COMP-SCI 5540 Principles of Big Data Management

University of Missouri-Kansas City

Department of Computer Science and Electrical Engineering

Project Report

GitHub URL: https://github.com/bharathkumarna/Principles-of-BigData

Team – 6

Abhiram Reddy Nalla

Bharath Kumar Natesan Arumugam

Sai Kumar Ponnamaneni

Sibi Chakravarthy Ramesh

Design Steps:

1. Collect social media data (tweets) using any theme as filter and store it as JSON files.
2. A Spark Context is created to establish connection to Spark Cluster.
3. SQL Context class is created which represents an entry point into all functionality in Spark SQL.
4. Data Frames are created based on content of JSON file and register it to tables.
5. Run SQL queries programmatically using SQL function on registered tables.
6. Store the returned results in mongoDB using mongoDB-spark connector.
7. Using mLab Data API the results are fetched and visualized using Google Charts.

Libraries:

Spark Core contains the basic functionality of Spark and Spark SQL is Spark's package for working with Structured data.

1. org.apache.spark:spark-core_2.11:2.0.02
2. org.apache.spark:spark-sql_2.11:2.0.02

Signpost has been designed to work in conjunction with Apache HTTPComponents library for signing HTTP messages on the Scala platform in conformance with the OAuth Core 1.0 standard.

3. oauth.signpost:signpost-commonshttp4:1.2.1.22
4. org.apache.directory.studio:org.apache.httpcomponents.httpclient:4.02
5. signpost-core-1.2.1.22
6. org.apache.directory.studio:org.apache.httpcomponents.httpcore:4.02

Tweepy – An easy-to-use Python library for accessing the Twitter API.

7. tweepy-3.5.0

MongoDB Connector for spark – provides integration between MongoDB and Apache Spark

8. org.mongodb:mongodb-driver:3.4.02
9. mongo-spark-connector_2.10-2.0.02

APIs:

1. Twitter public REST APIs - GET followers/ids

Resource URL: *https://api.twitter.com/1.1/followers/ids.json*

Returns a collection of user IDs for every user following the specified user.

2. mLab Data APIs - GET /databases/{database}/collections/{collection}

Resource URL: https://api.mlab.com/api/1/databases/my-db/collections/my-coll?apiKey=myAPIKey

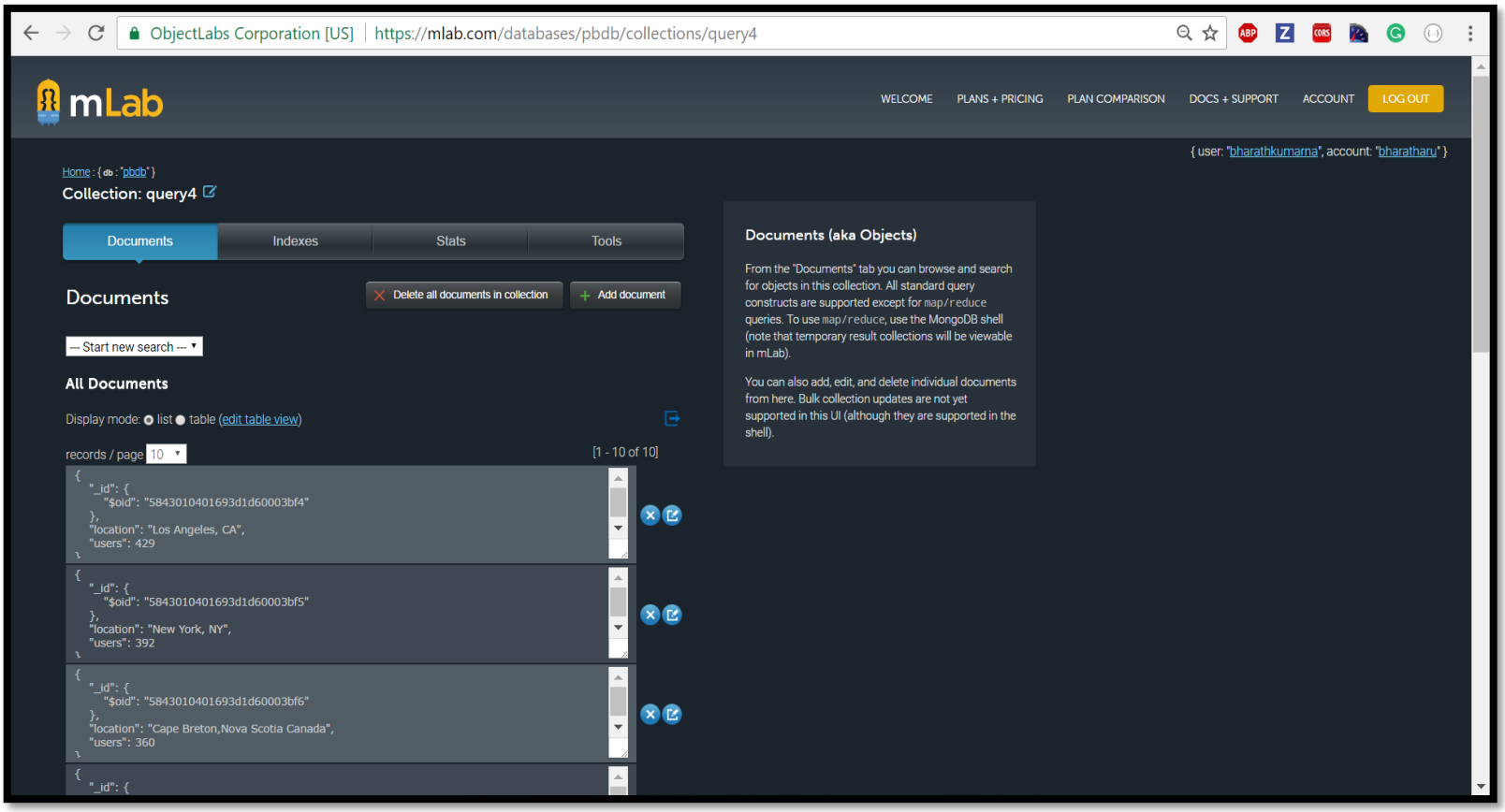
Return documents in the specified collection.

Programming/Web Languages:

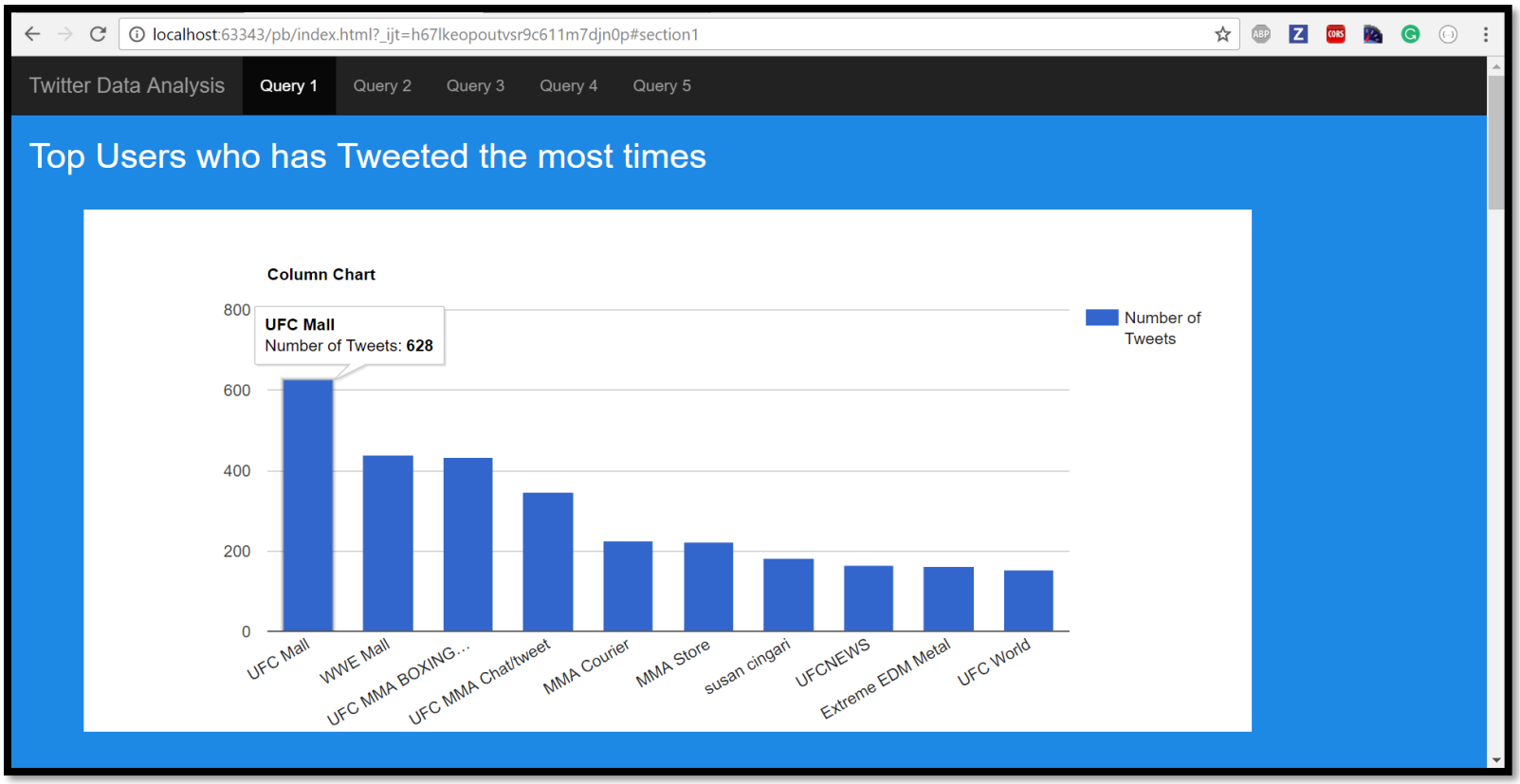
1. Scala – to run Spark Programs.
2. Python – to run Tweets collection program.
3. HTML5, CSS3 – to design user interface and front-end development.
4. JavaScript – to do API calls and visualize using Google Charts

Output

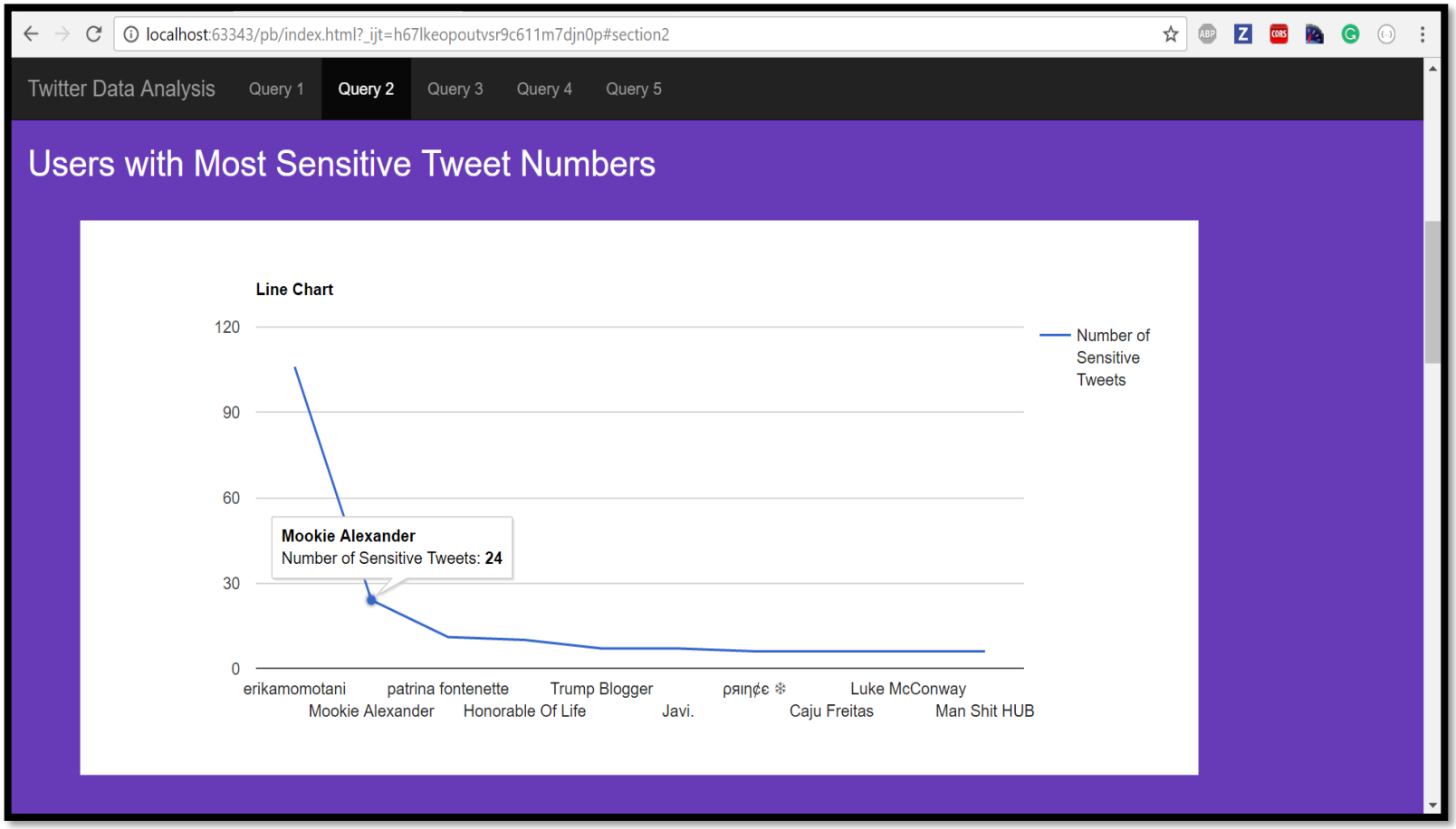
MongoDB:



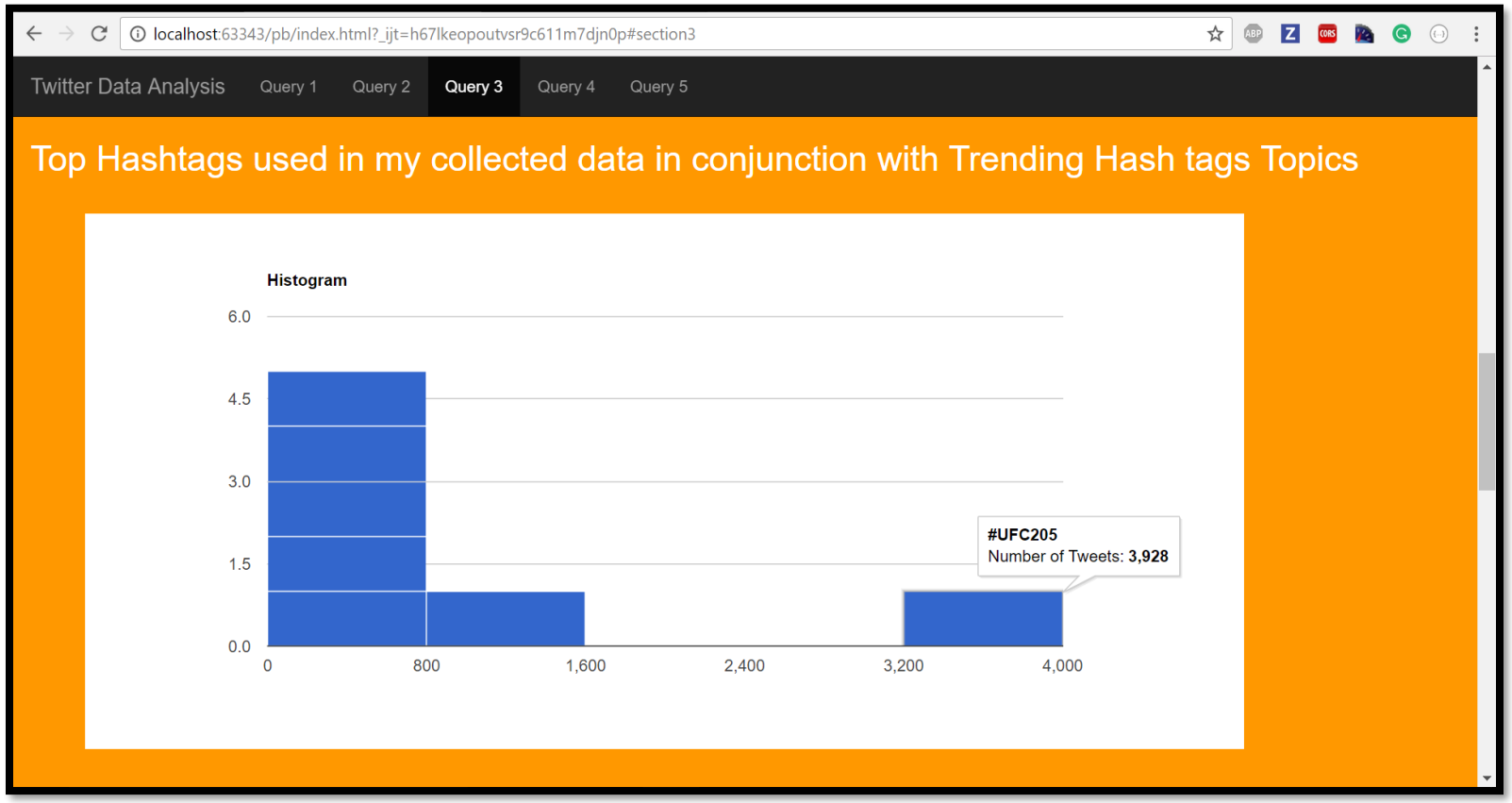
Query1:



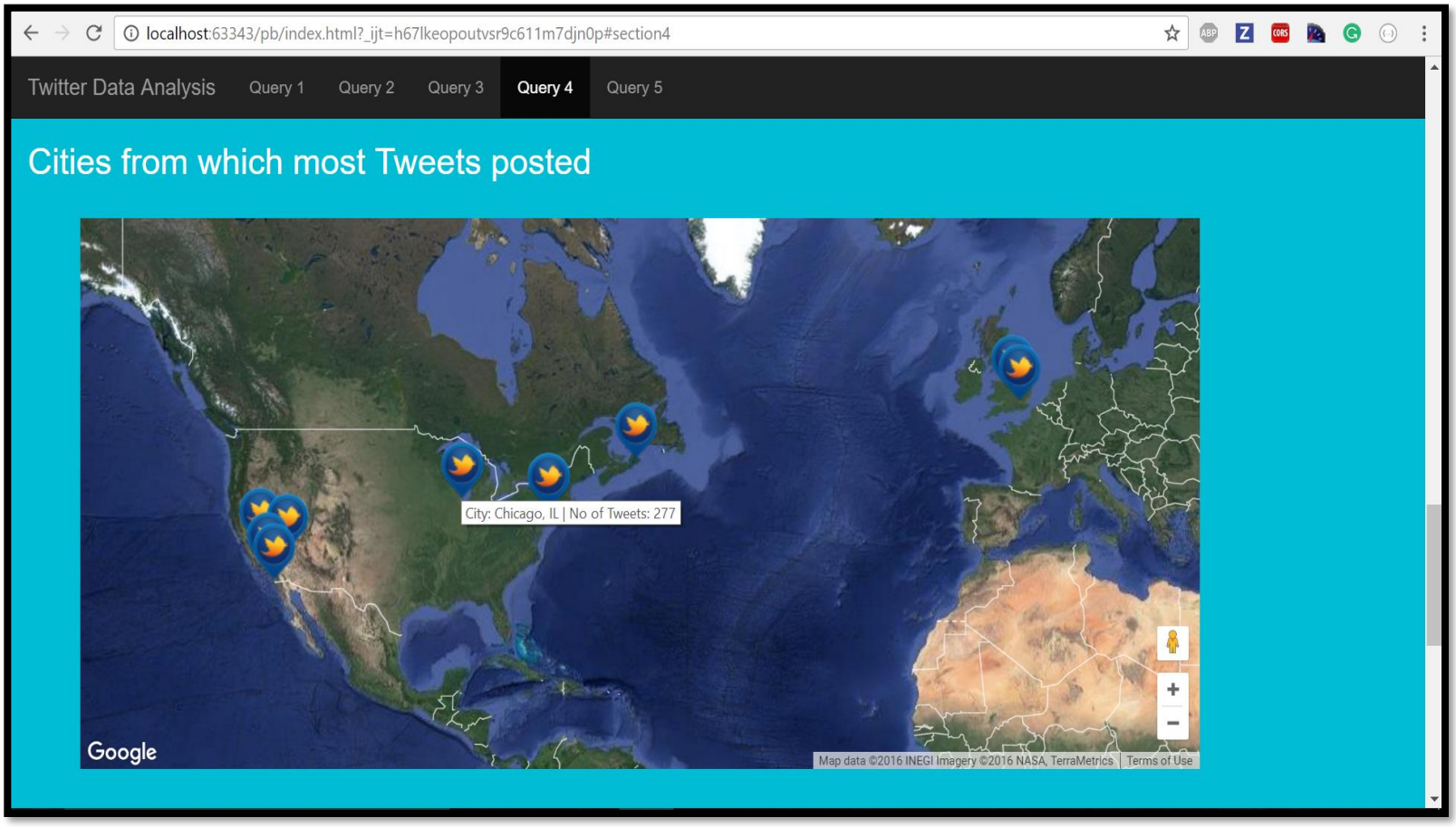
Query2:



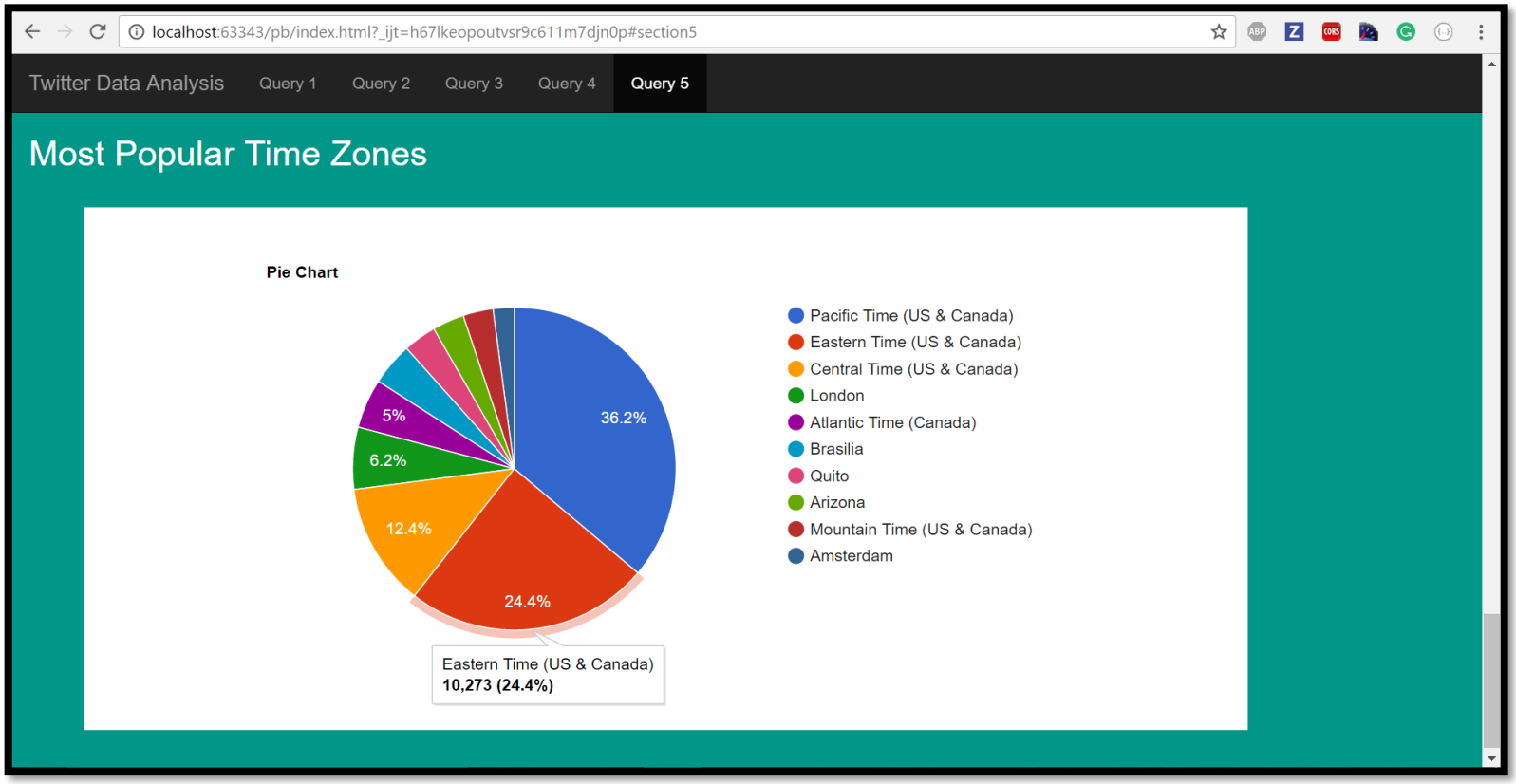
Query3:



Query4:



Query5:



Code:

Collecting Tweets:

```
from tweepy.streaming import
StreamListener from tweepy import
OAuthHandler from tweepy import Stream

#Twitter Authentication
access_token = "1048610250-QQZ8D05FWBIon130QSgjg0XGDN0dw3lXXhP7KFt"
access_token_secret = "RRiMG6c7mIY61apEJWSwoxMMaSVN8tQwIcuK627ugp46r"
consumer_key = "RRAnQIWfiuDBpJm94OWgwmpEF"
consumer_secret = "uXj3hPKmkU931K8ye5FMZemBUky4UyEQxQCz2Ej5qyS4zp0Ddw"
class
StdOutListener(StreamListener):
     def on_data(self,
data):
        print(data)         with
open('fetched_tweet.json','a') as tf:
tf.write(data)
return True
if __name__ ==
'__main__':     l =
StdOutListener()
    auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)     stream
= Stream(auth, l)

#Filter Tweets according to theme
stream.filter(track=['UFC','WWE'])
```

Spark SQL Program:

```
import oauth.signpost.commonshttp.CommonsHttpOAuthConsumer
import org.apache.commons.io.IOUtils
import org.apache.http.client.methods.HttpGet
import org.apache.http.impl.client.DefaultHttpClient
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.sql.SQLContext

object sample {
  //Twitter Authentication
  val AccessToken = "1048610250-
QQZ8D05FWBIon130QSgjg0XGDN0dw3lXXhP7KFt";
  val AccessSecret = "RRiMG6c7mIY61apEJWSwoxMMaSVN8tQwIcuK627ugp46r";
  val ConsumerKey = "RRAnQIWfiuDBpJm94OWgwmpEF";
  val ConsumerSecret =
"uXj3hPKmkU931K8ye5FMZemBUky4UyEQxQCz2Ej5qyS4zp0Ddw";

  def main(args: Array[String]) {

    System.setProperty("hadoop.home.dir","C:\\hadoop-
2.3.0\\bin\\tweet")
    val conf = new
SparkConf().setAppName("CountSpark").setMaster("local[2]").set("spark.s
ql.warehouse.dir","file:///c:/tmp/spark-
warehouse").set("spark.mongodb.output.uri","mongodb://user:user@ds11960
8.mlab.com:19608/pbdb?replicaSet=rs-ds119608")
    val sc = new SparkContext(conf)
    val sqlcontext = new SQLContext(sc)
    import sqlcontext.implicits._

    val tweetsfile =
sqlcontext.read.json("C:\\Users\\bn4n5\\workspace\\Pb-
ass\\mypackage\\fetched_tweet.json")
    tweetsfile.registerTempTable("querytable1")
    //val Query1,Query2,Query3,Query4,Query5
    var a='Y'
    while (a=='Y') {
```

```
    //Menu Option
    println("****** Analytical Queries using Apache Spark ******")
    println("1=>Top Users who has Tweeted the most times")
    println("2=>Users with Most Sensitive Tweet Numbers")
    println("3=>Top Hashtags used in my collected data in conjunction
with Trending Hash tags Topics")
    println("4=>Cities from which most Tweets posted")
    println("5=>Most Popular Time Zones")
    println("Enter your choice:")

    val choice=readInt()
      choice match {

        case 1 =>
          val Query1 = sqlcontext.sql("select
user.name,user.screen_name, count(user.followers_count) as tweetsCount
from querytable1 group by user.screen_name,user.name order by
tweetsCount desc limit 10")
          Query1.show()
          //Query 1 calling public API
          val name = readLine("Enter screen name to find user IDs for
every user following the specified user:")
          val consumer = new CommonsHttpOAuthConsumer(ConsumerKey,
ConsumerSecret)
          consumer.setTokenWithSecret(AccessToken, AccessSecret)
          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(IOUtils.toString(response.getEntity().getContent()))
          println("Press Y to continue or N to exit:")
          a = readChar()
          //Query Result copied to MongoDB
          Query1.write.option("collection",
"query1").mode("overwrite").format("com.mongodb.spark.sql").save()

        case 2 =>
          val Query2 = sqlcontext.sql("select
user.name,count(user.name) as no_of_sensitive_tweets from querytable1
where possibly_sensitive=true and user.lang='en' group by user.name
order by no_of_sensitive_tweets desc limit 10")
```

```
          Query2.show()
          println("Press Y to continue or N to exit:")
          a = readChar()
          //Query Result copied to MongoDB
          Query2.write.option("collection",
"query2").mode("overwrite").format("com.mongodb.spark.sql").save()

        case 3 =>
          //Query 3 uses data in the PopularHahtagsAndTopics.txt file
posted on Blackboard in conjunction with my collected data
          val text = sc.textFile("C:\\Users\\bn4n5\\workspace\\Pb-
ass\\mypackage\\PopularHahtagsAndTopics.txt").map(_.split("/n")).map(fr
t => Text(frt(0))).toDF()
          text.registerTempTable("querytable")
          val Query=sqlcontext.sql("select querytable.name from
querytable where querytable.name like '%#UFC%' or querytable.name like
'%#WWE%' or querytable.name like '%#MMA%' ")
          Query.registerTempTable("querytable3")
          val Query3 = sqlcontext.sql("select
querytable3.name,count(querytable1.text) as count from querytable1 join
querytable3 on querytable1.text like concat ('%',querytable3.name,'%')
group by querytable3.name order by count desc limit 10 ")
          Query3.show();
          println("Press Y to continue or N to exit:")
          a = readChar()
          //Query Result copied to MongoDB
          Query3.write.option("collection",
"query3").mode("overwrite").format("com.mongodb.spark.sql").save()

        case 4 =>
          val Query4=sqlcontext.sql("select user.location,count(*) as
users from querytable1 where user.location like '%,%' and user.location
not like '%1%' group by user.location order by users desc limit 10")
          Query4.show()
          println("Press Y to continue or N to exit:")
          a = readChar()
          //Query Result copied to MongoDB
          Query4.write.option("collection",
"query4").mode("overwrite").format("com.mongodb.spark.sql").save()

        case 5 =>
          val Query5=sqlcontext.sql("select user.time_zone,count(*) as
users from querytable1 where user.time_zone <> 'null' group by
```

```
user.time_zone order by users desc limit 10")
          Query5.show()
          println("Press Y to continue or N to exit:")
          a = readChar()
          //Query Result copied to MongoDB
          Query5.write.option("collection",
"query5").mode("overwrite").format("com.mongodb.spark.sql").save()

      }

    }
  }
}
case class Text(name: String)
```

Home Page: index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Twitter Data Analysis</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min
.css">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js">
</script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.j
s"></script>
    <script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAWhSE1HAi753_M8r
cFTfcbBXUQInf8y6c&v=3.exp&sensor=true"></script>
    <script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript" src="js/columnchart.js"></script>
    <script type="text/javascript" src="js/piechart.js"></script>
    <script type="text/javascript" src="js/linechart.js"></script>
    <script type="text/javascript" src="js/histogram.js"></script>
    <script type="text/javascript" src="js/map.js"></script>
```

```
    <style>
        body {
            position: relative;
        }
        #section1 {padding-top:50px;height:620px;color: #fff;
background-color: #1E88E5;}
        #section2 {padding-top:50px;height:620px;color: #fff;
background-color: #673ab7;}
        #section3 {padding-top:50px;height:620px;color: #fff;
background-color: #ff9800;}
        #section4 {padding-top:50px;height:620px;color: #fff;
background-color: #00bcd4;}
        #section5 {padding-top:50px;height:620px;color: #fff;
background-color: #009688;}
    </style>
</head>
<body data-spy="scroll" data-target=".navbar" data-offset="50">

<nav class="navbar navbar-inverse navbar-fixed-top">
    <div class="container-fluid">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-
toggle="collapse" data-target="#myNavbar">
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="#">Twitter Data Analysis</a>
        </div>
        <div>
            <div class="collapse navbar-collapse" id="myNavbar">
                <ul class="nav navbar-nav">
                    <li><a href="#section1">Query 1</a></li>
                    <li><a href="#section2">Query 2</a></li>
                    <li><a href="#section3">Query 3</a></li>
                    <li><a href="#section4">Query 4</a></li>
                    <li><a href="#section5">Query 5</a></li>
                </ul>
            </div>
        </div>
    </div>
</nav>
<div id="section1" class="container-fluid">
```

```
    <h2>Top Users who has Tweeted the most times</h2><br>
    <div class="container"> <form id="bar_chart"></form></div>
</div>
<div id="section2" class="container-fluid">
    <h2>Users with Most Sensitive Tweet Numbers</h2><br>
    <div class="container"> <form id="line_chart"></form></div>
</div>
<div id="section3" class="container-fluid">
    <h2>Top Hashtags used in my collected data in conjunction with
Trending Hash tags Topics</h2><br>
    <div class="container"> <form id="histogram_chart"></form></div>
</div>
<div id="section4" class="container-fluid">
    <h2>Cities from which most Tweets posted</h2><br>
    <div class="container"> <form id="map_chart"style="width: 1000px;
height: 450px;"> </form></div>
</div>
<div id="section5" class="container-fluid">
    <h2>Most Popular Time Zones</h2><br>
    <div class="container"> <form id="pie_chart"></form></div>
</div>
</body>
</html>
```

Column Chart.js

```
google.charts.load('current', {'packages':['corechart']});
google.charts.setOnLoadCallback(drawChart);

function drawChart() {
    //API call
    var data_file =
"https://api.mlab.com/api/1/databases/pbdb/collections/query1?apiKey=qu
Ja8qCv_KGUvY5S3Qnf9EDnWzoDvSQA";
    var http_request = new XMLHttpRequest();

    http_request.onreadystatechange = function(){

        if (http_request.readyState == 4  ){
            var jsonObj = JSON.parse(http_request.responseText);
            var data = new google.visualization.DataTable();
            data.addColumn('string', 'name');
            data.addColumn('number', 'Number of Tweets');
```

```
            for(var i=0;i<jsonObj.length;i++)
            {
                data.addRows([
                    [jsonObj[i].name, jsonObj[i].tweetsCount]
                ]);
            }
            // Set chart options
            var options = {
                'title':"Column Chart",
                'width':1000,
                'height':450};

            var chart = new
google.visualization.ColumnChart(document.getElementById('bar_chart'));
            chart.draw(data, options);
        }
    }

    http_request.open("GET", data_file, true);
    http_request.send();
}
```

Piechart.js

```
google.charts.load('current', {'packages':['corechart']});
google.charts.setOnLoadCallback(drawChart);

function drawChart() {
    //API call
    var data_file =
"https://api.mlab.com/api/1/databases/pbdb/collections/query5?apiKey=qu
Ja8qCv_KGUvY5S3Qnf9EDnWzoDvSQA";
    var http_request = new XMLHttpRequest();
    http_request.onreadystatechange = function(){

        if (http_request.readyState == 4  ){
            var jsonObj = JSON.parse(http_request.responseText);
            var data = new google.visualization.DataTable();
            data.addColumn('string', 'Time Zone');
            data.addColumn('number', 'Number of Tweets');
            for(var i=0;i<jsonObj.length;i++)
            {
                data.addRows([
                    [jsonObj[i].time_zone, jsonObj[i].users]
```

```
                ]);
            }
            // Set chart options
            var options = {
                'title':"Pie Chart",
                'width':1000,
                'height':450};

            var chart = new
google.visualization.PieChart(document.getElementById('pie_chart'));
            chart.draw(data, options);
        }
    }

    http_request.open("GET", data_file, true);
    http_request.send();
}
```

Histogram.js

```
google.charts.load('current', {'packages':['corechart']});
google.charts.setOnLoadCallback(drawChart);

function drawChart() {
    //API call
    var data_file =
"https://api.mlab.com/api/1/databases/pbdb/collections/query3?apiKey=qu
Ja8qCv_KGUvY5S3Qnf9EDnWzoDvSQA";
    var http_request = new XMLHttpRequest();
    http_request.onreadystatechange = function(){

        if (http_request.readyState == 4  ){
            var jsonObj = JSON.parse(http_request.responseText);
            var data = new google.visualization.DataTable();
            data.addColumn('string', 'Hashtags');
            data.addColumn('number', 'Number of Tweets');
            for(var i=0;i<jsonObj.length;i++)
            {
                data.addRows([
                    [jsonObj[i].name, jsonObj[i].count]
                ]);
            }
            // Set chart options
```

```
            var options = {
                'title':"Histogram",
                'width':1000, 'height':450, legend: { position: 'none'
}};

            var chart = new
google.visualization.Histogram(document.getElementById('histogram_chart
'));
            chart.draw(data, options);
        }
    }
    http_request.open("GET", data_file, true);
    http_request.send();
}
```

Linechart.js

```
google.charts.load('current', {'packages':['corechart']});
google.charts.setOnLoadCallback(drawChart);

function drawChart() {
    //API call
    var data_file =
"https://api.mlab.com/api/1/databases/pbdb/collections/query2?apiKey=qu
Ja8qCv_KGUvY5S3Qnf9EDnWzoDvSQA";
    var http_request = new XMLHttpRequest();

    http_request.onreadystatechange = function(){

        if (http_request.readyState == 4  ){
            var jsonObj = JSON.parse(http_request.responseText);
            var data = new google.visualization.DataTable();
            data.addColumn('string', 'name');
            data.addColumn('number', 'Number of Sensitive Tweets');
            for(var i=0;i<jsonObj.length;i++)
            {
                data.addRows([
                    [jsonObj[i].name,
jsonObj[i].no_of_sensitive_tweets]
                ]);
            }
            // Set chart options
            var options = {
                'title':"Line Chart",
```

```
                'width':1000,
                'height':450};

            var chart = new
google.visualization.LineChart(document.getElementById('line_chart'));
            chart.draw(data, options);
        }
    }

    http_request.open("GET", data_file, true);
    http_request.send();
}
```

Map.js

```
google.charts.load('upcoming', { 'packages': ['map'] });
google.charts.setOnLoadCallback(drawMap);

function drawMap() {
    //API call
    var data_file =
"https://api.mlab.com/api/1/databases/pbdb/collections/query4?apiKey=qu
Ja8qCv_KGUvY5S3Qnf9EDnWzoDvSQA";
    var http_request = new XMLHttpRequest();

    http_request.onreadystatechange = function() {
        if (http_request.readyState == 4) {
            var jsonObj = JSON.parse(http_request.responseText);
            var data = new google.visualization.DataTable();
            data.addColumn('string', 'Location');
            data.addColumn('string', 'Number of users');
            for (var i = 0; i < jsonObj.length; i++) {
                data.addRows([
                    [jsonObj[i].location, "City:
"+jsonObj[i].location+" | No of Tweets: "+jsonObj[i].users]
                ]);
            }
            var options = {
                zoomLevel: 3,
                showTooltip: true,
                showInfoWindow: true,
                icons:{
                    default:{
```

```
normal:'http://icons.iconarchive.com/icons/graphics-vibe/media-pin-
social/48/twitter-icon.png'
                    }
                }
            };

            var map = new
google.visualization.Map(document.getElementById('map_chart'));

            map.draw(data, options);
        }
    }
    http_request.open("GET", data_file, true);
    http_request.send();
}
```