2021/2022 CA675 Cloud Technologies

Assignment 1 - Data Analysis using Cloud Technologies

Romil Sakariya

Student ID: 21264095

GitLab project for this assignment: CA-675 Assignment 1

Email: romil.sakariya2@mail.dcu.ie

Step 1: Getting data from Stack Exchange (Data Acquisition/Collection)

After carefully analysing the tasks and going through database schema of StackExchage, I could deduce that the following columns are required to complete the assignment tasks:

1. id:int
2. Score:int
3. ViewCount:int
4. Body:chararray
5. OwnerUserId:int
6. OwnerDisplayName:chararray
7. Title:chararray
8. Tags:chararray

I ran the following query on the Data Explorer feature of the StackExchange system:

```
SELECT id, Score, ViewCount, Body, OwnerUserId, OwnerDisplayName, Title, Tags
FROM posts where posts.ViewCount>(upperlimit) AND
posts.ViewCount<=(lowerlimit)
```

The following table describes the values of lowerlimit and upperlimit I used in the above query and the different batch files I attained from them:

File Name	upperlimit value	lowerlimit value	Number of rows returned
QueryResults1.csv	1000000	No lowerlimit for first round	1506
QueryResults2.csv	125000	1000000	49955
QueryResults3.csv	75000	125000	48183
QueryResults4.csv	55000	75000	45298
QueryResults5.csv	43000	55000	46933
QueryResults6.csv	42000	43000	5180

All commands used to accomplish this step: link

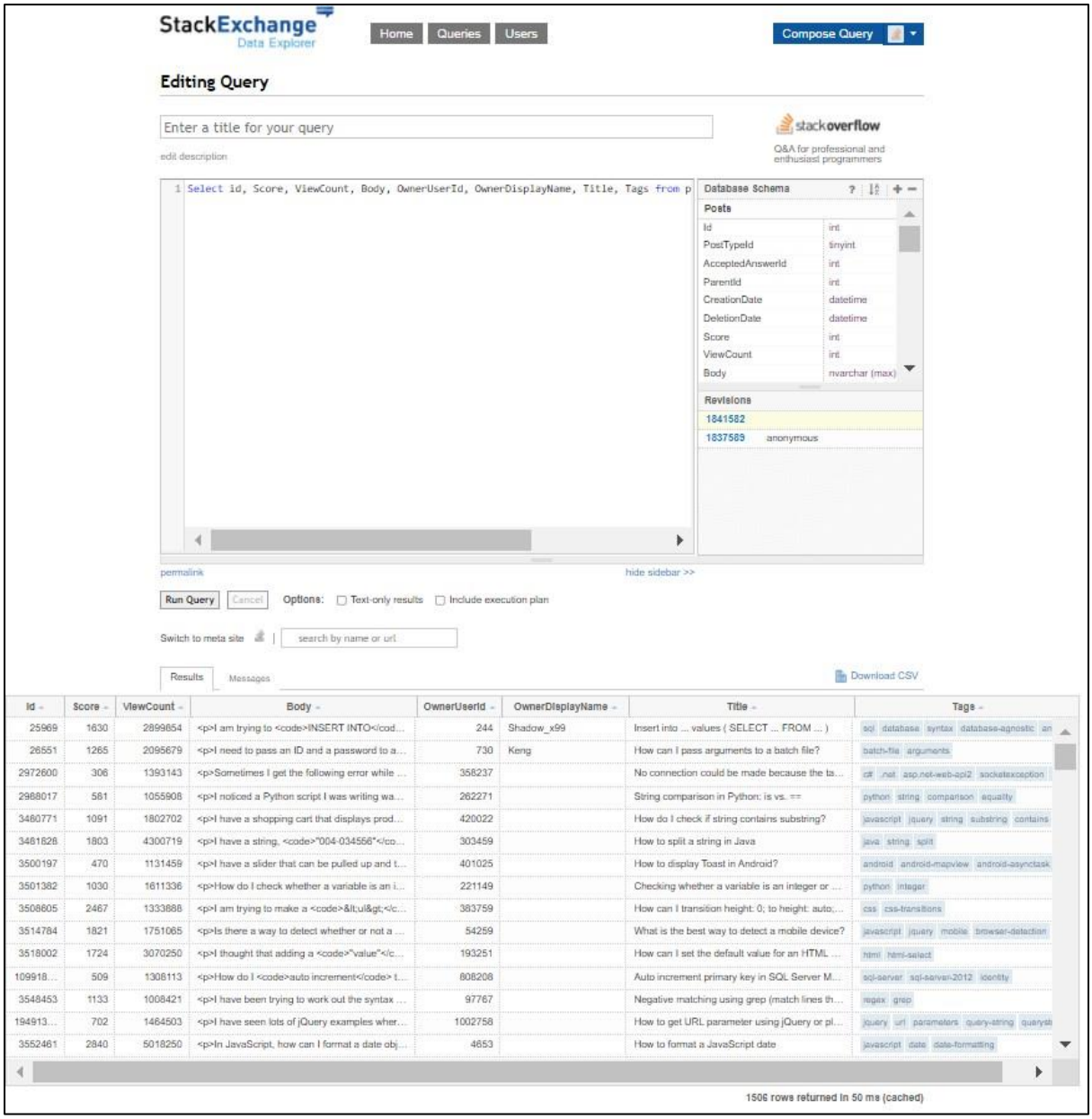

Fig 1: StackExchange Data Explorer with query to extract posts having a viewcount greater than 1000000

Step 2: Data Cleaning using Excel and Pig

Once I had acquired all the above batch files, I had to clean them and combine them into one CSV file comprising of approximately 200000 posts sorted by their ViewCount.

To clean the data, I initially used Microsoft Excel. Since the batch files were reasonably small, I could open them in Excel comfortably and view the schema, analyse the type of data and perform minor cleansing operations such as getting rid of any empty or duplicate entries and special characters from the content in the body column.

Now in order to combine all these batch files and cleanse the data even further, I used Apache Pig.

I used Google Cloud Platform's Dataproc service to create a Hadoop cluster of VMs with one masternode and two datanodes. I uploaded the batch files to the master VM and then transferred them to the HDFS system of the GCP cluster.

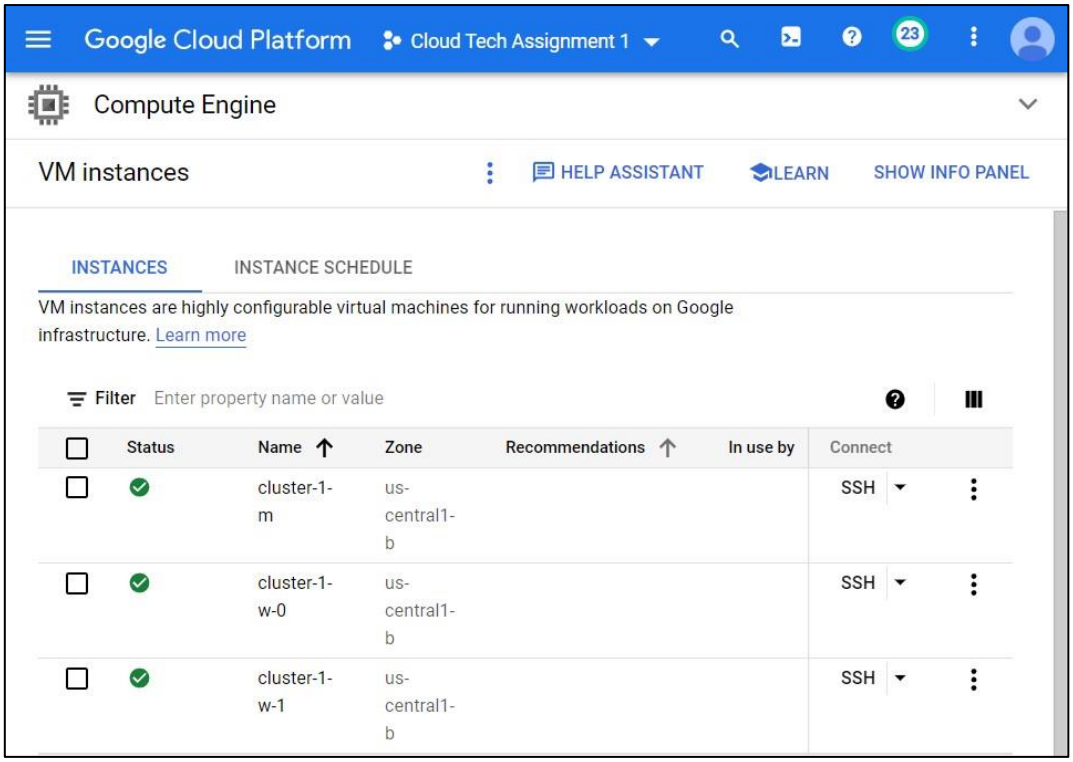

Fig 2: GCP Dataproc Cluster with all the nodes listed

```
romil_sakariya2@cluster-1-m:~$ ls
test.csv
romil_sakariya2@cluster-1-m:~$ ls
QueryResults1.csv  QueryResults3.csv  QueryResults5.csv  test.csv
QueryResults2.csv  QueryResults4.csv  QueryResults6.csv
romil_sakariya2@cluster-1-m:~$ hdfs dfs -ls /
Found 3 items
drwxr-xr-x   - romil_sakariya2 hadoop          0 2021-10-25 18:01 /assignment
drwxrwxrwt   - hdfs            hadoop          0 2021-10-25 15:21 /tmp
drwxrwxrwt   - hdfs            hadoop          0 2021-10-25 15:20 /user
romil_sakariya2@cluster-1-m:~$ hdfs dfs -ls /assignment
Found 1 items
-rw-r--r--   2 romil_sakariya2 hadoop         43 2021-10-25 18:01 /assignment/test.csv
romil_sakariya2@cluster-1-m:~$ hdfs dfs -put QueryResults1.csv /assignment
romil_sakariya2@cluster-1-m:~$ hdfs dfs -put QueryResults2.csv /assignment
romil_sakariya2@cluster-1-m:~$ hdfs dfs -put QueryResults3.csv /assignment
romil_sakariya2@cluster-1-m:~$ hdfs dfs -put QueryResults4.csv /assignment
romil_sakariya2@cluster-1-m:~$ hdfs dfs -put QueryResults5.csv /assignment
romil_sakariya2@cluster-1-m:~$ hdfs dfs -put QueryResults6.csv /assignment
romil_sakariya2@cluster-1-m:~$ hdfs dfs -ls /assignment
Found 7 items
-rw-r--r--   2 romil_sakariya2 hadoop    1084982 2021-10-25 23:37 /assignment/QueryResults1.csv
-rw-r--r--   2 romil_sakariya2 hadoop   48788602 2021-10-25 23:37 /assignment/QueryResults2.csv
-rw-r--r--   2 romil_sakariya2 hadoop   53840533 2021-10-25 23:37 /assignment/QueryResults3.csv
-rw-r--r--   2 romil_sakariya2 hadoop   53177616 2021-10-25 23:37 /assignment/QueryResults4.csv
-rw-r--r--   2 romil_sakariya2 hadoop   58429121 2021-10-25 23:37 /assignment/QueryResults5.csv
-rw-r--r--   2 romil_sakariya2 hadoop    6476415 2021-10-25 23:37 /assignment/QueryResults6.csv
-rw-r--r--   2 romil_sakariya2 hadoop         43 2021-10-25 18:01 /assignment/test.csv
romil_sakariya2@cluster-1-m:~$
```

Fig 3: Putting batch files in HDFS from local cloud storage

Once all batch files were in the HDFS system of my cluster, I used the following pig commands to load, transform and extract data:

```
A = LOAD 'QueryResults1.csv' USING PigStorage(',') AS (Id:int, Score:int,
ViewCount:int, Body:chararray, OwnerUserId:int, OwnerDisplayName:chararray,
Title:chararray, Tags:chararray);

a = FILTER A BY Id>1;
```

```
……

F = LOAD 'QueryResults6.csv' USING PigStorage(',') AS (Id:int, Score:int,
ViewCount:int, Body:chararray, OwnerUserId:int, OwnerDisplayName:chararray,
Title:chararray, Tags:chararray);

f = FILTER F BY Id>1;

G = UNION a,b,c,d,e,f;

H = DISTINCT G;

STORE H INTO ‘/assignment/FinalData USING PigStorage(‘,’);

hdfs dfs -get /assignment/FinalData/part-r-00000 /home/romil_sakariya2
```

I used Pig to clean my data owing to its ability to handle unstructured data and quick computation time. At this stage using Hive would have been counterproductive since the data was still in a very raw unstructured format.

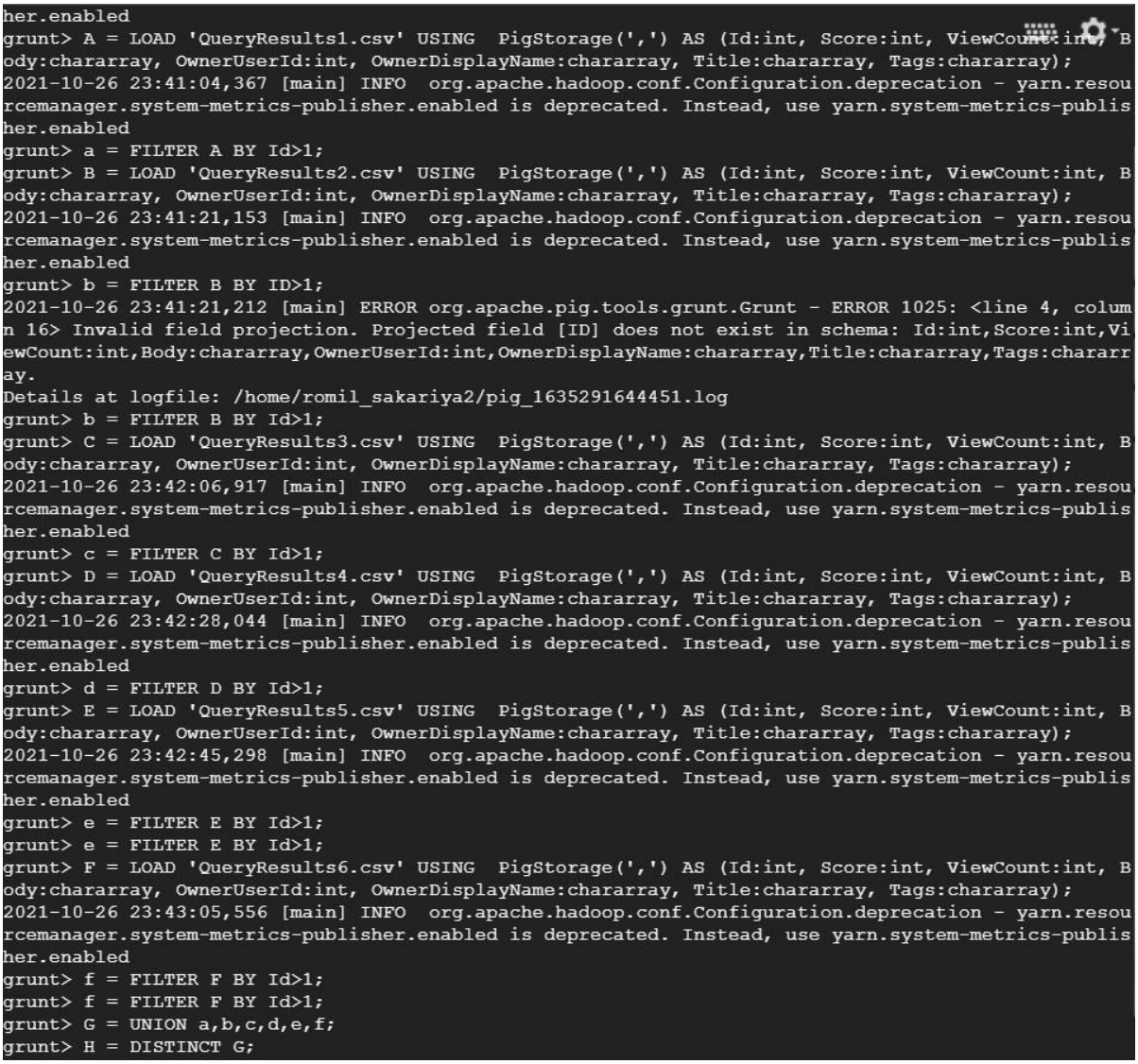

Fig 4: Loading and cleaning data in Pig

All commands used to accomplish this task: [link]

Step 3: Fetching the top 10 posts by score using Pig

After obtaining the complete data set from Pig, I used the same relation (H) to fetch the top 10 posts by score using the following Pig commands:

```
sorted_by_score = ORDER H BY Score DESC;

task1 = LIMIT sorted_by_score 10;
```

I used Pig to complete this task because all the required cleaned data was loaded in relation H. Using just a sort command I was able to sort the entire data set in descending order of score. I could have also done the same using Hive but that would require explicitly creating data tables and loading data again.

All commands used to accomplish this task: link

Output File: Task2.2.1

Step 4: Fetching top 10 users by total post score

Now that I have the complete **structured** data set, I loaded it into a Hive table:

```
CREATE EXTERNAL TABLE finaldata (id INT, score INT, viewcount INT, body
STRING, owneruserid INT, ownerdisplayname STRING, title STRING, tags STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

LOAD DATA LOCAL INPATH '/home/romil_sakariya2/finaloutput.csv' INTO TABLE
finaldata;
```

Now using Hive table functions like SUM() and partitions I could easily calculate cumulative sum of post scores of a particular user in the following manner:

```
CREATE TABLE users1 AS SELECT id, score, owneruserid, ownerdisplayname FROM
finaldata WHERE owneruserid IS NOT NULL;

CREATE TABLE userspostsum AS SELECT id, score, owneruserid, ownerdisplayname,
SUM(score) OVER (PARTITION BY owneruserid ORDER BY owneruserid ROWS BETWEEN
UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) FROM users1;
```

I ranked the rows further in a partition of ownerID and added the first row of each partition into a new table (called tempunique1). Now simply sorting and limiting this table gave me the required result.

```
CREATE TABLE temp AS SELECT id, score, owneruserid, ownerdisplayname,
sum_window_0, row_number() OVER (PARTITION BY owneruserid) AS rownumber FROM
userspostsum;

CREATE TABLE tempunique1 AS SELECT id, score, owneruserid, ownerdisplayname,
sum_window_0 asscoresum FROM temp WHERE rownumber=1;

CREATE TABLE task_2_2 AS SELECT id, score, owneruserid, ownerdisplayname,
asscoresum totalscoresum FROM temp ORDER BY totalscoresum DESC LIMIT 10;
```

All commands used to accomplish this task: link

Output File: Task2.2.2

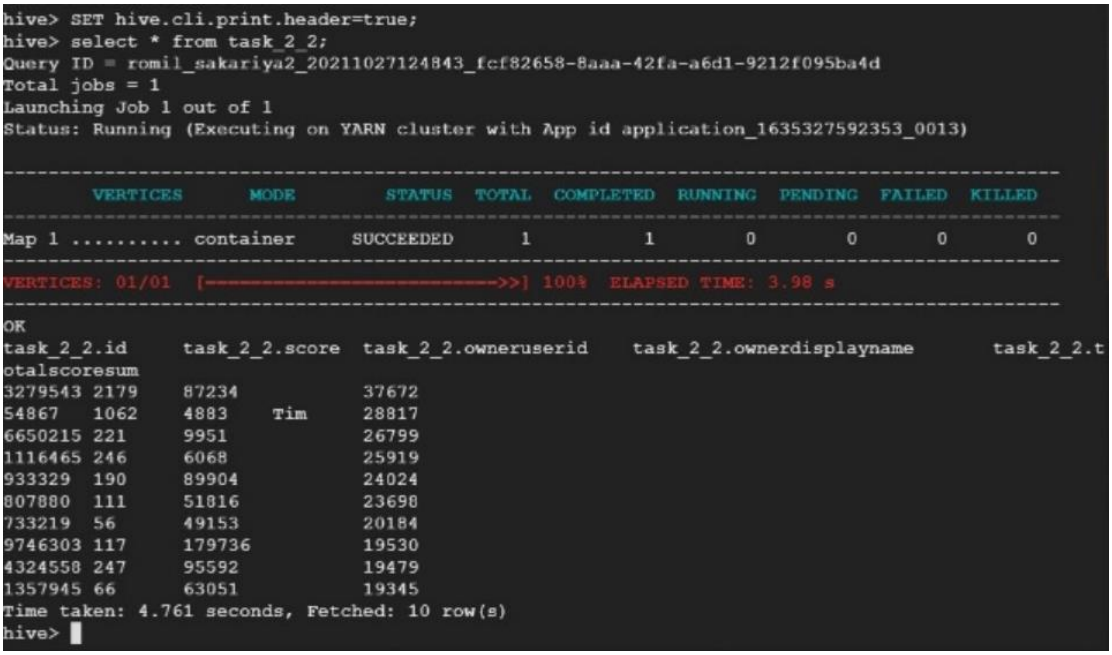

Fig 5: Result of top 10 users by total post score with their total post scores

Step 5: Fetching the total number of distinct users who used the word “cloud” in one of their posts

To make sure all the instances of the word “cloud” are covered, I first converted all the text in the body, title and tags column to lower case using the following commands:

```
CREATE TABLE lowercased_finaldata (id INT, score INT, viewcount INT, body STRING, owneruserid INT, ownerdisplayname STRING, title STRING, tags STRING);

INSERT INTO lowercased_finaldata (id, score, viewcount, body, owneruserid, ownerdisplayname, title, tags) SELECT id, score, viewcount, LOWER(body), owneruserid, ownerdisplayname, LOWER (title), LOWER(tags) from finaldata;
```

Once the all the posts’ title, body and tags were converted into lower case, I began looking for all possible occurrences of the word “CLOUD” in them. I loaded all these occurrences in a separate table. Counting the distinct number of owneruserids in this table will give the total number of distinct users who used the word “cloud” in one of their posts.

```
CREATE TABLE cloud AS SELECT * FROM lowercased_finaldata WHERE body||title||tags RLIKE 'cloud';

INSERT INTO cloud SELECT * FROM lowercased_finaldata WHERE body||title||tags RLIKE '-cloud-';

……

INSERT INTO cloud SELECT * FROM lowercased_finaldata WHERE body||title||tags RLIKE 'cloud-';

SELECT COUNT(DISTINCT owneruserid) FROM cloud;
```

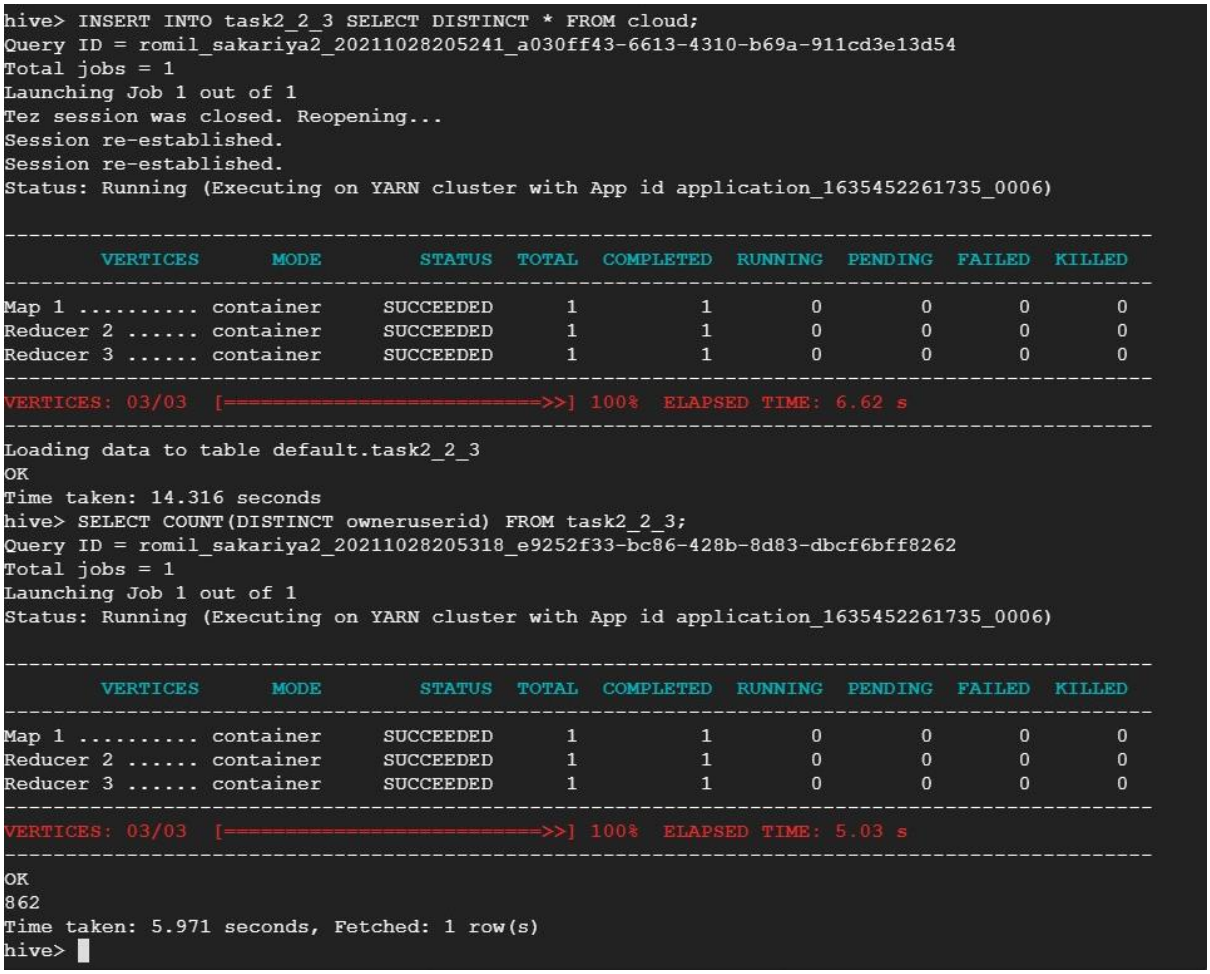

Fig 6: Output for task 2.2.3

The total number of distinct users who used the word “cloud” in one of their posts is 862.

All commands used to accomplish this task: link

Output File: Task2.2.3