Big Data Management and Analytics

Big Data Management

Assignment3

Aravindan Srinivasan

2981707

aravindan@student.griffith.ie

Answers:

Initially, Deleted the existing streaming directory and created a new streaming directory.

Code:

```
import shutil

shutil.rmtree('streaming/')

os.mkdir("streaming/")
```

Then, I have accessed two log files “2019-03-01.csv” and “2019-03-02.csv” and loaded into the streaming directory, while the application is listening. Below image1 represents that our application started and accepting the values from the live stream.

Code:

```
from pyspark.streaming import StreamingContext

import time

ssc = StreamingContext(sc , 5)

lines = ssc.textFileStream("file:///home/aravindan81222/streaming/")

words=lines.flatMap(lambda line:line.split(" "))

pairs=words.map(lambda word:(word,1))

wordCounts=pairs.reduceByKey(lambda x,y : x+y)

wordCounts.pprint()

ssc.start()

time.sleep(60)

ssc.stop(stopSparkContext=False)
```

Image1:

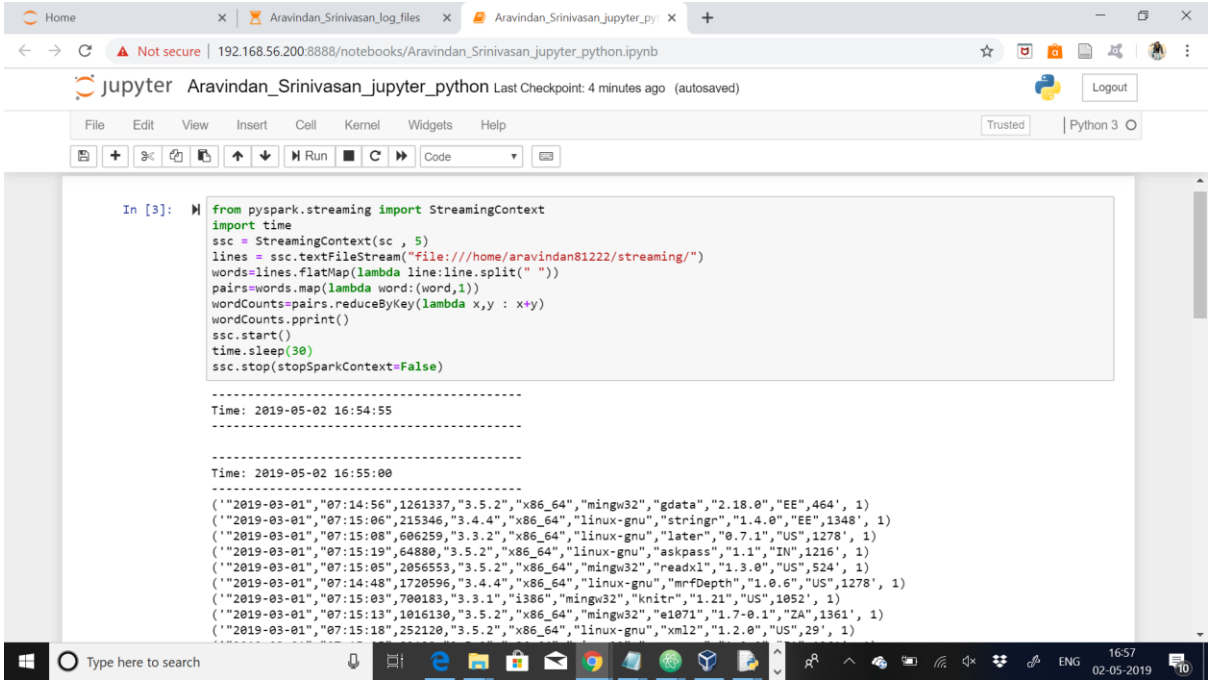
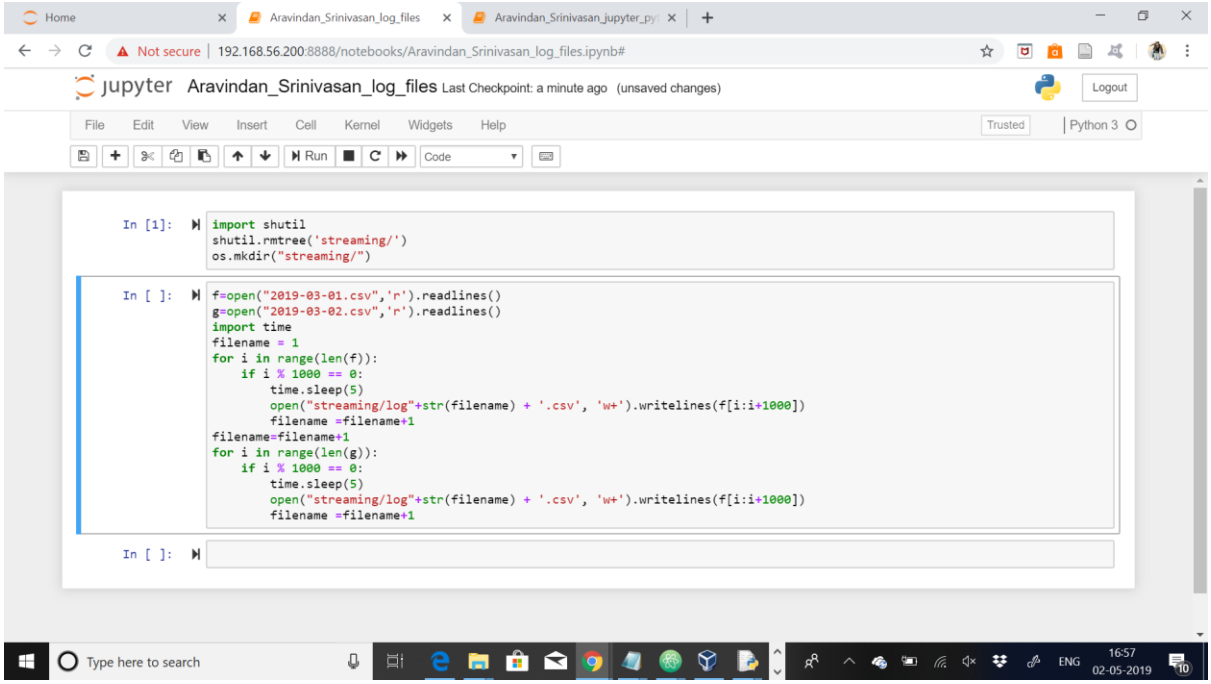


Image2 represents the code for accessing the log files and diving 1000 lines and posting into each of its log file (For instance log1,log2 etc).

Code:

```
f=open("2019-03-01.csv",'r').readlines()

g=open("2019-03-02.csv",'r').readlines()

import time

filename = 1

for i in range(len(f)):

    if i % 1000 == 0:

        time.sleep(5)

        open("streaming/log"+str(filename) + '.csv', 'w+').writelines(f[i:i+1000])

        filename =filename+1

filename=filename+1

for i in range(len(g)):

    if i % 1000 == 0:

        time.sleep(5)

        open("streaming/log"+str(filename) + '.csv', 'w+').writelines(f[i:i+1000])

        filename =filename+1
```

Image2:



Queries: Its time to access each and every query with the streaming computations with 5 seconds batch interval.

Query:

To find number of downloads for ggplot2 package

This query wants us to find total number of ggplot2 packages in the streaming computation (5 seconds).

Code:

```
from pyspark.streaming import StreamingContext

import time

ssc=StreamingContext(sc, 5)

lines = ssc.textFileStream("file:///home/aravindan81222/streaming/")

downloads_RDD = lines.map(lambda x: x.split(','))

def remove_quotation(x):

        return([xx.replace("","")for xx in x])

downloads_RDD = downloads_RDD.map(remove_quotation)

five = downloads_RDD.map(lambda x: (x[6], 1))

five = five.reduceByKey(lambda a,b: a + b)

five = five.filter(lambda x: "ggplot2" in x)
```

#five.foreachRDD(save)

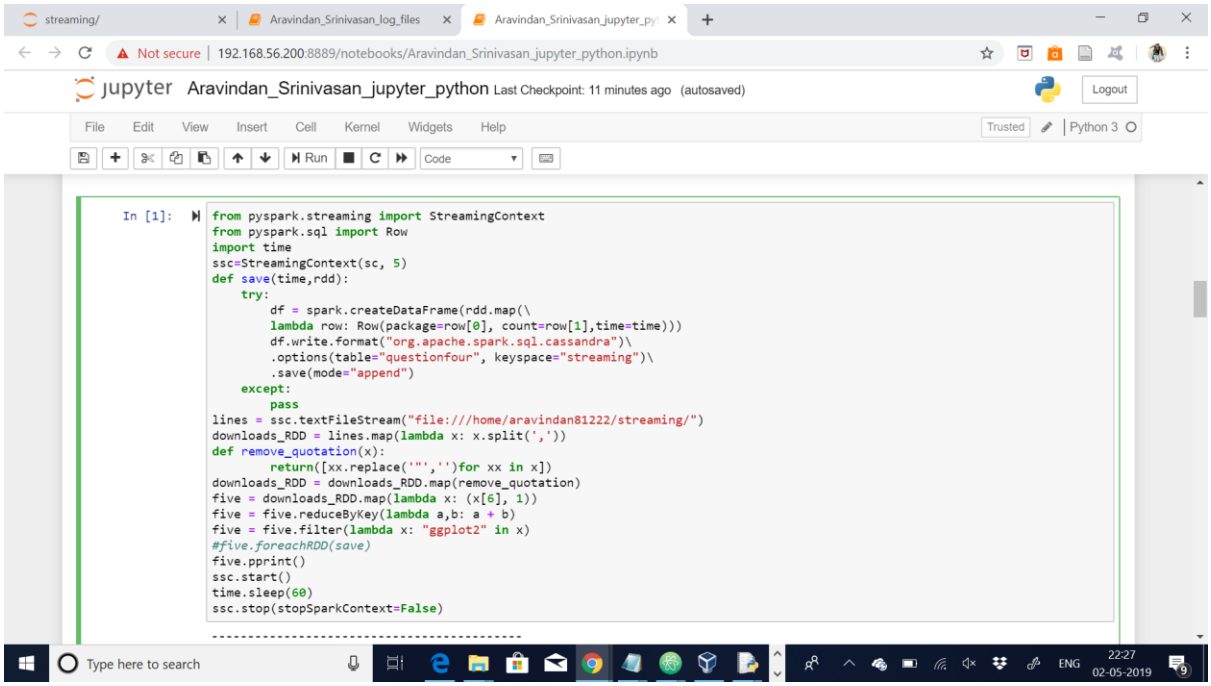
five.pprint()

ssc.start()

time.sleep(60)

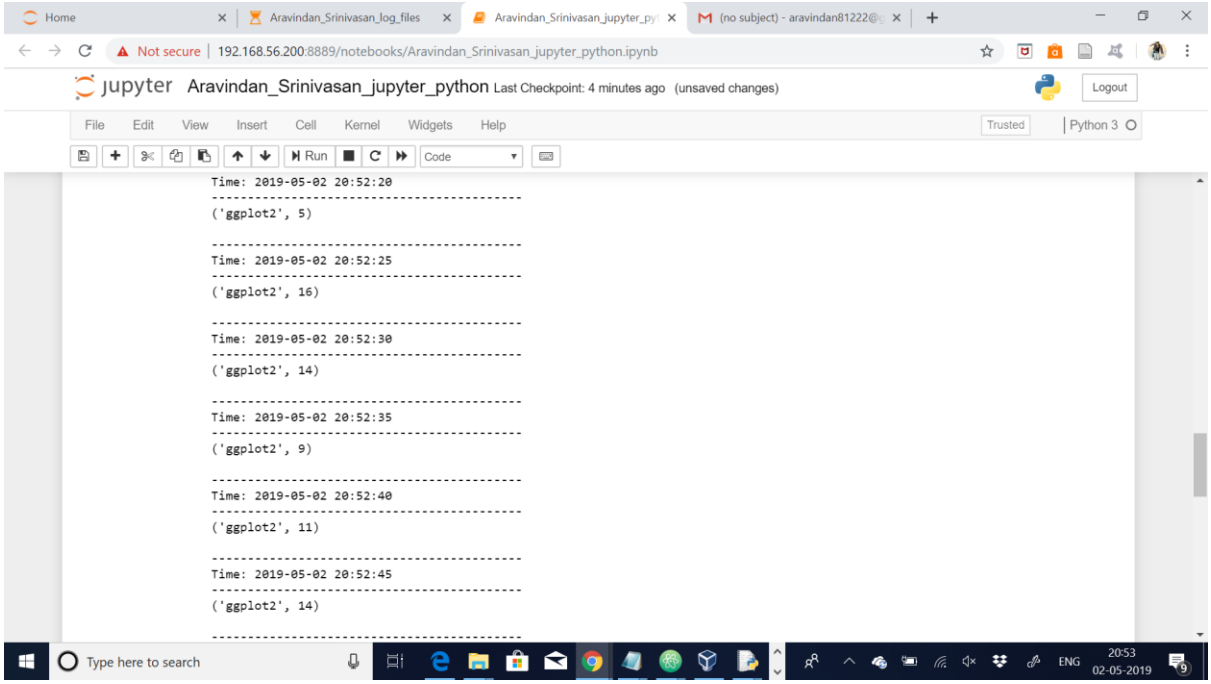
ssc.stop(stopSparkContext=False)

image:



Now it should display ggplot2

<u>Image:</u>



Query:

To calculate the number of downloads of each package.

In this query, we are about to find the total number of each and every package.

Code:

```
from pyspark.streaming import StreamingContext
import time
ssc  = StreamingContext(sc, 5)
lines = ssc.textFileStream("file:///home/aravindan81222/streaming/")
downloads_RDD = lines.map(lambda x: x.split(','))
def remove_quotation(x):
        return([xx.replace("","")for xx in x])
downloads_RDD = downloads_RDD.map(remove_quotation)
words = downloads_RDD.map(lambda x: (x[6], 1))
words = words.reduceByKey(lambda a , b: a + b)
#values.foreachRDD(save)
words.pprint()
ssc.start()
time.sleep(60)
ssc.stop(stopSparkContext=False)
```

image:

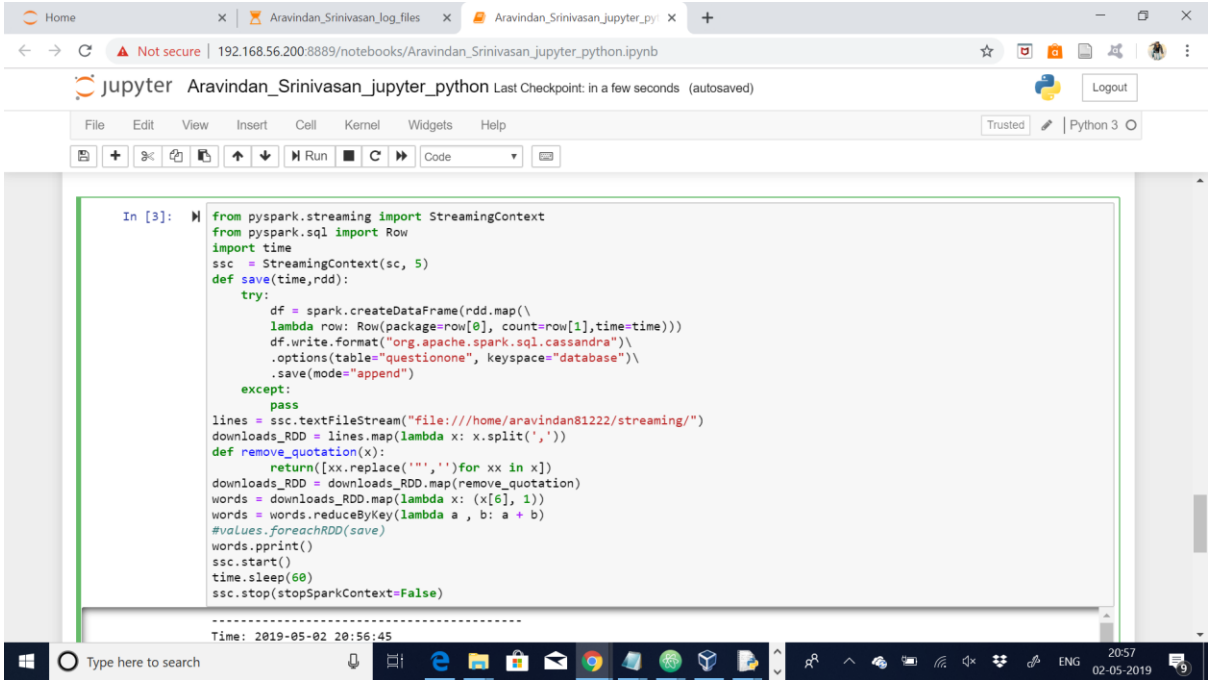
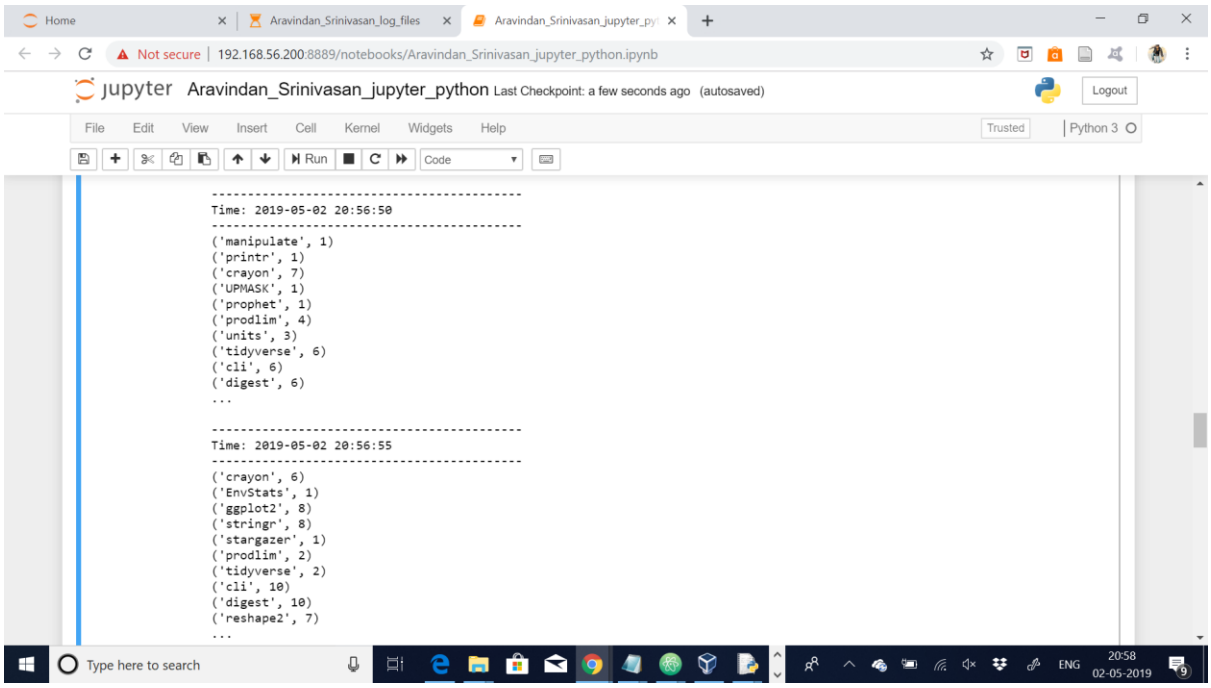


Image:



Query:

To find the top most downloaded package. We are about to find the most downloaded package.

Code:

from pyspark.streaming import StreamingContext

import time

```
ssc  = StreamingContext(sc, 5)
lines = ssc.textFileStream("file:///home/aravindan81222/streaming/")
downloads_RDD=lines.map(lambda line:line.split(","))
def remove_quotation(x):
        return([xx.replace('"','')for xx in x])
downloads_RDD=downloads_RDD.map(remove_quotation)
words=downloads_RDD.map(lambda y:(y[6],1))
words = words.reduceByKey(lambda a , b: a + b)
words = words.transform((lambda variable:variable.sortBy(lambda x:(-x[1]))))
words.pprint(1)
ssc.start()
time.sleep(60)
ssc.stop(stopSparkContext=False)
```

image:

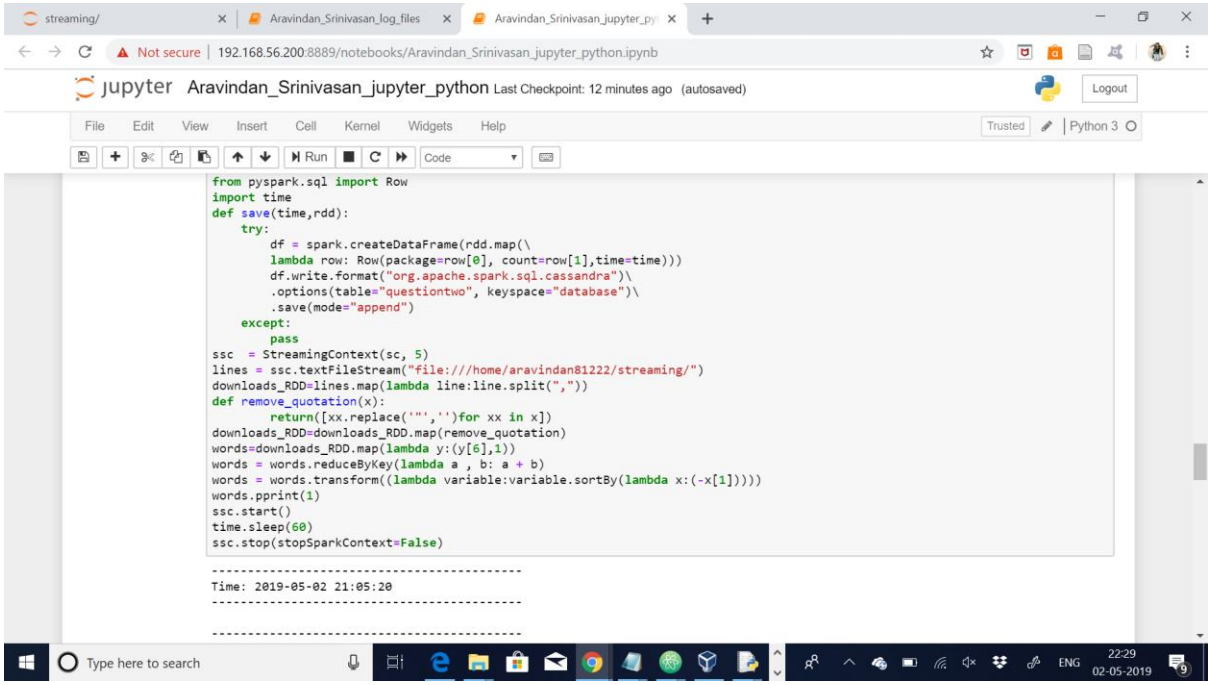
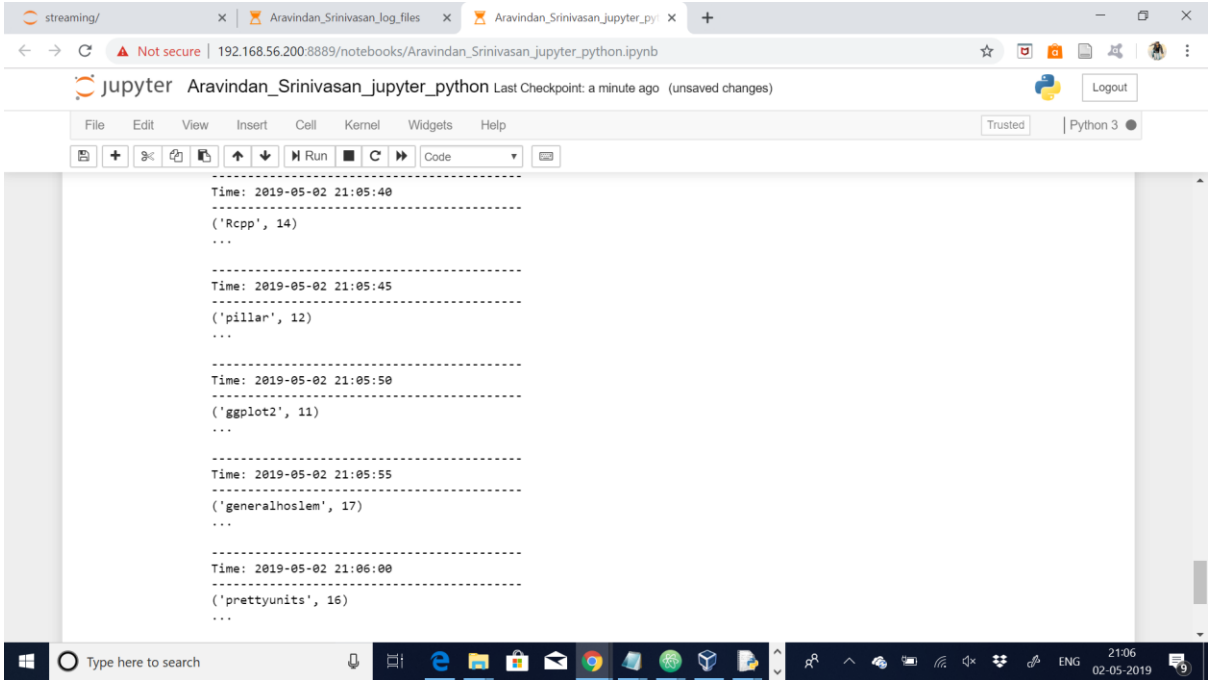


Image:



Query:

To find the top 5 countries along with number of downloads. In this query, we are finding the top 5 countries along with the number of downloads.

Code:

```
from pyspark.streaming import StreamingContext

import time

ssc  = StreamingContext(sc, 5)

lines = ssc.textFileStream("file:///home/aravindan81222/streaming/")

downloads_RDD=lines.map(lambda line:line.split(","))

def remove_quotation(x):

        return([xx.replace("","")for xx in x])

downloads_RDD=downloads_RDD.map(remove_quotation)

words=downloads_RDD.map(lambda y:(y[8],1))

words = words.reduceByKey(lambda a , b: a + b)

words = words.transform((lambda variable:variable.sortBy(lambda x:(-x[1]))))

words.pprint(5)

ssc.start()

time.sleep(60)

ssc.stop(stopSparkContext=False)
```

image:

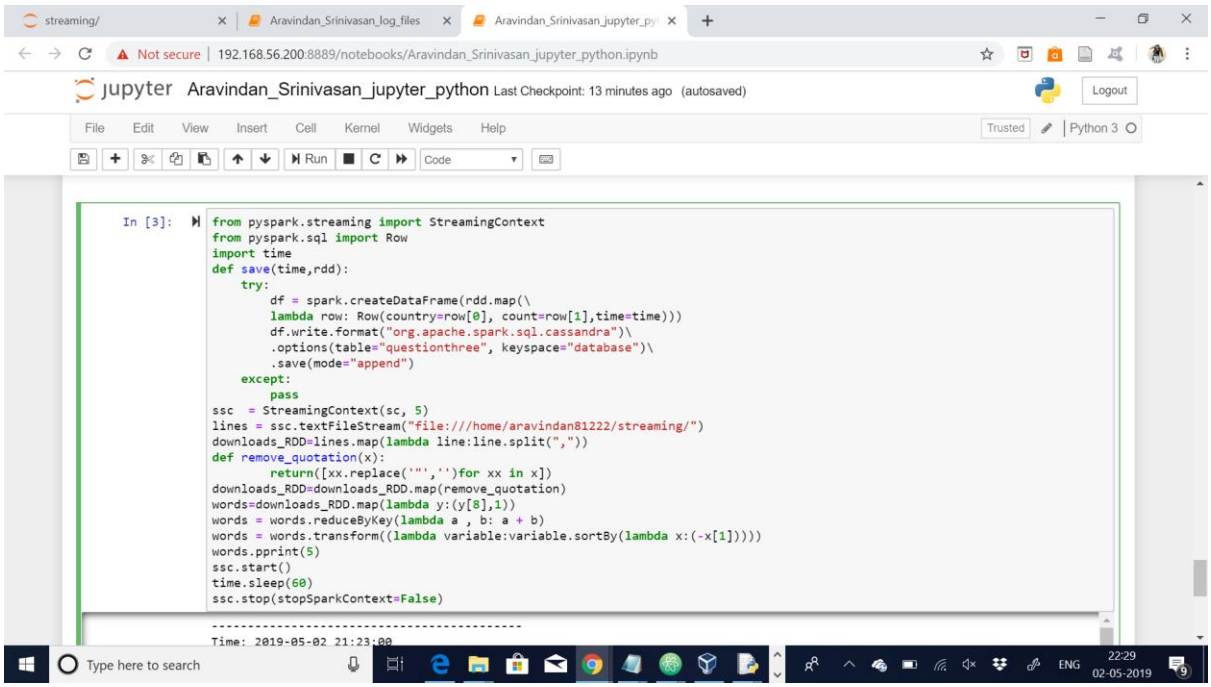
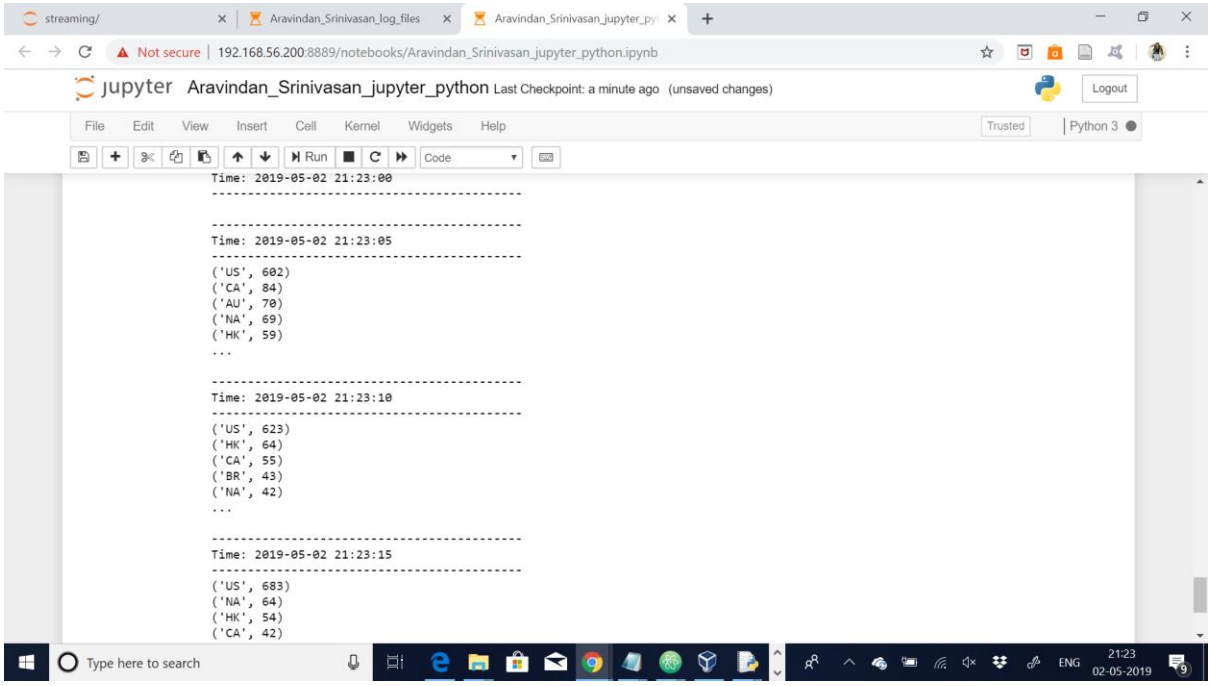


Image:



Cassandra:

Partial output:

Query for creating tables:

streaming(keyspace):

keyspace creation:

CREATE KEYSPACE streaming WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };

use streaming;

1.CREATE TABLE questionone (time text, package text, count int, PRIMARY KEY(time, package));

<u>Code:</u>

```
def save(time,rdd):
    try:
        df = spark.createDataFrame(rdd.map(\
        lambda row: Row(package=row[0], count=row[1],time=time)))
        df.write.format("org.apache.spark.sql.cassandra")\
        .options(table="questionone", keyspace="database")\
        .save(mode="append")
    except:
        pass
```

2. CREATE TABLE questiontwo (time text, package text, count int, PRIMARY KEY(time, package) ;

<u>Code:</u>

```
def save(time,rdd):
    try:
        df = spark.createDataFrame(rdd.map(\
        lambda row: Row(package=row[0], count=row[1],time=time)))
        df.write.format("org.apache.spark.sql.cassandra")\
        .options(table="questiontwo", keyspace="database")\
        .save(mode="append")
    except:
        pass
```

3. CREATE TABLE questionthree (time text, country text, count int, PRIMARY KEY(time, package));

<u>Code:</u>

```
def save(time,rdd):
    try:
        df = spark.createDataFrame(rdd.map(\
        lambda row: Row(country=row[0], count=row[1],time=time)))
        df.write.format("org.apache.spark.sql.cassandra")\
        .options(table="questionthree", keyspace="database")\
        .save(mode="append")
    except:
        pass
```

4.CREATE TABLE questionfour (time text, package text, count int, PRIMARY KEY(time, package));

<u>Code:</u>

```
def save(time,rdd):
    try:
        df = spark.createDataFrame(rdd.map(\
        lambda row: Row(package=row[0], count=row[1],time=time)))
        df.write.format("org.apache.spark.sql.cassandra")\
        .options(table="questionfour", keyspace="streaming")\
        .save(mode="append")
    except:
        pass
```