

Lab 7 Spark Delta Lake

BUAN 6390.001 – Analytics Practicum

Name: Swastik Bhatnagar
Net ID: Sxb220210

Connecting to Powershell:

The screenshot displays the Microsoft Azure portal interface. The top navigation bar includes the Azure logo, a search bar, and a user profile. The main content area is divided into three sections: 'Azure services' with icons for various services like Kubernetes, Virtual machines, and App Services; 'Resources' with a table for recent and favorite resources; and 'Navigate' with links to Subscriptions, Resource groups, All resources, and Dashboard. A 'Cloud Shell' icon is visible in the top right corner. Below the main content area, a terminal window is open, showing a PowerShell session. The terminal output includes instructions for using the Cloud Shell, a warning about the subscription, and the start of the authentication and Azure drive setup process. The terminal prompt is 'PS /home/user1-49711676:'.

Microsoft Azure

Search resources, services, and docs (3+)

Cloud Shell

User1-49711676@LOG... 100% 9000 sec's cloudshell

Azure services

Create a resource Quickstart Center Azure AI services Kubernetes services Virtual machines App Services Storage accounts SQL databases Azure Cosmos DB More services

Resources

Recent Favorite

Name Type Last Viewed

No resources have been viewed recently

View all resources

Navigate

Subscriptions Resource groups All resources Dashboard

Switch to Bash Restart Manage files New session Editor Web preview Settings Help

Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

Subscription used to launch your CloudShell f0f9867-8ef9-4d89-b098-27c6cd2d9221 is not registered to Microsoft.CloudShell Namespace. Please follow these instructions "https://aka.ms/RegisterCloudShell" to register. In future, unregistered subscriptions will have restricted access to CloudShell service.

Your Cloud Shell session will be ephemeral so no files or system changes will persist beyond your current session.

NOTE: Azure Cloud Shell now includes Predictive IntelliSense! Learn more: https://aka.ms/CloudShell/IntelliSense

VERBOSE: Authenticating to Azure ...
VERBOSE: Building your Azure drive ...
PS /home/user1-49711676:

1 Hr 58 Min Remaining

Instructions Resources Help

Before you start

You'll need an Azure subscription in which you have administrative-level access.

Provision an Azure Synapse Analytics workspace

You'll need an Azure Synapse Analytics workspace with access to data lake storage and an Apache Spark pool that you can use to query and process files in the data lake.

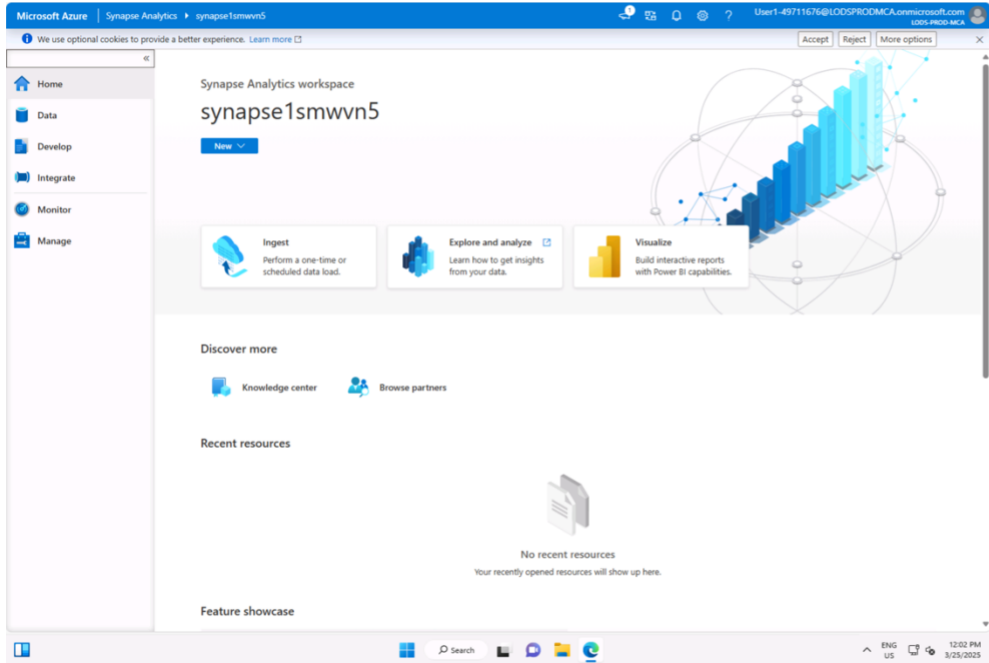
In this exercise, you'll use a combination of a PowerShell script and an ARM template to provision an Azure Synapse Analytics workspace.

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:
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.

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**.

End >

Opening Synapse Studio:



The screenshot shows the Microsoft Azure Synapse Analytics workspace 'synapse1smwvn5'. The interface includes a left sidebar with navigation options: Home, Data, Develop, Integrate, Monitor, and Manage. The main area displays the workspace name and a 'New' button. Below this, there are three primary actions: 'Ingest' (Perform a one-time or scheduled data load), 'Explore and analyze' (Learn how to get insights from your data), and 'Visualize' (Build interactive reports with Power BI capabilities). A 'Discover more' section links to the Knowledge center and Browse partners. A 'Recent resources' section shows 'No recent resources' with a note that recently opened resources will show up here. A 'Feature showcase' section is also visible. The top right corner shows the user 'User1-49711676@L0D5PR0DMCA.onmicrosoft.com' and the date '3/25/2025'.

Explore the data in the data lake

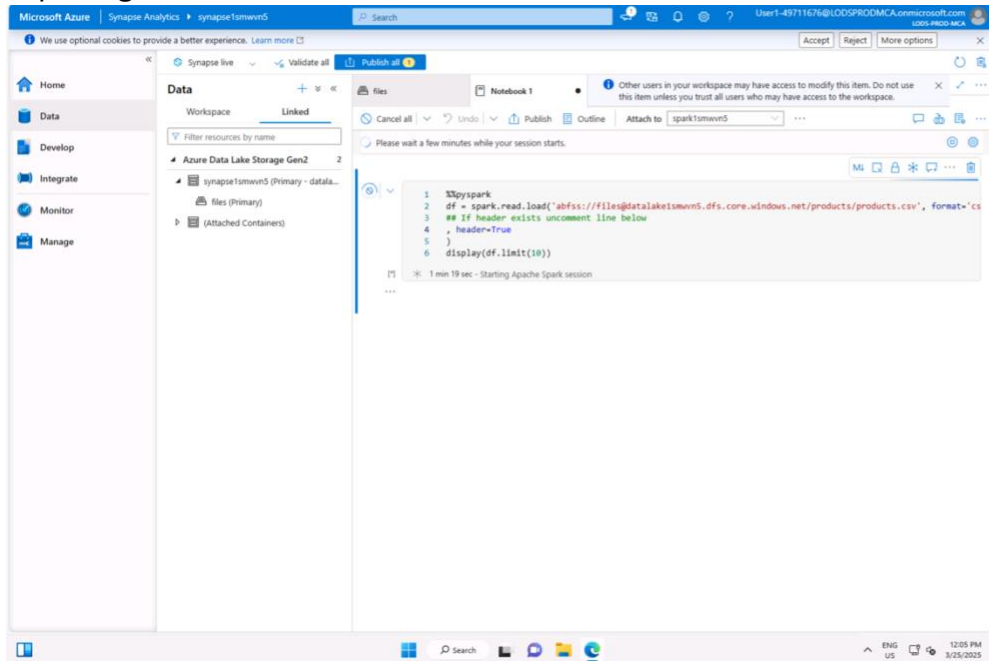
1. After the script has completed, in the Azure portal, go to the **dp203-xxxxxxx** resource group that it created, and select your Synapse workspace.
2. In the **Overview** page for your Synapse workspace, in the **Open Synapse Studio** card, select **Open** to open Synapse Studio in a new browser tab, 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 **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*)**.
5. Expand your storage account and verify that it contains a file system container named **files**.
6. Select the **files** container, and note that it contains a folder named **products**. This folder contains the data you are going to work with in this exercise.
7. Open the **products** folder, and observe that it contains a file named **products.csv**.
8. Select **products.csv**, and then in the **New notebook** list on the toolbar, select **Load to DataFrame**.
9. In the **Notebook 1** pane that opens, in the **Attach to** list, select the **sparkxxxxxxx** Spark pool and ensure that the **Language** is set to **PySpark (Python)**.
10. Review the code in the first (and only) cell in the notebook, which should look like this:

```
Python
%%pyspark
df = spark.read.load('abfss://files@datalake1smwvn5.dfs.core.windows.net/products/products.csv', format='csv', header=True)
display(df.limit(10))
```

11. Uncomment the `header=True` line (because the

End >

Exploring data in data lake:



The screenshot shows the Microsoft Azure Synapse Analytics workspace 'synapse1smwvn5' with the **Data** page selected. The **Linked** tab is active, showing the workspace's connection to the **Azure Data Lake Storage Gen2** account. The **files** container is expanded, showing the **products** folder. The **products.csv** file is selected. The **Notebook 1** pane is open, showing the code for loading the data into a DataFrame. The code is as follows:

```
%%pyspark
df = spark.read.load('abfss://files@datalake1smwvn5.dfs.core.windows.net/products/products.csv', format='csv', header=True)
display(df.limit(10))
```

The notebook shows a status of '1 min 19 sec - Starting Apache Spark session'.

Explore the data in the data lake

1. After the script has completed, in the Azure portal, go to the **dp203-xxxxxxx** resource group that it created, and select your Synapse workspace.
2. In the **Overview** page for your Synapse workspace, 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 **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*)**.
5. Expand your storage account and verify that it contains a file system container named **files**.
6. Select the **files** container, and note that it contains a folder named **products**. This folder contains the data you are going to work with in this exercise.
7. Open the **products** folder, and observe that it contains a file named **products.csv**.
8. Select **products.csv**, and then in the **New notebook** list on the toolbar, select **Load to DataFrame**.
9. In the **Notebook 1** pane that opens, in the **Attach to** list, select the **sparkxxxxxxx** Spark pool and ensure that the **Language** is set to **PySpark (Python)**.
10. Review the code in the first (and only) cell in the notebook, which should look like this:

```
Python
%%pyspark
df = spark.read.load('abfss://files@datalake1smwvn5.dfs.core.windows.net/products/products.csv', format='csv', header=True)
display(df.limit(10))
```

11. Uncomment the `header=True` line (because the **products.csv** file has the column headers in the first

End >

Loading the data in delta table:

The screenshot shows a Synapse notebook with the following code and output:

```
1 delta_table_path = "/delta/products-delta"
2 df.write.format("delta").save(delta_table_path)
```

Job execution Succeeded Spark 2 executors 8 cores

```
1 from delta.tables import *
2 from pyspark.sql.functions import *
3
4 # Create a deltaTable object
5 deltaTable = DeltaTable.forPath(spark, delta_table_path)
6
7 # Update the table (reduce price of product 771 by 10%)
8 deltaTable.update(
9     condition = "ProductID = 771",
10     set = { "ListPrice": "ListPrice * 0.9" })
11
12 # View the updated data as a dataframe
13 deltaTable.toDF().show(10)
```

Job execution Succeeded Spark 2 executors 8 cores

ProductID	ProductName	Category	ListPrice
771	Mountain-100 Silv...	Mountain Bikes	3059.9991
772	Mountain-100 Silv...	Mountain Bikes	3399.9900
773	Mountain-100 Silv...	Mountain Bikes	3399.9900
774	Mountain-100 Silv...	Mountain Bikes	3399.9900
775	Mountain-100 Blac...	Mountain Bikes	3374.9900
776	Mountain-100 Blac...	Mountain Bikes	3374.9900
777	Mountain-100 Blac...	Mountain Bikes	3374.9900

On the right side, there are instructions and code snippets:

View the updated data as a dataframe
`deltaTable.toDF().show(10)`

The data is loaded into a **DeltaTable** object and updated. You can see the update reflected in the query results.

4. Add another new code cell with the following code and run it:

```
Python
1 new_df = spark.read.format("delta")
2 new_df.show(10)
```

The code loads the delta table data into a data frame from its location in the data lake, verifying that the change you made via a **DeltaTable** object has been persisted.

5. Modify the code you just ran as follows, specifying the option to use the **time travel** feature of delta lake to view a previous version of the data.

```
Python
1 new_df = spark.read.format("delta")
2 new_df.show(10)
```

When you run the modified code, the results show the original version of the data.

6. Add another new code cell with the following code and run it:

```
Python
1 deltaTable.history(10).show(20, False)
```

The history of the last 20 changes to the table is shown - there should be two (the original creation, and the update you made.)

Create catalog tables

So far you've worked with delta tables by loading data from the folder containing the parquet files on which the table is based. You can define catalog tables that encapsulate the data and provide a named table entity that you can reference in SQL code. Spark supports two kinds of catalog

Creating Catalog tables:- External Table:

The screenshot shows a Synapse notebook with the following code and output:

```
1 spark.sql("CREATE DATABASE Adventureworks")
2 spark.sql("CREATE TABLE Adventureworks.ProductsExternal USING DELTA LOCATION '{0}'.format(delta_table_path)")
3 spark.sql("DESCRIBE EXTENDED Adventureworks.ProductsExternal").show(truncate=False)
```

Job execution Succeeded Spark 2 executors 8 cores

col_name	data_type	comment
ProductID	string	
ProductName	string	
Category	string	
ListPrice	string	

Detailed Table Information

Name	spark_catalog.adventureworks.productsexternal
Type	EXTERNAL
Location	abfss://files@datalakeiswmv5.dfs.core.windows.net/delta/products-delta
Provider	delta
Owner	trusted-service-user
Table Properties	[delta.minReaderVersion=1,delta.minWriterVersion=2]

On the right side, there are instructions and code snippets:

1. In a new code cell, add and run the following code:

```
Python
1 spark.sql("CREATE DATABASE Adventureworks")
2 spark.sql("CREATE TABLE Adventureworks.ProductsExternal USING DELTA LOCATION '{0}'.format(delta_table_path)")
3 spark.sql("DESCRIBE EXTENDED Adventureworks.ProductsExternal").show(truncate=False)
```

This code creates a new database named **Adventureworks** and then creates an external table named **ProductsExternal** in that database based on the path to the parquet files you defined previously. It then displays a description of the table's properties. Note that the **Location** property is the path you specified.

2. Add a new code cell, and then enter and run the following code:

```
SQL
1 USE Adventureworks;
2 SELECT * FROM ProductsExternal;
```

The code uses SQL to switch context to the **Adventureworks** database (which returns no data) and then queries the **ProductsExternal** table (which returns a resultset containing the products data in the Delta Lake table).

Create a managed table

1. In a new code cell, add and run the following code:

```
Python
1 df.write.format("delta").saveAsTable(spark.sql("DESCRIBE EXTENDED Adventureworks.ProductsExternal").show(truncate=False))
```

This code creates a managed table named **ProductsManaged** based on the Dataframe you originally loaded from the **products.csv** file (before you updated the price of product 771). You do not specify a path for the parquet files used by the table - this is managed for you in the Hive metastore, and shown in the **Location** property in the table description (in the

The screenshot displays the Synapse Studio interface. On the left, the navigation pane shows 'Home', 'Data', 'Develop', 'Integrate', 'Monitor', and 'Manage'. The main area is titled 'Workspace' and shows a folder named 'synapse1mwms5 (Primary - delta...)'. Inside this folder, there are files like 'files (Primary)' and 'Attached Container'. A notebook titled 'Notebook 1' is open, showing a PySpark script:

```
1 df.write.format("delta").saveAsTable("AdventureWorks.ProductManaged")
2 spark.sql("DESCRIBE EXTENDED AdventureWorks.ProductManaged").show(truncate=False)
```

Below the code, a status bar indicates 'Job execution Succeeded Spark 2 execution 8 cores'. To the right of the code editor, a preview of the table schema is shown:

[col_name]	[data_type]
[ProductID]	[string]
[ProductName]	[string]
[Category]	[string]
[ListPrice]	[string]

On the far right, a snippet of SQL code is visible:

```
USE AdventureWorks;
SELECT * FROM ProductsExternal;
```

Microsoft Azure | Synapse Analytics | synapse1smwvns

Search

User1-49711676@L0DSPRODMCA.onmicrosoft.com

Accept Reject More options

Home Data Develop Integrate Monitor Manage

Data Workspace Linked

Filter resources by name

Azure Data Lake Storage Gen2 2

synapse1smwvns (Primary - data...

files (Primary)

(Attached Containers)

Run all Undo Publish Outline Attach to spark1smwvns Language PySpark (Python) Variables

Ready

```
1 %%sql
2
3 USE Adventureworks;
4
5 CREATE TABLE Products
6 USING DELTA
7 LOCATION '/delta/products-delta';
```

[13] ✓ 3 sec - Command executed in 5 sec 234 ms by User1-49711676 on 12:26:04 PM, 3/25/25

No data available

No data available

```
1 %%sql
2
3 USE Adventureworks;
4
5 SELECT * FROM Products;
```

[14] ✓ 3 sec - Command executed in 2 sec 1 ms by User1-49711676 on 12:26:37 PM, 3/25/25

Job execution Succeeded Spark 2 executors 8 cores

View in monitoring Open Spark UI

No data available

View Table Chart Export results

ProductID	ProductName	Category	ListPrice
771	Mountain-100 Silver, 38	Mountain Bikes	3059.991
772	Mountain-100 Silver, 42	Mountain Bikes	3399.9900
773	Mountain-100 Silver, 44	Mountain Bikes	3399.9900
774	Mountain-100 Silver, 48	Mountain Bikes	3399.9900
775	Mountain-100 Black, 38	Mountain Bikes	3374.9900
776	Mountain-100 Black, 42	Mountain Bikes	3374.9900
777	Mountain-100 Black, 44	Mountain Bikes	3374.9900
778	Mountain-100 Black, 48	Mountain Bikes	3374.9900
779	Mountain-200 Silver, 38	Mountain Bikes	2319.9900

ENG US 12:26 PM 3/25/2025

Using Delta table for streaming data:

Microsoft Azure | Synapse Analytics | synapse1smwv5

Home Data Develop Integrate Monitor Manage

Workspace Linked

Filter resources by name

Azure Data Lake Storage Gen2 2

synapse1smwv5 (Primary - data-lake-storage-gen2)

files (Primary)

(Attached Containers)

Ready

```
10 jsonSchema = StructType([
11   StructField("device", StringType(), False),
12   StructField("status", StringType(), False)
13 ])
14 iotstream = spark.readStream.schema(jsonSchema).option("maxFilesPerTrigger", 1).json(inputPath)
15
16 # Write some event data to the folder
17 device_data = [{"device": "Dev1", "status": "ok"}]
18 [{"device": "Dev1", "status": "ok"}]
19 [{"device": "Dev1", "status": "ok"}]
20 [{"device": "Dev2", "status": "error"}]
21 [{"device": "Dev1", "status": "ok"}]
22 [{"device": "Dev1", "status": "error"}]
23 [{"device": "Dev2", "status": "ok"}]
24 [{"device": "Dev2", "status": "error"}]
25 [{"device": "Dev1", "status": "ok"}]
26 mssparkutils.fs.put(inputPath + "data.txt", device_data, True)
27 print("Source stream created...")
```

✓ 1 sec - Command executed in 616 ms by User1-49711676 on 12:28:44 PM, 3/25/25

Source stream created...

+ Code + Markdown

```
1 # Write the stream to a delta table
2 delta_stream_table_path = "/delta/iotdevicedata"
3 checkpointpath = "/delta/checkpoint"
4 deltalStream = iotstream.writeStream.format("delta").option("checkpointLocation", checkpointpath).start(delta_stream_table_path)
5 print("Streaming to delta sink...")
```

✓ 1 sec - Command executed in 1 sec 125 ms by User1-49711676 on 12:29:39 PM, 3/25/25

Streaming to delta sink...

+ Code + Markdown

```
1 # Read the data in delta format into a dataframe
2 df = spark.read.format("delta").load(delta_stream_table_path)
3 display(df)
```

✓ 7 sec - Command executed in 6 sec 969 ms by User1-49711676 on 12:29:59 PM, 3/25/25

Job execution Succeeded Spark 2 executors 8 cores

View in monitoring Open Spark UI3

View Table Chart Export results

Microsoft Azure | Synapse Analytics | synapse1smwv5

Home Data Develop Integrate Monitor Manage

Workspace Linked

Filter resources by name

Azure Data Lake Storage Gen2 2

synapse1smwv5 (Primary - data-lake-storage-gen2)

files (Primary)

(Attached Containers)

Ready

```
1 # create a catalog table based on the streaming sink
2 spark.sql("CREATE TABLE IotDeviceData USING DELTA LOCATION '{0}'".format(delta_stream_table_path))
```

✓ 1 sec - Command executed in 1 sec 946 ms by User1-49711676 on 12:31:07 PM, 3/25/25

DataFrame[]

+ Code + Markdown

```
1 %sql
2
3 SELECT * FROM IotDeviceData;
```

✓ 1 sec - Command executed in 1 sec 878 ms by User1-49711676 on 12:31:33 PM, 3/25/25

Job execution Succeeded Spark 2 executors 8 cores

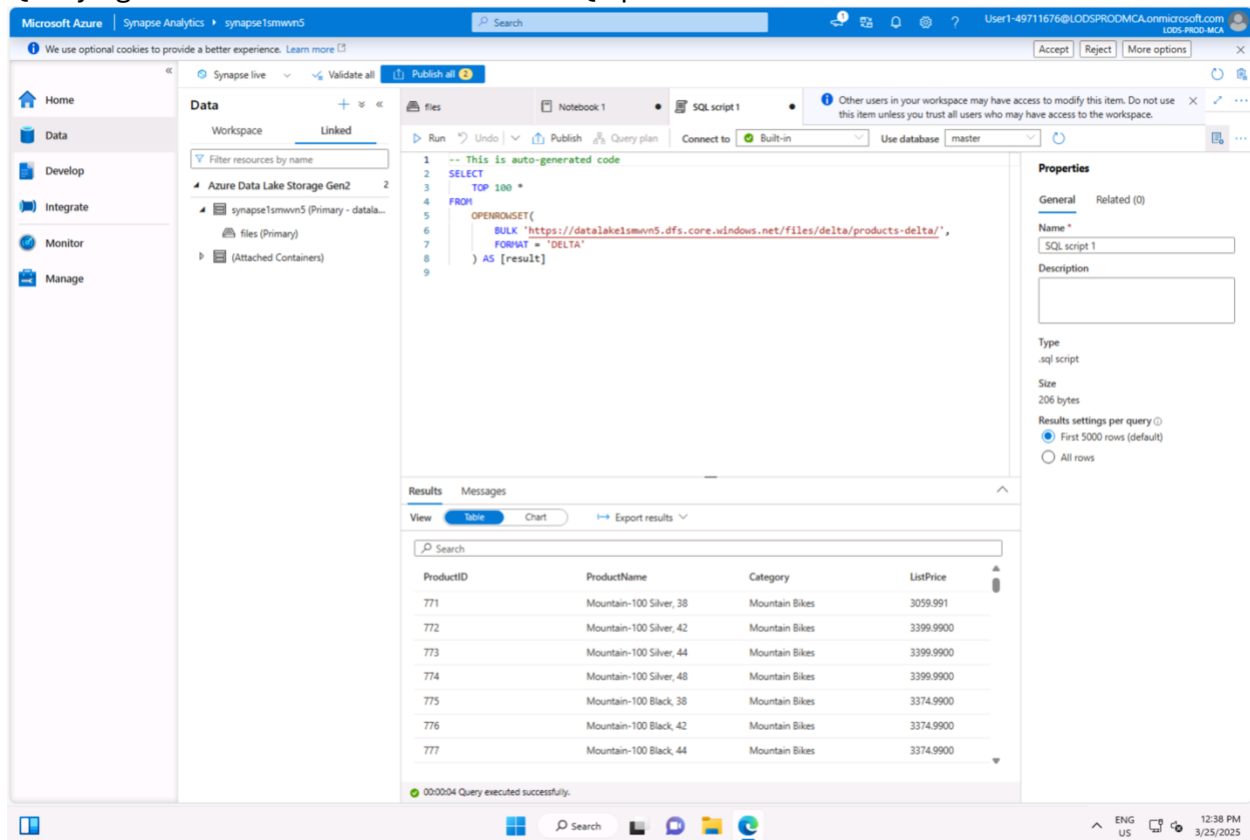
View in monitoring Open Spark UI3

View Table Chart Export results

device	status
Dev1	ok
Dev1	ok
Dev1	ok
Dev2	error
Dev1	ok
Dev1	error
Dev2	ok
Dev2	error
Dev1	ok

```
1 # Add more data to the source stream
2 more_data = [{"device": "Dev1", "status": "ok"}]
3 [{"device": "Dev1", "status": "ok"}]
```

Querying a delta table from a serverless SQL pool:



The screenshot shows the Microsoft Azure Synapse Analytics interface. The left sidebar contains navigation options: Home, Data, Develop, Integrate, Monitor, and Manage. The main workspace is titled "Data" and shows a "Workspace" view with a "Linked" section. The "Linked" section lists resources: "Azure Data Lake Storage Gen2", "synapse1smwv5 (Primary - dataa...", "files (Primary)", and "(Attached Containers)".

The central pane displays a SQL script in a "SQL script 1" editor. The script is as follows:

```
1 -- This is auto-generated code
2 SELECT
3   TOP 100 *
4 FROM
5   OPENROWSET(
6     BULK 'https://datalake1smwv5.dfs.core.windows.net/files/delta/products-delta/',
7     FORMAT = 'DELTA'
8   ) AS [result]
9
```

The right pane shows the "Properties" section for the script, including fields for Name, Description, Type, Size, and Results settings per query.

The bottom pane displays the "Results" of the query execution. The results are shown in a table view with the following columns: ProductID, ProductName, Category, and ListPrice. The results are as follows:

ProductID	ProductName	Category	ListPrice
771	Mountain-100 Silver, 38	Mountain Bikes	3059.991
772	Mountain-100 Silver, 42	Mountain Bikes	3399.9900
773	Mountain-100 Silver, 44	Mountain Bikes	3399.9900
774	Mountain-100 Silver, 48	Mountain Bikes	3399.9900
775	Mountain-100 Black, 38	Mountain Bikes	3374.9900
776	Mountain-100 Black, 42	Mountain Bikes	3374.9900
777	Mountain-100 Black, 44	Mountain Bikes	3374.9900

The status bar at the bottom indicates "00:00:04 Query executed successfully." The Windows taskbar at the bottom shows the time as 12:38 PM on 3/25/2025.

Deleting Azure resources:

The screenshot displays the Microsoft Azure portal interface. The top navigation bar shows the user is logged in as 'User1-49711676@LODS...'. The main content area is divided into two panels. The left panel, titled 'Resource groups', shows a list of resource groups under the 'dp203-1smwvn5' group. The right panel, titled 'Delete a resource group', shows the details of the resource group to be deleted, including its name and the list of dependent resources to be deleted.

Resource groups

LODS-Prod-MCA (LODSPROD.MCA.onmicrosoft.co...)

+ Create ... Group by none

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

☐ Name 1

☒ dp203-1smwvn5

☐ synapseworkspace-managedrg-30

Showing 1 - 2 of 2. Display count: 10

dp203-1smwvn5

Resource group

Search

+ Create Manage view Delete resource

Overview

Activity log

Access control (IAM)

Tags

Resource visualizer

Events

Settings

Cost Management

Monitoring

Automation

Help

Resources Recommendations

Filter for any field... Type equals all

Showing 1 to 3 of 3 records. Show hidden types

☐ Name ↑

☐ datalake1smwvn5

☐ spark1smwvn5 (synapse1smwvn5/spark1smwvn5)

☐ synapse1smwvn5

Delete a resource group

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

Resource group to be deleted

dp203-1smwvn5

Dependent resources to be deleted (3)

All dependent resources, including hidden types, are shown

Name	Resource type
datalake1smwvn5	Storage account
spark1smwvn5 (synapse1smwvn5/spark1smwvn5)	Apache Spark pool
synapse1smwvn5	Synapse workspace

Enter resource group name to confirm deletion *

dp203-1smwvn5

Delete Cancel

Switch to Bash Restart Manage files New session Editor Web preview Settings Help

```
SnapshotTime
ContinuationToken
VersionId
IsLatestVersion
AccessTier
TagCount
Tags
ListBlobProperties
Context
Name
: Microsoft.WindowsAzure.Commands.Storage.Common.AzureStorageContext
: products/products.csv

Script completed at 03/25/2025 19:00:36
PS /home/user1-49711676/dp-203/Allfiles/labs/07>
```

ENG US 12:39 PM 3/25/2025

Conclusion:

In this lab I learned how to connect to PowerShell, navigate Synapse Studio, and explore data in a Data Lake. Setting up and managing Delta tables, both external and managed, gave me a better understanding of how data is structured and queried. I also worked with streaming data and saw how Delta tables handle real-time updates. Querying from a serverless SQL pool was a great way to see how Delta Lake integrates with other tools.