Project Report: Data Ingestion from S3 to RDS with Fallback to AWS Glue using Dockerized Python Application

1. Introduction

This project focuses on building a resilient and automated data ingestion pipeline using AWS services and Docker. The pipeline reads data from an Amazon S3 bucket, attempts to upload it to an Amazon RDS (MySQL-compatible) database, and if that fails, falls back to AWS Glue for data cataloging. The entire process is containerized using Docker to ensure portability and ease of deployment.

2. Objectives

The objective of this project is to:- Automate data ingestion from S3 to RDS- Implement a fallback mechanism using AWS Glue- Package the solution using Docker for scalable deployment

3. Requirements

Software Requirements:- Python 3.9+- Docker- AWS CLI (configured with credentials)- boto3, pandas, sqlalchemy, pymysqlAWS Resources:- S3 Bucket (for CSV file)- RDS MySQL-compatible instance- AWS Glue (Database &

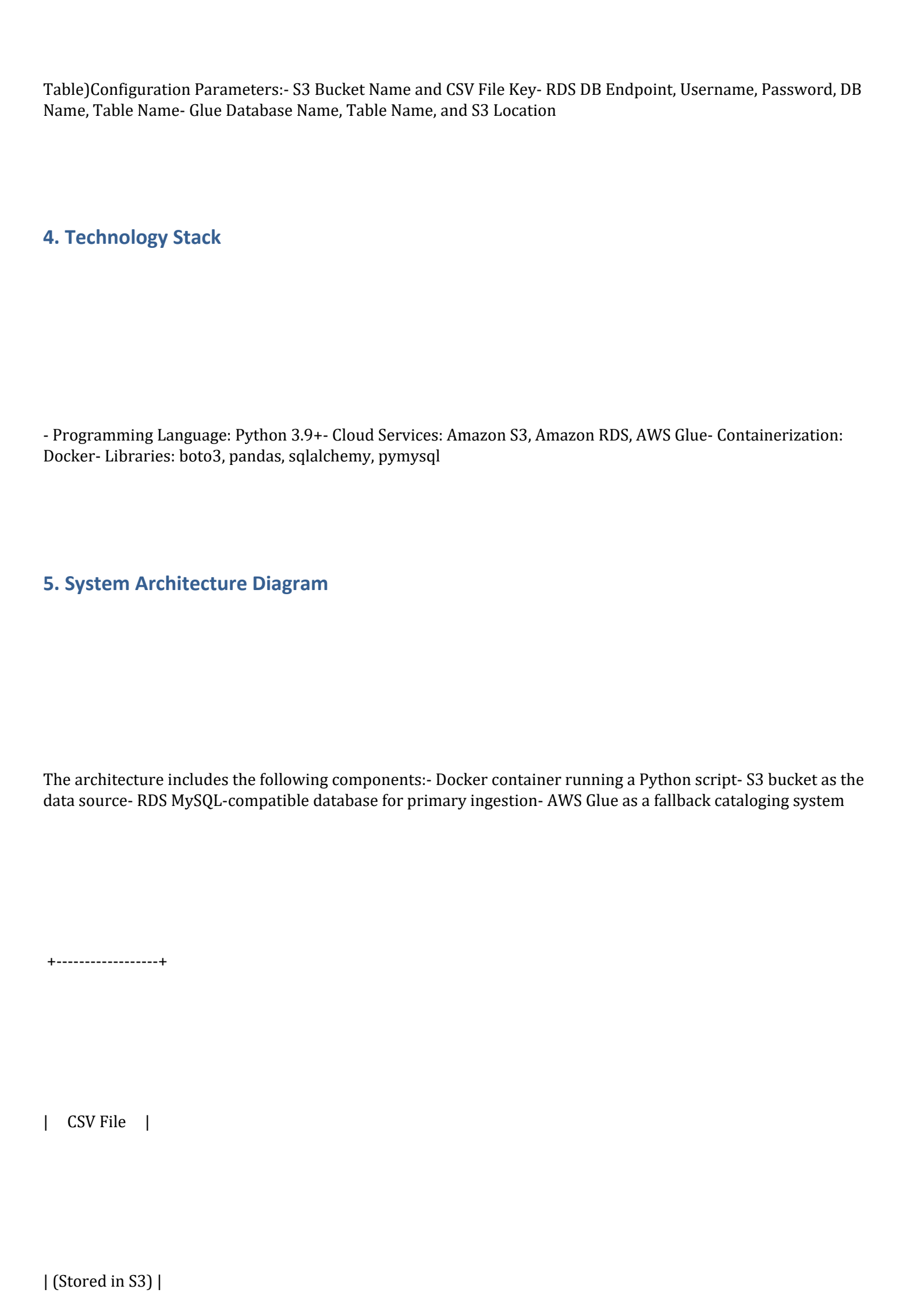
Table)Configuration Parameters:- S3 Bucket Name and CSV File Key- RDS DB Endpoint, Username, Password, DB Name, Table Name- Glue Database Name, Table Name, and S3 Location

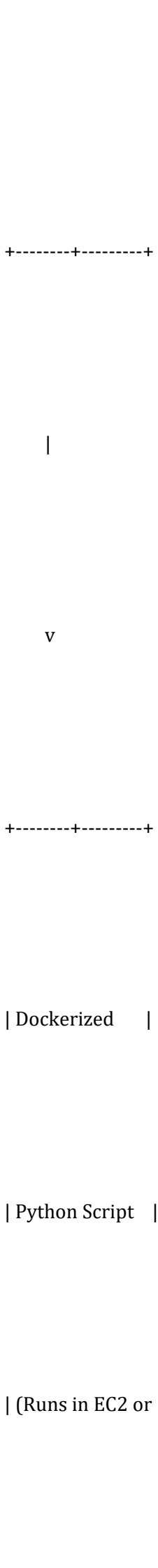
4. Technology Stack

- Programming Language: Python 3.9+- Cloud Services: Amazon S3, Amazon RDS, AWS Glue- Containerization: Docker- Libraries: boto3, pandas, sqlalchemy, pymysql

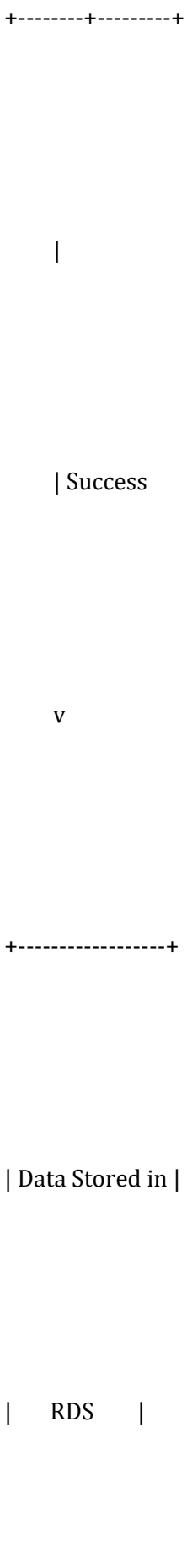
5. System Architecture Diagram

The architecture includes the following components:- Docker container running a Python script- S3 bucket as the data source- RDS MySQL-compatible database for primary ingestion- AWS Glue as a fallback cataloging system




+--------+---------+
|
v
+--------+---------+
| Dockerized |
| Python Script |
| (Runs in EC2 or |


| local container)|
+--------+---------+
|
v
+--------+---------+
| Attempt to push |
| data to RDS |
| (MySQL-compatible)|


+--------+---------+
|
| Success
v
+------------------+
| Data Stored in |
| RDS |


+------------------+
|
| Failure
v
+---------+----------+
| Fallback to |
| AWS Glue |

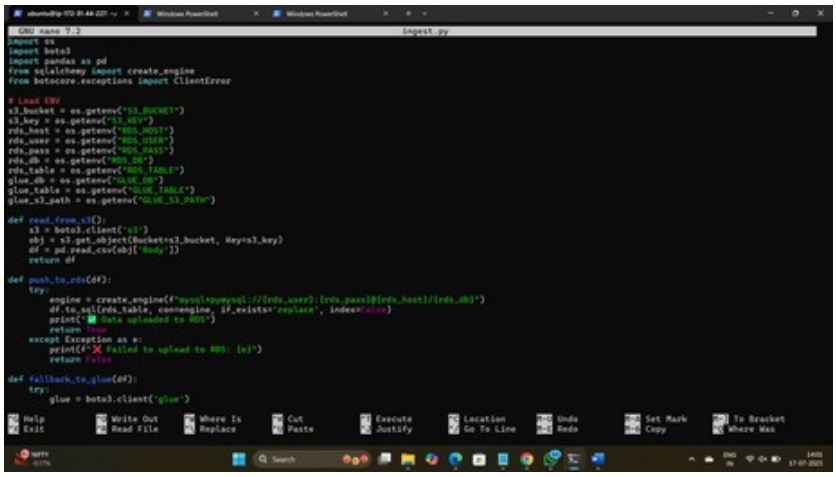

+--------+---------+
|
v
+------------------+
| Glue Table in |
| Data Catalog |
| with S3 Location |

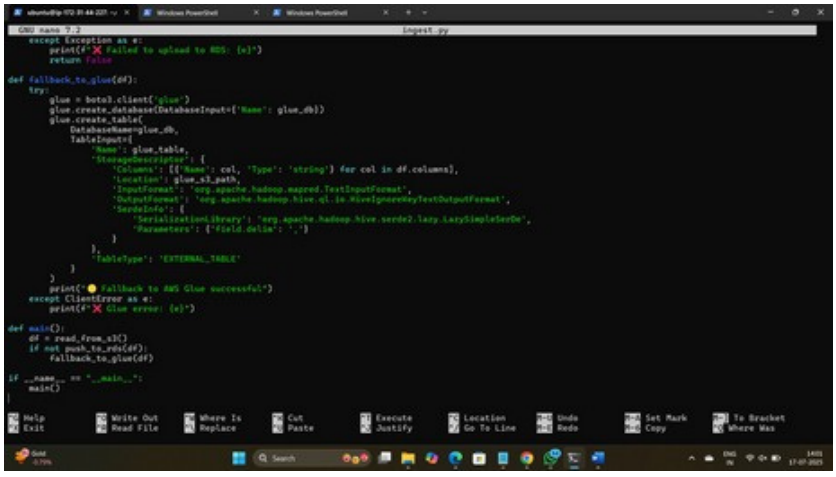
+------------------+

6. Implementation Steps

Step 1: Python Script Development

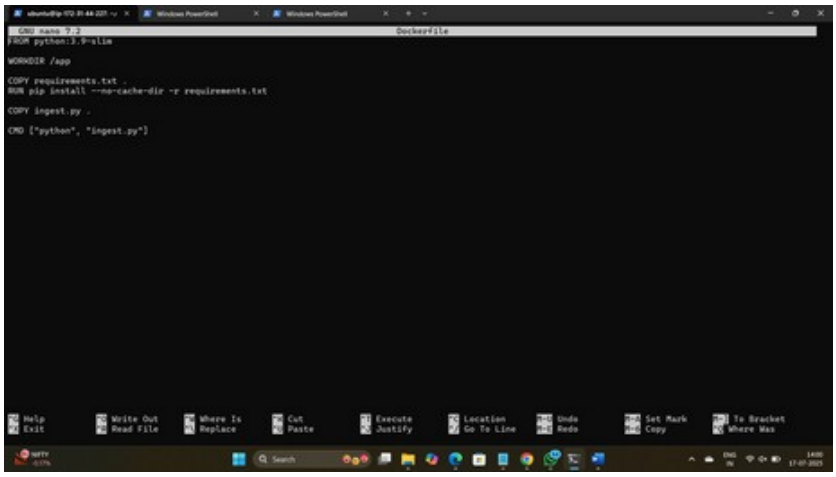
• Read CSV file from S3 using boto3• Parse the data using pandas• Attempt to upload to RDS using SQLAlchemy• If RDS fails, create a table in AWS Glue and register the S3 location





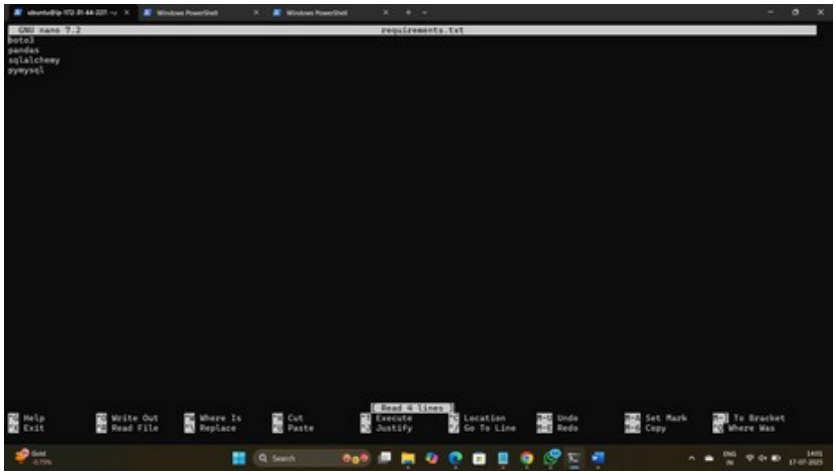
Step 2: Dockerfile Creation

• Use Python 3.9 base image• Install necessary dependencies• Copy Python script into container• Run script on container startup



Step 3: Requirements File

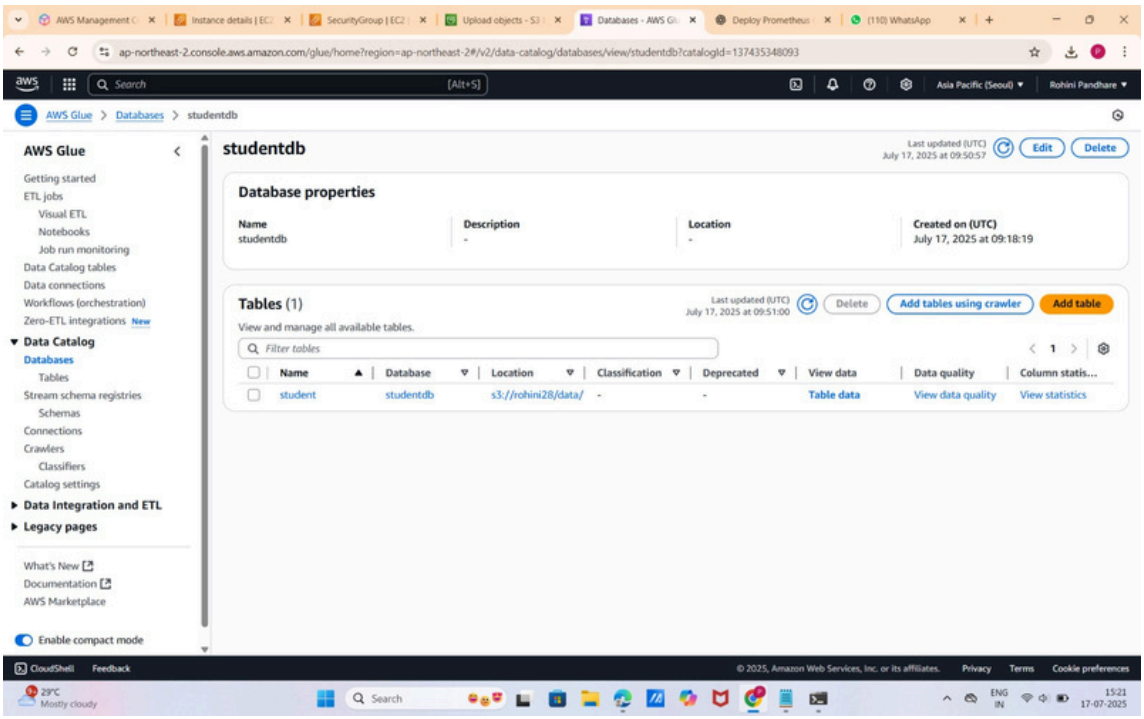
• Include all necessary Python dependencies (boto3, pandas, sqlalchemy, pymysql)



Step 4: Create database

Step 5 : Create S3 bucket

Step 6: Create AWS Glue Database



Step 7: Create IAM role

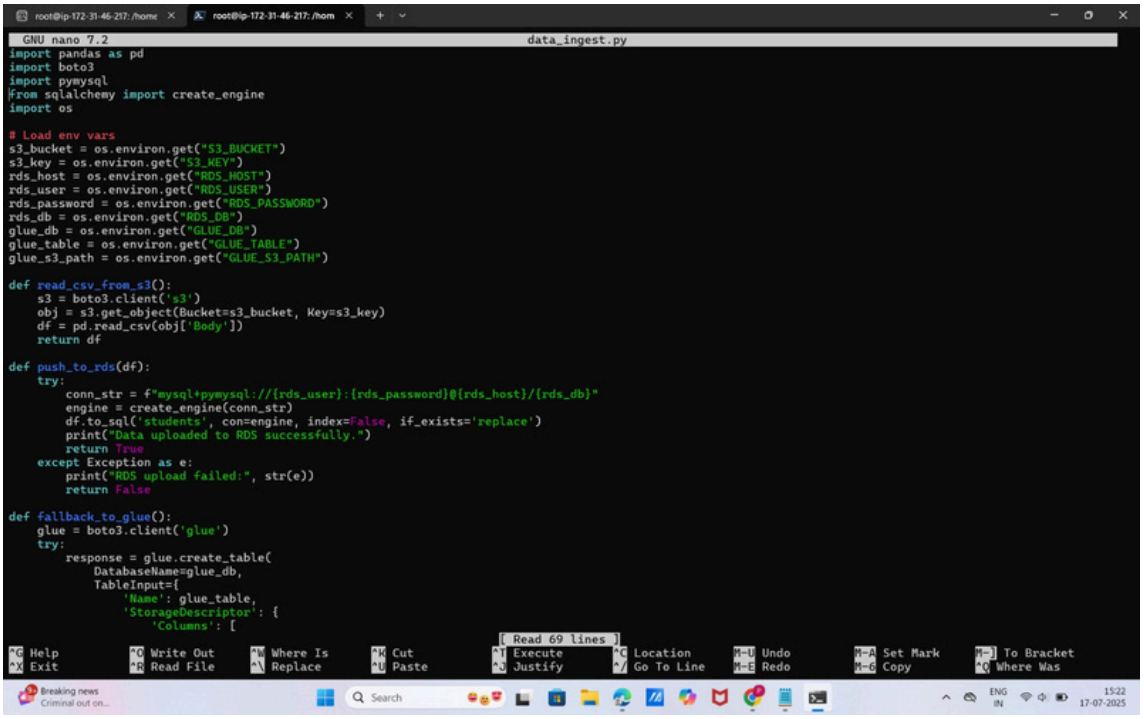
AmazonS3FullAccess

AWSGlueConsoleFullAccess

Step 8: Image Build and Container Run

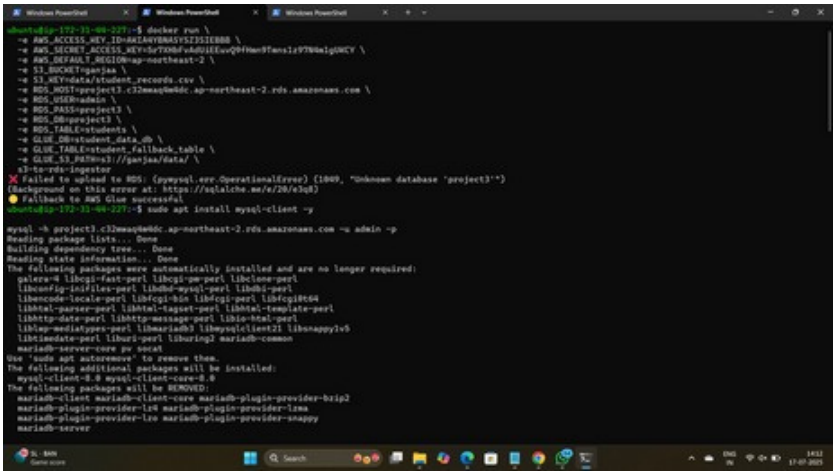
• Build the Docker image using `docker build -t data-ingestor .`• Run the container with AWS credentials passed as environment variables

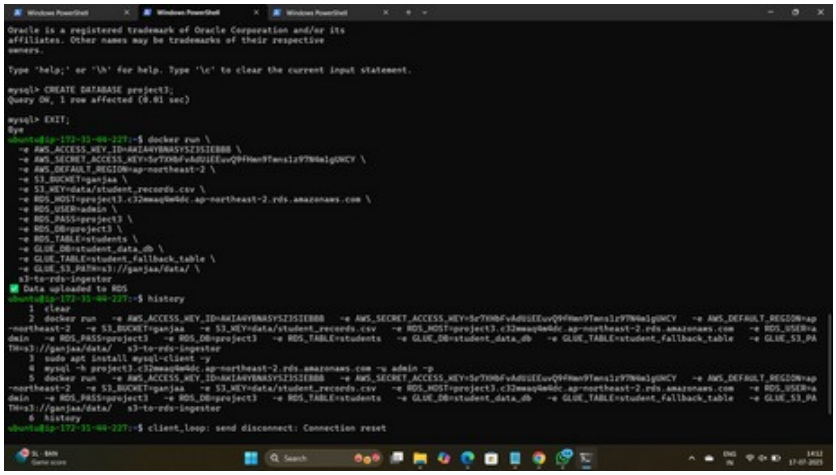
---



Step 9: Data Verification

• Confirm that data was uploaded to RDS (MySQL Workbench or CLI)• If RDS fails, confirm Glue table creation in AWS Glue Data Catalog





7. Results

- Data was successfully read from S3.- Data was inserted into RDS database.- On RDS failure, fallback to AWS Glue succeeded.- Docker logs showed successful execution.[Screenshot Placeholder - Final logs or Glue catalog screen]

8. Conclusion

This project demonstrates a robust, scalable, and automated data ingestion system using cloud-native tools and containerization. The Dockerized Python application ensures consistent deployment, while AWS S3, RDS, and Glue provide scalable storage, compute, and fallback capabilities. The implementation strengthens cloud architecture reliability and fault tolerance.

repository link:

https://github.com/rohinipandhare12/S3-RDS-Glue-Fallback

Name- Rohii Pandhare

Cloud & DevOps Intern Cravita Technologies