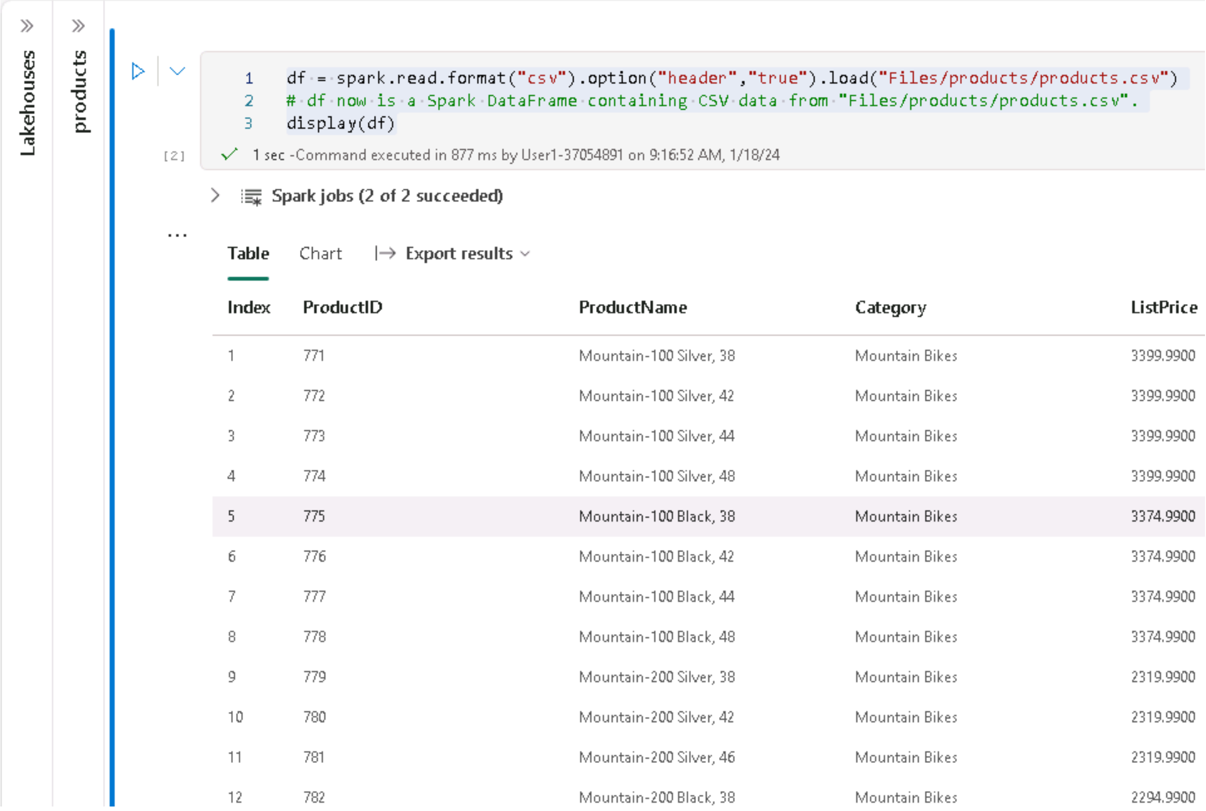
in your organization may have access to notebooks and Spark job definitions in this workspace. Carefully review this item before running it.

```
# Welcome to your new notebook
# Type here in the cell editor to add code!
```

c. In the Lakehouse explorer pane on the left, expand Files and select products to reveal a new pane showing the products.csv file you uploaded previously:



d. In the … menu for products.csv, select Load data > Spark. A new code cell containing the following code should be added to the notebook: Run it.



```
df = spark.read.format("csv").option("header","true").load("Files/products/products.csv")
# df now is a Spark DataFrame containing CSV data from "Files/products/products.csv".
display(df)
```

[2] ✓ 1 sec -Command executed in 877 ms by User1-37054891 on 9:16:52 AM, 1/18/24

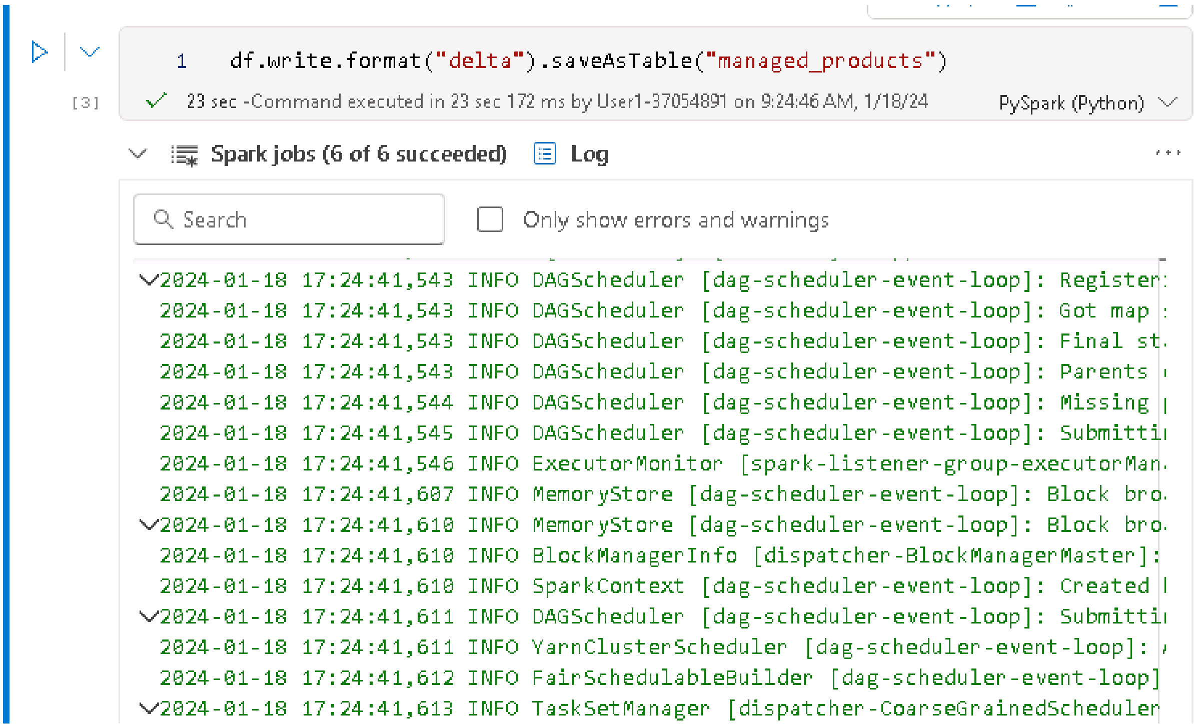
Spark jobs (2 of 2 succeeded)

Index	ProductID	ProductName	Category	ListPrice
1	771	Mountain-100 Silver, 38	Mountain Bikes	3399.9900
2	772	Mountain-100 Silver, 42	Mountain Bikes	3399.9900
3	773	Mountain-100 Silver, 44	Mountain Bikes	3399.9900
4	774	Mountain-100 Silver, 48	Mountain Bikes	3399.9900
5	775	Mountain-100 Black, 38	Mountain Bikes	3374.9900
6	776	Mountain-100 Black, 42	Mountain Bikes	3374.9900
7	777	Mountain-100 Black, 44	Mountain Bikes	3374.9900
8	778	Mountain-100 Black, 48	Mountain Bikes	3374.9900
9	779	Mountain-200 Silver, 38	Mountain Bikes	2319.9900
10	780	Mountain-200 Silver, 42	Mountain Bikes	2319.9900
11	781	Mountain-200 Silver, 46	Mountain Bikes	2319.9900
12	782	Mountain-200 Black, 38	Mountain Bikes	2294.9900

Create delta tables

Create a *managed* table; *Managed* tables are tables for which both the schema metadata and the data files are managed by Fabric. The data files for the table are created in the Tables folder.

a. Under the results returned by the first code cell, use the + Code button to add a new code cell if one doesn't already exist. Then enter the following code in the new cell and run it:



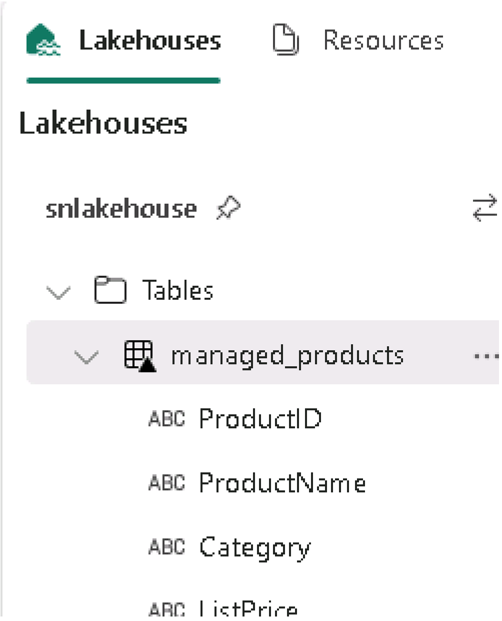
```
df.write.format("delta").saveAsTable("managed_products")
```

[3] ✓ 23 sec -Command executed in 23 sec 172 ms by User1-37054891 on 9:24:46 AM, 1/18/24

PySpark (Python)
Spark jobs (6 of 6 succeeded)
Log
Search
□ Only show errors and warnings

```
2024-01-18 17:24:41,543 INFO DAGScheduler [dag-scheduler-event-loop]: Register
2024-01-18 17:24:41,543 INFO DAGScheduler [dag-scheduler-event-loop]: Got map
2024-01-18 17:24:41,543 INFO DAGScheduler [dag-scheduler-event-loop]: Final st
2024-01-18 17:24:41,543 INFO DAGScheduler [dag-scheduler-event-loop]: Parents
2024-01-18 17:24:41,544 INFO DAGScheduler [dag-scheduler-event-loop]: Missing
2024-01-18 17:24:41,545 INFO DAGScheduler [dag-scheduler-event-loop]: Submitti
2024-01-18 17:24:41,546 INFO ExecutorMonitor [spark-listener-group-executorMan
2024-01-18 17:24:41,607 INFO MemoryStore [dag-scheduler-event-loop]: Block bro
2024-01-18 17:24:41,610 INFO MemoryStore [dag-scheduler-event-loop]: Block bro
2024-01-18 17:24:41,610 INFO BlockManagerInfo [dispatcher-BlockManagerMaster]:
2024-01-18 17:24:41,610 INFO SparkContext [dag-scheduler-event-loop]: Created
2024-01-18 17:24:41,611 INFO DAGScheduler [dag-scheduler-event-loop]: Submitti
2024-01-18 17:24:41,611 INFO YarnClusterScheduler [dag-scheduler-event-loop]:
2024-01-18 17:24:41,612 INFO FairSchedulableBuilder [dag-scheduler-event-loop]
2024-01-18 17:24:41,613 INFO TaskSetManager [dispatcher-CoarseGrainedScheduler
```

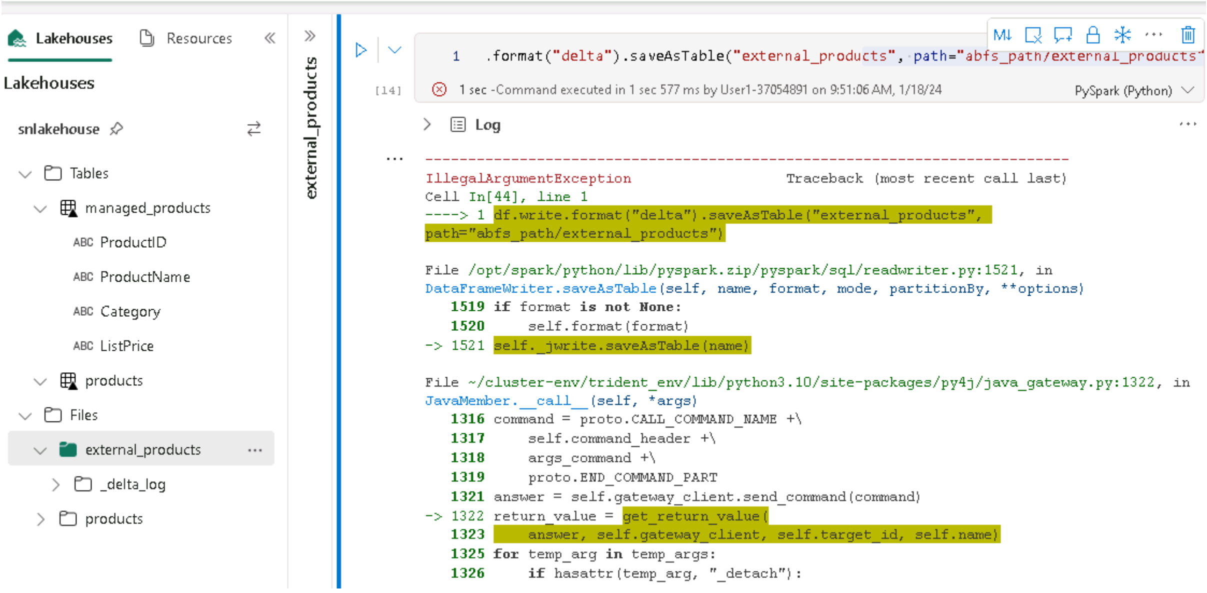
b. In the Lakehouse explorer pane, in the … menu for the Tables folder, select Refresh. Then expand the Tables node and verify that the managed_products table has been created.



Create an *external* table

You can also create *external* tables for which the schema metadata is defined in the metastore for the lakehouse, but the data files are stored in an external location.

a. Add another new code cell, and add the following code to it:



```
.format("delta").saveAsTable("external_products", path="abfs_path/external_products"
```

[14] 1 sec -Command executed in 1 sec 577 ms by User1-37054891 on 9:51:06 AM, 1/18/24

```
---------------------------------------------------------------------------
IllegalArgumentException                  Traceback (most recent call last)
Cell In[44], line 1
----> 1 df.write.format("delta").saveAsTable("external_products",
path="abfs_path/external_products")

File /opt/spark/python/lib/pyspark.zip/pyspark/sql/readwriter.py:1521, in
DataFrameWriter.saveAsTable(self, name, format, mode, partitionBy, **options)
   1519 if format is not None:
   1520     self.format(format)
-> 1521 self._jwrite.saveAsTable(name)

File ~/cluster-env/trident_env/lib/python3.10/site-packages/py4j/java_gateway.py:1322, in
JavaMember.__call__(self, *args)
   1316 command = proto.CALL_COMMAND_NAME +\
   1317     self.command_header +\
   1318     args_command +\
   1319     proto.END_COMMAND_PART
   1321 answer = self.gateway_client.send_command(command)
-> 1322 return_value = get_return_value(
   1323     answer, self.gateway_client, self.target_id, self.name)
   1325 for temp_arg in temp_args:
   1326     if hasattr(temp_arg, "_detach"):
```

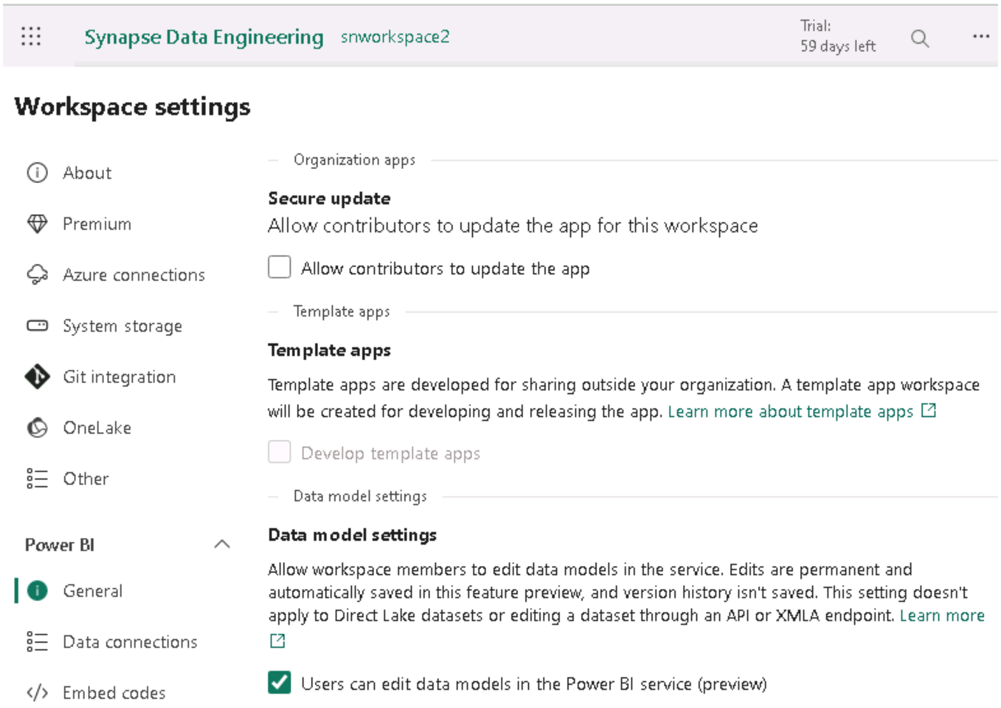
b. In the code you entered into the code cell, replace abfs_path with the path you copied to the clipboard so that the code saves the dataframe as an external table with data files in a folder named external_products in your Files folder location

Create a medallion architecture in a Microsoft Fabric lakehouse

Module: Organize a Fabric lakehouse using medallion architecture design

- Microsoft Fabric is an end-to-end analytics platform that provides a single, integrated environment for data professionals and the business to collaborate on data projects. Fabric provides a set of integrated services that enable you to ingest, store, process, and analyse data in a single environment.

1. Create a workspace: Select synapse data engineering, create a new workspace, navigate to the workspace setting and enable ‘data modelling editing’.



Workspace settings

Organization apps

Secure update

Allow contributors to update the app for this workspace

□ Allow contributors to update the app

Template apps

Template apps

Template apps are developed for sharing outside your organization. A template app workspace will be created for developing and releasing the app. Learn more about template apps

□ Develop template apps

Data model settings

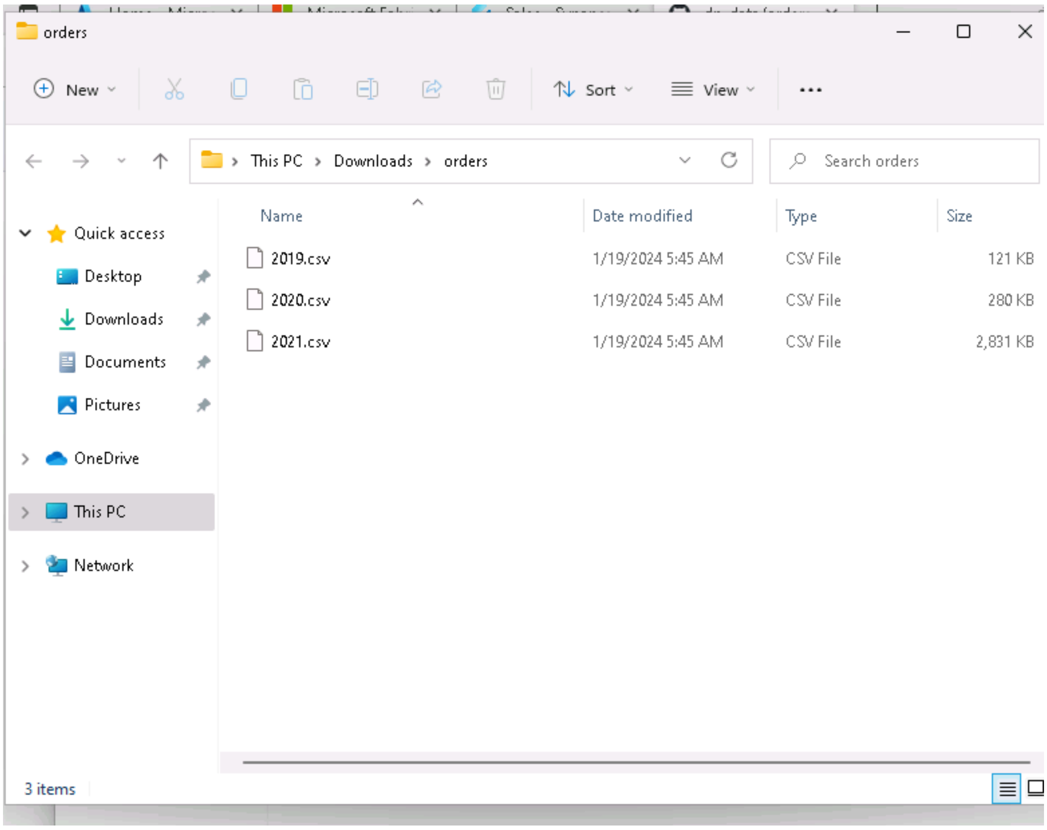
Data model settings

Allow workspace members to edit data models in the service. Edits are permanent and automatically saved in this feature preview, and version history isn't saved. This setting doesn't apply to Direct Lake datasets or editing a dataset through an API or XMLA endpoint. Learn more

☑ Users can edit data models in the Power BI service (preview)

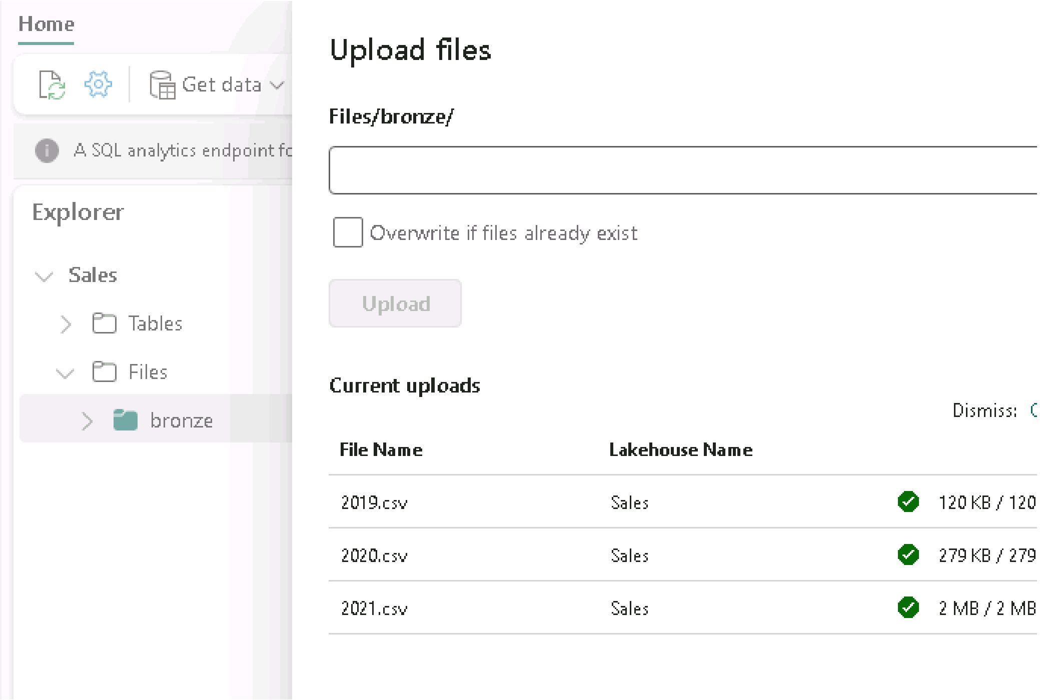
2. Create a lakehouse and upload data to bronze layer

a. created a new lakehouse named sales, extracted folder from github containing three sales data.



Name	Date modified	Type	Size
2019.csv	1/19/2024 5:45 AM	CSV File	121 KB
2020.csv	1/19/2024 5:45 AM	CSV File	280 KB
2021.csv	1/19/2024 5:45 AM	CSV File	2,831 KB

b. Created a new subfolder names bronze under files and upload the extracted files.



Upload files

Files/bronze/

□ Overwrite if files already exist

Current uploads

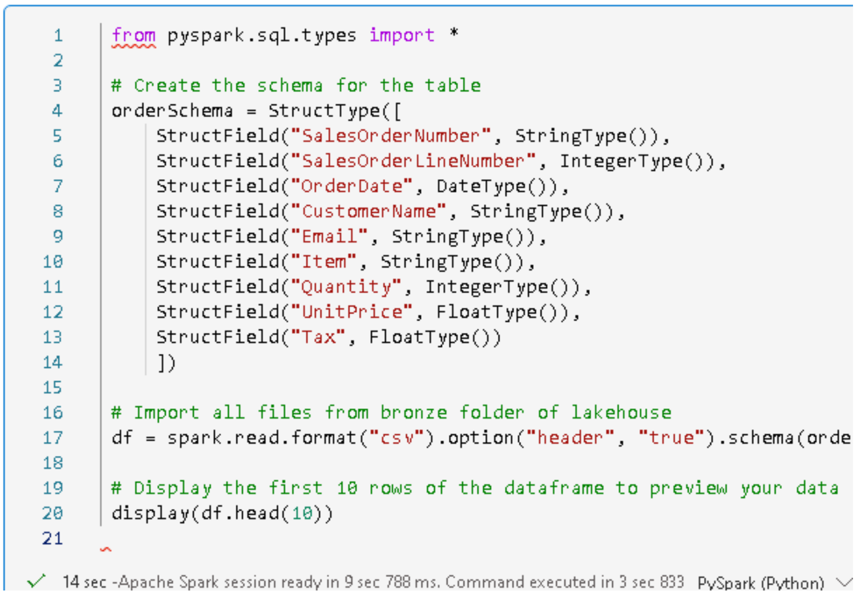
Dismiss: (

File Name	Lakehouse Name	
2019.csv	Sales	✓ 120 KB / 120
2020.csv	Sales	✓ 279 KB / 279
2021.csv	Sales	✓ 2 MB / 2 MB

3. Transform data and load to silver delta table

a. Created a new notebook named transform data for silver and inserted a new sparks code into the code cell and ran it. The code loaded the data from the CSV files in the bronze folder into a Spark dataframe, and then displayed the first 10 rows of the dataframe.

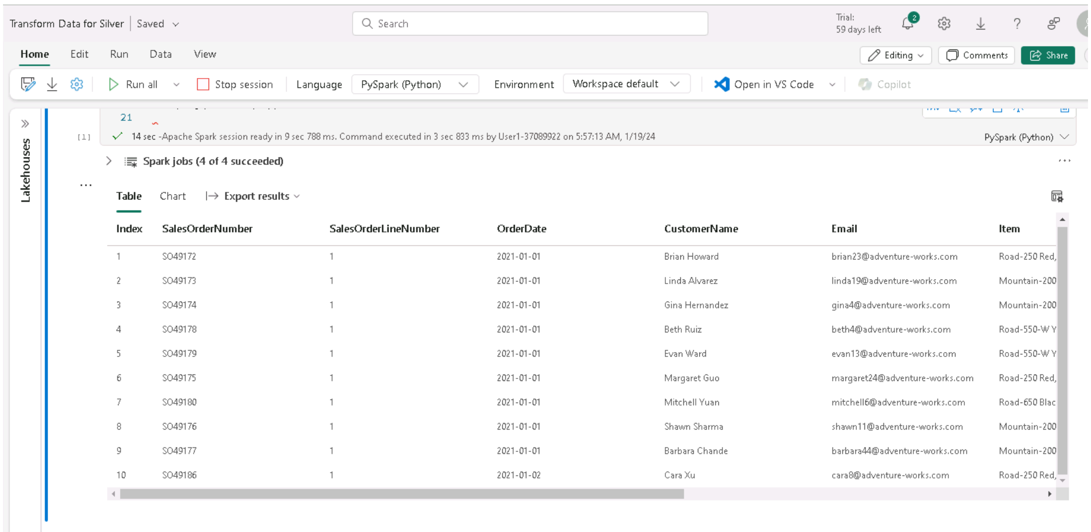
The code:



```
from pyspark.sql.types import *

# Create the schema for the table
orderSchema = StructType([
    StructField("SalesOrderNumber", StringType()),
    StructField("SalesOrderLineNumber", IntegerType()),
    StructField("OrderDate", DateType()),
    StructField("CustomerName", StringType()),
    StructField("Email", StringType()),
    StructField("Item", StringType()),
    StructField("Quantity", IntegerType()),
    StructField("UnitPrice", FloatType()),
    StructField("Tax", FloatType())
    ])

# Import all files from bronze folder of lakehouse
df = spark.read.format("csv").option("header", "true").schema(orde

# Display the first 10 rows of the dataframe to preview your data
display(df.head(10))

```

✓ 14 sec -Apache Spark session ready in 9 sec 788 ms. Command executed in 3 sec 833 PySpark (Python)

The output:



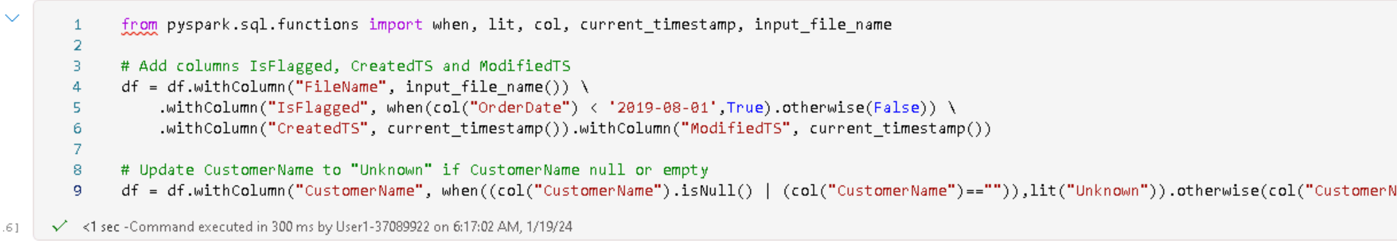
[1] ✓ 14 sec -Apache Spark session ready in 9 sec 788 ms. Command executed in 3 sec 833 ms by User1-37089922 on 5:57:13 AM, 1/19/24 PySpark (Python)

Spark jobs (4 of 4 succeeded)

Table Chart Export results

Index	SalesOrderNumber	SalesOrderLineNumber	OrderDate	CustomerName	Email	Item
1	SO49172	1	2021-01-01	Brian Howard	brian23@adventure-works.com	Road-250 Red,
2	SO49173	1	2021-01-01	Linda Alvarez	linda19@adventure-works.com	Mountain-200
3	SO49174	1	2021-01-01	Gina Hernandez	gina4@adventure-works.com	Mountain-200
4	SO49178	1	2021-01-01	Beth Ruiz	beth4@adventure-works.com	Road-550-W Y
5	SO49179	1	2021-01-01	Evan Ward	evan13@adventure-works.com	Road-550-W Y
6	SO49175	1	2021-01-01	Margaret Guo	margaret24@adventure-works.com	Road-250 Red,
7	SO49180	1	2021-01-01	Mitchell Yuan	mitchell6@adventure-works.com	Road-650 Blac
8	SO49176	1	2021-01-01	Shawn Sharma	shawn11@adventure-works.com	Mountain-200
9	SO49177	1	2021-01-01	Barbara Chande	barbara44@adventure-works.com	Mountain-200
10	SO49186	1	2021-01-02	Cara Xu	cara8@adventure-works.com	Road-250 Red,

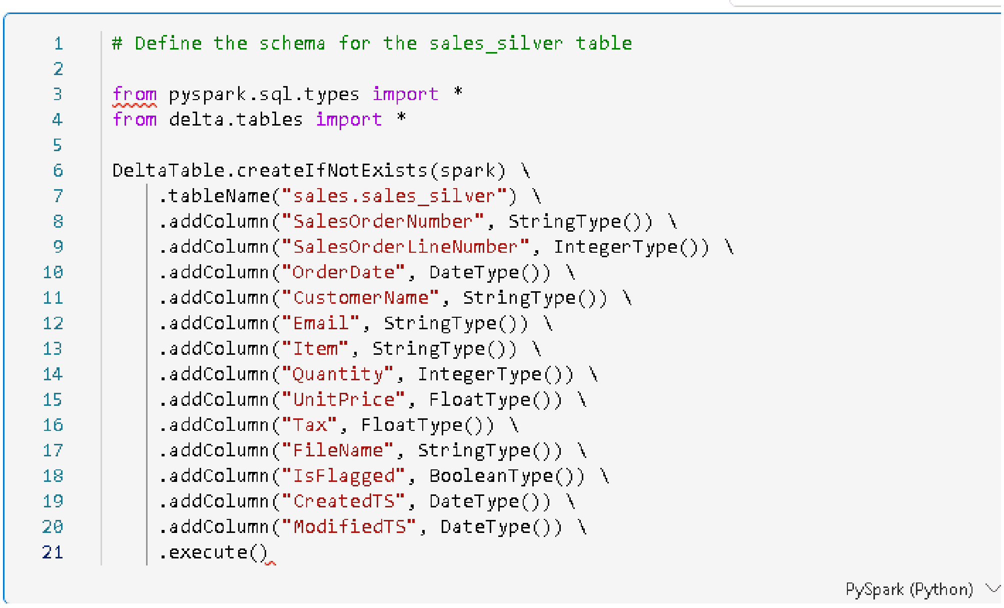
b. Now adding columns for data validation and cleanup. The first line of the code imports the necessary functions from PySpark. then adding new columns to the dataframe to track the source file name, whether the order was flagged as being a before the fiscal year of interest, and when the row was created and modified.



```
from pyspark.sql.functions import when, lit, col, current_timestamp, input_file_name

# Add columns IsFlagged, CreatedTS and ModifiedTS
df = df.withColumn("FileName", input_file_name()) \
    .withColumn("IsFlagged", when(col("OrderDate") < '2019-08-01',True).otherwise(False)) \
    .withColumn("CreatedTS", current_timestamp()).withColumn("ModifiedTS", current_timestamp())

# Update CustomerName to "Unknown" if CustomerName null or empty
df = df.withColumn("CustomerName", when((col("CustomerName").isNull() | (col("CustomerName")=="")),lit("Unknown")).otherwise(col("CustomerN
```

.6] ✓ <1 sec -Command executed in 300 ms by User1-37089922 on 6:17:02 AM, 1/19/24

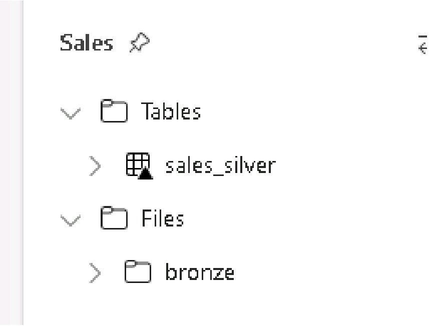
c. Now define the scheme for the sales silver table.



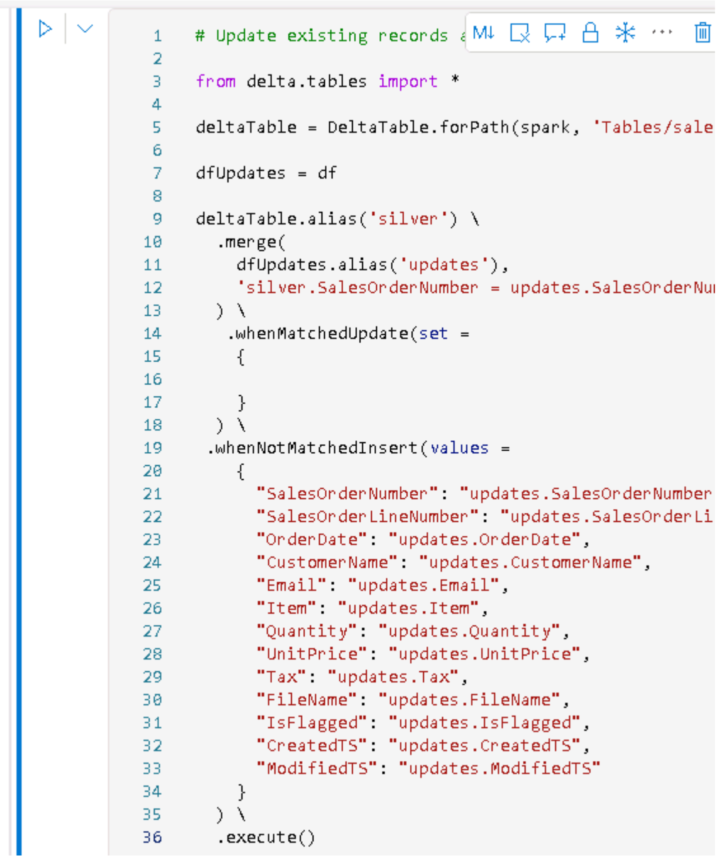
```
# Define the schema for the sales_silver table

from pyspark.sql.types import *
from delta.tables import *

DeltaTable.createIfNotExists(spark) \
    .tableName("sales.sales_silver") \
    .addColumn("SalesOrderNumber", StringType()) \
    .addColumn("SalesOrderLineNumber", IntegerType()) \
    .addColumn("OrderDate", DateType()) \
    .addColumn("CustomerName", StringType()) \
    .addColumn("Email", StringType()) \
    .addColumn("Item", StringType()) \
    .addColumn("Quantity", IntegerType()) \
    .addColumn("UnitPrice", FloatType()) \
    .addColumn("Tax", FloatType()) \
    .addColumn("FileName", StringType()) \
    .addColumn("IsFlagged", BooleanType()) \
    .addColumn("CreatedTS", DateType()) \
    .addColumn("ModifiedTS", DateType()) \
    .execute()
```

PySpark (Python)

d. New sales silver table was created. The triangle indicates that it's a delta table.



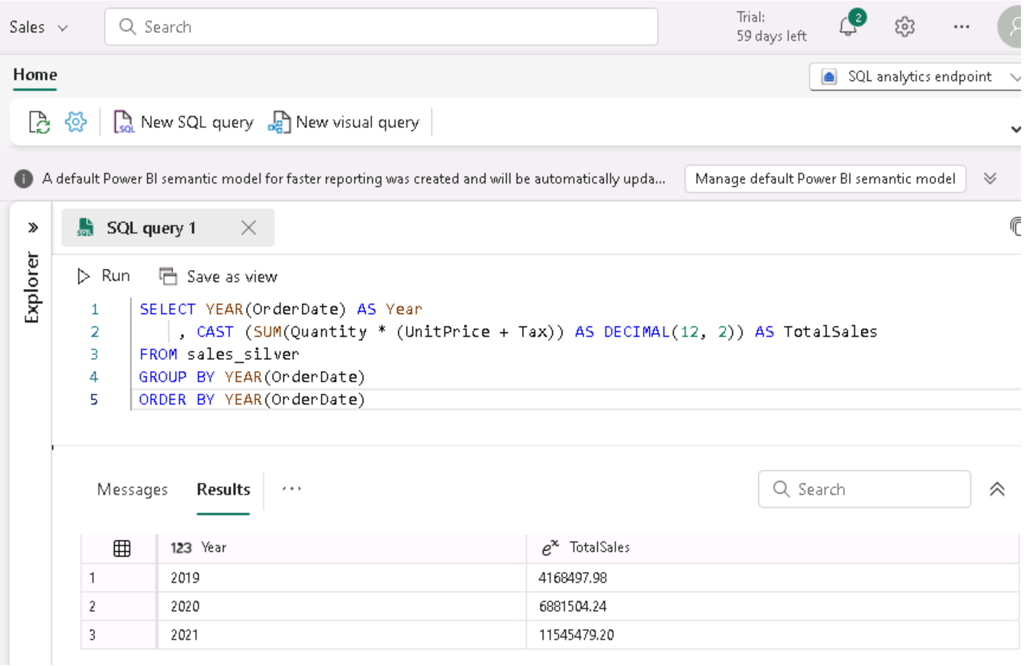
e. perform an upsert operation on a Delta table, updating existing records based on specific conditions and inserting new records when no match is found. Add a new code block. This operation is important because it enables you to update existing records in the table based on the values of specific columns, and insert new records when no match is found.



```
# Update existing records a

from delta.tables import *

deltaTable = DeltaTable.forPath(spark, 'Tables/sale

dfUpdates = df

deltaTable.alias('silver') \
  .merge(
    dfUpdates.alias('updates'),
    'silver.SalesOrderNumber = updates.SalesOrderNu
  ) \
   .whenMatchedUpdate(set =
    {

    }
  ) \
 .whenNotMatchedInsert(values =
    {
      "SalesOrderNumber": "updates.SalesOrderNumber
      "SalesOrderLineNumber": "updates.SalesOrderLi
      "OrderDate": "updates.OrderDate",
      "CustomerName": "updates.CustomerName",
      "Email": "updates.Email",
      "Item": "updates.Item",
      "Quantity": "updates.Quantity",
      "UnitPrice": "updates.UnitPrice",
      "Tax": "updates.Tax",
      "FileName": "updates.FileName",
      "IsFlagged": "updates.IsFlagged",
      "CreatedTS": "updates.CreatedTS",
      "ModifiedTS": "updates.ModifiedTS"
    }
  ) \
  .execute()
```

4. Explore data in the silver layer using the SQL endpoint

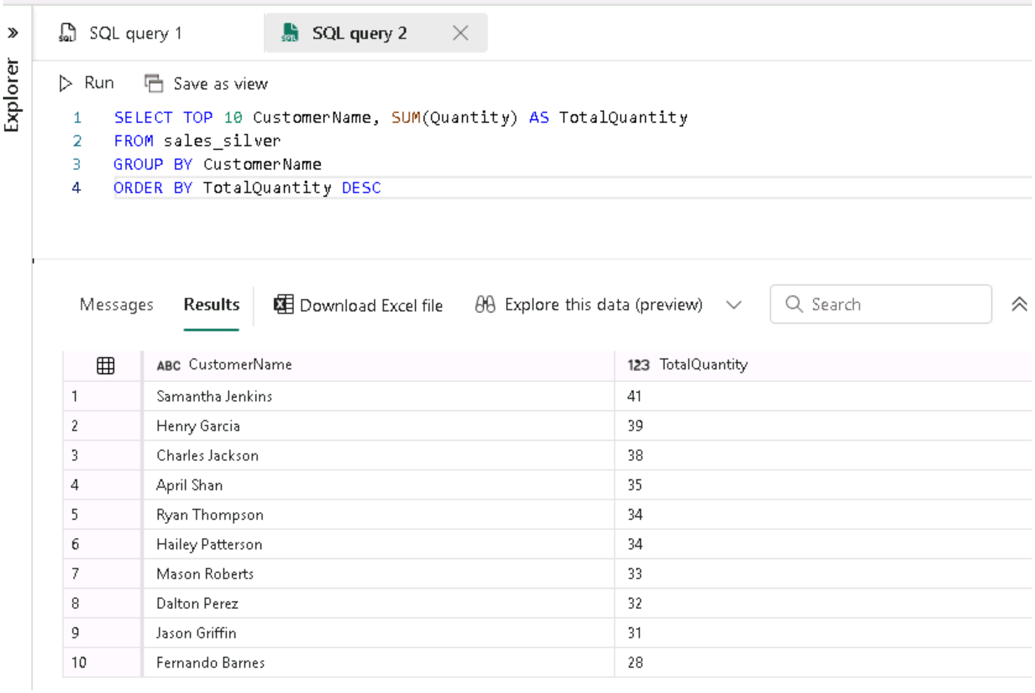
a. This query calculates the total sales for each year in the sales_silver table.



```
SELECT YEAR(OrderDate) AS Year
    , CAST (SUM(Quantity * (UnitPrice + Tax)) AS DECIMAL(12, 2)) AS TotalSales
FROM sales_silver
GROUP BY YEAR(OrderDate)
ORDER BY YEAR(OrderDate)
```

	Year	TotalSales
1	2019	4168497.98
2	2020	6881504.24
3	2021	11545479.20

b. This query calculates the total quantity of items purchased by each customer in the sales_silver table, and then returns the top 10 customers in terms of quantity.

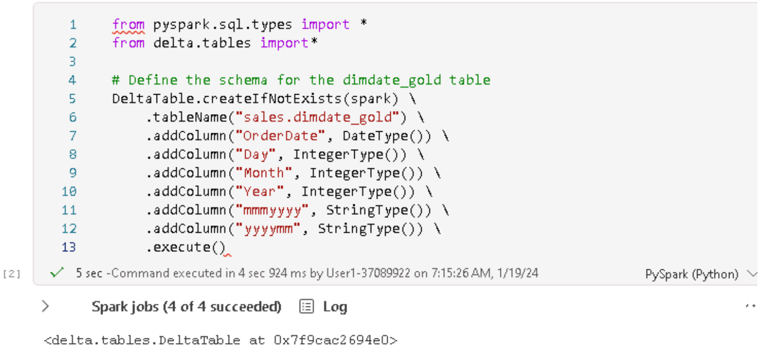


```sql
SELECT TOP 10 CustomerName, SUM(Quantity) AS TotalQuantity
FROM sales_silver
GROUP BY CustomerName
ORDER BY TotalQuantity DESC
```

	CustomerName	TotalQuantity
1	Samantha Jenkins	41
2	Henry Garcia	39
3	Charles Jackson	38
4	April Shan	35
5	Ryan Thompson	34
6	Hailey Patterson	34
7	Mason Roberts	33
8	Dalton Perez	32
9	Jason Griffin	31
10	Fernando Barnes	28

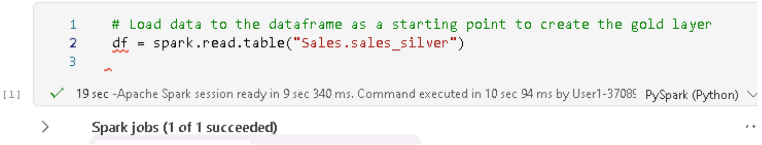
4. Transform Data for Gold Layer

a. Create a new notebook named transform data for gold and add Sales Lakehouse, this will add sales_silver table under tables.
b. Load schema and then add a new code to define the schema



```python
from pyspark.sql.types import *
from delta.tables import*

# Define the schema for the dimdate_gold table
DeltaTable.createIfNotExists(spark) \
    .tableName("sales.dimdate_gold") \
    .addColumn("OrderDate", DateType()) \
    .addColumn("Day", IntegerType()) \
    .addColumn("Month", IntegerType()) \
    .addColumn("Year", IntegerType()) \
    .addColumn("mmmyyyy", StringType()) \
    .addColumn("yyyymm", StringType()) \
    .execute()
```

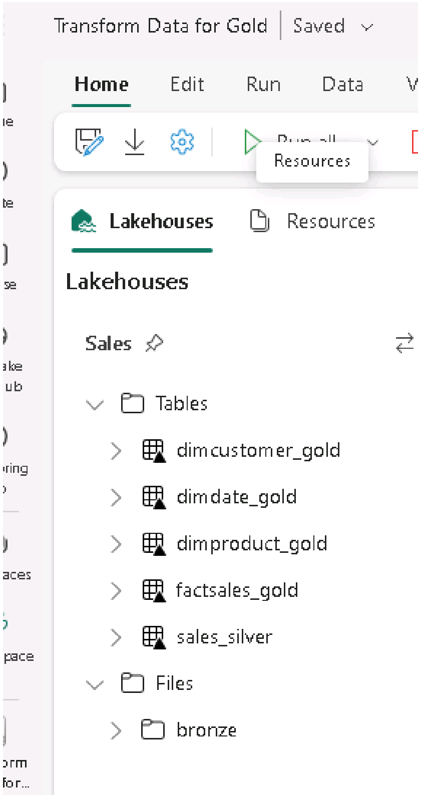
```
[2] ✓ 5 sec -Command executed in 4 sec 924 ms by User1-37089922 on 7:15:26 AM, 1/19/24    PySpark (Python)
Spark jobs (4 of 4 succeeded)   Log
<delta.tables.DeltaTable at 0x7f9cac2694e0>
```

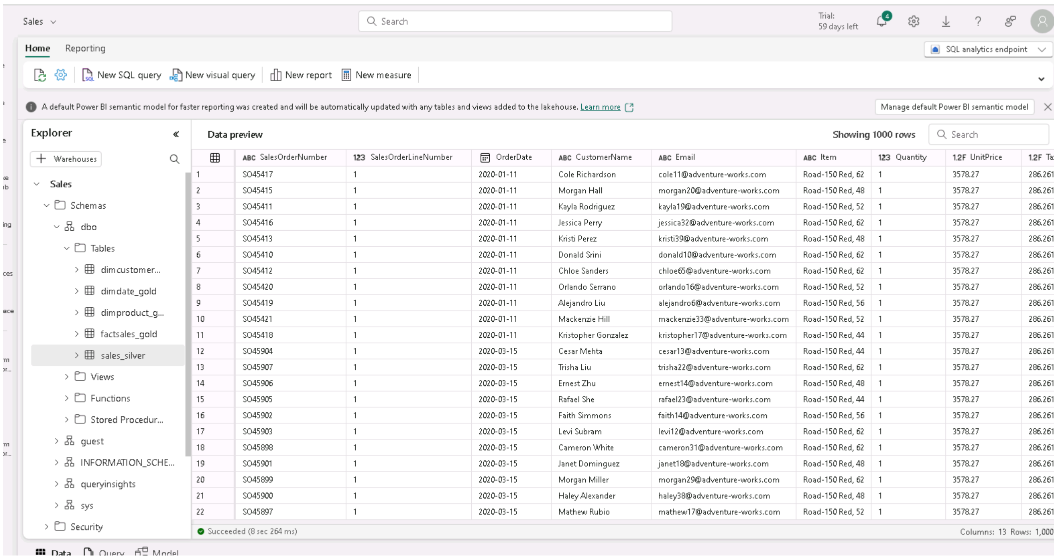


```python
# Load data to the dataframe as a starting point to create the gold layer
df = spark.read.table("Sales.sales_silver")

```

```
[1] ✓ 19 sec -Apache Spark session ready in 9 sec 340 ms. Command executed in 10 sec 94 ms by User1-3708§ PySpark (Python)
Spark jobs (1 of 1 succeeded)
```

c. Created 4 tables which was curated, modelled gold layer that can be used for reporting and analysis.





A default Power BI semantic model for faster reporting was created and will be automatically updated with any tables and views added to the lakehouse. Learn more

Data preview

Showing 1000 rows

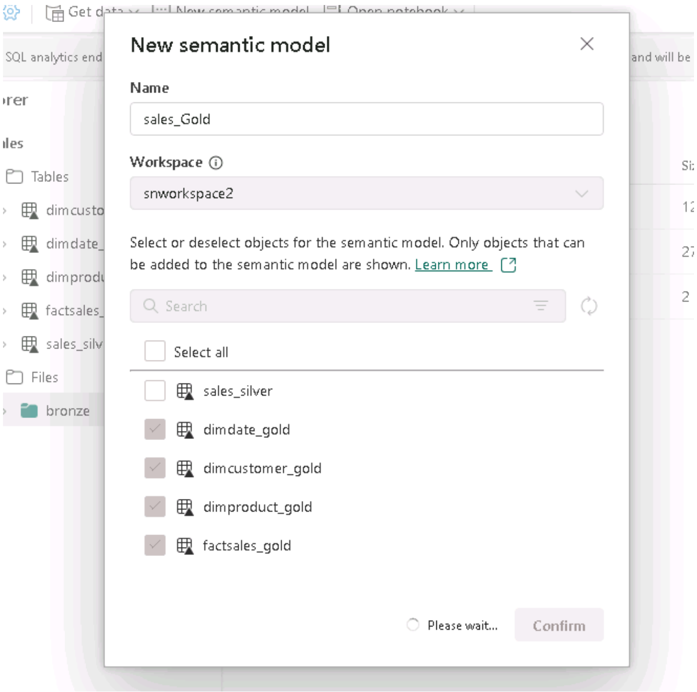
	SalesOrderNumber	SalesOrderLineNumber	OrderDate	CustomerName	Email	Item	Quantity	UnitPrice	Ta
1	SO45417	1	2020-01-11	Cole Richardson	cole11@adventure-works.com	Road-150 Red, 62	1	3578.27	286.261
2	SO45415	1	2020-01-11	Morgan Hall	morgan20@adventure-works.com	Road-150 Red, 48	1	3578.27	286.261
3	SO45411	1	2020-01-11	Kayla Rodriguez	kayla19@adventure-works.com	Road-150 Red, 52	1	3578.27	286.261
4	SO45416	1	2020-01-11	Jessica Perry	jessica32@adventure-works.com	Road-150 Red, 62	1	3578.27	286.261
5	SO45413	1	2020-01-11	Kristi Perez	kristi39@adventure-works.com	Road-150 Red, 48	1	3578.27	286.261
6	SO45410	1	2020-01-11	Donald Srini	donald10@adventure-works.com	Road-150 Red, 62	1	3578.27	286.261
7	SO45412	1	2020-01-11	Chloe Sanders	chloe65@adventure-works.com	Road-150 Red, 62	1	3578.27	286.261
8	SO45420	1	2020-01-11	Orlando Serrano	orlando16@adventure-works.com	Road-150 Red, 52	1	3578.27	286.261
9	SO45419	1	2020-01-11	Alejandro Liu	alejandro6@adventure-works.com	Road-150 Red, 56	1	3578.27	286.261
10	SO45421	1	2020-01-11	Mackenzie Hill	mackenzie33@adventure-works.com	Road-150 Red, 52	1	3578.27	286.261
11	SO45418	1	2020-01-11	Kristopher Gonzalez	kristopher17@adventure-works.com	Road-150 Red, 44	1	3578.27	286.261
12	SO45904	1	2020-03-15	Cesar Mehta	cesar13@adventure-works.com	Road-150 Red, 44	1	3578.27	286.261
13	SO45907	1	2020-03-15	Trisha Liu	trisha22@adventure-works.com	Road-150 Red, 62	1	3578.27	286.261
14	SO45906	1	2020-03-15	Ernest Zhu	ernest14@adventure-works.com	Road-150 Red, 48	1	3578.27	286.261
15	SO45905	1	2020-03-15	Rafael She	rafael23@adventure-works.com	Road-150 Red, 44	1	3578.27	286.261
16	SO45902	1	2020-03-15	Faith Simmons	faith14@adventure-works.com	Road-150 Red, 56	1	3578.27	286.261
17	SO45903	1	2020-03-15	Levi Subram	levi12@adventure-works.com	Road-150 Red, 62	1	3578.27	286.261
18	SO45898	1	2020-03-15	Cameron White	cameron31@adventure-works.com	Road-150 Red, 62	1	3578.27	286.261
19	SO45901	1	2020-03-15	Janet Dominguez	janet18@adventure-works.com	Road-150 Red, 48	1	3578.27	286.261
20	SO45899	1	2020-03-15	Morgan Miller	morgan29@adventure-works.com	Road-150 Red, 62	1	3578.27	286.261
21	SO45900	1	2020-03-15	Haley Alexander	haley38@adventure-works.com	Road-150 Red, 48	1	3578.27	286.261
22	SO45897	1	2020-03-15	Mathew Rubio	mathew17@adventure-works.com	Road-150 Red, 52	1	3578.27	286.261

Succeeded (8 sec 264 ms)

Columns: 13 Rows: 1,000

5. Create a database

   a. create a new dataset that includes the gold tables.



New semantic model

Name: sales_Gold

Workspace: snworkspace2

Select or deselect objects for the semantic model. Only objects that can be added to the semantic model are shown. Learn more

□ Select all

□ sales_silver

☑ dimdate_gold

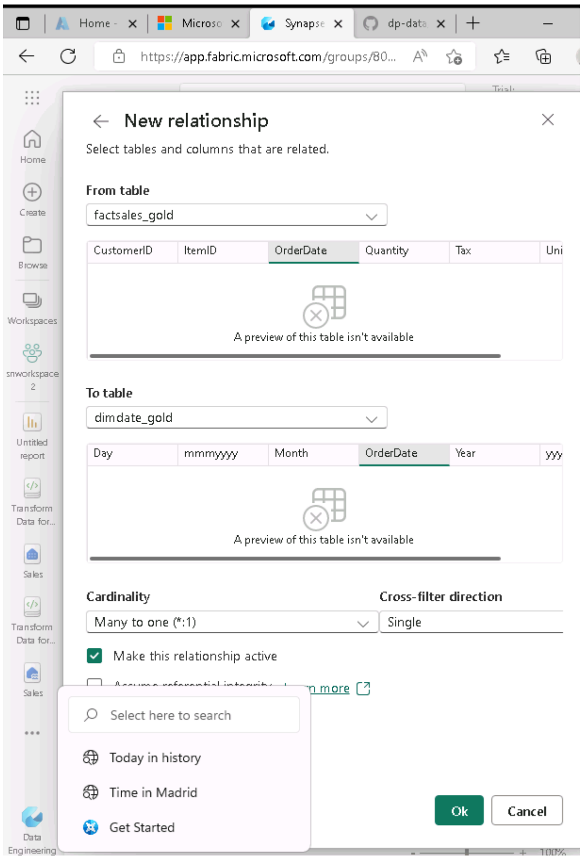
☑ dimcustomer_gold

☑ dimproduct_gold

☑ factsales_gold

B. set relationship.

The common relationship is sharing order date and it is many to 1.



New relationship

Select tables and columns that are related.

From table: factsales_gold

CustomerID	ItemID	OrderDate	Quantity	Tax	Uni

A preview of this table isn't available

To table: dimdate_gold

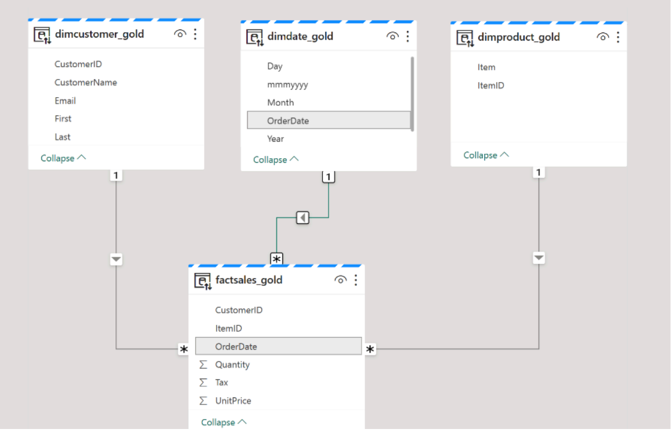
Day	mmmyyyy	Month	OrderDate	Year	yyy

A preview of this table isn't available

Cardinality: Many to one (*:1)

Cross-filter direction: Single

☑ Make this relationship active


dimcustomer_gold
CustomerID
CustomerName
Email
First
Last
Collapse
dimdate_gold
Day
mmmyyyy
Month
OrderDate
Year
Collapse
dimproduct_gold
Item
ItemID
Collapse
1
1
1
*
*
*
factsales_gold
CustomerID
ItemID
OrderDate
Quantity
Tax
UnitPrice
Collapse