Phase 2 Report

Analysis of Tweets on Corona Virus / COVID-19

Principles of Big Data Management

Srichakradhar Reddy Nagireddy – 16298670 – [snp8b@mail.umkc.edu](mailto:snp8b@mail.umkc.edu)Sathwik Banginimatam – 12571133 – [sb5n8@mail.umkc.edu](mailto:sb5n8@mail.umkc.edu)Sai Bhavani Nikita Rayapareddy –16274994 [– srnxb@mail.umkc.edu](mailto:–srnxb@mail.umkc.edu)

**Introduction**:

We have created a Big Data Spark pipeline using Apache Spark, AWS Kinesis and Hadoop. We have performed the time series analysis on Twitter data filtered by hashtags. We have analyzed the sentiment, user location, time, source, language and few other parameters of the tweets.

The streaming data format can be manipulated in pySpark with Pandas. We have handled the streaming data in a way of near real-time processing.

**Technology and Tools Used:**

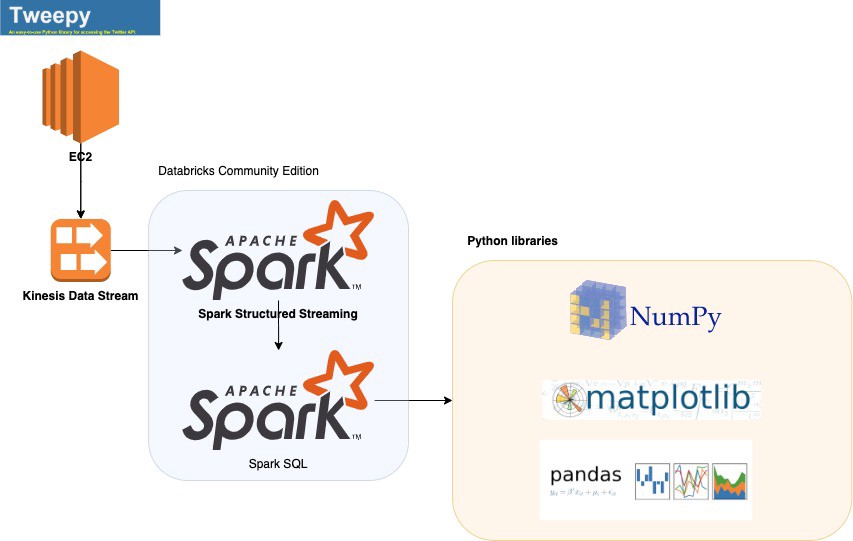
* Environment: Ubuntu
* Tools: VS Code, Spark 2.4.5, AWS Kinesis
* Java REST API for backend services
* AngularJS for front end
* Plotly.js, AmCharts, Charts.js, Google Maps for visualization

**Github URL:**

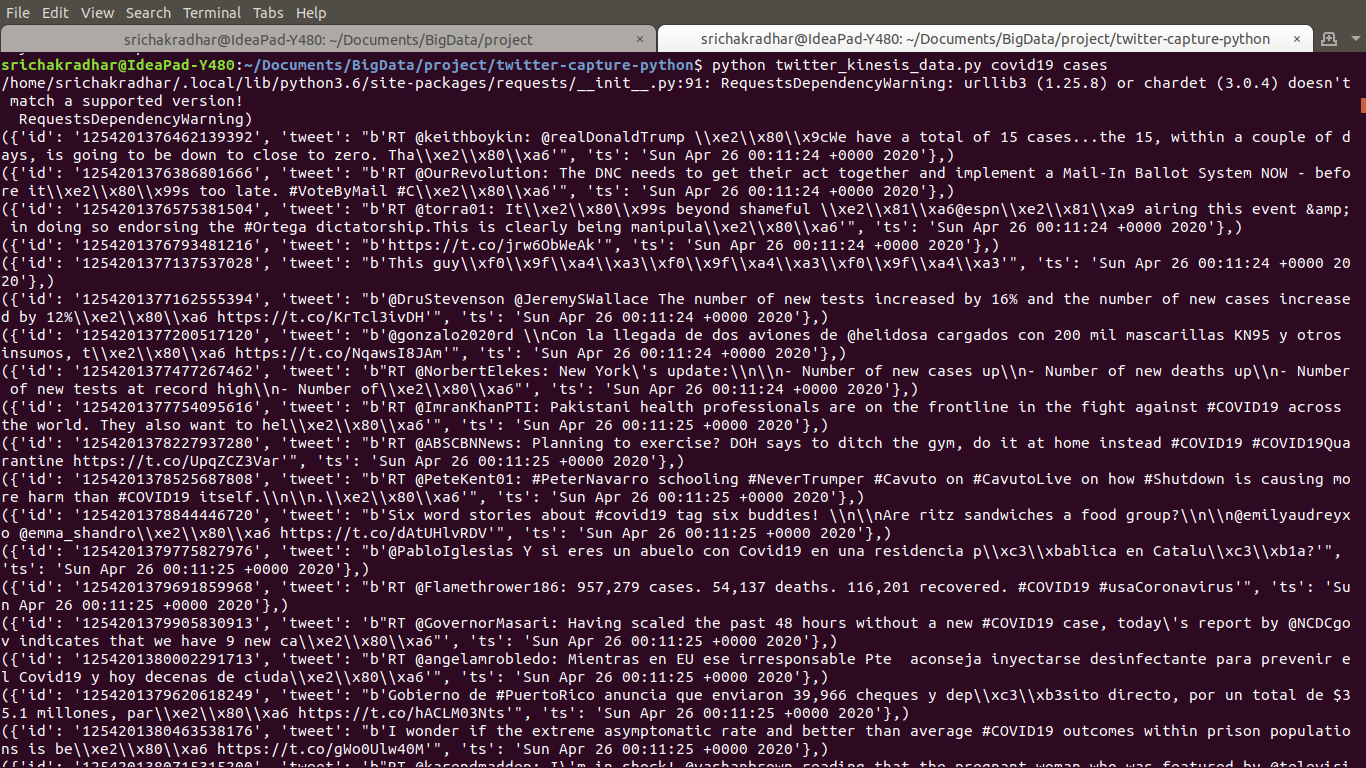
**Part 1: Capture tweets**

We used Tweepy to capture the tweets.

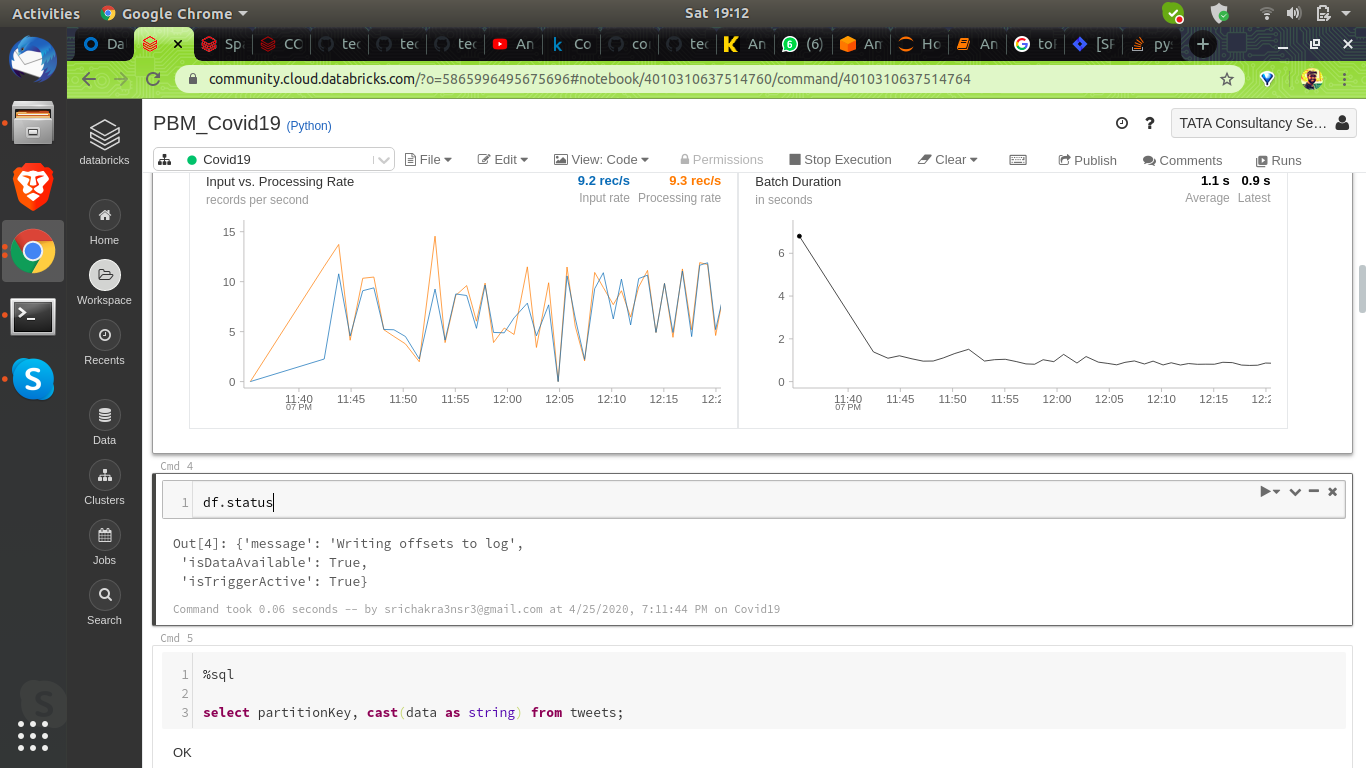
**Part 2: Project Architecture:**



**Tweet Collection Streaming:**

****

**Spark Streaming** is a scalable fault-tolerant streaming processing system based on RDD that natively supports both batch and streaming workloads.



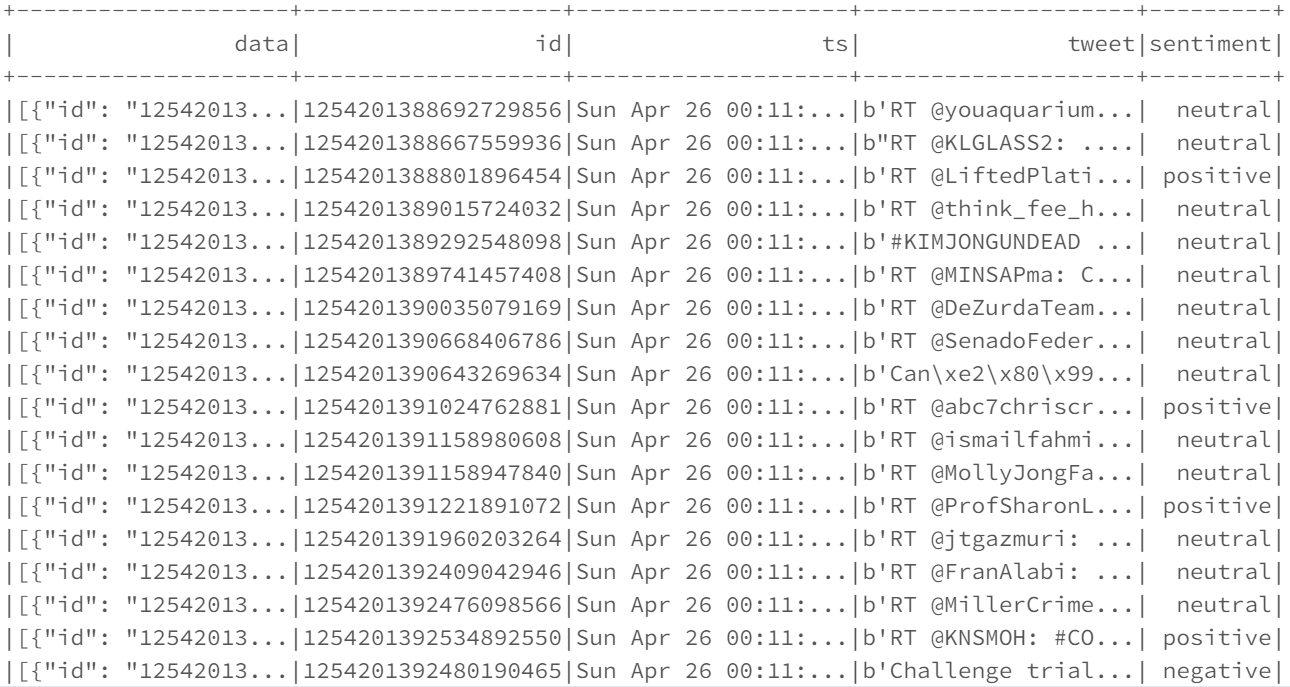
**Spark SQL** allows us to manipulate DataFrames.

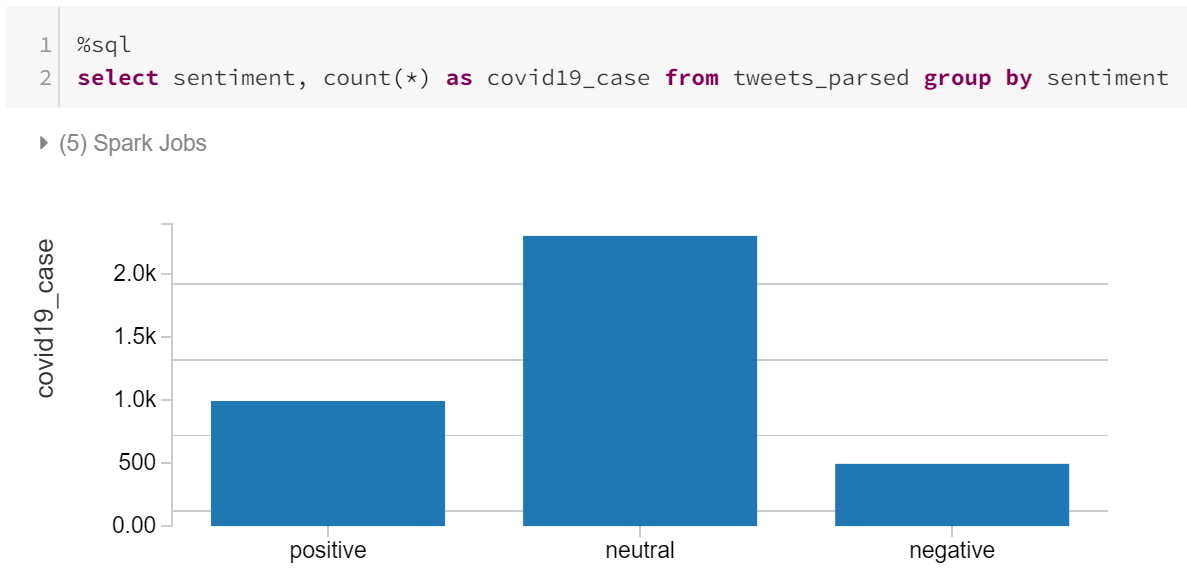
**Part 3: Queries:**

1. Sentiment



**Description**: To identify Positive, Negative and Neutral reactions of people on Covid-19 from the tweets.





1. User profile – location from where the tweet is posted

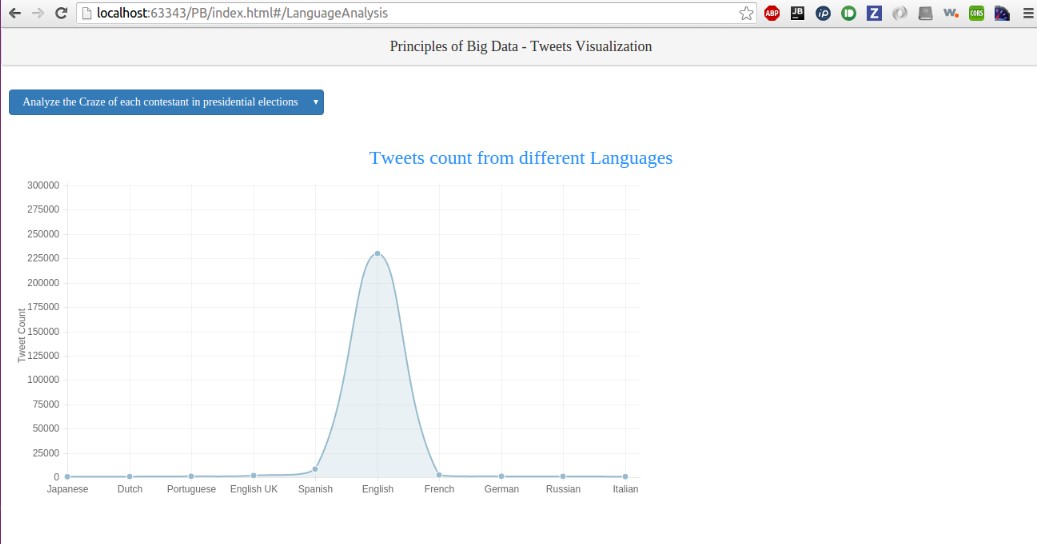
SELECT user.name, text, geo, user.profile\_image\_url from tweets where geo IS NOT NULL

**Description**: This query collects tweeted username, user location coordinates, tweeted text and is profile image url. This data is used to display on world map using coordinates.

1. Number of tweets per language

SELECT user.lang, COUNT(\*) as lang\_user\_count from tweets WHERE user.lang IS NOT NULL  
GROUP BY user.lang ORDER BY lang\_user\_count DESC LIMIT 10

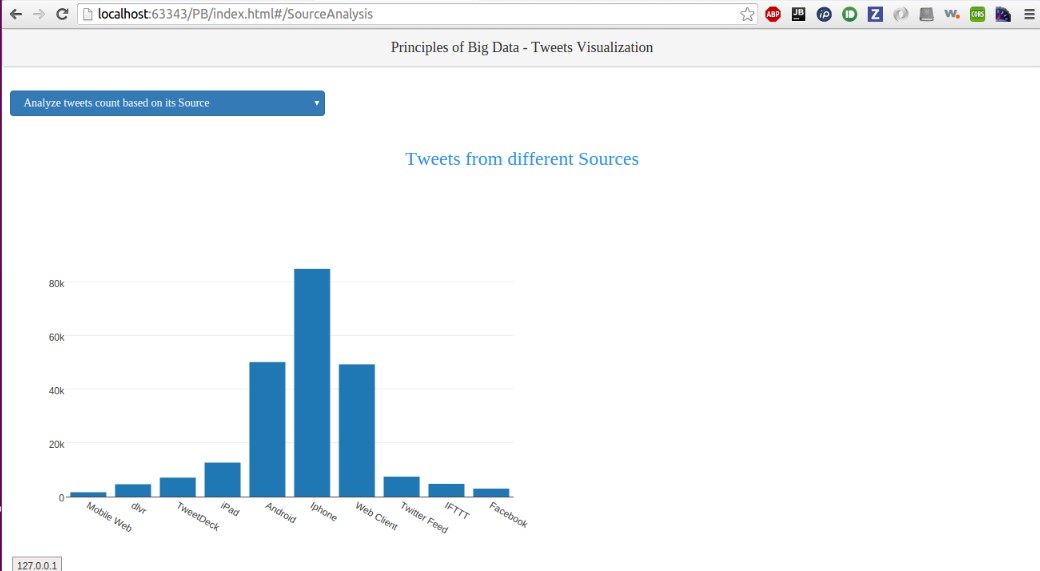
**Description:** This query collects the language information of each tweet tweeted along with the total count in each language.



1. Souce of Tweet – iPad, TweetDeck, IFTTT, Facebook, Android, iPhone, Mobile Web, Web Client, etc.

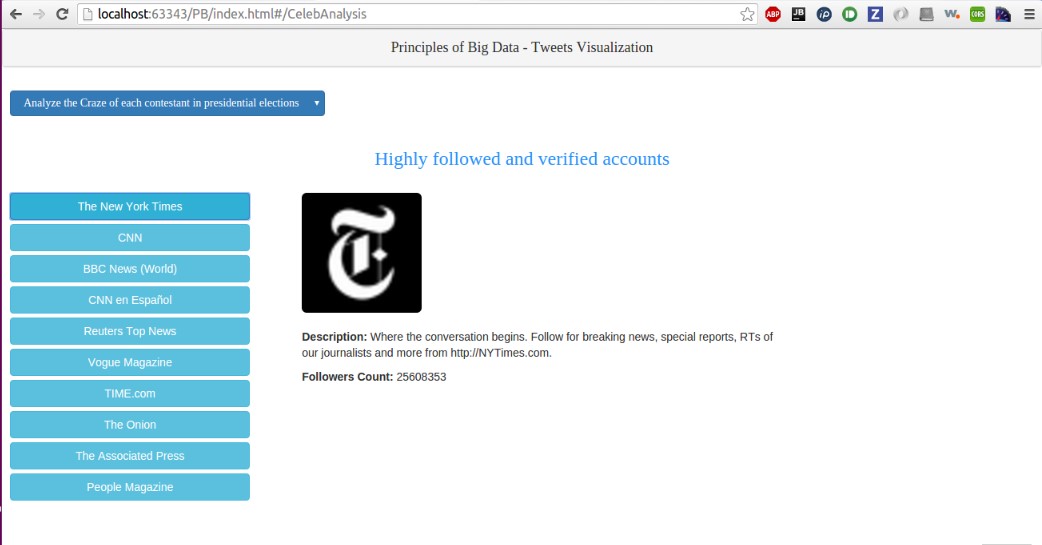
SELECT source, count(source) as c from tweets group by source order by c desc limit 10

**Description**: This query collects information about the source of tweet from where it generated.



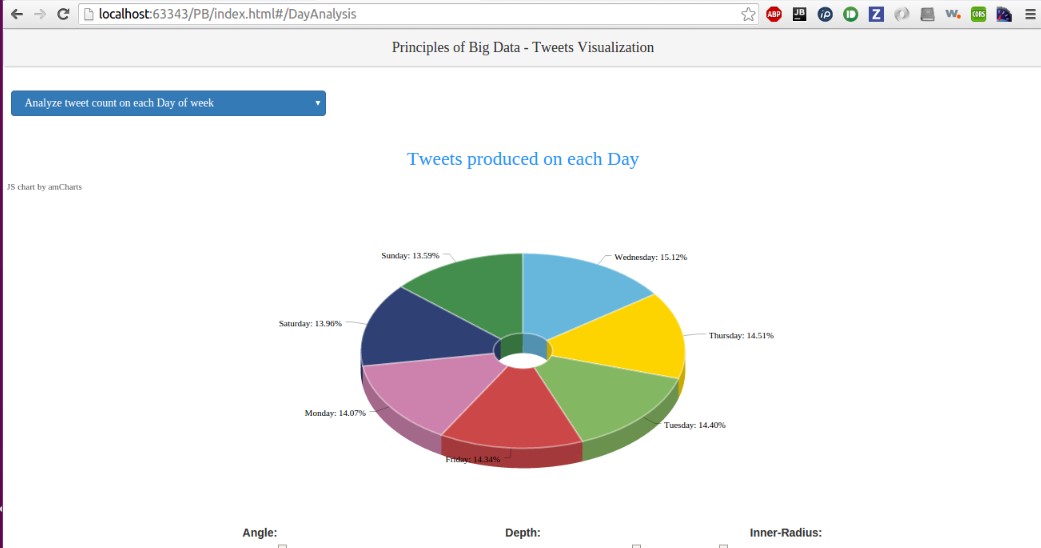
1. All verified accounts along with the description of account, name, followers count and profile image url

SELECT DISTINCT user.name,user.followers\_count as c, user.profile\_image\_url, user.description  
FROM tweets where user.verified = true ORDER BY c desc limit 25



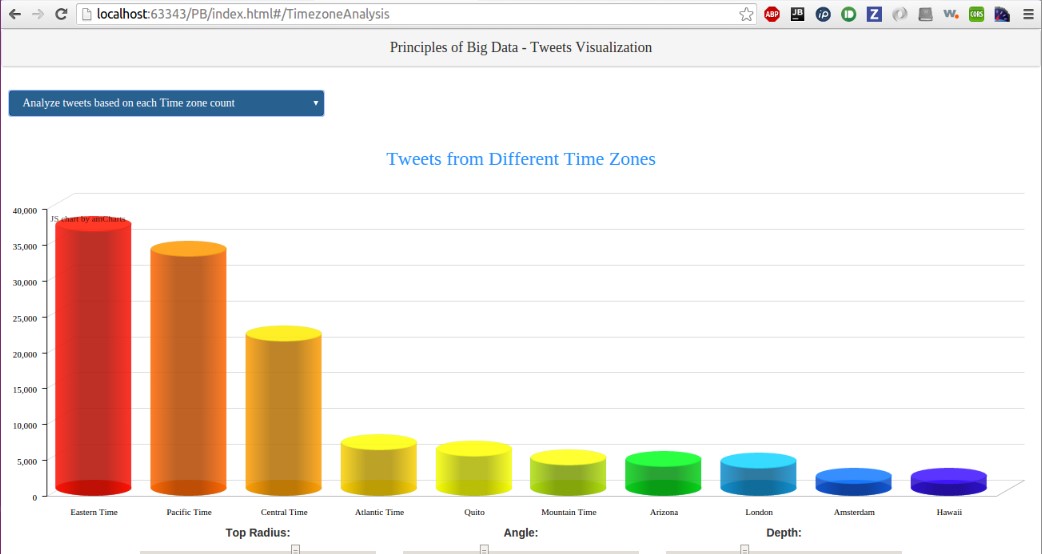
1. query to collect information about on which day of the week was the tweet created

SELECT SUBSTR(user.created\_at, 1, 3) as day, COUNT(\*) as count from tweets WHERE user.created\_at is NOT NULL GROUP BY SUBSTR(user.created\_at, 1, 3) order by count DESC



1. Time zone from where each tweet is generated and outputs the count of each time zone

SELECT user.time\_zone, COUNT(\*) AS count from tweets where user.time\_zone is NOT NULL  
GROUP BY user.time\_zone ORDER BY count DESC limit 10



1. Average of friends count

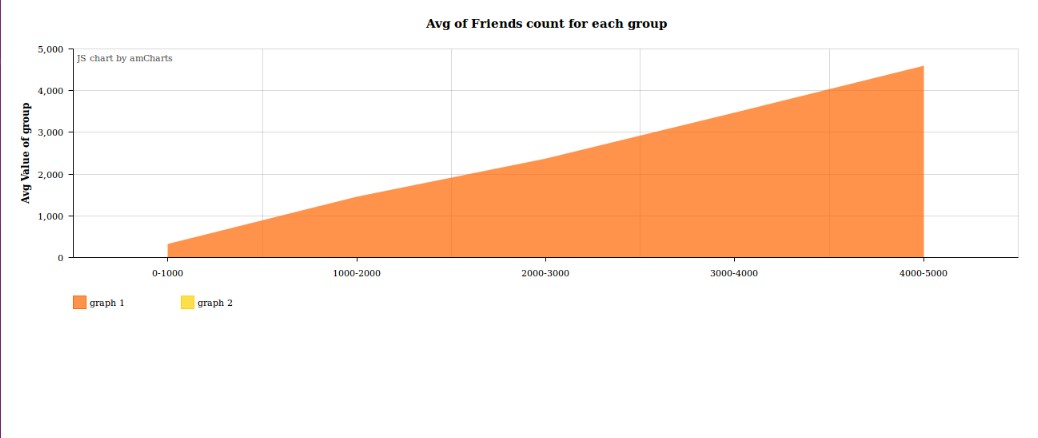
val userName = sqlContext.sql("SELECT avg(user.friends\_count) from tweets user.friends\_count IS NOT NULL and user.friends\_count between 0 and 1000")

val userName = sqlContext.sql("SELECT avg(user.friends\_count) from tweets user.friends\_count IS NOT NULL and user.friends\_count between 1000 and 2000")

val userName = sqlContext.sql("SELECT avg(user.friends\_count) from tweets user.friends\_count IS NOT NULL and user.friends\_count between 2000 and 3000")

val userName = sqlContext.sql("SELECT avg(user.friends\_count) from tweets user.friends\_count IS NOT NULL and user.friends\_count between 3000 and 4000")

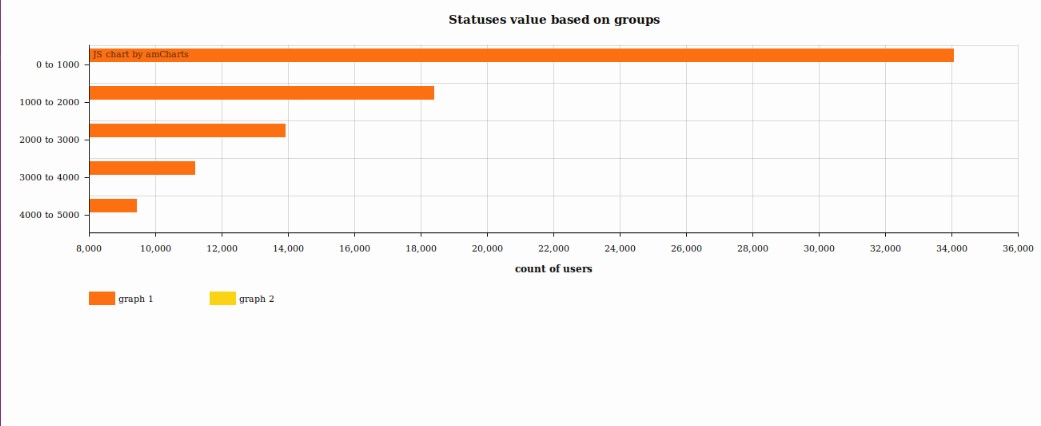
val userName = sqlContext.sql("SELECT avg(user.friends\_count) from tweets user.friends\_count IS NOT NULL and user.friends\_count between 4000 and 5000")



1. **Statuses value based on groups**

val userName = sqlContext.sql("SELECT count(user.statuses\_count) from tweets WHERE user.statuses\_count IS NOT NULL and user.statuses\_count between 0 and 1000") val userName = sqlContext.sql("SELECT count(user.statuses\_count) from tweets WHERE user.statuses\_count IS NOT NULL and user.statuses\_count between 1000 and 2000") val userName = sqlContext.sql("SELECT count(user.statuses\_count) from tweets WHERE user.statuses\_count IS NOT NULL and user.statuses\_count between 2000 and 3000") val userName = sqlContext.sql("SELECT count(user.statuses\_count) from tweets WHERE user.statuses\_count IS NOT NULL and user.statuses\_count between 3000 and 4000") val userName = sqlContext.sql("SELECT count(user.statuses\_count) from tweets WHERE user.statuses\_count IS NOT NULL and user.statuses\_count between 4000 and 5000")

**Description:** Statuses value based on groups



1. Frequency of tweets from different countries (Indicating spread of virus in Italy vs Spain vs US, etc..) List of countries sorted by number of cases
2. % Tweets that mention Deaths vs Confirmed cases