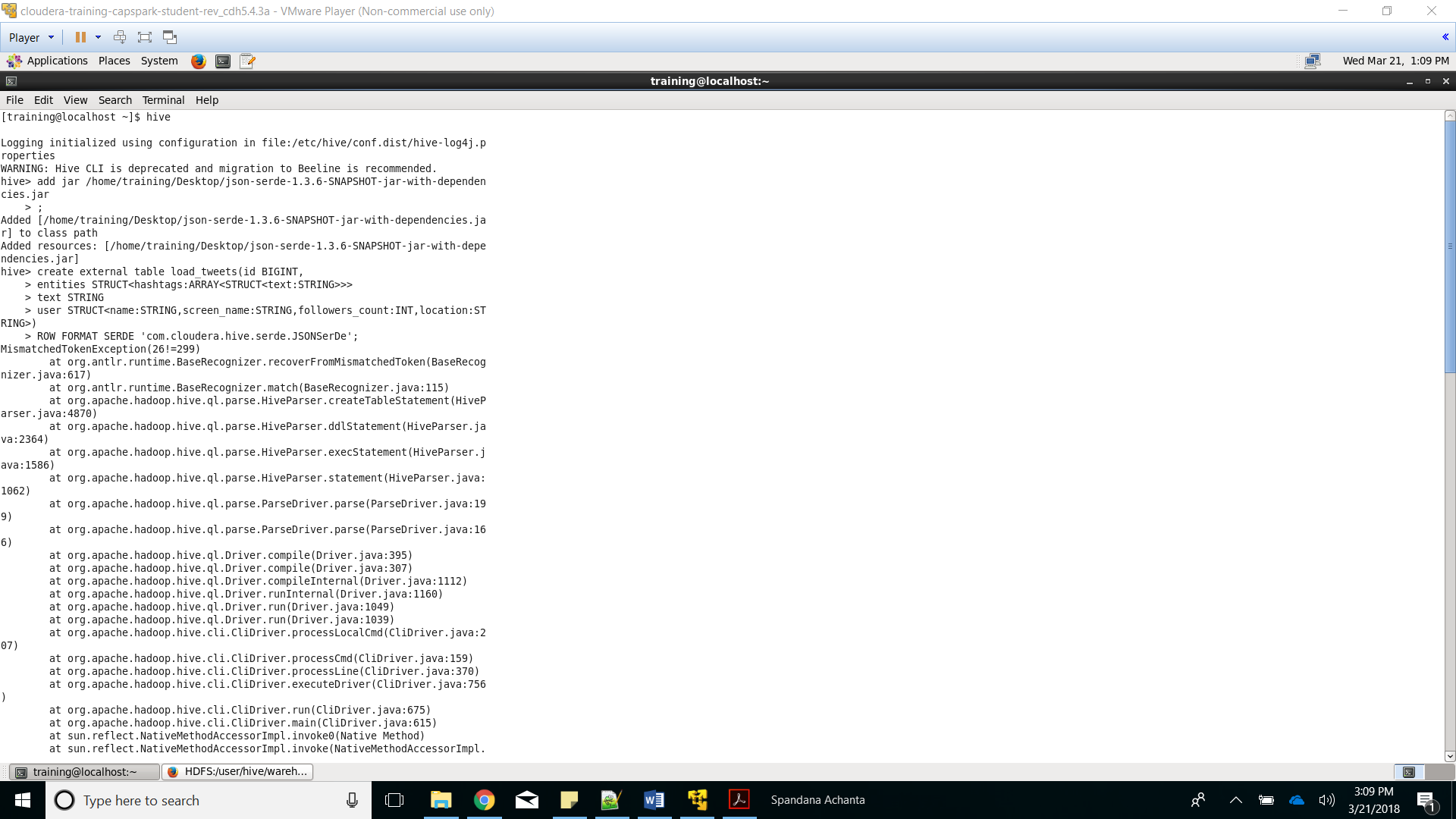
ADD JAR /home/training/Desktop/json-serde-1.3.6-SNAPSHOT-jar-with-dependencies.jar



Added the JSON Serde to hive resources to use for manipulating JSON files

create external table load\_tweets(id BIGINT,

created\_at STRING,

place STRUCT<full\_name STRING>,

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

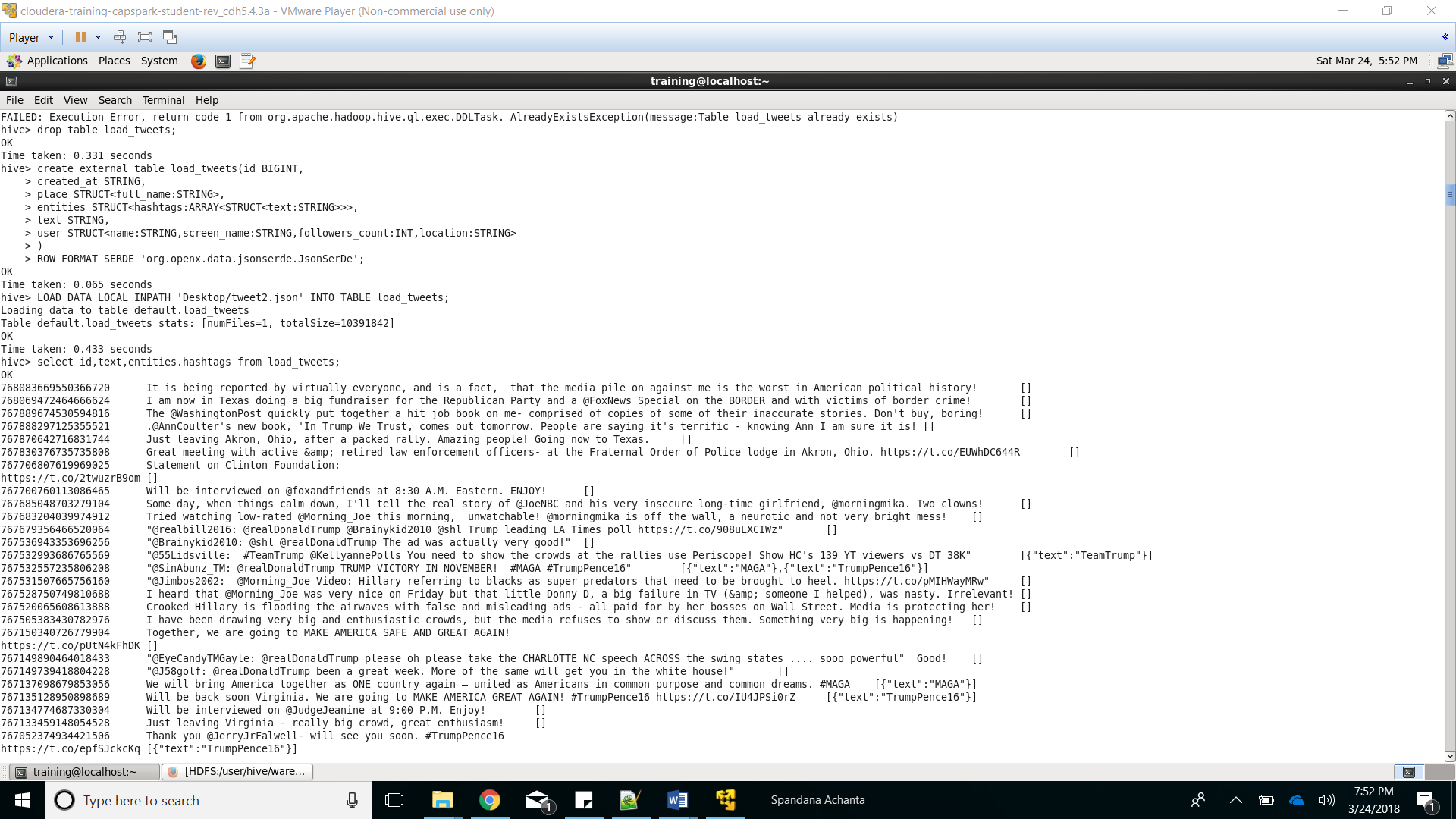
text STRING,

user STRUCT<name:STRING,screen\_name:STRING,followers\_count:INT,location:STRING>

)

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

LOAD DATA LOCAL INPATH 'Desktop/tweet2.json' INTO TABLE load\_tweets;



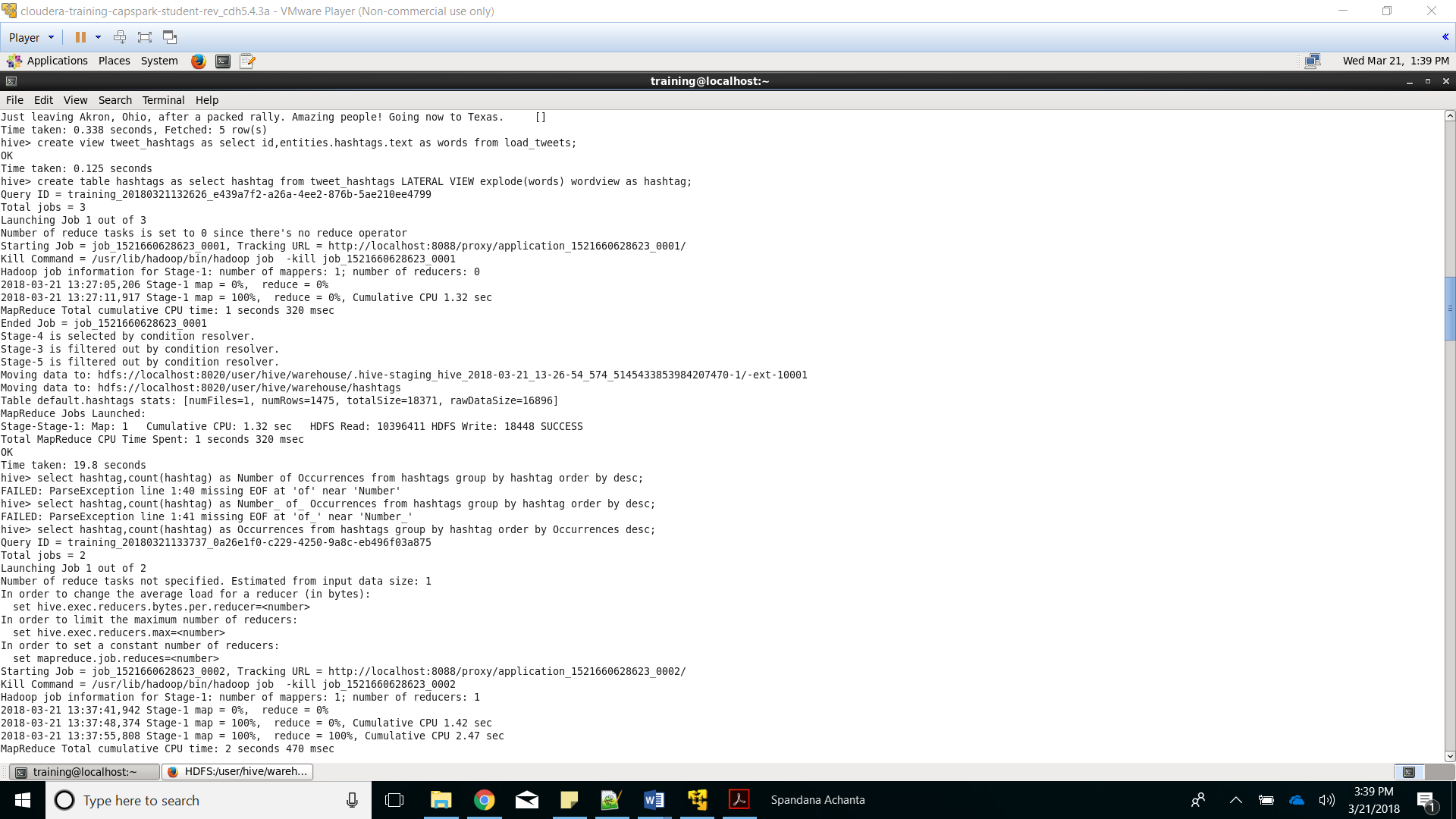
Displaying some of the data to verify

Creating table to store tweet data

create view tweet\_hashtags as select id,entities.hashtags.text as words from load\_tweets;

create table hashtags as select hashtag from tweet\_hashtags LATERAL VIEW explode(words) wordview as hashtag

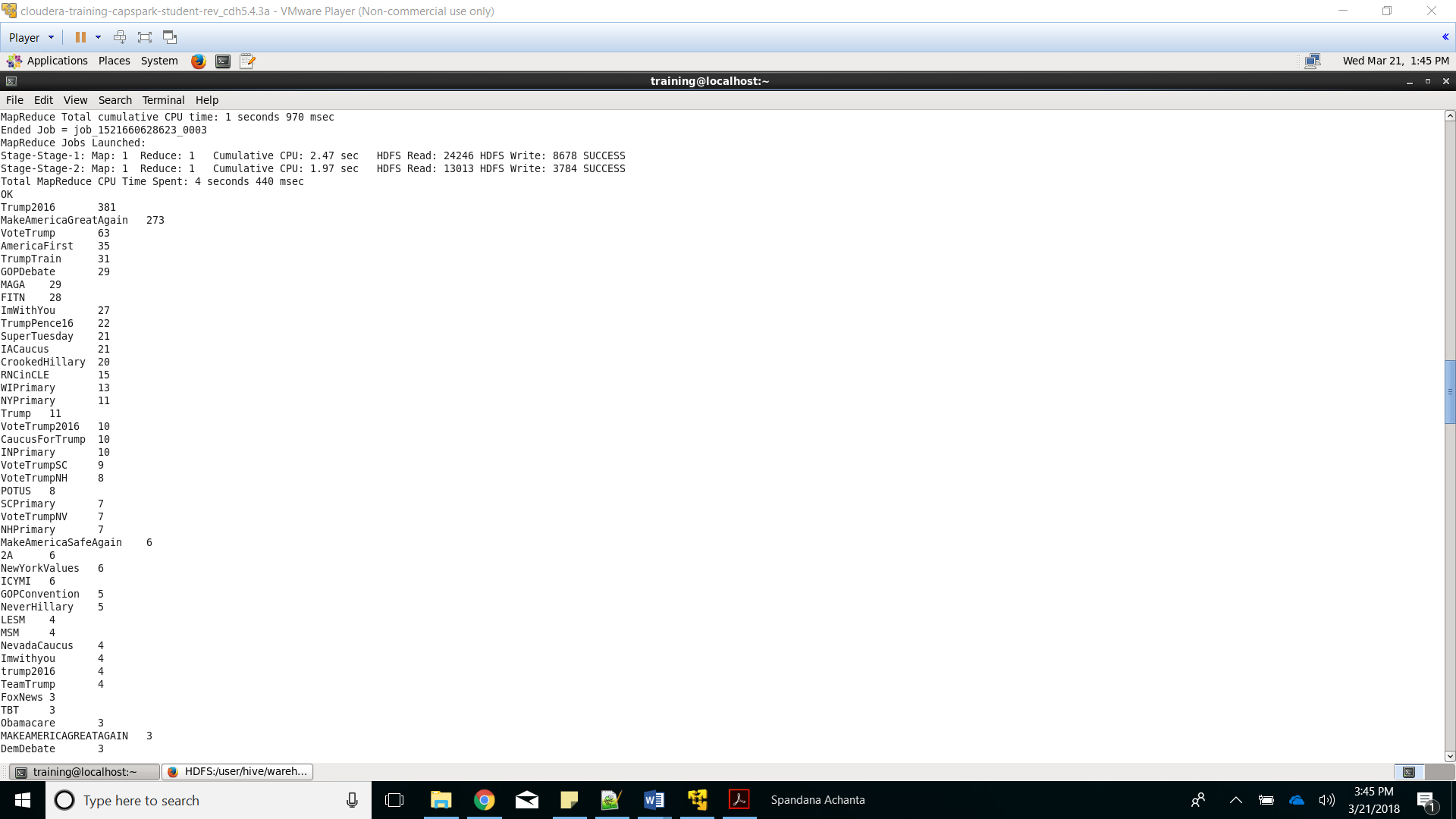
select hashtag,count(hashtag) as Occurrences from hashtags group by hashtag order by Occurrences desc;



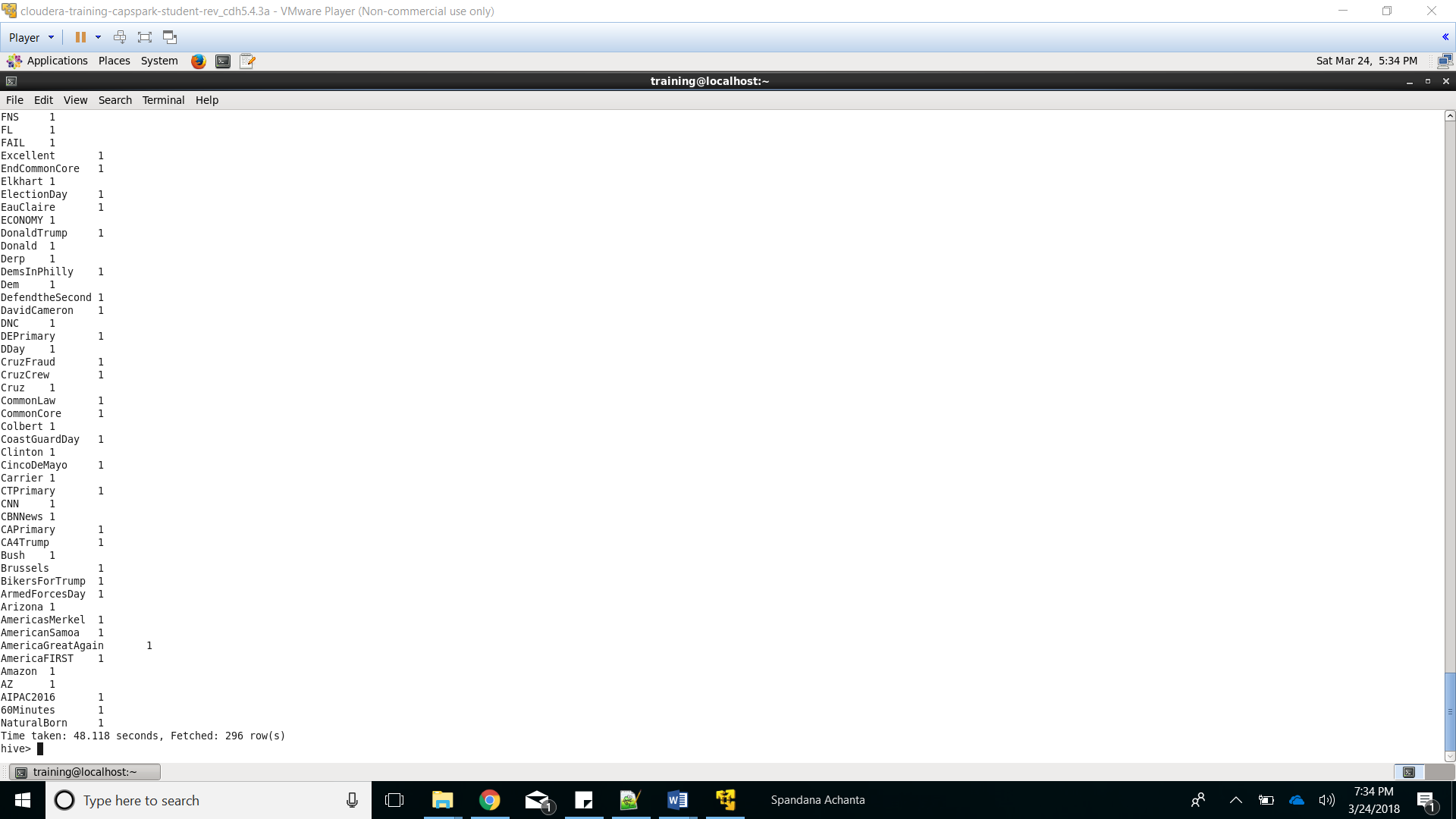
Counting occurrences of each hashtag and ordering from highest to lowest occurence

Created table hashtags splits each word in hashtag text to separate rows and stores in column hashtag

Created view tweet\_hashtags that stores id, and text of hashtags from load\_tweets



Result of previous query displayed



There were 296 distinct hashtags

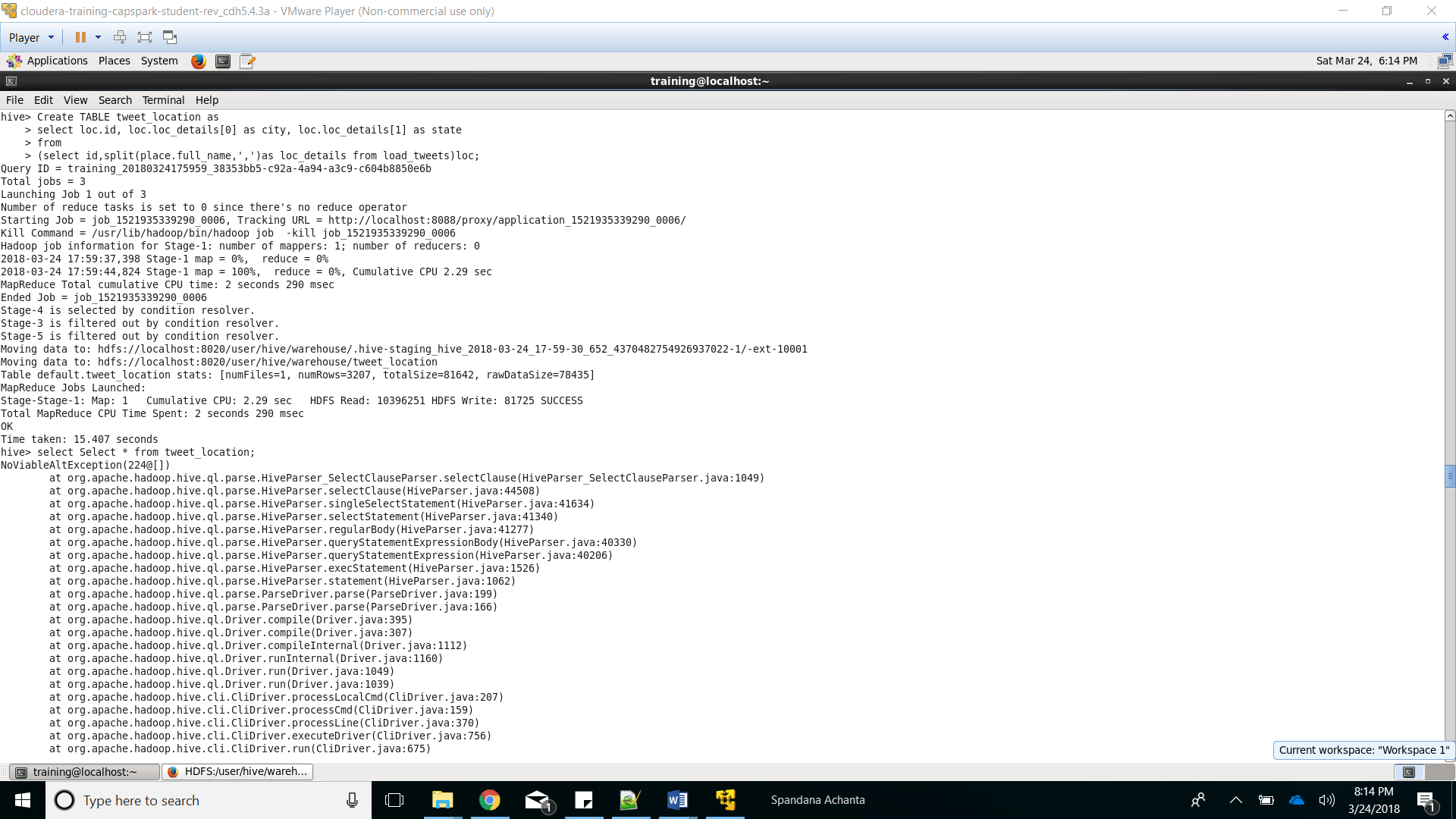
*Here I assumed that we are going to find the states from where Trump has tweets the most*

Create TABLE tweet\_location as

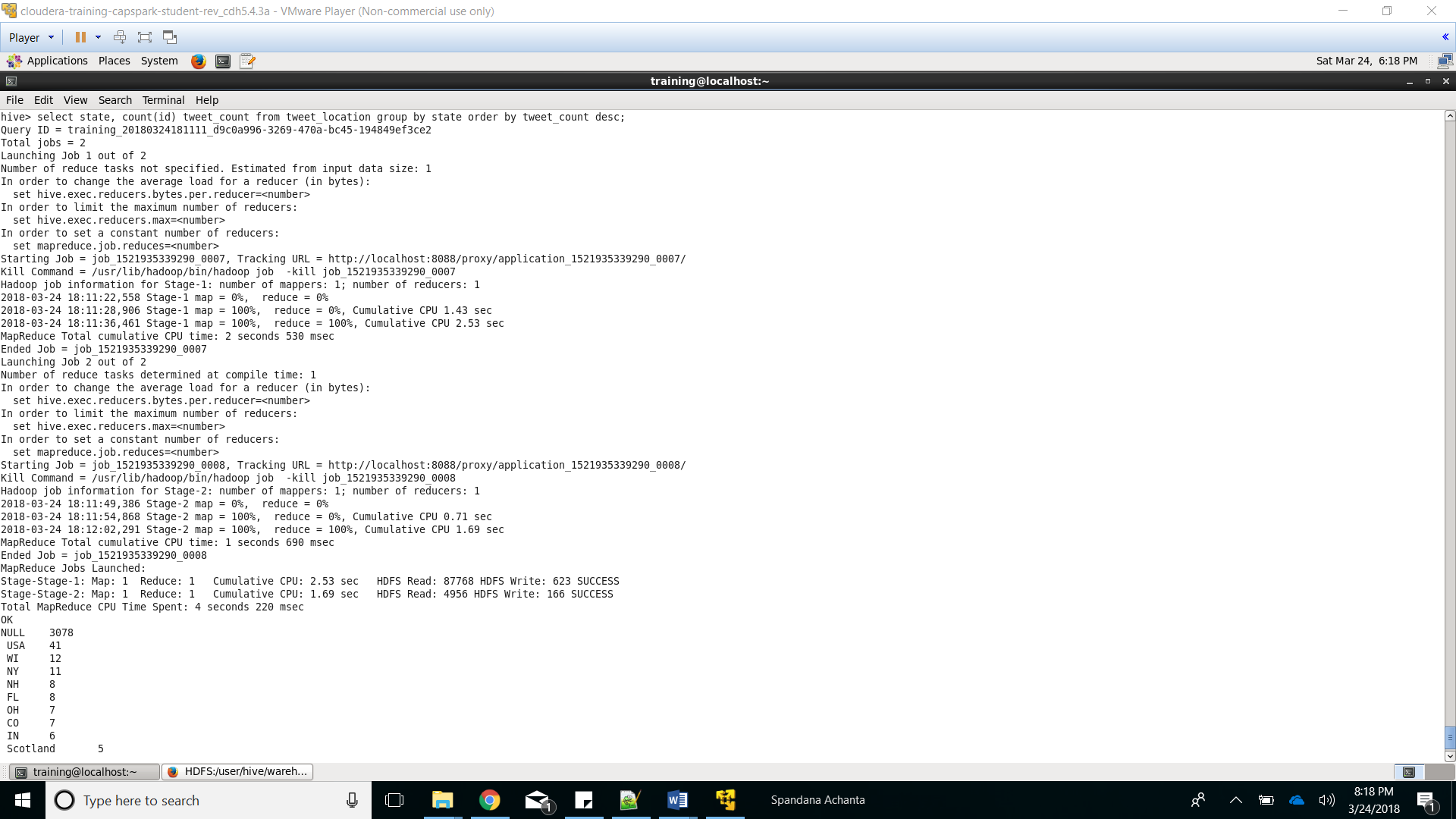
select loc.id, loc.loc\_details[0] as city, loc.loc\_details[1] as state from

(select id,split(place.full\_name,',')as loc\_details from load\_tweets)loc;

select state, count(id) tweet\_count from tweet\_location group by state order by tweet\_count desc;



Created table named tweet\_location to get separate lines for location level details of the tweet

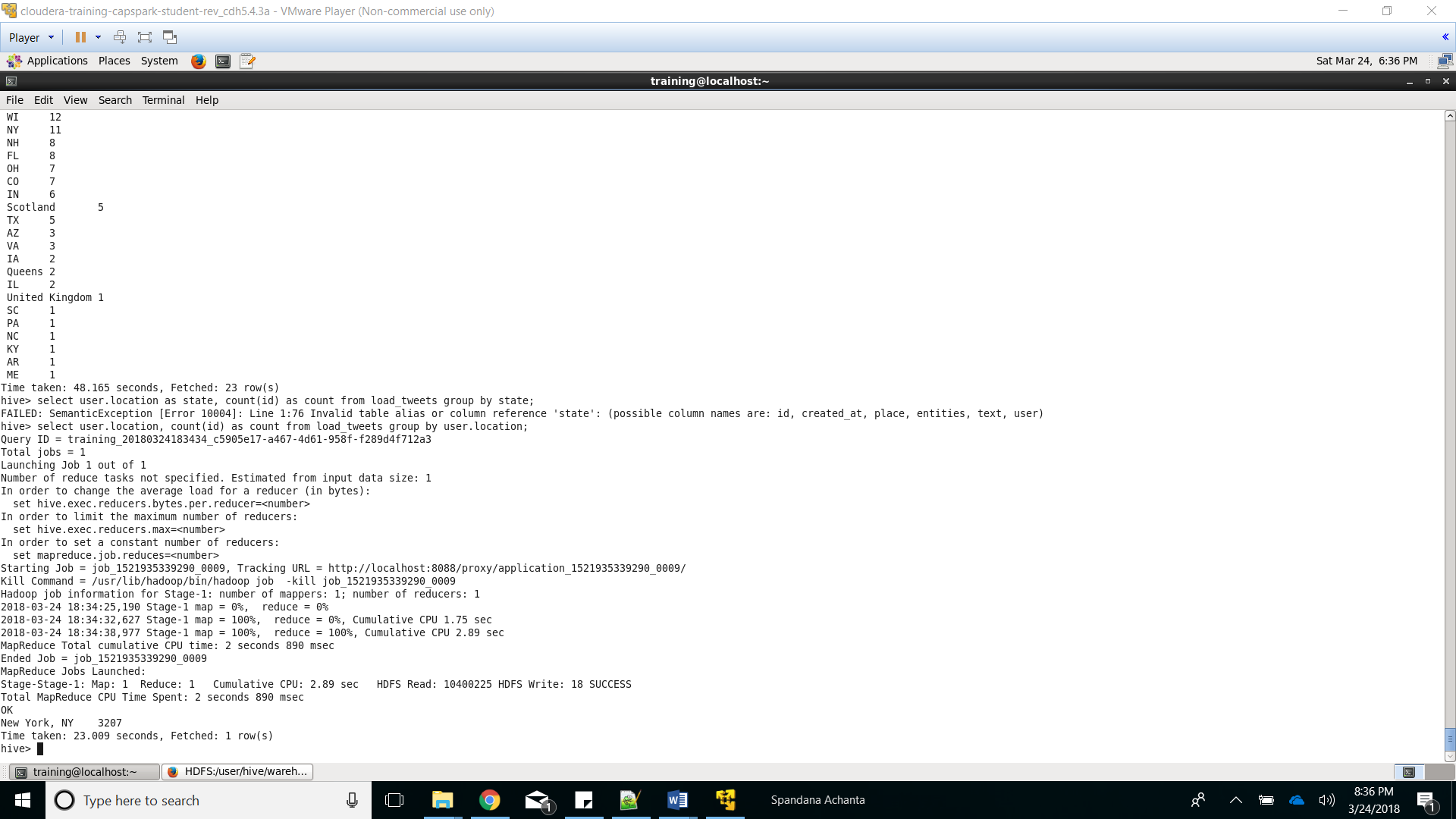


Number of tweets made grouped by different states is displayed.

Neglecting ‘Null’ and ‘USA’ the most active tweets were made from Wichita

*Here I assumed that we are going to find the states by the location in the user section of the tweets*

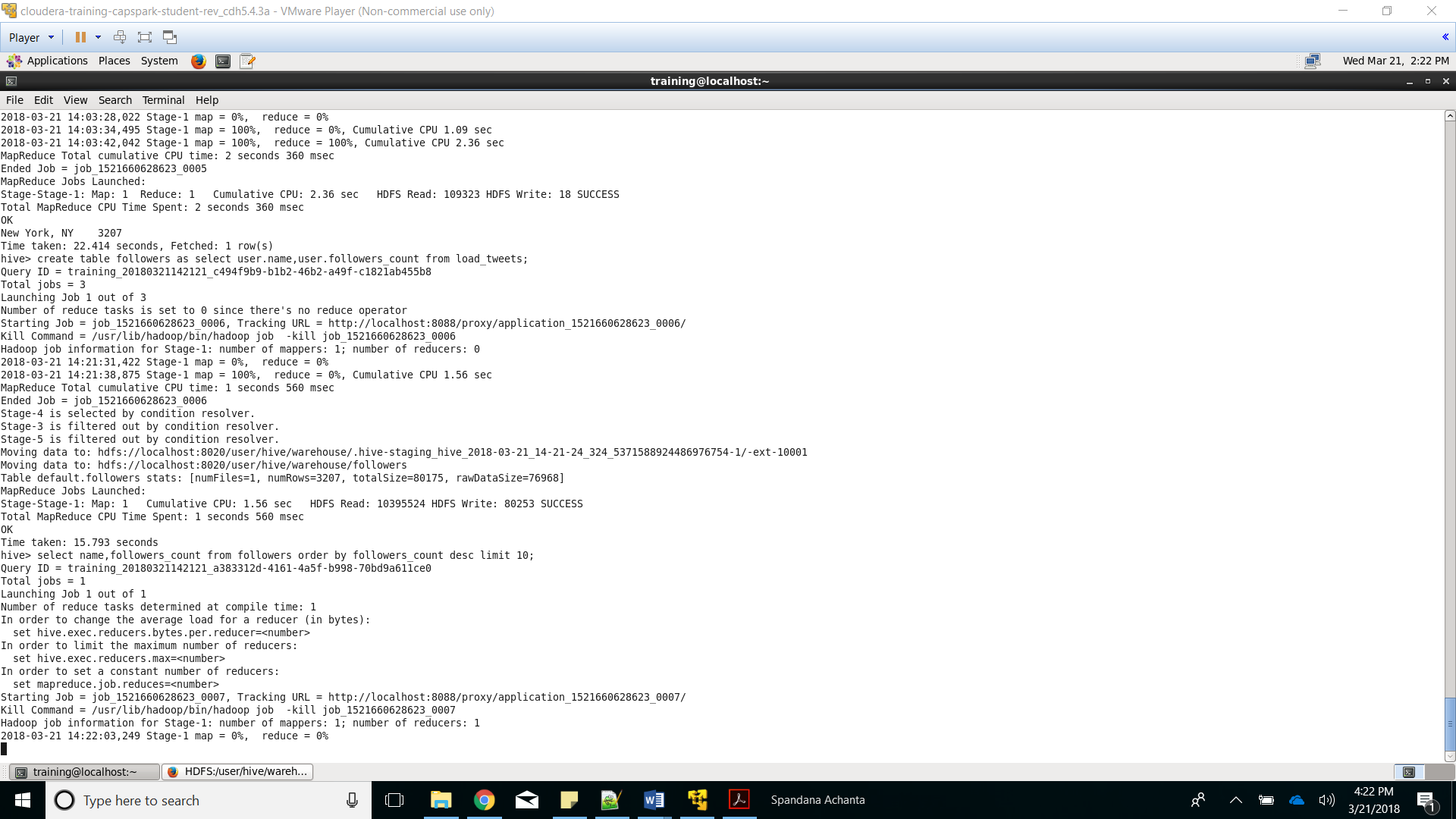
select user.location, count(id) as count from load\_tweets group by user.location;



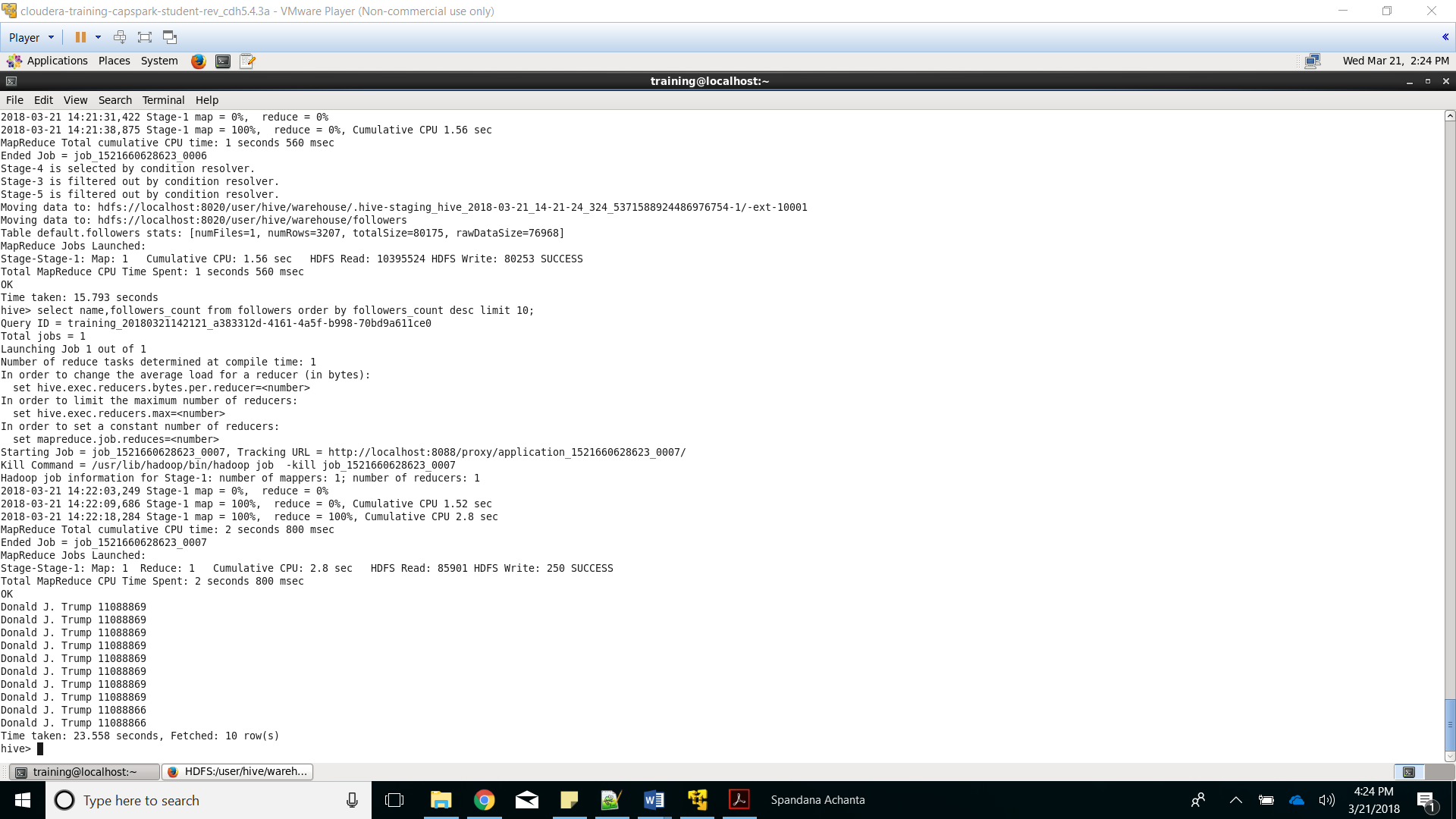
This query gives the location of the user profile that has highest tweets which is New York

create table followers as select user.name,user.followers\_count from load\_tweets;

select name,followers\_count from followers order by followers\_count desc limit 10;



Creating table called ‘followers’ to store data about user name corresponding followers\_count



Top 10 users with highest followers\_count

create table split\_words as select id,split(text,' ') as words,FROM\_UNIXTIME(UNIX\_TIMESTAMP(created\_at,"EEE MMM d HH:mm:ss Z yyyy"),'yyyy-MM-dd') as date, user.name as name from load\_tweets;

select \* from split\_words limit 5;

create table tweet\_words as select id,date,name,word from split\_words LATERAL VIEW explode(words) w as word;

select \* from tweet\_words limit 5;

create table dictionary(word string,score int)

ROW FORMAT delimited

FIELDS TERMINATED BY '\t';

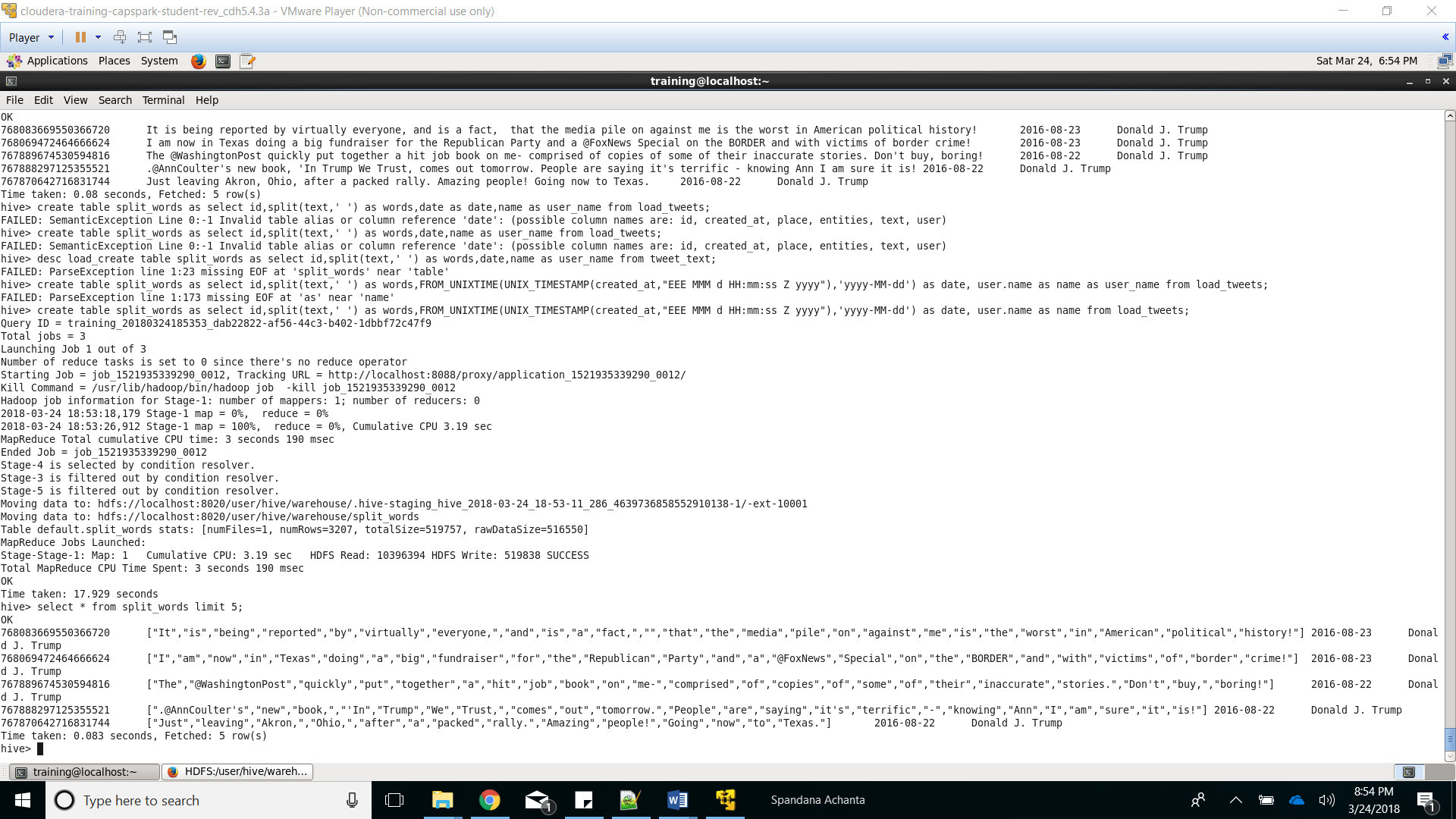
Load data local inpath 'Desktop/dictionary.txt into table dictionary;

select \* from dictionary limit 5;

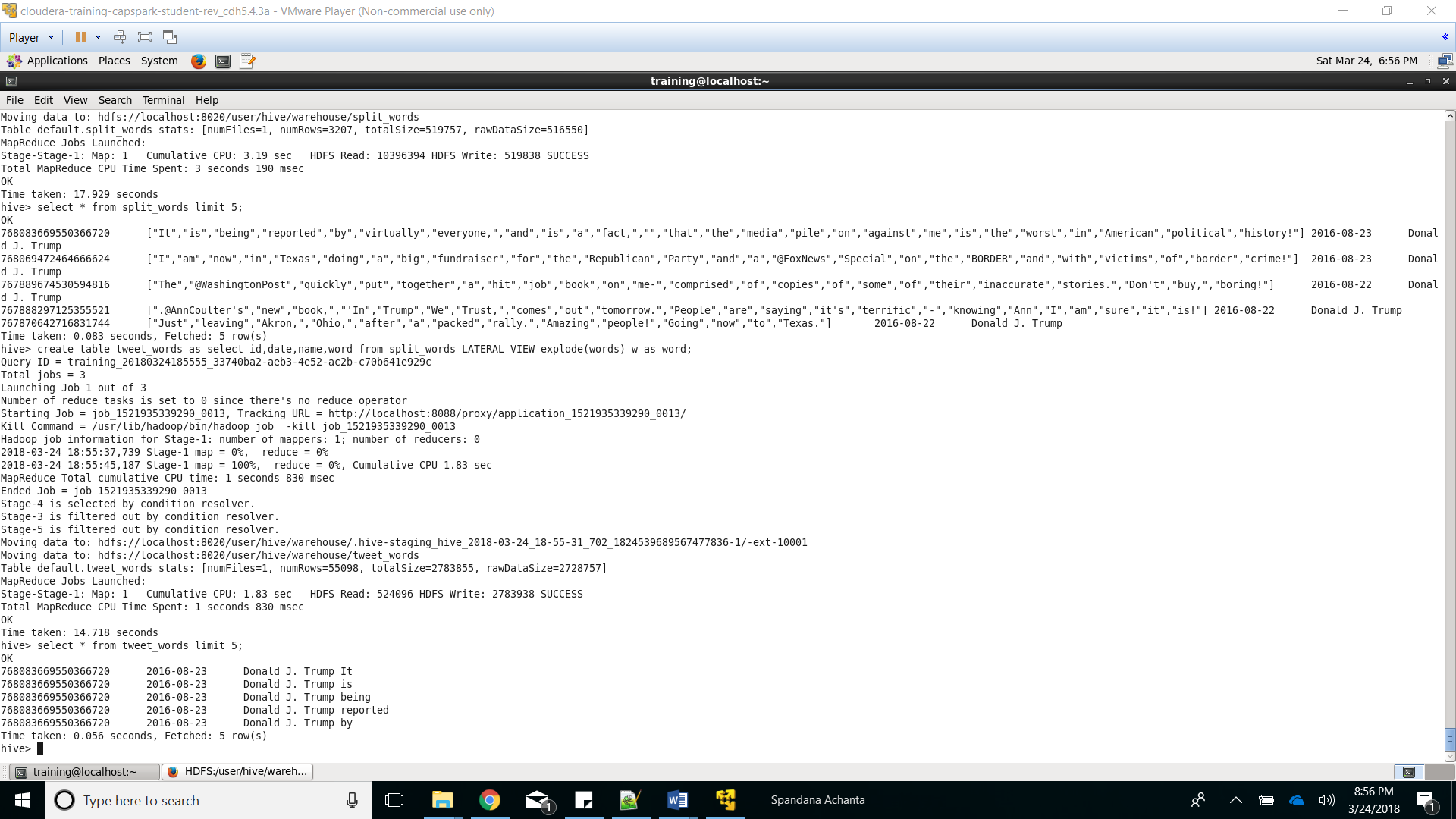
create table word\_join as select tweet\_words.id,tweet\_words.word,tweet\_words.date,tweet\_words.name,dictionary.score from dictionary RIGHT OUTER JOIN tweet\_words ON dictionary.word=tweet\_words.word;

select \* from word\_join limit 5;

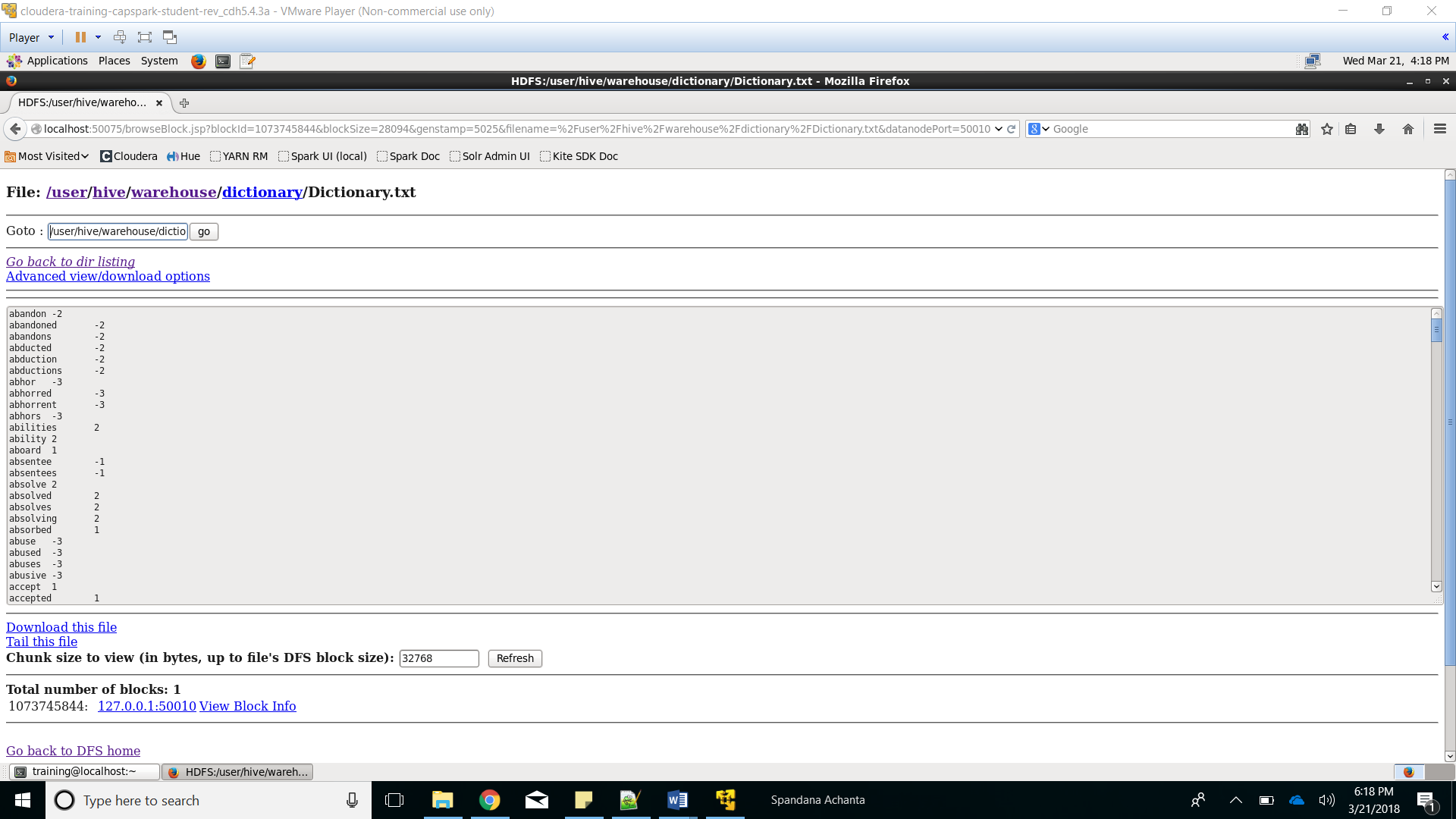
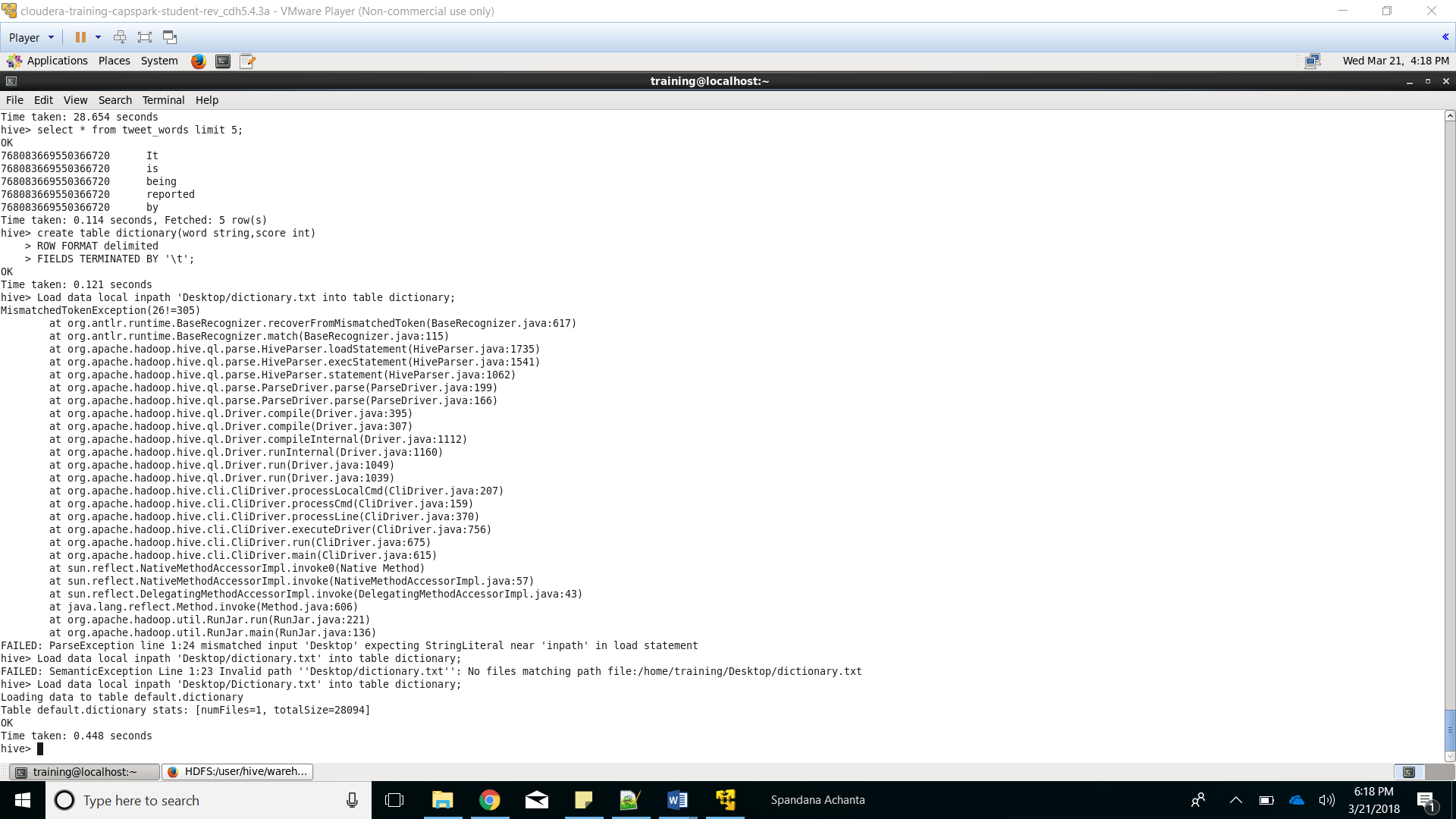
select id,name,date,AVG(score) as rating from word\_join group by id,date,name order by rating desc;



Created a table called split\_words that stores tweet id and words from each tweet collected in the form of an array



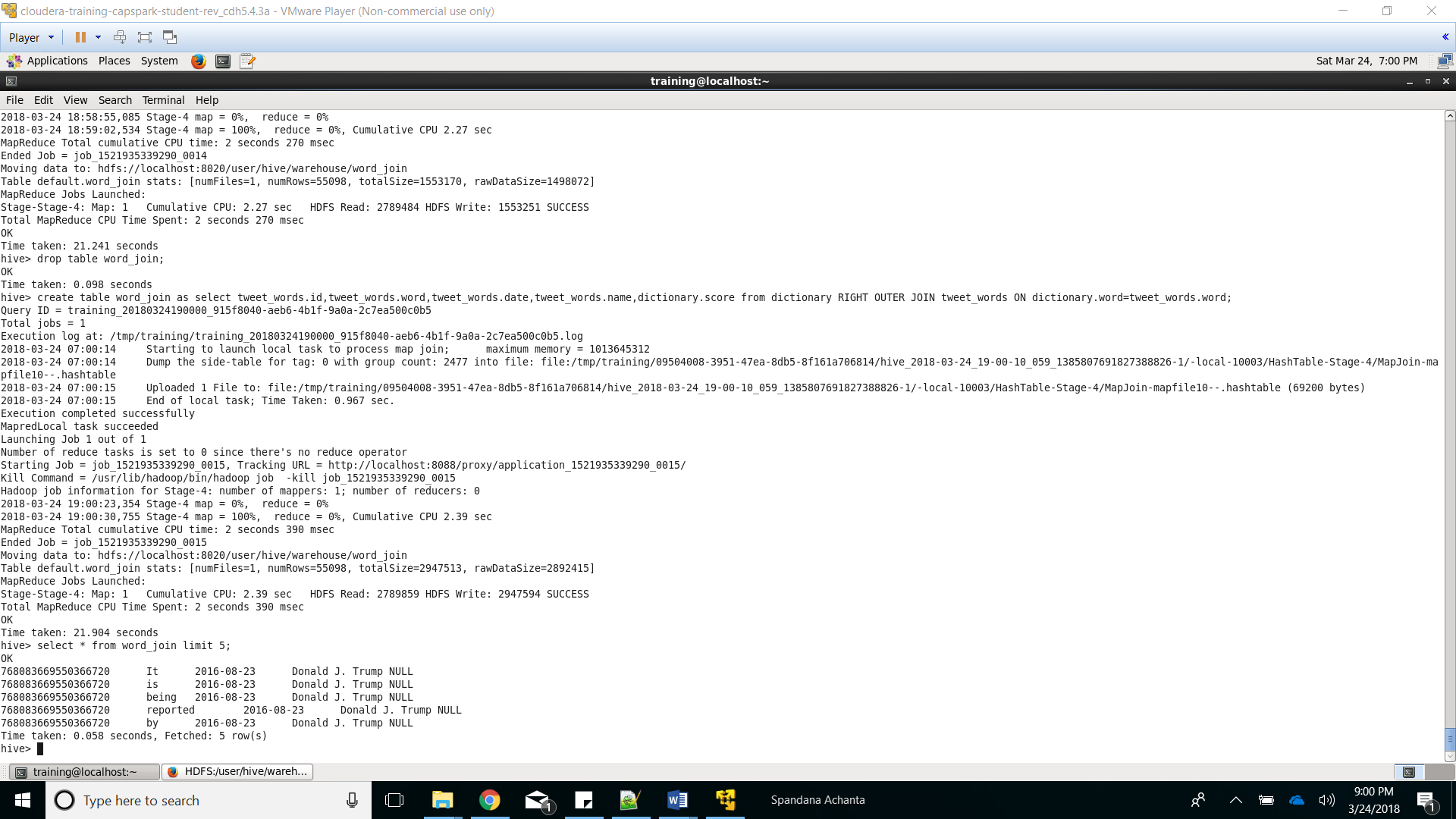
Created a table named tweet\_words that stored each word of each tweet in a different row



Validating the creation and loading of dictionary data

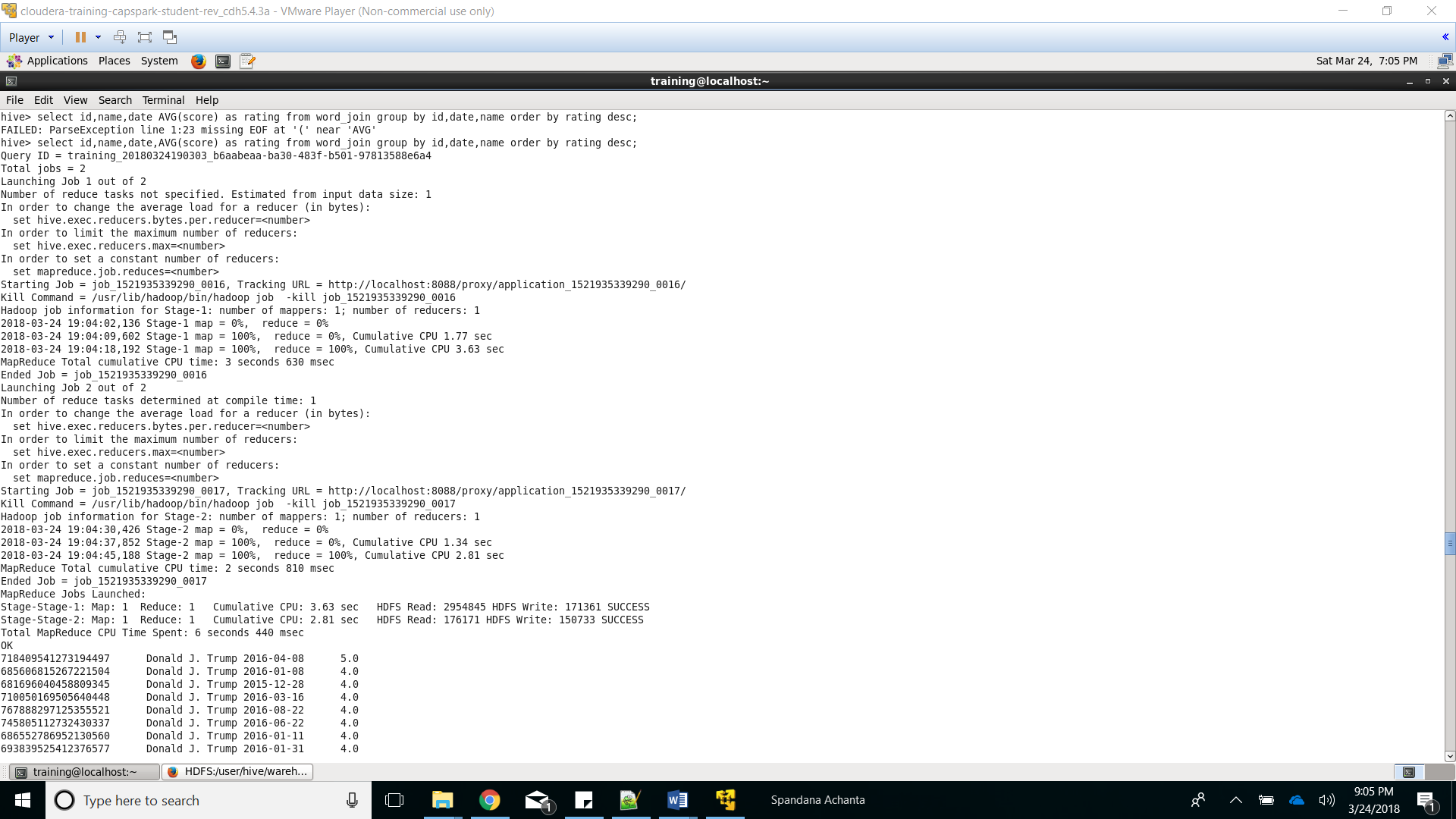
Loaded the Dictionary.txt file into the created table

Created table to store the dictionary containing for rating of words



Verifying contents of the join table

Created a join between tweet\_words and dictionary



Displaying the result of the select query

Calculating the average rating for each tweet based on score for each word in the tweet

Create table Final\_sentiment AS

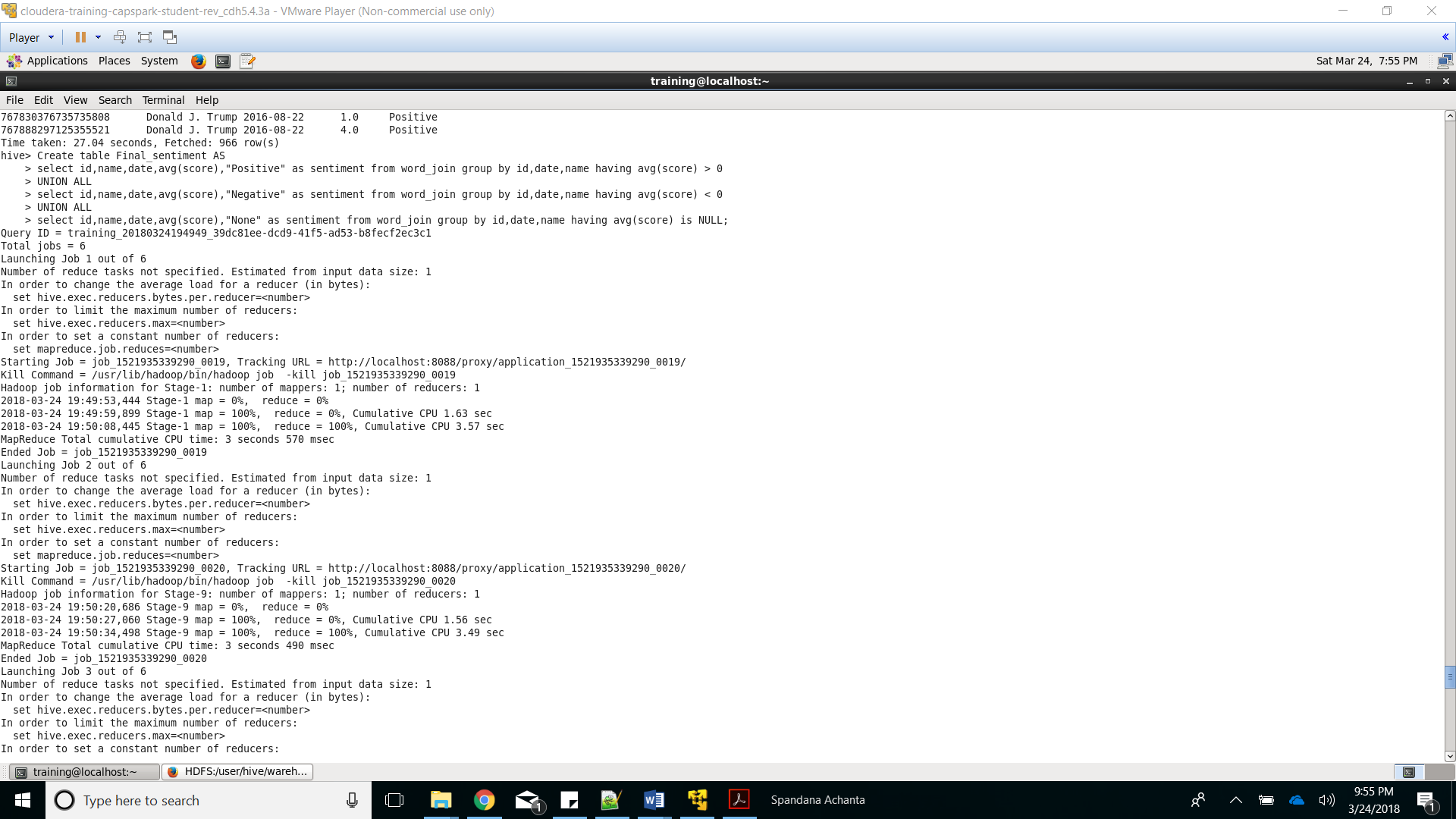
select id,name,date,avg(score),"Positive" as sentiment from word\_join group by id,date,name having avg(score) > 0

UNION ALL

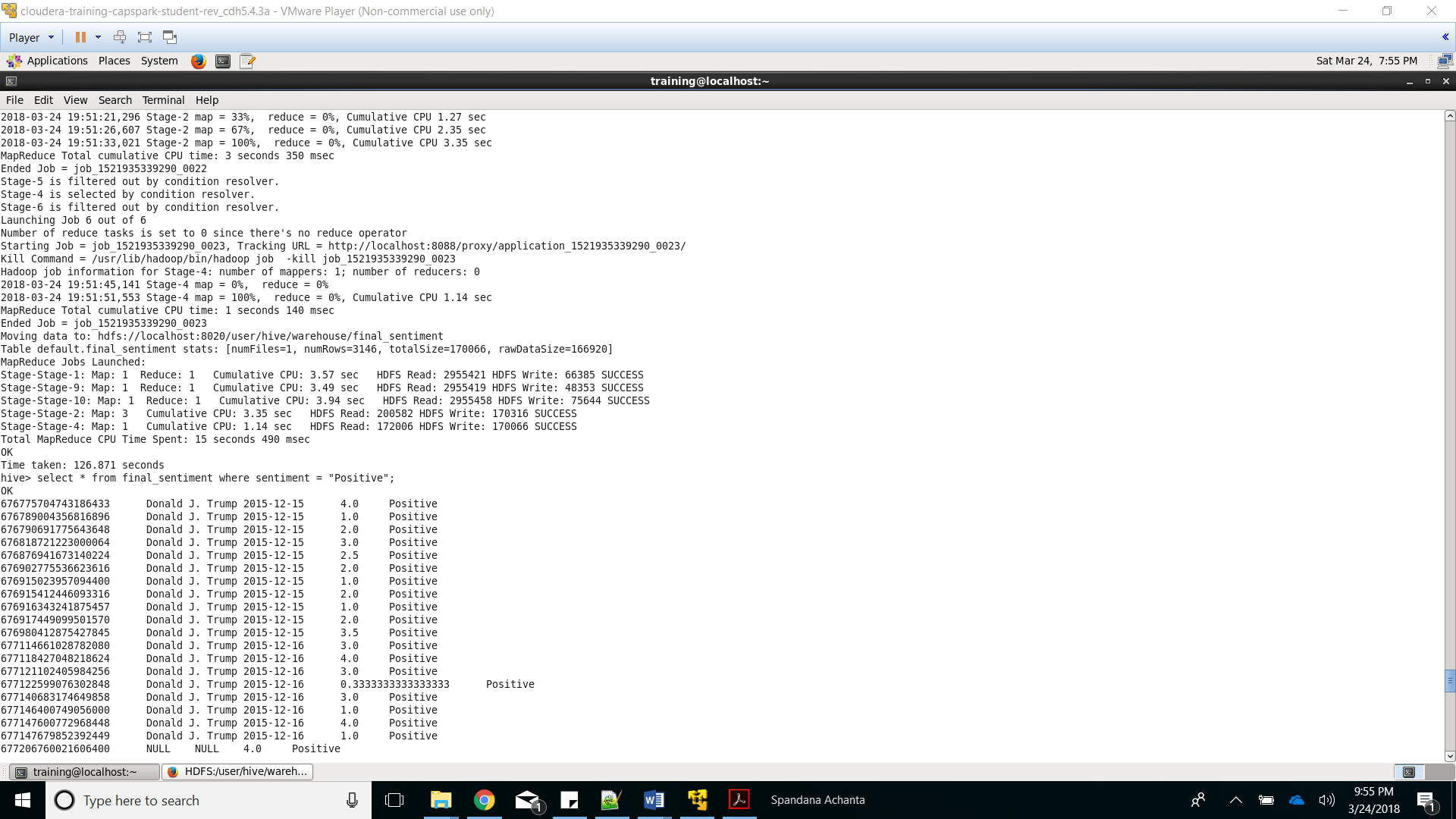
select id,name,date,avg(score),"Negative" as sentiment from word\_join group by id,date,name having avg(score) < 0

UNION ALL

select id,name,date,avg(score),"None" as sentiment from word\_join group by id,date,name having avg(score) is NULL;



Created a table name ‘Final\_Sentiment’ to store the result of previous query with a new column sentiment showing positive negative or None



Verifying contents of the Final\_Sentiment Table