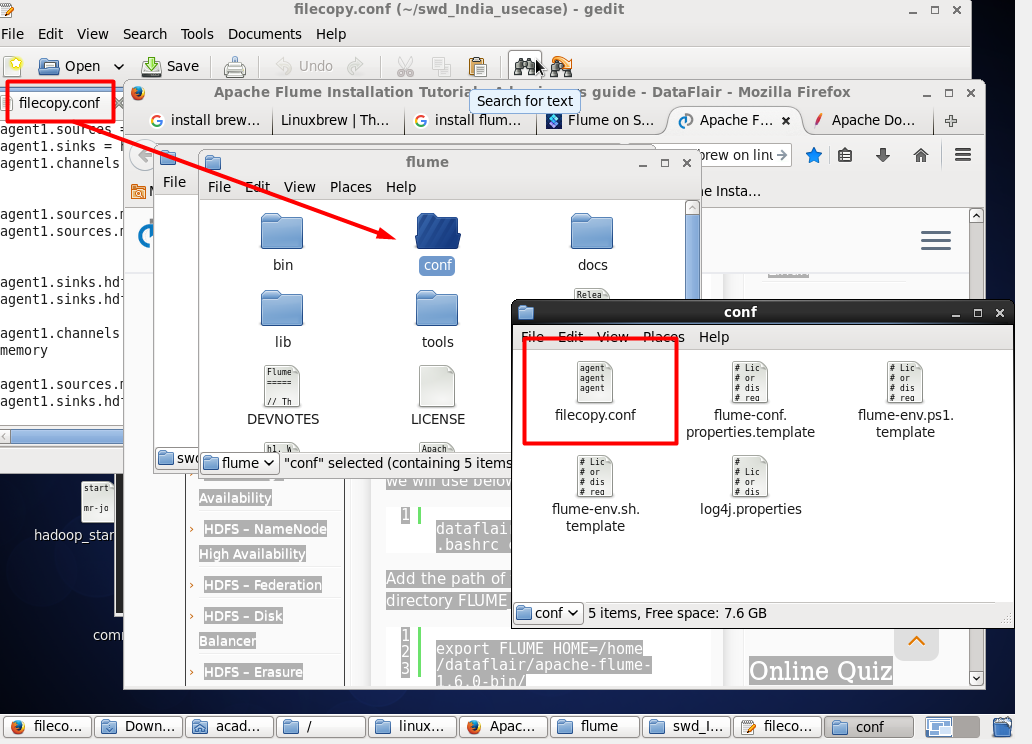


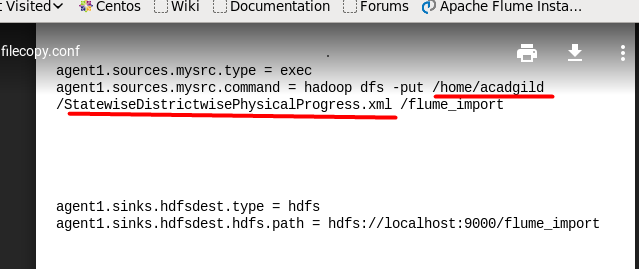
First of all we need to move the config file (filecopy.conf) to special configuration folder of flume

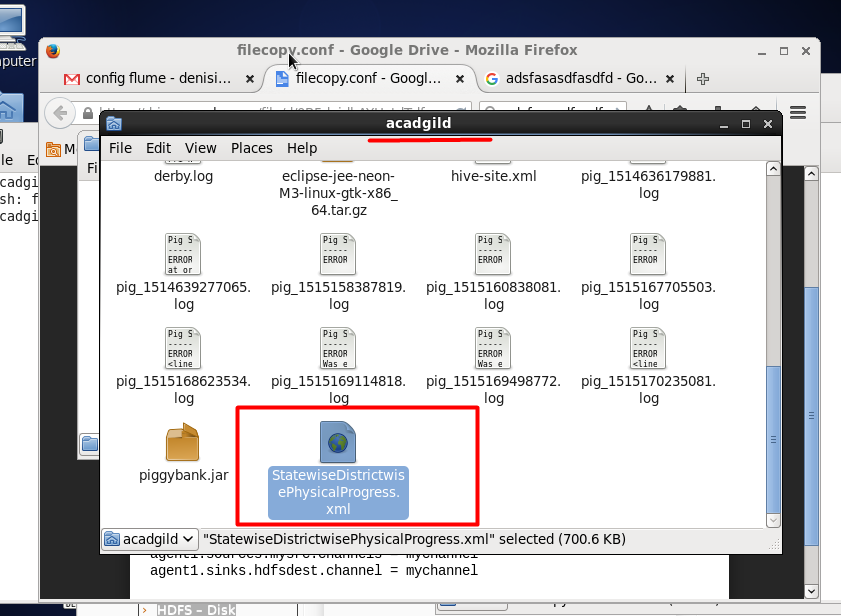


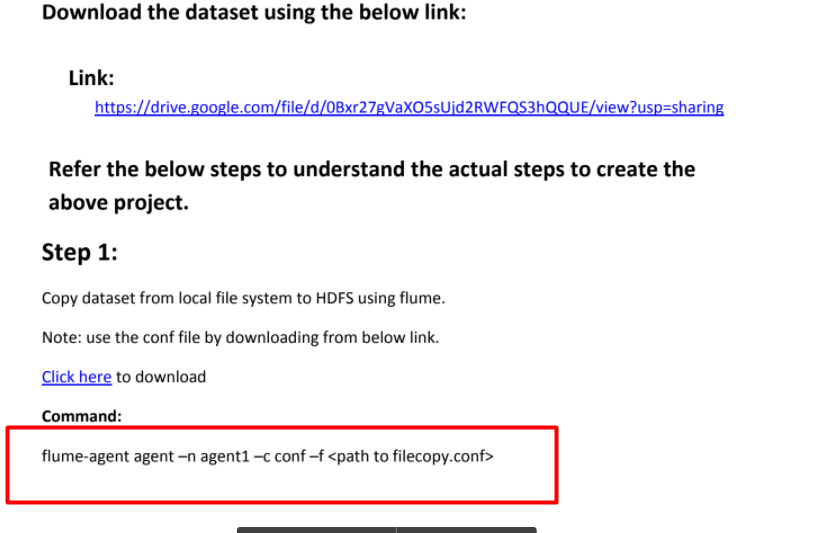
**Now we need to move our dataset to the hdfs**

For this purpose lets use the command that given to us at the project assignment description

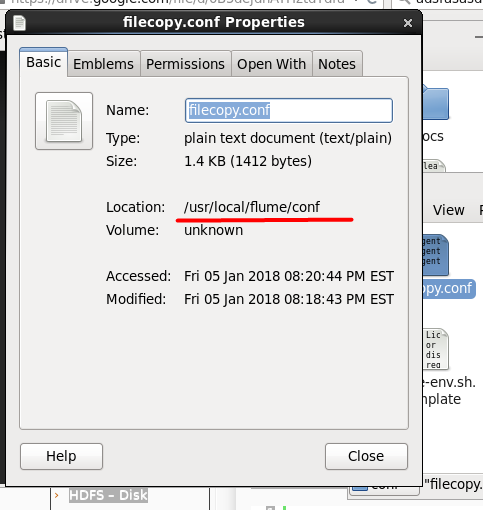
Also we need to move dataset under the path that written in the config file







And just add a path to the config file which we just moved

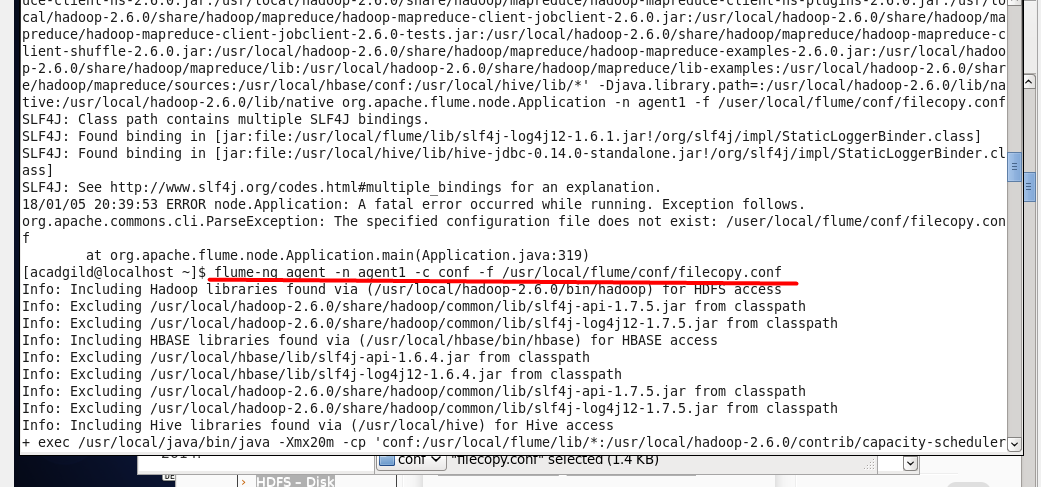


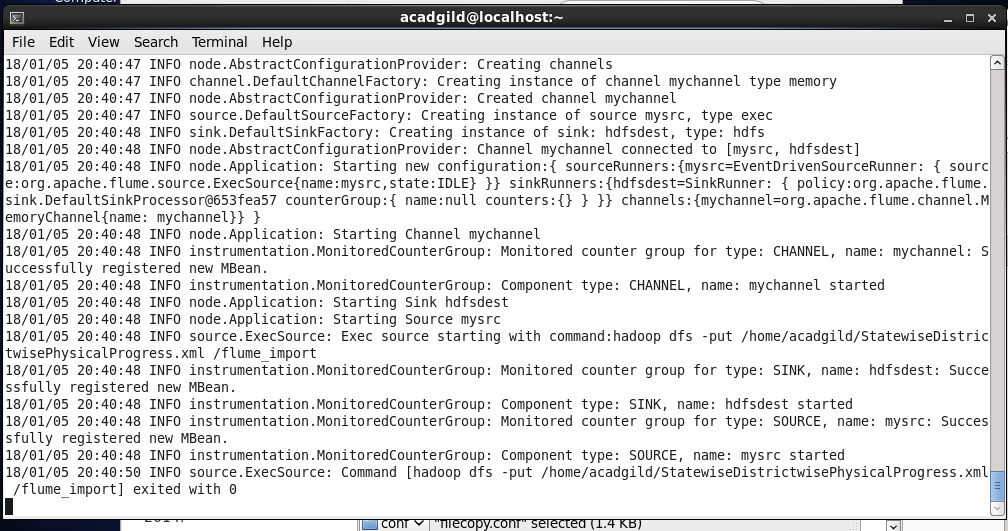
The full command will be look like :

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

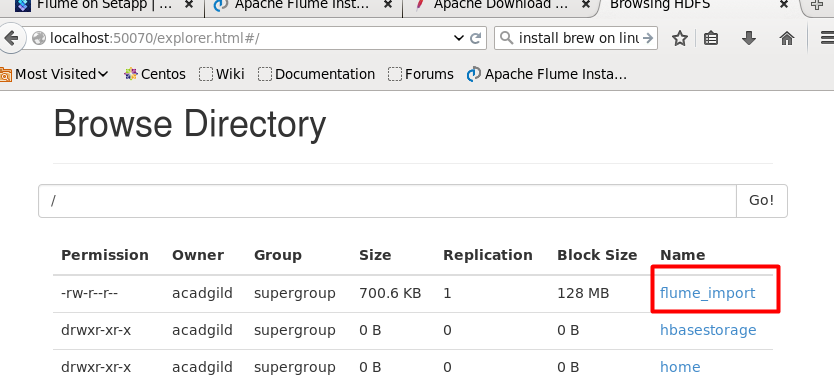
flume-ng agent -n agent1 -c conf -f /usr/local/flume/conf/filecopy.conf

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

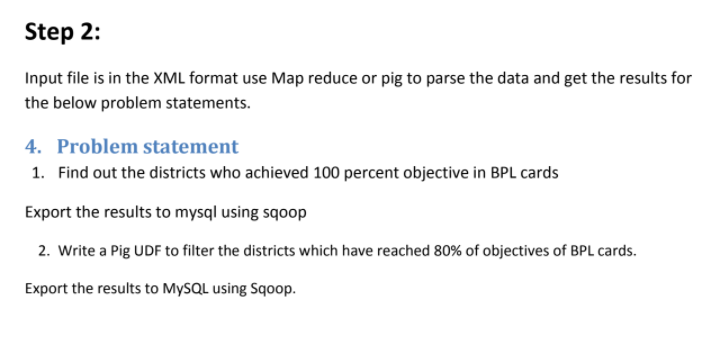




Let’s check that data indeed moved



Now we can move to step two

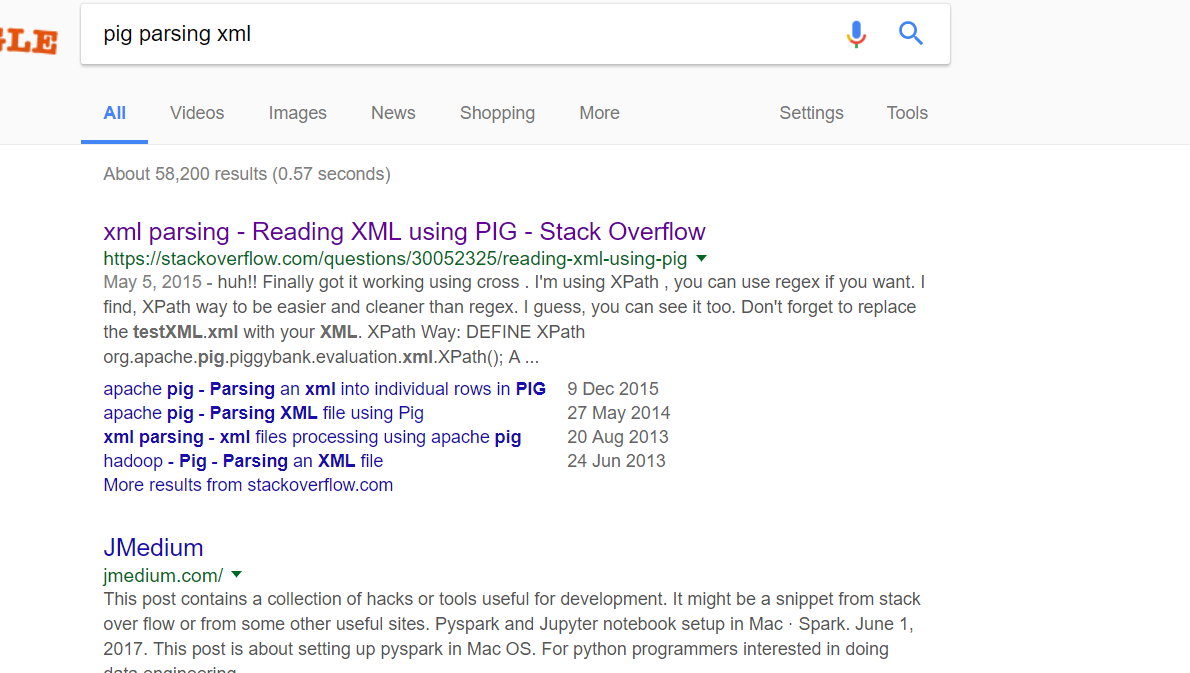


First of all let's parse and dump result in the terminal and than store it and move to final destination.

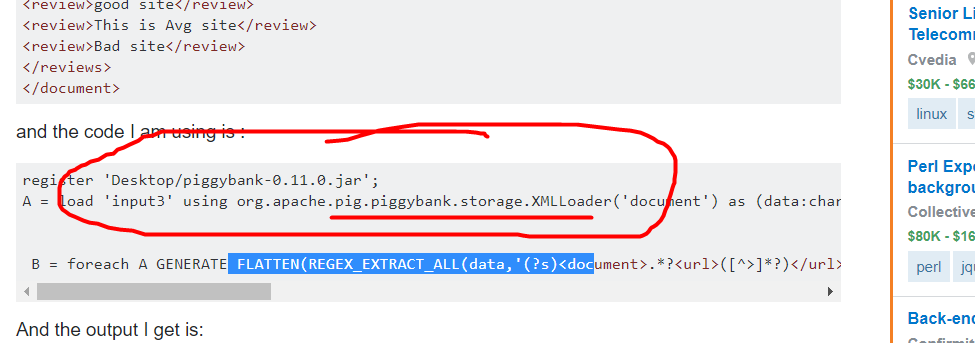
Let’s focus on this part of the project:



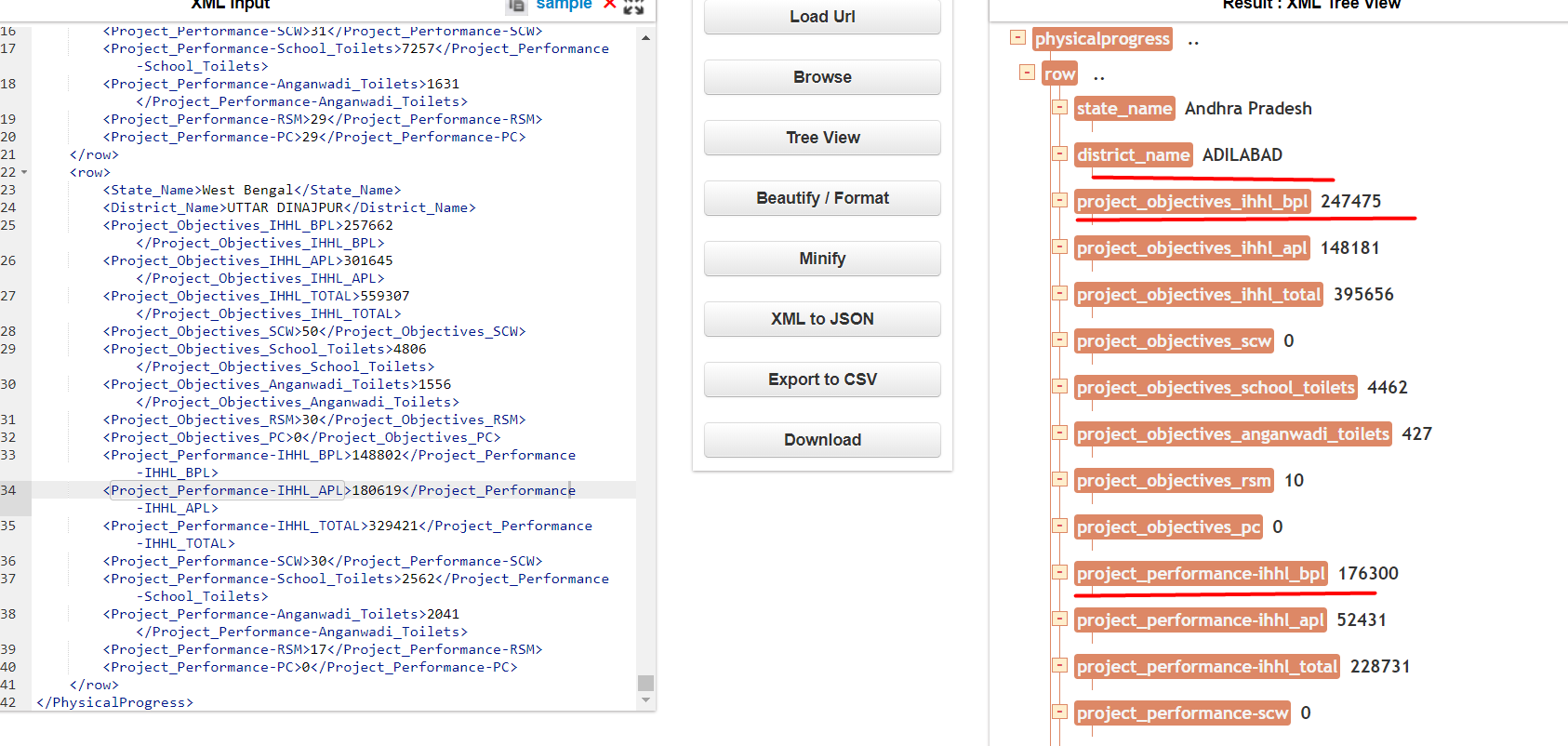
Make some research



As I believe in piggybank there are a class that will do work for us regarding the parsing the data



Now lets take a look on the xml file and try to figure out what is BPL mean and what the column it is.



So I believe for this part of the assignment, we should use the columns :

**District\_name**

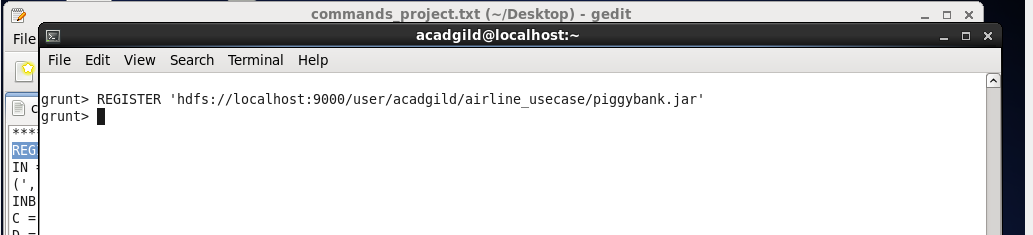
**Project\_objectives\_ihhl\_bpl**

**Project\_performance\_ihhl\_bpl**

And condition for our filter will be

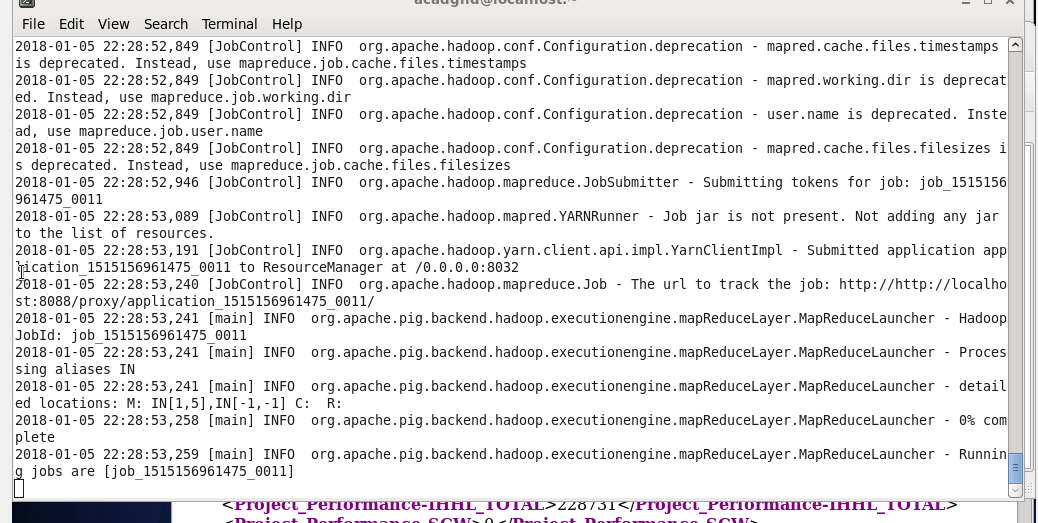
**Project\_objectives\_ihhl\_bpl== Project\_performance\_ihhl\_bpl**

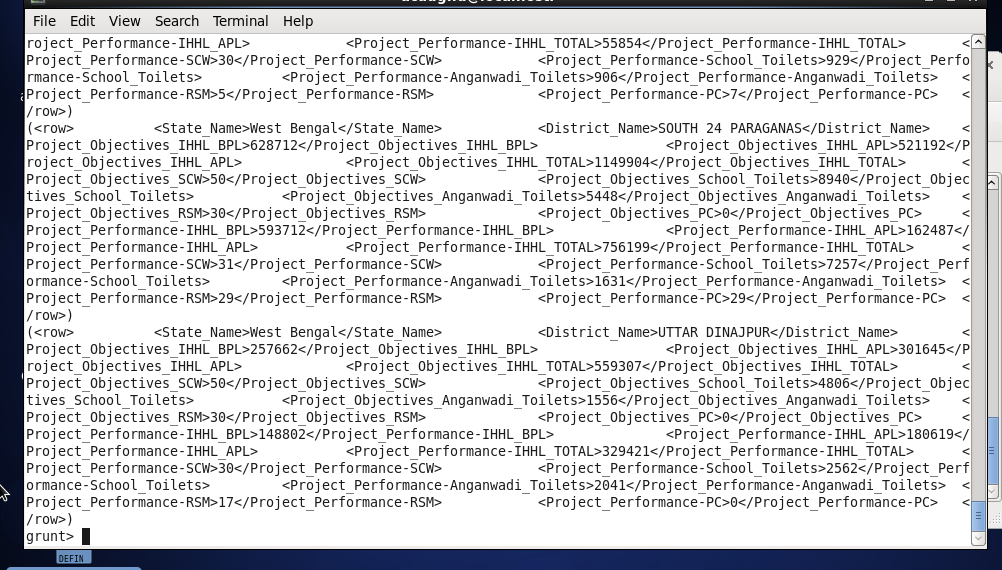
Ok now lets load the data and write the business logic for our pig filter :





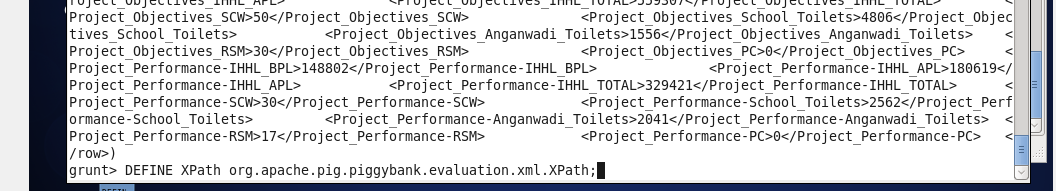
Lets dump IN to see some intermediate output .

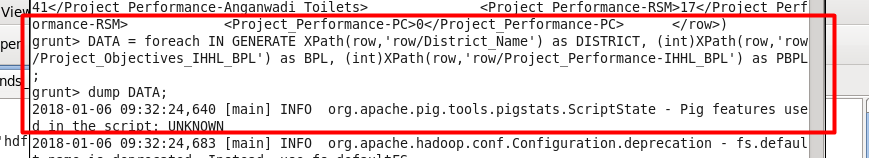




Ok looks good so far

This will be out parser





So let's dump the result first to understand that we are on the right track.



We received nice looking tuple. Let’s move on.

**NOTION : There could be case sensitivity issue so be focused.**

Now we can apply filter and get the result

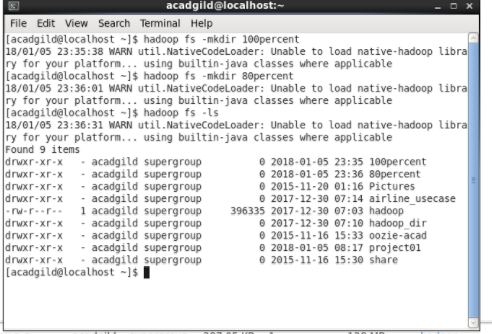


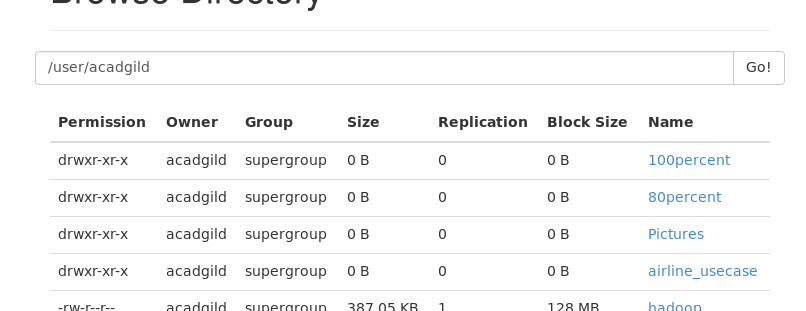


**STORAGE**

Now let's figure out how to store it:

Let's open another terminal window and create folders for 100percent and for the 80percent ( later will discuss )





Now let's put our data to hdfs



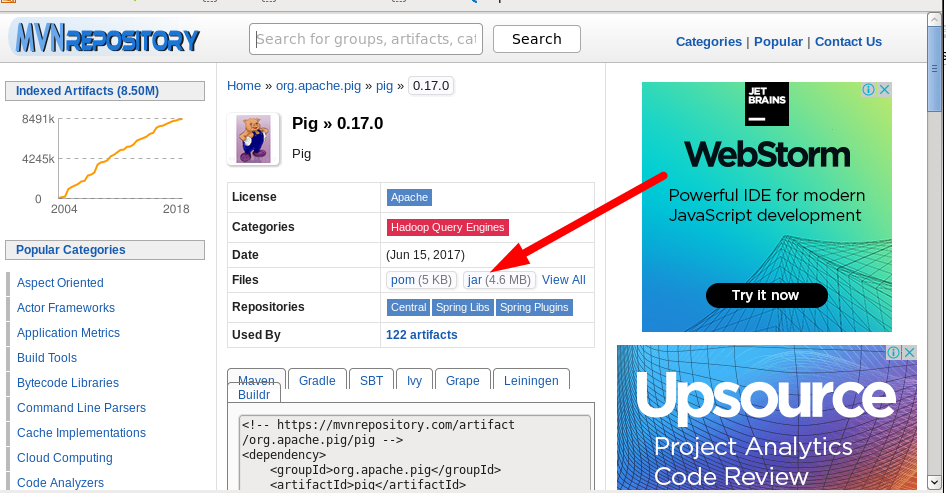
\_\_\_

Lets now solve the problem when :



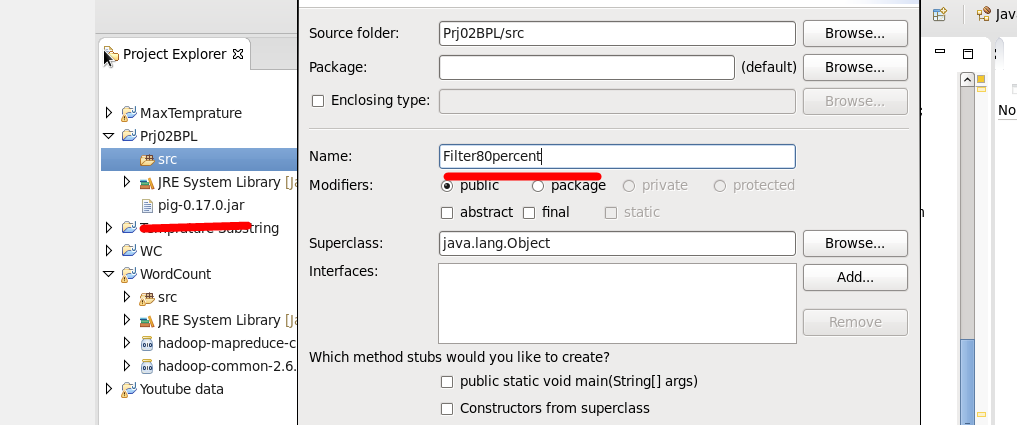
So will be pretty much the same accept :

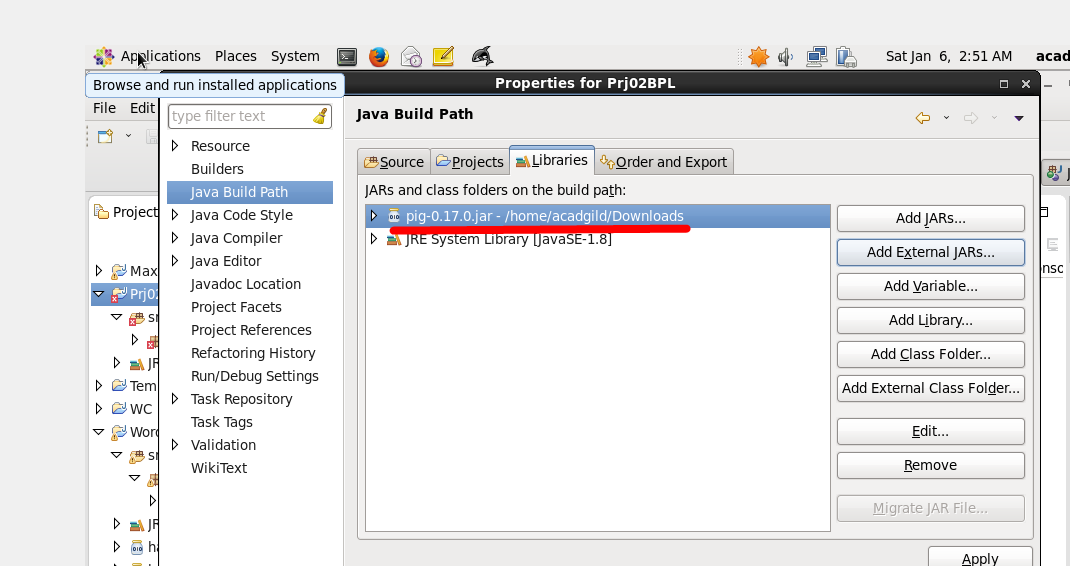




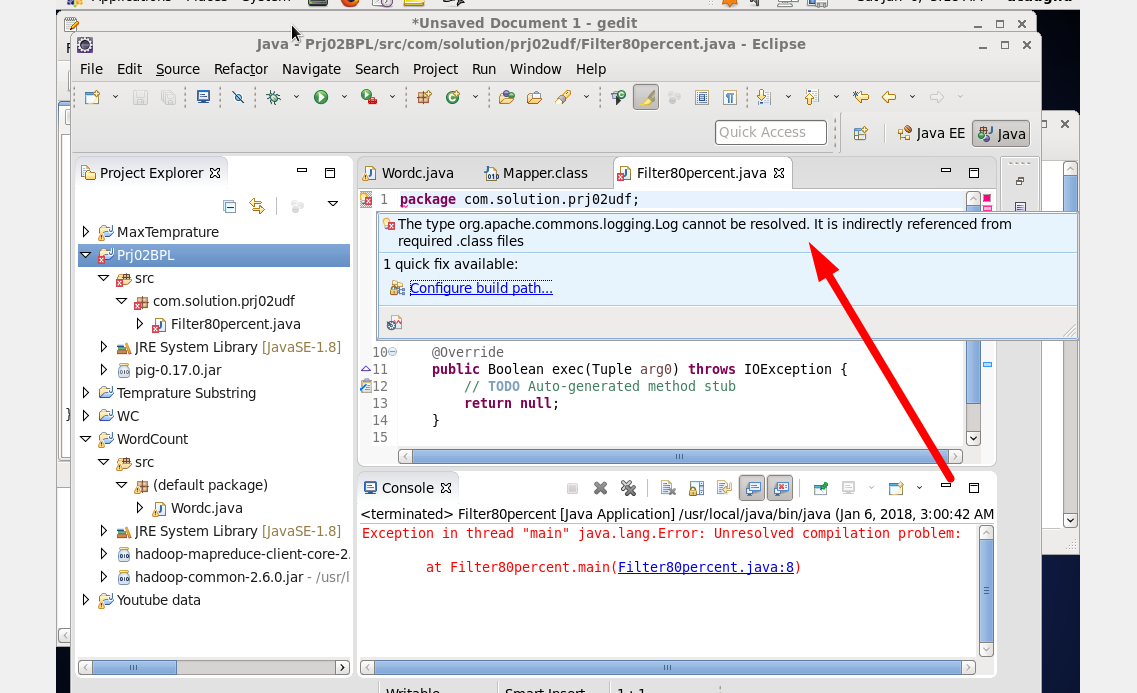
Downloading dependency

Creating java project



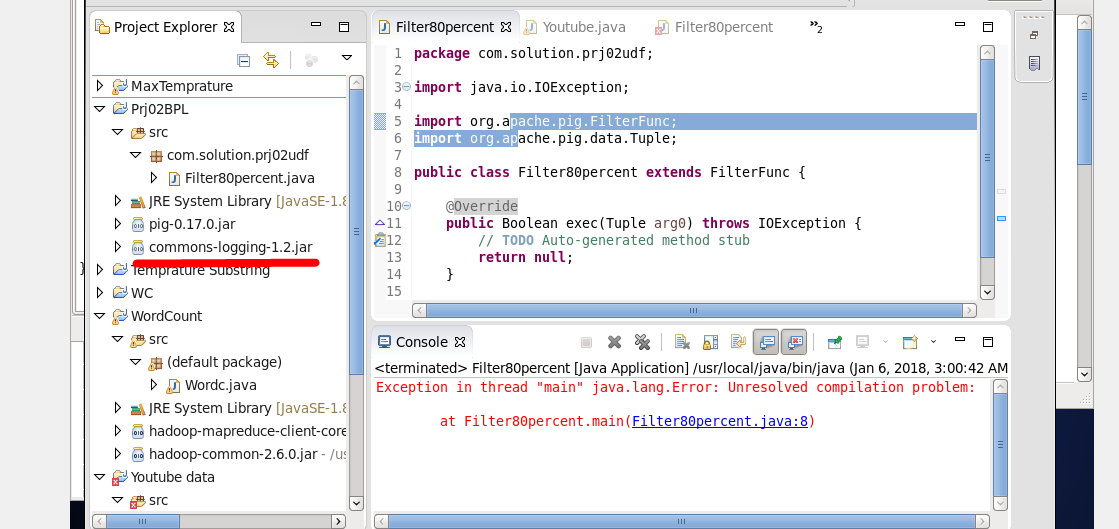


Add link to the jar file

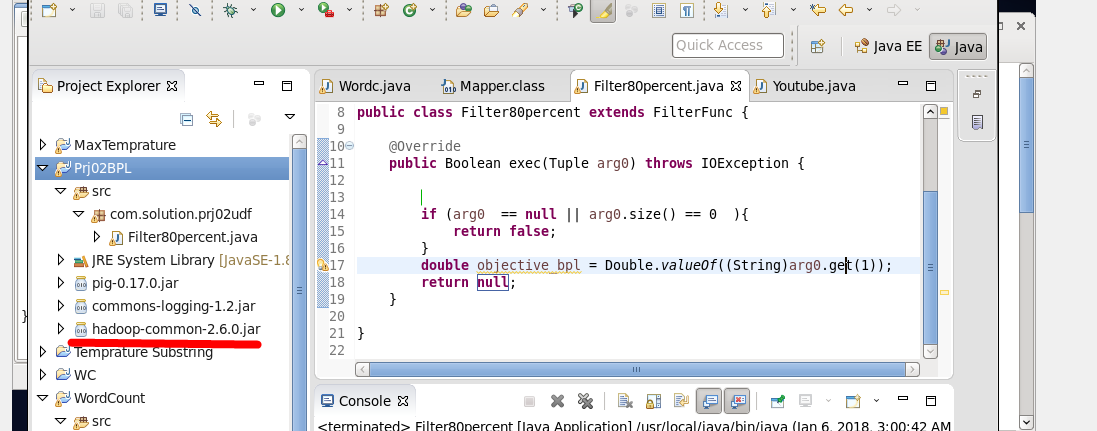


When we add pig jar we receive the exception related to logging that could resolved by adding

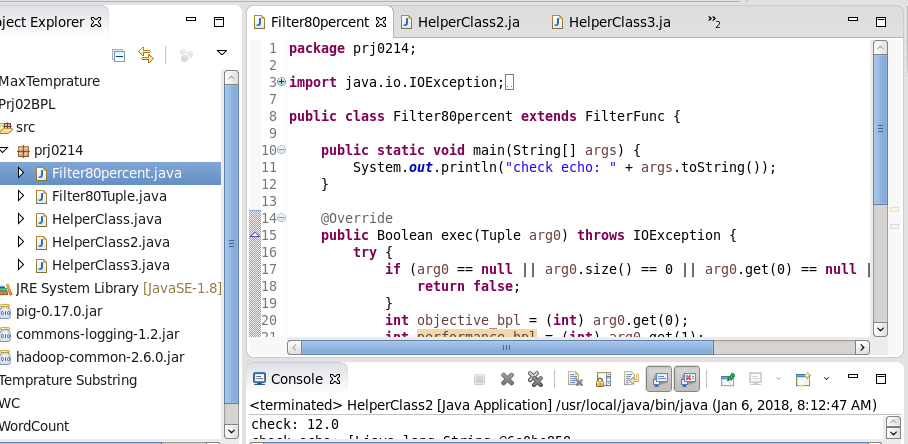
this jar file :



Also we need to add next dependency :

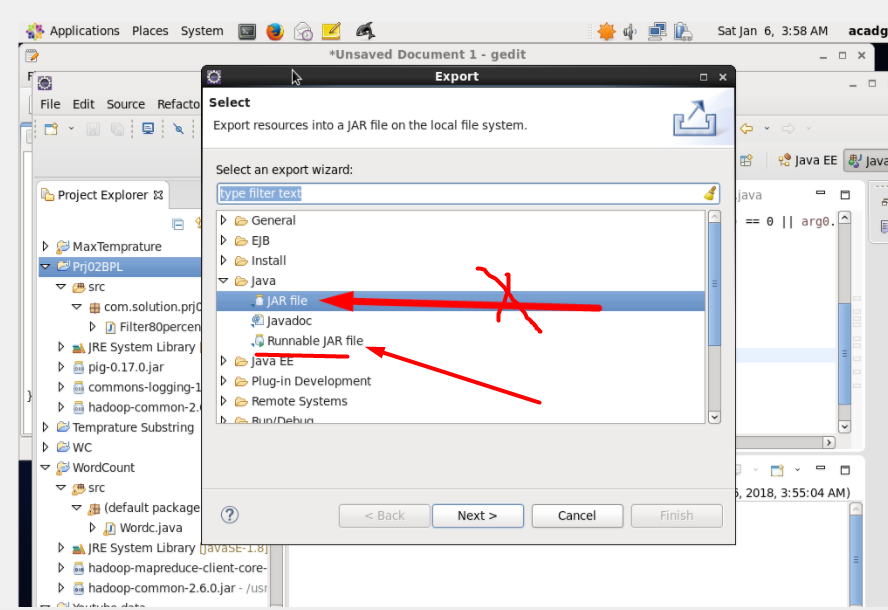


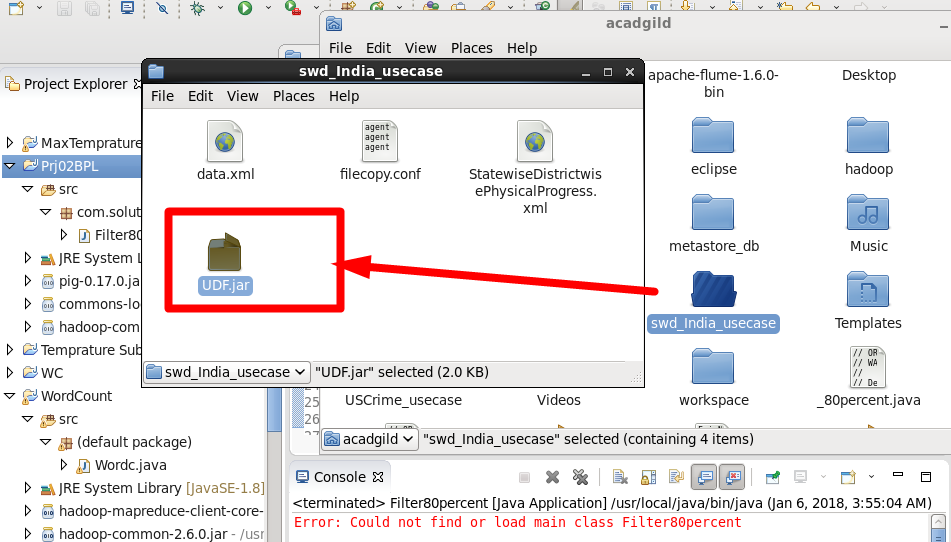
At the pic below we receive our first implementation of the filter so let’s try and see what we get in the pig shell. We should receive the same result as we make the filtration by using just pig build in functions.

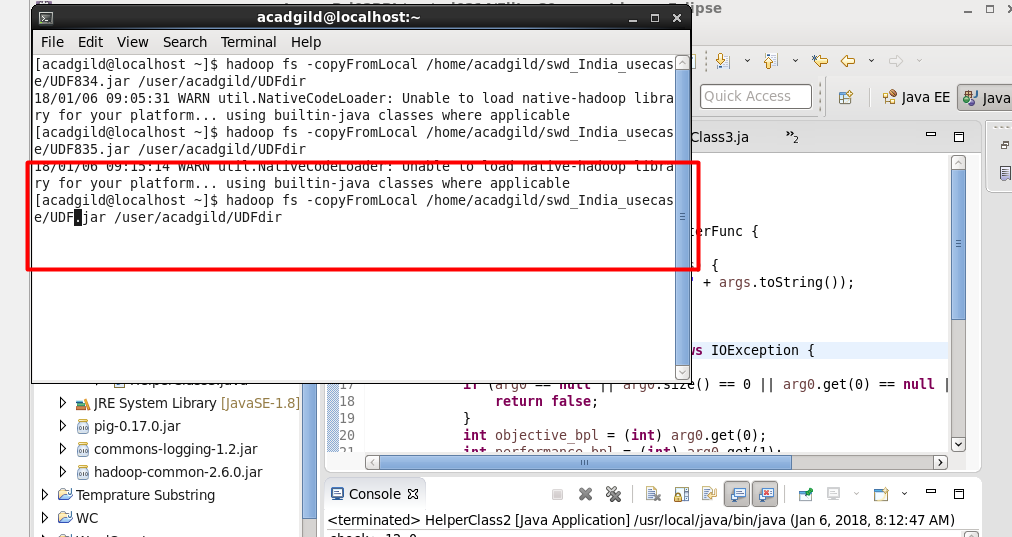




Export the solution to the jar file







Copy the jar file to hdfs

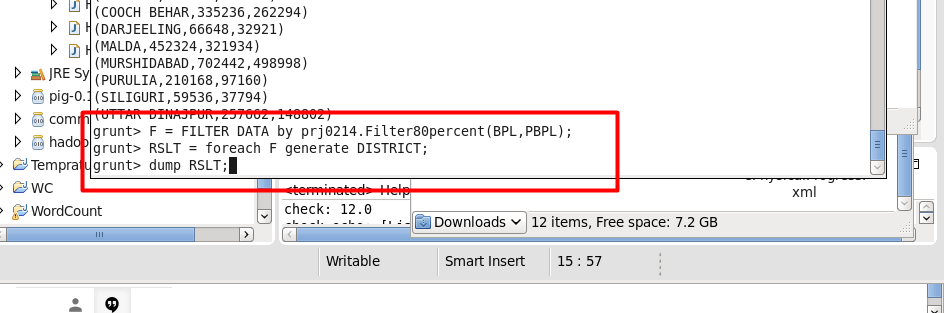


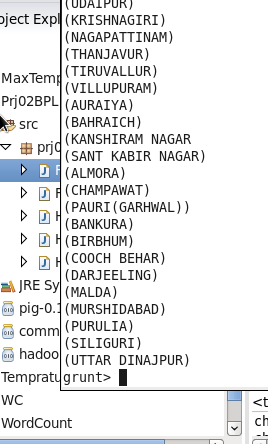
Register the jar



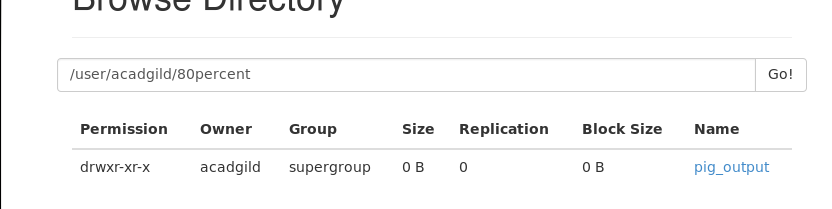
Now let's do the same by using the UDF filter the solution should match in case we are correct.

Now we can apply filter and get out final result and write data to hdfs









Let's check the file context



Check new output result



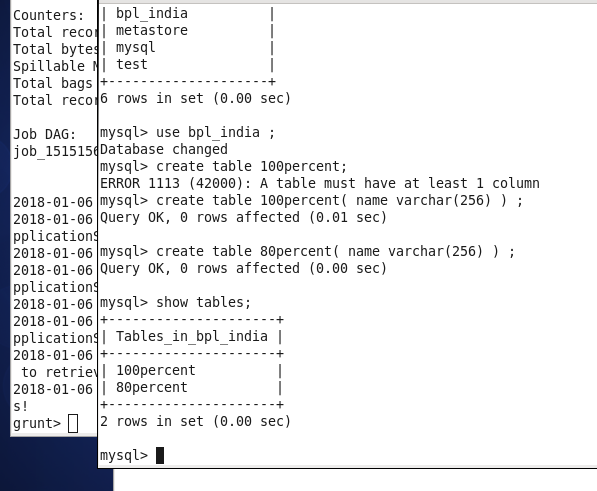
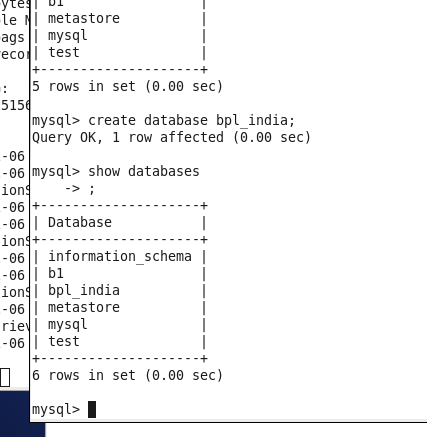
Okey now looks good.

**Moving From HDFS TO MySQL**

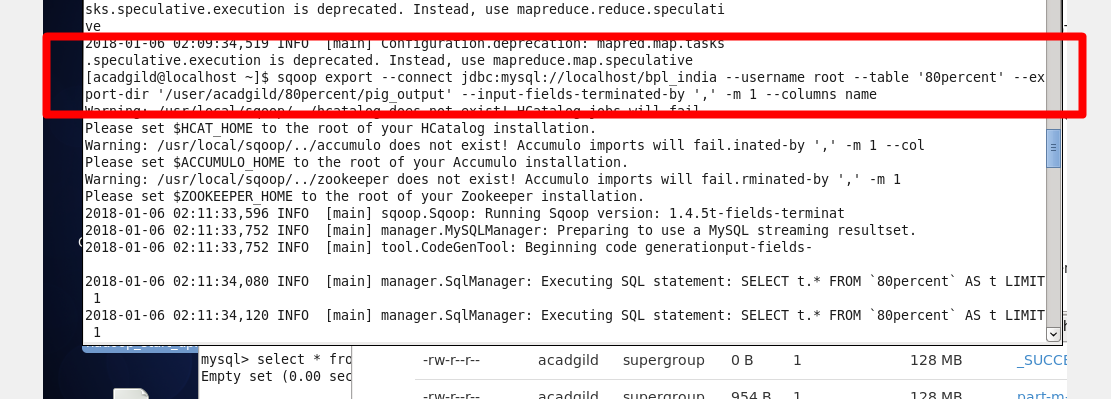
So

Now lets create mysql db ( to move data from hdfs to mysql)

Open mysql shell and create db with two tables for 100 and 80 percent rate correspondingly



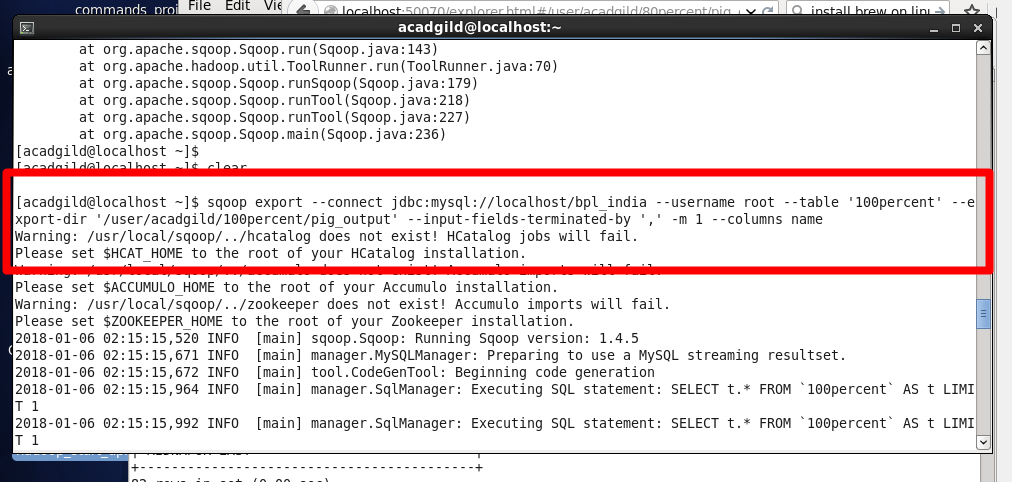
Now lets mode data from hdfs to mysql ;



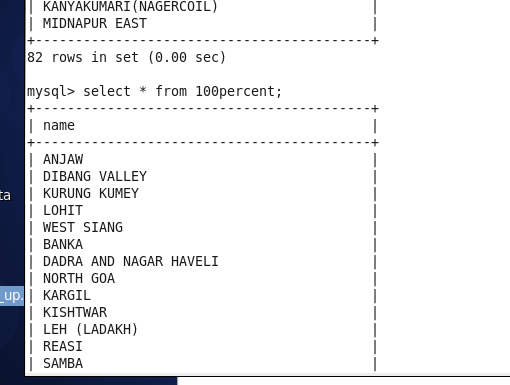
Check data presence



Lets do the same for the “100 percent” table



Check data presence



The project is done !