INTRODUCTION

There are many datasets available in the twitter. Understanding of each data set is difficult. To make it easy to understand we perform some operations on it using different algorithms.

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.

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

SOFTWARE'S USED:

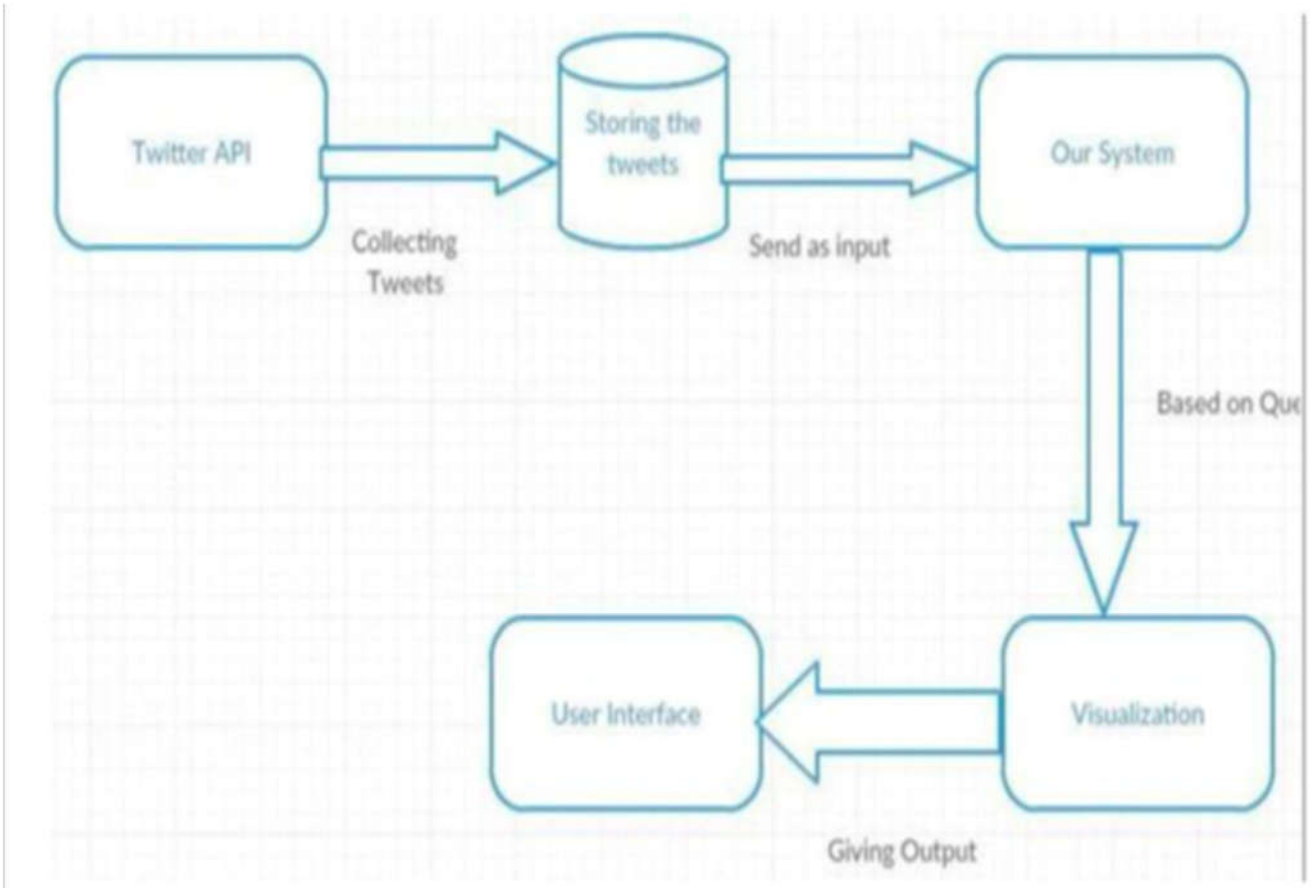
- Apache Spark and Scala
- IntelliJ IDEA
- Pycharm
- Twitter Developer Account
- Twitter4J Library for Tweets Extraction
- Tableau

DATASET : Twitter dataset (Phones / E accessories)

Detailed Description of Data set :

This data set contains data mainly about e accessories i.e mobiles . We have chosen this dataset because mobiles are used by everyone and there are different kinds of mobiles available in the market and analyzing this data makes it more interesting.

ARCHITECTURE :


Twitter API
Collecting Tweets
Storing the tweets
Send as input
Our System
Based on Qu
Visualization
User Interface
Giving Output

IMPLEMENTATION:

Wordcount is done first in Apache Spark and Hadoop

Consumer API keys

jdY2pqlZ5uVWds4CQGIMfwxiN **(API key)**

hfjqljwzHCVG0fCLBDV9vFTJUSkXpX9uhcoy7uvFR7qJq96zGZ **(API secret key)**

Access token & access token secret

1042137305329352705-9uqx96PFyct3pTezrr29RVTSn6VXgB **(Access token)**

DLswmmzGGy4KRQReYF14hwgOetvbxqAP0N7mhuMwEwEg**(Access token secret)**
Read and write (Access level)

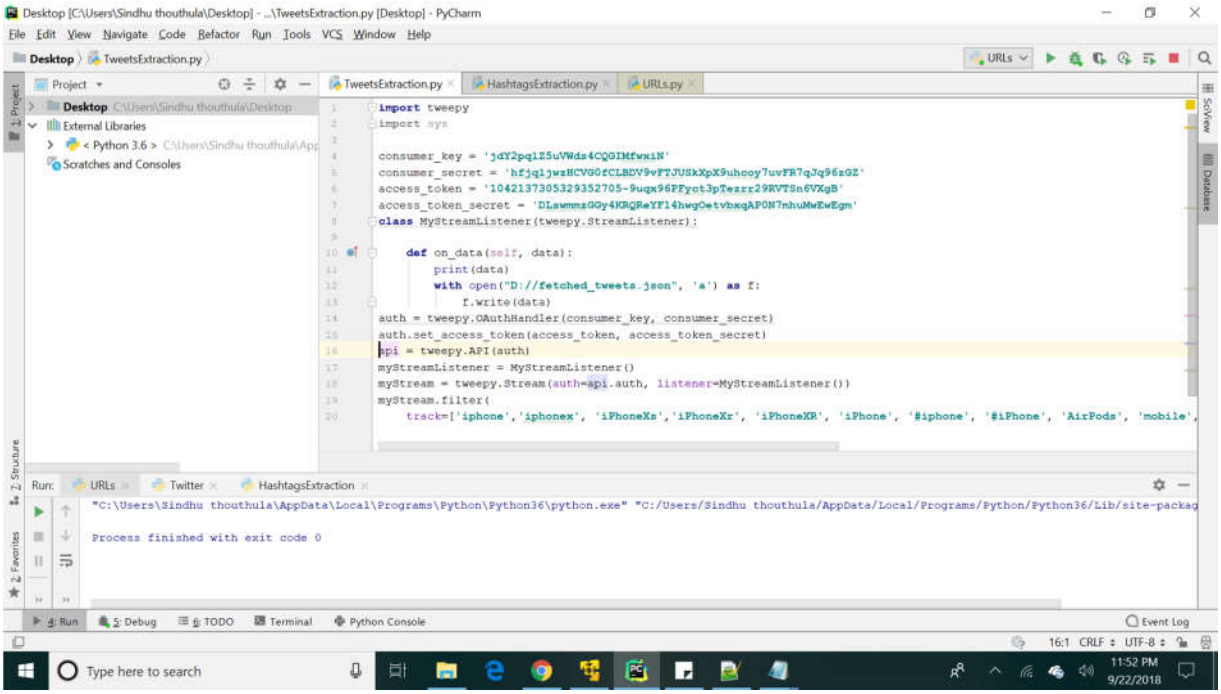
- To collect the tweets in JSON format, a Python program is written, the output of the program contains the tweets with all the details like the IDs, URLs, Hashtags, Created at, Text etc.

RELATED SCREENSHOTS :

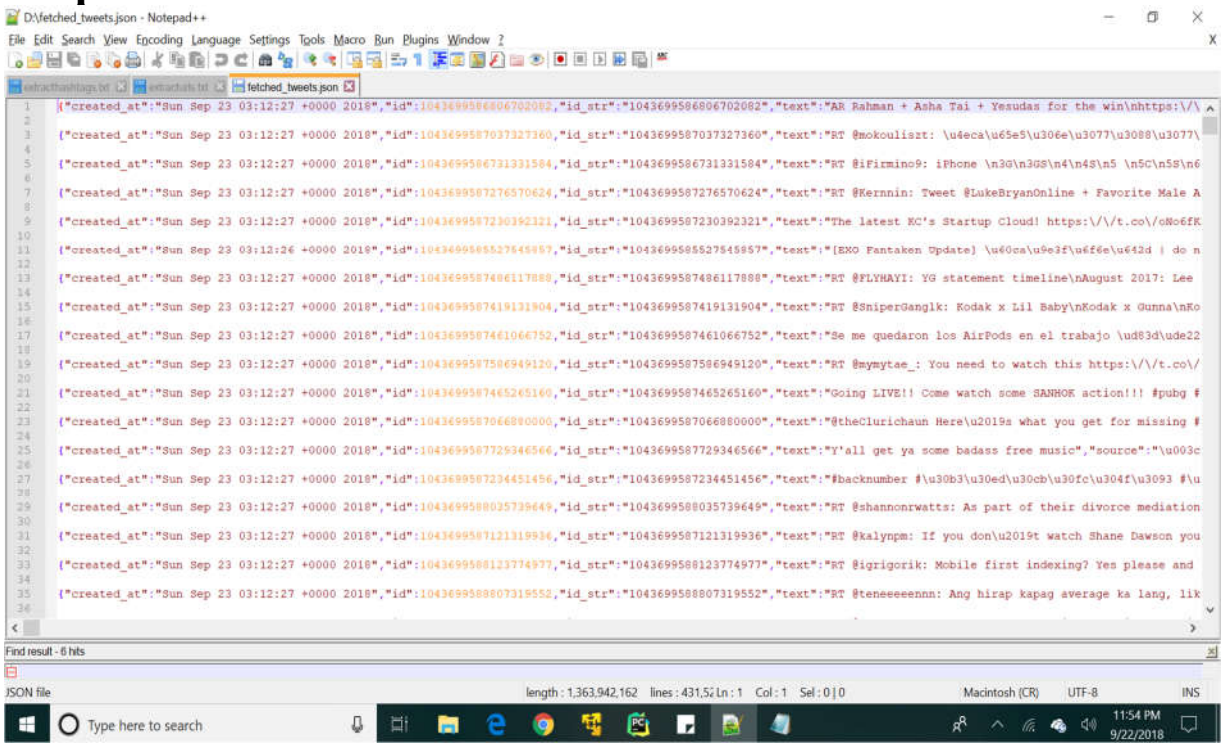
Tokens and Keys Generated :

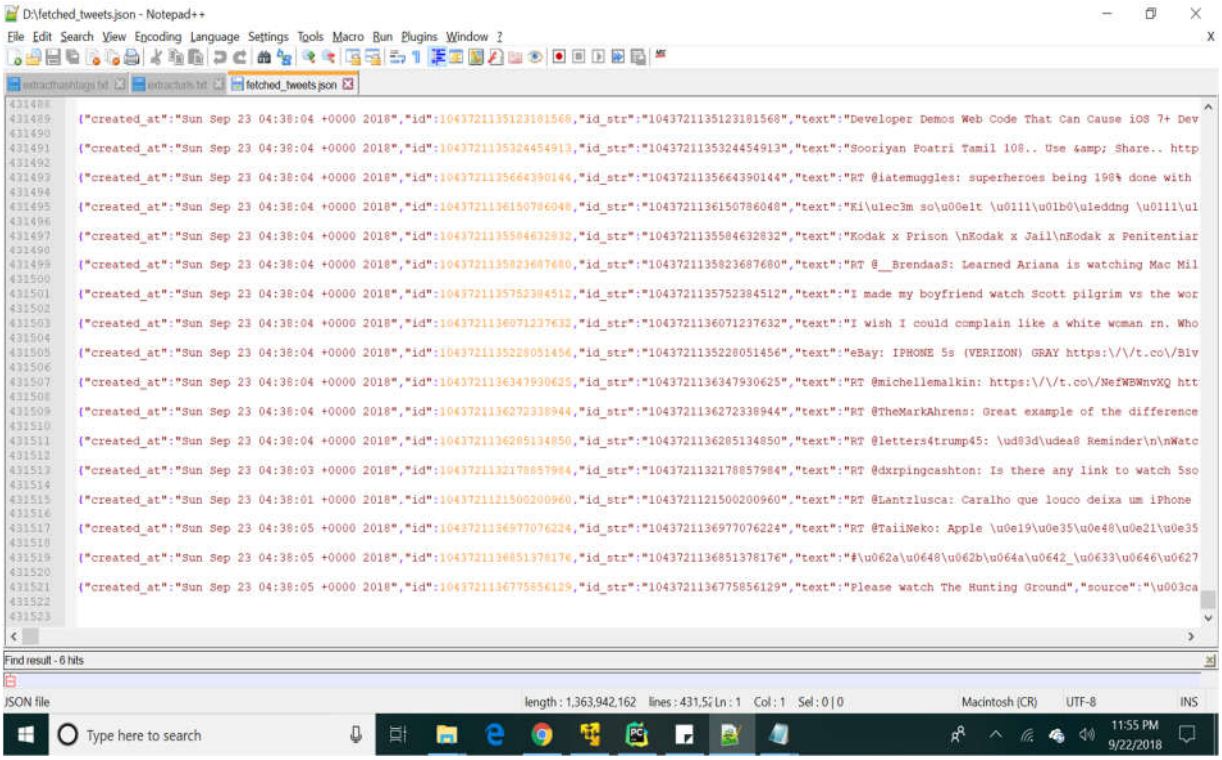
```
Consumer API keys:-

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 :


Desktop [C:\Users\Sindhu thouthula\Desktop] - ...\TweetsExtraction.py [Desktop] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
TweetsExtraction.py
HashtagsExtraction.py
URLs.py
import tweepy
import sys
consumer_key = 'jdY2pqlZ5uVWds4CQGIMfwxiN'
consumer_secret = 'hfjqljwzHCVG0fCLBDV9vFTJUSkXpX9uhooy7uvFR7qJq96zGZ'
access_token = '1042137305329352705-9uqx96PFyot3pTezrr29RVTSn6VXgB'
access_token_secret = 'DLswmmzGGy4KRQReYF14hwgOetvbxgAP0M7mhuMwEwEgm'
class MyStreamListener(tweepy.StreamListener):
def on_data(self, data):
print(data)
with open("D://fetched_tweets.json", 'a') as f:
f.write(data)
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)
myStreamListener = MyStreamListener()
myStream = tweepy.Stream(auth=api.auth, listener=MyStreamListener())
myStream.filter(
track=['iphone','iphonex', 'iPhoneXs','iPhoneXr', 'iPhoneXR', 'iPhone', '#iphone', '#iPhone', 'AirPods', 'mobile',
Run: URLs Twitter HashtagsExtraction
"C:\Users\Sindhu thouthula\AppData\Local\Programs\Python\Python36\python.exe" "C:/Users/Sindhu thouthula/AppData/Local/Programs/Python/Python36/Lib/site-packag
Process finished with exit code 0
Run Debug TODO Terminal Python Console
Event Log
16:1 CRLF UTF-8
Type here to search
11:52 PM
9/22/2018

Output:

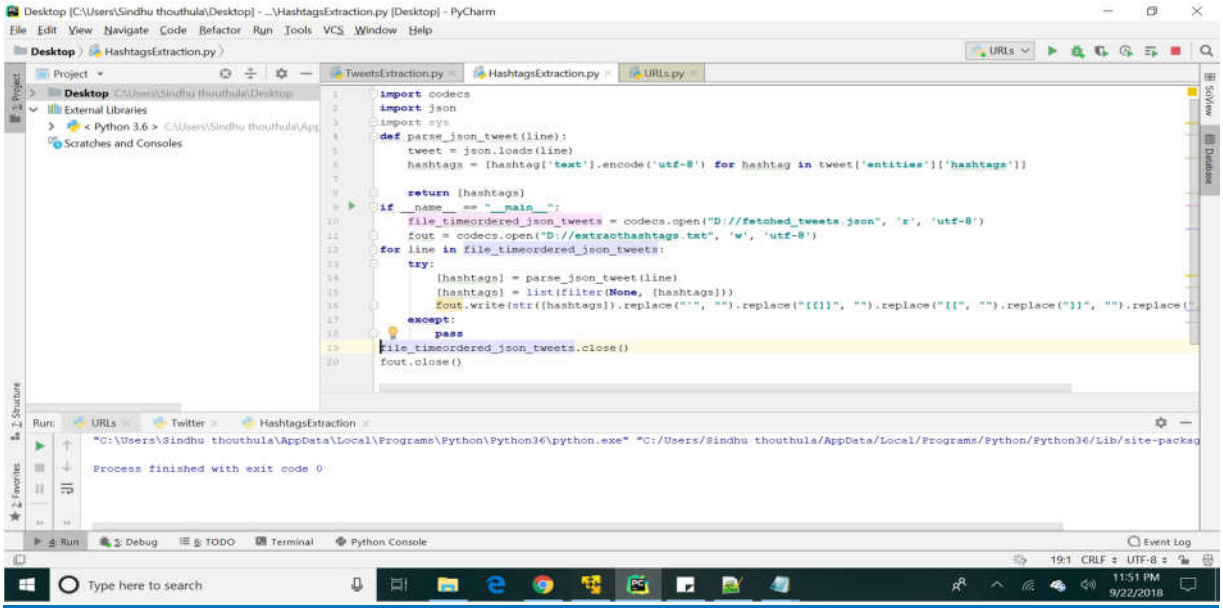

D:\fetched_tweets.json - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
fetched_tweets.json
{"created_at":"Sun Sep 23 03:12:27 +0000 2018","id":1043699586806702082,"id_str":"1043699586806702082","text":"AR Rahman + Asha Tai + Yesudas for the win\nhttps:\/\
{"created_at":"Sun Sep 23 03:12:27 +0000 2018","id":1043699587037327360,"id_str":"1043699587037327360","text":"RT @mokouliszt: \u4eca\u65e5\u306e\u3077\u3088\u3077\
{"created_at":"Sun Sep 23 03:12:27 +0000 2018","id":1043699586731331584,"id_str":"1043699586731331584","text":"RT @iFirmino9: iPhone \n3G\n3GS\n4\n4S\n5 \n5C\n5S\n6
{"created_at":"Sun Sep 23 03:12:27 +0000 2018","id":1043699587276570624,"id_str":"1043699587276570624","text":"RT @Kernnin: Tweet @LukeBryanOnline + Favorite Male A
{"created_at":"Sun Sep 23 03:12:27 +0000 2018","id":1043699587230392321,"id_str":"1043699587230392321","text":"The latest KC's Startup Cloud! https:\/\/t.co\/oNo6fK
{"created_at":"Sun Sep 23 03:12:26 +0000 2018","id":1043699585527545857,"id_str":"1043699585527545857","text":"[EXO Fantaken Update] \u60ca\u9e3f\u6f6e\u642d | do n
{"created_at":"Sun Sep 23 03:12:27 +0000 2018","id":1043699587486117888,"id_str":"1043699587486117888","text":"RT @FLYHAYI: YG statement timeline\nAugust 2017: Lee
{"created_at":"Sun Sep 23 03:12:27 +0000 2018","id":1043699587419131904,"id_str":"1043699587419131904","text":"RT @SniperGanglk: Kodak x Lil Baby\nKodak x Gunna\nKo
{"created_at":"Sun Sep 23 03:12:27 +0000 2018","id":1043699587461066752,"id_str":"1043699587461066752","text":"Se me quedaron los AirPods en el trabajo \ud83d\ude22
{"created_at":"Sun Sep 23 03:12:27 +0000 2018","id":1043699587586949120,"id_str":"1043699587586949120","text":"RT @mymytae_: You need to watch this https:\/\/t.co\/
{"created_at":"Sun Sep 23 03:12:27 +0000 2018","id":1043699587465265160,"id_str":"1043699587465265160","text":"Going LIVE!! Come watch some SANHOK action!!! #pubg #
{"created_at":"Sun Sep 23 03:12:27 +0000 2018","id":1043699587066880000,"id_str":"1043699587066880000","text":"@theClurichaun Here\u2019s what you get for missing #
{"created_at":"Sun Sep 23 03:12:27 +0000 2018","id":1043699587729346566,"id_str":"1043699587729346566","text":"Y'all get ya some badass free music","source":"\u003c
{"created_at":"Sun Sep 23 03:12:27 +0000 2018","id":1043699587234451456,"id_str":"1043699587234451456","text":"#backnumber #\u30b3\u30ed\u30cb\u30fc\u304f\u3093 #\u
{"created_at":"Sun Sep 23 03:12:27 +0000 2018","id":1043699588035739649,"id_str":"1043699588035739649","text":"RT @shannonrwatts: As part of their divorce mediation
{"created_at":"Sun Sep 23 03:12:27 +0000 2018","id":1043699587121319936,"id_str":"1043699587121319936","text":"RT @kalynpm: If you don\u2019t watch Shane Dawson you
{"created_at":"Sun Sep 23 03:12:27 +0000 2018","id":1043699588123774977,"id_str":"1043699588123774977","text":"RT @igrigorik: Mobile first indexing? Yes please and
{"created_at":"Sun Sep 23 03:12:27 +0000 2018","id":1043699588807319552,"id_str":"1043699588807319552","text":"RT @teneeeeennn: Ang hirap kapag average ka lang, lik
Find result - 6 hits
JSON file
length : 1,363,942,162 lines : 431,52 Ln : 1 Col : 1 Sel : 0 | 0
Macintosh (CR) UTF-8 INS
Type here to search
11:54 PM
9/22/2018



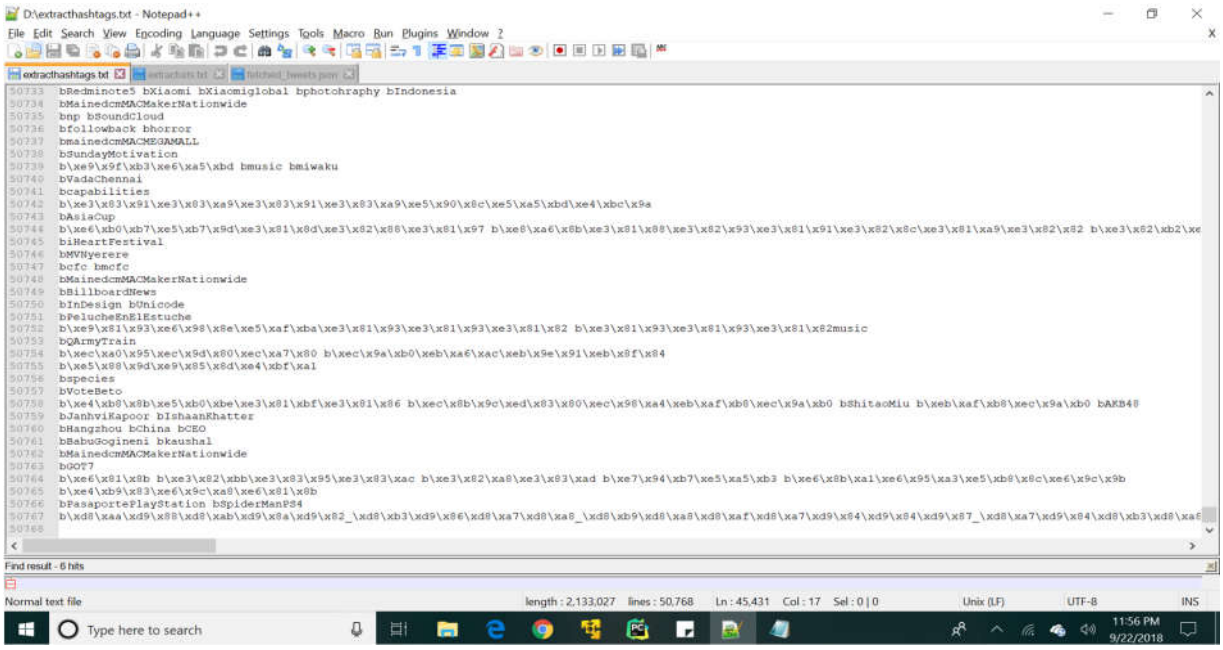
2) HASHTAGS AND URLS EXTRACTION :

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.

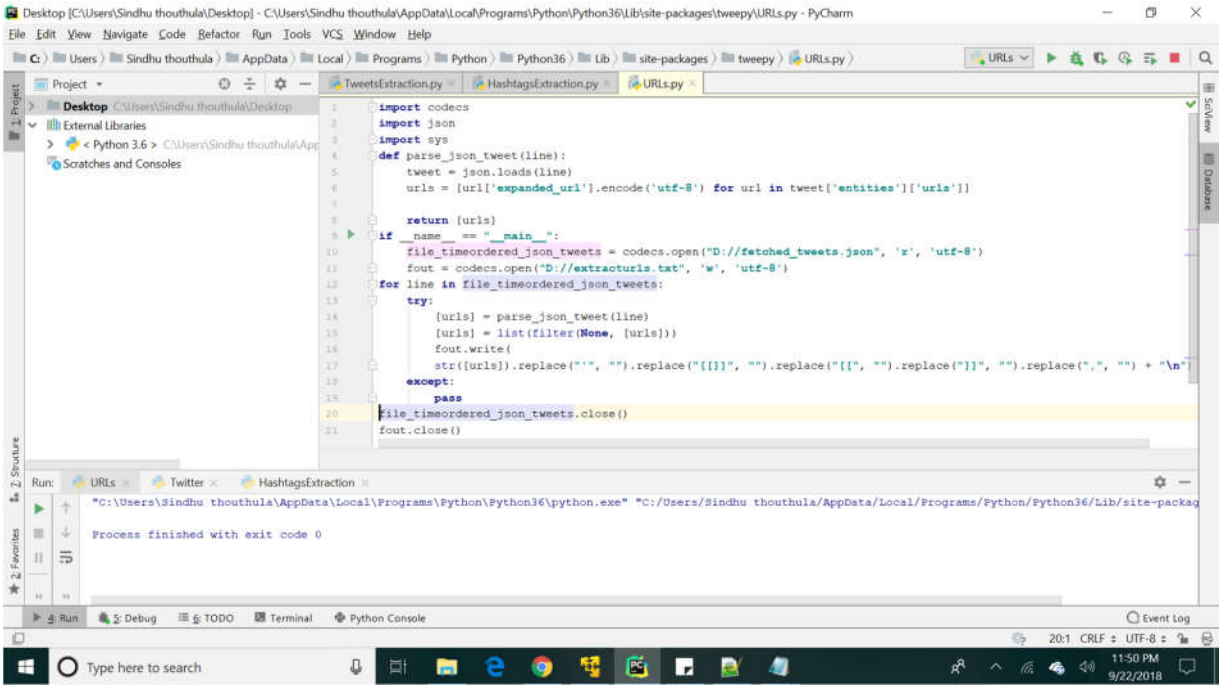
Hashtags Code :



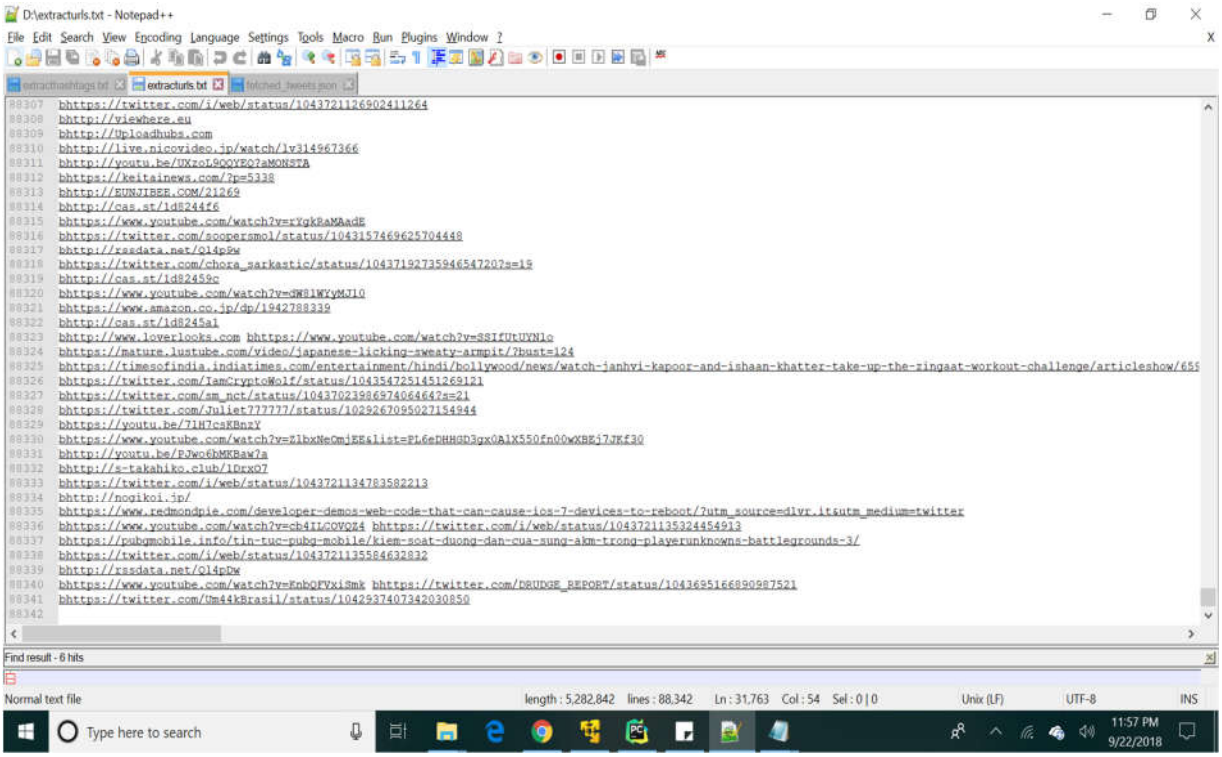
Output:



URLs Code :


Desktop [C:\Users\Sindhu thouthula\Desktop] - C:\Users\Sindhu thouthula\AppData\Local\Programs\Python\Python36\Lib\site-packages\tweepy\URLs.py - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
TweetsExtraction.py
HashtagsExtraction.py
URLs.py
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()
Run: URLs Twitter HashtagsExtraction
"C:\Users\Sindhu thouthula\AppData\Local\Programs\Python\Python36\python.exe" "C:/Users/Sindhu thouthula/AppData/Local/Programs/Python/Python36/Lib/site-packag
Process finished with exit code 0
4: Run 5: Debug 6: TODO Terminal Python Console
Event Log
20:1 CRLF UTF-8
Type here to search
11:50 PM
9/22/2018

Output :


D:\extracturls.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
extracthashtags.txt
extracturls.txt
fetched_tweets.json
88307 bhttps://twitter.com/i/web/status/1043721126902411264
88308 bhttp://viewhere.eu
88309 bhttp://Uploadhubs.com
88310 bhttp://live.nicovideo.jp/watch/lv314967366
88311 bhttp://youtu.be/UXzoL9QQYEQ?aMONSTA
88312 bhttps://keitainews.com/?p=5338
88313 bhttp://EUNJIBEE.COM/21269
88314 bhttp://cas.st/1d8244f6
88315 bhttps://www.youtube.com/watch?v=rYgkRaMAadE
88316 bhttps://twitter.com/soopersmol/status/1043157469625704448
88317 bhttp://rssdata.net/Ql4p9w
88318 bhttps://twitter.com/chora_sarkastic/status/1043719273594654720?s=19
88319 bhttp://cas.st/1d82459c
88320 bhttps://www.youtube.com/watch?v=dW81WYyMJ10
88321 bhttps://www.amazon.co.jp/dp/1942788339
88322 bhttp://cas.st/1d8245a1
88323 bhttp://www.loverlooks.com bhttps://www.youtube.com/watch?v=SSIfUtUYN1o
88324 bhttps://mature.lustube.com/video/japanese-licking-sweaty-armpit/?bust=124
88325 bhttps://timesofindia.indiatimes.com/entertainment/hindi/bollywood/news/watch-janhvi-kapoor-and-ishaan-khatter-take-up-the-zingaat-workout-challenge/articleshow/65
88326 bhttps://twitter.com/IamCryptoWolf/status/1043547251451269121
88327 bhttps://twitter.com/sm_nct/status/1043702398697406464?s=21
88328 bhttps://twitter.com/Juliet777777/status/1029267095027154944
88329 bhttps://youtu.be/71H7csKBnzY
88330 bhttps://www.youtube.com/watch?v=ZlbxNeOmjEE&list=PL6eDHH0D3gx0AlX550fn00WXBEj7JKf30
88331 bhttp://youtu.be/PJwo6bMKBaw?a
88332 bhttp://s-takahiko.club/1DrxO7
88333 bhttps://twitter.com/i/web/status/1043721134783582213
88334 bhttp://nogikoi.jp/
88335 bhttps://www.redmondpie.com/developer-demos-web-code-that-can-cause-ios-7-devices-to-reboot/?utm_source=dlvr.it&utm_medium=twitter
88336 bhttps://www.youtube.com/watch?v=cb4ILCOVQZ4 bhttps://twitter.com/i/web/status/1043721135324454913
88337 bhttps://pubgmobile.info/tin-tuc-pubg-mobile/kiem-soat-duong-dan-cua-sung-akm-trong-playerunknowns-battlegrounds-3/
88338 bhttps://twitter.com/i/web/status/1043721135584632832
88339 bhttp://rssdata.net/Ql4pDw
88340 bhttps://www.youtube.com/watch?v=EnbQFVxi3mk bhttps://twitter.com/DBUDGE_REPORT/status/1043695166890987521
88341 bhttps://twitter.com/Um44kBrasil/status/1042937407342030850
88342
Find result - 6 hits
Normal text file
length : 5,282,842 lines : 88,342 Ln : 31,763 Col : 54 Sel : 0 | 0
Unix (LF) UTF-8 INS
Type here to search
11:57 PM
9/22/2018

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

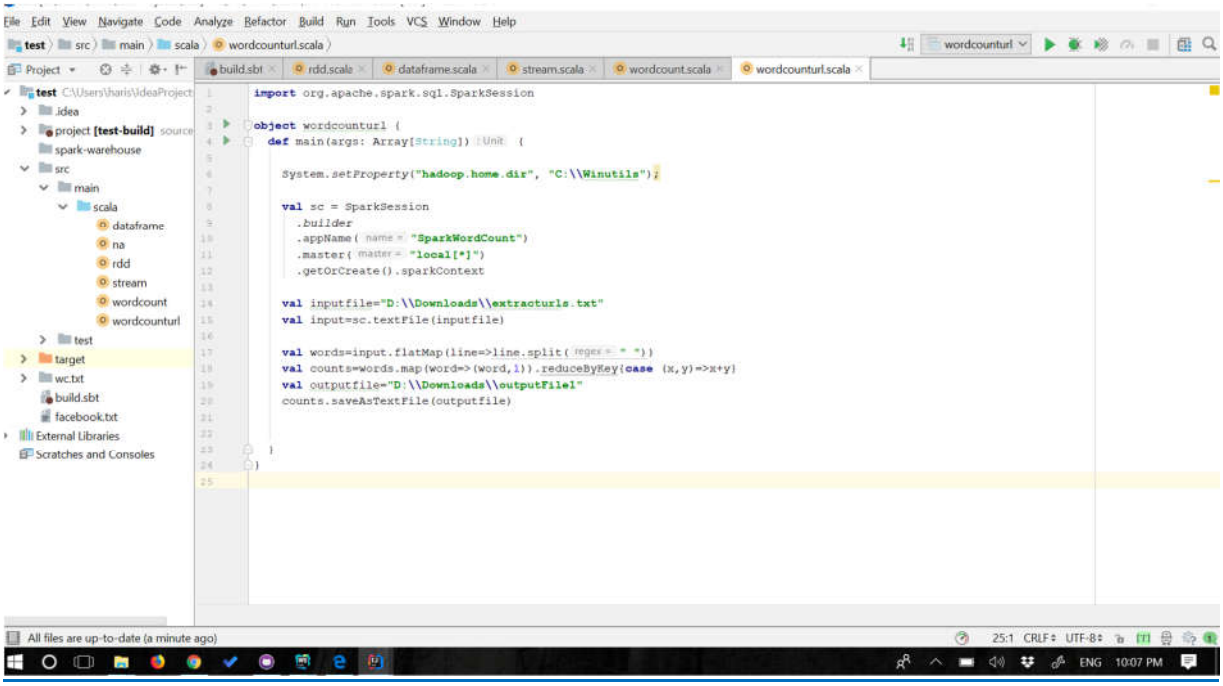
Word Count in Apache Spark :

- Here we made use of IntelliJ IDEA to run the Word Count program on the extracted Hashtags and URLs in Apache Spark and Scala.
- Here in this IDEA to execute the Word Count on the extracted hashtags and URLs, Spark is integrated on Scala.
- We have given the values and input files using the following :

```
val inputfile="D:\\Downloads\\extracthashtags.txt"
val input=sc.textFile (inputfile)

val words=input.flatMap(line=>line.split(regex= " "))
val counts=words.map (word=>(word, 1)).reduceByKey(case
(x,y)=>x+y)
val outputfile="D:\\Downloads\\outputFile"
counts.saveAsTextFile (outputfile)
```

- Here the sc.text file in Spark creates Resilent Distributed Datasets with each line considered as an element.
- RDDs are immutable and partitioned collection of records which can only be created by coarse grained operations such as the map, filter, group etc.
- Once the file is read and stored in the variable textfile, wordcount is executed with the help of the MapReduce function.
- MapReduce function is broken into two phases, the Map phase and Reduce phase where each has got a key value pair as an input and output.
- Once the reduce function groups the words with the count, the collected data is stored in the output file.

Spark :

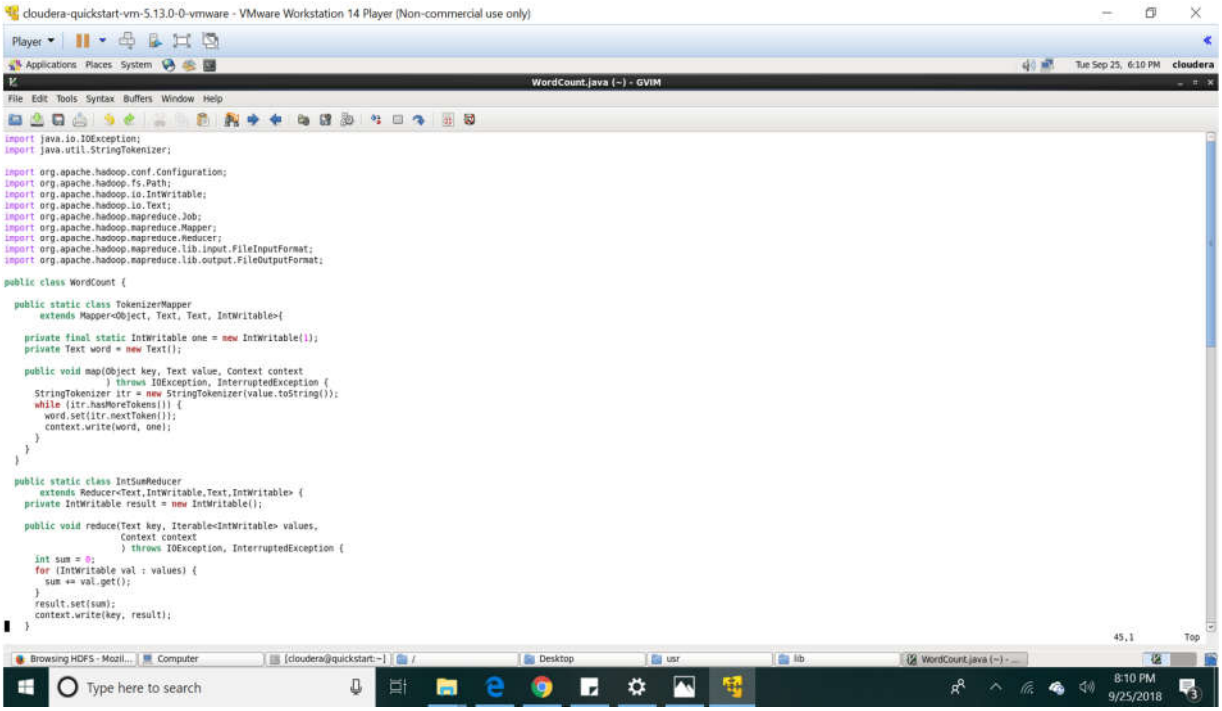

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
test src main scala wordcounturl.scala
wordcounturl
Project
build.sbt rdd.scala dataframe.scala stream.scala wordcount.scala wordcounturl.scala
test C:\Users\haris\IdeaProject
.idea
project [test-build]
spark-warehouse
src
main
scala
dataframe
na
rdd
stream
wordcount
wordcounturl
test
target
wc.txt
build.sbt
facebook.txt
External Libraries
Scratches and Consoles
import org.apache.spark.sql.SparkSession
object wordcounturl {
def main(args: Array[String]) : Unit = {
System.setProperty("hadoop.home.dir", "C:\\Winutils");
val sc = SparkSession
.builder
.appName(name = "SparkWordCount")
.master(master = "local[*]")
.getOrCreate().sparkContext
val inputfile="D:\\Downloads\\extracturls.txt"
val input=sc.textFile(inputfile)
val words=input.flatMap(line=>line.split(regex = " "))
val counts=words.map(word=>(word,1)).reduceByKey{case (x,y)=>x+y}
val outputfile="D:\\Downloads\\outputFile1"
counts.saveAsTextFile(outputfile)
}
}
All files are up-to-date (a minute ago)
25:1 CRLF UTF-8
ENG 10:07 PM

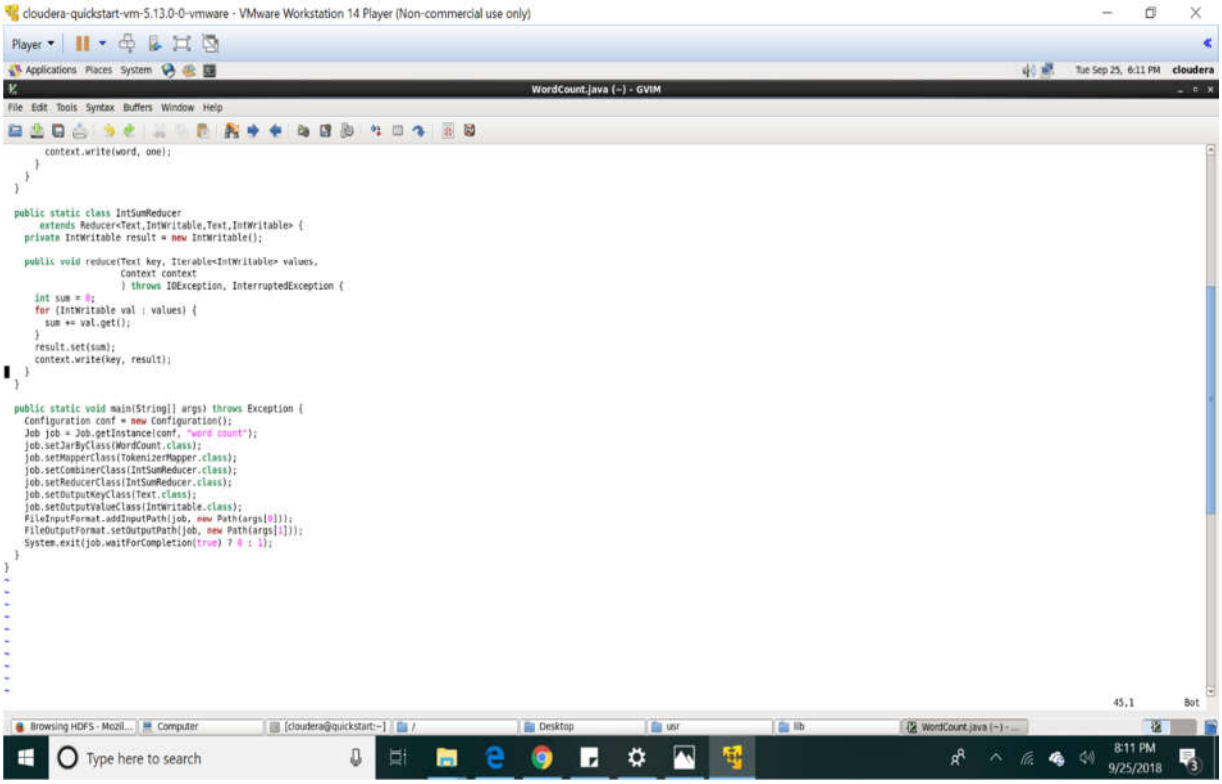
WordCount in Hadoop :

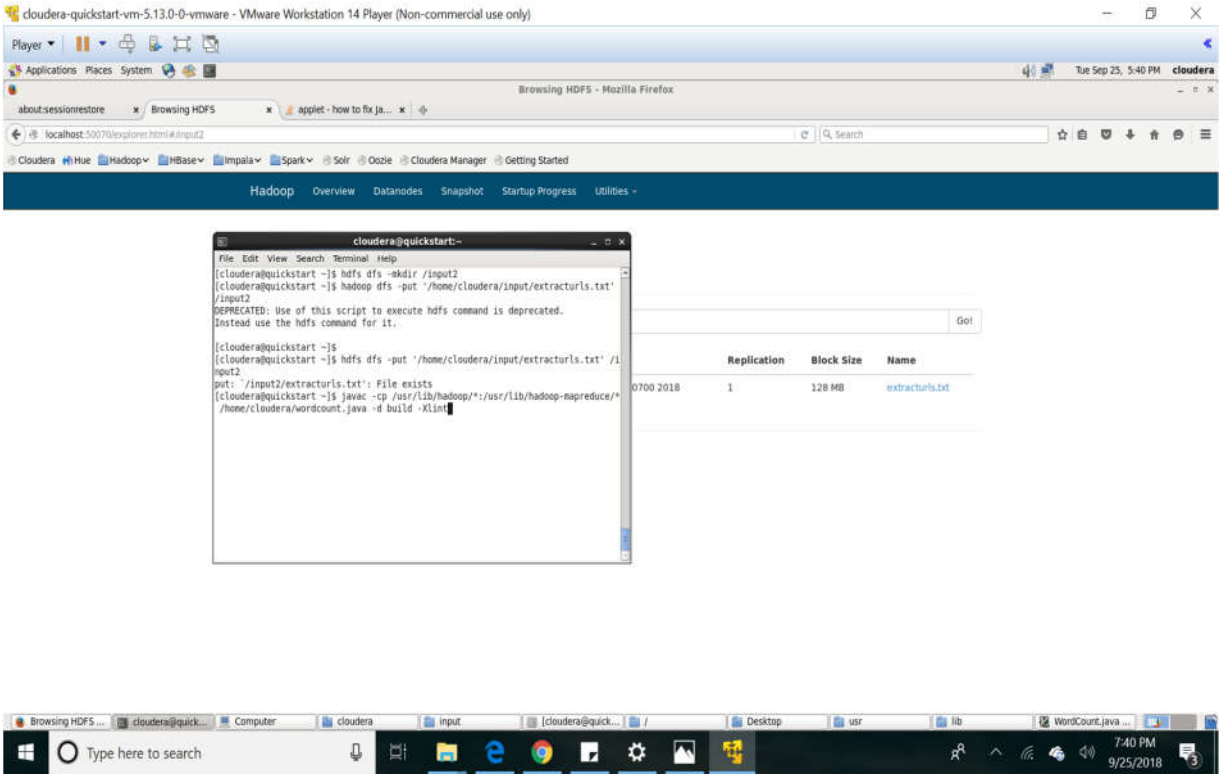
- To run the wordcount, we have written the code in Java language.
- In this program, the TokenizerMapper class is created with the extension of Mapper class and by overriding the map method in the Mapper class, we implemented the Map Function.
- Here the Mapper function takes a key value pair as an input and outputs the key value pair.
- The IntSumReducer class is created by extending the org.apache.hadoop.mapreduce.Reducer class and the reduce method is implemented by overriding the reduce method from the Reducer class.
- The main method in the code is to setup all the configurations and runs the map reduce job.
- In Hadoop to execute the hashtags and URLs, the .jar file is exported to cloudera.
- Using the following commands, the wordcount on hashtags and URLs is performed 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
```

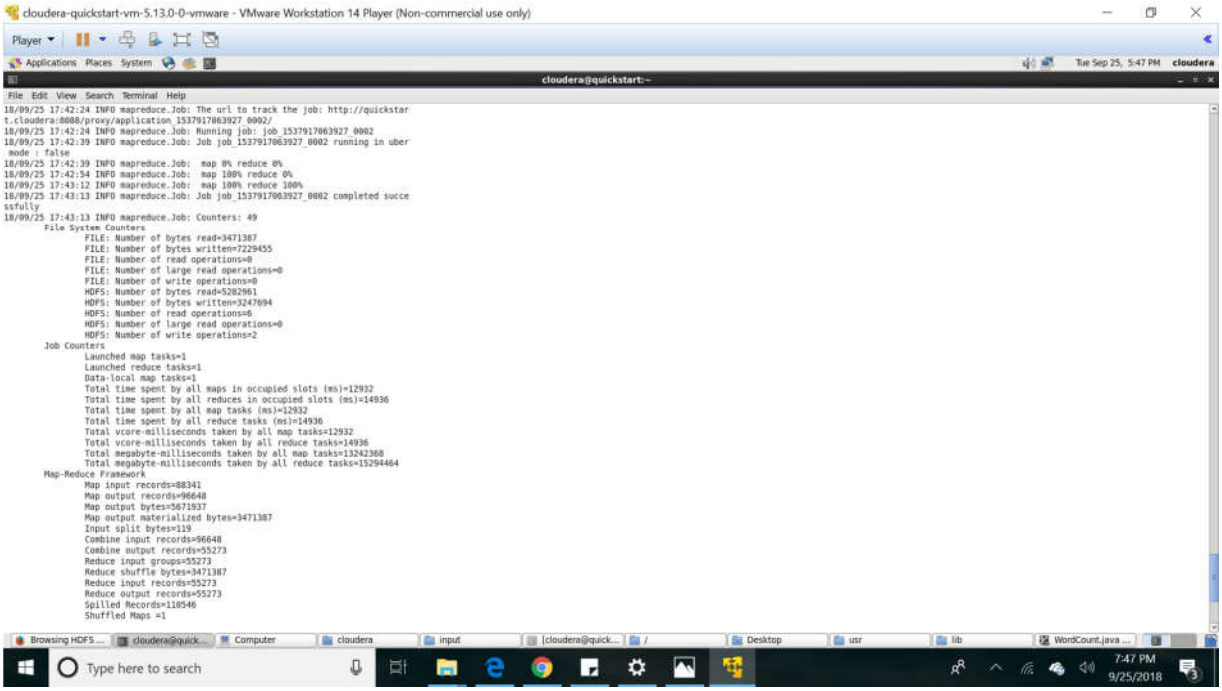
WordCount Java Program – Hadoop :


cloudera-quickstart-vm-5.13.0-0-vmware - VMware Workstation 14 Player (Non-commercial use only)
Player
Applications Places System
Tue Sep 25, 6:10 PM cloudera
WordCount.java (~) - GVIM
File Edit Tools Syntax Buffers Window Help
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCount {
public static class TokenizerMapper
extends Mapper<Object, Text, Text, IntWritable>{
private final static IntWritable one = new IntWritable(1);
private Text word = new Text();
public void map(Object key, Text value, Context context
) throws IOException, InterruptedException {
StringTokenizer itr = new StringTokenizer(value.toString());
while (itr.hasMoreTokens()) {
word.set(itr.nextToken());
context.write(word, one);
}
}
}
public static class IntSumReducer
extends Reducer<Text,IntWritable,Text,IntWritable> {
private IntWritable result = new IntWritable();
public void reduce(Text key, Iterable<IntWritable> values,
Context context
) throws IOException, InterruptedException {
int sum = 0;
for (IntWritable val : values) {
sum += val.get();
}
result.set(sum);
context.write(key, result);
}
}
45,1 Top
Browsing HDFS - Mozil... Computer [cloudera@quickstart:~] / Desktop usr lib WordCount.java (~) - ...
Type here to search
8:10 PM
9/25/2018


cloudera-quickstart-vm-5.13.0-0-vmware - VMware Workstation 14 Player (Non-commercial use only)
Player
Applications Places System
Tue Sep 25, 6:11 PM cloudera
WordCount.java (~) - GVIM
File Edit Tools Syntax Buffers Window Help
context.write(word, one);
}
}
}
public static class IntSumReducer
extends Reducer<Text,IntWritable,Text,IntWritable> {
private IntWritable result = new IntWritable();
public void reduce(Text key, Iterable<IntWritable> values,
Context context
) throws IOException, InterruptedException {
int sum = 0;
for (IntWritable val : values) {
sum += val.get();
}
result.set(sum);
context.write(key, result);
}
}
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
Job job = Job.getInstance(conf, "word count");
job.setJarByClass(WordCount.class);
job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(IntSumReducer.class);
job.setReducerClass(IntSumReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
45,1 Bot
Browsing HDFS - Mozil... Computer [cloudera@quickstart:~] / Desktop usr lib WordCount.java (~) - ...
Type here to search
8:11 PM
9/25/2018



Output : map 100% reduce 100%

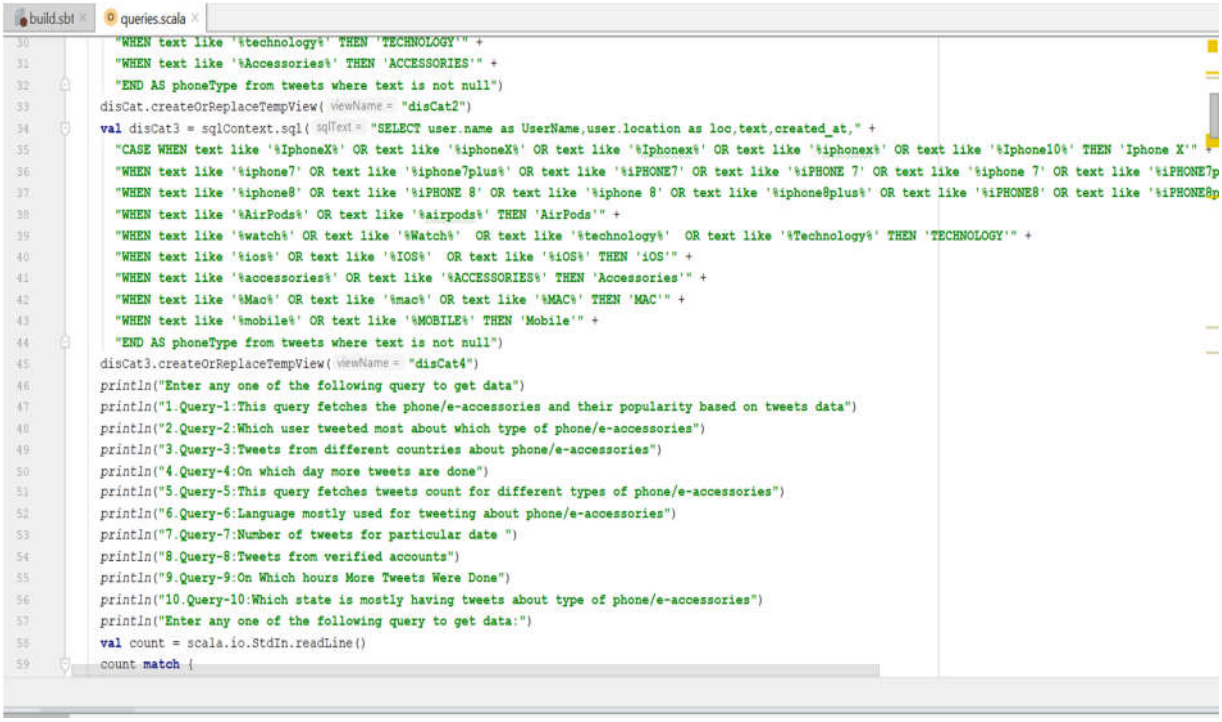


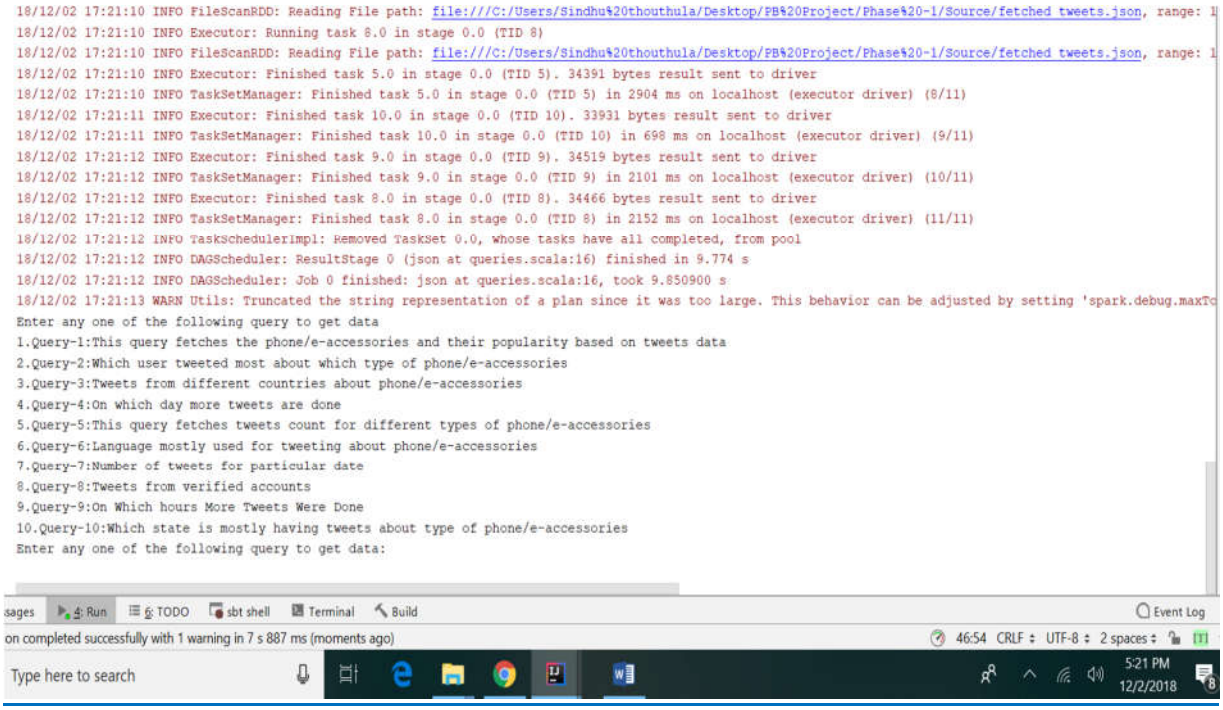
- Initially collected the tweets in JSON format for which a Python program is written, the output of the program contains the tweets with all the details like the IDs, URLs, Hashtags, Created at, Text etc.
- The twitter data is collected on the concept based on to analyze & visualize the data regarding 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.

ANALYTICAL QUERIES:

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





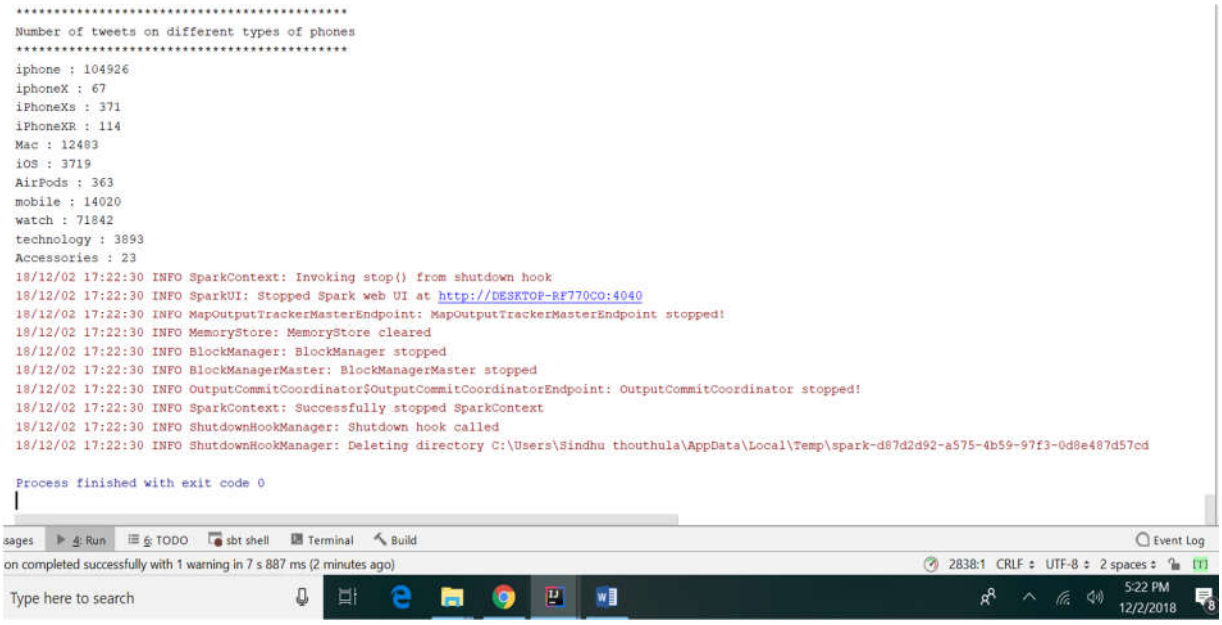
1.Query to fetch the tweets of phones/e-accessories based on the popularity.

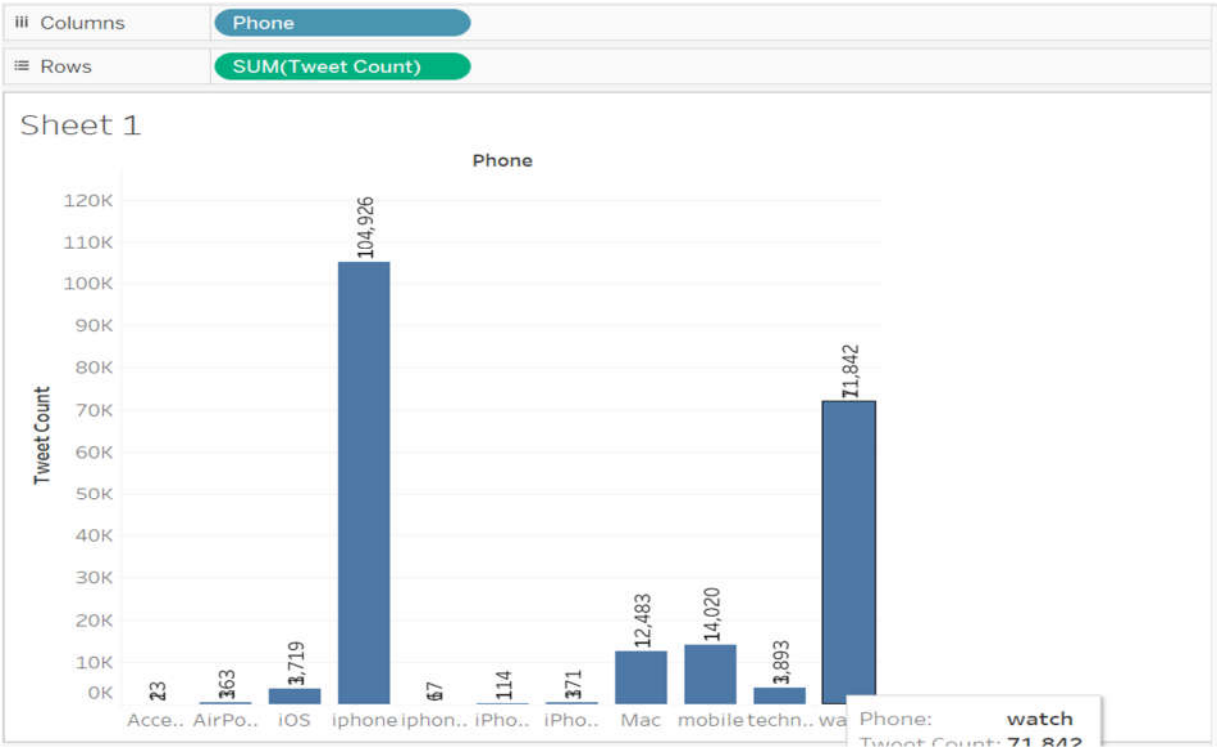
This query is written to analyze the tweets that are made by the users on the particular key-words which would reflect the popularity of the phones/e-accessories. It would result the count i.e., how many times the phone/e-accessories would appear in the tweets made.

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))
```

Output :





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

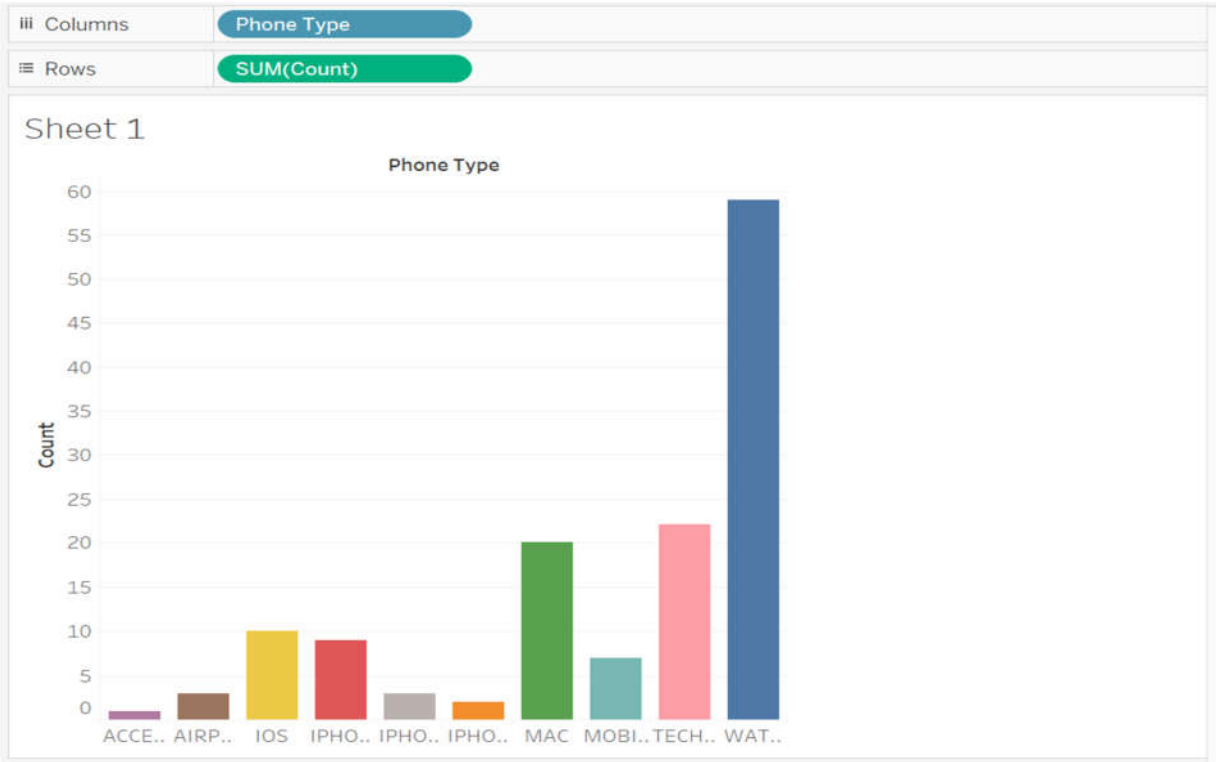
This query is written to find the user that most tweeted about the phone/e-accessories so that it would result the count of how many times a user tweeted at most for each kind particularly.

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 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 :

```
18/12/02 17:25:21 INFO DAGScheduler: Job 3 finished: show at queries.scala:119, took 0.240341 s
+--------------------+-----------+-----+
|            UserName|  phoneType|count|
+--------------------+-----------+-----+
|     Art_By_Lamarcus|     IPHONE|    9|
|                はら|   IPHONEXS|    2|
|        Shane Salkey|   IPHONEXR|    3|
|    jenna montemayor|        MAC|   20|
|Muhammad Farhan B...|        IOS|   10|
|          Ryan Ngala|    AIRPODS|    3|
| Bharti Airtel India|     MOBILE|    7|
|       94.9 THE BULL|      WATCH|   59|
|              ramana| TECHNOLOGY|   22|
|     Official_TfShop|ACCESSORIES|    1|
+--------------------+-----------+-----+

18/12/02 17:25:21 INFO SparkContext: Invoking stop() from shutdown hook
18/12/02 17:25:21 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-RF77OCO:4040
18/12/02 17:25:21 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/12/02 17:25:23 INFO MemoryStore: MemoryStore cleared
18/12/02 17:25:23 INFO BlockManager: BlockManager stopped
18/12/02 17:25:23 INFO BlockManagerMaster: BlockManagerMaster stopped
18/12/02 17:25:23 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/12/02 17:25:23 INFO SparkContext: Successfully stopped SparkContext
18/12/02 17:25:23 INFO ShutdownHookManager: Shutdown hook called
18/12/02 17:25:23 INFO ShutdownHookManager: Deleting directory C:\Users\Sindhu thouthula\AppData\Local\Temp\spark-f11bdba4-dc07-4da7-a022-6d8f5f5ba85f

Process finished with exit code 0
```



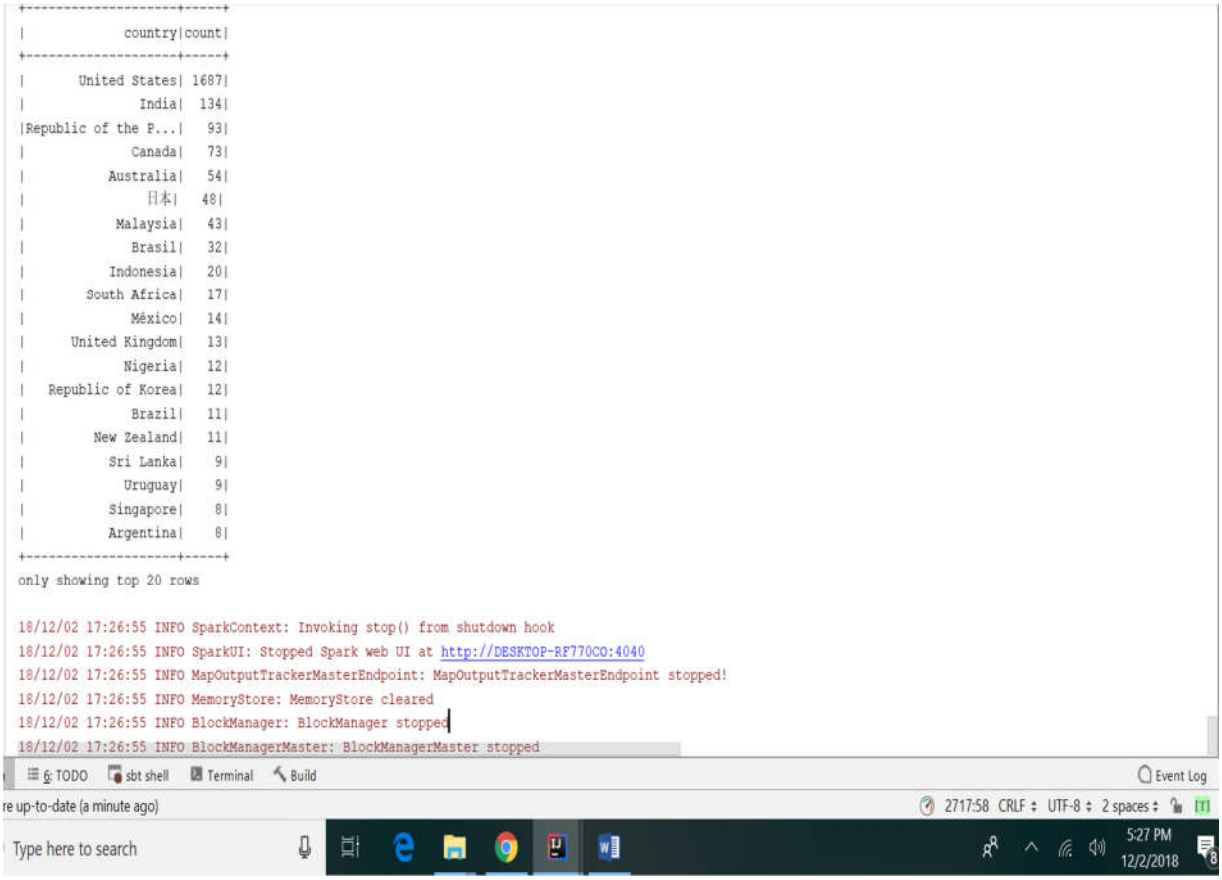
3.Query for fetching tweets from different countries.

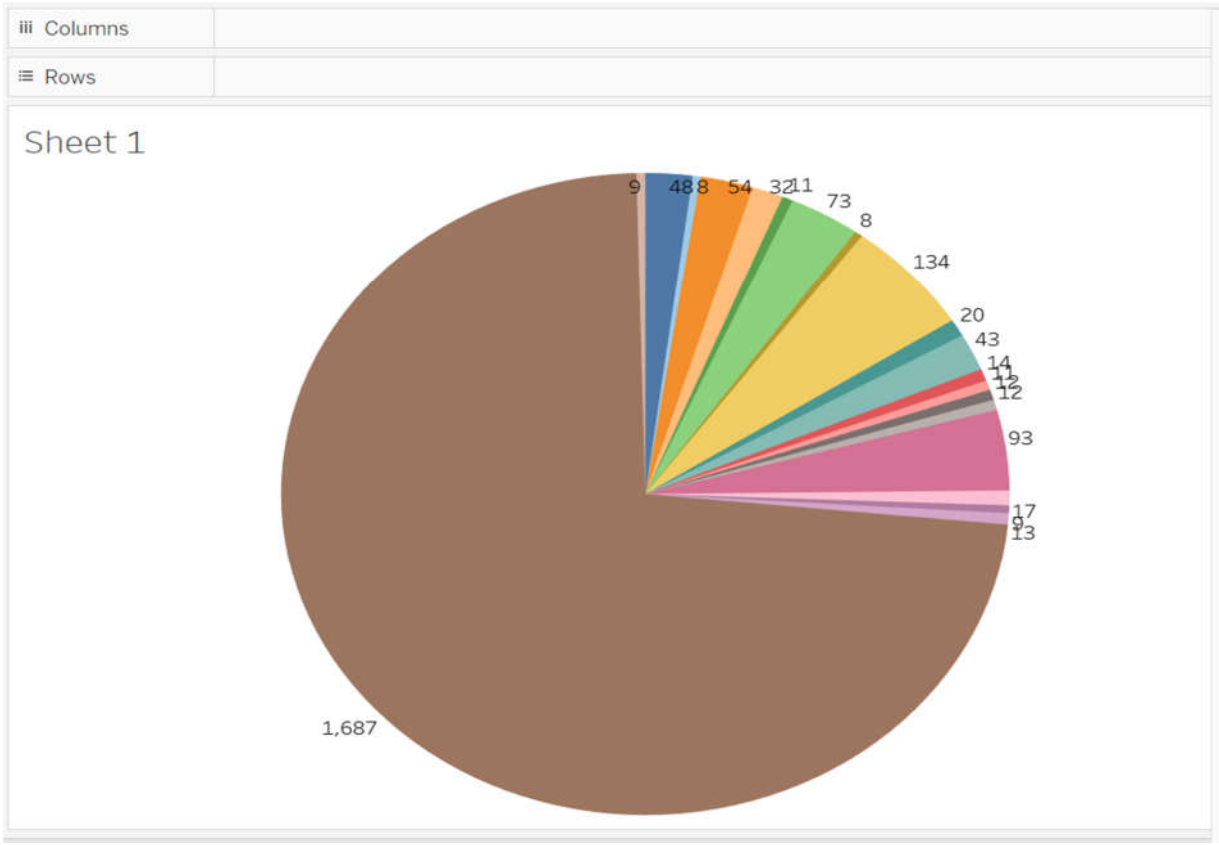
This query is written to find the tweets based on the locations such that it would count how many tweets are posted about the phones/e-accessories from different countries.

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()
```

Output :


+--------------------+-----+
| country|count|
+--------------------+-----+
| United States| 1687|
| India| 134|
|Republic of the P...| 93|
Canada	73
Australia	54
日本	48
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-RF770CO: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
6: TODO
sbt shell
Terminal
Build
Event Log
re up-to-date (a minute ago)
2717:58
CRLF
UTF-8
2 spaces
Type here to search
5:27 PM
12/2/2018



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

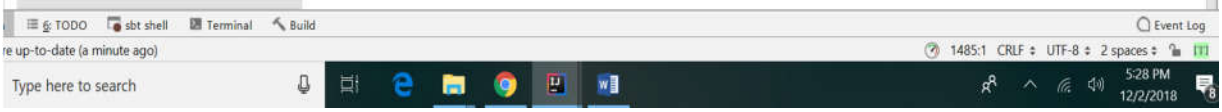
This query is written to find out the day on which more tweets were done so that it would count about giving us a figure.

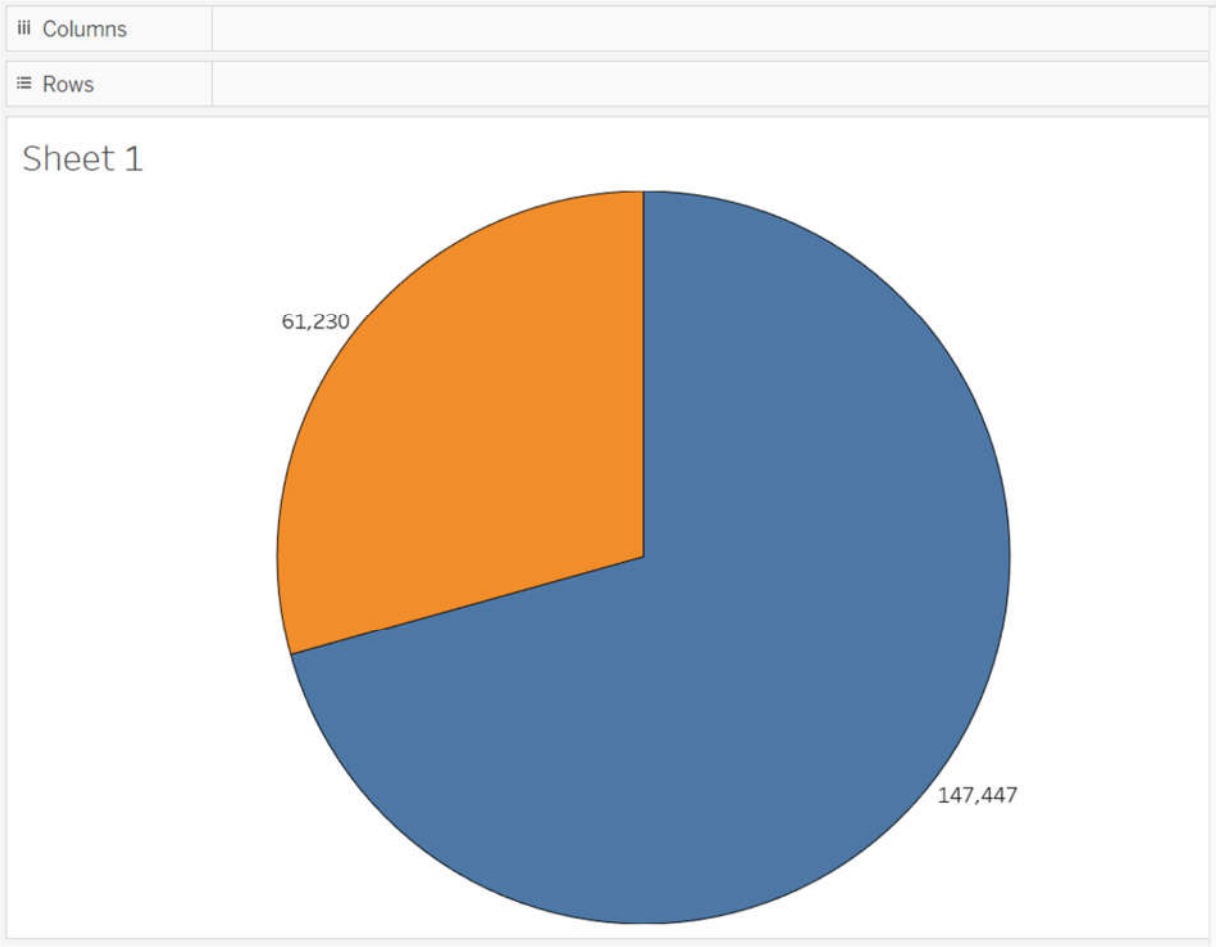
Query-Code :

```scala
/*-------------------------------Query 4 : On which Day More Tweets are posted-----------------------------------*/
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 day1 order by count(*) desc")

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

Output :

```
+-------+---------+
|    Day|Day_Count|
+-------+---------+
|WEEKDAY|   147447|
|WEEKEND|    61230|
+-------+---------+

18/12/02 17:28:36 INFO SparkContext: Invoking stop() from shutdown hook
18/12/02 17:28:36 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-RF770CO:4040
18/12/02 17:28:36 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/12/02 17:28:36 INFO MemoryStore: MemoryStore cleared
18/12/02 17:28:36 INFO BlockManager: BlockManager stopped
18/12/02 17:28:36 INFO BlockManagerMaster: BlockManagerMaster stopped
18/12/02 17:28:36 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/12/02 17:28:36 INFO SparkContext: Successfully stopped SparkContext
18/12/02 17:28:36 INFO ShutdownHookManager: Shutdown hook called
18/12/02 17:28:36 INFO ShutdownHookManager: Deleting directory C:\Users\Sindhu thouthula\AppData\Local\Temp\spark-ecf4d02d-88b4-49e9-9492-6bc5317ea4a2

Process finished with exit code 0
```





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

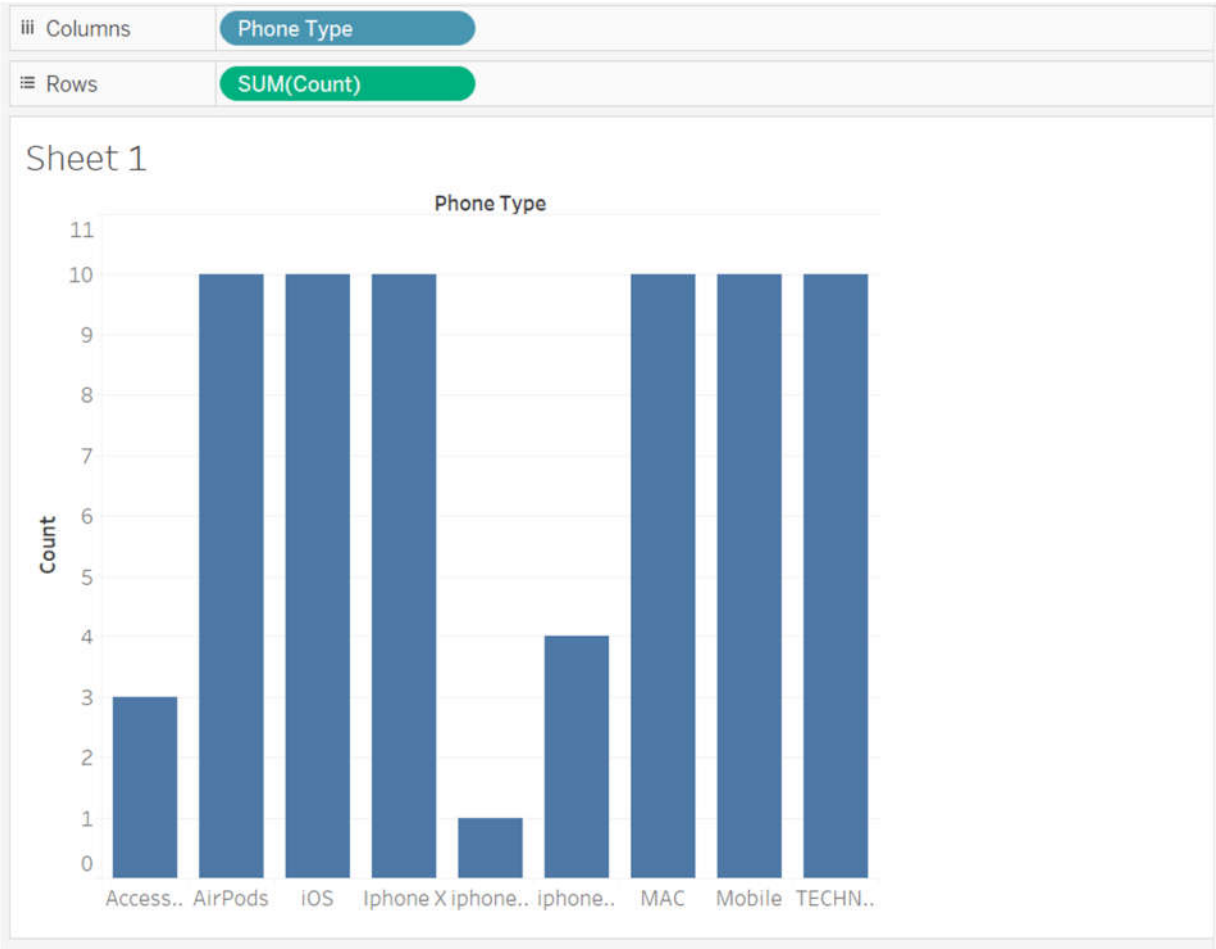
This query is written to extract the tweets that made on the different series of the phone/e-accessories so that it would count for it.

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()
```

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-RF770CO: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 thouthula\AppData\Local\Temp\spark-91623298-27d9-4328-a6f7-e08018788efe

Process finished with exit code 0
```



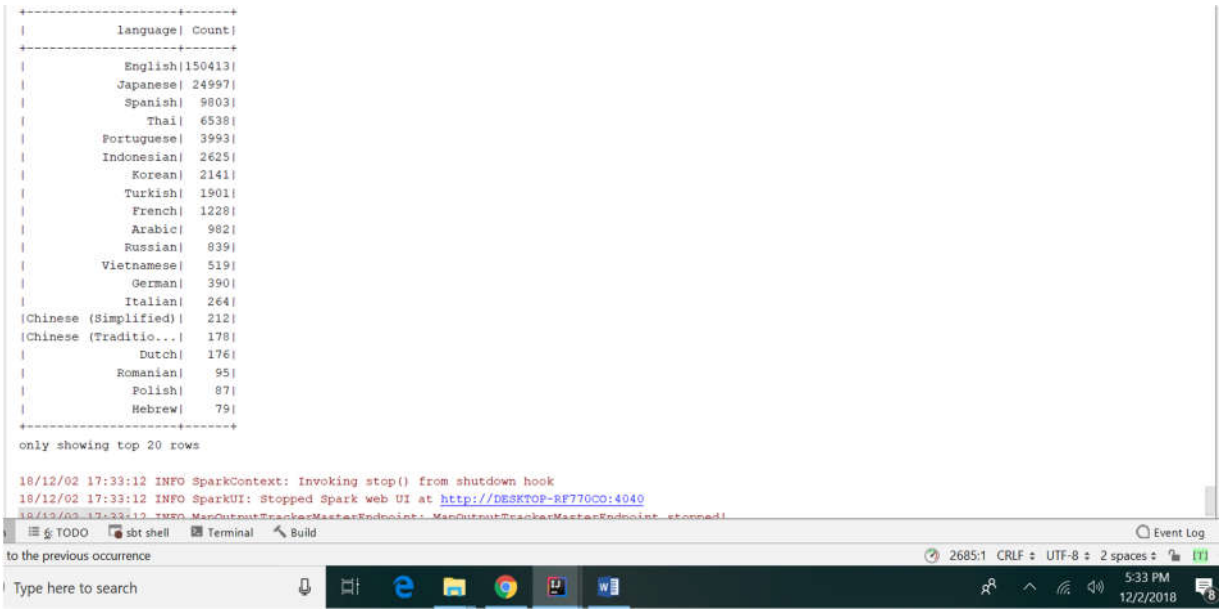
6.Query to fetch the languages mostly used for tweeting about the phone/e-accessories.

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.

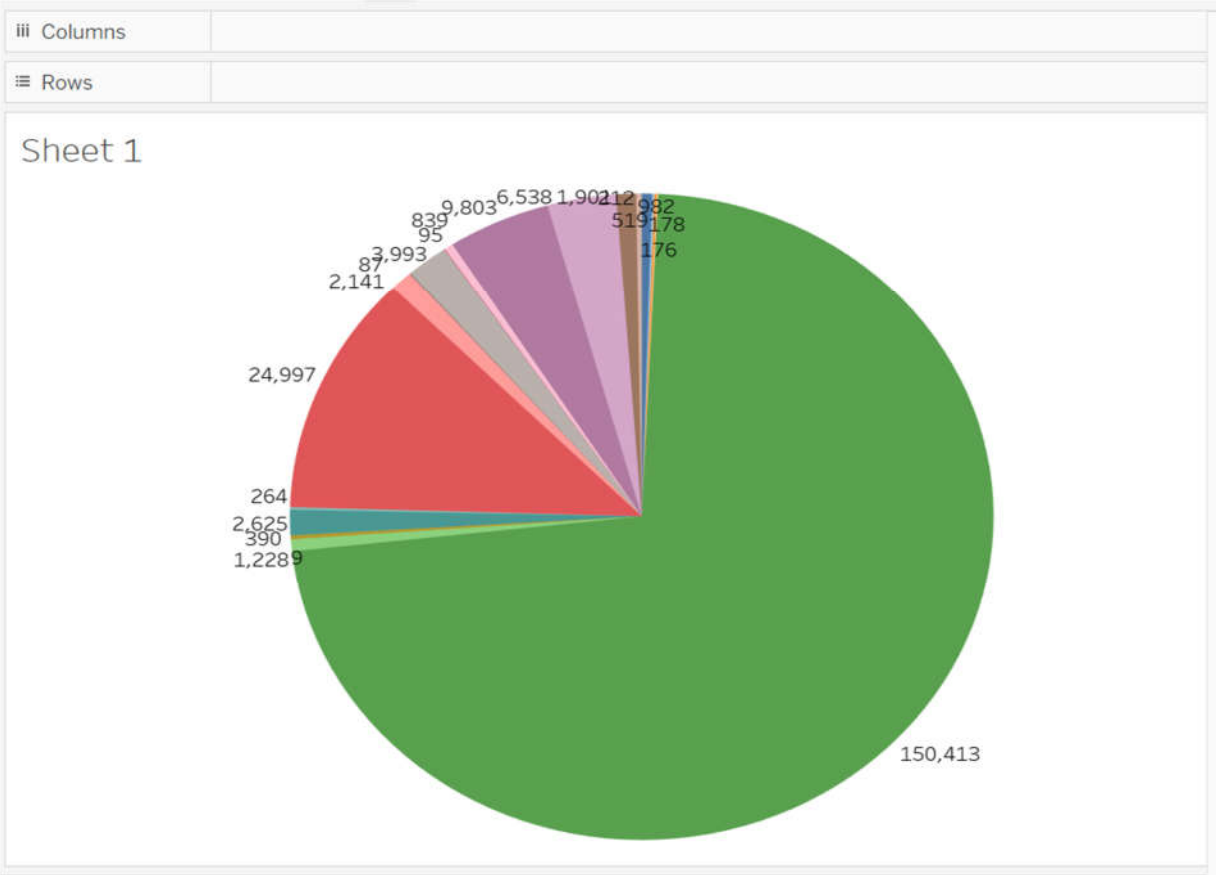
Query-Code :

```
/*----------------------------------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|   839|
|          Vietnamese|   519|
|              German|   390|
|             Italian|   264|
|Chinese (Simplified)|   212|
|Chinese (Traditio...|   178|
|               Dutch|   176|
|            Romanian|    95|
|              Polish|    87|
|              Hebrew|    79|
+--------------------+------+
only showing top 20 rows

18/12/02 17:33:12 INFO SparkContext: Invoking stop() from shutdown hook
18/12/02 17:33:12 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-RF770CO:4040
```



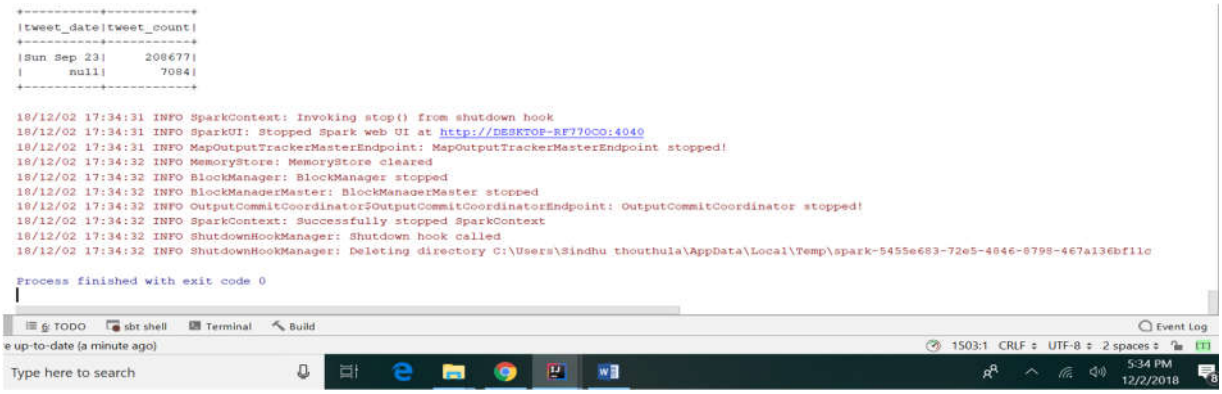
7.Query for fetching the count of the tweets made on a particular day.

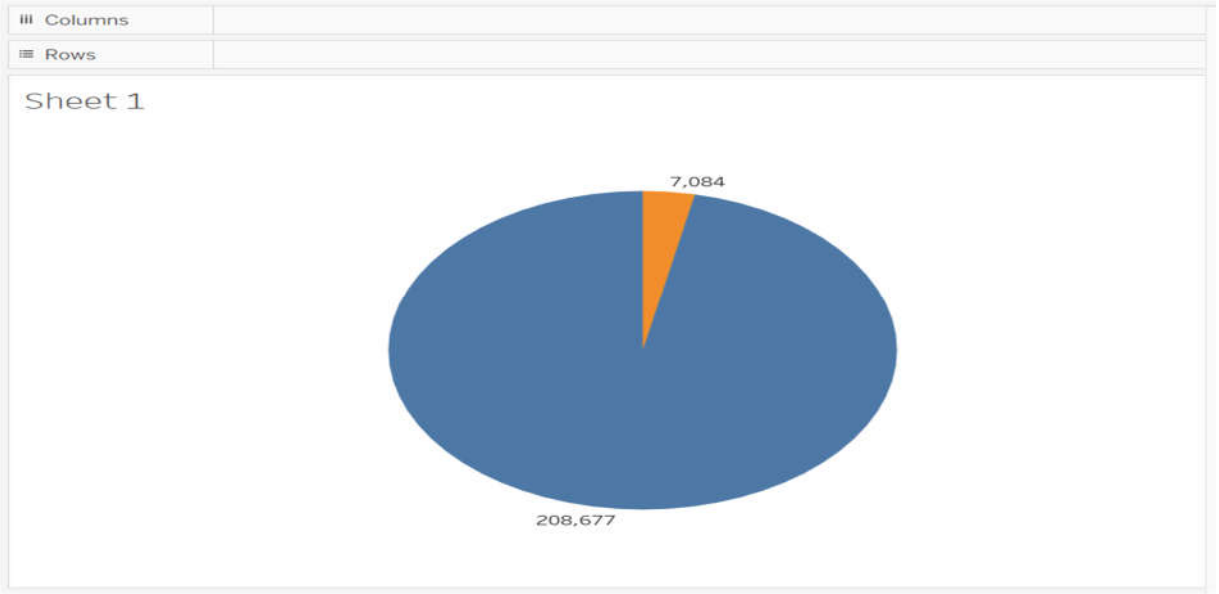
This query is written to analyze, depending upon the collected data for each day how many tweets are made.

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()
```

Output :





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

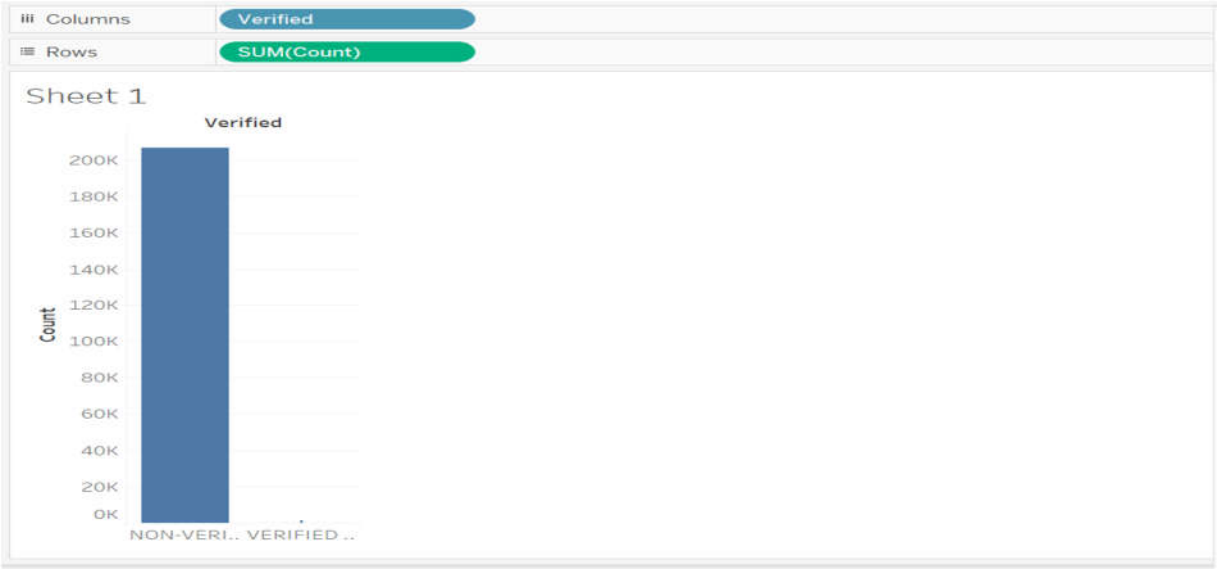
This query is written to make an analysis on the number of verified users. This query counts for the tweets made from the verified accounts only and gives us the value.

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()
```

Output :

```
+--------------------+------+
|            Verified| Count|
+--------------------+------+
|NON-VERIFIED ACCOUNT|206624|
|    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-RF770CO: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\Sindhu thouthula\AppData\Local\Temp\spark-0c2af815-7cf9-4058-9dfe-d1d96a8b1413

Process finished with exit code 0
```

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

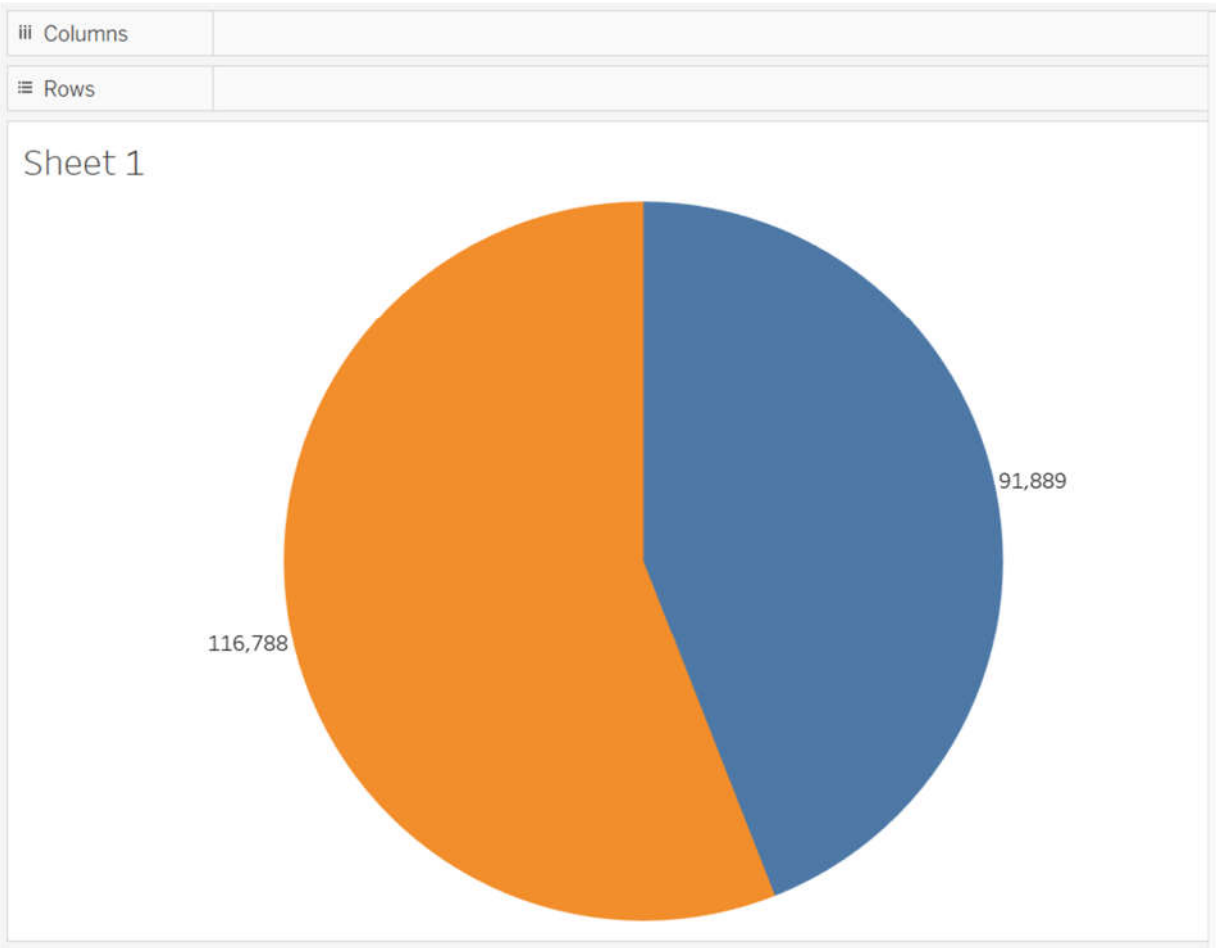
This query is written to analyze on which hours tweets are made like mornings, afternoon, evenings etc.

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("**********************************")
```

Output :

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

18/12/02 17:37:18 INFO BlockManagerInfo: Removed broadcast_4_piece0 on DESKTOP-RF770CO:55163 in memory (size: 15.5 KB, free: 892.2 MB)
18/12/02 17:37:18 INFO SparkContext: Invoking stop() from shutdown hook
18/12/02 17:37:18 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-RF770CO:4040
18/12/02 17:37:18 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/12/02 17:37:18 INFO MemoryStore: MemoryStore cleared
18/12/02 17:37:18 INFO BlockManager: BlockManager stopped
18/12/02 17:37:18 INFO BlockManagerMaster: BlockManagerMaster stopped
18/12/02 17:37:18 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/12/02 17:37:18 INFO SparkContext: Successfully stopped SparkContext
18/12/02 17:37:18 INFO ShutdownHookManager: Shutdown hook called
18/12/02 17:37:18 INFO ShutdownHookManager: Deleting directory C:\Users\Sindhu thouthula\AppData\Local\Temp\spark-212bdae2-d417-4685-81e9-67daa806a02e

Process finished with exit code 0
```



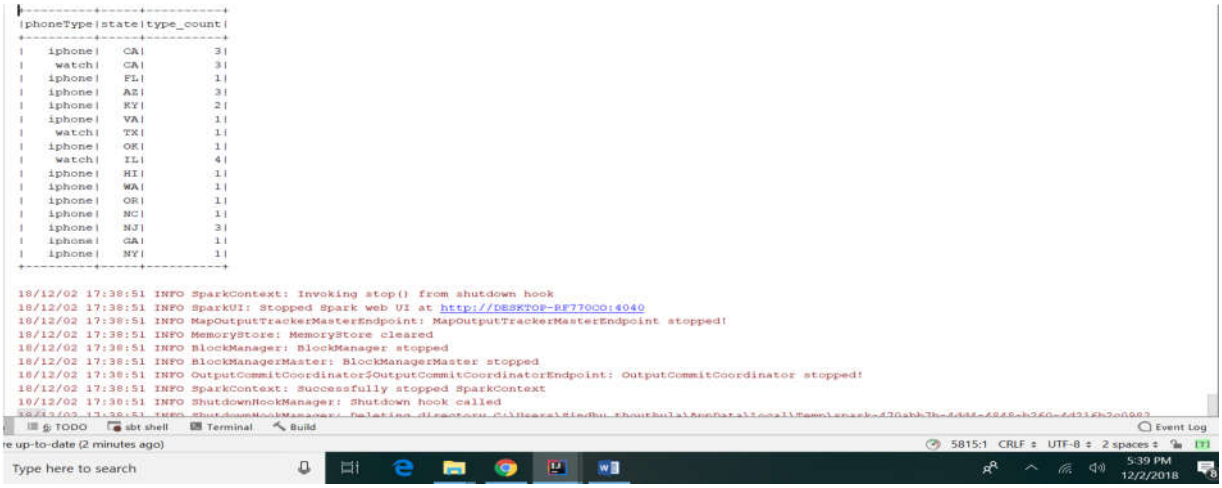
10.Query for fetching tweets based on which state has more tweets about particular type of phone/e-accessories.

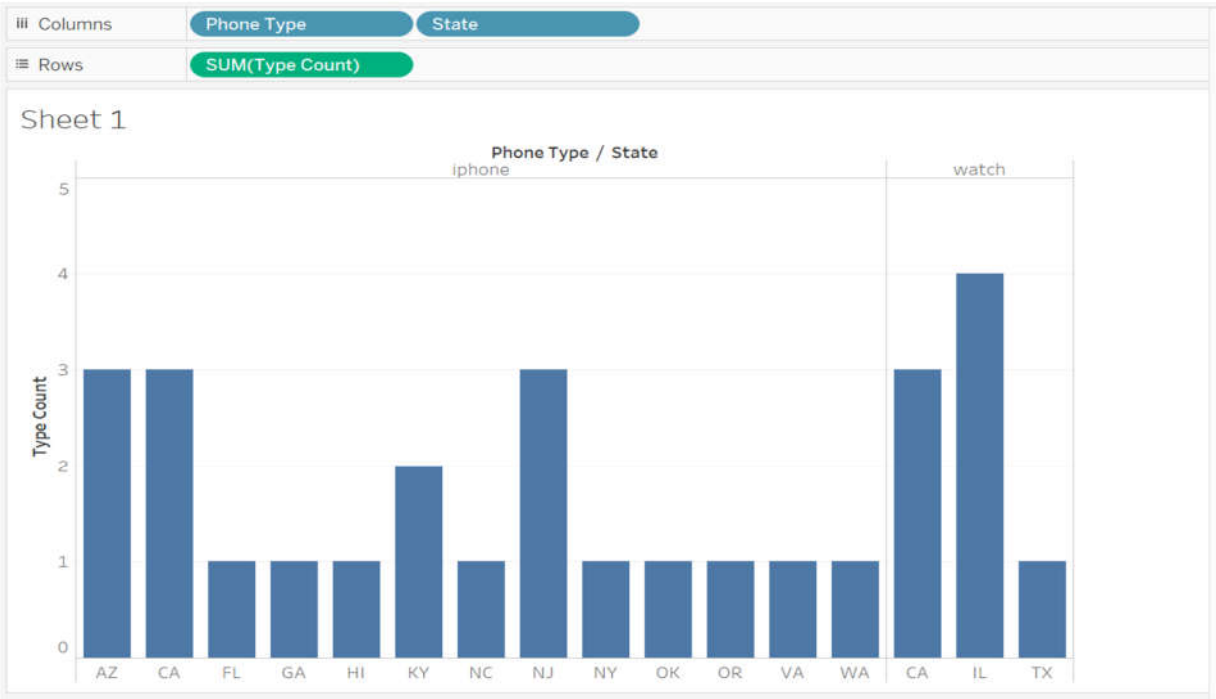
This query is written to check which state users made most number of tweets about a type of phone/e-accessories.

Query-Code :

```
/*-----------------------------------Query 10 Which state is mostly having tweets about type of phone ----------------------------------*/
case "10" =>
  val iphoneRDD = sqlContext.sql( sqlText = """ SELECT 'iphone' as phoneType, user.location as loc from tweets where text LIKE '%#iphone%' """)
  val iphoneXRDD = sqlContext.sql( sqlText = """ SELECT 'iphoneX' as phoneType, user.location as loc from tweets where text LIKE '%#iphoneX%' """)
  val iPhoneXsRDD = sqlContext.sql( sqlText = """ SELECT 'iPhoneXs' as phoneType, user.location as loc from tweets where text LIKE '%#iPhoneXs%' """)
  val watchRDD = sqlContext.sql( sqlText = """ SELECT 'watch' as phoneType, user.location as loc from tweets where text LIKE '%#watch%' """)
  //val breakRDD =  sqlContext.sql("""" SELECT 'Breakfast' as Mealtype, SUBSTRING(created_at,12,2) as hour, user.location as loc from dfs where text
  //val brunchRDD =  sqlContext.sql("""" SELECT 'Brunch' as Mealtype, SUBSTRING(created_at,12,2) as hour, user.location as loc from dfs where text LI
  val sql2RDD = iphoneRDD.union(iphoneXRDD).union(iPhoneXsRDD).union(watchRDD)
  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  '%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%'
```

Output :





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

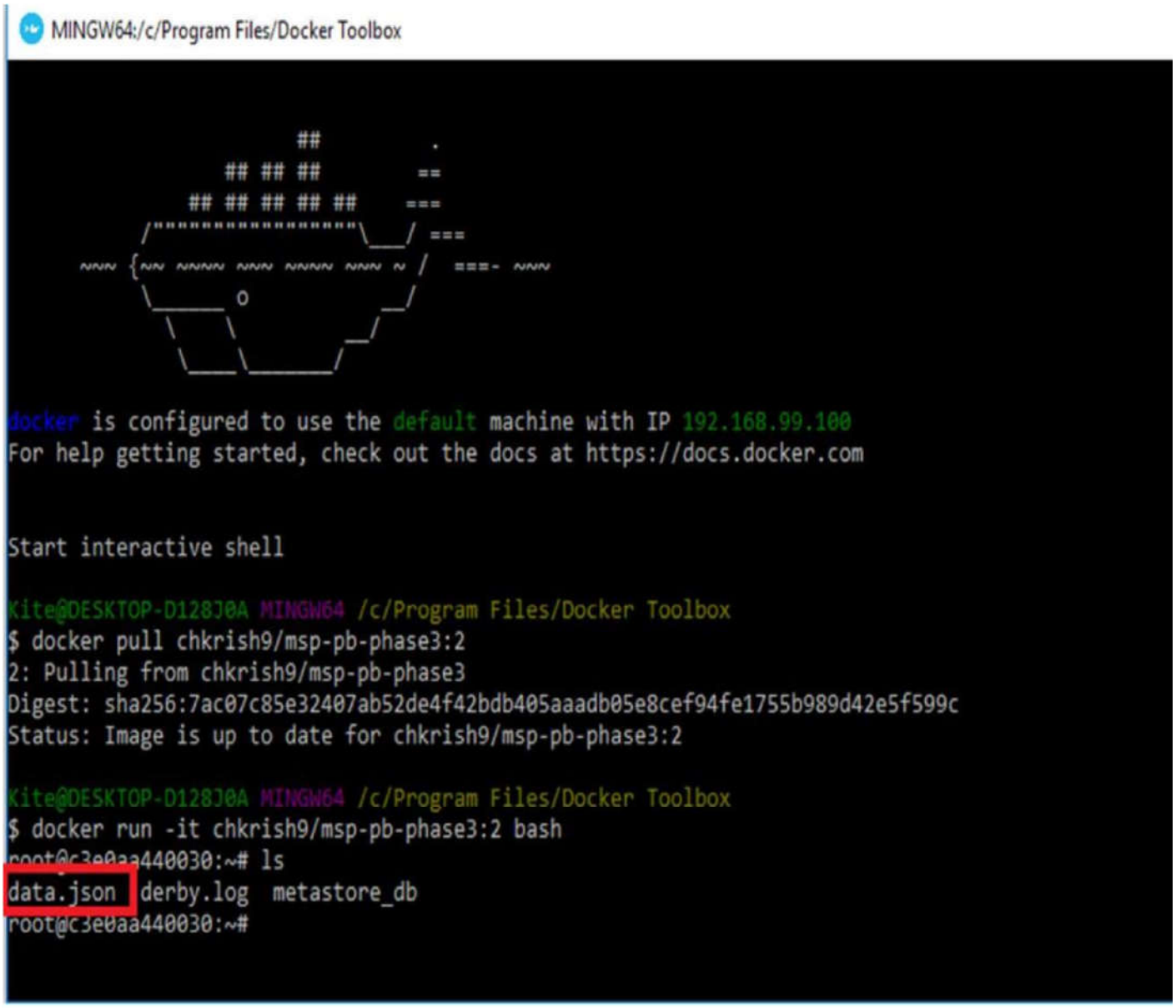
docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at https://docs.docker.com

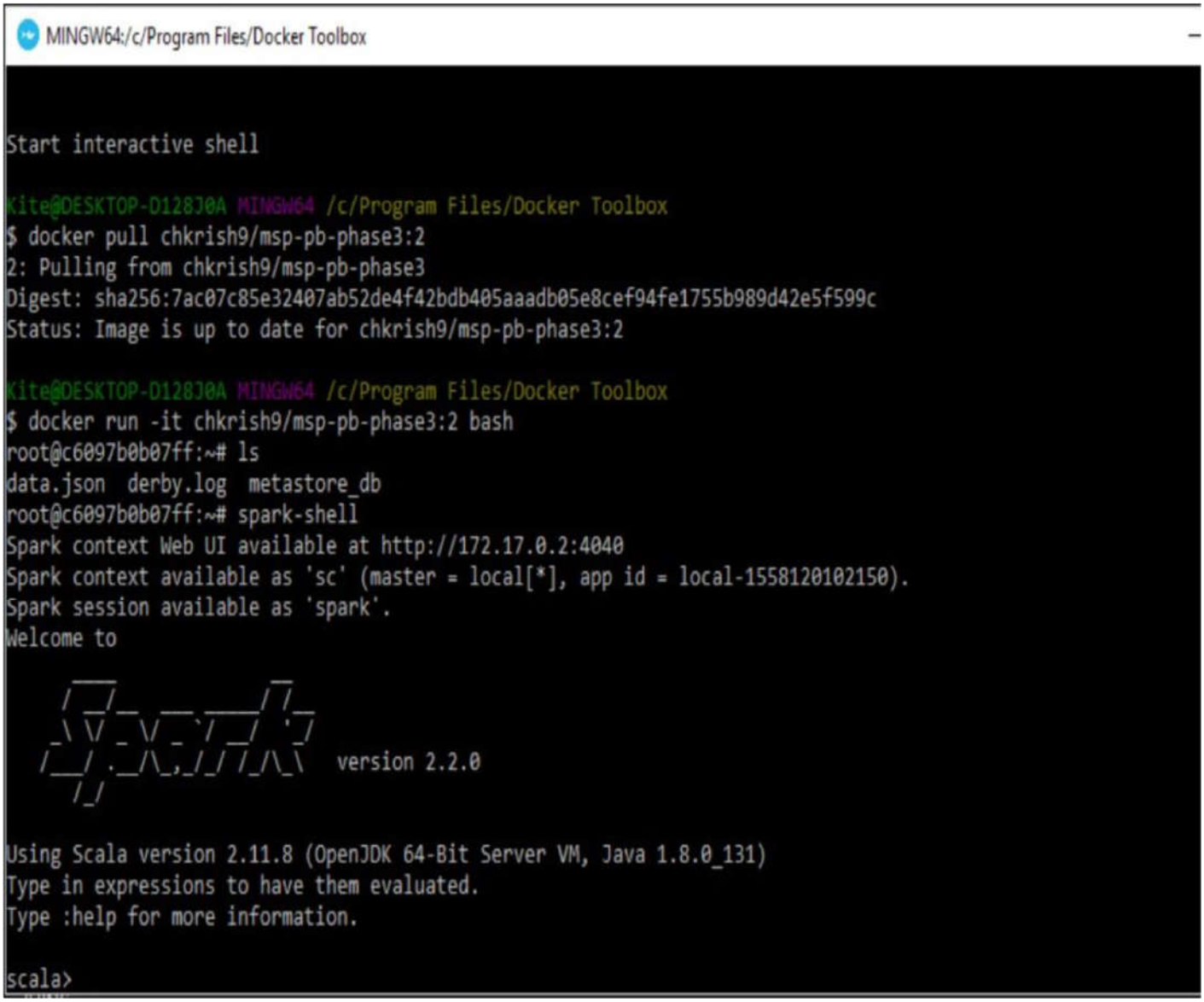
Start interactive shell

```
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

```
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
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>
```

PROJECT MANAGEMENT :

WORK COMPLETED :

Pravalhika Kampally :

Collected Data and executed first 3 quires.

Joshmitha Tammareddy :

Executes 4 to 7 Quires and worked for Docker

Rupesh Sai Ram Doddala :

Executed 7 to 10 quires and worked for visualization.