Stage1: Collection of data from twitter using a keyword “AWARDS” from a twitter streaming API to analyze the data.

- A twitter application is created for accessing twitter data
- Python code is used to get required authorizations for connecting twitter API
- Tweepy library from python file is used to download the streaming tweets filtered on ‘Awards’
- All the data is collected into local system in json and text format

Sample tweet data from json file: (single record)

```
{"created_at":"Tue Nov 01 04:13:01 +0000
2016","id":793304813937188865,"id_str":"793304813937188865","text":"201
5 Click! StarWars Awards\u6295\u7968\u4e2d https:\/\/t.co\/uWPbp8d2dd
#SHINee \u73fe\u5728\u306e\u7dcf\u5408 1\u4f4d,
\u6295\u7968\u9032\u884c\u4e2d!!! - 2016-Nov-01 13:13:02
@thefactnews","source":"\u003ca href=\"http:\/\/starwars.tf.co.kr\"
rel=\"nofollow\"\u003eClick!
StarWars\u003c\/a\u003e","truncated":false,"in_reply_to_status_id":null,"in_re
ply_to_status_id_str":null,"in_reply_to_user_id":null,"in_reply_to_user_id_str"
:null,"in_reply_to_screen_name":null,"user":{"id":1427984354,"id_str":"14279
84354","name":"momomomo_ri","screen_name":"momomomo_ri","location":
n
ull,"url":null,"description":null,"protected":false,"verified":false,"followers_cou
nt":0,"friends_count":3,"listed_count":0,"favourites_count":135,"statuses_count
":244,"created_at":"Tue May 14 14:18:57 +0000
2013","utc_offset":28800,"time_zone":"Irkutsk","geo_enabled":false,"lang":"ja
","contributors_enabled":false,"is_translator":false,"profile_background_color":
"C0DEED","profile_background_image_url":"http:\/\/abs.twimg.com\/images\/
t
hemes\/theme1\/bg.png","profile_background_image_url_https":"https:\/\/abs.t
wimg.com\/images\/themes\/theme1\/bg.png","profile_background_tile":false,"
profile_link_color":"0084B4","profile_sidebar_border_color":"C0DEED","prof
ile_sidebar_fill_color":"DDEEF6","profile_text_color":"333333","profile_use_
```

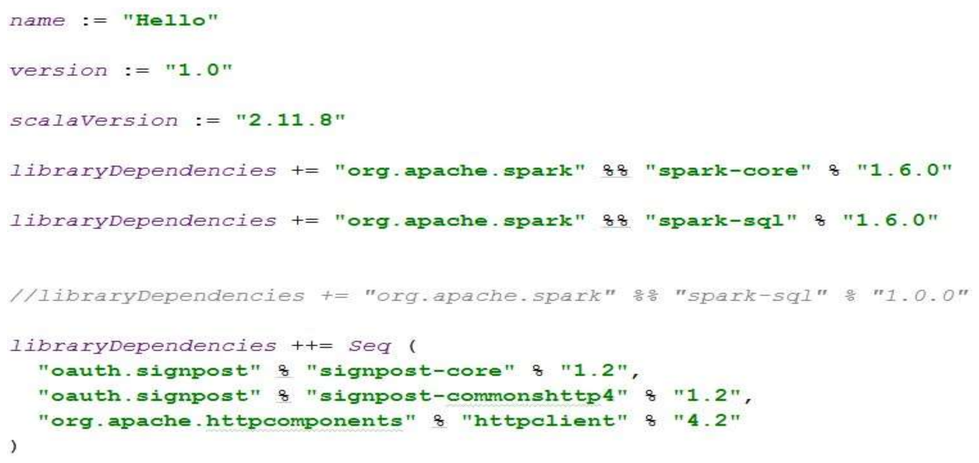
background_image":true,"profile_image_url":"http:\/\/pbs.twimg.com\/profile_images\/784954873397653505\/DpJmci72_normal.jpg","profile_image_url_https":"https:\/\/pbs.twimg.com\/profile_images\/784954873397653505\/DpJmci72_normal.jpg","profile_banner_url":"https:\/\/pbs.twimg.com\/profile_banners\/1427984354\/1475982829","default_profile":true,"default_profile_image":false,"following":null,"follow_request_sent":null,"notifications":null},"geo":null,"coordinates":null,"place":null,"contributors":null,"is_quote_status":false,"retweet_count":0,"favorite_count":0,"entities":{"hashtags":[{"text":"SHINee","indices":[55,62]}],"urls":[{"url":"https:\/\/t.co\/uWPbp8d2dd","expanded_url":"http:\/\/starwars.thefactjp.com","display_url":"starwars.thefactjp.com","indices":[31,54]}],"user_mentions":[{"screen_name":"thefactnews","name":"\ub354\ud329\ud2b8","id":167266914,"id_str":"167266914","indices":[105,117]}],"symbols":[]},"favorited":false,"retweeted":false,"possibly_sensitive":false,"filter_level":"low","lang":"ja","timestamp_ms":"1477973581190"}

Stage 2: Loading the dependencies

We can add both managed and unmanaged dependencies in our SBT projects. We add a *libraryDependencies* line in our *build.sbt* file:

```
libraryDependencies += "org.apache.spark" %% "spark-core" % "1.6.0"
```

The configuration lines in *build.sbt* file must be separated by blank lines , the following lines are the configuration lines in our *build.sbt* file:



```
name := "Hello"

version := "1.0"

scalaVersion := "2.11.8"

libraryDependencies += "org.apache.spark" %% "spark-core" % "1.6.0"

libraryDependencies += "org.apache.spark" %% "spark-sql" % "1.6.0"


//libraryDependencies += "org.apache.spark" %% "spark-sql" % "1.0.0"

libraryDependencies ++= Seq (
  "oauth.signpost" % "signpost-core" % "1.2",
  "oauth.signpost" % "signpost-commonshttp4" % "1.2",
  "org.apache.httpcomponents" % "httpclient" % "4.2"
)
```

Now create an object file in \src\main\scala\. In this project *Awards.scala* is the object file created. In this we set Hadoop home directory.

Stage 3: Loading tweets data

We load json file to perform Spark SQL Data frame and text file is loaded to perform Spark SQL RDD's.

The following line is added to load the json data:

```
sqlContext.read.json("C:\\Users\\SNEHADIDIGAM\\Desktop\\Winuti
ls\\data2.json")
```

The following line is added to load the json data:

```
sc.textFile("C:\\Users\\SNEHA
DIDIGAM\\Desktop\\Winutils\\tweets1.txt")
```

Stage 4: Implementing data

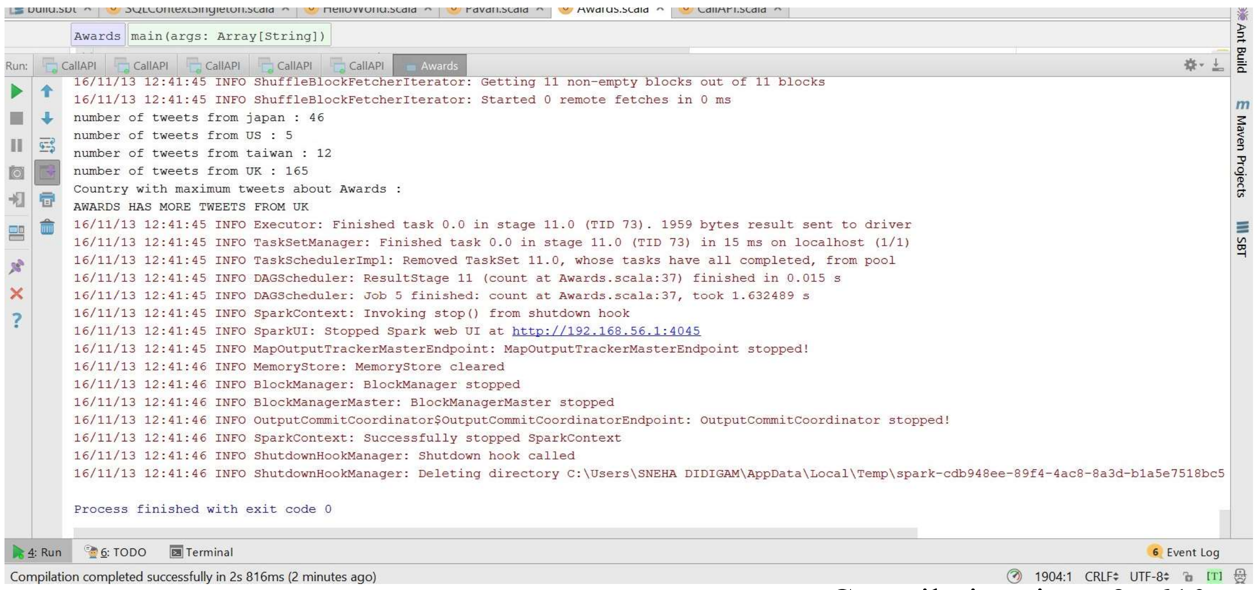
Following are the 5 analytical queries written in SparkRDD and Data frame queries

QUERY 1: TOP COUNTRY TWEETED ABOUT TOPIC "AWARDS"

Description: Query here gives the country name with more number of tweets posted about “Awards” among different countries

```
val japanT=sqlContext.sql("select text from dframetable where user.location='Japan'
and text like '%Awards%' ") val japanTC=japanT.count()

val UST=sqlContext.sql("select text from dframetable where user.location='US' and text
like '%Awards%'") val USTC=UST.count()

val taiwanT=sqlContext.sql("select text from dframetable where
user.location='Taiwan'and text like '%Awards%' ")
val taiwanTC=taiwanT.count()

val UKT=sqlContext.sql("select text from dframetable where user.location='UK' and text
like '%Awards%'") val UKTC=UKT.count()

println("number of tweets from japan : "+japanTC)
println("number of tweets from US : "+USTC) println("number
of tweets from taiwan : "+taiwanTC) println("number of
tweets from UK : "+UKTC) println("Country with maximum
tweets about Awards : ")

if(japanTC > USTC && japanTC > taiwanTC && japanTC > UKTC ){
println("AWARDS HAS MORE TWEETS FROM JAPAN")
}
if(USTC > japanTC && USTC > taiwanTC && USTC > UKTC ){
println("AWARDS HAS MORE TWEETS FROM US")
}
if(taiwanTC > USTC && taiwanTC > japanTC && taiwanTC > UKTC ){
println("AWARDS HAS MORE TWEETS FROM JAPAN")
}
if(UKTC > japanTC && UKTC > taiwanTC && UKTC > USTC ){
println("AWARDS HAS MORE TWEETS FROM UK")
}
```

OUTPUT:



```
16/11/13 12:41:45 INFO ShuffleBlockFetcherIterator: Getting 11 non-empty blocks out of 11 blocks
16/11/13 12:41:45 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
number of tweets from japan : 46
number of tweets from US : 5
number of tweets from taiwan : 12
number of tweets from UK : 165
Country with maximum tweets about Awards :
AWARDS HAS MORE TWEETS FROM UK
16/11/13 12:41:45 INFO Executor: Finished task 0.0 in stage 11.0 (TID 73). 1959 bytes result sent to driver
16/11/13 12:41:45 INFO TaskSetManager: Finished task 0.0 in stage 11.0 (TID 73) in 15 ms on localhost (1/1)
16/11/13 12:41:45 INFO TaskSchedulerImpl: Removed TaskSet 11.0, whose tasks have all completed, from pool
16/11/13 12:41:45 INFO DAGScheduler: ResultStage 11 (count at Awards.scala:37) finished in 0.015 s
16/11/13 12:41:45 INFO DAGScheduler: Job 5 finished: count at Awards.scala:37, took 1.632489 s
16/11/13 12:41:45 INFO SparkContext: Invoking stop() from shutdown hook
16/11/13 12:41:45 INFO SparkUI: Stopped Spark web UI at http://192.168.56.1:4045
16/11/13 12:41:45 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
16/11/13 12:41:46 INFO MemoryStore: MemoryStore cleared
16/11/13 12:41:46 INFO BlockManager: BlockManager stopped
16/11/13 12:41:46 INFO BlockManagerMaster: BlockManagerMaster stopped
16/11/13 12:41:46 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
16/11/13 12:41:46 INFO SparkContext: Successfully stopped SparkContext
16/11/13 12:41:46 INFO ShutdownHookManager: Shutdown hook called
16/11/13 12:41:46 INFO ShutdownHookManager: Deleting directory C:\Users\SNEHA DIDIGAM\AppData\Local\Temp\spark-cdb948ee-89f4-4ac8-8a3d-b1a5e7518bc5

Process finished with exit code 0
```

Compilation completed successfully in 2s 816ms (2 minutes ago)

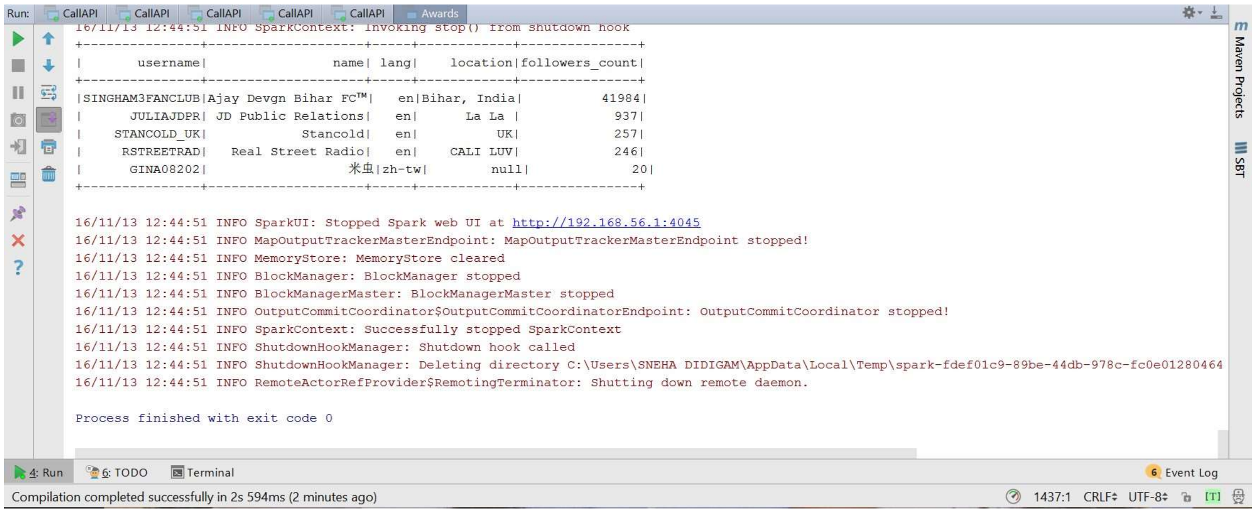
Compilation time: 3s 610ms

QUERY 2: FAMOUS 5 USERS WHO TWEETED ABOUT "OSCAR" FROM SAME TIME_ZONE

Description: Query gives top 5 users (based on followers count) who tweeted about "OSCAR" from "Pacific Time" time zone .

```
val queryyy=sqlContext.sql("select upper(user.screen_name) as username, user.name,
user.lang, user.location, user.followers_count from dframetable where text LIKE
'%Oscar%' and user.time_zone='Pacific Time (US & Canada)' order by
user.followers_count desc limit 5")
```

OUTPUT:



```
16/11/13 12:44:51 INFO SparkContext: Invoking stop() from shutdown hook
+---------------+--------------------+-----+------------+---------------+
|       username|                name| lang|    location|followers_count|
+---------------+--------------------+-----+------------+---------------+
|SINGHAM3FANCLUB|Ajay Devgn Bihar FC™|   en|Bihar, India|          41984|
|      JULIAJDPR| JD Public Relations|   en|      La La |            937|
|    STANCOLD_UK|            Stancold|   en|          UK|            257|
|     RSTREETRAD|   Real Street Radio|   en|    CALI LUV|            246|
|      GINA08202|                  米虫|zh-tw|        null|             20|
+---------------+--------------------+-----+------------+---------------+

16/11/13 12:44:51 INFO SparkUI: Stopped Spark web UI at http://192.168.56.1:4045
16/11/13 12:44:51 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
16/11/13 12:44:51 INFO MemoryStore: MemoryStore cleared
16/11/13 12:44:51 INFO BlockManager: BlockManager stopped
16/11/13 12:44:51 INFO BlockManagerMaster: BlockManagerMaster stopped
16/11/13 12:44:51 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
16/11/13 12:44:51 INFO SparkContext: Successfully stopped SparkContext
16/11/13 12:44:51 INFO ShutdownHookManager: Shutdown hook called
16/11/13 12:44:51 INFO ShutdownHookManager: Deleting directory C:\Users\SNEHA DIDIGAM\AppData\Local\Temp\spark-fdef01c9-89be-44db-978c-fc0e01280464
16/11/13 12:44:51 INFO RemoteActorRefProvider$RemotingTerminator: Shutting down remote daemon.

Process finished with exit code 0
```

```
Compilation completed successfully in 2s 594ms (2 minutes ago)
```

Compilation time: 3s 595ms

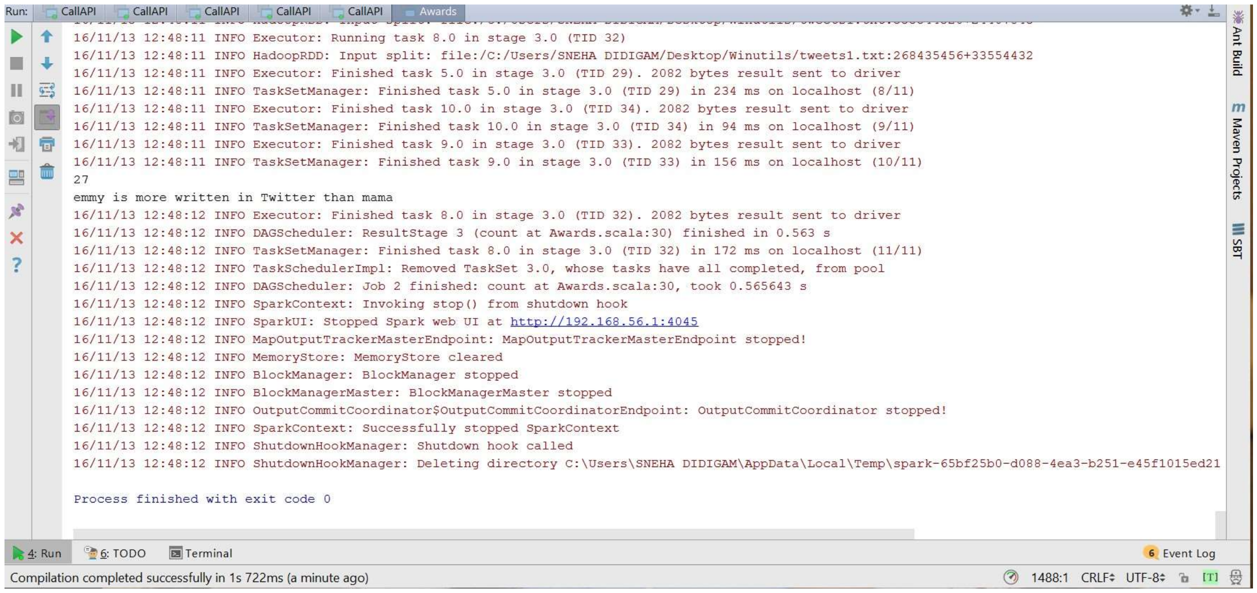
QUERY 3: COMPARE MAXIMUM TWEETS BETWEEN “EMMY AWARDS” AND “MAMA AWARDS”

Description: RDD Query here compares the number of tweets posted on EMMY and MAMA Awards. We used count() and compareTo() operations.

```
val rddData=sc.textFile("C:\\Users\\SNEHA DIDIGAM\\Desktop\\Winutils\\tweets1.txt") val
mamacount = rddData.filter(line=>line.contains("#mama")).count() println(mamacount)
val oscarcount=rddData.filter(l=>l.contains("#Emmy")).count()
println(oscarcount)
if(mamacount.compareTo(oscarcount)==1) {
  println("mama is more written in Twitter than emmy")
}
else
{
  println("emmy is more written in Twitter than mama")
}
```

OUTPUT:

```
mama awards tweets :15 emmy awards tweets :27 emmy

awards has more tweets than mama awardsQUERY 5
```



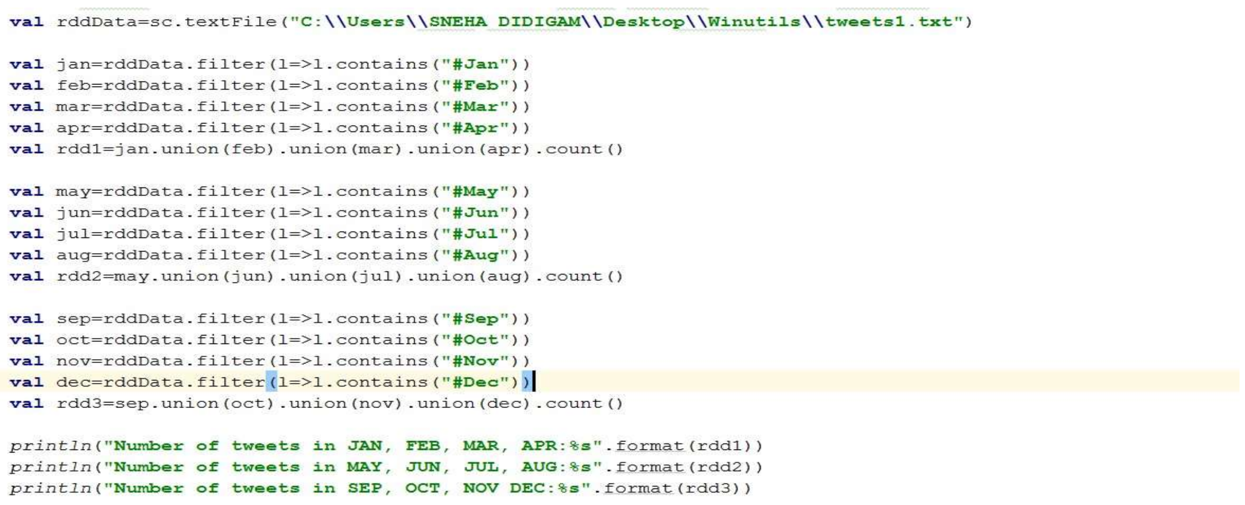
```
16/11/13 12:48:11 INFO Executor: Running task 8.0 in stage 3.0 (TID 32)
16/11/13 12:48:11 INFO HadoopRDD: Input split: file:/C:/Users/SNEHA DIDIGAM/Desktop/Winutils/tweets1.txt:268435456+33554432
16/11/13 12:48:11 INFO Executor: Finished task 5.0 in stage 3.0 (TID 29). 2082 bytes result sent to driver
16/11/13 12:48:11 INFO TaskSetManager: Finished task 5.0 in stage 3.0 (TID 29) in 234 ms on localhost (8/11)
16/11/13 12:48:11 INFO Executor: Finished task 10.0 in stage 3.0 (TID 34). 2082 bytes result sent to driver
16/11/13 12:48:11 INFO TaskSetManager: Finished task 10.0 in stage 3.0 (TID 34) in 94 ms on localhost (9/11)
16/11/13 12:48:11 INFO Executor: Finished task 9.0 in stage 3.0 (TID 33). 2082 bytes result sent to driver
16/11/13 12:48:11 INFO TaskSetManager: Finished task 9.0 in stage 3.0 (TID 33) in 156 ms on localhost (10/11)
27
emmy is more written in Twitter than mama
16/11/13 12:48:12 INFO Executor: Finished task 8.0 in stage 3.0 (TID 32). 2082 bytes result sent to driver
16/11/13 12:48:12 INFO DAGScheduler: ResultStage 3 (count at Awards.scala:30) finished in 0.563 s
16/11/13 12:48:12 INFO TaskSetManager: Finished task 8.0 in stage 3.0 (TID 32) in 172 ms on localhost (11/11)
16/11/13 12:48:12 INFO TaskSchedulerImpl: Removed TaskSet 3.0, whose tasks have all completed, from pool
16/11/13 12:48:12 INFO DAGScheduler: Job 2 finished: count at Awards.scala:30, took 0.565643 s
16/11/13 12:48:12 INFO SparkContext: Invoking stop() from shutdown hook
16/11/13 12:48:12 INFO SparkUI: Stopped Spark web UI at http://192.168.56.1:4045
16/11/13 12:48:12 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
16/11/13 12:48:12 INFO MemoryStore: MemoryStore cleared
16/11/13 12:48:12 INFO BlockManager: BlockManager stopped
16/11/13 12:48:12 INFO BlockManagerMaster: BlockManagerMaster stopped
16/11/13 12:48:12 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
16/11/13 12:48:12 INFO SparkContext: Successfully stopped SparkContext
16/11/13 12:48:12 INFO ShutdownHookManager: Shutdown hook called
16/11/13 12:48:12 INFO ShutdownHookManager: Deleting directory C:\Users\SNEHA DIDIGAM\AppData\Local\Temp\spark-65bf25b0-d088-4ea3-b251-e45f1015ed21

Process finished with exit code 0
```

Compilation completed successfully in 1s 722ms (a minute ago)

Compilation time: 4s 203ms

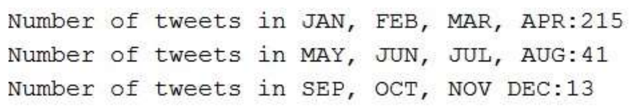
QUERY 4: NUMBER OF TWEETS BASED ON MONTH

Description: The following RDD query performs union operation filtered based months. We have performed union() operation.



```
val rddData=sc.textFile("C:\\Users\\SNEHA DIDIGAM\\Desktop\\Winutils\\tweets1.txt")

val jan=rddData.filter(l=>l.contains("#Jan"))
val feb=rddData.filter(l=>l.contains("#Feb"))
val mar=rddData.filter(l=>l.contains("#Mar"))
val apr=rddData.filter(l=>l.contains("#Apr"))
val rdd1=jan.union(feb).union(mar).union(apr).count()

val may=rddData.filter(l=>l.contains("#May"))
val jun=rddData.filter(l=>l.contains("#Jun"))
val jul=rddData.filter(l=>l.contains("#Jul"))
val aug=rddData.filter(l=>l.contains("#Aug"))
val rdd2=may.union(jun).union(jul).union(aug).count()

val sep=rddData.filter(l=>l.contains("#Sep"))
val oct=rddData.filter(l=>l.contains("#Oct"))
val nov=rddData.filter(l=>l.contains("#Nov"))
val dec=rddData.filter(l=>l.contains("#Dec"))
val rdd3=sep.union(oct).union(nov).union(dec).count()

println("Number of tweets in JAN, FEB, MAR, APR:%s".format(rdd1))
println("Number of tweets in MAY, JUN, JUL, AUG:%s".format(rdd2))
println("Number of tweets in SEP, OCT, NOV DEC:%s".format(rdd3))
```

OUTPUT:



```
Number of tweets in JAN, FEB, MAR, APR:215
Number of tweets in MAY, JUN, JUL, AUG:41
Number of tweets in SEP, OCT, NOV DEC:13
```

Compilation time: 3s 309ms

HASHTAGS QUERY

Description: Common hashtags from DataFrame table(Collected Twitter Data) and HashTable Hashtags data

```
val hTags= sqlContext.read.json("C:\\Users\\SNEHA
DIDIGAM\\Desktop\\Winutils\\hashtags.txt")
val hTagsdframe=hTags.toDF().withColumnRenamed("_corrupt_record","name") val
hashTable= hTagsdframe.registerTempTable("hashTable")


val hashquery=sqlContext.sql("select t.text as text,d.name as hashtags from
dframetable t JOIN hashTable d on t.text like '%d.name%' ") hashquery.show()
```

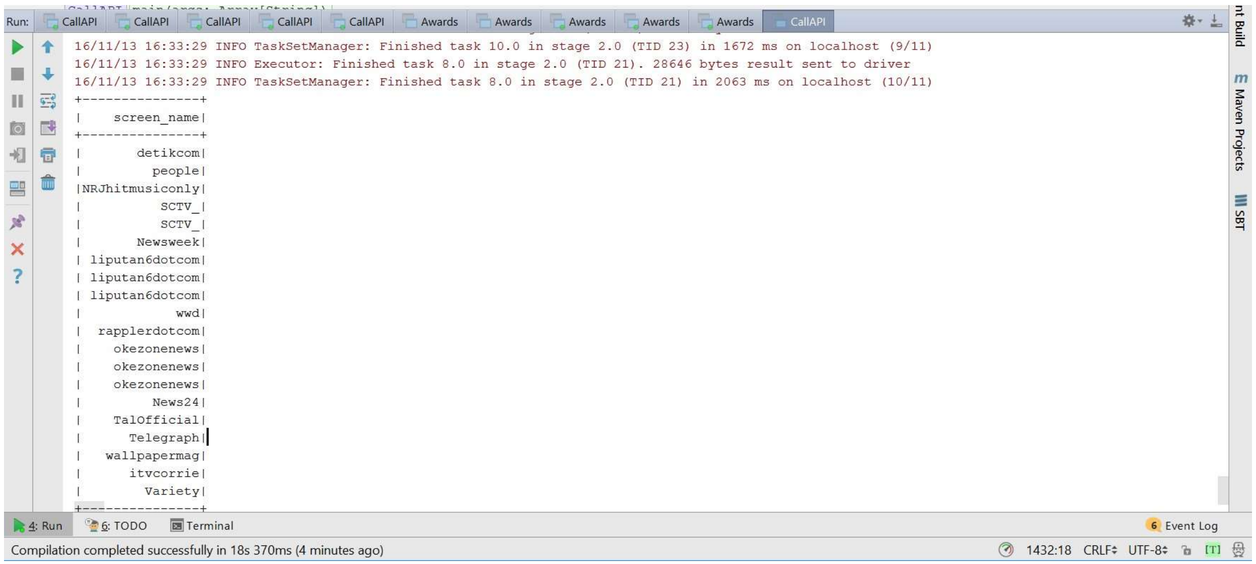
QUERY 5: CALLING THE PUBLIC API'S

Description: The following code is added in a new Scala file as object which is used to call the public APIs from Twitter

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


    val consKey = "eJxqZ4TOHItrLmyS2DWAQj7h0"
    val consSecret = "WtHaGRkVCoH1TvvmzyT1ZWLiNkgGStlsRuQG9Qt0bDpjZQGpe5"
val accessToken = "790262378738819072-XySQD2vbTj8ynWHeFXAiKXxzT58uvER"
val accessSecret = "m70V88ex33VN5lRJZOaSL1nNY4GzR2jqDVwZPcjSATmVB"


    val cons = new CommonsHttpOAuthConsumer(consKey, consSecret)
cons.setTokenWithSecret(accessToken, accessSecret)
System.setProperty("hadoop.home.dir","C:\\Users\\SNEHA
DIDIGAM\\Desktop\\Winutils")
    val sparkConf=new SparkConf().setAppName("CallAPI").setMaster("local[*]")
val sc=new SparkContext(sparkConf)
    val sqlContext=new org.apache.spark.sql.SQLContext(sc)
val dframe=sqlContext.read.json("C:\\Users\\SNEHA
DIDIGAM\\Desktop\\Winutils\\data2.json")
dframe.registerTempTable("dframetable")
dframe.printSchema()
    val username=sqlContext.sql("select user.screen_name from dframetable where text
like '%Awards%' order by user.followers_count desc")     username.show()
    val usern = scala.io.StdIn.readLine()
val getReq = new
HttpGet("https://api.twitter.com/1.1/trends/available.json?screen_name="+usern)
cons.sign(getReq)
    val client= new DefaultHttpClient()
    val results = client.execute(getReq)
println(IOUtils.toString(results.getEntity().getContent()))


  }
}
```

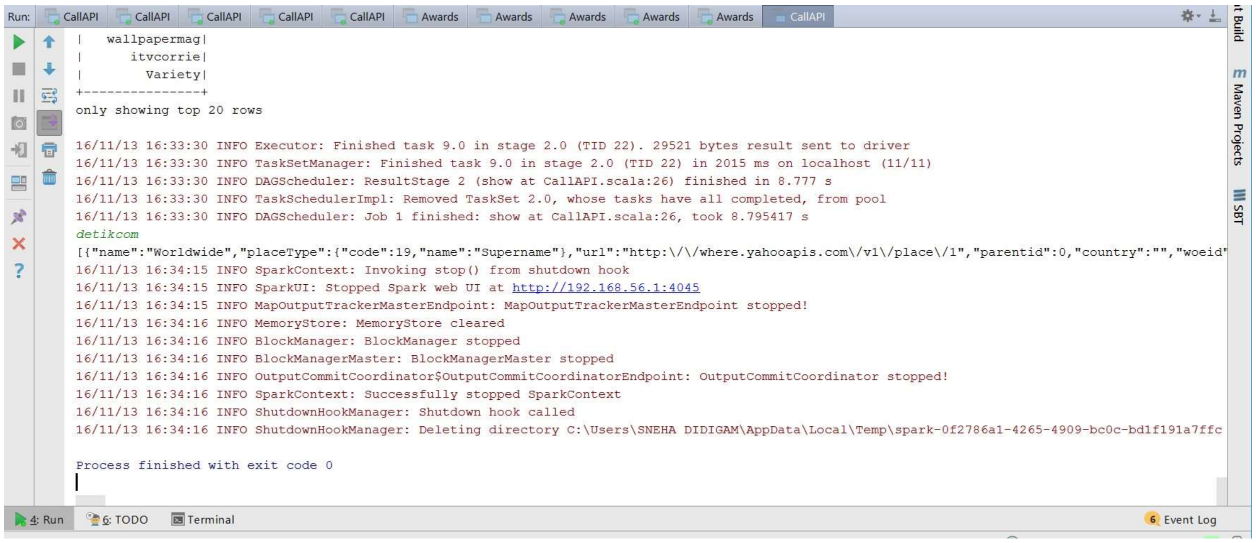
OUTPUT:

Results are:



```
16/11/13 16:33:29 INFO TaskSetManager: Finished task 10.0 in stage 2.0 (TID 23) in 1672 ms on localhost (9/11)
16/11/13 16:33:29 INFO Executor: Finished task 8.0 in stage 2.0 (TID 21). 28646 bytes result sent to driver
16/11/13 16:33:29 INFO TaskSetManager: Finished task 8.0 in stage 2.0 (TID 21) in 2063 ms on localhost (10/11)
+---------------+
|    screen_name|
+---------------+
|       detikcom|
|         people|
|NRJhitmusiconly|
|          SCTV_|
|          SCTV_|
|       Newsweek|
| liputan6dotcom|
| liputan6dotcom|
| liputan6dotcom|
|            wwd|
|  rapplerdotcom|
|    okezonenews|
|    okezonenews|
|    okezonenews|
|         News24|
|    TalOfficial|
|      Telegraph|
|   wallpapermag|
|      itvcorrie|
|        Variety|
+---------------+
```

Compilation completed successfully in 18s 370ms (4 minutes ago)



```
|   wallpapermag|
|      itvcorrie|
|        Variety|
+---------------+
only showing top 20 rows

16/11/13 16:33:30 INFO Executor: Finished task 9.0 in stage 2.0 (TID 22). 29521 bytes result sent to driver
16/11/13 16:33:30 INFO TaskSetManager: Finished task 9.0 in stage 2.0 (TID 22) in 2015 ms on localhost (11/11)
16/11/13 16:33:30 INFO DAGScheduler: ResultStage 2 (show at CallAPI.scala:26) finished in 8.777 s
16/11/13 16:33:30 INFO TaskSchedulerImpl: Removed TaskSet 2.0, whose tasks have all completed, from pool
16/11/13 16:33:30 INFO DAGScheduler: Job 1 finished: show at CallAPI.scala:26, took 8.795417 s
detikcom
[{"name":"Worldwide","placeType":{"code":19,"name":"Supername"},"url":"http:\/\/where.yahooapis.com\/v1\/place\/1","parentid":0,"country":"","woeid"
16/11/13 16:34:15 INFO SparkContext: Invoking stop() from shutdown hook
16/11/13 16:34:15 INFO SparkUI: Stopped Spark web UI at http://192.168.56.1:4045
16/11/13 16:34:15 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
16/11/13 16:34:16 INFO MemoryStore: MemoryStore cleared
16/11/13 16:34:16 INFO BlockManager: BlockManager stopped
16/11/13 16:34:16 INFO BlockManagerMaster: BlockManagerMaster stopped
16/11/13 16:34:16 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
16/11/13 16:34:16 INFO SparkContext: Successfully stopped SparkContext
16/11/13 16:34:16 INFO ShutdownHookManager: Shutdown hook called
16/11/13 16:34:16 INFO ShutdownHookManager: Deleting directory C:\Users\SNEHA DIDIGAM\AppData\Local\Temp\spark-0f2786a1-4265-4909-bc0c-bd1f191a7ffc

Process finished with exit code 0
```