**QUERIES ALONG WITH THEIR SNAPSHOTS**

|  |  |
| --- | --- |
| **Query 1** | Load the Netflix dataset of Daily Top 10 Shows/Movies into a relation by using Apache Pig |
|  | netflix = load 'Project/netflix.csv' using PigStorage(',') as (topdate:chararray,dailyrank:int, lwrank:int,title:chararray,type:chararray,excl:chararray,rdate:chararray,topdays:int,score:int); |
|  |  |
| **Query 2** | Display title, type and release date of each show/movie |
|  | rel1 = foreach netflix generate title, type, rdate; |
|  |  |
| **Query 3** | Display the show/movie details in descending order by Viewership Score |
|  | order\_score = order netflix by score desc; |
|  |  |
| **Query 4** | Display details of 100 show/movies from a sample of 10% of the above dataset |
|  | sample\_netflix = sample netflix 0.1;  limit\_sample = limit sample\_netflix 100; |
|  |  |
| **Query 5** | Display the details of movies only |
|  | movies\_netflix = filter netflix by LOWER(type)=='movie'; |
|  |  |
| **Query 6** | Display the details of movies having viewership score more than 75 |
|  | viewer\_netflix = filter netflix by score>75; |
|  |  |
| **Query 7** | Display the title and type of the shows/movies present in Netflix Exclusive and being in top 10 for more than 7 times (and with two conditions) |
|  | topexclusive = foreach (filter netflix by excl=='Yes' and topdays==7) generate title, type; |
|  |  |
| **Query 8** | Categorize the titles according to their type |
|  | group\_titles = group netflix by type; |
|  |  |
| **Query 9** | Display the count of the titles and the average of their viewership score type-wise |
|  | group\_count = foreach group\_titles generate group, COUNT(netflix), AVG(netflix.score); |
|  |  |
| **Query 10** | Convert each field of a tuple to an individual tuple such that those should be in a bag (ToBag) |
|  | bag\_netflix = foreach netflix generate TOBAG(topdate, dailyrank, lwrank, title, type, excl, rdate, topdays, score); |
|  |  |
| **Query 11** | Display all the titles which starts with ‘The’ and contains ‘of’ within it. |
|  | the\_titles = filter netflix by STARTSWITH(title,'The') and title matches '.\*of.\*'; |
|  |  |
| **Query 12** | Display the titles, type and score of type TV Shows after changing it to TV Series |
|  | series\_netflix = foreach netflix generate title, REPLACE(type,'Show','Series'), score; |
|  |  |
| **Query 13** | Generate four relations containing titles of each type TV Show, Movies, Concert and Stand-up Comedy (each in one relation) and store each one of them |
|  | split netflix into tvshows if(type=='TV Show'), movies if(type=='Movie'), comedy if(type matches '.\*Comedy.\*'), concert if (type matches '.\*Concert.\*');  store tvshows into 'TV Shows' using PigStorage(',');  store movies into 'Movies' using PigStorage(',');  store comedy into 'Comedy' using PigStorage(',');  store concert into 'Concert' using PigStorage(','); |
|  |  |
| **Query 14** | Group the relations of TV Shows and Movies on the basis of if these are present in Netflix Exclusive or not, and show the count of each |
|  | movies\_group = group movies by excl;  tvshows\_group = group tvshows by excl;  movies\_exl\_count = foreach movies\_group generate group, COUNT(movies);  tv\_exl\_count = foreach tvshows\_group generate group, COUNT(tvshows); |
|  |  |
| **Query 15** | Rank the titles on the basis of Viewership Score and display their rank, titles and type |
|  | ranked\_netflix = rank netflix by score;  ranked\_titles = foreach ranked\_netflix generate $0, title, type; |
|  |  |
| **Query 16** | Display the details of titles released in 2020 |
|  | recent\_netflix = filter netflix by GetYear(ToDate(rdate,'MM-dd-yyyy'))==2020; |
|  |  |
| **Query 17** | Display the titles and type which were released before three years and are not present in Netflix Exclusive |
|  | before\_titles = filter netflix by YearsBetween(CurrentTime(),ToDate(rdate,'MM-dd-yyyy'))>3 and excl is null; |
|  |  |
| **Query 18** | Display the titles and their type which were released in the month July and are in top 3 in July of 2020 |
|  | july\_titles = filter netflix by GetMonth(ToDate(rdate,'MM-dd-yyyy'))==7 and dailyrank==3 and GetMonth(ToDate(topdate,'dd-MM-yyyy'))==7; |
|  |  |
| **Query 19** | Display the count of TV Shows year-wise |
|  | tvyear\_group = group tvshows by GetYear(ToDate(rdate,'MM-dd-yyyy'));  tvyear\_count = foreach tvyear\_group generate group, COUNT(tvshows); |
|  |  |
| **Query 20** | Display the titles which were released in between of 10th and 30th week of year 2015 to 2018 |
|  | week\_year\_titles = filter netflix by GetWeek(ToDate(rdate,'MM-dd-yyyy'))>=10 and GetWeek(ToDate(rdate,'MM-dd-yyyy'))<=30 and GetYear(ToDate(rdate,'MM-dd-yyyy'))>=2015 and GetYear(ToDate(rdate,'MM-dd-yyyy'))<=2018; |
|  |  |

**CASE STUDY 1**

**Most occurred Start Letter**

Analyze large text file ‘shakespeare.txt’, and

1. Find the most occurred start letter
2. Calculate the wordcount for each word.

Loading the file ‘shakespeare.txt’ into a relation ‘rel’:

rel = load 'Project/shakespeare.txt' as (line:chararray);

Tokenizes each word from the ‘line’ field of above relation ‘rel’ as word:

words = foreach rel generate flatten(TOKENIZE(line)) as word;

Extracting first letter of each word:

letters = foreach words generate SUBSTRING(word,0,1) as letter;

Removing the tuples if any contains NULL values:

new\_letters = filter letters by letter is not NULL;

Grouping the letters on the basis of their similarity:

grp = group new\_letters by letter;

Counting the letter present in each bag of the groups:

count\_letter = foreach grp generate group, COUNT(new\_letters);

Sorting the letter in descending order:

order\_letter = order count\_letter by $1 desc;

Generating the letter having count maximum among all

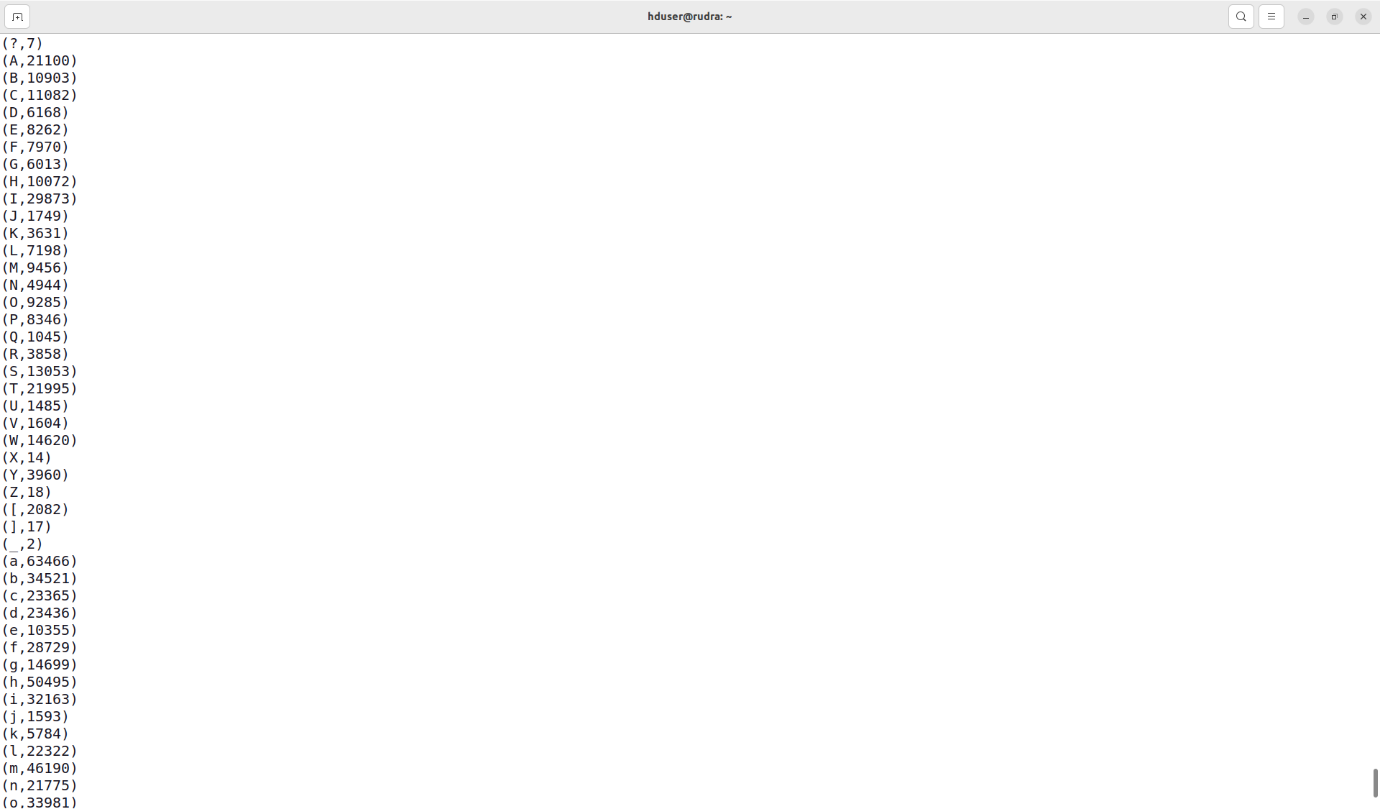
grp2 = group order\_letter all;

x = foreach grp2 {top= TOP(1,1,order\_letter); generate top;};

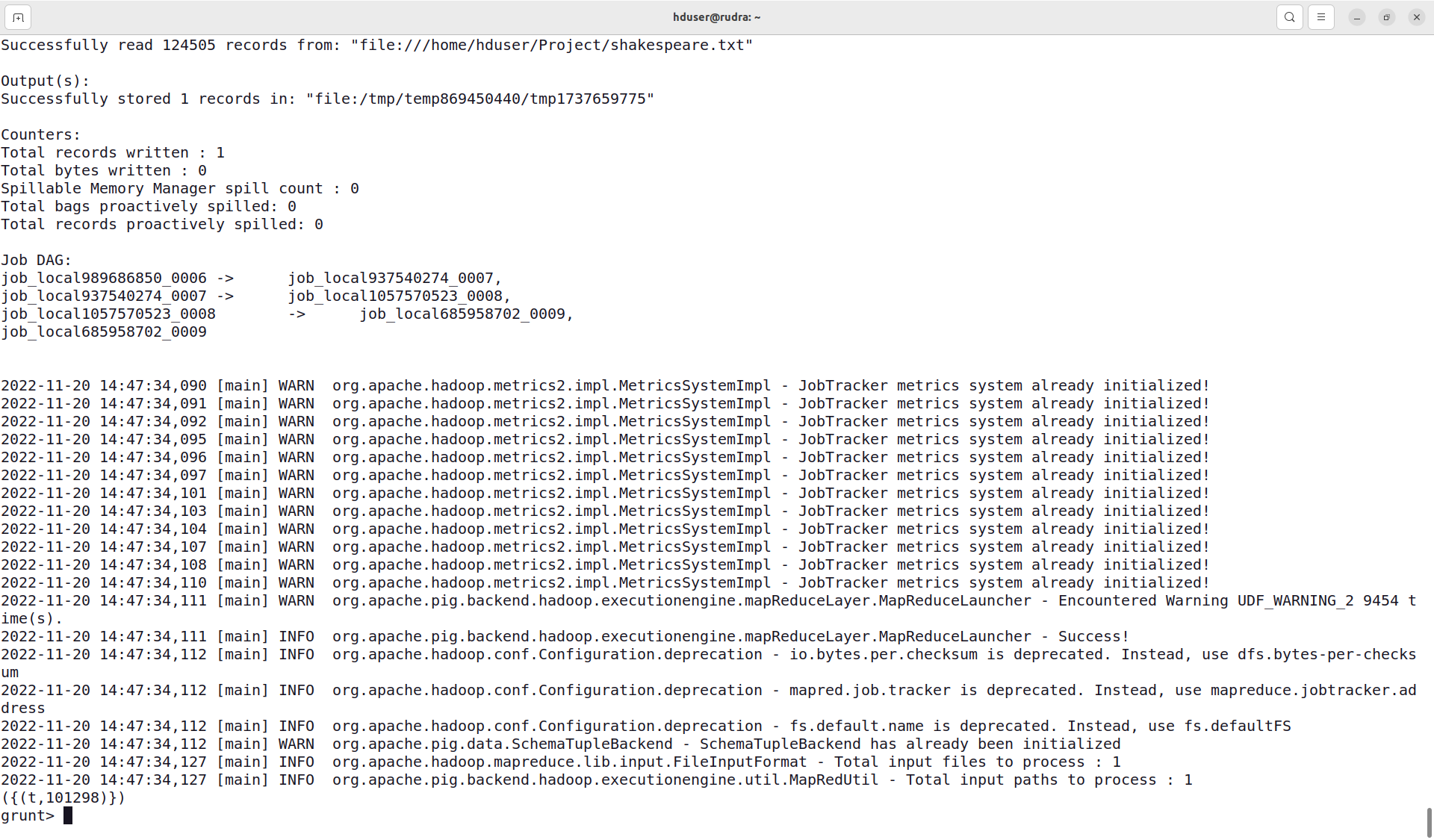
Display the count of occurrences of all letters and letter having maximum occurrence

dump x

dump count\_letter

****

Occurrences of each Letter

****

Most Occurred Start Letter and it’s occurence

**CASE STUDY 2**

**Word Count**

Removal of stopwords and punctuation from a large text file. Calculate the occurrence of each word from the file afterwards.

Loading the ‘shakespeare.txt’ file into relation ‘wordfile’ and ‘stop-word-list.csv’ file into a relation ‘stopword’:

wordfile = load 'Project/shakespeare.txt' as (line:chararray);

stopword = load 'Project/stop-word-list.csv' using PigStorage();

Tokenize each word after removing punctuations and changing those to Lower case from ‘wordfile’ relation:

words = foreach wordfile generate flatten(TOKENIZE(REPLACE(LOWER(TRIM(line)),'[\\p{Punct},\\p{Cntrl}]', ' '))) as word;

Removing any tuple if containing NULL values from ‘words’ relation:

new\_words = filter words by word is not NULL;

Tokenize each word from ‘stopword’ relation:

stopword = foreach stopword generate flatten(TOKENIZE($0)) as (sword:chararray);

Combine ‘wordfile’ and ‘stopword’ relation on the basis of words:

join\_word = join words by word left outer, stopword by sword;

Removing the tuples where ‘stopword’ field ‘sword’ is NULL:

filter\_word = filter join\_word by stopword::sword is NULL;

Now extracting the words remaining in the ‘words’ relation:

words1 = foreach filter\_word generate words::word as word;

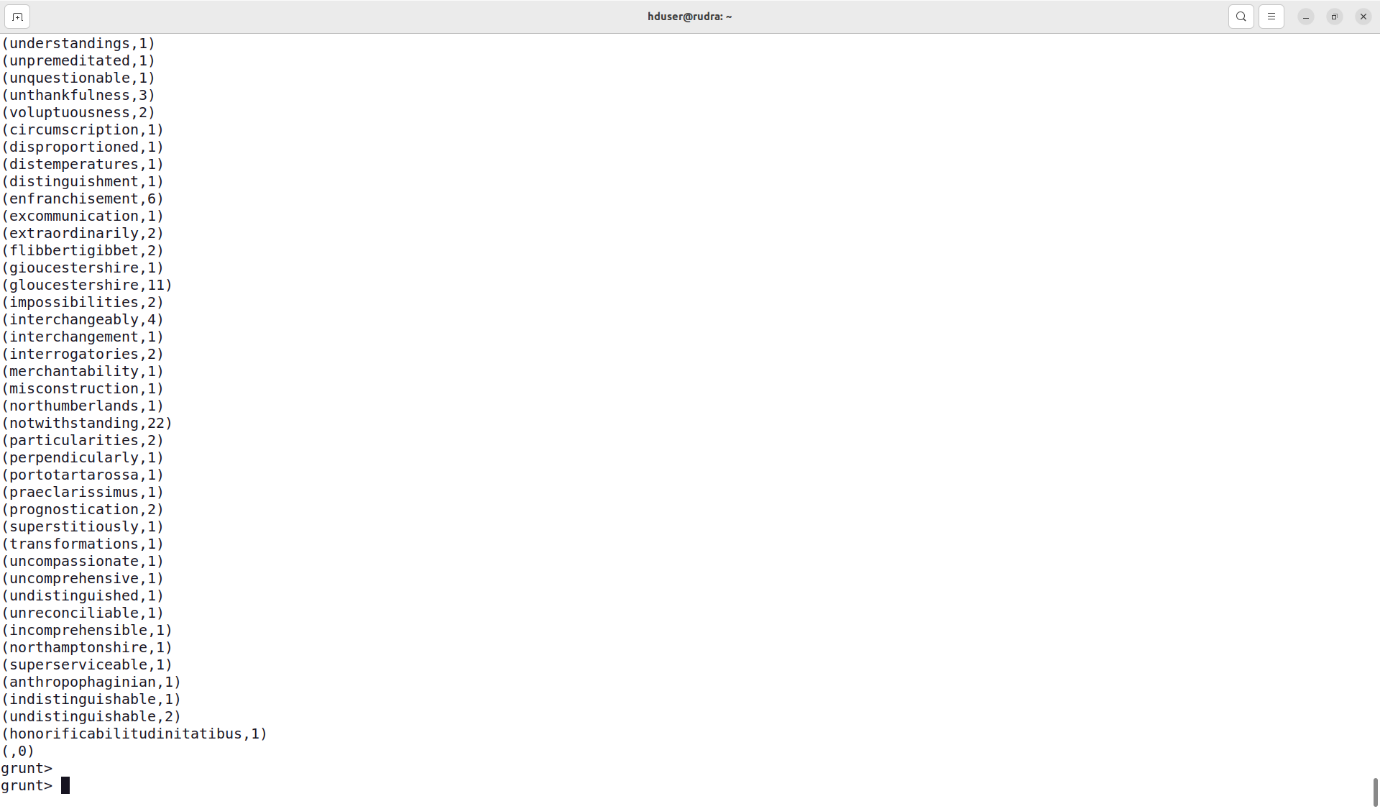
Group the words on the basis of their simnilarity:

grp = group words1 by word;

Count the occurrences of the word present:

count\_word = foreach grp generate group,COUNT(words1);

dump count\_word

****

Word count – Occurrence of each Word