Final-Submission – Logic Explanation

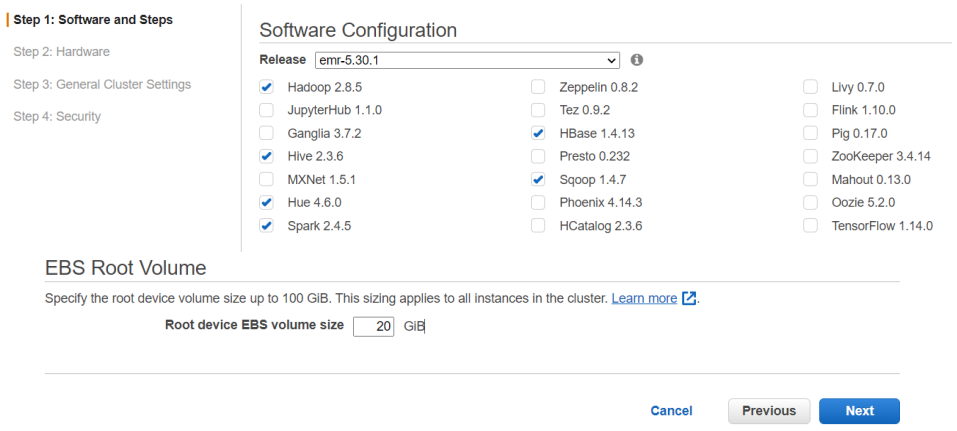
- **Explanation of the solution to the streaming layer problem**

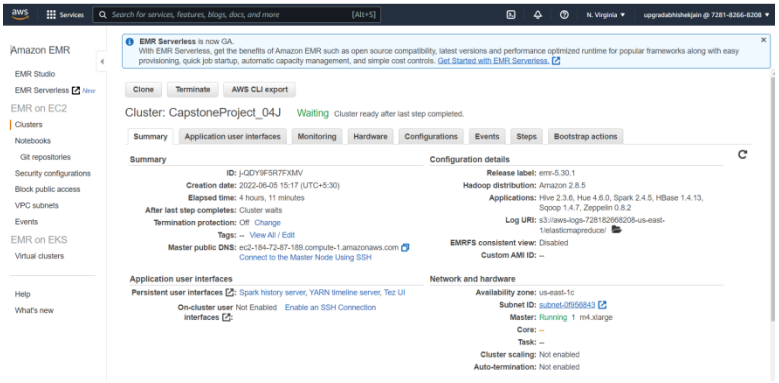
*****Please zoom to 180% or 200% to see screenshots with better clarity**

1. In order to complete below tasks, I have created EMR cluster with **Hadoop, Sqoop, Hive, HBase and Spark,** Root device EBS volume size **as** 20 GB

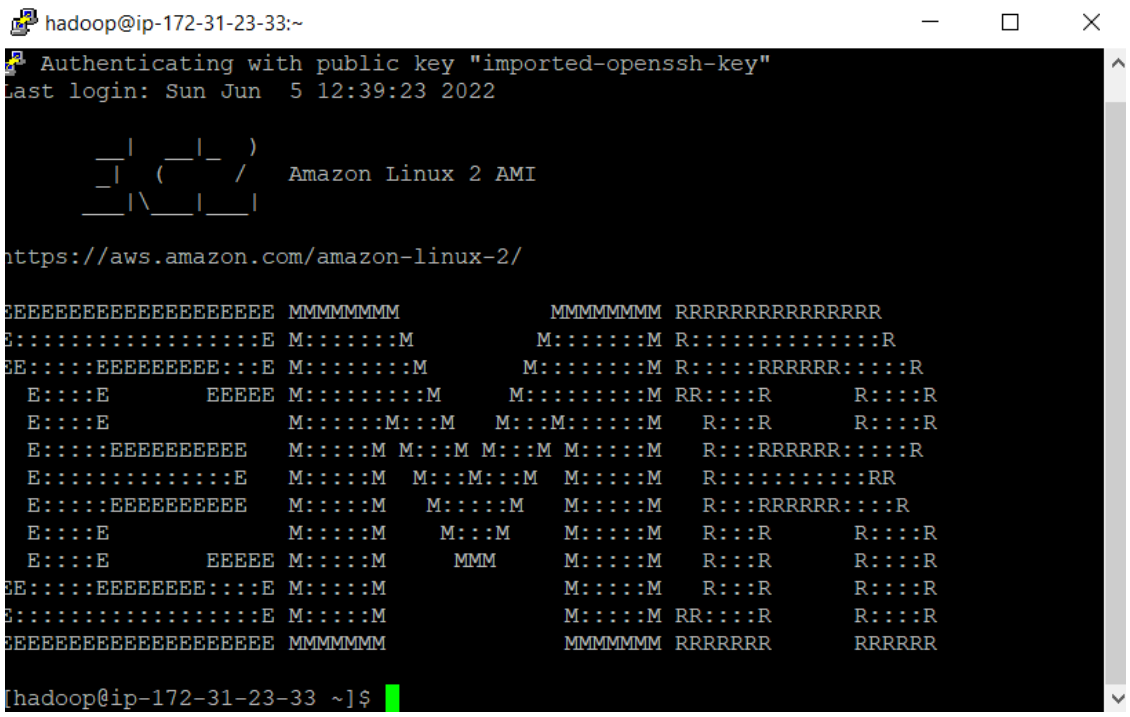
- **Task 5**: Create a streaming data processing framework that ingests real-time POS transaction data from Kafka. The transaction data is then validated based on the three rules' parameters (stored in the NoSQL database) discussed previously.
- **Task 6**: Update the transactions data along with the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

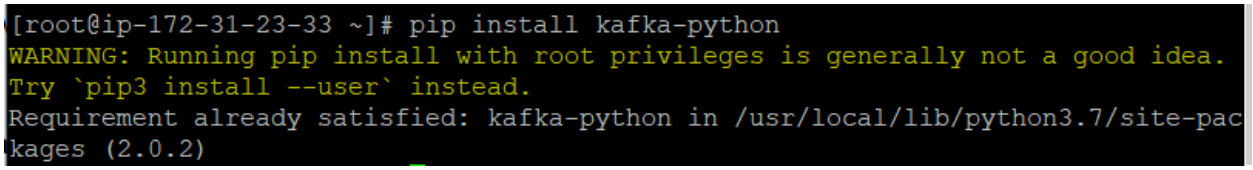




2. **Logged into EMR instance as “hadoop”:**

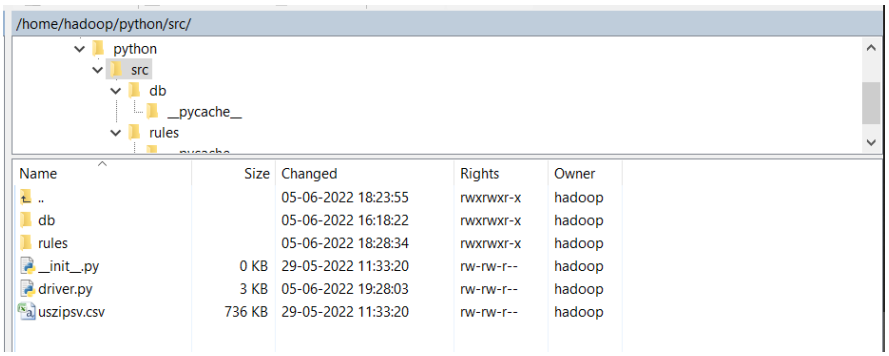


3. Switch to root user and run pip install kafka-python and then again use “sudo -i –u hadoop” to be a hadoop user

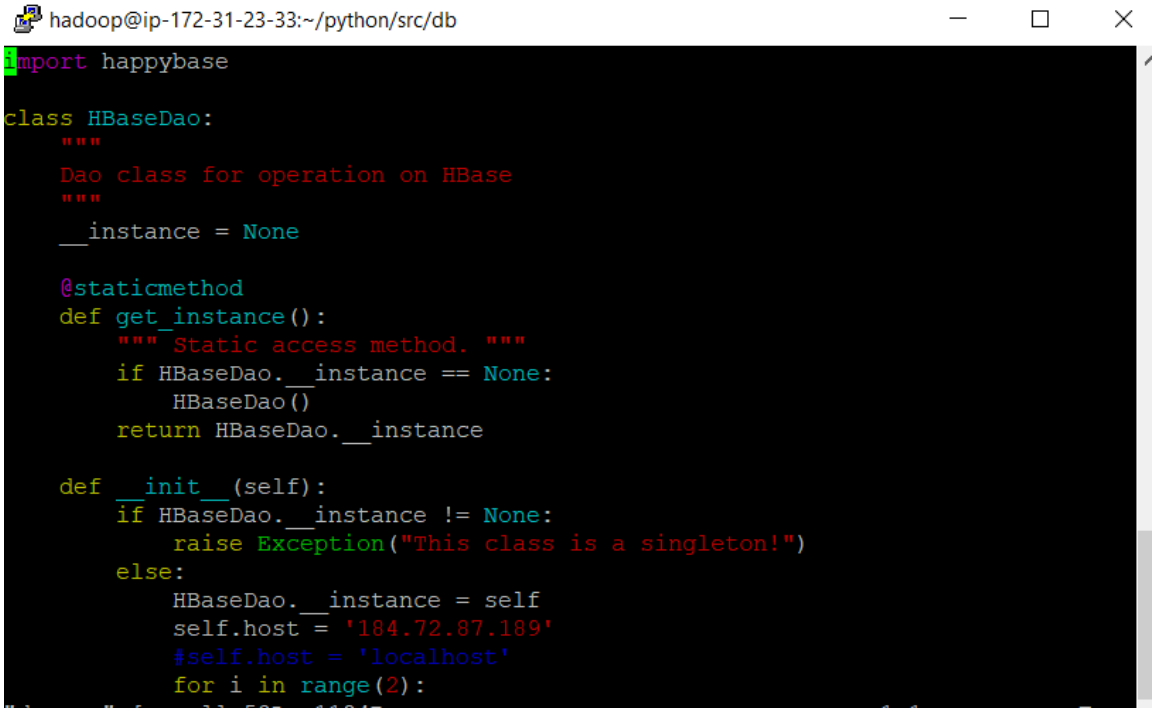


4. Run the following commands in order to Install Happy base and start thrift server
    - **sudo yum update**
    - **sudo yum install python3-devel**
    - **pip install happybase**
    - **/usr/lib/hbase/bin/hbase-daemon.sh start thrift -p 9090**

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 hadoop instance via WinSCP.



6. Updated the Public IP of your EC2 Instance "**184.72.87.189**"(self.host) in **dao.py** file



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'
```

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)

```
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

```
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, I updated the 'driver.py' file with the following code
Setting up the system dependencies and importing necessary libraries and modules

```
#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
1. 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()
```

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

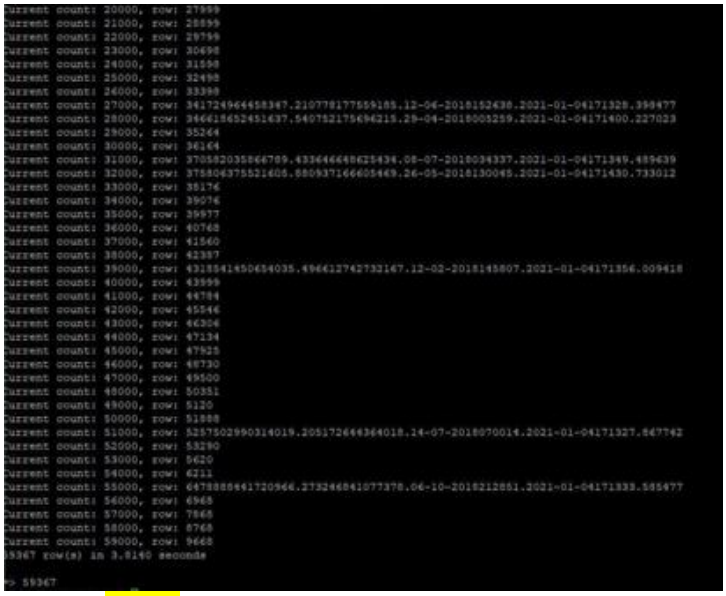
```
hadoop@ip-172-31-23-33:~/python/src
[hadoop@ip-172-31-23-33 src]$ export SPARK_KAFKA_VERSION=0.10
[hadoop@ip-172-31-23-33 src]$ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 driver.py
```

```
hadoop@ip-172-31-23-33:~/python/src
[hadoop@ip-172-31-23-33 src]$ export SPARK_KAFKA_VERSION=0.10
[hadoop@ip-172-31-23-33 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-0cc3580a-528f-4d96-a1ed-f668602ac5ce;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 528ms :: artifacts dl 21ms
        :: 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 in 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-0cc3580a-528f-4d96-a1ed-f668602ac5ce
        confs: [default]
        0 artifacts copied, 6 already retrieved (0kB/12ms)
22/06/05 12:59:23 INFO SparkContext: Running Spark version 2.4.5-amzn-0
22/06/05 12:59:23 INFO SparkContext: Submitted application: CreditFraud
22/06/05 12:59:23 INFO SecurityManager: Changing view acls to: hadoop
22/06/05 12:59:23 INFO SecurityManager: Changing modify acls to: hadoop
22/06/05 12:59:23 INFO SecurityManager: Changing view acls groups to:
22/06/05 12:59:23 INFO SecurityManager: Changing modify acls groups to:
22/06/05 12:59:23 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions:
sers  with modify permissions: Set(hadoop); groups with modify permissions: Set()
22/06/05 12:59:24 INFO Utils: Successfully started service 'sparkDriver' on port 33835.
22/06/05 12:59:24 INFO SparkEnv: Registering MapOutputTracker
22/06/05 12:59:24 INFO SparkEnv: Registering BlockManagerMaster
22/06/05 12:59:24 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology infc
22/06/05 12:59:24 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
22/06/05 12:59:24 INFO DiskBlockManager: Created local directory at /mnt/tmp/blockmgr-bd558144-04a9-4075-a87f-96fa8c568453
22/06/05 12:59:24 INFO MemoryStore: MemoryStore started with capacity 1028.8 MB
22/06/05 12:59:24 INFO SparkEnv: Registering OutputCommitCoordinator
22/06/05 12:59:25 INFO Utils: Successfully started service 'SparkUI' on port 4040.
22/06/05 12:59:25 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://ip-172-31-23-33.ec2.internal:4040
22/06/05 12:59:25 INFO Utils: Using initial executors = 50, max of spark.dynamicAllocation.initialExecutors, spark.dynamicAllocat
```

17. Check 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|7865627771140812|84758  |2017-12-31 04:15:03|FRAUD  |
|348702330256514 |37495066290  |7454328|4669525713935O8|93645   |2017-12-31 09:56:42|GENUINE|
|348702330256514 |37495066290  |4013428|45845320330319 |15868   |2017-12-31 05:38:54|GENUINE|
|348702330256514 |37495066290  |5495353|5454996219656 97|79033  |2017-12-31 21:51:54|GENUINE|
|348702330256514 |37495066290  |3966214|3692663422725O1|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|1820313834431 15|26346  |2017-12-31 01:52:32|FRAUD  |
|5189563368503974|117826301530 |8938921|7997482464110 19|76934  |2017-12-31 05:20:53|FRAUD  |
|5189563368503974|117826301530 |1786366|131276818071265|63431   |2017-12-31 14:29:38|GENUINE|
|5189563368503974|117826301530 |9142237|5642402596789O3|50635   |2017-12-31 19:37:19|GENUINE|
|5407073344486464|1147922084344|6885448|887913906711117|59031   |2017-12-31 07:53:53|FRAUD  |
|5407073344486464|1147922084344|4028209|1162660511181 82|80118  |2017-12-31 01:06:50|FRAUD  |
|5407073344486464|1147922084344|3858369|896105817613325|53820   |2017-12-31 17:37:26|GENUINE|
|5407073344486464|1147922084344|9307733|7293741160164 79|14898  |2017-12-31 04:50:16|FRAUD  |
|5407073344486464|1147922084344|4011296|5433733673196 47|44028  |2017-12-31 13:09:34|GENUINE|
|5407073344486464|1147922084344|9492531|211980095659371|49453   |2017-12-31 14:12:26|GENUINE|
|5407073344486464|1147922084344|7550074|3455330881120 99|15030  |2017-12-31 02:34:52|FRAUD  |
+----------------+-------------+-------+---------------+--------+-------------------+-------+
nly showing top 20 rows
```

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



Total number for record is **59367** which is matching with given requirement