Building Data Analytics Solutions Using Amazon Redshift:

Lab 2 - Data Analytics Using Amazon Redshift Spectrum

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at *AWS Training and Certification*.

Lab overview

Loading data into an Amazon Redshift cluster might not always be practical for your company needs. You are tasked with implementing Amazon Redshift Spectrum to run analytics against data residing in your data lake within Amazon Simple Storage Service (Amazon S3). You are also asked to provide a solution for some data analytics across data sources, and you want to use an Amazon Redshift federated query to access data from a managed relational database service, such as Amazon Aurora.

OBJECTIVES

By the end of this lab, you will be able to:

- Use Amazon Redshift Spectrum to create an external table.
- Query data stored in Amazon S3.
- Query Amazon Aurora data from the Amazon Redshift query editor with federated query access.
- Use the UNLOAD command to save query results to Amazon S3.
- Use Data API to interact with the cluster.

TECHNICAL KNOWLEDGE PREREQUISITES

- Experience with Cloud platforms.
- Basic navigation of the AWS Management Console.
- Basic knowledge of Amazon Redshift.

DURATION

This lab requires approximately *45* minutes to complete.

ICON KEY

Various icons are used throughout this lab to call attention to certain aspects of the guide. The following list explains the purpose for each one:

- **Command:** A command that you must run.
- **Expected output:** A sample output that you can use to verify the output of a command or edited file.
- **Note:** A hint, tip, or important guidance.
- **Learn more:** Where to find more information.
- **Copy edit:** A time when copying a command, script, or other text to a text editor (to edit specific variables within it) might be easier than editing directly in the command line or terminal.
- **Answer:** An answer to a question or challenge.

Start lab

1. To launch the lab, at the top of the page, choose Start lab.

Caution: You must wait for the provisioned AWS services to be ready before you can continue.

2. To open the lab, choose Open Console.

You are automatically signed in to the AWS Management Console in a new web browser tab.

WARNING: Do not change the Region unless instructed.

COMMON SIGN-IN ERRORS

Error: You must first sign out

If you see the message, **You must first log out before logging into a different AWS account:**

- Choose the **click here** link.
- Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
- Choose Open Console again.

Error: Choosing Start Lab has no effect

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

- Add the lab domain name to your pop-up or script blocker's allow list or turn it off.
- Refresh the page and try again.

SERVICES USED IN THIS LAB

Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the AWS Cloud. An Amazon Redshift data warehouse is a collection of computing resources called *nodes*, which are organized into a group called a *cluster*. Each cluster runs an Amazon Redshift engine and contains one or more databases. For more information about Amazon Redshift, refer to *Amazon Redshift Documentation* in the **Additional Resources** section at the end of this lab.

Amazon Aurora is a fully managed relational database engine that's compatible with MySQL and PostgreSQL. You likely already know how MySQL and PostgreSQL combine the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases. The code, tools, and applications you use today with your existing MySQL and PostgreSQL databases can be used with Aurora. For more information about Amazon Aurora, refer to *What is Amazon Aurora?* in the **Additional Resources** section at the end of this lab.

AWS Secrets Manager can help replace hardcoded credentials in your code, including passwords. An API call to Secrets Manager can retrieve a *secret* programmatically. Because the secret no longer exists in your code, it can't be compromised by someone examining the code. Also, you can configure Secrets Manager to automatically rotate a secret for you, according to a specified schedule. This way, you can replace long-term secrets with short-term secrets, significantly reducing the risk of compromise. For more information about AWS Secrets Manager, refer to *What is AWS Secrets Manager?* in the **Additional Resources** section at the end of this lab.

AWS SERVICES NOT USED IN THIS LAB

AWS service capabilities used in this lab are limited to what the lab requires. Expect errors when accessing other services or performing actions beyond those provided in this lab guide.

Task 1: Explore the lab environment

In this task, you review the lab environment to gain a better understanding of the resources you work with through this lab. You also load sample data into Amazon Aurora database.

LAB ARCHITECTURE

First, examine the lab architecture.

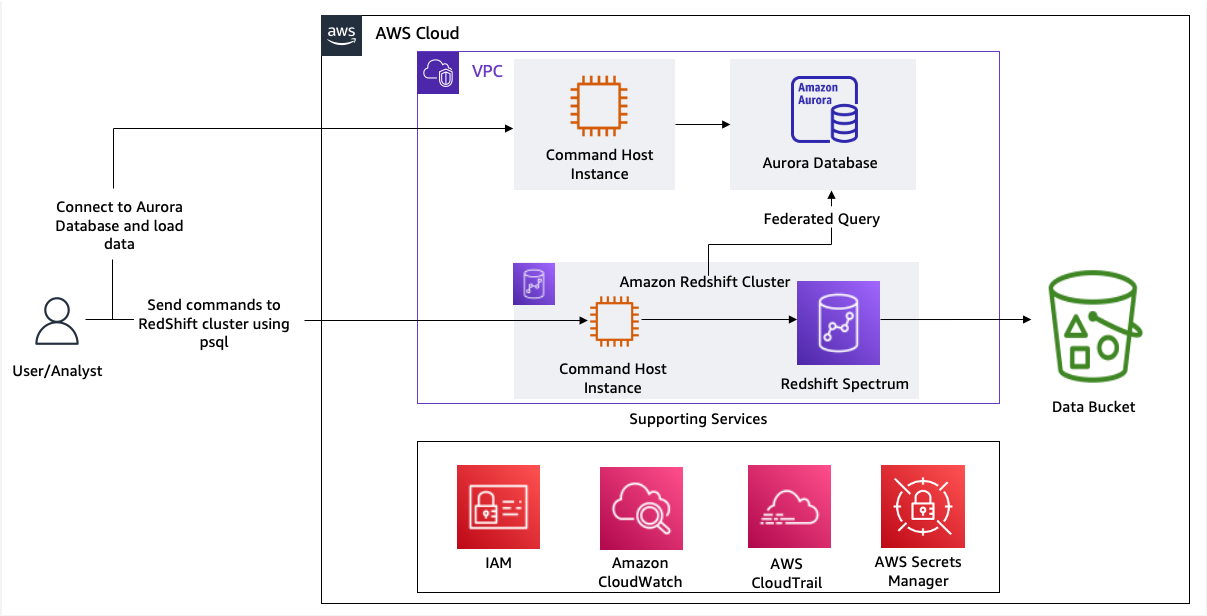

Image description: The preceding diagram depicts the connection between an external user and Amazon Redshift cluster nodes via a CommandHost EC2 instance within an Amazon Virtual Private Cloud. The external user also connects with an Aurora Database via another CommandHost EC2 instance. Lastly, the diagram depicts other supporting services like IAM, Amazon CloudWatch, AWS CloudTrail, and AWS Secrets Manager.

During the lab deployment process, the following resources are created for you:

- An Amazon Virtual Private Cloud (Amazon VPC)
- One Amazon Redshift cluster in a private subnet
- One Amazon Aurora database in a private subnet
- Two Amazon Elastic Compute Cloud (Amazon EC2) instances, used as **Command Hosts**, in a public subnet
- An S3 bucket that contains data that you load into Amazon Redshift

You use the CommandHost EC2 instance to interact with the Redshift databases. You use the second CommandHost EC2 instance to run AWS CLI commands and to interact with the Amazon Aurora database.

REVIEW THE FOLDERS IN THE AMAZON S3 BUCKET

3. If you have not already done so, follow the steps in the Start Lab section to log into the AWS Management Console.
4. At the top-right corner of the page, verify that the **AWS Region** matches the **Region** listed to the left of these instructions.

Choose the **Region** drop-down menu to display the list of AWS Regions and their associated codes. For example, **US West (Oregon)** has a code of **us-west-2**.

5. At the top of the page, in the unified search bar, search for and choose

S3

.

6. Choose the link for the bucket with **databucket** in the name.
7. Choose the **data/** link to open the folder.

There is one CSV file in the folder named **stock_prices.csv**. The file contains actual trading data from January 2, 2001 through September 14, 2021, similar to this:

Trade_Date	Ticker	High	Low	Open_value	Close	Volume	Adj_Close
2020-01-02	aapl	75.15	73.79	74.05	75.08	135480400.0	74.20
2020-01-02	sq	64.05	62.95	62.99	63.83	5264700	63.83
2020-01-02	amzn	1898.01	1864.15	1875.01	1898.01	4029000	1898.01
2020-01-02	ge	11.96	11.23	11.23	11.93	87421800.0	11.86
2020-01-02	m	17.27	16.39	17.18	16.52	26388100.0	15.86
2020-01-02	tsla	86.14	84.34	84.90	86.05	47660500.0	86.05
2020-01-02	msft	160.73	158.33	158.78	160.62	22622100.0	158.20

LOAD DATA INTO THE AURORA DATABASE

During the lab environment build process, an Amazon Aurora database with a PostgreSQL engine was created.

Next, upload the data from the S3 bucket to the Aurora database.

8. In the list of parameters to the left of these instructions, copy the **CommandHostUrl** value and paste it into a new web browser tab.

When you navigate to the given URL, an AWS Systems Manager Session Manager console connection to the instance opens. A set of commands are run automatically when you connect to the instance that change to the user's home directory, display the path of the working directory, and set a number of environment variables, similar to this:

```
******************************
**** This is OUTPUT ONLY. ****
******************************

cd $HOME; pwd; export Region=us-west-2; source ./script.sh
sh-4.2$ cd $HOME; pwd; export Region=us-west-2; source ./script.sh
/home/ec2-user
sh-4.2$
```

The environment variables that are set are intended to reduce the number of manual edits to the commands run on the instance throughout this lab. An AWS Systems Manager document is used to specify the AWS Systems Manager Session Manager connection preferences. For more information about configuring AWS Systems Manager Session Manager connection preferences, refer to *Configure SSM Session Preferences* in the **Additional Resources** section at the end of this lab.

9. In the terminal window, enter the following commands to connect to the Aurora cluster node, create a new schema and database, and load the data from the S3 bucket:

On a Windows-based computer, you might need to use **Ctrl + Shift + V** or open the context (right-click) menu to paste text into a Session Manager console window.

```
# Get Aurora database cluster endpoint and export to the Environment.
export HOST=$(aws rds describe-db-clusters | jq '.DBClusters[0].Endpoint' | tr
-d '"')

# Get Aurora username and export to the Environment
```

```
export username=$(aws rds describe-db-clusters | jq
'.DBClusters[0].MasterUsername' | tr -d '"')

# Get Database name and export to the environment
export dbname=$(aws rds describe-db-clusters | jq
'.DBClusters[0].DatabaseName' | tr -d '"')

# Connect to the Aurora cluster using psql, the db.sql script creates a
schema, creates a stocks table, and loads the data.
psql -h $HOST -U $username -d $dbname -password $PGPassword -p 5432 -f
/home/ec2-user/db.sql

# Copy data to stocks table
psql -h $HOST -U $username -d $dbname -password $PGPassword -p 5432 -c '\COPY
stocks FROM ''/home/ec2-user/stocks.csv'' CSV HEADER'

# Connect to database to validate the data with a simple SQL query
psql -h $HOST -U $username -d $dbname -password $PGPassword -p 5432

-- Validate the data by querying all stack values on January 3, 2019
SELECT * FROM stocks WHERE Trade_Date LIKE '2019-01-03' ORDER BY Ticker;

# Quit the psql shell
\q
```

The script you just ran performs the following tasks:

- Creates a *stocksummary* schema in the Aurora database
- Creates a *stocks* table
- Copies the stocks CSV data to the *stocks* table
- Connects to a database psql console
- Queries the data from the *stocks* table
- Exits the psql shell

Expected output:

```
*****************************
**** This is OUTPUT ONLY. ****
*****************************

sh-4.2$ # Get Aurora database cluster endpoint and export to the Environment.
sh-4.2$ export HOST=$(aws rds describe-db-clusters | jq
'.DBClusters[0].Endpoint' | tr -d '"')
sh-4.2$
sh-4.2$ # Get Aurora username and export to the Environment
sh-4.2$ export username=$(aws rds describe-db-clusters | jq
'.DBClusters[0].MasterUsername' | tr -d '"')
sh-4.2$
sh-4.2$ # Get Database name and export to the environment
sh-4.2$ export dbname=$(aws rds describe-db-clusters | jq
'.DBClusters[0].DatabaseName' | tr -d '"')
sh-4.2$
sh-4.2$ # Connect to the Aurora cluster using psql, the db.sql script creates
a schema, creates a stocks table, and loads the data.
sh-4.2$ psql -h $HOST -U $username -d $dbname -password $PGPassword -p 5432 -f
/home/ec2-user/db.sql
```

```
CREATE SCHEMA
CREATE TABLE
sh-4.2$
sh-4.2$ # Copy data to stocks table
sh-4.2$ psql -h $HOST -U $username -d $dbname -password $PGPassword -p 5432 -c
'\COPY stocks FROM ''/home/ec2-user/stocks.csv'' CSV HEADER'
sh-4.2$
sh-4.2$ # Connect to database to validate the data with a simple SQL query
sh-4.2$ psql -h $HOST -U $username -d $dbname -password $PGPassword -p 5432
psql (9.2.24, server 12.8)
WARNING: psql version 9.2, server version 12.0.
         Some psql features might not work.
SSL connection (cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256)
Type "help" for help.

stocksummary=>
stocksummary=> -- Validate the data by querying all stack values on January 3,
2019
stocksummary=> SELECT * FROM stocks WHERE Trade_Date LIKE '2019-01-03' ORDER
BY Ticker;
 trade_date | ticker |  high   |   low   |  open   |  close  |  volume   |
adj_close
------------+--------+---------+---------+---------+---------+-----------+----
-------
 2019-01-03 | aal    |   31.85 |   28.81 |   31.69 |   30.06 |  16822000 |
29.58
 2019-01-03 | aapl   |   36.43 |   35.50 |   35.99 |   35.55 | 365248800 |
34.56
 2019-01-03 | amzn   | 1538.00 | 1497.11 | 1520.01 | 1500.28 |   6975600 |
1500.28
 2019-01-03 | ba     |  319.74 |  309.40 |  319.49 |  310.90 |   5705600 |
302.10
 2019-01-03 | bac    |   25.04 |   24.45 |   24.94 |   24.56 |  66599600 |
23.06
 2019-01-03 | c      |   53.62 |   52.22 |   53.41 |   52.56 |  21183000 |
48.01
 2019-01-03 | coke   |  203.59 |  177.45 |  180.10 |  177.89 |     27200 |
176.08
 2019-01-03 | dis    |  108.65 |  105.94 |  108.48 |  106.33 |  10594700 |
105.05
 2019-01-03 | f      |    7.99 |    7.78 |    7.97 |    7.78 |  39172400 |
7.17
 2019-01-03 | ge     |   63.08 |   59.85 |   61.69 |   62.00 |  15983370 |
61.35
 2019-01-03 | gs     |  171.77 |  168.29 |  170.66 |  169.51 |   4060200 |
160.22
 2019-01-03 | hsy    |  107.35 |  104.91 |  105.23 |  106.24 |   1236700 |
100.18
 2019-01-03 | intc   |   46.28 |   44.39 |   46.15 |   44.49 |  32267300 |
41.57
 2019-01-03 | kodk   |    2.55 |    2.48 |    2.53 |    2.50 |    123300 |
2.50
 2019-01-03 | m      |   30.78 |   29.71 |   30.42 |   29.76 |   7897400 |
26.39
 2019-01-03 | ma     |  187.51 |  180.98 |  187.50 |  181.18 |   5070900 |
178.56
```

```
 2019-01-03 | msft   |  100.19 |   97.20 |  100.10 |   97.40 |  42579100 |
94.36
 2019-01-03 | nke    |   73.32 |   71.21 |   73.25 |   72.75 |   8007400 |
70.90
 2019-01-03 | pg     |   92.50 |   90.38 |   90.94 |   90.64 |   9820200 |
84.45
 2019-01-03 | pypl   |   84.75 |   81.91 |   84.36 |   82.09 |   9650700 |
82.09
 2019-01-03 | sq     |   56.73 |   52.26 |   55.58 |   52.42 |  19076300 |
52.42
 2019-01-03 | tsla   |   61.88 |   59.48 |   61.40 |   60.07 |  34826000 |
60.07
 2019-01-03 | v      |  131.28 |  127.88 |  131.21 |  128.13 |   9428300 |
125.98
 2019-01-03 | wmt    |   94.71 |   92.70 |   93.21 |   92.86 |   8277300 |
88.48
(24 rows)

stocksummary=>
stocksummary=> # Quit the psql shell
stocksummary-> \q
sh-4.2$
```

10. Close your browser tab with the EC2 instance terminal connection and return to your browser tab with the **AWS Management Console**.

Now that you have reviewed the lab environment and loaded the sample data into Amazon Aurora, you're ready to begin!

Task 2: Use Redshift Spectrum to analyze data in the Amazon S3 bucket

In this task, you use the **psql** interface via **CommandHostUrlRedShift** URL (value found in the left pane of these instructions) to create an external table in Amazon Redshift that references data stored in an Amazon S3 bucket. You then run queries against the external table, which uses Redshift Spectrum to process the data directly from the S3 bucket.

Unlike a normal Amazon Redshift table, an external table references data stored in Amazon S3 (data lake).

To create an external table, start by defining an external schema. The external schema references a database in the external data catalog and provides the AWS Identity and Access Management (IAM) role identifier (Amazon Resource Name [ARN]) that authorizes your cluster to access Amazon S3 on your behalf.

DIRECTIONS FOR CONNECTING TO THE COMMAND HOST TO USE PSQL

11. **Copy edit:** Copy the **CommandHostUrlRedShift** value found in the left pane of these instructions into a new browser tab to access the command host terminal.
12. **Command:** Run the following commands on the command host:

Note: Replace the string **<INSERT_REDSHIFT_CLUSTER_ENDPOINT>** with the value of **RedshiftEndpoint** provided to the left of these instructions.

```
cd ~
psql -U dbadmin -h '<INSERT_REDSHIFT_CLUSTER_ENDPOINT>' -d lab -p 5439
```

Expected output: Your values differ from what is seen below.

```
******************************
**** This is OUTPUT ONLY. ****
******************************

home/ec2-user
sh-4.2$ cd ~
sh-4.2$ psql -U dbadmin -h redshiftcluster.chjn8kpmblv3.us-east-
1.redshift.amazonaws.com -d lab -p 5439
psql (13.7, server 8.0.2)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits:
256, compression: off)
Type "help" for help.

lab=#
```

This should log you into the database and give you a prompt where you can enter **SQL** commands used in this lab.

CREATE AN EXTERNAL SCHEMA AND TABLE

13. Using the psql prompt, enter the following query to create an external schema named **spectrum**:

- Replace the **INSERT_REDSHIFT_ROLE** placeholder value with the **RedshiftRole** value listed to the left of these instructions. (Be sure to keep the single quote marks.)

```
CREATE EXTERNAL SCHEMA spectrum
FROM DATA CATALOG
DATABASE 'spectrumdb'
IAM_ROLE 'INSERT_REDSHIFT_ROLE'
CREATE EXTERNAL DATABASE IF NOT EXISTS;
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************

INFO:  External database "spectrumdb" created
CREATE SCHEMA
lab=#
```

Next, create an external table that is stored in the **spectrum schema**.

14. Using the psql prompt, enter the following query to create a new external table named **stocksummary** within the **spectrum** schema:

- Replace the **INSERT_DATA_BUCKET_NAME** placeholder value with the **DataBucket** value listed to the left of these instructions.

```
DROP TABLE IF EXISTS spectrum.stocksummary;
CREATE EXTERNAL TABLE spectrum.stocksummary(
    Trade_Date VARCHAR(15),
    Ticker VARCHAR(5),
    High DECIMAL(8,2),
    Low DECIMAL(8,2),
    Open_value DECIMAL(8,2),
    Close DECIMAL(8,2),
    Volume DECIMAL(15),
    Adj_Close DECIMAL(8,2)
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION 's3://INSERT_DATA_BUCKET_NAME/data/';
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************

INFO:  External table "stocksummary" does not exist and will be skipped
CREATE EXTERNAL TABLE
lab=#
```

VALIDATE THE DATA FROM QUERY EDITOR

In this step, Redshift Spectrum directly queries the data from the Amazon S3 bucket. There is no data movement from storage to an Amazon Redshift cluster. This way, you can perform data analytics at the petabyte scale.

15. Using the psql prompt, enter the following query to retrieve data from the **stocksummary** table in the **spectrum** schema:

```
SELECT * FROM spectrum.stocksummary WHERE Trade_Date LIKE '2021-09-09' ORDER
BY Ticker;
```

Expected output:

The query results should display the details for stocks that were traded on September 9, 2021, similar to this:

```
******************************
**** This is OUTPUT ONLY. ****
******************************

 trade_date | ticker |  high   |   low   | open_value |  close  |  volume  |
adj_close
------------+--------+---------+---------+------------+---------+----------+--
---------
```

```
 2021-09-09 | aal    |   20.61 |   19.01 |      19.10 |   20.20 | 60077200 |
20.20
 2021-09-09 | aapl   |  156.11 |  153.95 |     155.49 |  154.07 | 57305700 |
154.07
 2021-09-09 | amzn   | 3549.99 | 3480.37 |    3526.02 | 3484.16 |  2719200 |
3484.16
 2021-09-09 | ba     |  216.61 |  210.72 |     211.15 |  213.94 |  9242800 |
213.94
 2021-09-09 | bac    |   41.35 |   40.56 |      40.66 |   40.93 | 36266400 |
40.93
 2021-09-09 | c      |   71.10 |   69.91 |      69.97 |   70.46 | 14212900 |
70.46
 2021-09-09 | chwy   |   76.89 |   75.54 |      75.99 |   76.60 |  2386900 |
76.60
 2021-09-09 | coke   |  398.45 |  393.25 |     397.75 |  394.14 |    16200 |
394.14
 2021-09-09 | dis    |  187.58 |  184.57 |     185.15 |  185.91 |  7190700 |
185.91
 2021-09-09 | f      |   12.95 |   12.72 |      12.95 |   12.76 | 68806400 |
12.76
 2021-09-09 | ge     |  104.92 |  102.11 |     102.36 |  103.29 |  9386500 |
103.29
 2021-09-09 | gs     |  408.88 |  403.51 |     404.67 |  404.58 |  2424600 |
404.58
 2021-09-09 | hsy    |  176.68 |  175.00 |     176.58 |  175.40 |   623500 |
175.40
 2021-09-09 | intc   |   53.89 |   53.34 |      53.66 |   53.40 | 13495800 |
53.40
 2021-09-09 | kodk   |    7.38 |    7.01 |       7.01 |    7.24 |  1765000 |
7.24
 2021-09-09 | m      |   21.88 |   21.01 |      21.33 |   21.64 | 20695400 |
21.64
 2021-09-09 | ma     |  353.64 |  349.92 |     350.71 |  351.41 |  3716200 |
351.41
 2021-09-09 | msft   |  302.14 |  297.00 |     300.82 |  297.25 | 19927000 |
297.25
 2021-09-09 | nke    |  166.15 |  163.20 |     165.26 |  163.34 |  8414800 |
163.34
 2021-09-09 | pg     |  144.88 |  143.65 |     144.57 |  143.99 |  6354700 |
143.99
 2021-09-09 | pypl   |  289.37 |  285.08 |     287.45 |  286.88 |  4170000 |
286.88
 2021-09-09 | sq     |  258.63 |  251.38 |     256.00 |  251.54 |  5606200 |
251.54
 2021-09-09 | tsla   |  762.10 |  751.63 |     753.41 |  754.86 | 14077700 |
754.86
 2021-09-09 | v      |  230.00 |  227.10 |     229.31 |  227.49 |  5853000 |
227.49
 2021-09-09 | wmt    |  147.61 |  145.84 |     147.46 |  146.42 |  7442900 |
146.42
(25 rows)

lab=#
```

Congratulations! You have successfully created a Redshift schema and external table to query data stored in Amazon S3.

Task 3: Analyze data stored in Aurora using federated queries

To reduce data movement over the network, Amazon Redshift distributes part of the computation for federated queries directly into remote operational databases. Amazon Redshift also uses its parallel processing capacity to help run these queries, as needed.

When running federated queries, Amazon Redshift first makes a client connection to an Aurora DB instance from a leader node to retrieve table metadata. From a compute node, Amazon Redshift issues subqueries with a predicate pushed down, and retrieves the result rows. Amazon Redshift then distributes the result rows among compute nodes for further processing.

In this task, you analyze data stored in Aurora using federated queries.

CREATE AN EXTERNAL SCHEMA FOR FEDERATED QUERY OPERATIONS

16. Using the psql prompt, enter the following query to create an external schema named **federated** for the Aurora database table named **stocksummary**:

- Replace the **INSERT_AURORA_ENDPOINT_URI** placeholder value with the **AuroraEndPointURI** value listed to the left of these instructions. (Be sure to keep the single quote marks.)
- Replace the **INSERT_REDSHIFT_ROLE** placeholder value with the **RedshiftRole** value listed to the left of these instructions. (Be sure to keep the single quote marks.)
- Replace the **INSERT_SECRET_ARN** placeholder value with the **AuroraSecretARN** value listed to the left of these instructions. (Be sure to keep the single quote marks.)

```
CREATE EXTERNAL SCHEMA federated
FROM POSTGRES
DATABASE 'stocksummary'
URI 'INSERT_AURORA_ENDPOINT_URI'
IAM_ROLE 'INSERT_REDSHIFT_ROLE'
SECRET_ARN 'INSERT_AURORA_SECRET_ARN';
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************

CREATE SCHEMA
lab=#
```

VALIDATE DATA ACCESS FROM THE AURORA DATABASE

17. Using the psql prompt, enter the following query to retrieve data directly from the Aurora database table:

```
SELECT * FROM federated.stocks WHERE Trade_Date LIKE '2021-09-09' ORDER BY
Ticker;
```

Expected output:

The query results should display the details for stocks that were traded on September 9, 2021, similar to this:

```
******************************
**** This is OUTPUT ONLY. ****
******************************

 trade_date | ticker |  high   |   low   |  open   |  close  |  volume  |
adj_close
------------+--------+---------+---------+---------+---------+----------+-----
------
 2021-09-09 | aal    |   20.61 |   19.01 |   19.10 |   20.20 | 60077200 |
20.20
 2021-09-09 | aapl   |  156.11 |  153.95 |  155.49 |  154.07 | 57305700 |
154.07
 2021-09-09 | amzn   | 3549.99 | 3480.37 | 3526.02 | 3484.16 |  2719200 |
3484.16
 2021-09-09 | ba     |  216.61 |  210.72 |  211.15 |  213.94 |  9242800 |
213.94
 2021-09-09 | bac    |   41.35 |   40.56 |   40.66 |   40.93 | 36266400 |
40.93
 2021-09-09 | c      |   71.10 |   69.91 |   69.97 |   70.46 | 14212900 |
70.46
 2021-09-09 | chwy   |   76.89 |   75.54 |   75.99 |   76.60 |  2386900 |
76.60
 2021-09-09 | coke   |  398.45 |  393.25 |  397.75 |  394.14 |    16200 |
394.14
 2021-09-09 | dis    |  187.58 |  184.57 |  185.15 |  185.91 |  7190700 |
185.91
 2021-09-09 | f      |   12.95 |   12.72 |   12.95 |   12.76 | 68806400 |
12.76
 2021-09-09 | ge     |  104.92 |  102.11 |  102.36 |  103.29 |  9386500 |
103.29
 2021-09-09 | gs     |  408.88 |  403.51 |  404.67 |  404.58 |  2424600 |
404.58
 2021-09-09 | hsy    |  176.68 |  175.00 |  176.58 |  175.40 |   623500 |
175.40
 2021-09-09 | intc   |   53.89 |   53.34 |   53.66 |   53.40 | 13495800 |
53.40
 2021-09-09 | kodk   |    7.38 |    7.01 |    7.01 |    7.24 |  1765000 |
7.24
 2021-09-09 | m      |   21.88 |   21.01 |   21.33 |   21.64 | 20695400 |
21.64
 2021-09-09 | ma     |  353.64 |  349.92 |  350.71 |  351.41 |  3716200 |
351.41
 2021-09-09 | msft   |  302.14 |  297.00 |  300.82 |  297.25 | 19927000 |
297.25
 2021-09-09 | nke    |  166.15 |  163.20 |  165.26 |  163.34 |  8414800 |
163.34
 2021-09-09 | pg     |  144.88 |  143.65 |  144.57 |  143.99 |  6354700 |
143.99
 2021-09-09 | pypl   |  289.37 |  285.08 |  287.45 |  286.88 |  4170000 |
286.88
```

```
 2021-09-09 | sq     |  258.63 |  251.38 |  256.00 |  251.54 |  5606200 |
251.54
 2021-09-09 | tsla   |  762.10 |  751.63 |  753.41 |  754.86 | 14077700 |
754.86
 2021-09-09 | v      |  230.00 |  227.10 |  229.31 |  227.49 |  5853000 |
227.49
 2021-09-09 | wmt    |  147.61 |  145.84 |  147.46 |  146.42 |  7442900 |
146.42
(25 rows)

lab=#
```

Congratulations! You have successfully used federated queries to retrieve data from Amazon Aurora in Amazon Redshift.

Task 4: Unload analyzed data to Amazon S3

Apache Parquet is the default file format to unload your data from Amazon Redshift through a data lake export. Parquet is an efficient, open-columnar storage format for analytics. The Parquet format is up to twice as fast to unload and consumes up to six times less storage in Amazon S3, compared to text formats.

In this task, you use the *UNLOAD* command with a *SELECT* statement to move data from database tables to a set of files in an Amazon S3 bucket.

Here is some background information about UNLOAD:

- UNLOAD automatically encrypts data files using Amazon S3 server-side encryption (SSE-S3).
- Besides the default file format, Apache Parquet, you can also use other popular file formats and compression algorithms.

For more information about UNLOAD syntax, refer to *UNLOAD* in the **Additional Resources** section at the end of this lab.

UNLOAD THE QUERY RESULTS TO AMAZON S3

Suppose you've received a special request for a client who wants to analyze only large cap stocks. In this scenario, you can UNLOAD large cap stock data and analyze it on a new dataset, which improves performance and requires less data to scan.

First, you *UNLOAD* aapl and amzn stock data to a new S3 bucket location.

18. Using the psql prompt, enter the following query to unload the selected data to the specified Amazon S3 bucket:

- Replace the **INSERT_DATA_BUCKET_NAME** placeholder value with the **DataBucket** value listed to the left of these instructions. (Be sure to keep the single quote marks.)
- Replace the **INSERT_REDSHIFT_ROLE** placeholder value with the **RedshiftRole** value listed to the left of these instructions. (Be sure to keep the single quote marks.)

```
UNLOAD ('SELECT * FROM federated.stocks ORDER BY Trade_Date')
```

```
TO 's3://INSERT_DATA_BUCKET_NAME/unload/stocks_'
IAM_ROLE 'INSERT_REDSHIFT_ROLE'
PARQUET
maxfilesize 5 mb
ALLOWOVERWRITE;
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************

INFO:  UNLOAD completed, 108230 record(s) unloaded successfully.
UNLOAD
lab=#
```

Keep the **CommandHostUrlRedShift** Session Manager session running. You come back to this window at the end of Task 6.

VERIFY DATA IN THE AMAZON S3 DATA BUCKET

Next, verify that the data was exported to the S3 bucket successfully.

19. Return to the browser tab open to the AWS Management Console.
20. At the top of the page, in the unified search bar, search for and choose

    S3

    .
21. Choose the link for the bucket with **databucket** in the name.
22. Choose the **unload/** link to open the folder.

Notice that the bucket now includes one or more files with the prefix *stocks_*, which are in Parquet format.

Congratulations! You have successfully unloaded data from Amazon Redshift to an S3 bucket.

Task 5: Challenge - Access unloaded data from the Amazon S3 bucket

Challenge yourself! Your task is to access the data you uploaded to the S3 bucket in the previous task and complete a basic query using the **psql** interface via **CommandHostUrlRedShift** Session Manager session.

The *Create external table* syntax follows. You are accessing the files in Parquet format.

```
CREATE EXTERNAL TABLE
external_schema.table_name
[ PARTITIONED BY (col_name [, … ] ) ]
[ ROW FORMAT DELIMITED row_format ]
STORED AS file_format
LOCATION { 's3://bucket/folder/' }
```

```
[ TABLE PROPERTIES ( 'property_name'='property_value' [, ...] ) ]
 AS
 { select_statement }
```

If you get stuck, refer to *Task 5 challenge solution* in the **Appendix** section.

Task 6: Use the Amazon Redshift Data API to interact with Redshift clusters

In certain scenarios, you might want to use a simple API endpoint to interact with Amazon Redshift. With Amazon Redshift Data API, you don't need to configure Java Database Connectivity (JDBC) or Open Database Connectivity (ODBC) to load or query data.

In this task, you use the AWS CLI and Amazon Redshift Data API to make API calls to Amazon Redshift.

For more information, refer to *Using the Amazon Redshift Data API* in the **Additional Resources** section at the end of this lab.

First, connect to the **CommandHost** EC2 instance.

23. In the list of parameters to the left of these instructions, copy the **CommandHostUrl** value and paste it into a new web browser tab.

When you navigate to the given URL, an AWS Systems Manager Session Manager console connection to the instance opens. A set of commands are run automatically when you connect to the instance that change to the user's home directory, display the path of the working directory, and set a number of environment variables, similar to this:

```
****************************
**** This is OUTPUT ONLY. ****
****************************

cd $HOME; pwd; export Region=us-west-2; source ./script.sh
sh-4.2$ cd $HOME; pwd; export Region=us-west-2; source ./script.sh
/home/ec2-user
sh-4.2$
```

24. Enter the following command to create a new schema named **data_api_demo** in the **lab** database:

```
aws redshift-data execute-statement \
     --database lab  \
     --cluster-identifier redshiftcluster \
     --secret-arn $RedshiftSecretArn \
     --sql "CREATE SCHEMA data_api_demo;" \
     --region $Region
```

Expected output:

The output should display metadata from the command run, such as the **Database** and run **Id**, similar to this:

```
******************************
**** This is OUTPUT ONLY. ****
******************************

{
    "ClusterIdentifier": "redshiftcluster",
    "CreatedAt": "2023-02-23T22:27:43.443000+00:00",
    "Database": "lab",
    "Id": "75507852-54b1-4669-8421-912387a0d491",
    "SecretArn": "arn:aws:secretsmanager:us-east-
1:396162753347:secret:RedshiftSecrets-Ww0vkx"
}
```

25. Enter the following command to retrieve the status of the command you just ran:

- Replace the **<INSERT_RUN_ID>** placeholder value with the value of **Id** from the output of the previous command.

```
aws redshift-data describe-statement --id '<INSERT_RUN_ID>'
```

Expected output:

The output should display information about the command run, such as its **Status** and **Id**, similar to this:

```
******************************
**** This is OUTPUT ONLY. ****
******************************

{
    "ClusterIdentifier": "redshiftcluster",
    "CreatedAt": "2023-02-23T22:27:43.443000+00:00",
    "Duration": 47264577,
    "HasResultSet": false,
    "Id": "75507852-54b1-4669-8421-912387a0d491",
    "QueryString": "CREATE SCHEMA data_api_demo;",
    "RedshiftPid": 1073946917,
    "RedshiftQueryId": -1,
    "ResultRows": 0,
    "ResultSize": 0,
    "SecretArn": "arn:aws:secretsmanager:us-east-
1:396162753347:secret:RedshiftSecrets-Ww0vkx",
    "Status": "FINISHED",
    "UpdatedAt": "2023-02-23T22:27:44.203000+00:00"
}
```

The *aws redshift-data describe-statement* command is used to verify the status of the queries you run in the CLI. While you are not asked to run the command after every step in this task, feel free to do so if you would like more insight into the outputs, or if you are troublehsooting an issue.

26. Verify the **Status** is **FINISHED**.

If the **Status** is **FAILED**, check the error message included in the output to determine where there might be an issue and try to run the previous command again.

27. Enter the following command to create a new table named **stocks_da** in the **data_api_demo** schema:

```
aws redshift-data execute-statement \
    --database lab \
    --cluster-identifier redshiftcluster \
    --secret-arn $RedshiftSecretArn  \
    --region $Region \
    --sql "CREATE TABLE data_api_demo.stocks_da (
        Trade_Date VARCHAR(15),
        Ticker VARCHAR(5),
        High DECIMAL(8,2),
        Low DECIMAL(8,2),
        Open_value DECIMAL(8,2),
        Close DECIMAL(8,2),
        Volume DECIMAL(15),
        Adj_Close DECIMAL(8,2)
        )"
```

Expected output:

The output should display metadata from the query run, such as the **Database** and run **Id**.

```
*****************************
**** This is OUTPUT ONLY. ****
*****************************

{
    "ClusterIdentifier": "redshiftcluster",
    "CreatedAt": "2023-02-23T22:33:03.594000+00:00",
    "Database": "lab",
    "Id": "bea7c77a-68ca-458c-9e57-7b1a83aa297f",
    "SecretArn": "arn:aws:secretsmanager:us-east-
1:396162753347:secret:RedshiftSecrets-Ww0vkx"
}
```

28. Enter the following command to import data from the S3 bucket to the **stocks_da** table

```
aws redshift-data execute-statement \
    --database lab \
    --cluster-identifier redshiftcluster \
    --secret-arn $RedshiftSecretArn \
    --region $Region \
    --sql "COPY data_api_demo.stocks_da
          FROM 's3://$DataBucket/data/stock_prices.csv'
          IAM_ROLE '$RedshiftRoleArn'
          DATEFORMAT 'auto'
          IGNOREHEADER 1
          DELIMITER ','
          IGNOREBLANKLINES;"
```

Expected output:

The output should display metadata from the query run, such as the **Database** and run **Id**.

```
*****************************
**** This is OUTPUT ONLY. ****
*****************************

{
    "ClusterIdentifier": "redshiftcluster",
    "CreatedAt": "2023-02-23T22:34:45.056000+00:00",
    "Database": "lab",
    "Id": "ed8ffd86-c937-45e0-8999-19955a15a5e1",
    "SecretArn": "arn:aws:secretsmanager:us-east-
1:396162753347:secret:RedshiftSecrets-Ww0vkx"
}
```

29. Enter the following command to query the **stocks_da** table to retrieve the total number of days where the closing price is greater than the opening price for each stock since January 1, 2020:

```
aws redshift-data execute-statement \
    --database lab \
    --cluster-identifier redshiftcluster \
    --secret-arn $RedshiftSecretArn \
    --region $Region \
    --sql "select ticker, count(*)
          from data_api_demo.stocks_da
          where close > open_value
          and trade_date > '2020-01-01'
          group by ticker
          order by count(*) desc;"
```

Expected output:

The output should display metadata from the query run, such as the **Database** and run **Id**.

```
*****************************
**** This is OUTPUT ONLY. ****
*****************************

{
    "ClusterIdentifier": "redshiftcluster",
    "CreatedAt": "2023-02-23T22:36:08.906000+00:00",
    "Database": "lab",
    "Id": "5f192465-0396-4fc2-b49c-d83e37b28460",
    "SecretArn": "arn:aws:secretsmanager:us-east-
1:396162753347:secret:RedshiftSecrets-Ww0vkx"
}
```

30. Enter the following command to retrieve the results of the data query:

- Replace the **<INSERT_RUN_ID>** placeholder value with the value of **Id** from the output of the previous command.

```
aws redshift-data get-statement-result --id '<INSERT_RUN_ID>'
```

Expected output:

The output should display the ticker and number of days that the close price was greater than the open price, in JSON format, similar to this:

- The **stringValue** value is the **ticker** value from the dataset.
- The **longValue** value is the count of the number of days that the close price was greater than the open price.

```
*****************************
**** This is OUTPUT ONLY. ****
*****************************
```


```
{
    "Records": [
        [
            {
                "stringValue": "msft"
            },
            {
                "longValue": 232
            }
        ],
        [
            {
                "stringValue": "pg"
            },
            {
                "longValue": 230
            }
        ],
        [
            {
                "stringValue": "aapl"
            },
            {
                "longValue": 225
            }
        ],
...
```


Depending on the size of your window, you might find that viewing the results in table or text form at is more useful than JSON. Try adding `--output table` or `--output text` to the end of the previous command.

You can run the same query to verify your results.

31. Return to your browser tab open to the **CommandHostUrlRedShift** Session Manager console.
32. Enter the following query of the **stocks_da** table to retrieve the total number of days where the closing price is greater than the opening price for each stock since January 1, 2020:

```
select ticker, count(*) from data_api_demo.stocks_da
    where close > open_value
    and trade_date > '2020-01-01'
```

```
    group by ticker
    order by count(*) desc;
```

Expected output:

The output should display the ticker and number of days that the close price was greater than the open price, similar to this:

```
******************************
**** This is OUTPUT ONLY. ****
******************************

 ticker | count
--------+-------
 msft   |   232
 pg     |   230
 aapl   |   225
 bac    |   225
 pypl   |   222
 gs     |   222
 ma     |   218
 nke    |   215
 chwy   |   215
 v      |   214
 amzn   |   212
 tsla   |   209
 hsy    |   209
 c      |   208
 intc   |   208
 sq     |   206
 wmt    |   203
 dis    |   201
 coke   |   197
 ge     |   189
 m      |   185
 aal    |   184
 ba     |   180
 f      |   175
 kodk   |   169
(25 rows)

lab=#
```

Congratulations! You have successfully used Amazon Redshift Data API to interact with the Amazon Redshift database.

Conclusion

Congratulations! You have successfully:

- Used Amazon Redshift Spectrum to create an external table.
- Queried data stored in Amazon S3.

- Queried Amazon Aurora data from the Amazon Redshift query editor with federated query access.
- Used the UNLOAD command to save query results to Amazon S3.
- Used Data API to interact with the cluster.

End lab

Follow these steps to close the console and end your lab.

33. Return to the **AWS Management Console**.
34. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
35. Choose End lab and then confirm that you want to end your lab.

Additional Resources

- Amazon Redshift Documentation
- What is Amazon Aurora?
- What is AWS Secrets Manager?
- Configure session preferences
- UNLOAD
- Using the Amazon Redshift Data API

Appendix

TASK 5 CHALLENGE SOLUTION

Code to create the table:

- Replace the **INSERT_DATA_BUCKET_NAME** placeholder value with the **DataBucket** value listed to the left of these instructions.

```
CREATE EXTERNAL TABLE spectrum.unloadstocks(
    Trade_Date VARCHAR(15),
    Ticker VARCHAR(5),
    High DECIMAL(8,2),
    Low DECIMAL(8,2),
    Open_value DECIMAL(8,2),
    Close DECIMAL(8,2),
    Volume DECIMAL(15),
    Adj_Close DECIMAL(8,2)
)
STORED AS PARQUET
LOCATION 's3://INSERT_DATA_BUCKET_NAME/unload/' ;
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************

CREATE EXTERNAL TABLE
lab=#
```

Code to perform a basic query of the data:

```
SELECT * FROM spectrum.unloadstocks LIMIT 10;
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************

 trade_date | ticker |  high  |  low   | open_value | close  |  volume   |
adj_close
------------+--------+--------+--------+------------+--------+-----------+----
-------
 2001-01-02 | aapl   |   0.27 |   0.26 |            |   0.27 | 452312000 |
0.23
 2001-01-02 | amzn   |  16.00 |  13.63 |            |  13.88 |   9203500 |
13.88
 2001-01-02 | ge     | 360.58 | 327.88 |            | 336.54 |   4788901 |
188.05
 2001-01-02 | m      |  17.50 |  17.13 |            |  17.19 |   2583400 |
10.60
 2001-01-02 | f      |  24.94 |  23.75 |            |  24.31 |   5226500 |
13.33
 2001-01-02 | gs     | 105.50 |  97.50 |            | 100.25 |   3202400 |
77.89
 2001-01-02 | wmt    |  55.06 |  52.69 |            |  53.88 |   8813600 |
36.54
 2001-01-02 | ba     |  65.31 |  60.56 |            |  62.00 |   3762500 |
40.51
 2001-01-02 | pg     |  39.44 |  38.38 |            |  39.25 |   6847800 |
22.42
 2001-01-02 | coke   |  38.00 |  36.50 |            |  36.75 |      3800 |
27.26
(10 rows)

lab=#
```

Once you are done, return to Task 6.