Scripts Execution

Screenshots of the execution of the scripts written

As part of the project, broadly, you are required to perform the following tasks:
Task 1: Load the transactions history data (card_transactions.csv) in a NoSQL database.

1. We have pushed the card_transactions.csv file to the S3
2. Once we are inside the server, we will push the csv from the S3 bucket to the hdfs folder using the following.

```
[hadoop@ip-10-0-7-47 ~]$ aws s3 cp s3://capstonecreditcard/card_transactions.csv .
download: s3://capstonecreditcard/card_transactions.csv to ./card_transactions.csv
```

3. We will store the card_transactions data into the HBase DB using happy base

```
[hadoop@ip-10-0-7-47 ~]$ python csv_transaction.py
Connected to  localhost
table created : card_transactions
Data inserted: card_transactions
[connection closed
```

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

1. Run the scoop code for transfer the data to HDFS
    i. Card_member

```
[hadoop@ip-10-0-7-47 ~]$ sqoop import \
> --connect jdbc:mysql://upgradawsrds1.cyaielc9bmnf.us-east-1.rds.amazonaws.com/cred_financials_data \
> --table card_member \
> --username upgraduser \
> --password upgraduser \
> --target-dir /user/hadoop/card_member \
> --m 1
```

    ii. Member_score

```
[hadoop@ip-10-0-7-47 ~]$ sqoop import \
> --connect jdbc:mysql://upgradawsrds1.cyaielc9bmnf.us-east-1.rds.amazonaws.com/cred_financials_data \
> --table member_score \
> --username upgraduser \
> --password upgraduser \
> --target-dir /user/hadoop/member_score \
> --m 1
```

2. Check if the data is loaded
    i. Card_member

```
[hadoop@ip-10-0-7-47 ~]$ hadoop fs -cat /user/hadoop/card_member/part-m-00000 |head
340028465709212,009250698176266,2012-02-08 06:04:13.0,05/13,United States,Barberton
340054675199675,835873341185231,2017-03-10 09:24:44.0,03/17,United States,Fort Dodge
340082915339645,512969555857346,2014-02-15 06:30:30.0,07/14,United States,Graham
340134186926007,887711945571282,2012-02-05 01:21:58.0,02/13,United States,Dix Hills
340265728490548,680324265406190,2014-03-29 07:49:14.0,11/14,United States,Rancho Cucamonga
340268219434811,929799084911715,2012-07-08 02:46:08.0,08/12,United States,San Francisco
340379737226464,089615510858348,2010-03-10 00:06:42.0,09/10,United States,Clinton
340383645652108,181180599313885,2012-02-24 05:32:44.0,10/16,United States,West New York
340803866934451,417664728506297,2015-05-21 04:30:45.0,08/17,United States,Beaverton
340889618969736,459292914761635,2013-04-23 08:40:11.0,11/15,United States,West Palm Beach
```

    ii. Member Score

```
[hadoop@ip-10-0-7-47 ~]$ hadoop fs -cat /user/hadoop/member_score/part-m-00000 |head
000037495066290,339
000117826301530,289
001147922084344,393
001314074991813,225
001739553947511,642
003761426295463,413
004494068832701,217
006836124210484,504
006991872634058,697
007955566230397,372
```

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

```
import happybase
import uuid

connection=happybase.Connection('localhost')

connection.open()
print('Connected to ', connection.host)

try:
    connection.create_table('lookup',
    {
        'card_info': dict(),
        'transaction_info':dict()
        })
    print('table created : lookup')

except Exception as e :
    print(e)

    # close connection to the HBASe
connection.close()
print('connection closed')
```

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

For loading the data to the lookup table we will need the

i. csv_transactions hbase table from hbase
ii. Member_score table from HDFS
iii. card_member table from HDFS

We will create hive tables for each of the tables and then merge to create the required query for the hive table

1. Create and use relevant Databse:

```
hive> create database capstone;
OK
Time taken: 2.623 seconds
hive> use capstone;
OK
Time taken: 0.068 seconds
```

2. Create Member_Score internal table

```
hive> create table if not exists member_score (
    >         member_id BIGINT,
    >         score SMALLINT)
    > row format delimited
    > fields terminated by ','
    > lines terminated by '\n'
    > stored as textfile;
OK
Time taken: 0.976 seconds
```

3. load the data to the Member_Score table

```
hive> load data inpath '/user/hadoop/member_score' into table member_score;
Loading data to table capstone.member_score
OK
Time taken: 2.159 seconds
```

4. Check the Member_Score data

```
hive> select * from member_score limit 5;
OK
member_score.member_id  member_score.score
37495066290     339
117826301530    289
1147922084344   393
1314074991813   225
1739553947511   642
Time taken: 0.192 seconds, Fetched: 5 row(s)
```

5. Create card_member internal table

```
hive> create table if not exists card_member (
    >     card_id BIGINT,
    >     member_id BIGINT,
    >     member_joining_dt TIMESTAMP,
    >     card_purchase_dt varchar(10),
    >     country varchar(50),
    >     city varchar(50)     )
    > row format delimited
    > fields terminated by ','
    > lines terminated by '\n'
    > stored as textfile;
OK
Time taken: 0.088 seconds
```

6. load the data to the card_member table

```
hive> load data inpath '/user/hadoop/card_member' into table card_member;
Loading data to table capstone.card_member
OK
Time taken: 0.41 seconds
```

7. Check the card_member data

```
hive> select * from card_member limit 5;
OK
card_member.card_id     card_member.member_id   card_member.member_joining_dt   card_member.card_purchase_dt    card_member.country     card_member.city
340028465709212 9250698176266   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
Time taken: 0.174 seconds, Fetched: 5 row(s)
```

8. Create an external table which will link to the card_transactions in hbase bd

```
hive> create external table if not exists ext_past_transaction (
    >     key varchar(100),
    >     card_id BIGINT,
    >     member_id BIGINT,
    >     amount INT,
    >     postcode INT,
    >     pos_id BIGINT,
    >      transaction_dt varchar(50),
    >     status varchar(50)
    >  )
    > row format delimited
    > fields terminated by ','
    > lines terminated by '\n'
    > Stored by 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    > With serdeproperties
    > ("hbase.columns.mapping"=':key,\
    > card_info:card_id,\
    > member_info:member_id,\
    > transaction_info:amount,\
    > transaction_info:postcode,\
    > transaction_info:pos_id,\
    > transaction_info:transaction_dt,\
    > transaction_info:status')
    >      TBLPROPERTIES("hbase.table.name"="card_transactions");
OK
Time taken: 1.044 seconds
```

9. Check the card_transactions data

```
hive> select * from ext_past_transaction limit 5;
OK
ext_past_transaction.key        ext_past_transaction.card_id    ext_past_transaction.member_id  ext_past_transaction.amount     ext_past_transaction.postcode   ext_past_transaction.pos_id     ext_past_transaction.transaction_dt     ext_past_transaction.status
00,000
0ia     348702330256514 37495066290     9084849 33946   614677375609919 11-02-2018 00:00:00     GENUINE
00 :0000
0ia     348702330256514 37495066290     330148  33946   614677375609919 11-02-2018 00:00:00     GENUINE
000(0000
0ia     348702330256514 37495066290     136052  33946   614677375609919 11-02-2018 00:00:00     GENUINE
000000
0ia     348702330256514 37495066290     4310362 33946   614677375609919 11-02-2018 00:00:00     GENUINE
0(00000
0ia     348702330256514 37495066290     9097094 33946   614677375609919 11-02-2018 00:00:00     GENUINE
Time taken: 0.536 seconds, Fetched: 5 row(s)
```

10. Create an external table which will link to the lookup table in hbase bd

```
hive> create external table if not exists ext_lookup (
    >
    >     Card_id BIGINT,
    >     Upper_control_limit INT,
    >     last_Postcode INT,
    >     last_Transaction_dt timestamp,
    >     credit_score smallint
    >  )
    > row format delimited
    > fields terminated by ','
    > lines terminated by '\n'
    > Stored by 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    > With serdeproperties
    > ("hbase.columns.mapping"=':key,\
    > transaction_info:Upper_control_limit,\
    > transaction_info:last_Postcode,\
    > transaction_info:last_Transaction_dt,\
    > transaction_info:credit_score')
    >      TBLPROPERTIES("hbase.table.name"="lookup");
OK
Time taken: 0.161 seconds
```

11. Insert the data to the lookup table

```
hive> insert into ext_lookup
    > (card_id, upper_control_limit, last_postcode, last_transaction_dt, credit_score)
    >
    > select card_id,
    > avg+ (3*std) as UCL ,
    > postcode,
    > from_unixtime(transaction_dt, 'yyyy-MM-dd HH:mm:ss') ,
    >  score
    > from
    > (select * from
    > (select card_id,
    > member_id,
    > postcode,
    > unix_timestamp(transaction_dt, 'dd-MM-yyyy HH:mm:ss') as transaction_dt,
    > row_number() over (partition by card_id order by unix_timestamp(transaction_dt, 'dd-MM-yyyy HH:mm:ss') desc) as row_num,
    > AVG(amount) OVER( partition by card_id ORDER BY unix_timestamp(transaction_dt, 'dd-MM-yyyy HH:mm:ss') desc ROWS BETWEEN CURRENT ROW and 9 following ) as avg,
    > STD(amount) OVER( partition by card_id ORDER BY unix_timestamp(transaction_dt, 'dd-MM-yyyy HH:mm:ss') desc ROWS BETWEEN CURRENT ROW and 9 following ) as std
    > from ext_past_transaction
    > where status='GENUINE'
    > ) moving_avg
    >  where row_num=1) temp_query
    >
    > left outer join member_score
    > on temp_query.member_id=member_score.member_id;
Query ID = hadoop_20230116141848_a24623f7-df37-4fc5-a3a8-ed0f3bc7f8a9
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1673877695623_0003)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Map 3 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 14.29 s
--------------------------------------------------------------------------------
OK
_col0   _col1   _col2   _col3   _col4
Time taken: 18.215 seconds
```

12. Check the data for the lookup table

```
hive> select * from ext_lookup limit 5;
OK
ext_lookup.card_id      ext_lookup.upper_control_limit  ext_lookup.last_postcode        ext_lookup.last_transaction_dt  ext_lookup.credit_score
340028465709212 16331555        24658   2018-01-02 03:25:35     233
340054675199675 14156079        50140   2018-01-15 19:43:23     631
340082915339645 15285685        17844   2018-01-26 19:03:47     407
340134186926007 15239767        67576   2018-01-18 23:12:50     614
340265728490548 16084916        72435   2018-01-21 02:07:35     202
Time taken: 0.167 seconds, Fetched: 5 row(s)
```

13. Cross verify using the hbase command

```
hbase(main):002:0> scan 'lookup'
ROW                             COLUMN+CELL
 340028465709212                column=transaction_info:Upper_control_limit, timestamp=1673878745054, value=16331555
 340028465709212                column=transaction_info:credit_score, timestamp=1673878745054, value=233
 340028465709212                column=transaction_info:last_Postcode, timestamp=1673878745054, value=24658
 340028465709212                column=transaction_info:last_Transaction_dt, timestamp=1673878745054, value=2018-01-02 03
                                :25:35
 340054675199675                column=transaction_info:Upper_control_limit, timestamp=1673878746706, value=14156079
 340054675199675                column=transaction_info:credit_score, timestamp=1673878746706, value=631
 340054675199675                column=transaction_info:last_Postcode, timestamp=1673878746706, value=50140
 340054675199675                column=transaction_info:last_Transaction_dt, timestamp=1673878746706, value=2018-01-15 19
                                :43:23
 340082915339645                column=transaction_info:Upper_control_limit, timestamp=1673878746706, value=15285685
 340082915339645                column=transaction_info:credit_score, timestamp=1673878746706, value=407
 340082915339645                column=transaction_info:last_Postcode, timestamp=1673878746706, value=17844
 340082915339645                column=transaction_info:last_Transaction_dt, timestamp=1673878746706, value=2018-01-26 19
                                :03:47
 340134186926007                column=transaction_info:Upper_control_limit, timestamp=1673878746706, value=15239767
 340134186926007                column=transaction_info:credit_score, timestamp=1673878746706, value=614
 340134186926007                column=transaction_info:last_Postcode, timestamp=1673878746706, value=67576
 340134186926007                column=transaction_info:last_Transaction_dt, timestamp=1673878746706, value=2018-01-18 23
                                :12:50
 340265728490548                column=transaction_info:Upper_control_limit, timestamp=1673878746706, value=16084916
 340265728490548                column=transaction_info:credit_score, timestamp=1673878746706, value=202
 340265728490548                column=transaction_info:last_Postcode, timestamp=1673878746706, value=72435
 340265728490548                column=transaction_info:last_Transaction_dt, timestamp=1673878746706, value=2018-01-21 02
                                :07:35
```