

Data pipeline using S3,RDS,Glue

The goal is to aggregate customer debit card purchases on a daily basis and update the aggregated data in a MySQL table hosted on Amazon RDS. I used AWS S3 for data storage, AWS Glue for data processing, and Amazon RDS for data persistence.

Step 1: Generate data and store in S3 using Hive style partition

1. Generate mock daily transaction data and store it in CSV files.
2. Upload daily transaction CSV files to an AWS S3 bucket using a Hive-style partition.

```
C: > Users > Rachana > generate_transactions.py > ...
1  import csv
2  import random
3  from datetime import datetime, timedelta
4
5  names = ["Alice", "Bob", "Charlie", "Diana"]
6  card_types = ["Visa", "MasterCard", "Amex"]
7  banks = ["Bank A", "Bank B", "Bank C"]
8
9  def generate_data(date_str, num_records=10):
10     filename = f"transactions_{date_str}.csv"
11     with open(filename, mode="w", newline="") as file:
12         writer = csv.writer(file)
13         writer.writerow([
14             "customer_id", "name", "debit_card_number",
15             "debit_card_type", "bank_name", "transaction_date", "amount_spend"
16         ])
17         for i in range(num_records):
18             writer.writerow([
19                 random.randint(1000, 9999), # customer_id
20                 random.choice(names), # name
21                 str(random.randint(4000000000000000, 4999999999999999)), # card number
22                 random.choice(card_types), # card type
23                 random.choice(banks), # bank
24                 date_str, # transaction_date
25                 round(random.uniform(10, 500), 2) # amount_spend
26             ])
27     print(f"{filename} created.")
28
29 # Generate files for the last 3 days
30 today = datetime.now()
31 for i in range(3):
32     date_str = (today - timedelta(days=i)).strftime("%Y-%m-%d")
33     generate_data(date_str, 20)
```

[debit-transactions-aug](#) > [raw_data/](#)

raw_data/

Objects

Properties

Objects (3)

Copy S3 URI

Copy URL

Download

Open

Delete

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size
<input type="checkbox"/>	<div><div></div>date=2025-08-18/</div>	Folder	-	
<input type="checkbox"/>	<div><div></div>date=2025-08-19/</div>	Folder	-	
<input type="checkbox"/>	<div><div></div>date=2025-08-20/</div>	Folder	-	

Step 2: Set up a MySQL table in Amazon RDS to store aggregated transaction data.

1. Go to AWS Console → RDS → Create database.
2. Choose:
 - Engine: MySQL
 - Create username/password.
3. Once created, note the endpoint (e.g., mydb.xxxxx.us-east-1.rds.amazonaws.com).
4. Connect via MySQL Workbench and create the table:

```
CREATE DATABASE transactions_db;
```

```
USE transactions_db;
```

```
CREATE TABLE aggregated_transactions (
```

```
customer_id INT,
```

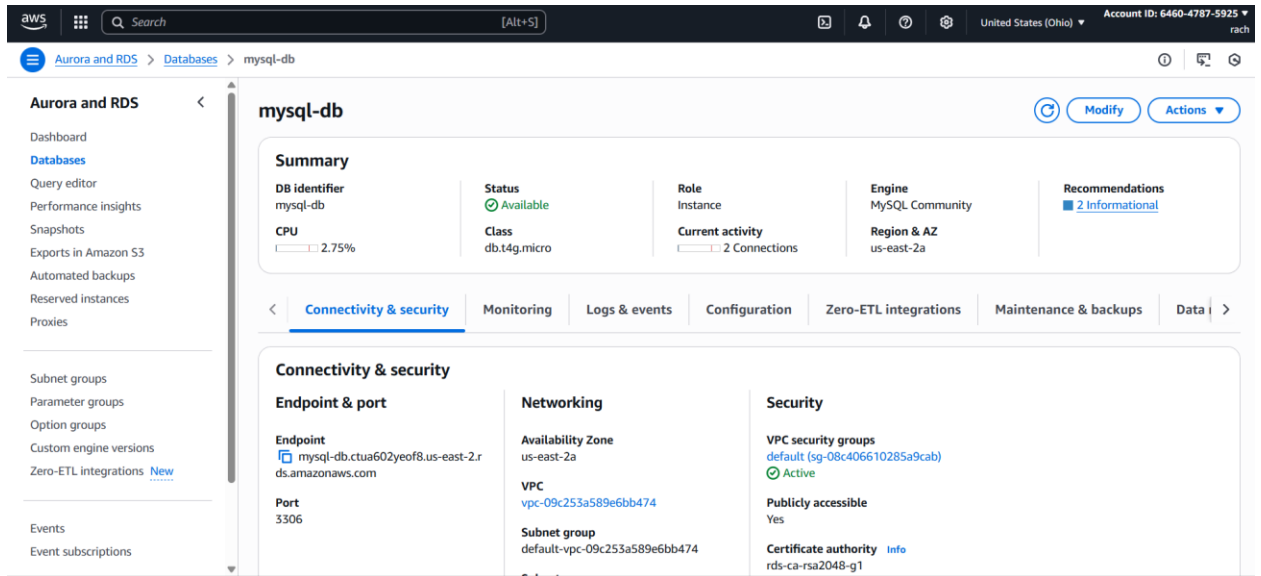
```
debit_card_number VARCHAR(20),
```

```
bank_name VARCHAR(50),
```

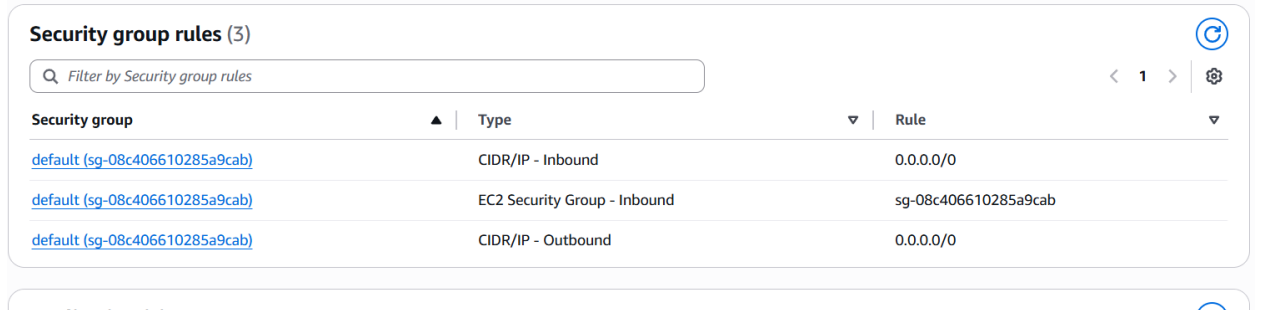
```
total_amount_spend DECIMAL(10,2),
```

PRIMARY KEY (customer_id, debit_card_number)

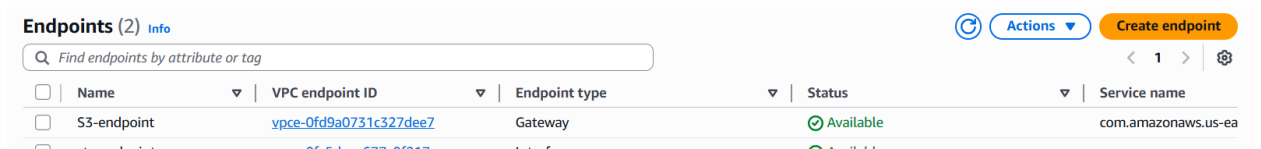
);



5. Make sure security group of rds has port 3306 open for inbound requests



6. Also create an endpoint for S3 in RDS VPC



Step 3: Write an AWS Glue job to process daily transactions from S3, aggregate them, and update the RDS MySQL table

1. Set up AWS Glue:

- First make sure glue has all necessary IAM permissions

Permissions policies (8) [Info](#)

You can attach up to 10 managed policies.

Filter by Type

Search

All types

<input type="checkbox"/>	Policy name	Type	Attached entities
<input type="checkbox"/>	AdministratorAccess	AWS managed - job function	1
<input type="checkbox"/>	AmazonRDSDataFullAccess	AWS managed	1
<input type="checkbox"/>	AmazonRDSFullAccess	AWS managed	1
<input type="checkbox"/>	AmazonRedshiftAllCommandsFullAccess	AWS managed	1
<input type="checkbox"/>	AmazonRedshiftFullAccess	AWS managed	1
<input type="checkbox"/>	AmazonS3FullAccess	AWS managed	5
<input type="checkbox"/>	AWSGlueConsoleFullAccess	AWS managed	1
<input type="checkbox"/>	SecretsManagerReadWrite	AWS managed	1

2. Go to Glue → Create Crawler (point to S3 bucket).

3. Let it create a Glue Data Catalog table.

AWS Glue > **Crawlers** > **sales-data-s3**

Getting started
ETL jobs
Visual ETL
Notebooks
Job run monitoring
Data Catalog tables
Data connections
Workflows (orchestration)
Zero-ETL integrations [New](#)

▼ Data Catalog
Databases
Tables
Stream schema registries
Schemas
Connections
Crawlers
Classifiers
Catalog settings

► Data Integration and ETL
► Legacy pages

What's New

sales-data-s3

Last updated (UTC)
August 20, 2025 at 02:57:22

[Run crawler](#) [Edit](#) [Delete](#)

Crawler properties

Name sales-data-s3	IAM role for_glue	Database sales-metadata-db	State READY
Description -	Security configuration -	Lake Formation configuration -	Table prefix s3_input_
Maximum table threshold -			

► Advanced settings

Crawler runs | Schedule | Data sources | Classifiers | Tags

Crawler runs (1)

The list of crawler runs for this crawler.

Filter data

Filter by a date and time range

<input type="radio"/>	Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
<input type="radio"/>	August 19, 2025 at 01:43...	August 19, 2025 at 01:43...	45 s	Completed	0.139	-

Announcing new optimization features for Apache Iceberg tables
Optimize storage for Apache Iceberg tables with automatic snapshot retention and orphan file deletion. [Learn more](#)

Tables

A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target in a job definition.

Tables (1)

Last updated (UTC)
August 20, 2025 at 02:55:40

[Delete](#) [Add tables using crawler](#) [Add table](#)

View and manage all available tables.

Filter tables

<input type="checkbox"/>	Name	Database	Location	Classification	Deprecated	View data	Data quality	Column statis...
<input type="checkbox"/>	s3_input_raw_data	sales-metadata-db	s3://sales-data-tele	CSV	-	Table data	View data quality	View statistics

Data Catalog showing schema from glue crawler

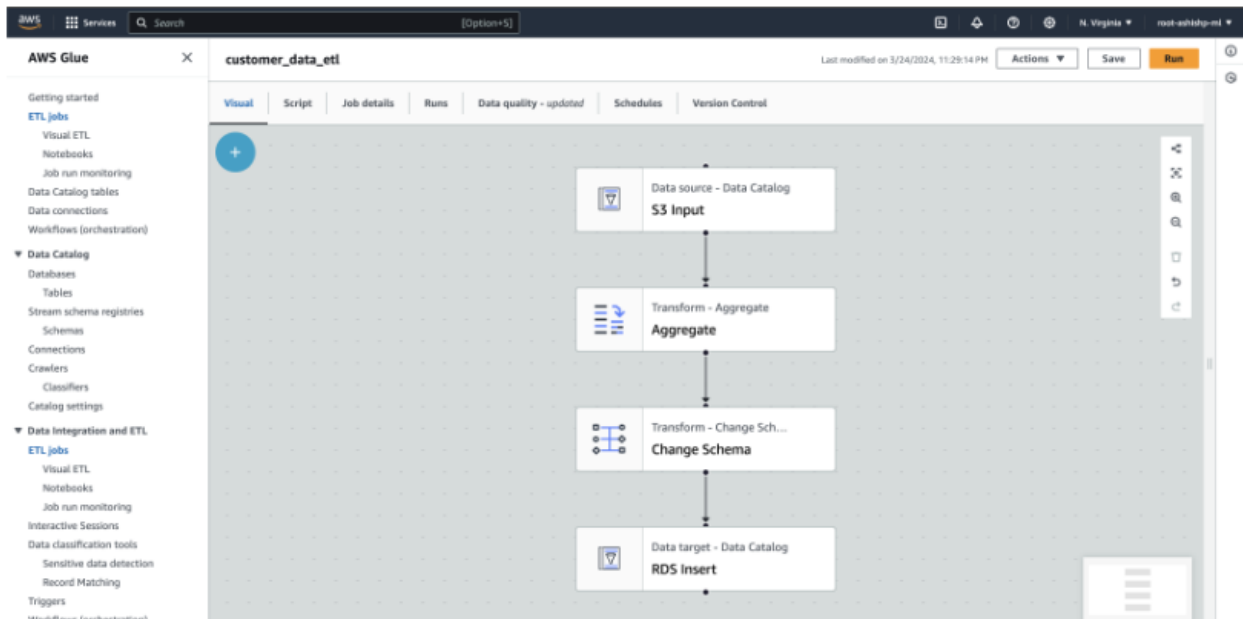
Schema	Partitions	Indexes	Column statistics - new
Schema (7) View and manage the table schema. <input type="text" value="Filter schemas"/>			
#	Column name	Data type	Partition key
1	customerid	bigint	-
2	firstname	string	-
3	lastname	string	-
4	plantype	string	-
5	monthlycharge	bigint	-
6	subscriptiondate	string	-
7	date	string	Partition (0)

Step 4: Create a Glue Connection to RDS:

1. Go to Glue → Connections → Add connection.
2. Type: JDBC → MySQL → Enter RDS endpoint, DB name, username, password.
3. Store credentials in **AWS Secrets Manager** for security.

Step 5: Create a Glue Job to aggregate data from S3 and push to RDS

AWS Glue



Incremental Load

The screenshot shows the 'Job details' tab for the workflow 'customer_data_etl'. The 'Job bookmark' option is highlighted with a red circle and set to 'Enable'. Other options include:

- Automatically scale the number of workers**: ☐ AWS Glue will optimize costs and resource usage by dynamically scaling the number of workers up and down throughout the job run. Requires Glue 3.0 or later.
- Requested number of workers**: The number of workers you want AWS Glue to allocate to this job.
- Generate job insights**: ☐ AWS Glue will analyze your job runs and provide insights on how to optimize your jobs and the reasons for job failures.
- Job bookmark** (highlighted): Specifies how AWS Glue preserves job bookmark when the job runs. It can remember previously processed data (Enabled), update state information (Paced), or ignore state information (Disabled).
- Flex execution**: ☐ Reduce costs by running this job on spare capacity. Ideal for non-urgent workloads that don't require fast jobs start times or consistent execution times. See recommendations, limitations and pricing in the help panel by clicking on the Info link above.
- Number of retries**:
- Job timeout (minutes)**: Set the execution time. The default is 2,880 minutes (48 hours) for a Glue ETL job. No job timeout is defaulted for a Glue Streaming job.

The bottom section is labeled 'Advanced properties'.

The screenshot displays the AWS Glue console interface. On the left sidebar, navigation options include Getting started, ETL jobs, Visual ETL, Notebooks, Job run monitoring, Data Catalog tables, Data connections, Workflows (orchestration), Data Catalog (with sub-items like Databases, Tables, Stream schema registries, Schemas, Connections, Crawlers, Classifiers, and Catalog settings), Data Integration and ETL, and ETL jobs.

The main panel shows the details for the job 'customer_data_etl'. At the top, it indicates the job was last modified on 3/24/2024 at 11:29:14 PM. Below this are tabs for Visual, Script, Job details (selected), Runs, Data quality - updated, Schedules, and Version Control.

The 'Job runs' section shows 1/2 runs. A table lists the following run:

Run status	Retries	Start time (UTC)	End time (UTC)	Duration	Capacity (DPU's)	Worker type	Glue version
Succeeded	0	2024/03/25 06:29:21	2024/03/25 06:30:57	1 m 39 s	2 DPU's	G.1X	4.0

A second row is partially visible, showing a 'Failed' status.

Below the job runs table, there are sections for 'Run details' (selected), Input arguments (11), Continuous logs, Run insights, Metrics, and Spark UI. The 'Run details' section provides further information about the selected run, including its name ('customer_data_etl'), start/end times, ID, worker type (G.1X), log group name ('aws-glue/jobs'), and retry attempt number (1).