Scripts Execution

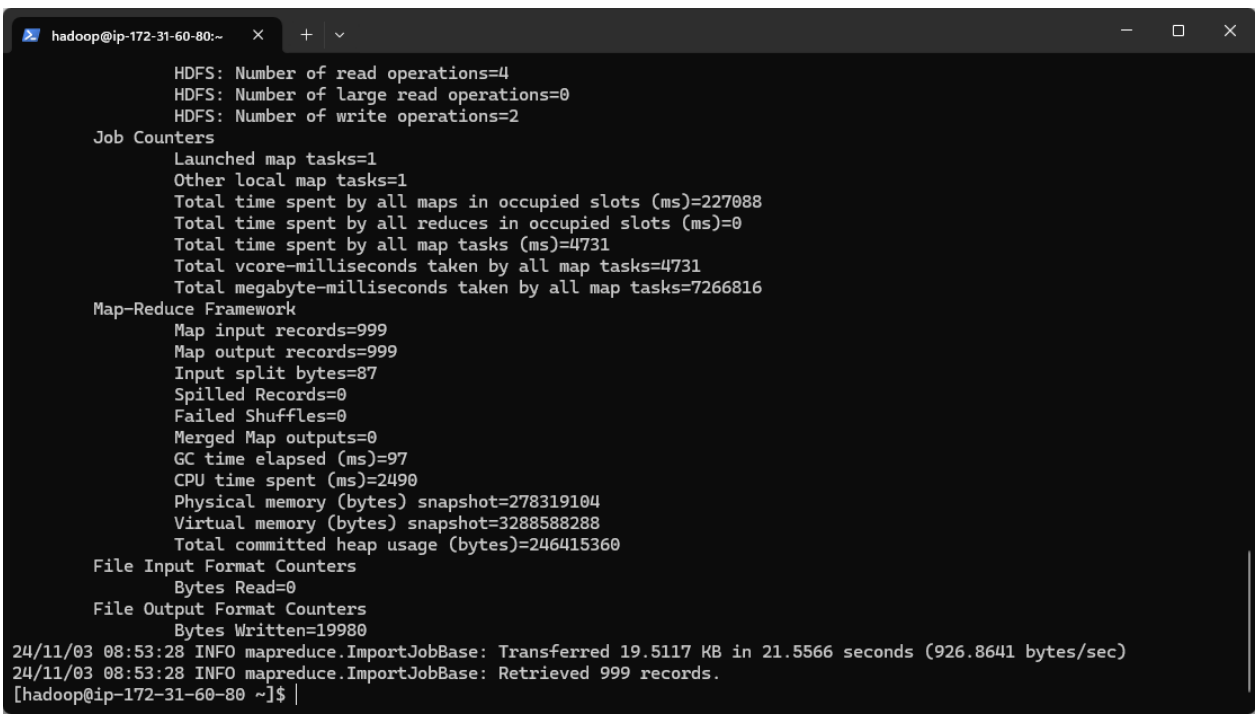
Explanation of the solution to the batch layer problem

<Properly explain the step by step process followed for the completion of tasks till **task 4.** The steps should be properly documented here.>

• Sqoop command to import tables from RDS to HDFS:

Table 1: Command for member_score table:

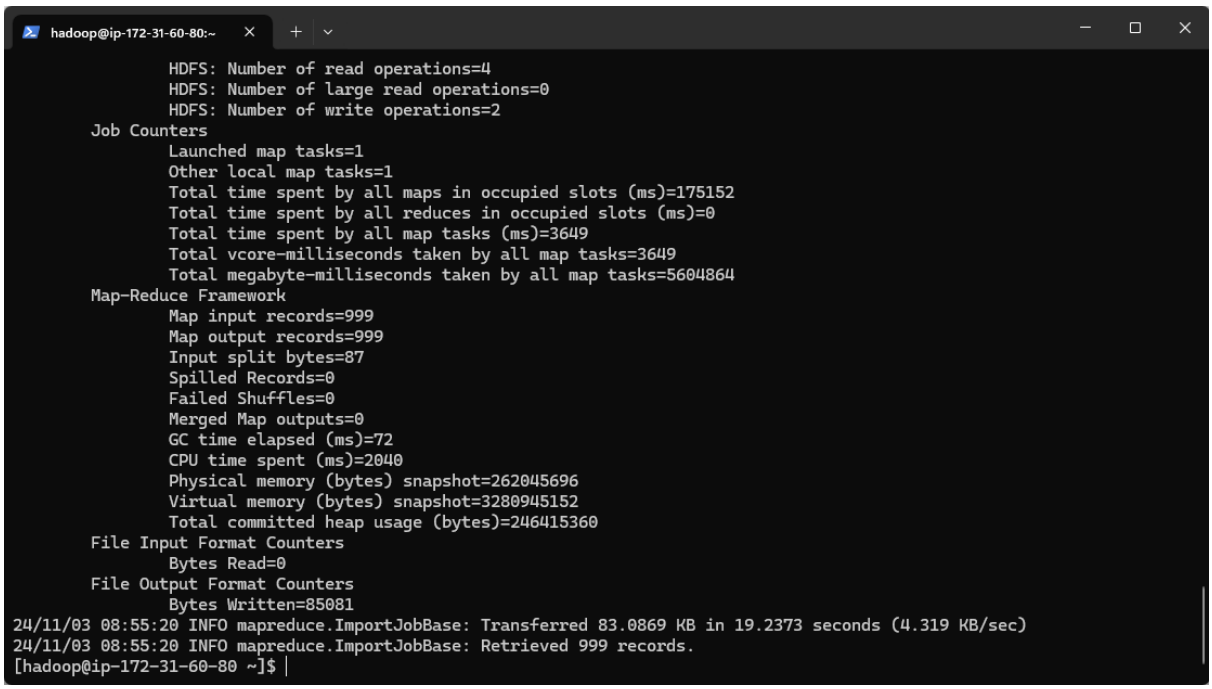
```
sqoop import \
--connect jdbc:mysql://upgradawsrds1.cyaielc9bmnf.us-east-
1.rds.amazonaws.com/cred_financials_data \
--table member_score \
--username upgraduser --password upgraduser \
--target-dir /user/root/cap_project/member_score \
-m 1
```



© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

Table 2: Command for card_member table:

```
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/root/cap_project/card_member \
-m 1
```



```
hadoop@ip-172-31-60-80:~
                HDFS: Number of read operations=4
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Other local map tasks=1
                Total time spent by all maps in occupied slots (ms)=175152
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=3649
                Total vcore-milliseconds taken by all map tasks=3649
                Total megabyte-milliseconds taken by all map tasks=5604864
        Map-Reduce Framework
                Map input records=999
                Map output records=999
                Input split bytes=87
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=72
                CPU time spent (ms)=2040
                Physical memory (bytes) snapshot=262045696
                Virtual memory (bytes) snapshot=3280945152
                Total committed heap usage (bytes)=246415360
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=85081
24/11/03 08:55:20 INFO mapreduce.ImportJobBase: Transferred 83.0869 KB in 19.2373 seconds (4.319 KB/sec)
24/11/03 08:55:20 INFO mapreduce.ImportJobBase: Retrieved 999 records.
[hadoop@ip-172-31-60-80 ~]$
```

- Verify the data inside the imported directories:

```
•   hadoop fs -cat /user/root/cap_project/member_score/part-m-00000
•   hadoop fs -cat /user/root/cap_project/card_member/part-m-00000
```

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

```
973974062665904,565
974087224071871,682
975111321017949,306
976362362966586,210
976740397894598,303
978465390240911,270
978786807247400,652
979218131207765,552
979274357975688,353
981612675682175,265
982171321477033,683
983248478716258,473
983390343419665,183
985095728641698,325
985619195348046,310
987837606639725,623
988661034778977,496
988666952112281,531
989174793157941,533
989548647998824,506
990571384923260,469
990740662283435,202
991784667284668,621
991958482478439,215
992104321998571,658
992552823055811,694
994983851226493,687
996411635289270,335
997128952368160,683
[hadoop@ip-172-31-60-80 ~]$
```

```
6562510549485881,659982919406634,2015-07-04 06:39:15.0,11/16,United States,Mount Vernon
6566718331696237,335526117403543,2013-09-19 10:34:17.0,08/14,United States,Sand Springs
6568438963325843,183038102441540,2017-07-06 05:25:55.0,07/17,United States,North Las Vegas
6570061543816002,722794052951029,2017-01-10 03:01:36.0,02/17,United States,Sedalia
6573615549671948,985619195348046,2015-09-29 08:21:37.0,12/16,United States,Belleville
6574255180086418,891702243060747,2012-07-19 07:47:35.0,07/15,United States,Selma
6574656060675651,555127039477442,2010-06-02 00:11:01.0,11/10,United States,Coral Terrace
6577397607792660,479331531880555,2013-06-13 00:02:59.0,02/16,United States,Norco
6578328793401039,149907932138330,2016-03-21 02:51:11.0,11/16,United States,Wasco
6580194731591489,218337397923739,2018-01-09 00:32:14.0,01/18,United States,Oxford
6582135370631324,389050970153068,2016-07-17 10:26:54.0,08/16,United States,Roy
6582950745622331,185289866247329,2012-04-17 03:37:04.0,03/13,United States,Hutchinson
6583283909793527,398607904778893,2015-08-27 03:19:32.0,01/17,United States,Galesburg
6585229076432173,887283455220105,2015-03-06 00:53:06.0,05/16,United States,Security-Widefield
6587114664896609,623414156260264,2015-10-05 06:45:35.0,03/17,United States,Hobart
6588637504808417,678537053778506,2012-05-30 04:37:45.0,04/14,United States,Concord
6589539191695876,832393339334706,2014-01-26 11:59:35.0,09/15,United States,Roseville
6589855658684769,193386908982058,2010-02-07 09:52:17.0,11/15,United States,Westfield
6589894320960430,276119478113309,2011-08-08 02:25:29.0,11/15,United States,Blaine
6590907016354002,849289511810887,2016-10-28 07:32:23.0,07/17,United States,Reading
6591175617713393,135615957283320,2015-05-16 03:21:15.0,08/16,United States,Arvin
6592184145413632,714853788321318,2010-03-12 08:11:17.0,08/16,United States,Mehlville
6594248319343442,846162645575666,2016-02-15 02:32:58.0,12/16,United States,Lyndhurst
6595638658736751,377918213614639,2015-08-13 10:58:24.0,09/17,United States,Mahwah
6595814135833988,236864426408837,2011-02-18 06:45:46.0,06/14,United States,Johnston
6595928469079750,422173403853722,2012-02-10 02:19:20.0,02/14,United States,Raleigh
6597703848279563,657991752750272,2017-02-16 07:45:34.0,03/17,United States,Pinewood
6598830758632447,660350842993890,2013-07-26 02:22:17.0,12/16,United States,Tampa
6599900931314251,928036864799687,2017-09-09 03:31:54.0,10/17,United States,Loves Park
[hadoop@ip-172-31-60-80 ~]$
```

- Command to load card_transactions.csv to HDFS after moving to EC2-USER:

```
hadoop fs -copyFromLocal /home/hadoop/card_transactions.csv
/user/root/cap_project/card_transactions.csv
```

- Connect to instance over putty and load the jupyter notebook from root user:

```
jupyter notebook --port 7861 --allow-root
```

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

• Loading member score data stored in central AWS RDS:

```
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, IntegerType
```

```
memberschema = StructType([
          StructField('member_id', StringType(), False),
          StructField('score', IntegerType(), False),
         ])

memf = spark.read.csv("hdfs:/user/root/cap_project/member_score", header=False,
schema=memberschema)
memf.show()
```

```
memf.show()
```

```
+---------------+-----+
|      member_id|score|
+---------------+-----+
|000037495066290|  339|
|000117826301530|  289|
|001147922084344|  393|
|001314074991813|  225|
|001739553947511|  642|
|003761426295463|  413|
|004494068832701|  217|
|006836124210484|  504|
|006991872634058|  697|
|007955566230397|  372|
|008732267588672|  213|
```

• Loading the card holder's data stored in central AWS RDS:

```
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType

# Initialize SparkSession
spark = SparkSession.builder \
    .appName("Card Holder Data Load") \
    .getOrCreate()

# Define the schema for card holder's data
cardschema = StructType([
    StructField('card_id', StringType(), False),
    StructField('member_id', StringType(), False),
    StructField('member_joining_dt', StringType(), False),
    StructField('card_purchase_dt', StringType(), False),
    StructField('country', StringType(), False),
    StructField('city', StringType(), False),
])

# Read the data from HDFS
cardf = spark.read.csv("hdfs:/user/root/cap_project/card_member", header=False, schema=cardschema)

# Display the data
cardf.show()
```

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

```
cardschema = StructType([StructField('card_id', StringType(),False),
                         StructField('member_id', StringType(),False),
                         StructField('member_joining_dt', StringType(),False),
                         StructField('card_purchase_dt', StringType(),False),
                         StructField('country', StringType(),False),
                         StructField('city', StringType(),False),
                         ])
```

```
cardf = spark.read.csv("hdfs:/user/root/cap_project/card_member", header = False, schema = cardschema)
```

```
cardf.show()
```

```
+---------------+---------------+--------------------+----------------+-------------+-----------------+
|        card_id|      member_id|   member_joining_dt|card_purchase_dt|      country|             city|
+---------------+---------------+--------------------+----------------+-------------+-----------------+
|340028465709212|009250698176266|2012-02-08 06:04:...|           05/13|United States|        Barberton|
|340054675199675|835873341185231|2017-03-10 09:24:...|           03/17|United States|       Fort Dodge|
|340082915339645|512969555857346|2014-02-15 06:30:...|           07/14|United States|           Graham|
|340134186926007|887711945571282|2012-02-05 01:21:...|           02/13|United States|        Dix Hills|
|340265728490548|680324265406190|2014-03-29 07:49:...|           11/14|United States| Rancho Cucamonga|
|340268219434811|929799084911715|2012-07-08 02:46:...|           08/12|United States|    San Francisco|
|340379737226464|089615510858348|2010-03-10 00:06:...|           09/10|United States|          Clinton|
|340383645652108|181180599313885|2012-02-24 05:32:...|           10/16|United States|    West New York|
|340803866934451|417664728506297|2015-05-21 04:30:...|           08/17|United States|        Beaverton|
```

• Loading card_transaction data from csv:

```
transasction = StructType([StructField('card_id', StringType(),False),
                           StructField('member_id', StringType(),False),
                           StructField('amount', IntegerType(),False),
                           StructField('postcode', StringType(),False),
                           StructField('pos_id', StringType(),False),
                           StructField('transaction_dt', StringType(),False),
                           StructField('status', StringType(),False),
                           ])
```

```
tranf = spark.read.csv("hdfs:/user/root/cap_project/card_transactions.csv", header = True, schema = transasction)
```

```
tranf= tranf.filter(tranf.status!='FRAUD')
```

```
tranf.show()
```

```
+---------------+---------------+-------+--------+---------------+-------------------+-------+
|        card_id|      member_id| amount|postcode|         pos_id|     transaction_dt| status|
+---------------+---------------+-------+--------+---------------+-------------------+-------+
|348702330256514|000037495066290|9084849|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290| 330148|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290| 136052|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290|4310362|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290|9097094|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290|2291118|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290|4900011|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290| 633447|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290|6259303|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290| 369067|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290|1193207|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
```

• Joining the member_score and card_member on member_id to extract credit score of each member and selecting the required fields:

```
# Join member_score and card_member DataFrames on 'member_id'
score = memf.join(cardf, memf.member_id == cardf.member_id, how='left')

# Print the schema to verify the join
score.printSchema()

# Select the relevant columns: 'member_id', 'score', and 'card_id'
score = score.select('member_id', 'score', 'card_id')

# Show a preview of the score DataFrame
score.show()
```

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

```
score = memf.join(cardf, memf.mem_id == cardf.member_id,how='LEFT')
```

```
score.printSchema()
```

```
root
 |-- mem_id: string (nullable = true)
 |-- score: integer (nullable = true)
 |-- card_id: string (nullable = true)
 |-- member_id: string (nullable = true)
 |-- member_joining_dt: string (nullable = true)
 |-- card_purchase_dt: string (nullable = true)
 |-- country: string (nullable = true)
 |-- city: string (nullable = true)
```

```
score = score.select('mem_id', 'score', 'card_id')
```

```
score.show()
```

```
+---------------+-----+----------------+
|         mem_id|score|         card_id|
+---------------+-----+----------------+
|000037495066290|  339| 348702330256514|
|000117826301530|  289|5189563368503974|
|001147922084344|  393|5407073344486464|
|001314074991813|  225| 378303738095292|
|001739553947511|  642| 348413196172048|
|003761426295463|  413| 348536585266345|
|004494068832701|  217|5515987071565183|
```

- Joining both the history transactions CSV and score DF:

```
hist = tranf.join(score, tranf.member_id == score.mem_id,how='outer')
```

```
hist.printSchema()
```

```
root
 |-- card_id: string (nullable = true)
 |-- member_id: string (nullable = true)
 |-- amount: integer (nullable = true)
 |-- postcode: string (nullable = true)
 |-- pos_id: string (nullable = true)
 |-- transaction_dt: string (nullable = true)
 |-- status: string (nullable = true)
 |-- mem_id: string (nullable = true)
 |-- score: integer (nullable = true)
 |-- cardid: string (nullable = true)
```

```
hist = hist.select('card_id', 'amount', 'postcode', 'pos_id','transaction_dt','status','score')
```

```
hist.show()
```

```
+---------------+-------+--------+---------------+-------------------+-------+-----+
|        card_id| amount|postcode|         pos_id|     transaction_dt| status|score|
+---------------+-------+--------+---------------+-------------------+-------+-----+
|340379737226464|6126197|   46933|167473544283898|01-05-2016 08:10:50|GENUINE|  229|
|340379737226464|7949232|   61840|664980919335952|01-10-2016 10:38:52|GENUINE|  229|
|340379737226464| 943839|   91743|633038040069180|02-08-2016 00:31:25|GENUINE|  229|
|340379737226464|3764114|   91743|633038040069180|02-08-2016 21:35:27|GENUINE|  229|
|340379737226464|6221251|   98384|064948657945290|02-10-2016 14:44:14|GENUINE|  229|
|340379737226464|2868312|   26032|856772774421259|02-12-2016 21:55:43|GENUINE|  229|
|340379737226464|4418586|   20129|390339673634463|02-12-2017 17:05:51|GENUINE|  229|
```

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

- Compute the max transaction date:

```
look_up_table = history.groupBy('card_id').agg(f.max("transaction_date").alias('transaction_date'))
```

```
look_up_table.show()
```

```
+----------------+-------------------+
|         card_id|   transaction_date|
+----------------+-------------------+
| 340379737226464|2018-01-27 00:19:47|
| 377201318164757|2017-11-28 16:32:22|
| 348962542187595|2018-01-29 17:17:14|
|4389973676463558|2018-01-26 13:47:46|
|5403923427969691|2018-01-22 23:46:19|
| 345406224887566|2017-12-25 04:03:58|
|6562510549485881|2018-01-17 08:35:27|
|5508842242491554|2018-01-31 14:55:58|
|4407230633003235|2018-01-27 07:21:08|
| 379321864695232|2018-01-03 00:29:37|
```

- Inner join on look up table dataset on card_id:

```
look_up_table = look_up_table.join(score, look_up_table.card_id == score.cardid,how='INNER')
```

```
look_up_table.show()
```

```
+----------------+-------------------+---------------+-----+----------------+
|         card_id|   transaction_date|         mem_id|score|          cardid|
+----------------+-------------------+---------------+-----+----------------+
| 340379737226464|2018-01-27 00:19:47|089615510858348|  229| 340379737226464|
| 345406224887566|2017-12-25 04:03:58|296206661780881|  349| 345406224887566|
| 348962542187595|2018-01-29 17:17:14|366246487993992|  522| 348962542187595|
| 377201318164757|2017-11-28 16:32:22|924475891017022|  432| 377201318164757|
```

- Calculate the Upper Control Limit (UCL), UCL = Moving Average + 3 * (Standard Deviation), We shall first calculate the moving average of card amount for last 10 transactions.
- Create a window over existing dataframe and aggregate the same card_id, the dataframe is grouped by card_id and then order by transaction_date.

```
window = Window.partitionBy(history['card_id']).orderBy(history['transaction_date'].desc())

history_df = history.select('*', f.rank().over(window).alias('rank')).filter(f.col('rank') <= 10)
```

```
history_df.show()
```

```
+---------------+-------+--------+---------------+-------+-----+-------------------+----+
|        card_id| amount|postcode|         pos_id| status|score|   transaction_date|rank|
+---------------+-------+--------+---------------+-------+-----+-------------------+----+
|340379737226464|1784098|   26656|000383013889790|GENUINE|  229|2018-01-27 00:19:47|   1|
|340379737226464|3759577|   61334|016312401940277|GENUINE|  229|2018-01-18 14:26:09|   2|
|340379737226464|4080612|   51338|562082278231631|GENUINE|  229|2018-01-14 20:54:02|   3|
|340379737226464|4242710|   96105|285501971776349|GENUINE|  229|2018-01-11 19:09:55|   4|
|340379737226464|9061517|   40932|232455833079472|GENUINE|  229|2018-01-10 20:20:33|   5|
|340379737226464| 102248|   40932|232455833079472|GENUINE|  229|2018-01-10 15:04:33|   6|
|340379737226464|7445128|   50455|915439934619047|GENUINE|  229|2018-01-07 23:52:27|   7|
|340379737226464|5706163|   50455|915439934619047|GENUINE|  229|2018-01-07 22:07:07|   8|
|340379737226464|8090127|   18626|359283931604637|GENUINE|  229|2017-12-29 13:24:07|   9|
|340379737226464|9282351|   41859|808326141065551|GENUINE|  229|2017-12-28 19:50:46|  10|
```

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

To import all SQL functions to pyspark, we need to import the necessary functions (import pyspark.sql.functions as f)

```
history_df = history_df.groupBy("card_id").agg(f.round(f.avg('amount'),2).alias('moving_avg'), \
                                                f.round(f.stddev('amount'),2).alias('Std_Dev'))
```

```
+----------------+----------+----------+
|         card_id|moving_avg|   Std_Dev|
+----------------+----------+----------+
| 340379737226464| 5355453.1|3107063.55|
| 345406224887566| 5488456.5|3252527.52|
| 348962542187595| 5735629.0|3089916.54|
| 377201318164757| 5742377.7|2768545.84|
| 379321864695232| 4713319.1|3203114.94|
|4389973676463558| 4923904.7| 2306771.9|
|4407230633003235| 4348891.3|3274883.95|
```

- Calculate UCL from the computed standard deviation and moving average:

```
history_df = history_df.withColumn('UCL',history_df.moving_avg+3*(history_df.Std_Dev))
```

```
+----------------+----------+----------+--------------------+
|         card_id|moving_avg|   Std_Dev|                 UCL|
+----------------+----------+----------+--------------------+
| 340379737226464| 5355453.1|3107063.55|1.4676643749999998E7|
| 345406224887566| 5488456.5|3252527.52|       1.524603906E7|
| 348962542187595| 5735629.0|3089916.54|1.5005378620000001E7|
| 377201318164757| 5742377.7|2768545.84|1.4048015219999999E7|
| 379321864695232| 4713319.1|3203114.94|       1.432266392E7|
|4389973676463558| 4923904.7| 2306771.9|1.1844220399999999E7|
|4407230633003235| 4348891.3|3274883.95|1.4173543150000002E7|
```

- 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
```

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

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

1st step is to connect with hbase:

```
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)})
```

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

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

Batch insert data into the table:

```python
# 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
```

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved