Section 1 **Amazon Redshift**
DATA 516 October, 2020

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.

As we discussed in the previous section, Redshift is not available via AWS Educate so you will need a "normal" AWS account for this. Information on Redshift free tier is available at: //aws.amazon.com/redshift/free-trial. Only use `dc2.large` instance type for the homework and the tutorial.

Prerequisites

Before deploying a redshift cluster you will need to fulfill two prerequisites:

- **Create IAM role**: Sign-in to the AWS web console. Follow the link to Identity and Access Management (IAM) page under Security, Identity and Compliance section on the AWS web console.
    1. Click on **Roles** on the left panel and click on the blue button to create role.
    2. Select type of trusted entity: AWS Service.
    3. Choose the service that will use this role: Redshift.
    4. Select your use case: Redshift - Customizable.
    5. Click Next: Permissions.
    6. On Attach policy page choose **AmazonS3ReadOnlyAccess**, **AWSGlueConsole-FullAccess**, and **AmazonRedshiftQueryEditor**, click next
    7. Name your role `RedshiftRole`.
    8. Review, create role and copy the **Role ARN** to your clipboard or a file on your laptop.
- If you have a super old AWS account (created before 2013) you will need to create a default VPC. Instructions for creating a default VPC are here.

Deploy Redshift cluster

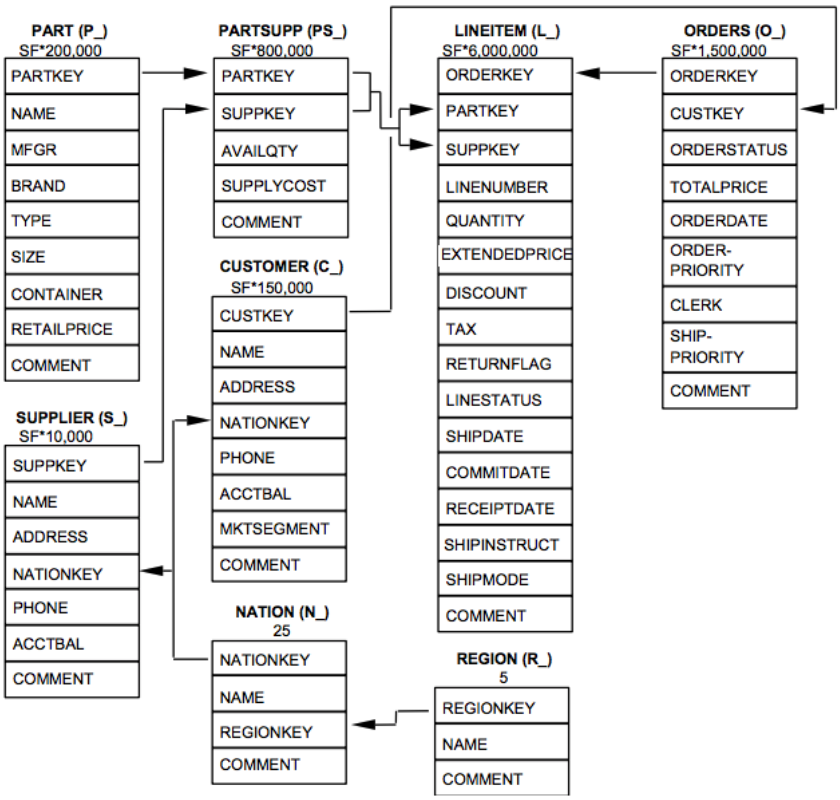
Now we are ready to deploy Amazon Redshift. On the Amazon web console click on Amazon Redshift under the databases section. On the Amazon Redshift Dashboard, choose Launch Cluster and specify the following:

- Node type: `dc2.large` (note: `dc2.large` is the recommended instance for both this section and the homework)
- Number of compute nodes: 2
- Cluster Identifier: e.g. *testcluster*.
- Database Name: e.g. *tpch*.

- Database Port: type the port number on which the database will accept connections (*5439*).
- Master User Name: *awsuser*. You will use this username and password to connect to your database after the cluster is available.
- Password: type a password for the master user account.
- Pick the IAM role you created in the previous step

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

Connect to the cluster Once the cluster is deployed (should take 5-10 minutes) click on the "Query Editor" link in your browser. Enter the username, password, and database name (probably "tpch") you used when launching your cluster.

To import data into your cluster, first create the relations and then import data into them. Following create table commands create tables in the Redshift cluster. Note that the Redshift browser-based query editor will only execute one query at a time, even if you paste them all into the editor.

```
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. To do this you need the Role ARN from the redshift role created in the prerequisites. Run the following commands to import the data from S3 to your Redshift cluster. Replace <> with the Role ARN created previously (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 orders from 's3://uwdb/tpch/uniform/1GB/orders.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 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 at here.

On my cluster, importing the 1GB dataset took about ten 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

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

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

Redshift Spectrum

Redshift spectrum enables you to run SQL queries against data in Amazon S3. First create an external database:

```
create external schema tpchs3
from data catalog
database 'tpchs3'
iam_role 'arn:aws:iam::<>'
create external database if not exists;
```

Next create the external tables, these tables do not import data locally,

```
create external table tpchs3.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))
row format delimited
fields terminated by '|'
stored as textfile
location 's3://uwdb/tpch/athena/customer/'
table properties ('numRows'='150000');
```

You can now run queries against these external tables.

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;
```