**LOAD**

Loads data from the file system.

Syntax

LOAD 'data' [USING function] [AS schema];

Suppose we have a data file called emp.

The fields are tab-delimited.

The records are comma separated.

101,swathi,30000  
102,rahila,33000  
103,malar,32000  
104,swathi,36000  
105,rahila,35000  
106,nila,20000  
107,kalai,25000

In this example the default load function, PigStorage, loads data from emp to form relation employee\_details.

The two LOAD statements are equivalent.

Note that, because no schema is specified, the fields are not named and all fields default to type bytearray.

Employee\_details= LOAD '/user/acadgild/emp';

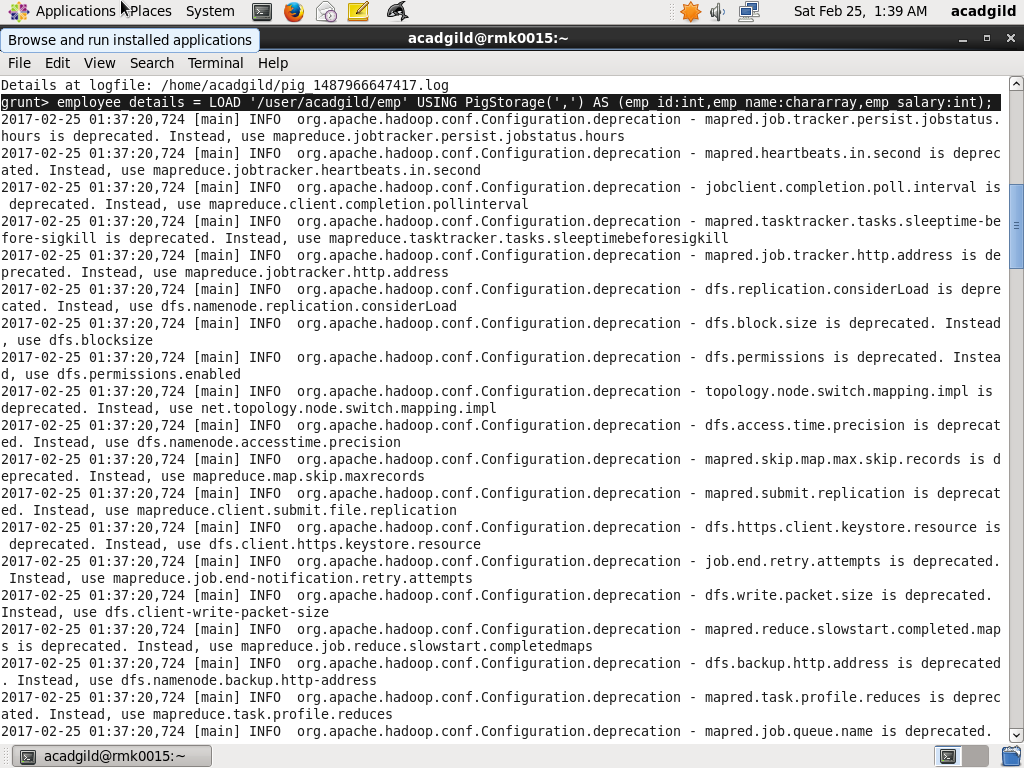
Employee\_details= LOAD '/user/acadgild/emp' USING PigStorage(',') ;

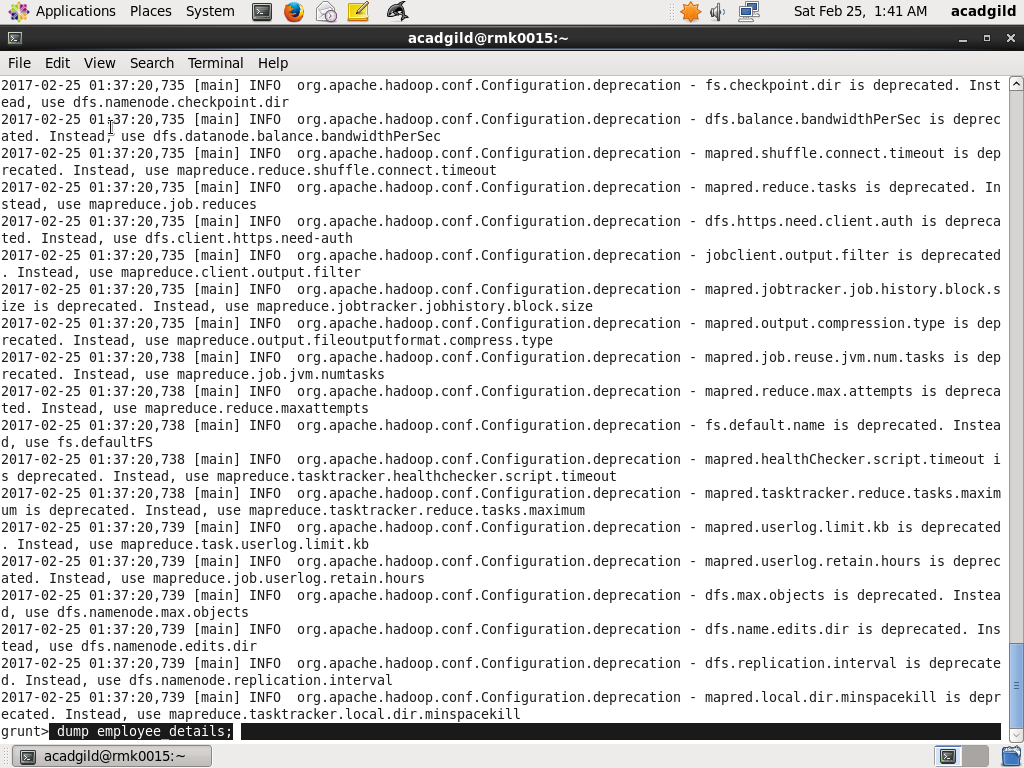
In below example, the data is given with specified datatype.

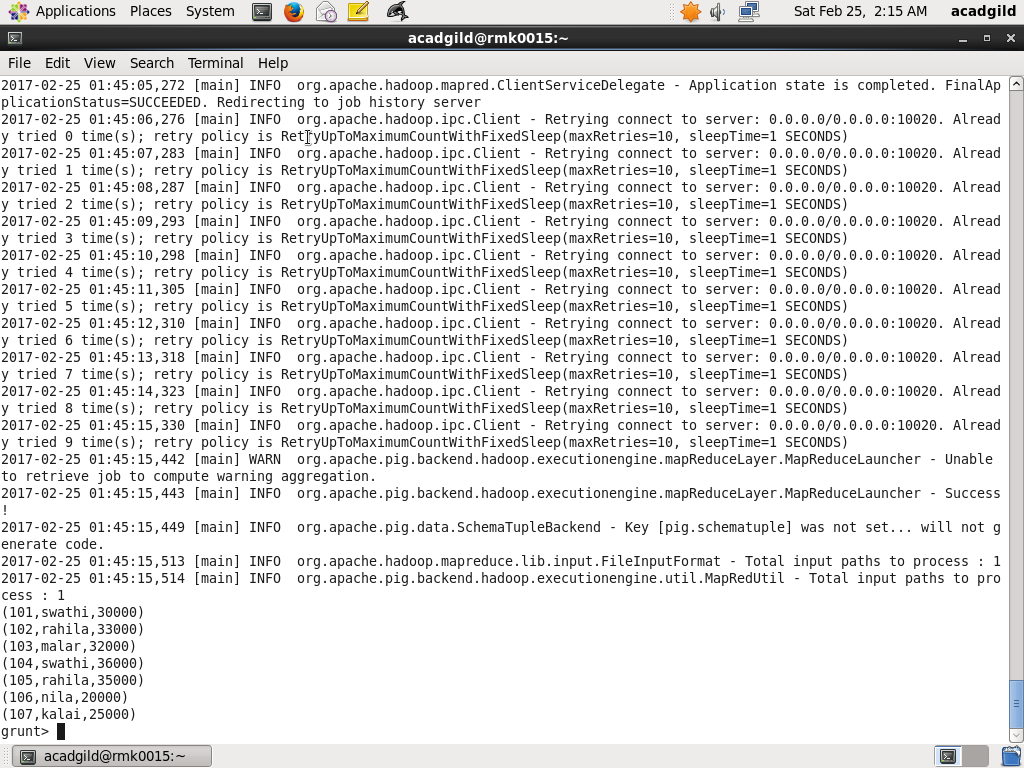
Employee\_details= LOAD '/user/acadgild/emp' USING PigStorage(',') as (emp\_id:int,emp\_name:chararray,emp\_salary:int) ;

DUMP employee\_details;

(101,swathi,30000)  
(102,rahila,33000)  
(103,malar,32000)  
(104,swathi,36000)  
(105,rahila,35000)  
(106,nila,20000)  
(107,kalai,25000)







**STORE**

Stores or saves results to the file system.

**Syntax**

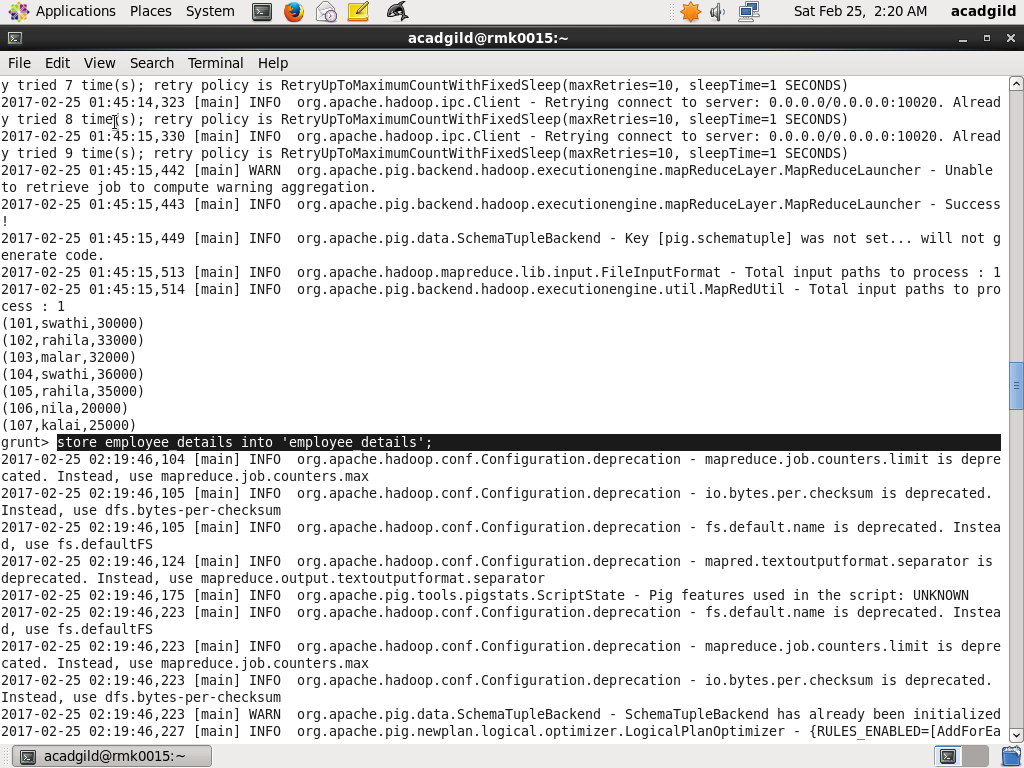
STORE alias INTO 'directory' [USING function];

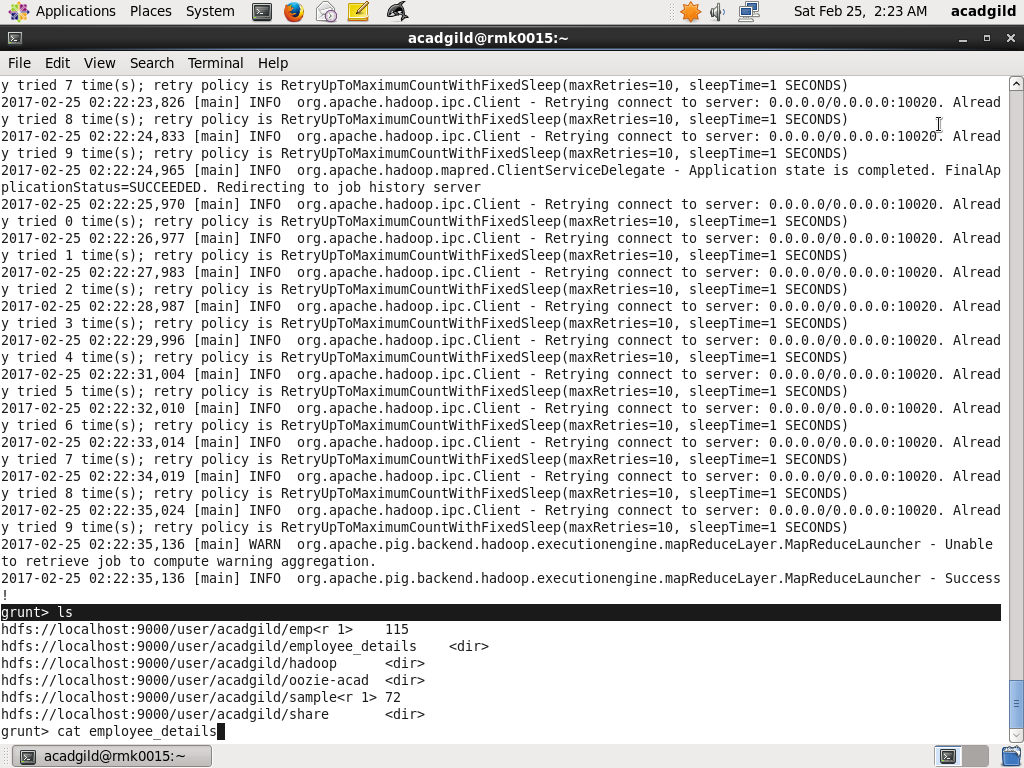
Employee\_details= LOAD '/user/acadgild/emp' USING PigStorage(',') as (emp\_id:int,emp\_name:chararray,emp\_salary:int) ;

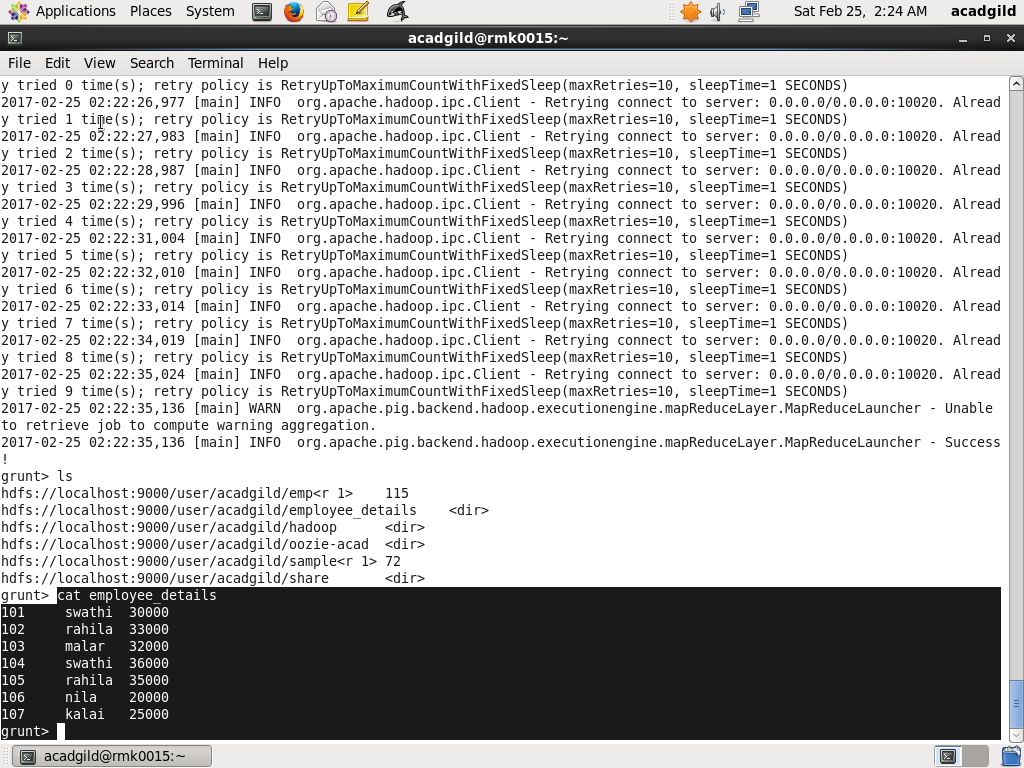
DUMP employee\_details;

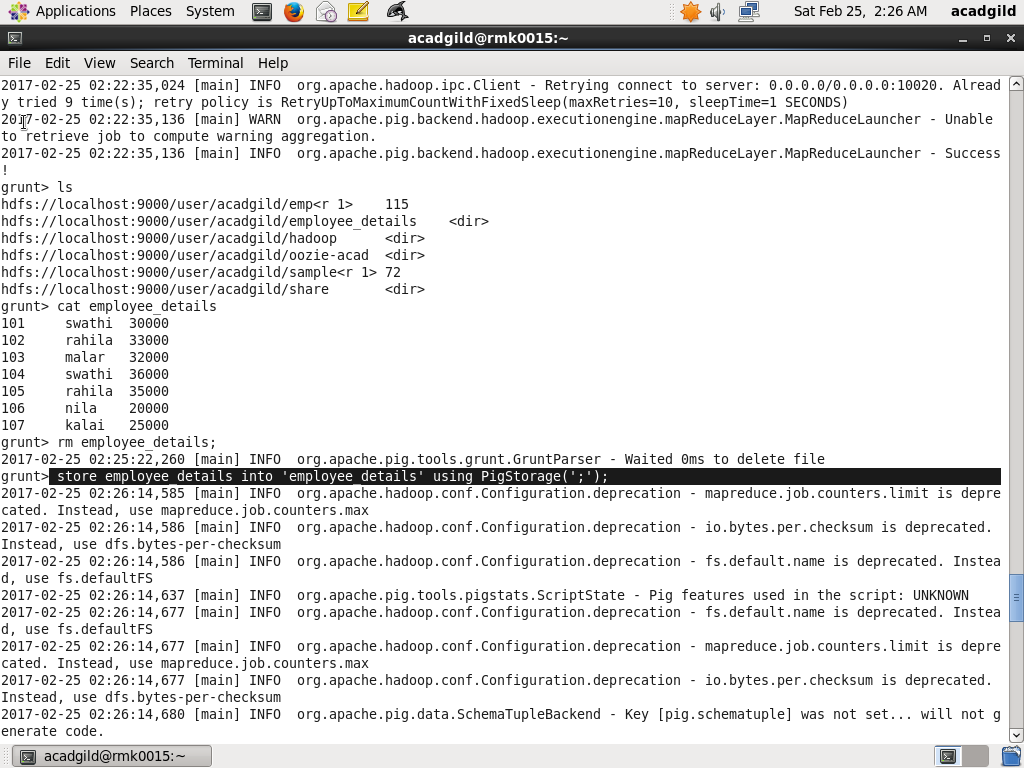
STORE employee\_details INTO 'employee\_details' USING PigStorage (';');

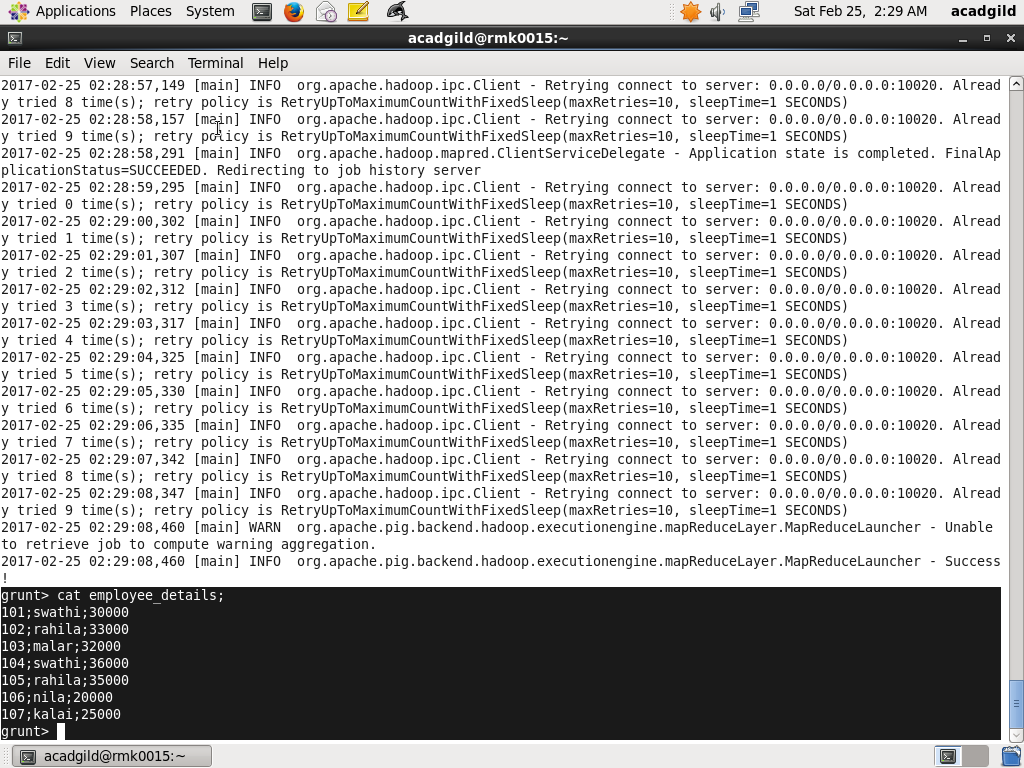
CAT employee\_details;

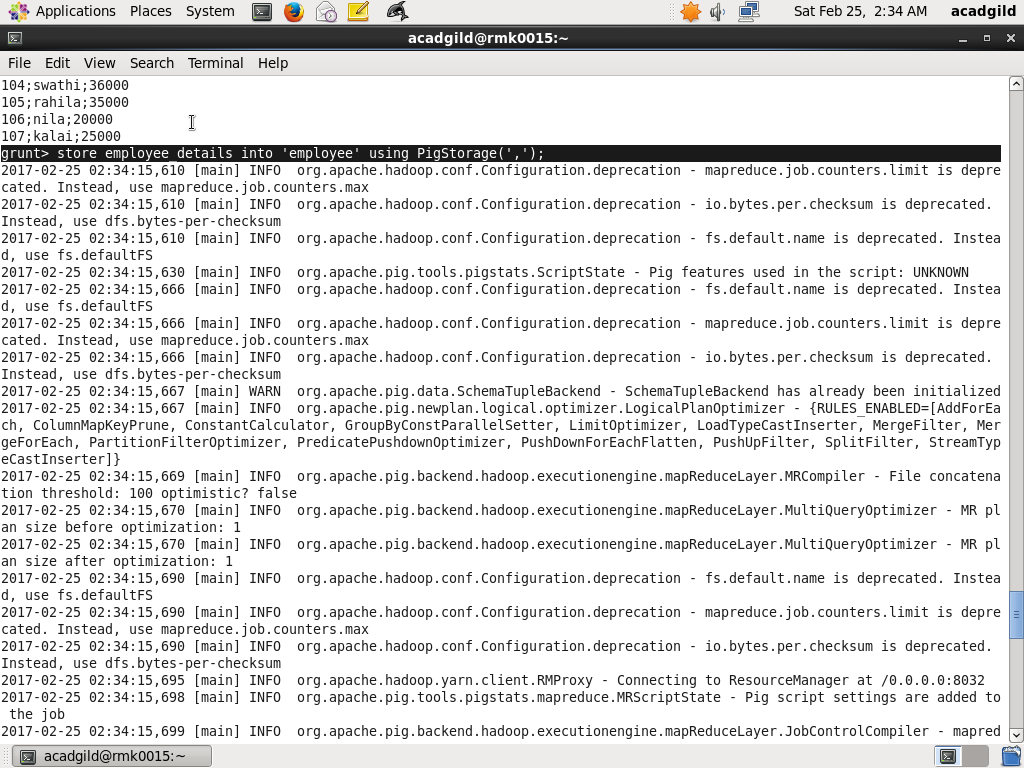


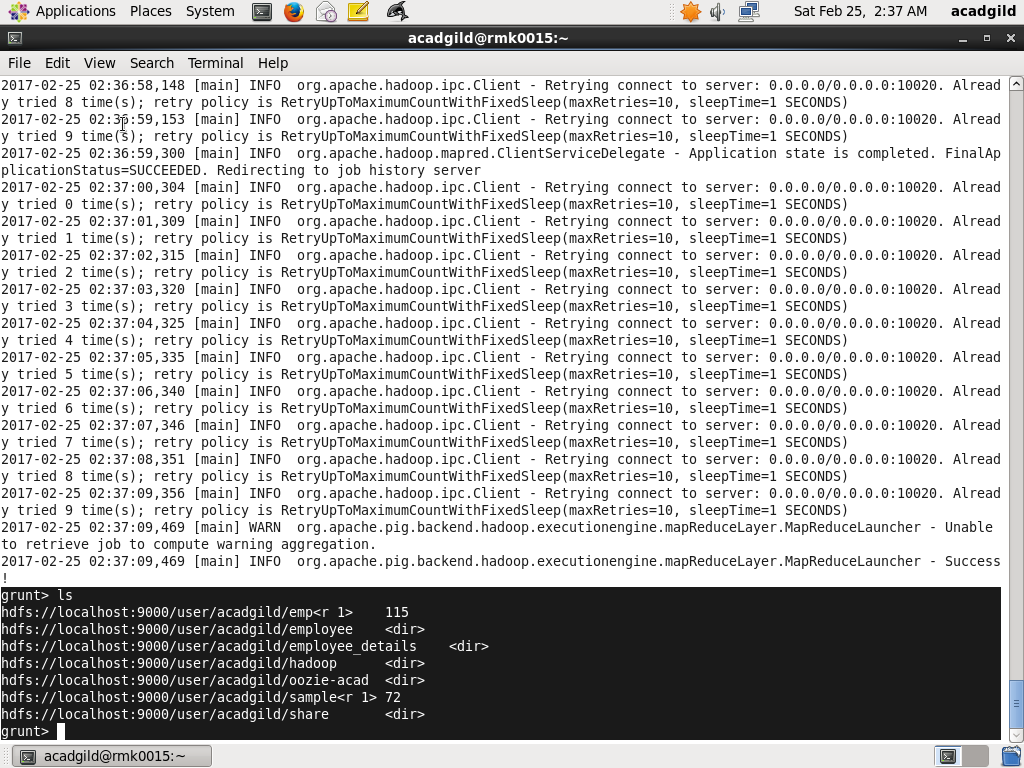


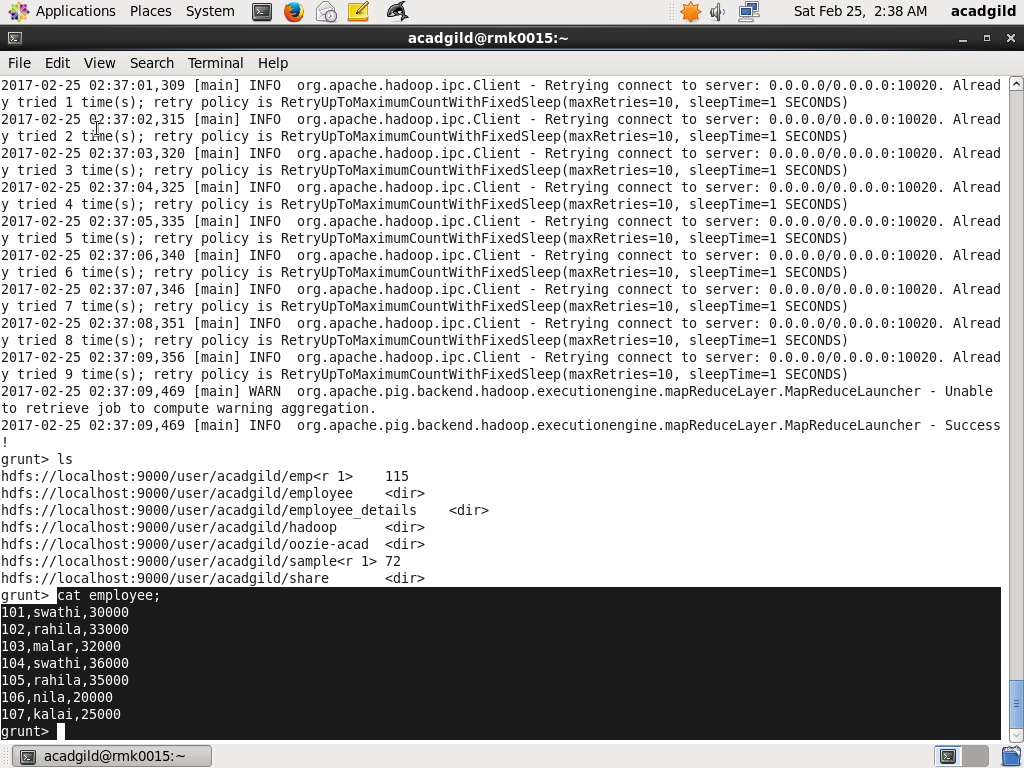


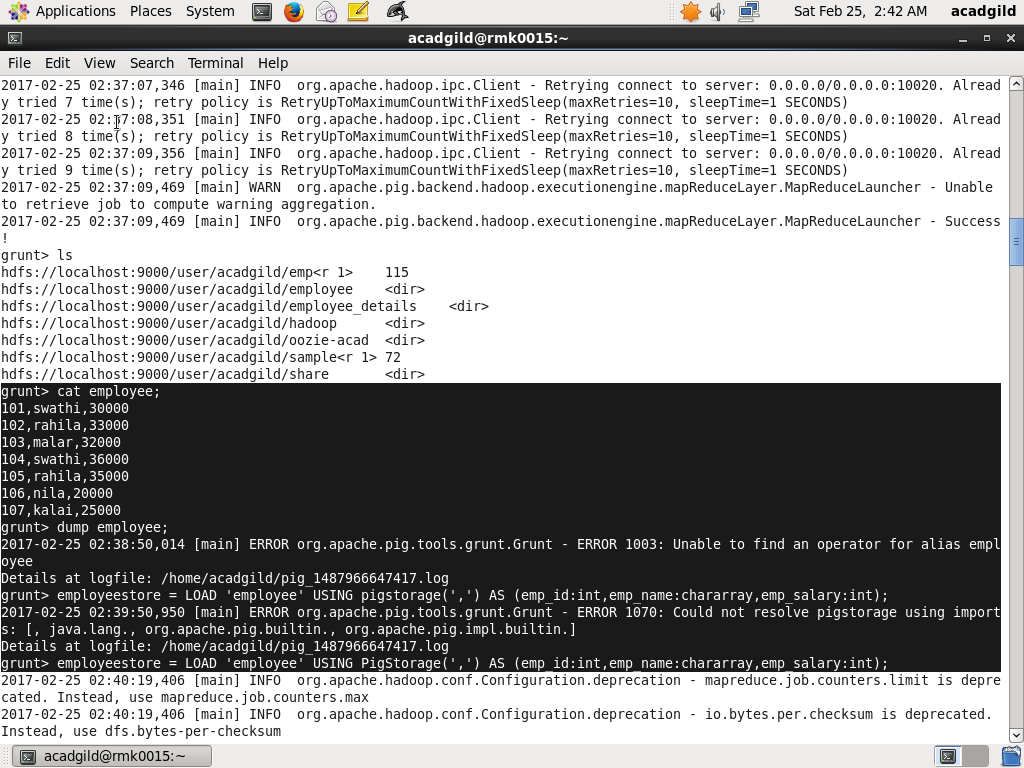


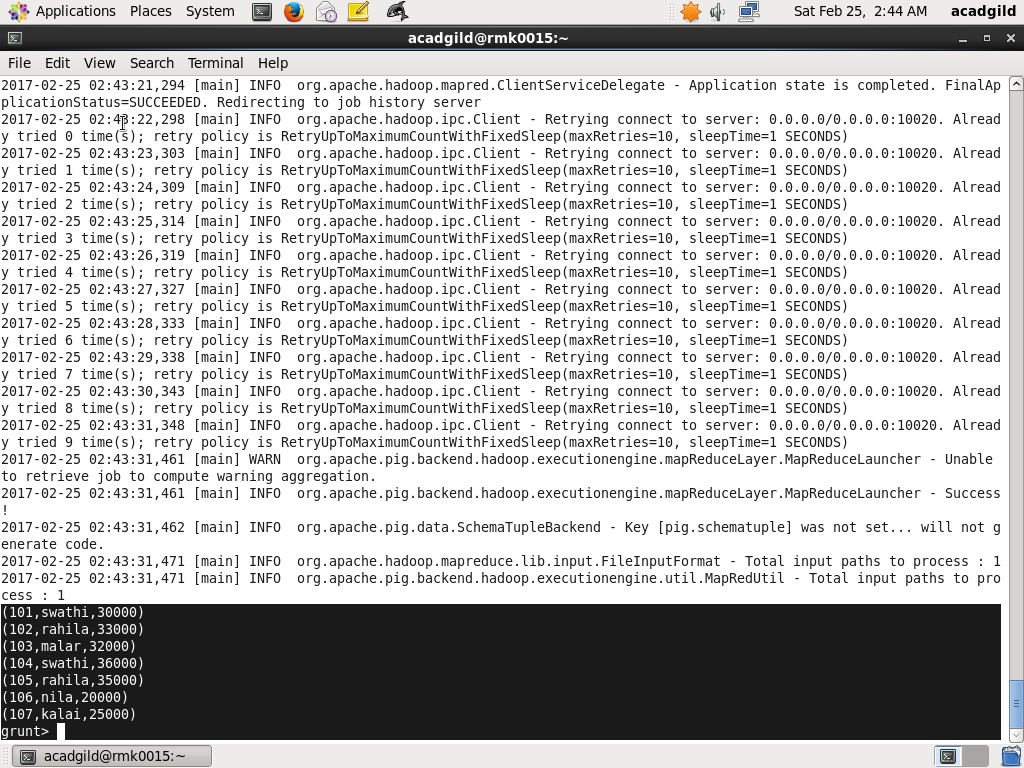












**DUMP**

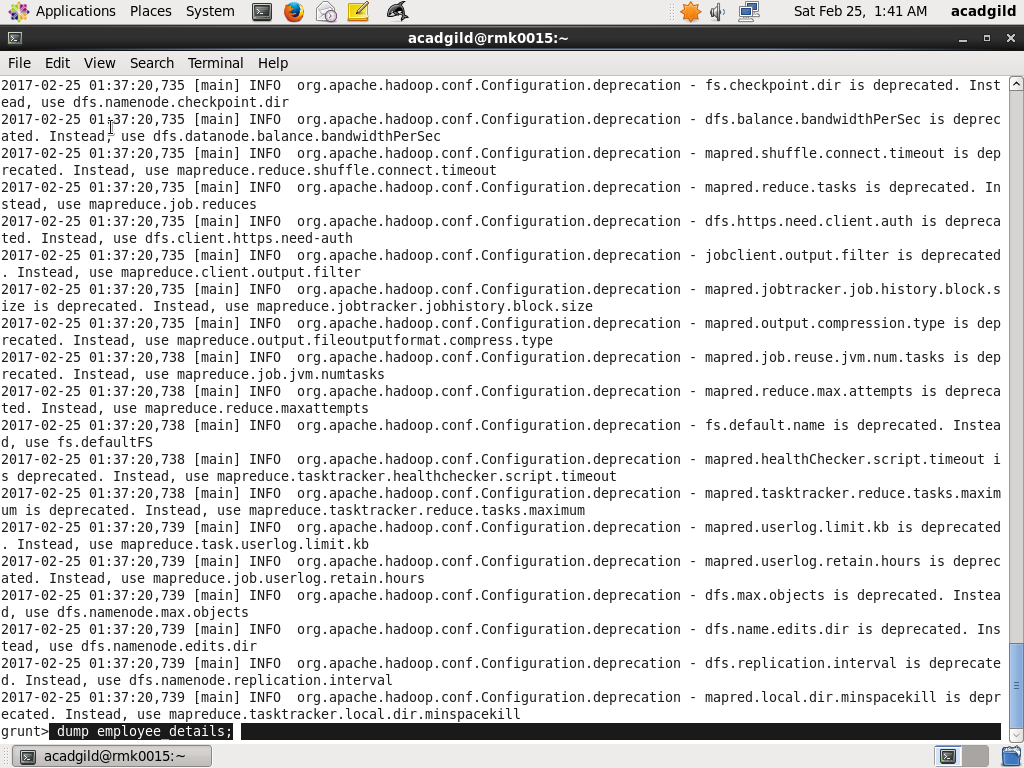
Dumps or displays results to screen.

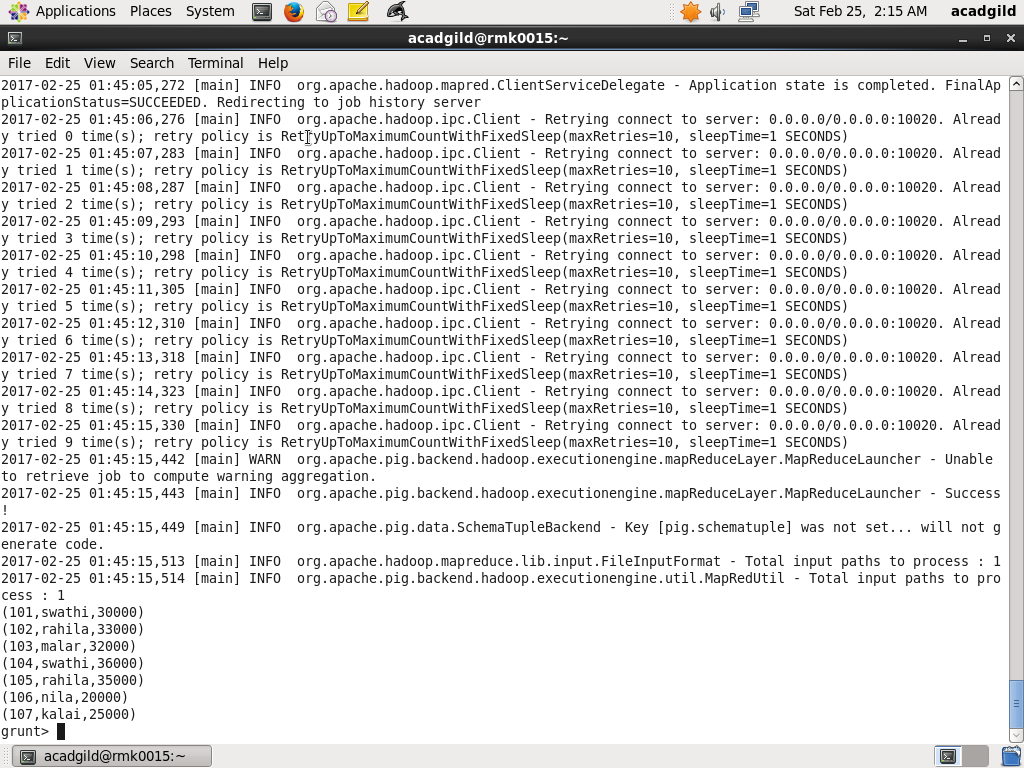
Syntax

DUMP alias;

Example:

DUMP employee\_details;





**FOREACH**

Generates data transformations based on columns of data.

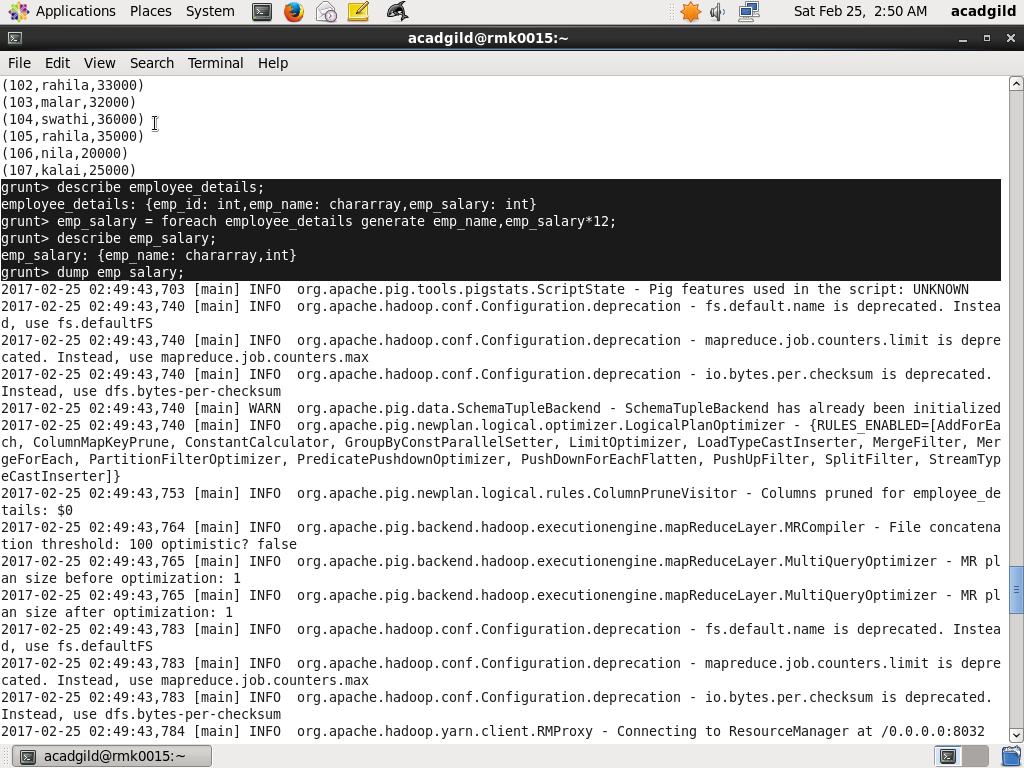
Syntax

alias = FOREACH generate\_operations [AS schema];

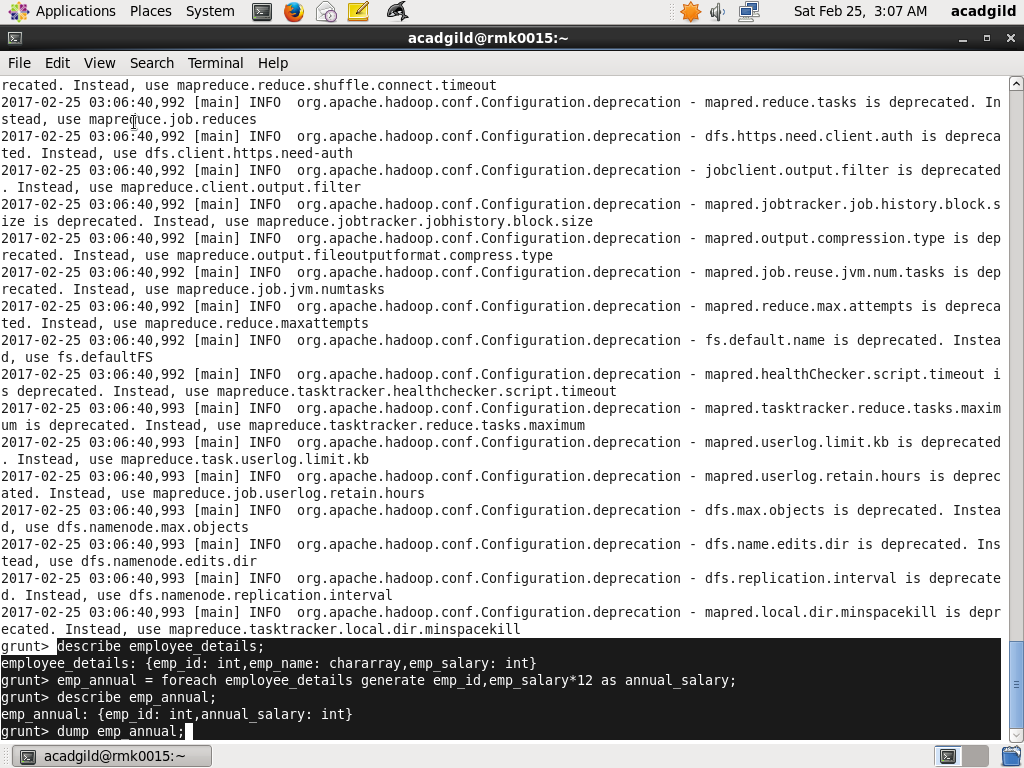
Examples:

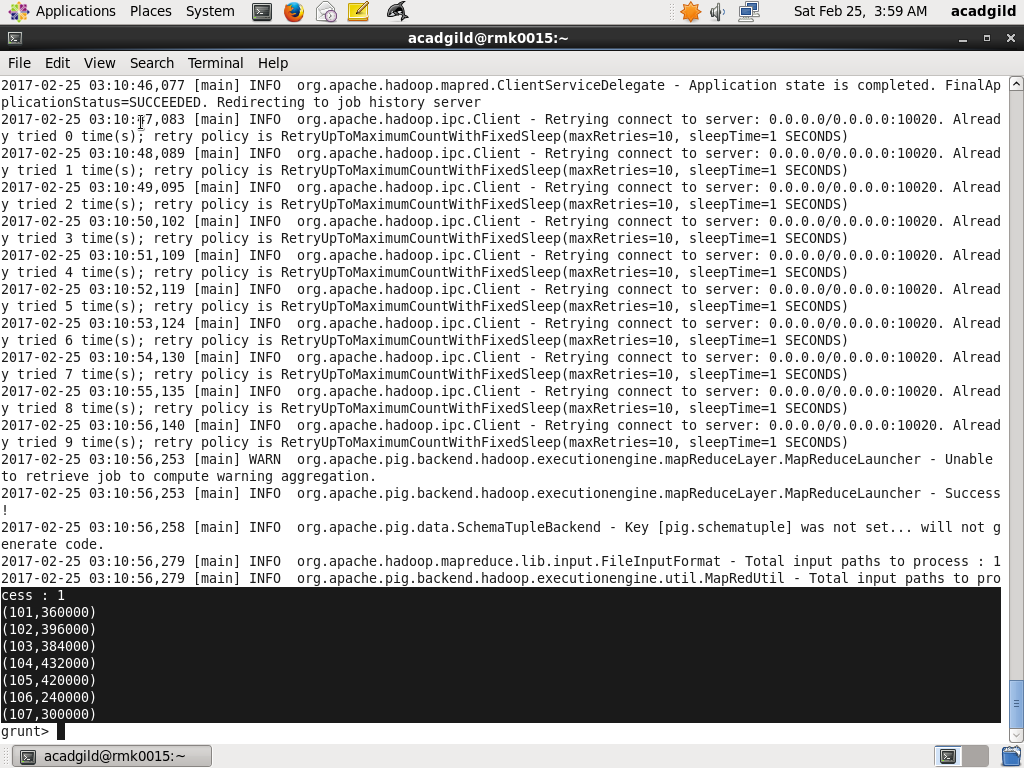
Emp\_salary = FOREACH employee\_details GENERATE emp\_name,emp\_salary\*12;

Emp\_salary = FOREACH employee\_details GENERATE emp\_name,emp\_salary\*12 as annual\_salary;









**FILTER**

Selects tuples from a relation based on some condition.

Syntax

alias = FILTER alias BY expression;

Employee\_details=LOAD ‘/user/acadgild/emp’ using PigStorage(‘,’) as (emp\_id:int,emp\_name:chararray,emp\_salary:int);

DUMP employee\_details;

In this example the condition states that if the emp\_id field less than 105, then include the tuple with relation filter\_employee.

Filter\_employee = FILTER employee\_details BY emp\_id<105;

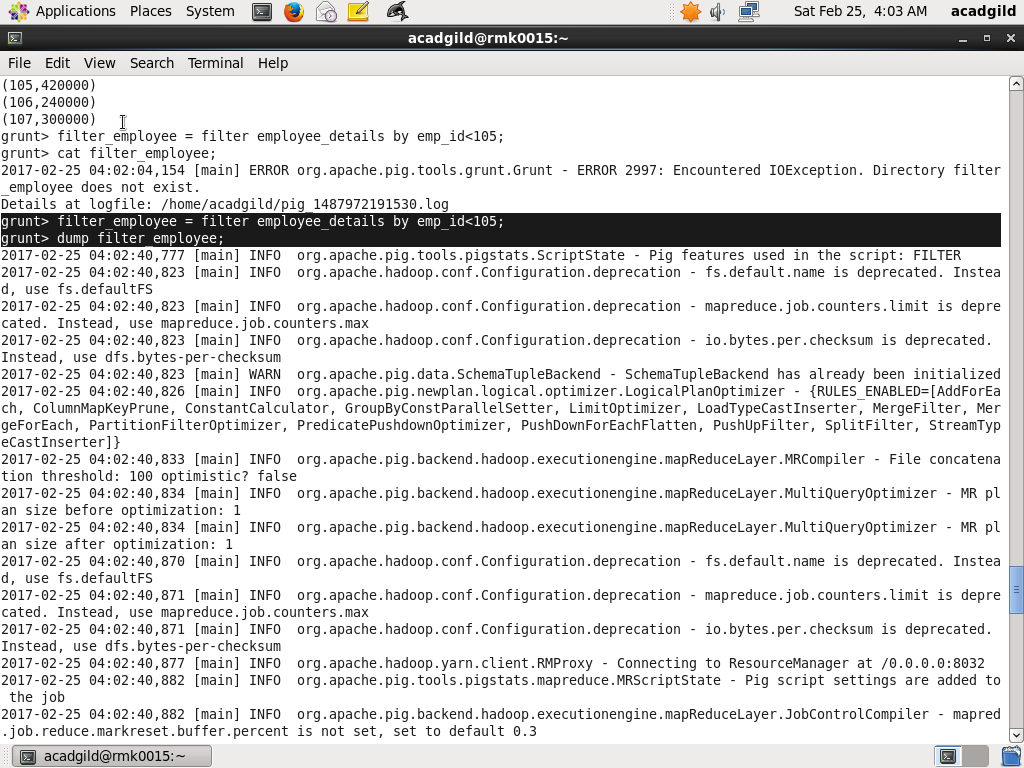
DUMP filter\_employee;

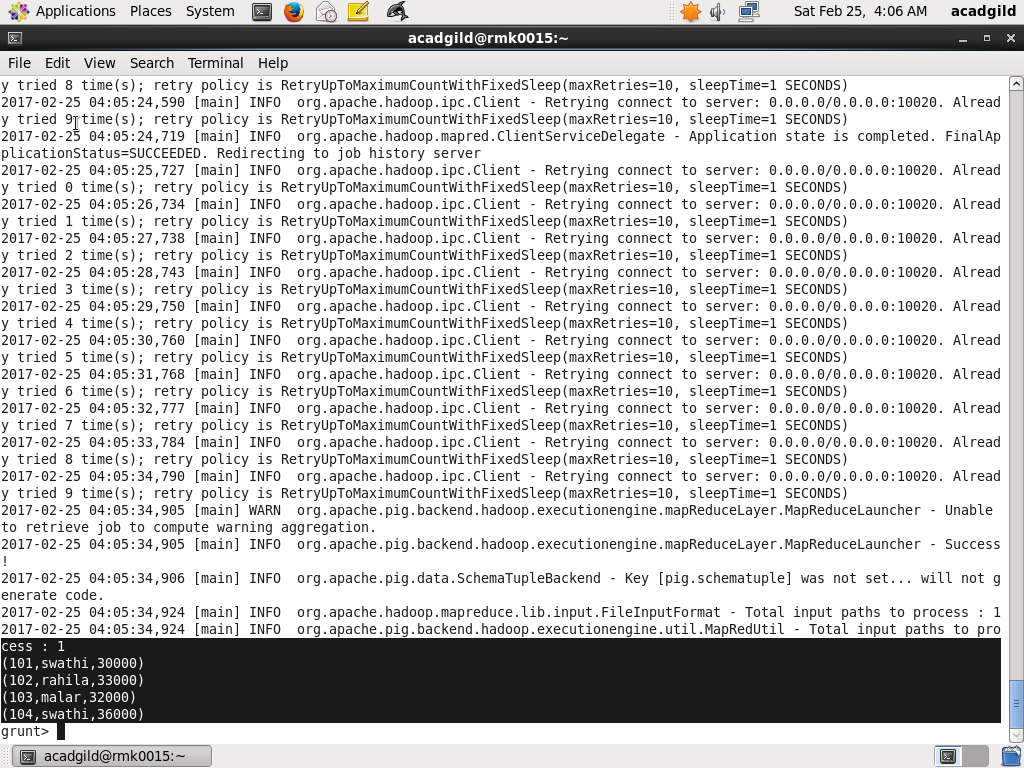
(101,swathi,30000)

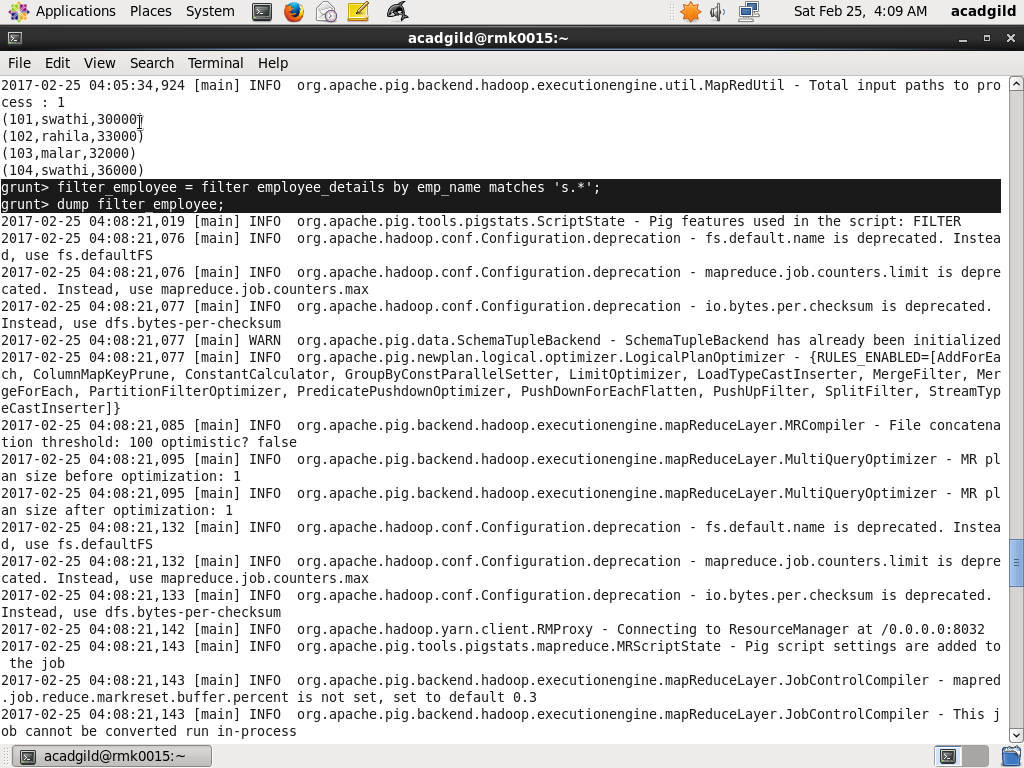
(102,rahila,33000)

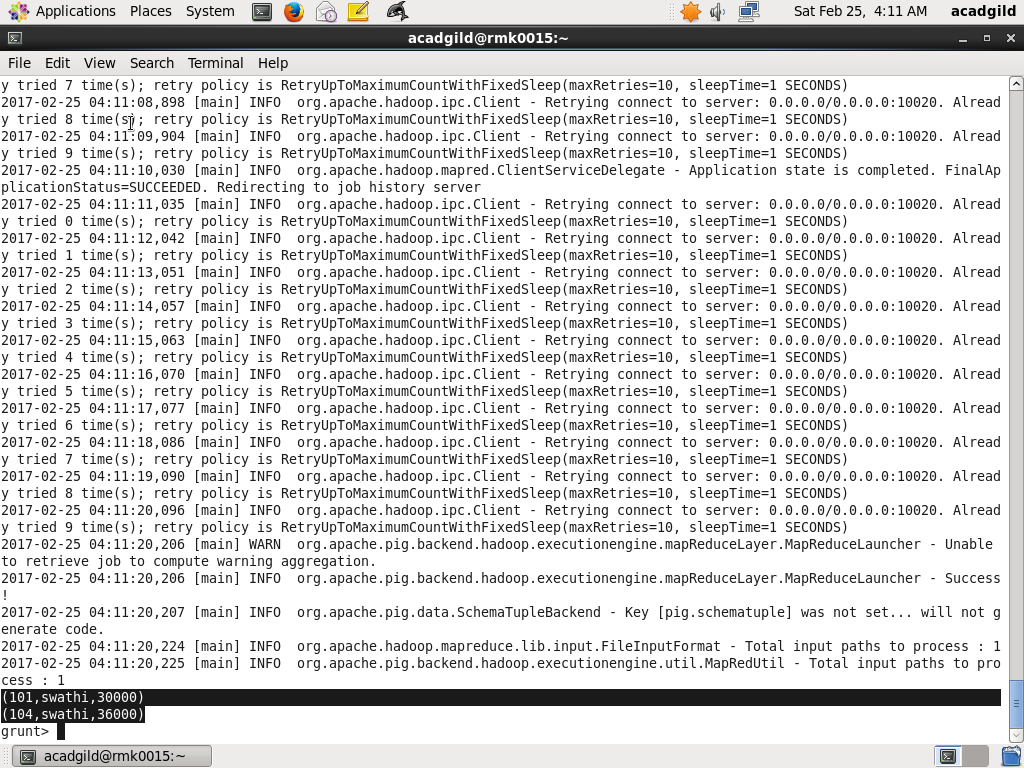
(103,malar,32000)

(104,swathi,36000)









**GROUP BY**

Groups the data in one or multiple relations.

Syntax:

alias = GROUP alias { ALL | BY expression} [, alias ALL | BY expression …] [PARALLEL n];

The GROUP operator groups together tuples that have the same group key .

The key field will be a tuple if the group key has more than one field, otherwise it will be the same type as that of the group key. The result of a GROUP operation is a relation that includes one tuple per group. This tuple contains two fields:

The first field is named group and is the same type as the group key.

The second field takes the name of the original relation and is type bag.

The names of both fields are generated by the system.

Employee\_details= load '/user/acadgild/emp' AS (emp\_id:int,emp\_name:chararray,emp\_salary:int);

DESCRIBE employee\_details;

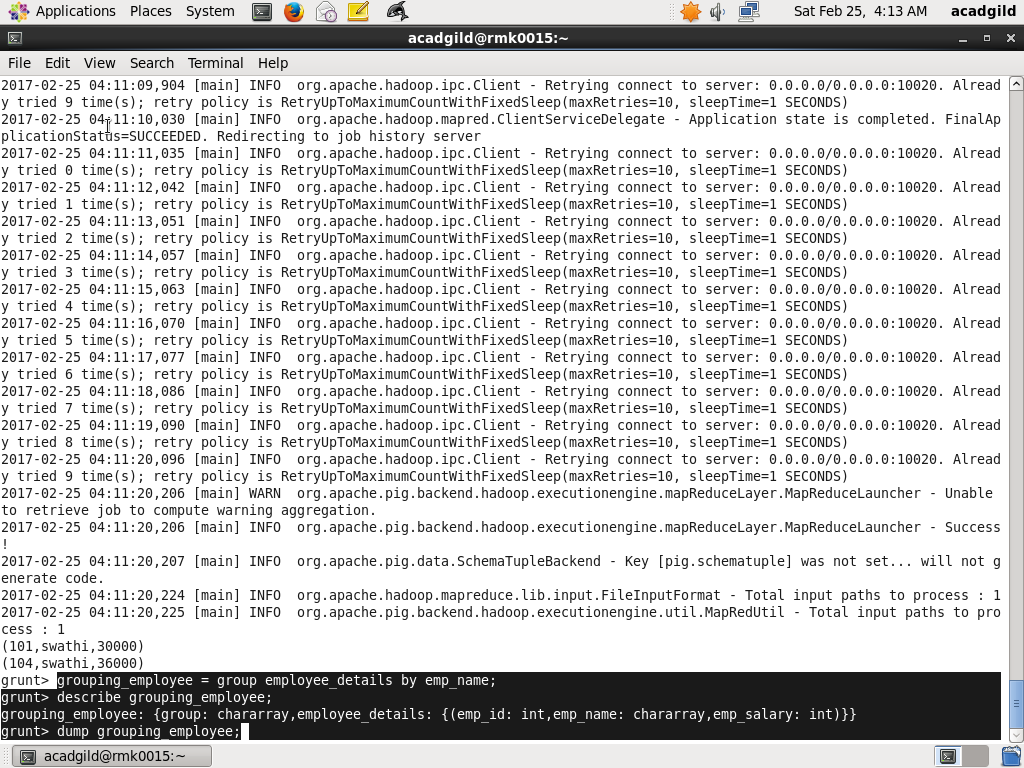
Employee\_details: {emp\_id:int,emp\_name: chararray,emp\_salary: int}

Grouping\_employee= group employee\_details by emp\_name;

Describe grouping\_employee;

Grouping\_employee: {group:chararray,employee\_details: {(emp\_id: int,emp\_name: chararray,emp\_salary: int)}

DUMP grouping\_employee;



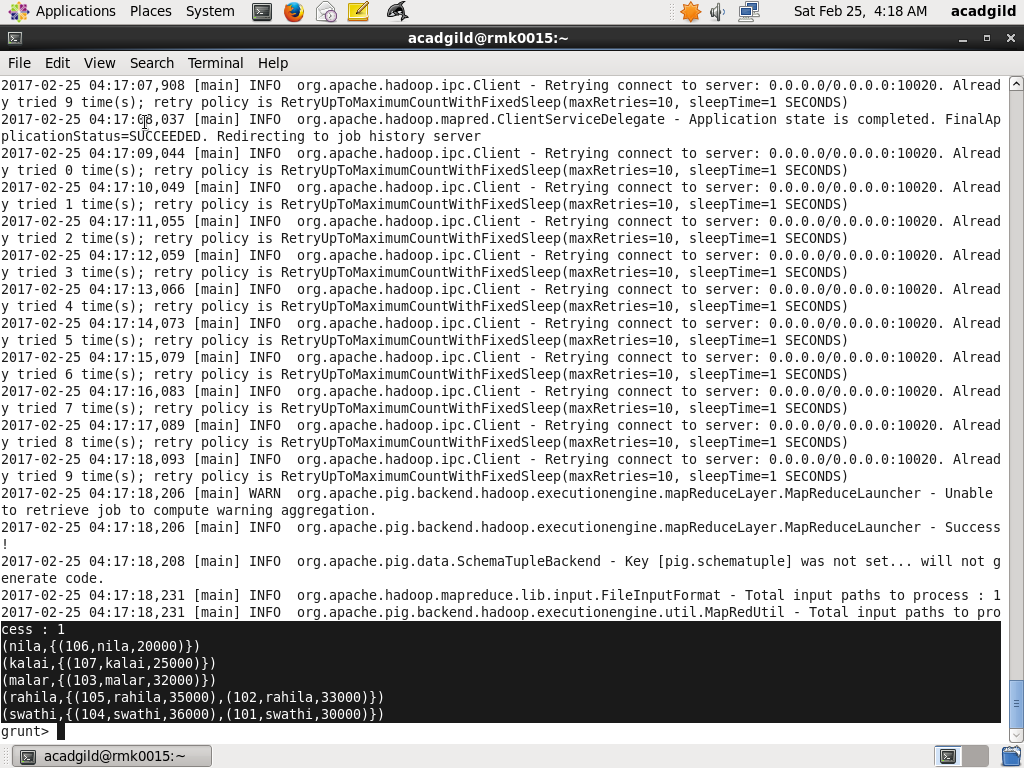
(nila,{(106,nila,20000)})

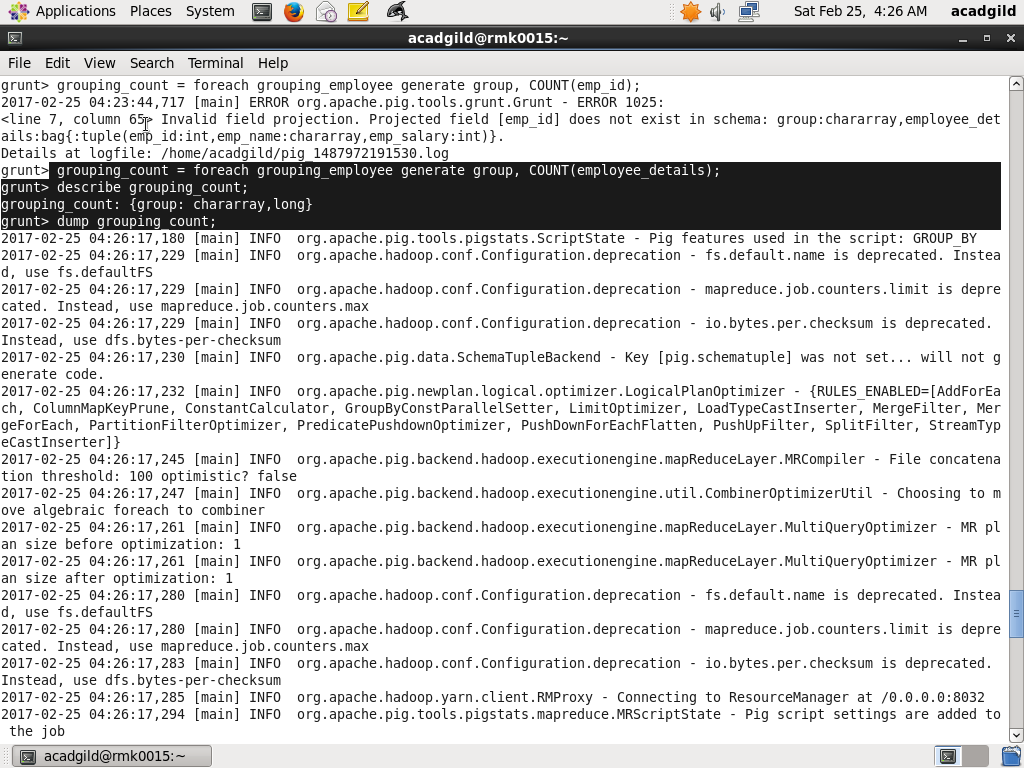
(kalai,{(107,kalai,25000)})

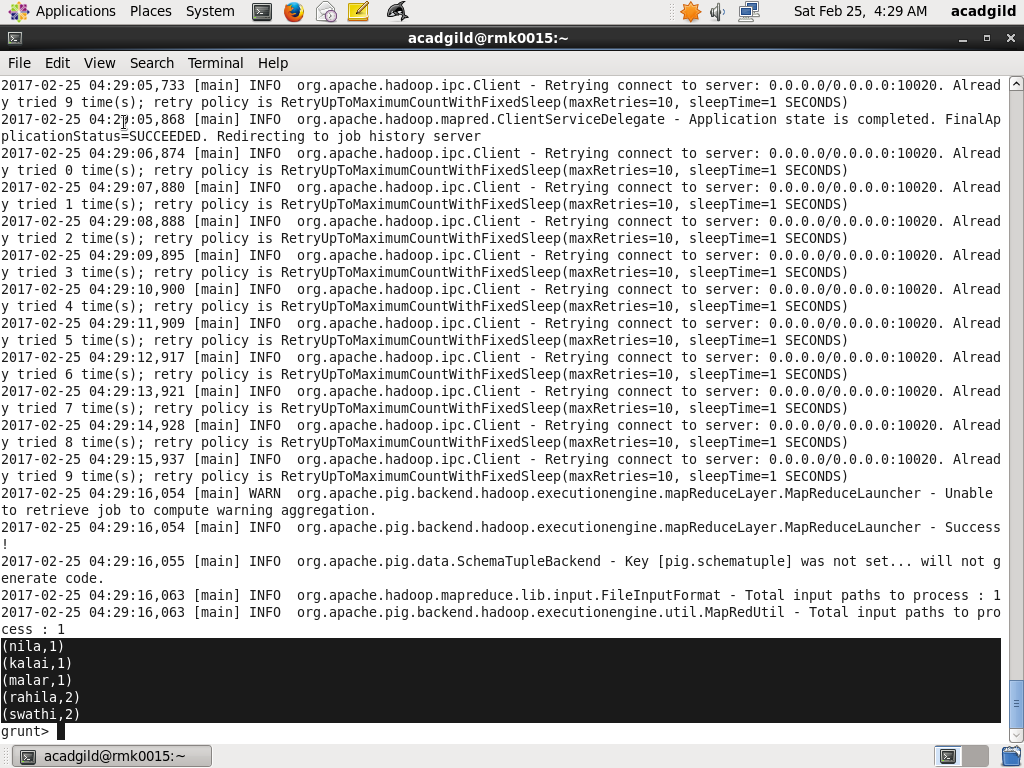
(malar,{(103,malar,32000)})

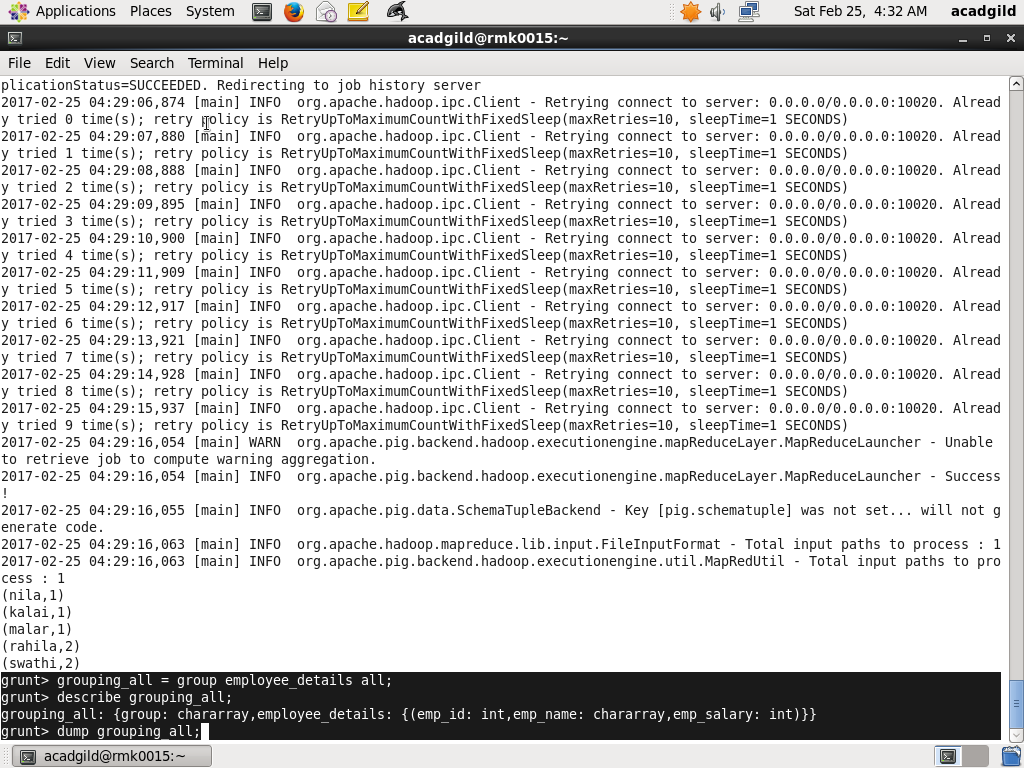
(rahila,{(105,rahila,35000),(102,rahila,33000)})

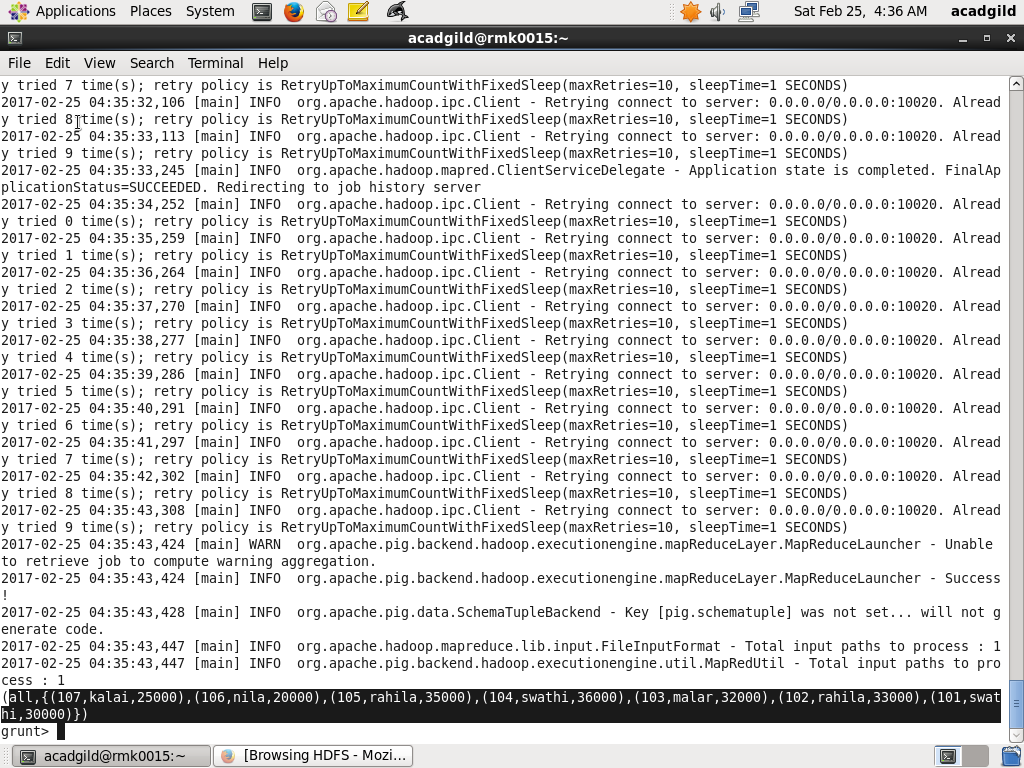
(swathi,{(104,swathi,36000),(101,swathi,30000)})











**ORDER BY**

Sorts a relation based on one or more fields.

Syntax

alias = ORDER alias BY { \* [ASC|DESC] | field\_alias [ASC|DESC] [, field\_alias [ASC|DESC] …] } [PARALLEL n]; Terms alias

The name of a relation.

ASC

Sort in ascending order.

DESC

Sort in descending order.

For example,

Ordering\_employee = order emp\_annual by annual\_salary desc;

Dump ordering\_employee;



(104,432000)

(105,420000)

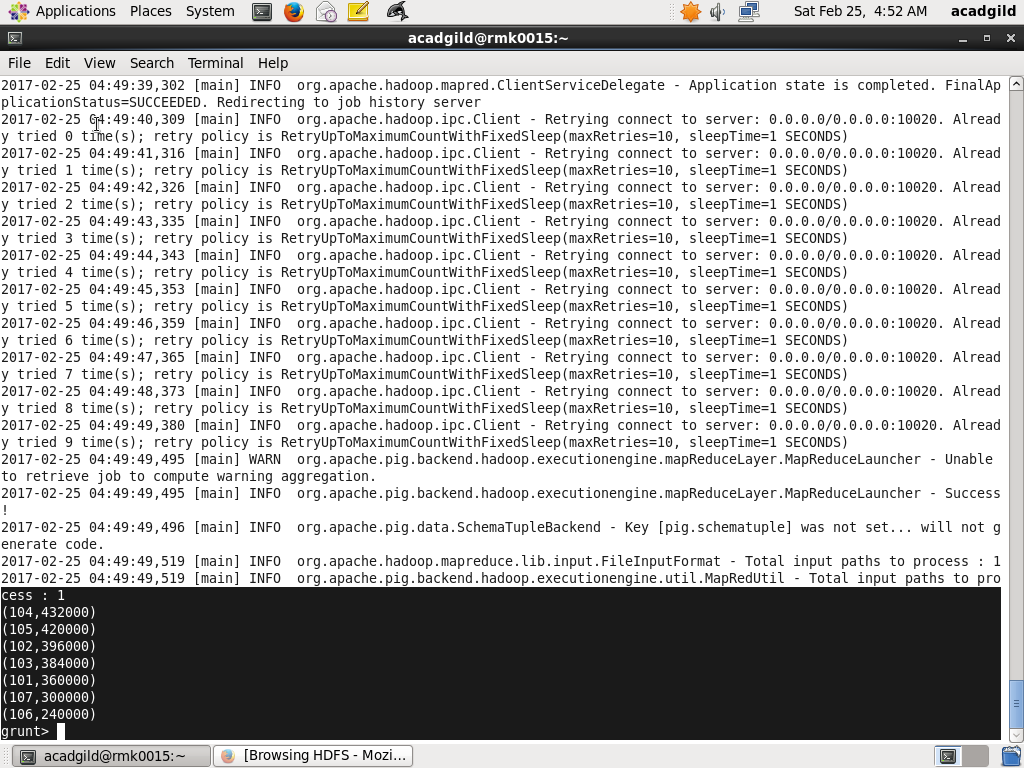
(102,396000)

(103,384000)

(!011,360000)

(107,300000)

(106,240000)



**DESCRIBE**

Returns the schema of an alias.

Syntax

DESCRIBE alias;

For example,

DESCRIBE employee\_details;

