PHASE – 1

BY

Pravalhika Kampally (Class ID: 12)

Joshmitha Thammareddy (Class ID: 25)

Rupesh Sai Ram Doddala(Class ID: 7)

GOALS

Motivation:

This project is motivated to develop a system to store, analyze and visualize Twitters tweets.

Significance:

It provides a wealth of information that helps to create meaningful tweets that resonates with target audience. Compare followers with different personas, demographics, interests and consumer behaviours to see brand measures up etc. Watch individual Tweet performance, cumulative overview to compare monthly activity etc.

Objectives:

- Work on the tweets related to Search Engine and to figure out how to store them in Spark SQL.
- Write interesting analytical queries to explore and understand the data collected.
- Develop interesting visualizations of written queries.

Features:

Planning to add Sentiment Analysis on the data and analyze whether tweet is positive or negative.

Reference:

https://www.digitalvidya.com/blog/twitter-sentiment-analysis-introduction-and-techniques/

https://www.earthdatascience.org/courses/earth-analytics/get-data-using-apis/use-twitter-api-r/

PHASE 1

OBJECTIVE:

The main purpose of this project is to develop a system to store, analyze and visualize Twitter's tweets. The tasks to be performed in this phase are as follows :

- To work on the tweets related to the recently released mobile phones & their accessories and to figure out how to store them in Spark SQL.
- To write interesting analytical queries to explore and understand the data collected.
- To develop interesting visualizations of the above written queries.

DATASET : Twitter data set(Phones/E-Accessories)

IMPLEMENTATION:

- Initially collected the tweets in JSON format for which a Python program is written, the output of the program contains the tweets with all the details like the IDs, URLs, Hashtags, Created at, Text etc.

- The twitter data is collected on the concept based on to analyse & visualize the data regarding various phone/e-accessories.
- The extracted JSON tweets are persisted into the Apache Spark in the form of tables.
- Query written in Scala language will be sent to spark server and the outputs files are stored in the form of CSV/JSON files.
- These CSV/JSON output files are used to visualize the data using Bar Graphs, Pie Charts through Tableau.
- Key-words used in the tweets extraction are as follows :
  iphone, iphonex, iPhoneXs, iPhoneXR, iPhoneXr, iPhone, #iPhone, AirPods, mobile, watch, technology, Accessories, Mac, iOS, update, music, latest etc.

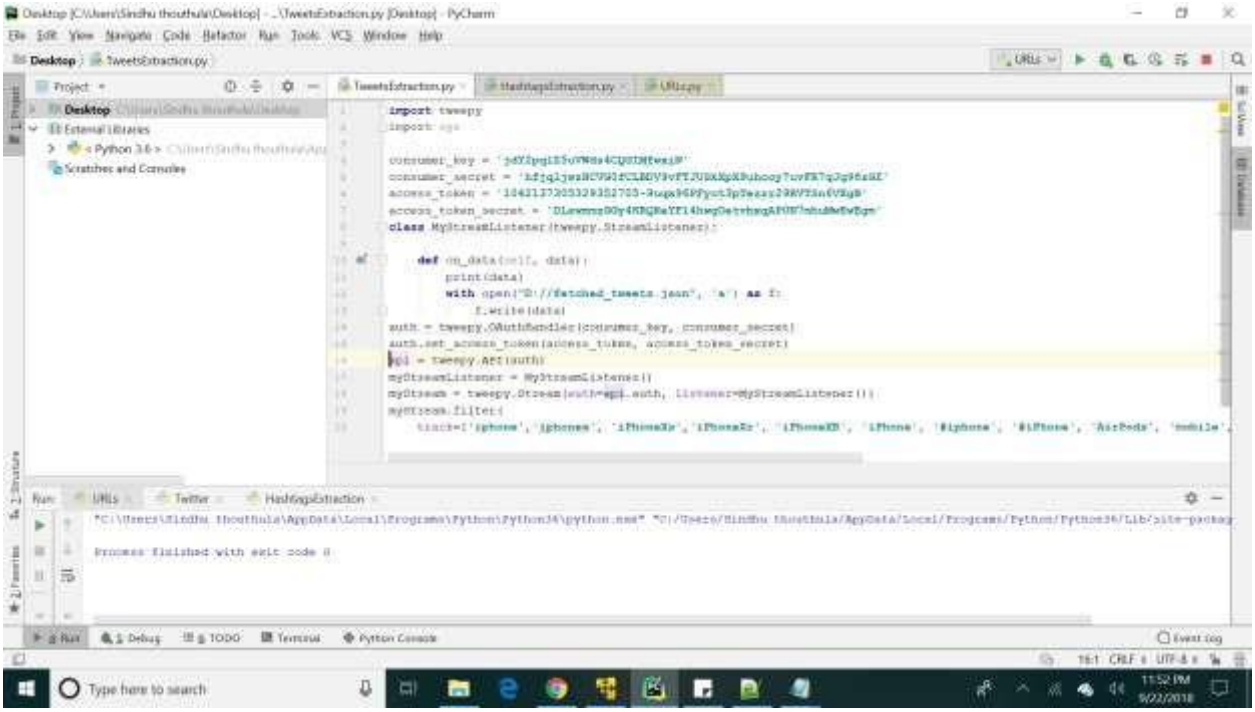
1) TWEETS COLLECTION :

- Initially a Twitter Developer Account is created using this url. https://developer.twitter.com
- The credentials to access the Twitter API are generated in the form of API ACCESS_TOKEN, ACCESS_SECRET, CONSUMER_KEY, CONSUMER_SECRET and are as follows :

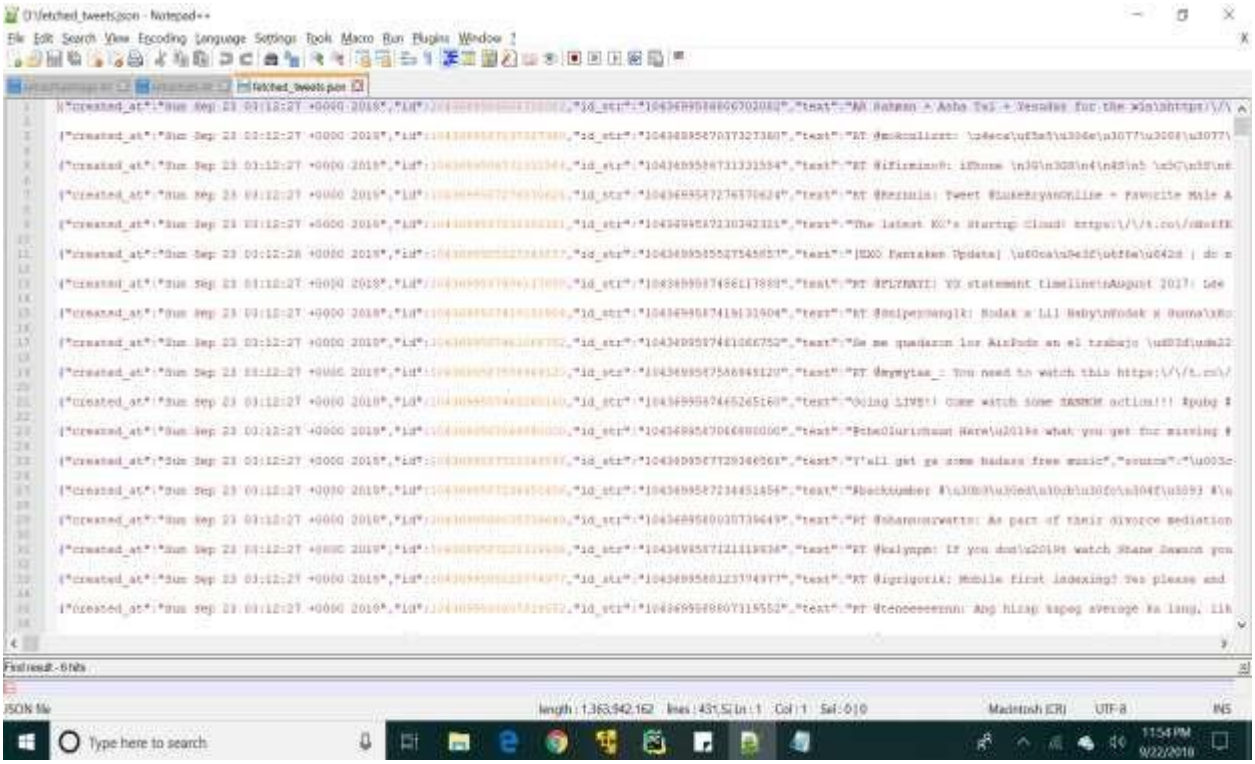
Tokens and Keys Generated :

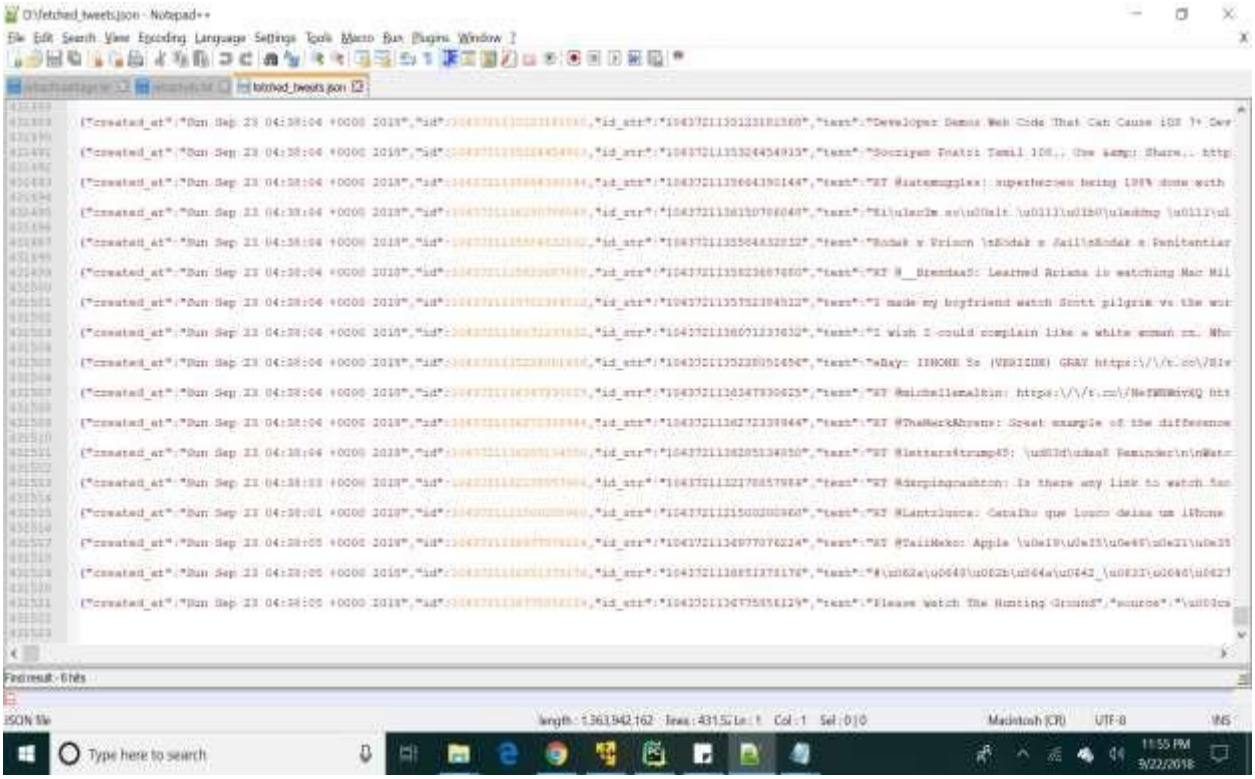
```
Consumer API keys:-

1) jdY2pqlZ5uVWds4CQGIMfwxiN (API key)

2) hfjqljwzHCVG0fCLBDV9vFTJUSkXpX9uhcoy7uvFR7qJq96zGZ (API secret key)

Access token & access token secret:-

1) 1042137305329352705-9uqx96PFyct3pTezrr29RVTSn6VXgB (Access token)

2) DLswmmzGGy4KRQReYF14hwgOetvbxqAP0N7mhuMwEwEgm (Access token secret)
Read and write (Access level)
```

Python Code for Tweets Collection :



Collected Tweets Output :



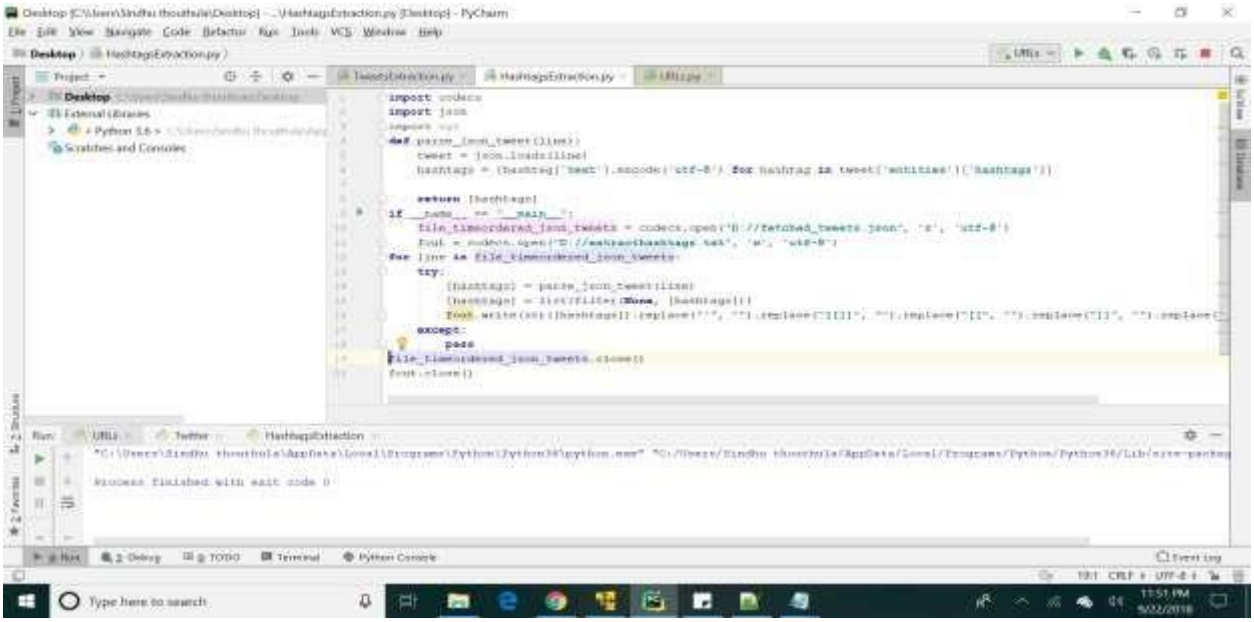


2) HASHTAGS AND URLS EXTRACTION :

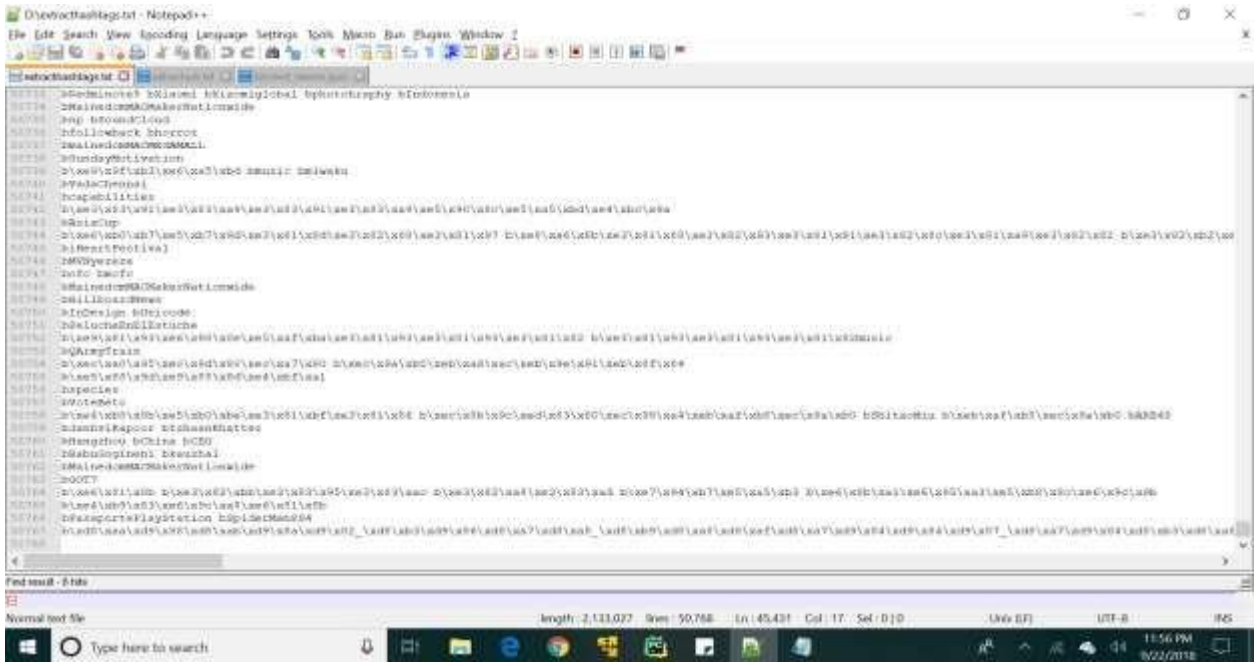
To extract the hashtags and URLs from the collected Tweets, we have again run two Python programs through which we generated the files containing the hashtags and URls alone.

RELATED SCREENSHOTS :

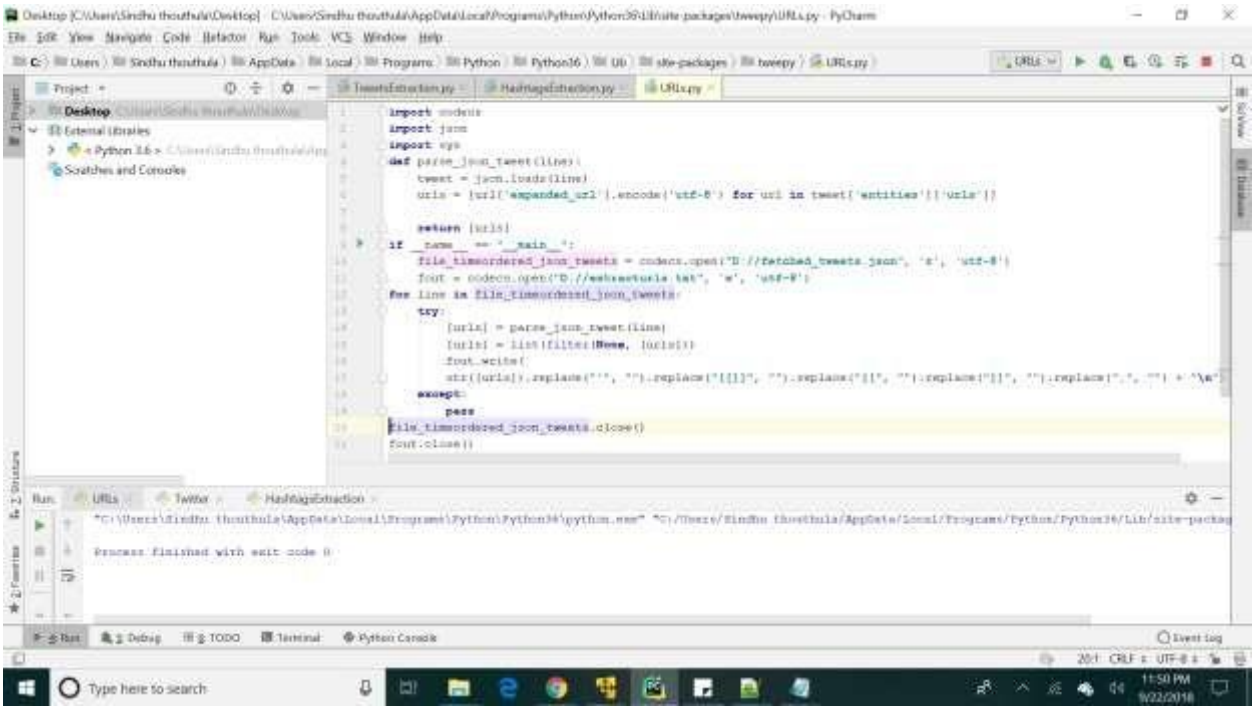
Hashtags Code :



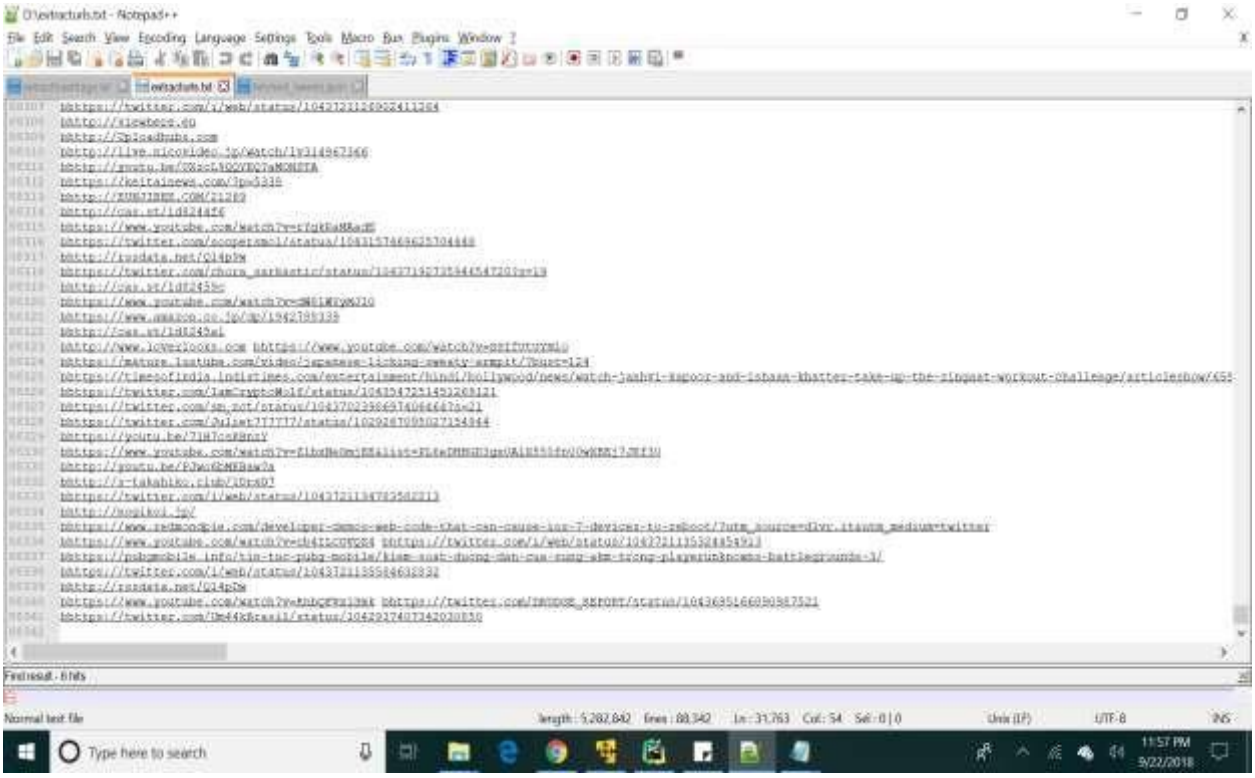
Hashtags Output :



URLs Code :

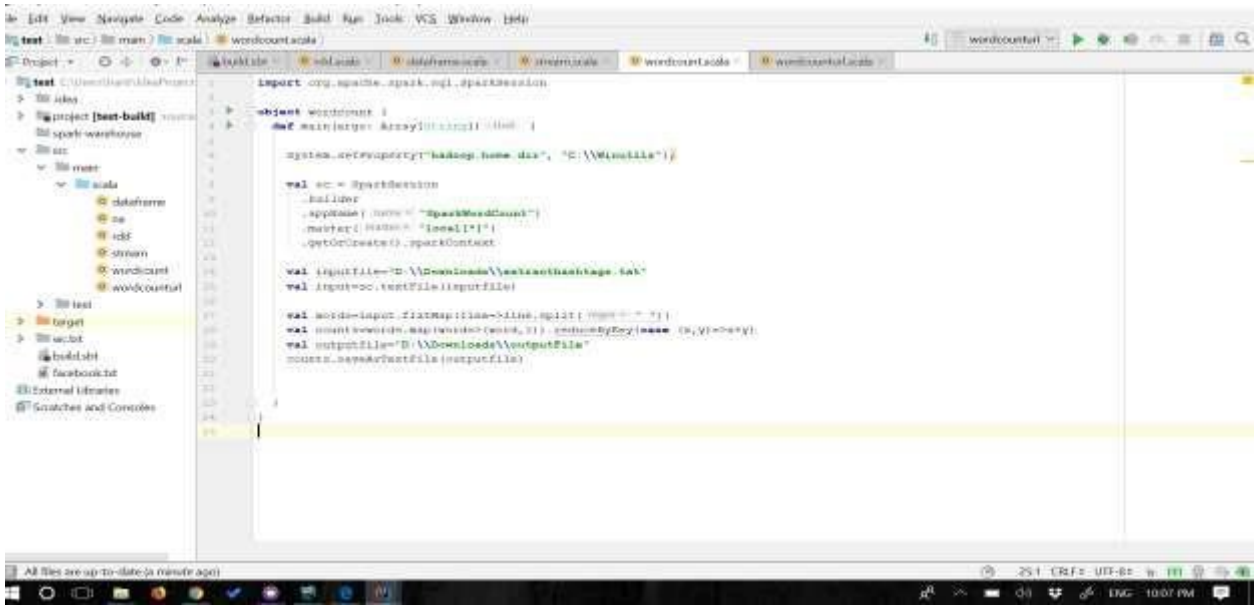

import codecs
import json
import sys
def parse_json_tweet(line):
tweet = json.loads(line)
urls = [url['expanded_url'].encode('utf-8') for url in tweet['entities']['urls']]
return [urls]
if __name__ == '__main__':
file_timeordered_json_tweets = codecs.open('D://fetched_tweets.json', 'r', 'utf-8')
fout = codecs.open('D://extracturls.txt', 'w', 'utf-8')
for line in file_timeordered_json_tweets:
try:
[urls] = parse_json_tweet(line)
[urls] = list(filter(None, [urls]))
fout.write(
str([urls]).replace("'", "").replace("[[]", "").replace("[]", "").replace("]]", "").replace(",", "") + '\n')
except:
pass
file_timeordered_json_tweets.close()
fout.close()
Process finished with exit code 0

URLs Output :

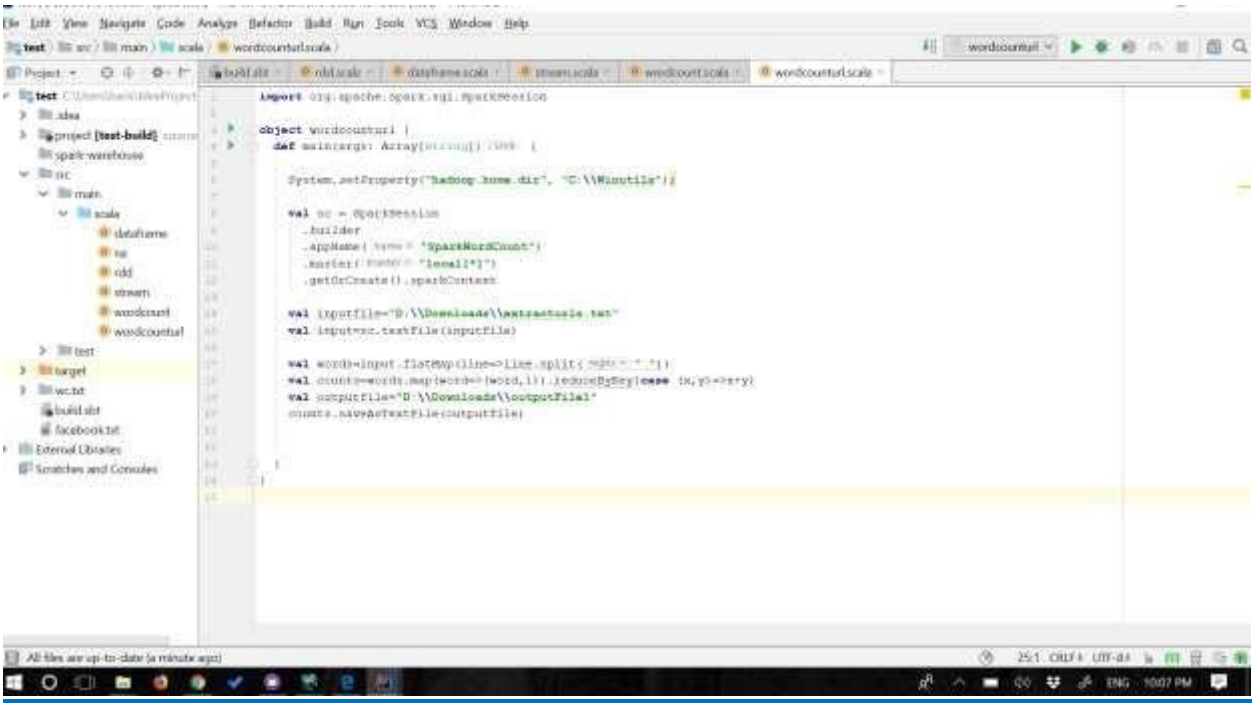

D:\extracturls.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Normal text file

3) RUNNING THE WORDCOUNT IN APACHE HADOOP AND APACHE SPARK :

Hashtags Extraction Code – Spark :



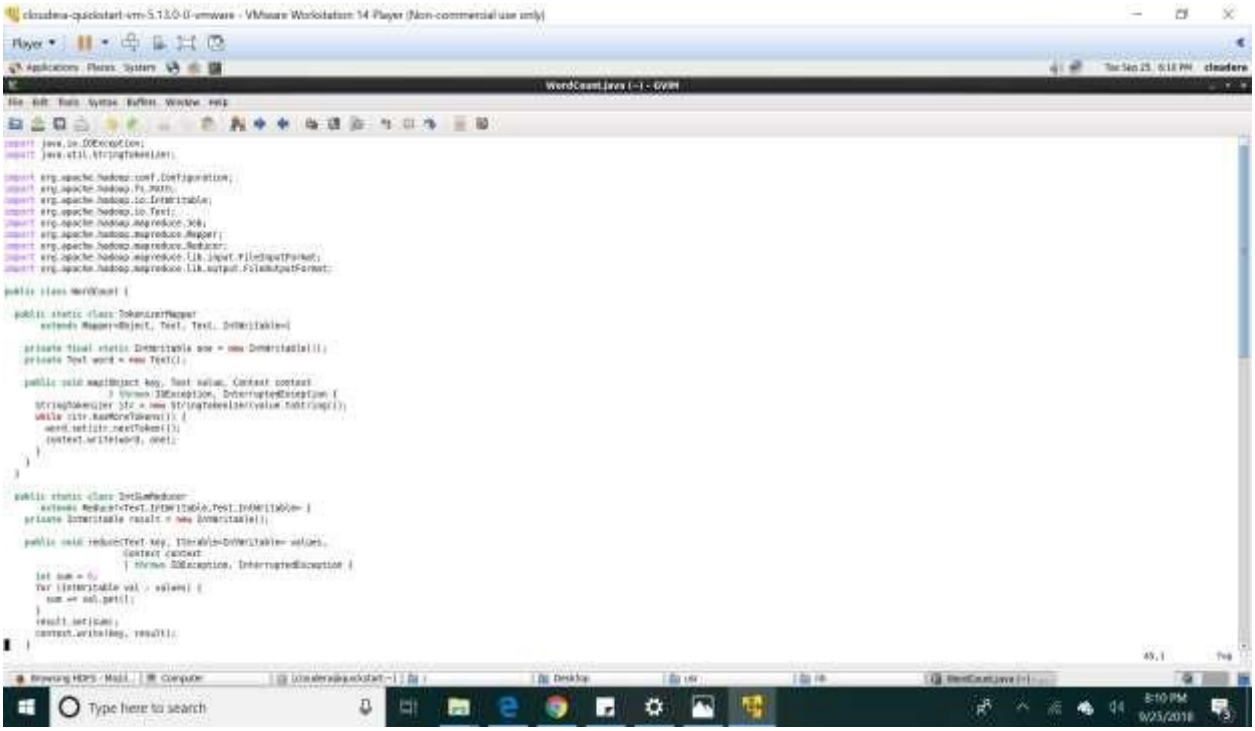
URLs Extraction Code – Spark :

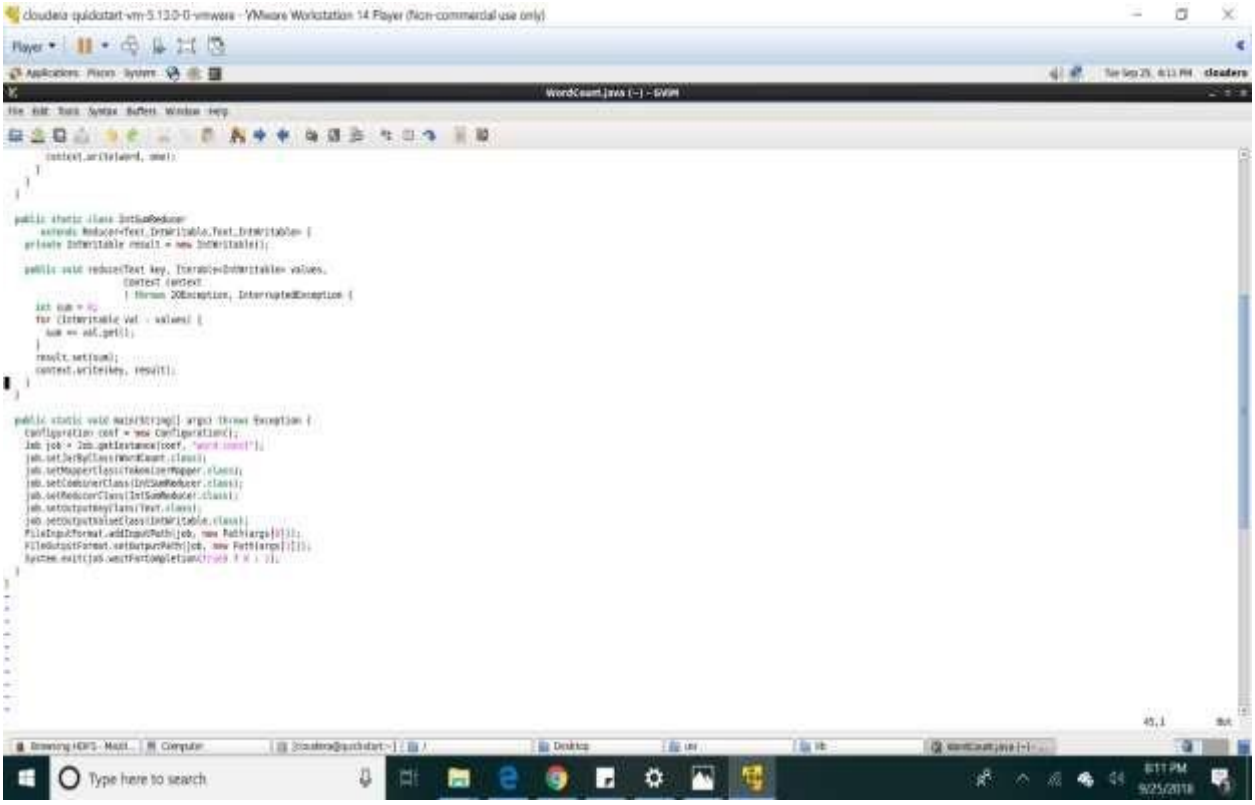


WordCount in Hadoop :

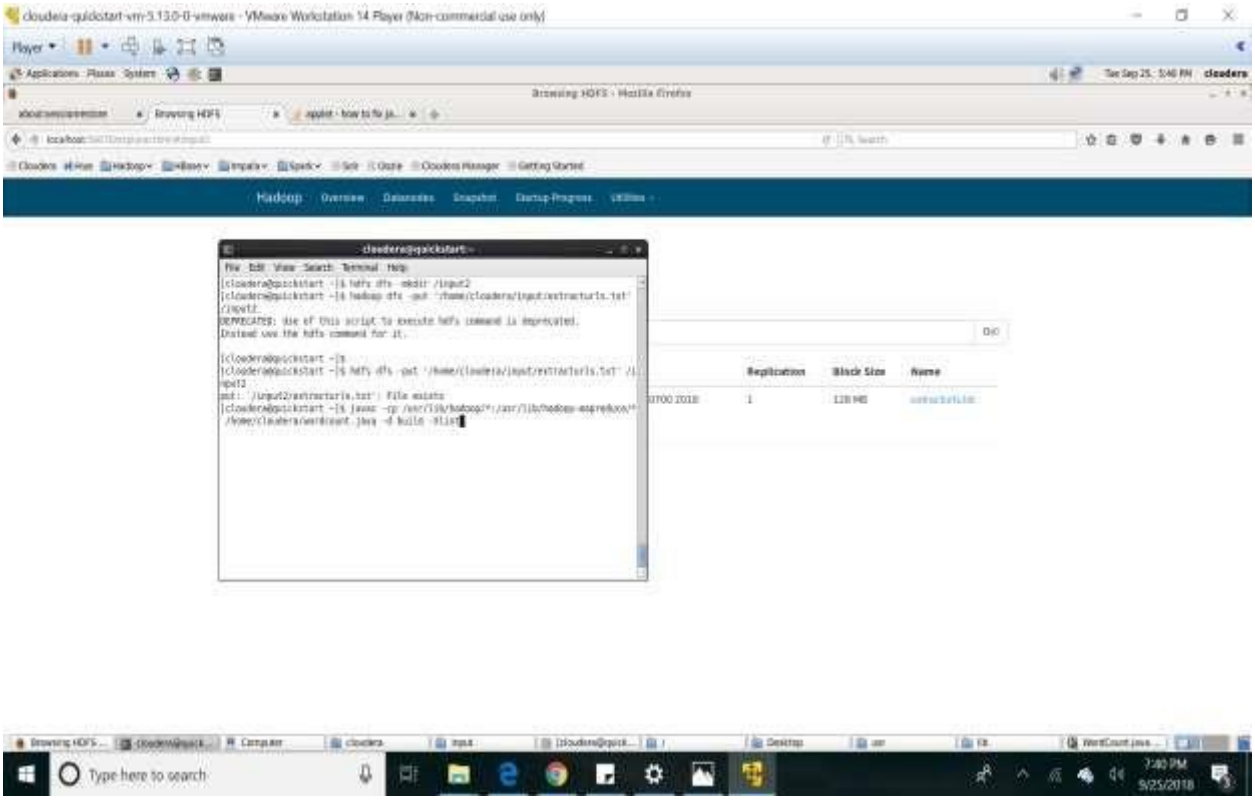
```
$ cat > /home/cloudera/extracturls.txt
$ hdfs dfs -mkdir /input2
$ hdfs dfs -put '/home/cloudera/input/extracturls.txt' /input2
$ hdfs dfs -cat /input2/extracturls.txt
$ hadoop jar wordcount.jar WordCount /input2 /wordcountoutput2
$ hdfs dfs -get /wordcountoutput2 /home/cloudera/output2
$ hdfs dfs -cat/output2/part-r-00000
```

WordCount Java Program – Hadoop :

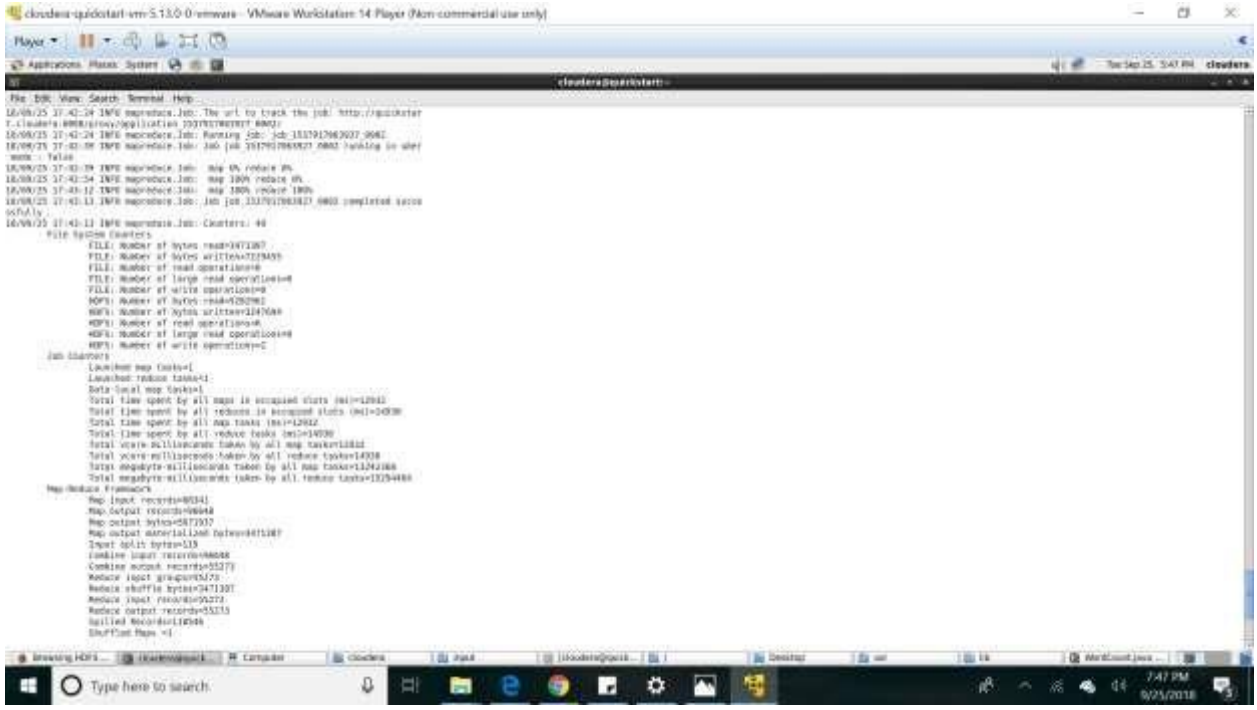




HDFS Commands to run the WordCount in Hadoop :



MapReduce Successful Output : map 100% reduce 100%



TWITTER ANALYSIS & VISUALIZATION OF PHONES/E-ACCESSORIES

ANALYTICAL QUERIES:

The following are the 10 queries on which we performed the visualizations.

```
build.sbt    queries.scala
  "WHEN text like '%technology%' THEN 'TECHNOLOGY'" +
  "WHEN text like '%Accessories%' THEN 'ACCESSORIES'" +
  "END AS phoneType from tweets where text is not null")
disCat.createOrReplaceTempView("disCat2")
val disCat3 = sqlContext.sql("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 '%Iphone10%' THEN 'Iphone X'" +
  "WHEN text like '%iphone7' OR text like '%iphone7plus%' OR text like '%iPHONE7' OR text like '%iPHONE 7' OR text like '%iphone 7' OR text like '%iPHONE7p
  "WHEN text like '%iphone8' OR text like '%iPHONE 8' OR text like '%iphone 8' OR text like '%iphone8plus%' OR text like '%iPHONE8' OR text like '%iPHONE8p
  "WHEN text like '%AirPods%' OR text like '%airpods%' THEN 'AirPods'" +
  "WHEN text like '%watch%' OR text like '%Watch%'  OR text like '%technology%'  OR text like '%Technology%' THEN 'TECHNOLOGY'" +
  "WHEN text like '%ios%' OR text like '%IOS%'  OR text like '%iOS%' THEN 'iOS'" +
  "WHEN text like '%accessories%' OR text like '%ACCESSORIES%' THEN 'Accessories'" +
  "WHEN text like '%Mac%' OR text like '%mac%' OR text like '%MAC%' THEN 'MAC'" +
  "WHEN text like '%mobile%' OR text like '%MOBILE%' THEN 'Mobile'" +
  "END AS phoneType from tweets where text is not null")
disCat3.createOrReplaceTempView("disCat4")
println("Enter any one of the following query to get data")
println("1.Query-1:This query fetches the phone/e-accessories and their popularity based on tweets data")
println("2.Query-2:Which user tweeted most about which type of phone/e-accessories")
println("3.Query-3:Tweets from different countries about phone/e-accessories")
println("4.Query-4:On which day more tweets are done")
println("5.Query-5:This query fetches tweets count for different types of phone/e-accessories")
println("6.Query-6:Language mostly used for tweeting about phone/e-accessories")
println("7.Query-7:Number of tweets for particular date ")
println("8.Query-8:Tweets from verified accounts")
println("9.Query-9:On Which hours More Tweets Were Done")
println("10.Query-10:Which state is mostly having tweets about type of phone/e-accessories")
println("Enter any one of the following query to get data:")
val count = scala.io.StdIn.readLine()
count match {
```

```
19/12/02 17:21:10 INFO FileScanRDD: Reading File path: file:///C:/Users/Sindhu%20thouthula/Desktop/PB%20Project/Phase%20-1/Source/fetched_tweets.json, range: 0
19/12/02 17:21:10 INFO Executor: Running task 5.0 in stage 0.0 (TID 5)
19/12/02 17:21:10 INFO FileScanRDD: Reading File path: file:///C:/Users/Sindhu%20thouthula/Desktop/PB%20Project/Phase%20-1/Source/fetched_tweets.json, range: 0
19/12/02 17:21:10 INFO Executor: Finished task 5.0 in stage 0.0 (TID 5). 34391 bytes result sent to driver
19/12/02 17:21:10 INFO TaskSetManager: Finished task 5.0 in stage 0.0 (TID 5) in 2904 ms on localhost (executor driver) (0/11)
19/12/02 17:21:11 INFO Executor: Finished task 10.0 in stage 0.0 (TID 10). 33931 bytes result sent to driver
19/12/02 17:21:11 INFO TaskSetManager: Finished task 10.0 in stage 0.0 (TID 10) in 698 ms on localhost (executor driver) (9/11)
19/12/02 17:21:12 INFO Executor: Finished task 9.0 in stage 0.0 (TID 9). 34519 bytes result sent to driver
19/12/02 17:21:12 INFO TaskSetManager: Finished task 9.0 in stage 0.0 (TID 9) in 2101 ms on localhost (executor driver) (10/11)
19/12/02 17:21:12 INFO Executor: Finished task 8.0 in stage 0.0 (TID 8). 34466 bytes result sent to driver
19/12/02 17:21:12 INFO TaskSetManager: Finished task 8.0 in stage 0.0 (TID 8) in 2152 ms on localhost (executor driver) (11/11)
19/12/02 17:21:12 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
19/12/02 17:21:12 INFO DAGScheduler: ResultStage 0 (json at queries.scala:16) finished in 9.774 s
19/12/02 17:21:12 INFO DAGScheduler: Job 0 finished: json at queries.scala:16, took 9.850900 s
19/12/02 17:21:13 WARN Utils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.debug.maxFi
Enter any one of the following query to get data
1.Query-1:This query fetches the phone/e-accessories and their popularity based on tweets data
2.Query-2:Which user tweeted most about which type of phone/e-accessories
3.Query-3:Tweets from different countries about phone/e-accessories
4.Query-4:On which day more tweets are done
5.Query-5:This query fetches tweets count for different types of phone/e-accessories
6.Query-6:Language mostly used for tweeting about phone/e-accessories
7.Query-7:Number of tweets for particular date
8.Query-8:Tweets from verified accounts
9.Query-9:On Which hours More Tweets Were Done
10.Query-10:Which state is mostly having tweets about type of phone/e-accessories
Enter any one of the following query to get data:
Run   TODO   sbt shell   Terminal   Build   Event Log
on completed successfully with 1 warning in 7 s 887 ms (moments ago)   46:54 CRLF ÷ UTF-8 ÷ 2 spaces ÷
Type here to search   5:21 PM 12/2/2019
```

1.Query to fetch the tweets of phones/e-accessories based on the popularity.

This query is written to analyze the tweets that are made by the users on the particular key-words which would reflect the popularity of the phones/e-accessories. It would result the count i.e., how many times the phone/e-accessories would appear in the tweets made.

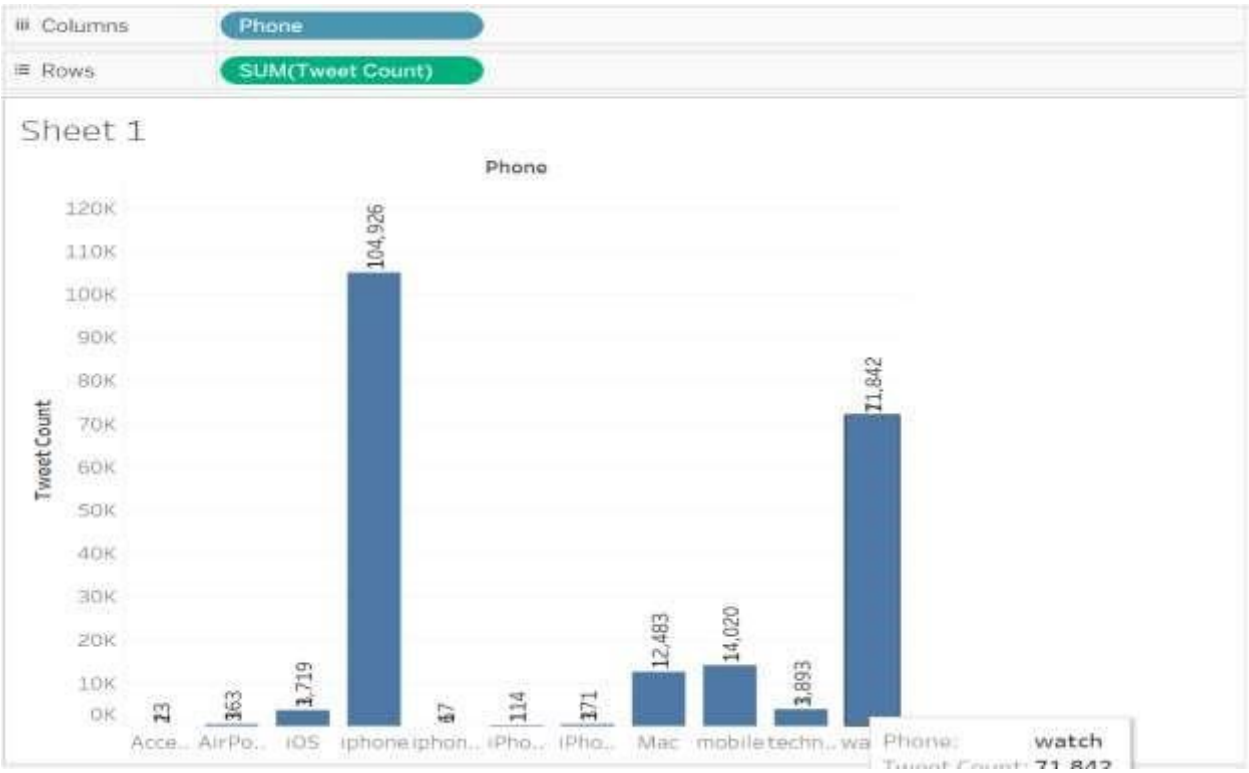
Query-Code :

```
/*-------------------Query 1: This query fetches the phones and its popularity based on tweets data-------------------*/
val textFile = sc.textFile( path = "C:\\Users\\Sindhu thouthula\\Desktop\\PB Project\\Phase -1\\Source\\fetched_tweets.json")
val iphone = (textFile.filter(line => line.contains("iphone")).count())
val iphoneX = (textFile.filter(line => line.contains("iphoneX")).count())
val iPhoneXs = (textFile.filter(line => line.contains("iPhoneXs")).count())
val iPhoneXR = (textFile.filter(line => line.contains("iPhoneXR")).count())
val Mac = (textFile.filter(line => line.contains("Mac")).count())
val iOS = (textFile.filter(line => line.contains("iOS")).count())
val AirPods = (textFile.filter(line => line.contains("AirPods")).count())
val mobile = (textFile.filter(line => line.contains("mobile")).count())
val watch = (textFile.filter(line => line.contains("watch")).count())
val technology = (textFile.filter(line => line.contains("technology")).count())
val Accessories = (textFile.filter(line => line.contains("Accessories")).count())
println("*********************************************")
println("Number of tweets on different types of phones")
println("*********************************************")
println("iphone : %s".format(iphone))
println("iphoneX : %s".format(iphoneX))
println("iPhoneXs : %s".format(iPhoneXs))
println("iPhoneXR : %s".format(iPhoneXR))
println("Mac : %s".format(Mac))
println("iOS : %s".format(iOS))
println("AirPods : %s".format(AirPods))
println("mobile : %s".format(mobile))
println("watch : %s".format(watch))
println("technology : %s".format(technology))
println("Accessories : %s".format(Accessories))
```

Executed Query Output :

```
*********************************************
Number of tweets on different types of phones
*********************************************
iphone : 104926
iphoneX : 67
iPhoneXs : 371
iPhoneXR : 114
Mac : 12483
iOS : 3719
AirPods : 363
mobile : 14020
watch : 71842
technology : 3893
Accessories : 23
18/12/02 17:22:30 INFO SparkContext: Invoking stop() from shutdown hook
18/12/02 17:22:30 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-RF7IOCD:4040
18/12/02 17:22:30 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/12/02 17:22:30 INFO MemoryStore: MemoryStore cleared
18/12/02 17:22:30 INFO BlockManager: BlockManager stopped
18/12/02 17:22:30 INFO BlockManagerMaster: BlockManagerMaster stopped
18/12/02 17:22:30 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/12/02 17:22:30 INFO SparkContext: Successfully stopped SparkContext
18/12/02 17:22:30 INFO ShutdownHookManager: Shutdown hook called
18/12/02 17:22:30 INFO ShutdownHookManager: Deleting directory C:\Users\Sindhu thouthula\AppData\Local\Temp\spark-d87d2d92-a575-4b59-97f3-0dbe87d57cd

Process finished with exit code 0
```

Executed Query Output Visualization :



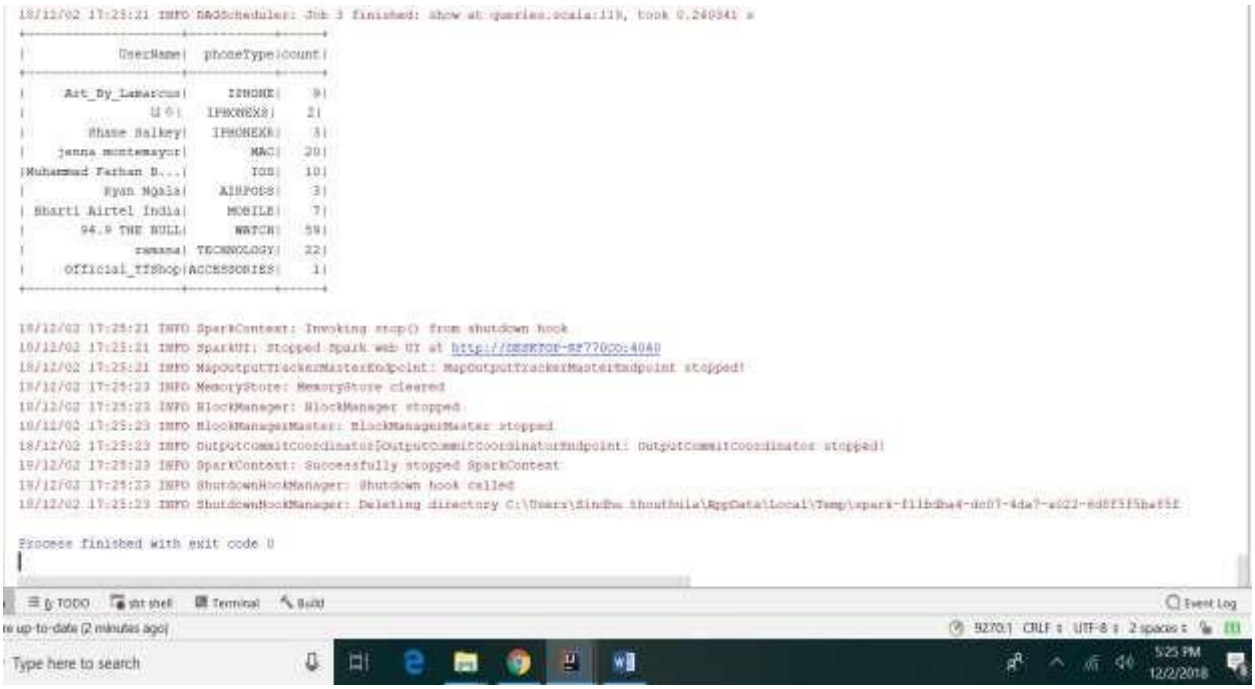
2.Query for finding which user tweeted more about the type of phone/e-accessories.

This query is written to find the user that most tweeted about the phone/e-accessories so that it would result the count of how many times a user tweeted at most for each kind particularly.

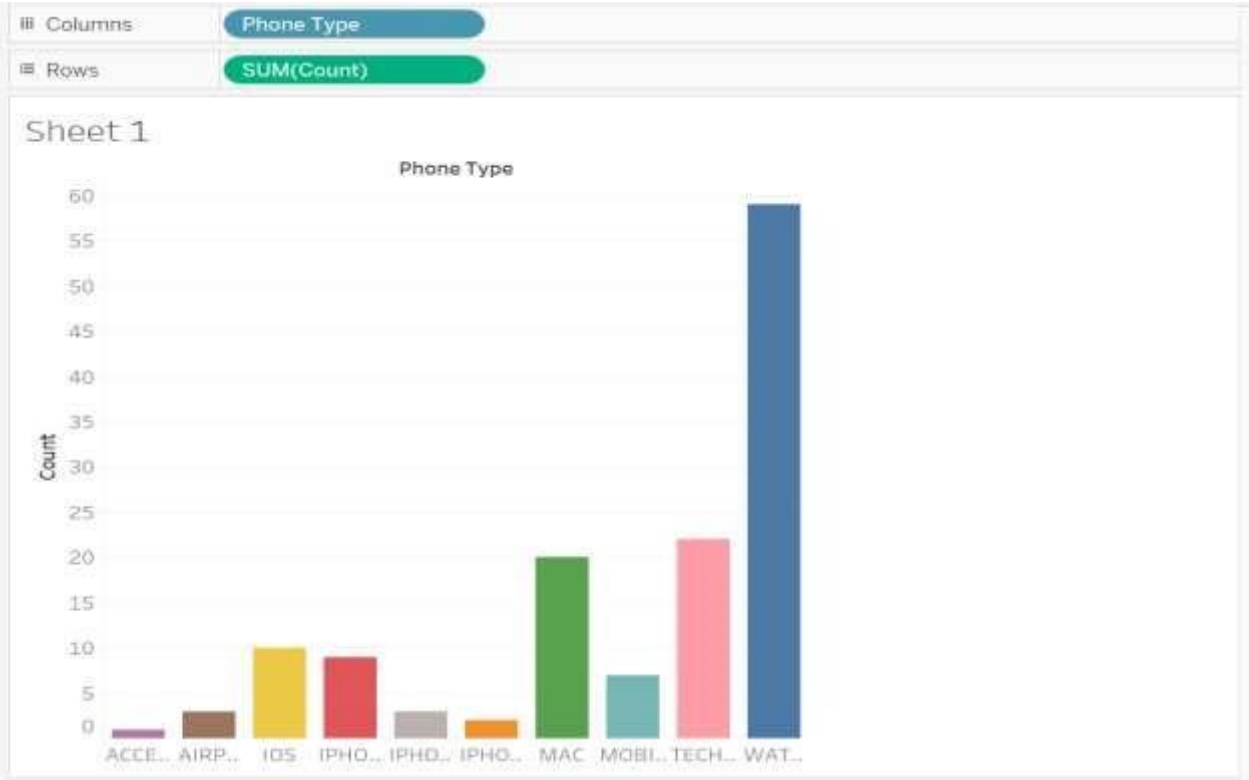
Query-Code :

```
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,'IPHONEX' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='IPHONEX' " +
  "group by UserName order by count desc limit 1 ")
val r3 = sqlContext.sql( sqlText = "SELECT UserName,'IPHONEXS' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='IPHONEXS' " +
  "group by UserName order by count desc limit 1 ")
val r4 = sqlContext.sql( sqlText = "SELECT UserName,'IPHONEXR' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='IPHONEXR' " +
  "group by UserName order by count desc limit 1 ")
val r5 = sqlContext.sql( sqlText = "SELECT UserName,'MAC' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='MAC' " +
  "group by UserName order by count desc limit 1 ")
val r6 = sqlContext.sql( sqlText = "SELECT UserName,'IOS' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='IOS' " +
  "group by UserName order by count desc limit 1 ")
val r7 = sqlContext.sql( sqlText = "SELECT UserName,'AIRPODS' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='AIRPODS' " +
  "group by UserName order by count desc limit 1 ")
val r8 = sqlContext.sql( sqlText = "SELECT UserName,'MOBILE' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='MOBILE' " +
  "group by UserName order by count desc limit 1 ")
val r9 = sqlContext.sql( sqlText = "SELECT UserName,'WATCH' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='WATCH' " +
  "group by UserName order by count desc limit 1")
val r10 = sqlContext.sql( sqlText = "SELECT UserName,'TECHNOLOGY' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='TECHNOLOGY' " +
  "group by UserName order by count desc limit 1")
val r11 = sqlContext.sql( sqlText = "SELECT UserName,'ACCESSORIES' as phoneType,count(*) as count FROM disCat2 WHERE phoneType='ACCESSORIES' " +
  "group by UserName order by count desc limit 1 ")

val r8d1 = 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("****************************************")
r8d1.show()
```

Executed Query Output :



Executed Query Output Visualization :



3.Query for fetching tweets from different countries.

This query is written to find the tweets based on the locations such that it would count how many tweets are posted about the phones/e-accessories from different countries.

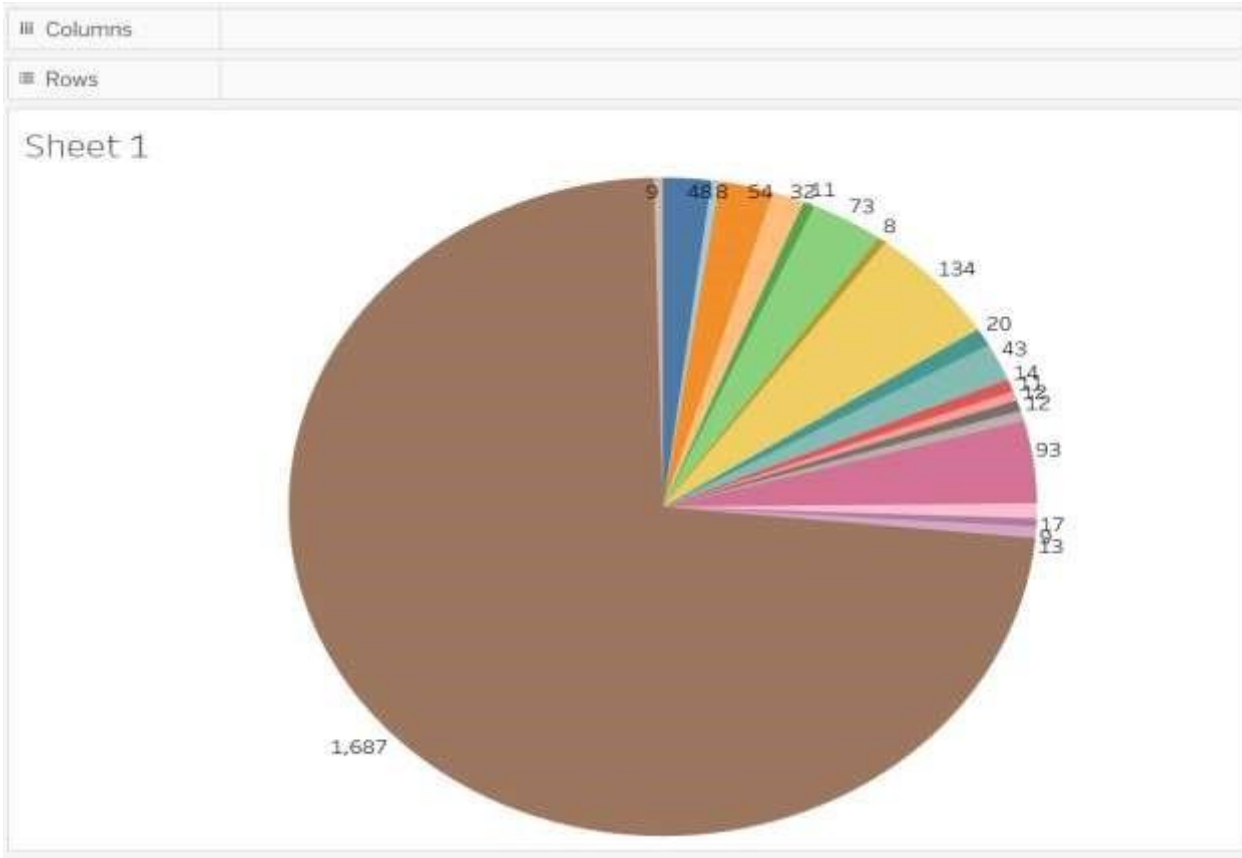
Query-Code :

```
/*-----------------------------------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.country is not null " + "GROUP BY
  countrytweetscount.createOrReplaceTempView( viewName = "countrytweetscount")
  println("****************************************")
  println("Tweets from different countries")
  println("****************************************")
  countrytweetscount.show()
```

Executed Query Output :

```
+--------------------+-----+
|             country|count|
+--------------------+-----+
|       United States| 1607|
|               India|  134|
|Republic of the P...|   93|
|              Canada|   73|
|           Australia|   54|
|                 ||&|   40|
|            Malaysia|   43|
|             Brasil|   32|
|           Indonesia|   20|
|        South Africa|   17|
|             México|   14|
|      United Kingdom|   13|
|            Nigeria|   12|
|   Republic of Korea|   12|
|             Brazil|   11|
|         New Zealand|   11|
|           Sri Lanka|    9|
|            Uruguay|    9|
|           Singapore|    8|
|           Argentina|    8|
+--------------------+-----+
only showing top 20 rows

18/12/02 17:26:55 INFO SparkContext: Invoking stop() from shutdown hook
18/12/02 17:26:55 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-BF7TOCD:4040
18/12/02 17:26:55 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/12/02 17:26:55 INFO MemoryStore: MemoryStore cleared
18/12/02 17:26:55 INFO BlockManager: BlockManager stopped
18/12/02 17:26:55 INFO BlockManagerMaster: BlockManagerMaster stopped
```

Executed Query Output Visualization :



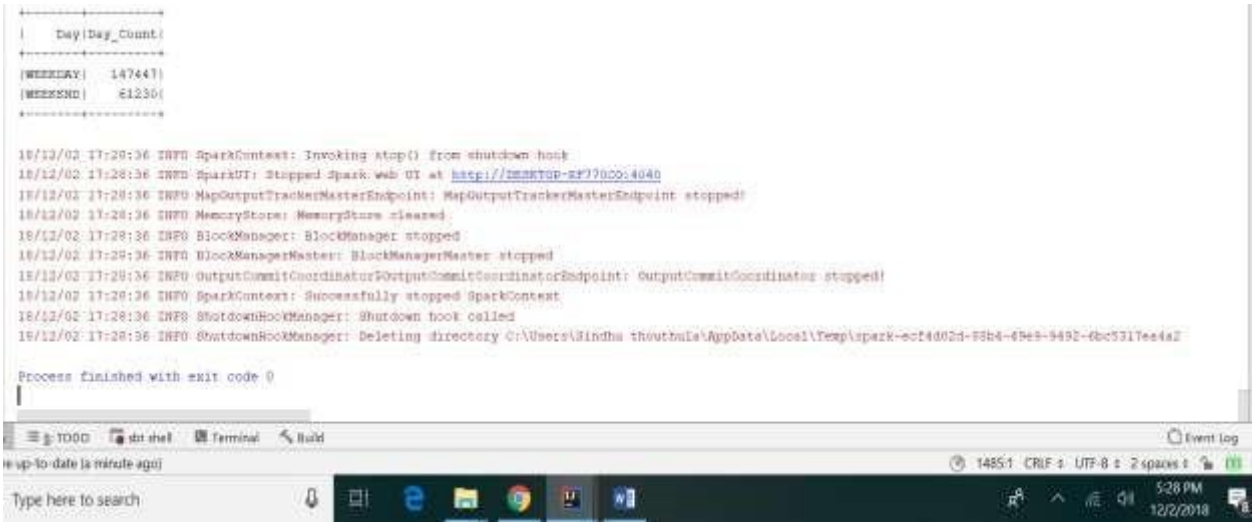
4.Query to fetch tweets to check on which day most of the tweets were made.

This query is written to find out the day on which more tweets were done so that it would count about giving us a figure.

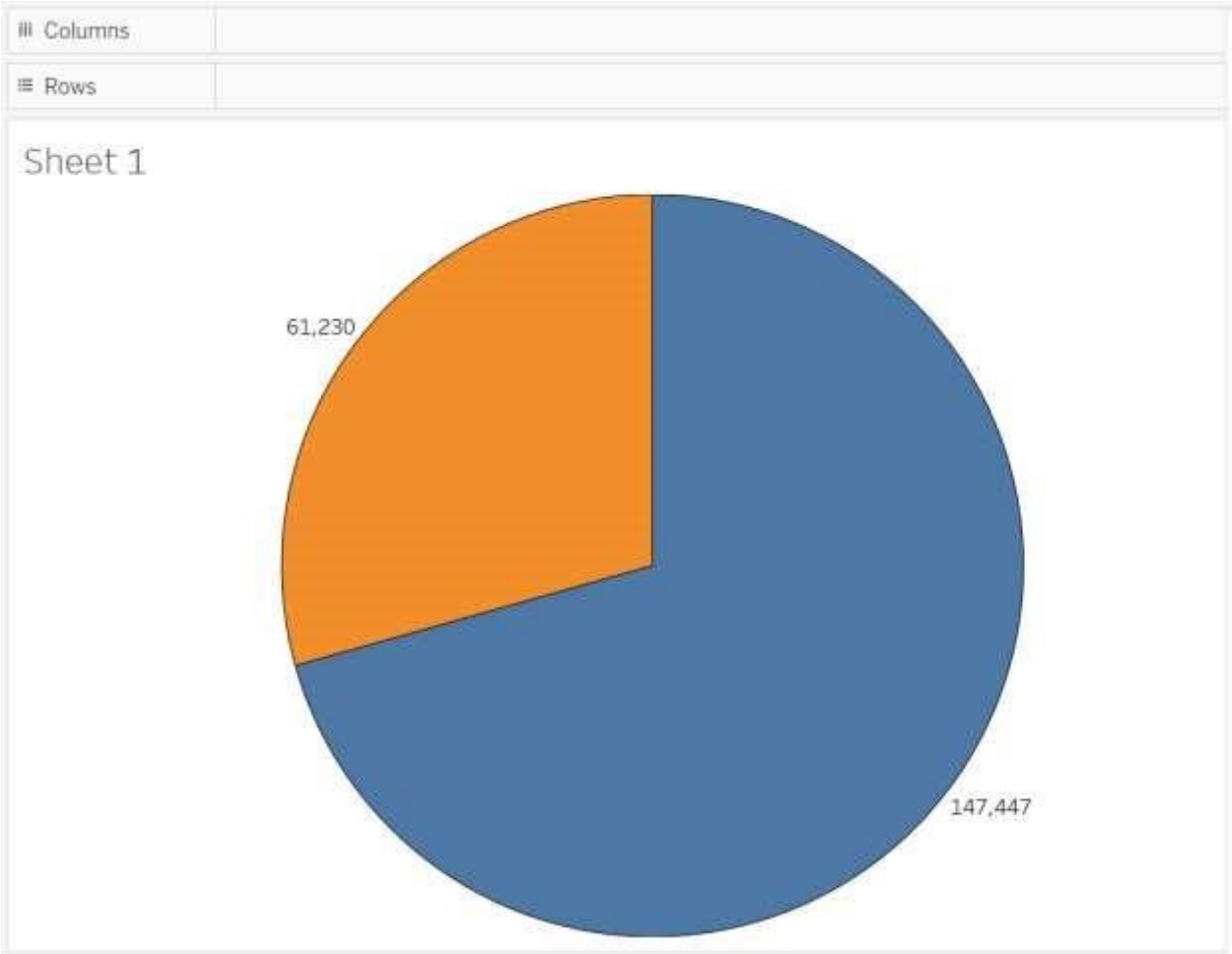
Query-Code :

```
/*------------------------------------------Query # : On which Day More Tweets are posted------------------------------------------*/
case "6" =>
  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 day1 order by count(*) desc")

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

Executed Query Output :



Executed Query Output Visualization :



5.Query to fetch the tweets for the various series of the phone/e-accessories.

This query is written to extract the tweets that made on the different series of the phone/eaccessories so that it would count for it.

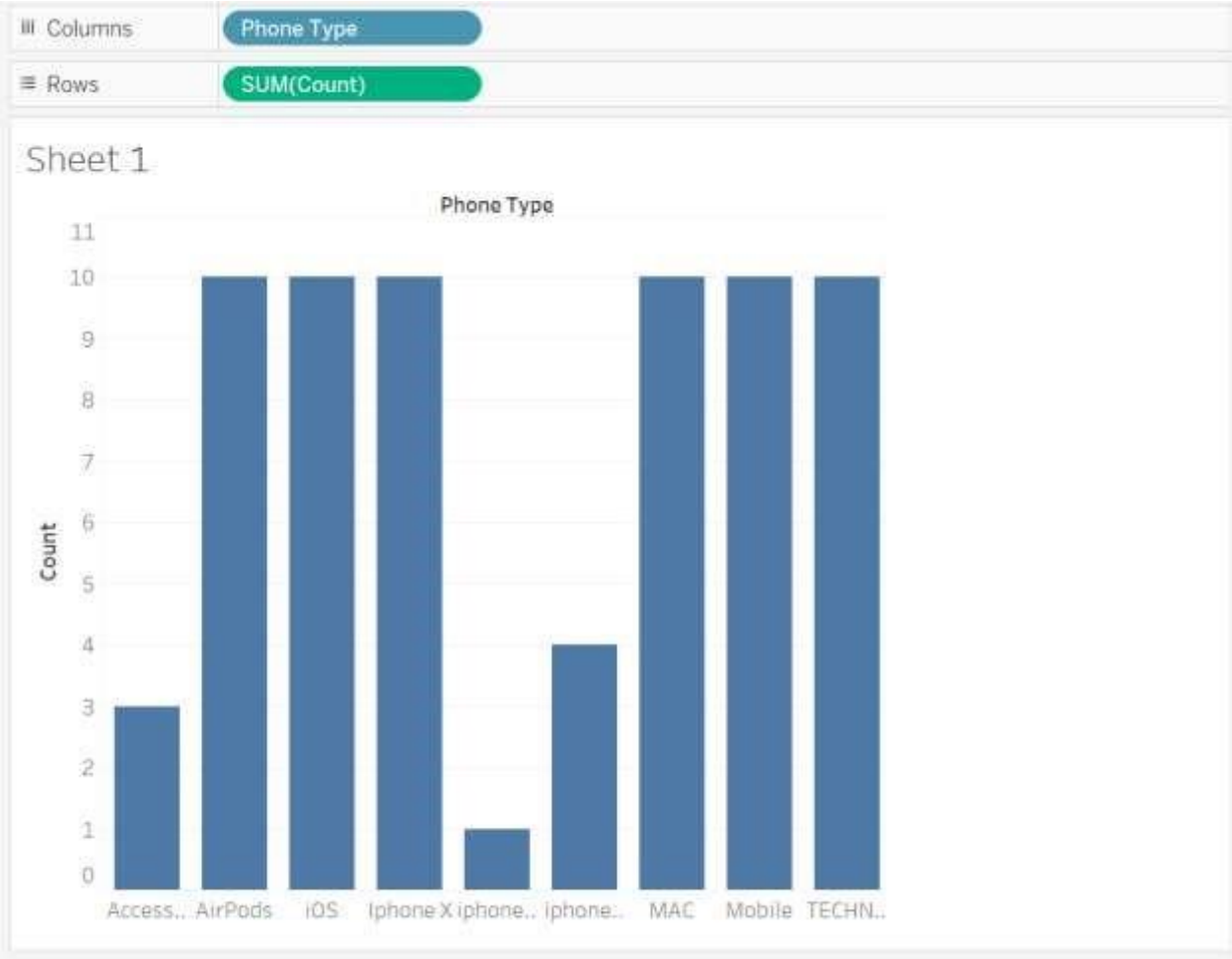
Query-Code :

```
/*-----------------------------------Query 5: Tweets count for different types of phone models-----------------------------------*/
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,'AirPods' as phoneType,count(*) as count FROM disCat4 WHERE phoneType='AirPods' " +
    "group by loc order by count desc limit 10")
  val r5 = sqlContext.sql( sqlText = "SELECT loc,'TECHNOLOGY' as phoneType,count(*) as count FROM disCat4 WHERE phoneType='TECHNOLOGY' " +
    "group by loc order by count desc limit 10")
  val r6 = sqlContext.sql( sqlText = "SELECT loc,'iOS' as phoneType,count(*) as count FROM disCat4 WHERE phoneType='iOS' " +
    "group by loc order by count desc limit 10")
  val r7 = sqlContext.sql( sqlText = "SELECT loc,'Accessories' as phoneType,count(*) as count FROM disCat4 WHERE phoneType='Accessories' " +
    "group by loc order by count desc limit 10")
  val r8 = sqlContext.sql( sqlText = "SELECT loc,'MAC' as phoneType,count(*) as count FROM disCat4 WHERE phoneType='MAC' " +
    "group by loc order by count desc limit 10")
  val r9 = sqlContext.sql( sqlText = "SELECT loc,'Mobile' as phoneType,count(*) as count FROM disCat4 WHERE phoneType='Mobile' " +
    "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()
```

Executed Query Output :

```
+--------------+-----+
|     phonetype|Count|
+--------------+-----+
|           iOS|   10|
|      Iphone X|   10|
|       AirPods|   10|
|   Accessories|    3|
|           MAC|   10|
|    TECHNOLOGY|   10|
|        Mobile|   10|
|iphone8 Series|    4|
|iphone7 Series|    1|
+--------------+-----+

18/12/02 17:31:14 INFO SparkContext: Invoking stop() from shutdown hook
18/12/02 17:31:14 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-RF7DCO:4040
18/12/02 17:31:14 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/12/02 17:31:16 INFO MemoryStore: MemoryStore cleared
18/12/02 17:31:16 INFO BlockManager: BlockManager stopped
18/12/02 17:31:16 INFO BlockManagerMaster: BlockManagerMaster stopped
18/12/02 17:31:16 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/12/02 17:31:16 INFO SparkContext: Successfully stopped SparkContext
18/12/02 17:31:16 INFO ShutdownHookManager: Shutdown hook called
18/12/02 17:31:16 INFO ShutdownHookManager: Deleting directory C:\Users\Sindhu thouthala\AppData\Local\Temp\spark-91623296-27d9-4328-a6f7-e0901875efe

Process finished with exit code 0
```

Executed Query Output Visualization :



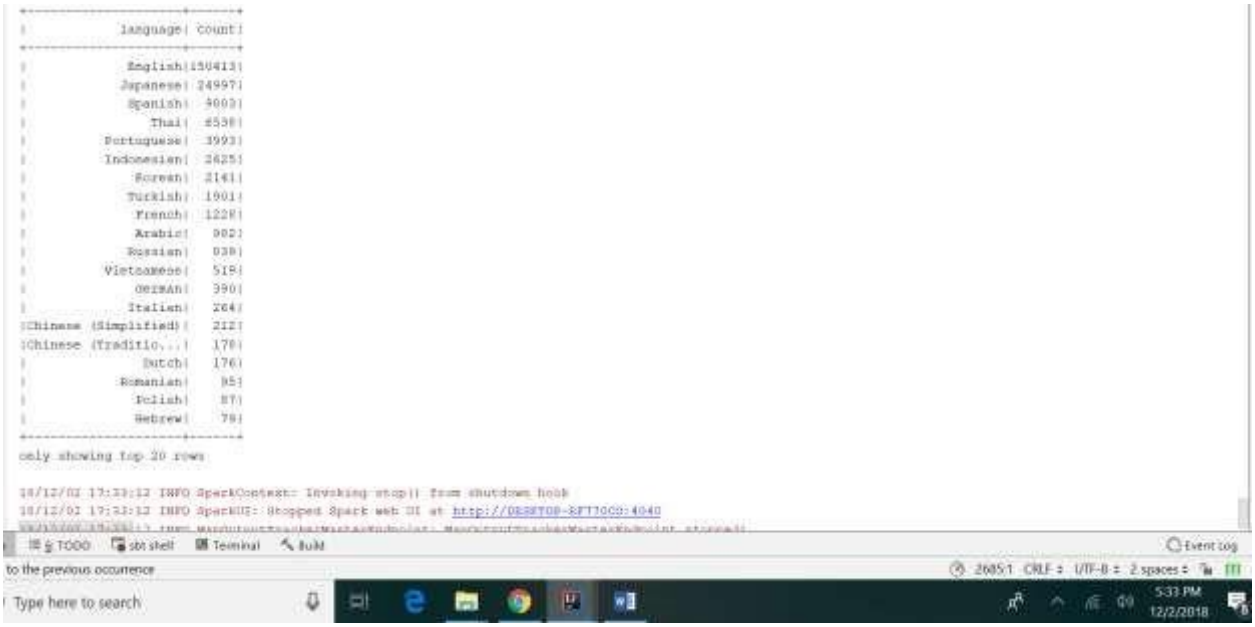
6.Query to fetch the languages mostly used for tweeting about the phone/e-accessories.

This query is written to analyze the language mostly used by the users so that it would count how many times the users tweeted about the phone/e-accessories in a particular language.

Query-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'" +
```

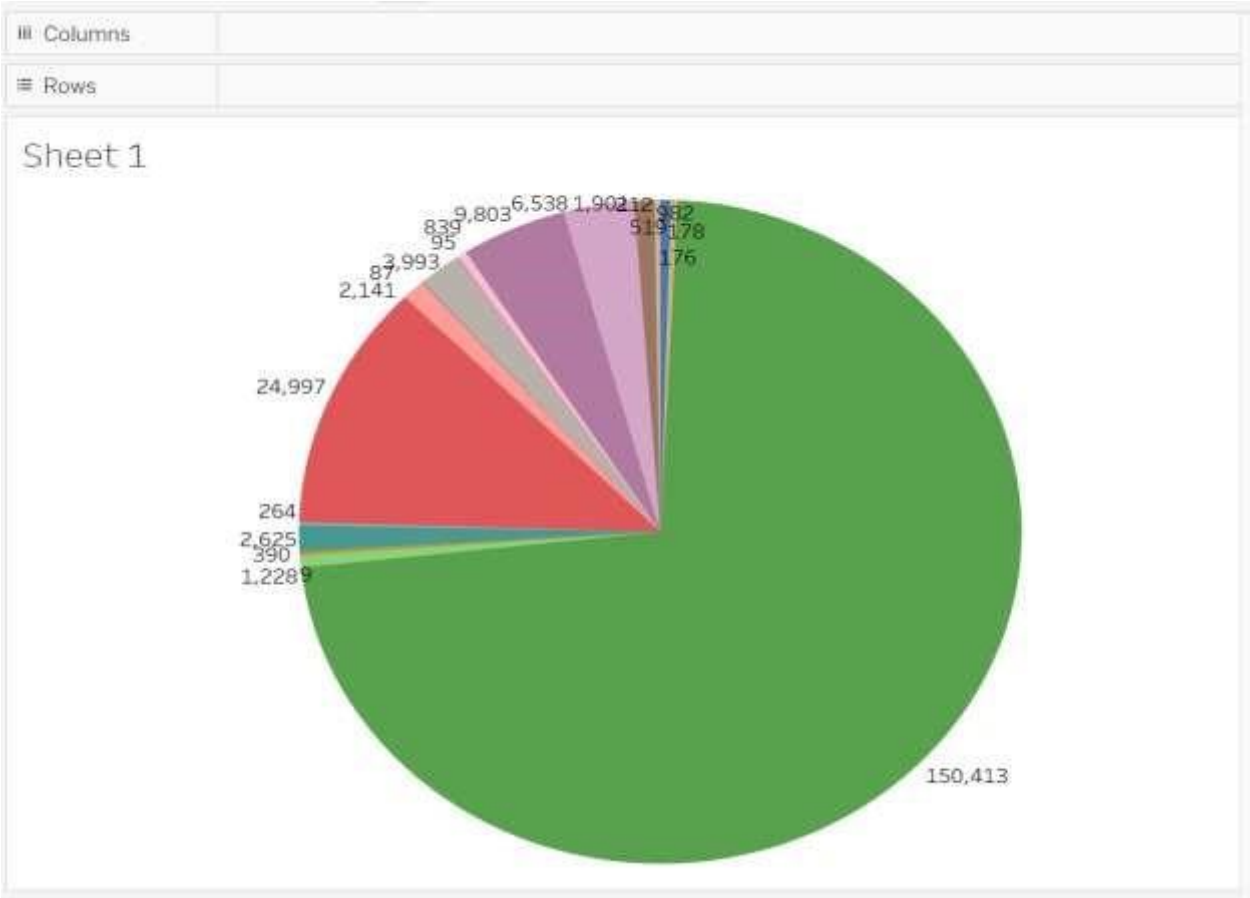
Executed Query Output :



language	count
English	150413
Japanese	24997
Spanish	9803
Thai	6538
Portuguese	3993
Indonesian	2625
Korean	2141
Turkish	1901
French	1228
Arabic	982
Russian	938
Vietnamese	519
German	390
Italian	264
Chinese (Simplified)	212
Chinese (Traditio...)	178
Dutch	176
Romanian	95
Polish	87
Hebrew	78

only showing top 20 rows

Executed Query Output Visualization :



7.Query for fetching the count of the tweets made on a particular day.

This query is written to analyze, depending upon the collected data for each day how many tweets are made.

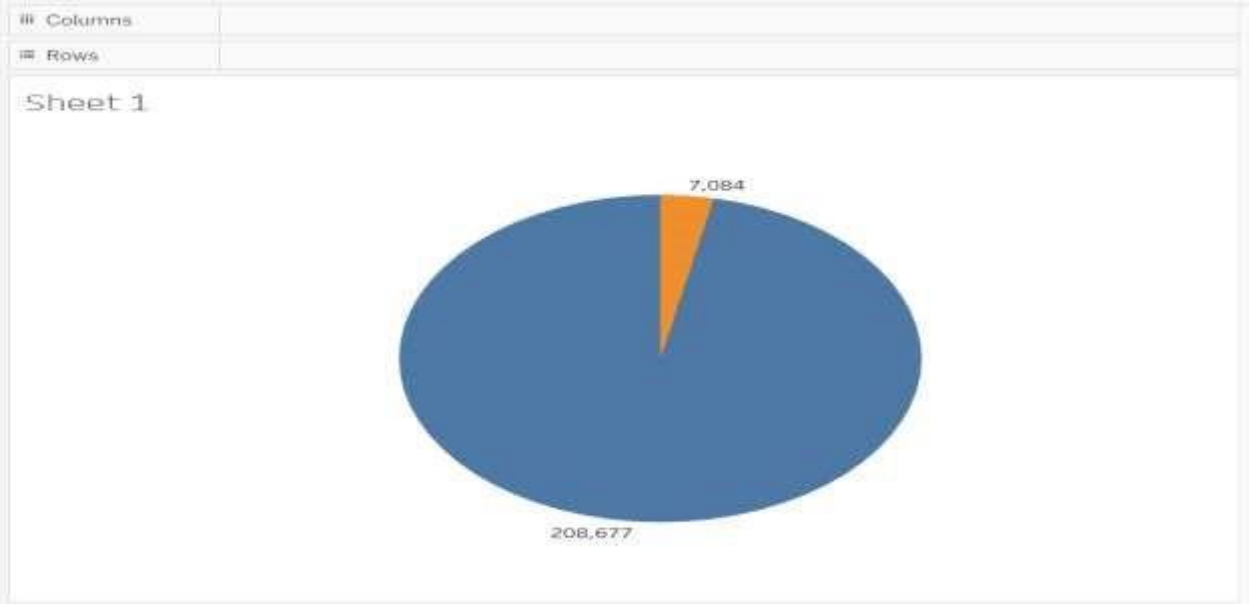
Query-Code :

```
/*-----------------------------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  BY SUBSTR(created_at, 0,
  tweetcount.createOrReplaceTempView( viewName = "tweetcount")
  println("****************************************")
  println("tweet Count")
  println("****************************************")
  tweetcount.show()
```

Executed Query Output :

```
+-----------+------------+
|tweet_date|tweet_count|
+-----------+------------+
|Sun Sep 23|      208677|
|     null|        7084|
+-----------+------------+

19/12/02 17:34:31 INFO SparkContext: Invoking stop() from shutdown hook
19/12/02 17:34:31 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-KF770ou:4040
19/12/02 17:34:31 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
19/12/02 17:34:32 INFO MemoryStore: MemoryStore cleared
19/12/02 17:34:32 INFO BlockManager: BlockManager stopped
19/12/02 17:34:32 INFO BlockManagerMaster: BlockManagerMaster stopped
19/12/02 17:34:32 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
19/12/02 17:34:32 INFO SparkContext: Successfully stopped SparkContext
19/12/02 17:34:32 INFO ShutdownHookManager: Shutdown hook called
19/12/02 17:34:32 INFO ShutdownHookManager: Deleting directory C:\Users\sindhu thouthula\AppData\Local\Temp\spark-545bee33-7265-4046-8706-4e7a13b8f1ie

Process finished with exit code 0
```

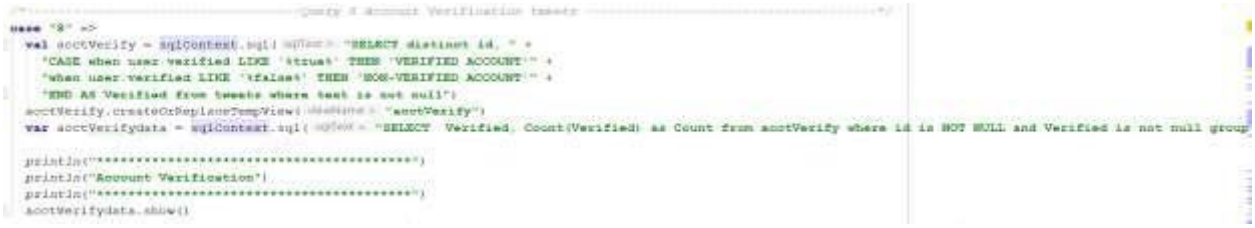
Executed Query Output Visualization :



8.Query for fetching the tweets made from verified accounts.

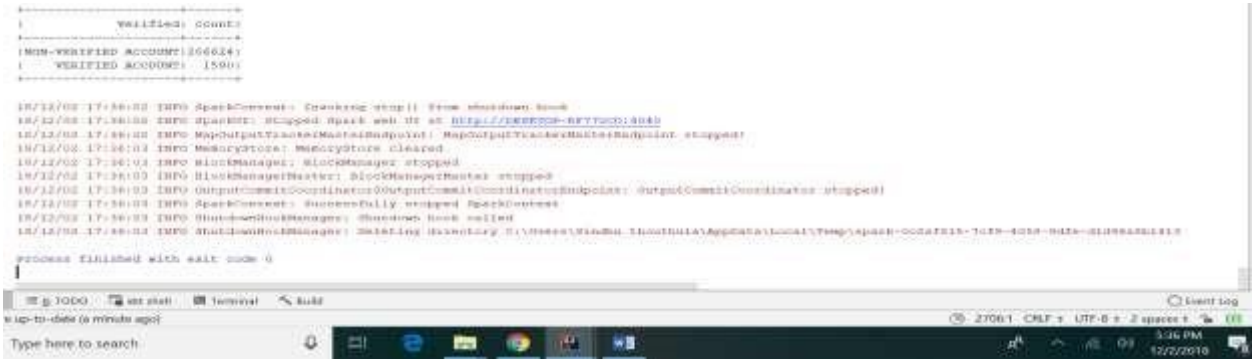
This query is written to make an analysis on the number of verified users. This query counts for the tweets made from the verified accounts only and gives us the value.

Query-Code :



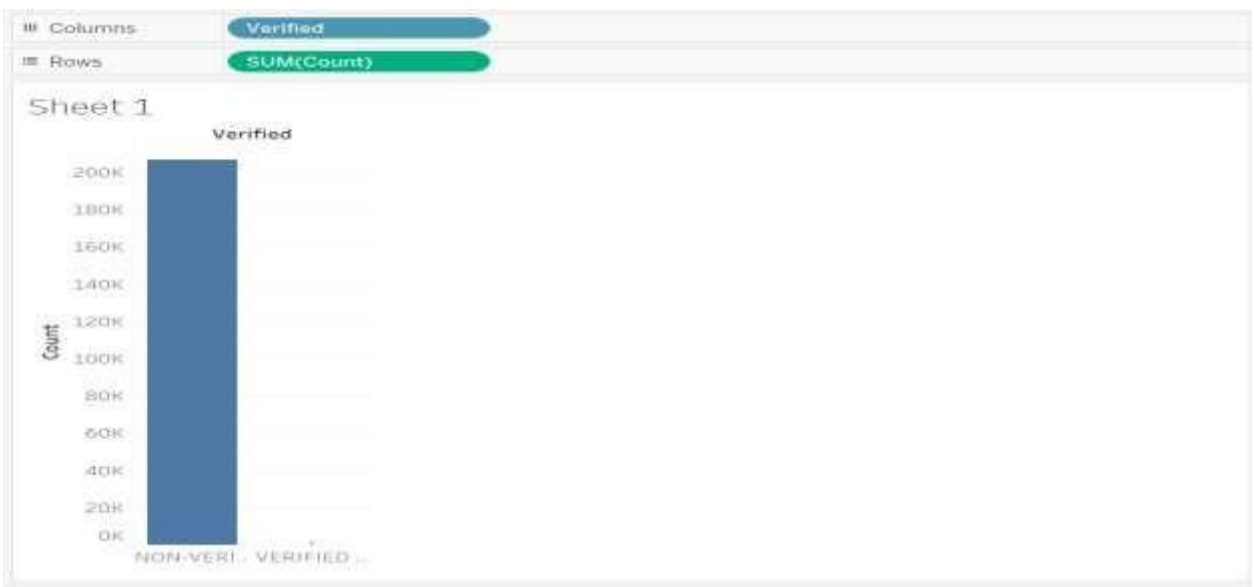
```
/*------------------------------------Query 8 Account Verification tweets------------------------------------*/
case "8" =>
  val acctVerify = sqlContext.sql( sqlText = "SELECT distinct id, " +
    "CASE when user.verified LIKE 'true%' THEN 'VERIFIED ACCOUNT'" +
    "when user.verified LIKE 'false%' 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 and Verified is not null group

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

Executed Query Output :



```
+----------------------+-------+
|            Verified| count|
+----------------------+-------+
|NON-VERIFIED ACCOUNT|266624|
|    VERIFIED ACCOUNT|  1590|
+----------------------+-------+

18/12/02 17:36:02 INFO SparkContext: Invoking stop() from shutdown hook
18/12/02 17:36:02 INFO SparkUI: stopped Spark web UI at http://DESKTOP-RFT7UC0:4040
18/12/02 17:36:02 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/12/02 17:36:03 INFO MemoryStore: MemoryStore cleared
18/12/02 17:36:03 INFO BlockManager: BlockManager stopped
18/12/02 17:36:03 INFO BlockManagerMaster: BlockManagerMaster stopped
18/12/02 17:36:03 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/12/02 17:36:03 INFO SparkContext: Successfully stopped SparkContext
18/12/02 17:36:03 INFO ShutdownHookManager: Shutdown hook called
18/12/02 17:36:03 INFO ShutdownHookManager: Deleting directory C:\Users\Vindhu Thunthula\AppData\Local\Temp\spark-ccc2f815-7c19-4c58-b4fb-d1d9ea8b1b1b

process finished with exit code 0
```

Executed Query Output Visualization :



9.Query for fetching the tweets based on the hours on which most of them were made.

This query is written to analyze on which hours tweets are made like mornings, afternoon, evenings etc.

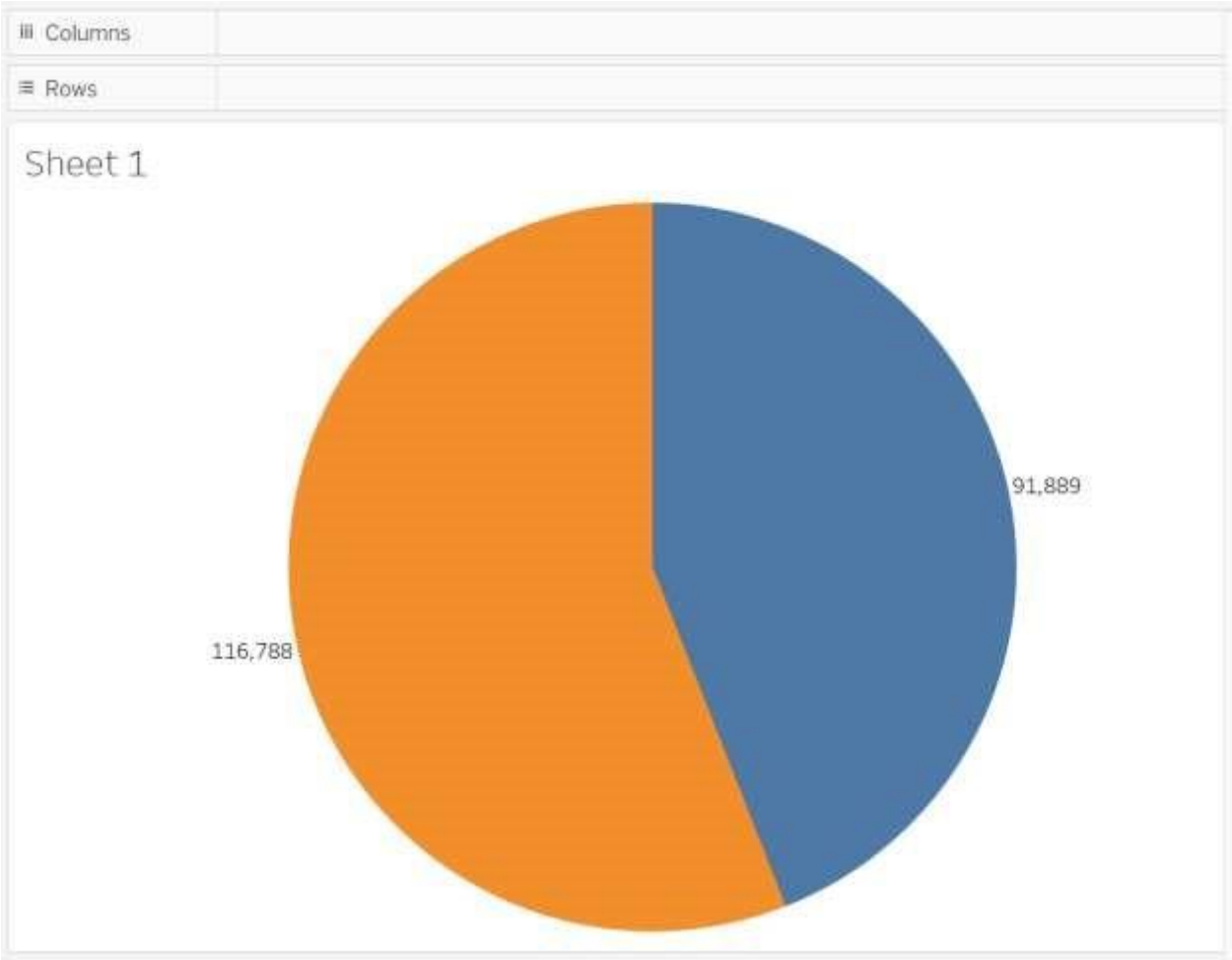
Query-Code :

```
/*--------------------------------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 group by time order by count(*)

  println("**********************************")
  println("On Which hours More Tweets Were Done")
  println("**********************************")
```

Executed Query Output :

```
+-------------+-------------+
|         hour|tweets_count|
+-------------+-------------+
|     midnight|       116788|
|early Morning|        91889|
+-------------+-------------+

18/12/02 17:37:10 INFO BlockManagerInfo: Removed broadcast_4_piece0 on DESKTOP-RF77QCO:55163 in memory (size: 15.5 KB, free: 892.2 MB)
18/12/02 17:37:10 INFO SparkContext: Invoking stop() from shutdown hook
18/12/02 17:37:10 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-RF77QCO:4040
18/12/02 17:37:10 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/12/02 17:37:10 INFO MemoryStore: MemoryStore cleared
18/12/02 17:37:10 INFO BlockManager: BlockManager stopped
18/12/02 17:37:10 INFO BlockManagerMaster: BlockManagerMaster stopped
18/12/02 17:37:10 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/12/02 17:37:10 INFO SparkContext: Successfully stopped SparkContext
18/12/02 17:37:10 INFO ShutdownHookManager: Shutdown hook called
18/12/02 17:37:10 INFO ShutdownHookManager: Deleting directory C:\Users\Sindhu thouthula\AppData\Local\Temp\spark-212bdae2-d417-4683-81e9-67daa806a02e

Process finished with exit code 0
```

Executed Query Output Visualization :



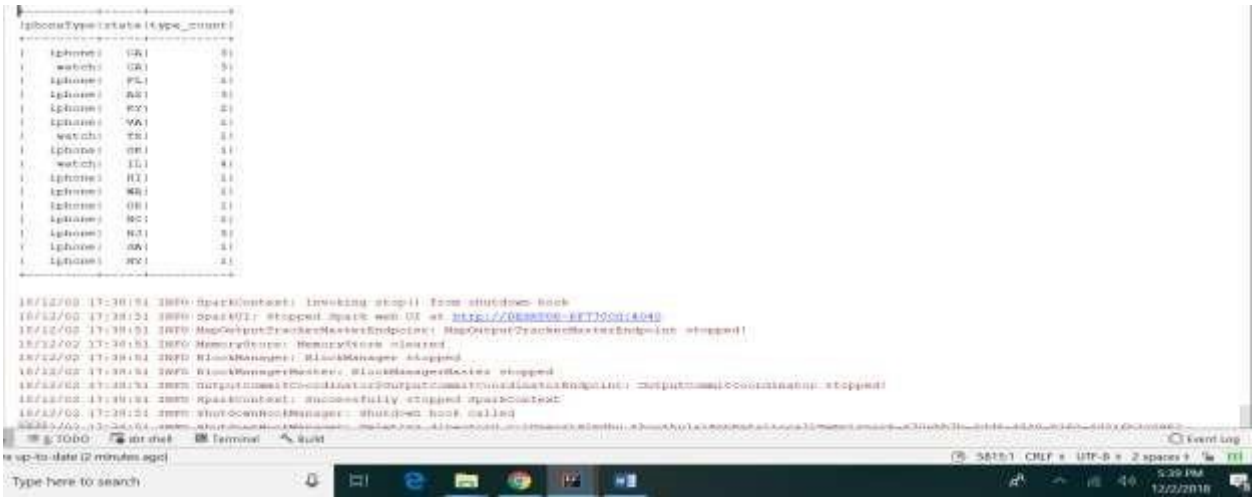
10.Query for fetching tweets based on which state has more tweets about particular type of phone/e-accessories.

This query is written to check which state users made most number of tweets about a type of phone/e-accessories.

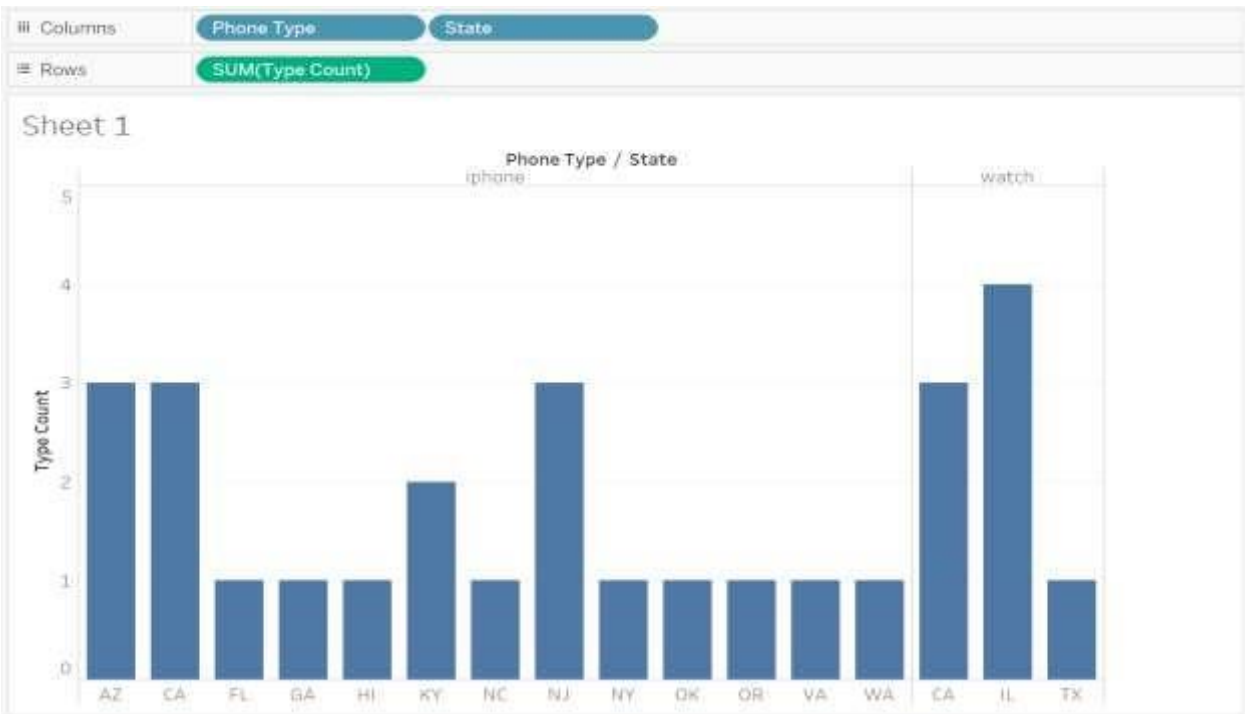
Query-Code :

```
/*------------------------------------Query 10 Which state is mostly having tweets about type of phone------------------------------------*/
case "10" =>
  val iphoneRDD = sqlContext.sql( """ SELECT 'iphone' as phoneType, user.location as loc from tweets where text LIKE '%#iphone%' """)
  val iphoneXRDD = sqlContext.sql( """ SELECT 'iphoneX' as phoneType, user.location as loc from tweets where text LIKE '%#iphoneX%' """)
  val iPhoneXsRDD = sqlContext.sql( """ SELECT 'iPhoneXs' as phoneType, user.location as loc from tweets where text LIKE '%#iPhoneXs%' """)
  val watchRDD = sqlContext.sql( """ SELECT 'watch' as phoneType, user.location as loc from tweets where text LIKE '%#watch%' """)
  //val breakRDD =  sqlContext.sql("""" SELECT 'breakfast' as Mealtype, COUNT(DISTINCT(created_at,12,2)) as hour, user.location as loc from df2 where text
  //val dinnerRDD =  sqlContext.sql("""" SELECT 'dinner' as Mealtype, COUNT(DISTINCT(created_at,12,2)) as hour, user.location as loc from df2 where text LIK
  val sql2RDD = iphoneRDD.union(iphoneXRDD).union(iPhoneXsRDD).union(watchRDD)
  sql2RDD.createOrReplaceTempView("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  '%C
      |OR loc LIKE  '%Florida%'
      |OR loc LIKE  '%Georgia%'
      |OR loc LIKE  '%Hawaii%'
      |OR loc LIKE  '%Idaho%'
      |OR loc LIKE  '%Illinois%'
      |OR loc LIKE  '%Indiana%'
```

Executed Query Output :



Executed Query Output Visualization :



CODE TESTING:

We can test the code by opening the Project folder using IntelliJ IDEA and can run the code and see that the queries are being executed, later we can see the visualizations can be opened in Tableau and we can click on each of them to see the visualizations of ten queries.

Individual task: 3 queries each with respective visualization

Collection of data and Word Count (together)

Work to be Completed: Sentiment Analysis.

PHASE 2:

For the phase 2 we have implemented our project in docker.

DOCKER:

Docker is a popular independent software container platform that allows you to build and ship your applications, along with all its environments, libraries and dependencies in containers

MINGW64:/c/Program Files/Docker Toolbox

```
                        ##         .
                  ## ## ##        ==
               ## ## ## ## ##    ===
           /"""""""""""""""""\___/ ===
      ~~~ {~~ ~~~~ ~~~ ~~~~ ~~~ ~ /  ===- ~~~
           \_____ o           __/
             \    \         __/
              \____\_______/

Docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at https://docs.docker.com


Start interactive shell

Kite@DESKTOP-D128J0A MINGW64 /c/Program Files/Docker Toolbox
$ docker pull chkrish9/msp-pb-phase3:2
2: Pulling from chkrish9/msp-pb-phase3
Digest: sha256:7ac07c85e32407ab52de4f42bdb405aaadb05e8cef94fe1755b989d42e5f599c
Status: Image is up to date for chkrish9/msp-pb-phase3:2

Kite@DESKTOP-D128J0A MINGW64 /c/Program Files/Docker Toolbox
$
```

MINGW64:/c/Program Files/Docker Toolbox

```
                        ##         .
                  ## ## ##        ==
               ## ## ## ## ##    ===
           /"""""""""""""""""\___/ ===
      ~~~ {~~ ~~~~ ~~~ ~~~~ ~~~ ~ /  ===- ~~~
           \_____ o           __/
             \    \         __/
              \____\_______/

docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at https://docs.docker.com


Start interactive shell

Kite@DESKTOP-D128J0A MINGW64 /c/Program Files/Docker Toolbox
$ docker pull chkrish9/msp-pb-phase3:2
2: Pulling from chkrish9/msp-pb-phase3
Digest: sha256:7ac07c85e32407ab52de4f42bdb405aaadb05e8cef94fe1755b989d42e5f599c
Status: Image is up to date for chkrish9/msp-pb-phase3:2

Kite@DESKTOP-D128J0A MINGW64 /c/Program Files/Docker Toolbox
$ docker run -it chkrish9/msp-pb-phase3:2 bash
root@863486393d93:~#
```

MINGW64:/c/Program Files/Docker Toolbox

```
                        ##         .
                  ## ## ##        ==
               ## ## ## ## ##    ===
           /"""""""""""""""""\___/ ===
      ~~~ {~~ ~~~~ ~~~ ~~~~ ~~~ ~ /  ===- ~~~
           \_____ o           __/
             \    \         __/
              \____\_______/

docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at https://docs.docker.com


Start interactive shell

Kite@DESKTOP-D128J0A MINGW64 /c/Program Files/Docker Toolbox
$ docker pull chkrish9/msp-pb-phase3:2
2: Pulling from chkrish9/msp-pb-phase3
Digest: sha256:7ac07c85e32407ab52de4f42bdb405aaadb05e8cef94fe1755b989d42e5f599c
Status: Image is up to date for chkrish9/msp-pb-phase3:2

Kite@DESKTOP-D128J0A MINGW64 /c/Program Files/Docker Toolbox
$ docker run -it chkrish9/msp-pb-phase3:2 bash
root@c3e0aa440030:~# ls
data.json  derby.log  metastore_db
root@c3e0aa440030:~#
```

MINGW64:/c/Program Files/Docker Toolbox

```
Start interactive shell

Kite@DESKTOP-D128J0A MINGW64 /c/Program Files/Docker Toolbox
$ docker pull chkrish9/msp-pb-phase3:2
2: Pulling from chkrish9/msp-pb-phase3
Digest: sha256:7ac07c85e32407ab52de4f42bdb405aaadb05e8cef94fe1755b989d42e5f599c
Status: Image is up to date for chkrish9/msp-pb-phase3:2

Kite@DESKTOP-D128J0A MINGW64 /c/Program Files/Docker Toolbox
$ docker run -it chkrish9/msp-pb-phase3:2 bash
root@c6097b0b07ff:~# ls
data.json  derby.log  metastore_db
root@c6097b0b07ff:~# spark-shell
Spark context Web UI available at http://172.17.0.2:4040
Spark context available as 'sc' (master = local[*], app id = local-1558120102150).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.2.0
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_131)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

MINGW64:/c/Program Files/Docker Toolbox

```
data.json  derby.log  metastore_db
root@c6097b0b07ff:~# spark-shell
Spark context Web UI available at http://172.17.0.2:4040
Spark context available as 'sc' (master = local[*], app id = local-1558120102150).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.2.0
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_131)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val df = spark.read.json("data.json")
df: org.apache.spark.sql.DataFrame = [_corrupt_record: string, contributors: string ... 36 more fields]

scala> import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.SQLContext

scala> val sqlContext = new 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@1e08f0cd

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

scala>
```