



Project Title:

Real-Time and Historical Sales Data Integration Using Amazon Redshift Spectrum



Problem Scenario:

An e-commerce company's data engineering team needs to analyze both **real-time sales data stored in Amazon Redshift** and **historical sales data stored in Amazon S3** (in Parquet/CSV format). The challenge is to query both datasets **together** without:

- Duplicating the S3 data into Redshift
 - Manually importing historical files
 - Running frequent ETL pipelines
-



Objective:

To build a solution that enables seamless querying of **transactional (live)** and **historical (S3)** sales data using **Amazon Redshift Spectrum**.



Solution Overview:

We implemented a hybrid analytics system using:

- **Amazon S3** for storing historical data partitioned by year and month
 - **AWS Glue Data Catalog** for crawling and cataloging external S3 data
 - **Amazon Redshift Serverless** for real-time querying
 - **Redshift Spectrum** to federate queries across internal Redshift and external S3 data
-



Implementation Steps:

1 Created and Uploaded Historical Sales Data to S3

- Created structured CSV files with fields:
 - `product_id`, `quantity`, `revenue`
 - (Partition columns `year` and `month` were *excluded* from CSV body but added as folder structure)

- Uploaded files using path structure:

```
s3://ecommerce-sales--data--bucket/historical_sales/year=2023/month=1/
```

2 Configured AWS Glue Data Catalog

- Created Glue Database: ecommerce_sales_db
- Created a **Glue Crawler** pointing to:


```
s3://ecommerce-sales--data--bucket/historical_sales/
```
- Crawler detected partitions and created a table:


```
ecommerce_sales_data_bucket
```

 with inferred schema and partitions year, month

3 Created an IAM Role for Redshift Spectrum Access

- IAM Role Name: my-iam-redshift-role
- Attached the following policies:
 - AmazonS3ReadOnlyAccess
 - AWSGlueConsoleFullAccess (or AWSGlueCatalogReadOnlyAccess)
- Attached the IAM role to Redshift Serverless **namespace settings**

4 Created External Schema in Redshift

Connected Redshift to the Glue Data Catalog using:

```
CREATE EXTERNAL SCHEMA spectrum_schema
FROM DATA CATALOG
DATABASE 'ecommerce_sales_db'
IAM_ROLE 'arn:aws:iam::535002864770:role/my-iam-redshift-role'
CREATE EXTERNAL DATABASE IF NOT EXISTS;
```

5 Created Redshift Internal Table (Transactional Data)

```
CREATE TABLE redshift_sales (
  product_id INT,
  quantity INT,
  revenue FLOAT
);

-- Sample data
INSERT INTO redshift_sales VALUES (101, 2, 200), (102, 1, 120);
```

6 Joined Redshift and Spectrum Tables

Combined real-time and historical data using:

```

SELECT
    r.product_id,
    r.quantity AS live_qty,
    h.quantity AS historical_qty,
    r.revenue AS live_revenue,
    h.revenue AS historical_revenue
FROM
    redshift_sales r
JOIN
    spectrum_schema.ecommerce_sales_data_bucket h
ON
    r.product_id = h.product_id
WHERE
    h.year = 2023 AND h.month = 1;
  
```

This query successfully fetched data from both Redshift and S3 in a single result set.

Outcome:

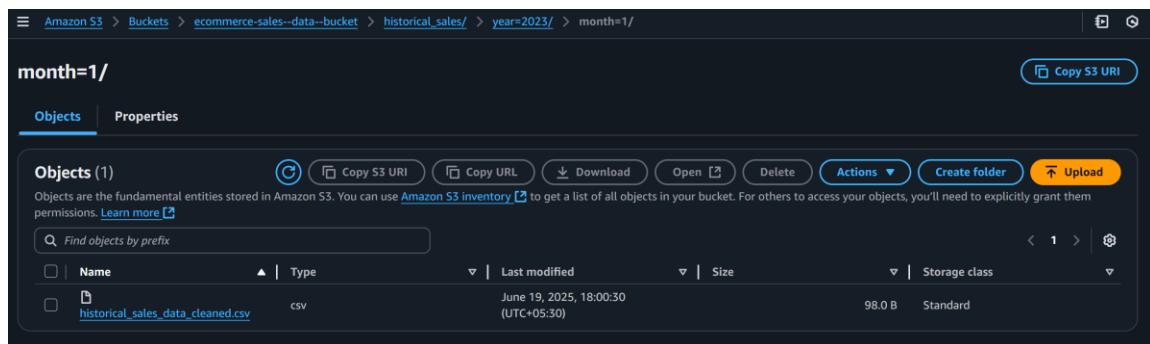
- Successfully queried **live transactional** data and **historical S3** data together
 - Verified **Redshift Spectrum** setup and IAM access
 - Achieved **zero duplication** and no data import overhead
-

Benefits:

-  No need to load S3 data into Redshift
 -  Efficient partition-based pruning from S3
 -  Fast, federated query across real-time and historical sales
 -  Unified analytics layer for dashboards and reports
-

Workspace Output (Paste Screenshots/Logs Below):

[S3 folder structure screenshot]



The screenshot shows the AWS S3 console interface. The path in the top navigation bar is: Amazon S3 > Buckets > ecommerce-sales-data-bucket > historical_sales/ > year=2023/ > month=1/. The main area displays a table of objects under the heading "Objects (1)". The table includes columns for Name, Type, Last modified, Size, and Storage class. One object is listed: "historical_sales_data_cleaned.csv" (Type: CSV, Last modified: June 19, 2025, 18:00:30 (UTC+05:30), Size: 98.0 B, Storage class: Standard). The "Actions" column contains options like Copy S3 URI, Copy URL, Download, Open, Delete, Create folder, and Upload.

[Glue crawler setup screenshot]

The screenshot shows the AWS Glue Crawler configuration page for a crawler named 'historical_sales'. The top navigation bar indicates the crawler is 'successfully starting'. The main section displays 'Crawler properties' including the name, IAM role ('my-iam-glue-role'), database ('ecommerce_sales_db'), and state ('STOPPING'). Below this is an 'Advanced settings' section. The 'Crawler runs' tab is selected, showing two completed runs on June 19, 2025, at 12:30:46 and 11:42:18, each taking 1 minute and 23 seconds. Buttons for 'Stop run', 'View CloudWatch logs', and 'View run details' are available.

[Glue catalog table structure]

The screenshot shows the AWS Glue Catalog Table structure for a table named 'ecommerce_sales_data_bucket'. The table is associated with the database 'ecommerce_sales_db' and has a CSV classification. It is located in the S3 path 's3://ecommerce-sales--data--bucket/'. The table was last updated on June 19, 2025, at 12:32:09. The 'Advanced properties' section is collapsed. The 'Schema' tab is selected, displaying the table schema with six columns: product_id, quantity, revenue, partition_0, year, and month. The schema can be edited via 'Edit schema as JSON' or 'Edit schema' buttons.

#	Column name	Data type	Partition key	Comment
1	product_id	bigint	-	-
2	quantity	bigint	-	-
3	revenue	double	-	-
4	partition_0	string	Partition (0)	-
5	year	string	Partition (1)	-
6	month	string	Partition (2)	-

[Redshift SQL editor queries and output]

The screenshot shows the Redshift SQL editor interface. On the left is a sidebar with icons for Editor, Queries, Notebooks, Charts, History, Scheduled queries, and a gear icon. The main area has a title bar "Untitled 1" with tabs for Run, Explain, Isolated session, and Serverless mode. Below the title bar is a search bar "Filter resources" and a dropdown menu showing "Serverless: my-project-wor...". The code editor contains the following SQL:

```
1 CREATE EXTERNAL SCHEMA spectrum_schema_1
2 FROM DATA CATALOG
3 DATABASE 'ecommerce_sales_db'
4 IAM_ROLE 'arn:aws:iam::535002864770:role/my-iam-redshift-role'
5 CREATE EXTERNAL DATABASE IF NOT EXISTS;
```

The results pane below the editor shows a "Summary" section with an info message: "External database 'ecommerce_sales_db' already exists". It also displays performance metrics: "Returned rows: 0", "Query ID: 4583", "Elapsed time: 45ms", and "Result set query". The result set itself is empty.

[Query result screenshot combining Redshift + Spectrum data]

This screenshot shows a similar setup to the first one, with the Redshift SQL editor interface and a sidebar on the left. The main area contains a query:

```
2 r.product_id,
3 r.quantity AS live_qty,
4 h.quantity AS historical_qty,
5 r.revenue AS live_revenue,
6 h.revenue AS historical_revenue
7 FROM
8 redshift_sales r
9 JOIN
10 spectrum_schema.ecommerce_sales_data_bucket h
11 ON
12 r.product_id = h.product_id
13 WHERE
14 h.year = 2023 AND h.month = 1;
```

The results pane shows a table titled "Result 1 (4)" with the following data:

product_id	live_qty	historical_qty	live_revenue	historical_revenue
101	2	20	200	2000
101	2	20	200	2000
102	1	35	120	3500
102	1	35	120	3500

 Prepared by:

Atmakuru Siva Sandeep

 Date: 19 June 2025

 Tags:

#AWS #RedshiftSpectrum #GlueCatalog #S3Analytics #HybridDataWarehouse
#FederatedQueries