Twitter Data Analysis

Assignment 2

Amit Sawant

# Question 1

Create Table

create table tweet2(id bigint,text string,created\_at string,retweet\_count int,user

struct<location:string,id\_str:bigint,name:string,screen\_name:string,followers\_count

:int>,

quoted\_status struct< user : struct<location:string,followers\_count:int,name:string> >

)ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe';

Load json file into table:

load data local inpath 'Desktop/twitter.json' overwrite into table tweet2;

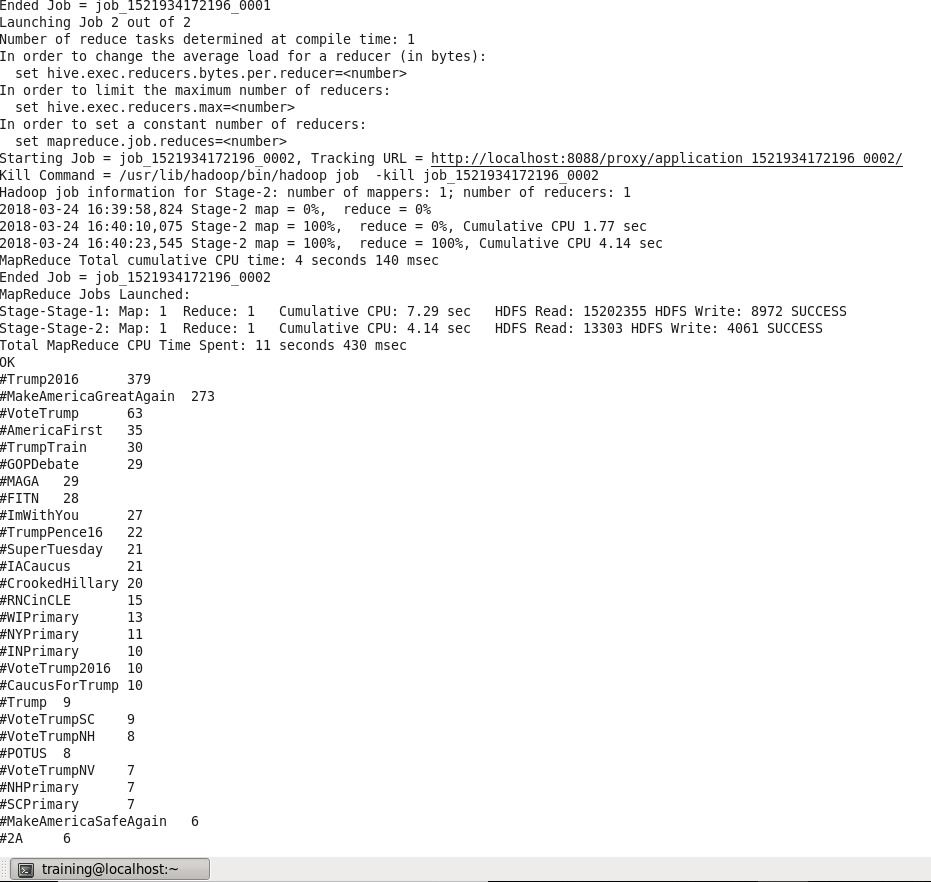
# 1.a)

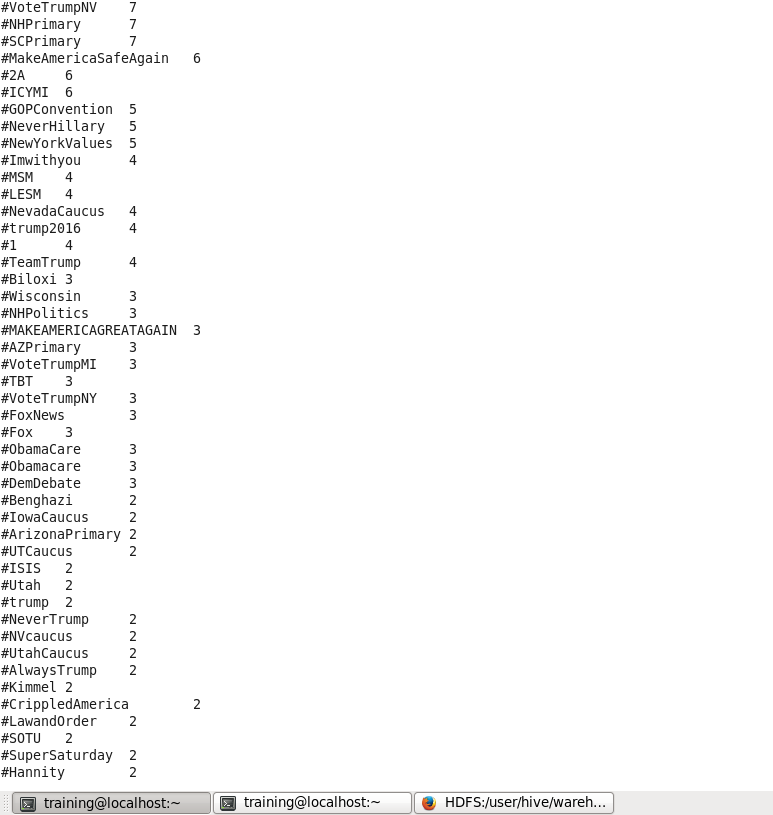
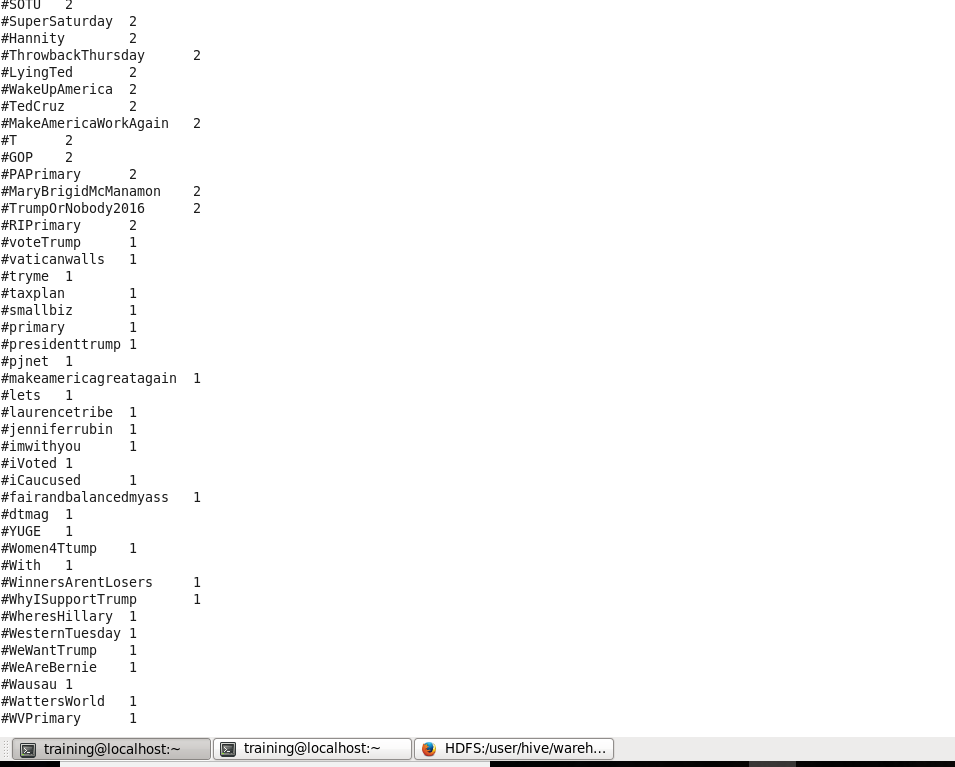
CODE:

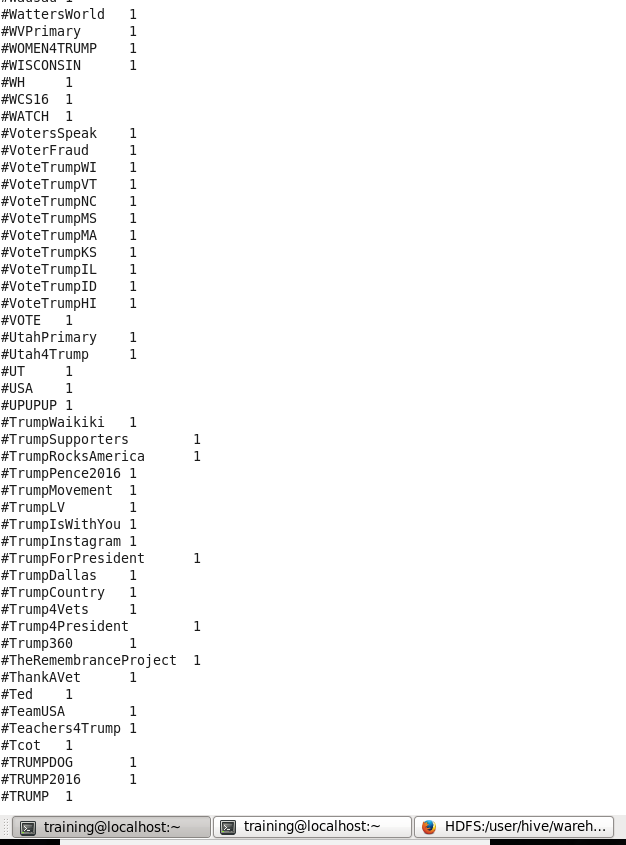
SELECT word, count(1) as wcount from tweet2 LATERAL VIEW

explode(split(regexp\_replace(trim(text),"[^#A-Za-z0-9]"," "), ' ')) text\_ex as word

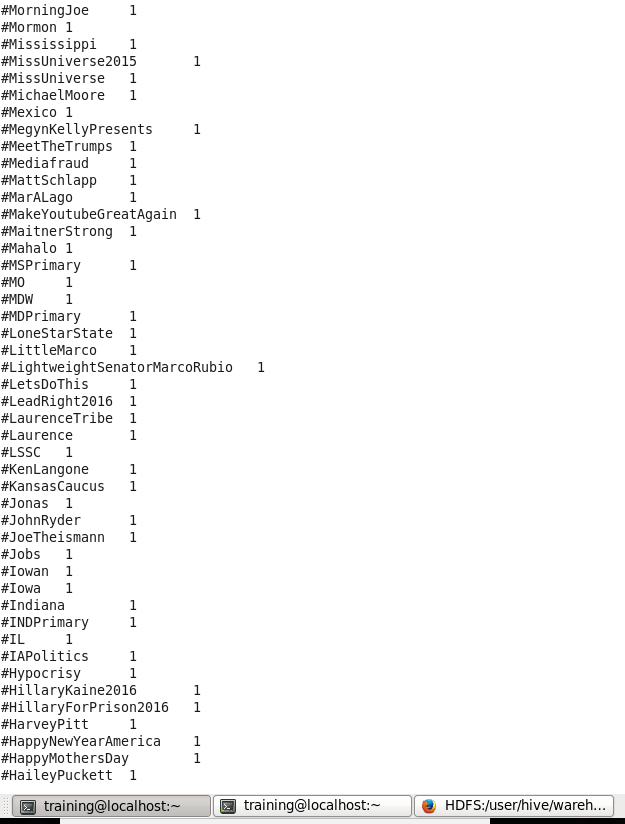
WHERE word rlike "^#[a-zA-Z0-9]+$" GROUP BY word ORDER BY wcount desc;

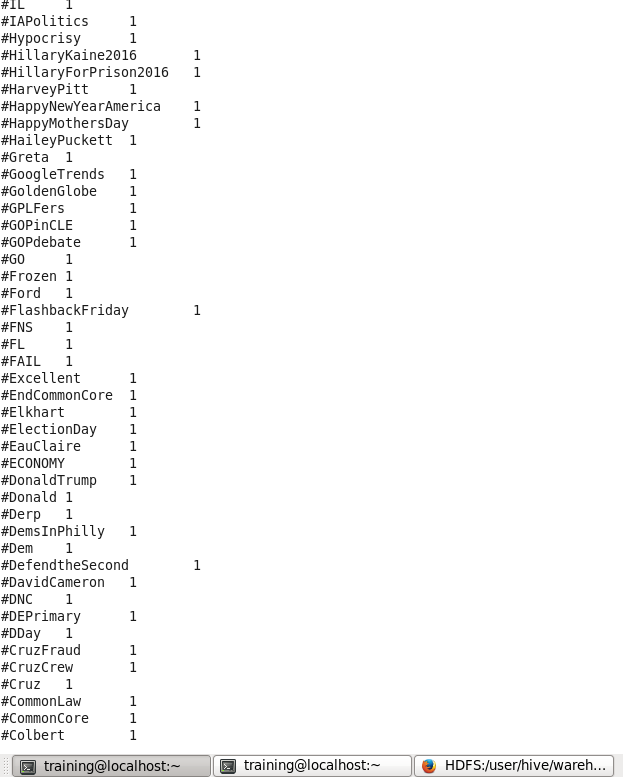


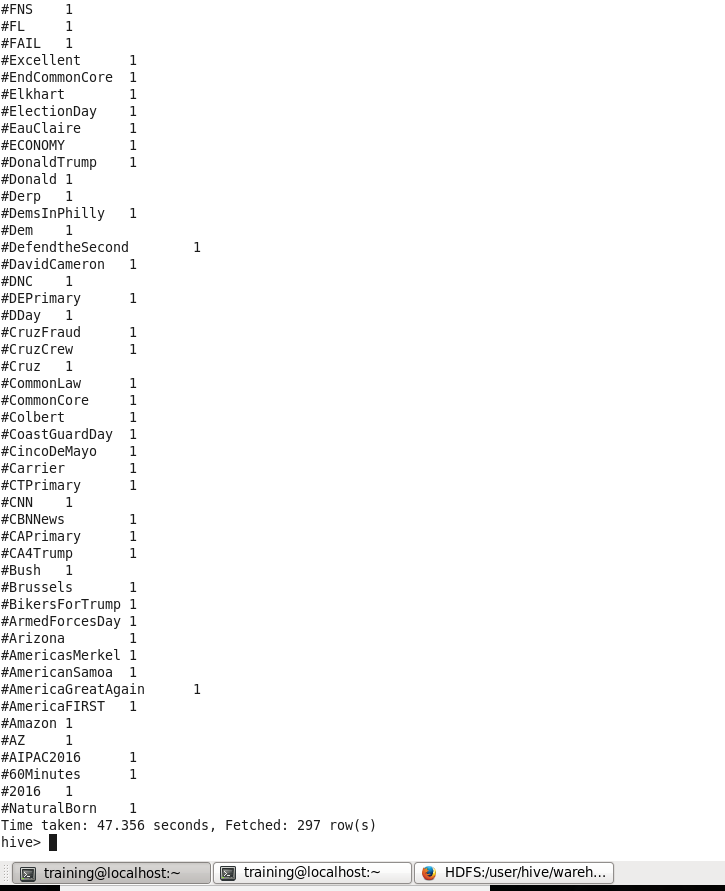








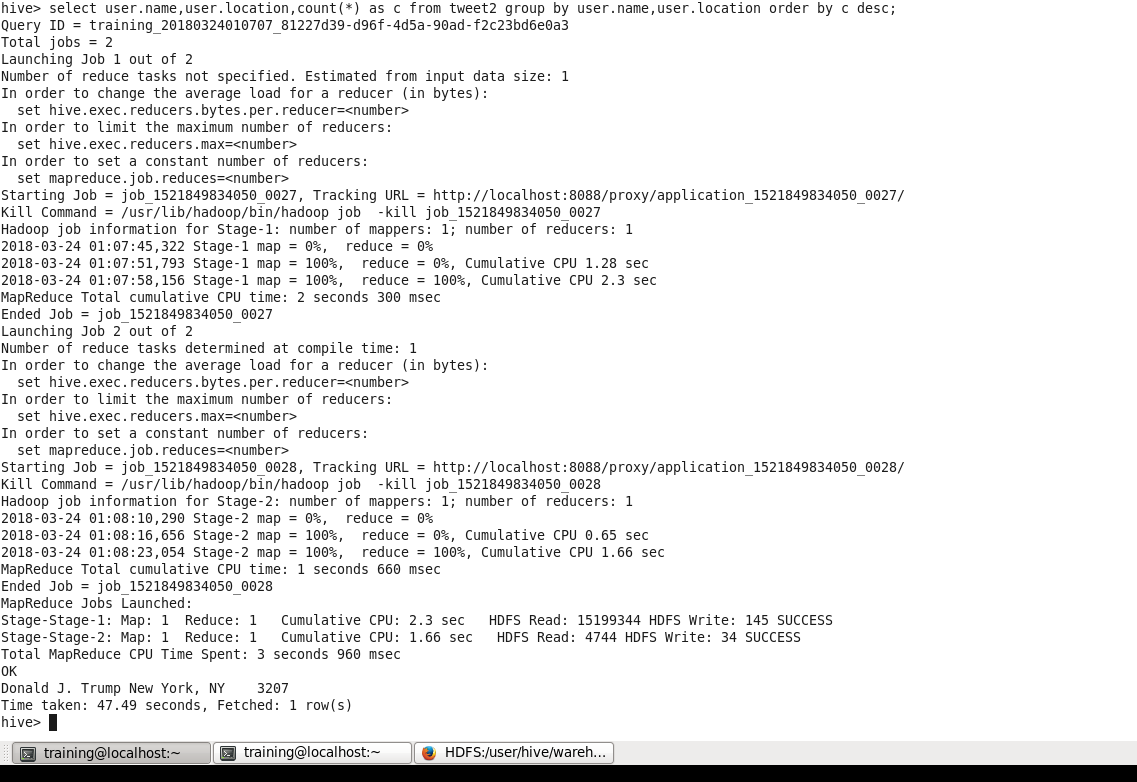




# 1.b)

CODE:

select user.name,user.location,count(\*) as c from tweet2 group by user.name,user.location order by c desc;

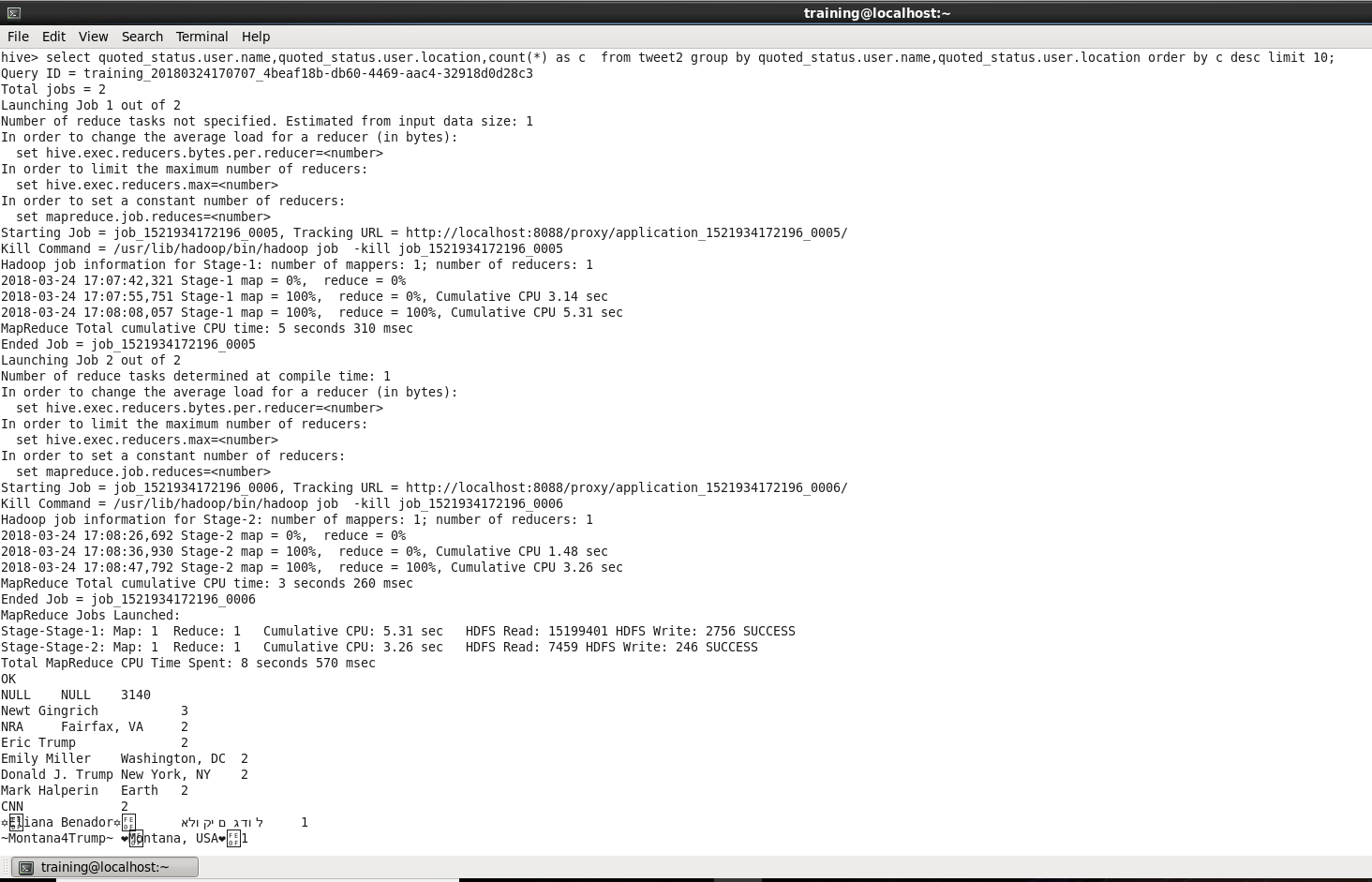


New York has the most active users   
We can see only Donald trump because most of the tweets are from him in the form of either his own tweets or retweets of other users.

Users from other States can be found using the query below

CODE:

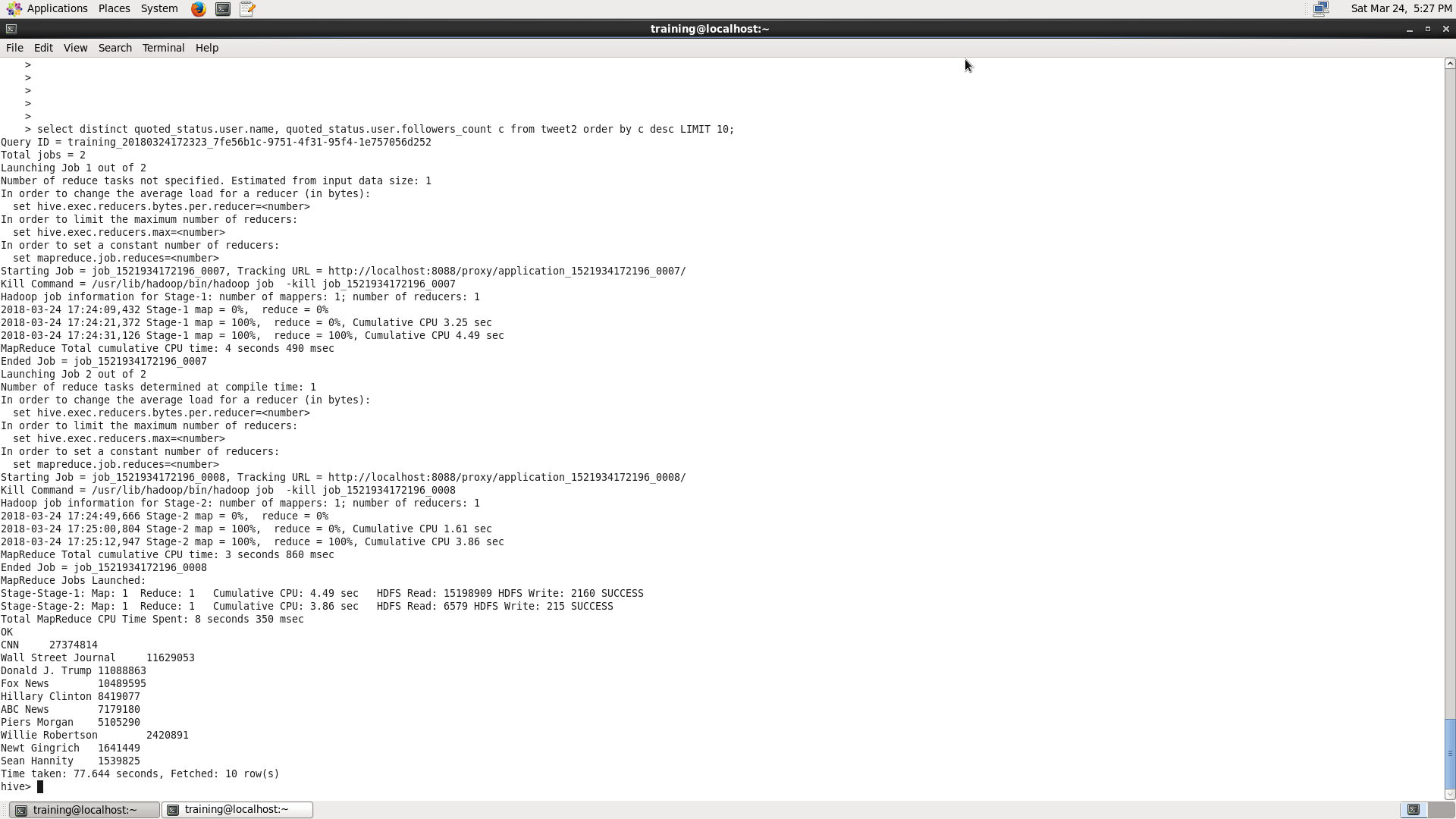
select quoted\_status.user.name,quoted\_status.user.location,count(\*) as c from tweet2 group by quoted\_status.user.name,quoted\_status.user.location order by c desc limit 10;



# 1.c)

CODE:

select distinct quoted\_status.user.name, quoted\_status.user.followers\_count c from tweet2 order by c desc LIMIT 10;



# 1.d)

CODE:

Loading Dictionary.txt in a table

create table dictionary(word string,score int) ROW FORMAT DELIMITED

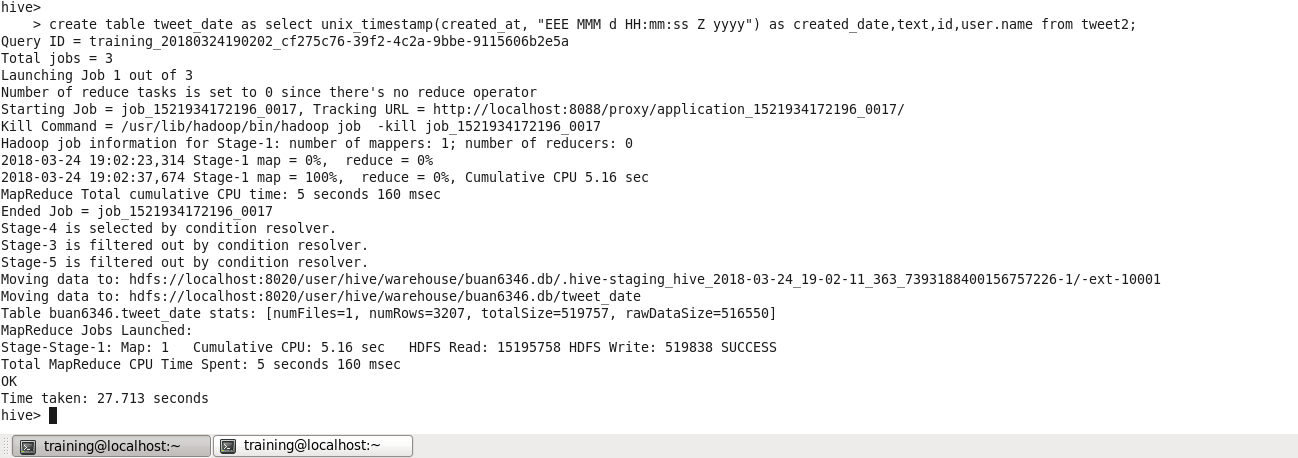
FIELDS TERMINATED BY '\t';

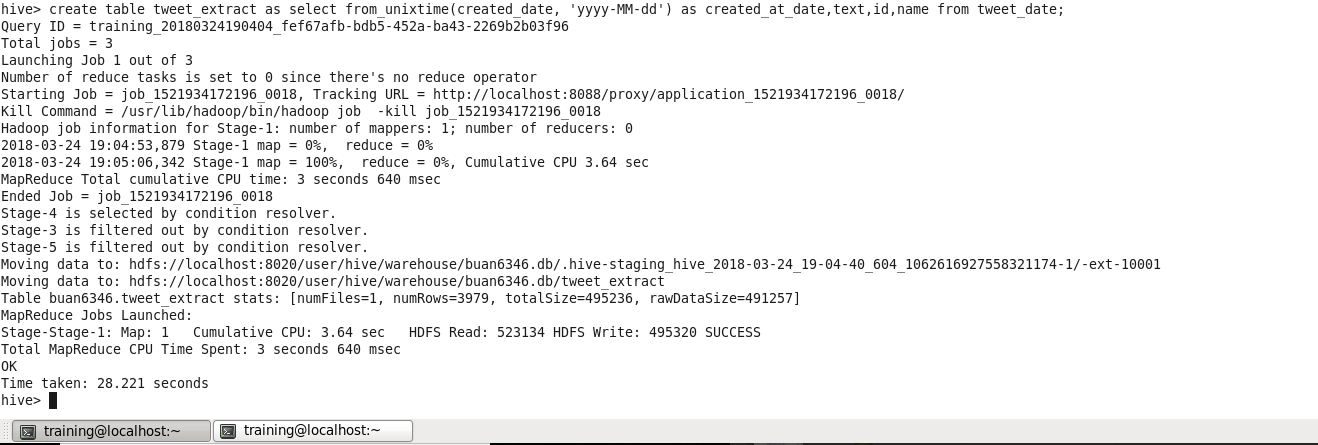
load data local inpath ‘Desktop/Dictionary.txt’ into table dictionary;

Converting created\_at date into YYYY-MM-DD format

create table tweet\_date as select unix\_timestamp(created\_at, "EEE MMM d HH:mm:ss Z yyyy") as created\_date,text,id,user.name from tweet2;

create table tweet\_extract as select from\_unixtime(created\_date, 'yyyy-MM-dd') as created\_at\_date,text,id,name from tweet\_date;

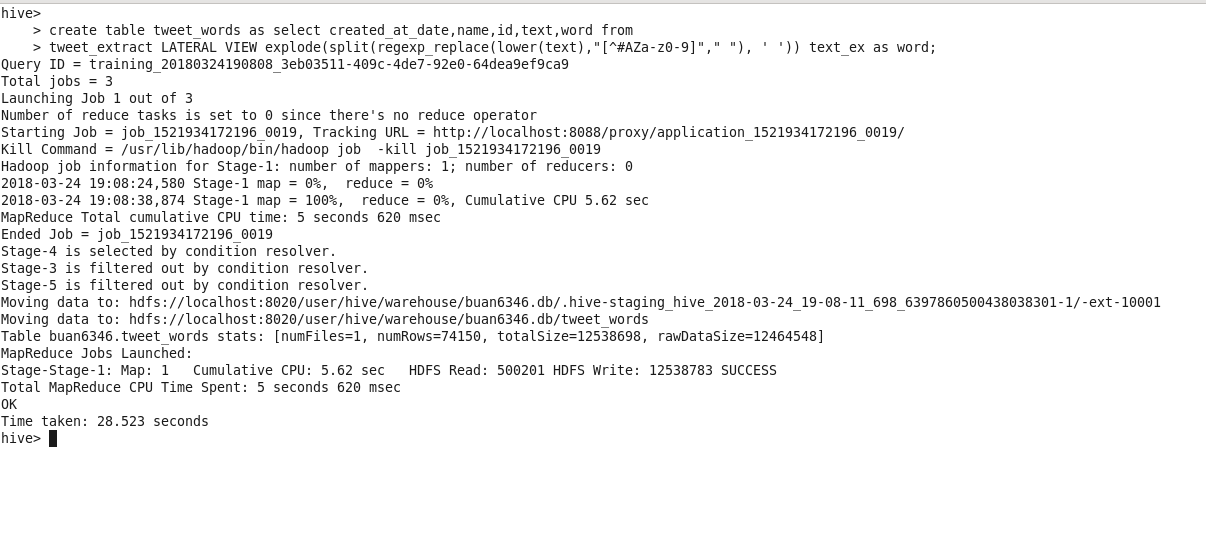




Extracting words from tweet text

create table tweet\_words as select created\_at\_date,name,id,text,word from

tweet\_extract LATERAL VIEW explode(split(regexp\_replace(lower(text),"[^#AZa-z0-9]"," "), ' ')) text\_ex as word;

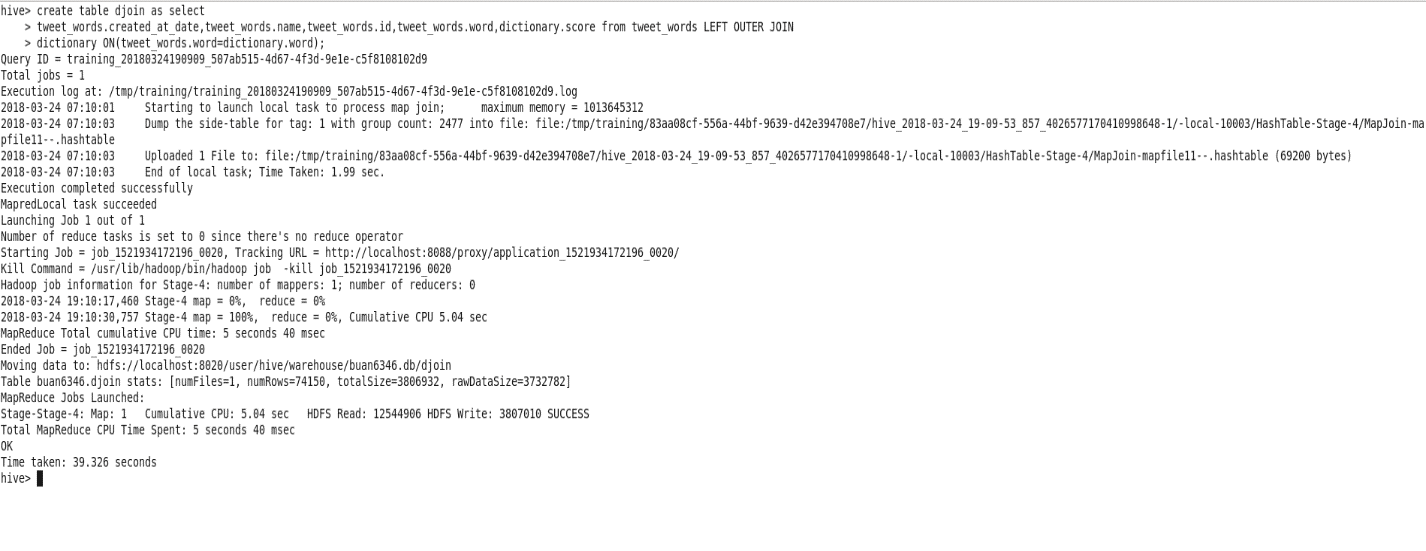


Joining Dictionary and Tweet words

create table djoin as select

tweet\_words.created\_at\_date,tweet\_words.name,tweet\_words.id,tweet\_words.word,dictionary.score from tweet\_words LEFT OUTER JOIN

dictionary ON(tweet\_words.word=dictionary.word);



Sentiment analysis with polarity score

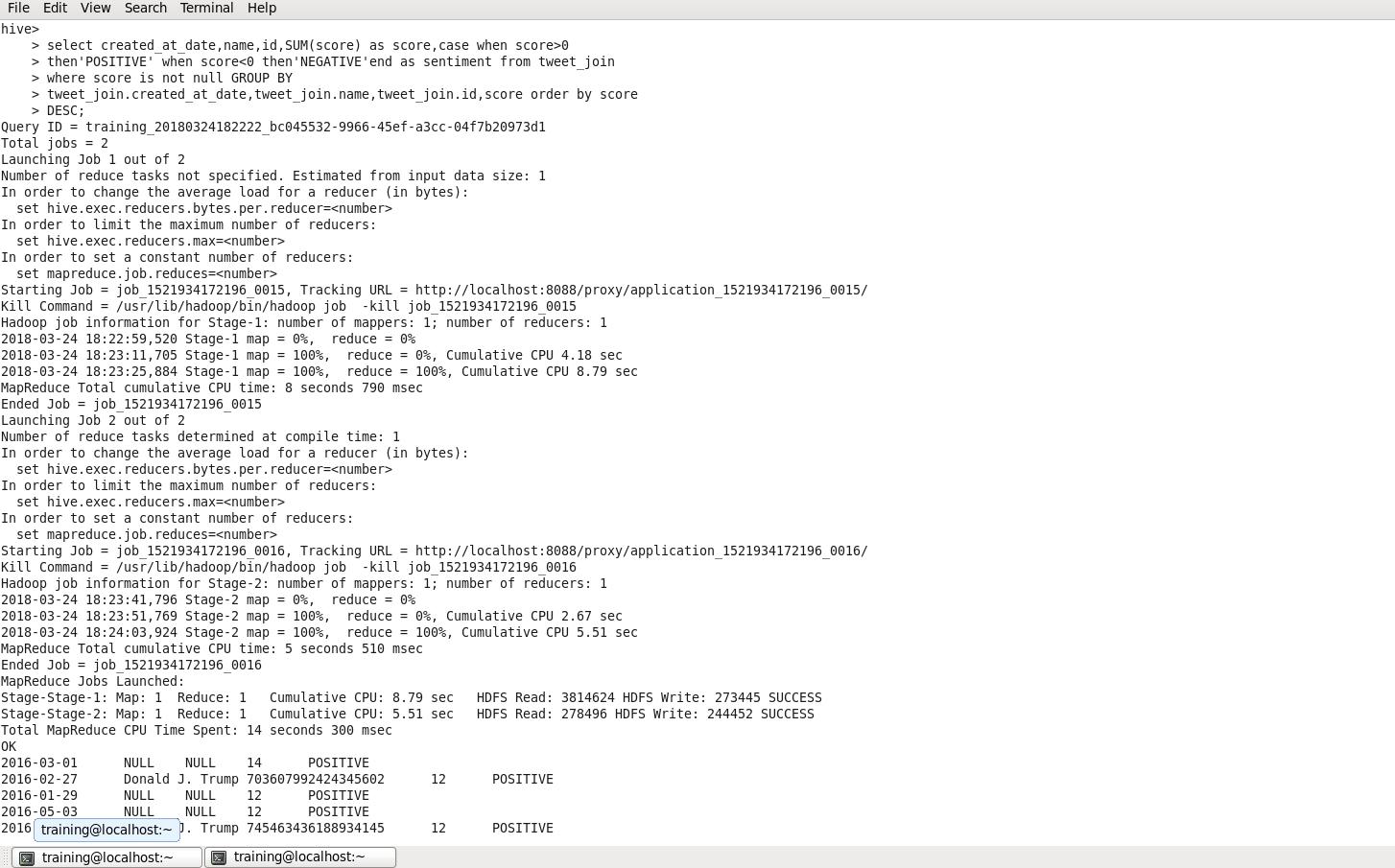
select created\_at\_date,name,id,SUM(score) as score,case when score>0

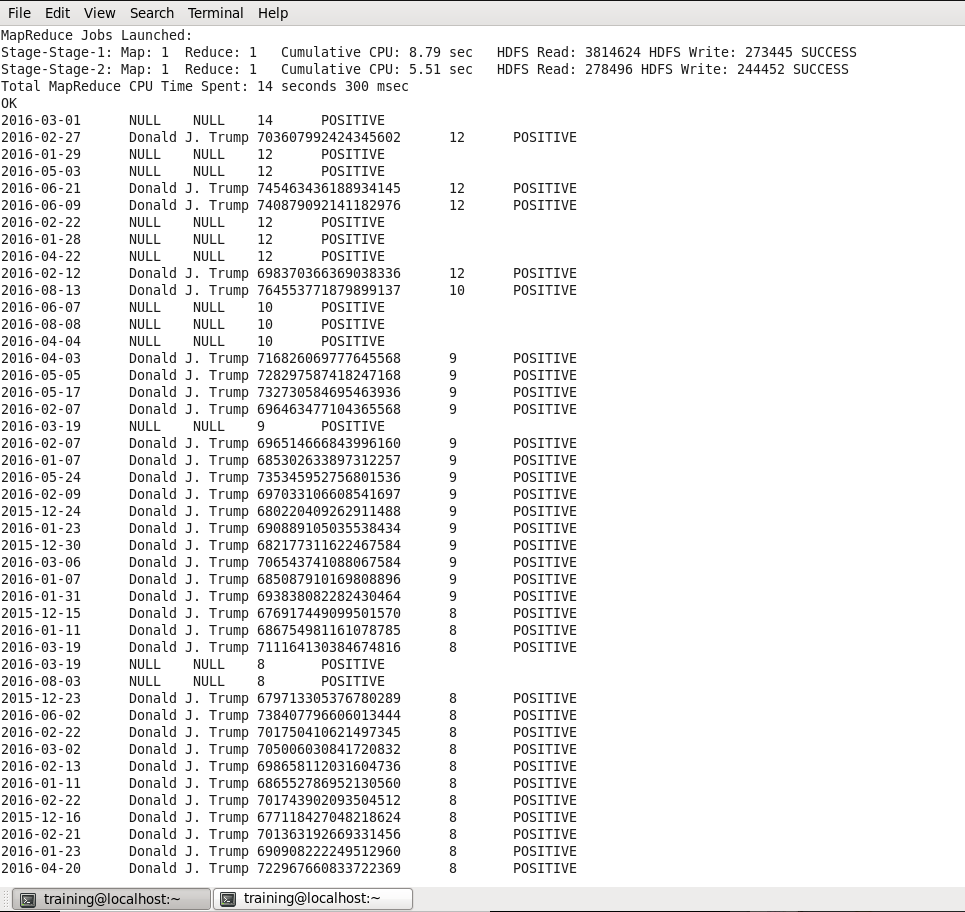
then'POSITIVE' when score<0 then'NEGATIVE'end as sentiment from djoin

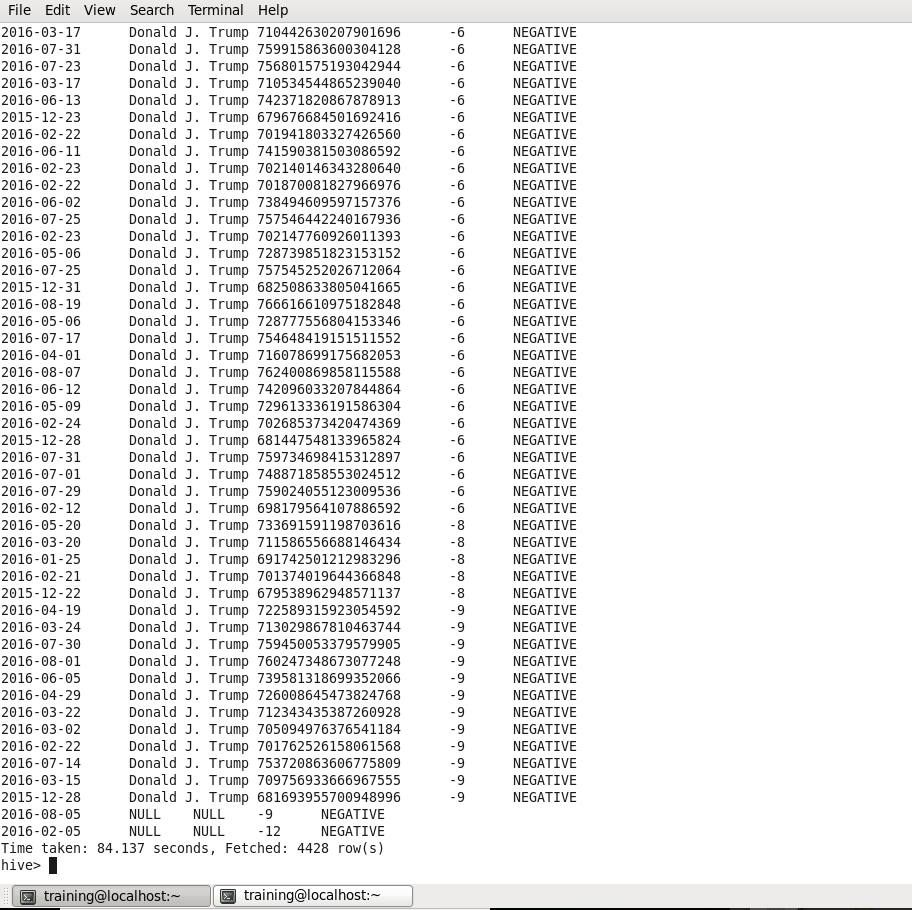
where score is not null GROUP BY

djoin.created\_at\_date,djoin.name,djoin.id,score order by score

DESC;







# Question 2:

Problems Observed:

* We have limited words in the dictionary. As a result, we got a lot of NULL values in the result of the analysis. The dictionary should be able to accommodate all the words (ideally) to be able to correctly score each tweet accurately.
* Ironic sentences will be misinterpreted because we are only comparing the words.
* There is a lack of emotional and sarcastic understanding of the sentences. Sarcastic sentences express negative opinion about a target using positive words in unique way  
  e.g. “Nice perfume. You must shower in it.”

The sentence contains only positive words but actually it expresses a negative sentiment.

* Difficulty in recognizing an entity  
  There is a need to separate out the text about a specific entity and then analyze sentiment towards it.  
  e.g. “I hate Microsoft, but I like Linux”.  
  A simple approach will label it as neutral, however, it carries a specific sentiment for both the entities present in the statement.
* Difficultly in handling Comparison  
  e.g. “My shoes are better than yours”  
  This will be classified as positive for both you and me  
  However, it will fail to understand the comparison.

Possible Solutions:

* Use an algorithm to build the dictionary. Initially use seed words with common synonyms and antonyms and then with every iteration, new words (previously absent in the dictionary) keep getting added to the dictionary. Using this approach we can have a near ideal dictionary with all the words.
* We can use Machine learning for sentiment analysis. Naive Bayes and Decision trees machine learning algorithms can be used for sentiment analysis. Machine learning algorithms learn over time on their own and provide accurate results. Machine learning algorithms will be able to learn the emotional and sarcastic nature of the sentences and also the subtle differences in the normal and sarcastic tone.