Logic Explanation
Mid Submission

STEPS:

1. Please find below submitted task as part of mid submission of capstone project-Credit card fraud detection.

    1. Task1-LoadNoSQL.pdf
    2. Task - SqoopDataIngestion.pdf
    3. Task3 - CreateNoSQL.pdf
    4. Task4 - PreAnalysis.pdf
    5. Task4 - Logic Mid.pdf
    6. Task4 - Scripts Execution.pdf

    HBase is used as NoSQL database for this project.

2. There are two sqoop jobs: First for incremental load of card_member table and Second for full load of member_score table.

3. Directory Setup in HDFS for the project.

    hadoop fs -mkdir /capstone_project

4. All hadoop shell, hive, hbase and sqoop commands are executed via ec2-user.

5. Download card_transactions.csv from Upgrad learning portal in the capstone project section. Upload the file in S3 bucket. Post that create a directory in HDFS and copy card_transactions.csv in that location.

    hadoop fs -mkdir /capstone_project/card_transactions
    hadoop distcp s3://creditcapstone/credit/card_transactions.csv /capstone_project/card_transactions/

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

1. Create new database named capstone_project in EMR cluster console using 'hive' command.

```
CREATE DATABASE capstone_project;

USE capstone_project;
```

(Set the DB)

2. Creation of external table card_transactions_ext table :

```
CREATE EXTERNAL TABLEIF NOT EXISTS card_transactions_ext( `card_id` string, `me
mber_id` string, `amount` DOUBLE, `postcode` string, `pos_id` string, `transaction_dt` st
ring, `status` string) row format delimited fields terminated BY ',' location '/
capstone_project/card_transactions' tblproperties ("skip.header.line.count"="1");
```

3. Create table card_transactions_orc for better performance.

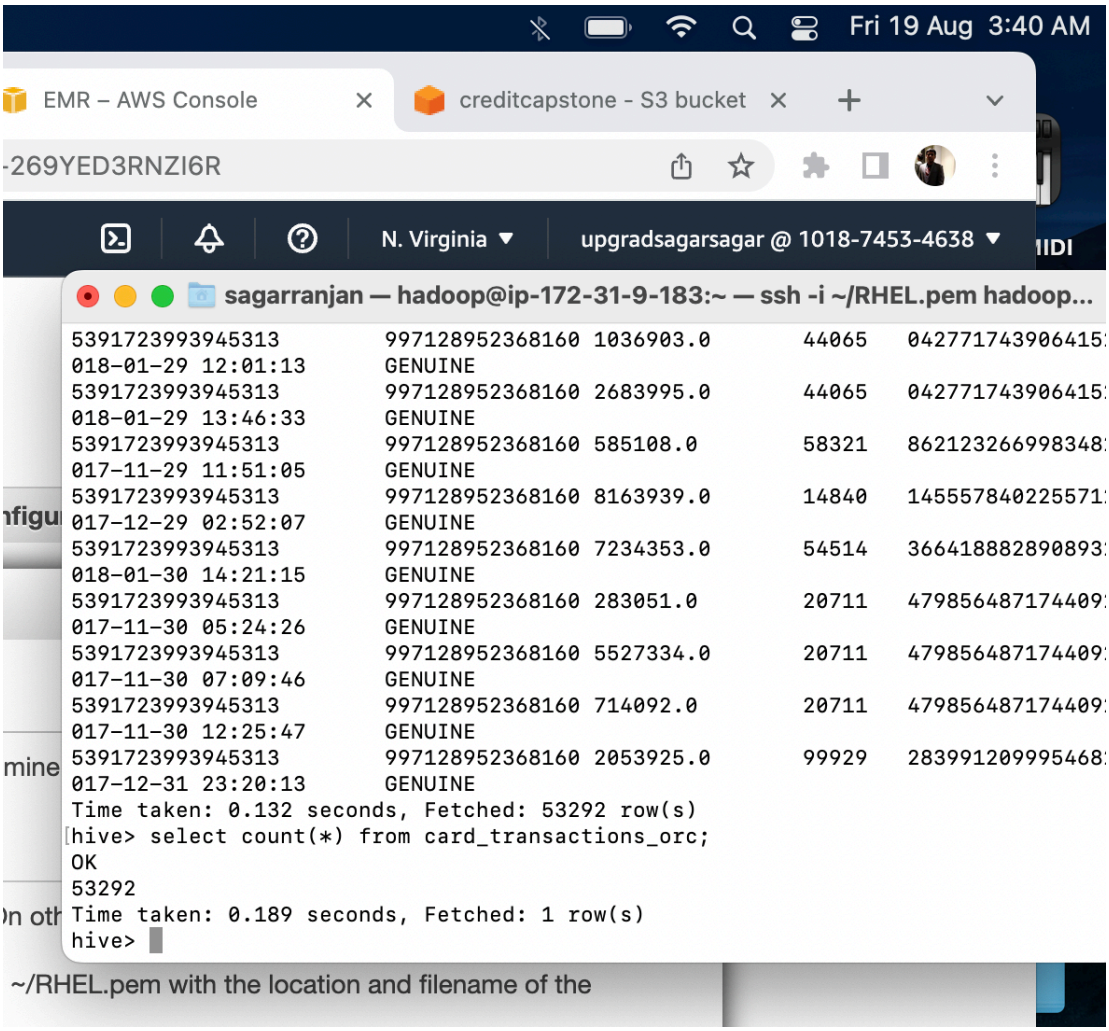
```
CREATE TABLE IF NOT EXISTS card_transactions_orc
(
   `card_id`         STRING,
   `member_id`       STRING,
   `amount`          DOUBLE,
   `postcode`        STRING,
   `pos_id`          STRING,
   `transaction_dt`  TIMESTAMP,
   `status`          STRING
) stored AS orc tblproperties ("orc.compress"="SNAPPY");
```

5. Load data in card_transactions_orc:

```
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;
```

6. Check the number of records from card_transactions_orc table.

   select count(*) from card_transactions_orc;



7. Table card_transactions_hbase hive-hbase:

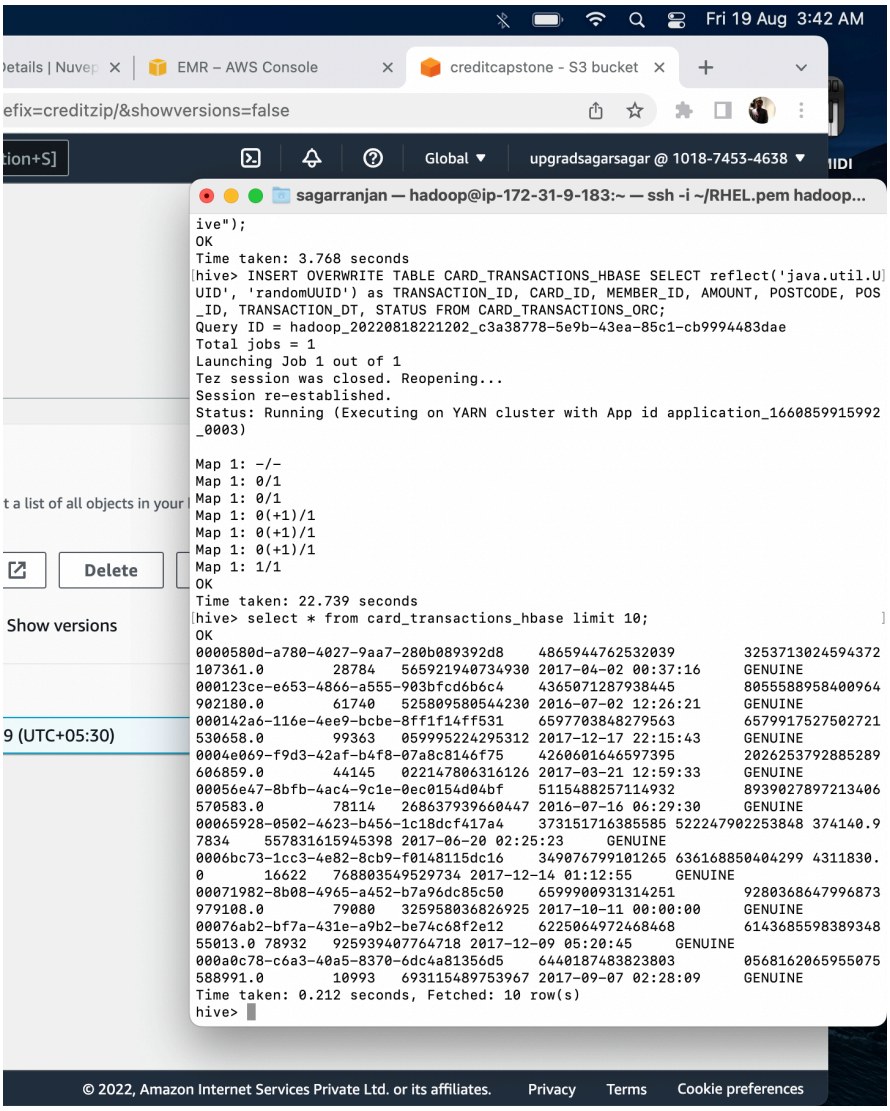
```
CREATE TABLE card_transactions_hbase
 (
    `transaction_id` STRING,
    `card_id`        STRING,
    `member_id`      STRING,
    `amount`         DOUBLE,
    `postcode`       STRING,
    `pos_id`         STRING,
    `transaction_dt` TIMESTAMP,
    `status`         STRING
  ) row format delimited 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_tran
sactions_family:pos_id, card_transactions_family:transaction_dt, card_transac
tions_family:status"
) tblproperties ("hbase.table.name"="card_transactions_hive");
```

8. Load data into card_transactions_hbase

```
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;
```

9. Fetching first ten transactions

```
SELECT * FROM   card_transactions_hbase LIMIT  10;
```




```
ive");
OK
Time taken: 3.768 seconds
hive> INSERT OVERWRITE TABLE CARD_TRANSACTIONS_HBASE SELECT reflect('java.util.U
UID', 'randomUUID') as TRANSACTION_ID, CARD_ID, MEMBER_ID, AMOUNT, POSTCODE, POS
_ID, TRANSACTION_DT, STATUS FROM CARD_TRANSACTIONS_ORC;
Query ID = hadoop_20220818221202_c3a38778-5e9b-43ea-85c1-cb9994483dae
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_1660859915992
_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: 22.739 seconds
hive> select * from card_transactions_hbase limit 10;
OK
0000580d-a780-4027-9aa7-280b089392d8    4865944762532039        3253713024594372
107361.0        28784   565921940734930 2017-04-02 00:37:16     GENUINE
000123ce-e653-4866-a555-903bfcd6b6c4    4365071287938445        8055588958400964
902180.0        61740   525809580544230 2016-07-02 12:26:21     GENUINE
000142a6-116e-4ee9-bcbe-8ff1f14ff531    6597703848279563        6579917527502721
530658.0        99363   059995224295312 2017-12-17 22:15:43     GENUINE
0004e069-f9d3-42af-b4f8-07a8c8146f75    4260601646597395        2026253792885289
606859.0        44145   022147806316126 2017-03-21 12:59:33     GENUINE
00056e47-8bfb-4ac4-9c1e-0ec0154d04bf    5115488257114932        8939027897213406
570583.0        78114   268637939660447 2016-07-16 06:29:30     GENUINE
00065928-0502-4623-b456-1c18dcf417a4    373151716385585 522247902253848 374140.9
7834    557831615945398 2017-06-20 02:25:23     GENUINE
0006bc73-1cc3-4e82-8cb9-f0148115dc16    349076799101265 636168850404299 4311830.
0       16622   768803549529734 2017-12-14 01:12:55     GENUINE
00071982-8b08-4965-a452-b7a96dc85c50    6599900931314251        9280368647996873
979108.0        79080   325958036826925 2017-10-11 00:00:00     GENUINE
00076ab2-bf7a-431e-a9b2-be74c68f2e12    6225064972468468        6143685598389348
55013.0 78932   925939407764718 2017-12-09 05:20:45     GENUINE
000a0c78-c6a3-40a5-8370-6dc4a81356d5    6440187483823803        0568162065955075
588991.0        10993   693115489753967 2017-09-07 02:28:09     GENUINE
Time taken: 0.212 seconds, Fetched: 10 row(s)
hive>
```

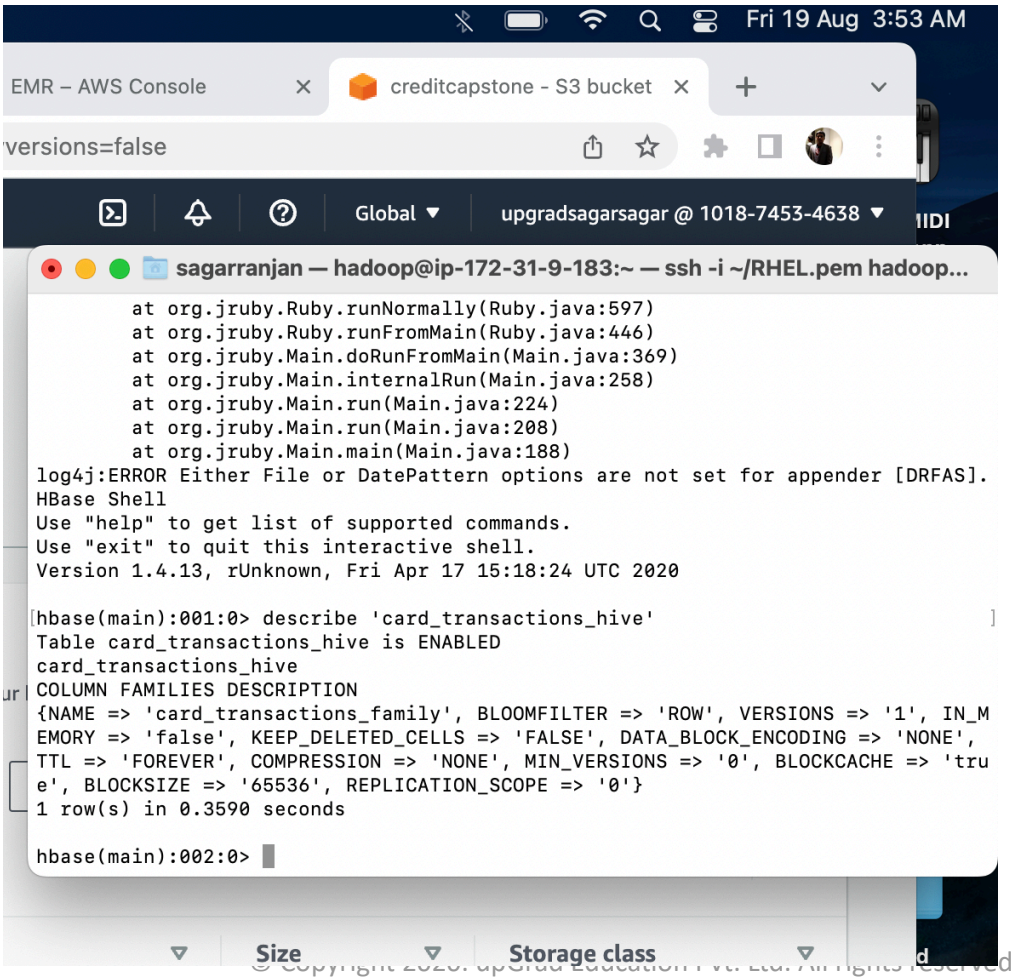
10. Lookup_data_hbase : Visible to HBase as well with name as lookup_data_hive.

```
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_f
amily:postcode, lookup_transaction_family:transaction_dt"
) tblproperties ("hbase.table.name" = "lookup_data_hive");
```

HBase Commands :

1. HBase shell from EMR cluster(Command : hbase shell)

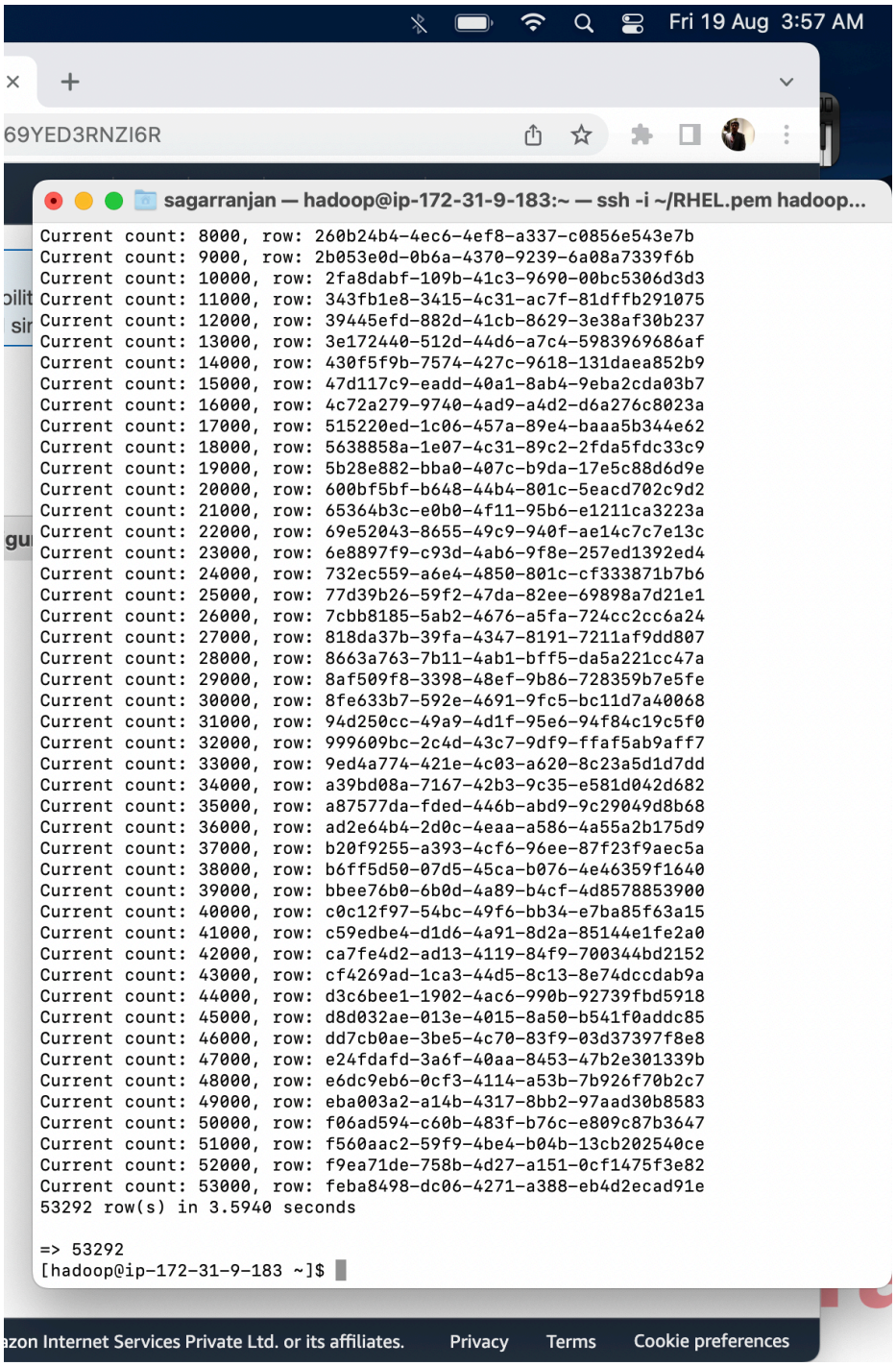
   describe 'card_transactions_hive'



2. Counting Records in card_transactions_hive in

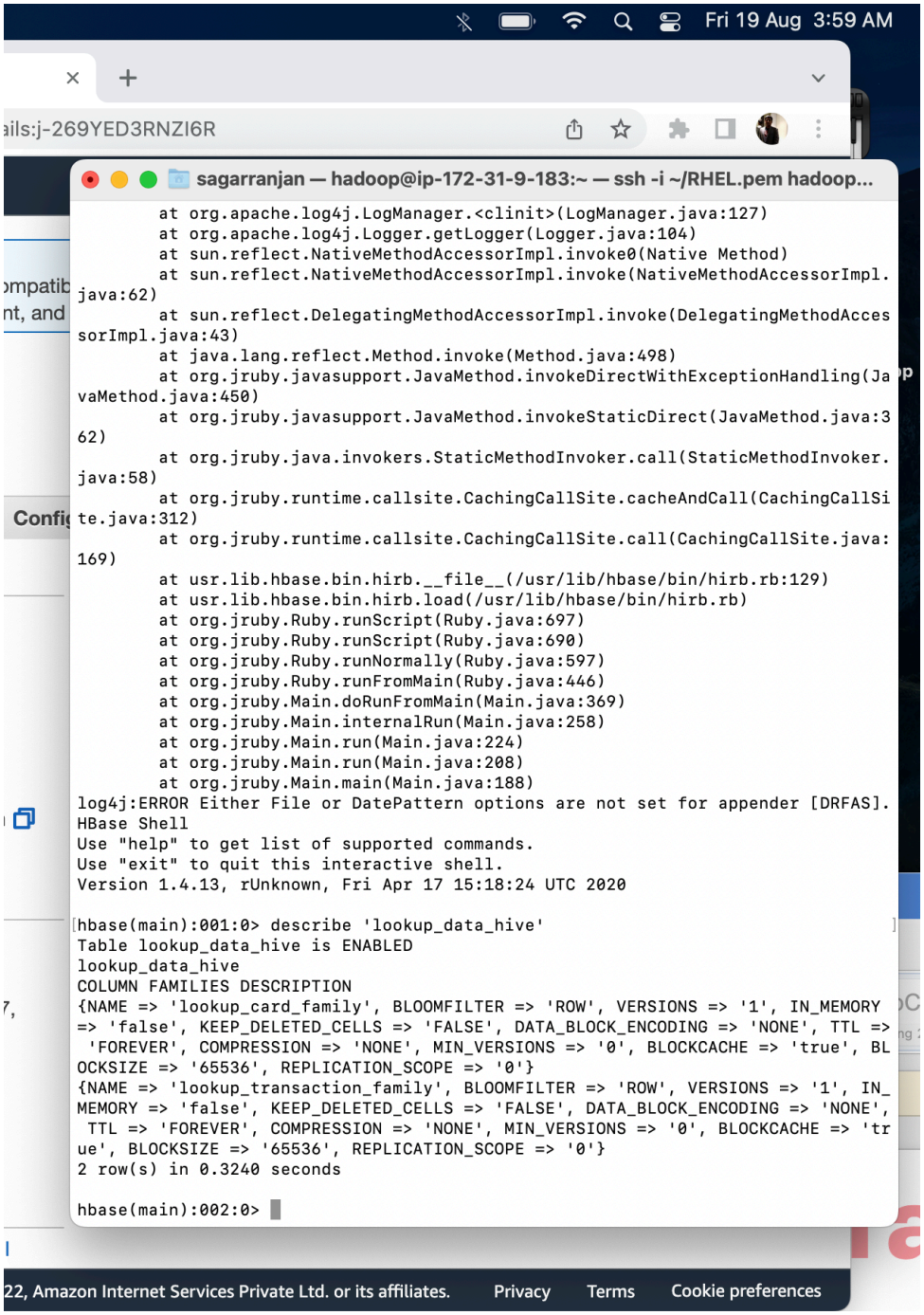
HBase

COUNT 'card_transactions_hive'



3. Describe lookup_data_hive hive-hbase in hBase

describe 'lookup_data_hive'



4.

In HBase, lookup_data_hive table has been altered and set VERSIONS to 10 for lookup_transaction_family.

Need to store last 10 transactions in lookup table so altering VERSIONS to 10.

I have created tow column in lookup table namely lookup_card_family and lookup_transaction_family.

```
ALTER 'lookup_data_hive', {NAME => 'lookup_transaction_family',
VERSIONS => 10}
```

For ingesting data from the RDS server into HDFS.

1. Run below Sqoop command to import member_score table from RDS into HDFS, from command prompt.

   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 '/capstone_project/member_score' -m1

2. Run below Sqoop command to import card_member table from RDS into HDFS, from command prompt.

   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 '/capstone_project/card_member' -m 1

---

1. Hive : Create external table card_member_ext which will point to HDFS location to hold (card_member table) date in RDS.

   CREATE EXTERNAL TABLEIF 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 '/capstone_project/card_member';

2. Create external table member_score_ext which will point to HDFS location to hold data from (member_score table) in RDS.

   CREATE EXTERNAL TABLEIF NOT EXISTS member_score_ext(`member_id` string, `score` int) row format delimited fields terminated BY ',' location '/capstone_project/member_score';

3. For 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");
```

4. For performance.

```
CREATE TABLE IF NOT EXISTS member_score_orc
  (
     `member_id` STRING,
     `score`     INT
  ) stored AS orc tblproperties ("orc.compress"="SNAPPY");
```

5. Load data into card_member_orc.

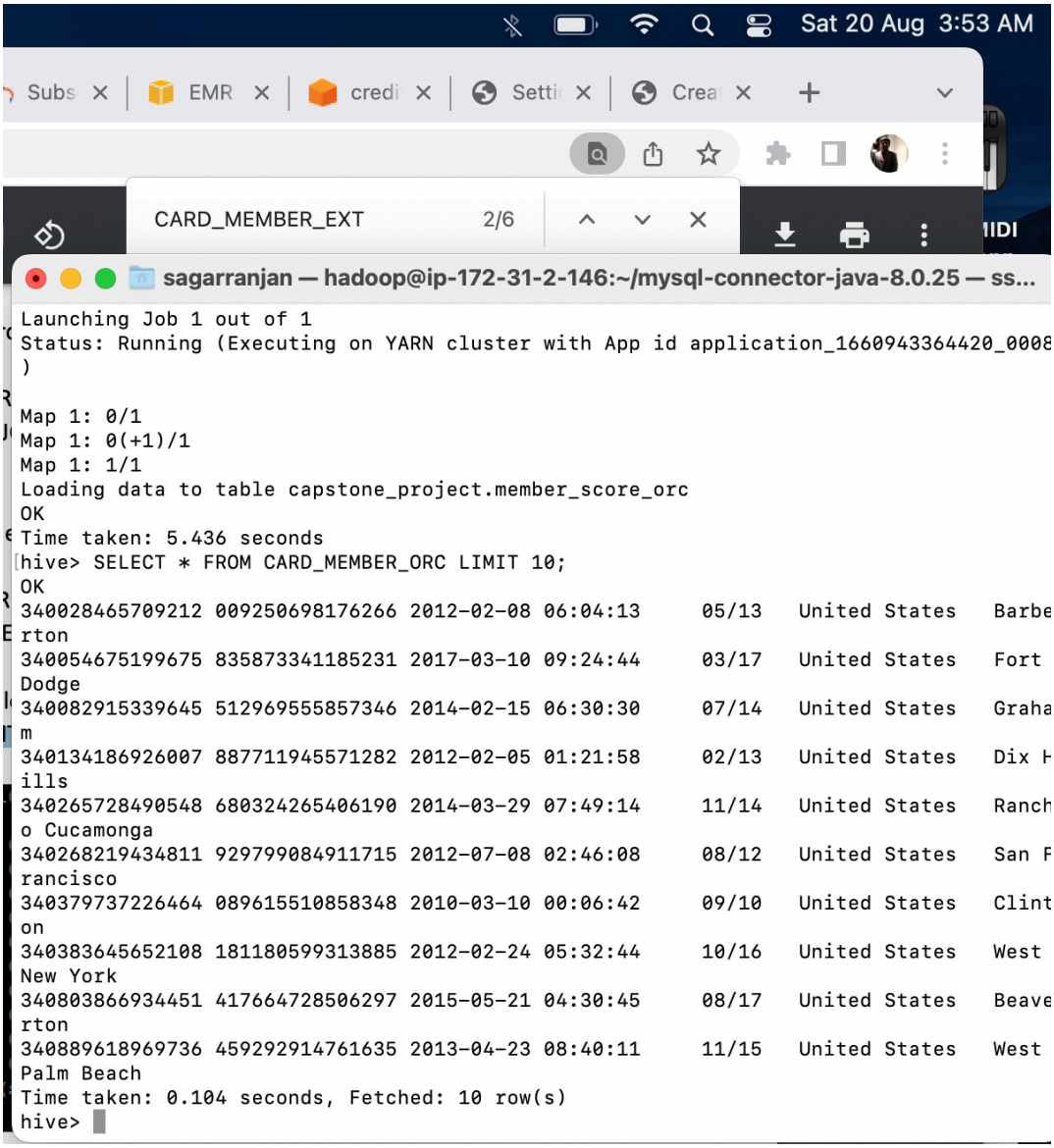
```
INSERT overwrite TABLE card_member_orc
SELECT card_id,
       member_id,
       member_joining_dt,
       card_purchase_dt,
       country,
       city
FROM   card_member_ext;
```

6. Load data into member_score_orc

```
INSERT overwrite TABLE member_score_orcSELECT member_id,
       score
FROM   member_score_ext;
```

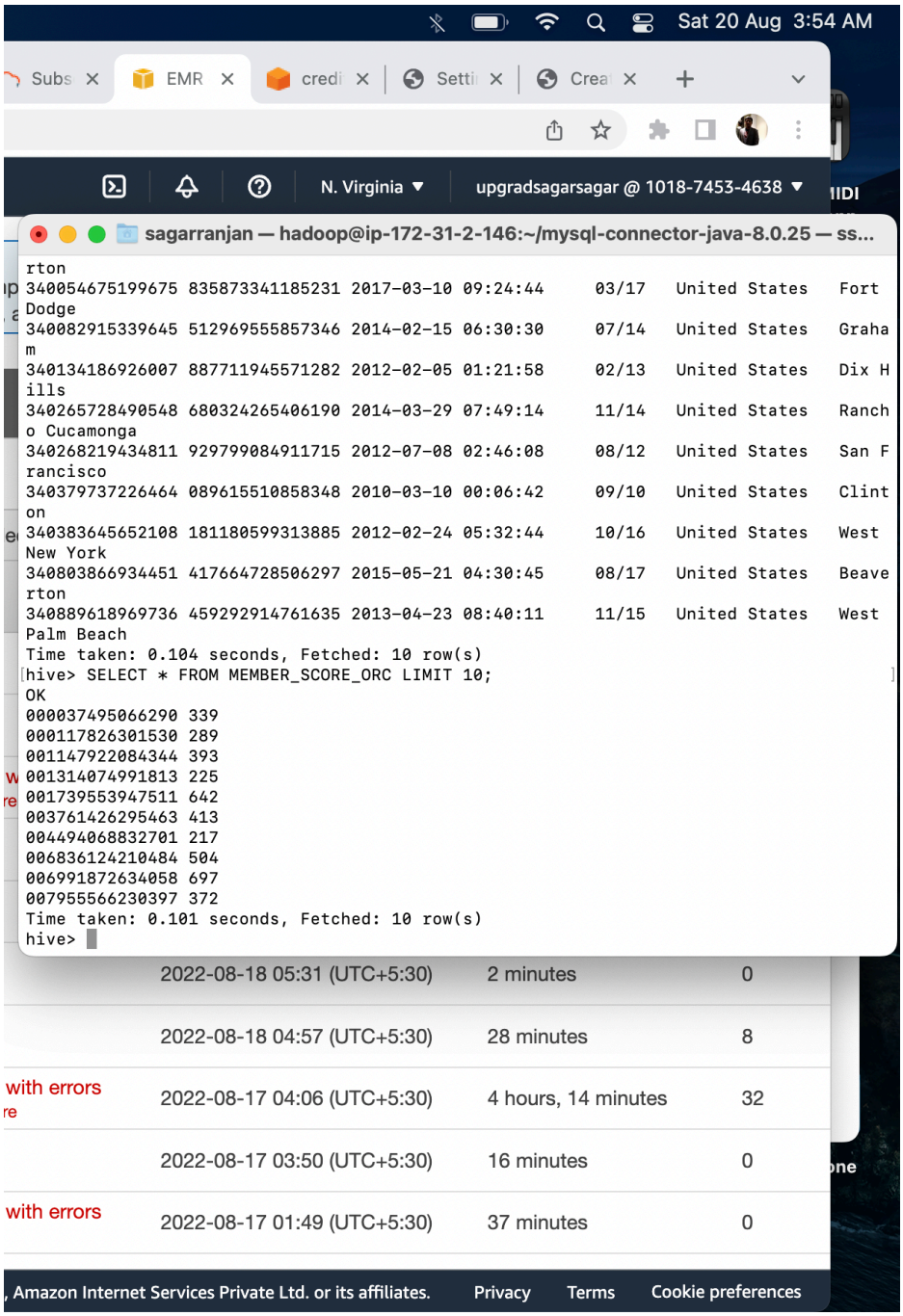
7. Verify data in card_member_orc table.

```
SELECT * FROM   card_member_orc LIMIT  10;
```



8. Verify some data in member_score_orc table.

```
SELECT * FROM   member_score_orc LIMIT  10;
```



Write a script to calculate the moving average and standard deviation of the last 10 transactions for each card_id for the data present in Hadoop and NoSQL database. If the total number of transactions for a particular card_id is less than 10, then calculate the parameters based on the total number of records available for that card_id. The script should be able to extract and feed the other relevant data ('postcode' ,'transaction_dt' ,'score', etc.) for the look-up table along with card_id and UCL.

Hive :

1. Table ranked_card_transactions_orc to store last 10 transactions for each card_id.

```
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");
```

2. Table card_ucl_orc to store UCL values for each card_id.

```
CREATE TABLE IF NOT EXISTS card_ucl_orc
  (
     `card_id` STRING,
     `ucl`     DOUBLE
  ) stored AS orc tblproperties ("orc.compress"="SNAPPY");
```

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_id ORDER BY a.transa
ction_dt DESC, amount DESC) AS rank
                FROM     (
                                SELECT card_id,
                                       amount,
                                       postcode,
                                       transaction_dt
                                FROM   card_transactions_hbase
                                WHERE  status = 'GENUINE') a ) b
WHERE  b.rank <= 10;
```

4. Load data in card_ucl_orc table

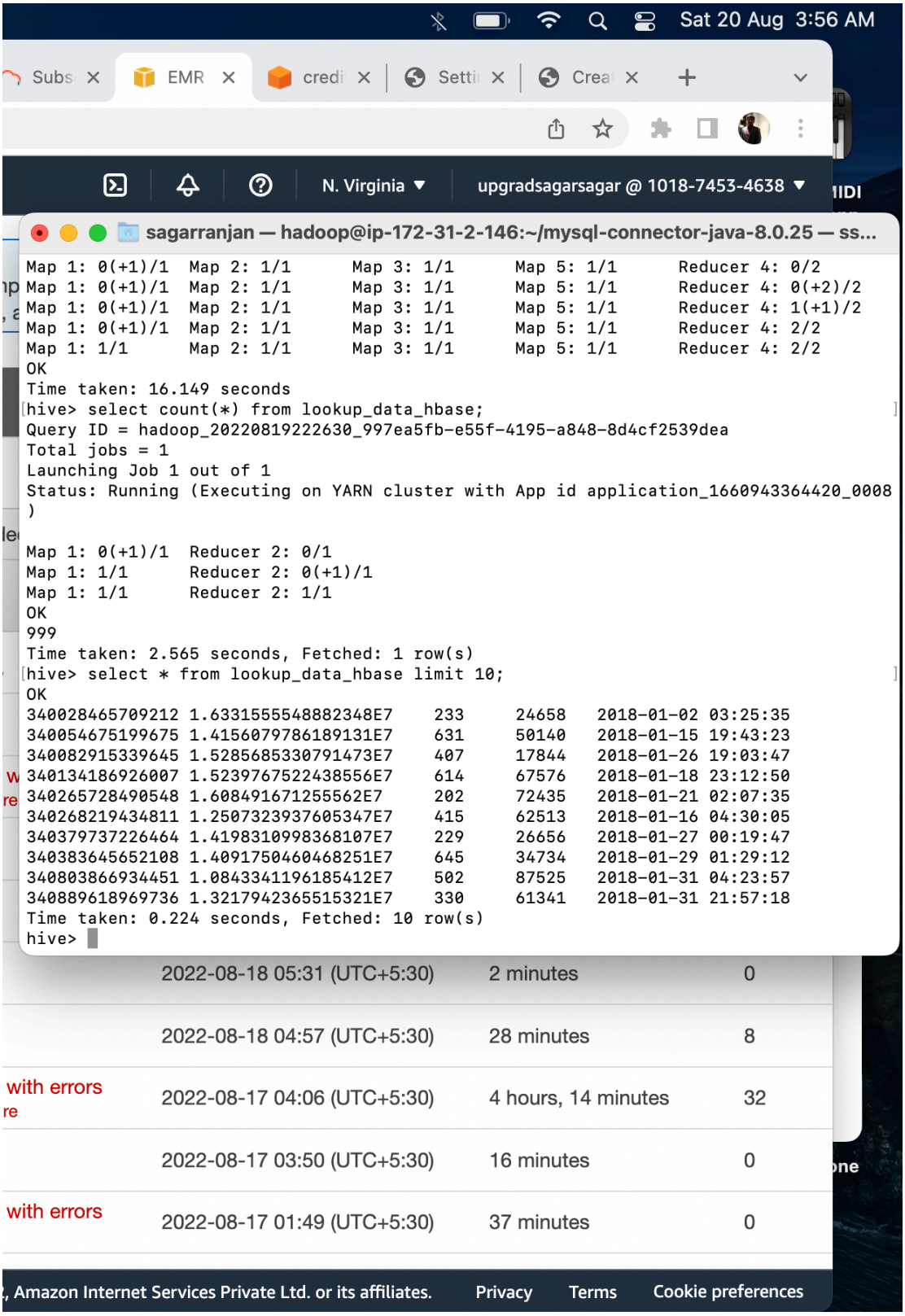
```
INSERT overwrite TABLE card_ucl_orc
SELECT a.card_id,
       (a.average + (3 * a.standard_deviation)) AS ucl
FROM   (
                SELECT   card_id,
                         avg(amount)    AS average,
                         stddev(amount) AS standard_deviation
                FROM     ranked_card_transactions_orc
                GROUP BY card_id) a;
```

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_dt
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 c
ms
ON     rcto.card_id = cms.card_id
WHERE  rcto.rank = 1;
```

6. Count in

   lookup_data_hbase table.

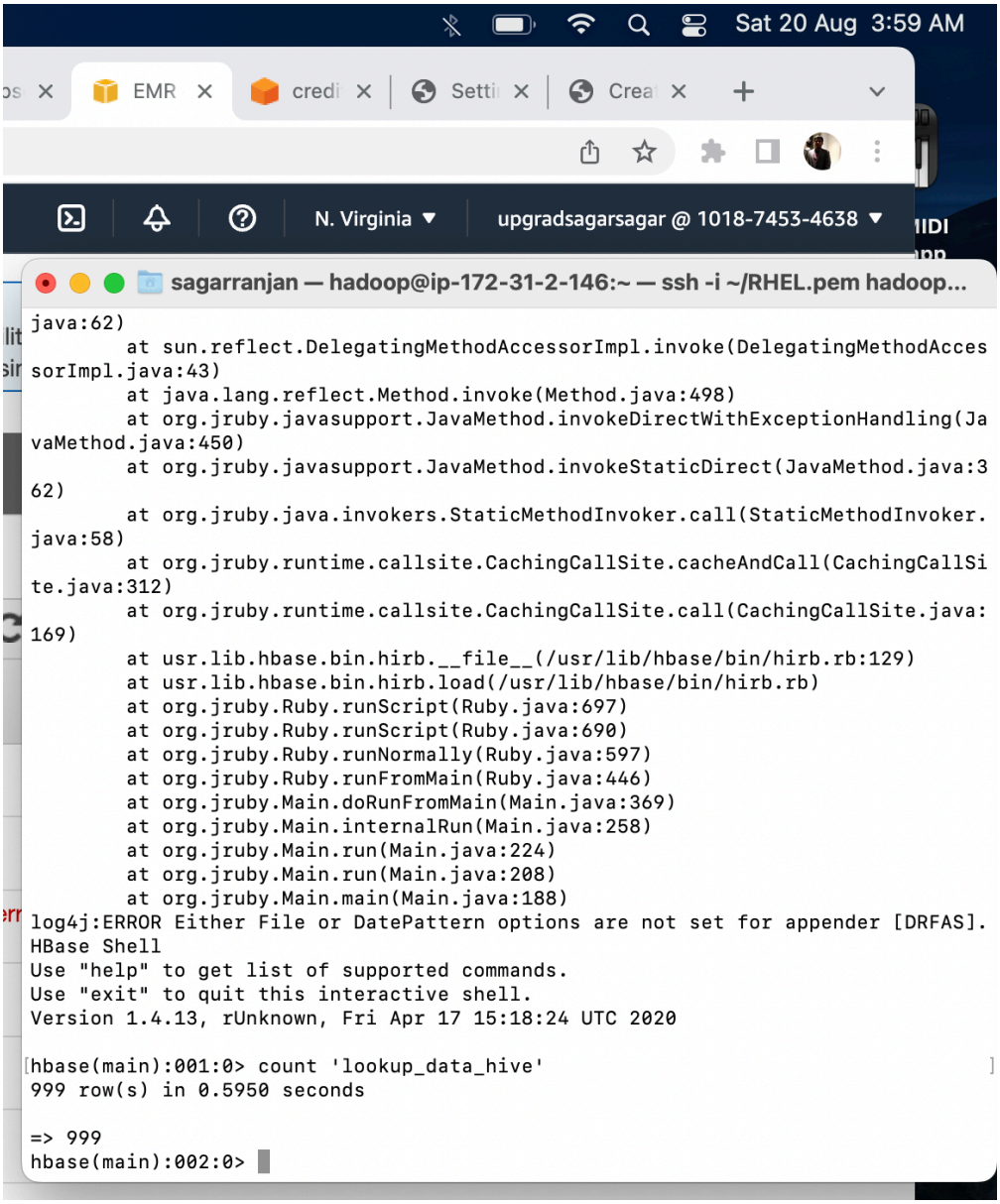


HBase Commands:

1. Start HBase shell from command prompt. In HBase,

   Verify count in lookup_data_hive table.

```
COUNT 'lookup_data_hive'
```



2. In HBase, verify data in lookup_data_hive table.

```
scan 'lookup_data_hive'
```