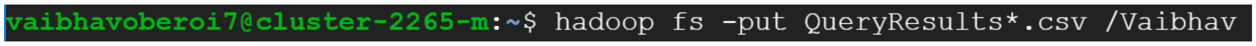
Big Data Analysis - Vaibhav Oberoi

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.
2. **ETL using Pig**
   Step 1 : We upload the 4 CSV files to the cluster

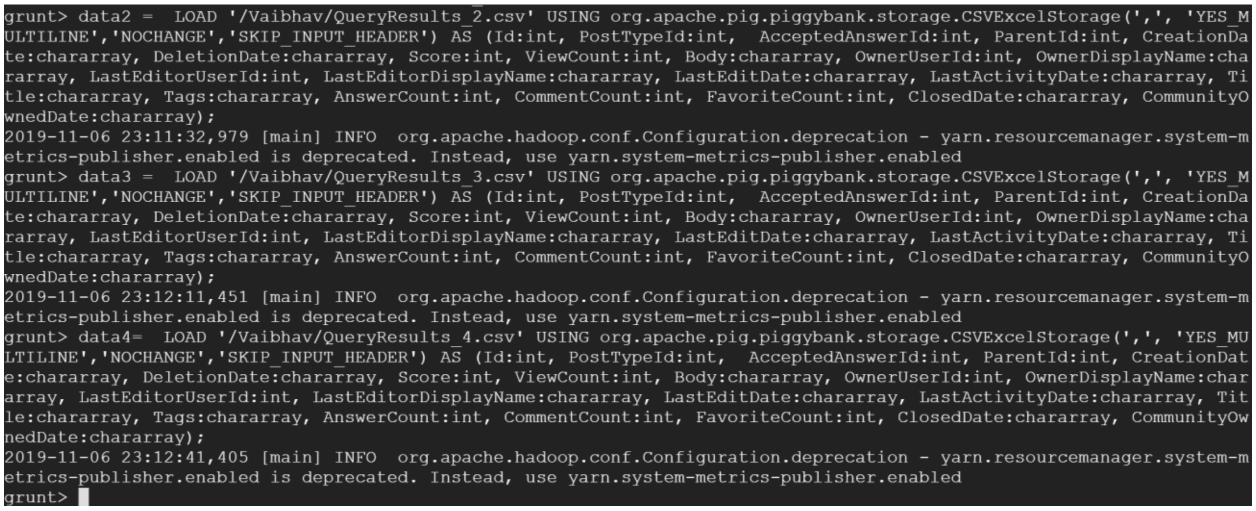
   Step 2 : We make a directory in HDFS to upload the files
   Hadoop fs -mkdir /Vaibhav

   Step 3 : We 'Put' the files to the HDFS directory



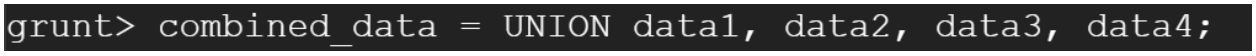
```
vaibhavoberoi7@cluster-2265-m:~$ hadoop fs -put QueryResults*.csv /Vaibhav
```

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:



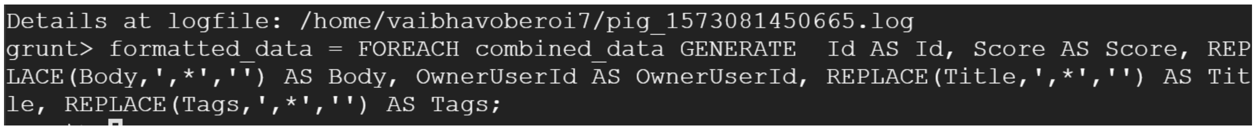
```
grunt> data2 =  LOAD '/Vaibhav/QueryResults_2.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'YES_M
ULTILINE','NOCHANGE','SKIP_INPUT_HEADER') AS (Id:int, PostTypeId:int,  AcceptedAnswerId:int, ParentId:int, CreationDa
te:chararray, DeletionDate:chararray, Score:int, ViewCount:int, Body:chararray, OwnerUserId:int, OwnerDisplayName:cha
rarray, LastEditorUserId:int, LastEditorDisplayName:chararray, LastEditDate:chararray, LastActivityDate:chararray, Ti
tle:chararray, Tags:chararray, AnswerCount:int, CommentCount:int, FavoriteCount:int, ClosedDate:chararray, CommunityO
wnedDate:chararray);
2019-11-06 23:11:32,979 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - yarn.resourcemanager.system-m
etrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publisher.enabled
grunt> data3 =  LOAD '/Vaibhav/QueryResults_3.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'YES_M
ULTILINE','NOCHANGE','SKIP_INPUT_HEADER') AS (Id:int, PostTypeId:int,  AcceptedAnswerId:int, ParentId:int, CreationDa
te:chararray, DeletionDate:chararray, Score:int, ViewCount:int, Body:chararray, OwnerUserId:int, OwnerDisplayName:cha
rarray, LastEditorUserId:int, LastEditorDisplayName:chararray, LastEditDate:chararray, LastActivityDate:chararray, Ti
tle:chararray, Tags:chararray, AnswerCount:int, CommentCount:int, FavoriteCount:int, ClosedDate:chararray, CommunityO
wnedDate:chararray);
2019-11-06 23:12:11,451 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - yarn.resourcemanager.system-m
etrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publisher.enabled
grunt> data4=  LOAD '/Vaibhav/QueryResults_4.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'YES_MU
LTILINE','NOCHANGE','SKIP_INPUT_HEADER') AS (Id:int, PostTypeId:int,  AcceptedAnswerId:int, ParentId:int, CreationDat
e:chararray, DeletionDate:chararray, Score:int, ViewCount:int, Body:chararray, OwnerUserId:int, OwnerDisplayName:char
array, LastEditorUserId:int, LastEditorDisplayName:chararray, LastEditDate:chararray, LastActivityDate:chararray, Tit
le:chararray, Tags:chararray, AnswerCount:int, CommentCount:int, FavoriteCount:int, ClosedDate:chararray, CommunityOw
nedDate:chararray);
2019-11-06 23:12:41,405 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - yarn.resourcemanager.system-m
etrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publisher.enabled
grunt>
```

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



```
grunt> combined_data = UNION data1, data2, data3, data4;
```

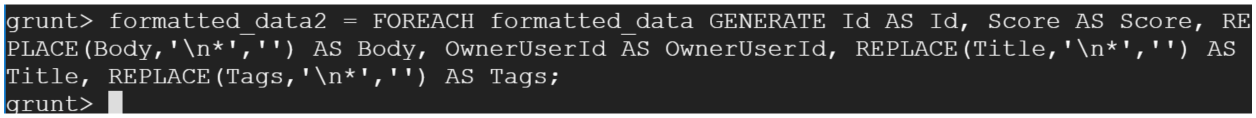
Step 6 : Then we shortlist columns and remove commas:



```
Details at logfile: /home/vaibhavoberoi7/pig_1573081450665.log
grunt> formatted_data = FOREACH combined_data GENERATE  Id AS Id, Score AS Score, REP
LACE(Body,',*','') AS Body, OwnerUserId AS OwnerUserId, REPLACE(Title,',*','') AS Tit
le, REPLACE(Tags,',*','') AS Tags;
```

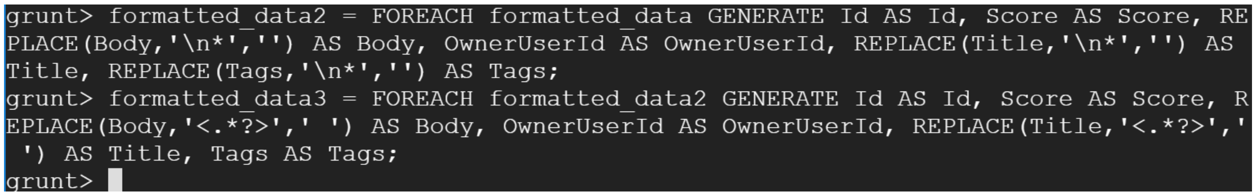
Further formatting:

Step 7 : Removing \n



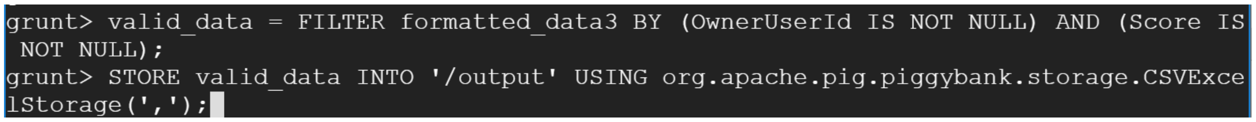
```
grunt> formatted_data2 = FOREACH formatted_data GENERATE Id AS Id, Score AS Score, RE
PLACE(Body,'\n*','') AS Body, OwnerUserId AS OwnerUserId, REPLACE(Title,'\n*','') AS
Title, REPLACE(Tags,'\n*','') AS Tags;
grunt>
```

Step 8 : Remove HTML tags like <p>



```
grunt> formatted_data2 = FOREACH formatted_data GENERATE Id AS Id, Score AS Score, RE
PLACE(Body,'\n*','') AS Body, OwnerUserId AS OwnerUserId, REPLACE(Title,'\n*','') AS
Title, REPLACE(Tags,'\n*','') AS Tags;
grunt> formatted_data3 = FOREACH formatted_data2 GENERATE Id AS Id, Score AS Score, R
EPLACE(Body,'<.*?>',' ') AS Body, OwnerUserId AS OwnerUserId, REPLACE(Title,'<.*?>','
 ') AS Title, Tags AS Tags;
grunt>
```

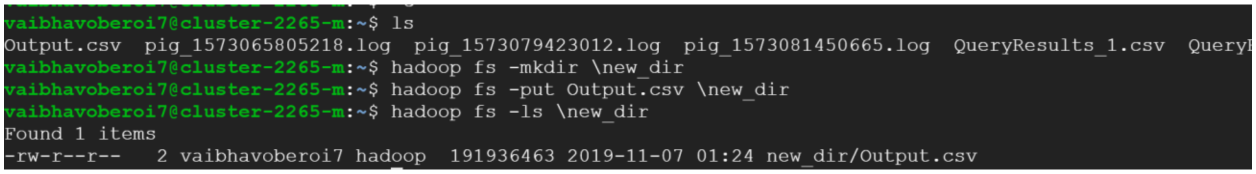
Step 9:We have to Filter out null values and store the output into a new variable
valid_data = FILTER formatted_data3 BY (OwnerUserId IS NOT NULL) AND (Score IS NOT NULL);
Step 10: Finally, we Store the valid data into HDFS directory



```
grunt> valid_data = FILTER formatted_data3 BY (OwnerUserId IS NOT NULL) AND (Score IS
 NOT NULL);
grunt> STORE valid_data INTO '/output' USING org.apache.pig.piggybank.storage.CSVExce
lStorage(',');
```

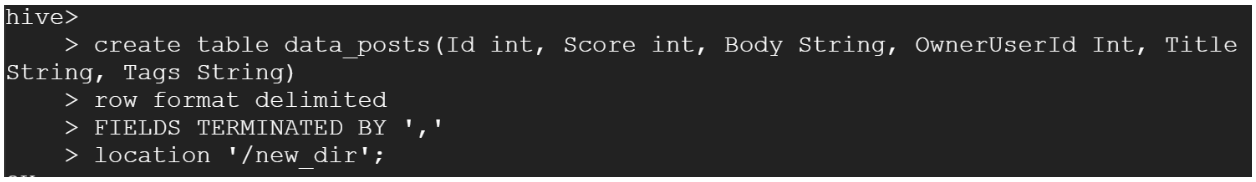
3. **HIVE**

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



```
vaibhavoberoi7@cluster-2265-m:~$ ls
Output.csv  pig_1573065805218.log  pig_1573079423012.log  pig_1573081450665.log  QueryResults_1.csv  QueryR
vaibhavoberoi7@cluster-2265-m:~$ hadoop fs -mkdir \new_dir
vaibhavoberoi7@cluster-2265-m:~$ hadoop fs -put Output.csv \new_dir
vaibhavoberoi7@cluster-2265-m:~$ hadoop fs -ls \new_dir
Found 1 items
-rw-r--r--   2 vaibhavoberoi7 hadoop  191936463 2019-11-07 01:24 new_dir/Output.csv
```

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



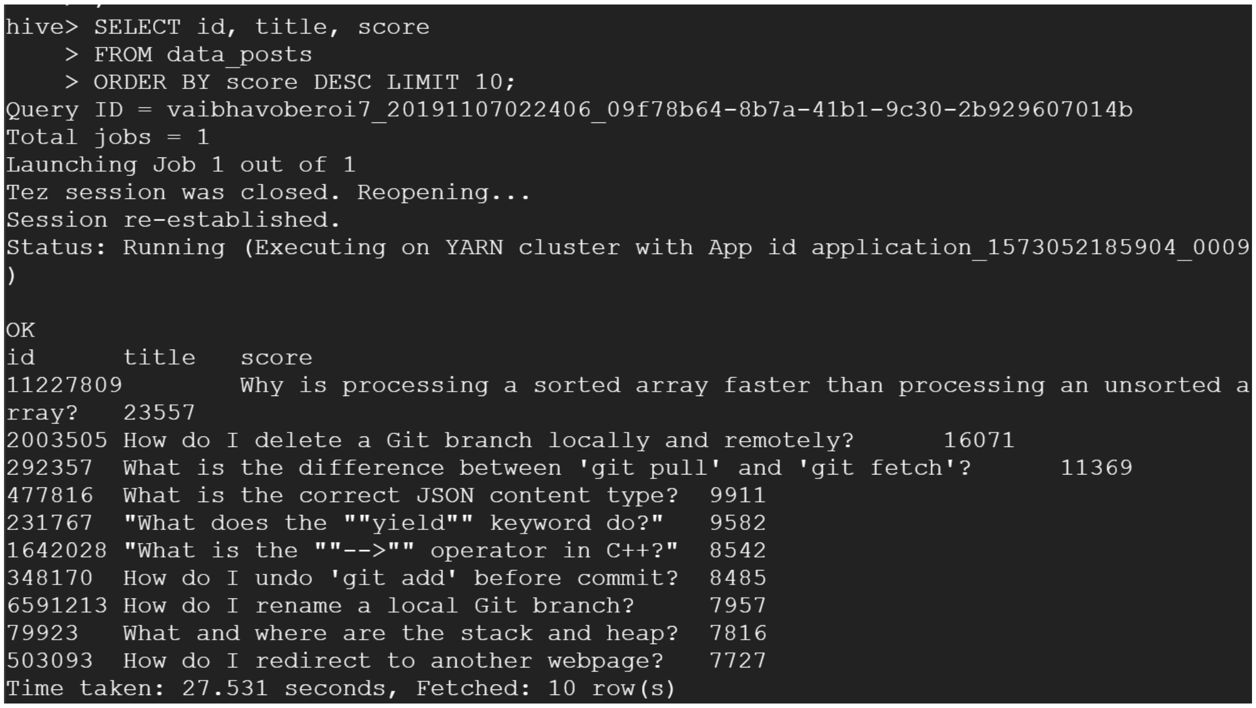
```
hive>
    > create table data_posts(Id int, Score int, Body String, OwnerUserId Int, Title
String, Tags String)
    > row format delimited
    > FIELDS TERMINATED BY ','
    > location '/new_dir';
```

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.

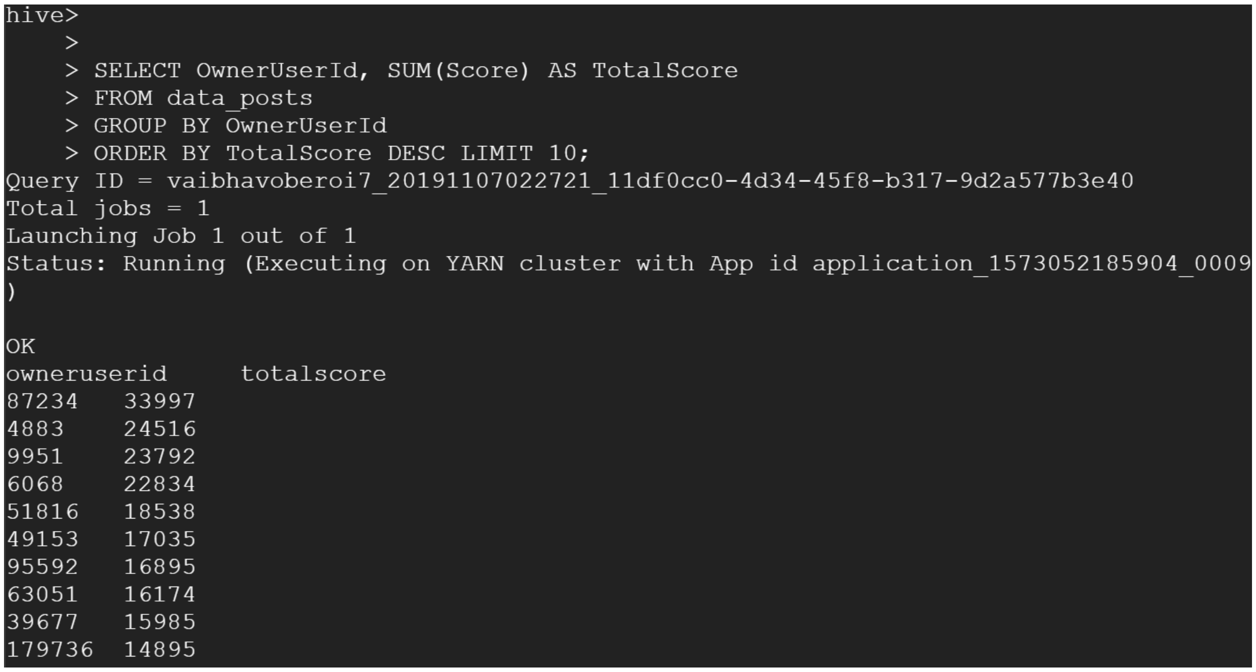


```
hive> SELECT id, title, score
    > FROM data_posts
    > ORDER BY score DESC LIMIT 10;
Query ID = vaibhavoberoi7_20191107022406_09f78b64-8b7a-41b1-9c30-2b929607014b
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1573052185904_0009
)

OK
id      title   score
11227809        Why is processing a sorted array faster than processing an unsorted a
rray?   23557
2003505 How do I delete a Git branch locally and remotely?      16071
292357  What is the difference between 'git pull' and 'git fetch'?      11369
477816  What is the correct JSON content type?  9911
231767  "What does the ""yield"" keyword do?"   9582
1642028 "What is the ""-->"" operator in C++?"  8542
348170  How do I undo 'git add' before commit?  8485
6591213 How do I rename a local Git branch?     7957
79923   What and where are the stack and heap?  7816
503093  How do I redirect to another webpage?   7727
Time taken: 27.531 seconds, Fetched: 10 row(s)
```

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

Applying group by functionality to OwnerUserId and aggregating the column Score

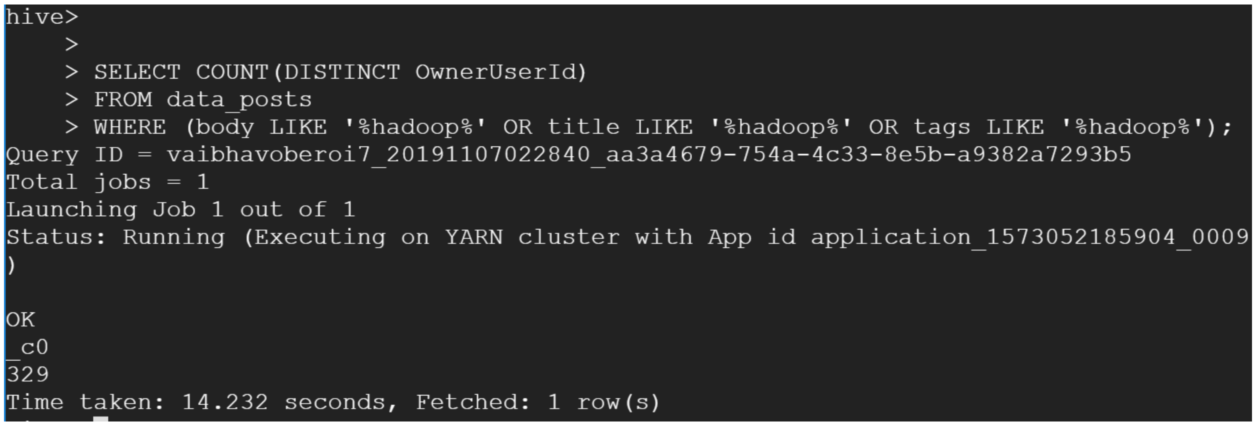


```
hive>
    >
    > SELECT OwnerUserId, SUM(Score) AS TotalScore
    > FROM data_posts
    > GROUP BY OwnerUserId
    > ORDER BY TotalScore DESC LIMIT 10;
Query ID = vaibhavoberoi7_20191107022721_11df0cc0-4d34-45f8-b317-9d2a577b3e40
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1573052185904_0009
)

OK
owneruserid     totalscore
87234   33997
4883    24516
9951    23792
6068    22834
51816   18538
49153   17035
95592   16895
63051   16174
39677   15985
179736  14895
```

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

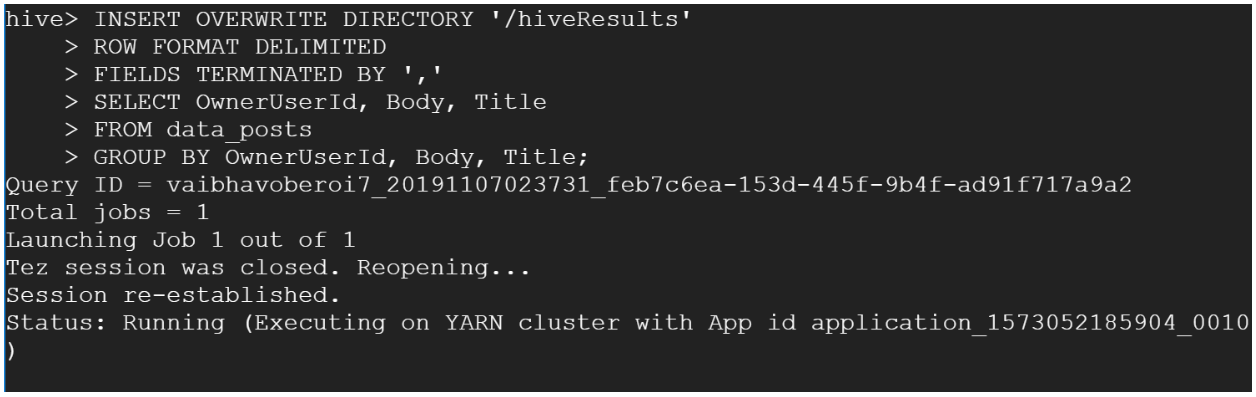
Printing out the count of distinct OwnerUserId from our table data_posts in which 'Hadoop' has been mentioned.



```
hive>
    >
    > SELECT COUNT(DISTINCT OwnerUserId)
    > FROM data_posts
    > WHERE (body LIKE '%hadoop%' OR title LIKE '%hadoop%' OR tags LIKE '%hadoop%');
Query ID = vaibhavoberoi7_20191107022840_aa3a4679-754a-4c33-8e5b-a9382a7293b5
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1573052185904_0009
)

OK
_c0
329
Time taken: 14.232 seconds, Fetched: 1 row(s)
```

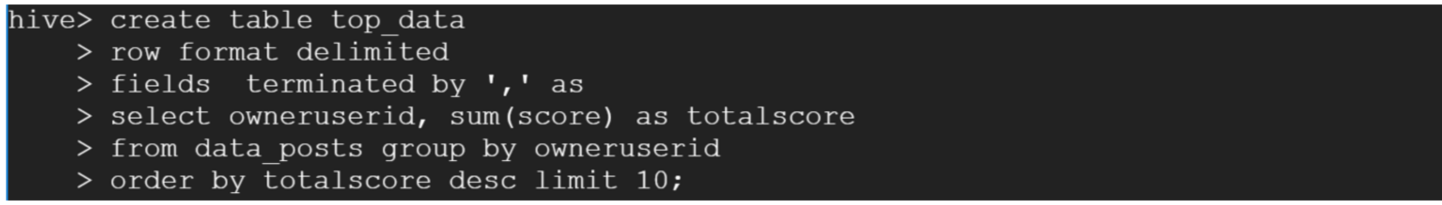
Step 7 : Creating a new directory, and saving our table in it by grouping by OwnerUserId, Body, Title.



```
hive> INSERT OVERWRITE DIRECTORY '/hiveResults'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > SELECT OwnerUserId, Body, Title
    > FROM data_posts
    > GROUP BY OwnerUserId, Body, Title;
Query ID = vaibhavoberoi7_20191107023731_feb7c6ea-153d-445f-9b4f-ad91f717a9a2
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1573052185904_0010
)
```

4. TF/IDF

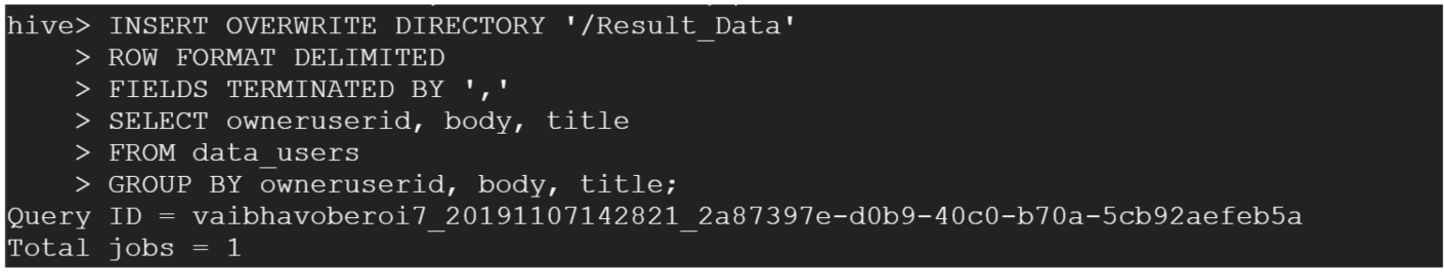
Step 1 : As we have to query TF-IDF for the top 10 users (Query 3.2), we will create a table to store its output Creating table 'top_data' to store the output.



```
hive> create table top_data
    > row format delimited
    > fields  terminated by ',' as
    > select owneruserid, sum(score) as totalscore
    > from data_posts group by owneruserid
    > order by totalscore desc limit 10;
```

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

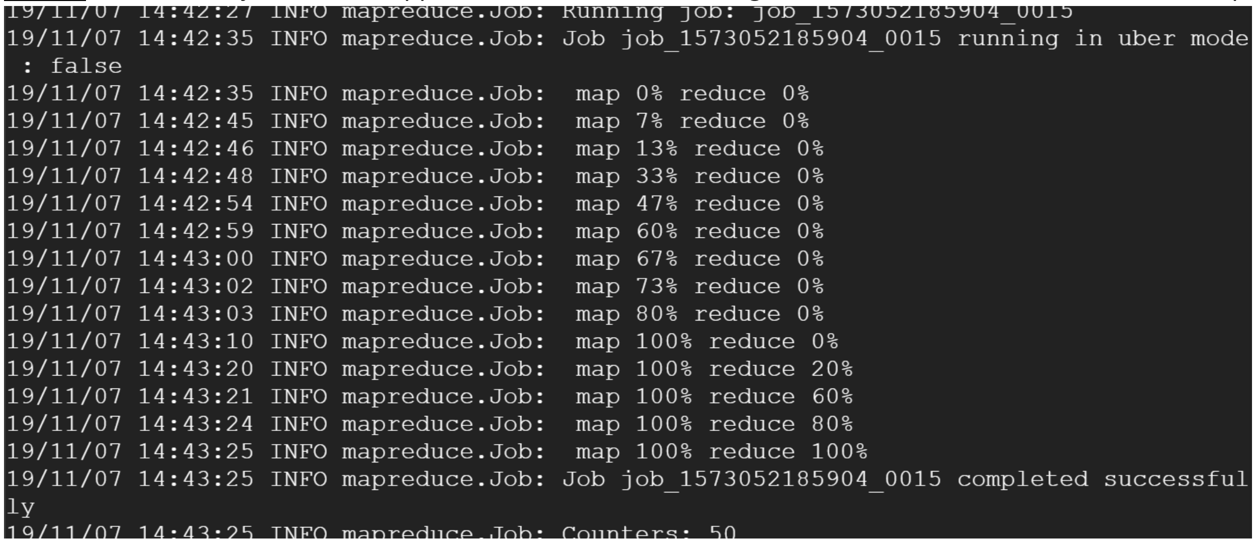
Step 3 : Before proceeding to Map Reduce, we store this table's data into a csv file, as this file would be used for input to MapReduce Phase 1.



```
hive> INSERT OVERWRITE DIRECTORY '/Result_Data'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > SELECT owneruserid, body, title
    > FROM data_users
    > GROUP BY owneruserid, body, title;
Query ID = vaibhavoberoi7_20191107142821_2a87397e-d0b9-40c0-b70a-5cb92aefeb5a
Total jobs = 1
```

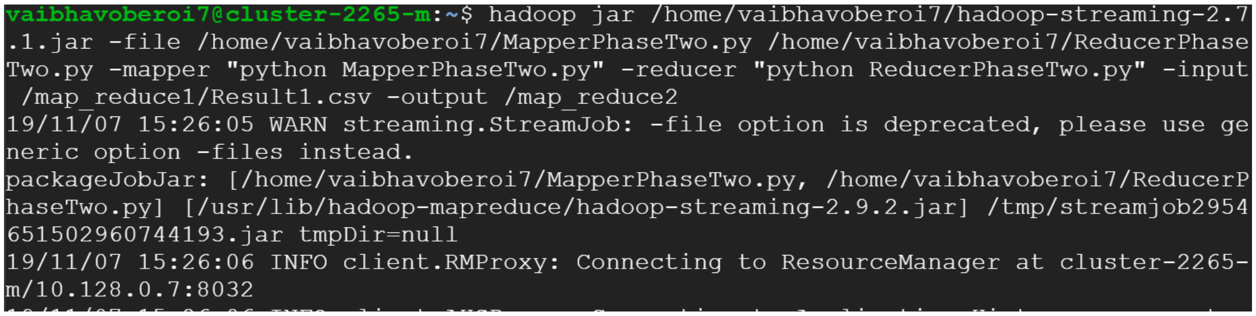
Step 4 : We make changes to add stop words to mapper phase one file and upload Mapper&Reducer for Phases 1,2 and 3; and uploading the jar file to our cluster.

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



```
19/11/07 14:42:27 INFO mapreduce.Job: Running job: job_1573052185904_0015
19/11/07 14:42:35 INFO mapreduce.Job: Job job_1573052185904_0015 running in uber mode
 : false
19/11/07 14:42:35 INFO mapreduce.Job:  map 0% reduce 0%
19/11/07 14:42:45 INFO mapreduce.Job:  map 7% reduce 0%
19/11/07 14:42:46 INFO mapreduce.Job:  map 13% reduce 0%
19/11/07 14:42:48 INFO mapreduce.Job:  map 33% reduce 0%
19/11/07 14:42:54 INFO mapreduce.Job:  map 47% reduce 0%
19/11/07 14:42:59 INFO mapreduce.Job:  map 60% reduce 0%
19/11/07 14:43:00 INFO mapreduce.Job:  map 67% reduce 0%
19/11/07 14:43:02 INFO mapreduce.Job:  map 73% reduce 0%
19/11/07 14:43:03 INFO mapreduce.Job:  map 80% reduce 0%
19/11/07 14:43:10 INFO mapreduce.Job:  map 100% reduce 0%
19/11/07 14:43:20 INFO mapreduce.Job:  map 100% reduce 20%
19/11/07 14:43:21 INFO mapreduce.Job:  map 100% reduce 60%
19/11/07 14:43:24 INFO mapreduce.Job:  map 100% reduce 80%
19/11/07 14:43:25 INFO mapreduce.Job:  map 100% reduce 100%
19/11/07 14:43:25 INFO mapreduce.Job: Job job_1573052185904_0015 completed successful
ly
19/11/07 14:43:25 INFO mapreduce.Job: Counters: 50
```

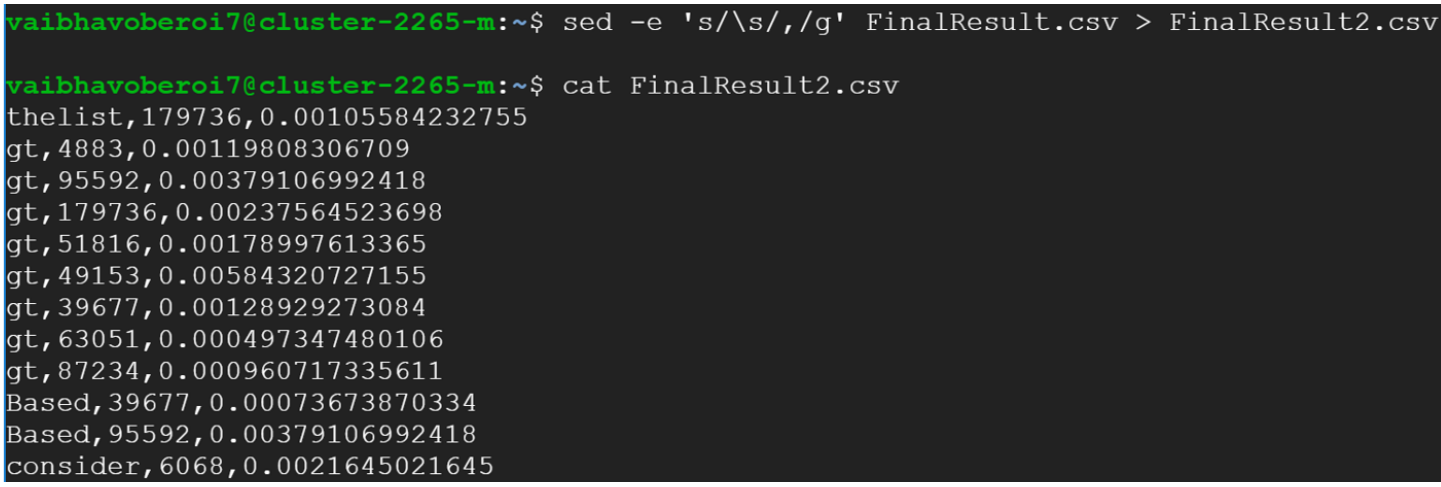
Step 6 : We combine the 4 result files from Step 5, upload them to HDFS and run Phase Two.



```
vaibhavoberoi7@cluster-2265-m:~$ hadoop jar /home/vaibhavoberoi7/hadoop-streaming-2.7
.1.jar -file /home/vaibhavoberoi7/MapperPhaseTwo.py /home/vaibhavoberoi7/ReducerPhase
Two.py -mapper "python MapperPhaseTwo.py" -reducer "python ReducerPhaseTwo.py" -input
 /map_reduce1/Result1.csv -output /map_reduce2
19/11/07 15:26:05 WARN streaming.StreamJob: -file option is deprecated, please use ge
neric option -files instead.
packageJobJar: [/home/vaibhavoberoi7/MapperPhaseTwo.py, /home/vaibhavoberoi7/ReducerP
haseTwo.py] [/usr/lib/hadoop-mapreduce/hadoop-streaming-2.9.2.jar] /tmp/streamjob2954
651502960744193.jar tmpDir=null
19/11/07 15:26:06 INFO client.RMProxy: Connecting to ResourceManager at cluster-2265-
m/10.128.0.7:8032
```

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

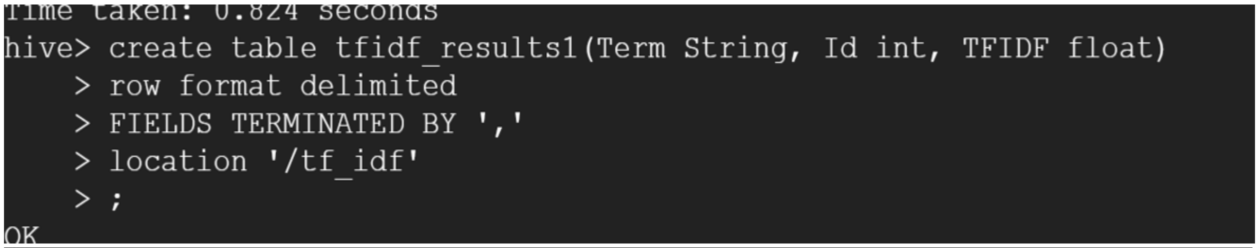
Step 8 : Printing final results after combining the result files from Step 7.



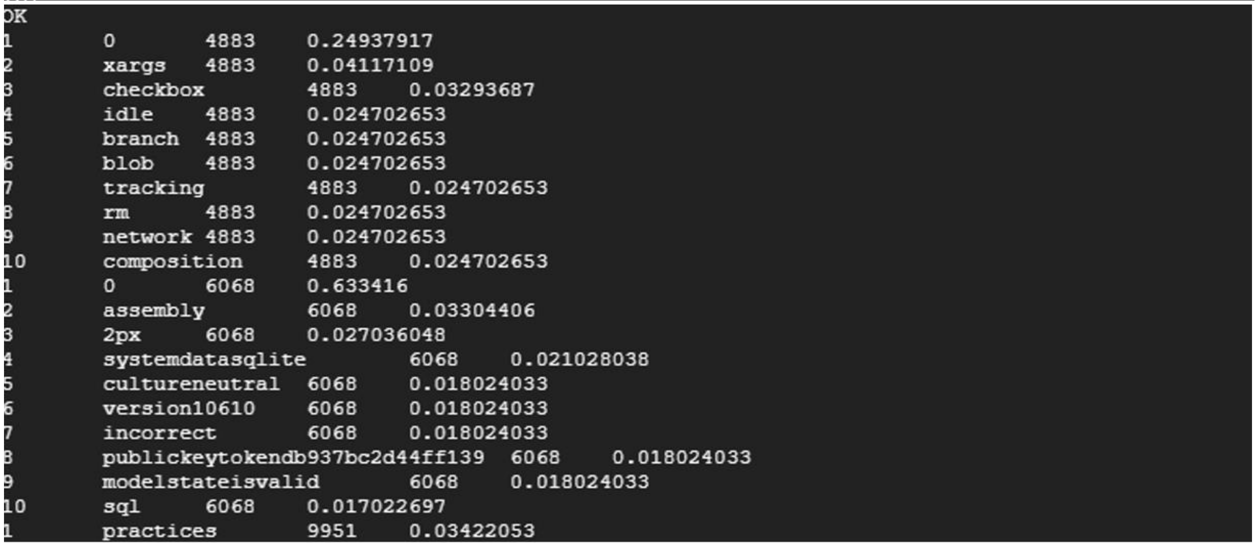
```
vaibhavoberoi7@cluster-2265-m:~$ sed -e 's/\s/,/g' FinalResult.csv > FinalResult2.csv

vaibhavoberoi7@cluster-2265-m:~$ cat FinalResult2.csv
thelist,179736,0.00105584232755
gt,4883,0.00119808306709
gt,95592,0.00379106992418
gt,179736,0.00237564523698
gt,51816,0.00178997613365
gt,49153,0.00584320727155
gt,39677,0.00128929273084
gt,63051,0.000497347480106
gt,87234,0.000960717335611
Based,39677,0.00073673870334
Based,95592,0.00379106992418
consider,6068,0.0021645021645
```

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.



```
Time taken: 0.824 seconds
hive> create table tfidf_results1(Term String, Id int, TFIDF float)
    > row format delimited
    > FIELDS TERMINATED BY ','
    > location '/tf_idf'
    > ;
OK
```



```
OK
1       0       4883    0.24937917
2       xargs   4883    0.04117109
3       checkbox        4883    0.03293687
4       idle    4883    0.024702653
5       branch  4883    0.024702653
6       blob    4883    0.024702653
7       tracking        4883    0.024702653
8       rm      4883    0.024702653
9       network 4883    0.024702653
10      composition     4883    0.024702653
1       0       6068    0.633416
2       assembly        6068    0.03304406
3       2px     6068    0.027036048
4       systemdatasqlite        6068    0.021028038
5       cultureneutral  6068    0.018024033
6       version10610    6068    0.018024033
7       incorrect       6068    0.018024033
8       publickeytokendb937bc2d44ff139  6068    0.018024033
9       modelstateisvalid       6068    0.018024033
10      sql     6068    0.017022697
1       practices       9951    0.03422053
```