Session 2: Database Setup and Sqoop Code Run on AWS EMR

Step 1: Connect to AWS EMR (via PuTTY, etc.). You have already learnt how to connect to AWS EMR in the previous modules.

Note: Change to root user on AWS EMR, if it is not root user by default, by typing the following command on the terminal: **sudo -i**.

Step 2: Log in to MySQL in your AWS EMR terminal [Username for MySQL: root; Password: 123].

```
mysql -u root -p
Enter Password:
123
```

```
[root@ip-172-31-45-198 ~]# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 5.5.68-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

You can now exit MySQL to start downloading the MySQL data sets.

In order to download the data set from the S3 location, you have to run following commands:

```
wget -P /root/ https://s3.amazonaws.com/sqoop.dse.2020/flights_data.txt

wget -P /root/ https://s3.amazonaws.com/sqoop.dse.2020/online_data.txt
```

Once the data files are downloaded to the local AWS EMR instance, they need to be moved from /root to some HDFS location for further analysis (you will see further).

Step 3: Now, first, we have to create a database and then create some tables inside it so that we can further load the documents and data into them. These tables would be used for demonstrating the import-all-table command for Sqoop.

The following commands are used to move the data from the local root directory to the HDFS location, so it could be further exported to MySQL.

```
[root@ip-172-31-45-59 ~]# hadoop fs -put /root/flights_data.txt
/user/root/flights_data
[root@ip-172-31-45-59 ~]# hadoop fs -put /root/online_data.txt
/user/root/online_data
```

After this, verify the data in the HDFS location by using the following commands:

```
[root@ip-172-31-45-59 ~]# hadoop fs -ls /user/root/flights_data
[root@ip-172-31-45-59 ~]# hadoop fs -ls /user/root/online_data
```

Next, run the following commands to create the tables:

```
mysql>CREATE DATABASE recipes_database;

mysql>USE recipes_database;

mysql>CREATE TABLE recipes (recipe_id INT NOT NULL, recipe_name VARCHAR(30)
NOT NULL, PRIMARY KEY (recipe_id), UNIQUE (recipe_name));

mysql>INSERT INTO recipes (recipe_id, recipe_name) VALUES (1,'Tacos'),
(2,'Tomato Soup'), (3, 'Grilled Cheese');

mysql>CREATE TABLE ingredients ( ingredient_id INT NOT NULL,
```

```
ingredient_name VARCHAR(30) NOT NULL, ingredient_price INT NOT NULL,
PRIMARY KEY (ingredient_id),  UNIQUE (ingredient_name));

mysql>INSERT INTO ingredients (ingredient_id, ingredient_name,
ingredient_price) VALUES (1, 'Beef', 5), (2, 'Lettuce', 1), (3, 'Tomatoes',
2), (4,'Taco Shell', 2), (5, 'Cheese', 3), (6, 'Milk', 1), (7, 'Bread', 2);


mysql>CREATE TABLE recipe_ingredients ( recipe_id int NOT NULL,
ingredient_id INT NOT NULL, amount INT NOT NULL, PRIMARY KEY
(recipe_id,ingredient_id));

mysql>INSERT INTO recipe_ingredients (recipe_id, ingredient_id, amount)
VALUES (1,1,1), (1,2,2), (1,3,2), (1,4,3), (1,5,1), (2,3,2), (2,6,1),
(3,5,1), (3,7,2);
```

NOTE: Please make sure that you use the single quote and not double quotes while entering the ingredient names in the tables.

Step 4: Now, you can run the following commands (one-by-one) to verify whether the data has been loaded successfully into the tables

```
mysql>show tables;

mysql>select * from ingredients;

mysql>select * from recipe_ingredients;

mysql>select * from recipes;
```

Step 5: Next, you need to create a database, which you would use for the rest of the sqoop import and export commands

```
mysql>create database test;

mysql>use test;

mysql>create table employee (id INT, first_name VARCHAR(150), designation
VARCHAR(150), salary INT, PRIMARY KEY (id));

mysql>insert into employee values (100,'Harbhajan','Software
```

```
Engineer',5000);

mysql>insert into employee values (101,'Yuvraj','Senior Software
Engineer',7000);

mysql>insert into employee values (102,'MS Dhoni','Manager',10000);

mysql>insert into employee values (103,'Sachin Tendulkar','Senior
Manager',11000);

mysql>insert into employee values (104,'Virat Kohli',null, 7000);

mysql>select * from employee;

mysql>create table test.retailinfo(invoiceno varchar(150),stockcode
varchar(150),description varchar(150),quantity int,invoicedate
varchar(150),unitprice double,customerid int,country varchar(150));

mysql>create table test.flights_info (destination VARCHAR(150), origin
VARCHAR(150), count INT);

mysql> show tables;
```

Step 6: Practice the following Sqoop commands as you go through the videos in the segments ahead.

Note: We will import the data from the database into directories in Hadoop. Make sure you do not have directories with the same name created already, to avoid conflict issues.

Note: You can follow the following steps using the **hadoop** user in EMR as well. Please make sure that you substitute the corresponding paths with the hadoop directory paths accordingly.

Note: Please make sure that you are changing the <Public DNS> part in the commands below with the Public DNS of the Master Node of your EMR Cluster.

Master public DNS: ec2-3-92-244-104.compute-1.amazonaws.com

Connect to the Master Node Using SSH

Segment 5 - Exporting Data: Sqoop Export

Step 1: In order to load the data from a file in the HDFS to an RDBMS, you can use the Sqoop export command. Run the following commands:

```
[root@ip-172-31-45-59 ~]#sqoop export --connect jdbc:mysql://<Public
DNS>:3306/test \
--table retailinfo \
--username root --password 123 \
--export-dir /user/root/online_data \
--fields-terminated-by ',' --lines-terminated-by '\n'
```

Step 2: Similarly, we will export flight data, using the Sqoop export command

```
[root@ip-172-31-45-59 ~]#sqoop export --connect jdbc:mysql://<Public
DNS>:3306/test \
--table flights_info \
--username root --password 123 \
--export-dir /user/root/flights_data \
--fields-terminated-by ',' --lines-terminated-by '\n'
```

Step 3: After exporting the data using Sqoop, data verification can be done by looking at the data in MySQL, as shown below:

```
mysql> select count(*) from test.retailinfo;

mysql> select count(*) from test.flights_info;
```

Segment 6 - Importing Data: Sqoop Import

This is a basic Sqoop import command, which is used to pull data from MySQL and insert it into a specific target directory in the HDFS.

Step 1: Before running the Sqoop command, for verification, we have to check how many records are present in the MySQL table.

```
mysql>select count(*) from flights_info;
```

Note: Before running the Sqoop import command, make sure the target directory is not already created; otherwise, the Sqoop import command will throw an error.

Step 2: After getting a count of the records in the MySQL table, you have to verify the number of records imported using the Sqoop command. So, run the Sqoop import command and verify.

```
[root@ip-172-31-45-59 ~]# hadoop fs -rm -r
/user/root/flights_basic_command


[root@ip-172-31-45-59 ~]#sqoop import \
--connect jdbc:mysql://<Public DNS>:3306/test \
--table flights_info \
--username root --password 123 \
--target-dir /user/root/flights_basic_command \
-m 1

[root@ip-172-31-45-59 ~]# hadoop fs -ls
/user/root/flights_basic_command
```

After importing the data from Sqoop, if the count of the records matches the count of the records in MySQL, then it means that the import has been successful.

Segment 7 – Importing Data: Importing all Tables using sqoop

Sqoop has a command for importing all MySQL tables belonging to a specific database in a single execution itself.

Before executing the command, make sure to count the number of tables in MySQL, because the same number of target directories would be created in the home Hadoop location, i.e., '**hadoop fs -ls /user/root/**'.

Note: A point to be noted here is that all the tables in MySQL should have a primary key defined; otherwise, the command will throw an error.

Step 1: Run the following command:

```
[root@ip-172-31-45-59 ~]# hadoop fs -rm -r /user/root/recipes


[root@ip-172-31-45-59 ~]# hadoop fs -rm -r
/user/root/recipe_ingredients


[root@ip-172-31-45-59 ~]# hadoop fs -rm -r /user/root/ingredients


[root@ip-172-31-45-59 ~]# sqoop import-all-tables \
--connect jdbc:mysql://<Public DNS>:3306/recipes_database \
--username root \
--password 123


[root@ip-172-31-45-59 ~]# hadoop fs -ls /user/root/ingredients


[root@ip-172-31-45-59 ~]# hadoop fs -ls /user/root/recipes


[root@ip-172-31-45-59 ~]# hadoop fs -ls /user/root/recipe_ingredients
```

Step 2: After executing the import-all-tables command, verify the data in the Hadoop home location of the corresponding user. You should have the same number of directories corresponding to the number of tables in MySQL

Segment 8 – Importing Data: Handling NULL Values

Step 1: Sqoop has a command that provides an option to handle NULL values as well. If we do not put this clause, then Sqoop will import NULL values as a string, with the values of these strings written as 'null'. Hence, to make sure NULL values are imported as they are in Hadoop, select the following clause while importing data using Sqoop:

```
[root@ip-172-31-45-59 ~]# hadoop fs -rm -r
/user/root/employee_null_command

[root@ip-172-31-45-59 ~]#sqoop import \
--connect jdbc:mysql://<Public DNS>:3306/test \
--table employee \
--username root --password 123 \
--null-string '\\N' --null-non-string '\\N' \
--target-dir /user/root/employee_null_command \
-m 1

[root@ip-172-31-45-59 ~]# hadoop fs -ls
/user/root/employee_null_command

[root@ip-172-31-45-59 ~]# hadoop fs -cat
/user/root/employee_null_command/part-m-00000
```

Segment 9 – Importing Data: Handling Mappers for a Sqoop job

Sqoop has a command that provides an option to handle the number of mappers as well. You can increase and decrease the number of mappers as per your cluster memory.

The number of part files getting created is equal to the number of mappers used in the Sqoop command.

Step 1: Run the following command:

```
[root@ip-172-31-45-59 ~]# hadoop fs -rm -r
/user/root/employee_mapper_command
```

```
[root@ip-172-31-45-59 ~]# sqoop import \
--connect jdbc:mysql://<Public DNS>:3306/test \
--table employee \
--username root --password 123 \
--target-dir /user/root/employee_mapper_command \
--split-by id -m 4

[root@ip-172-31-45-59 ~]# hadoop fs -ls
/user/root/employee_mapper_command
```

After running this Sqoop command, the number of files getting created in the target directory should be equal to the number of mappers used in the Sqoop command.

Note: While specifying the number of mappers, make sure to provide the split id column, which is usually the primary key of that table.

Segment 10 – Importing Data: Importing Data in Different File Formats

Sqoop can be used to import data in different file formats. It does not matter how the data is stored in MySQL; when you provide the option in the Sqoop command to store the data in a specific file format, then the Hadoop MapReduce job running at the backend takes care of it automatically.

Step 1: Run the following command:

```
[root@ip-172-31-45-59 ~]# hadoop fs -rm -r
/user/root/flights_parquet_command

[root@ip-172-31-45-59 ~]# sqoop import \
--connect jdbc:mysql://<Public DNS>:3306/test \
--table flights_info \
--username root --password 123 \
--target-dir /user/root/flights_parquet_command \
-m 1 \
--as-parquetfile
```

The command above will store the data as a '**parquet file**' in the target directory.

Step 2: Now run the following command:

```
[root@ip-172-31-45-59 ~]# hadoop fs -rm -r
/user/root/flights_sequence_command

[root@ip-172-31-45-59 ~]# sqoop import --connect jdbc:mysql://<Public
DNS>:3306/test \
--table flights_info \
--username root --password 123 \
--target-dir /user/root/flights_sequence_command \
-m 1 --as-sequencefile
```

The command above will store the data as a '**sequence file**' in the target directory.

Segment 11 – Importing Data: Compression using Sqoop

We can compress the data as well while using the sqoop import command.

Step 1: In order to verify whether the data is getting compressed, first, run the basic Sqoop command and note down the size of the file in the target directory

```
[root@ip-172-31-45-59 ~]# hadoop fs -rm -r
/user/root/flights_sequence_command

[root@ip-172-31-45-59 ~]# sqoop import --connect jdbc:mysql://<Public
DNS>:3306/test \
--table flights_info \
--username root --password 123 \
--target-dir /user/root/flights_sequence_command \
-m 1 --as-sequencefile
```

Step 2: Next, run the Sqoop command with the compression codec enabled

```
[root@ip-172-31-45-59 ~]# hadoop fs -rm -r
/user/root/flights_compress_command

[root@ip-172-31-45-59 ~]# sqoop import --connect jdbc:mysql://<Public
DNS>:3306/test \
--table flights_info \
```

```
--username root --password 123 \
--target-dir /user/root/flights_compress_command \
-m 1 --as-sequencefile \
--compression-codec org.apache.hadoop.io.compress.SnappyCodec
```

Step 3: Navigate to the target directories and note down the file sizes with and without compression enabled

```
hadoop fs -ls /user/root/flights_sequence_command

hadoop fs -ls /user/root/flights_compress_command
```