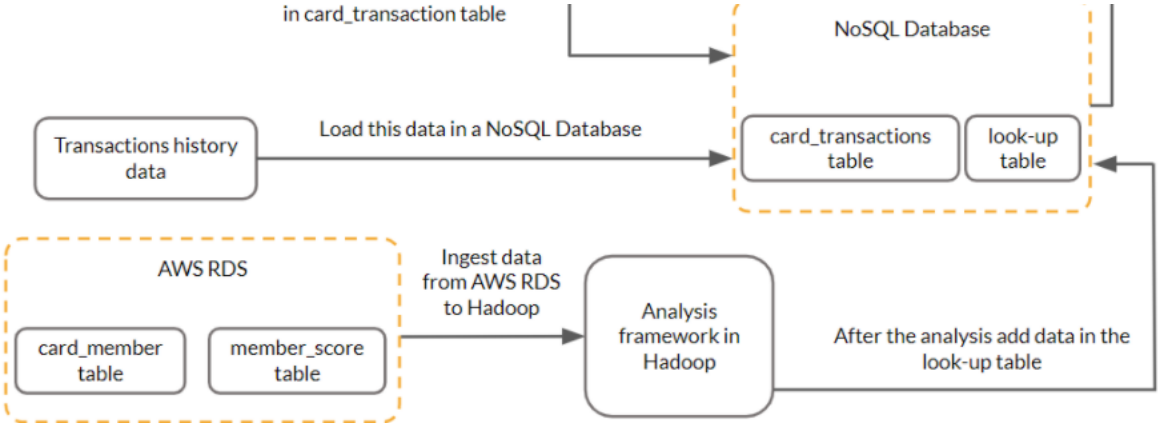
Scripts Execution

Screenshots of the execution of the scripts written

We have **following tasks** to be performed as per **batch layer problem** (till task4)

1. Task1: Load the transactions history data (card_transactions.csv) in a NoSQL database
2. Task2: Ingest the relevant data from AWS RDS to Hadoop
3. Task3: Create a lookup-table with columns specified
4. Task4: Load the data in lookup table



1. **Load the transactions history data (card_transactions.csv)** : Load Transactions history data to HDFS by first copying to local and then to destination

hadoop fs -put card_transactions.csv /user/root/capstone_project

2. Now, connect to putty instance.
3. load jupyter notebook from root user, by using command
   jupyter notebook --port 7861 --allow-root
4. Open a new notebook and load a spark context.
5. Start reading card_transactions data as well as the data after ingesting from AWS RDS

2. Ingest data from AWS RDS to hadoop using Sqoop

Table 1:member_score

sqoop import \

--connect jdbc:mysql://upgradawsrds1.cyaielc9bmnf.us-east1.rds.amazonaws.com/cred_financials_data \
--table member_score \
--username upgraduser --password upgraduser \
--target-dir /user/root/capstone_project/member_score \
-m 1

Table 2: card_member

sqoop import \
--connect jdbc:mysql://upgradawsrds1.cyaielc9bmnf.us-east1.rds.amazonaws.com/cred_financials_data \
--table card_member \
--username upgraduser --password upgraduser \
--target-dir /user/root/capstone_project/card_member \
-m 1

```
[root@ip-10-0-0-149 ~]# hadoop fs -ls /user/root/capstone_project
Found 2 items
drwxr-xr-x   - root supergroup          0 2021-10-16 07:29 /user/root/capstone_project/card_member
drwxr-xr-x   - root supergroup          0 2021-10-16 07:23 /user/root/capstone_project/member_score
```

Card member data

First load data from RDS to hdfs and then create data frame

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

```
+---------------+---------------+-------------------+----------------+-------------+-----------------+
|        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|
|340889618969736|459292914761635|2013-04-23 08:40:...|           11/15|United States|  West Palm Beach|
|340924125838453|188119365574843|2011-04-12 04:28:...|           12/13|United States|      Scottsbluff|
|341005627432127|872138964937565|2013-09-08 03:16:...|           02/17|United States|          Chillum|
|341029651579925|974087224071871|2011-01-14 00:20:...|           08/12|United States|   Valley Station|
|341311317050937|561687420200207|2014-03-18 06:23:...|           02/15|United States|        Vincennes|
|341344252914274|695906467918552|2012-03-02 03:21:...|           03/13|United States|        Columbine|
|341363858179050|009190444424572|2012-02-19 05:16:...|           04/14|United States|      Cheektowaga|
|341519629171378|533670008048847|2013-05-13 07:59:...|           01/15|United States|       Centennial|
|341641153427489|230523184584316|2013-03-25 08:51:...|           11/15|United States|       Colchester|
|341719092861087|304847505155781|2015-12-06 08:06:...|           11/17|United States|     Vernon Hills|
|341722035429601|979218131207765|2015-12-22 10:46:...|           01/17|United States|Elk Grove Village|
+---------------+---------------+-------------------+----------------+-------------+-----------------+
only showing top 20 rows
```

Member score data

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

```
#read the data
mem_df = spark.read.csv("hdfs:/user/root/capstone_project/member_score", header = False, schema = memberschema)
```

```
mem_df.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|
|008765307152821|  399|
|009136568025042|  308|
|009190444424572|  559|
|009250698176266|  233|
|009873334520465|  298|
|011716573646690|  249|
|011877954983420|  497|
|012390918683920|  407|
|012731668664932|  612|
+---------------+-----+
only showing top 20 rows
```

Past Transactions data

23]:
```
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),
                        ])
```

Reading Past Transactions data (source as csv)

25]:
```
trans_df = spark.read.csv("hdfs:/user/root/capstone_project/card_transactions.csv", header = True, schema = transasction)
```

```
+---------------+---------------+-------+--------+---------------+-------------------+-------+
|        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|
|348702330256514|000037495066290|9335696|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290|2241736|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290| 457701|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290|7176668|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290|5585098|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290|7918756|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290|1611089|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290| 217221|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
|348702330256514|000037495066290|2617991|   33946|614677375609919|11-02-2018 00:00:00|GENUINE|
+---------------+---------------+-------+--------+---------------+-------------------+-------+
only showing top 20 rows
```

3. Creating lookup table:

Join the card_member and member_score tables to extract credit score of each member

```
score = mem_df.join(card_df, mem_df.mem_id == card_df.member_id,how='LEFT')
```

```
score.count()
```

```
999
```

```
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)
```

Extract required columns from the joined dataframe

```
score.show()
```

```
+---------------+-----+----------------+
|         mem_id|score|          cardid|
+---------------+-----+----------------+
|000037495066290|  339| 348702330256514|
|000117826301530|  289|5189563368503974|
|001147922084344|  393|5407073344486464|
|001314074991813|  225| 378303738095292|
|001739553947511|  642| 348413196172048|
|003761426295463|  413| 348536585266345|
|004494068832701|  217|5515987071565183|
|006836124210484|  504|5400251558458125|
|006991872634058|  697|4573337022888445|
|007955566230397|  372|4708912758619517|
|008732267588672|  213|5342400571435088|
|008765307152821|  399|4237648081700588|
|009136568025042|  308| 371814781663843|
|009190444424572|  559| 341363858179050|
|009250698176266|  233| 340028465709212|
|009873334520465|  298|5495445301620991|
|011716573646690|  249|4795844193055110|
|011877954983420|  497|5164771396791995|
|012390918683920|  407|5423921058459194|
|012731668664932|  612|5379610024035907|
+---------------+-----+----------------+
only showing top 20 rows
```

Join both Transaction history and score Dataframe which is a merged and extracted data frame from both RDS tables.

```
hist = trans_df.join(score, trans_df.member_id == score.mem_id,how='outer') #outer join on member_id between score ar
```

```
hist.count() #53210
```

```
53210
```

```
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|
|340379737226464|7439113|   91763|315067016872305|03-04-2017 11:43:59|GENUINE|  229|
|340379737226464|8217180|   16063|208378790148728|03-05-2017 16:47:43|GENUINE|  229|
|340379737226464|8505852|   64070|695556848392133|03-06-2017 03:07:27|GENUINE|  229|
|340379737226464|8535431|   29817|683602833507395|04-08-2016 20:59:31|GENUINE|  229|
|340379737226464|6317993|   28425|258522244165233|05-05-2017 00:23:45|GENUINE|  229|
|340379737226464|3256860|   16845|933410474855991|05-10-2017 15:09:09|GENUINE|  229|
|340379737226464|1423779|   97640|789378980336517|06-02-2017 02:10:00|GENUINE|  229|
|340379737226464|3783517|   70552|963177679534627|06-12-2016 03:10:30|GENUINE|  229|
|340379737226464|3300714|   75750|072728631441941|07-01-2017 05:52:58|GENUINE|  229|
|340379737226464|5706163|   50455|915439934619047|07-01-2018 22:07:07|GENUINE|  229|
|340379737226464|7445128|   50455|915439934619047|07-01-2018 23:52:27|GENUINE|  229|
|340379737226464| 140120|   18915|691571327905821|07-02-2017 20:18:04|GENUINE|  229|
|340379737226464|7720484|   48423|548702836055067|07-03-2016 14:59:35|GENUINE|  229|
+---------------+-------+--------+---------------+-------------------+-------+-----+
only showing top 20 rows
```

To calculate the latest transaction date of that card:

- group the merged dataset on card_id
- aggreagte to max of transaction date.
- Alias the aggregated date as transaction_date

```
+---------------+-------------------+
|        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|
| 340028465709212|2018-01-02 03:25:35|
| 349143706735646|2018-01-29 22:33:14|
|4126356979547079|2018-01-24 16:09:03|
|5543219113990484|2018-01-13 18:34:00|
|5464688416792307|2018-01-26 19:03:47|
|6011273561157733|2018-02-01 01:27:58|
|4484950467600170|2018-01-10 08:03:13|
|4818950814628962|2018-01-31 00:53:15|
|5573293264792992|2018-01-31 14:55:57|
|6011985140563103|2018-01-30 02:03:54|
+---------------+-------------------+
only showing top 20 rows
```

Join previous last step data frame (score) with look_up_table dataset created above. This step frames all required cols for look_up_table except the UCL.

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

```
lookup_table.count()  #check the count (999)
```

```
999
```

```
lookup_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|
| 379321864695232|2018-01-03 00:29:37|082567374418739|  297| 379321864695232|
|4389973676463558|2018-01-26 13:47:46|295554828848966|  400|4389973676463558|
|4407230633003235|2018-01-27 07:21:08|761335698364860|  567|4407230633003235|
|5403923427969691|2018-01-22 23:46:19|922077754605834|  324|5403923427969691|
|5508842242491554|2018-01-31 14:55:58|634200295989311|  585|5508842242491554|
|6562510549485881|2018-01-17 08:35:27|659982919406634|  518|6562510549485881|
| 340028465709212|2018-01-02 03:25:35|009250698176266|  233| 340028465709212|
| 349143706735646|2018-01-29 22:33:14|343824445342591|  298| 349143706735646|
|4126356979547079|2018-01-24 16:09:03|015582765997171|  345|4126356979547079|
|4484950467600170|2018-01-10 08:03:13|570539968421790|  462|4484950467600170|
|4818950814628962|2018-01-31 00:53:15|819006616594636|  660|4818950814628962|
|5464688416792307|2018-01-26 19:03:47|434792568351651|  469|5464688416792307|
|5543219113990484|2018-01-13 18:34:00|501241235491851|  494|5543219113990484|
|5573293264792992|2018-01-31 14:55:57|350307876868039|  284|5573293264792992|
|6011273561157733|2018-02-01 01:27:58|314862932674883|  411|6011273561157733|
|6011985140563103|2018-01-30 02:03:54|393165367933607|  350|6011985140563103|
+----------------+-------------------+---------------+-----+----------------+
only showing top 20 rows
```

Calculating UCL:

- Calculate the moving average and standard deviation of the last 10 transactions for each card_id for the data present in Hadoop and NoSQL database
- With the fresh dataframe, use member ID once again as common key and join with card_transaction.csv to load postcode, pos_id, status, amount & transaction date fields from history transactions
- open a window frame where we group input dataframe rows on card_id and order by transaction date to get all transactions on card in chronological order

```
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|
|345406224887566|1135534|   53034|146838238062262|GENUINE|  349|2017-12-25 04:03:58|   1|
|345406224887566|5190295|   88036|821406924682103|GENUINE|  349|2017-12-20 04:41:07|   2|
|345406224887566|5970187|   28334|024341862357645|GENUINE|  349|2017-11-30 05:24:25|   3|
|345406224887566|3854486|   48880|172521878612232|GENUINE|  349|2017-09-21 00:01:58|   4|
|345406224887566|1242240|   14510|536497882467098|GENUINE|  349|2017-06-11 16:31:45|   5|
|345406224887566|9222549|   68358|875905403447795|GENUINE|  349|2017-06-10 21:13:03|   6|
|345406224887566|8726784|   64487|617331009748827|GENUINE|  349|2017-03-16 03:04:40|   7|
|345406224887566|2415599|   99137|751829480922658|GENUINE|  349|2017-03-08 12:29:44|   8|
|345406224887566|9671941|   65614|607206139883123|GENUINE|  349|2017-01-21 08:42:47|   9|
|345406224887566|7454950|   18249|368724323320131|GENUINE|  349|2016-12-30 04:46:01|  10|
+---------------+-------+--------+---------------+-------+-----+-------------------+----+
only showing top 20 rows
```

Import sql function and then calculate Stddev on amount field
UCL i.e. moving average + 3 * (standard deviation)

```
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'))
history_df.show()
```

```
+---------------+----------+----------+
|        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|
|5403923427969691| 5375495.6|2913510.72|
|5508842242491554| 4570725.9|3229905.04|
|6562510549485881| 5551056.9|2501552.48|
| 340028465709212| 6863758.9|3326644.65|
| 349143706735646| 5453372.9|3424332.26|
|4126356979547079| 4286400.2|2909676.26|
|4484950467600170| 4550480.5|3171538.48|
|4818950814628962| 2210428.9| 958307.87|
|5464688416792307| 4985938.2|2379084.95|
|5543219113990484| 4033586.9|2969107.42|
|5573293264792992| 3929994.0|2589503.93|
|6011273561157733| 4634624.8|2801886.17|
|6011985140563103| 5302878.9| 3088988.7|
```

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

```
+----------------+----------+----------+--------------------+
|         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|1.18442203999999999E7|
|4407230633003235| 4348891.3|3274883.95|1.4173543150000002E7|
|5403923427969691| 5375495.6|2913510.72|       1.411602776E7|
|5508842242491554| 4570725.9|3229905.04|1.4260441020000001E7|
|6562510549485881| 5551056.9|2501552.48|       1.305571434E7|
| 340028465709212| 6863758.9|3326644.65|       1.684369285E7|
| 349143706735646| 5453372.9|3424332.26|       1.572636968E7|
|4126356979547079| 4286400.2|2909676.26|       1.301542898E7|
|4484950467600170| 4550480.5|3171538.48|       1.406509594E7|
|4818950814628962| 2210428.9| 958307.87|          5085352.51|
|5464688416792307| 4985938.2|2379084.95|       1.212319305E7|
|5543219113990484| 4033586.9|2969107.42|       1.294090916E7|
|5573293264792992| 3929994.0|2589503.93|1.1698505790000001E7|
|6011273561157733| 4634624.8|2801886.17|1.30402833099999999E7|
|6011985140563103| 5302878.9| 3088988.7|1.4569845000000002E7|
```

Join latest dataframe with previous to get all the required data. Final lookup table looks as below:

```
lookup_table.show()  #Final data set look as below
```

```
+----------------+-------------------+-------+--------+---------------+-------+-----+--------------------+
|         card_id|   transaction_date| amount|postcode|         pos_id| status|score|                 UCL|
+----------------+-------------------+-------+--------+---------------+-------+-----+--------------------+
| 340379737226464|2018-01-27 00:19:47|1784098|   26656|000383013889790|GENUINE|  229|1.467664374999999E7|
| 345406224887566|2017-12-25 04:03:58|1135534|   53034|146838238062262|GENUINE|  349|       1.524603906E7|
| 348962542187595|2018-01-29 17:17:14|7408949|   27830|453850044027107|GENUINE|  522|1.5005378620000001E7|
| 377201318164757|2017-11-28 16:32:22|4799826|   84302|287431794718846|GENUINE|  432|1.4048015219999999E7|
| 379321864695232|2018-01-03 00:29:37|5702120|   98837|638380208258390|GENUINE|  297|       1.432266392E7|
|4389973676463558|2018-01-26 13:47:46|7196505|   10985|588476547410852|GENUINE|  400|1.18442203999999999E7|
|4407230633003235|2018-01-27 07:21:08|  38579|   50167|697070998627535|GENUINE|  567|1.4173543150000002E7|
|5403923427969691|2018-01-22 23:46:19|1576154|   17350|734614251977032|GENUINE|  324|       1.411602776E7|
|5508842242491554|2018-01-31 14:55:58|2710473|   12986|990193545769550|GENUINE|  585|1.4260441020000001E7|
|6562510549485881|2018-01-17 08:35:27|5939348|   35440|901627725704672|GENUINE|  518|       1.305571434E7|
| 340028465709212|2018-01-02 03:25:35|8696557|   24658|246987608008994|GENUINE|  233|       1.684369285E7|
| 349143706735646|2018-01-29 22:33:14|9246599|   99101|743905143665678|GENUINE|  298|       1.572636968E7|
|4126356979547079|2018-01-24 16:09:03|1770784|   14475|698032801419746|GENUINE|  345|       1.301542898E7|
|4484950467600170|2018-01-10 08:03:13|2284955|   13324|653851258729390|GENUINE|  462|       1.406509594E7|
|4818950814628962|2018-01-31 00:53:15|2316346|   88081|127695801600255|GENUINE|  660|          5085352.51|
|5464688416792307|2018-01-26 19:03:47|4067979|   71670|111365575664933|GENUINE|  469|       1.212319305E7|
|5543219113990484|2018-01-13 18:34:00| 549641|   62273|039213658608911|GENUINE|  494|       1.294090916E7|
|5573293264792992|2018-01-31 14:55:57|4827477|   27012|805073498705051|GENUINE|  284|1.16985057900000001E7|
|6011273561157733|2018-02-01 01:27:58|5272574|   45305|063916192266113|GENUINE|  411|1.30402833099999999E7|
|6011985140563103|2018-01-30 02:03:54|1725430|   36587|914045782120401|GENUINE|  350|1.4569845000000002E7|
+----------------+-------------------+-------+--------+---------------+-------+-----+--------------------+
only showing top 20 rows
```

Drop duplicates on this DF to remove redundant transactions done of card_id, transaction date, score & post code.

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

```
lookup_table.count()  #1000
```

```
1000
```

4. Loading dataframe to lookup table.

NoSQL used as Hbase for this purpose to write bulk data into hbase tables.

Steps:
* Create a connection with Hbase
* Check if table you want to create already exists and create one if it doesn't exist
* Batch load data from dataframe to table created.

```
import happybase
#create connection
connection = happybase.Connection('localhost', port=9090 ,autoconnect=False)
```

Functions for Open Connection, close connection and list tables

```
def open_connection():
    connection.open()
#close the opened connection
def close_connection():
    connection.close()
#list all tables in Hbase
def list_tables():
    print "fetching all table"
    open_connection()
    tables = connection.tables()
    close_connection()
    print "all tables fetched"
    return tables
```

```
#create the required 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"
#get the pointer to a table
def get_table(name):
    open_connection()
    table = connection.table(name)
    close_connection()
    return table
```

```
create_table('lookup_table', {'info' : dict(max_versions=5) })
```

```
creating table lookup_table
fetching all table
all tables fetched
table created
```

```
#batch insert data in lookup table
def batch_insert_data(df,tableName):
 print "starting batch insert of events"
 table = get_table(tableName)
 open_connection()
 rows_count=0

#Creating a rowkey for better data query. RowKey is the cardId .
 rowKey_dict={}
 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()
```

```
batch_insert_data(lookup_table,'lookup_table')
```

```
starting batch insert of events
batch insert done
```

```
# create table of card_transactions.csv file.
create_table('card_transactions', {'info' : dict(max_versions=5) })
```

```
creating table card_transactions
fetching all table
all tables fetched
table created
```

```
def batch_insert_csvdata(filename,tableName):
    print "starting batch insert of events"
    file = open(filename, "r")
    table = get_table(tableName)
    open_connection()
    i=0

    for line in file:
        temp = line.strip().split(",")

        #Skip the first row
        if temp[0]!='card_id':

            table.put(bytes(i) , { 'info:card_id':bytes(temp[0]),
                                   'info:member_id':bytes(temp[1]),
                                   'info:amount':bytes(temp[2]),
                                   'info:postcode':bytes(temp[3]),
                                   'info:pos_id':bytes(temp[4]),
                                   'info:transaction_dt':bytes(temp[5]),
                                   'info:status':bytes(temp[6])})
        i=i+1



    file.close()
    print "batch insert done"
    close_connection()
```

```
#Batch insert data of card_transactions.csv file.
batch_insert_csvdata('card_transactions.csv','card_transactions')
```

```
starting batch insert of events
batch insert done
```

Validate the table created and data in Hbase:

1) Login to putty as root user
2) Start thrift server using below command
/opt/cloudera/parcels/CDH/lib/hbase/bin/hbase-daemon.sh start thrift -p 9090

3) Give command hbase shell
4) Give command “list”

```
hbase(main):007:0> list
TABLE
card_transactions
lookup_table
2 row(s) in 0.0070 seconds

=> ["card_transactions", "lookup_table"]
hbase(main):008:0>
```

root@ip-10-0-0-149:~

```
 6591175617713393                    column=info:transaction_date, timestamp=1634375007372, value=2018-01-31 13:10:37
 6592184145413632                    column=info:UCL, timestamp=1634375006787, value=13734342.65
 6592184145413632                    column=info:card_id, timestamp=1634375006787, value=6592184145413632
 6592184145413632                    column=info:postcode, timestamp=1634375006787, value=53186
 6592184145413632                    column=info:score, timestamp=1634375006787, value=456
 6592184145413632                    column=info:transaction_date, timestamp=1634375006787, value=2018-01-28 00:54:30
 6594248319343442                    column=info:UCL, timestamp=1634375006872, value=15065362.77
 6594248319343442                    column=info:card_id, timestamp=1634375006872, value=6594248319343442
 6594248319343442                    column=info:postcode, timestamp=1634375006872, value=24927
 6594248319343442                    column=info:score, timestamp=1634375006872, value=350
 6594248319343442                    column=info:transaction_date, timestamp=1634375006872, value=2018-01-31 23:42:38
 6595638658736751                    column=info:UCL, timestamp=1634375007621, value=14005069.97
 6595638658736751                    column=info:card_id, timestamp=1634375007621, value=6595638658736751
 6595638658736751                    column=info:postcode, timestamp=1634375007621, value=68328
 6595638658736751                    column=info:score, timestamp=1634375007621, value=310
 6595638658736751                    column=info:transaction_date, timestamp=1634375007621, value=2018-01-30 10:50:34
 6595814135833988                    column=info:UCL, timestamp=1634375007288, value=14332708.84
 6595814135833988                    column=info:card_id, timestamp=1634375007288, value=6595814135833988
 6595814135833988                    column=info:postcode, timestamp=1634375007288, value=22508
 6595814135833988                    column=info:score, timestamp=1634375007288, value=210
 6595814135833988                    column=info:transaction_date, timestamp=1634375007288, value=2018-01-30 02:03:54
 6595928469079750                    column=info:UCL, timestamp=1634375008323, value=11824730.01
 6595928469079750                    column=info:card_id, timestamp=1634375008323, value=6595928469079750
 6595928469079750                    column=info:postcode, timestamp=1634375008323, value=98349
 6595928469079750                    column=info:score, timestamp=1634375008323, value=412
 6595928469079750                    column=info:transaction_date, timestamp=1634375008323, value=2018-01-24 12:38:22
 6597703848279563                    column=info:UCL, timestamp=1634375007681, value=15250624.49
 6597703848279563                    column=info:card_id, timestamp=1634375007681, value=6597703848279563
 6597703848279563                    column=info:postcode, timestamp=1634375007681, value=95699
 6597703848279563                    column=info:score, timestamp=1634375007681, value=218
 6597703848279563                    column=info:transaction_date, timestamp=1634375007681, value=2018-01-27 10:51:49
 6598830758632447                    column=info:UCL, timestamp=1634375007878, value=12685782.48
 6598830758632447                    column=info:card_id, timestamp=1634375007878, value=6598830758632447
 6598830758632447                    column=info:postcode, timestamp=1634375007878, value=19421
 6598830758632447                    column=info:score, timestamp=1634375007878, value=293
 6598830758632447                    column=info:transaction_date, timestamp=1634375007878, value=2018-01-30 00:18:34
 6599900931314251                    column=info:UCL, timestamp=1634375008288, value=12487392.07
 6599900931314251                    column=info:card_id, timestamp=1634375008288, value=6599900931314251
 6599900931314251                    column=info:postcode, timestamp=1634375008288, value=97423
 6599900931314251                    column=info:score, timestamp=1634375008288, value=297
 6599900931314251                    column=info:transaction_date, timestamp=1634375008288, value=2018-01-31 11:25:16
999 row(s) in 0.4250 seconds

hbase(main):009:0>
```