AZURE CASE STUDY

HARSHINI V

QUESTION:

Create a ETL pipeline of ingestion & transform and load queries on any data set and initiate the pipeline from workflow using notebook.

1. Create a notebook with etl queries

2. Run the notebook from workflow pipeline in azure databricks workspace.

INTRODUCTION:

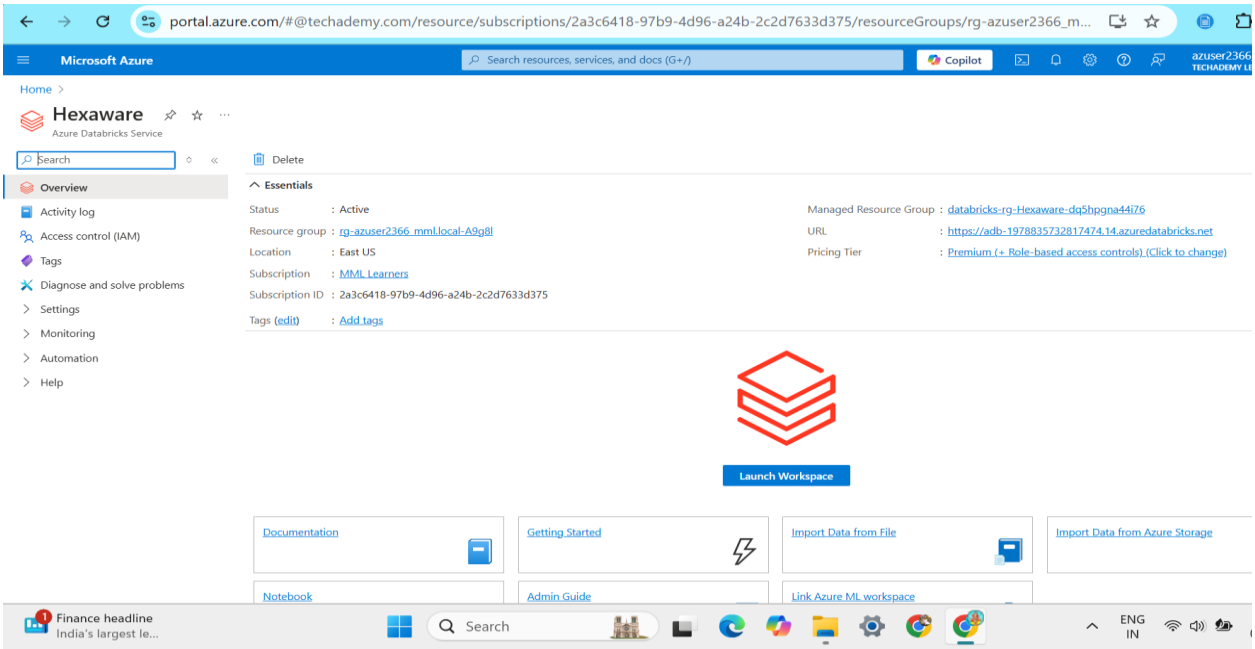
This document outlines the complete process of creating and executing an ETL (Extract, Transform, Load) pipeline on Azure Databricks using data stored in **Azure Blob Storage**. The pipeline reads a retail sales dataset, applies necessary data transformations, and stores the results in a Delta table for further analysis.

The key components of this pipeline are:

- **Extract**: Data ingestion from Azure Blob Storage.
- **Transform**: Data cleaning and transformation using PySpark.
- **Load**: Writing the cleaned data into a Delta table for subsequent use.

SET UP AZURE DATABRICKS WORKSPACE:

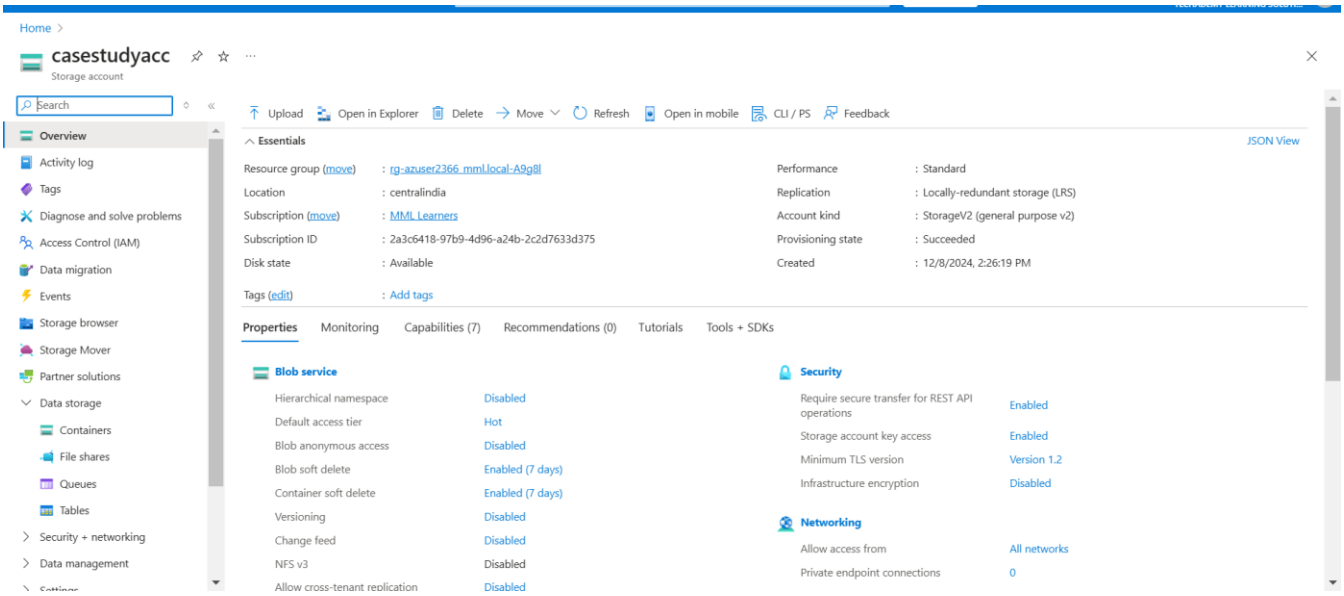
To begin, an **Azure Databricks Workspace** was created in the **Azure Portal**. This workspace serves as the environment for running notebooks, managing clusters, and scheduling jobs. By creating the workspace, you gain the ability to leverage Spark clusters for distributed data processing and the collaborative features of Databricks notebooks.



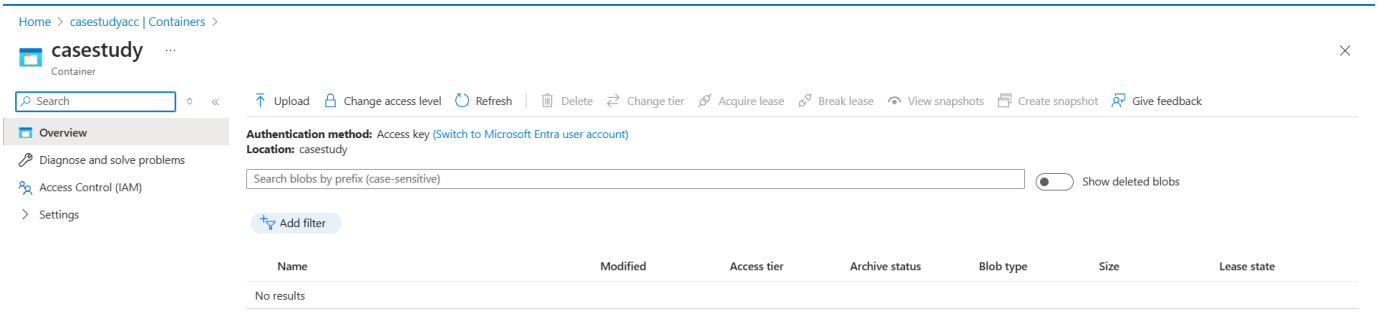
Once the workspace was created, a **cluster** was set up within Databricks. The cluster configuration was chosen based on the workload requirements, such as the appropriate **Spark version** and the number of workers needed for parallel processing. This cluster will execute the notebook and handle the data transformations in the ETL pipeline.

CREATE A STORAGE ACCOUNT:

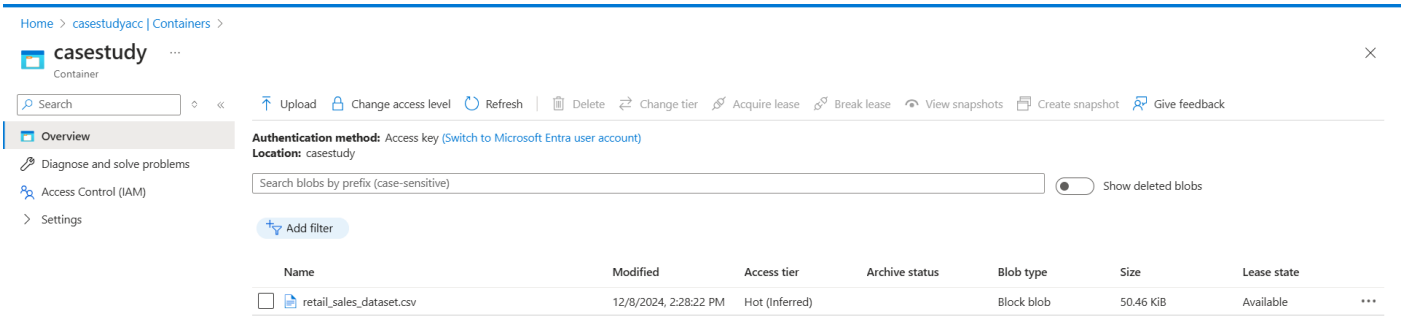
Next, an **Azure Storage Account** was created within the **Azure Portal**. This storage account is crucial for holding the source data for the ETL pipeline.



A **container** was created inside the storage account, which is where the dataset (e.g., a CSV file) is uploaded. This dataset will be read by Databricks for processing.

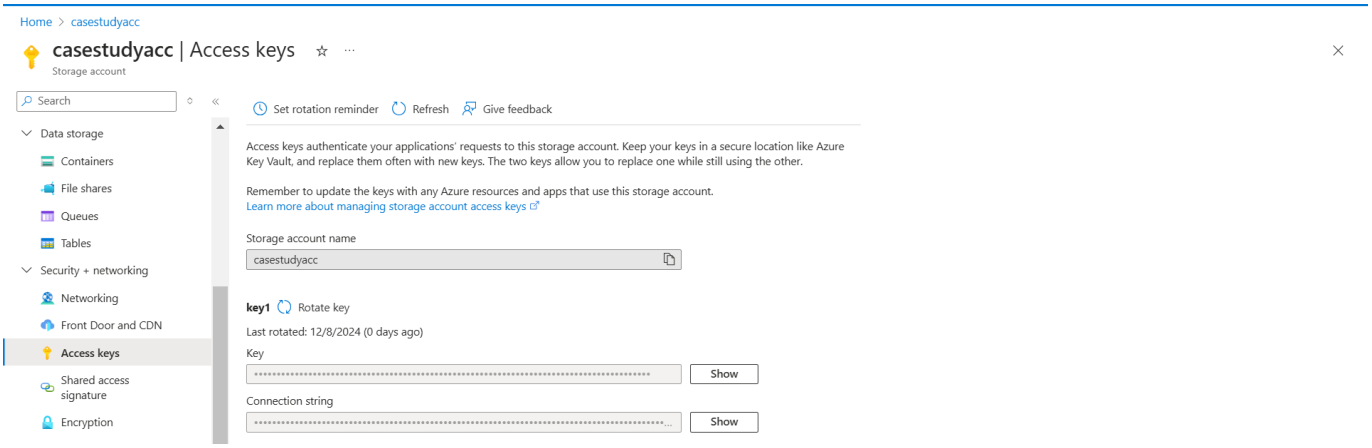


After the container was created, the **dataset** was uploaded to this container. The dataset is the raw data that will undergo transformations during the ETL pipeline. In this case, the uploaded dataset could include product sales data or any other relevant data for analysis.



GENERATE ACCESS KEY:

To securely access the data in the Azure Blob Storage, an **Access Key** was generated within the **Azure Storage Account**. This access key allows Databricks to authenticate and authorize the ETL pipeline to read the dataset from Blob Storage. Instead of using an SAS token, the Access Key method was chosen to provide secure, direct access to the storage account.



The key was then configured in the Databricks environment to enable communication between Databricks and Azure Blob Storage. This step ensures that Databricks can access the storage account and read the dataset into the Spark cluster for further processing.

With these environment setup steps completed, the foundation for running the ETL pipeline on **Azure Databricks** was established, ensuring that the necessary resources and permissions were in place for secure and efficient data processing.

CREATE THE ETL NOTEBOOK:

The process began by navigating to the **Workspace** tab in the **Databricks environment**. In this tab, a new **Notebook** was created. The notebook was associated with the pre-configured **cluster** to ensure that the necessary resources (e.g., memory, CPUs) were available for executing the code.

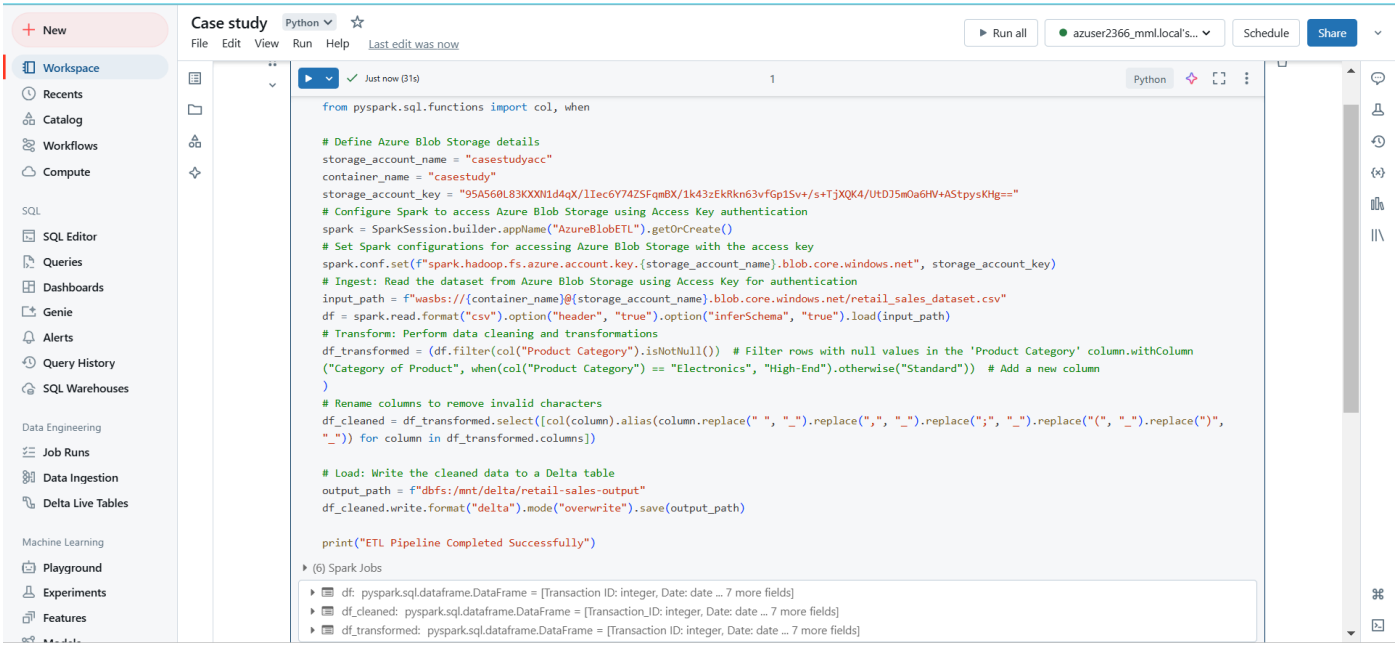
ETL QUERIES:

The notebook contains the core logic for the **ETL pipeline**. In this pipeline, the first step is to **read the data** from Azure Blob Storage. This is achieved by specifying the path to the dataset stored in the Azure Storage Account, which is accessed using the **Access Key** configured in the previous step. The data is loaded into the notebook using PySpark's **DataFrame API**. Specifically, the spark.read.format("csv") command is used to load a CSV file, with options like header=True to include the column headers and inferSchema=True to automatically detect the data types of the columns.

Next, the **transformation** phase begins, where the raw data is cleaned and transformed. In this case, PySpark functions like filter() and withColumn() are applied. For example, rows with **null values** in the Product Category column are removed, and a new column, "Category of Product," is added. This new column categorizes products as "High-End" if the product category is "Electronics" and as "Standard" otherwise. These transformations prepare the data for further analysis or reporting.

Finally, the **load** phase involves writing the transformed data back to Azure. In this example, the processed data is saved to a **Delta table** using df_transformed.write.format("delta").mode("overwrite").save(output_path). The Delta format ensures that the data is stored efficiently and can be easily queried and updated in the future.

By completing these steps in the notebook, the ETL pipeline is designed to read, transform, and load data efficiently within the Databricks environment. The notebook serves as the heart of the ETL process, and by following these steps, the entire pipeline can be automated and executed at scale.

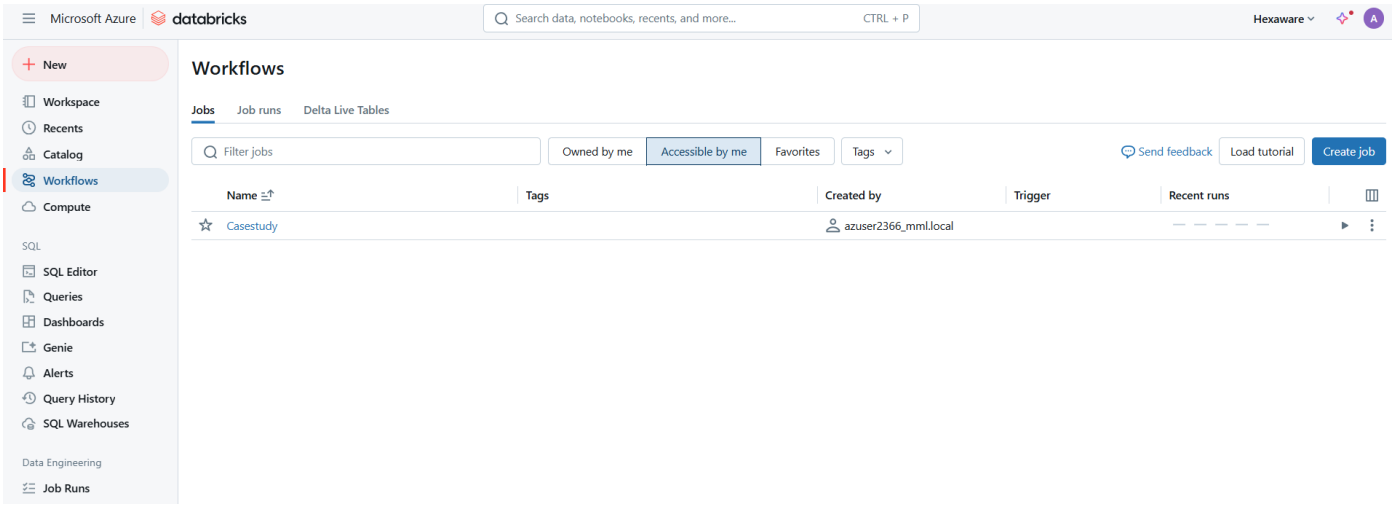


CREATE AND CONFIGURE THE WORKFLOW:

The workflow creation process begins by navigating to the **Workflows** section within the Databricks workspace. In this section, users can manage, schedule, and run various tasks, including notebooks and jobs. The workflow feature is essential for automating the execution of the ETL pipeline, ensuring that it runs smoothly at scheduled intervals or when triggered.

CREATE A NEW WORKFLOW:

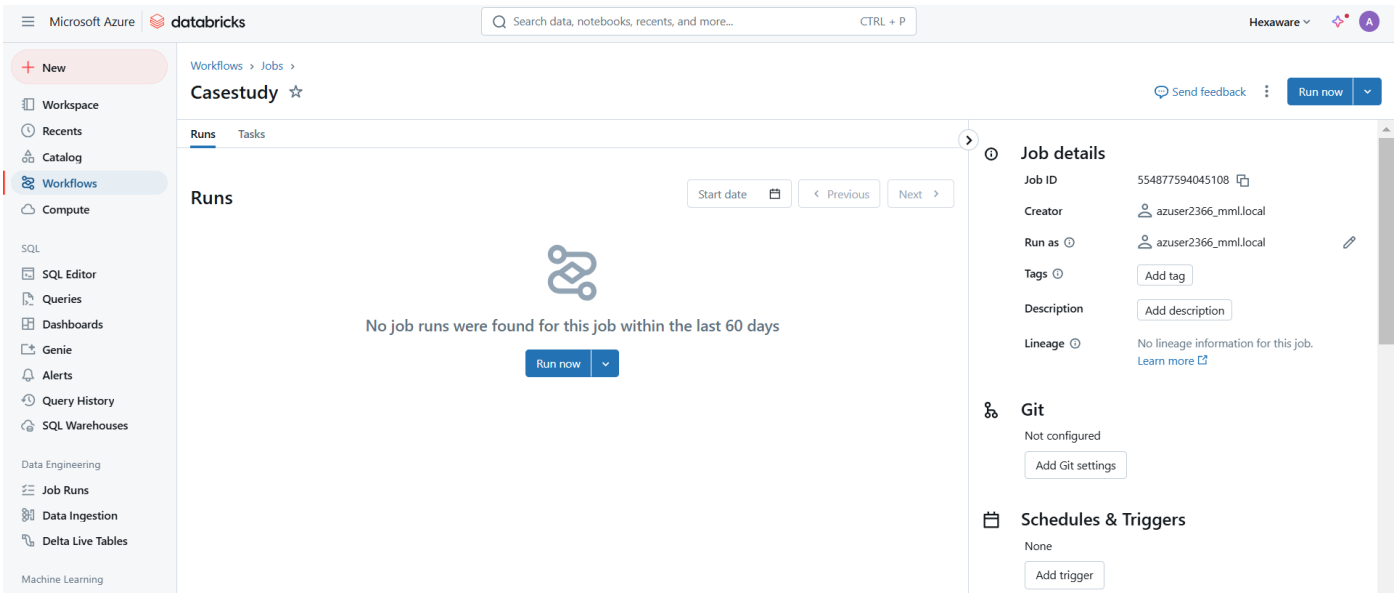
Once in the Workflows section, a new **workflow** was created by selecting the **Create Workflow** option. A meaningful name for the workflow was chosen, such as "Casestudy," to reflect its purpose clearly. This workflow will be used to execute the previously created ETL notebook, thereby automating the extraction, transformation, and loading of data from Azure Blob Storage to a Delta table.



SET UP WORKFLOW TASKS:

The next step in the workflow creation was to define the tasks that will be executed as part of the workflow. The first task was set as a **Notebook** type, ensuring that the workflow would run the ETL notebook that was created earlier. This task type allows the notebook to be executed in the correct environment, with access to the cluster and all necessary resources.

For the task configuration, the **Notebook Path** was specified to point to the notebook previously created in the workspace. By selecting this notebook path, the workflow is linked to the exact notebook that contains the ETL logic, ensuring that the transformations and data processing steps are executed.

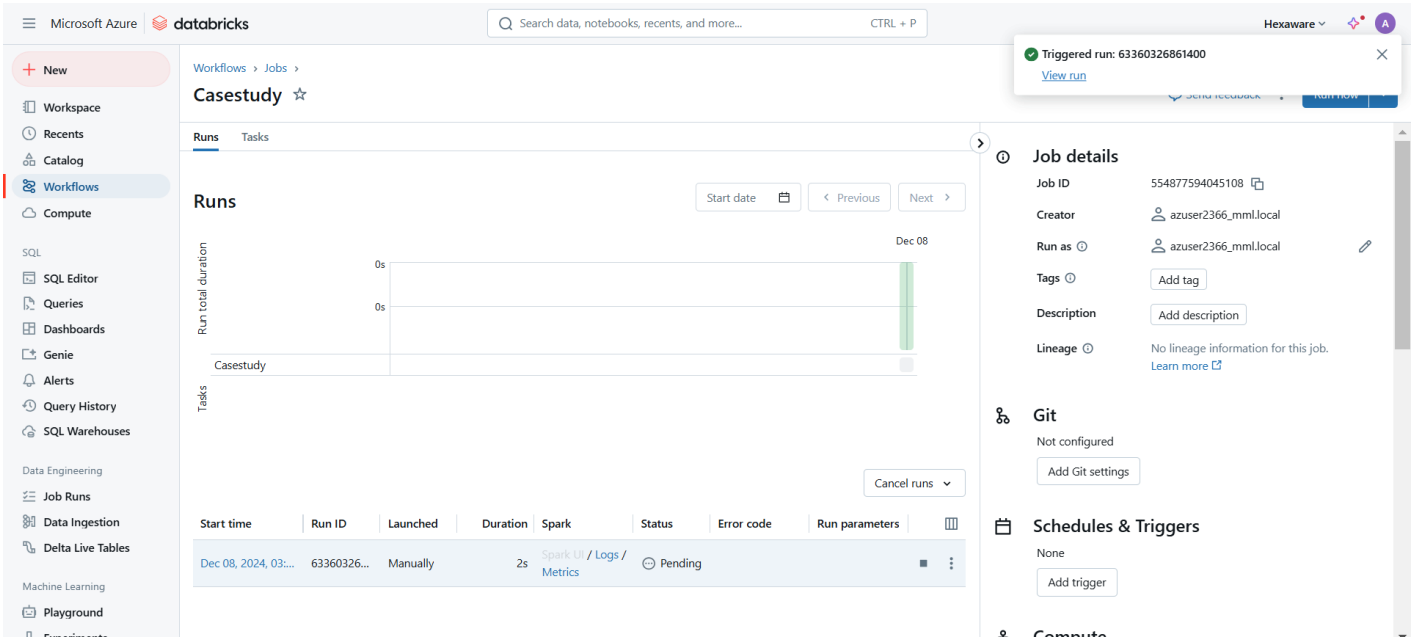


Additionally, the **Cluster** that was configured earlier in the Databricks workspace was assigned to the task. This cluster is critical as it provides the computational resources (e.g., CPU, memory) necessary to run the notebook. By assigning the cluster to the task, the notebook will have access to the required resources to execute the ETL pipeline efficiently.

This setup ensures that the ETL process is executed consistently and automatically, based on the parameters defined in the workflow configuration.

RUN THE WORKFLOW:

Once the workflow setup is complete, the next step is to test its functionality. To initiate the ETL pipeline, the **Run Now** option was selected from the **Workflow Dashboard**. This action immediately triggers the execution of the ETL process, which includes reading the data from Azure Blob Storage, performing the required transformations, and then loading the processed data into a Delta table.



The **Run Now** feature is particularly useful for testing the workflow's behavior before setting it up for regular execution. By triggering the workflow manually, one can verify that all steps in the ETL pipeline execute without errors and that the data flows through the extraction, transformation, and loading phases successfully.

This documentation provides a detailed overview of creating and running an ETL pipeline in **Azure Databricks**, from setup and data extraction to transformation and loading into a Delta table, along with workflow automation.