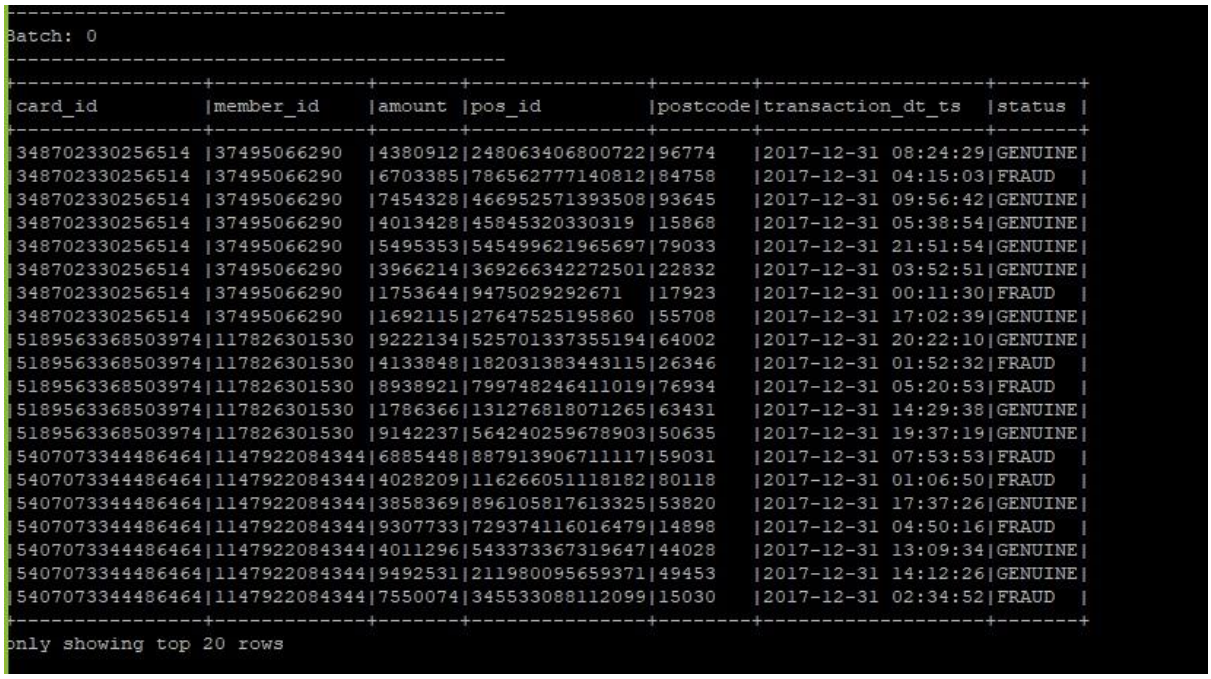
Logic Final

1) Import all necessary libraries and functions.
2) Define spark context and add .py files required along with csv given in resources list.
3) Connect to kafka topic using topic name “transactions-topic-verified” and server 18.211.252.152:9092.
4) Read the kafka stream into appropriate schema to make data readable.
5) Look Up Table Name: look_up_table
   Card Transaction table Name: card_transactions
6) Define following user defined functions to perform activities required for rule execution and determine if transaction is fraudulent or genuine.
    a. Name of function: ucl_data
       Input: CARD_ID
       Output: UCL from look_up_table
    b. Name of function: score_data
       Input: CARD_ID
       Output: Credit Score from look up table.
    c. Name of function: postcode_data
       Input: card_id
       Output: post code from look up table.
    d. Name of function: distance_calc
       Input: post codes from lookup table & kafka stream.
       Output: Distance between 2 locations of current transaction and previous transaction.
    e. Name of function: time_cal
       Input: transaction date from lookup table & kafka stream
       Output: difference between transaction dates in seconds.
    f. Name of function: lTransD_data
       Input: CARD_ID
       Output: transaction date from look up table.
    g. Name of function: speed_calc
       Input: Distance & Time calculated from above distance_calc & time_cal functions
       Output: Speed which is mathematically calculated by multiplying distance * 1000 and dividing by time.
    h. Name of function: status_res
       Input: Amount from current transaction read thru kafka stream, UCL from look up table, Credit_Score from look up table & Speed calculated from user defined functions.
       Output: Status as transaction if its genuine or fraud.

7) Execute above user define functions in same order given above. These functions perform us all required logic to deduce if transaction is fraud or genuine. These functions are agents to derive inputs to function status_res (function H).
8) Here are the rules performed on top of inputs supplied to function H.
    a. If current transaction amount is greater than UCL of look up table for that card_id, mark transaction as Fraud. Else, proceed to check below:
        i. If credit score of that card_id under process is less than 250, reject transaction as FRAUD. Else, proceed to check below:

1. If speed calculated is greater than 250, recognize the transaction as “FRAUD”. If speed is between 0 and 250, mark the transaction as genuine.

9) To summarize, a transaction is qualified to be genuine only when:
    a. Credit score of member is greater than 200,
    b. Speed is between 0 & 250
    c. Amount on current transaction is less than UCL calculated.
10) Functions “A”, “B”, “C”, “F” & “H” contact dao.py to call the look up table (table details given in point 5 above) for designated purposes.
    In process of calling dao.py from this driver.py file, I followed approach called “Import” which loads other .py files in same directory.
    Establish a spark context to add python files and csv files before we put the command import.
11) Function “D” uses geomap.py to calculate distance between last transaction & current transaction locations. This is in turn used in calculating speed which is one of factors for determining status of transaction.
12) Function “H” status_res also calls look_up_table using write_data function when transaction is genuine.
    Apart from this, it updates card_transactions table with latest information of posid, amount, transaction date and member ID.



```
-------------------------------------------
Batch: 0
-------------------------------------------
+----------------+-------------+-------+---------------+--------+-------------------+-------+
|card_id         |member_id    |amount |pos_id         |postcode|transaction_dt_ts  |status |
+----------------+-------------+-------+---------------+--------+-------------------+-------+
|348702330256514 |37495066290  |4380912|248063406800722|96774   |2017-12-31 08:24:29|GENUINE|
|348702330256514 |37495066290  |6703385|786562777140812|84758   |2017-12-31 04:15:03|FRAUD  |
|348702330256514 |37495066290  |7454328|466952571393508|93645   |2017-12-31 09:56:42|GENUINE|
|348702330256514 |37495066290  |4013428|45845320330319 |15868   |2017-12-31 05:38:54|GENUINE|
|348702330256514 |37495066290  |5495353|545499621965697|79033   |2017-12-31 21:51:54|GENUINE|
|348702330256514 |37495066290  |3966214|369266342272501|22832   |2017-12-31 03:52:51|GENUINE|
|348702330256514 |37495066290  |1753644|9475029292671  |17923   |2017-12-31 00:11:30|FRAUD  |
|348702330256514 |37495066290  |1692115|27647525195860 |55708   |2017-12-31 17:02:39|GENUINE|
|5189563368503974|117826301530 |9222134|5257013373551 94|64002   |2017-12-31 20:22:10|GENUINE|
|5189563368503974|117826301530 |4133848|182031383443115|26346   |2017-12-31 01:52:32|FRAUD  |
|5189563368503974|117826301530 |8938921|799748246411019|76934   |2017-12-31 05:20:53|FRAUD  |
|5189563368503974|117826301530 |1786366|131276818071265|63431   |2017-12-31 14:29:38|GENUINE|
|5189563368503974|117826301530 |9142237|5642402596789 03|50635   |2017-12-31 19:37:19|GENUINE|
|5407073344486464|1147922084344|6885448|887913906711117|59031   |2017-12-31 07:53:53|FRAUD  |
|5407073344486464|1147922084344|4028209|116266051118182|80118   |2017-12-31 01:06:50|FRAUD  |
|5407073344486464|1147922084344|3858369|896105817613325|53820   |2017-12-31 17:37:26|GENUINE|
|5407073344486464|1147922084344|9307733|729374116016479|14898   |2017-12-31 04:50:16|FRAUD  |
|5407073344486464|1147922084344|4011296|543373367319647|44028   |2017-12-31 13:09:34|GENUINE|
|5407073344486464|1147922084344|9492531|211980095659371|49453   |2017-12-31 14:12:26|GENUINE|
|5407073344486464|1147922084344|7550074|345533088112099|15030   |2017-12-31 02:34:52|FRAUD  |
+----------------+-------------+-------+---------------+--------+-------------------+-------+
only showing top 20 rows
```

```
Current count: 20000, row: 27999
Current count: 21000, row: 28899
Current count: 22000, row: 29799
Current count: 23000, row: 30698
Current count: 24000, row: 31598
Current count: 25000, row: 32498
Current count: 26000, row: 33398
Current count: 27000, row: 341724964458347.210778177559185.12-06-2018152638.2021-01-04171328.398477
Current count: 28000, row: 346618652451637.540752175696215.29-04-2018005259.2021-01-04171400.227023
Current count: 29000, row: 35264
Current count: 30000, row: 36164
Current count: 31000, row: 370582035866789.433646648625434.08-07-2018034337.2021-01-04171349.489639
Current count: 32000, row: 375806375521605.880937166605469.26-05-2018130045.2021-01-04171430.733012
Current count: 33000, row: 38176
Current count: 34000, row: 39076
Current count: 35000, row: 39977
Current count: 36000, row: 40768
Current count: 37000, row: 41560
Current count: 38000, row: 42387
Current count: 39000, row: 4318541450654035.496612742732167.12-02-2018145807.2021-01-04171356.009418
Current count: 40000, row: 43999
Current count: 41000, row: 44784
Current count: 42000, row: 45546
Current count: 43000, row: 46306
Current count: 44000, row: 47134
Current count: 45000, row: 47925
Current count: 46000, row: 48730
Current count: 47000, row: 49500
Current count: 48000, row: 50351
Current count: 49000, row: 5120
Current count: 50000, row: 51888
Current count: 51000, row: 5257502990314019.205172644364018.14-07-2018070014.2021-01-04171327.867742
Current count: 52000, row: 53290
Current count: 53000, row: 5620
Current count: 54000, row: 6211
Current count: 55000, row: 6478888441720966.273246841077378.06-10-2018212851.2021-01-04171333.585477
Current count: 56000, row: 6968
Current count: 57000, row: 7868
Current count: 58000, row: 8768
Current count: 59000, row: 9668
59367 row(s) in 3.8140 seconds

=> 59367
hbase(main):003:0>
```