**Data Analysis Using Cloud Technologies**

|  |  |  |
| --- | --- | --- |
| Name: Rohit Nair | Student No: 20210378 | Date: 22nd Nov 2020 |

Git Hub Link: <https://github.com/nrohit78/PigHive_StackExhangeData>

# Introduction

The objective of this project is to procure data from stack exchange, transform and clean it in Pig, stored the cleaned result in Hive and perform information retrieval (TF-IDF) in Hive using Hivemall.

4 CSV files, consisting of 50 thousand records each are downloaded from Stack Exchange. The table structure is as follows:

*Id: int*

*PostTypeId: tinyint*

*AcceptedAnswerId: int*

*ParentId: int*

*CreationDate: datetime*

*DeletionDate: datetime*

*Score: int*

*ViewCount: int*

*Body: nvarchar (max)*

*OwnerUserId: int*

*OwnerDisplayName: nvarchar (40)*

*LastEditorUserId: int*

*LastEditorDisplayName: nvarchar (40)*

*LastEditDate: datetime*

*LastActivityDate: datetime*

*Title: nvarchar (250)*

*Tags: nvarchar (250)*

*AnswerCount: int*

*CommentCount: int*

*FavoriteCount: int*

*ClosedDate: datetime*

*CommunityOwnedDate: datetime*

*ContentLicense: varchar (12)*

# Tasks

1. Acquire the top 200,000 posts by viewcount from stack exchange.
2. Using Pig or MapReduce , extract, transform and load the data as applicable
3. Using Hive and/or MapReduce , get:
   1. I. The top 10 posts by score.
   2. The top 10 users by post score.
   3. The number of distinct users, who used the word “Hadoop” in one of their posts
4. Using Mapreduce /Pig/Hive calculate the per user TF IDF.

# Data Acquisition

Stack Exchange provides 50000 records atmost in each file. To fetch 2 lakh records, we will have to create 4 queries which are as follows:

select TOP 50000 \* from posts

where posts.ViewCount > 110000

ORDER BY posts.ViewCount DESC;

select TOP 50000 \* from posts

where posts.ViewCount < 112524 AND posts.ViewCount > 60000 AND posts.Id != 904910

ORDER BY posts.ViewCount DESC;

select TOP 50000 \* from posts

where posts.ViewCount < 66244 AND posts.ViewCount > 45000 AND posts.Id != 20482207

ORDER BY posts.ViewCount DESC;

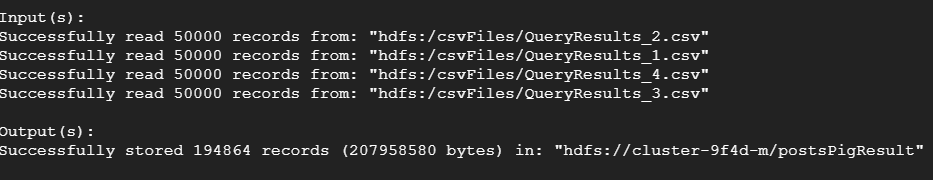
select TOP 50000 \* from posts

where posts.ViewCount < 47291 AND posts.ViewCount > 30000 AND posts.Id not in (24853847,45351434,488811,2293592,14476448)

ORDER BY posts.ViewCount DESC;

# Pig Data Transformation

Each CSV file is loaded using “LOAD” command and then combined using “UNION” command. As multiline text in the CSV file was creating an issue while loading CSVExcelStorage [1] was used instead of PigStorage. Id and OwnerUserId were needed for further processing, all records that didn’t have a value in these 2 columns were filtered out. After fitering records Body, Score, Id, ViewCount, OwnerUserId, OwnerDisplayName, Title, Tags columns were generated and the cleaned data was stored.



# Hive Loading And Query

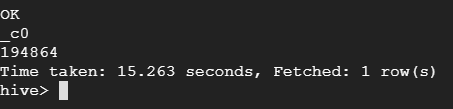
A table in hive was created to load data with the following create statement:

CREATE TABLE postsDB.posts

(Body string, Score int, Id int, ViewCount int, OwnerUserId int, OwnerDisplayName string, Title string, Tags string)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

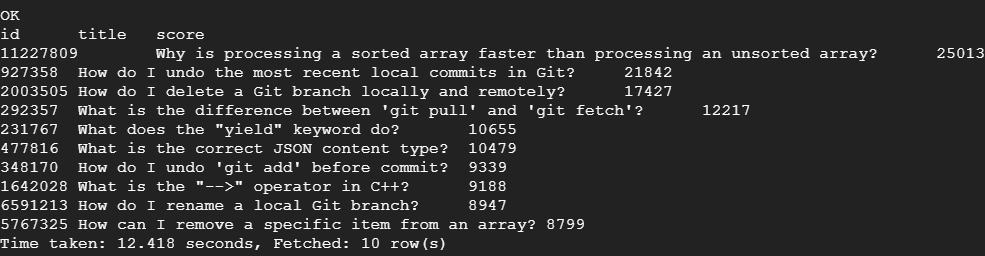
Pig generates 4 output files which is used to load data into the Hive table using 4 different load commands. The total number of records inserted into the hive table were 194864 which is equal to the records generated by the Pig.



Hive queries for task 2 are as follows:

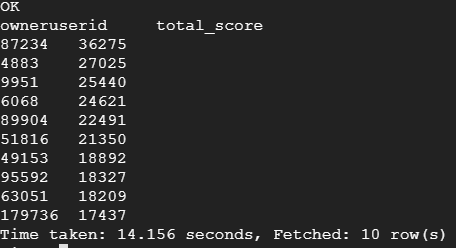
SELECT Id, Title, Score FROM postsDB.posts

ORDER BY Score DESC LIMIT 10;



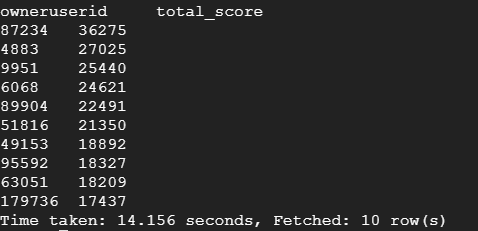
SELECT OwnerUserId, SUM(Score) AS Total\_Score FROM postsDB.posts GROUP BY OwnerUserId

ORDER BY Total\_Score DESC LIMIT 10;



SELECT COUNT(DISTINCT OwnerUserId) AS User\_Count FROM postsDB.posts

WHERE (LOWER(Body) LIKE '%hadoop%' OR LOWER(Title) LIKE '%hadoop%' OR LOWER(Tags) LIKE '%hadoop%');

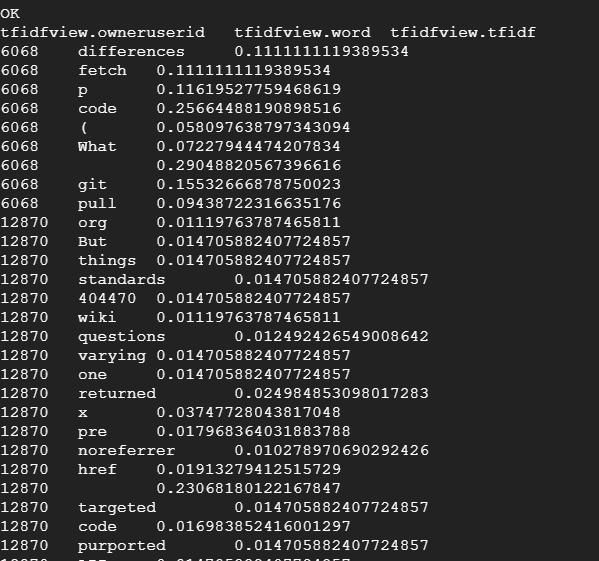


In the 3rd query, a case insensitive search is done. This fetches all records irrespective of the case. A case sensitive search can also be done by removing “LOWER” function.

# TF-IDF Using Hive

The data for the last task is being stored in a separate table and this table is being used to calculate the TF-IDF.

First, the entire body section must be split into different words and stop words need to be removed while doing so. To handle this, Hivemall [2][3] was used. Each record is being split into its corresponding word; values of the word frequency, the document frequency and finally TF-IDF is being calculated by the different views that have been created.



These are a few records from the final view that calculates TF-IDF for each user.

# Note

1. All files and screenshots are added to the GitHub repo shared in the beginning of this document.
2. The screenshots added in this document are cropped to keep the document short and concise.

# References

1. *“CSVExcelStorage,”* [Online]

Available: <https://pig.apache.org/docs/r0.17.0/api/org/apache/pig/piggybank/storage/CSVExcelStorage.html>

1. *“TF-IDF Term Weighting,”* [Online].

Available: <https://hivemall.incubator.apache.org/userguide/ft_engineering/tfidf.html>.

1. *“Hivemall User Manual,”* [Online]

Available:

<https://hivemall.incubator.apache.org/userguide/getting_started/installation.html>