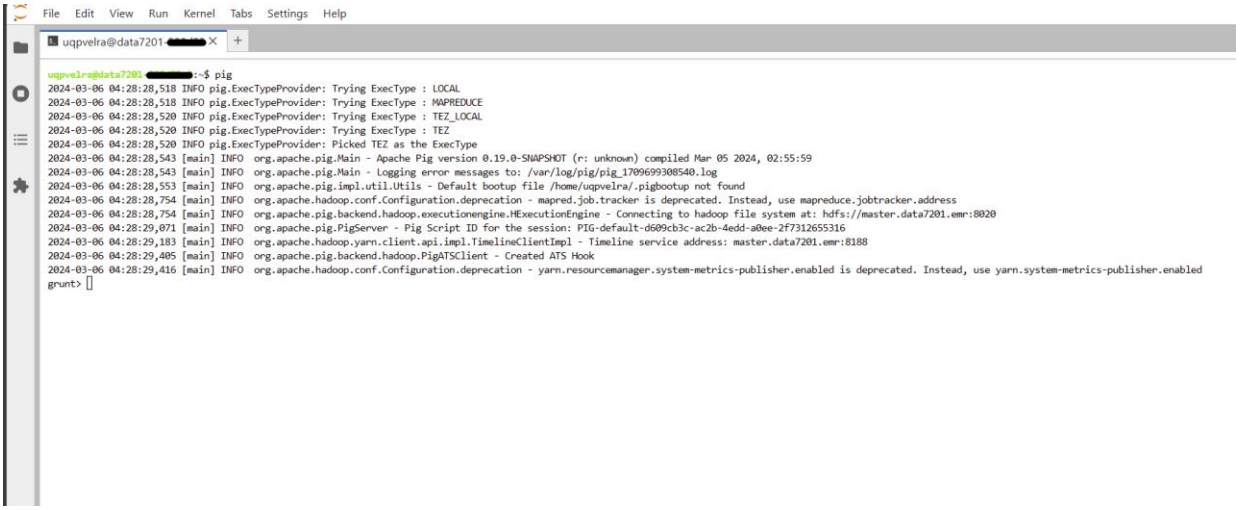
Week 4 - DATA7201 Practical session – Pig scripts

Setting up pig

Open pig via the command “pig” in the terminal.



Use Linux command in Pig

Let’s create a new folder for the prac, call it ‘prac-2’ in HDFS at your user folder. Submit in pig:

grunt> mkdir prac-2

As you can see, pig can also help us to operate HDFS.

Word Count - PIG

The first PIG script example is a “word count” problem. Given the text file of the last prac, we want to count how many times each word appears in that file. This can be done by the following script:

```
1 lines = LOAD '/data/pg46.txt' AS (line:chararray);
2 words = FOREACH lines GENERATE FLATTEN(TOKENIZE(line)) as word;
3 grouped = GROUP words BY word;
4 wordcount = FOREACH grouped GENERATE group, COUNT(words) as c;
5 DUMP wordcount;
6 STORE wordcount INTO '/user/USERNAME/output/pg46count.txt' USING PigStorage('|');
```

Line 1: loads text data from a file stored on HDFS reading one line at a time with each line being an array of characters.
Line 2: uses TOKENIZE(line) to split words using spaces taking each line separately and producing a set of terms called words.
Line 3: creates groups of the same words appearing across different lines.
Line 4: creates a list of word-frequency pairs.
Line 5 and 6: will show the results to console and store it into a file in HDFS respectively.

1. Exercise

Run the script by taking into consideration:

- Use the file loaded in prac-1
- Save the file with the counts in the new folder prac-2

In the final stats, did you have the same number of lines read as the result in prac 1?

```
Success!


DAG 0:
                                  Name: PigLatin:DefaultJobName-0_scope-0
                         ApplicationId: job_1709297397972_0072
                     TotalLaunchedTasks: 2
                          FileBytesRead: 238215
                       FileBytesWritten: 238151
                          HdfsBytesRead: 1256167
                       HdfsBytesWritten: 306124
      SpillableMemoryManager spill count: 0
                Bags proactively spilled: 0
             Records proactively spilled: 0

DAG Plan:
Tez vertex scope-17     ->      Tez vertex scope-18,
Tez vertex scope-18

Vertex Stats:
VertexId Parallelism TotalTasks   InputRecords   ReduceInputRecords  OutputRecords  FileBytesRead FileBytesWritten  HdfsBytesRead HdfsBytesWritten Alias        Feature Outputs
scope-17          1          1          23244                    0         217845             64           238151        1256167               0 grouped,lines,wordcount,words

scope-18          1          1              0                28352          28352         238151                0              0          306124 wordcount    GROUP_BY        hdfs:/
/master.data7201.emr:8020/user/uqpvelra/prac-2/moby10bcount.txt,

Input(s):
Successfully read 23244 records (1256167 bytes) from: "hdfs://master.data7201.emr:8020/user/uqpvelra/prac-1/moby10b.txt"

Output(s):
Successfully stored 28352 records (306124 bytes) in: "hdfs://master.data7201.emr:8020/user/uqpvelra/prac-2/moby10bcount.txt"
```

Debugging – describe

"Describe" is useful when you want to know the format of the fields. Let's use it with grouped:

grunt> describe grouped;

grouped: {group: chararray,words: {(token: chararray)}}

Debugging – limit

"Limit" is not a debugging tool, but it can be useful as well. It is useful to understand how the pipeline works.

Let's add the following code between the lines 1 and 2:

limit_lines = limit lines 10;

Replace "lines" with "limit_lines" in the line 3, and dump one by one the following aliases (don't dump them all at once):

- limit_lines
- words
- grouped
- wordcount

dump limit_lines

(**The Project Gutenberg Etext of Moby Dick, by Herman Melville**)
(#3 in our series by Herman Melville)

()
(This Project Gutenberg version of Moby Dick is based on a combination)
(of the etext from the ERIS project at Virginia Tech and another from)
(Project Gutenberg's archives, as compared to a public-domain hard copy.)
()
(Copyright laws are changing all over the world, be sure to check)
(the copyright laws for your country before posting these files!!)
()

Dump words

(The)
(Project)
(Gutenberg)
(Etext)
(of)
(Moby)
(Dick)
(by)
(Herman)
(Melville)
(#3)
(in)
(our)
(series)
(by)
(Herman)
(Melville)
()
(This)
(Project)
(Gutenberg)
(version)
(of)
(Moby)
(Dick)
(is)
(based)
(on)
(a)
(combination)
(of)
(the)
(etext)
(from)
(the)
(ERIS)
(project)
(at)
(Virginia)
(Tech)
(and)
(another)
(from)
(Project)
(Gutenberg’s)
(archives)
(as)
(compared)
(to)
(a)
(public-domain)
(hard)

(copy.)
()
(Copyright)
(laws)
(are)
(changing)
(all)
(over)
(the)
(world)
(be)
(sure)
(to)
(check)
(the)
(copyright)
(laws)
(for)
(your)
(country)
(before)
(posting)
(these)
(files!!)
()

Dump grouped

(a,{(a),(a)})
(#3,{(#3)})
(as,{(as)})
(at,{(at)})
(be,{(be)})
(by,{(by),(by)})
(in,{(in)})
(is,{(is)})
(of,{(of),(of),(of)})
(on,{(on)})
(to,{(to),(to)})
(The,{(The)})
(all,{(all)})
(and,{(and)})
(are,{(are)})
(for,{(for)})
(our,{(our)})
(the,{(the),(the),(the),(the)})
(Dick,{(Dick),(Dick)})
(ERIS,{(ERIS)})
(Moby,{(Moby),(Moby)})
(Tech,{(Tech)})
(This,{(This)})
(from,{(from),(from)})
(hard,{(hard)})
(laws,{(laws),(laws)})
(over,{(over)})
(sure,{(sure)})
(your,{(your)})
(Etext,{(Etext)})
(based,{(based)})
(check,{(check)})
(copy.,{(copy.)})
(etext,{(etext)})
(these,{(these)})

(world,{(world)})
(Herman,{(Herman),(Herman)})
(before,{(before)})
(series,{(series)})
(Project,{(Project),(Project),(Project)})
(another,{(another)})
(country,{(country)})
(files!!,{(files!!)})
(posting,{(posting)})
(project,{(project)})
(version,{(version)})
(Melville,{(Melville),(Melville)})
(Virginia,{(Virginia)})
(archives,{(archives)})
(changing,{(changing)})
(compared,{(compared)})
(Copyright,{(Copyright)})
(Gutenberg,{(Gutenberg),(Gutenberg)})
(copyright,{(copyright)})
(Gutenberg's,{(Gutenberg's)})
(combination,{(combination)})
(public-domain,{(public-domain)})
(,{(),(),()})

Dump wordcount

(a,2)
(#3,1)
(as,1)
(at,1)
(be,1)
(by,2)
(in,1)
(is,1)
(of,3)
(on,1)
(to,2)
(The,1)
(all,1)
(and,1)
(are,1)
(for,1)
(our,1)
(the,4)
(Dick,2)
(ERIS,1)
(Moby,2)
(Tech,1)
(This,1)
(from,2)
(hard,1)
(laws,2)
(over,1)
(sure,1)
(your,1)
(Etext,1)
(based,1)
(check,1)
(copy.,1)
(etext,1)
(these,1)
(world,1)
(Herman,2)

(before,1)
(series,1)
(Project,3)
(another,1)
(country,1)
(files!!,1)
(posting,1)
(project,1)
(version,1)
(Melville,2)
(Virginia,1)
(archives,1)
(changing,1)
(compared,1)
(Copyright,1)
(Gutenberg,2)
(copyright,1)
(Gutenberg's,1)
(combination,1)
(public-domain,1)
(,0)

2. Exercise

Count word frequency and find the top words in the HDFS file /data/candy.txt (a list of candy recipes). Is there any ingredient in the top list?
[Hint: use something like “sorted = RANK wordcount BY c DESC;”]

```
lines =LOAD '/data/candy.txt' AS (line:chararray);
words_token = FOREACH lines GENERATE TOKENIZE(line);
words = FOREACH words_token GENERATE FLATTEN(bag_of_tokenTuples_from_line) as token;
grouped = GROUP words BY token;
wordcount = FOREACH grouped GENERATE group, COUNT(words) as counts;
sorted = RANK wordcount BY counts DESC;
top10 = limit sorted 10;
dump top10;
```

(1,and,1700)
(2,c,1321)
(3,1,1308)
(4,the,1265)
(5,in,1228)
(6,a,1063)
(7,to,986)
(8,until,748)
(9,1/2,673)
(10,2,665)

3. Exercise

Count word frequency and find the top words in t2.txt (the script of the Terminator 2 movie – download from blackboard). You will see that the word 'the' and 'The' both appears in top 20: Adapt your script by counting words ignoring case (upper/lower case). You can do this by transforming all words in lowercase first, and then count.
[Hint: use the PIG function LOWER() in a similar way as TOKENIZE()]

Return only the top 20 words in the file by sorting the words by frequency and limiting the results to 20 using the RANK and LIMIT commands.

The full list of PIG commands with examples is here:
Pig Latin Basics (apache.org)
https://pig.apache.org/docs/r0.9.1/func.html

Upload t2.txt in your zone using jupyter
In terminal cd /var/www/notebooks
Make directory prac-3 and put t2.txt file to the hdfs

```
hdfs dfs -mkdir prac-3
hdfs dfs -put t2.txt prac-3
```

```
lines =LOAD '/user/uqpvelra/prac-3/t2.txt' AS (line:chararray);
words_token = FOREACH lines GENERATE TOKENIZE(line);
words = FOREACH words_token GENERATE FLATTEN(bag_of_tokenTuples_from_line) as token;
words_lower = FOREACH words GENERATE LOWER(token) as token;
grouped = GROUP words_lower BY token;
wordcount = FOREACH grouped GENERATE group, COUNT(words_lower) as counts;
sorted = RANK wordcount BY counts DESC;
top20 = limit sorted 20;
dump top20;
```

```
(1,the,2826)
(2,a,921)
(3,and,849)
(4,of,674)
(5,to,646)
(6,in,498)
(7,is,464)
(8,terminator,442)
(9,john,440)
(10,it,422)
(11,he,404)
(12,his,360)
(13,on,332)
(14,sarah,316)
(15,she,287)
(16,with,285)
(17,her,268)
(18,as,261)
(19,at,257)
(20,into,233)
```