

# **End-to-End Unified Data Analytics Pipeline**



## ***Know Your Data***



**User Guide For Data Engineers**

**Version: V-1.0**

**Prepared By: Sharawan Kumar Thapa**

**Table of Contents**

## Introduction

## 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.

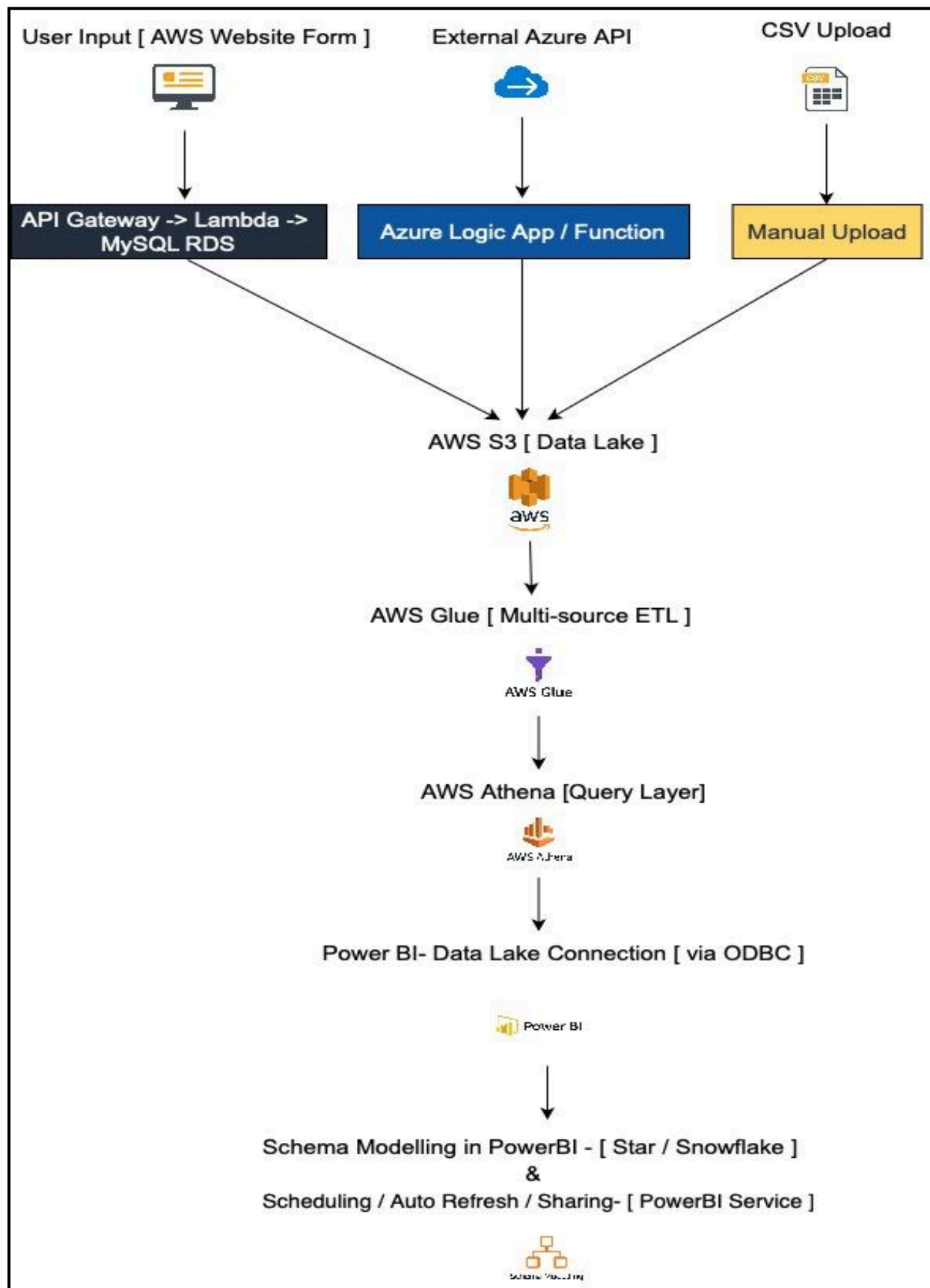
**Data Sources:** Data are collected from three sources i.e. AWS RDS, Azure services and Manual data ingestion through CSV files.

#### **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**



## 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/](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.

← → ↻ https://mydata.knowyour.com

### Invoice Form

Invoice ID:  
INV78140

Branch:  
A

City:  
Tasmania

Customer type:  
Member

Gender:  
Female

Product line:  
Food and beverages

Unit price:  
14.06

Quantity:  
4

Tax 5%:  
2.81

Total:  
59.05

Date:  
09/02/2024

Time:  
11:52 pm

Payment:  
Cash

COGS:  
56.24

Gross margin %:  
4.76

Gross income:  
2.81

Rating:  
6.6

Submit

## **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.

Databases (1)

Group resources

Modify

Actions

Create database

Filter by databases

< 1 >

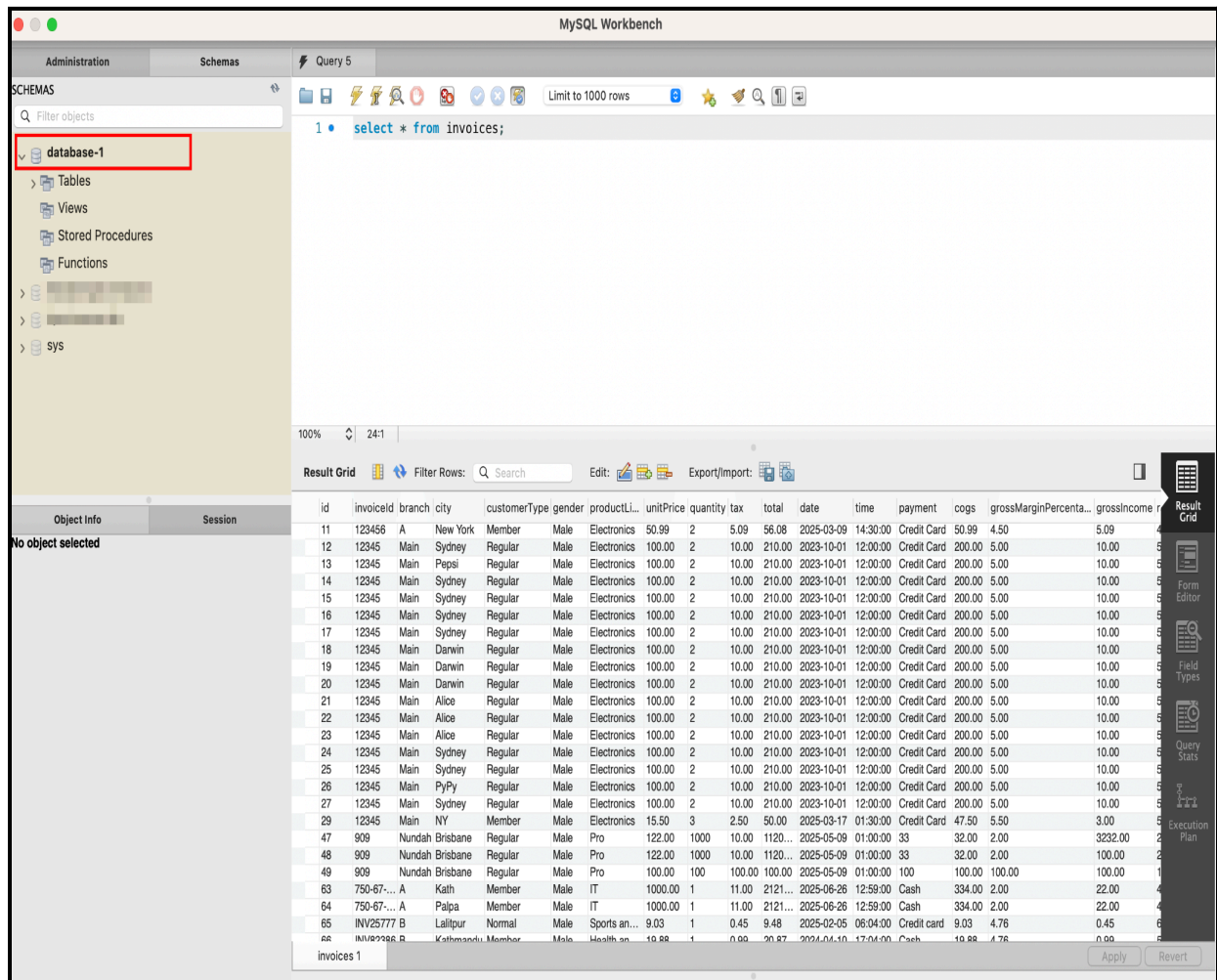
DB identifier	Status	Role	Engine	Region ...	Size
<div><div></div>database-1</div>	<div>Available</div>	Instance	MySQL Co...		db.t4g.micro

### **Step 3: MySQL Workbench Usage**

**Purpose:** Connect securely to AWS RDS

**Process:**

01. Manual queries
02. Troubleshooting
03. Schema updates



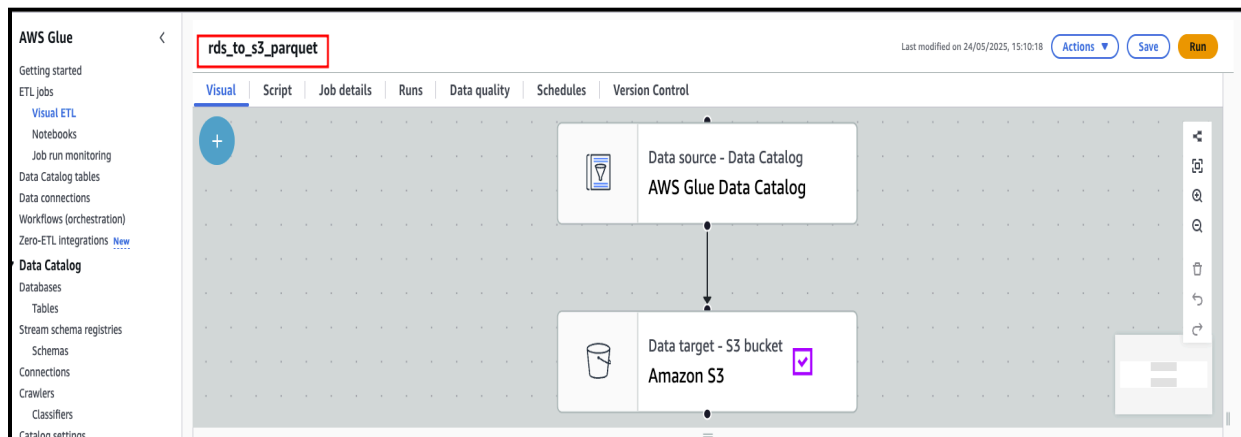
## Step 4: AWS Glue (ETL)

**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.)

The screenshot shows the 'Schedules' tab for the job 'rds\_to\_s3\_parquet'. The schedule is set to 'At 55 minutes past the hour' with a 'Cron Expression' of '55 0/1 \* \* \* \*'. The status is 'Activated'.

Description	Schedule	Cron Expression	Status
mysql to athena update trigger	At 55 minutes past the hour	55 0/1 * * * *	Activated

## 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.

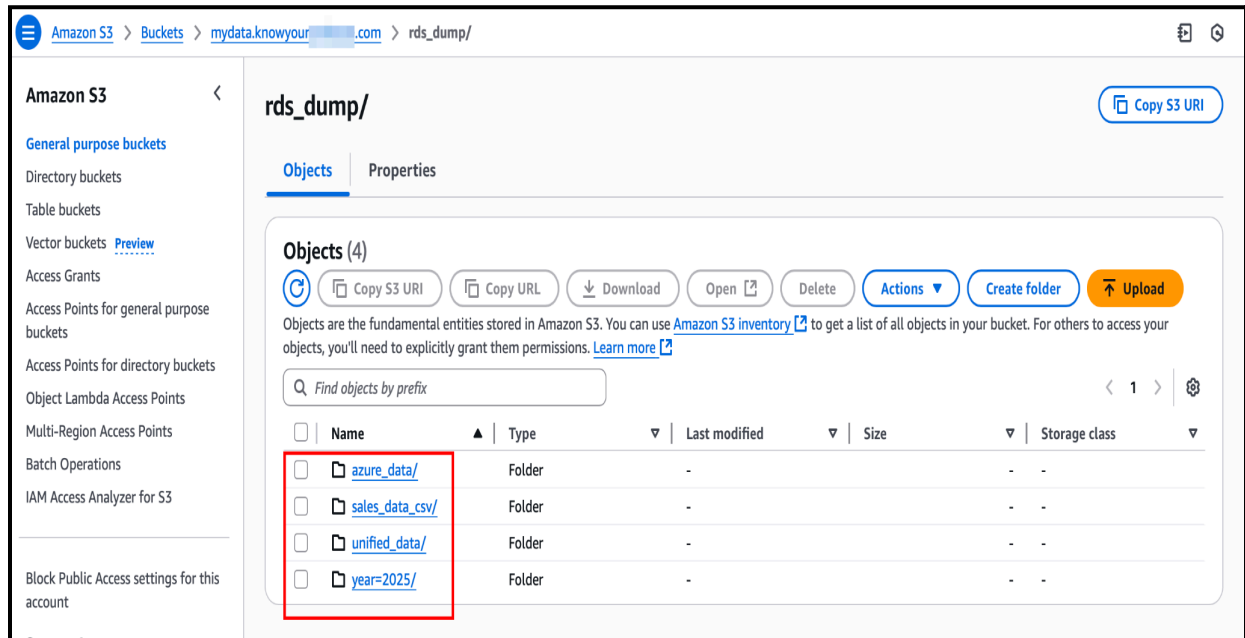
### Process:

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.

Microsoft Azure Databricks Compute - Databricks

Search data, notebooks, recent, and more...

Start Page

+ New

- Workspace
- Recents
- Catalog
- Jobs & Pipelines
- Compute**
- Data Engineering
- Job Runs

### Compute

All-purpose compute Job compute Pools Apps

Filter compute you have access to Created by Only pinned Create compute

State	Name	Runtime	Active mem...	Active cores	Active DBU...	Source	Creator	Notebooks
●	Sharawan Thapa's Cluster 2025-07-19 1...	16.4	32 GB	8 cores	4.68	UI	Sharawan Thapa	-

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

Microsoft Azure Databricks Create Synthetic Data and insert into Database - Databricks

Search data, notebooks, recent, and mor... P

Start Page

+ New

- Workspace
- Recents
- Catalog
- Jobs & Pipel...
- Compute
- Data Engine...
- Job Runs
- AI/ML
- Playground
- Experiments
- Features
- Models
- Serving

### Create Synthetic Data and insert into Database

Python Tabs: OFF

File Edit View Run Help Last edit was 9 days ago

Run all Terminated Schedule Share

```
python
from faker import Faker
import pandas as pd
import random

fake = Faker()

n = 1000

branches = ["A", "B", "C"]
cities = ["New York", "Los Angeles", "Chicago"]
customer_types = ["Member", "Normal"]
genders = ["Male", "Female"]
product_lines = [
    "Health and beauty", "Electronic accessories", "Home and lifestyle",
    "Sports and travel", "Food and beverages", "Fashion accessories"
]
payments = ["Cash", "Credit card", "Ewallet"]

data = []
for i in range(1, n+1):
    quantity = random.randint(1, 10)
    unit_price = round(random.uniform(5.0, 100.0), 2)
    tax = round(unit_price * quantity * 0.05, 2)
    total = round(unit_price * quantity + tax, 2)
    cogs = round(unit_price * quantity, 2)
    gross_income = tax
    gross_margin_percentage = 4.76
    rating = round(random.uniform(1.0, 10.0), 1)
    dt = fake.date_time_this_year()

    data.append({
        "id": i,
        "invoiceid": fake.uuid4(),
        "branch": random.choice(branches),
        "city": random.choice(cities),
        "customer_type": random.choice(customer_types),
        "gender": random.choice(genders),
        "productline": random.choice(product_lines),
        "unitprice": unit_price,
        "quantity": quantity,
        "tax": tax,
        "total": total,
        "date": dt.date().isoformat(),
        "time": dt.time().isoformat(),
        "payment": random.choice(payments),
        "cogs": cogs,
        "grossmarginpercentage": gross_margin_percentage,
        "grossincome": gross_income,
        "rating": rating
    })

df = pd.DataFrame(data)
display(df)
```

df: pandas.core.frame.DataFrame = [id: int64, invoiceid: object ... 16 more fields]

	id	invoiceid	branch	city	customer...	gender	productli...	unitprice	quantity	tax	total	date	time	payment	cogs	grossma...
1	0e2447e7-5c1f-44cc-acc...	C	Los A...	Member	Male	Electronic a...	20.14	5	5.04	105.74	2025...	17:42:03...	Cash	100.7		
2	9e613ad7-4e5c-4b0b-b3d...	B	Los A...	Member	Male	Sports and L...	39.78	3	5.97	125.31	2025...	04:59:06...	Cash	119.34		
3	ce9671f1-2fec-493a-ac02...	A	Chicago	Member	Female	Home and lif...	21.97	5	5.49	115.34	2025...	11:29:05...	Credit card	109.85		
4	b44a5e03-c5d7-4f59-b0ee...	C	Chicago	Member	Male	Sports and L...	46.62	5	11.66	244.76	2025...	15:44:41...	Ewallet	233.1		
5	50623825-acd4-4c06-804...	A	Chicago	Normal	Female	Sports and L...	15.02	1	0.75	15.77	2025...	11:22:49...	Ewallet	15.02		
6	e47c12a-e7b0-40f5-8f13...	C	Chicago	Normal	Female	Home and lif...	52.46	2	5.25	110.17	2025...	23:38:11...	Cash	104.92		
7	401b0413-8201-453a-ab5...	A	Los A...	Member	Male	Home and lif...	22.05	2	2.21	46.31	2025...	07:54:01...	Cash	44.1		
8	87349d6e-e16c-4336-967...	C	Chicago	Normal	Female	Fashion acc...	6.01	10	3	63.1	2025...	03:36:42...	Ewallet	60.1		

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

The screenshot shows the Databricks Catalog Explorer interface. On the left sidebar, the 'Catalog' tab is selected. Under 'Legacy', the 'hive\_metastore' folder is expanded, and the 'synthetic\_data' table is highlighted with a red box. The main panel shows the 'Overview' of the 'hive\_metastore' with a search bar and a list of schemas: 'azure\_database\_table', 'default', and 'syntheticdb'.

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

The screenshot shows the Databricks Jobs & Pipelines interface. The 'Job runs' tab is selected, displaying a timeline of job runs. A red box highlights the 'Launched' column in the table below. The table lists several job runs, all of which were 'Succeeded'.

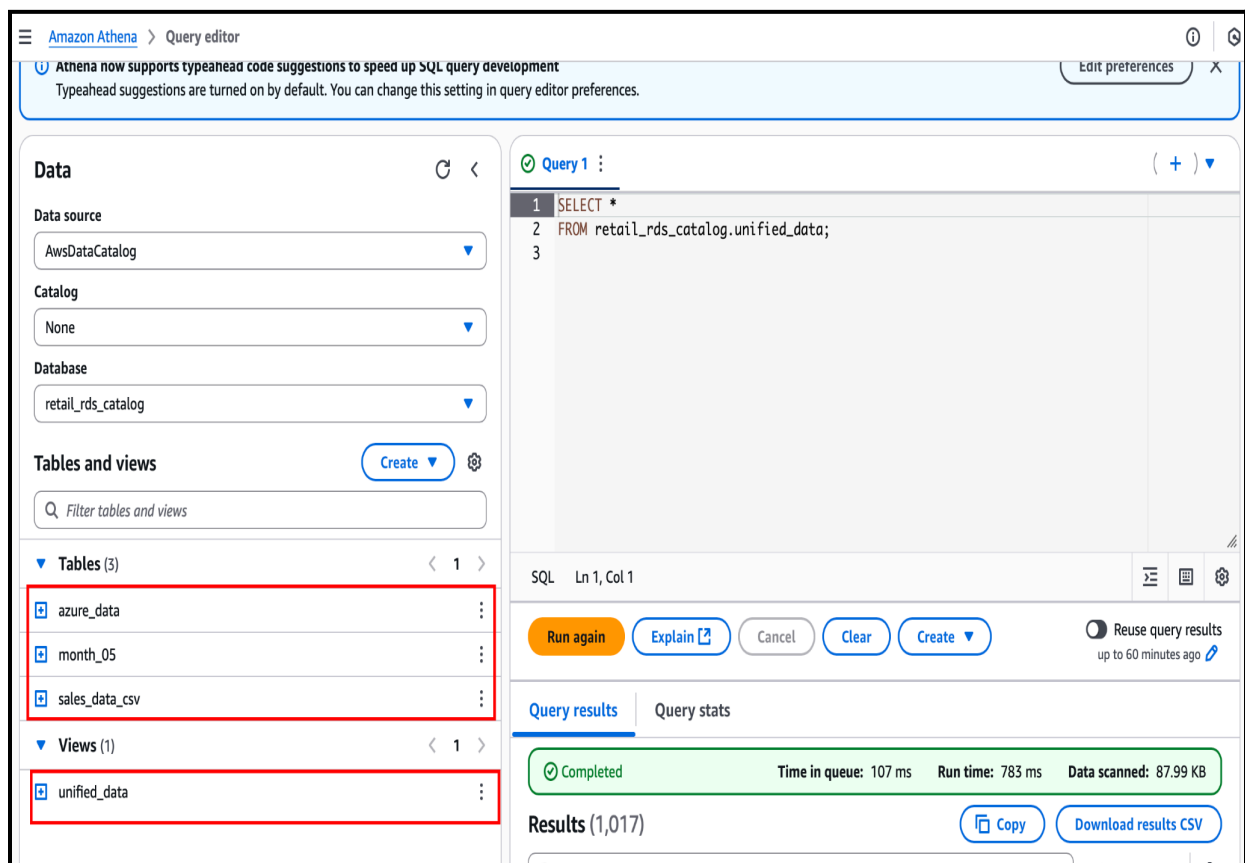
Start time	Job	Run as	Launched	Duration	Status	Error code	Run parameters
Jul 20, 2025 at 06:27 PM	New Job Jul 20, 202...	Sharawan Thapa	By scheduler	20s	✓ Succeeded		
Jul 20, 2025 at 05:48 PM	New Job Jul 20, 202...	Sharawan Thapa	Manually	18s	✓ Succeeded		
Jul 20, 2025 at 05:45 PM	New Job Jul 20, 202...	Sharawan Thapa	Manually	23s	✓ Succeeded		
Jul 20, 2025 at 05:34 PM	New Job Jul 20, 202...	Sharawan Thapa	By scheduler	23s	✓ Succeeded		
Jul 20, 2025 at 04:17 PM	row_for_scheduled_j...	Sharawan Thapa	By scheduler	10s	✓ Succeeded		

## 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`).



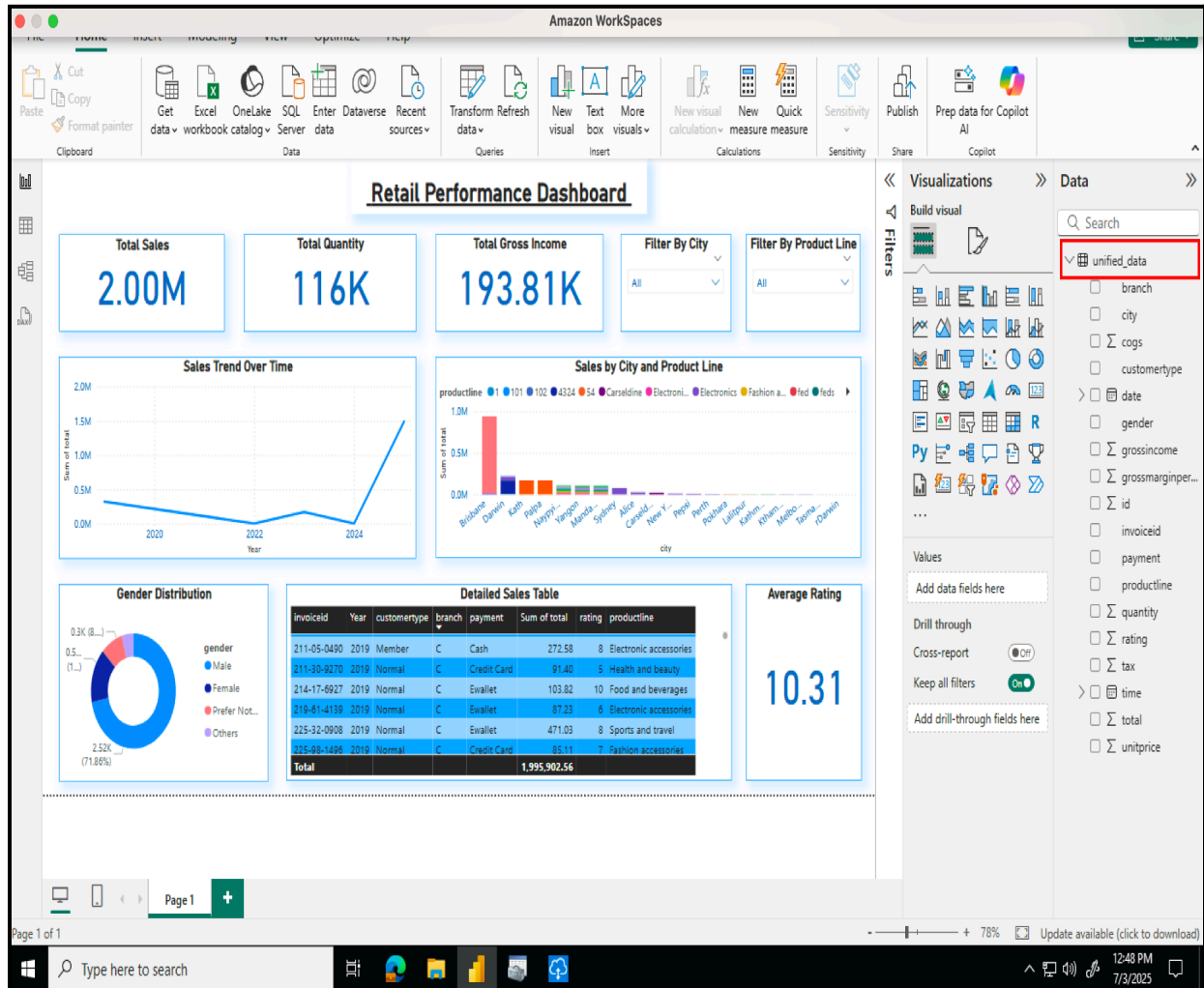
## Step 8: ODBC Connection to Athena (Power BI Desktop)

**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.



## 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

The screenshot shows the Power BI 'My workspace' interface. At the top, there's a search bar and a 'Workspace settings' link. Below the search bar, there are buttons for 'New item', 'New folder', 'Import', and 'Migrate'. A 'Filter by keyword' input and a 'Filter' dropdown are also present. The main content area has a heading 'Choose from predesigned task flows or add a task to build one' and a subtext 'Select from one of Microsoft's predesigned task flows or add a task to start building one yourself.' Below this, there are two buttons: 'Select a predesigned task flow' and 'Add a task'. A link 'Import a task flow' is also visible. At the bottom, there's a table with columns: Name, Type, Task, Owner, Refreshed, Next refresh, Endorsement, and Sensitivity. The table lists several tasks, including 'my data process', 'Retail Performance Dashboard', and 'total sales dash'. The 'Next refresh' column for the 'Retail Performance Dashboard' task is highlighted with a red box, showing '7/4/2025, 1:00:00...'.

Name	Type	Task	Owner	Refreshed	Next refresh	Endorsement	Sensitivity
my data process	Report	—	Sharawan Th...	5/28/2025, 6:05:3...	—	—	—
my data process	Semantic mo...	—	Sharawan Th...	5/28/2025, 6:...	N/A	—	—
Retail Performance Dashboard	Report	—	Sharawan Th...	7/3/2025, 1:01:38...	—	—	—
Retail Performance Dashboard	Semantic mo...	—	Sharawan Th...	7/3/2025, 1:01:3...	7/4/2025, 1:00:0...	—	—
total sales dash	Dashboard	—	Sharawan Th...	—	—	—	—

## 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.

Power Automate

Environments: Know Your (de...)

ST

Edit Share Save As Delete Send a copy Export Analytics Turn off Repair tips off Flow checker

Flows > Notify on Total Sales Threshold

**Details** Edit

Flow	Notify on Total Sales Threshold	Status	On
Primary owner	Sharawan Thapa	Created	Jul 4, 10:03 AM
		Modified	Jul 4, 10:03 AM
		Type	Automated
		Plan	This flow runs on owner's plan

**Connections** Edit

Power BI SharawanThapa@Know'

Co-owners Set primary owner Share

ST Sharawan Thapa

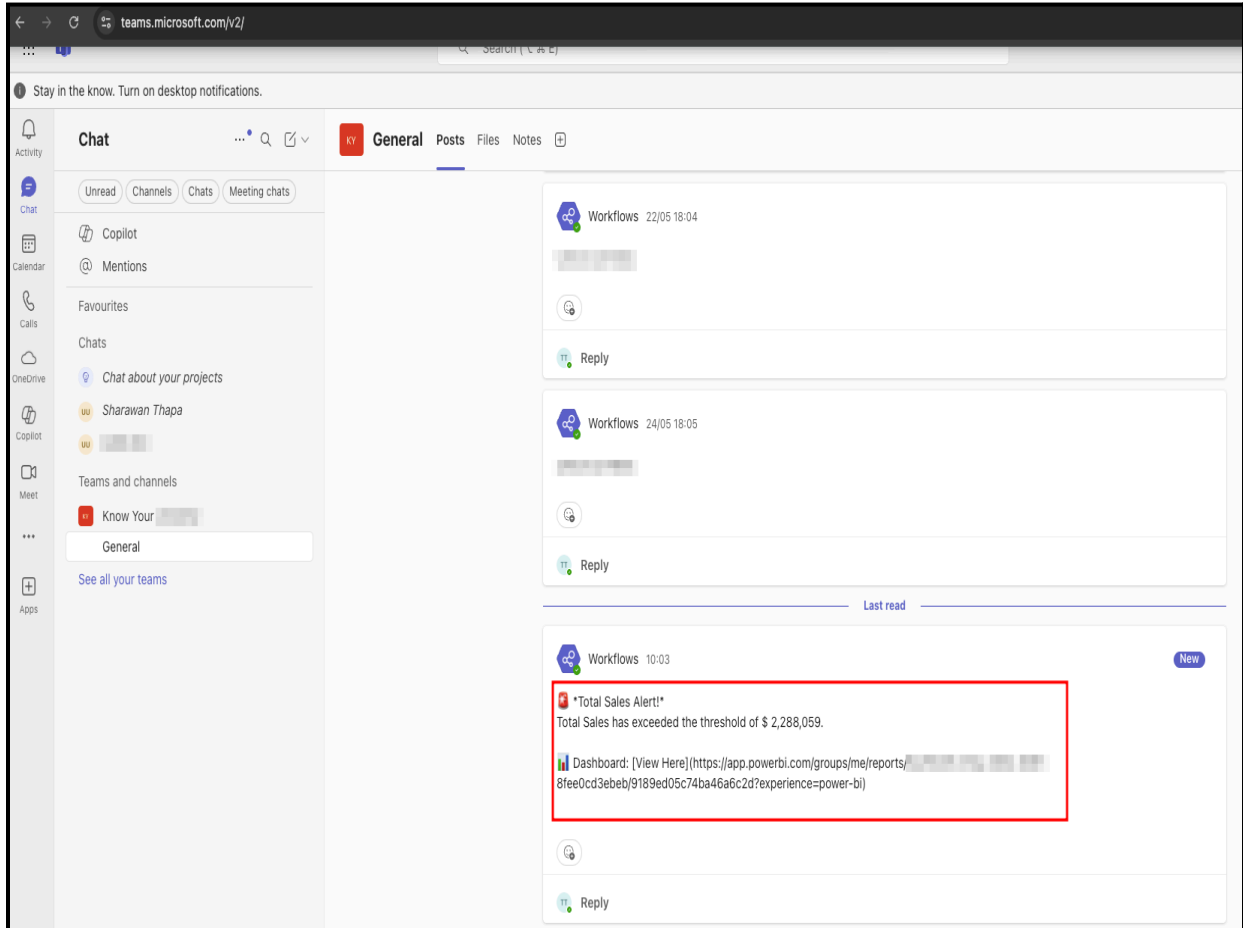
**Associated apps and flows** Edit

You don't have any apps associated with this flow. [Learn more](#)

**28-day run history** Edit columns All runs

Start	Duration	Status
Jul 4, 10:03 AM (40 min ago)	00:00:03	Succeeded

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