# *TWITTER DATA ANALYTICS*

## PRINCPLES OF BIGDATA MANAGEMENT


**Project By**

**1. Archana Reddy Basani (16250971)**

**2. Mourya Praharsha Bobbili (16251221)**

**3. Dileep Kumar Durgam (16261222)**

# Project Goal

- Collect Tweets using Twitter's Streaming APIs (e.g., 100K Tweets)
    - https://dev.twitter.com/docs/streaming-apis
    - Search online for documentation
- Extract all the hashtags and URLs in the tweets
- Run the Word Count example in Apache Hadoop and Apache Spark on the extracted hashtags/URLs and collect the output and log files from Hadoop.
- Add a README file.

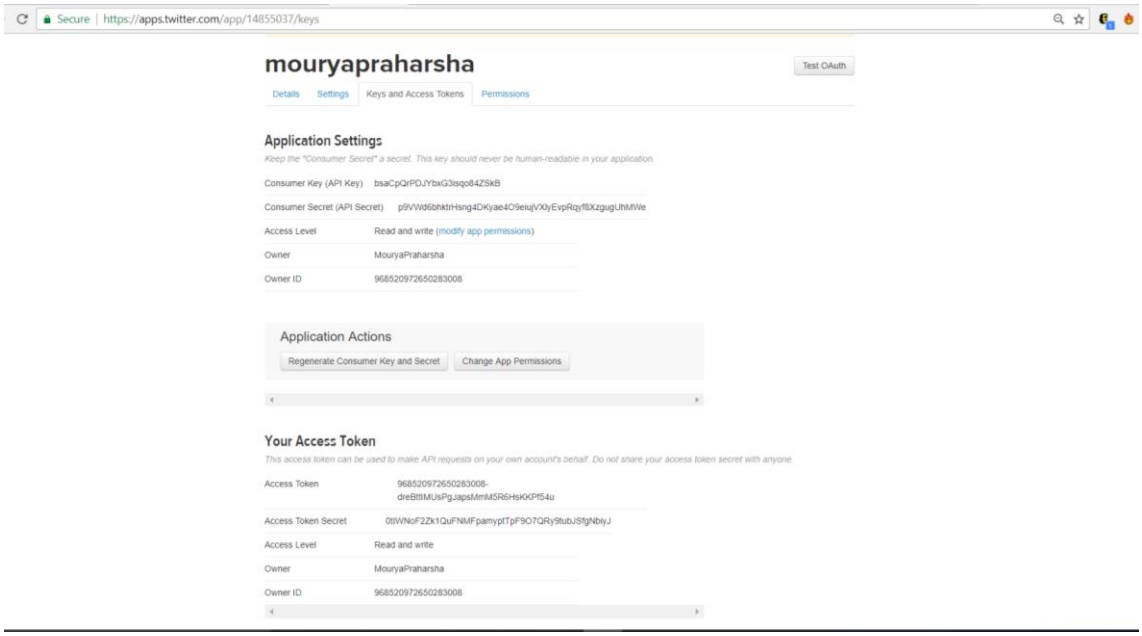
# Project Abstract


- Created a twitter API keys and then Collected 100K Tweets using Tweepy API using python in JSON format.
- Collected the tweets from the 7 top hashtags that were trending in the twitter.
- Extracted all the hashtags and URLs in the tweets.
- Created a directory in HDFS for all these 7 hashtags.
- Now pushed the tweets into that directory.
- Searched for a keyword in both the “text” and “hashtags” columns in a tweet and gave a final word count in Apache Hadoop and Apache Spark.
- Added the log files that are collected from Hadoop.

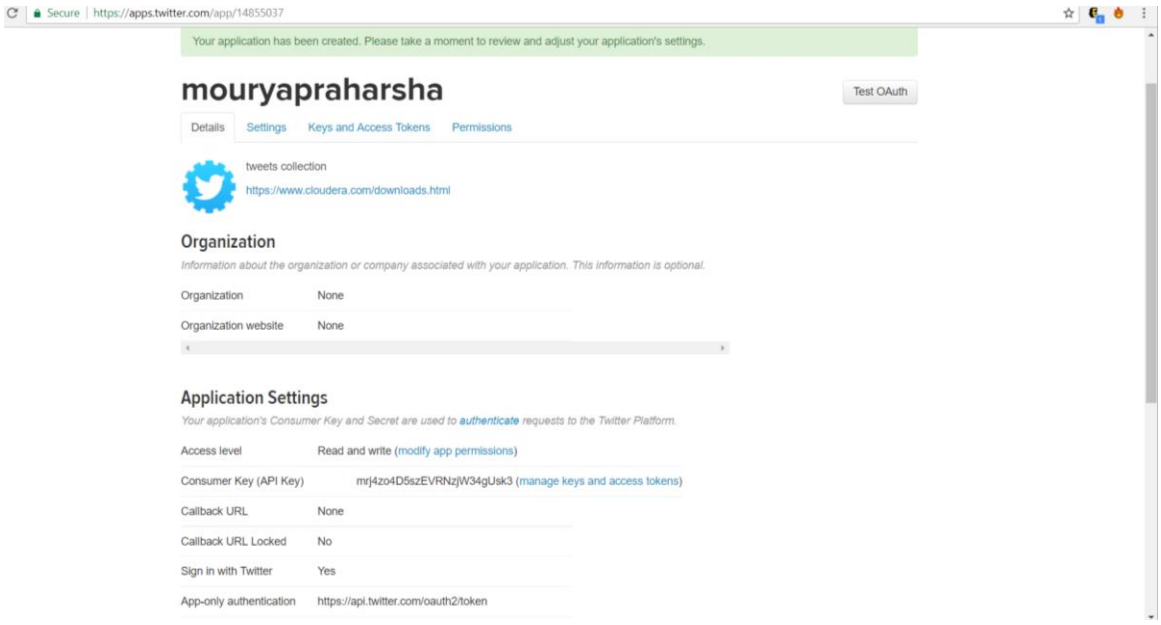
# Tweets Api Key

Created  a twitter api key using the twitter developers account ,the key is enabled for creating the new application

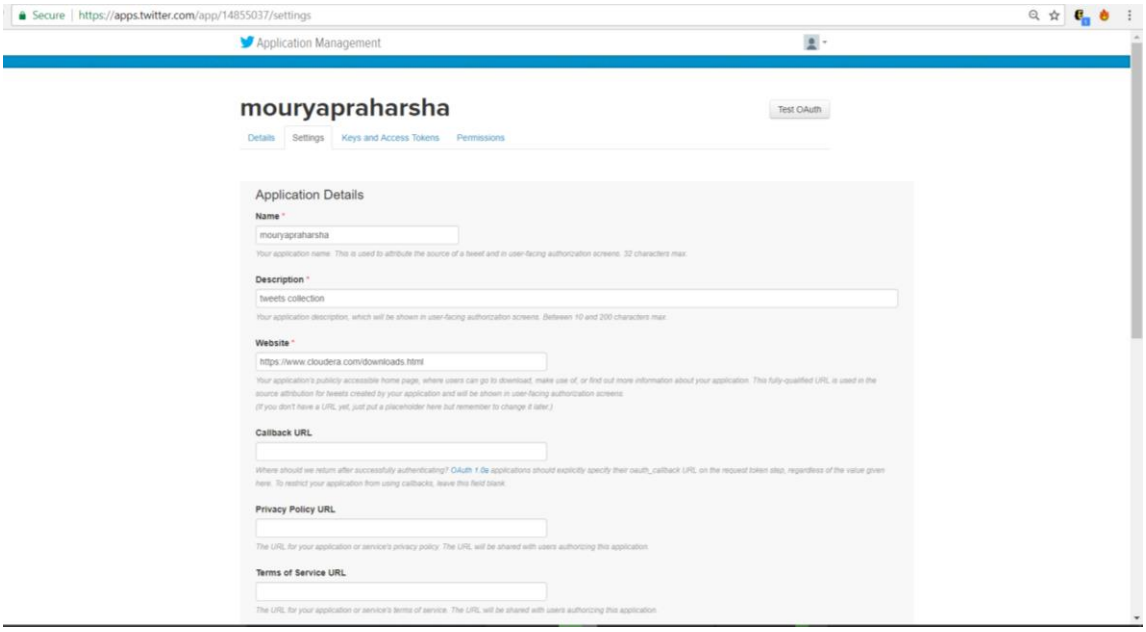
- **Key and Access tokens**



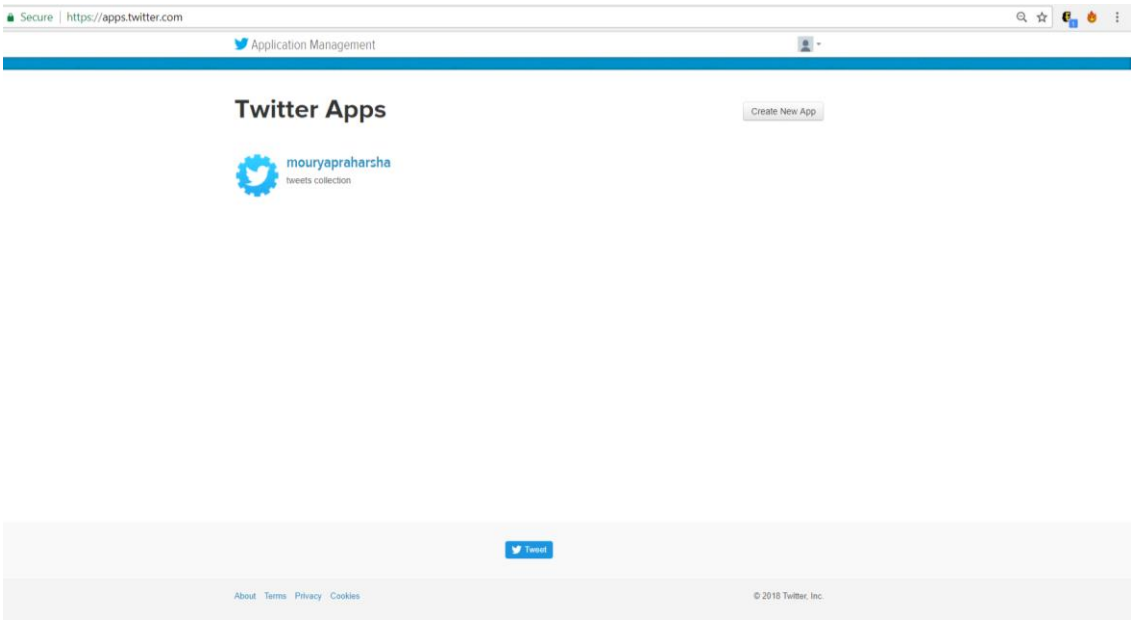

- **Tweets Collection details**



- **Twitter App Management settings**




- **Twitter App Management**



# Operating Systems

Ubuntu operating system was used for the generation of the tweet and extraction of the tweets. The code is run in python 3.6.2.

- **Operating System: Ubuntu.**
- **Software tools: Hadoop, Spark.**
- **Coding Language: Python 3.6.2.**

## 1. Tweets Collection Code.

```python
import sys
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream

access_token = "968520972650283008-dreBttIMUsPgJapsMmM5R6HsKKPf54u"
access_token_secret = "0tIWNoP2Zk1QuFNMPpamypptTpP9O7QRy9tubJSfgNbiyJ"
consumer_key = "bsaCpQrPDJYbxG3isqo84ZSkB"
consumer_secret = "p9VWd6bhktrHsng4DKyae4O9eiujVXlyEvpRqyf8XzgugUhMWe"


class StdOutListener(StreamListener):
    def on_data(self, data):
        print(data)
        with open('fetched_tweets.json', 'a') as tf:
            tf.write(data)
        return True


if __name__ == '__main__':
    l = StdOutListener()
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    stream = Stream(auth, l)

    stream.filter(track=['floridashooting','RIP','Sridevi','Syria','Nationalpancakeday','NFL','basketball'])
```

## 2. Tweets Extraction of hashtags and url's.

```python
import codecs
import json
import sys

def parse_json_tweet(line):
    tweet = json.loads(line)
    htags = [hashtag['text'] for hashtag in tweet['entities']['hashtags']]
    urls = [url['expanded_url'] for url in tweet['entities']['urls']]
    return [htags, urls]
if __name__ == "__main__":
    file_timeordered_json_tweets = codecs.open("C:\\Users\\ARCHANA REDDY\\Desktop\\pb\\fetched_tweets.json", 'r', 'utf-8')
    fout = codecs.open("C:\\Users\\ARCHANA REDDY\\Desktop\\pb\\tweetsoutput.txt", 'w','utf-8')
for line in file_timeordered_json_tweets:
    try:
        [htags, urls] = parse_json_tweet(line)
        fout.write(str([htags, urls]) + "\n")
    except:
        pass

file_timeordered_json_tweets.close()
fout.close()
```

## 3.Tweets Output after extraction

C:\Users\ARCHANA REDDY\Desktop\pb\tweetsoutput.txt - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

part-r-00000 | WordCount.java | tweetsoutput.txt

```
[[], []]
[[u'Syria'], []]
[[], []]
[[], []]
[[], [u'https://twitter.com/i/web/status/968660504699899905']]
[[], []]
[[], []]
[[u'Syria'], []]
[[], [u'https://twitter.com/i/web/status/968660506599927808']]
[[u'SriDevi', u'sridevideath'], []]
[[u'RIPCricket'], []]
[[u'WeNeedPeacefulSyria'], []]
[[], []]
[[u'DPRK', u'Syria'], [u'https://twitter.com/jaketapper/status/968620703158325248']]
[[], []]
[[], [u'http://dailydolphin.blog.palmbeachpost.com/2018/02/27/2018-nfl-draft-te-ian-thomas-purpose-was-formed-by-difficult-childhood/']]
[[u'Sridevi'], []]
[[], []]
[[u'FenwickFriars'], []]
[[], [u'https://twitter.com/mila_jeik/status/968289179191906309']]
[[], [u'http://ale.io/dhu4j']]
[[], []]
[[], []]
[[u'Syria'], []]
[[], []]
[[], [u'http://ptv.io/2Xnk']]
[[u'Syria'], []]
[[], []]
[[u'SyriaIsBleeding'], []]
[[], []]
[[], []]
[[], []]
[[], []]
[[u'Sridevi', u'\u0939\u092e\u0915\u094b_\u092d\u0940_\u092e\u0930\u0928\u093e_\u0939\u0948'], []]
[[], []]
[[u'Sridevi', u'LetHerRestInPeace'], []]
[[u'Cowboys'], [u'https://www.bloggingtheboys.com/2018/2/27/17058904/its-semi-official-dez-bvant-caught-it-dallas-cowboys-green-bay-packers?utm_campaign=bloggingtheboys&utm_content=chor
[[], []]
[[], []]
[[], []]
[[], []]
[[], []]
```

Normal text file | length : 65,08,822 lines : 1,67,255 | Ln : 1 Col : 1 Sel : 0 | 0 | Unix (LF) | UTF-8 | INS

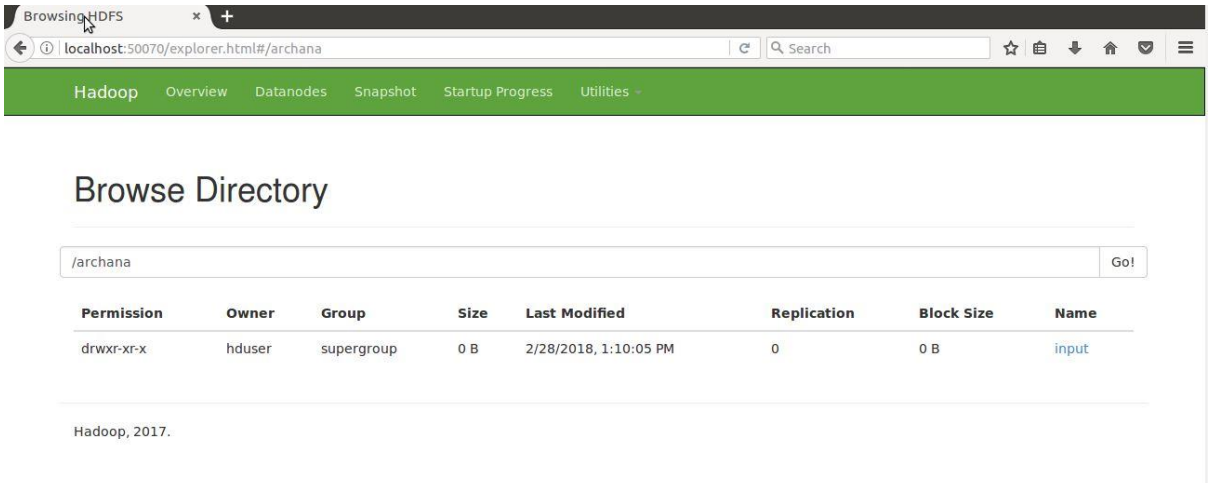
## 4.Hadoop Word Count Program

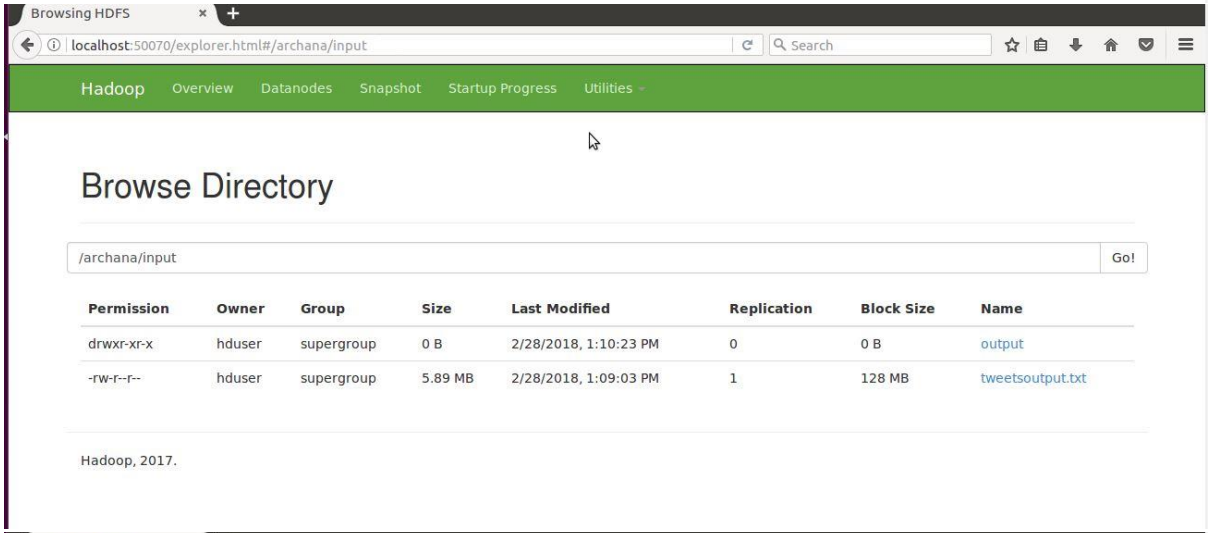
tweets_output.txt | part-r-00000 | WordCount.java

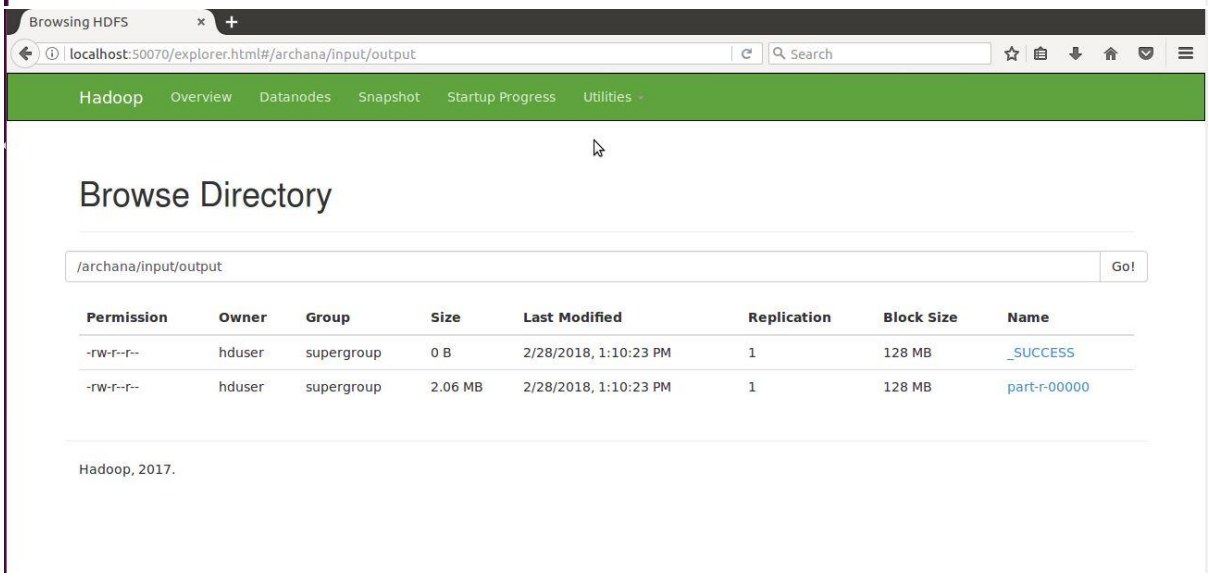
```java
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCount {
  public static class TokenizerMapper
      extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }
  public static class IntSumReducer
      extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }
  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
```

Java source file | length : 2,138 lines : 56 | Ln : 35 Col : 19 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

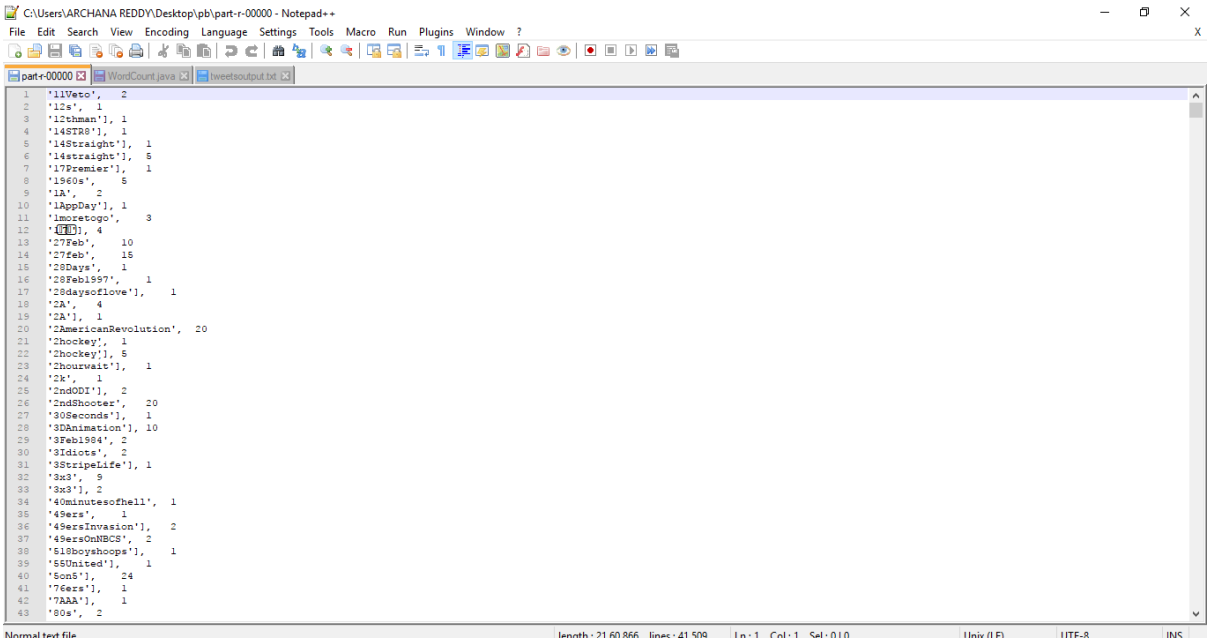
## 5.Hadoop Outputs



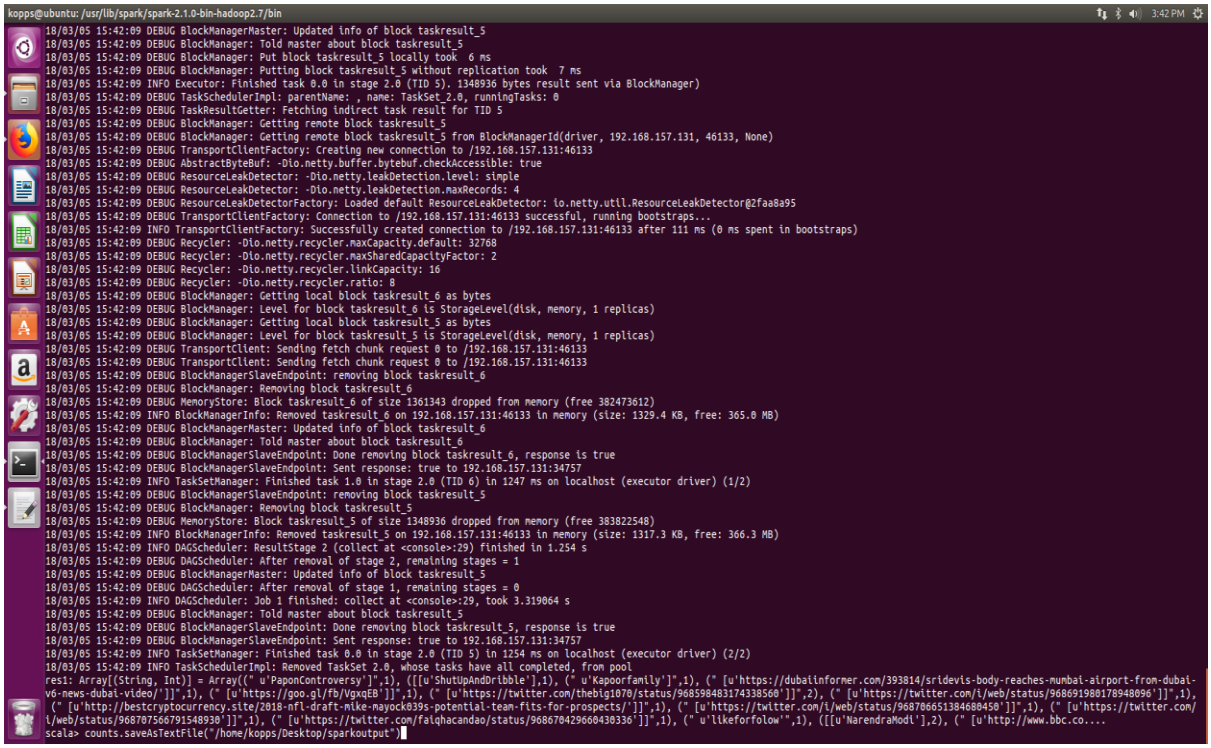



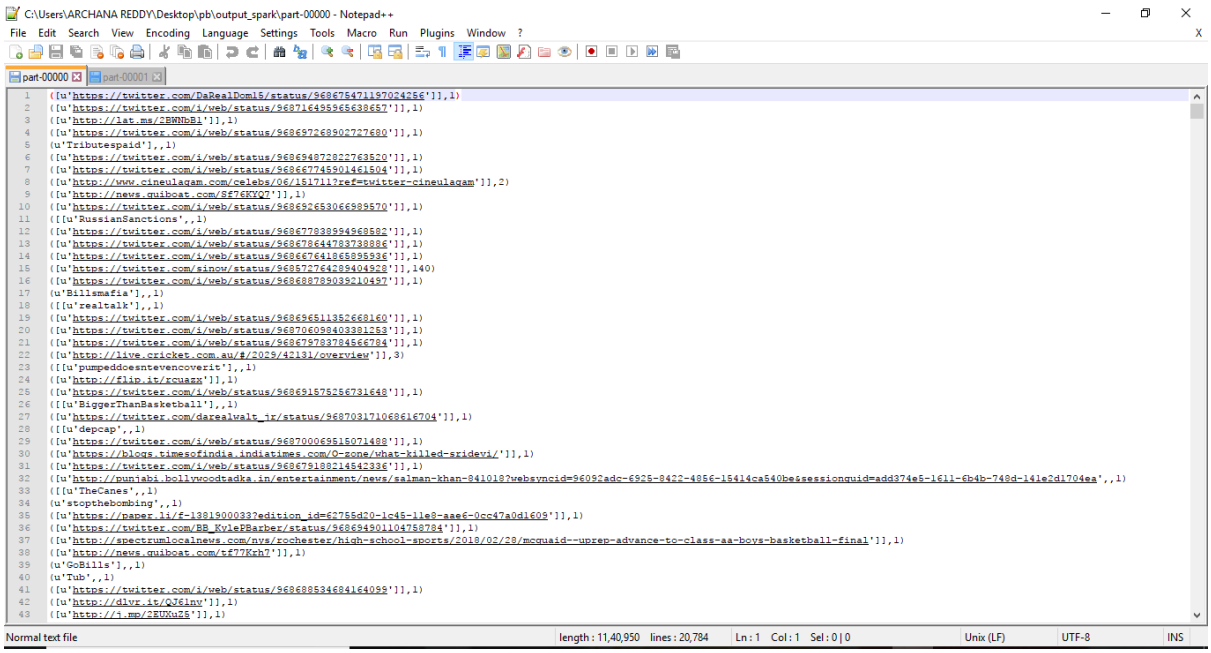
## 6.Hadoop part-r-00000



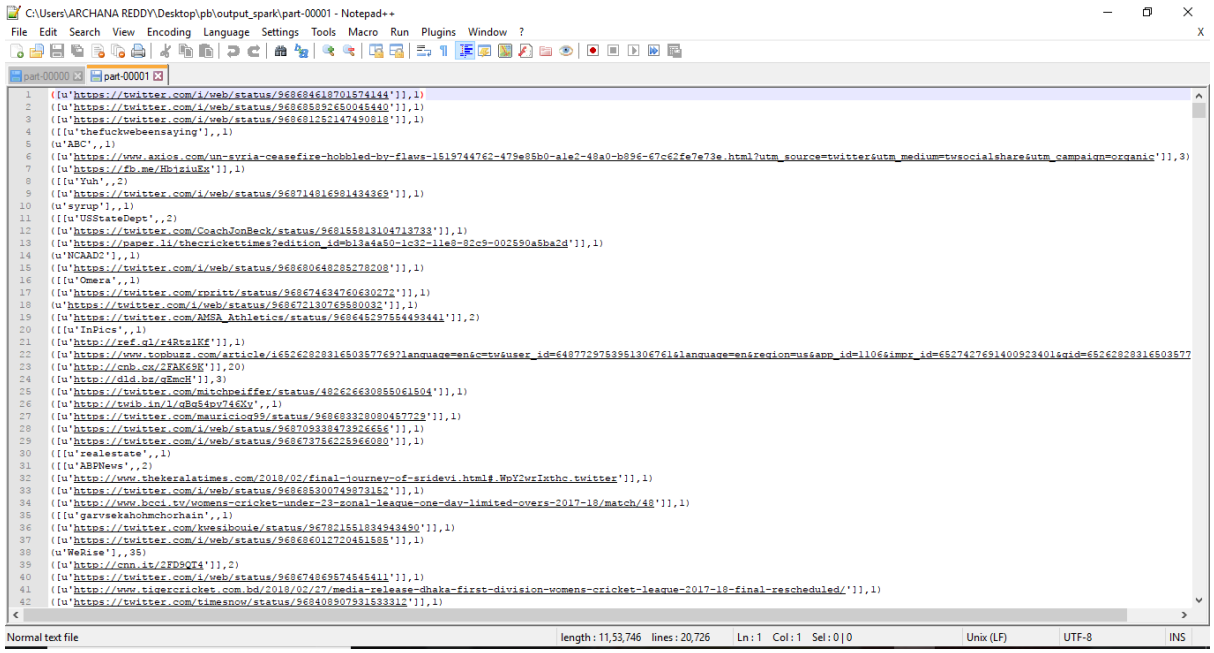

## 7.Spark word count program



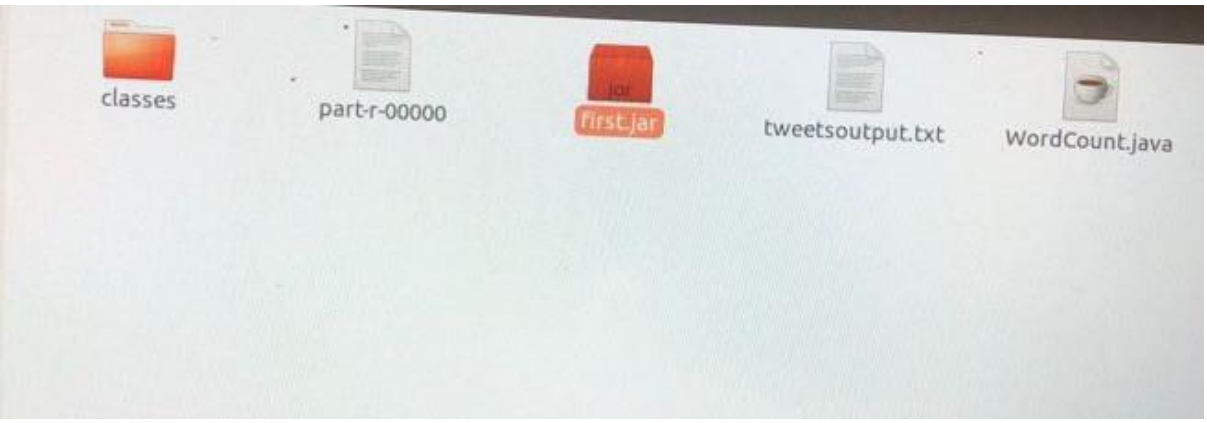
## 8.Spark Part-r-00000




## 9.Spark part-r-00001



**HDFS**




**Google Drive Link:**

https://drive.google.com/open?id=1JPBvoMpQHfHj-w2ldo19YknTw3Wl0sfS

**Reference:**

https://www.cloudera.com/

https://github.com/SivagamiNambi/Twitter-Sentiment-Analysis

# PHASE 2 IMPLEMENTATION

# TWEETS STORAGE USING SPARK SQL

## ABSTRACT:

The fundamental point of this task is to break down the huge information gathered from online networking (twitter). In this task, we have gathered twitter information (tweets) on some slanting themes “SYRIA”, “FLORIDASHOOTING”, “RIP”, “SRIDEVI”, “NFL”, “BASKETBALL”, “NATIONALPANCAKEDAY” and we have broken down the gathered enormous information utilizing Apache Spark. We have actualized distinctive questions utilizing Spark Data outlines and an open API to break down the gathered information and drawn some fascinating yields from inquiry investigation.

### IMPLEMENTATION:

- Collected twitter data (tweets) related to “SYRIA”, “FLORIDASHOOTING”, “RIP”, “SRIDEVI”, “NFL”, “BASKETBALL”, “NATIONALPANCAKEDAY” in JSON format.
- Developed the environment IntelliJ for Scala and Spark development.
- Queries has been written and displayed as per the analysis.
- Explanation of the ten queries and their outputs (captured screenshots) are documented.

### SETTING UP OF ENVIRONMENT:

In our undertaking, we utilized IntelliJ for Scala and Spark improvement. IntelliJ Scala blend is the best, free setup for Scala and Spark improvement. To run IntelliJ, we require Java JDK introduced in our Framework. Also, by utilize Spark APIs make Scala question and import Spark shakes as library conditions in IntelliJ lastly add some Spark API calls to the made protest. Presently IntelliJ for Scala and Spark improvement condition is setup and we are prepared to actualize distinctive questions (Spark RDDs and Data frames) on our gathered stream of tweets for examination.

In this increment of project, we have taken the JSON file from the first phase and stored it in the form of Main table and queries are written in the SQL language for the extraction of the outputs and hash table is assigned for the output designed.

Ten queries are written in the SQL language and executed in the SCALA code which was written for execution.

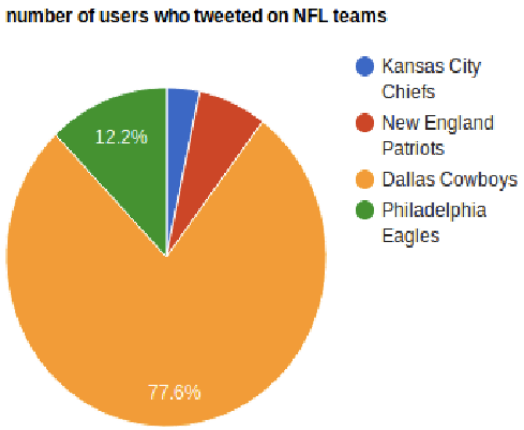
## QUERY 1:

```
val post=sqlContext.sql("select user.name,text from TweetTable where text like '%Kansas City chiefs%'  ")
// val post=sqlContext.sql("select  count(user.name),count(text) from TweetTable where text like '%Dallas CowBoys%' ")
//val post=sqlContext.sql("select user.name,text from TweetTable where text like '%New England patriots%' ")
//val post=sqlContext.sql("select user.name,text from TweetTable where text like '%Philadelphia Eagles%' ")
//      post.show()
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("outputfootballeagles")
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("outputfootballCowboys")
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("outputfootballChiefs")
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("outputfootballPatriots")
```

## DESCRIPTION OF QUERY 1:

The Query gives the Number of users who tweeted on NFL teams and ouput is saved in text format.

## VISUALIZATION OF QUERY 1:



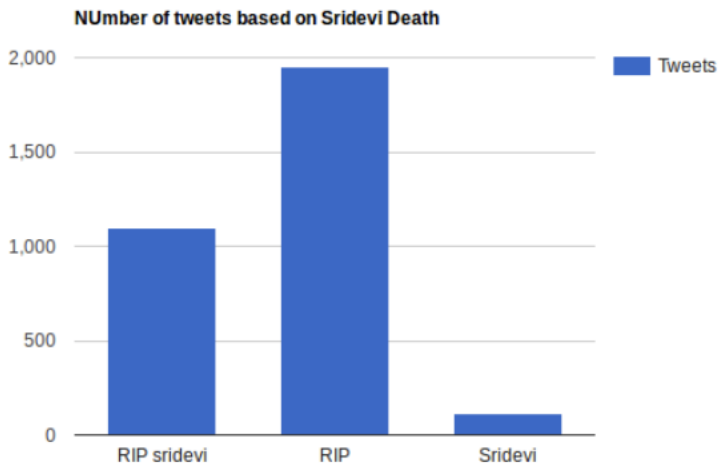

## QUERY 2:

```
val post=sqlContext.sql("select count(user.name), count(text) from TweetTable where text like '%Sridevi%' ")

val post=sqlContext.sql("select count(user.name), count(text) from TweetTable where text like '%RIP%' ")

val post=sqlContext.sql("select count(user.name), count(text) from TweetTable where text like '%RIP Sridevi%' "
    post.show()
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("output6")

post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("output7")
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("output8")
```

## DESCRIPTION OF QUERY 2:

This query is number of tweets based on Sridevi death and output is saved in text format.

## VISUALIZATION OF QUERY 2:



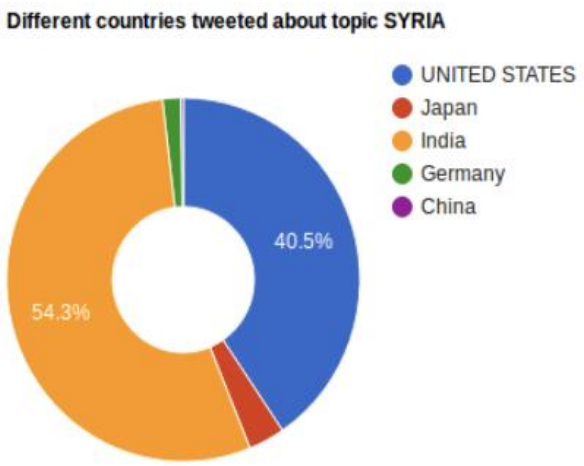

## QUERY 3:

```
val post=sqlContext.sql("select user.name, SUBSTRING(user.created_at,5,3) AS month,SUBSTRING(user.created_at,
    12,8) AS time from TweetTable where text LIKE '%nfl%' group by user")
//   val post=sqlContext.sql("select count(text), count(user.location) from TweetTable where text like '%Syria%'
    and user.location = 'China'")
val post=sqlContext.sql("select count(text), count(user.location) from TweetTable where text like '%Syria%' and
    user.location = 'India'")
val post=sqlContext.sql("select count(text), count(user.location) from TweetTable where text like '%Syria%' and
    user.location = 'United States'")
val post=sqlContext.sql("select count(text), count(user.location) from TweetTable where text like '%Syria%' and
    user.location = 'Germany'")

val post=sqlContext.sql("select count(text), count(user.location) from TweetTable where text like '%Syria%' and
    user.location = 'Japan'")post.show()
//    post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Germany")

post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("India")

post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("China")

post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Japan")

post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("United States")
```

## DESCRIPTION OF QUERY 3:

This query is of Different countries tweeted about “SYRIA” and output is saved in text format.

## VISUALIZATION OF QUERY 3:



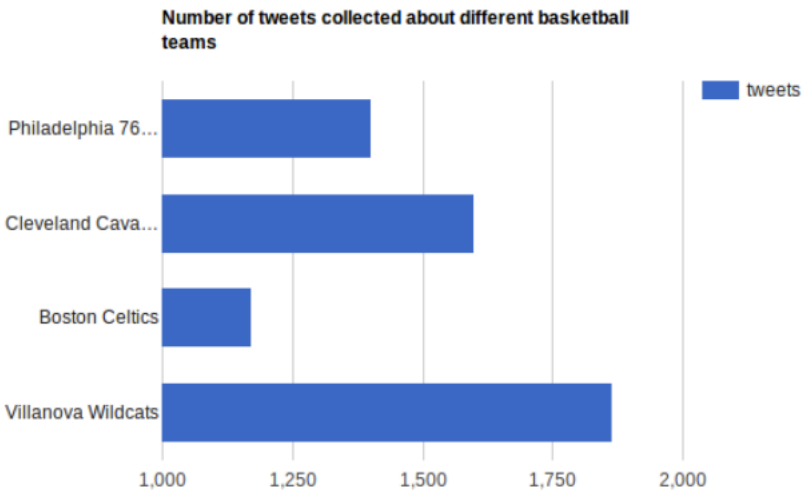

## Query 4:

```
/val post=sqlContext.sql("select count(user.screen_name),count(text) from TweetTable where text like '%Villanova Wildcats%'")

val post=sqlContext.sql("select count(user.screen_name),count(text) from TweetTable where text like '%Philadelphia 76ers%'")

val post=sqlContext.sql("select count(user.screen_name),count(text) from TweetTable where text like '%Boston Celtics%'")
val post=sqlContext.sql("select count(user.screen_name),count(text) from TweetTable where text like '%Cleveland cavalaries%'")//    post.show()
//    post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("output4")

post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("output5")

post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("output6")

post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("output7")

post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("output8")
```

## DESCRIPTION OF QUERY 4:

This query is about Number of tweets collected about different basketball teams output is saved in text format.

## VISUALIZATION OF QUERY 4



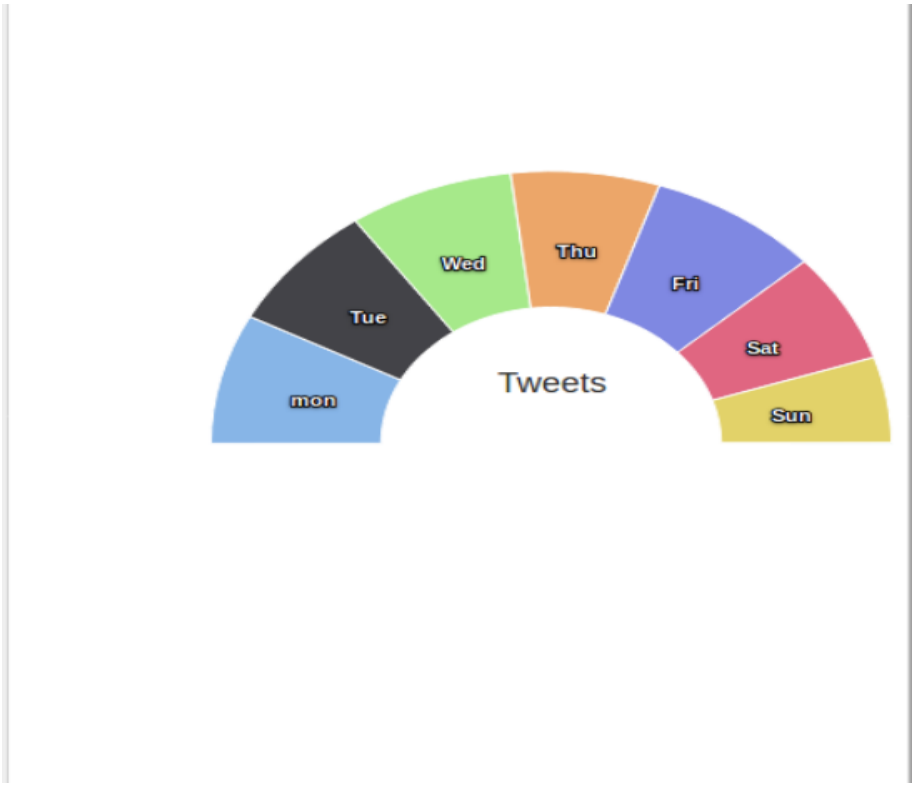
## Query 5:

```
val post=sqlContext.sql("SELECT count(user.name),count(text)  FROM TweetTable where text like '%basketball%' and user.created at  like '%mon%' ")
post.show()
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Mon")
val post=sqlContext.sql("SELECT count(user.name),count(text)  FROM TweetTable where text like '%basketball%' and user.created at  like '%tue%' ")
post.show()
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Tue")
val post=sqlContext.sql("SELECT count(user.name),count(text)  FROM TweetTable where text like '%basketball%' and user.created at  like '%wed%' ")
post.show()
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Wed")
val post=sqlContext.sql("SELECT count(user.name),count(text)  FROM TweetTable where text like '%basketball%' and user.created at  like '%thu%' ")
post.show()
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Thu")
val post=sqlContext.sql("SELECT count(user.name),count(text)  FROM TweetTable where text like '%basketball%' and user.created at  like '%fri%' ")
post.show()
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Fri")
val post=sqlContext.sql("SELECT count(user.name),count(text)  FROM TweetTable where text like '%basketball%' and user.created at  like '%sat%' ")
post.show()
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Sat")
val post=sqlContext.sql("SELECT count(user.name),count(text)  FROM TweetTable where text like '%basketball%' and user.created at  like '%sun%' ")
post.show()
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Sun")
```

## DESCRIPTION OF QUERY 5:

This query gives number of users who tweed about basketball in a day of week.

## VISUALIZATION OF QUERY 5:


Wed
Thu
Fri
Tue
Sat
Tweets
mon
Sun

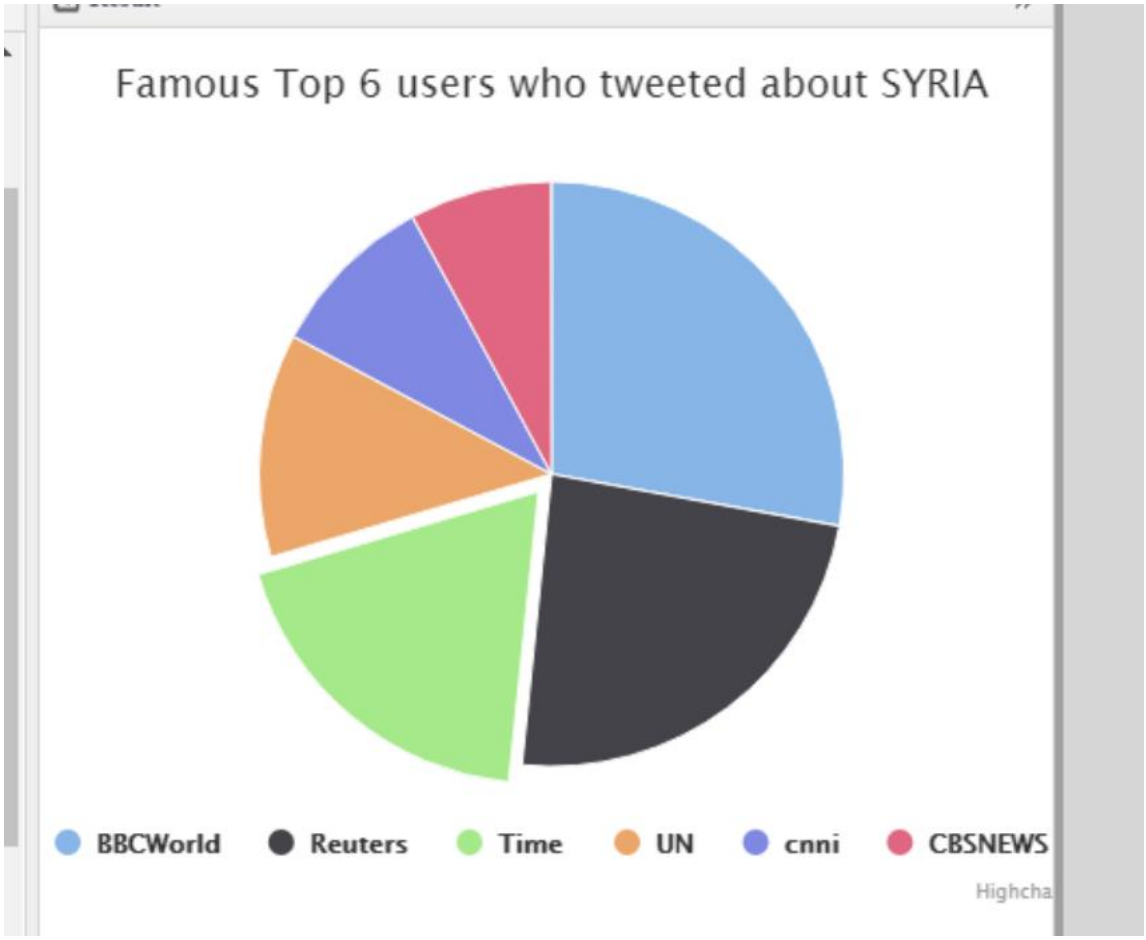
## Query 6:

```
val post=sqlContext.sql(sqltext="SELECT user.screen_name,user.followers_count  FROM TweetTable WHERE text like
    '%Syria%' ORDER BY user.followers_count DESC LIMIT 10")

post.show()
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("TestDemo")
```

## DESCRIPTION OF QUERY 6:

This query is about, tweets count of  Top famous 6 users who tweeted about “SYRIA” and output is saved in a text file named “testDemo”.

## VISUALIZATION OF QUERY 6:


Famous Top 6 users who tweeted about SYRIA
BBCWorld
Reuters
Time
UN
cnni
CBSNEWS
Highcha

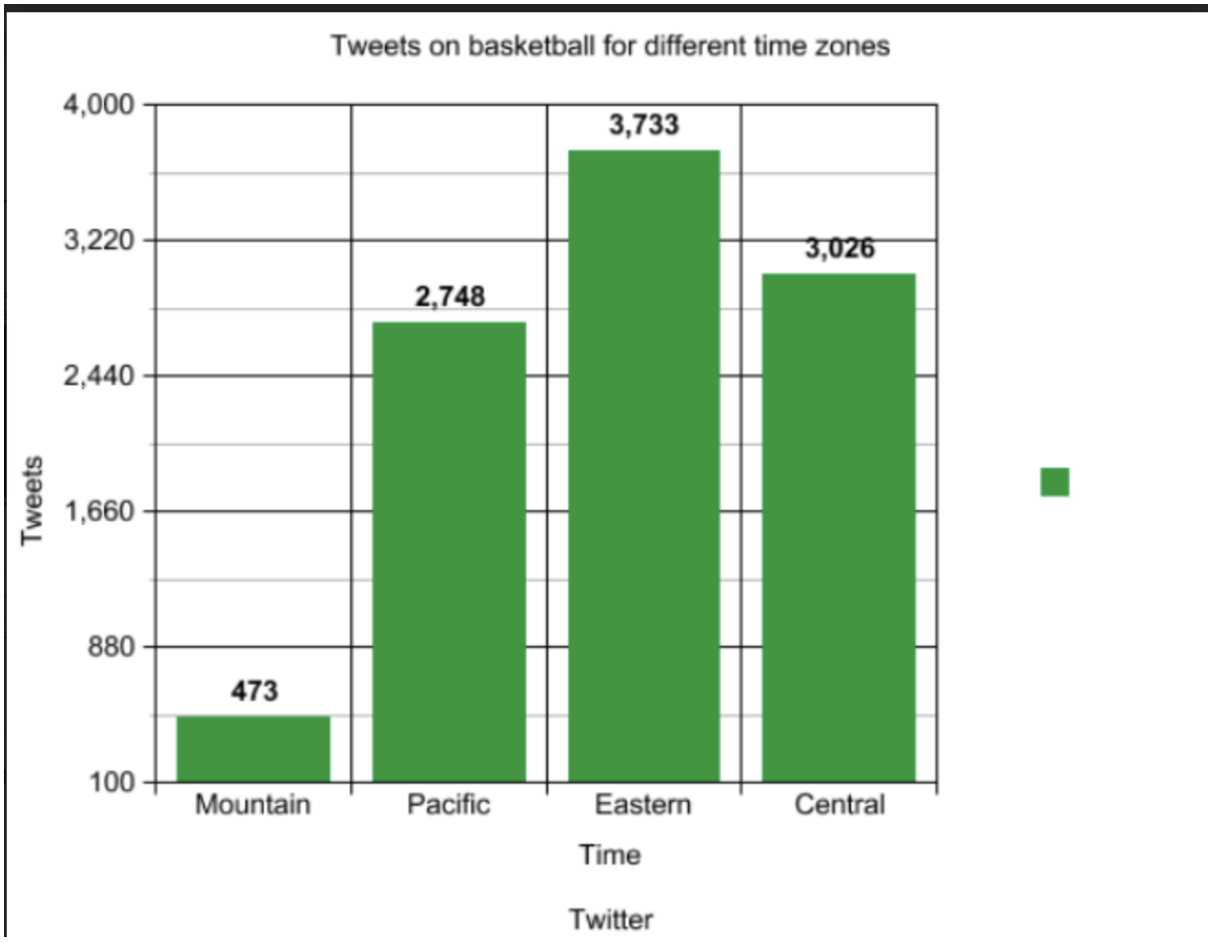
# Query 7:

```
i) val post=sqlContext.sql("select count(user.time_zone),count(user.screen_name) FROM TweetTable WHERE
    user.time_zone like '%Central%' and text like '%basketball%' ")
post.show()
post.map(x=> (x(0),x(1))).coalsce( numPartitions = 1, shuffle = true).saveAsTextFile( path = "testDemo12")

ii) val post=sqlContext.sql("select count(user.time_zone),count(user.screen_name) FROM TweetTable WHERE
    user.time_zone like '%Eastern%' and text like '%basketball%' ")
post.show()
post.map(x=> (x(0),x(1))).coalsce( numPartitions = 1, shuffle = true).saveAsTextFile( path = "testDemo1")

iii) val post=sqlContext.sql("select count(user.time_zone),count(user.screen_name) FROM TweetTable WHERE
    user.time_zone like '%Mountain%' and text like '%basketball%' ")
post.show()
post.map(x=> (x(0),x(1))).coalsce( numPartitions = 1, shuffle = true).saveAsTextFile( path = "testDemo1234")

iv) val post=sqlContext.sql("select count(user.time_zone),count(user.screen_name) FROM TweetTable WHERE
    user.time_zone like '%Pacific%' and text like '%basketball%' ")
post.show()
post.map(x=> (x(0),x(1))).coalsce( numPartitions = 1, shuffle = true).saveAsTextFile( path = "testDemo123")
```

# DESCRIPTION OF QUERY 7:

This query is of tweets count on basketball for different time zones.

## VISUALIZATION OF QUERY 7:


Tweets on basketball for different time zones
4,000
3,220
2,440
1,660
880
100
Tweets
473
2,748
3,733
3,026
Mountain
Pacific
Eastern
Central
Time
Twitter

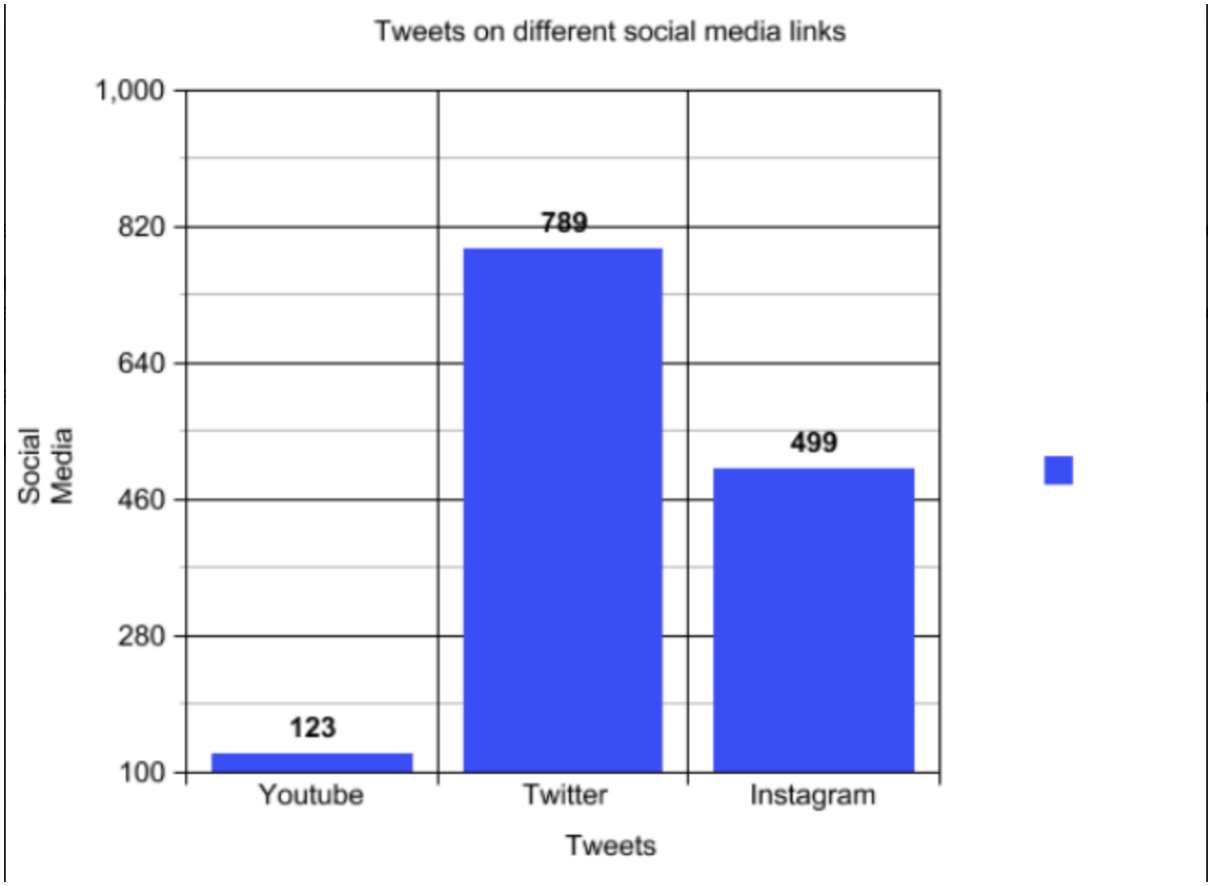
## Query 8:

```
val post=sqlContext.sql( sqlText = "SELECT count (text), count(user.screen_name) FROM TweetTable WHERE
    user.description like '%youtu.b%' ")
post.show()
post.map(x=> (x(0),x(1))).coalesce( numPartitions = 1, shuffle = true).saveAsTextFile( path = "youtube")
val post=sqlContext.sql( sqlText = "SELECT count (text), count(user.screen_name) FROM TweetTable WHERE
    user.description like '%twitter%' ")
post.show()
post.map(x=> (x(0),x(1))).coalesce( numPartitions = 1, shuffle = true).saveAsTextFile( path = "Twitter")
val post=sqlContext.sql( sqlText = "SELECT count (text), count(user.screen_name) FROM TweetTable WHERE
    user.description like '%Instagram%' ")
post.show()
post.map(x=> (x(0),x(1))).coalesce( numPartitions = 1, shuffle = true).saveAsTextFile( path = "Instagram")
```

## DESCRIPTION OF QUERY 8:

This query is “Different Social media links based on user count.”

## VISUALIZATION OF QUERY 8:



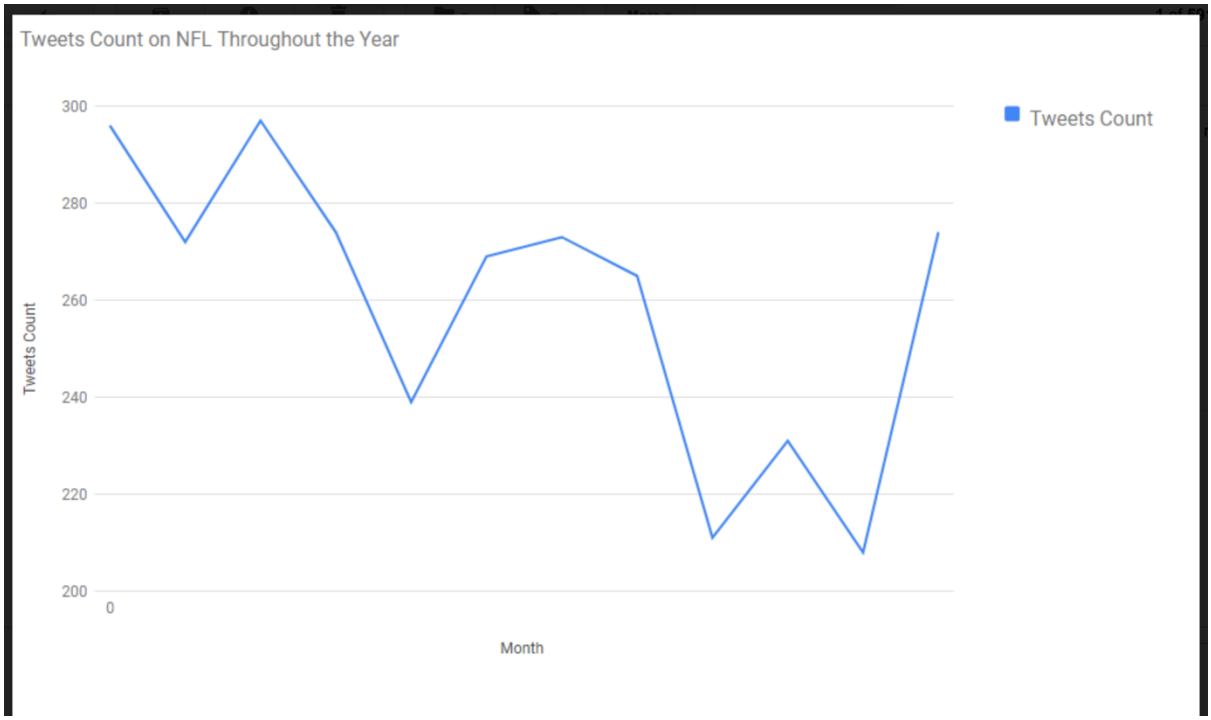

## Query 9:

```
val post=sqlContext.sql("SELECT count(text),count(user.screen_name)  FROM TweetTable where user.created_at like
    '%Mar%'and text like '%nfl%' ")
post.show()
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("March")
```

## DESCRIPTION OF QUERY 9:

This query is “ Tweets count on NFL Throughout the year”

## VISUALIZATION OF QUERY 9:




## Query 10:

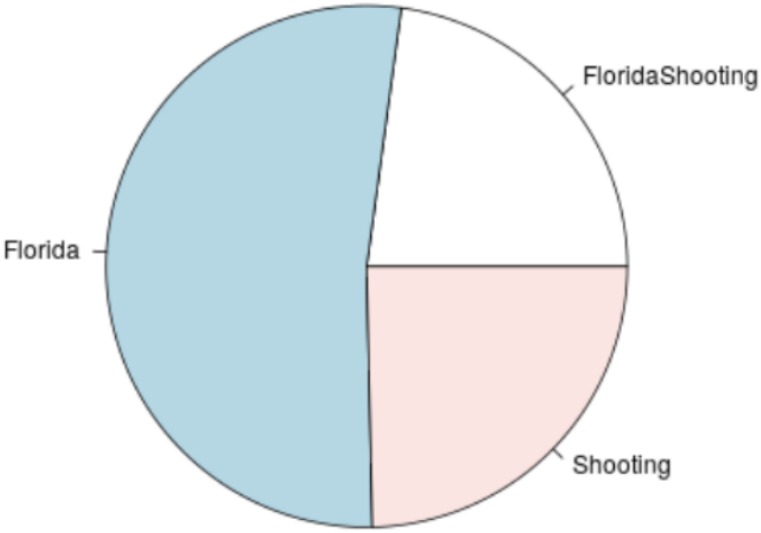
```
val post=sqlContext.sql("select count(user.name), count(text) from TweetTable where text like
    '%FloridaShooting%' ")

post.show()
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Fs")
val post=sqlContext.sql("select count(user.name), count(text) from TweetTable where text like '%Florida%' ")

post.show()
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Fs1")
val post=sqlContext.sql("select count(user.name), count(text) from TweetTable where text like '%Shooting%' ")

post.show()
post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Fs2")
```

## DESCRIPTION OF QUERY 10:

This query counts the number of users tweeted on florida shooting.

## VISUALIZATION OF QUERY 10:

```
$Rscript main.r
png
  2
```



: