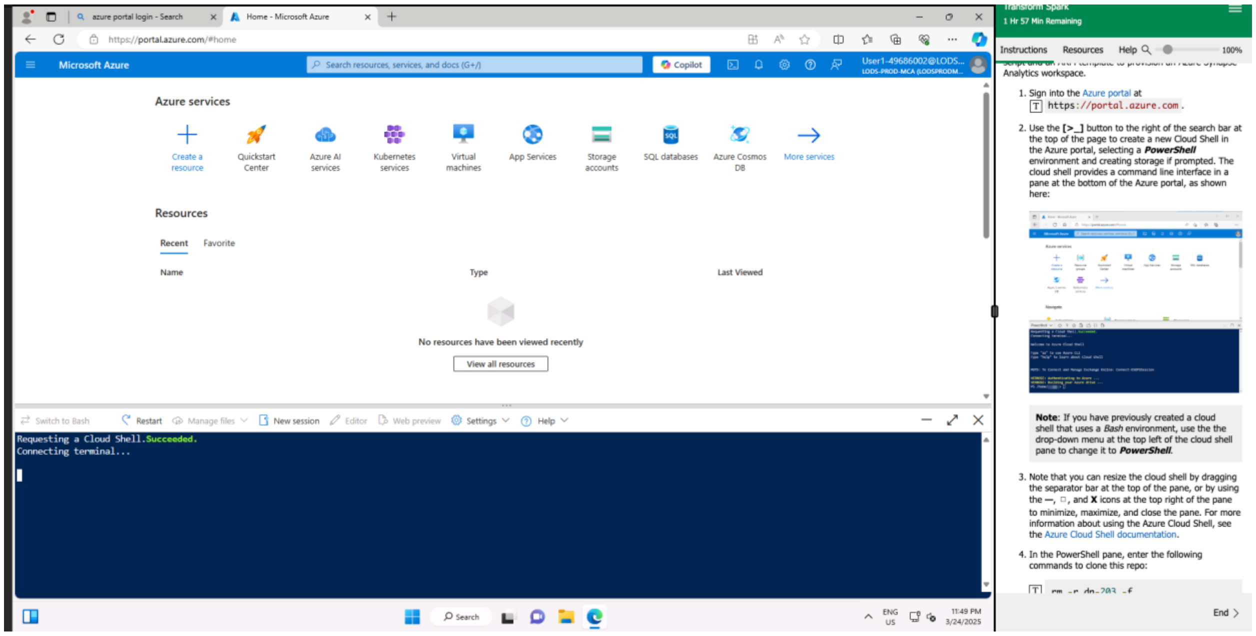
Lab 6 Transform Spark
BUAN 6390.001 – Analytics Practicum

Name: Swastik Bhatnagar
Net ID: Sxb220210

Connecting to Powershell:



```
Requesting a Cloud Shell.Succeeded.
Connecting terminal...
```

Transform Spark
1 Hr 57 Min Remaining

Analytics workspace.

1. Sign into the Azure portal at `https://portal.azure.com`.
2. Use the **[>_]** button to the right of the search bar at the top of the page to create a new Cloud Shell in the Azure portal, selecting a ***PowerShell*** environment and creating storage if prompted. The cloud shell provides a command line interface in a pane at the bottom of the Azure portal, as shown here:

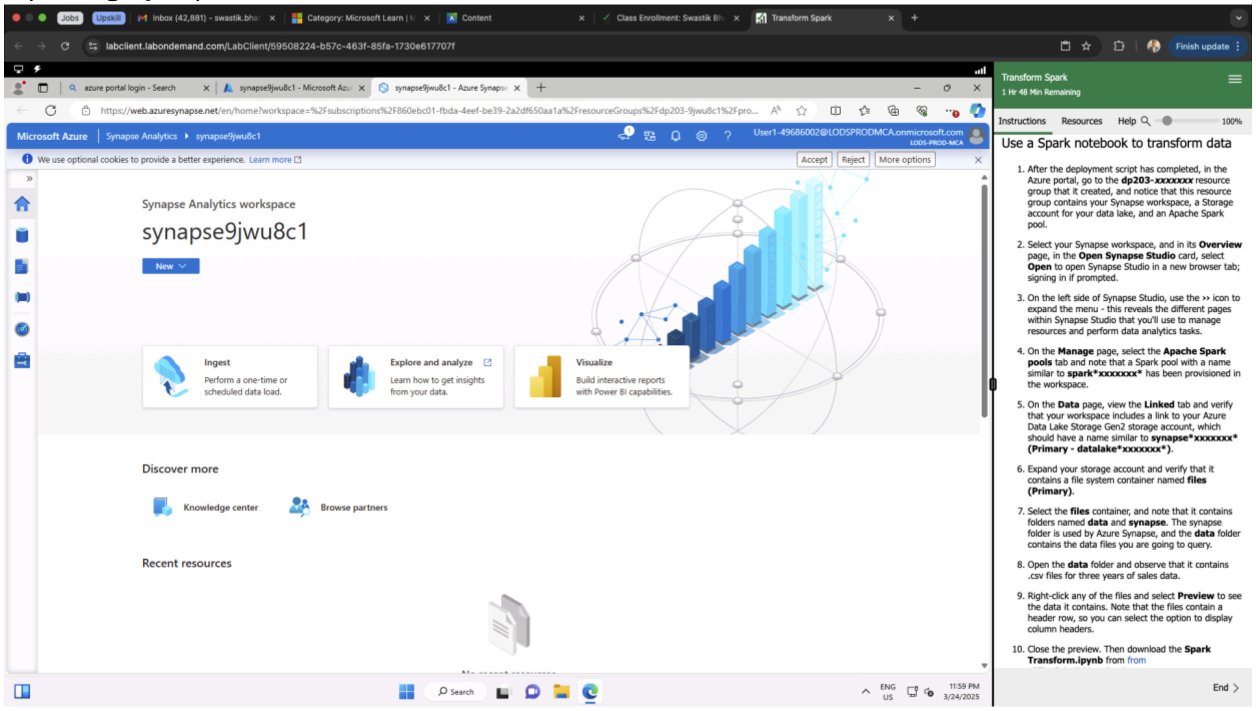
   Note: If you have previously created a cloud shell that uses a *Bash* environment, use the the drop-down menu at the top left of the cloud shell pane to change it to ***PowerShell***.

3. Note that you can resize the cloud shell by dragging the separator bar at the top of the pane, or by using the —, ◻, and **X** icons at the top right of the pane to minimize, maximize, and close the pane. For more information about using the Azure Cloud Shell, see the Azure Cloud Shell documentation.
4. In the PowerShell pane, enter the following commands to clone this repo:

```
rm -r dp-203 -f
```

Using Spark notebook to transform data:

Opening Synapse Studio:



Synapse Analytics workspace

synapse9jwu8c1

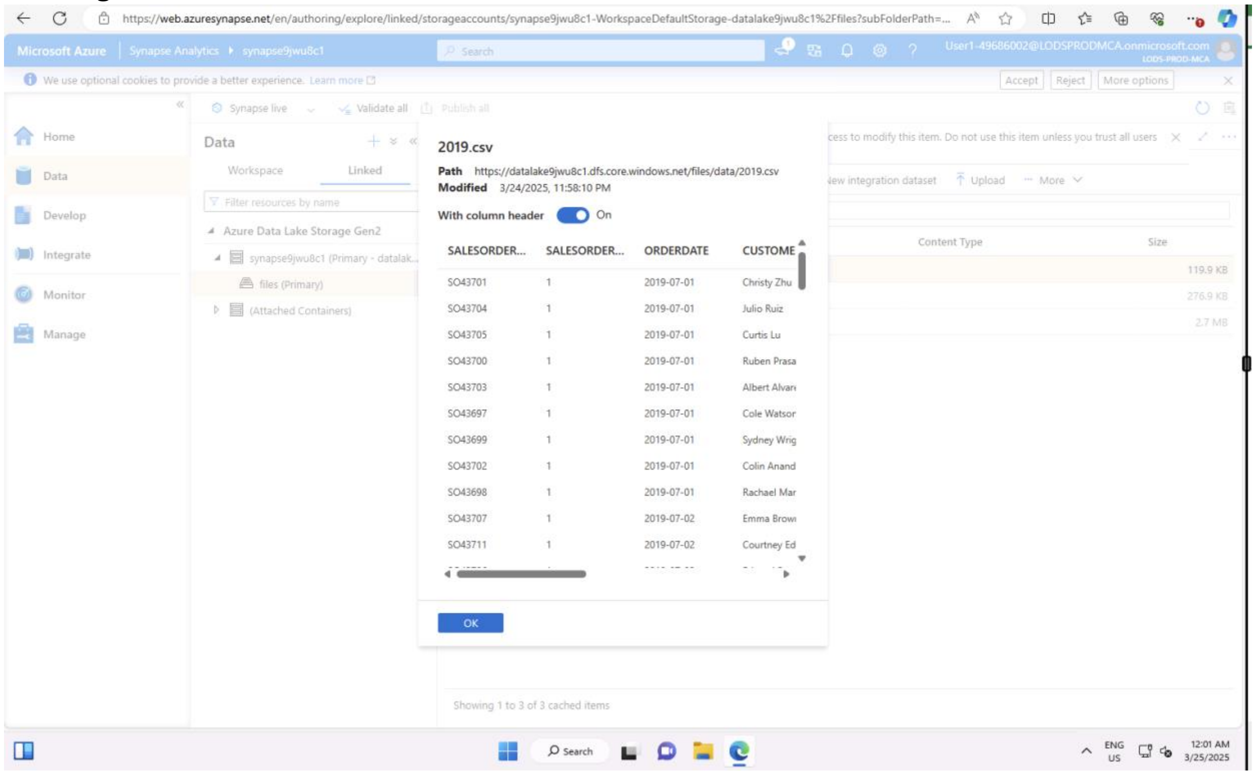
Transform Spark

1 Hr 48 Min Remaining

Use a Spark notebook to transform data

1. After the deployment script has completed, in the Azure portal, go to the **dp203-*xxxxxxx*** resource group that it created, and notice that this resource group contains your Synapse workspace, a Storage account for your data lake, and an Apache Spark pool.
2. Select your Synapse workspace, and in its **Overview** page, in the **Open Synapse Studio** card, select **Open** to open Synapse Studio in a new browser tab; signing in if prompted.
3. On the left side of Synapse Studio, use the ›› icon to expand the menu - this reveals the different pages within Synapse Studio that you'll use to manage resources and perform data analytics tasks.
4. On the **Manage** page, select the **Apache Spark pools** tab and note that a Spark pool with a name similar to **spark*xxxxxxx*** has been provisioned in the workspace.
5. On the **Data** page, view the **Linked** tab and verify that your workspace includes a link to your Azure Data Lake Storage Gen2 storage account, which should have a name similar to **synapse*xxxxxxx* (Primary - datalake*xxxxxxx*)**.
6. Expand your storage account and verify that it contains a file system container named **files (Primary)**.
7. Select the **files** container, and note that it contains folders named **data** and **synapse**. The synapse folder is used by Azure Synapse, and the **data** folder contains the data files you are going to query.
8. Open the **data** folder and observe that it contains .csv files for three years of sales data.
9. Right-click any of the files and select **Preview** to see the data it contains. Note that the files contain a header row, so you can select the option to display column headers.
10. Close the preview. Then download the **Spark Transform.ipynb** from from

Viewing the data:



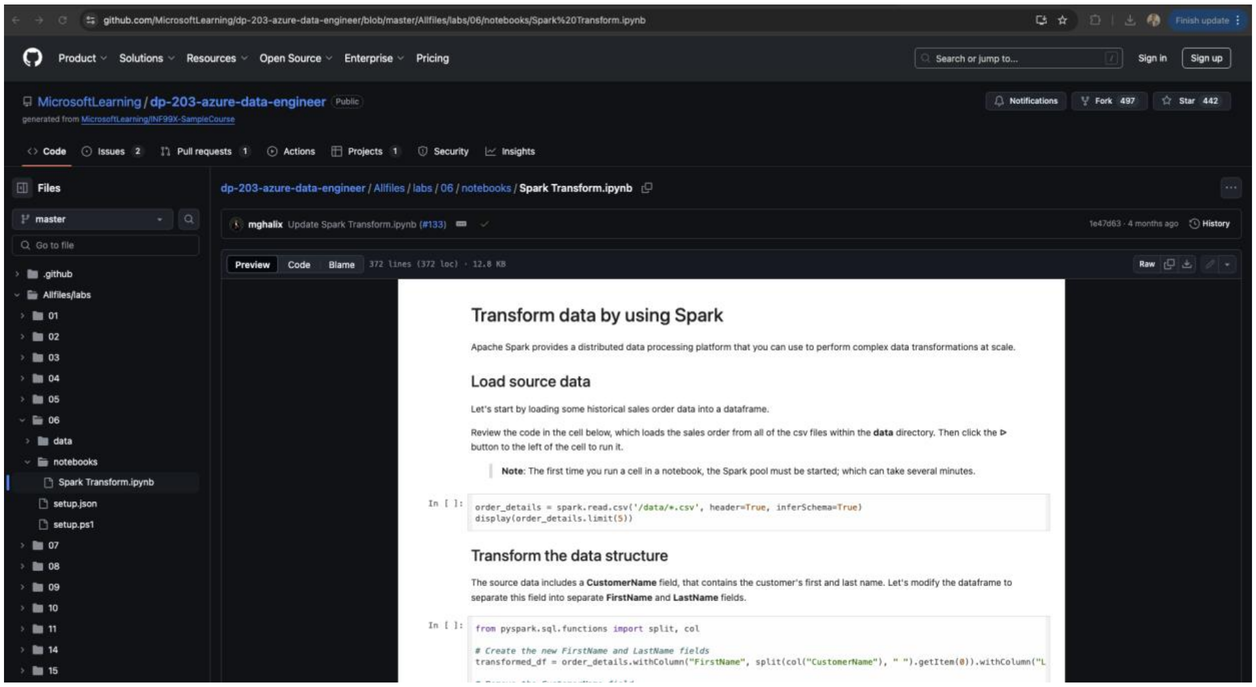
2019.csv

Path https://datalake9jwu8c1.dfs.core.windows.net/files/data/2019.csv
Modified 3/24/2025, 11:58:10 PM

With column header On

SALESORDER...	SALESORDER...	ORDERDATE	CUSTOME
SO43701	1	2019-07-01	Christy Zhu
SO43704	1	2019-07-01	Julio Ruiz
SO43705	1	2019-07-01	Curtis Lu
SO43700	1	2019-07-01	Ruben Prasa
SO43703	1	2019-07-01	Albert Alvare
SO43697	1	2019-07-01	Cole Watsor
SO43699	1	2019-07-01	Sydney Wrig
SO43702	1	2019-07-01	Colin Anand
SO43698	1	2019-07-01	Rachael Mar
SO43707	1	2019-07-02	Emma Browi
SO43711	1	2019-07-02	Courtney Ed

Using Github to download file:



github.com/MicrosoftLearning/dp-203-azure-data-engineer/blob/master/Allfiles/labs/06/notebooks/Spark%20Transform.ipynb

MicrosoftLearning / dp-203-azure-data-engineer

dp-203-azure-data-engineer / Allfiles / labs / 06 / notebooks / Spark Transform.ipynb

mghalix Update Spark Transform.ipynb (#133)

372 lines (372 loc) · 12.8 KB

Transform data by using Spark

Apache Spark provides a distributed data processing platform that you can use to perform complex data transformations at scale.

Load source data

Let's start by loading some historical sales order data into a dataframe.

Review the code in the cell below, which loads the sales order from all of the csv files within the **data** directory. Then click the ▷ button to the left of the cell to run it.

> **Note**: The first time you run a cell in a notebook, the Spark pool must be started; which can take several minutes.

```
In [ ]: order_details = spark.read.csv('/data/*.csv', header=True, inferSchema=True)
        display(order_details.limit(5))
```

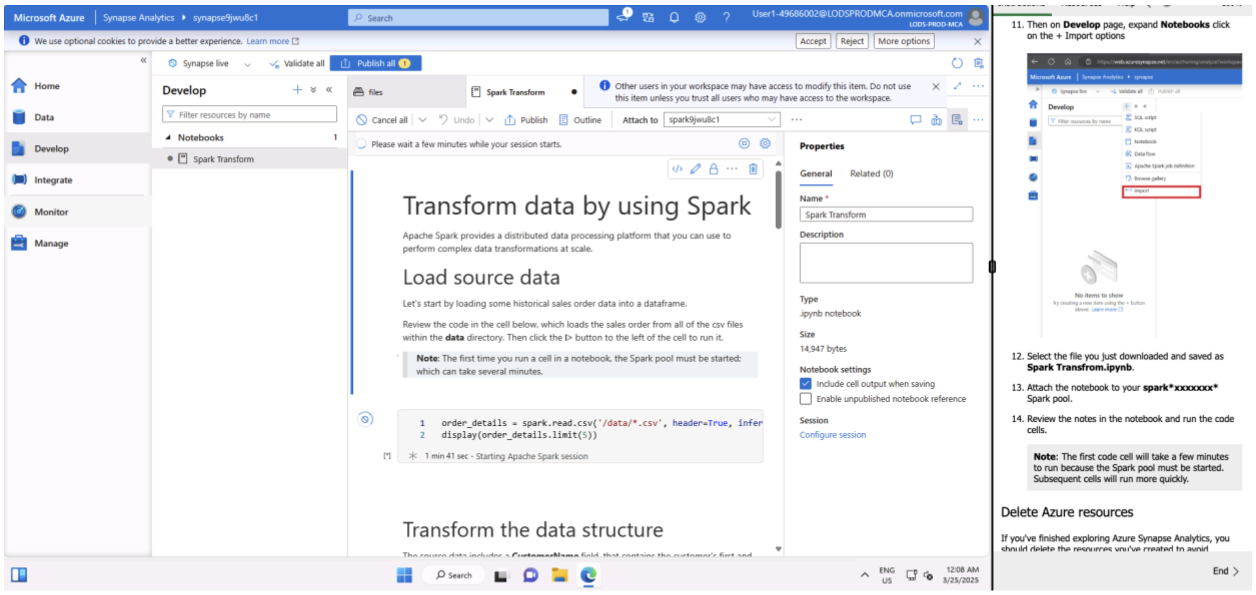
Transform the data structure

The source data includes a **CustomerName** field, that contains the customer's first and last name. Let's modify the dataframe to separate this field into separate **FirstName** and **LastName** fields.

```
In [ ]: from pyspark.sql.functions import split, col

        # Create the new FirstName and LastName fields
        transformed_df = order_details.withColumn("FirstName", split(col("CustomerName"), " ").getItem(0)).withColumn("L
```

Importing the file Spark Transform.ipynb and attaching the notebook to spark pool:



Microsoft Azure | Synapse Analytics ▸ synapse9jwu8c1

User1-49686002@LODSPRODMCA.onmicrosoft.com
LODS-PROD-MCA

We use optional cookies to provide a better experience. Learn more

Develop

Notebooks 1

Spark Transform

Other users in your workspace may have access to modify this item. Do not use this item unless you trust all users who may have access to the workspace.

Attach to spark9jwu8c1

Please wait a few minutes while your session starts.

Transform data by using Spark

Apache Spark provides a distributed data processing platform that you can use to perform complex data transformations at scale.

Load source data

Let's start by loading some historical sales order data into a dataframe.

Review the code in the cell below, which loads the sales order from all of the csv files within the **data** directory. Then click the ▷ button to the left of the cell to run it.

> **Note**: The first time you run a cell in a notebook, the Spark pool must be started: which can take several minutes.

```
order_details = spark.read.csv('/data/*.csv', header=True, infer
display(order_details.limit(5))
```

```
[*] 1 min 41 sec - Starting Apache Spark session
```

Transform the data structure

Properties

Name *: Spark Transform

Description:

Type: .ipynb notebook

Size: 14,947 bytes

Notebook settings
- [✓] Include cell output when saving
- [ ] Enable unpublished notebook reference

Session: Configure session

11. Then on **Develop** page, expand **Notebooks** click on the + Import options

12. Select the file you just downloaded and saved as **Spark Transfrom.ipynb**.

13. Attach the notebook to your **spark*xxxxxxx*** Spark pool.

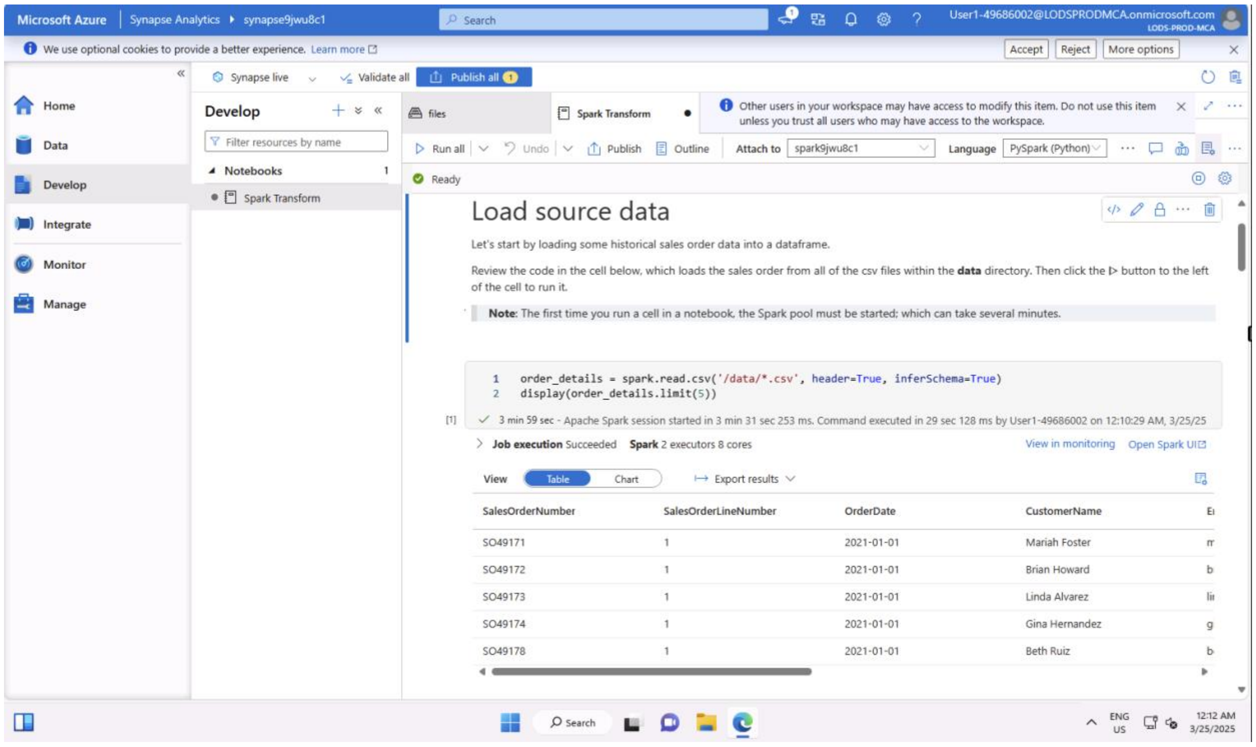
14. Review the notes in the notebook and run the code cells.

> **Note**: The first code cell will take a few minutes to run because the Spark pool must be started. Subsequent cells will run more quickly.

Delete Azure resources

If you've finished exploring Azure Synapse Analytics, you should delete the resources you've created to avoid

Loading the Source data:



Microsoft Azure | Synapse Analytics ▸ synapse9jwu8c1
Search
User1-49686002@LODSPRODMCA.onmicrosoft.com
LODS-PROD-MCA
We use optional cookies to provide a better experience. Learn more
Accept
Reject
More options
Synapse live
Validate all
Publish all 1
Home
Data
Develop
Integrate
Monitor
Manage
Develop
Filter resources by name
Notebooks 1
Spark Transform
files
Spark Transform
Other users in your workspace may have access to modify this item. Do not use this item unless you trust all users who may have access to the workspace.
Run all
Undo
Publish
Outline
Attach to spark9jwu8c1
Language PySpark (Python)
Ready
Search
ENG US
12:12 AM 3/25/2025

Load source data

Let's start by loading some historical sales order data into a dataframe.

Review the code in the cell below, which loads the sales order from all of the csv files within the **data** directory. Then click the ▷ button to the left of the cell to run it.

Note: The first time you run a cell in a notebook, the Spark pool must be started; which can take several minutes.

```
order_details = spark.read.csv('/data/*.csv', header=True, inferSchema=True)
display(order_details.limit(5))
```

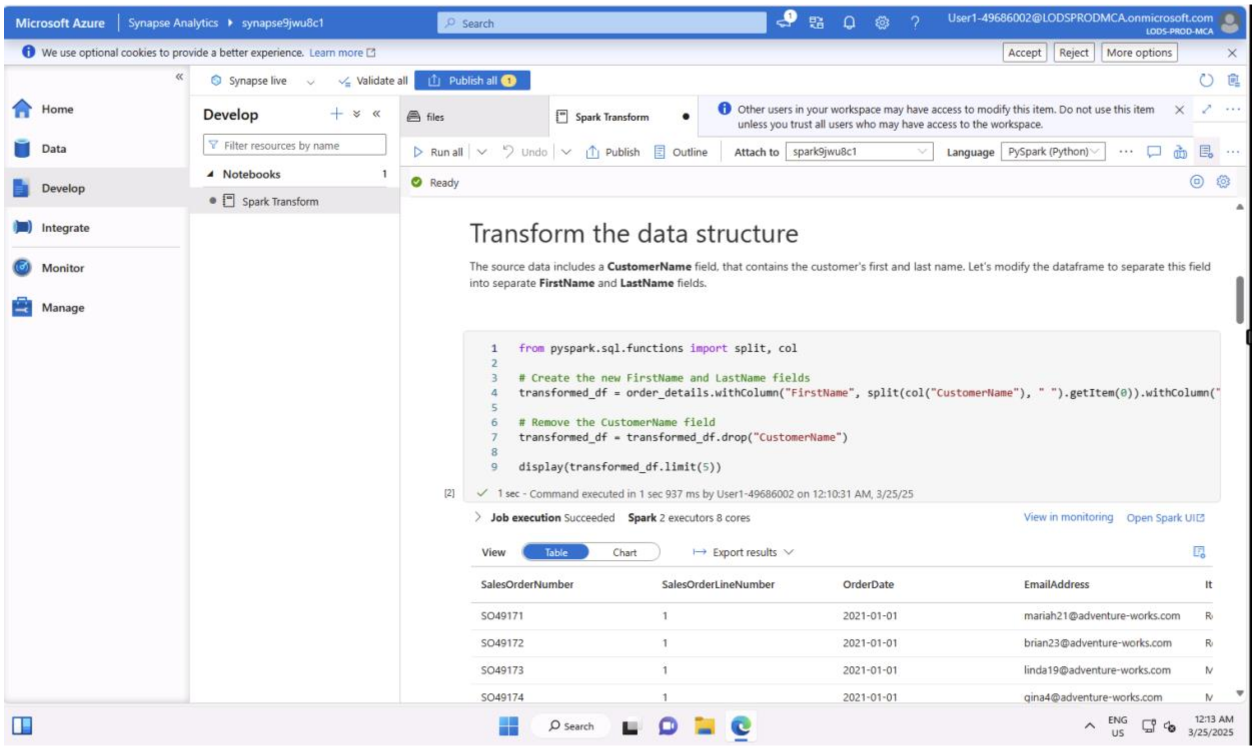
[1] ✓ 3 min 59 sec - Apache Spark session started in 3 min 31 sec 253 ms. Command executed in 29 sec 128 ms by User1-49686002 on 12:10:29 AM, 3/25/25

Job execution Succeeded **Spark** 2 executors 8 cores View in monitoring Open Spark UI

View Table Chart Export results

SalesOrderNumber	SalesOrderLineNumber	OrderDate	CustomerName	E
SO49171	1	2021-01-01	Mariah Foster	m
SO49172	1	2021-01-01	Brian Howard	b
SO49173	1	2021-01-01	Linda Alvarez	li
SO49174	1	2021-01-01	Gina Hernandez	g
SO49178	1	2021-01-01	Beth Ruiz	b

Transforming the Data Structure:



Microsoft Azure | Synapse Analytics ▸ synapse9jwu8c1
Search
User1-49686002@LODSPRODMCA.onmicrosoft.com
LODS-PROD-MCA
We use optional cookies to provide a better experience. Learn more
Accept
Reject
More options
Synapse live
Validate all
Publish all 1
Home
Data
Develop
Integrate
Monitor
Manage
Develop
Filter resources by name
Notebooks 1
Spark Transform
files
Spark Transform
Other users in your workspace may have access to modify this item. Do not use this item unless you trust all users who may have access to the workspace.
Run all
Undo
Publish
Outline
Attach to spark9jwu8c1
Language PySpark (Python)
Ready
Search
ENG US
12:13 AM 3/25/2025

Transform the data structure

The source data includes a **CustomerName** field, that contains the customer's first and last name. Let's modify the dataframe to separate this field into separate **FirstName** and **LastName** fields.

```
from pyspark.sql.functions import split, col

# Create the new FirstName and LastName fields
transformed_df = order_details.withColumn("FirstName", split(col("CustomerName"), " ").getItem(0)).withColumn("

# Remove the CustomerName field
transformed_df = transformed_df.drop("CustomerName")

display(transformed_df.limit(5))
```

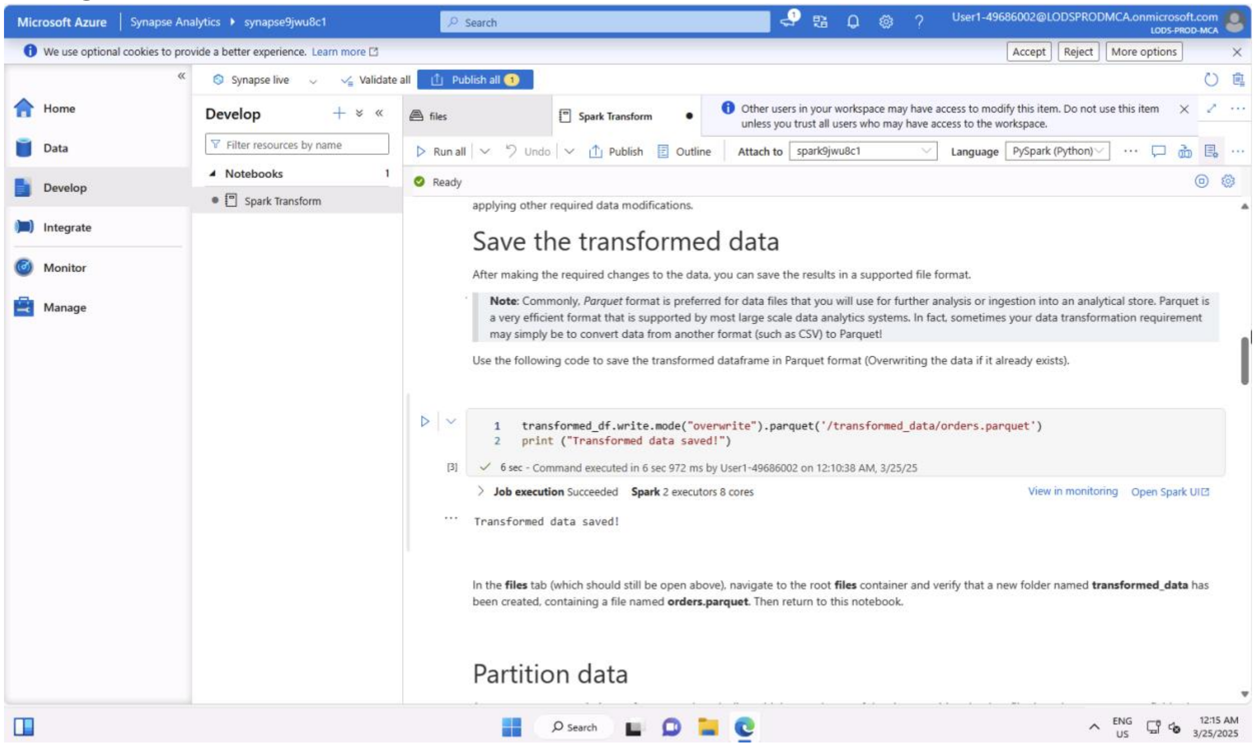
[2] ✓ 1 sec - Command executed in 1 sec 937 ms by User1-49686002 on 12:10:31 AM, 3/25/25

Job execution Succeeded **Spark** 2 executors 8 cores View in monitoring Open Spark UI

View Table Chart Export results

SalesOrderNumber	SalesOrderLineNumber	OrderDate	EmailAddress	It
SO49171	1	2021-01-01	mariah21@adventure-works.com	R
SO49172	1	2021-01-01	brian23@adventure-works.com	R
SO49173	1	2021-01-01	linda19@adventure-works.com	M
SO49174	1	2021-01-01	gina4@adventure-works.com	M

Saving the transformed data:



applying other required data modifications.

Save the transformed data

After making the required changes to the data, you can save the results in a supported file format.

> **Note**: Commonly, *Parquet* format is preferred for data files that you will use for further analysis or ingestion into an analytical store. Parquet is a very efficient format that is supported by most large scale data analytics systems. In fact, sometimes your data transformation requirement may simply be to convert data from another format (such as CSV) to Parquet!

Use the following code to save the transformed dataframe in Parquet format (Overwriting the data if it already exists).

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

[3] 6 sec - Command executed in 6 sec 972 ms by User1-49686002 on 12:10:38 AM, 3/25/25

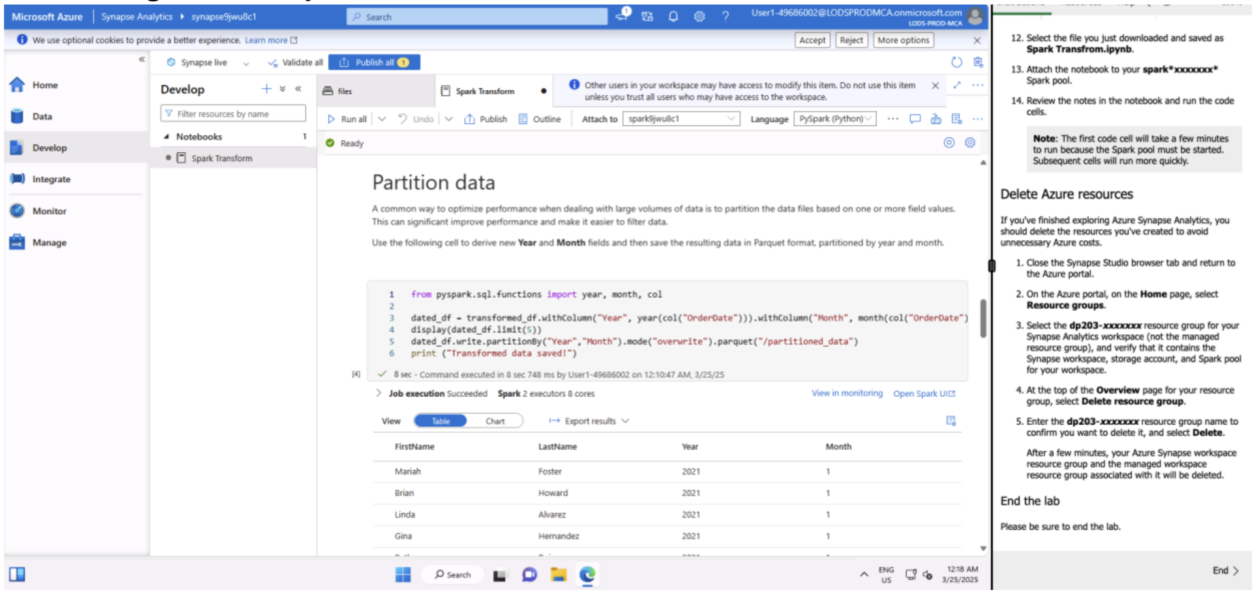
Job execution Succeeded **Spark** 2 executors 8 cores

```
Transformed data saved!
```

In the **files** tab (which should still be open above), navigate to the root **files** container and verify that a new folder named **transformed_data** has been created, containing a file named **orders.parquet**. Then return to this notebook.

Partition data

Partitioning the data by Year and month:



Partition data

A common way to optimize performance when dealing with large volumes of data is to partition the data files based on one or more field values. This can significant improve performance and make it easier to filter data.

Use the following cell to derive new **Year** and **Month** fields and then save the resulting data in Parquet format, partitioned by year and month.

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

dated_df = transformed_df.withColumn("Year", year(col("OrderDate"))).withColumn("Month", month(col("OrderDate")
display(dated_df.limit(5))
dated_df.write.partitionBy("Year","Month").mode("overwrite").parquet("/partitioned_data")
print ("Transformed data saved!")
```

[4] 8 sec - Command executed in 8 sec 748 ms by User1-49686002 on 12:10:47 AM, 3/25/25

Job execution Succeeded **Spark** 2 executors 8 cores

FirstName	LastName	Year	Month
Mariah	Foster	2021	1
Brian	Howard	2021	1
Linda	Alvarez	2021	1
Gina	Hernandez	2021	1

12. Select the file you just downloaded and saved as **Spark Transfrom.ipynb**.
13. Attach the notebook to your **spark*xxxxxxx*** Spark pool.
14. Review the notes in the notebook and run the code cells.

> **Note**: The first code cell will take a few minutes to run because the Spark pool must be started. Subsequent cells will run more quickly.

Delete Azure resources

If you've finished exploring Azure Synapse Analytics, you should delete the resources you've created to avoid unnecessary Azure costs.

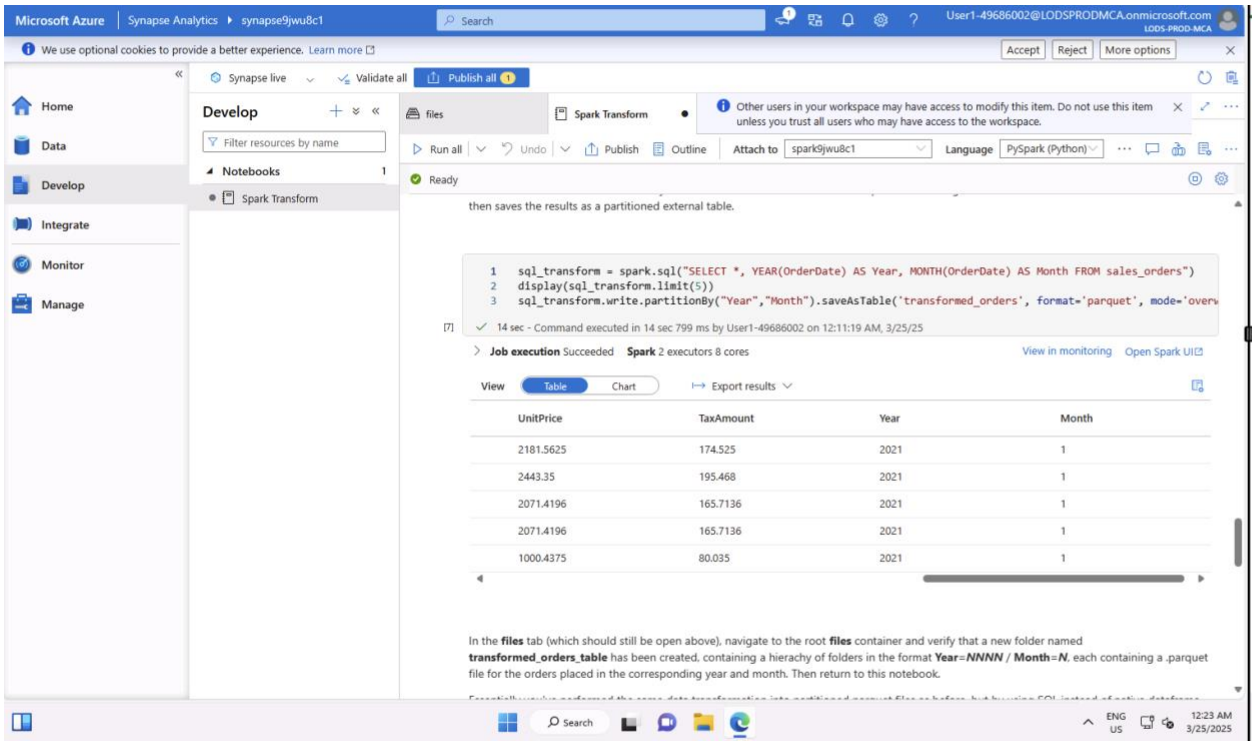
1. Close the Synapse Studio browser tab and return to the Azure portal.
2. On the Azure portal, on the **Home** page, select **Resource groups**.
3. Select the **dp203-*xxxxxxx*** resource group for your Synapse Analytics workspace (not the managed resource group), and verify that it contains the Synapse workspace, storage account, and Spark pool for your workspace.
4. At the top of the **Overview** page for your resource group, select **Delete resource group**.
5. Enter the **dp203-*xxxxxxx*** resource group name to confirm you want to delete it, and select **Delete**.

   After a few minutes, your Azure Synapse workspace resource group and the managed workspace resource group associated with it will be deleted.

End the lab

Please be sure to end the lab.

Using SQL to transform data:



then saves the results as a partitioned external table.

```
sql_transform = spark.sql("SELECT *, YEAR(OrderDate) AS Year, MONTH(OrderDate) AS Month FROM sales_orders")
display(sql_transform.limit(5))
sql_transform.write.partitionBy("Year","Month").saveAsTable('transformed_orders', format='parquet', mode='overw
```

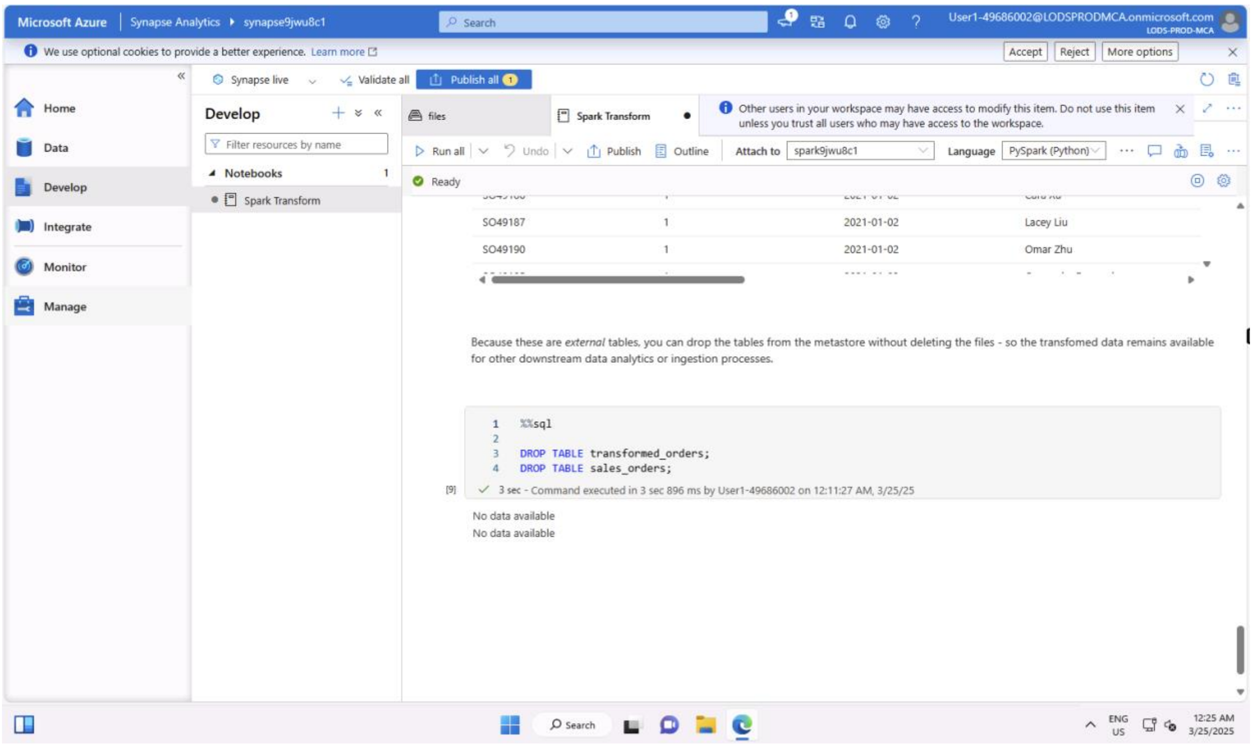
[7] ✓ 14 sec - Command executed in 14 sec 799 ms by User1-49686002 on 12:11:19 AM, 3/25/25

Job execution Succeeded Spark 2 executors 8 cores

UnitPrice	TaxAmount	Year	Month
2181.5625	174.525	2021	1
2443.35	195.468	2021	1
2071.4196	165.7136	2021	1
2071.4196	165.7136	2021	1
1000.4375	80.035	2021	1

In the **files** tab (which should still be open above), navigate to the root **files** container and verify that a new folder named **transformed_orders_table** has been created, containing a hierachy of folders in the format **Year=*NNNN*** / **Month=*N***, each containing a .parquet file for the orders placed in the corresponding year and month. Then return to this notebook.

Dropping the external tables from the metastore without deleting the files:



SO49187	1	2021-01-02	Lacey Liu
SO49190	1	2021-01-02	Omar Zhu

Because these are *external* tables, you can drop the tables from the metastore without deleting the files - so the transfomed data remains available for other downstream data analytics or ingestion processes.

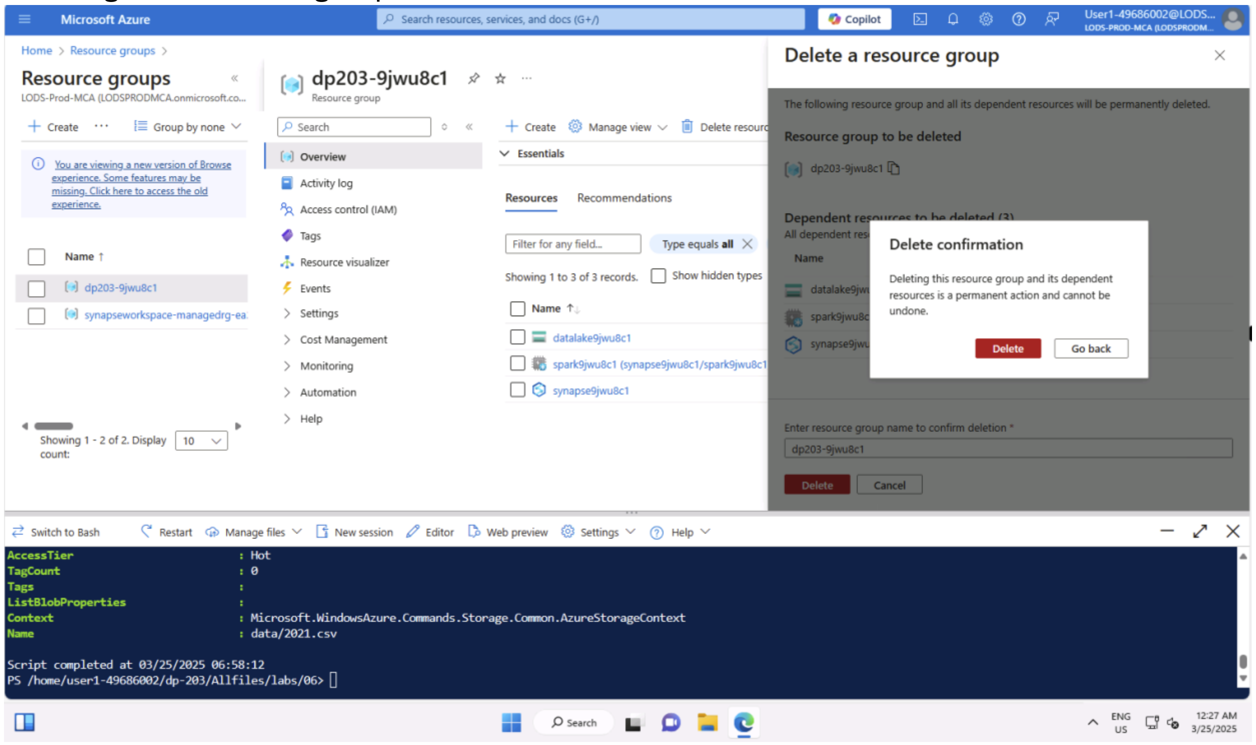
```
%%sql

DROP TABLE transformed_orders;
DROP TABLE sales_orders;
```

[9] ✓ 3 sec - Command executed in 3 sec 896 ms by User1-49686002 on 12:11:27 AM, 3/25/25

No data available
No data available

Deleting the Resource group:



You are viewing a new version of Browse experience. Some features may be missing. Click here to access the old experience.

The following resource group and all its dependent resources will be permanently deleted.

Deleting this resource group and its dependent resources is a permanent action and cannot be undone.

```
AccessTier                  : Hot
TagCount                    : 0
Tags                        :
ListBlobProperties          :
Context                     : Microsoft.WindowsAzure.Commands.Storage.Common.AzureStorageContext
Name                        : data/2021.csv

Script completed at 03/25/2025 06:58:12
PS /home/user1-49686002/dp-203/Allfiles/labs/06> 
```

Conclusion:

In this lab, I learned how to use Spark for data transformation, from setting up in Synapse Studio to working with GitHub and Spark notebooks. I explored data, applied transformations, and optimized it by partitioning by year and month. Using SQL, I refined the data while also managing external tables without losing underlying files. Lastly, I practiced resource management by deleting the resource group.