Scripts Execution

Explanation of the solution to the batch layer problem

1. Created EMR with Hadoop, Hive, Hbase, Hcatalog, Spark, Hue and Sqoop.
2. Moved the Historical Data(card_transactions.csv) using WinSCP to Hadoop.
3. Created HDFS directory and moved data from local EC2 to HDFS so Hive & HBase can access it.

```
hadoop fs -mkdir /user/CCFD_project
```

```
hadoop fs -put /home/hadoop/card_transactions.csv
/user/CCFD_project /card_transactions.csv
```

4. Creating Hive table for card_transactions and loading historical data into it.

   Code to be run in Hive shell:

   i. Create and use the database.

   ```
   create database ccfd;
   use ccfd;
   ```

   ii. Create external table card_transactions_ext pointing to HDFS path.

   ```
   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/CCFD_Project/card_transactions.csv'
   TBLPROPERTIES ("skip.header.line.count"="1");
   ```

iii. Create table card_transactions_orc

```
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;
```

Load data in card_transactions_ext and checking the data.

```
LOAD DATA INPATH '/user/CCFD_project/card_transactions.csv’
INTO TABLE card_transactions_ext;

select count(*) from card_transactions_ext;
```

iv. Inserting data into ORC table

```
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;
```

v. Verifying transaction_dt and year in card_transactions_orc

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

vi. Creating card_transactions_hbase hive-hbase integrated 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");
```

vii. Loading data in card_transactions_hbase which will be visible in HBase as well.

```
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;
```

viii. Check some data in card_transactions_hbase

```
select * from card_transactions_hbase limit 10;
```

ix. Creating lookup_data_hbase hive-hbase integrated table which will be visible in HBase as well.

```
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");
```

x. Validating lookup table:

```
Describe lookup_data_Hbase
```

5. HBase Shell commands:

i. Checking the details of card_transactions_hive hive-hbase integrated table.

```
describe 'card_transactions_hive'
```

ii. Checking the count in card_transactions_hive in Hbase.

```
count 'card_transactions_hive'
```

iii. Checking the details of lookup_data_hive hive-hbase integrated table.

```
describe 'lookup_data_hive'
```

iv. Altering the lookup_data_hive table and set VERSIONS to 10 for lookup_transaction_family
[This will enable multiple degree of versioning, keeping the track of last 10 records.]

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

v. Confirming details of lookup_data_hive and confirm that VERSIONS is set to 10 for lookup_transaction_family

```
describe 'lookup_data_hive'
```

6. Importing AWS RDS data:

   i. First, we will install MySQL connector before starting with Apache Sqoop.

```
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/
```

   ii. Scoop import for card member table:

```
sqoop import \
--connect jdbc:mysql://upgradawsrds1.cyaielc9bmnf.us-east-
1.rds.amazonaws.com/cred_financials_data \
--username upgraduser \
--password upgraduser \
--table card_member \
--target-dir /user/CCFD_project/card_member \
-m 1
```

   iii. Scoop import for member score table:

```
sqoop import \
--connect jdbc:mysql://upgradawsrds1.cyaielc9bmnf.us-east-
1.rds.amazonaws.com/cred_financials_data \
--username upgraduser \
--password upgraduser \
--table member_score \
--target-dir /user/CCFD_project/member_score \
-m 1
```

iv. Checking the files:

```
hadoop fs -ls /user/CCFD_project/card_member
```

```
hadoop fs -ls /user/CCFD_project/member_score
```

7. Creating external tables for RDS data:

```
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/CCFD_project/card_member';
```

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

8. Creating ORC tables from the external tables:

```
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");
```

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

9. Inserting data into ORC tables:

```
INSERT OVERWRITE TABLE CARD_MEMBER_ORC
COUNTRY, CITY FROM CARD_MEMBER_EXT;
```

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

10. Calculating moving average and standard deviation of last 10 transactions:

Create 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");
```

Create 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");
```

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.TRANSACTION_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;
```

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;
```

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 CMS
ON RCTO.CARD_ID = CMS.CARD_ID
WHERE RCTO.RANK = 1;
```

Verifying the count in lookup_data_hbase table.

```
SELECT COUNT(*) FROM LOOKUP_DATA_HBASE;
```

Verifying same data in lookup_data_hbase table.

```
SELECT * FROM LOOKUP_DATA_HBASE LIMIT 10;
```

11. Checking the data in HBase:

Checking count in lookup_data_hive table.

```
count 'lookup_data_hive'
```

Checking data in lookup_data_hive table.

```
scan 'lookup_data_hive'
```