Mid-Submission – Logic Explanation

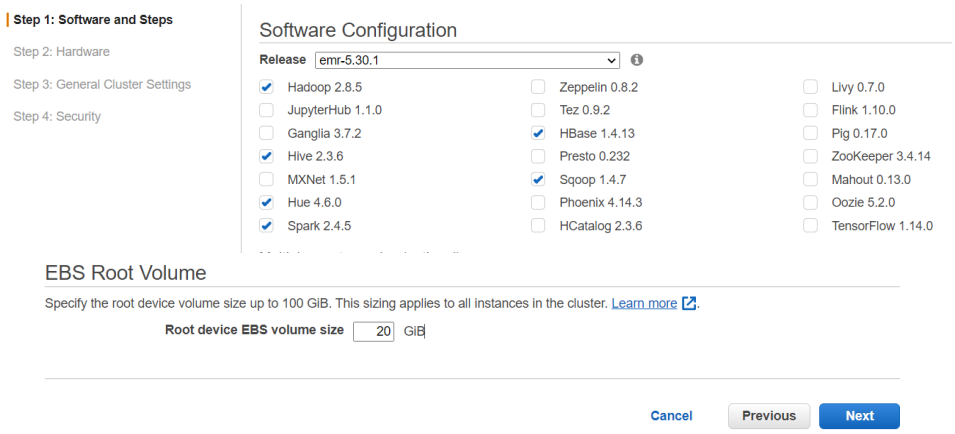
- **Explanation of the solution to the batch layer problem**

*****Please zoom to 180% or 200% to see screenshots with better clarity.**

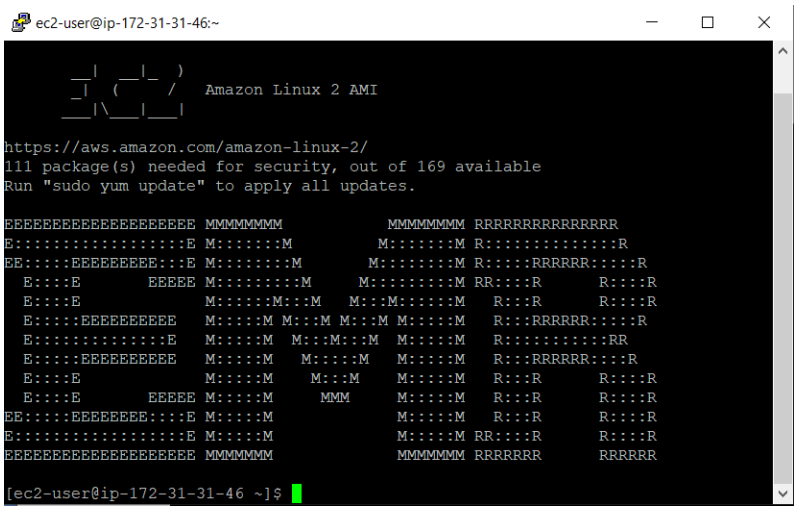
1. In order to complete below tasks, I have created EMR cluster with **Hadoop, Sqoop, Hive, HBase and Spark,** Root device EBS volume size **as** 20 GB

- **Task 1**: Load the transactions history data (card_transactions.csv) in a NoSQL database.
- **Task 2**: Ingest the relevant data from AWS RDS to Hadoop.
- **Task 3**: Create a look-up table with columns specified earlier in the problem statement.
- **Task 4**: After creating the table, you need to load the relevant data in the lookup table.

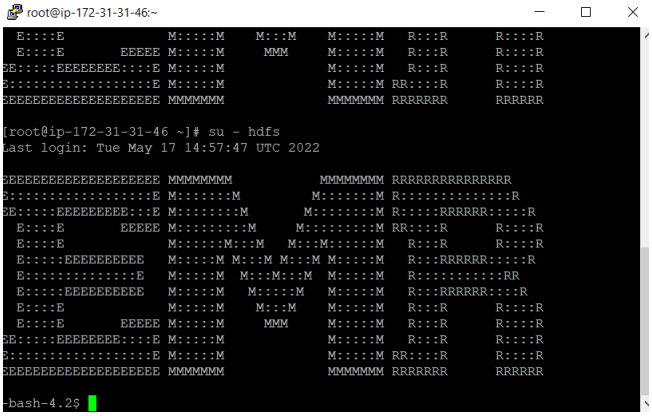
EMR Cluster Configuration:



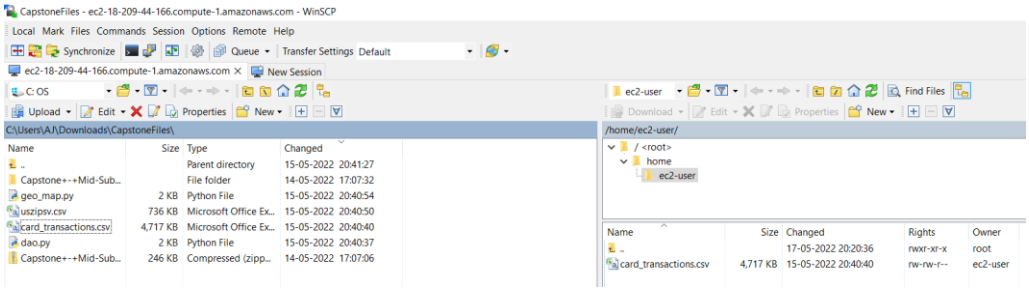
2. **Logged into EMR instance as "ec2-user"**



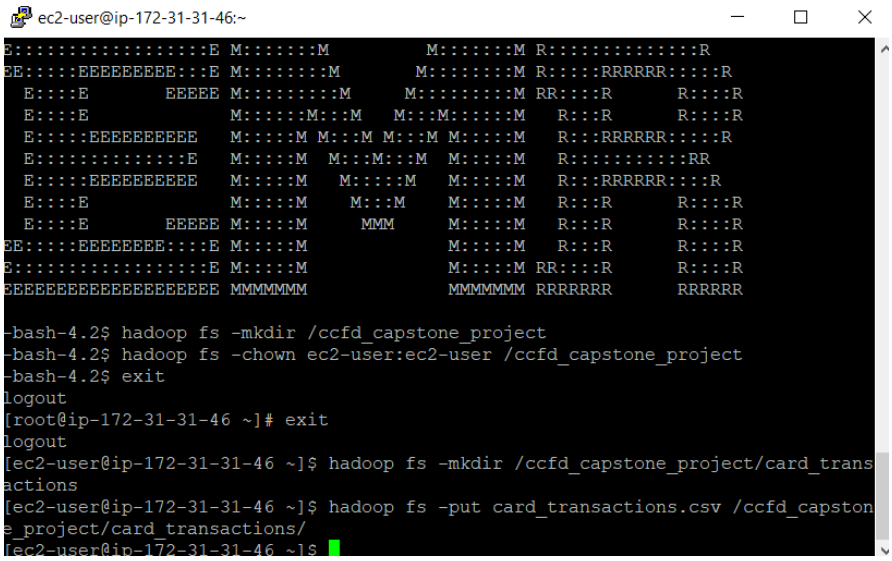
3. Switch to root user and then to hdfs user.
   Create directory and change its ownership -> exit from hdfs user -> exit from root user back to ec2-user.
   sudo su –
   su – hdfs
   hadoop fs -mkdir /ccfd_capstone_project
   hadoop fs -chown ec2-user:ec2-user /ccfd_capstone_project



4. Downloaded **card_transactions.csv** from the resource section of the capstone project from the learning platform and transfer it to ec2 instance via WinSCP.



5. Create a directory in HDFS and copy card_transactions.csv in that location.
   hadoop fs -mkdir /ccfd_capstone_project/card_transactions
   hadoop fs -put card_transactions.csv /ccfd_capstone_project/card_transactions/



Now our basic setup is ready for the project. We can now start with completing desired tasks

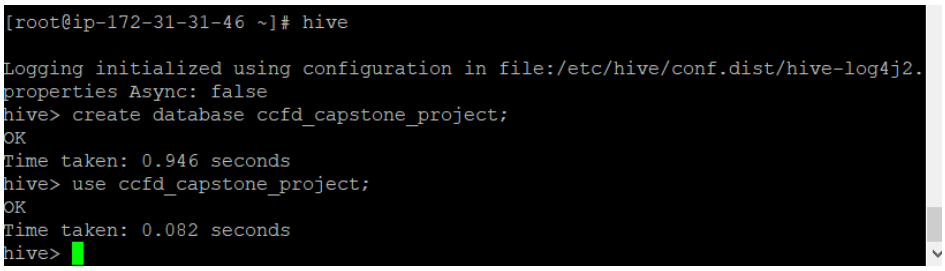
❖ Task 1: Load the transactions history data (card_ transactions.csv) in a NoSQL database.

---------- Hive Operations: Starts Here ----------

1. Start hive and create new database named ccfd_capstone_project -> switch to ccfd_capstone_project database

   create database ccfd_capstone_project;

   use ccfd_capstone_project;



2. Set below 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;

```
Time taken: 0.946 seconds
hive> use ccfd_capstone_project;
OK
Time taken: 0.082 seconds
hive> set hive.auto.convert.join=false;
hive> set hive.stats.autogather=true;
hive> set orc.compress=SNAPPY;
hive> set hive.exec.compress.output=true;
hive> set mapred.output.compression.codec=org.apache.hadoop.io.compress.SnappyCo
dec; set mapred.output.compression.type=BLOCK;
hive> set mapreduce.map.java.opts=-Xmx5G; set mapreduce.reduce.java.opts=-Xmx5G;

hive> set mapred.child.java.opts=-Xmx5G -XX:+UseConcMarkSweepGC -XX:-UseGCOverhe
adLimit;
hive> set hive.auto.convert.join=false;
hive>
```

3. Create 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 '/ccfd_capstone_project/card_transactions' TBLPROPERTIES
("skip.header.line.count"="1");
```

```
hive> 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 '/ccfd_capstone_project/card_transactions' TBLPROPERTIES ("skip.h
eader.line.count"="1");
OK
Time taken: 0.423 seconds
```

4. Create table "**card_transactions_orc**" in ORC format for better performance.

```
CREATE TABLE IF NOT EXISTS CARD_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");
```

```
hive> CREATE TABLE IF NOT EXISTS CARD_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");
OK
Time taken: 0.431 seconds
hive>
```

5. Load data in "**card_transactions_orc**" table and type cast **transaction_dt** column in timestamp format

   INSERT OVERWRITE TABLE CARD_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;

```
hive> INSERT OVERWRITE TABLE CARD_TRANSACTIONS_ORC SELECT CARD_ID, MEMBER_ID, AM
OUNT, POSTCODE, POS_ID,
    > CAST(FROM_UNIXTIME(UNIX_TIMESTAMP(TRANSACTION_DT,'dd-MM-yyyy HH:mm:ss')) A
S TIMESTAMP), STATUS
    > FROM CARD_TRANSACTIONS_EXT;
Query ID = root_20220517153514_c5f7123a-31ca-42b1-b5e2-ac12f4b4cedf
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1652799563314
_0002)

Map 1: 0/1
Map 1: 0/1
Map 1: 0(+1)/1
Map 1: 0(+1)/1
Map 1: 0/1
Map 1: 1/1
Loading data to table ccfd_capstone_project.card_transactions_orc
OK
Time taken: 19.689 seconds
hive>
```

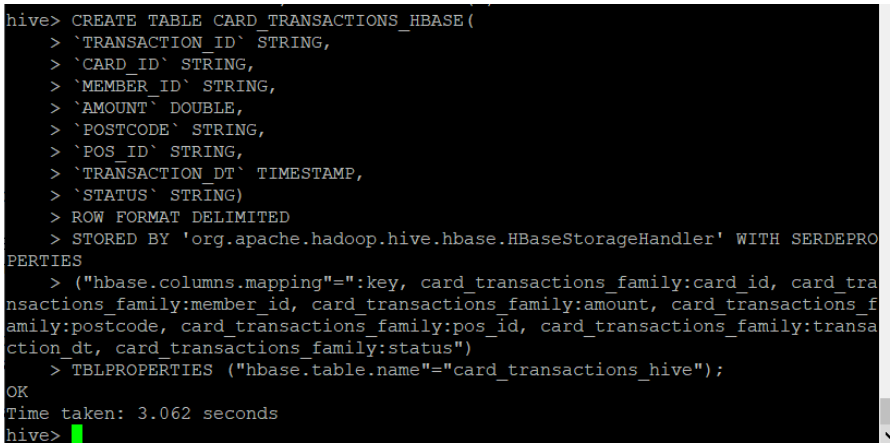
6. Verify **transaction_dt** and **year** columns in "**card_transactions_orc**" table.

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

```
hive> select year(transaction_dt), transaction_dt from card_transactions_orc lim
it 10;
OK
2018    2018-02-11 00:00:00
2018    2018-02-11 00:00:00
2018    2018-02-11 00:00:00
2018    2018-02-11 00:00:00
2018    2018-02-11 00:00:00
2018    2018-02-11 00:00:00
2018    2018-02-11 00:00:00
2018    2018-02-11 00:00:00
2018    2018-02-11 00:00:00
2018    2018-02-11 00:00:00
Time taken: 0.215 seconds, Fetched: 10 row(s)
```

7. Create hive-hbase integrated table which will be visible in HBase as well. "card_transactions_hbase" table

```
CREATE TABLE CARD_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_transactions_family:transaction_dt, card_transactions_family:status")
TBLPROPERTIES ("hbase.table.name"="card_transactions_hive");
```



8. Load data in "**card_transactions_hbase**" table which will be visible in HBase as well with table name as "**card_transactions_hive**".Using randomUUID to populate TRANSACTION_ID field (row key).

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

```
hive> INSERT OVERWRITE TABLE CARD_TRANSACTIONS_HBASE SELECT
    > reflect('java.util.UUID', 'randomUUID') as TRANSACTION_ID, CARD_ID, MEMBER
_ID, AMOUNT, POSTCODE, POS_ID, TRANSACTION_DT, STATUS
    > FROM CARD_TRANSACTIONS_ORC;
Query ID = root_20220517154447_cc5b8e5b-545e-48fd-bb6b-7bc6b86864a6
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1652799563314
_0003)

Map 1: -/-
Map 1: 0/1
Map 1: 0/1
Map 1: 0(+1)/1
Map 1: 0(+1)/1
Map 1: 0(+1)/1
Map 1: 1/1
OK
Time taken: 21.663 seconds
hive>
```

9. Verify data in "**card_transactions_hbase**" table.
   select * from card_transactions_hbase limit 10;

```
hive> select * from card_transactions_hbase limit 10;
OK
0000a2c5-ea89-4f33-bd77-f15911bcb220    6489878454988664        2972683110025794
926428.0        70039   373258348446110 2017-11-11 00:00:00     GENUINE
0000a7b9-1ed8-4b18-bb89-f00005526e85    6460729612153589        2666295188960983
746192.0        29821   615631599094052 2016-04-10 22:22:08     GENUINE
0000f150-93c7-4782-aea9-944bc7d8422b    6225866702124777        4526654546968662
148203.0        71933   706583735674375 2016-07-21 16:24:22     GENUINE
00012bac-26e6-470d-ba52-b2daa7eb8310    6011780257723719        3750740370124313
14328.0 10530   924255902406661 2016-10-06 03:00:22      GENUINE
00022e01-5c25-4b5e-b907-370657ffe698    6225310494984197        8102520065819358
244089.0        40902   481889839995759 2016-06-27 00:46:10     GENUINE
000240e2-4f9a-4113-a31c-8738fd70ee33    375667514735949 611432563010764 5864897.
0       56438   306783814643367 2016-05-14 17:27:18     GENUINE
0004296f-0791-4c48-87f1-96f338807d17    6512496325844338        8534108024046004
123520.0        53817   963411541449912 2017-05-15 20:13:28     GENUINE
0005b9f6-fa80-42c7-9d43-8ee70e2111eb    374437449333250 738960224159727 3287814.
0       98637   321930814986285 2017-04-12 11:40:18     GENUINE
00078892-b59b-4f03-ac33-aef76e44fc19    6463116552169683        3060518870723702
899853.0        87421   988456562894160 2016-12-23 05:59:07     GENUINE
0007c150-10ce-4eab-aa20-843b9fb4ff13    346829826446934 872862304291422 7944691.
0       50059   496425180344856 2017-02-25 20:10:42     GENUINE
Time taken: 0.284 seconds, Fetched: 10 row(s)
hive>
```

---------- Hive Operations: Ends Here ----------

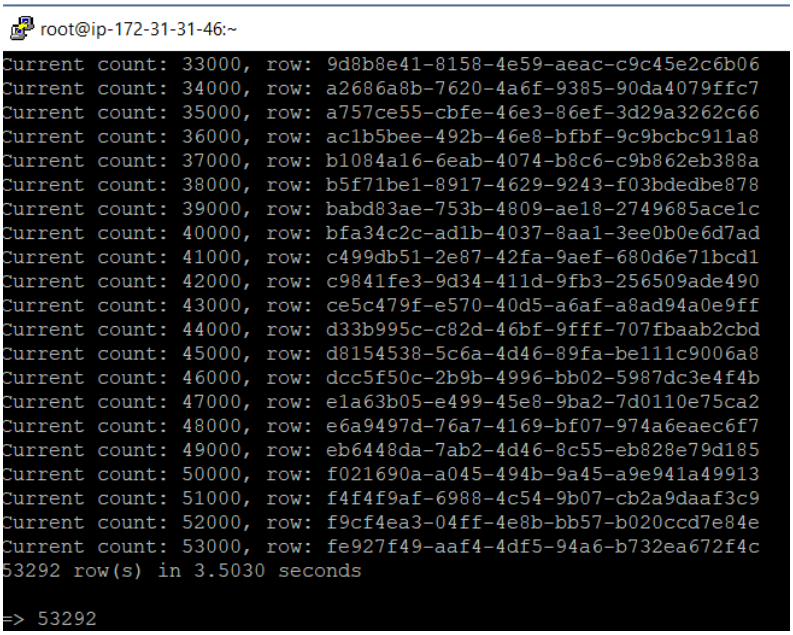
---------- Hbase Operations: Starts Here ----------

1. Start HBase and verify details of "**card_transactions_hive**" table (hive-hbase integrated table).
   describe 'card_transactions_hive'

```
hbase(main):001:0> describe 'card_transactions_hive'
Table card_transactions_hive is ENABLED
card_transactions_hive
COLUMN FAMILIES DESCRIPTION
{NAME => 'card_transactions_family', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'fal
se', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESS
ION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_S
COPE => '0'}
1 row(s) in 0.3120 seconds
```

2. Verify count of "**card_transactions_hive**" table

   Command : count 'card_transactions_hive'



---------- Hbase Operations: Ends Here ----------

Count of the "card_transactions_hive" table is 53292 which is matching with given requirement

❖ Task 2: Ingest the relevant data from AWS RDS to Hadoop.

---------- Sqoop Operations: Starts Here----------

1. Run 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 '/ccfd_capstone_project/member_score' \
-m 1
```

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

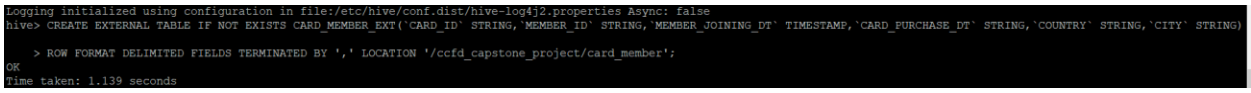
```
sqoop import --connect jdbc:mysql://upgradawsrds1.cyaielc9bmnf.us-east-
1.rds.amazonaws.com/cred_financials_data \
--username upgraduser \
--password upgraduser \
--table card_member \
--null-string 'NA' \
--null-non-string '\\N' \
--delete-target-dir \
--target-dir '/ccfd_capstone_project/card_member'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

---------- Sqoop Operations: Ends Here----------

---------- Hive Operations: Starts Here----------

1. Start hive and Create 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
'/ccfd_capstone_project/card_member';
```



2. Create external table "**member_score_ext**" to hold data from member_score table in RDS.

```
CREATE EXTERNAL TABLE IF NOT EXISTS MEMBER_SCORE_EXT(
`MEMBER_ID` STRING,
`SCORE` INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/ccfd_capstone_project/member_score';
```

3. Create "**card_member_orc**" table. For better performance.

```
CREATE TABLE IF NOT EXISTS CARD_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");
```

```
hive> CREATE TABLE IF NOT EXISTS CARD_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");
OK
Time taken: 0.348 seconds
hive>
```

4. Create "**member_score_orc**" table. For better performance.

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

```
hive> CREATE TABLE IF NOT EXISTS MEMBER_SCORE_ORC(
    > `MEMBER_ID` STRING,
    > `SCORE` INT) STORED AS ORC
    > TBLPROPERTIES ("orc.compress"="SNAPPY");
OK
Time taken: 0.075 seconds
hive>
```

5. Load data into "**card_member_orc**" table from "**card_member_ext**" table.

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

```
hive> INSERT OVERWRITE TABLE CARD_MEMBER_ORC
    > SELECT CARD_ID, MEMBER_ID, MEMBER_JOINING_DT, CARD_PURCHASE_DT, COUNTRY, C
ITY FROM CARD_MEMBER_EXT;
Query ID = root_20220517171744_c7913db6-a53e-4131-910e-e48ce7fba7f5
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1652805410081
_0010)

Map 1: 0/1
Map 1: 0/1
```

6. Load data into "**member_score_orc**" table from "**member_score_ext**" table.

```
INSERT OVERWRITE TABLE MEMBER_SCORE_ORC
SELECT MEMBER_ID, SCORE FROM MEMBER_SCORE_EXT;
```

```
hive> INSERT OVERWRITE TABLE MEMBER_SCORE_ORC
    > SELECT MEMBER_ID, SCORE FROM MEMBER_SCORE_EXT;
Query ID = root_20220517171927_321ca1d3-6e35-41dd-9400-4f9e94f87717
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1652805410081
_0010)

Map 1: 0/1
Map 1: 0/1
Map 1: 0(+1)/1
Map 1: 1/1
Loading data to table default.member_score_orc
OK
Time taken: 8.12 seconds
hive>
```

7. Verify data in "**card_member_orc**" table.

```
SELECT * FROM CARD_MEMBER_ORC LIMIT 10;
```

```
hive> SELECT * FROM CARD_MEMBER_ORC LIMIT 10;
OK
340028465709212 009250698176266 2012-02-08 06:04:13     05/13   United States   Barberton
340054675199675 835873341185231 2017-03-10 09:24:44     03/17   United States   Fort Dodge
340082915339645 512969555857346 2014-02-15 06:30:30     07/14   United States   Graham
340134186926007 887711945571282 2012-02-05 01:21:58     02/13   United States   Dix Hills
340265728490548 680324265406190 2014-03-29 07:49:14     11/14   United States   Rancho Cucamonga
340268219434811 929799084911715 2012-07-08 02:46:08     08/12   United States   San Francisco
340379737226464 089615510858348 2010-03-10 00:06:42     09/10   United States   Clinton
340383645652108 181180599313885 2012-02-24 05:32:44     10/16   United States   West New York
340803866934451 417664728506297 2015-05-21 04:30:45     08/17   United States   Beaverton
340889618969736 459292914761635 2013-04-23 08:40:11     11/15   United States   West Palm Beach
Time taken: 0.111 seconds, Fetched: 10 row(s)
hive>
```

8. Verify data in "**member_score_orc**" table.

SELECT * FROM MEMBER_SCORE_ORC LIMIT 10;

```
hive> SELECT * FROM MEMBER_SCORE_ORC LIMIT 10;
OK
000037495066290 339
000117826301530 289
001147922084344 393
001314074991813 225
001739553947511 642
003761426295463 413
004494068832701 217
006836124210484 504
006991872634058 697
007955566230397 372
Time taken: 0.098 seconds, Fetched: 10 row(s)
hive>
```

---------- **Hive Operations: Ends Here**----------

❖ **Task 3:** Create a look-up table with columns specified earlier in the problem statement.

Create “lookup_data_hbase” table (hive-hbase integrated table) which will be visible in HBase (lookup_data_hive).

---------- **Hive Operations: Starts Here**----------

CREATE TABLE 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_transaction_family:postcode, lookup_transaction_family:transaction_dt") TBLPROPERTIES ("hbase.table.name" = "lookup_data_hive");

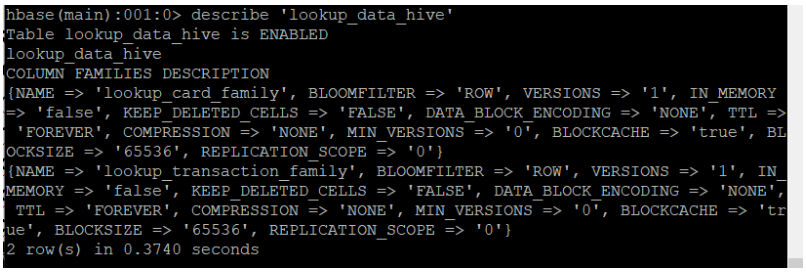
```
hive> CREATE TABLE LOOKUP_DATA_HBASE(`CARD_ID` STRING,`UCL` DOUBLE, `SCORE` INT,
 `POSTCODE` STRING, `TRANSACTION_DT` TIMESTAMP) STORED BY 'org.apache.hadoop.hiv
e.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES ("hbase.columns.mapping"=":key
, lookup_card_family:ucl, lookup_card_family:score, lookup_transaction_family:po
stcode, lookup_transaction_family:transaction_dt") TBLPROPERTIES ("hbase.table.n
ame" = "lookup_data_hive");
OK
Time taken: 2.698 seconds
hive>
```

---------- **Hive Operations: Ends Here**----------

---------- **Hbase Operations: Starts Here**----------

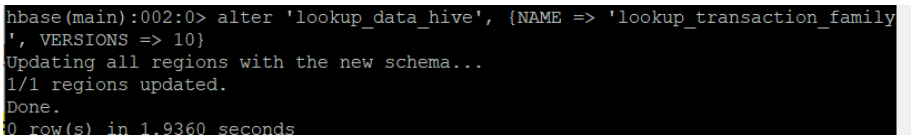
- Verify details of **lookup_data_hive** (hive-hbase integrated) table :

  describe 'lookup_data_hive'

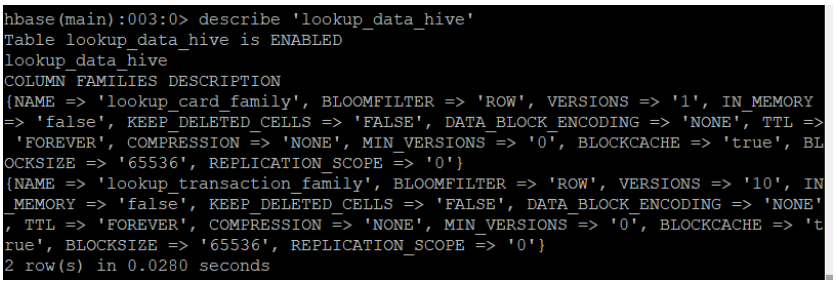


- Alter "**lookup_data_hive**" table and set VERSIONS to 10 for lookup_transaction_family. We are supposed to store last 10 transactions in lookup table so altering VERSIONS to 10.

  alter 'lookup_data_hive', {NAME => 'lookup_transaction_family', VERSIONS => 10}



- Verify details of "lookup_data_hive" (hive-hbase integrated) table after altering version to 10 :
  describe 'lookup_data_hive'



---------- **Hbase Operations: Starts Here**----------

❖ **Task 4:** After creating the table, you need to load the relevant data in the lookup table.

---------- Hive Operations: Starts Here ----------

1. Start hive and Create table "**ranked_card_transactions_orc**" to store last 10 transactions for each card_id. For better performance.

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

```
hive> CREATE TABLE IF NOT EXISTS RANKED_CARD_TRANSACTIONS_ORC(
    > `CARD_ID` STRING,
    > `AMOUNT` DOUBLE,
    > `POSTCODE` STRING,
    > `TRANSACTION_DT` TIMESTAMP,
    > `RANK` INT) STORED AS ORC
    > TBLPROPERTIES ("orc.compress"="SNAPPY");
OK
Time taken: 1.439 seconds
```

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

```
CREATE TABLE IF NOT EXISTS CARD_UCL_ORC(
`CARD_ID` STRING,
`UCL` DOUBLE) STORED AS ORC
TBLPROPERTIES ("orc.compress"="SNAPPY");
```

```
hive> CREATE TABLE IF NOT EXISTS CARD_UCL_ORC(
    > `CARD_ID` STRING,
    > `UCL` DOUBLE) STORED AS ORC
    > TBLPROPERTIES ("orc.compress"="SNAPPY");
OK
Time taken: 0.019 seconds
```

3. Load data in "**ranked_card_transactions_orc**" table

```
INSERT OVERWRITE TABLE RANKED_CARD_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_IDORDER BY A.TRANSACTION_DT DESC, AMOUNT DESC) AS RANK FROM
(SELECT CARD_ID, AMOUNT, POSTCODE, TRANSACTION_DT FROM
CARD_TRANSACTIONS_HBASE WHERESTATUS = 'GENUINE') A ) B WHERE B.RANK <= 10;
```

```
hive> INSERT OVERWRITE TABLE RANKED_CARD_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(PAR
TITION BY A.CARD_ID ORDER BY A.TRANSACTION_DT DESC, AMOUNT DESC) AS RANK FROM
    > (SELECT CARD_ID, AMOUNT, POSTCODE, TRANSACTION_DT FROM CARD_TRANSACTIONS_H
BASE WHERE STATUS = 'GENUINE') A ) B WHERE B.RANK <= 10;
Query ID = root_20220517175149_96a2fdc8-4660-45e9-9f74-2f9d31544d34
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1652805410081
_0013)

Map 1: -/-      Reducer 2: 0/2
Map 1: 0/1      Reducer 2: 0/2
Map 1: 0/1      Reducer 2: 0/2
Map 1: 0(+1)/1  Reducer 2: 0/2
Map 1: 0(+1)/1  Reducer 2: 0/2
Map 1: 1/1      Reducer 2: 0(+1)/2
Map 1: 1/1      Reducer 2: 1(+0)/2
Map 1: 1/1      Reducer 2: 1(+1)/2
Map 1: 1/1      Reducer 2: 2/2
Loading data to table ccfd_capstone_project.ranked_card_transactions_orc
OK
Time taken: 24.169 seconds
hive>
```

4. Load data in "**card_ucl_orc**" table. In innermost query, select card_id, average of amount and standard deviation of amount from card_transactions_orc. In outermost query, select card_id and compute UCL using average and standard deviation with formula (avg + (3 * stddev)). Insert all this data in card_ucl_orc.

```
INSERT OVERWRITE TABLE CARD_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_CARD_TRANSACTIONS_ORC
GROUP BY CARD_ID) A;
```

```
hive> INSERT OVERWRITE TABLE CARD_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_CARD_TRANSACTIONS_ORC
    > GROUP BY CARD_ID) A;
Query ID = root_20220517175347_40452f1b-db90-4db8-91ba-ed616b4ad0b3
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1652805410081_0013)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 ..... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ..... container     SUCCEEDED      2          2        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 6.96 s
----------------------------------------------------------------------------------------------
Loading data to table ccfd_capstone_project.card_ucl_orc
OK
Time taken: 8.337 seconds
hive>
```

5. Load data in **lookup_data_hbase** table.

```
INSERT OVERWRITE TABLE LOOKUP_DATA_HBASE
SELECT RCTO.CARD_ID, CUO.UCL, CMS.SCORE, RCTO.POSTCODE,
RCTO.TRANSACTION_DTFROM  RANKED_CARD_TRANSACTIONS_ORC RCTO
JOIN CARD_UCL_ORC CUO
ON CUO.CARD_ID =
RCTO.CARD_IDJOIN (
SELECT DISTINCT CARD.CARD_ID,
SCORE.SCOREFROM
CARD_MEMBER_ORC CARD
```

JOIN MEMBER_SCORE_ORC SCORE
ON CARD.MEMBER_ID =
SCORE.MEMBER_ID) AS CMSON
RCTO.CARD_ID = CMS.CARD_ID
WHERE RCTO.RANK = 1;

```
hive> INSERT OVERWRITE TABLE LOOKUP_DATA_HBASE
    > SELECT RCTO.CARD_ID, CUO.UCL, CMS.SCORE, RCTO.POSTCODE, RCTO.TRANSACTION_D
T FROM RANKED_CARD_TRANSACTIONS_ORC RCTO
    > JOIN CARD_UCL_ORC CUO
    > ON CUO.CARD_ID = RCTO.CARD_ID JOIN (
    > SELECT DISTINCT CARD.CARD_ID, SCORE.SCORE FROM CARD_MEMBER_ORC CARD
    > JOIN MEMBER_SCORE_ORC SCORE
    > ON CARD.MEMBER_ID = SCORE.MEMBER_ID) AS CMS ON RCTO.CARD_ID = CMS.CARD_ID
    > WHERE RCTO.RANK = 1;
No Stats for ccfd_capstone_project@ranked_card_transactions_orc, Columns: postco
de, rank, transaction_dt, card_id
No Stats for ccfd_capstone_project@card_ucl_orc, Columns: card_id, ucl
No Stats for ccfd_capstone_project@card_member_orc, Columns: member_id, card_id
No Stats for ccfd_capstone_project@member_score_orc, Columns: member_id, score
Query ID = root_20220518175747_e92f09f7-c3d4-4187-9f10-06d466c94feb
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1652895313854
_0006)

Map 1: 0/1      Map 2: 0/1      Map 3: 0/1      Map 5: 0/1      Reducer 4: 0/2
Map 1: 0/1      Map 2: 0/1      Map 3: 0/1      Map 5: 0/1      Reducer 4: 0/2
Map 1: 0/1      Map 2: 0(+1)/1  Map 3: 0/1      Map 5: 0(+1)/1  Reducer 4: 0/2
Map 1: 0/1      Map 2: 0(+1)/1  Map 3: 0/1      Map 5: 0(+1)/1  Reducer 4: 0/2
Map 1: 0/1      Map 2: 1/1      Map 3: 0/1      Map 5: 1/1      Reducer 4: 0/2
Map 1: 0(+1)/1  Map 2: 1/1      Map 3: 0(+1)/1  Map 5: 1/1      Reducer 4: 0/2
Map 1: 0(+1)/1  Map 2: 1/1      Map 3: 1/1      Map 5: 1/1      Reducer 4: 0/2
Map 1: 0(+1)/1  Map 2: 1/1      Map 3: 1/1      Map 5: 1/1      Reducer 4: 0(+1)
/2
Map 1: 0(+1)/1  Map 2: 1/1      Map 3: 1/1      Map 5: 1/1      Reducer 4: 2/2
Map 1: 1/1      Map 2: 1/1      Map 3: 1/1      Map 5: 1/1      Reducer 4: 2/2
OK
Time taken: 16.515 seconds
```

6. Verify count in "**lookup_data_hbase**" table.
   select count(*) from lookup_data_hbase;

```
hive> select count(*) from lookup_data_hbase;
Query ID = root_20220518180133_e681452e-a17f-4545-9e3c-bd6f68b48455
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1652895313854
_0007)

Map 1: -/-      Reducer 2: 0/1
Map 1: 0/1      Reducer 2: 0/1
Map 1: 0/1      Reducer 2: 0/1
Map 1: 0(+1)/1  Reducer 2: 0/1
Map 1: 1/1      Reducer 2: 0(+1)/1
Map 1: 1/1      Reducer 2: 1/1
OK
999
Time taken: 14.205 seconds, Fetched: 1 row(s)
hive>
```

Total number for record is **999** which is matching with given requirement.

7. Verify some data in "**lookup_data_hbase**" table.
   select * from lookup_data_hbase limit 10;

```
hive> select * from lookup_data_hbase limit 10;
OK
340028465709212 1.6331555548882348E7    233     24658   2018-01-02 03:25:35
340054675199675 1.4156079786189131E7    631     50140   2018-01-15 19:43:23
340082915339645 1.5285685330791473E7    407     17844   2018-01-26 19:03:47
340134186926007 1.5239767522438556E7    614     67576   2018-01-18 23:12:50
340265728490548 1.608491671255562E7     202     72435   2018-01-21 02:07:35
340268219434811 1.2507239376053 47E7    415     62513   2018-01-16 04:30:05
340379737226464 1.4198310998368107E7    229     26656   2018-01-27 00:19:47
340383645652108 1.4091750460468251E7    645     34734   2018-01-29 01:29:12
340803866934451 1.0843341196185412E7    502     87525   2018-01-31 04:23:57
340889618969736 1.3217942365515321E7    330     61341   2018-01-31 21:57:18
Time taken: 0.488 seconds, Fetched: 10 row(s)
```

---------- Hive Operations: Ends Here ----------

---------- Hbase Operations: Starts Here ----------

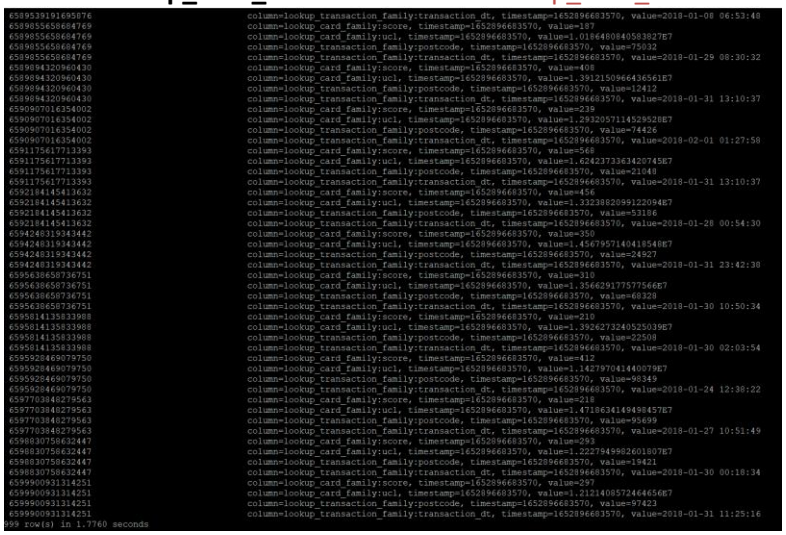
1. Start HBase shell and verify count in "**lookup_data_hive**" table.
   count 'lookup_data_hive'

```
hbase(main):001:0> count 'lookup_data_hive'
999 row(s) in 0.4970 seconds

=> 999
hbase(main):002:0>
```

Total number for record is **999** which is matching with given requirement.

2. Verify data in "**lookup_data_hive**" table. scan 'lookup_data_hive'



---------- Hbase Operations: Ends Here ----------