AWS-AZURE Unified Data Analytics Pipeline

Know Your Data

User Guide For Data Engineers

Version: V-1.0

Prepared By: Sharawan Kumar Thapa

Table of Contents

Architecture Overview

Step-by-Step Process

- Step 1: Data Input Form
- Step 2: AWS RDS (MySQL)
- Step 3: MySQL Workbench Usage
- Step 4: AWS Glue (ETL)
- Step 5: Amazon S3 – Data Lake
- Step 6: Azure Pipeline - Databricks
- Step 7: Amazon Athena
- Step 8: ODBC Connection to Athena (Power BI Desktop)
- Step 9: Power BI Service – Publishing & Scheduled Refresh
- Step 10: Power Automate – Notifications

In-Progress Tasks and Upcoming version updates [V 2.0]

Introduction

Purpose: This document guides Data Engineers in managing and scaling an end-to-end cloud-based data pipeline, ensuring smooth data flow from data input to business insights through Power BI dashboards.

Scope: The pipeline covers the entire process from data collection to reporting, including data ingestion, storage, transformation, querying, and automated dashboard refresh.

Data Sources: Data are collected from three sources i.e. AWS RDS, Azure services and Manual data ingestion through CSV files.

Technologies involved:
AWS hosted website Input Form → RDS → MySQL Workbench → Glue → S3 Data Lake (AWS RDS + CSV + Azure Service) → Athena → ODBC → Power BI → Power BI Service Schedule → Power Automate

---

Architecture Overview

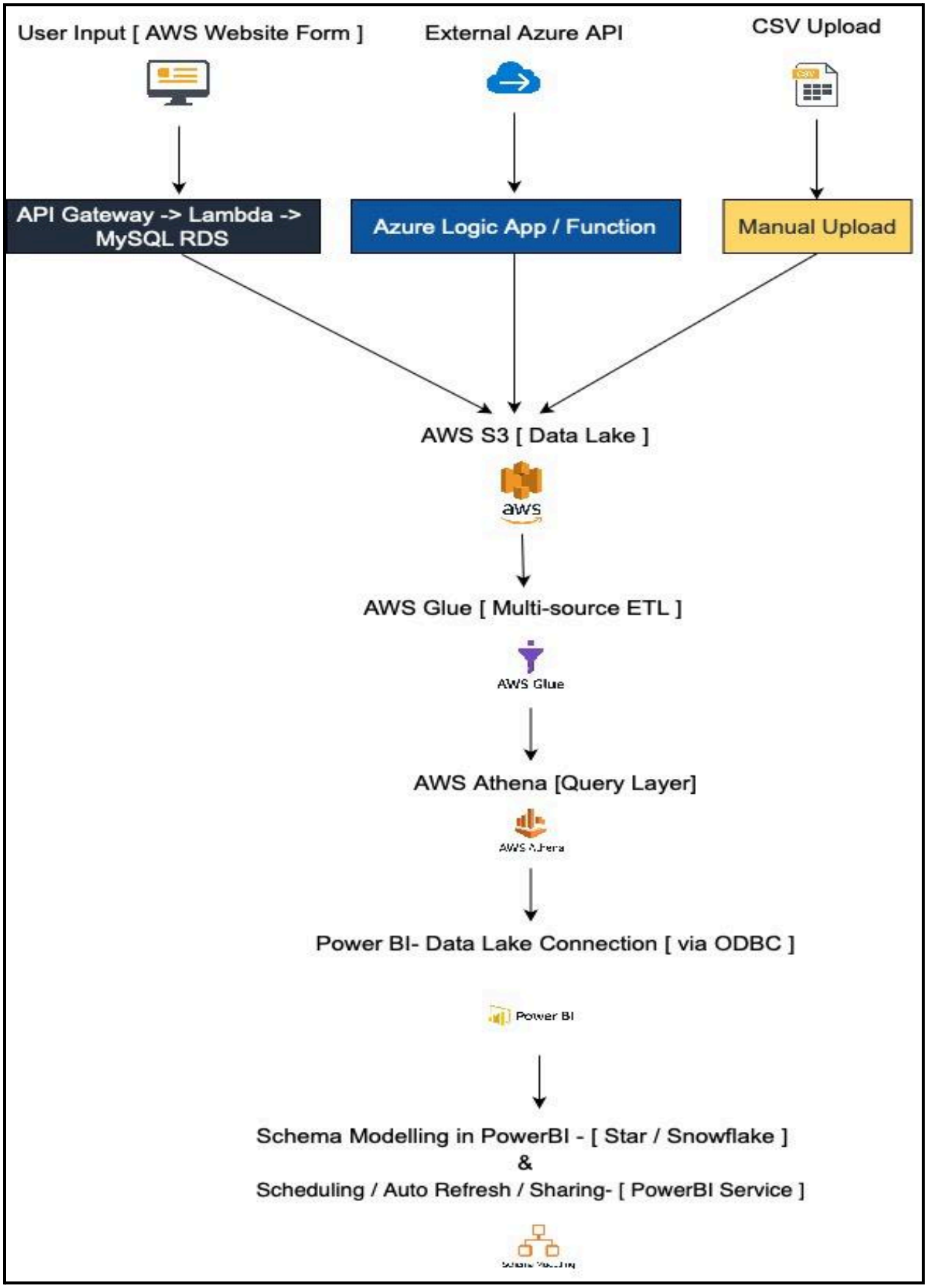

User Input [AWS Website Form]
External Azure API
CSV Upload
API Gateway -> Lambda -> MySQL RDS
Azure Logic App / Function
Manual Upload
AWS S3 [Data Lake]
aws
AWS Glue [Multi-source ETL]
AWS Glue
AWS Athena [Query Layer]
Power BI- Data Lake Connection [via ODBC]
Power BI
Schema Modelling in PowerBI - [Star / Snowflake]
&
Scheduling / Auto Refresh / Sharing- [PowerBI Service]

Figure: Data pipeline Architecture Overview

Step-by-Step Process

Step 1: Data Input Form

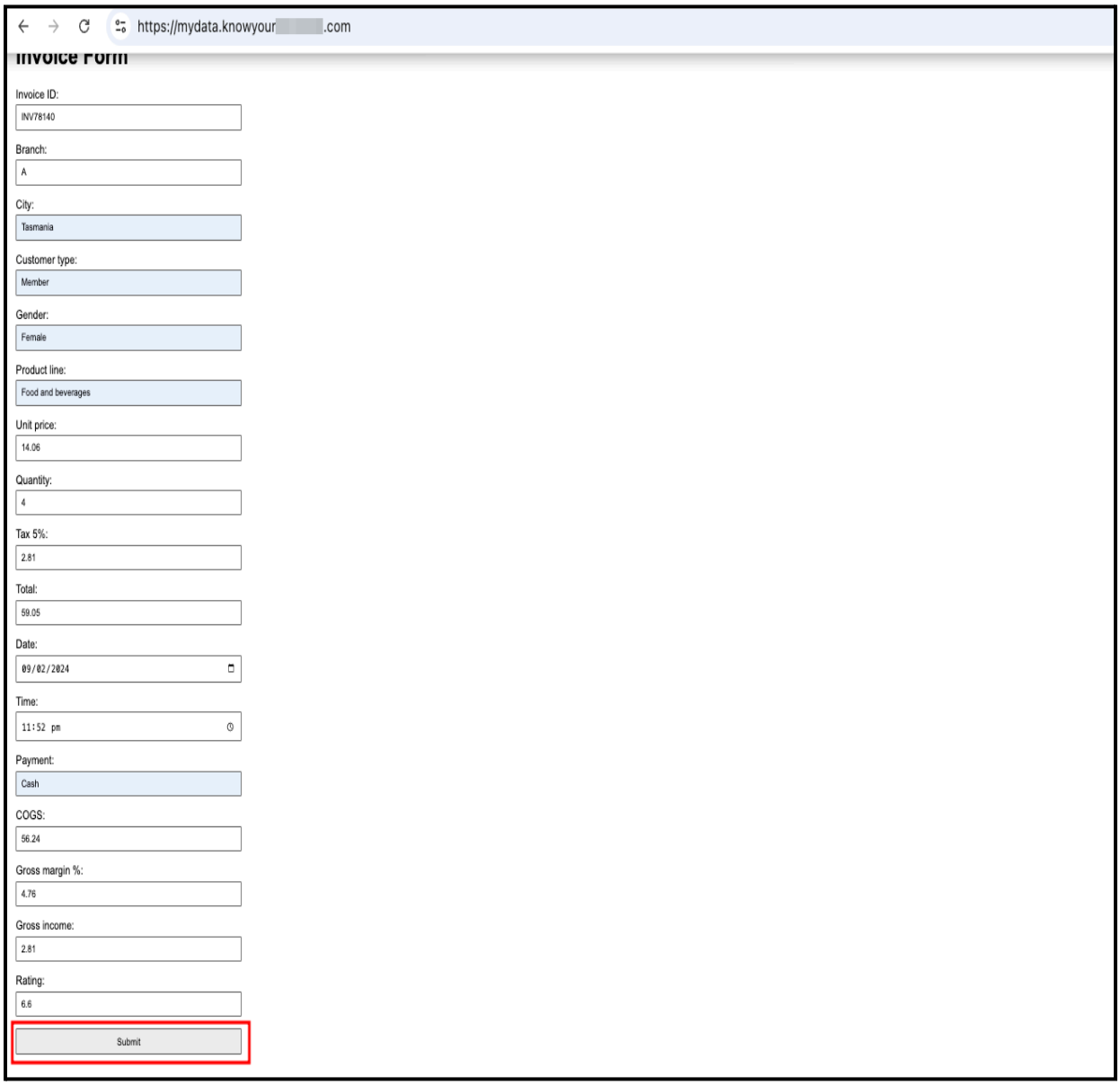
Purpose: Collects user-entered data through a web form hosted on AWS **[** https://mydata.knowyour********.com/ **]** using HTML Form / Web App / API Gateway / Lambda / Route 53

Process:

Created a web form with fields like `Invoice ID`, `Branch`, `Date`, etc.

01. Connect the form backend to RDS using APIs (Lambda + API Gateway).

02. Validate data before writing to RDS.

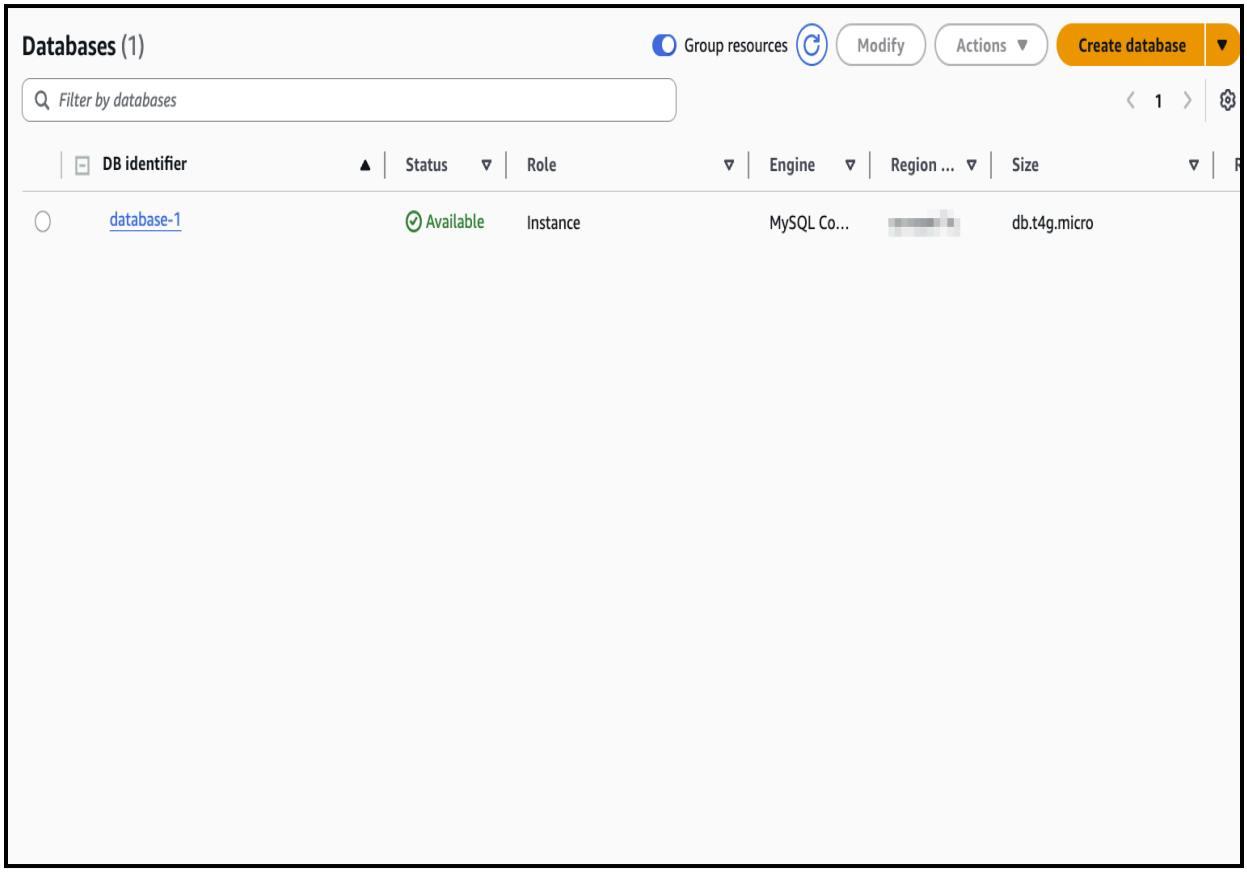


Step 2: AWS RDS (MySQL)

Purpose: Relational database to store raw transactional data.

Process:

01. Set up an RDS instance with MySQL.
02. Configure security groups (allow form/web app to write, Glue to read).
03. Use MySQL Workbench to Query data, Validate entries, Perform backups or schema changes if needed.

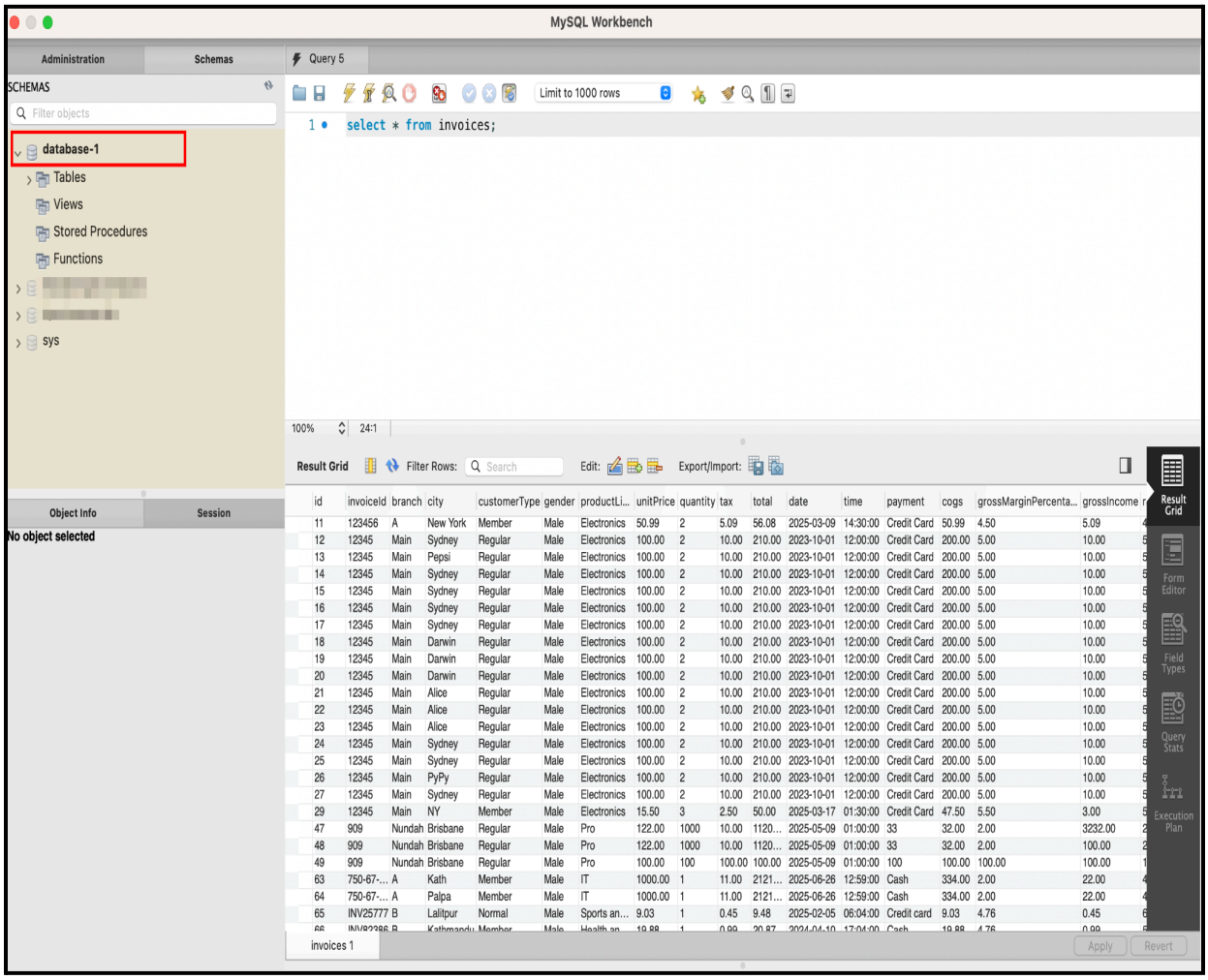


Step 3: MySQL Workbench Usage

Purpose: Connect securely to AWS RDS

Process:

01. Manual queries
02. Troubleshooting
03. Schema updates



Step 4: AWS Glue (ETL)

Purpose: Extract from MySQL → Transform → Load to S3 in Parquet format. Performs ETL (Extract, Transform, Load) processes — moves, cleans, and transforms data into a query-ready format in S3.

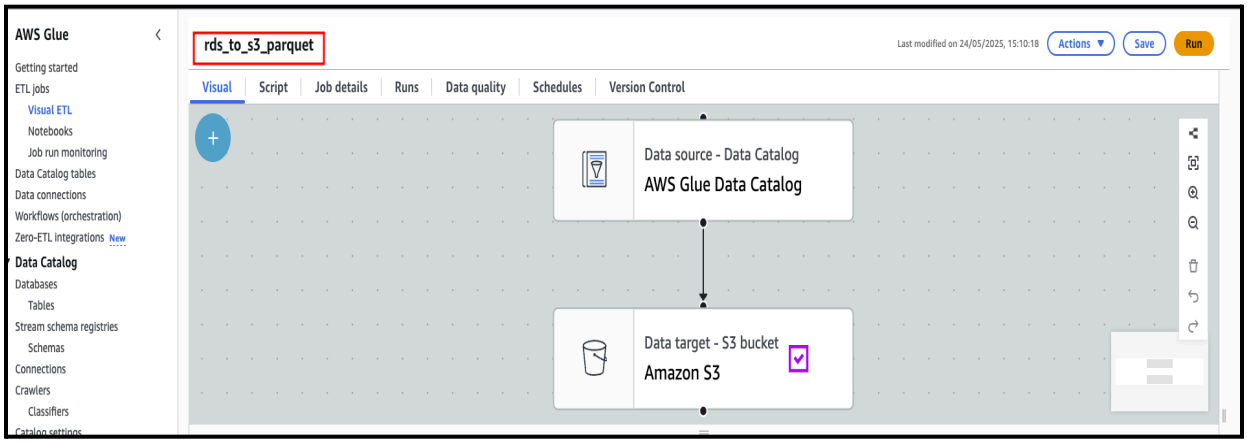
Process:

01. Set up a Glue Crawler to connect to RDS (JDBC Connection).

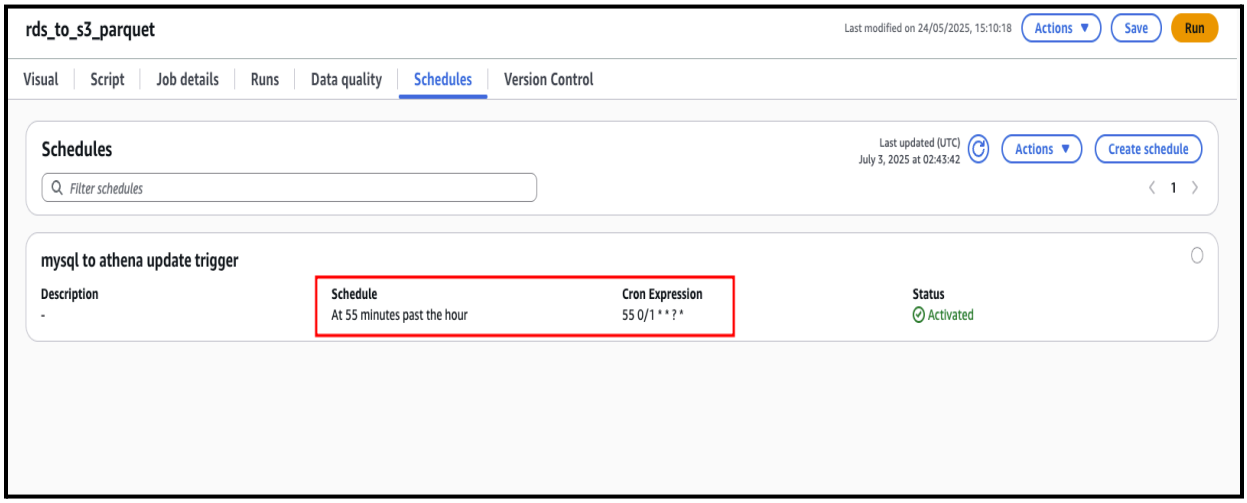
02. Create a Glue Job to export data from RDS to S3:

    a. Output format: Parquet

b. Folder structure: e.g.,
`s3://data-lake/sales_data/year=2025/month=05/`



03. Schedule Glue jobs (hourly, daily, etc.)



Step 5: Amazon S3 – Data Lake

Purpose: Central repository for transformed data in Parquet. Central storage for raw, structured, and semi-structured data from multiple sources [Azure Databricks, AWS RDS, Static CSV upload and unified_data]. Create Glue crawlers that scan S3 folders to create Athena tables.

Process:

Create Folder Structure in data lake:

`s3://data-lake/`

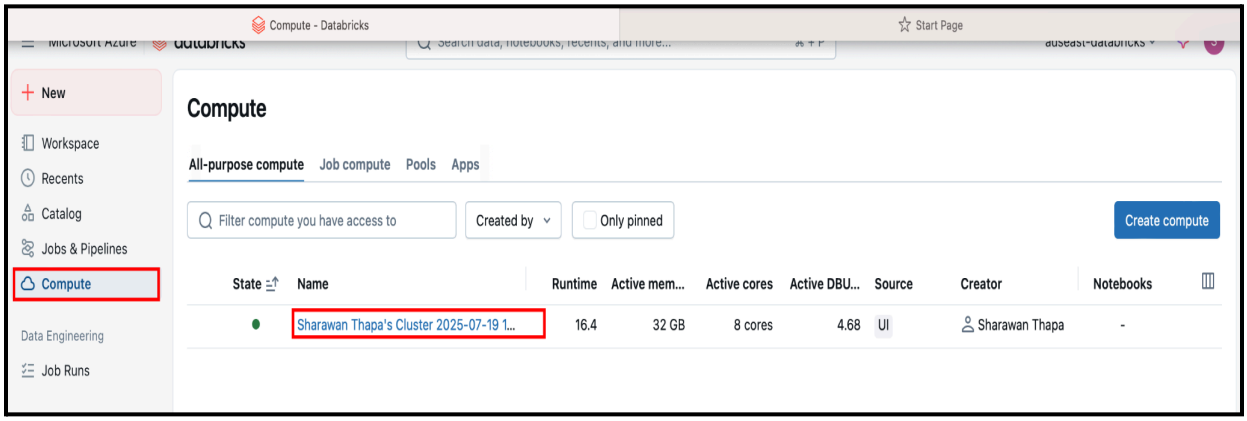
```
└── azure_data/
    └── sales_data_csv/
        └── unified_data/
            └── year=2025/
```

Step 6: Azure Pipeline - Databricks

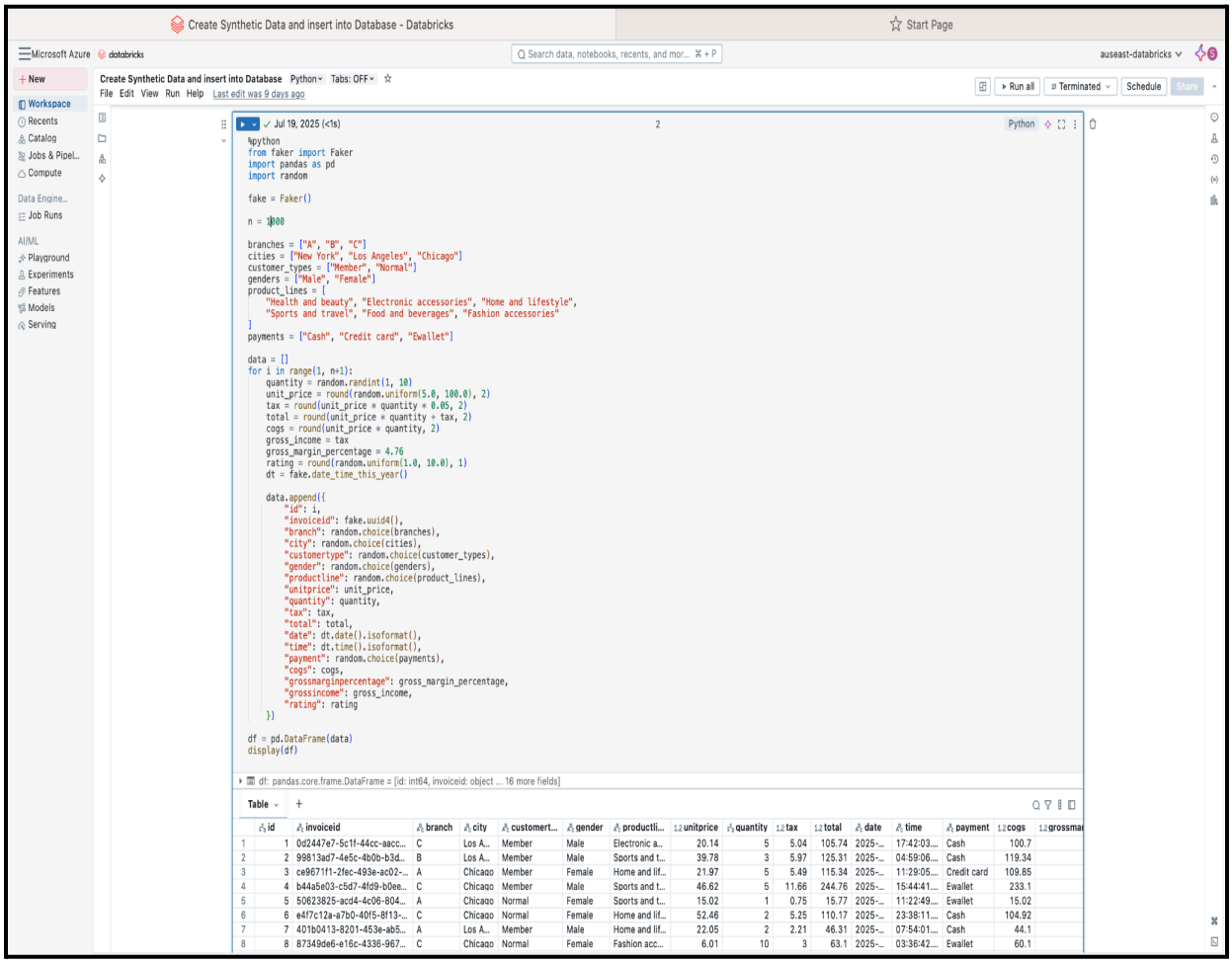
Purpose: To generate, transform, and store structured synthetic data from Azure Databricks into AWS S3 in Parquet format, enabling downstream analytics through AWS Glue and Athena.

Process: Azure Databricks is used to generate realistic sales data with Python and Faker, store it in a table (`syntheticdb.synthetic_data`), and export it to AWS S3 in Parquet format.

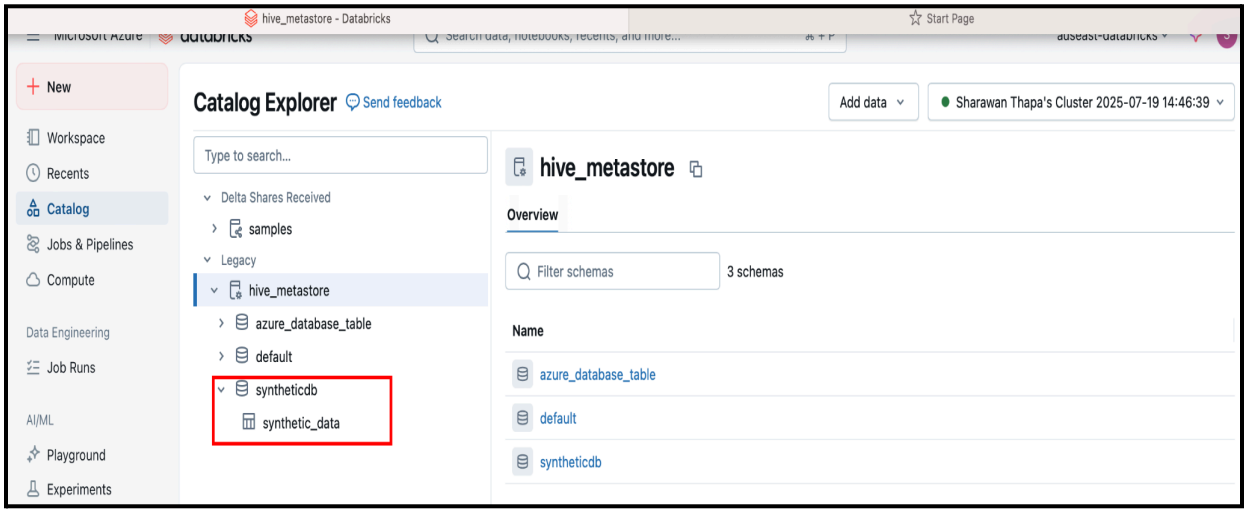
01. **Export to S3** using Spark (`.write.parquet`). **Compute Cluster** created for Azure Databricks.



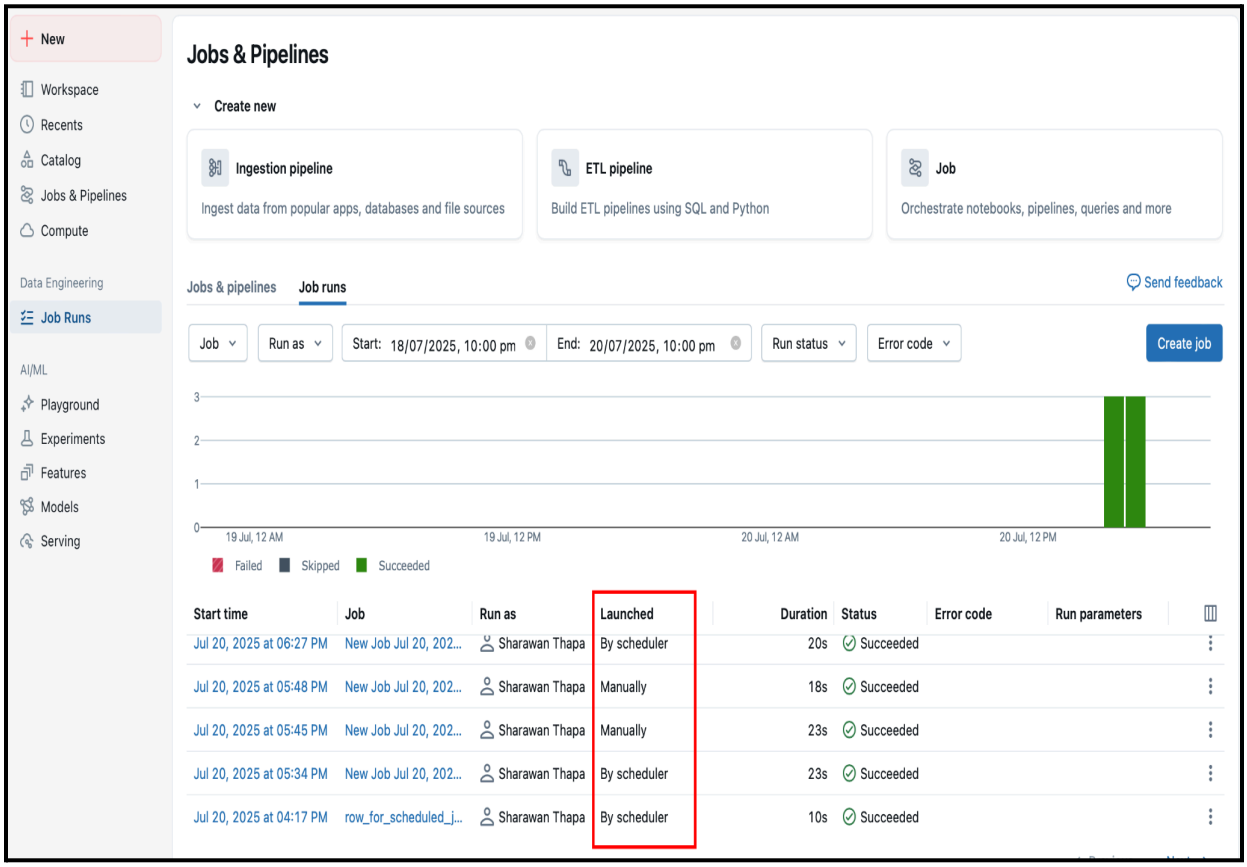
02. Generate realistic, random sales transaction data using **Python and Faker** for testing ETL pipelines and analytics.



03. **Store Data** in a Delta table (`syntheticdb.synthetic_data`) for reuse and querying.



04. **Export to S3** using Spark (`.write.parquet`).



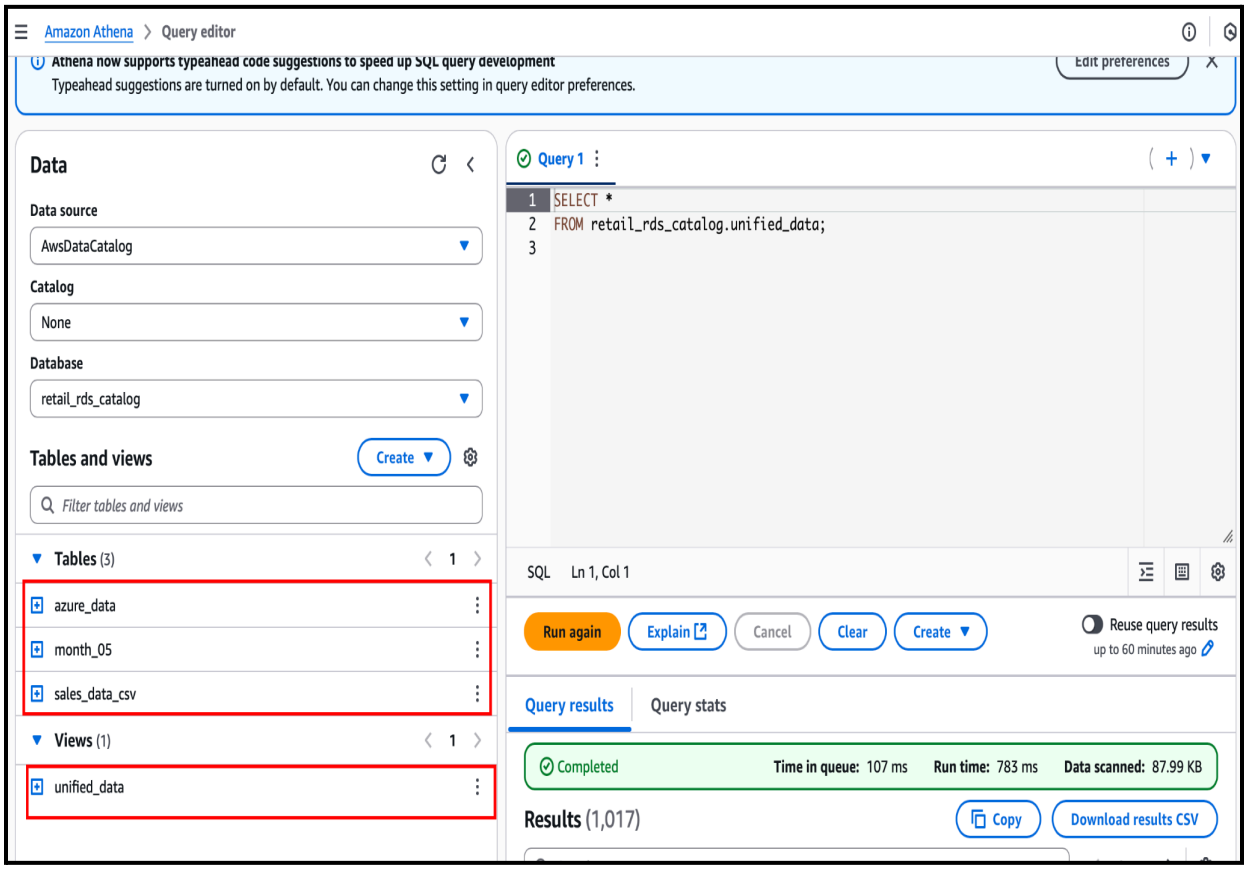
Step 7: Amazon Athena

Purpose: SQL engine to query data directly from S3; acts as the data access layer for reporting tools.

Process:

01. Write SQL queries that read from the S3 Data Lake, and can include filters, exclusions, or even create new “cleaned” tables or views for downstream use.

02. Create views like `unified_data` that merge multiple sources (`month_05 + azure_data sales_data_csv`).



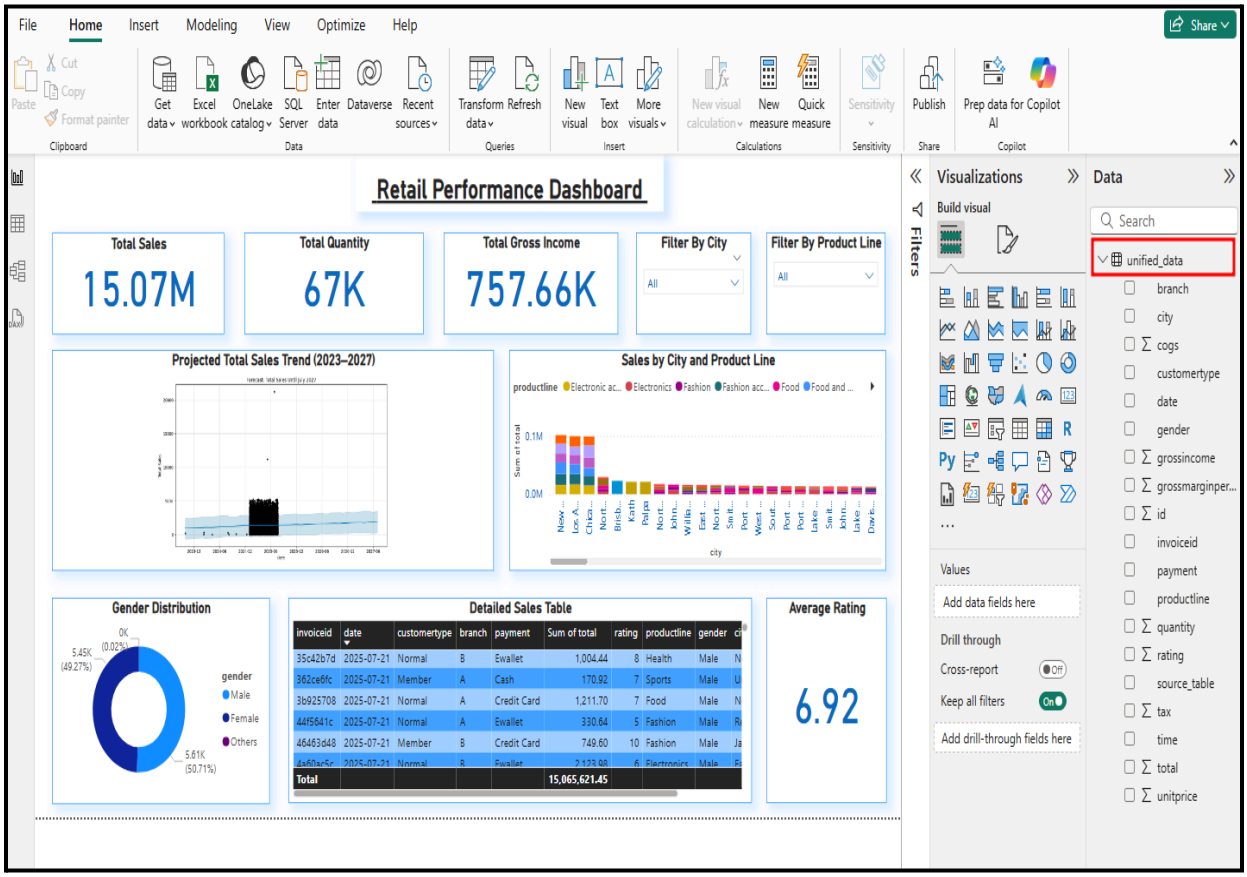
Step 8: ODBC Connection to Athena (Power BI Desktop)

Purpose: Connects to Athena to fetch query results in PowerBI Desktop, enables report building and data visualization.

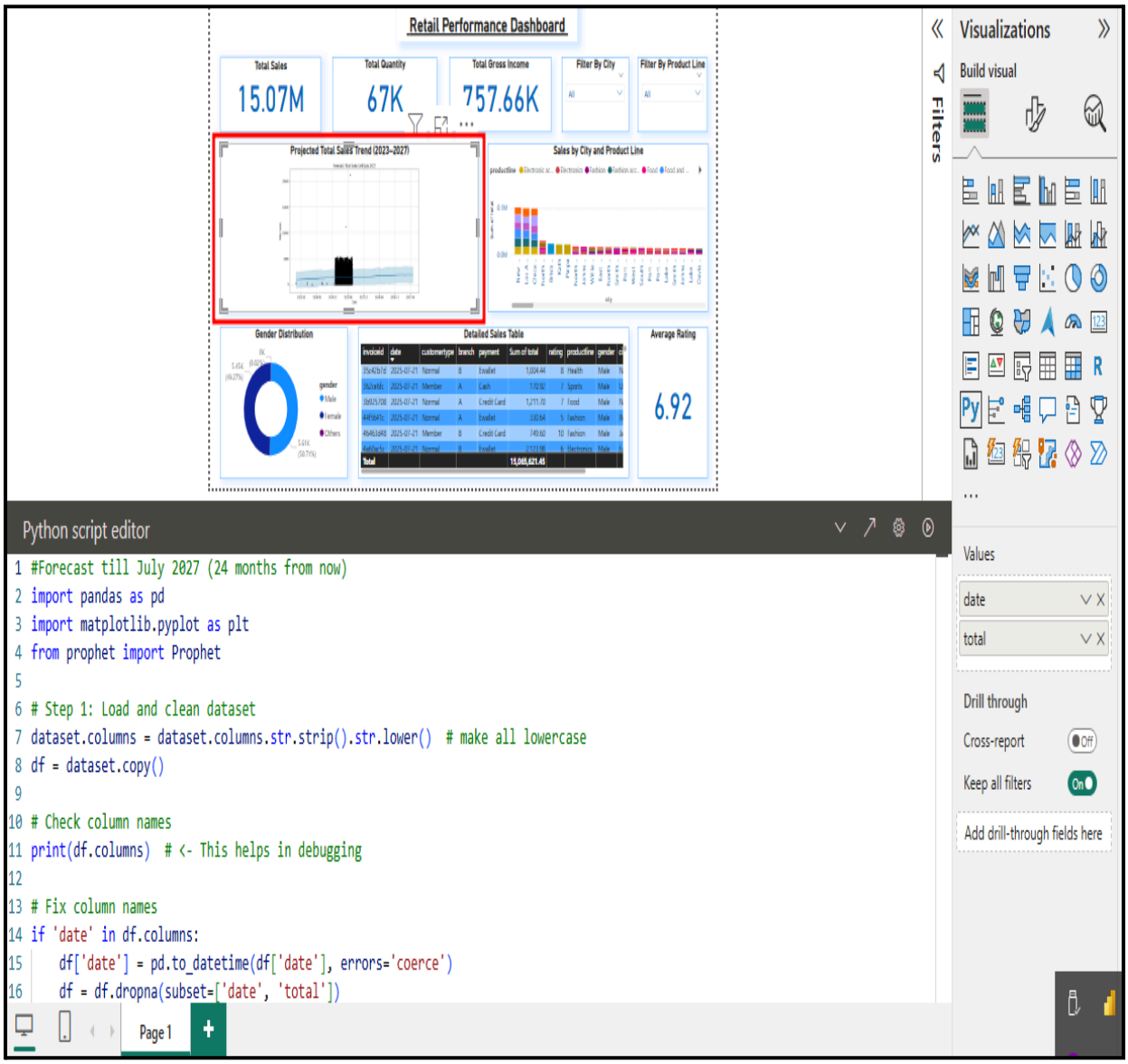
Process:

01. In Power BI: **Get Data → ODBC → Athena DSN → Select tables/views like `unified_data`.**

02. Build reports, charts, KPIs, and forecasts.



03. Built Python predictive modelling to forecast Total Sales up to July 2027 using open-source Prophet library in PowerBI. It processes historical sales data and generates future projections with confidence intervals.



Step 9: Power BI Service – Publishing & Scheduled Refresh

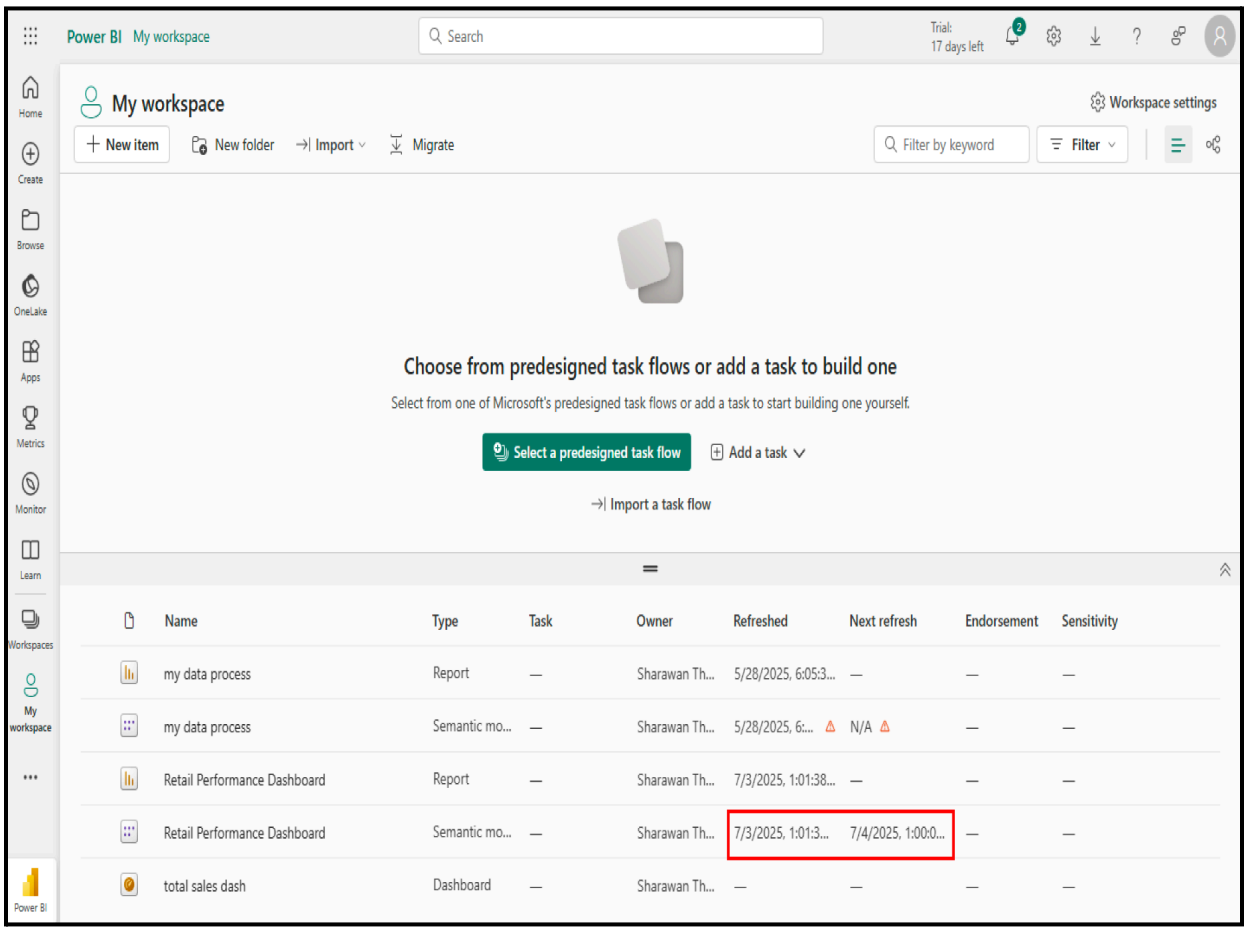
Purpose: To publish reports to Power BI Service for collaboration, and automate data updates through scheduled refresh

Process:

01. Publish `.pbix` report to Power BI Service workspace

02. Configure Athena ODBC credentials in Power BI Service (via Gateway)

03. Set up **Scheduled Refresh**: Frequency (e.g., hourly, daily), Alerts for refresh failures

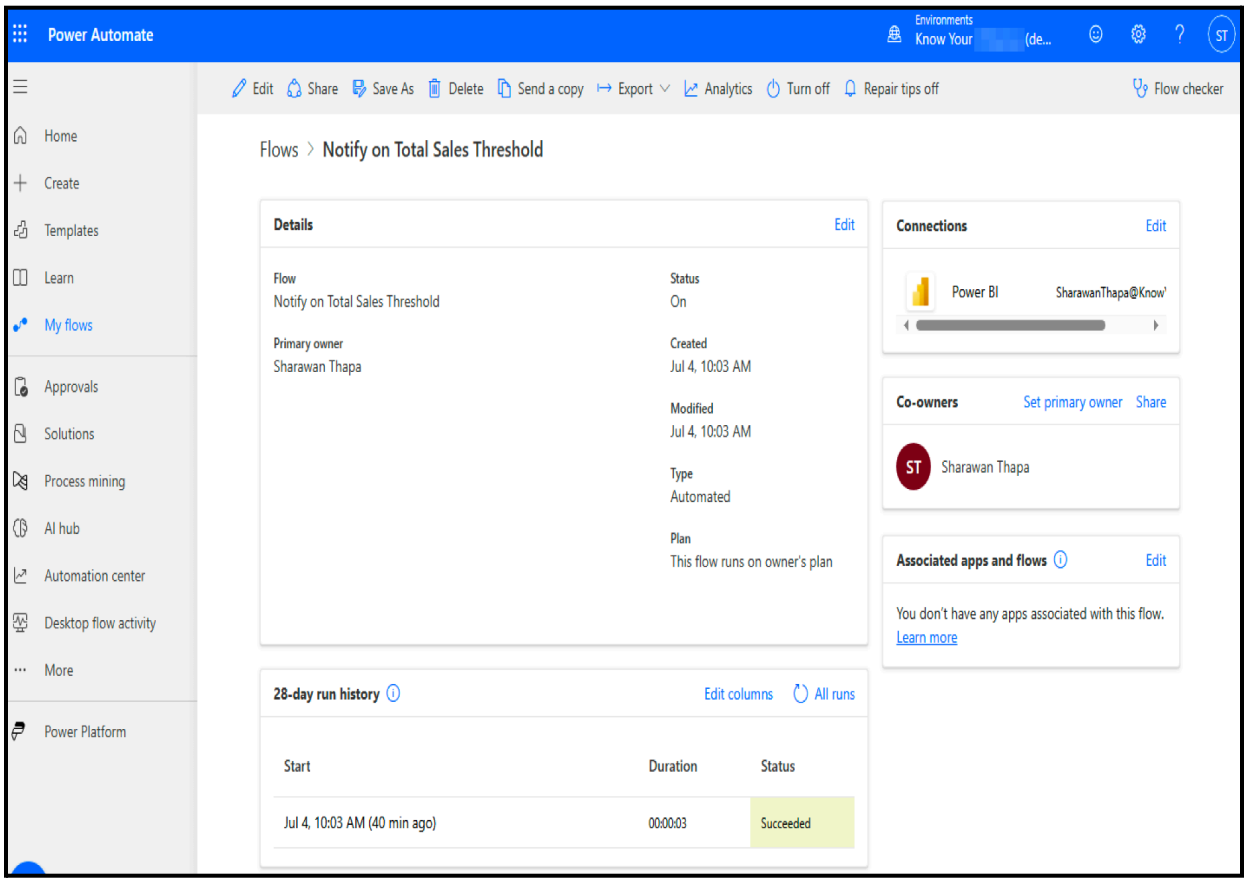


Step 10: Power Automate – Notifications

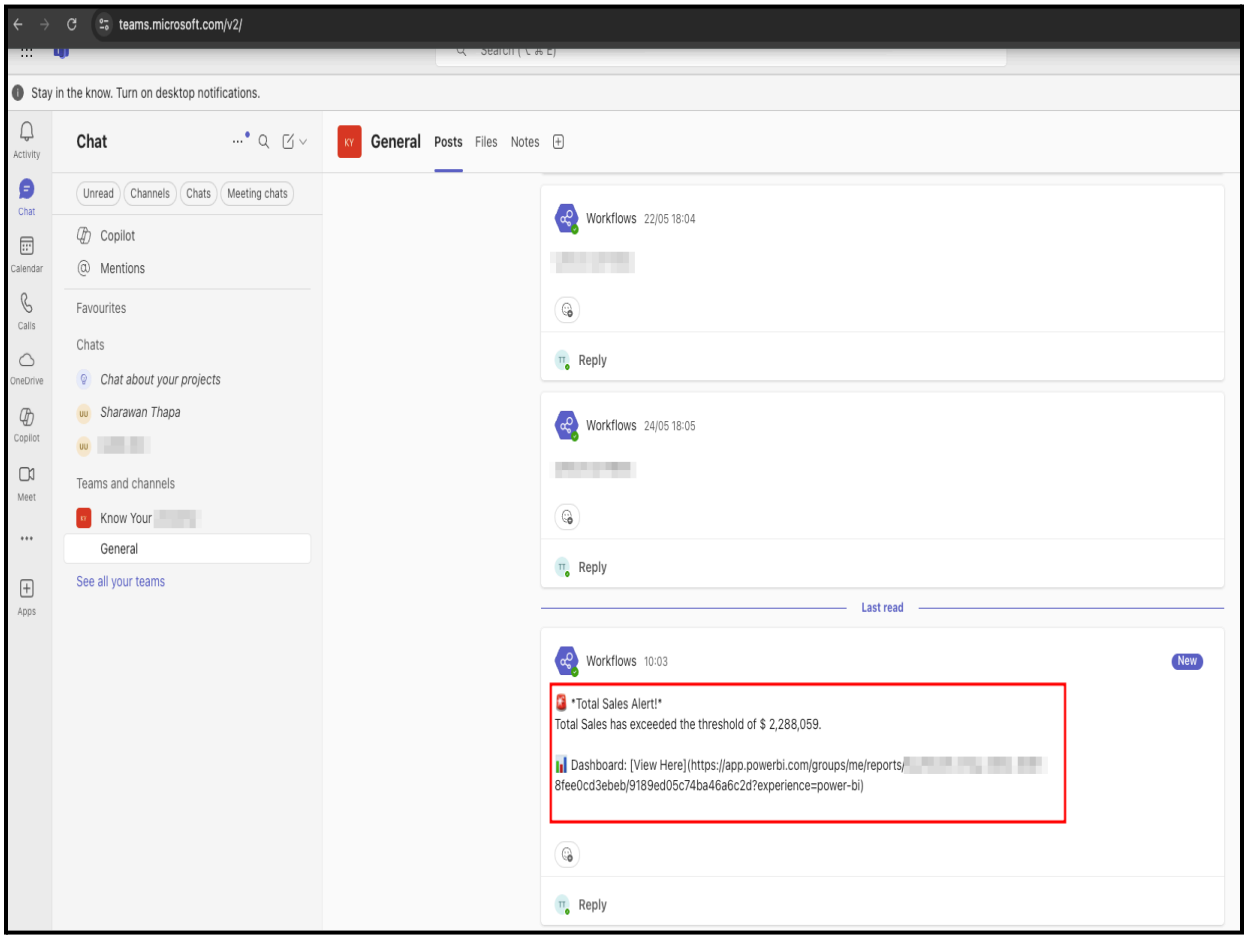
Purpose: Trigger email or Teams notification when Power BI refresh fails when the defined Power BI KPI threshold is exceeded, ensuring timely awareness and action.

Process:

01. Build a flow in Power Automate using the When a Power BI data-driven alert is triggered connector.
02. Configure the flow to post a message to a specified Teams channel or send an email.



03. Include relevant details in the notification, such as current metric value and a link to the Power BI dashboard.



In-Progress Tasks and Upcoming version updates [V 2.0]

---

01. Star/ Snowflake Schema Modelling
02. Real-time streaming pipelines (e.g., Kinesis/Event Hubs → Data Lake)
03. CI/CD pipelines for data workflows (version control & automated tests)