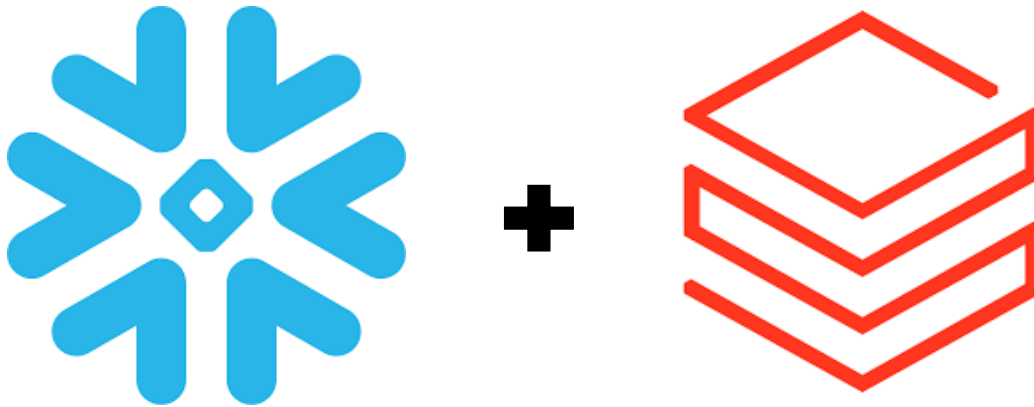


Add Data from Snowflake using Azure
Databricks notebook



In this article we are going to use azure Databricks pre-build notebook for Snowflake to:

- Create a table on the fly in Snowflake by using Data Science & Engineering App.
- Read the same table from snowflake and create a Delta table
- Querying the same table by using the Databricks SQL app.

Prerequisite:

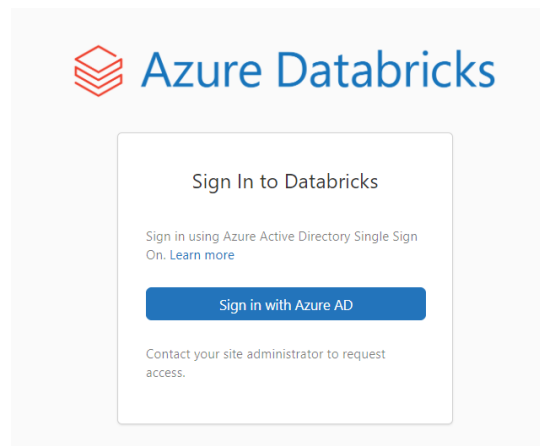
- 1- You need an Azure Databricks Premium workspace and Snowflake Account on Azure.
- 2- Access to Azure Key Vault

So, let's see how it's working.

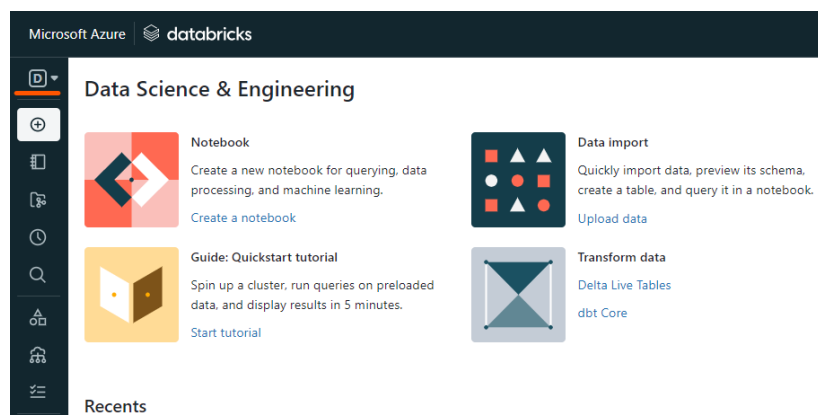
Before we start lets login the Databricks workspace

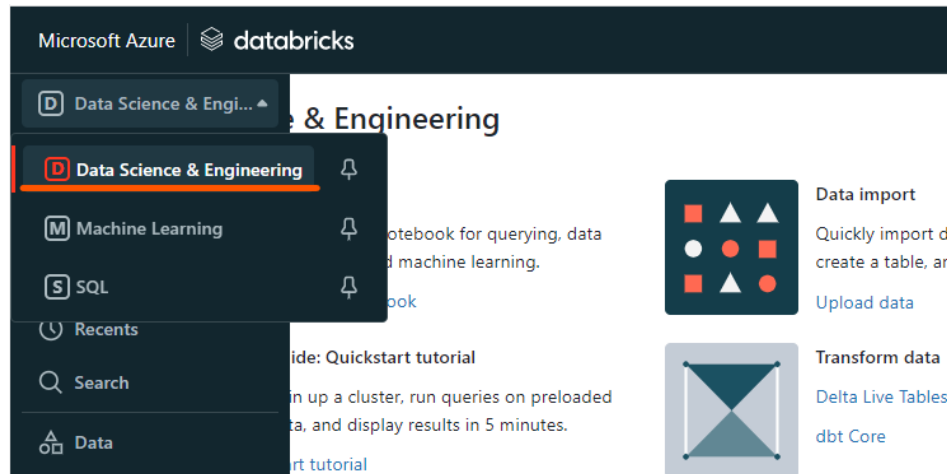
- Create a table on the fly in Snowflake

- 1- From Azure Databricks, click sign in with Azure AD to the workspace

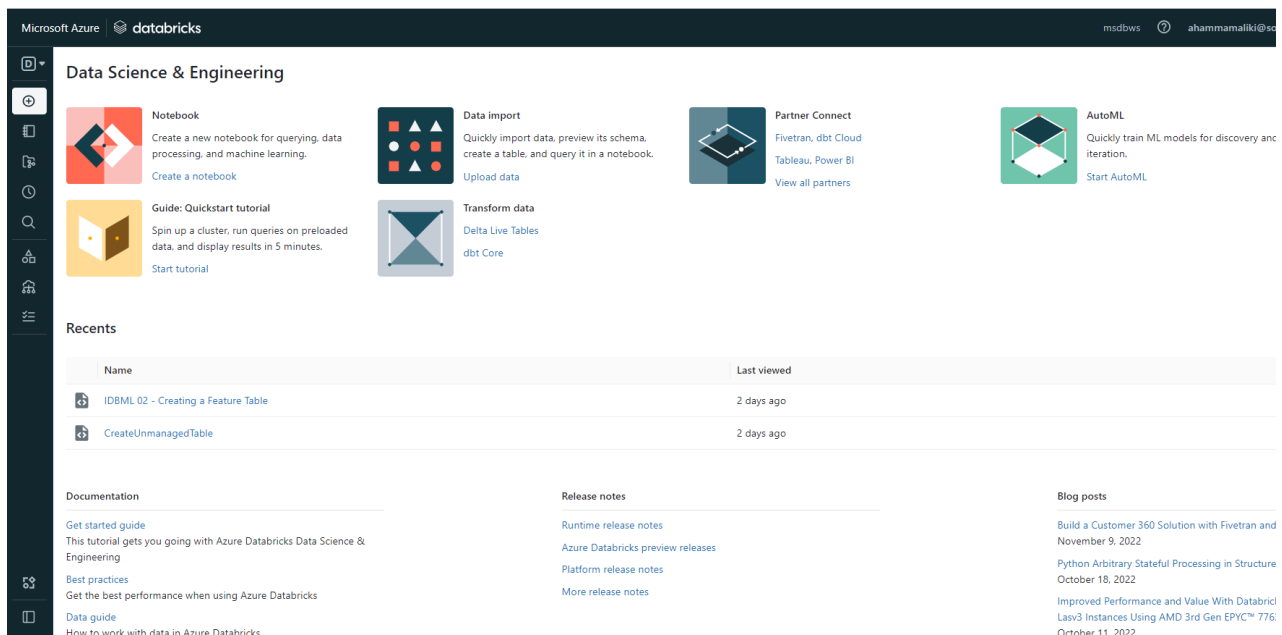


- 2- At the left side bar menu select the app switch to select the app you need to use.



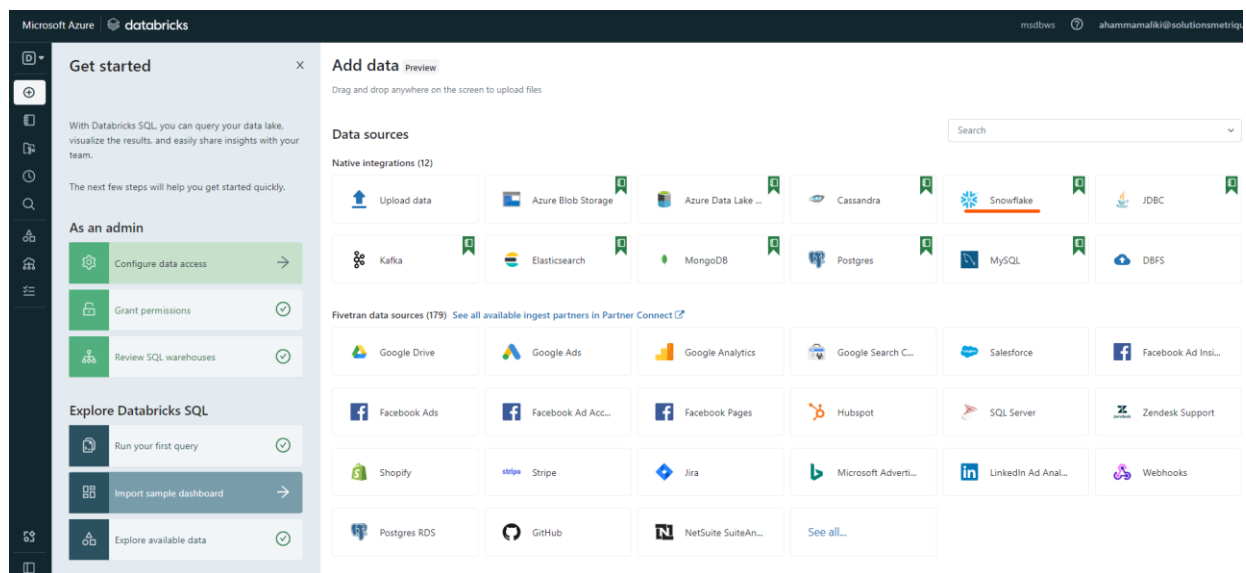
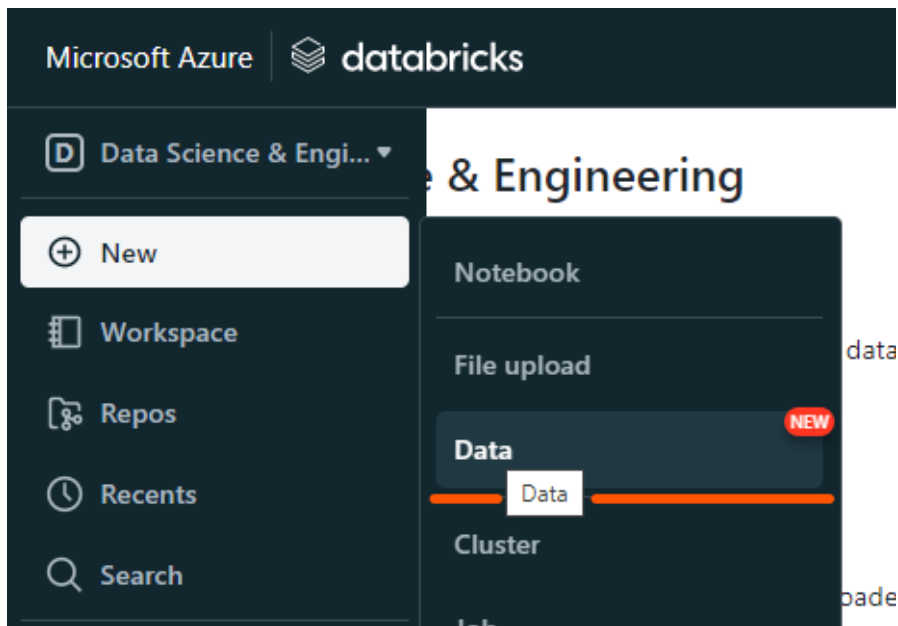


- 3- By default, when you login the workspace to are directly landing to the Data Science & Engineering App page

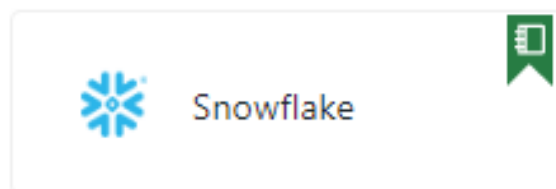


To add a data in Databricks we have several ways to do that. In this article we are going to use the **Add data** page from Data option in the Create menu.

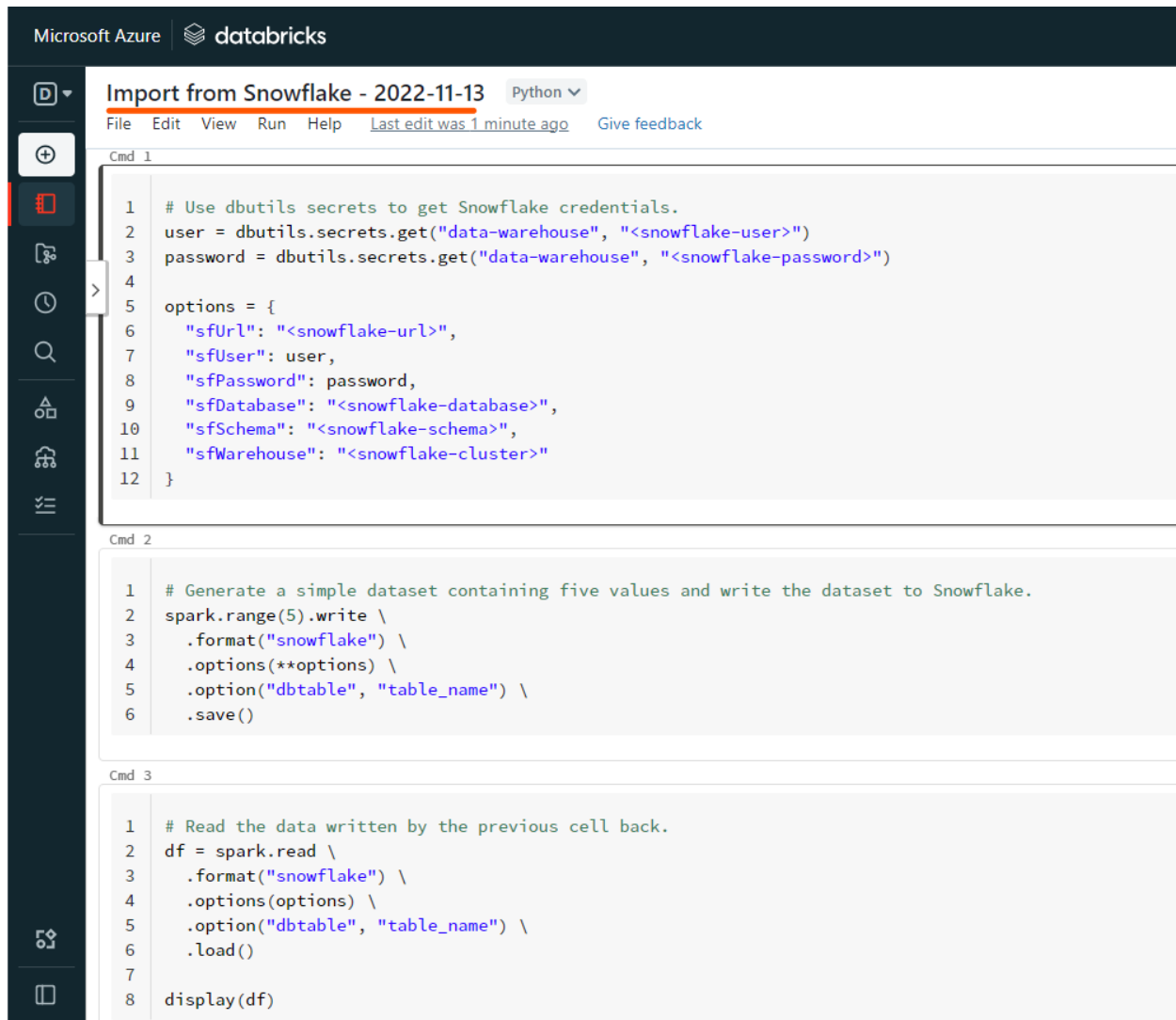
- 4- Click on New on + sign and in the drop list, select Data.
Automatically I have been taking to the Add Data page where I have multiple data sources
Native integrations prebuild notebooks and data ingestion for partners prebuild apps.



5- Select Snowflake prebuild notebook



6- A notebook named Import from Snowflake - Date is created



The screenshot shows a Databricks notebook interface with the title "Import from Snowflake - 2022-11-13" and a Python language selector. The notebook contains three commands:

```
Cmd 1
1 # Use dbutils secrets to get Snowflake credentials.
2 user = dbutils.secrets.get("data-warehouse", "<snowflake-user>")
3 password = dbutils.secrets.get("data-warehouse", "<snowflake-password>")
4
5 options = {
6     "sfUrl": "<snowflake-url>",
7     "sfUser": user,
8     "sfPassword": password,
9     "sfDatabase": "<snowflake-database>",
10    "sfSchema": "<snowflake-schema>",
11    "sfWarehouse": "<snowflake-cluster>"
12 }
```

```
Cmd 2
1 # Generate a simple dataset containing five values and write the dataset to Snowflake.
2 spark.range(5).write \
3     .format("snowflake") \
4     .options(**options) \
5     .option("dbtable", "table_name") \
6     .save()
```

```
Cmd 3
1 # Read the data written by the previous cell back.
2 df = spark.read \
3     .format("snowflake") \
4     .options(options) \
5     .option("dbtable", "table_name") \
6     .load()
7
8 display(df)
```

We have 4 commands in this notebook.

Let see each of them.

The first command allows us to setup connection between Azure Databricks workspace and Snowflake.

To successfully execute this command, you need to have:

Snowflake sysadmin credentials,

information about the Snowflake Datawarehouse like the URL, User, Password, Database name, Schema.

You also need to create a secret scope for Azure Databricks.

```
Cmd 1

1 # Use dbutils secrets to get Snowflake credentials.
2 user = dbutils.secrets.get("snowflake", "snow-user")
3 password = dbutils.secrets.get("snowflake", "snow-password")
4
5 options = {
6     "sfUrl": "https://[REDACTED]canada-central.azure.snowflakecomputing.com/",
7     "sfUser": user,
8     "sfPassword": password,
9     "sfDatabase": "DATABRICKS",
10    "sfSchema": "public",
11    "sfWarehouse": "COMPUTE_WH"
12 }
```

Command took 2.35 seconds -- by ahammamaliki@solutionsmetriques.ca at 11/12/2022, 8:01:48 PM on Spark

The second command generate a simple table with 5 values and write the data into Snowflake. The table is created automatically, and the data loaded for you

```
Cmd 2

1 # Generate a simple dataset containing five values and write the dataset to Snowflake.
2 spark.range(5).write \
3     .format("snowflake") \
4     .options(**options) \
5     .option("dbtable", "PUBLIC.Databricks_table") \
6     .save()
```

The third one read the Data written by the previous command and display the content

```
Cmd 3

1 # Read the data written by the previous cell back.
2 df = spark.read \
3     .format("snowflake") \
4     .options(**options) \
5     .option("dbtable", "Databricks_table") \
6     .load()
```

The fourth command write the data from the Spark Data frame and save it as Delta table

```
Cmd 4

1 # Write the data to a Delta table
2
3 df.write.format("delta").saveAsTable("sf_ingest_snowflake_table")
```

Note: for the purpose of this article, I already created a Database named DATABRICKS in snowflake, also create secret scope for Databricks and Azure key Vault secrets for Snowflake and create Cluster for the notebooks. You need to be at minimum SYSADMIN role in Snowflake.

You can learn how to create a scope for Databricks here: <https://learn.microsoft.com/en-us/azure/databricks/security/secrets/secret-scopes>

And Azure Key Vault Secret here: <https://learn.microsoft.com/en-us/azure/key-vault/general/quick-create-portal?source=recommendations>

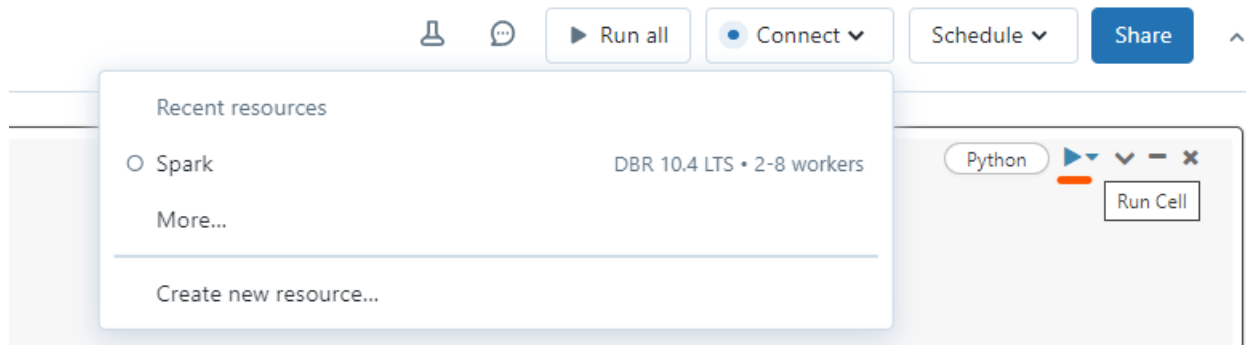
The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes the Microsoft Azure logo and a search bar. The left sidebar shows the 'Secrets' page for a key vault. The main content area displays a table of secrets with columns: Name, Type, and Status. The secrets listed are:

Name	Type	Status
Databricks-ADLS-Key		✓ Enabled
Databricks-Blob-key		✓ Enabled
Databricks-Synapse-Analytics-Token		✓ Enabled
Databricks-Blob-Key		✓ Enabled
Dwh-password		✓ Enabled
Dwh-Username		✓ Enabled
sm-databricks-token-key		✓ Enabled
snow-password		✓ Enabled
snow-user		✓ Enabled

Below the screenshot is a snippet of the Snowflake Databricks interface. The top navigation bar includes the Databricks logo and a search bar. The left sidebar shows the 'Databases' page for the 'DATABRICKS' database. The main content area displays a table with columns: Table Name, Schema, Creation Time, Owner, Rows, Size, and Comment.

Now let's execute the first command

- 1- Select the run cell button to fire up the cluster and execute the first command



2- Click Start, attach, and run

Start compute resource?



Your most recent compute resource **Spark** is currently terminated. Would you like to start and attach to it to run your selected cells?

☐ Automatically launch and attach without prompting

[Select a different compute resource](#)

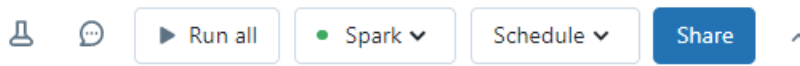
Cancel

Start, attach and run

```
Cmd 1
1 # Use dbutils secrets to get Snowflake credentials.
2 user = dbutils.secrets.get("snowflake", "snow-user")
3 password = dbutils.secrets.get("snowflake", "snow-password")
4
5 options = {
6     "sfUrl": "https://.canada-central.azure.snowflakecomputing.com/",
7     "sfUser": user,
8     "sfPassword": password,
9     "sfDatabase": "DATABRICKS",
10    "sfSchema": "public",
11    "sfWarehouse": "COMPUTE_WH"
12 }
```

Cancel Waiting for cluster to start: Finding instances for new nodes, acquiring more instances if necessary

The cluster is starting at this stage and try to scale out if needed.




```
Cmd 1

1 # Use dbutils secrets to get Snowflake credentials.
2 user = dbutils.secrets.get("snowflake", "snow-user")
3 password = dbutils.secrets.get("snowflake", "snow-password")
4
5 options = {
6     "sfUrl": "https://[redacted].canada-central.azure.snowflakecomputing.com/",
7     "sfUser": user,
8     "sfPassword": password,
9     "sfDatabase": "DATABRICKS",
10    "sfSchema": "public",
11    "sfWarehouse": "COMPUTE_WH"
12 }
```

Command took 2.35 seconds -- by ahammaliki@solutionsmetrics.ca at 11/12/2022, 8:01:48 PM on Spark

The cluster is fire up and the command successfully executed.

3- Same way, click run cell for the second command

```
Cmd 2


1 # Generate a simple dataset containing five values and write the dataset to Snowflake.
2 spark.range(5).write \
3     .format("snowflake") \
4     .options(**options) \
5     .option("dbtable", "PUBLIC.Databricks_table") \
6     .save()
```

▼ (1) Spark Jobs
▼ Job 0 View (Stages: 1/1)
Stage 0: 4/4 ⓘ

Command took 11.02 seconds -- by ahammaliki@solutionsmetrics.ca at 11/12/2022, 8:08:58 PM on Spark

The second command executed successfully.

Now let check in the Snowflake Database the result of this execution. We should have a table named **Databricks_table** under **PUBLIC** schema.



Databases

Shares

Marketplace

Warehouses

Worksheets

History

Databases > DATABRICKS

Tables

Views

Schemas

Stages

File Formats

Sequences

Pipes

⊕ Create...

⊕ Create Like...

📄 Clone...

⬆ Load Data...

✕ Drop...

🔗 Transfer Ownership

Table Name	Schema	Creation Time ▼	Owner	Rows	Size	Comment
<u>DATABRICKS_TABLE</u>	PUBLIC	8:09:06 PM	SYSADMIN	<div><div></div>5</div>	<div><div></div>1KB</div>	

The screenshot shows the Databricks workspace interface. On the left, a sidebar lists database objects under 'DATABRICKS', including 'INFORMATION_SCHEMA', 'PUBLIC', and 'Tables'. The 'Tables' section is expanded, showing 'DATABRICKS_TABLE'. The main area displays a 'New Worksheet' with a 'Run' button and a 'Data Preview' tab. The table 'DATABRICKS_TABLE' is shown with 5 rows and 1 column. The table structure is as follows:

Row
1
2
3
4
5

The table is created with **5 rows** under **PUBLIC** schema

We could see from the Snowflake History tab the detail about the operation

Status	Query ID	SQL Text	User	Warehouse	Size	Session ID	Start Time	End Time	Total Duration	Byt...	Client Info	Rows
This data was last updated at 8:12:49 PM. Press C at the top right to get the latest data. Close												
✓	01a84565...	commit	AHAMMAM...	COMPUTE_WH		2030016321...	8:09:08 PM	8:09:08 PM	59ms		✓ JDBC 3.13.3	
✓	01a84565...	copy into PUBLIC.Databricks_table FROM @spark_connector_load_stage_Lsq25n0F3/YByWPlu3v/...	AHAMMAM...	COMPUTE_WH	X-Small	2030016321...	8:09:06 PM	8:09:06 PM	1.5s	128	✓ JDBC 3.13.3	5
✓	01a84565...	create table if not exists identifier(PUBLIC.Databricks_table) ("ID" INTEGER NOT NULL)	AHAMMAM...	COMPUTE_WH		2030016321...	8:09:06 PM	8:09:06 PM	196ms		✓ JDBC 3.13.3	
✗	01a84565...	desc table identifier(PUBLIC.Databricks_table)	AHAMMAM...	COMPUTE_WH		2030016321...	8:09:06 PM	8:09:06 PM	108ms		✓ JDBC 3.13.3	
✓	01a84565...	after session set search_path='\$current'	AHAMMAM...	COMPUTE_WH		2030016321...	8:09:06 PM	8:09:06 PM	83ms		✓ JDBC 3.13.3	
✓	01a84565...	PUT file:///tmp/dummy_location_spark_connector_tmp/ @spark_connector_load_stage_Lsq25n0F3	AHAMMAM...	COMPUTE_WH		2030016321...	8:09:01 PM	8:09:02 PM	131ms		✓ JDBC 3.13.3	
✓	01a84565...	GET @spark_connector_load_stage_Lsq25n0F3 file:///tmp/dummy_location_spark_connector_tmp/	AHAMMAM...	COMPUTE_WH		2030016321...	8:09:01 PM	8:09:01 PM	141ms		✓ JDBC 3.13.3	
✓	01a84565...	create temporary stage if not exists identifier(spark_connector_load_stage_Lsq25n0F3)	AHAMMAM...	COMPUTE_WH		2030016321...	8:09:01 PM	8:09:01 PM	211ms		✓ JDBC 3.13.3	
✓	01a84565...	after session set timezone = 'Etc/UTC'; timestamp_ntz_output_format = 'YYYY-MM-DD HH24:MI:SS.F...	AHAMMAM...	COMPUTE_WH		2030016321...	8:09:01 PM	8:09:01 PM	84ms		✓ JDBC 3.13.3	
✗	01a84565...	desc table identifier(PUBLIC.Databricks_table)	AHAMMAM...	COMPUTE_WH		2030016321...	8:09:00 PM	8:09:00 PM	53ms		✓ JDBC 3.13.3	
✓	01a84565...	after session set search_path='\$current'	AHAMMAM...	COMPUTE_WH		2030016321...	8:09:00 PM	8:09:00 PM	70ms		✓ JDBC 3.13.3	
✓	01a84548...	SHOW GRANTS TO USER identifier("AHAMMAMALUK")	AHAMMAM...	COMPUTE_WH		2030016321...	7:45:31 PM	7:45:31 PM	67ms		✓ Snowflake UI 202211...	
✓	01a84548...	DROP TABLE "DATABRICKS"."PUBLIC"."DATABRICKS_TABLE";	AHAMMAM...	COMPUTE_WH		2030016321...	7:43:54 PM	7:43:54 PM	74ms		✓ Snowflake UI 202211...	
✓	01a82a85...	CREATE DATABASE SM_DWH_LQA;	AHAMMAM...	COMPUTE_WH		2030016320...	11/8/2022, 6:33...	11/8/2022, 6:33...	170ms		✓ Snowflake UI 202211...	
✓	01a82a85...	CREATE DATABASE SM_DWH_DEV;	AHAMMAM...	COMPUTE_WH		2030016320...	11/8/2022, 6:33...	11/8/2022, 6:33...	188ms		✓ Snowflake UI 202211...	
✓	01a82969...	SELECT "ID" FROM Databricks_table	AHAMMAM...	COMPUTE_WH		2030016320...	11/7/2022, 8:45...	11/7/2022, 8:45...	41ms		✓ JDBC 3.13.3	
✓	01a82969...	after session set timezone = 'Etc/UTC'; timestamp_ntz_output_format = 'YYYY-MM-DD HH24:MI:SS.F...	AHAMMAM...	COMPUTE_WH		2030016320...	11/7/2022, 8:45...	11/7/2022, 8:45...	60ms		✓ JDBC 3.13.3	
✓	01a82969...	SP1 FCT "ID" FROM Databricks_table	AHAMMAM...	COMPUTE_WH		2030016320...	11/7/2022, 8:45...	11/7/2022, 8:45...	191ms		✓ JDBC 3.13.3	

- Read the same table from snowflake and create a Delta table

In this section we are going to reload the table we have created in snowflake and save it as Delta table into Azure Databricks Lakehouse.

- 1- From the notebook let's execute the third command

```
Cmd 3
1 # Read the data written by the previous cell back.
2 df = spark.read \
3   .format("snowflake") \
4   .options(**options) \
5   .option("dbtable", "Databricks_table") \
6   .load()
7 display(df)
```

▶ (1) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [ID: decimal(38,0)]

Table ▾ +

	ID
1	1
2	0
3	2
4	3
5	4

Showing all 5 rows. | 3.46 seconds runtime

Command took 3.46 seconds -- by ahammaliki@solutionsmetriques.ca at 11/12/2022, 8:21:50 PM on Spark

The command executed successfully with the 5 rows read from Snowflake table.

2- Execute the last command to save the data frame as a delta table

```
Cmd 4
1 # Write the data to a Delta table
2
3 df.write.format("delta").saveAsTable("sf_ingest_snowflake_table")
```

▼ (4) Spark Jobs

- ▶ Job 2 [View](#) (Stages: 1/1)
- ▶ Job 3 [View](#) (Stages: 1/1)
- ▶ Job 4 [View](#) (Stages: 1/1, 1 skipped)
- ▶ Job 5 [View](#) (Stages: 1/1, 2 skipped)

Command took 15.71 seconds -- by ahammaliki@solutionsmetriques.ca at 11/12/2022, 8:26:43 PM on Spark

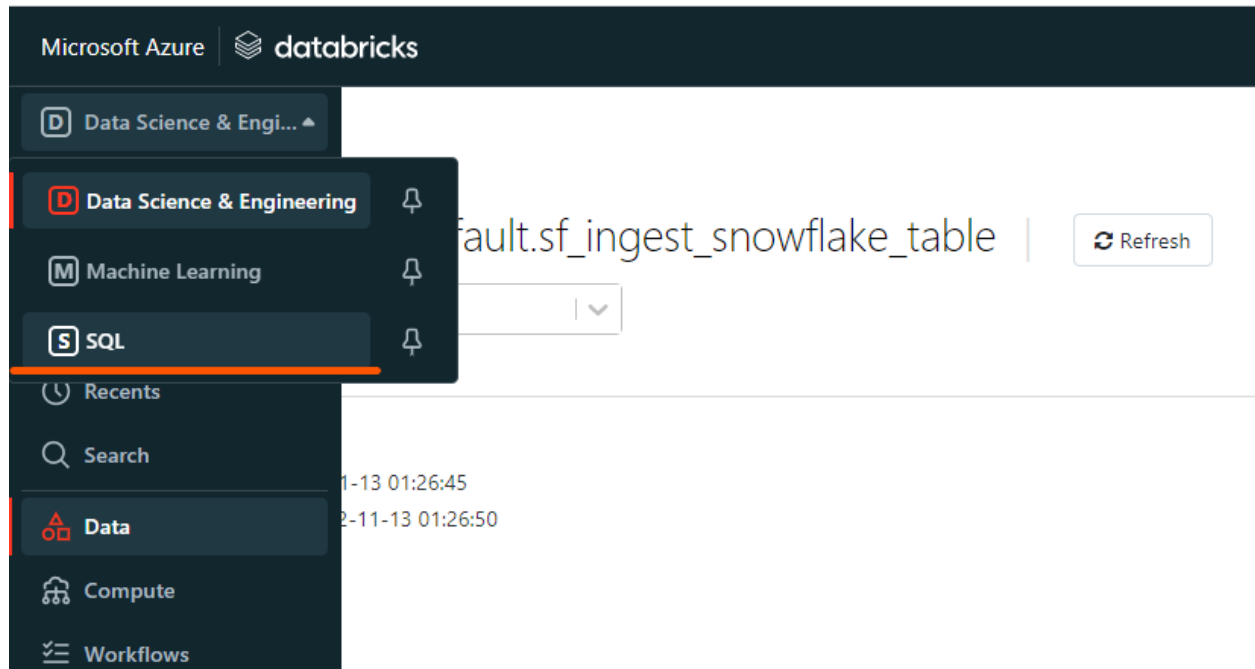
The command executed.

Now that we have our table created and saved as delta table, we can access to it through the Data menu or share it with the data analyst through and SQL endpoint.

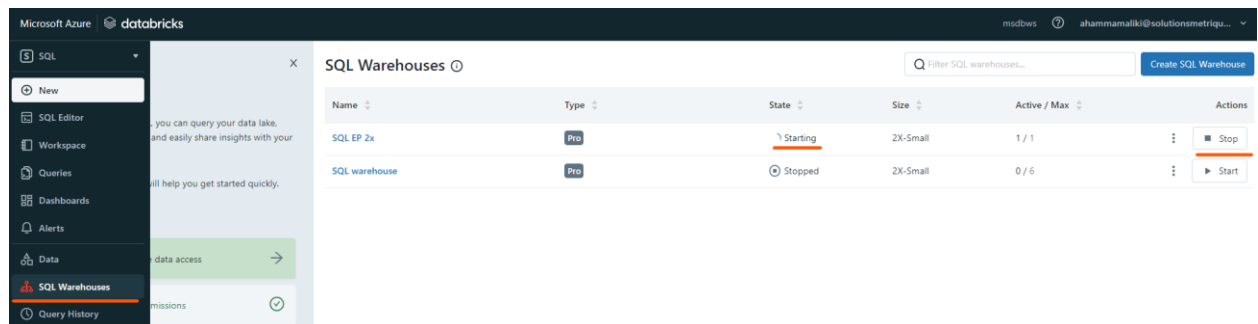
3- Navigate and select Data menu bouton

- Querying the same table by using the Databricks SQL app.

1- Use the App switcher and select **Databricks SQL app**



2- From the Azure Databricks SQL App page click on SQL Warehouses to start the cluster



SQL EP 2x

Overview		Connection details	Monitoring
Status	✓ Running		
Name	SQL EP 2x (ID: 45f696e2ff3c01e4)		
Type	Pro		
Cluster size	2X-Small		
Auto stop	After 15 minutes of inactivity		
Scaling	Cluster count: Active 1 Min 1 Max 1		
Channel	Preview (v 2022.35)		
Spot instance policy	Cost optimized		
Created by	ahammamaliki@solutionsmetriques.ca		

Now that we have the confirmation let's switch to the Data Explorer page.

3- From the Azure Databricks SQL App page click on Data menu.

The table below was created through a notebook and stored as a delta table.

Data Explorer

hive_metastore > default

Filter tables...

sf_ingest_snowflake_table

default

Schema Not set Hide comment

Default Hive database

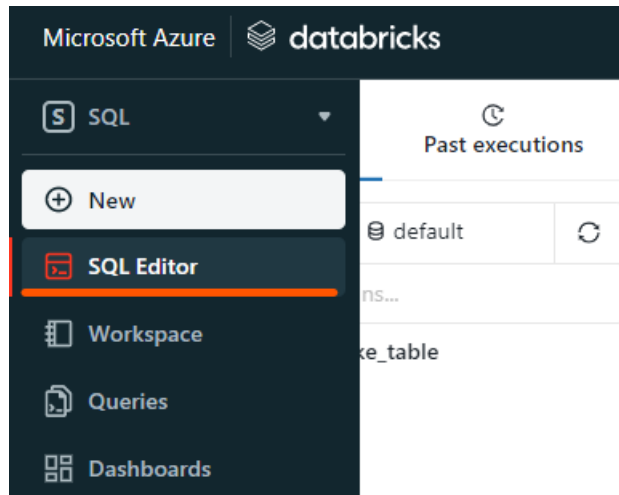
Tables Details Permissions

1 Tables

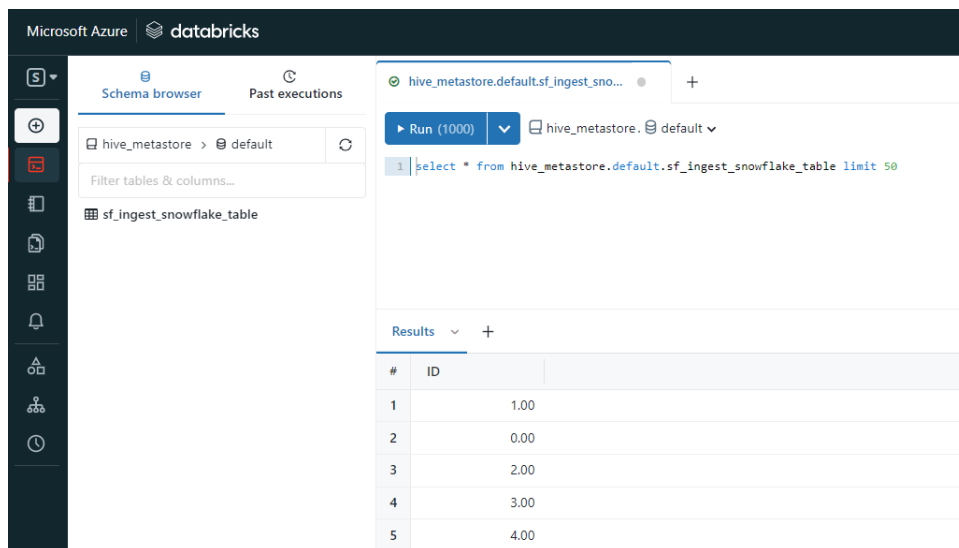
Name

sf_ingest_snowflake_table

4- Now select the SQL Editor menu



5- Run the Query



Conclusion:

Azure Databricks **Add data** (mode preview) feature page have pre-build notebooks and pre-build data ingestion apps that you can use to fast Ingest, Extract Transform and Load data into your Organisation Data platform. It is also helps you applying the best practices and having a easy hands on to the Azure Databricks workspace.