A Project Report on

"Principles of Big Data Management"

Phase-2

By

Ashok Yaganti	(ayktb@mail.umkc.edu)	(16197877)
Venkata Sasidhar Kanumuri	(vk6fb@mail.umkc.edu)	(16208516)
Ravi Teja Yakkula	(rybp3@mail.umkc.edu)	(16186983)

Under the esteemed guidance of

Dr. Praveen Rao

UMKC

UNIVERSITY OF MISSOURI

KANSAS CITY

INDEX

1. Introduction:

Twitter Analysis Application was developed for analyzing the twitter data by writing the analytical queries on Twitter Data.

1.1. Requirements Specification:

- Twitter Analysis Application requires the Twitter data to write the queries.
- Twitter Analysis Application needs the database to store the Twitter data.
- Twitter Analysis requires the Apache Spark for processing the twitter data.

1.2. Software Interface

- **Client:** Web Browser, HTML, HTML5, Javascript, d3.js.
- **Framework:** Apache Spark, Eclipse
- **Data Base Server:** Spark SQL.

2. Architectural Design

The SDD is a representation of a software system that is used as a interface to communicate software design information. This document contains description of the high level architecture used to develop the system. Communicating at a high level, it will form the basis for the Software Detailed Design and implementation.

This aims at decomposing the entire project into many modules, concurrent processes and data which will help in developing the software easily. A top level description of Project will be given, dividing it into its modules and explain their relation. The modules in the system contain public methods that run parallel processes and use data that has been modified during the system's active life period.

A module is a well-defined subsystem that is useful in various applications. Each module has a well-defined purpose. The modules can be individually compiled and can be stored individually in a library. These are easier to build. The entire project is decomposed into 3 modules.

- **Front module (GUI):**

  This module basically provides a graphical interface for those who cannot work with command line based applications.

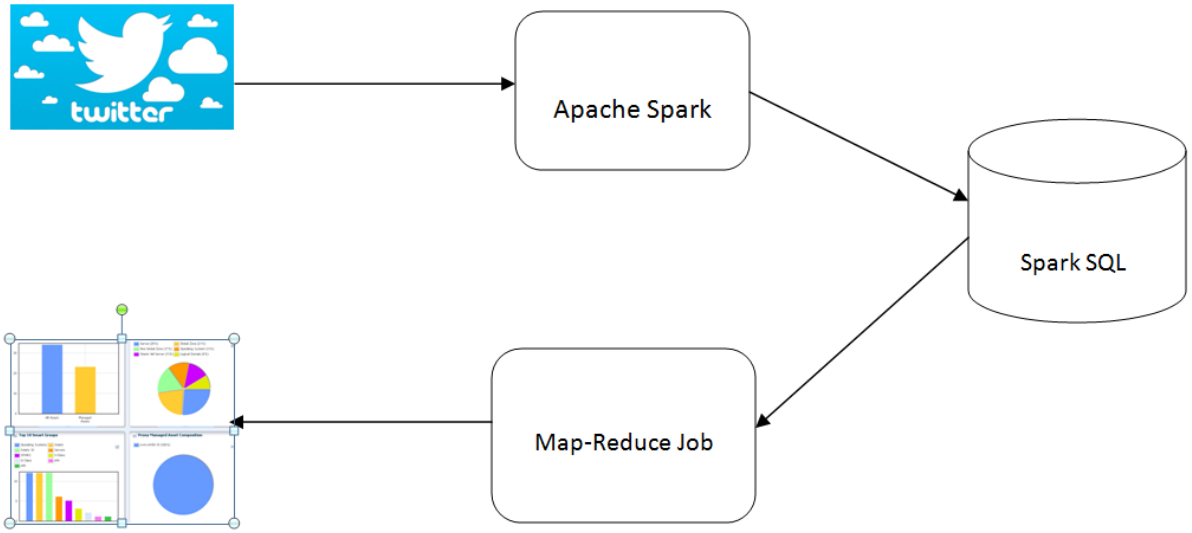
- **Database module:**

  This module provides the main functionality for the information management system.

- **Middle module:**

  This module provides a connection between the front and the business tier. It includes a request module and is as an interface that can be used by the front- tier

- **Business module:** This module contains all the functionalities that validate, update and control the data entering into the database.



3. Implementation

3.1. Server side Implementation.

From server side we used "Java" code to run the analytical queries in SPARK environment. First we load the data in spark, and then stores in spark SQL. Based on the query it executes and run the map reduce operation and displays the output.

Source Code for Analytical Queries. (TwitterAnalyticalQueries.java)

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.Collections;
import java.util.Comparator;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;

import javax.swing.JOptionPane;

import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.sql.api.java.JavaSQLContext;
import org.apache.spark.sql.api.java.JavaSchemaRDD;
import org.apache.spark.sql.api.java.Row;

import scala.Tuple2;

public class TwiiterAnalyticalQueries {

       private static String pathToFile =
"C:/Users/ashok/PBPhase2/com.umkc.ashok/TwitterData_Latest.txt";

      public static void main(String[] args) {
            while(true)
            {
            int choice=Integer.parseInt(JOptionPane.showInputDialog("Please
enter Your Option:\n 1. Top 8 Languages \n 2. Top 8 Users Tweet Count \n 3.
Top 8 Followers \n 4. Top 8 Background Image Color \n 5. User Names Having
more than 600000 Friends\n 6.Tweets on a Particular time\n
7.SentimentAnalysisQuery \n 8.Games Query \n 9.Tweet Status"));

            switch(choice)
            {
            case 1:
                    Top8LanguageQuery();
                        break;
            case 2:
                  Top8UsersTweetsCount();
                  break;
```

```
            case 3:
                  Top8UsersFollowers();
                  break;
            case 4:
                  Top8BackgroundImageColor();
                  break;
            case 5:
                  UserNamesHavingmorethan600000Friends();
                  break;
            case 6:
                  TimeQuery();
                  break;
            case 7:
                  SentimentAnalysisQuery();
                  break;
            case 8:
                  GamesQuery();
                  break;
            case 9:
                  TweetStatusQuery();
                  break;
            default: JOptionPane.showMessageDialog(null, "Invalid Option
please Enter from 1 to 8");
                              break;
                  }
            }
      }

    public static void Top8LanguageQuery()
      {

      SparkConf conf = new SparkConf().setAppName("User
mining").setMaster("local[*]");

         JavaSparkContext sc = new JavaSparkContext(conf);

         JavaRDD<Tweet1> tweets = sc.textFile(pathToFile).map(line ->
Parse.parseJsonToTweet(line));

         groupUserByName(tweets);

         nbTweetByUser(tweets);

         sc.stop();

         String htmlurl =
"http://twitteranalysispbm.mybluemix.net/TopLanguages.html";
         try {

      java.awt.Desktop.getDesktop().browse(java.net.URI.create(htmlurl));
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }

    private static void groupUserByName(JavaRDD<Tweet1> tweets)
    {
```

```
          tweets.groupBy(tweet -> tweet.getLang());
    }

    private static void nbTweetByUser(JavaRDD<Tweet1> tweets)
    {

      try
      {
           FileWriter fw= new
FileWriter("C:/Users/ashok/PBPhase2/TwitterAnalysis/WebContent/query1.csv");
          JavaPairRDD<String, Integer> nb = tweets.mapToPair(tweet -> new
Tuple2<>(tweet.getLang(), 1))
                                                    .reduceByKey((x, y) -> x +
y);

          Map<String, Integer> res = nb.collectAsMap();
          nb.foreach(line -> System.out.println(line));

          List<Map.Entry<String, Integer>> list = new
LinkedList<Map.Entry<String, Integer>>(res.entrySet());

                  // Sort list with comparator, to compare the Map values
                  Collections.sort(list, new Comparator<Map.Entry<String,
Integer>>() {
                        public int compare(Map.Entry<String, Integer> o1,
                                                Map.Entry<String, Integer>
o2) {
                              return
(o1.getValue()).compareTo(o2.getValue());
                        }
                  });
                  Collections.reverse(list);
                  // Convert sorted map back to a Map
                  Map<String, Integer> sortedMap = new LinkedHashMap<String,
Integer>();
                  for (Iterator<Map.Entry<String, Integer>> it =
list.iterator(); it.hasNext();) {
                        Map.Entry<String, Integer> entry = it.next();
                        sortedMap.put(entry.getKey(), entry.getValue());
                  }
                  fw.append("Language");
                  fw.append(',');
                  fw.append("Count");
                  fw.append("\n");

                  for (Map.Entry<String, Integer> entry :
sortedMap.entrySet()) {
                        System.out.println("[Key] : " + entry.getKey()
                                          + " [Value] : " +
entry.getValue());

                        if(entry.getKey()==null)
                        {
                        continue;
                        }
                        else
                        {
```

```
                        fw.append(entry.getKey());
                        fw.append(',');
                        fw.append(entry.getValue().toString());
                        fw.append("\n");
                        }
                    }
                    fw.close();
                }
                catch (Exception exp)
                {
                }
        }
        public static void Top8UsersTweetsCount()
        {
            SparkConf conf = new SparkConf().setAppName("User
mining").setMaster("local[*]");

            JavaSparkContext sc = new JavaSparkContext(conf);

            JavaSQLContext sqlContext = new JavaSQLContext(sc);

            JavaSchemaRDD tweets = sqlContext.jsonFile(pathToFile);

            tweets.registerAsTable("tweetTable");

            tweets.printSchema();

            nbTweetByUser(sqlContext);

            sc.stop();

            String htmlurl =
"http://twitteranalysispbm.mybluemix.net/FrequentTweetUsers.html";
          try {

        java.awt.Desktop.getDesktop().browse(java.net.URI.create(htmlurl));
              } catch (IOException e) {
                    e.printStackTrace();
              }
        }

        private static void nbTweetByUser(JavaSQLContext sqlContext)
        {
               try
               {

                     FileWriter fw= new
FileWriter("C:/Users/ashok/PBPhase2/TwitterAnalysis/WebContent/query2.csv");

            JavaSchemaRDD count = sqlContext.sql("SELECT
user.name,user.statuses_count AS c FROM tweetTable " +
                                                "ORDER BY c");


         List<org.apache.spark.sql.api.java.Row> rows = count.collect();

         Collections.reverse(rows);
```

```
 String rows123=rows.toString();

String[] array = rows123.split("],");

 System.out.println(rows123);

 fw.append("Name");
   fw.append(',');
   fw.append("Count");
   fw.append("\n");



   for(int i = 0; i < 8; i++)
   {
         if(i==0)
         {
               fw.append(array[0].substring(2));
               fw.append(',');
               fw.append("\n");
         }
         else {
         fw.append(array[i].substring(2));
         fw.append(',');
         fw.append("\n");
         }
   }

   fw.close();


}
       catch (Exception exp)
       {
       }

 }


public static void Top8UsersFollowers()
{
      SparkConf conf = new SparkConf().setAppName("User
mining").setMaster("local[*]");

   JavaSparkContext sc = new JavaSparkContext(conf);

   JavaSQLContext sqlContext = new JavaSQLContext(sc);

   JavaSchemaRDD tweets = sqlContext.jsonFile(pathToFile);

   tweets.registerAsTable("tweetTable");

  tweets.printSchema();

 nbTweetByFollower(sqlContext);
```

```
        sc.stop();

        String htmlurl =
"http://twitteranalysispbm.mybluemix.net/FamousPersons.html";
        try {

      java.awt.Desktop.getDesktop().browse(java.net.URI.create(htmlurl));
            } catch (IOException e) {
                  e.printStackTrace();
            }

      }

       private static void nbTweetByFollower(JavaSQLContext sqlContext) {

              try
              {
                    FileWriter fw= new
FileWriter("C:/Users/ashok/PBPhase2/TwitterAnalysis/WebContent/query3.csv");

          JavaSchemaRDD count = sqlContext.sql("SELECT DISTINCT
user.screen_name, user.followers_count AS c FROM tweetTable " +
                                                "ORDER BY c");

          List<org.apache.spark.sql.api.java.Row> rows = count.collect();

             Collections.reverse(rows);

               String rows123=rows.toString();

             String[] array = rows123.split("],");



          fw.append("Name");
            fw.append(',');
            fw.append("Count");
            fw.append("\n");



            for(int i = 0; i < 8; i++)
            {
                  if(i==0)
                  {
                        fw.append(array[0].substring(2));
                        fw.append(',');
                        fw.append("\n");
                  }
                  else {
                  fw.append(array[i].substring(2));
                  fw.append(',');
                fw.append("\n");
                  }
            }

            fw.close();
```

```
        }
                catch (Exception exp)
                {
                }


         }

      public static void Top8BackgroundImageColor()
      {
            SparkConf conf = new SparkConf().setAppName("User
mining").setMaster("local[*]");

        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaSQLContext sqlContext = new JavaSQLContext(sc);

        JavaSchemaRDD tweets = sqlContext.jsonFile(pathToFile);

        tweets.registerAsTable("tweetTable");

        tweets.printSchema();

        nbTweetByBackgroundImageColor(sqlContext);

        sc.stop();

        String htmlurl =
"http://twitteranalysispbm.mybluemix.net/BackgroundColors.html";
        try {

      java.awt.Desktop.getDesktop().browse(java.net.URI.create(htmlurl));
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }

  private static void nbTweetByBackgroundImageColor(JavaSQLContext
sqlContext)
  {

           try
              {
             FileWriter fw= new
FileWriter("C:/Users/ashok/PBPhase2/TwitterAnalysis/WebContent/query4.csv");

             JavaSchemaRDD count = sqlContext.sql("SELECT
user.profile_background_color, COUNT(*) AS c FROM tweetTable " +

      "Group By user.profile_background_color " +
                                                          "order by c" );

              List<org.apache.spark.sql.api.java.Row> rows = count.collect();

             Collections.reverse(rows);
```

```
            String rows123=rows.toString();

            String[] array = rows123.split("],");

            System.out.println(rows123);

        fw.append("ColorCode");
            fw.append(',');
            fw.append("Count");
            fw.append("\n");

            for(int i = 0; i < 12; i++)
            {
                  if((i==0)||(i==1)||(i==2))
                  {
                        continue;
                  }

                  else if(i == array.length-1)
                  {
                        fw.append(array[i].substring(2,array[i].length()-2));
                        fw.append(',');
                        fw.append("\n");
                  }
                  else {
                  fw.append(array[i].substring(2));
                  fw.append(',');
                  fw.append("\n");
                  }
            }

            fw.close();


        }
              catch (Exception exp)
              {
              }


         }


      public static void UserNamesHavingmorethan600000Friends()
      {
             SparkConf conf = new SparkConf().setAppName("User
mining").setMaster("local[*]");

         JavaSparkContext sc = new JavaSparkContext(conf);

         JavaSQLContext sqlContext = new JavaSQLContext(sc);

         JavaSchemaRDD tweets = sqlContext.jsonFile(pathToFile);

         tweets.registerAsTable("tweetTable");
```

```
         tweets.printSchema();

          nbTweetByFriends(sqlContext);

          sc.stop();
          String htmlurl =
"http://twitteranalysispbm.mybluemix.net/MoreFreinds.html";
          try {

      java.awt.Desktop.getDesktop().browse(java.net.URI.create(htmlurl));
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }

       private static void nbTweetByFriends(JavaSQLContext sqlContext) {

              try
              {
                    FileWriter fw= new
FileWriter("C:/Users/ashok/PBPhase2/TwitterAnalysis/WebContent/query5.csv");

          JavaSchemaRDD count = sqlContext.sql("SELECT DISTINCT
user.screen_name, user.friends_count AS c FROM tweetTable " +
                                                                        "WHERE
user.friends_count>'600000'" +
                                                        "order by c" );

          List<org.apache.spark.sql.api.java.Row> rows = count.collect();

       Collections.reverse(rows);

          String rows123=rows.toString();

         String[] array = rows123.split("],");

          System.out.println(rows123);

          fw.append("Name");
            fw.append(',');
            fw.append("Count");
            fw.append("\n");



            for(int i = 0; i < array.length; i++)
            {
                  if(i==0)
                  {
                        fw.append(array[0].substring(2));
                        fw.append(',');
                        fw.append("\n");
                  }
                  else if(i == array.length-1)
                  {
                        fw.append(array[i].substring(2,array[i].length()-2));
                        fw.append(',');
```

```
                        fw.append("\n");
                    }
                    else {
                    fw.append(array[i].substring(2));
                    fw.append(',');
                    fw.append("\n");
                    }
                }

                fw.close();


            }
                    catch (Exception exp)
                    {
                    }


             }


          public static void TimeQuery()
          {

                 SparkConf conf = new SparkConf().setAppName("User
mining").setMaster("local[*]");

             JavaSparkContext sc = new JavaSparkContext(conf);
             JavaSQLContext sqlContext = new JavaSQLContext(sc);

             JavaSchemaRDD tweets = sqlContext.jsonFile(pathToFile);

            tweets.registerAsTable("tweetTable");

           tweets.printSchema();

           nbTweetByTime(sqlContext);

           sc.stop();

           String htmlurl =
"http://twitteranalysispbm.mybluemix.net/MostTweetTimes.html";
           try {

          java.awt.Desktop.getDesktop().browse(java.net.URI.create(htmlurl));
                } catch (IOException e) {
                    e.printStackTrace();
                }
          }

           private static void nbTweetByTime(JavaSQLContext sqlContext)
           {
                    try
                    {
                        FileWriter fw= new
FileWriter("C:/Users/ashok/PBPhase2/TwitterAnalysis/WebContent/query6.csv");
```

```
          JavaSchemaRDD count = sqlContext.sql("SELECT created_at, COUNT(*)
AS c FROM tweetTable " +
                                                                    "Group By
created_at " +
                                                        "order by c" );

          List<org.apache.spark.sql.api.java.Row> rows = count.collect();

        Collections.reverse(rows);

          String rows123=rows.toString();

          String[] array = rows123.split("],");

          System.out.println(rows123);

          fw.append("Time");
            fw.append(',');
            fw.append("Count");
            fw.append("\n");



            for(int i = 0; i < 9; i++)
            {
                  if(i==0)
                  {
                        continue;
                  }
                  else if(i == array.length-1)
                  {
                        fw.append(array[i].substring(2,array[i].length()-2));
                        fw.append(',');
                        fw.append("\n");
                  }
                  else {
                  fw.append(array[i].substring(2));
                  fw.append(',');
                  fw.append("\n");
                  }
            }

            fw.close();


        }
              catch (Exception exp)
              {
              }


         }

      public static void SentimentAnalysisQuery()
      {
```

```
        SparkConf conf = new SparkConf().setAppName("User
mining").setMaster("local[*]");

        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaSQLContext sqlContext = new JavaSQLContext(sc);

        JavaSchemaRDD tweets = sqlContext.jsonFile(pathToFile);

        tweets.registerAsTable("tweetTable");

        tweets.printSchema();

        nbTweetBySentiment(sqlContext);

        sc.stop();

        String htmlurl =
"http://twitteranalysispbm.mybluemix.net/SentimentAnalysis.html";
        try {

      java.awt.Desktop.getDesktop().browse(java.net.URI.create(htmlurl));
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }

      private static void nbTweetBySentiment(JavaSQLContext sqlContext)

      {
          try
            {

             FileWriter fw= new
FileWriter("C:/Users/ashok/PBPhase2/TwitterAnalysis/WebContent/query7.csv");

        JavaSchemaRDD count = sqlContext.sql("SELECT  COUNT(*) AS c FROM
tweetTable " +
                                                                    "WHERE text
LIKE '%abundant%' OR text LIKE '%accessible%' OR text LIKE '%accurate%' OR
text LIKE '%award%' OR text LIKE '%awesome%' OR text LIKE '%beautiful%' OR
text LIKE '%affirmation%' OR text LIKE '%amicable%' OR text LIKE
'%appreciate%' OR text LIKE '%approve%' OR text LIKE '%attractive%' OR text
LIKE '%benefit%' OR text LIKE '%bless%' OR text LIKE '%bonus%' OR text LIKE
'%brave%' OR text LIKE '%bright%' OR text LIKE '%brilliant%' OR text LIKE
'%celebrate%' OR text LIKE '%champion%' OR text LIKE '%charm%' OR text LIKE
'%cheer%' OR text LIKE '%clever%' OR text LIKE '%colorful%' OR text LIKE
'%comfort%' OR text LIKE '%compliment%' OR text LIKE '%confidence%' OR text
LIKE '%congratulation%' OR text LIKE '%cute%' OR text LIKE '%good%' OR text
LIKE '%happy%' OR text LIKE '%cool%' OR text LIKE '%easy%' OR text LIKE
'%effective%' OR text LIKE '%efficient%' OR text LIKE '%fair%' OR text LIKE
'%excite%' OR text LIKE '%fast%' OR text LIKE '%fine%' OR text LIKE
'%fortunate%' OR text LIKE '%free%' OR text LIKE '%fresh%' OR text LIKE
'%fun%' OR text LIKE '%gain%' OR text LIKE '%gem%' OR text LIKE '%gorgeous%'
OR text LIKE '%grand%' OR text LIKE '%handsome%' OR text LIKE '%healthy%' OR
text LIKE '%honest%' OR text LIKE '%humor%' OR text LIKE '%important%' OR
text LIKE '%impress%' OR text LIKE '%improve%' OR text LIKE '%joy%' OR text
```

```
LIKE '%love%' OR text LIKE '%perfect%' OR text LIKE '%pleasant%' OR text LIKE
'%compliment%' OR text LIKE '%pleasure%' OR text LIKE '%precious%' OR text
LIKE '%prolific%' OR text LIKE '%prudent%' OR text LIKE '%happy%' OR text
LIKE '%cool%' OR text LIKE '%proven%' OR text LIKE '%effective%' OR text LIKE
'%efficient%' OR text LIKE '%restored%'  ");
        JavaSchemaRDD count1 = sqlContext.sql("SELECT  COUNT(*) AS c FROM
tweetTable " +
                        "WHERE text LIKE '%abuse%' OR text LIKE '%abyss%' OR
text LIKE '%absurd%' OR text LIKE '%akward%' OR text LIKE '%adverse%' OR text
LIKE '%agony%' OR text LIKE '%annoying%' OR text LIKE '%anti%' OR text LIKE
'%arrogant%' OR text LIKE '%assassinate%' OR text LIKE '%aversion%' OR text
LIKE '%backward%' OR text LIKE '%bad%' OR text LIKE '%brutal%' OR text LIKE
'%battered%' OR text LIKE '%berate%' OR text LIKE '%bewitch%' OR text LIKE
'%berate%' OR text LIKE '%blunder%' OR text LIKE '%complain%' OR text LIKE
'%conflict%' OR text LIKE '%confound%' OR text LIKE '%contagious%' OR text
LIKE '%contaminated%' OR text LIKE '%contravene%' OR text LIKE '%corruption%'
OR text LIKE '%corrupt%' OR text LIKE '%coward%' OR text LIKE '%cruel%' OR
text LIKE '%sad%' OR text LIKE '%danger%' OR text LIKE '%debase%' OR text
LIKE '%decline%' OR text LIKE '%deceive%' OR text LIKE '%defamation%' OR text
LIKE '%demon%' OR text LIKE '%demolish%' OR text LIKE '%denied%' OR text LIKE
'%demolish%' OR text LIKE '%depress%' OR text LIKE '%deny%' OR text LIKE
'%destroy%' OR text LIKE '%devastation%' OR text LIKE '%disadvantage%' OR
text LIKE '%disappointed%' OR text LIKE '%discord%' OR text LIKE '%evil%' OR
text LIKE '%gossip%' OR text LIKE '%hard%' OR text LIKE '%gloom%' OR text
LIKE '%hate%' OR text LIKE '%hazard%' OR text LIKE '%fuck%' OR text LIKE
'%horrible%' OR text LIKE '%idiot%' OR text LIKE '%imperfect%' OR text LIKE
'%inefficient%' OR text LIKE '%inflammation%' OR text LIKE '%ironic%' OR text
LIKE '%irritate%' OR text LIKE '%jealous%' OR text LIKE '%lag%' OR text LIKE
'%lie%' OR text LIKE '%malignant%' OR text LIKE '%malign%' OR text LIKE
'%noisy%' OR text LIKE '%odd%' OR text LIKE '%offence%' OR text LIKE
'%offend%' OR text LIKE '%offensive%' OR text LIKE '%bad%' OR text LIKE
'%unhappy%' OR text LIKE '%weak%'");

       List<Row> positive=count.collect();
       String positive12=positive.toString();
       String positive1 = positive12.substring(positive12.indexOf("[") + 2,
positive12.indexOf("]"));

       List<Row> negative=count1.collect();
       String negative12=negative.toString();
       String negative1 = negative12.substring(negative12.indexOf("[") + 2,
negative12.indexOf("]"));


           fw.append("Words");
             fw.append(',');
             fw.append("Count");
             fw.append("\n");
             fw.append("PositiveTweets");
             fw.append(',');
             fw.append(positive1);
             fw.append("\n");
             fw.append("NegativeTweets");
             fw.append(',');
             fw.append("-"+negative1);
             fw.append("\n");
```

```
            fw.close();


        }
                catch (Exception exp)
                {
                }


         }

      public static void GamesQuery()
      {
             SparkConf conf = new SparkConf().setAppName("User
mining").setMaster("local[*]");

         JavaSparkContext sc = new JavaSparkContext(conf);

         JavaSQLContext sqlContext = new JavaSQLContext(sc);

         JavaSchemaRDD tweets = sqlContext.jsonFile(pathToFile);

         tweets.registerAsTable("tweetTable");

         tweets.printSchema();

         nbTweetByGamesQuery(sqlContext);

        sc.stop();

        String htmlurl =
"http://twitteranalysispbm.mybluemix.net/TopGames.html";
        try {

      java.awt.Desktop.getDesktop().browse(java.net.URI.create(htmlurl));
            } catch (IOException e) {
                  e.printStackTrace();
            }

      }

       private static void nbTweetByGamesQuery(JavaSQLContext sqlContext) {

              try
              {
                    FileWriter fw= new
FileWriter("C:/Users/ashok/PBPhase2/TwitterAnalysis/WebContent/query8.csv");

          JavaSchemaRDD count = sqlContext.sql("SELECT  COUNT(*) AS c FROM
tweetTable " +
                                                                  "WHERE text
LIKE '%cricket%'");
          JavaSchemaRDD count1 = sqlContext.sql("SELECT  COUNT(*) AS c FROM
tweetTable " +
```

```
                        "WHERE text LIKE '%tennis%'");
          JavaSchemaRDD count2 = sqlContext.sql("SELECT  COUNT(*) AS c FROM
tweetTable " +
                        "WHERE text LIKE '%Baseball%'");
          JavaSchemaRDD count3 = sqlContext.sql("SELECT  COUNT(*) AS c FROM
tweetTable " +
                        "WHERE text LIKE '%soccer%'");
          JavaSchemaRDD count4 = sqlContext.sql("SELECT  COUNT(*) AS c FROM
tweetTable " +
                        "WHERE text LIKE '%basketball%'");
          JavaSchemaRDD count5 = sqlContext.sql("SELECT  COUNT(*) AS c FROM
tweetTable " +
                        "WHERE text LIKE '%Golf%'");

       List<Row> cricket=count.collect();
       String cricket12=cricket.toString();
       String cricket1 = cricket12.substring(cricket12.indexOf("[") + 2,
cricket12.indexOf("]"));

       List<Row> tennis=count1.collect();
       String tennis12=tennis.toString();
       String tennis1 = tennis12.substring(tennis12.indexOf("[") + 2,
tennis12.indexOf("]"));

       List<Row> Baseball=count2.collect();
       String Baseball12=Baseball.toString();
       String Baseball1 = Baseball12.substring(Baseball12.indexOf("[") + 2,
Baseball12.indexOf("]"));

       List<Row> soccer=count3.collect();
       String soccer12=soccer.toString();
       String soccer1 = soccer12.substring(soccer12.indexOf("[") + 2,
soccer12.indexOf("]"));

       List<Row> basketball=count4.collect();
       String basketball12=basketball.toString();
       String basketball1 = basketball12.substring(basketball12.indexOf("[")
+ 2, basketball12.indexOf("]"));

       List<Row> Golf=count5.collect();
       String Golf12=Golf.toString();
       String Golf1 = Golf12.substring(Golf12.indexOf("[") + 2,
Golf12.indexOf("]"));


          fw.append("GameName");
            fw.append(',');
            fw.append("Count");
            fw.append("\n");
            fw.append("Cricket");
            fw.append(',');
            fw.append(cricket1);
            fw.append("\n");
            fw.append("Tennis");
            fw.append(',');
            fw.append(tennis1);
            fw.append("\n");
```

```
            fw.append("Baseball");
            fw.append(',');
            fw.append(Baseball1);
            fw.append("\n");
            fw.append("soccer");
            fw.append(',');
            fw.append(soccer1);
            fw.append("\n");
            fw.append("Basketball");
            fw.append(',');
            fw.append(basketball1);
            fw.append("\n");
            fw.append("Golf");
            fw.append(',');
            fw.append(Golf1);
            fw.append("\n");



            fw.close();


        }
               catch (Exception exp)
               {
               }


         }

        public static void TweetStatusQuery()
        {
                    String pathToFile =
"C:/Users/ashok/PBPhase2/com.umkc.ashok/TwitterData_Latest.txt";
                SparkConf conf = new SparkConf().setAppName("User
mining").setMaster("local[*]");

                JavaSparkContext sc = new JavaSparkContext(conf);

                JavaSQLContext sqlContext = new JavaSQLContext(sc);

                JavaSchemaRDD tweets = sqlContext.jsonFile(pathToFile);

                tweets.registerAsTable("tweetTable");

                tweets.printSchema();

                nbTweetByStatus(sqlContext);

                sc.stop();

                String htmlurl =
"http://twitteranalysispbm.mybluemix.net/tweet_status_analysis.html";
               try {

      java.awt.Desktop.getDesktop().browse(java.net.URI.create(htmlurl));
                    } catch (IOException e) {
```

```
                        e.printStackTrace();
                    }
        }

        private static void nbTweetByStatus(JavaSQLContext sqlContext)
        {

                try
                {
                        FileWriter fw= new
FileWriter("C:/Users/ashok/PBPhase2/TwitterAnalysis/WebContent/query9.csv");

            JavaSchemaRDD totalcount = sqlContext.sql("SELECT  COUNT(*) AS c
FROM tweetTable ");

            JavaSchemaRDD count = sqlContext.sql("SELECT  COUNT(*) AS c FROM
tweetTable " +
                                                                    "WHERE
retweeted_status.retweet_count>0 ");

            JavaSchemaRDD count1 = sqlContext.sql("SELECT  COUNT(*) AS c FROM
tweetTable " +
                                                        "WHERE retweet_count=0
");


        List<Row> totalrows=totalcount.collect();
        String totalrows12=totalrows.toString();
        String totalrows1 = totalrows12.substring(totalrows12.indexOf("[") +
2, totalrows12.indexOf("]"));

        List<Row> retweetcount=count.collect();
        String retweetcount12=retweetcount.toString();
        String retweetcount1 =
retweetcount12.substring(retweetcount12.indexOf("[") + 2,
retweetcount12.indexOf("]"));

        List<Row> notretweetcount=count1.collect();
        String notretweetcount12=notretweetcount.toString();
        String notretweetcount1 =
notretweetcount12.substring(notretweetcount12.indexOf("[") + 2,
notretweetcount12.indexOf("]"));

        System.out.println(totalrows1);
        System.out.println(retweetcount1);

       int totalrows123=Integer.parseInt(totalrows1);

       int retweet123=Integer.parseInt(retweetcount1);

       int notretweet=Integer.parseInt(notretweetcount1);

        int deletedtweet=totalrows123- (retweet123+notretweet);

        System.out.println(notretweet);

       double retweetPercentage=((retweet123*100)/totalrows123);
```

```
        double notweetPercentage=((notretweet*100)/totalrows123);
        float deletedtweetPercentage=((deletedtweet*100)/totalrows123);

        System.out.println(retweetPercentage);
        System.out.println(notweetPercentage);
        System.out.println(deletedtweetPercentage);

        String retweetPercentage1=Double.toString(retweetPercentage);
        String notweetPercentage1=Double.toString(notweetPercentage);
        String deletedtweetPercentage1=Float.toString(deletedtweetPercentage);



            fw.append("TweetStatus");
              fw.append(',');
              fw.append("Percentage");
              fw.append("\n");
              fw.append("Retweet Percentage");
              fw.append(',');
              fw.append(retweetPercentage1);
              fw.append("\n");
              fw.append("Not Retweet Percentage");
              fw.append(',');
              fw.append(notweetPercentage1);
              fw.append("\n");
              fw.append("deleted tweets Percentage");
              fw.append(',');
              fw.append(deletedtweetPercentage1);
              fw.append("\n");





              fw.close();


        }
                catch (Exception exp)
                {
                }


         }

}
```

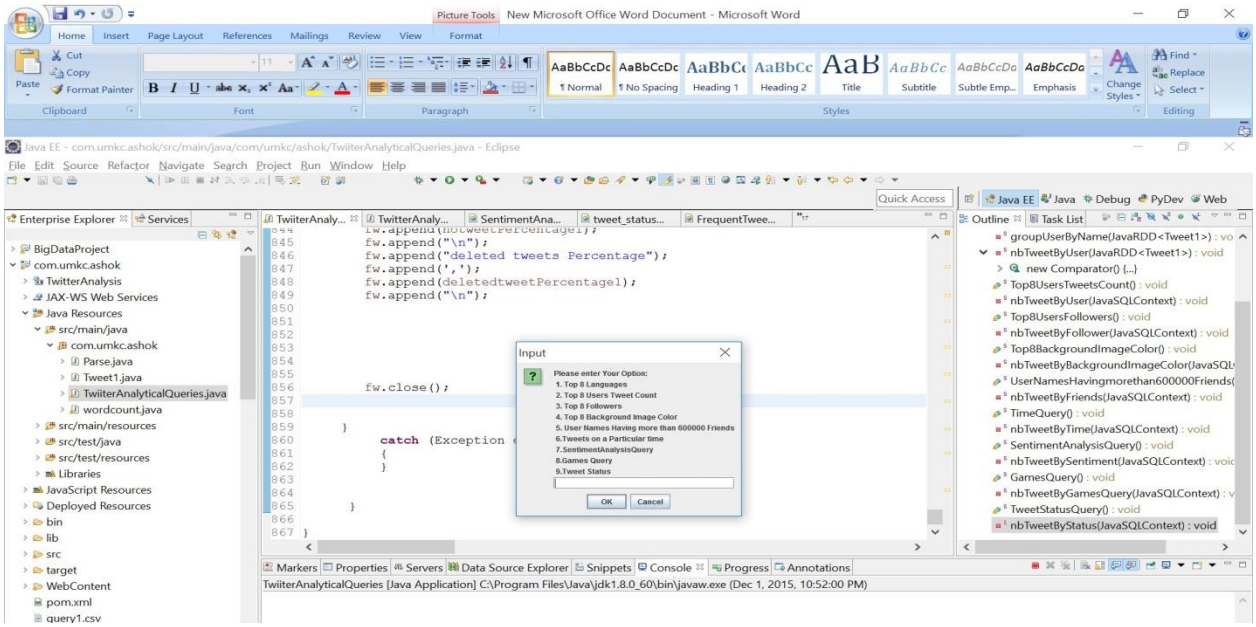
3.2. Client side implementation

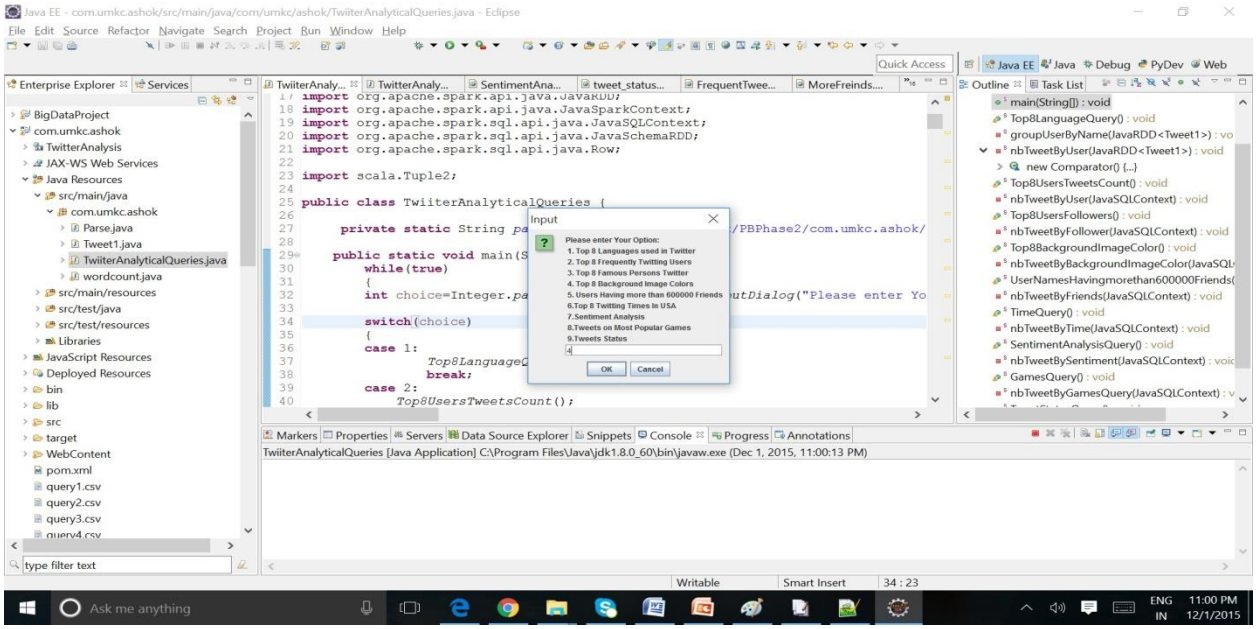
For the Client side we have used HTML, HTML5, JavaScript to visualize the data.

We have used d3.js examples for visualizing the twitter data.

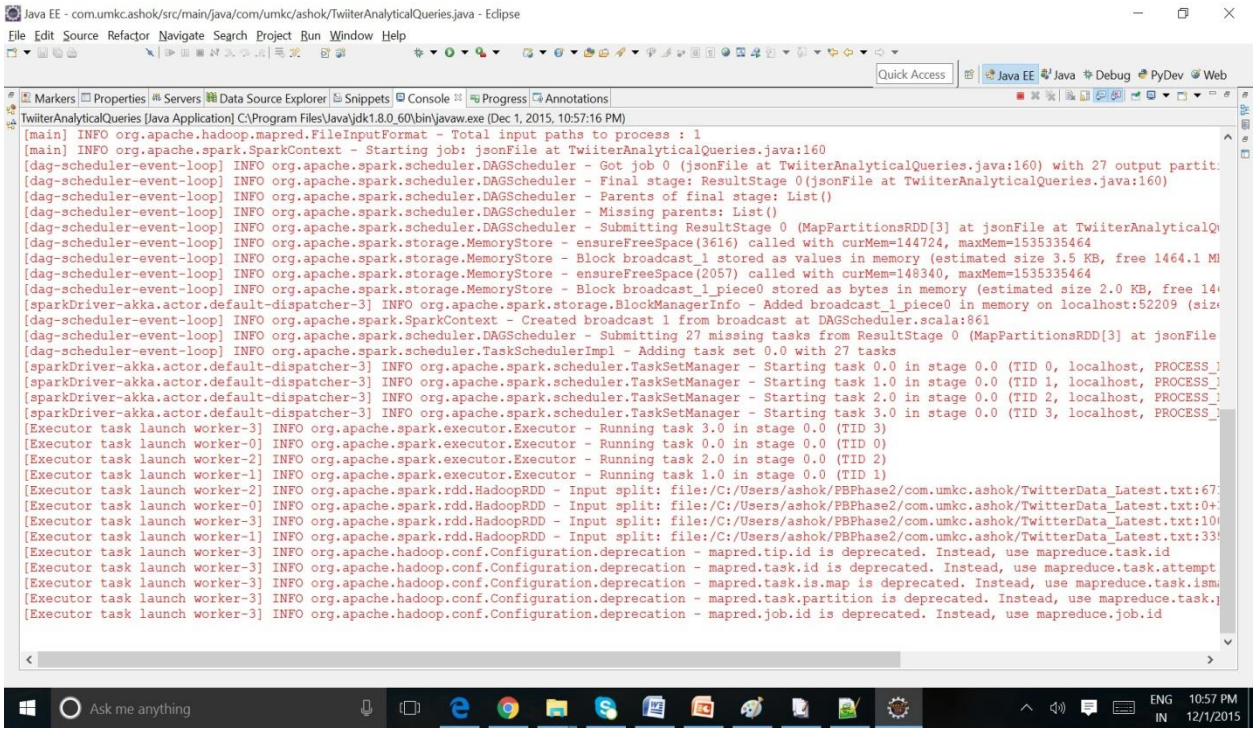
4. Implementation Results

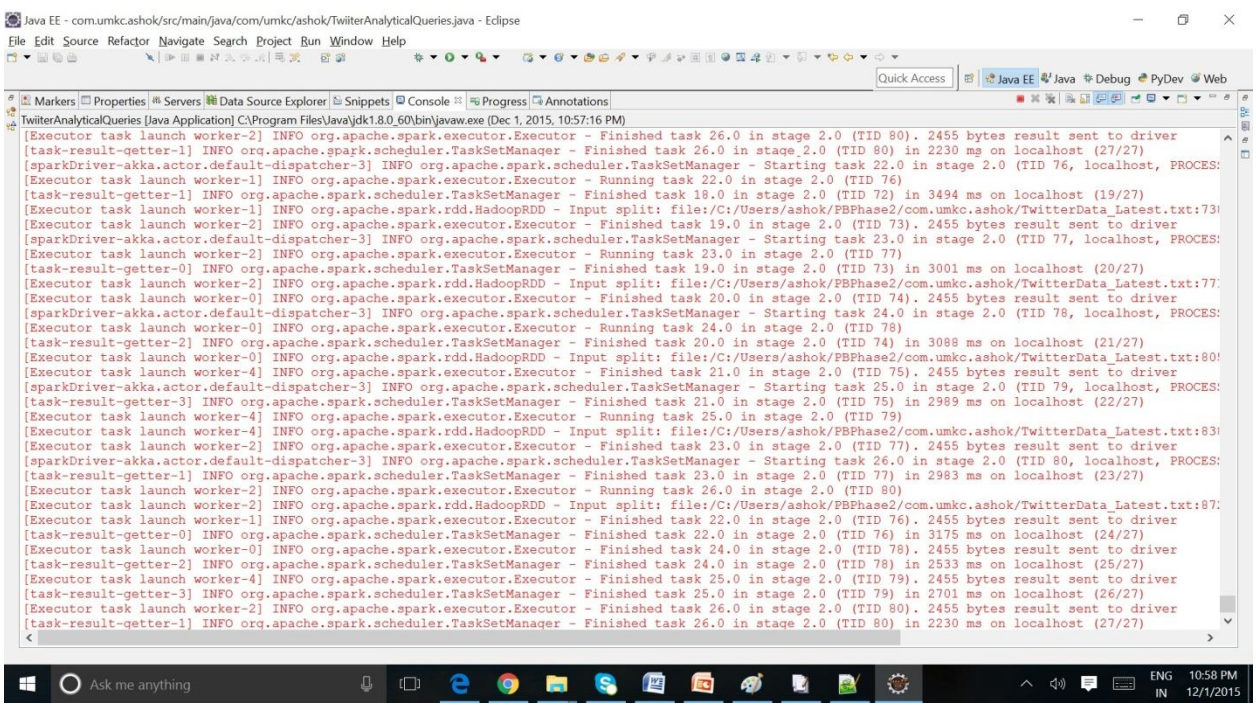
1. Selecting the query from the Joption Pane Window.



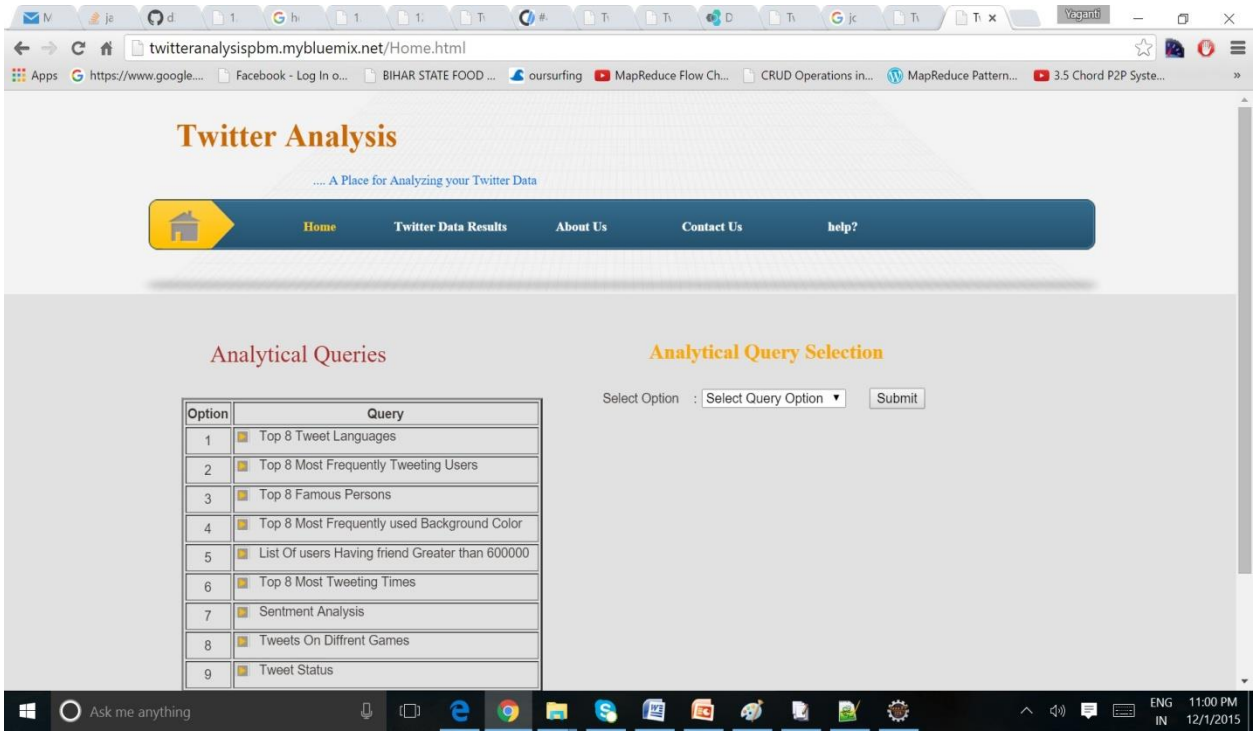


2. Background Process Running in Apache Spark.

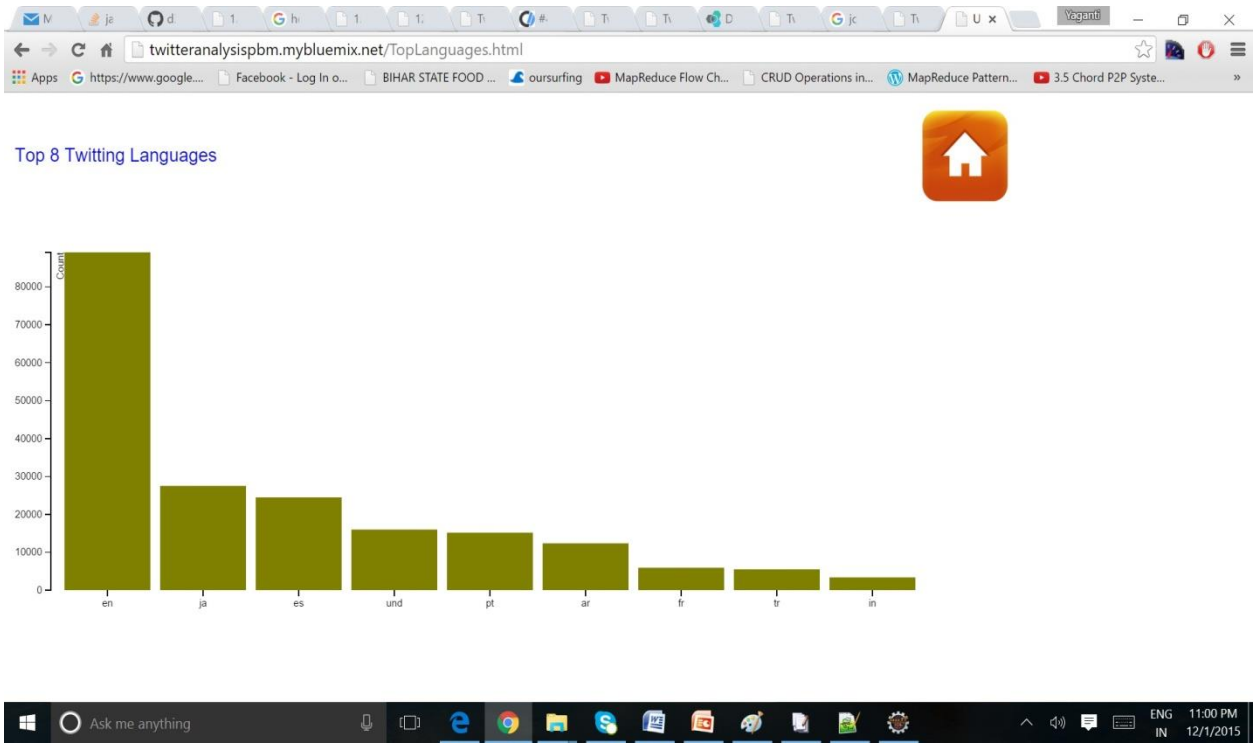




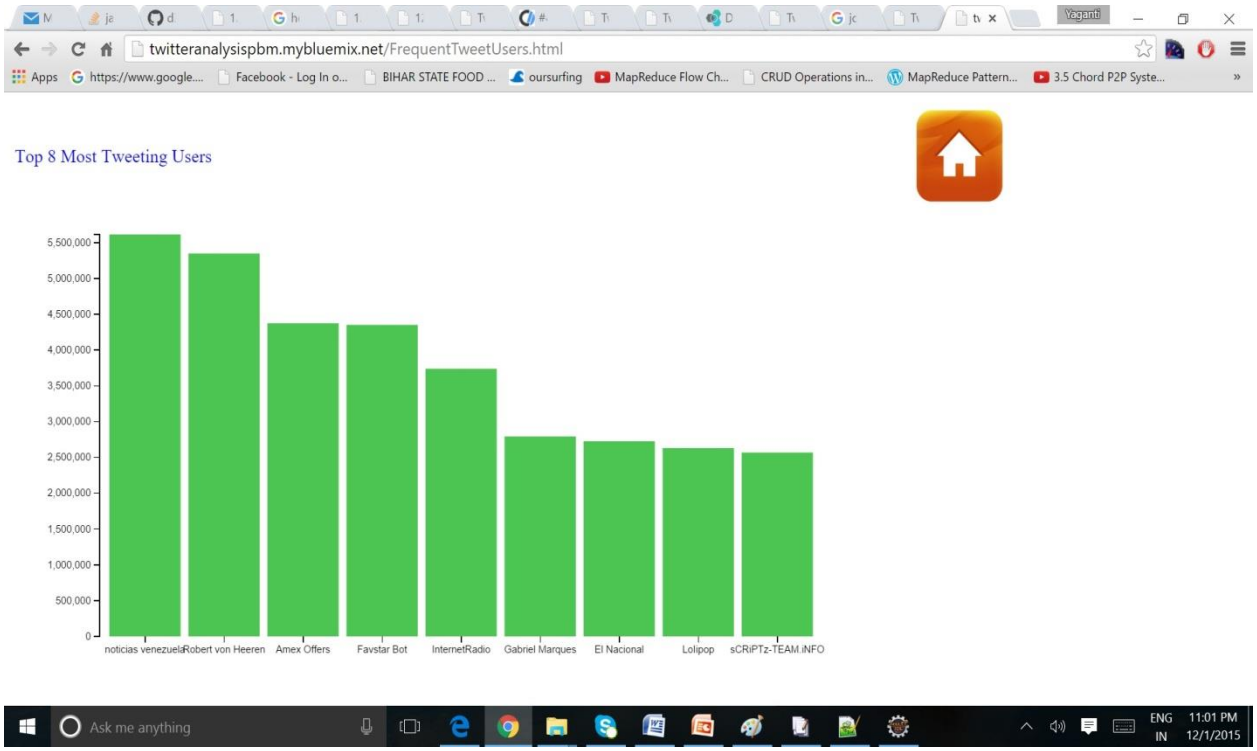
3. Home Page for selecting the Option...



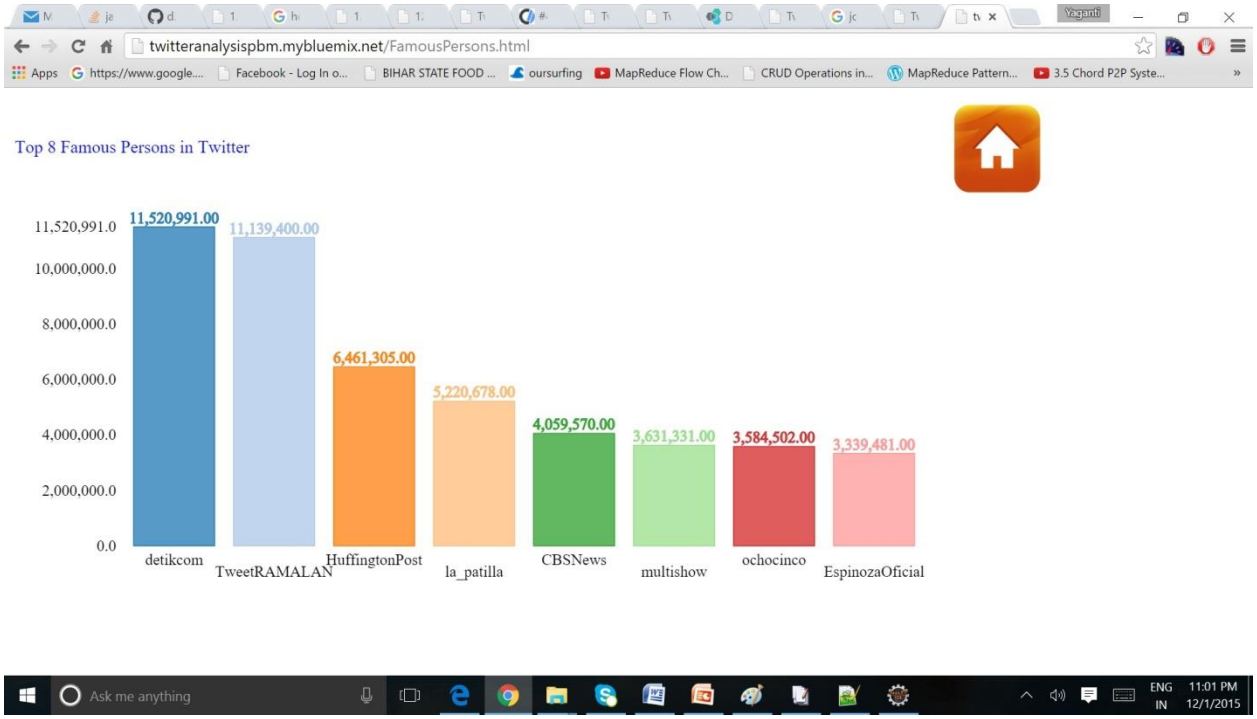
4. Top 8 Twitting Languages Visualization.



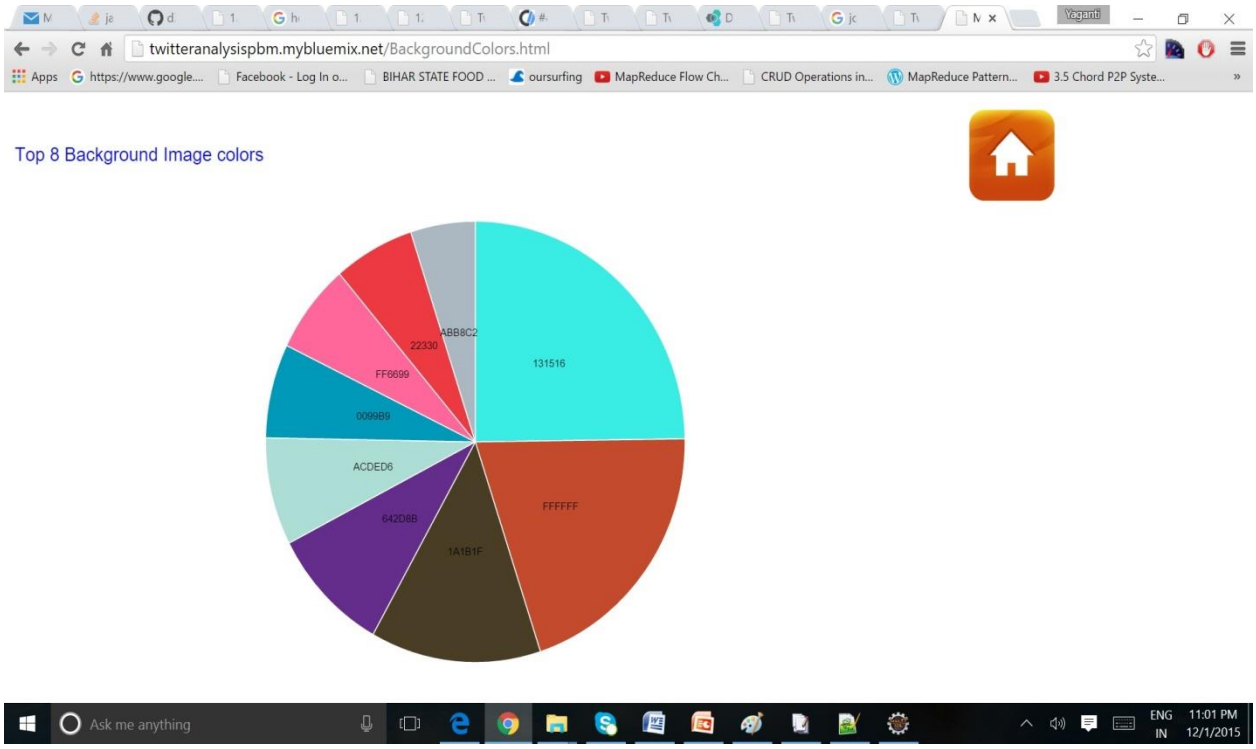
5. Top8 Twitting Users.



6. Most Famous users in Twitter.

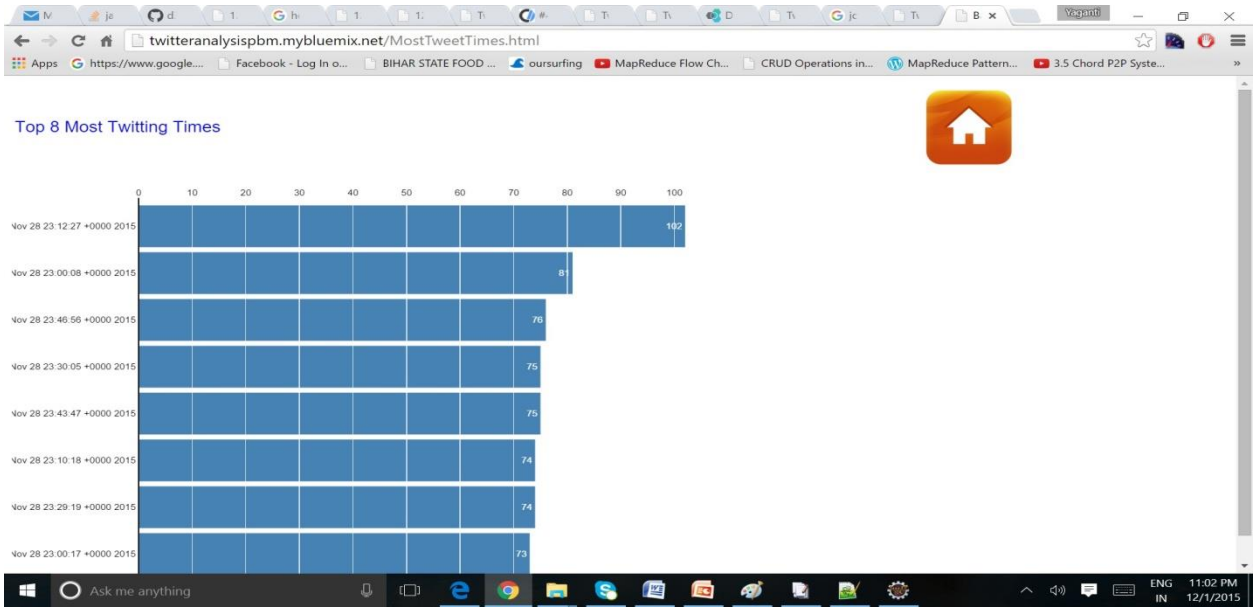


7. Top 8 Backgroung Image Colors that are using Twitter users.

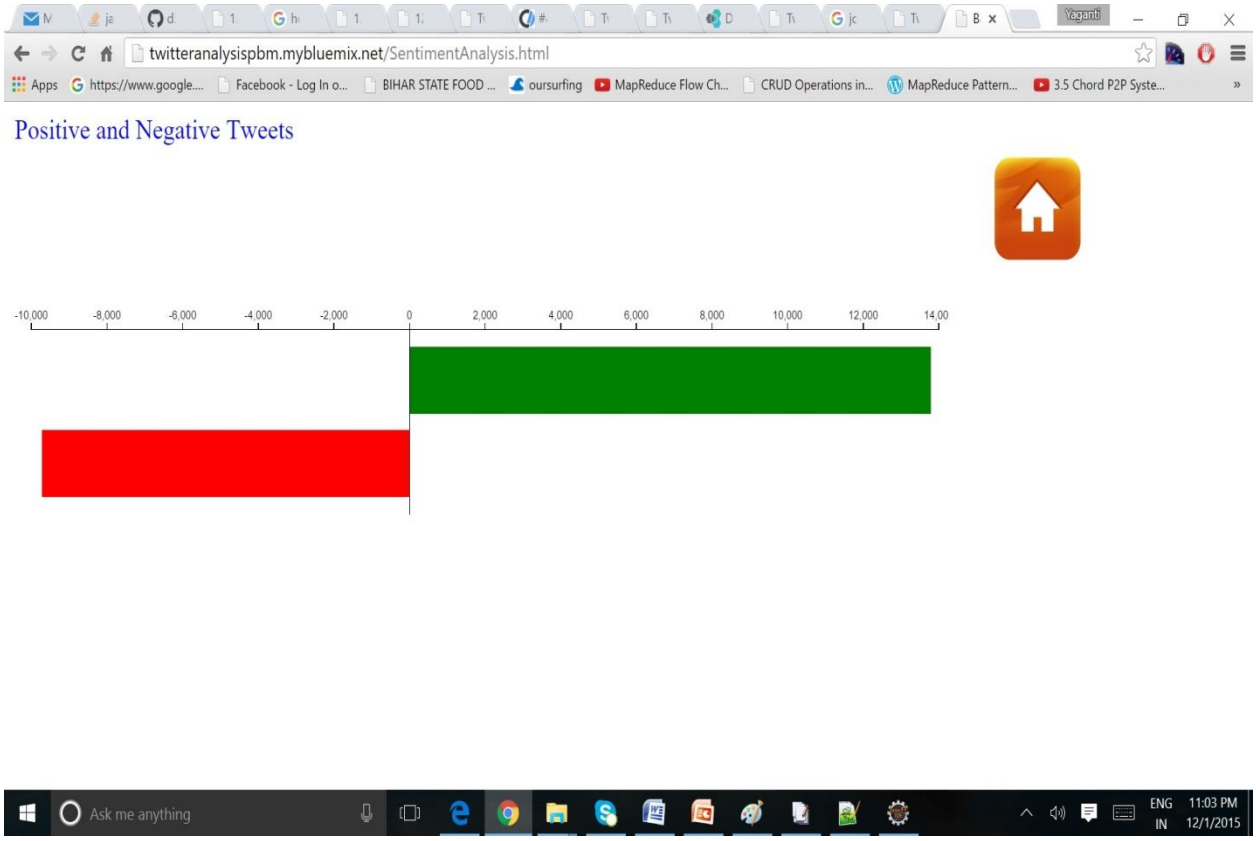


8. Users having the friend more than 600000

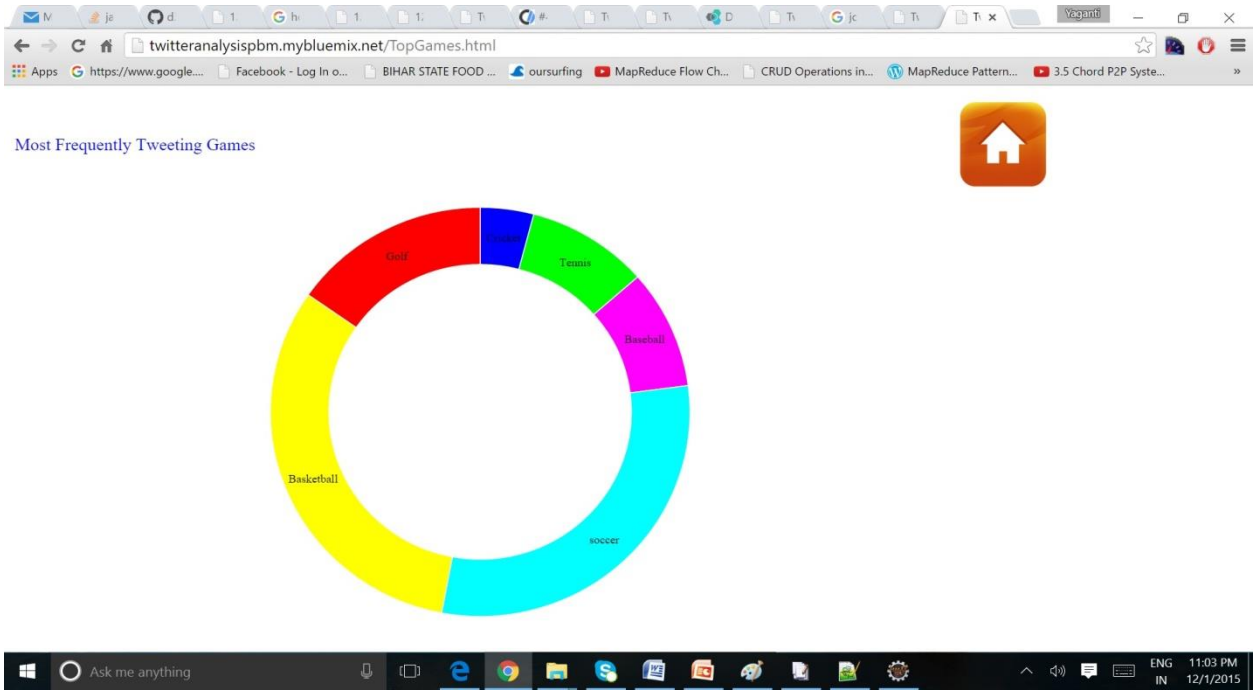
9. The Top 8 Time that the users twitting More in USA



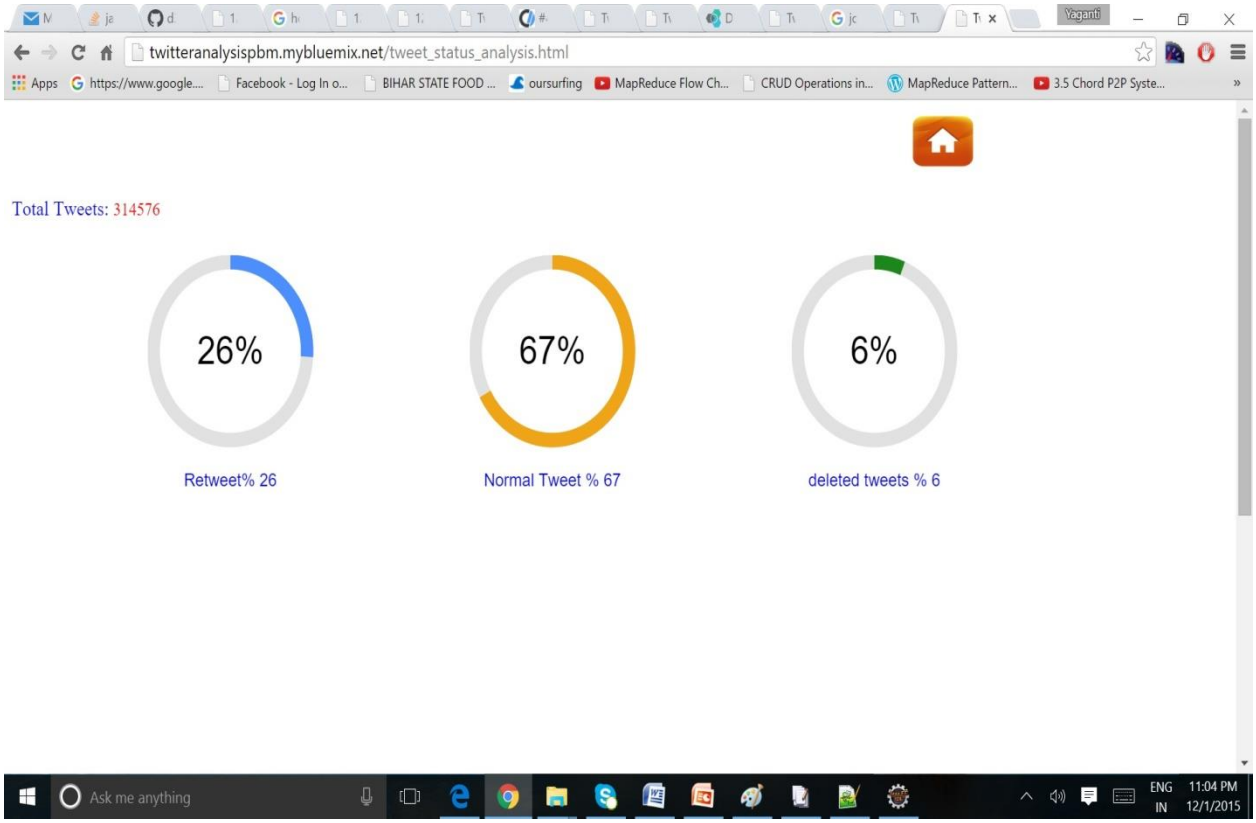
10. Sentment Analysis by Positive and Negative Words.



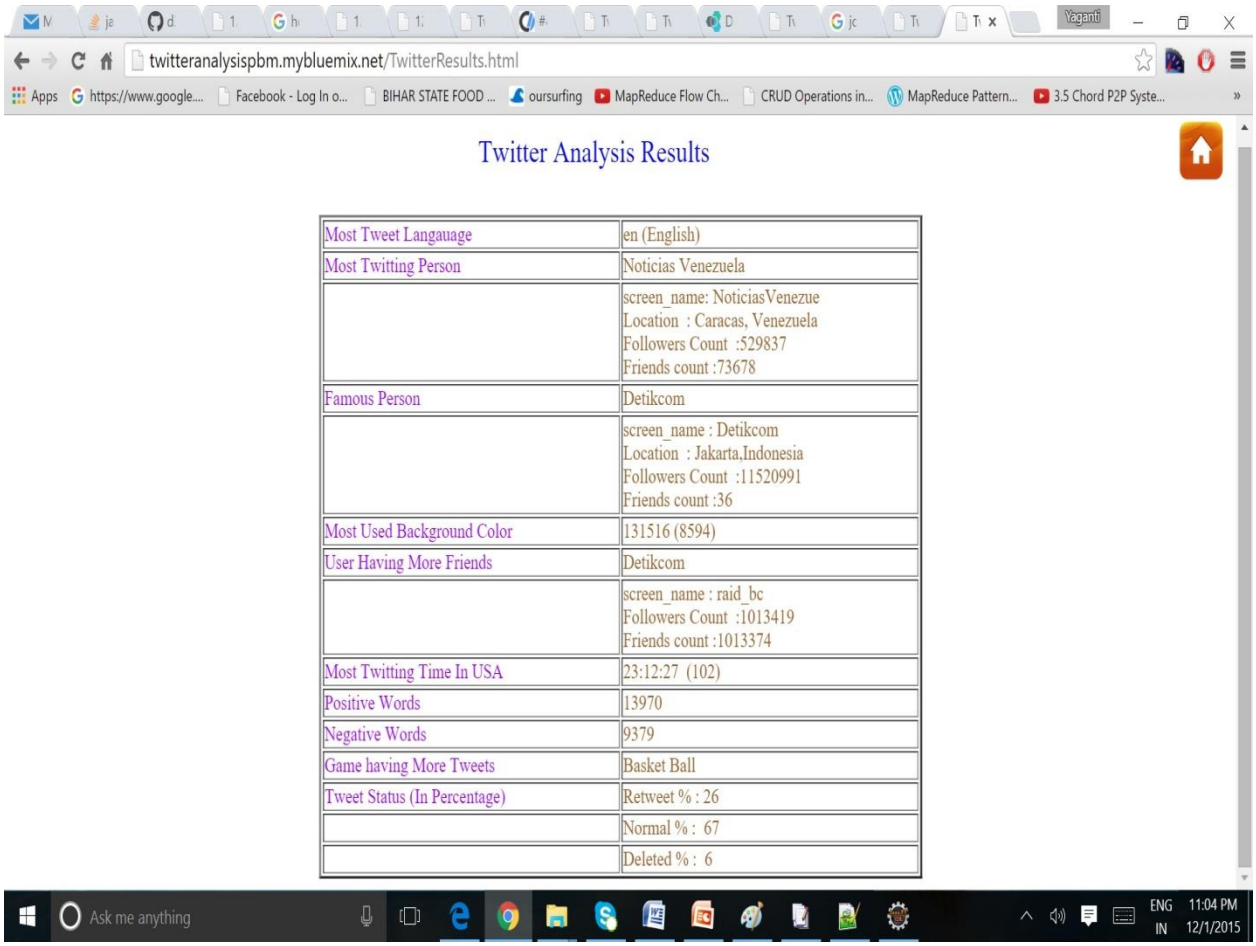
11. User Tweets contains most popular Games.



12. Tweets Status.



13. Twitter Analysis Results on Whole Application.



Most Tweet Langauage	en (English)
Most Twitting Person	Noticias Venezuela
	screen_name: NoticiasVenezue Location : Caracas, Venezuela Followers Count :529837 Friends count :73678
Famous Person	Detikcom
	screen_name : Detikcom Location : Jakarta,Indonesia Followers Count :11520991 Friends count :36
Most Used Background Color	131516 (8594)
User Having More Friends	Detikcom
	screen_name : raid_bc Followers Count :1013419 Friends count :1013374
Most Twitting Time In USA	23:12:27 (102)
Positive Words	13970
Negative Words	9379
Game having More Tweets	Basket Ball
Tweet Status (In Percentage)	Retweet % : 26
	Normal % : 67
	Deleted % : 6

5. Deployment

We have deployed our project in Bluemix and source code available in Github. Please find the URL for the deployed project.

Bluemix URL : http://twitteranalysispbm.mybluemix.net/

Github URL : https://github.com/AshokYaganti/PB_Phase2_TwitterAnalysis/

6. References

- “https://github.com/mbostock/d3/wiki/Gallery”
- “https://github.com/stefani75/workspace/tree/b90a63f2f3028fb358b28a77ac416b223d37a52a/projet1/Hands-On-Spark-java-solution/src/main/java/com/duchessfr/spark”
- “https://spark.apache.org/docs/1.3.0/sql-programming-guide.html#generic-loadsave-functions /”
- “ http://www.taywils.me/2013/11/05/javasparkframeworktutorial.html”