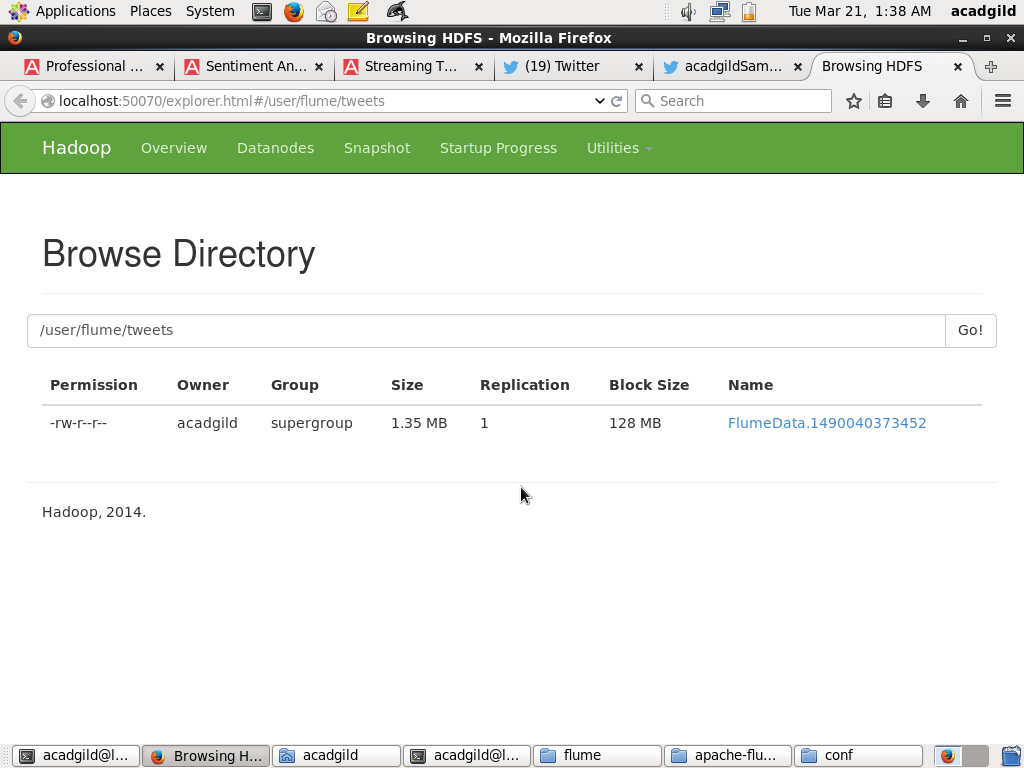
**Twitter Sentiment Analysis:**

Steps involved to perform sentiment analysis on twitter data:

**🡪First step is to save all the twitter data into HDFS:**

Here is the location : /user/flume/tweets



**🡪Register required jars.**

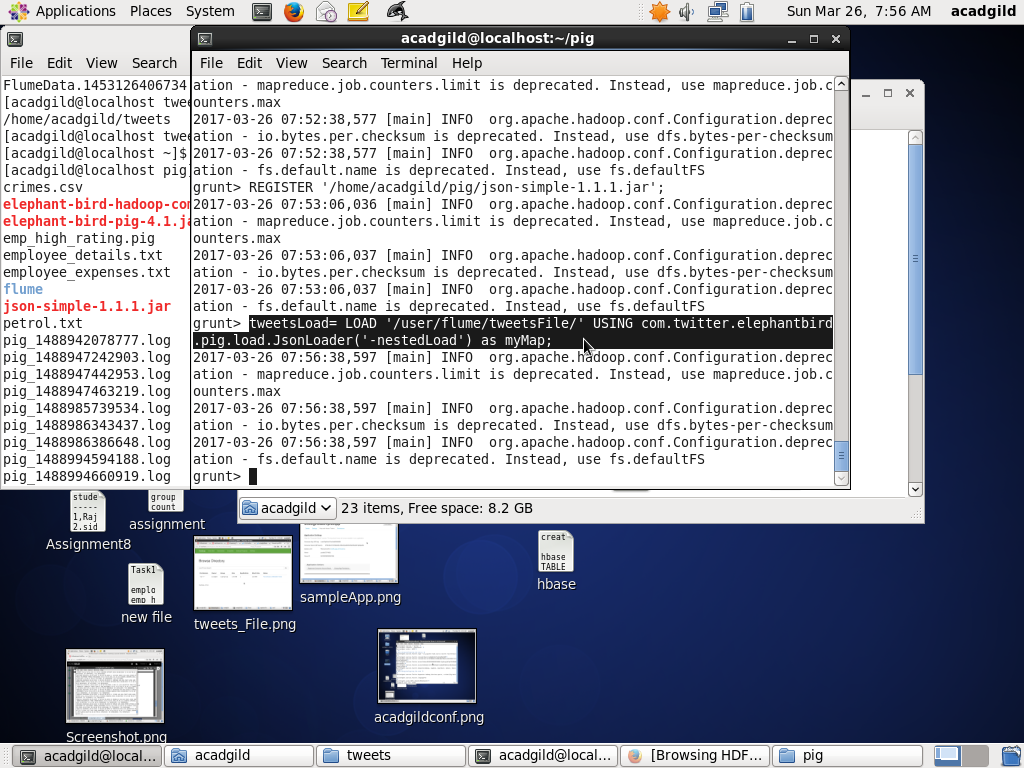
REGISTER '/home/acadgild/pig/elephant-bird-hadoop-compat-4.1.jar';

REGISTER '/home/acadgild/pig/elephant-bird-pig-4.1.jar';

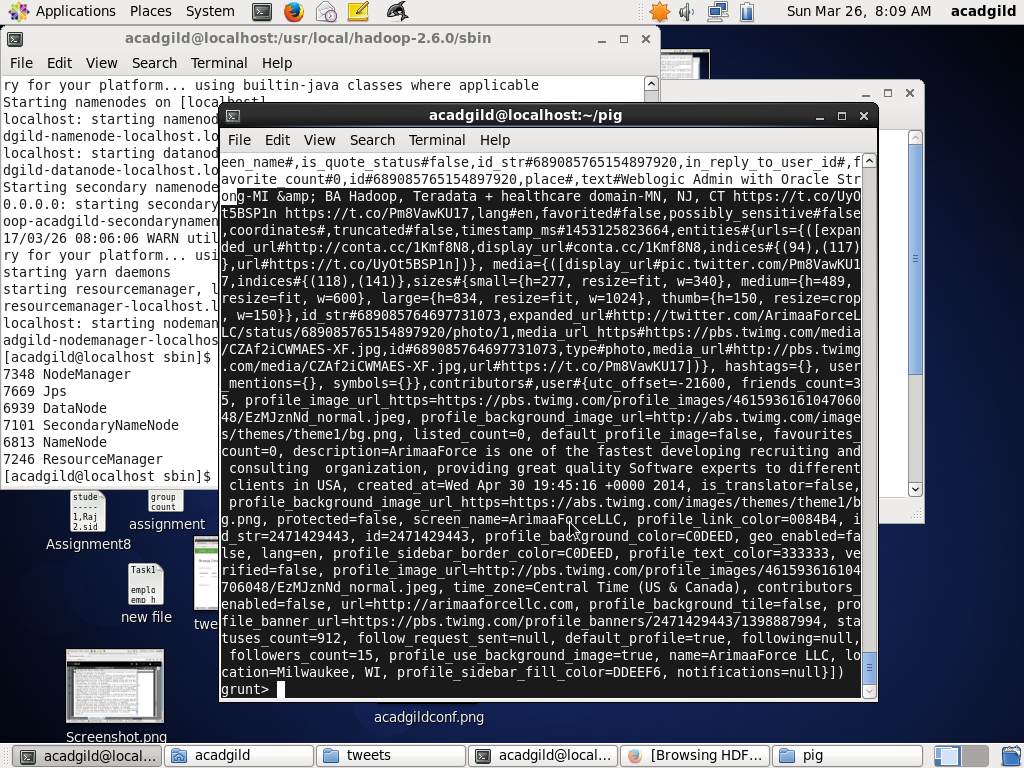
REGISTER '/home/acadgild/pig/json-simple-1.1.1.jar';

**🡪Load tweets files using elephant-bird JsonLoader.**

tweetsLoad = LOAD '/user/flume/tweets/' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') AS myMap;

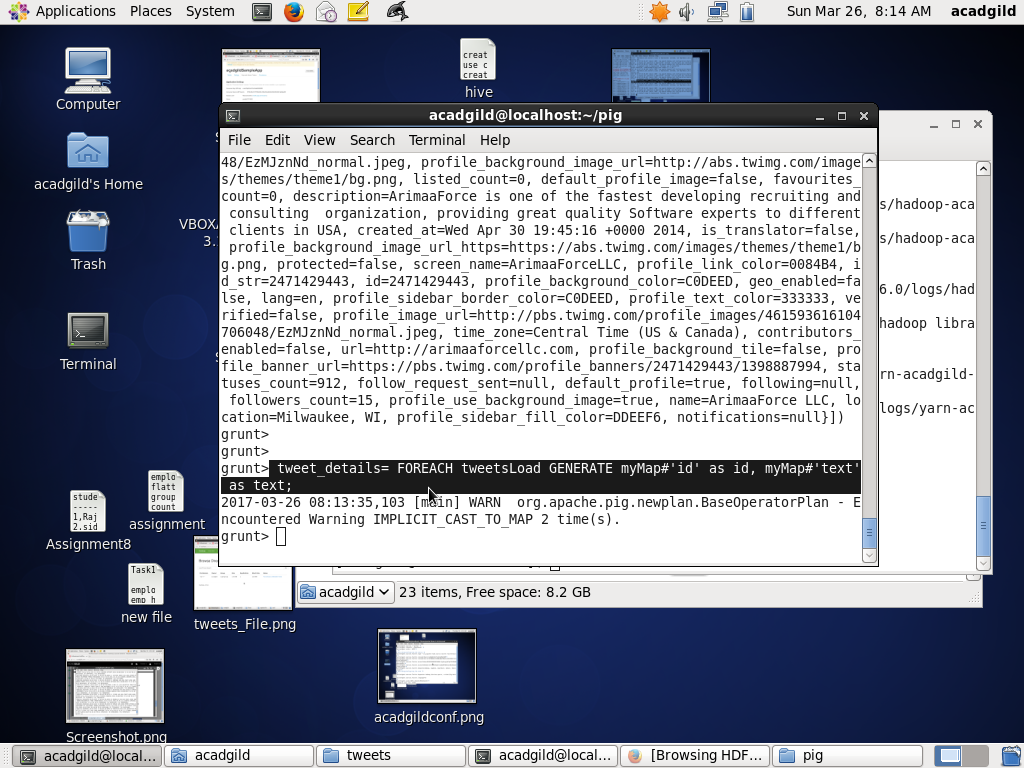


**DUMP tweetsLoad;**



**🡪Extracting id and tweet text from above tweets:**

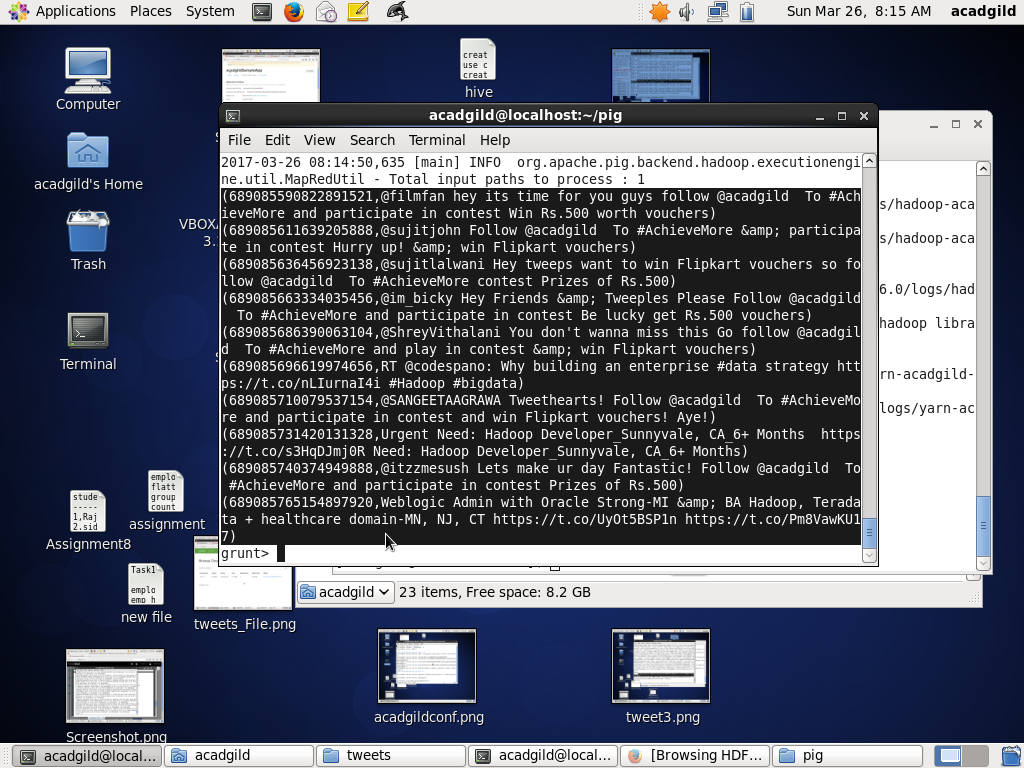
tweet\_details= FOREACH tweetsLoad GENERATE myMap#'id' as id, myMap#'text' as text;



**DESCRIBE tweet\_details;**

tweet\_details: {id: bytearray,text: bytearray}

**DUMP tweet\_details:**



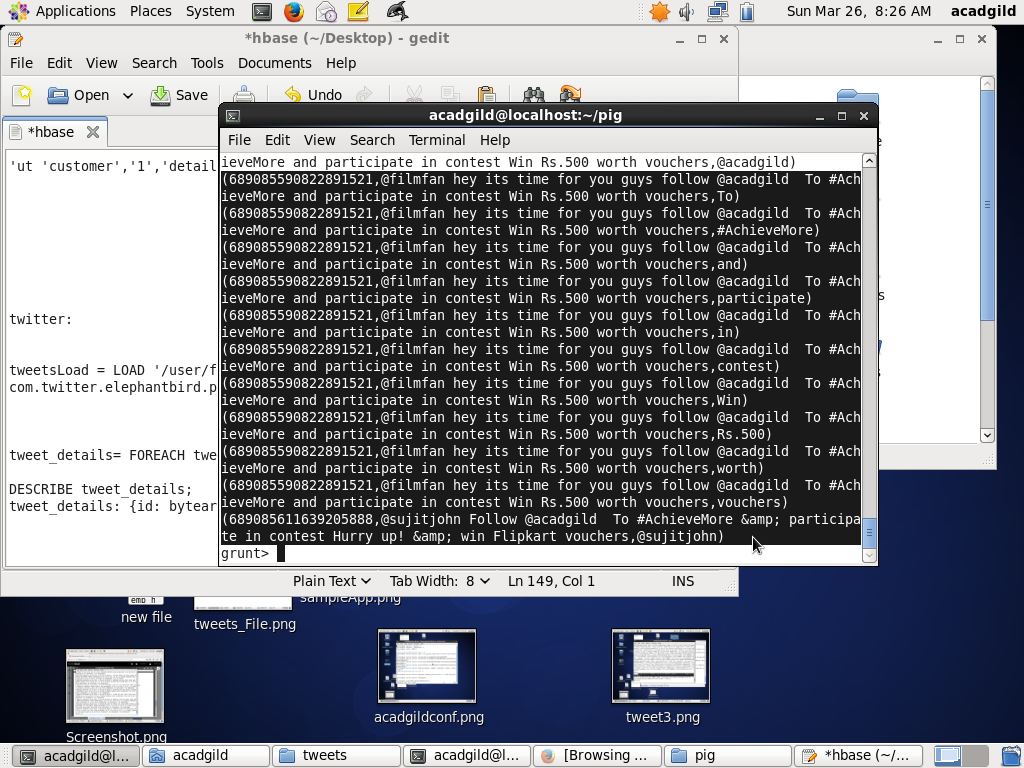
**🡪Extracting words for tweet text:**

tokens= FOREACH tweet\_details GENERATE id,text,FLATTEN(TOKENIZE(text)) as word;

**DESCRIBE tokens;**

tokens: {id: bytearray,text: bytearray,word: chararray}

**DUMP tokens;**

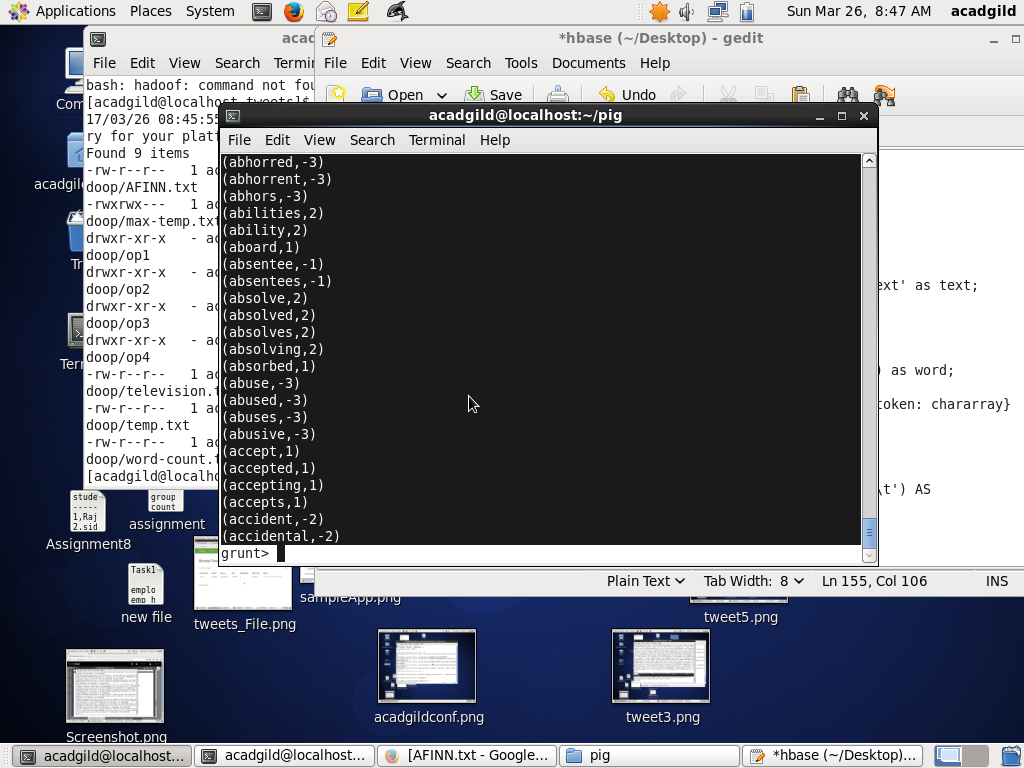


Now, to analyze sentiment for tweets , we need AFINN dictionary which rate each and every word.

**🡪Loading AFINN dictionary file:**

affin\_dic= LOAD '/user/acadgild/hadoop/AFINN.txt' USING PigStorage('\t') AS (word:chararray,rating:int);

**DUMP affin\_dic;**



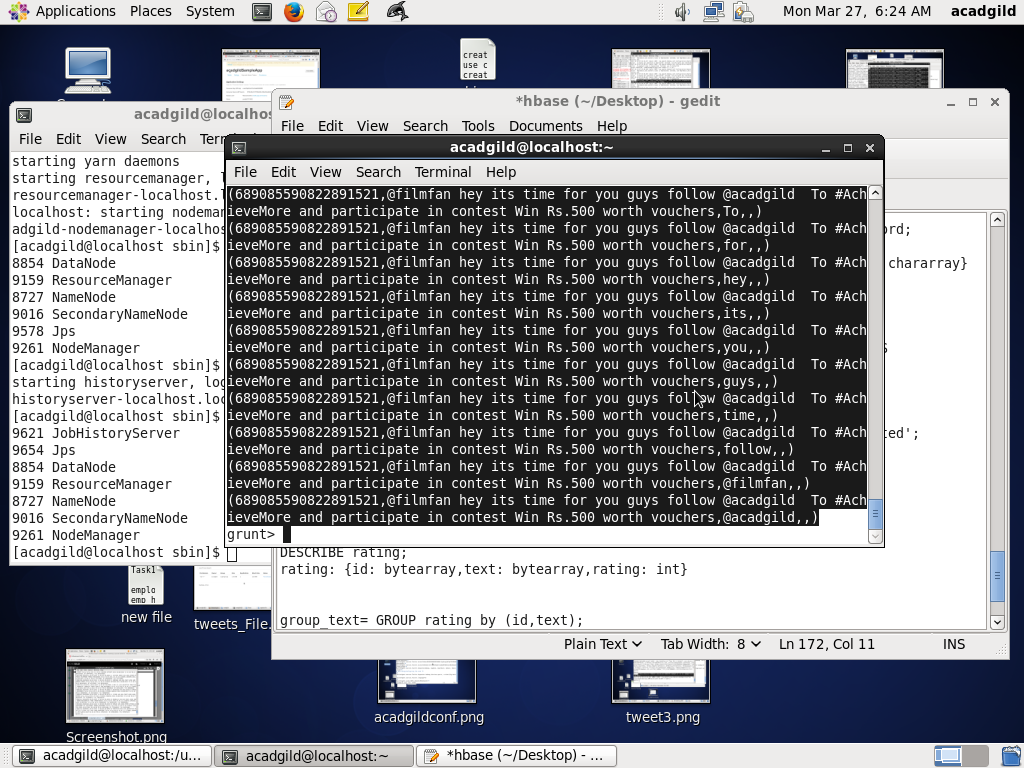
**🡪Joining ‘tokens’ and ‘affin\_dic’ by word:**

word\_join= JOIN tokens BY word LEFT OUTER, affin\_dic BY word USING 'replicated';

**DESCRIBE word\_join;**

word\_join: {tokens::id: bytearray,tokens::text: bytearray,tokens::word: chararray,affin\_dic::word: chararray,affin\_dic::rating: int}

**DUMP word\_join;**



**🡪Getting id and text from ‘tokens’ and rating from ‘affin\_dic’:**

rating= FOREACH word\_join GENERATE tokens::id as id,tokens::text as text,affin\_dic::rating as rating;

**DESCRIBE rating;**

rating: {id: bytearray,text: bytearray,rating: int}

**DUMP rating;**



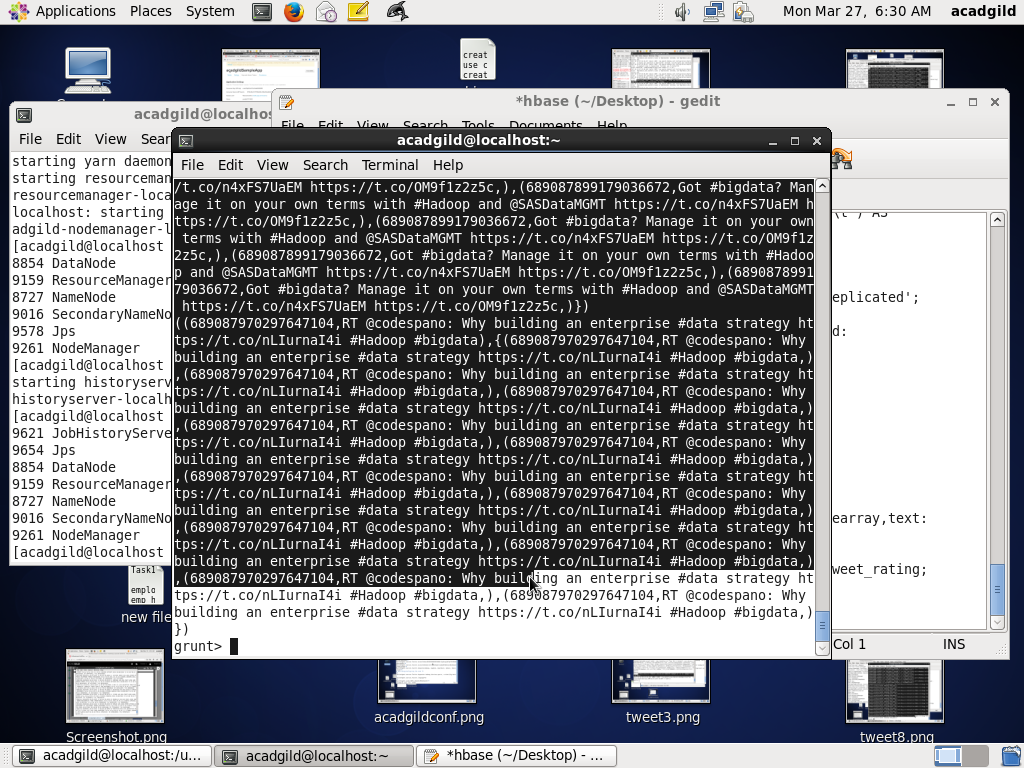
**🡪Grouping ‘rating’ relation by id and text:**

group\_text= GROUP rating by (id,text);

**DESCRIBE group\_text;**

group\_text: {group: (id: bytearray,text: bytearray),rating: {(id: bytearray,text: bytearray,rating: int)}}

**DUMP group\_text;**



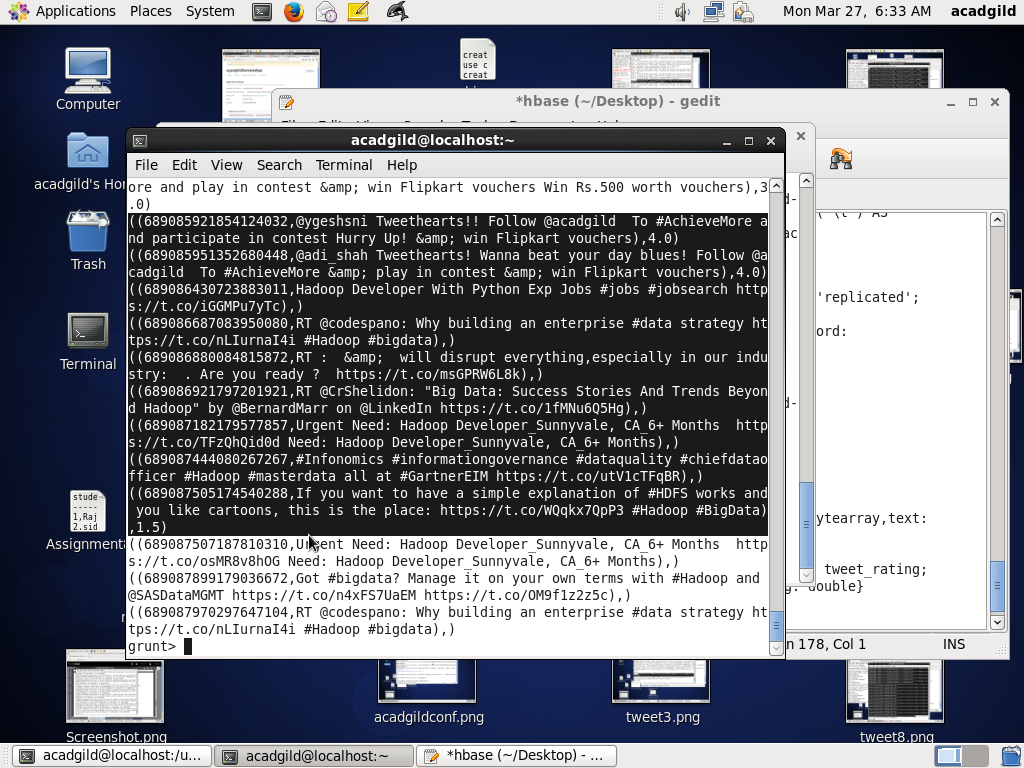
**🡪Calculating Average of rating from ‘group\_text’ relation:**

avg\_rating= FOREACH group\_text GENERATE group,AVG(rating.rating) as tweet\_rating;

**DESCRIBE avg\_rating;**

avg\_rating: {group: (id: bytearray,text: bytearray),tweet\_rating: double}

**DUMP avg\_rating;**



**🡪Filtering only positive tweets:**

positive\_tweets= FILTER avg\_rating by tweet\_rating>=0;

**DUMP positive\_tweets;**

