ANALYZING & VISUALIZING TWITTER DATA



Under the Guidance of:

**Prof. Praveen Rao**

**Venubabu Kolla**

**Hastimal Jangid**

**Dara, Venkata Sai Sandeep(vd7d3@mail.umkc.edu)**

**Podili, Venkata Krishna(vp327@mail.umkc.edu)**

**Puri, Varun(**[vp4gb@mail.umkc.edu](mailto:vp4gb@mail.umkc.edu)**)**

Topic: Traveling

Phase-2

Twitter Big Data Analytics using Spark

**CS5540PB Project Report**

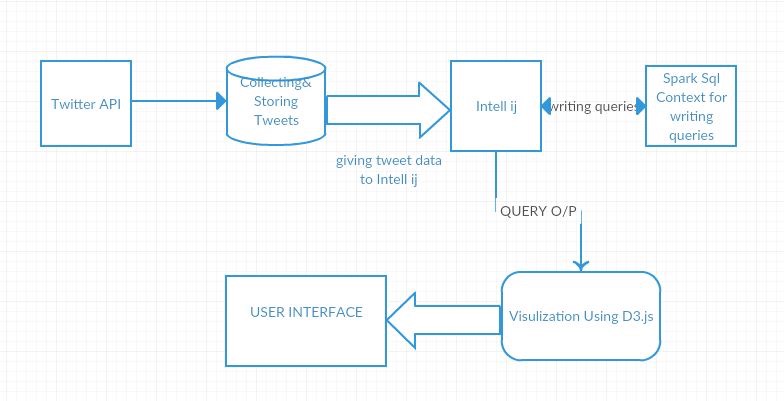
**Introduction:-**

Big Data is an evolving term that describes any voluminous amount of structured and unstructured data that has the potential to be mined for information. Big-data analytics is the process of examining large datasets containing a variety of datatypes i.e. biodata to uncover hidden patterns, unknown correlations, market trends, customer preferences and other useful business information. With Biodata analytics, data scientists and other can analyze huge volumes of data that conventional analysts and business intelligence solutions can’t touch. A latest and trending feather in the Hadoop’s Cap is Apache SPARK. Apache spark is an open source engine developed specifically for handling large scale data processing and analytics. Spark enable user to run large scale data analytics application across various clustered computers. In our Project we have use the IntelliJ Scala plugin to analysis our data and get the results.

**Project Description:-**

“We travel not to escape life, but for life not to escape us”. Traveling is about exploring world to yourself. Now-a-days, one of the best part of social Networking world is that you can live the Journeys or Moment of your friends, your icons from your mobile or laptop screen. People share their beautiful experience, their journeys, their activities, by posting their photographs, Videos on social networking sites. This ever growing unstructured data provide seamless opportunities in every dimension for everyone for example many startup or establish traveling business to identify their clients. In our project we have tried to collected the tweets based on topics with Hashtag traveling, Mountain, Beaches, Vacation, Wanderlust, Tourism and then perform some analysis on the collected data using Spark SQL Context by performing the analytical queries on it. Finally we have some visualizations like big sheets tools to display the output.

**ARCHITECTURAL DIAGARAM:-**



**Software Modules:**

**1) Data Collection:**

We have collected tweets from twitter and stored them in the form of JSON files. The collection of tweets is done using python twitter streaming. The search is done based upon the following keywords: #Travel, #Beaches, #Beautiful, #Holidays, #Nature, #Tourism, #vacation, #wanderlust, #traveler-gram, #travelling, #traveldairies, #instatravel, #weekendtravel etc.

**2) Data Analysis:**

We used JSONLINT tool (on <http://jsonlint.com/>) to analyze the schema of the data collected. Based on the schema we designed our queries.

**3) Data processing:**

Here are the steps that we followed for data processing

* Tweets: - Created one single JSON file Tweets by combining all collected files using code. Contains all tweets collected based on the hashtags mentioned above.
* Configured Scala on IntelliJ (installed Scala plugins on IntelliJ) and wrote queries using spark SQL context

**4) Data Visualization:**

For the Visualization of the output we have used the **High Charts cloud** services (https://cloud.highcharts.com/). Here are the steps used for visualization:

* After running the query we got the output in the JSON format in text files.
* We use the konlone.io tool to convert the output from JSON to CSV format.
* Further we uploaded the output file on high chart cloud link and selected appropriate graph type.

**Analytical Queries & Visualization:**

**Query1: Based on Language**

In this query we find the count of the language most used by travelers.

**Explanation:**

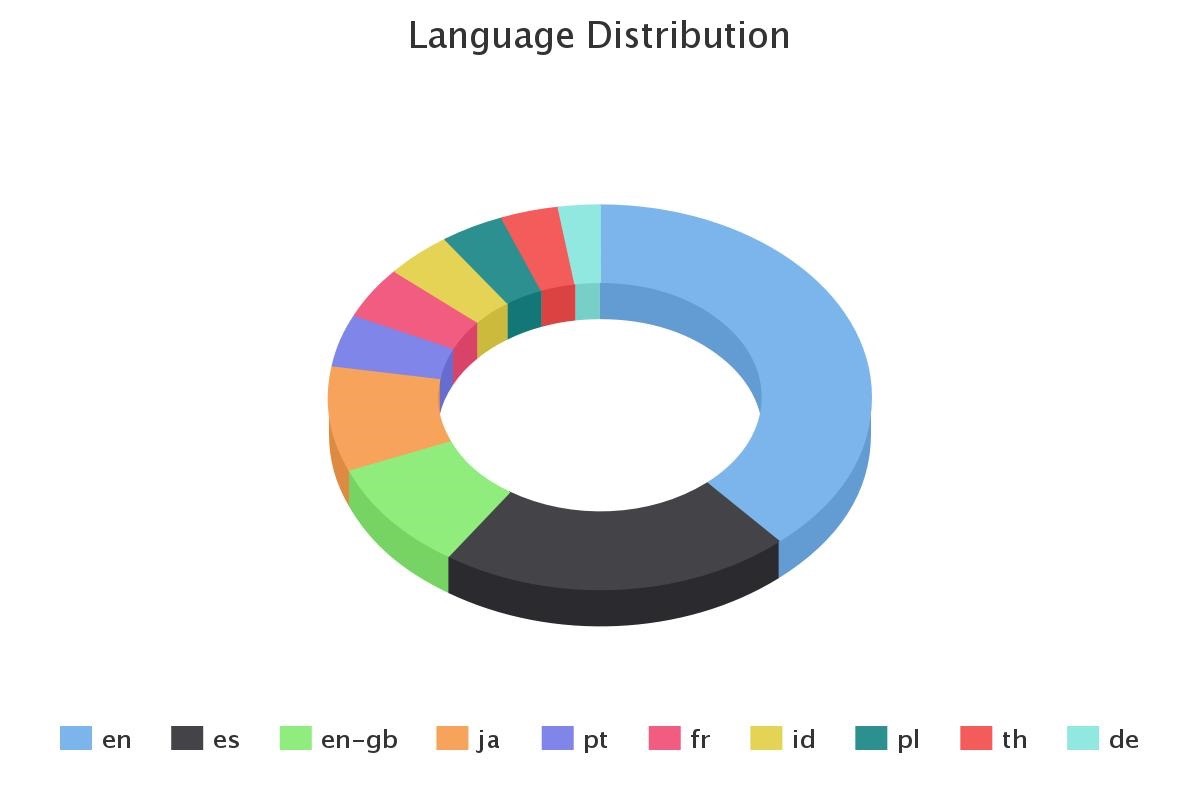
Here we are retrieving the count of all languages in columns “**user.lang**” and have used GROUP BY clause to show top 10 languages used by travelers. Output of the query is stored in file Q1.csv

**Query:**

**select user.lang,count(\*) as count from Tweets where user.lang IS not NULL group by user.lang order by count desc limit 10").toJSON.coalesce(1).saveAsTextFile("/Users/Desktop/Q1.csv")**

**Visualization:**

We have converted data in output file to CVS format as High charts accepts CSV data.



**Query2: Based on month**

To visualize in which month most people prefer to travel.

**Explanation**:

We have written two queries to achieve this

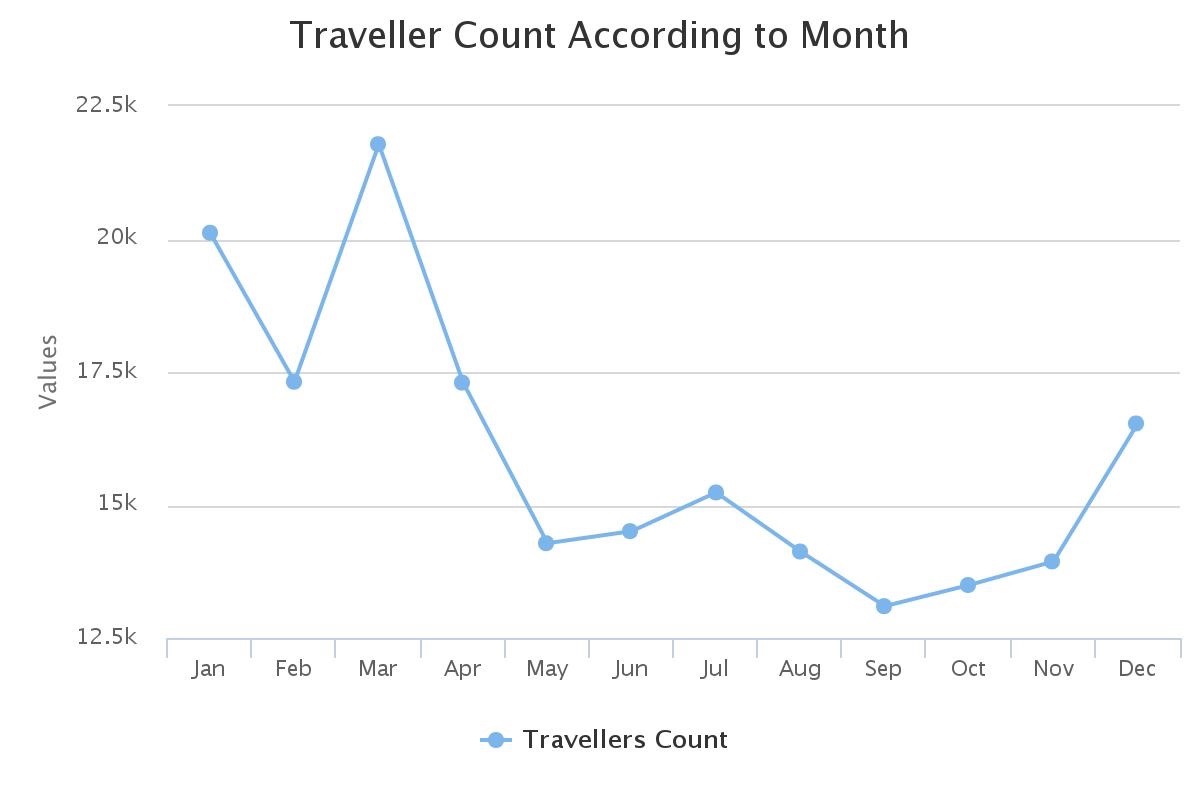
1. In first query we first took a substring (starting from 5th char to 7th character) from user. Created column and stored them in table Months.
2. In second query we selected date (actually month) and counted distinct date and used GROUP BY and DESC clause to show months in descending order in which people prefer to travel.
3. Output is stored in Q2.csv file.

**Query:**

* **val test = select substring(user.created\_at,5,3) as date from Tweets where user.created\_at is not null ")**
* **test.registerTempTable("Months")**
* **val query2=sqlContext.sql("select date,count(\*) as cnt from Months group by date order by cnt desc”).toJSON.coalesce(1).saveAsTextFile("/Users/Desktop/Q2.csv")**

**Visualization:**

We have converted data in output file to CVS format as High charts accepts CSV data.

****

**Query 3: To display top places where people want to visit.**

**Explanation**:-

We have written three queries to achieve this

1. In first query we first looked for keywords “Beach”, ”Mountain”, ”Valley”, “Dessert” and “Valleys” in column text content and stored the output in a table “places”
2. In second query we applied a switch statement and looked for the same keywords used in first query and if found then stored that keyword (in caps) in column FAMOUSPLACES. Stored output of query in test2 table.
3. Selected FAMOUSPLACES from test2 table and counted those places using count and used GROUP by (FAMOUSPLACES) and ORDER by clause to display famous places in descending order.

**Query:**

* **Val test = sqlContext.sql( “SELECT text from Tweets where text like '%Beach%' or text like '%Mountain%' or text like '%valley%' or text like '%Dessert%' or text like '%Landscape%'")**
* **test.registerTempTable("places")**
* **val test1 = sqlContext.sql(**

**"SELECT CASE WHEN text like '%Beach%' THEN 'BEACHES'" +**

**"WHEN text like '%Mountain%' THEN 'MOUNTAINS'" +**

**"WHEN text like '%valley%' THEN 'VALLEYS'" +**

**"WHEN text like '%Dessert%' THEN 'Desserts'" +**

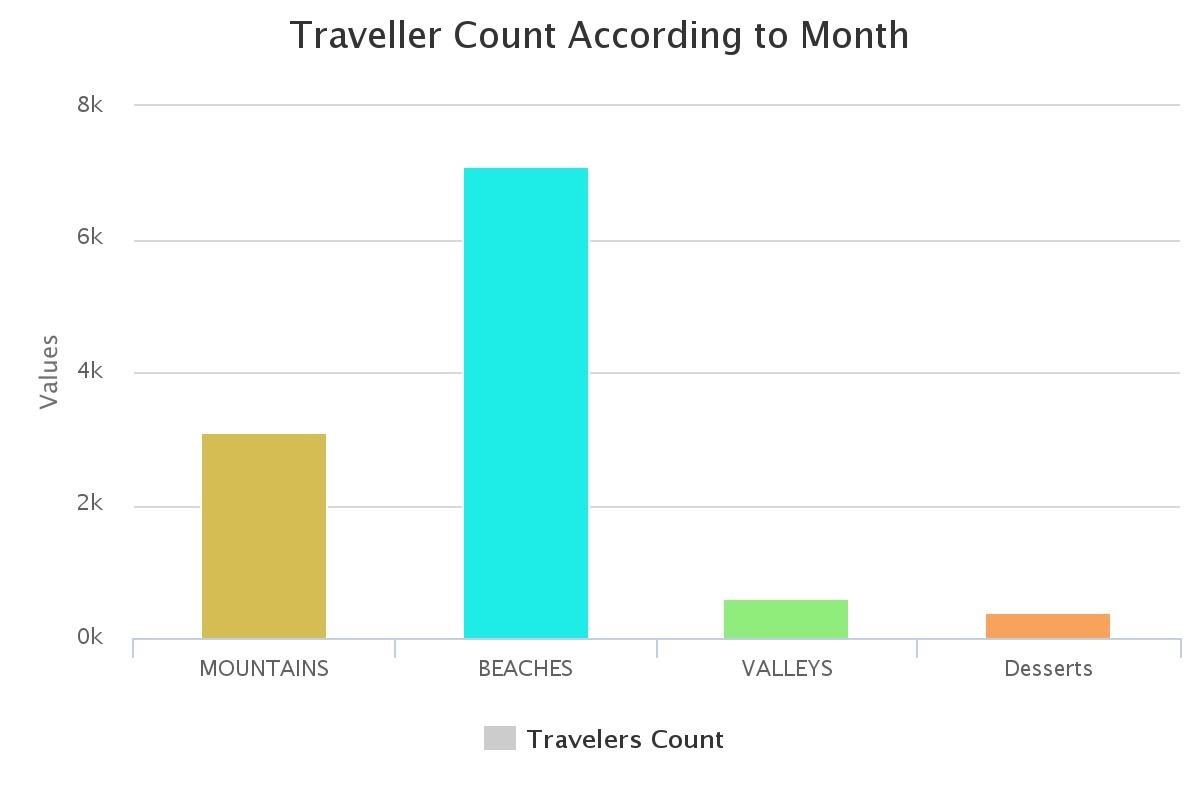
**"WHEN text LIKE '%Landscape%' THEN 'LANDSCAPE'" +**

**"END AS FAMOUSPLACES from places where text is not null ")**

* **test1.registerTempTable("test2")**
* **val test3 = sqlContext.sql("select FAMOUSPLACES, Count(\*) as Count from test2 where FAMOUSPLACES is not null group by FAMOUSPLACES order by Count DESC").toJSON.coalesce(1).saveAsTextFile("/Users/Desktop/Q3.csv")**

**Visualization:**

We have converted data in output file to CVS format as High charts accepts CSV data.



**Query 4: Select the top countries who tweet about traveling**

**Explanation:**

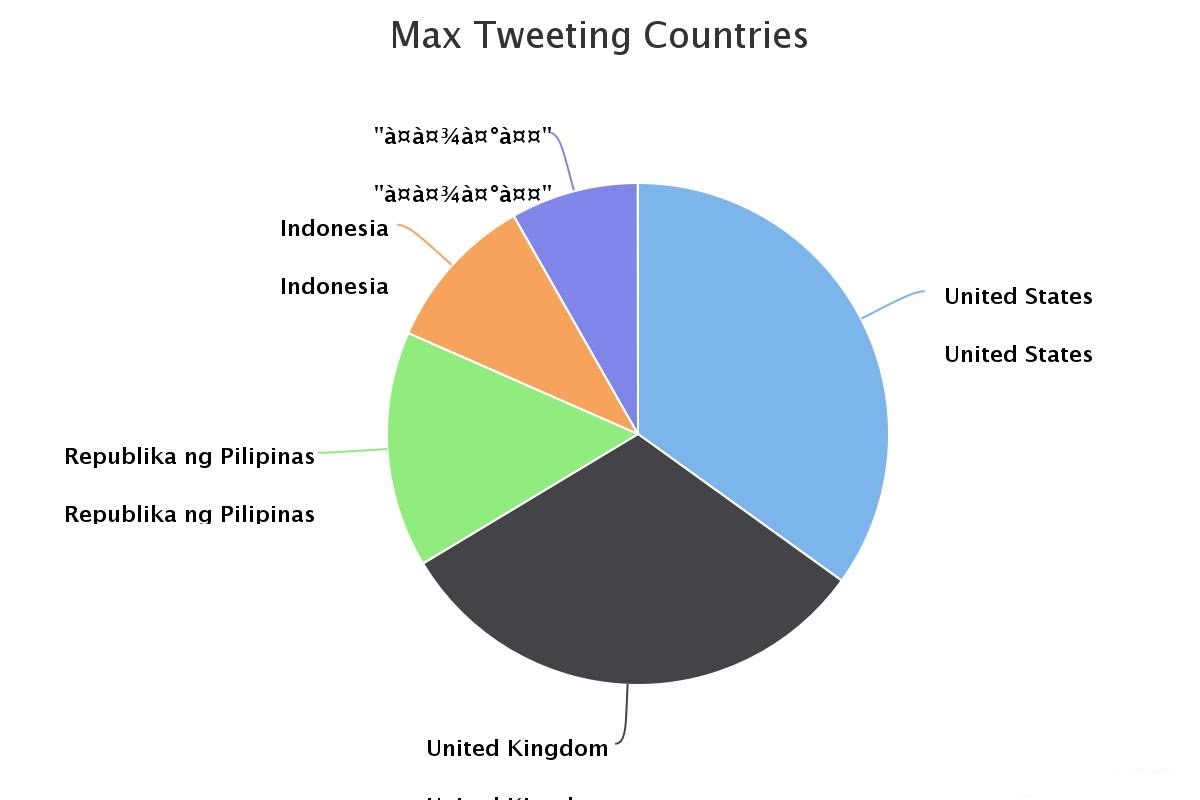
Selected “**place.country**” and counted places and used GROUP by and ORDER By clause to show top 5 countries who tweet most about travelling.

**Query:**

**Val test = sqlContext.sql(SELECT place.country,COUNT(\*) AS country\_count from Tweets WHERE place.country is not null GROUP by place.country order by country\_count desc limit 5 ").toJSON.coalesce(1).saveAsTextFile("/Users/Desktop/Q4.csv")**

**Visualization:**

We have converted data in output file to CVS format as High charts accepts CSV data.



**Query 5: Select the top famous hashtags**

**Explanation:**

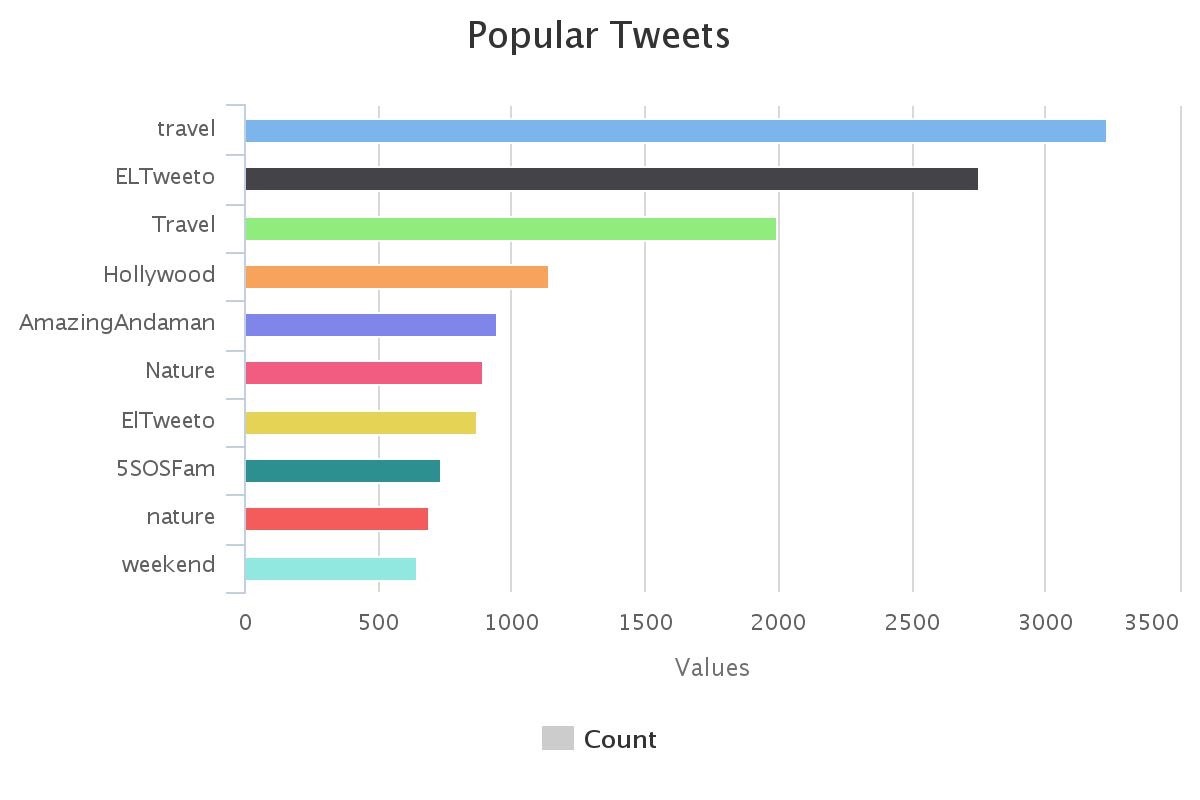
Selected data from column entities. Hashtags[0].text and counted them using count used GROUP BY and ORDER BY clause to display top 10 hashtags. Output is stored in output file Q5.csv

**Query:**

**Val test = sqlContext.sql ("SELECT entities.hashtags[0].text, count(entities.hashtags[0].text) as famous\_tags FROM Tweets group by entities.hashtags[0].text order by famous\_tags desc limit 10").toJSON.coalesce(1).saveAsTextFile("/Users/Desktop/Q5.csv")**

**Visualization:**

We have converted data in output file to CVS format as High charts accepts CSV data.



**Query 6: Based on the Time Zone**

**Explanation:** We have used three queries to achieve this task.

1. In the first query first we selected “**user.time\_zone**” and counted it and used GROUP BY and ORDER BY clauses to get the time zones in descending format. Stored output in test1 table.
2. In the second query we selected “retweeted\_status.user.time\_zone” and counted it and used GROUP BY and ORDER BY clauses to get retweet time\_zone in descending format. Output is stored on test2 table.
3. In the third query we applied join on two table’s test1 and test2 created in first two queries on condition when time\_zone of both tables are equal. Selected time\_zone, Tweet\_count and retweet\_count and used ORDER BY clause to display Tweet\_count in descending orders. Output is stored in Q6.csv file.

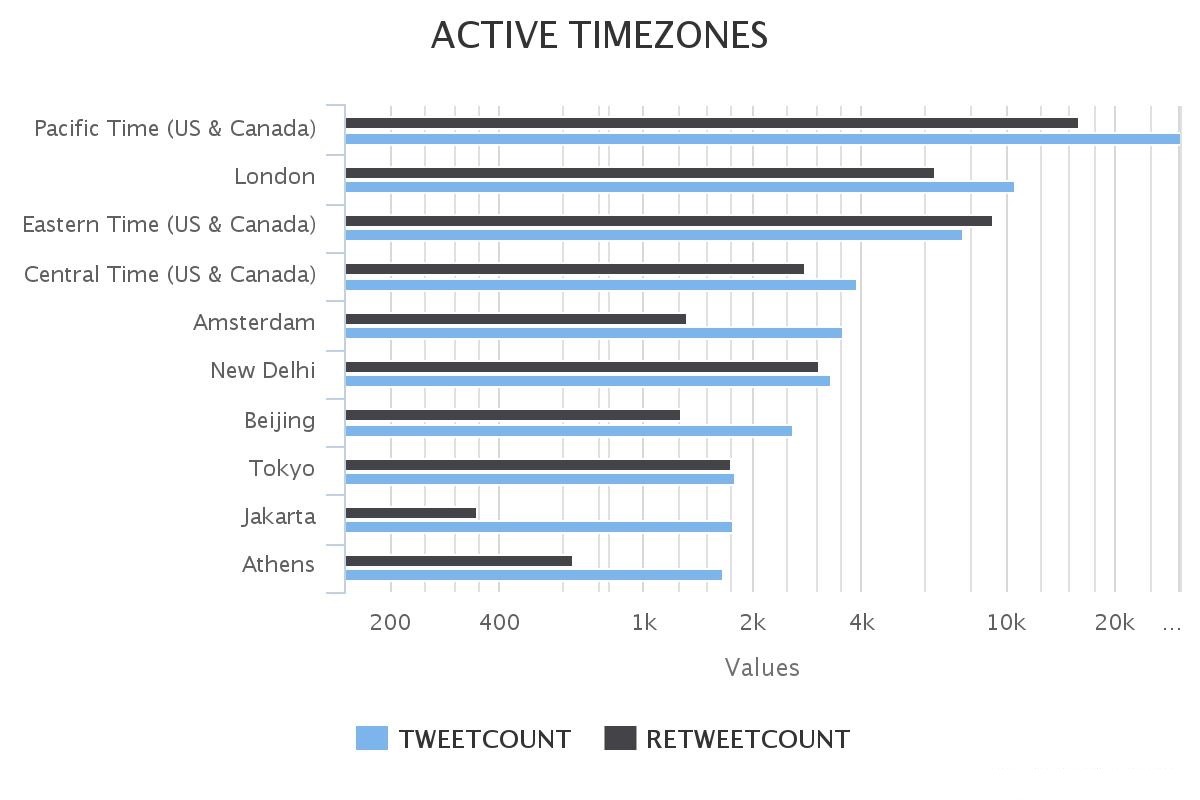
**Query:**

* **val test1 = sqlContext.sql ("select user.time\_zone as time\_zone, count(\*) as Tweet\_count from Tweets where user.time\_zone is not null group by user.time\_zone order by Tweet\_countdesc")**
* **test1.registerTempTable("test1")**

* **val test2 = sqlContext.sql("select retweeted\_status.user.time\_zone as time\_zone, count(\*) as Retweet\_count from tweets where retweeted\_status.user.time\_zone is not null group by retweeted\_status.user.time\_zone order by Retweet\_countdesc")**
* **test2.registerTempTable("test2")**
* **val Query6 = sqlContext.sql("select test1.time\_zone, test1.Tweet\_count, test2.Retweet\_count from x1 inner join test2 on test1.time\_zone = test2.time\_zone order by test1.Tweet\_count desc"**)**.toJSON.coalesce(1).saveAsTextFile("/Users/Desktop/Q6.csv")**

**Visualization:**

We have converted data in output file to CVS format as High charts accepts CSV data.



**Query 7: Query to find celebrity accounts.**

**Explanation:**

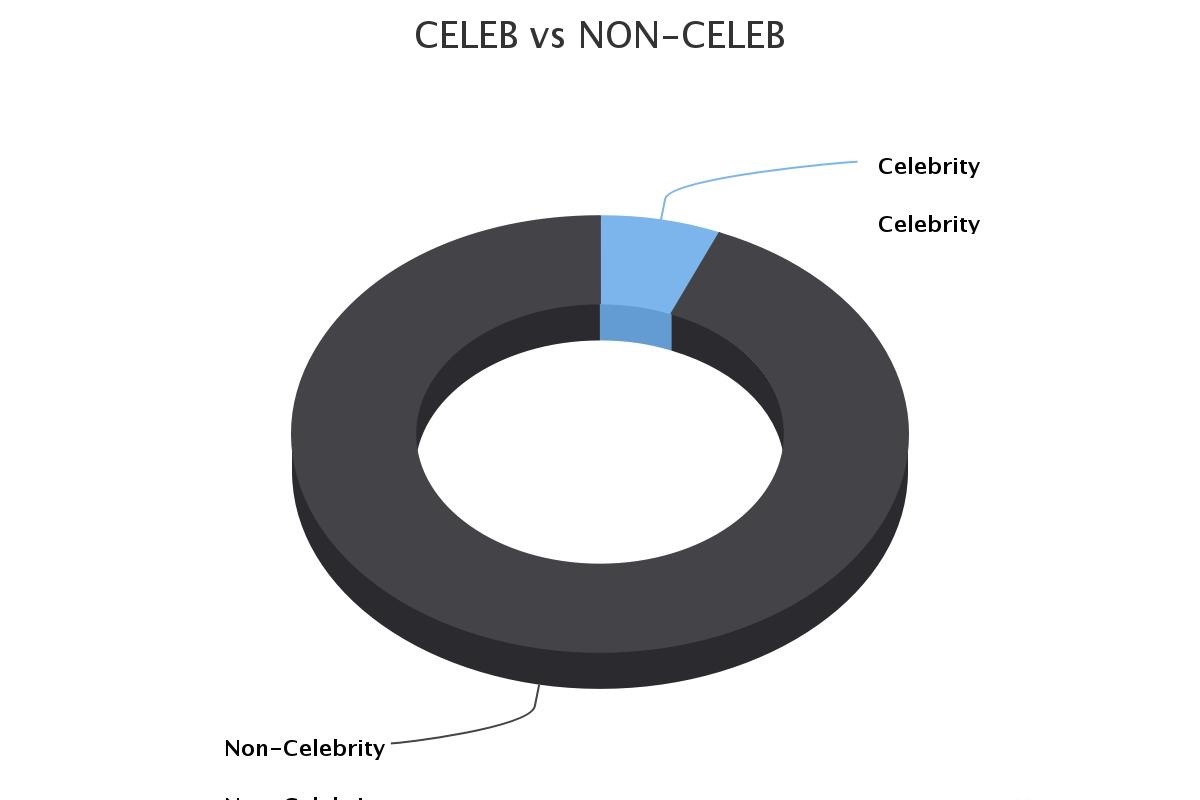
In this query we have selected “**user.verified**” and counted distinct verified accounts using distinct keyword and used GROUP BY clause to show count of celebrity accounts. Output is stored in Q7.csv file

**Query:**

**val test1 = sqlContext.sql (“select user.verified, count(distinct user.id)as count from Tweets where user.verified is not null group by user.verified”) .toJSON.coalesce(1).saveAsTextFile("/Users/Desktop/Q7.csv")**

**Visualization:**

We have converted data in output file to CVS format as High charts accepts CSV data.



**Query 8: Based on the Favorite mode of Transportation**

**Explanation:**

We have used five queries to complete this task.

1. In the first query we have selected those values from column text where keywords “bus, cruise, flight, train and driving are found and also selected “**user.location”** and stored the output into a table TRAVELBY.
2. In the second query we selected the above keywords from text column and stored them as same keywords (in caps) under column travelling and stored the output in table test2.
3. In the third query we selected the travelling mode (represented by travelling column) and location and counted occurrence of each travelling and location pair, used clauses GROUP BY and ORDER BY clauses to generate the output in descending order. Stored the result in test3 table.
4. In the fourth query selected maximum count of a value from travelling column and used GROUP BY clause to show data by travelling. Stored output in table test4.
5. In the 5th query applied join on two table’s test3, test4 where travelling columns and count column values are equal. Selected travelling and count information and stored the result in output table Q8.csv.

**Query:**

* **val test = sqlContext.sql ("SELECT text, user.location from Tweets where text like '%bus%' or text like '%cruise%' or text like '%flight%' or text like '%train%' or text like '%driving%'")**
* **test.registerTempTable("TRAVELBY")**
* **val test1 = sqlContext.sql(**

**"SELECT CASE WHEN text like '%bus%' THEN 'Bus Travelling'" +**

**"WHEN text like '%cruise%' THEN 'Cruise Travelling'" +**

**"WHEN text like '%flight%' THEN 'Fligh Traveling'" +**

**"WHEN text like '%train%' THEN 'TRain Traveling'" +**

**"WHEN text LIKE '%driving%' THEN 'Own Vehicle'" +**

**"END AS travelling,location from TRAVELBY where text is not null and location is not null ")**

* **test1.registerTempTable("test2")**
* **val test3 = sqlContext.sql("select travelling,location, Count(\*) as Count from test2 group by travelling,location order by Count DESC ")**
* **test3.registerTempTable("test3")**
* **val test4 = sqlContext.sql("select travelling,max(Count) as Count from test3 group by travelling")**
* **test4. registerTempTable("test4")**
* **val test5 = sqlContext.sql("select location,t1.travelling,t1.Count from test3 t1,test4 t2 where t1.travelling=t2.travelling and t1.Count=t2.Count").toJSON.coalesce(1).saveAsTextFile("/Users/Desktop/Q8.csv**

