UNIVERSITY OF MISSOURI-KANSAS CITY

PROJECT TITLE:-

COLLECTION OF TWITTER DATA USING TWITTER STREAMING API'S AND STORE, ANALYZE, AND VISUALIZE TWITTER'S TWEETS

SUBJECT:- Principles of Bigdata Management

PHASE-2

COLLECTION OF TWITTER DATA USING TWITTER API'S, ANALYZE USING SPARKSQL AND VISUALIZE USING JS CHARTS

1. Kantha Rao Patchava (16280105)
2. Premchand Lingamgunta (16279756)
3. Kranthi kiran Reddy Vanga (16274077)

Instructor

Dr. PRAVEEN RAO, Ph.D.

ABSTRACT

In this project collection of Twitter data by using Twitter Streaming API's in CSV format. The collected data pushed into the spark SQL and by using spark SQL code we can run interesting queries on that data Then save that output to local file system. Now visualize that output data by using JS charts by amcharts..

Problem Statement: Data analysis is done by collecting data manually. This is a frantic work. Social networking websites is helping in analyzing data easily. But the problem is it is unstructured data.

Existing System: In existing system we can store only structured data and low amount of data. In addition to this, it had scalability program.

Proposed System: In this Spark Sql can store high amount of data like petabytes. Now spark can store unstructured data as tables and gives fields automatically. So, we can run queries and we will get outputs respectively.

Introduction:

In Phase-1, we collected tweets in CSV format and then from that data we extracted Hashtags and URL's from the tweets which are CSV format by using Python. Then we stored the extracted data in text file and moved to HDFS and ran Word Count in Hadoop and Apache Spark and stored the output. So, finally the output file is copied to local file system.

In Phase-2, We are going to analyze the twitter data by using SparkSQL. We planned to write queries to get data from that unstructured data. Then we will develop visualization on that output of sparkSQL queries like pie charts, bar graphs and line Graph etc. We will do this whole process on tweets that related to mobile phones companies in different countries.

Analytic Queries:

1. Which mobile company has more tweets?
2. Which user tweeted more on which company?
3. On which day more tweets are done in a comapany?
4. Number of tweets of each company based on different language?
5. Number of tweets based on day of the week in different companies?
6. Number of tweets came for different user screen names who tweeted more tweets ?
7. Compare company name hashtags with blackboard tags
8. Popular languages used for tweeting tweets about mobile companies?
9. Account verification Tweets?
10. Top Tweet text and Retweet count?

Distribution of Work:

- All the team members divided each work and then together that work to visualize the queries.
- All the 10 different queries shared among the team members like each person has three and will be working accordingly.
- Documentation will be done by each member their work respectively.

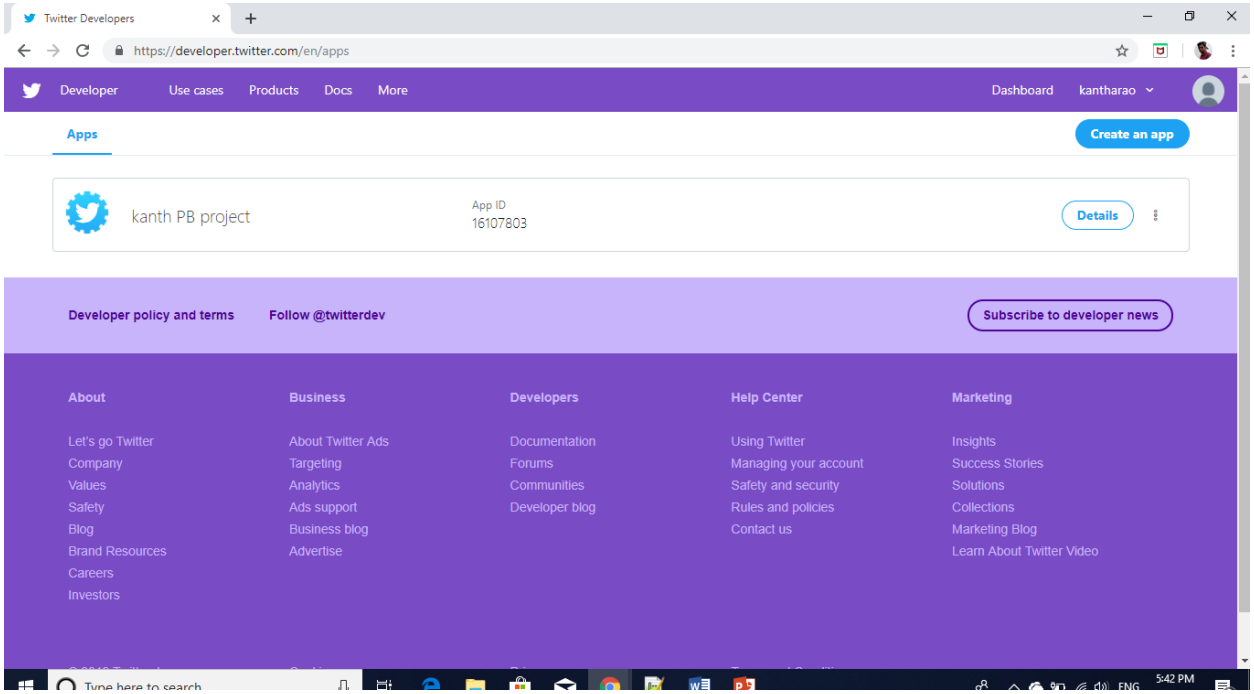
.

TWITTER DEVELOPER SIGN UP FOR TWEETS

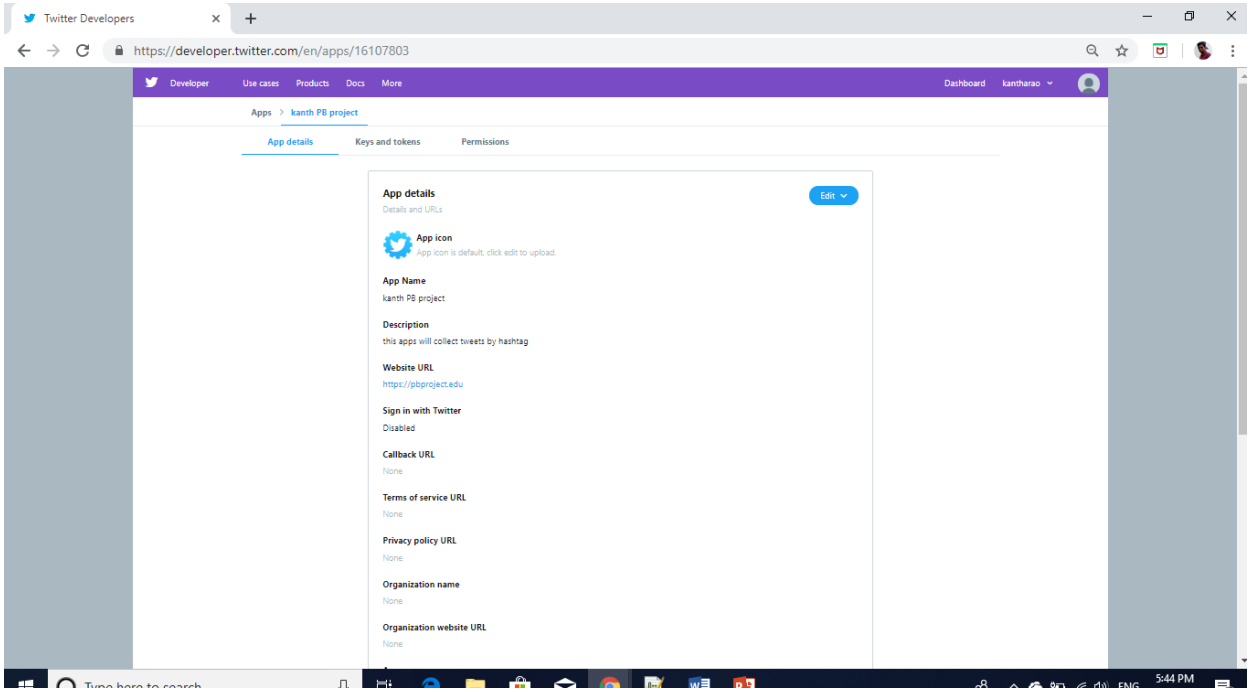
Before collection of tweets, it is important to sign in with twitter and join the documentation part with the following link :https://dev.twitter.com/resources/signup.

Create a Twitter Application with these we can create 4 keys – Consumer key, Consumer secret key, Access token and Access token secret which are later used for the collection of tweets

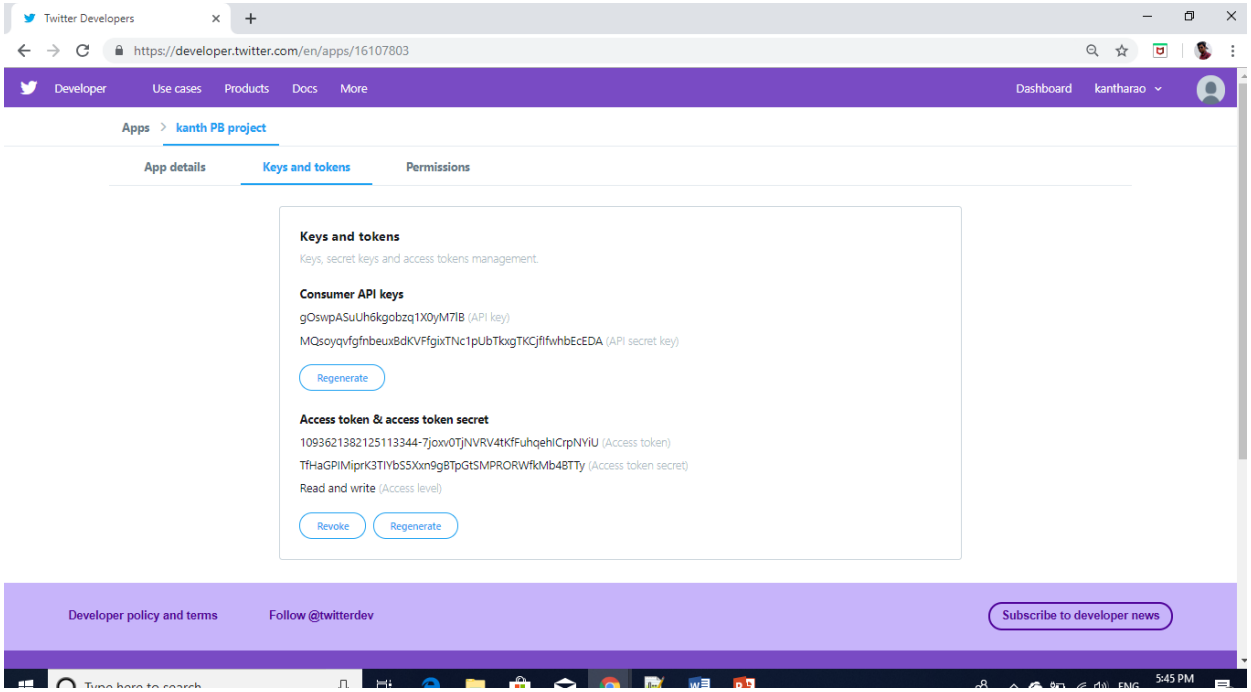
Project creation



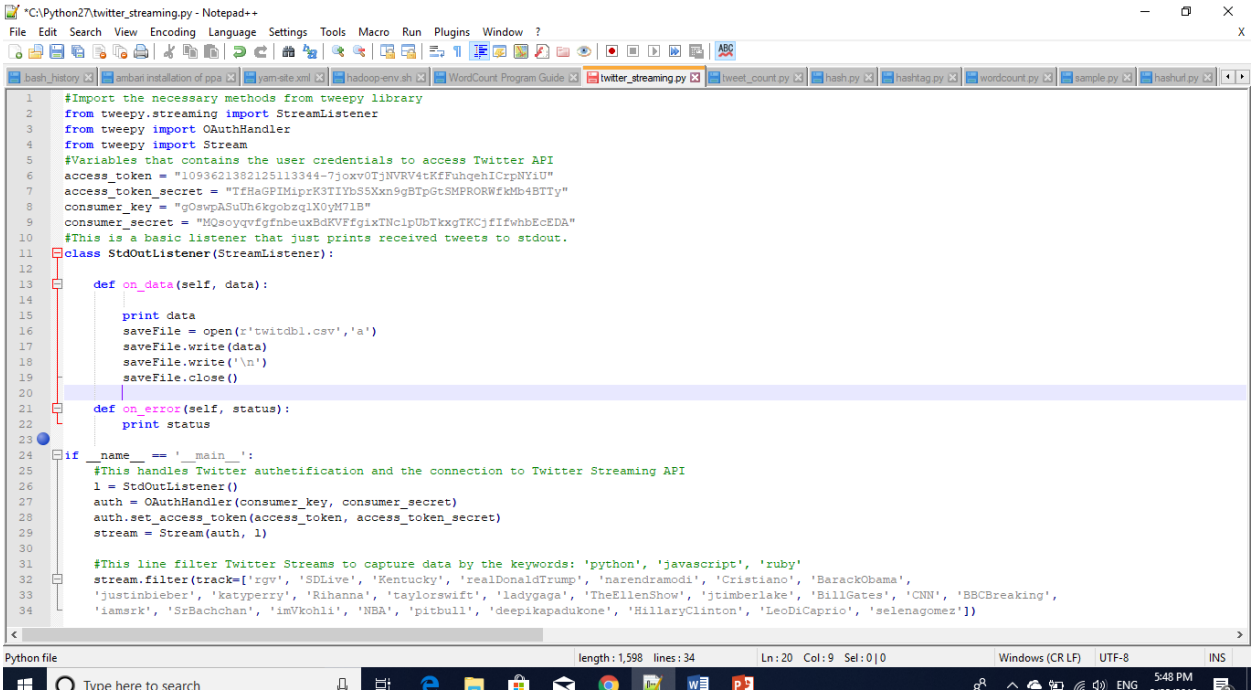
App details



Keys and tokens



Code to collect tweets



Source Code: twitter_straming.py

#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 = "1093621382125113344-7joxv0TjNVRV4tKfFuhqehICrpNYiU"

access_token_secret = "TfHaGPIMiprK3TIYbS5Xxn9gBTpGtSMPRORWfkMb4BTTy"

consumer_key = "gOswpASuUh6kgobzq1X0yM7lB"

consumer_secret = "MQsoyqvfgfnbeuxBdKVFfgixTNc1pUbTkxgTKCjfIfwhbEcEDA"

#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.csv','a')
                    saveFile.write(data)
                    saveFile.write('\n')
                    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:
    stream.filter(track=['apple', 'iphone', 'hauwai', 'gionee', 'nokia', 'redmi', 'coolpad', 'microsoft mobiles', 'karbon', 'celkon', ' HTC', 'motorola', 'lenovo', 'sony', 'LG mobiles', ' one plus', ' Samsung mobiles', ' vivo', 'oppo', 'honor'])
```

SPARK PROGRAMMING:

Steps: Start of SparkSQL and Loading of twitter data into SparkSQL.

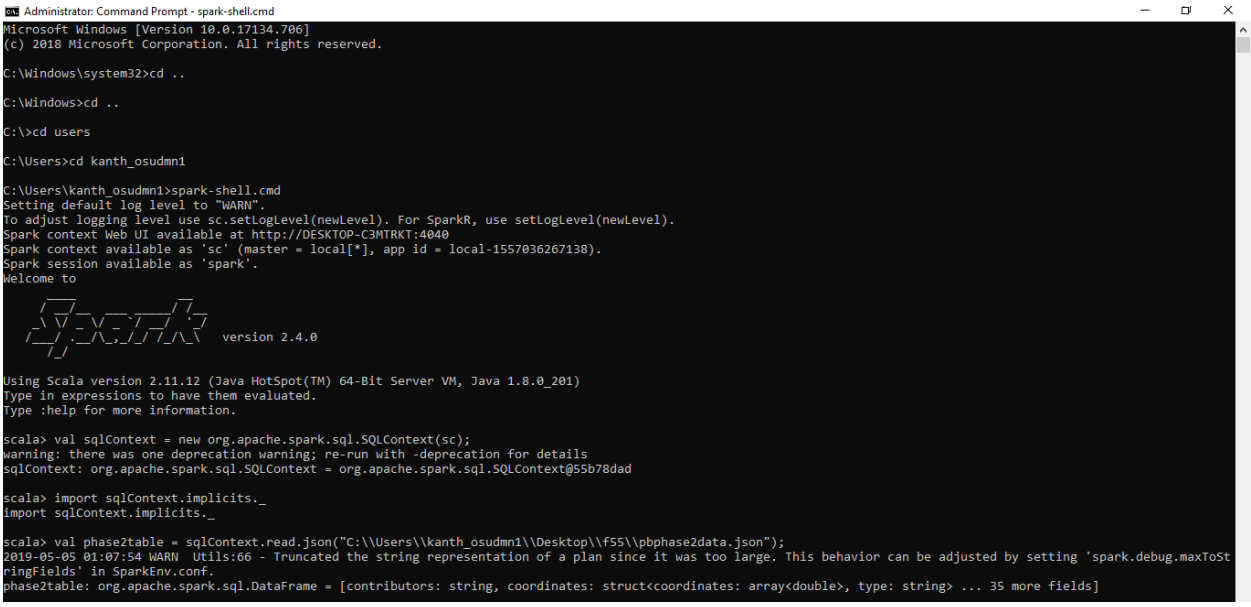
Code:

```
C:\Users\kanth_osudmn1>spark-shell.cmd
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://DESKTOP-C3MTRKT:4040
Spark context available as 'sc' (master = local[*], app id = local-1557103265492).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.0
      /_/

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_201)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val sqlContext = new org.apache.spark.sql.SQLContext(sc);
warning: there was one deprecation warning; re-run with -deprecation for details
sqlContext: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@1df8e685

scala> import sqlContext.implicits._
import sqlContext.implicits._
scala> val phase2table =
sqlContext.read.json("C:\\Users\\kanth_osudmn1\\Desktop\\f55\\pbphase2data.json");
```

2019-05-05 19:44:14 WARN Utils:66 - Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.debug.maxToStringFields' in SparkEnv.conf.

phase2table: org.apache.spark.sql.DataFrame = [contributors: string, coordinates: struct<coordinates: array<double>, type: string> ... 35 more fields]

SparkSQL initialization ouput screen:



```
Administrator: Command Prompt - spark-shell.cmd
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd ..

C:\Windows>cd ..

C:\>cd users

C:\Users>cd kanth_osudmn1

C:\Users\kanth_osudmn1>spark-shell.cmd
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://DESKTOP-C3MTRKT:4040
Spark context available as 'sc' (master = local[*], app id = local-1557036267138).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.0
      /_/

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_201)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val sqlContext = new org.apache.spark.sql.SQLContext(sc);
warning: there was one deprecation warning; re-run with -deprecation for details
sqlContext: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@55b78dad

scala> import sqlContext.implicits._
import sqlContext.implicits._

scala> val phase2table = sqlContext.read.json("C:\\Users\\kanth_osudmn1\\Desktop\\f55\\pbphase2data.json");
2019-05-05 01:07:54 WARN  Utils:66 - Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.debug.maxToSt
ringFields' in SparkEnv.conf.
phase2table: org.apache.spark.sql.DataFrame = [contributors: string, coordinates: struct<coordinates: array<double>, type: string> ... 35 more fields]
```

Query1:

scala> val query1 = sqlContext.sql("SELECT substring(user.created_at,5,3) as month, count(user.id) from pb2table group by month");

2019-05-05 01:10:31 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();

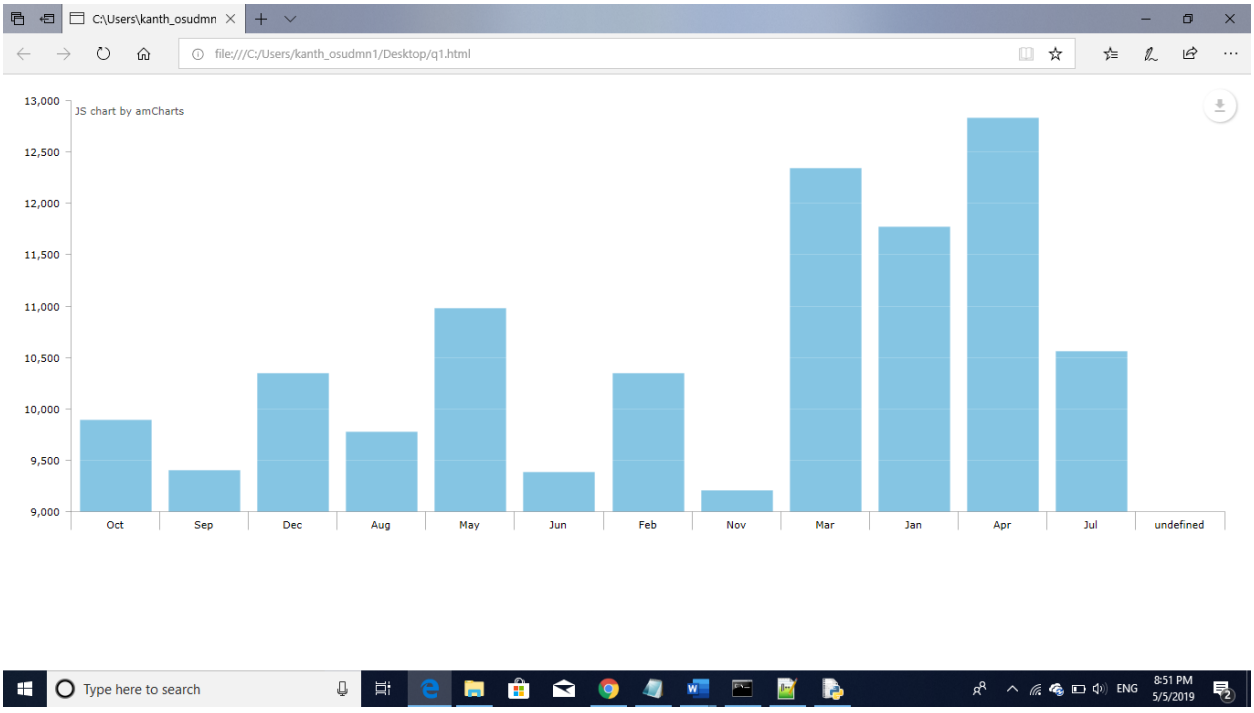
```
+-----+----------------------+

|month|count(user.id AS `id`)|

+-----+----------------------+

|  Oct|                 9891|
|  Sep|                 9400|
|  Dec|                10342|
|  Aug|                 9775|
|  May|                 10980|
|  Jun|                9379|
|  Feb|                10344|
|  Nov|                 9205|
|  Mar|                 12345|
|  Jan|                11774|
|  Apr|                12832|
|  Jul|               10558|
+-----+----------------------+
```

scala>query1.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q1.csv");

query1 output screen:

Administrator: Command Prompt - spark-shell.cmd

```
scala> phase2table.registerTempTable("pb2table");
warning: there was one deprecation warning; re-run with -deprecation for details

scala> val query1 = sqlContext.sql("SELECT substring(user.created_at,5,3) as month, count(user.id) from pb2table group by month");
2019-05-05 01:10:31 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();
+-----+---------------------+
|month|count(user.id AS `id`)|
+-----+---------------------+
|  Oct|                 9891|
|  Sep|                 9400|
|  Dec|                10342|
|  Aug|                 9775|
|  May|                10980|
|  Jun|                 9379|
|  Feb|                10344|
|  Nov|                 9205|
|  Mar|                12345|
|  Jan|                11774|
|  Apr|                12832|
|  Jul|                10558|
+-----+---------------------+


scala> query1.coalesce(1).write.format(com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\query1.csv");
<console>:1: error: identifier expected but string literal found.
query1.coalesce(1).write.format(com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\query1.csv");
                                                          ^
<console>:1: error: unclosed string literal
query1.coalesce(1).write.format(com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\query1.csv");
                                                                                                                   ^

scala> query1.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q1.csv");
```



Query2:

scala> val query2=sqlContext.sql("SELECT count(*) as count, user.name from pb2table where user.name is not null group by user.name order by count desc limit 10");

query2: org.apache.spark.sql.DataFrame = [count: bigint, name: string]

```
scala> query2.show();
+-----+--------------+
|count|          name|
+-----+--------------+
|  226|    reza islam|
|  189|             .|
|  135|             ?|
|  102|Ashraful Islam|
|   99|             ?|
|   91|         Aizen|
|   89|         Kevin|
|   80|             ?|
|   75|  AssetPodcast|
|   73|    Box Azteca|
+-----+--------------+
```

scala>query2.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q2.csv");

Query2 output screen:

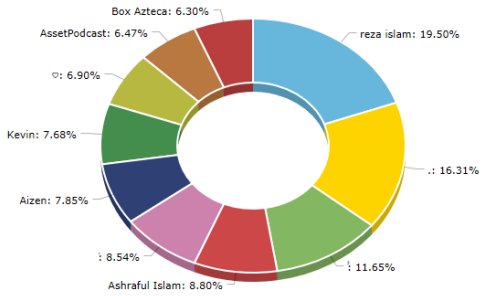
Administrator: Command Prompt - spark-shell.cmd

```
scala> val query2=sqlContext.sql("SELECT count(*) as count, user.name from pb2table where user.name is not null group by user.name order by count desc limit 10");
query2: org.apache.spark.sql.DataFrame = [count: bigint, name: string]

scala> query2.show();
+-----+--------------+
|count|          name|
+-----+--------------+
|  226|    reza islam|
|  189|             .|
|  135|             ?|
|  102|Ashraful Islam|
|   99|             ?|
|   91|         Aizen|
|   89|         Kevin|
|   80|             ?|
|   75|  AssetPodcast|
|   73|    Box Azteca|
+-----+--------------+


scala> query2.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q2.csv");
```

C:\Users\kanth_osudmn

file:///C:/Users/kanth_osudmn1/Desktop/q2.html

JS chart by amCharts



Type here to search

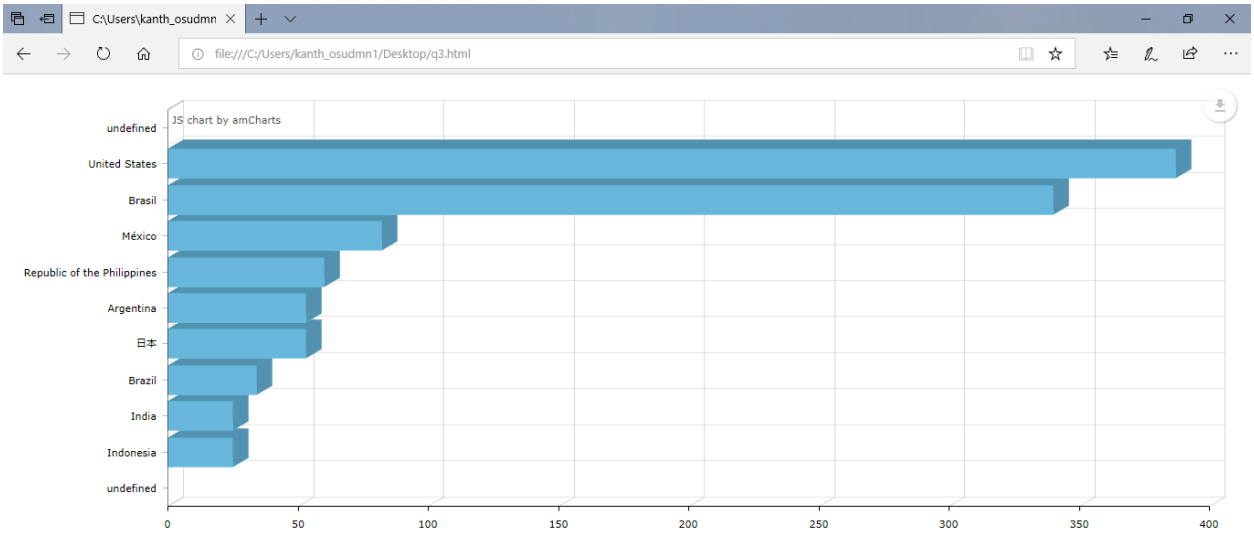
ENG 8:54 PM 5/5/2019

Query3:

```
scala> val query3 = sqlContext.sql("SELECT place.country,count(*) AS count FROM pb2table GROUP BY place.country ORDER BY count DESC limit 10");

query3: org.apache.spark.sql.DataFrame = [country: string, count: bigint]


scala> query3.show();
+--------------------+------+
|             country| count|
+--------------------+------+
|                null|125471|
|       United States|   387|
|              Brasil|   340|
|              México|    82|
|Republic of the P...|    60|
|           Argentina|    53|
|                 ??|    53|
|              Brazil|    34|
|               India|    25|
|           Indonesia|    25|
+--------------------+------+


scala>query3.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q3.csv");
```

Query3 output screen:

```
scala> val query3 = sqlContext.sql("SELECT place.country,count(*) AS count FROM pb2table GROUP BY place.country ORDER BY count DESC limit 10");
query3: org.apache.spark.sql.DataFrame = [country: string, count: bigint]

scala> query3.show();
+--------------------+------+
|             country| count|
+--------------------+------+
|                null|125471|
|       United States|   387|
|              Brasil|   340|
|              México|    82|
|Republic of the P...|    60|
|           Argentina|    53|
|                ??|    53|
|             Brazil|    34|
|              India|    25|
|           Indonesia|    25|
+--------------------+------+


scala> query3.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q3.csv");
```



Query4:

scala> val query4 = sqlContext.sql("SELECT user.name, user.followers_count, user.lang FROM pb2table WHERE text like '%sport%' order by user.followers_count desc limit 15");

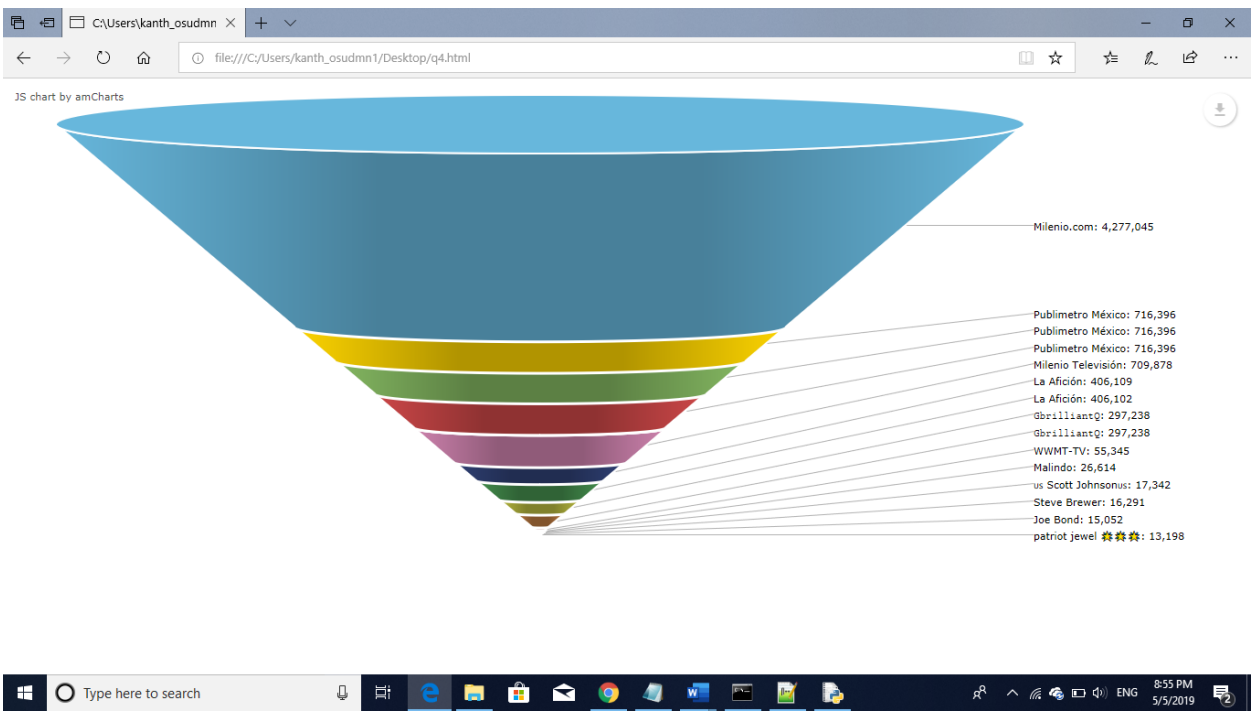
query4: org.apache.spark.sql.DataFrame = [name: string, followers_count: bigint ... 1 more field]

```
scala> query4.show();
+--------------------+---------------+----+
|               name|followers_count|lang|
+--------------------+---------------+----+
|        Milenio.com|        4277045|  es|
|   Publimetro México|         716396|  es|
|   Publimetro México|         716396|  es|
|   Publimetro México|         716396|  es|
|  Milenio Televisión|         709878|  es|
|         La Afición|         406109|  es|
|         La Afición|         406102|  es|
|?????????...|         297238|  en|
|?????????...|         297238|  en|
|            WWMT-TV|          55345|  en|
|            Malindo|          26614|  en|
|?? Scott Johnso...|          17342|  en|
|        Steve Brewer|          16291|  en|
|           Joe Bond|          15052|  en|
|patriot jewel ???|          13198|  en|
+--------------------+---------------+----+
```

scala>query4.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q4.csv");

Query4 output screen:

```
Administrator: Command Prompt - spark-shell.cmd

scala> val query4 = sqlContext.sql("SELECT user.name, user.followers_count, user.lang FROM pb2table WHERE text like '%sport%' order by user.followers_count desc limit 1
5");
query4: org.apache.spark.sql.DataFrame = [name: string, followers_count: bigint ... 1 more field]

scala> query4.show();
+--------------------+---------------+----+
|                name|followers_count|lang|
+--------------------+---------------+----+
|         Milenio.com|        4277045|  es|
|   Publimetro México|         716396|  es|
|   Publimetro México|         716396|  es|
|   Publimetro México|         716396|  es|
|  Milenio Televisión|         709878|  es|
|          La Afición|         406109|  es|
|          La Afición|         406102|  es|
|??????????...|         297238|  en|
|??????????...|         297238|  en|
|             WWMT-TV|          55345|  en|
|             Malindo|          26614|  en|
|?? Scott Johnso...|         17342|  en|
|        Steve Brewer|          16291|  en|
|            Joe Bond|          15052|  en|
|patriot jewel ???|         13198|  en|
+--------------------+---------------+----+


scala> query4.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q4.csv");
```



Query5:

```
scala> val Query5 = sqlContext.sql("SELECT user.lang, count(*) AS count FROM pb2table 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();
+-----+-----+
| lang|count|
+-----+-----+
|   en|57217|
|   es|27362|
|   pt|20137|
|   ja|13347|
|   th| 2770|
|   id| 1326|
|   ko| 1186|
|   ar|  956|
|   fr|  723|
|en-gb|  329|
+-----+-----+


scala>Query5.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\
Desktop\\phase2\\q5.csv");
```

Query5 output screen:

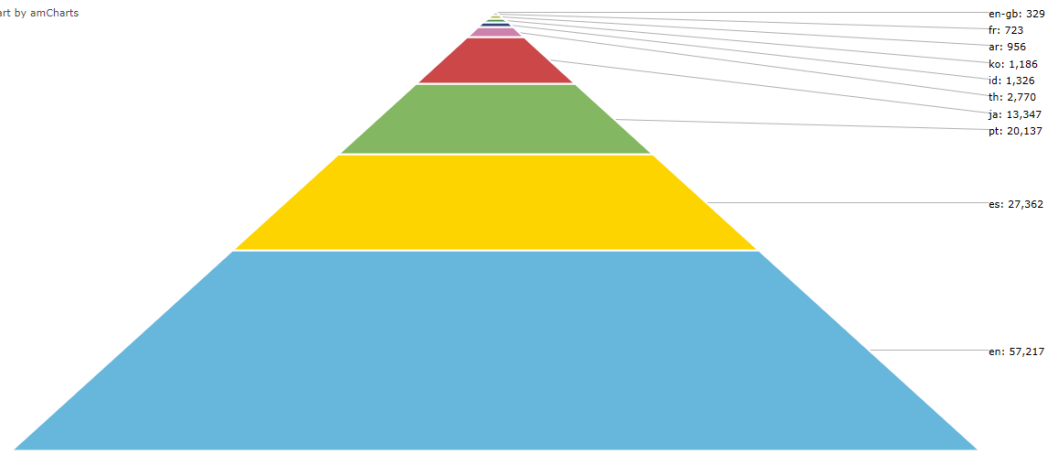
Administrator: Command Prompt - spark-shell.cmd

```
scala> query4.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q4.csv");

scala> val Query5 = sqlContext.sql("SELECT user.lang, count(*) AS count FROM pb2table 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();
+-----+-----+
| lang|count|
+-----+-----+
|   en|57217|
|   es|27362|
|   pt|20137|
|   ja|13347|
|   th| 2770|
|   id| 1326|
|   ko| 1186|
|   ar|  956|
|   fr|  723|
|en-gb|  329|
+-----+-----+


scala> Query5.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q5.csv");
```

C:\Users\kanth_osudmn

file:///C:/Users/kanth_osudmn1/Desktop/q5.html

JS chart by amCharts



Type here to search

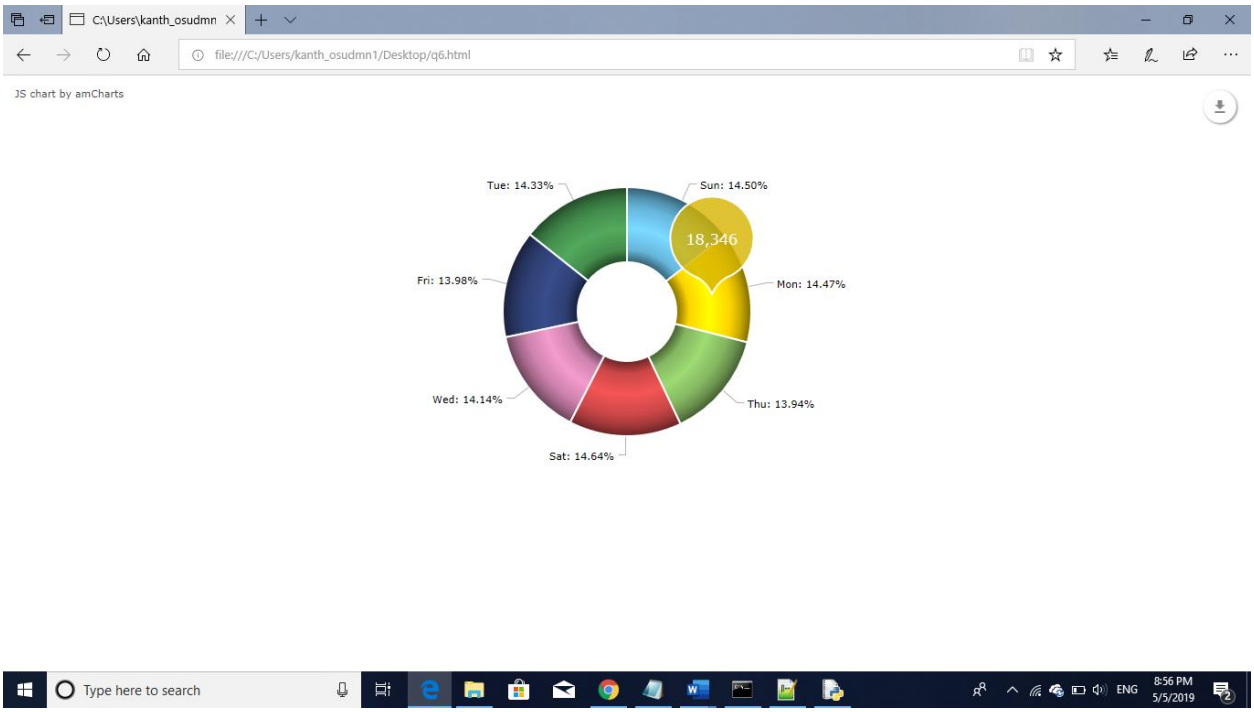
ENG 8:56 PM 5/5/2019

Query6:

```
scala> val query6=sqlContext.sql("SELECT substring(user.created_at,1,3) as day,count(*) as count from pb2table group by day");

query6: org.apache.spark.sql.DataFrame = [day: string, count: bigint]


scala> query6.show();

+---+-----+

|day|count|

+---+-----+

|Sun|18395|

|Mon|18346|

|Thu|17682|

|Sat|18569|

|Wed|17932|

|Fri|17725|

|Tue|18176|

+---+-----+


scala>query6.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q6.csv");
```

Query6 output screen:

```
scala> val query6=sqlContext.sql("SELECT substring(user.created_at,1,3) as day,count(*) as count from pb2table group by day");
query6: org.apache.spark.sql.DataFrame = [day: string, count: bigint]

scala> query6.show();
+---+-----+
|day|count|
+---+-----+
|Sun|18395|
|Mon|18346|
|Thu|17682|
|Sat|18569|
|Wed|17932|
|Fri|17725|
|Tue|18176|
+---+-----+


scala> query6.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q6.csv");
```



Query7:

scala> val query7 = sqlContext.sql("SELECT count(*) as count, user.screen_name FROM pb2table group by user.screen_name LIMIT 15");

query7: org.apache.spark.sql.DataFrame = [count: bigint, screen_name: string]

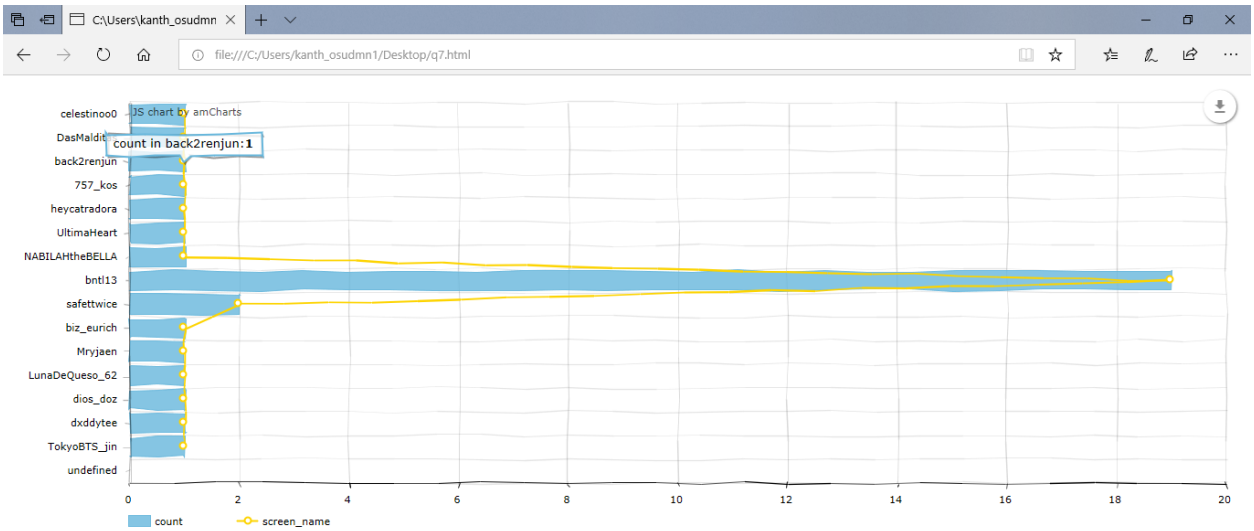
```
scala> query7.show();
+-----+---------------+
|count|    screen_name|
+-----+---------------+
|    1|    celestinoo0|
|    1|    DasMalditas|
|    1|    back2renjun|
|    1|       757_kos|
|    1|   heycatradora|
|    1|    UltimaHeart|
|    1|NABILAHtheBELLA|
|   19|        bntl13|
|    2|     safettwice|
|    1|     biz_eurich|
|    1|       Mryjaen|
|    1| LunaDeQueso_62|
|    1|      dios_doz|
|    1|      dxddytee|
|    1|   TokyoBTS_jin|
+-----+---------------+
```

scala>query7.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q7.csv");

Query7 output screen:

Administrator: Command Prompt - spark-shell.cmd

```
scala>  val query7 = sqlContext.sql("SELECT count(*) as count, user.screen_name FROM pb2table group by user.screen_name LIMIT 15");
query7: org.apache.spark.sql.DataFrame = [count: bigint, screen_name: string]

scala> query7.show();
+-----+---------------+
|count|    screen_name|
+-----+---------------+
|    1|    celestinoo0|
|    1|    DasMalditas|
|    1|    back2renjun|
|    1|        757_kos|
|    1|   heycatradora|
|    1|    UltimaHeart|
|    1|NABILAHtheBELLA|
|   19|         bntl13|
|    2|     safettwice|
|    1|     biz_eurich|
|    1|        Mryjaen|
|    1| LunaDeQueso_62|
|    1|       dios_doz|
|    1|       dxddytee|
|    1|   TokyoBTS_jin|
+-----+---------------+


scala> query7.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q7.csv");
```



Query8:

```
scala> val query8 = sqlContext.sql("SELECT user.verified,user.screen_name,user.followers_count FROM pb2table WHERE user.verified = false  ORDER BY user.followers_count DESC LIMIT 15");

query8: org.apache.spark.sql.DataFrame = [verified: boolean, screen_name: string ... 1 more field]

scala> query8.show();
+--------+---------------+---------------+
|verified|    screen_name|followers_count|
+--------+---------------+---------------+
|   false|    ItsFoodPorn|       3253123|
|   false|           Fact|       1962223|
|   false|        PlobJai|       1429506|
|   false|      bharianmy|       1240449|
|   false| VillegasPoljak|       1052042|
|   false| VillegasPoljak|       1052042|
|   false|UnreveaISecrets|       1002686|
|   false|       sakamobi|       1002203|
|   false|   godsgirl8494|        791449|
|   false|       lacuarta|        783303|
|   false|    SneakerNews|        710630|
|   false|     Zapata_zos|        706390|
|   false|      i_YaserSh|        687030|
|   false|      i_YaserSh|        687028|
|   false|      i_YaserSh|        687028|
+--------+---------------+---------------+


scala>query8.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Deskop\\phase2\\q8.csv");
```

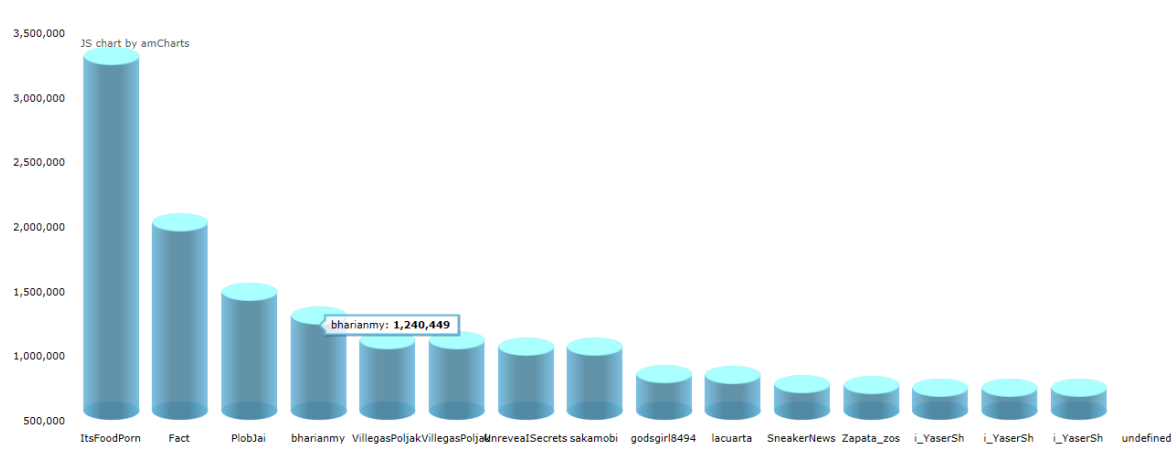
Query8 output screen:

Administrator: Command Prompt - spark-shell.cmd

```
scala> val query8 = sqlContext.sql("SELECT user.verified,user.screen_name,user.followers_count FROM pb2table WHERE user.verified = false  ORDER BY user.followers_count
DESC LIMIT 15");
query8: org.apache.spark.sql.DataFrame = [verified: boolean, screen_name: string ... 1 more field]

scala> query8.show();
+--------+---------------+---------------+
|verified|    screen_name|followers_count|
+--------+---------------+---------------+
|   false|    ItsFoodPorn|        3253123|
|   false|          Fact|        1962223|
|   false|        PlobJai|        1429506|
|   false|      bharianmy|        1240449|
|   false| VillegasPoljak|        1052042|
|   false| VillegasPoljak|        1052042|
|   false|UnreveaISecrets|        1002686|
|   false|       sakamobi|        1002203|
|   false|   godsgirl8494|         791449|
|   false|       lacuarta|         783303|
|   false|    SneakerNews|         710630|
|   false|     Zapata_zos|         706390|
|   false|      i_YaserSh|         687030|
|   false|      i_YaserSh|         687028|
|   false|      i_YaserSh|         687028|
+--------+---------------+---------------+


scala> query8.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q8.csv");
```

C:\Users\kanth_osudmn

file:///C:/Users/kanth_osudmn1/Desktop/q8.html



Type here to search

ENG 8:58 PM 5/5/2019

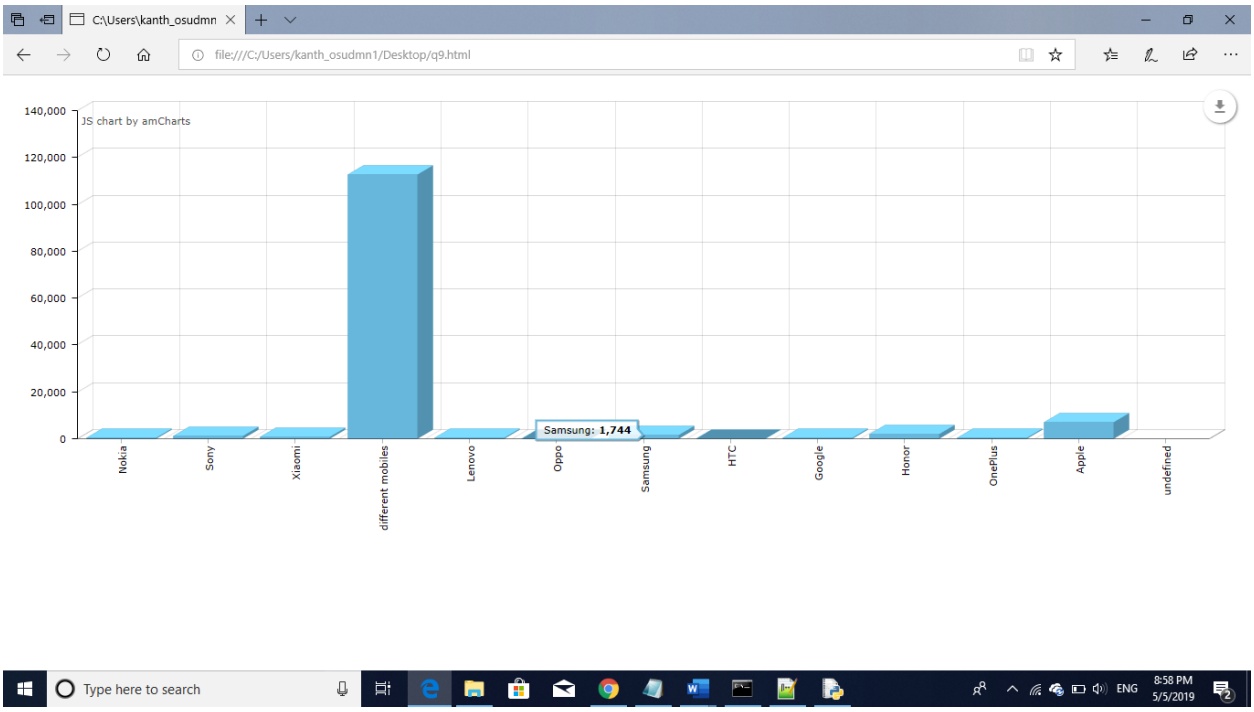
Query9:

```
scala> val Query9=sqlContext.sql("select count(*) as count,k.text from (select case when text like '%Nokia%' then 'Nokia' when text like '%Xiaomi%' then 'Xiaomi' when text like '%Oppo%' then 'Oppo' when text like '%OnePlus%' then 'OnePlus' when text like '%Lenovo%' then 'Lenovo' when text like '%Honor%' then 'Honor' WHEN text like '%Samsung%' THEN 'Samsung' WHEN text like '%Sony%' THEN 'Sony' WHEN text like '%Apple%' THEN 'Apple' WHEN text like '%HTC%' THEN 'HTC' WHEN text like '%Google%' THEN 'Google' else 'different mobiles' end as text from pb2table)k group by k.text");

Query9: org.apache.spark.sql.DataFrame = [count: bigint, text: string]

scala> Query9.show();
+------+-----------------+
| count|             text|
+------+-----------------+
|   202|            Nokia|
|  1195|             Sony|
|   747|           Xiaomi|
|112799|different mobiles|
|   292|           Lenovo|
|   146|            Oppo|
|  1744|          Samsung|
|    58|             HTC|
|   235|           Google|
|  2107|            Honor|
|   339|          OnePlus|
|  6961|            Apple|
+------+-----------------+


scala>Query9.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q9.csv");
```

Query9 output screen:

Administrator: Command Prompt - spark-shell.cmd

```
scala> val Query9=sqlContext.sql("select count(*) as count,k.text from (select case when text like '%Nokia%' then 'Nokia' when text like '%Xiaomi%' then 'Xiaomi' when t
ext like '%Oppo%' then 'Oppo' when text like '%OnePlus%' then 'OnePlus' when text like '%Lenovo%' then 'Lenovo' when text like '%Honor%' then 'Honor' WHEN text like '%S
amsung%' THEN 'Samsung' WHEN text like '%Sony%' THEN 'Sony' WHEN text like '%Apple%' THEN 'Apple' WHEN text like '%HTC%' THEN 'HTC' WHEN text like '%Google%' THEN 'Goog
le' else 'different mobiles' end as text from pb2table)k group by k.text");
Query9: org.apache.spark.sql.DataFrame = [count: bigint, text: string]

scala> query9.show();
<console>:29: error: not found: value query9
       query9.show();
       ^

scala> Query9.show();
+------+-----------------+
| count|             text|
+------+-----------------+
|   202|            Nokia|
|  1195|             Sony|
|   747|           Xiaomi|
|112799|different mobiles|
|   292|           Lenovo|
|   146|             Oppo|
|  1744|          Samsung|
|    58|              HTC|
|   235|           Google|
|  2107|            Honor|
|   339|          OnePlus|
|  6961|            Apple|
+------+-----------------+


scala> Query9.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q9.csv");
```



Query10:

scala> val query10 = sqlContext.sql("SELECT user.screen_name,text,retweeted_status.retweet_count FROM pb2table ORDER BY retweeted_status.retweet_count DESC LIMIT 20");

query10: org.apache.spark.sql.DataFrame = [screen_name: string, text: string ... 1 more field]

```
scala> query10.show();
+---------------+--------------------+-------------+
|    screen_name|                text|retweet_count|
+---------------+--------------------+-------------+
|       JayDaGxD|RT @KingJames: U ...|       620211|
|     JDrizzle29|RT @KingJames: U ...|       620211|
|       The__RAD|RT @KingJames: U ...|       620209|
|     WhaTheHect|RT @KingJames: U ...|       620208|
| JudithGustman9|RT @TheGrefgYT: ?...|       417184|
|AimeeCa39957037|RT @ChrisEvans: O...|       348120|
|      cieloAmi1|RT @AppleMusic: L...|       240525|
|      meyodmin_|RT @AppleMusic: L...|       240522|
| Fah_tanyarat77|RT @AppleMusic: L...|       240522|
| KhnhVn27396485|RT @AppleMusic: L...|       240520|
| taekookshohoho|RT @AppleMusic: L...|       240519|
|Julieta97543462|RT @AppleMusic: L...|       240515|
|AngelicaLaurea5|RT @AppleMusic: L...|       240512|
| karlalaardilla|RT @AppleMusic: L...|       240506|
|myyouth62438753|RT @AppleMusic: L...|       240504|
|   IvethSaldaa2|RT @AppleMusic: L...|       240504|
|   bts130613_jk|RT @AppleMusic: L...|       240503|
|       2243_bts|RT @AppleMusic: L...|       240503|
|        wuvsss_|RT @AppleMusic: L...|       240503|
| BurgosChamorro|RT @AppleMusic: L...|       240502|
+---------------+--------------------+-------------+
```

scala>query10.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q10.csv");

Query10 output screen:

Administrator: Command Prompt - spark-shell.cmd

```
scala> val query10 = sqlContext.sql("SELECT user.screen_name,text,retweeted_status.retweet_count FROM pb2table ORDER BY retweeted_status.retweet_count DESC LIMIT 20");
query10: org.apache.spark.sql.DataFrame = [screen_name: string, text: string ... 1 more field]

scala> query10.show();
+---------------+--------------------+-------------+
|    screen_name|                text|retweet_count|
+---------------+--------------------+-------------+
|       JayDaGxD|RT @KingJames: U ...|       620211|
|     JDrizzle29|RT @KingJames: U ...|       620211|
|       The__RAD|RT @KingJames: U ...|       620209|
|     WhaTheHect|RT @KingJames: U ...|       620208|
| JudithGustman9|RT @TheGrefgYT: ?...|       417184|
|AimeeCa39957037|RT @ChrisEvans: O...|       348120|
|      cieloAmi1|RT @AppleMusic: L...|       240525|
|      meyodmin_|RT @AppleMusic: L...|       240522|
| Fah_tanyarat77|RT @AppleMusic: L...|       240522|
| KhnhVn27396485|RT @AppleMusic: L...|       240520|
| taekookshohoho|RT @AppleMusic: L...|       240519|
|Julieta97543462|RT @AppleMusic: L...|       240515|
|AngelicaLaurea5|RT @AppleMusic: L...|       240512|
| karlalaardilla|RT @AppleMusic: L...|       240506|
|myyouth62438753|RT @AppleMusic: L...|       240504|
|   IvethSaldaa2|RT @AppleMusic: L...|       240504|
|   bts130613_jk|RT @AppleMusic: L...|       240503|
|       2243_bts|RT @AppleMusic: L...|       240503|
|        wuvsss_|RT @AppleMusic: L...|       240503|
| BurgosChamorro|RT @AppleMusic: L...|       240502|
+---------------+--------------------+-------------+


scala> query10.coalesce(1).write.format("com.databricks.spark.csv").save("C:\\Users\\kanth_osudmn1\\Desktop\\phase2\\q10.csv");

scala>
```

C:\Users\kanth_osudmn

file:///C:/Users/kanth_osudmn1/Desktop/q10.html

JS chart by amCharts