Lab 02: Extending Data Preparation with Python notebook

Introduction

In this lab, you will use a Python notebook to execute a resource expensive data preparation activity using Spark cluster managed by Fabric.

Objectives

After completing this lab, you will be better able to:

1. Import the various Python notebooks to the Fabric Environment
2. Run a first notebook to ingest data from the Bronze zone (CSV files) to DELTA tables
3. Run a second notebook to prepare a bigger sales dataset, merging 2 files (using SORT and MERGE join) to the DELTA table

Estimated time to complete this lab

60 minutes

Contents

Lab Prerequisites

- Workspace: Fabric, Power Premium or Fabric trial
- Individual license: Power Pro or Premium Per User account

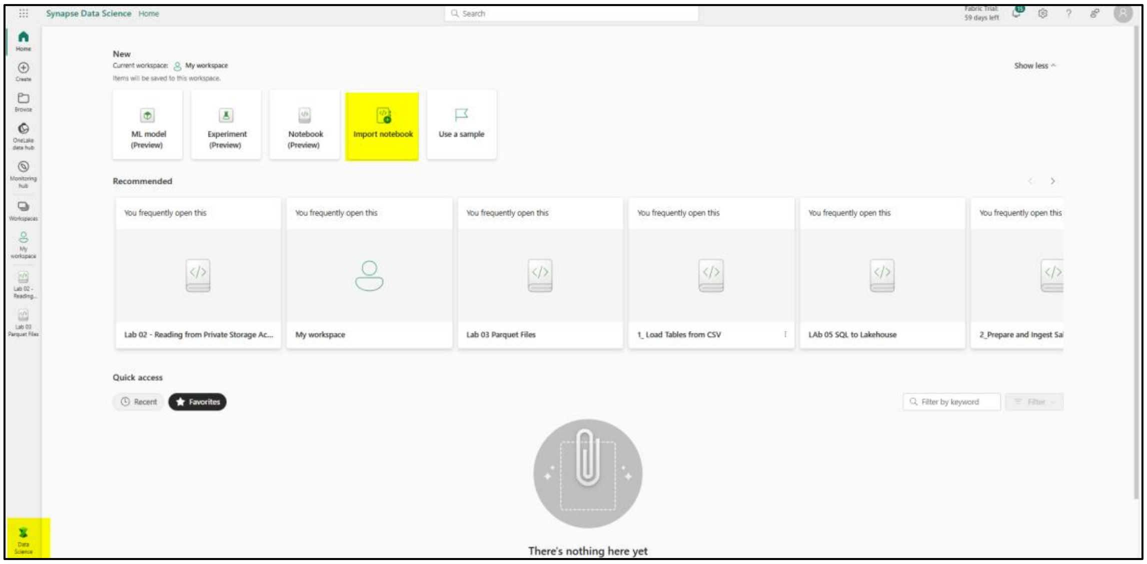
Information provided by your training provider

- Trial tenant (if applicable): login & password, workspace to use for the lab.
- Azure Data Lake Gen2 (containing data sources): account name & shared access signature.

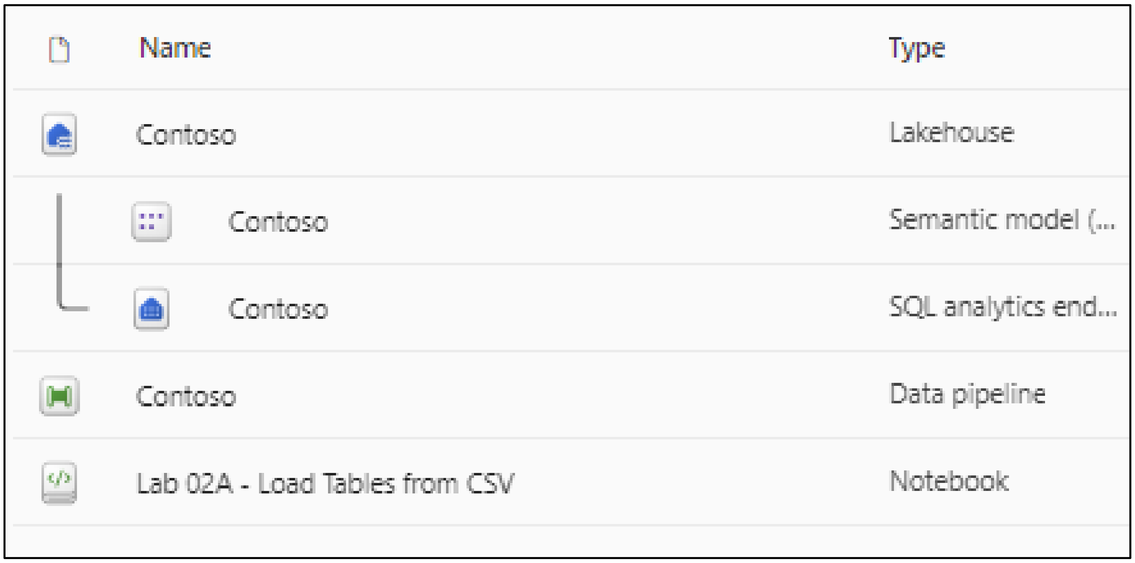
Task 1: Import Python notebooks

In this task, you will import 2 Python notebooks, to be used later for data preparation and ingestion.

- Choose the Data Science from the Microsoft Fabric menu and select Import Notebook.
- Choose upload and select the notebook Lab 02A - Load Tables from CSV.ipynb

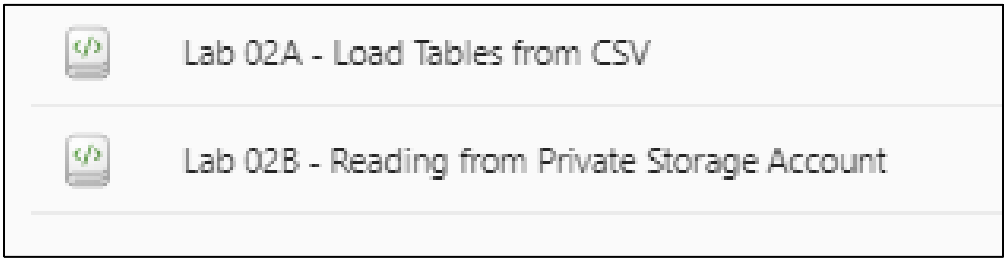


- The imported Notebook should appear in the Workspace



Name	Type
Contoso	Lakehouse
Contoso	Semantic model (...
Contoso	SQL analytics end...
Contoso	Data pipeline
Lab 02A - Load Tables from CSV	Notebook

- Repeat the same operation for the second notebook Lab 02B - Reading from Private Storage Account.ipynb
- The 2 notebooks should appear in the workspace



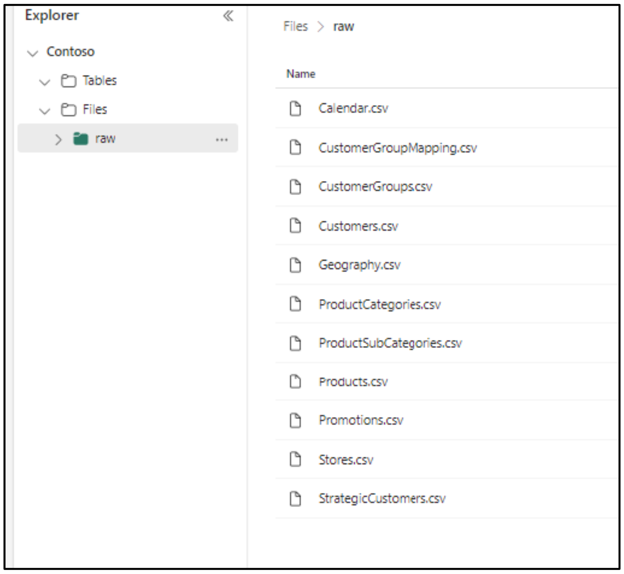
Lab 02A - Load Tables from CSV

Lab 02B - Reading from Private Storage Account

- In the next Task, you will have to :
    - Configure the default Lakehouse for each notebook
    - Configure some settings in the code
    - Execute the Python code and understand the logic.

Task 2: Ingest data from Bronze zone (CSV files) to the Gold zone (Delta tables)

In this task, you will use a Notebook to read data from CSV files stored in the Bronze zone (also named unmanaged zone) and ingest the content to the Gold zone using the DELTA format:



Explorer

Contoso

Tables

Files

raw

Files > raw

Name

Calendar.csv

CustomerGroupMapping.csv

CustomerGroups.csv

Customers.csv

Geography.csv

ProductCategories.csv

ProductSubCategories.csv

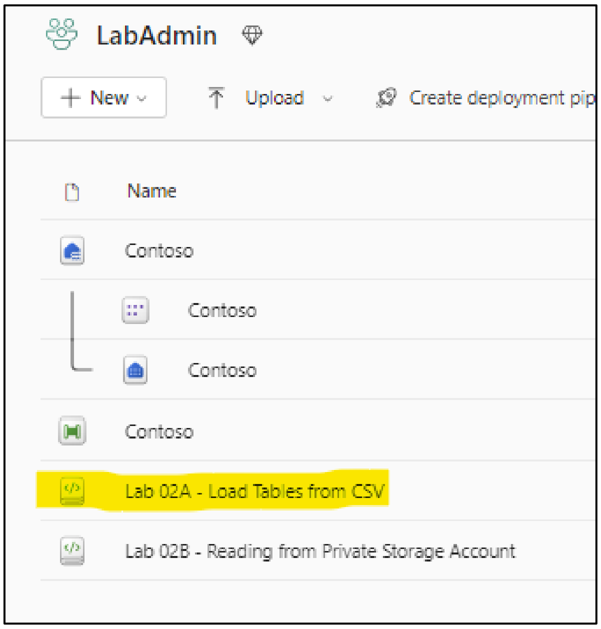
Products.csv

Promotions.csv

Stores.csv

StrategicCustomers.csv

- For each CSV files, the script will detect the file structure, create the corresponding table structure, and ingest the file content to the table dynamically.
- From the Lab workspace, open the notebook named "Lab 02A – Load Tables from CSV"



LabAdmin

Name

Contoso

Contoso

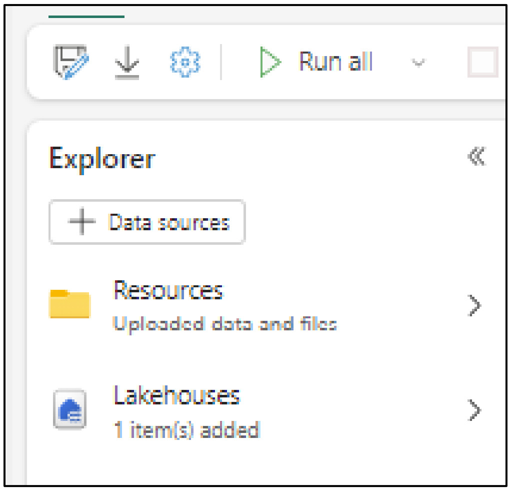
Contoso

Contoso

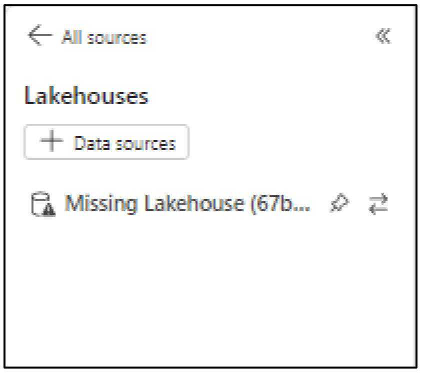
Lab 02A - Load Tables from CSV

Lab 02B - Reading from Private Storage Account

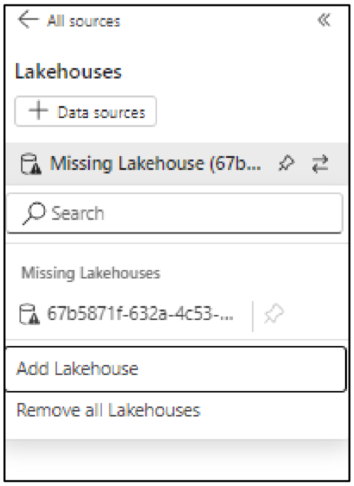
- On the Explorer panel, select the Lakehouses item



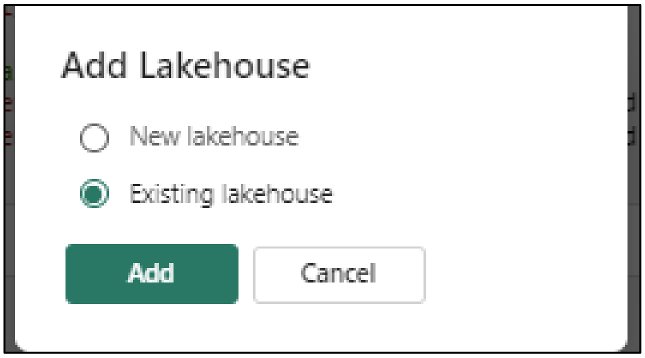
- The imported Notebook contains a reference to a Lakehouse which does not exist in your environment, that is why the Missing Lakehouse warning appears. You will have to attach the Notebook to the Contoso Lakehouse created previously.



- Click on the Missing Lakehouse warning and select Add Lakehouse.



- Select Existing Lakehouse and click on Add

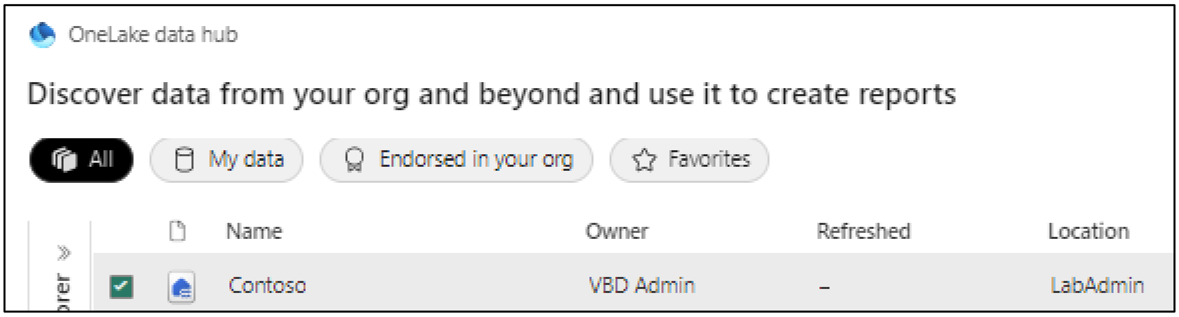


Add Lakehouse

○ New lakehouse

● Existing lakehouse

- On the OneLake data hub, select the Contoso Lakehouse from the Lab Workspace

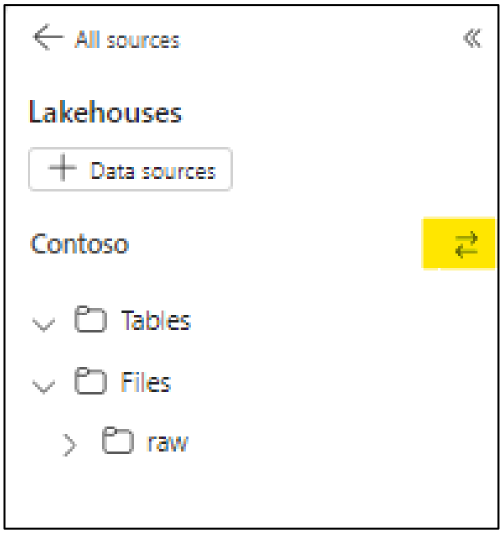


OneLake data hub

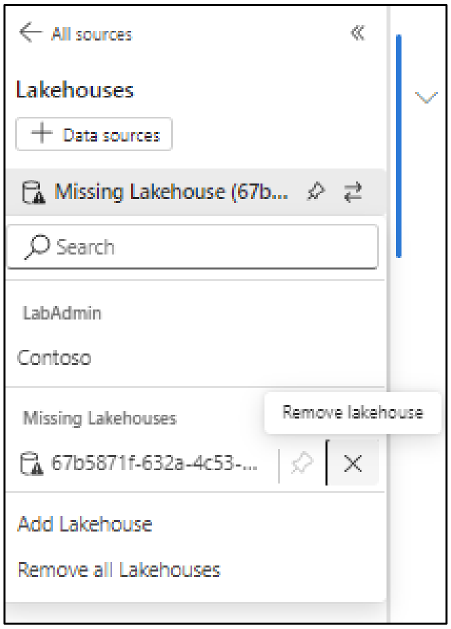
Discover data from your org and beyond and use it to create reports

	Name	Owner	Refreshed	Location
☑	Contoso	VBD Admin	–	LabAdmin

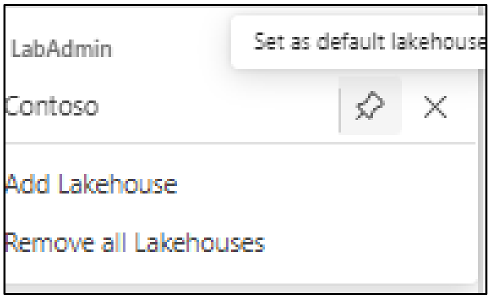
- The structure of the selected Lakehouse should now appear in the Notebook interface. Click on the Lakehouse selector.



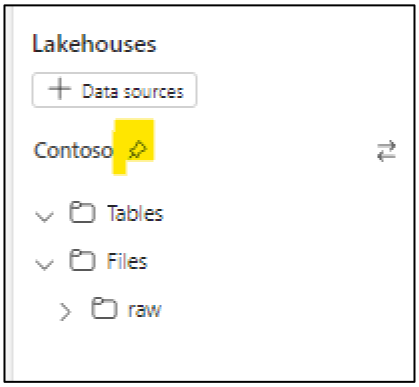
- Remove the missing Lakehouse reference



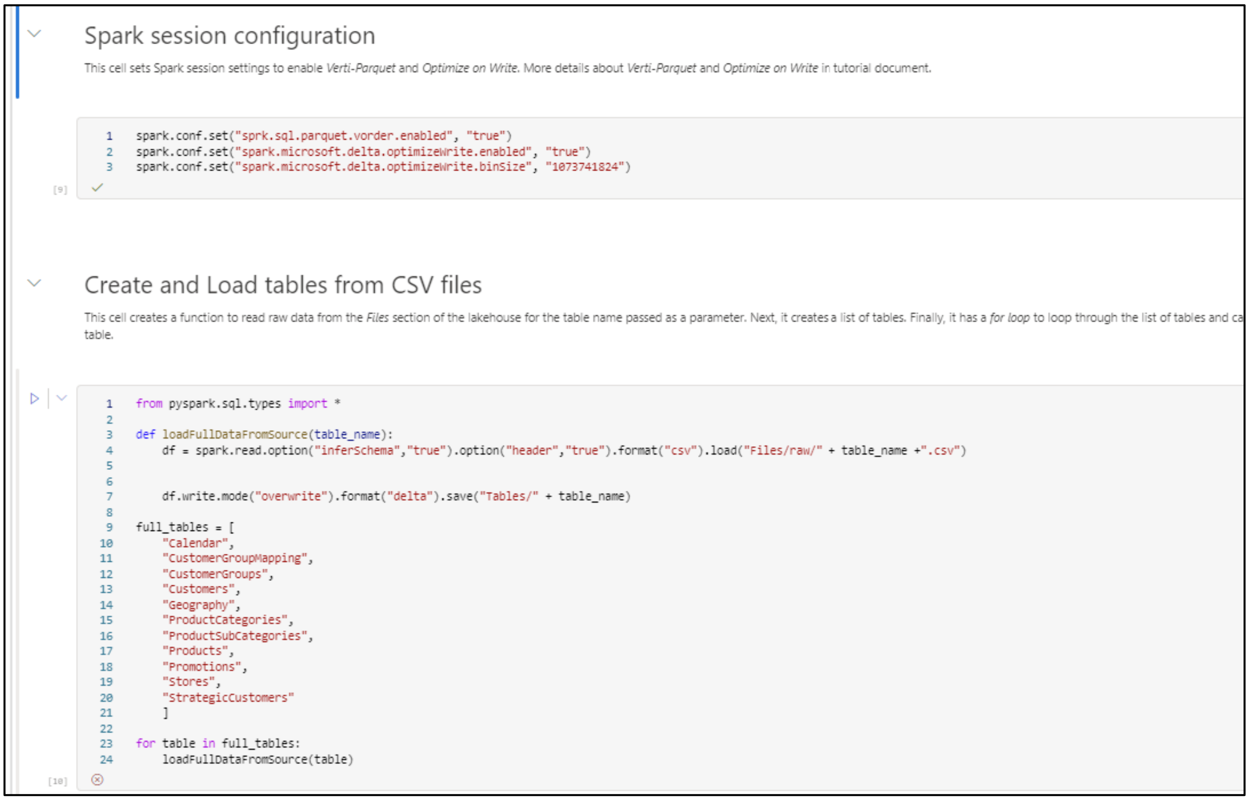
- Then select the Contoso Lakehouse and set it as the default Lakehouse for your notebook.



- As the default Lakehouse, the Contoso Lakehouse should now be pinned.



- Review the Python script content:
    - Spark session configuration
    - Create and Load tables from CSV files.



Spark session configuration

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

```
spark.conf.set("sprk.sql.parquet.vorder.enabled", "true")
spark.conf.set("spark.microsoft.delta.optimizeWrite.enabled", "true")
spark.conf.set("spark.microsoft.delta.optimizeWrite.binSize", "1073741824")
```

[9]

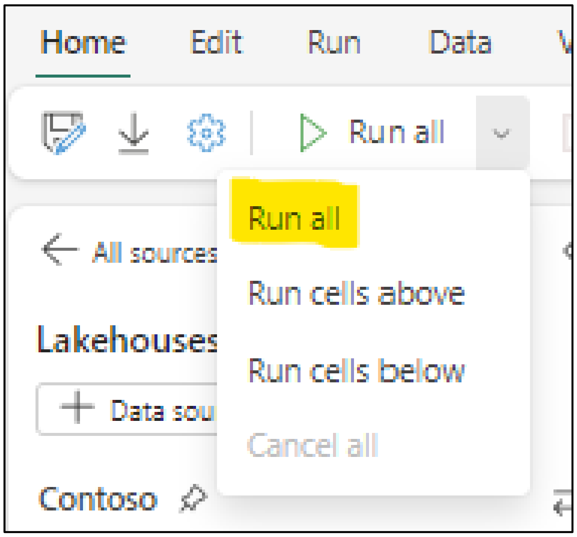
Create and Load tables from CSV files

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 tables. Finally, it has a *for loop* to loop through the list of tables and ca table.

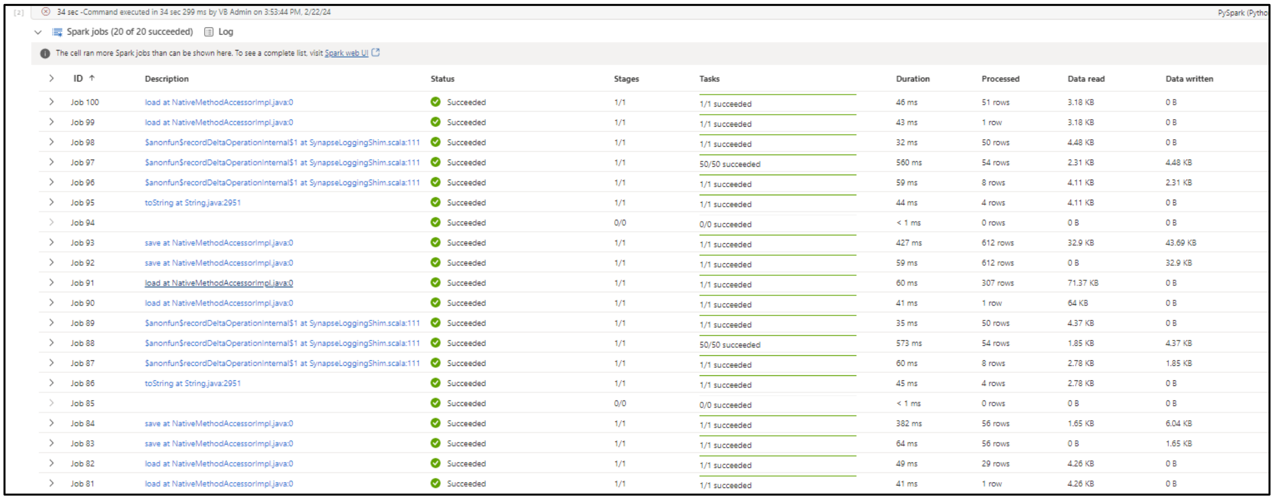
```
from pyspark.sql.types import *

def loadFullDataFromSource(table_name):
    df = spark.read.option("inferSchema","true").option("header","true").format("csv").load("Files/raw/" + table_name +".csv")


    df.write.mode("overwrite").format("delta").save("Tables/" + table_name)

full_tables = [
    "Calendar",
    "CustomerGroupMapping",
    "CustomerGroups",
    "Customers",
    "Geography",
    "ProductCategories",
    "ProductSubCategories",
    "Products",
    "Promotions",
    "Stores",
    "StrategicCustomers"
    ]

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

[10]

- Click on Run all to execute the full Python script



- At the end of the job execution, you can get more details about each Spark job involved during the code execution.



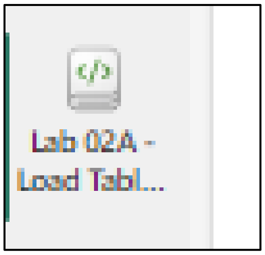
[2] 34 sec -Command executed in 34 sec 299 ms by VB Admin on 3:53:44 PM, 2/22/24 PySpark (Pytho

Spark jobs (20 of 20 succeeded) Log

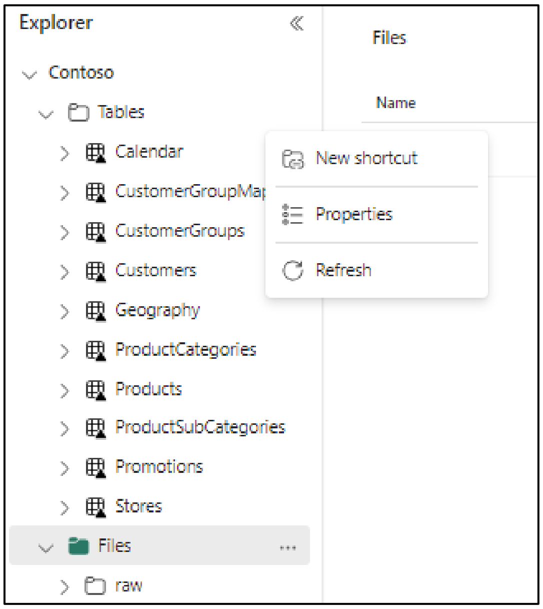
The cell ran more Spark jobs than can be shown here. To see a complete list, visit Spark web UI

ID ↑	Description	Status	Stages	Tasks	Duration	Processed	Data read	Data written
Job 100	load at NativeMethodAccessorImpl.java:0	✓ Succeeded	1/1	1/1 succeeded	46 ms	51 rows	3.18 KB	0 B
Job 99	load at NativeMethodAccessorImpl.java:0	✓ Succeeded	1/1	1/1 succeeded	43 ms	1 row	3.18 KB	0 B
Job 98	$anonfun$recordDeltaOperationInternal$1 at SynapseLoggingShim.scala:111	✓ Succeeded	1/1	1/1 succeeded	32 ms	50 rows	4.48 KB	0 B
Job 97	$anonfun$recordDeltaOperationInternal$1 at SynapseLoggingShim.scala:111	✓ Succeeded	1/1	50/50 succeeded	560 ms	54 rows	2.31 KB	4.48 KB
Job 96	$anonfun$recordDeltaOperationInternal$1 at SynapseLoggingShim.scala:111	✓ Succeeded	1/1	1/1 succeeded	59 ms	8 rows	4.11 KB	2.31 KB
Job 95	toString at String.java:2951	✓ Succeeded	1/1	1/1 succeeded	44 ms	4 rows	4.11 KB	0 B
Job 94		✓ Succeeded	0/0	0/0 succeeded	< 1 ms	0 rows	0 B	0 B
Job 93	save at NativeMethodAccessorImpl.java:0	✓ Succeeded	1/1	1/1 succeeded	427 ms	612 rows	32.9 KB	43.69 KB
Job 92	save at NativeMethodAccessorImpl.java:0	✓ Succeeded	1/1	1/1 succeeded	59 ms	612 rows	0 B	32.9 KB
Job 91	load at NativeMethodAccessorImpl.java:0	✓ Succeeded	1/1	1/1 succeeded	60 ms	307 rows	71.37 KB	0 B
Job 90	load at NativeMethodAccessorImpl.java:0	✓ Succeeded	1/1	1/1 succeeded	41 ms	1 row	64 KB	0 B
Job 89	$anonfun$recordDeltaOperationInternal$1 at SynapseLoggingShim.scala:111	✓ Succeeded	1/1	1/1 succeeded	35 ms	50 rows	4.37 KB	0 B
Job 88	$anonfun$recordDeltaOperationInternal$1 at SynapseLoggingShim.scala:111	✓ Succeeded	1/1	50/50 succeeded	573 ms	54 rows	1.85 KB	4.37 KB
Job 87	$anonfun$recordDeltaOperationInternal$1 at SynapseLoggingShim.scala:111	✓ Succeeded	1/1	1/1 succeeded	60 ms	8 rows	2.78 KB	1.85 KB
Job 86	toString at String.java:2951	✓ Succeeded	1/1	1/1 succeeded	45 ms	4 rows	2.78 KB	0 B
Job 85		✓ Succeeded	0/0	0/0 succeeded	< 1 ms	0 rows	0 B	0 B
Job 84	save at NativeMethodAccessorImpl.java:0	✓ Succeeded	1/1	1/1 succeeded	382 ms	56 rows	1.65 KB	6.04 KB
Job 83	save at NativeMethodAccessorImpl.java:0	✓ Succeeded	1/1	1/1 succeeded	64 ms	56 rows	0 B	1.65 KB
Job 82	load at NativeMethodAccessorImpl.java:0	✓ Succeeded	1/1	1/1 succeeded	49 ms	29 rows	4.26 KB	0 B
Job 81	load at NativeMethodAccessorImpl.java:0	✓ Succeeded	1/1	1/1 succeeded	41 ms	1 row	4.26 KB	0 B

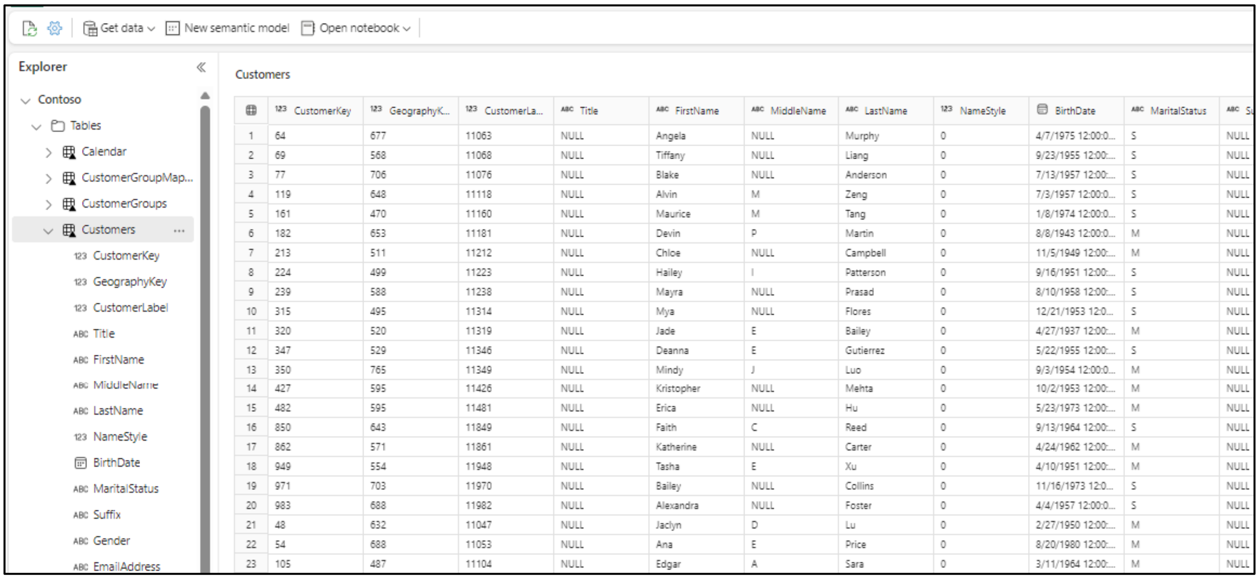
- Close the Opened notebook.



- In the Lakehouse explorer, expand the Tables node to reveal the created tables. Use the Refresh tables if necessary.



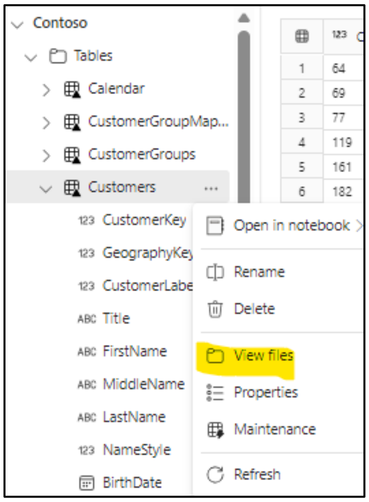
- By selecting one of the tables, you can see the table content.

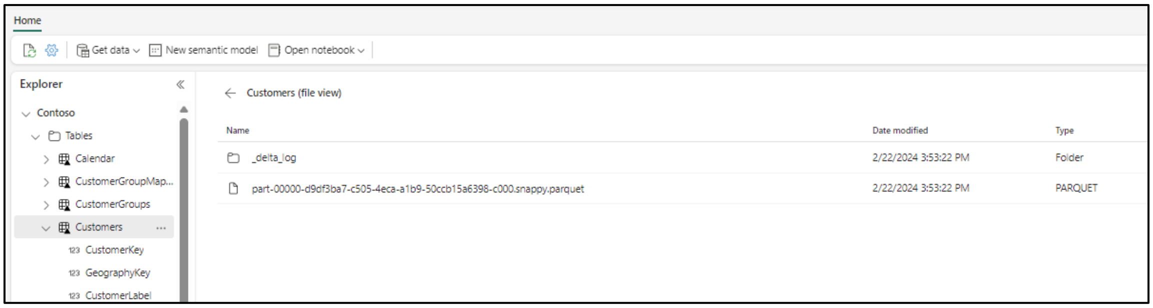


Customers

	CustomerKey	GeographyK...	CustomerLa...	Title	FirstName	MiddleName	LastName	NameStyle	BirthDate	MaritalStatus	Su
1	64	677	11063	NULL	Angela	NULL	Murphy	0	4/7/1975 12:00:0...	S	NULL
2	69	568	11068	NULL	Tiffany	NULL	Liang	0	9/23/1955 12:00:...	S	NULL
3	77	706	11076	NULL	Blake	NULL	Anderson	0	7/13/1957 12:00:...	S	NULL
4	119	648	11118	NULL	Alvin	M	Zeng	0	7/3/1957 12:00:0...	S	NULL
5	161	470	11160	NULL	Maurice	M	Tang	0	1/8/1974 12:00:0...	S	NULL
6	182	653	11181	NULL	Devin	P	Martin	0	8/8/1943 12:00:0...	M	NULL
7	213	511	11212	NULL	Chloe	NULL	Campbell	0	11/5/1949 12:00:...	M	NULL
8	224	499	11223	NULL	Hailey	I	Patterson	0	9/16/1951 12:00:...	S	NULL
9	239	588	11238	NULL	Mayra	NULL	Prasad	0	8/10/1958 12:00:...	S	NULL
10	315	495	11314	NULL	Mya	NULL	Flores	0	12/21/1953 12:0...	S	NULL
11	320	520	11319	NULL	Jade	E	Bailey	0	4/27/1937 12:00:...	M	NULL
12	347	529	11346	NULL	Deanna	E	Gutierrez	0	5/22/1955 12:00:...	S	NULL
13	350	765	11349	NULL	Mindy	J	Luo	0	9/3/1954 12:00:0...	M	NULL
14	427	595	11426	NULL	Kristopher	NULL	Mehta	0	10/2/1953 12:00:...	M	NULL
15	482	595	11481	NULL	Erica	NULL	Hu	0	5/23/1973 12:00:...	M	NULL
16	850	643	11849	NULL	Faith	C	Reed	0	9/13/1964 12:00:...	S	NULL
17	862	571	11861	NULL	Katherine	NULL	Carter	0	4/24/1962 12:00:...	M	NULL
18	949	554	11948	NULL	Tasha	E	Xu	0	4/10/1951 12:00:...	M	NULL
19	971	703	11970	NULL	Bailey	NULL	Collins	0	11/16/1973 12:0...	S	NULL
20	983	688	11982	NULL	Alexandra	NULL	Foster	0	4/4/1957 12:00:0...	S	NULL
21	48	632	11047	NULL	Jaclyn	D	Lu	0	2/27/1950 12:00:...	M	NULL
22	54	688	11053	NULL	Ana	E	Price	0	8/20/1980 12:00:...	M	NULL
23	105	487	11104	NULL	Edgar	A	Sara	0	3/11/1964 12:00:...	M	NULL

- You can also see the underlying DELTA table structure to display the PARQUET file(s) dans the DELTA log file.



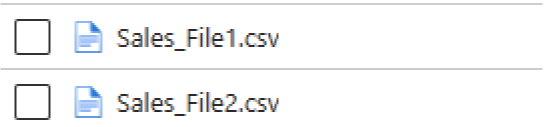


Customers (file view)

Name	Date modified	Type
_delta_log	2/22/2024 3:53:22 PM	Folder
part-00000-d9df3ba7-c505-4eca-a1b9-50ccb15a6398-c000.snappy.parquet	2/22/2024 3:53:22 PM	PARQUET

Task 3: Load data from ADLS Gen 2, apply scalable transformation then load DELTA table

In this task, you'll be working with two files: Sales_File1.csv and Sales_File2.csv, each file contains 1M rows. These files are still stored in an external ADLS Gen 2 account, and the Spark Task will load data in a data frame.



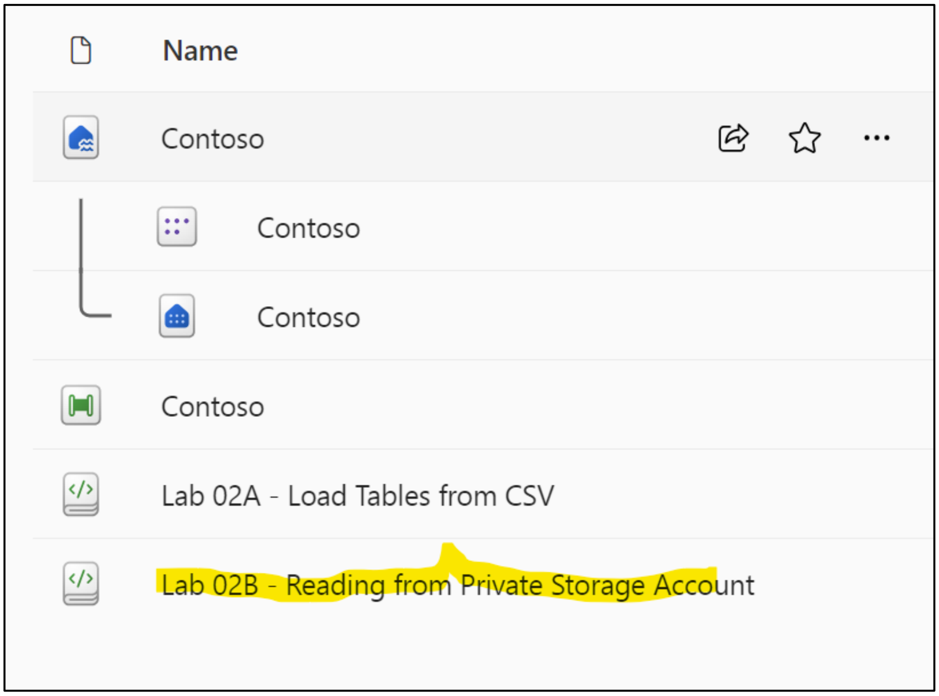
□ Sales_File1.csv

□ Sales_File2.csv

Your task is to sort the files based on a specified set of columns and then perform a merge join (1 row on file 1 to be joined with 1 rows on file 1) using these columns.

Once you've completed the join, you'll need to write the results back to a Delta table in the Managed Lakehouse.

- From the lab workspace, open the Notebook Lab 02B - Reading from Private Storage Account.



Name

Contoso

Contoso

Contoso

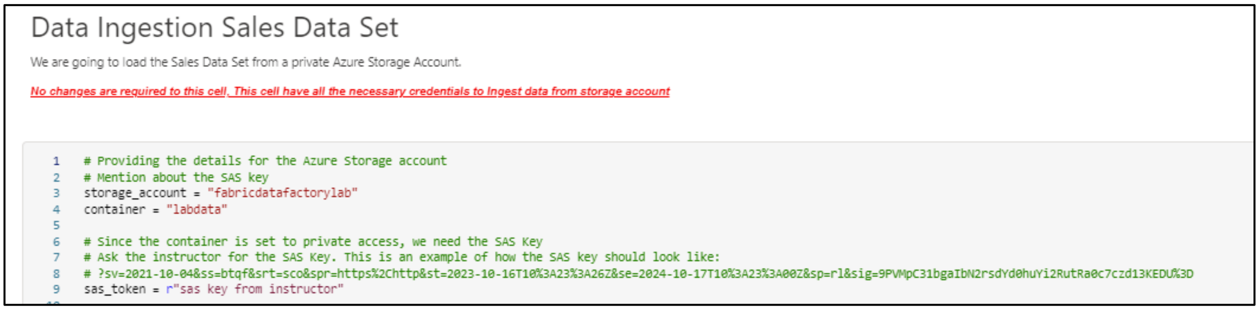
Contoso

Lab 02A - Load Tables from CSV

Lab 02B - Reading from Private Storage Account

- As you did during the previous task, use the Lakehouse selector to define the Contoso Lakehouse as the default one, and remove the wrong one.

- You also need to update the Python script to specify how to connect to the ADLS Gen 2 account (your trainer will share the information) :
  - Line 3 : the storage account name
  - Line 4 : the container
  - Line 9 : the SAS token



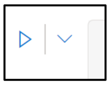
Data Ingestion Sales Data Set

We are going to load the Sales Data Set from a private Azure Storage Account.

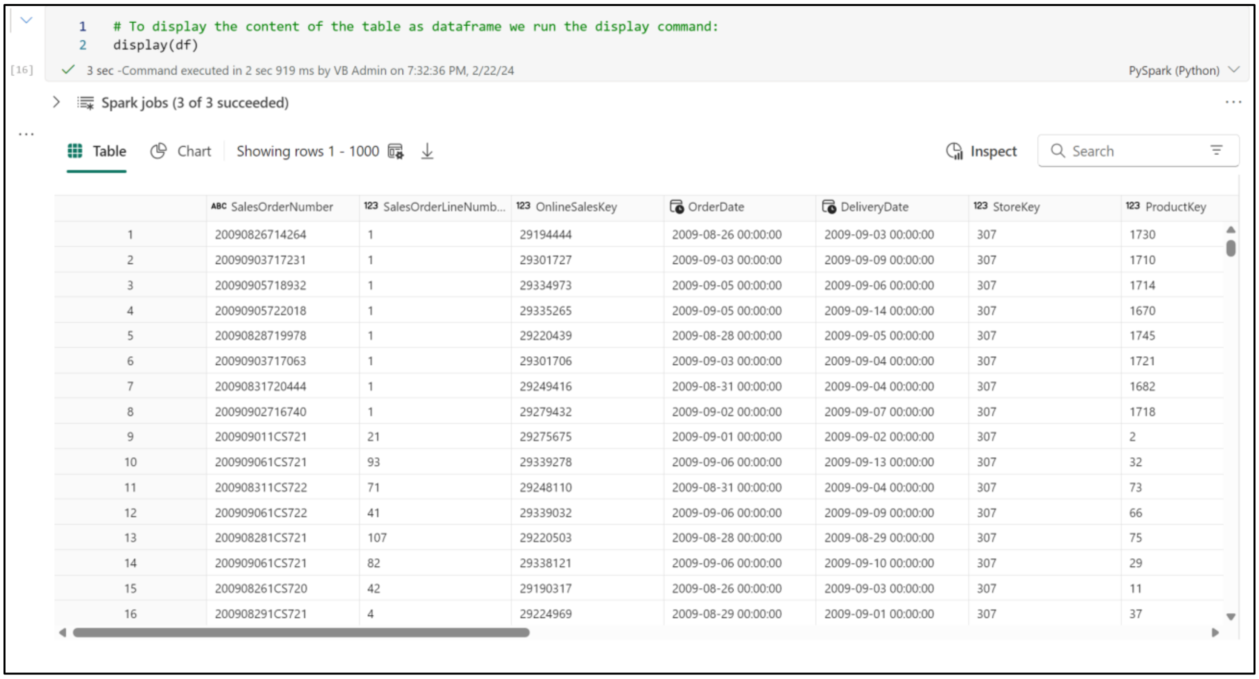
No changes are required to this cell. This cell have all the necessary credentials to Ingest data from storage account

```
1   # Providing the details for the Azure Storage account
2   # Mention about the SAS key
3   storage_account = "fabricdatafactorylab"
4   container = "labdata"
5
6   # Since the container is set to private access, we need the SAS Key
7   # Ask the instructor for the SAS Key. This is an example of how the SAS key should look like:
8   # ?sv=2021-10-04&ss=btqf&srt=sco&spr=https%2Chttp&st=2023-10-16T10%3A23%3A26Z&se=2024-10-17T10%3A23%3A00Z&sp=rl&sig=9PVMpC31bgaIbN2rsdYd0huYi2RutRa0c7czd13KEDU%3D
9   sas_token = r"sas key from instructor"
```

- Once the notebook is configured, execute each cell individually to understand the task performed :

  - Create 2 dataframes from the 2 CSV files (1M rows per file)
  - Show a subset of the tables, the structure and the number of rows
  - Sort each dataframes with the columns SalesOrderNumber and SalesOrderLIneNumber
  - Perform a Merge Join between the 2 files using the sorted columns
  - Drop the Sales table if it already exists in the Lakehouse
  - Load the Dataframe in a new table in the Lakehouse
  - Count the number of rows on the table: 1M
  - Display some rows on the load table



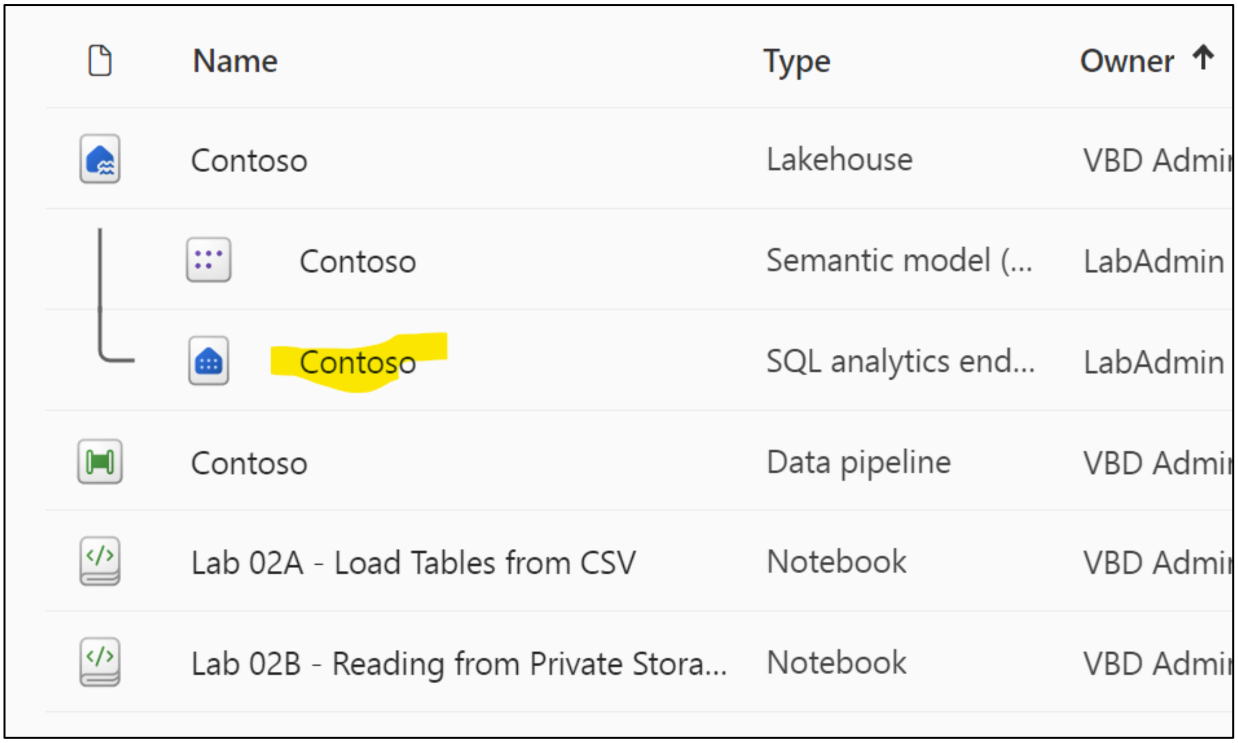
```
# To display the content of the table as dataframe we run the display command:
display(df)
```

[16] ✓ 3 sec -Command executed in 2 sec 919 ms by VB Admin on 7:32:36 PM, 2/22/24 PySpark (Python)

> Spark jobs (3 of 3 succeeded)

	SalesOrderNumber	SalesOrderLineNumb...	OnlineSalesKey	OrderDate	DeliveryDate	StoreKey	ProductKey
1	20090826714264	1	29194444	2009-08-26 00:00:00	2009-09-03 00:00:00	307	1730
2	20090903717231	1	29301727	2009-09-03 00:00:00	2009-09-09 00:00:00	307	1710
3	20090905718932	1	29334973	2009-09-05 00:00:00	2009-09-06 00:00:00	307	1714
4	20090905722018	1	29335265	2009-09-05 00:00:00	2009-09-14 00:00:00	307	1670
5	20090828719978	1	29220439	2009-08-28 00:00:00	2009-09-05 00:00:00	307	1745
6	20090903717063	1	29301706	2009-09-03 00:00:00	2009-09-04 00:00:00	307	1721
7	20090831720444	1	29249416	2009-08-31 00:00:00	2009-09-04 00:00:00	307	1682
8	20090902716740	1	29279432	2009-09-02 00:00:00	2009-09-07 00:00:00	307	1718
9	200909011CS721	21	29275675	2009-09-01 00:00:00	2009-09-02 00:00:00	307	2
10	200909061CS721	93	29339278	2009-09-06 00:00:00	2009-09-13 00:00:00	307	32
11	200908311CS722	71	29248110	2009-08-31 00:00:00	2009-09-04 00:00:00	307	73
12	200909061CS722	41	29339032	2009-09-06 00:00:00	2009-09-09 00:00:00	307	66
13	200908281CS721	107	29220503	2009-08-28 00:00:00	2009-08-29 00:00:00	307	75
14	200909061CS721	82	29338121	2009-09-06 00:00:00	2009-09-10 00:00:00	307	29
15	200908261CS720	42	29190317	2009-08-26 00:00:00	2009-09-03 00:00:00	307	11
16	200908291CS721	4	29224969	2009-08-29 00:00:00	2009-09-01 00:00:00	307	37

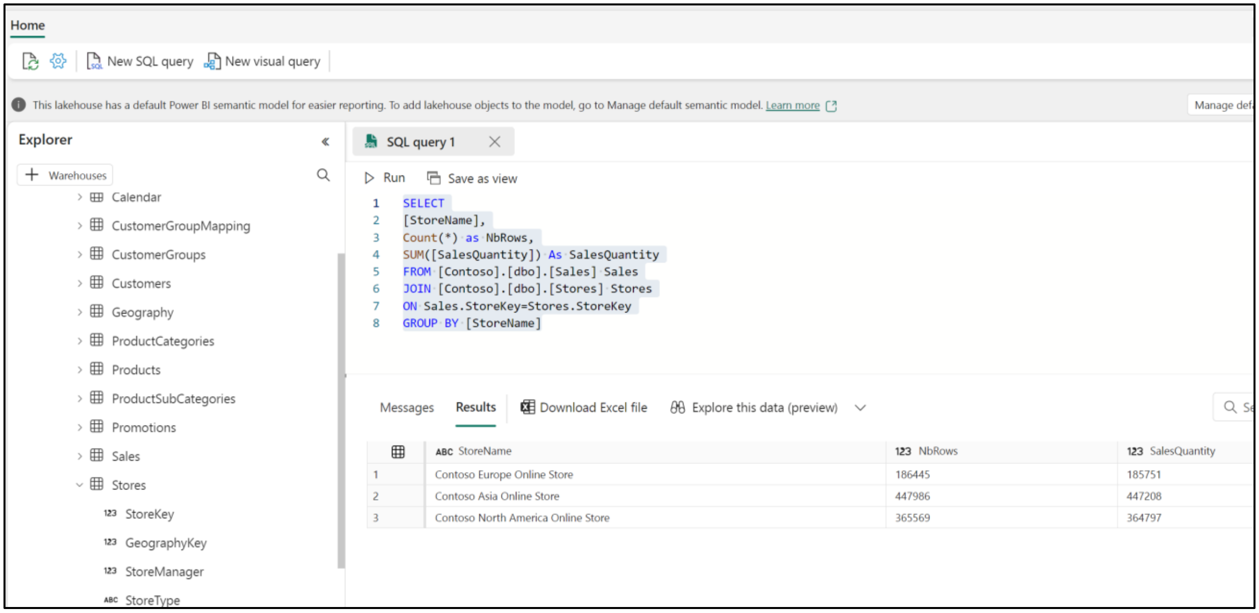
- From the Lab workspace, open the SQL analytics endpoint of the Contoso Lakehouse



Name	Type	Owner ↑
Contoso	Lakehouse	VBD Admi
Contoso	Semantic model (...	LabAdmin
Contoso	SQL analytics end...	LabAdmin
Contoso	Data pipeline	VBD Admi
Lab 02A - Load Tables from CSV	Notebook	VBD Admi
Lab 02B - Reading from Private Stora...	Notebook	VBD Admi

- Create a new SQL query using the following snippet to display the number of rows on the fact tables, and the aggregated quantity per store ;

```
SELECT
[StoreName],Count(*) as NbRows,SUM([SalesQuantity]) As SalesQuantity
FROM [Contoso].[dbo].[Sales] Sales
JOIN [Contoso].[dbo].[Stores] Stores ON Sales.StoreKey=Stores.StoreKey
GROUP BY [StoreName]
```

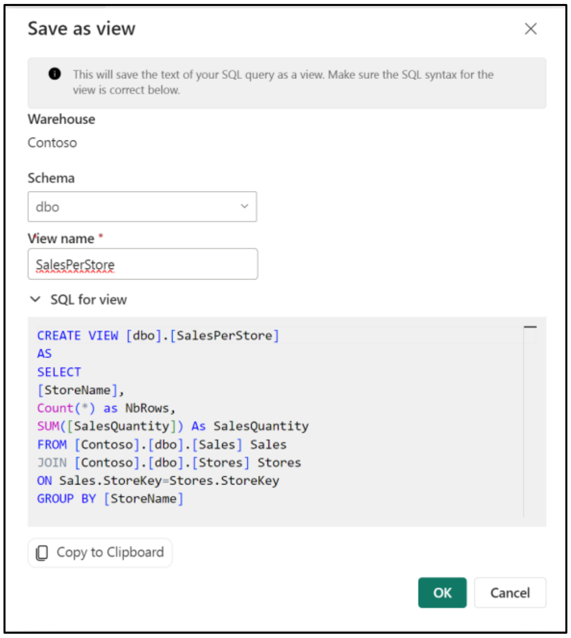


This lakehouse has a default Power BI semantic model for easier reporting. To add lakehouse objects to the model, go to Manage default semantic model. Learn more

```
SELECT
[StoreName],
Count(*) as NbRows,
SUM([SalesQuantity]) As SalesQuantity
FROM [Contoso].[dbo].[Sales] Sales
JOIN [Contoso].[dbo].[Stores] Stores
ON Sales.StoreKey=Stores.StoreKey
GROUP BY [StoreName]
```

	StoreName	NbRows	SalesQuantity
1	Contoso Europe Online Store	186445	185751
2	Contoso Asia Online Store	447986	447208
3	Contoso North America Online Store	365569	364797

- Use the Save as View option to keep the query



This will save the text of your SQL query as a view. Make sure the SQL syntax for the view is correct below.

Warehouse: Contoso

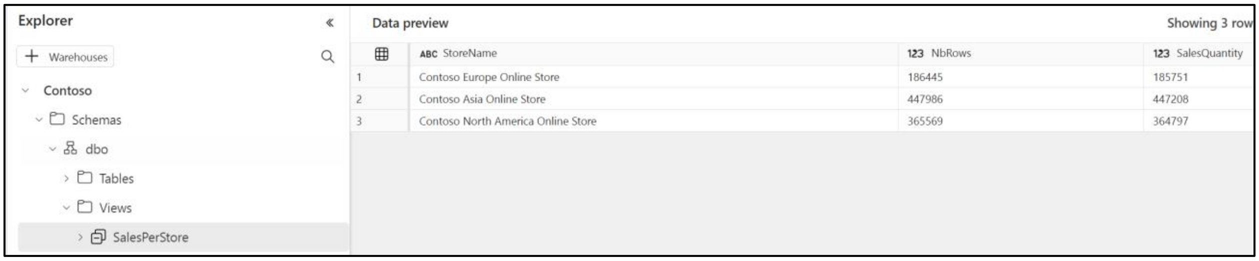
Schema: dbo

View name *: SalesPerStore

SQL for view

```
CREATE VIEW [dbo].[SalesPerStore]
AS
SELECT
[StoreName],
Count(*) as NbRows,
SUM([SalesQuantity]) As SalesQuantity
FROM [Contoso].[dbo].[Sales] Sales
JOIN [Contoso].[dbo].[Stores] Stores
ON Sales.StoreKey=Stores.StoreKey
GROUP BY [StoreName]
```

- Make sure the newly created SQL view works – it will be used to control data ingestion in the next labs.



	StoreName	NbRows	SalesQuantity
1	Contoso Europe Online Store	186445	185751
2	Contoso Asia Online Store	447986	447208
3	Contoso North America Online Store	365569	364797