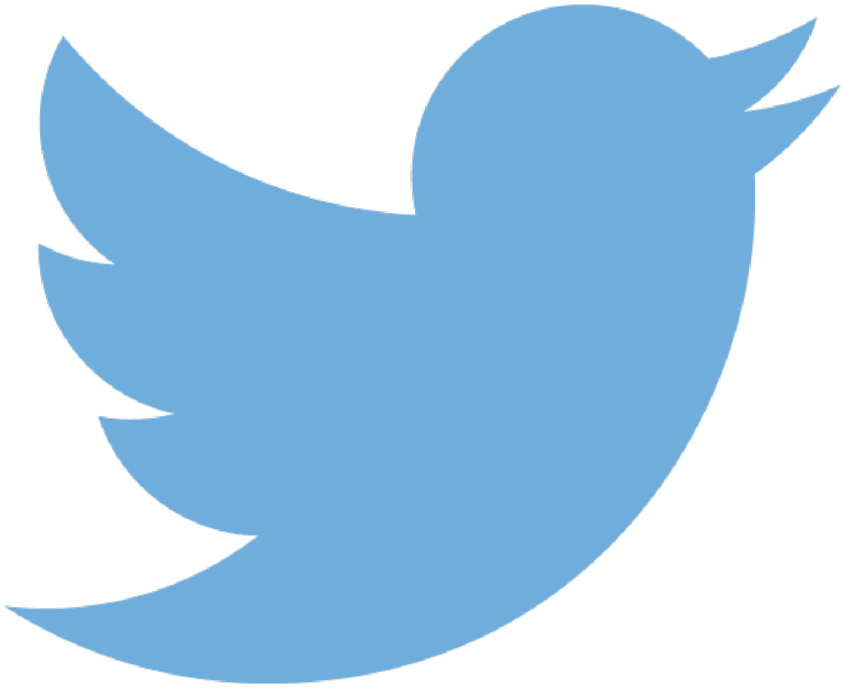
# Principles of Big Data Management

## Project 3




Team

1. Saketh Garuda (sg7kf)
2. Mudunuri Sri Sai Sarat Chandra Varma (smyx4)
3. Yalamanchili Sowmya (syb7c)
4. Nandanamudi Sreelakshmi (snhnc)

# 1.Introduction

The main objective of this project is analyzing the twitter data about ‘Politics’ category and analyze the data using Apace Spark. We choose ‘Politics’ as our topic to do big data analysis. Based on twitter tweets, we predicted some interesting analysis on Politics using thousands of tweets tweeted by different people around the globe. First we collected the tweets from twitter API based on the category ‘Politics’. After that, we analyzed the data that we have collected. By using the analysis, we written some interesting SQL queries useful to give a proper result for the analysis.

Here we used Spark to processing the twitter data. Because it has many advantages like

- Speed: Run programs up t0 100 times faster than Hadoop Map Reduce in memory.
- Generality: Performing complex analytics, streaming and combines SQL.
- Platform Independent: Spark runs on Hadoop, standalone or in a cloud environment.
- Ease of Use: Able to write applications in Java, Scala, Python, R.

The entire document gives the walk through and scope of the working environment of the project.

# 2.System Requirements

### Software Requirements:

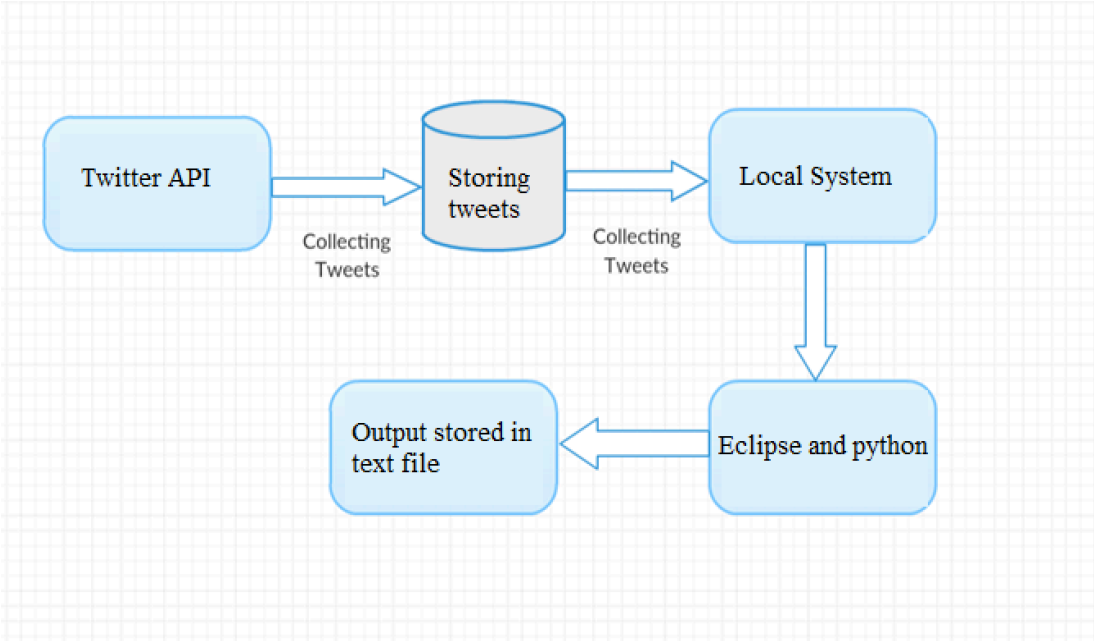
- Python 3.5
- Net Beans
- Apache Spark
- JDK 1.8
- Intellij IDEA
- Scala 2.11.8

### Programming Languages:

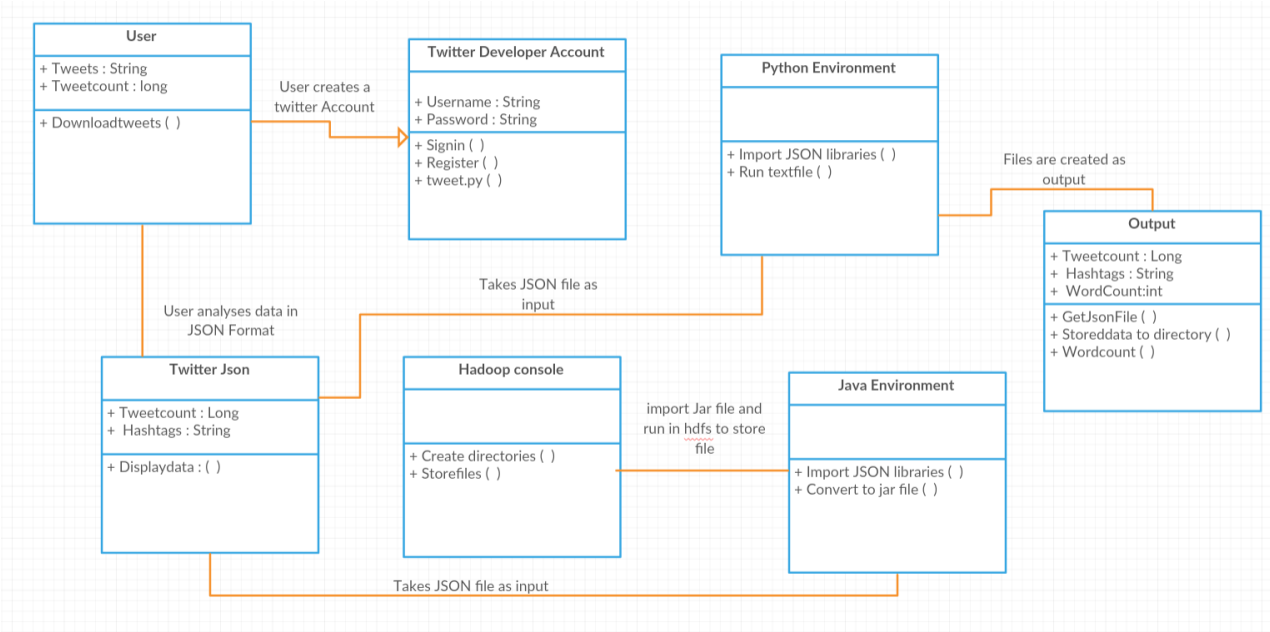
- Java
- Python
- Scala
- SQL
- HTML
- JSP

# 3.Software Architecture and Design

- Twitter tweets are extracted using the API by running python program using the twitter tokens. The collected tweets are stored in JSON format in the local system.



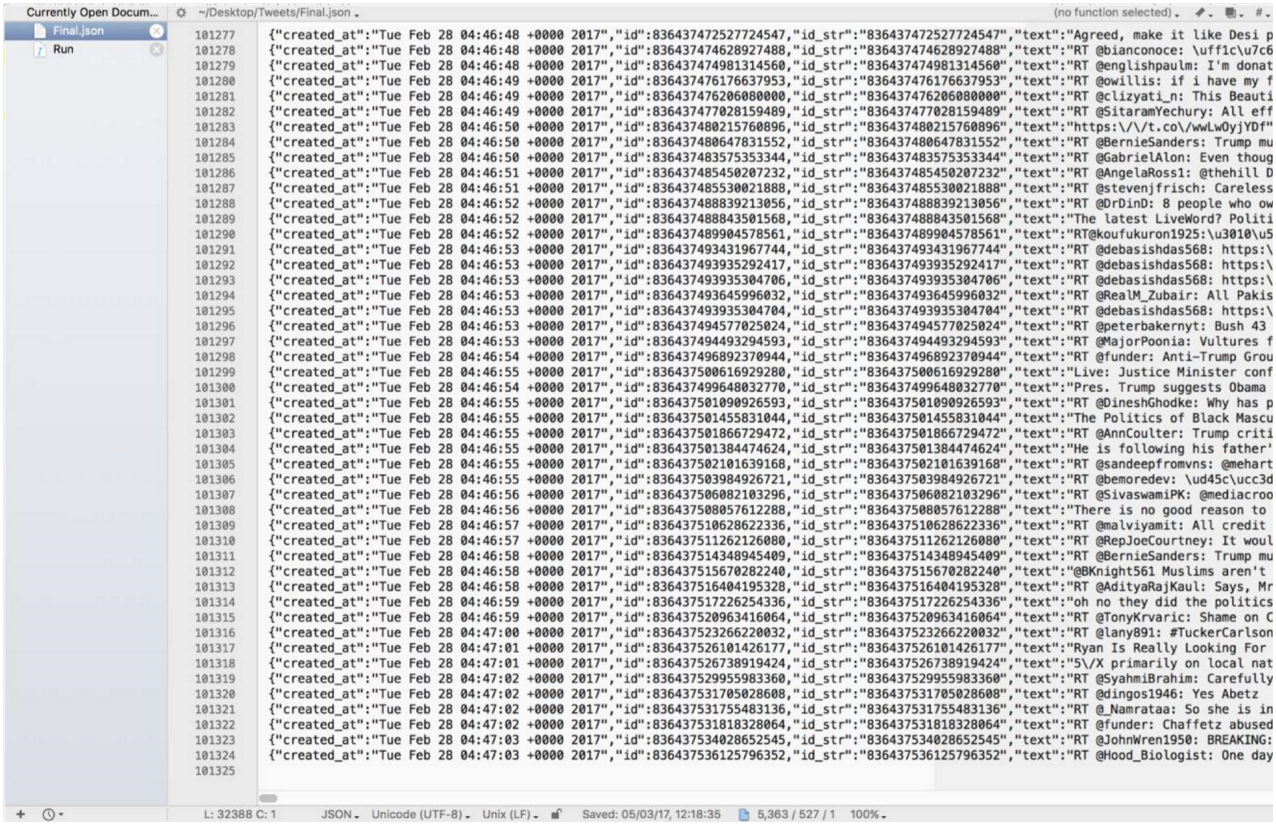

- Here is the class diagram for the project



# 4.Main Requirements

## Collect tweets in JavaScript Object Notation

- We've collected 100k+ tweets related to category 'Politics' into a JSON file using the twitter API. JSON file is used for future work in retrieving the text and top 10 best times from the entire list of tweets. Here is the screenshot of JSON file,




After collecting the 100k tweets, we've written a python code for separating text from tweets file and it is stored in tweet text file.

- Below screenshot indicates the python code for retrieving text from the tweets file. We have given the input **"Tweets.txt"** file and parsed the JSON data with **"tweet['text']"** and stored it an output file.

TweetText.py - C:\Users\sreel\Desktop\PBProject2\Main Requirements\TweetText.py (3.6.0)

File Edit Format Run Options Window Help

```
# Import the necessary package to process data in JSON format
try:
    import json
except ImportError:
    import simplejson as json

# We use the file saved from last step as example
tweets_filename = 'Tweets.txt'
tweets_file = open(tweets_filename, "r")

for line in tweets_file:
    try:
        # Read in one line of the file, convert it into a json object
        tweet = json.loads(line.strip())
        if 'text' in tweet: # only messages contains 'text' field is a tweet

            print (tweet['text']) # content of the tweet

    except:
        # read in a line is not in JSON format (sometimes error occured)
        continue
```

This is the input tweets file

This is the criteria to retrieve the content of the tweet

Ln: 1 Col: 1

- Here is the screenshot of output TweetsText file

tweettext - Notepad

File Edit Format View Help

```
RT @Del_ivered: John Boothman, Johnny-come-lately to @theSNP has form in bullying claims,so ideal for SNP,his politics have changed. https:…
"I'm sorry, I don't want to see him"-father of #NavySeal killed in #Yemen re: Trump as son's casket arrived in U.S.… https://t.co/Ad2IA9tDCE
@HackettTerence @gregsteube @NRA https://t.co/3AAXMWSL8w
RT @mikiebarb: "He is shocked that he is not in control of the press." Must read from @GlennThrush &amp;amp; @grynbaum https://t.co/NoNtDp7UFj
RT @ALT_DOJ: Everyone keep an eye on @ossoff if you live in GA get out and help if you can! https://t.co/AGltnfMxn3
RT @iSupportPTI: @DunyaNews @betterpakistan @WajSKhan Panama Money Laundring Network telling Imran Khan how to do politics.We are sick of S…
RT @MarcusC22973194: BREAKING AND URGENT
JASON CHAFFETZ NOT ONLY TAKES RUSSIAN MONEY, HE IS ALSO LINKED TO TRUMP THROUGH CHINA…
RT @KeithOlbermann: This Senator has been an amoral jackass from day one. https://t.co/q6Cqo2oAxx
RT @camboviet: Chelsea Clinton Attends ‘I am Muslim Too’ Protest in NYC https://t.co/aiiy2Txnyk
RT @ShaunKing: Will Republicans investigate the failed Yemen raid like they did Benghazi?

Family of slain SEAL is demanding it.

https://…
RT @ShaunKing: Will Republicans investigate the failed Yemen raid like they did Benghazi?

Family of slain SEAL is demanding it.

https://…
RT @funder: Chaffetz abused his power to try &amp;amp; reverse a law–in WASHINGTON DC. HE NEEDS TO RESIGN.

#russiagate #trumprussia #rt https://t.…
RT @yashar: Trump now has his own version of a Benghazi mom. The father of the Navy Seal killed in Yemen wants answers.…
@JoeNBC Let's see Sanders campaign emails. Politics is unfair &amp;amp; not for the faint hearted-Sanders did nada in 36 years to form 3rd party
Farage SLAMS May for not showing support to Le Pen https://t.co/doz43ueIc2
RT @ShaunKing: 1. William Owens was so disgusted w/ Trump's Yemen raid that killed his son, Ryan, that he refused to meet w/ Trump

https:/…
RT @ZekeJMiller: Slain SEAL’s dad wants answers: ‘Don’t hide behind my son’s death’ https://t.co/wFEHNfZagR
RT @DoctorRobin: Donald Trump advisor Sheriff David A Clarke was in Russia while Mike Flynn dined with Putin https://t.co/33eDtmA3y7 #Trump…
RT @tongs_ya_bass: The most divisive part of Scottish politics is when English politicians travel north to tell us we're divisive! #SadiqKh…
RT @jaketapper: Father of SEAL killed in Yemen raid spurned meeting with Trump, wants answers | Miami Herald https://t.co/7mmbWfmu1d
Slain SEAL’s dad wants answers: ‘Don’t hide behind my son’s death’ https://t.co/oQbBc5PAEt
https://t.co/dlDICcQwCK
RT @europa_nexus: Hedge-fund billionaire and Donald Trump backer 'played key role in Brexit campaign' https://t.co/Q5PfMoSNNW
How fascist facebook account data helped the right wing know they would win the election:

https://t.co/hfsJEZbzBP
RT @SenCortezMasto: 21 female U.S. senators is not enough. Women need more seats at the table. Period.

My interview w/ @ELLEmagazine: ht…
RT @KeithOlbermann: This Senator has been an amoral jackass from day one. https://t.co/q6Cqo2oAxx
```

# Analyzing Twitter Data

## Query 1: Most Popular Users

In this query, we are fetching the popular users from the tweets by combining the data from two table i.e. users with more followers and users with more tweets. With the help of the data frames, we are able to fetch the records in sequential manner.

**Query code:**

```
val join= sqlContext.sql("SELECT NT.u_sn AS N, FC.ft FROM NT " +
  "JOIN FC ON (NT.u_sn = FC.ust) GROUP BY " +
  "NT.u_sn,FC.ft ORDER BY FC.ft DESC")
//join.show()
join.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("src/main/resources/MostPopUsers")
```

**Output:**

part-00000 | part-00000

```
1  (Politics4All,4384)
2  (airiters,1755)
3  (airiters,1754)
4  (sharebossindia,713)
5  (USGovReport,339)
6  (USGovReport,338)
7  (More_BRB_Songs,26)
8
```

## Query 2: Analyzing Retweet count based on the Location of the User

In this query, we used data frames and Spark SQL for retrieving the data such as user screen name and his tweets corresponding Retweet count based on the location of the user. And from the results we have analyzed that users from the location USA has more popular tweets as their tweets recount is higher.

**Query code:**

```
index{
  // Data frame for Retweet count for username and language filter (main)

  val re_tweet_query = sqlContext.sql("select user.name as name, retweeted_status.retweet_count " +
    "as cnt, user.location as location from " +
    "zone_refiner where user.name is not NULL and " +
    "user.location like '%,%' and " +
    "user.location not like '%1%' order by cnt desc limit 11")

  //re_tweet_query.show()
  re_tweet_query.save("location_ret2","json")
}
```

**Output:**

part-00000


```
{"name":"Mike Smart","cnt":133389,"location":"Plymouth, Minnesota, USA"}
{"name":"Brandon Paquette","cnt":133381,"location":"Livonia, MI"}
{"name":"Christopher Hewitt","cnt":133380,"location":"Los Angeles, CA"}
{"name":"Sheandearmas","cnt":133379,"location":"Brooklyn, NY"}
{"name":"Danny Sweets","cnt":133378,"location":"California, USA"}
{"name":"Kurt Cocaine","cnt":133376,"location":"Fresno, CA"}
{"name":"ðŸŒŸtiff ðŸ»","cnt":133373,"location":"Texas, USA"}
{"name":"Matt Fegan","cnt":133372,"location":"Denver, CO"}
{"name":"Sumi","cnt":133371,"location":"New York, NY"}
{"name":".arcticbreath","cnt":92641,"location":"Colorado Springs, CO"}
```


## Query 3: Analyzing location specific device usage

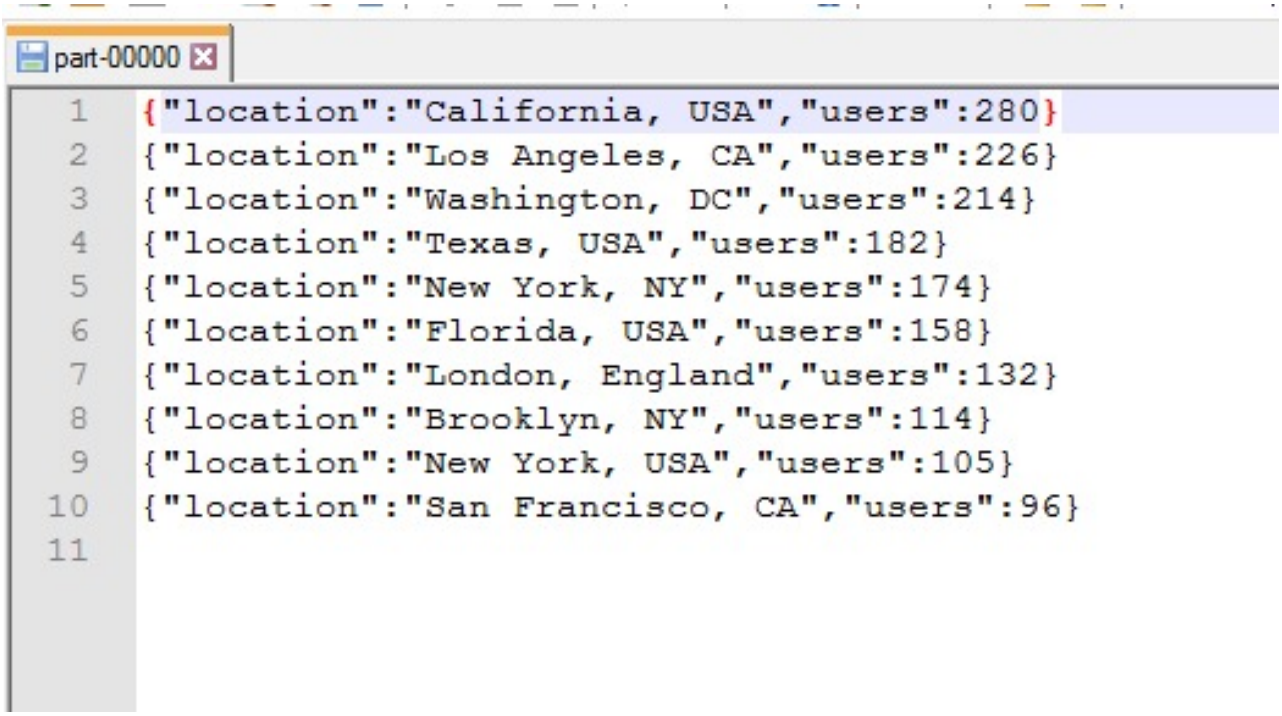
By using data frames we are able to categorize the tweets posted by the user with respective to the device. This query also analyzes the most used device based on the location.

**Query code:**

```
val trendsFile = sqlContext.jsonFile("C:\\Users\\User\\Desktop\\Folders\\pbproject3\\Final.json")
trendsFile.registerTempTable("urltrends")

index {
  val trends_query = sqlContext.sql("select user.location,count(*) as users from urltrends where " +
    "user.location like '%,%' and user.location not like '%1%' " +
    "and source like '%Twitter for iPad%' group by user.location order by users desc limit 10")
  //trends_query.show()
  trends_query.save("tipad_file", "json")
}
```

**Android devices Output:**


**iPhone devices Output:**



**Web Client devices Output:**

part-00000 part-00000

```
1  {"location":"Washington, DC","users":173}
2  {"location":"California, USA","users":135}
3  {"location":"New York, NY","users":131}
4  {"location":"London, England","users":115}
5  {"location":"Los Angeles, CA","users":111}
6  {"location":"Florida, USA","users":95}
7  {"location":"Chicago, IL","users":68}
8  {"location":"Texas, USA","users":65}
9  {"location":"New York, USA","users":62}
10 {"location":"San Francisco, CA","users":61}
11
```

## Query 4: Most Popular Hashtags

We are able to populate the most popular hashtags from the tweets collection by using data frames.

**Query code:**

```
val Hash_Tags=sqlContext.sql("SELECT HashTags,count(*) AS ht_count FROM " +
  "Table21 GROUP BY HashTags ORDER BY ht_count DESC LIMIT 10")
```

**Output:**

part-00000 part-00000

```
1  (politics,1859)
2  (resist,1095)
3  (Politics,1071)
4  (TrumpRussia,958)
5  (Trump,903)
6  (Oscars,817)
7  (trumprussia,559)
8  (auspol,495)
9  (russiagate,483)
10 (Brexit,418)
11
```

## Query 5: Analyzing number of tweets posted on particular day based on region

In this query, we used RDD transformation's and action's to analyze most popular day where more number of tweets are posted i.e. best day to post the tweet based on location.

**Query code:**

```
index {
  // RDD for different politics analysis

  val Sunday= (textFile.filter(line => line.contains("Sun")).count())
  val Monday= (textFile.filter(line => line.contains("Mon")).count())
  val Tuesday= (textFile.filter(line => line.contains("Tue")).count())
  val Wednesday= (textFile.filter(line => line.contains("Wed")).count())
  val Thursday= (textFile.filter(line => line.contains("Thu")).count())
  val Friday= (textFile.filter(line => line.contains("Fri")).count())
  val Saturday= (textFile.filter(line => line.contains("Sat")).count())

  println(("UK Region: \n" + "Number of tweets posted " +
    "on Sunday : %s \n Number of tweets posted on Monday : %s \n " + "Number of tweets " +
    "posted on Tuesday : %s \n Number of tweets posted on Wednesday : %s \n " +
    "Number of tweets posted on Thursday : %s \n Number of tweets " +
    "posted on Friday : %s \n Number of tweets " +
    "posted on Saturday : %s").format(Sunday,Monday,Tuesday,
    Wednesday,Thursday,Friday,Saturday))

}
```

**Output:**

```
17/05/01 17:58:34 INFO DAGScheduler: Job 7 finished: count at
US Region:
Number of tweets posted on Sunday : 1453
 Number of tweets posted on Monday : 1060
 Number of tweets posted on Tuesday : 1052
 Number of tweets posted on Wednesday : 604
 Number of tweets posted on Thursday : 515
 Number of tweets posted on Friday : 697
 Number of tweets posted on Saturday : 677
Execution time : 1.475358961 sec
17/05/01 17:58:34 INFO SparkUI: Stopped Spark web UI at http:
17/05/01 17:58:34 INFO DAGScheduler: Stopping DAGScheduler
17/05/01 17:58:34 INFO MapOutputTrackerMasterEndpoint: MapOut
```

## Query 6: Analyzing hashtags from the given Trends file

In this query, we used RDD transformation's and action's to analyze most hashtag count.

### Query code:

```
index {
  // RDD for different trends analysis

  val le= (textFile.filter(line => line.contains("#LEmissionPolitique")).count())
  val ga= (textFile.filter(line => line.contains("#Gala15GHVIP5")).count())
  val ro= (textFile.filter(line => line.contains("#RockInRio")).count())
  val sc= (textFile.filter(line => line.contains("#Scandal")).count())
  val so= (textFile.filter(line => line.contains("#SignOfTheTimes")).count())
  val ye= (textFile.filter(line => line.contains("#5YearswithEXO")).count())
  val tt= (textFile.filter(line => line.contains("#TimeTravelToThe90sIn4Words")).count())
  val cg= (textFile.filter(line => line.contains("#ChineseGP")).count())
  val dt= (textFile.filter(line => line.contains("#DodgeTheArrowAteneo")).count())
  val uf= (textFile.filter(line => line.contains("#UFC210")).count())

  println(("LEmissionPolitique : %s \n Gala15GHVIP5 : %s \n " + "RockInRio : %s \n Scandal : %s \n " +
    "SignOfTheTimes : %s \n 5YearswithEXO : %s \n " +
    "TimeTravelToThe90sIn4Words : %s"+ "ChineseGP : %s \n " +
    "DodgeTheArrowAteneo : %s \n " + "UFC210 : %s \n").format(le,ga,ro,sc,so,ye,tt,cg,dt,uf))

}
```

### Output:

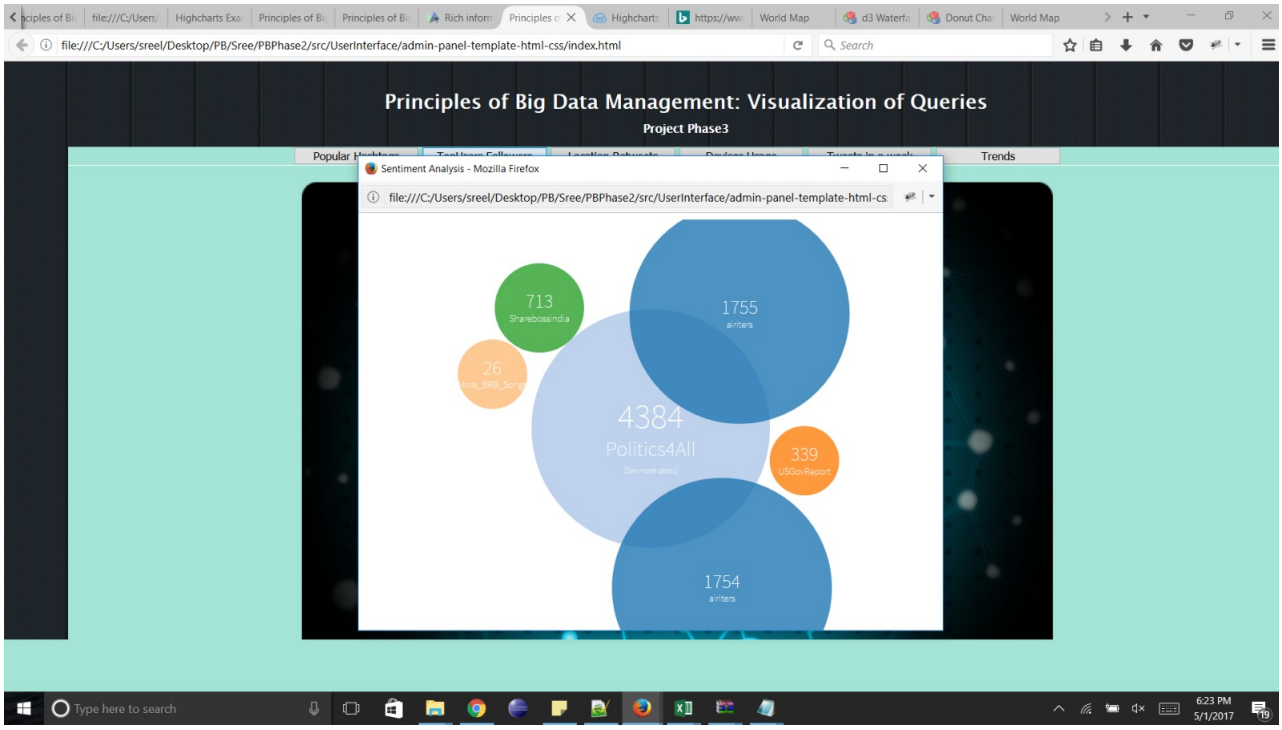

```
index {
  // RDD for different trends analysis

  val le= (textFile.filter(line => line.contains("#LEmissionPolitique")).count())
  val ga= (textFile.filter(line => line.contains("#Gala15GHVIP5")).count())
  val ro= (textFile.filter(line => line.contains("#RockInRio")).count())
  val sc= (textFile.filter(line => line.contains("#Scandal")).count())
  val so= (textFile.filter(line => line.contains("#SignOfTheTimes")).count())
  val ye= (textFile.filter(line => line.contains("#5YearswithEXO")).count())
  val tt= (textFile.filter(line => line.contains("#TimeTravelToThe90sIn4Words")).count())
  val cg= (textFile.filter(line => line.contains("#ChineseGP")).count())
  val dt= (textFile.filter(line => line.contains("#DodgeTheArrowAteneo")).count())
  val uf= (textFile.filter(line => line.contains("#UFC210")).count())

  println(("LEmissionPolitique : %s \n Gala15GHVIP5 : %s \n " + "RockInRio : %s \n Scandal : %s \n " +
    "SignOfTheTimes : %s \n 5YearswithEXO : %s \n " +
    "TimeTravelToThe90sIn4Words : %s"+ "ChineseGP : %s \n " +
    "DodgeTheArrowAteneo : %s \n " + "UFC210 : %s \n").format(le,ga,ro,sc,so,ye,tt,cg,dt,uf))

}
```

# 5.Extra Requirement

As per the extra requirement we have implemented a graphical user interface that enables the user to dynamically execute the analytical tasks. Below are the visualizations for each query that are specified above,

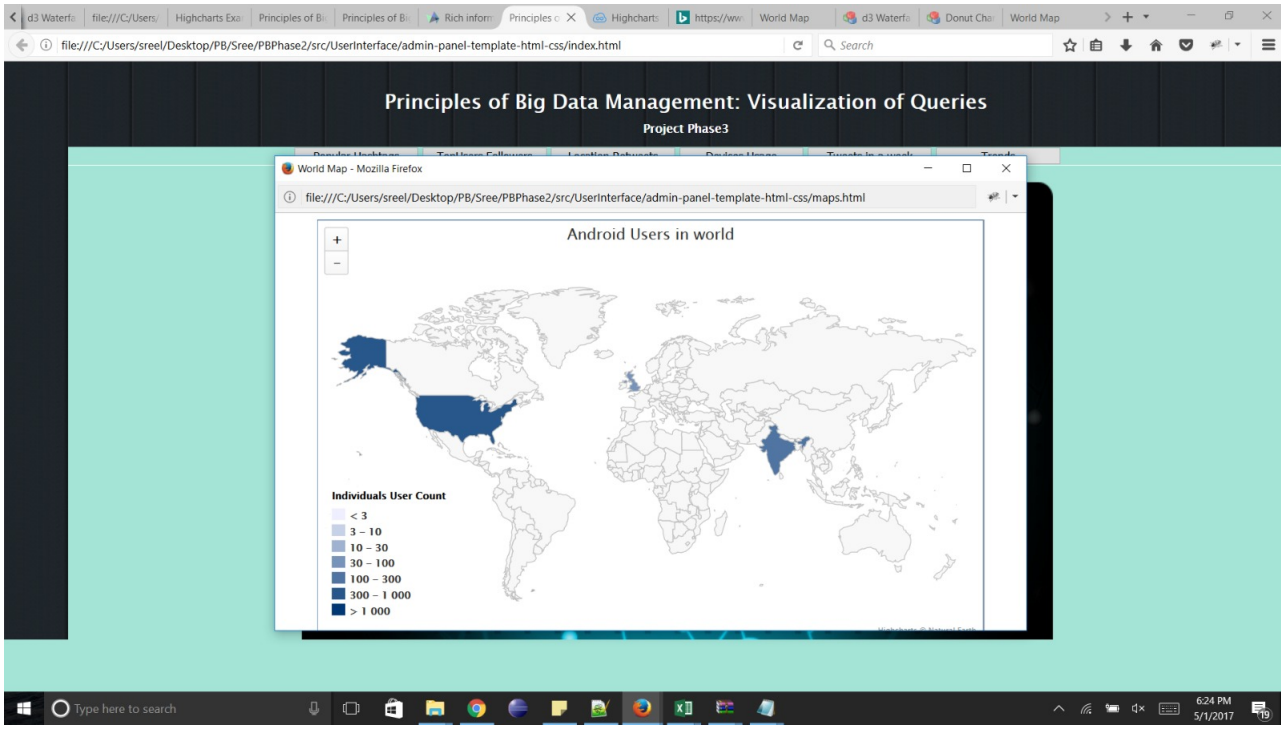
## Homepage Design:

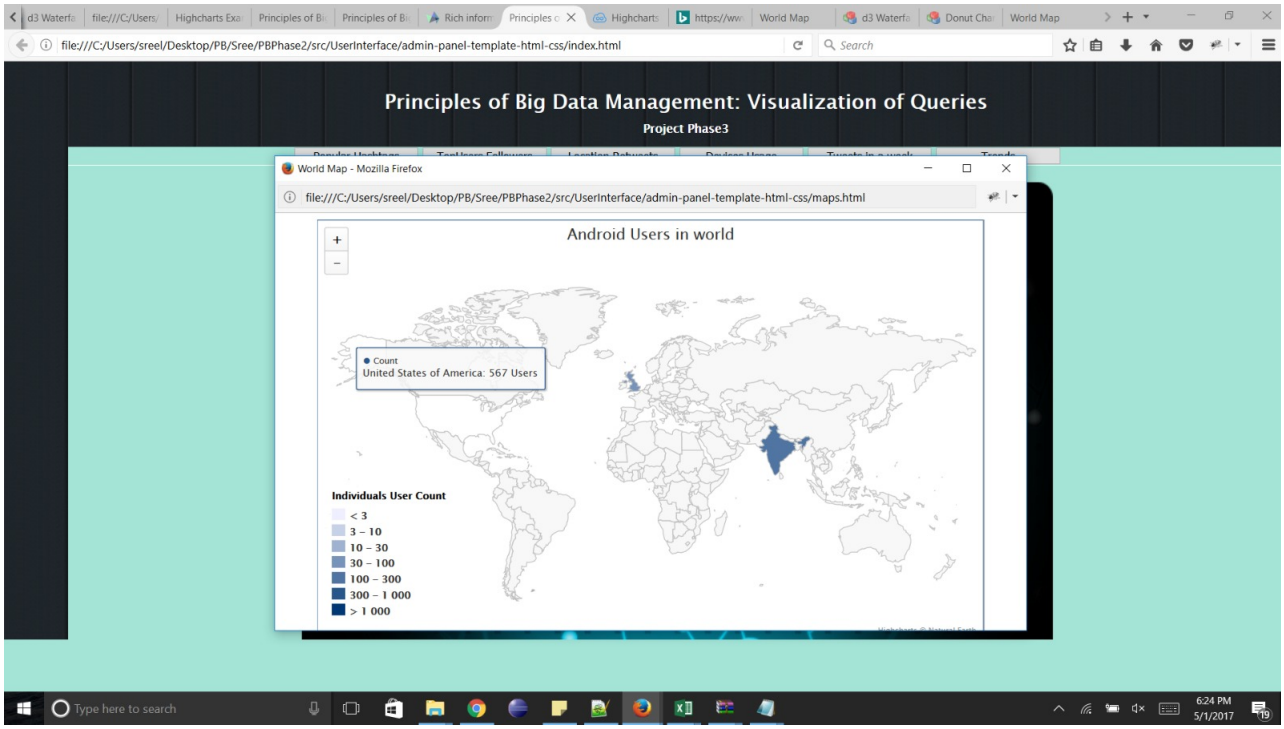

## Query 1:

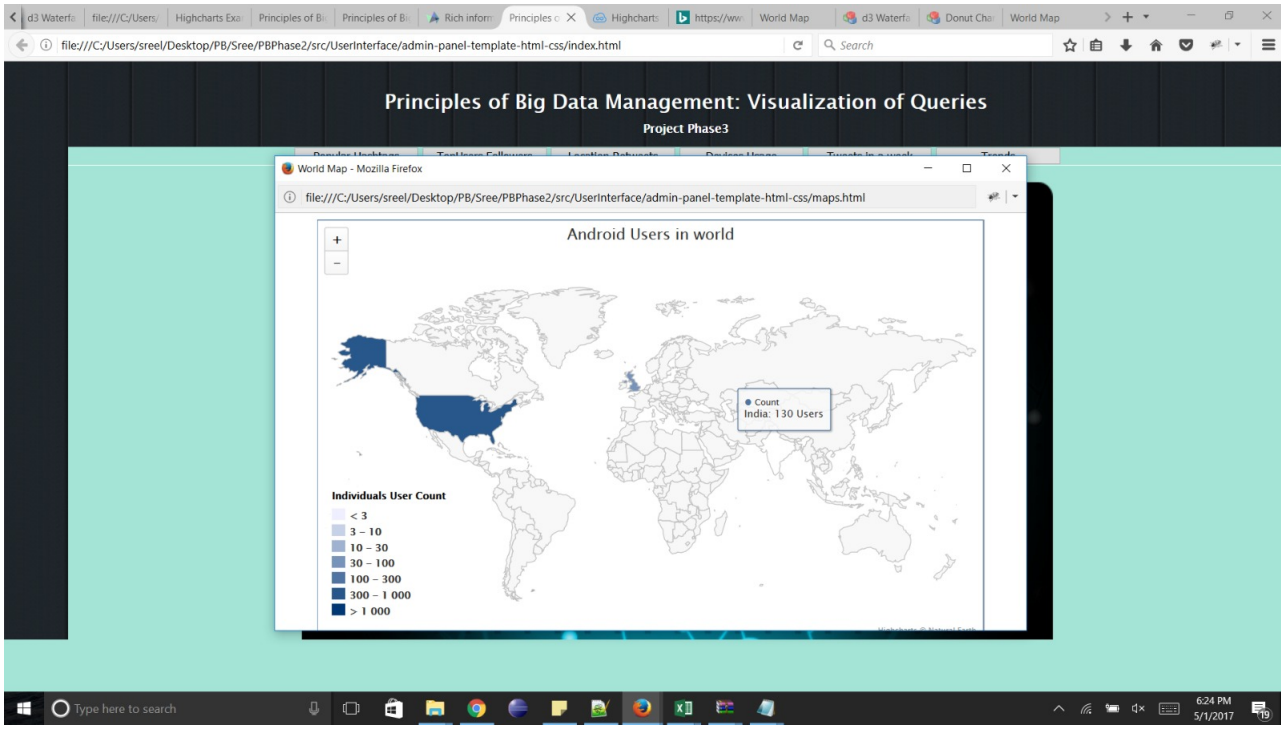


# Query 2:

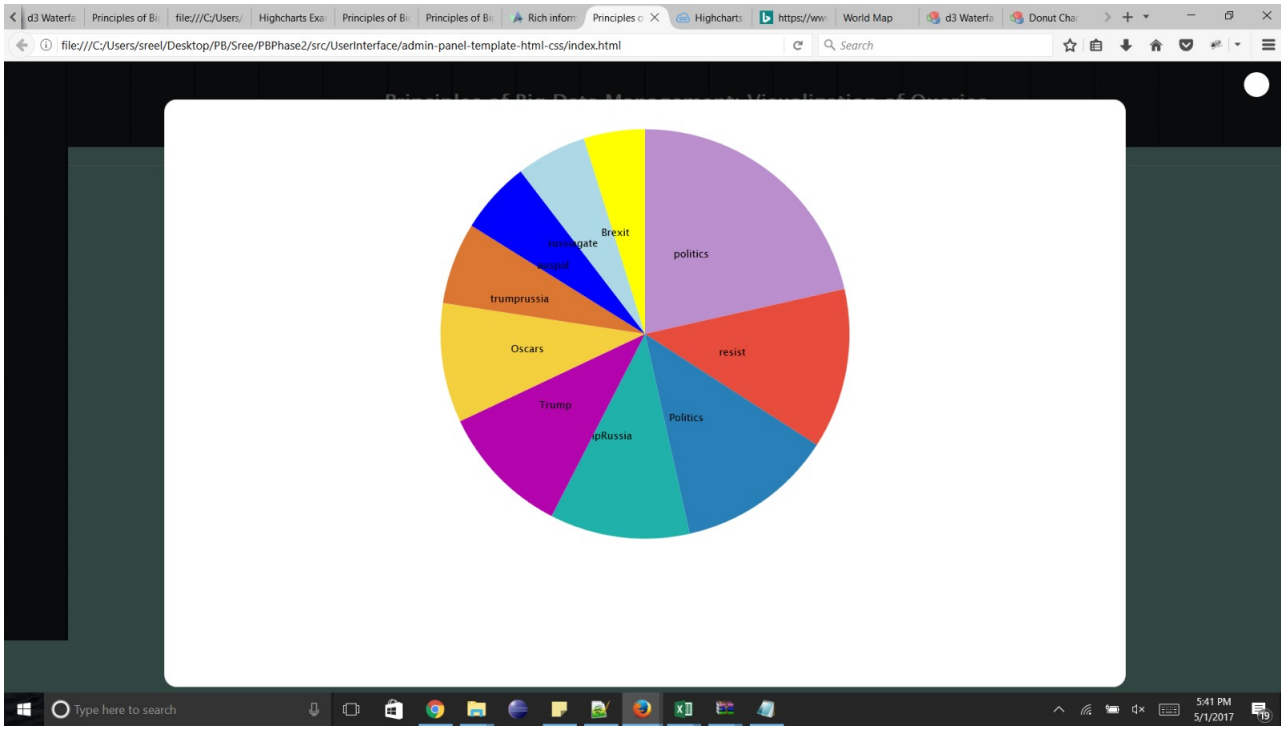

# Query 3:



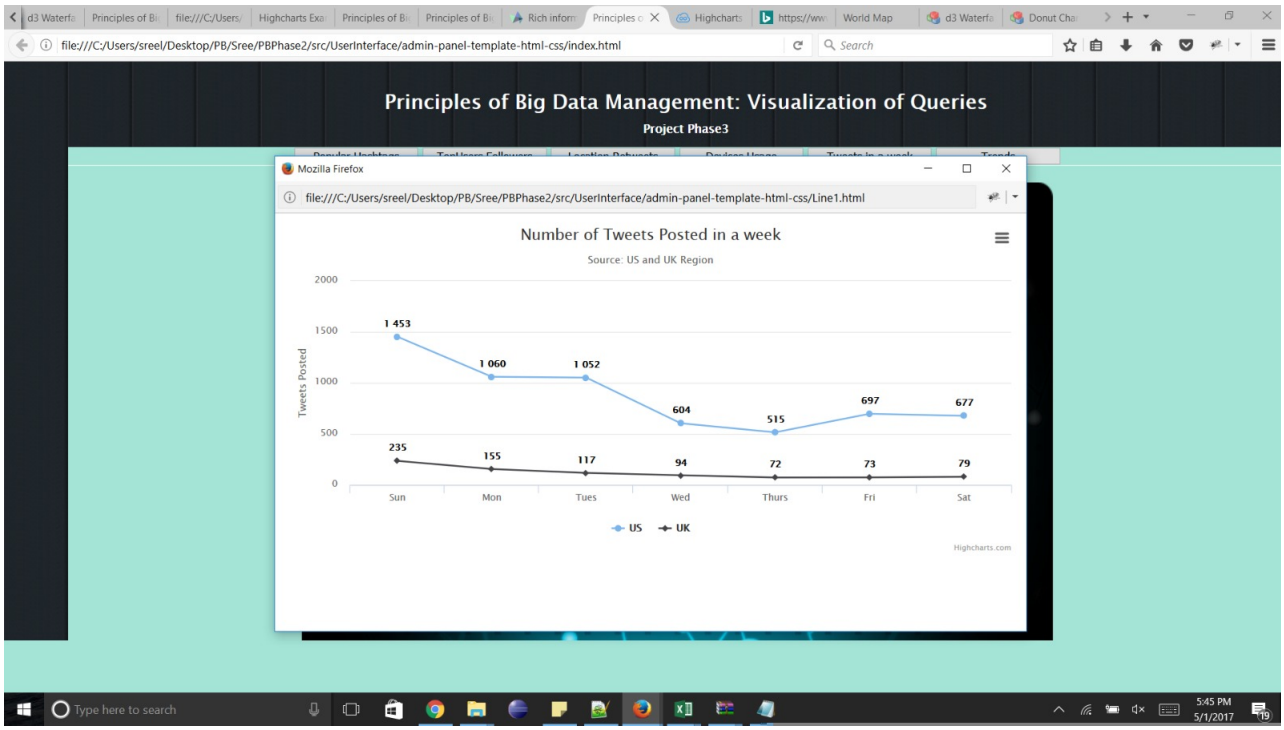

Principles of Big Data Management: Visualization of Queries
Project Phase3
World Map - Mozilla Firefox
file:///C:/Users/sreel/Desktop/PB/Sree/PBPhase2/src/UserInterface/admin-panel-template-html-css/maps.html
Android Users in world
Count
United States of America: 567 Users
Individuals User Count
< 3
3 – 10
10 – 30
30 – 100
100 – 300
300 – 1 000
> 1 000


Principles of Big Data Management: Visualization of Queries
Project Phase3
World Map - Mozilla Firefox
file:///C:/Users/sreel/Desktop/PB/Sree/PBPhase2/src/UserInterface/admin-panel-template-html-css/maps.html
Android Users in world
Count
India: 130 Users
Individuals User Count
< 3
3 – 10
10 – 30
30 – 100
100 – 300
300 – 1 000
> 1 000

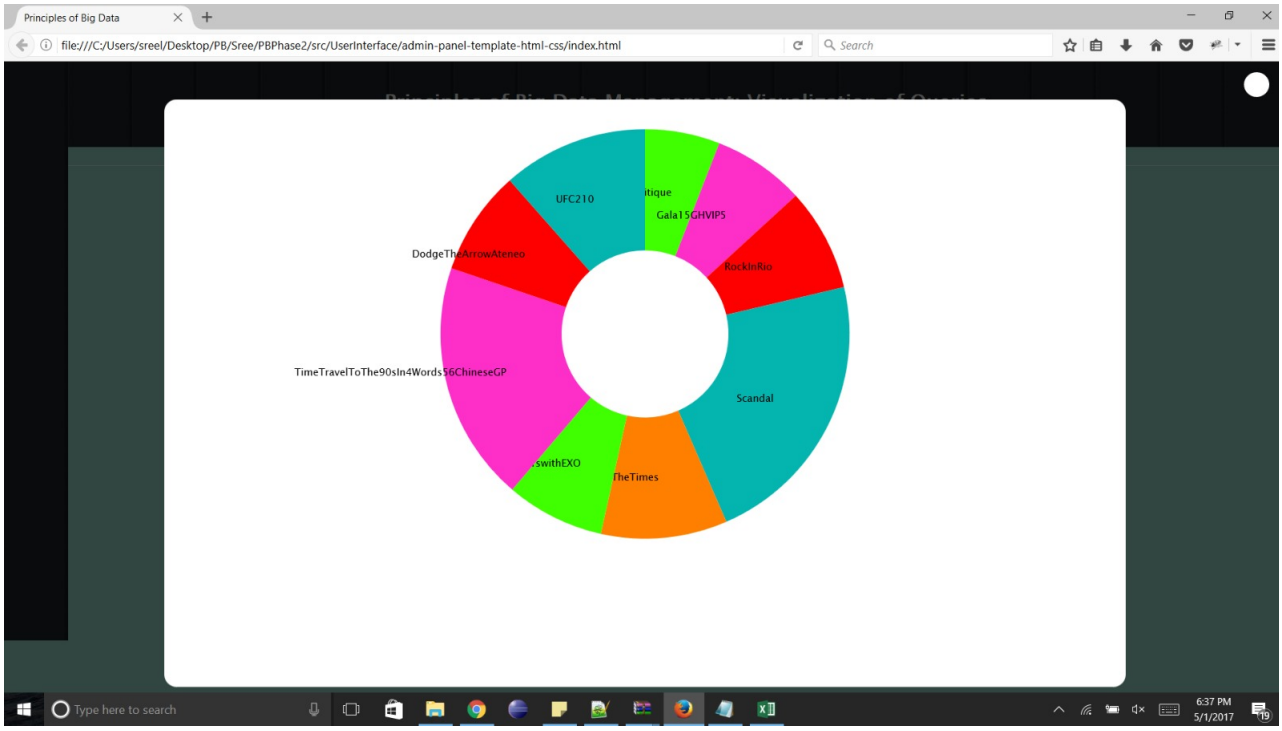
# Query 4:




# Query 5:



**Query 6:**




# 6.References


- http://stackoverflow.com
- http://www.tweepy.org/
- https://www.cloudera.com/

- http://www.aegissofttech.com/Articles/how-to-get-top-n-words-count-using-bigdata-hadoop-mapreduce-paradigm-with-developers-assistance.html
- http://bl.ocks.org/mbostock/3887235
- http://www.highcharts.com/demo/line-basic/gray
- http://bl.ocks.org/phuonghuynh/54a2f97950feadb45b07
- http://www.highcharts.com/maps/demo/data-class-ranges