PB

Principles of Big Data Management Phase-3

Fall 2016

Submitted by: (Team 11)

Moulika Chadalavada(16234180)
Sampath Gattu(16231927)
Nikitha Kona(16235551)
Prathyusha(16231926)

Table of Contents

1. Introduction

1.1 About Twitter

Twitter is a massive social networking service that enables users to send data read short 140 character messages called ‘tweets’. More than 140 million active users publish over 400 million tweets every day. Twitter was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone and Noah Glass and launched in July 2006. Twitters speed and ease of publication have made it an important communication medium for people from all walks of life. This stream of messages from a variety of users contains information on an array of topics, including conventional new stories, events of local interest, opinions, real-time events. Twitter APIs provide access to tweets from a time range, from a user, with a keyword.

1.2 Proposed Project

We choose ‘Diseases’ as our topic to do big data analysis. Based on twitter tweets, we predicted some interesting analysis on Diseases using thousands of tweets tweeted by different people. First we collected the tweets from twitter API based on some key words related to Disease. After that, we analyzed the data that we have collected. By using the analysis, we written some interesting SQL queries useful to give a proper result for the analysis.

Here we used Spark to processing the twitter data. Because it has many advantages like

- Speed: Run Programs up to 100x faster than Hadoop Map reduce in memory.
- Ease of Use: Write applications quickly in Java, Scala, Python, R.
- Generality: Combine SQL, streaming, and complex analytics.
- Runs Everywhere: Spark runs on Hadoop, Mesos, standalone, or in the cloud

2. System Requirements

2.1 Software Requirements

- Python 3.5
- JDK 1.8
- Scala 2.11.8
- Intellij IDEA
- Apache Spark
- Eclipse

2.2 Language Requirements

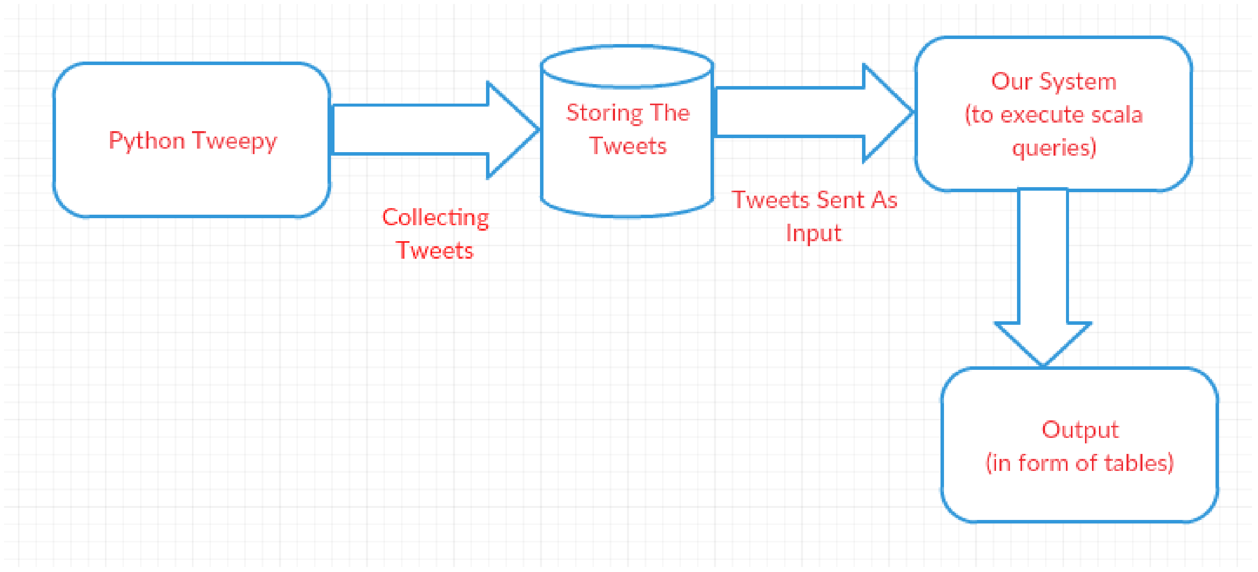
- Python
- Scala
- SQL
- HTML
- JSP
- Java
- Servlets

3. About the Theme

We have selected the theme diseases, which provides the analysis on the twitter tweets based on the queries. Initially tweets were collected using hashtags related to disease types such as Heart Stroke, Cancer, Chicken Pox etc. Using this data, the queries are written in Scala which gives analysis on which disease more tweets were done, which user made more number of tweets on which disease, which state people in USA are more involved in tweeting about the diseases, top languages using which tweets were posted. The extracted data is displayed in tables. Based on this data visualization is done in next phase.

4. System Architecture

First we generated credential for accessing twitter. By using these credentials, we wrote a python program to collect twitter tweets based on keywords related to food. Tweets were stored in a text file in a JSON format. We will give these JSON file to SQL queries for analysis with Spark, Intellij with Scala program with queries.



5. Collecting Twitter Tweets

5.1 Generating Access Tokens

First we generated keys for accessing twitter API. For this we need to register our application by using, http://apps.twitter.com. We generated access token, access token secret, consumer key, consumer secret

ACCESS_TOKEN = "1974127951-N1QRXNCsVCywXl67BU1Wx7VlJ1fw2TlScZDY07s"
ACCESS_SECRET = "zLLfZ4ZF9BxvgzjXAkjJFAVFOL4P1i0fLSaZzc6LrnqZJ"
CONSUMER_KEY = "RfqrQLUtEAvwqSmNpGjZydmOn"
CONSUMER_SECRET = "1XdUvxMDfcL5eE89oAo7ZO8UESv6H7HacNa9B0Y7cGFa5MiPjo"

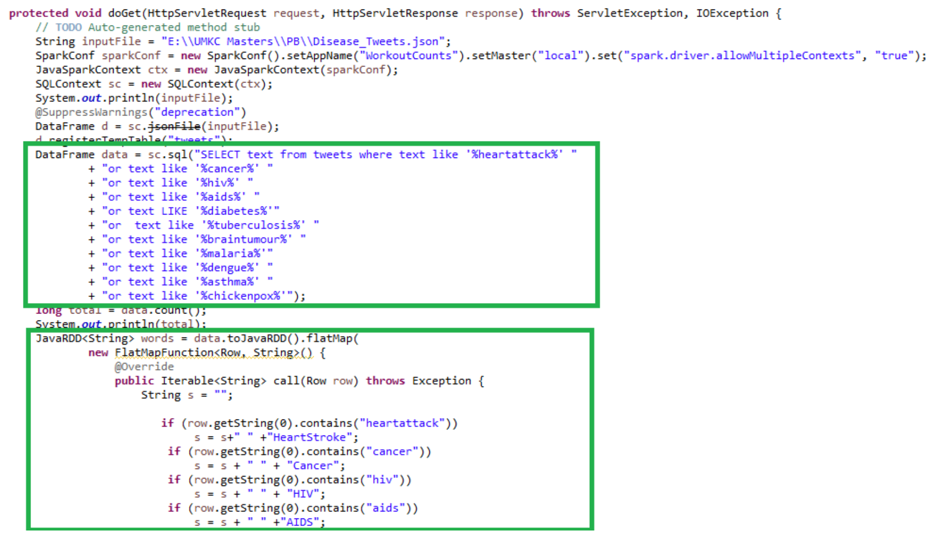
5.2 Streaming Twitter Tweets

After that we have written a python program for streaming twitter tweets. As our theme is related to 'Diseases' we used few hashtags such heartstroke,cancer,malaria,braintumour,chickenpox etc. From twitter, we have streamed almost 100000 instances. Tweets were stored in JSON (JavaScript Object Notation) format in a file.

6. Analyzing Twitter Data

6.1 Query 1: Popular Tweets on Different Diseases (Dynamically)

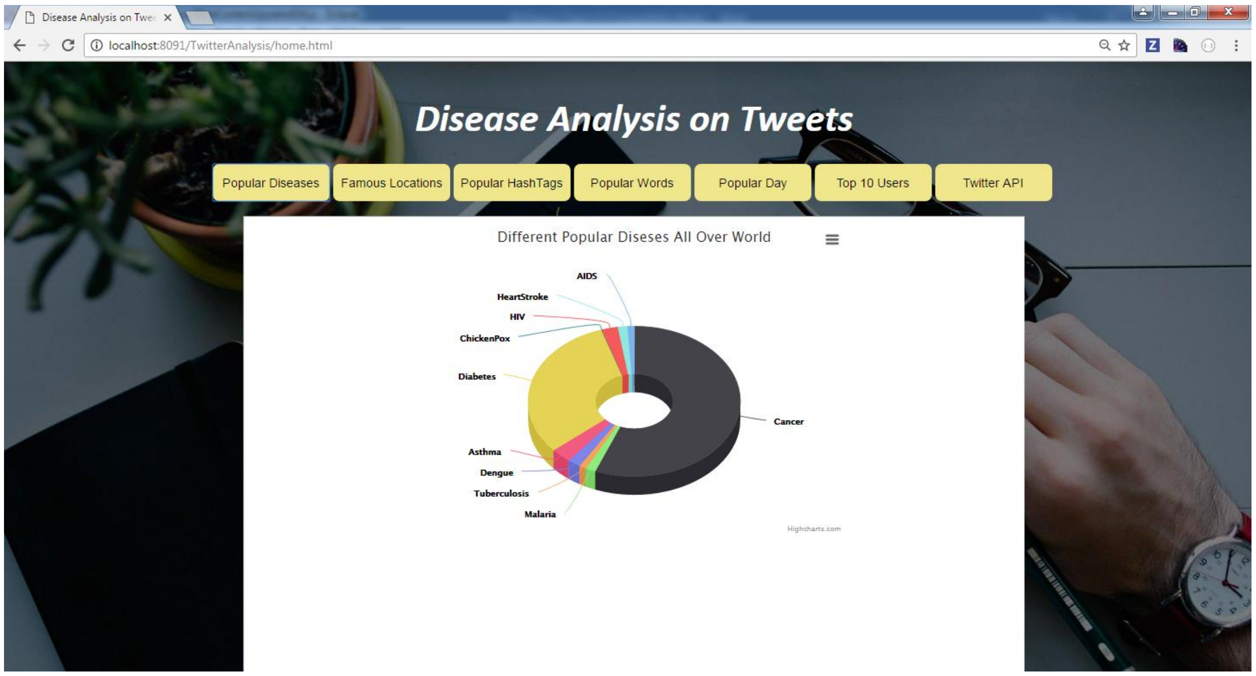
In this query, we are fetching the diseases and its tweets count in the file. This query is written using RDD, where we are fetching the count of diseases using hashtags using **filter** and the count is printed further.



```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    String inputFile = "E:\\UMKC Masters\\PB\\Disease_Tweets.json";
    SparkConf sparkConf = new SparkConf().setAppName("WorkoutCounts").setMaster("local").set("spark.driver.allowMultipleContexts", "true");
    JavaSparkContext ctx = new JavaSparkContext(sparkConf);
    SQLContext sc = new SQLContext(ctx);
    System.out.println(inputFile);
    @SuppressWarnings("deprecation")
    DataFrame d = sc.~~jsonFile~~(inputFile);
    d.registerTempTable("tweets");
    DataFrame data = sc.sql("SELECT text from tweets where text like '%heartattack%' "
            + "or text like '%cancer%' "
            + "or text like '%hiv%' "
            + "or text like '%aids%' "
            + "or text LIKE '%diabetes%'"
            + "or  text like '%tuberculosis%' "
            + "or text like '%braintumour%' "
            + "or text like '%malaria%'"
            + "or text like '%dengue%' "
            + "or text like '%asthma%' "
            + "or text like '%chickenpox%'");
    long total = data.count();
    System.out.println(total);
    JavaRDD<String> words = data.toJavaRDD().flatMap(
            new FlatMapFunction<Row, String>() {
                @Override
                public Iterable<String> call(Row row) throws Exception {
                    String s = "";

                        if (row.getString(0).contains("heartattack"))
                            s = s+" " +"HeartStroke";
                         if (row.getString(0).contains("cancer"))
                            s = s + " " + "Cancer";
                         if (row.getString(0).contains("hiv"))
                            s = s + " " + "HIV";
                         if (row.getString(0).contains("aids"))
                            s = s + " " +"AIDS";
```

```
        $(document).ready(function() {

        chart = new Highcharts.Chart({
            chart: {
                renderTo: 'container',
                type: 'pie',
                options3d: {
                    enabled: true,
                    alpha: 45
                }
            },
            title:
            {
                text:'Different Popular Diseses All Over World'
            },
            plotOptions: {
                pie: {

                    innerSize:100,
                    depth:45,
                    dataLabels: {
                        formatter: function(){
                            console.log(this);
                                 this.point.visible = true;
                                return this.key;
                            }
                           }
                        }

            },

            series: [{
                data: [
                    <%for (int i = 0; i < keys.size() - 1; i++) {%>
            {name:'<%=keys.get(i)%>', y:<%=values.get(i)%>},
        <%}%>
                    {name:'<%=keys.get(keys.size() - 1)%>', y:<%=values.get(values.size() - 1)%>}
                ]
            }]
        });
    });
```

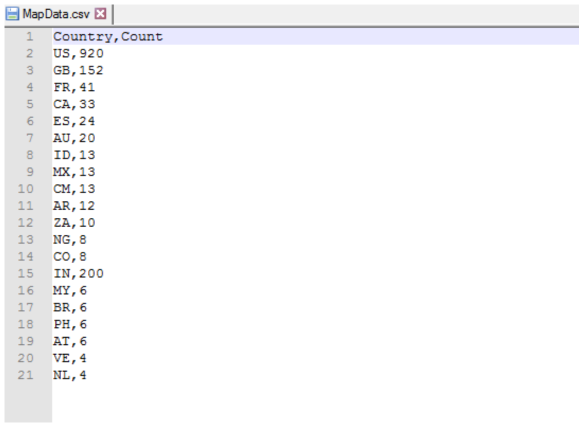
Visually, we used a PIE Chart to determine on which disease more number of tweets were posted. The visualization is dynamic, in which we used servlet to connect front end with code and it would be able to scale the PIE Chart accordingly.



6.2 Query 2: Countries that tweeted more on Diseases (Google Maps)

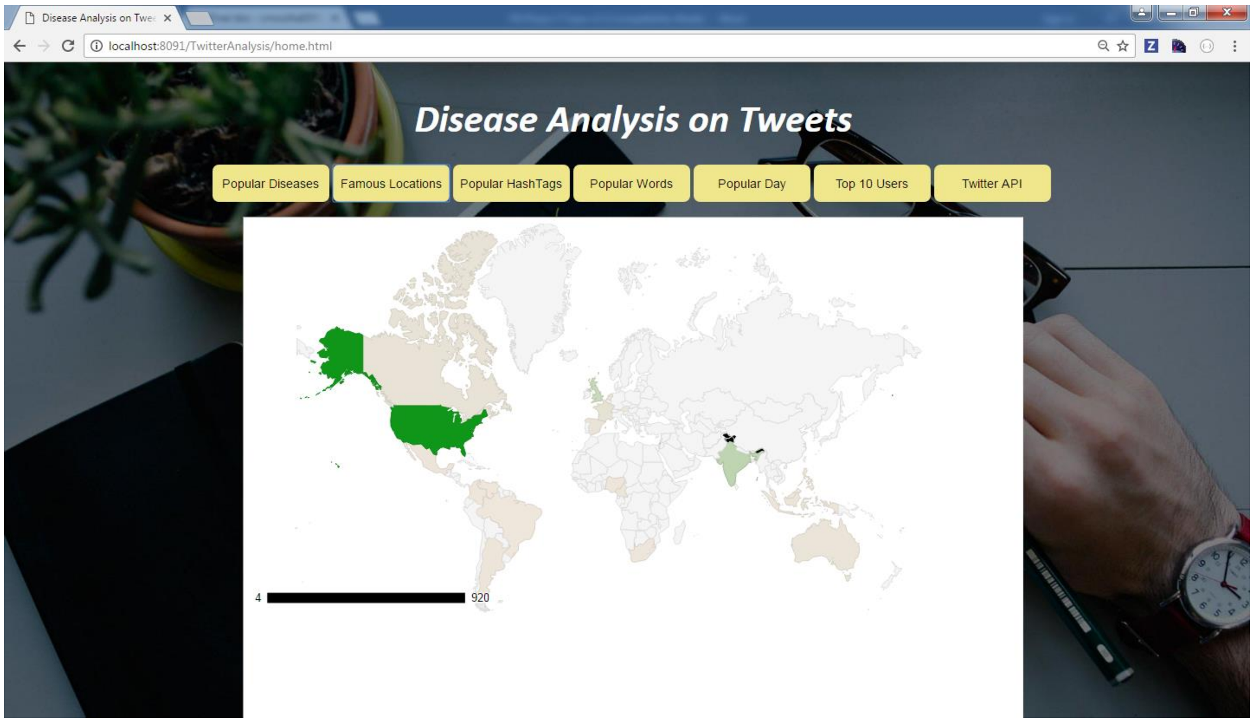
In this query, the top countries that tweeted more on diseases is fetched. First the location in tweets are fetched from tweets file and count is displayed as shown below. The data is stored in .csv format and the file is read and Visualization is done on Google Maps.

```
val q= sqlContext.sql("select place.country_code as location,count(*) as cnt from tweets " +
  "where place.country_code is not NULL group by place.country_code order by cnt desc ")
q.show()
```



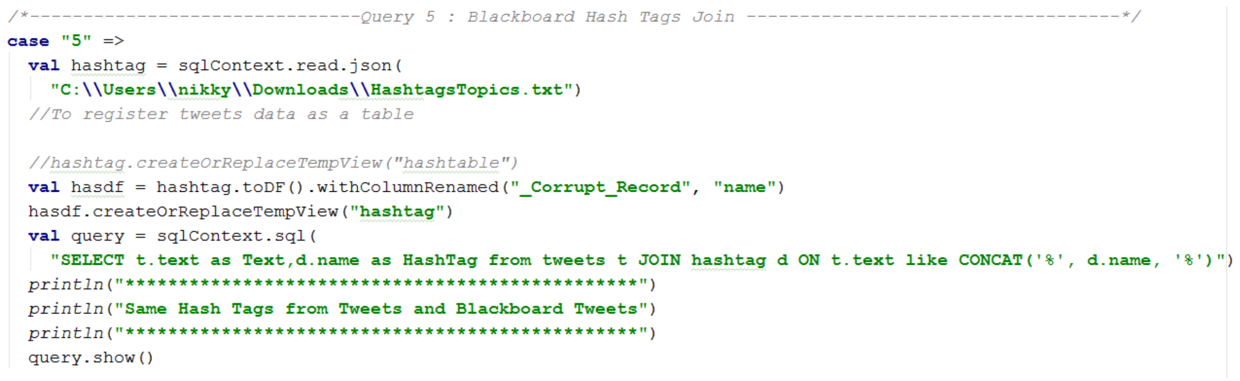
```
MapData.csv
Country,Count
US,920
GB,152
FR,41
CA,33
ES,24
AU,20
ID,13
MX,13
CM,13
AR,12
ZA,10
NG,8
CO,8
IN,200
MY,6
BR,6
PH,6
AT,6
VE,4
NL,4
```

Visually, we used a google Map Chart to determine which country tweeted more number of tweets. The visualization isn't dynamics though, in which we first run the scala code in Intellij and saved the output in a file and finally passed this file to Map chart.

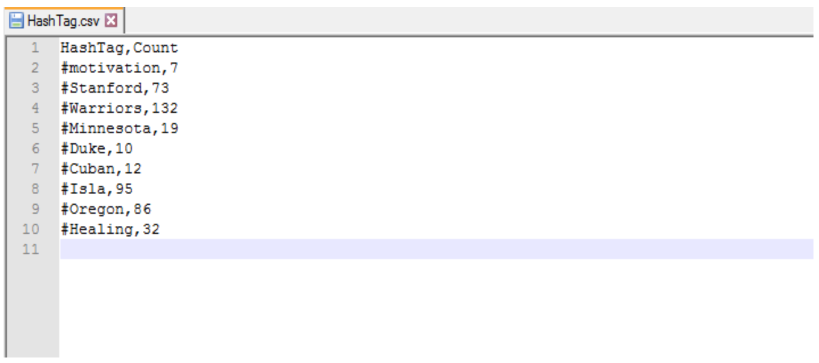


6.3 Query 3: Popular Hashtags

In this query, we took popular hash tags text file from blackboard and performed JOIN operation with hash tags from diseases tweets file. The fetched data is stored in .csv format to do visualization.



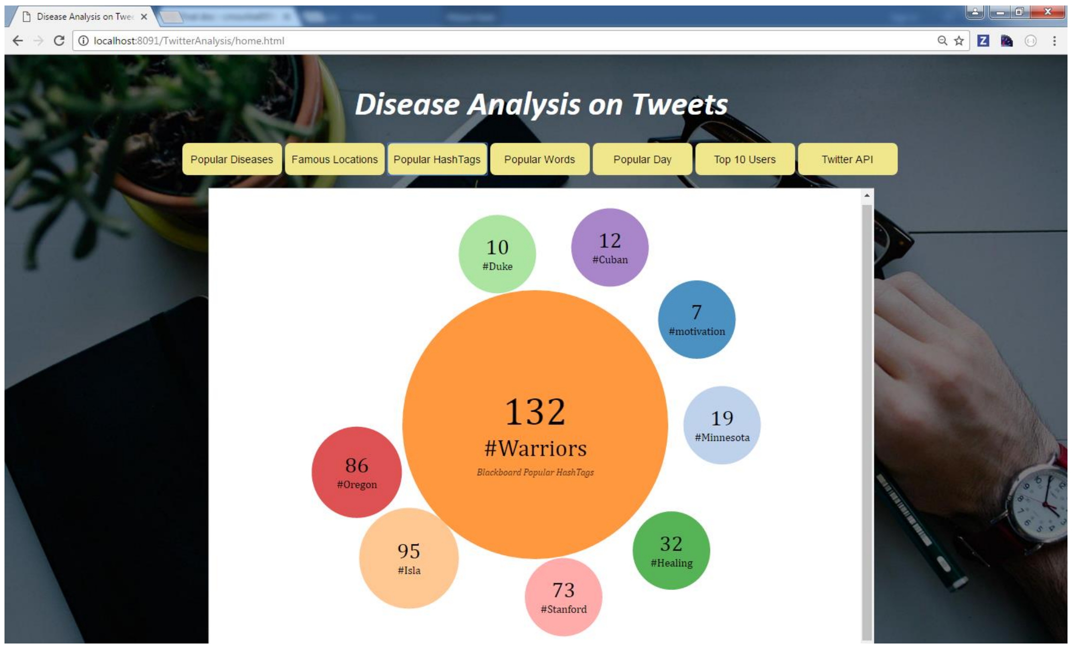
```
/*-------------------------------Query 5 : Blackboard Hash Tags Join -----------------------------------*/
case "5" =>
  val hashtag = sqlContext.read.json(
    "C:\\Users\\nikky\\Downloads\\HashtagsTopics.txt")
  //To register tweets data as a table

  //hashtag.createOrReplaceTempView("hashtable")
  val hasdf = hashtag.toDF().withColumnRenamed("_Corrupt_Record", "name")
  hasdf.createOrReplaceTempView("hashtag")
  val query = sqlContext.sql(
    "SELECT t.text as Text,d.name as HashTag from tweets t JOIN hashtag d ON t.text like CONCAT('%', d.name, '%')")
  println("**************************************************")
  println("Same Hash Tags from Tweets and Blackboard Tweets")
  println("**************************************************")
  query.show()
```



HashTag.csv

```
HashTag,Count
#motivation,7
#Stanford,73
#Warriors,132
#Minnesota,19
#Duke,10
#Cuban,12
#Isla,95
#Oregon,86
#Healing,32
```

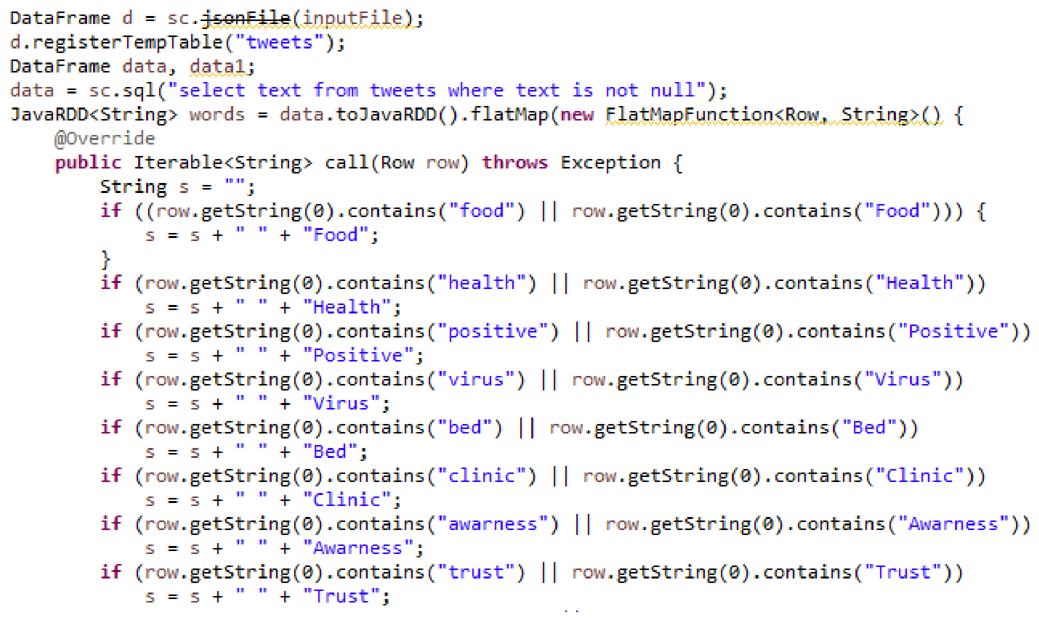
The data in csv file is used to do visualization and the representation is shown in **Bubble Chart.** According to below image, **#Warriors** tag is frequently occurred HashTag in tweets file.



6.4 Query 4: Most Popular Tweeted Words (Dynamically)

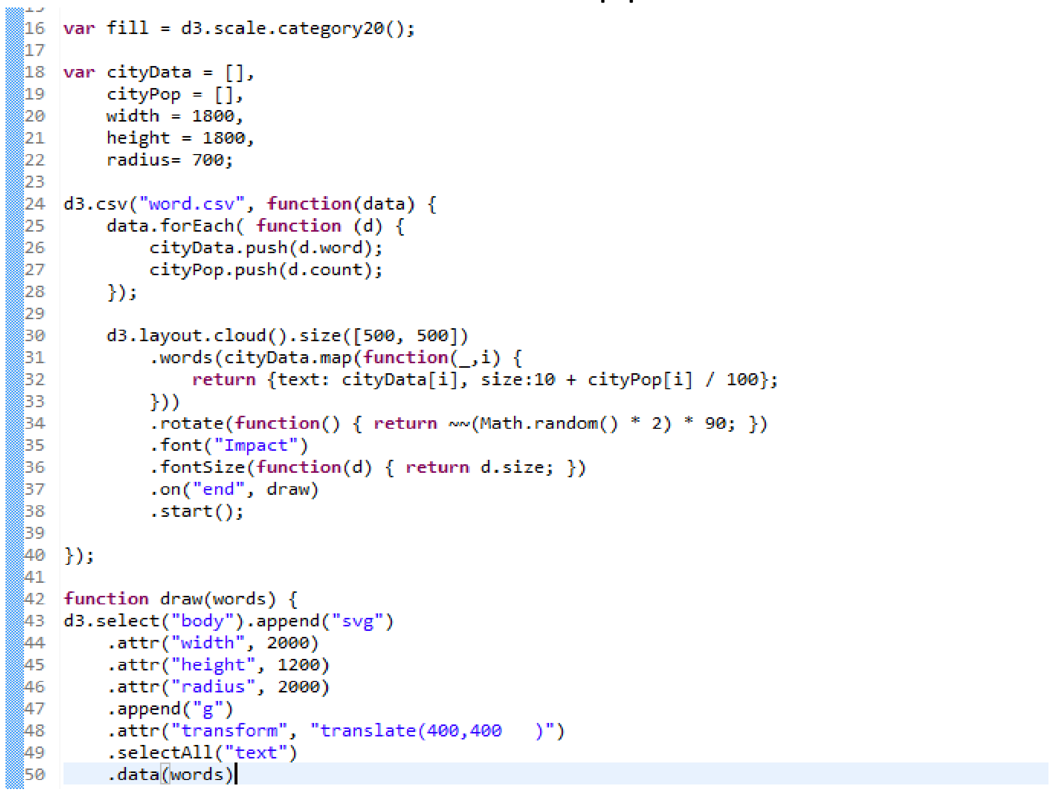
In this query, most occurring words in tweets on diseases is fetched. On the fetched data visualization is done dynamically.

Below query is used to fetch most popular words



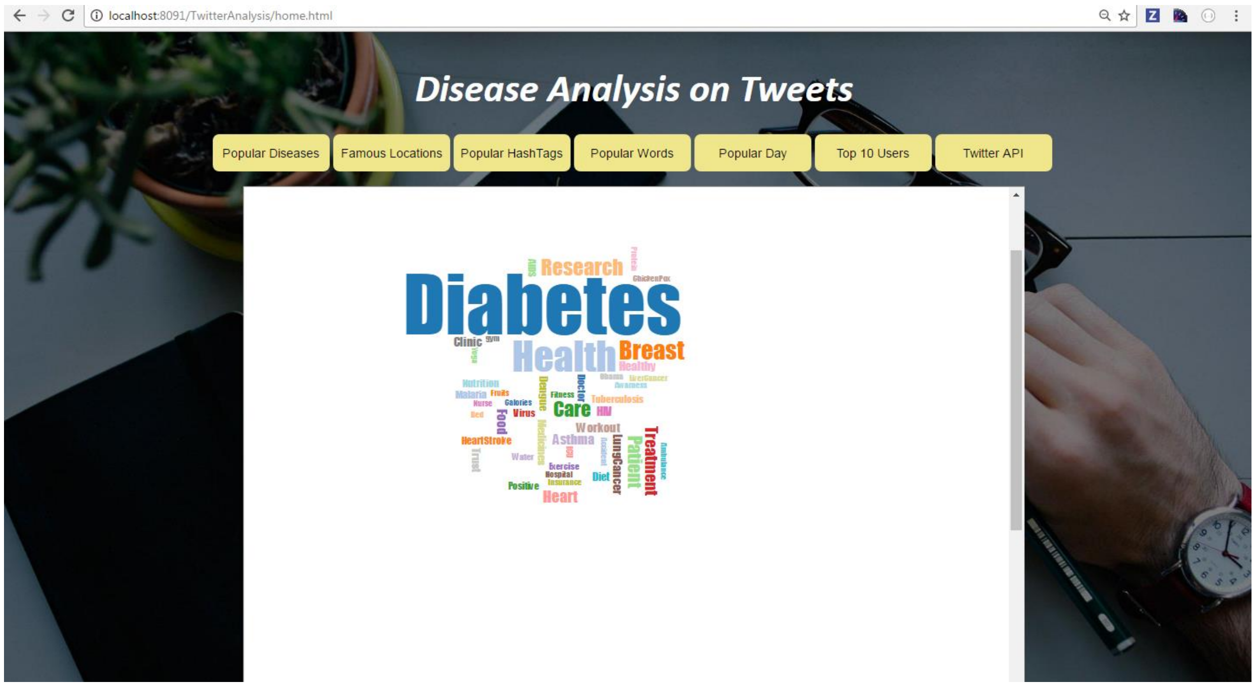
```
DataFrame d = sc.~~jsonFile~~(inputFile);
d.registerTempTable("tweets");
DataFrame data, data1;
data = sc.sql("select text from tweets where text is not null");
JavaRDD<String> words = data.toJavaRDD().flatMap(new FlatMapFunction<Row, String>() {
    @Override
    public Iterable<String> call(Row row) throws Exception {
        String s = "";
        if ((row.getString(0).contains("food") || row.getString(0).contains("Food"))) {
            s = s + " " + "Food";
        }
        if (row.getString(0).contains("health") || row.getString(0).contains("Health"))
            s = s + " " + "Health";
        if (row.getString(0).contains("positive") || row.getString(0).contains("Positive"))
            s = s + " " + "Positive";
        if (row.getString(0).contains("virus") || row.getString(0).contains("Virus"))
            s = s + " " + "Virus";
        if (row.getString(0).contains("bed") || row.getString(0).contains("Bed"))
            s = s + " " + "Bed";
        if (row.getString(0).contains("clinic") || row.getString(0).contains("Clinic"))
            s = s + " " + "Clinic";
        if (row.getString(0).contains("awarness") || row.getString(0).contains("Awarness"))
            s = s + " " + "Awarness";
        if (row.getString(0).contains("trust") || row.getString(0).contains("Trust"))
            s = s + " " + "Trust";
```

Visualization code for most popular words



```
var fill = d3.scale.category20();

var cityData = [],
    cityPop = [],
    width = 1800,
    height = 1800,
    radius= 700;

d3.csv("word.csv", function(data) {
    data.forEach( function (d) {
        cityData.push(d.word);
        cityPop.push(d.count);
    });

    d3.layout.cloud().size([500, 500])
        .words(cityData.map(function(_,i) {
            return {text: cityData[i], size:10 + cityPop[i] / 100};
        }))
        .rotate(function() { return ~~(Math.random() * 2) * 90; })
        .font("Impact")
        .fontSize(function(d) { return d.size; })
        .on("end", draw)
        .start();

});

function draw(words) {
d3.select("body").append("svg")
    .attr("width", 2000)
    .attr("height", 1200)
    .attr("radius", 2000)
    .append("g")
    .attr("transform", "translate(400,400    )")
    .selectAll("text")
    .data(words)
```

Based on the words and its counts the data is dynamically sent to wordcloud.jsp where the visualization logic is performed and below visualization is displayed on UI.

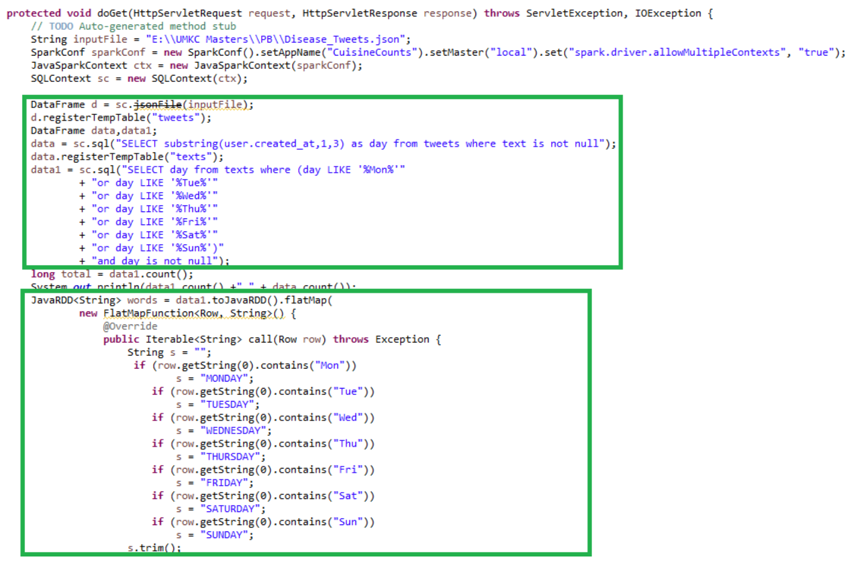


Query Analysis: According to the query, more popular word used in tweets is Diabetes

6.5 Query 5: On which day of week, more tweets are done on diseases (Dynamically)

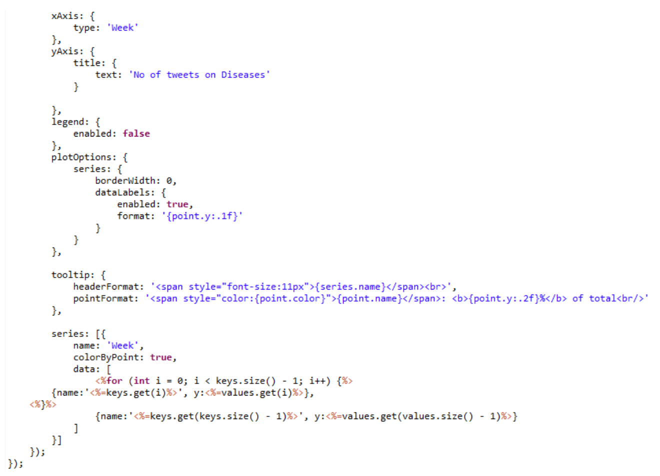
In this query, data is fetched based on which day of week more tweets are done on Diseases. Initially created_at is fetched from tweets file and count of tweets is done on each day of week.

Java Code to fetch Week and Count from Tweets

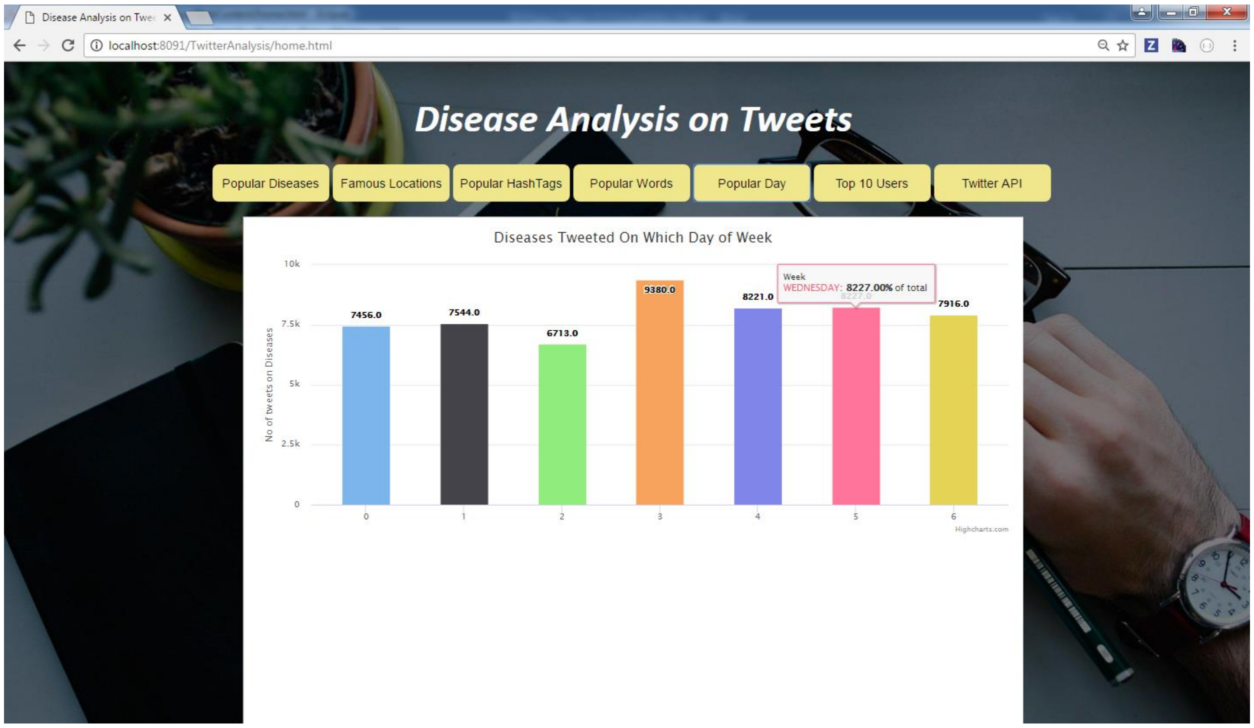


```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    String inputFile = "E:\\UMKC Masters\\PB\\Disease_Tweets.json";
    SparkConf sparkConf = new SparkConf().setAppName("CuisineCounts").setMaster("local").set("spark.driver.allowMultipleContexts", "true");
    JavaSparkContext ctx = new JavaSparkContext(sparkConf);
    SQLContext sc = new SQLContext(ctx);

    DataFrame d = sc.~~jsonFile~~(inputFile);
    d.registerTempTable("tweets");
    DataFrame data,data1;
    data = sc.sql("SELECT substring(user.created_at,1,3) as day from tweets where text is not null");
    data.registerTempTable("texts");
    data1 = sc.sql("SELECT day from texts where (day LIKE '%Mon%'"
            + "or day LIKE '%Tue%'"
            + "or day LIKE '%Wed%'"
            + "or day LIKE '%Thu%'"
            + "or day LIKE '%Fri%'"
            + "or day LIKE '%Sat%'"
            + "or day LIKE '%Sun%')"
            + "and day is not null");
    long total = data1.count();
    System.out.println(data1.count() +" " + data.count());
    JavaRDD<String> words = data1.toJavaRDD().flatMap(
            new FlatMapFunction<Row, String>() {
                @Override
                public Iterable<String> call(Row row) throws Exception {
                    String s = "";
                     if (row.getString(0).contains("Mon"))
                            s = "MONDAY";
                        if (row.getString(0).contains("Tue"))
                            s = "TUESDAY";
                        if (row.getString(0).contains("Wed"))
                            s = "WEDNESDAY";
                        if (row.getString(0).contains("Thu"))
                            s = "THURSDAY";
                        if (row.getString(0).contains("Fri"))
                            s = "FRIDAY";
                        if (row.getString(0).contains("Sat"))
                            s = "SATURDAY";
                        if (row.getString(0).contains("Sun"))
                            s = "SUNDAY";
                    s.trim();
```

JSP Code to do Visualization on Week and its count



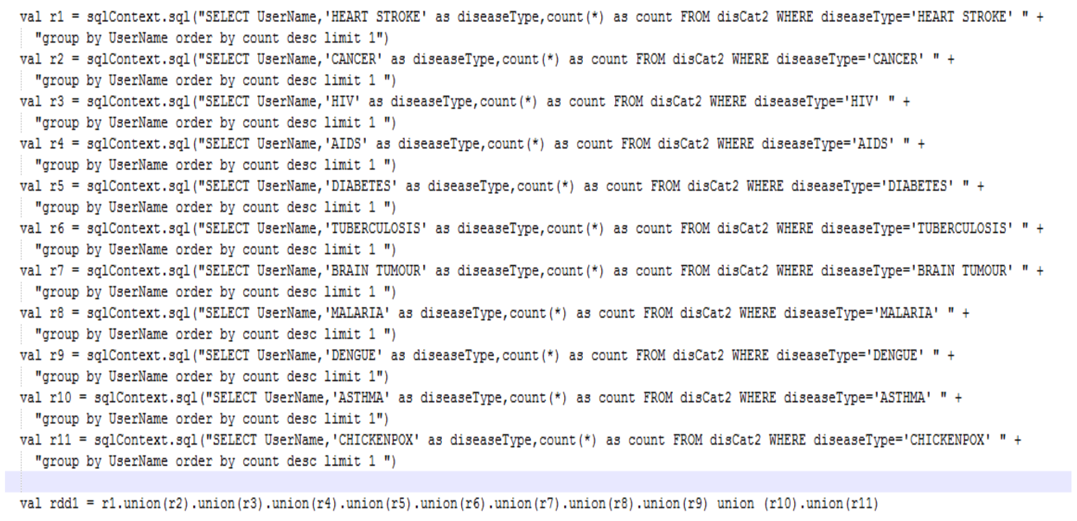
```
        xAxis: {
            type: 'Week'
        },
        yAxis: {
            title: {
                text: 'No of tweets on Diseases'
            }

        },
        legend: {
            enabled: false
        },
        plotOptions: {
            series: {
                borderWidth: 0,
                dataLabels: {
                    enabled: true,
                    format: '{point.y:.1f}'
                }
            }
        },

        tooltip: {
            headerFormat: '<span style="font-size:11px">{series.name}</span><br>',
            pointFormat: '<span style="color:{point.color}">{point.name}</span>: <b>{point.y:.2f}%</b> of total<br/>'
        },

        series: [{
            name: 'Week',
            colorByPoint: true,
            data: [
                <%for (int i = 0; i < keys.size() - 1; i++) {%>
        {name:'<%=keys.get(i)%>', y:<%=values.get(i)%>},
    <%}%>
                {name:'<%=keys.get(keys.size() - 1)%>', y:<%=values.get(values.size() - 1)%>}
            ]
        }]
    });
});
```



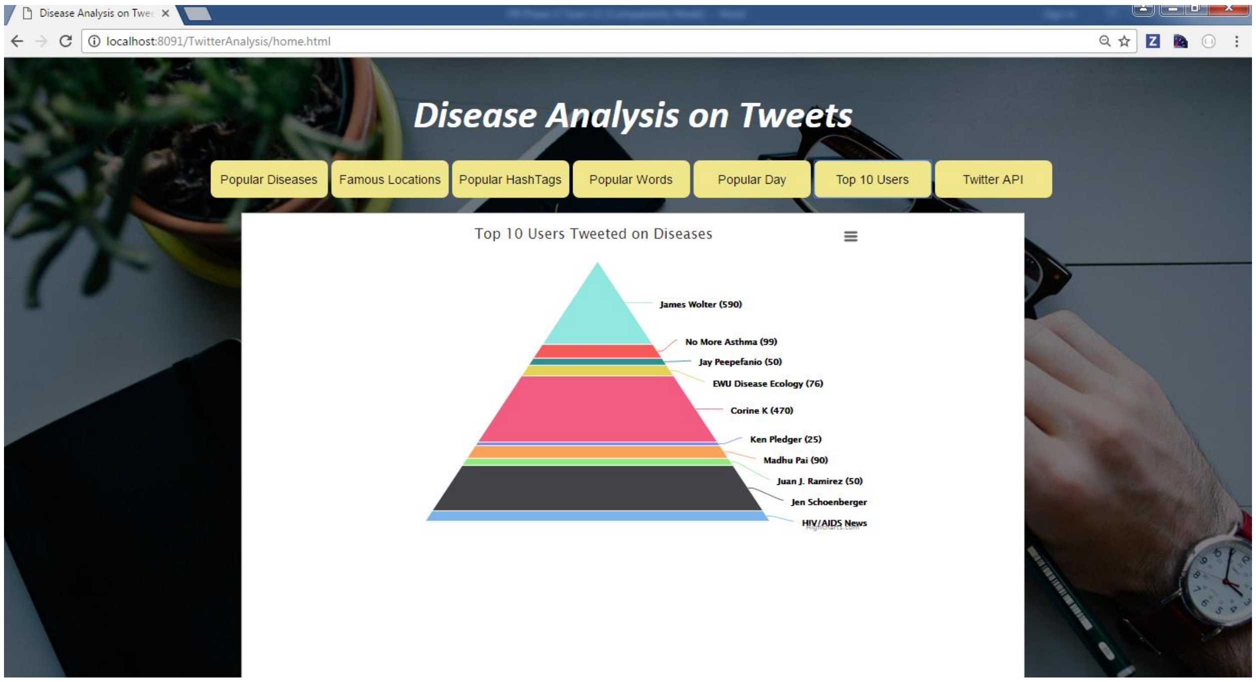
Query Analysis: According to the query, on Thursday more tweets are done and less tweets are done on Saturday

6.6 Query 6: Top 10 Users Tweeted on Diseases

In this query the we are fetching top 10 users who tweeted more on diseases. This query is written using RDD. Initially for each disease, the top tweeted user is fetched and **UNION** RDD is used to club all the diseases. The results are stored in .csv file to do visualization



```
val r1 = sqlContext.sql("SELECT UserName,'HEART STROKE' as diseaseType,count(*) as count FROM disCat2 WHERE diseaseType='HEART STROKE' " +
  "group by UserName order by count desc limit 1")
val r2 = sqlContext.sql("SELECT UserName,'CANCER' as diseaseType,count(*) as count FROM disCat2 WHERE diseaseType='CANCER' " +
  "group by UserName order by count desc limit 1 ")
val r3 = sqlContext.sql("SELECT UserName,'HIV' as diseaseType,count(*) as count FROM disCat2 WHERE diseaseType='HIV' " +
  "group by UserName order by count desc limit 1 ")
val r4 = sqlContext.sql("SELECT UserName,'AIDS' as diseaseType,count(*) as count FROM disCat2 WHERE diseaseType='AIDS' " +
  "group by UserName order by count desc limit 1 ")
val r5 = sqlContext.sql("SELECT UserName,'DIABETES' as diseaseType,count(*) as count FROM disCat2 WHERE diseaseType='DIABETES' " +
  "group by UserName order by count desc limit 1 ")
val r6 = sqlContext.sql("SELECT UserName,'TUBERCULOSIS' as diseaseType,count(*) as count FROM disCat2 WHERE diseaseType='TUBERCULOSIS' " +
  "group by UserName order by count desc limit 1 ")
val r7 = sqlContext.sql("SELECT UserName,'BRAIN TUMOUR' as diseaseType,count(*) as count FROM disCat2 WHERE diseaseType='BRAIN TUMOUR' " +
  "group by UserName order by count desc limit 1")
val r8 = sqlContext.sql("SELECT UserName,'MALARIA' as diseaseType,count(*) as count FROM disCat2 WHERE diseaseType='MALARIA' " +
  "group by UserName order by count desc limit 1 ")
val r9 = sqlContext.sql("SELECT UserName,'DENGUE' as diseaseType,count(*) as count FROM disCat2 WHERE diseaseType='DENGUE' " +
  "group by UserName order by count desc limit 1")
val r10 = sqlContext.sql("SELECT UserName,'ASTHMA' as diseaseType,count(*) as count FROM disCat2 WHERE diseaseType='ASTHMA' " +
  "group by UserName order by count desc limit 1")
val r11 = sqlContext.sql("SELECT UserName,'CHICKENPOX' as diseaseType,count(*) as count FROM disCat2 WHERE diseaseType='CHICKENPOX' " +
  "group by UserName order by count desc limit 1 ")

val rdd1 = r1.union(r2).union(r3).union(r4).union(r5).union(r6).union(r7).union(r8).union(r9) union (r10).union(r11)
```



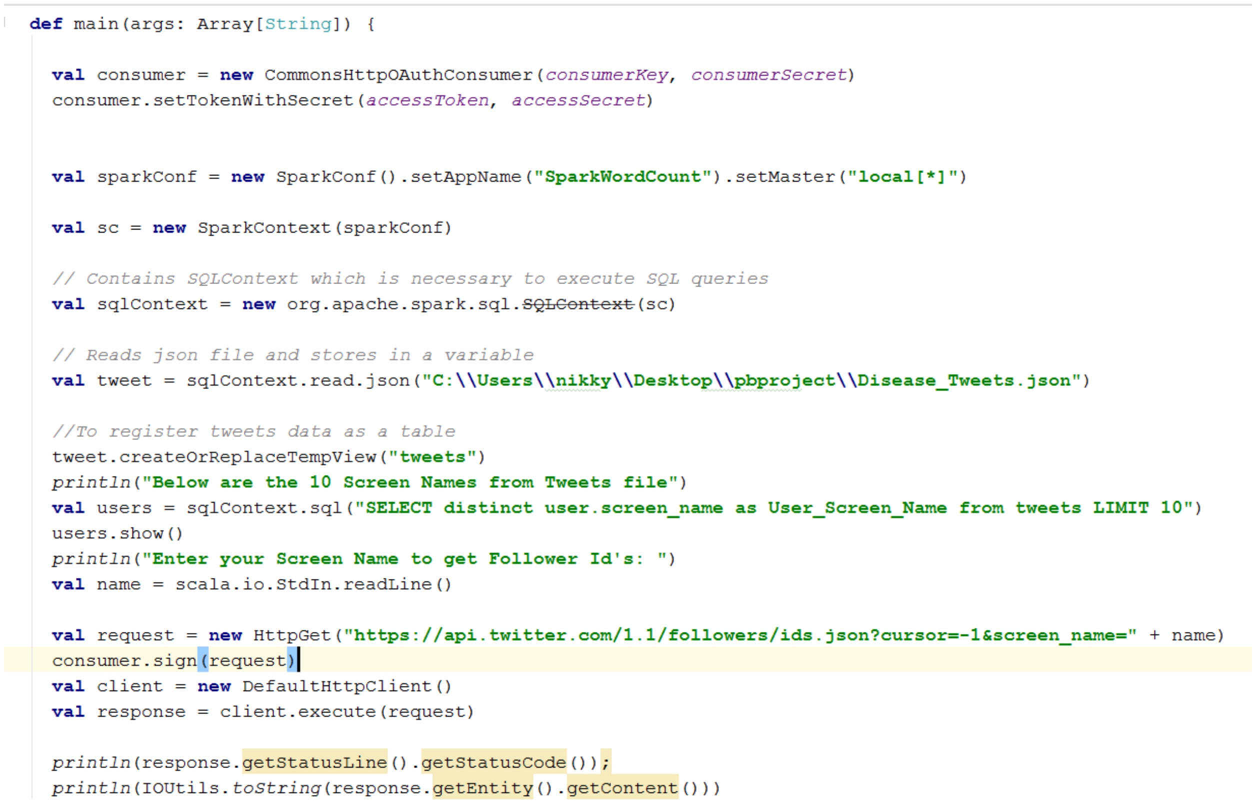
Query Analysis: According to the query, user who tweeted more on diseases is James Wolter.

6.7 Query 7: Follower Id’s count using Twitter API

Twitter Get Followers ids API is used. A query to display five screen names from the tweets file is written. When the query is executed a table with ten screen names is displayed in the table.

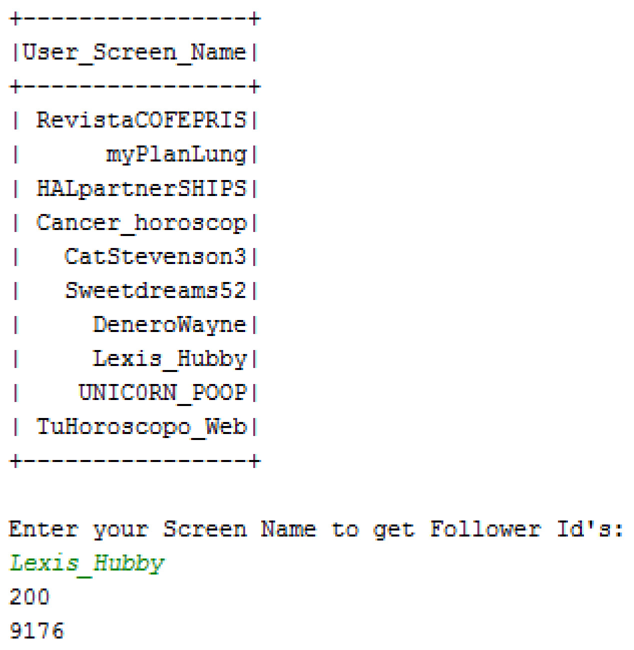
Val request = new HttpGet("https://api.twitter.com/1.1/followers/ids.json?cursor=-1&screen_name=" + name)

First the user is given a Choice to enter a screen name of his choice. Once the screen name has been inputted the follower's id



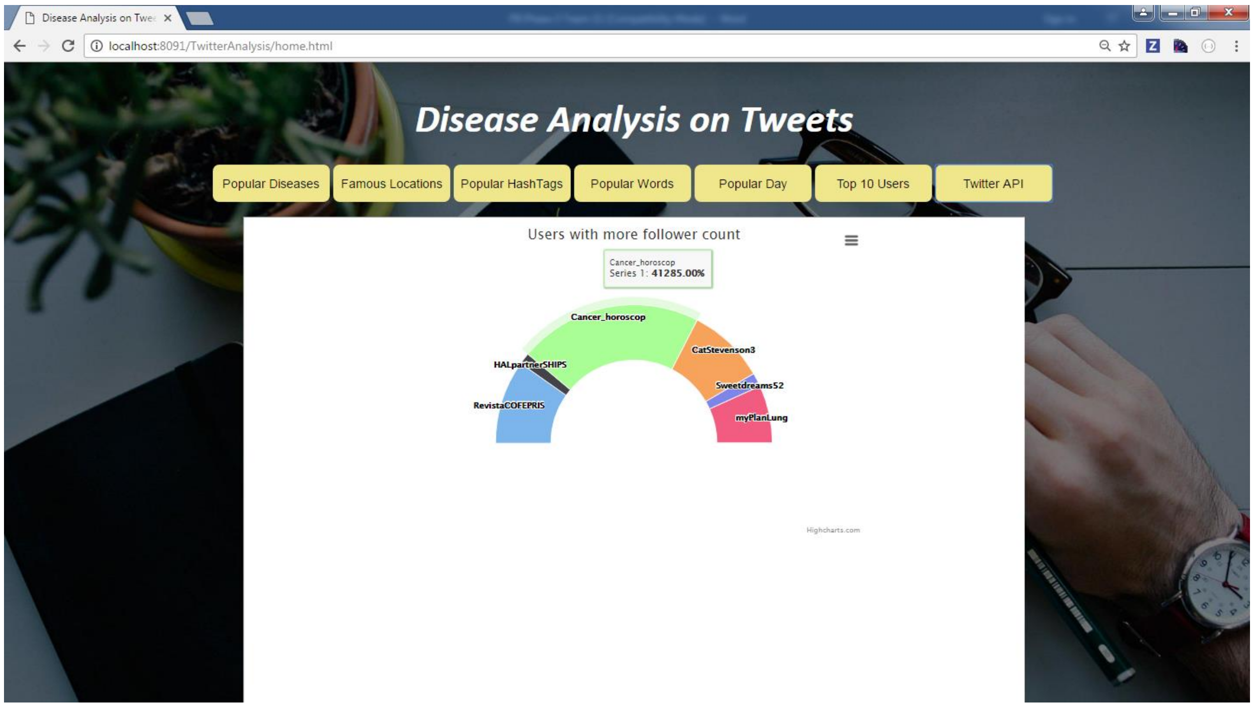
```
def main(args: Array[String]) {

  val consumer = new CommonsHttpOAuthConsumer(consumerKey, consumerSecret)
  consumer.setTokenWithSecret(accessToken, accessSecret)


  val sparkConf = new SparkConf().setAppName("SparkWordCount").setMaster("local[*]")

  val sc = new SparkContext(sparkConf)

  // Contains SQLContext which is necessary to execute SQL queries
  val sqlContext = new org.apache.spark.sql.~~SQLContext~~(sc)

  // Reads json file and stores in a variable
  val tweet = sqlContext.read.json("C:\\Users\\nikky\\Desktop\\pbproject\\Disease_Tweets.json")

  //To register tweets data as a table
  tweet.createOrReplaceTempView("tweets")
  println("Below are the 10 Screen Names from Tweets file")
  val users = sqlContext.sql("SELECT distinct user.screen_name as User_Screen_Name from tweets LIMIT 10")
  users.show()
  println("Enter your Screen Name to get Follower Id's: ")
  val name = scala.io.StdIn.readLine()

  val request = new HttpGet("https://api.twitter.com/1.1/followers/ids.json?cursor=-1&screen_name=" + name)
  consumer.sign(request)
  val client = new DefaultHttpClient()
  val response = client.execute(request)

  println(response.getStatusLine().getStatusCode());
  println(IOUtils.toString(response.getEntity().getContent()))
```

Once screen name **RevistaCOFEPRIS** is entered the follower id's count are displayed as shown below



```
+----------------+
|User_Screen_Name|
+----------------+
| RevistaCOFEPRIS|
|      myPlanLung|
| HALpartnerSHIPS|
| Cancer_horoscop|
|   CatStevenson3|
|   Sweetdreams52|
|     DeneroWayne|
|     Lexis_Hubby|
|    UNICORN_POOP|
| TuHoroscopo_Web|
+----------------+

Enter your Screen Name to get Follower Id's:
Lexis_Hubby
200
9176
```

The screen name and the count is displayed using the PIE Chart in visualization as below.



Query Analysis: According to the query, user who has more follower's is cancer_horoscop

Tweets Location:

https://www.dropbox.com/s/04zebrisw6jm6n0/Disease_Tweets.json?dl=0

Source Code:

https://github.com/cmoulika009/Principles-of-Big-Data-Management/tree/master/PB%20Phase-3-%20Team-11/TwitterAnalysis

7. References

https://apps.twitter.com

http://www.tutorialspoint.com

https://www.jetbrains.com/idea/