**Twitter Data Analysis**

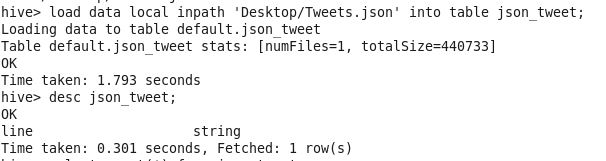
**Venkata Naga Satya Sangeeta Eluri**

**vxe160230**

**1(a) What are the hashtags tweeted and how many times are they used?**

Initially, we create a table in hive – named “json\_tweet” and load “Tweets.json” file in to it.





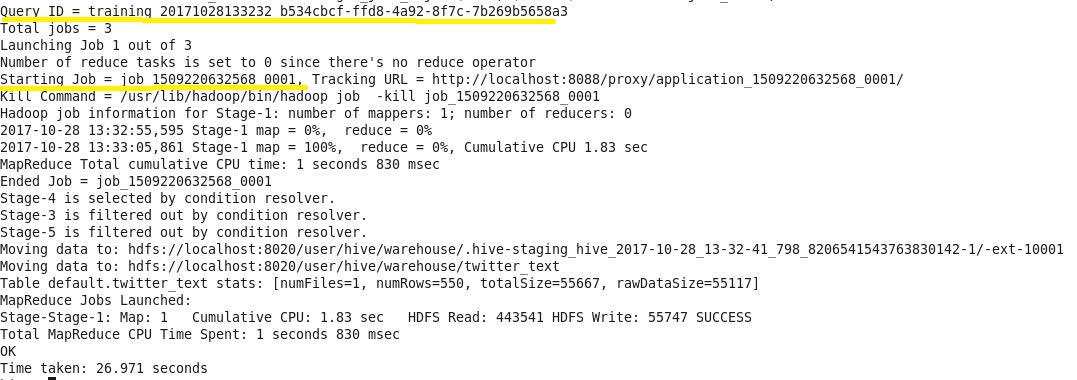
**Steps:**

1. Extract “text” from tweets data into a table called – “twitter\_text”

**Creation:**

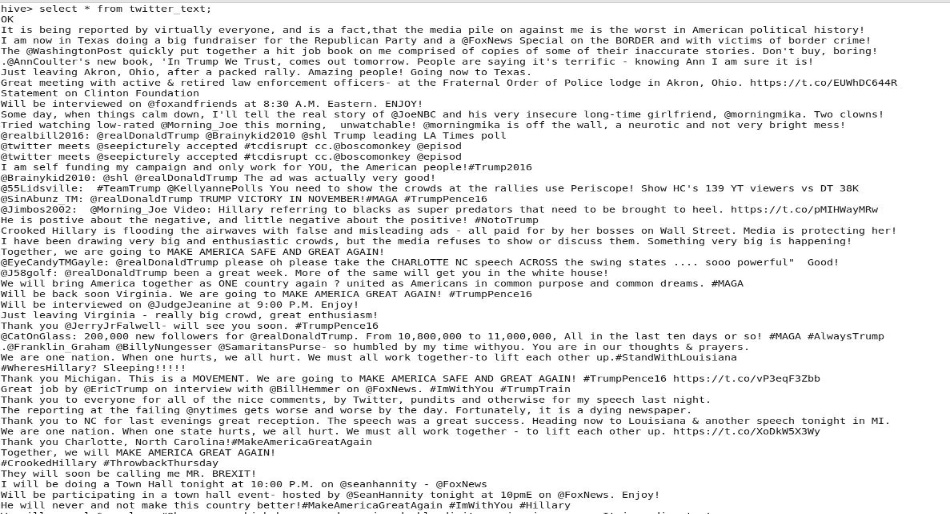
***Create table twitter\_text as select get\_json\_object(line,’$.text’) as text from json\_tweet;***





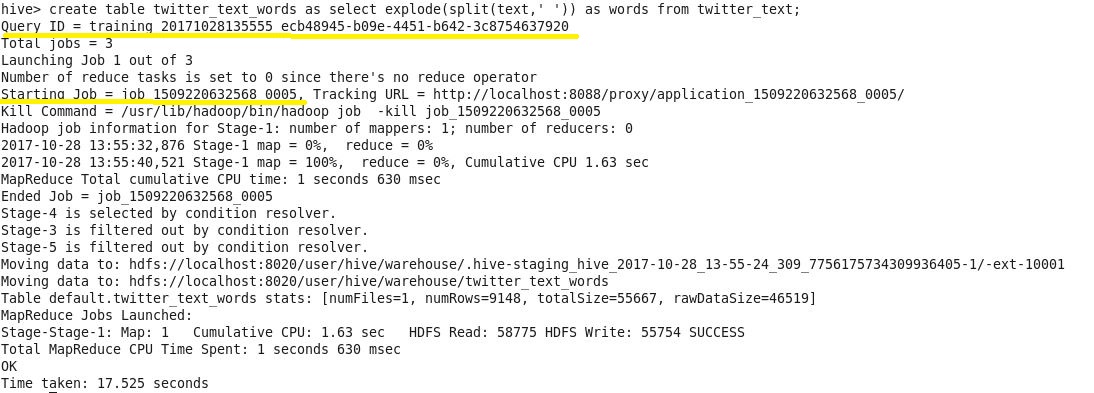
**Output:**

***Select \* from twitter\_text;***



2. Split “text” into words – stored in table name “twitter\_text\_words”

***Create table twitter\_text\_words as select explode(split(text,’ ‘)) as words from twitter\_text;***



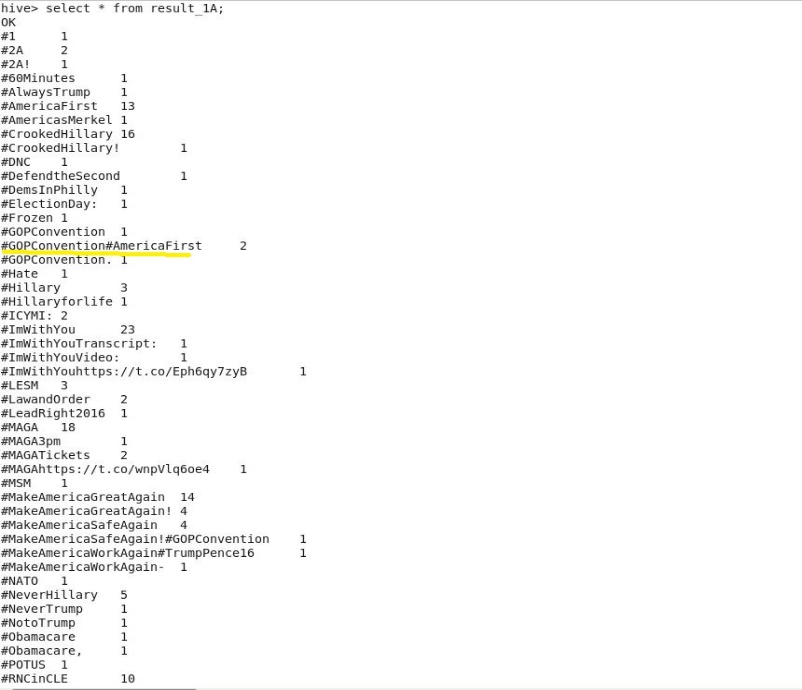
3. Retrieve all words that have hashtags – stored in table “hashtags”

***Create table hashtags as select \* from twitter\_text\_words where words RLIKE ‘^#’;***



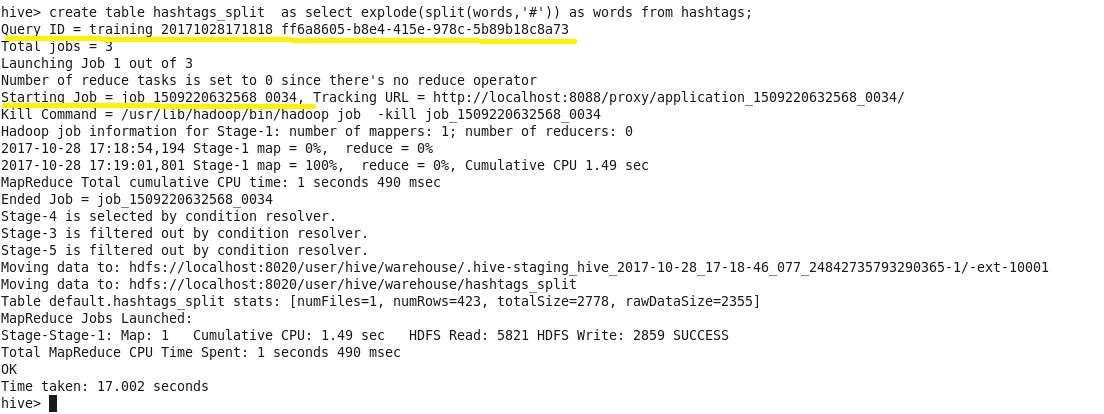
4. However, the result of table “hashtags” has got a few words with #tags without a space. Therefore, we need to split these words further by ‘#’ to get all of the hashtaged words and store in table – “hashtags\_split”. Find screenshot **below.**

***Select \* from result\_1A;***

****

**Table creation with required words :**

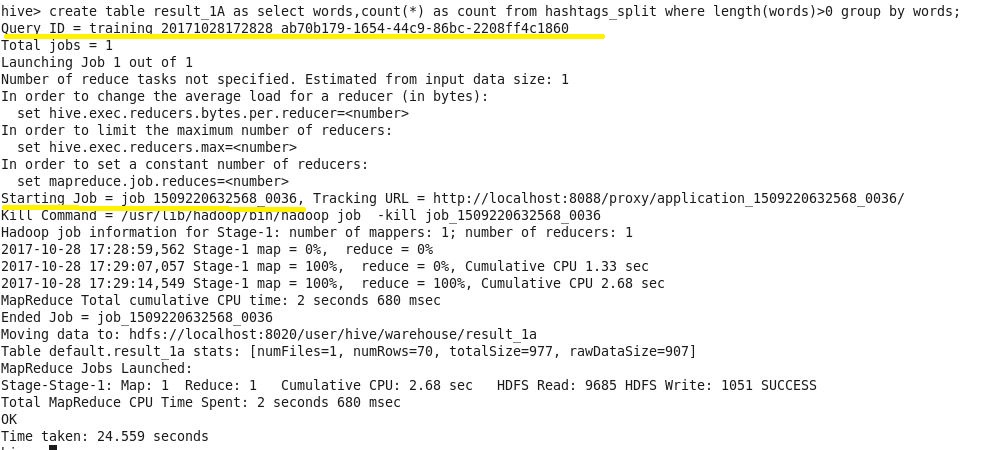
***Create table hashtags\_split as select explode(split(words,’#’)) as words from hashtags;***



5. **Result** – hashtags with respective counts are stored in “result\_1A” table:

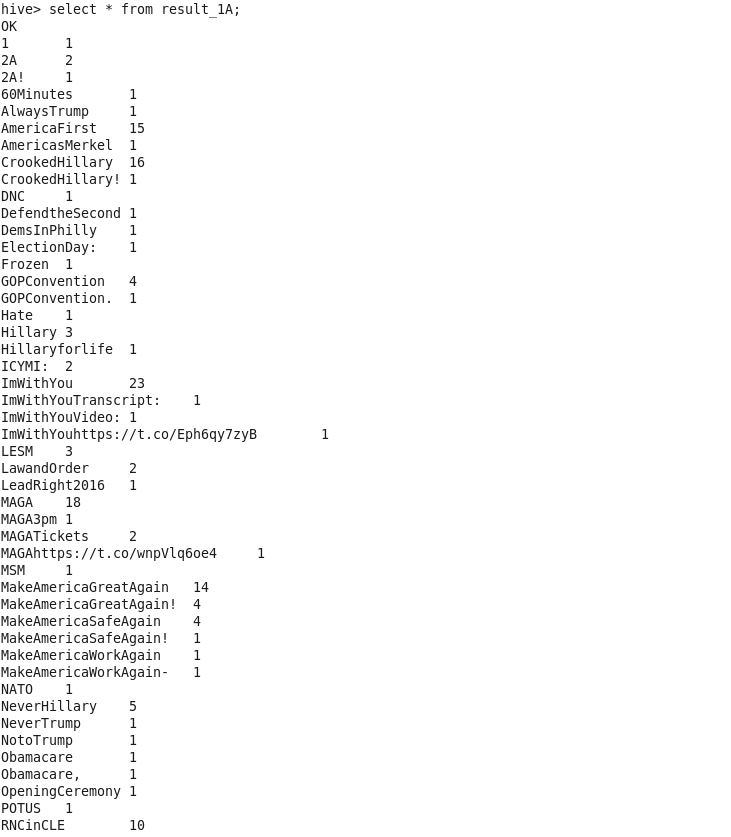
**Creation:**

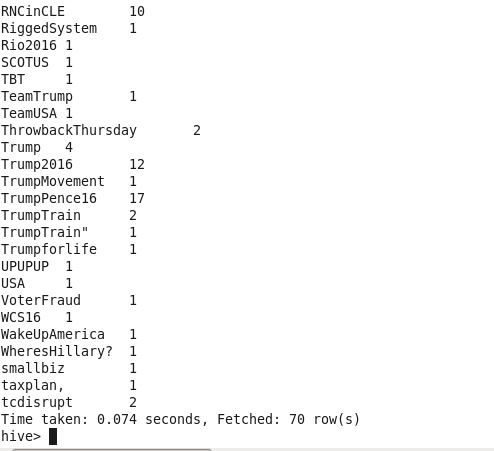
***Create table result\_1A as select words,count(\*) as count from hashtags\_split where length(words)>0 group by words;***



**Output:**

***Select \* from result\_1A;***





**Table creation with required fileds for use in following questions.**

**---** Table creation – “twitter\_data” , with fileds – “id, created\_at,text,favourites, location”

***Create table twitter\_data as select get\_json\_object(line,’$.id’) as id, get\_json\_object(line,’$.created\_at’) as created\_at, get\_json\_object(line,’$.user.location’) as location, get\_json\_object(line,’$.user.favourites\_count’) as favourites, get\_json\_object(line,’$.text’) as text from json\_tweet;***

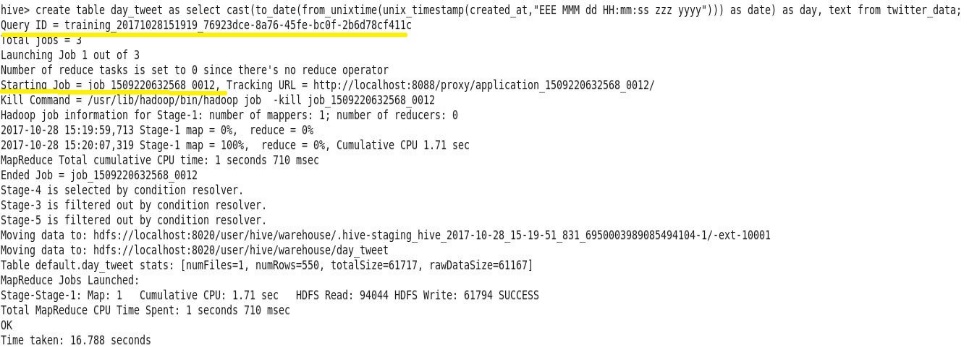


**1(b)** What is the most trending hashtag in a day and how many times are they tweeted? [Note: day should be in the format ‘yyyy-mm-dd’]

**Steps:**

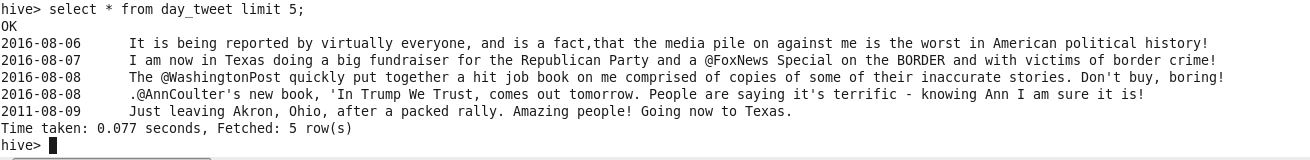
1. Create a table “day\_tweet” with tweeted text and tweet date, as required.

***Create table day\_tweet as select cast(to\_date(from\_unixtime(unix\_timestamp(created\_at,”EEE MMM dd HH:mm:ss zzz yyyy”))) as date) as day, text from twitter\_data;***



**Output:**

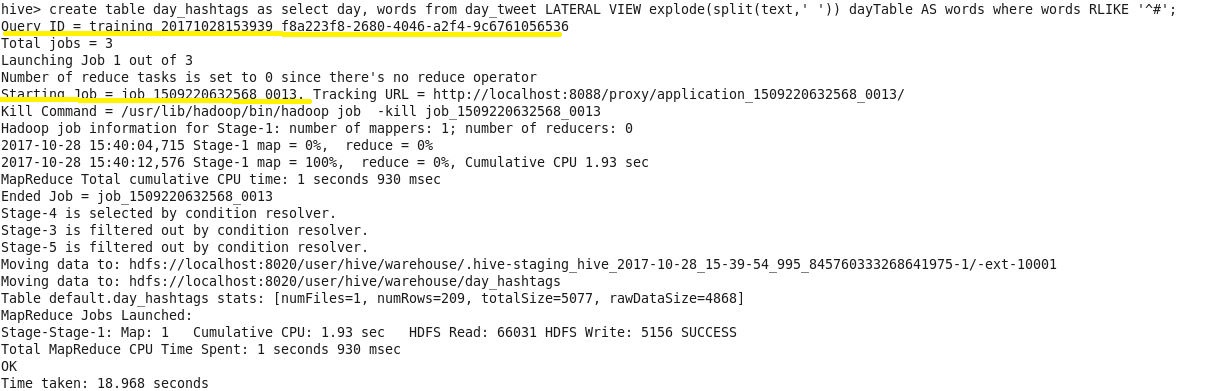
***Select \* from day\_tweet limit 5;***



2. Create table – “day\_hashtags” that has hashtagged words. However, there are a few hashtags that are without a space. Therefore, we have another table – “day\_hashtags\_split” that has our final list of words along with created dates.

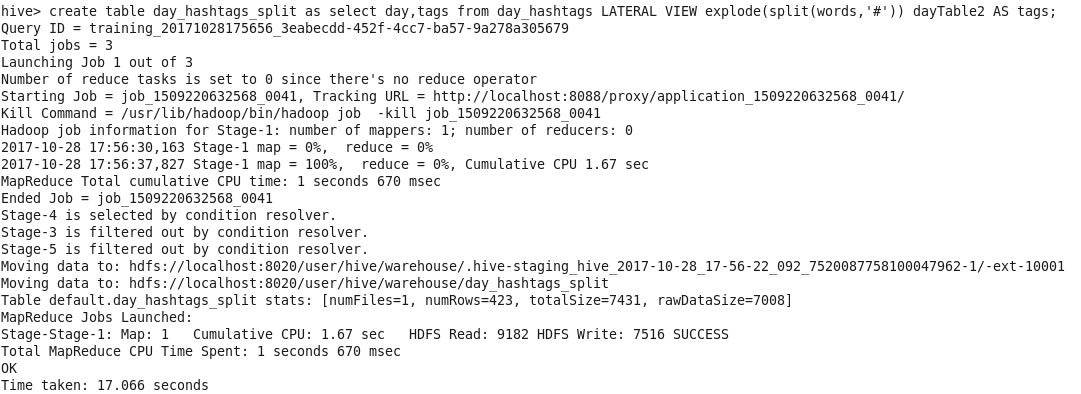
**“day\_hashtags” – table creation**

***Create table day\_hashtags as select day, words from day\_tweet LATERAL VIEW explode(split(text,’ ‘)) dayTable AS words where words RLIKE ‘^#’;***



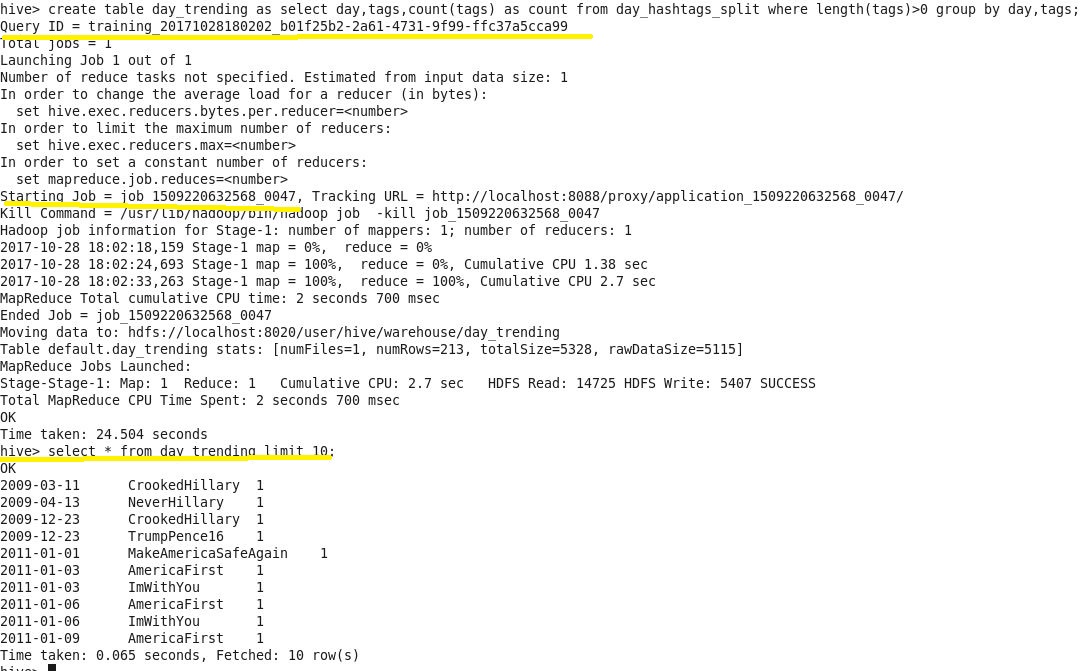
**“day\_hashtags\_split” – table creation**

***Create table day\_hashtags\_split as select day, tags from day\_hashtags LATERAL VIEW explode(split(words,’# ‘)) dayTable2 AS tags;***

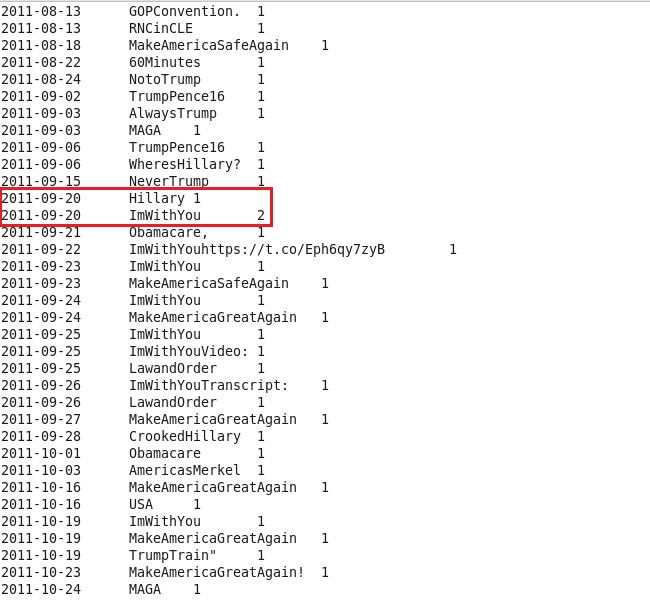


3. We have a table that consists of hashtag, created day and a column which gives count of a unique day and hashtag combination. – “day\_trending”

***Create table day\_trending as select day,tags,count(tags) as count from day\_hashtags\_split where length(tags)>0 group by day,tags;***



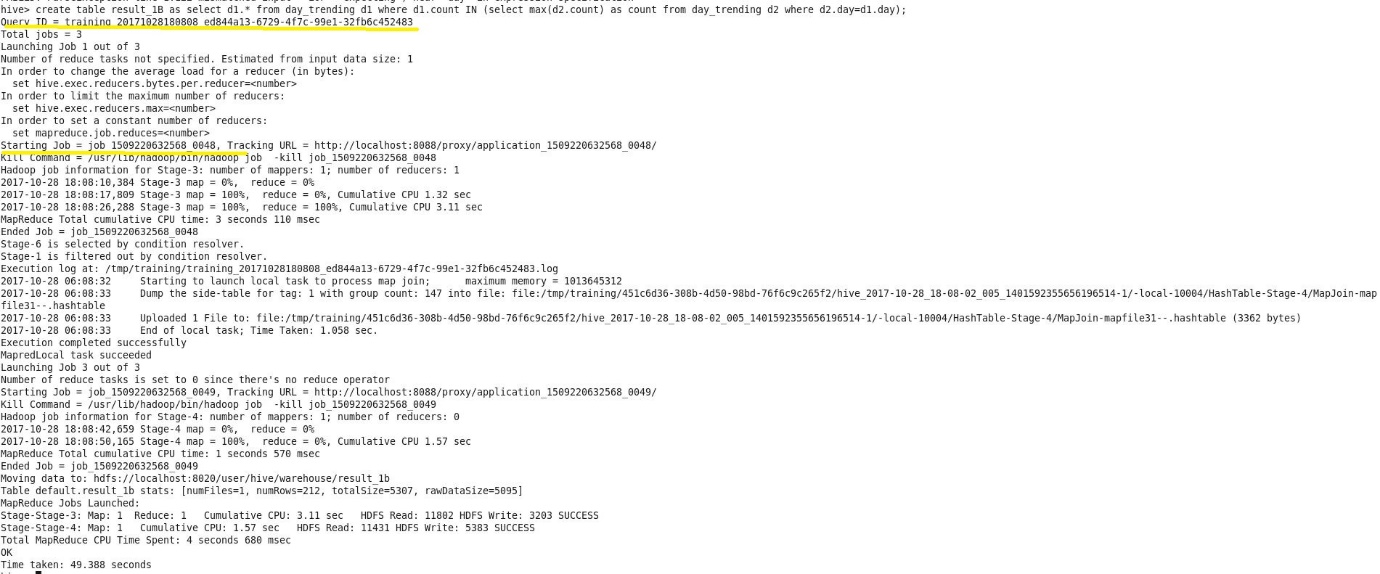
**\*\* we have same word repeated in a day on “2011-09-20”. Except for this date, other dates have a hashtag tweeted only once. (see screenshot below)**



4. Finally, we need to get a hashtag, which trends most for a day. As far as data is concerned, the combination of day and a hashtag is repeated mostly, once. Therefore, most of the result has “1” as count of most trended word. And, each day has more than one trending word.

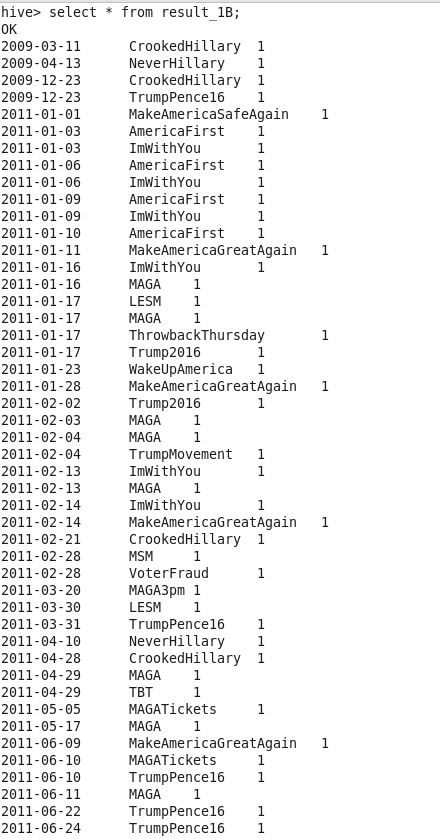
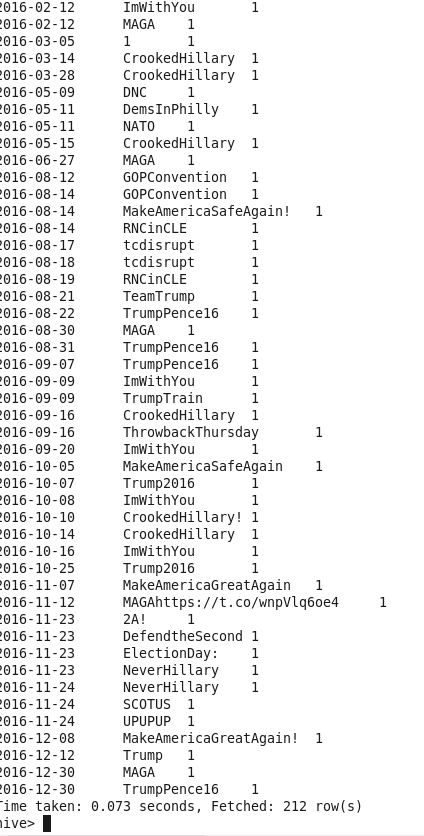
Now, create a final result table – “result\_1B” that has hashtags, day and trending word for the day(count column)

***Create table result\_1B as select d1.\* from day\_trending d1 where d1.count IN(select max(d2.count) as count from day\_trending d2 where d2.day=d1.day);***

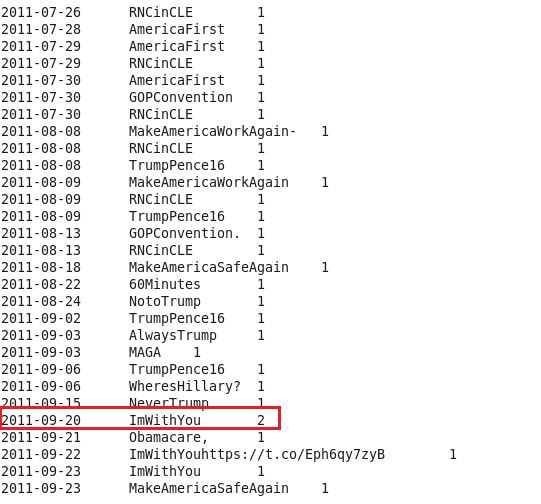


**Output:**

***Select \* from result\_1B;***

**\*\* now we can see that the date “2011-09-20” has only the hashtag that is repeated most/trended most.**

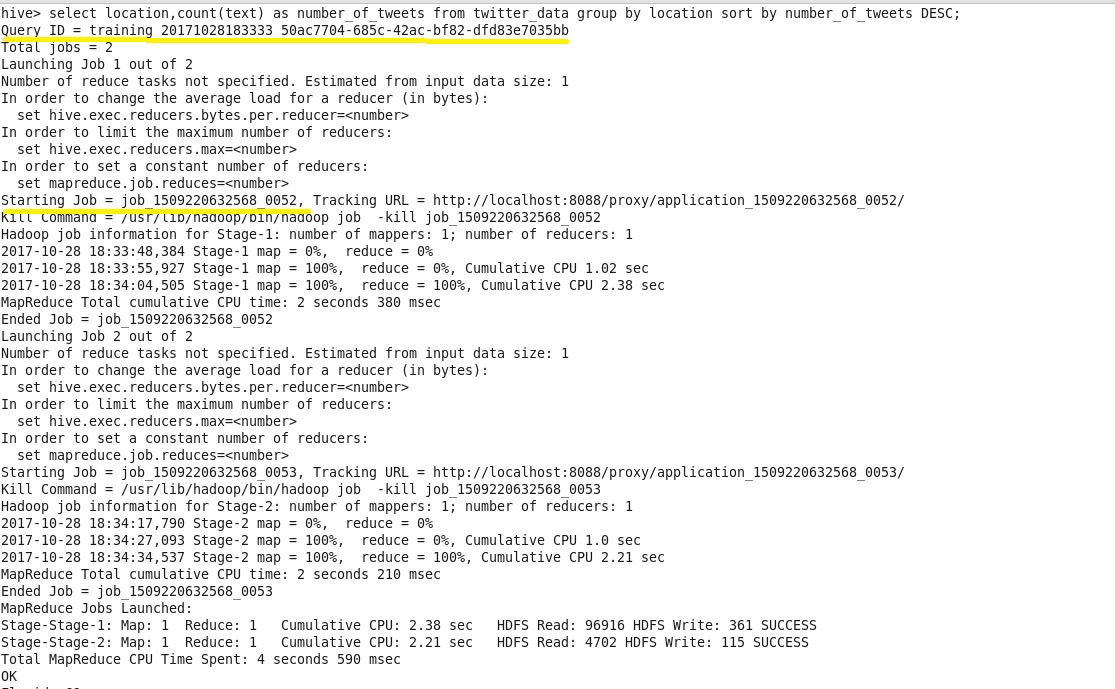


**1(C)** Which state users are most active, and how many tweets are posted by them?

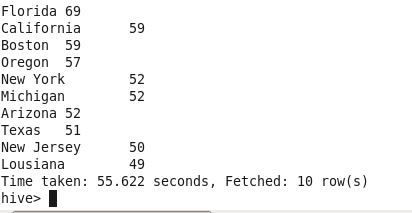
**Steps:**

1. **User activity** is known by the **number of tweets tweeted by them**. Similarly, to find the state that is most active, we extract “location and count(text)” that helps to answer the question.

***Select location, count(text) as number\_of\_tweets from twitter\_data group by location sort by number\_of\_tweets DESC;***



**Output:**



\*\* Users of **“Florida” state** are most active, and they have posted **69 tweets**.

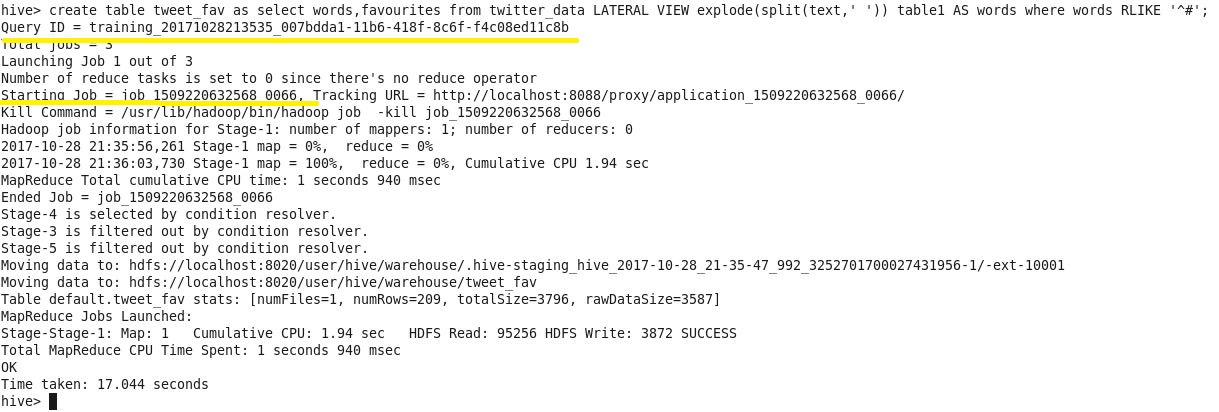
**1(d)** What is the total number of favourites received for each hashtag?

**Steps:**

1. We extra hashtags and count favourites with respect to them.

Initially, get text (in words format) and favourites in to a table – “tweet\_fav”

***Create table tweet\_fav as select words,favourites from twitter\_data LATERAL VIEW explode(split(text,’ ‘)) table1 AS words where words RLIKE ‘^#’;***



1. Since we have a few hashtags without space, we split the selected words based on’#’, now.

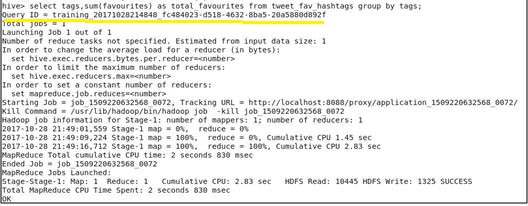
These words with favourites are saved in a table – ‘tweet\_fav\_hashtags’

***Create table tweet\_fav\_hashtags as select tags,favourites from tweet\_fav LATERAL VIEW explode(split(words,’#’)) table2 AS tags where length(tags)>0;***

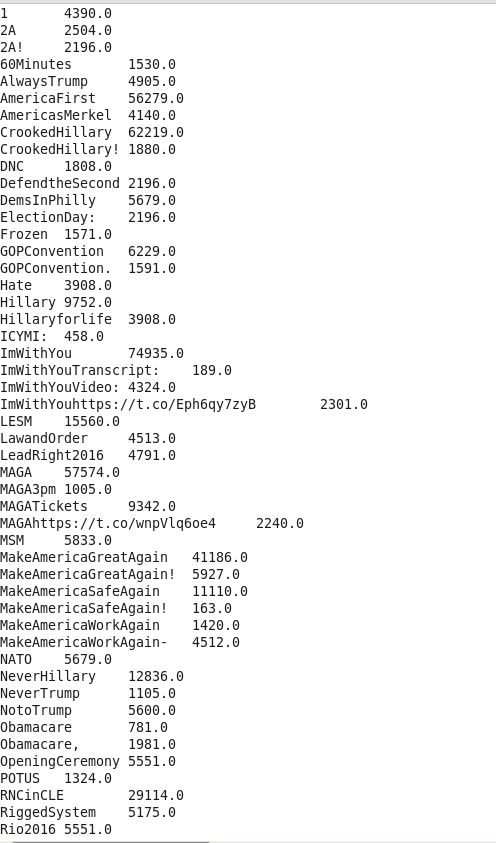
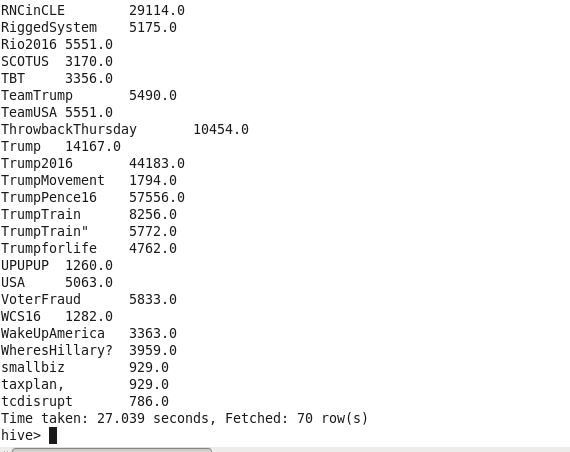


1. Finally, we can retrieve hashtag and its favourite count using the following:

***Select tags, sum(favourites) as total\_favourites from tweet\_fav\_hashtags group by tags;***



**Output:**

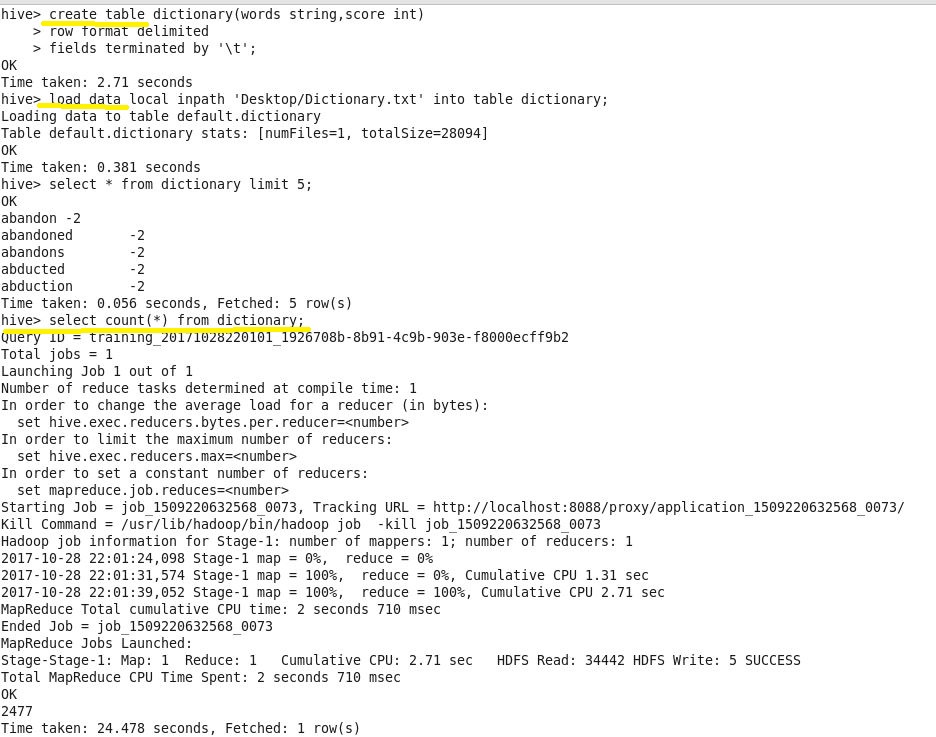
**1(e)** Does each tweet have a positive or negative sentiment?

Note: Include the date in the format ’yyyy-mm-dd’, with tweet id, user name and the score.

**Steps:**

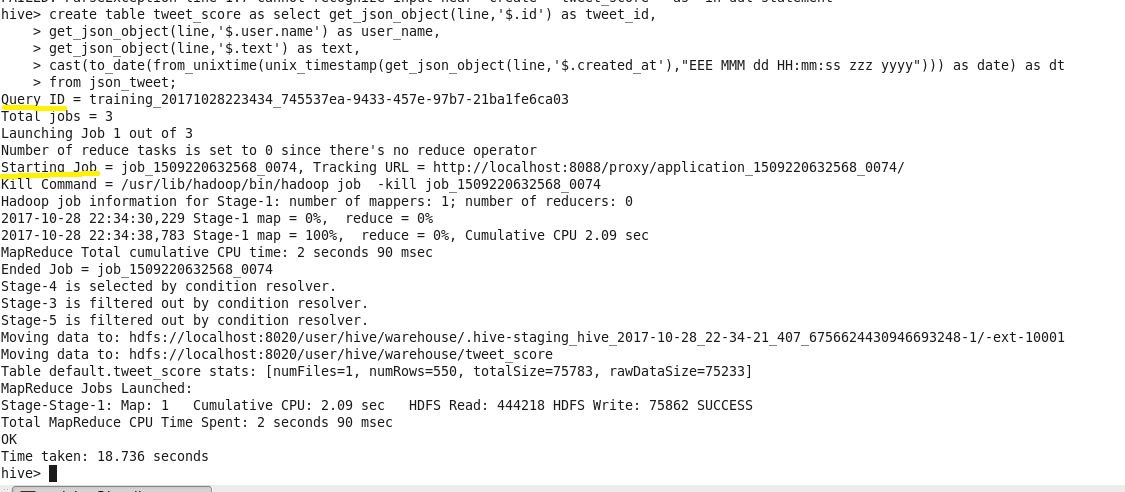
1. Initially, create a dictionary table and load dictionary data in to it – “dictionary”

***Create table dictionary(words string, score int);***



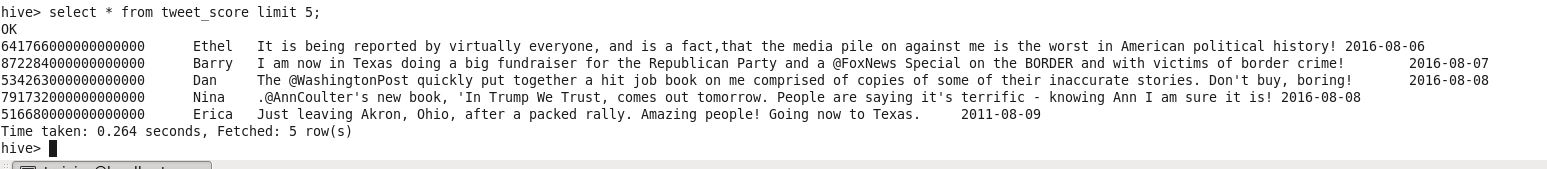
2. Create a table “tweet\_score” that has the required fields (text, created\_at, tweet\_id, user\_name) from twitter data file.

***Create table tweet\_score as select get\_json\_object(line,’$.id’) as tweet\_id, get\_json\_object(line,’$.user.name’) as user\_name, get\_json\_object(line,’$.text’) as text, cast(to\_date(from\_unixtime(unix\_timestamp(get\_json\_object(line,’$.created\_at’),”EEE MMM dd HH:mm:ss zzz yyyy”))) as date) as dt from json\_tweet;***



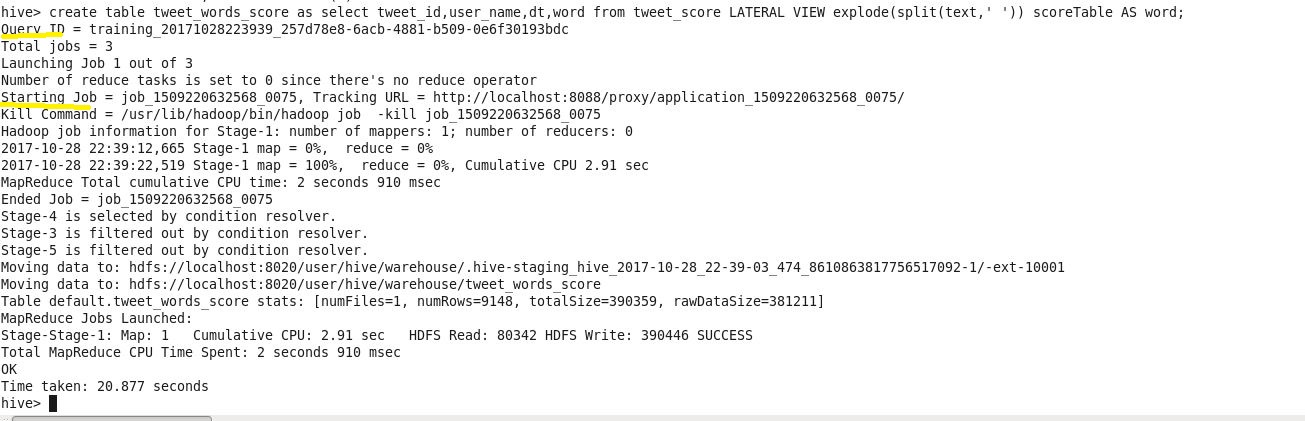
**Output:**

***Select \* from tweet\_score limit 5;***



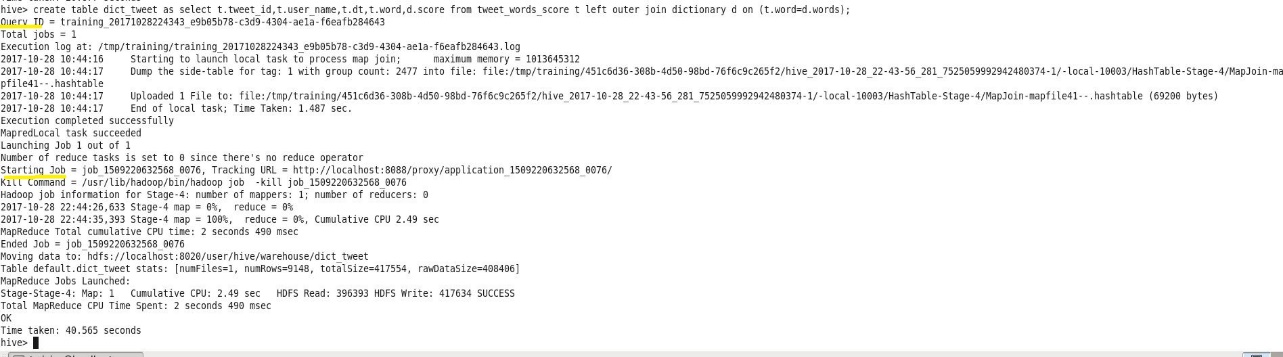
3. Create a table “tweet\_words\_Score” that has text in form of words, along with other required fields.

***Create table tweet\_words\_score as select tweet\_id,user\_name,dt,word from tweet\_score LATERAL VIEW explode(split(text,’ ‘)) scoreTable AS word;***



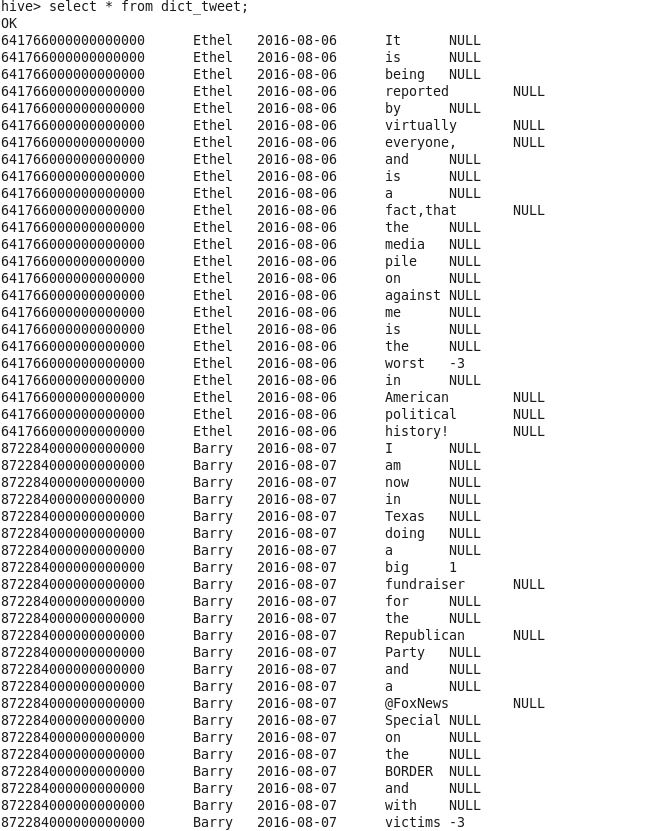
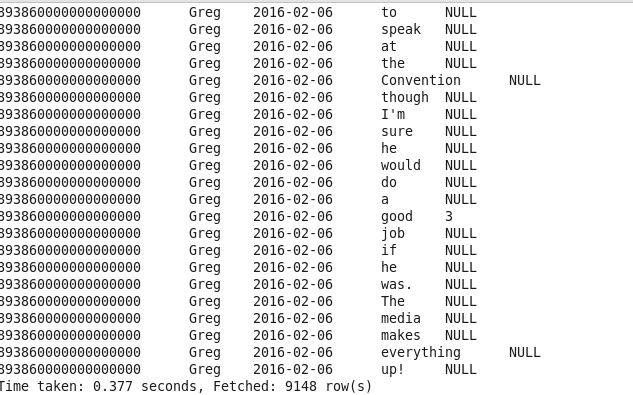
4. Create a table “dict\_tweet” that culminates twitter data with that of dictionary data. This helps us to obtain a score of each word, which is obtained in step-3.

***Create table dict\_tweet as select t.tweet\_id,t.user\_name,t.dt,t.word,d.score from tweet\_words\_score t left outer join dictionary d on (t.word=d.words);***



**Output:**

***Select \* from dict\_tweet;***

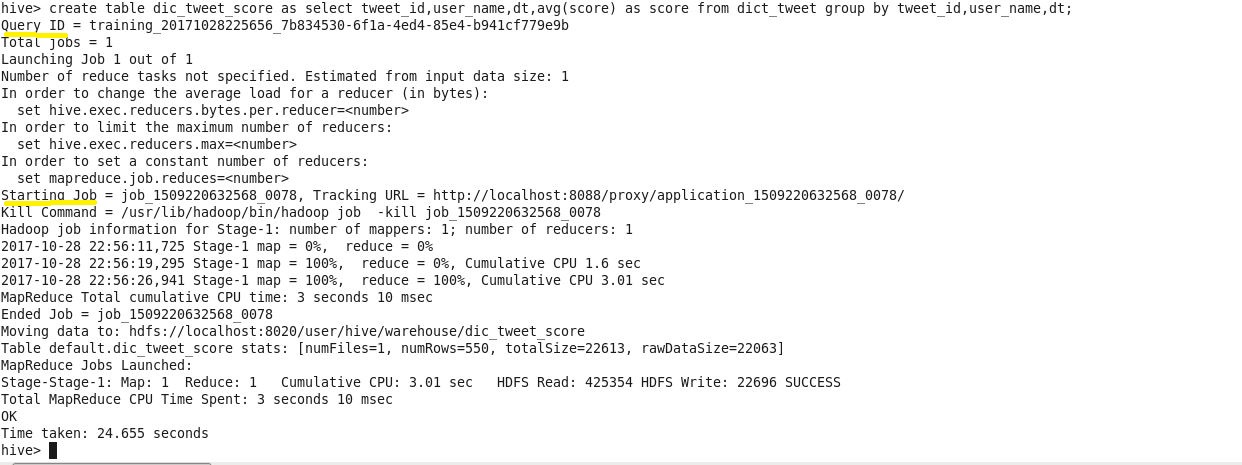
5. In order to obtain score of each tweet, we need to find score of each word and perform a aggregation on the words of text.

**Score of words – step4**

Score of tweet is found as follows: (table – “dic\_tweet\_score”)

***Create table dict\_tweet\_score as select tweet\_id,user\_name,dt,avg(score) as score from dict\_tweet group by tweet\_id,user\_name,dt;***

***\*\* average or sum could be used. Either of them will give us same call/insight(sentiment score analysis)***



6. Finally, we associate positive or negative sentiment to tweets, as follows. (table – “tweet\_sentiment”)

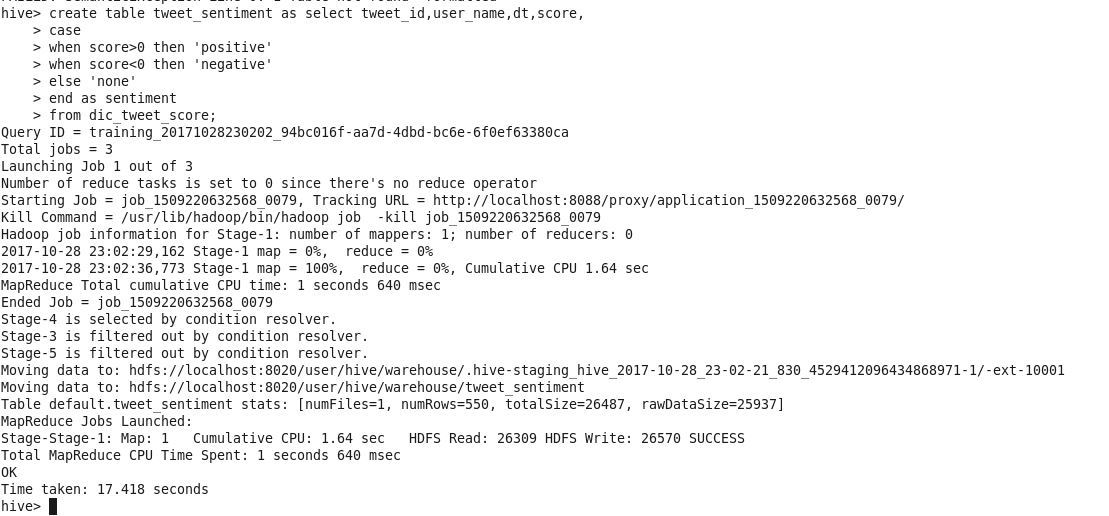
***Create table tweet\_sentiment as select tweet\_id,user\_name,dt,score, case***

***When score>0 then ‘positive’***

***When score<0 then ‘negative’***

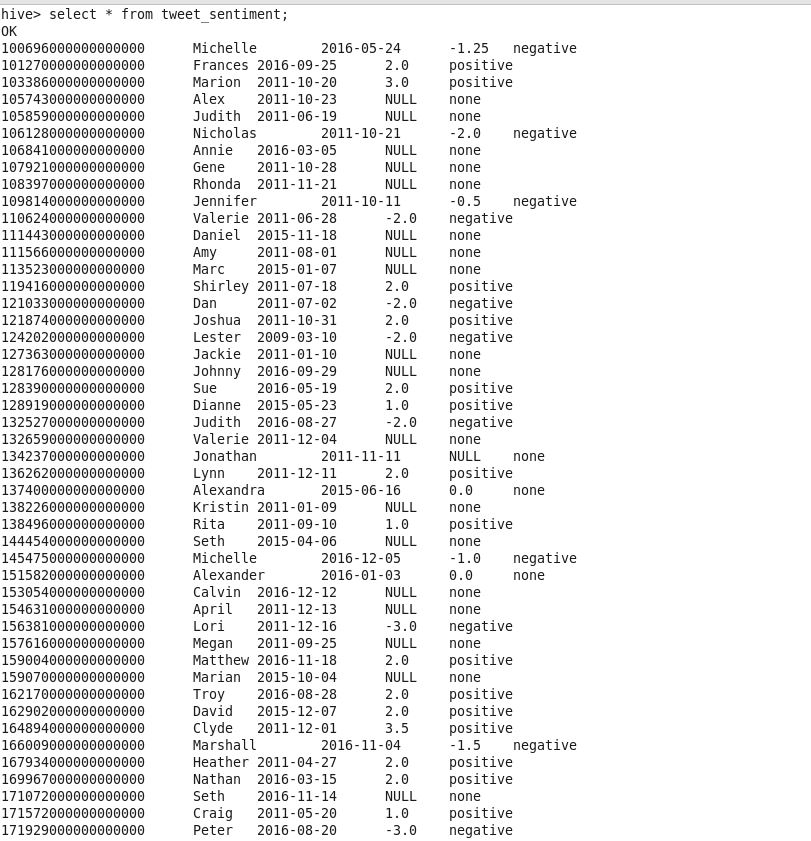
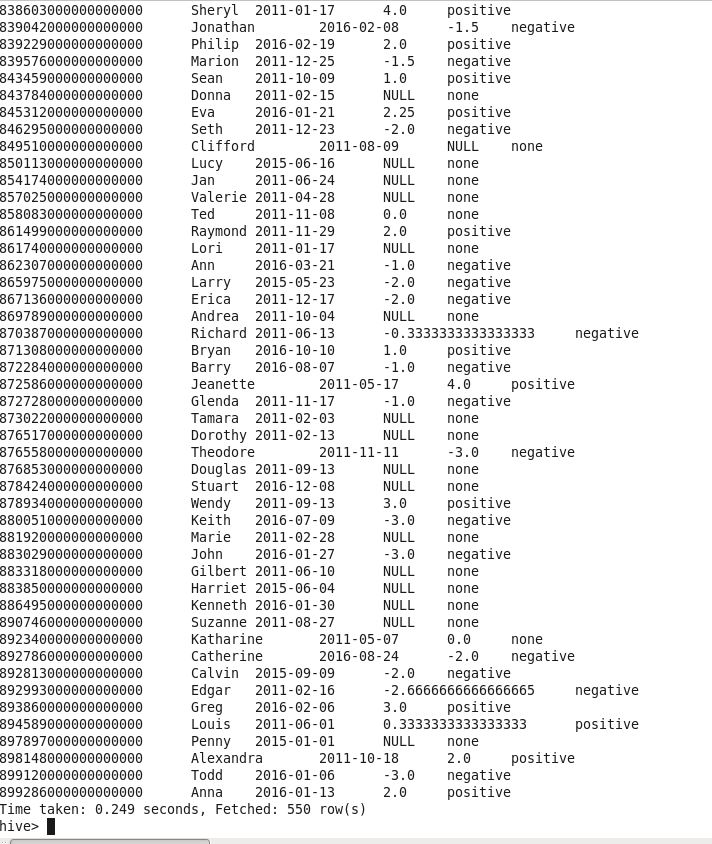
***else ‘none’***

***end as sentiment from dict\_tweet\_score;***



**Output:**

***Select \* from tweet\_sentiment;***

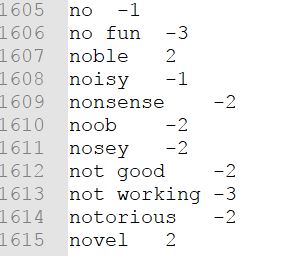
**1(f)** Do you believe that the process outlined above has issues? If so, briefly explain how the process can be improved.

**Yes, I believe so.**

* **Firstly,** the dictionary file does not have considerable/required words to associate appropriate sentiment to each tweet.
* There are many words in a tweet that are not mapped in the dictionary of words. Therefore, such words have a “None” sentiment associated with them. This has an impact on entire tweet sentiment, while we find average score value of a tweet.
* As **success of analysis lies in the data**, to get hold of this (above mentioned) issue**, the dictionary file needs to have extensive set of words.**
* **Next,** the words are associated a score based on what they mean in general sense. However, these words might have a different meaning in the content. **Hence, the score of the word needs to be contextual rather than generic. Since correlation is never a causation, one needs to have data and make a granular contextual analysis to get solutions/results.**
* For example, “prevent” has got negative score. However, it could have a positive meaning in a situation – “prevents fever”. Further, “playful” might have negative meaning, but the dictionary file has given a positive score to it.
* Similarly, **the word “not” has not got any score.**

Word obscene has got negative score. Usage of “not obscene” will make the meaning/sense positive. But, as there is no score associated with “not”, the negative score obscene has a clearly, big impact on a tweet. This could be overcomed for a better and realistic analysis.

**Screenshots:**

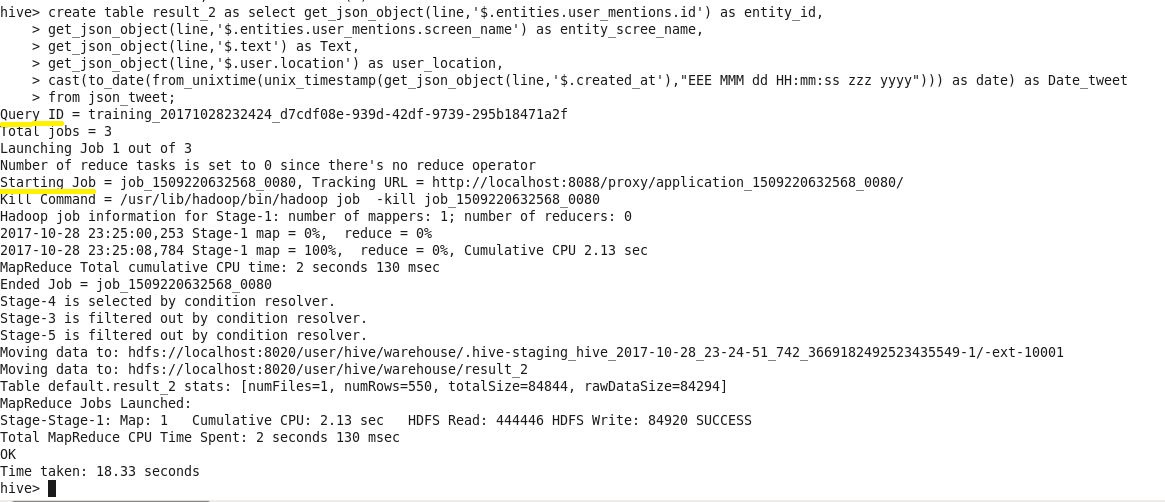
  

2. Twitter data comprises data types that are hard to handle at times. Using the same twitter data, display all the rows with the following columns: entity Id, entity screen\_name, Text, Date, user location, date.

**Steps:**

* As the twitter data is stored in “json\_tweet” table, we make use of the same to retrieve our required data. The result is stored in a table named “result\_2”

***Create table result\_2 as select get\_json\_object(line,’$.entities.user\_mentions.id’) as entry\_id, get\_json\_object(line,’$.entities.user\_mentions.screen\_name’) as entity\_screen\_name, get\_json\_object(line,’$.text’) as Text, get\_json\_object(line,’$.user.location’) as location, cast(to\_date(from\_unixtime(unix\_timestamp(get\_json\_object(line,’$.created\_at’),”EEE MMM dd HH:mm:ss zzz yyyy”))) as date) as Date\_tweet from json\_tweet;***



**Output:** For example, Output of the above is limited to 40 rows



**NOTE:**

By using the following setting in hive,

**set hive.cli.print.header=true**;

We can have column names displayed, along with the data(rows).