<u>CS5540 - Principles of Big Data Management</u>

A system to store, analyze, and visualize Twitter's tweets

Project Phase 2 Report

Team Details

Ruthvic Punyamurtula – RPMGZ

Navya Ramya Sirisha Pillala – NP7YD

Bhanu Sudheer – BVHKF

Submitted contents

1. **Phase 2 report**
2. **Apache Zeppelin Json file**
3. **SparkTweets.scala – Scala code file**
4. **Intellij Testing output files**
5. **Zeppelin query and visualization Screenshots**

Goal

Design and implement ideas using Apache Spark

- To store the tweets in Spark SQL
- Write queries to explore and understand the data – at least 10 queries
- Develop interesting visualizations (e.g., pie chart, heat map, bar graphs)

Phase 2

1. **Tweet collection:**

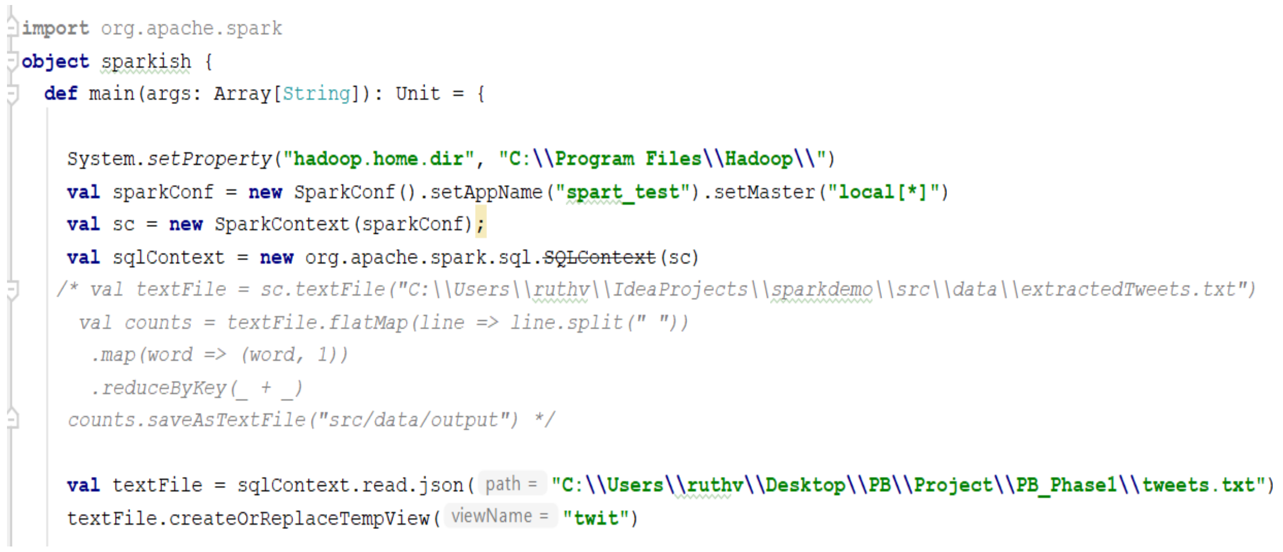
    - For phase 2, we have collected new set of tweets with key words related to “ipl”, “avengers”, “Syria”, “robotics” – in order to bring insights on the latest topics.
    - Also performed word count on the new tweets

2. **IntelliJ - SPARK:**

    - We have installed IntelliJ to simulate the SPARK environment in windows system. Then stored the collected tweet file into Spark localhost.

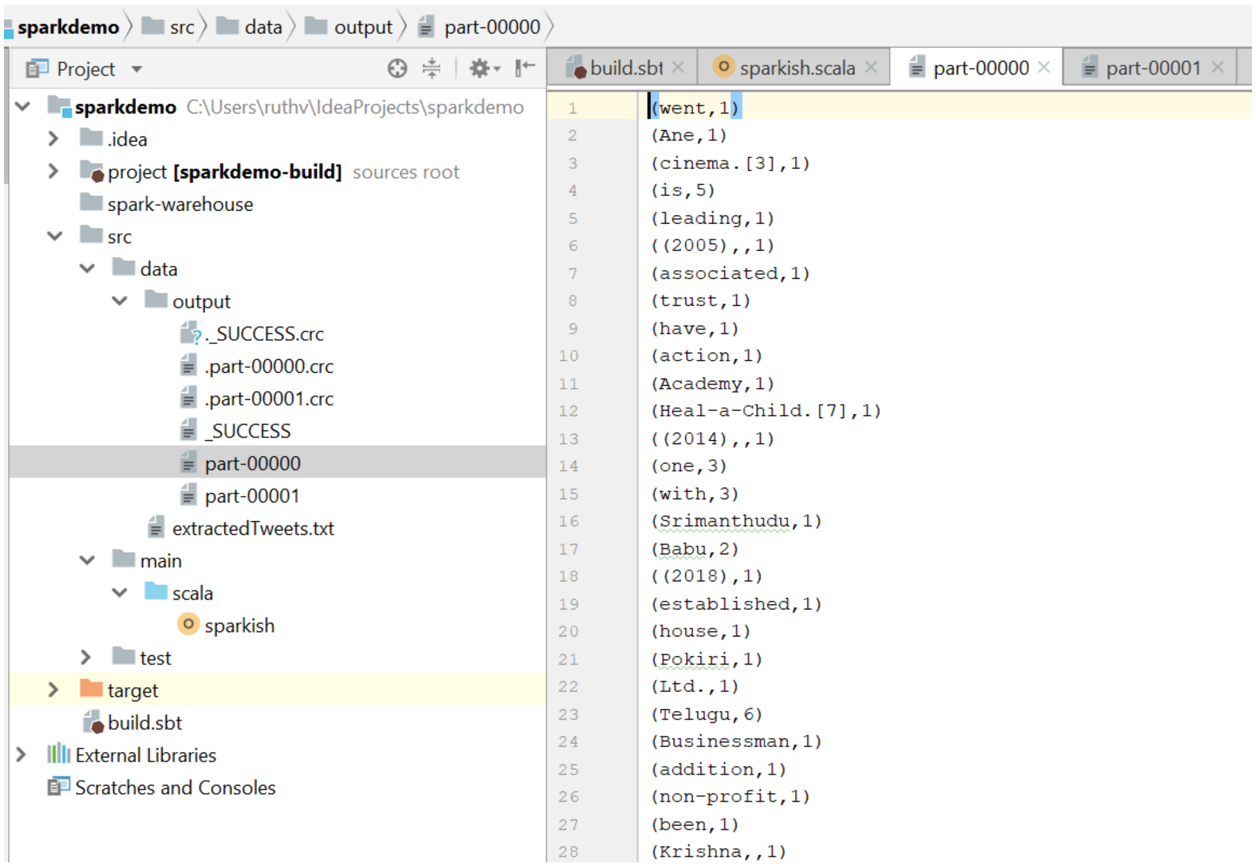
    - We have used Scala version = 2.11.8 and then wrote the SQL queries to retrieve the results

Code to load tweet file into spark



```
import org.apache.spark
object sparkish {
  def main(args: Array[String]): Unit = {

    System.setProperty("hadoop.home.dir", "C:\\Program Files\\Hadoop\\")
    val sparkConf = new SparkConf().setAppName("spart_test").setMaster("local[*]")
    val sc = new SparkContext(sparkConf);
    val sqlContext = new org.apache.spark.sql.~~SQLContext~~(sc)
   /* val textFile = sc.textFile("C:\\Users\\ruthv\\IdeaProjects\\sparkdemo\\src\\data\\extractedTweets.txt")
     val counts = textFile.flatMap(line => line.split(" "))
      .map(word => (word, 1))
      .reduceByKey(_ + _)
    counts.saveAsTextFile("src/data/output") */

    val textFile = sqlContext.read.json( path = "C:\\Users\\ruthv\\Desktop\\PB\\Project\\PB_Phase1\\tweets.txt")
    textFile.createOrReplaceTempView( viewName = "twit")
```

Result of word count

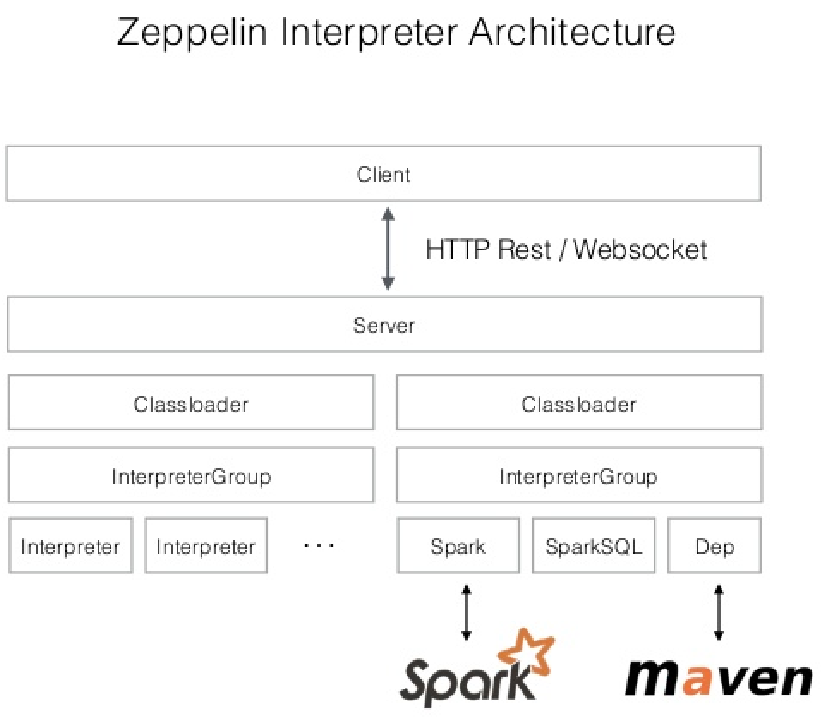


```
(went,1)
(Ane,1)
(cinema.[3],1)
(is,5)
(leading,1)
((2005),,1)
(associated,1)
(trust,1)
(have,1)
(action,1)
(Academy,1)
(Heal-a-Child.[7],1)
((2014),,1)
(one,3)
(with,3)
(Srimanthudu,1)
(Babu,2)
((2018),1)
(established,1)
(house,1)
(Pokiri,1)
(Ltd.,1)
(Telugu,6)
(Businessman,1)
(addition,1)
(non-profit,1)
(been,1)
(Krishna,,1)
```

3. Apache – Zeppelin

We have used Apache – Zeppelin to show the live demo of the SQL queries and also visualize the data in the real time

Apache – Zeppelin Architecture



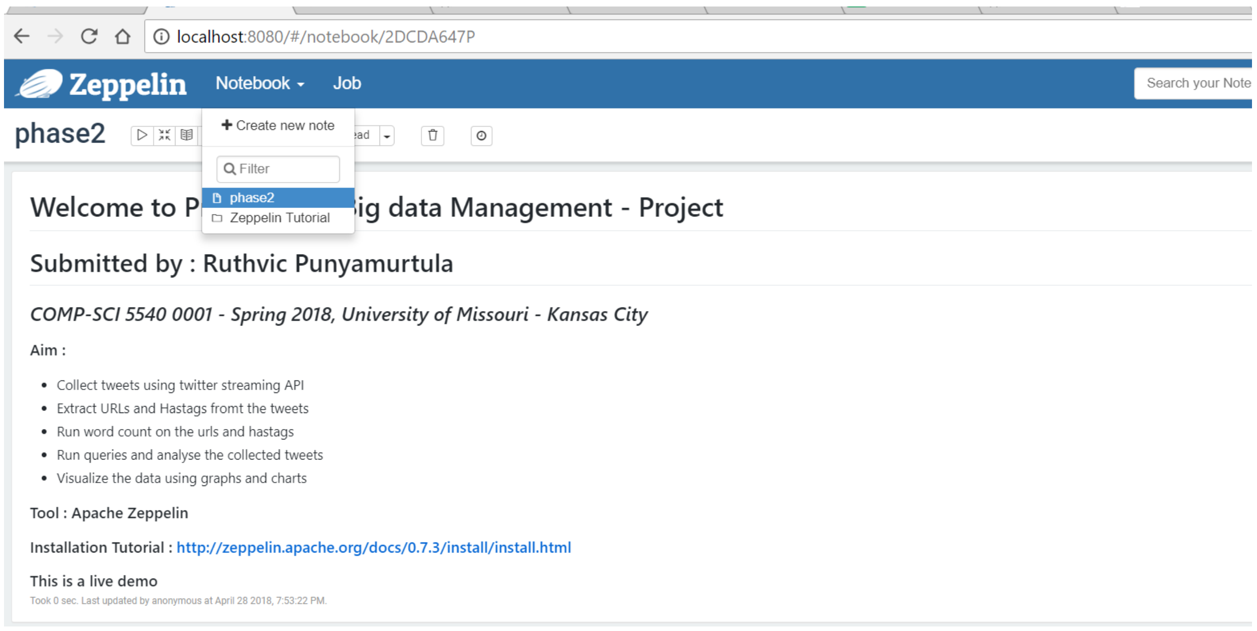
Ref : https://www.slideshare.net/KSLUG/kslug-zeppelin

Apache – Zeppelin Installation

Download the binary package with “all” interpreters and extract it in local.
Go to bin -> Zeppelin.cmd in windows command prompt, to start the Zeppelin server.

Once the server is started, goto http://localhost:8080 and click on the “Notebook” drop down and select -> create a new note

In the top section, We have added introduction and brief note using “%md” - the markdown interpreter to insert the text



Welcome to P… …ig data Management - Project

Submitted by : Ruthvic Punyamurtula

COMP-SCI 5540 0001 - Spring 2018, University of Missouri - Kansas City

Aim :

- Collect tweets using twitter streaming API
- Extract URLs and Hastags fromt the tweets
- Run word count on the urls and hastags
- Run queries and analyse the collected tweets
- Visualize the data using graphs and charts

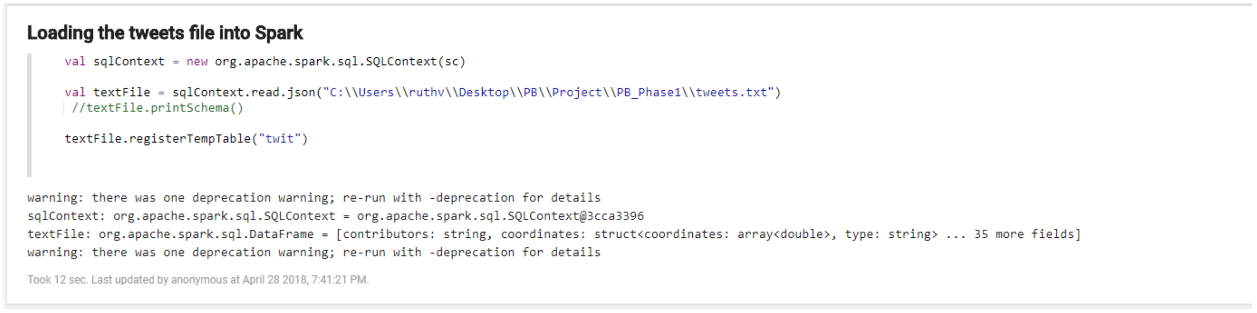
Tool : Apache Zeppelin

Installation Tutorial : http://zeppelin.apache.org/docs/0.7.3/install/install.html

This is a live demo

Took 0 sec. Last updated by anonymous at April 28 2018, 7:53:22 PM.

In the second section, we load the data into Spark and run the queries to obtain the results. The below screenshot explains this.



Loading the tweets file into Spark

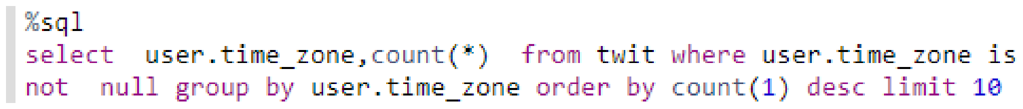
```
val sqlContext = new org.apache.spark.sql.SQLContext(sc)

val textFile = sqlContext.read.json("C:\\Users\\ruthv\\Desktop\\PB\\Project\\PB_Phase1\\tweets.txt")
 //textFile.printSchema()

textFile.registerTempTable("twit")
```

```
warning: there was one deprecation warning; re-run with -deprecation for details
sqlContext: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@3cca3396
textFile: org.apache.spark.sql.DataFrame = [contributors: string, coordinates: struct<coordinates: array<double>, type: string> ... 35 more fields]
warning: there was one deprecation warning; re-run with -deprecation for details
```

Took 12 sec. Last updated by anonymous at April 28 2018, 7:41:21 PM.

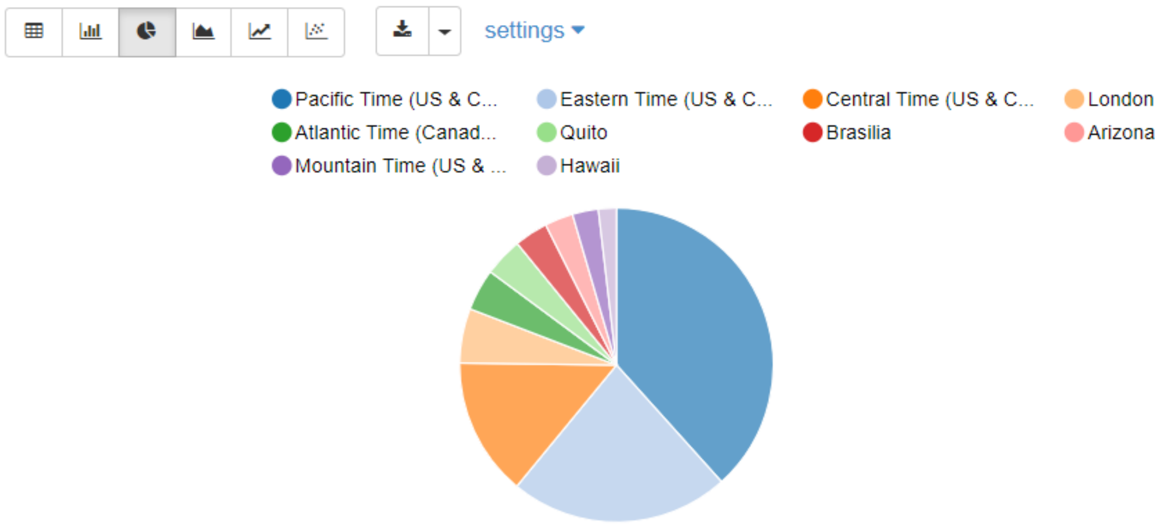
Now as the data is loaded and RDD is created for this, we run the queries

Query 1 – Tweets based on time zone



```
%sql
select  user.time_zone,count(*)  from twit where user.time_zone is
not  null group by user.time_zone order by count(1) desc limit 10
```

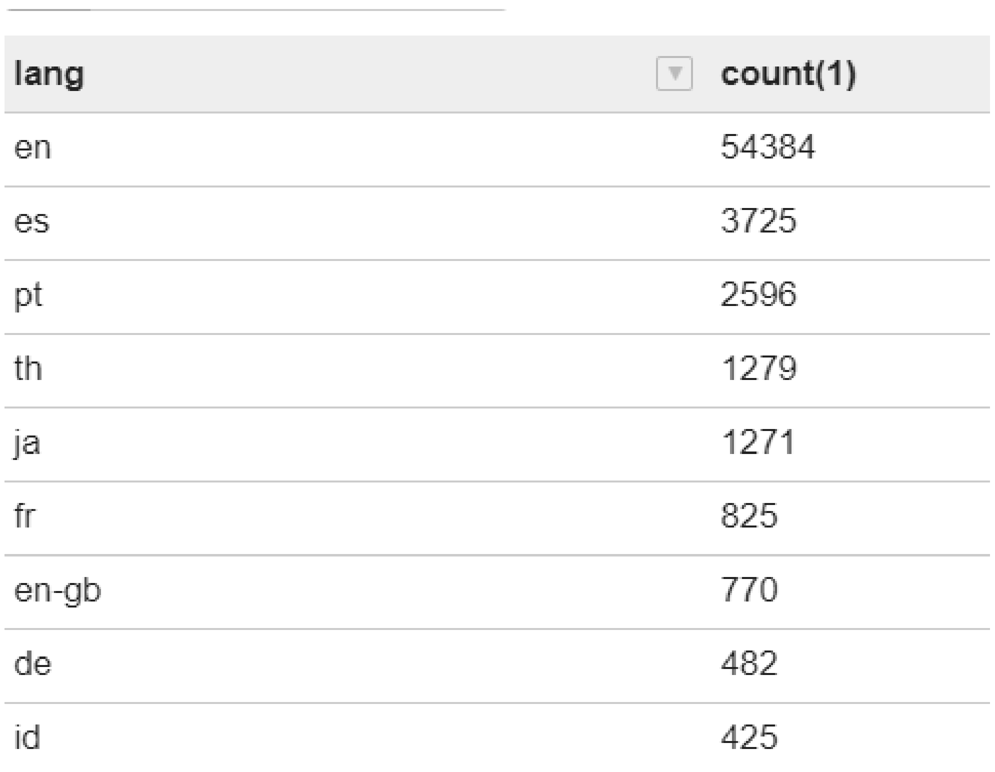
Zeppelin provides us with real time data visualization in the form of table/bar graph/pie chart/area chart/line chart/scatter chart



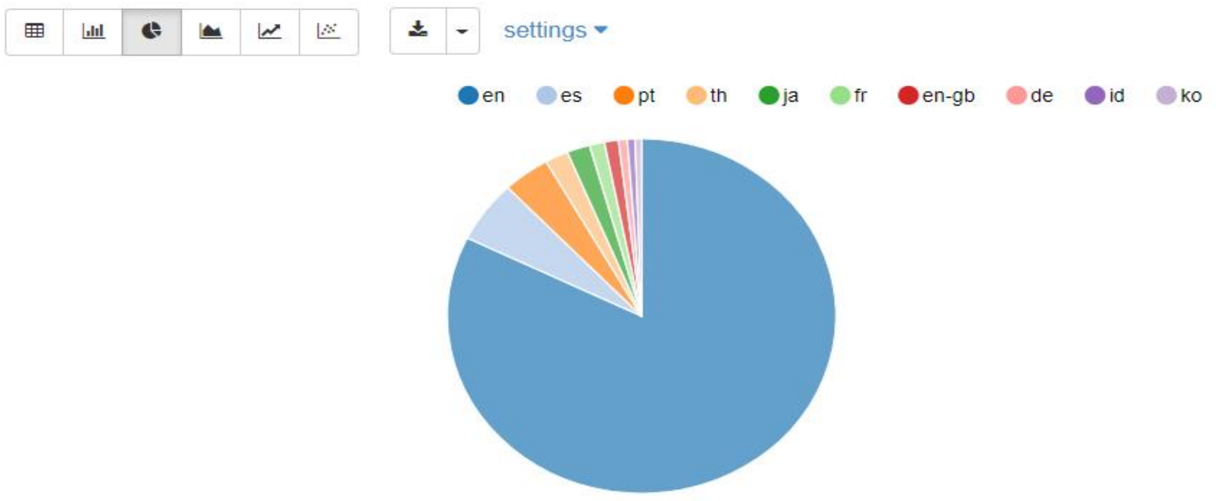
Query 2 – Tweets count based on the user input language

```
%sql
select user.lang,count(*)  from twit where user.lang is not null  group by user.lang order by
count(1) desc limit 10
```

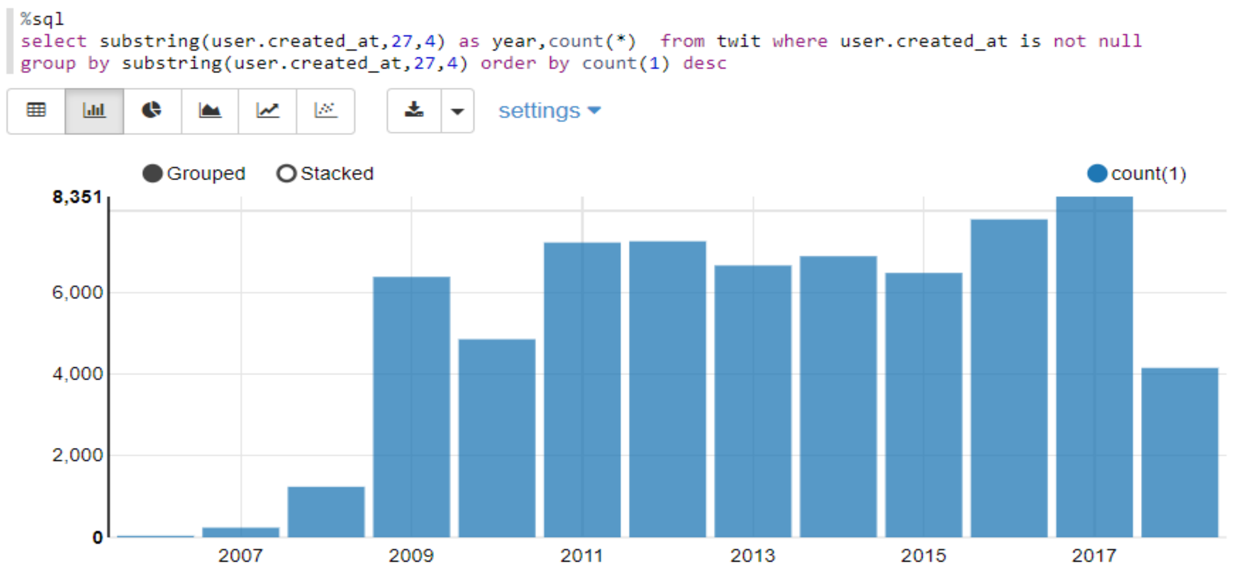
Data visualization for this is



lang	count(1)
en	54384
es	3725
pt	2596
th	1279
ja	1271
fr	825
en-gb	770
de	482
id	425



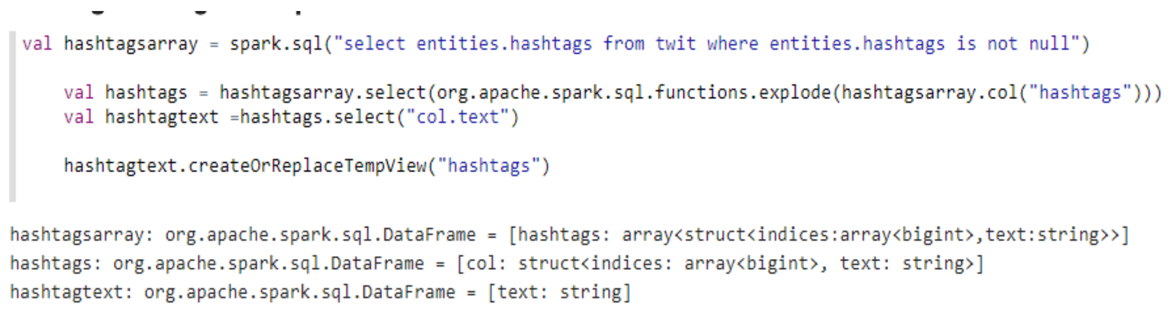
Query 3 – Users created per year



```
%sql
select substring(user.created_at,27,4) as year,count(*)  from twit where user.created_at is not null
group by substring(user.created_at,27,4) order by count(1) desc
```

Query 4 – Top Hashtags

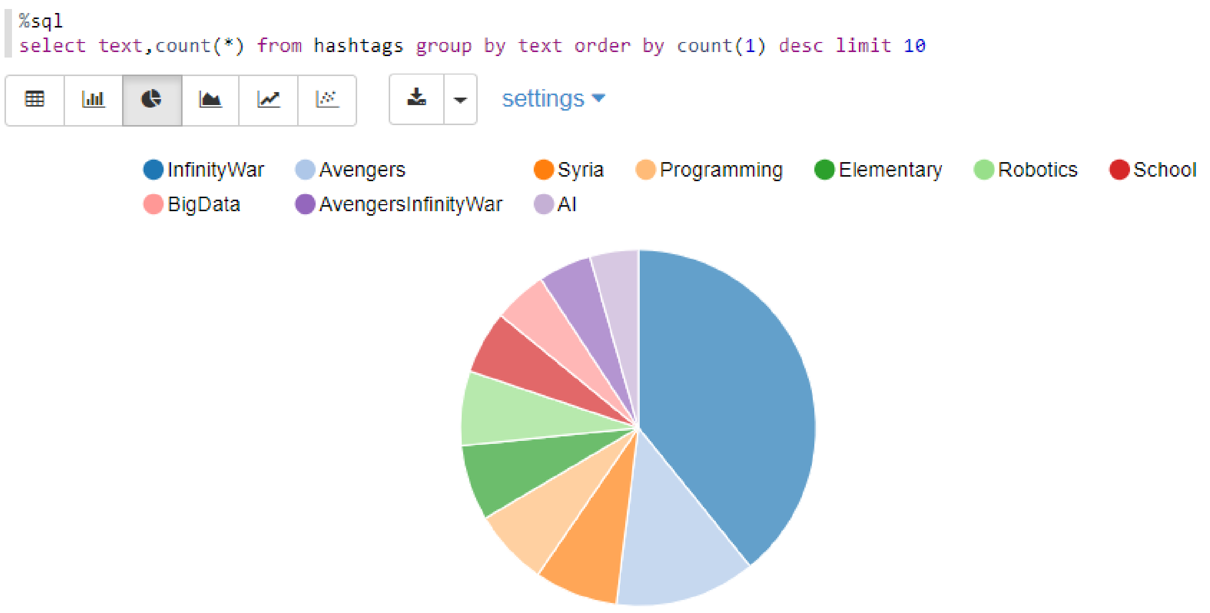
We separate the hashtags from tweets and load them into separate data frame and run the query on it



```
val hashtagsarray = spark.sql("select entities.hashtags from twit where entities.hashtags is not null")

    val hashtags = hashtagsarray.select(org.apache.spark.sql.functions.explode(hashtagsarray.col("hashtags")))
    val hashtagtext =hashtags.select("col.text")

    hashtagtext.createOrReplaceTempView("hashtags")
```

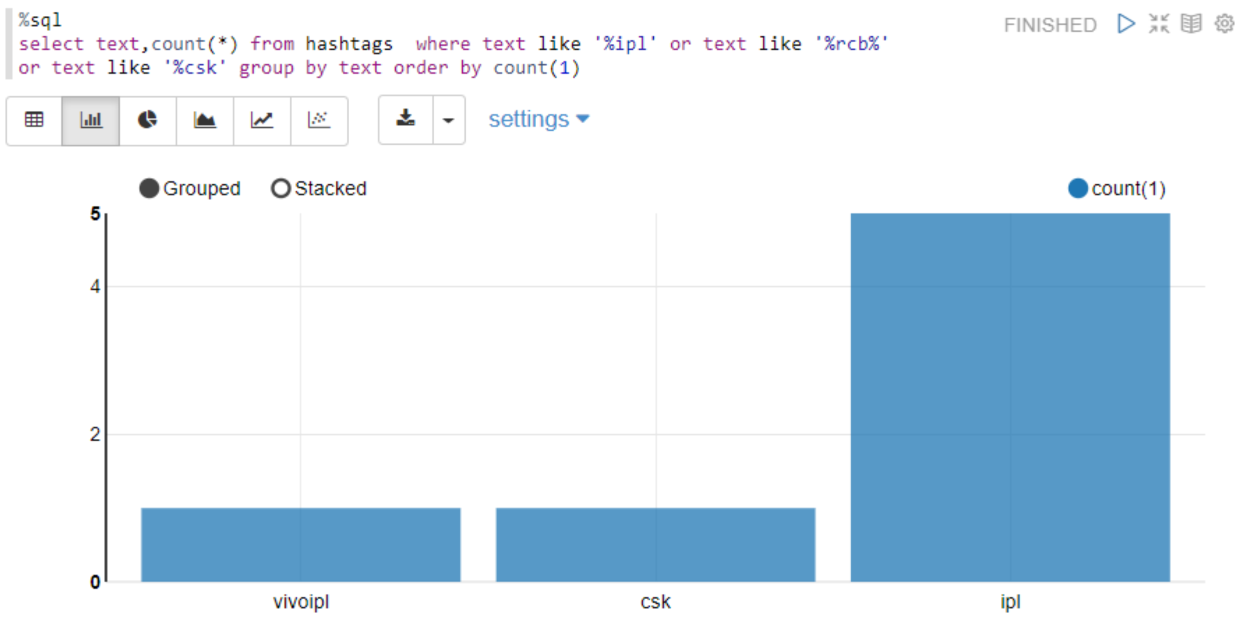
```
hashtagsarray: org.apache.spark.sql.DataFrame = [hashtags: array<struct<indices:array<bigint>,text:string>>]
hashtags: org.apache.spark.sql.DataFrame = [col: struct<indices: array<bigint>, text: string>]
hashtagtext: org.apache.spark.sql.DataFrame = [text: string]
```

Query with the result and visualization



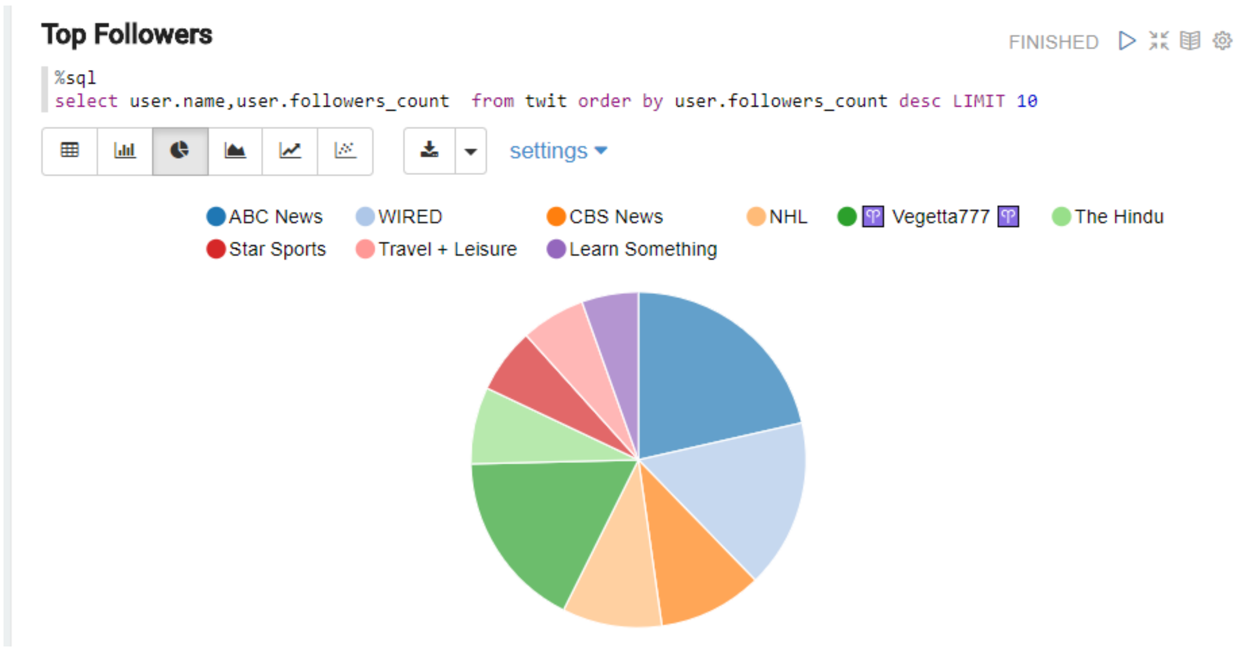
```
%sql
select text,count(*) from hashtags group by text order by count(1) desc limit 10
```

Query 5 – Filtering IPL related tweets



```
%sql
select text,count(*) from hashtags  where text like '%ipl' or text like '%rcb%'
or text like '%csk' group by text order by count(1)
```

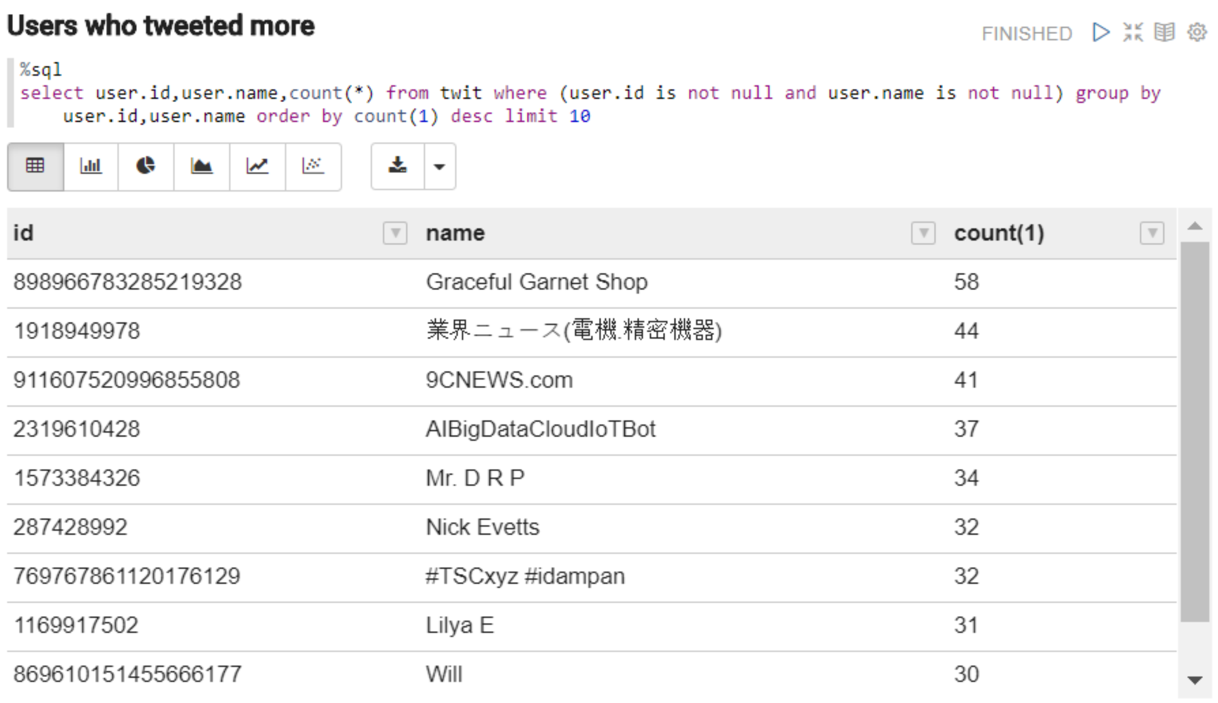
Query 6 – Top followers



Top Followers

```
%sql
select user.name,user.followers_count  from twit order by user.followers_count desc LIMIT 10
```

Query 7 – Users who tweeted more

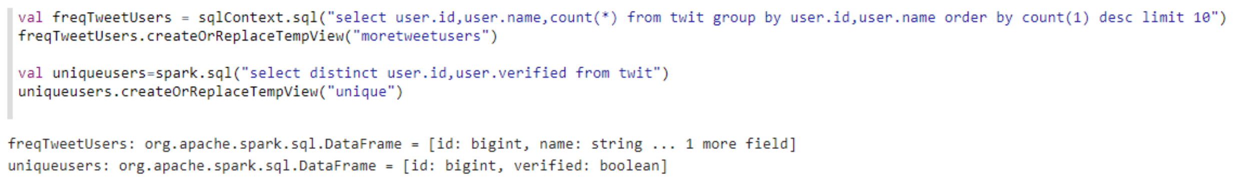


Users who tweeted more

```
%sql
select user.id,user.name,count(*) from twit where (user.id is not null and user.name is not null) group by
    user.id,user.name order by count(1) desc limit 10
```

id	name	count(1)
898966783285219328	Graceful Garnet Shop	58
1918949978	業界ニュース(電機.精密機器)	44
911607520996855808	9CNEWS.com	41
2319610428	AIBigDataCloudIoTBot	37
1573384326	Mr. D R P	34
287428992	Nick Evetts	32
769767861120176129	#TSCxyz #idampan	32
1169917502	Lilya E	31
869610151455666177	Will	30

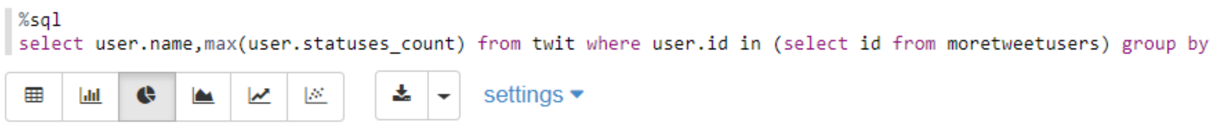
Data frames to find users with more tweets and also to find verified vs unverified users



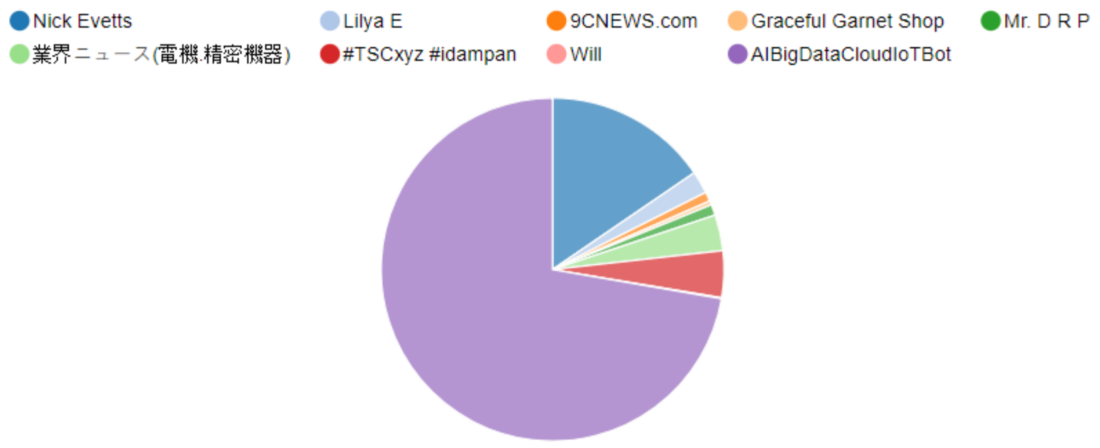
```
val freqTweetUsers = sqlContext.sql("select user.id,user.name,count(*) from twit group by user.id,user.name order by count(1) desc limit 10")
freqTweetUsers.createOrReplaceTempView("moretweetusers")

val uniqueusers=spark.sql("select distinct user.id,user.verified from twit")
uniqueusers.createOrReplaceTempView("unique")
```

```
freqTweetUsers: org.apache.spark.sql.DataFrame = [id: bigint, name: string ... 1 more field]
uniqueusers: org.apache.spark.sql.DataFrame = [id: bigint, verified: boolean]
```

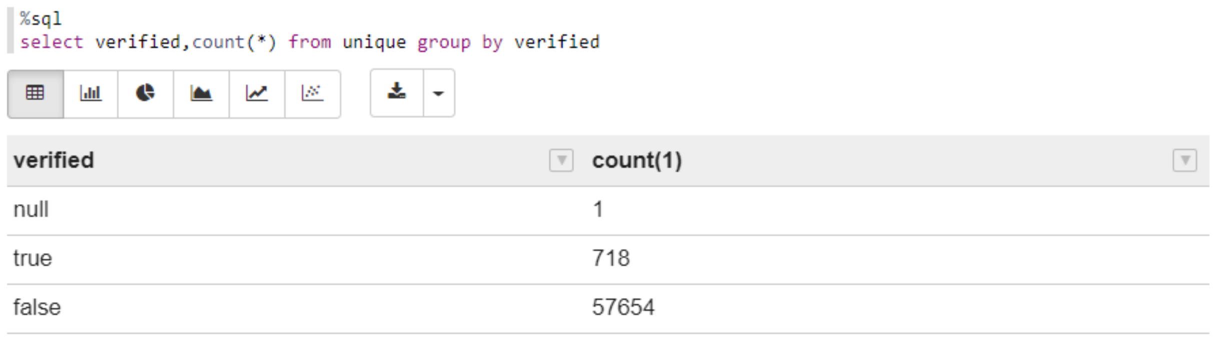
Query 8 – Max status count



```
%sql
select user.name,max(user.statuses_count) from twit where user.id in (select id from moretweetusers) group by
```



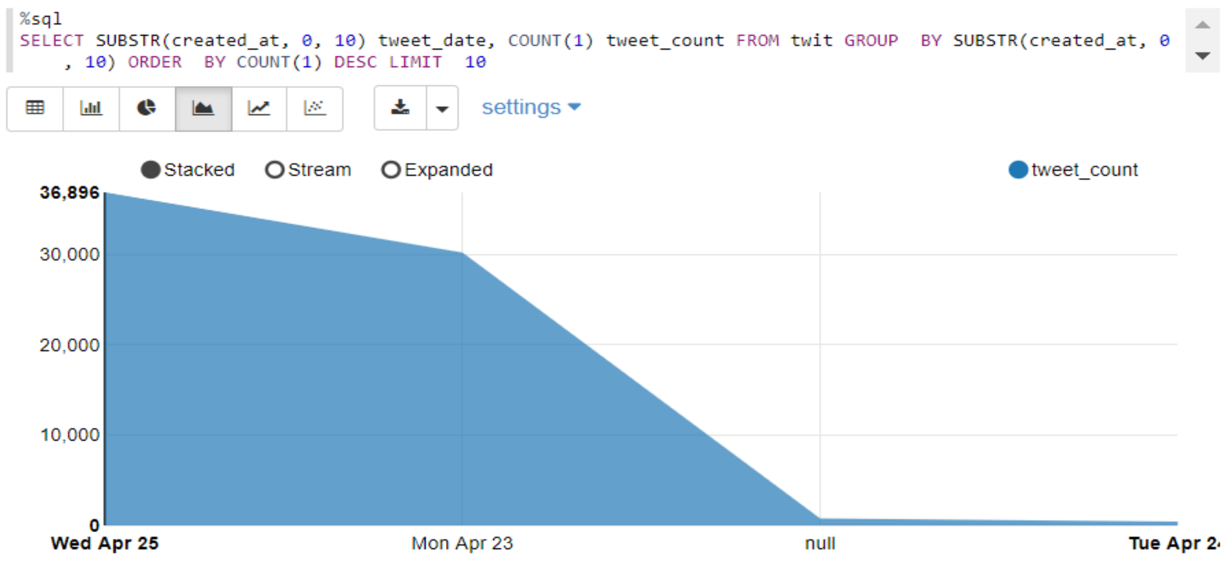
Query 9 – Verified vs Unverified users



```
%sql
select verified,count(*) from unique group by verified
```

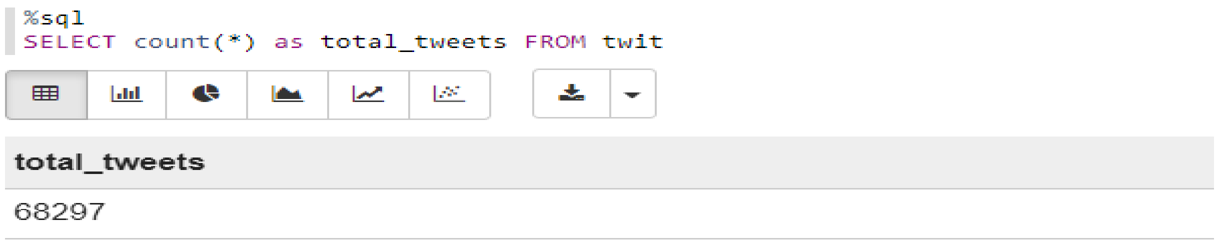
verified	count(1)
null	1
true	718
false	57654

Query 10 – Days and date with most tweets



```
%sql
SELECT SUBSTR(created_at, 0, 10) tweet_date, COUNT(1) tweet_count FROM twit GROUP  BY SUBSTR(created_at, 0
    , 10) ORDER  BY COUNT(1) DESC LIMIT  10
```

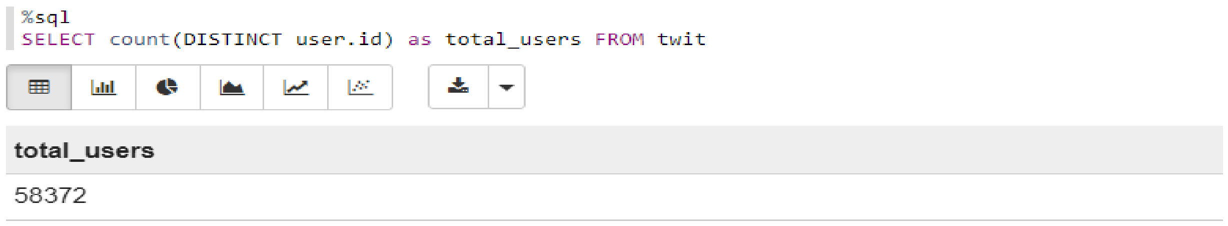
Query 11 – Total tweets



```
%sql
SELECT count(*) as total_tweets FROM twit
```

total_tweets
68297

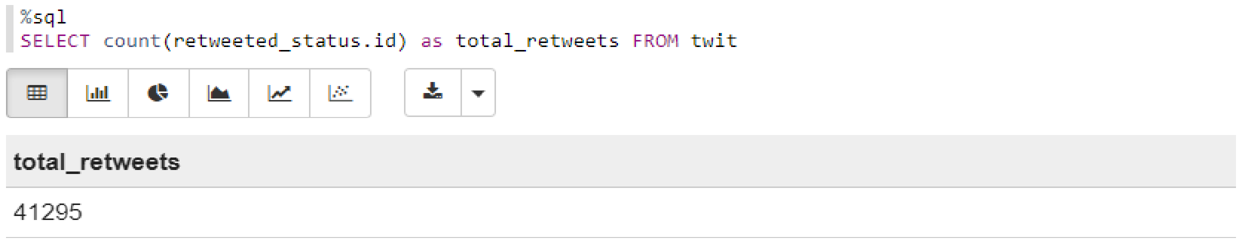
Query 12 – Total users



```
%sql
SELECT count(DISTINCT user.id) as total_users FROM twit
```

total_users
58372

Query 13 – Total Retweets



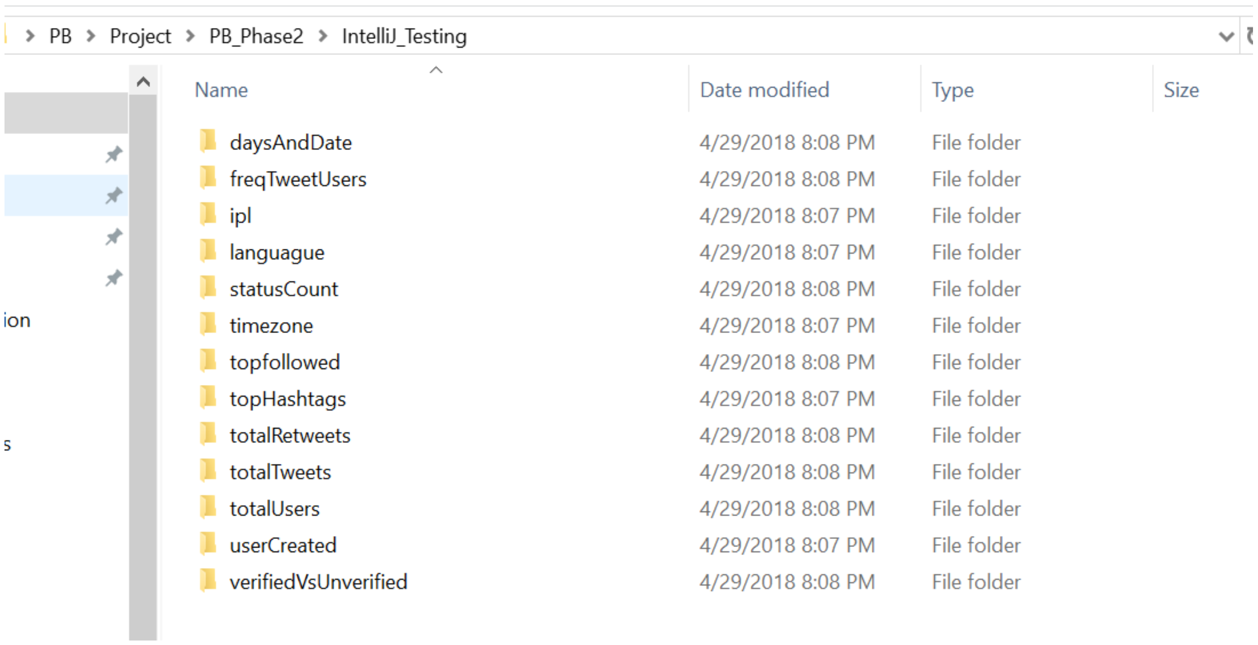
```
%sql
SELECT count(retweeted_status.id) as total_retweets FROM twit
```

total_retweets
41295

4. Testing

We have tested the result of the queries in IntelliJ – Spark environment and verified the results obtained with that of the Zeppelin results.

The below folders contain the output CSV for the queries



Name	Date modified	Type	Size
daysAndDate	4/29/2018 8:08 PM	File folder	
freqTweetUsers	4/29/2018 8:08 PM	File folder	
ipl	4/29/2018 8:07 PM	File folder	
languague	4/29/2018 8:07 PM	File folder	
statusCount	4/29/2018 8:08 PM	File folder	
timezone	4/29/2018 8:07 PM	File folder	
topfollowed	4/29/2018 8:08 PM	File folder	
topHashtags	4/29/2018 8:07 PM	File folder	
totalRetweets	4/29/2018 8:08 PM	File folder	
totalTweets	4/29/2018 8:08 PM	File folder	
totalUsers	4/29/2018 8:08 PM	File folder	
userCreated	4/29/2018 8:07 PM	File folder	
verifiedVsUnverified	4/29/2018 8:08 PM	File folder	

5. Learning Outcome

- Twitter streaming and handling huge data in Hadoop and spark
- Running queries to retrieve the data from Spark Data frames
- Data visualization to interpret the results using Apache – Zeppelin
- Explored other visualization tools – google charts, high charts

6. Challenges

- Initial Zeppelin setup was tough
- Not many tutorials to understand the hands-on use of Zeppelin

7. References

- https://www.coursera.org/learn/open-source-tools-for-data-science/lecture/gtChC/zeppelin-for-scala
- https://scalegrid.io/blog/data-visualization-using-apache-zeppelin/
- https://zeppelin.apache.org/docs/0.5.5-incubating/tutorial/tutorial.html
- http://zeppelin.apache.org/docs/0.7.3/install/install.html#downloading-binary-package
- https://www.youtube.com/watch?v=CfhYFqNyjGc
- https://sourceforge.net/p/zeppelin/wiki/markdown_syntax/#section-1

Thank You