



**COLLECTING TWEETS USING TWITTER STREAMING API’S**

**Sub: PRINCIPLES OF BIG DATA**

|  |  |
| --- | --- |
| Avinash Banala | (16249987) |
| Sai Smaran Chinthala | (16251196) |
| Satya Venkata Ranga Janaki Sriram Mentey | (16251319) |

Instructor

Dr. PRAVEEN RAO, Ph.D.

# ABSTRACT

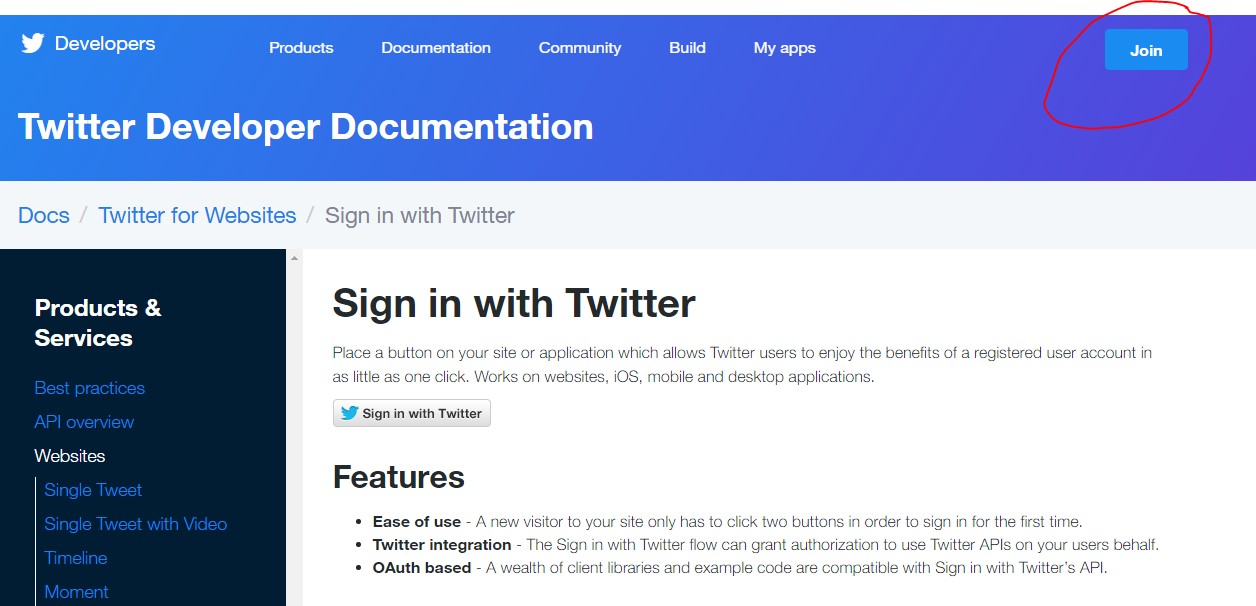
The principle point of this task is to build up a framework to analyze twitter data. Twitter is the long range interpersonal communication site which was brought together to mine the tweets from it and investigate the information utilizing Python. The hashtags utilized as a part of twitter are isolated into various indexes of HDFS (Hadoop Distributed File System).

In this trending period, there are billions of dynamic clients in interpersonal organizations, for example, Facebook, Twitter, Instagram, and so on. Mining information from such networks(Twitter) have many reasons and help in insights, commercials, arrange investigation or just interest.

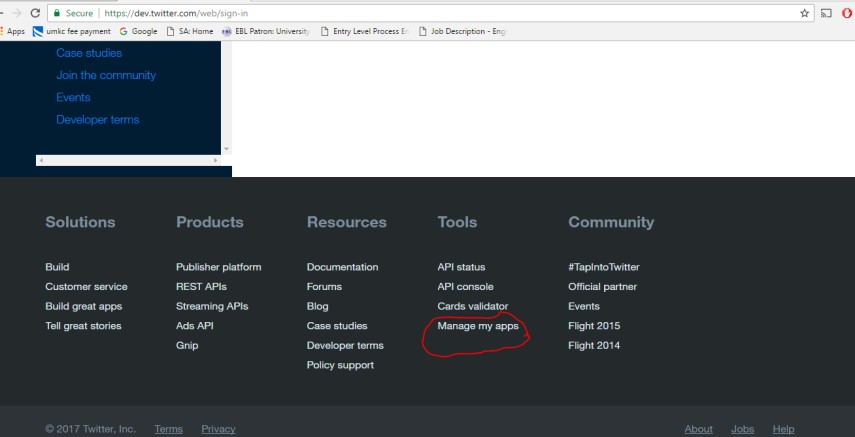
At first the tweets are gathered in JSON (JavaScript Object Notation) arrangement and afterward they are recorded independently. The most continuous 10 hashtags are each put away in a different "Hashtags index" in HDFS. The tweets without hashtag are put away in a "None Directory" and whatever remains of the tweets are put away in “Directory”.

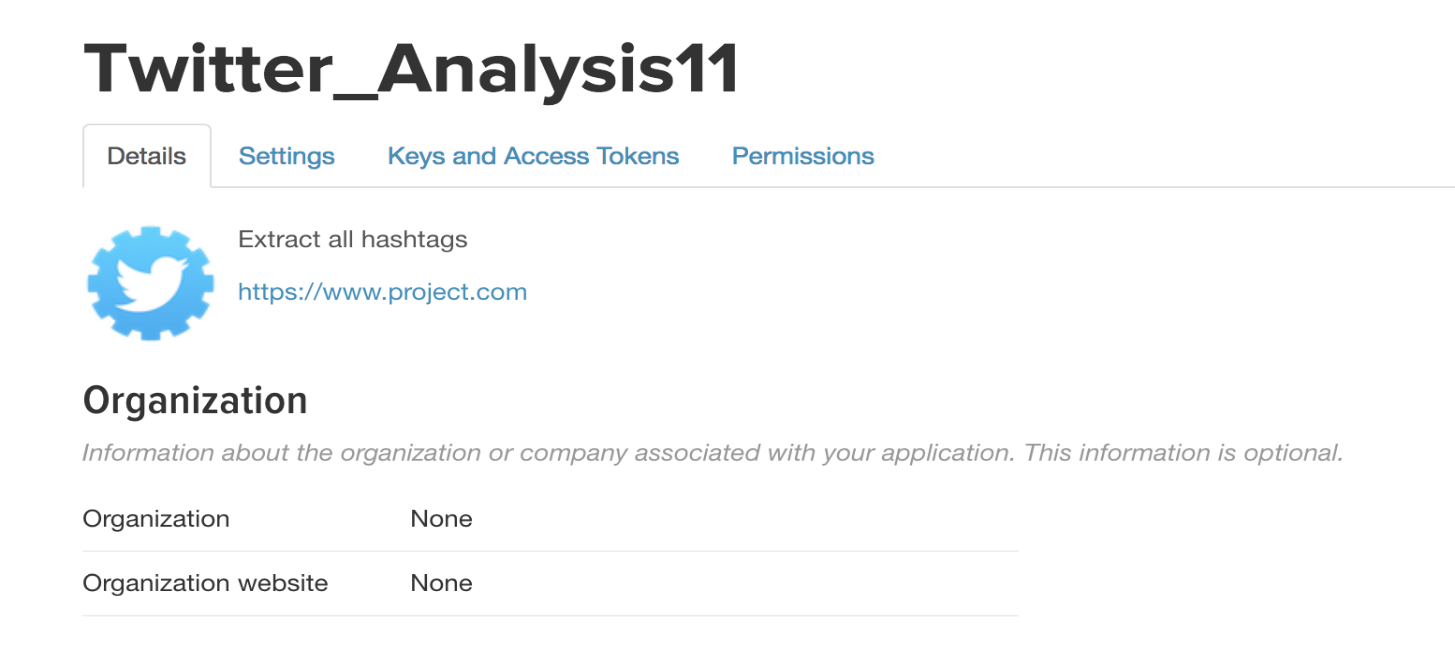
# TWITTER DEVELOPER SIGN UP FOR TWEETS

Before collection of tweets, it is important to sign in with twitter and join the documentation part with the following link :<https://dev.twitter.com/resources/signup>.

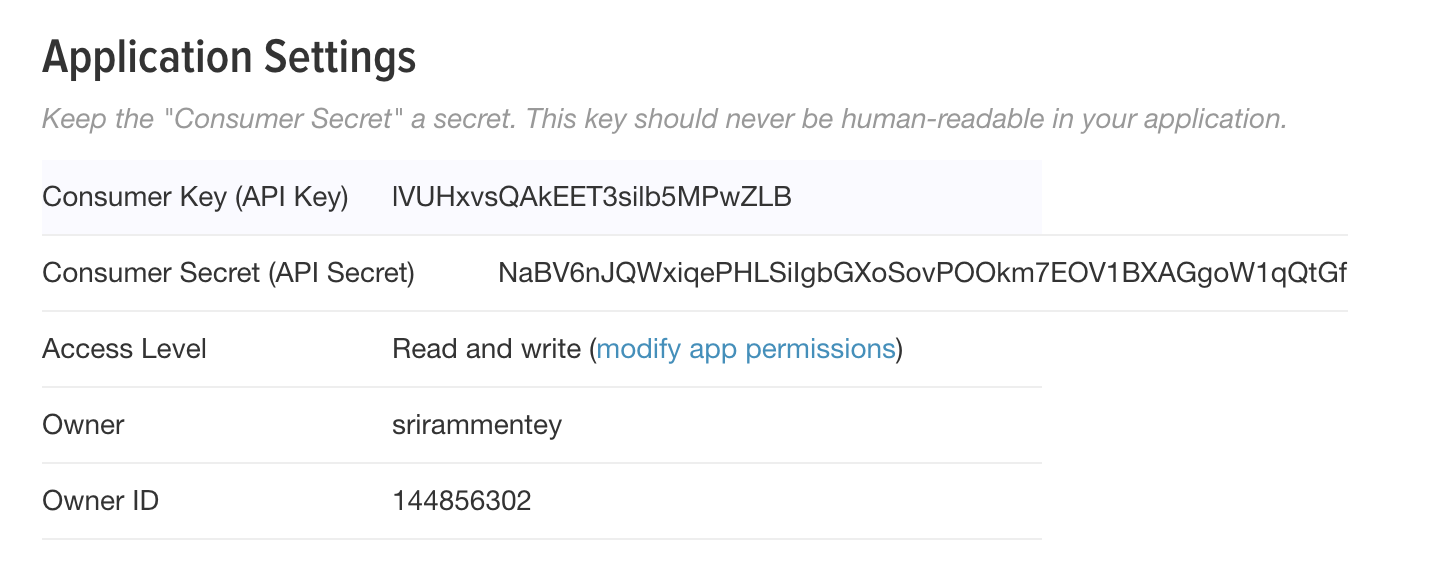


The account has been created. Then the application has been designed using the “Mange my apps” .This application generates 4 keys – Consumer key, Consumer secret key, Access token and Access token secret which are later used for the collection of tweets.

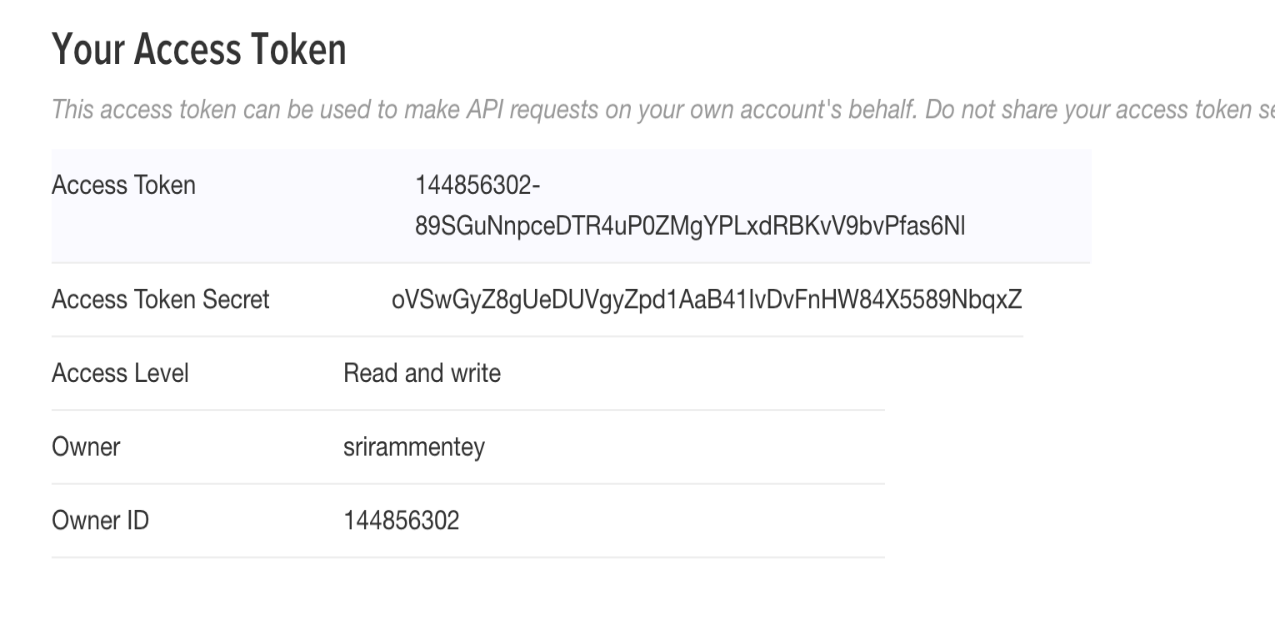




**CONSUMER API KEY:**



**ACCESS TOKEN KEY:**



The above screen shots illustrate the creation of a new application in Twitter Developer account and generation of the token keys.

Operating System: Ubuntu 16.04 Tools used: Hadoop 2.7.3 Scripting Language: Python 2

**EXTRACTING TWITTER TWEETS USING TWEEPY API**

**TWEEPY API**

The Tweepy API class provides access to the entire twitter RESTful API methods. Each method can accept various parameters and return responses. When we invoke an API method most of the time returned to us will be a Tweepy model class instance. This will contain the data returned from Twitter which we can then use inside our application.



Above is the code written in python which is used to collect the tweets from twitter which are trending with the word ‘HURRICANE’

**TWEETS SEPARATION WITH URLS AND HASHTAGS**

The tweets which are collected from the twitter are separated as per the category wise which is named as URLs and Hashtags using the below code.

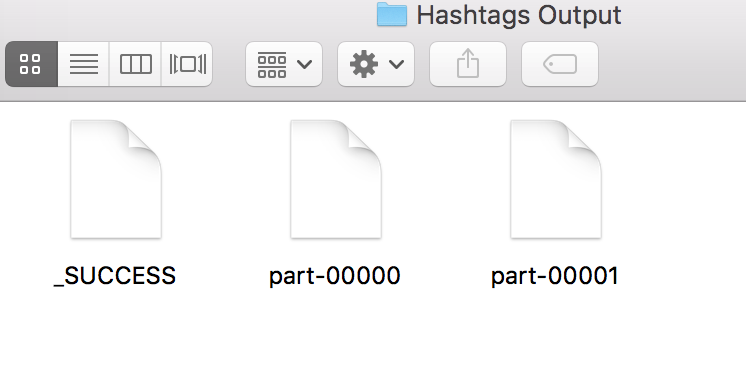


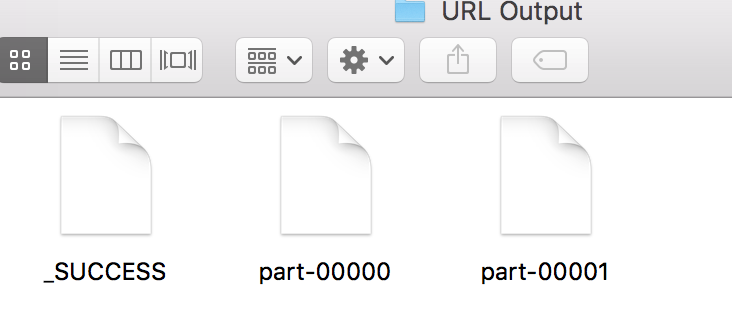
The above code is executed with the below commands and saved HASHTAGS in hashtag.txt and URLs in url.txt

**WORD COUNT USING APACHE SPARK**



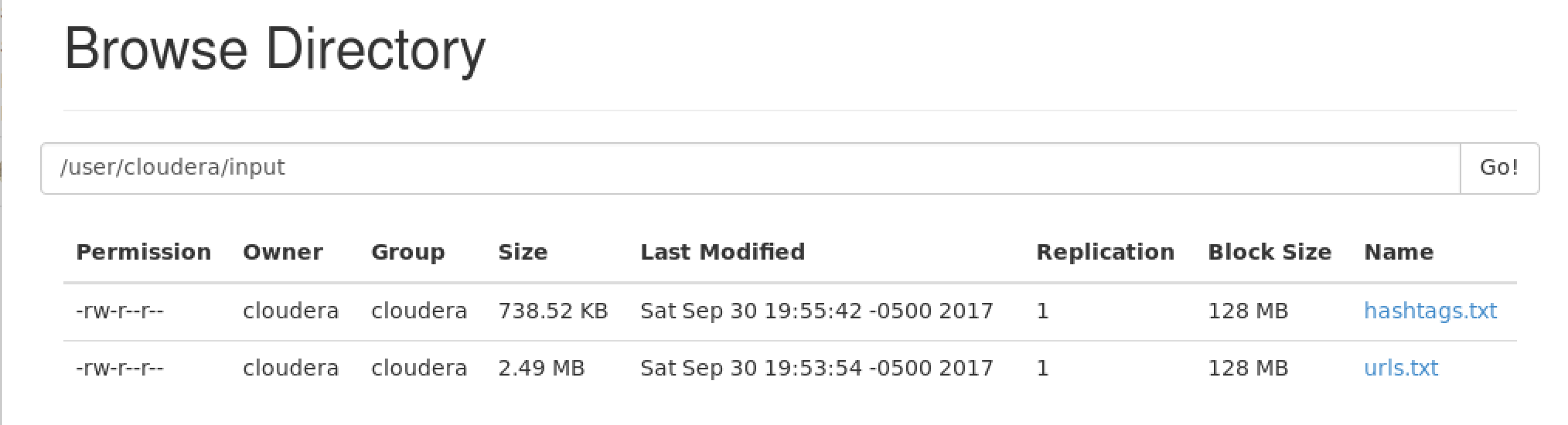
The outputs are saved in local directories which is mentioned below:



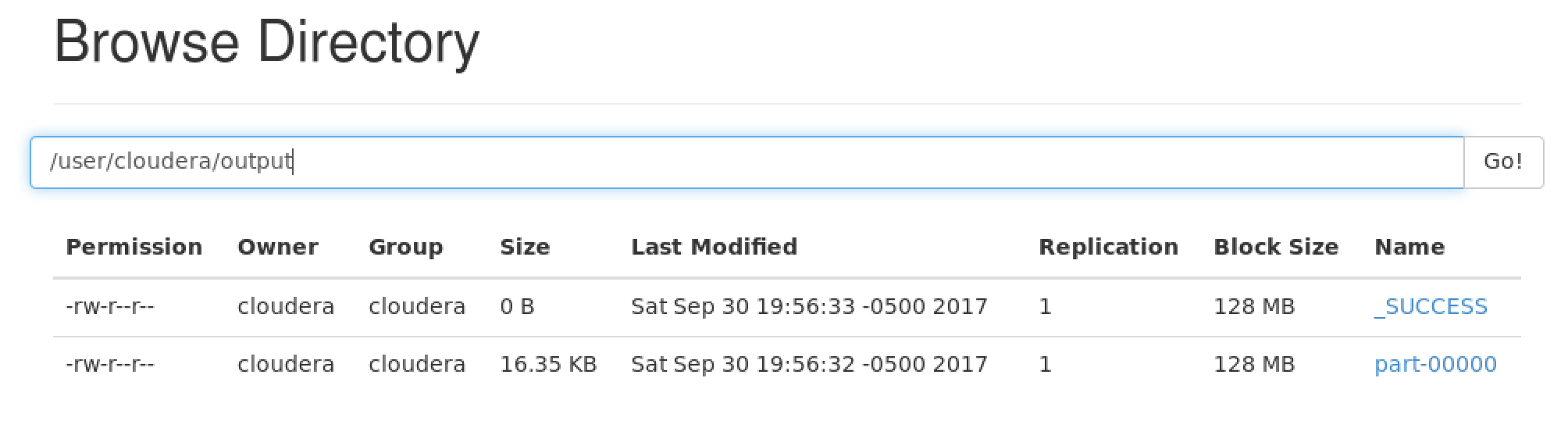


**WORD COUNT USING APACHE HADOOP**

**INPUT FILES:**

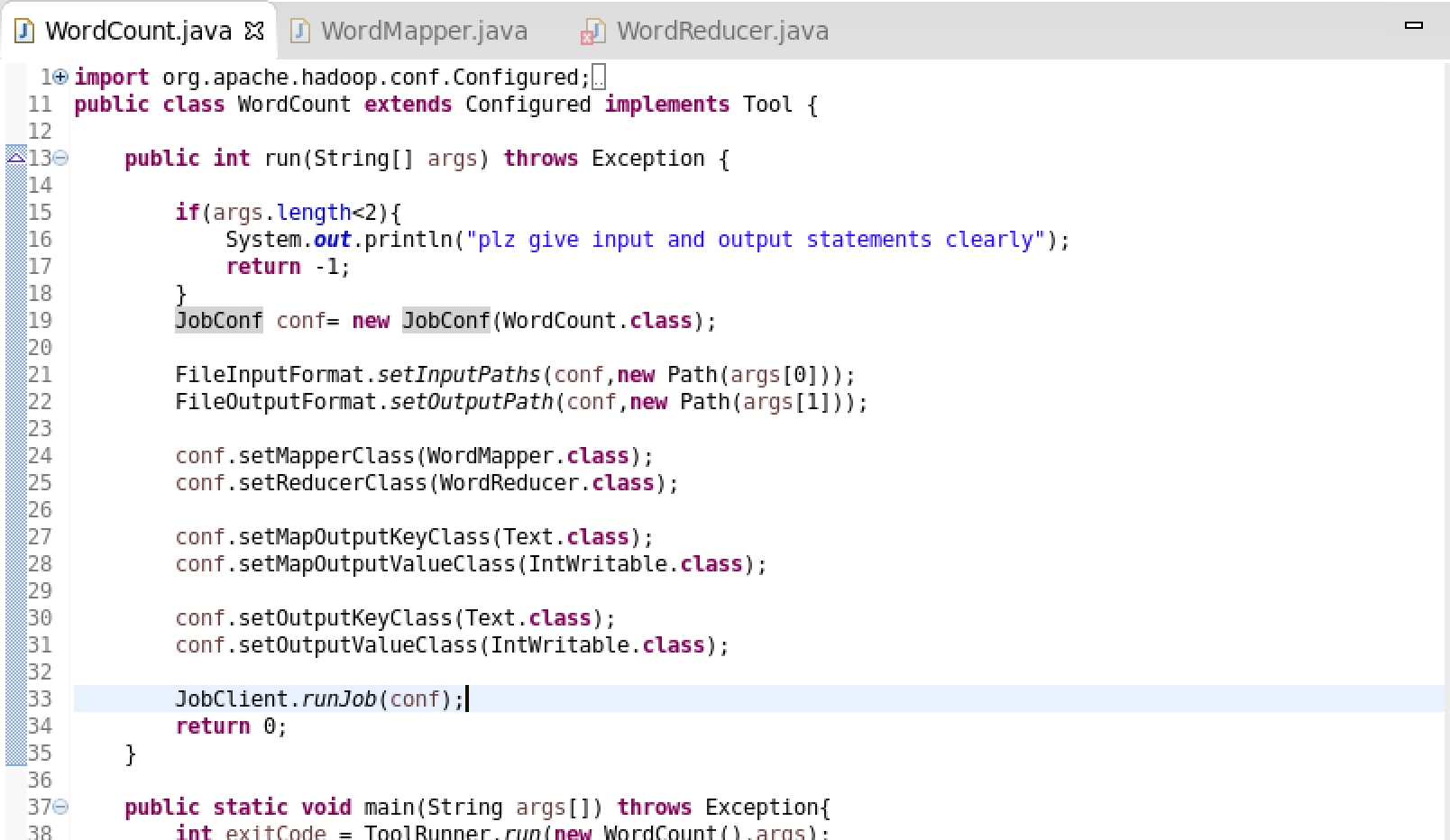
****

**WORD COUNT FOR HASHTAGS AND URLS:**

****

**CODE FOR WORD COUNT**

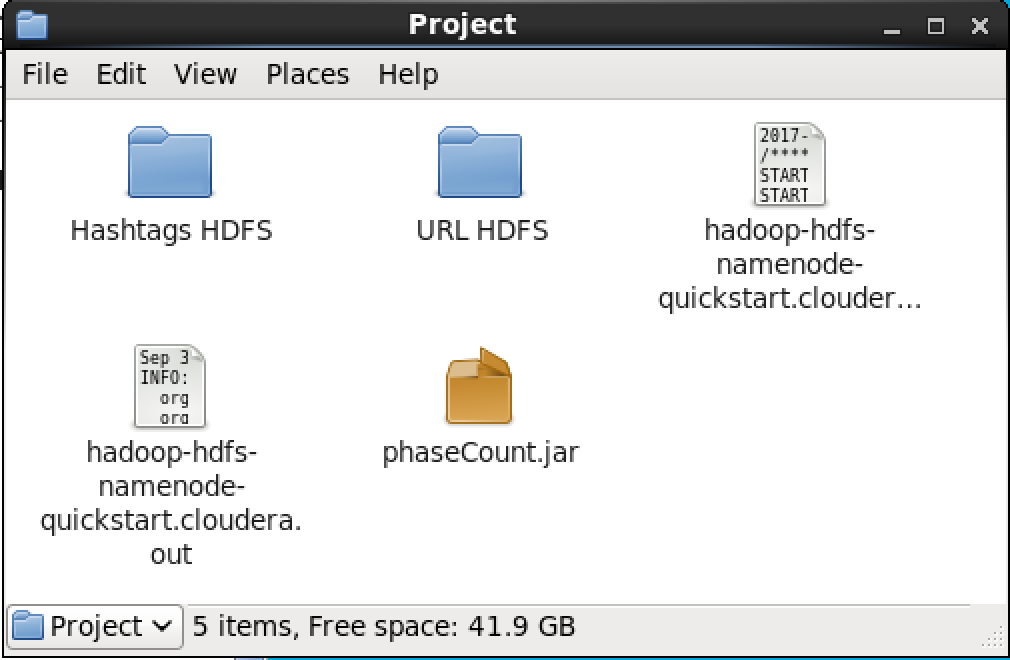
Below is the code which is written in java format.

.

**Log File and Output Files:**

**.**log and .output files are downloaded from the below links and stored in the local directory

****

****

**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

"HURRICANE" 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 “**HURRICANE**” 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.

**ANALYTICAL QUERIES:**

**FIRST QUERY:**

**SELECT \* FROM MainTable where lang='en’**

It is written for loading the tweet file. This query denotes the number of people who speaks English language.

**SECOND QUERY:**

**SELECT entities.hashtags.text AS ht FROM MainTable WHERE entities.hashtags.text IS NOT**

**NULL**

It is used to retrieve hashtags from the Lang\_Refiner Table. This query executes the entities which found

on the Main table where hashtags is not made null in its query.

**THIRD QUERY:**

**SELECT ht FROM Table2**

It acquires the data from the table 2 to split file into words. The ht denotes the hashtag in the above

designed query.

**FOURTH QUERY:**

**SELECT user.screen\_name AS u\_sn,count(\*) AS u\_count FROM Lang\_Refiner GROUP BY**

**user.screen\_name ORDER BY u\_count DESC LIMIT 10**

It displays the hashtag Data Frame. Lang\_Refiner denotes the query stored in it and represents the count of

screen names of the users.

**FIFTH QUERY:**

**SELECT user.followers\_count AS ft,user.screen\_name AS ust,count(\*) AS f\_count FROM**

**Lang\_Refiner GROUP BY user.followers\_count,user.screen\_name ORDER BY f\_count DESC**

**LIMIT 50**

Highest followers has been designed for this query. This query essence the context of followers count file

and is grouped with the screen name with the help of order by in descending.

**SIXTH QUERY:**

**SELECT user.friends\_count AS st,user.screen\_name AS ust,count(\*) AS fr\_count FROM**

**Lang\_Refiner GROUP BY user.friends\_count,user.screen\_name ORDER BY fr\_count DESC**

**LIMIT 50**

This query conceptualise about the friends count which is given as st and the count as fr\_count from the

defined Lang\_Refiner which as same described as above in the order of descending. It explains about

getting the highest number of friends.

**SEVENTH QUERY:**

**SELECT user.statuses\_count AS st,user.screen\_name AS ust,count(\*) AS s\_count FROM**

**Lang\_Refiner GROUP BY user.statuses\_count,user.screen\_name ORDER BY s\_count DESC**

**LIMIT 10**

It is designed for getting the high statuses. It explains about the statuses\_count of the user and the count of

all values from the Lang\_Refiner which is paired as per the designed outcomes and in the order of

descending.

**EIGHTH QUERY:**

**SELECT user.lang, COUNT(\*) as cnt FROM MainTable GROUP BY user.lang ORDER BY cnt**

**DESC limit 15**

In this query, users has been acquired with the highest language used.

**NINTH QUERY:**

**SELECT user.screen\_name, user.location FROM Lang\_Refiner where user.location IS NOT NULL**

This query defines about the user location which is not meant to be null.Getting the users with respect to

Location

**TENTH QUERY:**

**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**

Joining the two tables to get popular users with highest followers count.

**ELEVENTH QUERY:**

**SELECT user.screen\_name, user.time\_zone FROM Lang\_Refiner ORDER BY user.time\_zone**

**DESC**

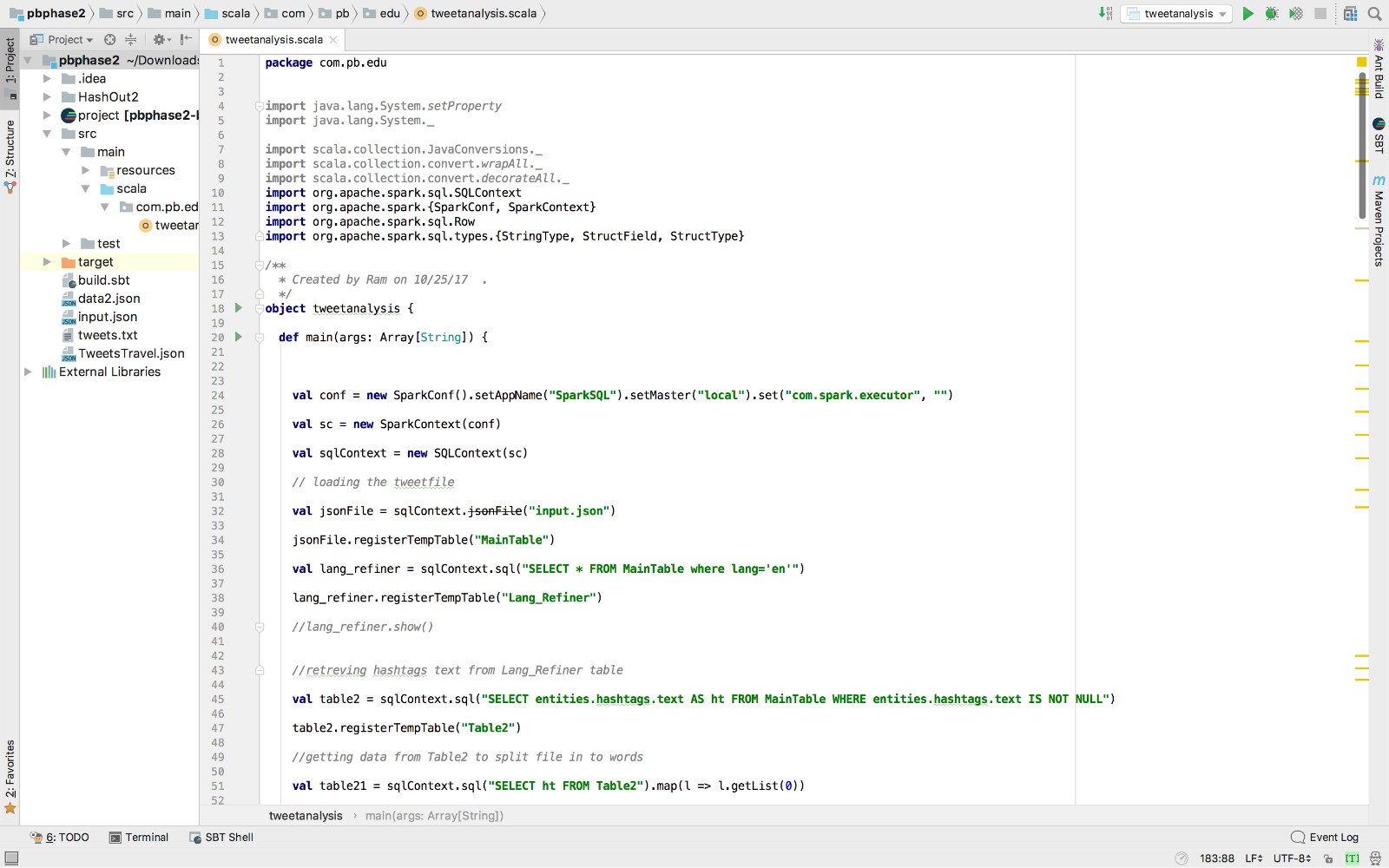
Getting the users with respect to time zones has been acquire from the above query.

**CODE FOR TWEET ANALYSIS**

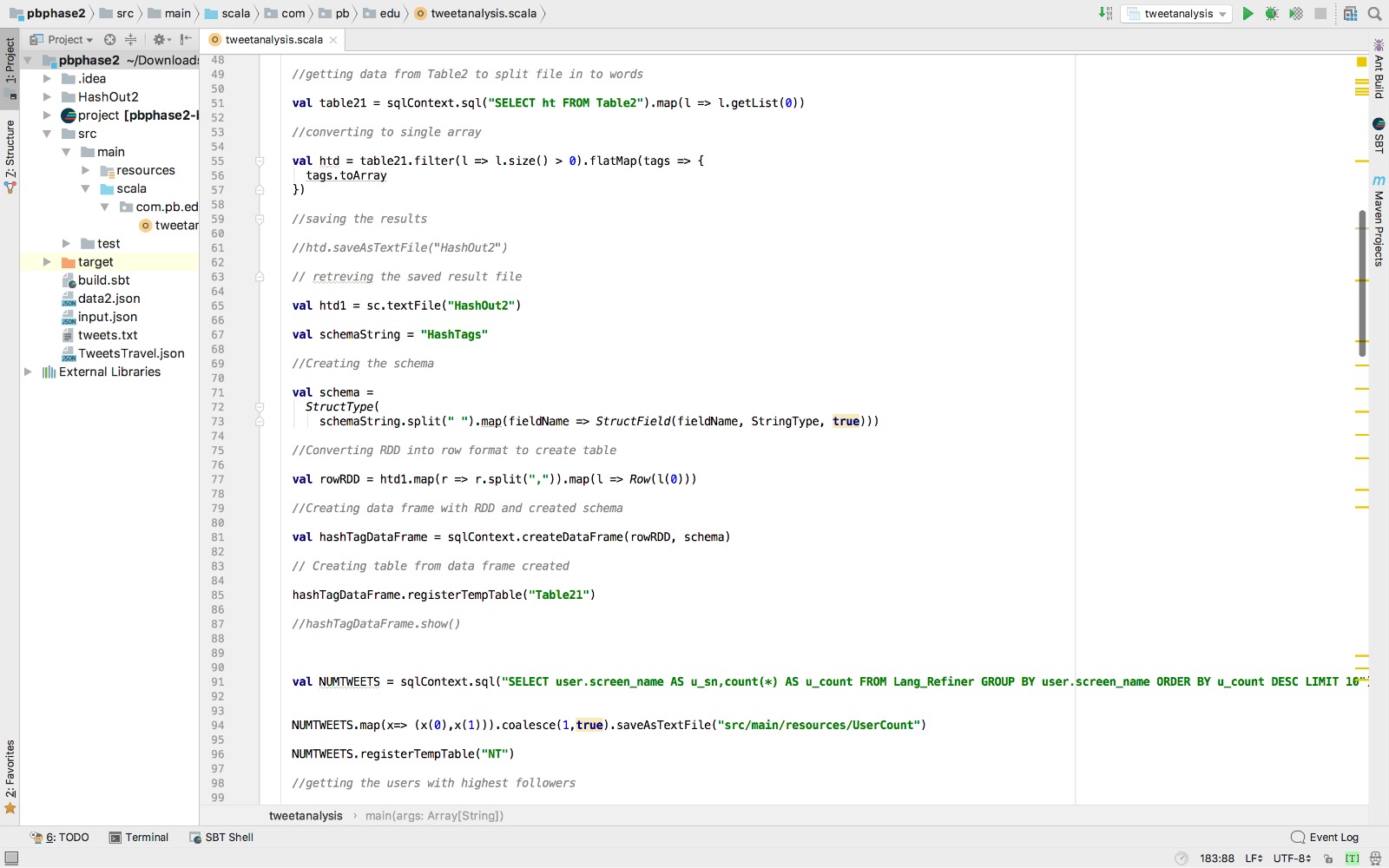
The code has been written in the Scala format for the implementation of queries in spark with the help of SQL

language.

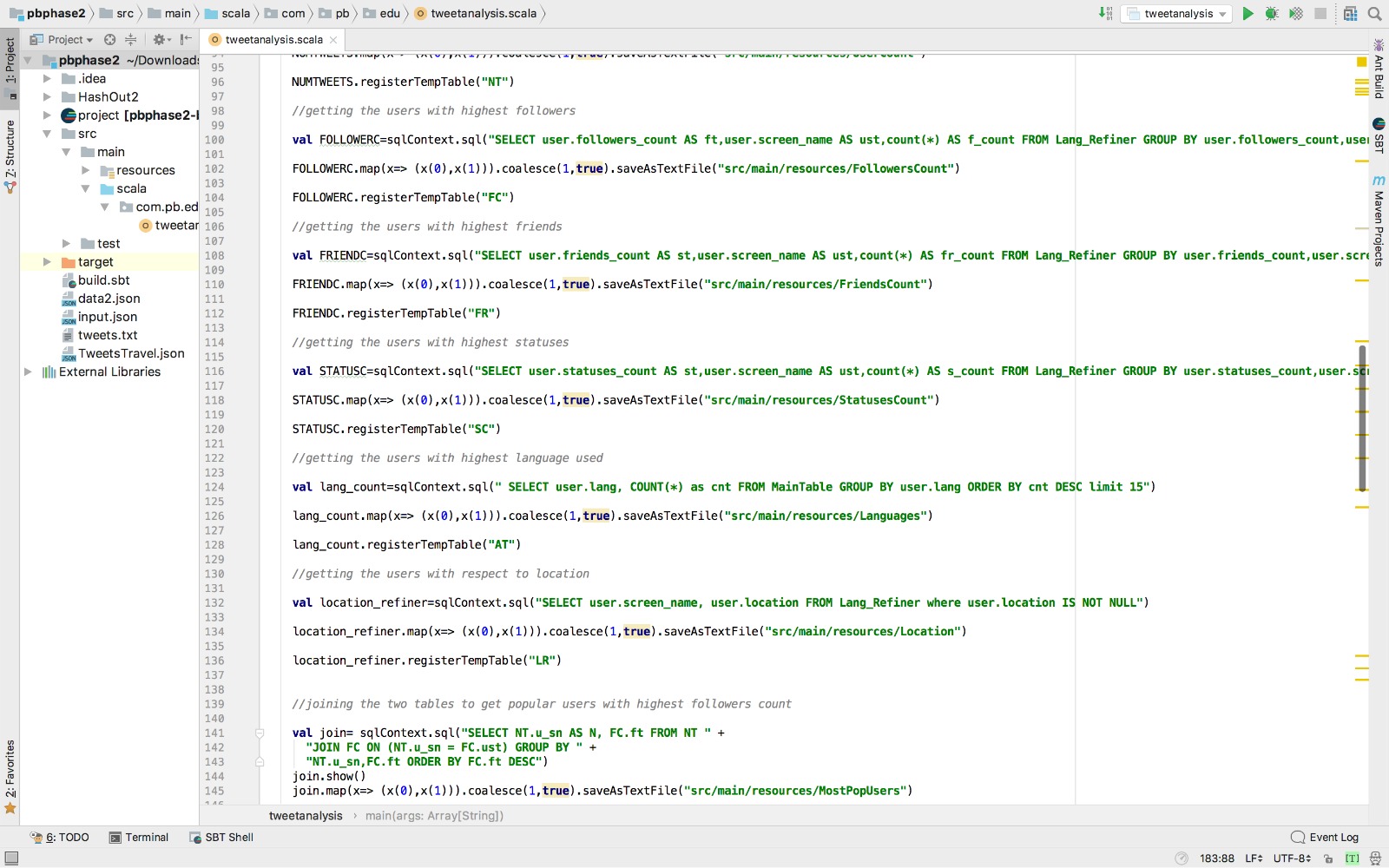
**CODE SNIPPETS:**



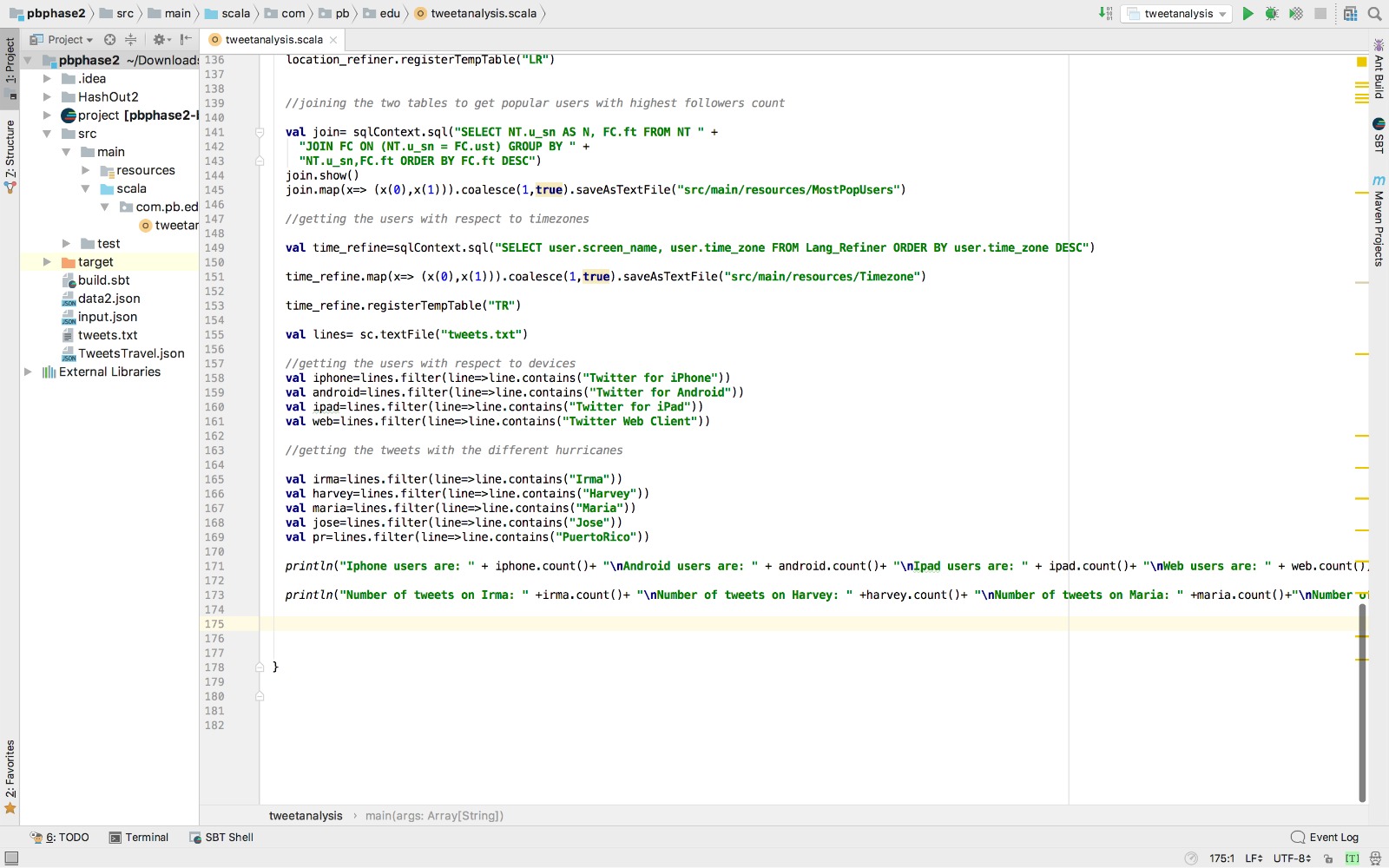
**CONTINUED**..

****

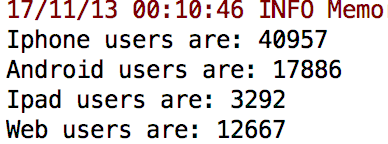
**CONTINUED..**

****

**CONTINUED..**

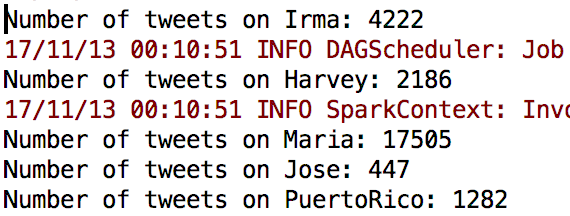


**DEVICE TWEETS OUTPUT:**



**NUMBER OF TWEETS POSTED ABOUT HURRICANE**

**OUTPUT:**

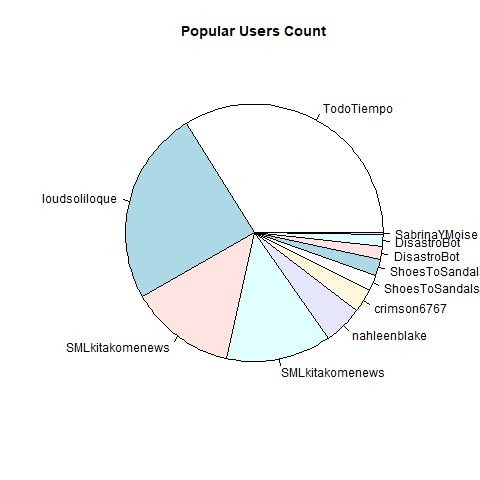
****

**VISUALISATION STRUCTURES**

**PIE CHART:**

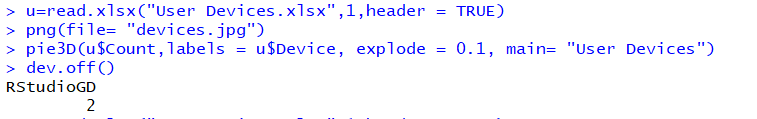
It is a graph which is used to illustrate the numerical proportion.

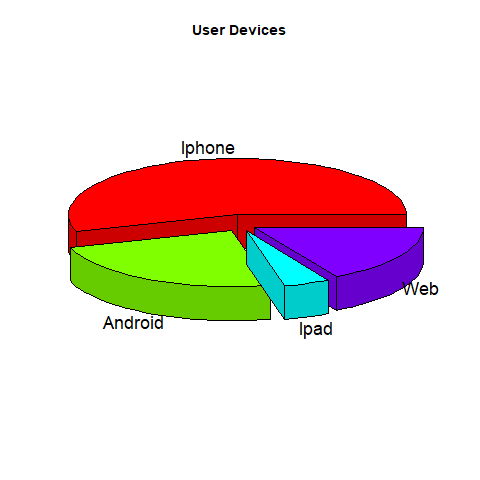
**POPULAR USERS COUNT:**



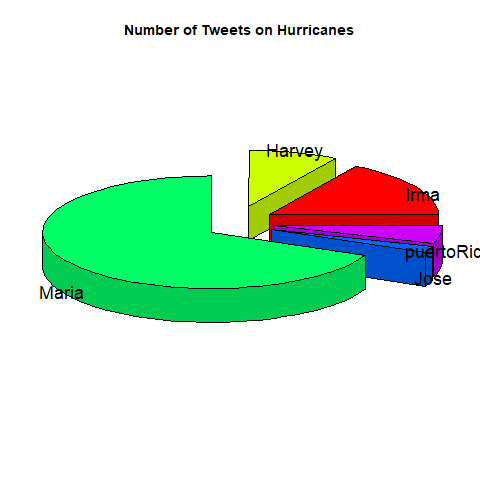
**DEVICES VISUALISATION:**

**CODE FOR PIE CHART:**

****



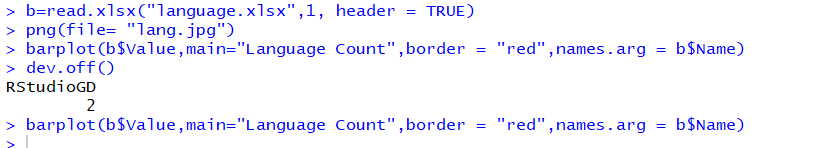
**HURRICANE TWEETS:**



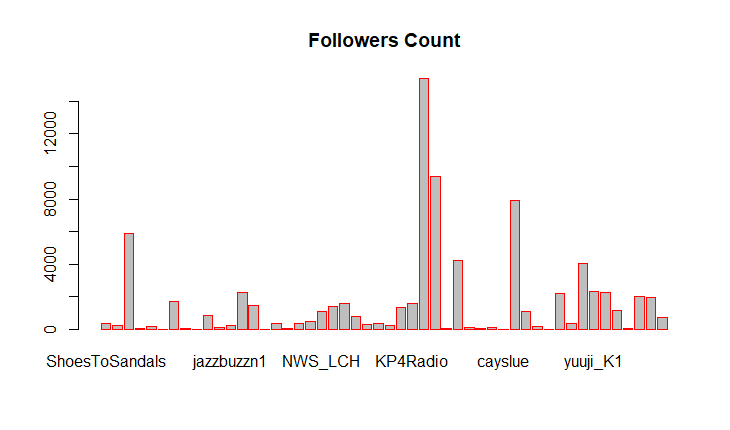
**BAR GRAPH:**

It represents the data with the bars that presents vertically and horizontally.

**CODE FOR BAR GRAPH:**



**FOLLOWERS COUNT:**



**LANGUAGE COUNT:**

