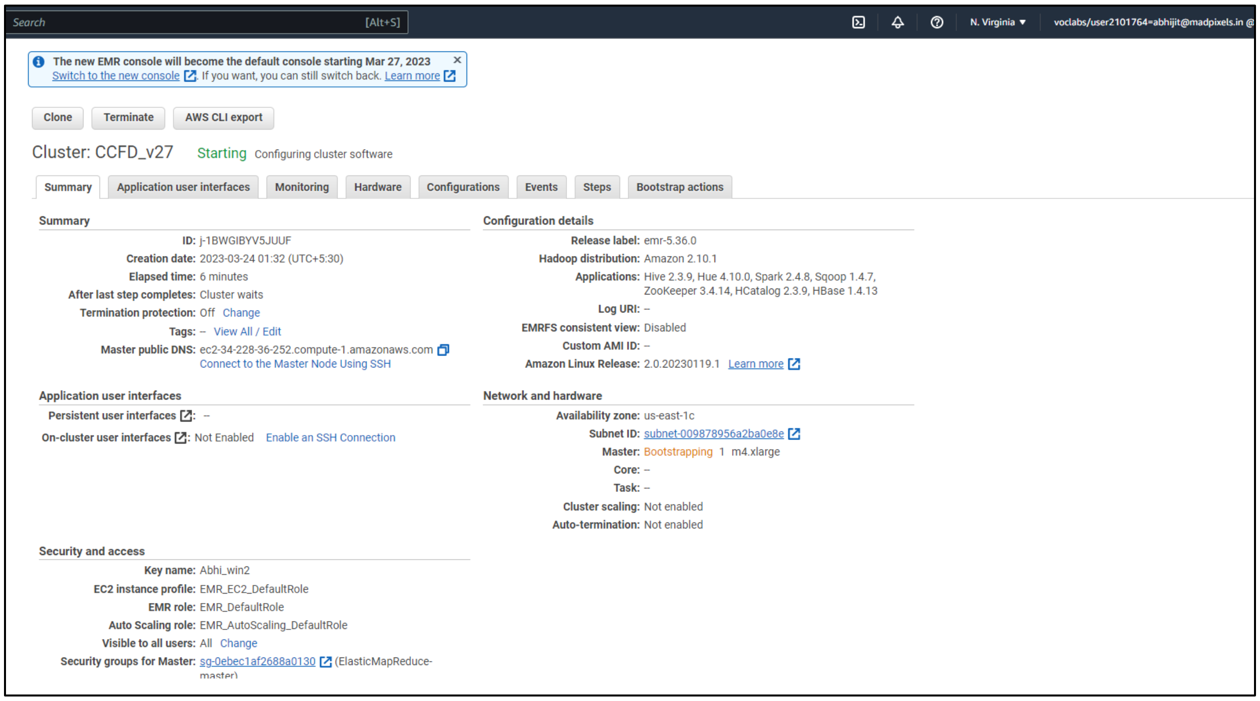
Scripts Execution

Explanation of the solution to the streaming layer problem

1. Created cluster.



Search [Alt+S] N. Virginia voclabs/user2101764=abhijit@madpixels.in @

The new EMR console will become the default console starting Mar 27, 2023
Switch to the new console. If you want, you can still switch back. Learn more

Cluster: CCFD_v27 Starting Configuring cluster software

Summary

ID:	j-1BWGIBYV5JUUF
Creation date:	2023-03-24 01:32 (UTC+5:30)
Elapsed time:	6 minutes
After last step completes:	Cluster waits
Termination protection:	Off Change
Tags:	– View All / Edit
Master public DNS:	ec2-34-228-36-252.compute-1.amazonaws.com Connect to the Master Node Using SSH

Configuration details

Release label:	emr-5.36.0
Hadoop distribution:	Amazon 2.10.1
Applications:	Hive 2.3.9, Hue 4.10.0, Spark 2.4.8, Sqoop 1.4.7, ZooKeeper 3.4.14, HCatalog 2.3.9, HBase 1.4.13
Log URI:	–
EMRFS consistent view:	Disabled
Custom AMI ID:	–
Amazon Linux Release:	2.0.20230119.1 Learn more

Application user interfaces

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

Network and hardware

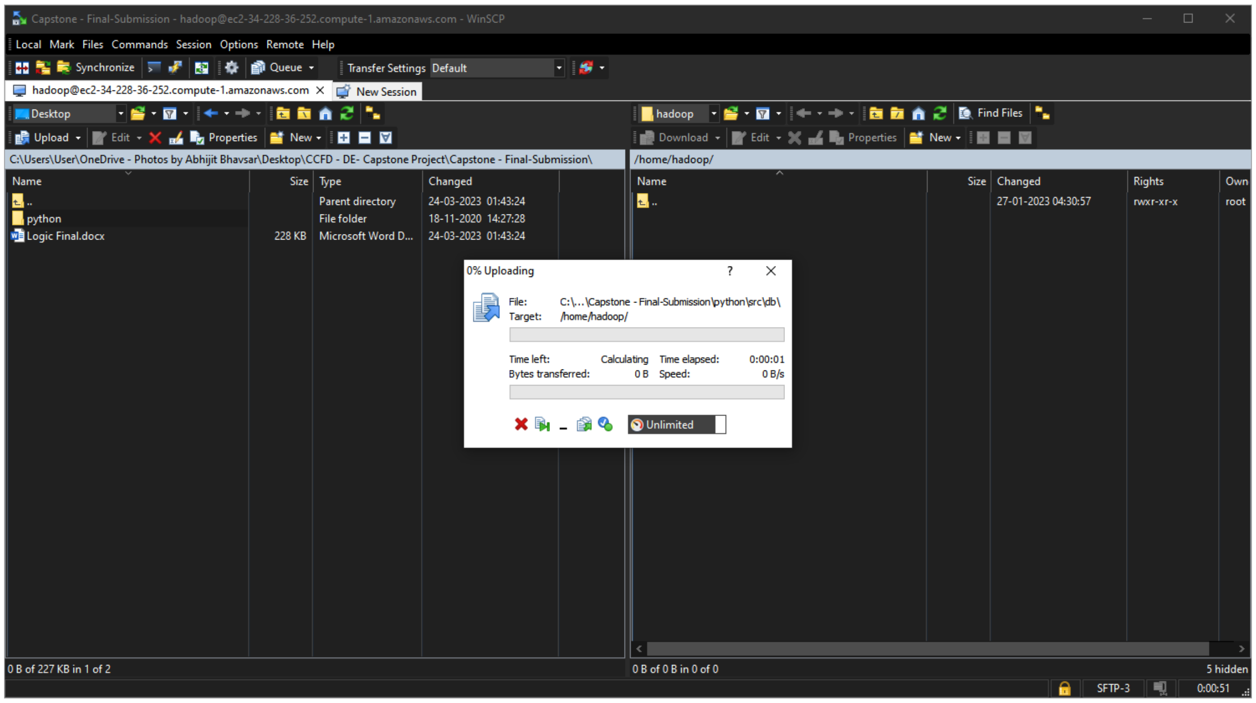
Availability zone:	us-east-1c
Subnet ID:	subnet-009878956a2ba0e8e
Master:	Bootstrapping 1 m4.xlarge
Core:	–
Task:	–
Cluster scaling:	Not enabled
Auto-termination:	Not enabled

Security and access

Key name:	Abhi_win2
EC2 instance profile:	EMR_EC2_DefaultRole
EMR role:	EMR_DefaultRole
Auto Scaling role:	EMR_AutoScaling_DefaultRole
Visible to all users:	All Change
Security groups for Master:	sg-0ebec1af2688a0130 (ElasticMapReduce-master)

Console Screenshot 1: Created EMR with Hadoop, Hive, HBase, HCatalog, Spark, ZooKeeper, Hue and Sqoop.

2. Transferring Sample data to the cluster using WinSCP.


Capstone - Final-Submission - hadoop@ec2-34-228-36-252.compute-1.amazonaws.com - WinSCP

Local Mark Files Commands Session Options Remote Help

Transfer Settings Default

hadoop@ec2-34-228-36-252.compute-1.amazonaws.com | New Session

C:\Users\User\OneDrive - Photos by Abhijit Bhavsar\Desktop\CCFD - DE- Capstone Project\Capstone - Final-Submission\

Name	Size	Type	Changed
..		Parent directory	24-03-2023 01:43:24
python		File folder	18-11-2020 14:27:28
Logic Final.docx	228 KB	Microsoft Word D...	24-03-2023 01:43:24

/home/hadoop/

Name	Size	Changed	Rights	Own
..		27-01-2023 04:30:57	rwxr-xr-x	root

0% Uploading

File: C:\...\Capstone - Final-Submission\python\src\db\
Target: /home/hadoop/

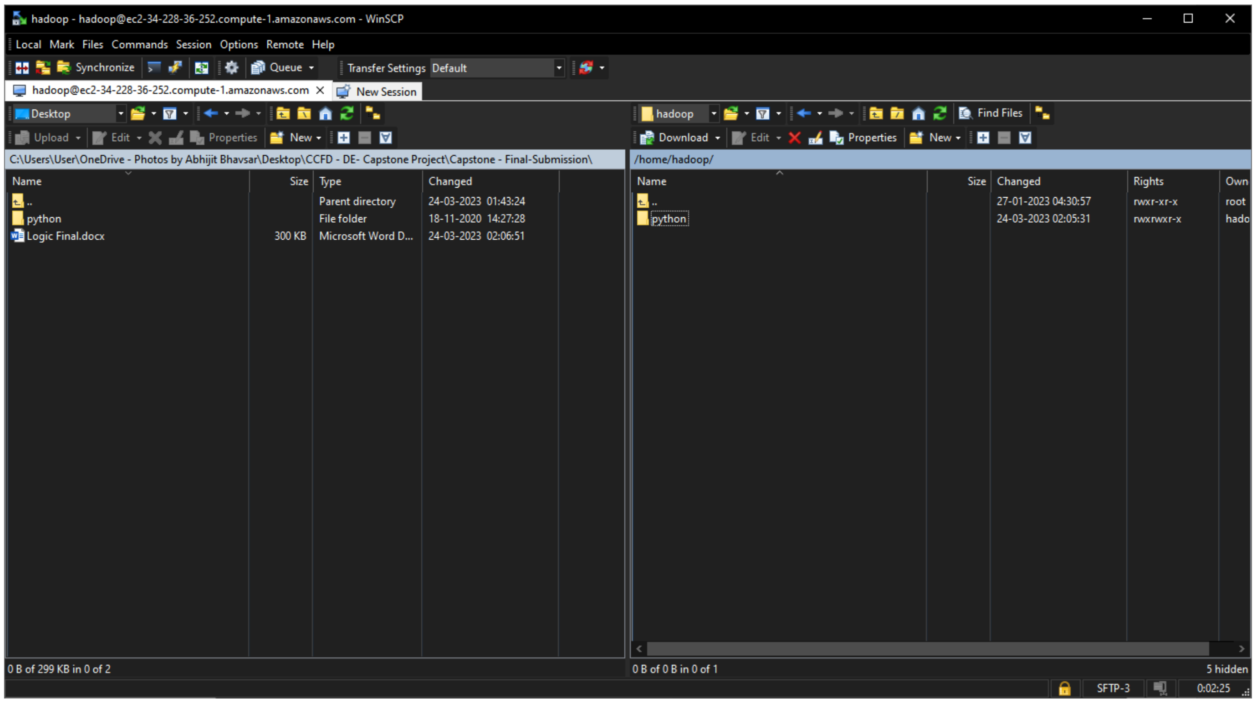
Time left: Calculating Time elapsed: 0:00:01
Bytes transferred: 0 B Speed: 0 B/s

Unlimited

0 B of 227 KB in 1 of 2 | 0 B of 0 B in 0 of 0 | 5 hidden

SFTP-3 | 0:00:51

Console Screenshot 2: Copying data from local machine.


hadoop - hadoop@ec2-34-228-36-252.compute-1.amazonaws.com - WinSCP

Local Mark Files Commands Session Options Remote Help

Transfer Settings Default

hadoop@ec2-34-228-36-252.compute-1.amazonaws.com | New Session

C:\Users\User\OneDrive - Photos by Abhijit Bhavsar\Desktop\CCFD - DE- Capstone Project\Capstone - Final-Submission\

Name	Size	Type	Changed
..		Parent directory	24-03-2023 01:43:24
python		File folder	18-11-2020 14:27:28
Logic Final.docx	300 KB	Microsoft Word D...	24-03-2023 02:06:51

/home/hadoop/

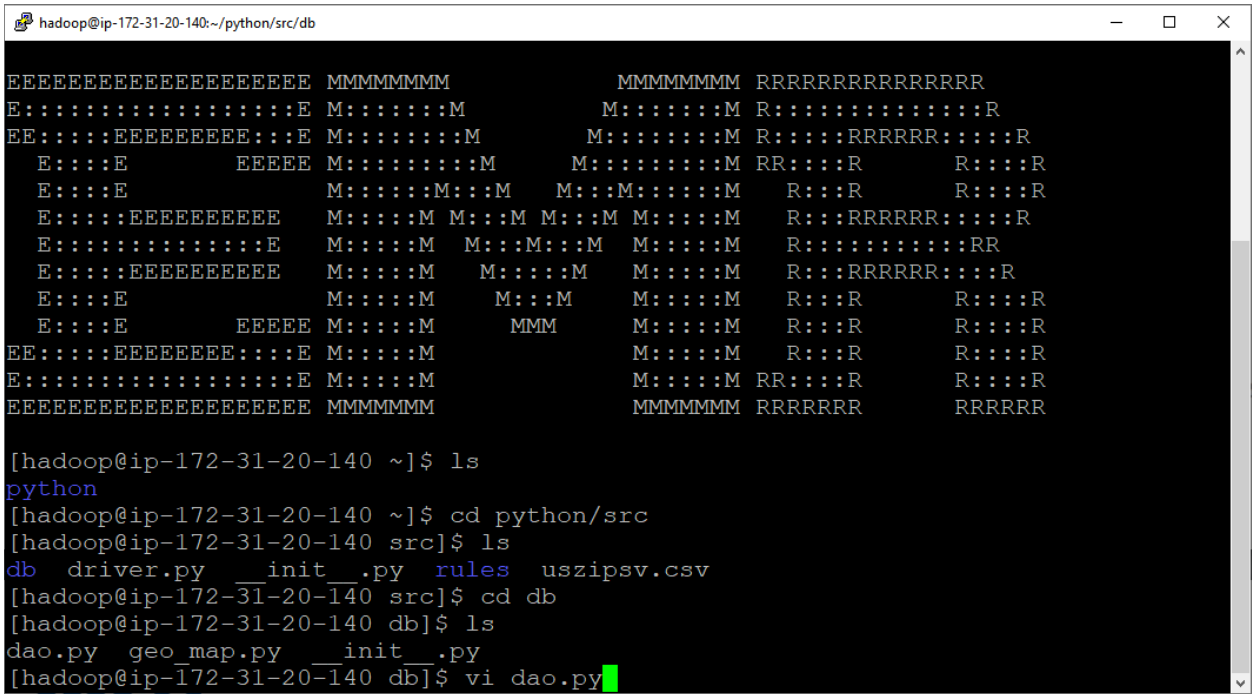
Name	Size	Changed	Rights	Own
..		27-01-2023 04:30:57	rwxr-xr-x	root
python		24-03-2023 02:05:31	rwxrwxr-x	hado

0 B of 299 KB in 0 of 2 | 0 B of 0 B in 0 of 1 | 5 hidden

SFTP-3 | 0:02:25

Console Screenshot 3: Copied the sample data provided by Upgrad.

3. After importing data into Hadoop Cluster, Editing **Dao.py** file

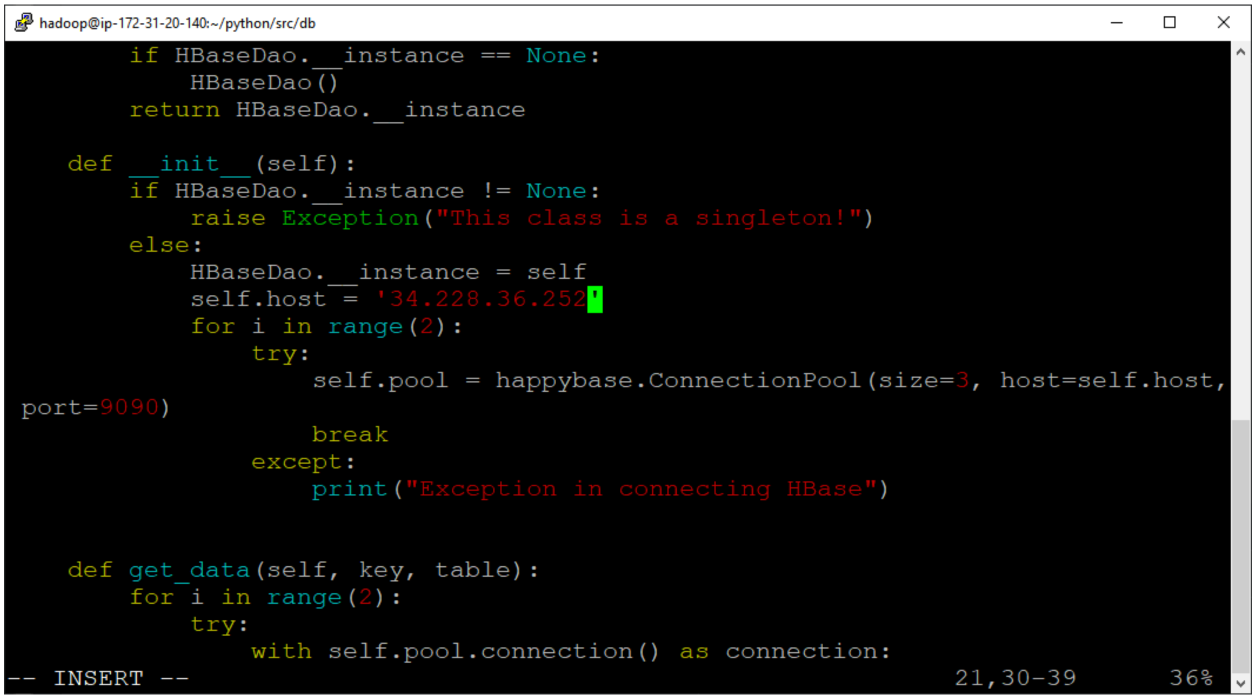


hadoop@ip-172-31-20-140:~/python/src/db

```
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-172-31-20-140 ~]$ ls
python
[hadoop@ip-172-31-20-140 ~]$ cd python/src
[hadoop@ip-172-31-20-140 src]$ ls
db  driver.py  __init__.py  rules  uszipsv.csv
[hadoop@ip-172-31-20-140 src]$ cd db
[hadoop@ip-172-31-20-140 db]$ ls
dao.py  geo_map.py  __init__.py
[hadoop@ip-172-31-20-140 db]$ vi dao.py
```

Console Screenshot 4: Editing dao.py file.

4. Replacing self.host with Public IP address of our Master node.



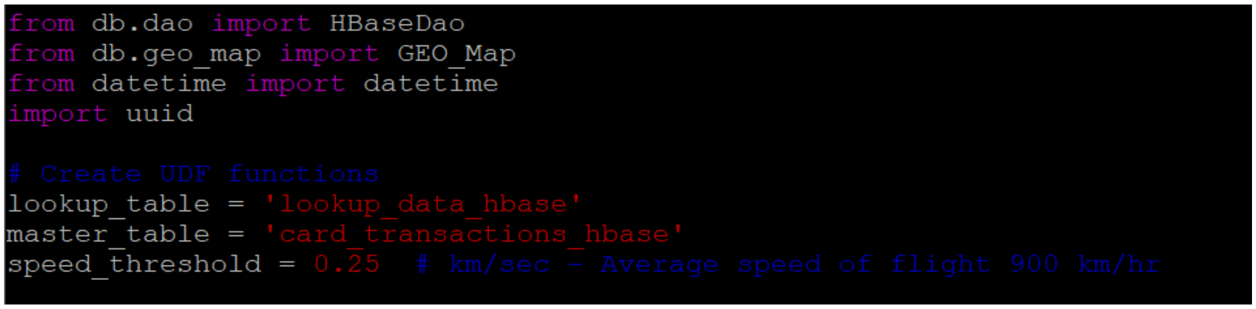
hadoop@ip-172-31-20-140:~/python/src/db

```
        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 = '34.228.36.252'
            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:
-- INSERT --                                                      21,30-39      36%
```

Console Screenshot 5: Replacing self-host IP address with master note public IP.

5. Updating rules.py with:

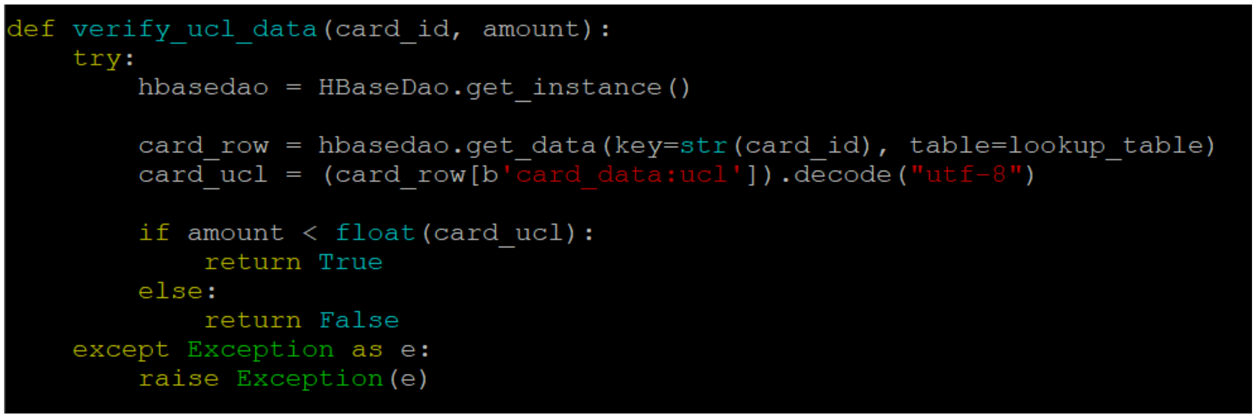
lookup_table = 'lookup_data_hbase'
master_table = 'card_transactions_hbase'



```
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
```

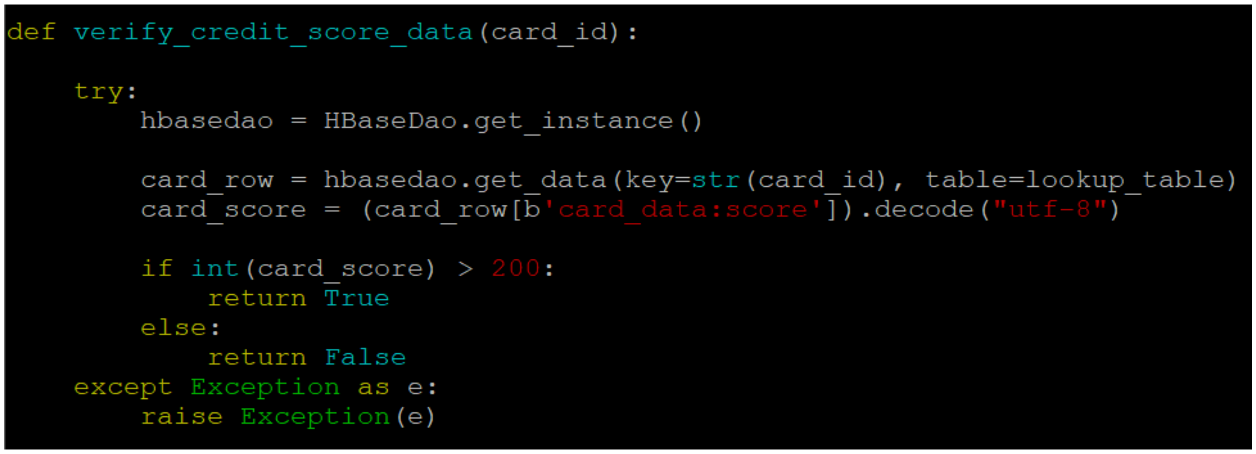
Console Screenshot 6: Created UDF function.

6. 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)



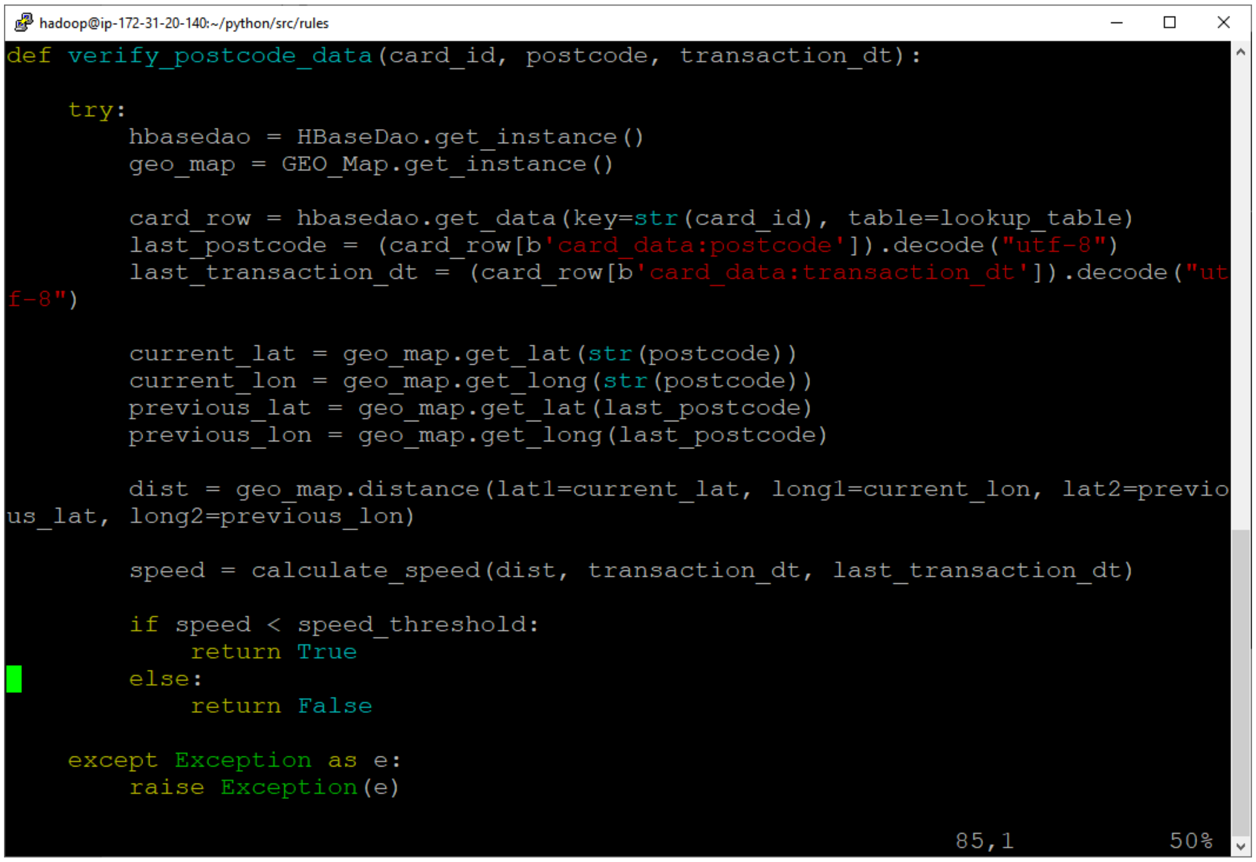
```python
def verify_ucl_data(card_id, amount):
    try:
        hbasedao = HBaseDao.get_instance()

        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)
```

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



```python
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)
```

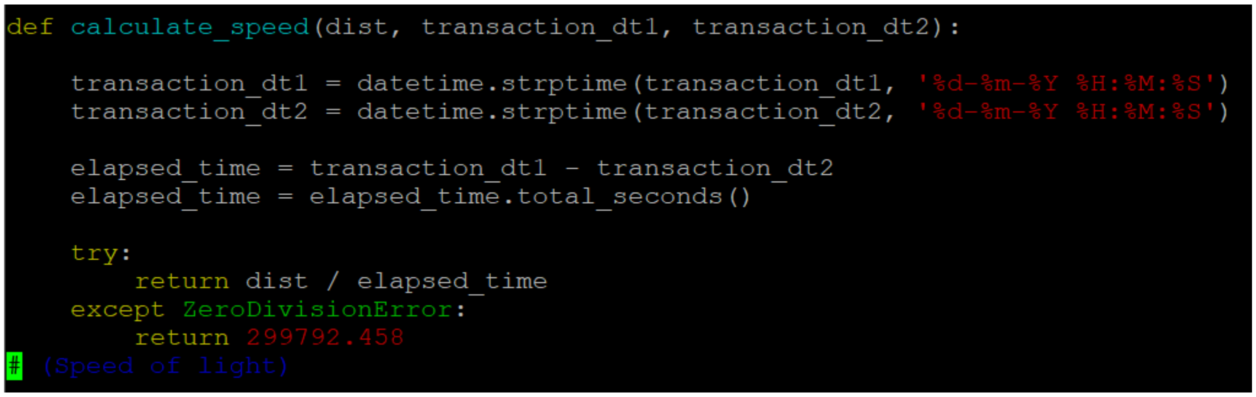
8. verify_postcode_data: Function to verify the following zipcode rules. ZIP code distance.



hadoop@ip-172-31-20-140:~/python/src/rules

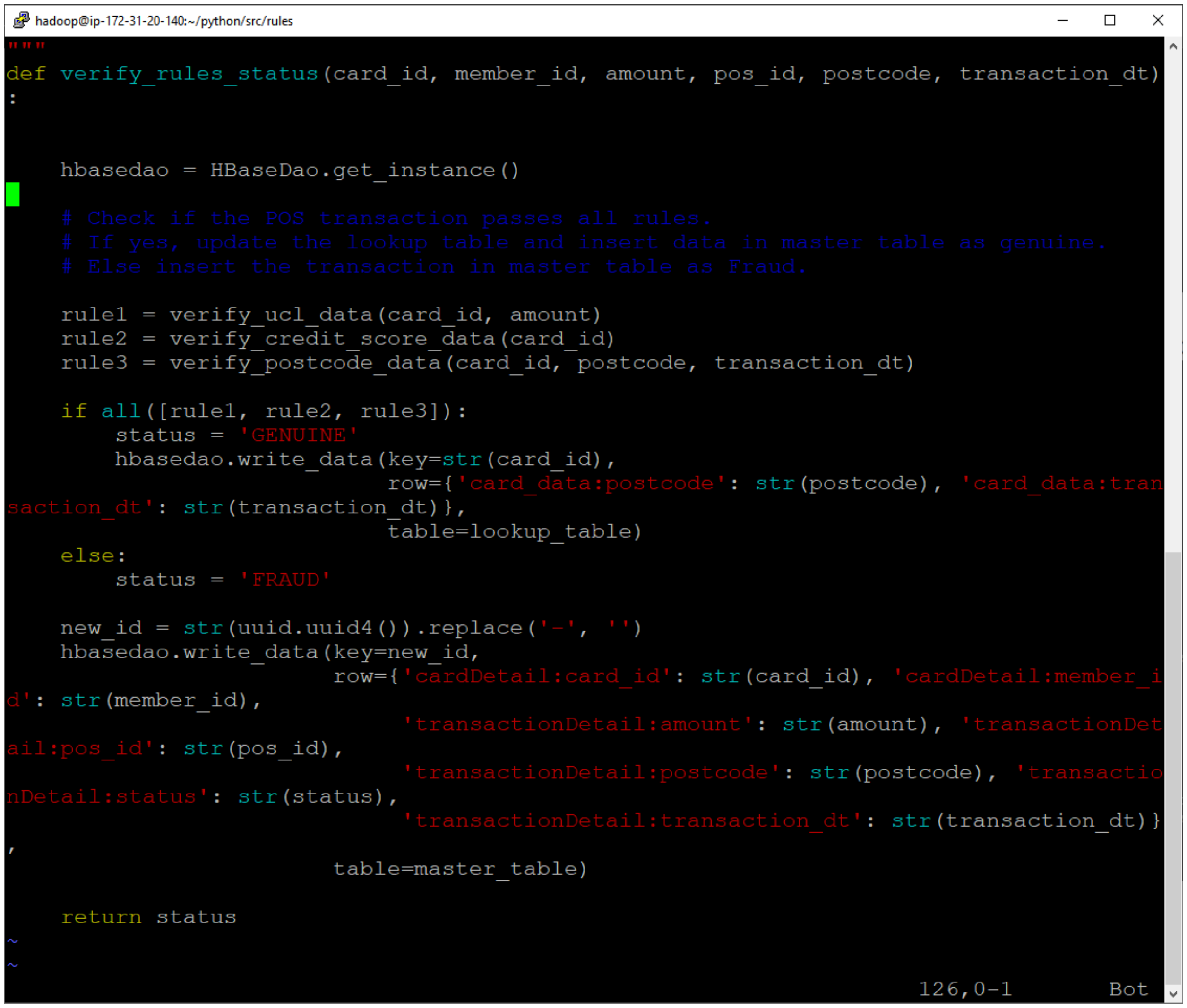
```
def verify_postcode_data(card_id, postcode, transaction_dt):

    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("ut
f-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=previo
us_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)
                                                              85,1          50%
```

9. 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)
```

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

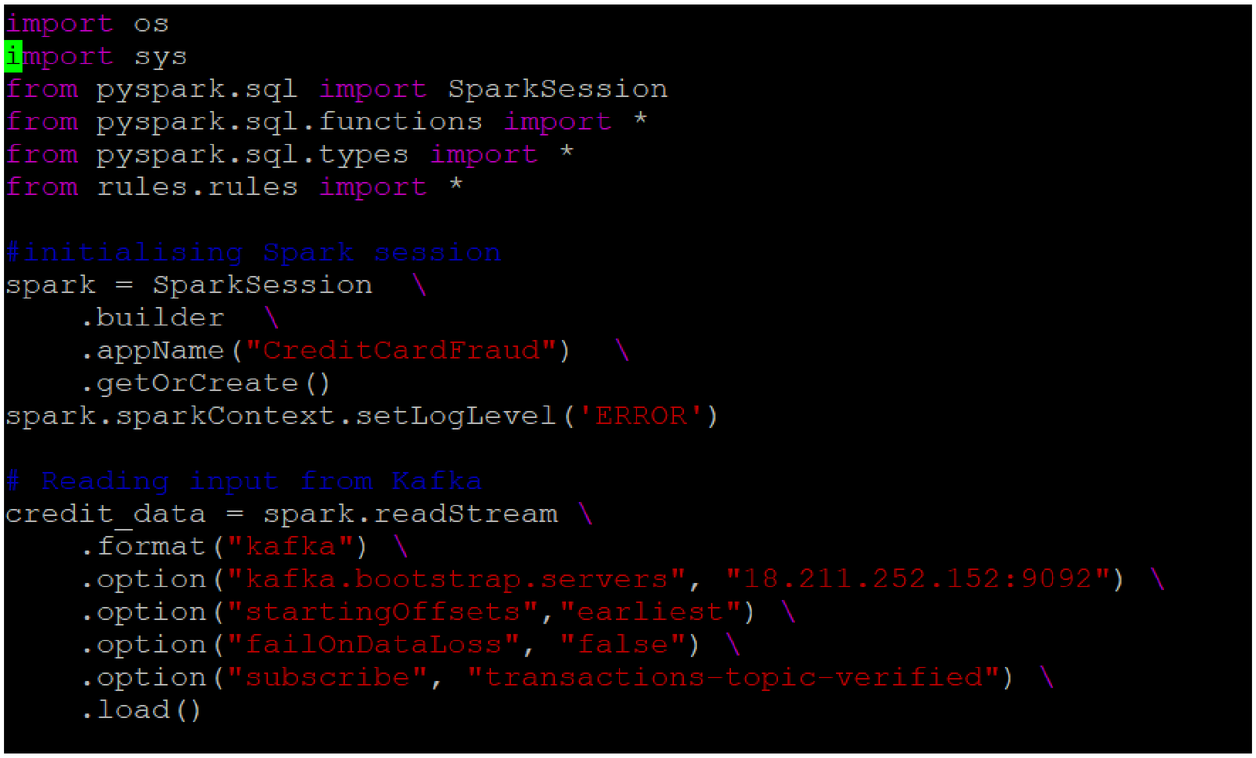


hadoop@ip-172-31-20-140:~/python/src/rules

```
"""
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:tran
saction_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_i
d': str(member_id),
                             'transactionDetail:amount': str(amount), 'transactionDet
ail:pos_id': str(pos_id),
                             'transactionDetail:postcode': str(postcode), 'transactio
nDetail:status': str(status),
                             'transactionDetail:transaction_dt': str(transaction_dt)}
,
                        table=master_table)

    return status
~
~
                                                              126,0-1       Bot
```

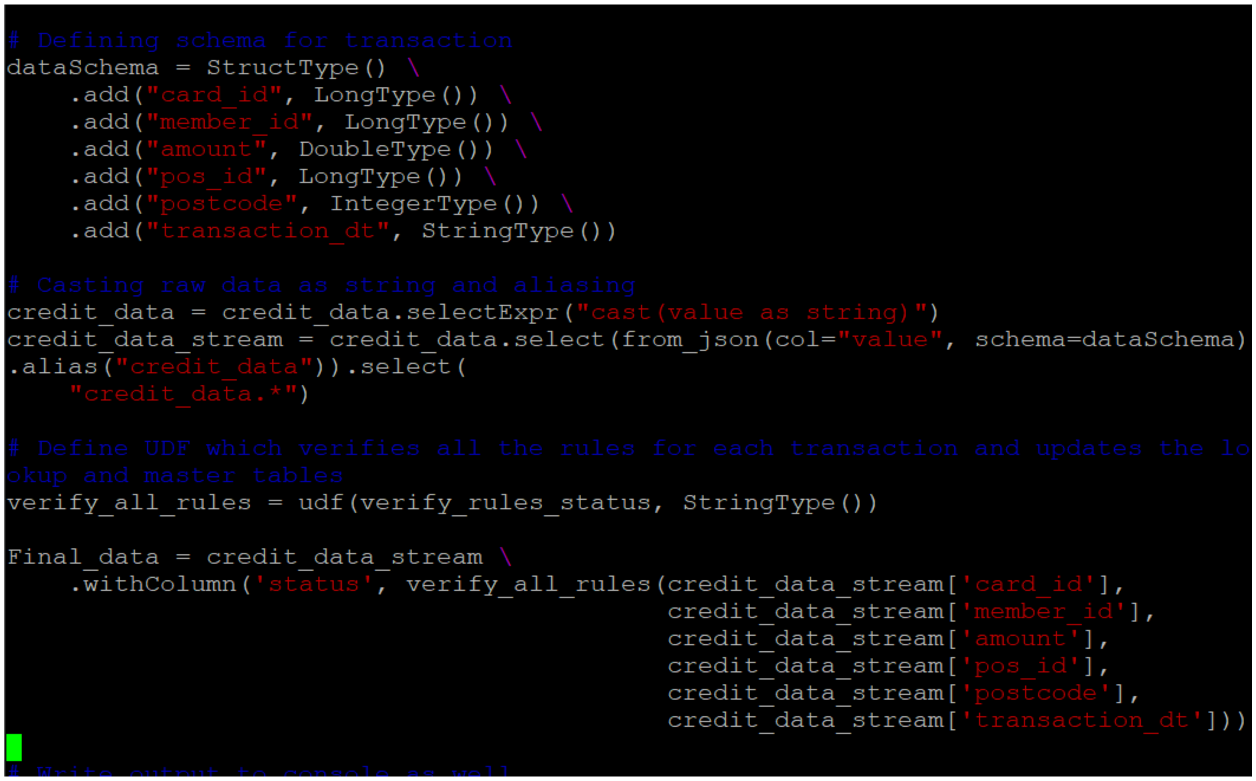
11. Now we are updating the **driver.py**
    a. Importing dependencies and setting Kafka consumer.
    b. Connecting to Kafka use the following details:
       Bootstrap-server: 18.211.252.152
       Port Number: 9092
       Topic: transactions-topic-verified

    c. Reading Input from Kafka



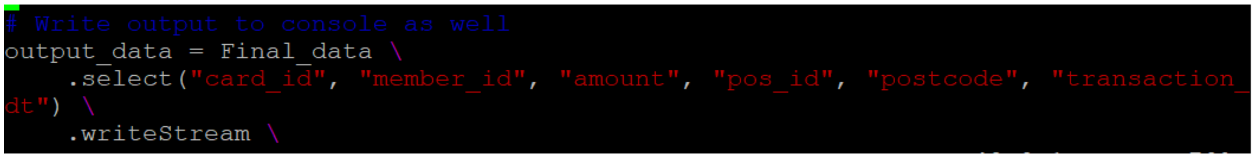
```
import os
import sys
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
from rules.rules import *

#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()
```

12. Defining JSON schema of each transaction & reading the raw JSON data from Kafka as ‘credit_data_stream’ and Defining UDF’s to verify rules.



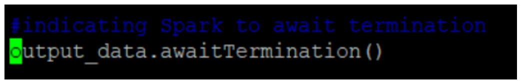
```
# Defining schema for transaction
dataSchema = StructType() \
    .add("card_id", LongType()) \
    .add("member_id", LongType()) \
    .add("amount", DoubleType()) \
    .add("pos_id", LongType()) \
    .add("postcode", IntegerType()) \
    .add("transaction_dt", StringType())

# 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 lo
okup 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']))

# Write output to console as well
```

13. Write output data to console.



```
# Write output to console as well
output_data = Final_data \
    .select("card_id", "member_id", "amount", "pos_id", "postcode", "transaction_
dt") \
    .writeStream \
```

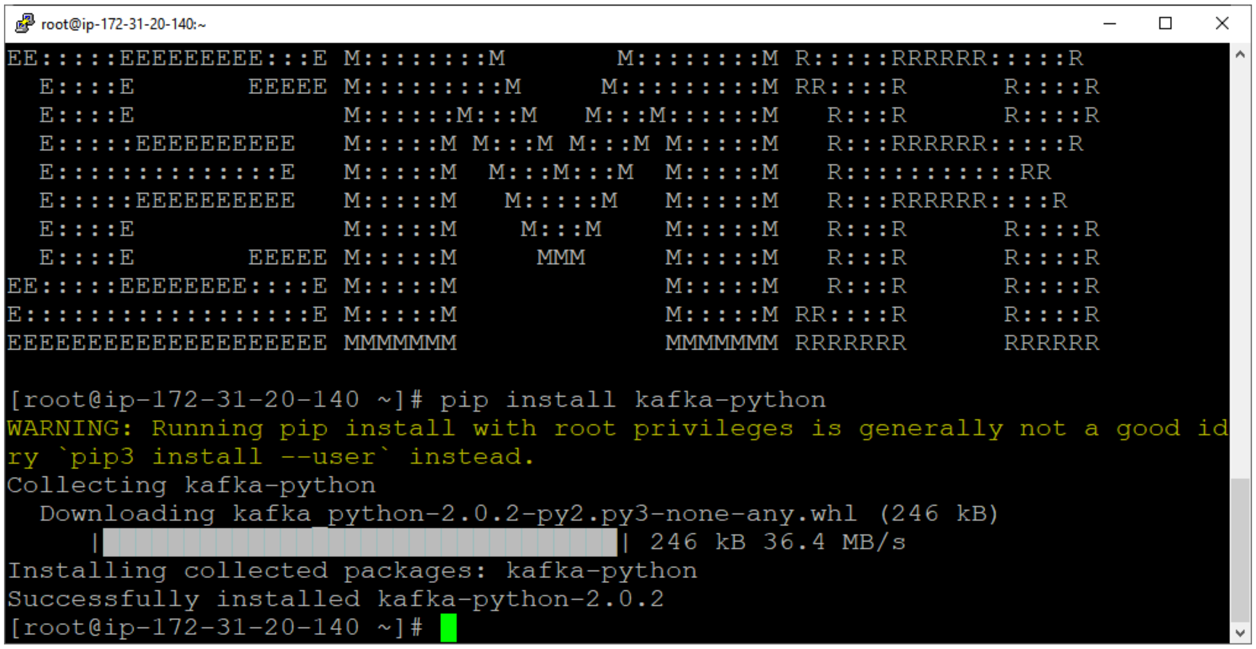
14. Spark Termination.



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

15. Installing Kafka Python using Root privileges

```
Sudo -i
pip install kafka-python
```



```
root@ip-172-31-20-140:~
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

[root@ip-172-31-20-140 ~]# pip install kafka-python
WARNING: Running pip install with root privileges is generally not a good id
ry `pip3 install --user` instead.
Collecting kafka-python
  Downloading kafka_python-2.0.2-py2.py3-none-any.whl (246 kB)
     |████████████████████████████████| 246 kB 36.4 MB/s
Installing collected packages: kafka-python
Successfully installed kafka-python-2.0.2
[root@ip-172-31-20-140 ~]#
```

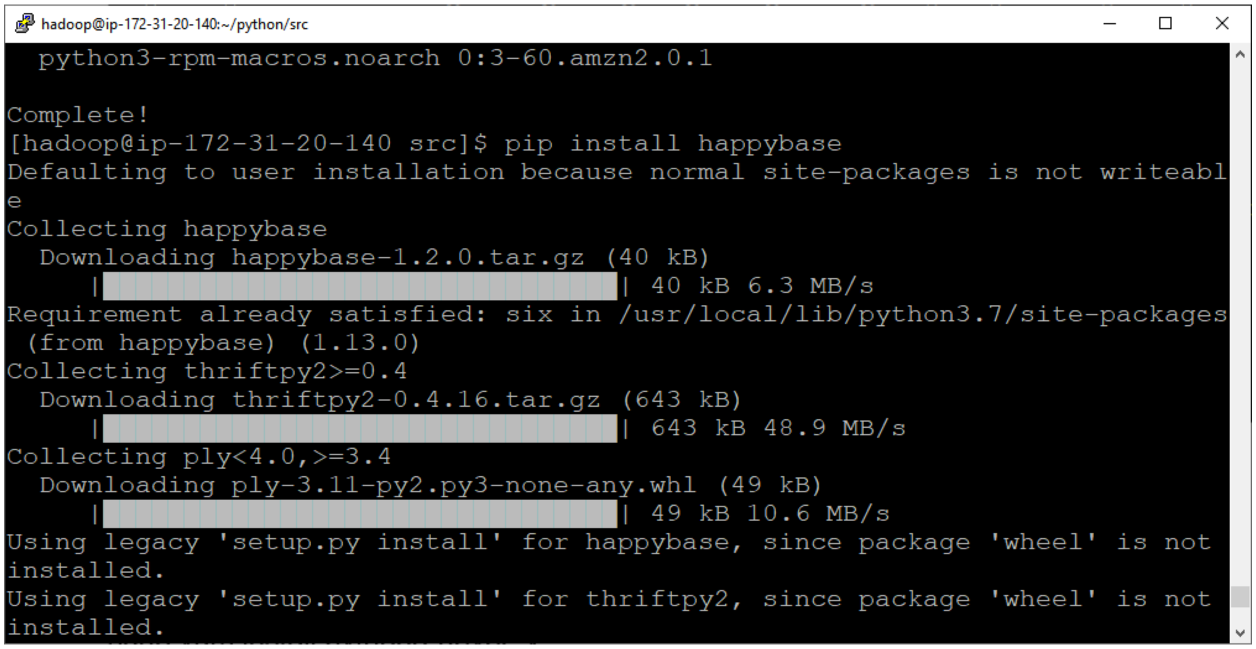
Console Screenshot 7: Installing Kafka python

16. Updating all the Linux packages

```
sudo yum update -y
```

17. Installing Happybase & Pandas

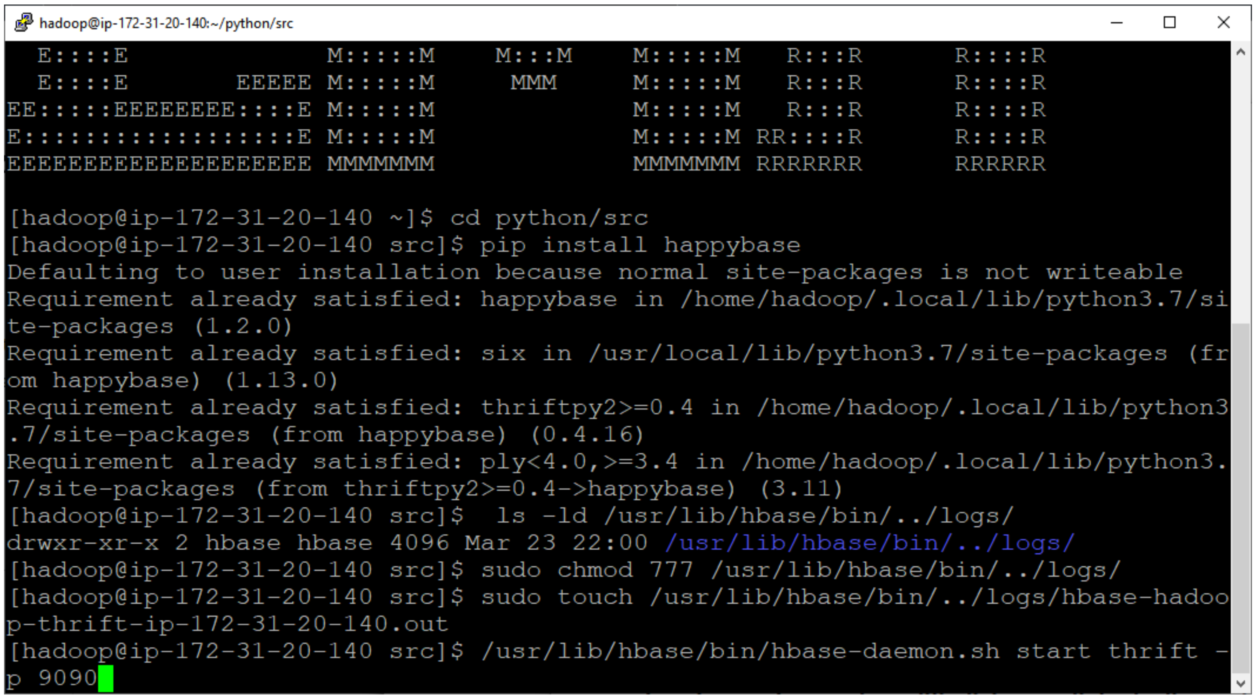
```
sudo yum install python3-devel -y
pip install happybase
pip install pandas
```



```
  python3-rpm-macros.noarch 0:3-60.amzn2.0.1

Complete!
[hadoop@ip-172-31-20-140 src]$ pip install happybase
Defaulting to user installation because normal site-packages is not writeabl
e
Collecting happybase
  Downloading happybase-1.2.0.tar.gz (40 kB)
     |████████████████████████████████| 40 kB 6.3 MB/s
Requirement already satisfied: six in /usr/local/lib/python3.7/site-packages
 (from happybase) (1.13.0)
Collecting thriftpy2>=0.4
  Downloading thriftpy2-0.4.16.tar.gz (643 kB)
     |████████████████████████████████| 643 kB 48.9 MB/s
Collecting ply<4.0,>=3.4
  Downloading ply-3.11-py2.py3-none-any.whl (49 kB)
     |████████████████████████████████| 49 kB 10.6 MB/s
Using legacy 'setup.py install' for happybase, since package 'wheel' is not
installed.
Using legacy 'setup.py install' for thriftpy2, since package 'wheel' is not
installed.
```

18. Making sure the permissions to directory for thrift services and starting the thrift service.

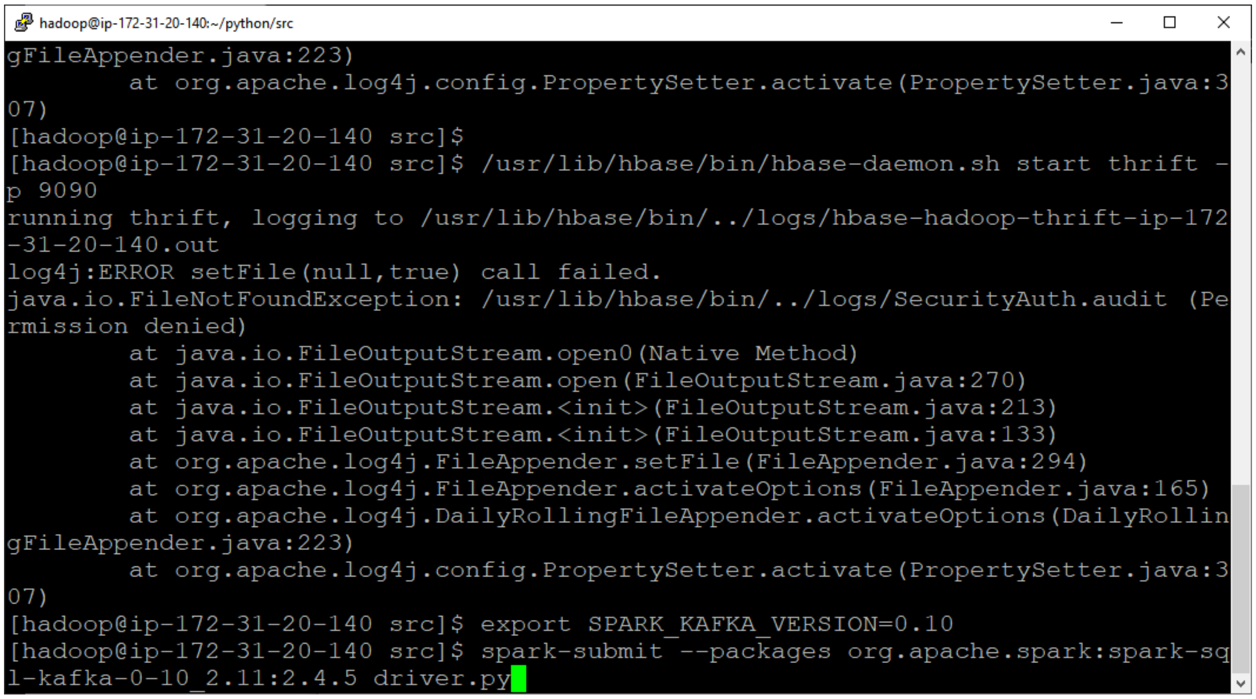
```
ls -ld /usr/lib/hbase/bin/../logs/
sudo chmod 777 /usr/lib/hbase/bin/../logs/
sudo touch /usr/lib/hbase/bin/../logs/hbase-hadoop-thrift-ip-172-31-20-140.out
/usr/lib/hbase/bin/hbase-daemon.sh start thrift -p 9090
```



```
  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-172-31-20-140 ~]$ cd python/src
[hadoop@ip-172-31-20-140 src]$ pip install happybase
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: happybase in /home/hadoop/.local/lib/python3.7/si
te-packages (1.2.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/site-packages (fr
om happybase) (1.13.0)
Requirement already satisfied: thriftpy2>=0.4 in /home/hadoop/.local/lib/python3
.7/site-packages (from happybase) (0.4.16)
Requirement already satisfied: ply<4.0,>=3.4 in /home/hadoop/.local/lib/python3.
7/site-packages (from thriftpy2>=0.4->happybase) (3.11)
[hadoop@ip-172-31-20-140 src]$  ls -ld /usr/lib/hbase/bin/../logs/
drwxr-xr-x 2 hbase hbase 4096 Mar 23 22:00 /usr/lib/hbase/bin/../logs/
[hadoop@ip-172-31-20-140 src]$ sudo chmod 777 /usr/lib/hbase/bin/../logs/
[hadoop@ip-172-31-20-140 src]$ sudo touch /usr/lib/hbase/bin/../logs/hbase-hadoo
p-thrift-ip-172-31-20-140.out
[hadoop@ip-172-31-20-140 src]$ /usr/lib/hbase/bin/hbase-daemon.sh start thrift -
p 9090
```

19. Setting Kafka version and running spark submit command.

```
export SPARK_KAFKA_VERSION=0.10
spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 driver.py
```

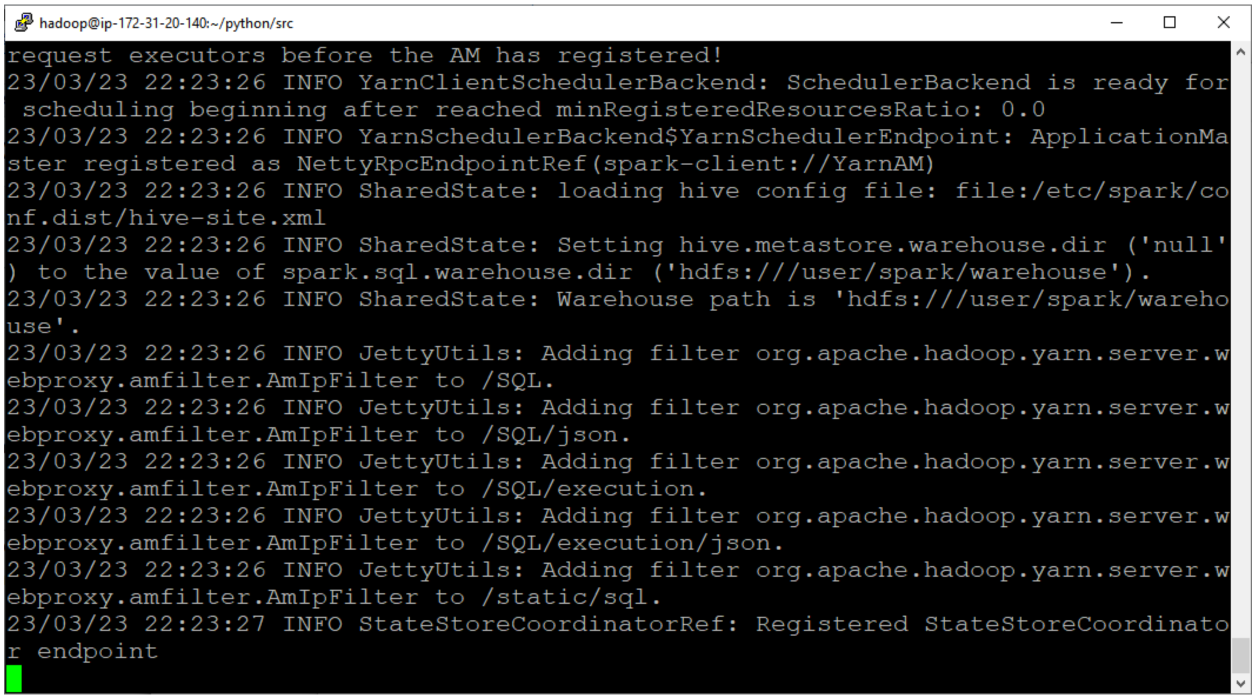


hadoop@ip-172-31-20-140:~/python/src

```
gFileAppender.java:223)
        at org.apache.log4j.config.PropertySetter.activate(PropertySetter.java:3
07)
[hadoop@ip-172-31-20-140 src]$
[hadoop@ip-172-31-20-140 src]$ /usr/lib/hbase/bin/hbase-daemon.sh start thrift -
p 9090
running thrift, logging to /usr/lib/hbase/bin/../logs/hbase-hadoop-thrift-ip-172
-31-20-140.out
log4j:ERROR setFile(null,true) call failed.
java.io.FileNotFoundException: /usr/lib/hbase/bin/../logs/SecurityAuth.audit (Pe
rmission denied)
        at java.io.FileOutputStream.open0(Native Method)
        at java.io.FileOutputStream.open(FileOutputStream.java:270)
        at java.io.FileOutputStream.<init>(FileOutputStream.java:213)
        at java.io.FileOutputStream.<init>(FileOutputStream.java:133)
        at org.apache.log4j.FileAppender.setFile(FileAppender.java:294)
        at org.apache.log4j.FileAppender.activateOptions(FileAppender.java:165)
        at org.apache.log4j.DailyRollingFileAppender.activateOptions(DailyRollin
gFileAppender.java:223)
        at org.apache.log4j.config.PropertySetter.activate(PropertySetter.java:3
07)
[hadoop@ip-172-31-20-140 src]$ export SPARK_KAFKA_VERSION=0.10
[hadoop@ip-172-31-20-140 src]$ spark-submit --packages org.apache.spark:spark-sq
l-kafka-0-10_2.11:2.4.5 driver.py
```

Console Screenshot 8: Spark Submit Command

18. Count Data in Hbase: count 'lookup_data_hive’



hadoop@ip-172-31-20-140:~/python/src

```
request executors before the AM has registered!
23/03/23 22:23:26 INFO YarnClientSchedulerBackend: SchedulerBackend is ready for
 scheduling beginning after reached minRegisteredResourcesRatio: 0.0
23/03/23 22:23:26 INFO YarnSchedulerBackend$YarnSchedulerEndpoint: ApplicationMa
ster registered as NettyRpcEndpointRef(spark-client://YarnAM)
23/03/23 22:23:26 INFO SharedState: loading hive config file: file:/etc/spark/co
nf.dist/hive-site.xml
23/03/23 22:23:26 INFO SharedState: Setting hive.metastore.warehouse.dir ('null'
) to the value of spark.sql.warehouse.dir ('hdfs:///user/spark/warehouse').
23/03/23 22:23:26 INFO SharedState: Warehouse path is 'hdfs:///user/spark/wareho
use'.
23/03/23 22:23:26 INFO JettyUtils: Adding filter org.apache.hadoop.yarn.server.w
ebproxy.amfilter.AmIpFilter to /SQL.
23/03/23 22:23:26 INFO JettyUtils: Adding filter org.apache.hadoop.yarn.server.w
ebproxy.amfilter.AmIpFilter to /SQL/json.
23/03/23 22:23:26 INFO JettyUtils: Adding filter org.apache.hadoop.yarn.server.w
ebproxy.amfilter.AmIpFilter to /SQL/execution.
23/03/23 22:23:26 INFO JettyUtils: Adding filter org.apache.hadoop.yarn.server.w
ebproxy.amfilter.AmIpFilter to /SQL/execution/json.
23/03/23 22:23:26 INFO JettyUtils: Adding filter org.apache.hadoop.yarn.server.w
ebproxy.amfilter.AmIpFilter to /static/sql.
23/03/23 22:23:27 INFO StateStoreCoordinatorRef: Registered StateStoreCoordinato
r endpoint
```

Please Note:

1. Due to the issue with Kafka server, I could not be able to get any data. However, once Batch 0 has data printed into console. We should be able to verify the number of entries.

2. We also tried doing it with the static data which was shared by upgrad in text format. We bypassed Kafka and imported directly using spark streaming from the local storage but even after numerous attempts spark could not access or find the file. We checked the read and write permission as well, but we kept getting the error that file does not exist.

Code for which was replaced in with Kafka Consumer details:

```
# Reading input from a text file

credit_data = spark.readStream \
    .option("inferSchema", "true") \
    .text("/home/hadoop/python/src/transactions-topic-verified.txt")
```