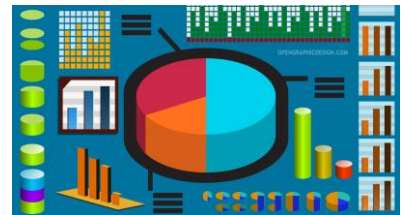
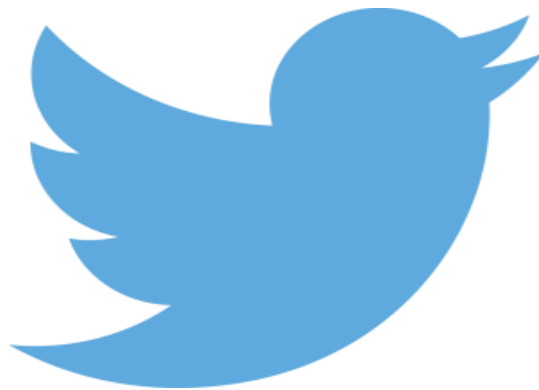


# Principles of Big Data Management Phase-2

Fall 2017



Submitted by: (Team 13)

Pragathi Thammaneni

Sridevi Mallipudi

Fatema Hasta

## Analyzing and Visualizing Twitter data on Mobiles

### Objective:

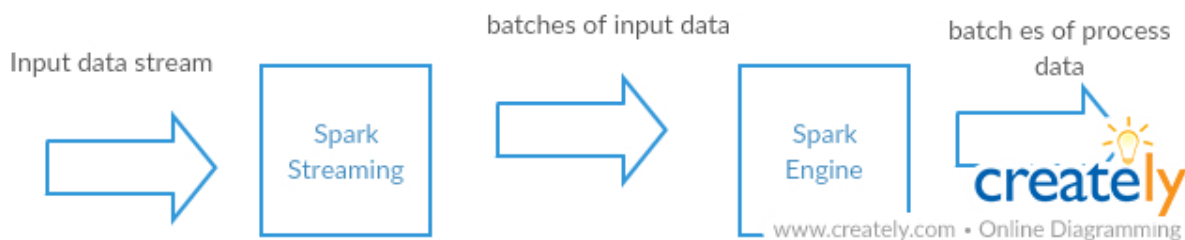
**Applications/Software's Used:** Apache Spark SQL, Scala, Cloudera, Twitter Developer Account, Python.

### Collecting tweets from Twitter:

- Firstly, we have created a developer account in Twitter using below link.  
<https://apps.twitter.com/>
- Below are the variables that contains the user credentials to access Twitter API
  - API\_ACCESS\_TOKEN = "1329248834-R3X8SrLzchMQyBOSRtV1QSJYfis2lidRex4RXeq"
  - ACCESS\_SECRET = "g6pwJFjRTQK2cRi01rGV3kpr6C9v7sQNTjrtPLtiETc4"
  - CONSUMER\_KEY = "I82JLsLffFzGMrytXhF4DAnf2"
  - CONSUMER\_SECRET = "Fh4985bvDNeF2RurZsseuDYBnbFv3cLmpdyizFapBkr1uS954"
- We have written python program that is used to fetch tweets in JSON format. (TweetsExtract.py)
- The tweet data is collected on the concept based on to analyze and visualize the data regarding various mobile phones.

Sample tweets are collected for the key words by using python program  
`#iphone', '#Samsung', '#Moto', '#Redmi', '#Xiaomi', '#Nokia', '#lenovo', '#oppo', '#OnePlus', '#BlackBerry', '#HTC`

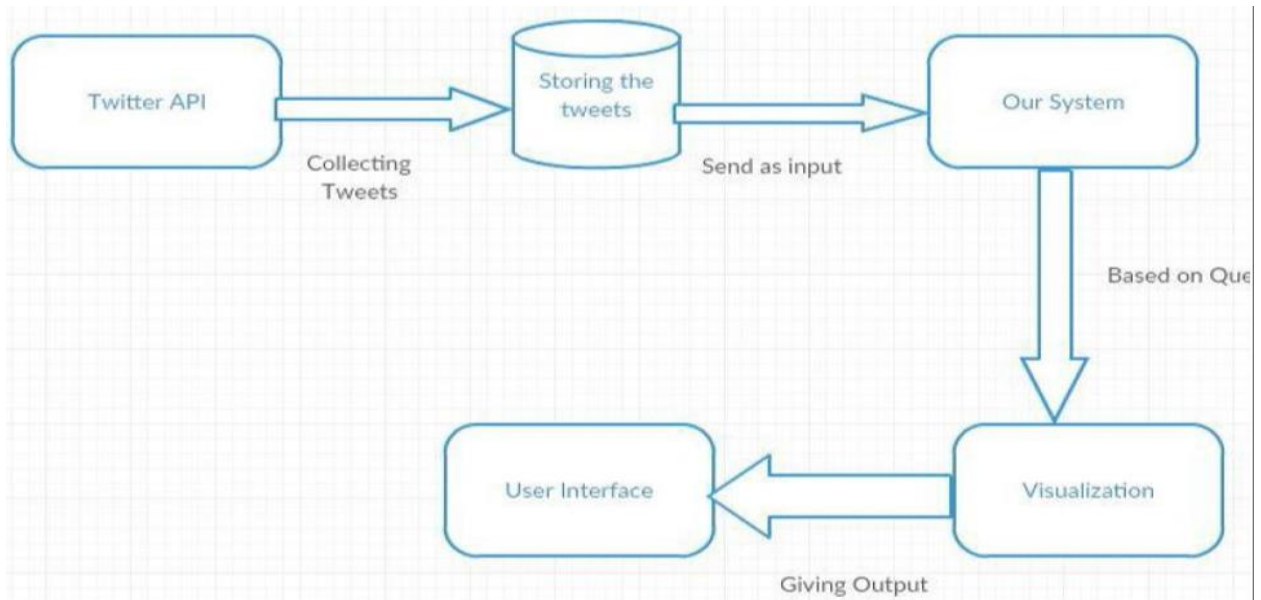
- 1) Spark streaming, the data can be -ingested – from different sources like twitter and can perform high level complex algorithms like queries and the processed data can be pushed out to the file systems. To stream the twitter data twitter.utils contains all the built in functionality -to- stream data from twitter.



- **SqlContext** which is a new- session, with separated SQL configurations, registered functions temporary tables (data frames to store the all relational functionality in Spark SQL).

The main theme of this project is to do big data analytics on the Mobile phones. Based on the twitter tweets, we predicted few interesting query analysis and visualization on mobile phone twitter data. First, we collected the tweets regarding the mobile phone data from the twitter API. By using the collected data, we build 10 interesting queries which results a data analysis on the twitter data. Visualization is done for on the result sets and viewed in some pictorial representations like pie chart and bar graphs.

#### Architecture diagram:



#### Sample JSON Object Structure:

The tweets are saved and stored with the object format as JSON.

```

{
  "created_at": "Fri Nov 10 16:11:34 +0000 2017",
  "id": "929018704041349120",
  "id_str": "929018704041349120",
  "text": "You know #SteveJobs hustled his way up to @Apple (and they still got the best hustle! You got that #iphoneX don't y\u2026 https://t.co/GqUSbaqhFF",
  "display_text_range": [
    0,
    140
  ]
}
  
```

```
],  
  "source": "\u003ca href=\"https://mobile.twitter.com\" rel=\"nofollow\" \u003eTwitter  
Lite\u003c/a\u003e",  
  "truncated": true,  
  "in_reply_to_status_id": null,  
  "in_reply_to_status_id_str": null,  
  "in_reply_to_user_id": null,  
  "in_reply_to_user_id_str": null,  
  "in_reply_to_screen_name": null,  
  "user": {  
    "id": 923408384358940674,  
    "id_str": "923408384358940674",  
    "name": "Retired Ratchet",  
    "screen_name": "retiredratchet",  
    "location": "Las Vegas, NV",  
    "url": null,  
    "description": "#Blitter for the culture:\nRatchet is a slang term in hip hop that in the  
strictest sense refers to an uncouth female & is a Louisianan regiolect of \"wretched\"",  
    "translator_type": "none",  
    "protected": false,  
    "verified": false,  
    "followers_count": 156,  
    "friends_count": 224,  
    "listed_count": 0,  
    "favourites_count": 993,  
    "statuses_count": 618,  
    "created_at": "Thu Oct 26 04:38:09 +0000 2017",  
    "utc_offset": null,
```

```
"time_zone":null,
"geo_enabled":false,
"lang":"en",
"contributors_enabled":false,
"is_translator":false,
"profile_background_color":"F5F8FA",
"profile_background_image_url":"",
"profile_background_image_url_https":"",
"profile_background_tile":false,
"profile_link_color":"1DA1F2",
"profile_sidebar_border_color":"CODEED",
"profile_sidebar_fill_color":"DDEEF6",
"profile_text_color":"333333",
"profile_use_background_image":true,

"profile_image_url":"http://pbs.twimg.com/profile_images/923427538038173696/tBE9KGIV_normal.jpg",

"profile_image_url_https":"https://pbs.twimg.com/profile_images/923427538038173696/tBE9KGIV_normal.jpg",

"profile_banner_url":"https://pbs.twimg.com/profile_banners/923408384358940674/1510329291",

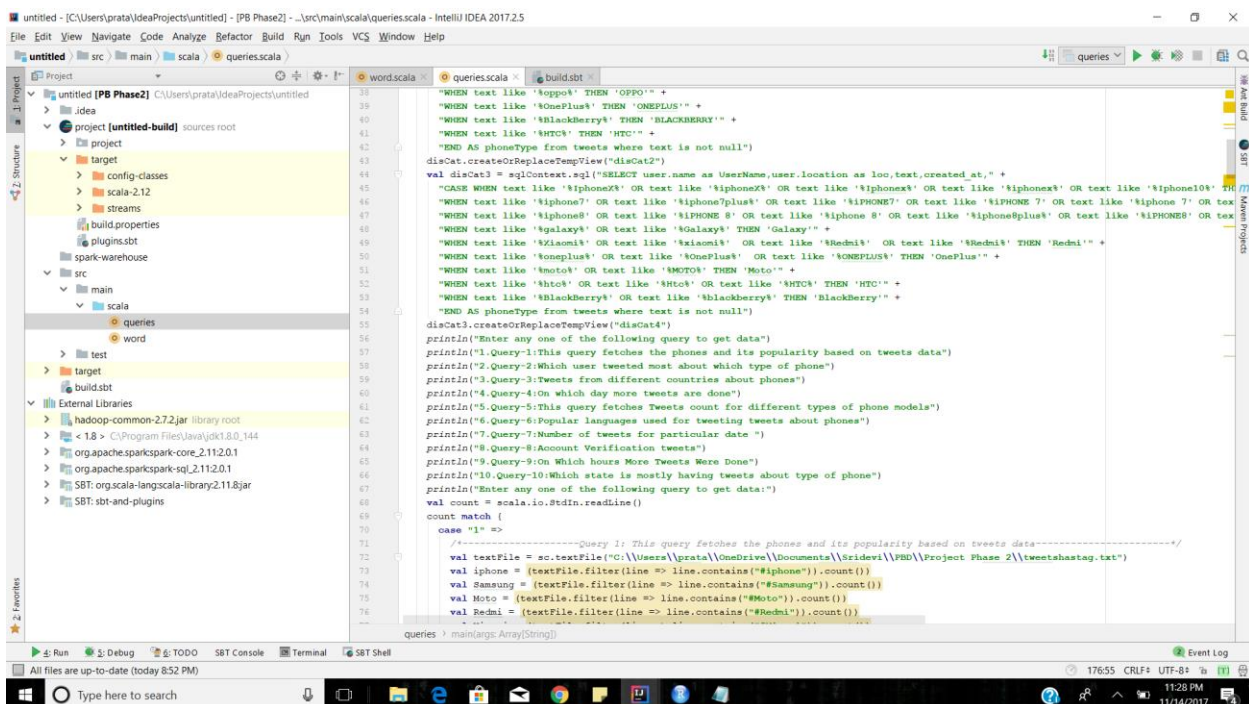
"default_profile":true,
"default_profile_image":false,
"following":null,
"follow_request_sent":null,
"notifications":null
},
```

```
"geo":null,  
"coordinates":null,  
"place":null,  
"contributors":null,  
"quoted_status_id":927672133739835392,...  
}
```

## 2) Queries and outputs:

Below console shows indexes for queries to perform which query by the input choice given, to get the corresponding data.

**Code:**



**Output:**

```

1  import org.apache.spark.SparkContext
2  import org.apache.spark.SparkConf
3  import org.apache.spark.sql.catalyst.plans.logical.Union
4  import org.apache.spark.sql.types.{DateType, FloatType}
5  import org.apache.spark.sql.functions._
6  import org.apache.spark.sql.catalyst.encoders.ExpressionEncoder
7  import org.apache.spark.sql.Encoder

17/11/14 20:21:33 INFO deprecation: mapred.task.is.map is deprecated. Instead, use mapreduce.task.ismap
17/11/14 20:21:33 INFO deprecation: mapred.task.partition is deprecated. Instead, use mapreduce.task.partition
17/11/14 20:21:33 INFO deprecation: mapred.job.id is deprecated. Instead, use mapreduce.job.id
17/11/14 20:21:34 INFO Executor: Finished task 0.0 in stage 0.0 (TID 0). 30260 bytes result sent to driver
17/11/14 20:21:34 INFO Executor: Finished task 1.0 in stage 0.0 (TID 1). 30674 bytes result sent to driver
17/11/14 20:21:34 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 1746 ms on localhost (1/2)
17/11/14 20:21:34 INFO TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in 1718 ms on localhost (2/2)
17/11/14 20:21:34 INFO DAGScheduler: ResultStage 0 (json at queries.scala:16) finished in 1.807 s
17/11/14 20:21:34 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
17/11/14 20:21:34 INFO DAGScheduler: Job 0 finished: json at queries.scala:16, took 2.030963 s
17/11/14 20:21:35 INFO BlockManagerInfo: Removed broadcast_1_piece0 on 192.168.0.15:61060 in memory (size: 4.8 KB, free: 1983.3 MB)
17/11/14 20:21:37 WARN Utils: 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.
17/11/14 20:21:38 INFO SparkSqlParser: Parsing command: tweets
17/11/14 20:21:38 INFO SparkSqlParser: Parsing command: SELECT user.name as UserName,user.location as loc,text,created_at,CASE WHEN text like '%iphone%' THEN 'IPHONE'WHEN text like '%Samsung%' THEN
17/11/14 20:21:38 INFO SparkSqlParser: Parsing command: SELECT user.name as UserName,user.location as loc,text,created_at,CASE WHEN text like '%iphone%' OR text like '%Samsung%' OR text like '%ipho
17/11/14 20:21:38 INFO SparkSqlParser: Parsing command: disCat4
Enter any one of the following query to get data
1.Query-1:This query fetches the phones and its popularity based on tweets data
2.Query-2:Which user tweeted most about which type of phone
3.Query-3:Tweets from different countries about phones
4.Query-4:On which day more tweets are done
5.Query-5:This query fetches Tweets count for different types of phone models
6.Query-6:Popular languages used for tweeting tweets about phones
7.Query-7:Number of tweets for particular date
8.Query-8:Account Verification tweets
9.Query-9:On Which hours More Tweets Were Done
10.Query-10:Which state is mostly having tweets about type of phone
Enter any one of the following query to get data:

```

**Query 1: Query for fetching the tweets corresponding to phones and its popularity count on the tweets depending up the on tweets data collected.**

**Answer:** This query is built to analyze the tweets – what the count of each model which are tweeted by the users in the collected tweets which directly reflects the popularity of each phone model.

As we collected the data related to mobile phones

`#iphone', '#Samsung', '#Moto', '#Redmi', '#Xiaomi', '#Nokia', '#lenovo', '#oppo', '#OnePlus', '#BlackBerry', '#HTC`

Such that built a query to perform the count operation on each model. So, it results the count -how many times a phone model appears in the tweets.

**Code:**



```

62 println("6.Query-6:Popular languages used for tweeting tweets about phones")
63 println("7.Query-7:Number of tweets for particular date ")
64 println("8.Query-8:Bookmark Verification tweets")
65 println("9.Query-9:On Which hours More Tweets Were Done")
66 println("10.Query-10:Which state is mostly having tweets about type of phone")
67 println("Enter any one of the following query to get data:")
68 val count = scala.io.StdIn.readLine()
69 count match {
70   case "1" =>
71     /*-----Query 1: This query fetches the phones and its popularity based on tweets data-----*/
72     val textFile = sc.textFile("C:\\Users\\prata\\OneDrive\\Documents\\xridevi\\PBD\\Project Phase 2\\tweetshastag.txt")
73     val iPhone = (textFile.filter(line => line.contains("#iPhone")).count())
74     val Samsung = (textFile.filter(line => line.contains("#Samsung")).count())
75     val Moto = (textFile.filter(line => line.contains("#Moto")).count())
76     val Redmi = (textFile.filter(line => line.contains("#Redmi")).count())
77     val Xiaomi = (textFile.filter(line => line.contains("#Xiaomi")).count())
78     val Nokia = (textFile.filter(line => line.contains("#Nokia")).count())
79     val lenovo = (textFile.filter(line => line.contains("#lenovo")).count())
80     val oppo = (textFile.filter(line => line.contains("#oppo")).count())
81     val OnePlus = (textFile.filter(line => line.contains("#OnePlus")).count())
82     val BlackBerry = (textFile.filter(line => line.contains("#BlackBerry")).count())
83     val HTC = (textFile.filter(line => line.contains("#HTC")).count())
84     println("*****")
85     println("Number of tweets on different types of phones")
86     println("*****")
87     println("iPhone : %s".format(iPhone))
88     println("Samsung : %s".format(Samsung))
89     println("Moto : %s".format(Moto))
90     println("Redmi : %s".format(Redmi))
91     println("Xiaomi : %s".format(Xiaomi))
92     println("Nokia : %s".format(Nokia))
93     println("Lenovo : %s".format(lenovo))
94     println("Oppo : %s".format(oppo))
95     println("OnePlus : %s".format(OnePlus))
96     println("BlackBerry : %s".format(BlackBerry))
97     println("HTC : %s".format(HTC))
98     /*-----Query 2: Which user tweeted most about which type of phone-----*/
99   case "2" =>
100

```

## Output:

```

181 val r8 = sqlContext.sql("SELECT loc, 'HTC' as phoneType, count(*) as count FROM discat4 WHERE phoneType='HTC' " +
182 "group by loc order by count desc limit 10")
183 val r9 = sqlContext.sql("SELECT loc, 'BlackBerry' as phoneType, count(*) as count FROM discat4 WHERE phoneType='BlackBerry' " +
184 "group by loc order by count desc limit 10")
185 val rdd1 = r1.union(r2).union(r3).union(r4).union(r5).union(r6).union(r7).union(r8).union(r9)
186 rdd1.createOrReplaceTempView("rdd1")
187 val res=sqlContext.sql("SELECT phoneType, Count(*) as Count from rdd1 where phoneType is not null group by phoneType")
188 queries : main(args: Array[String])

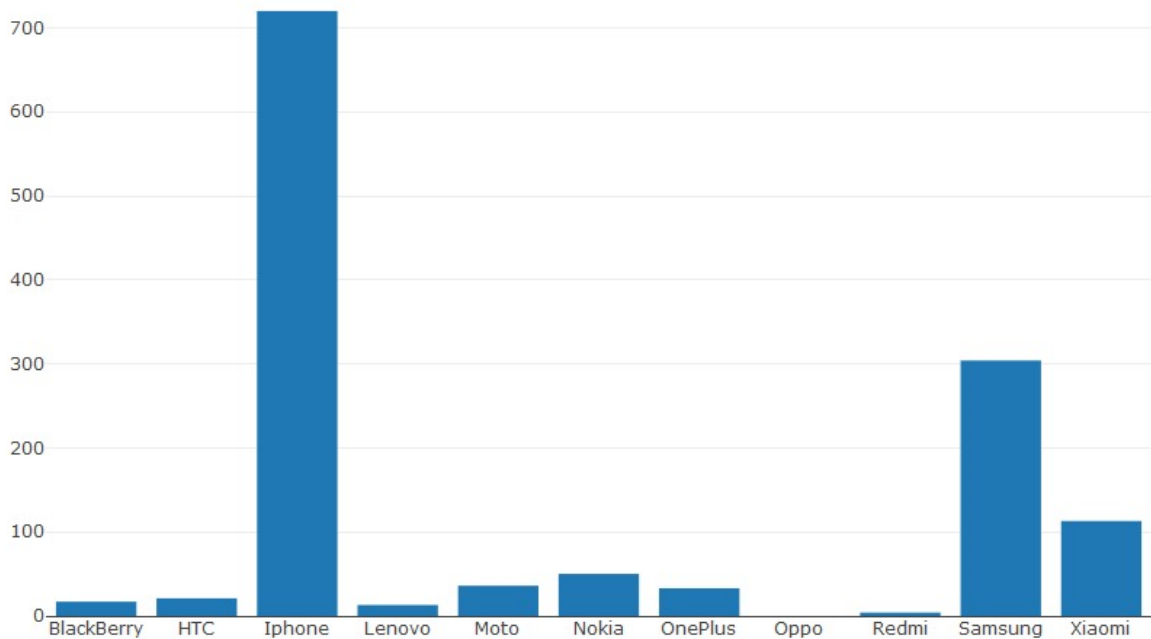
```

```

17/11/14 20:30:20 INFO TaskSchedulerImpl: Removed TaskSet 11-0, whose tasks have all completed, from pool
17/11/14 20:30:20 INFO DAGScheduler: ResultStage 11 (count at queries.scala:83) finished in 0.034 s
17/11/14 20:30:20 INFO DAGScheduler: Job 11 finished: count at queries.scala:83, took 0.045503 s
*****
Number of tweets on different types of phones
*****
iPhone : 720
Samsung : 304
Moto : 36
Redmi : 4
Xiaomi : 113
Nokia : 50
Lenovo : 13
Oppo : 0
OnePlus : 33
BlackBerry : 17
HTC : 21
17/11/14 20:30:20 INFO SparkContext: Invoking stop() from shutdown hook
17/11/14 20:30:20 INFO SparkUI: Stopped Spark web UI at http://192.168.0.15:4040
17/11/14 20:30:20 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
17/11/14 20:30:20 INFO MemoryStore: MemoryStore cleared
17/11/14 20:30:20 INFO BlockManager: BlockManager stopped
17/11/14 20:30:20 INFO BlockManagerMaster: BlockManagerMaster stopped
17/11/14 20:30:20 INFO OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
17/11/14 20:30:20 INFO SparkContext: Successfully stopped SparkContext
17/11/14 20:30:20 INFO ShutdownHookManager: Shutdown hook called
17/11/14 20:30:20 INFO ShutdownHookManager: Deleting directory C:\Users\prata\AppData\Local\Temp\spark-a0c5d8ba-57dc-4422-996c-c614be8408de
Process finished with exit code 0

```



**Visualization:****Query 2: Query for fetching which user tweeted most about which type of phone.**

**Answer:** This query is built to analyze the tweets – based on which user tweeted most about which type of phone.

As we collected the data related to mobile phones

```
#iphone', '#Samsung', '#Moto', '#Redmi', '#Xiaomi', '#Nokia', '#lenovo', '#oppo', '#OnePlus', '#BlackBerry', '#HTC
```

Such that built a query to perform the user tweeted most about which type of phone. So, it results the count -how many times a user tweeted at most for each mobile phone.

**Code:**

```

121 val r1 = sqlContext.sql("SELECT UserName, 'IPHONE' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='IPHONE' " +
122   "group by UserName order by count desc limit 1")
123 val r2 = sqlContext.sql("SELECT UserName, 'SAMSUNG' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='SAMSUNG' " +
124   "group by UserName order by count desc limit 1")
125 val r3 = sqlContext.sql("SELECT UserName, 'MOTO' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='MOTO' " +
126   "group by UserName order by count desc limit 1")
127 val r4 = sqlContext.sql("SELECT UserName, 'REDMI' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='REDMI' " +
128   "group by UserName order by count desc limit 1")
129 val r5 = sqlContext.sql("SELECT UserName, 'XIAOMI' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='XIAOMI' " +
130   "group by UserName order by count desc limit 1")
131 val r6 = sqlContext.sql("SELECT UserName, 'NOKIA' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='NOKIA' " +
132   "group by UserName order by count desc limit 1")
133 val r7 = sqlContext.sql("SELECT UserName, 'LENOVO' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='LENOVO' " +
134   "group by UserName order by count desc limit 1")
135 val r8 = sqlContext.sql("SELECT UserName, 'OPPO' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='OPPO' " +
136   "group by UserName order by count desc limit 1")
137 val r9 = sqlContext.sql("SELECT UserName, 'ONEPLUS' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='ONEPLUS' " +
138   "group by UserName order by count desc limit 1")
139 val r10 = sqlContext.sql("SELECT UserName, 'BLACKBERRY' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='BLACKBERRY' " +
140   "group by UserName order by count desc limit 1")
141 val r11 = sqlContext.sql("SELECT UserName, 'HTC' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='HTC' " +
142   "group by UserName order by count desc limit 1")
143
144 val rdd1 = r1.union(r2).union(r3).union(r4).union(r5).union(r6).union(r7).union(r8).union(r9).union(r10).union(r11)
145
146 println("*****")
147 println("Which user tweeted more on which type of phone")
148 println("*****")
149 rdd1.show()
150
151 //-----Query 3: Tweets from different countries about phones -----*/
152
153 queries > main(args: Array[String])

```

## Output:

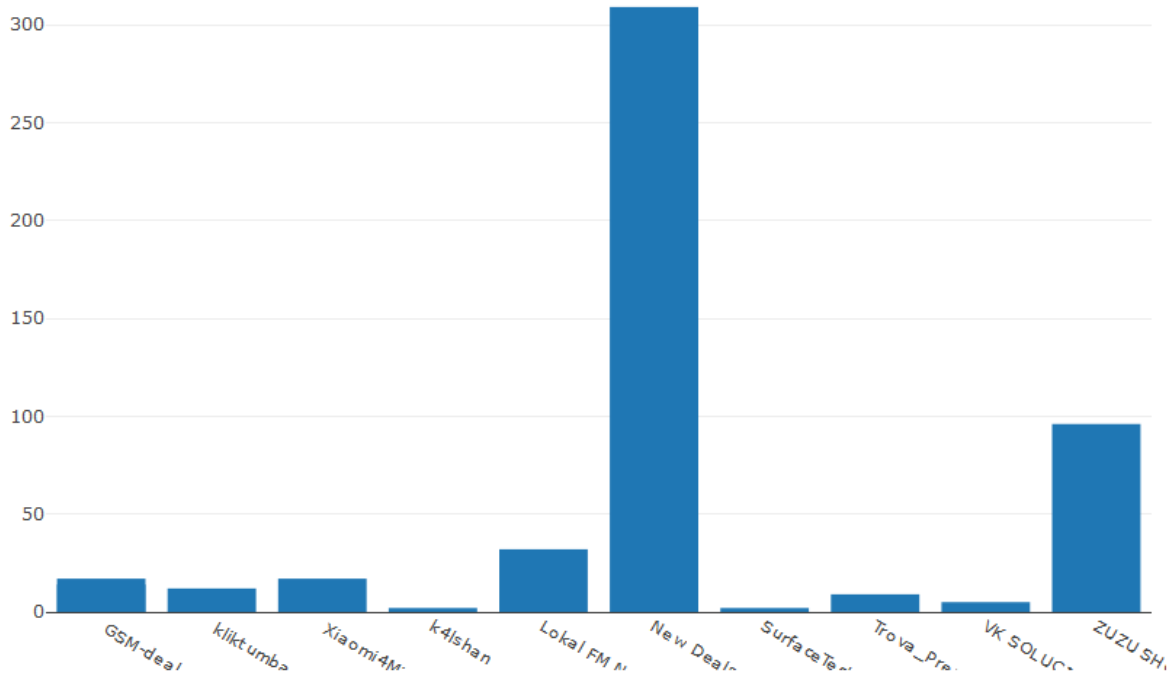
```

17/11/14 20:26:49 INFO CodeGenerator: Code generated in 7.196205 ms
17/11/14 20:26:49 INFO SparkContext: Invoking stop() from shutdown hook
+-----+
| UserName | phoneType|count|
+-----+
| New Deals | IPHONE | 309 |
| ZUSU SHOP | SAMSUNG | 96 |
| Trova_Prezzi_s_News | MOTO | 9 |
| klittumbas | REDMI | 12 |
| Xiaomi4MI | XIAOMI | 17 |
| GSN-deals.nl | NOKIA | 17 |
| VR SOLUCIONES | LENOVO | 5 |
| kilahan | OPPO | 2 |
| SurfaceTech | ONEPLUS | 2 |
| Lokal FM NOVA | BLACKBERRY | 32 |
| GSN-deals.nl | HTC | 14 |
+-----+

17/11/14 20:26:49 INFO sparkUI: Stopped Spark web UI at http://192.168.0.15:4040
17/11/14 20:26:49 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
17/11/14 20:26:57 INFO MemoryStore: MemoryStore cleared
17/11/14 20:26:57 INFO BlockManager: BlockManager stopped
17/11/14 20:26:57 INFO BlockManagerMaster: BlockManagerMaster stopped
17/11/14 20:26:57 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
17/11/14 20:26:57 INFO SparkContext: Successfully stopped SparkContext
17/11/14 20:26:57 INFO ShutdownHookManager: Shutdown hook called
17/11/14 20:26:57 INFO ShutdownHookManager: Deleting directory C:\Users\prata\AppData\Local\Temp\spark-ba664be3-f10c-42e1-93aa-ddc088652c5b

Process finished with exit code 0

```

**Visualization:****Query 3: Query for fetching Tweets from different countries about phones**

**Answer:** This query is built to analyze the tweets – based on which country tweeted more about on phones.

As we collected the data related to mobile phones

`#iphone', '#Samsung', '#Moto', '#Redmi', '#Xiaomi', '#Nokia', '#lenovo', '#oppo', '#OnePlus', '#BlackBerry', '#HTC`

Such that built a query for finding out tweets from different countries about phones. So, it results the count -how many tweets from different countries posted about phones.

**Code:**

```

108 "group by UserName order by count desc limit 1"
109 val r5 = sqlContext.sql("SELECT UserName, 'XIAOMI' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='XIAOMI' " +
110 "group by UserName order by count desc limit 1")
111 val r6 = sqlContext.sql("SELECT UserName, 'NOKIA' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='NOKIA' " +
112 "group by UserName order by count desc limit 1")
113 val r7 = sqlContext.sql("SELECT UserName, 'LENOVO' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='LENOVO' " +
114 "group by UserName order by count desc limit 1")
115 val r8 = sqlContext.sql("SELECT UserName, 'OPPO' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='OPPO' " +
116 "group by UserName order by count desc limit 1")
117 val r9 = sqlContext.sql("SELECT UserName, 'ONEPLUS' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='ONEPLUS' " +
118 "group by UserName order by count desc limit 1")
119 val r10 = sqlContext.sql("SELECT UserName, 'BLACKBERRY' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='BLACKBERRY' " +
120 "group by UserName order by count desc limit 1")
121 val r11 = sqlContext.sql("SELECT UserName, 'HTC' as phoneType, count(*) as count FROM disCat2 WHERE phoneType='HTC' " +
122 "group by UserName order by count desc limit 1")
123
124 val rdd1 = r1.union(r2).union(r3).union(r4).union(r5).union(r6).union(r7).union(r8).union(r9).union(r10).union(r11)
125
126 println("*****")
127 println("Which user tweeted more on which type of phone")
128 println("*****")
129 rdd1.show()
130
131 /*-----Query 3: Tweets from different countries about phones -----*/
132 case "3" =>
133 val countrytweetscount = sqlContext.sql("SELECT distinct place.country, count(*) as count FROM tweets where place.country is not null " +
134 "countrytweetscount.createOrReplaceTempView('countrytweetscount')")
135 println("*****")
136 println("Tweets from different countries")
137 println("*****")
138 countrytweetscount.show()
139
140 /*-----Query 4 : On which Day More Tweets are posted-----*/
141 case "4" =>
142 val day_data = sqlContext.sql("SELECT substring(user.created_at,1,3) as day from tweets where text is not null")
143 day_data.createOrReplaceTempView("day_data")
144
145 val days_final = sqlContext.sql(
146 "SELECT Case
147 [when day LIKE '%Mon%' then 'WEEKDAY'
148
149 queries > main(args: Array[String])

```

## Output:

```

181 val r8 = sqlContext.sql("SELECT loc, 'HTC' as phoneType, count(*) as count FROM disCat4 WHERE phoneType='HTC' " +
182 "group by loc order by count desc limit 10")
183 val r9 = sqlContext.sql("SELECT loc, 'BlackBerry' as phoneType, count(*) as count FROM disCat4 WHERE phoneType='BlackBerry' " +
184 "group by loc order by count desc limit 10")
185 val rdd1 = r1.union(r2).union(r3).union(r4).union(r5).union(r6).union(r7).union(r8).union(r9)
186 rdd1.createOrReplaceTempView("rdd1")
187 val res = sqlContext.sql("SELECT phoneType, Count(*) as Count from rdd1 where phoneType is not null group by phoneType")
188
189 queries > main(args: Array[String])

```

country|count|

日本	1301
United States	581
Australia	181
United Kingdom	141
Canada	91
Indonesia	81
Mexico	61
Italia	61
Mexico	51
Brasil	51
España	51
India	41
Italia	41
Turkiye	31
Spain	31
Egypt	31
Deutschland	31
Malaysia	31
France	21
Germany	21

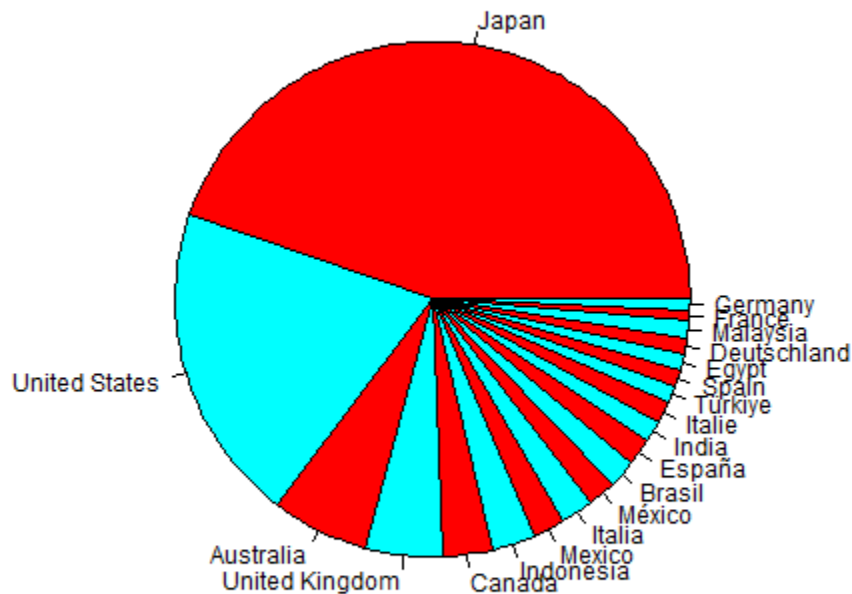
only showing top 20 rows

```

17/11/14 20:31:45 INFO SparkUI: Stopped Spark web UI at http://192.168.0.15:4040
17/11/14 20:31:45 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
17/11/14 20:31:46 INFO MemoryStore: MemoryStore cleared
17/11/14 20:31:46 INFO BlockManager: BlockManager stopped

```

## Visualization:

**Tweets about phones from different countries**

**Query 4: Query for on which day more tweets are done.**

**Answer:** This query is built to analyze the tweets – based on which day more tweets are done.

MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY as Weekday

SATURDAY, SUNDAY as Weekend

As we collected the data related to mobile phones

`#iphone', '#Samsung', '#Moto', '#Redmi', '#Xiaomi', '#Nokia', '#lenovo', '#oppo', '#OnePlus', '#BlackBerry', '#HTC`

Such that built a query to perform for analyzing on which day the more tweets are posted. So, it results the count – about giving a figure of on what day & how many tweets are done.

## Code:

```

countrytweetscount.createOrReplaceTempView("countrytweetscount")
println("=====Query 4: Tweets from different countries=====")
println("countrytweetscount.show()")
countrytweetscount.show()

/*=====Query 4: On which Day More Tweets are posted=====*/
case "4" =>
val day_data = sqlContext.sql("SELECT substr(user.created_at,1,3) as day from tweets where text is not null")
day_data.createOrReplaceTempView("day_data")

val days_final = sqlContext.sql(
  """ SELECT Case
    | when day LIKE '%Mon%' then 'WEEKDAY'
    | when day LIKE '%Tue%' then 'WEEKDAY'
    | when day LIKE '%Wed%' then 'WEEKDAY'
    | when day LIKE '%Thu%' then 'WEEKDAY'
    | when day LIKE '%Fri%' then 'WEEKDAY'
    | when day LIKE '%Sat%' then 'WEEKEND'
    | when day LIKE '%Sun%' then 'WEEKEND'
    | else
    | null
    | end as day1 from day_data where day is not null""").stripMargin)

days_final.createOrReplaceTempView("days_final")

val res = sqlContext.sql("SELECT day1 as Day,Count(*) as Day_Count from days_final where day1 is not null group by day1 order by count")

println("=====")
println("On Which Day More Tweets Were Done")
println("=====")
res.show()

/*=====Query 5: Tweets count for different types of phone models=====*/
case "5" =>
val r1 = sqlContext.sql("SELECT loc,'iPhone X' as phoneType,count(*) as count FROM disCat4 WHERE phoneType='iPhone X' " +
  "group by loc order by count desc limit 10")
val r2 = sqlContext.sql("SELECT loc,'iPhone7 Series' as phoneType,count(*) as count FROM disCat4 WHERE phoneType='iPhone7 Series' " +
  "group by loc order by count desc limit 10")
val r3 = sqlContext.sql("SELECT loc,'iPhone8 Series' as phoneType,count(*) as count FROM disCat4 WHERE phoneType='iPhone8 Series' " +
  "group by loc order by count desc limit 10")
val rdd1 = r1.union(r2).union(r3).union(r4).union(r5).union(r6).union(r7).union(r8).union(r9)
rdd1.createOrReplaceTempView("rdd1")
val res=sqlContext.sql("SELECT phoneType, Count(*) as Count from rdd1 where phoneType is not null group by phoneType")
res.show()

```

## Output:

```

17/11/14 20:33:32 INFO ShuffleBlockFetcherIterator: Getting 4 non-empty blocks out of 4 blocks
17/11/14 20:33:32 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
17/11/14 20:33:32 INFO Executor: Finished task 45.0 in stage 2.0 (TID 204). 4056 bytes result sent to driver
17/11/14 20:33:32 INFO Executor: Finished task 120.0 in stage 2.0 (TID 205). 3965 bytes result sent to driver
17/11/14 20:33:32 INFO TaskSetManager: Finished task 45.0 in stage 2.0 (TID 204) in 32 ms on localhost (199/200)
17/11/14 20:33:32 INFO TaskSetManager: Finished task 120.0 in stage 2.0 (TID 205) in 24 ms on localhost (200/200)
17/11/14 20:33:32 INFO TaskSchedulerImpl: Removed TaskSet 2.0, whose tasks have all completed, from pool
17/11/14 20:33:32 INFO DAGScheduler: ResultStage 2 (show at queries.scala:164) finished in 1.126 s
17/11/14 20:33:32 INFO DAGScheduler: Job 1 finished: show at queries.scala:164, took 1.996115 s
17/11/14 20:33:32 INFO CodeGenerator: Code generated in 6.483915 ms

+-----+
| Day|Day_Count|
+-----+
| WEEKDAY |      8271 |
| WEEKEND |      4042 |
+-----+

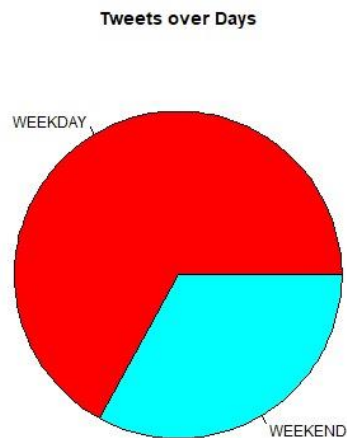
17/11/14 20:33:32 INFO SparkContext: Invoking stop() from shutdown hook
17/11/14 20:33:32 INFO SparkUI: Stopped Spark web UI at http://192.168.0.15:4040
17/11/14 20:33:32 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
17/11/14 20:33:32 INFO MemoryStore: MemoryStore cleared
17/11/14 20:33:32 INFO BlockManager: BlockManager stopped
17/11/14 20:33:32 INFO BlockManagerMaster: BlockManagerMaster stopped
17/11/14 20:33:32 INFO OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
17/11/14 20:33:32 INFO SparkContext: Successfully stopped SparkContext
17/11/14 20:33:32 INFO ShutdownHookManager: Shutdown hook called
17/11/14 20:33:32 INFO ShutdownHookManager: Deleting directory C:\Users\prata\AppData\Local\Temp\spark-e568b13e-2965-45b5-b4ec-d808bd1f37d5

Process finished with exit code 0

```

## Visualization:





**Query 5: This query fetches Tweets count for different series of phone models.**

**Answer:** This query is built to analyze the tweets – for the individual series of each mobile phone.

As we collected the data related to mobile phones  
`#iphone', '#Samsung', '#Moto', '#Redmi', '#Xiaomi', '#Nokia', '#lenovo', '#oppo', '#OnePlus', '#BlackBerry', '#HTC`

Such that built a query to perform the user tweeted most which type of series for mobile phone. So, it results the tweets count -for different series of phone models.



## Code:

```

161 println("*****")
162 println("On Which Day More Tweets Were Done")
163 println("*****")
164 res.show()
165
166 -----Query 5: Tweets count for different types of phone models -----
167
168 case '5' =>
169   val r1 = sqlContext.sql("SELECT loc, 'iPhone X' as phoneType, count(*) as count FROM disCat4 WHERE phoneType='iPhone X' " +
170     "group by loc order by count desc limit 10")
171   val r2 = sqlContext.sql("SELECT loc, 'iPhone7 Series' as phoneType, count(*) as count FROM disCat4 WHERE phoneType='iPhone7 Series' " +
172     "group by loc order by count desc limit 10")
173   val r3 = sqlContext.sql("SELECT loc, 'iPhone8 Series' as phoneType, count(*) as count FROM disCat4 WHERE phoneType='iPhone8 Series' " +
174     "group by loc order by count desc limit 10")
175   val r4 = sqlContext.sql("SELECT loc, 'Galaxy' as phoneType, count(*) as count FROM disCat4 WHERE phoneType='Galaxy' " +
176     "group by loc order by count desc limit 10")
177   val r5 = sqlContext.sql("SELECT loc, 'Redmi' as phoneType, count(*) as count FROM disCat4 WHERE phoneType='Redmi' " +
178     "group by loc order by count desc limit 10")
179   val r6 = sqlContext.sql("SELECT loc, 'OnePlus' as phoneType, count(*) as count FROM disCat4 WHERE phoneType='OnePlus' " +
180     "group by loc order by count desc limit 10")
181   val r7 = sqlContext.sql("SELECT loc, 'Moto' as phoneType, count(*) as count FROM disCat4 WHERE phoneType='Moto' " +
182     "group by loc order by count desc limit 10")
183   val r8 = sqlContext.sql("SELECT loc, 'HTC' as phoneType, count(*) as count FROM disCat4 WHERE phoneType='HTC' " +
184     "group by loc order by count desc limit 10")
185   val r9 = sqlContext.sql("SELECT loc, 'BlackBerry' as phoneType, count(*) as count FROM disCat4 WHERE phoneType='BlackBerry' " +
186     "group by loc order by count desc limit 10")
187   val rddl = r1.union(r2).union(r3).union(r4).union(r5).union(r6).union(r7).union(r8).union(r9)
188     rddl.createOrReplaceTempView("rddl")
189   val res=sqlContext.sql("SELECT phoneType, Count(*) as Count from rddl where phoneType is not null group by phoneType")
190   println("*****")
191   println("Model Type")
192   res.show()
193
194 -----Query 6: Popular languages used for tweeting tweets about phones -----
195
196 case '6' =>
197   val langWstCount = sqlContext.sql("SELECT distinct id, " +
198     "CASE when user.lang LIKE 'en%' then 'English' "+
199     "when user.lang LIKE 'ja%' then 'Japanese' "+
200     "when user.lang LIKE 'es%' then 'Spanish' "+
201     "when user.lang LIKE 'fr%' then 'French' "+
202     "when user.lang LIKE 'it%' then 'Italian' "+

```

## Output:

```

154 | null
155 | end as day1 from day_data where day is not null""'.stripMargin)
156
157 days_final.createOrReplaceTempView("days_final")
158
159 val res = sqlContext.sql("SELECT day1 as Day,Count(*) as Day_Count from days_final where day1 is not null group by day1 order by count")
160
161 queries > main(args: Array[String])

```

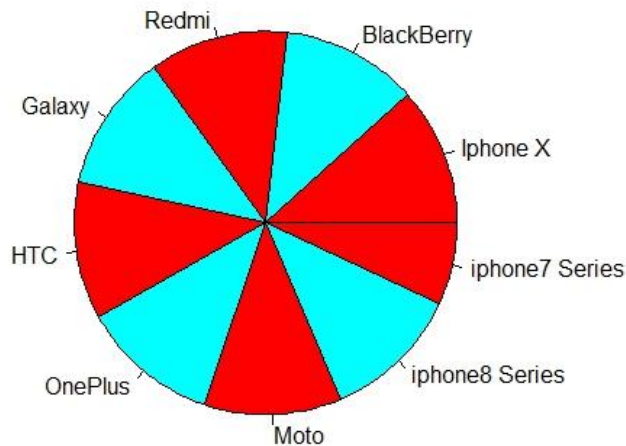
Run: queries

```

17/11/14 20:42:40 INFO DAGScheduler: ResultStage 40 (show at queries.scala:191) finished in 0.640 s
17/11/14 20:42:40 INFO DAGScheduler: Job 2 finished: show at queries.scala:191, took 0.689976 s
17/11/14 20:42:40 INFO CodeGenerator: Code generated in 5.951667 ms
17/11/14 20:42:40 INFO SparkContext: Invoking stop() from shutdown hook
-----+-----
| phoneType|Count|
|-----+-----|
| iPhone X| 10|
| iPhone 7 Series| 10|
| BlackBerry| 10|
| Redmi| 10|
| Galaxy| 10|
| HTC| 10|
| OnePlus| 10|
| Moto| 10|
| iPhone 8 Series| 10|
| iPhone 7 Series| 6|
-----+-----
17/11/14 20:42:40 INFO SparkUI: Stopped Spark web UI at http://192.168.0.15:4040
17/11/14 20:42:40 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
17/11/14 20:42:43 INFO MemoryStore: MemoryStore cleared
17/11/14 20:42:43 INFO BlockManager: BlockManager stopped
17/11/14 20:42:43 INFO BlockManagerMaster: BlockManagerMaster stopped
17/11/14 20:42:43 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
17/11/14 20:42:43 INFO SparkContext: Successfully stopped SparkContext
17/11/14 20:42:43 INFO ShutdownHookManager: Shutdown hook called
17/11/14 20:42:43 INFO ShutdownHookManager: Deleting directory C:\Users\prata\AppData\Local\Temp\spark-3b7a2656-1e34-4d95-b2df-3aee7c8fd0be
Process finished with exit code 0

```

## Visualization:

**Tweets Count for different phone Series****Query 6: Query for fetching Popular languages used for tweeting tweets about phones.**

**Answer:** This query is built to analyze the tweets – popular languages used for tweeting tweets about phones.

As we collected the data related to mobile phones

```
#iphone', '#Samsung', '#Moto', '#Redmi', '#Xiaomi', '#Nokia', '#lenovo', '#oppo', '#OnePlus', '#BlackBerry', '#HTC
```

Such that built a query to analyze the popular languages used. So, it results the count – of most used language by the users.

**Code:**

```

192 case "6" =>
193   val langWstCount = sqlContext.sql("SELECT distinct id," +
194     "CASE when user.lang LIKE 'en%' then 'English'"+
195     "when user.lang LIKE 'ja%' then 'Japanese'"+
196     "when user.lang LIKE 'es%' then 'Spanish'"+
197     "when user.lang LIKE 'fr%' then 'French'"+
198     "when user.lang LIKE 'it%' then 'Italian'"+
199     "when user.lang LIKE 'ru%' then 'Russian'"+
200     "when user.lang LIKE 'ar%' then 'Arabic'"+
201     "when user.lang LIKE 'bn%' then 'Bengali'"+
202     "when user.lang LIKE 'cs%' then 'Czech'"+
203     "when user.lang LIKE 'da%' then 'Danish'"+
204     "when user.lang LIKE 'de%' then 'German'"+
205     "when user.lang LIKE 'el%' then 'Greek'"+
206     "when user.lang LIKE 'fa%' then 'Persian'"+
207     "when user.lang LIKE 'fi%' then 'Finnish'"+
208     "when user.lang LIKE 'fil%' then 'Filipino'"+
209     "when user.lang LIKE 'he%' then 'Hebrew'"+
210     "when user.lang LIKE 'hi%' then 'Hindi'"+
211     "when user.lang LIKE 'hu%' then 'Hungarian'"+
212     "when user.lang LIKE 'id%' then 'Indonesian'"+
213     "when user.lang LIKE 'ko%' then 'Korean'"+
214     "when user.lang LIKE 'ms%' then 'Malay'"+
215     "when user.lang LIKE 'nl%' then 'Dutch'"+
216     "when user.lang LIKE 'no%' then 'Norwegian'"+
217     "when user.lang LIKE 'pl%' then 'Polish'"+
218     "when user.lang LIKE 'pt%' then 'Portuguese'"+
219     "when user.lang LIKE 'ro%' then 'Romanian'"+
220     "when user.lang LIKE 'sv%' then 'Swedish'"+
221     "when user.lang LIKE 'th%' then 'Thai'"+
222     "when user.lang LIKE 'tr%' then 'Turkish'"+
223     "when user.lang LIKE 'uk%' then 'Ukrainian'"+
224     "when user.lang LIKE 'ur%' then 'Urdu'"+
225     "when user.lang LIKE 'vi%' then 'Vietnamese'"+
226     "when user.lang LIKE 'zh-cn%' then 'Chinese (Simplified)'"+
227     "when user.lang LIKE 'zh-tw%' then 'Chinese (Traditional)'"+
228     "END AS language from tweets where text is not null")
229   langWstCount.createOrReplaceTempView("langWstCount")
230   queries > main(args:Array[String])

```

## Output:

```

154 | null
155 | end as day1 from day_data where day is not null"".stripMargin)
156
157 days_final.createOrReplaceTempView("days_final")
158
159 val res = sqlContext.sql("SELECT day1 as Day_Count(*) as Day_Count from days_final where day1 is not null group by day1 order by count")
160
queries > main(args:Array[String])

```

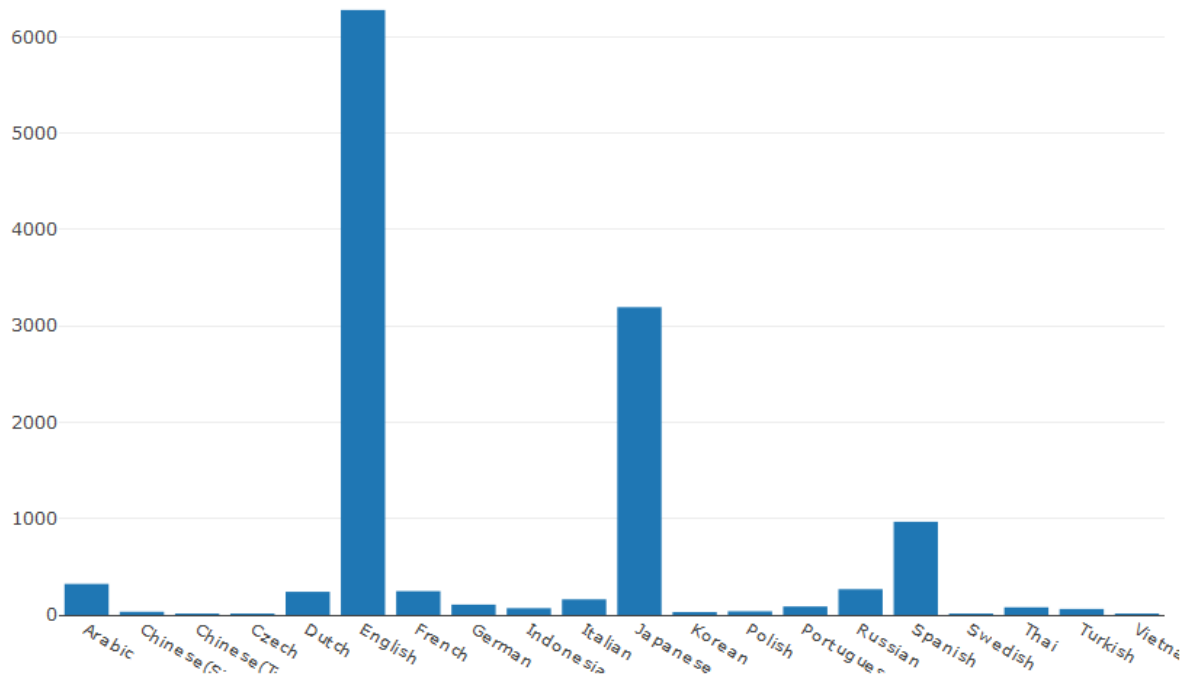
```

17/11/14 20:44:30 INFO TaskSchedulerImpl: Removed taskset 3.0, whose tasks have all completed, from pool
17/11/14 20:44:30 INFO DAGScheduler: ResultStage 3 (show at queries.scala:236) finished in 0.601 s
17/11/14 20:44:30 INFO DAGScheduler: Job 1 finished: show at queries.scala:236, took 0.051158 s
17/11/14 20:44:30 INFO CodeGenerator: Code generated in 0.242772 ms
-----+-----+
| language|Count|
-----+-----+
| English| 6279|
| Japanese| 3194|
| Spanish| 367|
| Arabic| 322|
| Russian| 267|
| French| 247|
| Dutch| 240|
| Italian| 162|
| German| 107|
| Portuguese| 88|
| Thai| 79|
| Indonesian| 70|
| Turkish| 62|
| Polish| 38|
| Chinese (Simplified)| 33|
| Korean| 22|
| Czech| 12|
| Chinese (Traditio...| 8|
| Swedish| 8|
| Vietnamese| 7|
-----+-----+
only showing top 20 rows

17/11/14 20:44:30 INFO SparkContext: Invoking stop() from shutdown hook

```

## Visualization:



#### Query 7: Query for fetching Number of tweets for date.

**Answer:** This query is built to analyze the tweets – based on each individual date how many tweets are posted.

As we collected the data related to mobile phones

`#iphone', '#Samsung', '#Moto', '#Redmi', '#Xiaomi', '#Nokia', '#lenovo', '#oppo', '#OnePlus', '#BlackBerry', '#HTC`

Such that built a query to analyze, depending up on the collected data for each date how many tweets are posted .

**Code:**

```

130 langWstCount.createOrReplaceTempView("langWstCount")
131 var langWstDataCount=sqlContext.sql("SELECT language, Count(language) as Count from langWstCount where id is NOT NULL and language is not null")
132
133 println("*****")
134 println("Language")
135 println("*****")
136 langWstDataCount.show()
137
138 /*-----Query 7 number of tweets for particular date -----*/
139 case "7" =>
140 val tweetcount=sqlContext.sql("SELECT SUBSTR(created_at, 0, 10) tweet_date, COUNT(1) tweet_count FROM tweets GROUP BY SUBSTR(created_at, 0, 10)")
141 tweetcount.createOrReplaceTempView("tweetcount")
142 println("*****")
143 println("tweet Count")
144 println("*****")
145 tweetcount.show()
146
147 /*-----Query 8 Account Verification tweets -----*/
148 case "8" =>
149 val acctVerify=sqlContext.sql("SELECT distinct id, " +
150 "CASE when user_verified LIKE 'false' THEN 'NON-VERIFIED ACCOUNT' "+
151 "END AS Verified from tweets where text is not null")
152 acctVerify.createOrReplaceTempView("acctVerify")
153 var acctVerifyData=sqlContext.sql("SELECT Verified, Count(Verified) as Count from acctVerify where id is NOT NULL and Verified is not null")
154 println("*****")
155 println("Account Verification")
156 println("*****")
157 acctVerifyData.show()
158
159 /*-----Query 9 On Which hours More Tweets Were Done -----*/
160 case "9" =>
161 val timehour = sqlContext.sql("SELECT SUBSTRING(created_at,12,2) as hour from tweets where text is not null")
162 timehour.createOrReplaceTempView("timehour")
163
164 val timeAnalysis=sqlContext.sql("SELECT Case
165 |when hour>=0 and hour <4 then 'midnight'
166 |when hour>=4 and hour <7 then 'early Morning'
167 |when hour>=7 and hour <12 then 'Morning'
168 |when hour>=12 and hour <15 then 'afternoon'
169 END AS hour from timehour")
170 timeAnalysis.createOrReplaceTempView("timeAnalysis")
171 timeAnalysis.show()
172
173 queries > main(args:Array[String])

```

## Output:

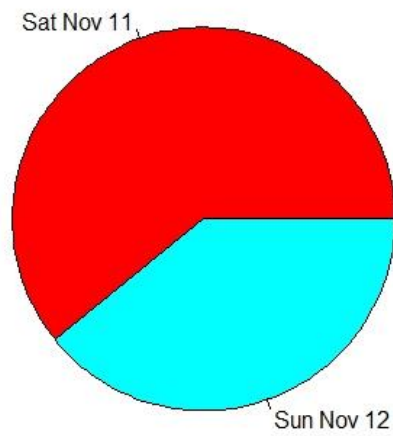
```

17/11/14 20:46:23 INFO TaskSetManager: Finished task 104.0 in stage 2.0 (TID 187) in 119 ms on localhost (197/200)
17/11/14 20:46:23 INFO Executor: Finished task 123.0 in stage 2.0 (TID 204). 4002 bytes result sent to driver
17/11/14 20:46:23 INFO Executor: Finished task 42.0 in stage 2.0 (TID 203). 3899 bytes result sent to driver
17/11/14 20:46:23 INFO Executor: Finished task 130.0 in stage 2.0 (TID 205). 4002 bytes result sent to driver
17/11/14 20:46:23 INFO TaskSetManager: Finished task 42.0 in stage 2.0 (TID 203) in 36 ms on localhost (196/200)
17/11/14 20:46:23 INFO TaskSetManager: Finished task 123.0 in stage 2.0 (TID 204) in 32 ms on localhost (199/200)
17/11/14 20:46:23 INFO TaskSetManager: Finished task 130.0 in stage 2.0 (TID 205) in 18 ms on localhost (200/200)
17/11/14 20:46:23 INFO DAGSchedulerImpl: Removed TaskSet 2.0, whose tasks have all completed, from pool
17/11/14 20:46:23 INFO DAGScheduler: ResultStage 2 (show at queries.scala:244) finished in 1.117 s
17/11/14 20:46:23 INFO DAGScheduler: Job 1 finished: show at queries.scala:244, took 1.764210 s
17/11/14 20:46:23 INFO CodeGenerator: Code generated in 7.225926 ms
+-----+
|tweet_date|tweet_count|
+-----+
|Sat Nov 11| 7464|
|Sun Nov 12| 4805|
| null| 47|
+-----+
17/11/14 20:46:23 INFO SparkContext: Invoking stop() from shutdown hook
17/11/14 20:46:23 INFO SparkUI: Stopped Spark web UI at http://192.168.0.15:4040
17/11/14 20:46:23 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
17/11/14 20:46:23 INFO MemoryStore: MemoryStore cleared
17/11/14 20:46:23 INFO BlockManager: BlockManager stopped
17/11/14 20:46:23 INFO BlockManagerMaster: BlockManagerMaster stopped
17/11/14 20:46:23 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
17/11/14 20:46:23 INFO SparkContext: Successfully stopped SparkContext
17/11/14 20:46:23 INFO ShutdownHookManager: Shutdown hook called
17/11/14 20:46:23 INFO ShutdownHookManager: Deleting directory C:\Users\prata\AppData\Local\Temp\spark-4406e049-4aa0-49b7-b11d-7db688967f5c

```

## Visualization:



**Tweets Count for dates****Query 8: Query for fetching Account Verified tweets.**

**Answer:** This query is built to analyze the tweets – based on account verified tweets

As we collected the data related to mobile phones

`#iphone', '#Samsung', '#Moto', '#Redmi', '#Xiaomi', '#Nokia', '#lenovo', '#oppo', '#OnePlus', '#BlackBerry', '#HTC`

Such that built a query to analyze how many users are verified users. The accounts mainly differentiated by official check. Thus, counting the tweets which are posted from official accounts.

**Code:**

```

130 langWstCount.createOrReplaceTempView("langWstCount")
131 var langWstDataCount=sqlContext.sql("SELECT language, Count(language) as Count from langWstCount where id is NOT NULL and language is not null")
132
133 println("=====")
134 println("Language")
135 println("=====")
136 langWstDataCount.show()
137
138 /*-----Query 7 number of tweets for particular date -----*/
139
140 case "7" =>
141 val tweetcount=sqlContext.sql("SELECT SUBSTR(created_at, 0, 10) tweet_date, COUNT(1) tweet_count FROM tweets GROUP BY SUBSTR(created_at, 0, 10)")
142 tweetcount.createOrReplaceTempView("tweetcount")
143 println("=====")
144 println("tweet Count")
145 println("=====")
146 tweetcount.show()
147
148 /*-----Query 8 Account Verification tweets -----*/
149
150 case "8" =>
151 val acctVerify=sqlContext.sql("SELECT distinct id, " +
152 "CASE when user_verified LIKE 'false' THEN 'NON-VERIFIED ACCOUNT' "+
153 "END AS Verified from tweets where text is not null")
154 acctVerify.createOrReplaceTempView("acctVerify")
155 var acctVerifyData=sqlContext.sql("SELECT Verified, Count(Verified) as Count from acctVerify where id is NOT NULL and Verified is not null")
156 println("=====")
157 println("Account Verification")
158 println("=====")
159 acctVerifyData.show()
160
161 /*-----Query 9 On Which hours More Tweets Were Done -----*/
162
163 case "9" =>
164 val timehour = sqlContext.sql("SELECT SUBSTRING(created_at,12,2) as hour from tweets where text is not null")
165 timehour.createOrReplaceTempView("timehour")
166
167 val timeAnalysis=sqlContext.sql("SELECT Case
168 |when hour>=0 and hour <4 then 'midnight'
169 |when hour>=4 and hour <7 then 'early Morning'
170 |when hour>=7 and hour <12 then 'Morning'
171 |when hour>=12 and hour <15 then 'afternoon'
172 |")
173 timeAnalysis.createOrReplaceTempView("timeAnalysis")
174 timeAnalysis.show()
175
176 queries > main(args:Array[String])
  
```

## Output:

```

17/11/14 20:50:25 INFO Executor: Finished task 157.0 in stage 3.0 (TID 401). 4559 bytes result sent to driver
17/11/14 20:50:25 INFO TaskSetManager: Finished task 157.0 in stage 3.0 (TID 401) in 12 ms on localhost (158/200)
17/11/14 20:50:25 INFO Executor: Finished task 130.0 in stage 3.0 (TID 404). 4757 bytes result sent to driver
17/11/14 20:50:25 INFO TaskSetManager: Finished task 130.0 in stage 3.0 (TID 404) in 18 ms on localhost (155/200)
17/11/14 20:50:25 INFO Executor: Finished task 161.0 in stage 3.0 (TID 405). 4765 bytes result sent to driver
17/11/14 20:50:25 INFO TaskSetManager: Finished task 161.0 in stage 3.0 (TID 405) in 51 ms on localhost (200/200)
17/11/14 20:50:25 INFO TaskSchedulerImpl: Removed TaskSet 3.0, whose tasks have all completed, from pool
17/11/14 20:50:25 INFO DAGScheduler: ResultStage 3 (show at queries.scala:257) finished in 0.560 s
17/11/14 20:50:25 INFO DAGScheduler: Job 1 finished: show at queries.scala:257, took 6.232292 s
=====
| Verified|Count|
|-----|-----|
|NON-VERIFIED ACCOUNT|12221|
| VERIFIED ACCOUNT| 44|
=====

17/11/14 20:50:25 INFO CodeGenerator: Code generated in 5.746598 ms
17/11/14 20:50:25 INFO SparkContext: Invoking stop() from shutdown hook
17/11/14 20:50:25 INFO SparkUI: Stopped Spark web UI at http://192.168.0.15:4040
17/11/14 20:50:25 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
17/11/14 20:50:26 INFO MemoryStore: MemoryStore cleared
17/11/14 20:50:26 INFO BlockManager: BlockManager stopped
17/11/14 20:50:26 INFO BlockManagerMaster: BlockManagerMaster stopped
17/11/14 20:50:26 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
17/11/14 20:50:26 INFO SparkContext: Successfully stopped SparkContext
17/11/14 20:50:26 INFO ShutdownHookManager: Shutdown hook called
17/11/14 20:50:26 INFO ShutdownHookManager: Deleting directory C:\Users\prata\AppData\Local\Temp\spark-aaf6a703-aeef-4f8b-80de-fa02bb029c23

Process finished with exit code 0
  
```

## Visualization:





### Query 9: Query for fetching On Which Hours More Tweets Were Done.

**Answer:** This query is built to analyze the tweets – based on hours of the tweets.

As we collected the data related to mobile phones

```
#iphone', '#Samsung', '#Moto', '#Redmi', '#Xiaomi', '#Nokia', '#lenovo', '#oppo', '#OnePlus', '#BlackBerry', '#HTC
```

Such that built a query to analyze when the tweets are tweeted like-Morning, afternoon, night. So ,it results the time analysis of tweets

**Code:**

The image shows a screenshot of the IntelliJ IDEA IDE interface. The top bar indicates the project is 'untitled' and the current file is 'queries.scala'. The left sidebar shows the project structure, including folders like 'src', 'main', 'scala', and 'queries'. The main editor displays a Scala file 'queries.scala' containing SQL queries for tweet analysis. The bottom status bar shows the current file is 'queries.scala' and the project is 'untitled'. The code in the editor includes comments like 'Query 9 On Which hours More Tweets Were Done' and 'Query 10 Which state is mostly having tweets about type of phone'. The code uses SparkContext and SQLContext to execute queries and create temporary views.

**Output:**

```

17/11/14 20:51:30 INFO Executor: Finished task 199.0 in stage 2.0 (TID 205). 4056 bytes result sent to driver
17/11/14 20:51:30 INFO TaskSetManager: Finished task 199.0 in stage 2.0 (TID 205) in 12 ms on localhost (199/200)
17/11/14 20:51:30 INFO Executor: Finished task 196.0 in stage 2.0 (TID 199). 3732 bytes result sent to driver
17/11/14 20:51:30 INFO TaskSetManager: Finished task 196.0 in stage 2.0 (TID 199) in 47 ms on localhost (200/200)
17/11/14 20:51:30 INFO TaskSchedulerImpl: Removed TaskSet 2.0, whose tasks have all completed, from pool
17/11/14 20:51:30 INFO DAGScheduler: ResultStage 2 (show at queries.scala:280) finished in 1.042 s
17/11/14 20:51:30 INFO DAGScheduler: Job 1 finished: show at queries.scala:280, took 1.757369 s
17/11/14 20:51:30 INFO CodeGenerator: Code generated in 15.818041 ms

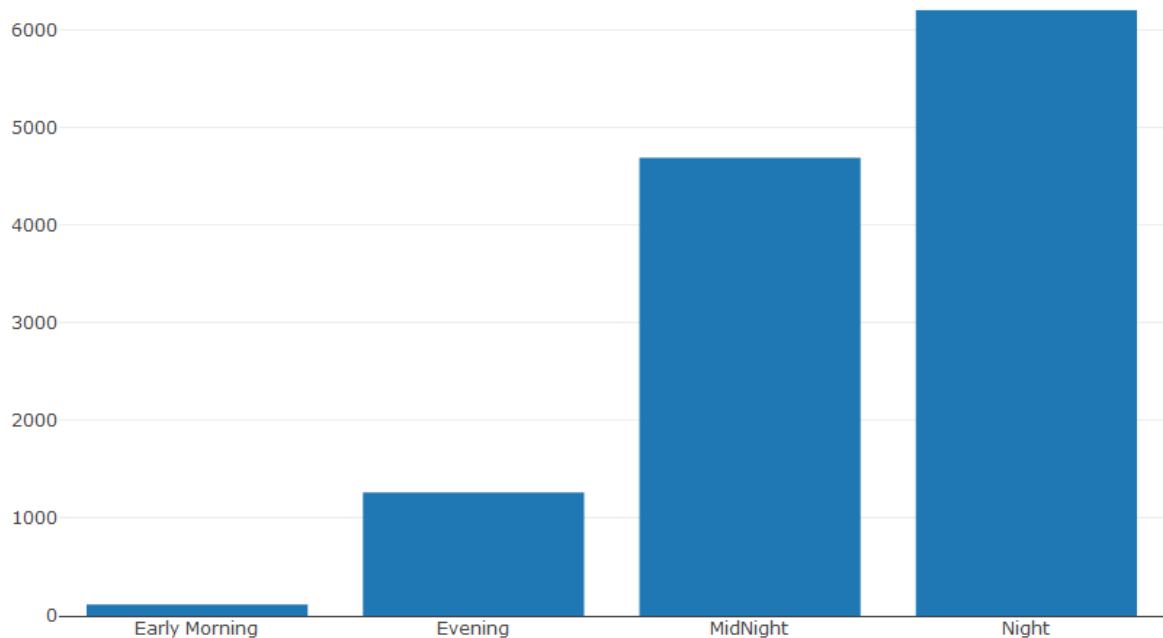
+-----+
|      hour[tweets_count]      |
+-----+
|      night|         6201|
|  midnight|         4689|
|    evening|         1263|
|early Morning|          116|
+-----+

17/11/14 20:51:30 INFO SparkContext: Invoking stop() from shutdown hook
17/11/14 20:51:30 INFO SparkUI: Stopped Spark web UI at http://192.168.0.15:4040
17/11/14 20:51:30 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
17/11/14 20:51:30 INFO MemoryStore: MemoryStore cleared
17/11/14 20:51:30 INFO BlockManager: BlockManager stopped
17/11/14 20:51:30 INFO BlockManagerMaster: BlockManagerMaster stopped
17/11/14 20:51:30 INFO OutputCommitCoordinator: OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
17/11/14 20:51:30 INFO SparkContext: Successfully stopped SparkContext
17/11/14 20:51:30 INFO ShutdownHookManager: Shutdown hook called
17/11/14 20:51:30 INFO ShutdownHookManager: Deleting directory C:\Users\prata\AppData\Local\Temp\spark-fa872145-53f2-4530-a157-e5f69218f0aa

Process finished with exit code 0

```

### Visualization:



**Query 10: Query for fetching Which state is mostly having tweets about type of phone.**

**Answer:** This query is built to analyze the tweets – **Which state is mostly having tweets about type of phone**

As we collected the data related to mobile phones  
 #iphone', '#Samsung', '#Moto', '#Redmi', '#Xiaomi', '#Nokia', '#lenovo', '#oppo', '#OnePlus', '#BlackBerry', '#HTC

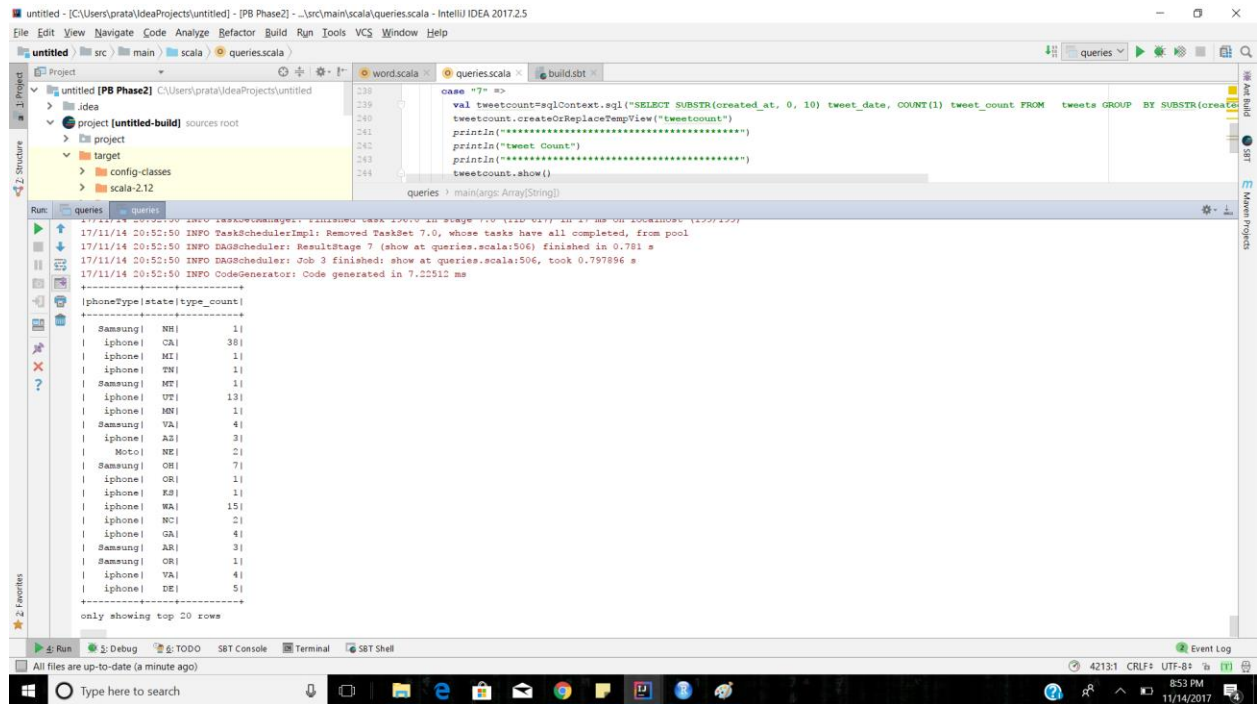
Such that built a query to perform

**Code:**

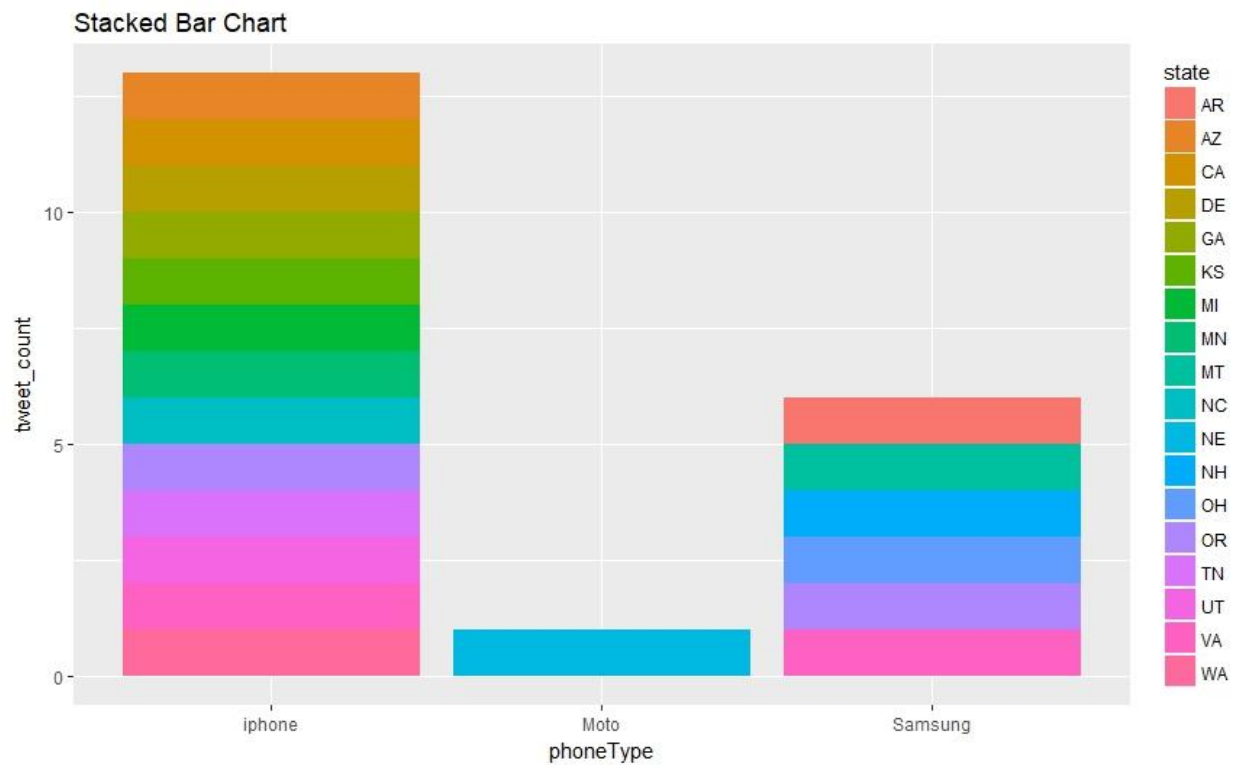
```

279 //println("***** how many tweets were found *****")
280 println("*****")
281 res.show()
282
283 //-----Query 10 Which state is mostly having tweets about type of phone -----
284
285 case "10" =>
286   val iphoneRDD = sqlContext.sql("SELECT 'iphone' as phoneType, user.location as loc from tweets where text LIKE '%#iphone%'")
287   val SamsungRDD = sqlContext.sql("SELECT 'Samsung' as phoneType, user.location as loc from tweets where text LIKE '%#Samsung%'")
288   val MotoRDD = sqlContext.sql("SELECT 'Moto' as phoneType, user.location as loc from tweets where text LIKE '%#Moto%'")
289   val RedmiRDD = sqlContext.sql("SELECT 'Redmi' as phoneType, user.location as loc from tweets where text LIKE '%#Redmi%'")
290   val XiaomiRDD = sqlContext.sql("SELECT 'Xiaomi' as phoneType, user.location as loc from tweets where text LIKE '%#Xiaomi%'")
291   val NokiaRDD = sqlContext.sql("SELECT 'Nokia' as phoneType, user.location as loc from tweets where text LIKE '%#Nokia%'")
292   val lenovoRDD = sqlContext.sql("SELECT 'lenovo' as phoneType, user.location as loc from tweets where text LIKE '%#lenovo%'")
293   val oppoRDD = sqlContext.sql("SELECT 'oppo' as phoneType, user.location as loc from tweets where text LIKE '%#oppo%'")
294   val OnePlusRDD = sqlContext.sql("SELECT 'OnePlus' as phoneType, user.location as loc from tweets where text LIKE '%#OnePlus%'")
295   val BlackBerryRDD = sqlContext.sql("SELECT 'BlackBerry' as phoneType, user.location as loc from tweets where text LIKE '%#BlackBerry%'")
296   val HTC RDD = sqlContext.sql("SELECT 'HTC' as phoneType, user.location as loc from tweets where text LIKE '%#HTC%'")
297   //val brunchRDD = sqlContext.sql("SELECT 'Brunch' as MealType, SUBSTRING(created_at,12,2) as hour, user.location as loc from dfs")
298   //val brunchRDD = sqlContext.sql("SELECT 'Brunch' as MealType, SUBSTRING(created_at,12,2) as hour, user.location as loc from dfs")
299   val sqlRDD = iphoneRDD.union(SamsungRDD).union(MotoRDD).union(RedmiRDD).union(XiaomiRDD).union(NokiaRDD).union(lenovoRDD).union(oppoRDD).union(OnePlusRDD).union(BlackBerryRDD).union(HTC RDD)
300   sqlRDD.createOrReplaceTempView("sqlRDD")
301   val locate = sqlContext.sql(
302     """
303     SELECT phoneType, loc from sqlRDD where
304     (loc LIKE '%Alaska%' OR loc LIKE '%Arizona%' OR loc LIKE '%Arkansas%' OR loc LIKE '%California%' OR loc LIKE '%Colorado%' OR
305     loc LIKE '%Florida%'
306     OR loc LIKE '%Georgia%'
307     OR loc LIKE '%Hawaii%'
308     OR loc LIKE '%Idaho%'
309     OR loc LIKE '%Illinois%'
310     OR loc LIKE '%Indiana%'
311     OR loc LIKE '%Iowa%'
312     OR loc LIKE '%Kansas%'
313     OR loc LIKE '%Kentucky%'
314     OR loc LIKE '%Louisiana%'
315     OR loc LIKE '%Maine%'
316     OR loc LIKE '%Maryland%'
317     OR loc LIKE '%Massachusetts%'
318     OR loc LIKE '%Michigan%'
319     OR loc LIKE '%Minnesota%'
320     OR loc LIKE '%Mississippi%'
321     OR loc LIKE '%Missouri%'
322     OR loc LIKE '%Montana%'
323     OR loc LIKE '%Nebraska%'
324     OR loc LIKE '%Nevada%'
325     OR loc LIKE '%New Hampshire%'
326     OR loc LIKE '%New Jersey%'
327     OR loc LIKE '%New Mexico%'
328     OR loc LIKE '%New York%'
329     )
330     """
331   )
332   queries = mainArgs: Array[String]
  
```

**Output:**



### Visualization:



## **Source Code output log files link (Shared via drive)**

[https://drive.google.com/open?id=1YjOF\\_-tB8eqBvxPjePwLqpeDTFspDw5A](https://drive.google.com/open?id=1YjOF_-tB8eqBvxPjePwLqpeDTFspDw5A)