

## Running Pig Latin Macro

---

edureka!

**edureka!**

© Brain4ce Education Solutions Pvt. Ltd.

## Table of Contents

Introduction on PIG Latin Macro.....	2
Running Sample PIG Latin Macro from Grunt:.....	4
Running PIG macro from PIG script.....	6

edureka!

## Introduction on PIG Latin Macro:

Macro is a name given to set of instructions to replace or run the instructions automatically by macro.

- A macro is a name given to a block of the code which can be substituted where the code
- Snippet is to be used for more than once.
- The speed of the execution of the program is the major advantage of using a macro.
- It saves a lot of time that is spent by the compiler for invoking / calling the functions.
- It reduces the length of the program.

Pig Latin supports the definition, expansion, and import of macros.

\* Macros are supported in PIG Latin from version 0.9

To run the PIG Latin we have to define the Macro.

### Syntax For Pig Latin Macro:

```
DEFINE macro_name (parameters) RETURNS {void | alias } { pig_latin_fragment };
```

### Terms:

**DEFINE** and **RETURNS** are keywords.

### macro\_name:

- ★ The name used to represent the group of PIG Latin instructions.
- ★ Macro names are combination of Characters, Numeric and Underscore symbol.
- ★ First letter of Macro name should be starts with Character and followed by group of Characters, Numbers and Underscore symbol.

### Parameters:

(Optional) A comma-separated list of one or more parameters.

Pig macros support four types of parameters:

1. alias (IDENTIFIER)
2. integer
3. float
4. string literal (quoted string)

### Void:

If the macro has no return any alias, then void must be specified.

### Alias:

(Optional) A comma-separated list of one or more return aliases (Pig relations) that are referenced in the Pig Latin fragment. The alias must exist in the macro in the form \$<alias>.

If the macro has no return alias, then void must be specified.

### Pig\_latin\_fragment:

One or more Pig Latin statements, enclosed in curly brackets.

### Macro Definition

We can define the PIG Latin macro in command line (In grunt shell) or importing PIG macro script.

PIG Macro definition should be prior to first use.

Recursive references of PIG macros are not allowed.

### Restrictions in PIG Macro Definition:

- Macros are not allowed inside a FOREACH nested block.
- The REGISTER statement is not supported in Macros.
- The shell commands (used with Grunt) are not supported.

### Running Sample PIG Latin Macro from Grunt:

#### Use Case:

Arranging ranks in Ascending order using PIG macro.

The data set contain the username and Ranks for the user in Random order, We have to arrange in ascending order according to the Ranks using PIG macros.

Step-1: We should define the Macro before use it in PIG statements.

```
grunt> DEFINE macro_pig(A, sortkey) RETURNS C {  
>>   $C = ORDER $A BY $sortkey;  
>> };  
grunt> █
```

Step-2: Loading Data set into PIG.

```
grunt> X = load 'Desktop/input' using PigStorage(',') as (name:chararray,ranking:int);  
fs.default.name is deprecated. Instead, use fs.defaultFS  
io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum  
grunt> █
```

Check the data is loaded correctly or not by using DUMP.

```
grunt> dump X;
io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
fs.default.name is deprecated. Instead, use fs.defaultFS
Total input paths to process : 1
Saved output of task 'attempt_0001_m_000001_1' to file:/tmp/temp2100915468/tmp1125749312/_temporary/0/task_0001_m_000001
Total input paths to process : 1
(xyz,3)
(abc,2)
(hjk,4)
(lmn,5)
(opq,1)
grunt>
```

**Step-3:** Running PIG macro to arrange the Ranks in Order.  
Pass the PIG schema and Column to the Macro.

```
grunt> Y = macro_pig(X,ranking);
grunt>
```

Output of the PIG Macro by running DUMP command.

```
grunt> dump Y;
io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
fs.default.name is deprecated. Instead, use fs.defaultFS
session.id is deprecated. Instead, use dfs.metrics.session-id
Initializing JVM Metrics with processName=JobTracker, sessionId=
mapred.job.reduce.markreset.buffer.percent is deprecated. Instead, use mapreduce.reduce.markreset.buffer.percent
mapred.output.compress is deprecated. Instead, use mapreduce.output.fileoutputformat.compress
mapred.job.tracker.http.address is deprecated. Instead, use mapreduce.jobtracker.http.address
Cannot initialize JVM Metrics with processName=JobTracker, sessionId= - already initialized
No job jar file set. User classes may not be found. See Job or Job#setJar(String).
Total input paths to process : 1
number of splits:1
```

```
(opq,1)
(abc,2)
(xyz,3)
(hjk,4)
(lmn,5)
grunt>
```

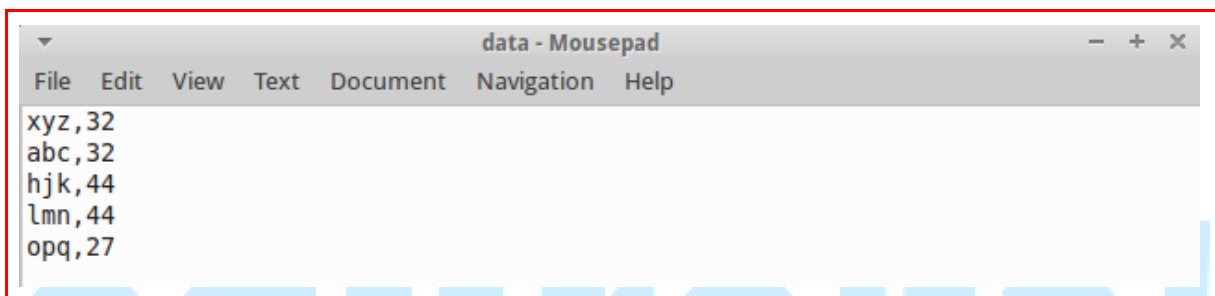
### Running PIG macro from PIG script:

1. Write the PIG macro in a file and save the file with any name extension pig.
2. Import the Pig Macro from the grunt shell.
3. Use the macro in PIG Latin statements.

**Use Case:** Counting the uses according to the age.

Input file contain the users and age of the users.

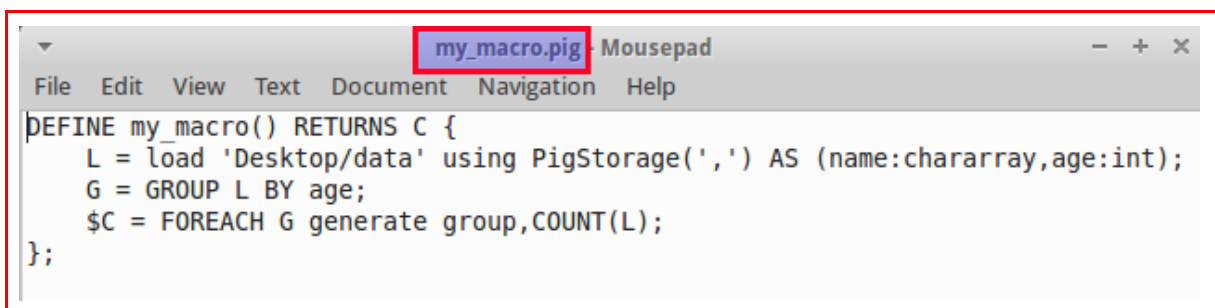
Input data set:



```
File Edit View Text Document Navigation Help
xyz,32
abc,32
hjk,44
lmn,44
opq,27
```

**Step-1:** Write the PIG macro in a file and save the file with extension .pig

Macro script file name **my\_macro.pig**



```
File Edit View Text Document Navigation Help
DEFINE my_macro() RETURNS C {
  L = load 'Desktop/data' using PigStorage(',') AS (name:chararray,age:int);
  G = GROUP L BY age;
  $C = FOREACH G generate group,COUNT(L);
};
```

**Step-2:** Import the micro script into PIG shell.

**The macro path should be in single quotation marks.**

```
grunt> import 'Desktop/my_macro.pig';  
grunt> █
```

**Step-3:** Running micro by giving by using micro name.

In this example Macro name is **my\_micro()**.

```
grunt> counting = my_macro();  
fs.default.name is deprecated. Instead, use fs.defaultFS  
io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum  
grunt> █
```

Use DUMP command to check the output of the macro.

```
grunt> dump counting;  
io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum  
fs.default.name is deprecated. Instead, use fs.defaultFS  
Cannot initialize JVM Metrics with processName=JobTracker, sessionId= - already initialized  
Cannot initialize JVM Metrics with processName=JobTracker, sessionId= - already initialized  
No job jar file set. User classes may not be found. See Job or Job#setJar(String).  
Total input paths to process : 1  
number of splits:1  
.....
```

```
(27,1)  
(32,2)  
(44,2)  
grunt> █
```

**Reference Links:**

<http://hortonworks.com/blog/new-apache-pig-features-part-1-macro/>