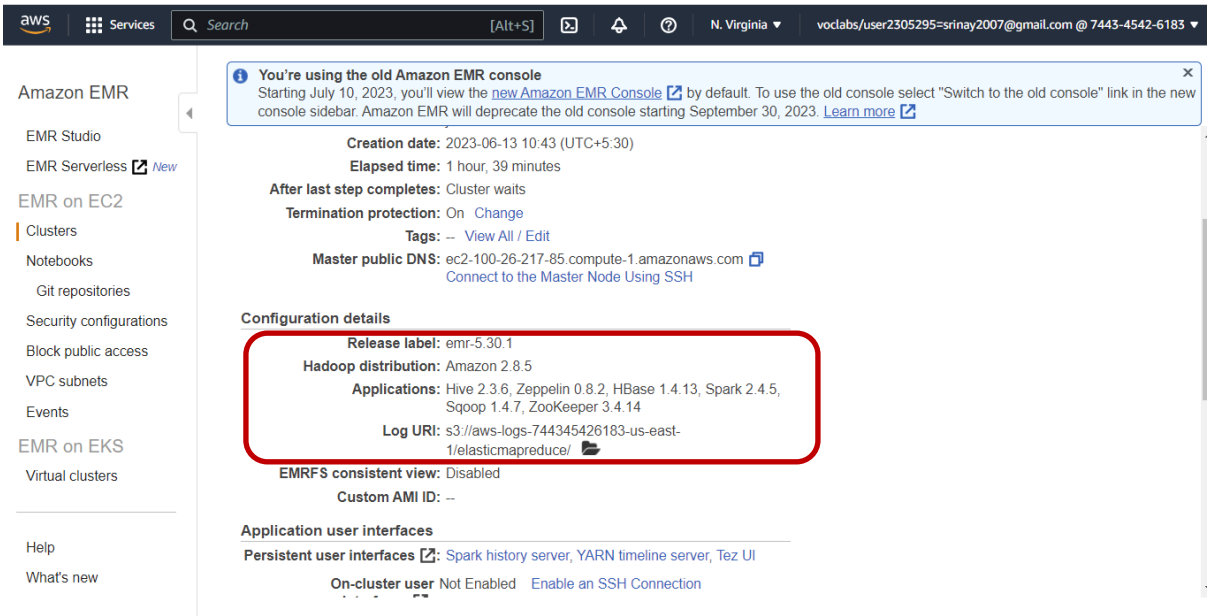
Logic of Credit Card Fraud Detection Project – Final Submission

As part of **Credit Card Fraud Detection Project** final submission below tasks are performed.

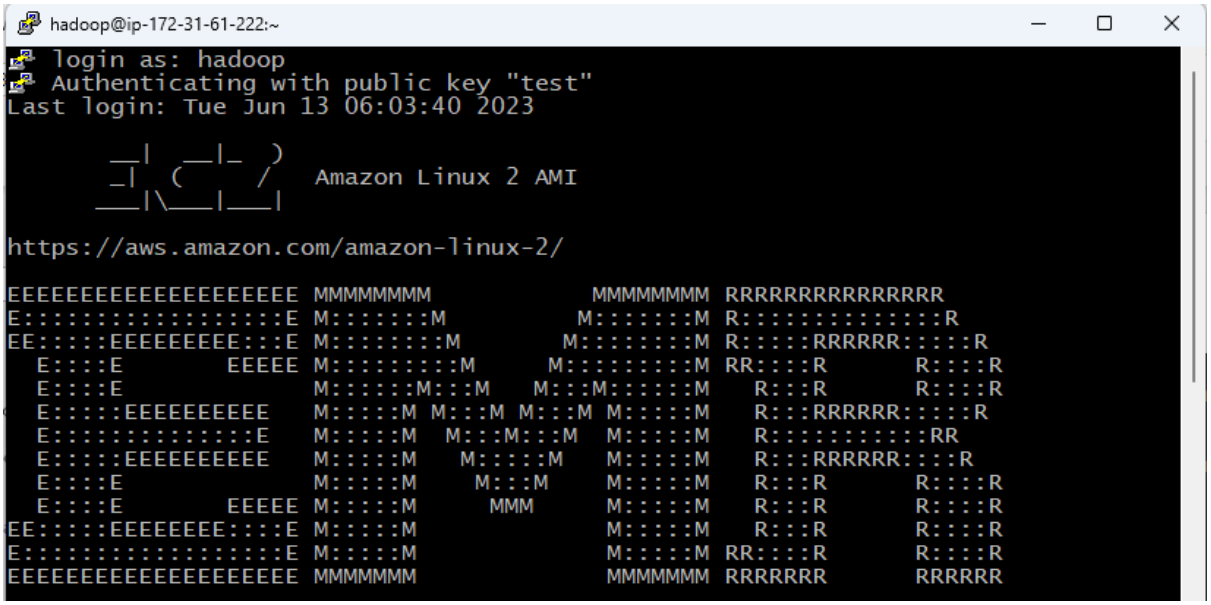
• **Task 5**: Create a streaming data processing framework that ingests real-time POS transaction data from Kafka. The transaction data is then validated based on the three rules' parameters (stored in the NoSQL database) discussed previously.
• **Task 6**: Update the transactions data along with the 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.

To complete above task below steps are performed.

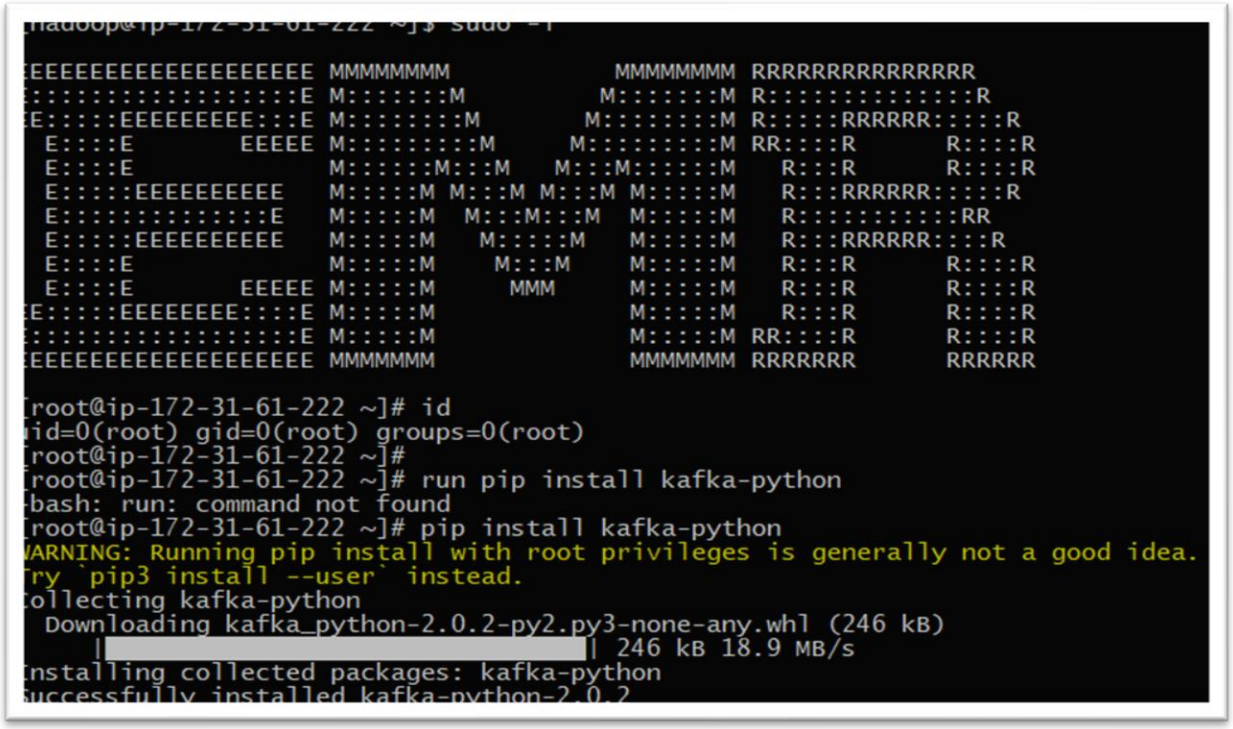
1. EMR Cluster is set up with **Hadoop, Sqoop, Hive, HBase and Spark,** Root device EBS volume size **as** 20 GB



2. Logged into EMR instance as “hadoop”:



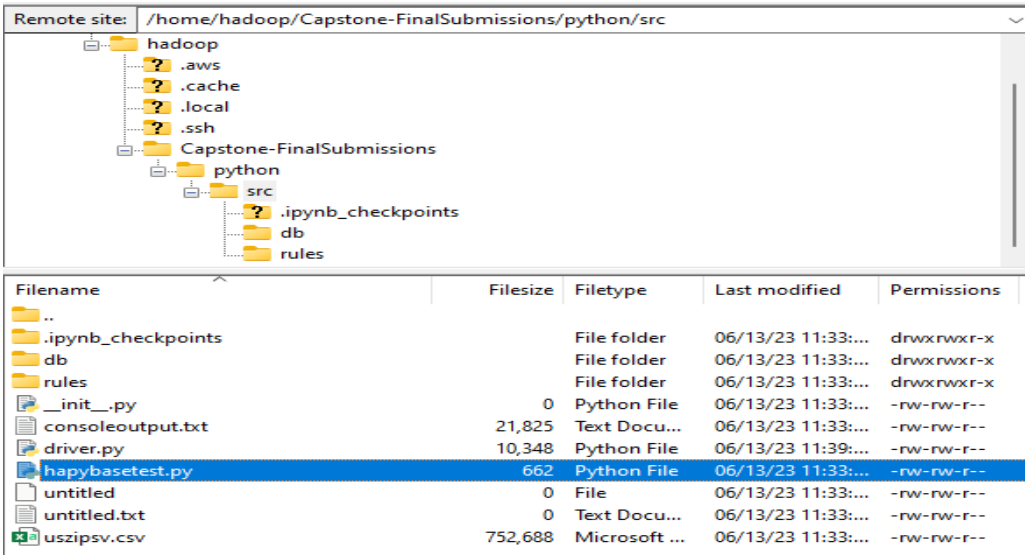
3. Switch to root user and run pip install kafka-python and then again use “sudo -i –u hadoop” to be a hadoop user



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

- **sudo yum update**
- **sudo yum install python3-devel**
- **sudo pip install pandas**
- **sudo pip install happybase**

5. Downloaded db-> dao.py , geomap.py ,rules-> rules.py ,driver.py ,unzipsv.csv from the resource section of the capstone project from the learning platform and transfer it to ec2 instance via **filezilla**.



6. Updated the self.host = localhost in dao.py file

```
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 = 'localhost'
            for i in range(2):
                try:
                    self.pool = happybase.ConnectionPool(size=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:
                    t = connection.table(table)
                    row = t.row(key)
                    return row
```

7. Updated rules.py with following parameters:

lookup_table = 'lookup_data_hive'
master_table = 'card_transactions_hive'

```
# Create UDF functions
lookup_table = 'lookup_data_hive'
master_table = 'card_transactions_hive'
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 rule Transaction amount should be less than Upper control limit (UCL)

```
"""
Function to verify the UCL rule
Transaction amount should be less than Upper control limit (UCL)
:param card_id: (Long) Card id of the card customer
:param amount: (Double) The transaction amount
:return: (Boolean)
"""
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 zipcode rules.ZIP code distance

```python
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

```python
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
```

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, updated the 'driver.py' file with the following code

Setting up system dependencies and importing necessary libraries and modules

```
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
```

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

```
#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

```
# 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

```
# 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']))
```

13.Set the Kafka Version using the following command

export SPARK_KAFKA_VERSION=0.10

14.Run 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 > consoleoutput.txt

```
[hadoop@ip-172-31-81-156 ~]$ export SPARK_KAFKA_VERSION=0.10
[hadoop@ip-172-31-81-156 ~]$ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 driver.py > consoleo
utput.txt
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-7d90c783-3b6b-4173-8eb0-1525f9174b0b;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 383ms :: artifacts dl 11ms
        :: 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-7d90c783-3b6b-4173-8eb0-1525f9174b0b
        confs: [default]
        0 artifacts copied, 6 already retrieved (0kB/10ms)
23/06/18 06:42:56 INFO SparkContext: Running Spark version 2.4.5-amzn-0
23/06/18 06:42:56 INFO SparkContext: Submitted application: CapStone_Project
23/06/18 06:42:56 INFO SecurityManager: Changing view acls to: hadoop
23/06/18 06:42:56 INFO SecurityManager: Changing modify acls to: hadoop
23/06/18 06:42:56 INFO SecurityManager: Changing view acls groups to:
23/06/18 06:42:56 INFO SecurityManager: Changing modify acls groups to:
23/06/18 06:42:56 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permi
ssions: Set(hadoop); groups with view permissions: Set(); users  with modify permissions: Set(hadoop); groups with modify
permissions: Set()
23/06/18 06:42:56 INFO Utils: Successfully started service 'sparkDriver' on port 39679.
23/06/18 06:42:57 INFO SparkEnv: Registering MapOutputTracker
23/06/18 06:42:57 INFO SparkEnv: Registering BlockManagerMaster
23/06/18 06:42:57 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topolo
gy information
23/06/18 06:42:57 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
23/06/18 06:42:57 INFO DiskBlockManager: Created local directory at /mnt/tmp/blockmgr-151cdfc0-0b35-4435-9034-9c91fd53d122
23/06/18 06:42:57 INFO MemoryStore: MemoryStore started with capacity 1008.9 MB
23/06/18 06:42:57 INFO SparkEnv: Registering OutputCommitCoordinator
23/06/18 06:42:57 INFO Utils: Successfully started service 'SparkUI' on port 4040.
23/06/18 06:42:57 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://ip-172-31-81-156.ec2.internal:4040
23/06/18 06:42:57 INFO Utils: Using initial executors = 50, max of spark.dynamicAllocation.initialExecutors, spark.dynamic
Allocation.minExecutors and spark.executor.instances
23/06/18 06:42:58 INFO RMProxy: Connecting to ResourceManager at ip-172-31-81-156.ec2.internal/172.31.81.156:8032
23/06/18 06:42:58 INFO Client: Requesting a new application from cluster with 1 NodeManagers
23/06/18 06:42:58 INFO Client: Verifying our application has not requested more than the maximum memory capability of the
cluster (12288 MB per container)
23/06/18 06:42:58 INFO Client: Will allocate AM container, with 896 MB memory including 384 MB overhead
23/06/18 06:42:58 INFO Client: Setting up container launch context for our AM
23/06/18 06:42:58 INFO Client: Setting up the launch environment for our AM container
```

15.Check Output in console :

```
[hadoop@ip-172-31-81-156 ~]$ cat consoleoutput.txt
+----------------+-------------+---------+---------------+--------+-------------------+--------+
|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
```

16.Count Data in Hbase table card_transactions_hive

count 'card_transactions_hive'

```
hbase(main):003:0> count 'card_transactions_hive1'
Current count: 1000, row: 341519629171378-10-03-2017 15:24:29-4153279
Current count: 2000, row: 342917304031089-14-05-2017 00:20:04-9918738
Current count: 3000, row: 344143554194093.605980287520001.04-03-2018_05_11_13.2020-11-08_19_44_56.366442
Current count: 4000, row: 345082216803859-20-11-2017 17:10:46-5256223
Current count: 5000, row: 346119019394205.443551645061859.26-11-2018_20_33_06.2020-11-08_19_44_48.255094
Current count: 6000, row: 347566265344782-15-09-2017 03:35:05-2934182
Current count: 7000, row: 348546826126582-04-03-2016 07:59:28-4376850
Current count: 8000, row: 349612084165184-09-03-2016 14:23:39-8910184
Current count: 9000, row: 371085417586954-29-05-2016 06:49:09-8027336
Current count: 10000, row: 372885987499499-09-03-2017 02:32:26-4787732
Current count: 11000, row: 373866218194240-17-12-2016 02:30:53-4944880
Current count: 12000, row: 375771346371172-05-07-2017 11:05:55-2387800
Current count: 13000, row: 376529531867128.858544169966966.24-02-2018_02_03_50.2020-11-08_19_45_16.515291
Current count: 14000, row: 377613757081826-30-01-2018 09:05:15-9712373
Current count: 15000, row: 378827123583890-18-11-2017 17:46:42-3336581
Current count: 16000, row: 402141715453983O-14-12-2016 02:32:07-903392
Current count: 17000, row: 408689124165530O-24-11-2017 14:13:34-411151
Current count: 18000, row: 415366857425661 7.489643949865990.19-10-2018_02_32_22.2020-11-08_19_44_44.763813
Current count: 19000, row: 426455357918658 7-10-07-2017 01:42:05-2647153
Current count: 20000, row: 436712547862653 6-17-09-2017 13:31:11-4402322
Current count: 21000, row: 441129694689813 3-05-07-2017 02:19:14-6703658
Current count: 22000, row: 449284809815893 1-03-10-2017 06:58:23-7748034
Current count: 23000, row: 453974417702905 0-03-02-2016 18:43:46-5722786
Current count: 24000, row: 458558700168456 1.447917467655871.07-12-2018_03_15_33.2020-11-08_19_44_48.736927
Current count: 25000, row: 462537189977216 6-15-08-2016 06:17:13-9151039
Current count: 26000, row: 471941857493681 7.500004843659347.27-07-2018_00_46_17.2020-11-08_19_44_45.787387
Current count: 27000, row: 478287946462144 8-06-08-2017 15:33:42-2935471
Current count: 28000, row: 486555600618798 0.454102358826314.30-07-2018_12_30_09.2020-11-08_19_44_41.274872
Current count: 29000, row: 490725300063053-08-11-2016 16:49:32-360371
Current count: 30000, row: 490154842413702 7-12-01-2017 12:17:18-1101566
Current count: 31000, row: 512731899946655 9.391603008295087.02-09-2018_12_07_56.2020-11-08_19_44_35.859457
Current count: 32000, row: 515530601431575 1-16-04-2016 22:19:40-881860
Current count: 33000, row: 517573581960702 8-11-02-2018 00:00:00-7263153
Current count: 34000, row: 520412327372970 0-27-05-2016 16:11:45-1852352
Current count: 35000, row: 523145603633330 4-06-11-2017 12:35:36-7465352
Current count: 36000, row: 526377449100205 2-22-02-2016 03:23:05-2401368
Current count: 37000, row: 530086808977175 7-11-02-2018 00:00:09-580
Current count: 38000, row: 534342202924794 8-28-09-2016 17:41:27-9546692
Current count: 39000, row: 538097818417560 0-11-09-2016 13:07:31-3663489
Current count: 40000, row: 541185684273591 9.558289402750917.25-10-2018_08_48_24.2020-11-08_19_44_51.753408
Current count: 41000, row: 546245758939871 0-18-09-2017 10:35:12-6468520
Current count: 42000, row: 551705793715843 9.625723534759441.11-10-2018_03_54_06.2020-11-08_19_45_14.042005
Current count: 43000, row: 557242753031123 6-03-06-2017 11:54:08-211936
Current count: 44000, row: 559527327750757 3-26-12-2017 04:38:41-7702865
Current count: 45000, row: 601128943286496 8-09-02-2016 23:57:19-7385513
Current count: 46000, row: 601159037486901 8-08-07-2017 00:32:41-8045248
Current count: 47000, row: 601179868661487 6-07-10-2017 12:47:53-7750412
Current count: 48000, row: 622188108999587 2-18-01-2018 12:40:50-5665918
Current count: 49000, row: 622393873993491 6-05-04-2017 23:25:25-1810513
Current count: 50000, row: 622547268181975 8.718958735965781.26-02-2018_07_08_36.2020-11-08_19_45_06.645371
Current count: 51000, row: 622792529947923 0-12-09-2016 04:55:32-154454
Current count: 52000, row: 644378302306959 7.523325937872991.12-02-2018_23_14_53.2020-11-08_19_44_47.135768
Current count: 53000, row: 645758718464447 7-08-12-2017 15:18:04-9181829
Current count: 54000, row: 647369654591832 2-29-05-2017 23:03:18-1745203
Current count: 55000, row: 648987845498864 4-25-11-2017 06:01:35-7454969
Current count: 56000, row: 651556725832491 5-07-06-2016 17:17:28-7334493
Current count: 57000, row: 654595232237998 4.600766853828700.11-07-2018_20_41_28.2020-11-08_19_44_55.679562
Current count: 58000, row: 657465606867565 1-05-06-2016 02:05:23-118517
Current count: 59000, row: 659563865873675 1-15-10-2017 12:09:29-8227092
59376 row(s) in 4.3340 seconds
```

Total **59376 records** are found in **credit_transactions_hive table** after spark submit command.