**Querying Data from Multiple Redshift Spectrum Tables**

**Introduction**

In this lab, we will utilize Redshift Spectrum to create several tables from data stored in S3, and then test to ensure we are able to perform queries that include joins of the tables we have created.

**Scenario**

You have been tasked with testing accessing data stored in S3 from a Redshift cluster. Your team is hoping to realize cost savings by moving infrequently accessed non-latency sensitive data outside of your company's production Redshift cluster. They've provided the required tables below with the requisite column headers, as well as test data in an S3 bucket. You will need to create Redshift Spectrum tables with the appropriate DDL and run the provided test query to ensure Redshift Spectrum will function for the planned use case.

**Additional Resources**

[Github repo](https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/blob/master/Lab_Assets/querying_data_from_multiple_redshift_spectrum_tables/solution.sql)

[Redshift Create External Table](https://docs.aws.amazon.com/redshift/latest/dg/r_CREATE_EXTERNAL_TABLE.html)

**Required Tables**

**Names**  
id\_name  
id\_value  
gender  
name\_title  
name\_first  
name\_last

**Location**  
id\_name  
id\_value  
location\_street\_number  
location\_street\_name  
location\_city  
location\_state  
location\_country  
location\_postcode  
location\_coordinates\_latitude  
location\_coordinates\_longitude  
location\_timezone\_offset  
location\_timezone\_description  
nat

**Age**  
id\_name  
id\_value  
dob\_date  
dob\_age  
registered\_date  
registered\_age

**Contact**  
id\_name  
id\_value  
email  
phone  
cell

**Picture**  
id\_name  
id\_value  
picture\_large  
picture\_medium  
picture\_thumbnail

**Test Query**

select

names.name\_first as first\_name,

names.name\_last as last\_name,

location.location\_state as state,

age.dob\_age as age,

contact.cell as cell,

picture.picture\_large as picture

from users\_data.names

join users\_data.location on users\_data.names.id\_value = users\_data.location.id\_value

join users\_data.age on users\_data.names.id\_value = users\_data.age.id\_value

join users\_data.contact on users\_data.names.id\_value = users\_data.contact.id\_value

join users\_data.picture on users\_data.names.id\_value = users\_data.picture.id\_value

order by age

limit 10;

**Solution**

Log in to the AWS Management Console using the credentials provided for the lab. Make sure you're in the us-east-1 (N. Virginia) region.

**Inspect the Lab Environment**

1. Navigate to Amazon Redshift using the *Services* menu or the unified search bar. Ensure you're in the *Northern Virginia* region.
2. Select the **Clusters** link in the sidebar menu. You should have 1 cluster provided.
3. Select the **users-cluster** link to view the cluster details. The cluster's status should be *Available*.
4. Navigate to S3 using the *Services* menu or the unified search bar. You should have one S3 bucket named users-data-<ACCOUNT\_NUMBER> provided.
5. Select the **users-data-ACCOUNT\_NUMBER** bucket link.
6. Check the checkbox to the left of the **users\_1\_flat.json** object, then use the *Actions* dropdown to select **Download**.
7. Open the downloaded JSON file and review the data. You will create partitions based on the values in this data so you can run multiple queries.

**Create the Redshift Spectrum Tables**

**Connect to the Database and Create a Schema**

1. Navigate back to Amazon Redshift and select **Editor** in the sidebar menu to open the query editor.
2. On the right, click **Connect to database**.
3. On the *Connect to database* window, fill in the connection details:
   * For *Connection*, ensure **Create a new connection** is selected.
   * For *Authentication*, ensure **Temporary credentials** is selected.
   * Use the *Cluster* dropdown to select **users-cluster (Available)**.
   * In the *Database name* field, enter users.
   * In the *Database user* field, enter users\_admin.
4. Click **Connect**. The status of the cluster should now be *Connected*.
5. Copy the *users\_data* SQL query provided in the [Github repo](https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/blob/master/Lab_Assets/querying_data_from_multiple_redshift_spectrum_tables/solution.sql) and paste it into the query editor.
6. create external schema users\_data
7. from data catalog
8. database 'users'
9. iam\_role ''

create external database if not exists;

1. Copy the IAM role from the lab resources and paste it onto the *iam\_role* line.
2. Click **Run** to run the query.
3. After the query is complete, use the *Select schema* dropdown on the left to select **users\_data**. There is nothing in this schema yet, so you need to create your tables.

**Create the *names* table**

1. Clear the existing query out of the query editor.
2. Copy the *users\_data.names* SQL query provided in the [Github repo](https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/blob/master/Lab_Assets/querying_data_from_multiple_redshift_spectrum_tables/solution.sql) and paste it into the query editor.
3. create external table users\_data.names(
4. id\_name varchar(32),
5. id\_value varchar(64),
6. gender varchar(16),
7. name\_title varchar(32),
8. name\_first varchar(64),
9. name\_last varchar(64)
10. )
11. ROW FORMAT SERDE
12. 'org.openx.data.jsonserde.JsonSerDe'

LOCATION '';

1. Copy your S3 bucket name and paste it onto the *LOCATION* line using the format s3://users-data-<ACCOUNT\_NUMBER>.
2. Click **Run** to run the query. After the query is complete, you can see the *users\_data* schema on the left now shows the *names* table.
3. Test the *names* table:
   * Clear the *names* query out of the query editor.
   * Enter the following query:

select \* from users\_data.names limit 10;

* + Click **Run** to run the query. After the query is complete, you should see the results on screen below the *Query results* section. These results include all the variables you entered when you created the *names* table.

**Create the *location* table**

1. Clear the existing query out of the query editor.
2. Copy the *users\_data.location* SQL query provided in the [Github repo](https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/blob/master/Lab_Assets/querying_data_from_multiple_redshift_spectrum_tables/solution.sql) and paste it into the query editor.
3. create external table users\_data.location(
4. id\_name varchar(32),
5. id\_value varchar(32),
6. location\_street\_number int,
7. location\_street\_name varchar(64),
8. location\_city varchar(32),
9. location\_state varchar(32),
10. location\_country varchar(32),
11. location\_postcode varchar(32),
12. location\_coordinates\_latitude varchar(64),
13. location\_coordinates\_longitude varchar(64),
14. location\_timezone\_offset varchar(32),
15. location\_timezone\_description varchar(32),
16. nat varchar(16)
17. )
18. ROW FORMAT SERDE
19. 'org.openx.data.jsonserde.JsonSerDe'

LOCATION '';

1. Copy your S3 bucket name and paste it onto the *LOCATION* line using the format s3://users-data-<ACCOUNT\_NUMBER>.
2. Click **Run** to run the query. After the query is complete, you can see the *users\_data* schema on the left now shows the *location* table.

**Create the *age* table**

1. Clear the existing query out of the query editor.
2. Copy the *users\_data.age* SQL query provided in the [Github repo](https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/blob/master/Lab_Assets/querying_data_from_multiple_redshift_spectrum_tables/solution.sql) and paste it into the query editor.
3. create external table users\_data.age(
4. id\_name varchar(32),
5. id\_value varchar(32),
6. dob\_date varchar(32),
7. dob\_age int,
8. registered\_date varchar(32),
9. registered\_age int
10. )
11. ROW FORMAT SERDE
12. 'org.openx.data.jsonserde.JsonSerDe'

LOCATION '';

1. Copy your S3 bucket name and paste it onto the *LOCATION* line using the format s3://users-data-<ACCOUNT\_NUMBER>.
2. Click **Run** to run the query. After the query is complete, you can see the *users\_data* schema on the left now shows the *age* table.

**Create the *contact* table**

1. Clear the existing query out of the query editor.
2. Copy the *users\_data.contact* SQL query provided in the [Github repo](https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/blob/master/Lab_Assets/querying_data_from_multiple_redshift_spectrum_tables/solution.sql) and paste it into the query editor.
3. create external table users\_data.contact(
4. id\_name varchar(32),
5. id\_value varchar(32),
6. email varchar(32),
7. phone varchar(32),
8. cell varchar(32)
9. )
10. ROW FORMAT SERDE
11. 'org.openx.data.jsonserde.JsonSerDe'

LOCATION '';

1. Copy your S3 bucket name and paste it onto the *LOCATION* line using the format s3://users-data-<ACCOUNT\_NUMBER>.
2. Click **Run** to run the query. After the query is complete, you can see the *users\_data* schema on the left now shows the *contact* table.

**Create the *picture* table**

1. Clear the existing query out of the query editor.
2. Copy the *users\_data.picture* SQL query provided in the [Github repo](https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/blob/master/Lab_Assets/querying_data_from_multiple_redshift_spectrum_tables/solution.sql) and paste it into the query editor.
3. create external table users\_data.picture(
4. id\_name varchar(32),
5. id\_value varchar(32),
6. picture\_large varchar(64),
7. picture\_medium varchar(64),
8. picture\_thumbnail varchar(64)
9. )
10. ROW FORMAT SERDE
11. 'org.openx.data.jsonserde.JsonSerDe'

LOCATION '';

1. Copy your S3 bucket name and paste it onto the *LOCATION* line using the format s3://users-data-<ACCOUNT\_NUMBER>.
2. Click **Run** to run the query. After the query is complete, you can see the *users\_data* schema on the left now shows the *picture* table.

**Test Your Newly Created Redshift Spectrum Tables**

1. Clear the existing query out of the query editor.
2. Copy the test query provided as the last query in the [Github repo](https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/blob/master/Lab_Assets/querying_data_from_multiple_redshift_spectrum_tables/solution.sql) and paste it into the query editor.
3. select
4. names.name\_first as first\_name,
5. names.name\_last as last\_name,
6. location.location\_state as state,
7. age.dob\_age as age,
8. contact.cell as cell,
9. picture.picture\_large as picture
10. from users\_data.names
11. join users\_data.location on users\_data.names.id\_value = users\_data.location.id\_value
12. join users\_data.age on users\_data.names.id\_value = users\_data.age.id\_value
13. join users\_data.contact on users\_data.names.id\_value = users\_data.contact.id\_value
14. join users\_data.picture on users\_data.names.id\_value = users\_data.picture.id\_value
15. order by age

limit 10;

1. Click **Run**. The query ensures your tables are correctly formed and it is possible to perform joins between them. This may take a few moments while the query accesses each object record and filters out any records that do not meet the query criteria.

After the query is complete, you should see the results on screen below the *Query results* section. These results indicate that you have successfully joined the 5 Redshift Spectrum tables together.

**Conclusion**

Congratulations — you've completed this hands-on lab!