



MICROSOFT FABRIC

Spark

Summary

To use Microsoft Fabric to manipulate data with Spark

Jerry Lau

elarning4jerry@gmail.com

Table of contents

Introduction	2
Upload files	3
Create a notebook.....	4
Create a DataFrame	5
Explore data in a DataFrame	9
Use Spark to transform data files – PySpark.....	12
Use Spark to transform data files – SQL.....	16
Visualize data with Spark.....	18
Clean up resources	20

Introduction

In this demo:

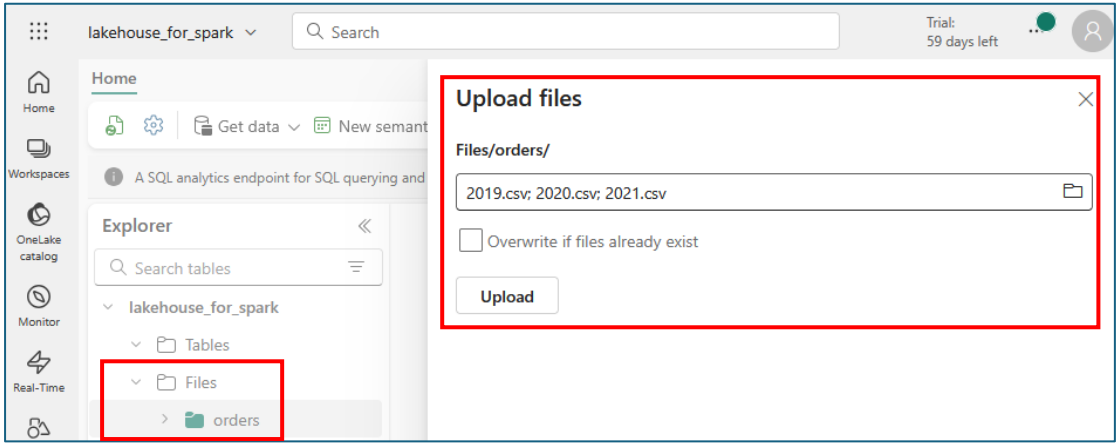
1. A new workspace will be created.
2. A lakehouse will be created to store files.
3. A file will be uploaded to lakehouse and transformed into table.
4. The data will be analyzed and aggregated by SQL query and visual query.
5. The data will be visualized in Power BI.

Assumption: you have initialized your Fabric environment like workspace and Lakehouse.

Reference: <https://microsoftlearning.github.io/mslearn-fabric/Instructions/Labs/02-analyze-spark.html>

Upload files

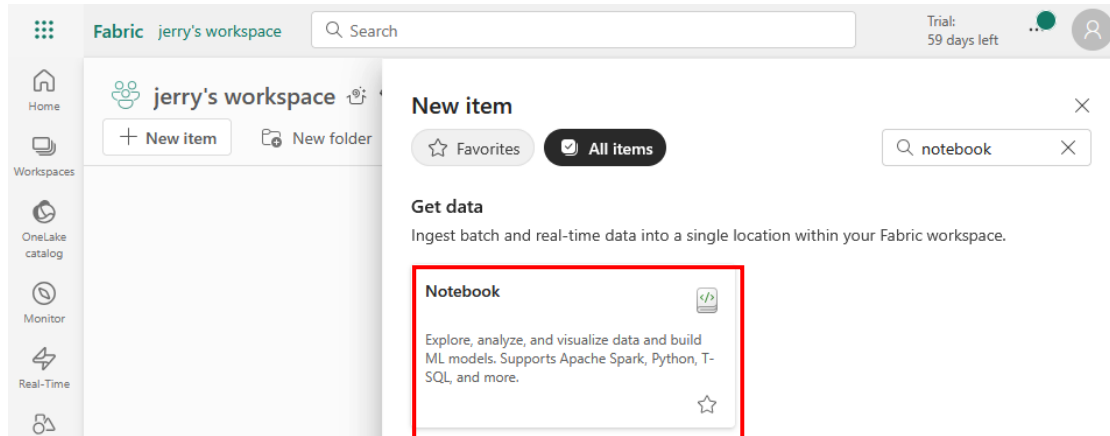
Create a subfolder “order” under “Files” and upload a file to it



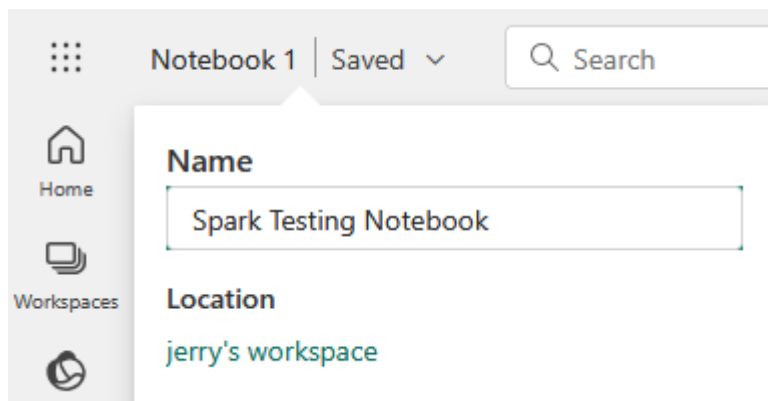
Current uploads		Dismiss: Completed All		
File Name	Lakehouse Name			
2019.csv	lakehouse_for_spark	✓	120 KB / 120 KB	🗑️
2020.csv	lakehouse_for_spark	✓	279 KB / 279 KB	🗑️
2021.csv	lakehouse_for_spark	✓	2 MB / 2 MB	🗑️

Create a notebook

Go back to workspace, click “+ New item”, find “Notebook” and click it.

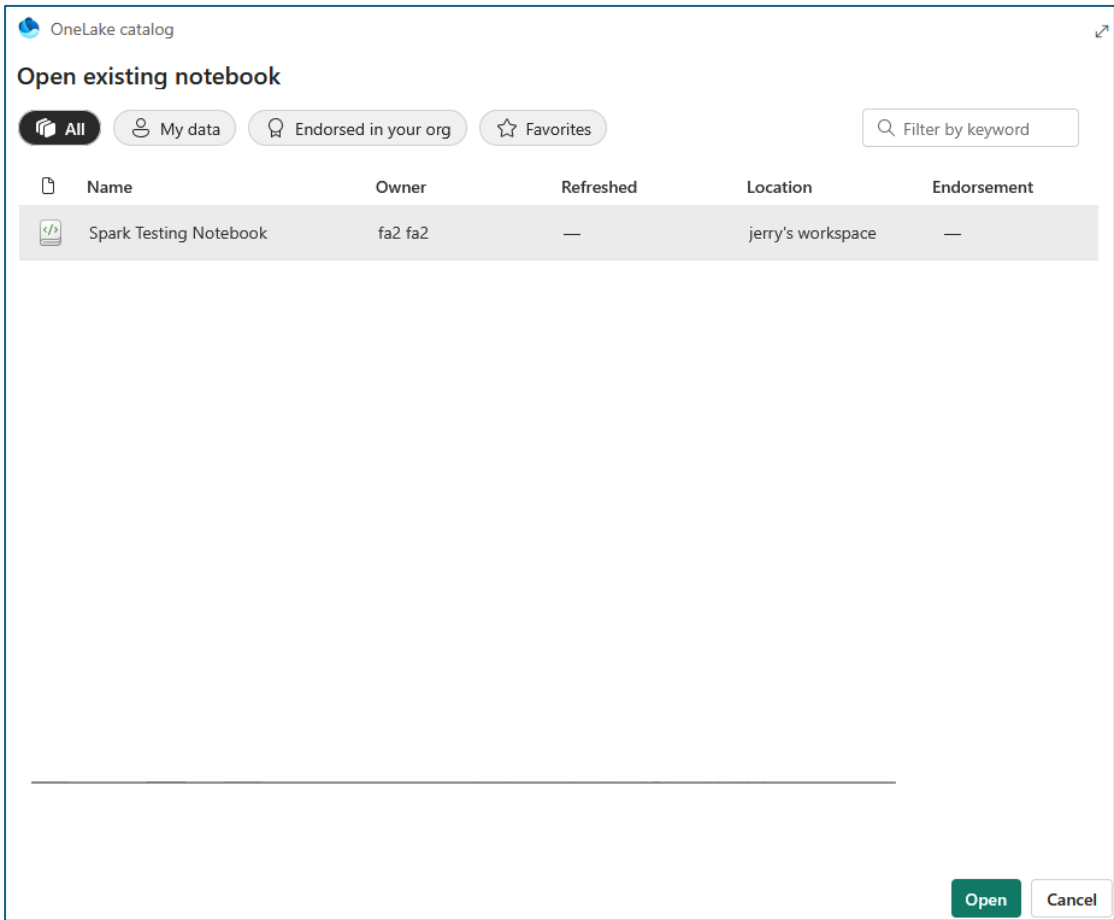
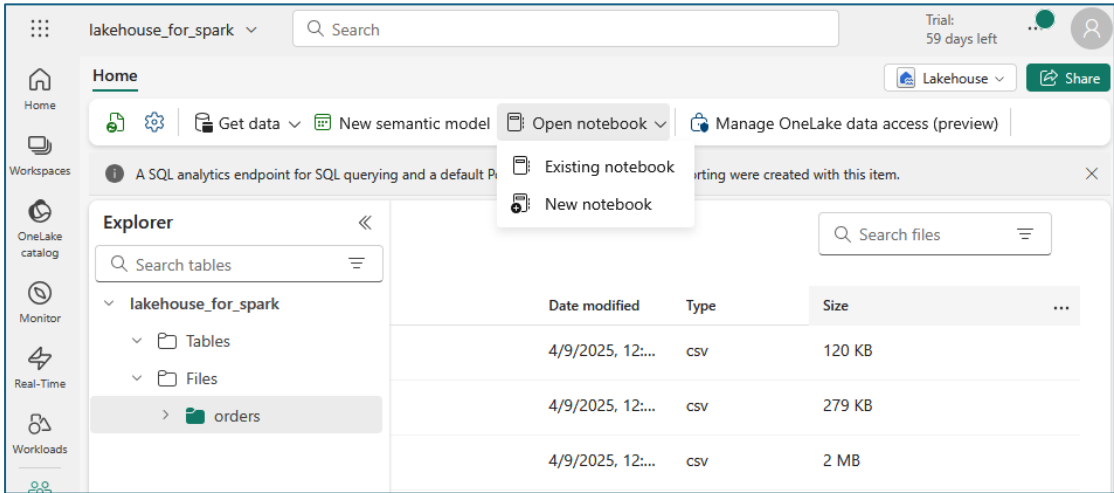


At top left corner, click the auto-generated notebook name (Notebook 1) to rename it.

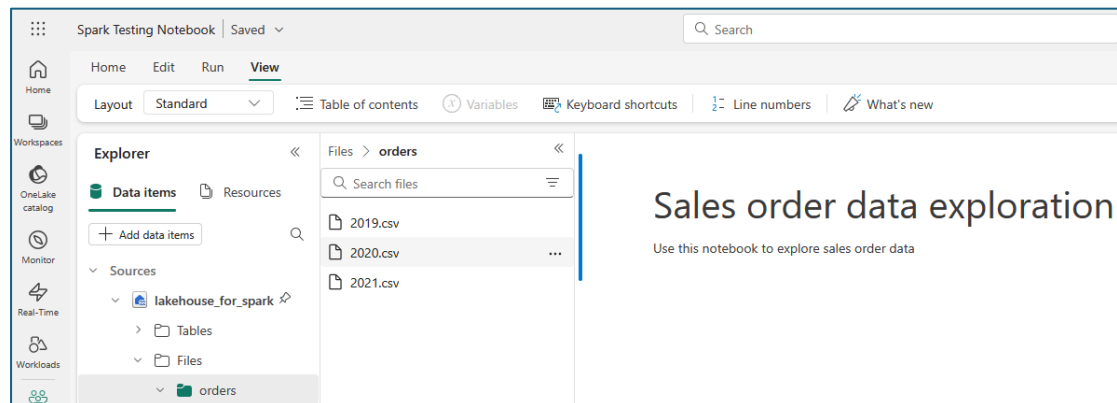


Create a DataFrame

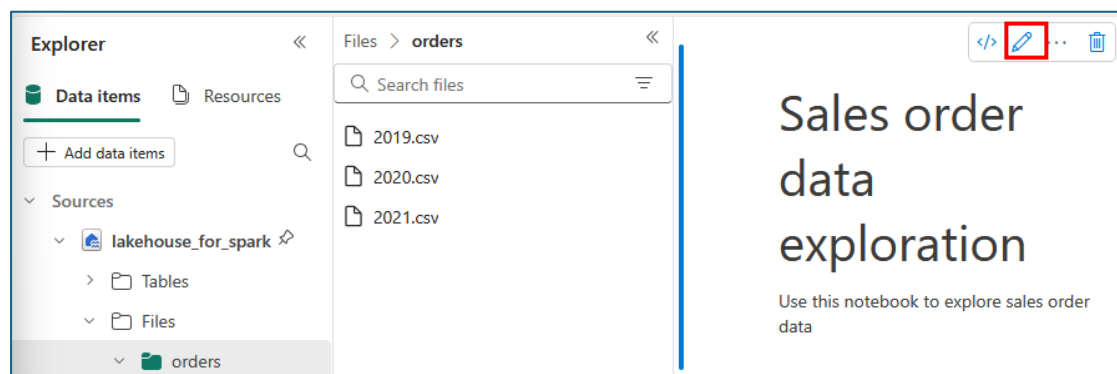
Go back to the uploaded file, at the top menu, click “Open notebook” -> “Open existing notebook”, choose the one created earlier



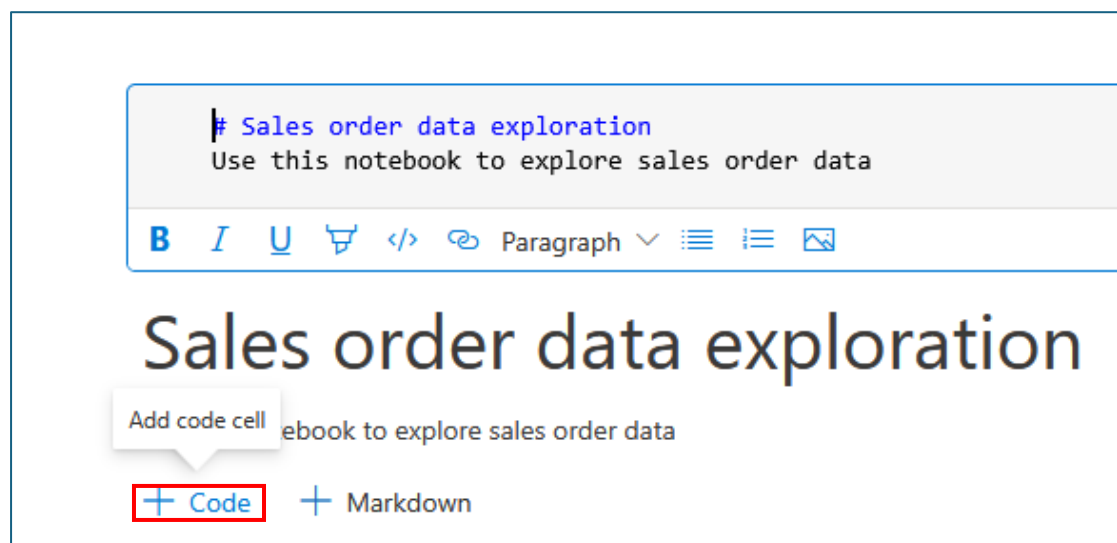
You should be able to see the files uploaded earlier on left hand side (try to expand the folder or side bar if not)



At top right corner of the notebook, click “Edit” (pencil icon) to switch to edit mode.



Add a code block at the bottom (hover over if not seeing it).



Add a snippet to read data from order folder. Notice that PySpark is the current

language selected. Put option header:false because the file does not contain a header

Click “Run” (triangle icon) on the left to execute the code (that will take some time)

```
1 df = spark.read.format("csv").option("header", "false").load("Files/orders/2019.csv")
2 # df now is a Spark DataFrame containing CSV data from "Files/orders/2019.csv".
3 display(df)
```

[1] PySpark (Python) ▾

14 sec - Session ready in 10 sec 241 ms. Command executed in 3 sec 966 ms by fa2 on 1:59:42 PM, 4/09/25 PySpark (Python) ▾

> Spark jobs (1 of 1 succeeded) Resources

Table + New chart Data Wrangler 9 columns, 1000 rows ▾

Table view Download ▾ Search

	ABC_c0	ABC_c1	ABC_c2	ABC_c3	ABC_c4	ABC_c5	ABC_c6	ABC_c7
1	SO43701	1	2019-07-01	Christy Zhu	christy12@...	Mountain...	1	3399
2	SO43704	1	2019-07-01	Julio Ruiz	julio1@adv...	Mountain...	1	3374
3	SO43705	1	2019-07-01	Curtis Lu	curtis9@ad...	Mountain...	1	3399
4	SO43700	1	2019-07-01	Ruben Pras...	ruben10@...	Road-650 ...	1	699.0
5	SO43703	1	2019-07-01	Albert Alva...	albert7@a...	Road-150 ...	1	3578

Inspect

To assign header to the dataframe, create a StructType. Revise the code accordingly, then re-run the code.

```
1 from pyspark.sql.types import *
2
3 orderSchema = StructType([
4     StructField("SalesOrderNumber", StringType()),
5     StructField("SalesOrderLineNumber", IntegerType()),
6     StructField("OrderDate", DateType()),
7     StructField("CustomerName", StringType()),
8     StructField("Email", StringType()),
9     StructField("Item", StringType()),
10    StructField("Quantity", IntegerType()),
11    StructField("UnitPrice", FloatType()),
12    StructField("Tax", FloatType())
13 ])
14
15 df = spark.read.format("csv").schema(orderSchema).load("Files/orders/2019.csv")
16
17 display(df)
```


Table

+ New chart

Data Wrangler

9 columns, 1000 rows

Table view

Download

Search

	ABC SalesOrderNumber	123 SalesOrderLineNumber	OrderDate	ABC CustomerName	ABC Email
1	SO43701	1	2019-07-01	Christy Zhu	christy12@...
2	SO43704	1	2019-07-01	Julio Ruiz	julio1@adv...
3	SO43705	1	2019-07-01	Curtis Lu	curtis9@ad...
4	SO43700	1	2019-07-01	Ruben Prasad	ruben10@...
5	SO43703	1	2019-07-01	Albert Alvarez	albert7@a...

Further update the load statement so it loads all the csv files from the folder.

```
df = spark.read.format("csv").schema(orderSchema).load("Files/orders/*.csv")
```

Explore data in a DataFrame

Create a new code block and insert the snippet as below, and run the code.

```
1 customers = df['CustomerName', 'Email']
2
3 print(customers.count())
4 print(customers.distinct().count())
5
6 display(customers.distinct())
```

PySpark (Python) ▾

```
1 customers = df['CustomerName', 'Email']
2
3 print(customers.count())
4 print(customers.distinct().count())
5
6 display(customers.distinct())
```

✓ 4 sec - Command executed in 4 sec 623 ms by fa2 on 2:16:38 PM, 4/09/25

PySpark (Python) ▾

> Spark jobs (7 of 7 succeeded) Resources Log ...

1201

1201

Table

+ New chart

2 columns, 1000 rows

Table view

Download

Search



Inspect

	ABC CustomerName	ABC Email	
1	Jonathon Gutierrez	jonathon8...	
2	Blake Butler	blake62@a...	
3	Melissa Perry	melissa2@...	
4	Hailey James	hailey17@...	
5	Jasmine West	jasmine37...	
6	Teresa Ruiz	teresa3@a...	

Revise with a where clause and rerun

```
1 customers = df["CustomerName", "Email"].where(df['Item']=='Road-150 Red, 52')
2 #print(customers.count())
3 #print(customers.distinct().count())
4 display(customers.distinct())
```

✓ <1 sec - Command executed in 768 ms by fa2 on 2:30:32 PM, 4/09/25

>  Spark jobs (2 of 2 succeeded)  Resources




 Table  New chart

Table view

	ABC CustomerName	ABC Email	
1	Melissa Perry	melissa2@...	
2	Natalie Moore	natalie75@...	
3	Levi Sai	levi5@adv...	
4	Calvin Deng	calvin1@a...	
5	Wyatt Martinez	wyatt18@a...	

Create a new code block to do SQL-like commands e.g. aggregation and sorting

```

1  from pyspark.sql.functions import *
2
3  yearlySales = df.select(
4      year(col("OrderDate")).alias("Year"),
5      month(col("OrderDate")).alias("Month")
6  ).groupBy("Year", "Month").count().orderBy("Year", "Month")
7
8  display(yearlySales)

```

✓ <1 sec - Command executed in 798 ms by fa2 on 2:38:20 PM, 4/09/25

>  Spark jobs (2 of 2 succeeded)  Resources  Log




 Table  New chart

Table view

	123 Year	123 Month	12L count	
1	2019	7	289	
2	2019	8	159	
3	2019	9	161	
4	2019	10	174	
5	2019	11	230	
6	2019	12	188	
7	2020	1	193	
8	2020	2	177	
9	2020	3	219	

Use Spark to transform data files – PySpark

Compute new columns with “withColumn” method, and store in a new dataframe.

```
1  from pyspark.sql.functions import *
2
3  # Create Year and Month columns
4  transformed_df = df.withColumn(
5      "Year", year(col("OrderDate"))
6  ).withColumn(
7      "Month", month(col("OrderDate"))
8  )
9
10 # Create the new FirstName and LastName fields
11 transformed_df = transformed_df.withColumn(
12     "FirstName", split(col("CustomerName"), " ").getItem(0)
13 ).withColumn(
14     "LastName", split(col("CustomerName"), " ").getItem(1)
15 )
16
17 # Filter and reorder columns
18 transformed_df = transformed_df[["SalesOrderNumber", "SalesOrderLineNumber",
19     "OrderDate", "Year",
20     "Month", "FirstName",
21     "LastName", "Email",
22     "Item", "Quantity",
23     "UnitPrice", "Tax"]]
24
25 # Display the first five orders
26 display(transformed_df.limit(5))
```

✓ <1 sec - Command executed in 738 ms by fa2 on 2:45:36 PM, 4/09/25 PySpark (Python) ▾

> ⚙ Spark jobs (1 of 1 succeeded) 📄 Resources 📄 Log ...

Table + New chart ⚙ 12 columns, 5 rows ▾ ...

Table view ⬇ Download ▾ 🔍 Search

	ABC SalesOrderNumber	123 SalesOrderLineNumber	📅 OrderDate	123 Year	123 Month	ABC FirstN
1	SO49171	1	2021-01-01	2021	1	Mariah
2	SO49172	1	2021-01-01	2021	1	Brian
3	SO49173	1	2021-01-01	2021	1	Linda

Inspect

More DataFrame methods:

<https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/dataframe.html>

You can write the DataFrame back to Lakehouse file

```
1  transformed_df.write.mode("overwrite").parquet('Files/transformed_data/orders')
2  print ("Transformed data saved!")
```

✓ 4 sec - Command executed in 4 sec 539 ms by fa2 on 2:49:55 PM, 4/09/25 PySpark (Python) ▾

> ⚙ Spark jobs (1 of 1 succeeded) 📄 Resources 📄 Log ...

Check after running the code

lakehouse_for_spark

Home

Get data New semantic model Open notebook Manage OneLake data access (preview)

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

Explorer

lakehouse_for_spark

- Tables
- Files
 - orders
 - transformed_data
 - orders

Files > transformed_data > orders

Name	Date modified	Type	Size
_SUCCESS	4/9/2025, 2:49:54 PM		0 B
part-00000-ffac877e-fc69-484c-8ba8-3c25ba45422a-c000.snappy.parquet	4/9/2025, 2:49:53 PM	parquet	396 KB
part-00001-ffac877e-fc69-484c-8ba8-3c25ba45422a-c000.snappy.parquet	4/9/2025, 2:49:53 PM	parquet	68 KB
part-00002-ffac877e-fc69-484c-8ba8-3c25ba45422a-c000.snappy.parquet	4/9/2025, 2:49:53 PM	parquet	34 KB

Read the saved data from PySpark

```

1 orders_df = spark.read.format("parquet").load("Files/transformed_data/orders")
2 display(orders_df)

```

✓ 13 sec - Session ready in 9 sec 346 ms. Command executed in 3 sec 619 ms by fa2 on 3:18:47 PM, 4/09/25 PySpark (Python)

> Spark jobs (2 of 2 succeeded) Resources

Table + New chart Data Wrangler 12 columns, 1000 rows

Table view Download Search

	ABC SalesOrderNumber	123 SalesOrderLineNumber	OrderDate	123 Year	123 Month	ABC Fi
1	SO49171	1	2021-01-01	2021	1	Marj
2	SO49172	1	2021-01-01	2021	1	Brian

Inspect

You can also explicitly partition the data to save

```

1 orders_df.write.partitionBy("Year", "Month")\
2   .mode("overwrite").parquet("Files/partitioned_data")
3   print ("Transformed data saved!")

```

✓ 4 sec - Command executed in 4 sec 598 ms by fa2 on 3:20:31 PM, 4/09/25

Explorer

Search tables

lakehouse_for_spark

Tables

Files

orders

partitioned_data

Year=2019

Month=10

Month=11

Month=12

Month=7

Month=8

Month=9

Year=2020

Year=2021

Files > partitioned_data

Name

Year=2019

Year=2020

Year=2021

_SUCCESS

Files > partitioned_data > Year=2019 > **Month=12**

Name



part-00002-59be4a37-4bd8-4317-9bfc-a6c43dc24f0e.c000.snappy.parquet

Again you can retrieve the saved data in PySpark

```
1 orders_2021_df = spark.read.format("parquet")\  
2   .load("Files/partitioned_data/Year=2021/Month=*")\  
3   display(orders_2021_df)
```


✓ 9 sec - Command executed in 9 sec 540 ms by fa2 on 3:25:18 PM, 4/09/25

PySpark (Python) ▾


>  Spark jobs (2 of 2 succeeded)  Resources  Log ...

 Table

+ New chart

 Data Wrangler

≡

 10 columns, 1000 rows ▾



...

Table view

↓ Download ▾

 Search

«

	ABC SalesOrderNumber	123 SalesOrderLineNumber	 OrderDate	ABC FirstName	ABC LastName
1	SO59195	1	2021-11-01	Alexia	Hayes
2	SO59195	2	2021-11-01	Alexia	Hayes
3	SO59196	1	2021-11-01	Anthony	Garcia

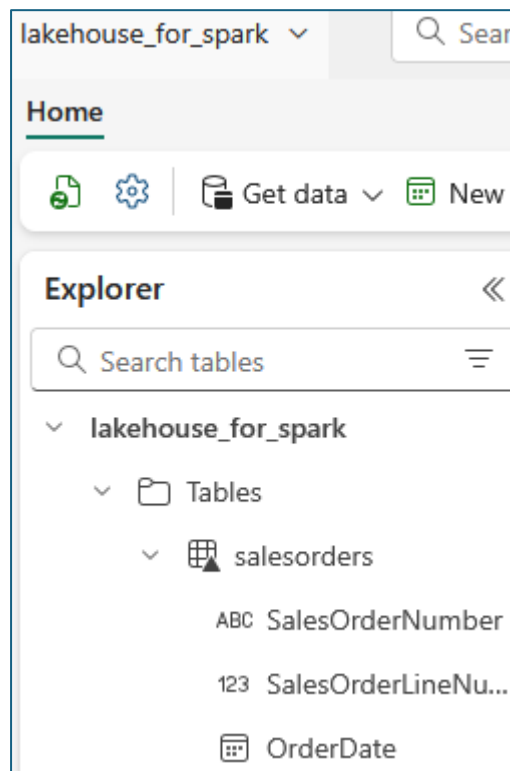
Inspect

Use Spark to transform data files – SQL

Save the DataFrame df to table under lakehouse

```
1 # Create a new table
2 df.write.format("delta").saveAsTable("salesorders")
3
4 # Get the table description
5 spark.sql("DESCRIBE EXTENDED salesorders").show(truncate=False)
```

✓ 20 sec - Command executed in 20 sec 114 ms by fa2 on 3:30:52 PM, 4/09/25 PySpark (Python) ▾



Retrieve Lakehouse table data (still using PySpark)

```
1 df = spark.sql("SELECT * FROM lakehouse_for_spark.salesorders LIMIT 1000")
2 display(df)
```

✓ 9 sec - Command executed in 9 sec 642 ms by fa2 on 3:39:41 PM, 4/09/25 PySpark (Python) ▾

> Spark jobs (5 of 5 succeeded) Resources Log

Table + New chart Data Wrangler 9 columns, 1000 rows ▾

Table view Download ▾ Search

	ABC SalesOrderNumber	123 SalesOrderLineNumber	OrderDate	ABC CustomerName	ABC Email
1	SO49171	1	2021-01-01	Mariah Foster	mariah21...
2	SO49172	1	2021-01-01	Brian Howard	brian23@a...
3	SO49173	1	2021-01-01	Linda Alvarez	linda19@a...

Inspect

Now switch to use Spark SQL engine to retrieve data

1

2

3

4

5

6

%%sql

SELECT YEAR(OrderDate) AS OrderYear,

SUM((UnitPrice * Quantity) + Tax) AS GrossRevenue

FROM salesorders

GROUP BY YEAR(OrderDate)

ORDER BY OrderYear

✓ 2 sec - Command executed in 2 sec 175 ms by fa2 on 3:40:42 PM, 4/09/25

Spark SQL

>

Spark jobs (3 of 3 succeeded)

Resources

Log

...

Table

+ New chart

2 columns, 3 rows

...

Table view

Download

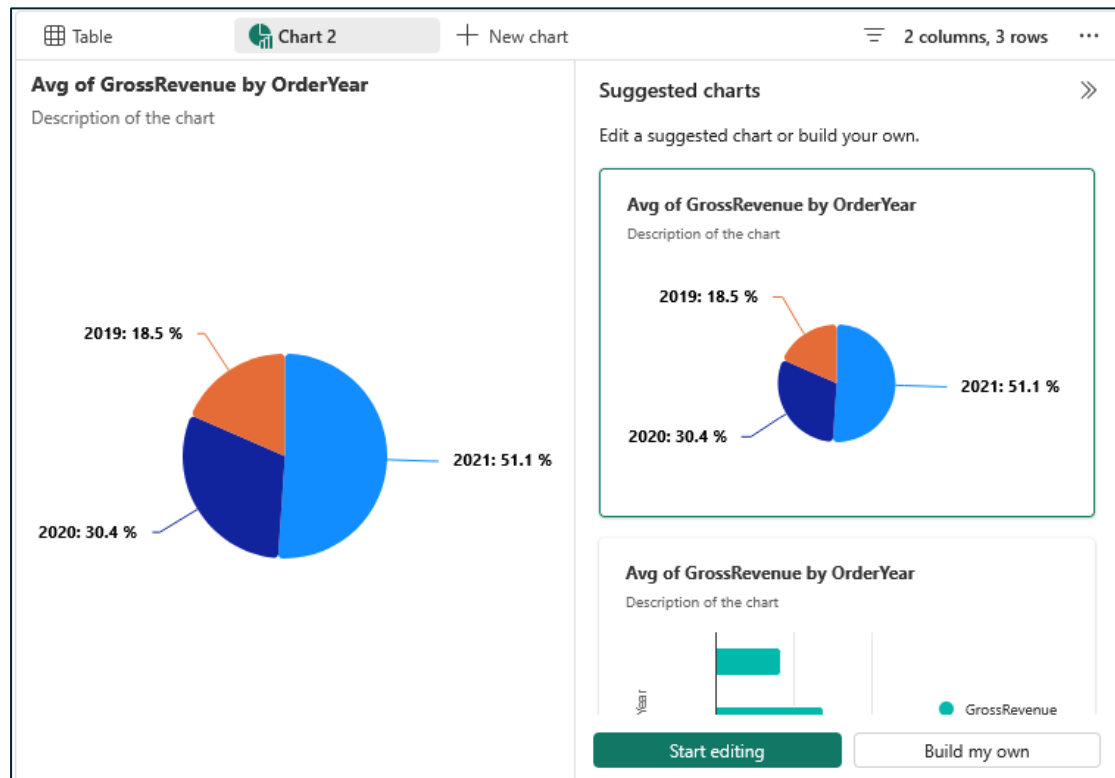
Search

Inspect

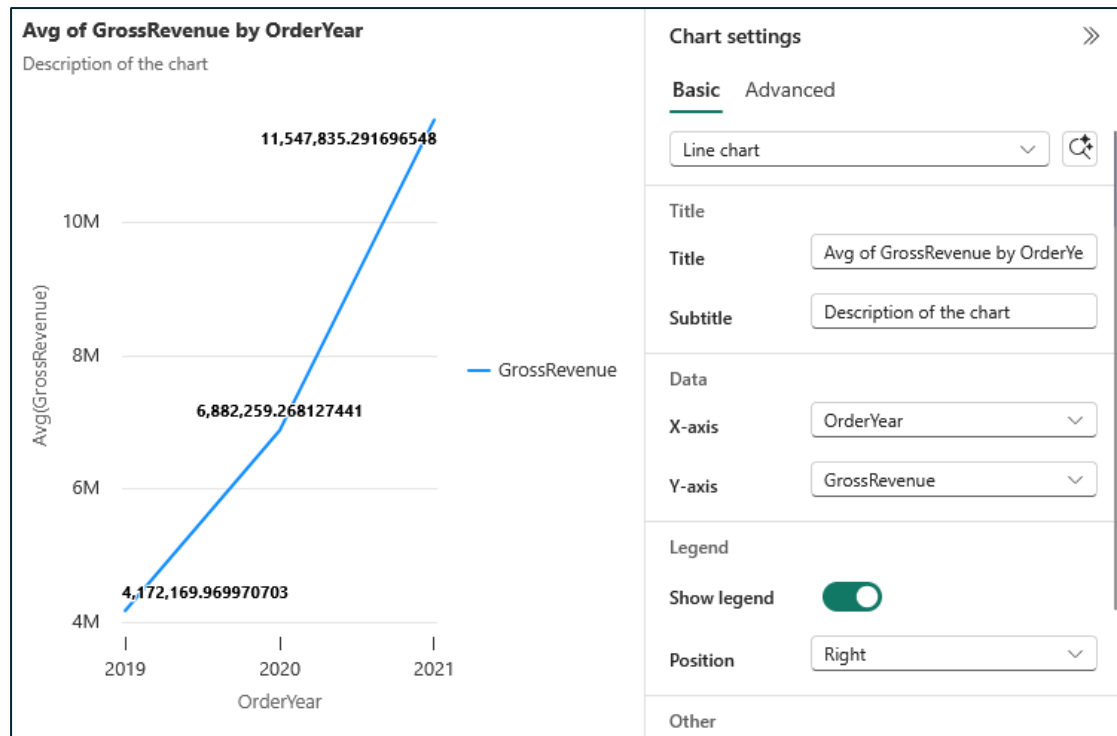
	123 OrderYear	12 GrossRevenue	
1	2019	4172169.969970703	
2	2020	6882259.268127441	
3	2021	11547835.2916965...	

Visualize data with Spark

To continue with previous result, in the result pane, click “+ New chart” to visualize the data



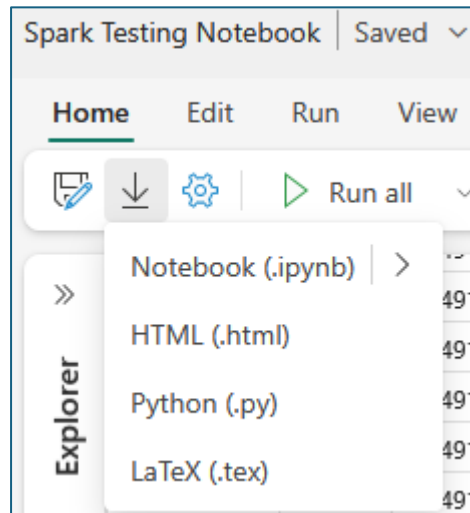
A suggested chart is created. At bottom right corner, click “Start editing” to format if needed e.g. changing to a line chart



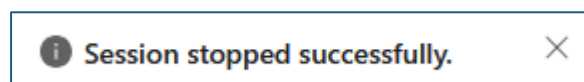
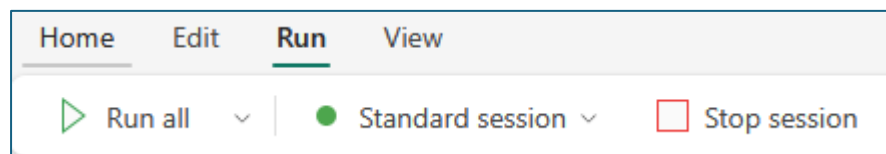
If more advanced or flexible chart requirements are needed, use Python matplotlib.pyplot and seaborn to build more customized chart.

Clean up resources

Before cleaning up the resources, I downloaded the Notebook for reference and backup.



Now, on the top menu, switch to “Run” and click “Stop session”



This is to release the resource and avoid extra cost consumption.

If you need to clean up the workspace as well, go to “Workspace settings” of your created workspace. Click “Remove this workspace”, click “Delete” to confirm.