**CST8921 – Cloud Industry Trends**

**Lab 4 – Azure Databricks and analyzing files with Databricks**

## Introduction

Azure Databricks is a platform that analyzes, manages and processes large-scale data workloads with Apache Spark. Apache Spark is a data processing engine that excels at processing data at break-neck speeds as well as providing API for major languages such as Scala, Java, Python and R. Spark can be used to process data within any major application or warehouse provider that requires big data processing such as analytics and machine learning.

Within this lab, I created a Databricks workspace while exploring the creation of node clusters and notebooks, in addition to managing/displaying data from a sample CSV file. For the tutorial, I chose a smaller CSV file of 100 customers to start with, as the goal was to experiment and explore Databricks. Within Part 2 of the lab, I explore Delta Lakes within Azure Databricks and compare external and managed tables. The lab also shows how to organize streaming data into delta tables. Part 3 of the lab is creating a data factory pipeline and executing a Databricks notebook from the pipeline.

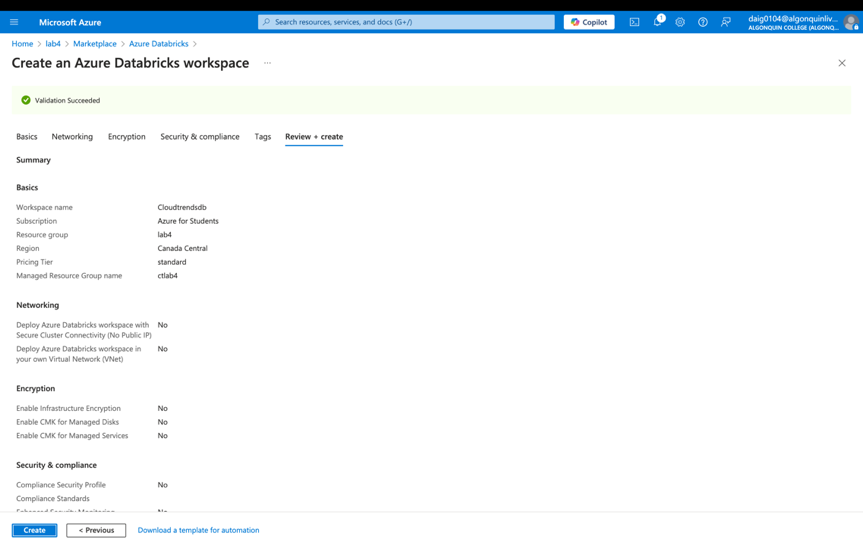
## Objective

The Objective is to familiarize myself with Azure Databricks, and Azure Data Factory to analyze, interact and transform data sources.

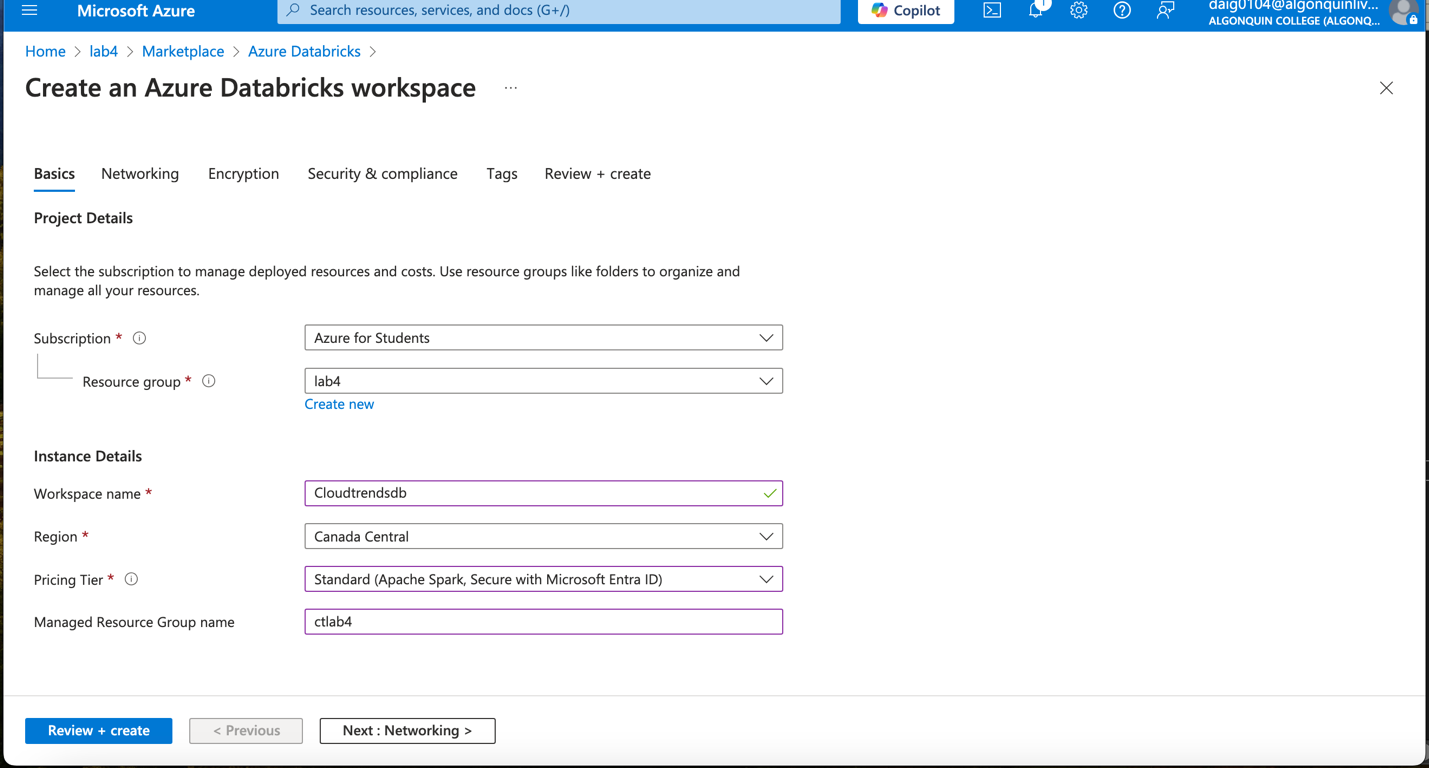
# Lab Activity Screenshots

## Part 1: Explore Databricks notebooks

1. Provision Azure and create Databricks workspace



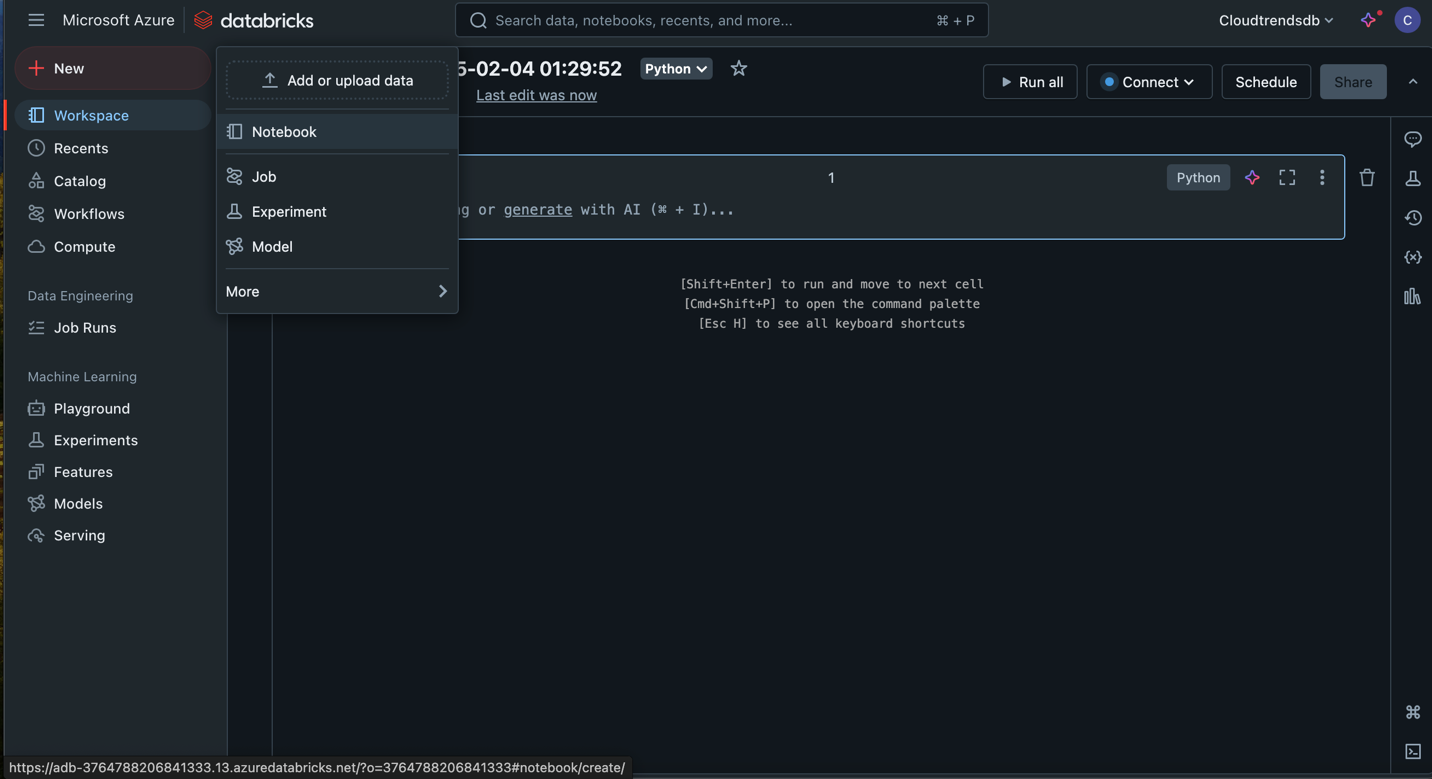
1. Azure Databricks Review:



1. Creating a single node cluster in the workspace I picked F4 for cheapest option.



1. creating a notebook to explore & analyze data data



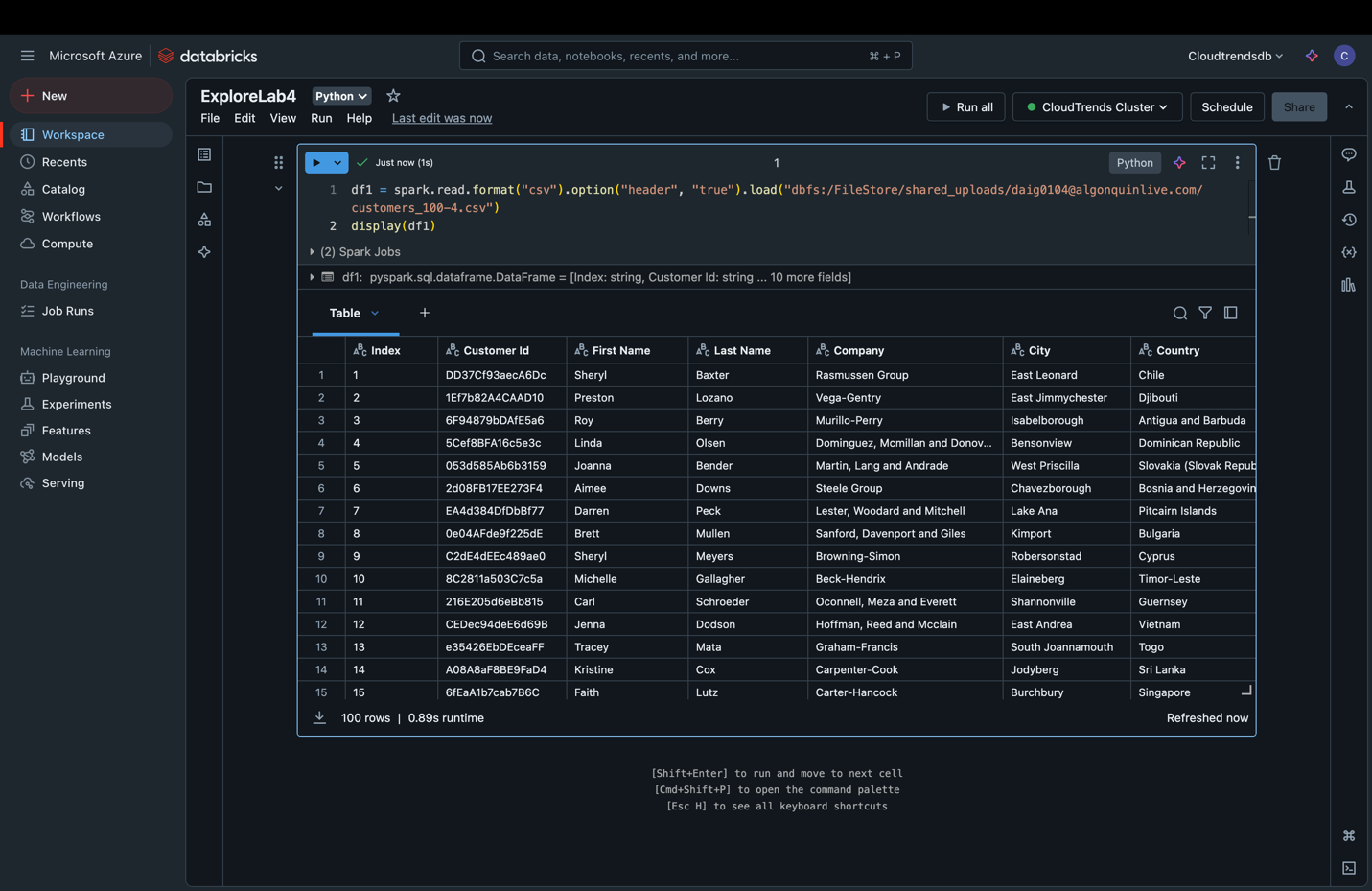
1. Uploading the file to the DBFS directory in the workspace.



1. Copying the sample PySpark code to the clipboard. Changing the file name from products to the file name uploaded to dbfs.

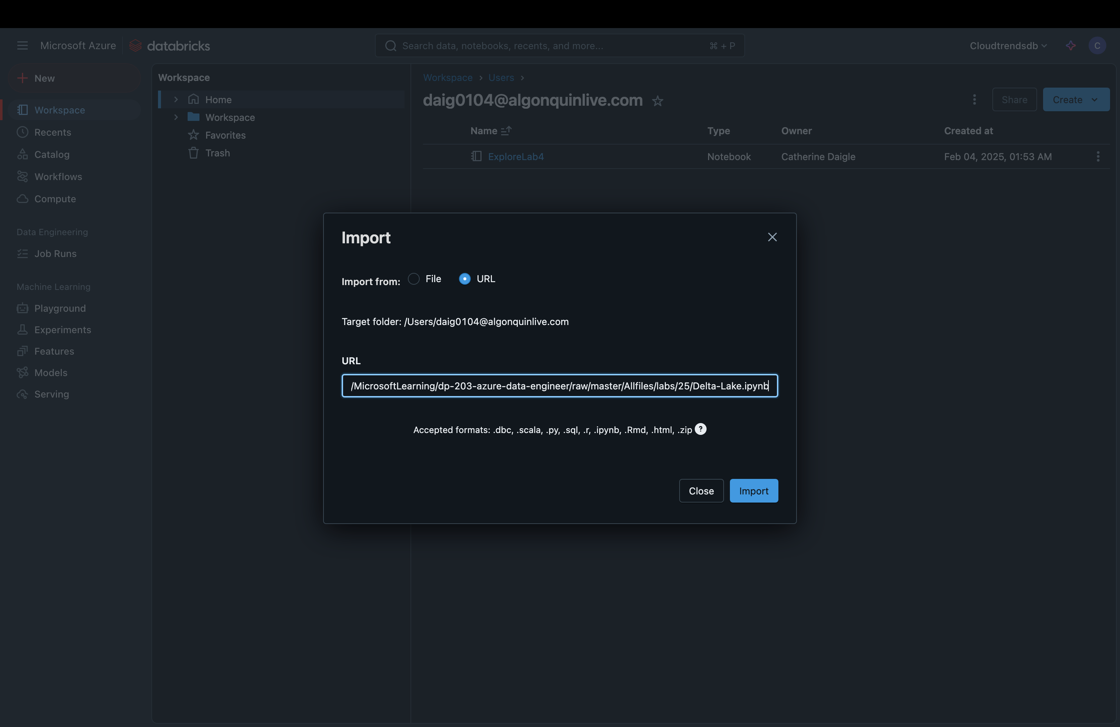


1. Displaying the data with display(df1)

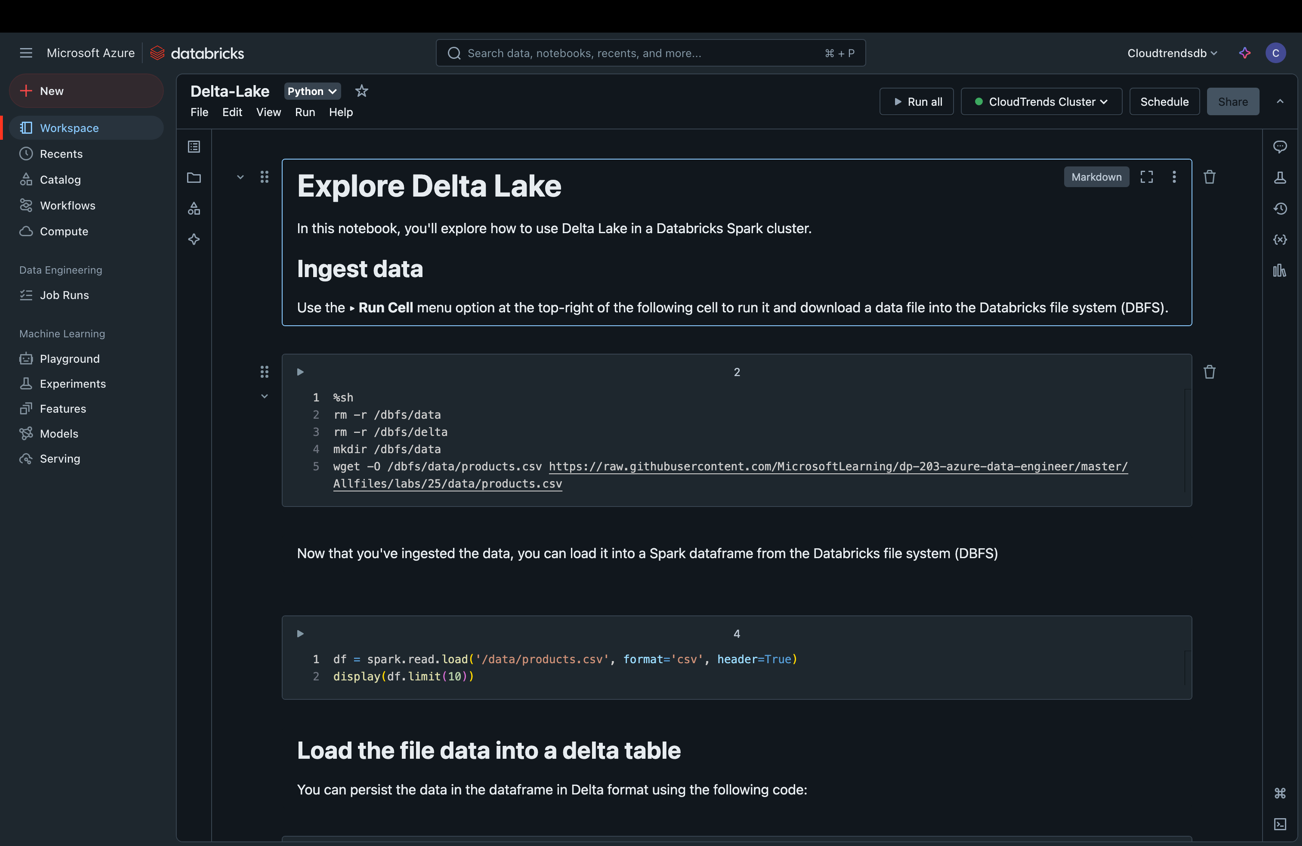


**Part 2: Use Delta Lake in Azure Databricks**

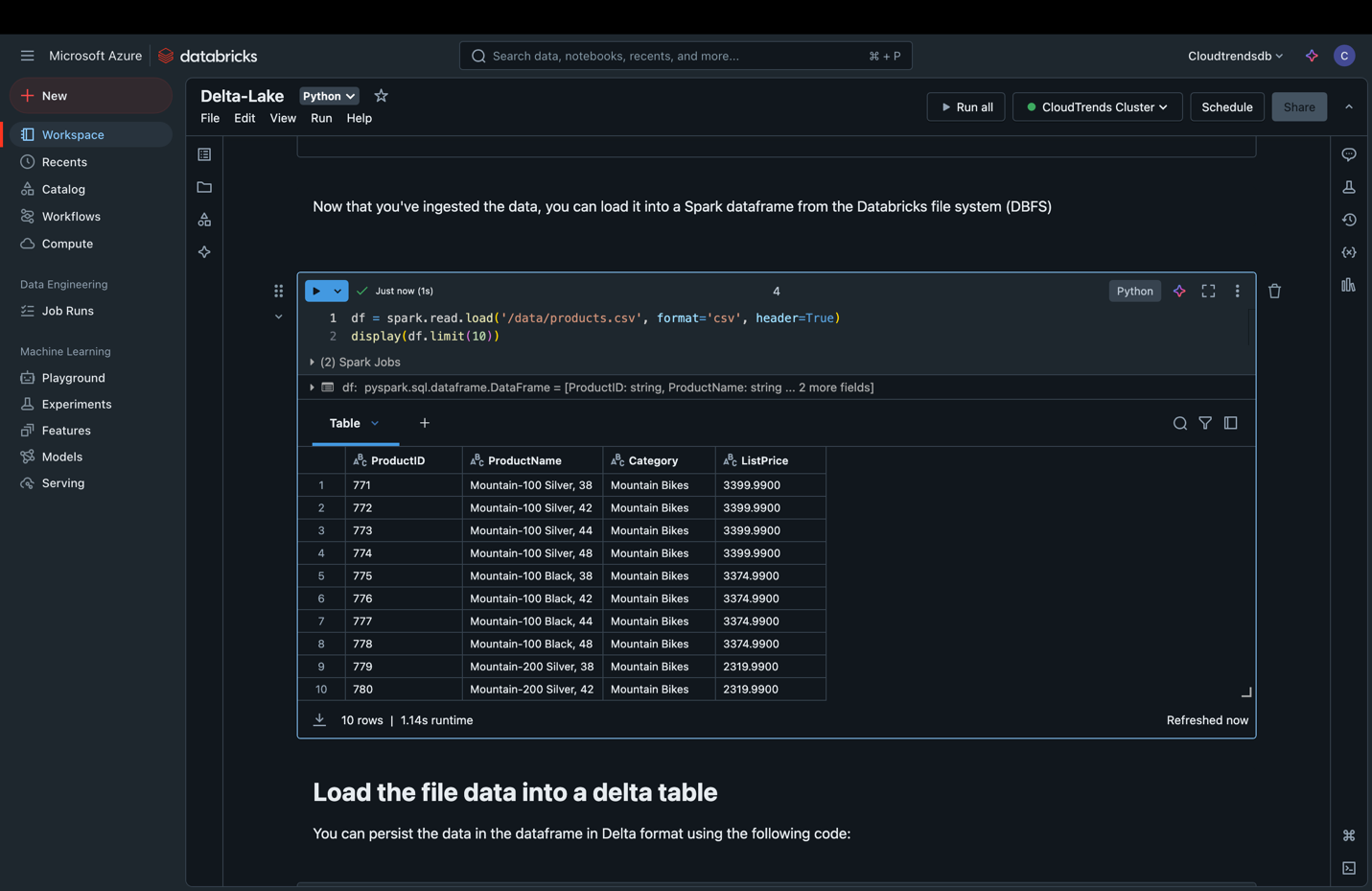
1. Importing github URL to the home notebook



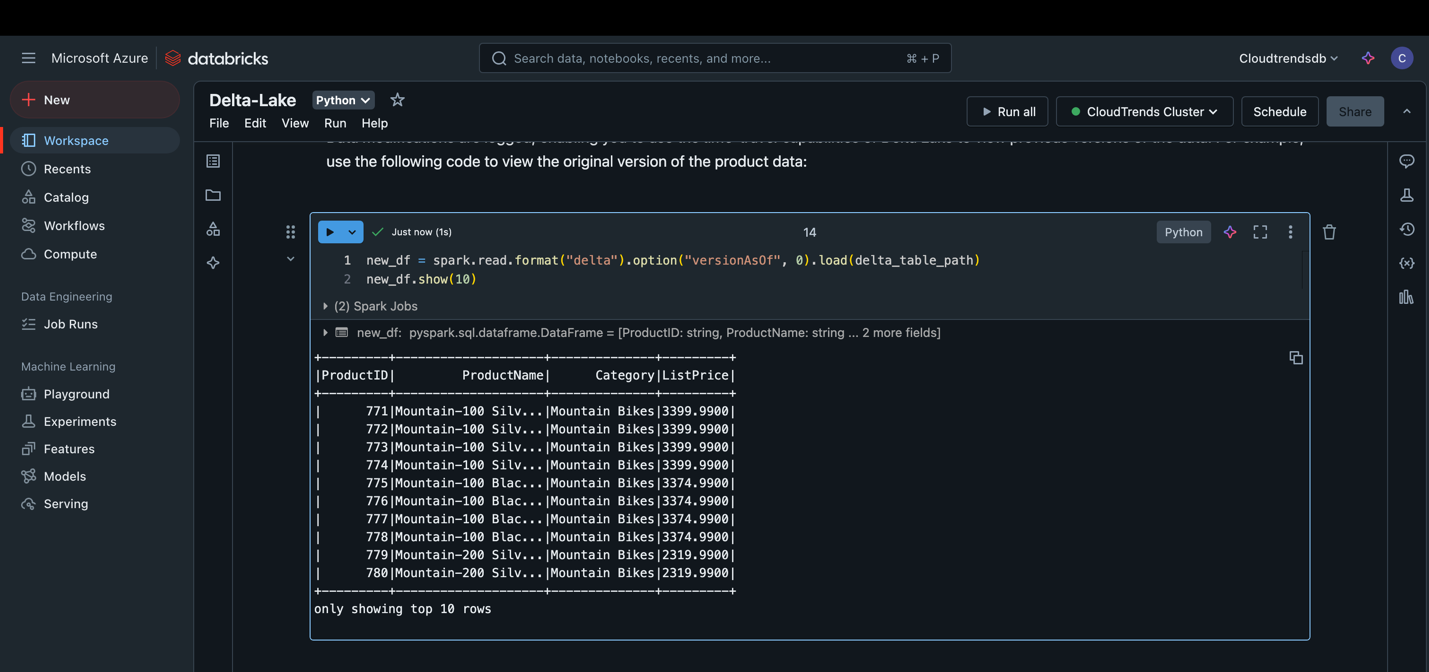
1. Connecting the notebook to the cluster (top right, CloudTrends Cluster), and running the cells to explore delta lake functionality.



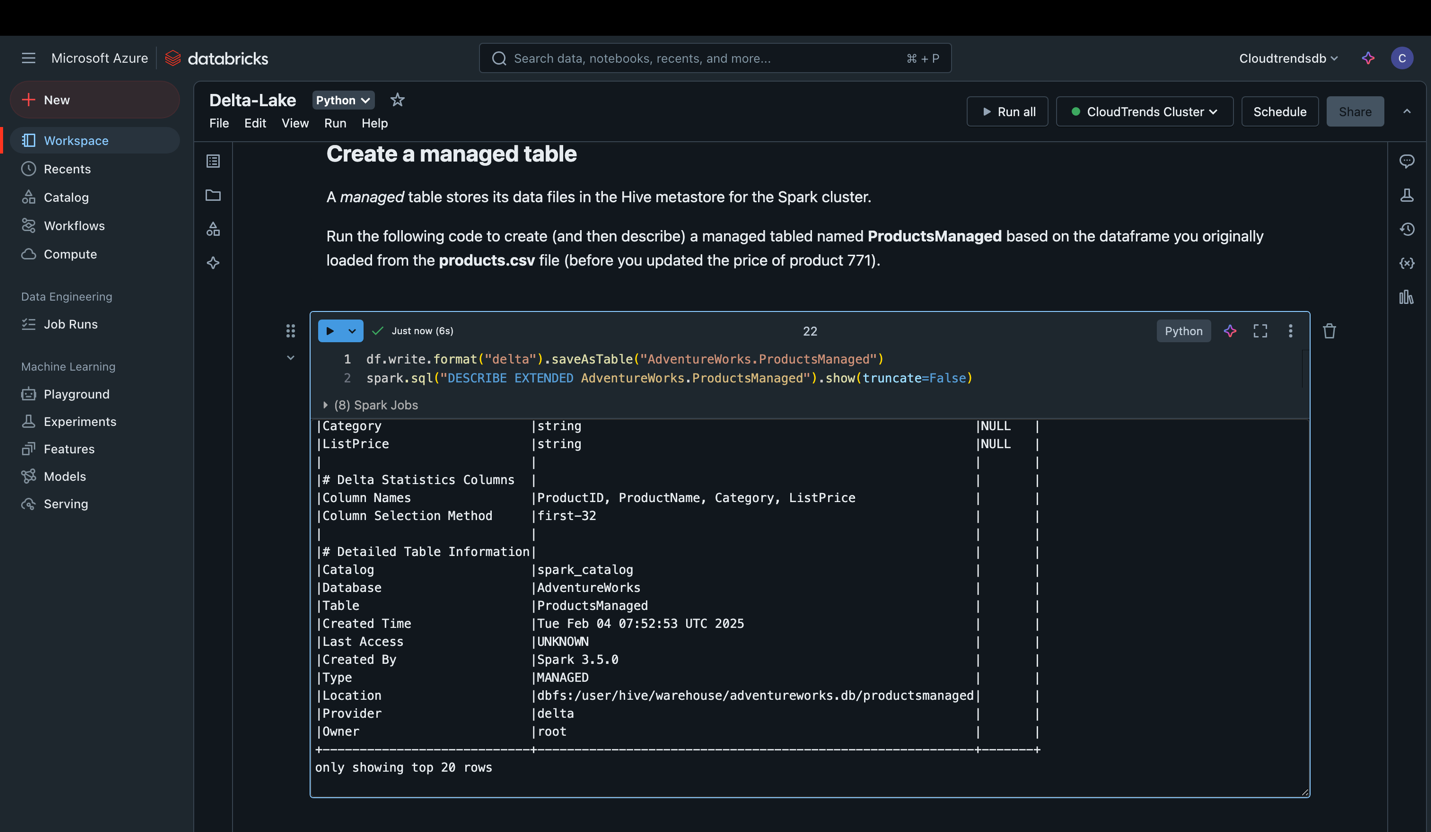
1. Loading data into a spark dataframe:



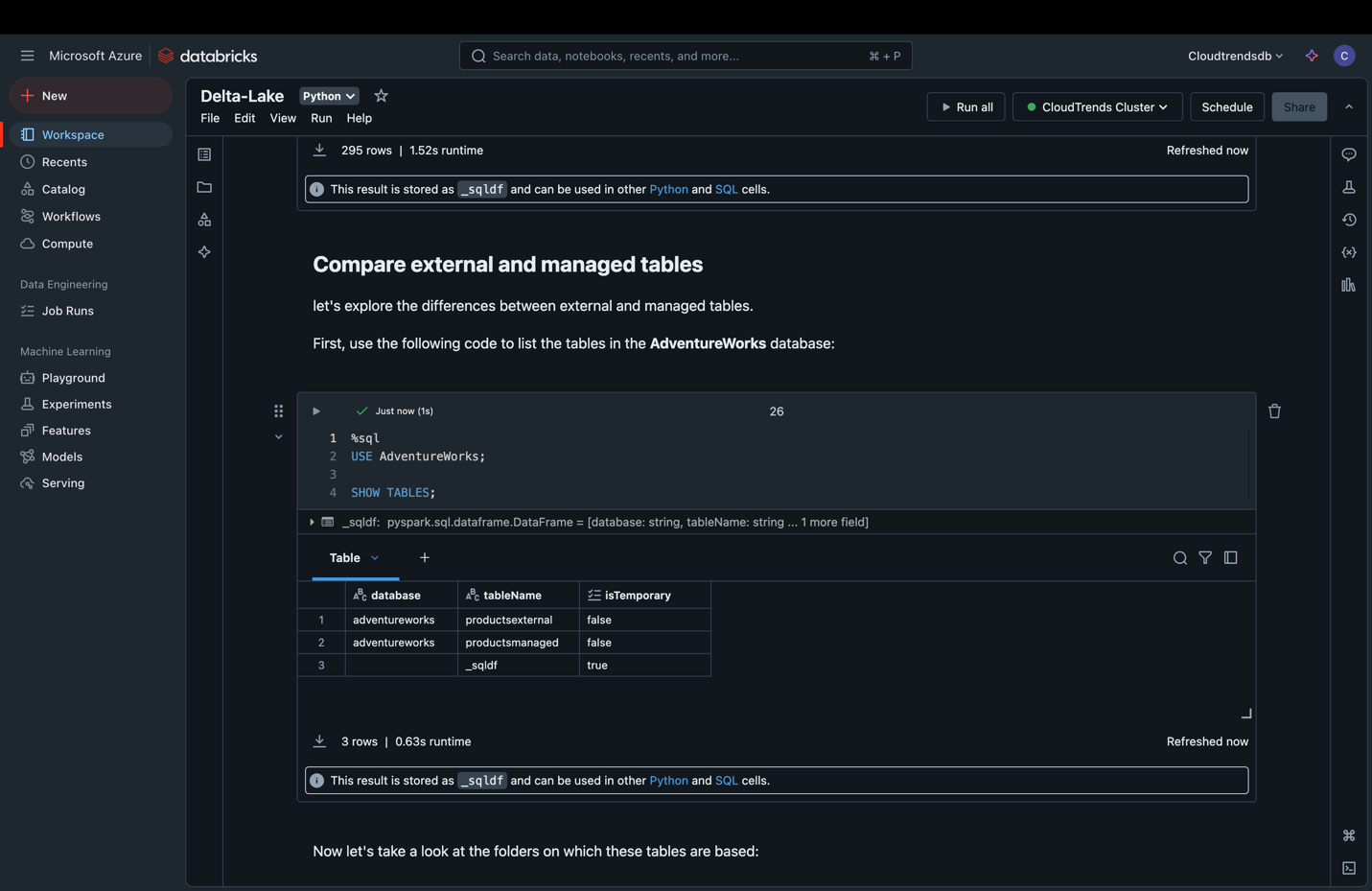
1. Original version of product data.



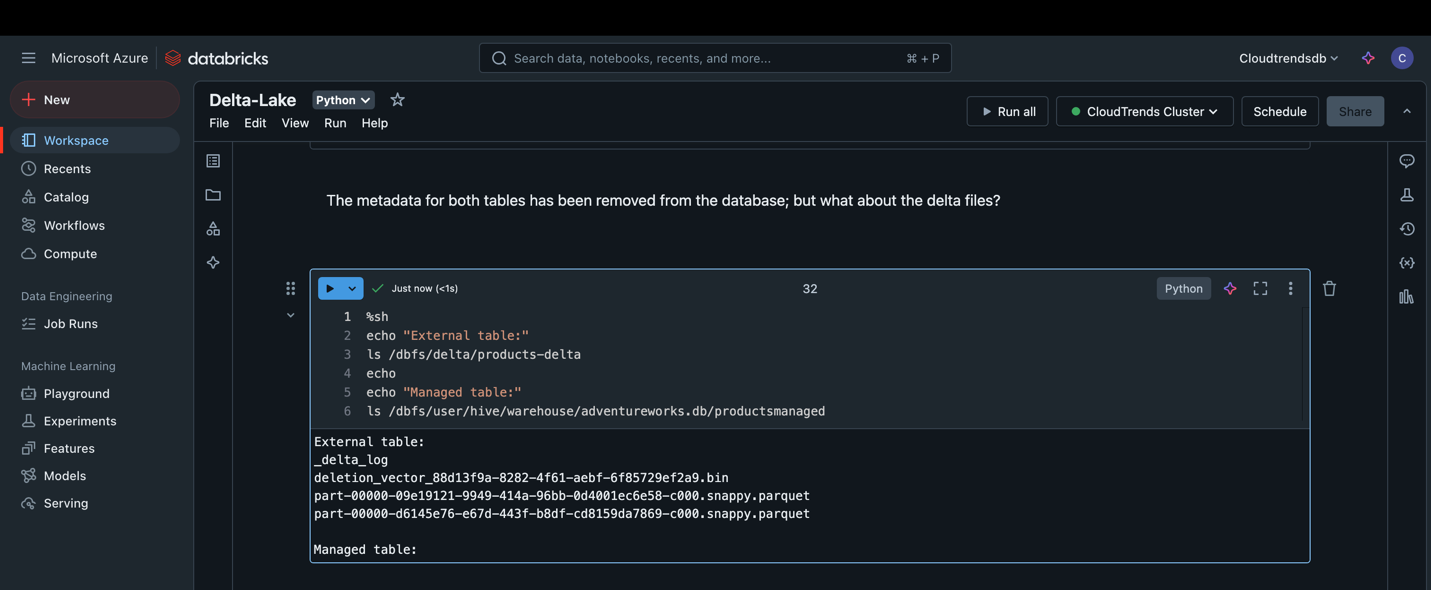
1. Creating a managed table



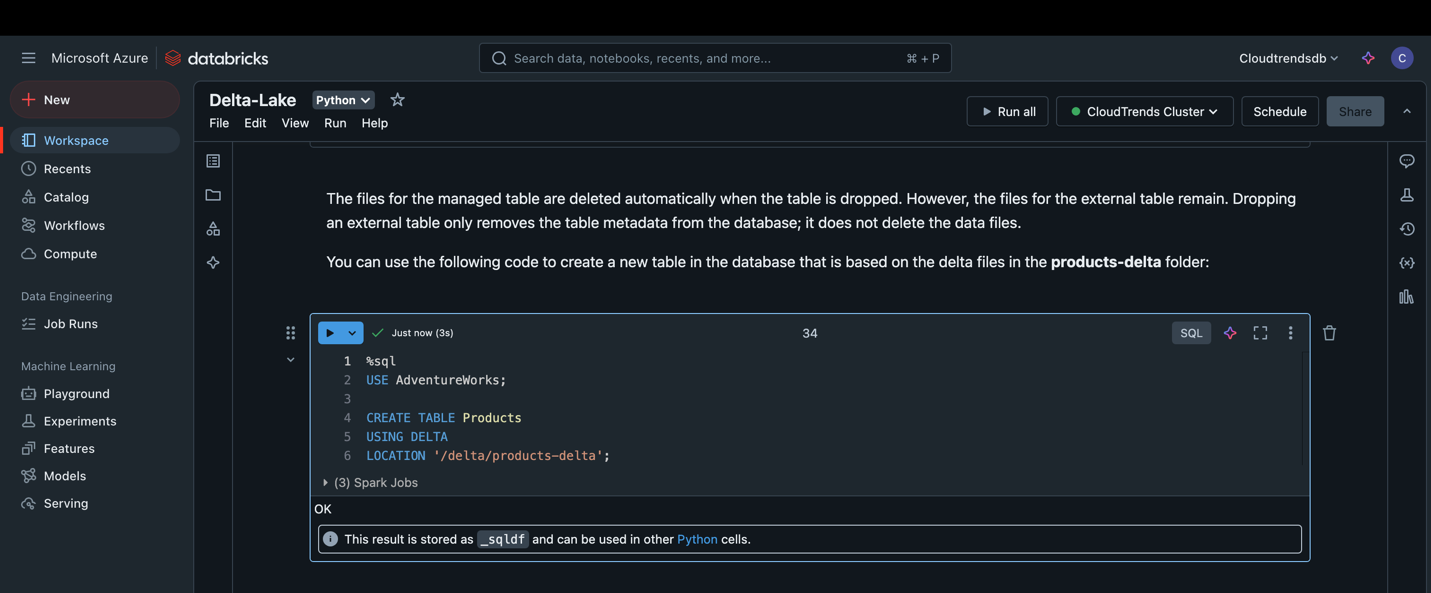
1. Comparing external and managed tables,



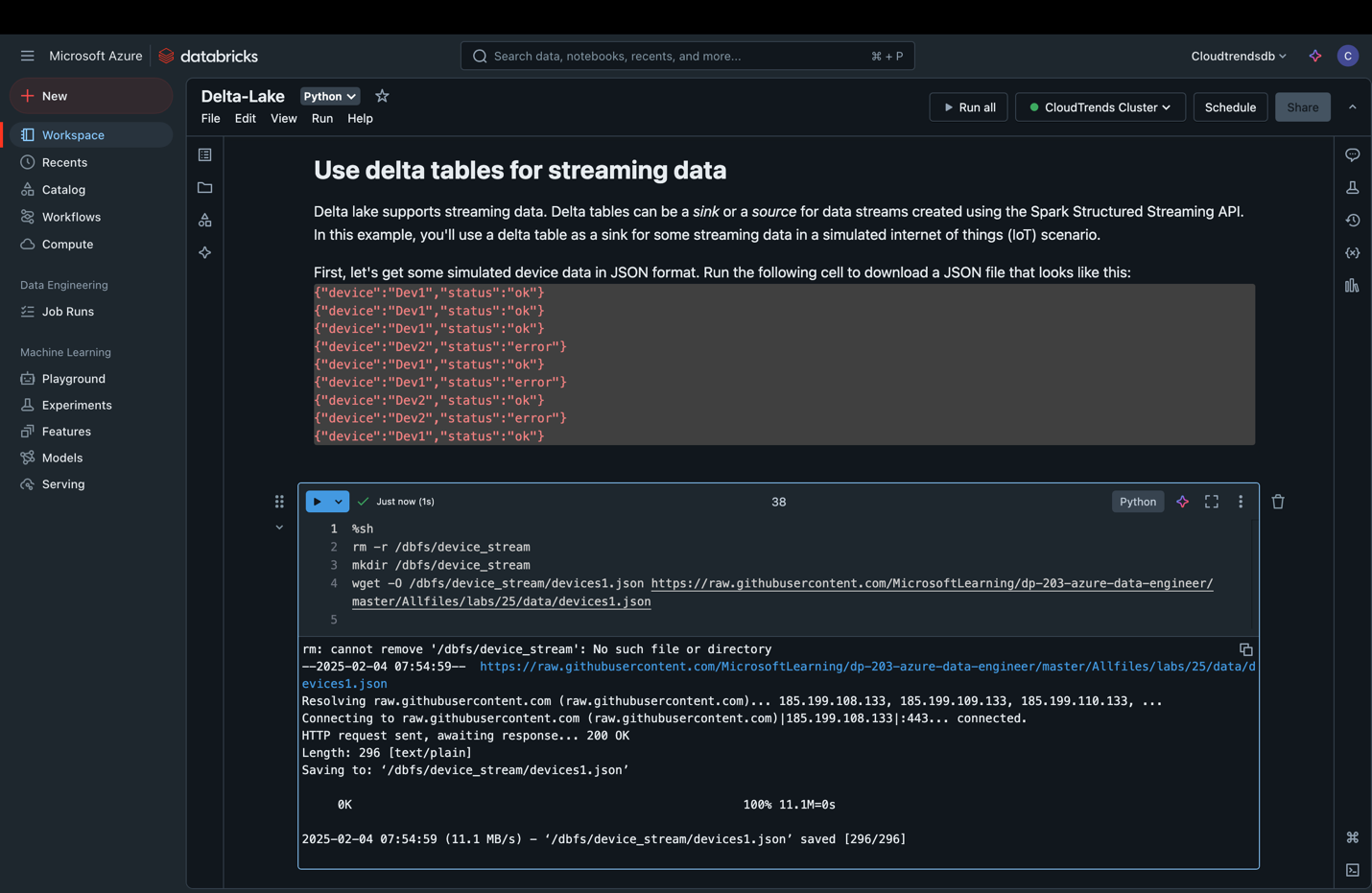
1. Checking metadata for both tables which have been removed when the tables were dropped



1. Checking the delta files, where managed tables are deleted but external table remain.



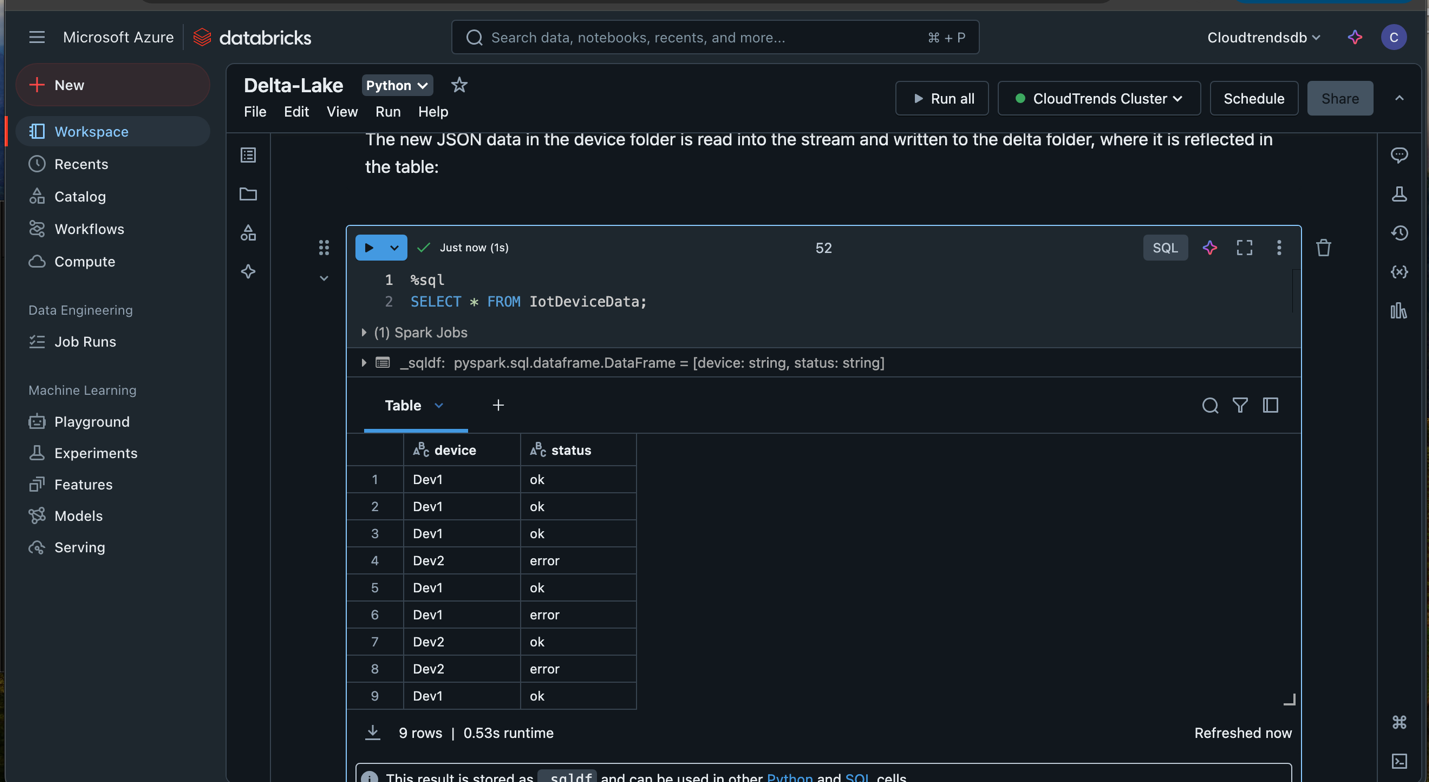
1. Using delta tables for streaming data



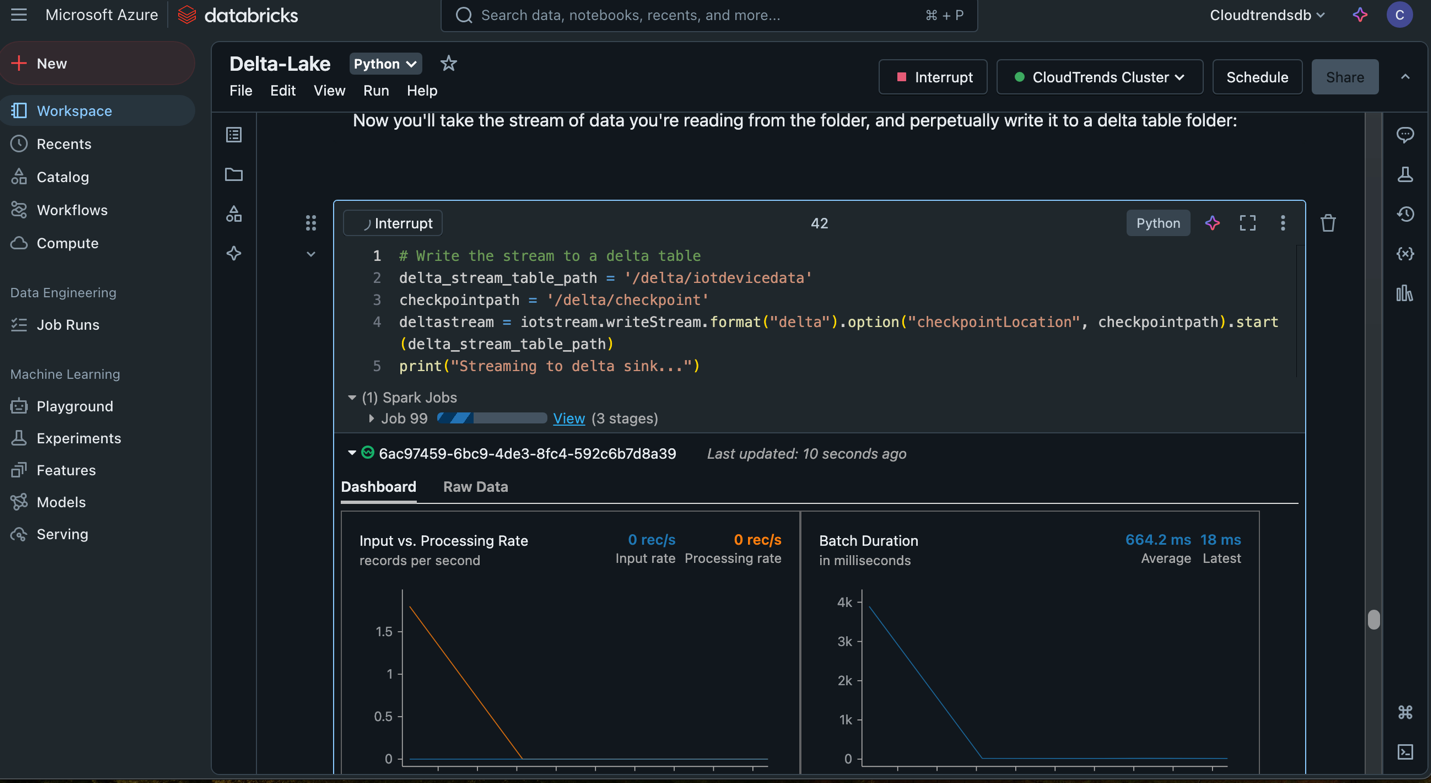
1. Creating source stream and using spark structured streaming



1. Reading the JSON data within the stream and apply the data to a table

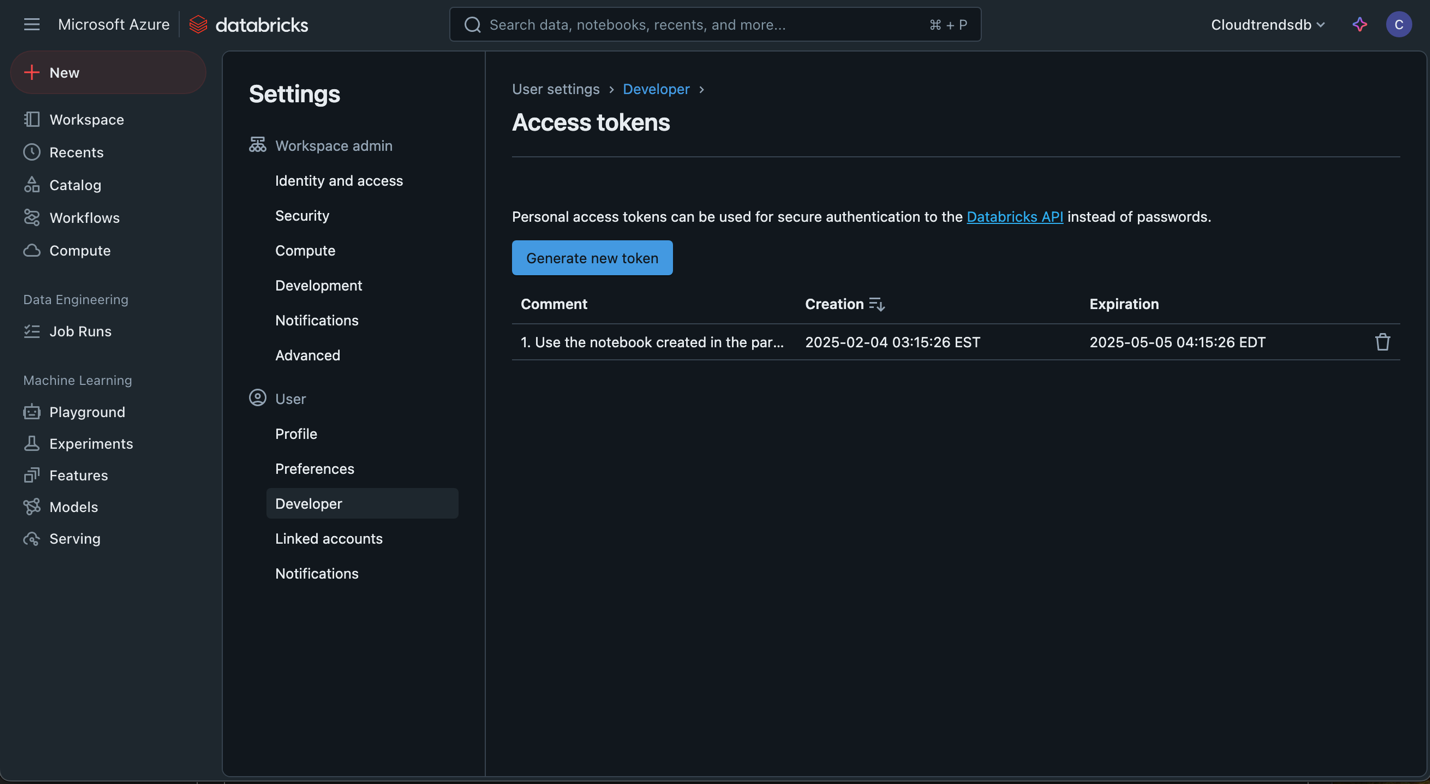


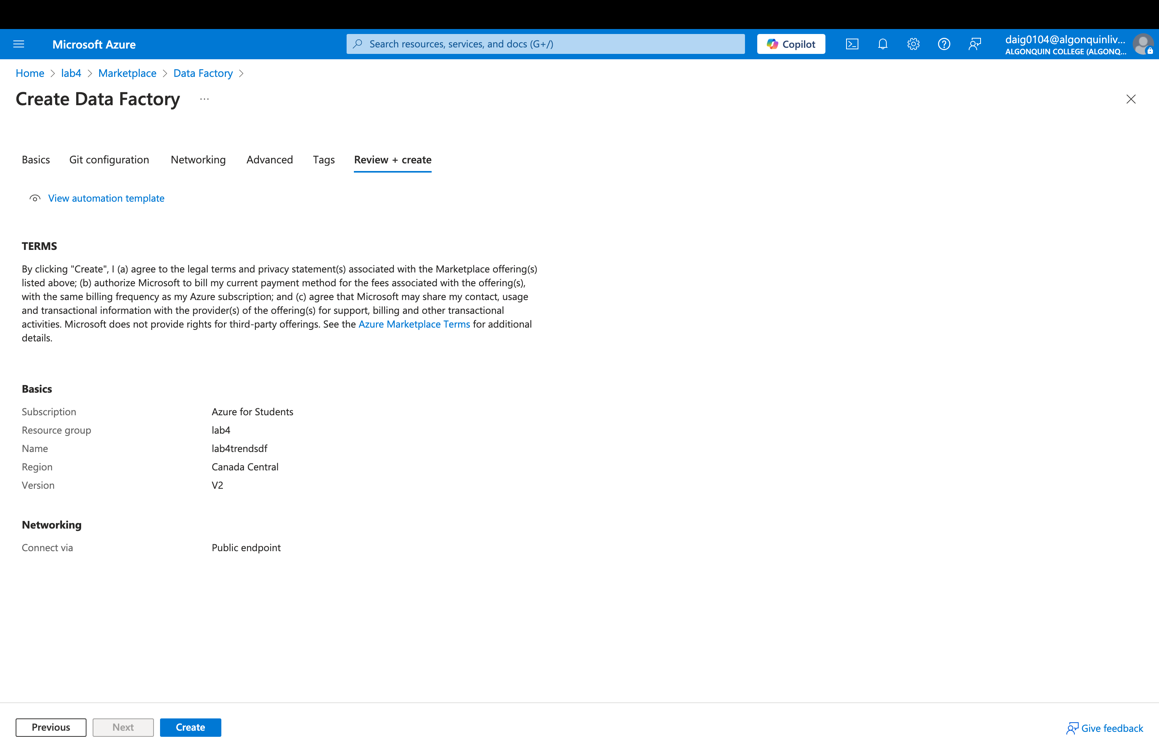
1. Perpetually write the stream to a delta table folder.

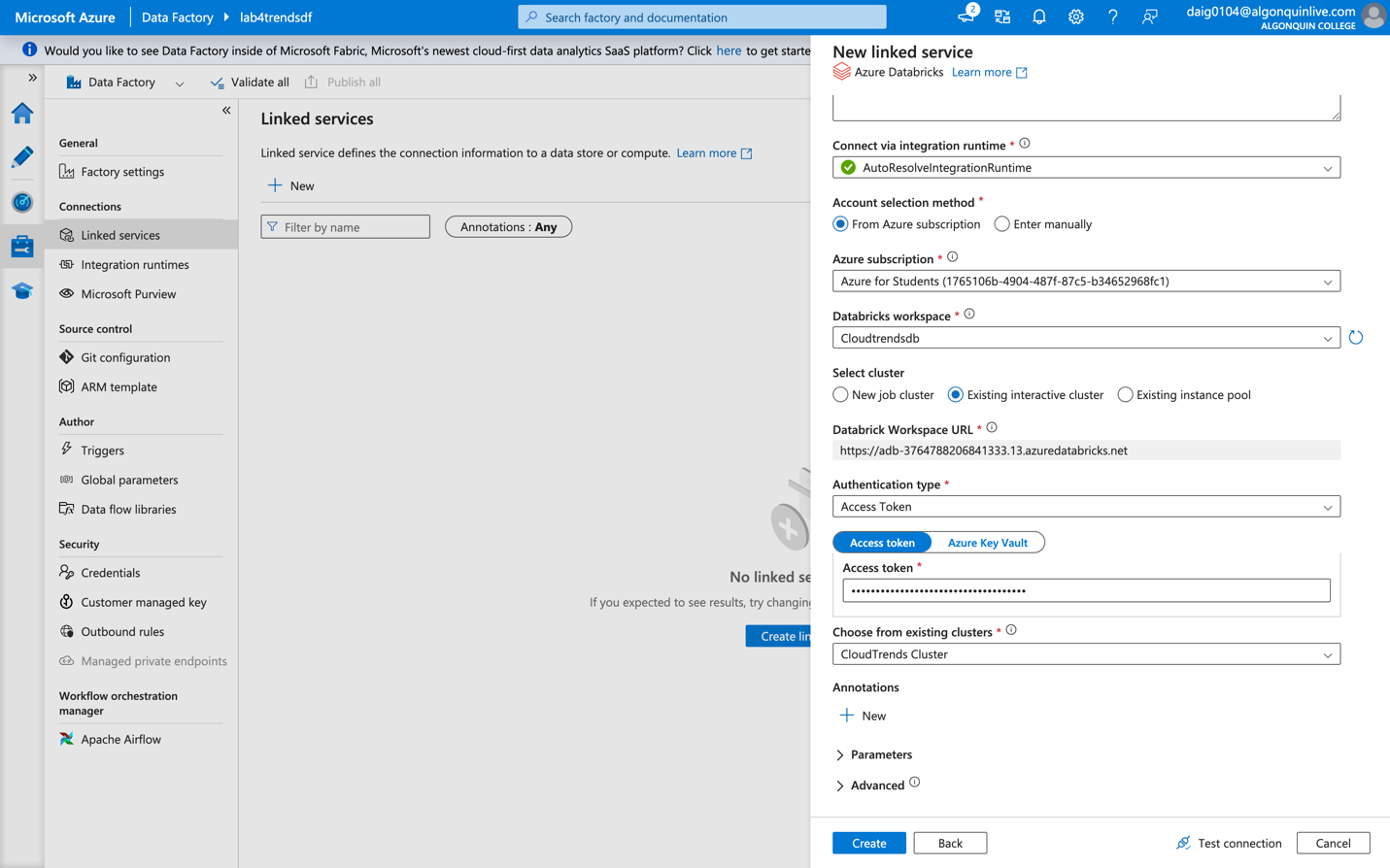


**Part 3: Execute Databricks notebook from Azure data factory**

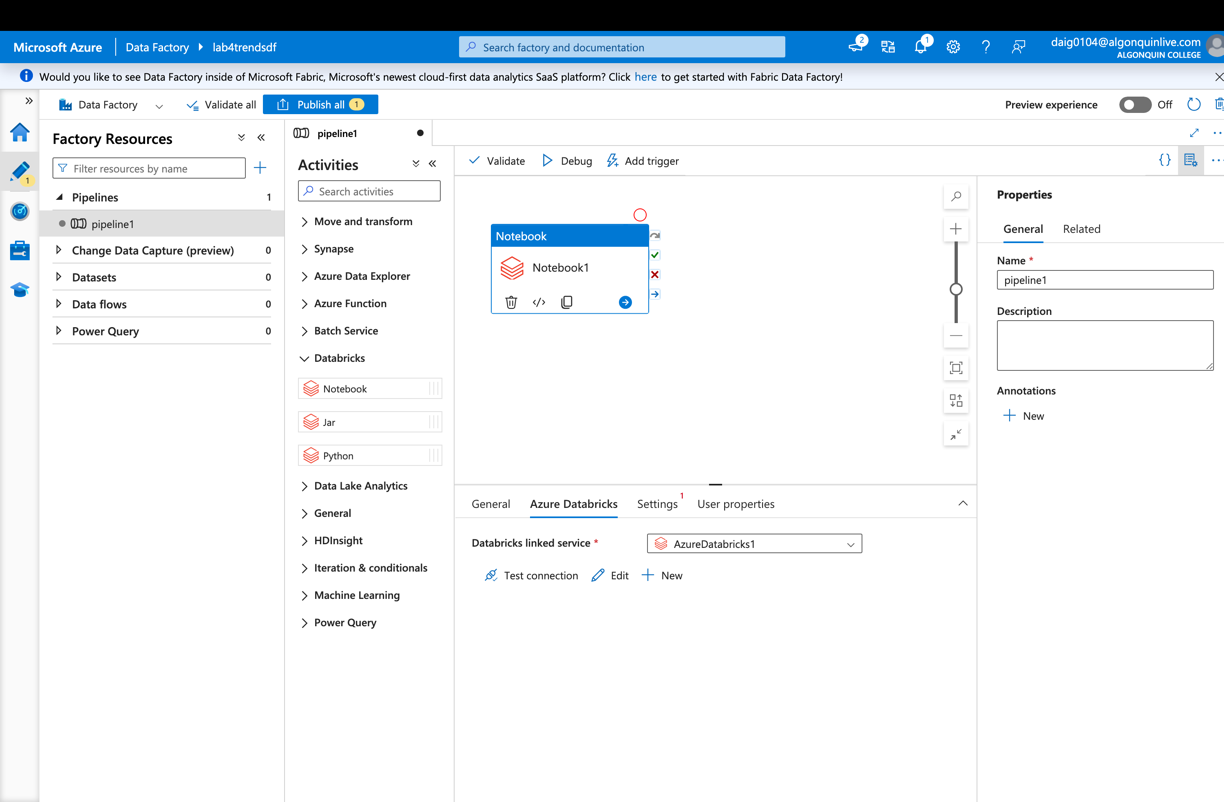
1. generating access token from databricks notebook from part 1. The access token was: dapib3895628afd7448d118d0adab43e8059.



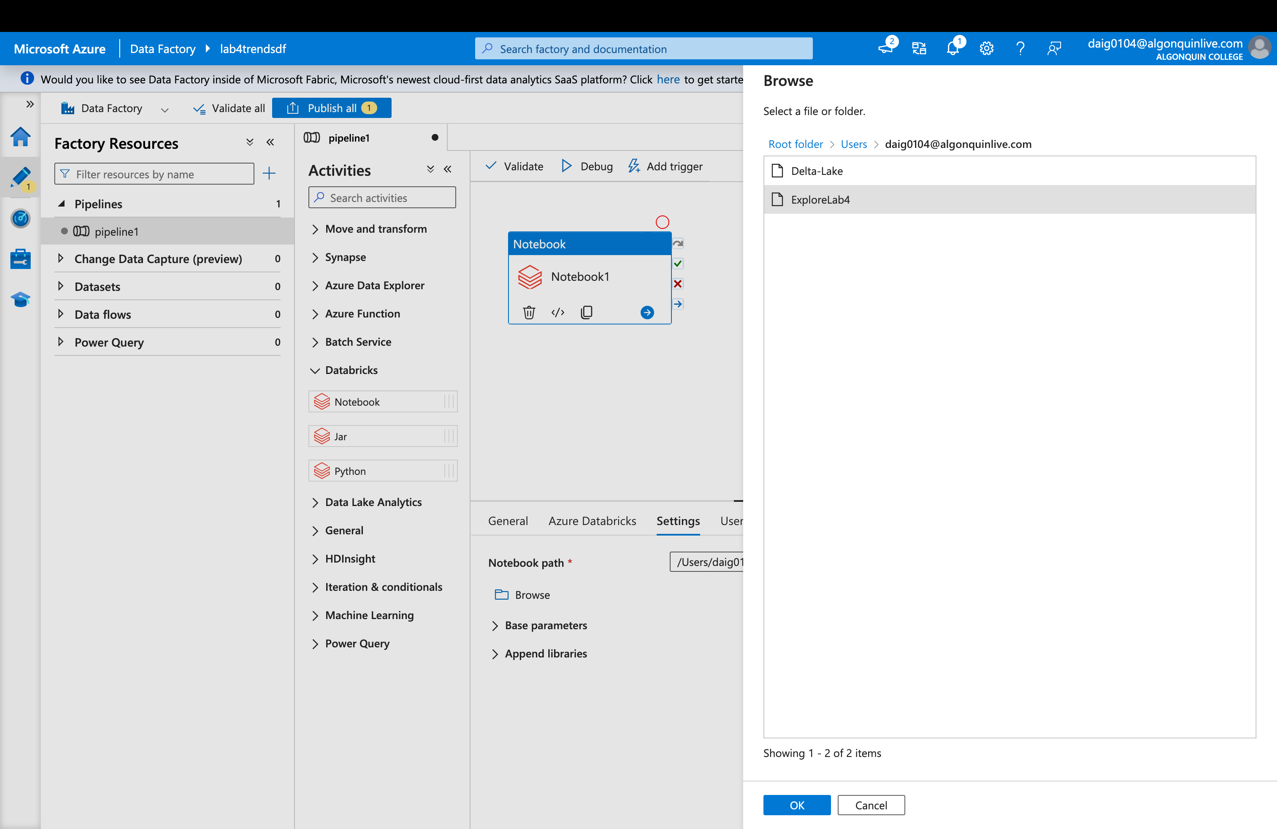
1. creating azure data factory review
2. Creating a linked service to the Databricks workspace and inputting the access token from the steps before.



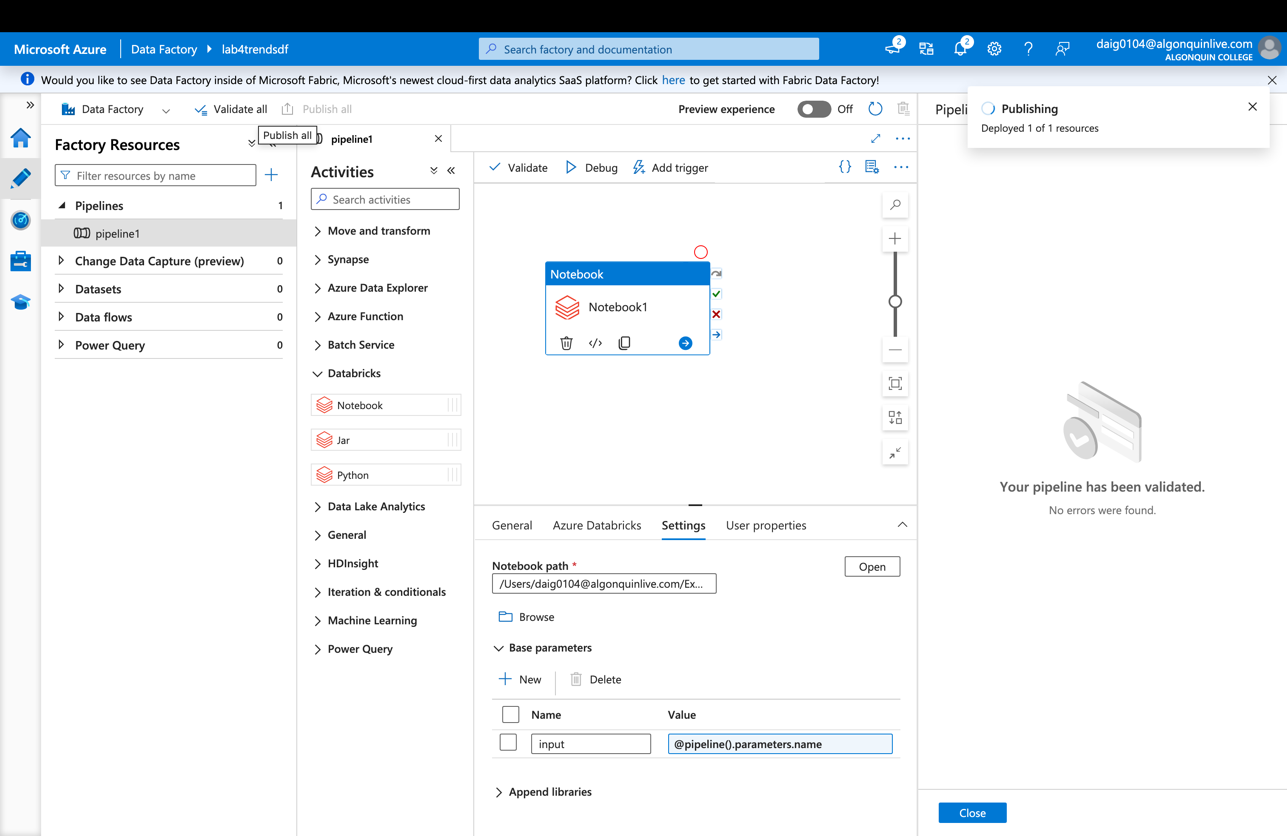
1. Create a pipeline to run the notebook from data factory



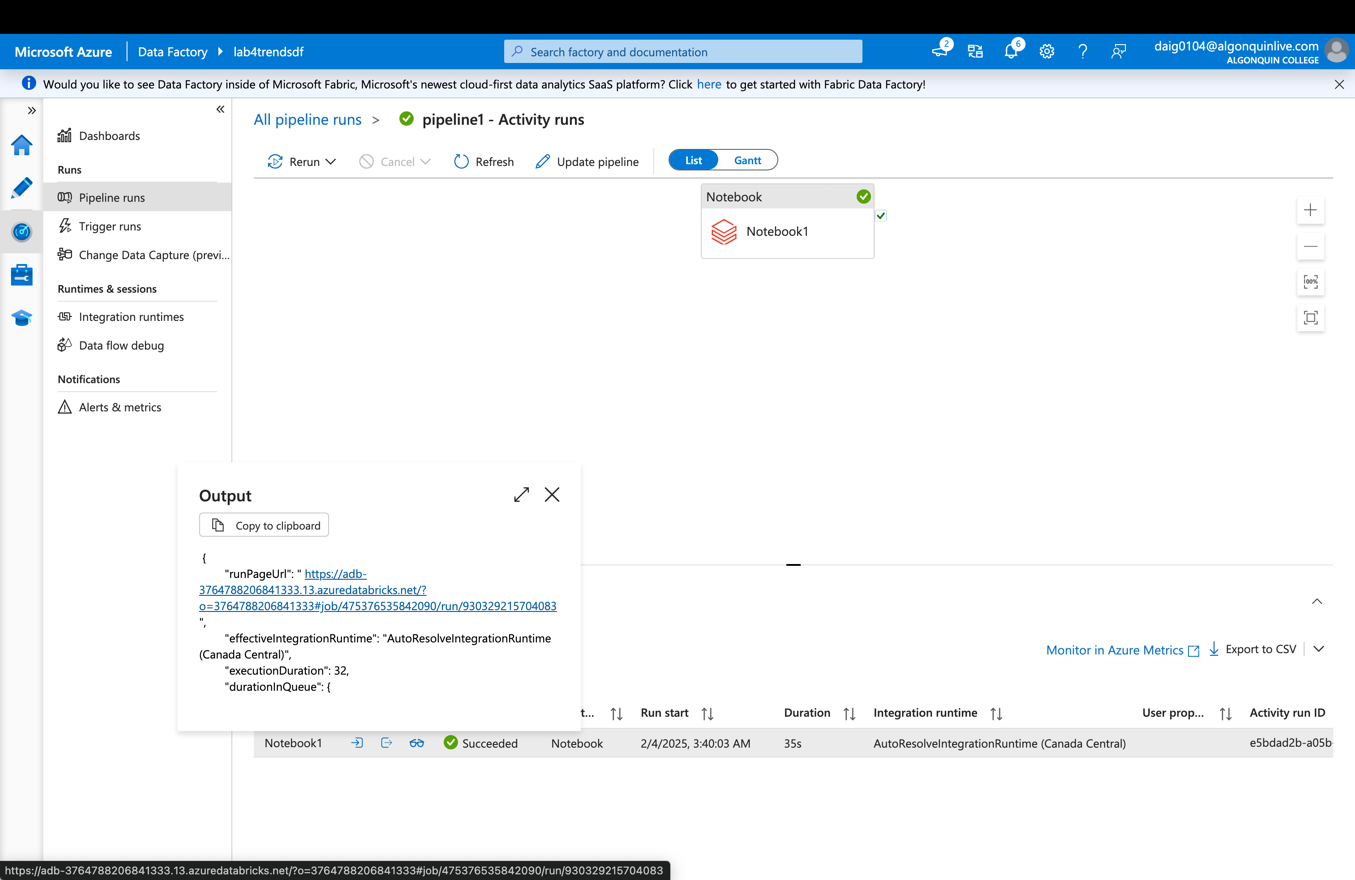
1. Linking pipeline to the notebook from part 1



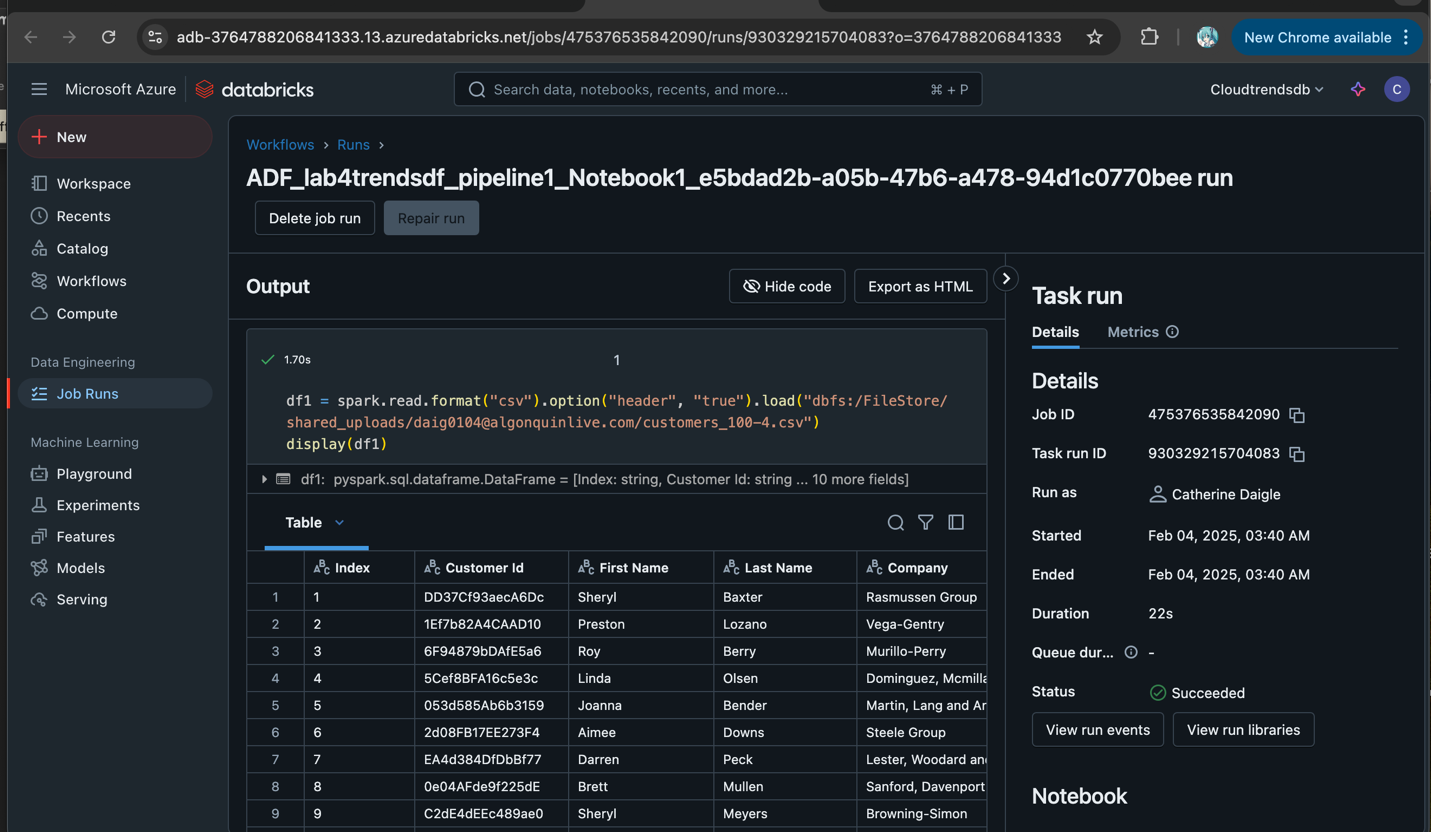
1. Verifying and publishing the pipeline (here I delete that input name it was not needed and caused an error)



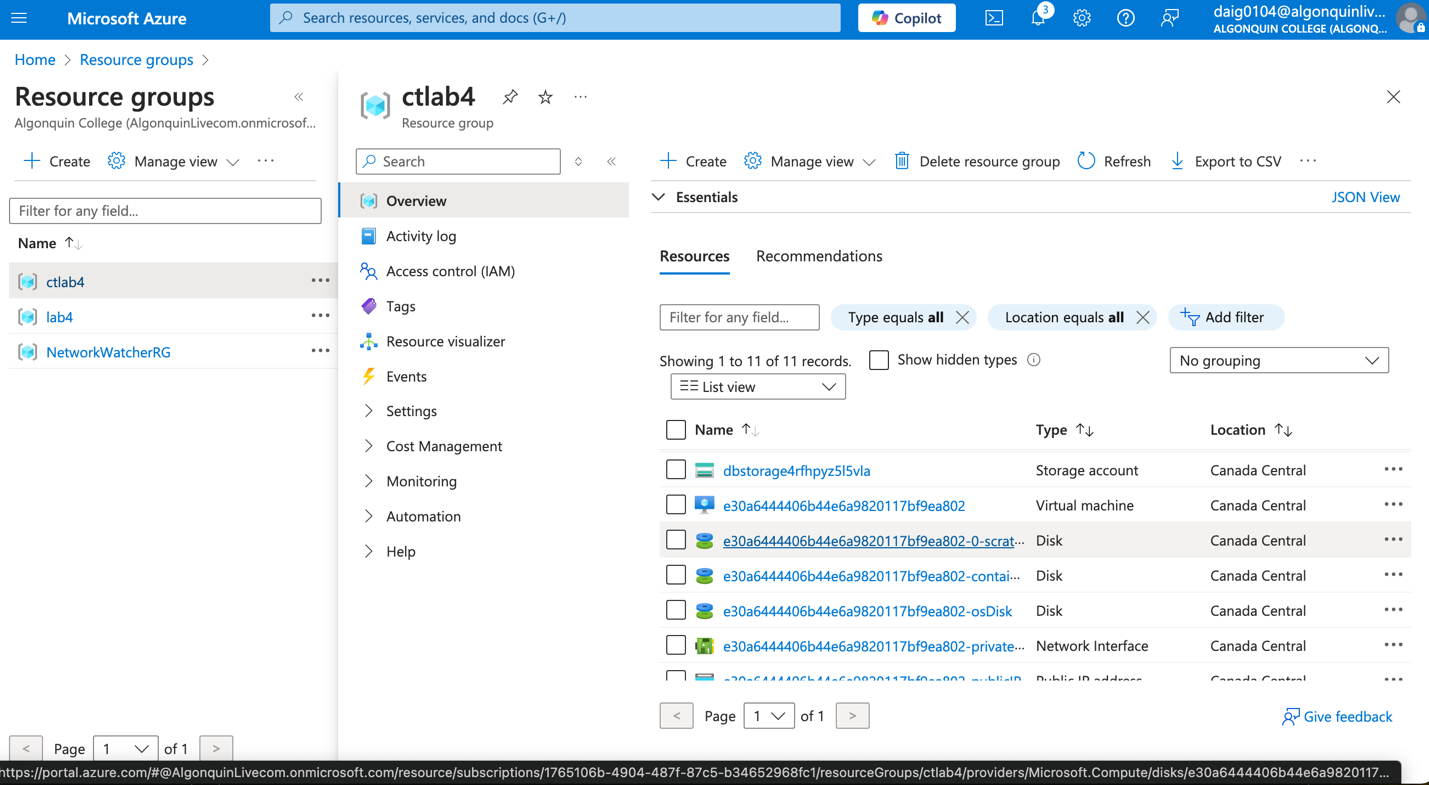
1. Execute the pipeline and monitor the run to see the status of pipeline. The pipeline displayed as succeeded and outputs a link to the notebook in databricks



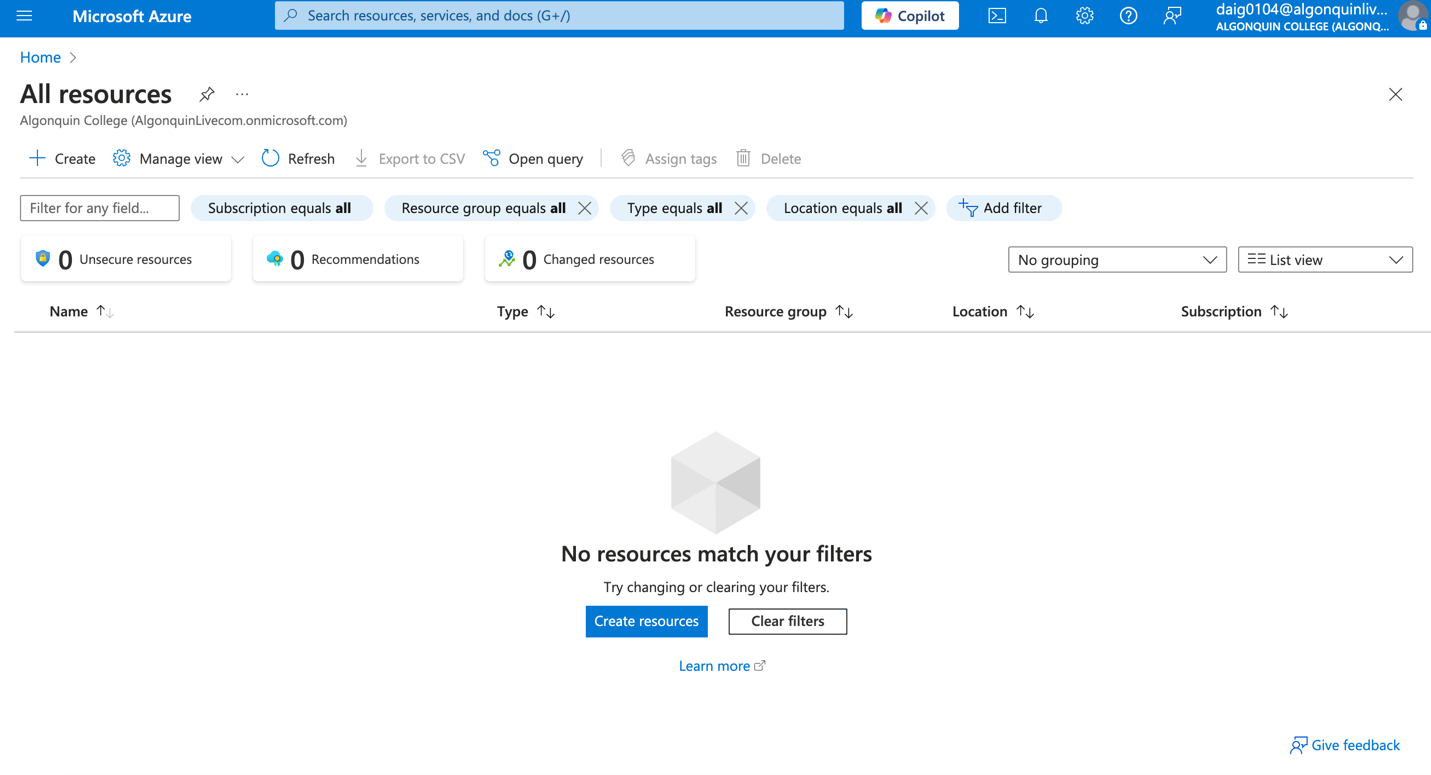
1. The outputted table from the pipeline



1. Deleting all the resources, here I noticed when making the cluster node, the VM and databricks resources are created here.



1. End deleted result.



## Result:

For the results in part one, I successfully created a cluster called CloudTrends and for the Node type, I chose Standard F4 as it was the cheapest option I could find. I have also uploaded a sample CSV of customers-100 to the DBSF Directory. From there, using the provided PySpark code, I displayed the data table.

In Part 2 I explored using Delta Lake within data bricks by following the tutorial and compared managed and external tables. The result of dropping both tables revealed that although the metadata for both tables is removed, the delta files for the external table remain. Lastly, I explored analyzing a table with streaming data and perpetually writing the stream to a delta table folder.

In part 3, the result was a created Data factory that successfully executes the Databrick notebook from part one within the pipeline. The result is the output of the table within part one shown in Screenshot #5 in part 3.

## Conclusion:

In conclusion, I explored how to use Databricks and Datafactory to analyze and process data from a sample CSV file. One issue was that I had trouble finding where to upload the CSV file. Databricks have multiple ways of uploading data. One was clicking new + data and uploading the file to DBSF. The data would appear within the catalogue as a table but did not give a sample PySpark code. I only retrieved the sample code when uploading the data within the part 1 notebook under the file tab. With that, I was able to display the CSV table.

In part 2, I explored the tutorial and created both managed and external tables in addition to processing streaming data into tables. In the tutorial, I learned about the different types of tables and their uses. Managed tables are stored in the hive metastore and you do not need to specify a path. When you delete a managed table, the data files are deleted with it. External tables data are stored externally, to create an external table, you must specify the path to the data location. When an external table is deleted, only the metadata is deleted, not the data files. External tables are great for cloud storage systems as the data of a cloud storage system would have to be managed externally.

For part 3, I created a data factory to link to the notebook from step one. To do that, I had to generate an access token within Databrick and copy the token to the new linked service’s Access token. For the cluster, I chose existing as I have already created a cluster in part one. After that, I added a new Databricks pipeline and connected it to the linked service. I then add the notebook’s path within the pipeline’s settings, verify and publish. There were a couple of issues I encountered at the time, but they were easily solved while creating the pipeline I also added some parameters (such as name and input, as you can see in Screenshot #15) for it. The parameters caused an error in publishing. It was solved after deleting those parameters.

Overall, I found that as clusters and resources are created, they are also made in the managed resource group when specifying the Azure Databricks workspace. I suspect that the VM added called e30a6444… is the VM that runs the Standard\_F4 single-node cluster.

# Bibliography

mssaperla, docs-preview-bot, irinaskaya, & mariesalet. (2024, September 26). *What is Azure Databricks?* Retrieved February 2025, from Microsoft Learn: https://learn.microsoft.com/en-us/azure/databricks/introduction/

Databricks. (2024, December 18). *What is Delta Lake?* Retrieved February 2025, from databricks: https://docs.databricks.com/en/delta/index.html

Databricks. (2025, January 10). *What is a table?* Retrieved February 2025, from databricks: https://docs.databricks.com/en/tables/tables.html

IBM. (2025, January 24). *What is Apache Spark?* Retrieved February 2025, from IBM: https://www.ibm.com/think/topics/apache-spark

JeannedArk. (2023, August 20). *Explore Delta Lake*. Retrieved February 2025, from Github: https://github.com/MicrosoftLearning/dp-203-azure-data-engineer/blob/master/Allfiles/labs/25/Delta-Lake.ipynb