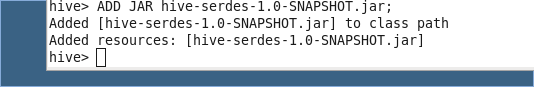
Twitter Data Analysis

I will be using JSON SerDe Interface to read the tweets given in JSON file format.

Jar needs to be added every time before using SerDe Interface. Commands are as belows:

ADD JAR hive-serdes-1.0-SNAPSHOT.jar;



Now before importing the data, we will define the JSON schema in the Hive table schema, making it much easier to issue queries against. Below is the create table command used to create a table in Hive:

CREATE TABLE complex\_json

(retweet\_count int,

created\_at string,

text string,

id string,

source string,

in\_reply\_to\_screen\_name string,

user struct<location : string,

id : string,

id\_str : string,

name:string,

screen\_name:string,

geo\_enabled:string,

lang:string,

protected:string,

verified:string,

followers\_count:string,

friends\_count:string,

listed\_count:string,

favourites\_count:string,

statuses\_count:string,

profile\_background\_color : string> ,

contributors string,

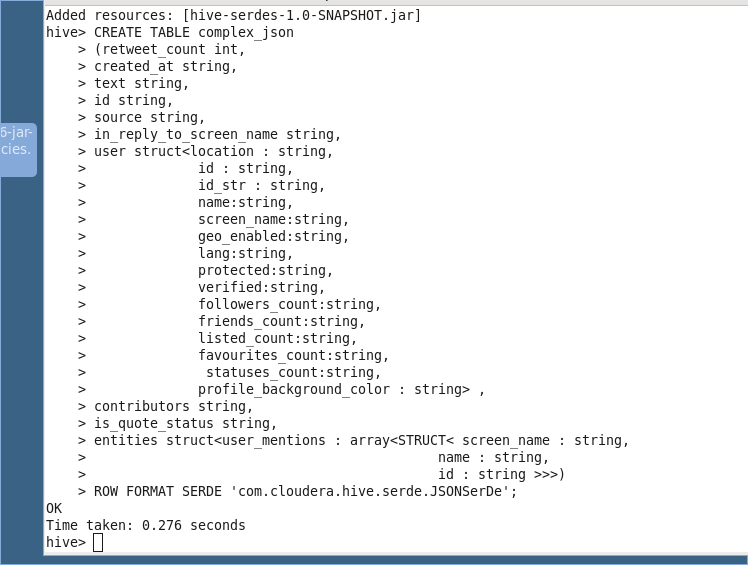
is\_quote\_status string,

entities struct<user\_mentions : array<STRUCT< screen\_name : string,

name : string,

id : string >>>)

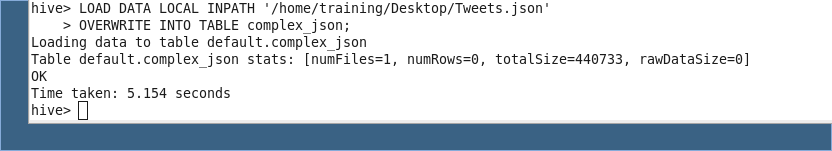
ROW FORMAT SERDE ‘com.cloudera.hive.serde.JSONSerDe’



We have created the schema in Hive. Now we will load the data from the file ‘Tweets.json’ stored on Desktop to the Schema.

Command to load the data into the hive schema is as below:

LOAD DATA LOCAL INPATH ‘/home/training/Desktop/Tweets.json’OVERWRITE INTO TABLE complex\_json;



Now we will check whether the data is successfully loaded or not using

Select \* from complex\_json LIMIT 5;

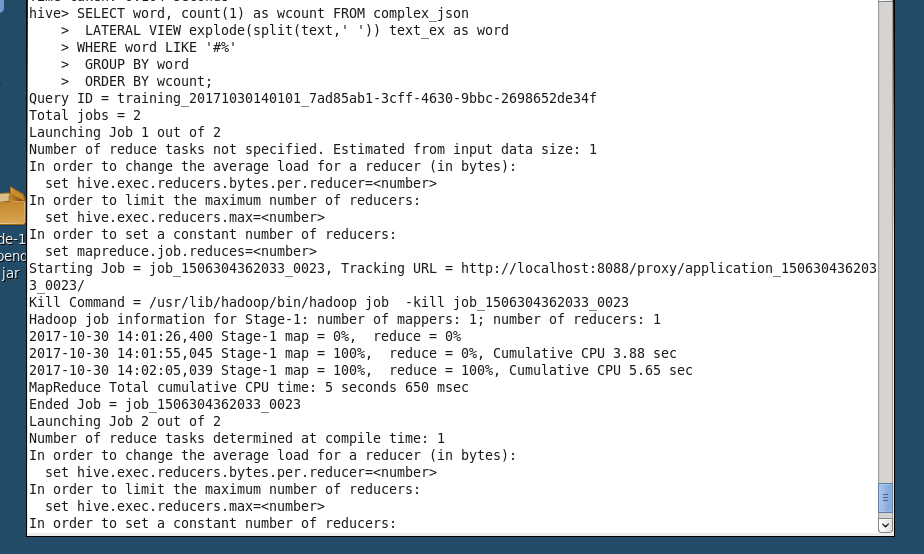


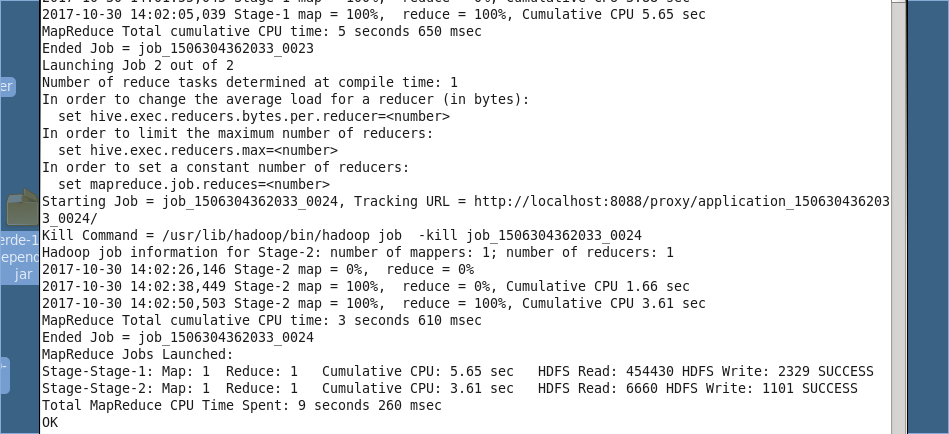
AS we can see the data is loaded.

hashtags tweeted and how many times are they used

Query used :

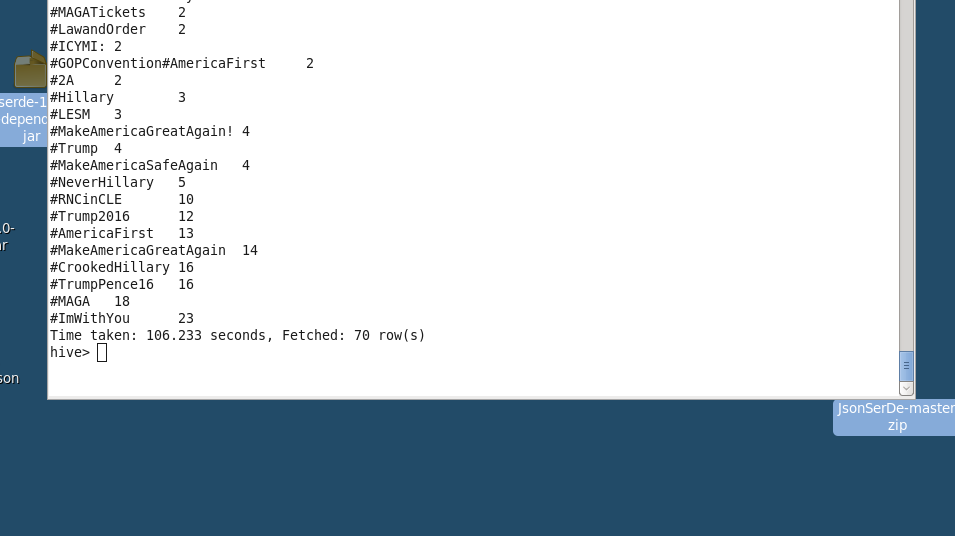
SELECT word, count(1) as wcount FROM complex\_json LATERAL VIEW explode(split(text,' '))text\_ex as word WHERE word LIKE '#%' GROUP BY word, ORDER BY wcount;





OUTPUT :





Q. What is the most trending hashtag in a day and how many times are they tweeted? [Note: day should be in the format ‘yyyy-mm-dd’]

A.)

First we will convert the date the date in the required format using unix\_timestamp and from\_unixtime. We will also create a new table with text and date before querying out the trending hashtags.

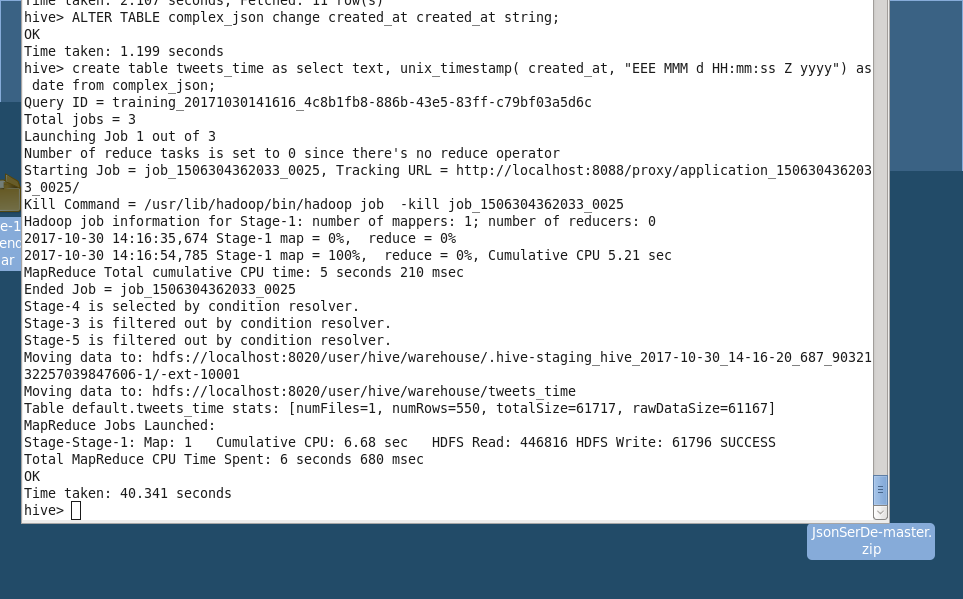
Query:

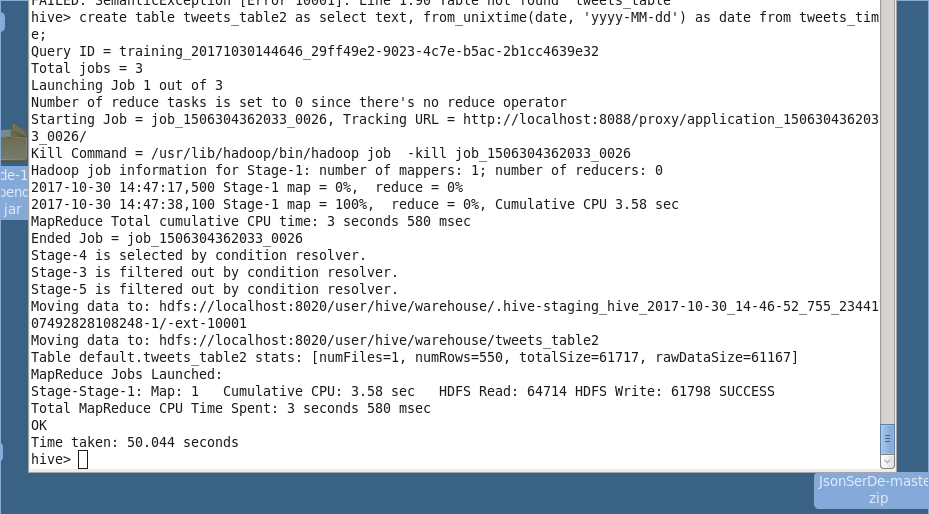
CREATE TABLE tweets\_time as select text, unix\_timestamp(created\_at,”EEE MMM d HH:mm:ss Z yyyy”) as date from complex\_json;

Now we will convert the date

CREATE TABLE tweets\_table2 AS select text, from\_unixtime(date,’yyyy-MM-dd’) AS date from tweets\_time;

Following are the 2 screenshots for the above queries.





Now we will query out the most trending hashtags using the following query.

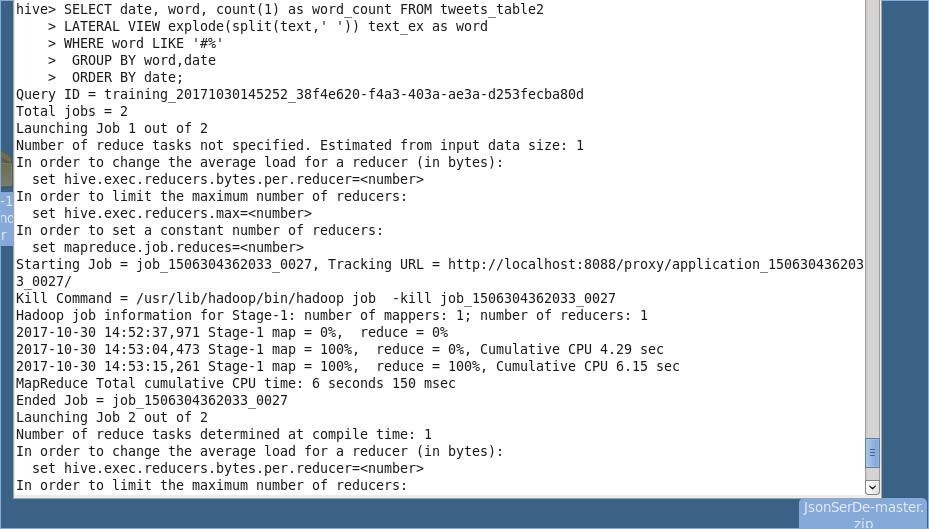
SELECT date,word, count(1) as word\_count FROM tweets\_table2

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

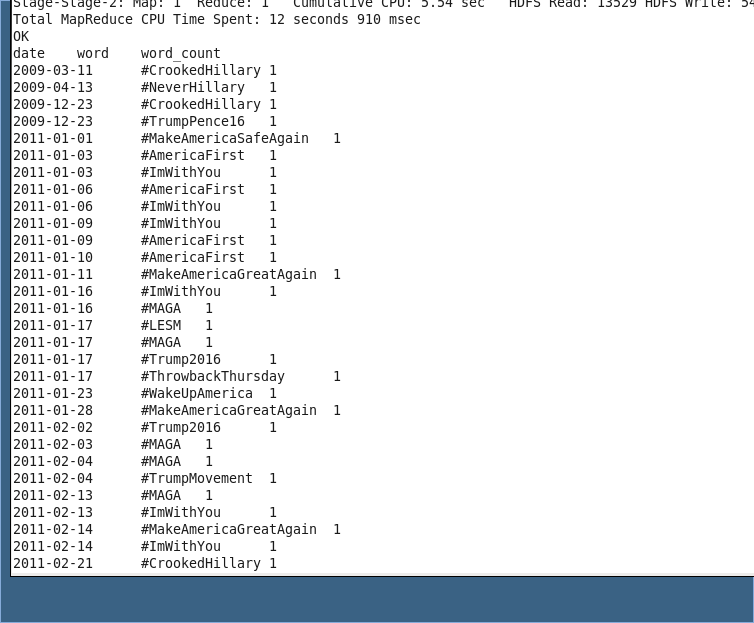
WHERE word LIKE '#%'

GROUP BY word,date

ORDER BY date;



OUTPUT



Q.1)Which state users are most active, and how many tweets are posted by them?

A.1)

Query :

SELECT user.location, COUNT(DISTINCT text) AS tweets

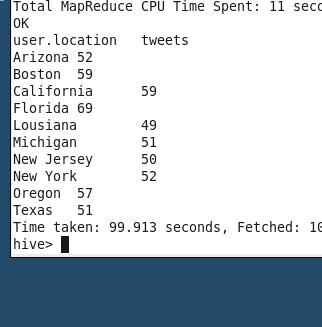
FROM complex\_json

GROUP BY user.location

ORDER BY user.location;



OUTPUT:

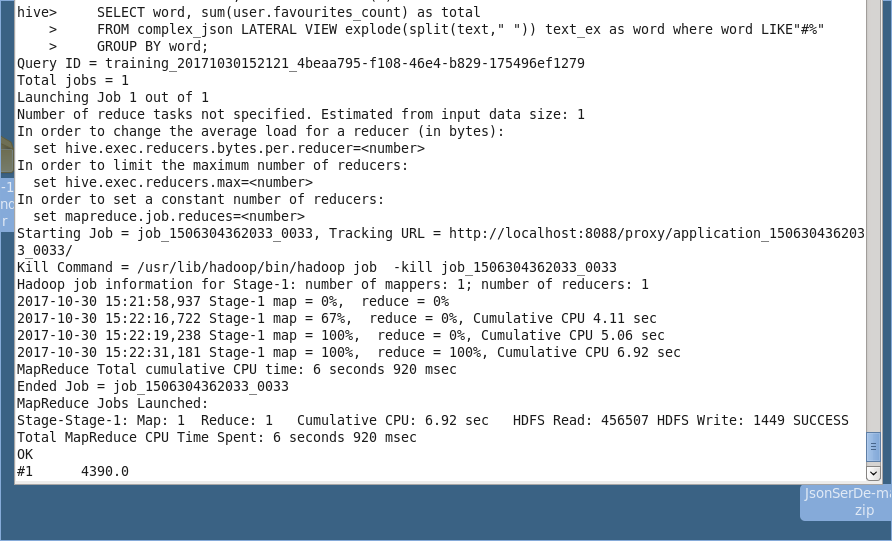


Q.1) What is the total number of favorites received for each hashtag?

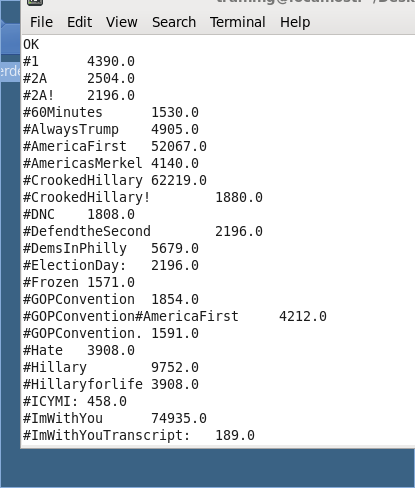
A.1) Query :

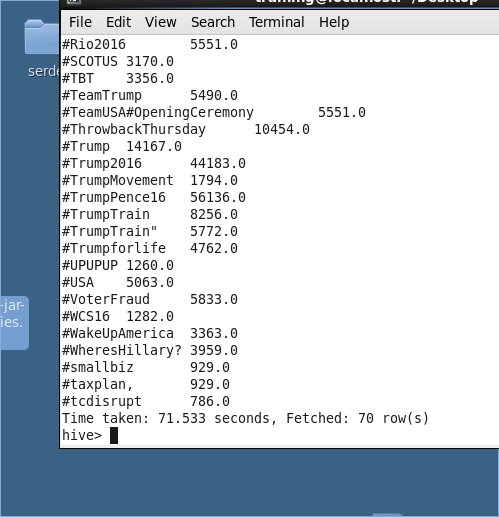
SELECT word, sum(user.favourites\_count) as total

FROM complex\_json LATERAL VIEW explode(split(text," ")) text\_ex as word where word LIKE"#%" GROUP BY word;



OUTPUT





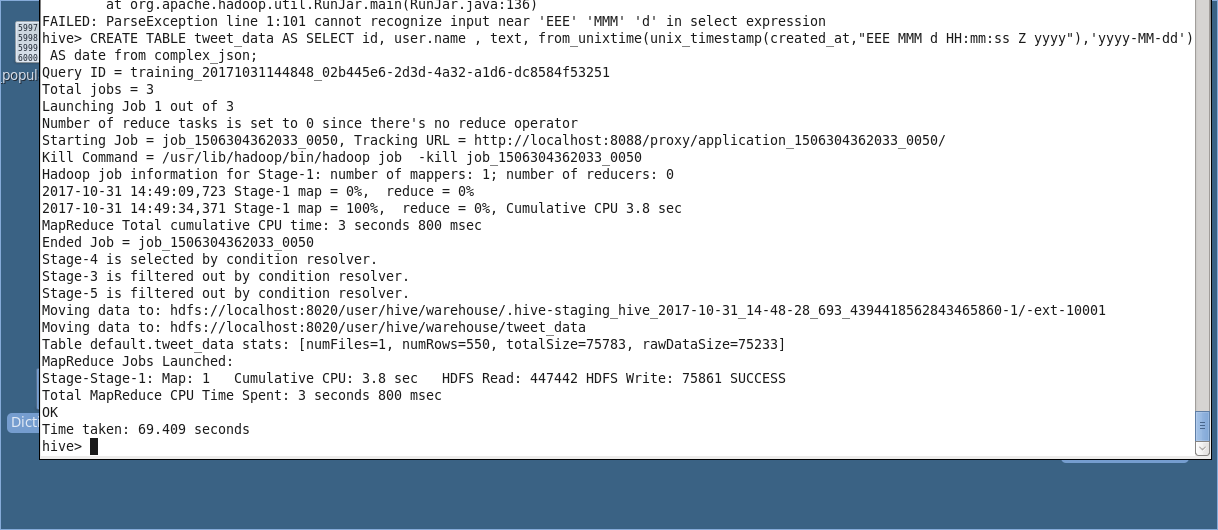
Q.) Does each tweet have a positive or negative sentiment?

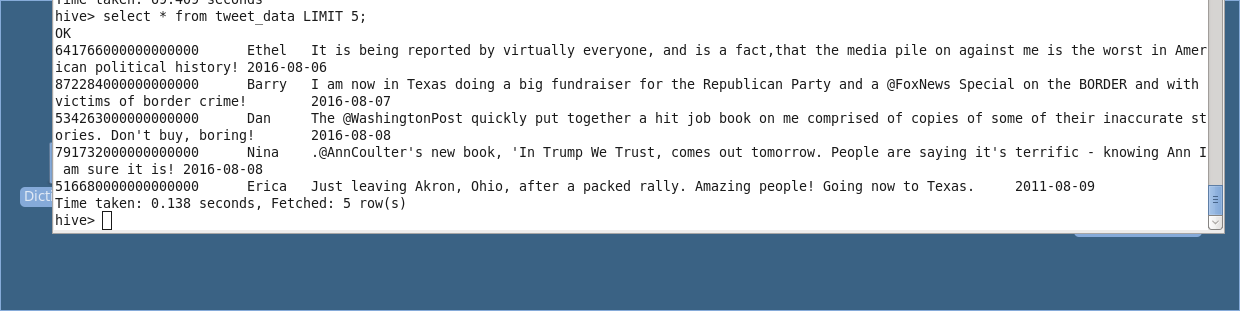
A.)

We will create a new table with user name, id and date from complex\_json before joining the data with the score table

Query:

CREATE TABLE tweet\_data AS SELECT id, user.name , text, from\_unixtime(unix\_timestamp(created\_at,”EEE MMM d HH:mm:ss Z yyyy”),’yyyy-MM-dd’) AS date from complex\_json





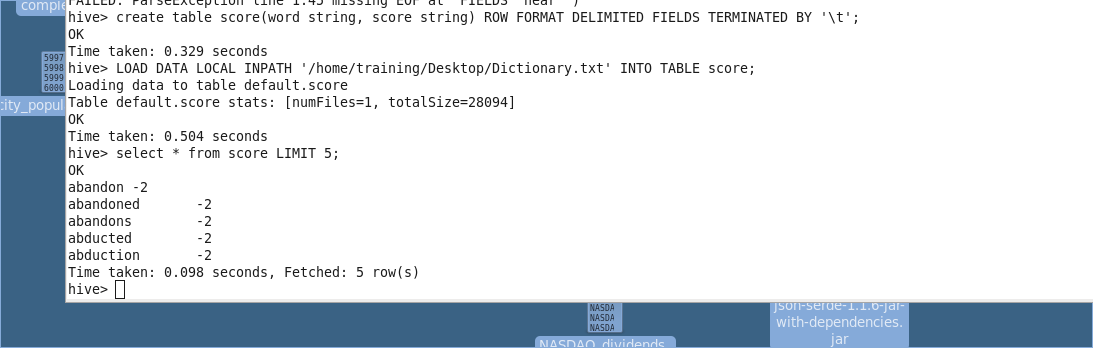
Now we will load the data into the score table. First we need to create the schema for the same.

Query :

CREATE TABLE score (word string, score string);

Query to load the data:

LOAD DATA LOCAL INPATH ‘/home/training/Desktop/Dictionary.txt’ INTO TABLE score;



Now we will create a table where each tweet will be separated into each word of each tweet.

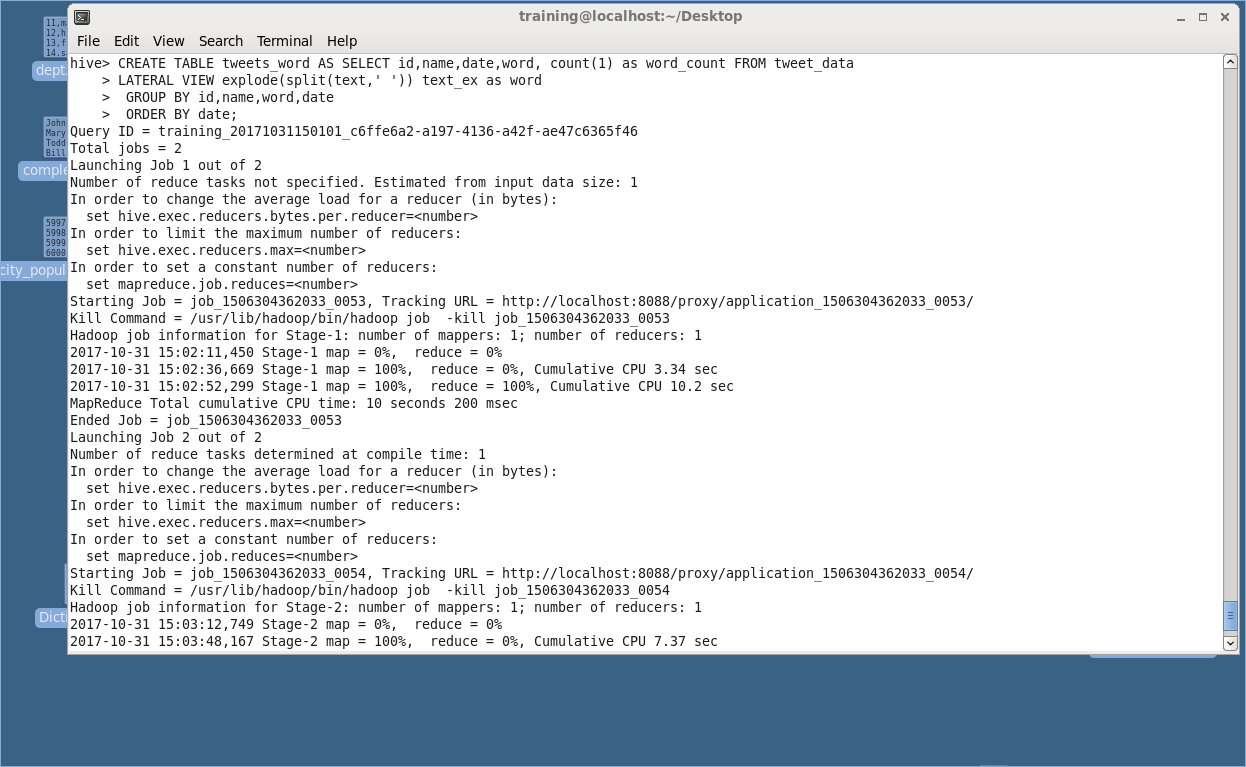
Query :

CREATE TABLE tweets\_word AS SELECT id,name,date,word, count(1) as word\_count FROM tweet\_data

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

GROUP BY id,name,word,date

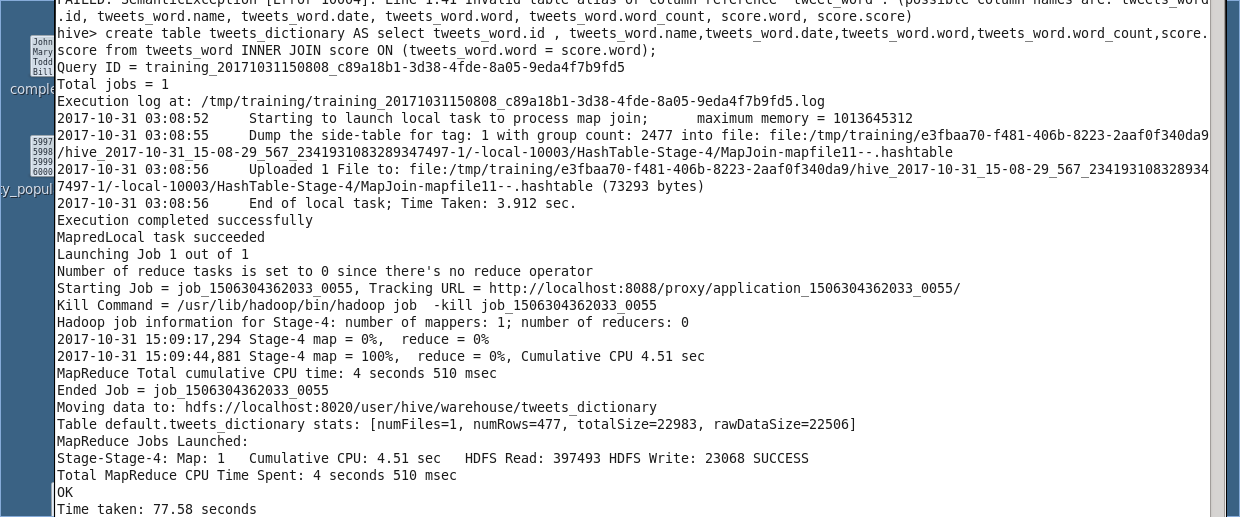
ORDER BY date;



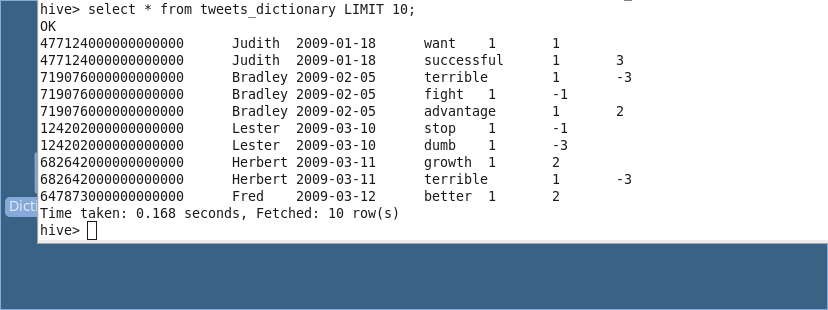
Now we will join the table tweets\_word and score table by word and create a tweets\_dictionary table.

Query:

create table tweets\_dictionary AS select tweets\_word.id , tweets\_word.name,tweets\_word.date,tweets\_word.word,tweets\_word.word\_count,score.score from tweets\_word INNER JOIN score ON (tweets\_word.word = score.word);



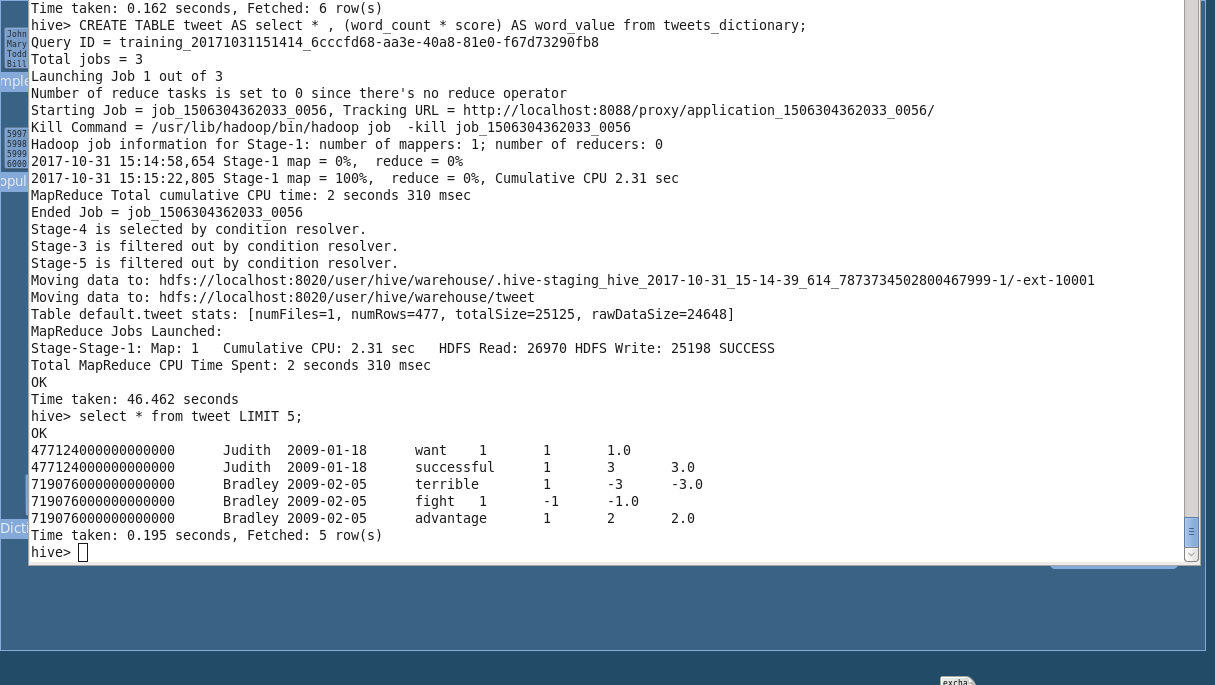
We will get a table with only those words that are present in the score table also.



Now we will multiply the word count with score for each word.

Query :

CREATE TABLE tweet AS select \* , (word\_count \* score) AS word\_value from tweets\_dictionary;



Now we will group the table by id and use the sum function to determine whether each tweet is positive or negative.

Query :

CREATE TABLE tweet\_score AS select id,name,date,SUM(word\_value) AS score from tweet GROUP BY id,name,date;



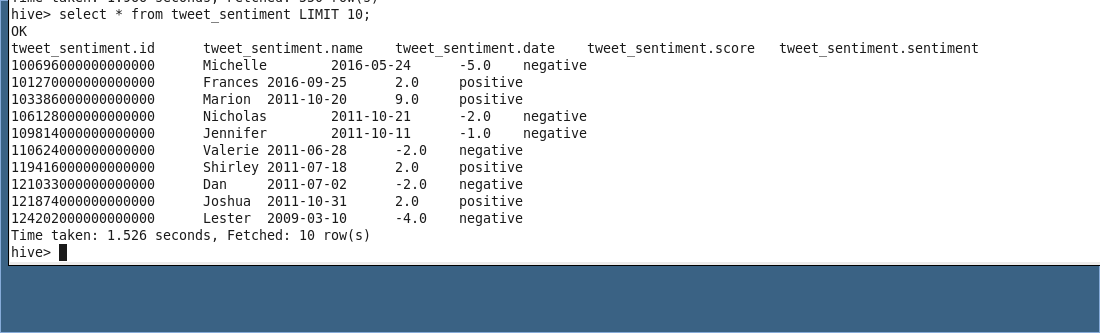
Using when and then, we will create sentiment table which will decide whether the tweet by a specific user id was positive or negative and add a column for the same.

Query :

CREATE TABLE tweet\_sentiment AS select \*, CASE WHEN tweet\_score.score >= 0 then ‘positive’ else ‘negative’ end sentiment from tweet\_score;



OUTPUT:



Q) Do you believe that the process outlined above has issues? If so, briefly explain how the process can be improved.

A)

Yes. The process above can be improved.

We need to clean the data since there are tweets such as #2A and #2A! which apparently must be the same thing. Cleaning the data would have given us different results for the most trending hashtags.

Also, to parse the data we need to first define the schema of the table. Any change in the JSON file format would lead to numerous changes in the table schema and the queries. We can use Avro file format to read columns dynamically.

Query:

select entities.user\_mentions.id, entities.user\_mentions.screen\_name , text , from\_unixtime(unix\_timestamp(created\_at,”EEE MMM d HH:mm:ss Z yyyy”),’yyyy-MM-dd’) AS date , user.location from complex\_json

