Scripts Execution

Screenshots of the execution of the scripts written

Joining the dataframe with previous dataframe on card_id:

```
history_df = history_df.select('card_id','UCL')
```

```
look_up_table = look_up_table.join(history_df,on=['card_id'])
```

```
look_up_table.show()
```

```
+----------------+-------------------+-----+--------+--------------------+
|         card_id|   transaction_date|score|postcode|                 UCL|
+----------------+-------------------+-----+--------+--------------------+
| 340379737226464|2018-01-27 00:19:47|  229|   26656|1.4676643749999998E7|
| 345406224887566|2017-12-25 04:03:58|  349|   53034|       1.524603906E7|
| 348962542187595|2018-01-29 17:17:14|  522|   27830|1.5005378620000001E7|
| 377201318164757|2017-11-28 16:32:22|  432|   84302|1.4048015219999999E7|
| 379321864695232|2018-01-03 00:29:37|  297|   98837|       1.432266392E7|
|4389973676463558|2018-01-26 13:47:46|  400|   10985|1.1844220399999999E7|
|4407230633003235|2018-01-27 07:21:08|  567|   50167|1.4173543150000002E7|
|5403923427969691|2018-01-22 23:46:19|  324|   17350|       1.411602776E7|
+----------------+-------------------+-----+--------+--------------------+
```

Remove duplicate on redundant transactions done on card_id, transaction_date and postcode:

```
look_up_table = look_up_table.dropDuplicates((['card_id','transaction_date','postcode']))
```

```
look_up_table.count()
```

```
1000
```

Load the dataframe into the look up table, happybase API shall be used.

```
import happybase

# Establish connection to HBase
connection = happybase.Connection('localhost', port=9090, autoconnect=False)

# Function to open the connection
def open_connection():
    connection.open()

# Function to close the connection
def close_connection():
    connection.close()

# Function to list all tables in HBase
def list_tables():
    print("Fetching all tables")
    open_connection()
    tables = connection.tables()
    close_connection()
    print("All tables fetched")
    return tables
```

```
# Function to create a new HBase table
def create_table(name, cf):
    print("Creating table " + name)
    tables = list_tables()
    if name not in tables:
        open_connection()
        connection.create_table(name, cf)
        close_connection()
        print("Table created")
    else:
        print("Table already present")

# Function to get a table object
def get_table(name):
    open_connection()
    table = connection.table(name)
    close_connection()
    return table
```

If table does not exist, create the table:

```
# Create the lookup table with the specified column family
create_table('look_up_table', {'info': dict(max_versions=5)})
```

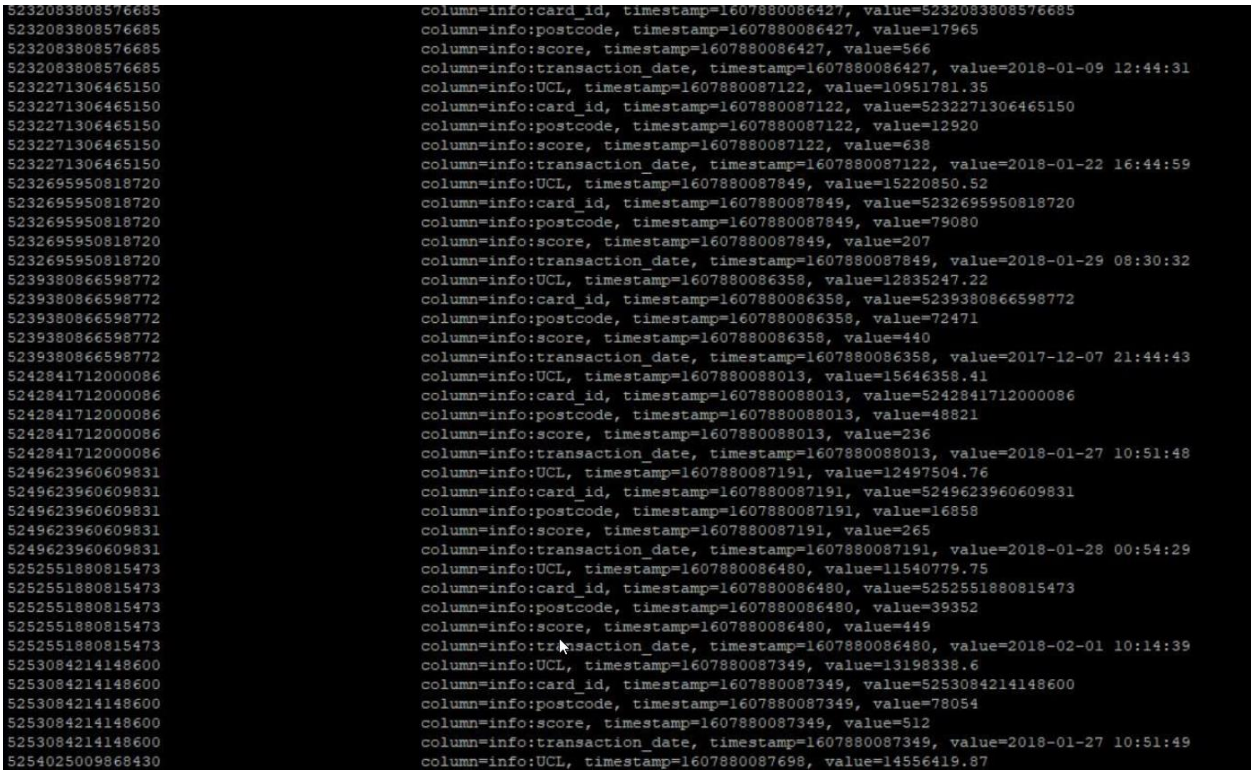
```
Creating table 'look_up_table'...
Fetching all tables...
All tables fetched.
Table 'look_up_table' created successfully.
```

Batch insert data into the table:

```
# Function to batch insert data into the HBase table
def batch_insert_data(df, tableName):
    print("Starting batch insert of events")
    table = get_table(tableName)
    open_connection()
    with table.batch(batch_size=4) as b:
        for row in df.rdd.collect():
            b.put(
                bytes(row.card_id),
                {
                    'info:card_id': bytes(row.card_id),
                    'info:transaction_date': bytes(row.transaction_date),
                    'info:score': bytes(row.score),
                    'info:postcode': bytes(row.postcode),
                    'info:UCL': bytes(row.UCL)
                }
            )
    print("Batch insert done")
    close_connection()

# Insert data into HBase table from the DataFrame
batch_insert_data(look_up_table, 'look_up_table')
```

Once the batch insertion is complete, login to putty as root user and enter Hbase shell



```
5232083808576685                        column=info:card_id, timestamp=1607880086427, value=5232083808576685
5232083808576685                        column=info:postcode, timestamp=1607880086427, value=17965
5232083808576685                        column=info:score, timestamp=1607880086427, value=566
5232083808576685                        column=info:transaction_date, timestamp=1607880086427, value=2018-01-09 12:44:31
5232271306465150                        column=info:UCL, timestamp=1607880087122, value=10951781.35
5232271306465150                        column=info:card_id, timestamp=1607880087122, value=5232271306465150
5232271306465150                        column=info:postcode, timestamp=1607880087122, value=12920
5232271306465150                        column=info:score, timestamp=1607880087122, value=638
5232271306465150                        column=info:transaction_date, timestamp=1607880087122, value=2018-01-22 16:44:59
5232695950818720                        column=info:UCL, timestamp=1607880087849, value=15220850.52
5232695950818720                        column=info:card_id, timestamp=1607880087849, value=5232695950818720
5232695950818720                        column=info:postcode, timestamp=1607880087849, value=79080
5232695950818720                        column=info:score, timestamp=1607880087849, value=207
5232695950818720                        column=info:transaction_date, timestamp=1607880087849, value=2018-01-29 08:30:32
5239380866598772                        column=info:UCL, timestamp=1607880086358, value=12835247.22
5239380866598772                        column=info:card_id, timestamp=1607880086358, value=5239380866598772
5239380866598772                        column=info:postcode, timestamp=1607880086358, value=72471
5239380866598772                        column=info:score, timestamp=1607880086358, value=440
5239380866598772                        column=info:transaction_date, timestamp=1607880086358, value=2017-12-07 21:44:43
5242841712000086                        column=info:UCL, timestamp=1607880088013, value=15646358.41
5242841712000086                        column=info:card_id, timestamp=1607880088013, value=5242841712000086
5242841712000086                        column=info:postcode, timestamp=1607880088013, value=48821
5242841712000086                        column=info:score, timestamp=1607880088013, value=236
5242841712000086                        column=info:transaction_date, timestamp=1607880088013, value=2018-01-27 10:51:48
5249623960609831                        column=info:UCL, timestamp=1607880087191, value=12497504.76
5249623960609831                        column=info:card_id, timestamp=1607880087191, value=5249623960609831
5249623960609831                        column=info:postcode, timestamp=1607880087191, value=16858
5249623960609831                        column=info:score, timestamp=1607880087191, value=265
5249623960609831                        column=info:transaction_date, timestamp=1607880087191, value=2018-01-28 00:54:29
5252551880815473                        column=info:UCL, timestamp=1607880086480, value=11540779.75
5252551880815473                        column=info:card_id, timestamp=1607880086480, value=5252551880815473
5252551880815473                        column=info:postcode, timestamp=1607880086480, value=39352
5252551880815473                        column=info:score, timestamp=1607880086480, value=449
5252551880815473                        column=info:transaction_date, timestamp=1607880086480, value=2018-02-01 10:14:39
5253084214148600                        column=info:UCL, timestamp=1607880087349, value=13198338.6
5253084214148600                        column=info:card_id, timestamp=1607880087349, value=5253084214148600
5253084214148600                        column=info:postcode, timestamp=1607880087349, value=78054
5253084214148600                        column=info:score, timestamp=1607880087349, value=512
5253084214148600                        column=info:transaction_date, timestamp=1607880087349, value=2018-01-27 10:51:49
5254025009868430                        column=info:UCL, timestamp=1607880087698, value=14556419.87
```