CA675 Assignment 01: Data Analysis

Name: Maria Elveera Monis
Student ID: 20211083
E-mail: maria.monis2@mail.dcu.ie

Git Repository link

https://github.com/MariaMonis2/675Assignment01

Task 01: Data Acquisition

We are required to retrieve the 200,000 posts by viewcount from the stack exchange website. The main issue while acquiring the dataset is that we can download only 50,000 records at a time.

- We should at least execute 4 to 5 queries in total to obtain 200,000 posts. The first step is to figure out the range of values in "ViewCount" field that constitutes the top 200,000 data set. I discovered by doing a series of attempts that lower bound value for "ViewCount" field should be greater than "41423". The data range is explained clearly in DataFetching document uploaded in Git.
  select count(*) from posts where posts.ViewCount > 41423 – 200,008 records
- Since we can retrieve 50,000 records at a time, We can break the query "**where posts.ViewCount > 41423**" into 4 parts so that each can retrieve 50,000 records sorting them in a descending order.
  select top 50000 * from posts where posts.ViewCount < 53350 and posts.ViewCount > 41423 order by posts.ViewCount desc
  select top 50000 * from posts where posts.ViewCount < 74800 and posts.ViewCount > 53350 order by posts.ViewCount desc
  select top 50000 * from posts where posts.ViewCount < 128000 and posts.ViewCount > 74870 order by posts.ViewCount desc
  select top 50000 * from posts where posts.ViewCount > 128000 order by posts.ViewCount desc
- The below query is for retrieving the missed dataset to retrieve 200,000 records
  select top 150 * from posts where posts.ViewCount < 41423 order by posts.ViewCount desc

Task 02: - Data Cleaning and ETL (Extract, Transform and Load) the data

Data Cleaning through PIG

I created a cluster(cluster-ad98) under GCP and created a VM instance and uploaded the raw datasets into it. Below is the screenshot for reference.

```
maria_monis2@cluster-ad98-m:~$ ls -l
total 493728
-rw-r--r-- 1 maria_monis2 maria_monis2 62211707 Oct 27 21:30 RawDataset_1.csv
-rw-r--r-- 1 maria_monis2 maria_monis2 54678184 Oct 27 21:29 RawDataset_2.csv
-rw-r--r-- 1 maria_monis2 maria_monis2 65439234 Oct 27 21:31 RawDataset_3.csv
-rw-r--r-- 1 maria_monis2 maria_monis2 68689000 Oct 27 21:32 RawDataset_4.csv
-rw-r--r-- 1 maria_monis2 maria_monis2   229177 Oct 27 22:19 RawDataset_5.csv
```

After the datasets are uploaded, the data is cleaned using PIG by running PIG.

Browse Directory

/maria/ Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	maria_monis2	hadoop	59.33 MB	Oct 28 03:20	1	128 MB	RawDataset_1.csv
-rw-r--r--	maria_monis2	hadoop	52.15 MB	Oct 28 03:20	1	128 MB	RawDataset_2.csv
-rw-r--r--	maria_monis2	hadoop	62.41 MB	Oct 28 03:20	1	128 MB	RawDataset_3.csv
-rw-r--r--	maria_monis2	hadoop	65.51 MB	Oct 28 03:20	1	128 MB	RawDataset_4.csv
-rw-r--r--	maria_monis2	hadoop	223.81 KB	Oct 28 03:53	1	128 MB	RawDataset_5.csv

The dataset is loaded into PIG from local by the following command

hdfs dfs -copyFromLocal RawDataset_1.csv /maria/

The next step is to clean the data by using PIG commands. Below are the steps I have followed to clean the data by refering the github link which is available in **reference** section.

Step 1: Load all the dataset (5 files) into PIG by the below command:

raw1 = load 'hdfs://cluster-ad98-m/maria/RawDataset_1.csv' using org.apache.pig.piggybank.storage.CSVExcelStorage(',','YES_MULTILINE','UNIX','SKIP_INPUT_HEADER') AS (Id:int, PostTypeId:int, AcceptedAnswerId:int, ParentId:int, CreationDate:chararray, DeletionDate:chararray, Score:int, ViewCount:int, Body:chararray, OwnerUserId:int, OwnerDisplayName:chararray, LastEditorUserId:int, LastEditorDisplayName:chararray, LastEditDate:chararray, LastActivityDate:chararray, Title:chararray, Tags:chararray, AnswerCount:int, CommentCount:int, FavoriteCount:int, ClosedDate:chararray, CommunityOwnedDate:chararray, ContentLicense:chararray);

Step 2: concatenate all the files by using UNION function

rawdata = UNION raw1, raw2, raw3, raw4, raw5;

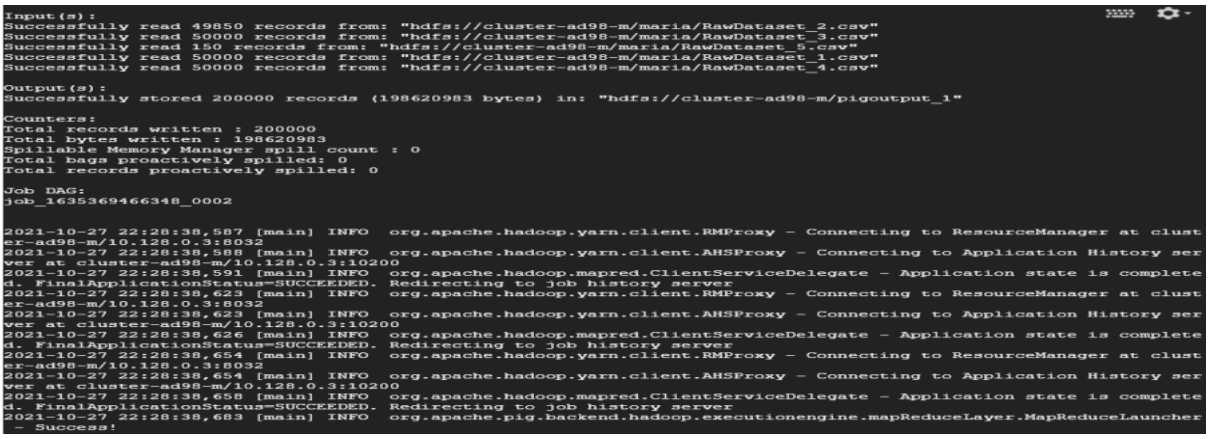
Step 3: clean the concatenated dataset by removing the special characters of "Body" and "Title" column with the help of replace function.

cleandata = FOREACH rawdata GENERATE Id, Score,ViewCount,REPLACE(REPLACE(Body,'<[^>]*>',''),'[~!@#%^&*()?,;":\n\t-=+\\`.]',' ') as Body,OwnerUserId,OwnerDisplayName,LastEditorDisplayName,REPLACE(REPLACE(Title,'<[^>]*>',''),'[~!@#%^&*()?,;":\n\t-=+\\`.]',' ') as Title,Tags;

Step 4: Store the cleaned dataset into hdfs storage by executing the following command.

STORE cleandata INTO 'hdfs://cluster-ad98-m/pigoutput_1' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',');

Below screenshot is for reference



```
Input(s):
Successfully read 49850 records from: "hdfs://cluster-ad98-m/maria/RawDataset_2.csv"
Successfully read 50000 records from: "hdfs://cluster-ad98-m/maria/RawDataset_3.csv"
Successfully read 150 records from: "hdfs://cluster-ad98-m/maria/RawDataset_5.csv"
Successfully read 50000 records from: "hdfs://cluster-ad98-m/maria/RawDataset_1.csv"
Successfully read 50000 records from: "hdfs://cluster-ad98-m/maria/RawDataset_4.csv"

Output(s):
Successfully stored 200000 records (198620983 bytes) in: "hdfs://cluster-ad98-m/pigoutput_1"

Counters:
Total records written : 200000
Total bytes written : 198620983
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1635369466348_0002


2021-10-27 22:28:38,587 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at clust
er-ad98-m/10.128.0.3:8032
2021-10-27 22:28:38,588 [main] INFO  org.apache.hadoop.yarn.client.AHSProxy - Connecting to Application History ser
ver at cluster-ad98-m/10.128.0.3:10200
2021-10-27 22:28:38,591 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is complete
d. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2021-10-27 22:28:38,623 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at clust
er-ad98-m/10.128.0.3:8032
2021-10-27 22:28:38,623 [main] INFO  org.apache.hadoop.yarn.client.AHSProxy - Connecting to Application History ser
ver at cluster-ad98-m/10.128.0.3:10200
2021-10-27 22:28:38,626 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is complete
d. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2021-10-27 22:28:38,654 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at clust
er-ad98-m/10.128.0.3:8032
2021-10-27 22:28:38,654 [main] INFO  org.apache.hadoop.yarn.client.AHSProxy - Connecting to Application History ser
ver at cluster-ad98-m/10.128.0.3:10200
2021-10-27 22:28:38,658 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is complete
d. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2021-10-27 22:28:38,683 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher
 - Success!
```

Querying data through Hive

data is being pulled from hdfs after cleaning the dataset in PIG. Run Hive by using "Hive" command.

Create a table in Hive and load the data into the newly created table.

- **CREATE TABLE STACK(Id INT ,Score INT,ViewCount INT,Body STRING, OwnerUserId INT, OwnerDisplayName STRING,LastEditorDisplayName STRING,Title STRING,Tags STRING)**
  ROW FORMAT DELIMITED

FIELDS TERMINATED BY ',';

- **LOAD DATA INPATH 'hdfs://cluster-ad98-m/pigoutput_1' INTO TABLE STACK;**

1. **The top 10 posts by score**

   select id, title, score from stack order by score desc limit 10;

```
Query ID = maria_monis2_20211030235447_15681810-8d04-4e10-8bb0-2f8a8f001219
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1635637931607_0001)

--------------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      8          8        0        0       0       0
Reducer 2 ...... container     SUCCEEDED     16         16        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 14.86 s
--------------------------------------------------------------------------------------------------
OK
11227809        Why is processing a sorted array faster than processing an unsorted array       25933
927358  How do I undo the most recent local commits in Git      23348
2003505 How do I delete a Git branch locally and remotely       18514
292357  What is the difference between 'git pull' and 'git fetch'       12834
231767  What does the  yield  keyword do        11551
477816  What is the correct JSON content type   10921
348170  How do I undo 'git add' before commit   10079
5767325 How can I remove a specific item from an array  9931
6591213 How do I rename a local Git branch      9792
1642028 What is the    >  operator in C C       9560
Time taken: 15.898 seconds, Fetched: 10 row(s)
hive>
```

2. **The top 10 users by post score**

   select sum(score) as TotalPostScore, owneruserid as users from stack where owneruserid IS NOT NULL group by ownerUserId order by TotalPostScore desc limit 10;

```
hive> select sum(score) as TotalPostScore, owneruserid as users from stack where owneruserid IS NOT NULL group by o
wnerUserId order by TotalPostScore desc limit 10;
Query ID = maria_monis2_20211031000943_32528f0a-a041-48fe-8279-74512a4ab066
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1635638896575_0002)

--------------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      8          8        0        0       0       0
Reducer 2 ...... container     SUCCEEDED     16         16        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 16.36 s
--------------------------------------------------------------------------------------------------
OK
75344   87234
57634   4883
53598   9951
51888   6068
48048   89904
47438   51816
40406   49153
39060   179736
38958   95592
38690   63051
Time taken: 17.664 seconds, Fetched: 10 row(s)
hive>
```

3. **The number of distinct users, who used the word "cloud" in one of their posts**

   select count (distinct owneruserid) from stack where (lower(body) like '%cloud%' or lower(title) like '%cloud%' or lower(tags) like '%cloud%');

```
hive> select count (distinct owneruserid) from stack where (lower(body) like '%cloud%' or lower(title) like '%cloud
%' or lower(tags) like '%cloud%');
Query ID = maria_monis2_20211031001100_7f69865b-a245-45e0-ae02-ab98867cb51e
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1635638896575_0002)

--------------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      8          8        0        0       0       0
Reducer 2 ...... container     SUCCEEDED     16         16        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 17.34 s
--------------------------------------------------------------------------------------------------
OK
918
Time taken: 18.485 seconds, Fetched: 1 row(s)
hive>
```

Calculate the per-user TF-IDF of the top 10 terms for each of the top 10 users

To install Hivemall, Downloaded the hivemall jar file and placed it into local folder by using the following command

wget https://github.com/myui/hivemall/releases/download/v0.4.2-rc.2/hivemall-core-0.4.2-rc.2-with-dependencies.jar

In Hive, add the jar file as well as define-all.Hive file. Define macros for TF-IDF computation

add jar hivemall-core-0.4.2-rc.2-with-dependencies.jar;
source define-all.hive;
create temporary macro max2(a INT, b INT) if(a>b,a,b);
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);

Now, create tables to calculate TF-IDF by refering to the github link shared in references.

create table topUsers as select owneruserid, sum(score) as TotalScore from stack group by OwnerUserId order by TotalScore desc limit 10;

create table topUsers1 as select d.OwnerUserID,title from stack d join topUsers t on d.OwnerUserID = t.OwnerUserID;

Now, created a view named topUsersExplode which will select ownerUserId and eachword from "topUsers1" table. Also created 2 more views named term_freq and doc_freq.

create or replace view topUsersExplode as select ownerUserId, eachword from topUsers1 LATERAL VIEW explode(tokenize(Title, True)) t as eachword where not is_stopword(eachword);

create or replace view term_freq as select ownerUserid, eachword, freq from (select ownerUserId, tf(eachword) as word2freq from topUsersExplode group by ownerUserId) t LATERAL VIEW explode(word2freq) t2 as eachword, freq;

create or replace view doc_freq as select eachword, count(distinct ownerUserID) docs from topUsersExplode group by eachword;

The below query creates a view name tfidf which will have ownerUserId, each word and the frequency calculated for each word

create or replace view tfidf as select tf.ownerUserId, tf.word, tfidf(tf.freq,df.docs,10) as tfidf from term_freq tf JOIN doc_freq df on (tf.word = df.word) order by tfidf desc;

The below query shows all the result from the view "tfidf" with frequency calculated for each word

select * from tfidf;

The results are available in the below screenshot

```
ive> create or replace view tfidf as select tf.ownerUserId, tf.eachword, tfidf(tf.freq,df.docs,10) as tfidf from term_freq tf JOIN doc_freq df on (tf.eachword = df.eachword)
rder by tfidf desc;
K
ime taken: 0.159 seconds
ive> select * from tfidf;
ARNING: Order/Sort by without limit in sub query or view [tfidf] is removed, as it's pointless and bad for performance.
uery ID = maria_monis2_20211028120050_e77c70ca-d88b-4d63-9954-efff60c3399c
otal jobs = 1
aunching Job 1 out of 1
ez session was closed. Reopening...
ession re-established.
ession re-established.
tatus: Running (Executing on YARN cluster with App id application_1635417888635_0004)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
ap 1 .......... container     SUCCEEDED      1          1        0        0       0       0
educer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
educer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
ERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 7.66 s
--------------------------------------------------------------------------------
K
883    find    0.016944726841288636
883    subversion      0.012121211737394333
883    write   0.018459135721920383
883    convert 0.008472363420644318
883    expressions     0.012121211737394333
883    difference      0.018459135721920383
883    ruby    0.027688703582880576
883    checkbox        0.012121211737394333
883    model   0.012121211737394333
883    symbol  0.012121211737394333
883    windows 0.024242423474788666
883    merge   0.009229567860960191
883    tcp     0.012121211737394333
883    function        0.009229567860960191
```

```
95592   convention      0.0476190485060215
95592   virtualenv      0.08090333504675243
179736  spaces  0.0029547305052710997
179736  logcat  0.0034782609436661005
179736  objectid        0.0034782609436661005
179736  entropy 0.0034782609436661005
179736  virtualenv      0.0029547305052710997
179736  files   0.0042499087848578325
179736  exif    0.0034782609436661005
179736  give    0.0034782609436661005
179736  variable        0.004862400133752198
179736  file    0.008034100762363 68
179736  assign  0.0034782609436661005
179736  regex   0.0029547305052710997
179736  id      0.0059094610105421995
179736  importing       0.0034782609436661005
179736  position        0.0029547305052710997
179736  something       0.010434783063828945
179736  non     0.0034782609436661005
179736  read    0.0029547305052710997
179736  raising 0.0034782609436661005
179736  activity        0.0034782609436661005
179736  information     0.0034782609436661005
179736  letter  0.0034782609436661005
179736  values  0.002431200066876099
179736  circle  0.0034782609436661005
179736  reset   0.0029547305052710997
179736  meta    0.0034782609436661005
179736  simple  0.008864191713599439
Time taken: 15.268 seconds, Fetched: 1249 row(s)
hive>
```

References

https://github.com/mainkoon81/DCU-project-03-BigData-DataWarehouse

https://hivemall.apache.org/download.html