BATCH PROCESSING

As part of batch processing, we are using the following tools/applications:

- **Apache Sqoop:** Data import from AWS RDS to HDFS in EMR cluster
- **Apache Hive:** Data query and analysis on top of imported source data in HDFS.
- **Apache Sark:** Generate KPI/derived metrics from Hive tables with faster execution engine (Data preparation).
- **Apache HBase:** Store the pre-processed data and generated metrics from Hive and Spark and can be used for real time processing (Spark-Hive-HBase integration).

PROBLEM APPROACH

1. Prepare the data ingestion shell script to get the card member and member score tables data from AWS RDS to HDFS in EMR cluster using Apache Sqoop.
2. Prepare the necessary HQL scripts to create hive tables for card member and member score data on top of imported source data in HDFS which can be used for data processing in the later stages.

   Card Member table:
   - Create a staging table for card member pointing to imported data location with proper attribute types.
   - Create a final bucketed table with 8 buckets splitted on card_id and member_id attributes to achieve joins optimization during data processing and load the data from staging table in the form of Parquet for faster data access with less storage occupied.

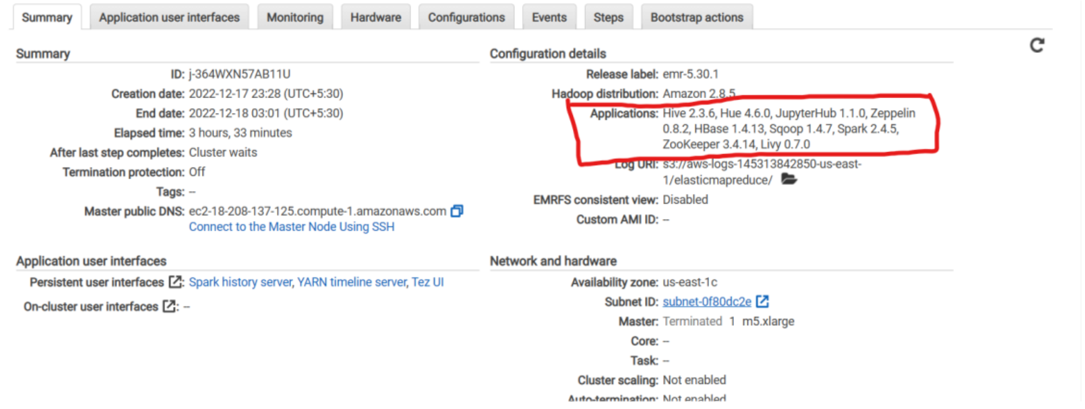
   Member Score table:
   - Create a staging table for member score pointing to imported data location with proper attribute types.
   - Create a final bucketed table with 8 buckets splitted on member_id attribute to achieve joins optimization during data processing and load the data from staging in the form of Parquet for faster data access with less storage occupied.
3. Prepare the necessary HQL and shell scripts to create card transactions table in Hive and HBase to store historical data provided (Hive-HBase integration).
   - Create a source card transactions table to dump the raw data from CSV file provided to Hive table.
   - Create a staging card transactions table and load the data from source table by transforming transaction_dt attribute data to proper format (**'dd-MM-yyyy HH:mm:ss' to 'yyyy-MM-dd HH:mm:ss'**).
   - Create a final bucketed table with 16 buckets splitted on card_id and member_id attributes (considering the volume and growth of real time transactions) to achieve joins optimization during data processing and load the data from staging table with HBase storage handler so that the same data will be available in card transactions HBase table which can be used for real time processing.

4. Prepare the necessary PySpark, HQL and shell scripts to create card lookup table in Hive and HBase (Hive-HBase integration) and load the required metrics data by generating from member score and card transactions tables which will be used for data analysis in real time.
    - Fetch and calculate the attributes and metrics such as **card_id, ucl, postcode, transaction_dt, credit_score** by joining member score and card transactions tables and perform calculations using Apache Spark (PySpark - Spark SQL module). The reason for choosing Apache Spark over Apache Tez and MapReduce engines is because of faster data processing capability which comes with in-memory storage feature for intermediate results and reduces disk I/O operations overhead.
    - Once the data is generated using Spark, create a staging lookup table in Hive and load the data from Spark dataframe.
    - In Hive, create a final bucketed table with 8 buckets splitted on card_id and load the data from staging table with HBase storage handler so that the same data will be available in card lookup HBase table which can be used for real time processing.

DEPLOYMENT PLAN

PRE-REQUISITES:

1. Create an EMR cluster with **Hive, HBase, Sqoop, Spark, Zookeeper** applications installed in it as shown below:



Summary

ID: j-364WXN57AB11U
Creation date: 2022-12-17 23:28 (UTC+5:30)
End date: 2022-12-18 03:01 (UTC+5:30)
Elapsed time: 3 hours, 33 minutes
After last step completes: Cluster waits
Termination protection: Off
Tags: --
Master public DNS: ec2-18-208-137-125.compute-1.amazonaws.com
Connect to the Master Node Using SSH

Configuration details

Release label: emr-5.30.1
Hadoop distribution: Amazon 2.8.5
Applications: Hive 2.3.6, Hue 4.6.0, JupyterHub 1.1.0, Zeppelin 0.8.2, HBase 1.4.13, Sqoop 1.4.7, Spark 2.4.5, ZooKeeper 3.4.14, Livy 0.7.0
Log URI: s3://aws-logs-145313842850-us-east-1/elasticmapreduce/
EMRFS consistent view: Disabled
Custom AMI ID: --

Application user interfaces

Persistent user interfaces: Spark history server, YARN timeline server, Tez UI
On-cluster user interfaces: --

Network and hardware

Availability zone: us-east-1c
Subnet ID: subnet-0f80dc2e
Master: Terminated 1 m5.xlarge
Core: --
Task: --
Cluster scaling: Not enabled
Auto-termination: Not enabled

Applications such as Hue, JupyterHub, Zeppelin, Livy are optional and depends based on your preference.

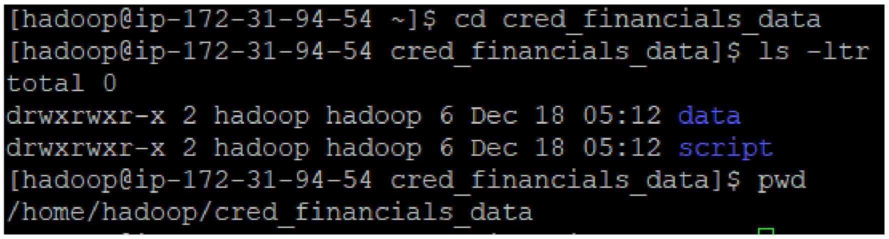
2. Download and install MySQL connector in AWS EMR for performing Sqoop data ingestion from AWS RDS to HDFS.
cd /home/hadoop
wget https://de-mysql-connector.s3.amazonaws.com/mysql-connector-java-8.0.25.tar.gz
tar -xvf mysql-connector-java-8.0.25.tar.gz

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

3. Download and install Hive-HBase handler jar file which will be used for card lookup data preparation for Hive and Hbase tables using PySpark. and place it in spark libraries folder. Also place some other required HBase jar files from HBase libraries to spark libraries folder.
   cd /home/hadoop
   wget https://repo1.maven.org/maven2/org/apache/hive/hive-hbase-handler/3.1.3/hive-hbase-handler-3.1.3.jar
   sudo cp hive-hbase-handler-3.1.3.jar /usr/lib/spark/jars/
   sudo cp /usr/lib/hbase/*.jar /usr/lib/spark/jars/
   sudo cp /usr/lib/hbase/lib/htrace-core-3.1.0-incubating.jar /usr/lib/spark/jars/
   sudo cp /usr/lib/hbase/lib/guava-12.0.1.jar /usr/lib/spark/jars/
   sudo cp /usr/lib/hbase/lib/metrics-core-2.2.0.jar /usr/lib/spark/jars/

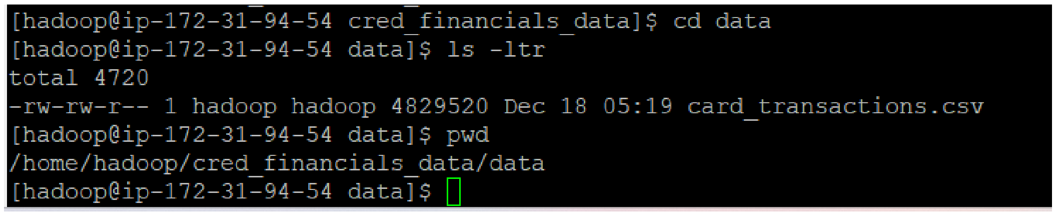
DEPLOYMENT STEPS:

1. Login to AWS EMR and create the required directories under home directory to place all the data and script files.
   cd /home/hadoop
   mkdir -p /home/hadoop/cred_financials_data/data
   mkdir -p /home/hadoop/cred_financials_data/script



```
[hadoop@ip-172-31-94-54 ~]$ cd cred_financials_data
[hadoop@ip-172-31-94-54 cred_financials_data]$ ls -ltr
total 0
drwxrwxr-x 2 hadoop hadoop 6 Dec 18 05:12 data
drwxrwxr-x 2 hadoop hadoop 6 Dec 18 05:12 script
[hadoop@ip-172-31-94-54 cred_financials_data]$ pwd
/home/hadoop/cred_financials_data
```

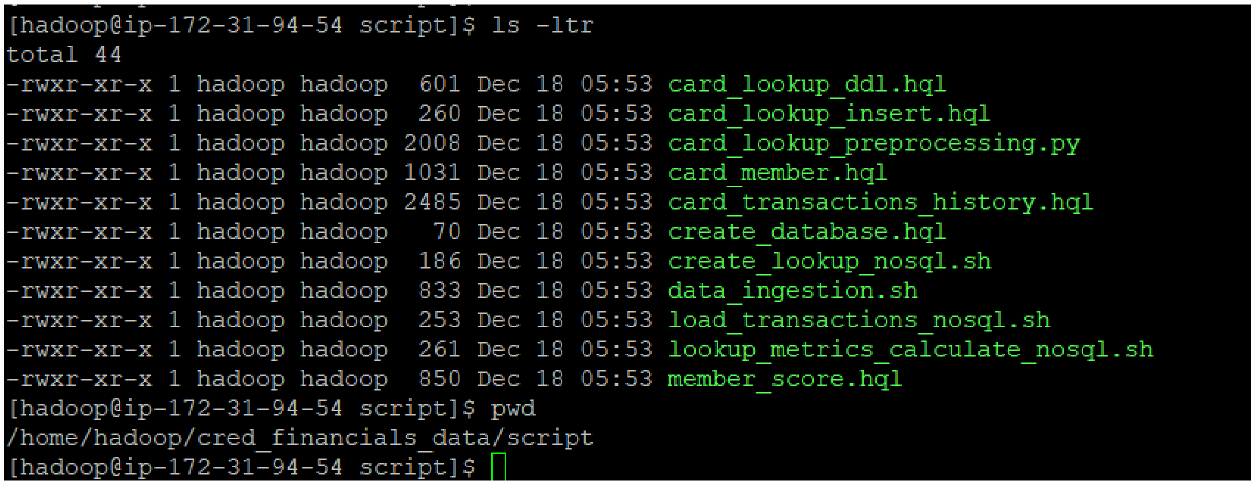
2. Place the following data and script execution files prepared for batch processing in the above created directories either using **WinSCP** tool or **scp** command and change the permissions to the script files.
   Data file:
   a. **card_transactions.csv:** This is the source card transactions history data provided in CSV format and needed to be loaded in NoSQL table (HBase). This file is placed inside data folder as shown below:



```
[hadoop@ip-172-31-94-54 cred_financials_data]$ cd data
[hadoop@ip-172-31-94-54 data]$ ls -ltr
total 4720
-rw-rw-r-- 1 hadoop hadoop 4829520 Dec 18 05:19 card_transactions.csv
[hadoop@ip-172-31-94-54 data]$ pwd
/home/hadoop/cred_financials_data/data
[hadoop@ip-172-31-94-54 data]$
```

Script files:

a. **data_ingestion.sh:** This script contains Sqoop commands to import card member and member score tables data from AWS RDS to HDFS
b. **create_database.hql:** This script creates **'cred_financials_data'** database in Hive.
c. **card_member.hql:** This script creates card member staging table on top of imported data from AWS RDS. Later it creates and loads the final bucketed table from staging table.
d. **member_score.hql:** This script creates member score staging table on top of imported data from AWS RDS. Later it creates and loads the final bucketed table from staging table.
e. **card_transactions_history.hql:** This script has three layers: source, staging and bucketed table (Hive-HBase integration). It starts with creating source table in Hive and load the data from card_transactions.csv file uploaded. Next it creates staging table performing type conversion on **transaction_dt** attribute. Finally, it creates bucketed Hive table with HBase storage handler and get loaded from staging table.
f. **load_transactions_nosql.sh:** This is a wrapper script to create card transactions table in HBase and Hive and load the historical transactions data into it.
g. **card_lookup_ddl.hql:** This script creates a card lookup bucketed table structure in Hive with HBase storage handler (Hive-HBase integration).
h. **create_lookup_nosql.sh:** This is a wrapper script to create card lookup table in HBase and Hive.
i. **card_lookup_preprocessing.py:** This is a PySpark script implemented with Spark SQL module to get and generate required metrics such as **card_id, ucl, postcode, transaction_dt and credit_score** from member score and card transaction hive tables and store the result in a staging table in Hive.
j. **card_lookup_insert.hql:** This script loads the card lookup data generated using PySpark code from staging to final bucketed table in Hive and Hbase.
k. **lookup_metrics_calculate_nosql.sh:** This is wrapper script to load card lookup data into HBase and Hive tables. It starts by executing PySpark code to generate the lookup data and then call card_lookup_insert.hql script to load it into card lookup tables in bucketed table (Hive-HBase).

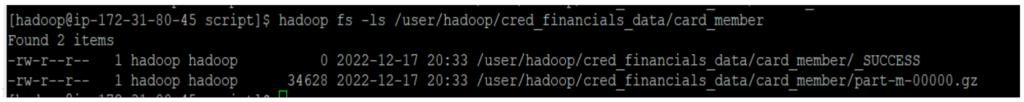


```
[hadoop@ip-172-31-94-54 script]$ ls -ltr
total 44
-rwxr-xr-x 1 hadoop hadoop  601 Dec 18 05:53 card_lookup_ddl.hql
-rwxr-xr-x 1 hadoop hadoop  260 Dec 18 05:53 card_lookup_insert.hql
-rwxr-xr-x 1 hadoop hadoop 2008 Dec 18 05:53 card_lookup_preprocessing.py
-rwxr-xr-x 1 hadoop hadoop 1031 Dec 18 05:53 card_member.hql
-rwxr-xr-x 1 hadoop hadoop 2485 Dec 18 05:53 card_transactions_history.hql
-rwxr-xr-x 1 hadoop hadoop   70 Dec 18 05:53 create_database.hql
-rwxr-xr-x 1 hadoop hadoop  186 Dec 18 05:53 create_lookup_nosql.sh
-rwxr-xr-x 1 hadoop hadoop  833 Dec 18 05:53 data_ingestion.sh
-rwxr-xr-x 1 hadoop hadoop  253 Dec 18 05:53 load_transactions_nosql.sh
-rwxr-xr-x 1 hadoop hadoop  261 Dec 18 05:53 lookup_metrics_calculate_nosql.sh
-rwxr-xr-x 1 hadoop hadoop  850 Dec 18 05:53 member_score.hql
[hadoop@ip-172-31-94-54 script]$ pwd
/home/hadoop/cred_financials_data/script
[hadoop@ip-172-31-94-54 script]$
```

Refer the scripts folder attached or the other documents such as **SqoopDataIngestion.pdf, CreateNoSQL.pdf, LoadNoSQL.pdf, PreAnalysis.pdf and ScriptsExecution.pdf** to know more about code logic.

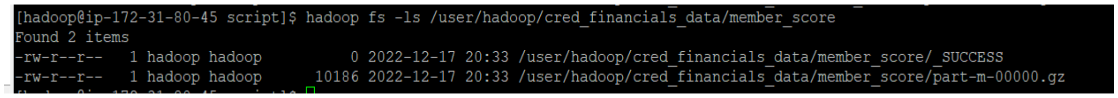
3. Import card member and member score data from AWS RDS to HDFS by running **data_ingestion.sh** script
   <u>Script execution command:</u>
   /home/hadoop/cred_financials_data/script/data_ingestion.sh upgradawsrds1.cyaielc9bmnf.us-east-1.rds.amazonaws.com cred_financials_data upgraduser upgraduser

   Card Member:

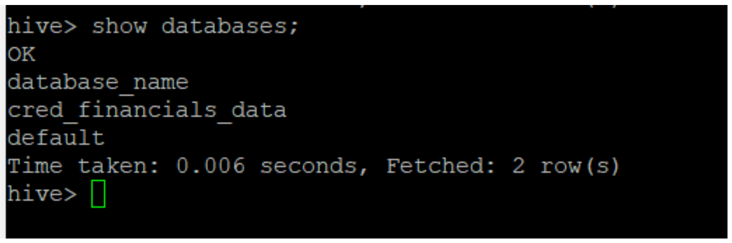
```
[hadoop@ip-172-31-80-45 script]$ hadoop fs -ls /user/hadoop/cred_financials_data/card_member
Found 2 items
-rw-r--r--   1 hadoop hadoop          0 2022-12-17 20:33 /user/hadoop/cred_financials_data/card_member/_SUCCESS
-rw-r--r--   1 hadoop hadoop      34628 2022-12-17 20:33 /user/hadoop/cred_financials_data/card_member/part-m-00000.gz
```

   Member Score:

```
[hadoop@ip-172-31-80-45 script]$ hadoop fs -ls /user/hadoop/cred_financials_data/member_score
Found 2 items
-rw-r--r--   1 hadoop hadoop          0 2022-12-17 20:33 /user/hadoop/cred_financials_data/member_score/_SUCCESS
-rw-r--r--   1 hadoop hadoop      10186 2022-12-17 20:33 /user/hadoop/cred_financials_data/member_score/part-m-00000.gz
```

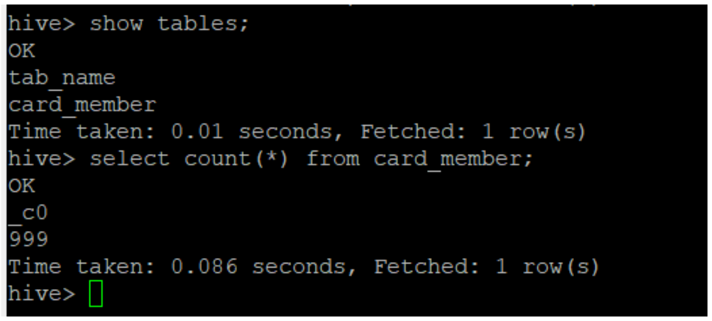
4. Create **'cred_financials_data'** database in Hive by running **create_database.hql** script.
   <u>Script execution command:</u>
   hive -f /home/hadoop/cred_financials_data/script/create_database.hql



```
hive> show databases;
OK
database_name
cred_financials_data
default
Time taken: 0.006 seconds, Fetched: 2 row(s)
hive> 
```

5. Create Hive table for card member on top of imported data by running **card_member.hql** script.
   <u>Script execution command:</u>
   hive -f /home/hadoop/cred_financials_data/script/card_member.hql

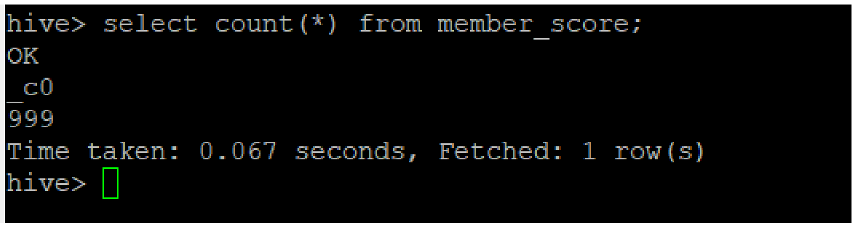
   Hive table:



```
hive> show tables;
OK
tab_name
card_member
Time taken: 0.01 seconds, Fetched: 1 row(s)
hive> select count(*) from card_member;
OK
_c0
999
Time taken: 0.086 seconds, Fetched: 1 row(s)
hive> 
```

6. Create Hive table for member score on top of imported data by running **member_score.hql** script.
   <u>Script execution command:</u>
   hive -f /home/hadoop/cred_financials_data/script/member_score.hql

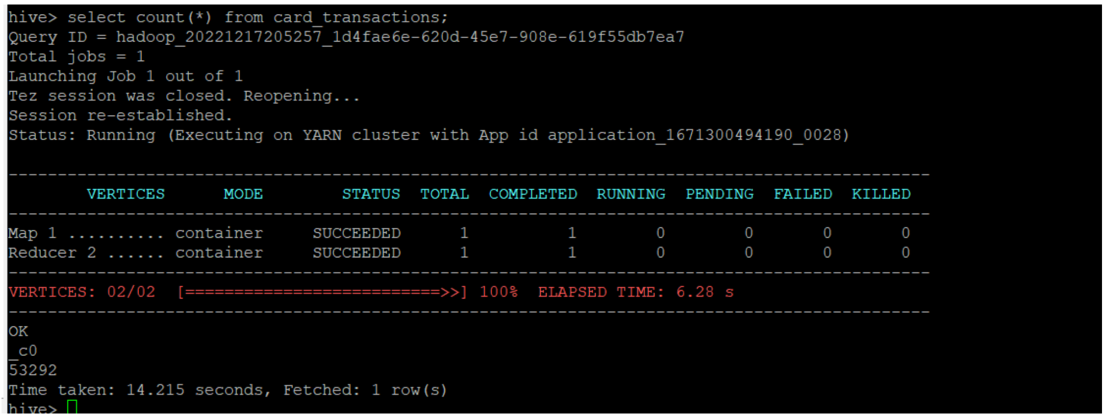
   Hive table:



```
hive> select count(*) from member_score;
OK
_c0
999
Time taken: 0.067 seconds, Fetched: 1 row(s)
hive> 
```

7. Create card transactions external table in HBase and Hive (Hive-HBase integration) and load the card transactions historical data into it by running **load_transactions_nosql.sh** wrapper script.
   <u>Script execution command:</u>
   /home/hadoop/cred_financials_data/script/load_transactions_nosql.sh

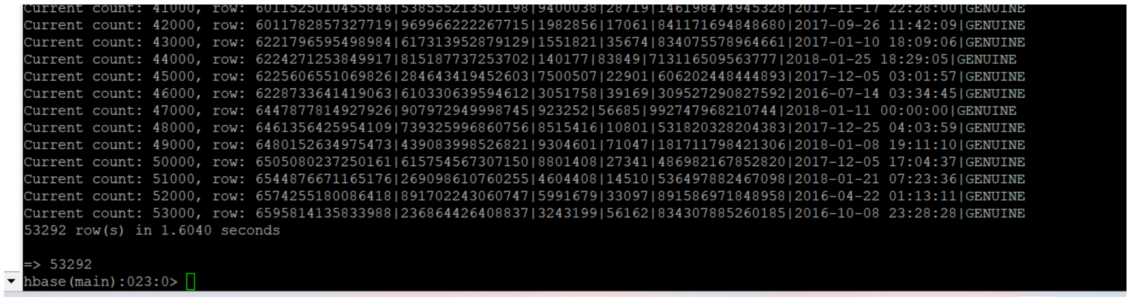
Hive table:



```
hive> select count(*) from card_transactions;
Query ID = hadoop_20221217205257_1d4fae6e-620d-45e7-908e-619f55db7ea7
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_1671300494190_0028)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 6.28 s
----------------------------------------------------------------------------------------------
OK
_c0
53292
Time taken: 14.215 seconds, Fetched: 1 row(s)
hive>
```

HBase table:

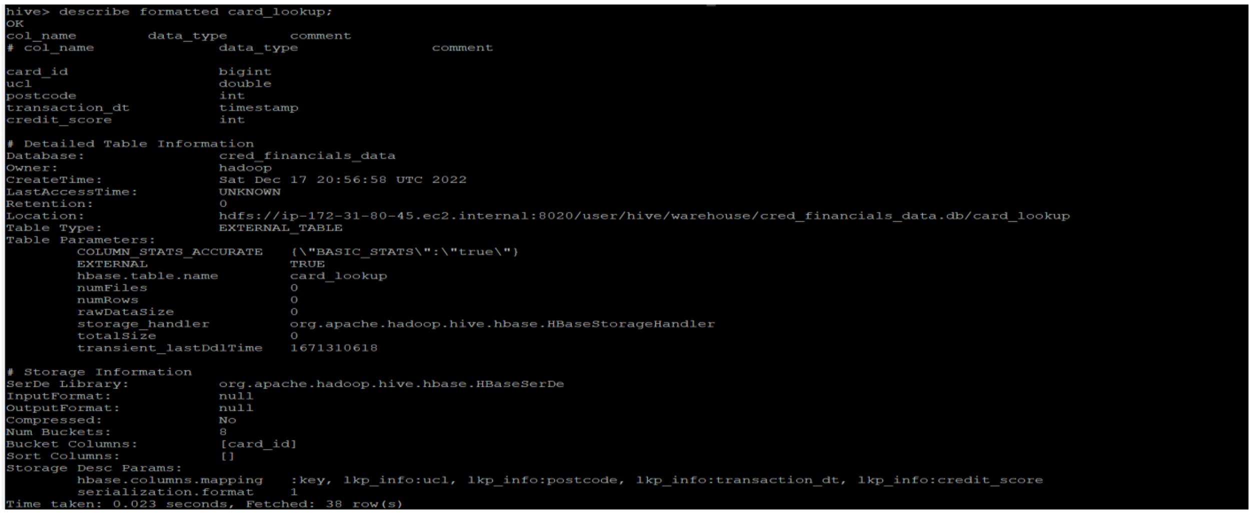


```
Current count: 41000, row: 6011525010455848|5385552135011198|9400038|28719|1461984749453328|2017-11-17 22:28:00|GENUINE
Current count: 42000, row: 6011782857327719|969966222267715|1982856|17061|841171694848680|2017-09-26 11:42:09|GENUINE
Current count: 43000, row: 6221796595498984|617313952879129|1551821|35674|834075578964661|2017-01-10 18:09:06|GENUINE
Current count: 44000, row: 6224271253849917|815187737253702|140177|83849|713116509563777|2018-01-25 18:29:05|GENUINE
Current count: 45000, row: 6225606551069826|284643419452603|7500507|22901|606202448444893|2017-12-05 03:01:57|GENUINE
Current count: 46000, row: 6228733641419063|610330639594612|3051758|39169|309527290827592|2016-07-14 03:34:45|GENUINE
Current count: 47000, row: 6447877814927926|907972949998745|923252|56685|992747968210744|2018-01-11 00:00:00|GENUINE
Current count: 48000, row: 6461356425954109|739325996860756|8515416|10801|531820328204383|2017-12-25 04:03:59|GENUINE
Current count: 49000, row: 6480152634975473|439083998526821|9304601|71047|181711798421306|2018-01-08 19:11:10|GENUINE
Current count: 50000, row: 6505080237250161|615754567307150|8801408|27341|486982167852820|2017-12-05 17:04:37|GENUINE
Current count: 51000, row: 6544876671165176|269098610760255|4604408|14510|536497882467098|2018-01-21 07:23:36|GENUINE
Current count: 52000, row: 6574255180086418|891702243060747|5991679|33097|891586971848958|2016-04-22 01:13:11|GENUINE
Current count: 53000, row: 6595814135833988|236864426408837|3243199|56162|834307885260185|2016-10-08 23:28:28|GENUINE
53292 row(s) in 1.6040 seconds

=> 53292
hbase(main):023:0>
```

8. Create card lookup table in HBase and Hive (Hive-HBase integration) by running **create_lookup_nosql.sh** wrapper script
<u>**Script execution command:**</u>
/home/hadoop/cred_financials_data/script/create_lookup_nosql.sh

Hive table:



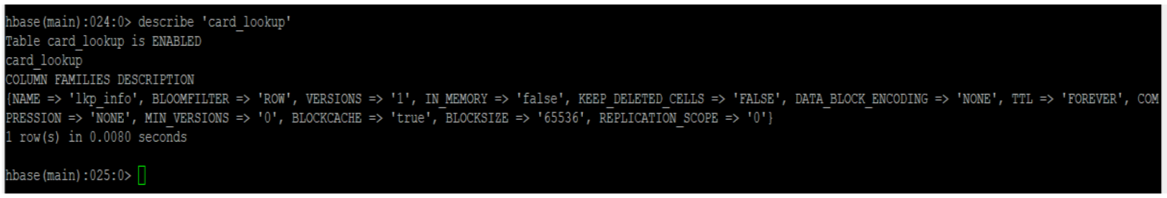
```
hive> describe formatted card_lookup;
OK
col_name        data_type       comment
# col_name              data_type               comment

card_id                 bigint
ucl                     double
postcode                int
transaction_dt          timestamp
credit_score            int

# Detailed Table Information
Database:               cred_financials_data
Owner:                  hadoop
CreateTime:             Sat Dec 17 20:56:58 UTC 2022
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-80-45.ec2.internal:8020/user/hive/warehouse/cred_financials_data.db/card_lookup
Table Type:             EXTERNAL_TABLE
Table Parameters:
        COLUMN_STATS_ACCURATE   {\"BASIC_STATS\":\"true\"}
        EXTERNAL                TRUE
        hbase.table.name        card_lookup
        numFiles                0
        numRows                 0
        rawDataSize             0
        storage_handler         org.apache.hadoop.hive.hbase.HBaseStorageHandler
        totalSize               0
        transient_lastDdlTime   1671310618

# Storage Information
SerDe Library:          org.apache.hadoop.hive.hbase.HBaseSerDe
InputFormat:            null
OutputFormat:           null
Compressed:             No
Num Buckets:            8
Bucket Columns:         [card_id]
Sort Columns:           []
Storage Desc Params:
        hbase.columns.mapping   :key, lkp_info:ucl, lkp_info:postcode, lkp_info:transaction_dt, lkp_info:credit_score
        serialization.format    1
Time taken: 0.023 seconds, Fetched: 38 row(s)
```

HBase table:

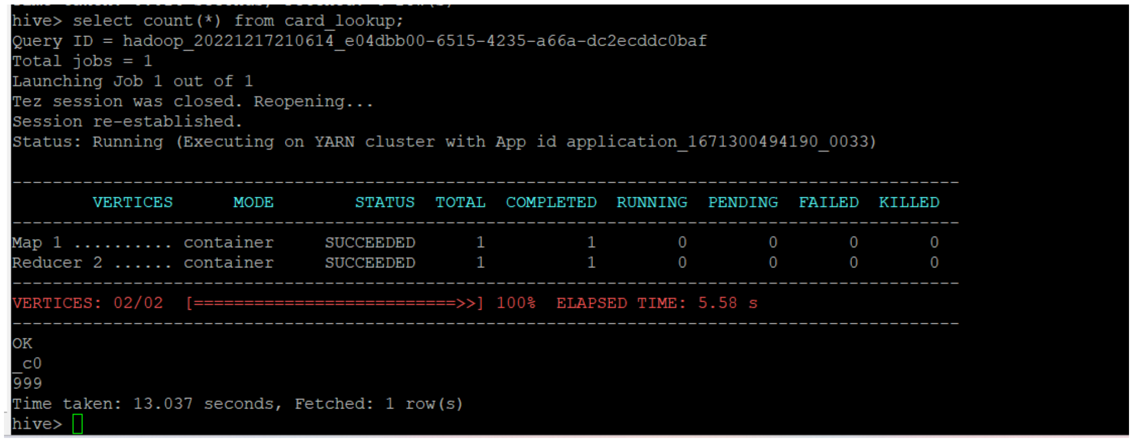


```
hbase(main):024:0> describe 'card_lookup'
Table card_lookup is ENABLED
card_lookup
COLUMN FAMILIES DESCRIPTION
{NAME => 'lkp_info', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COM
PRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
1 row(s) in 0.0080 seconds

hbase(main):025:0>
```

9. Prepare the card lookup data from member score and card transactions hive tables and load it into Hive and HBase tables by running **lookup_metrics_calculate_nosql.sh** wrapper script
   <u>Script execution command:</u>
   /home/hadoop/cred_financials_data/script/lookup_metrics_calculate_nosql.sh

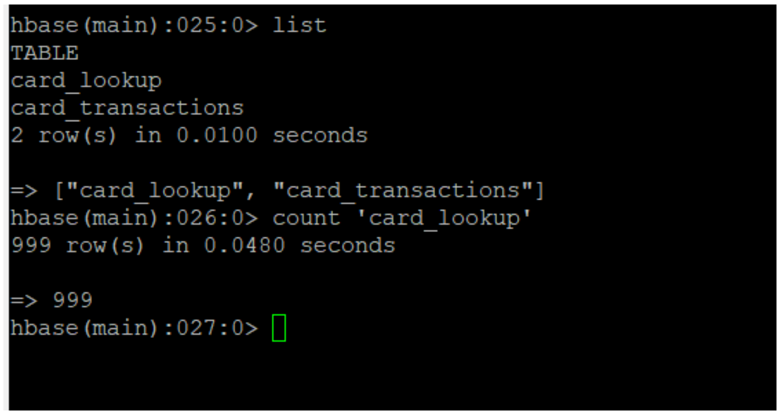
Hive table:



```
hive> select count(*) from card_lookup;
Query ID = hadoop_20221217210614_e04dbb00-6515-4235-a66a-dc2ecddc0baf
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_1671300494190_0033)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 5.58 s
----------------------------------------------------------------------------------------------
OK
_c0
999
Time taken: 13.037 seconds, Fetched: 1 row(s)
hive>
```

HBase table:



```
hbase(main):025:0> list
TABLE
card_lookup
card_transactions
2 row(s) in 0.0100 seconds

=> ["card_lookup", "card_transactions"]
hbase(main):026:0> count 'card_lookup'
999 row(s) in 0.0480 seconds

=> 999
hbase(main):027:0>
```

After successful deployment of all the above steps we are good to consume real time transactions streaming data from Kafka server and perform analysis on top of it by using the pre-processed card lookup data.