**Steps Involved:**

1. **Acquiring top 200,00 records from Stackexchange based on View Count**

<Kindly refer to other attached files for codes, commands and queries>

4 queries were run one by one on data.stackexchange.com to get top 200,000 posts ordered by decreasing order of ViewCount.

1. **ETL using Pig**

Step 1 : We upload the 4 CSV files to the cluster

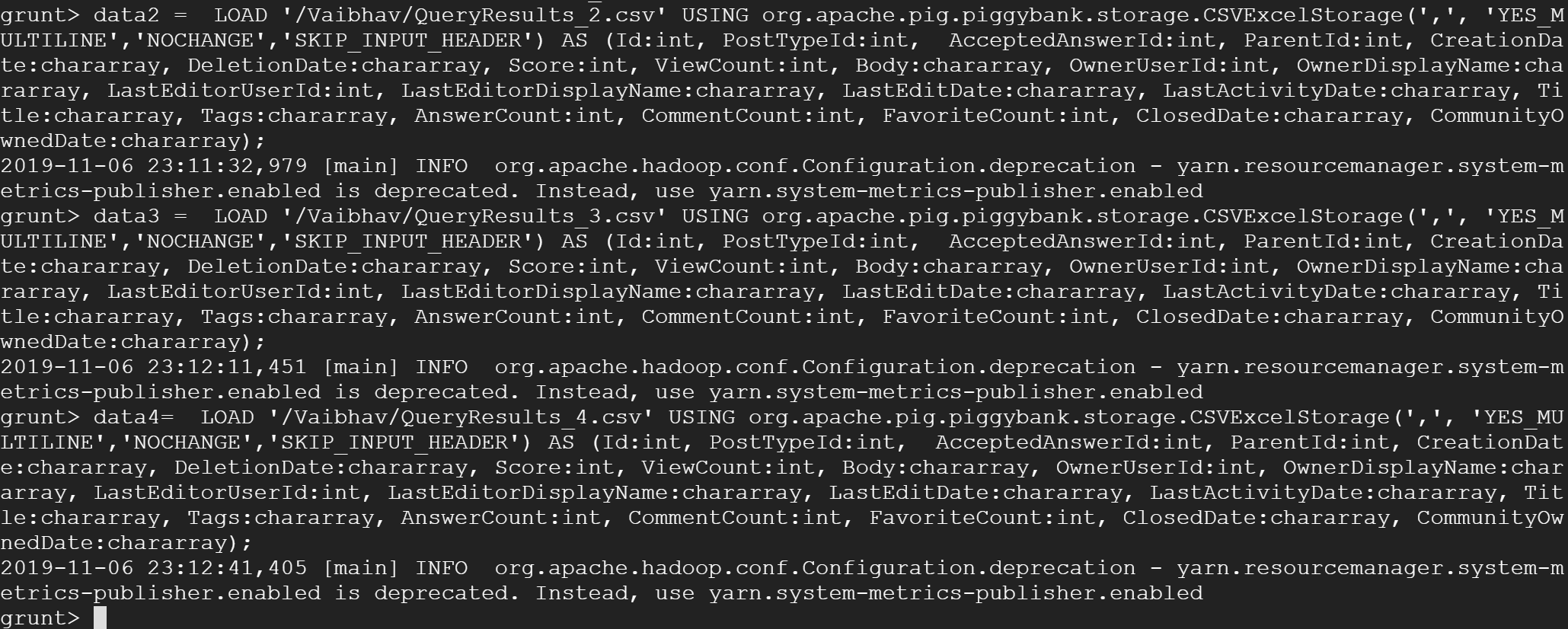
Step 2 : We make a directory in HDFS to upload the files

Hadoop fs -mkdir /Vaibhav

Step 3 : We ‘Put’ the files to the HDFS directory



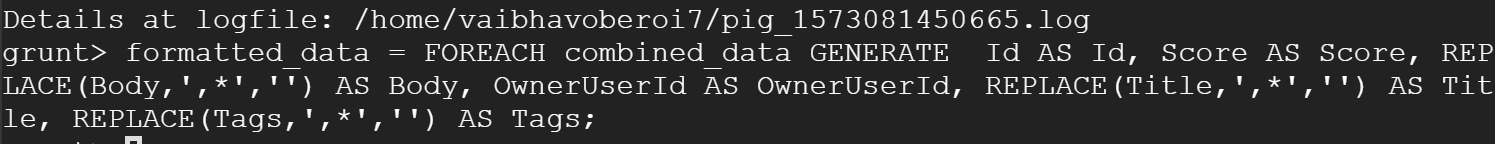
Step 4 : Now, we open ‘Pig’ and load the data files using CSVExcelStorage so as to use the added functionality of YES\_MULTILINE (to ignore \n) and skipping input header:



Step 5 : Now that we have the 4 data files, we combine them

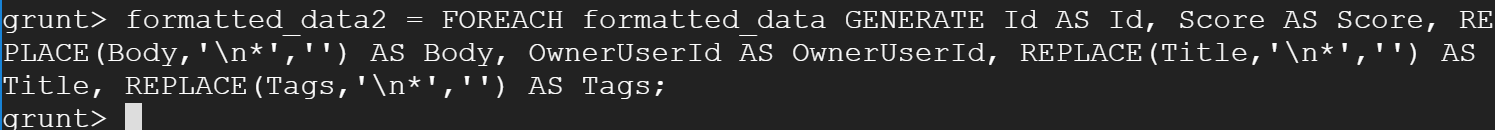


Step 6 : Then we shortlist columns and remove commas:

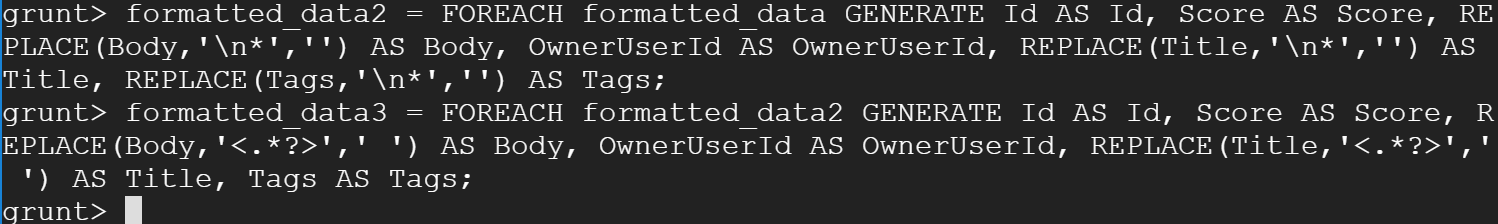


*Further formatting:*

Step 7 : Removing \n



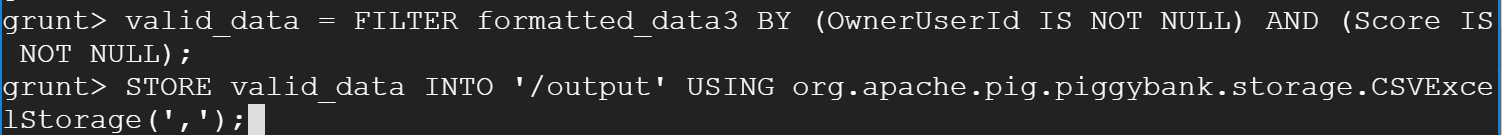
Step 8 : Remove HTML tags like <p>



Step 9:We have to Filter out null values and store the output into a new variable

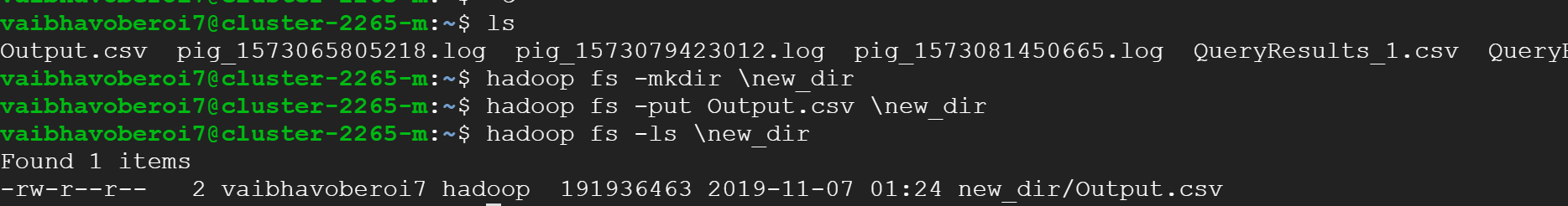
valid\_data = FILTER formatted\_data3 BY (OwnerUserId IS NOT NULL) AND (Score IS NOT NULL);

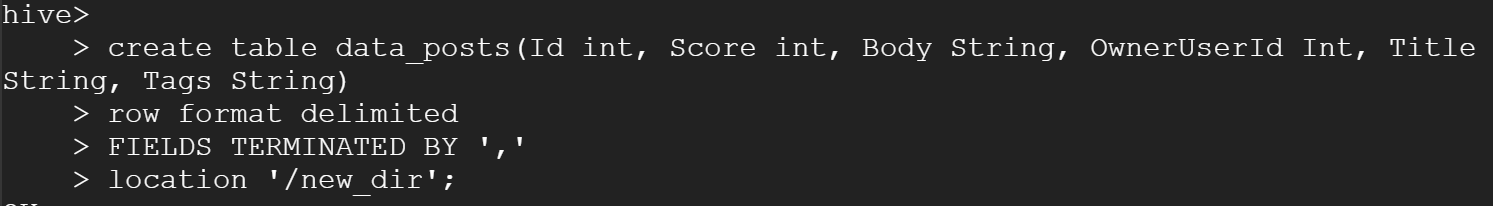
Step 10: Finally, we Store the valid data into HDFS directory



1. **HIVE**

Step 1 **:** We Create a new HDFS directory and putting our Pig Output file over there.



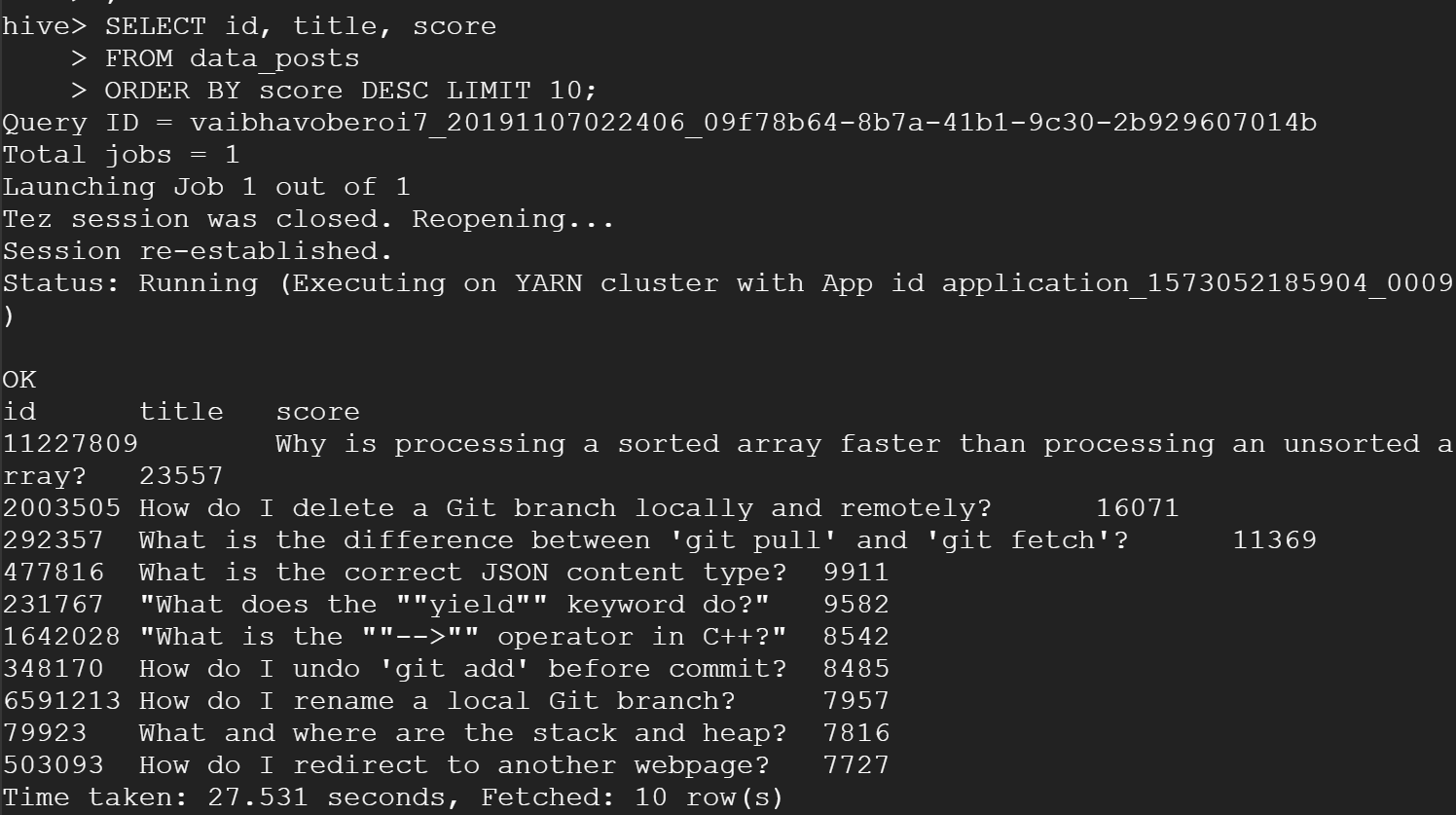
Step 2 : Creating table data\_posts to load the data from our CSV combined file.  


Step 3 : Setting Header to True , so as to print headers in the output .set hive.cli.print.header=true

*Querying with hive:*

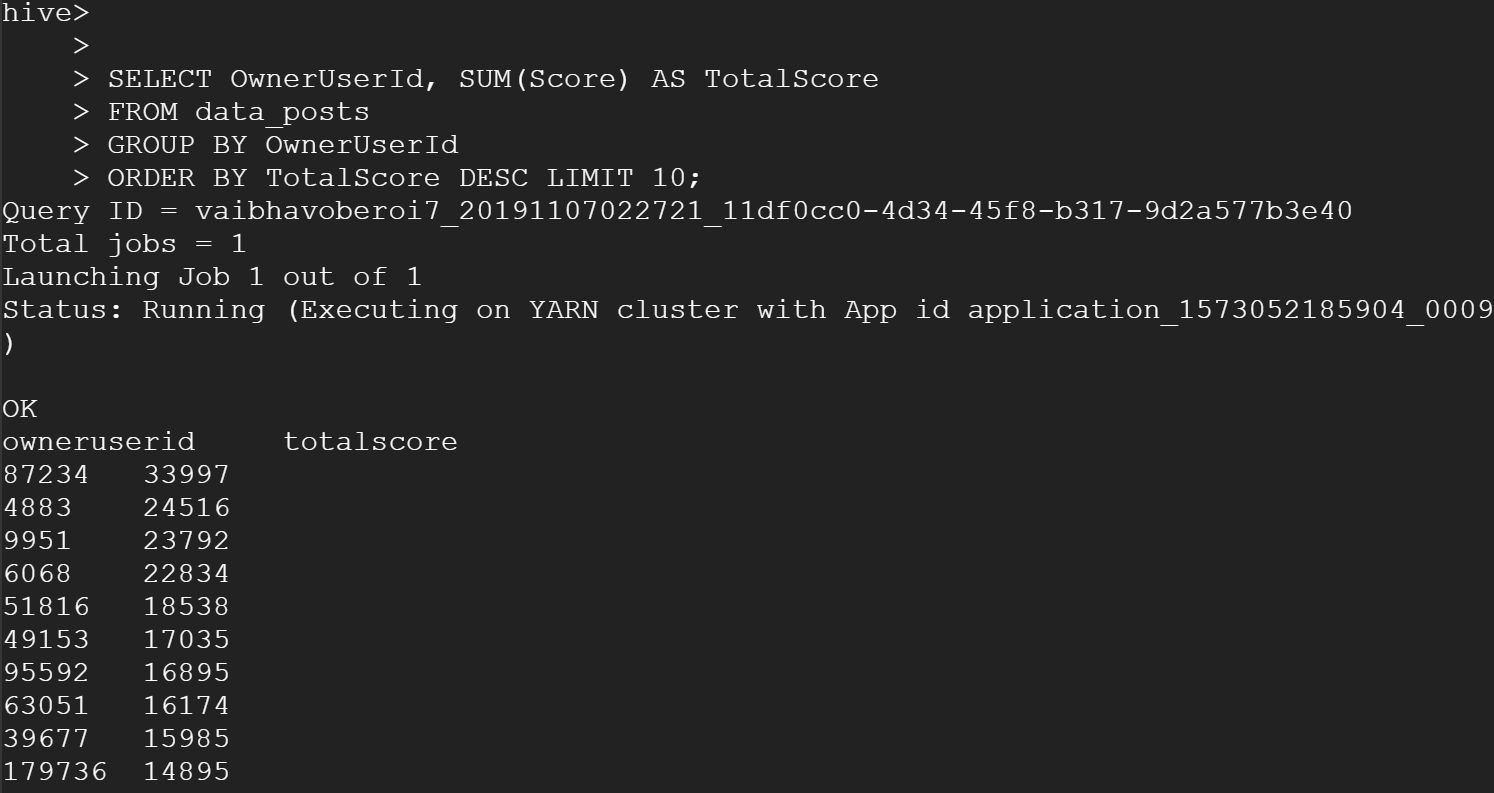
Step 4 : -- first query - 1. The top 10 posts by score

Selecting the columns id, title and score from our table, data\_posts and ordering the output in a descending manner.



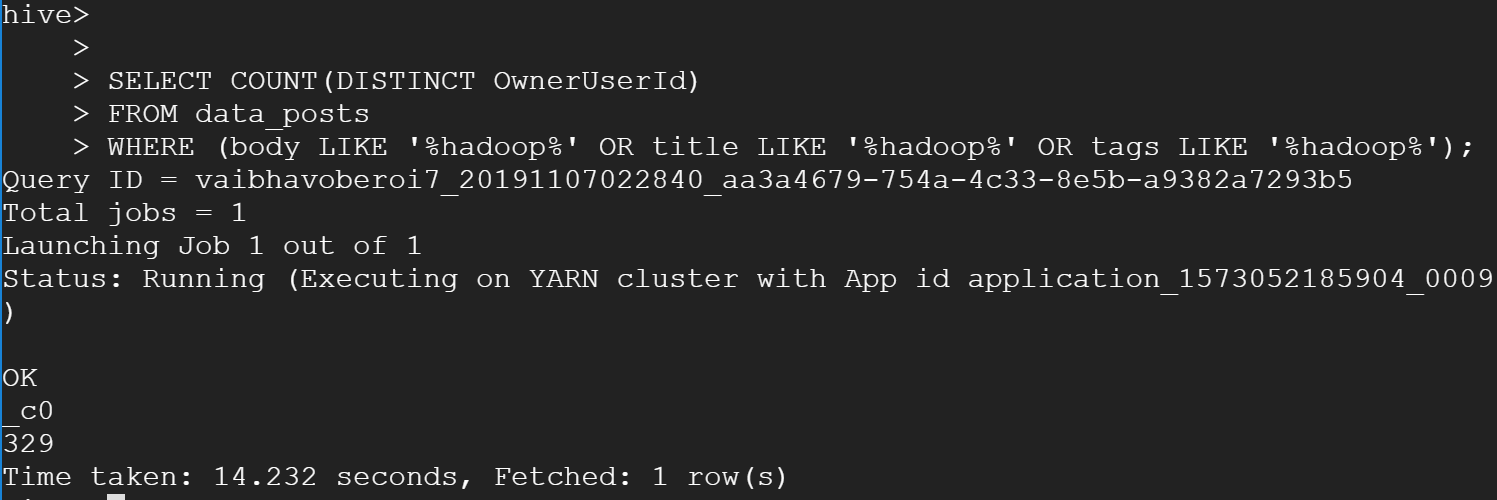
Step 5 : -- second query - 2. The top 10 users by post score

Applying group by functionality to OwnerUserId and aggregating the column Score

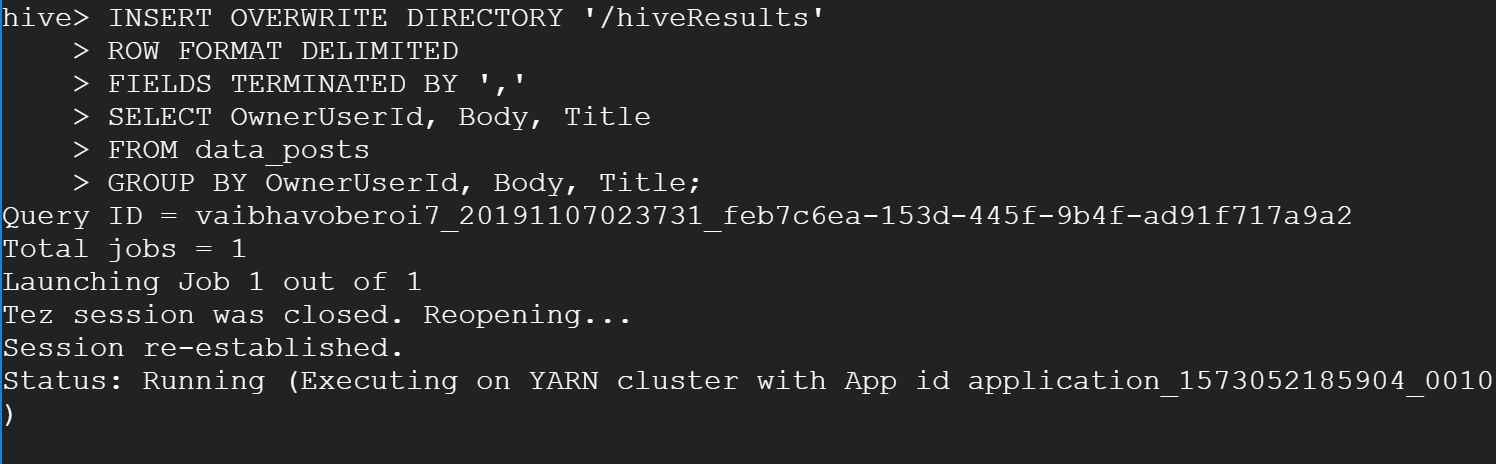


Step 6 : -- third query - 3. The number of distinct users, who used the word 'hadoop' in one of their posts

Printing out the count of distinct OwnerUserId from our table data\_posts in which ‘Hadoop’ has been mentioned.



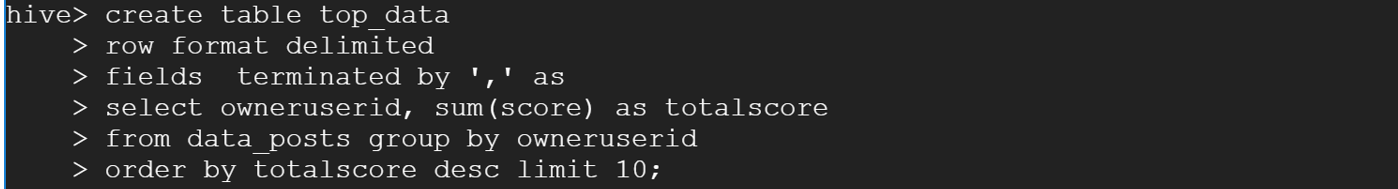
Step 7 : Creating a new directory, and saving our table in it by grouping by OwnerUserId, Body, Title.



1. **TF/IDF**

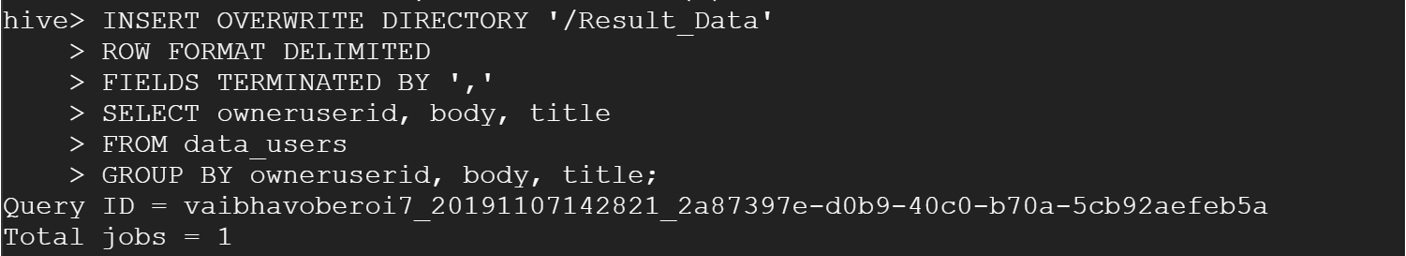
Step 1 : As we have to query TF-IDF for the top 10 users (Query 3.2), we will create a table to store its output

Creating table ‘top\_data’ to store the output.



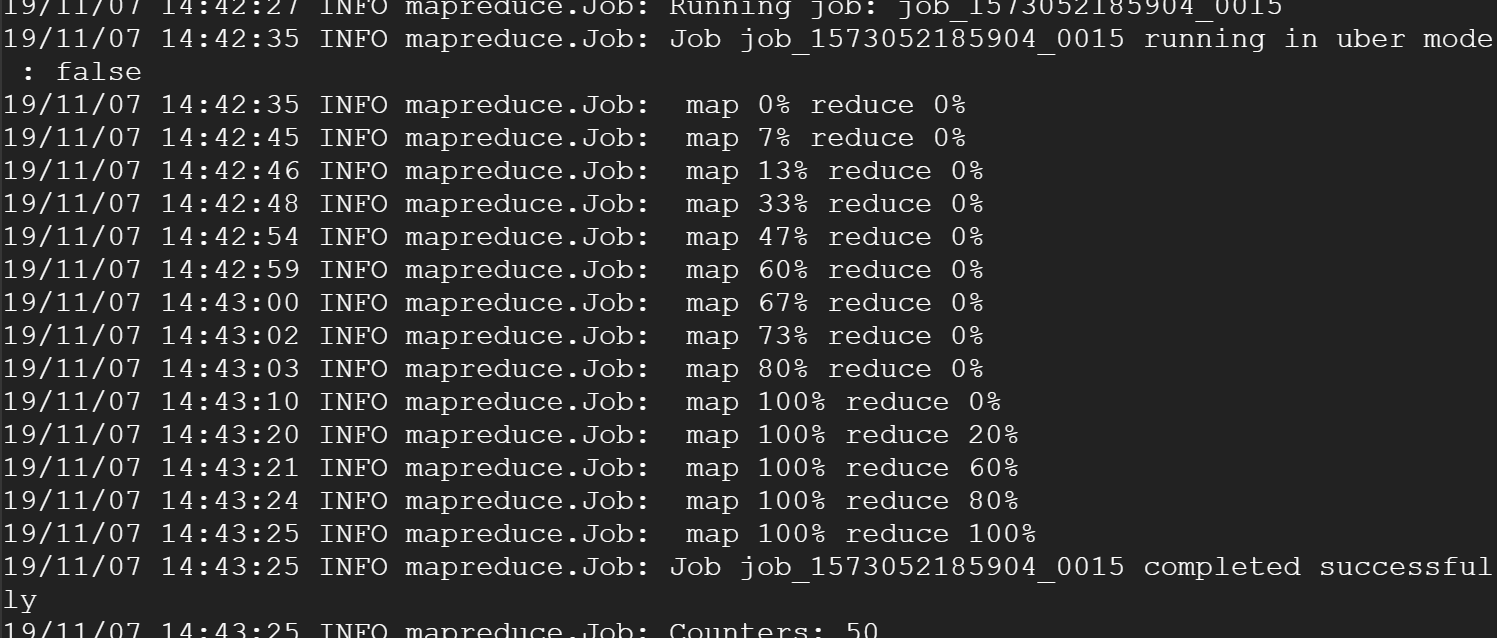
Step 2 : Now we extract the columns body, title and tags for these top 10 users and create a new table

Step 3 : Before proceeding to Map Reduce, we store this table’s data into a csv file, as this file would be used for input to MapReduce Phase 1.

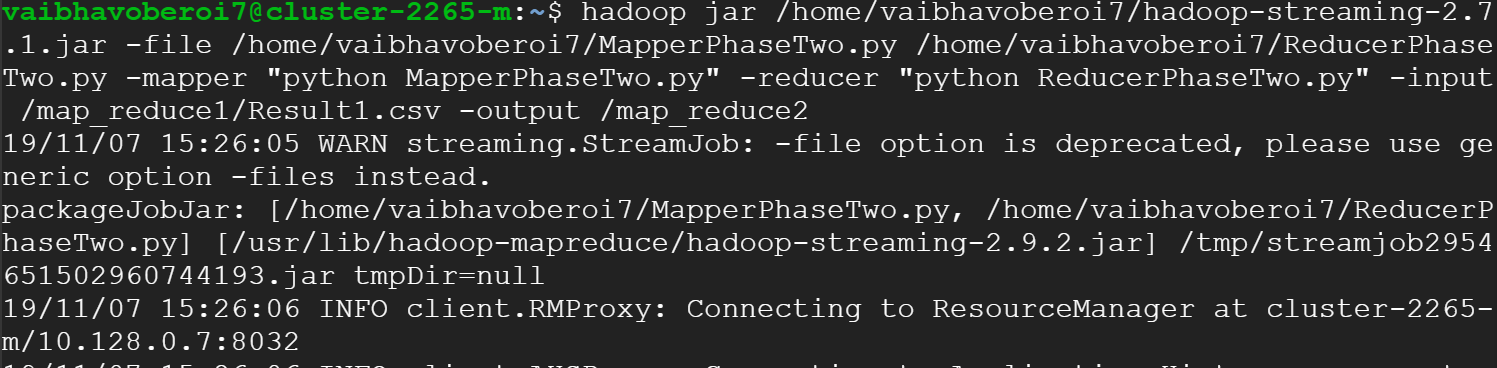


Step 4 : We make changes to add stop words to mapper phase one file and upload Mapper&Reducer for Phases 1,2 and 3; and uploading the jar file to our cluster.

Step 5 : We run the jar file for Mapper & Reducer Phase 1 and give the Hive Stored Results file as the input

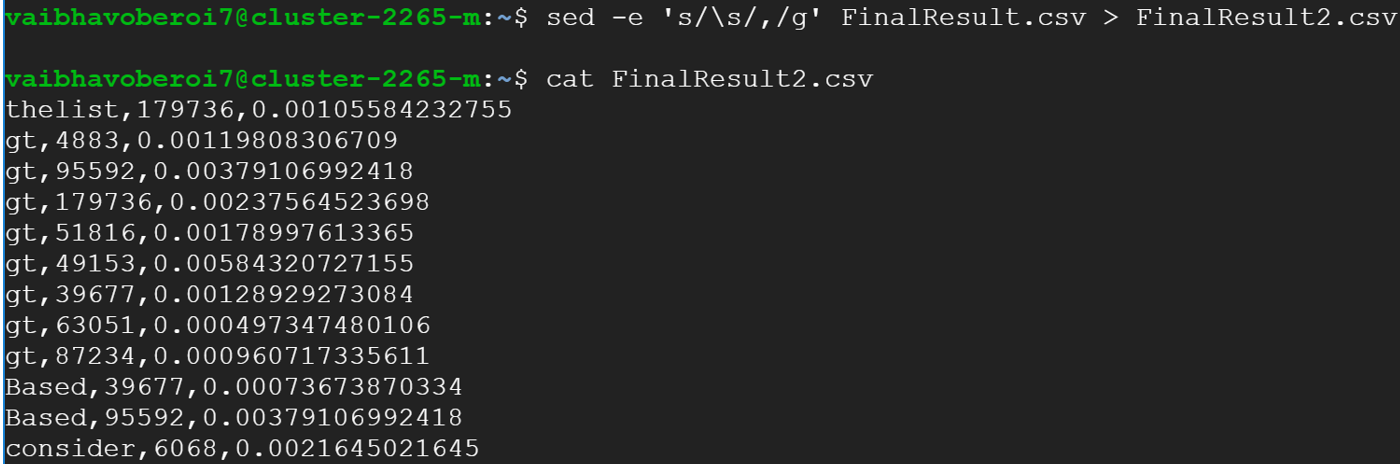


Step 6 : We combine the 4 result files from Step 5, upload them to HDFS and run Phase Two.

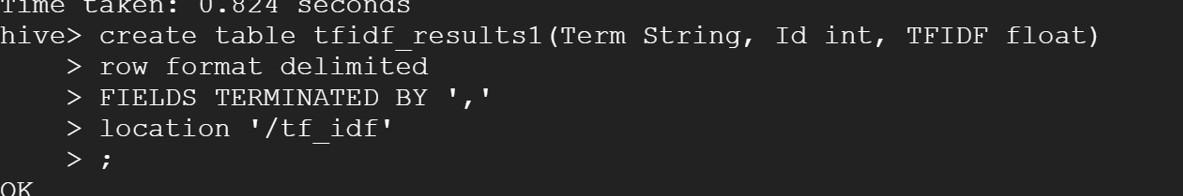


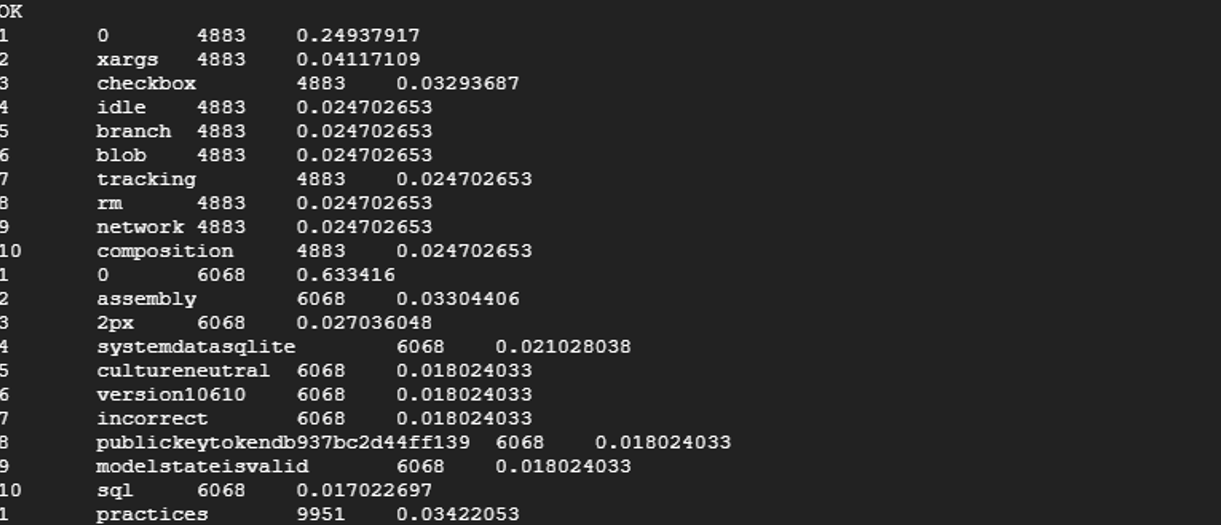
Step 7 : We combine the 4 result files from Step 6, upload them to HDFS and run Phase Three.

Step 8 : Printing final results after combining the result files from Step 7.



Step 9 : Now that we have got the results, we insert them into a table ‘tfidf\_results1’, in order to find the terms and their respective tfidf scores for the top 10 users.





Acknowledgements :

The mapreduce programs are referred to from <https://github.com/SatishUC15/TFIDF-HadoopMapReduce#tfidf-hadoop> , edited with minor changes (e.g. addition of stop words and the related if condition in MapperPhaseOne.py).