Data Lake Exploration & Optimization

Project Documentation

Table of Contents

- Project Statement
- Project Overview
- Prerequisites
- Project Objectives
- Architecture Diagram
- ER Diagram
- Azure Resources Used for the Project
- Tools Used
- How It Works
- Execution Overview
- Practical Implementation on Azure Portal
- Analysis Results
- Implementation-Tasks Performed
- Successful Output Generated
- Strategies that can be used in Optimising the process
- Conclusion

1. Project Statement

The main goal of this project is to **explore and optimize data stored in an Azure Data Lake** using **PySparkSQL** within **Azure Databricks**. By leveraging Databricks, we can perform **seamless data ingestion, transformation, querying, and visualization** at scale.

This project demonstrates how organizations can build an efficient **data analytics pipeline** where raw data is ingested into a Data Lake, processed using PySparkSQL, optimized with Delta format for performance, and then visualized for decision-making.

---

2. Project Overview

This project walks through the **end-to-end process** of building a data analytics workflow on Azure. The major steps include:

- **Data Ingestion:** Loading CSV data (employees.csv) into **Azure Data Lake Storage (ADLS)**.
- **Storage Mounting:** Connecting ADLS with **Azure Databricks** using secure authentication (SAS token).
- **Exploration with PySparkSQL:** Running SQL-like queries on employee data to understand and analyze the dataset.
- **Data Optimization:** Converting raw data into **Delta Lake format** for improved performance, ACID transactions, and scalability.
- **Analytics & Visualization:** Running advanced queries (e.g., top salaries, department statistics, role-based salary averages) and creating charts using Databricks built-in visualization tools.

By the end of the project, we establish a **scalable and optimized pipeline** for handling structured data in a cloud-native way.

---

3. Prerequisites

To implement this project, the following requirements are needed:

- Azure Subscription with active credits (student subscription or paid).
- Azure Storage Account with at least one container to hold datasets.
- Azure Databricks Workspace with an active cluster for execution.

- **Dataset:** employees.csv containing employee records such as ID, name, department, role, salary, etc.
- **SAS Token** (Secure Access Signature) to mount and connect Databricks with the storage account securely.

---

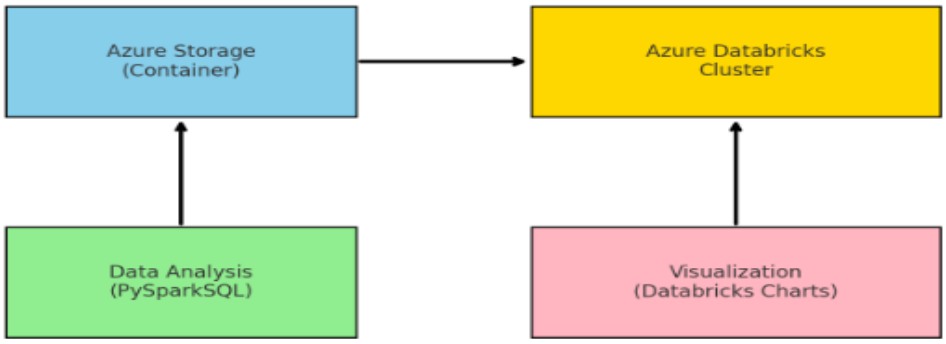
4. Project Objectives

The objectives of this project are as follows:

1. **Connect Azure Databricks with Azure Data Lake Storage** using a secure method.
2. **Perform exploratory data analysis** on employee dataset using PySparkSQL queries.
3. **Optimize data storage** by converting raw CSV into Delta Lake format (Bronze → Silver layers).
4. **Generate valuable insights** such as:
    - Top employees with highest salaries.
    - Department-wise employee distribution.
    - Average salaries per role.
    - Trends in employee data for analytics.
5. **Visualize results** using Databricks charts for better understanding and presentation.

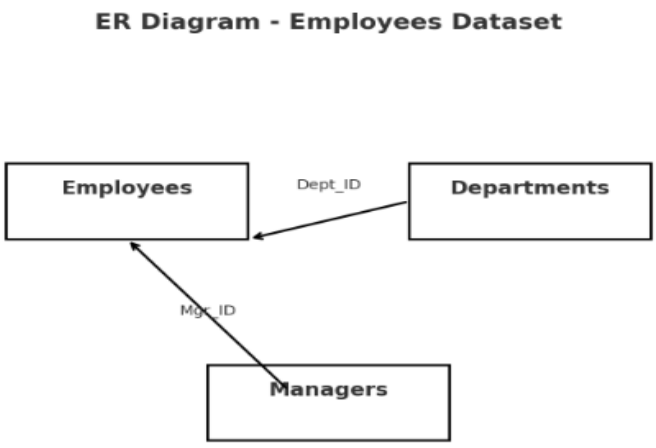
---

5. Architecture Diagram



---

6. ER Diagram



---

7. Azure Resources Used for the Project

- **Azure Storage Account:** sivagamistudentstorage
- **Container Name:** mycontainer (stores employees.csv)
- **Azure Databricks:** Workspace and Cluster (used for computation and query execution).
- **Azure Data Lake:** Serves as the central repository for raw and optimized data.

---

8. Tools Used

- **Azure Portal –** For provisioning and managing cloud resources (Storage, Databricks).
- **Azure Databricks –** For big data processing, analytics, and machine learning.
- **PySparkSQL –** For running SQL queries on structured data inside Spark.
- **Azure Storage Explorer (Optional) –** For managing datasets inside containers.
- **Databricks UI –** For creating notebooks, running queries, and building visualizations.

---

9. How It Works

The workflow of this project can be summarized as follows:

1. **Upload dataset to Data Lake:** The employees.csv file is uploaded to the container (mycontainer) inside the storage account.
2. **Generate SAS token:** A secure token is created to allow Databricks to access the storage account.
3. **Mount storage to Databricks:** Using the SAS token, we mount the container into the Databricks file system (DBFS).
4. **Read and Explore Data:** Load the dataset into a Spark DataFrame and run PySparkSQL queries for analysis.
5. **Data Optimization:** Store processed data in **Delta Lake format** to support ACID transactions and better query performance.
6. **Bronze → Silver Architecture:**
    - **Bronze Layer:** Raw ingested CSV data.
    - **Silver Layer:** Cleaned and optimized Delta tables.
7. **Analytics & Visualization:** Queries are run to derive insights and Databricks' visualization features are used to generate charts.

---

10. Execution Overview

- **Data Source:** Employee dataset (employees.csv) stored in Data Lake.
- **Storage:** Azure Blob Storage container (mycontainer).
- **Processing Engine:** Azure Databricks cluster running PySparkSQL.
- **Optimization:** Data converted to Delta Lake format for performance.
- **Visualization:** Databricks notebook charts (bar chart, pie chart, etc.) created from SQL query outputs.

---

11. Practical Implementation on Azure Portal

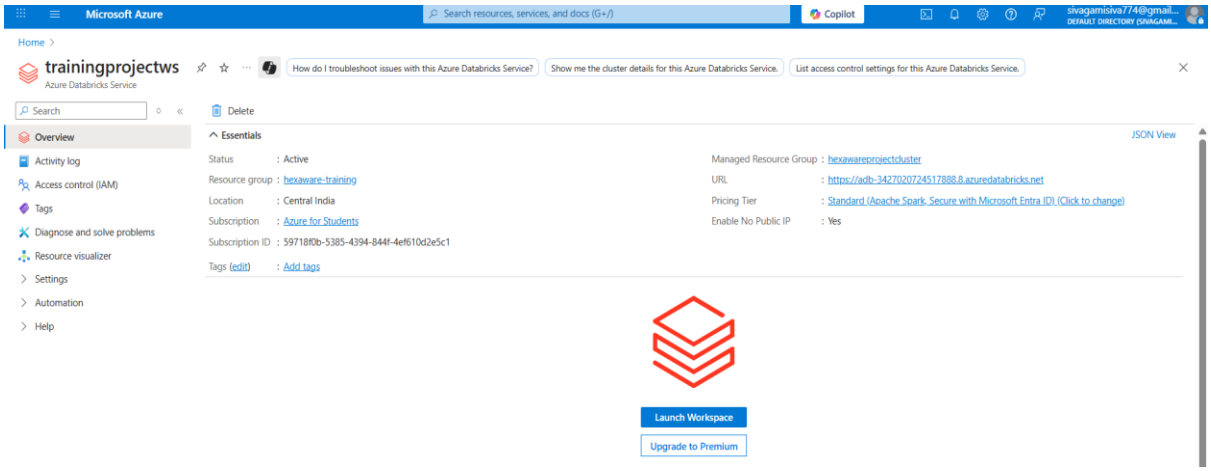
The practical steps carried out on Azure Portal and Databricks are:

1. **Create Storage Account** → Add a container (mycontainer).
2. **Upload Dataset** → Place employees.csv in the container.
3. **Create Databricks Workspace** → Launch Databricks and set up a cluster.
4. **Generate SAS Token** → From the storage account for authentication.
5. **Mount Storage in Databricks** → Use the SAS token to mount the container.
6. **Load Data in Notebook** → Read employees.csv into Spark DataFrame.
7. **Run SQL Queries** → Examples:
    - Top 10 highest salaries.
    - Average salary by department.
    - Role-wise statistics.
8. **Convert to Delta Format** → Save optimized data in Silver layer.
9. **Visualize Data** → Create charts (bar, pie, histogram) within Databricks.

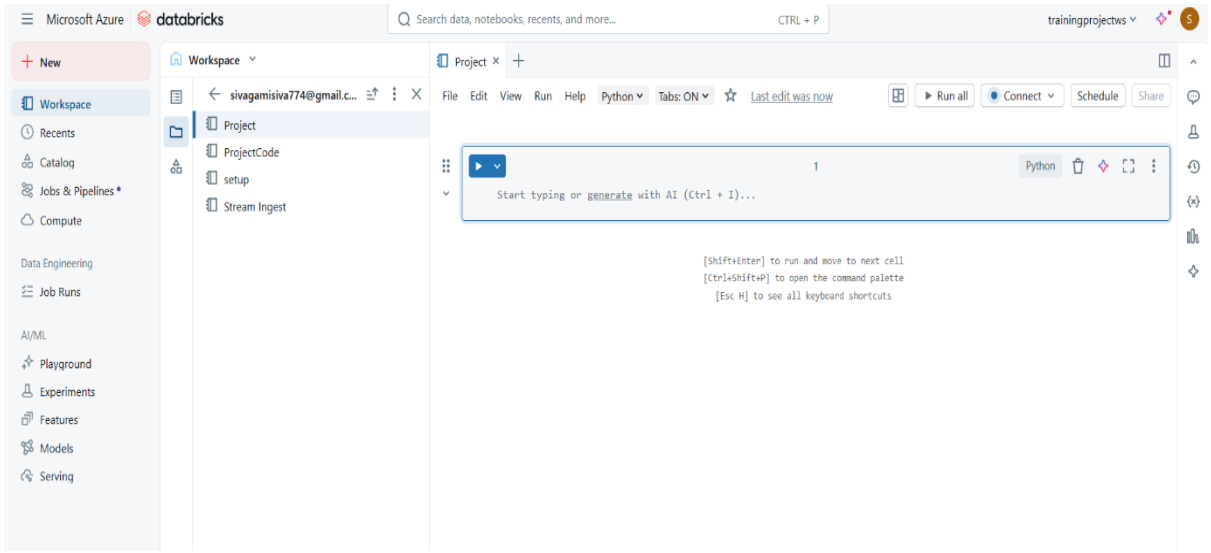
Step 1: Set Up Azure Databricks

- Created **Databricks workspace** and cluster in Azure Portal.
- Started cluster before working.

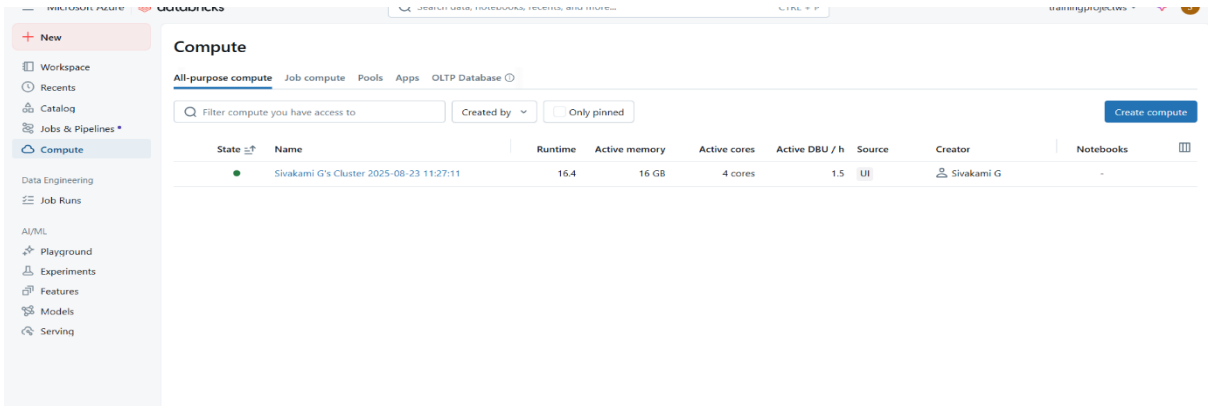
Screenshot: Databricks Workspace



Screenshot: New Notebook Creation



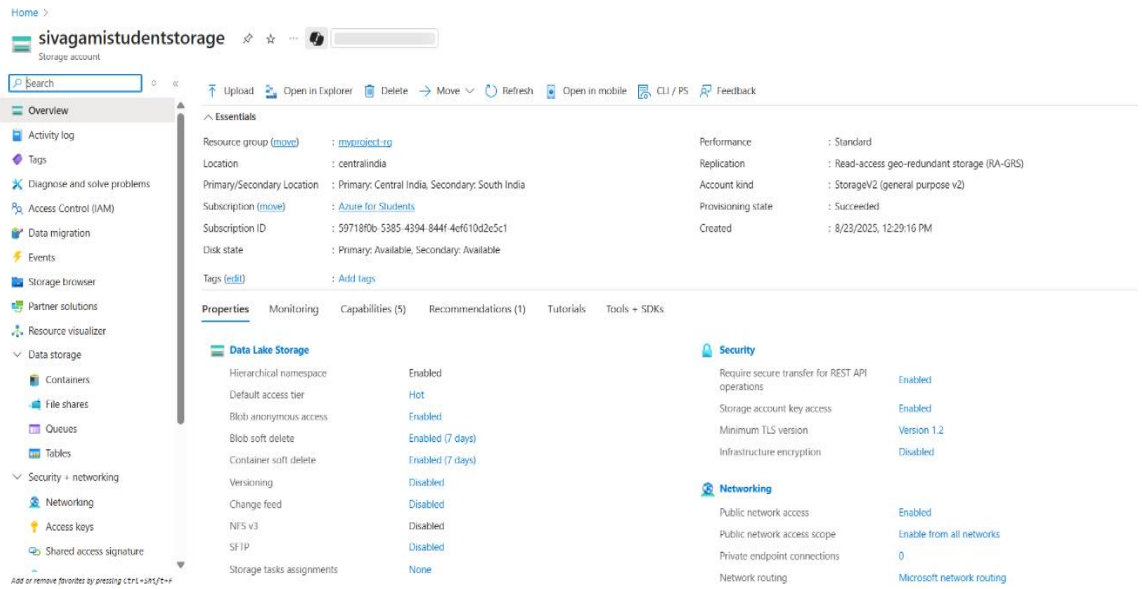
Screenshot: Cluster Running



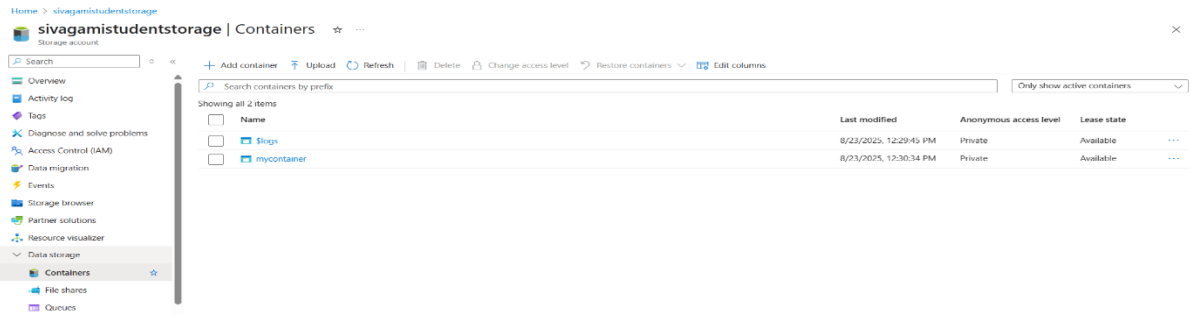
Step 2: Create Azure Storage Account & Container

- Created storage account: sivagamistudentstorage.
- Inside it, created container: mycontainer.

Screenshot: Storage Account Created



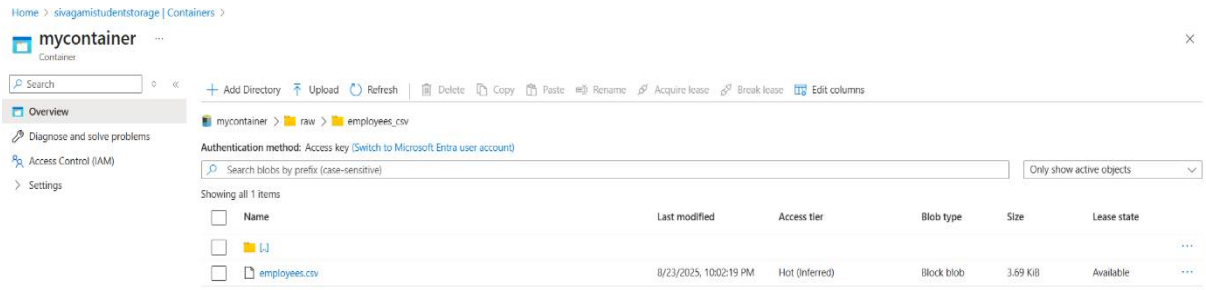
Screenshot: Container Created



Step 3: Upload Dataset

- Uploaded employees.csv file into mycontainer.

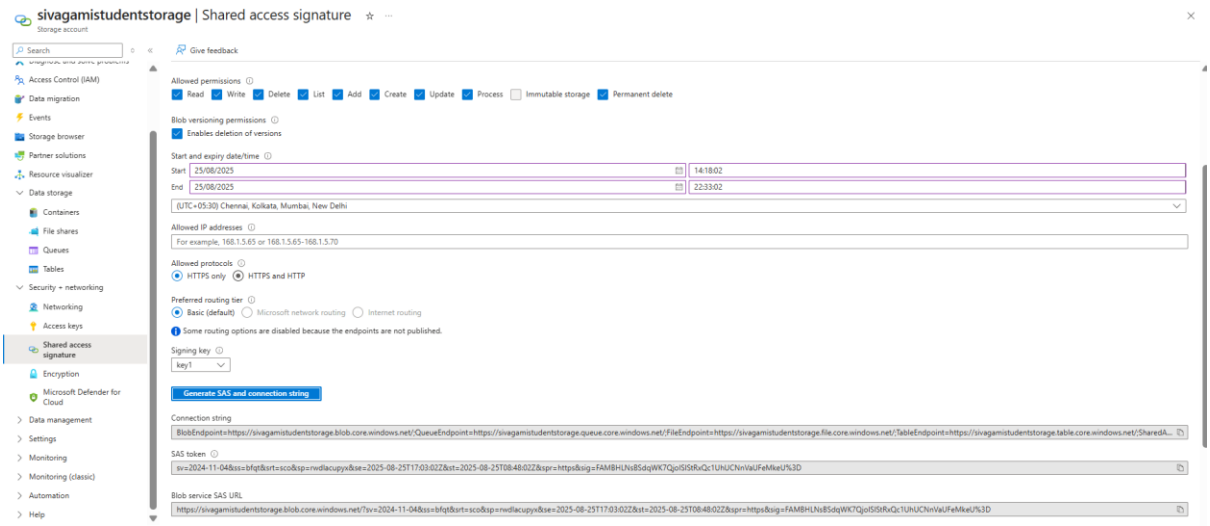
Screenshot: employees.csv Uploaded



Step 4: Generate SAS Token

- In storage account → **Shared Access Signature**.
- Enabled all permissions and generated token.
- Copied SAS token string for later use.

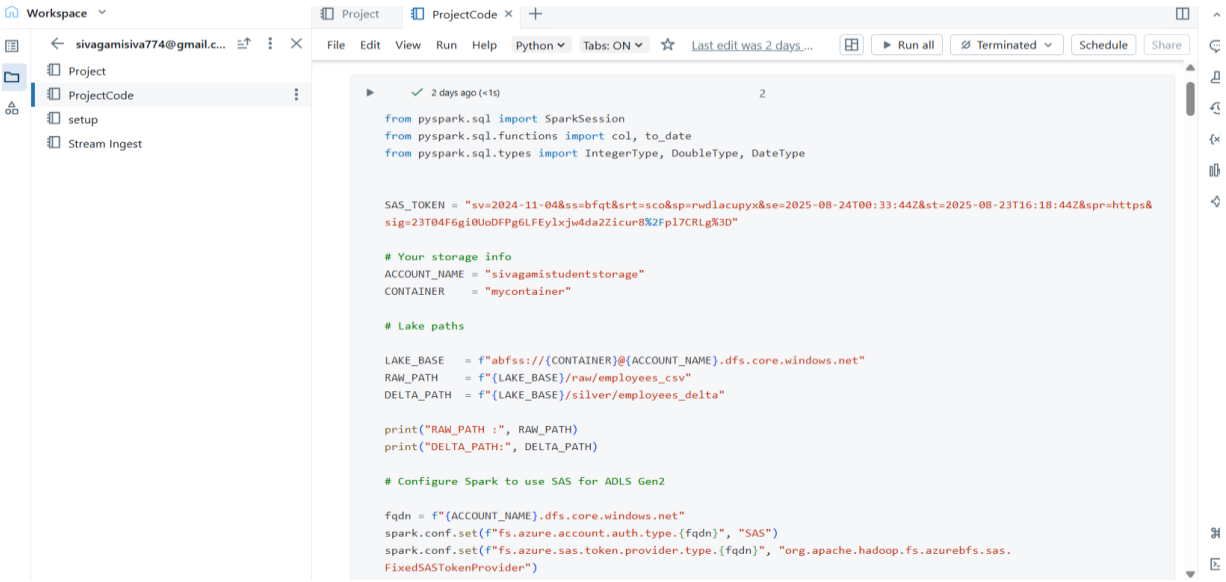
Screenshot: SAS Token Generated

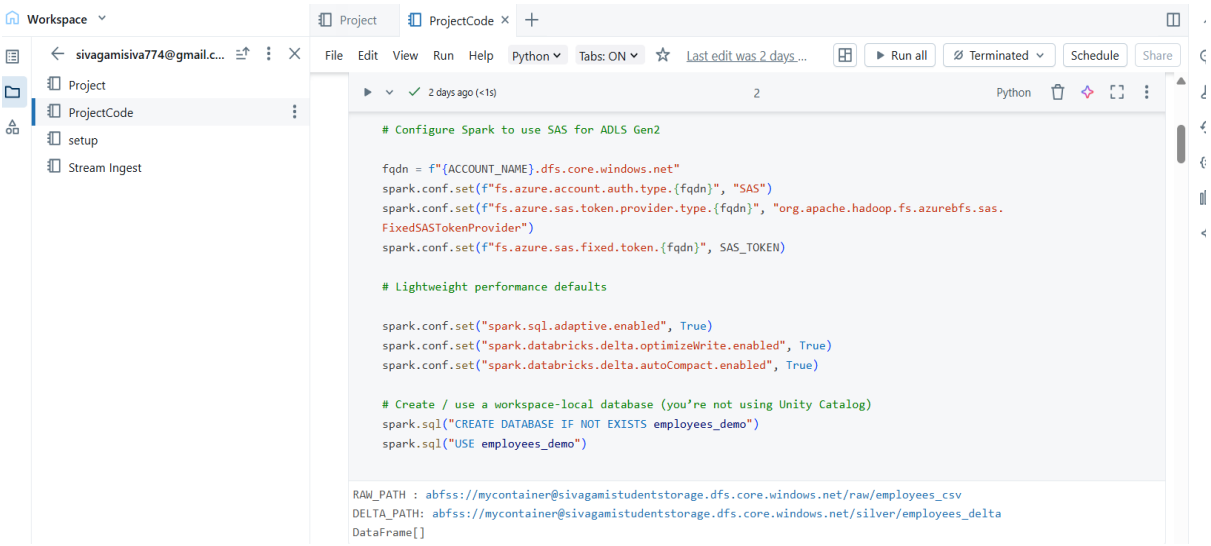


Step 5: Mount Storage in Databricks

- Used PySpark to mount container using SAS token.

Screenshot: Mounting Code in Databricks

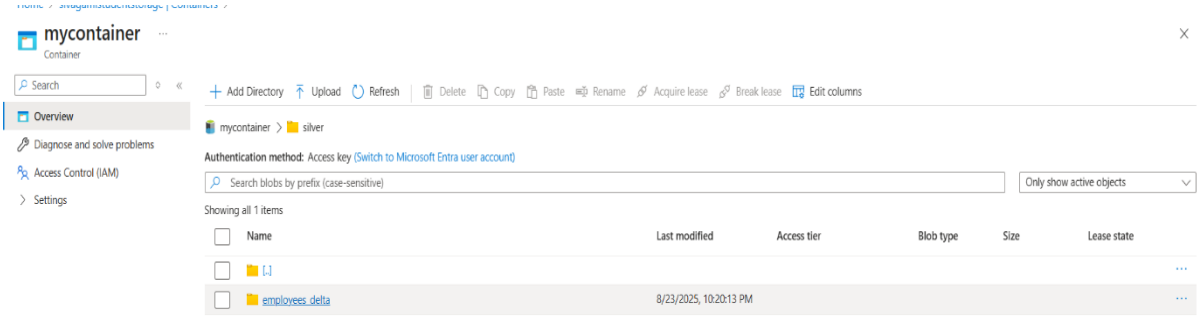




Step 6: Create Employees Delta File

- Converted raw CSV into Delta format.
- Saved into mycontainer/silver/employees_delta.

Screenshot: employees_delta File Created



Step 7: Read CSV from Lake

- Used spark.read.csv() to read employees.csv.

Screenshot: Read CSV Code Output

```
▸ (5) Spark Jobs
▸ df_raw: pyspark.sql.dataframe.DataFrame = [EMPLOYEE_ID: integer, FIRST_NAME: string ... 9 more fields]
|        201|   Michael|Hartstein|MHARTSTE|515.123.5555|17-FEB-04|  MK_MAN| 13000|            - |        100|
0|
|        202|       Pat|      Fay|    PFAY|603.123.6666|17-AUG-05|  MK_REP|  6000|            - |        201|
0|
+-----------+----------+---------+--------+------------+---------+--------+------+--------------+-----------+-----------
-+
only showing top 5 rows
root
 |-- EMPLOYEE_ID: integer (nullable = true)
 |-- FIRST_NAME: string (nullable = true)
 |-- LAST_NAME: string (nullable = true)
 |-- EMAIL: string (nullable = true)
 |-- PHONE_NUMBER: string (nullable = true)
 |-- HIRE_DATE: string (nullable = true)
 |-- JOB_ID: string (nullable = true)
 |-- SALARY: integer (nullable = true)
 |-- COMMISSION_PCT: string (nullable = true)
 |-- MANAGER_ID: string (nullable = true)
 |-- DEPARTMENT_ID: integer (nullable = true)

Rows: 50
```

Step 8: Write to File (Delta Format)

- Wrote employees dataset into Delta format for optimization.

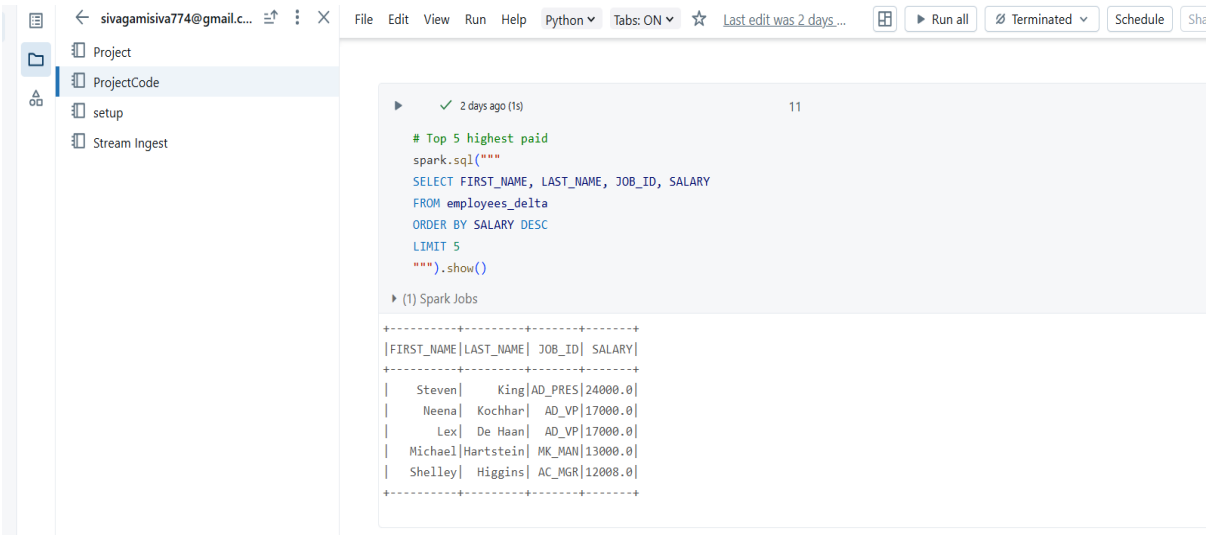
Screenshot: Writing to Delta File

---

12. Analysis Results

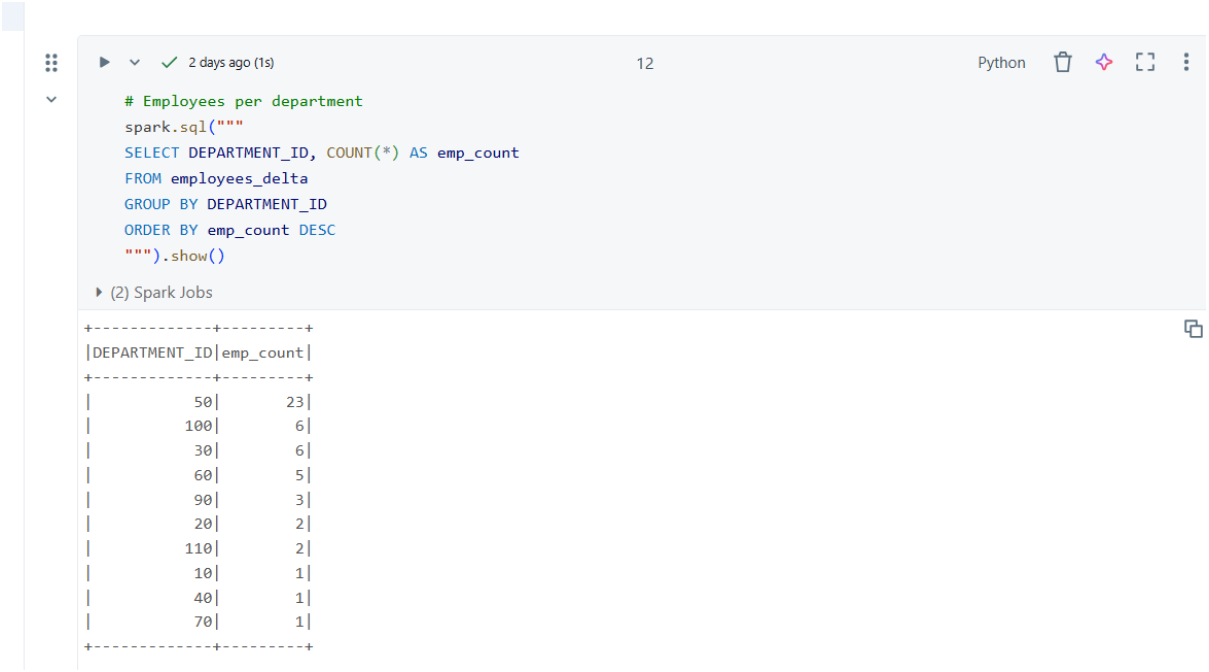
◆ Query 1: Top 5 Highest Paid Employees

SQL Query + Result



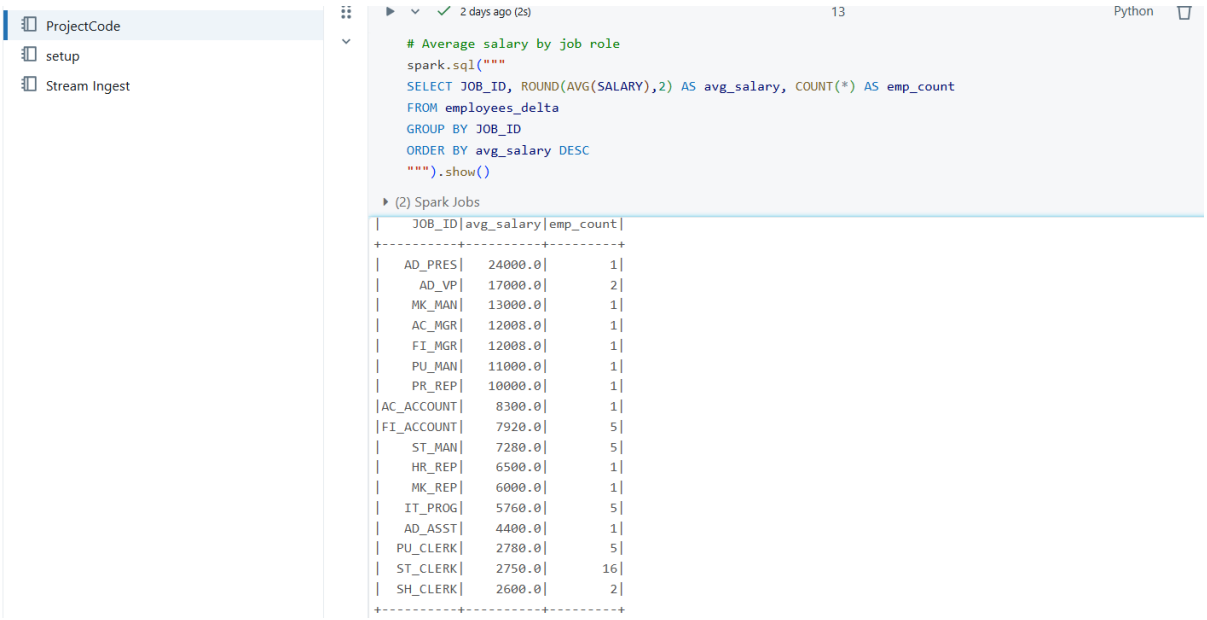
- **Query 2: Employees per Department**

SQL Query + Result



- **Query 3: Average Salary by Job Role**

SQL Query + Result



- **Query 4: Manager-Employee Counts**

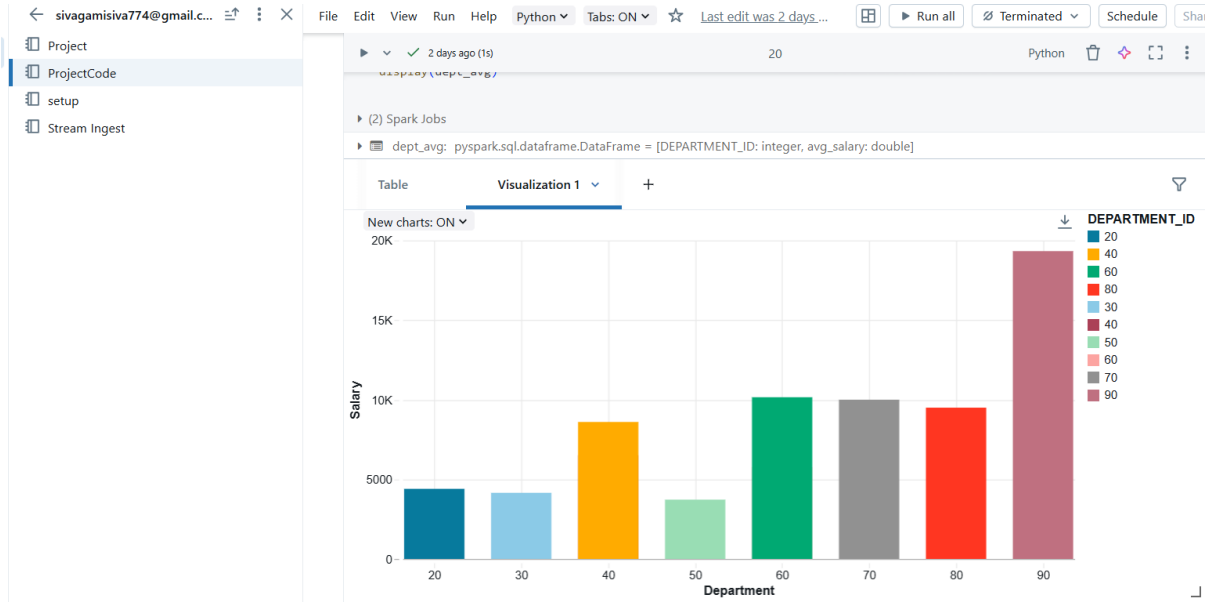
SQL Query + Result

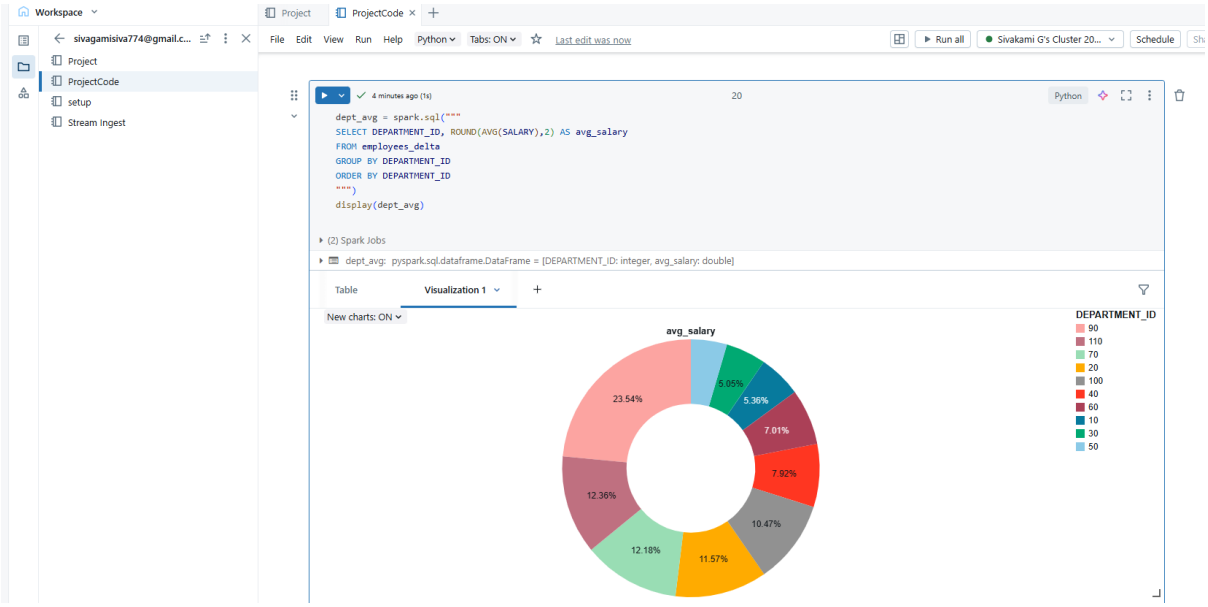
---

13. Successful Ouput Generated - Chart Visualization

- Used Databricks chart option to visualize employees per department.

Chart Visualization





---

14. Implementation Tasks Performed

- Set up Azure Storage Account and container.
- Uploaded employees.csv dataset.
- Created and configured Azure Databricks Workspace.

- Generated SAS Token for secure access.
- Mounted ADLS to Databricks File System (DBFS).
- Explored dataset with PySparkSQL queries.
- Converted raw data to Delta format (Bronze → Silver).
- Performed analytics and insights extraction.
- Built visualizations inside Databricks notebooks.

---

15. Strategies that can be used in Optimising the process

- **Use Delta Lake Format** – Enables ACID transactions, schema enforcement, and faster queries compared to raw CSV/Parquet.
- **Partitioning & Bucketing** – Organize data by columns (e.g., department, date) for efficient filtering and reduced scan times.
- **Caching & Persisting Data** – Cache frequently used DataFrames or tables in Databricks memory for faster iterative queries.
- **Optimize File Sizes** – Compact small files into larger ones to reduce overhead and improve Spark performance.
- **Leverage Z-Ordering & Indexing** – Use Databricks Z-ORDER and indexes for efficient query pruning and faster lookups.

---

16. Conclusion

This project successfully demonstrated how to:

- Connect Azure Databricks with Data Lake.
- Explore and optimize data using PySparkSQL.
- Perform analytics and generate visual insights.
- Optimize data storage with Delta Lake.

The integration of Databricks and ADLS provided a **scalable, efficient, and powerful data analysis environment** for real-world data engineering tasks.