Final-Submission - Scripts Execution

Explanation of the solution to the streaming layer problem

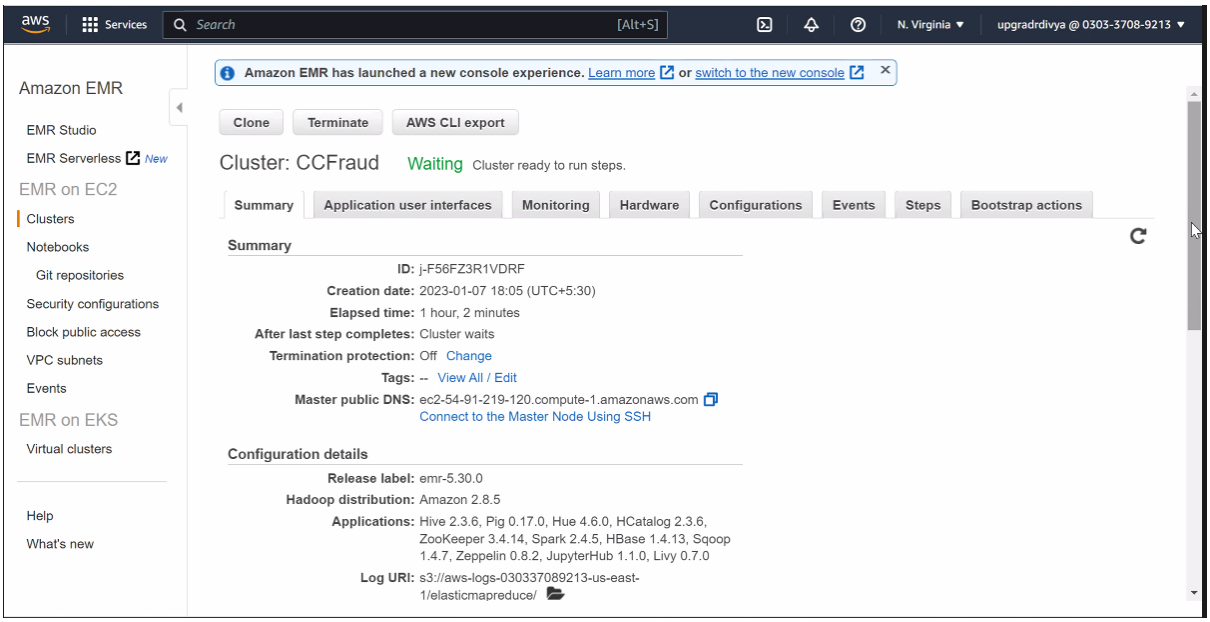
- For the below mentioned tasks, an EMR cluster with **Hadoop, Sqoop, Hive, HBase and Spark,** Root device EBS volume size **as** 20 GB was created.
- Tasks summary –

• **Task 5**: A streaming data processing framework that ingests real-time POS transaction data from Kafka was created. Further, the transaction data was validated based on the three rules' parameters (stored in the NoSQL database) discussed previously.

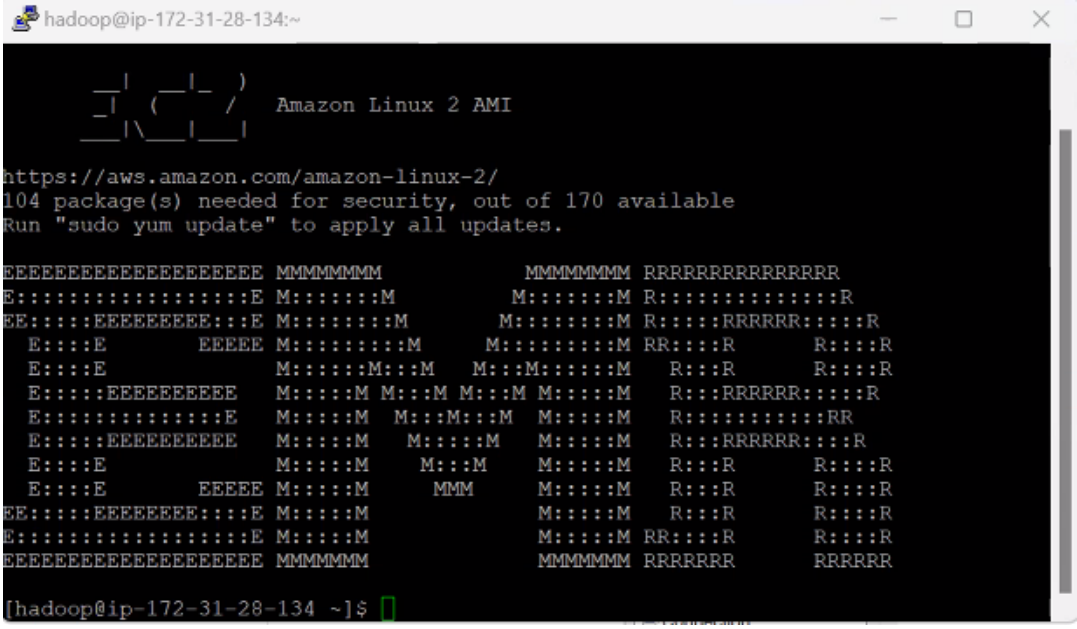
• **Task 6**: Updating the transactions data along with the status (fraud/genuine) in the card_transactions table.

• **Task 7**: Storing the 'postcode' and 'transaction_dt' of the current transaction in the look-up table of the NoSQL database if the transaction was classified as genuine.

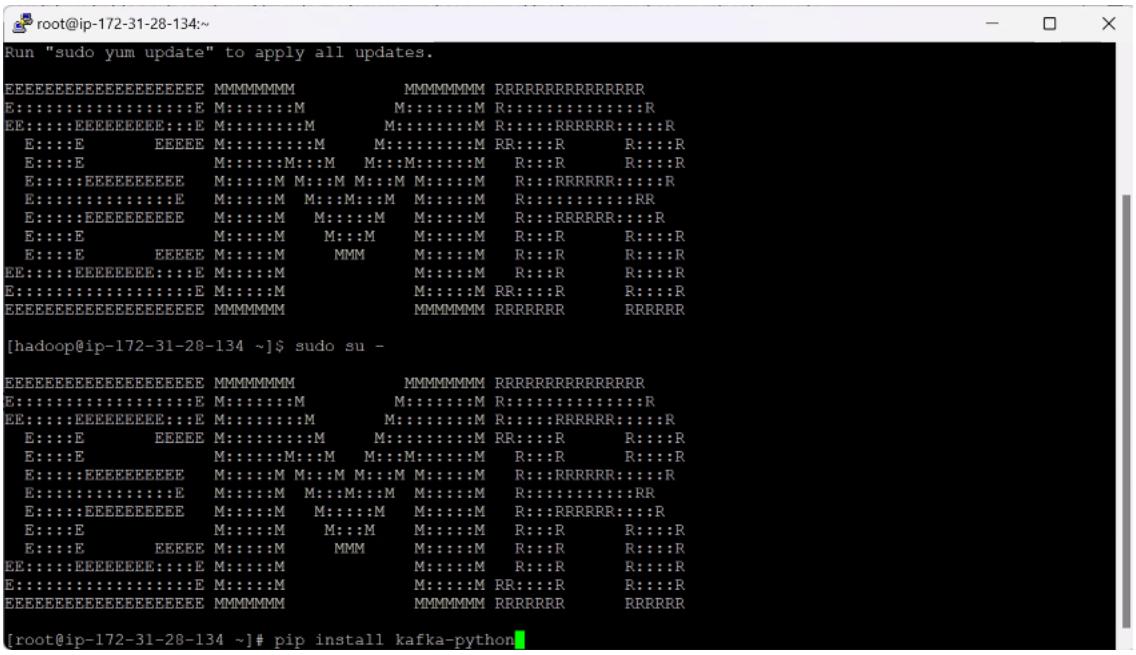
EMR Cluster Configuration:



1. Logging into EMR instance as “hadoop”:



2. Switching to root user and running pip install kafka-python and further running “sudo -i –u hadoop” to be a hadoop user



3. Running 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**

hadoop@ip-172-31-28-134:~

```
  rubygem-psych.x86_64 0:2.0.0-36.amzn2.0.3
  rubygem-rdoc.noarch 0:4.0.0-36.amzn2.0.3
  rubygems.noarch 0:2.0.14.1-36.amzn2.0.3
  selinux-policy.noarch 0:3.13.1-192.amzn2.6.8
  selinux-policy-targeted.noarch 0:3.13.1-192.amzn2.6.8
  strace.x86_64 0:4.26-1.amzn2.0.1
  sysctl-defaults.noarch 0:1.0-3.amzn2
  system-lsb.x86_64 0:4.1-27.amzn2.3.6
  system-lsb-core.x86_64 0:4.1-27.amzn2.3.6
  system-lsb-cxx.x86_64 0:4.1-27.amzn2.3.6
  system-lsb-desktop.x86_64 0:4.1-27.amzn2.3.6
  system-lsb-languages.x86_64 0:4.1-27.amzn2.3.6
  system-lsb-submod-multimedia.x86_64 0:4.1-27.amzn2.3.6
  system-lsb-submod-security.x86_64 0:4.1-27.amzn2.3.6
  system-release.x86_64 1:2-14.amzn2
  systemtap-runtime.x86_64 0:4.5-1.amzn2.0.1
  systemtap-sdt-devel.x86_64 0:4.5-1.amzn2.0.1
  texlive-base.noarch 2:2012-38.20130427_r30134.amzn2.0.5
  texlive-dvipng.noarch 2:svn26689.1.14-38.amzn2.0.5
  texlive-kpathsea.noarch 2:svn28792.0-38.amzn2.0.5
  tkinter.x86_64 0:2.7.18-1.amzn2.0.5
  tzdata.noarch 0:2022f-1.amzn2.0.1
  update-motd.noarch 0:1.1.2-2.amzn2.0.2
  xfsprogs.x86_64 0:5.0.0-10.amzn2.0.1
  yum.noarch 0:3.4.3-158.amzn2.0.6

Replaced:
  grub2.x86_64 1:2.02-35.amzn2.0.4                          grub2-tools.x86_64 1:2.02-35.amzn2.0.4
  python-colorama.noarch 0:0.3.2-3.amzn2                    python-six.noarch 0:1.9.0-2.amzn2
  system-lsb-printing.x86_64 0:4.1-27.amzn2.3.5

Complete!
[hadoop@ip-172-31-28-134 ~]$
```

hadoop@ip-172-31-28-134:~

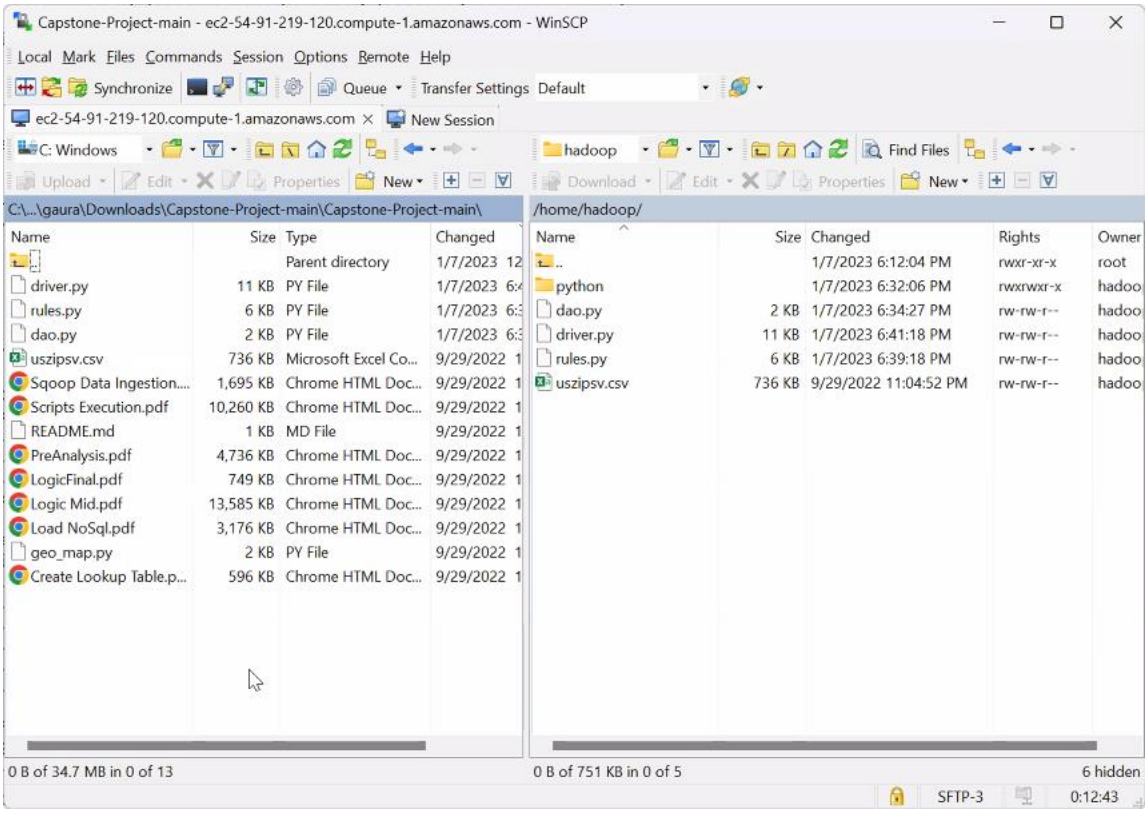
```
Total download size: 288 k
Installed size: 753 k
Is this ok [y/d/N]: y
Downloading packages:
(1/4): python-srpm-macros-3-60.amzn2.0.1.noarch.rpm                          |  18 kB  00:00:00
(2/4): python-rpm-macros-3-60.amzn2.0.1.noarch.rpm                           |  14 kB  00:00:00
(3/4): python3-rpm-macros-3-60.amzn2.0.1.noarch.rpm                          |  12 kB  00:00:00
(4/4): python3-devel-3.7.15-1.amzn2.0.2.x86_64.rpm                           | 244 kB  00:00:00
--------------------------------------------------------------------------------------------------
Total                                                            2.6 MB/s | 288 kB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : python-srpm-macros-3-60.amzn2.0.1.noarch                                    1/4
  Installing : python-rpm-macros-3-60.amzn2.0.1.noarch                                     2/4
  Installing : python3-rpm-macros-3-60.amzn2.0.1.noarch                                    3/4
  Installing : python3-devel-3.7.15-1.amzn2.0.2.x86_64                                     4/4
  Verifying  : python-rpm-macros-3-60.amzn2.0.1.noarch                                     1/4
  Verifying  : python3-rpm-macros-3-60.amzn2.0.1.noarch                                    2/4
  Verifying  : python3-devel-3.7.15-1.amzn2.0.2.x86_64                                     3/4
  Verifying  : python-srpm-macros-3-60.amzn2.0.1.noarch                                    4/4

Installed:
  python3-devel.x86_64 0:3.7.15-1.amzn2.0.2

Dependency Installed:
  python-rpm-macros.noarch 0:3-60.amzn2.0.1              python-srpm-macros.noarch 0:3-60.amzn2.0.1
  python3-rpm-macros.noarch 0:3-60.amzn2.0.1

Complete!
[hadoop@ip-172-31-28-134 ~]$
```

4. Downloaded files **db-> dao.py , geomap.py ,rules-> rules.py ,driver.py ,unzipsv.csv** from the resource section of the capstone project and transferred it to hadoop instance via WinSCP.




```
[hadoop@ip-172-31-28-134 ~]$ ls
dao.py  driver.py  python  rules.py  uszipsv.csv
[hadoop@ip-172-31-28-134 ~]$
```

5. Updated the Public IP of the EC2 Instance "**10.100.84.25**" (self.host) in **dao.py** file.

```
import happybase

class HBaseDao:
        """
        Dao class for operation on HBase
        """
        __instance = None

        @staticmethod
        def get_instance():
                """ Static access method. """
                if HBaseDao.__instance == None:
                        HBaseDao()
                return HBaseDao.__instance

        def __init__(self):
                if HBaseDao.__instance != None:
                        raise Exception("This class is a singleton!")
                else:
                        HBaseDao.__instance = self
                        self.host = '10.100.84.25'
                        for i in range(2):
                                try:
                                        self.pool = happybase.ConnectionPool(siz
e=3, host=self.host, port=9090)
                                        break
                                except:
                                        print("Exception in connecting HBase")


        def get_data(self, key, table):
                for i in range(2):
                        try:
                                with self.pool.connection() as connection:
```

6. Updated **rules.py** with following parameters:

   lookup_table = 'CC_LOOKUP_DATA_HBASE'
   master_table = 'CC_ 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 = 'CC_LOOKUP_DATA_HBASE'
master_table = 'CC_transactions_hbase'
speed_threshold = 0.25  # km/sec - Average speed of flight 900 km/hr
```

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

   verify_ucl_data : Function to verify the UCL rule that the 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 for 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 zipcode 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 : Function to calculate the speed from distance and transaction timestamp differentials.

```
hadoop@ip-172-31-28-134:~
    except Exception as e:
        raise Exception(e)

"""
A function to calculate the speed from distance and transaction timestamp differentials
:param dist: (Float) Distance between postcodes
:param transaction_dt1: Transaction timestamp from the table
:param transaction_dt2: Transaction timestamp from the POS
:return: (Float) Speed
"""
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[hadoop@ip-172-31-28-134 ~]$
```

8. Further, updating the '**driver.py**' file with the following code

Setting up the system dependencies and importing necessary libraries and modules

hadoop@ip-172-31-28-134:~

```
#importing necessary libraries
import sys
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
import happybase
import math
import pandas as pd
from datetime import datetime
import uuid
```

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

   - **Connect to kafka topic using**

   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()
```

10. Defining the JSON schema of each transaction

```
# 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())
```

11. Reading the raw JSON data from Kafka as 'credit_data_stream' and defining UDF's to verify rules

```
# 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")).select(
    "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']))
```

12. Code to display output in console

```
# Write output to console as well
output_data = Final_data \
    .select("card_id", "member_id", "amount", "pos_id", "postcode", "transaction_dt") \
    .writeStream \
    .trigger(processingTime = '1 seconds') \
    .outputMode("append") \
    .format("console") \
    .option("truncate", "false") \
    .start()
```

13. Define spark termination

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

14. Setting the Kafka Version using the following command

export SPARK_KAFKA_VERSION=0.10

15. Running the spark-submit command, specifying the Spark-SQL-Kafka package and python file

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

```
hadoop@ip-172-31-28-134:~
[hadoop@ip-172-31-28-134 ~]$ export SPARK_KAFKA_VERSION=0.10
[hadoop@ip-172-31-28-134 ~]$ spark-submit --packages org.apache.spark:
spark-sql-kafka-0-10_2.11:2.4.5 driver.py
```

```
hadoop@ip-172-31-28-134:~
        [SUCCESSFUL ] org.lz4#lz4-java;1.4.0!lz4-java.jar (12ms)
downloading https://repo1.maven.org/maven2/org/xerial/snappy/snappy-java/1.1.7.3/snappy-java-1.1.7.3.jar ...
        [SUCCESSFUL ] org.xerial.snappy#snappy-java;1.1.7.3!snappy-java.jar(bundle) (67ms)
downloading https://repo1.maven.org/maven2/org/slf4j/slf4j-api/1.7.16/slf4j-api-1.7.16.jar ...
        [SUCCESSFUL ] org.slf4j#slf4j-api;1.7.16!slf4j-api.jar (5ms)
:: resolution report :: resolve 1235ms :: artifacts dl 186ms
        :: 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   |   6   |   6   |   0   ||   6   |   6   |
        ---------------------------------------------------------------------
```

16. Checking the output in console:

```
atch: 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|525701337355194|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|564240259678903|50635   |2017-12-31 19:37:19|GENUINE|
|5407073344486464|114792208434 4|6885448|887913906711117|59031   |2017-12-31 07:53:53|FRAUD  |
|5407073344486464|114792208434 4|4028209|116266051118182|80118   |2017-12-31 01:06:50|FRAUD  |
|5407073344486464|114792208434 4|3858369|896105817613325|53820   |2017-12-31 17:37:26|GENUINE|
|5407073344486464|114792208434 4|9307733|729374116016479|14898   |2017-12-31 04:50:16|FRAUD  |
|5407073344486464|114792208434 4|4011296|543373367319647|44028   |2017-12-31 13:09:34|GENUINE|
|5407073344486464|114792208434 4|9492531|211980095659371|49453   |2017-12-31 14:12:26|GENUINE|
|5407073344486464|114792208434 4|7550074|345533088112099|15030   |2017-12-31 02:34:52|FRAUD  |
+----------------+-------------+-------+----------------+--------+-------------------+-------+
nly showing top 20 rows
```

17. Count Data in Hbase: **count 'lookup_data_hive'**

```
urrent count: 20000, row: 27888
urrent count: 21000, row: 28859
urrent count: 22000, row: 29759
urrent count: 23000, row: 30498
urrent count: 24000, row: 31598
urrent count: 25000, row: 32498
urrent count: 26000, row: 33388
urrent count: 27000, row: 341724964658347.210778177555185.12-04-2018152430.2021-01-04171328.368477
urrent count: 28000, row: 3666156524514 37.540752175694215.28-04-2018005259.2021-01-04171400.227028
urrent count: 29000, row: 35264
urrent count: 30000, row: 36168
urrent count: 31000, row: 3705820335666789.433446469625434.06-07-2018094337.2021-01-04171349.499439
urrent count: 32000, row: 3756068755521405.880937166605449.26-05-2018130045.2021-01-04171430.733012
urrent count: 33000, row: 35174
urrent count: 34000, row: 35874
urrent count: 35000, row: 35977
urrent count: 36000, row: 40748
urrent count: 37000, row: 41560
urrent count: 38000, row: 42387
urrent count: 39000, row: 431554145046434035.4944127427327147.12-02-2018145807.2021-01-04171356.009416
urrent count: 40000, row: 43999
urrent count: 41000, row: 44784
urrent count: 42000, row: 45346
urrent count: 43000, row: 46358
urrent count: 44000, row: 47134
urrent count: 45000, row: 47925
urrent count: 46000, row: 48730
urrent count: 47000, row: 49500
urrent count: 48000, row: 50351
urrent count: 49000, row: 5120
urrent count: 50000, row: 51888
urrent count: 51000, row: 5287502990314019.2051724643868018.14-07-2018075014.2021-01-04171327.947742
urrent count: 52000, row: 53290
urrent count: 53000, row: 5420
urrent count: 54000, row: 4211
urrent count: 55000, row: 4475865841720966.2732468410777378.06-10-2018212051.2021-01-04171333.585977
urrent count: 56000, row: 4945
urrent count: 57000, row: 7865
urrent count: 58000, row: 8765
urrent count: 59000, row: 9645
9367 row(s) in 3.8140 seconds
```

*Total number of records is **59367** which is matching with given requirement.*