Scripts Execution

Explanation of the solution to the batch layer problem

Ingest the relevant data from AWS RDS to Hadoop.

Table 1: Card member

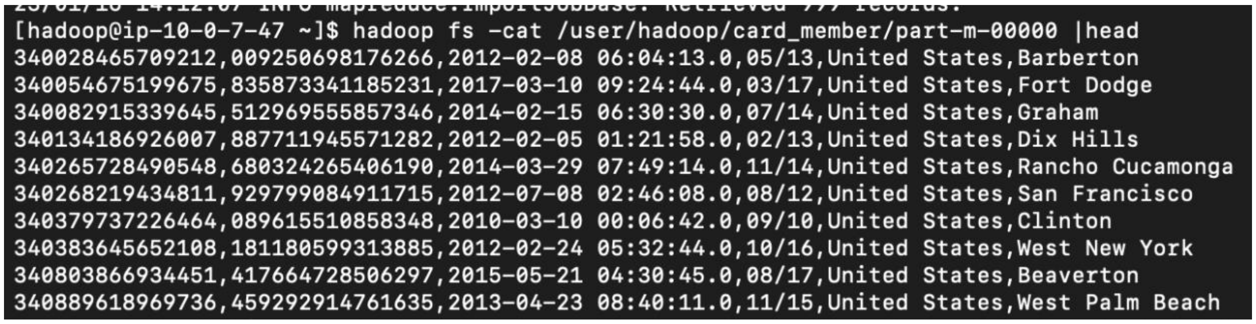
```
sqoop import \
--connect jdbc:mysql://upgradawsrds1.cyaielc9bmnf.us-east-1.rds.amazonaws.com/cred_financials_data \
--table card_member \
--username upgraduser \
--password upgraduser \
--target-dir /user/hadoop/card_member \
--m 1
```

Table 2: member_score

```
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
```

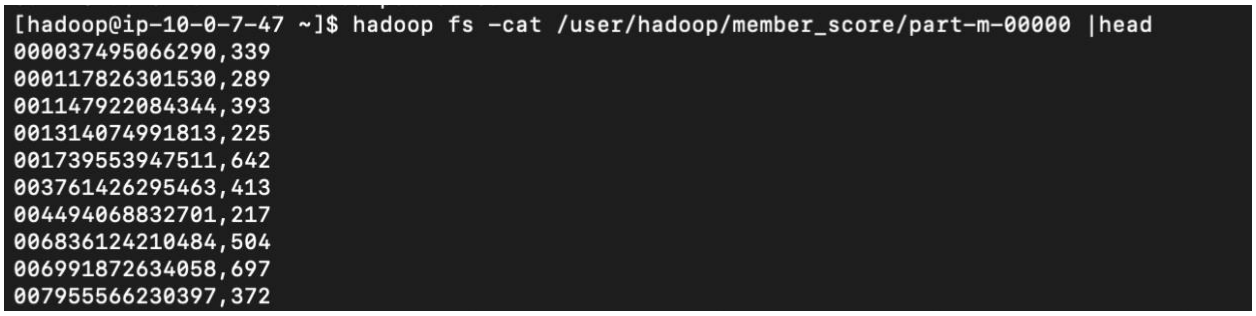
Check if the data is loaded

```
hadoop fs -cat /user/hadoop/card_member/part-m-00000 |head
```



```
[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
```

```
hadoop fs -cat /user/hadoop/member_score/part-m-00000 |head
```



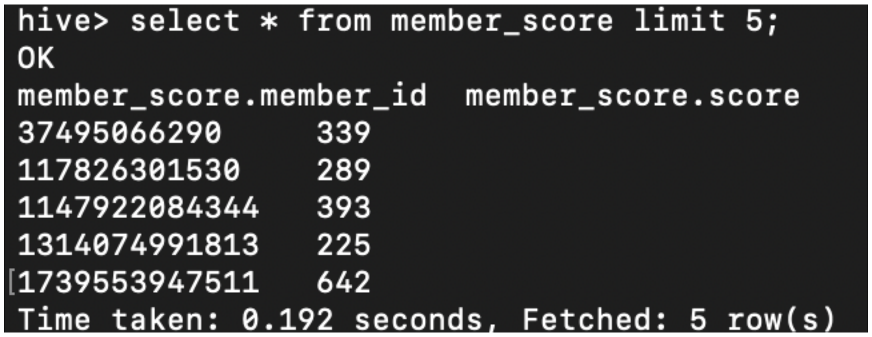
```
[hadoop@ip-10-0-7-47 ~]$ hadoop fs -cat /user/hadoop/member_score/part-m-00000 |head
000037495066290,339
000117826301530,289
001147922084344,393
001314074991813,225
001739553947511,642
003761426295463,413
004494068832701,217
006836124210484,504
006991872634058,697
007955566230397,372
```

Creating member score table in 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;
```

Loading member score data to HIVE:

```
load data inpath '/user/hadoop/member_score' into table member_score;
```



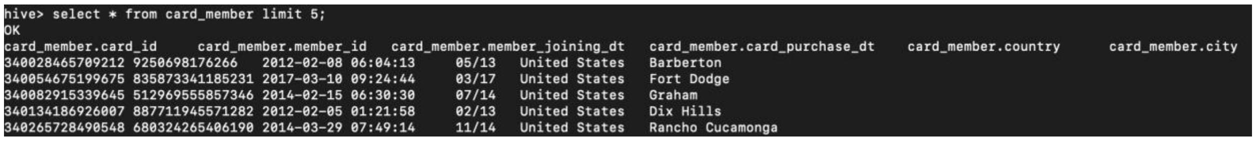
```
hive> select * from member_score limit 5;
OK
member_score.member_id  member_score.score
37495066290     339
117826301530    289
1147922084344   393
1314074991813   225
1739553947511   642
Time taken: 0.192 seconds, Fetched: 5 row(s)
```

Creating card member table in 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;
```

Loading card member data to HIVE:

```
load data inpath '/user/hadoop/card_member' into table card_member;
```



```
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
```

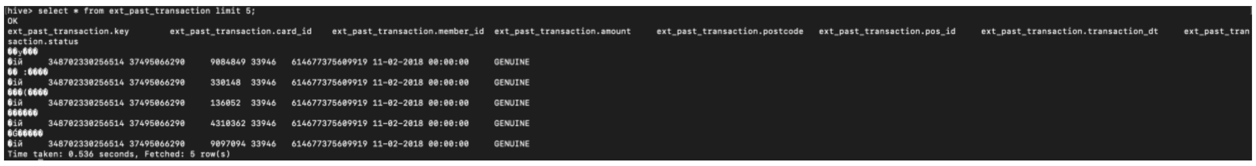
Command to load card_transactions.csv to HBASE Database:

```
import happybase
import uuid

connection=happybase.Connection('localhost')

connection.open()
print('Connected to ', connection.host)
try:
    connection.create_table('card_transactions',
    {
        'card_info': dict(),
        'member_info':dict(),
        'transaction_info':dict()
        })

    print('table created : card_transactions')

except Exception as e :
    print(e)
    connection.close()
    print('connection closed')

card_transactions=connection.table('card_transactions')

# put data to the table
with open('card_transactions.csv', 'r') as data:
    header= data.readline()
    lines= data.readlines()
    for line in lines:
        line=line.strip()
        line= line.split(",")
        card_transactions.put(uuid.uuid1().bytes, {'card_info:card_id': line[0],
        'member_info:member_id': line[1],
        'transaction_info:amount': line[2],
        'transaction_info:postcode': line[3],
        'transaction_info:pos_id': line[4],
        'transaction_info:transaction_dt': line[5],
        'transaction_info:status': line[6]
        })
print('Data inserted: card_transactions')

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

Creating transaction table in HIVE which is linked to the hbase db.

```
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");
```

Check for the data if loaded properly:



```
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_tran
saction.status
��y���
�iй     348702330256514 37495066290     9084849 33946   614677375609919 11-02-2018 00:00:00     GENUINE
�� :����
�iй     348702330256514 37495066290     330148  33946   614677375609919 11-02-2018 00:00:00     GENUINE
���(����
�iй     348702330256514 37495066290     136052  33946   614677375609919 11-02-2018 00:00:00     GENUINE
������
�iй     348702330256514 37495066290     4310362 33946   614677375609919 11-02-2018 00:00:00     GENUINE
�G�����
�iй     348702330256514 37495066290     9097094 33946   614677375609919 11-02-2018 00:00:00     GENUINE
Time taken: 0.536 seconds, Fetched: 5 row(s)
```

Command to create lookup table to HBASE Database:

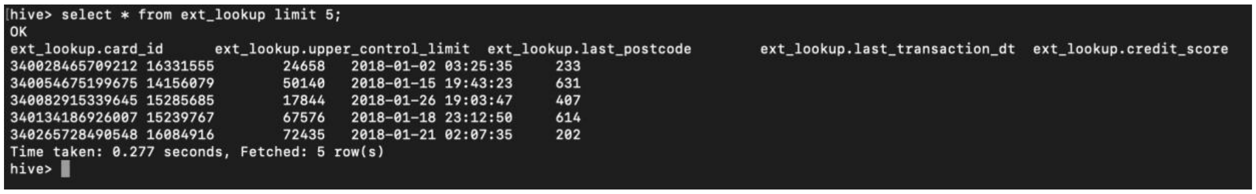
```
import happybase
import uuid

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)
   connection.close()
   print('connection closed')

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

Creating transaction table in HIVE which is linked to the HBase db.

```
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");
```

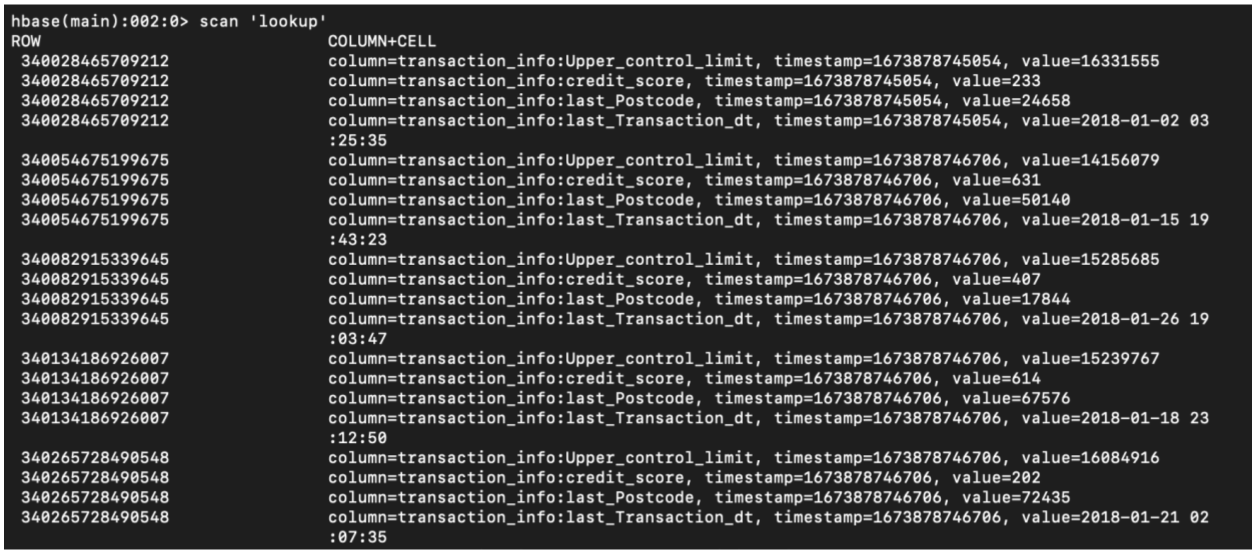
Load the data to the lookup table:

```
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,
STDDEV(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;
```



```
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: 0.277 seconds, Fetched: 5 row(s)
hive>
```

Let us check if the data is loaded to the lookup table in the HBase table:



```
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
```