Phase II- Design and Implementation of Twitter Analysis using Spark

By,

Tejaswi Ganne(16225529), tgqkc@mail.umkc.edu

Muddu Latha(16226316), mltx9@mail.umkc.edu

Rajaramya Reddy(16226315), rjph6@mail.umkc.edu

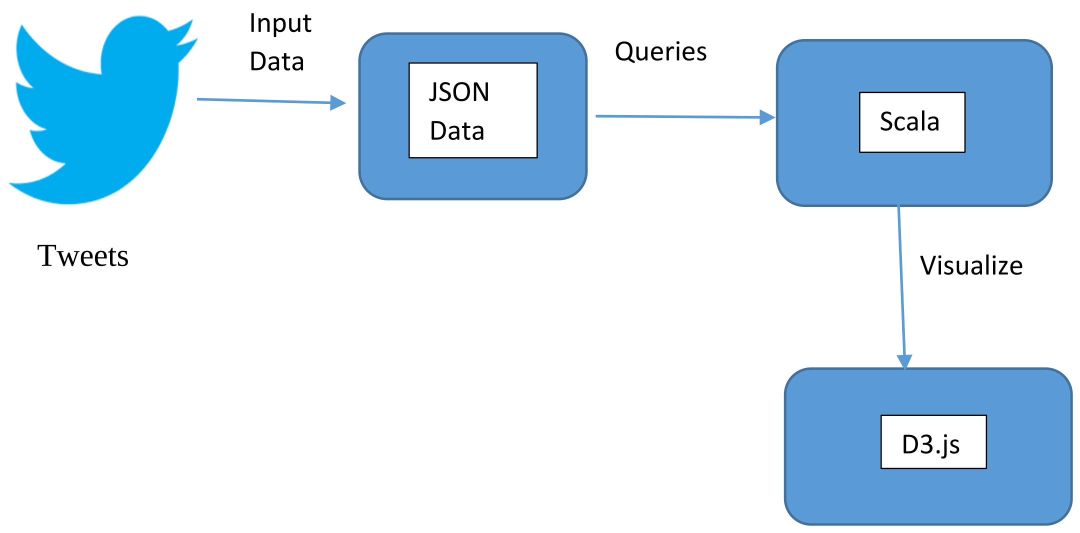
Introduction:

We collected tweets on cricket and basket ball . Then we started analyzing the data on the tweets. Using different attributes we have written queries and executed. The query outputs are visualized and different graphs are generated based on the visualized data. This report gives the delineation of the data collected from the twitter and how the visualization is performed.

Technologies Used:

1. Scala
2. Spark (for running spark SQL queries)
3. D3.js (for Visualization)
4. IntelliJ
5. Python

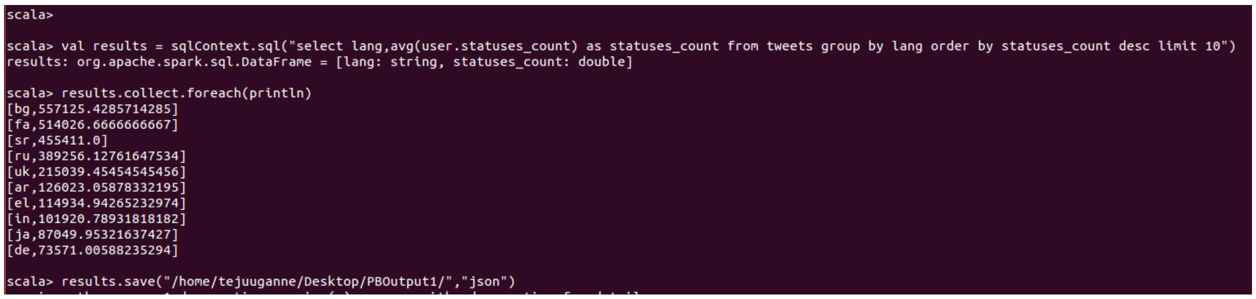
Design:



Queries:

1. This query is to get the count of tweets in different languages

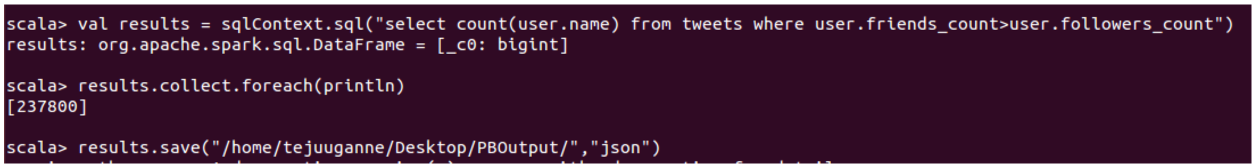
sqlContext.sql("select lang,avg(user.statuses_count) as statuses_count from tweets group by lang order by statuses_count desc limit 10")



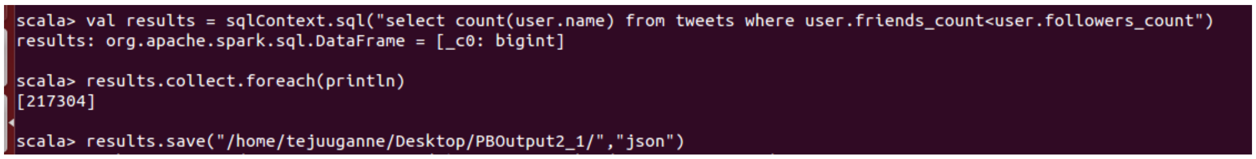
```
scala>

scala> val results = sqlContext.sql("select lang,avg(user.statuses_count) as statuses_count from tweets group by lang order by statuses_count desc limit 10")
results: org.apache.spark.sql.DataFrame = [lang: string, statuses_count: double]

scala> results.collect.foreach(println)
[bg,557125.4285714285]
[fa,514026.6666666667]
[sr,455411.0]
[ru,389256.12761647534]
[uk,215039.45454545456]
[ar,126023.05878332195]
[el,114934.94265232974]
[in,101920.78931818182]
[ja,87049.95321637427]
[de,73571.00588235294]

scala> results.save("/home/tejuuganne/Desktop/PBOutput1/","json")
```

2. This query gives the comparison between friends count and followers count

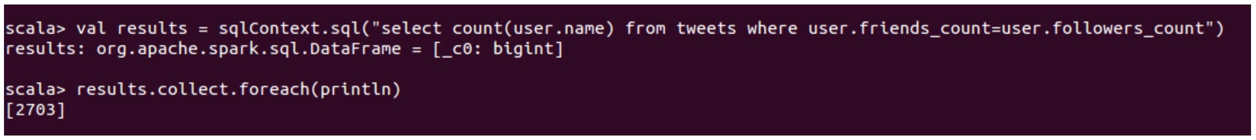
a)
val results = sqlContext.sql("select count(user.name) from tweets where user.friends_count>user.followers_count")
b)
val results = sqlContext.sql("select count(user.name) from tweets where user.friends_count<user.followers_count")
c)
val results = sqlContext.sql("select count(user.name) from tweets where user.friends_count=user.followers_count")



```
scala> val results = sqlContext.sql("select count(user.name) from tweets where user.friends_count>user.followers_count")
results: org.apache.spark.sql.DataFrame = [_c0: bigint]

scala> results.collect.foreach(println)
[237800]

scala> results.save("/home/tejuuganne/Desktop/PBOutput/","json")
```



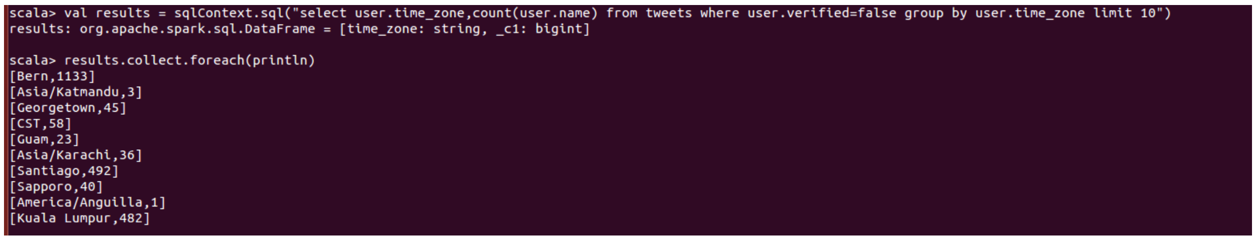
```
scala> val results = sqlContext.sql("select count(user.name) from tweets where user.friends_count<user.followers_count")
results: org.apache.spark.sql.DataFrame = [_c0: bigint]

scala> results.collect.foreach(println)
[217304]

scala> results.save("/home/tejuuganne/Desktop/PBOutput2_1/","json")
```



```
scala> val results = sqlContext.sql("select count(user.name) from tweets where user.friends_count=user.followers_count")
results: org.apache.spark.sql.DataFrame = [_c0: bigint]

scala> results.collect.foreach(println)
[2703]
```

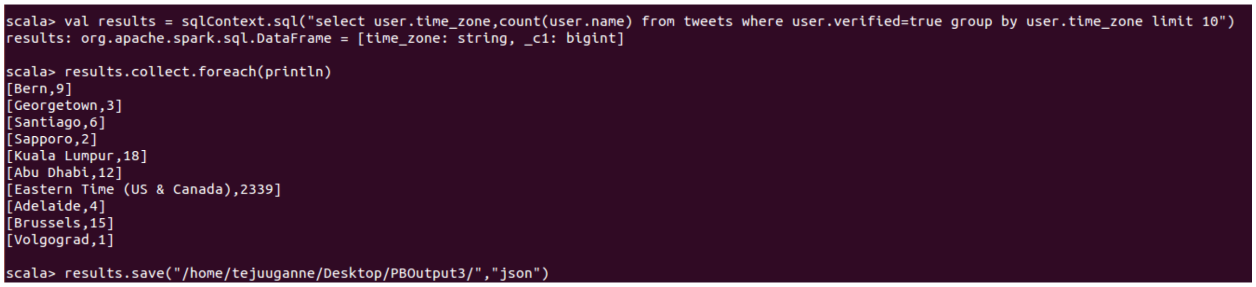
3. This query gives the count of verified users in different time zones

a)
val results = sqlContext.sql("select user.time_zone,count(user.name) from tweets where user.verified=false group by user.time_zone limit 10")



```
scala> val results = sqlContext.sql("select user.time_zone,count(user.name) from tweets where user.verified=false group by user.time_zone limit 10")
results: org.apache.spark.sql.DataFrame = [time_zone: string, _c1: bigint]

scala> results.collect.foreach(println)
[Bern,1133]
[Asia/Katmandu,3]
[Georgetown,45]
[CST,58]
[Guam,23]
[Asia/Karachi,36]
[Santiago,492]
[Sapporo,40]
[America/Anguilla,1]
[Kuala Lumpur,482]
```

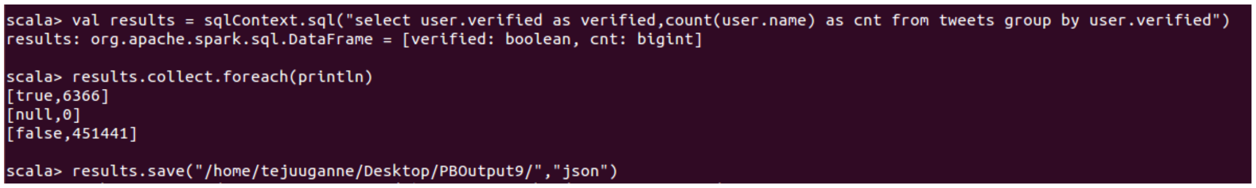
b)
val results = sqlContext.sql("select user.time_zone,count(user.name) from tweets where user.verified=true group by user.time_zone limit 10")



```
scala> val results = sqlContext.sql("select user.time_zone,count(user.name) from tweets where user.verified=true group by user.time_zone limit 10")
results: org.apache.spark.sql.DataFrame = [time_zone: string, _c1: bigint]

scala> results.collect.foreach(println)
[Bern,9]
[Georgetown,3]
[Santiago,6]
[Sapporo,2]
[Kuala Lumpur,18]
[Abu Dhabi,12]
[Eastern Time (US & Canada),2339]
[Adelaide,4]
[Brussels,15]
[Volgograd,1]

scala> results.save("/home/tejuuganne/Desktop/PBOutput3/","json")
```

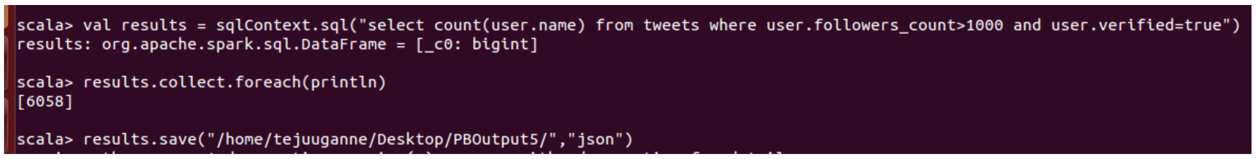
4. This query gives the number of verified users

val results = sqlContext.sql("select user.verified as verified,count(user.name) as cnt from tweets group by user.verified")



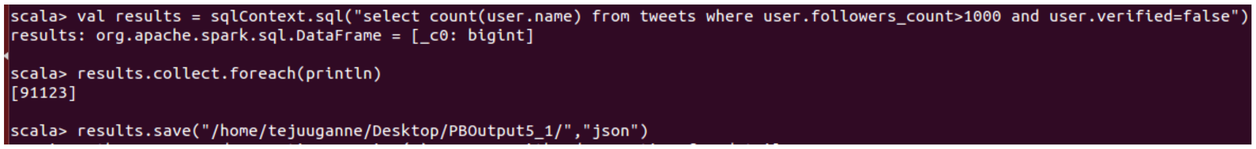
```
scala> val results = sqlContext.sql("select user.verified as verified,count(user.name) as cnt from tweets group by user.verified")
results: org.apache.spark.sql.DataFrame = [verified: boolean, cnt: bigint]

scala> results.collect.foreach(println)
[true,6366]
[null,0]
[false,451441]

scala> results.save("/home/tejuuganne/Desktop/PBOutput9/","json")
```

5. This query gives the count of number of verified and non verified users for the followers count>1000
a)
val results = sqlContext.sql("select count(user.name) from tweets where user.followers_count>1000 and user.verified=true")



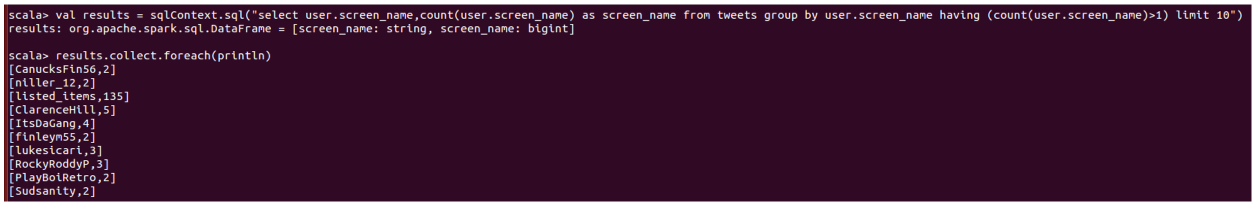
```
scala> val results = sqlContext.sql("select count(user.name) from tweets where user.followers_count>1000 and user.verified=true")
results: org.apache.spark.sql.DataFrame = [_c0: bigint]

scala> results.collect.foreach(println)
[6058]

scala> results.save("/home/tejuuganne/Desktop/PBOutput5/","json")
```

b)
val results = sqlContext.sql("select count(user.name) from tweets where user.followers_count>1000 and user.verified=false")



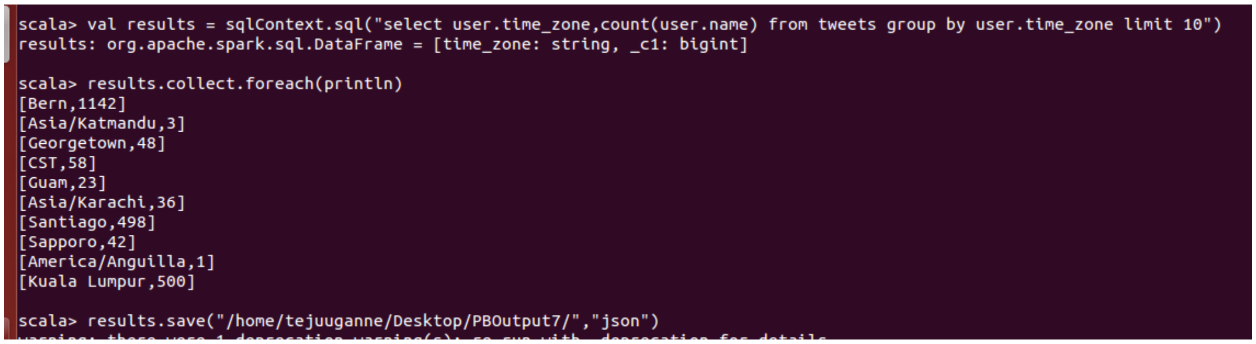
```
scala> val results = sqlContext.sql("select count(user.name) from tweets where user.followers_count>1000 and user.verified=false")
results: org.apache.spark.sql.DataFrame = [_c0: bigint]

scala> results.collect.foreach(println)
[91123]

scala> results.save("/home/tejuuganne/Desktop/PBOutput5_1/","json")
```

6. This query gives the count of number of users with similar screen name.
val results = sqlContext.sql("select user.screen_name,count(user.screen_name) from tweets group by user.screen_name having (count(user.screen_name)>1) limit 10")



```
scala> val results = sqlContext.sql("select user.screen_name,count(user.screen_name) as screen_name from tweets group by user.screen_name having (count(user.screen_name)>1) limit 10")
results: org.apache.spark.sql.DataFrame = [screen_name: string, screen_name: bigint]

scala> results.collect.foreach(println)
[CanucksFin56,2]
[niller_12,2]
[listed_items,135]
[ClarenceHill,5]
[ItsDaGang,4]
[finleym55,2]
[lukesicari,3]
[RockyRoddyP,3]
[PlayBoiRetro,2]
[Sudsanity,2]
```

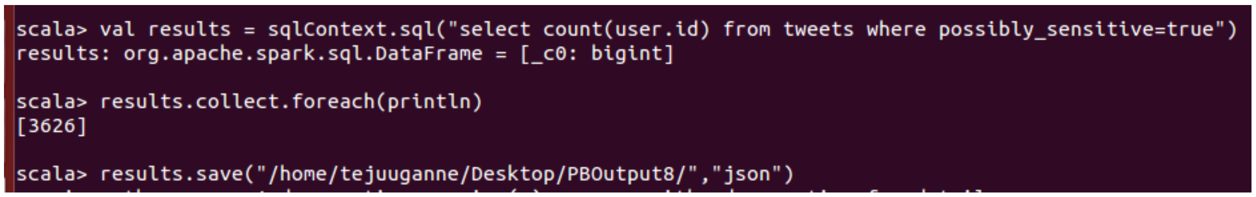
7. This query gives the count of maximum number of users in different countries
val results = sqlContext.sql("select user.time_zone,count(user.name) from tweets group by user.time_zone limit 10")



```
scala> val results = sqlContext.sql("select user.time_zone,count(user.name) from tweets group by user.time_zone limit 10")
results: org.apache.spark.sql.DataFrame = [time_zone: string, _c1: bigint]

scala> results.collect.foreach(println)
[Bern,1142]
[Asia/Katmandu,3]
[Georgetown,48]
[CST,58]
[Guam,23]
[Asia/Karachi,36]
[Santiago,498]
[Sapporo,42]
[America/Anguilla,1]
[Kuala Lumpur,500]

scala> results.save("/home/tejuuganne/Desktop/PBOutput7/","json")
```

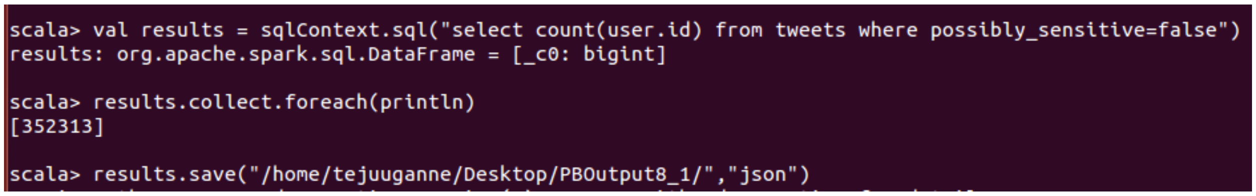
8. This query gives the count of users with possibly sensitivity where the valuses were true and false.

a)val results = sqlContext.sql("select count(user.id) from tweets where possibly_sensitive=true")



```
scala> val results = sqlContext.sql("select count(user.id) from tweets where possibly_sensitive=true")
results: org.apache.spark.sql.DataFrame = [_c0: bigint]

scala> results.collect.foreach(println)
[3626]

scala> results.save("/home/tejuuganne/Desktop/PBOutput8/","json")
```

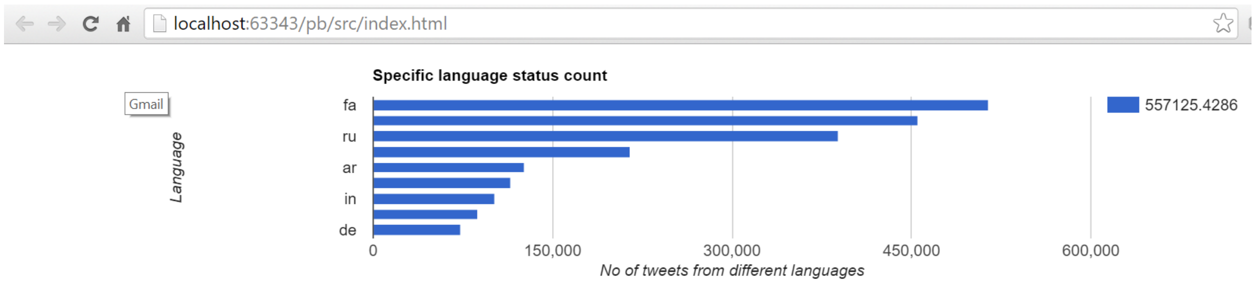
b)val results = sqlContext.sql("select count(user.id) from tweets where possibly_sensitive=false")



```
scala> val results = sqlContext.sql("select count(user.id) from tweets where possibly_sensitive=false")
results: org.apache.spark.sql.DataFrame = [_c0: bigint]

scala> results.collect.foreach(println)
[352313]

scala> results.save("/home/tejuuganne/Desktop/PBOutput8_1/","json")
```

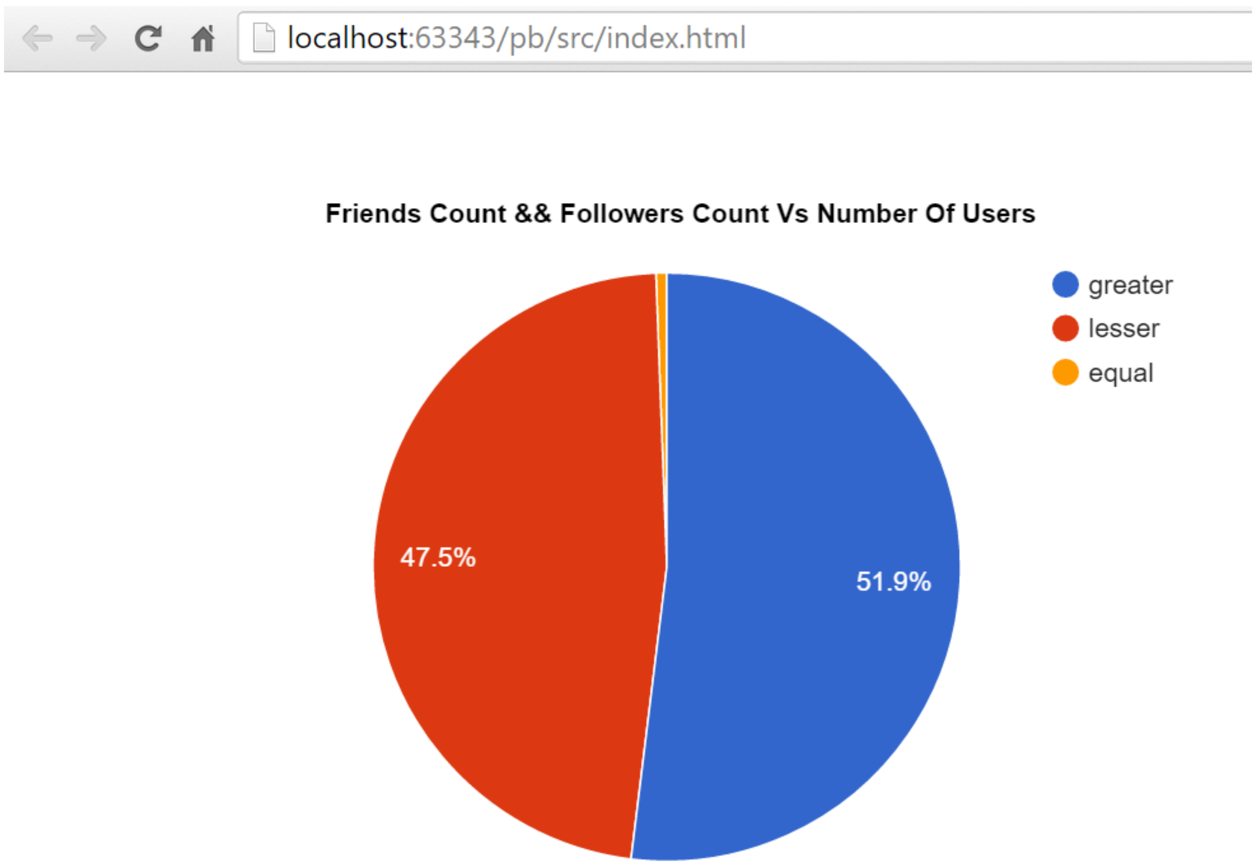
Visualization

1.

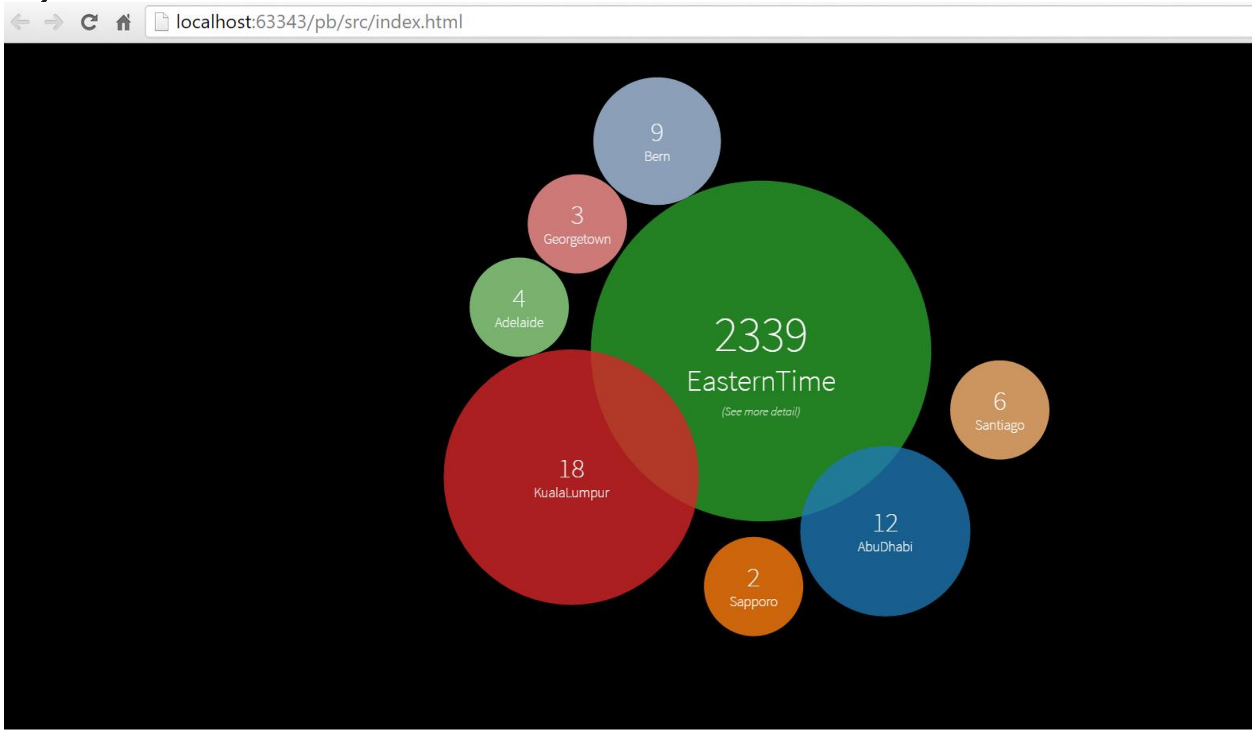


localhost:63343/pb/src/index.html

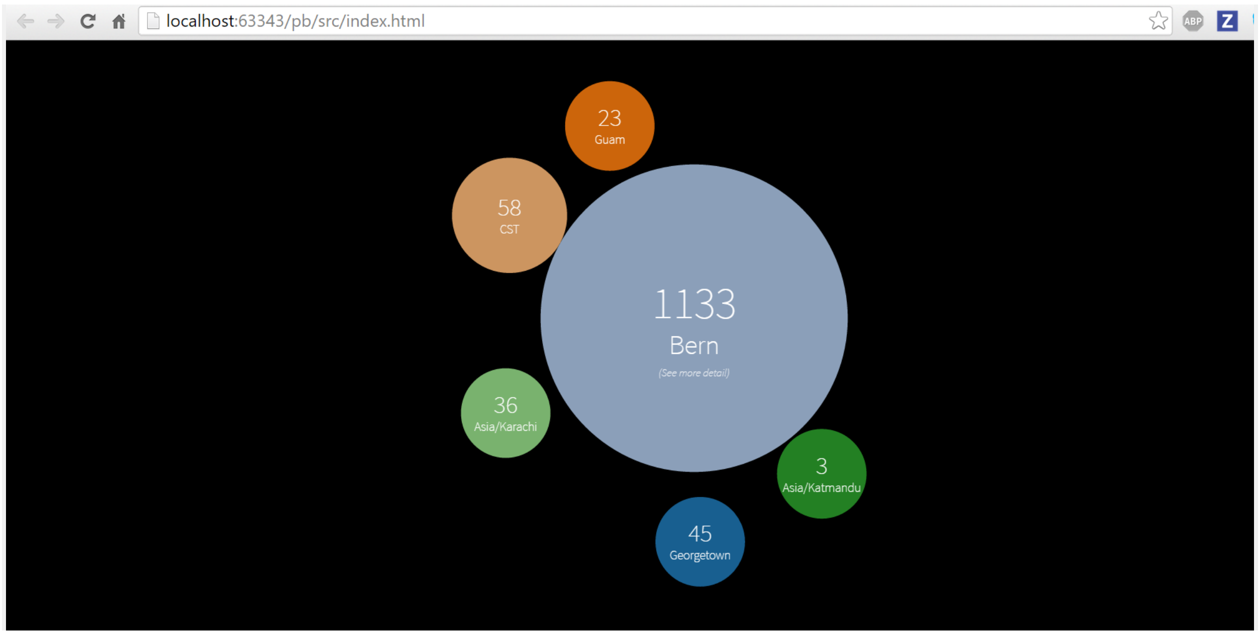
2


localhost:63343/pb/src/index.html
Friends Count && Followers Count Vs Number Of Users
greater
lesser
equal
47.5%
51.9%

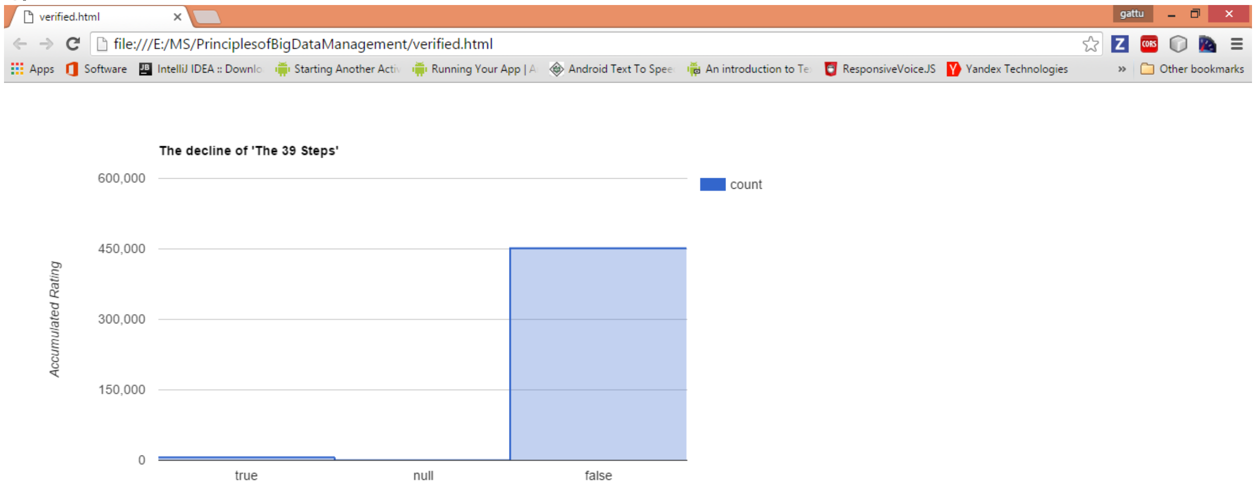
3a)


localhost:63343/pb/src/index.html
9
Bern
3
Georgetown
4
Adelaide
2339
EasternTime
(See more detail)
6
Santiago
18
KualaLumpur
12
AbuDhabi
2
Sapporo

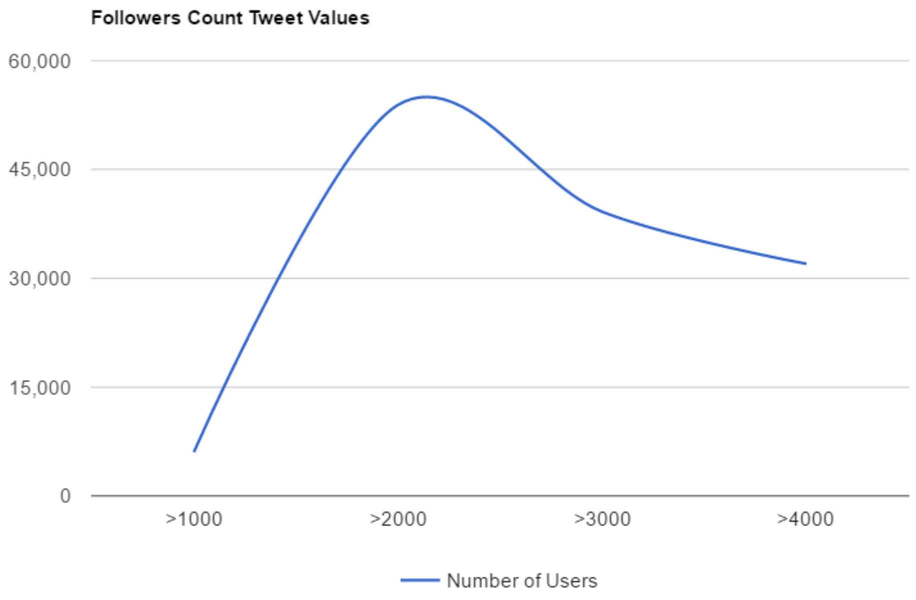
3b)


localhost:63343/pb/src/index.html
23
Guam
58
CST
1133
Bern
(See more detail)
36
Asia/Karachi
3
Asia/Katmandu
45
Georgetown

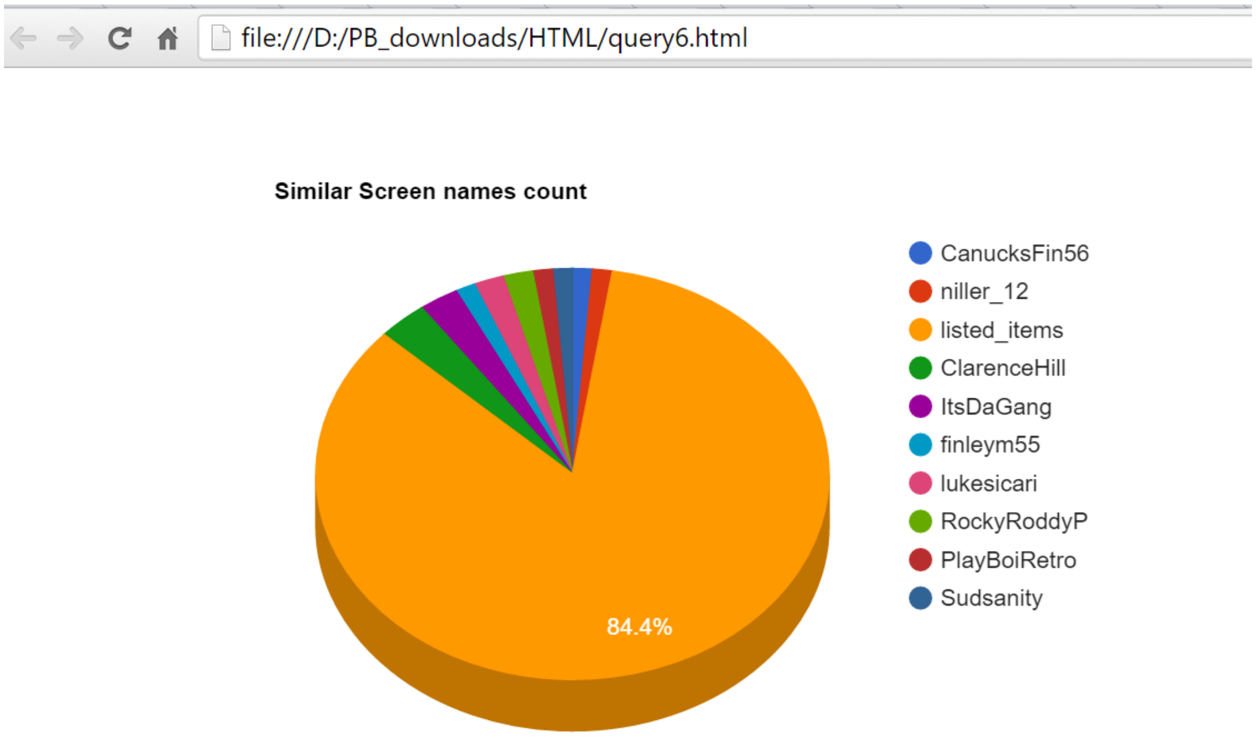
4.


verified.html
file:///E:/MS/PrinciplesofBigDataManagement/verified.html
Apps
Software
IntelliJ IDEA :: Downlo
Starting Another Acti
Running Your App | A
Android Text To Spee
An introduction to Te
ResponsiveVoiceJS
Yandex Technologies
Other bookmarks
The decline of 'The 39 Steps'
count
Accumulated Rating
600,000
450,000
300,000
150,000
0
true
null
false

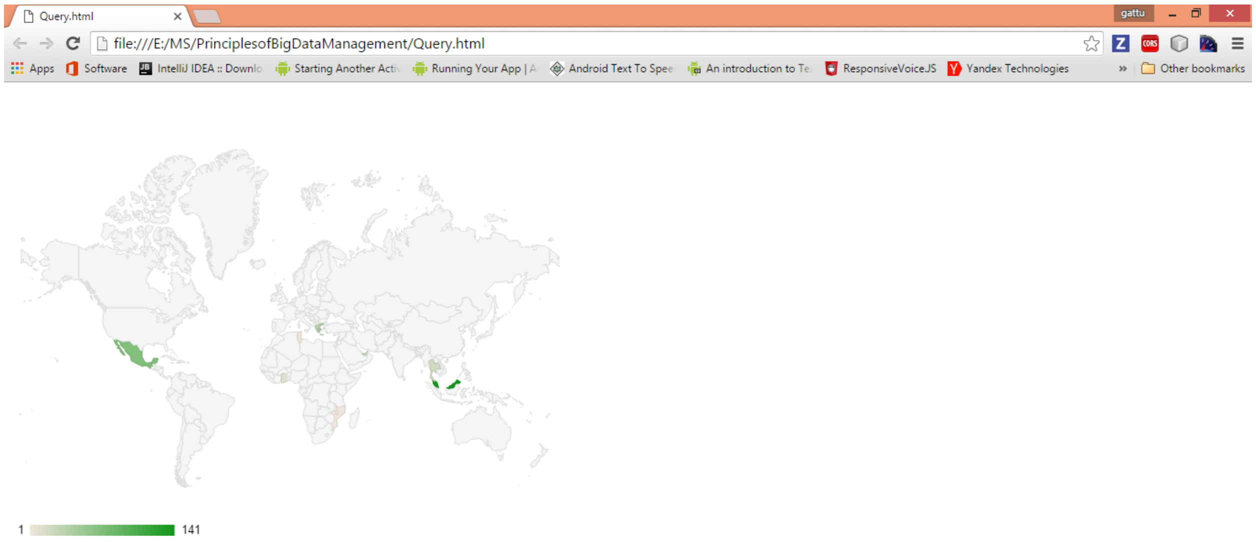
5.


Followers Count Tweet Values
60,000
45,000
30,000
15,000
0
>1000
>2000
>3000
>4000
Number of Users

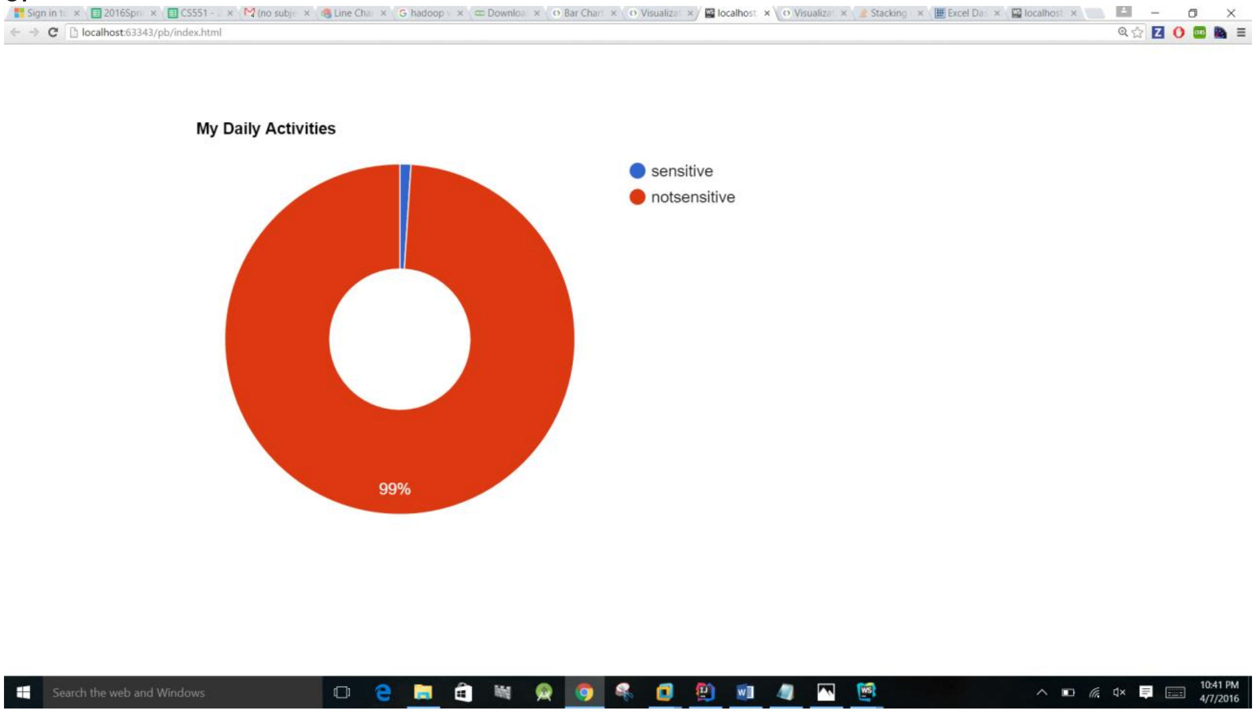
6.


file:///D:/PB_downloads/HTML/query6.html
Similar Screen names count
84.4%
CanucksFin56
niller_12
listed_items
ClarenceHill
ItsDaGang
finleym55
lukesicari
RockyRoddyP
PlayBoiRetro
Sudsanity

7.


Query.html
gattu
file:///E:/MS/PrinciplesofBigDataManagement/Query.html
Apps
Software
IntelliJ IDEA :: Downlo
Starting Another Acti
Running Your App | A
Android Text To Spee
An introduction to Te
ResponsiveVoice.JS
Yandex Technologies
Other bookmarks
1
141

8.



Github link for the source and destination

https://github.com/MudduLatha/PBPhase-2

References:

https://developer.ibm.com/clouddataservices/docs/spark/tutorials-and-samples/build-sql-queries/

http://marcobonzanini.com/2015/03/02/mining-twitter-data-with-python-part-1/

https://d3js.org/

https://developers.google.com/chart/