Principles of Big Data Management Phase-2

Submitted by

Gopi Chand Bodepudi (16278459)
VasuDeva Madala (16280568)
Sravan Kumar Pagadala (16280777)

1. Introduction

1.1 About Twitter

Twitter is a massive social networking service that enables users to send data read short 140 character messages called 'tweets'. More than 326 million active users publish over 500 million tweets every day. Twitter was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone and Noah Glass and launched in July 2006. Twitters speed and ease of publication have made it an important communication medium for people from all walks of life. This stream of messages from a variety of users contains information on an array of topics, including conventional new stories, events of local interest, opinions, real-time events. Twitter APIs provide access to tweets from a time range, from a user, with a keyword.

1.2 Proposed Project

We choose 'Sports' as our topic to do big data analysis. Based on twitter tweets, we predicted some interesting analysis on Sports using thousands of tweets tweeted by different people. First, we collected the tweets from twitter API based on some key words related to Sports. After that, we analysed the data that we have collected. By using the analysis, we written some interesting SQL queries useful to give a proper result for the analysis.

Here we used Spark to process 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. System Requirements

2.1 Software Requirements

Python 3.7
Scala 2.4.0
Apache spark

2.2 Language Requirements

Python, Scala, sql

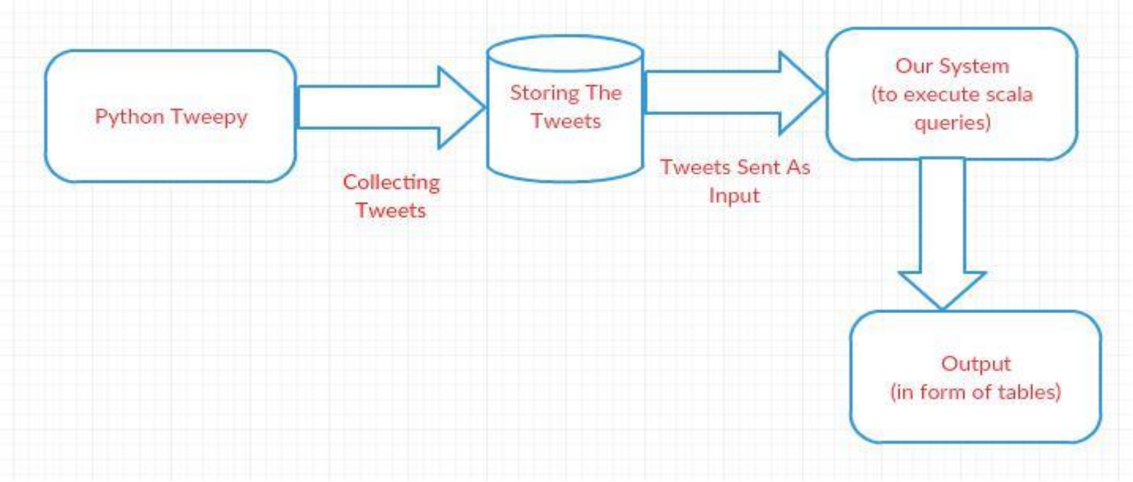
3. About the Theme

We have selected the theme sports, which provides the analysis on the twitter tweets based on the queries. Initially tweets were collected using hashtags related to sport types such as Cricket, football, Tennis etc. Using this data, the queries are written in Scala which gives analysis on which sport more tweets were done, which user made more number of tweets on which sport, which state people in USA are more involved in tweeting about the sports, top languages using which tweets were posted. The extracted data is displayed in tables. Based on this data visualization is done.

4. System Architecture

First, we generated credential for accessing twitter. By using these credentials, we wrote a python program to collect twitter tweets based on keywords related to sports. Tweets were stored in a text file in a JSON format. We will give these JSON file to SQL queries for analysis with Spark, Intellij with Scala program with queries.

System Architecture



5. Collecting Twitter Tweets

5.1 Generating Access Tokens

First, we generated keys for accessing twitter API. For this we need to register our application by using, http://apps.twitter.com. We generated access token, access token secret, consumer key, consumer secret

access_token = "2219941182-hJEd5re1y7lbZmVlyZySZvVsJf88fP6um3SsC3r"
access_token_secret = "BntHym97rzCisKS3BFXqrBgQbgokklZEBcqHXixGJQtX8"
consumer_key = "187ztf3hxmT3Nm3YonFzcAvEB"
consumer_secret = "hTqPaSjNXw21GXmPCey6CZBCZRoO1EbTkbVO4zMv77kN8Ikq0P"

5.2 Streaming Twitter Tweets

After that we have written a python program for streaming twitter tweets. As our theme is related to 'Sports' we used few hashtags such as cricket, football, tennis, rugby etc. From twitter, we have streamed almost 100000 instances. Tweets were stored in JSON (JavaScript Object Notation) format in a file.

Tweets Source code : https://github.com/gbhmh/Principles-of-BigData-Phase2/tree/master/Source/python%20program

6. Analyzing Twitter Data

Initially the twitter tweets json file is read in Scala and stored in temporary table.

```
spark-shell.cmd

val sqlContext = new org.apache.spark.sql.SQLContext(sc);

import sqlContext.implicits._
```

```
val tweetstable = sqlContext.read.json("C:\\Users\\gopichand\\Desktop\\pb\\phase2\\twitdb1.json");

tweetstable.registerTempTable("tweetDatatable");
```

Using this temporary tweets data is analysed by writing queries in Scala.

Link for Sql queries and their outputs in Sparksql:

https://github.com/gbhmh/Principles-of-BigData-Phase2/tree/master/Source/queries%20outputs%20screenshots

Link for Visualisations output screenshots:

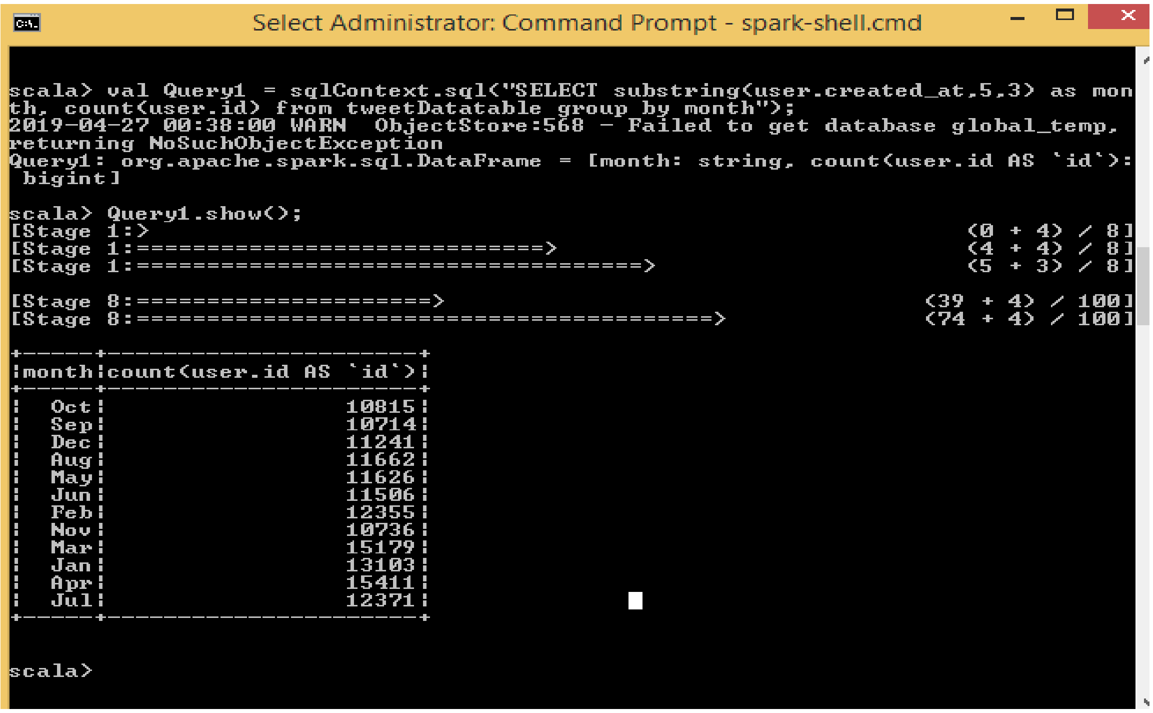
https://github.com/gbhmh/Principles-of-BigData-Phase2/tree/master/Source/queries%20visualisations

Link for html files(using JSamcharts) used for doing visualisation:

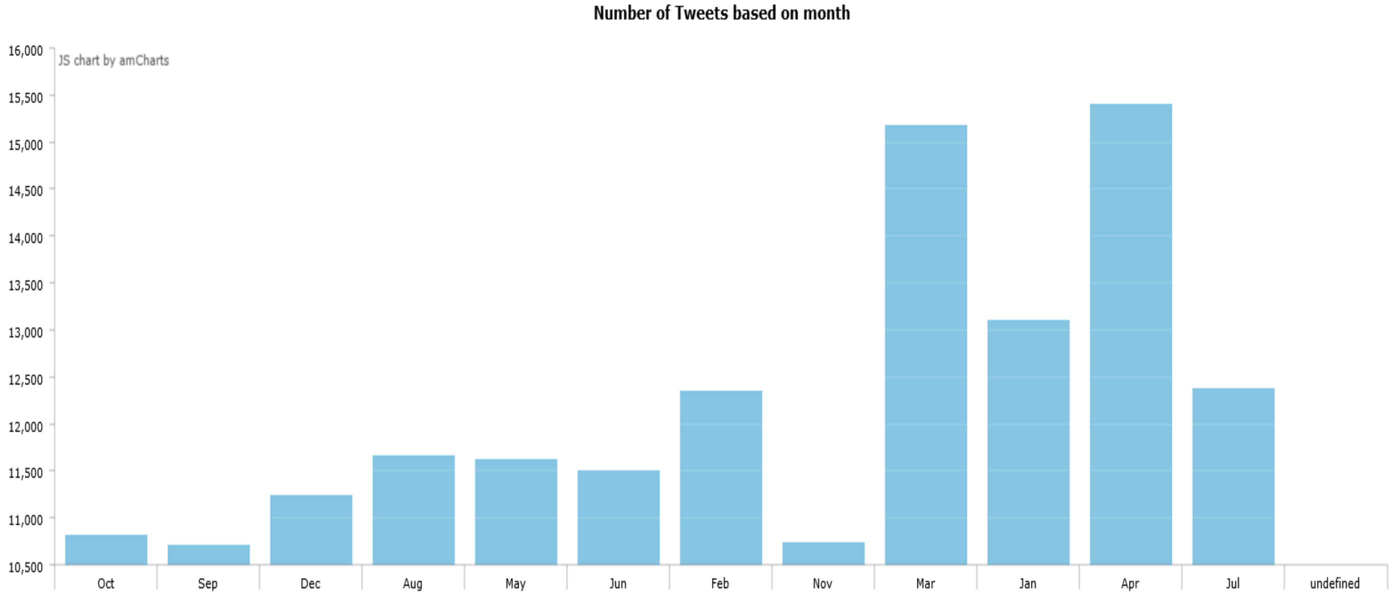
https://github.com/gbhmh/Principles-of-BigData-Phase2/tree/master/Source/visualisation%20files

Please find the queries, outputs and their visualisations below. We have used JS charts by Armchats for doing the visualisation. We have done analysis on the data by ***writing 17 queries***.

Query 1: Number of tweets based on month

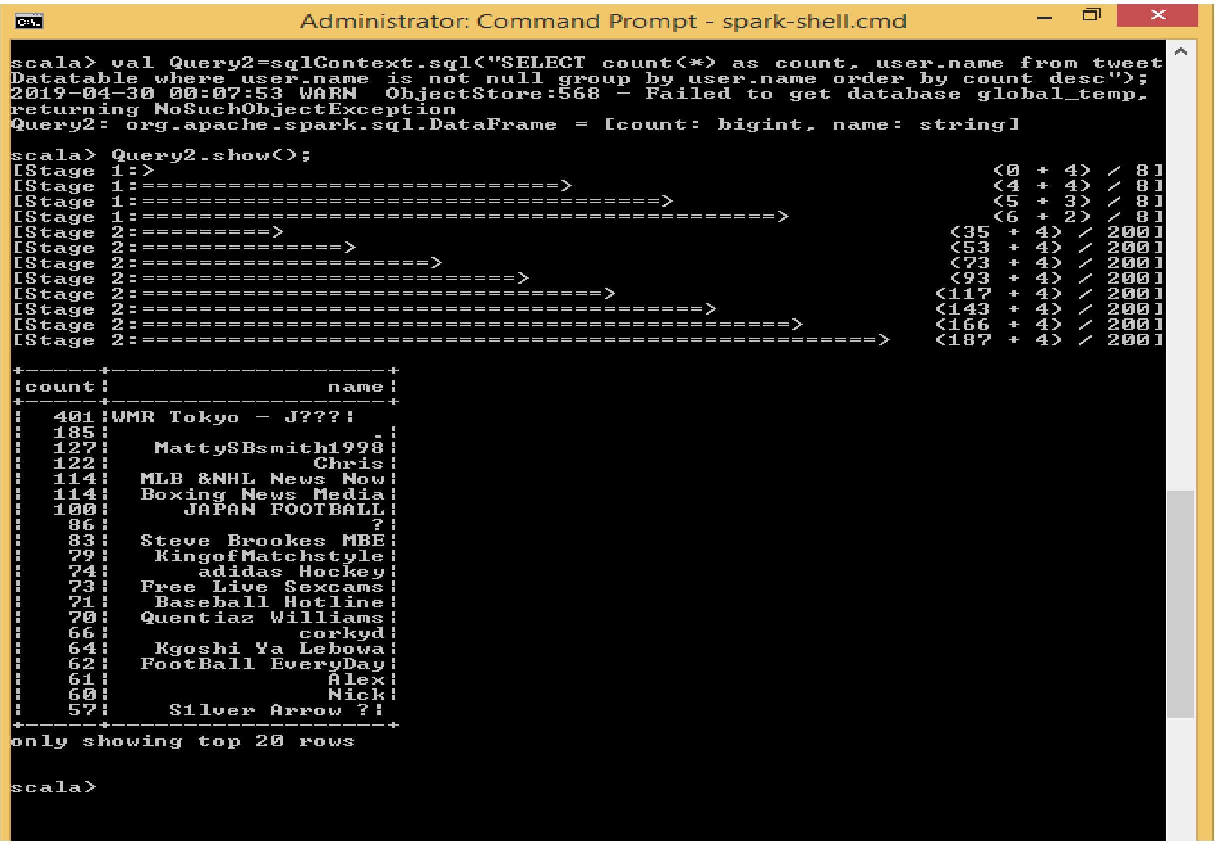


```
scala> val Query1 = sqlContext.sql("SELECT substring(user.created_at,5,3) as mon
th, count(user.id) from tweetDatatable group by month");
2019-04-27 00:38:00 WARN  ObjectStore:568 - Failed to get database global_temp,
returning NoSuchObjectException
Query1: org.apache.spark.sql.DataFrame = [month: string, count(user.id AS `id`):
 bigint]

scala> Query1.show();
[Stage 1:>                                                          (0 + 4) / 8]
[Stage 1:=============================>                             (4 + 4) / 8]
[Stage 1:====================================>                      (5 + 3) / 8]

[Stage 8:=====================>                                   (39 + 4) / 100]
[Stage 8:========================================>                (74 + 4) / 100]

+-----+---------------------+
¦month¦count(user.id AS `id`)¦
+-----+---------------------+
¦  Oct¦                10815¦
¦  Sep¦                10714¦
¦  Dec¦                11241¦
¦  Aug¦                11662¦
¦  May¦                11626¦
¦  Jun¦                11506¦
¦  Feb¦                12355¦
¦  Nov¦                10736¦
¦  Mar¦                15179¦
¦  Jan¦                13103¦
¦  Apr¦                15411¦
¦  Jul¦                12371¦
+-----+---------------------+


scala>
```

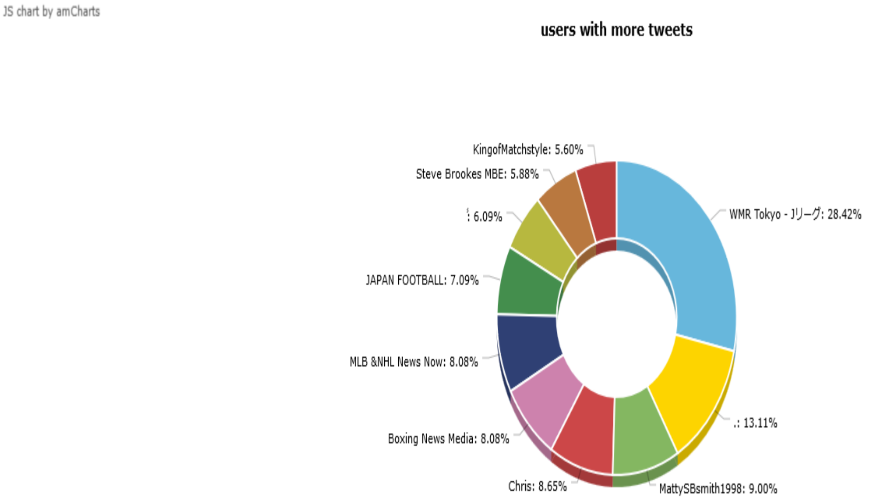


Number of Tweets based on month

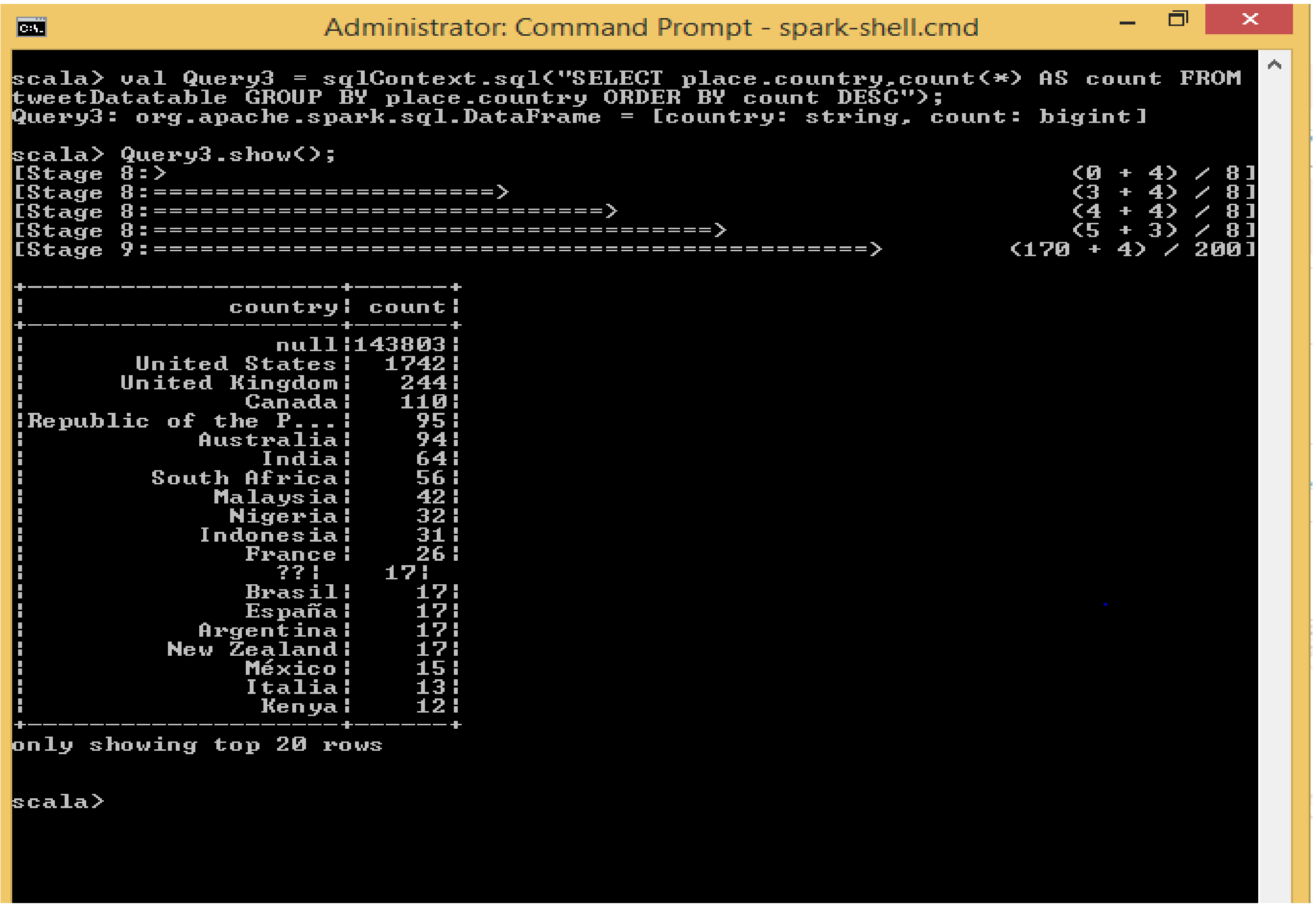
Query 2: Users with more tweets



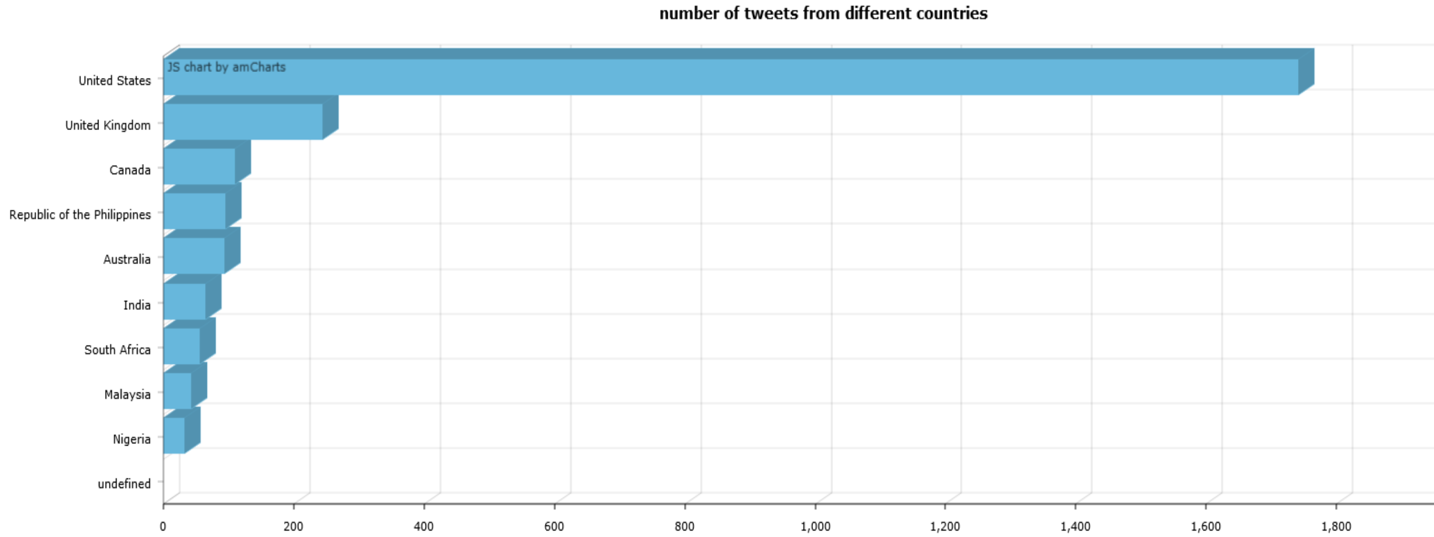
```
scala> val Query2=sqlContext.sql("SELECT count(*) as count, user.name from tweet
Datatable where user.name is not null group by user.name order by count desc");
2019-04-30 00:07:53 WARN  ObjectStore:568 - Failed to get database global_temp,
returning NoSuchObjectException
Query2: org.apache.spark.sql.DataFrame = [count: bigint, name: string]

scala> Query2.show();
[Stage 1:>                                                          (0 + 4) / 8]
[Stage 1:=============================>                             (4 + 4) / 8]
[Stage 1:====================================>                      (5 + 3) / 8]
[Stage 1:============================================>              (6 + 2) / 8]
[Stage 2:=========>                                              (35 + 4) / 200]
[Stage 2:==============>                                         (53 + 4) / 200]
[Stage 2:====================>                                   (73 + 4) / 200]
[Stage 2:==========================>                             (93 + 4) / 200]
[Stage 2:================================>                      (117 + 4) / 200]
[Stage 2:=======================================>               (143 + 4) / 200]
[Stage 2:=============================================>         (166 + 4) / 200]
[Stage 2:===================================================>   (187 + 4) / 200]

+-----+-------------------+
¦count¦               name¦
+-----+-------------------+
¦  401¦WMR Tokyo - J???¦
¦  185¦                  .¦
¦  127¦   MattySBsmith1998¦
¦  122¦              Chris¦
¦  114¦  MLB &NHL News Now¦
¦  114¦  Boxing News Media¦
¦  100¦      JAPAN FOOTBALL¦
¦   86¦                  ?¦
¦   83¦  Steve Brookes MBE¦
¦   79¦   KingofMatchstyle¦
¦   74¦      adidas Hockey¦
¦   73¦  Free Live Sexcams¦
¦   71¦   Baseball Hotline¦
¦   70¦  Quentiaz Williams¦
¦   66¦             corkyd¦
¦   64¦   Kgoshi Ya Lebowa¦
¦   62¦  FootBall EveryDay¦
¦   61¦               Alex¦
¦   60¦               Nick¦
¦   57¦     S1lver Arrow ?¦
+-----+-------------------+
only showing top 20 rows

scala>
```



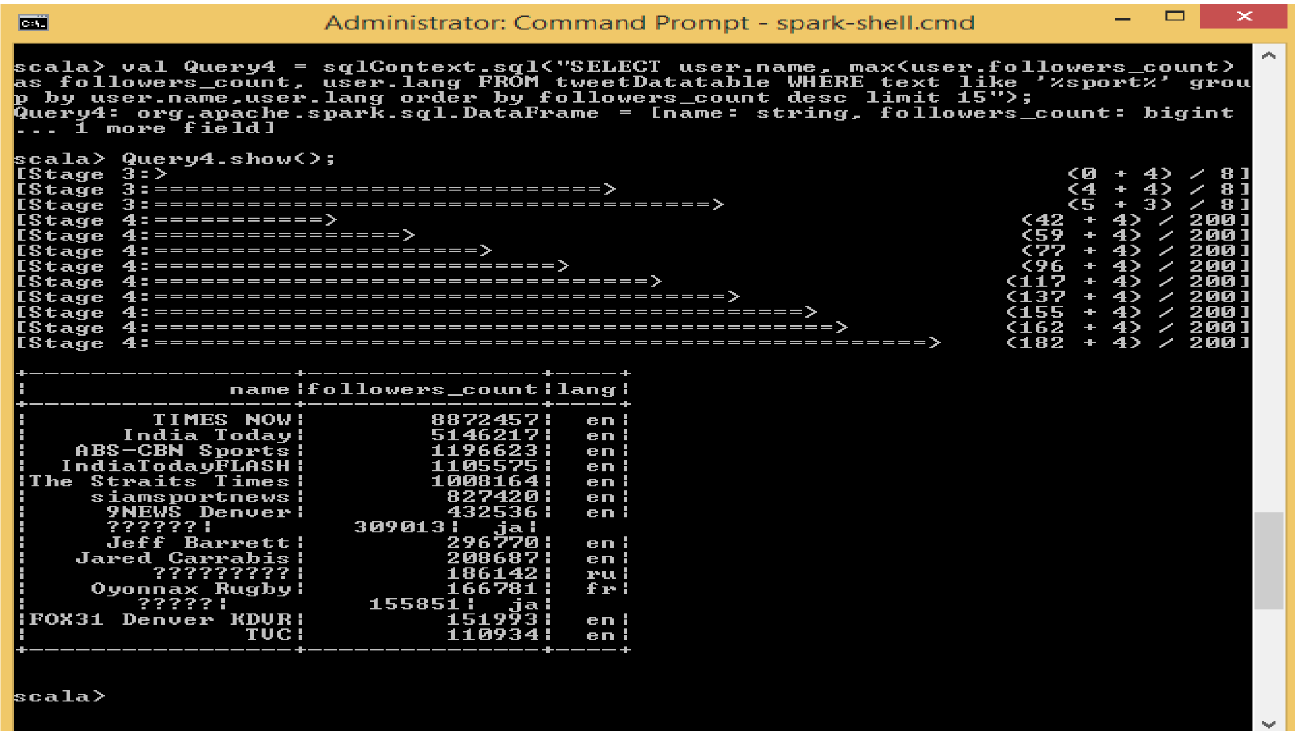
Query 3: Number of tweets from different countries



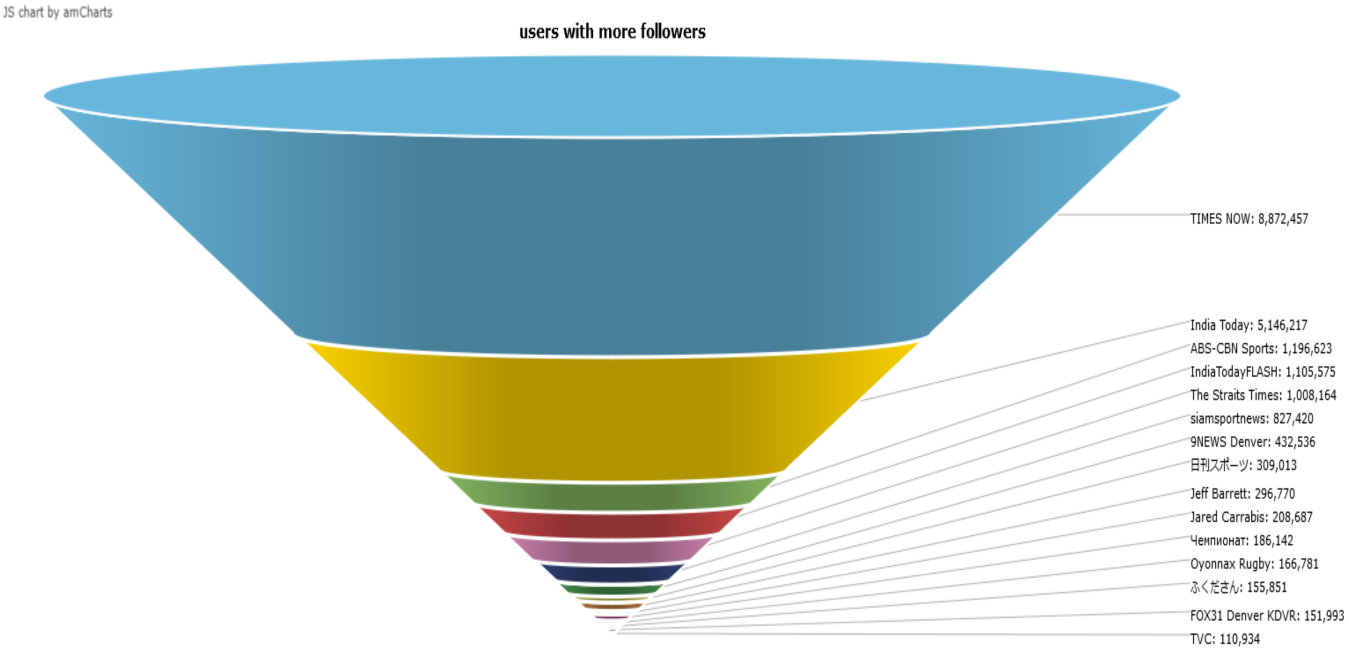
```
scala> val Query3 = sqlContext.sql("SELECT place.country,count(*) AS count FROM
tweetDatatable GROUP BY place.country ORDER BY count DESC");
Query3: org.apache.spark.sql.DataFrame = [country: string, count: bigint]

scala> Query3.show();
[Stage 8:>                                                          (0 + 4) / 8]
[Stage 8:======================>                                    (3 + 4) / 8]
[Stage 8:=============================>                             (4 + 4) / 8]
[Stage 8:====================================>                      (5 + 3) / 8]
[Stage 9:==============================================>        (170 + 4) / 200]

+--------------------+------+
|             country| count|
+--------------------+------+
|                null|143803|
|       United States|  1742|
|      United Kingdom|   244|
|              Canada|   110|
|Republic of the P...|    95|
|           Australia|    94|
|               India|    64|
|        South Africa|    56|
|            Malaysia|    42|
|             Nigeria|    32|
|           Indonesia|    31|
|              France|    26|
|                ??|    17|
|              Brasil|    17|
|              España|    17|
|           Argentina|    17|
|         New Zealand|    17|
|              México|    15|
|              Italia|    13|
|               Kenya|    12|
+--------------------+------+
only showing top 20 rows

scala>
```



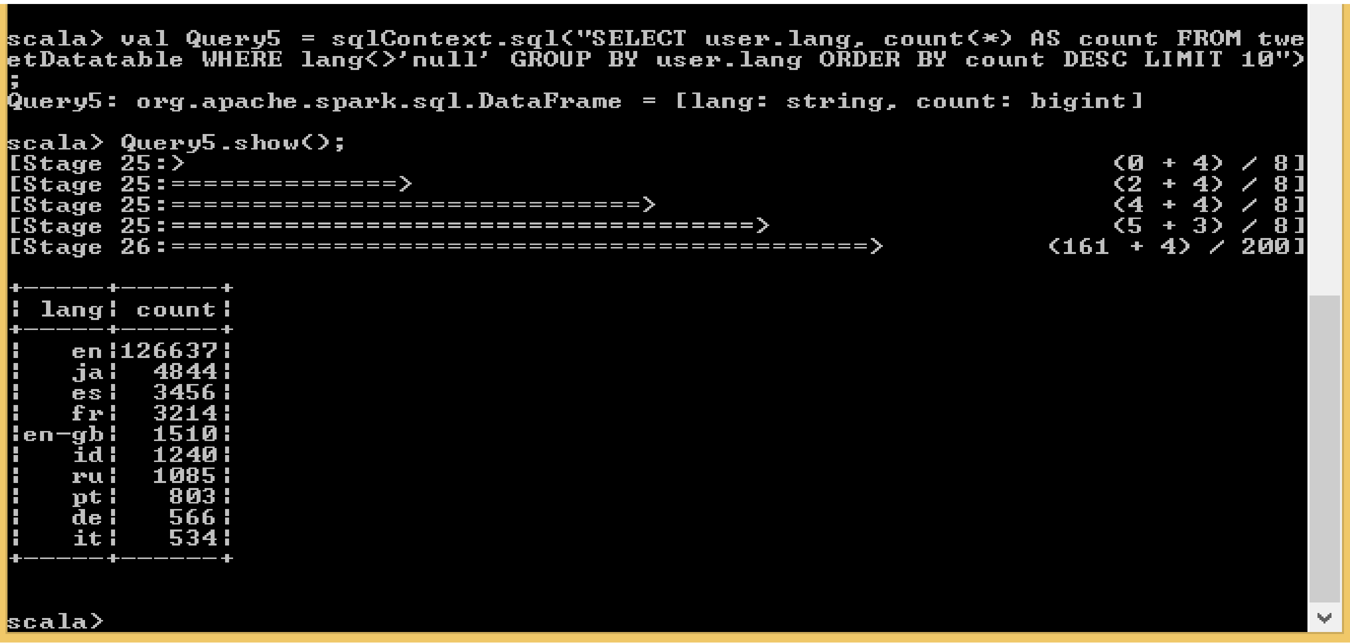
Query 4: Users with more followers



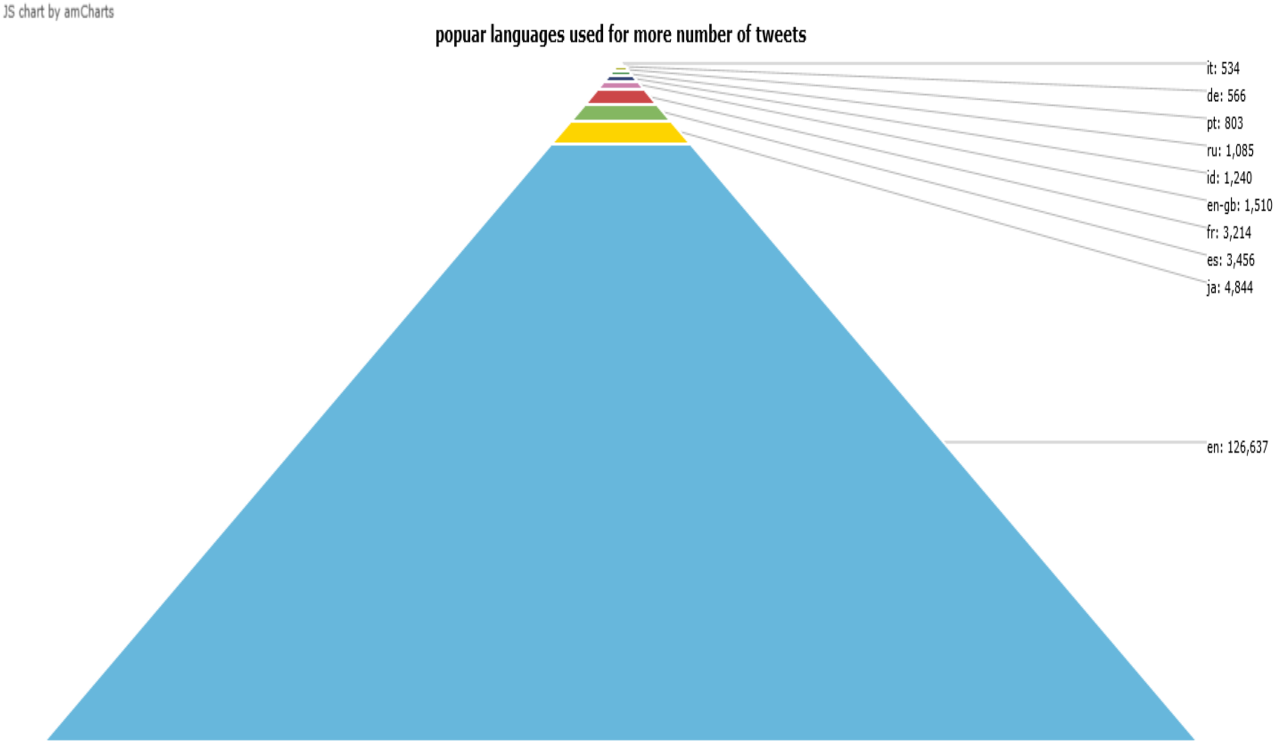
```
scala> val Query4 = sqlContext.sql("SELECT user.name, max(user.followers_count)
as followers_count, user.lang FROM tweetDatatable WHERE text like '%sport%' grou
p by user.name,user.lang order by followers_count desc limit 15");
Query4: org.apache.spark.sql.DataFrame = [name: string, followers_count: bigint
... 1 more field]

scala> Query4.show();
[Stage 3:>                                                          (0 + 4) / 8]
[Stage 3:============================>                              (4 + 4) / 8]
[Stage 3:===================================>                       (5 + 3) / 8]
[Stage 4:==========>                                            (42 + 4) / 200]
[Stage 4:===============>                                       (59 + 4) / 200]
[Stage 4:====================>                                  (77 + 4) / 200]
[Stage 4:=========================>                             (96 + 4) / 200]
[Stage 4:===============================>                     (117 + 4) / 200]
[Stage 4:====================================>                (137 + 4) / 200]
[Stage 4:=========================================>           (155 + 4) / 200]
[Stage 4:===========================================>         (162 + 4) / 200]
[Stage 4:=================================================>   (182 + 4) / 200]

+-----------------+---------------+----+
|             name|followers_count|lang|
+-----------------+---------------+----+
|        TIMES NOW|        8872457|  en|
|      India Today|        5146217|  en|
|   ABS-CBN Sports|        1196623|  en|
|  IndiaTodayFLASH|        1105575|  en|
|The Straits Times|        1008164|  en|
|    siamsportnews|         827420|  en|
|     9NEWS Denver|         432536|  en|
|     ???????|        309013|  ja|
|     Jeff Barrett|         296770|  en|
|   Jared Carrabis|         208687|  en|
|       ??????????|         186142|  ru|
|    Oyonnax Rugby|         166781|  fr|
|       ?????|         155851|  ja|
|FOX31 Denver KDVR|         151993|  en|
|              TVC|         110934|  en|
+-----------------+---------------+----+


scala>
```



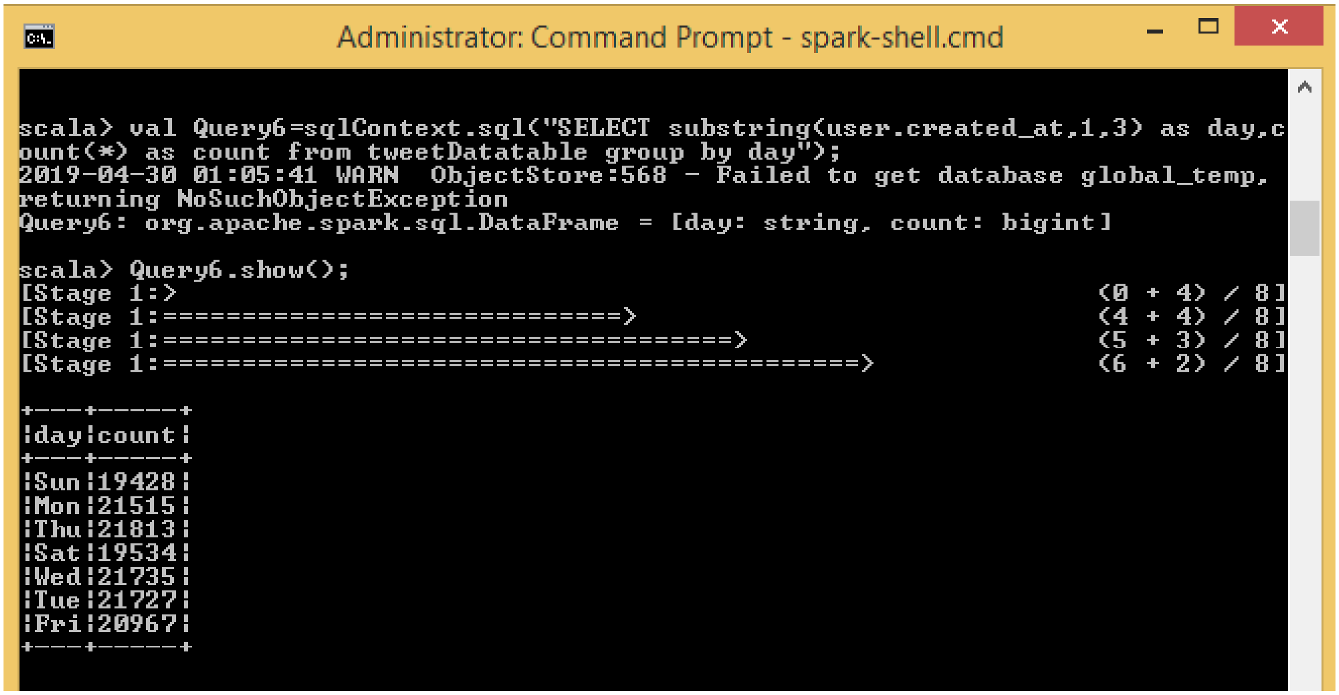
Query 5: Popular languages used for more number of tweets



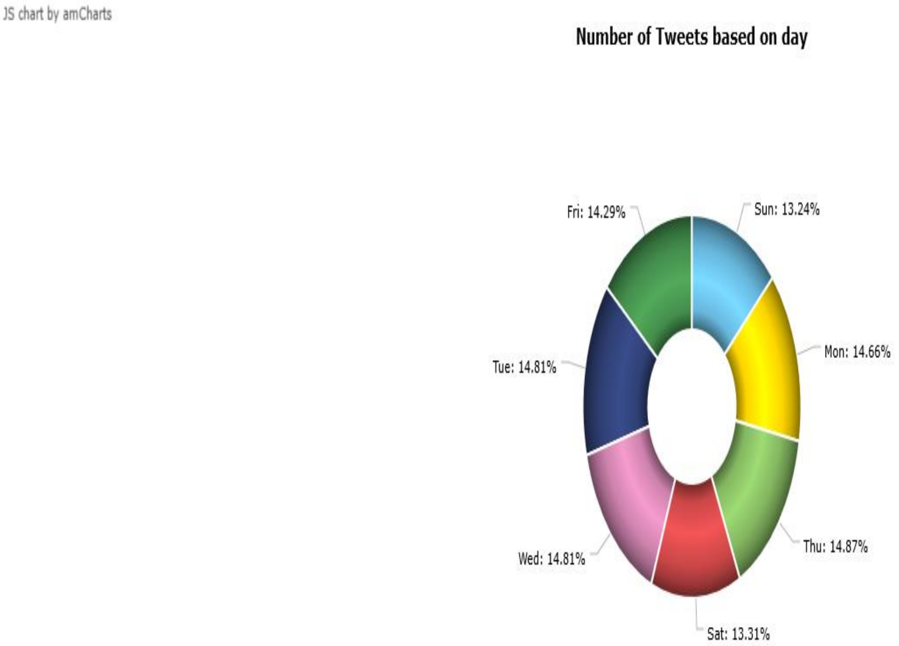
```
scala> val Query5 = sqlContext.sql("SELECT user.lang, count(*) AS count FROM twe
etDatatable WHERE lang<>’null’ GROUP BY user.lang ORDER BY count DESC LIMIT 10")
;
Query5: org.apache.spark.sql.DataFrame = [lang: string, count: bigint]

scala> Query5.show();
[Stage 25:>                                                         (0 + 4) / 8]
[Stage 25:==============>                                           (2 + 4) / 8]
[Stage 25:=============================>                            (4 + 4) / 8]
[Stage 25:====================================>                     (5 + 3) / 8]
[Stage 26:===========================================>          (161 + 4) / 200]

+-----+------+
| lang| count|
+-----+------+
|   en|126637|
|   ja|  4844|
|   es|  3456|
|   fr|  3214|
|en-gb|  1510|
|   id|  1240|
|   ru|  1085|
|   pt|   803|
|   de|   566|
|   it|   534|
+-----+------+


scala>
```



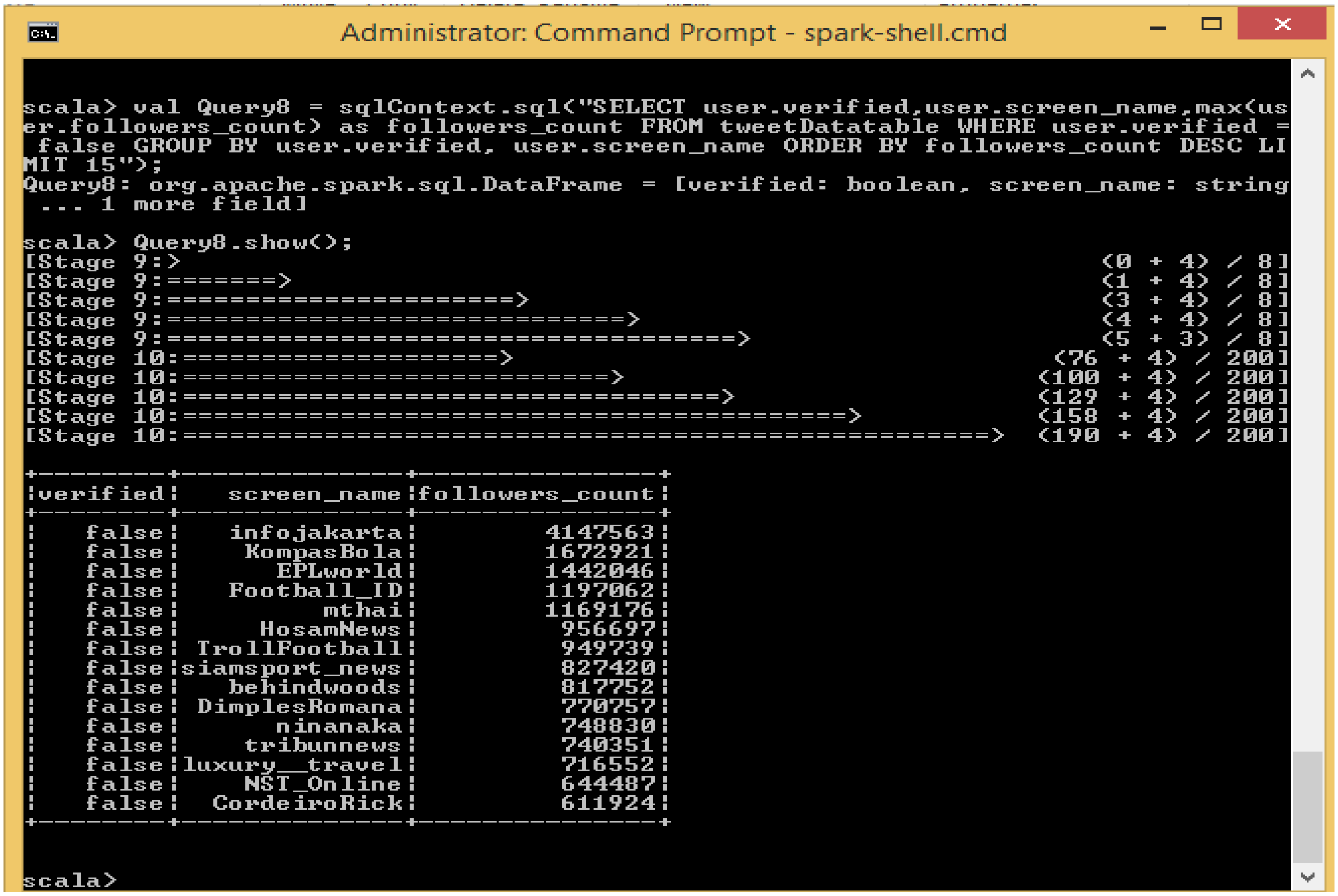
Query 6: Number of tweets based on day



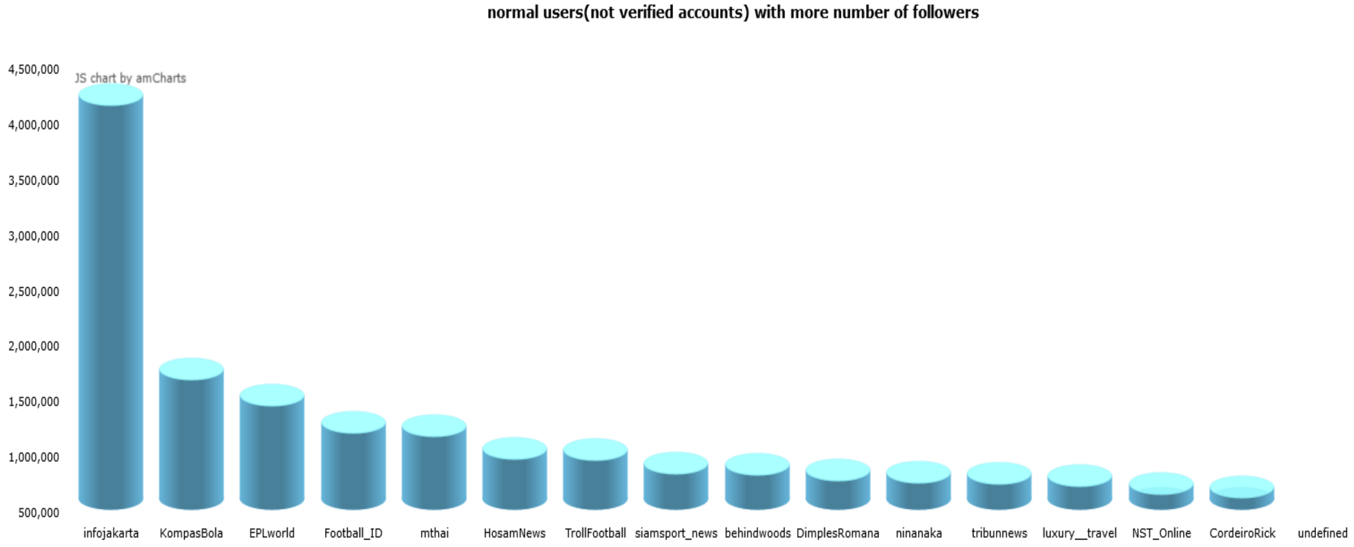
```
scala> val Query6=sqlContext.sql("SELECT substring(user.created_at,1,3) as day,c
ount(*) as count from tweetDatatable group by day");
2019-04-30 01:05:41 WARN  ObjectStore:568 - Failed to get database global_temp,
returning NoSuchObjectException
Query6: org.apache.spark.sql.DataFrame = [day: string, count: bigint]

scala> Query6.show();
[Stage 1:>                                                          (0 + 4) / 8]
[Stage 1:=============================>                              (4 + 4) / 8]
[Stage 1:====================================>                       (5 + 3) / 8]
[Stage 1:============================================>               (6 + 2) / 8]

+---+-----+
¦day¦count¦
+---+-----+
¦Sun¦19428¦
¦Mon¦21515¦
¦Thu¦21813¦
¦Sat¦19534¦
¦Wed¦21735¦
¦Tue¦21727¦
¦Fri¦20967¦
+---+-----+
```



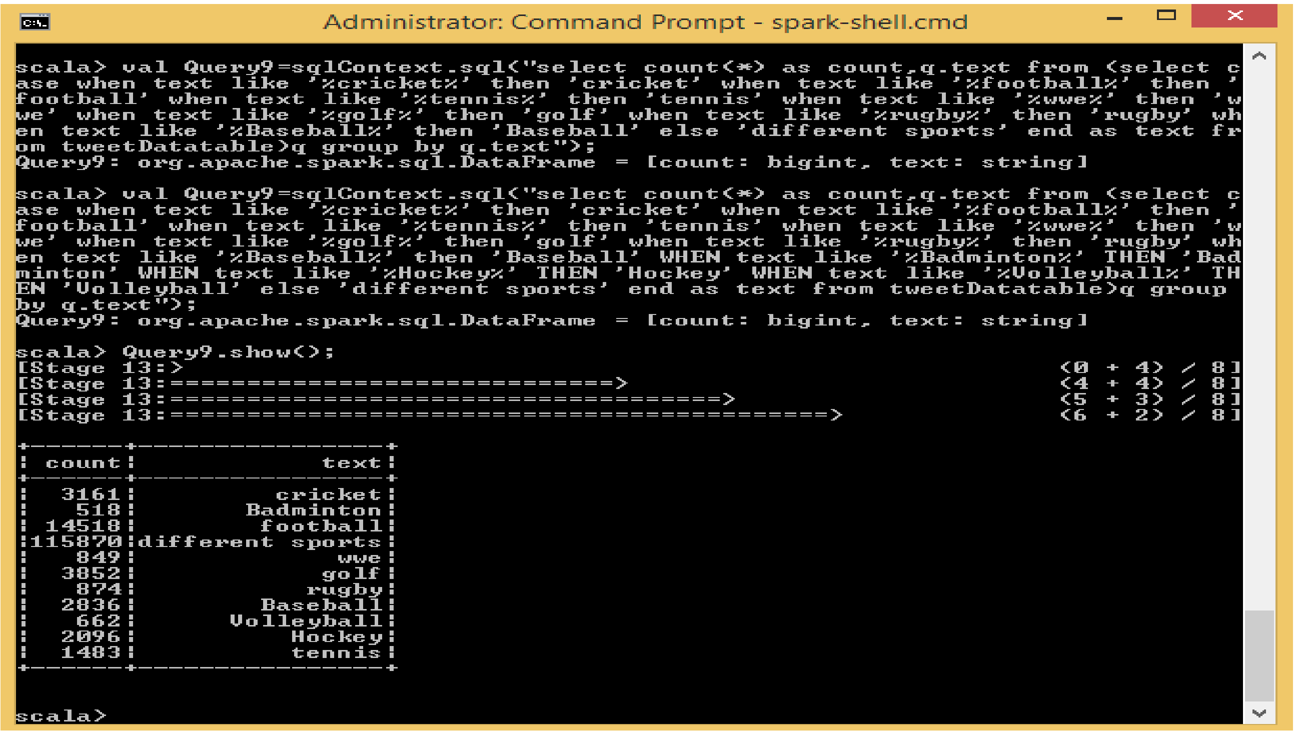
Query 8: Normal users(not verified accounts) with more number of followers



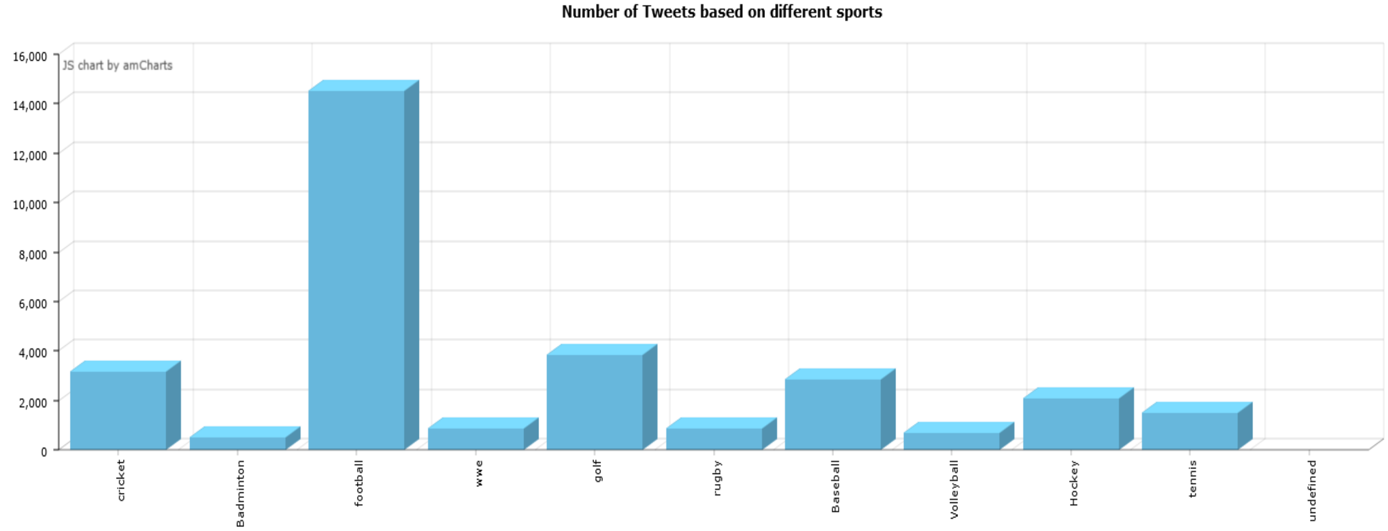
```
scala> val Query8 = sqlContext.sql("SELECT user.verified,user.screen_name,max(us
er.followers_count) as followers_count FROM tweetDatatable WHERE user.verified =
 false GROUP BY user.verified, user.screen_name ORDER BY followers_count DESC LI
MIT 15");
Query8: org.apache.spark.sql.DataFrame = [verified: boolean, screen_name: string
 ... 1 more field]

scala> Query8.show();
[Stage 9:>                                                          (0 + 4) / 8]
[Stage 9:=======>                                                   (1 + 4) / 8]
[Stage 9:======================>                                    (3 + 4) / 8]
[Stage 9:=============================>                             (4 + 4) / 8]
[Stage 9:====================================>                      (5 + 3) / 8]
[Stage 10:====================>                                  (76 + 4) / 200]
[Stage 10:===========================>                          (100 + 4) / 200]
[Stage 10:==================================>                   (129 + 4) / 200]
[Stage 10:==========================================>           (158 + 4) / 200]
[Stage 10:===================================================>  (190 + 4) / 200]

+--------+--------------+---------------+
|verified|   screen_name|followers_count|
+--------+--------------+---------------+
|   false|   infojakarta|        4147563|
|   false|    KompasBola|        1672921|
|   false|      EPLworld|        1442046|
|   false|   Football_ID|        1197062|
|   false|         mthai|        1169176|
|   false|     HosamNews|         956697|
|   false| TrollFootball|         949739|
|   false|siamsport_news|         827420|
|   false|   behindwoods|         817752|
|   false| DimplesRomana|         770757|
|   false|      ninanaka|         748830|
|   false|    tribunnews|         740351|
|   false|luxury__travel|         716552|
|   false|    NST_Online|         644487|
|   false|  CordeiroRick|         611924|
+--------+--------------+---------------+


scala>
```



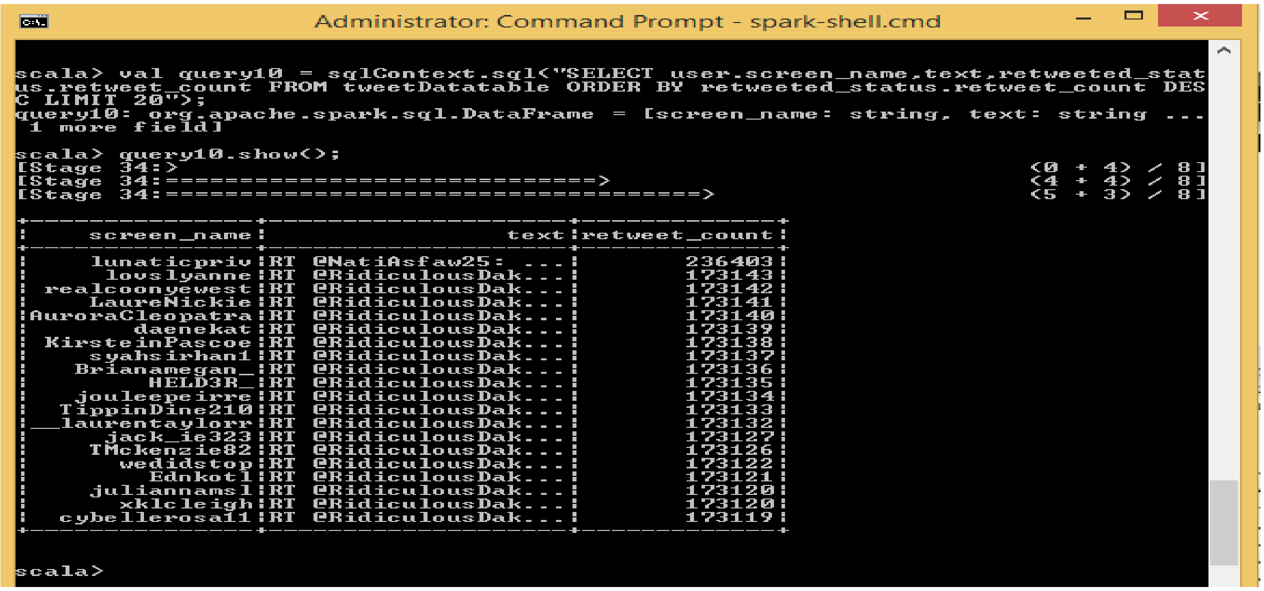
Query 9 : Number of tweets based on different sports



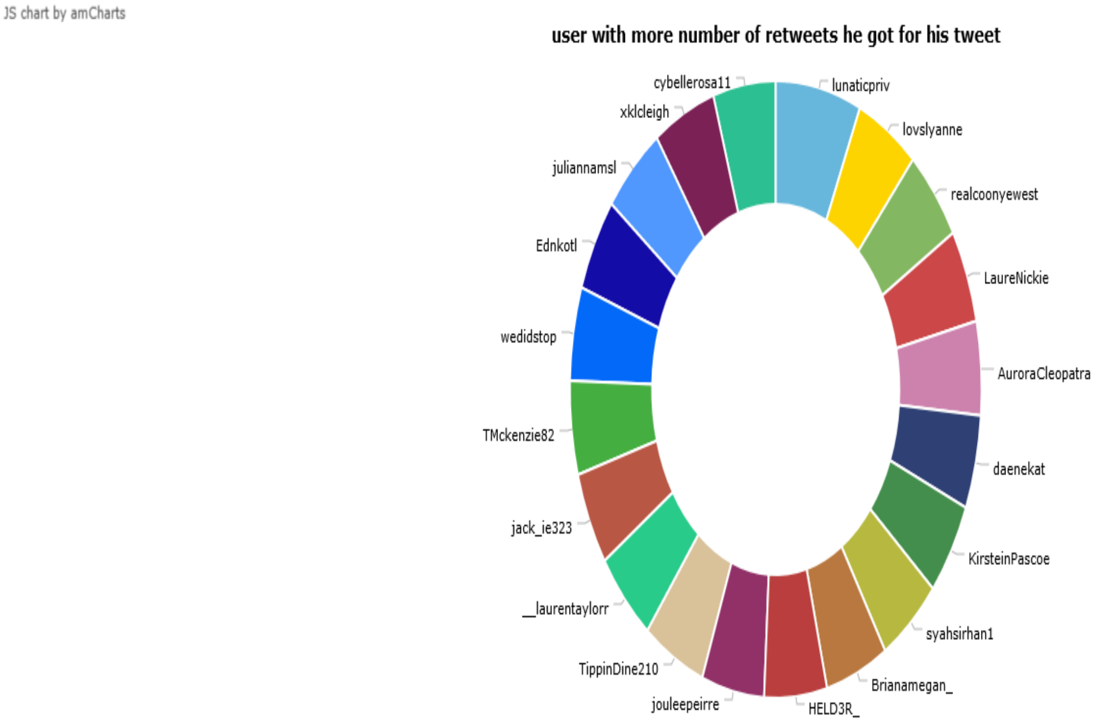
```
scala> val Query9=sqlContext.sql("select count(*) as count,q.text from (select c
ase when text like '%cricket%' then 'cricket' when text like '%football%' then '
football' when text like '%tennis%' then 'tennis' when text like '%wwe%' then 'w
we' when text like '%golf%' then 'golf' when text like '%rugby%' then 'rugby' wh
en text like '%Baseball%' then 'Baseball' else 'different sports' end as text fr
om tweetDatatable)q group by q.text");
Query9: org.apache.spark.sql.DataFrame = [count: bigint, text: string]

scala> val Query9=sqlContext.sql("select count(*) as count,q.text from (select c
ase when text like '%cricket%' then 'cricket' when text like '%football%' then '
football' when text like '%tennis%' then 'tennis' when text like '%wwe%' then 'w
we' when text like '%golf%' then 'golf' when text like '%rugby%' then 'rugby' wh
en text like '%Baseball%' then 'Baseball' WHEN text like '%Badminton%' THEN 'Bad
minton' WHEN text like '%Hockey%' THEN 'Hockey' WHEN text like '%Volleyball%' TH
EN 'Volleyball' else 'different sports' end as text from tweetDatatable)q group
by q.text");
Query9: org.apache.spark.sql.DataFrame = [count: bigint, text: string]

scala> Query9.show();
[Stage 13:>                                                         (0 + 4) / 8]
[Stage 13:=============================>                              (4 + 4) / 8]
[Stage 13:====================================>                       (5 + 3) / 8]
[Stage 13:===========================================>                (6 + 2) / 8]

+------+----------------+
| count|            text|
+------+----------------+
|  3161|         cricket|
|   518|       Badminton|
| 14518|        football|
|115870|different sports|
|   849|             wwe|
|  3852|            golf|
|   874|           rugby|
|  2836|        Baseball|
|   662|      Volleyball|
|  2096|          Hockey|
|  1483|          tennis|
+------+----------------+


scala>
```



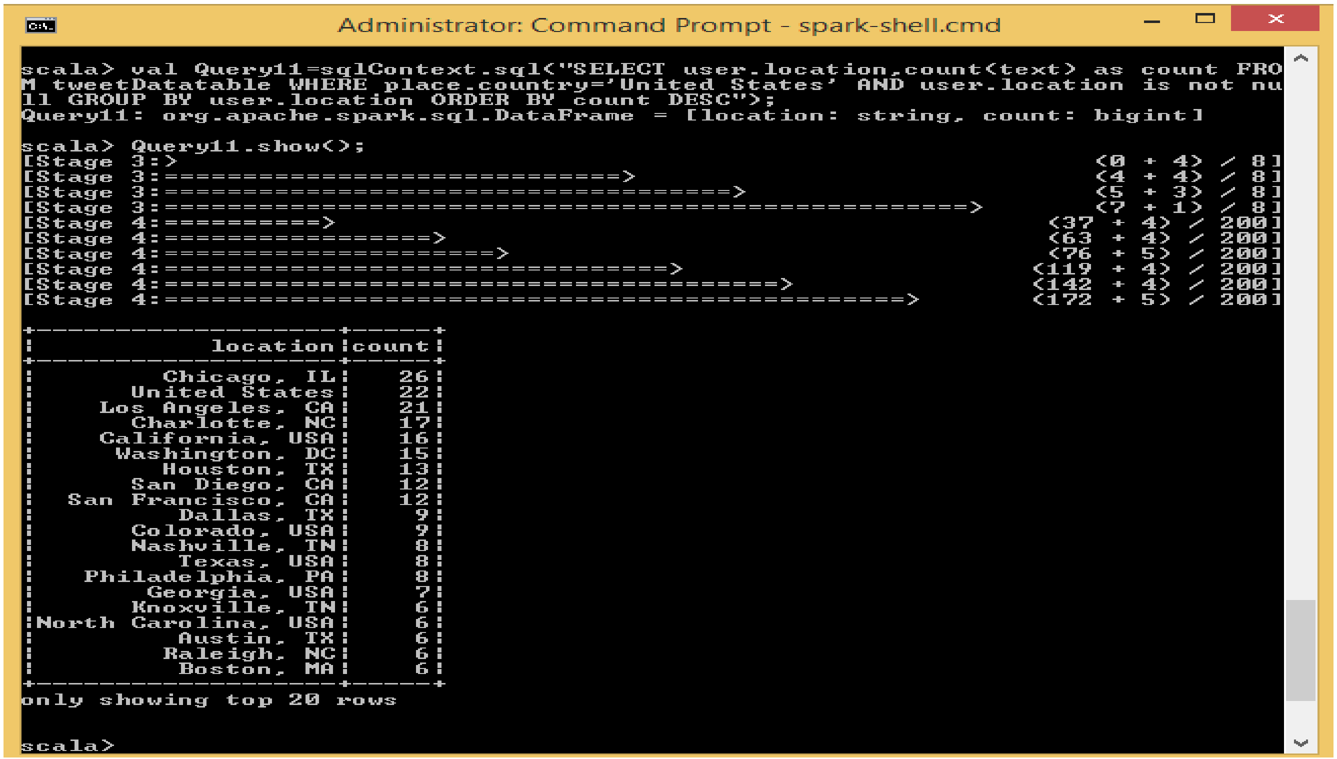
Query 10: User with more number of retweets he got for his tweet



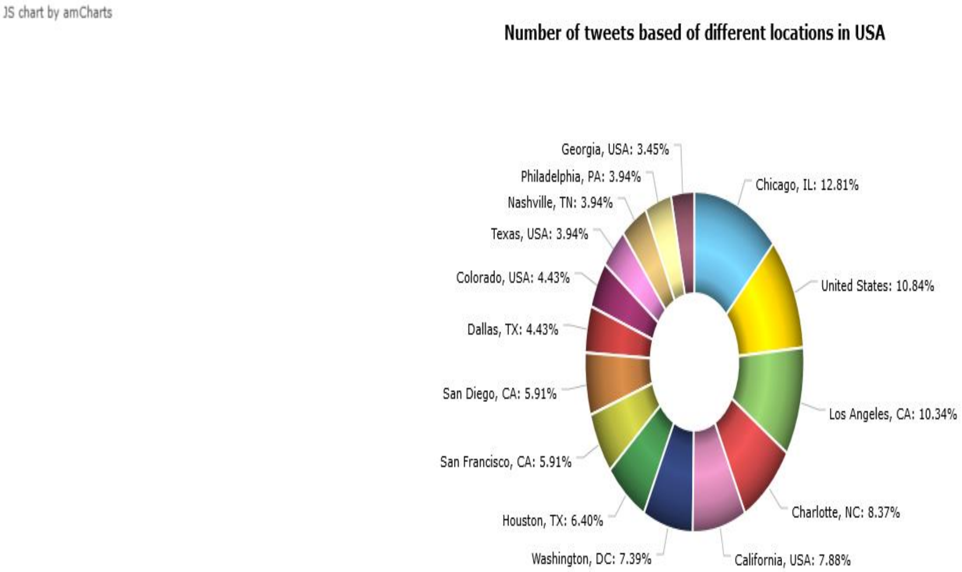
```
scala> val query10 = sqlContext.sql("SELECT user.screen_name,text,retweeted_stat
us.retweet_count FROM tweetDatatable ORDER BY retweeted_status.retweet_count DES
C LIMIT 20");
query10: org.apache.spark.sql.DataFrame = [screen_name: string, text: string ...
 1 more field]

scala> query10.show();
[Stage 34:>                                                         (0 + 4) / 8]
[Stage 34:=============================>                           (4 + 4) / 8]
[Stage 34:====================================>                    (5 + 3) / 8]
+---------------+--------------------+-------------+
|    screen_name|                text|retweet_count|
+---------------+--------------------+-------------+
|    lunaticpriv|RT @NatiAsfaw25: ...|       236403|
|     lovslyanne|RT @RidiculousDak...|       173143|
| realcoonyewest|RT @RidiculousDak...|       173142|
|    LaureNickie|RT @RidiculousDak...|       173141|
|AuroraCleopatra|RT @RidiculousDak...|       173140|
|       daenekat|RT @RidiculousDak...|       173139|
| KirsteinPascoe|RT @RidiculousDak...|       173138|
|    syahsirhan1|RT @RidiculousDak...|       173137|
|   Brianamegan_|RT @RidiculousDak...|       173136|
|        HELD3R_|RT @RidiculousDak...|       173135|
|   jouleepeirre|RT @RidiculousDak...|       173134|
|  TippinDine210|RT @RidiculousDak...|       173133|
|__laurentaylorr|RT @RidiculousDak...|       173132|
|     jack_ie323|RT @RidiculousDak...|       173127|
|    TMckenzie82|RT @RidiculousDak...|       173126|
|      wedidstop|RT @RidiculousDak...|       173122|
|        Ednkotl|RT @RidiculousDak...|       173121|
|    juliannamsl|RT @RidiculousDak...|       173120|
|      xklcleigh|RT @RidiculousDak...|       173120|
|  cybellerosa11|RT @RidiculousDak...|       173119|
+---------------+--------------------+-------------+


scala>
```



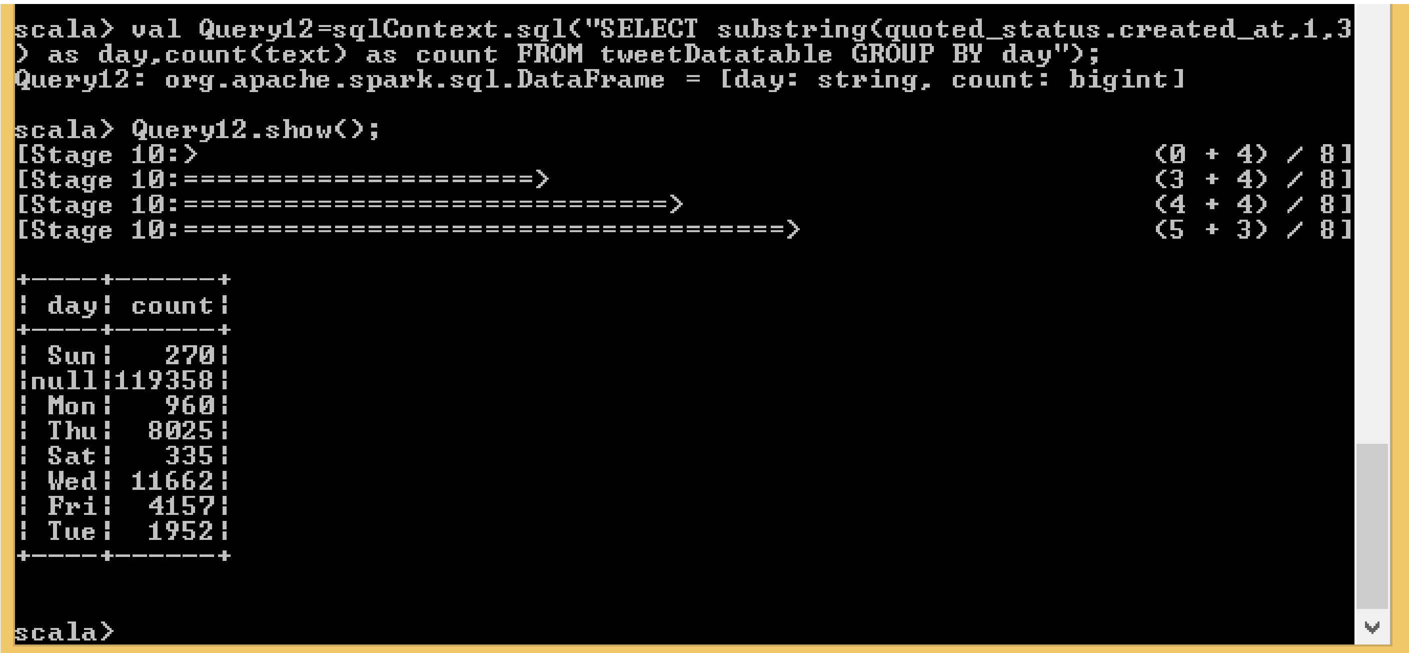
Query 11: Number of tweets based of different locations in USA



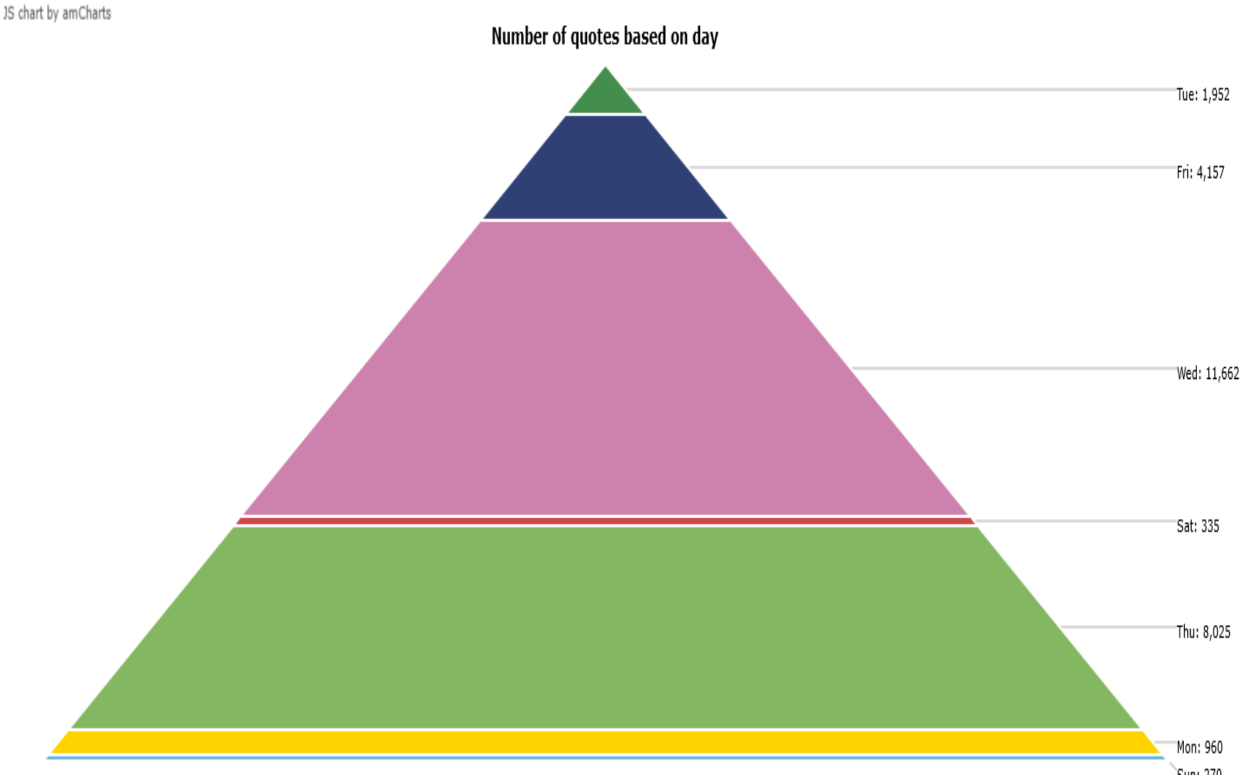
```
scala> val Query11=sqlContext.sql("SELECT user.location,count(text) as count FRO
M tweetDatatable WHERE place.country='United States' AND user.location is not nu
ll GROUP BY user.location ORDER BY count DESC");
Query11: org.apache.spark.sql.DataFrame = [location: string, count: bigint]

scala> Query11.show();
[Stage 3:>                                                          (0 + 4) / 8]
[Stage 3:=============================>                             (4 + 4) / 8]
[Stage 3:====================================>                      (5 + 3) / 8]
[Stage 3:===================================================>       (7 + 1) / 8]
[Stage 4:==========>                                            (37 + 4) / 200]
[Stage 4:=================>                                     (63 + 4) / 200]
[Stage 4:=====================>                                 (76 + 5) / 200]
[Stage 4:================================>                     (119 + 4) / 200]
[Stage 4:=======================================>              (142 + 4) / 200]
[Stage 4:===============================================>      (172 + 5) / 200]

+--------------------+-----+
|            location|count|
+--------------------+-----+
|         Chicago, IL|   26|
|       United States|   22|
|     Los Angeles, CA|   21|
|       Charlotte, NC|   17|
|     California, USA|   16|
|      Washington, DC|   15|
|         Houston, TX|   13|
|       San Diego, CA|   12|
|   San Francisco, CA|   12|
|          Dallas, TX|    9|
|       Colorado, USA|    9|
|       Nashville, TN|    8|
|          Texas, USA|    8|
|    Philadelphia, PA|    8|
|        Georgia, USA|    7|
|       Knoxville, TN|    6|
|North Carolina, USA|    6|
|          Austin, TX|    6|
|         Raleigh, NC|    6|
|          Boston, MA|    6|
+--------------------+-----+
only showing top 20 rows

scala>
```



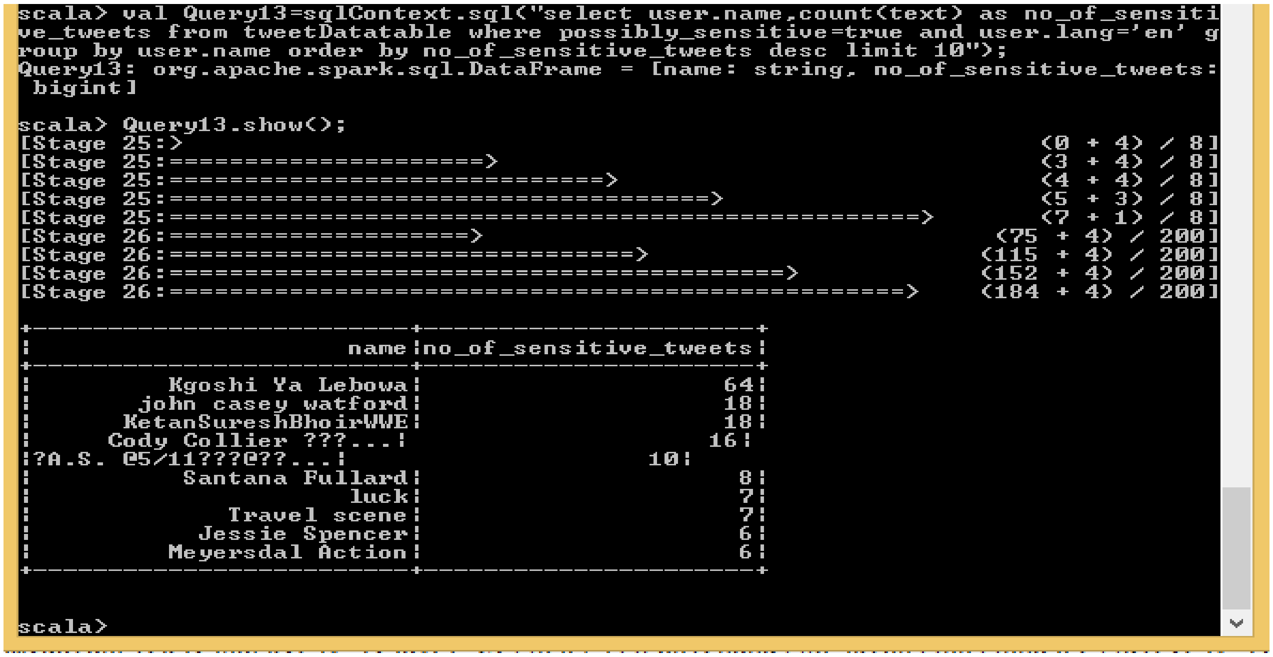
Query 12: Number of quotes based on day



```
scala> val Query12=sqlContext.sql("SELECT substring(quoted_status.created_at,1,3
) as day,count(text) as count FROM tweetDatatable GROUP BY day");
Query12: org.apache.spark.sql.DataFrame = [day: string, count: bigint]

scala> Query12.show();
[Stage 10:>                                                          (0 + 4) / 8]
[Stage 10:=====================>                                    (3 + 4) / 8]
[Stage 10:=============================>                            (4 + 4) / 8]
[Stage 10:====================================>                     (5 + 3) / 8]

+----+------+
| day| count|
+----+------+
| Sun|   270|
|null|119358|
| Mon|   960|
| Thu|  8025|
| Sat|   335|
| Wed| 11662|
| Fri|  4157|
| Tue|  1952|
+----+------+


scala>
```

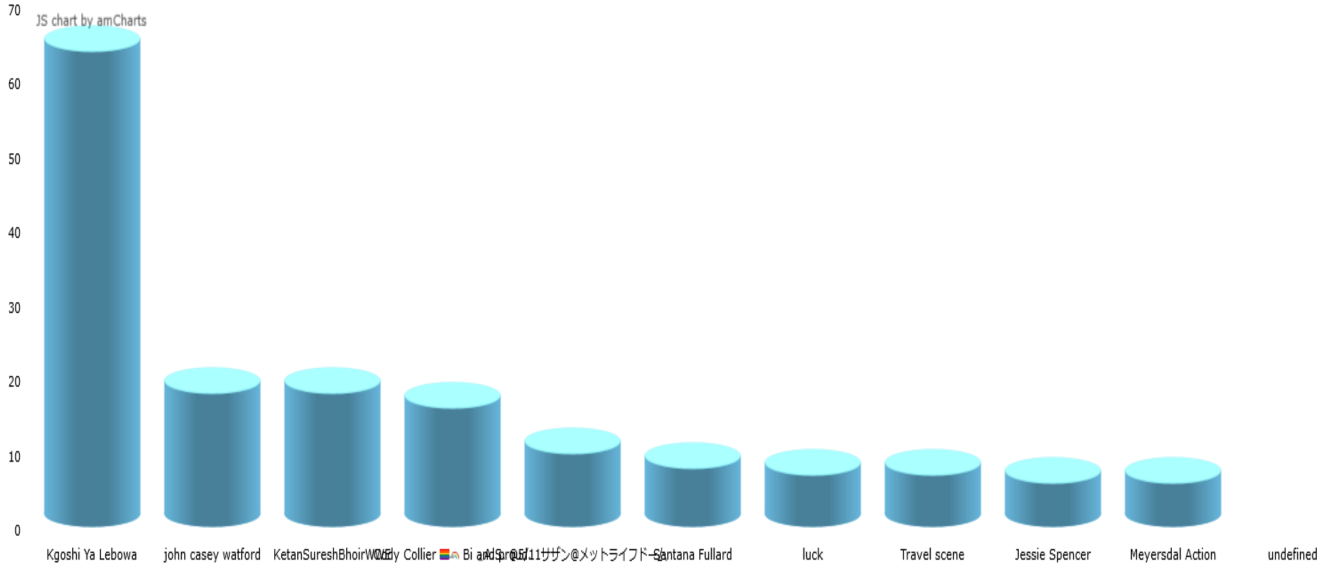


Query 13: Users with most sensitive tweet numbers

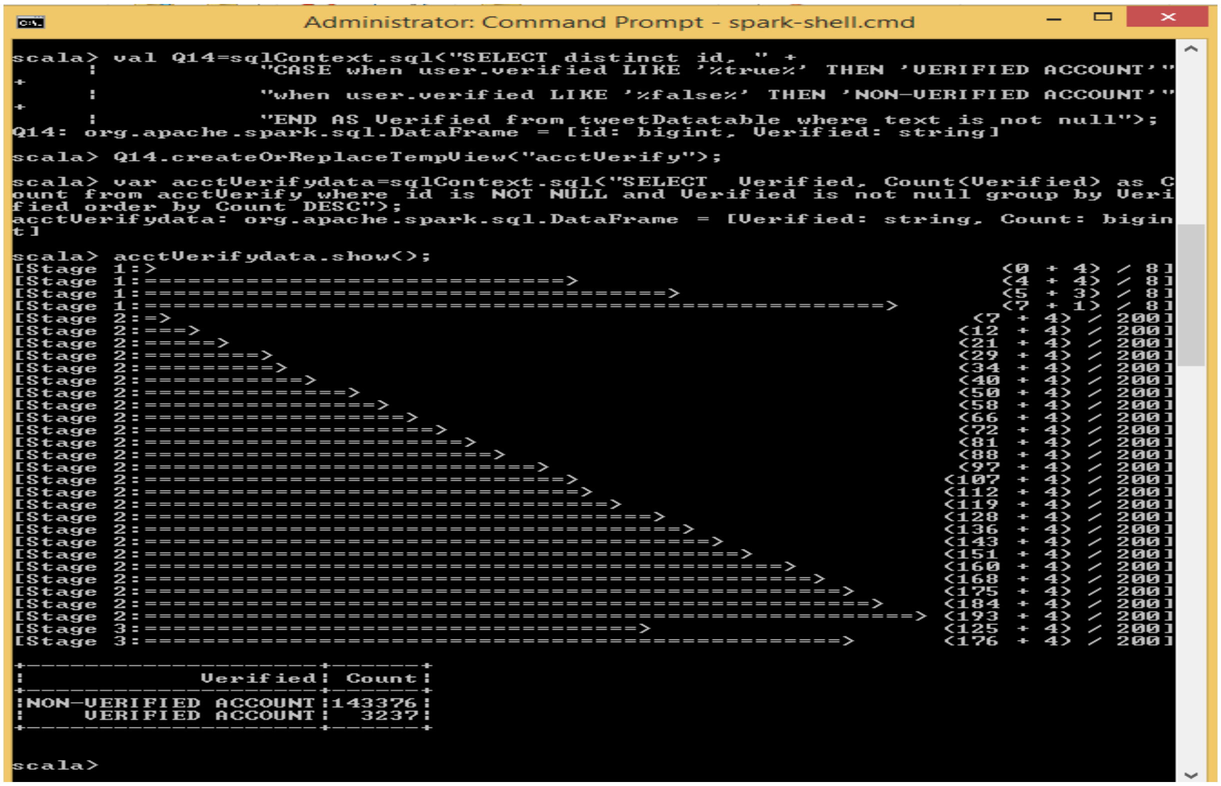


```
scala> val Query13=sqlContext.sql("select user.name,count(text) as no_of_sensiti
ve_tweets from tweetDatatable where possibly_sensitive=true and user.lang='en' g
roup by user.name order by no_of_sensitive_tweets desc limit 10");
Query13: org.apache.spark.sql.DataFrame = [name: string, no_of_sensitive_tweets:
 bigint]

scala> Query13.show();
[Stage 25:>                                                          (0 + 4) / 8]
[Stage 25:=====================>                                    (3 + 4) / 8]
[Stage 25:=============================>                            (4 + 4) / 8]
[Stage 25:====================================>                     (5 + 3) / 8]
[Stage 25:==================================================>       (7 + 1) / 8]
[Stage 26:====================>                                  (75 + 4) / 200]
[Stage 26:===============================>                      (115 + 4) / 200]
[Stage 26:========================================>             (152 + 4) / 200]
[Stage 26:================================================>     (184 + 4) / 200]

+--------------------------+----------------------+
|                      name|no_of_sensitive_tweets|
+--------------------------+----------------------+
|          Kgoshi Ya Lebowa|                    64|
|        john casey watford|                    18|
|       KetanSureshBhoirWWE|                    18|
|       Cody Collier ???...|                    16|
|?A.S. @5/11???@??...|                    10|
|           Santana Fullard|                     8|
|                      luck|                     7|
|               Travel scene|                     7|
|            Jessie Spencer|                     6|
|          Meyersdal Action|                     6|
+--------------------------+----------------------+


scala>
```

Users with most sensitive tweet numbers

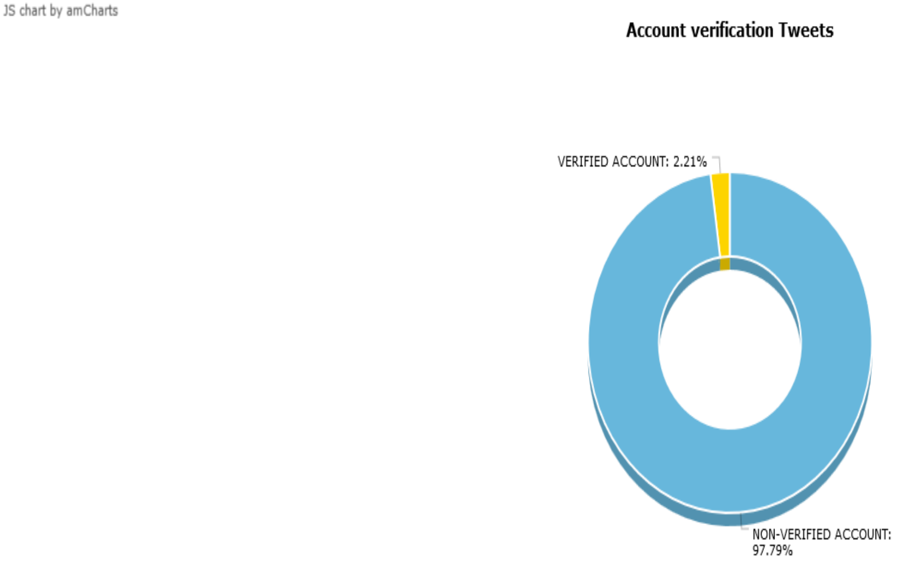


Query 14: Users with most sensitive tweet numbers

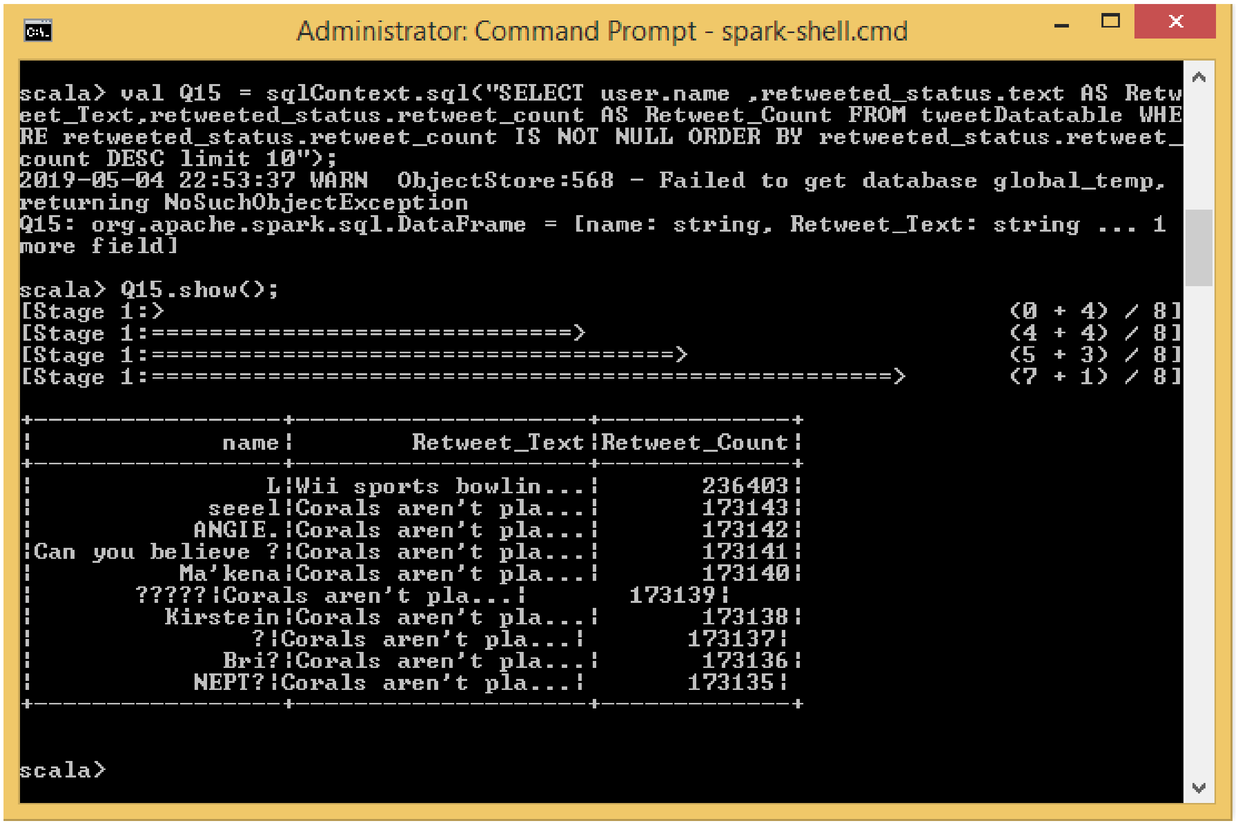


Administrator: Command Prompt - spark-shell.cmd

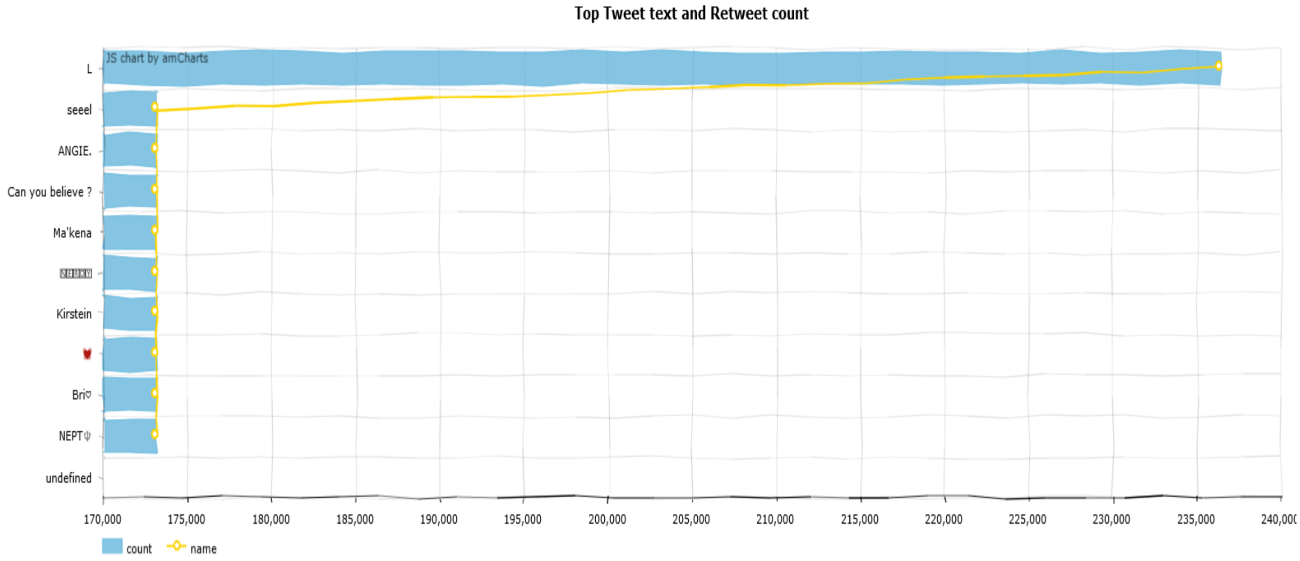
```
scala> val Q14=sqlContext.sql("SELECT distinct id, " +
     |                 "CASE when user.verified LIKE '%true%' THEN 'VERIFIED ACCOUNT'"
+
     |                 "when user.verified LIKE '%false%' THEN 'NON-VERIFIED ACCOUNT'"
+
     |                 "END AS Verified from tweetDatatable where text is not null");
Q14: org.apache.spark.sql.DataFrame = [id: bigint, Verified: string]

scala> Q14.createOrReplaceTempView("acctVerify");

scala> var acctVerifydata=sqlContext.sql("SELECT  Verified, Count(Verified) as C
ount from acctVerify where id is NOT NULL and Verified is not null group by Veri
fied order by Count DESC");
acctVerifydata: org.apache.spark.sql.DataFrame = [Verified: string, Count: bigin
t]

scala> acctVerifydata.show();
[Stage 1:>                                                          (0 + 4) / 8]
[Stage 1:=============================>                             (4 + 4) / 8]
[Stage 1:====================================>                      (5 + 3) / 8]
[Stage 1:===================================================>       (7 + 1) / 8]
[Stage 2:=>                                                     (7 + 4) / 200]
[Stage 2:===>                                                  (12 + 4) / 200]
[Stage 2:=====>                                                (21 + 4) / 200]
[Stage 2:========>                                             (29 + 4) / 200]
[Stage 2:=========>                                            (34 + 4) / 200]
[Stage 2:===========>                                          (40 + 4) / 200]
[Stage 2:==============>                                       (50 + 4) / 200]
[Stage 2:================>                                     (58 + 4) / 200]
[Stage 2:==================>                                   (66 + 4) / 200]
[Stage 2:====================>                                 (72 + 4) / 200]
[Stage 2:======================>                               (81 + 4) / 200]
[Stage 2:========================>                             (88 + 4) / 200]
[Stage 2:===========================>                          (97 + 4) / 200]
[Stage 2:=============================>                       (107 + 4) / 200]
[Stage 2:==============================>                      (112 + 4) / 200]
[Stage 2:================================>                    (119 + 4) / 200]
[Stage 2:===================================>                 (128 + 4) / 200]
[Stage 2:=====================================>               (136 + 4) / 200]
[Stage 2:=======================================>             (143 + 4) / 200]
[Stage 2:=========================================>           (151 + 4) / 200]
[Stage 2:============================================>        (160 + 4) / 200]
[Stage 2:==============================================>      (168 + 4) / 200]
[Stage 2:================================================>    (175 + 4) / 200]
[Stage 2:==================================================>  (184 + 4) / 200]
[Stage 2:=====================================================> (193 + 4) / 200]
[Stage 3:==================================>                  (125 + 4) / 200]
[Stage 3:================================================>    (176 + 4) / 200]

+--------------------+------+
|            Verified| Count|
+--------------------+------+
|NON-VERIFIED ACCOUNT|143376|
|    VERIFIED ACCOUNT|  3237|
+--------------------+------+


scala>
```



Query 15: Top Tweet text and Retweet count

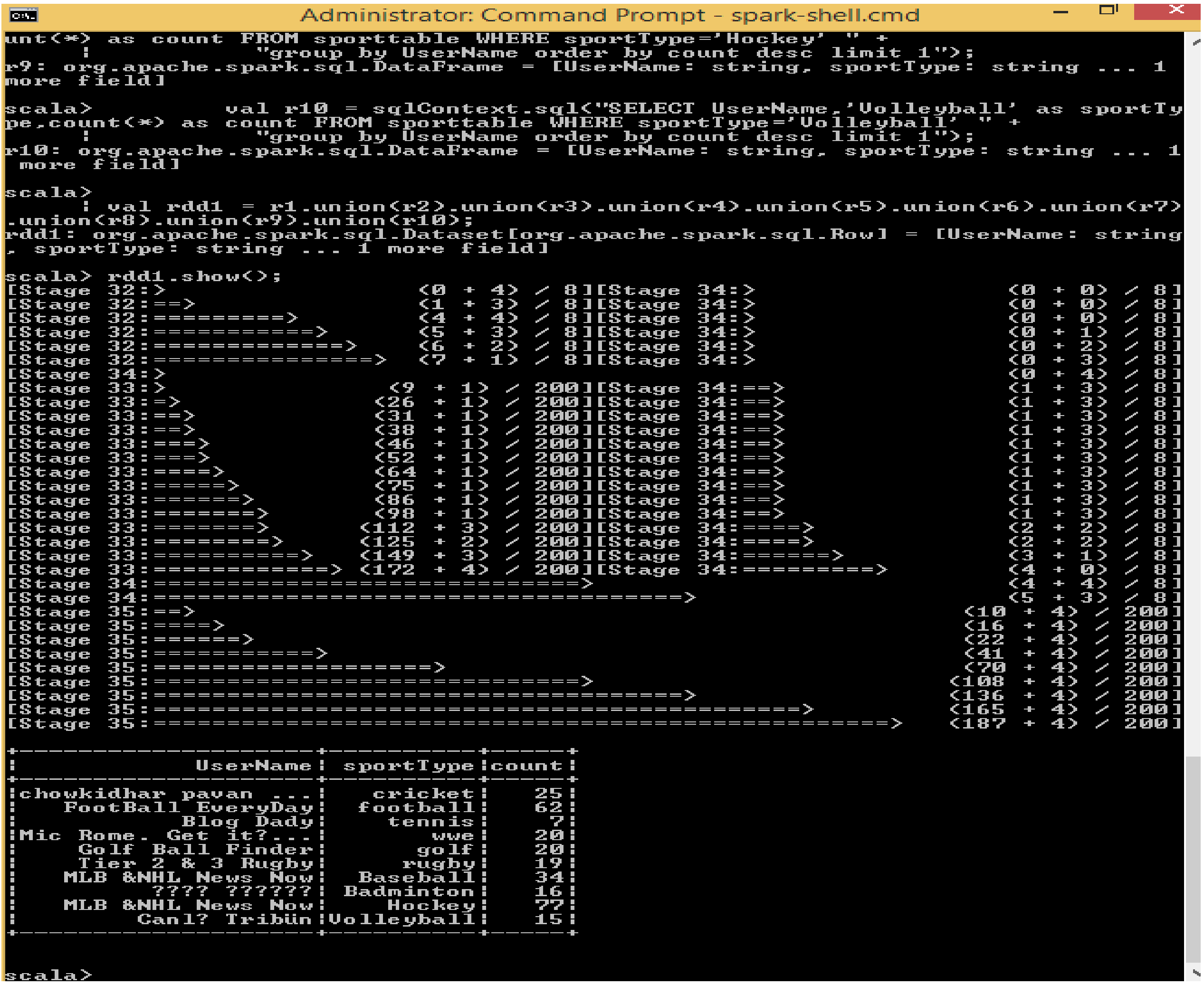


```
scala> val Q15 = sqlContext.sql("SELECT user.name ,retweeted_status.text AS Retw
eet_Text,retweeted_status.retweet_count AS Retweet_Count FROM tweetDatatable WHE
RE retweeted_status.retweet_count IS NOT NULL ORDER BY retweeted_status.retweet_
count DESC limit 10");
2019-05-04 22:53:37 WARN  ObjectStore:568 - Failed to get database global_temp,
returning NoSuchObjectException
Q15: org.apache.spark.sql.DataFrame = [name: string, Retweet_Text: string ... 1
more field]

scala> Q15.show();
[Stage 1:>                                                          (0 + 4) / 8]
[Stage 1:=============================>                             (4 + 4) / 8]
[Stage 1:====================================>                      (5 + 3) / 8]
[Stage 1:===================================================>       (7 + 1) / 8]

+-----------------+--------------------+-------------+
|             name|        Retweet_Text|Retweet_Count|
+-----------------+--------------------+-------------+
|                L|Wii sports bowlin...|       236403|
|            seeel|Corals aren't pla...|       173143|
|           ANGIE.|Corals aren't pla...|       173142|
|Can you believe ?|Corals aren't pla...|       173141|
|          Ma'kena|Corals aren't pla...|       173140|
|        ?????|Corals aren't pla...|       173139|
|         Kirstein|Corals aren't pla...|       173138|
|              ?|Corals aren't pla...|       173137|
|             Bri?|Corals aren't pla...|       173136|
|           NEPT?|Corals aren't pla...|       173135|
+-----------------+--------------------+-------------+


scala>
```

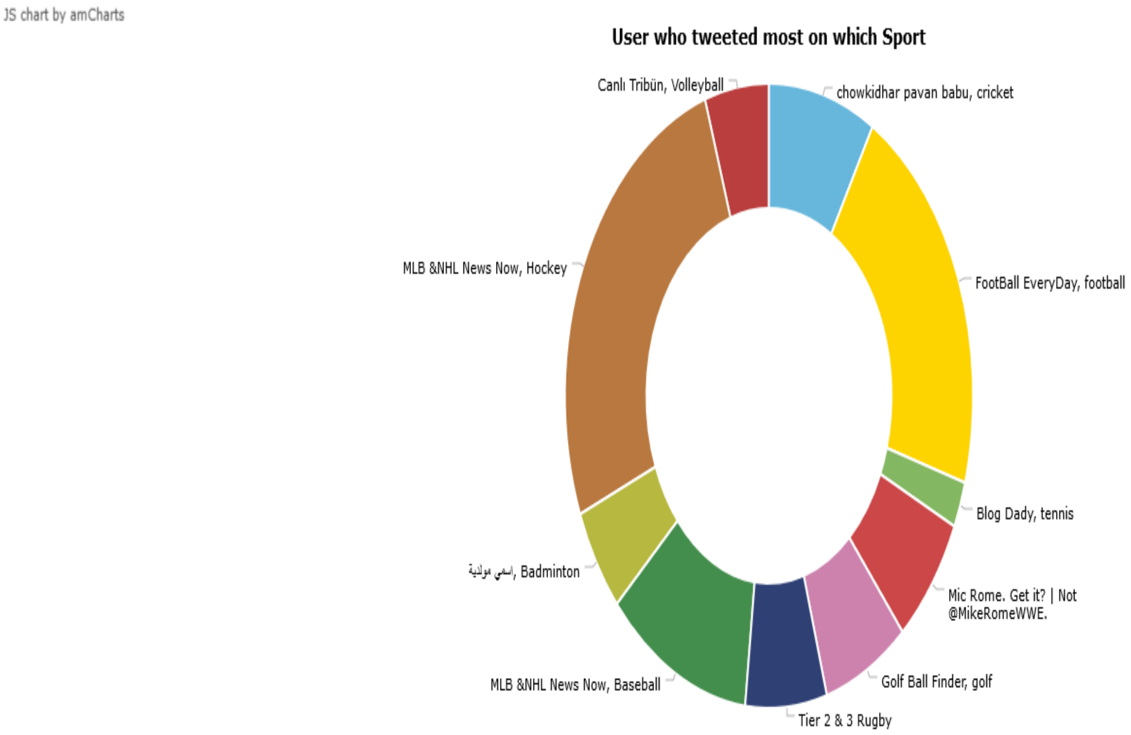


Query 16: User who tweeted most on which Sport

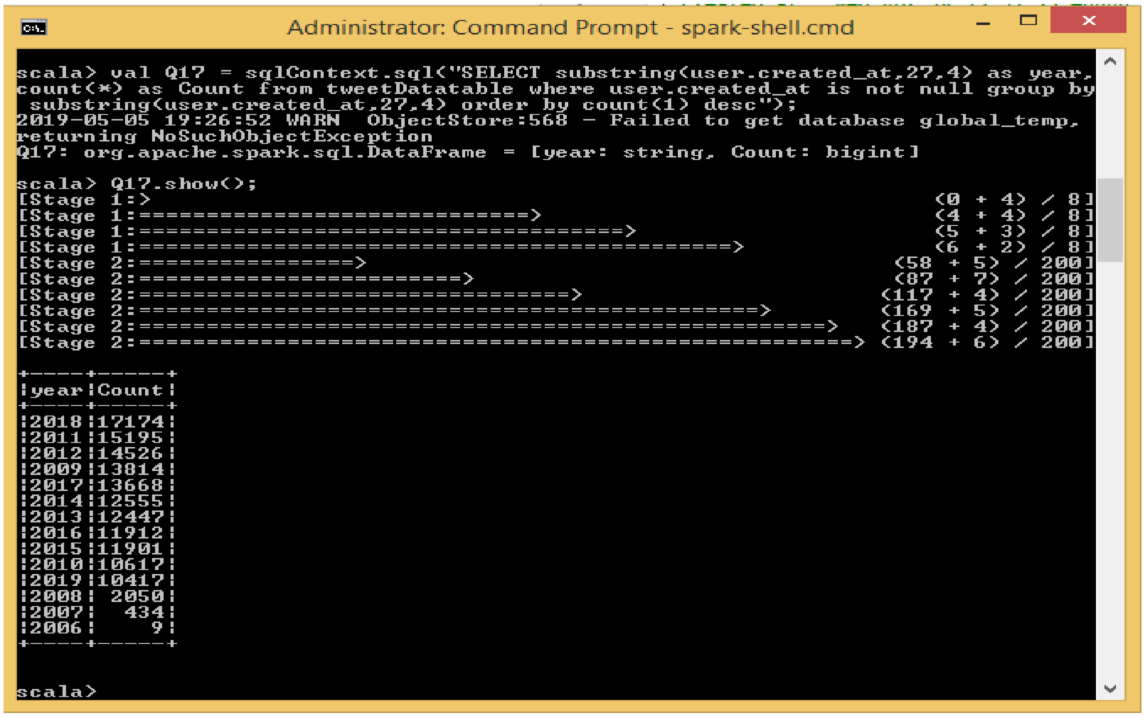
```
#User who tweeted most on which Sport
val Sports = sqlContext.sql("SELECT user.name as UserName,user.location as loc,text,created_at," +
      "CASE WHEN text like '%cricket%' THEN 'cricket'" +
      "WHEN text like '%football%' THEN 'football'" +
      "WHEN text like '%tennis%' THEN 'tennis'" +
      "WHEN text like '%wwe%' THEN 'wwe'" +
      "WHEN text LIKE '%golf%' THEN 'golf'" +
      "WHEN text like '%rugby%' THEN 'rugby'" +
      "WHEN text like '%Baseball%' THEN 'Baseball'" +
      "WHEN text like '%Badminton%' THEN 'Badminton'" +
      "WHEN text like '%Hockey%' THEN 'Hockey'" +
      "WHEN text like '%Volleyball%' THEN 'Volleyball'" +
      "END AS sportType from tweetDatatable where text is not null");

Sports.createOrReplaceTempView("sporttable");


 val r1 = sqlContext.sql("SELECT UserName,sportType,count(*) as count FROM sporttable WHERE sportType='cricket' " +
         "group by UserName,sportType order by count desc limit 1");
        val r2 = sqlContext.sql("SELECT UserName,'football' as sportType,count(*) as count FROM sporttable WHERE sportType='football' " +
         "group by UserName order by count desc limit 1 ");
        val r3 = sqlContext.sql("SELECT UserName,'tennis' as sportType,count(*) as count FROM sporttable WHERE sportType='tennis' " +
         "group by UserName order by count desc limit 1 ");
        val r4 = sqlContext.sql("SELECT UserName,'wwe' as sportType,count(*) as count FROM sporttable WHERE sportType='wwe' " +
         "group by UserName order by count desc limit 1 ");
        val r5 = sqlContext.sql("SELECT UserName,'golf' as sportType,count(*) as count FROM sporttable WHERE sportType='golf' " +
         "group by UserName order by count desc limit 1 ");
        val r6 = sqlContext.sql("SELECT UserName,'rugby' as sportType,count(*) as count FROM sporttable WHERE sportType='rugby' " +
         "group by UserName order by count desc limit 1 ");
        val r7 = sqlContext.sql("SELECT UserName,'Baseball' as sportType,count(*) as count FROM sporttable WHERE sportType='Baseball' " +
         "group by UserName order by count desc limit 1 ");
        val r8 = sqlContext.sql("SELECT UserName,'Badminton' as sportType,count(*) as count FROM sporttable WHERE sportType='Badminton' " +
         "group by UserName order by count desc limit 1 ");
        val r9 = sqlContext.sql("SELECT UserName,'Hockey' as sportType,count(*) as count FROM sporttable WHERE sportType='Hockey' " +
         "group by UserName order by count desc limit 1");
        val r10 = sqlContext.sql("SELECT UserName,'Volleyball' as sportType,count(*) as count FROM sporttable WHERE sportType='Volleyball' " +
         "group by UserName order by count desc limit 1");
```



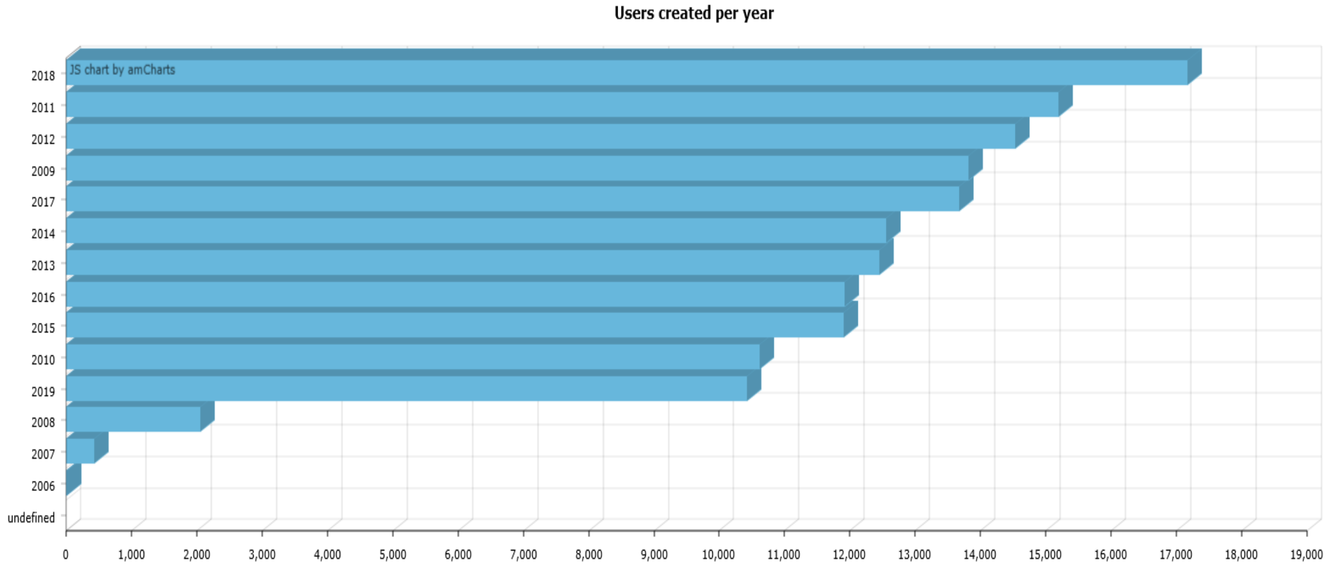
```
unt(*) as count FROM sporttable WHERE sportType='Hockey' " +
     |           "group by UserName order by count desc limit 1");
r9: org.apache.spark.sql.DataFrame = [UserName: string, sportType: string ... 1
more field]

scala>         val r10 = sqlContext.sql("SELECT UserName,'Volleyball' as sportTy
pe,count(*) as count FROM sporttable WHERE sportType='Volleyball' " +
     |           "group by UserName order by count desc limit 1");
r10: org.apache.spark.sql.DataFrame = [UserName: string, sportType: string ... 1
 more field]

scala>
     | val rdd1 = r1.union(r2).union(r3).union(r4).union(r5).union(r6).union(r7)
.union(r8).union(r9).union(r10);
rdd1: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [UserName: string
, sportType: string ... 1 more field]

scala> rdd1.show();
[Stage 32:>                 (0 + 4) / 8][Stage 34:>                 (0 + 0) / 8]
[Stage 32:==>               (1 + 3) / 8][Stage 34:>                 (0 + 0) / 8]
[Stage 32:=========>        (4 + 4) / 8][Stage 34:>                 (0 + 0) / 8]
[Stage 32:===========>      (5 + 3) / 8][Stage 34:>                 (0 + 1) / 8]
[Stage 32:=============>    (6 + 2) / 8][Stage 34:>                 (0 + 2) / 8]
[Stage 32:===============>  (7 + 1) / 8][Stage 34:>                 (0 + 3) / 8]
[Stage 34:>                                                         (0 + 4) / 8]
[Stage 33:>               (9 + 1) / 200][Stage 34:==>               (1 + 3) / 8]
[Stage 33:=>             (26 + 1) / 200][Stage 34:==>               (1 + 3) / 8]
[Stage 33:==>            (31 + 1) / 200][Stage 34:==>               (1 + 3) / 8]
[Stage 33:==>            (38 + 1) / 200][Stage 34:==>               (1 + 3) / 8]
[Stage 33:===>           (46 + 1) / 200][Stage 34:==>               (1 + 3) / 8]
[Stage 33:===>           (52 + 1) / 200][Stage 34:==>               (1 + 3) / 8]
[Stage 33:====>          (64 + 1) / 200][Stage 34:==>               (1 + 3) / 8]
[Stage 33:=====>         (75 + 1) / 200][Stage 34:==>               (1 + 3) / 8]
[Stage 33:======>        (86 + 1) / 200][Stage 34:==>               (1 + 3) / 8]
[Stage 33:=======>       (98 + 1) / 200][Stage 34:==>               (1 + 3) / 8]
[Stage 33:=======>      (112 + 3) / 200][Stage 34:====>             (2 + 2) / 8]
[Stage 33:========>     (125 + 2) / 200][Stage 34:====>             (2 + 2) / 8]
[Stage 33:==========>   (149 + 3) / 200][Stage 34:======>           (3 + 1) / 8]
[Stage 33:============> (172 + 4) / 200][Stage 34:=========>        (4 + 0) / 8]
[Stage 34:=============================>                            (4 + 4) / 8]
[Stage 34:====================================>                     (5 + 3) / 8]
[Stage 35:==>                                                    (10 + 4) / 200]
[Stage 35:====>                                                  (16 + 4) / 200]
[Stage 35:======>                                               (22 + 4) / 200]
[Stage 35:===========>                                          (41 + 4) / 200]
[Stage 35:===================>                                  (70 + 4) / 200]
[Stage 35:=============================>                       (108 + 4) / 200]
[Stage 35:====================================>                (136 + 4) / 200]
[Stage 35:============================================>        (165 + 4) / 200]
[Stage 35:==================================================>  (187 + 4) / 200]

+--------------------+----------+-----+
|            UserName| sportType|count|
+--------------------+----------+-----+
|chowkidhar pavan ...|   cricket|   25|
|   FootBall EveryDay|  football|   62|
|           Blog Dady|    tennis|    7|
|Mic Rome. Get it?...|       wwe|   20|
|    Golf Ball Finder|      golf|   20|
|    Tier 2 & 3 Rugby|     rugby|   19|
|   MLB &NHL News Now|  Baseball|   34|
|         ???? ??????| Badminton|   16|
|   MLB &NHL News Now|    Hockey|   77|
|        Canl? Tribün|Volleyball|   15|
+--------------------+----------+-----+


scala>
```



Query 17: Users created per year



```
scala> val Q17 = sqlContext.sql("SELECT substring(user.created_at,27,4) as year,
count(*) as Count from tweetDatatable where user.created_at is not null group by
 substring(user.created_at,27,4) order by count(1) desc");
2019-05-05 19:26:52 WARN  ObjectStore:568 - Failed to get database global_temp,
returning NoSuchObjectException
Q17: org.apache.spark.sql.DataFrame = [year: string, Count: bigint]

scala> Q17.show();
[Stage 1:>                                                          (0 + 4) / 8]
[Stage 1:=============================>                             (4 + 4) / 8]
[Stage 1:====================================>                      (5 + 3) / 8]
[Stage 1:============================================>              (6 + 2) / 8]
[Stage 2:================>                                       (58 + 5) / 200]
[Stage 2:========================>                               (87 + 7) / 200]
[Stage 2:================================>                      (117 + 4) / 200]
[Stage 2:==============================================>        (169 + 5) / 200]
[Stage 2:===================================================>   (187 + 4) / 200]
[Stage 2:=====================================================> (194 + 6) / 200]

+----+-----+
|year|Count|
+----+-----+
|2018|17174|
|2011|15195|
|2012|14526|
|2009|13814|
|2017|13668|
|2014|12555|
|2013|12447|
|2016|11912|
|2015|11901|
|2010|10617|
|2019|10417|
|2008| 2050|
|2007|  434|
|2006|    9|
+----+-----+

scala>
```



Software Testing:

By Unit Testing, the individual units or components are tested. Unit Testing of software has done during the development of application.

Code :

Open command prompt in 'run as administrator' mode

```
#starting spark shell, scala
1. spark-shell.cmd

#starting sparksql
2. val sqlContext = new org.apache.spark.sql.SQLContext(sc);

#Importing the SQL context gives access to all the public SQL functions
3. import sqlContext.implicits._

#importing extracted twitter data file from local system to sparksql
4. val tweetstable = sqlContext.read.json("C:\\Users\\gopichand\\Desktop\\pb\\phase2\\twitdb1.json");

#creating temporary table in sparksql
5. tweetstable.registerTempTable("tweetDatatable");

#number of users(accounts) created per month
6. val Query1 = sqlContext.sql("SELECT substring(user.created_at,5,3) as month, count(user.id) from
tweetDatatable group by month");

#to display output in spark
```

```
7. Query1.show();

#importing output from spark to local system(a folder q1.csv will be created in this path
C:\\Users\\gopichand\\Desktop\\pb\\phase2
9.
Query1.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\gopichand\\Desktop\\pb\\phase2\
\q1.csv");

#users with more tweets in descending order
10. val Query2=sqlContext.sql("SELECT count(*) as count, user.name from tweetDatatable where user.name is not
null group by user.name order by count desc limit 10");
11. Query2.show();
12.
Query2.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\gopichand\\Desktop\\pb\\phase2\
\q2.csv");

#number of tweets from different countries
13. val Query3 = sqlContext.sql("SELECT place.country,count(*) AS count FROM tweetDatatable GROUP BY
place.country ORDER BY count DESC limit 10");
14. Query3.show();
15.
Query3.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\gopichand\\Desktop\\pb\\phase2\
\q3.csv");

#users with more followers and language used, in descending order
16. val Query4 = sqlContext.sql("SELECT user.name, max(user.followers_count) as followers_count, user.lang
FROM tweetDatatable WHERE text like '%sport%' group by user.name,user.lang order by followers_count desc
limit 15");
17. Query4.show();
18.
Query4.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\gopichand\\Desktop\\pb\\phase2\
\q4.csv");

#popuar languages used for more number of tweets
19. val Query5 = sqlContext.sql("SELECT user.lang, count(*) AS count FROM tweetDatatable WHERE lang<>'null'
GROUP BY user.lang ORDER BY count DESC LIMIT 10");
20. Query5.show();
21.
Query5.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\gopichand\\Desktop\\pb\\phase2\
\q5.csv");

#number of users created according to day wise
22. val Query6=sqlContext.sql("SELECT substring(user.created_at,1,3) as day,count(*) as count from
tweetDatatable group by day");
23. Query6.show();
24.
Query6.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\gopichand\\Desktop\\pb\\phase2\
\q6.csv");

#number of tweets based on timezone
25. val Query7 = sqlContext.sql("SELECT user.time_zone,count(text) AS count FROM tweetDatatable GROUP BY
user.time_zone ORDER BY count DESC LIMIT 15");
26. Query7.show();
```

```
27. Query7.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\gopichand\\Desktop\\pb\\phase2\\q7.csv");

#normal users(not verified accunts) with more number of followers
28. val Query8 = sqlContext.sql("SELECT user.verified,user.screen_name,max(user.followers_count) as followers_count FROM tweetDatatable WHERE user.verified = false GROUP BY user.verified, user.screen_name ORDER BY followers_count DESC LIMIT 15");
29. Query8.show();
30. Query8.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\gopichand\\Desktop\\pb\\phase2\\q8.csv");

#number of tweets based on different sports
31. val Query9=sqlContext.sql("select count(*) as count,q.text from (select case when text like '%cricket%' then 'cricket' when text like '%football%' then 'football' when text like '%tennis%' then 'tennis' when text like '%wwe%' then 'wwe' when text like '%golf%' then 'golf' when text like '%rugby%' then 'rugby' when text like '%Baseball%' then 'Baseball' WHEN text like '%Badminton%' THEN 'Badminton' WHEN text like '%Hockey%' THEN 'Hockey' WHEN text like '%Volleyball%' THEN 'Volleyball' else 'different sports' end as text from tweetDatatable)q group by q.text");

 WHEN text like '%Baseball%' THEN 'Baseball WHEN text


32. Query9.show();
33. Query9.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\gopichand\\Desktop\\pb\\phase2\\q9.csv");

#user with more number of retweets he got for his tweet
34. val query10 = sqlContext.sql("SELECT user.screen_name,text,retweeted_status.retweet_count FROM tweetDatatable ORDER BY retweeted_status.retweet_count DESC LIMIT 20");
35. query10.show();
36. query10.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\gopichand\\Desktop\\pb\\phase2\\q10.csv");

#Number of tweets based of different location in USA
37. val Query11=sqlContext.sql("SELECT user.location,count(text) as count FROM tweetDatatable WHERE place.country='United States' AND user.location is not null GROUP BY user.location ORDER BY count DESC LIMIT 15");
38. Query11.show();
39. Query11.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\gopichand\\Desktop\\pb\\phase2\\q11.csv");

#number of quotes based on day
40. val Query12=sqlContext.sql("SELECT substring(quoted_status.created_at,1,3) as day,count(text) as count FROM tweetDatatable GROUP BY day");
41. Query12.show();
42. Query12.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\gopichand\\Desktop\\pb\\phase2\\q12.csv");
```

```
# Users with most sensitive tweet numbers
43. val Query13=sqlContext.sql("select user.name,count(text) as no_of_sensitive_tweets from tweetDatatable
where possibly_sensitive=true and user.lang='en' group by user.name order by no_of_sensitive_tweets desc limit
10");
44. Query13.show();
42.
Query13.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\gopichand\\Desktop\\pb\\phase2
\\q13.csv");

#User who tweeted most on which Sport
val Sports = sqlContext.sql("SELECT user.name as UserName,user.location as loc,text,created_at," +
     "CASE WHEN text like '%cricket%' THEN 'cricket'" +
     "WHEN text like '%football%' THEN 'football'" +
     "WHEN text like '%tennis%' THEN 'tennis'" +
     "WHEN text like '%wwe%' THEN 'wwe'" +
     "WHEN text LIKE '%golf%' THEN 'golf'" +
     "WHEN text like '%rugby%' THEN 'rugby'" +
     "WHEN text like '%Baseball%' THEN 'Baseball'" +
     "WHEN text like '%Badminton%' THEN 'Badminton'" +
     "WHEN text like '%Hockey%' THEN 'Hockey'" +
     "WHEN text like '%Volleyball%' THEN 'Volleyball'" +
     "END AS sportType from tweetDatatable where text is not null");

Sports.createOrReplaceTempView("sporttable");


 val r1 = sqlContext.sql("SELECT UserName,sportType,count(*) as count FROM sporttable WHERE
sportType='cricket' " +
         "group by UserName,sportType order by count desc limit 1");
       val r2 = sqlContext.sql("SELECT UserName,'football' as sportType,count(*) as count FROM sporttable WHERE
sportType='football' " +
         "group by UserName order by count desc limit 1 ");
       val r3 = sqlContext.sql("SELECT UserName,'tennis' as sportType,count(*) as count FROM sporttable WHERE
sportType='tennis' " +
         "group by UserName order by count desc limit 1 ");
       val r4 = sqlContext.sql("SELECT UserName,'wwe' as sportType,count(*) as count FROM sporttable WHERE
sportType='wwe' " +
         "group by UserName order by count desc limit 1 ");
       val r5 = sqlContext.sql("SELECT UserName,'golf' as sportType,count(*) as count FROM sporttable WHERE
sportType='golf' " +
         "group by UserName order by count desc limit 1 ");
       val r6 = sqlContext.sql("SELECT UserName,'rugby' as sportType,count(*) as count FROM sporttable WHERE
sportType='rugby' " +
         "group by UserName order by count desc limit 1 ");
       val r7 = sqlContext.sql("SELECT UserName,'Baseball' as sportType,count(*) as count FROM sporttable WHERE
sportType='Baseball' " +
         "group by UserName order by count desc limit 1 ");
       val r8 = sqlContext.sql("SELECT UserName,'Badminton' as sportType,count(*) as count FROM sporttable
WHERE sportType='Badminton' " +
         "group by UserName order by count desc limit 1 ");
       val r9 = sqlContext.sql("SELECT UserName,'Hockey' as sportType,count(*) as count FROM sporttable WHERE
sportType='Hockey' " +
     "group by UserName order by count desc limit 1");
   val r10 = sqlContext.sql("SELECT UserName,'Volleyball' as sportType,count(*) as count FROM sporttable
WHERE sportType='Volleyball' " +
     "group by UserName order by count desc limit 1");


val Q16 = r1.union(r2).union(r3).union(r4).union(r5).union(r6).union(r7).union(r8).union(r9).union(r10);
rdd1.show();
rdd1.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\gopichand\\Desktop\\pb\\phase2\\rd
d1.csv");


#Account verification Tweets
val Q14=sqlContext.sql("SELECT distinct id, " +
     "CASE when user.verified LIKE '%true%' THEN 'VERIFIED ACCOUNT'"+
     "when user.verified LIKE '%false%' THEN 'NON-VERIFIED ACCOUNT'"+
     "END AS Verified from tweetDatatable where text is not null");
   Q14.createOrReplaceTempView("acctVerify");
   var acctVerifydata=sqlContext.sql("SELECT  Verified, Count(Verified) as Count from acctVerify where id is NOT
NULL and Verified is not null group by Verified order by Count DESC");

acctVerifydata.show();
acctVerifydata.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\gopichand\\Desktop\\pb\\p
hase2\\q14.csv");


#Top Tweet text and Retweet count
val Q15 = sqlContext.sql("SELECT user.name ,retweeted_status.text AS
Retweet_Text,retweeted_status.retweet_count AS Retweet_Count FROM tweetDatatable WHERE
retweeted_status.retweet_count IS NOT NULL ORDER BY retweeted_status.retweet_count DESC limit 10");
Q15.show();
Q15.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\gopichand\\Desktop\\pb\\phase2\\q1
5.csv");

£Users created per year
val Q17 = sqlContext.sql("SELECT substring(user.created_at,27,4) as year,count(*) as Count from tweetDatatable
where user.created_at is not null group by substring(user.created_at,27,4) order by count(1) desc");
Q17.show();
Q17.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\gopichand\\Desktop\\pb\\phase2\\q1
7.csv");
```

Python code for extracting twitter data

```
#Import the necessary methods from tweepy library
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream

#Variables that contains the user credentials to access Twitter API
```

```
access_token = "2219941182-hJEd5re1y7lbZmVlyZySZvVsJf88fP6um3SsC3r"
access_token_secret = "BntHym97rzCisKS3BFXqrBgQbgokklZEBcqHXixGJQtX8"
consumer_key = "187ztf3hxmT3Nm3YonFzcAvEB"
consumer_secret = "hTqPaSjNXw21GXmPCey6CZBCZRoO1EbTkbVO4zMv77kN8Ikq0P"



#This is a basic listener that just prints received tweets to stdout.
class StdOutListener(StreamListener):

    def on_data(self, data):
        print (data)
        saveFile = open(r'twitdb1.json','a')
        saveFile.write(data)
        saveFile.close()


    def on_error(self, status):
        print (status)



if __name__ == '__main__':


    #This handles Twitter authetification and the connection to Twitter Streaming API
    l = StdOutListener()
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    stream = Stream(auth, l)

    #This line filter Twitter Streams to capture data by the keywords: 'python', 'javascript', 'ruby'
    stream.filter(track=['Baseball', 'Cricket', 'Football', 'Tennis', 'Golf', 'WWE', 'Badminton', 'Tennis', 'Baseball',
'Hockey', 'Volleyball', 'Rugby', 'Athletics', 'Boxing', 'MotoGP ', 'Cycling', 'Swimming', 'Snooker', 'Gymnastics',
'Handball', 'Skiing', 'Hurling', 'Bowling', 'Lacrosse', 'Archery', 'Bocce', 'Broomball', 'Croquet', 'Diving', 'Fencing',
'Darts', 'Dodgeball', 'Fishing', 'Foosball', 'Kayaking', 'Kickball', 'Racquetball', 'Powerlifting', 'Shooting', 'Sailing',
'Rowing'])
```