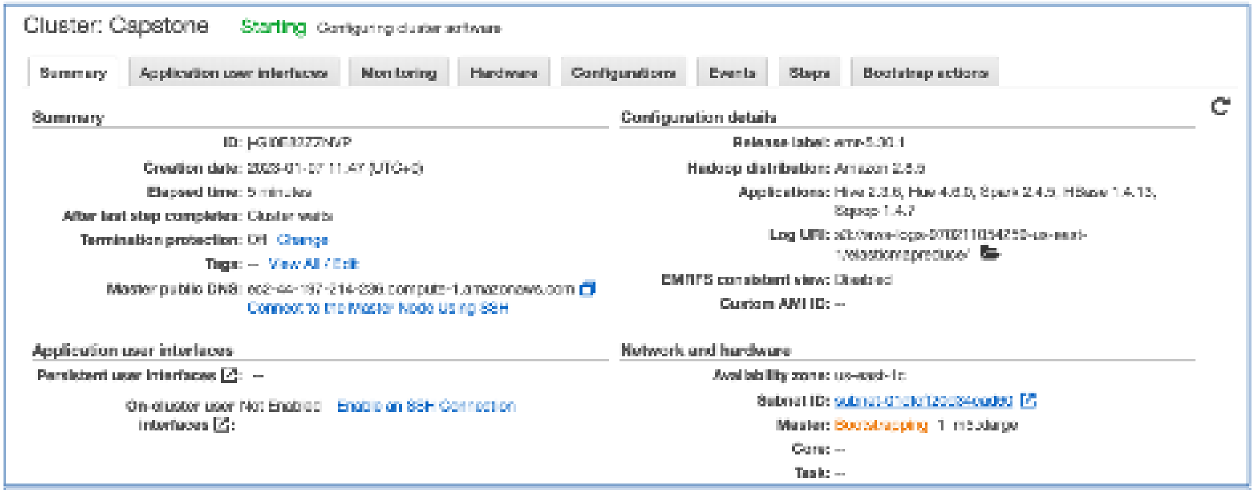
Final Submission: Scripts Execution

Explanation of the solution to the streaming layer problem

1. In order to complete below tasks, I have created EMR cluster with Hadoop, Sqoop, Hive, HBase, Hue and Spark, Root device EBS volume size as 20 GB. I have also updated the Yarn Configurations for EMR instance.

- **Task 5**: Create a streaming data processing framework that ingests real-time POS transaction data from Kafka. The transaction data is then validated based on the three rules' parameters (stored in the NoSQL database) discussed previously.
- **Task 6**: Update the transactions data along with the status (fraud/genuine) in the card transactions table.
- **Task 7**: Store the 'postcode' and 'transaction_dt' of the current transaction in the look-up table in the NoSQL database if the transaction was classified as genuine.

EMR Cluster Configuration:



Summary

ID: j-[illegible]
Creation date: 2023-01-07 11:47 (UTC+0)
Elapsed time: 5 minutes
After last step completes: Cluster waits
Termination protection: Off Change
Tags: -- View All / Edit
Master public DNS: ec2-44-197-214-236.compute-1.amazonaws.com
Connect to the Master Node Using SSH

Configuration details

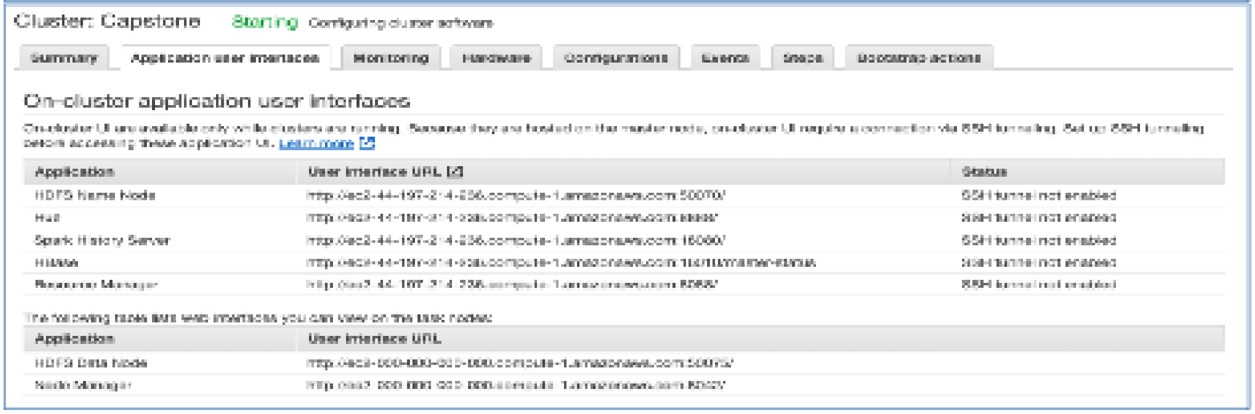
Release label: emr-5.30.1
Hadoop distribution: Amazon 2.8.5
Applications: Hive 2.3.6, Hue 4.6.0, Spark 2.4.5, HBase 1.4.13, Sqoop 1.4.7
Log URI: s3://aws-logs-[illegible]-us-east-1/elasticmapreduce/
EMRFS consistent view: Disabled
Custom AMI ID: --

Application user interfaces

Persistent user interfaces: --
On-cluster user interfaces: Not Enabled Enable an SSH Connection

Network and hardware

Availability zone: us-east-1c
Subnet ID: subnet-[illegible]
Master: Bootstrapping 1 m5.xlarge
Core: --
Task: --



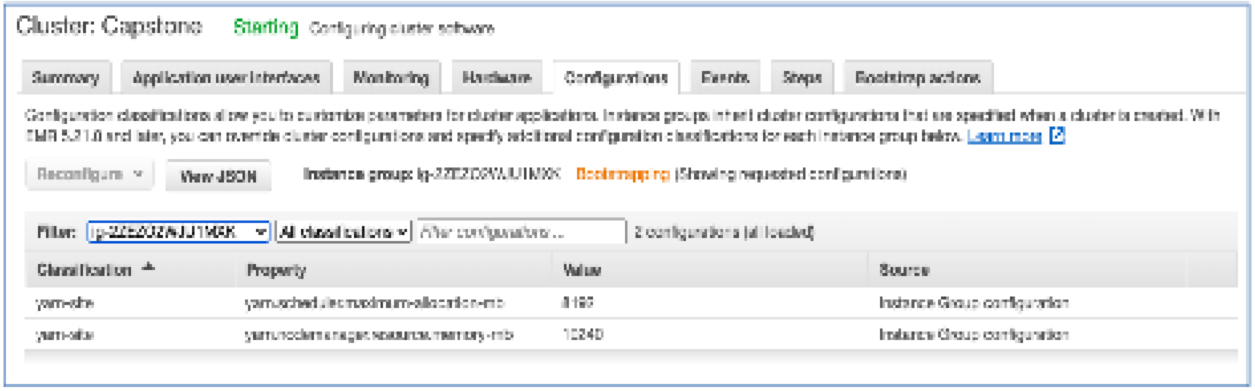
On-cluster application user interfaces

On-cluster UI are available only while clusters are running. Because they are hosted on the master node, on-cluster UI require a connection via SSH tunneling. Set up SSH tunneling before accessing these application UI. Learn more

Application	User interface URL	Status
HDFS Name Node	http://ec2-44-197-214-236.compute-1.amazonaws.com:50070/	SSH tunnel not enabled
Hue	http://ec2-44-197-214-236.compute-1.amazonaws.com:8888/	SSH tunnel not enabled
Spark History Server	http://ec2-44-197-214-236.compute-1.amazonaws.com:18080/	SSH tunnel not enabled
HBase	http://ec2-44-197-214-236.compute-1.amazonaws.com:16010/master-status	SSH tunnel not enabled
Resource Manager	http://ec2-44-197-214-236.compute-1.amazonaws.com:8088/	SSH tunnel not enabled

The following table lists web interfaces you can view on the task nodes:

Application	User interface URL
HDFS Data Node	http://ec2-###-###-###-###.compute-1.amazonaws.com:50075/
Node Manager	http://ec2-###-###-###-###.compute-1.amazonaws.com:8042/



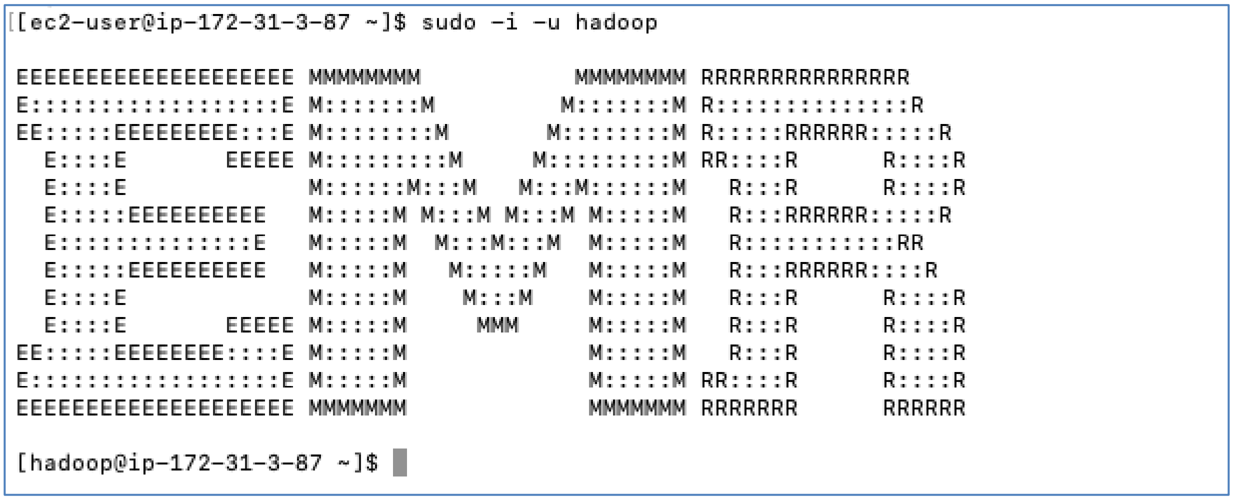
Configuration classifications allow you to customize parameters for cluster applications. Instance groups inherit cluster configurations that are specified when a cluster is created. With EMR 5.21.0 and later, you can override cluster configurations and specify additional configuration classifications for each instance group below. Learn more

Instance group: ig-2ZE2O2WJJ1MXK Bootstrap rp (Showing requested configurations)

Filter: ig-2ZE2O2WJJ1MXK | All classifications | 2 configurations (all loaded)

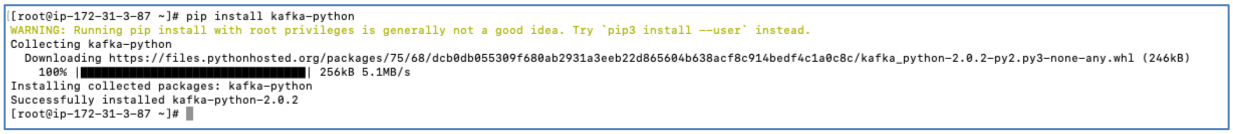
Classification	Property	Value	Source
yarn-site	yarn.scheduler.maximum-allocation-mb	8192	Instance Group configuration
yarn-site	yarn.nodemanager.resource.memory-mb	10240	Instance Group configuration

2. Logged into EMR instance as “hadoop” user:



```
[[ec2-user@ip-172-31-3-87 ~]$ sudo -i -u hadoop

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-3-87 ~]$ 
```

3. Switch to root user and run pip install kafka-python and then again use “sudo -i –u hadoop” to switch to hadoop user .

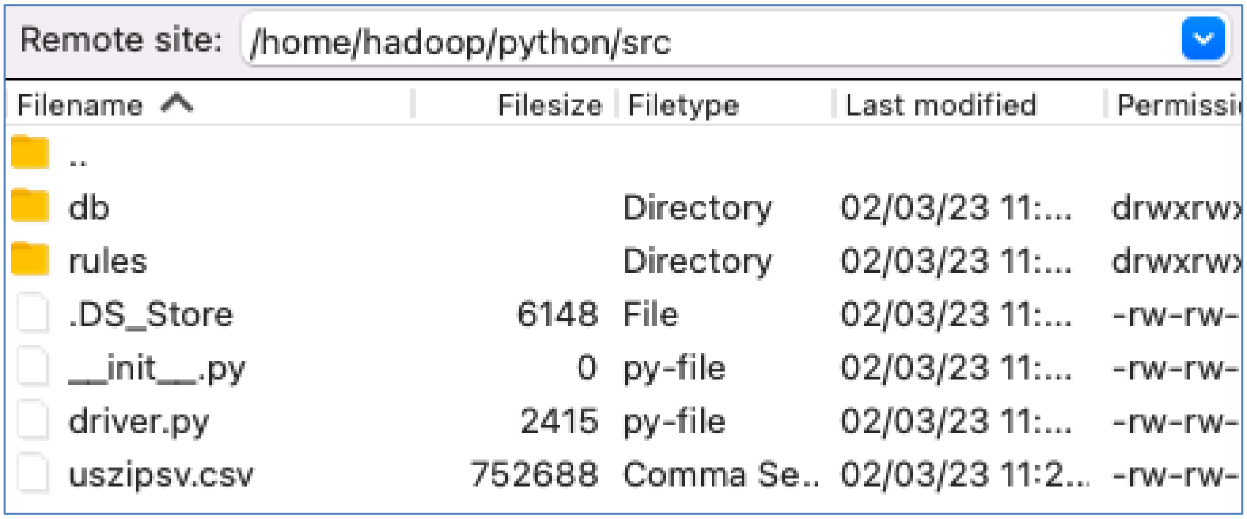


```
[[root@ip-172-31-3-87 ~]# pip install kafka-python
WARNING: Running pip install with root privileges is generally not a good idea. Try `pip3 install --user` instead.
Collecting kafka-python
  Downloading https://files.pythonhosted.org/packages/75/68/dcb0db055309f680ab2931a3eeb22d865604b638acf8c914bedf4c1a0c8c/kafka_python-2.0.2-py2.py3-none-any.whl (246kB)
    100% |████████████████████████████████| 256kB 5.1MB/s
Installing collected packages: kafka-python
Successfully installed kafka-python-2.0.2
[root@ip-172-31-3-87 ~]# 
```

4. Run the following commands in order to Install Happy base and start thrift server

- **sudo yum update**
- **sudo yum install python3-devel**
- **pip install happybase**
- **/usr/lib/hbase/bin/hbase-daemon.sh start thrift -p 9090**

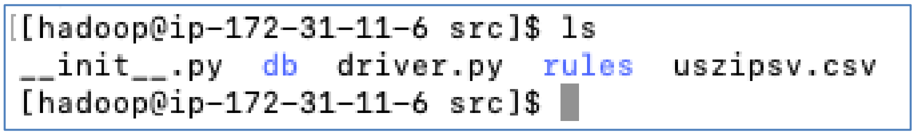
5. Download **db-> dao.py , geomap.py ,rules-> rules.py ,driver.py ,unzipsv.csv** from the resource section of the capstone project on the learning platform and transfer it to hadoop instance via FileZilla.



Remote site: /home/hadoop/python/src

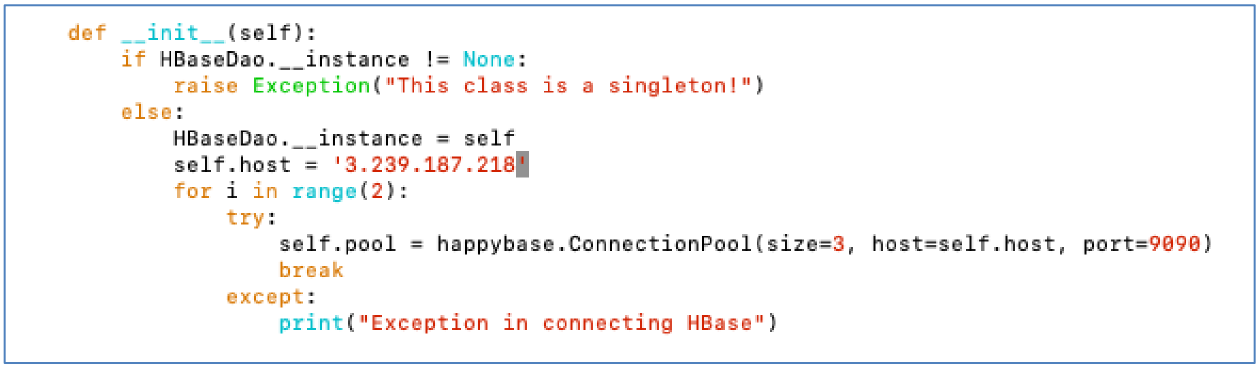
Filename	Filesize	Filetype	Last modified	Permissi
..				
db		Directory	02/03/23 11:...	drwxrwx
rules		Directory	02/03/23 11:...	drwxrwx
.DS_Store	6148	File	02/03/23 11:...	-rw-rw-
__init__.py	0	py-file	02/03/23 11:...	-rw-rw-
driver.py	2415	py-file	02/03/23 11:...	-rw-rw-
uszipsv.csv	752688	Comma Se..	02/03/23 11:2...	-rw-rw-

Checking if imported correctly



```
[hadoop@ip-172-31-11-6 src]$ ls
__init__.py  db  driver.py  rules  uszipsv.csv
[hadoop@ip-172-31-11-6 src]$
```

6. Updated the Public IP of your EC2 Instance “**3.239.187.218**”(self.host) in **dao.py** file

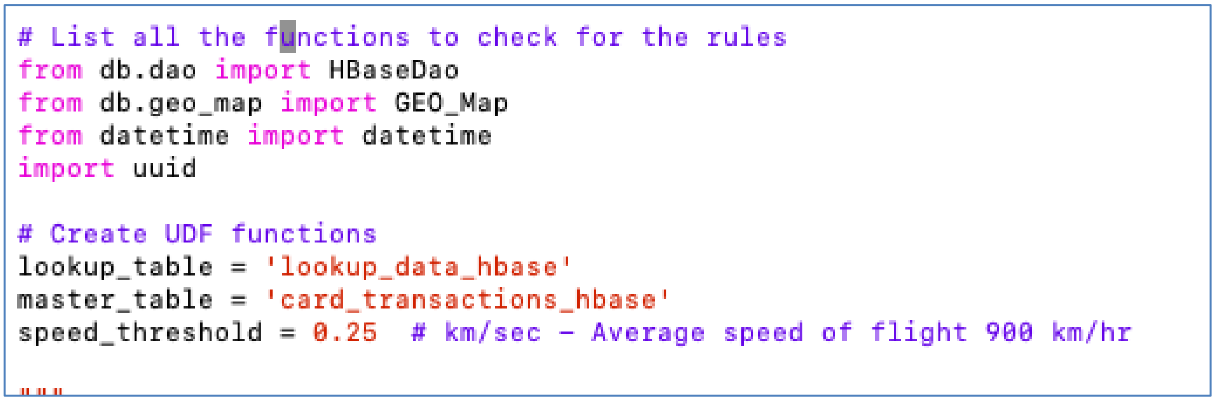


```
    def __init__(self):
        if HBaseDao.__instance != None:
            raise Exception("This class is a singleton!")
        else:
            HBaseDao.__instance = self
            self.host = '3.239.187.218'
            for i in range(2):
                try:
                    self.pool = happybase.ConnectionPool(size=3, host=self.host, port=9090)
                    break
                except:
                    print("Exception in connecting HBase")
```

7. Updated rules.py with following parameters:

   lookup_table = 'lookup_data_hbase'

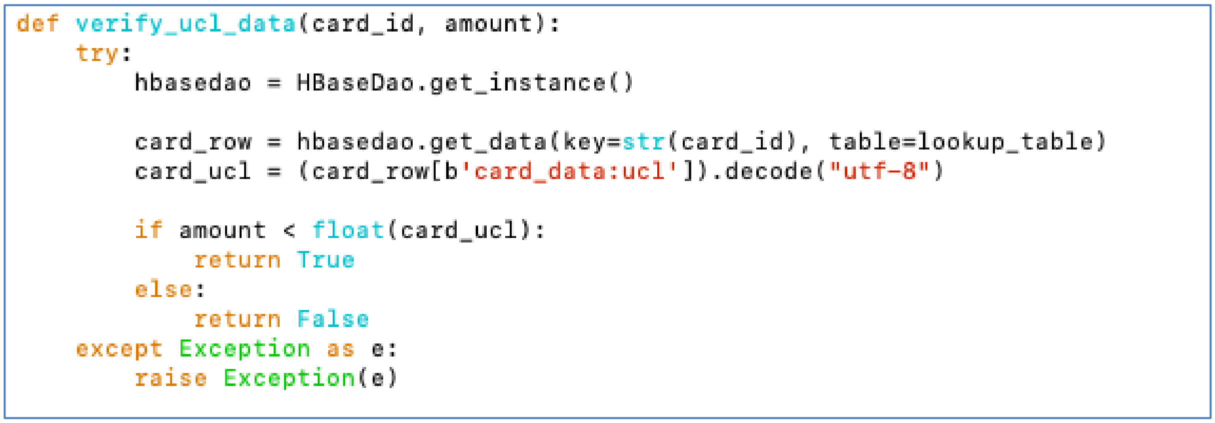
   master_table = 'card_transactions_hbase'



```
# List all the functions to check for the rules
from db.dao import HBaseDao
from db.geo_map import GEO_Map
from datetime import datetime
import uuid

# Create UDF functions
lookup_table = 'lookup_data_hbase'
master_table = 'card_transactions_hbase'
speed_threshold = 0.25  # km/sec - Average speed of flight 900 km/hr

"""
```

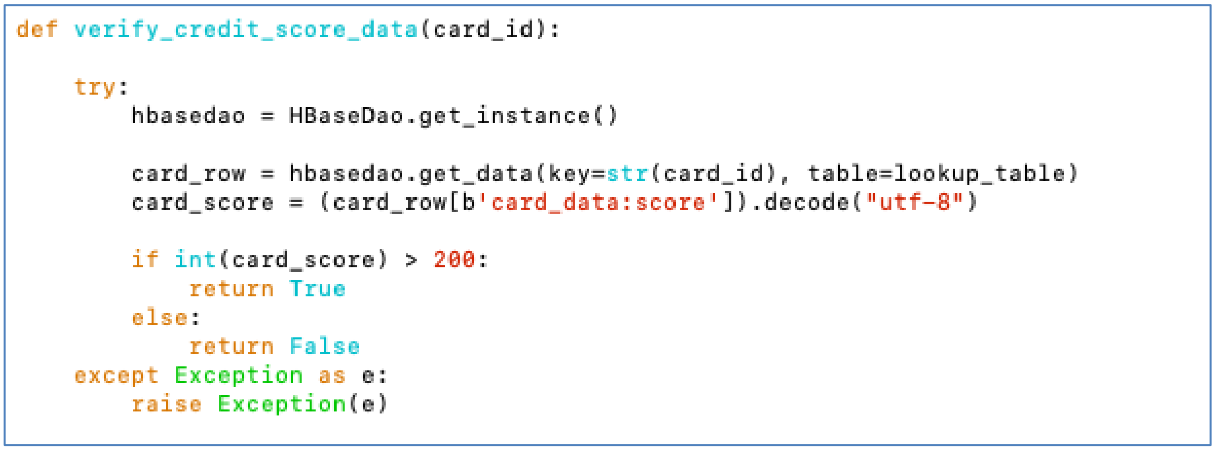
8. Created Python functions, containing the logic for the UDFs (rules.py)

- **verify_ucl_data** : Function to verify the UCL(upper control limit) rule Transaction amount should be less than (Upper control limit) UCL



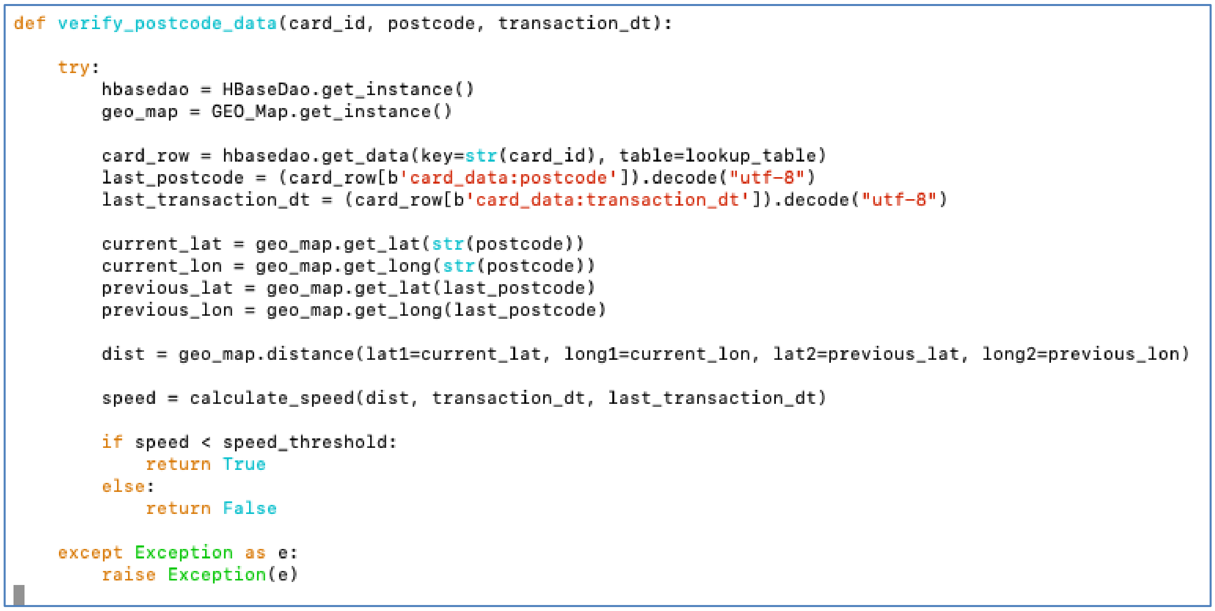
```
def verify_ucl_data(card_id, amount):
    try:
        hbasedao = HBaseDao.get_instance()

        card_row = hbasedao.get_data(key=str(card_id), table=lookup_table)
        card_ucl = (card_row[b'card_data:ucl']).decode("utf-8")

        if amount < float(card_ucl):
            return True
        else:
            return False
    except Exception as e:
        raise Exception(e)
```

- **verify_credit_score_data:** Function to verify the credit score rule .Credit score of each member should be greater than 200



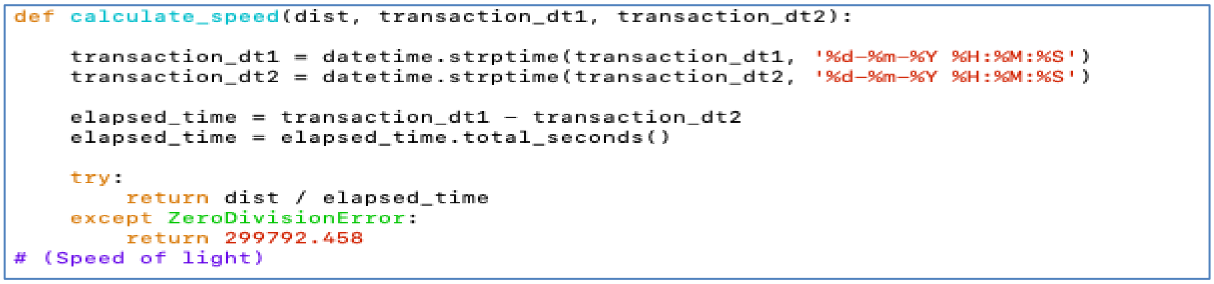
```
def verify_credit_score_data(card_id):

    try:
        hbasedao = HBaseDao.get_instance()

        card_row = hbasedao.get_data(key=str(card_id), table=lookup_table)
        card_score = (card_row[b'card_data:score']).decode("utf-8")

        if int(card_score) > 200:
            return True
        else:
            return False
    except Exception as e:
        raise Exception(e)
```

- **verify_postcode_data:** Function to verify the following postcode rules.ZIP code distance



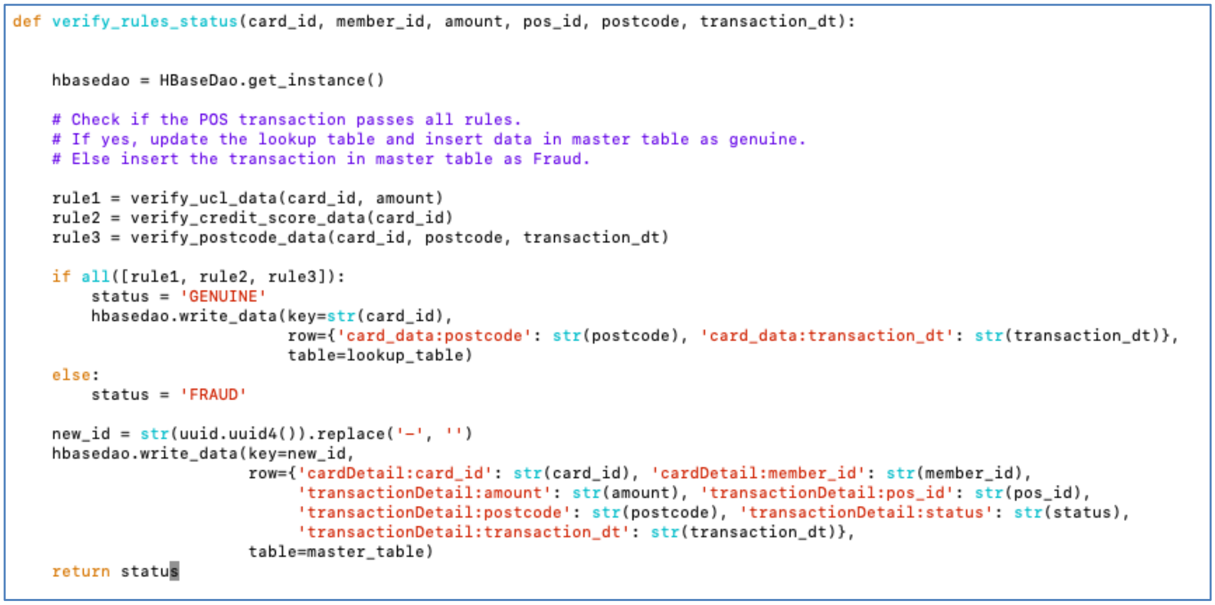
```
def verify_postcode_data(card_id, postcode, transaction_dt):

    try:
        hbasedao = HBaseDao.get_instance()
        geo_map = GEO_Map.get_instance()

        card_row = hbasedao.get_data(key=str(card_id), table=lookup_table)
        last_postcode = (card_row[b'card_data:postcode']).decode("utf-8")
        last_transaction_dt = (card_row[b'card_data:transaction_dt']).decode("utf-8")

        current_lat = geo_map.get_lat(str(postcode))
        current_lon = geo_map.get_long(str(postcode))
        previous_lat = geo_map.get_lat(last_postcode)
        previous_lon = geo_map.get_long(last_postcode)

        dist = geo_map.distance(lat1=current_lat, long1=current_lon, lat2=previous_lat, long2=previous_lon)

        speed = calculate_speed(dist, transaction_dt, last_transaction_dt)

        if speed < speed_threshold:
            return True
        else:
            return False

    except Exception as e:
        raise Exception(e)
```

- **calculate_speed :** A function to calculate the speed from distance and transaction timestamp differentials



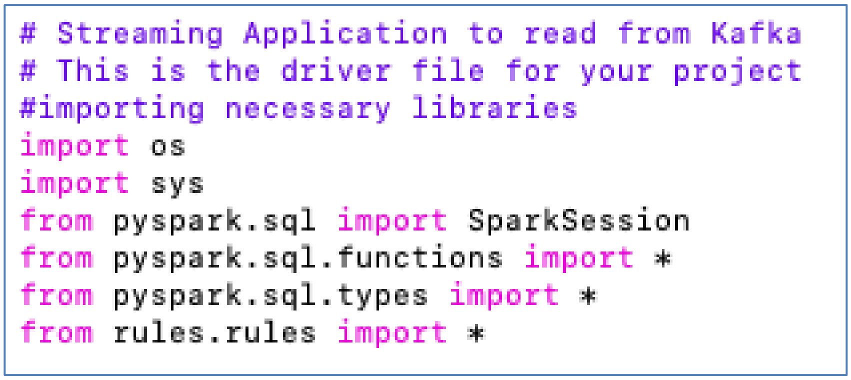
```
def calculate_speed(dist, transaction_dt1, transaction_dt2):

    transaction_dt1 = datetime.strptime(transaction_dt1, '%d-%m-%Y %H:%M:%S')
    transaction_dt2 = datetime.strptime(transaction_dt2, '%d-%m-%Y %H:%M:%S')

    elapsed_time = transaction_dt1 - transaction_dt2
    elapsed_time = elapsed_time.total_seconds()

    try:
        return dist / elapsed_time
    except ZeroDivisionError:
        return 299792.458
# (Speed of light)
```

- **verify_rules_status:** A function to verify all the three rules - ucl, credit score and speed



```
def verify_rules_status(card_id, member_id, amount, pos_id, postcode, transaction_dt):


    hbasedao = HBaseDao.get_instance()

    # Check if the POS transaction passes all rules.
    # If yes, update the lookup table and insert data in master table as genuine.
    # Else insert the transaction in master table as Fraud.

    rule1 = verify_ucl_data(card_id, amount)
    rule2 = verify_credit_score_data(card_id)
    rule3 = verify_postcode_data(card_id, postcode, transaction_dt)

    if all([rule1, rule2, rule3]):
        status = 'GENUINE'
        hbasedao.write_data(key=str(card_id),
                            row={'card_data:postcode': str(postcode), 'card_data:transaction_dt': str(transaction_dt)},
                            table=lookup_table)
    else:
        status = 'FRAUD'

    new_id = str(uuid.uuid4()).replace('-', '')
    hbasedao.write_data(key=new_id,
                        row={'cardDetail:card_id': str(card_id), 'cardDetail:member_id': str(member_id),
                             'transactionDetail:amount': str(amount), 'transactionDetail:pos_id': str(pos_id),
                             'transactionDetail:postcode': str(postcode), 'transactionDetail:status': str(status),
                             'transactionDetail:transaction_dt': str(transaction_dt)},
                        table=master_table)
    return status
```

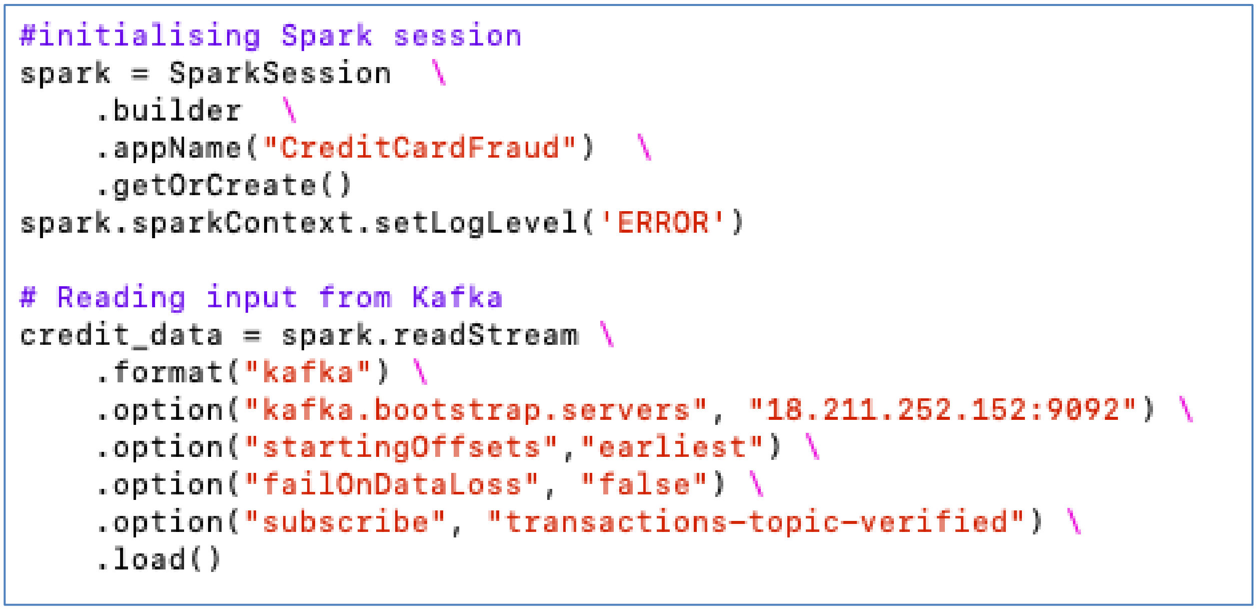
9. Next, update the ‘driver.py’ file with the following code
   Setting up the system dependencies and importing necessary libraries and modules



```
# Streaming Application to read from Kafka
# This is the driver file for your project
#importing necessary libraries
import os
import sys
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
from rules.rules import *
```

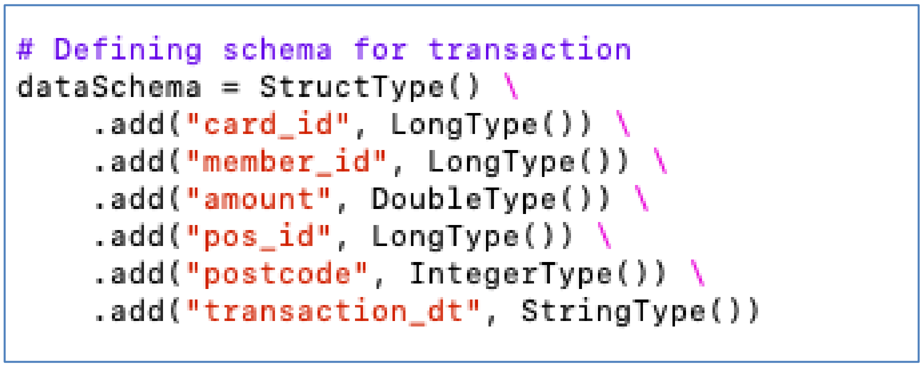
10. Initializing the Spark session and reading input data from Kafka mentioning the details of the Kafka broker, such as bootstrap server, port and topic name.

- Connect to kafka topic using below details :
  Bootstrap-server: 18.211.252.152
  Port Number: 9092
  Topic: transactions-topic-verified



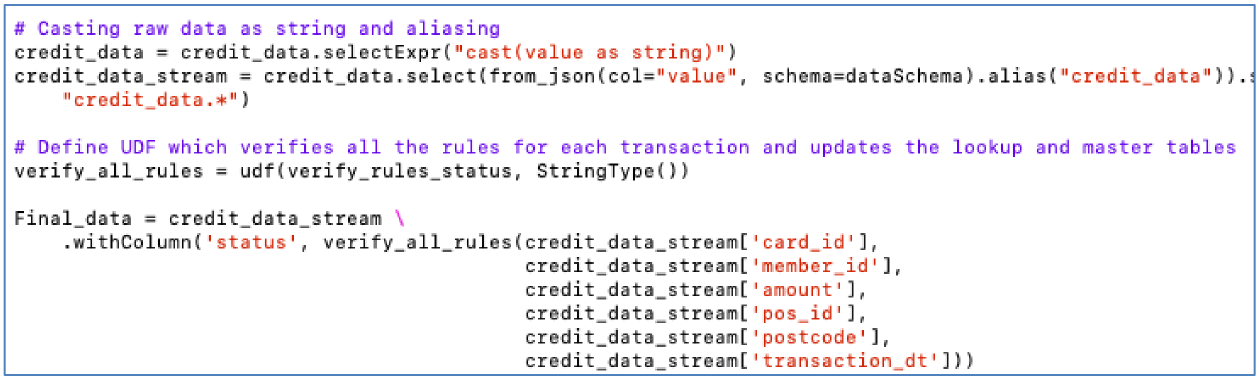
```
#initialising Spark session
spark = SparkSession  \
    .builder  \
    .appName("CreditCardFraud")  \
    .getOrCreate()
spark.sparkContext.setLogLevel('ERROR')

# Reading input from Kafka
credit_data = spark.readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "18.211.252.152:9092") \
    .option("startingOffsets","earliest") \
    .option("failOnDataLoss", "false") \
    .option("subscribe", "transactions-topic-verified") \
    .load()
```

11. Define JSON schema of each transactions from kafka topic.



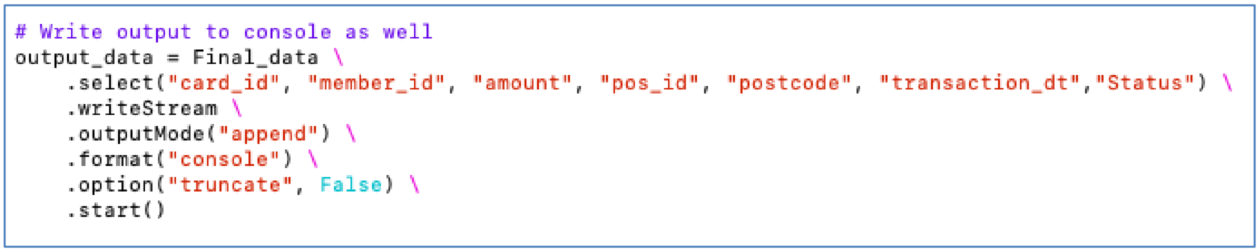
```
# Defining schema for transaction
dataSchema = StructType() \
    .add("card_id", LongType()) \
    .add("member_id", LongType()) \
    .add("amount", DoubleType()) \
    .add("pos_id", LongType()) \
    .add("postcode", IntegerType()) \
    .add("transaction_dt", StringType())
```

12. Read the raw JSON data from Kafka as ‘credit_data_stream’ and Define UDF’s to verify rules and also updates the lookup table and master table accordingly as coded in verify_rules_status.



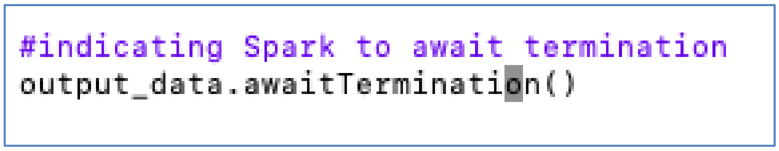
```
# Casting raw data as string and aliasing
credit_data = credit_data.selectExpr("cast(value as string)")
credit_data_stream = credit_data.select(from_json(col="value", schema=dataSchema).alias("credit_data")).
    "credit_data.*")

# Define UDF which verifies all the rules for each transaction and updates the lookup and master tables
verify_all_rules = udf(verify_rules_status, StringType())

Final_data = credit_data_stream \
    .withColumn('status', verify_all_rules(credit_data_stream['card_id'],
                                           credit_data_stream['member_id'],
                                           credit_data_stream['amount'],
                                           credit_data_stream['pos_id'],
                                           credit_data_stream['postcode'],
                                           credit_data_stream['transaction_dt']))
```

13. Displaying the output to the console



```
# Write output to console as well
output_data = Final_data \
     .select("card_id", "member_id", "amount", "pos_id", "postcode", "transaction_dt","Status") \
     .writeStream \
     .outputMode("append") \
     .format("console") \
     .option("truncate", False) \
     .start()
```

14. Define spark termination



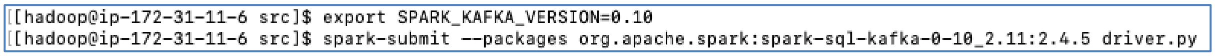
```
#indicating Spark to await termination
output_data.awaitTermination()
```

15. Set the Kafka Version using the following command

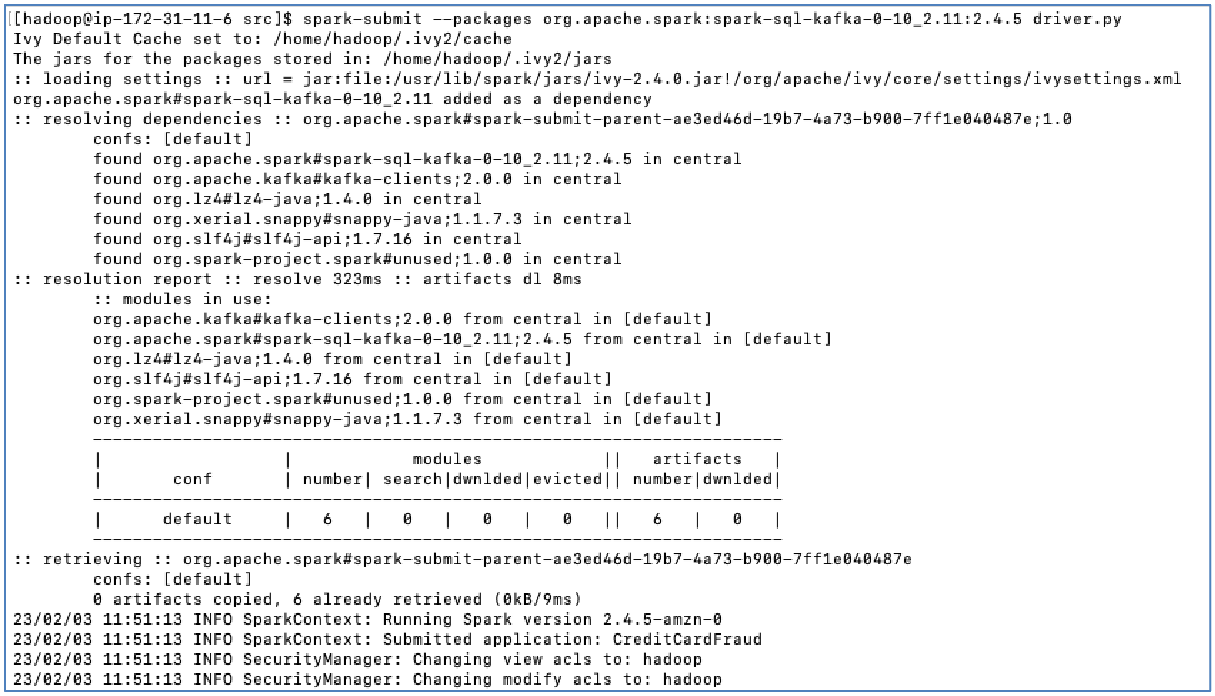
export SPARK_KAFKA_VERSION=0.10

16. Run the spark-submit command, specifying the Spark-SQL-Kafka package and python driver file

spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 driver.py

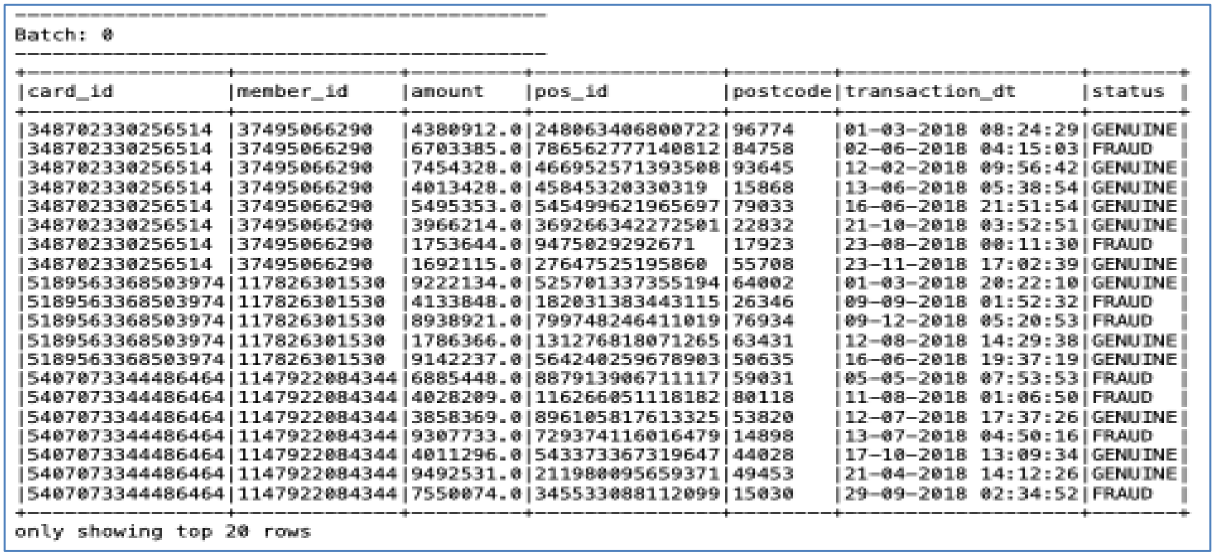


```
[[hadoop@ip-172-31-11-6 src]$ export SPARK_KAFKA_VERSION=0.10
[[hadoop@ip-172-31-11-6 src]$ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 driver.py
```



```
[[hadoop@ip-172-31-11-6 src]$ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 driver.py
Ivy Default Cache set to: /home/hadoop/.ivy2/cache
The jars for the packages stored in: /home/hadoop/.ivy2/jars
:: loading settings :: url = jar:file:/usr/lib/spark/jars/ivy-2.4.0.jar!/org/apache/ivy/core/settings/ivysettings.xml
org.apache.spark#spark-sql-kafka-0-10_2.11 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-ae3ed46d-19b7-4a73-b900-7ff1e040487e;1.0
        confs: [default]
        found org.apache.spark#spark-sql-kafka-0-10_2.11;2.4.5 in central
        found org.apache.kafka#kafka-clients;2.0.0 in central
        found org.lz4#lz4-java;1.4.0 in central
        found org.xerial.snappy#snappy-java;1.1.7.3 in central
        found org.slf4j#slf4j-api;1.7.16 in central
        found org.spark-project.spark#unused;1.0.0 in central
:: resolution report :: resolve 323ms :: artifacts dl 8ms
        :: modules in use:
        org.apache.kafka#kafka-clients;2.0.0 from central in [default]
        org.apache.spark#spark-sql-kafka-0-10_2.11;2.4.5 from central in [default]
        org.lz4#lz4-java;1.4.0 from central in [default]
        org.slf4j#slf4j-api;1.7.16 from central in [default]
        org.spark-project.spark#unused;1.0.0 from central in [default]
        org.xerial.snappy#snappy-java;1.1.7.3 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   6   |   0   |   0   |   0   ||   6   |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-ae3ed46d-19b7-4a73-b900-7ff1e040487e
        confs: [default]
        0 artifacts copied, 6 already retrieved (0kB/9ms)
23/02/03 11:51:13 INFO SparkContext: Running Spark version 2.4.5-amzn-0
23/02/03 11:51:13 INFO SparkContext: Submitted application: CreditCardFraud
23/02/03 11:51:13 INFO SecurityManager: Changing view acls to: hadoop
23/02/03 11:51:13 INFO SecurityManager: Changing modify acls to: hadoop
```

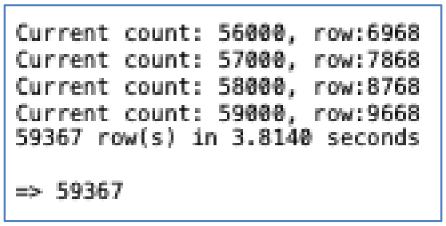
17. Check Output in console :



```
-------------------------------------------
Batch: 0
-------------------------------------------
+----------------+-------------+---------+---------------+--------+-------------------+-------+
|card_id         |member_id    |amount   |pos_id         |postcode|transaction_dt     |status |
+----------------+-------------+---------+---------------+--------+-------------------+-------+
|348702330256514 |37495066290  |4380912.0|248063406800722|96774   |01-03-2018 08:24:29|GENUINE|
|348702330256514 |37495066290  |6703385.0|786562777140812|84758   |02-06-2018 04:15:03|FRAUD  |
|348702330256514 |37495066290  |7454328.0|466952571393508|93645   |12-02-2018 09:56:42|GENUINE|
|348702330256514 |37495066290  |4013428.0|45845320330319 |15868   |13-06-2018 05:38:54|GENUINE|
|348702330256514 |37495066290  |5495353.0|545499621965697|79033   |16-06-2018 21:51:54|GENUINE|
|348702330256514 |37495066290  |3966214.0|369266342272501|22832   |21-10-2018 03:52:51|GENUINE|
|348702330256514 |37495066290  |1753644.0|9475029292671  |17923   |23-08-2018 00:11:30|FRAUD  |
|348702330256514 |37495066290  |1692115.0|27647525195860 |55708   |23-11-2018 17:02:39|GENUINE|
|5189563368503974|117826301530 |9222134.0|525701337355194|64002   |01-03-2018 20:22:10|GENUINE|
|5189563368503974|117826301530 |4133848.0|182031383443115|26346   |09-09-2018 01:52:32|FRAUD  |
|5189563368503974|117826301530 |8938921.0|799748246411019|76934   |09-12-2018 05:20:53|FRAUD  |
|5189563368503974|117826301530 |1786366.0|131276818071265|63431   |12-08-2018 14:29:38|GENUINE|
|5189563368503974|117826301530 |9142237.0|564240259678903|50635   |16-06-2018 19:37:19|GENUINE|
|5407073344486464|1147922084344|6885448.0|887913906711117|59031   |05-05-2018 07:53:53|FRAUD  |
|5407073344486464|1147922084344|4028209.0|116266051118182|80118   |11-08-2018 01:06:50|FRAUD  |
|5407073344486464|1147922084344|3858369.0|896105817613325|53820   |12-07-2018 17:37:26|GENUINE|
|5407073344486464|1147922084344|9307733.0|729374116016479|14898   |13-07-2018 04:50:16|FRAUD  |
|5407073344486464|1147922084344|4011296.0|543373367319647|44028   |17-10-2018 13:09:34|GENUINE|
|5407073344486464|1147922084344|9492531.0|211980095659371|49453   |21-04-2018 14:12:26|GENUINE|
|5407073344486464|1147922084344|7550074.0|345533088112099|15030   |29-09-2018 02:34:52|FRAUD  |
+----------------+-------------+---------+---------------+--------+-------------------+-------+
only showing top 20 rows
```

18. Count Data in ‘card_transaction_hive’ in Hbase:

count ‘card_transactions_hive’



```
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
```

Total number for record is **59367** which is matching with given requirement of records more than 59000 in card_transactions_hbase.