

Principles of Big Data Management

Project Phase 2

Team-3

Anusha Malineni

Sujitha Puthana

Achyuth Reddy N

Sri Sai Narayana Ramgopal Mangena

1. Introduction

The main aim of this project is analyzing big data collected from twitter about Sports using apache spark for fast analysis of big data. Here we collected twitter data on sports. Later we analyzed the extracted data using spark RDD and Data frames. To write a SQL commands to analyze twitter tweets.

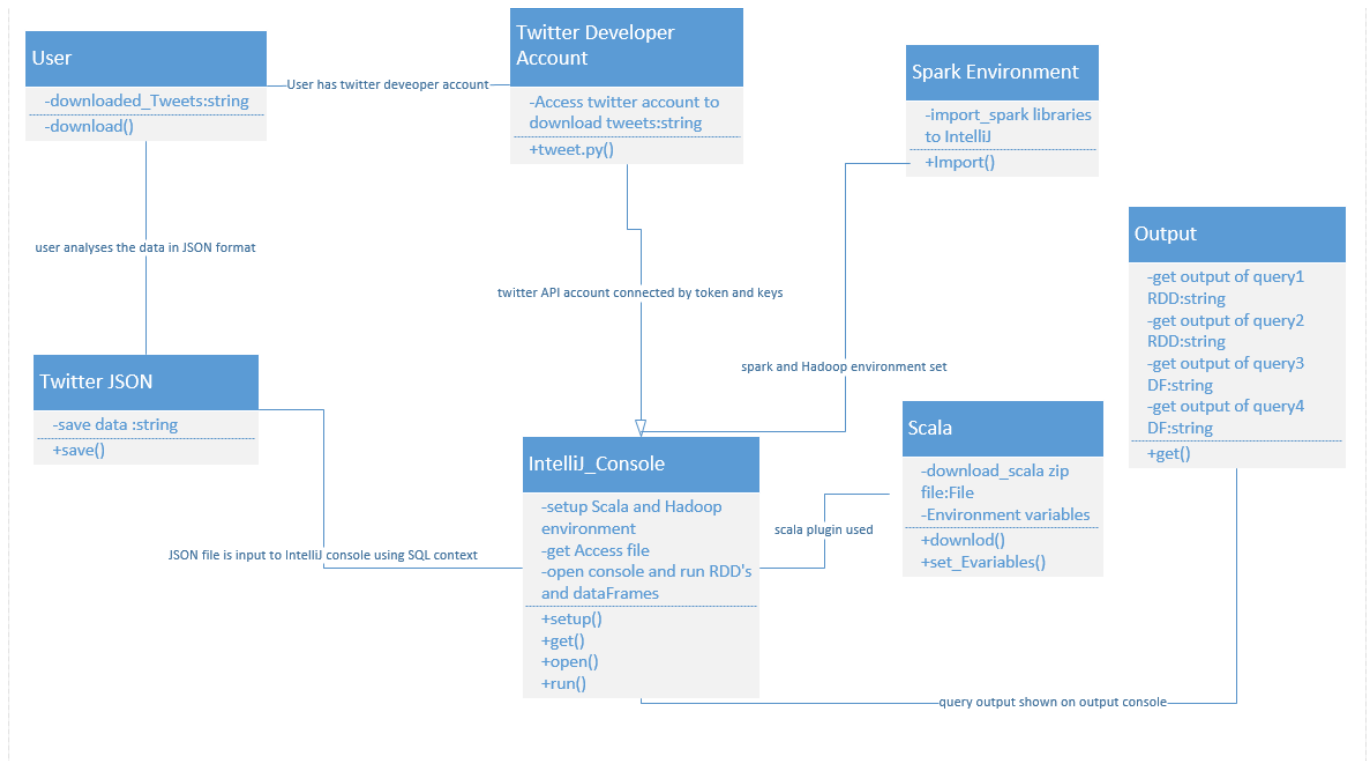
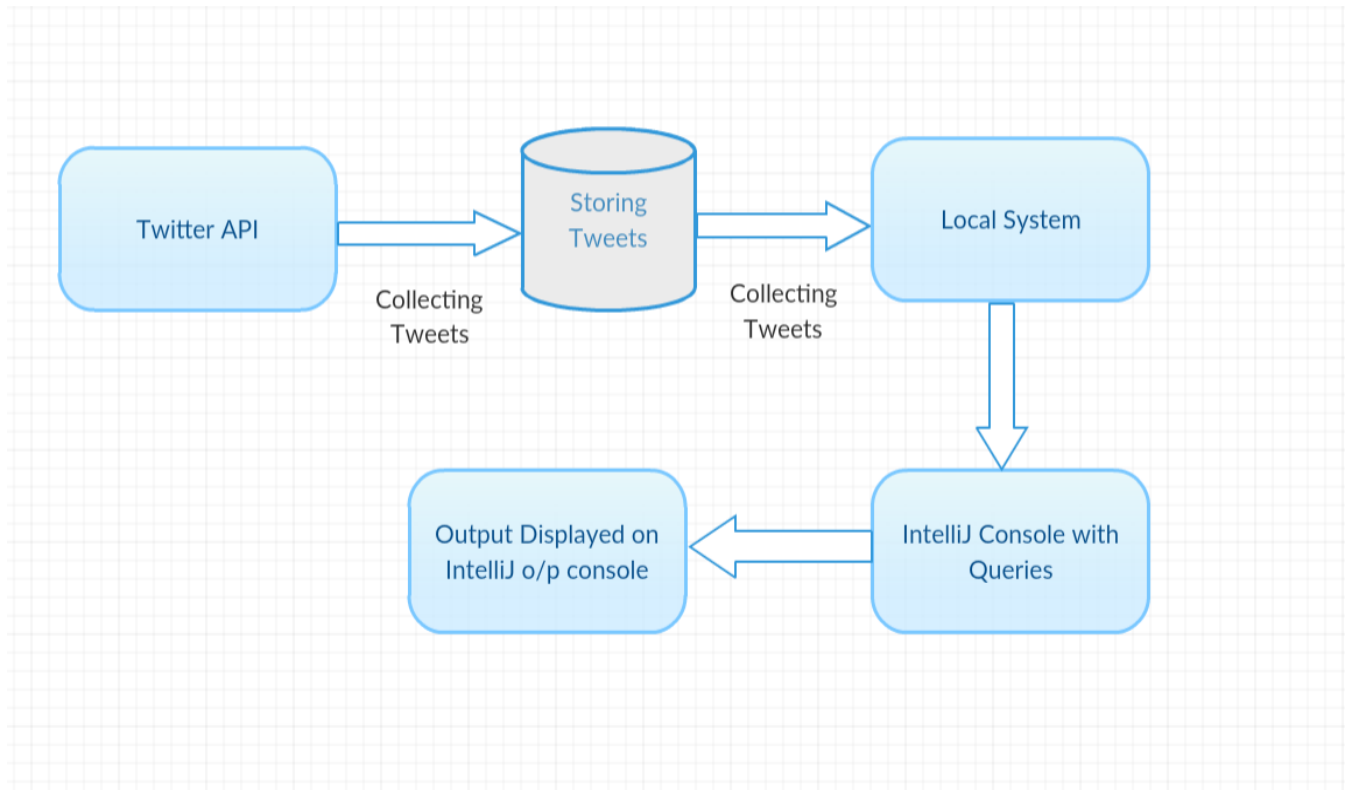
Here we used Spark to processing the twitter data. Because it has many advantages like

- 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. Software Architecture & Design

Twitter tweets are extracted using the API by running python program using the twitter tokens. The collected tweets are stored in JSON format which enables us to view the twitter data as table format using Notepad++ plug-ins. On the twitter JSON data we run the Spark RDD and Data frames analytical queries from IntelliJ using Scala plug-ins. These queries will give us the output in the IntelliJ output console.

Below diagram shows us the abstract architecture design and UML diagrams.



The Visio file is attached here



3. System Requirements:

Software Requirements:

- Apache Spark V 1.6
- Apache Hadoop V 2.6
- JDK 1.8
- Scala V 2.11
- IntelliJ Idea Community V 16

Programming Languages:

- Scala
- Python
- SQL

4. Analytical Queries:

Data Frames & RDD's:

Query1: This query gives us the number of retweet counts of the users, by which we can deduce that most of the people followed certain user's tweet. So, this query gives the result of the most popular person followed based on the retweet count.

```
val textFile = sc.textFile("C:\\Users\\malin\\Downloads\\Sports_tweets.json")

def time[A](f: => A) = {
  val s = System.nanoTime
  val ret = f
  println("Execution time for analysing retweeted count is : " + (System.nanoTime
- s) / 1e9 + " sec")
  ret
}

time{
  // Retweet query
```

Principles of Big Data Management-Project phase2

```
val re_tweet_query = sqlContext.sql("select user.name as name,  
retweeted_status.retweet_count as cnt from testtweets where user.name is not NULL  
order by cnt desc limit 20")  
re_tweet_query.show(false)  
re_tweet_query.save("retweeted_queries","json")  
}
```

The output of the above query is shown as below and the output file is attached.



re_tweet_count.json

The screenshot displays the IntelliJ IDEA interface with a Scala script named `analysis.scala` being executed. The script performs a SQL query to retrieve the top 20 tweets by retweet count from a table named `testtweets`. The results are saved to a JSON file named `retweeted_queries`. The console output shows the execution progress and the final results of the query.

Run: Sports_analysis

16/11/11 16:05:45 INFO DAGScheduler: ResultStage 1 (show at analysis.scala:137) finished in 3.084 s

16/11/11 16:05:45 INFO DAGScheduler: Job 1 finished: show at analysis.scala:137, took 3.098372 s

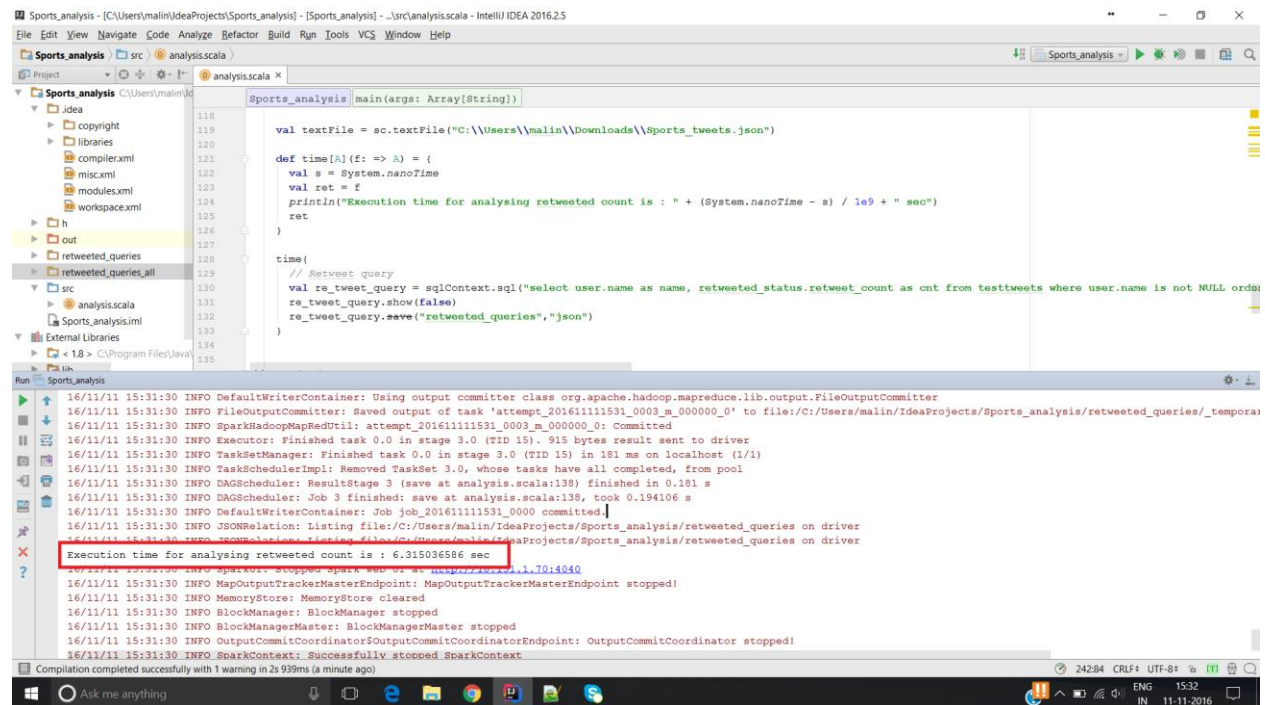
name	cnt
One Direction	81459
Ashok Bajjoan	30597
Ashok Bajjoan	26979
jake	22923
JJ Morgan	22922
イ	20245
宮上司	19872
TG	19871
ブンブン	19870
@kazu85	19869
Mammy	19868
しげ	19867
あゆ@ガルパンはいいぞ	19867
れいこ	19866
Kenken	19866
りよう	19866
Logan Iafrate	18444
Jen	18443
Ry	18442
Cassidy	18441

16/11/11 16:05:45 INFO MemoryStore: Block broadcast_6 stored as values in memory (estimated size 208.5 KB, free 700.8 KB)

Compilation completed successfully with 1 warning in 3s 295ms (a minute ago)

The run time of the above query is shown in the console of IntelliJ IDEA community.

Principles of Big Data Management-Project phase2



The screenshot shows an IDE window titled "Sports_analysis - [C:\Users\malin\IdeaProjects\Sports_analysis] - [Sports_analysis] - ...src\analysis.scala - IntelliJ IDEA 2016.2.5". The main editor displays a Scala file named "analysis.scala" with the following code:

```
118 Sports_analysis main(args: Array[String])
119
120 val textFile = sc.textFile("C:\Users\malin\Downloads\sports_tweets.json")
121
122 def time[A](f: => A) = {
123   val s = System.nanoTime
124   val ret = f
125   println("Execution time for analysing retweeted count is : " + (System.nanoTime - s) / 1e9 + " sec")
126   ret
127 }
128
129 time{
130   // Retweet query
131   val re_tweet_query = sqlContext.sql("select user.name as name, retweeted_status.retweet_count as cnt from testtweets where user.name is not NULL order by cnt desc")
132   re_tweet_query.show(false)
133   re_tweet_query.save("retweeted_queries", "json")
134 }
135
```

The Run window at the bottom shows the execution output:

```
16/11/11 15:31:30 INFO DefaultWriterContainer: Using output committer class org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
16/11/11 15:31:30 INFO FileOutputCommitter: Saved output of task 'attempt_201611111531_0003_m_000000_0' to file:/C:/Users/malin/IdeaProjects/Sports_analysis/retweeted_queries/_temporary_1
16/11/11 15:31:30 INFO SparkHadoopMapRedUtil: attempt_201611111531_0003_m_000000_0 Committed
16/11/11 15:31:30 INFO Executor: Finished task 0.0 in stage 3.0 (TID 15). 915 bytes result sent to driver
16/11/11 15:31:30 INFO TaskSetManager: Finished task 0.0 in stage 3.0 (TID 15) in 181 ms on localhost (1/1)
16/11/11 15:31:30 INFO TaskSchedulerImpl: Removed TaskSet 3.0, whose tasks have all completed, from pool
16/11/11 15:31:30 INFO DAGScheduler: ResultStage 3 (save at analysis.scala:138) finished in 0.181 s
16/11/11 15:31:30 INFO DAGScheduler: Job 3 finished: save at analysis.scala:138, took 0.194106 s
16/11/11 15:31:30 INFO DefaultWriterContainer: Job job_201611111531_0000 committed.
16/11/11 15:31:30 INFO JSONRelation: Listing file:/C:/Users/malin/IdeaProjects/Sports_analysis/retweeted_queries on driver
16/11/11 15:31:30 INFO JSONRelation: Listing file:/C:/Users/malin/IdeaProjects/Sports_analysis/retweeted_queries on driver
16/11/11 15:31:30 INFO SparkContext: Stopped Spark at 16/11/11 15:31:30
16/11/11 15:31:30 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
16/11/11 15:31:30 INFO MemoryStore: MemoryStore cleared
16/11/11 15:31:30 INFO BlockManager: BlockManager stopped
16/11/11 15:31:30 INFO BlockManagerMaster: BlockManagerMaster stopped
16/11/11 15:31:30 INFO OutputCommitCoordinatorOutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
16/11/11 15:31:30 INFO SparkContext: Successfully stopped SparkContext
Compilation completed successfully with 1 warning in 2s 939ms (a minute ago)
```

Query2: This RDD query will analyze, from which devices the tweets are tweeted by the users.

```
def time[A](f: => A) = {
  val s = System.nanoTime
  val ret = f
  println("Execution time for analysing which source user prefers is : " +
    (System.nanoTime - s) / 1e9 + " sec")
  ret
}

time{
  val android= (textFile.filter(line => line.contains("Twitter for
Android"))).count()
  val iphone= (textFile.filter(line => line.contains("Twitter for
iPhone"))).count()
  val ifttt= (textFile.filter(line => line.contains("IFTTT"))).count()
  val roundteam= (textFile.filter(line => line.contains("RoundTeam"))).count()
  val twitterfeed= (textFile.filter(line =>
line.contains("twitterfeed"))).count()
  val webClient= (textFile.filter(line => line.contains("Twitter Web
Client"))).count()
  val ipad= (textFile.filter(line => line.contains("Twitter for
iPad"))).count()
  val instagram= (textFile.filter(line => line.contains("Instagram"))).count()
  val tweetdeck= (textFile.filter(line => line.contains("TweetDeck"))).count()
  val hootsuite= (textFile.filter(line => line.contains("Hootsuite"))).count()
  val facebook= (textFile.filter(line => line.contains("Facebook"))).count()
  val paper= (textFile.filter(line => line.contains("Paper.li"))).count()
  val vine= (textFile.filter(line => line.contains("Vine - Make a
Scene"))).count()
}
```

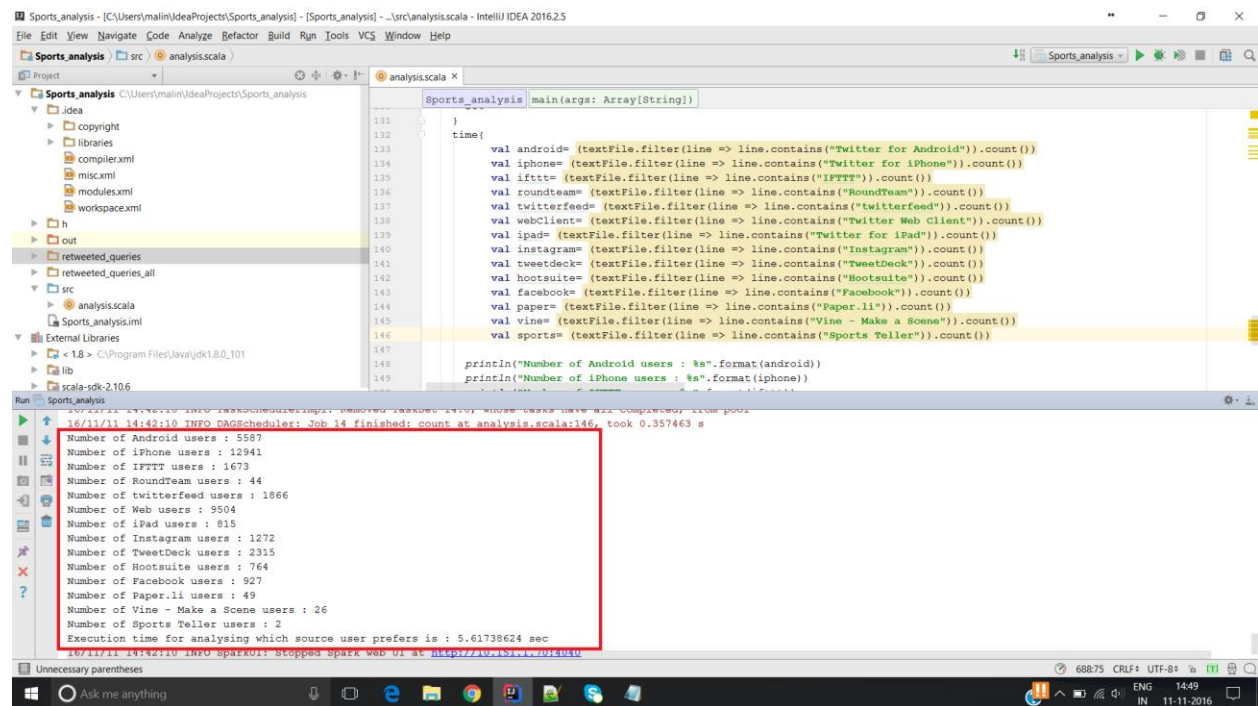
Principles of Big Data Management-Project phase2

```
val sports= (textFile.filter(line => line.contains("Sports
Teller"))).count())

println(("Number of Android users : %s\n Number of iPhone users : %s \n " +
"Number of IFTTT users : %s \n Number of RoundTeam users : %s \n " +
"Number of twitterfeed users : %s \n Number of Web users : %s\n Number of iPad
users : %s \n " +
"Number of Instagram users : %s \n Number of TweetDeck users : %s \n Number of
Hootsuite users : %s \n " +
"Number of Facebook users : %s \n Number of Paper.li users : %s \n Number of
Vine - Make a Scene users : %s \n " +
"Number of Sports Teller users : %s
\n").format(android,iphone,ifttt,roundteam,twitterfeed,webClient,ipad,instagram,tw
eetdeck,hootsuite,facebook,paper,vine,sports))

}
```

The output for the RDD and run time is shown in the below screen shot:



```
131 }
132 time{
133   val android= (textFile.filter(line => line.contains("Twitter for Android")).count())
134   val iphone= (textFile.filter(line => line.contains("Twitter for iPhone")).count())
135   val ifttt= (textFile.filter(line => line.contains("IFTTT")).count())
136   val roundteam= (textFile.filter(line => line.contains("RoundTeam")).count())
137   val twitterfeed= (textFile.filter(line => line.contains("twitterfeed")).count())
138   val webClient= (textFile.filter(line => line.contains("Twitter Web Client")).count())
139   val ipad= (textFile.filter(line => line.contains("Twitter for iPad")).count())
140   val instagram= (textFile.filter(line => line.contains("Instagram")).count())
141   val tweetdeck= (textFile.filter(line => line.contains("TweetDeck")).count())
142   val hootsuite= (textFile.filter(line => line.contains("Hootsuite")).count())
143   val facebook= (textFile.filter(line => line.contains("Facebook")).count())
144   val paper= (textFile.filter(line => line.contains("Paper.li")).count())
145   val vine= (textFile.filter(line => line.contains("Vine - Make a Scene")).count())
146   val sports= (textFile.filter(line => line.contains("Sports Teller")).count())
147
148   println("Number of Android users : %s".format(android))
149   println("Number of iPhone users : %s".format(iphone))
150 }
151
152 println("Number of Android users : %s\n Number of iPhone users : %s \n " +
153 "Number of IFTTT users : %s \n Number of RoundTeam users : %s \n " +
154 "Number of twitterfeed users : %s \n Number of Web users : %s\n Number of iPad
155 users : %s \n " +
156 "Number of Instagram users : %s \n Number of TweetDeck users : %s \n Number of
157 Hootsuite users : %s \n " +
158 "Number of Facebook users : %s \n Number of Paper.li users : %s \n Number of
159 Vine - Make a Scene users : %s \n " +
160 "Number of Sports Teller users : %s
161 \n").format(android,iphone,ifttt,roundteam,twitterfeed,webClient,ipad,instagram,tw
162 eetdeck,hootsuite,facebook,paper,vine,sports)
163 }
```

Run: 16/11/11 14:42:10 INFO DAGScheduler: Job 14 finished: count at analysis.scala:146, took 0.357463 s

```
Number of Android users : 5587
Number of iPhone users : 12941
Number of IFTTT users : 1673
Number of RoundTeam users : 44
Number of twitterfeed users : 1866
Number of Web users : 9504
Number of iPad users : 815
Number of Instagram users : 1272
Number of TweetDeck users : 2315
Number of Hootsuite users : 764
Number of Facebook users : 927
Number of Paper.li users : 49
Number of Vine - Make a Scene users : 26
Number of Sports Teller users : 2
Execution time for analysing which source user prefers is : 5.61738624 sec
16/11/11 14:42:10 INFO SparkUI: Stopped Spark web UI at http://114.49.22.201:4040/
```

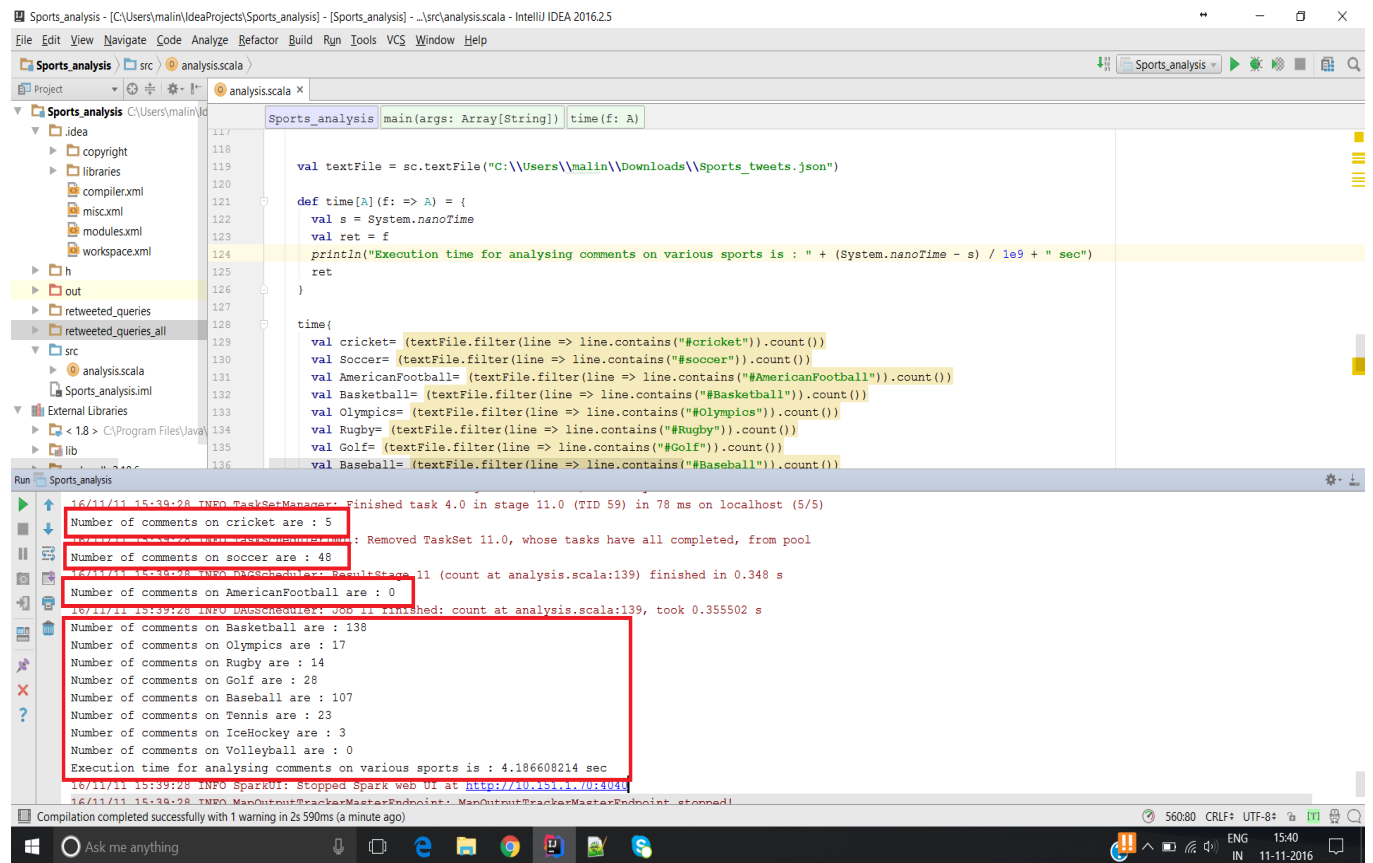
Query 3: This query will give the most popular Hashtag sports topic discussed in the twitter from the source we extracted.

```
def time[A](f: => A) = {
  val s = System.nanoTime
  val ret = f
  println("Execution time for analysing comments on various sports is : " +
(System.nanoTime - s) / 1e9 + " sec")
  ret
}
```

Principles of Big Data Management-Project phase2

```
time{
  val cricket= (textFile.filter(line => line.contains("#cricket")).count())
  val Soccer= (textFile.filter(line => line.contains("#soccer")).count())
  val AmericanFootball= (textFile.filter(line =>
line.contains("#AmericanFootball")).count())
  val Basketball= (textFile.filter(line => line.contains("#Basketball")).count())
  val Olympics= (textFile.filter(line => line.contains("#Olympics")).count())
  val Rugby= (textFile.filter(line => line.contains("#Rugby")).count())
  val Golf= (textFile.filter(line => line.contains("#Golf")).count())
  val Baseball= (textFile.filter(line => line.contains("#Baseball")).count())
  val Tennis= (textFile.filter(line => line.contains("#Tennis")).count())
  val IceHockey= (textFile.filter(line => line.contains("#IceHockey")).count())
  val Volleyball= (textFile.filter(line => line.contains("#Volleyball")).count())

  println("Number of comments on cricket are : %s \n Number of comments on soccer
are : %s \n " +
  "Number of comments on AmericanFootball are : %s \n Number of comments on
Basketball are : %s \n " +
  "Number of comments on Olympics are : %s \n Number of comments on Rugby are : %s
\n Number of comments on Golf are : %s \n " +
  "Number of comments on Baseball are : %s \n Number of comments on Tennis are :
%s \n Number of comments on IceHockey are : %s \n " +
  "Number of comments on Volleyball are :
%s").format(cricket,Soccer,AmericanFootball,Basketball,Olympics,Rugby,Golf,Basebal
l,Tennis,IceHockey,Volleyball)) }
```



The screenshot shows the IntelliJ IDEA interface with a Scala file named `analysis.scala` open. The code defines a `time` function that filters a text file for comments on various sports and counts them. The output is printed to the console.

```
val textFile = sc.textFile("C:\\Users\\malin\\Downloads\\Sports_tweets.json")

def time[A](f: => A) = {
  val s = System.nanoTime
  val ret = f
  println("Execution time for analysing comments on various sports is : " + (System.nanoTime - s) / 1e9 + " sec")
  ret
}

time{
  val cricket= (textFile.filter(line => line.contains("#cricket")).count())
  val Soccer= (textFile.filter(line => line.contains("#soccer")).count())
  val AmericanFootball= (textFile.filter(line => line.contains("#AmericanFootball")).count())
  val Basketball= (textFile.filter(line => line.contains("#Basketball")).count())
  val Olympics= (textFile.filter(line => line.contains("#Olympics")).count())
  val Rugby= (textFile.filter(line => line.contains("#Rugby")).count())
  val Golf= (textFile.filter(line => line.contains("#Golf")).count())
  val Baseball= (textFile.filter(line => line.contains("#Baseball")).count())
}
```

The console output shows the execution time and the count for each sport:

```
16/11/11 15:39:28 INFO TaskSetManager: Finished task 4.0 in stage 11.0 (TID 59) in 78 ms on localhost (5/5)
Number of comments on cricket are : 5
Number of comments on soccer are : 48
Number of comments on AmericanFootball are : 0
Number of comments on Basketball are : 138
Number of comments on Olympics are : 17
Number of comments on Rugby are : 14
Number of comments on Golf are : 28
Number of comments on Baseball are : 107
Number of comments on Tennis are : 23
Number of comments on IceHockey are : 3
Number of comments on Volleyball are : 0
Execution time for analysing comments on various sports is : 4.186608214 sec
16/11/11 15:39:28 INFO SparkUI: Stopped Spark web UI at http://10.154.1.70:4040
16/11/11 15:39:28 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
```


Query 4: This query retrieves the common hashtag topics when compared with hashtag topics given in Blackboard.

```
val analysis = sqlContext.jsonFile("C:\\Users\\putha\\Desktop\\sujitha\\prin of big
data\\project\\blackboard_hashtags.json")
analysis.registerTempTable("hashtags")
analysis.withColumnRenamed("test", "_corrupt_record")

// Renaming the column name with tag
val df = analysis.toDF().withColumnRenamed("_corrupt_record", "tag")
df.registerTempTable("hash1")

//selecting the hashtags presnt in hash1 and counts
val t = sqlContext.sql("select hash1.tag,count(testtweets.text) as count from
testtweets join hash1 on testtweets.text like concat ('%',hash1.tag,'%') group by
hash1.tag order by count desc limit 10")
t.show(false)
```

The screenshot shows the Spark IDE interface. The top pane contains the Scala code for Query 4. The bottom pane displays the execution logs, which include messages from the Executor, TaskSetManager, TaskSchedulerImpl, DAGScheduler, and SparkUI. The output of the query is shown as a table with two columns: 'tag' and 'count'.

tag	count
Yahoo	1157
Athletic	227
Dolphins	165
Texans	137
Curt Schilling	121
Steelers	119
Warriors	115
Thomas	109
Chargers	99
Patriots	95

The bottom pane also shows the SparkUI URL: <http://10.151.3.187:4041> and various status messages from the Spark ecosystem.

Query 5: This query uses twitter API, which takes userid as input and retrieve the first tweet from our extracted data with the tag that contains “soccer”.

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

    val consumer = new CommonsHttpOAuthConsumer(consumerKey, consumerSecret)
```

Principles of Big Data Management-Project phase2

```
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\\malin\\Downloads\\Sports_tweets.json")
    tweet.registerTempTable("testtweets")
    val s8 = sqlContext.sql("select user.name as name,user.id from testtweets
where text like '%soccer%'")
    s8.show(false)
    val userid = s8.first().getLong(1)
    println("Selected user id is:"+userid)

    val request = new
HttpGet(https://api.twitter.com/1.1/users/show.json?user\_id=+userid)
    consumer.sign(request)
    val client = new DefaultHttpClient()
    val response = client.execute(request)

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

Principles of Big Data Management-Project phase2

The screenshot displays the IntelliJ IDEA 2016.2.5 IDE with a project named 'Sports_analysis'. The main editor shows a Scala file 'Api.scala' with the following code:

```
main(args: Array[String]) {
  val consumer = new CommonsHttpURLConnection(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\\malin\\Downloads\\Sports_tweets.json")
  tweet.registerTempTable("testtweets")
  val s8 = sqlContext.sql("select user.name as name,user.id from testtweets where text like '%soccer%'")
  s8.show(false)
  val userid = s8.first().getLong(1)
  println("Selected user id is:" + userid)
}
```

The Run console at the bottom shows the execution output:

```
16/11/11 23:01:40 INFO Executor: Finished task 0.0 in stage 2.0 (TID 6). 2717 bytes result sent to driver
16/11/11 23:01:40 INFO TaskSetManager: Finished task 0.0 in stage 2.0 (TID 6) in 16 ms on localhost (1/1)
16/11/11 23:01:40 INFO TaskSchedulerImpl: Removed TaskSet 2.0, whose tasks have all completed, from pool
16/11/11 23:01:40 INFO DAGScheduler: ResultStage 2 (first at Api.scala:36) finished in 0.016 s
16/11/11 23:01:40 INFO DAGScheduler: Job 2 finished: first at Api.scala:36, took 0.038446 s
Selected user id is:2153103163
200
16/11/11 23:01:41 INFO SparkContext: Invoking stop() from shutdown hook
{"id":2153103163,"id_str":"2153103163","name":"Sports On Notice","screen_name":"SportsOnNotice","location":"Texas, USA","profile_location":{"id":"e0060cda70f5f341","url":"https://a:
16/11/11 23:01:41 INFO sparkUI: Stopped spark web UI at http://localhost:4040
16/11/11 23:01:41 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
16/11/11 23:01:41 INFO MemoryStore: MemoryStore cleared
16/11/11 23:01:41 INFO BlockManager: BlockManager stopped
16/11/11 23:01:41 INFO BlockManagerMaster: BlockManagerMaster stopped
16/11/11 23:01:41 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
16/11/11 23:01:41 INFO SparkContext: Successfully stopped SparkContext
16/11/11 23:01:41 INFO ShutdownHookManager: Shutdown hook called
16/11/11 23:01:41 INFO ShutdownHookManager: Deleting directory C:\Users\malin\AppData\Local\Temp\spark-99029f0c-39c7-4edd-adde-ae1b0ad89a5a

Compilation completed successfully with 1 warning in 3s 229ms (a minute ago)
```

The status bar at the bottom indicates the system time as 18:39, CRLF, UTF-8, and the date as 11-11-2016.