



TWITTER DATA ANALYTICS

PRINCIPLES 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**

The screenshot shows the 'Keys and Access Tokens' tab for the application 'mouryapraharsha'. It displays the 'Application Settings' section with fields for Consumer Key (API Key), Consumer Secret (API Secret), Access Level (Read and write), Owner (MouryaPraharsha), and Owner ID (968520972650283008). Below this is the 'Application Actions' section with buttons for 'Regenerate Consumer Key and Secret' and 'Change App Permissions'. The 'Your Access Token' section shows the Access Token, Access Token Secret, Access Level (Read and write), Owner (MouryaPraharsha), and Owner ID (968520972650283008).

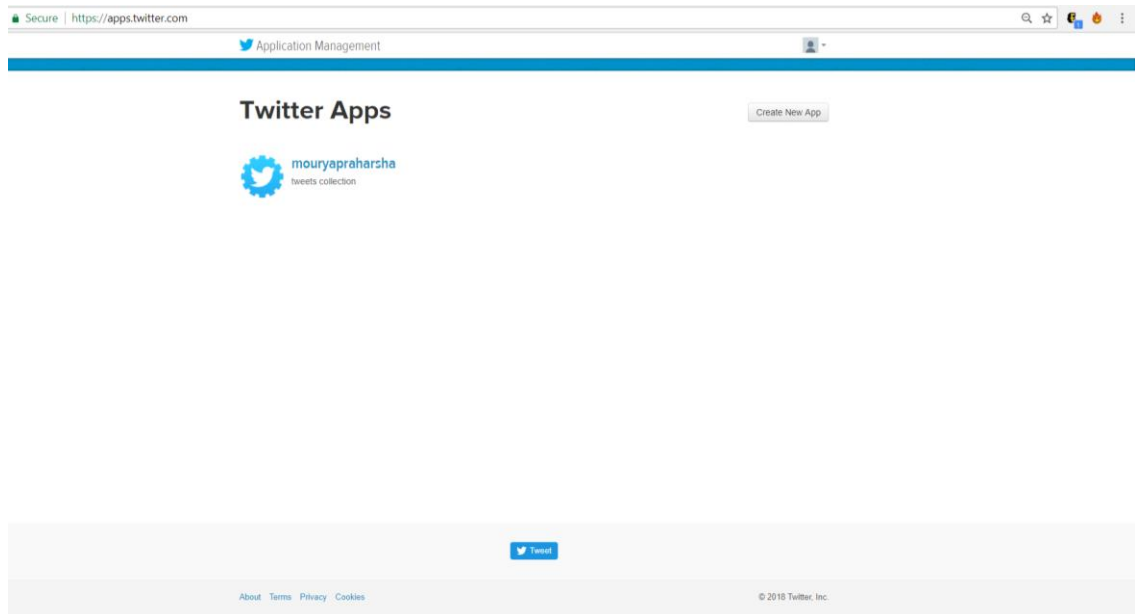
- **Tweets Collection details**

The screenshot shows the 'Settings' tab for the application 'mouryapraharsha'. It displays the 'Organization' section with fields for Organization (None) and Organization website (None). Below this is the 'Application Settings' section with fields for Access level (Read and write), Consumer Key (API Key) (mj4zo4D5szEVRNzjW34gU3k3), Callback URL (None), Callback URL Locked (No), Sign in with Twitter (Yes), and App-only authentication (https://api.twitter.com/oauth2/token).

- Twitter App Management settings

The screenshot shows the 'Application Management' page for a Twitter app named 'mouryapraharsha'. The page has a blue header with the Twitter logo and the app name. Below the header, there are tabs for 'Details', 'Settings', 'Keys and Access Tokens', and 'Permissions'. The 'Details' tab is selected. The 'Application Details' section contains several form fields: 'Name' (filled with 'mouryapraharsha'), 'Description' (filled with 'tweets collection'), 'Website' (filled with 'https://www.cloudera.com/downloads.html'), 'Callback URL' (empty), 'Privacy Policy URL' (empty), and 'Terms of Service URL' (empty). Each field has a small text description below it. A 'Test OAuth' button is located in the top right corner of the details section.

- Twitter App Management

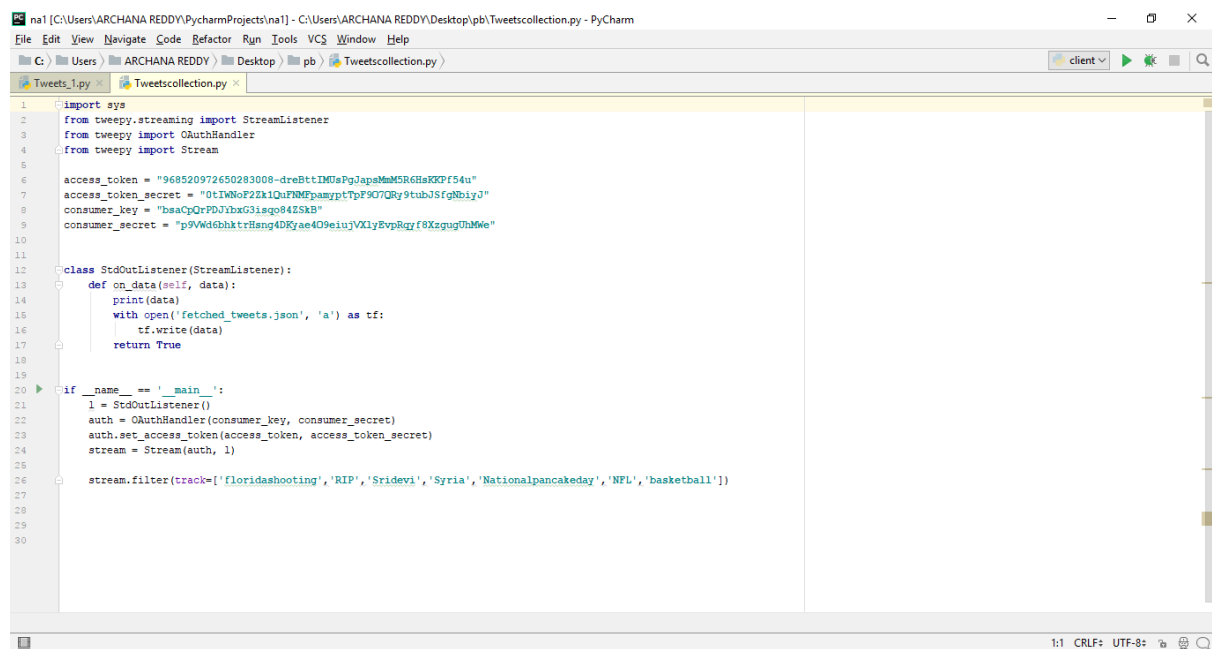


Operating Systems

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.

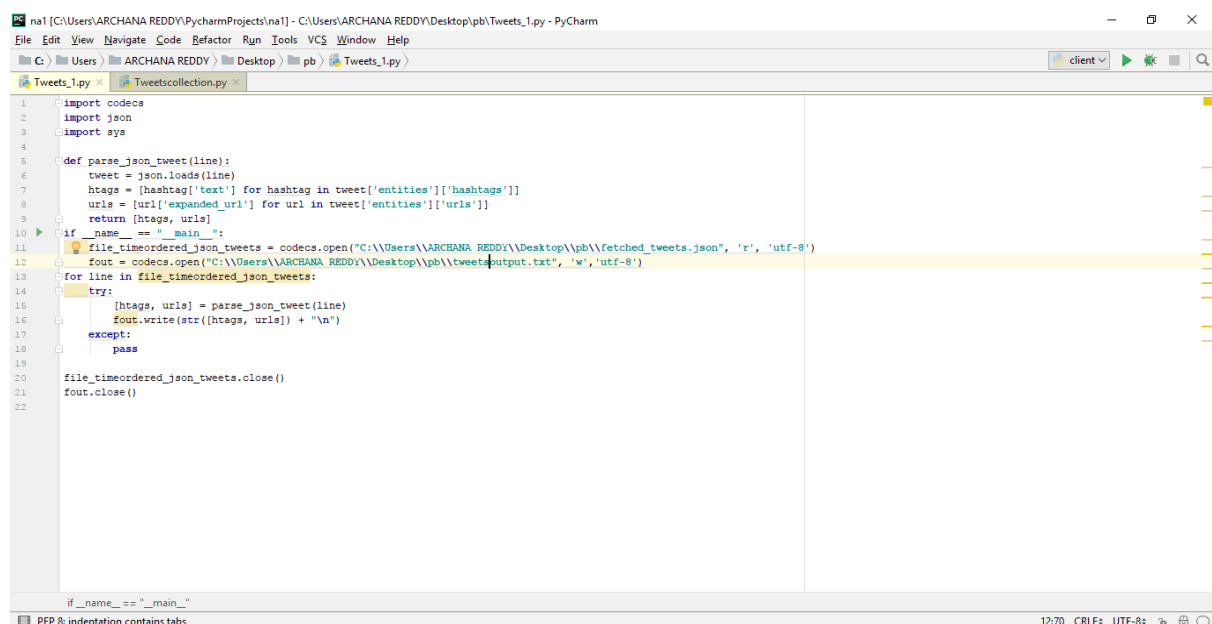
- **Operating System: Ubuntu.**
- **Software tools: Hadoop, Spark.**
- **Coding Language: Python 3.6.2.**

1. Tweets Collection Code.



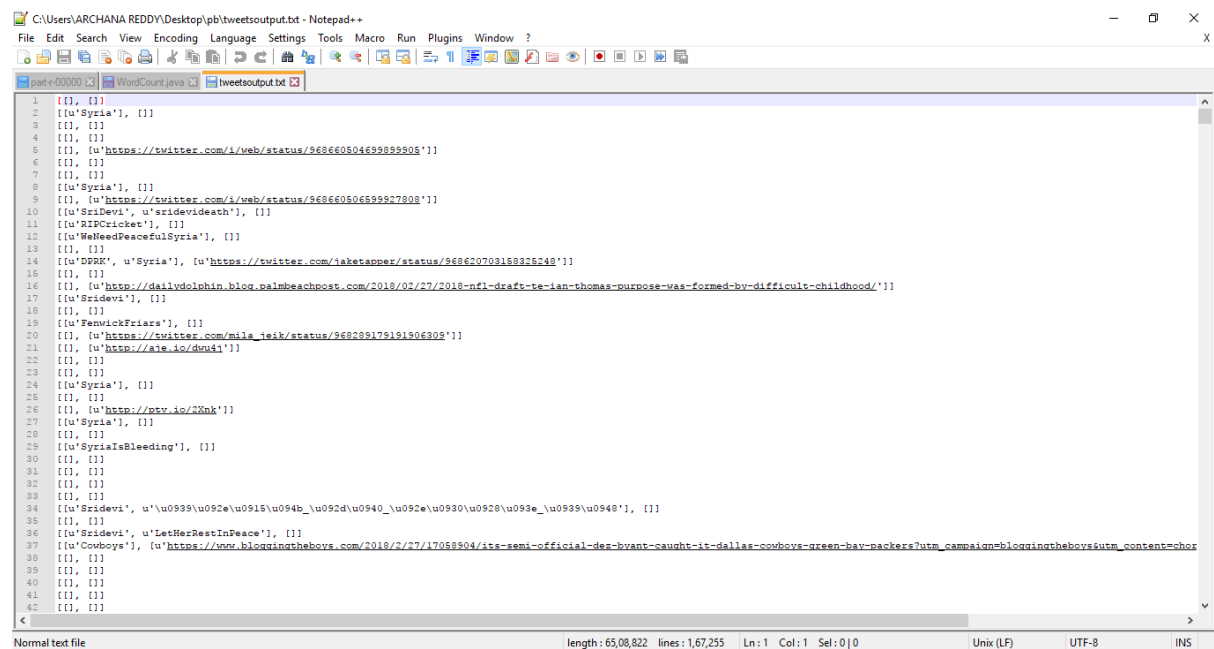
```
1 import sys
2 from tweepy.streaming import StreamListener
3 from tweepy import OAuthHandler
4 from tweepy import Stream
5
6 access_token = "968520972650283008-dreBttIMUaPgJapeMm5R6HsKGF54u"
7 access_token_secret = "0t1WNoF22k1QuFNMFPamyptTpF907QRy9tubJ5fgNbiyJ"
8 consumer_key = "bsaQpOrPD0jbxG3isqo84ZSKB"
9 consumer_secret = "p9VWd6bhktrHng40Kya409eiuJVXlyEvpRgyf8XzqgUaM6e"
10
11
12 class StdOutListener(StreamListener):
13     def on_data(self, data):
14         print(data)
15         with open('fetched_tweets.json', 'a') as tf:
16             tf.write(data)
17         return True
18
19
20 if __name__ == '__main__':
21     l = StdOutListener()
22     auth = OAuthHandler(consumer_key, consumer_secret)
23     auth.set_access_token(access_token, access_token_secret)
24     stream = Stream(auth, l)
25
26     stream.filter(track=['floridashooting', 'RIP', 'Sridevi', 'Syria', 'Nationalpancakeday', 'NFL', 'basketball'])
27
28
29
30
```

2. Tweets Extraction of hashtags and url's.



```
1 import codecs
2 import json
3 import sys
4
5 def parse_json_tweet(line):
6     tweet = json.loads(line)
7     htags = [hashtag['text'] for hashtag in tweet['entities']['hashtags']]
8     urls = [url['expanded_url'] for url in tweet['entities']['urls']]
9     return [htags, urls]
10
11 if __name__ == '__main__':
12     file_timeordered_json_tweets = codecs.open("C:\\Users\\ARCHANA REDDY\\Desktop\\pb\\fetched_tweets.json", 'r', 'utf-8')
13     fout = codecs.open("C:\\Users\\ARCHANA REDDY\\Desktop\\pb\\tweets_output.txt", 'w', 'utf-8')
14     for line in file_timeordered_json_tweets:
15         try:
16             [htags, urls] = parse_json_tweet(line)
17             fout.write(str([htags, urls]) + "\n")
18         except:
19             pass
20     file_timeordered_json_tweets.close()
21     fout.close()
22
23
24 if __name__ == '__main__':
25
26
```

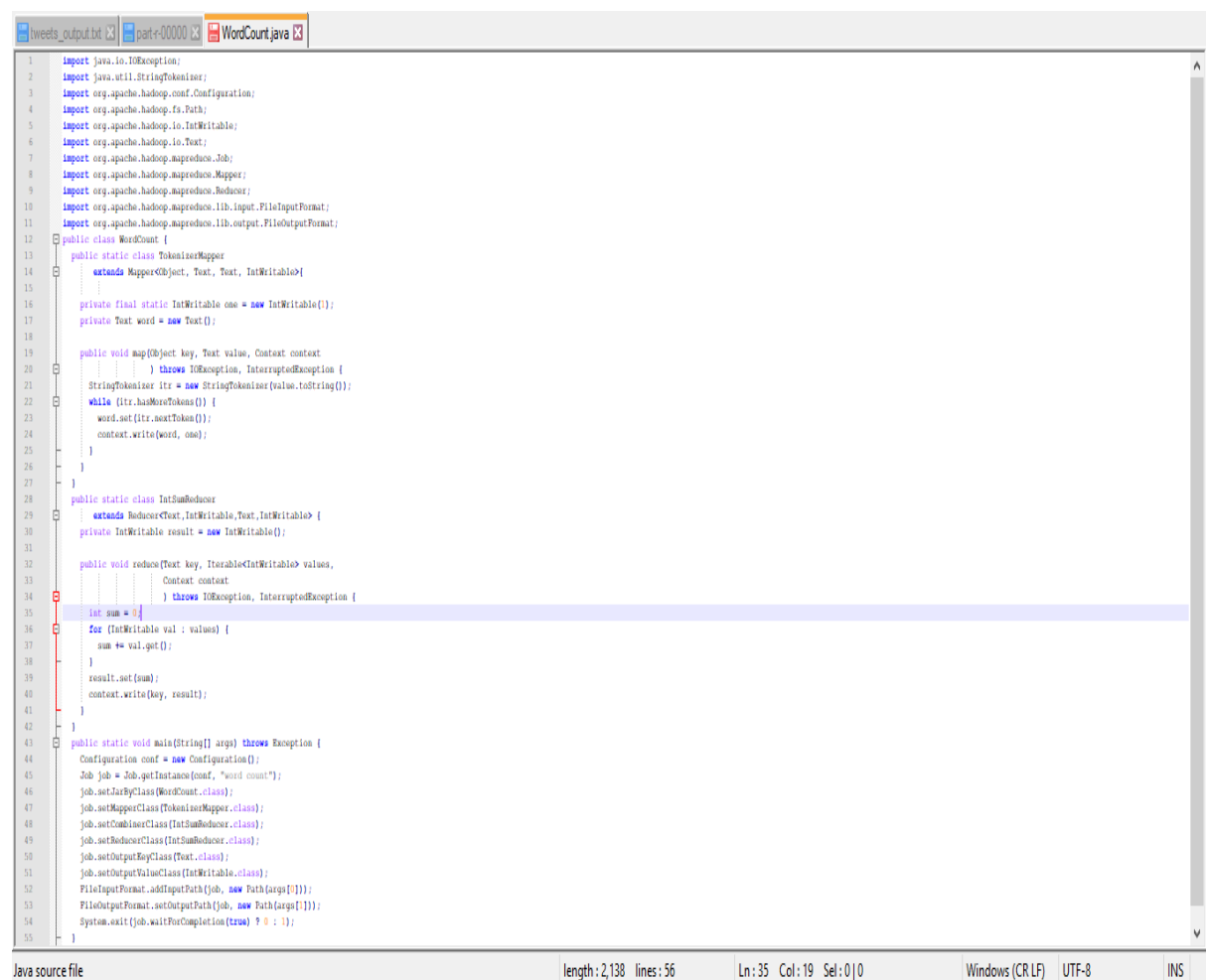
3. Tweets Output after extraction



```
1 [{"id": 1, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
2 [{"id": 2, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
3 [{"id": 3, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
4 [{"id": 4, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
5 [{"id": 5, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
6 [{"id": 6, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
7 [{"id": 7, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
8 [{"id": 8, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
9 [{"id": 9, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
10 [{"id": 10, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
11 [{"id": 11, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
12 [{"id": 12, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
13 [{"id": 13, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
14 [{"id": 14, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
15 [{"id": 15, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
16 [{"id": 16, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
17 [{"id": 17, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
18 [{"id": 18, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
19 [{"id": 19, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
20 [{"id": 20, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
21 [{"id": 21, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
22 [{"id": 22, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
23 [{"id": 23, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
24 [{"id": 24, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
25 [{"id": 25, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
26 [{"id": 26, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
27 [{"id": 27, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
28 [{"id": 28, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
29 [{"id": 29, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
30 [{"id": 30, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
31 [{"id": 31, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
32 [{"id": 32, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
33 [{"id": 33, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
34 [{"id": 34, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
35 [{"id": 35, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
36 [{"id": 36, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
37 [{"id": 37, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
38 [{"id": 38, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
39 [{"id": 39, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
40 [{"id": 40, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
41 [{"id": 41, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
42 [{"id": 42, ["Syria"], [{"url": "https://twitter.com/1/web/status/968660504698989905"}]}]
```

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



```
1 import java.io.IOException;
2 import java.util.StringTokenizer;
3 import org.apache.hadoop.conf.Configuration;
4 import org.apache.hadoop.fs.Path;
5 import org.apache.hadoop.io.IntWritable;
6 import org.apache.hadoop.io.Text;
7 import org.apache.hadoop.mapreduce.Job;
8 import org.apache.hadoop.mapreduce.Mapper;
9 import org.apache.hadoop.mapreduce.Reducer;
10 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
11 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
12 public class WordCount {
13     public static class TokenizerMapper
14         extends Mapper<Object, Text, Text, IntWritable> {
15
16         private final static IntWritable one = new IntWritable(1);
17         private Text word = new Text();
18
19         public void map(Object key, Text value, Context context
20             ) throws IOException, InterruptedException {
21             StringTokenizer itr = new StringTokenizer(value.toString());
22             while (itr.hasMoreTokens()) {
23                 word.set(itr.nextToken());
24                 context.write(word, one);
25             }
26         }
27     }
28
29     public static class IntSumReducer
30         extends Reducer<Text, IntWritable, Text, IntWritable> {
31         private IntWritable result = new IntWritable();
32
33         public void reduce(Text key, Iterable<IntWritable> values,
34             Context context
35             ) throws IOException, InterruptedException {
36             int sum = 0;
37             for (IntWritable val : values) {
38                 sum += val.get();
39             }
40             result.set(sum);
41             context.write(key, result);
42         }
43     }
44
45     public static void main(String[] args) throws Exception {
46         Configuration conf = new Configuration();
47         Job job = Job.getInstance(conf, "word count");
48         job.setJarByClass(WordCount.class);
49         job.setMapperClass(TokenizerMapper.class);
50         job.setReducerClass(IntSumReducer.class);
51         job.setOutputKeyClass(Text.class);
52         job.setOutputValueClass(IntWritable.class);
53         FileInputFormat.setInputPaths(job, new Path(args[0]));
54         FileOutputFormat.setOutputPath(job, new Path(args[1]));
55         System.exit(job.waitForCompletion(true) ? 0 : 1);
56     }
57 }
```

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

5.Hadoop Outputs

Browsing HDFS

localhost:50070/explorer.html#/archana

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

Browse Directory

/archana Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	2/28/2018, 1:10:05 PM	0	0 B	input

Hadoop, 2017.

Browsing HDFS

localhost:50070/explorer.html#/archana/input

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

Browse Directory

/archana/input Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	2/28/2018, 1:10:23 PM	0	0 B	output
-rw-r--r--	hduser	supergroup	5.89 MB	2/28/2018, 1:09:03 PM	1	128 MB	tweetsoutput.txt

Hadoop, 2017.

Browsing HDFS

localhost:50070/explorer.html#/archana/input/output

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

Browse Directory

/archana/input/output Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	supergroup	0 B	2/28/2018, 1:10:23 PM	1	128 MB	_SUCCESS
-rw-r--r--	hduser	supergroup	2.06 MB	2/28/2018, 1:10:23 PM	1	128 MB	part-r-00000

Hadoop, 2017.

6.Hadoop part-r-00000

```
C:\Users\ARCHANA REDDY\Desktop\pb\part-r-00000 - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
part-r-00000 WordCount.java tweetsoutput.txt
1 '11Veto', 2
2 '12s', 1
3 '12chman', 1
4 '14STR8', 1
5 '148straight', 1
6 '148straight', 5
7 '17Premier', 1
8 '1960s', 5
9 '1A', 2
10 '1AppDay', 1
11 '1moretogo', 3
12 '113', 4
13 '27Feb', 10
14 '27Feb', 15
15 '20Days', 1
16 '28Feb1997', 1
17 '28daysoflove', 1
18 '2A', 4
19 '2A', 1
20 '2AmericanRevolution', 20
21 '2hockey', 1
22 '2hockey', 5
23 '2hoursvast', 1
24 '2k', 1
25 '2ndODI', 2
26 '2ndShooon', 20
27 '30Seconds', 1
28 '3DAnimation', 10
29 '3Feb1984', 2
30 '3idions', 2
31 '38sculpture', 1
32 '3x3', 9
33 '3x3', 2
34 '40minutesofhell', 1
35 '49ers', 1
36 '49ersInvasion', 2
37 '49ersOnNBCS', 2
38 '618Boyschoops', 1
39 '68United', 1
40 '6on5', 24
41 '76ers', 1
42 '7AAA', 1
43 '80s', 2
```

7.Spark word count program

```
kopps@ubuntu: /usr/lib/spark/spark-2.1.0-bin-hadoop2.7/bin
18/03/05 15:42:09 DEBUG BlockManagerMaster: Updated info of block taskresult_5
18/03/05 15:42:09 DEBUG BlockManager: Told master about block taskresult_5
18/03/05 15:42:09 DEBUG BlockManager: Put block taskresult_5 locally took 6 ms
18/03/05 15:42:09 INFO Executor: Finished task 0.0 in stage 2.0 (TID 5). 1348936 bytes result sent via BlockManager)
18/03/05 15:42:09 DEBUG TaskSchedulerImpl: parentName: , name: TaskSet 2.0, runningTasks: 0
18/03/05 15:42:09 DEBUG TaskResultGetter: Fetching indirect task result for TID 5
18/03/05 15:42:09 DEBUG BlockManager: Getting remote block taskresult_5
18/03/05 15:42:09 DEBUG BlockManager: Getting remote block taskresult_5 from BlockManagerId(driver, 192.168.157.131, 46133, None)
18/03/05 15:42:09 INFO TransportClientFactory: Successfully created connection to /192.168.157.131:46133
18/03/05 15:42:09 DEBUG AbstractByteBuf: -Dio.netty.buffer.bytebuf.checkAccessible: true
18/03/05 15:42:09 DEBUG ResourceLeakDetector: -Dio.netty.leakDetection.level: simple
18/03/05 15:42:09 DEBUG ResourceLeakDetector: -Dio.netty.leakDetection.maxRecords: 4
18/03/05 15:42:09 DEBUG ResourceLeakDetectorFactory: Loaded default ResourceLeakDetector: io.netty.util.ResourceLeakDetector$2faa8a95
18/03/05 15:42:09 DEBUG TransportClientFactory: Connection to /192.168.157.131:46133 successful, running bootstraps...
18/03/05 15:42:09 INFO TransportClientFactory: Successfully created connection to /192.168.157.131:46133 after 111 ms (0 ms spent in bootstraps)
18/03/05 15:42:09 DEBUG Recycler: -Dio.netty.recycler.maxCapacity.default: 32768
18/03/05 15:42:09 DEBUG Recycler: -Dio.netty.recycler.maxSharedCapacityFactor: 2
18/03/05 15:42:09 DEBUG Recycler: -Dio.netty.recycler.linkCapacity: 16
18/03/05 15:42:09 DEBUG Recycler: -Dio.netty.recycler.ratio: 8
18/03/05 15:42:09 DEBUG BlockManager: Getting local block taskresult_6 as bytes
18/03/05 15:42:09 DEBUG BlockManager: Level for block taskresult_6 is StorageLevel(disk, memory, 1 replicas)
18/03/05 15:42:09 DEBUG BlockManager: Getting local block taskresult_5 as bytes
18/03/05 15:42:09 DEBUG BlockManager: Level for block taskresult_5 is StorageLevel(disk, memory, 1 replicas)
18/03/05 15:42:09 DEBUG TransportClient: Sending fetch chunk request 0 to /192.168.157.131:46133
18/03/05 15:42:09 DEBUG TransportClient: Sending fetch chunk request 0 to /192.168.157.131:46133
18/03/05 15:42:09 DEBUG BlockManagerSlaveEndpoint: removing block taskresult_6
18/03/05 15:42:09 DEBUG BlockManager: Removing block taskresult_6
18/03/05 15:42:09 DEBUG MemoryStore: Block taskresult_6 of size 1361343 dropped from memory (free 382473612)
18/03/05 15:42:09 INFO BlockManagerInfo: Removed taskresult_6 on 192.168.157.131:46133 in memory (size: 1329.4 KB, free: 365.0 MB)
18/03/05 15:42:09 DEBUG BlockManagerMaster: Updated info of block taskresult_6
18/03/05 15:42:09 DEBUG BlockManager: Told master about block taskresult_6
18/03/05 15:42:09 DEBUG BlockManagerSlaveEndpoint: Done removing block taskresult_6, response is true
18/03/05 15:42:09 DEBUG BlockManagerSlaveEndpoint: Sent response: true to 192.168.157.131:34757
18/03/05 15:42:09 INFO TaskSetManager: Finished task 1.0 in stage 2.0 (TID 6) in 1247 ms on localhost (executor driver) (1/2)
18/03/05 15:42:09 DEBUG BlockManagerSlaveEndpoint: removing block taskresult_5
18/03/05 15:42:09 DEBUG BlockManager: Removing block taskresult_5
18/03/05 15:42:09 DEBUG MemoryStore: Block taskresult_5 of size 1348936 dropped from memory (free 383822548)
18/03/05 15:42:09 INFO BlockManagerInfo: Removed taskresult_5 on 192.168.157.131:46133 in memory (size: 1317.3 KB, free: 366.3 MB)
18/03/05 15:42:09 INFO DAGScheduler: ResultStage 2 (collect at <console>:29) finished in 1.254 s
18/03/05 15:42:09 DEBUG DAGScheduler: After removal of stage 2, remaining stages = 1
18/03/05 15:42:09 DEBUG BlockManagerMaster: Updated info of block taskresult_5
18/03/05 15:42:09 INFO DAGScheduler: After removal of stage 1, remaining stages = 0
18/03/05 15:42:09 INFO DAGScheduler: Job 1 finished: collect at <console>:29, took 3.319064 s
18/03/05 15:42:09 DEBUG BlockManager: Told master about block taskresult_5
18/03/05 15:42:09 DEBUG BlockManagerSlaveEndpoint: Done removing block taskresult_5, response is true
18/03/05 15:42:09 DEBUG BlockManagerSlaveEndpoint: Sent response: true to 192.168.157.131:34757
18/03/05 15:42:09 INFO TaskSetManager: Finished task 0.0 in stage 2.0 (TID 5) in 1254 ms on localhost (executor driver) (2/2)
18/03/05 15:42:09 INFO TaskSchedulerImpl: Removed TaskSet 2.0, whose tasks have all completed, from pool
res: Array[String, Int] = Array(("u/PaponControversy"), ([u/ShutUpAndDrTable"], 1), ("u/Kapoorfamily"), 1), ("u/https://dubalInformer.com/393814/sridevis-body-reaches-mumbai-airport-from-dubai-vo-news-dubai-video/"], 1), ("u/https://poo.g/8b/VqvqEB"], 1), ("u/https://twitter.com/thebigpig/status/968598483174338560"], 2), ("u/https://twitter.com/t/web/status/9686691988178948096"], 1), ("u/https://twitter.com/t/web/status/968706651384680450"], 1), ("u/https://twitter.com/fatqacandao/status/968670429660430336"], 1), ("u/likeforfollow", 1), ([u/Narendramod"], 2), ("u/http://www.bbc.co....
scala> counts.saveAsTextFile("/home/kopps/Desktop/sparkoutput")
```

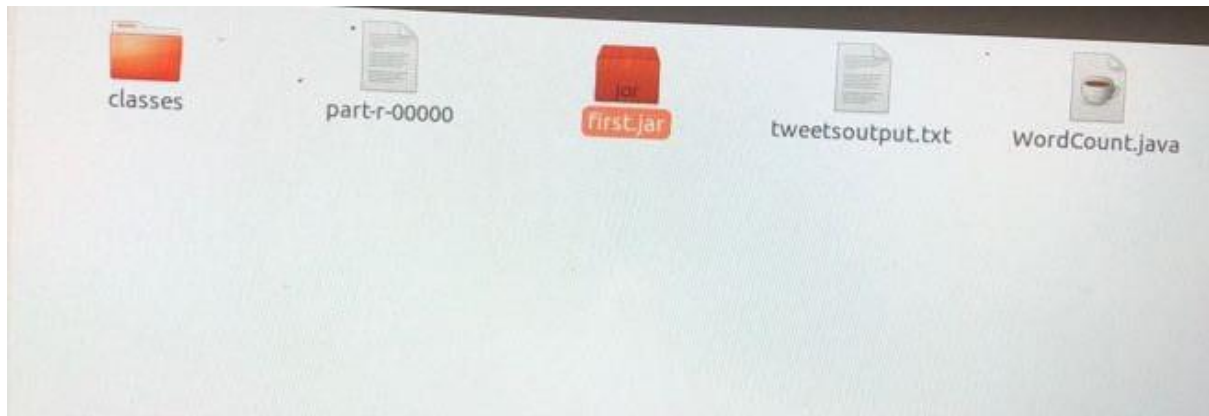
8. Spark Part-r-00000

```
C:\Users\ARCHANA REDDY\Desktop\pb\output_spark\part-00000 - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
part-00000
1 ([{"https://twitter.com/DaRealDom15/status/968676471167024286"}],1)
2 ([{"https://twitter.com/4/web/status/96871649556638662"}],1)
3 ([{"https://t.me/180MB1"}],1)
4 ([{"https://twitter.com/4/web/status/968697268802727680"}],1)
5 ([{"Tributespaid"}],1)
6 ([{"https://twitter.com/4/web/status/968694872682776852"}],1)
7 ([{"https://twitter.com/4/web/status/96867745501461504"}],1)
8 ([{"https://www.cineulagam.com/celebs/06/18/17/11?ref=twitter-cineulagam"}],2)
9 ([{"https://news.quibooat.com/8476KVQ2"}],1)
10 ([{"https://twitter.com/4/web/status/968652653066895970"}],1)
11 ([{"RussianSanctions"}],1)
12 ([{"https://twitter.com/4/web/status/968677838954968682"}],1)
13 ([{"https://twitter.com/4/web/status/968678644783738886"}],1)
14 ([{"https://twitter.com/4/web/status/96867641668955536"}],1)
15 ([{"https://twitter.com/snow/status/96867764895404552"}],1,140)
16 ([{"https://twitter.com/4/web/status/96868793935210457"}],1)
17 ([{"Bilismafia"}],1)
18 ([{"u'realtalk"}],1)
19 ([{"https://twitter.com/4/web/status/968686511352668160"}],1)
20 ([{"https://twitter.com/4/web/status/96870608840381263"}],1)
21 ([{"https://twitter.com/4/web/status/96867978378466784"}],1)
22 ([{"https://live.cricbet.com.au/#/2029/42131/overview"}],3)
23 ([{"u'pumpeddoesntevencoverit"}],1)
24 ([{"https://flip.it/2oagm"}],1)
25 ([{"https://twitter.com/4/web/status/968691578256731648"}],1)
26 ([{"u'BiggerThanBasketball"}],1)
27 ([{"https://twitter.com/darealval1x/status/96870317068616704"}],1)
28 ([{"u'pecep"}],1)
29 ([{"https://twitter.com/4/web/status/96870006951671488"}],1)
30 ([{"https://blogs.timesofindia.indiatimes.com/Q-gone/vhat-killed-sridevi"}],1)
31 ([{"https://twitter.com/4/web/status/968679188214642336"}],1)
32 ([{"https://punjabibollywoodadka.in/entertainment/news/salman-khan-841018?newsid=96092ad0-6926-8427-4856-1641ca540be&sessionid=ad9274e5-1611-6b4b-748d-141e2d1704ee"}],1)
33 ([{"u'TheCame"}],1)
34 ([{"u'stopthebombing"}],1)
35 ([{"https://paper.li/f-1381900033?edition_id=62755d70-1c45-11e8-aae6-0cc47a0d1602"}],1)
36 ([{"https://twitter.com/BB_KyleP8arbez/status/968684501104758784"}],1)
37 ([{"https://www.sportslocalines.com/news/sports/high-school-sports/2019/02/28/mcquaid-unrep-advance-to-class-aa-boy-basketball-final"}],1)
38 ([{"https://news.quibooat.com/8477Kxh"}],1)
39 ([{"u'GoBills"}],1)
40 ([{"u'Tub"}],1)
41 ([{"https://twitter.com/4/web/status/968688534684164055"}],1)
42 ([{"https://dive.it/Q361m"}],1)
43 ([{"https://i.wp/2EDUx25"}],1)
Normal text file length: 11,40,950 lines: 20,784 Ln: 1 Col: 1 Sel: 0 | 0 Unix (LF) UTF-8 INS
```

9. Spark part-r-00001

```
C:\Users\ARCHANA REDDY\Desktop\pb\output_spark\part-00001 - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
part-00000
1 ([{"https://twitter.com/4/web/status/9686846182701574141"}],1)
2 ([{"https://twitter.com/4/web/status/968688392650245442"}],1)
3 ([{"https://twitter.com/4/web/status/968681252147450810"}],1)
4 ([{"u'thefuckwebeensaying"}],1)
5 ([{"u'ABC"}],1)
6 ([{"https://www.axios.com/un-syria-cease-fire-hobbled-by-flare-1619744762-479e85b0-ale2-48a0-b896-67c62fe73e.html?utm_source=twitter&utm_medium=socialshare&utm_campaign=organic"}],3)
7 ([{"https://fb.me/Rb31u8s"}],1)
8 ([{"u'Yuh"}],2)
9 ([{"https://twitter.com/4/web/status/968714816581434362"}],1)
10 ([{"u'syup"}],1)
11 ([{"u'8StateDept"}],2)
12 ([{"https://twitter.com/CoachJonBeck/status/968155813104713733"}],1)
13 ([{"https://paper.li/theCricketsTimes?edition_id=b18a4a50-1c92-11e8-82c9-002590a56a2d"}],1)
14 ([{"u'WCAAd"}],1)
15 ([{"https://twitter.com/4/web/status/968680648285278208"}],1)
16 ([{"u'Omara"}],1)
17 ([{"https://twitter.com/rpxitt/status/96874634760630272"}],1)
18 ([{"https://twitter.com/4/web/status/968672130762880032"}],1)
19 ([{"https://twitter.com/NW8_hhletics/status/96864529758489441"}],2)
20 ([{"u'InPics"}],1)
21 ([{"https://ref.g1/r4Rta1KE"}],1)
22 ([{"https://www.southbase.com/article/165262828916503577?language=english&user_id=6487729753851306741&language=english&region=us&app_id=1106&impr_id=6527427691400923401&cid=65262828916503577"}],20)
23 ([{"https://gmb.co.uk/PAK58"}],1)
24 ([{"https://d1d1s8r8mH"}],1)
25 ([{"https://twitter.com/mitchneiffer/status/482626630855061504"}],1)
26 ([{"https://twib.in/1/dBn64py746Kv"}],1)
27 ([{"https://twitter.com/maukiclon9/status/968689328008487728"}],1)
28 ([{"https://twitter.com/4/web/status/96870338473924668"}],1)
29 ([{"https://twitter.com/4/web/status/96867378622586080"}],1)
30 ([{"u'realestate"}],1)
31 ([{"u'ABPNews"}],2)
32 ([{"https://www.theKeralatimes.com/2019/02/final-tourney-of-sridevi.html#.WpY2wTxbho.twitter"}],1)
33 ([{"https://twitter.com/4/web/status/968685300749873152"}],1)
34 ([{"https://www.bcci.tv/womens-cricket-under-23-sonal-league-one-day-limited-overs-2017-18/match/48"}],1)
35 ([{"u'garvsekhohmchorhain"}],1)
36 ([{"https://twitter.com/RueshBonnie/status/967921651034343490"}],1)
37 ([{"https://twitter.com/4/web/status/96868601270481808"}],1)
38 ([{"u'WebSite"}],35)
39 ([{"https://cnn.it/2F9QT4"}],2)
40 ([{"https://twitter.com/4/web/status/968674869574848411"}],1)
41 ([{"https://www.sportscrick.com/bd/2018/02/27/media-release-dhaka-first-division-womens-cricket-league-2017-18-final-rescheduled"}],1)
42 ([{"https://twitter.com/timesnow/status/968408907931533312"}],1)
Normal text file length: 11,53,746 lines: 20,726 Ln: 1 Col: 1 Sel: 0 | 0 Unix (LF) UTF-8 INS
```

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:

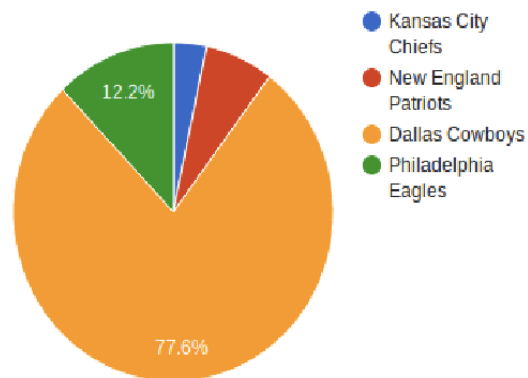
```
1
2
3 val post=sqlContext.sql("select user.name,text from TweetTable where text like '%Kansas City chiefs%' ")
4 // val post=sqlContext.sql("select count(user.name),count(text) from TweetTable where text like '%Dallas
   CowBoys%' ")
5 //val post=sqlContext.sql("select user.name,text from TweetTable where text like '%New England patriots%' ")
6 //val post=sqlContext.sql("select user.name,text from TweetTable where text like '%Philadelphia Eagles%' ")
7 //      post.show()
8 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("outputfootballeagles")
9 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("outputfootballCowboys")
10 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("outputfootballChiefs")
11 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("outputfootballPatriots")
12
13
```

DESCRIPTION OF QUERY 1:

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

VISUALIZATION OF QUERY 1:

number of users who tweeted on NFL teams



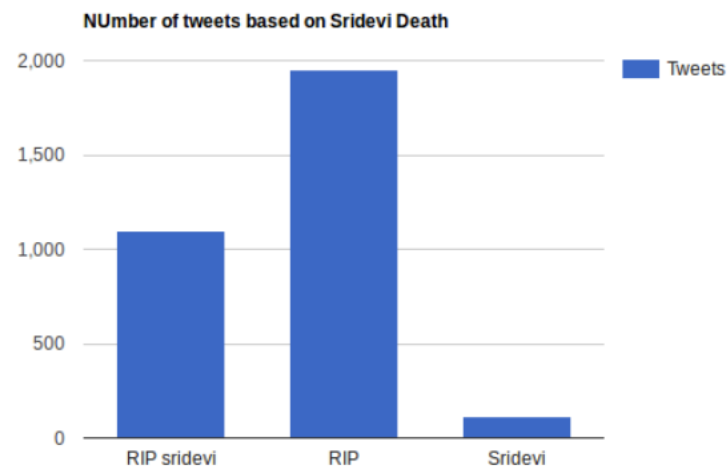
QUERY 2:

```
1
2
3 val post=sqlContext.sql("select count(user.name), count(text) from TweetTable where text like '%Sridevi%' ")
4
5 val post=sqlContext.sql("select count(user.name), count(text) from TweetTable where text like '%RIP%' ")
6
7 val post=sqlContext.sql("select count(user.name), count(text) from TweetTable where text like '%RIP Sridevi%' ")
8 post.show()
9
10 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("output6")
11
12 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("output7")
13 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:

```

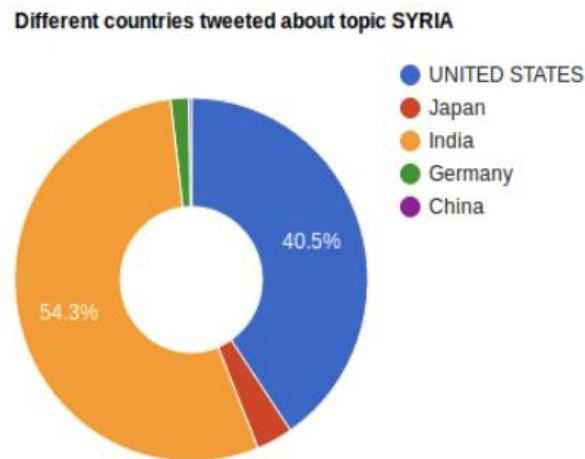
1
2
3 val post=sqlContext.sql("select user.name, SUBSTRING(user.created_at,5,3) AS month,SUBSTRING(user.created_at,
4 // val post=sqlContext.sql("select count(text), count(user.location) from TweetTable where text like '%Syria%'
5 and user.location = 'China'")
6 val post=sqlContext.sql("select count(text), count(user.location) from TweetTable where text like '%Syria%' and
7 user.location = 'India'")
8 val post=sqlContext.sql("select count(text), count(user.location) from TweetTable where text like '%Syria%' and
9 user.location = 'United States'")
10 val post=sqlContext.sql("select count(text), count(user.location) from TweetTable where text like '%Syria%' and
11 user.location = 'Japan'")post.show()
12 // post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Germany")
13
14 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("India")
15
16 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("China")
17
18 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Japan")
19
20 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:

```

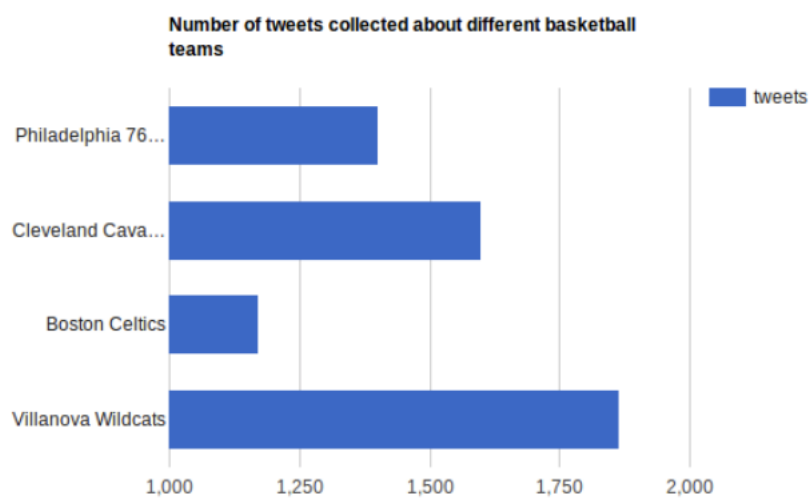
1 /val post=sqlContext.sql("select count(user.screen_name),count(text) from TweetTable where text like '%Villanova
  Wildcats%'")
2
3 val post=sqlContext.sql("select count(user.screen_name),count(text) from TweetTable where text like
  '%Philadelphia 76ers%'")
4
5 val post=sqlContext.sql("select count(user.screen_name),count(text) from TweetTable where text like '%Boston
  Celtics%'")
6 val post=sqlContext.sql("select count(user.screen_name),count(text) from TweetTable where text like '%Cleveland
  cavalaries%'")// post.show()
7 // post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("output4")
8
9 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("output5")
10
11 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("output6")
12
13 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("output7")
14
15 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("output8")
16 |
17

```

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



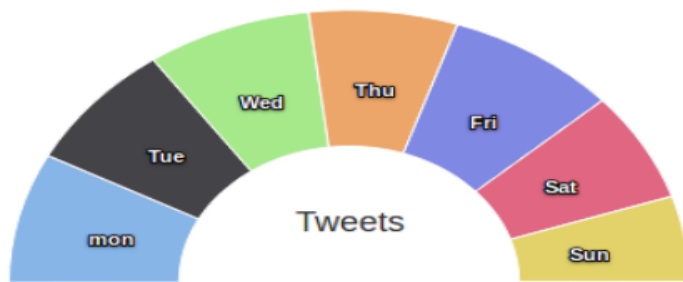
Query 5:

```
1 val post=sqlContext.sql("SELECT count(user.name),count(text) FROM TweetTable where text like '%basketball%' and
   user.created at like '%mon%' ")
2 post.show()
3 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Mon")
4 val post=sqlContext.sql("SELECT count(user.name),count(text) FROM TweetTable where text like '%basketball%' and
   user.created at like '%tue%' ")
5 post.show()
6 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Tue")
7 val post=sqlContext.sql("SELECT count(user.name),count(text) FROM TweetTable where text like '%basketball%' and
   user.created at like '%wed%' ")
8 post.show()
9 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Wed")
10 val post=sqlContext.sql("SELECT count(user.name),count(text) FROM TweetTable where text like '%basketball%' and
   user.created at like '%thu%' ")
11 post.show()
12 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Thu")
13 val post=sqlContext.sql("SELECT count(user.name),count(text) FROM TweetTable where text like '%basketball%' and
   user.created at like '%fri%' ")
14 post.show()
15 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Fri")
16 val post=sqlContext.sql("SELECT count(user.name),count(text) FROM TweetTable where text like '%basketball%' and
   user.created at like '%sat%' ")
17 post.show()
18 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Sat")
19 val post=sqlContext.sql("SELECT count(user.name),count(text) FROM TweetTable where text like '%basketball%' and
   user.created at like '%sun%' ")
20 post.show()
21 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Sun")
22
23
```

DESCRIPTION OF QUERY 5:

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

VISUALIZATION OF QUERY 5:



Query 6:

```
1
2 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")
3
4 post.show()
5 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("TestDemo")
6 |
```

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

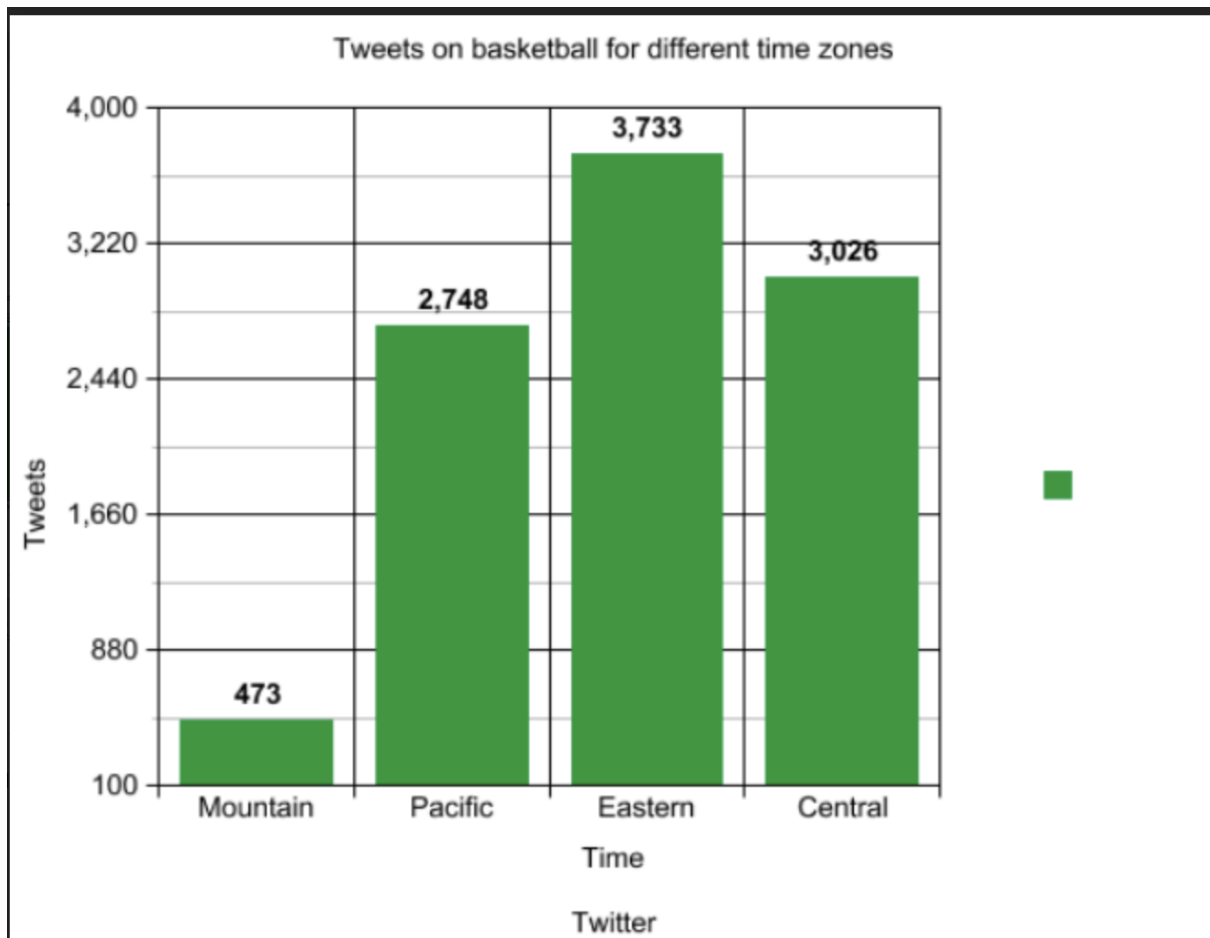
Query 7:

```
1
2 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%' ")
3 post.show()
4 post.map(x=> (x(0),x(1))).coalesce( numPartitions = 1, shuffle = true).saveAsTextFile( path = "testDemo12")
5
6 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%' ")
7 post.show()
8 post.map(x=> (x(0),x(1))).coalesce( numPartitions = 1, shuffle = true).saveAsTextFile( path = "testDemo1")
9
10 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%' ")
11 post.show()
12 post.map(x=> (x(0),x(1))).coalesce( numPartitions = 1, shuffle = true).saveAsTextFile( path = "testDemo1234")
13
14 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%' ")
15 post.show()
16 post.map(x=> (x(0),x(1))).coalesce( numPartitions = 1, shuffle = true).saveAsTextFile( path = "testDemo123")
17
```

DESCRIPTION OF QUERY 7:

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

VISUALIZATION OF QUERY 7:



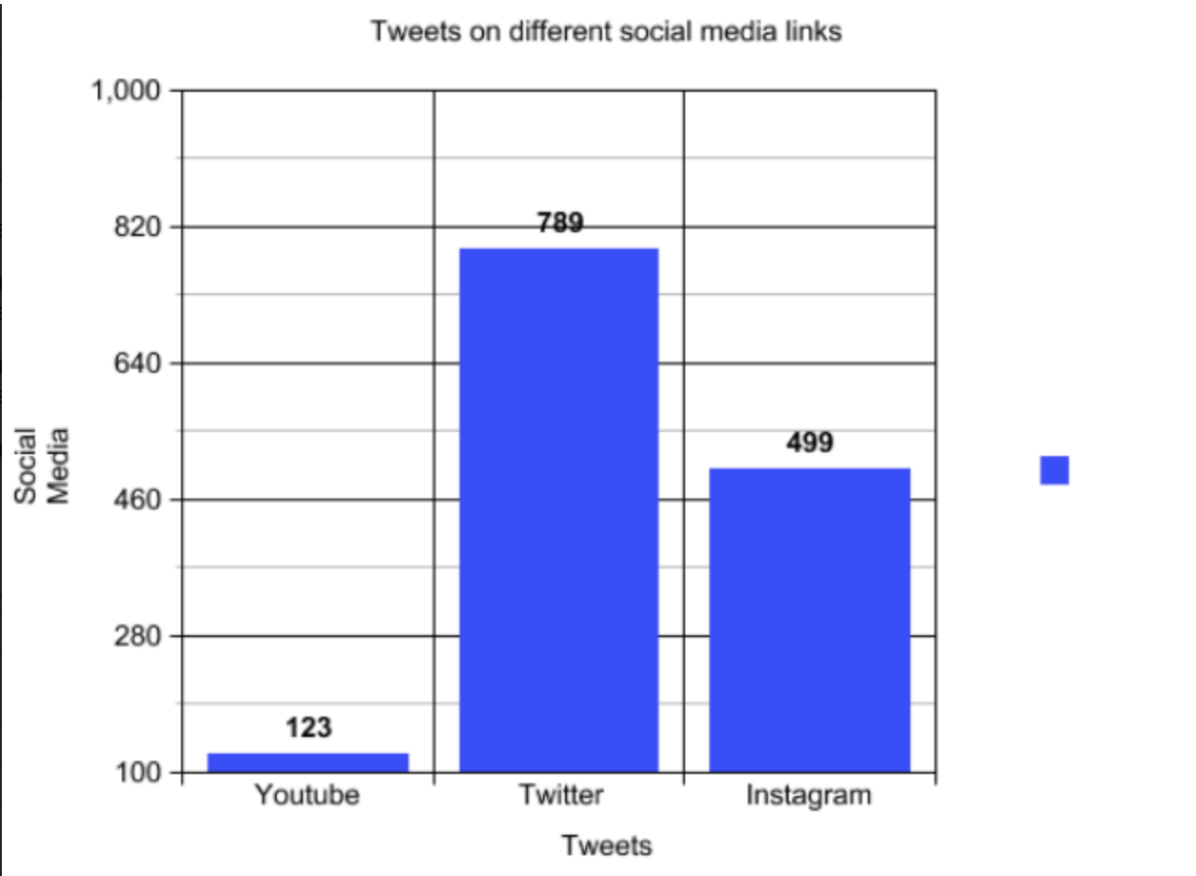
Query 8:

```
1
2 val post=sqlContext.sql( sqlText = "SELECT count (text), count(user.screen_name) FROM TweetTable WHERE
   user.description like '%youtu.b%' ")
3 post.show()
4 post.map(x=> (x(0),x(1))).coalesce( numPartitions = 1, shuffle = true).saveAsTextFile( path = "youtube")
5 val post=sqlContext.sql( sqlText = "SELECT count (text), count(user.screen_name) FROM TweetTable WHERE
   user.description like '%twitter%' ")
6 post.show()
7 post.map(x=> (x(0),x(1))).coalesce( numPartitions = 1, shuffle = true).saveAsTextFile( path = "Twitter")
8 val post=sqlContext.sql( sqlText = "SELECT count (text), count(user.screen_name) FROM TweetTable WHERE
   user.description like '%Instagram%' ")
9 post.show()
10 post.map(x=> (x(0),x(1))).coalesce( numPartitions = 1, shuffle = true).saveAsTextFile( path = "Instagram")
11
```

DESCRIPTION OF QUERY 8:

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

VISUALIZATION OF QUERY 8:



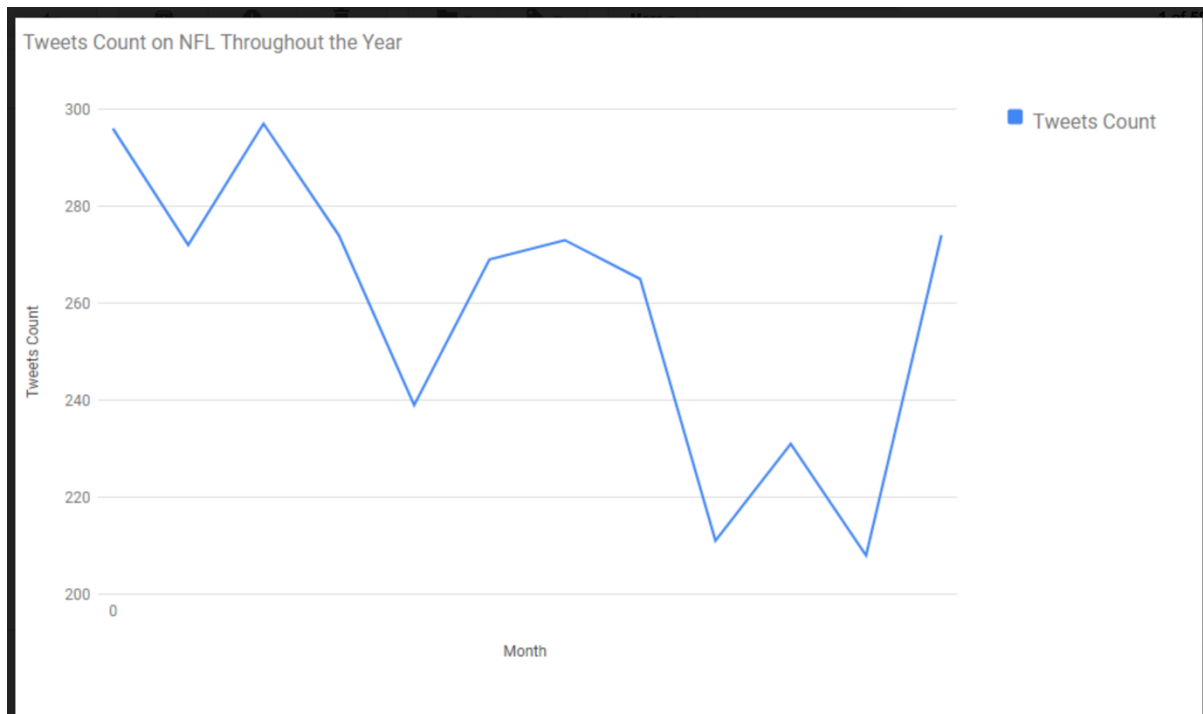
Query 9:

```
1  
2 val post=sqlContext.sql("SELECT count(text),count(user.screen_name) FROM TweetTable where user.created_at like  
   '%Mar%'and text like '%nfl%' ")  
3 post.show()  
4 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("March")  
5
```

DESCRIPTION OF QUERY 9:

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

VISUALIZATION OF QUERY 9:



Query 10:

```
1
2 val post=sqlContext.sql("select count(user.name), count(text) from TweetTable where text like
   '%FloridaShooting%' ")
3
4 post.show()
5 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Fs")
6 val post=sqlContext.sql("select count(user.name), count(text) from TweetTable where text like '%Florida%' ")
7
8 post.show()
9 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Fs1")
10 val post=sqlContext.sql("select count(user.name), count(text) from TweetTable where text like '%Shooting%' ")
11
12 post.show()
13 post.map(x=> (x(0),x(1))).coalesce(1,true).saveAsTextFile("Fs2")
14 |
```

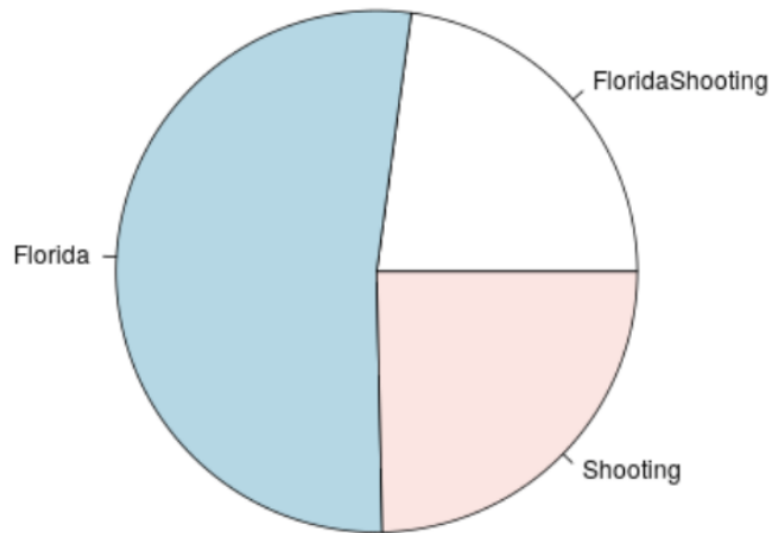
DESCRIPTION OF QUERY 10:

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

VISUALIZATION OF QUERY 10:

```
$Rscript main.r
```

```
png  
2
```























: