

ASSIGNMENT 4.3

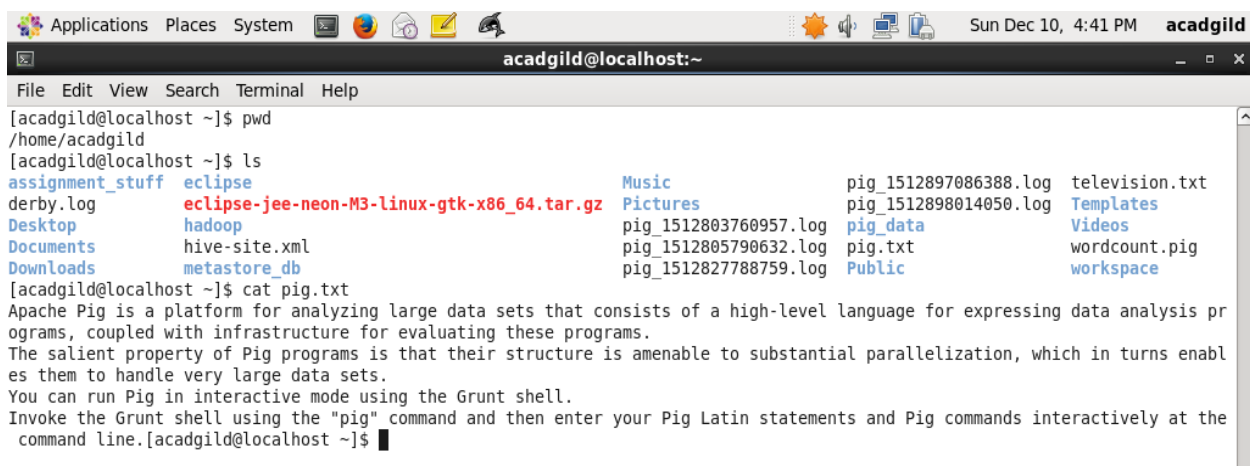
Problem Statement:

Write a program to implement word count using Pig.

Solution:

Here is the data in text file format that I have used as input for this problem:

Pig.txt:

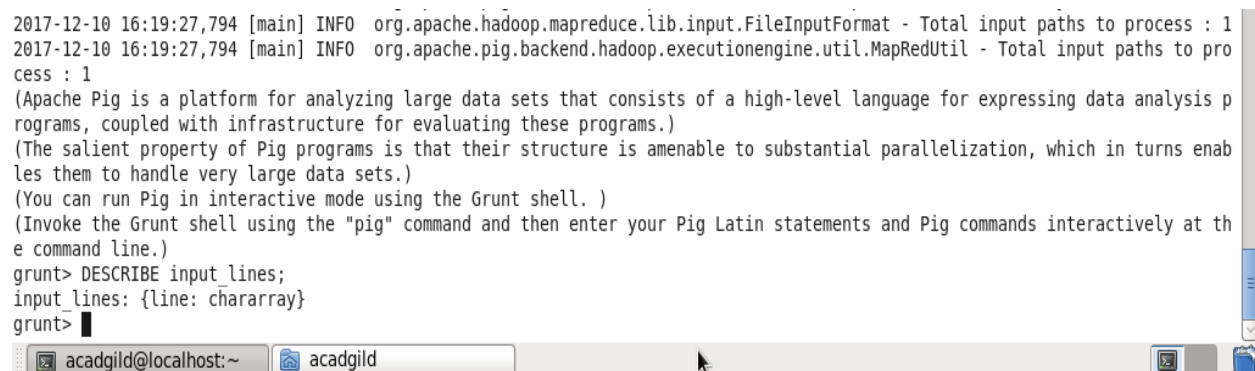


```
acadgild@localhost:~$ pwd
/home/acadgild
acadgild@localhost:~$ ls
assignment_stuff  eclipse               Music                pig_1512897086388.log  television.txt
derby.log         eclipse-jee-neon-M3-linux-gtk-x86_64.tar.gz Pictures            pig_1512898014050.log  Templates
Desktop          hadoop               pig_1512803760957.log pig_data              Videos
Documents        hive-site.xml        pig_1512805790632.log pig.txt               wordcount.pig
Downloads        metastore_db          pig_1512827788759.log Public                workspace
acadgild@localhost:~$ cat pig.txt
Apache Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs.
The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.
You can run Pig in interactive mode using the Grunt shell.
Invoke the Grunt shell using the "pig" command and then enter your Pig Latin statements and Pig commands interactively at the command line.[acadgild@localhost ~]$
```

Step 1: Load the text file into Pig environment.

input_lines = LOAD '/home/acadgild/pig.txt' AS (line:chararray);

Here 'input_lines' is a relation with an attribute 'line' of type character array that holds each line of the text file loaded.



```
2017-12-10 16:19:27,794 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 16:19:27,794 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(Apache Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs.)
(The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.)
(You can run Pig in interactive mode using the Grunt shell. )
(Invoke the Grunt shell using the "pig" command and then enter your Pig Latin statements and Pig commands interactively at the command line.)
grunt> DESCRIBE input_lines;
input_lines: {line: chararray}
grunt>
```

Step 2: Extract words from all the lines using Tokenize() function.

words = FOREACH input_lines GENERATE TOKENIZE(line) AS word;

Let's verify the result by using DUMP operation.

DUMP words;

```
2017-12-10 16:23:25,026 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 16:23:25,026 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
({(Apache),(Pig),(is),(a),(platform),(for),(analyzing),(large),(data),(sets),(that),(consists),(of),(a),(high-level),(language),(for),(expressing),(data),(analysis),(programs),(coupled),(with),(infrastructure),(for),(evaluating),(these),(programs.))})
({(The),(salient),(property),(of),(Pig),(programs),(is),(that),(their),(structure),(is),(amenable),(to),(substantial),(parallelization),(which),(in),(turns),(enables),(them),(to),(handle),(very),(large),(data),(sets.))})
({(You),(can),(run),(Pig),(in),(interactive),(mode),(using),(the),(Grunt),(shell.))})
({(Invoke),(the),(Grunt),(shell),(using),(the),(pig),(command),(and),(then),(enter),(your),(Pig),(Latin),(statements),(and),(Pig),(commands),(interactively),(at),(the),(command),(line.))})
grunt> DESCRIBE words;
words: {word: {tuple_of_tokens: (token: chararray)}}
```

We can see in the above output that each line has been converted to a bag holding every word in tuple form. We need to extract all these words from the bag. FLATTEN() function is used to unnest the tuples from a bag. Let's tweak the above query to do this:

words = FOREACH input_lines GENERATE FLATTEN(TOKENIZE(line)) AS word;

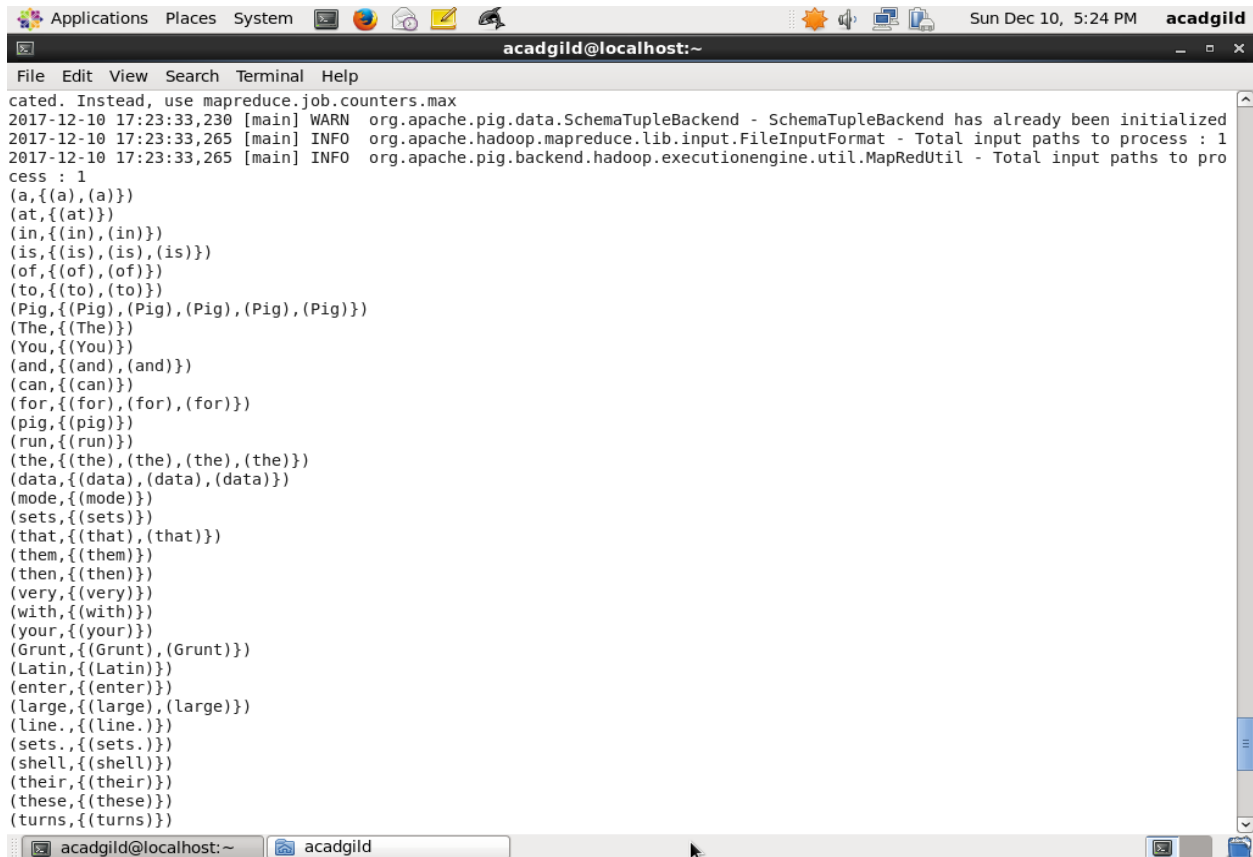
Let's verify the output:

```
Applications Places System acadgild@localhost:~
acadgild@localhost:~
File Edit View Search Terminal Help
(data)
(sets.)
(You)
(can)
(run)
(Pig)
(in)
(interactive)
(mode)
(using)
(the)
(Grunt)
(shell.)
(Invoke)
(the)
(Grunt)
(shell)
(using)
(the)
(pig)
(command)
(and)
(then)
(enter)
(your)
(Pig)
(Latin)
(statements)
(and)
(Pig)
(commands)
(interactively)
(at)
(the)
(command)
(line.)
grunt> DESCRIBE words;
words: {word: chararray}
grunt>
```

Step 3: Group the words by their occurrences.

grouped_words = GROUP words BY word;


Output: DUMP grouped_words;



```
acadmild@localhost:~  
File Edit View Search Terminal Help  
cated. Instead, use mapreduce.job.counters.max  
2017-12-10 17:23:33,230 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized  
2017-12-10 17:23:33,265 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1  
2017-12-10 17:23:33,265 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1  
(a,{(a),(a)})  
(at,{(at)})  
(in,{(in),(in)})  
(is,{(is),(is),(is)})  
(of,{(of),(of)})  
(to,{(to),(to)})  
(Pig,{(Pig),(Pig),(Pig),(Pig),(Pig)})  
(The,{(The)})  
(You,{(You)})  
(and,{(and),(and)})  
(can,{(can)})  
(for,{(for),(for),(for)})  
(pig,{(pig)})  
(run,{(run)})  
(the,{(the),(the),(the),(the)})  
(data,{(data),(data),(data)})  
(mode,{(mode)})  
(sets,{(sets)})  
(that,{(that),(that)})  
(them,{(them)})  
(then,{(then)})  
(very,{(very)})  
(with,{(with)})  
(your,{(your)})  
(Grunt,{(Grunt),(Grunt)})  
(Latin,{(Latin)})  
(enter,{(enter)})  
(large,{(large),(large)})  
(line,{(line)})  
(sets,{(sets)})  
(shell,{(shell)})  
(their,{(their)})  
(these,{(these)})  
(turns,{(turns)})
```

Let's look at the schema of above relation:

DESCRIBE grouped_words;



```
acadmild@localhost:~  
File Edit View Search Terminal Help  
(interactive,{(interactive)})  
(substantial,{(substantial)})  
(interactively,{(interactively)})  
(infrastructure,{(infrastructure)})  
(parallelization,{(parallelization)})  
grunt> DESCRIBE grouped_words;  
grouped words: {group: chararray,words: {(word: chararray)}}  
grunt>
```

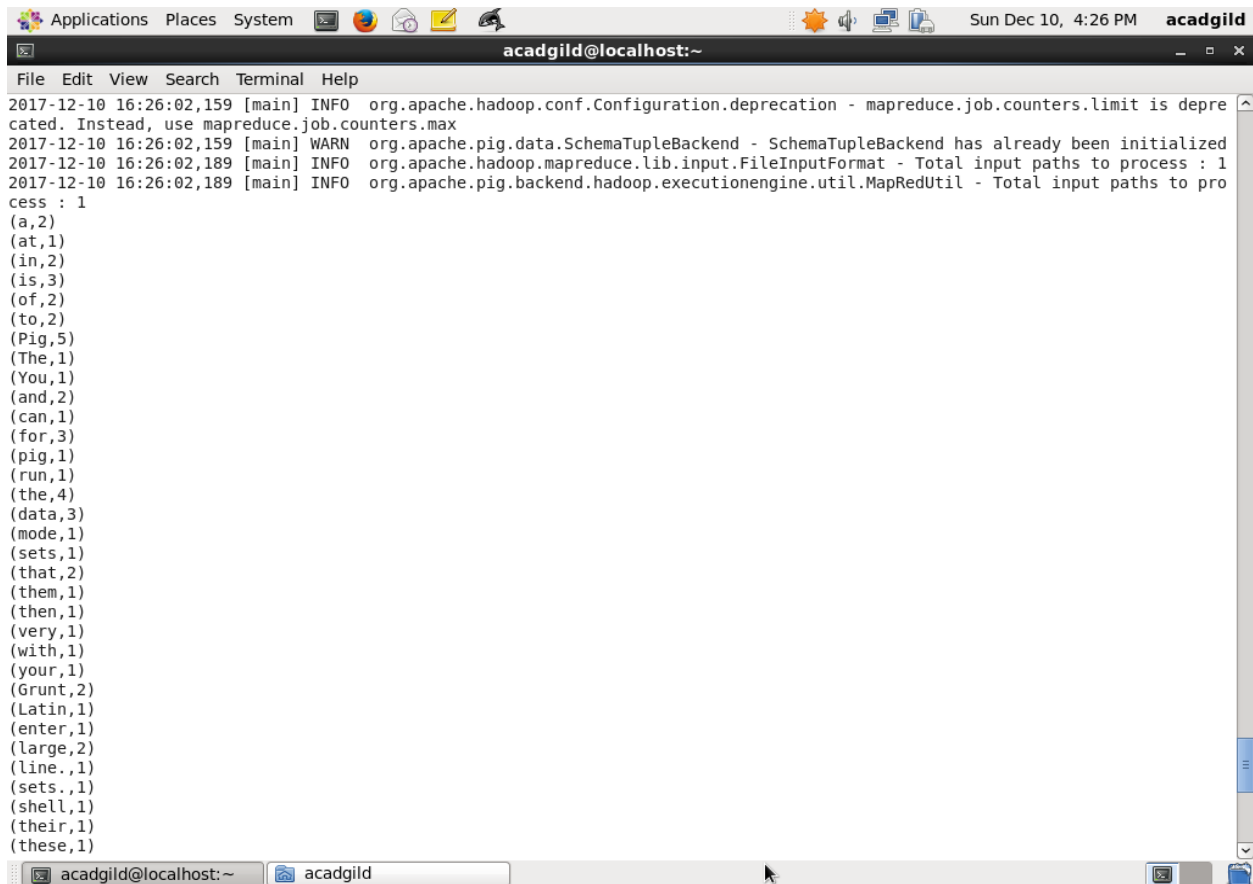
The relation 'words' has become part of 'grouped_words' schema and it contains a bag of all occurrences of each and every word as we saw in the result of 'DUMP grouped_words'.

Step 4: Get count of each word by applying COUNT() function.

```
count_of_word_occurences = FOREACH grouped_words GENERATE group,  
COUNT(words.word);
```

Let's verify the output of this query:

```
DUMP count_of_word_occurences;
```



```
File Edit View Search Terminal Help
2017-12-10 16:26:02,159 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapreduce.job.counters.limit is depre
cated. Instead, use mapreduce.job.counters.max
2017-12-10 16:26:02,159 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-12-10 16:26:02,189 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 16:26:02,189 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(a,2)
(at,1)
(in,2)
(is,3)
(of,2)
(to,2)
(Pig,5)
(The,1)
(You,1)
(and,2)
(can,1)
(for,3)
(pig,1)
(run,1)
(the,4)
(data,3)
(mode,1)
(sets,1)
(that,2)
(them,1)
(then,1)
(very,1)
(with,1)
(your,1)
(Grunt,2)
(Latin,1)
(enter,1)
(large,2)
(line.,1)
(sets.,1)
(shell,1)
(their,1)
(these,1)
```

This is the final outcome which we expected. It has grouped each word with its number of occurrences. Here is the script for word count combining all the steps performed above:

```
input_lines = LOAD '/home/acadgild/pig.txt' AS (line:chararray);
words = FOREACH input_lines GENERATE FLATTEN(TOKENIZE(line)) AS word;
grouped_words = GROUP words BY word;
count_of_word_occurences = FOREACH grouped_words GENERATE group,  
COUNT(words.word);
DUMP count_of_word_occurences;
```