**Twitter Data Analysis**

**Abhidha Arya (axa162031)**

**Finding out the hashtags used and how many times each hashtag is used.**

Adding the JAR file and creating table for Twitter feed data:

**Query:**

ADD JAR /home/training/Desktop/json-serde-1.3.8-jar-with-dependencies.jar;

CREATE TABLE Tweet\_data(

retweet\_count INT,

created\_at STRING,

text STRING,

ID BIGINT,

source STRING,

in\_reply\_to\_screen\_name STRING,

user STRUCT< location:STRING, id:BIGINT, id\_str:STRING, name:STRING, screen\_name:STRING, geo\_enabled:BOOLEAN, lang:STRING, protected:BOOLEAN, followers\_count:INT, friends\_count:INT, listed\_count:INT,

favourites\_count:INT, profile\_background\_color:INT>,

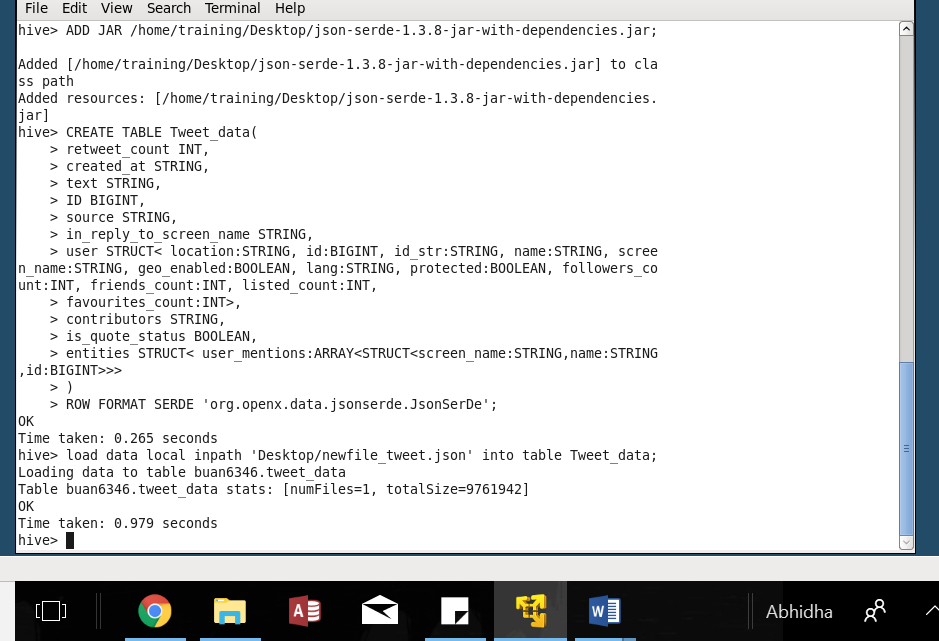
contributors STRING,

is\_quote\_status BOOLEAN,

entities STRUCT< user\_mentions:ARRAY<STRUCT<screen\_name:STRING,name:STRING,id:BIGINT>>>

)

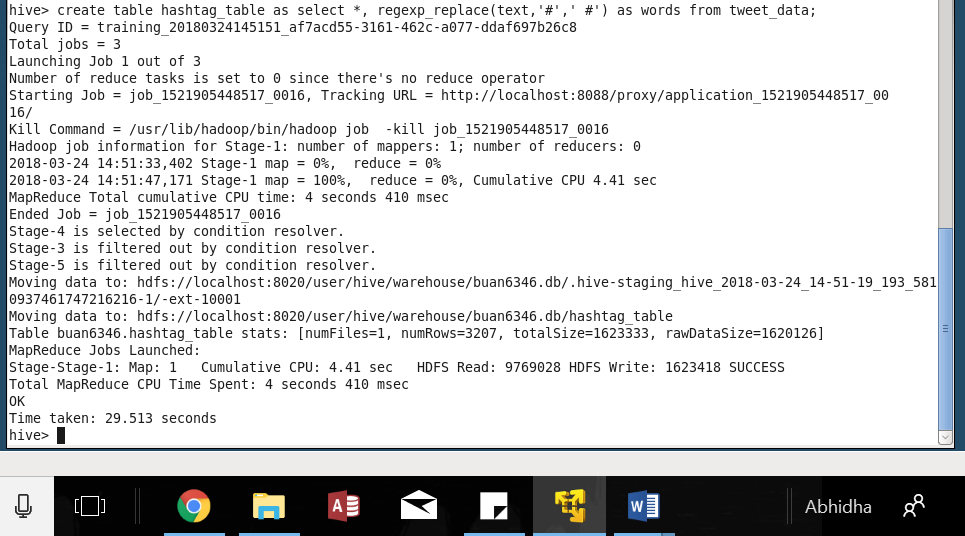
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe';



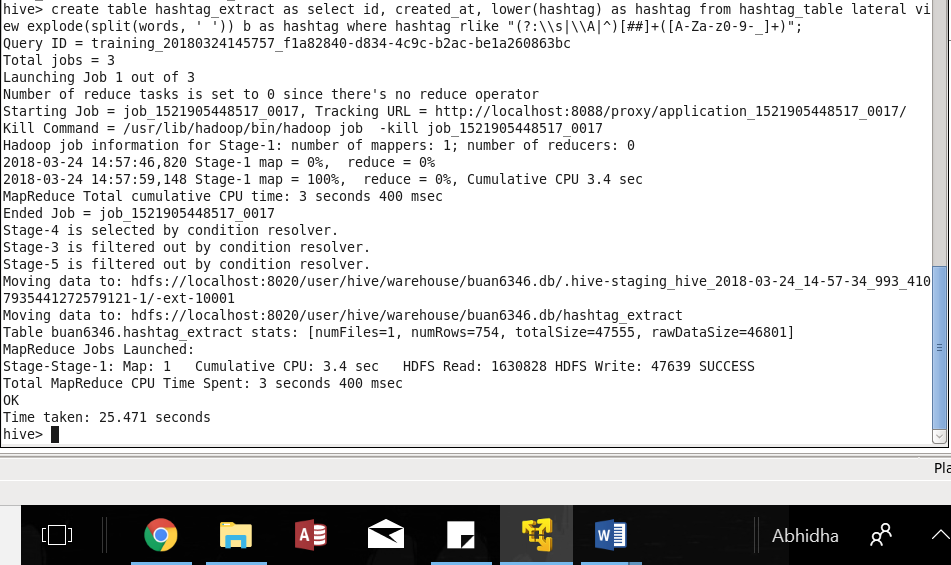
**Query for creating hashtag table:**

CREATE TABLE hashtag\_table AS SELECT \*, regexp\_replace(text,'#',' #') AS words FROM tweet\_data;

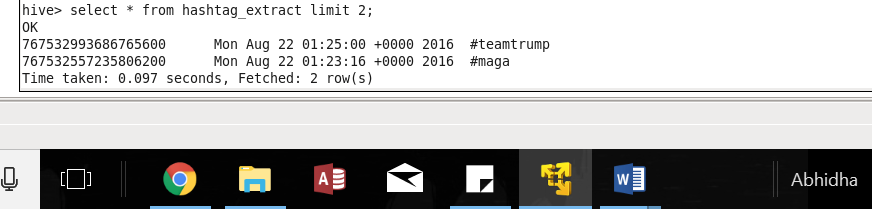
There are many hashtags which are placed next to each other, therefore calculating every hashtag in the text.



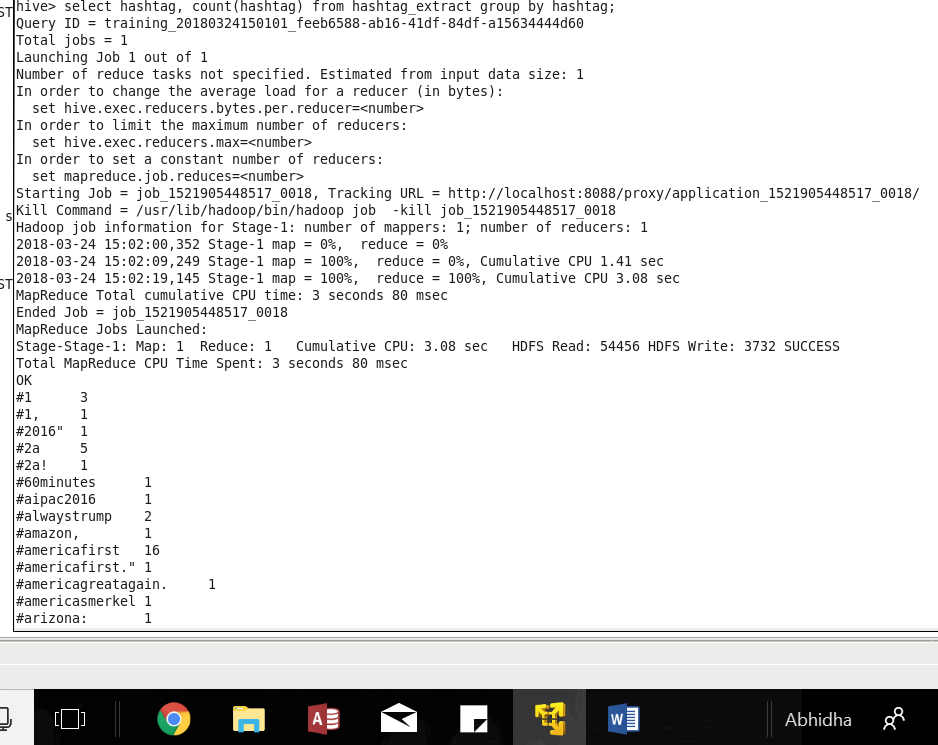
CREATE TABLE hashtag\_extract AS SELECT ID, created\_at, LOWER(hashtag) AS hashtag FROM hashtag\_table LATERAL VIEW explode(split(words, ' ')) b AS hashtag WHERE hashtag RLIKE "(?:\\s|\\A|^)[##]+([A-Za-z0-9-\_]+)";

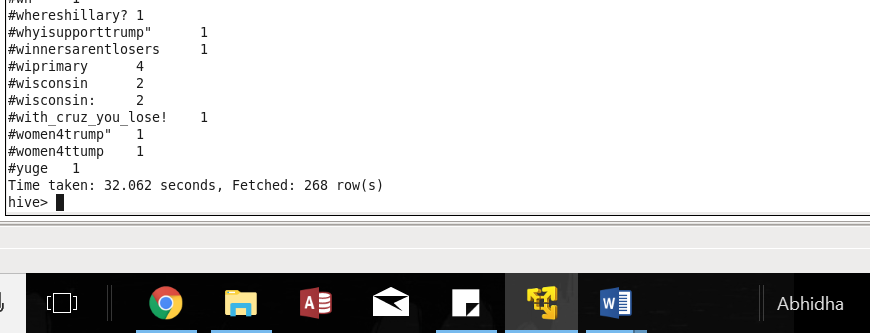


**Sample result:**



SELECT hashtag, count(hashtag) FROM hashtag\_extract GROUP BY hashtag;



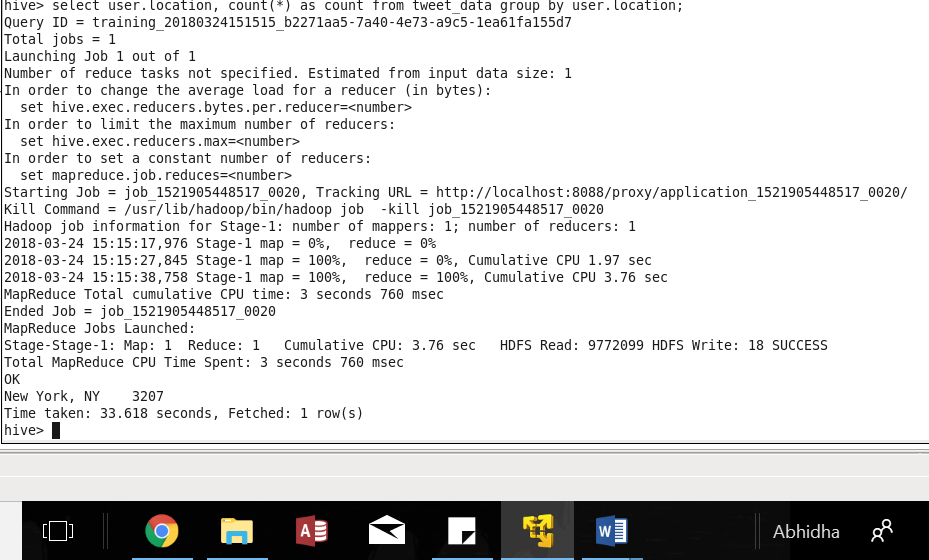


From the results we can see that there is a total of 268 hashtags used with the number of times they have been used.

**Finding out which State have the most active users and how many tweets are posted by State.**

Counting tweets as per the state:

SELECT user.location, COUNT(\*) as COUNT from tweet\_data GROUP BY user.location**;**



According to the results we only have NY state, with 3,207 tweets.

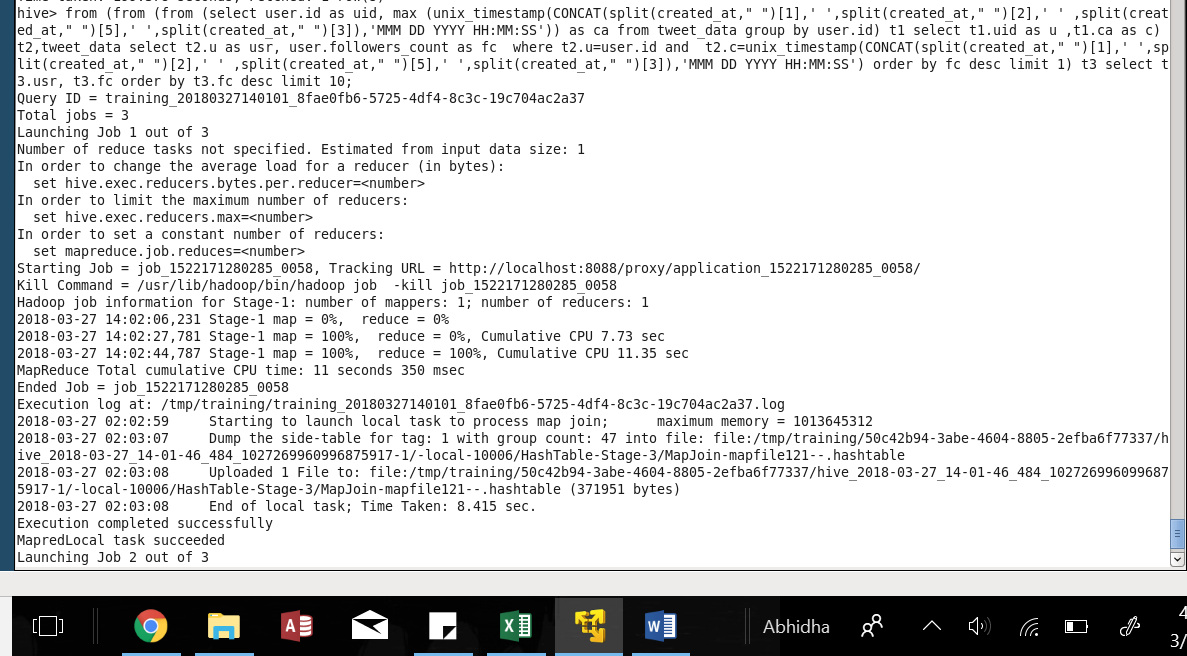
**Based on the user’s followers count, finding out who are the top ten users who have tweeted.**

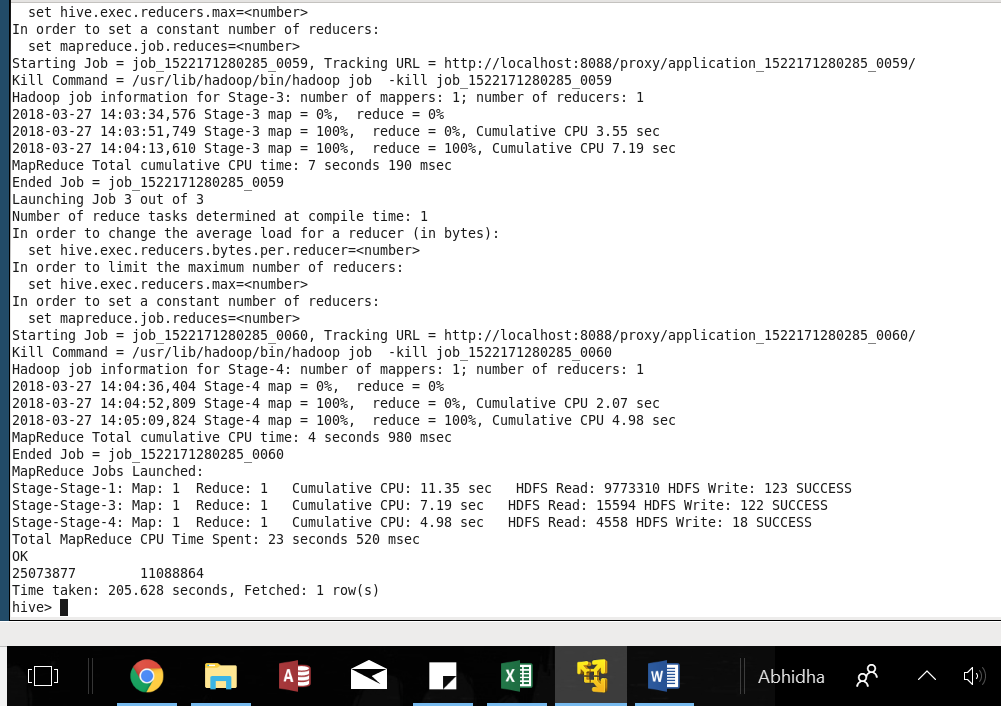
**Query:**

Counting the top 10 users who have tweeted based on the followers count:

First tweet time is converted into unix format and then maximum value of the same is selected, i.e. the latest tweet time. For that time, followers count is selected and since there are different values of the followers count for the same time, max of the followers count is selected. Since, our data only has one user ID only one row is retrieved.

FROM (from (from (select user.id as uid, max (unix\_timestamp(CONCAT(split(created\_at," ")[1],' ',split(created\_at," ")[2],' ' ,split(created\_at," ")[5],' ',split(created\_at," ")[3]),'MMM DD YYYY HH:MM:SS')) as ca from tweet\_data group by user.id) t1 select t1.uid as u ,t1.ca as c) t2,tweet\_data select t2.u as usr, user.followers\_count as fc where t2.u=user.id and t2.c=unix\_timestamp(CONCAT(split(created\_at," ")[1],' ',split(created\_at," ")[2],' ' ,split(created\_at," ")[5],' ',split(created\_at," ")[3]),'MMM DD YYYY HH:MM:SS') order by fc desc limit 1) t3 select t3.usr, t3.fc order by t3.fc desc limit 10;



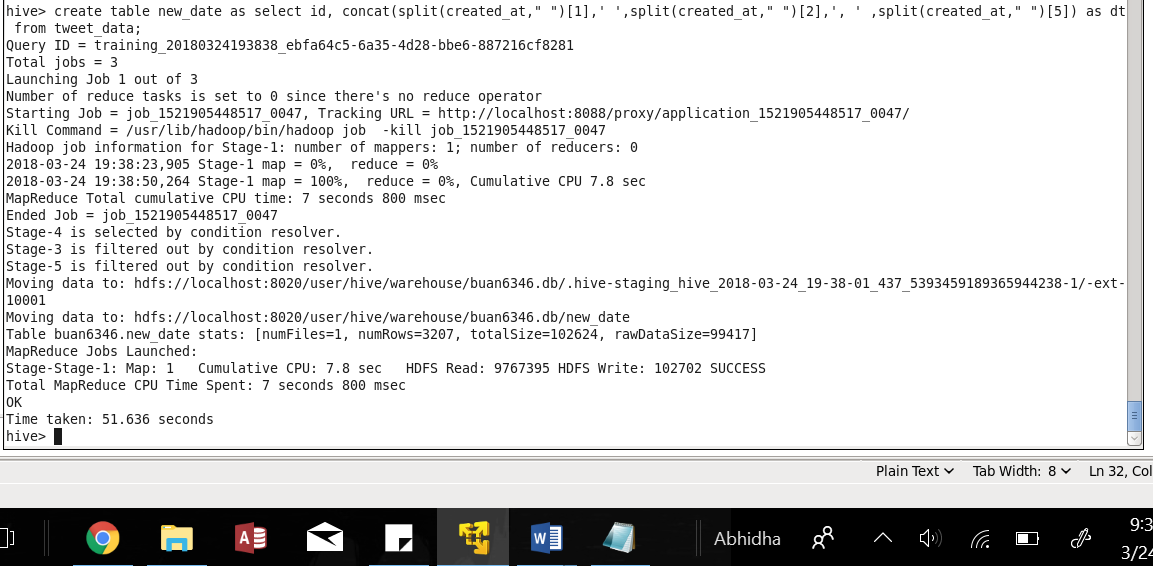


**Sentiment Analysis:**

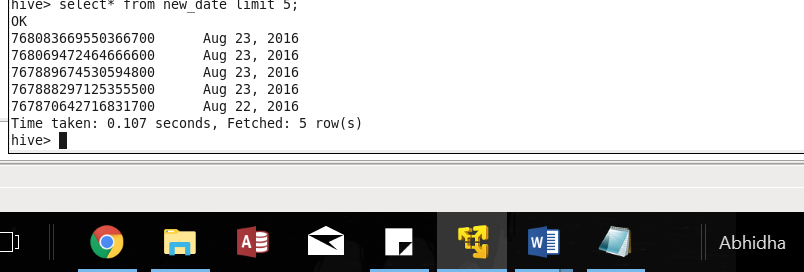
**Finding out the polarity score for each tweet that was posted and whether tweets have a positive or negative sentiment.**

To get the date in the required format:

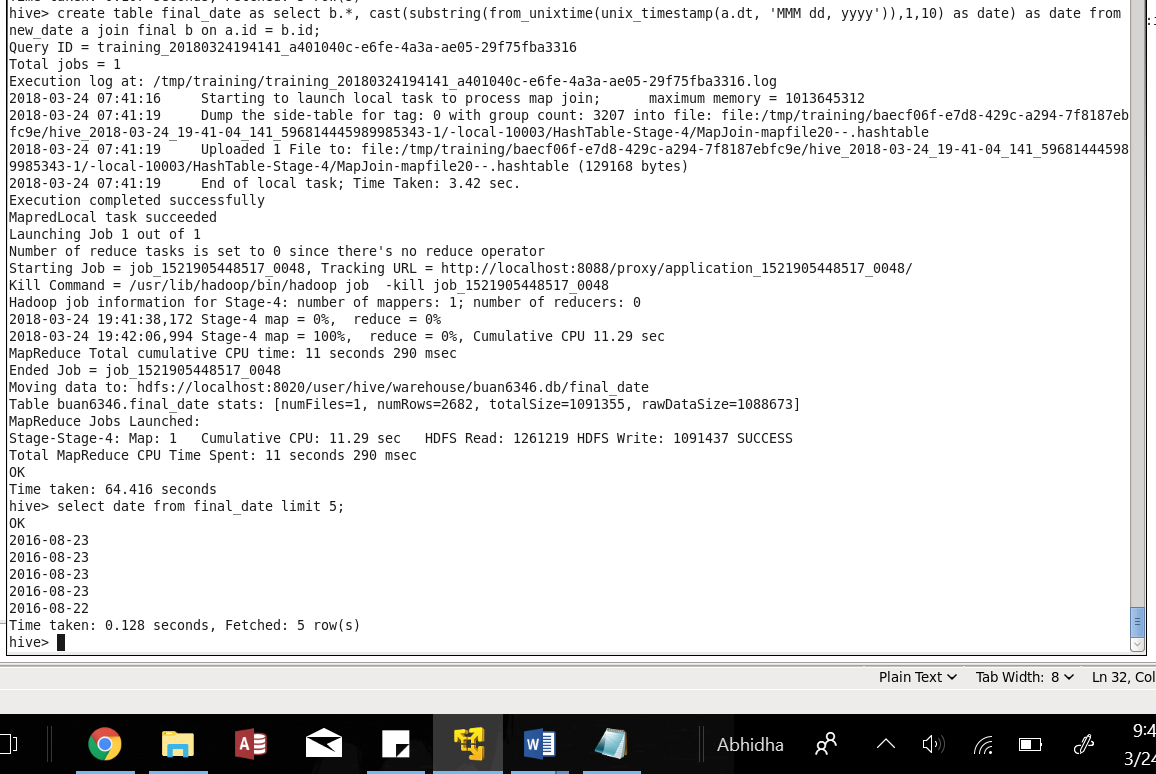
CREATE TABLE new\_date AS SELECT id, CONCAT(split(created\_at," ")[1],' ',split(created\_at," ")[2],', ' ,split(created\_at," ")[5]) AS dt FROM tweet\_data;



Sample result:



CREATE TABLE final\_date AS SELECT b.\*, CAST(SUBSTRING(from\_unixtime(unix\_timestamp(a.dt, 'MMM dd, yyyy')),1,10) AS date) AS date FROM new\_date a JOIN FINAL b on a.id = b.id;



**Sentiment Analysis:**

Query to create and load the table for Dictionary file:

CREATE TABLE dict(

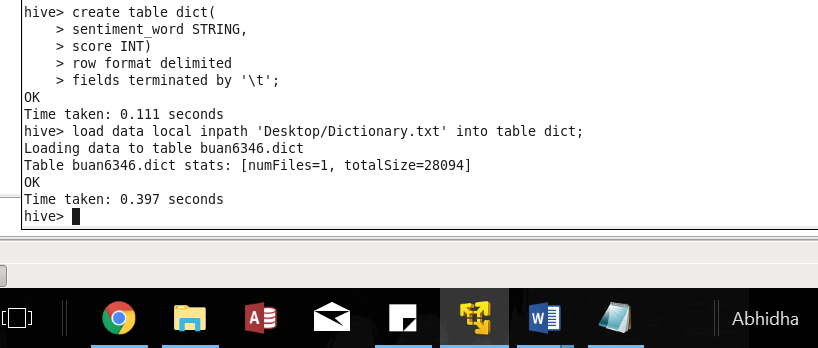
sentiment\_word STRING,

score INT)

row format delimited

fields terminated by '\t';

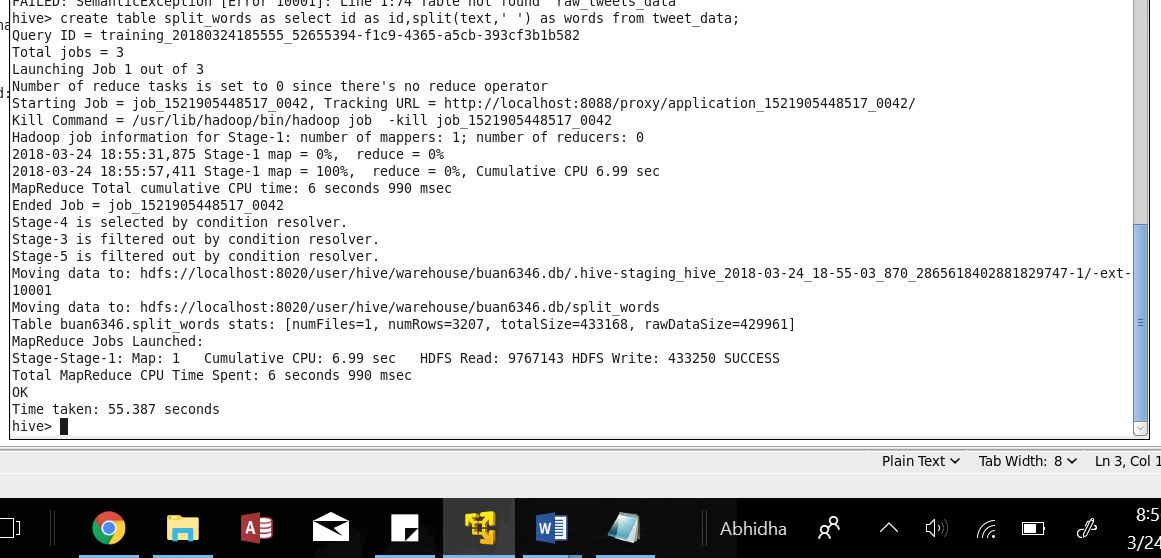
load data local inpath 'Desktop/Dictionary.txt' into table dict;



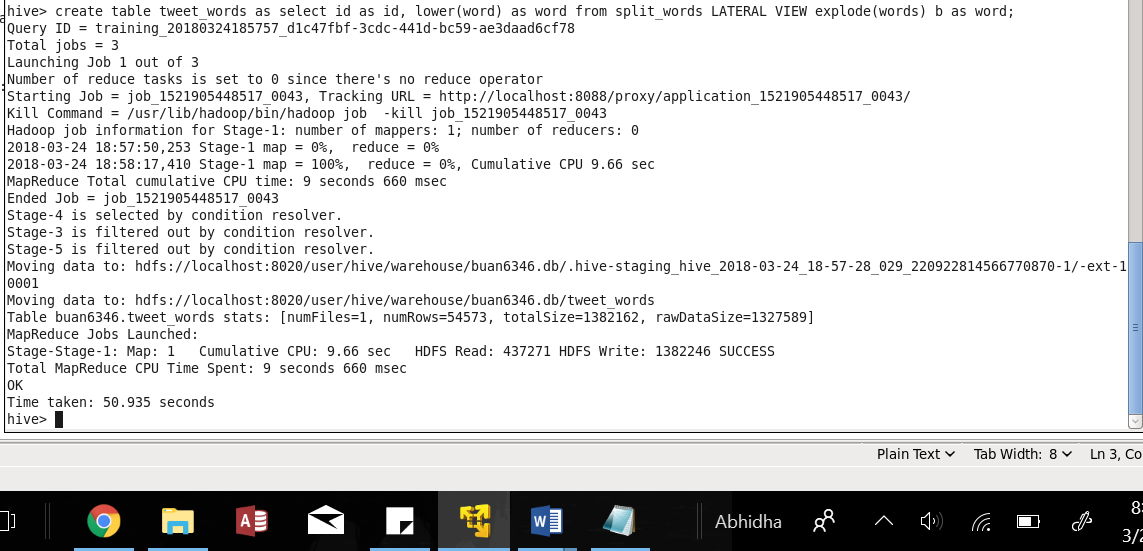
**Preparing data for sentiment Analysis**

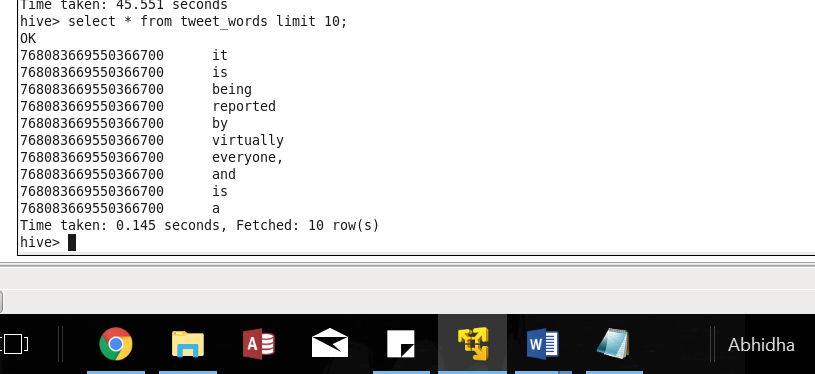
**Query:**

CREATE TABLE split\_words AS SELECT id AS id,split(text,' ') AS words FROM raw\_tweets\_data;

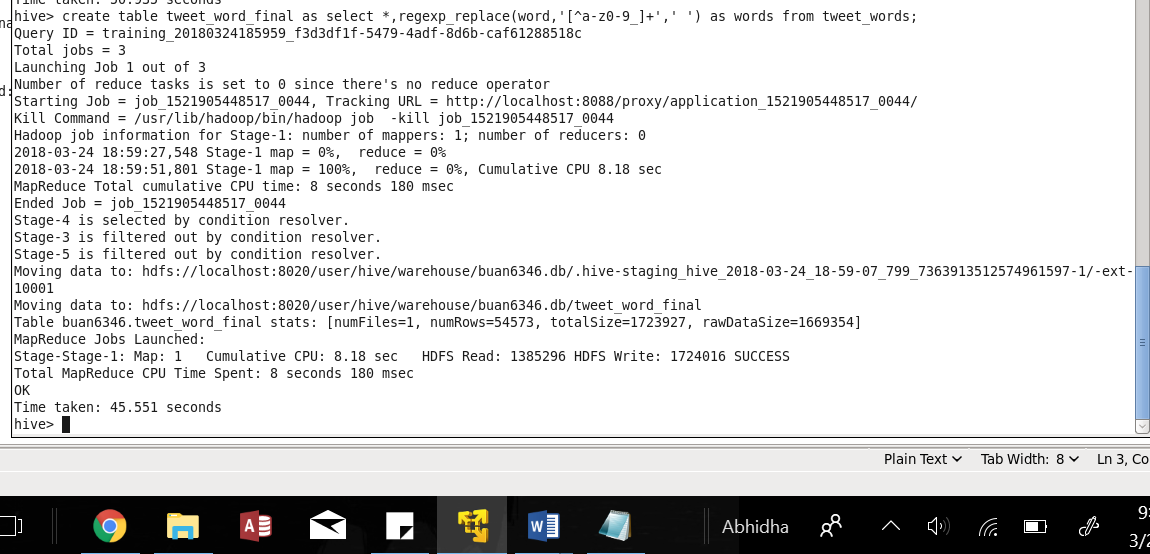


CREATE TABLE tweet\_words AS SELECT id AS id, lower(word) AS word FROM split\_words LATERAL VIEW explode(words) b AS word;

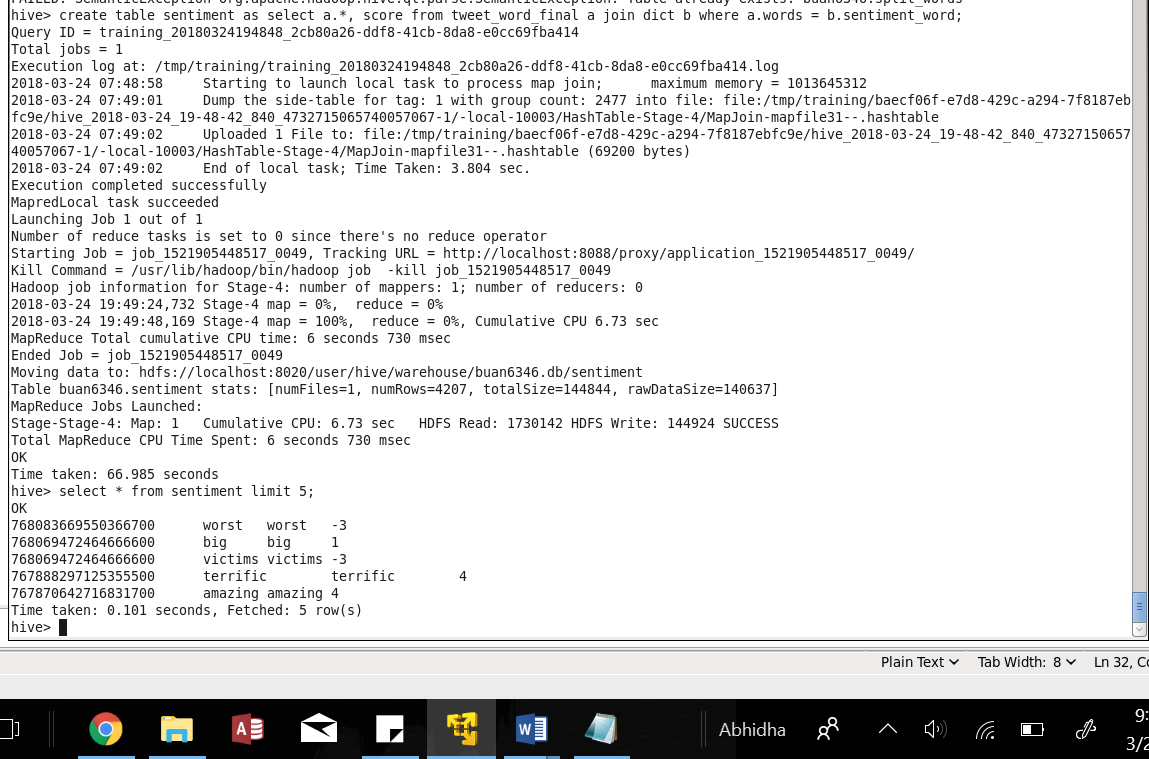




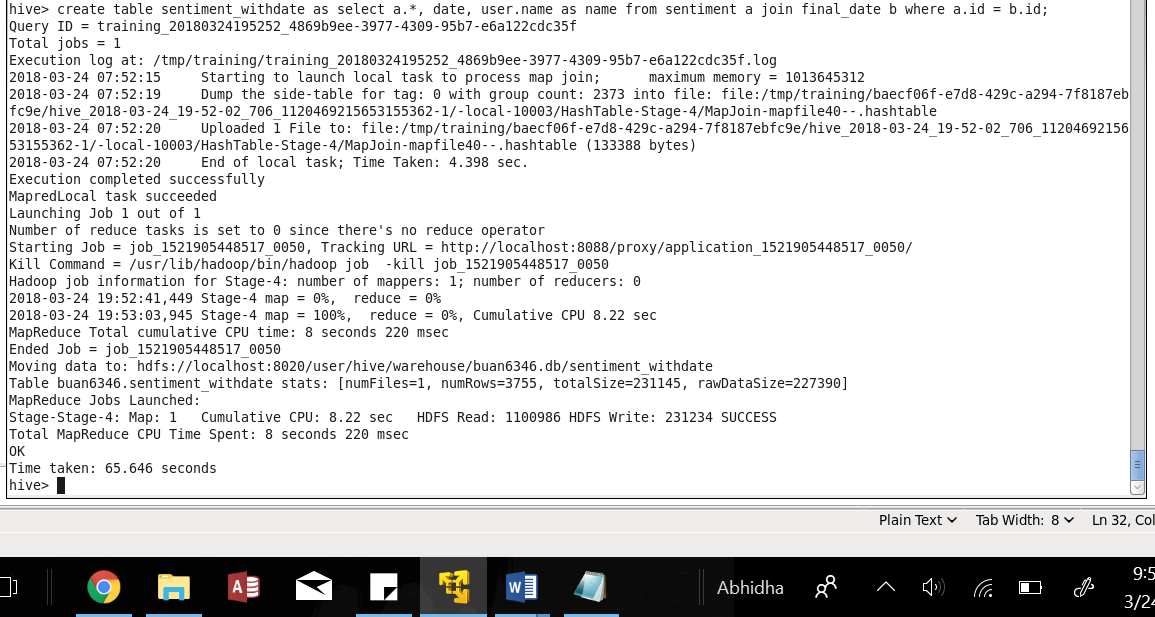
CREATE TABLE tweet\_word\_final AS SELECT \*,regexp\_replace(word,'[^a-z0-9\_]+',' ') AS words FROM tweet\_words;



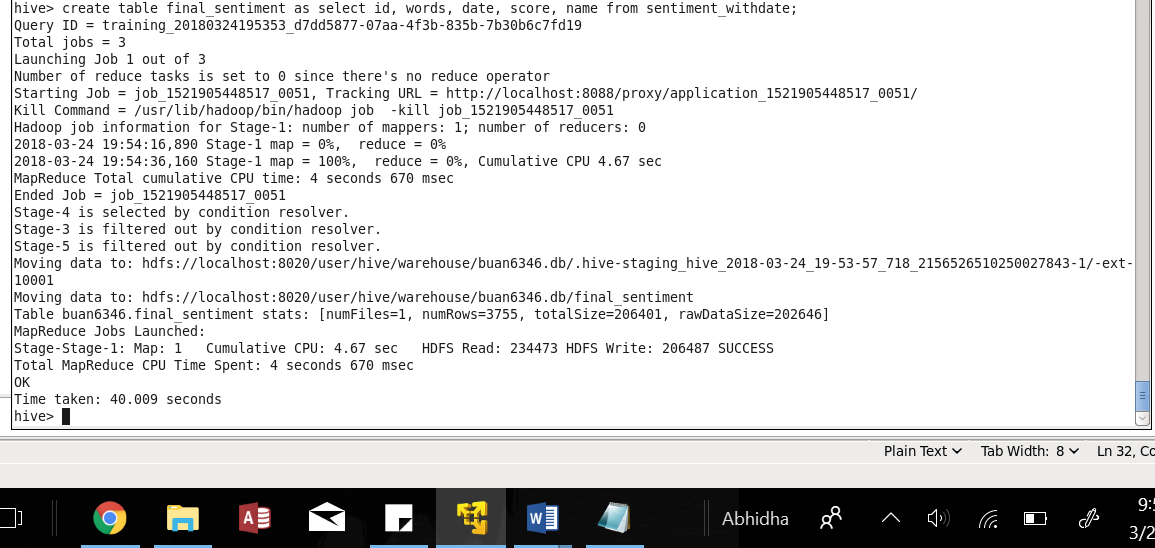
CREATE TABLE sentiment AS SELECT a.\*, score FROM tweet\_word\_final a JOIN dict b WHERE a.words = b.sentiment\_word;



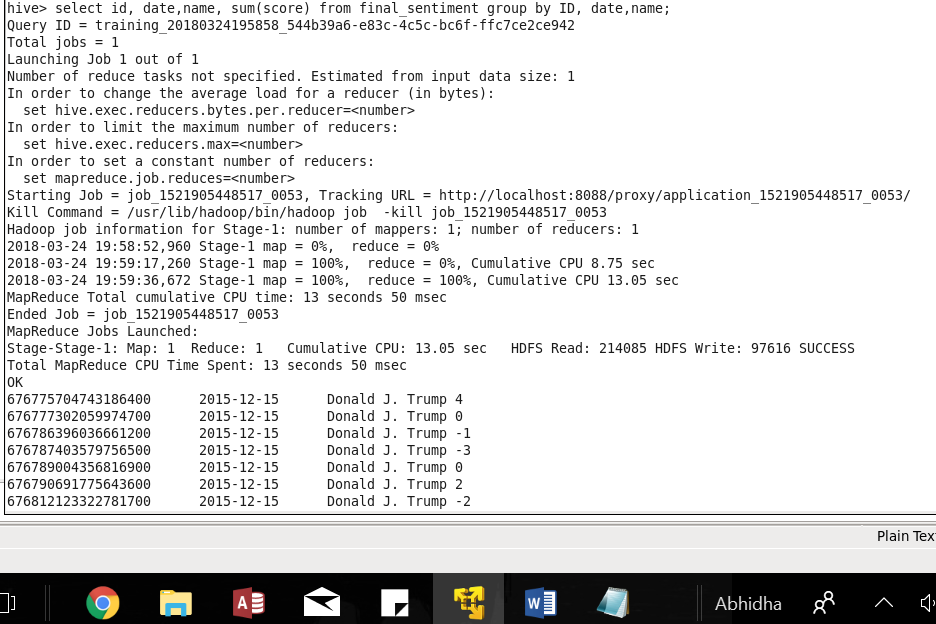
CREATE TABLE sentiment\_withdate AS SELECT a.\*, date, user.name AS name FROM sentiment a join final\_date b WHERE a.id = b.id;

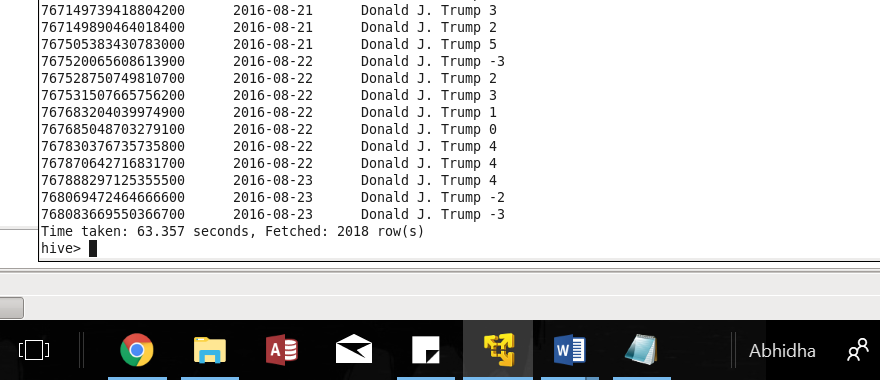


CREATE TABLE final\_sentiment AS SELECT ID, words, date, score, name FROM sentiment\_withdate;



SELECT ID, date,name, sum(score) FROM final\_sentiment GROUP BY ID, date, name;





Above is the polarity score for tweet ID, corresponding user name and date.

**Suggestions to improve the Sentiment Analysis results:**

The dictionary file doesn’t list the adjectives and adverbs which can be added before a word. For instance, if the word is beautiful and someone tweets very beautiful or too beautiful, the sentiment analysis will not show the true score. Also, there is a word “not good” included in the dictionary however, “not bad” or other words which can have adverbs such as not, are not there in the dictionary hence, those words will not get categorized correctly.

To improve we could go beyond words and short phrases and include the role of a word in a sentence and also include more examples of part of speech for each word. Also, meanings for emoticons could be added.