Scripts Execution

Following is the sequence of steps executed in order. We have explained steps till task 4 only.

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

1. Create and upload the Transactions file first to hadoop cluster in order to upload to NoSQL database.
   Command to do execute:
   hadoop fs -mkdir /user/capstone
   hadoop fs -put card_transactions.csv /user/capstone/card_transaction

2. We have selected the hive database so on hadoop cluster connect to hive.
   a. User command 'hive'
      Create database capstone;
      Use capstone;

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

3. Create table using command:

```
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/capstone/card_transaction'
TBLPROPERTIES ("skip.header.line.count"="1");
```

4. Create another table to handle date time datatyping issue and move from external table to internal table.

```
CREATE TABLE IF NOT EXISTS transactions_formatted (
`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. Move data using script:

```
INSERT OVERWRITE TABLE transactions_formatted
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. Next step is to do Sqoop job to import Member score and card member data from AMS RDS job to required location.

   a. Create the folders for these jobs using
      i. hadoop fs -mkdir /user/capstone/card_member
      ii. hadoop fs -mkdir /user/capstone/member_score

   b. Set hive parameter and snappy config paramters for fast performance:

```
Set Hive parameters:
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;
```

   c. Sqoop Job Commands:

```
sqoop import --connect jdbc:mysql://upgradawsrds1.cyaielc9bmnf.us-east-
1.rds.amazonaws.com:3306/cred_financials_data --username upgraduser --password
upgraduser --table member_score --null-string 'NA' --null-non-string '\\N' --delete-target-dir --
target-dir '/user/capstone/member_score' -m 1

sqoop import --connect jdbc:mysql://upgradawsrds1.cyaielc9bmnf.us-east-
1.rds.amazonaws.com:3306/cred_financials_data --username upgraduser --password
upgraduser --table card_member --null-string 'NA' --null-non-string '\\N' --delete-target-dir --
target-dir '/user/capstone/card_member'  -m 1
```

7. Create hive tables from the loaded tables from the sqoop data Ingestions results:
   a. Member_score

```
CREATE EXTERNAL TABLE IF NOT EXISTS member_score(
MEMBER_ID String,
score String)
```

```
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
LOCATION '/user/capstone/member_score';
```

b. Card_member table:

```
CREATE EXTERNAL TABLE IF NOT EXISTS card_member(
card_id string,
MEMBER_ID string,
score string,
tr_date string,
exp_date string,
country string,
area string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
LOCATION '/user/capstone/card_member';
```

Verify the record count in both the tables. These are 999.

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

8. Create main Lookup table: Added HBASE linkage in case we need to use dao script in future for kafka processing logic.

```
 CREATE TABLE card_member_lookup
(CARD_ID  STRING,
UCL DOUBLE,
POSTCODE STRING,
TRANSACTION_DT TIMESTAMP,
SCORE INT
)
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");
```

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

9. Load the Lookup table using the command:

```
 INSERT OVERWRITE TABLE card_member_lookup
SELECT trans.card_id,
```

```
      trans.moving_average+3*standard_deviation as UCL,
      POSTCODE,
      transaction_dt,
      member_score.score
FROM
(
SELECT
  card_id,
  AVG(amount)
       OVER(PARTITION BY card_id ORDER BY transaction_dt ROWS BETWEEN 9
PRECEDING AND CURRENT ROW)
       AS moving_average,
  STDDEV(amount)
       OVER(PARTITION BY card_id ORDER BY transaction_dt ROWS BETWEEN 9
PRECEDING AND CURRENT ROW)
       AS standard_deviation,
   transaction_dt,
   POSTCODE,
   ROW_NUMBER() OVER(PARTITION BY card_id ORDER BY transaction_dt DESC ) RN
FROM transactions_formatted
WHERE STATUS = 'GENUINE'
)trans
inner JOIN card_member on (trans.card_id=card_member.card_id)
inner JOIN member_score on (member_score.MEMBER_ID=card_member.MEMBER_ID)
WHERE RN=1;
```