Principles of Big Data Management
Phrase-2

Spring 2019

Submitted by:

Tiancheng Xie
Zeng Yue
Josh Oguntimehin

Analyzing and Visualizing Twitter data on Mobiles

Objective:

Applications/Software's Used: Scala, Apache Spark SQL, Vegas, Twitter API, Python.

Collecting tweets from Twitter:

- Firstly, we have created a developer account in Twitter using below link. https://apps.twitter.com/
- 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.

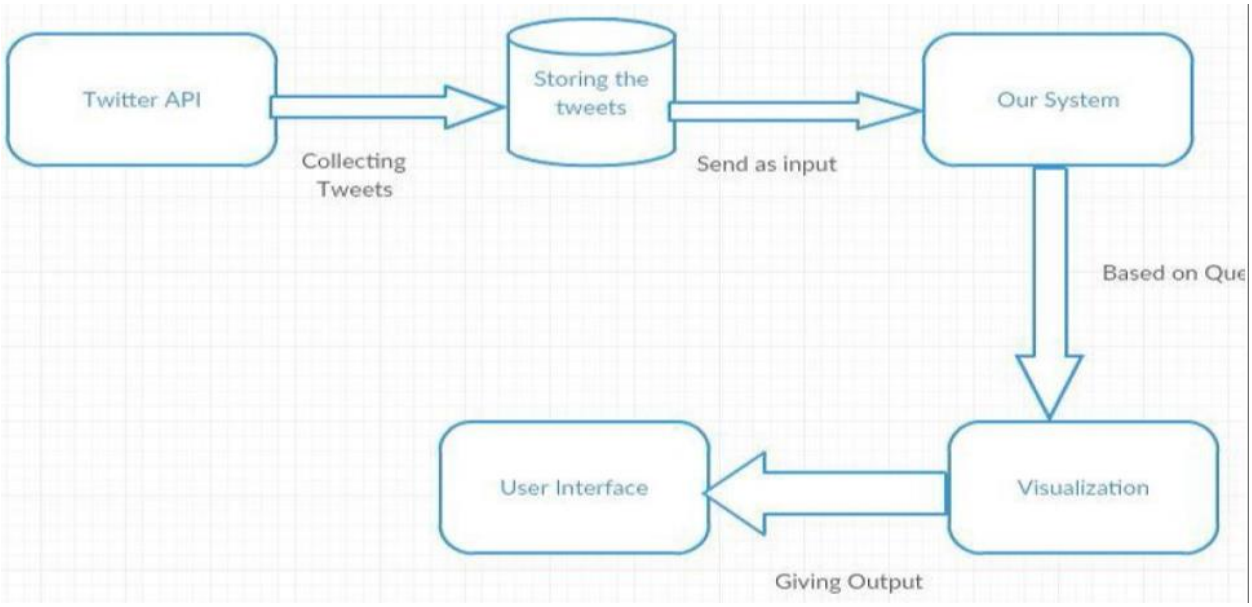
```
Sample tweets are collected for the key words by using python program
#iphone','#Samsung','#Moto','#Redmi','#Xiaomi','#Nokia','#lenovo','#oppo','#OnePlus','
#BlackBerry','#HTC
```

1) Using spark to stream the tweet,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 realized by using Vegas which is an awesome resource for matplotlib.

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\u2019t 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":"C0DEED",

"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,

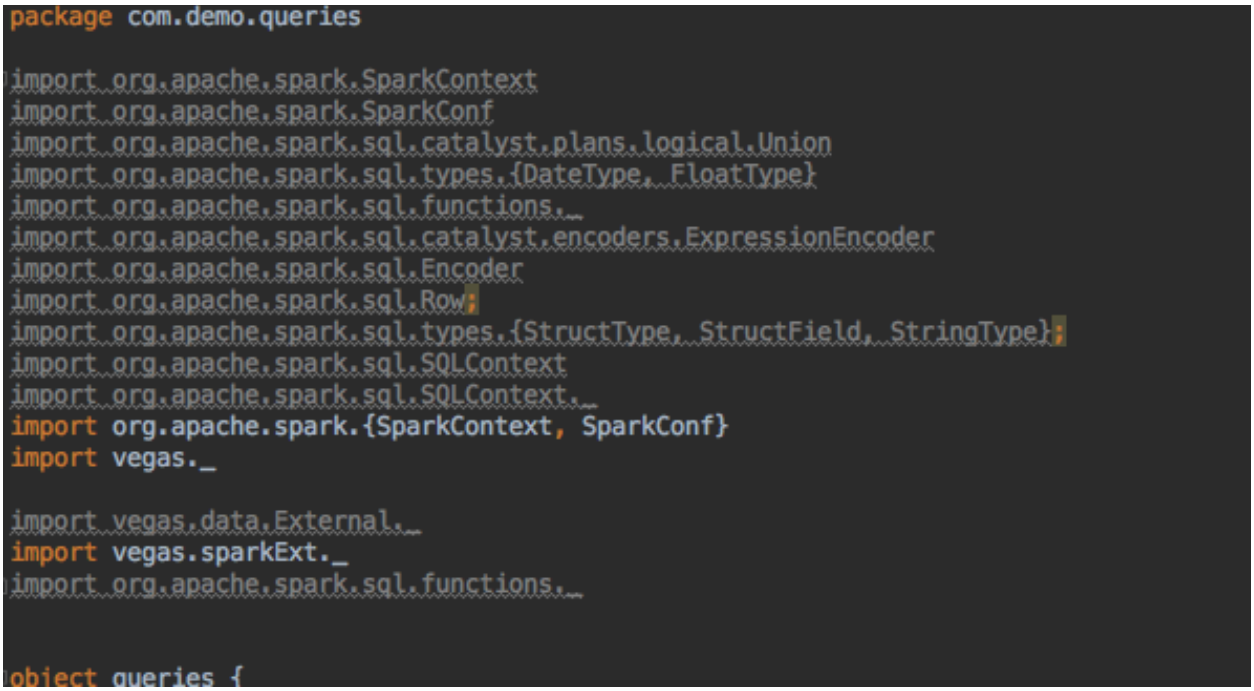
"quoted_status_id":927672133739835392,..

}

2)Queries and outputs:

1.Illustraing the library we are using in this lab for reference

Code:

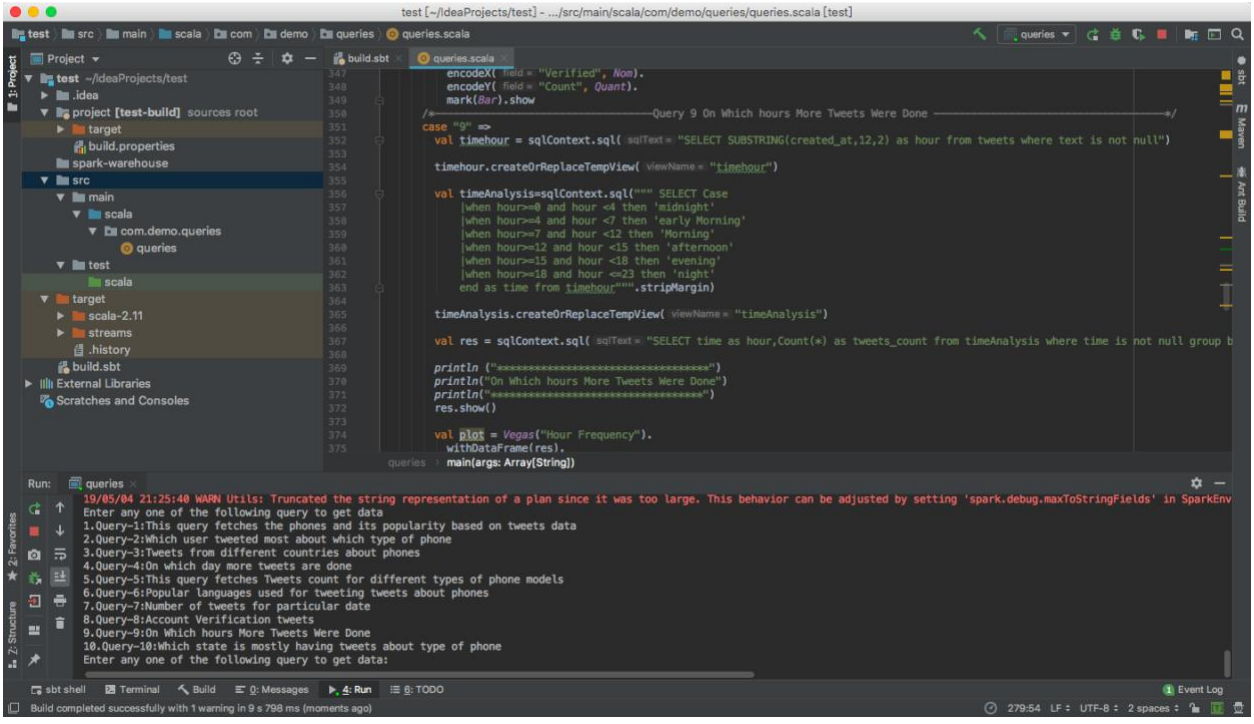


```
package com.demo.queries

import org.apache.spark.SparkContext
import org.apache.spark.SparkConf
import org.apache.spark.sql.catalyst.plans.logical.Union
import org.apache.spark.sql.types.{DateType, FloatType}
import org.apache.spark.sql.functions._
import org.apache.spark.sql.catalyst.encoders.ExpressionEncoder
import org.apache.spark.sql.Encoder
import org.apache.spark.sql.Row;
import org.apache.spark.sql.types.{StructType, StructField, StringType};
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.SQLContext._
import org.apache.spark.{SparkContext, SparkConf}
import vegas._

import vegas.data.External._
import vegas.sparkExt._
import org.apache.spark.sql.functions._


object queries {
```

2.The Menu of the program, for reading the source file, selecting the query we want to process.

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

    val sparkConf = new SparkConf().setAppName("SparkWordCount").setMaster("local[*]")

    val sc = new SparkContext(sparkConf)

    // Contains SQLContext which is necessary to execute SQL queries
    val sqlContext = new org.apache.spark.sql.SQLContext(sc)

    // Reads json file and stores in a variable
    //val tweet = sqlContext.read.json("C:\\Documents\\PB Phase 2\\tweetshastag.txt")
    val tweet = sqlContext.read.json( path = "/Users/hanson/Desktop/LEARNING/Big data/Principals-of-Big-Data-Project/Project Phase -2/f
    //val outputFile = "C:\\Documents\\outputFile"
    val outputFile = "/Users/hanson/Desktop/LEARNING/Big data/Principals-of-Big-Data-Project/Project Phase -2/outputFile"
    //To register tweets data as a table
    tweet.createOrReplaceTempView( viewName = "tweets")
    val disCat = sqlContext.sql( sqlText = "SELECT user.name as UserName,user.location as loc,text,created_at," +
      "CASE WHEN text like '%iphone%' THEN 'IPHONE'" +
      "WHEN text like '%Samsung%' THEN 'SAMSUNG'" +
      "WHEN text like '%Moto%' THEN 'MOTO'" +
      "WHEN text like '%Redmi%' THEN 'REDMI'" +
      "WHEN text like '%Xiaomi%' THEN 'XIAOMI'" +
      "WHEN text like '%Nokia%' THEN 'NOKIA'" +
      "WHEN text like '%lenovo%' THEN 'LENOVO'" +
      "WHEN text like '%oppo%' THEN 'OPPO'" +
      "WHEN text like '%OnePlus%' THEN 'ONEPLUS'" +
      "WHEN text like '%BlackBerry%' THEN 'BLACKBERRY'" +
      "WHEN text like '%HTC%' THEN 'HTC'" +
      "END AS phoneType from tweets where text is not null")
    disCat.createOrReplaceTempView( viewName = "disCat2")
    val disCat3 = sqlContext.sql( sqlText = "SELECT user.name as UserName,user.location as loc,text,created_at," +
      "CASE WHEN text like '%IphoneX%' OR text like '%iphoneX%' OR text like '%Iphonex%' OR text like '%iphonex%' OR text like '%Iphor
      "WHEN text like '%iphone7' OR text like '%iphone7plus%' OR text like '%iPHONE7' OR text like '%iPHONE 7' OR text like '%iphone 7
      "WHEN text like '%iphone8' OR text like '%iPHONE 8' OR text like '%iphone 8' OR text like '%iphone8plus%' OR text like '%iPHONE8
      "WHEN text like '%galaxy%' OR text like '%Galaxy%' THEN 'Galaxy'" +
      "WHEN text like '%Xiaomi%' OR text like '%xiaomi%'  OR text like '%Redmi%'  OR text like '%Redmi%' THEN 'Redmi'" +
      "WHEN text like '%oneplus%' OR text like '%OnePlus%'  OR text like '%ONEPLUS%' THEN 'OnePlus'" +
      "WHEN text like '%moto%' OR text like '%MOTO%' THEN 'Moto'" +
      "WHEN text like '%htc%' OR text like '%Htc%' OR text like '%HTC%' THEN 'HTC'" +
      "WHEN text like '%BlackBerry%' OR text like '%blackberry%' THEN 'BlackBerry'" +
      "END AS phoneType from tweets where text is not null")
    disCat3.createOrReplaceTempView( viewName = "disCat4")
```

Output:



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

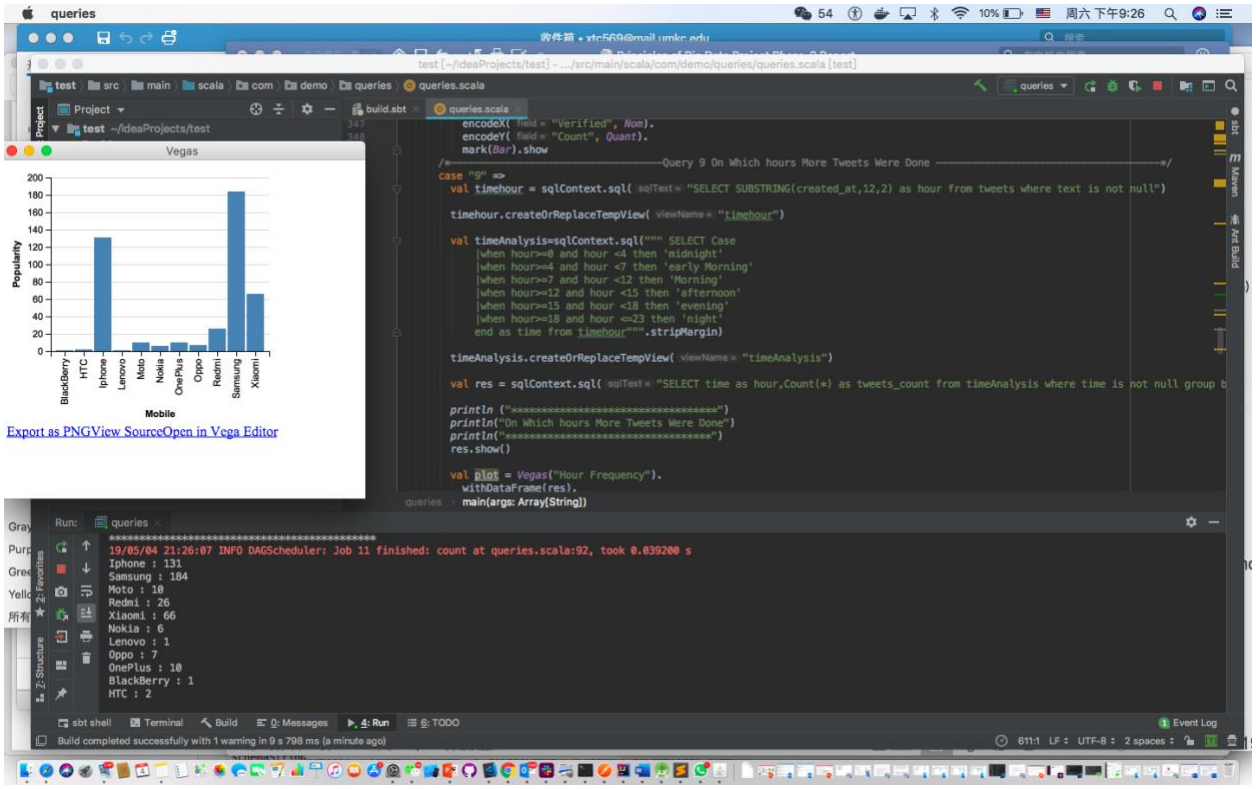
Such that built a query to perform the count operation on each model. So, it results the count -how many times a phone model appears in the tweets.

Code:

```
count match {
  case "1" =>
    /*----------Query 1: This query fetches the phones and its popularity based on tweets data----------*/
    val textFile = sc.textFile( path = "/Users/hanson/Desktop/LEARNING/Big data/Principals-of-Big-Data-Project/Project Phase -2/fet
    val iphone = (textFile.filter(line => line.contains("#iphone")).count())
    val Samsung = (textFile.filter(line => line.contains("#Samsung")).count())
    val Moto = (textFile.filter(line => line.contains("#Moto")).count())
    val Redmi = (textFile.filter(line => line.contains("#Redmi")).count())
    val Xiaomi = (textFile.filter(line => line.contains("#Xiaomi")).count())
    val Nokia = (textFile.filter(line => line.contains("#Nokia")).count())
    val lenovo = (textFile.filter(line => line.contains("#lenovo")).count())
    val oppo = (textFile.filter(line => line.contains("#oppo")).count())
    val OnePlus = (textFile.filter(line => line.contains("#OnePlus")).count())
    val BlackBerry = (textFile.filter(line => line.contains("#BlackBerry")).count())
    val HTC = (textFile.filter(line => line.contains("#HTC")).count())
    println("**********************************************")
    println("Number of tweets on different types of phones")
    println("**********************************************")
    println("Iphone : %s".format(iphone))
    println("Samsung : %s".format(Samsung))
    println("Moto : %s".format(Moto))
    println("Redmi : %s".format(Redmi))
    println("Xiaomi : %s".format(Xiaomi))
    println("Nokia : %s".format(Nokia))
    println("Lenovo : %s".format(lenovo))
    println("Oppo : %s".format(oppo))
    println("OnePlus : %s".format(OnePlus))
    println("BlackBerry : %s".format(BlackBerry))
    println("HTC : %s".format(HTC))
```

Output (Include visualization):

```
**********************************************
Number of tweets on different types of phones
**********************************************
Iphone : 131
Samsung : 184
Moto : 10
Redmi : 26
Xiaomi : 66
Nokia : 6
Lenovo : 1
Oppo : 7
OnePlus : 10
BlackBerry : 1
HTC : 2
```



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

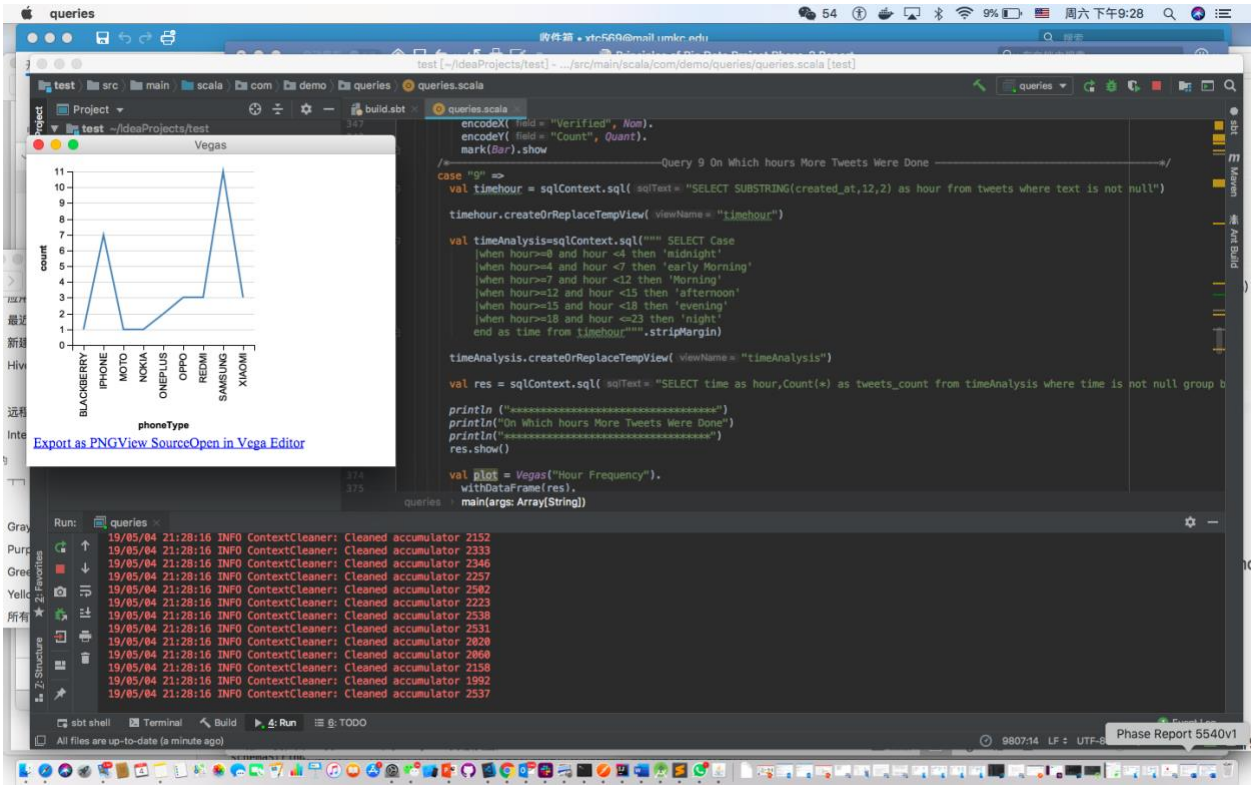
Such that built a query to perform the user tweeted most about which type of phone. So, it results the count -how many times a user tweeted at most for each mobile phone.

Code:

```
/*------------------------------Query 2:  Which user tweeted most about which type of phone-----------------------------------------
case "2" =>

  val r1 = sqlContext.sql( sqlText = "SELECT UserName,'IPHONE' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='IPHONE
    "group by UserName order by count desc limit 1")
  val r2 = sqlContext.sql( sqlText = "SELECT UserName,'SAMSUNG' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='SAMSU
    "group by UserName order by count desc limit 1 ")
  val r3 = sqlContext.sql( sqlText = "SELECT UserName,'MOTO' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='MOTO' "
    "group by UserName order by count desc limit 1 ")
  val r4 = sqlContext.sql( sqlText = "SELECT UserName,'REDMI' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='REDMI'
    "group by UserName order by count desc limit 1 ")
  val r5 = sqlContext.sql( sqlText = "SELECT UserName,'XIAOMI' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='XIAOMI
    "group by UserName order by count desc limit 1 ")
  val r6 = sqlContext.sql( sqlText = "SELECT UserName,'NOKIA' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='NOKIA'
    "group by UserName order by count desc limit 1 ")
  val r7 = sqlContext.sql( sqlText = "SELECT UserName,'LENOVO' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='LENOVO
    "group by UserName order by count desc limit 1 ")
  val r8 = sqlContext.sql( sqlText = "SELECT UserName,'OPPO' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='OPPO' "
    "group by UserName order by count desc limit 1 ")
  val r9 = sqlContext.sql( sqlText = "SELECT UserName,'ONEPLUS' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='ONEPL
    "group by UserName order by count desc limit 1")
  val r10 = sqlContext.sql( sqlText = "SELECT UserName,'BLACKBERRY' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='
    "group by UserName order by count desc limit 1")
  val r11 = sqlContext.sql( sqlText = "SELECT UserName,'HTC' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='HTC' " +
    "group by UserName order by count desc limit 1 ")

  val rdd1 = r1.union(r2).union(r3).union(r4).union(r5).union(r6).union(r7).union(r8).union(r9) union(r10).union(r11)


  println("*********************************************")
  println("Which user tweeted more on which type of phone")
  println("*********************************************")
  rdd1.show()
```

Output(include visualization):

```
+--------------------+----------+-----+
|            UserName| phoneType|count|
+--------------------+----------+-----+
|iphoneアプリ超!検索|    IPHONE|    7|
|              박연배|   SAMSUNG|   11|
|       EtabetaWeb|      MOTO|    1|
|      komputeramoh|     REDMI|    3|
|      Sumit dhanuk|    XIAOMI|    3|
|             Reasha|     NOKIA|    1|
|🐻 DianKarelina 🐻|      OPPO|    3|
|   The Phone Talks|   ONEPLUS|    2|
|          Nash Tee|BLACKBERRY|    1|
+--------------------+----------+-----+
```



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

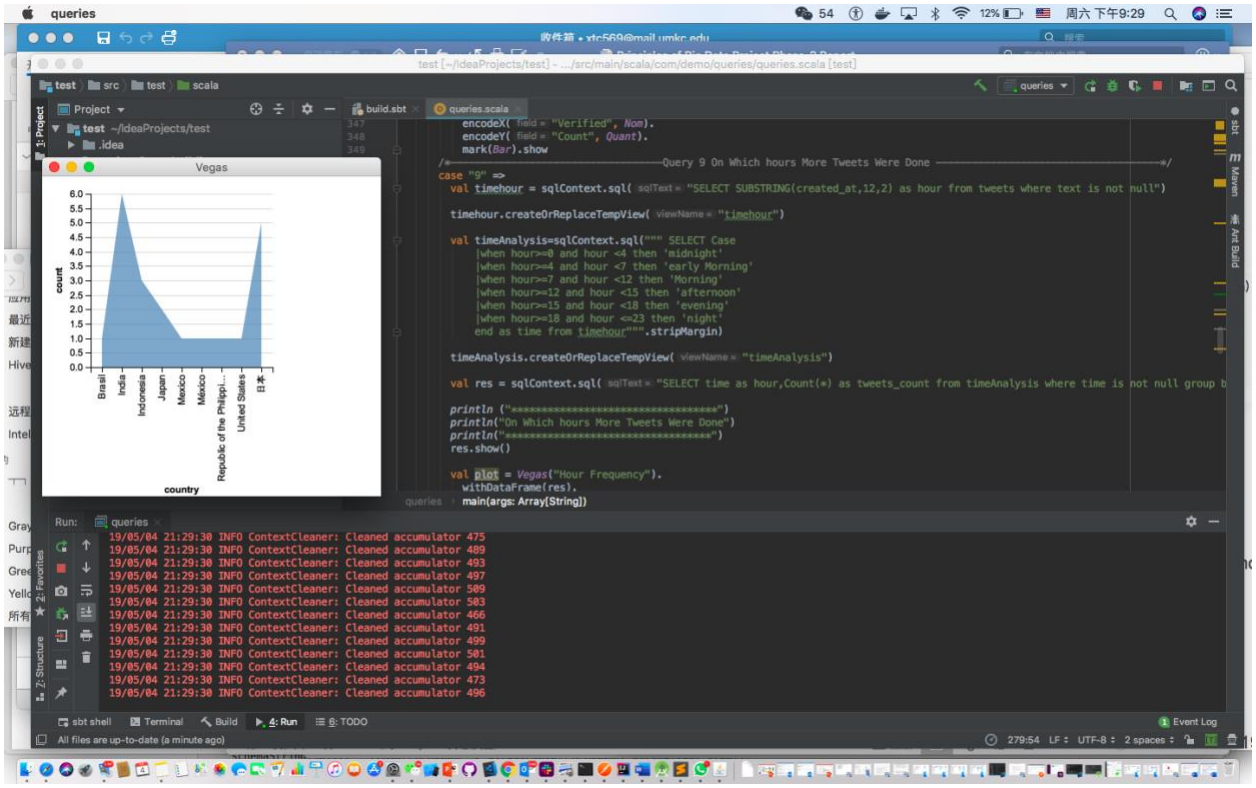
Such that built a query for finding out tweets from different countries about phones. So, it results the count -how many tweets from different countries posted about phones.

Code:

```
/*-------------------------------Query 3: Tweets from different countries about phones -------------------------------*/
case "3" =>
  val countrytweetscount=sqlContext.sql( sqlText = "SELECT distinct place.country, count(*) as count FROM tweets where place.coun
  countrytweetscount.createOrReplaceTempView( viewName = "countrytweetscount")
  println("**********************************************")
  println("Tweets from different countries")
  println("**********************************************")
  countrytweetscount.show()

  Vegas("Hashtags vs Country").
    withDataFrame(countrytweetscount).
    mark(Area).
    encodeX( field = "country", Nom).
    encodeY( field = "count", Quant, scale=Scale(bandSize = 12.0)).
    show
```

Output:

country	count
India	6
日本	5
Indonesia	3
Japan	2
Mexico	1
Republic of the P...	1
Brasil	1
United States	1
México	1



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

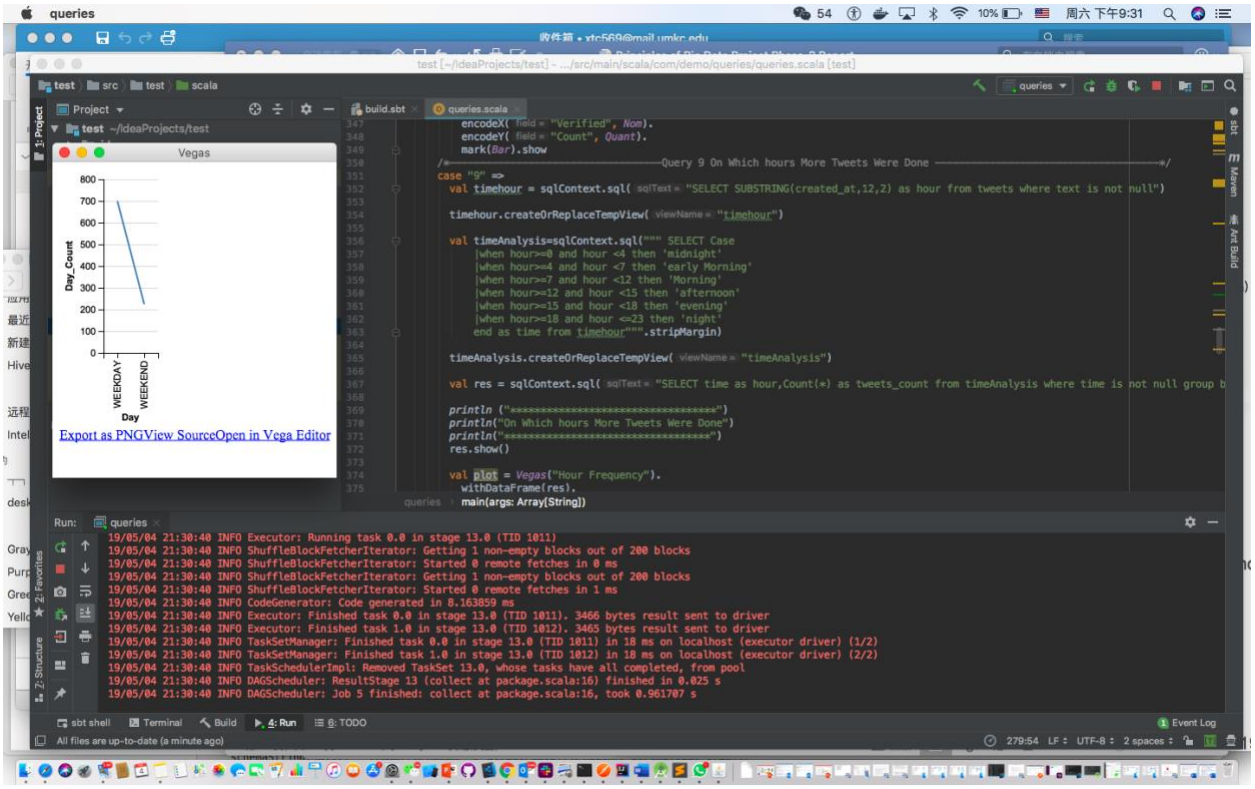
Such that built a query to perform for analyzing on which day the more tweets are posted. So, it results the count – about giving a figure of on what day & how many tweets are done.

Code:

```
case "4" =>
  val day_data = sqlContext.sql( sqlText = "SELECT substring(user.created_at,1,3) as day from tweets where text is not null")

  day_data.createOrReplaceTempView( viewName = "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( viewName = "days_final")

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

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


  Vegas("Tweet day count"). withDataFrame(res).
    mark(Line).
    encodeX( field = "Day", Nom).
    encodeY( field = "Day_Count", Quant). show
```

Output(include visualizaton):

```
+-------+---------+
|    Day|Day_Count|
+-------+---------+
|WEEKDAY|      701|
|WEEKEND|      226|
+-------+---------+
```



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**

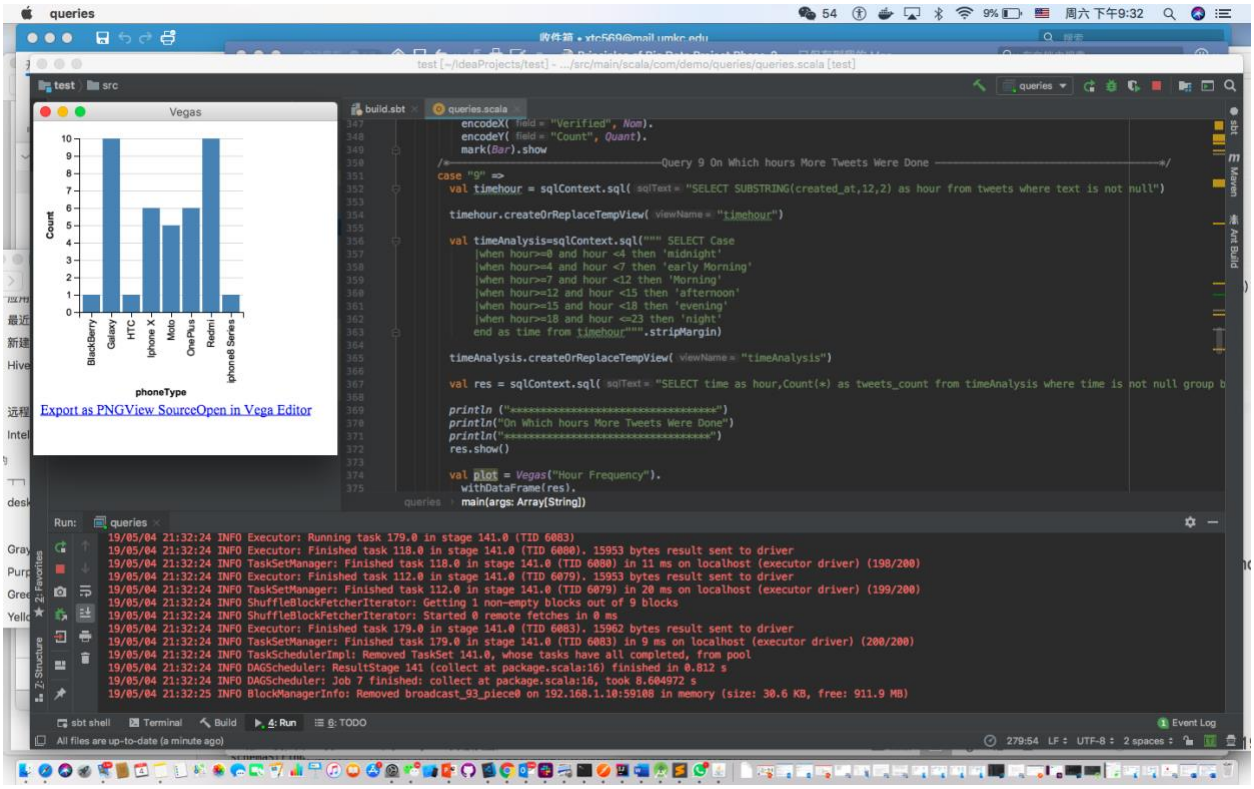
Such that built a query to perform the user tweeted most which type of series for mobile phone. So, it results the tweets count -for different series of phone models.

Code:

```
case "5" =>
  val r1 = sqlContext.sql( sqlText = "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( sqlText = "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( sqlText = "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 r4 = sqlContext.sql( sqlText = "SELECT loc,'Galaxy' as phoneType,count(*) as count FROM disCat4 WHERE phoneType='Galaxy' "
    "group by loc order by count desc limit 10")
  val r5 = sqlContext.sql( sqlText = "SELECT loc,'Redmi' as phoneType,count(*) as count FROM disCat4 WHERE phoneType='Redmi' " +
    "group by loc order by count desc limit 10")
  val r6 = sqlContext.sql( sqlText = "SELECT loc,'OnePlus' as phoneType,count(*) as count FROM disCat4 WHERE phoneType='OnePlus' "
    "group by loc order by count desc limit 10")
  val r7 = sqlContext.sql( sqlText = "SELECT loc,'Moto' as phoneType,count(*) as count FROM disCat4 WHERE phoneType='Moto' " +
    "group by loc order by count desc limit 10")
  val r8 = sqlContext.sql( sqlText = "SELECT loc,'HTC' as phoneType,count(*) as count FROM disCat4 WHERE phoneType='HTC' " +
    "group by loc order by count desc limit 10")
  val r9 = sqlContext.sql( sqlText = "SELECT loc,'BlackBerry' as phoneType,count(*) as count FROM disCat4 WHERE phoneType='BlackBerry' "
    "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( viewName = "rdd1")
  val res=sqlContext.sql( sqlText = "SELECT phoneType, Count(*) as Count from rdd1 where phoneType is not null group by phoneType")
  println("**********************************************")
  println("Model Type")
  println("**********************************************")
  res.show()

  val plot = Vegas("Different models series").
    withDataFrame(res).
    encodeX( field = "phoneType", Nom).
    encodeY( field = "Count", Quant).
    mark(Bar).show
```

Output(Visualization):

phoneType	Count
Iphone X	6
BlackBerry	1
Redmi	10
Galaxy	10
HTC	1
OnePlus	6
Moto	5
iphone8 Series	1



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.

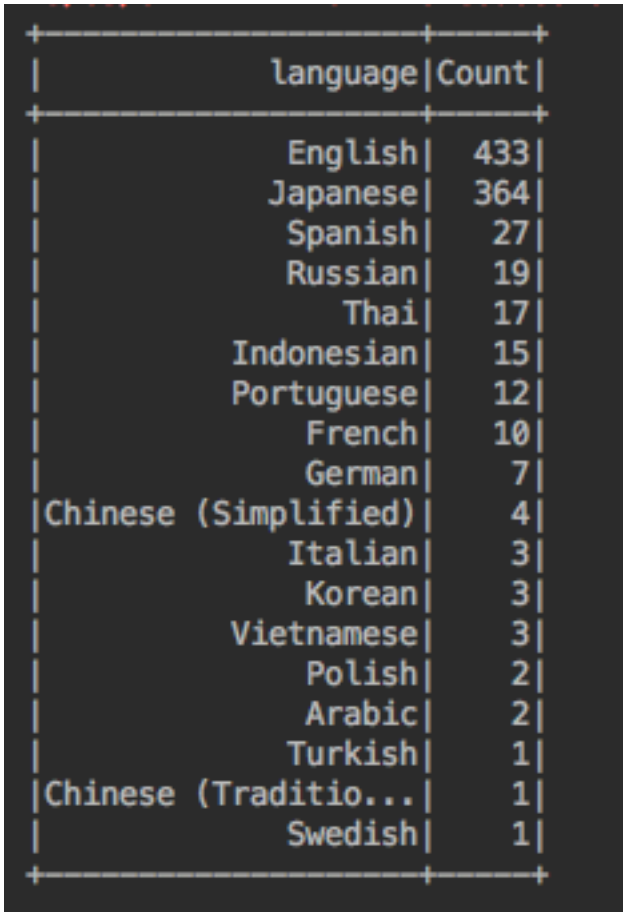
As we collected the data related to mobile phones
#iphone','#Samsung','#Moto','#Redmi','#Xiaomi','#Nokia','#lenovo','#oppo','#OnePlus','#BlackBerry','#HTC

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

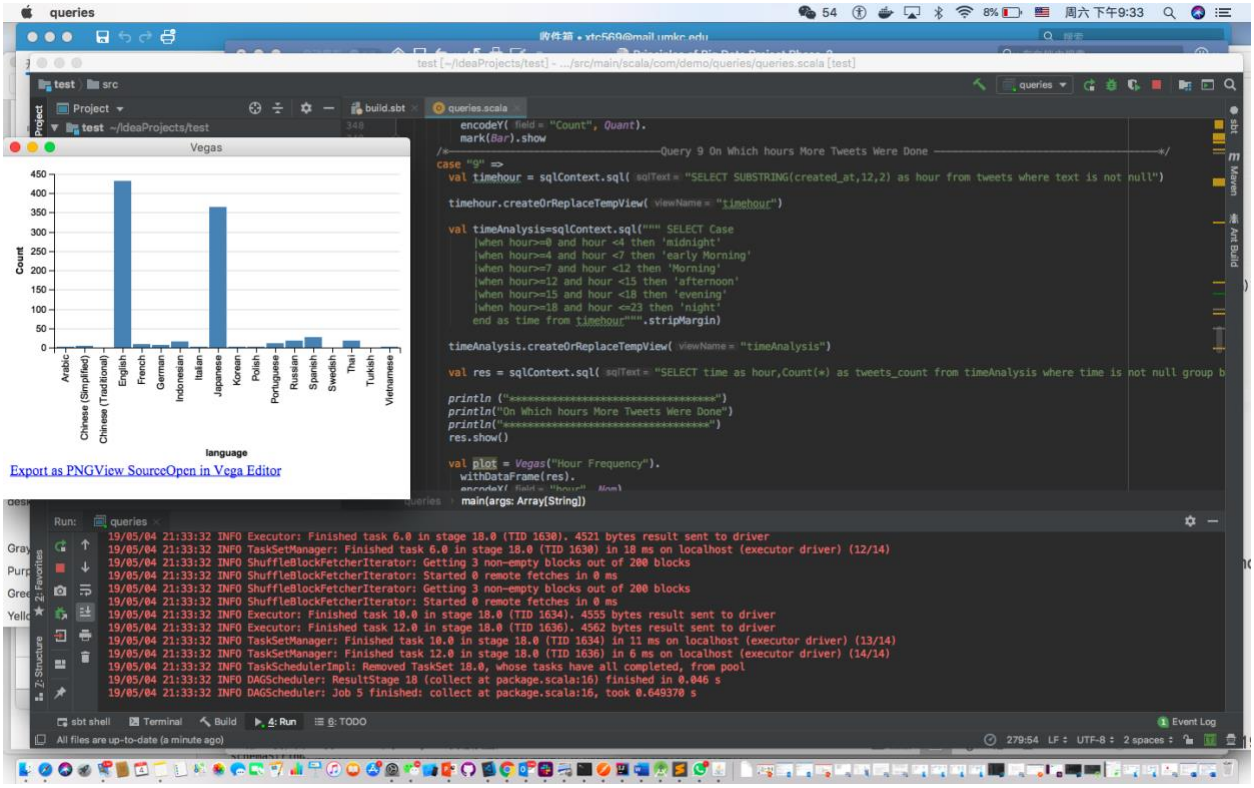
Code:

```
/*------------------------------Query 6 Popular languages used for tweeting tweets about phones ------------------------------*/
case "6" =>
  val langWstCount = sqlContext.sql( sqlText = "SELECT distinct id," +
    "CASE when user.lang LIKE '%en%' then 'English'"+
    "when user.lang LIKE '%ja%' then 'Japanese'"+
    "when user.lang LIKE '%es%' then 'Spanish'"+
    "when user.lang LIKE '%fr%' then 'French'"+
    "when user.lang LIKE '%it%' then 'Italian'"+
    "when user.lang LIKE '%ru%' then 'Russian'"+
    "when user.lang LIKE '%ar%' then 'Arabic'"+
    "when user.lang LIKE '%bn%' then 'Bengali'"+
    "when user.lang LIKE '%cs%' then 'Czech'"+
    "when user.lang LIKE '%da%' then 'Danish'"+
    "when user.lang LIKE '%de%' then 'German'"+
    "when user.lang LIKE '%el%' then 'Greek'"+
    "when user.lang LIKE '%fa%' then 'Persian'"+
    "when user.lang LIKE '%fi%' then 'Finnish'"+
    "when user.lang LIKE '%fil%' then 'Filipino'"+
    "when user.lang LIKE '%he%' then 'Hebrew'"+
    "when user.lang LIKE '%hi%' then 'Hindi'"+
    "when user.lang LIKE '%hu%' then 'Hungarian'"+
    "when user.lang LIKE '%id%' then 'Indonesian'"+
    "when user.lang LIKE '%ko%' then 'Korean'"+
    "when user.lang LIKE '%msa%' then 'Malay'"+
    "when user.lang LIKE '%nl%' then 'Dutch'"+
    "when user.lang LIKE '%no%' then 'Norwegian'"+
    "when user.lang LIKE '%pl%' then 'Polish'"+
    "when user.lang LIKE '%pt%' then 'Portuguese'"+
    "when user.lang LIKE '%ro%' then 'Romanian'"+
    "when user.lang LIKE '%sv%' then 'Swedish'"+
    "when user.lang LIKE '%th%' then 'Thai'"+
    "when user.lang LIKE '%tr%' then 'Turkish'"+
    "when user.lang LIKE '%uk%' then 'Ukrainian'"+
    "when user.lang LIKE '%ur%' then 'Urdu'"+
    "when user.lang LIKE '%vi%' then 'Vietnamese'"+
    "when user.lang LIKE '%zh-cn%' then 'Chinese (Simplified)'"+
    "when user.lang LIKE '%zh-tw%' then 'Chinese (Traditional)'"+
    "END AS language from tweets where text is not null")
  langWstCount.createOrReplaceTempView( viewName = "langWstCount")


  var langWstDataCount=sqlContext.sql( sqlText = "SELECT language, Count(language) as Count from langWstCount where id is NOT NULL a
```

Output(Visualization):



language	Count
English	433
Japanese	364
Spanish	27
Russian	19
Thai	17
Indonesian	15
Portuguese	12
French	10
German	7
Chinese (Simplified)	4
Italian	3
Korean	3
Vietnamese	3
Polish	2
Arabic	2
Turkish	1
Chinese (Traditio...	1
Swedish	1



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.

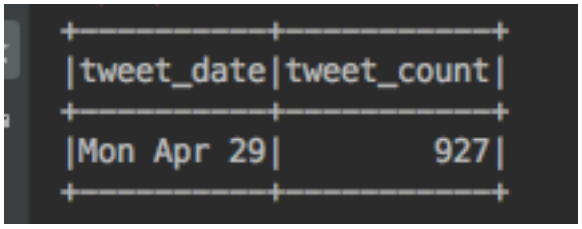
As we collected the data related to mobile phones
#iphone','#Samsung','#Moto','#Redmi','#Xiaomi','#Nokia','#lenovo','#oppo','#OnePlus','#BlackBerry','#HTC

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

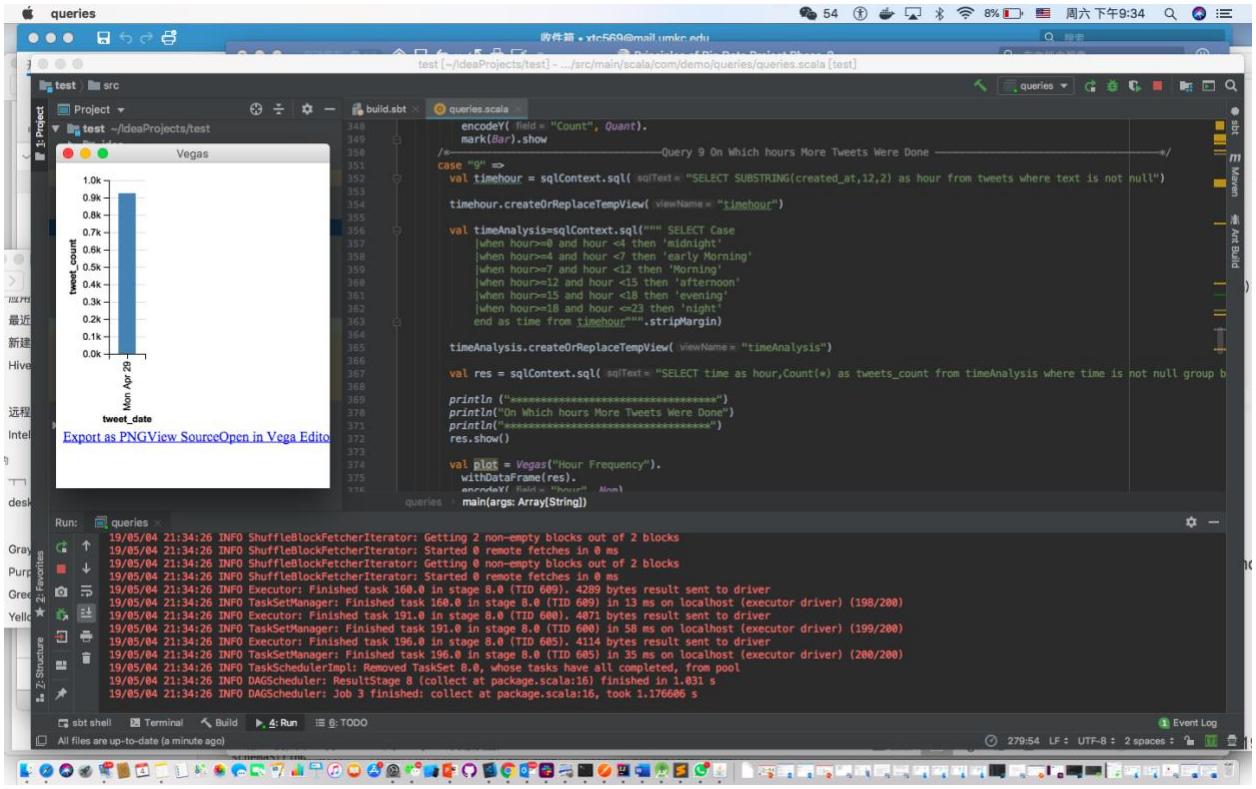
Code:

```
    mark(Bar).show
/*-------------------------Query 7 number of tweets for particular date -------------------------*/
case "7" =>
  val tweetcount=sqlContext.sql( sqlText = "SELECT SUBSTR(created_at, 0, 10) tweet_date, COUNT(1) tweet_count FROM   tweets GROUP
  tweetcount.createOrReplaceTempView( viewName = "tweetcount")
  println("*****************************************")
  println("tweet Count")
  println("*****************************************")
  tweetcount.show()

  Vegas("Particular date tweet"). withDataFrame(tweetcount).
    encodeX( field = "tweet_date", Nom).
    encodeY( field = "tweet_count", Quant).
    mark(Bar).show
/*-------------------------Query 8 Account Verification tweets -------------------------*/
```

Output(Visualization):



tweet_date	tweet_count
Mon Apr 29	927



Query 8: Query for fetching Account Verified tweets.

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

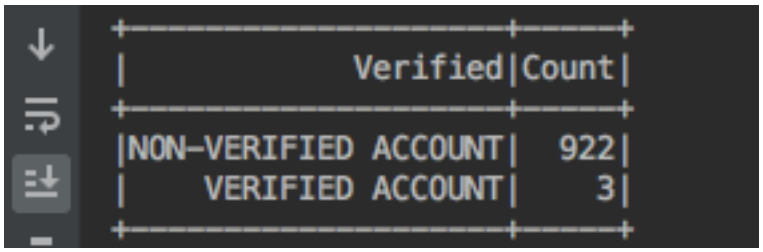
```
As we collected the data related to mobile phones
#iphone','#Samsung','#Moto','#Redmi','#Xiaomi','#Nokia','#lenovo','#oppo','#OnePlus','
#BlackBerry','#HTC
```

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.

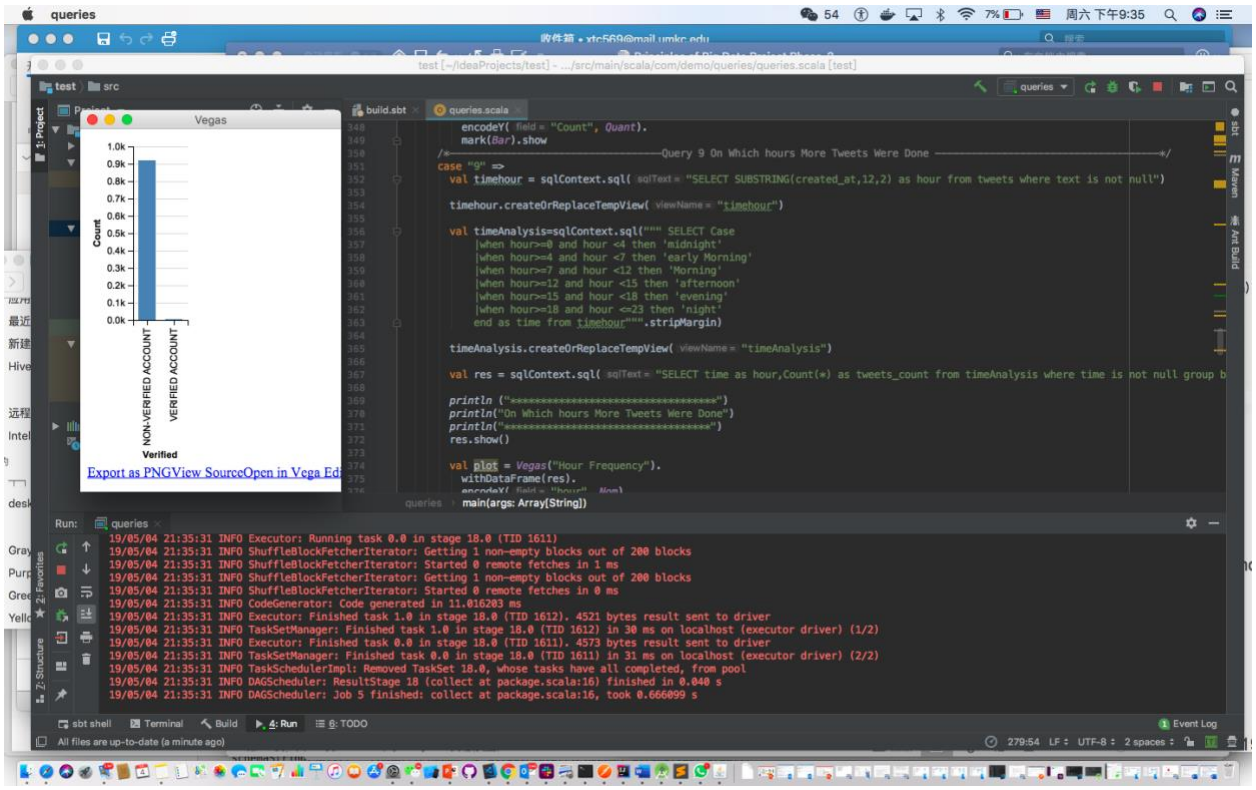
Code:

```
    encodeY( field = "tweet_count", Quant).
    mark(Bar).show
/*-------------------------------Query 8 Account Verification tweets ----------------------------------*/
case "8" =>
  val acctVerify=sqlContext.sql( sqlText = "SELECT distinct id, " +
    "CASE when user.verified LIKE '%true%' THEN 'VERIFIED ACCOUNT'"+
    "when user.verified LIKE '%false%' THEN 'NON-VERIFIED ACCOUNT'"+
    "END AS Verified from tweets where text is not null")
  acctVerify.createOrReplaceTempView( viewName = "acctVerify")
  var acctVerifydata=sqlContext.sql( sqlText = "SELECT  Verified, Count(Verified) as Count from acctVerify where id is NOT NULL

  println("**************************************")
  println("Account Verification")
  println("**************************************")
  acctVerifydata.show()

  Vegas("Account Verification"). withDataFrame(acctVerifydata).
    encodeX( field = "Verified", Nom).
    encodeY( field = "Count", Quant).
    mark(Bar).show
/*-------------------------------Query 9 On Which hours More Tweets Were Done ----------------------------------*/
```

Output(Visualization):



```
+--------------------+-----+
|            Verified|Count|
+--------------------+-----+
|NON-VERIFIED ACCOUNT|  922|
|    VERIFIED ACCOUNT|    3|
+--------------------+-----+
```



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

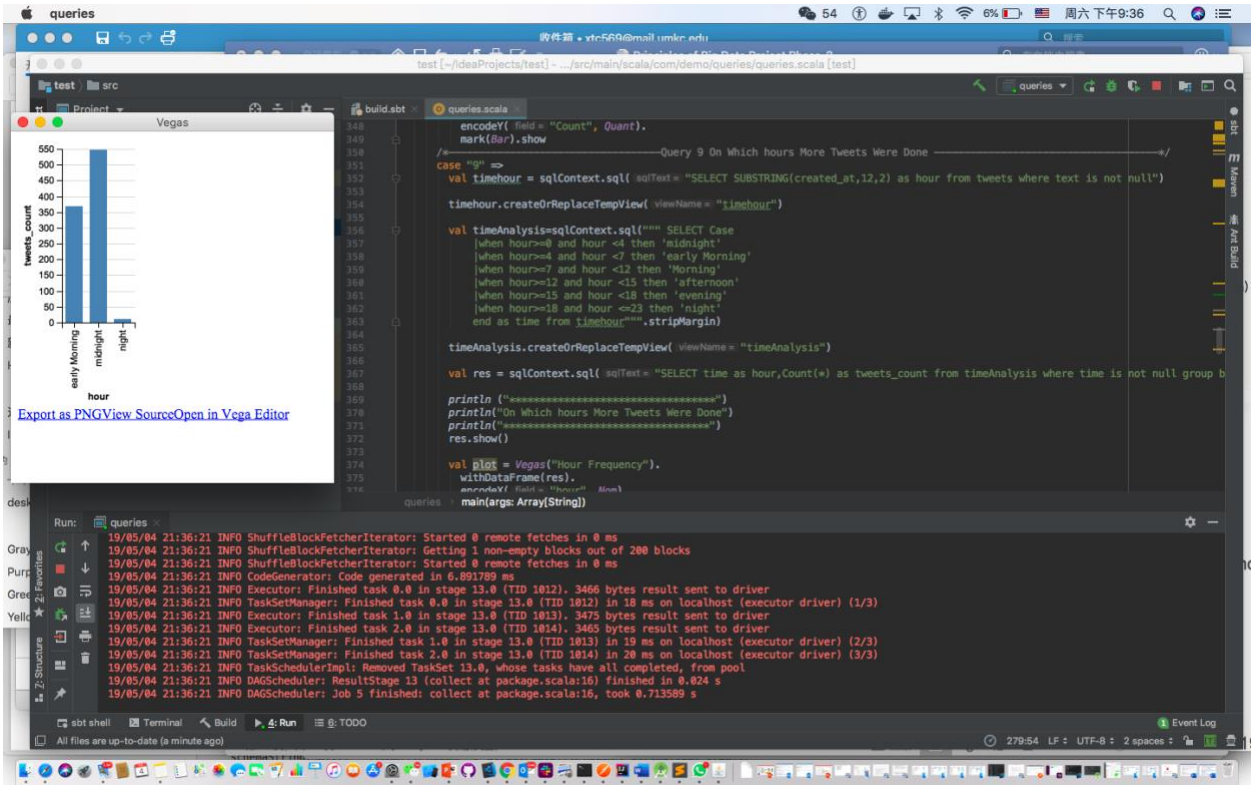
Such that built a query to analyze when the tweets are tweeted like-Morning, afternoon, night. So ,it results the time analysis of tweets

Code:

```
    mark(Bar).show
/*-------------------------------Query 9 On Which hours More Tweets Were Done ---------------------------------*/
case "9" =>
  val timehour = sqlContext.sql( sqlText = "SELECT SUBSTRING(created_at,12,2) as hour from tweets where text is not null")

  timehour.createOrReplaceTempView( viewName = "timehour")

  val timeAnalysis=sqlContext.sql(""" SELECT Case
      |when hour>=0 and hour <4 then 'midnight'
      |when hour>=4 and hour <7 then 'early Morning'
      |when hour>=7 and hour <12 then 'Morning'
      |when hour>=12 and hour <15 then 'afternoon'
      |when hour>=15 and hour <18 then 'evening'
      |when hour>=18 and hour <=23 then 'night'
      end as time from timehour""".stripMargin)

  timeAnalysis.createOrReplaceTempView( viewName = "timeAnalysis")

  val res = sqlContext.sql( sqlText = "SELECT time as hour,Count(*) as tweets_count from timeAnalysis where time is not null gr

  println ("**********************************")
  println("On Which hours More Tweets Were Done")
  println("**********************************")
  res.show()

  val plot = Vegas("Hour Frequency").
    withDataFrame(res).
    encodeX( field = "hour", Nom).
    encodeY( field = "tweets_count", Quant).
    mark(Bar).show
```

Output(Visualization):

```
19/05/04 21:41:14 INFO CodeGenerator: Code g
+--------------+------------+
|          hour|tweets_count|
+--------------+------------+
|      midnight|         548|
|early Morning|         368|
|         night|          11|
+--------------+------------+

19/05/04 21:41:14 INFO SparkContext: Invokir
```



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**

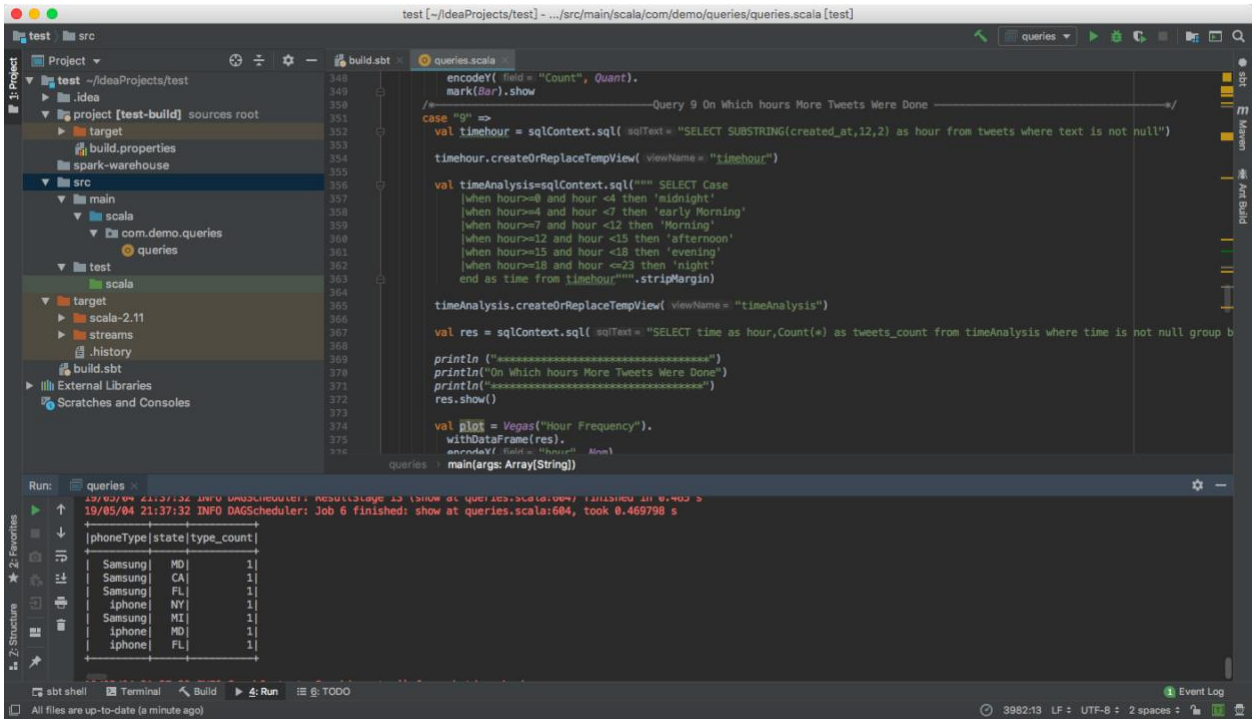
As we collected the data related to mobile phones **#iphone','#Samsung','#Moto','#Redmi','#Xiaomi','#Nokia','#lenovo','#oppo','#OnePlus','#BlackBerry','#HTC**

Such that built a query to perform

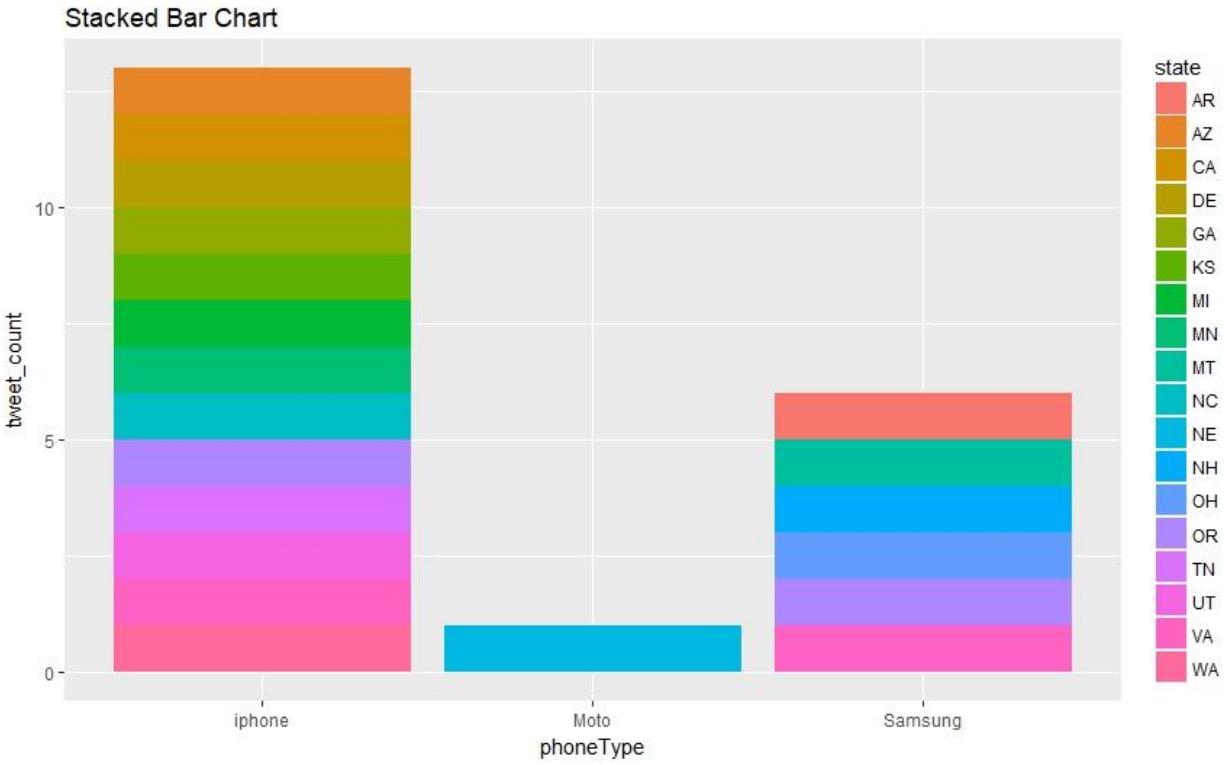
Code:

```
case "10" =>
  val iphoneRDD =  sqlContext.sql( sqlText = """ SELECT 'iphone' as phoneType, user.location as loc from tweets where text LIKE '%
  val SamsungRDD =  sqlContext.sql( sqlText = """ SELECT 'Samsung' as phoneType, user.location as loc from tweets where text LIKE '%
  val MotoRDD =  sqlContext.sql( sqlText = """ SELECT 'Moto' as phoneType, user.location as loc from tweets where text LIKE '%#Moto%
  val OneplusRDD =  sqlContext.sql( sqlText = """ SELECT 'Oneplus' as phoneType, user.location as loc from tweets where text LIKE '%
  //val breakRDD =  sqlContext.sql(""" SELECT 'Breakfast' as Mealtype, SUBSTRING(created_at,12,2) as hour, user.location as loc
  //val brunchRDD =  sqlContext.sql(""" SELECT 'Brunch' as Mealtype, SUBSTRING(created_at,12,2) as hour, user.location as loc from
  val sql2RDD = iphoneRDD.union(SamsungRDD).union(MotoRDD).union(OneplusRDD)
  sql2RDD.createOrReplaceTempView( viewName = "sql2RDD")
  val locate = sqlContext.sql(
    """ SELECT phoneType, loc from sql2RDD  where
      |loc LIKE  '%Alaska%' OR loc LIKE  '%Arizona%' OR loc LIKE  '%Arkansas%' OR loc LIKE  '%California%' OR loc LIKE  '%Colorado
      |OR loc LIKE  '%Florida%'
      |OR loc LIKE  '%Georgia%'
      |OR loc LIKE  '%Hawaii%'
      |OR loc LIKE  '%Idaho%'
      |OR loc LIKE  '%Illinois%'
      |OR loc LIKE  '%Indiana%'
      |OR loc LIKE  '%Iowa%'
      |OR loc LIKE  '%Kansas%'
      |OR loc LIKE  '%Kentucky%'
      |OR loc LIKE  '%Louisiana%'
      |OR loc LIKE  '%Maine%'
      |OR loc LIKE  '%Maryland%'
      |OR loc LIKE  '%Massachusetts%'
      |OR loc LIKE  '%Michigan%'
      |OR loc LIKE  '%Minnesota%'
      |OR loc LIKE  '%Mississippi%'
      |OR loc LIKE  '%Missouri%'
      |OR loc LIKE  '%Montana%'
      |OR loc LIKE  '%Nebraska%'
      |OR loc LIKE  '%Nevada%'
      |OR loc LIKE  '%NewHampshire%'
      |OR loc LIKE  '%NewJersey%'
      |OR loc LIKE  '%NewMexico%'
      |OR loc LIKE  '%NewYork%'
      |OR loc LIKE  '%NorthCarolina%'
      |OR loc LIKE  '%NorthDakota%'
      |OR loc LIKE  '%Ohio%'
      |OR loc LIKE  '%Oklahoma%'
      |OR loc LIKE  '%Oregon%'
      |OR loc LIKE  '%Pennsylvania%'
```

Output:



Visualization:



Reference:

https://spark.apache.org/

https://hadoop.apache.org/

http://link.galegroup.com.contentproxy.phoenix.edu/apps/doc/A487904581/GIC?u=uphoenix_uopx&sid=GIC&xid=ead46665

https://www.vegas-viz.org/

https://github.com/a190884810/CS5540