Principles of Big Data

Twitter Data Analysis using Apache Spark

Increment 2 Report

Instructor:

Dr. Praveen Rao

Presented by,

Prasanna Muppidi(pmyrd),

Santhosh Mohan Murarishetti(smnv7),

Anudeep Pandiri(apz7c).

Introduction

In the developer's account on dev.twitter.com, we have created an app for getting an authorization to collect Twitter data. In the developer's console, we have generated a streaming URL with required filters. By using the above streaming URL, we have accessed Twitter REST API v1.1 using CURL and OAuth feature. A command line URL is generated which connects us to the Twitter database and saves the data to a local JSON file in the working directory.

The streaming URL generated in the developer console.

https://stream.twitter.com/1.1/statuses/sample.json

The Curl command generated by using the Test OAuth feature.

curl --get 'https://stream.twitter.com/1.1/statuses/sample.json' --header 'Authorization: OAuth oauth_consumer_key="TjIDZ6XQX6TZOq64EZ49SatYb", oauth_nonce="d54db403fb54cc9e5a10a92bb2741e6e", oauth_signature="GqHWmZKgD8YO6rX5HgGKRuMFWGQ%3D", oauth_signature_method="HMAC-SHA1", oauth_timestamp="1457754994", oauth_token="453746488-vDRGN511Pk3g3tSvOhpgIdSRErFjXP5fClexkpWp", oauth_version="1.0"' --verbose> tweet.txt

We have used the above Curl command in Linux to collect Twitter data.

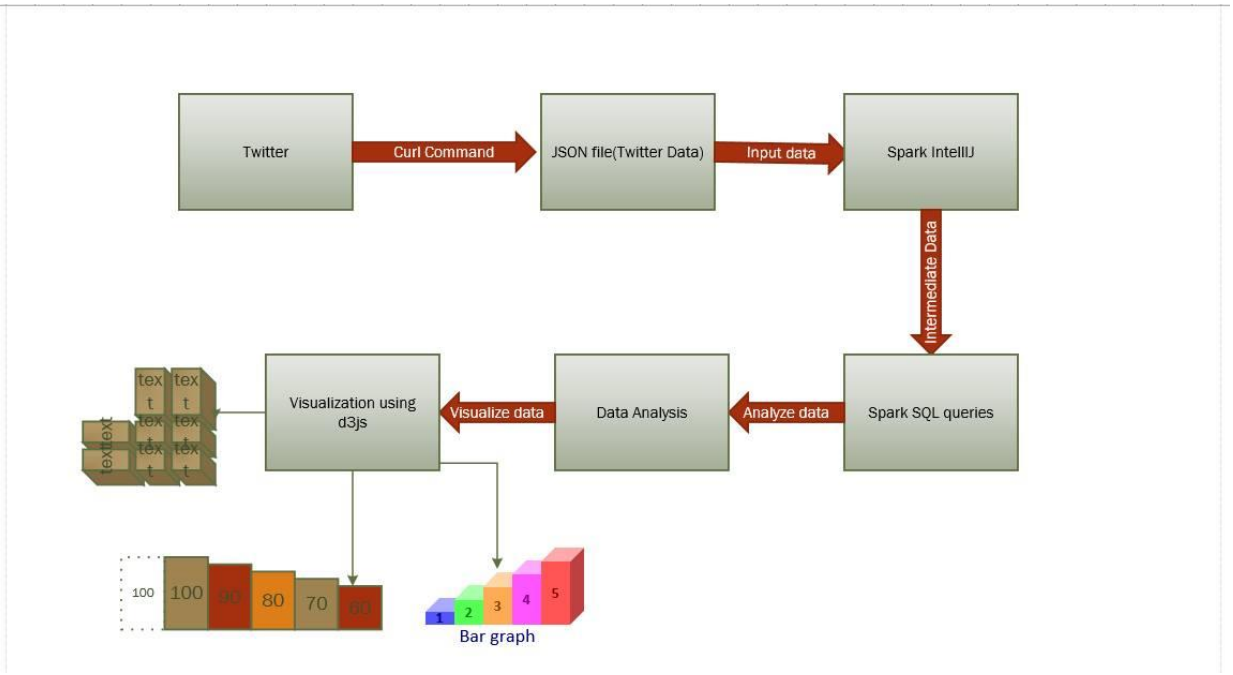


Fig: Architecture Diagram

IntelliJ IDEA

IntelliJ IDEA Community Edition is the open source version of IntelliJ IDEA, a premier IDE for JAVA, Scala, Groovy etc.

We have created a new project in IntelliJ with JAVA 1.7 SDK.

In the build.sbt file, we have provided Scala v2.11.7 and Spark v1.4.1. We have included the below Scala dependency libraries.

libraryDependencies ++= Seq(

"org.apache.spark" %% "spark-core" % "1.4.1" ,

"org.apache.spark" %% "spark-streaming" % "1.4.1",

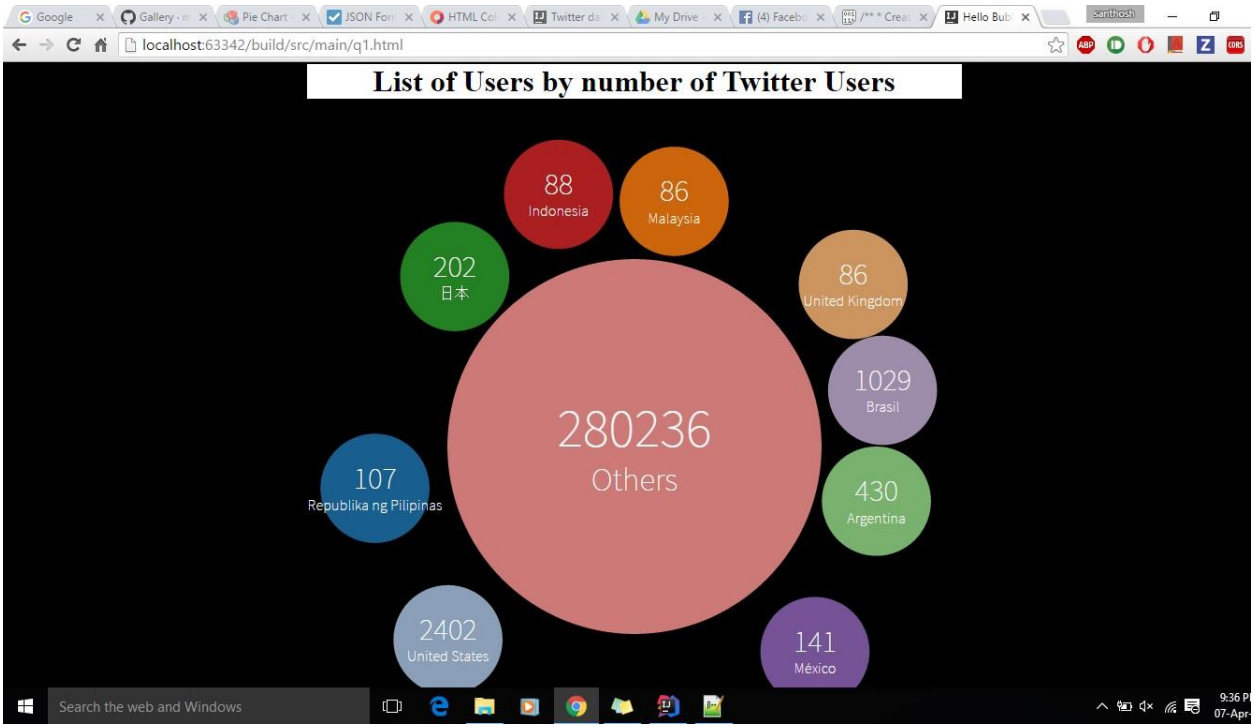
"org.apache.hadoop" % "hadoop-common" % "2.7.0" exclude ("org.apache.hadoop","hadoop-yarn-server- web-proxy"),

"org.apache.spark" %% "spark-mllib" % "1.4.1")

We have created a Scala object and started SparkSQL queries. The queries are as below.

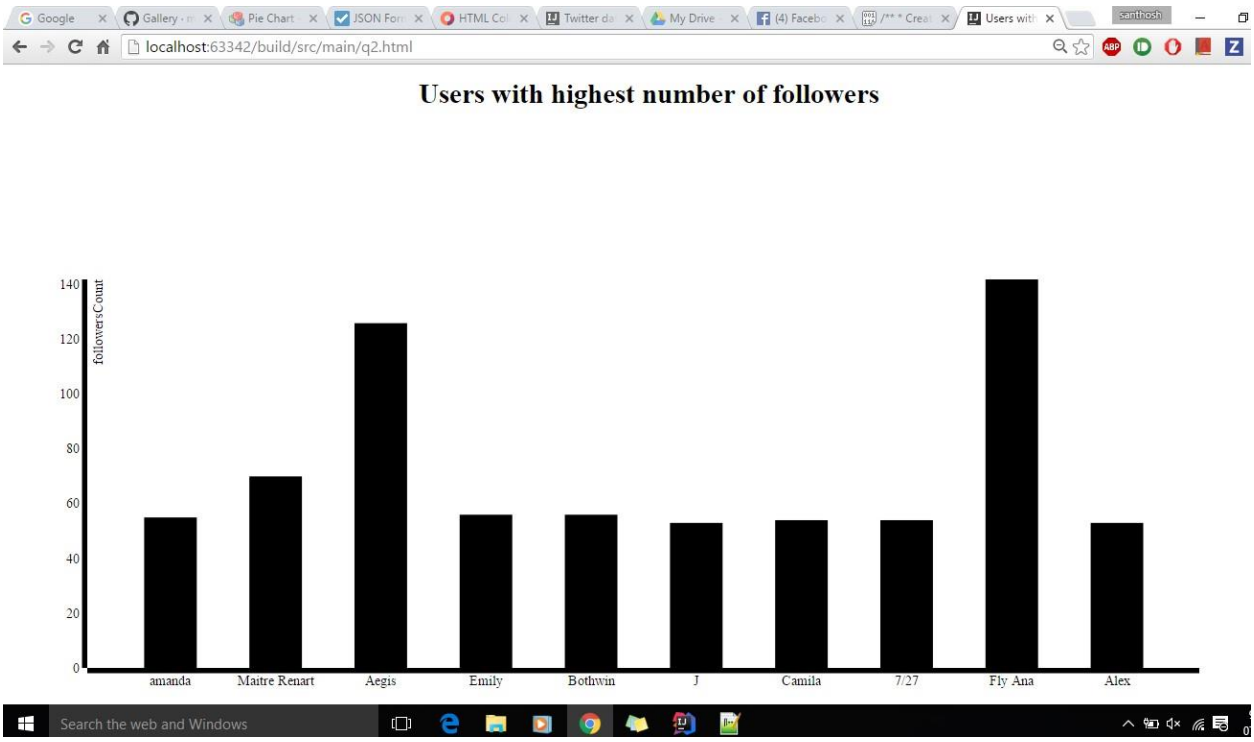
Query 1

```
val q1 = sqlContext.sql("select place.country, count(*) as countrycount from tweetsTable
GROUP by place.country order by countrycount desc limit 10")

q1.show()

q1.save("output1","json")
```



Query 2

```
val q2 = sqlContext.sql("select user.name, user.followers_count from tweetsTable where (user.name != NULL or user.name NOT LIKE '%.%' or user.name NOT LIKE '%,%'  or user.name != ',') order by followers_count desc limit 20")

q2.show()

q2.save("output2","json")
```



Query 3

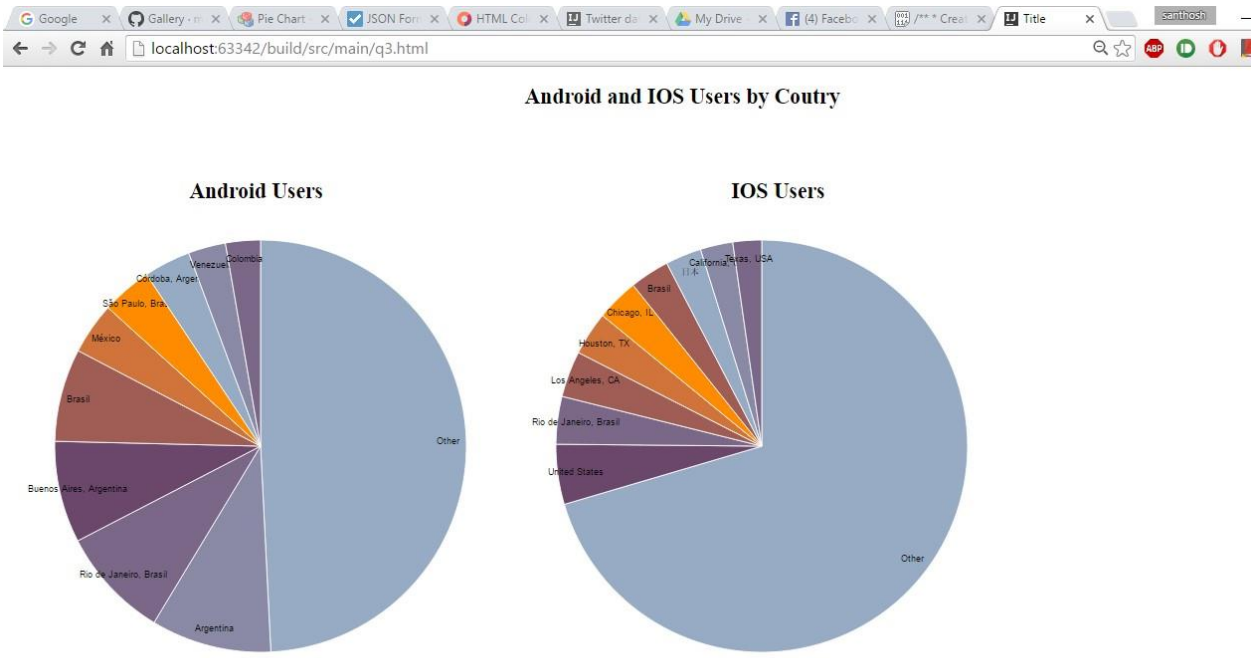
val q3 = sqlContext.sql("SELECT user.location, COUNT(*) AS android_count FROM tweetsTable WHERE source LIKE '%android%' GROUP BY user.location ORDER BY android_count DESC LIMIT 10")

q3.show()

q3.save("output3","json")

val q4 = sqlContext.sql("SELECT user.location, COUNT(*) AS ios_count FROM tweetsTable WHERE source LIKE '%iphone%' GROUP BY user.location ORDER BY ios_count DESC LIMIT 10")

q4.save("output4","json")

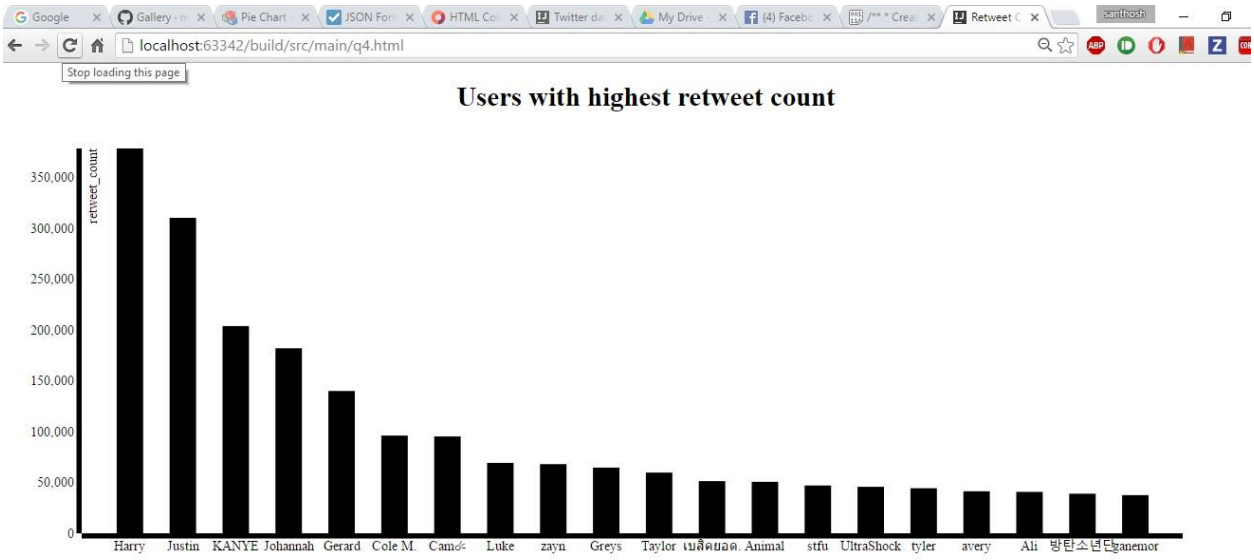


Query 4

valq5=sqlContext.sql("SELECTretweeted_status.user.name, max(retweeted_status.retweet_count) AS retweet_count FROM tweetsTable GROUP BY retweeted_status.user.name ORDER BY retweet_count DESC LIMIT 10")

q5.show()

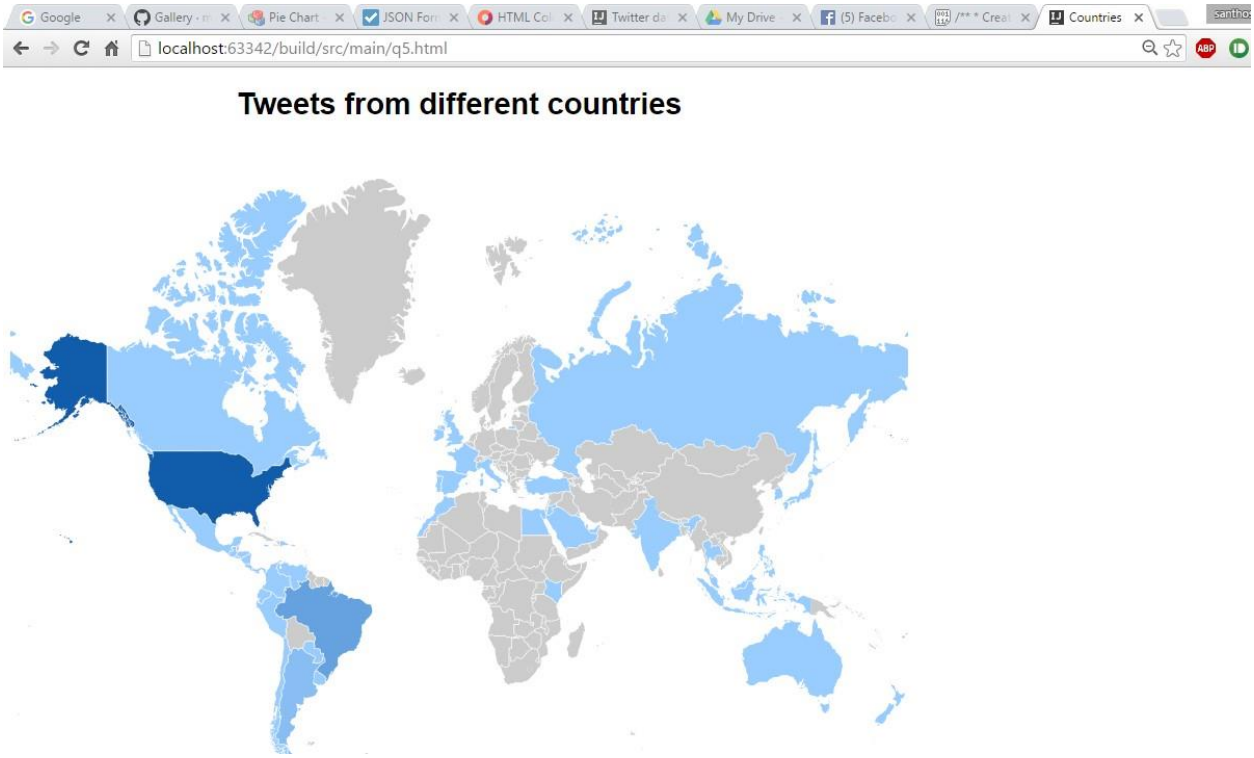
q5.save()



Query 5

val q6 = sqlContext.sql("SELECT place.country, count(*) as CountryCount from tweetsTable Group by place.country order by CountryCount desc limit 50")

q6.save("output6", "json")

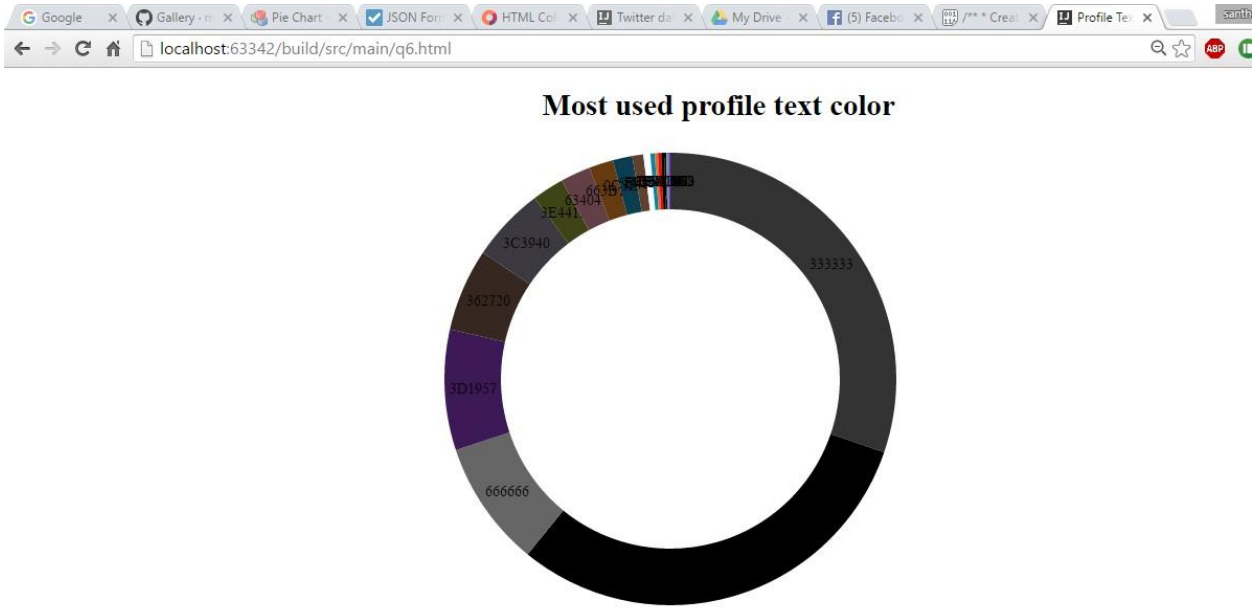


Query 6

val q7 = sqlContext.sql("SELECT user.profile_text_color, count(*) as textColorCount from tweetsTable Group by user.profile_text_color order by textColorCount desc limit 20")

q7.show()

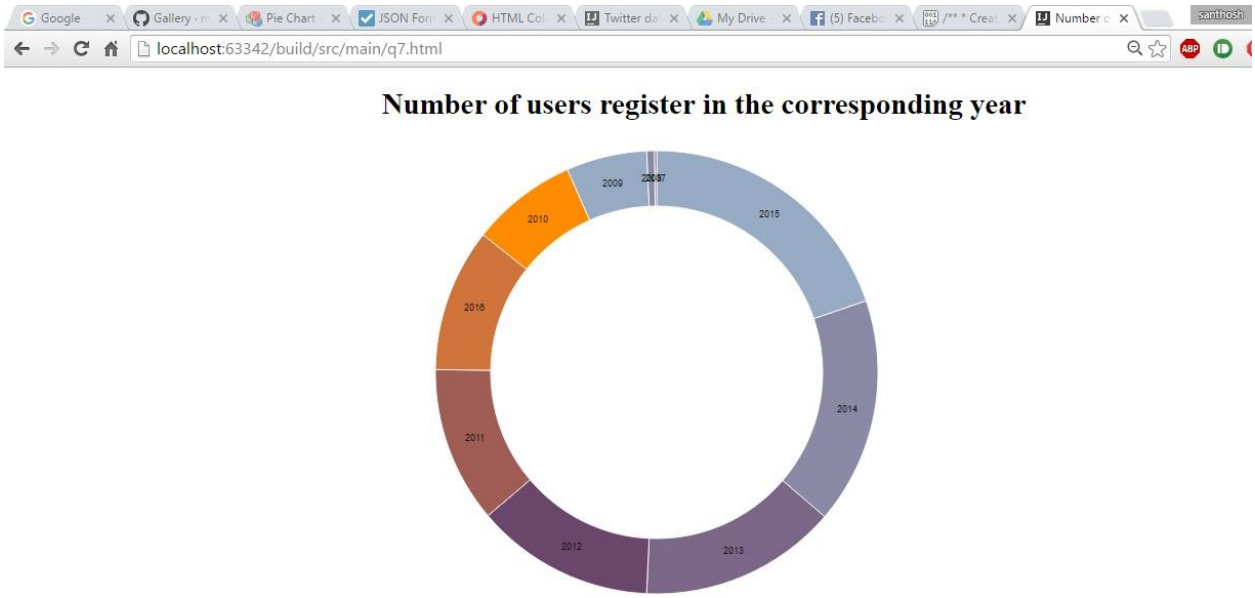
q7.save("output7", "json")



Query 7

val q8 = sqlContext.sql("SELECT SUBSTRING(user.created_at, 27,4), count(*) as userCount from tweetsTable group by SUBSTRING(user.created_at, 27,4) order by userCount desc limit 15")

q8.show()

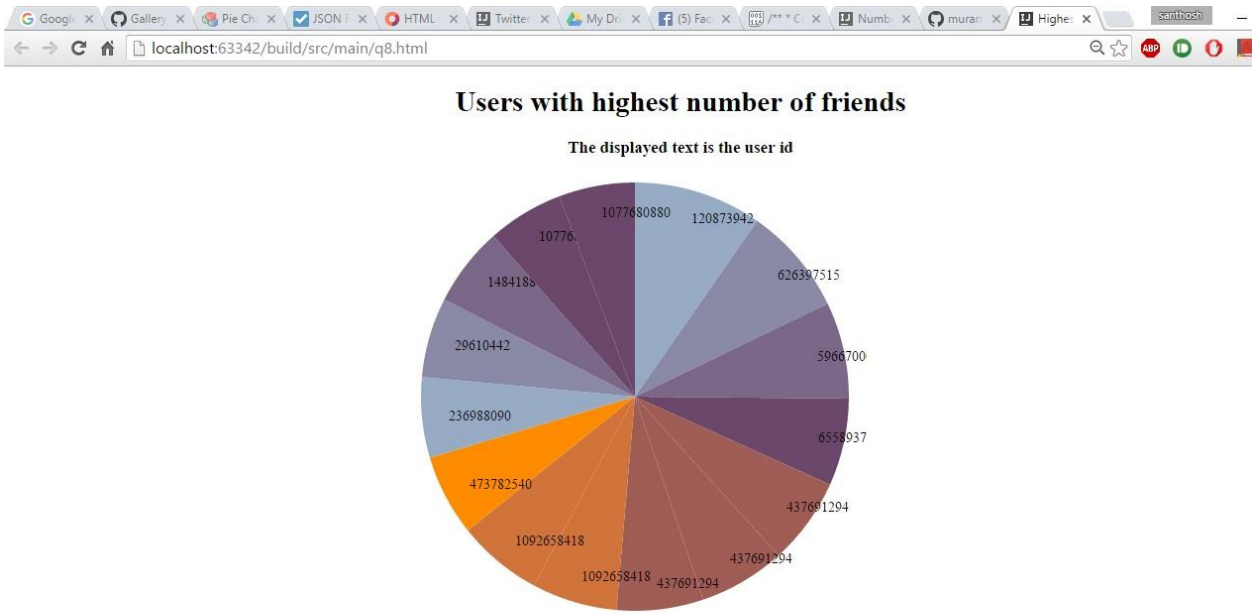


Query 8

val q9 = sqlContext.sql("SELECT user.id, user.friends_count as friendsCount from tweetsTable order by friendsCount desc limit 15")

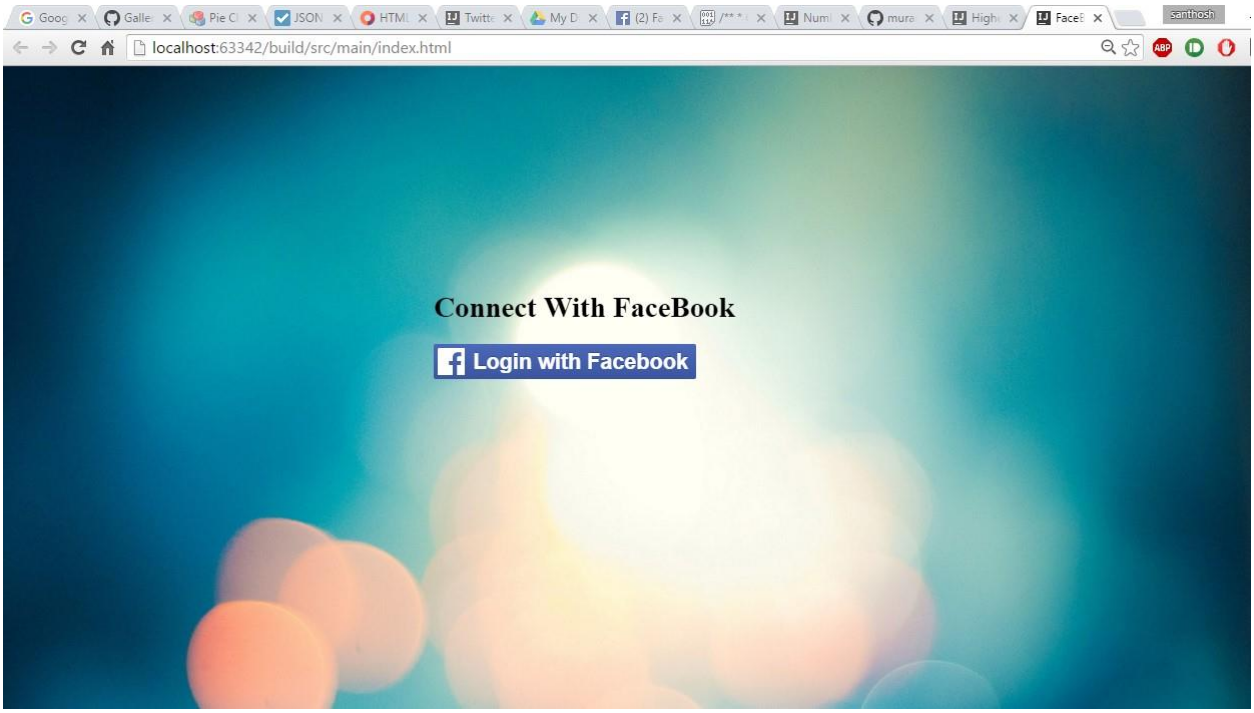
q9.show()

q9.save("output9", "json")



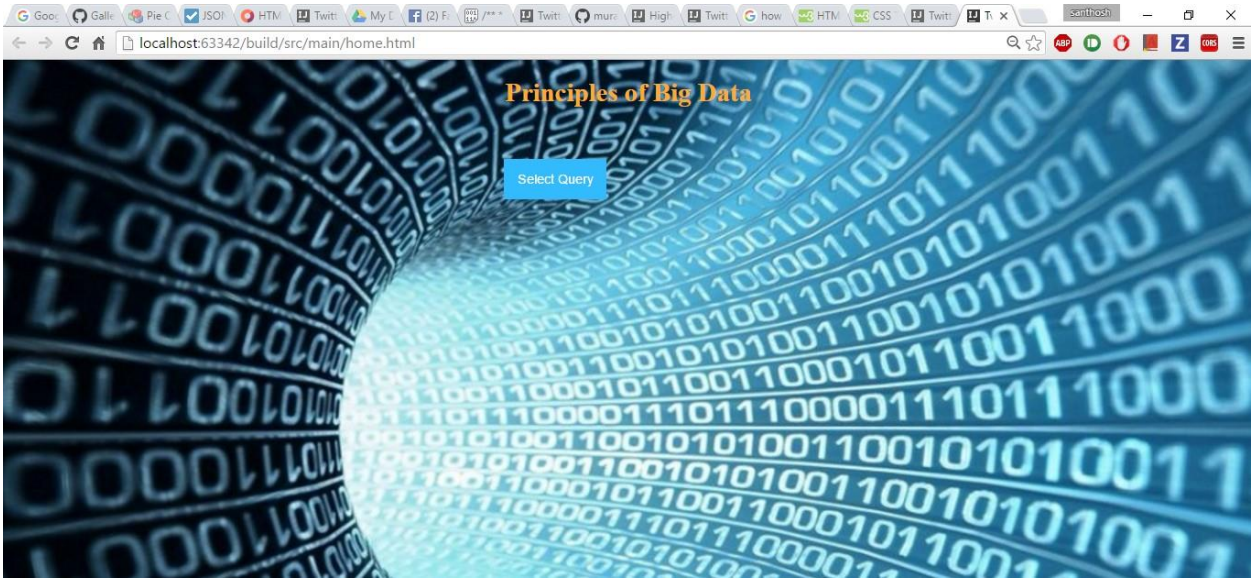
Login Page:

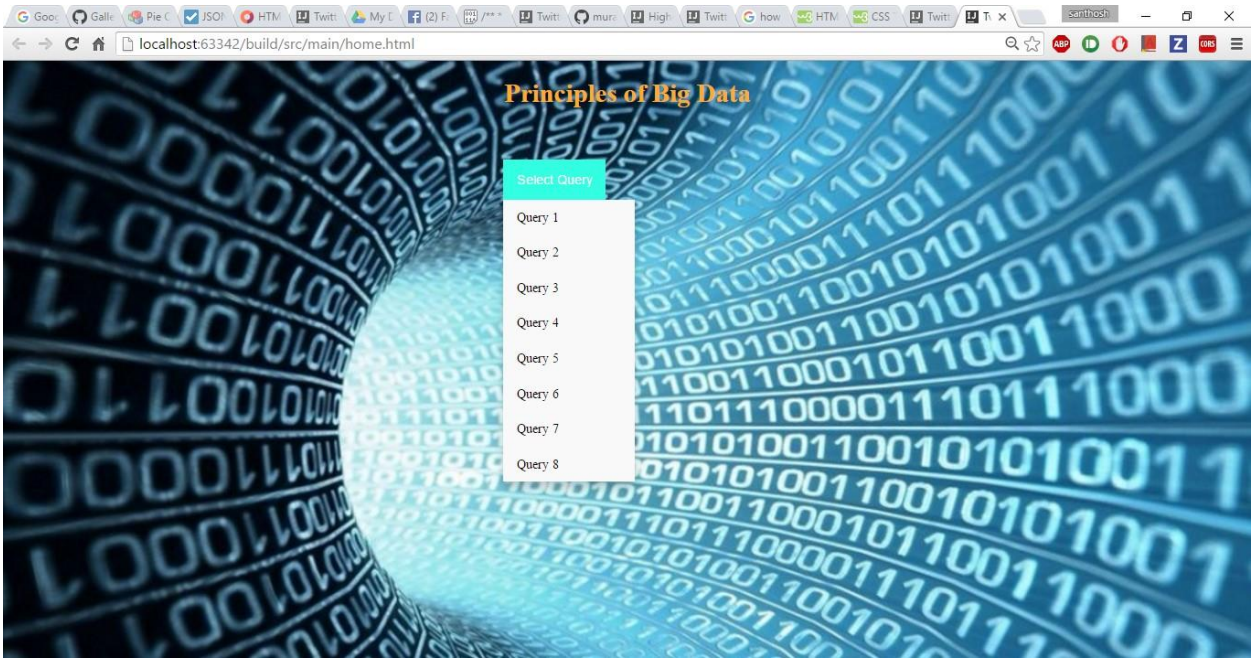
We have used Facebook API for logging in. After we enter correct facebook credentials, it will be redirected to home page.



Home Page:

There is a dropdown for selecting query.





After selecting the query, page will be redirected to corresponding graph page.

Twitter Data Analysis using IBM Bluemix

We have collected 1,00,200 tweets on **#Cinemas** using Insights for Twitter in IBM Bluemix and load the data to DashDB database. We have created Notebooks to write queries in Apache Spark and the visualizations are done using Python.

The below is the Scala code block to connect to the database and the store the data in a temporary table.

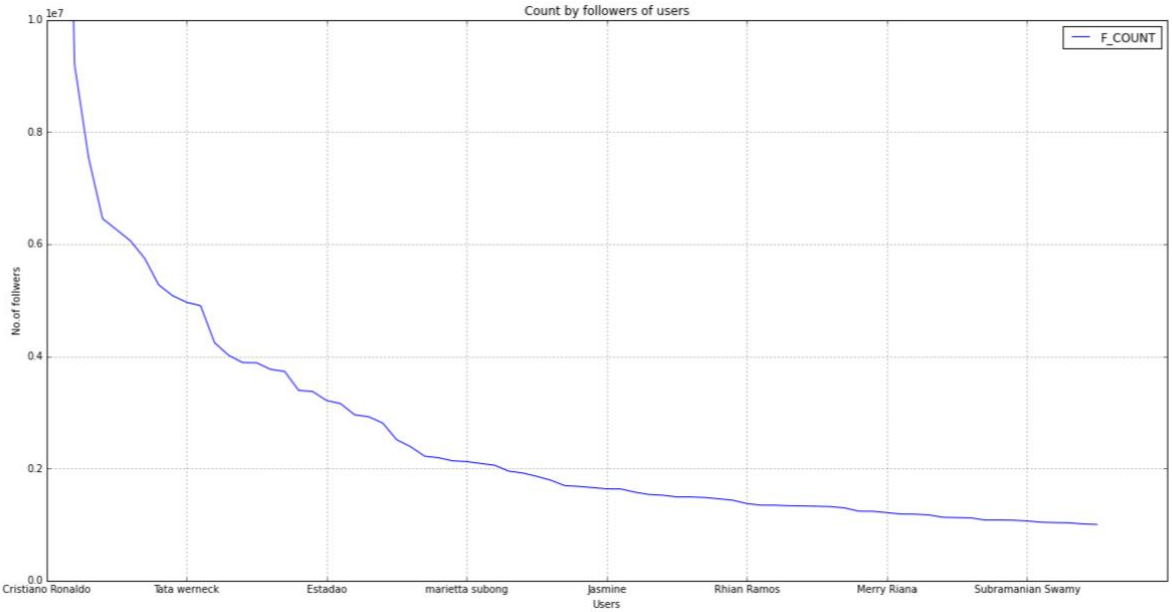
```
val sqlcontext = new org.apache.spark.sql.SQLContext(sc)

val dashdata = sqlcontext.load("jdbc", Map(

"url"->"jdbc:db2://dashdb-entry-yp-dal0907.services.dal.bluemix.net:50000/BLUDB:user=dash6999;password=tXtJ4ue9gvBf;",

"dbtable" -> "DASH6999.CINEMAS_TWEETS"))

dashdata.registerTempTable("TWEETS_TABLE")
```

The below are the queries used:

1. valQ1=sqlcontext.sql("SELECT USER_DISPLAY_NAME,MAX(USER_FOLLOWERS_COUNT) AS F_COUNT FROM TWEETS_TABLE GROUP by USER_DISPLAY_NAME ORDER BY F_COUNT DESC ")

   Scala Output:

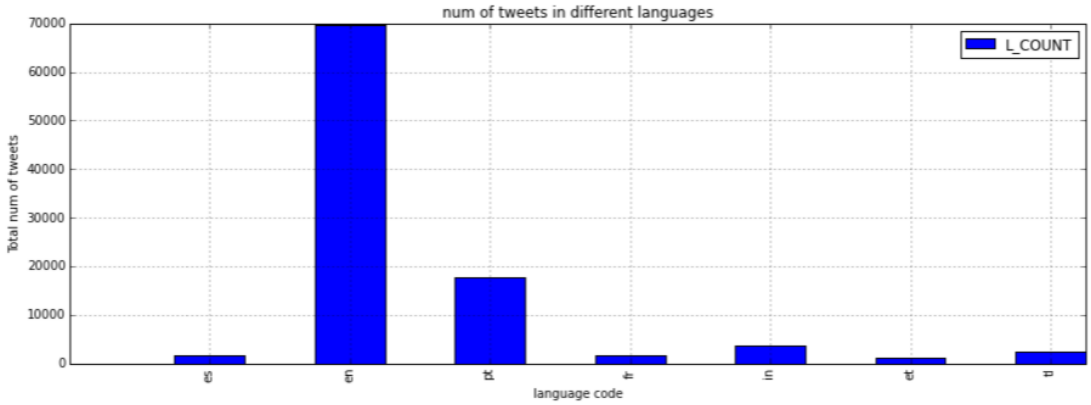
```
+------------------+--------+
| USER_DISPLAY_NAME| F_COUNT|
+------------------+--------+
| Cristiano Ronaldo|39506864|
|     One Direction|20634510|
|         luna maya| 9207377|
|Sabrina Sato Rahal| 7546246|
|    Danilo Gentili| 6456552|
|    Rafinha Bastos| 6258835|
|      bella thorne| 6057615|
|     Fabio Porchat| 5750875|
|    Times of India| 5276728|
|     Anna Kendrick| 5081638|
|      Tata werneck| 4964102|
|              Fiuk| 4903143|
|                G1| 4243041|
|     Portal R7.com| 4018408|
|  Folha de S.Paulo| 3888037|
| CNN International| 3884542|
|    Jack Whitehall| 3766587|
|    Jornal O Globo| 3730583|
|      ABS-CBN News| 3392508|
|          GMA News| 3371878|
+------------------+--------+
```



2. val Q2 = sqlcontext.sql("SELECT MESSAGE_LANGUAGE,COUNT(MESSAGE_ID) AS L_COUNT FROM TWEETS_TABLE GROUP by MESSAGE_LANGUAGE ORDER BY L_COUNT DESC ")

   Output:

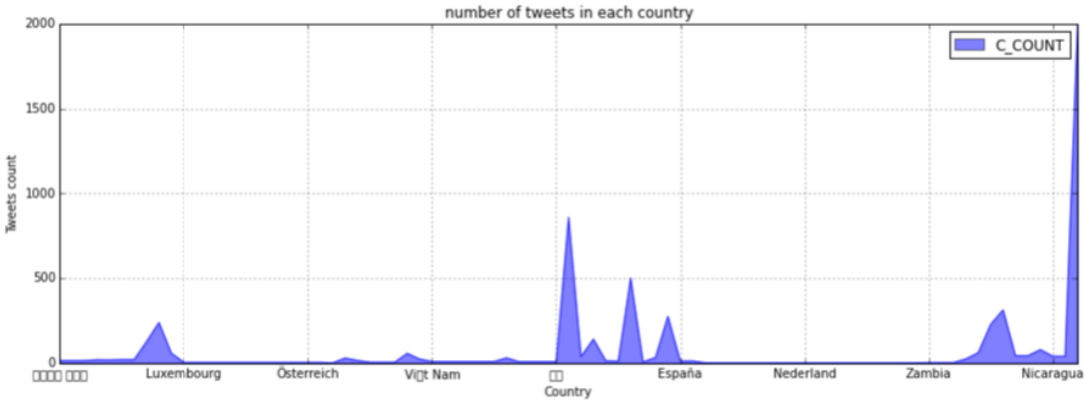
```
+----------------+-------+
|MESSAGE_LANGUAGE|L_COUNT|
+----------------+-------+
|              en|  69754|
|              pt|  17726|
|              in|   3752|
|              tl|   2353|
|              fr|   1778|
|              es|   1562|
|              et|   1060|
|              ja|    424|
|              de|    355|
|              it|    194|
|              ko|    101|
|              nl|     86|
|              da|     83|
|              zh|     82|
|              sk|     77|
|             und|     73|
|              lv|     63|
|              el|     60|
|              ht|     58|
|              ro|     56|
+----------------+-------+
```



3. val Q3 = sqlcontext.sql("SELECT MESSAGE_COUNTRY,COUNT(MESSAGE_ID) AS C_COUNT FROM TWEETS_TABLE GROUP by MESSAGE_COUNTRY ORDER BY C_COUNT DESC ")

   Output:

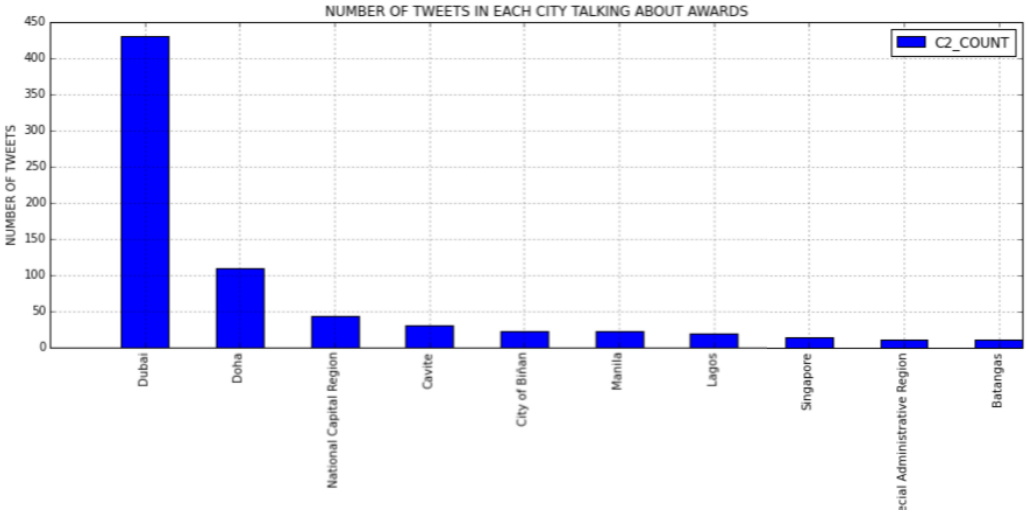
```
+--------------------+-------+
|     MESSAGE_COUNTRY|C_COUNT|
+--------------------+-------+
|                null|  94540|
|            Malaysia|   1997|
|       United States|    859|
|              Brasil|    501|
|               India|    313|
|Republika ng Pili...|    275|
|      United Kingdom|    239|
|Republic of the P...|    228|
|              Canada|    141|
|           Australia|    126|
|                  ??|     79|
|   ???????? ??????? |     61|
|           Indonesia|     58|
|              Italia|     57|
|  Dominican Republic|     43|
|         ???? ??????|     43|
|           Nicaragua|     40|
|              France|     40|
|            Portugal|     38|
|         Puerto Rico|     33|
+--------------------+-------+
```



4. val Q4 = sqlcontext.sql("SELECT USER_CITY, COUNT(MESSAGE_ID) AS C2_COUNT FROM TWEETS_TABLE WHERE (MESSAGE_BODY LIKE '%AWARD%' OR MESSAGE_BODY LIKE '%Award%' OR MESSAGE_BODY LIKE '%award%') AND USER_CITY IS NOT NULL GROUP BY USER_CITY ORDER BY C2_COUNT DESC")

   Output:

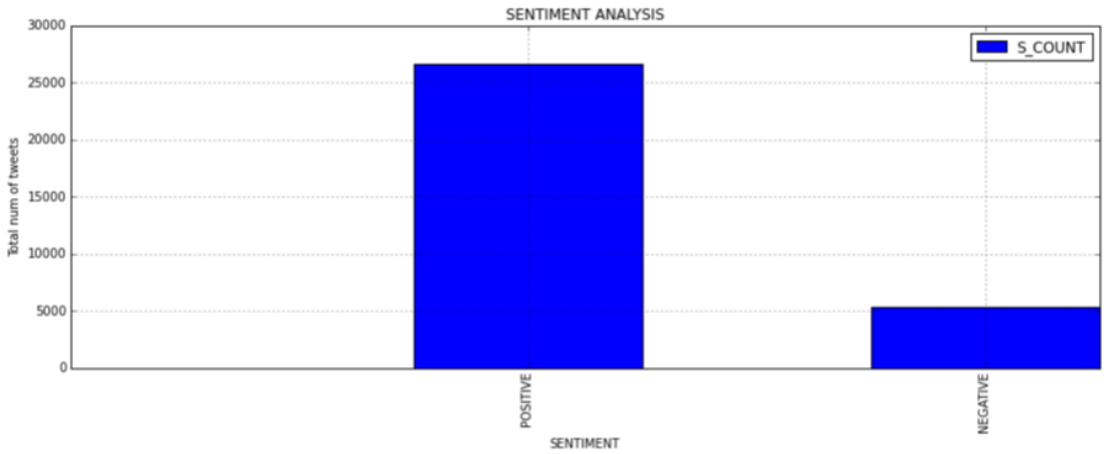
```
+--------------------+--------+
|           USER_CITY|C2_COUNT|
+--------------------+--------+
|               Dubai|     431|
|                Doha|     110|
|National Capital ...|      43|
|              Cavite|      31|
|       City of Bi?an|      23|
|              Manila|      22|
|               Lagos|      20|
|           Singapore|      14|
|Hong Kong Special...|      12|
|            Batangas|      11|
|               Davao|      10|
|              London|      10|
|          Pangasinan|       7|
|        Harrisonburg|       6|
|       New York City|       5|
|             Biliran|       5|
|           singapore|       5|
|           Cebu City|       5|
|              Iloilo|       5|
|              Ottawa|       4|
+--------------------+--------+
```



5. val Q5 = sqlcontext.sql("SELECT SENTIMENT_POLARITY,COUNT(MESSAGE_ID) AS S_COUNT FROM TWEETS_TABLE GROUP BY SENTIMENT_POLARITY ORDER BY S_COUNT DESC")

   Output:

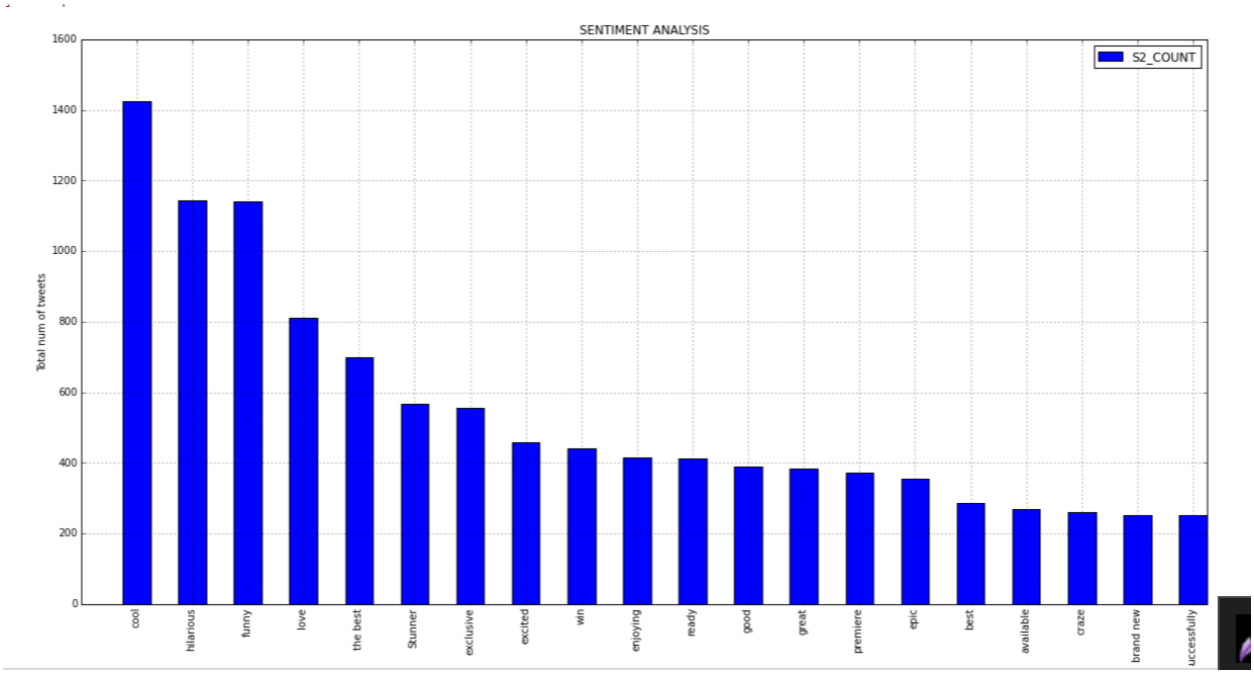
```
+------------------+-------+
|SENTIMENT_POLARITY|S_COUNT|
+------------------+-------+
|          POSITIVE|  26646|
|          NEGATIVE|   5345|
+------------------+-------+
```



6. val Q6 = sqlcontext.sql("SELECT SENTIMENT_TERM,COUNT(MESSAGE_ID) AS S2_COUNT FROM TWEETS_TABLE2 GROUP BY SENTIMENT_TERM ORDER BY S2_COUNT DESC")

   Output:

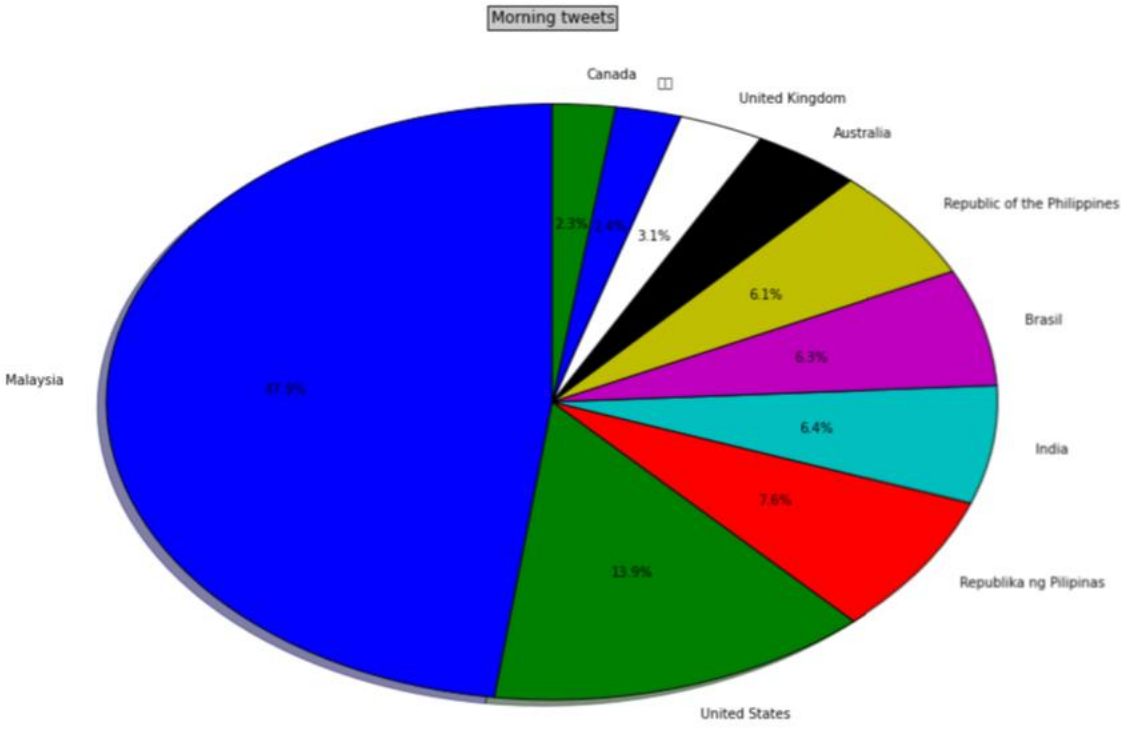
```
+--------------+--------+
|SENTIMENT_TERM|S2_COUNT|
+--------------+--------+
|          cool|    1424|
|     hilarious|    1144|
|         funny|    1141|
|          love|     810|
|      the best|     700|
|       Stunner|     568|
|     exclusive|     555|
|       excited|     458|
|           win|     440|
|      enjoying|     415|
|         ready|     412|
|          good|     388|
|         great|     384|
|      premiere|     372|
|          epic|     354|
|          best|     286|
|     available|     268|
|         craze|     261|
|     brand new|     252|
|  successfully|     251|
+--------------+--------+
```


SENTIMENT ANALYSIS
S2_COUNT
Total num of tweets
0
200
400
600
800
1000
1200
1400
1600
cool
hilarious
funny
love
the best
Stunner
exclusive
excited
win
enjoying
ready
good
great
premiere
epic
best
available
craze
brand new
uccessfully

7. val Q7 = sqlcontext.sql("SELECT MESSAGE_COUNTRY, COUNT(*) AS T_COUNT FROM TWEETS_TABLE WHERE ((MESSAGE_COUNTRY IS NOT NULL)AND (SUBSTRING(MESSAGE_POSTED_TIME ,12 ,2) BETWEEN 0 AND 12)) GROUP BY MESSAGE_COUNTRY ORDER BY T_COUNT DESC LIMIT 10")

   Output:

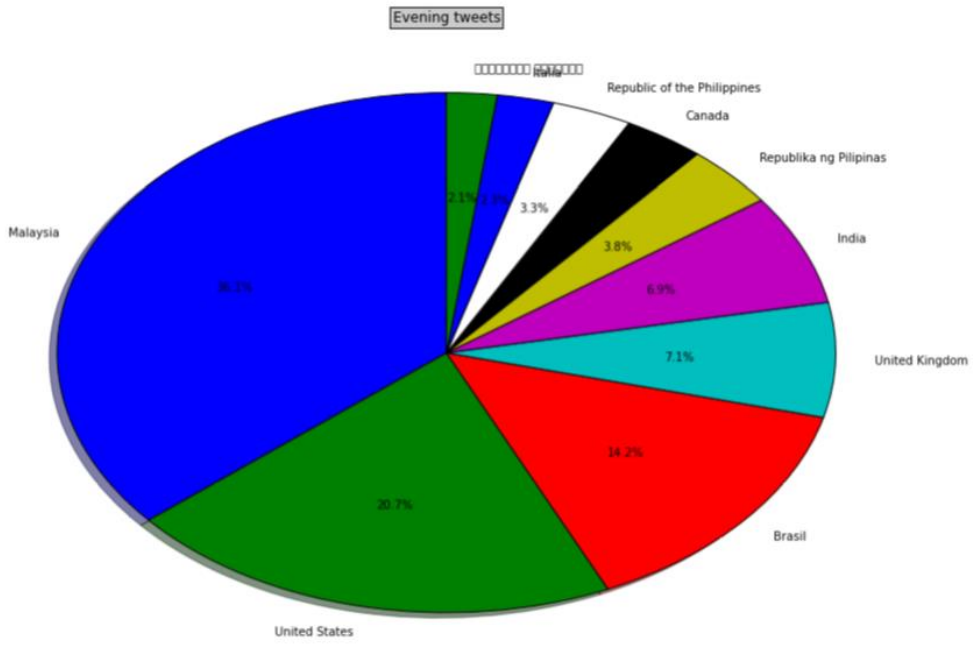
```
+--------------------+-------+
|     MESSAGE_COUNTRY|T_COUNT|
+--------------------+-------+
|            Malaysia|   1307|
|       United States|    379|
|Republika ng Pili...|    208|
|               India|    175|
|              Brasil|    173|
|Republic of the P...|    167|
|           Australia|    107|
|      United Kingdom|     84|
|                  ??|     65|
|              Canada|     62|
+--------------------+-------+
```



8. val Q8 = sqlcontext.sql("SELECT MESSAGE_COUNTRY, COUNT(*) AS T_COUNT FROM TWEETS_TABLE WHERE MESSAGE_COUNTRY IS NOT NULL AND SUBSTRING(MESSAGE_POSTED_TIME ,12 ,2) BETWEEN 12 AND 24 GROUP BY MESSAGE_COUNTRY ORDER BY T_COUNT DESC LIMIT 10")

   Output:

```
+--------------------+-------+
|     MESSAGE_COUNTRY|T_COUNT|
+--------------------+-------+
|            Malaysia|    846|
|       United States|    484|
|              Brasil|    332|
|      United Kingdom|    167|
|               India|    161|
|Republika ng Pili...|     90|
|              Canada|     79|
|Republic of the P...|     78|
|              Italia|     55|
|   ???????? ??????? |     49|
+--------------------+-------+
```



Github: https://github.com/murarishetty/bigdata-phase-2

Tweets file:
https://drive.google.com/open?id=0B4VHwW192C9HdWY2ak9OZEJDc2c

References:

https://github.com/mbostock/d3/wiki/Gallery

https://d3js.org

https://dev.twitter.com

https://developer.facebook.com

https://org.apache.com

https://spark.apache.com