

Task 2: Using Sqoop commands to ingest the data from RDS into the HBase Table.

1. First, we log in into the EMR instance and complete the initial steps of setup.
 - Now we run the following command to install the MySQL connector jar file

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

```
hadoop@ip-172-31-56-92:~$
1 row in set (49.48 sec)

MySQL [yellow_taxi]> exit;
Bye
[hadoop@ip-172-31-56-92 ~]$ wget https://de-mysql-connector.s3.amazonaws.com/mysql-connector-java-8.0.25.tar.gz
--2023-09-06 02:33:29-- 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)... 54.231.196.217, 3.5.8.196, 3.5.28.243, ...
Connecting to de-mysql-connector.s3.amazonaws.com (de-mysql-connector.s3.amazonaws.com)|54.231.196.217|: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 10.9MB/s in 0.2s

2023-09-06 02:33:30 (18.9 MB/s) - 'mysql-connector-java-8.0.25.tar.gz' saved [4079310/4079310]
```

- Now, we run the following step to extract the MySQL connector tar file

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

```
2023-09-06 02:33:30 (18.9 MB/s) - 'mysql-connector-java-8.0.25.tar.gz' saved [4079310/4079310]

[hadoop@ip-172-31-56-92 ~]$ 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/
```

- Now, we go to the MySQL Connector directory created in the previous step and then copy it to the Sqoop library to complete the installation.

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

```
hadoop@ip-172-31-56-92:~/mysql-connector-java-8.0.25$
mysql-connector-java-8.0.25/src/test/java/testsuite/x/internal/package-info.java
[hadoop@ip-172-31-56-92 ~]$ cd mysql-connector-java-8.0.25/
[hadoop@ip-172-31-56-92 mysql-connector-java-8.0.25]$ sudo cp mysql-connector-java-8.0.25.jar /usr/lib/sqoop/lib/
[hadoop@ip-172-31-56-92 mysql-connector-java-8.0.25]$ mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!
```

2. Having now installed the MySQL Connector. Now, we set up MySQL on EMR cluster and proceed
3. We then run the following command to ingest data from mySQL RDS to HBase table;

Note: --hbase-create-table : creates an HBase table if it does not exist

```
sqoop import --connect
jdbc:mysql://dbinstance.ca1depqqnwki.us-east-1.rds.amazonaws.com/yellow_taxi \
--username admin \
--password admin123 \
--table trip_records \
--hbase-table trip_records_hbase \
--column-family colfam1 \
--hbase-create-table \
--hbase-row-key tpep_pickup_datetime, tpep_dropoff_datetime \
--hbase-bulkload \
--split-by payment_type
```

4. Code explanation :

This Sqoop command transfers data from a MySQL database table named trip_records to an HBase table called trip_records_hbase. The function of each option in the command is broken down below;

- split-by : specifies a column from the MySQL table that will be used to split data into multiple HBase regions
- hbase-bulkload : uses HBase bulk load feature for faster data loading
- hbase-row-key : specifies one or more columns from the MySQL table that will be used as the row key in HBase
- hbase-create-table : creates an HBase table if it does not exist
- column-family : specifies the name of the column family in HBase where the imported data will be stored
- hbase-table : specifies the name of the HBase table to import data into
- table : specifies the name of the MySQL table to import data from
- password : specifies the password to use when connecting to the MySQL database
- username : specifies the username to use when connecting to the MySQL database

--connect : specifies the JDBC connection string for the MySQL database

```
root@ip-172-31-56-92:~#
Total vcore-milliseconds taken by all map tasks=2328553
Total vcore-milliseconds taken by all reduce tasks=2169934
Total megabyte-milliseconds taken by all map tasks=3576657408
Total megabyte-milliseconds taken by all reduce tasks=6666037248

Map-Reduce Framework
  Map input records=18880595
  Map output records=302089520
  Map output bytes=42389675891
  Map output materialized bytes=4956326622
  Input split bytes=591
  Combine input records=0
  Combine output records=0
  Reduce input groups=18842048
  Reduce shuffle bytes=4956326622
  Reduce input records=302089520
  Reduce output records=301472768
  Spilled Records=108363959
  Shuffled Maps=5
  Failed Shuffles=0
  Merged Map outputs=5
  GC time elapsed (ms)=14139
  CPU time spent (ms)=2947960
  Physical memory (bytes) snapshot=4995964928
  Virtual memory (bytes) snapshot=21269209088
  Total committed heap usage (bytes)=4336386048

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=25190732321
23/09/06 04:03:21 INFO mapreduce.ImportJobBase: Transferred 23.4607 GB in 2,405.0121 seconds (9.989 MB/sec)
23/09/06 04:03:21 INFO mapreduce.ImportJobBase: Retrieved 302089520 records.
23/09/06 04:03:21 WARN mapreduce.LoadIncrementalHFiles: managed connection cannot be used for bulkload. Creating unmanaged connection.
23/09/06 04:03:21 WARN mapreduce.LoadIncrementalHFiles: Skipping non-directory hdfs://ip-172-31-56-92.ec2.internal:8020/user/root/trip_records/_SUCCESS
23/09/06 04:03:22 INFO impl.MetricsConfig: loaded properties from hadoop-metrics2-hbase.properties
23/09/06 04:03:22 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
23/09/06 04:03:22 INFO impl.MetricsSystemImpl: HBase metrics system started
23/09/06 04:03:22 WARN mapreduce.LoadIncrementalHFiles: Trying to bulk load hfile hdfs://ip-172-31-56-92.ec2.internal:8020/user/root/trip_records/colfam1/a2b3203d1b41435c85a812c68021014b wi
th size: 11170159564 bytes can be problematic as it may lead to oversplitting.
23/09/06 04:03:22 WARN mapreduce.LoadIncrementalHFiles: Trying to bulk load hfile hdfs://ip-172-31-56-92.ec2.internal:8020/user/root/trip_records/colfam1/ec32b51f0ff14532b9a7549a08ac732a wi
th size: 11170152021 bytes can be problematic as it may lead to oversplitting.
23/09/06 04:03:22 INFO Configuration.deprecation: hbase.offheapcache.minblocksize is deprecated. Instead, use hbase.blockcache.minblocksize
(root@ip-172-31-56-92 ~) █
```

```
root@ip-172-31-56-92:~#
Current count: 18798000, row: 2017-02-28 21:21:11 2017-02-28 21:30:17
Current count: 18799000, row: 2017-02-28 21:24:14 2017-02-28 21:34:26
Current count: 18800000, row: 2017-02-28 21:27:19 2017-02-28 21:36:44
Current count: 18801000, row: 2017-02-28 21:30:23 2017-02-28 21:35:31
Current count: 18802000, row: 2017-02-28 21:33:20 2017-02-28 21:49:25
Current count: 18803000, row: 2017-02-28 21:36:14 2017-02-28 21:41:41
Current count: 18804000, row: 2017-02-28 21:39:21 2017-03-01 00:00:00
Current count: 18805000, row: 2017-02-28 21:42:24 2017-02-28 21:55:40
Current count: 18806000, row: 2017-02-28 21:45:26 2017-02-28 21:53:44
Current count: 18807000, row: 2017-02-28 21:48:20 2017-02-28 21:59:42
Current count: 18808000, row: 2017-02-28 21:51:17 2017-02-28 22:05:14
Current count: 18809000, row: 2017-02-28 21:54:09 2017-02-28 21:58:28
Current count: 18810000, row: 2017-02-28 21:56:56 2017-02-28 22:03:51
Current count: 18811000, row: 2017-02-28 21:59:44 2017-02-28 22:12:10
Current count: 18812000, row: 2017-02-28 22:02:30 2017-02-28 22:18:15
Current count: 18813000, row: 2017-02-28 22:05:28 2017-02-28 22:12:04
Current count: 18814000, row: 2017-02-28 22:08:24 2017-02-28 22:11:28
Current count: 18815000, row: 2017-02-28 22:11:22 2017-02-28 22:47:21
Current count: 18816000, row: 2017-02-28 22:14:13 2017-02-28 22:17:49
Current count: 18817000, row: 2017-02-28 22:17:10 2017-02-28 22:31:47
Current count: 18818000, row: 2017-02-28 22:20:13 2017-02-28 22:50:51
Current count: 18819000, row: 2017-02-28 22:23:26 2017-02-28 22:31:02
Current count: 18820000, row: 2017-02-28 22:26:41 2017-02-28 22:43:24
Current count: 18821000, row: 2017-02-28 22:30:01 2017-02-28 22:48:46
Current count: 18822000, row: 2017-02-28 22:33:28 2017-02-28 22:45:27
Current count: 18823000, row: 2017-02-28 22:36:59 2017-02-28 22:40:46
Current count: 18824000, row: 2017-02-28 22:40:27 2017-02-28 22:52:51
Current count: 18825000, row: 2017-02-28 22:44:04 2017-02-28 22:47:57
Current count: 18826000, row: 2017-02-28 22:47:43 2017-02-28 23:07:00
Current count: 18827000, row: 2017-02-28 22:51:25 2017-02-28 23:01:58
Current count: 18828000, row: 2017-02-28 22:55:19 2017-02-28 22:58:26
Current count: 18829000, row: 2017-02-28 22:59:09 2017-02-28 23:15:39
Current count: 18830000, row: 2017-02-28 23:03:23 2017-02-28 23:08:50
Current count: 18831000, row: 2017-02-28 23:07:34 2017-02-28 23:15:20
Current count: 18832000, row: 2017-02-28 23:11:45 2017-02-28 23:37:46
Current count: 18833000, row: 2017-02-28 23:15:47 2017-02-28 23:21:11
Current count: 18834000, row: 2017-02-28 23:19:45 2017-02-28 23:35:25
Current count: 18835000, row: 2017-02-28 23:23:52 2017-02-28 23:33:11
Current count: 18836000, row: 2017-02-28 23:28:20 2017-02-28 23:34:52
Current count: 18837000, row: 2017-02-28 23:32:55 2017-02-28 23:37:10
Current count: 18838000, row: 2017-02-28 23:37:54 2017-02-28 23:52:36
Current count: 18839000, row: 2017-02-28 23:42:59 2017-02-28 23:49:30
Current count: 18840000, row: 2017-02-28 23:48:15 2017-02-28 23:55:57
Current count: 18841000, row: 2017-02-28 23:53:47 2017-03-01 00:00:10
Current count: 18842000, row: 2017-02-28 23:59:39 2017-03-01 00:18:04
18842048 row(s) in 635.3760 seconds

-> 18842048
hbase(main) :004:0> █
```

hadoop@ip-172-31-56-92~

```
at org.jruby.javasupport.JavaMethod.invokeStaticDirect(JavaMethod.java:362)
at org.jruby.java.invokers.StaticMethodInvoker.call(StaticMethodInvoker.java:58)
at org.jruby.runtime.callsite.CachingCallSite.cacheAndCall(CachingCallSite.java:312)
at org.jruby.runtime.callsite.CachingCallSite.call(CachingCallSite.java:169)
at usr.lib.hbase.bin.hirb._file_ (/usr/lib/hbase/bin/hirb.rb:129)
at usr.lib.hbase.bin.hirb.load(/usr/lib/hbase/bin/hirb.rb)
at org.jruby.Ruby.runScript(Ruby.java:697)
at org.jruby.Ruby.runScript(Ruby.java:690)
at org.jruby.Ruby.runNormally(Ruby.java:597)
at org.jruby.Ruby.runFromMain(Ruby.java:446)
at org.jruby.Main.doRunFromMain(Main.java:369)
at org.jruby.Main.internalRun(Main.java:258)
at org.jruby.Main.run(Main.java:224)
at org.jruby.Main.run(Main.java:208)
at org.jruby.Main.main(Main.java:188)
log4j:ERROR Either File or DatePattern options are not set for appender [DRFAS].
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

hbase(main):001:0> list
TABLE
trip_records_hbase
1 row(s) in 0.2490 seconds

=> ["trip_records_hbase"]
hbase(main):002:0> describe 'trip_records_hbase'
Table trip_records_hbase is ENABLED
trip_records_hbase
COLUMN FAMILIES DESCRIPTION
(NAME => 'colfam1', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_V
ERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0')
1 row(s) in 0.1100 seconds

hbase(main):003:0> exit
```