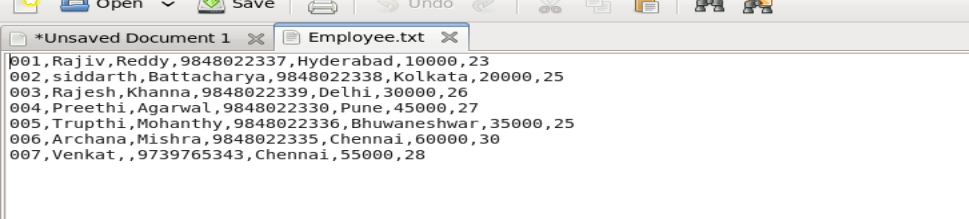
4.2 Pig commands

For practising Pig commands we need a data set.

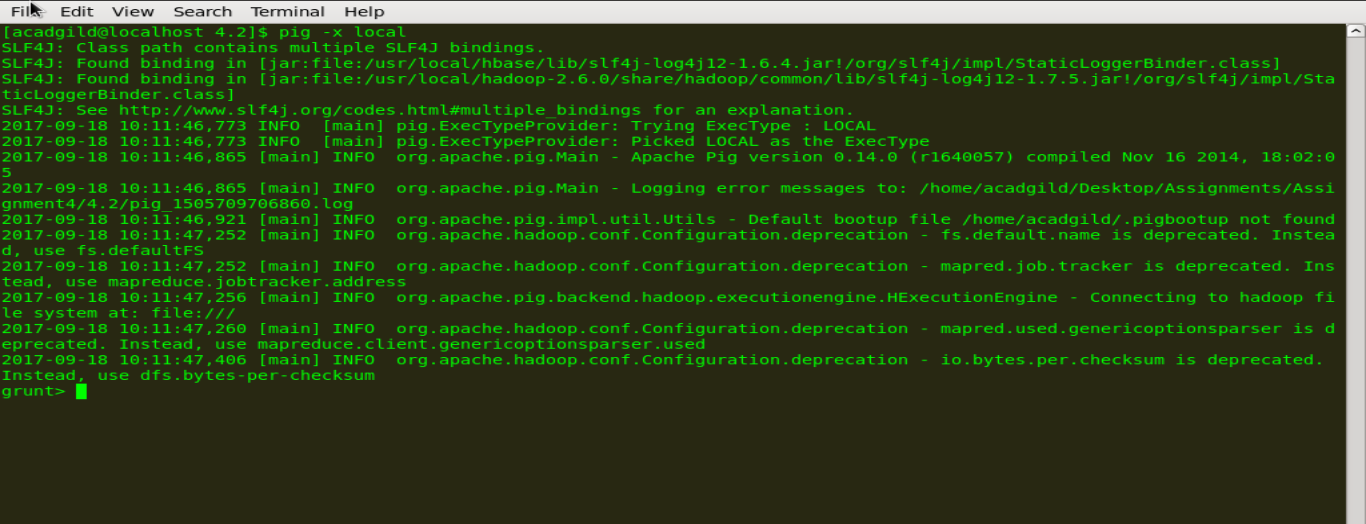
Following student data set is used.



Here I am solving the assignments in local mode

Hence to log into Pig Grunt shell with local mode we need to use command to

Pig –x local

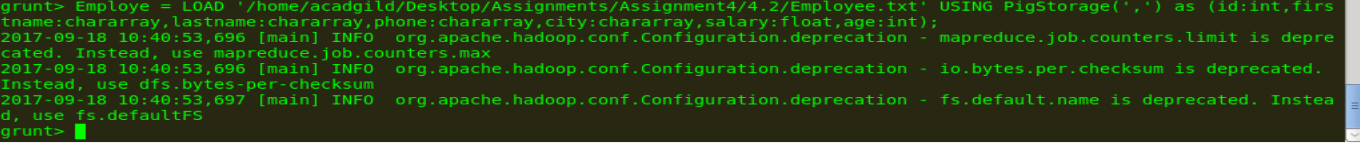


As i am using local mode, dataset can be read from local file system. Else if we are using mapreduce mode we need to put the dataset in hdfs.

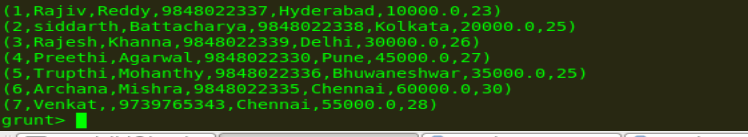
To perform any action on dataset first we need to load the dataset in Pig tables. Here in the dataset we have following columns

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| column | id | firstname | lastname | phone | city | Salary | age |
| datatype | int | char array | char array | char array | char array | float | int |

Loading Data set into pig tables



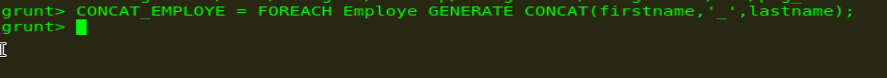
Verify data using DUMP statement



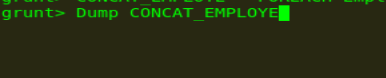
Pig commands:

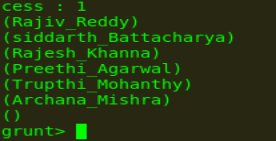
1. Concat

Here concat firstname and last name of employee



Display result

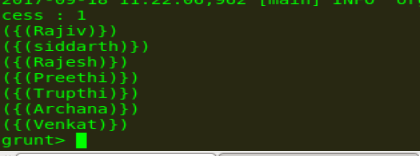




1. Tokenize

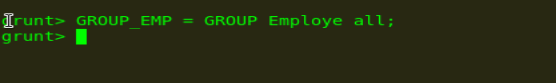
We can use the **TOKENIZE()** function to split a string.

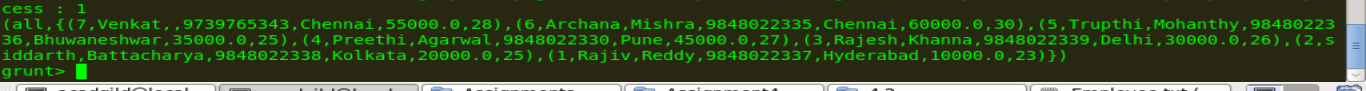




1. Sum

we now need to group our tuples









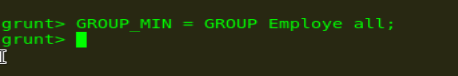


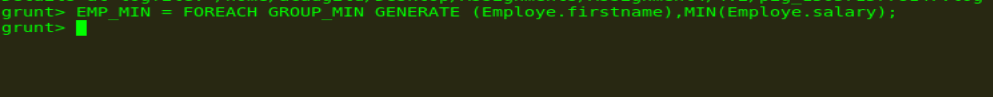
The first thing I tried to do was use the SUM function without grouping my results first but Pig needs you to group your results before trying to SUM a particular field. So, trying to SUM the entire records in a FOREACH will not work.

1. Min

To get the global minimum value, we need to perform a **Group All**operation, and calculate the minimum value using the MIN() function.

To get the minimum value of a group, we need to group it using the **Group By** operator and proceed with the minimum function.





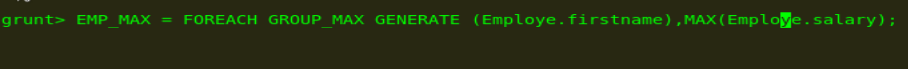


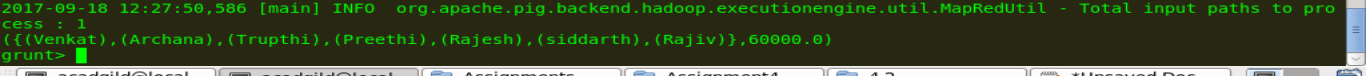
1. Max

To get the global maximum value, we need to perform a **Group All**operation, and calculate the maximum value using the MAX() function.

To get the maximum value of a group, we need to group it using the **Group By** operator and proceed with the maximum function.



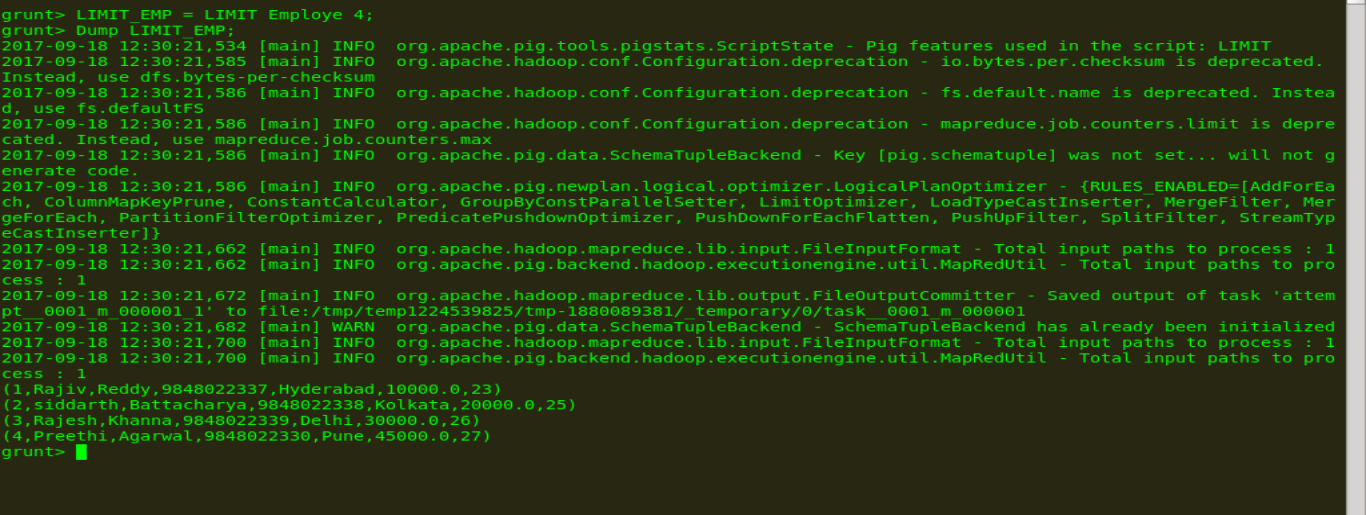




1. Limit

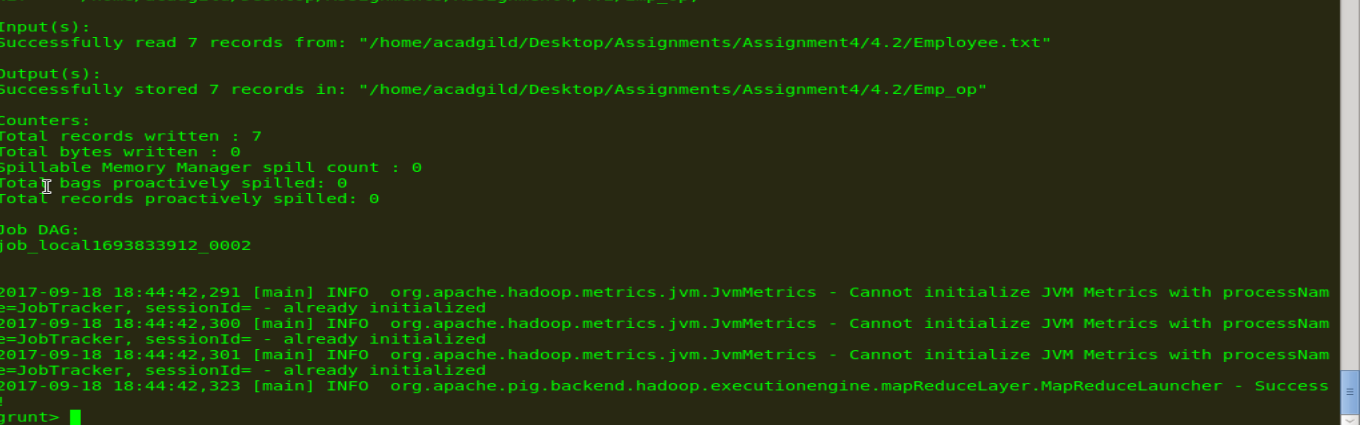
The **LIMIT** operator is used to get a limited number of tuples from a relation

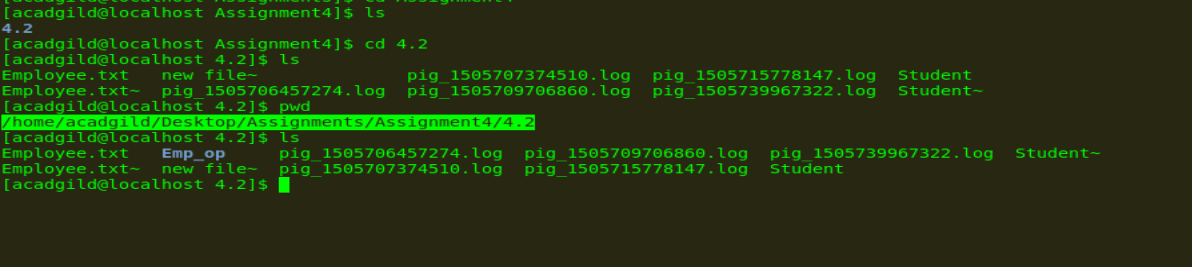


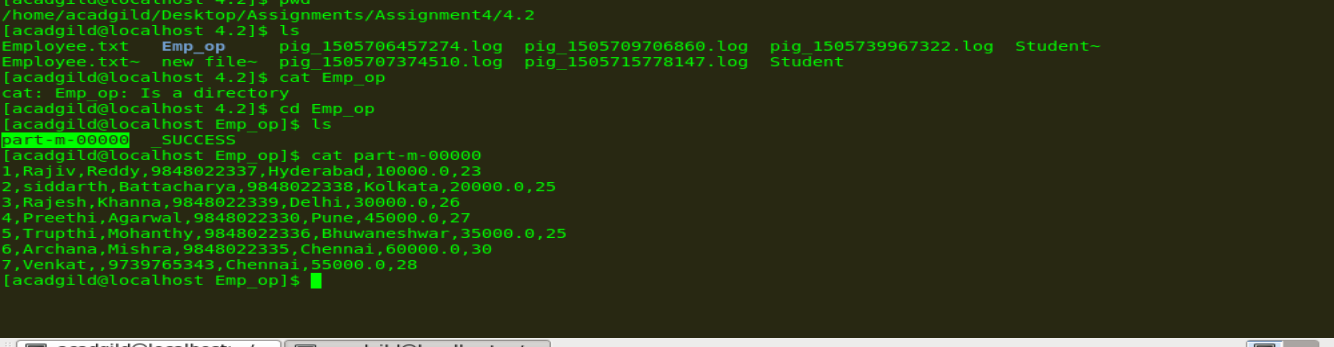


1. Store







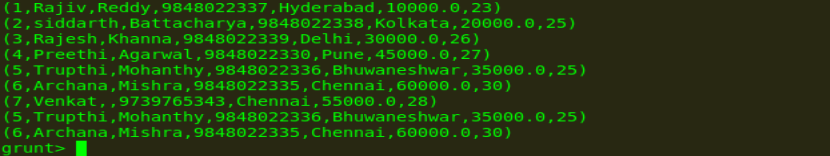


1. Distinct

The **DISTINCT** operator is used to remove redundant (duplicate) tuples from a relation.

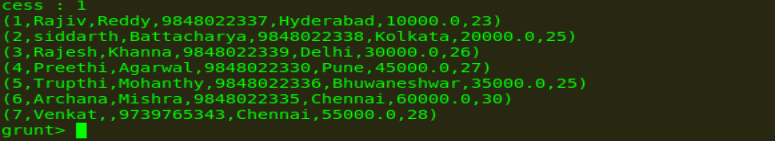
Since the dataset taken doesn’t have duplicate values adding duplicate data

Employe data









Duplicate records are eliminated.

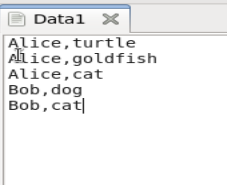
1. Flatten

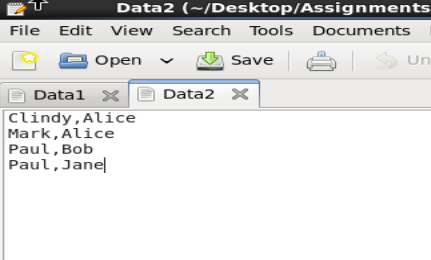
The FLATTEN operator looks like a UDF syntactically, but it is actually an operator that changes the structure of tuples and bags in a way that a UDF cannot. Flatten un-nests tuples as well as bags. The idea is the same, but the operation and result is different for each type of structure.

For tuples, flatten substitutes the fields of a tuple in place of the tuple. For example, consider a relation that has a tuple of the form (a, (b, c)). The expression GENERATE $0, flatten($1), will cause that tuple to become (a, b, c).

For bags, the situation becomes more complicated. When we un-nest a bag, we create new tuples. If we have a relation that is made up of tuples of the form ({(b,c),(d,e)}) and we apply GENERATE flatten($0), we end up with two tuples (b,c) and (d,e). When we remove a level of nesting in a bag, sometimes we cause a cross product to happen. For example, consider a relation that has a tuple of the form (a, {(b,c), (d,e)}), commonly produced by the GROUP operator. If we apply the expression GENERATE $0, flatten($1) to this tuple, we will create new tuples: (a, b, c) and (a, d, e).

To describe Flatten we need two data sets:

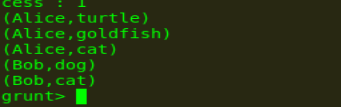


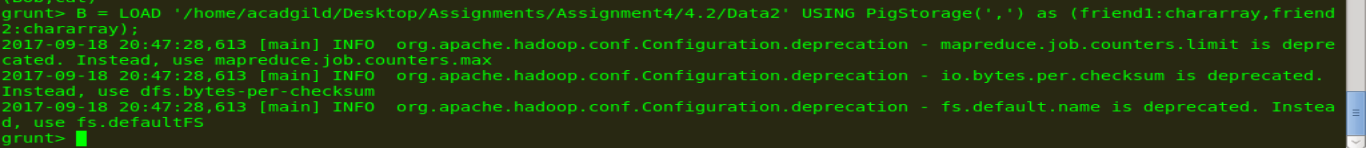


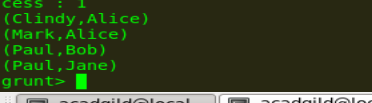
Data1: (owner,pet)

Data 2: (friend1,friend2)



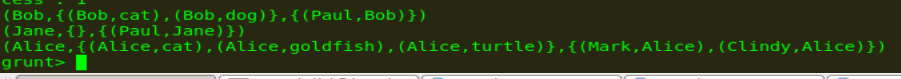






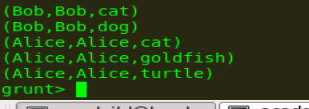






Relation X looks like this. A tuple is created for each unique key field. The tuple includes the key field and two bags. The first bag is the tuples from the first relation with the matching key field. The second bag is the tuples from the second relation with the matching key field. If no tuples match the key field, the bag is empty.





1. IsEmpty

The **IsEmpty()** function of Pig Latin is used to check if a bag or map is empty.

