**### TABLE FOR TWITTER DATASET ###**

*create database twitter\_analysis;*

*show databases;*

*use twitter\_analysis;*

To Delete Table

>DROP DATABASE IF EXISTS twitter CASCADE;

Inserting Local Data into hadoop:

hadoop fs -put /home/rayyan/eclipse/java-oxygen/eclipse/Twitter/twitter.json /user/hive/warehouse

**Add jar to include JSONSerDe in create table**

hive>ADD JAR /usr/local/hive/lib/hive-hcatalog-core-2.3.3.jar;

**Create Table to load external Tweets**

CREATE EXTERNAL TABLE election\_tweets(

id BIGINT,created\_at STRING,source STRING,favorited BOOLEAN,

retweet\_status STRUCT<text:STRING, `user` :STRUCT<screen\_name:STRING,name:STRING>,retweet\_count:INT>,

text STRING,

entities STRUCT<

hashtags:ARRAY<STRUCT<text:STRING>>>,

`user` STRUCT<

screen\_name:STRING,

friends\_count:INT,

followers\_count:INT,

statuses\_count:INT,

verified:BOOLEAN,

utc\_offset:INT,

time\_zone:STRING>,

in\_reply\_to\_screen\_name STRING

)

PARTITIONED BY (datehour INT)

ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'

LOCATION '/user//hduser/hive/databases/';

hive>load data inpath '/user/hduser/flume/FlumeData.\*' into TABLE election\_tweets;

**Query: Top 12 Hashtags**

insert overwrite directory '/user/hive/warehouse/'

SELECT

LOWER(hashtags.text),

COUNT(\*) AS total\_count

FROM election\_tweets

LATERAL VIEW EXPLODE(entities.hashtags) t1

AS hashtags

GROUP BY LOWER(hashtags.text)

ORDER BY total\_count DESC

LIMIT 12;

**Clean Results:**

SELECT word, count(1) as wcount FROM election\_tweets

LATERAL VIEW explode(split(text,' ')) text\_ex as word

WHERE word LIKE '#SoniaGandi|#soniagandhi|#RahulGanghi|#Modi|#BJP|#Kejriwal|#AAP|NarendraModi|#Indian|#RSS'

GROUP BY word

ORDER BY wcount DESC LIMIT 10;

**Results:**

#SoniaGandhi 211

#Indian 142

#RahulGandhiInMumbai 139

#BJP 112

#Modi 106

#RahulGandhi 53

@INCIndia 40

#NarendraModi 38

#soniagandhi 29

#RSS 18

#RahulGandhi 15

**Create Sentiment Dictionary**

CREATE EXTERNAL TABLE DICTIONARY(

word string,

polarity string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

STORED AS TEXTFILE

LOCATION '/user//hduser/hive/databases/';

**Load Data of Dictionary:**

load data inpath '/user/hive/dictionary' INTO TABLE dictionary;

**Create Stop Words:**

CREATE EXTERNAL TABLE STOPWORDS(

word string,

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\n'

STORED AS TEXTFILE

LOCATION '/user//hduser/hive/databases/';

**Load Data of StopWords:**

load data inpath '/user/hive/dictionary/stopwords.txt' INTO TABLE stopwords;

**Create Time Zones**

CREATE EXTERNAL TABLE TIME\_ZONE\_MAP(

time\_zone string,

country string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

STORED AS TEXTFILE

LOCATION '/user//hduser/hive/databases/';

**Load Data of TimeZone:**

load data inpath '/user/hive/timezone.txt' INTO TABLE time\_zone\_map;

**Simple Tweets:**

CREATE VIEW tweets\_simple AS

> SELECT

> id,

> `user`.screen\_name,

> source,

> retweet\_status.retweet\_count,

> entities.hashtags,

> cast ( from\_unixtime( unix\_timestamp(concat( '2016 ', substring(created\_at,5,15)), 'yyyy MMM dd hh:mm:ss')) as timestamp) ts,

> text,

> `user`.statuses\_count,

> `user`.friends\_count,

> `user`.followers\_count,

> `user`.time\_zone

> from election\_tweets;

**Remove Spams:**

CREATE VIEW tweets\_rem\_spam AS

> SELECT \*

> FROM tweets\_simple

> WHERE

> statuses\_count > 50 AND

> friends\_count/followers\_count > 0.01 AND

> length(text) > 10 AND

> size(hashtags) < 10;

**Clean the Tweets**

hive> CREATE VIEW tweets\_clean AS

> SELECT

> id,

> t.screen\_name,

> source,

> retweet\_count,

> t.hashtags,

> ts,

> text,

> m.country

> FROM tweets\_simple t LEFT OUTER JOIN time\_zone\_map m ON t.time\_zone = m.time\_zone;

**Create Views:**

create view l1 as select id, words from election\_tweets lateral view explode(sentences(lower(text))) dummy as words;

create view l2 as select id, word from l1 lateral view explode( words ) dummy as word;

create view l\_clean as select \* from l2 where l2.word not in (select \* from stopwords) and regexp\_replace(l2.word,'[^a-zA-Z0-9]','')!='';

**Converting Polarity into Numerical Values;**

create view l3 as select

> id,

> l2.word,

> case d.polarity

> when '-1' then -1

> when '-2' then -2

> when '-3' then -3

> when '-4' then -4

> when '-5' then -5

> when '5' then 5

> when '4' then 4

> when '3' then 3

> when '2' then 2

> when '1' then 1

> when '0' then 0

> else 0 end as polarity

> from l2 left outer join dictionary d on l2.word = d.word;

**Classification into positive,negitive and neutral**

create table tweets\_sentiment as select

> id,

> case

> when avg(polarity) > 0 then 'positive'

> when avg(polarity) < 0 then 'negitive'

> else 'neutral' end as sentiment

> from l3 group by id;

**Count Positive, Negitive and Neutral**

create view l8 as select sentiment, count(id) from tweets\_sentiment ts group by sentiment;

**Results:**

negitive 3004

positive 5324;

neutral 5975;

Links

<http://blog.cloudera.com/blog/2012/11/analyzing-twitter-data-with-hadoop-part-3-querying-semi-structured-data-with-hive/>

//2 Link

<https://acadgild.com/blog/sentiment-analysis-on-tweets-with-apache-hive-using-afinn-dictionary>

//3rd Link

<http://blog.cloudera.com/blog/2012/09/analyzing-twitter-data-with-hadoop/>

//4th Link

<http://www.havlena.net/en/business-analytics-intelligence/an-example-how-to-collect-analyze-and-visualize-social-media-data-with-jaql-and-hadoop/>

//avro hive

<http://hadooptutorial.info/twitter-data-analysis-using-hadoop-flume/>