Principles of Big Data
Project Phase-3

Submitted by,

TEAM 27

Sravan Kumar Appana -16221049

Sri Vidya Surya Haritha Jyothula - 16222098

Lakshmi Sai Krishna Ravilla – 16224836

Sainath Reddy - 16232714

Aim :

In this project,

1. We build a user interface that enables the user to select one of the 5 queries.
2. Then we display results of each selected query in any visualization which might be either in the form of bar chart or pie chart.

Theme :

We have selected “Iphone 7” as search keyword. We selected Iphone7 as our theme since it is the trending hot topic in todays technology world. Our project was started before the release of Iphone7. Many apple lovers were excited regarding its specifications, new changes in the upcoming iphone 7 world-wide. The number of tweets tweeted have been increased because of some major changes like eliminating 3.5mm headphone jack, etc.,

Hashtag – Iphone7

Here five queries are run using Apache Spark and Java programming.

1. Two queries are run using Spark RDDs.
2. Another Two queries are run using Spark Data Frames.
3. One query is called the public APIs so as to update some part of information in the collection and retrieval of some other data which is not included originally.

Libraries:

1. Maven dependency.
2. Two jar files.
3. Spark-core 2.10
4. Spark-SQL 2.10

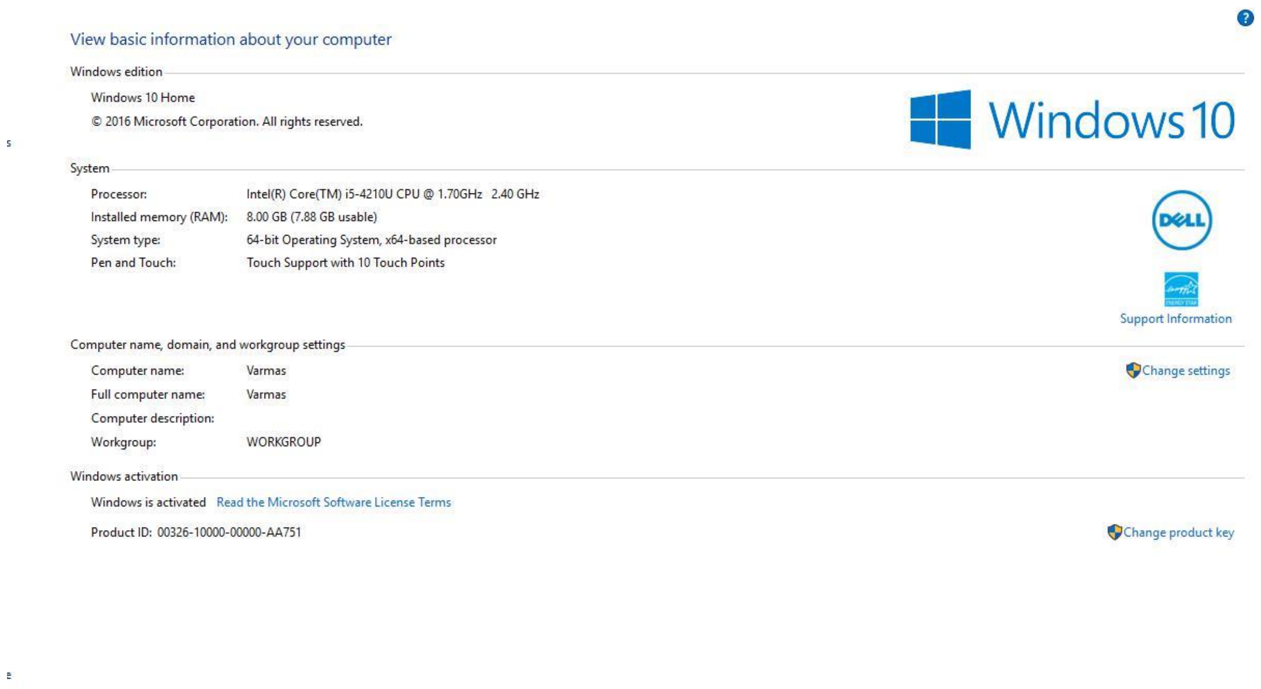
Programming Languages:

1. Environment: Eclipse
2. Tweets collection: Python
3. Programming: Java

API:

Twitter API.

Developing/Testing Machine’s specifications:



View basic information about your computer

Windows edition

Windows 10 Home

© 2016 Microsoft Corporation. All rights reserved.

System

Processor:	Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz 2.40 GHz
Installed memory (RAM):	8.00 GB (7.88 GB usable)
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	Touch Support with 10 Touch Points

Computer name, domain, and workgroup settings

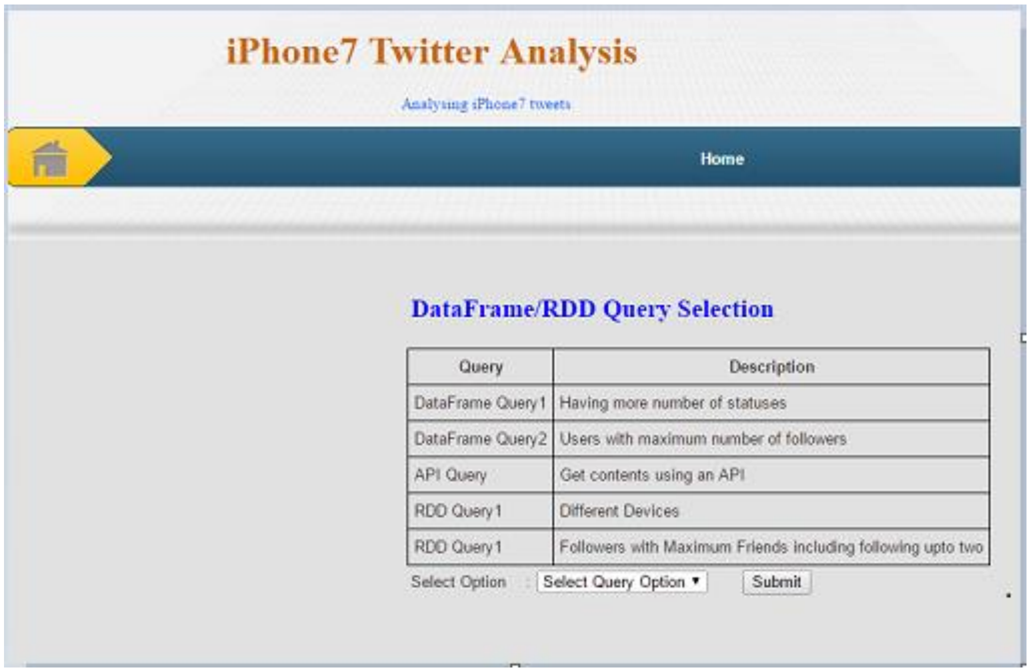
Computer name:	Varmas
Full computer name:	Varmas
Computer description:	
Workgroup:	WORKGROUP

Windows activation

Windows is activated Read the Microsoft Software License Terms

Product ID: 00326-10000-00000-AA751

HOME SCREEN:



iPhone7 Twitter Analysis

Analysing iPhone7 tweets

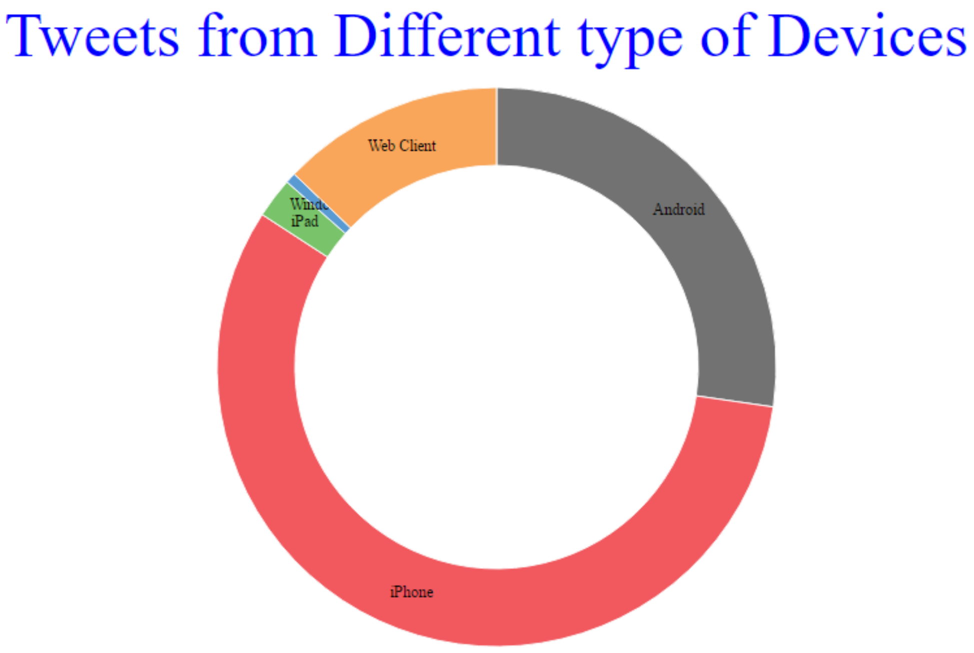
DataFrame/RDD Query Selection

Query	Description
DataFrame Query1	Having more number of statuses
DataFrame Query2	Users with maximum number of followers
API Query	Get contents using an API
RDD Query1	Different Devices
RDD Query1	Followers with Maximum Friends including following upto two

RDD Query 1: We used RDD to retrieve the output. This query gives the count of tweets grouped by different operating systems like android,windows,ios etc., of various users.

Query 1: "SELECT user.name,max(user.statuses_count) AS c FROM tweetTable " +
"GROUP BY user.name ORDER BY c desc limit 8

OUTPUT:



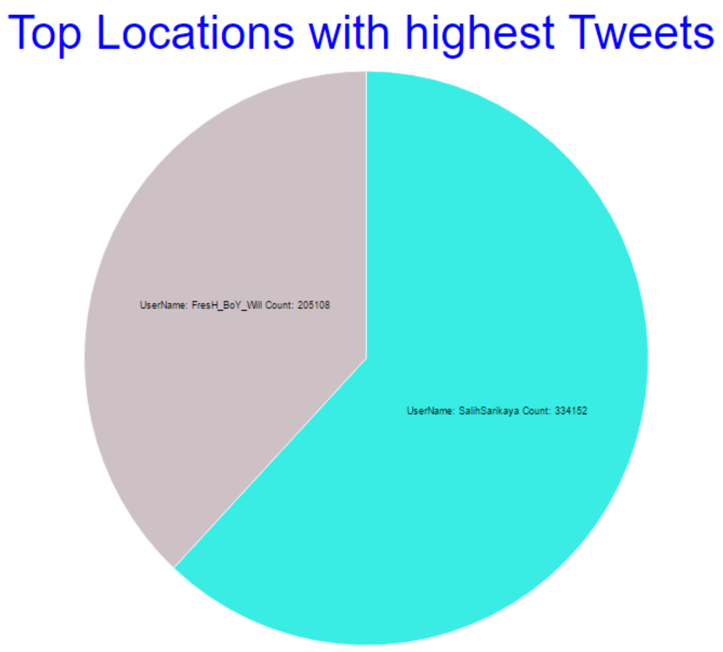
CODE:

```
 private static void query1(JavaSQLContext sqlContext) {

  try
  {

  File outputFile = new File("query1.csv");
 FileWriter fw= new FileWriter(outputFile);


 // Tweets from different kind of devices..!!

    JavaSchemaRDD count = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
     "WHERE source LIKE '%Android%'");
    JavaSchemaRDD count1 = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
"WHERE source LIKE '%iPhone%'");
    JavaSchemaRDD count2 = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
"WHERE source LIKE '%iPad%'");
    JavaSchemaRDD count3 = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
"WHERE source LIKE '%Windows%'");
    JavaSchemaRDD count4 = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
"WHERE source LIKE '%Web Client%'");
    //JavaSchemaRDD count5 = sqlContext.sql("SELECT  COUNT(*) AS c FROM tweetTable " +
// "WHERE source LIKE '%Golf%'");

 List<Row> Android=count.collect();
 String Android12=Android.toString();
 String Android1 = Android12.substring(Android12.indexOf("[") + 2, Android12.indexOf("]"));

 List<Row> iPhone=count1.collect();
 String iPhone12=iPhone.toString();
 String iPhone1 = iPhone12.substring(iPhone12.indexOf("[") + 2, iPhone12.indexOf("]"));

 List<Row> iPad=count2.collect();
 String iPad12=iPad.toString();
 String iPad1 = iPad12.substring(iPad12.indexOf("[") + 2, iPad12.indexOf("]"));

 List<Row> Windows=count3.collect();
 String Windows12=Windows.toString();
 String Windows1 = Windows12.substring(Windows12.indexOf("[") + 2, Windows12.indexOf("]"));

 List<Row> Web_Client=count4.collect();
 String Web_Client12=Web_Client.toString();
 String Web_Client1 = Web_Client12.substring(Web_Client12.indexOf("[") + 2, Web_Client12.indexOf("]"));

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

    fw.append("DeviceName");
fw.append(',');
fw.append("Count");
fw.append("\n");
fw.append("Android");
fw.append(',');
fw.append(Android1);
fw.append("\n");
fw.append("iPhone");
fw.append(',');
fw.append(iPhone1);
fw.append("\n");
fw.append("iPad");
fw.append(',');
fw.append(iPad1);
fw.append("\n");
fw.append("Windows");
fw.append(',');
fw.append(Windows1);
fw.append("\n");
fw.append("Web Client");
fw.append(',');
fw.append(Web_Client1);
fw.append("\n");
fw.close();
String path= outputFile.getAbsolutePath();
System.out.println(path);
 }
  catch (Exception exp)
  {
  System.out.println("Error");
  }

}
```

RDD Query 2: We used RDD to retrieve the output. This query gives the details of the number of friends greater than given amount.

Query 2: SELECT user.name, max(user.followers_count) AS c FROM tweetTable GROUP BY user.name ORDER BY c DESC limit 8.

OUTPUT:



CODE:

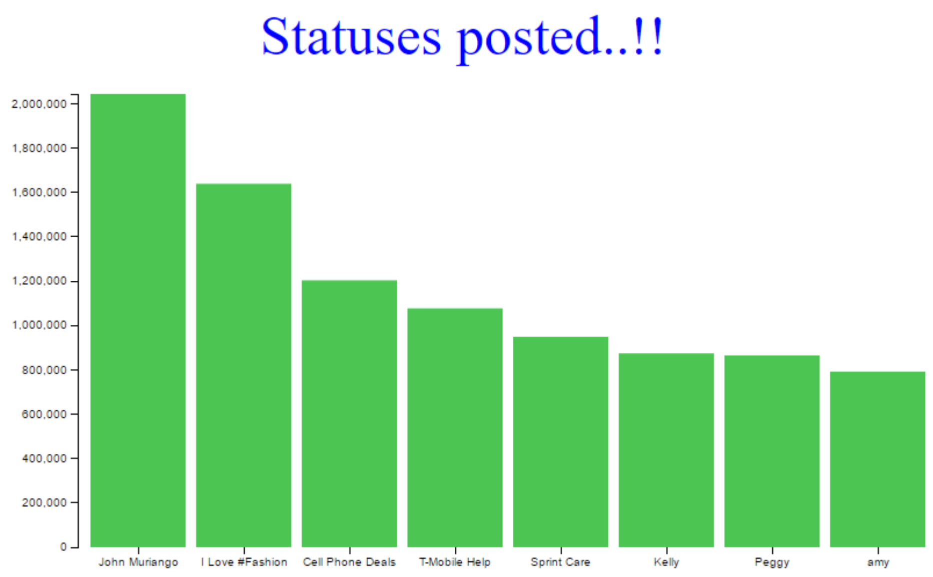
```
 private static void query2(JavaSQLContext sqlContext)
  {

  try
  {File outputFile = new File("query2.csv");
 FileWriter fw= new FileWriter(outputFile);

 //Users with maximum friends..!!

    JavaSchemaRDD count = sqlContext.sql("SELECT user.screen_name, max(user.friends_count) AS c
FROM tweetTable " +
     "WHERE user.friends_count>'150000'" +
                                        "group by user.screen_name order by c desc limit 20" );

    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(array.length);

    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)
  {
  }
   }
```

DATAFRAMES:

Query 3:

We used data frame to retrieve the output. This query is used to count the number of tweets by each user and it will show the count of top 8 users in the descending order.
Query 3: SELECT * from tweetTable desc limit 8 (API)

OUTPUT:



CODE:

```
public void DataframeQuery1()

{

java.util.Date date= new java.util.Date();

System.out.println("Start Query1");

 System.out.println(new Timestamp(date.getTime()));

        String pathToFile = url.toString();


         SparkConf conf = new SparkConf();

         conf.setAppName("Spark MultipleContest Test");

         conf.set("spark.driver.allowMultipleContexts", "true");

         conf.setMaster("local");


         JavaSparkContext sc = new JavaSparkContext(conf);


         SQLContext sqlContext = new SQLContext(sc);


         DataFrame tweets = sqlContext.read().json(pathToFile);


           tweets.registerTempTable("tweetTable");


      DataFrame followers = sqlContext.sql("SELECT user.name,max(user.statuses_count) AS c FROM
tweetTable " +
                                  "GROUP BY user.name ORDER BY c desc limit 8");


           Row[] rows = followers.collect();


           System.out.println(rows[0]);


         try

         {

          File outputFile = new
File("C:/Users/sravan/Downloads/SMT_EclipseWorkspace/SMT_EclipseWorkspace/PBproject/WebContent/Da
taFrameQuery1.csv");
```

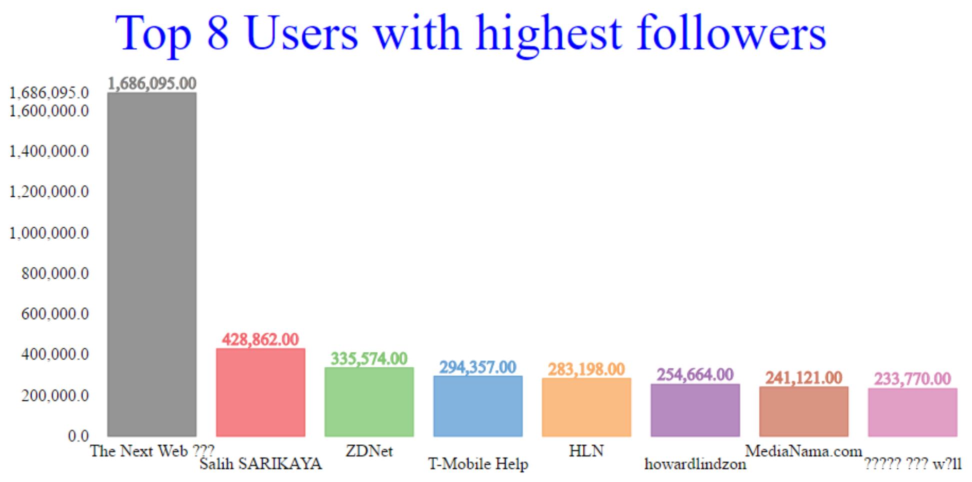
```
    FileWriter fw = new FileWriter(outputFile);



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

           for (int i=0;i<8;i++) {

            fw.append(rows[i].get(0).toString());
   fw.append(',');
   fw.append(rows[i].get(1).toString());
   fw.append("\n");

             }

           fw.close();

         }
         catch(Exception e)
         {

         }



        sc.stop();
        System.out.println("End Query1");
 System.out.println(new Timestamp(date.getTime()));
}
```

Query 4:

We used data frame to retrieve the output. This query gives the ranking of users based on number of followers in descending order.

```
Query 4:   SELECT  COUNT(*) AS c FROM tweetTable " +
     "WHERE source LIKE '%Android%'"
    SELECT  COUNT(*) AS c FROM tweetTable " +"WHERE source LIKE '%iPhone%'"
    SELECT  COUNT(*) AS c FROM tweetTable " +"WHERE source LIKE '%iPad%'"
    SELECT  COUNT(*) AS c FROM tweetTable " +"WHERE source LIKE '%Windows%'"
    SELECT  COUNT(*) AS c FROM tweetTable " +"WHERE source LIKE '%Web Client%'"
```

OUTPUT:



CODE:

```
public void DataframeQuery2()

{

java.util.Date date= new java.util.Date();

System.out.println("Start Query2");

 System.out.println(new Timestamp(date.getTime()));

 String pathToFile = url.toString();


        SparkConf conf = new SparkConf();

        conf.setAppName("Spark MultipleContest Test");

        conf.set("spark.driver.allowMultipleContexts", "true");

        conf.setMaster("local");
```

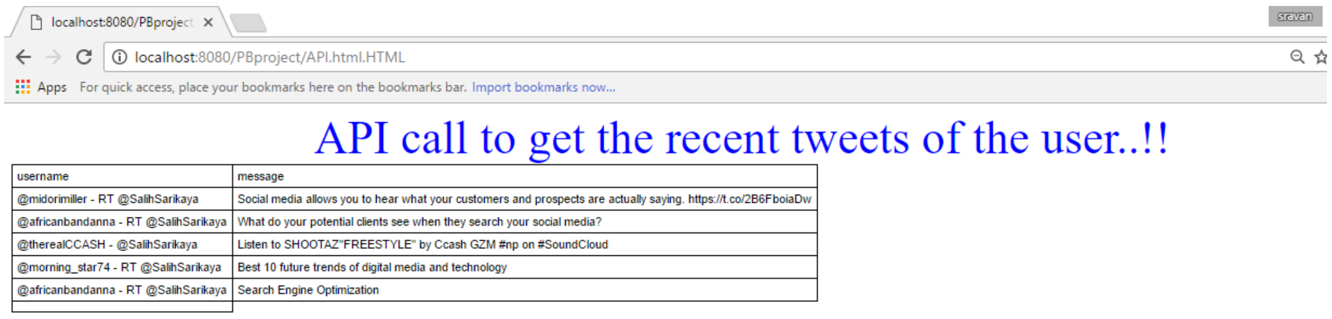
```
         JavaSparkContext sc = new JavaSparkContext(conf);

         SQLContext sqlContext = new SQLContext(sc);

         DataFrame tweets = sqlContext.read().json(pathToFile);

           tweets.registerTempTable("tweetTable");

         DataFrame times = sqlContext.sql("SELECT user.name, max(user.followers_count) AS c FROM
tweetTable GROUP BY user.name ORDER BY c DESC limit 8");

           Row[] rows = times.collect();
           System.out.println("Rows length"+rows.length);
           try{

   String rows123=rows.toString();
    System.out.println();
       String[] array = rows123.split("],");
       System.out.println("Array length"+array.length);
       System.out.println("Array[0]"+array[0]);

       FileWriter fw= new
FileWriter("C:/Users/sravan/Downloads/SMT_EclipseWorkspace/SMT_EclipseWorkspace/PBproject/WebCont
ent/DataFrameQuery2.csv");

    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)
  {
  }
        sc.stop();
        System.out.println("End Query2");
 System.out.println(new Timestamp(date.getTime()));
}
```

QUERY 5:

It requires calling public APIs . We take one of the outputs from the above 4 queries and fetch recent tweets by a particular user. The collected data is saved in .txt file.

Query 5: SELECT user.screen_name, max(user.friends_count) AS c FROM tweetTable " +
"WHERE user.friends_count>'150000'" +
"group by user.screen_name order by c desc limit 20;

OUTPUT:



localhost:8080/PBproject/API.html.HTML

API call to get the recent tweets of the user..!!

username	message
@midorimiller - RT @SalihSarikaya	Social media allows you to hear what your customers and prospects are actually saying. https://t.co/2B6FboiaDw
@africanbandanna - RT @SalihSarikaya	What do your potential clients see when they search your social media?
@therealCCASH - @SalihSarikaya	Listen to SHOOTAZ"FREESTYLE" by Ccash GZM #np on #SoundCloud
@morning_star74 - RT @SalihSarikaya	Best 10 future trends of digital media and technology
@africanbandanna - RT @SalihSarikaya	Search Engine Optimization

CODE:

```
ConfigurationBuilder cp= new ConfigurationBuilder();
cp.setDebugEnabled(true).setOAuthConsumerKey("BAQSR3afD856mF8NQOydbfYiI").setOAuthConsumerSec
ret("Vt8SImzSgM7ErdFOJDpFkKE2vgqQunAJXjIuVFSlJu0IptAYNj")
.setOAuthAccessToken("766182508698435585-
8SMuY4h6TtYggx4m0PyQ9UJTU31J0Xm").setOAuthAccessTokenSecret("l6rr6mQbX8vFZrNsBHkOwem6zst
7L9lQZZHKvdfUGKeU9");
TwitterFactory tf= new TwitterFactory(cp.build());
twitter4j.Twitter twitter= tf.getInstance();
try {
    Query query = new Query("FresH_BoY_Will");
    QueryResult result;
    do {
        result = twitter.search(query);
        List<Status> tweets = result.getTweets();
        for (Status tweet : tweets) {
            System.out.println("@" + tweet.getUser().getScreenName() + " - " + tweet.getText());
        }
    } while ((query = result.nextQuery()) != null);

Query query = new Query("SalihSarikaya");
    QueryResult result;
    do {
        result = twitter.search(query);
        List<Status> tweets = result.getTweets();
        for (Status tweet : tweets) {
            System.out.println("@" + tweet.getUser().getScreenName() + " - " + tweet.getText());
        }
    } while ((query = result.nextQuery()) != null);


    System.exit(0);
} catch (TwitterException te) {
    te.printStackTrace();
    System.out.println("Failed to search tweets: " + te.getMessage());
    System.exit(-1);
}

       response.sendRedirect("API.html");
break;
```