UNIVERSITY OF MISSOURI-KANSAS CITY

Principles Of Big Data

Project: Twitter Data Analysis

Phase 2

Professor: Dr. Praveen Rao

Team Members:

Name	Email	Student ID
Yamini Reddy Dontireddy	yddpw@mail.umkc.edu	16293335
Manideep Nadimpelli	mnpgc@mail.umkc.edu	16292336
Bhavya sri Sirasanagandla	Bs9r8@mail.umkc.edu	16292335

Version History

Version number	Date Modified	Author	Reviewer	Comments
01	9/15/2019	Yamini/Manideep/Bhavya	PB - Grader	Initial copy – Phase 1 Submitted
02	11/15/2019	Yamini/Manideep/Bhavya	PB - Grader	Phase – 2 Report

Project Scope

Scope of this project is to leverage the latest big data technologies such as Hadoop, Spark and APIs to analyze the Twitter data to gain any insights around a topic or trend or a media campaign.

Project is divided into three phases:

- A high-level analysis of specific hashtags and URLs in phase 1.
- Phase 2 requires a meaningful analysis of twitter data to analyze sentiments or real time trends or specific pattern/impact from a set of influencing twitter handles.
- A poster presentation with analytical queries performance metrics/details in Phase 3.

Acronyms

- WHO – World Health Organization
- NCD – Non Communicable Diseases
- AHA – American Heart Association
- CMS – Centre for Medicare & Medicaid Services
- BCBSA – Blue Cross Blue Shield Association
- API - Application Programming Interface
- JSON - JavaScript Object Notation
- URL – Uniform/Unique Resource Locator

Potential Use cases

Apart from typical usage of twitter for customer engagement through key words or searches through about specific product or campaigns; twitter can be leveraged in many ways.

- Twitter Analytics As A Service for brand promotion, customer acquisition
- Twitter Analytics for Diagnostics & Policing
- Twitter based insights such as sentiments, trends, patterns
- Twitter based security products

Scope of the project will cover analysis of Twitter interactions and campaigns from selected private, non-profit and public health institutions like Mayo Clinic, WHO, CMS etc of below metrics for the keywords and hashtags related NCD.

The solution performs web crawling and extracts data from Twitter to provide comprehensive metrics. Natural Language Processing is used to process extracted data and classify multi lingual sentiments into various categories such as Positive, Negative, Neutral and Mixed; which is not in current scope of the project.

The key benefits of using this service are quantitative and qualitative performance measurement of keywords/hashtags and success metrics like engagement ratio to measure the effectiveness any disease management program.

The solution can provide access to historical data through keywords/hashtags across official twitter accounts of selected institutions.

Technologies used

Python, Java, Apache Hadoop, Apache Spark & Tableau/amcharts

Implementation Steps

1. Twitter API keys and access tokens needs to be created in order to collect the tweets. This was done from the twitter's developer account.
2. Post the keys generation, using the Python code, specific tweets are collected and stored in the JSON format .
3. Retrieved twitter data is analysed and came up with 11 queries.

4. Then using pyspark library , created the tweets view in Spark SQL, executed the queries on the view and retrieved the output in JSON format.
5. Finally, with the retrieved output, using amcharts all the graphs are visualized and understood the insights .

Visualizations:

The output of the queries/insights are visualized differently using am4charts library.

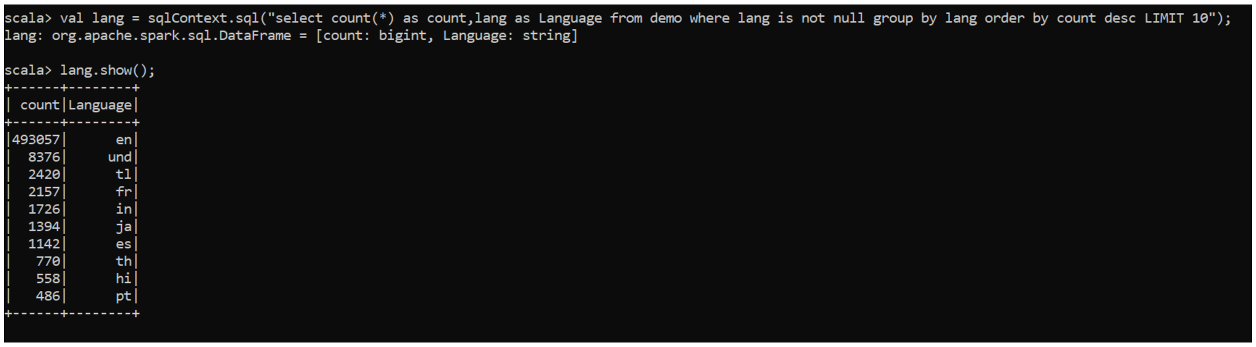
<u>Queries</u>

<u>General Metrics</u>

Social Media Platform	Sample Metrics	Description
Twitter	Languages	List of Languages appeared on the retrieved twitter data.
	Hashtags	Most popular hashtags from the retrieved twitter data.
	Locations	Coordinate's of various locations of the tweeted users in a Map view .
	Data sensitivity levels	Percentage of sensitive tweets and non-sensitive tweets.
	Devices	Classification of devices used by different account holders for tweeting.
	Geographical Distribution	Distribution of account holders across countries.

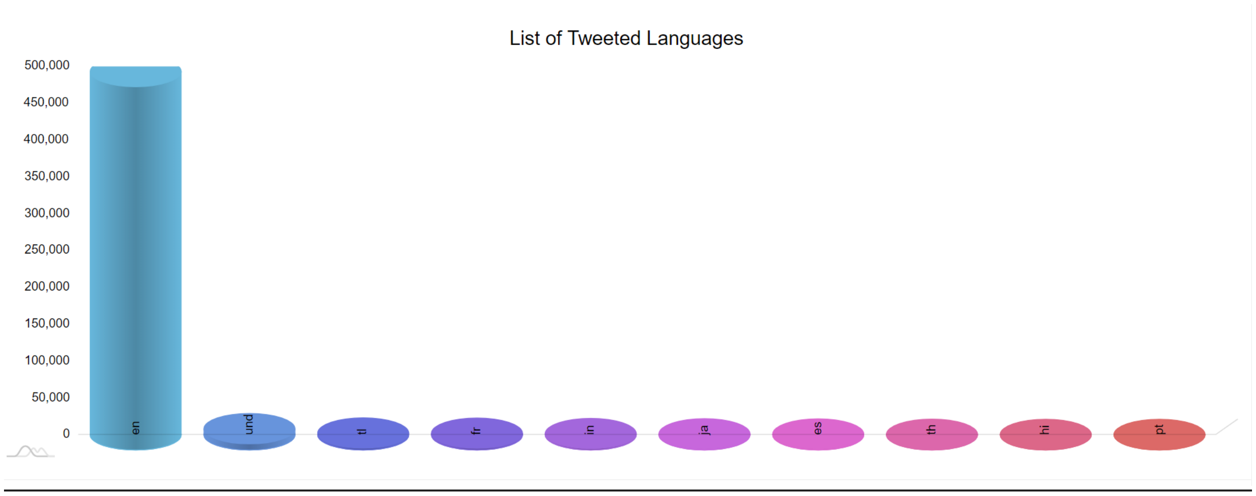
1.List of Languages appeared on the retrieved twitter data.

<u>Query and Output:</u>



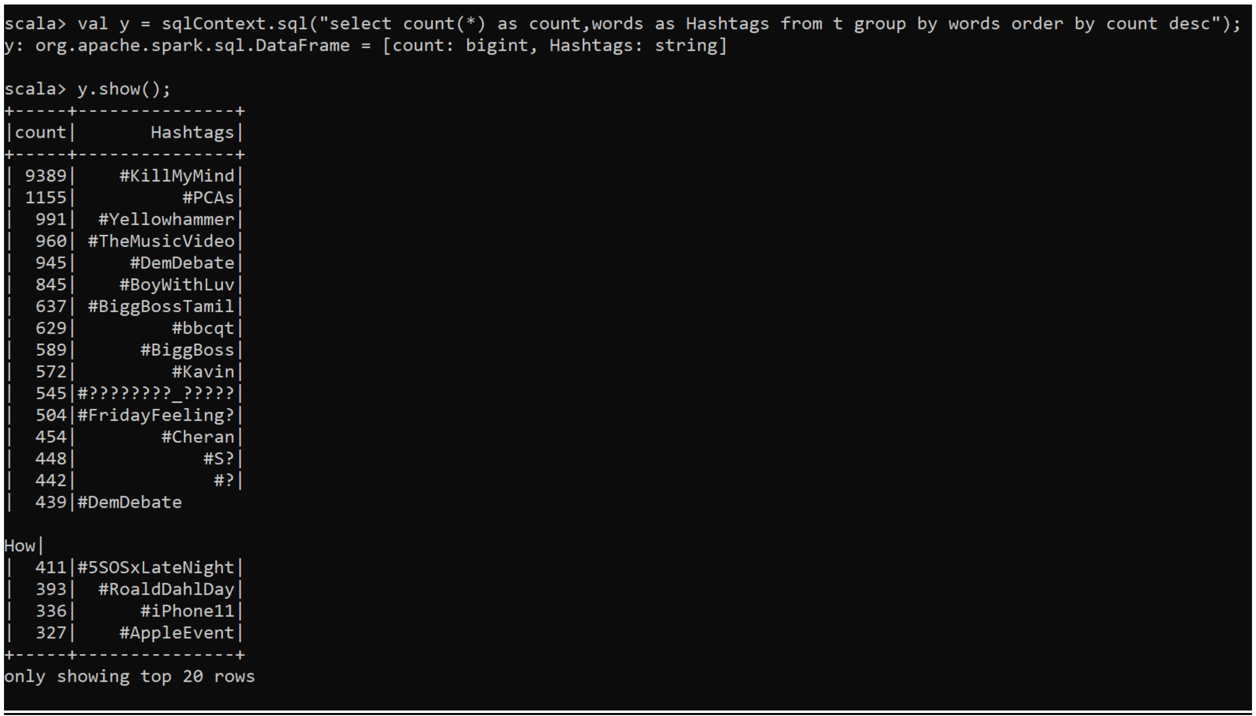
```
scala> val lang = sqlContext.sql("select count(*) as count,lang as Language from demo where lang is not null group by lang order by count desc LIMIT 10");
lang: org.apache.spark.sql.DataFrame = [count: bigint, Language: string]

scala> lang.show();
+------+--------+
| count|Language|
+------+--------+
|493057|      en|
|  8376|     und|
|  2420|      tl|
|  2157|      fr|
|  1726|      in|
|  1394|      ja|
|  1142|      es|
|   770|      th|
|   558|      hi|
|   486|      pt|
+------+--------+
```

<u>Visualization:</u>



2. Most popular hashtags from the retrieved twitter data

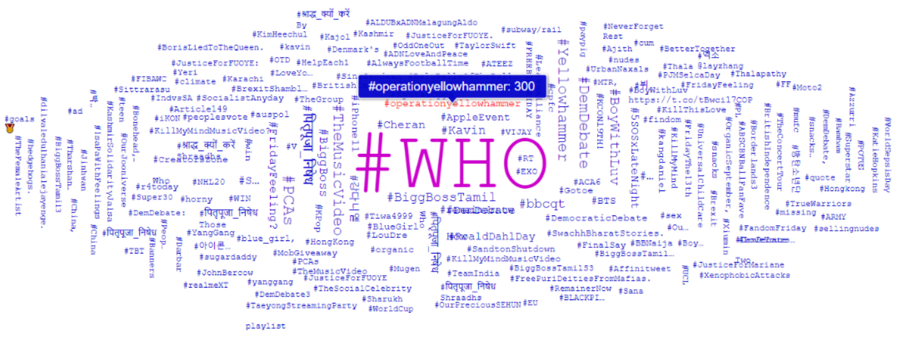
Query and Output:



```
scala> val y = sqlContext.sql("select count(*) as count,words as Hashtags from t group by words order by count desc");
y: org.apache.spark.sql.DataFrame = [count: bigint, Hashtags: string]

scala> y.show();
+-----+---------------+
|count|       Hashtags|
+-----+---------------+
| 9389|     #KillMyMind|
| 1155|          #PCAs|
|  991|   #Yellowhammer|
|  960|  #TheMusicVideo|
|  945|      #DemDebate|
|  845|     #BoyWithLuv|
|  637|  #BiggBossTamil|
|  629|         #bbcqt|
|  589|       #BiggBoss|
|  572|         #Kavin|
|  545|#????????_?????|
|  504|#FridayFeeling?|
|  454|        #Cheran|
|  448|            #S?|
|  442|             #?|
|  439|#DemDebate

How|
|  411|#5SOSxLateNight|
|  393|   #RoaldDahlDay|
|  336|       #iPhone11|
|  327|     #AppleEvent|
+-----+---------------+
only showing top 20 rows
```

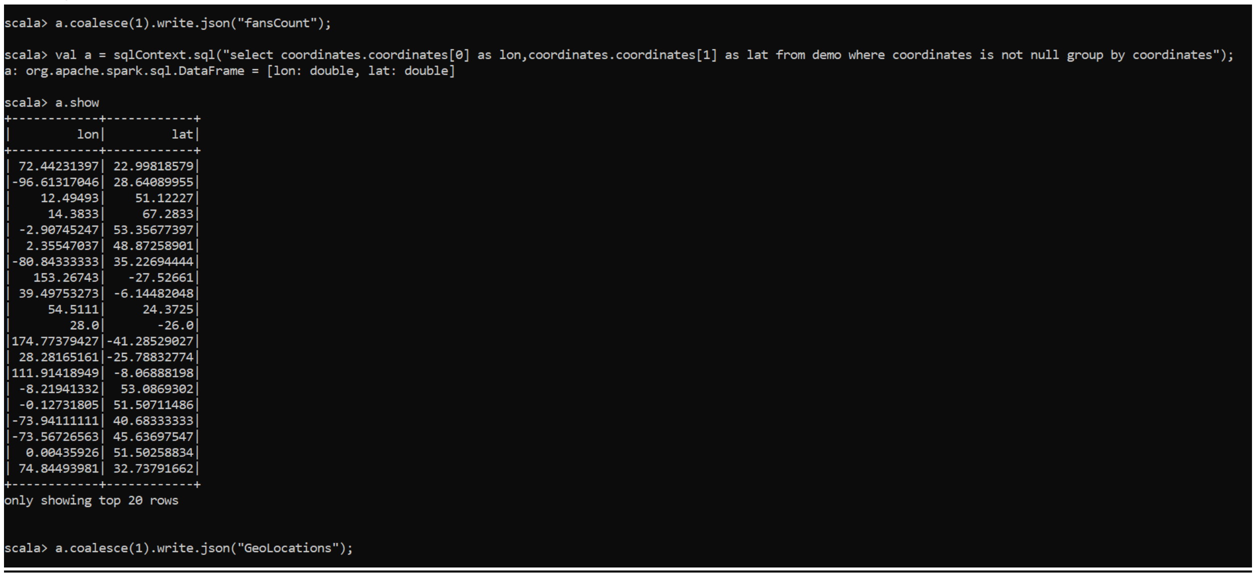
Visualization:

Most Popular Tags

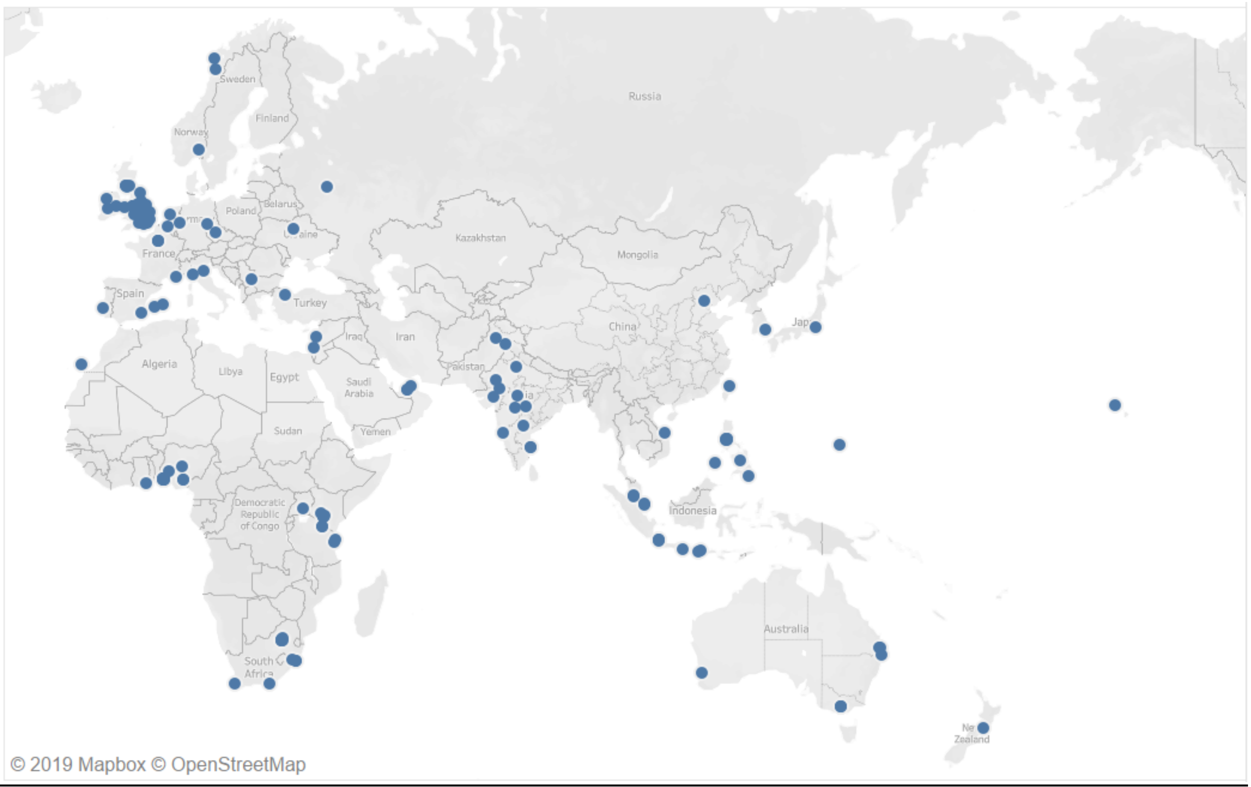


3.Cordinates of various locations of the tweeted users in a Map view .

Query & Output

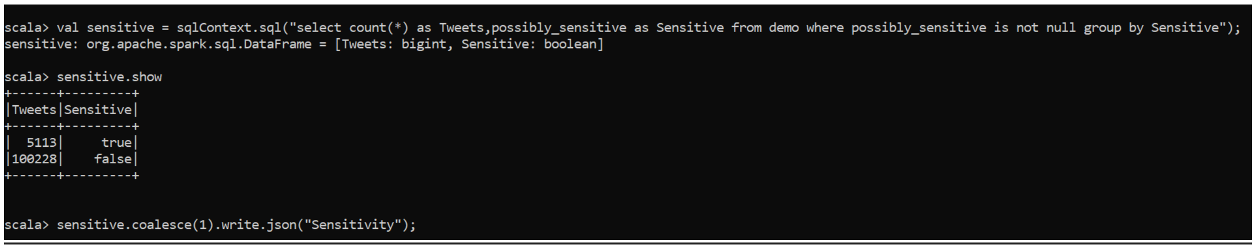

```
scala> a.coalesce(1).write.json("fansCount");

scala> val a = sqlContext.sql("select coordinates.coordinates[0] as lon,coordinates.coordinates[1] as lat from demo where coordinates is not null group by coordinates");
a: org.apache.spark.sql.DataFrame = [lon: double, lat: double]

scala> a.show
+------------+------------+
|         lon|         lat|
+------------+------------+
| 72.44231397| 22.99818579|
|-96.61317046| 28.64089955|
|    12.49493|    51.12227|
|     14.3833|     67.2833|
| -2.90745247| 53.35677397|
|  2.35547037| 48.87258901|
|-80.84333333| 35.22694444|
|   153.26743|   -27.52661|
| 39.49753273| -6.14482048|
|     54.5111|     24.3725|
|        28.0|       -26.0|
|174.77379427|-41.28529027|
| 28.28165161|-25.78832774|
|111.91418949| -8.06888198|
| -8.21941332|  53.0869302|
| -0.12731805| 51.50711486|
|-73.94111111| 40.68333333|
|-73.56726563| 45.63697547|
|  0.00435926| 51.50258834|
| 74.84493981| 32.73791662|
+------------+------------+
only showing top 20 rows


scala> a.coalesce(1).write.json("GeoLocations");
```

Visualization



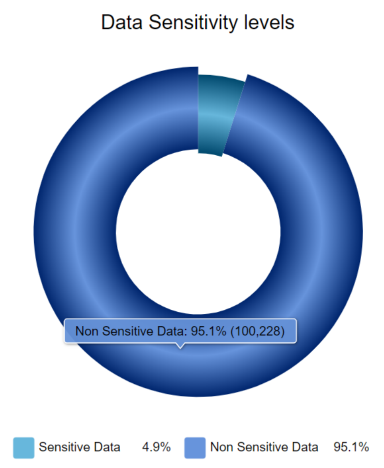
4. Percentage of sensitive tweets and non-sensitive tweets.

Query and Output:



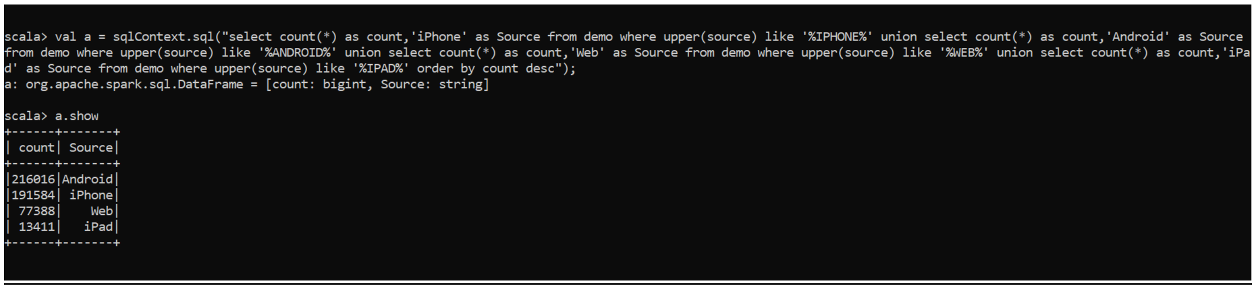
```
scala> val sensitive = sqlContext.sql("select count(*) as Tweets,possibly_sensitive as Sensitive from demo where possibly_sensitive is not null group by Sensitive");
sensitive: org.apache.spark.sql.DataFrame = [Tweets: bigint, Sensitive: boolean]

scala> sensitive.show
+------+---------+
|Tweets|Sensitive|
+------+---------+
|  5113|     true|
|100228|    false|
+------+---------+


scala> sensitive.coalesce(1).write.json("Sensitivity");
```

Visualization:



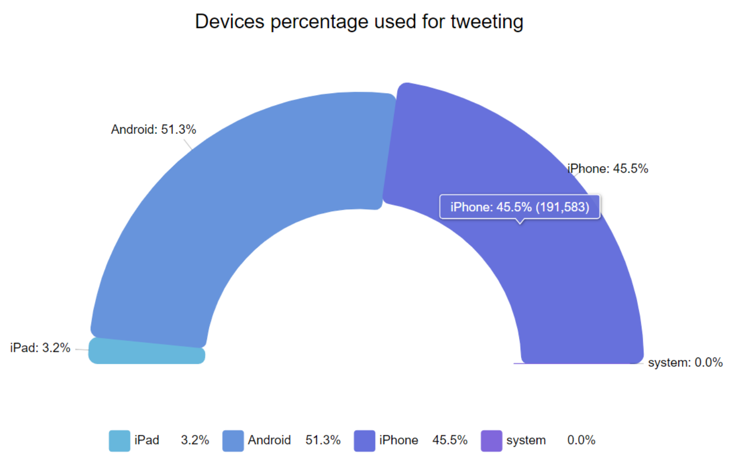
5. Classification of devices used by different account holders for tweeting.

<u>Query and Output:</u>



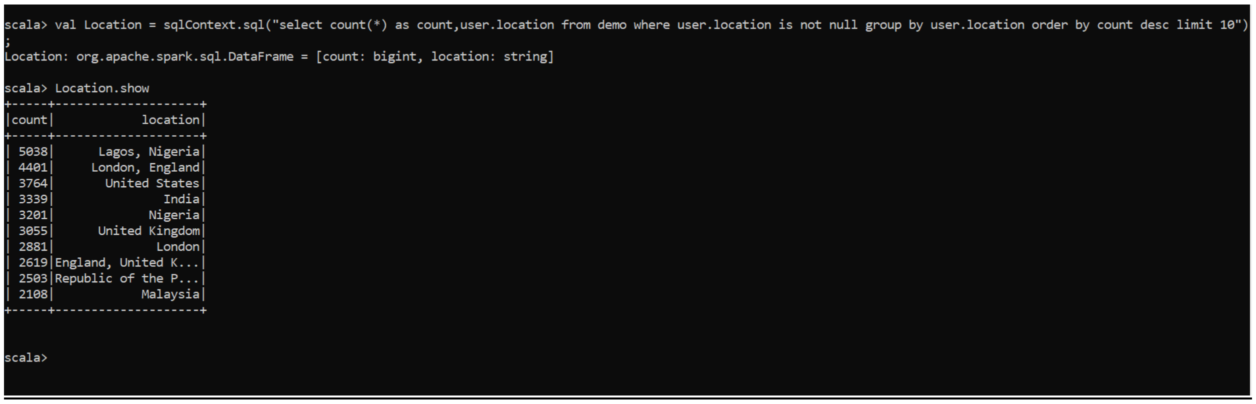
```
scala> val a = sqlContext.sql("select count(*) as count,'iPhone' as Source from demo where upper(source) like '%IPHONE%' union select count(*) as count,'Android' as Source
from demo where upper(source) like '%ANDROID%' union select count(*) as count,'Web' as Source from demo where upper(source) like '%WEB%' union select count(*) as count,'iPa
d' as Source from demo where upper(source) like '%IPAD%' order by count desc");
a: org.apache.spark.sql.DataFrame = [count: bigint, Source: string]

scala> a.show
+------+-------+
| count| Source|
+------+-------+
|216016|Android|
|191584| iPhone|
| 77388|    Web|
| 13411|   iPad|
+------+-------+
```

<u>Visualization:</u>



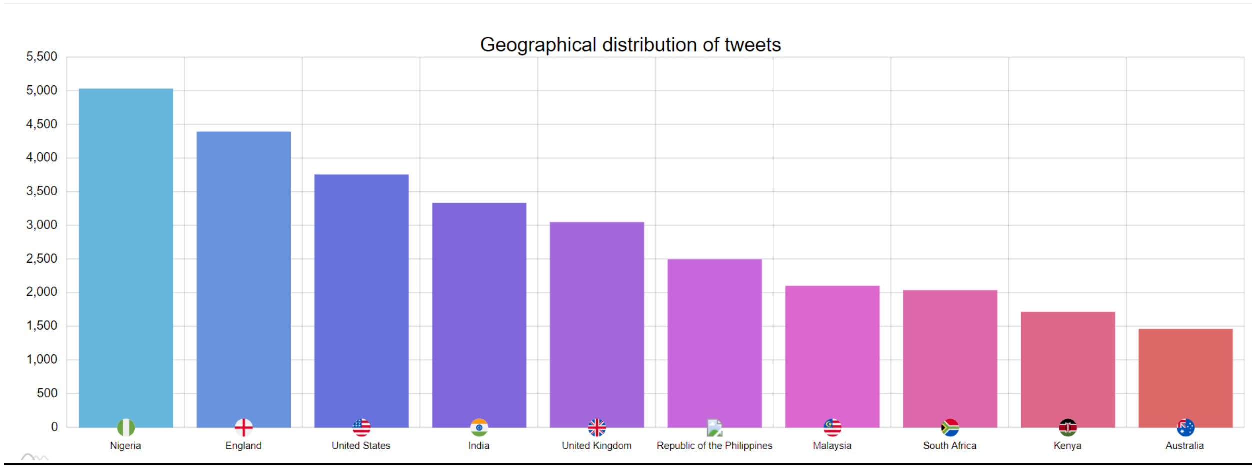
6.Geographical distribution of account holders across countries.

Query and Output



```
scala> val Location = sqlContext.sql("select count(*) as count,user.location from demo where user.location is not null group by user.location order by count desc limit 10")
;
Location: org.apache.spark.sql.DataFrame = [count: bigint, location: string]

scala> Location.show
+-----+-------------------+
|count|           location|
+-----+-------------------+
| 5038|      Lagos, Nigeria|
| 4401|     London, England|
| 3764|       United States|
| 3339|               India|
| 3201|             Nigeria|
| 3055|      United Kingdom|
| 2881|              London|
| 2619|England, United K...|
| 2503|Republic of the P...|
| 2108|            Malaysia|
+-----+-------------------+


scala>
```

Visualizations

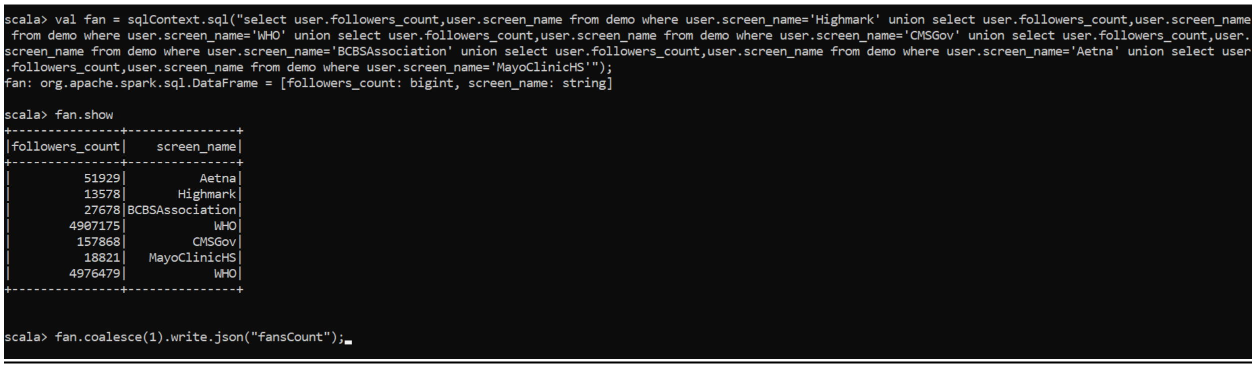


Health Organizations and NCD - Specific Metrics

Social Media Platform	Sample Metrics	Description
Twitter	Followers count	Followers count of various health organizations (WHO,Aetna,Mayo Clinic,Highmark BCBS etc)
	NCD Promotors	Percentage of NCD tweets by Health Institutions/NCD Promoters
	Top 10 Account Holders	Top 10/Active account holders , tweeted on health organizations such as Mayo, Cerner etc.
	Health Organization tweets	Total count of tweets, tweeted on Health Organizations.
	Holistic View	Hoslistic view of Health Oraganizations twitter data.

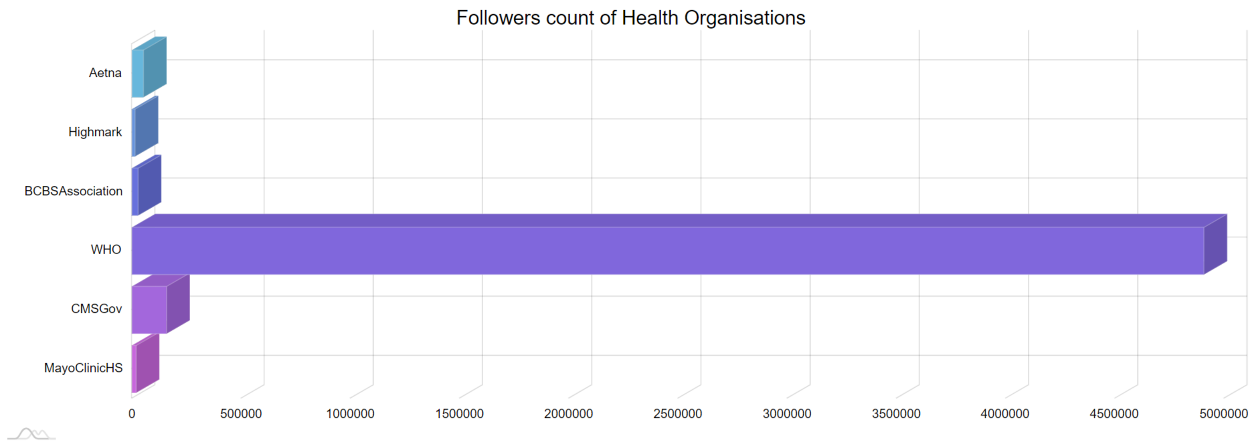
7.Followers count of various health organizations (WHO, Aetna, Mayo Clinic,Highmark BCBS etc)

Query & Output



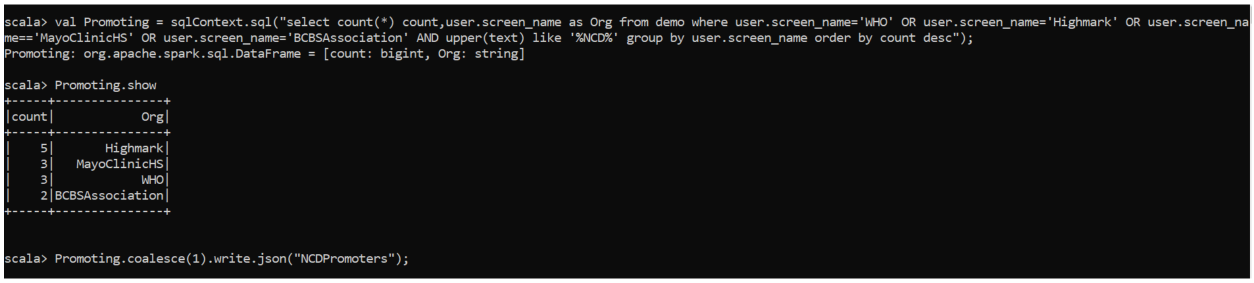
```
scala> val fan = sqlContext.sql("select user.followers_count,user.screen_name from demo where user.screen_name='Highmark' union select user.followers_count,user.screen_name
 from demo where user.screen_name='WHO' union select user.followers_count,user.screen_name from demo where user.screen_name='CMSGov' union select user.followers_count,user.
screen_name from demo where user.screen_name='BCBSAssociation' union select user.followers_count,user.screen_name from demo where user.screen_name='Aetna' union select user
.followers_count,user.screen_name from demo where user.screen_name='MayoClinicHS'");
fan: org.apache.spark.sql.DataFrame = [followers_count: bigint, screen_name: string]

scala> fan.show
+---------------+---------------+
|followers_count|    screen_name|
+---------------+---------------+
|          51929|          Aetna|
|          13578|       Highmark|
|          27678|BCBSAssociation|
|        4907175|            WHO|
|         157868|         CMSGov|
|          18821|   MayoClinicHS|
|        4976479|            WHO|
+---------------+---------------+


scala> fan.coalesce(1).write.json("fansCount");
```

Visualization



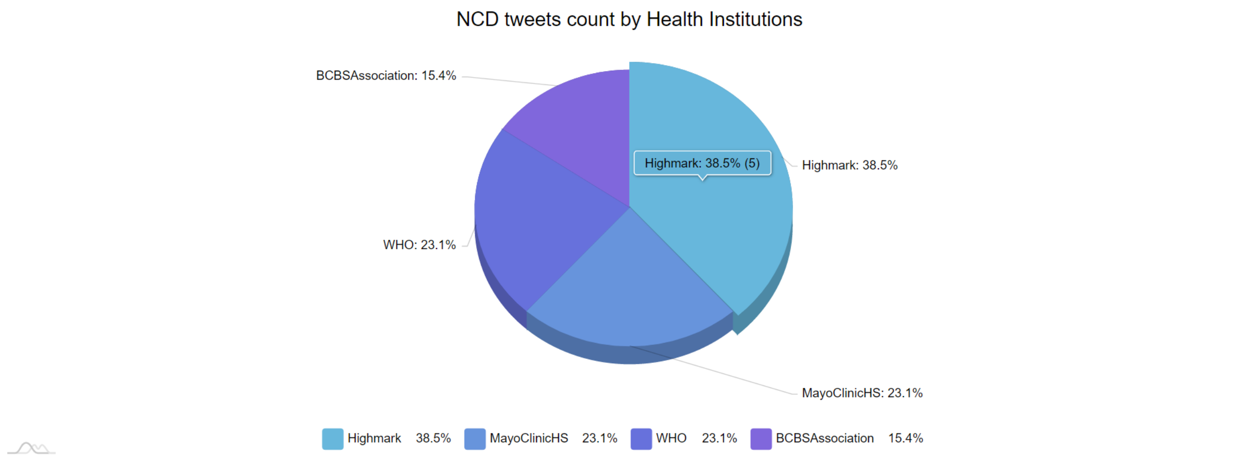
8.Percentage of NCD tweets by Health Institutions/NCD Promoters

Query & Output



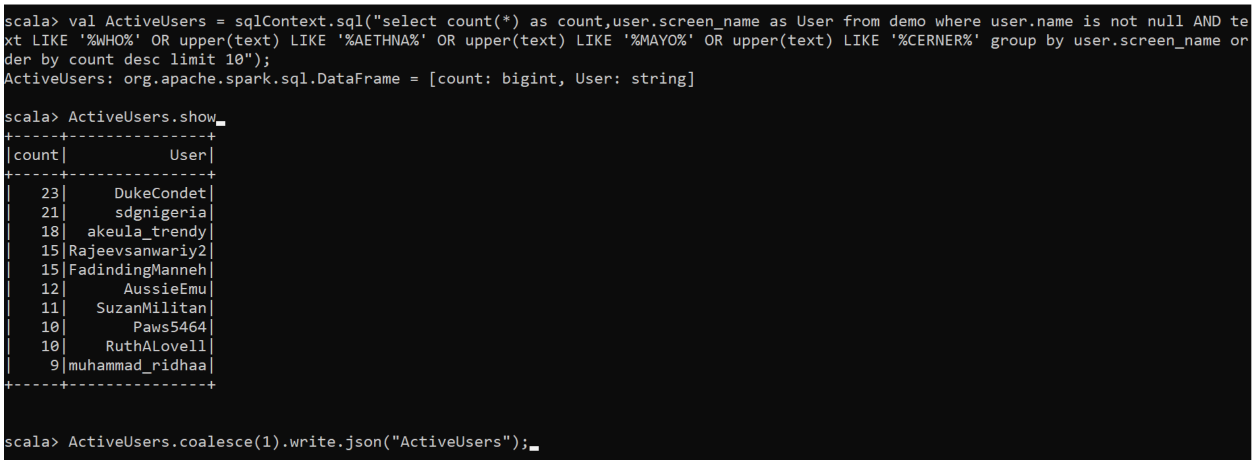
```
scala> val Promoting = sqlContext.sql("select count(*) count,user.screen_name as Org from demo where user.screen_name='WHO' OR user.screen_name='Highmark' OR user.screen_na
me=='MayoClinicHS' OR user.screen_name='BCBSAssociation' AND upper(text) like '%NCD%' group by user.screen_name order by count desc");
Promoting: org.apache.spark.sql.DataFrame = [count: bigint, Org: string]

scala> Promoting.show
+-----+---------------+
|count|            Org|
+-----+---------------+
|    5|       Highmark|
|    3|   MayoClinicHS|
|    3|            WHO|
|    2|BCBSAssociation|
+-----+---------------+


scala> Promoting.coalesce(1).write.json("NCDPromoters");
```

Visualization



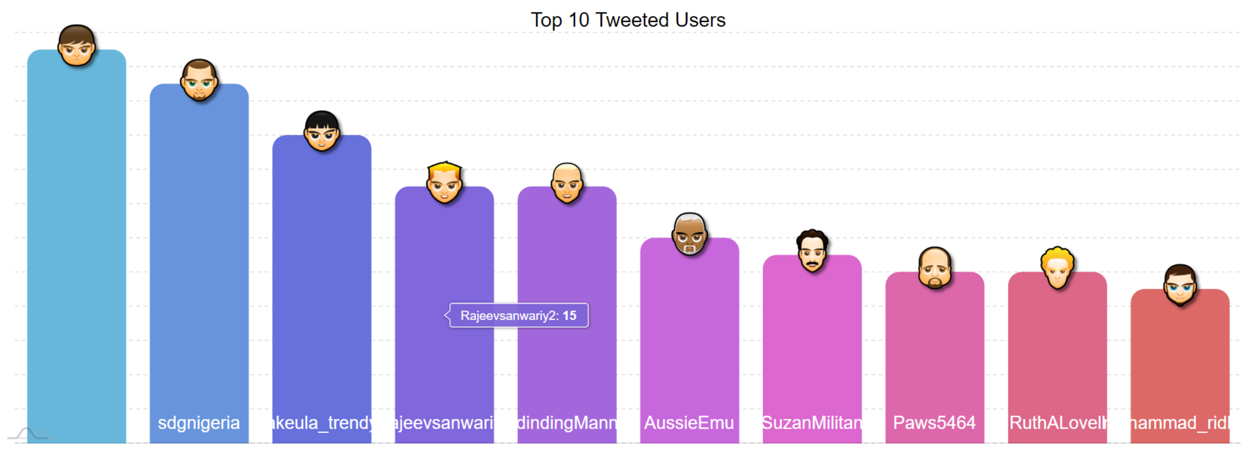
9.Top 10/Active account holders , tweeted on health organizations such as Mayo, Cerner etc.

Query & Output



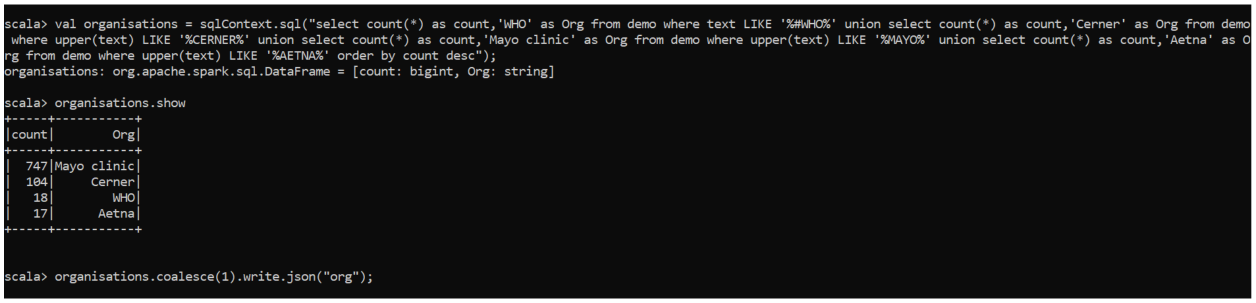
```
scala> val ActiveUsers = sqlContext.sql("select count(*) as count,user.screen_name as User from demo where user.name is not null AND te
xt LIKE '%WHO%' OR upper(text) LIKE '%AETHNA%' OR upper(text) LIKE '%MAYO%' OR upper(text) LIKE '%CERNER%' group by user.screen_name or
der by count desc limit 10");
ActiveUsers: org.apache.spark.sql.DataFrame = [count: bigint, User: string]

scala> ActiveUsers.show
+-----+---------------+
|count|           User|
+-----+---------------+
|   23|     DukeCondet|
|   21|     sdgnigeria|
|   18|   akeula_trendy|
|   15|Rajeevsanwariy2|
|   15|FadindingManneh|
|   12|       AussieEmu|
|   11|    SuzanMilitan|
|   10|        Paws5464|
|   10|      RuthALovell|
|    9|muhammad_ridhaa|
+-----+---------------+


scala> ActiveUsers.coalesce(1).write.json("ActiveUsers");
```

Visualization



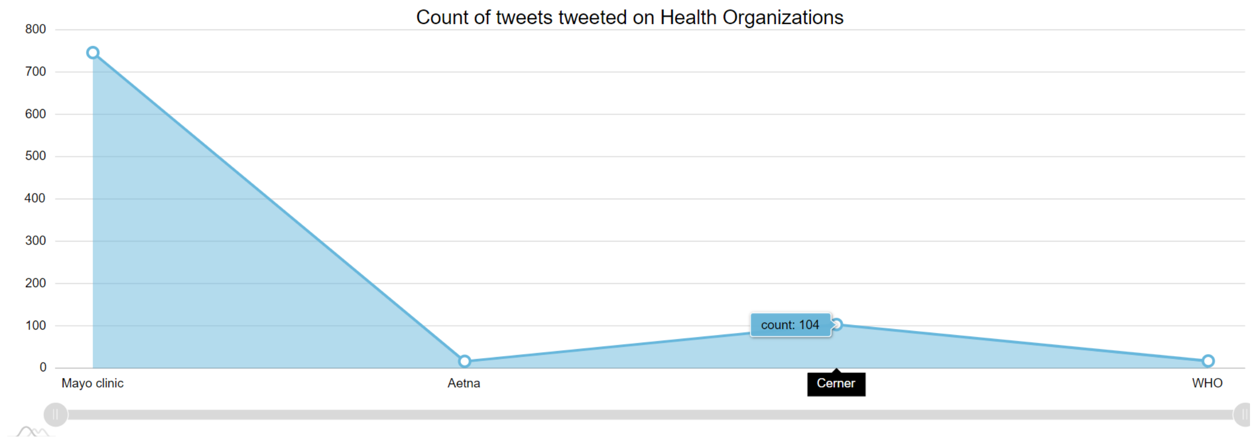
10.Total count of tweets, tweeted on Health Organizations.

Query & Output



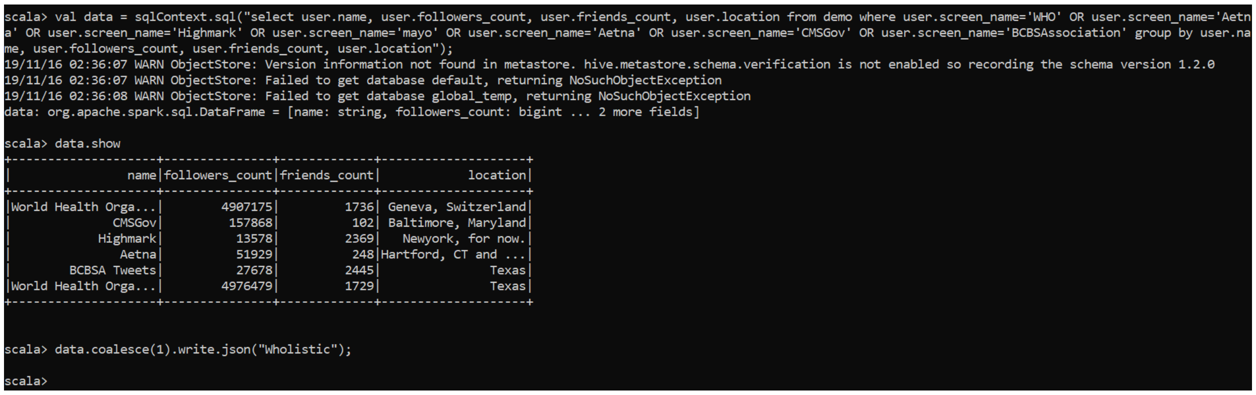
```
scala> val organisations = sqlContext.sql("select count(*) as count,'WHO' as Org from demo where text LIKE '%#WHO%' union select count(*) as count,'Cerner' as Org from demo
 where upper(text) LIKE '%CERNER%' union select count(*) as count,'Mayo clinic' as Org from demo where upper(text) LIKE '%MAYO%' union select count(*) as count,'Aetna' as O
rg from demo where upper(text) LIKE '%AETNA%' order by count desc");
organisations: org.apache.spark.sql.DataFrame = [count: bigint, Org: string]

scala> organisations.show
+-----+-----------+
|count|        Org|
+-----+-----------+
|  747|Mayo clinic|
|  104|     Cerner|
|   18|        WHO|
|   17|      Aetna|
+-----+-----------+


scala> organisations.coalesce(1).write.json("org");
```

Visualization



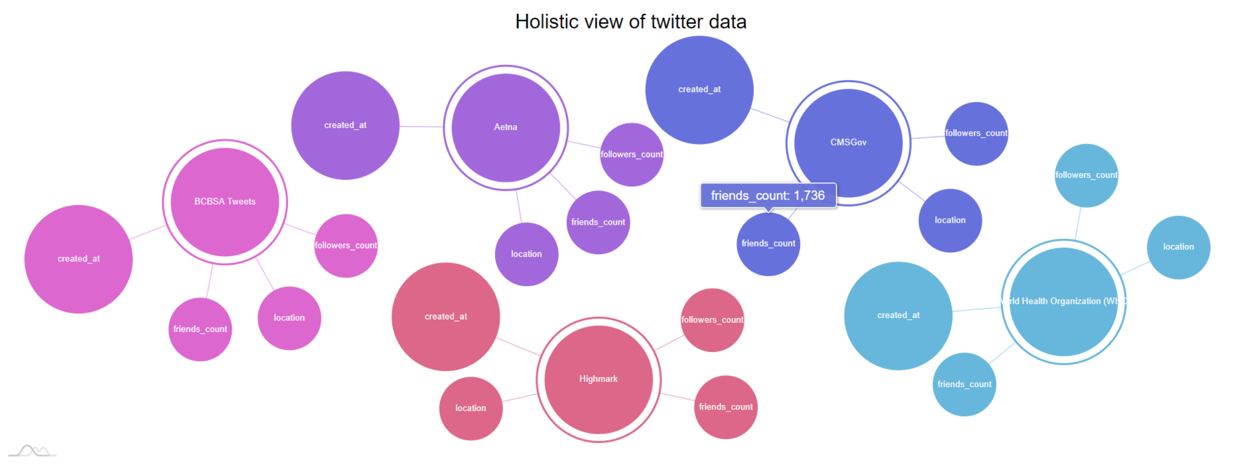
11.Hoslistic view of Health Organizations twitter data.

Query & Output



```
scala> val data = sqlContext.sql("select user.name, user.followers_count, user.friends_count, user.location from demo where user.screen_name='WHO' OR user.screen_name='Aetn
a' OR user.screen_name='Highmark' OR user.screen_name='mayo' OR user.screen_name='Aetna' OR user.screen_name='CMSGov' OR user.screen_name='BCBSAssociation' group by user.na
me, user.followers_count, user.friends_count, user.location");
19/11/16 02:36:07 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not enabled so recording the schema version 1.2.0
19/11/16 02:36:07 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException
19/11/16 02:36:08 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
data: org.apache.spark.sql.DataFrame = [name: string, followers_count: bigint ... 2 more fields]

scala> data.show
+--------------------+---------------+-------------+--------------------+
|                name|followers_count|friends_count|            location|
+--------------------+---------------+-------------+--------------------+
|World Health Orga...|        4907175|         1736| Geneva, Switzerland|
|              CMSGov|         157868|          102| Baltimore, Maryland|
|            Highmark|          13578|         2369|   Newyork, for now.|
|               Aetna|          51929|          248|Hartford, CT and ...|
|         BCBSA Tweets|          27678|         2445|               Texas|
|World Health Orga...|        4976479|         1729|               Texas|
+--------------------+---------------+-------------+--------------------+


scala> data.coalesce(1).write.json("Wholistic");

scala>
```

Visualization



Output Files Path

https://mailmissouri-my.sharepoint.com/:f:/r/personal/yddpw_mail_umkc_edu/Documents/PB%20Project-%20Phase-2?csf=1&e=JhFDxM

Project Summary

In this phase, the queries are executed in spark and the insights are visualized using am4charts for the twitter data.

Testing & Debugging

Each and every query is unit tested against the data set that's collected and also with the real time twitter data (from the accounts). All the defects that are identified in the process of testing are fixed by analyzing the data and the queries thoroughly.

Improvements Plan

An Angular 7 application to show all the above executed queries along with the docker container implementation will be taken care in the next week.

Appendix - References

- https://developer.twitter.com
- http://adilmoujahid.com/posts/2014/07/twitter-analytics/