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

Theme: (Wrestling)

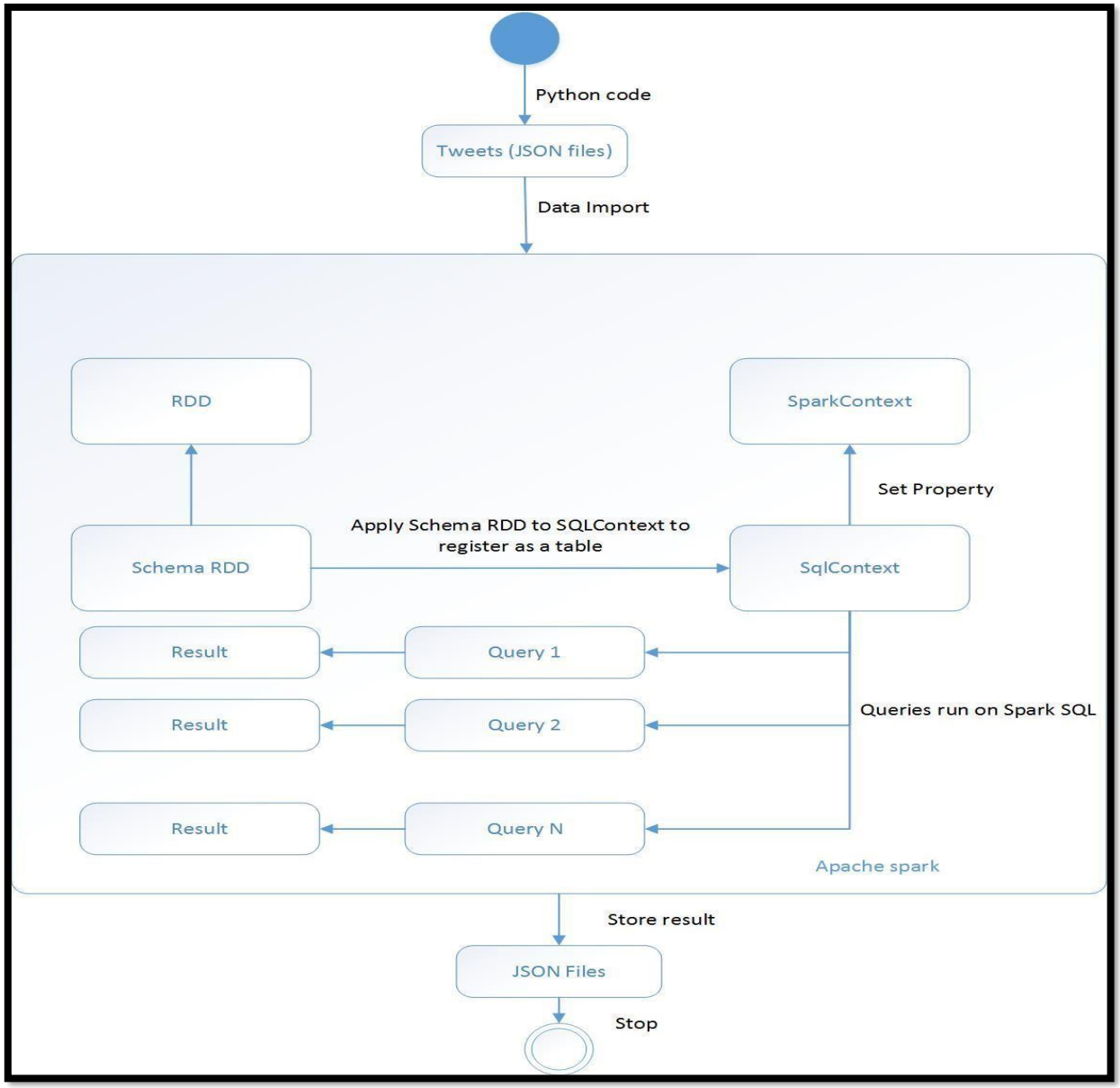
The Ultimate Fighting Championship (UFC) is the world's leading mixed martial arts (MMA) promoter and has held over 300 events to date. UFC is a combat sport abide by Unified Rules of Mixed Martial Arts where the outcomes are pre-determined and the matches are not choreographed. The UFC also connects with tens of millions of fans through its social media sites like Facebook, Instagram, and Twitter. The estimated tweets posted per hour (based on 1% sample) about #UFC is around 800.

World Wrestling Entertainment, Inc. (WWE) is an entertainment company that deals primarily in professional wrestling. WWE shows are purely entertainment based, featuring storyline-driven, scripted and choreographed matches. The estimated tweets posted per hour (based on 1% sample) about #WWE is around 750.

References:

1. Wikipedia
2. Hashtags.org Analytics

UML Diagram:


Python code
Tweets (JSON files)
Data Import
RDD
SparkContext
Set Property
Apply Schema RDD to SQLContext to register as a table
Schema RDD
SqlContext
Result
Query 1
Result
Query 2
Queries run on Spark SQL
Result
Query N
Apache spark
Store result
JSON Files
Stop

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 as JSON file.

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

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.

Programming Languages:

1. Scala – to run Spark Programs.
2. Python – to run Tweets collection program.

Environment:

Runtime Information:

Name	*Value*
Java Version	1.8.0_101 (Oracle Corporation)
Scala Version	version 2.11.8

Spark Properties:

Name	*Value*
spark.sql.warehouse.dir	file:///c:/tmp/spark-warehouse
spark.scheduler.mode	FIFO
spark.master	local[2]
spark.executor.id	driver
spark.driver.port	55681
spark.driver.host	192.168.1.146
spark.app.name	CountSpark
spark.app.id	local-1478459427915

System Properties:

Name	*Value*
file.encoding	UTF-8
hadoop.home.dir	C:\hadoop-2.3.0\bin\tweet
idea.launcher.bin.path	C:\Program Files (x86)\JetBrains\IntelliJ IDEA Community Edition 2016.2.5\bin
os.arch	amd64
os.name	Windows 10
os.version	10.0

Queries:

Query 1:

Description:

Query to display the top 10 users who tweeted the most times.

Code:

```
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")
```

Query 2:

Description:

Query to display the top 10 users with most Sensitive Tweet numbers.

Code:

```
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")
```

Query 3:

Description:

Query to display the top hashtags used in my collected tweets in conjunction with data in the HashtagsTopics.txt file posted on Blackboard.

Code:

```
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 ")
```

Query 4:

Description:

Query to display cities from which most tweets and retweets are posted.

Code:

```
Query4=string.flatMap(x=>(x.split(",\""))) .filter(line=>line.contains("location"))

.flatMap(x=>(x.split("location\":"))) .filter(x => x!="null").filter(x => x!="")

.filter(line=>line.contains(",")) .map(temp => (temp,1)) .reduceByKey(_+_) .sortBy(_._2,false)

.take(10) .foreach(println)
```

Query 5:

Description:

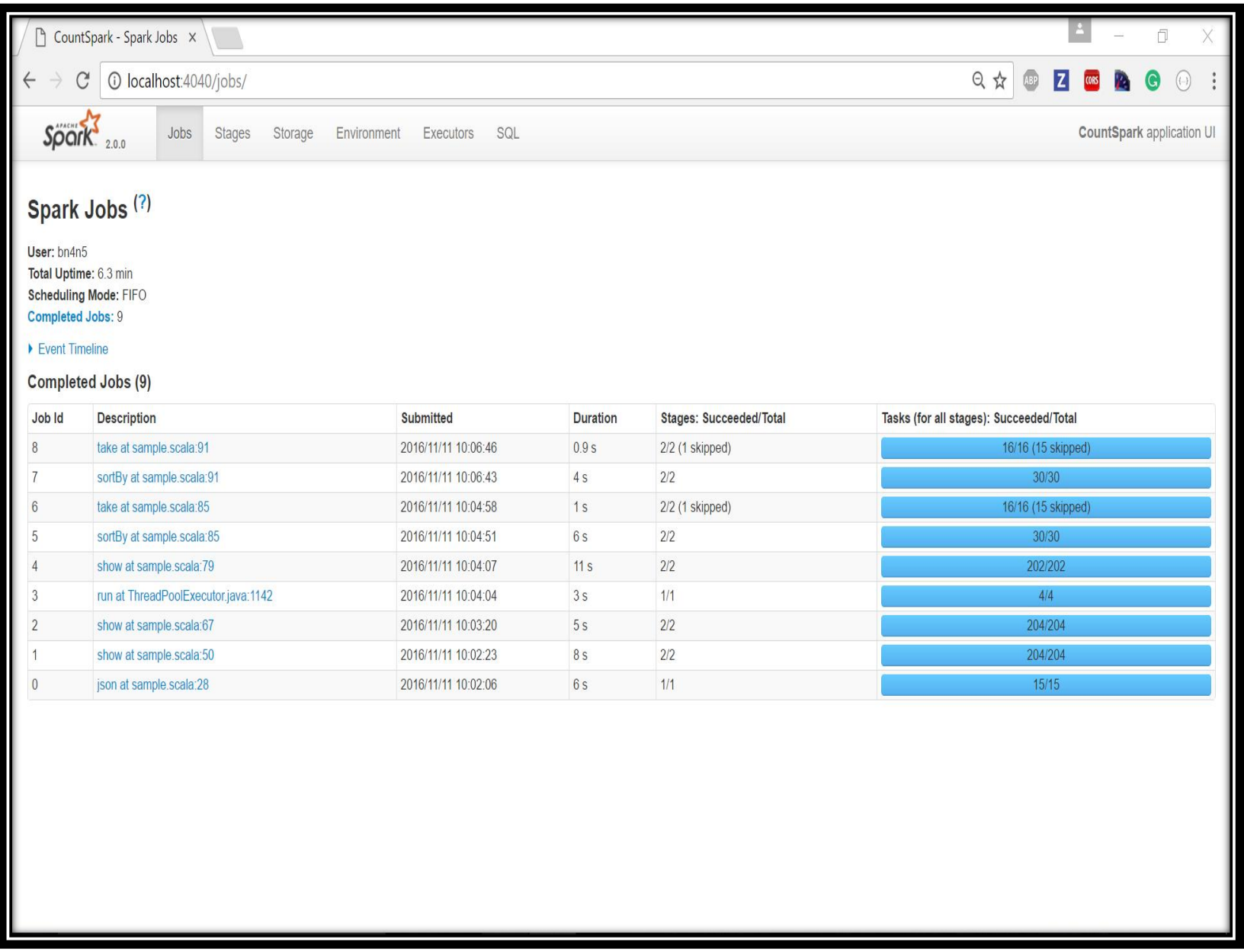
Query to display the most popular time zones.

Code:

```
Query5=string.flatMap(x =>(x.split(","))) .filter(line=>line.contains("time_zone"))

.flatMap(x =>(x.split("\"time_zone\":"))) .filter(x => x!="null") .filter(x => x!="") .map(temp =>

(temp,1)).reduceByKey(_+_).sortBy(_._2,false).take(10).foreach(println)
```

Runtime Measurements for Queries:

Query	***Total*** **(sec)**
Query 1 – Data Frame	8
Query 2 – Data Frame	5
Query 3 – Data Frame	14
Query 4 – RDD	7
Query 5 – RDD	4.9



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.
sql.warehouse.dir","file:///c:/tmp/spark-warehouse")     val sc = new
SparkContext(conf)     val sqlcontext = new SQLContext(sc)     import
sqlcontext.implicits._

    //Spark DataFrames
val tweetsfile =
sqlcontext.read.json("C:\\Users\\bn4n5\\workspace\\Pbass\\mypackage\\fe
tched_tweet.json")     tweetsfile.registerTempTable("querytable1")
```

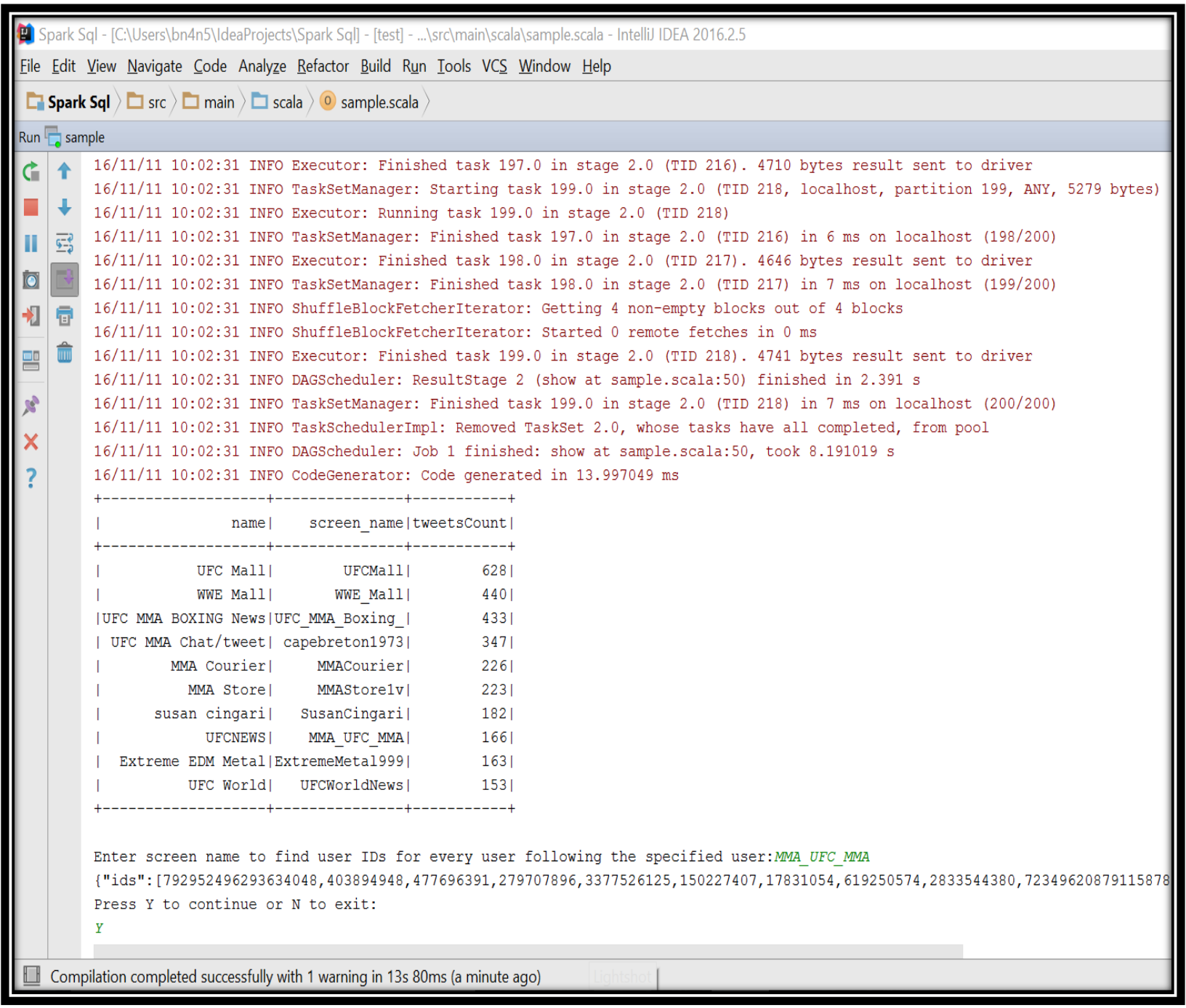
```
    //Spark RDD's
    val
string=sc.textFile("C:\\Users\\bn4n5\\workspace\\Pbass\\mypackage\\fetc
hed_tweet.json")
     var a='Y'
while (a=='Y') {
//Menu Option
    println("****** Analytical Queries using Apache Spark ******")
println("1=>Top Users who has Tweeted the most times")
println("2=>Users 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 and Retweets posted")
println("5=>Most Popular Time Zones")
println("Enter your choice:")     val
choice=readInt()       choice match {

        case 1 =>
          //Query 1 using Spark DataFrames
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.write.json("C:\\Users\\bn4n5\\workspace\\Pbass\\mypackage\\Query
1")
          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()

        case 2 =>
          //Query 2 using Spark DataFrames
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.write.json("C:\\Users\\bn4n5\\workspace\\Pbass\\mypack
age\\Query2")           Query2.show()
          println("Press Y to continue or N to exit:")
a = readChar()

        case 3 =>
          //Query 3 using Spark DataFrames
          //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\\Pbass\\mypackage\\PopularHah
tagsAndTopics.txt").map(_.split("/n")).map(f rt =>
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.write.json("C:\\Users\\bn4n5\\workspace\\Pbass\\mypack
age\\Query3")           Query3.show();
          println("Press Y to continue or N to exit:")
a = readChar()

     case 4 =>
          //Query 4 using Spark RDD's
          val Query4=string.flatMap(x
=>(x.split(",\""))).filter(line=>line.contains("location")).flatMap(x=
>(x.split("location\":"))).filter(x => x!="null").filter(x => x!="")
.filter(line=>line.contains(",")).map(temp => (temp,1))
.reduceByKey(_+_).sortBy(_._2,false).take(10).foreach(println)
          println("Press Y to continue or N to exit:")
          a = readChar()

        case 5 =>
          //Query 5 using Spark RDD's
          val Query5=string.flatMap(x
=>(x.split(","))).filter(line=>line.contains("time_zone")).flatMap(x
=>(x.split("\"time_zone\":"))).filter(x => x!="null").filter(x =>
x!="").map(temp => (temp,1)) .reduceByKey(_+_) .sortBy(_._2,false)
.take(10).foreach(println)
          println("Press Y to continue or N to exit:")
          a = readChar()
      }
    }
  }
}
case class Text(name: String)
```

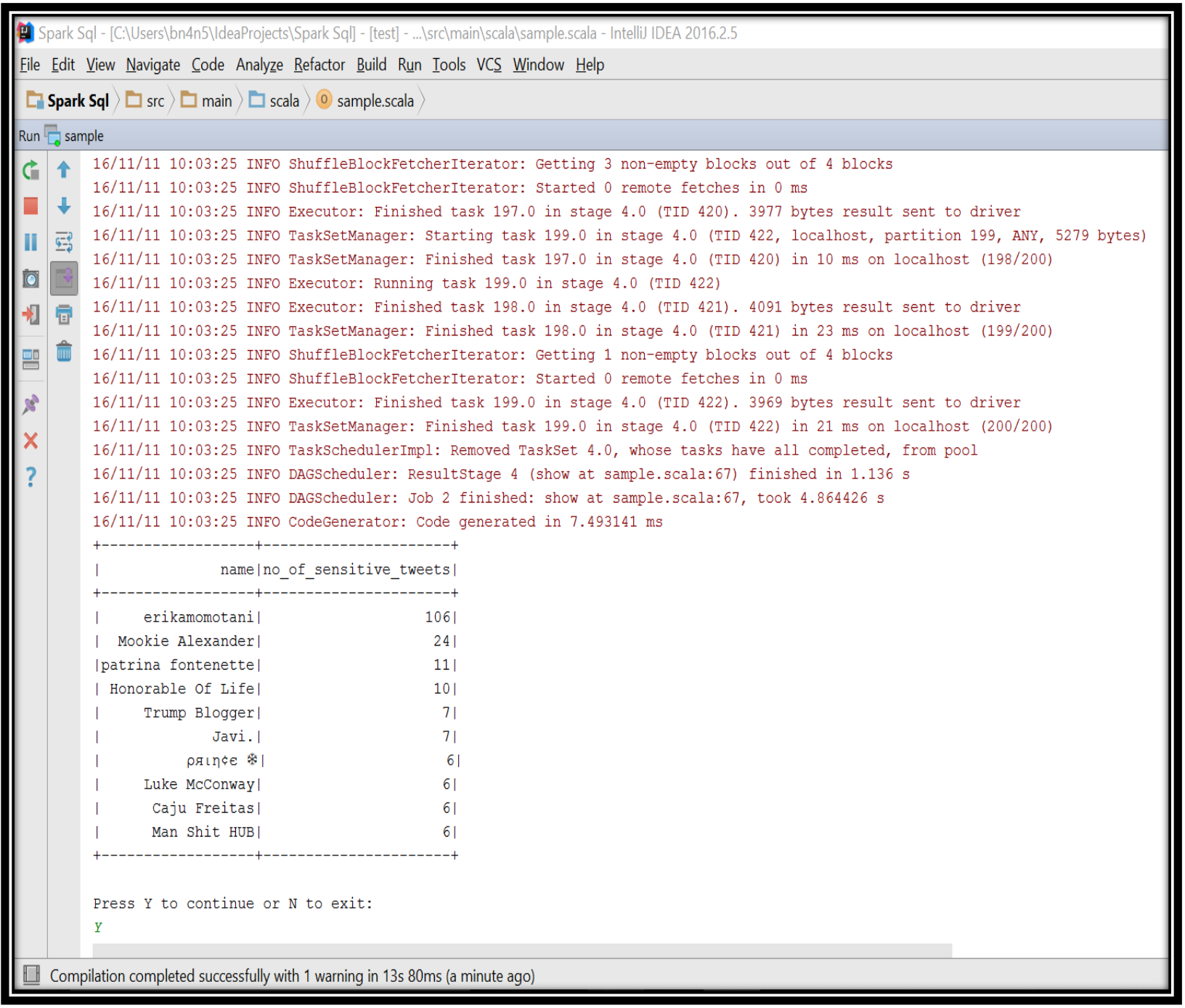
Output:

Query 1: **Top Users who has Tweeted the most times (Twitter API)**



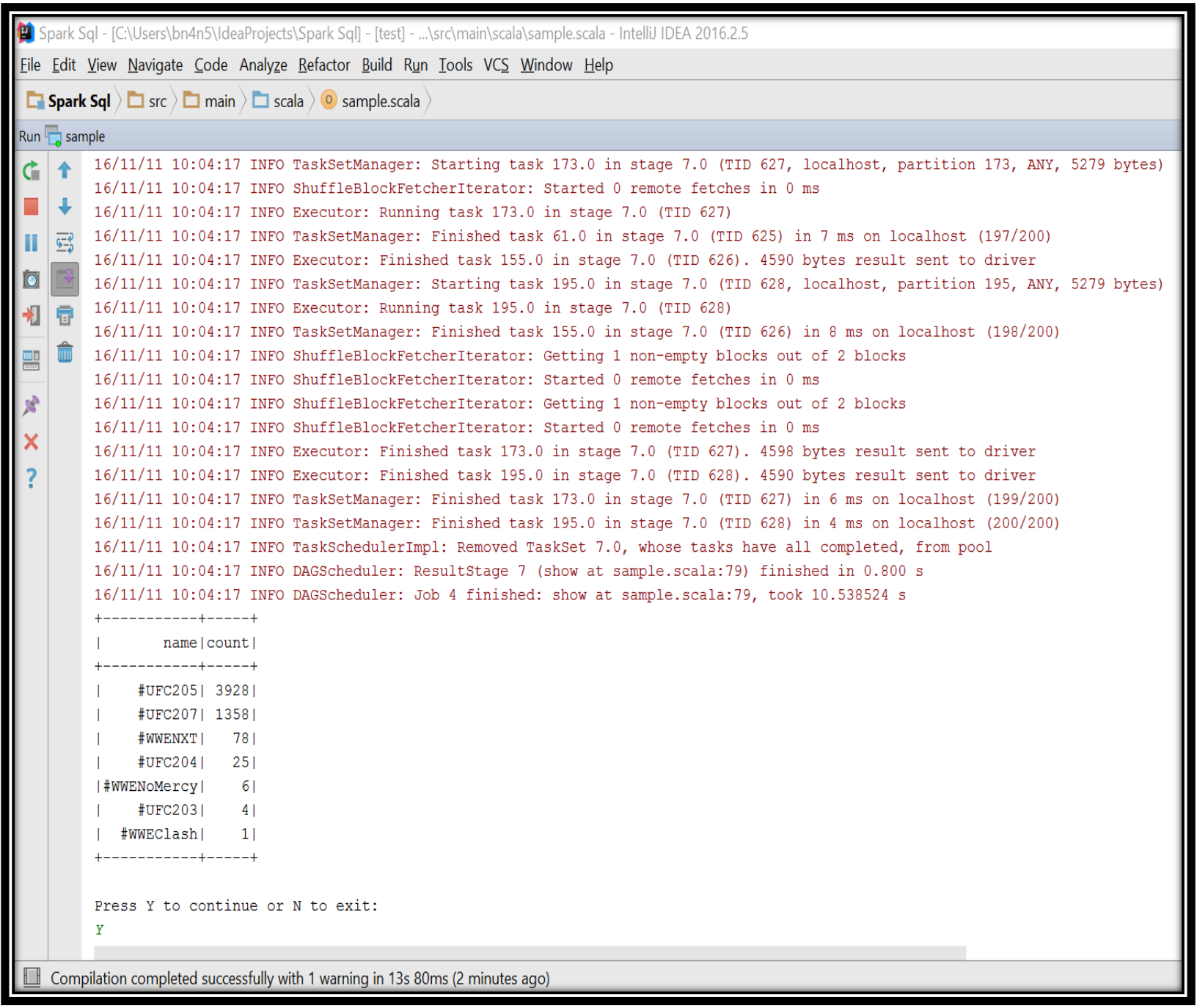
Query 2:

Users with Most Sensitive Tweet Numbers



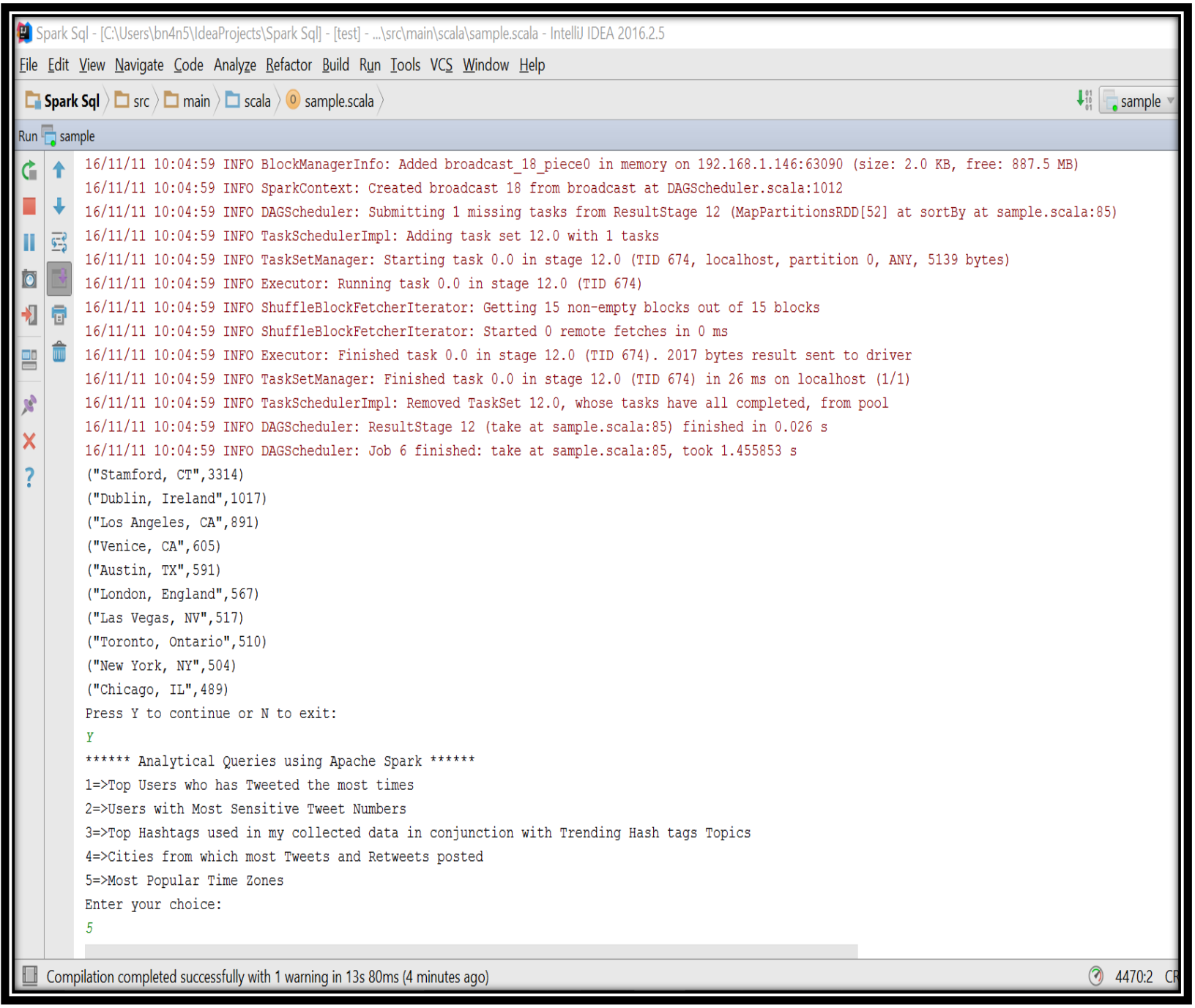
Query 3:

Top Hashtags used in my collected data in conjunction with Trending Hashtags Topics



Query 4:

Cities from which most Tweets and Retweets posted



Query 5:

Most Popular Time Zones