PRICIPLES OF BIG DATA MANAGEMENT (CS-5540)

PROJECT REPORT

Recreational Activities Trends across World Tweets Analysis

TEAM NUMBER 18

TEAM MEMBERS:

RASHMI TRIPATHI (rtrh6)

RITIKA CHOWDRI (rcc92)

NITIN BHANDARI (nbgm3)

HARMEET SINGH (hsp79)

Table of Contents

Introduction

Theme: Recreational Activities Trends across World

This project is a powerful web application to help you comparing you to others on many fronts in terms of recreation activities followed by people around the world. Recreation is part of any human life. All human beings need some break from their hectic schedule to spend time on activities they enjoy like music, sports, dance, watching television etc. This boost their energy multifold.

As part of this project we have collected 130000+ instances of tweets across the globe using Twitter APIs in form of JSON file. The JSON(JavaScript Object Notation) data is one of lightweight data-interchange format. It is very easy to interpret by human beings as well as machines.

Now on top of this tweets we are analyzing various trends across the globe using Spark SQL. We have designed meaningful queries to analyze various recreational activities people of various age groups choose and talk about, across the world by using twitter data and designing meaningful queries.

Some of the recreational activities we analyzed are as, we included:

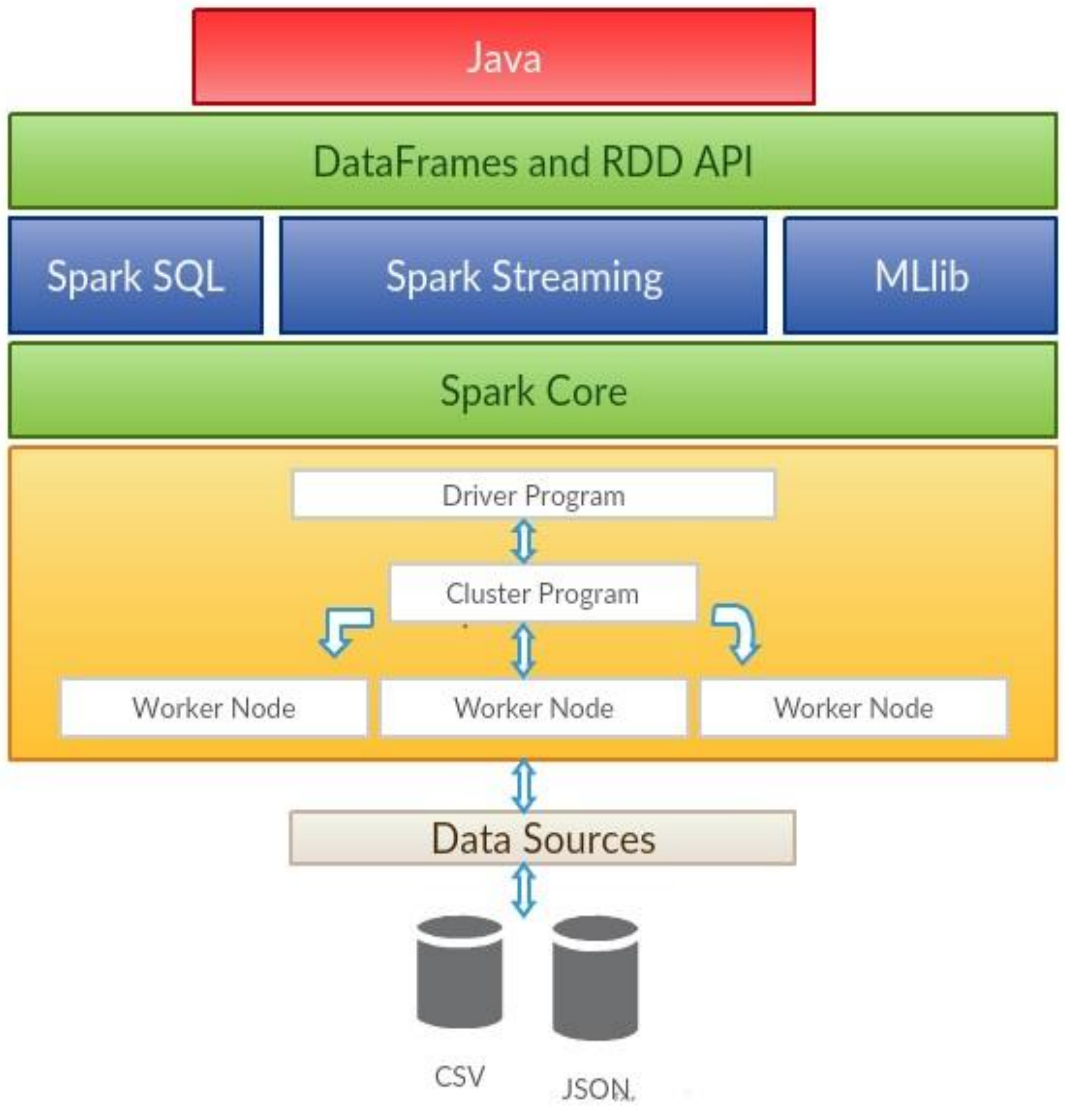
- Sports, Extreme Sports
- Travel, Trips, Adventure, Theme Parks, National Parks
- Watching, Listen, Eating activities
- Dancing, Yoga, Gym ,Relaxation activities
- Hobbies such as poetry, collection, DIY etc.

We particularly designed queries to analyze what all countries people involves in recreational activities and in turn revealing their work life balance. Also what all are the most popular activities, what king of age group they belong to, popular trending hashtags are part of this project.

GitHub URL: https://github.com/rashmitripathi/TwitterTrendAnalysis_BigData

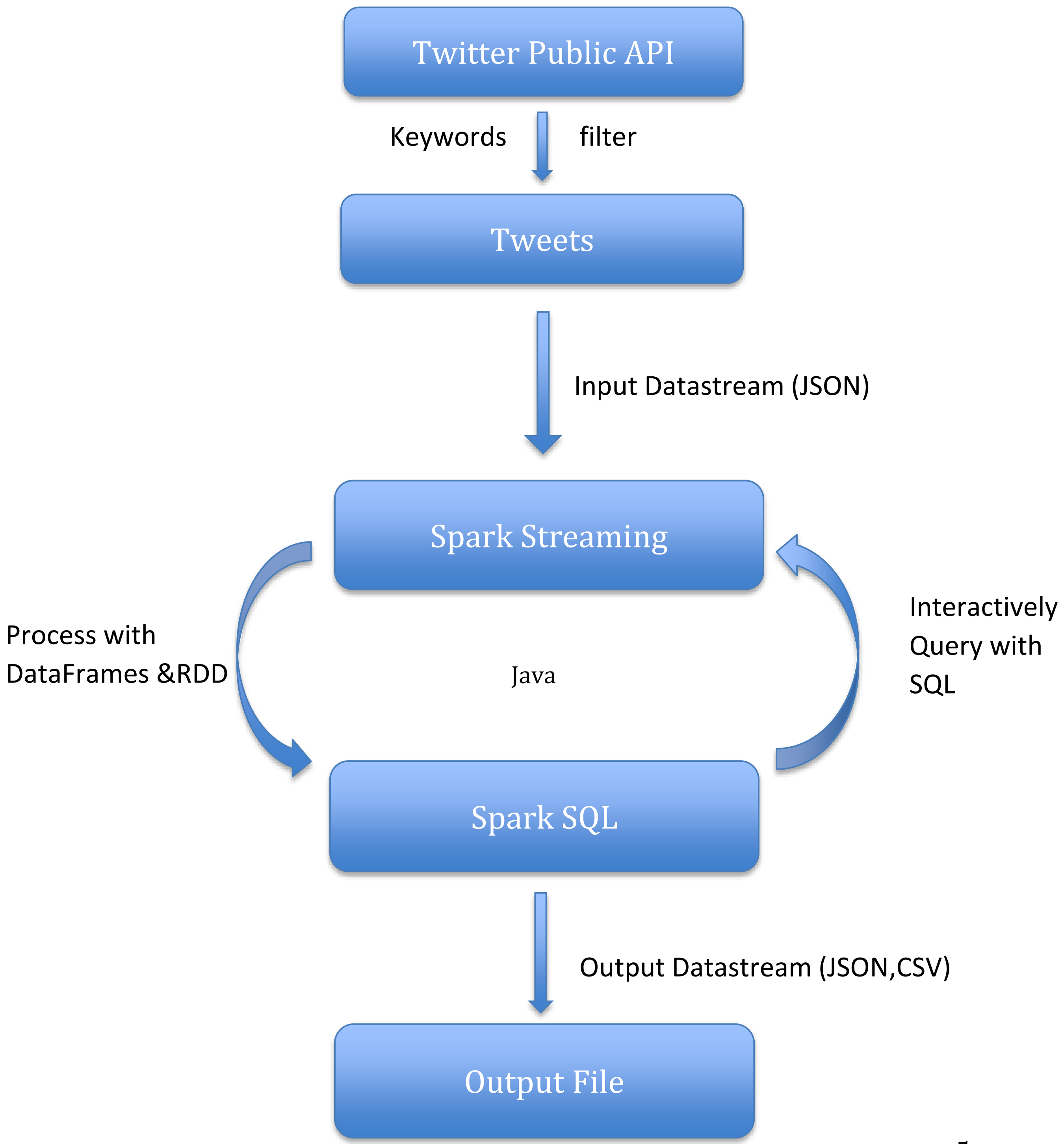
Architecture

Following is the software architecture for our project.



Process Flow

Following is the process flow for the software architecture:



System Setup

Operating System: Windows 10

IDE: Eclipse Neon 4.6.1

Browser: Google Chrome

Language: Java

Database: Apache Spark SQL

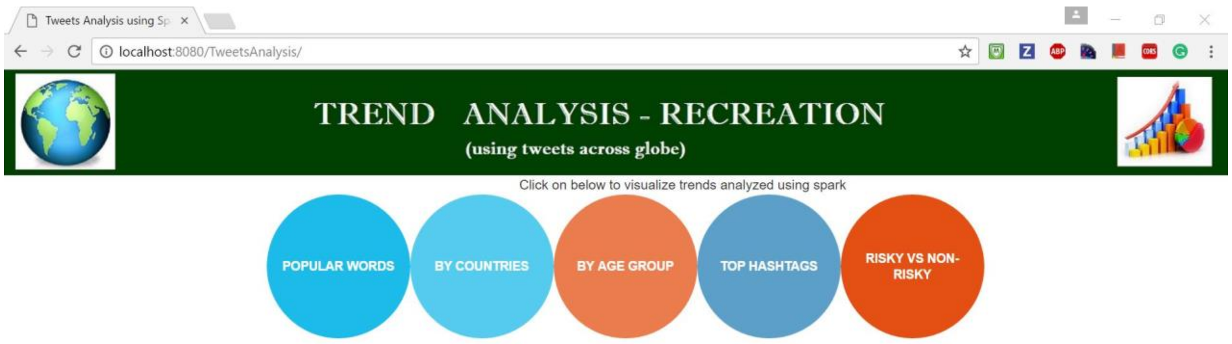
Library: Apache Spark Core, Spark Ml lib, Spark SQL, Twitter4j, Sun Jersey

Public API: uClassify

Visualization Charts: HighCharts.js, Google Geo Charts.js, d3.js

User Interface

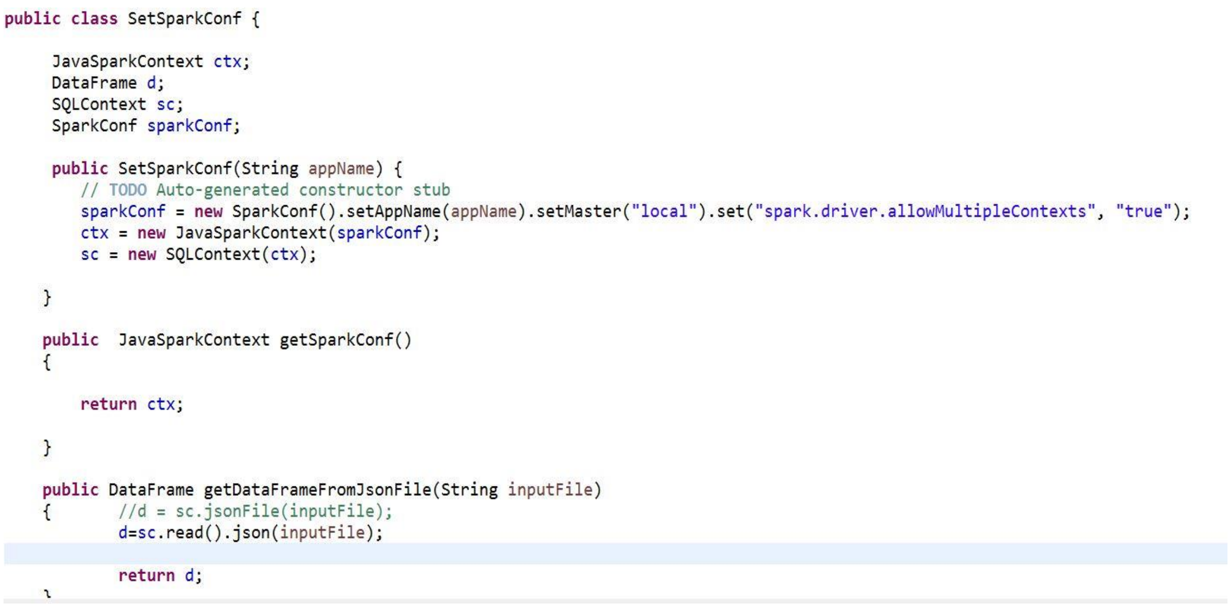
Snapshot:


Tweets Analysis using Sp
localhost:8080/TweetsAnalysis/
TREND ANALYSIS - RECREATION
(using tweets across globe)
Click on below to visualize trends analyzed using spark
POPULAR WORDS
BY COUNTRIES
BY AGE GROUP
TOP HASHTAGS
RISKY VS NON-RISKY

Queries

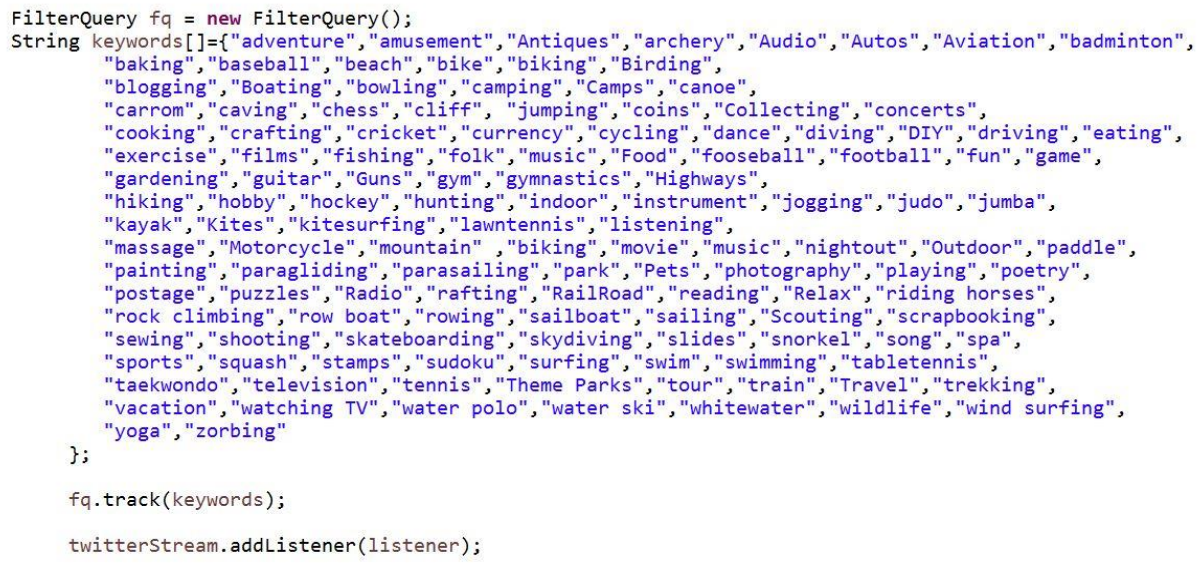
As part of this project, I have created one Java class to setup Spark Configuration and SQLContext. This class will be utilized in all further classes.

Source code for this is shown below:



```
public class SetSparkConf {

     JavaSparkContext ctx;
     DataFrame d;
     SQLContext sc;
     SparkConf sparkConf;

     public SetSparkConf(String appName) {
         // TODO Auto-generated constructor stub
         sparkConf = new SparkConf().setAppName(appName).setMaster("local").set("spark.driver.allowMultipleContexts", "true");
         ctx = new JavaSparkContext(sparkConf);
         sc = new SQLContext(ctx);

    }

    public  JavaSparkContext getSparkConf()
    {

         return ctx;

    }

    public DataFrame getDataFrameFromJsonFile(String inputFile)
    {        //d = sc.jsonFile(inputFile);
             d=sc.read().json(inputFile);

             return d;
```

Keywords to collect Tweets:



```
FilterQuery fq = new FilterQuery();
String keywords[]={"adventure","amusement","Antiques","archery","Audio","Autos","Aviation","badminton",
        "baking","baseball","beach","bike","biking","Birding",
        "blogging","Boating","bowling","camping","Camps","canoe",
        "carrom","caving","chess","cliff", "jumping","coins","Collecting","concerts",
        "cooking","crafting","cricket","currency","cycling","dance","diving","DIY","driving","eating",
        "exercise","films","fishing","folk","music","Food","fooseball","football","fun","game",
        "gardening","guitar","Guns","gym","gymnastics","Highways",
        "hiking","hobby","hockey","hunting","indoor","instrument","jogging","judo","jumba",
        "kayak","Kites","kitesurfing","lawntennis","listening",
        "massage","Motorcycle","mountain" ,"biking","movie","music","nightout","Outdoor","paddle",
        "painting","paragliding","parasailing","park","Pets","photography","playing","poetry",
        "postage","puzzles","Radio","rafting","RailRoad","reading","Relax","riding horses",
        "rock climbing","row boat","rowing","sailboat","sailing","Scouting","scrapbooking",
        "sewing","shooting","skateboarding","skydiving","slides","snorkel","song","spa",
        "sports","squash","stamps","sudoku","surfing","swim","swimming","tabletennis",
        "taekwondo","television","tennis","Theme Parks","tour","train","Travel","trekking",
        "vacation","watching TV","water polo","water ski","whitewater","wildlife","wind surfing",
        "yoga","zorbing"
    };

    fq.track(keywords);

    twitterStream.addListener(listener);
```

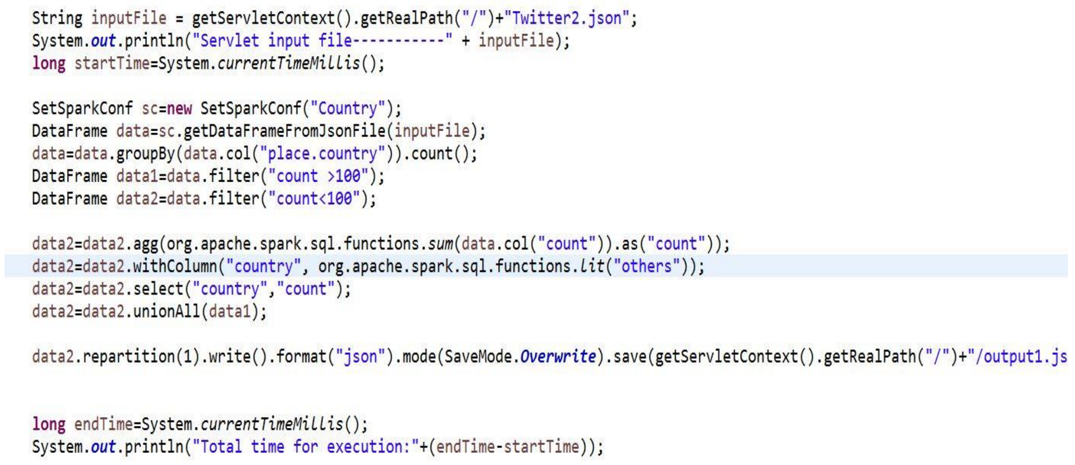
Query1: Dataframe to return top countries

Description: It returns the top countries where people talk about recreation. Numerous keywords related to sports, adventure, and activities were included to extract a large number of tweets.

Here the countries whose tweets are less than 100 will be considered as others. And the countries whose tweets are more than 100 will be respective countries and union operation is performed with the former dataframe.

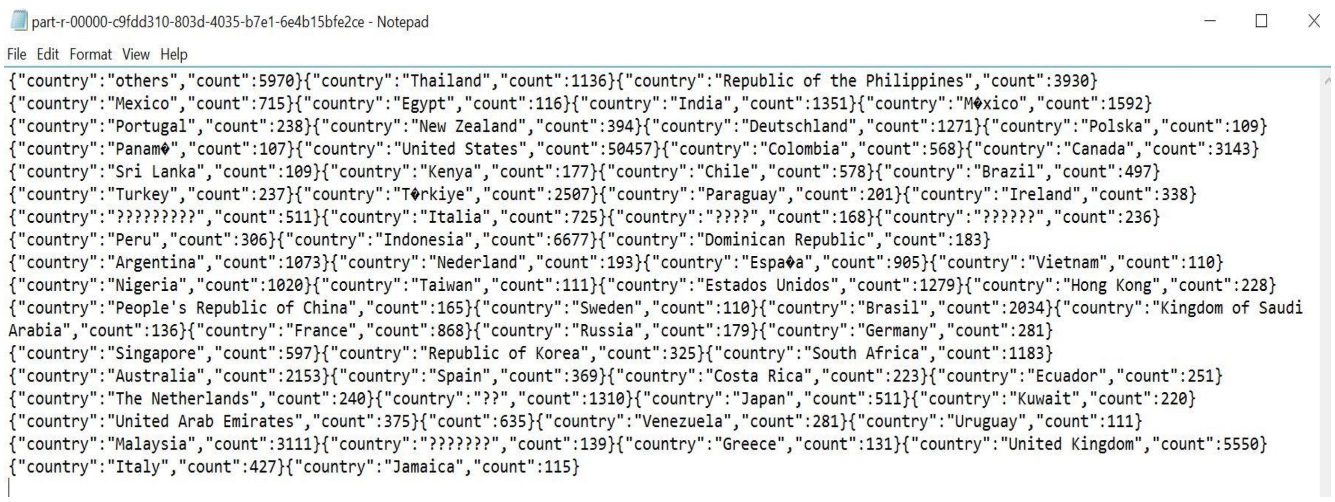
Special features: Dataframe , Union

Source code:



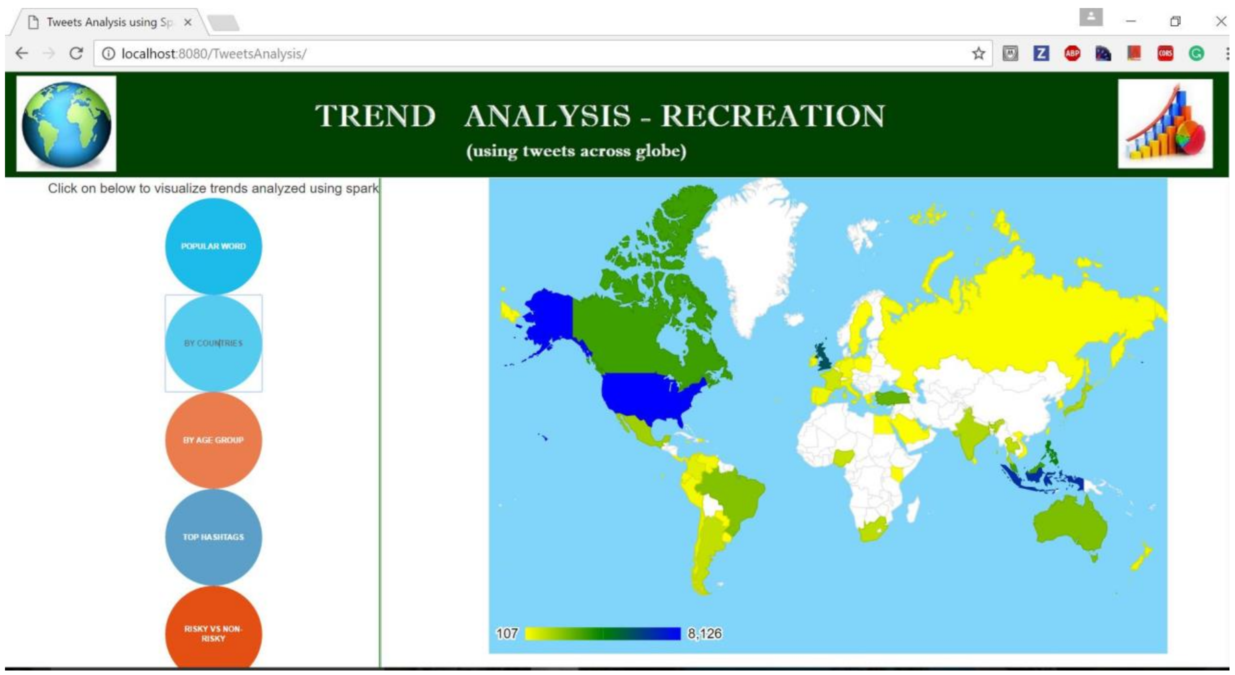
```
String inputFile = getServletContext().getRealPath("/")+"Twitter2.json";
System.out.println("Servlet input file-----------" + inputFile);
long startTime=System.currentTimeMillis();

SetSparkConf sc=new SetSparkConf("Country");
DataFrame data=sc.getDataFrameFromJsonFile(inputFile);
data=data.groupBy(data.col("place.country")).count();
DataFrame data1=data.filter("count >100");
DataFrame data2=data.filter("count<100");

data2=data2.agg(org.apache.spark.sql.functions.sum(data.col("count")).as("count"));
data2=data2.withColumn("country", org.apache.spark.sql.functions.lit("others"));
data2=data2.select("country","count");
data2=data2.unionAll(data1);

data2.repartition(1).write().format("json").mode(SaveMode.Overwrite).save(getServletContext().getRealPath("/")+"/output1.js


long endTime=System.currentTimeMillis();
System.out.println("Total time for execution:"+(endTime-startTime));
```

Output:


part-r-00000-c9fdd310-803d-4035-b7e1-6e4b15bfe2ce - Notepad
File Edit Format View Help

```
{"country":"others","count":5970}{"country":"Thailand","count":1136}{"country":"Republic of the Philippines","count":3930}
{"country":"Mexico","count":715}{"country":"Egypt","count":116}{"country":"India","count":1351}{"country":"M�xico","count":1592}
{"country":"Portugal","count":238}{"country":"New Zealand","count":394}{"country":"Deutschland","count":1271}{"country":"Polska","count":109}
{"country":"Panam�","count":107}{"country":"United States","count":50457}{"country":"Colombia","count":568}{"country":"Canada","count":3143}
{"country":"Sri Lanka","count":109}{"country":"Kenya","count":177}{"country":"Chile","count":578}{"country":"Brazil","count":497}
{"country":"Turkey","count":237}{"country":"T�rkiye","count":2507}{"country":"Paraguay","count":201}{"country":"Ireland","count":338}
{"country":"?????????","count":511}{"country":"Italia","count":725}{"country":"????","count":168}{"country":"??????","count":236}
{"country":"Peru","count":306}{"country":"Indonesia","count":6677}{"country":"Dominican Republic","count":183}
{"country":"Argentina","count":1073}{"country":"Nederland","count":193}{"country":"Espa�a","count":905}{"country":"Vietnam","count":110}
{"country":"Nigeria","count":1020}{"country":"Taiwan","count":111}{"country":"Estados Unidos","count":1279}{"country":"Hong Kong","count":228}
{"country":"People's Republic of China","count":165}{"country":"Sweden","count":110}{"country":"Brasil","count":2034}{"country":"Kingdom of Saudi
Arabia","count":136}{"country":"France","count":868}{"country":"Russia","count":179}{"country":"Germany","count":281}
{"country":"Singapore","count":597}{"country":"Republic of Korea","count":325}{"country":"South Africa","count":1183}
{"country":"Australia","count":2153}{"country":"Spain","count":369}{"country":"Costa Rica","count":223}{"country":"Ecuador","count":251}
{"country":"The Netherlands","count":240}{"country":"??","count":1310}{"country":"Japan","count":511}{"country":"Kuwait","count":220}
{"country":"United Arab Emirates","count":375}{"count":635}{"country":"Venezuela","count":281}{"country":"Uruguay","count":111}
{"country":"Malaysia","count":3111}{"country":"???????","count":139}{"country":"Greece","count":131}{"country":"United Kingdom","count":5550}
{"country":"Italy","count":427}{"country":"Jamaica","count":115}
```

Visualization:

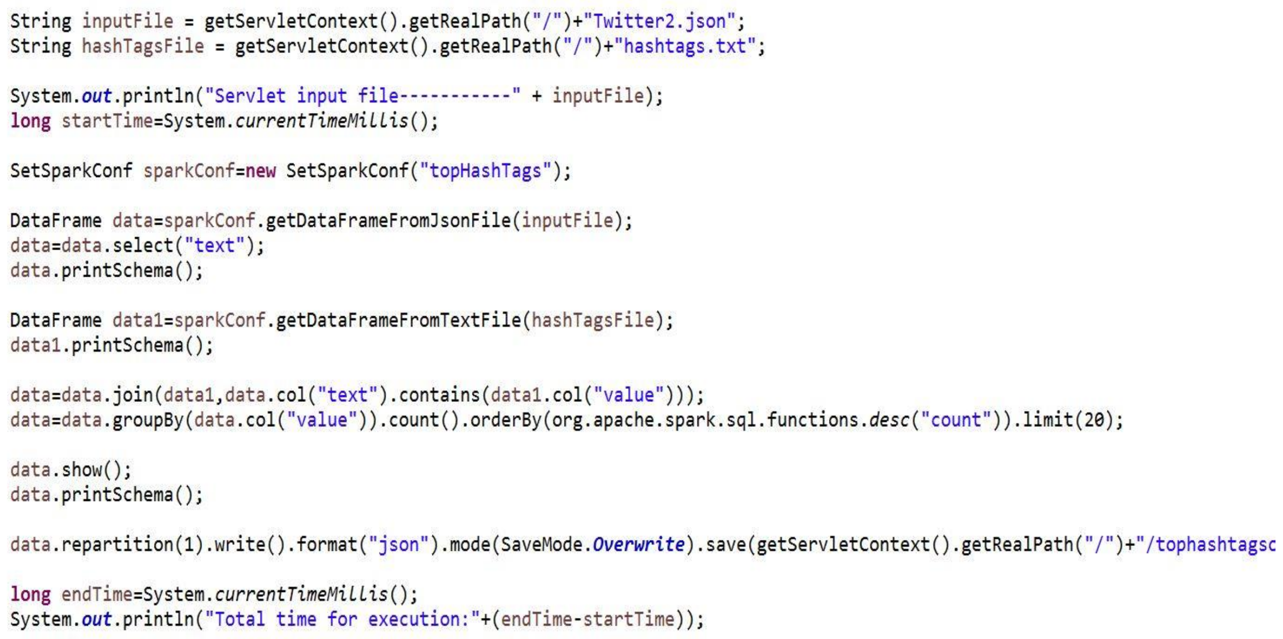

Tweets Analysis using Sp
localhost:8080/TweetsAnalysis/
TREND ANALYSIS - RECREATION
(using tweets across globe)
Click on below to visualize trends analyzed using spark
POPULAR WORD
BY COUNTRIES
BY AGE GROUP
TOP HASHTAGS
RISKY VS NON-RISKY
107
8,126

Query2: Dataframe to return top hashtags

Description : This query returns the top hashtags people used in their tweets. This query makes use of the hashtag file provided. It is basically join of hashtags from hashtag file plus tweets.

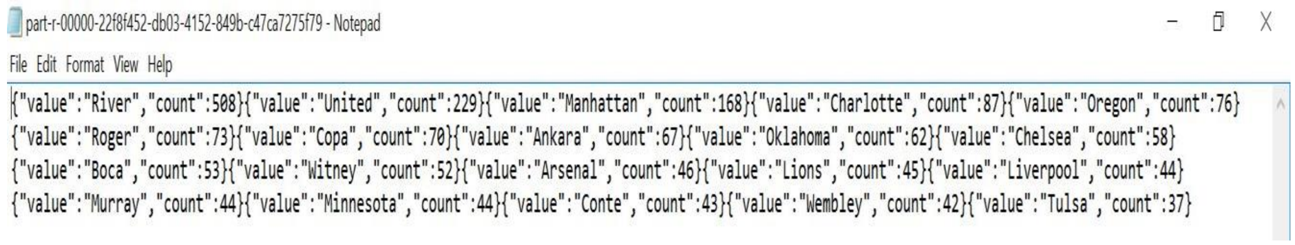
Special features: Dataframe , Join with another hashtags text file ,groupBy,orderBy

Source code:



```
String inputFile = getServletContext().getRealPath("/")+"Twitter2.json";
String hashTagsFile = getServletContext().getRealPath("/")+"hashtags.txt";

System.out.println("Servlet input file-----------" + inputFile);
long startTime=System.currentTimeMillis();

SetSparkConf sparkConf=new SetSparkConf("topHashTags");

DataFrame data=sparkConf.getDataFrameFromJsonFile(inputFile);
data=data.select("text");
data.printSchema();

DataFrame data1=sparkConf.getDataFrameFromTextFile(hashTagsFile);
data1.printSchema();

data=data.join(data1,data.col("text").contains(data1.col("value")));
data=data.groupBy(data.col("value")).count().orderBy(org.apache.spark.sql.functions.desc("count")).limit(20);

data.show();
data.printSchema();

data.repartition(1).write().format("json").mode(SaveMode.Overwrite).save(getServletContext().getRealPath("/")+"/tophashtagsc

long endTime=System.currentTimeMillis();
System.out.println("Total time for execution:"+(endTime-startTime));
```

Output:

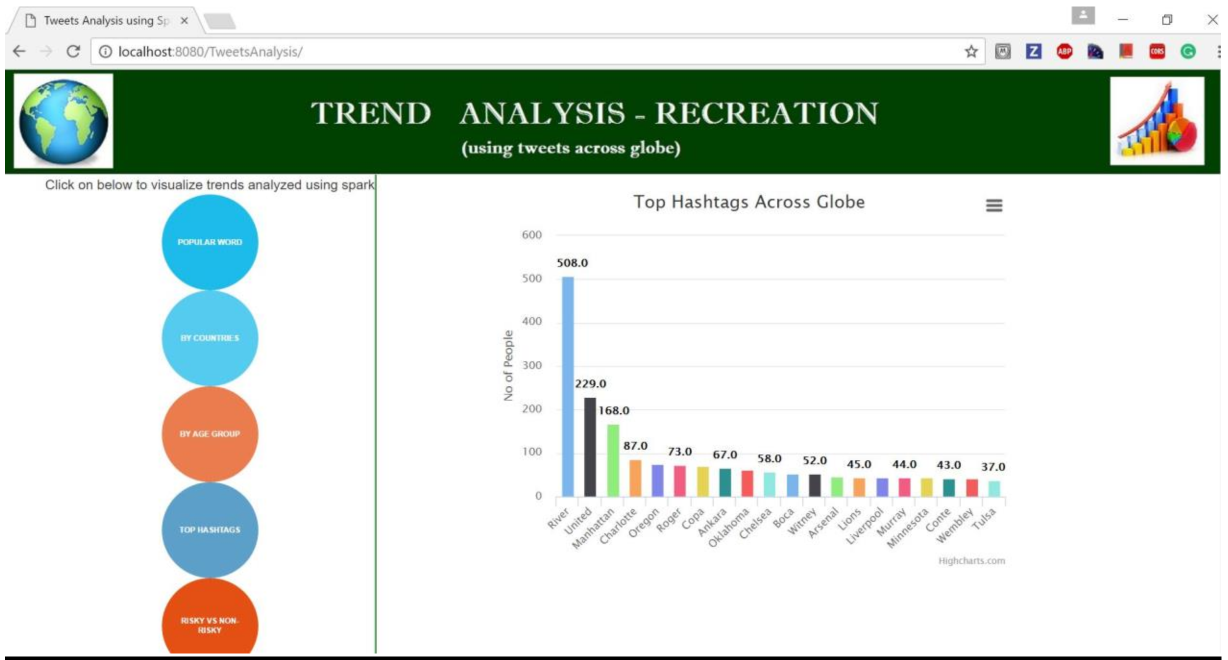


part-r-00000-22f8f452-db03-4152-849b-c47ca7275f79 - Notepad

File Edit Format View Help

```
{"value":"River","count":508}{"value":"United","count":229}{"value":"Manhattan","count":168}{"value":"Charlotte","count":87}{"value":"Oregon","count":76}
{"value":"Roger","count":73}{"value":"Copa","count":70}{"value":"Ankara","count":67}{"value":"Oklahoma","count":62}{"value":"Chelsea","count":58}
{"value":"Boca","count":53}{"value":"Witney","count":52}{"value":"Arsenal","count":46}{"value":"Lions","count":45}{"value":"Liverpool","count":44}
{"value":"Murray","count":44}{"value":"Minnesota","count":44}{"value":"Conte","count":43}{"value":"Wembley","count":42}{"value":"Tulsa","count":37}
```

Visualization:

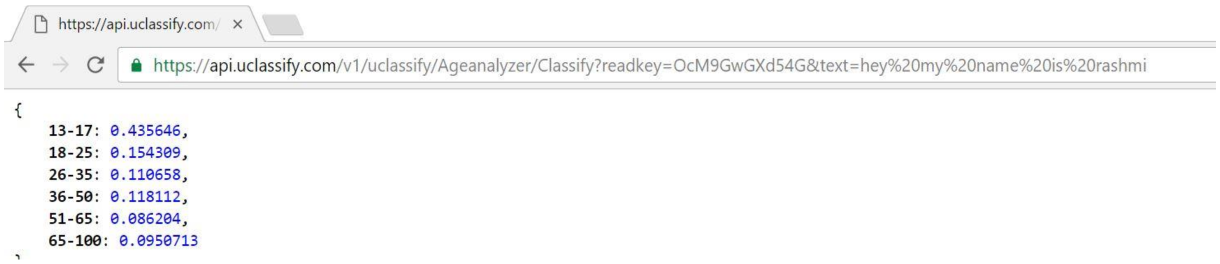


Query 3: Dataframe to call Public API to classify tweets based on age group

Description :
For query 3, a public API “UClassify” is used in which tweet text is fed to API and it returns the age group to which the user belongs who tweeted that particular text. Also this query is dynamically collecting tweets. As the uClassify calls have limitation, we have performed this query with 100 tweets.

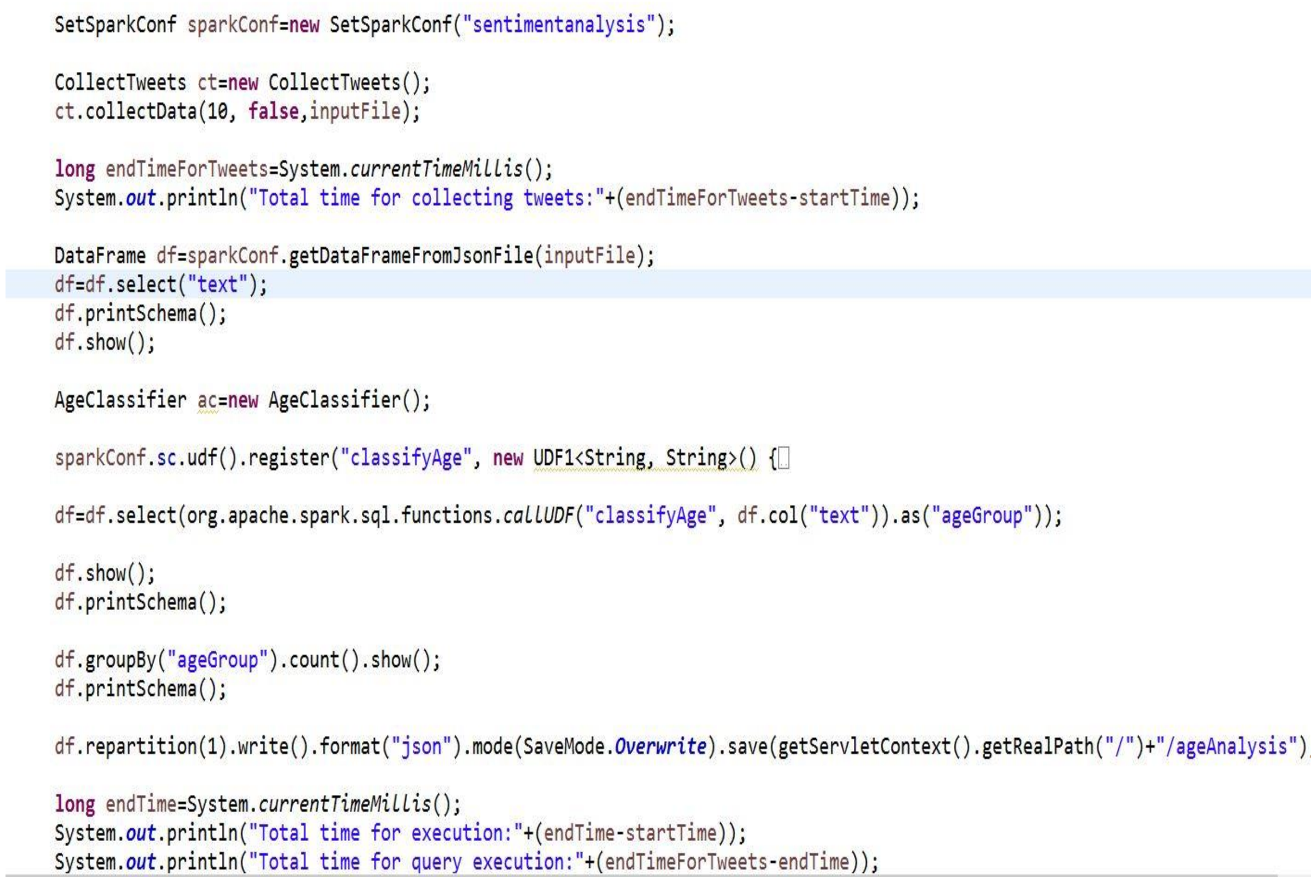
Special features: UDF, calling Public API

UClassify JSON Response:

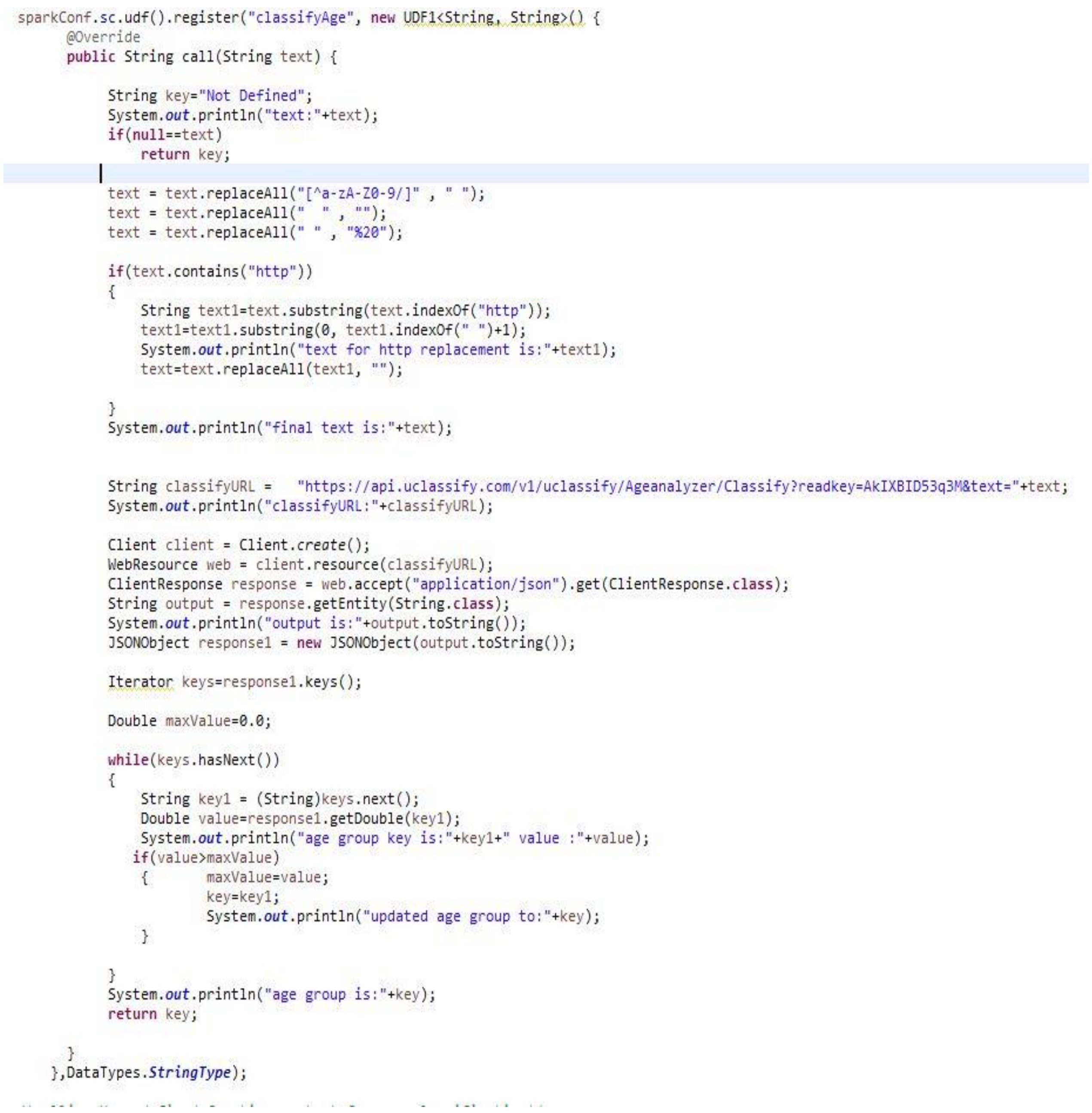

https://api.uclassify.com/v1/uclassify/Ageanalyzer/Classify?readkey=OcM9GwGXd54G&text=hey%20my%20name%20is%20rashmi

```
{
    13-17: 0.435646,
    18-25: 0.154309,
    26-35: 0.110658,
    36-50: 0.118112,
    51-65: 0.086204,
    65-100: 0.0950713
}
```

Source code: UDF source code:

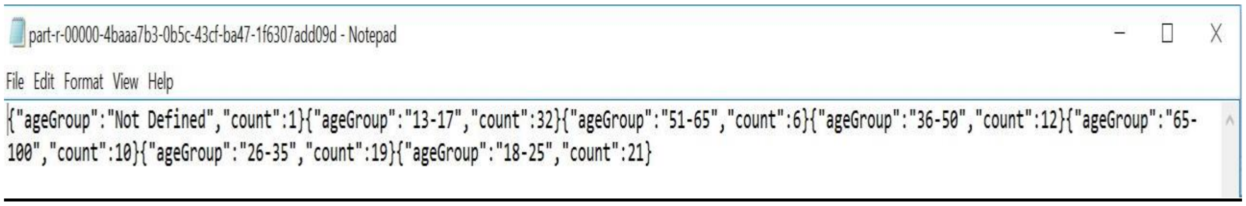


```
SetSparkConf sparkConf=new SetSparkConf("sentimentanalysis");

CollectTweets ct=new CollectTweets();
ct.collectData(10, false,inputFile);

long endTimeForTweets=System.currentTimeMillis();
System.out.println("Total time for collecting tweets:"+(endTimeForTweets-startTime));

DataFrame df=sparkConf.getDataFrameFromJsonFile(inputFile);
df=df.select("text");
df.printSchema();
df.show();

AgeClassifier ac=new AgeClassifier();

sparkConf.sc.udf().register("classifyAge", new UDF1<String, String>() {

df=df.select(org.apache.spark.sql.functions.callUDF("classifyAge", df.col("text")).as("ageGroup"));

df.show();
df.printSchema();

df.groupBy("ageGroup").count().show();
df.printSchema();

df.repartition(1).write().format("json").mode(SaveMode.Overwrite).save(getServletContext().getRealPath("/")+"/ageAnalysis")

long endTime=System.currentTimeMillis();
System.out.println("Total time for execution:"+(endTime-startTime));
System.out.println("Total time for query execution:"+(endTimeForTweets-endTime));
```



```
sparkConf.sc.udf().register("classifyAge", new UDF1<String, String>() {
        @Override
        public String call(String text) {

            String key="Not Defined";
            System.out.println("text:"+text);
            if(null==text)
                return key;

            text = text.replaceAll("[^a-zA-Z0-9/]" , " ");
            text = text.replaceAll("  " , "");
            text = text.replaceAll(" " , "%20");

            if(text.contains("http"))
            {
                String text1=text.substring(text.indexOf("http"));
                text1=text1.substring(0, text1.indexOf(" ")+1);
                System.out.println("text for http replacement is:"+text1);
                text=text.replaceAll(text1, "");

            }
            System.out.println("final text is:"+text);


            String classifyURL =   "https://api.uclassify.com/v1/uclassify/Ageanalyzer/Classify?readkey=AkIXBID53q3M&text="+text;
            System.out.println("classifyURL:"+classifyURL);

            Client client = Client.create();
            WebResource web = client.resource(classifyURL);
            ClientResponse response = web.accept("application/json").get(ClientResponse.class);
            String output = response.getEntity(String.class);
            System.out.println("output is:"+output.toString());
            JSONObject response1 = new JSONObject(output.toString());

            Iterator keys=response1.keys();

            Double maxValue=0.0;

            while(keys.hasNext())
            {
                String key1 = (String)keys.next();
                Double value=response1.getDouble(key1);
                System.out.println("age group key is:"+key1+" value :"+value);
               if(value>maxValue)
                {       maxValue=value;
                        key=key1;
                        System.out.println("updated age group to:"+key);
                }

            }
            System.out.println("age group is:"+key);
            return key;

        }
    },DataTypes.StringType);
```

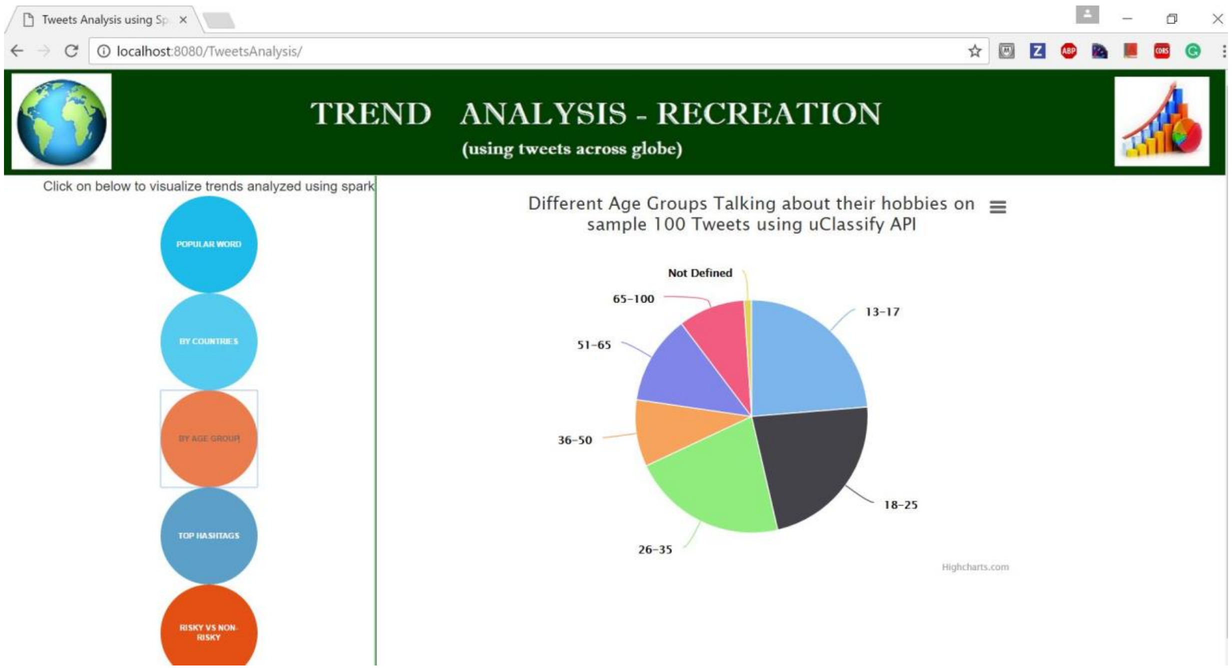
Output:



part-r-00000-4baaa7b3-0b5c-43cf-ba47-1f6307add09d - Notepad
File Edit Format View Help

```
{"ageGroup":"Not Defined","count":1}{"ageGroup":"13-17","count":32}{"ageGroup":"51-65","count":6}{"ageGroup":"36-50","count":12}{"ageGroup":"65-
100","count":10}{"ageGroup":"26-35","count":19}{"ageGroup":"18-25","count":21}
```

Visualization:

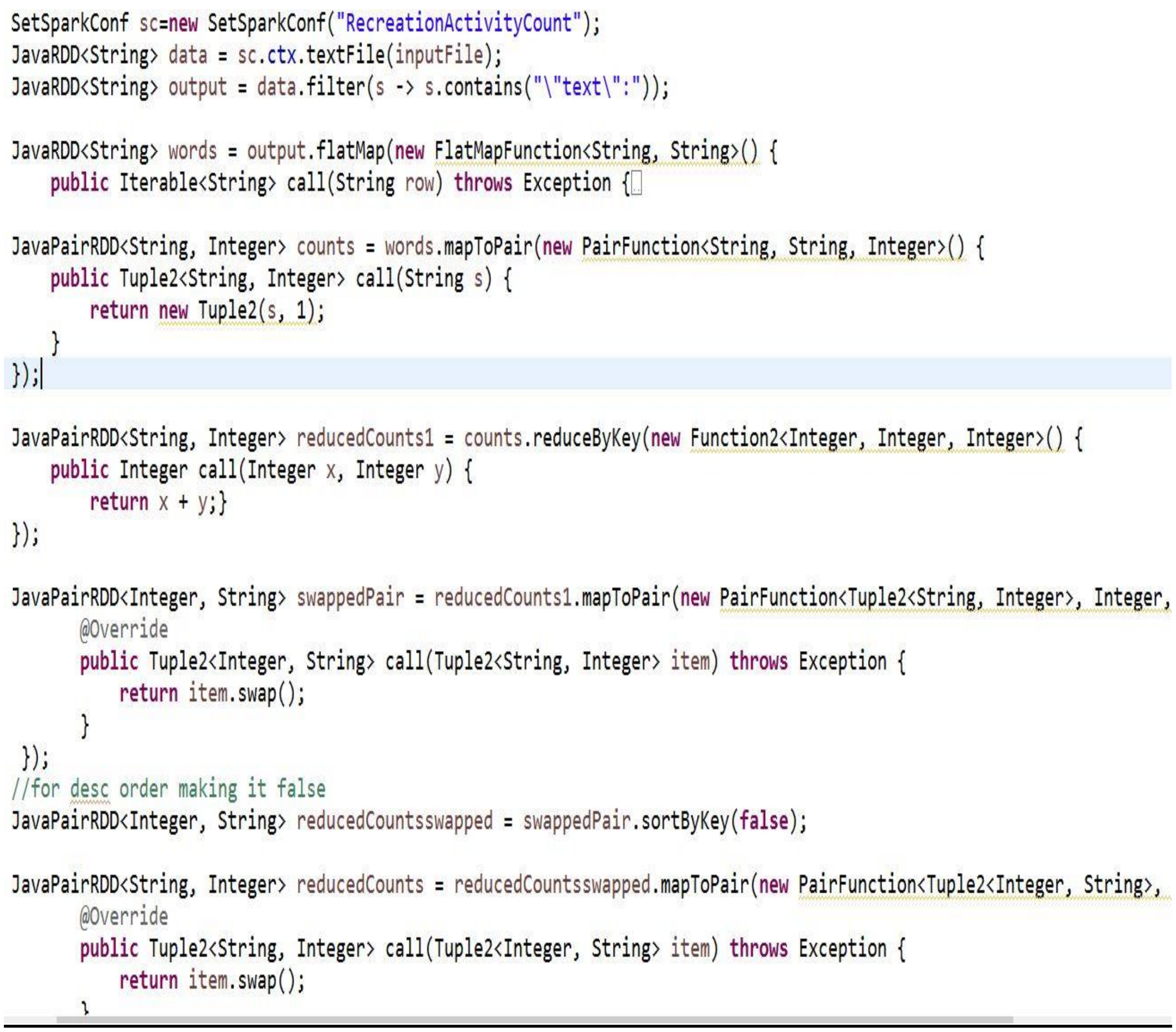

Tweets Analysis using Sp
localhost:8080/TweetsAnalysis/
TREND ANALYSIS - RECREATION
(using tweets across globe)
Click on below to visualize trends analyzed using spark
POPULAR WORD
BY COUNTRIES
BY AGE GROUP
TOP HASHTAGS
RISKY VS NON-RISKY
Different Age Groups Talking about their hobbies on sample 100 Tweets using uClassify API
Not Defined
65-100
51-65
36-50
26-35
18-25
13-17
Highcharts.com

Query 4: JAVA RDD query to get Top activities

Description : This query will perform the map and reduce on tweets . It will get the top activities among all countries.

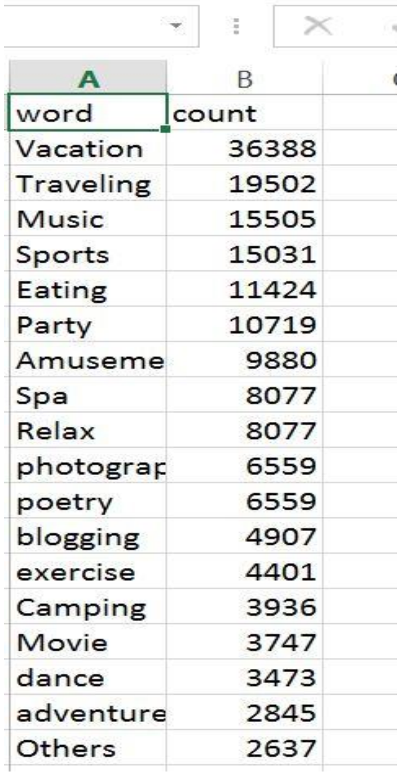
Special features: Java RDD, mapToPair, reduceByKey, flatMap, sortBy, swap

Source code:



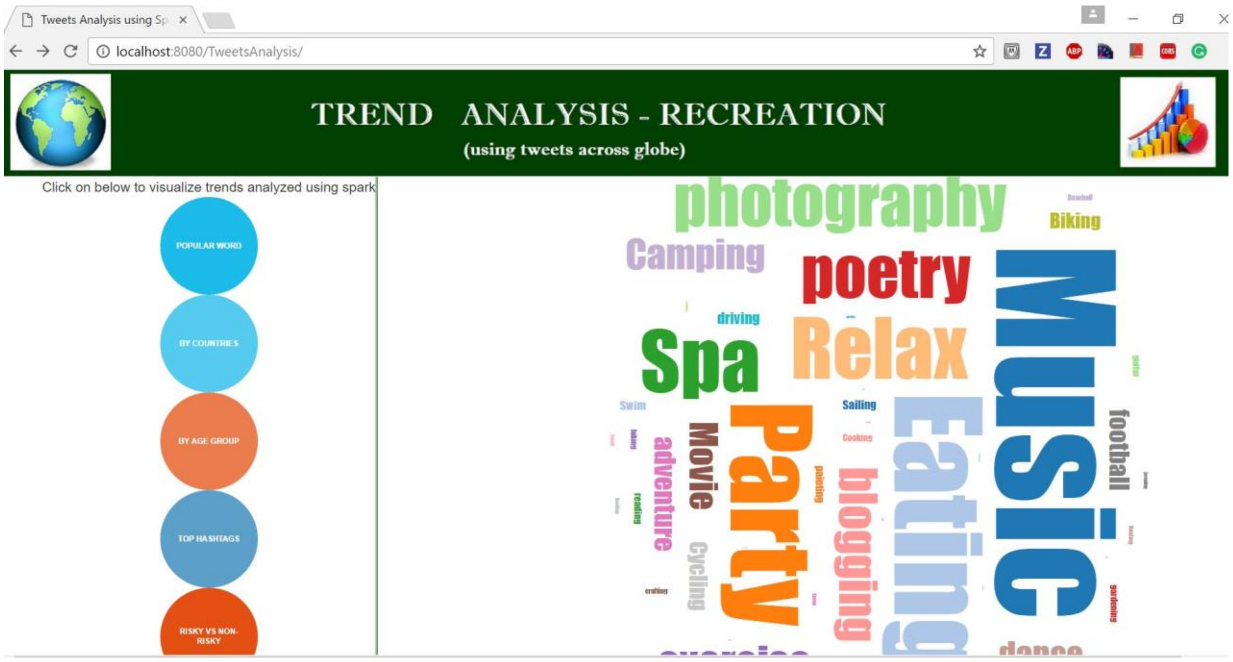
```
SetSparkConf sc=new SetSparkConf("RecreationActivityCount");
JavaRDD<String> data = sc.ctx.textFile(inputFile);
JavaRDD<String> output = data.filter(s -> s.contains("\"text\":"));

JavaRDD<String> words = output.flatMap(new FlatMapFunction<String, String>() {
    public Iterable<String> call(String row) throws Exception {

JavaPairRDD<String, Integer> counts = words.mapToPair(new PairFunction<String, String, Integer>() {
    public Tuple2<String, Integer> call(String s) {
        return new Tuple2(s, 1);
    }
});

JavaPairRDD<String, Integer> reducedCounts1 = counts.reduceByKey(new Function2<Integer, Integer, Integer>() {
    public Integer call(Integer x, Integer y) {
        return x + y;}
});

JavaPairRDD<Integer, String> swappedPair = reducedCounts1.mapToPair(new PairFunction<Tuple2<String, Integer>, Integer,
        @Override
        public Tuple2<Integer, String> call(Tuple2<String, Integer> item) throws Exception {
            return item.swap();
        }
 });
//for desc order making it false
JavaPairRDD<Integer, String> reducedCountsswapped = swappedPair.sortByKey(false);

JavaPairRDD<String, Integer> reducedCounts = reducedCountsswapped.mapToPair(new PairFunction<Tuple2<Integer, String>,
        @Override
        public Tuple2<String, Integer> call(Tuple2<Integer, String> item) throws Exception {
            return item.swap();
```

Output:



word	count
Vacation	36388
Traveling	19502
Music	15505
Sports	15031
Eating	11424
Party	10719
Amuseme	9880
Spa	8077
Relax	8077
photograp	6559
poetry	6559
blogging	4907
exercise	4401
Camping	3936
Movie	3747
dance	3473
adventure	2845
Others	2637

Visualization:

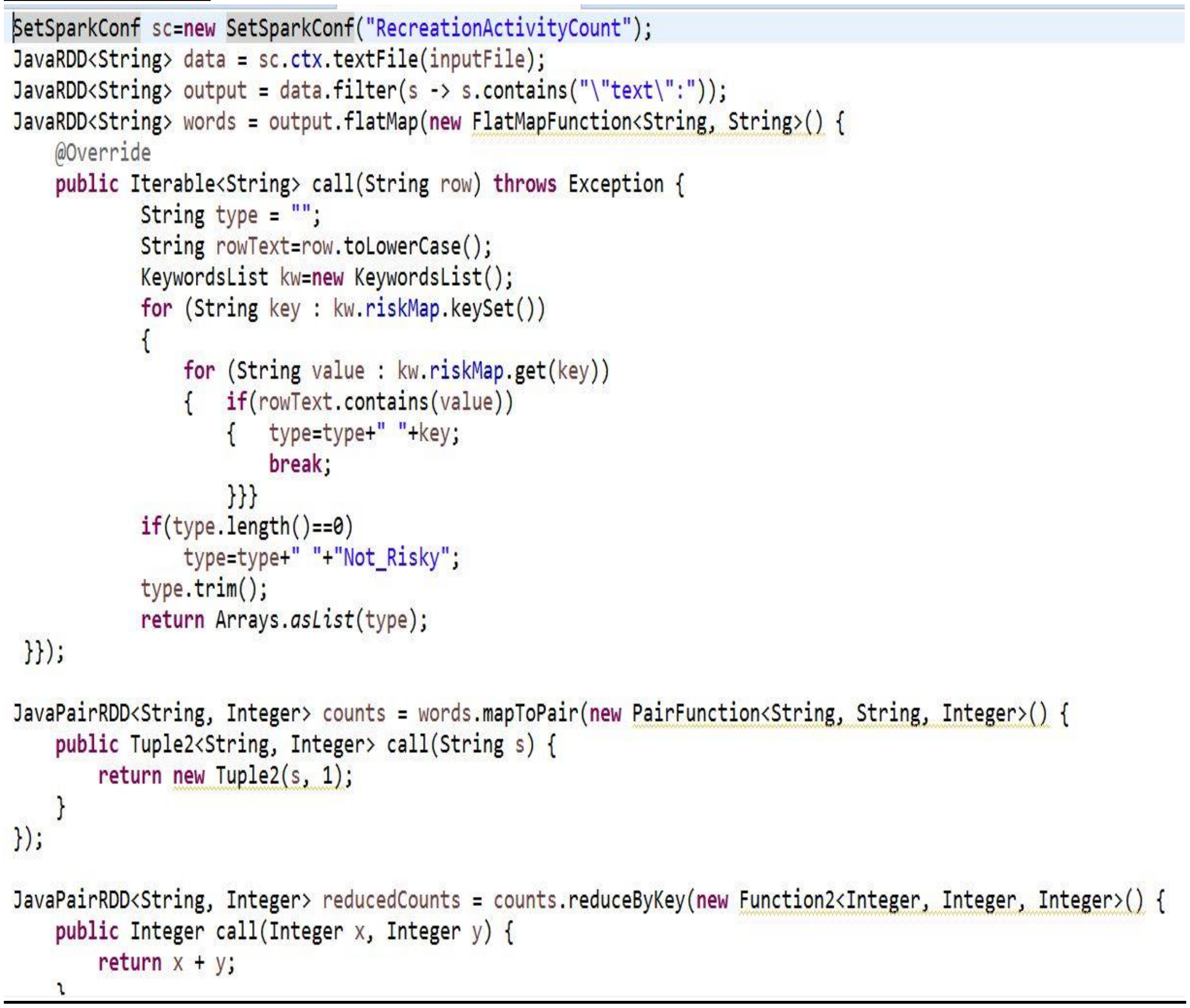


Query 5: JAVA RDD query to classify Risky vs Non Risky Activities

Description: This query will perform the map and reduce on tweets . It will classify the tweets based on people following risky activities like sky diving, jumping, kayak etc.

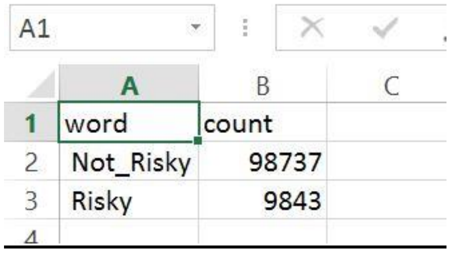
Special features: Java RDD, mapToPair, reduceByKey, flatMap

Source code:



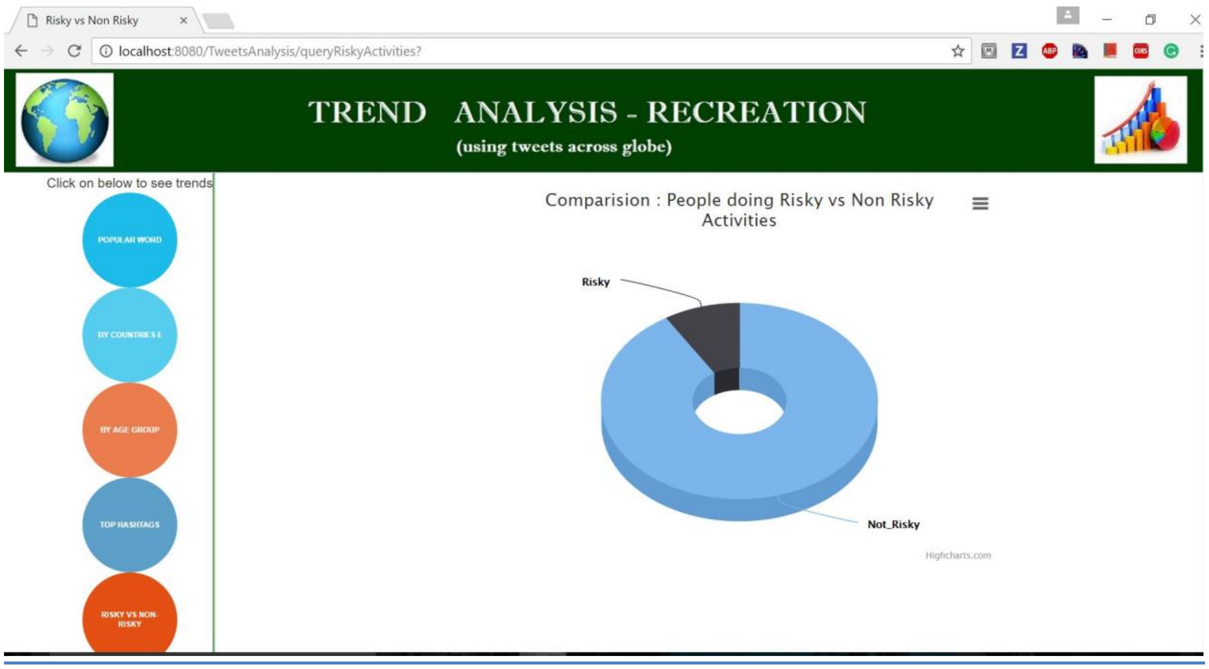
```
SetSparkConf sc=new SetSparkConf("RecreationActivityCount");
JavaRDD<String> data = sc.ctx.textFile(inputFile);
JavaRDD<String> output = data.filter(s -> s.contains("\"text\":"));
JavaRDD<String> words = output.flatMap(new FlatMapFunction<String, String>() {
    @Override
    public Iterable<String> call(String row) throws Exception {
            String type = "";
            String rowText=row.toLowerCase();
            KeywordsList kw=new KeywordsList();
            for (String key : kw.riskMap.keySet())
            {
                for (String value : kw.riskMap.get(key))
                {   if(rowText.contains(value))
                    {   type=type+" "+key;
                        break;
                    }}}
            if(type.length()==0)
                type=type+" "+"Not_Risky";
            type.trim();
            return Arrays.asList(type);
 }});

JavaPairRDD<String, Integer> counts = words.mapToPair(new PairFunction<String, String, Integer>() {
    public Tuple2<String, Integer> call(String s) {
        return new Tuple2(s, 1);
    }
});

JavaPairRDD<String, Integer> reducedCounts = counts.reduceByKey(new Function2<Integer, Integer, Integer>() {
    public Integer call(Integer x, Integer y) {
        return x + y;
```

Output:



	A	B	C
1	word	count	
2	Not_Risky	98737	
3	Risky	9843	
4			

Visualization:


Risky vs Non Risky
localhost:8080/TweetsAnalysis/queryRiskyActivities?
TREND ANALYSIS - RECREATION
(using tweets across globe)
Click on below to see trends
POPULAR WORD
BY COUNTRIES
BY AGE GROUP
TOP HASHTAGS
RISKY VS NON-RISKY
Comparision : People doing Risky vs Non Risky Activities
Risky
Not_Risky
Highcharts.com

Runtime Measurements

Below is total run time to execute above queries:

Q_No	Description	Type	Features	Running time (in ms)	(seconds)
1	By Countries	Dataframe	Union	37722	38
2	Top hashtags	Dataframe	join with hashtags file	285737	286
3	Age classification of tweets user	Dataframe	UDF with public API+ automatically collecting tweets	62811	63
4	Top activities	RDD	flatMap+ mapReduce+sortBy	42955	43
5	Risky vs non Risky	RDD	flatMap+ mapReduce	18667	19

Extra Credits

- Used google Map in query 1 to visualize. Please refer the code for the same in “show.diffrnt.graphs.js” file.
- For query 5 we are dynamically executed one of the queries upon the user’s click using form submit and Java servlet execution.
- W.r.t Phase 2 we have collected 100 tweets at runtime in query 3 and then executed out query

Github URL-Code Repository

https://github.com/rashmitripathi/TwitterTrendAnalysis_BigData

References

- https://spark.apache.org/docs/1.6.1/sql-programming-guide.html
- http://twitter4j.org/en/code-examples.html
- https://www.uclassify.com/docs
- http://stackoverflow.com/questions/35211993/how-to-join-mutiple-columns-in-spark-sql-using-java-for-filtering-in-dataframe
- http://stackoverflow.com/questions/18471043/twitter-stream-api-to-get-users-tweets
- http://getbootstrap.com/css/#grid-offsetting
- http://www.knowstack.com/webtech/charts_demo/highchartsdemo4.html
- https://developers.google.com/chart/interactive/docs/gallery/geochart