

Task 2. Use Sqoop command to ingest the data from RDS into the HBase Table.

Solution: We followed the below steps to complete the above task:

1. First, we logged into the EMR cluster and switch to root user by running **sudo -i** command.

```
[hadoop@ip-172-31-12-206 ~]$ sudo -i

EEEEEEEEEEEEEEEEEEEE MMMMMMMM          MMMMMMMM RRRRRRRRRRRRRRRR
E::::::::::::::::::::E M::::::::M          M::::::::M R:::::::::R
EE::::::::EEEEEEEE::E M::::::::M          M::::::::M R:::::::::R
  E::::E          EEEEE M::::::::M          M::::::::M RR::::R      R::::R
  E::::E          M:::::M:M          M::M::::M      R:::R      R::::R
  E::::EEEEEEEEEE M:::::M M::M M::M M::::M      R::RRRRRR::::R
  E::::::::::E M:::::M M::M:M M::::M      R:::::::::RR
  E::::EEEEEEEEEE M:::::M M::::M M::::M      R::RRRRRR::::R
  E::::E          M:::::M M::M M::::M      R:::R      R::::R
  E::::E          EEEEE M:::::M          MMM M:::::M      R:::R      R::::R
EE::::::::EEEEEEEE::E M:::::M          M:::::M      R:::R      R::::R
E::::::::::::::::::::E M:::::M          M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMMM          MMMMMMMM RRRRRRRR      RRRRRR
```

2. Next, we need to run the below commands to install the MySQL connector jar file:

wget <https://de-mysql-connector.s3.amazonaws.com/mysql-connector-java-8.0.25.tar.gz>

tar -xvf mysql-connector-java-8.0.25.tar.gz

```
root@ip-172-31-12-206:~# wget https://de-mysql-connector.s3.amazonaws.com/mysql-connector-java-8.0.25.tar.gz
--2024-04-09 06:59:00-- https://de-mysql-connector.s3.amazonaws.com/mysql-connector-java-8.0.25.tar.gz
Resolving de-mysql-connector.s3.amazonaws.com (de-mysql-connector.s3.amazonaws.com)... 52.217.73.132, 52.217.172.73, 54.231.163.81, ...
Connecting to de-mysql-connector.s3.amazonaws.com (de-mysql-connector.s3.amazonaws.com)|52.217.73.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4079310 (3.9M) [application/x-gzip]
Saving to: 'mysql-connector-java-8.0.25.tar.gz'

100%[=====>] 4,079,310 --.-K/s in 0.05s

2024-04-09 06:59:00 (82.0 MB/s) - 'mysql-connector-java-8.0.25.tar.gz' saved [4079310/4079310]

[root@ip-172-31-12-206 ~]# tar -xvf mysql-connector-java-8.0.25.tar.gz
mysql-connector-java-8.0.25/
mysql-connector-java-8.0.25/src/
mysql-connector-java-8.0.25/src/build/
mysql-connector-java-8.0.25/src/build/java/
mysql-connector-java-8.0.25/src/build/java/documentation/
mysql-connector-java-8.0.25/src/build/java/instrumentation/
mysql-connector-java-8.0.25/src/build/misc/
mysql-connector-java-8.0.25/src/build/misc/debian.in/
mysql-connector-java-8.0.25/src/build/misc/debian.in/source/
mysql-connector-java-8.0.25/src/demo/
mysql-connector-java-8.0.25/src/demo/java/
mysql-connector-java-8.0.25/src/demo/java/demo/
mysql-connector-java-8.0.25/src/demo/java/demo/x/
mysql-connector-java-8.0.25/src/demo/java/demo/x/devapi/
mysql-connector-java-8.0.25/src/generated/
mysql-connector-java-8.0.25/src/generated/java/
mysql-connector-java-8.0.25/src/generated/java/com/
mysql-connector-java-8.0.25/src/generated/java/com/mysql/
mysql-connector-java-8.0.25/src/generated/java/com/mysql/cj/
mysql-connector-java-8.0.25/src/generated/java/com/mysql/cj/x/
mysql-connector-java-8.0.25/src/generated/java/com/mysql/cj/x/protobuf/
mysql-connector-java-8.0.25/src/legacy/
mysql-connector-java-8.0.25/src/legacy/java/
mysql-connector-java-8.0.25/src/legacy/java/com/
```

cd mysql-connector-java-8.0.25/

sudo cp mysql-connector-java-8.0.25.jar /usr/lib/sqoop/lib/

```
[root@ip-172-31-12-206 ~]# cd mysql-connector-java-8.0.25/
[root@ip-172-31-12-206 mysql-connector-java-8.0.25]# sudo cp mysql-connector-java-8.0.25.jar /usr/lib/sqoop/lib/
```

3. Next, for ingesting data from AWS RDS's MySQL database instance to HBase table, we ran the below command:

```
scoop import \  
--connect jdbc:mysql://myrdsdatabaseinstance.c32geggok7gd.us-east-1.rds.amazonaws.com/TaxiDB \  
--username admin \  
--password admin123 \  
--table TripData \  
--hbase-create-table \  
--hbase-table TripData_hbase \  
--column-family cf1 \  
--hbase-row-key TripID \  
--hbase-bulkload \  
--split-by payment_type \  
-m 20
```

Explanation for the above command is as follows:

- **scoop import** – is a command used for importing data
- **--connect** – specifies the JDBC string of the MySQL database
- **--username** – specifies the username to connect to the MySQL database
- **--password** – specifies the password to connect to the MySQL database
- **--table** – specifies the MySQL table name from where the data will be imported
- **--hbase-create-table** – used to create the HBase table if not existing already
- **--hbase-table** – specifies the name of the HBase table into which the data will be imported to
- **--column-family cf1** – specifies the column family in HBase table
- **--hbase-row-key** – specifies the column which will be the row key for the HBase table
- **--hbase-bulkload** – used for faster data loading when data volume is very large
- **--split-by** – specifies the column based on which the HBase regions will be created
- **-m 20** – specifies the number of mappers

```
[root@ip-172-31-12-206 mysql-connector-java-8.0.25]# scoop import \  
> --connect jdbc:mysql://myrdsdatabaseinstance.c32geggok7gd.us-east-1.rds.amazonaws.com/TaxiDB \  
> --username admin \  
> --password admin123 \  
> --table TripData \  
> --hbase-create-table \  
> --hbase-table TripData_hbase \  
> --column-family cf1 \  
> --hbase-row-key TripID \  
> --hbase-bulkload \  
> --split-by payment_type \  
> -m 20  
Warning: /usr/lib/scoop/./hcatalog does not exist! HCatalog jobs will fail.  
Please set $HCAT_HOME to the root of your HCatalog installation.  
Warning: /usr/lib/scoop/./accumulo does not exist! Accumulo imports will fail.  
Please set $ACCUMULO_HOME to the root of your Accumulo installation.  
24/04/09 07:01:02 INFO scoop.Scoop: Running Scoop version: 1.4.7  
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: Found binding in [jar:file:/usr/share/aws/redshift/jdbc/redshift-jdbc42-1.2.37.1061.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.  
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]  
24/04/09 07:01:02 WARN tool.BaseScoopTool: Setting your password on the command-line is insecure. Consider using -P instead.  
24/04/09 07:01:02 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.  
24/04/09 07:01:02 INFO tool.CodeGenTool: Beginning code generation  
Loading class 'com.mysql.jdbc.Driver'. This is deprecated. The new driver class is 'com.mysql.cj.jdbc.Driver'. The driver is automatically registered via  
SPI and manual loading of the driver class is generally unnecessary.  
24/04/09 07:01:03 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM 'TripData' AS t LIMIT 1  
24/04/09 07:01:03 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM 'TripData' AS t LIMIT 1
```

```

root@ip-172-31-12-206:~/mysql-connector-java-8.0.25
Total megabyte-milliseconds taken by all reduce tasks=5089806336
Map-Reduce Framework
  Map input records=18880595
  Map output records=358731305
  Map output bytes=27429395833
  Map output materialized bytes=4689302570
  Input split bytes=591
  Combine input records=0
  Combine output records=0
  Reduce input groups=18880595
  Reduce shuffle bytes=4689302570
  Reduce input records=358731305
  Reduce output records=358731305
  Spilled Records=1260407481
  Shuffled Maps=5
  Failed Shuffles=0
  Merged Map outputs=5
  GC time elapsed (ms)=27105
  CPU time spent (ms)=3700050
  Physical memory (bytes) snapshot=5016383488
  Virtual memory (bytes) snapshot=21304918016
  Total committed heap usage (bytes)=4061659136

Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0

File Input Format Counters
  Bytes Read=0
File Output Format Counters
  Bytes Written=18362207532
24/04/09 07:49:37 INFO mapreduce.ImportJobBase: Transferred 17.1011 GB in 2,903.2981 seconds (6.0316 MB/sec)
24/04/09 07:49:37 INFO mapreduce.ImportJobBase: Retrieved 358731305 records.
24/04/09 07:49:37 WARN mapreduce.LoadIncrementalHFiles: managed connection cannot be used for bulkload. Creating unmanaged connection.
24/04/09 07:49:37 WARN mapreduce.LoadIncrementalHFiles: Skipping non-directory hdfs://ip-172-31-12-206.ec2.internal:8020/user/root/TripData/_SUCCESS
24/04/09 07:49:37 INFO impl.MetricsConfig: loaded properties from hadoop-metrics2-hbase.properties
24/04/09 07:49:37 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
24/04/09 07:49:37 INFO impl.MetricsSystemImpl: HBase metrics system started
24/04/09 07:49:37 WARN mapreduce.LoadIncrementalHFiles: Trying to bulk load hfile hdfs://ip-172-31-12-206.ec2.internal:8020/user/root/TripData/cf1/5173d20607a544739948b74337416357 with size: 11433635969 bytes can be problematic as it may lead to oversplitting.
24/04/09 07:49:37 INFO Configuration.deprecation: hbase.offheapcache.minblocksize is deprecated. Instead, use hbase.blockcache.minblocksize

```

- Next, we entered the shell of HBase by entering **HBase shell** command.

```

[root@ip-172-31-12-206 mysql-connector-java-8.0.25]# hbase shell
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 1.4.13, rUnknown, Fri Apr 17 15:18:24 UTC 2020

```

- Next, we ran the command **list** to see the list of tables present (it should have the HBase table we created).

```

hbase(main):001:0> list
TABLE
TripData_hbase
1 row(s) in 0.2750 seconds

=> ["TripData_hbase"]

```

- Next, we ran the command **describe 'TripData_hbase'** to see the table related details.

```

hbase(main):002:0> describe 'TripData_hbase'
Table TripData_hbase is ENABLED
TripData_hbase
COLUMN FAMILIES DESCRIPTION
(NAME => 'cf1', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0')
1 row(s) in 0.1160 seconds

```

7. Next, we ran the command **count 'TripData_hbase'** to see how many rows are present based on the row key.

root@ip-172-31-12-206:~/mysql-connector-java-8.0.25

```
hbase(main):003:0> count 'TripData_hbase'
Current count: 1000, row: 10000895
Current count: 2000, row: 10001795
Current count: 3000, row: 10002695
Current count: 4000, row: 10003595
Current count: 5000, row: 10004495
Current count: 6000, row: 10005395
Current count: 7000, row: 10006295
Current count: 8000, row: 10007195
Current count: 9000, row: 10008095
Current count: 10000, row: 10008996
Current count: 11000, row: 10009896
Current count: 12000, row: 10010795
Current count: 13000, row: 10011695
Current count: 14000, row: 10012595
Current count: 15000, row: 10013495
Current count: 16000, row: 10014395
Current count: 17000, row: 10015295
Current count: 18000, row: 10016195
Current count: 19000, row: 10017095
Current count: 20000, row: 10017996
Current count: 21000, row: 10018896
Current count: 22000, row: 10019796
Current count: 23000, row: 10020695
Current count: 24000, row: 10021595
Current count: 25000, row: 10022495
Current count: 26000, row: 10023395
Current count: 27000, row: 10024295
Current count: 28000, row: 10025195
Current count: 29000, row: 10026095
Current count: 30000, row: 10026996
Current count: 31000, row: 10027896
Current count: 32000, row: 10028796
Current count: 33000, row: 10029696
Current count: 34000, row: 10030595
Current count: 35000, row: 10031495
Current count: 36000, row: 10032395
Current count: 37000, row: 10033295
Current count: 38000, row: 10034195
Current count: 39000, row: 10035095
Current count: 40000, row: 10035996
Current count: 41000, row: 10036896
Current count: 42000, row: 10037796
```