

## Apache Pig Tutorial

### Pig Configuration to avoid displaying logs

We will modify the logging properties of Pig and set it to a mode where only fatal error are displayed. For that we will have to make changes in log4j.properties file available in the configuration directory of Pig.

The screenshot shows two windows. The top window is a terminal with the following commands and output:

```

[jayantmohite@localhost ~]$ ls /etc/pig
conf  conf.dist
[jayantmohite@localhost ~]$ ls /etc/pig/conf
build.properties  log4j.properties  pig.properties
[jayantmohite@localhost ~]$ sudo gedit /etc/pig/conf/log4j.properties
[sudo] password for jayantmohite:

```

The bottom window is a text editor showing the contents of `log4j.properties`. The following lines are highlighted with a red box:

```

# ***** Set root logger level to DEBUG and its only appender to A.
log4j.logger.org.apache.pig=fatal, A

```

The rest of the file contains license information and configuration for the appender A:

```

# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements. See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License. You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the license.
# ***** A is set to be a ConsoleAppender.
log4j.appender.A=org.apache.log4j.ConsoleAppender
# ***** A uses PatternLayout.
log4j.appender.A.layout=org.apache.log4j.PatternLayout
log4j.appender.A.layout.ConversionPattern=%-4r [%t] %-5p %c %x - %m%n

```

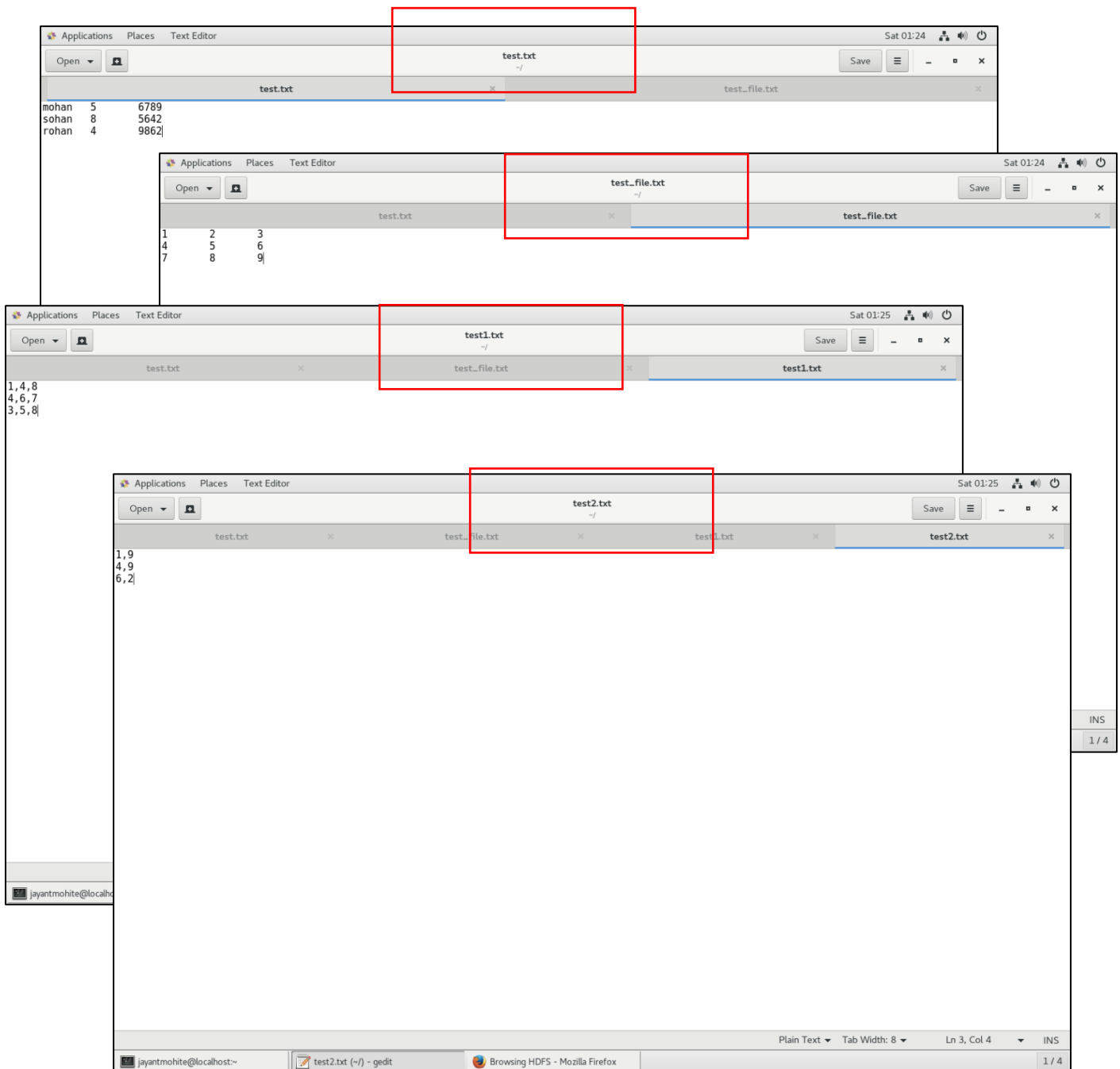
Steps:

Open the pig log properties by using the command  
`sudo gedit /etc/pig/conf/log4j.properties`  
 and the to it a line which states  
`log4j.logger.org.pache.pig=fatal, A`

Author: Jayant Mohite

For this lab session we will be using Pig in local mode.

Sample files used:



Author: Jayant Mohite

### Step Description:

In this step we will load the input file test.txt located at /home/jayantmohite/ with a schema of 3 columns namely name, emp\_num and salary.

In this case we are not specifying any delimiter as the delimiter used in the file test.txt is tab and in Pig tab (\t) is default delimiter.

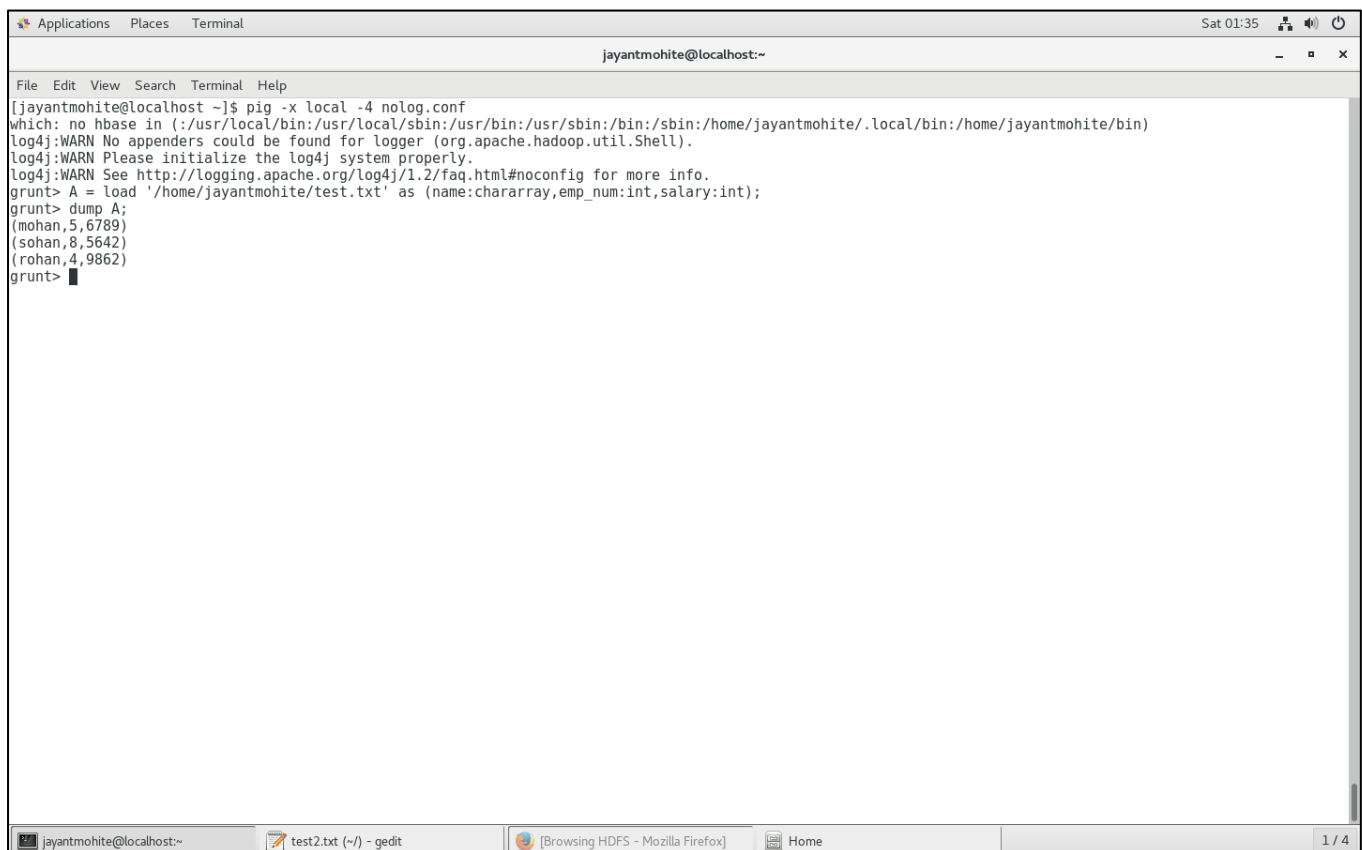
### Commands:

```
grunt> A = load '/home/jayantmohite/test.txt' as (name:chararray, emp_num:int, salary:int);
```

(In this command, A is the bag that we are creating)

```
grunt> dump A;
```

### Step Visualization:

A screenshot of a terminal window titled 'Applications Places Terminal' with a status bar showing 'Sat 01:35'. The terminal prompt is 'jayantmohite@localhost:~'. The user enters the command '[jayantmohite@localhost ~]\$ pig -x local -4 nolog.conf'. The terminal shows several warning messages from log4j and the Pig command prompt 'grunt>'. The user then enters 'A = load '/home/jayantmohite/test.txt' as (name:chararray, emp\_num:int, salary:int);' followed by 'grunt> dump A;'. The output shows three rows of data: (mohan,5,6789), (sohan,8,5642), and (rohan,4,9862). The terminal window has a menu bar with 'File Edit View Search Terminal Help' and a taskbar at the bottom with icons for 'jayantmohite@localhost:~', 'test2.txt (~/) - gedit', '[Browsing HDFS - Mozilla Firefox]', and 'Home'. The bottom right corner shows '1 / 4'.

Do note that all labs in this chapter are performed in the Pig Local Mode.

Author: Jayant Mohite

## Step Description

In this step we will create a bag B couple of time with the same input file name test\_file.txt. The difference in both these execution will be that first execution, we will not specify any schema but in second execution we will specify the schema.

In case where we don't specify the schema, we can reference columns by their positional parameters like column 1 is referenced as \$0; column 2 is referenced as \$1 and so on.

Commands:

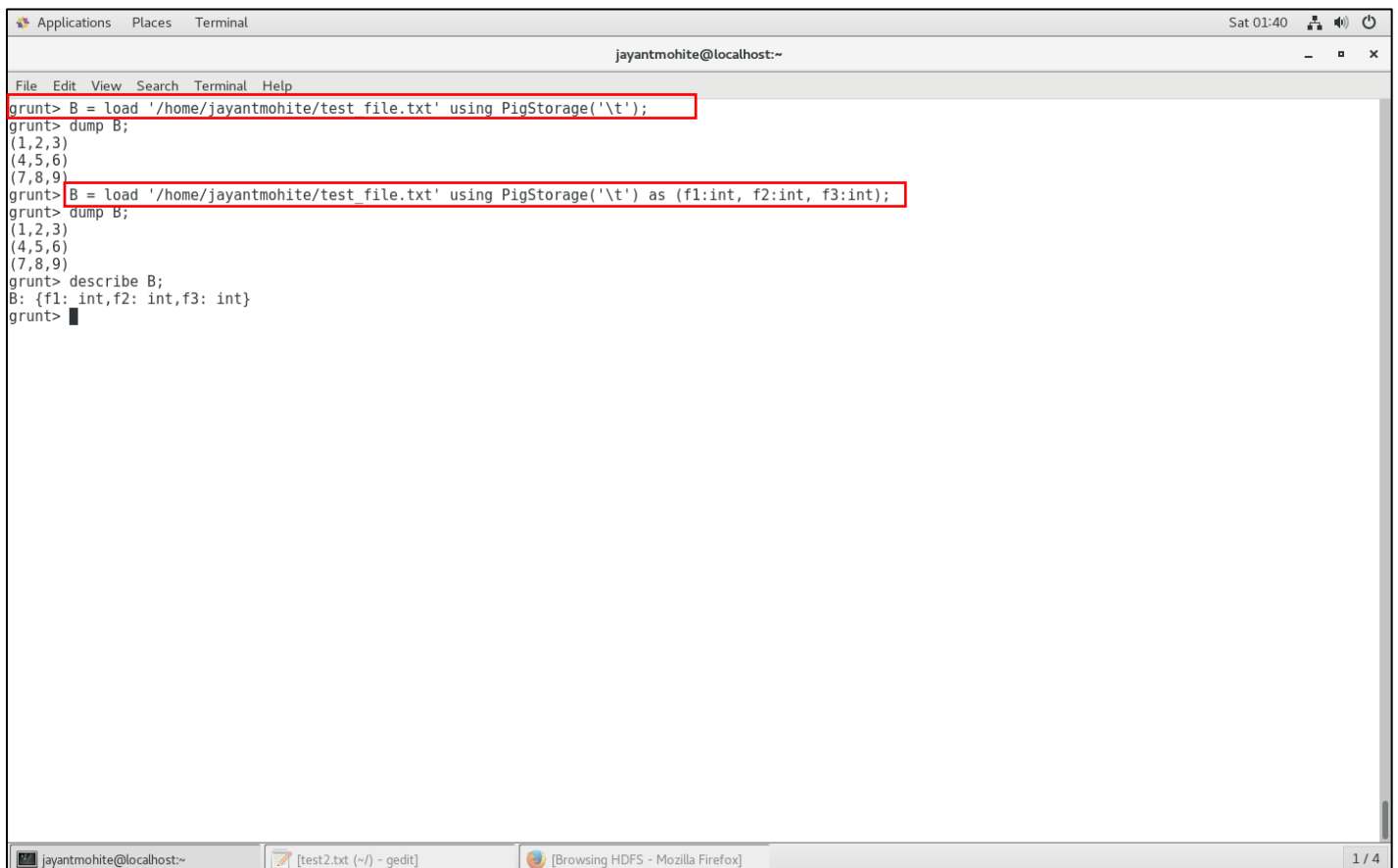
```
grunt> B = load '/home/jayantmohite/test_file.txt' using PigStorage('\t');
```

(In this command, though not required but we have specified the delimiter used in the file by using the function PigStorage. We are not specifying any schema in this command.)

```
grunt> B = load '/home/jayantmohite/test_file.txt' using PigStorage('\t') as (f1:int, f2:int, f3:int);
```

(In this command, we are specifying the schema for the bag which can be checked by the Describe command.)

Step Visualization:



```
Applications  Places  Terminal  Sat 01:40
jayantmohite@localhost:~
File Edit View Search Terminal Help
grunt> B = load '/home/jayantmohite/test file.txt' using PigStorage('\t');
grunt> dump B;
(1,2,3)
(4,5,6)
(7,8,9)
grunt> B = load '/home/jayantmohite/test file.txt' using PigStorage('\t') as (f1:int, f2:int, f3:int);
grunt> dump B;
(1,2,3)
(4,5,6)
(7,8,9)
grunt> describe B;
B: {f1: int,f2: int,f3: int}
grunt>
```

Author: Jayant Mohite

## Step Description:

In this step we will have a look at some operators in Pig like filter, foreach and generate.

### Commands:

```
grunt> B = load '/home/jayantmohite/test_file.txt' using PigStorage('\t') as (f1:int, f2:int, f3:int);
```

```
grunt> X = filter B by f3 == 3;
```

(In this command we are creating a bag by name X which will contains records from the bag B where the column f3 has a value equal to 3)

```
grunt> Z = filter B by (f1 == 8) or (not(f2 + f3 < f1));
```

```
grunt> C = foreach B generate f1 as col1;
```

(In this command we are creating a bag by name C, which will have data of all records only from column f1 of bag B. This new bag C will have a schema where the only column it has will have a name as col1)

## Step Visualization:

```
File Edit View Search Terminal Help
jayantmohite@localhost:~
grunt> B = load '/home/jayantmohite/test_file.txt' using PigStorage('\t') as (f1:int, f2:int, f3:int);
grunt> dump B;
(1,2,3)
(4,5,6)
(7,8,9)
grunt> X = filter B by f3 == 3;
grunt> dump X;
(1,2,3)
grunt> Y = filter B by (f1 == 8) or (not(f2+f3>f1));
grunt> dump Y;
(1,2,3)
grunt> Z = filter B by (f1 == 8) or (not(f2+f3<f1));
grunt> dump Z;
(1,2,3)
(4,5,6)
(7,8,9)
grunt> C = foreach B generate f1 as col1;
grunt> dump C;
(1)
(4)
(7)
grunt> describe C;
C: {col1: int}
grunt>
```

## Step Description:

In this step we will see how Group By works in Pig

### Commands:

```
grunt> A = load '/home/jayantmohite/myemp.txt' using PigStorage(',') as (id:int,
name:chararray, salary:int, country:chararray);
```

(for this you can use the same input file that we have used in Hive Partitions)

```
grunt> B = group A by country;
```

```
grunt> describe B;
```

(this command will show the schema of the bag B which contains records from bag A grouped by country. In this bag, every tuple will have 2 atoms. The 1<sup>st</sup> atom will be the key group and the 2<sup>nd</sup> atom will be the elements of bag A that belong to this group.)

## Step Visualization:

```

Applications  Places  Terminal
jayantmohite@localhost:~
File Edit View Search Terminal Help
grunt> A = load '/home/jayantmohite/myemp.txt' using PigStorage(',') as (id:int,name:chararray,salary:int,country:chararray);
grunt> dump A;
(1,abc,100,india)
(2,bcd,200,us)
(3,cde,300,india)
(4,def,400,india)
(5,efg,500,us)
(6, fgh,700,aus)
grunt> B = group A by country;
grunt> describe B;
B: {group: chararray,A: {(id: int,name: chararray,salary: int,country: chararray)}}
grunt> dump B;
(us,{(2,bcd,200,us),(5,efg,500,us)})
(aus,{(6, fgh,700,aus)})
(india,{(1,abc,100,india),(3,cde,300,india),(4,def,400,india)})
grunt>

```

jayantmohite@localhost:~ myemp.txt (~) - gedit [Browsing HDFS - Mozilla Firefox] Home 1 / 4

### Step Description:

In this step we will have a demonstration of the join and union commands of pig wherein we will be working with two bags in the same command.

### Command:

```
grunt> A = load '/home/jayantmohite/test1.txt' using PigStorage(',') as (a1:int, a2:int, a3:int);
```

```
grunt> B = load '/home/jayantmohite/test2.txt' using PigStorage(',') as (b1:int, b2:int);
```

```
grunt> X = join A by a1, B by b1;
```

(the data that we are loading in bag X can be visualized by the SQL query as

```
select * from A,B where A.a1 = B.b1)
```

```
grunt> Y = union A, B;
```

### Step Visualization:

```
File Edit View Search Terminal Help
jayantmohite@localhost:~
grunt> A = load '/home/jayantmohite/test1.txt' using PigStorage(',') as (a1:int, a2:int, a3:int);
grunt> dump A;
(1,4,8)
(4,6,7)
(3,5,8)
grunt> B = load '/home/jayantmohite/test2.txt' using PigStorage(',') as (b1:int, b2:int);
grunt> dump B;
(1,9)
(4,9)
(6,2)
grunt> X = join A by a1, B by b1;
grunt> dump X;
(1,4,8,1,9)
(4,6,7,4,9)
grunt> Y = union A,B;
grunt> dump Y;
(1,4,8)
(4,6,7)
(3,5,8)
(1,9)
(4,9)
(6,2)
grunt>
```

### Step Description:

In this step we will have a demonstration of the Sample command. Lets say we have an input data with 10 million records and we need to run some pig analytics on to it. And we are not sure if our program will yield the correct result. In order to verify this, it would be great if we could test our program on a subset of the actual data.

This is exactly what sample command does.

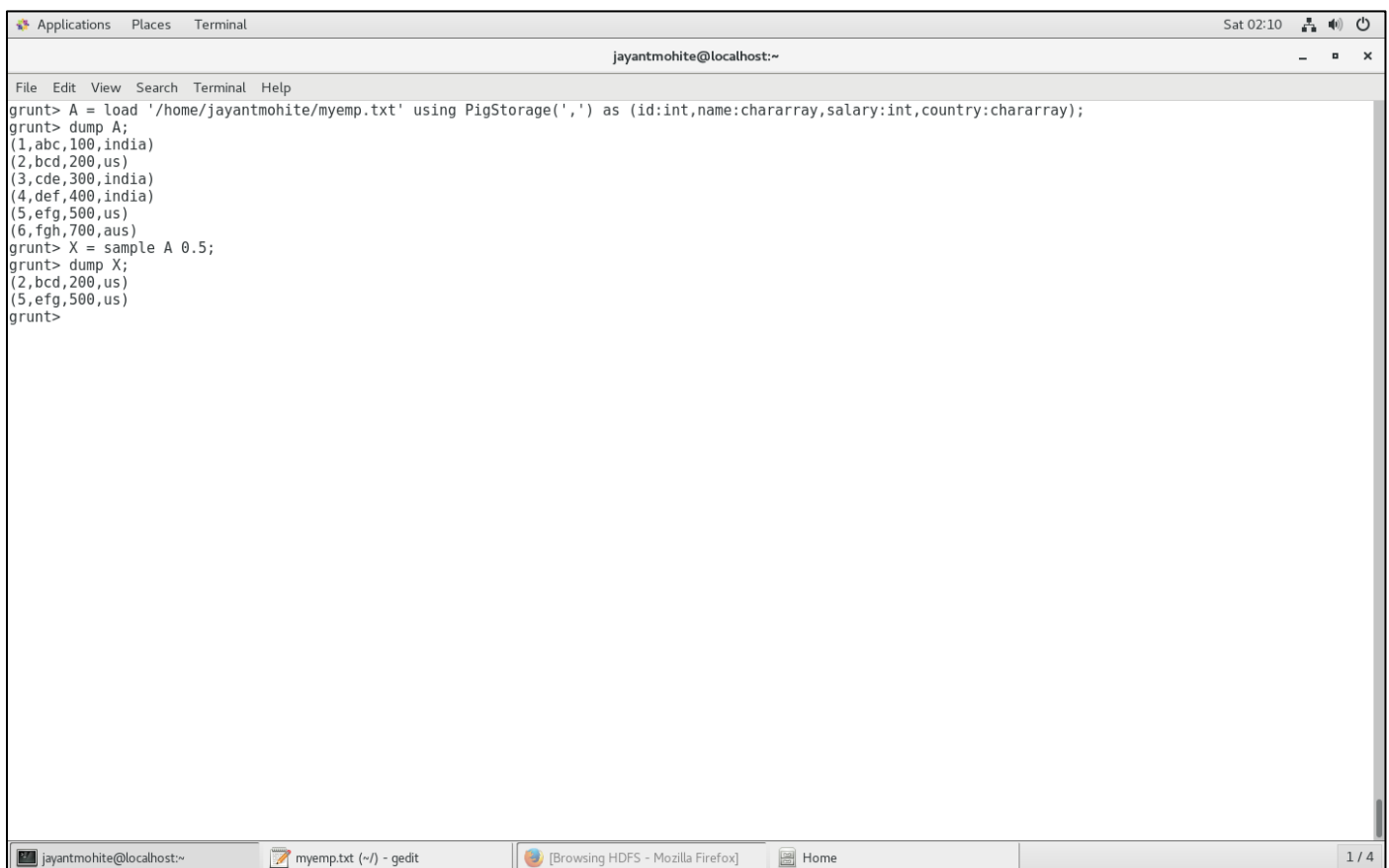
### Command:

```
grunt> A = load '/home/jayantmohite/myemp.txt' using PigStorage(',') as (id:int,
name:chararray, salary:int, country: chararray);
```

```
grunt> X = sample A 0.5;
```

(in here 0.5 is like 50% of the total input data)

### Step Visualization:

A screenshot of a Linux terminal window. The title bar shows 'Applications Places Terminal' and the date 'Sat 02:10'. The terminal prompt is 'jayantmohite@localhost:~'. The user enters the command 'grunt> A = load '/home/jayantmohite/myemp.txt' using PigStorage(',') as (id:int,name:chararray,salary:int,country:chararray);'. The prompt changes to 'grunt>'. The user enters 'dump A;'. The output shows six rows of data: (1,abc,100,india), (2,bcd,200,us), (3,cde,300,india), (4,def,400,india), (5,efg,500,us), and (6, fgh,700,aus). The user then enters 'grunt> X = sample A 0.5;'. The prompt changes to 'grunt>'. The user enters 'dump X;'. The output shows two rows of data: (2,bcd,200,us) and (5,efg,500,us). The user enters 'grunt>' again. The terminal window has a menu bar with 'File Edit View Search Terminal Help'. The bottom status bar shows 'jayantmohite@localhost:~', 'myemp.txt (~/) - gedit', '[Browsing HDFS - Mozilla Firefox]', 'Home', and '1 / 4'.



### Step Description:

In this step we will have a look at the split command which allows you to create multiple bags from an existing bag in the same command based on some criteria

### Commands:

```
grunt> A = load '/home/jayantmohite/test1.txt' using PigStorage(',') as (f1:int, f2:int, f3:int);
```

```
grunt> split A into x if f1 < 7, y if f2 == 5, z if (f3 < 6 or f3 > 6);
```

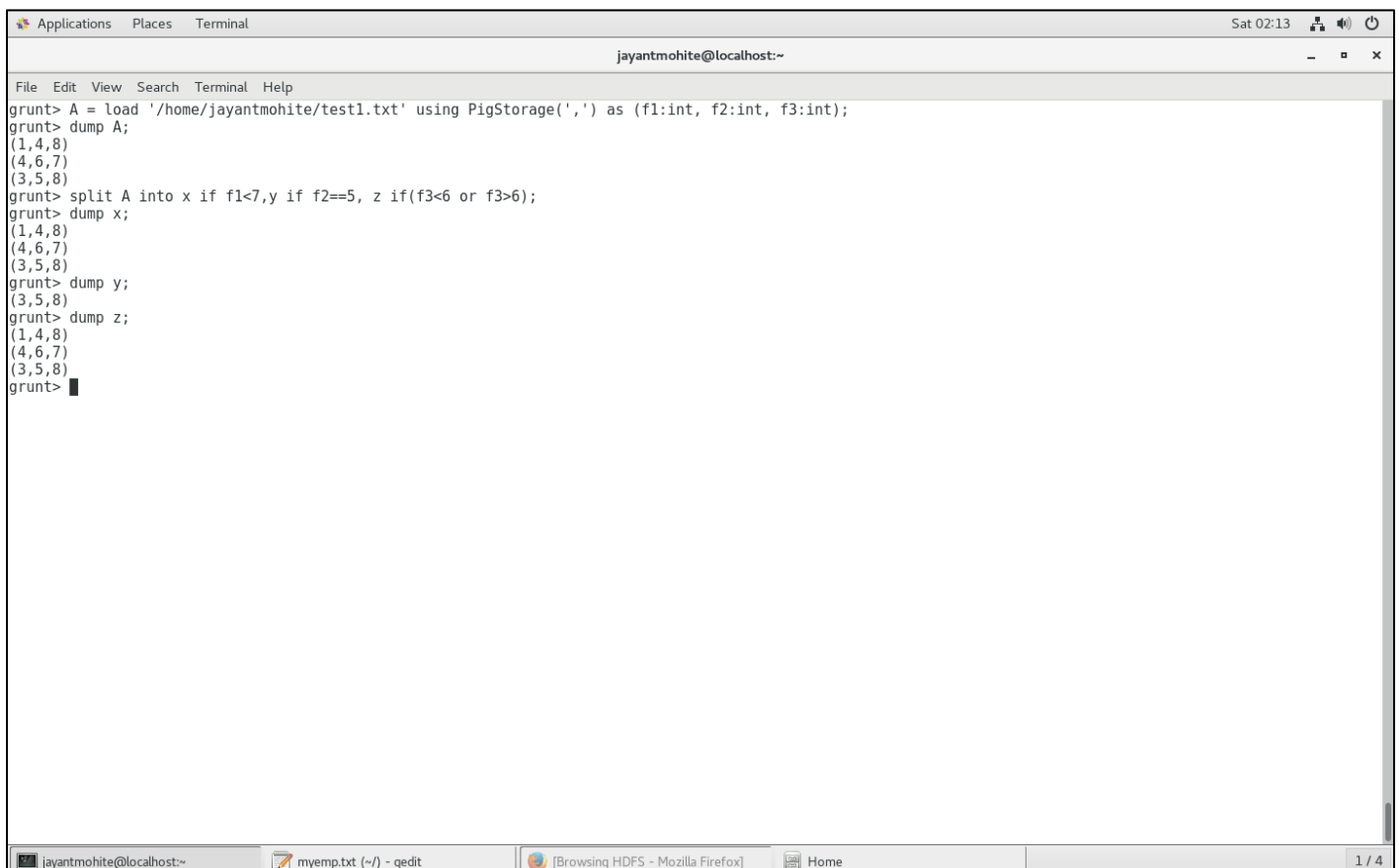
(this command will create the following 3 bags from data in bag A

bag x = all records from A that have the value of 1<sup>st</sup> column less than 7

bag y = all records from A that have value of 2<sup>nd</sup> column equal to 5

bag z = all records from A that have value of 3<sup>rd</sup> column not equal to 6)

### Step Visualization:



The screenshot shows a terminal window titled 'Applications Places Terminal' with the user 'jayantmohite@localhost'. The terminal displays the following commands and output:

```
File Edit View Search Terminal Help
grunt> A = load '/home/jayantmohite/test1.txt' using PigStorage(',') as (f1:int, f2:int, f3:int);
grunt> dump A;
(1,4,8)
(4,6,7)
(3,5,8)
grunt> split A into x if f1<7,y if f2==5, z if(f3<6 or f3>6);
grunt> dump x;
(1,4,8)
(4,6,7)
(3,5,8)
grunt> dump y;
(3,5,8)
grunt> dump z;
(1,4,8)
(4,6,7)
(3,5,8)
grunt>
```

The terminal window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The status bar at the bottom shows the user 'jayantmohite@localhost', the file 'myemp.txt (~) - gedit', the browser '[Browsing HDFS - Mozilla Firefox]', and the 'Home' button. The page number '1 / 4' is visible in the bottom right corner.

## Step Description:

In this step we will demonstrate some aggregate functions like Min, Max and Count

## Commands:

```
grunt> A = load '/home/jayantmohite/myemp.txt' using PigStorage(',') as (id:int,
name:chararray, salary:int, country:chararray);
```

```
grunt> B = group A by country;
```

```
grunt> C = foreach B generate A.country, AVG(A.salary);
```

```
grunt> D = foreach B generate group, MAX(A.salary);
```

```
grunt> E = foreach B generate group, MIN(A.salary);
```

```
grunt> F = foreach B generate COUNT(A);
```

## Step Visualization:

```
File Edit View Search Terminal Help
jayantmohite@localhost:~
grunt> A = load '/home/jayantmohite/myemp.txt' using PigStorage(',') as (id:int,name:chararray,salary:int,country:chararray);
grunt> dump A;
(1,abc,100,india)
(2,bcd,200,us)
(3,cde,300,india)
(4,def,400,india)
(5,efg,500,us)
(6,fgh,700,aus)
grunt> B = group A by country;
grunt> dump B;
(us,{(2,bcd,200,us),(5,efg,500,us)})
(aus,{(6,fgh,700,aus)})
(india,{(1,abc,100,india),(3,cde,300,india),(4,def,400,india)})
grunt> C = foreach B generate A.country, AVG(A.salary)
>> ;
grunt> dump C;
(({us},{(us)}),350.0)
(({aus},{(aus)}),700.0)
(({india},{(india)}),266.6666666666667)
grunt> D = foreach B generate group, MAX(A.salary);
grunt> dump D;
(us,500)
(aus,700)
(india,400)
grunt> E = foreach B generate group, MIN(A.salary);
grunt> dump E;
(us,200)
(aus,700)
(india,100)
grunt> F = foreach B generate COUNT(A);
grunt> dump F;
(2)
(1)
(3)
grunt>
```

That is all about the Pig Latin Commands. Explore more in the next chapters.

Author: Jayant Mohite