Section "1" **Amazon Redshift**
DATA 516 October, 2023

Objective

The goal of this tutorial is to get started with Amazon Redshift. In this tutorial will walk you through deploying a Redshift cluster, ingesting data, writing and running UDFs, and queries.

Redshift is available via AWS Academy so you will need an AWS Academy account for this. An activation link will be sent to your emails. If you have not received an account, please email the staff for assistance.

Accessing AWS Learner Lab

After logging into the portal(`https://www.awsacademy.com/LMS_Login`), you will be greeted with a Canvas like UI. This is not Canvas, but you should enrolled in a corresponding course: AWS Academy Learner Lab [59562]

Under Modules, enter the unit called "Learner Lab". Use the "Start Lab" button in the upper right to initialize a lab environment and after a few minutes the AWS status on the upper left will change to green.

Click on the status to enter a tailored version of AWS.

Note: There is a limited budget associated with each account, so it is important to kill any unused instances and we recommend only using `dc2.large` instance type for the homework and the tutorial.

Prerequisites

Before deploying a Redshift cluster you will need to set your regions:

- **Change Region**: In the upper right hand corner of screen you should see a region (e.g. Virginia). We will be working with containers on the west coast, so set your region to `us-west-2` (Oregon)

Deploy Redshift cluster

Now we are ready to deploy Amazon Redshift. On the Amazon web console click on Amazon Redshift under the databases section. On the Amazon Redshift Dashboard, click `Create cluster` and specify the following:

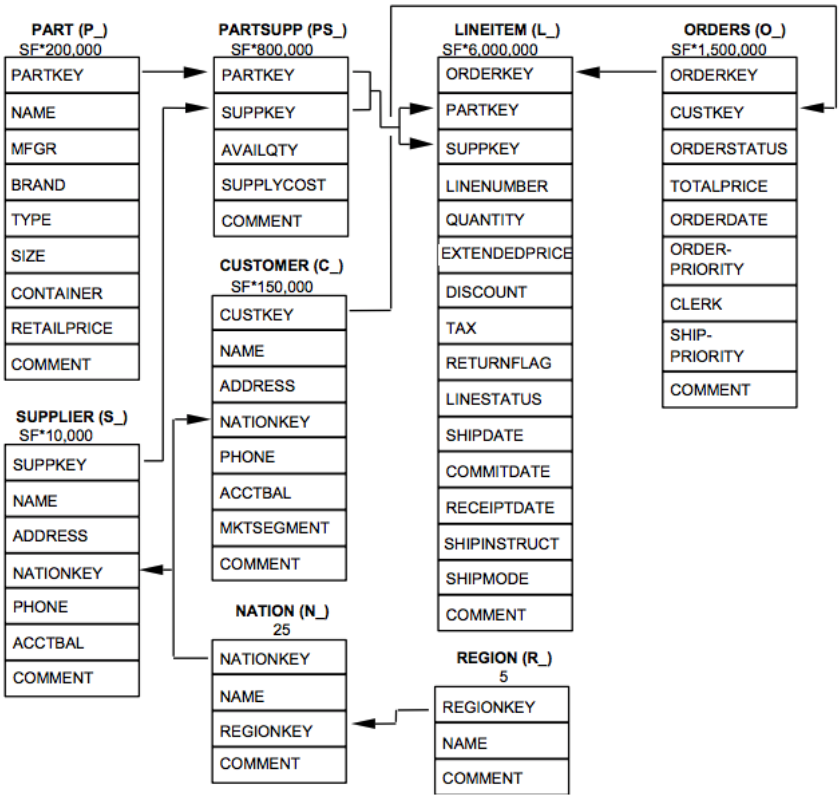
- Cluster identifier: e.g. `redshift-cluster-1`.
- Node Type: Select `dc2.large` as the node type.
- Number of Nodes: Select `2`
- Admin user name: `awsuser`. You will use this username and password to connect to your database after the cluster is available.

- Password: Select `Auto generate password`.
- Cluster Permissions: Associate the IAM Role `myRedshiftRole` (Open the role in another tab to reference later)

Click `Create cluster` and it will take around 5 to 10 minutes for the cluster to be `Available (or Maintenance)` in the `Status` column, before then it will show in the `Status` column that the cluster is `Modifying`.

The TPC-H Benchmark

In this assignment, we'll be working with TPC-H data. TPC-H is a standard decision support benchmark for big data analytics. It contains synthetic data and several queries representing a generalized complex analytics workload that answers critical business questions. The data size and distribution can be controlled via arguments while generating the synthetic data. Below is the schema of the benchmark database. We will be using TPC-H data two sets of different sizes for the homework: 1 GB and 10GB. We will not be using the standard TPC-H queries, however. you can learn more about TPC-H at:`http://www.tpc.org/tpch/`.



The figure above (from tpc.org) shows the TPC-H schema.

Connect and import data

Click on the "Query data" link in your cluster's webpage and select "Query in query editor". Then click on "Connect to database", and specify the following:

- Connection: "Create a new connection".

- Authentication: "Temporary credentials".
- Cluster: select your cluster.
- Database name: *dev*. You will connect to *dev* first, and create another database and work with this database created by you.
- Database user: *awsuser*.

Click on "connect" and you will be connected to *dev*.

Now create a new database called *tpch*: In the query editor, enter **create database tpch;'** and click on "Run".

Next, click on "Change connection" on the top right, and specify the following:

- Connection: "Create a new connection".
- Authentication: "Temporary credentials".
- Cluster: select your cluster.
- Database name: *tpch*.
- Database user: *awsuser*.

Click on "connect" and you will be connected to *tpch*.

To import data to your *tpch* database on your cluster, first create the relations and then import data into them. Following create table commands create tables in the Redshift cluster. A copy friendly version is available in the starter code folder.

```
CREATE TABLE customer(
C_CustKey int ,
C_Name varchar(64) ,
C_Address varchar(64) ,
C_NationKey int ,
C_Phone varchar(64) ,
C_AcctBal decimal(13, 2) ,
C_MktSegment varchar(64) ,
C_Comment varchar(120) ,
skip varchar(64)
);

CREATE TABLE lineitem(
L_OrderKey int ,
L_PartKey int ,
L_SuppKey int ,
L_LineNumber int ,
L_Quantity int ,
L_ExtendedPrice decimal(13, 2) ,
L_Discount decimal(13, 2) ,
L_Tax decimal(13, 2) ,
L_ReturnFlag varchar(64) ,
L_LineStatus varchar(64) ,
L_ShipDate datetime ,
```

```
L_CommitDate datetime ,
L_ReceiptDate datetime ,
L_ShipInstruct varchar(64) ,
L_ShipMode varchar(64) ,
L_Comment varchar(64) ,
skip varchar(64)
);

CREATE TABLE nation(
N_NationKey int ,
N_Name varchar(64) ,
N_RegionKey int ,
N_Comment varchar(160) ,
skip varchar(64)
);

CREATE TABLE orders(
O_OrderKey int ,
O_CustKey int ,
O_OrderStatus varchar(64) ,
O_TotalPrice decimal(13, 2) ,
O_OrderDate datetime ,
O_OrderPriority varchar(15) ,
O_Clerk varchar(64) ,
O_ShipPriority int ,
O_Comment varchar(80) ,
skip varchar(64)
);

CREATE TABLE part(
P_PartKey int ,
P_Name varchar(64) ,
P_Mfgr varchar(64) ,
P_Brand varchar(64) ,
P_Type varchar(64) ,
P_Size int ,
P_Container varchar(64) ,
P_RetailPrice decimal(13, 2) ,
P_Comment varchar(64) ,
skip varchar(64)
);

CREATE TABLE partsupp(
PS_PartKey int ,
PS_SuppKey int ,
PS_AvailQty int ,
PS_SupplyCost decimal(13, 2) ,
PS_Comment varchar(200) ,
skip varchar(64)
);

CREATE TABLE region(
R_RegionKey int ,
R_Name varchar(64) ,
R_Comment varchar(160) ,
```

```
skip varchar(64)
);

CREATE TABLE supplier(
S_SuppKey int ,
S_Name varchar(64) ,
S_Address varchar(64) ,
S_NationKey int ,
S_Phone varchar(18) ,
S_AcctBal decimal(13, 2) ,
S_Comment varchar(105) ,
skip varchar(64)
);
```

Once the tables are created, it's time to import data into the tables.
Run the following commands to import the data from S3 to your Redshift cluster.

Replace <> with the Role ARN of myRedshiftRole (make sure to remove the < and >).
The Role ARN should look something like `arn:aws:iam::788615689238:role/myRedshiftRole`.

```
copy customer from 's3://uwdb/tpch/uniform/1GB/customer.tbl' REGION 'us-west-2'
 CREDENTIALS 'aws_iam_role=<>' delimiter '|';

copy lineitem from 's3://uwdb/tpch/uniform/1GB/lineitem.tbl' REGION 'us-west-2'
 CREDENTIALS 'aws_iam_role=<>' delimiter '|';

copy nation from 's3://uwdb/tpch/uniform/1GB/nation.tbl' REGION 'us-west-2'
 CREDENTIALS 'aws_iam_role=<>' delimiter '|';

copy orders from 's3://uwdb/tpch/uniform/1GB/orders.tbl' REGION 'us-west-2'
 CREDENTIALS 'aws_iam_role=<>' delimiter '|';

copy part from 's3://uwdb/tpch/uniform/1GB/part.tbl' REGION 'us-west-2'
 CREDENTIALS 'aws_iam_role=<>' delimiter '|';

copy partsupp from 's3://uwdb/tpch/uniform/1GB/partsupp.tbl' REGION 'us-west-2'
 CREDENTIALS 'aws_iam_role=<>' delimiter '|';

copy region from 's3://uwdb/tpch/uniform/1GB/region.tbl' REGION 'us-west-2'
 CREDENTIALS 'aws_iam_role=<>' delimiter '|';

copy supplier from 's3://uwdb/tpch/uniform/1GB/supplier.tbl' REGION 'us-west-2'
 CREDENTIALS 'aws_iam_role=<>' delimiter '|';
```

For larger datasets, Redshift supports parallel upload. For this the data needs to be pre-split into multiple files. You can read more about how to do this here: `https://docs.aws.amazon.com/redshift/latest/dg/t_splitting-data-files.html`.

On my cluster, importing the 1GB dataset took about 5 minutes.

Run Queries

We're now ready to run a few queries on the TPC-H data:

- How many rows in the `lineitem` table? (6001215)

```
SELECT count(*) FROM lineitem
```

- What is the number of orders from each customer?

```
SELECT COUNT(distinct O_Orderkey) AS orders,
       c_name AS cust
       FROM customer
       JOIN orders ON C_CustKey=O_CustKey
       GROUP BY customer.c_name;
```

- Highest supply cost for each supplier

```
SELECT MAX(PS_SupplyCost) AS price, S_name AS supp
FROM supplier
JOIN partsupp ON PS_Suppkey=S_SuppKey
GROUP BY supplier.s_name, supplier.S_SuppKey
ORDER BY price DESC;
```

To view query runtimes select your cluster and then click on the "Execution" tab.

UDFs and Views

Amazon Redshift supports user defined functions(UDFs) in SQL and python. Once created the UDFs are stored in the Redshift database and can be invoked directly.

```
create function f_sql_greater_3 (float, float)
returns integer
stable
as $$
select case when $1 > $2 then 1
else 0
end
$$ language sql;
```

To use the UDF in a query (should return 3272745):

```
select sum(f_sql_greater_3(L_Discount, L_Tax)), count(*) from lineitem;
```

UDFs can also be in Python.

```
create function f_py_greater (a float, b float)
returns integer
stable
as $$
if a > b:
  return True
else:
  return False
$$ language plpythonu;
```

To use the UDF in a query (takes a while, should return 3272745):

```
select sum(f_py_greater (L_Discount, L_Tax)) from lineitem;
```

Always remember to pause your RedShift clusters when they are not in use! Amazon's free trial will allow you to continuously run a cluster for about two weeks, so make sure you pause or terminated your instances. However, note that you may choose to use the RedShift cluster in your project: for that reason, we recommend that you pause it for now, and remember to terminate it later, after the project.