Scripts Execution

Explanation of the solution to the batch layer problem

1. For the below mentioned tasks, an EMR cluster with **Hadoop, Sqoop, Hive, HBase and Spark,** Root device EBS volume size **as** 20 GB was created.

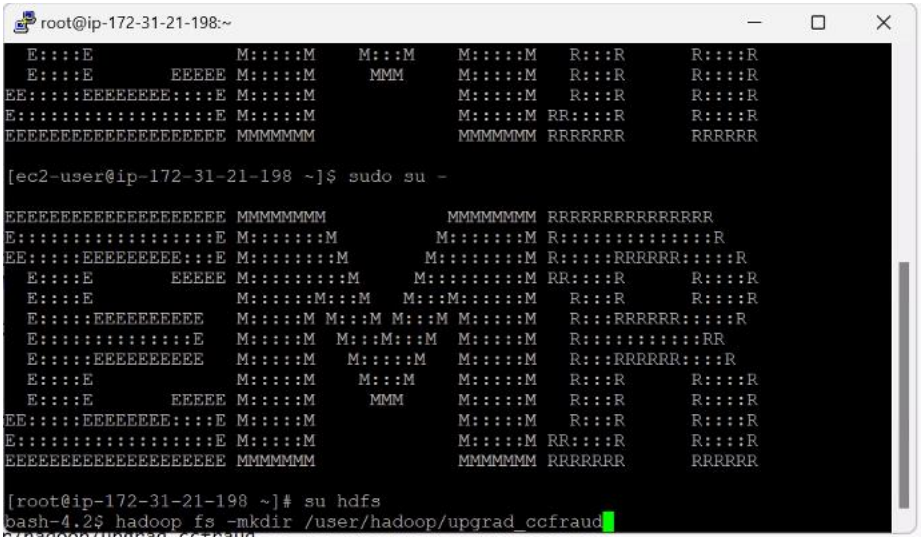
    - **Task 1**: Loading the historical transaction data (card_transactions.csv) in a NoSQL database.
    - **Task 2**: Ingesting the required data from AWS RDS to Hadoop.
    - **Task 3**: Creating a look-up table with the required columns as per the problem statement.
    - **Task 4**: After creation of the table, loading of the relevant data into the lookup table.

2. Logging into EMR instance as “ec2-user” through Putty - Gen

3. Switching to root user and then to HDFS user.

    - Create directory and change its ownership -> First, exit from HDFS user -> Exit from root user back to ec2-user.

      sudo su –
      su – hdfs
      hadoop fs -mkdir /user/hadoop/upgrad_ccfraud
      hadoop fs -chown ec2-user:ec2-user /user/hadoop/upgrad_ccfraud



4. Uploading the historical transactions file, card_transacations.csv file to ec2 instance.

5. Creating a directory in HDFS and copy card_transactions.csv to that location.

    hadoop fs -mkdir /user/hadoop/upgrad_ccfraud/card_transactions
    hadoop fs -put card_transactions.csv /user/hadoop/upgrad_ccfraud/card_transactions/

The initial steps for setting up the project is complete.

Task 1 :Loading Historical Transactions Data into NoSQL Database

1. Starting hive session and creating a new database named ccfraud_capstone -> switch to ccfraud_capstone database

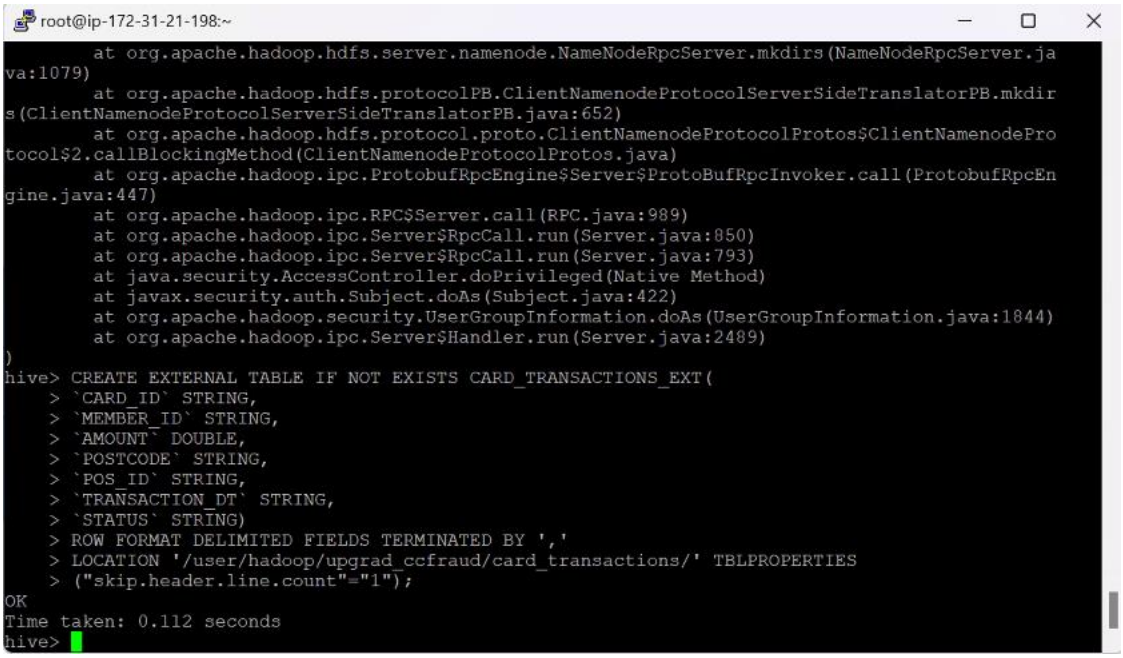
   create database ccfraud_capstone;
   use ccfraud_capstone;

2. Setting parameters for the hive session

   set hive.auto.convert.join=false;
   set hive.stats.autogather=true;
   set orc.compress=SNAPPY;
   set hive.exec.compress.output=true;
   set mapred.output.compression.codec=org.apache.hadoop.io.compress.SnappyCodec; set mapred.output.compression.type=BLOCK;
   set mapreduce.map.java.opts=-Xmx5G; set mapreduce.reduce.java.opts=-Xmx5G;
   set mapred.child.java.opts=-Xmx5G -XX:+UseConcMarkSweepGC -XX:-UseGCOverheadLimit;

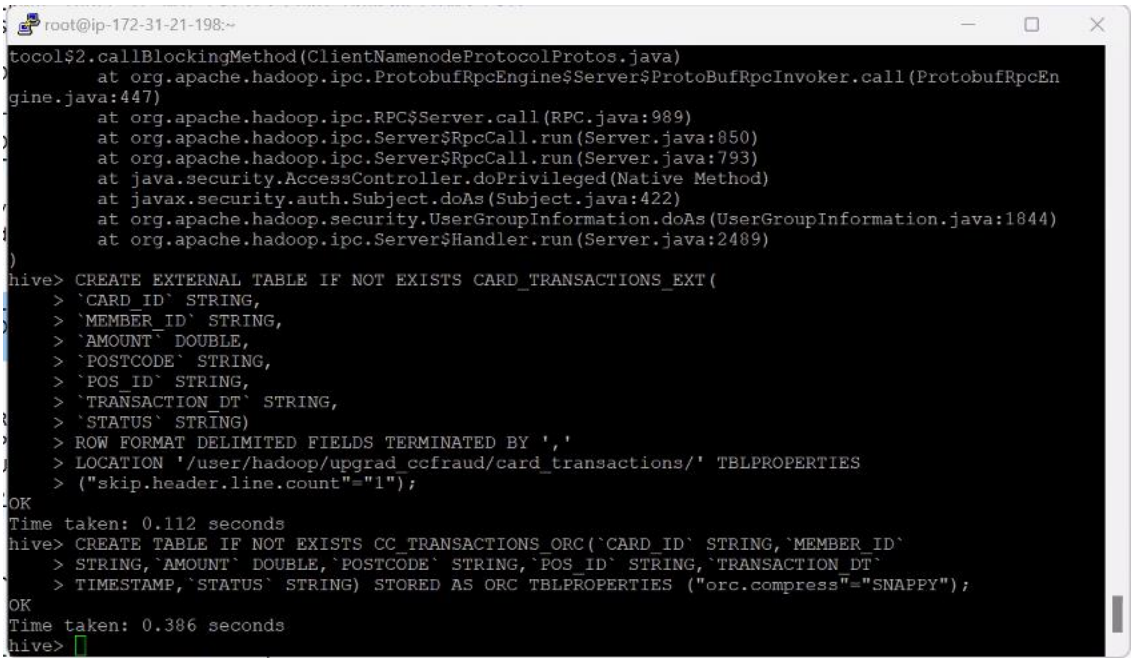
3. Creating an external table CARD_TRANSACTIONS_EXT

   CREATE EXTERNAL TABLE IF NOT EXISTS CARD_TRANSACTIONS_EXT(
   `CARD_ID` STRING,
   `MEMBER_ID` STRING,
   `AMOUNT` DOUBLE,
   `POSTCODE` STRING,
   `POS_ID` STRING,
   `TRANSACTION_DT` STRING,
   `STATUS` STRING)
   ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
   LOCATION '/user/hadoop/upgrad_ccfraud/card_transactions/' TBLPROPERTIES ("skip.header.line.count"="1");



4. Creating table " **CC_TRANSACTIONS_ORC**" in ORC format for better performance.

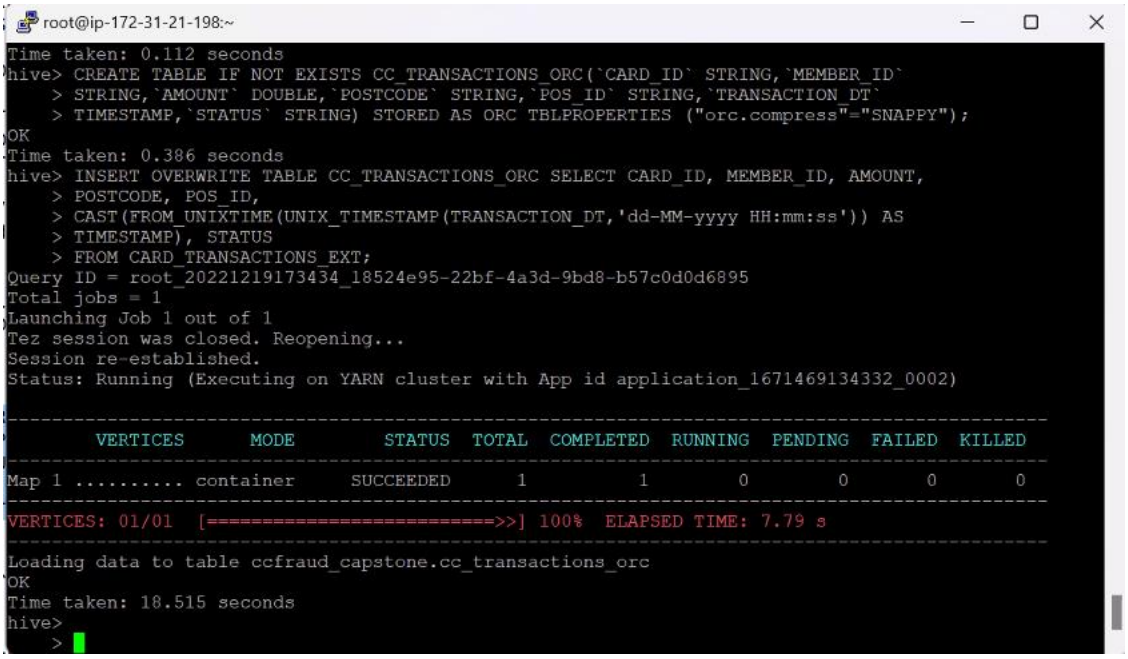
CREATE TABLE IF NOT EXISTS CC_TRANSACTIONS_ORC(`CARD_ID` STRING,`MEMBER_ID` STRING,`AMOUNT` DOUBLE,`POSTCODE` STRING,`POS_ID` STRING,`TRANSACTION_DT` TIMESTAMP,`STATUS` STRING) STORED AS ORC TBLPROPERTIES ("orc.compress"="SNAPPY");



5. Loading the data in "**CC_TRANSACTIONS_ORC**" table and type casting **transaction_dt** column to timestampformat

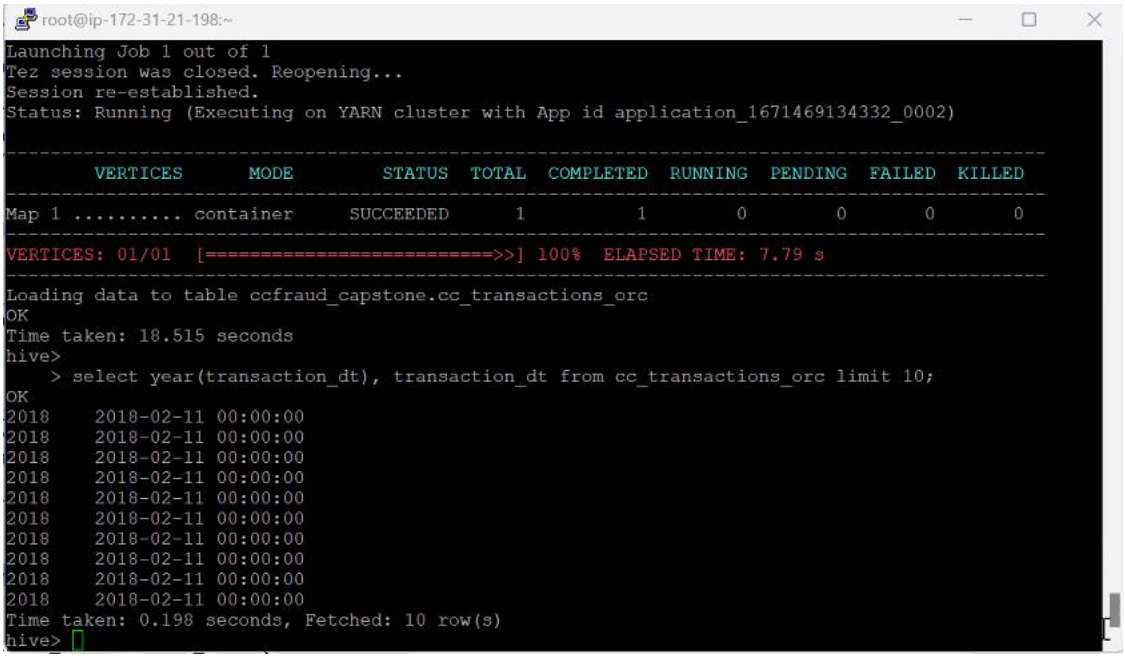
INSERT OVERWRITE TABLE CC_TRANSACTIONS_ORC SELECT CARD_ID, MEMBER_ID, AMOUNT, POSTCODE, POS_ID,
CAST(FROM_UNIXTIME(UNIX_TIMESTAMP(TRANSACTION_DT,'dd-MM-yyyy HH:mm:ss')) AS

TIMESTAMP), STATUS
FROM CARD_TRANSACTIONS_EXT;



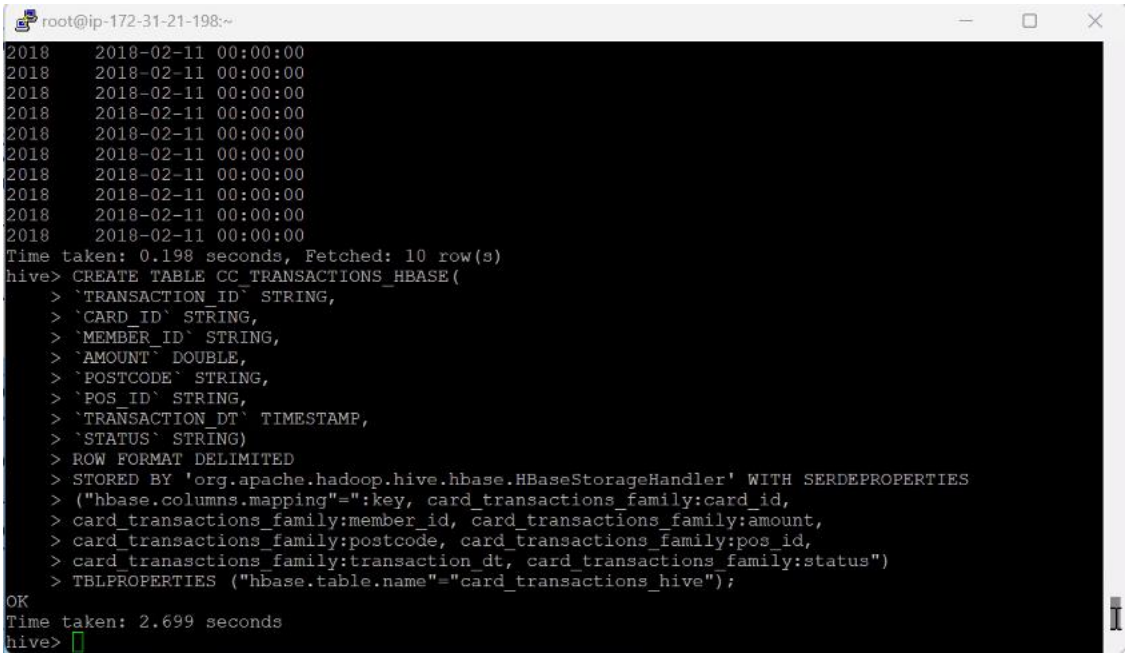
6. Verifying **transaction_dt** and **year** columns in " **CC_TRANSACTIONS_ORC** " table.

select year(transaction_dt), transaction_dt from cc_transactions_orc limit 10;



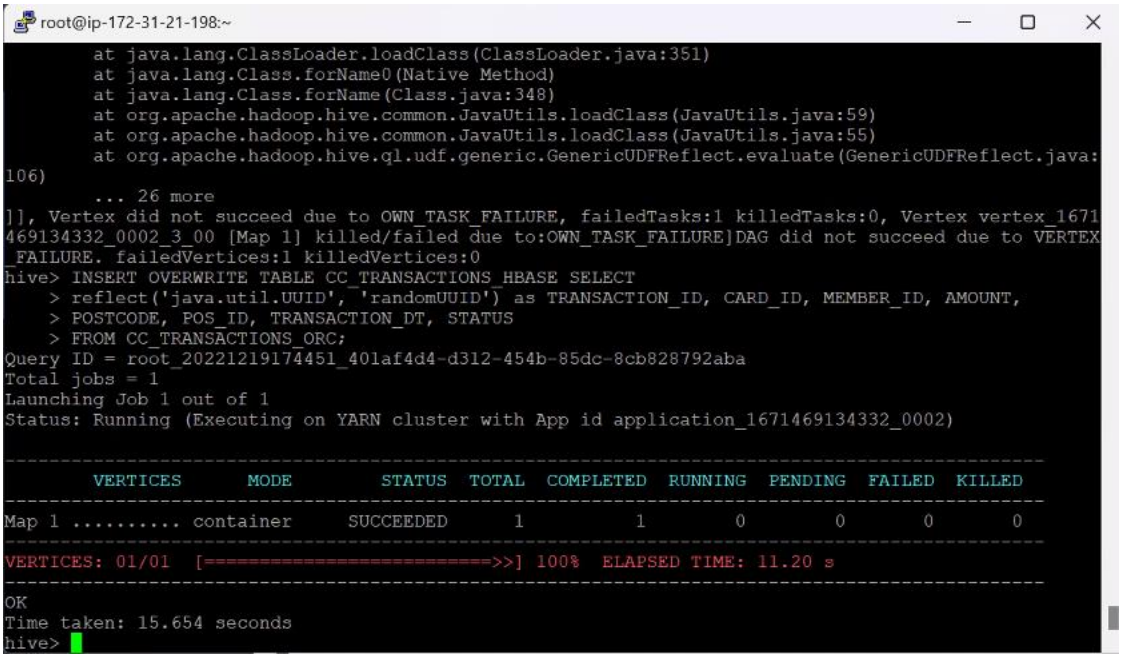
7. Creating an integrated hive - hbase table that will be visible in HBase as well." **CC_TRANSACTIONS_HBASE** " table

```
CREATE TABLE CC_TRANSACTIONS_HBASE(
`TRANSACTION_ID` STRING,
`CARD_ID` STRING,
`MEMBER_ID` STRING,
`AMOUNT` DOUBLE,
`POSTCODE` STRING,
`POS_ID` STRING,
`TRANSACTION_DT` TIMESTAMP,
`STATUS` STRING)
ROW FORMAT DELIMITED
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES
("hbase.columns.mapping"=":key, card_transactions_family:card_id,
card_transactions_family:member_id, card_transactions_family:amount,
card_transactions_family:postcode, card_transactions_family:pos_id,
card_tranasctions_family:transaction_dt, card_transactions_family:status")
TBLPROPERTIES ("hbase.table.name"="cc_transactions_hive");
```



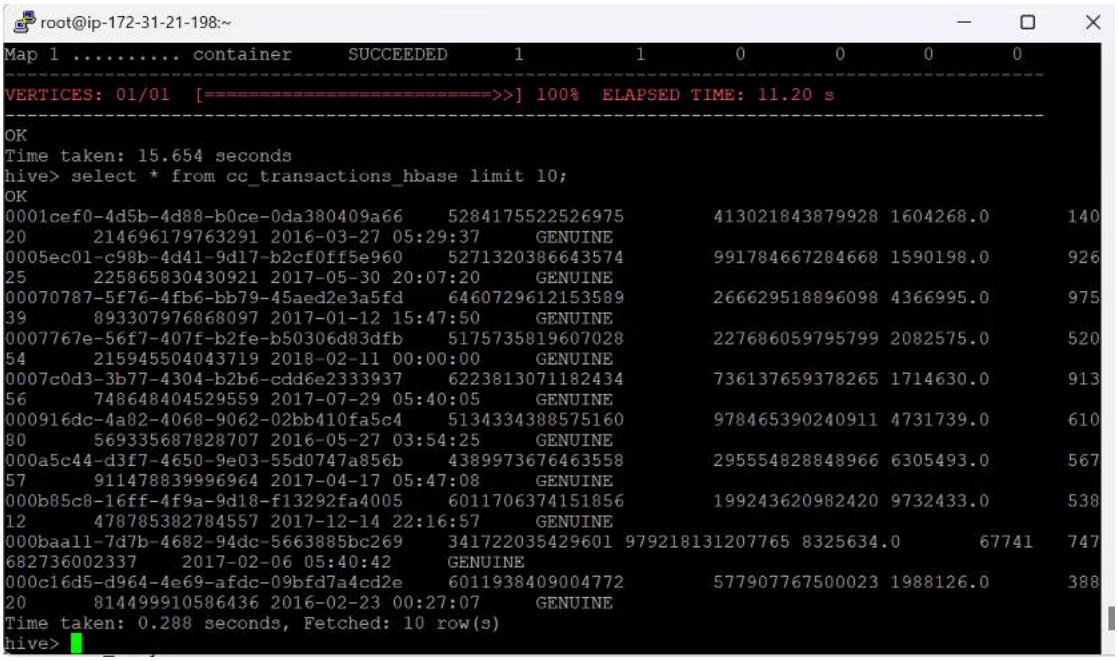
8. Loading data in "**CC_TRANSACTIONS_HBASE**" table that will be visible in HBase as well with table name as "**cc_transactions_hive**". Using randomUUID to populate TRANSACTION_ID field (row key).

```
INSERT OVERWRITE TABLE CC_TRANSACTIONS_HBASE SELECT
reflect('java.util.UUID', 'randomUUID') as TRANSACTION_ID, CARD_ID, MEMBER_ID, AMOUNT,
POSTCODE, POS_ID, TRANSACTION_DT, STATUS
FROM CC_TRANSACTIONS_ORC
```



9. Verify data in "**cc_transactions_hbase**" table.

select * from cc_transactions_hbase limit 10;

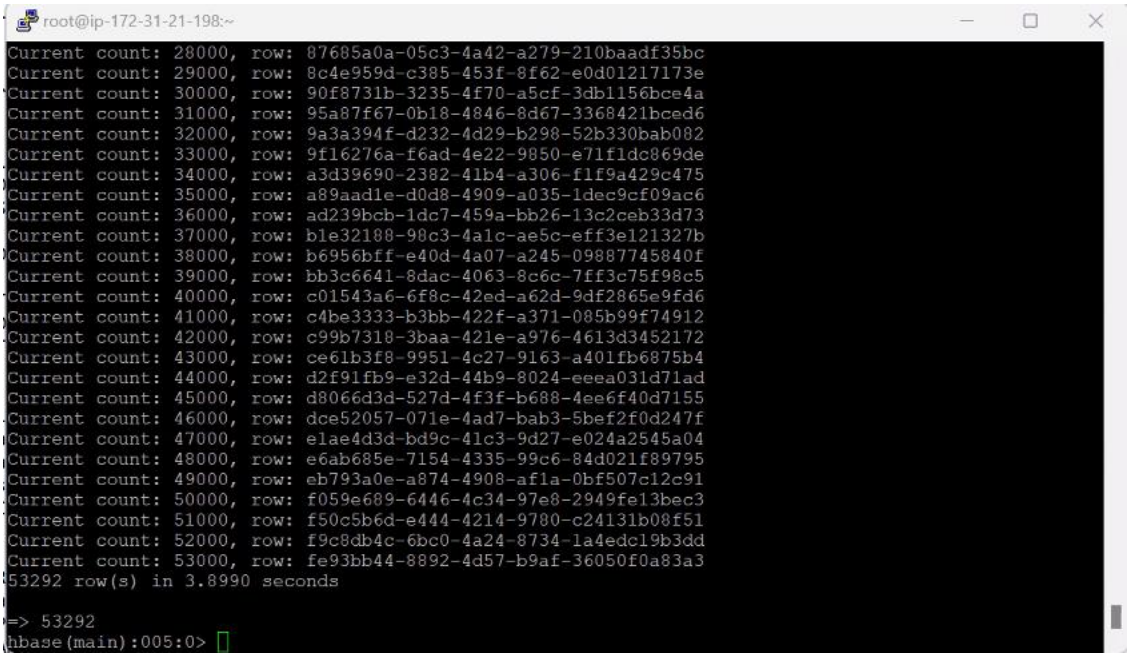


10. Starting HBase session and verifying details of "**cc_transactions_hive**" table (hive-hbase integrated table).

describe 'cc_transactions_hive'

11. Verify count of "**cc_transactions_hive**" table

count 'cc_transactions_hive'

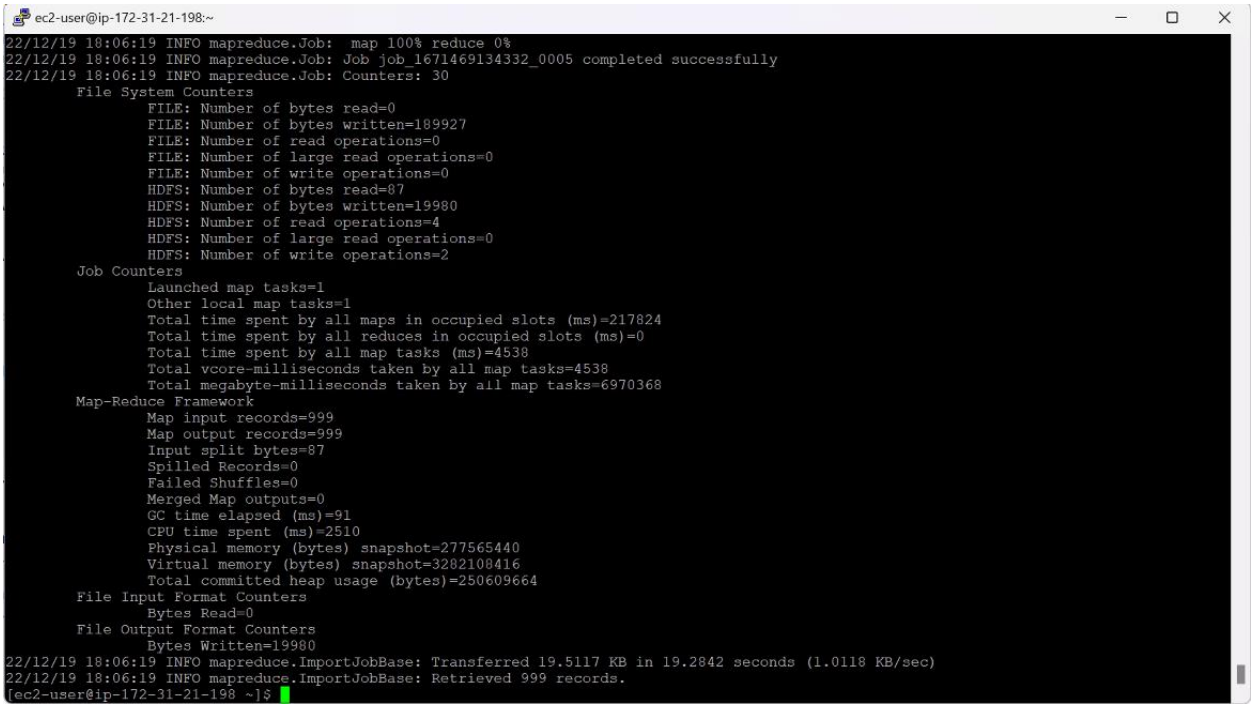


*Count of the "**cc_transactions_hive**" table is 53,292 which is matching with given requirement*

Task 2: Ingesting the required data from AWS RDS to Hadoop.

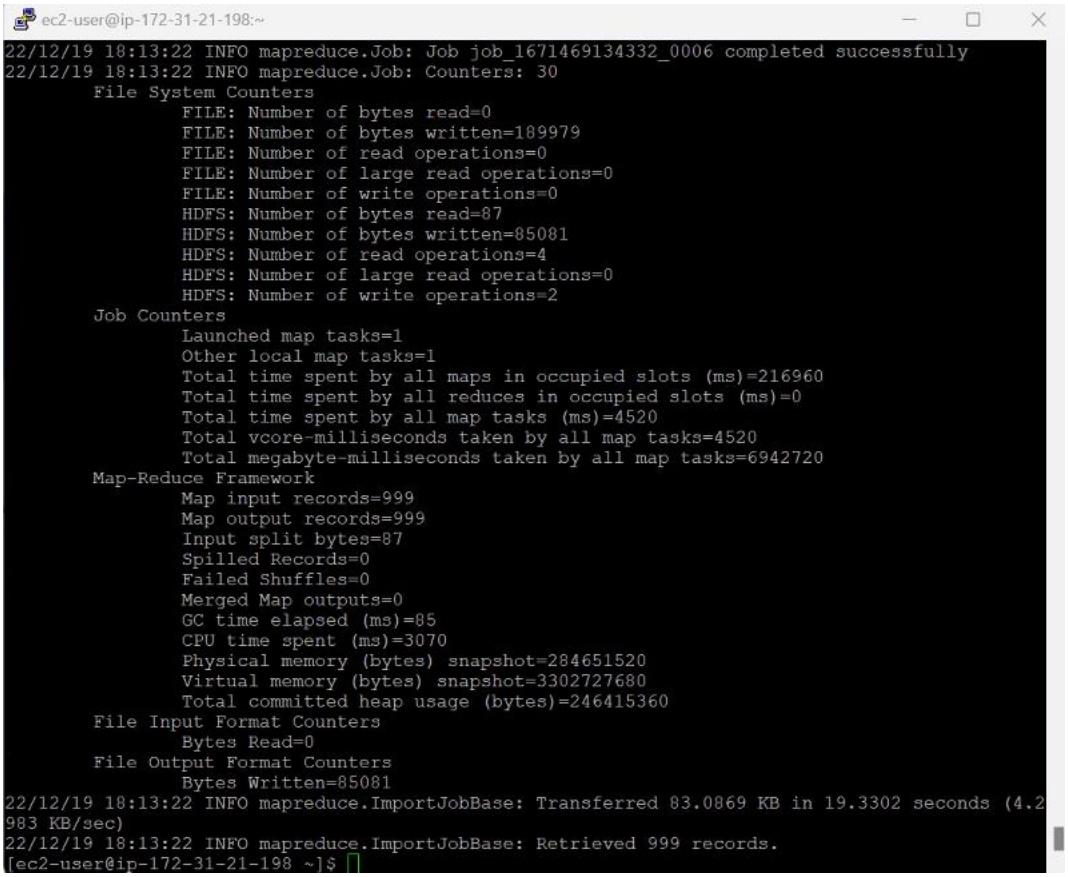
1. Running Sqoop command to import "member_score" table from RDS to HDFS.

**sqoop import --connect jdbc:mysql://upgradawsrds1.cyaielc9bmnf.us-east-1.rds.amazonaws.com/cred_financials_data **
**--username upgraduser **
**--password upgraduser **
**--table member_score **
**--null-string 'NA' **
**--null-non-string '\\N' **
**--delete-target-dir **
**--target-dir '/user/hadoop/upgrad_ccfraud/card_transactions/member_score' **
-m 1



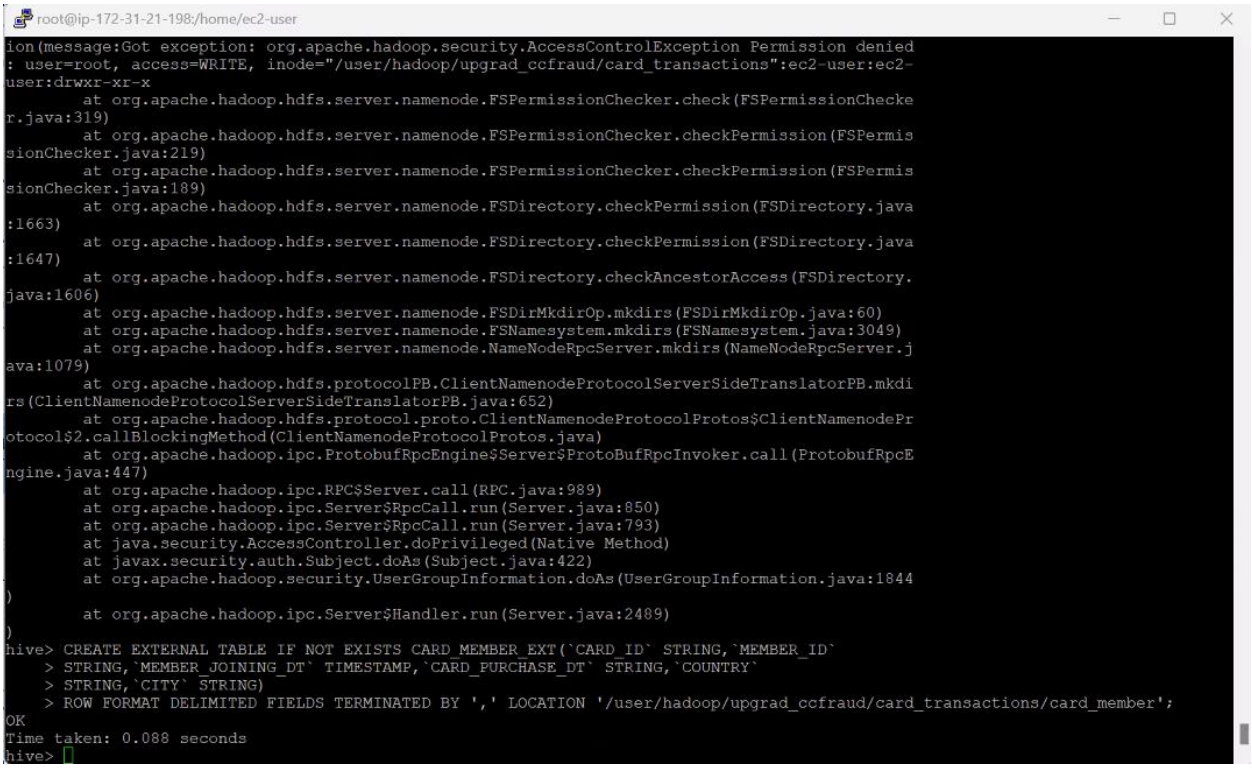
2. Running Sqoop command to import "**card_member**" table from RDS to HDFS.

```
sqoop import --connect jdbc:mysql://upgradawsrds1.cyaielc9bmnf.us-east-
1.rds.amazonaws.com/cred_financials_data \
--username upgraduser \
--password upgraduser \
--table card_member \
--null-string 'NA' \
--null-non-string '\\N' \
--delete-target-dir \
--target-dir '/user/hadoop/upgrad_ccfraud/card_transactions/card_member' \
-m 1
```



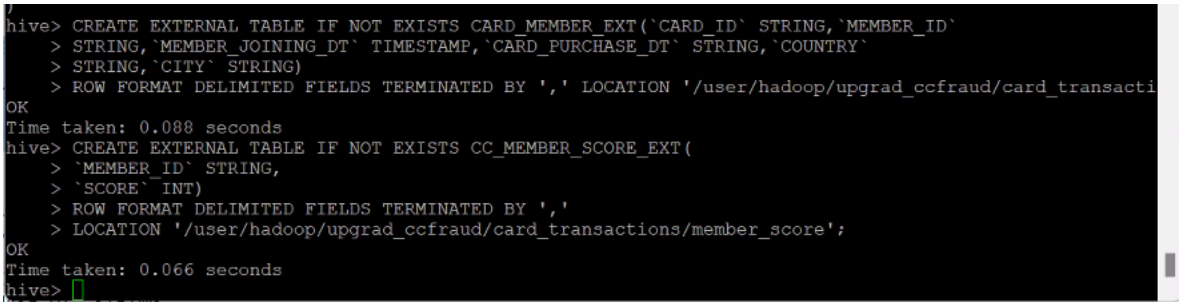
3. Starting hive session and creating an external table "**card_member_ext**" to hold data from card_member table in RDS.

CREATE EXTERNAL TABLE IF NOT EXISTS CARD_MEMBER_EXT(`CARD_ID` STRING,`MEMBER_ID` STRING,`MEMBER_JOINING_DT` TIMESTAMP,`CARD_PURCHASE_DT` STRING,`COUNTRY` STRING,`CITY` STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION '/user/hadoop/upgrad_ccfraud/card_transactions/card_member';



4. Create external table "**CC_MEMBER_SCORE_EXT** " to hold data from member_score table in RDS.

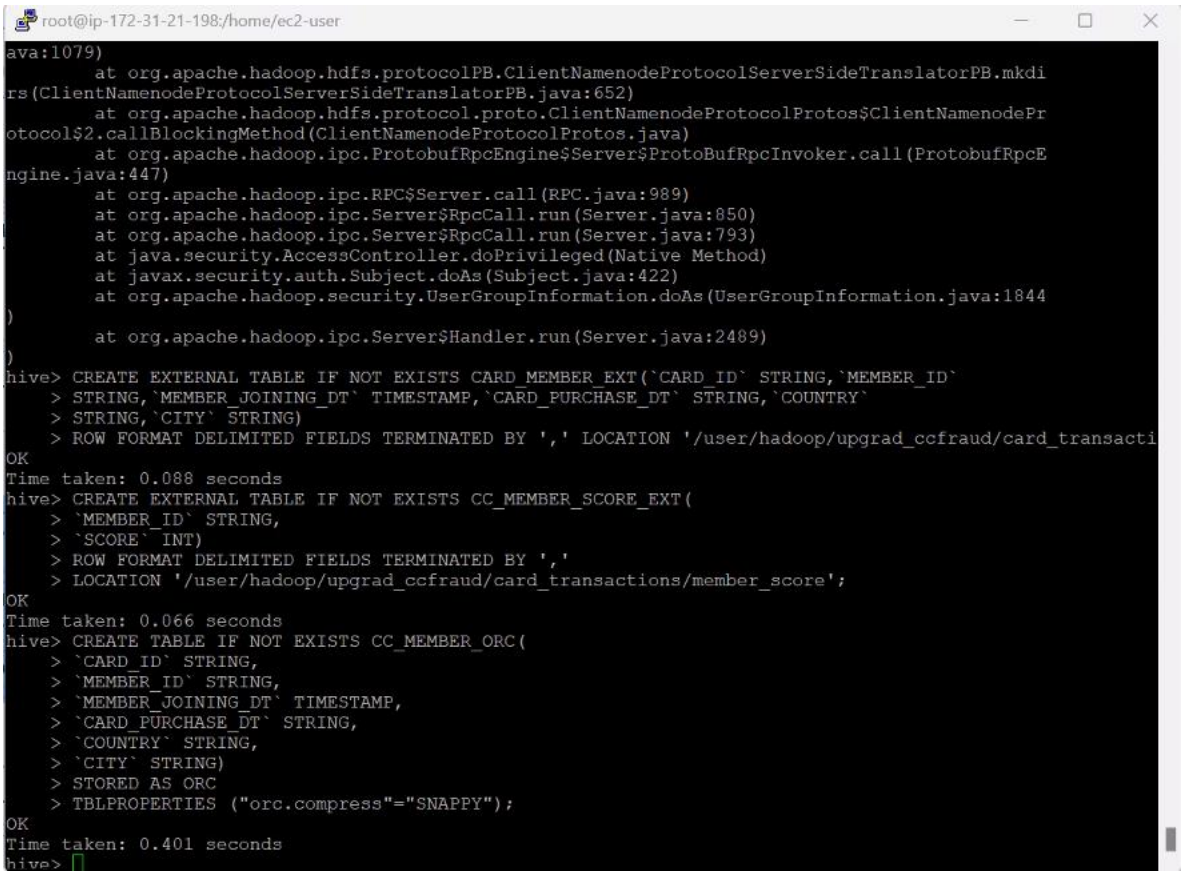
CREATE EXTERNAL TABLE IF NOT EXISTS CC_MEMBER_SCORE_EXT(
`MEMBER_ID` STRING,
`SCORE` INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/user/hadoop/upgrad_ccfraud/card_transactions/member_score';



5. Create " **CC_MEMBER_ORC**" table with ORC for better performance.

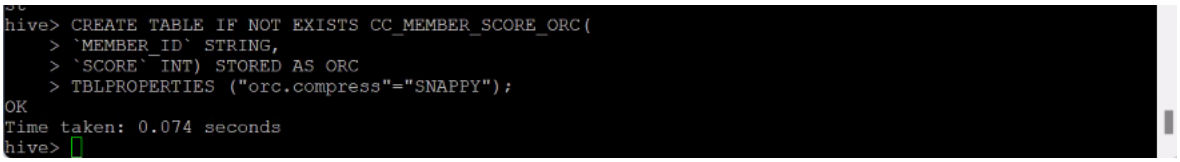
CREATE TABLE IF NOT EXISTS CC_MEMBER_ORC(
`CARD_ID` STRING,
`MEMBER_ID` STRING,
`MEMBER_JOINING_DT` TIMESTAMP,
`CARD_PURCHASE_DT` STRING,
`COUNTRY` STRING,
`CITY` STRING)

STORED AS ORC
TBLPROPERTIES ("orc.compress"="SNAPPY");



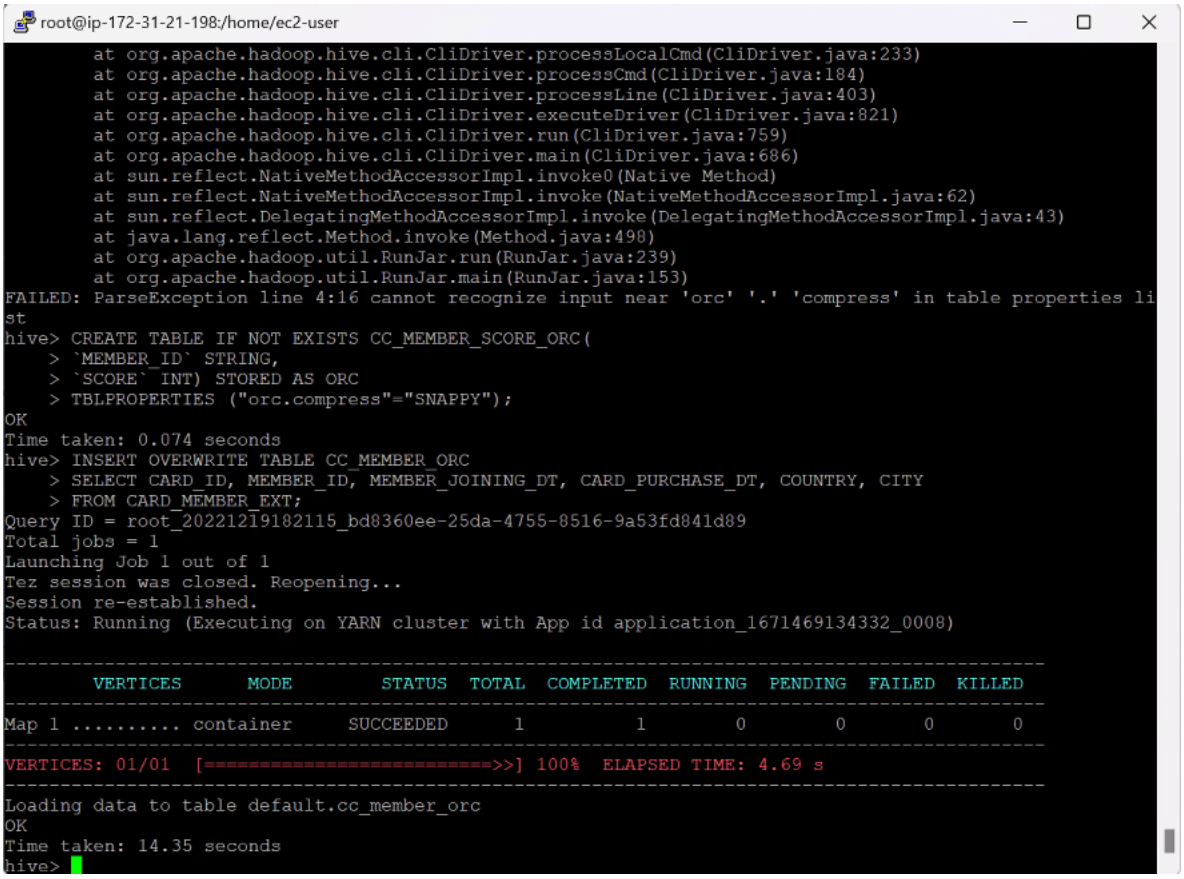
6. Create " **CC_MEMBER_SCORE_ORC** " table with ORC for better performance.

CREATE TABLE IF NOT EXISTS CC_MEMBER_SCORE_ORC(
`MEMBER_ID` STRING,
`SCORE` INT) STORED AS ORC
TBLPROPERTIES ("orc.compress"="SNAPPY");



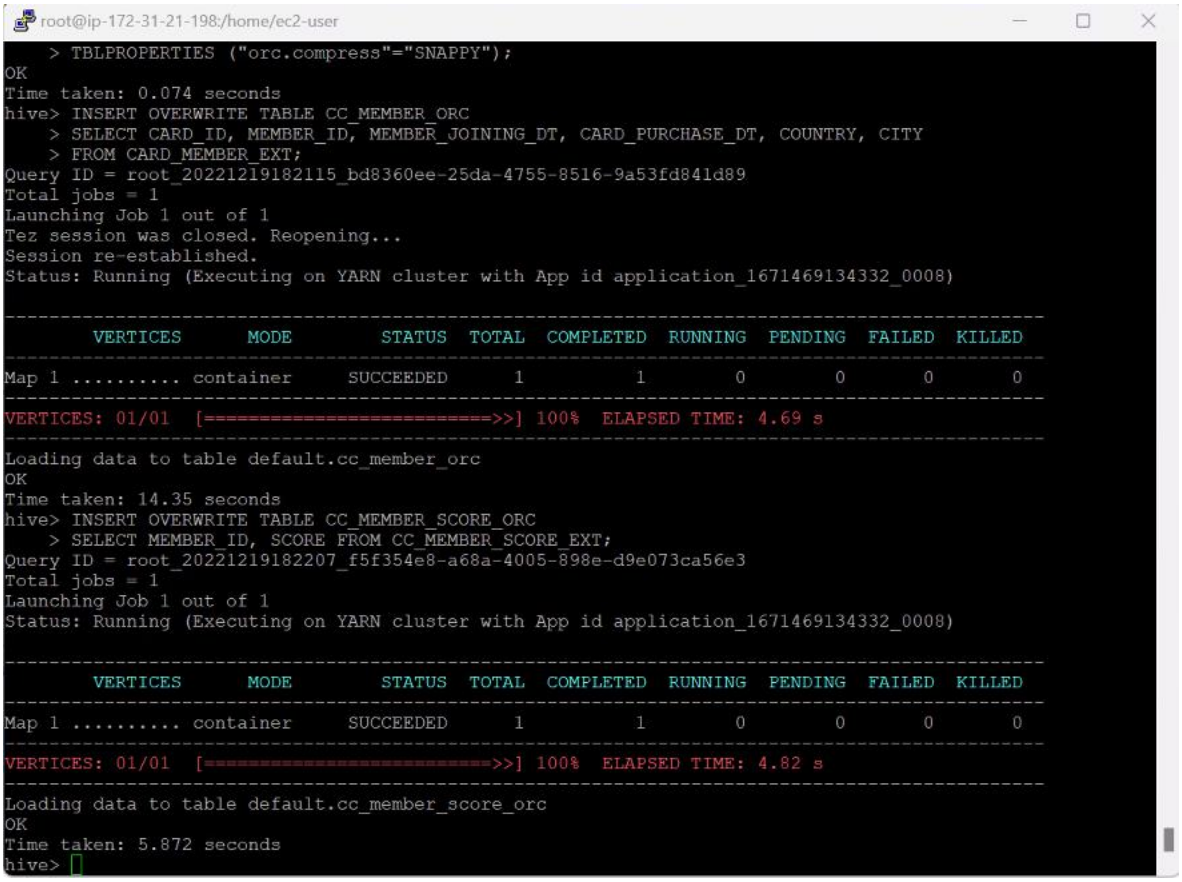
7. Load data into " **CC_MEMBER_ORC** " table from "**card_member_ext**" table.

INSERT OVERWRITE TABLE CC_MEMBER_ORC
SELECT CARD_ID, MEMBER_ID, MEMBER_JOINING_DT, CARD_PURCHASE_DT, COUNTRY, CITY
FROM CARD_MEMBER_EXT;



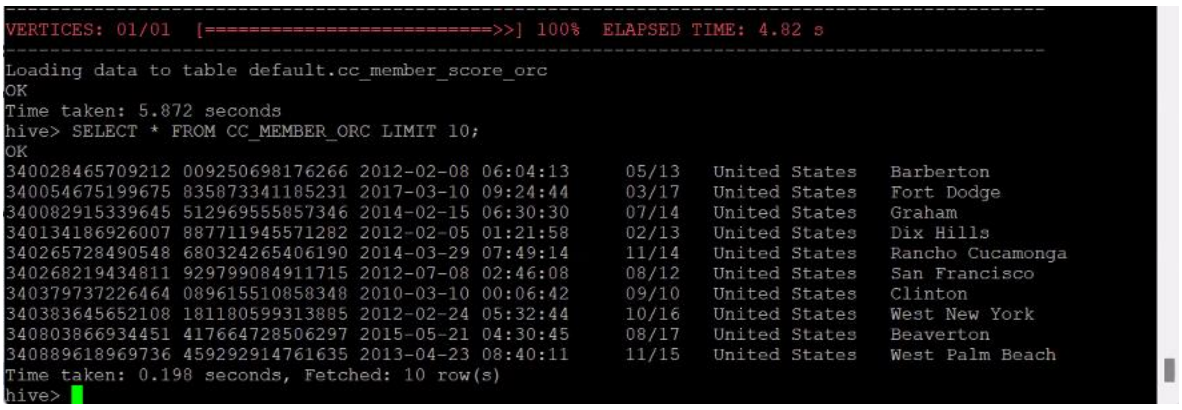
8. Load data into "**CC_MEMBER_SCORE_ORC**" table from "**CC_MEMBER_SCORE_EXT**" table.

INSERT OVERWRITE TABLE CC_MEMBER_SCORE_ORC
SELECT MEMBER_ID, SCORE FROM CC_MEMBER_SCORE_EXT;



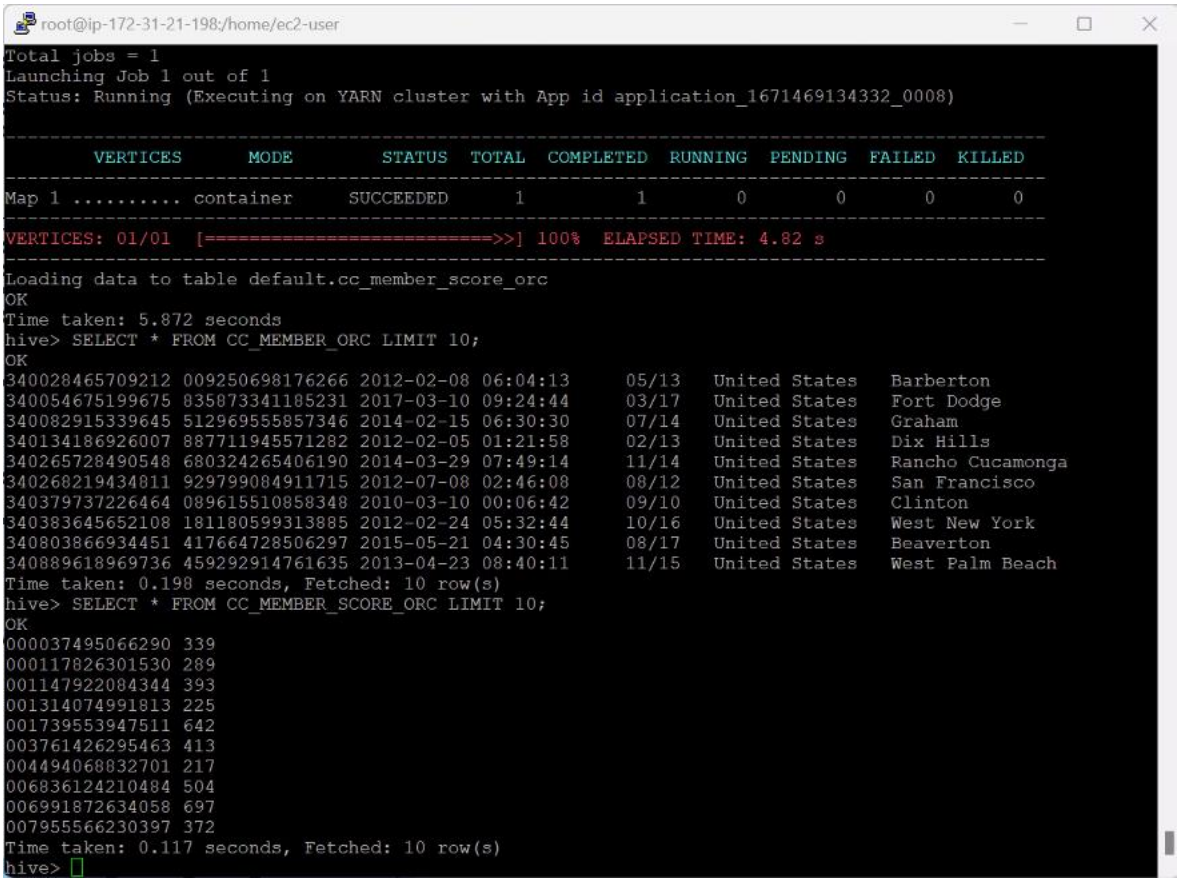
9. Verify data in "**CC_MEMBER_ORC** " table.

SELECT * FROM CC_MEMBER_ORC LIMIT 10;



10. Verify data in "**CC_MEMBER_SCORE_ORC**" table.

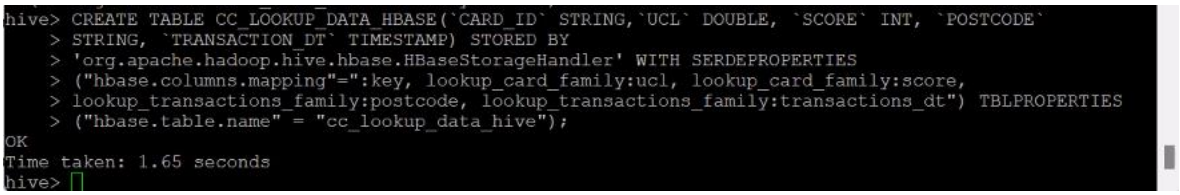
SELECT * FROM CC_MEMBER_SCORE_ORC LIMIT 10;



Task 3: Creating a look-up table with the required columns as per the problem statement.

1. Creating “CC_lookup_data_hbase” table (hive-hbase integrated table) which will be visible in HBase(lookup_data_hive).

   CREATE TABLE CC_LOOKUP_DATA_HBASE(`CARD_ID` STRING,`UCL` DOUBLE, `SCORE` INT, `POSTCODE`
   STRING, `TRANSACTION_DT` TIMESTAMP) STORED BY
   'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES
   ("hbase.columns.mapping"=":key, lookup_card_family:ucl,
   lookup_card_family:score,lookup_transactions_family:postcode,
   lookup_transactions_family:transactions_dt") TBLPROPERTIES
   ("hbase.table.name" = "cc_lookup_data_hive");



2. Verifying details of **cc_lookup_data_hive** (hive-hbase integrated) table:

   describe 'cc_lookup_data_hive'

```
hive> CREATE TABLE CC_LOOKUP_DATA_HBASE(`CARD_ID` STRING,`UCL` DOUBLE, `SCORE` INT, `POSTCODE`
    > STRING, `TRANSACTION_DT` TIMESTAMP) STORED BY
    > 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES
    > ("hbase.columns.mapping"=":key, lookup_card_family:ucl, lookup_card_family:score,
    > lookup_transactions_family:postcode, lookup_transactions_family:transactions_dt") TBLPROPERTIES
    > ("hbase.table.name" = "cc_lookup_data_hive");
OK
Time taken: 1.65 seconds
hive> exit;
[root@ip-172-31-21-198 ec2-user]# 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

hbase(main):001:0> describe 'cc_lookup_data_hive'
Table cc_lookup_data_hive is ENABLED
cc_lookup_data_hive
COLUMN FAMILIES DESCRIPTION
{NAME => 'lookup_card_family', 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'}
{NAME => 'lookup_transactions_family', 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'}
2 row(s) in 0.3090 seconds

hbase(main):002:0>
```

3. Altering "**cc_lookup_data_hive**" table and setting VERSIONS to 10 for lookup_transaction_family since as per the problem statement we have to store last 10 transactions in lookup table.

   alter 'cc_lookup_data_hive', {NAME => 'lookup_transactions_family', VERSIONS => 10}

```
hbase(main):001:0> describe 'cc_lookup_data_hive'
Table cc_lookup_data_hive is ENABLED
cc_lookup_data_hive
COLUMN FAMILIES DESCRIPTION
{NAME => 'lookup_card_family', 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'}
{NAME => 'lookup_transactions_family', 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'}
2 row(s) in 0.3090 seconds

hbase(main):002:0> alter 'cc_lookup_data_hive', {NAME => 'lookup_transactions_family', VERSIONS => 10}
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 2.4090 seconds

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

4. Verify details of "cc_lookup_data_hive" (hive-hbase integrated) table after altering version to 10 : **describe 'cc_lookup_data_hive'**

```
[root@ip-172-31-21-198 ec2-user]# describe 'cc_lookup_data_hive'
bash: describe: command not found
[root@ip-172-31-21-198 ec2-user]# 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

hbase(main):001:0> describe 'cc_lookup_data_hive'
Table cc_lookup_data_hive is ENABLED
cc_lookup_data_hive
COLUMN FAMILIES DESCRIPTION
{NAME => 'lookup_card_family', 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'}
{NAME => 'lookup_transactions_family', BLOOMFILTER => 'ROW', VERSIONS => '10', IN_MEMORY => 'false', KEE
P_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_
VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
2 row(s) in 0.2970 seconds

hbase(main):002:0>
```

Task 4: After creation of the table, loading of the relevant data into the lookup table.

1. Starting hive session and creating table "**ranked_cc_transactions_orc**" to store last 10 transactions for eachcard_id. for better performance.

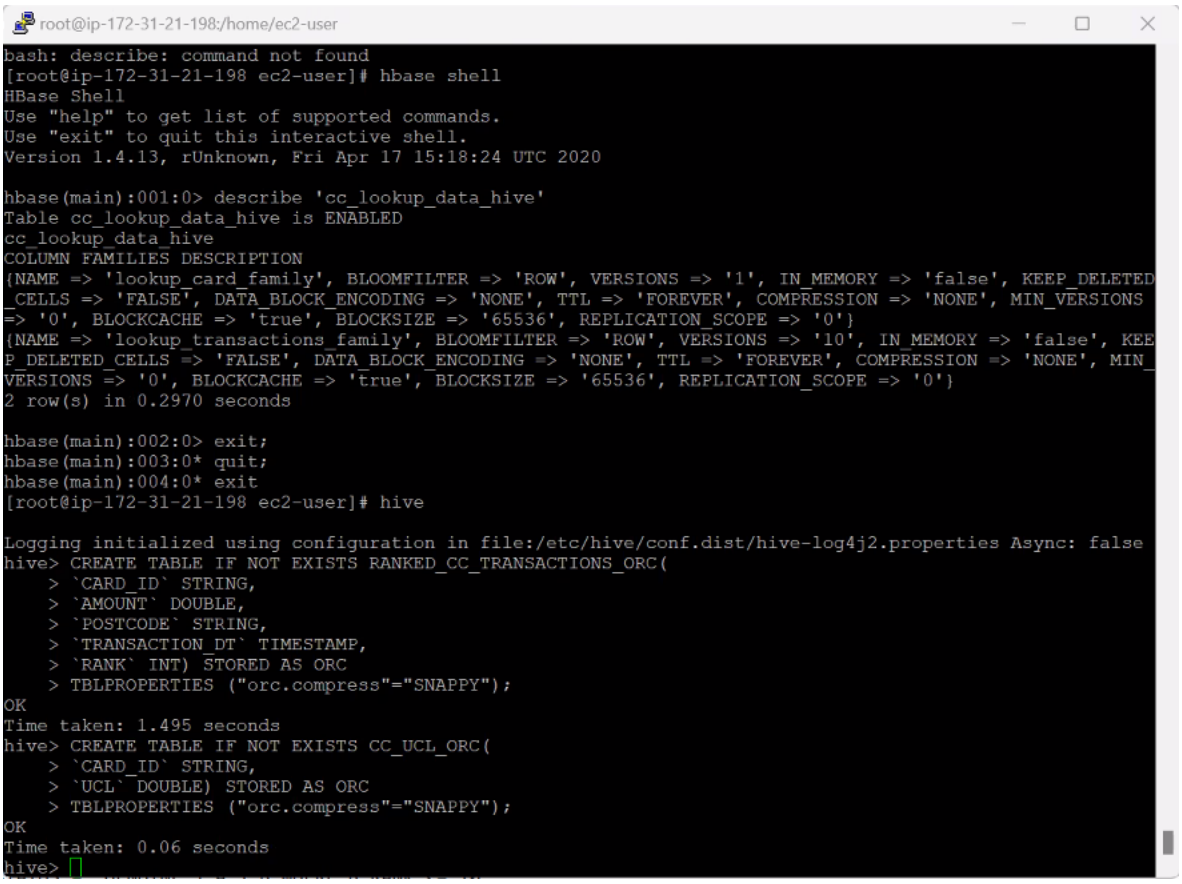
CREATE TABLE IF NOT EXISTS RANKED_CC_TRANSACTIONS_ORC(
`CARD_ID` STRING,
`AMOUNT` DOUBLE,
`POSTCODE` STRING,
`TRANSACTION_DT` TIMESTAMP,
`RANK` INT) STORED AS ORC
TBLPROPERTIES ("orc.compress"="SNAPPY");

```
hbase(main):001:0> describe 'cc_lookup_data_hive'
Table cc_lookup_data_hive is ENABLED
cc_lookup_data_hive
COLUMN FAMILIES DESCRIPTION
{NAME => 'lookup_card_family', 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'}
{NAME => 'lookup_transactions_family', BLOOMFILTER => 'ROW', VERSIONS => '10', IN_MEMORY => 'false', KEE
P_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_
VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
2 row(s) in 0.2970 seconds

hbase(main):002:0> exit;
hbase(main):003:0* quit;
hbase(main):004:0* exit
[root@ip-172-31-21-198 ec2-user]# hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false
hive> CREATE TABLE IF NOT EXISTS RANKED_CC_TRANSACTIONS_ORC(
    > `CARD_ID` STRING,
    > `AMOUNT` DOUBLE,
    > `POSTCODE` STRING,
    > `TRANSACTION_DT` TIMESTAMP,
    > `RANK` INT) STORED AS ORC
    > TBLPROPERTIES ("orc.compress"="SNAPPY");
OK
Time taken: 1.495 seconds
hive>
```

2. Create table "**cc_ucl_orc**" to store UCL values for each card_id. For better performance.

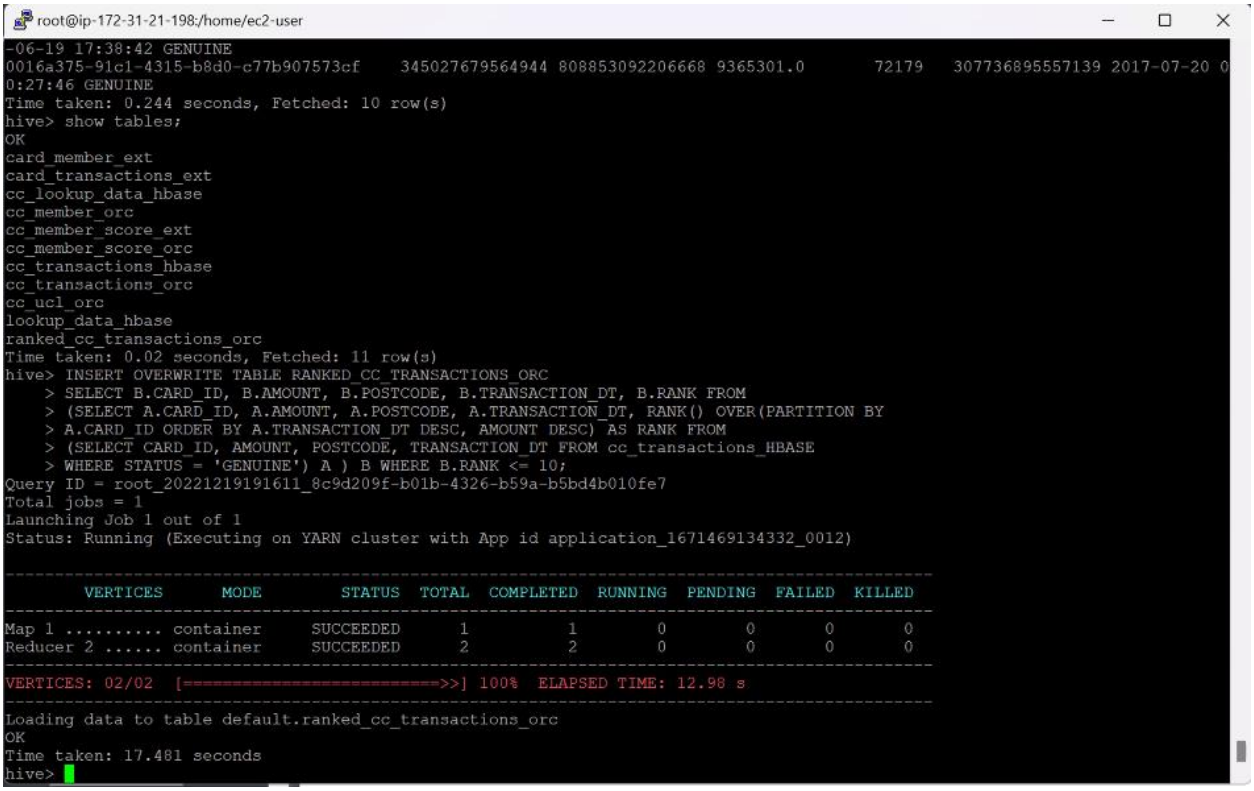
CREATE TABLE IF NOT EXISTS CC_UCL_ORC(
`CARD_ID` STRING,

`UCL` DOUBLE) STORED AS ORC
TBLPROPERTIES ("orc.compress"="SNAPPY");



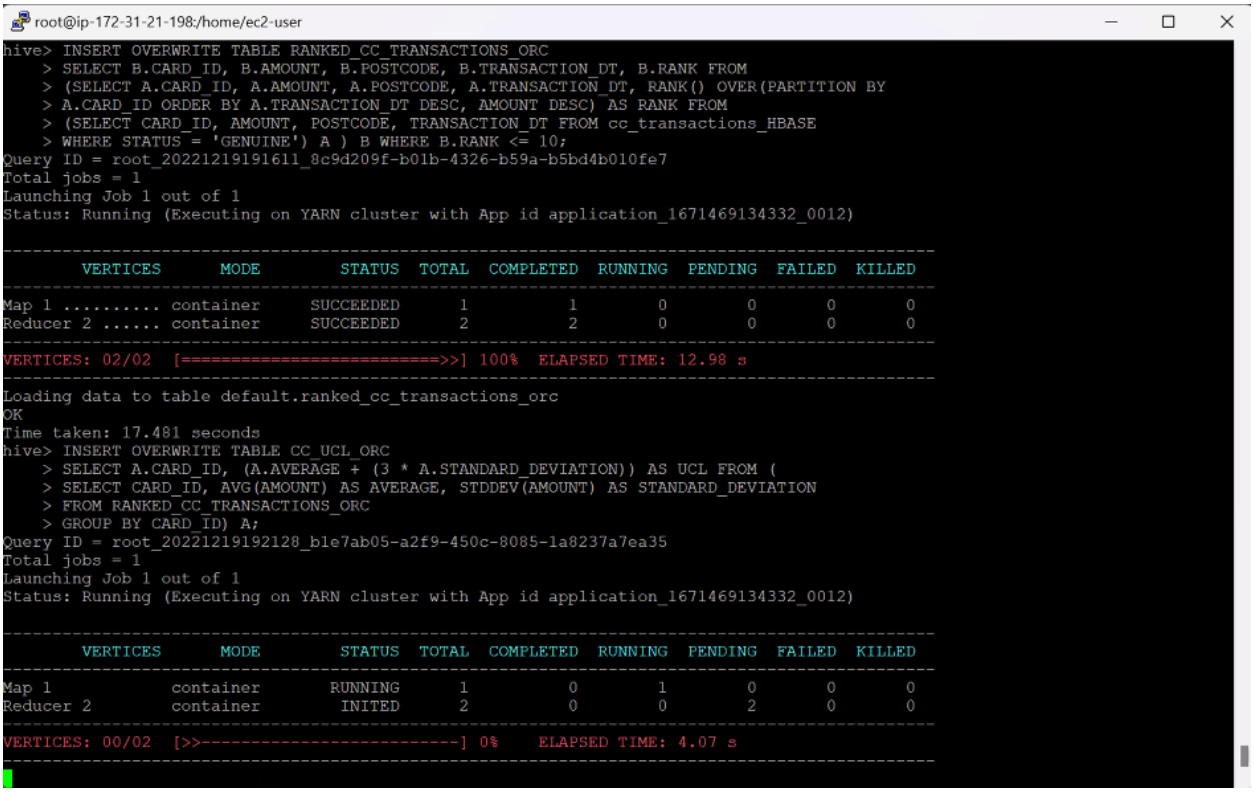
3. Loading data in "**ranked_cc_transactions_orc**" table

INSERT OVERWRITE TABLE RANKED_CC_TRANSACTIONS_ORC
SELECT B.CARD_ID, B.AMOUNT, B.POSTCODE, B.TRANSACTION_DT, B.RANK FROM
(SELECT A.CARD_ID, A.AMOUNT, A.POSTCODE, A.TRANSACTION_DT, RANK() OVER(PARTITION BY A.CARD_ID ORDER BY A.TRANSACTION_DT DESC, AMOUNT DESC) AS RANK FROM
(SELECT CARD_ID, AMOUNT, POSTCODE, TRANSACTION_DT FROM cc_transactions_HBASE WHERE STATUS = 'GENUINE') A) B WHERE B.RANK <= 10;



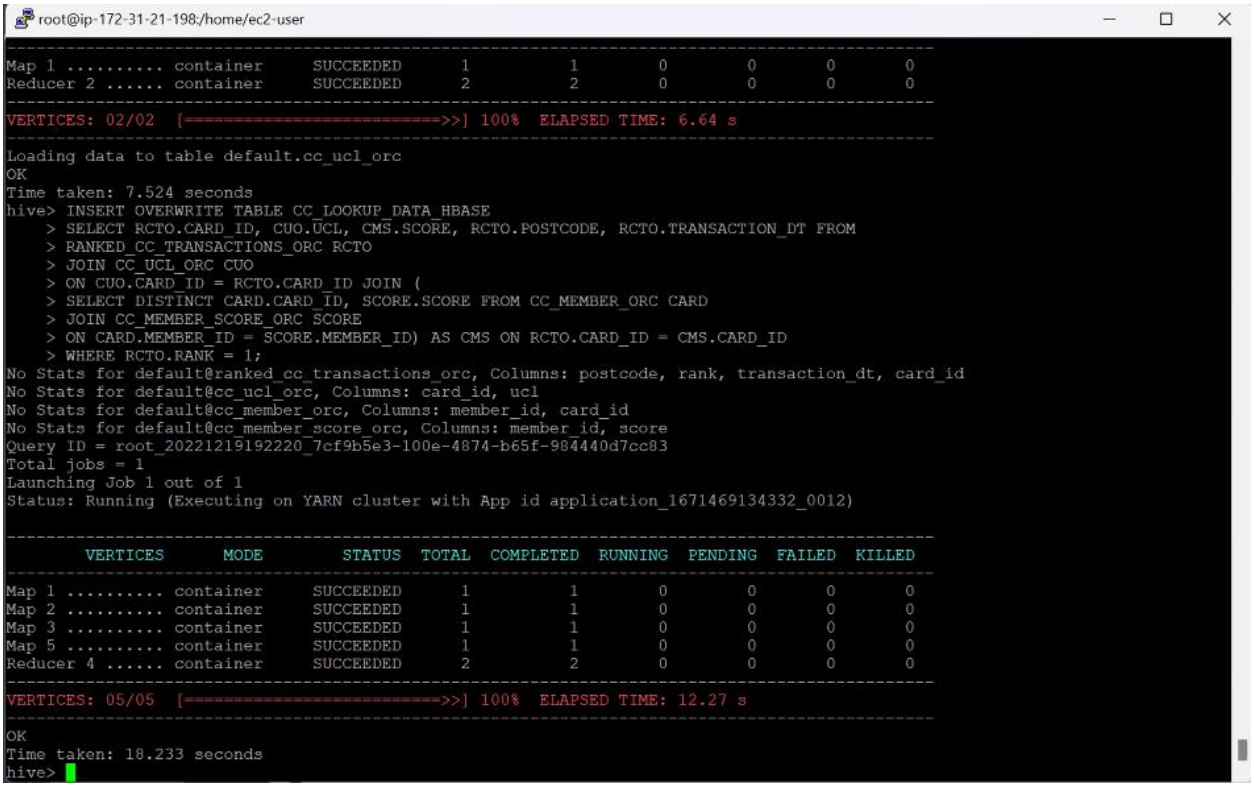
4. Loading data in "**cc_ucl_orc**" table.
   In the innermost query, select card_id, average of amount and standard deviation of amount from card_transactions_orc.
   In outermost query, select card_id andcompute UCL using average and standard deviation with formula (avg + (3 * stddev)). Inserting all thisdata in card_ucl_orc.

   INSERT OVERWRITE TABLE CC_UCL_ORC
   SELECT A.CARD_ID, (A.AVERAGE + (3 * A.STANDARD_DEVIATION)) AS UCL FROM (
   SELECT CARD_ID, AVG(AMOUNT) AS AVERAGE, STDDEV(AMOUNT) AS STANDARD_DEVIATION
   FROM RANKED_CC_TRANSACTIONS_ORC
   GROUP BY CARD_ID) A;



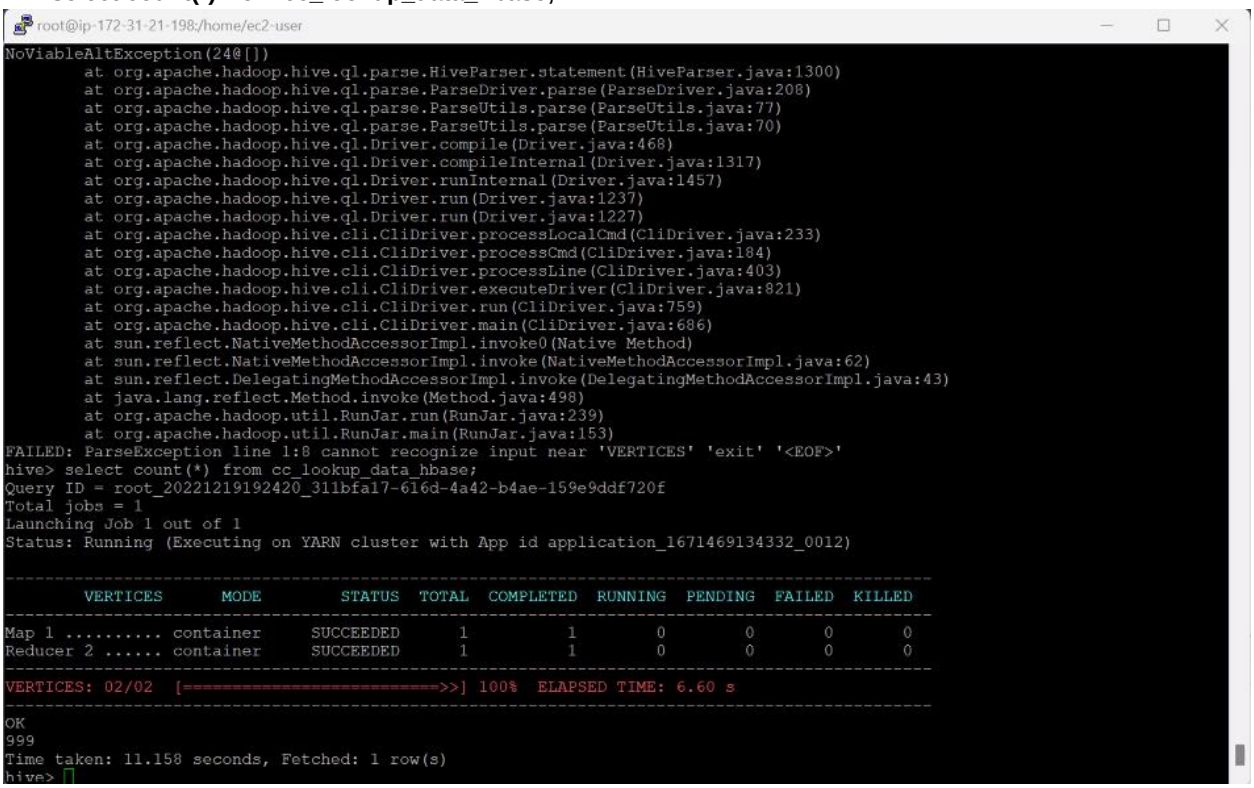
5. Load data in **cc_lookup_data_hbase** table.

INSERT OVERWRITE TABLE CC_LOOKUP_DATA_HBASE
SELECT RCTO.CARD_ID, CUO.UCL, CMS.SCORE, RCTO.POSTCODE, RCTO.TRANSACTION_DT FROM RANKED_CC_TRANSACTIONS_ORC RCTO
JOIN CC_UCL_ORC CUO
ON CUO.CARD_ID = RCTO.CARD_ID JOIN (
SELECT DISTINCT CARD.CARD_ID, SCORE.SCORE FROM CC_MEMBER_ORC CARD
JOIN CC_MEMBER_SCORE_ORC SCORE
ON CARD.MEMBER_ID = SCORE.MEMBER_ID) AS CMS ON RCTO.CARD_ID = CMS.CARD_ID
WHERE RCTO.RANK = 1;



6. Verifying count in "**cc_lookup_data_hbase**" table.

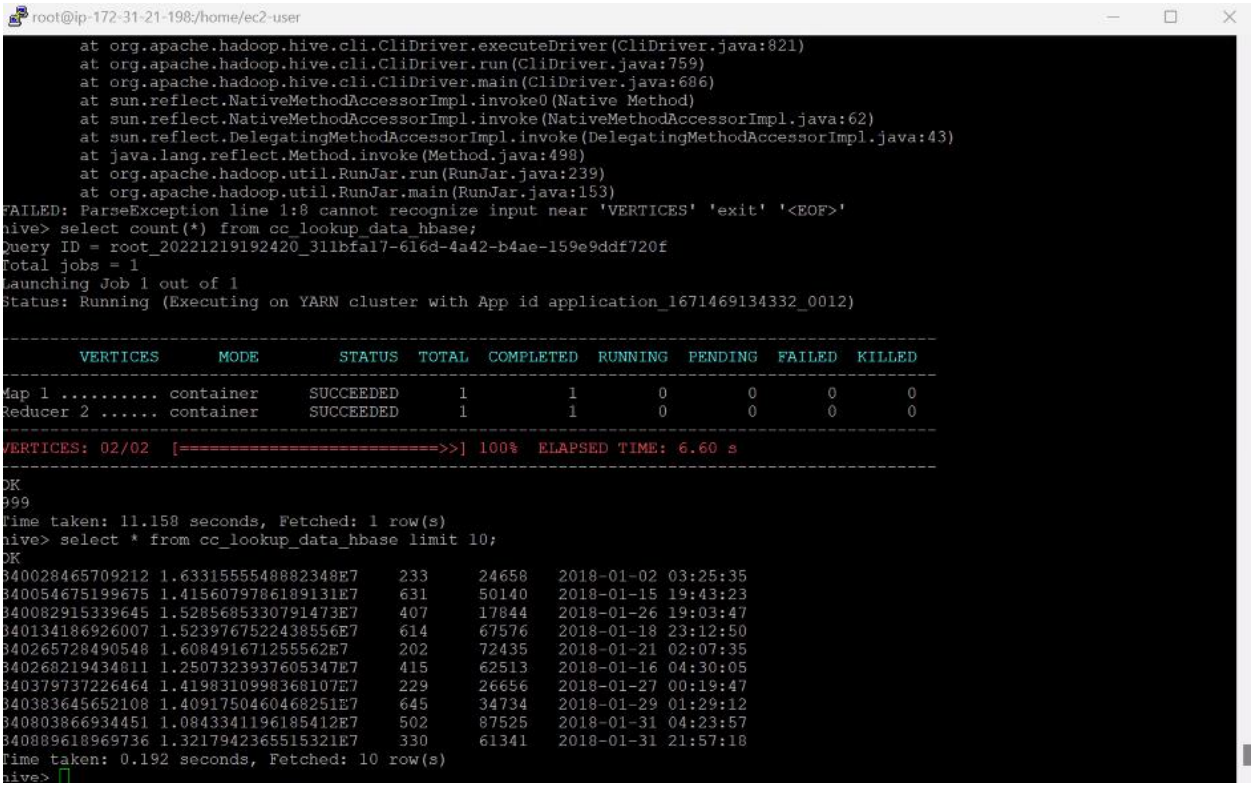
select count(*) from cc_lookup_data_hbase;



*Total number of records is **999** which is matching with given requirement.*

7. Verify some data in "**cc_lookup_data_hbase**" table.

select * from cc_lookup_data_hbase limit 10;



8. Start HBase shell session and verifying count in "**cc_lookup_data_hive**" table.

count 'cc_lookup_data_hive'

*Total number of records is **999** which is matching with given requirement.*

9. Verifying data in "**cc_lookup_data_hive**" table.

scan 'cc_lookup_data_hive'