CA-675 || CLOUD TECHNOLOGY || ASSIGNMENT 1 - DATA ANALYSIS

NAME	SHUBHAM VERMA
STUDENT ID	20210906
EMAIL	shubham.verma3@mail.dcu.ie
PROGRAMME OF STUDY	MSC. IN COMPUTING (DATA ANALYTICS)
GITHUB LINK	https://github.com/Shverma3696/CA675_assignment1_DataAnalysis

[Task 1] Data Extraction:

We are required to acquire the top 200,000 posts by ViewCount from the Stack Exchange site. Problem is that we can only download 50.000 records at a time.

[Task 2] Loading Data with PIG

Extract, transform and load the data as applicable

[Task 3] Querying with HIVE

The top 10 posts by score? The top 10 users by post score? The number of distinct users, who used the word 'Hadoop' in one of their posts?

[Task 4] Calculate the per-user TF-IDF with HIVE

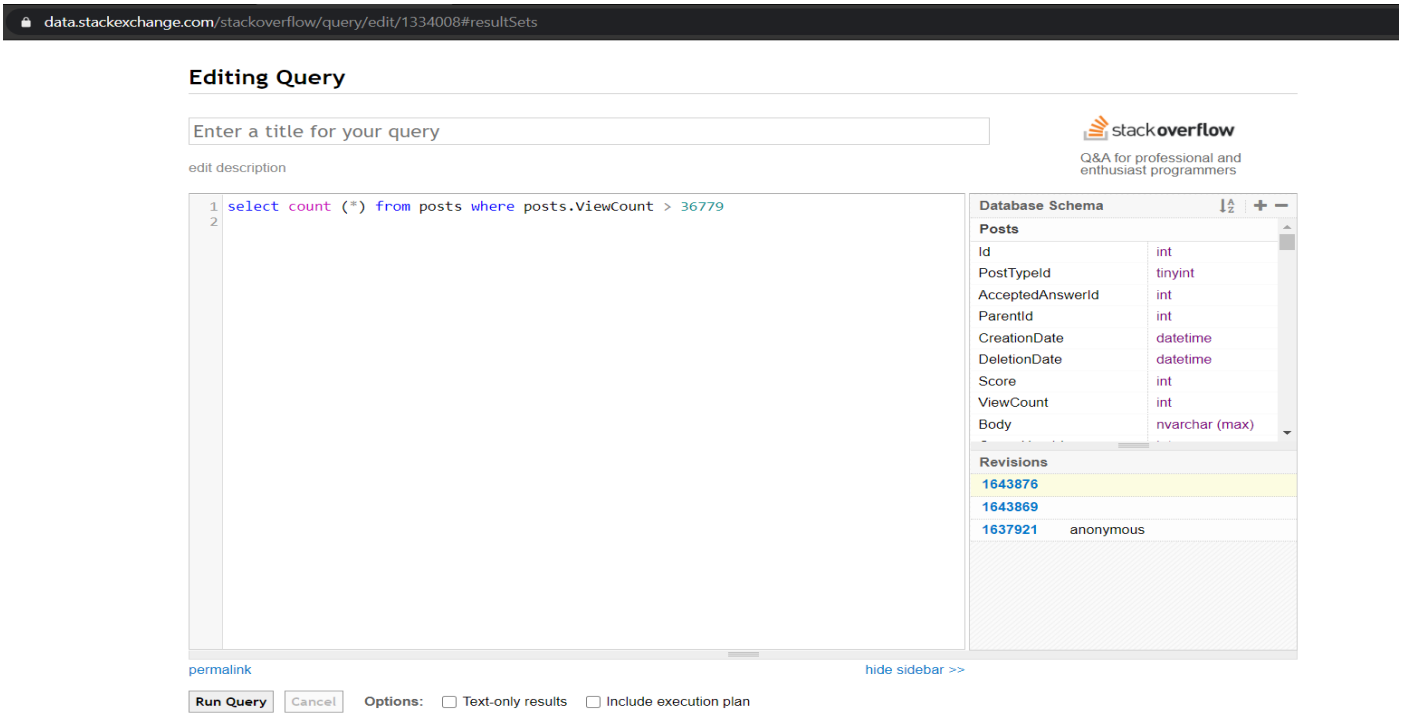
Find Top 10 terms used for each of the top 10 users by post score

“Stack Exchange is a network of question and answer websites on diverse topics in many different fields, each site covering a specific topic, where questions, answers, and users are subject to a reputation award process. The sites are modelled after Stack Overflow, a forum for computer programming questions that was the original site in this network.”

Stack Exchange Data Explorer (SEDE) https://data.stackexchange.com/stackoverflow/query/new

TASK 1 – Extracting the top 200,000 post by view count.

- Data Acquisition from Stack Exchange Data Explorer (SEDE).
- We can only download 50,000 records at a time, so we must run at least 4 queries in total to obtain 200,000 posts.
- To begin with, I had to find out the lower bound of the range of values by “ViewCount” field.
- After several hit and trial queries, I found out that the values in “ViewCount” greater than 36779 gave me 200,002 records.

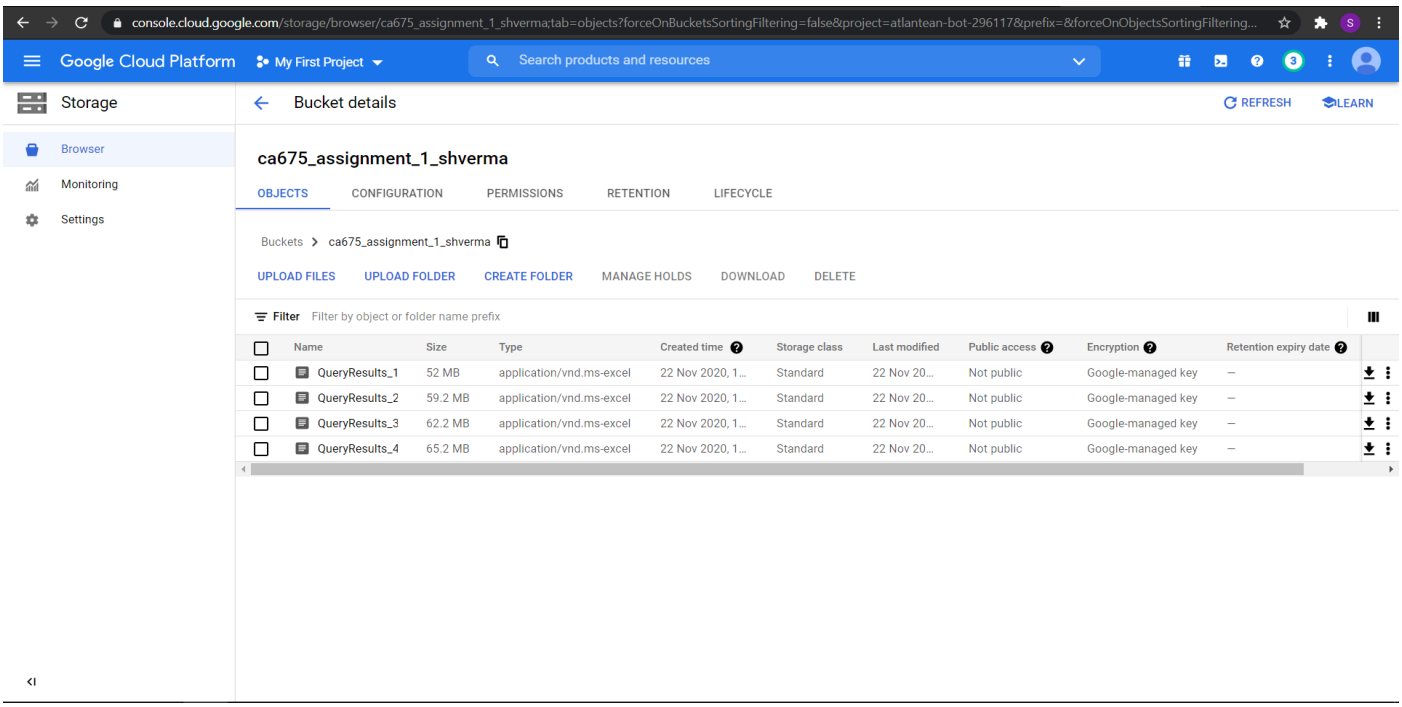


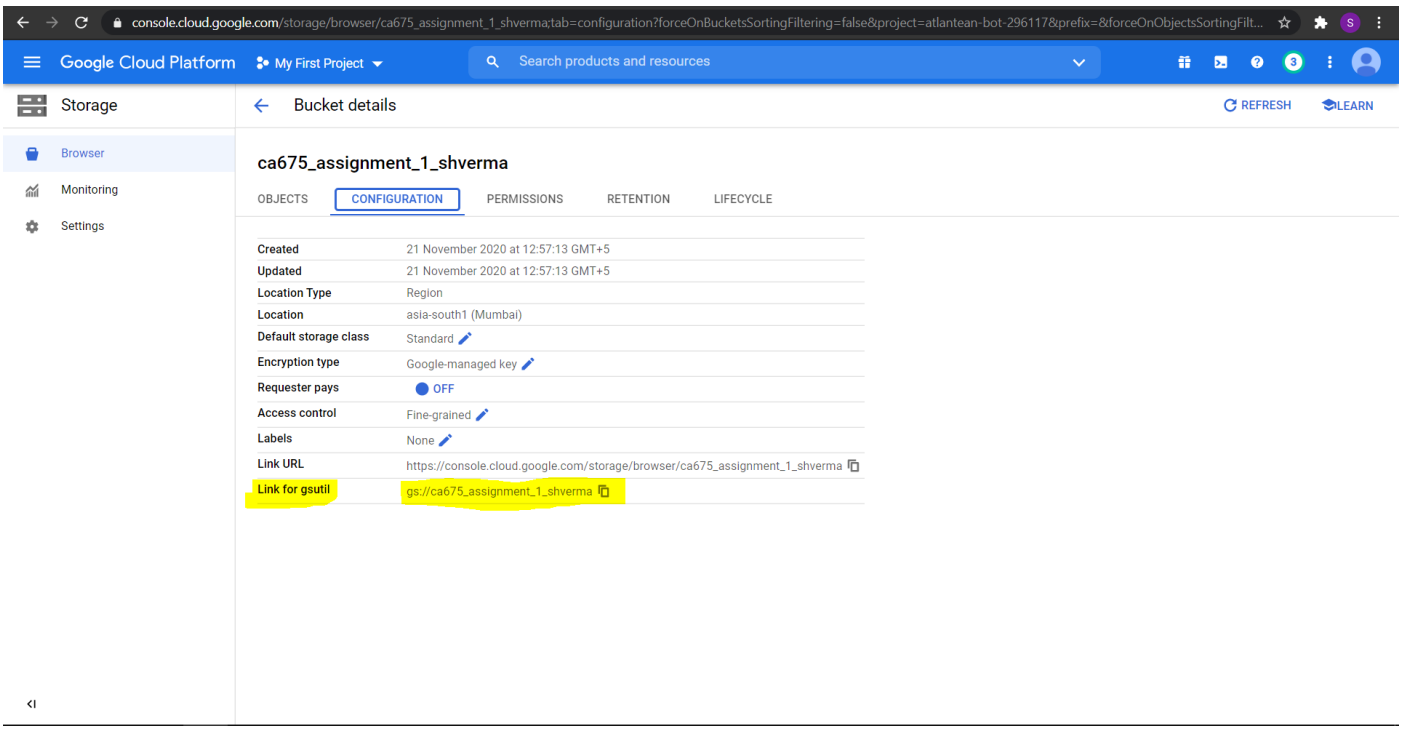
- Now, in order to obtain the top 50,000 records from these 200,002 we write the following query:
  - Select top 50,000 * from posts where post.ViewCount > 36779 order by posts.ViewCount DESC;
- For the next 50,000 records we take the lower bond of the above data and make it the new upper bound and so on and so forth:
  - Select top 50,000 * posts where posts.ViewCount <= 87000 order by posts.ViewCount DESC;
  - Select top 50,000 * posts where posts.ViewCount <= 51208 order by posts.ViewCount DESC;
  - Select top 50,000 * posts where posts.ViewCount <= 51208 order by posts.ViewCount DESC;

Now that we have downloaded all the files as csv files, we move on to the next task.

TASK 2 - Loading the data

For loading the data, I created a storage bucket on my Google Cloud Platform (GCP) console and then uploaded all the files extracted from Stack Exchange Data Explorer (SEDE) from my local machine. Futhermore, used this bucket's path for creating tables in HIVE.





TASK 3 - QUERYING with HIVE

- Hive > show databases > create database shubham > use shubham > create table > and so on……(please refer to the text file attached).

```
shubham_verma3@ca675-ass1-cluster1-m: ~ - Google Chrome
ssh.cloud.google.com/projects/atlantean-bot-296117/zones/us-central1-c/instances/ca675-ass1-cluster1-m?useAdminProxy=true&authuser=0&hl=en_GB&projectNumber=584099457243
data_analysis_posts_few_stg
data_analysis_posts_stg
Time taken: 0.102 seconds, Fetched: 3 row(s)
hive> drop table data_analysis_posts_few_stg;
OK
Time taken: 1.207 seconds
hive> show tables;
OK
data_analysis_posts
data_analysis_posts_stg
Time taken: 0.108 seconds, Fetched: 2 row(s)
hive> drop table data_analysis_posts_stg;
OK
Time taken: 0.292 seconds
hive> show tables;
OK
data_analysis_posts
Time taken: 0.1 seconds, Fetched: 1 row(s)
hive> create external table if not exists data_analysis_posts_stg
    > (
    > Id string,
    > PostTypeId string,
    > AcceptedAnswerId string,
    > ParentId string,
    > CreationDate string,
    > DeletionDate string,
    > Score string,
    > ViewCount string,
    > Body string,
    > OwnerUserId string,
    > OwnerDisplayName string,
    > LastEditorUserId string,
    > LastEditorDisplayName string,
    > LastEditDate string,
    > LastActivityDate string,
    > Title String,
    > Tags String,
    > AnswerCount string,
    > CommentCount string,
    > FavoriteCount string,
    > ClosedDate string,
    > CommunityOwnedDate string,
    > ContentLicense string
    > )
    > row format delimited
    > fields terminated by ','
    > stored as textfile location 'gs://ca675_assignment_1_shverma/';
OK
Time taken: 3.247 seconds
hive>
```

- **TASK 3.1 -** Top 10 posts by score.
- Hive > select id, title, score from data_analysis_posts_stg order by score desc limit 10;

```
hive> select id, title, Score from data_analysis_posts_stg order by Score limit 10;
Query ID = shubham_verma3_20201129211515_c1841f7f-81ac-4816-8a83-2a5060d6f5c6
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_1606683130610_0002)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      5          5        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 42.12 s
--------------------------------------------------------------------------------
OK
        NULL    NULL
<p>How can I fix this?</p>      NULL    NULL
        NULL    NULL
<blockquote>    NULL    NULL
  <p>red        NULL    NULL
</blockquote>   NULL    NULL
        NULL    NULL
"       NULL    NULL
"       NULL    NULL
"6785226"       NULL    NULL
Time taken: 51.157 seconds, Fetched: 10 row(s)
hive>
```

- **TASK 3.2 -** Top 10 users by post score.
- Hive > create table table2 as select OwnerUserId as A, SUM(Score) as B from data_analysis_posts_stg group by OwnerUserId;
- Hive > select * from table2 order by B desc limit 10;

```
--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 5.81 s
--------------------------------------------------------------------------------
OK
106709334       96766191
NULL    52186659
1630800.0       768600
28863   75155
28267   73749
36683   35055
9       20028
7       10020
4469)   7113
9000])</code> to be plotted against time and want the two trends to be of different colors and in Y-axis        6000
Time taken: 6.432 seconds, Fetched: 10 row(s)
hive>
```

- **TASK 3.3 -** The number of distinct users who used the word “Hadoop” in one of their posts.
- Hive > select COUNT(distinct OwnerUserId) from data_analysis_posts_stg where bodylike '%Hadoop%';

```
hive> select count(distinct owneruserid) from data_analysis_posts_stg where body like '%Hadoop%';
Query ID = shubham_verma3_20201129215144_5ccbc3fc-93a8-4df5-9575-94c266c0d405
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1606683130610_0004)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      5          5        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 30.33 s
--------------------------------------------------------------------------------
OK
22
Time taken: 31.029 seconds, Fetched: 1 row(s)
hive>
```

Task 4 - Using Hive calculate the per-user TF-IDF (just submit the top 10 terms for each of the top 10 users from Query 3.II)

- create temporary macro max2(x INT, y INT) if(x>y,x,y);
- create temporary macro tfidf(tf FLOAT, df_t INT, n_docs INT) tf * (log(10, CAST(n_docs as FLOAT)/max2(1,df_t)) + 1.0);
- create table tb1 as select OwnerUserId, Title,Score from data_analysis_posts_stg order by Score desc limit 10;
- create view Expld as select OwnerUserId, word from tb1 LATERAL VIEW explode(tokenize(Title, True)) t as word where not is_stopword(word);
- create view term_freq as select OwnerUserid, word, freq from (select OwnerUserId, tf(word) as word2freq from Expld group by OwnerUserId) t LATERAL VIEW explode(word2freq) t2 as word, freq;
- create or replace view tf_idf as select word, COUNT(distinct OwnerUserId) docs from Expld group by word;
- select COUNT(ownerUserId) from tf_idf;
- set hivevar:n_docs=10;

- create or replace view tf_idf as select tf.OwnerUserId, tf.word, tfidf(tf.freq, df.docs,${n_docs}) as tf_idf from term_freq tf JOIN document_frequency dfON(tf.word=df.word) order by tf_idf desc;
- select * from tf_idf;

```
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 2.66 s
--------------------------------------------------------------------------------
OK
6068    git     0.4     1
6068    pull    0.2     2
6068    fetch   0.2     2
6068    difference      0.2     2
12870   type    0.25    1
12870   content 0.25    1
12870   json    0.25    1
12870   correct 0.25    1
14069   undo    0.25    1
14069   git     0.25    1
14069   commit  0.25    1
14069   add     0.25    1
18300   keyword 0.5     1
18300   yield   0.5     1
87234   array   0.2     1
87234   processing      0.2     1
87234   unsorted        0.1     3
87234   operator        0.1     3
87234   --      0.1     3
87234   faster  0.1     3
87234   sorted  0.1     3
87234   c++     0.1     3
89904   undo    0.2     1
89904   git     0.2     1
89904   commits 0.2     1
89904   recent  0.2     1
89904   local   0.2     1
95592   branch  0.2     1
95592   git     0.2     1
95592   remotely        0.2     1
95592   locally 0.2     1
95592   delete  0.2     1
338204  rename  0.25    1
338204  branch  0.25    1
338204  local   0.25    1
338204  git     0.25    1
364969  array   0.25    1
364969  remove  0.25    1
364969  specific        0.25    1
364969  item    0.25    1
Time taken: 3.591 seconds, Fetched: 40 row(s)
```

References:

1.) https://www.tutorialspoint.com/hive/index.htm

2.) https://en.wikipedia.org/wiki/Tf%E2%80%93idf