**PIG DOCUMENTATION**

* By VIGNESH.R

**Apache Pig**

**Apache Pig** is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.

**Key properties:**

* **Ease of programming.** It is trivial to achieve parallel execution of simple, "embarrassingly parallel" data analysis tasks. Complex tasks comprised of multiple interrelated data transformations are explicitly encoded as data flow sequences, making them easy to write, understand, and maintain.
* **Optimization opportunities.** The way in which tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.
* **Extensibility.** Users can create their own functions to do special-purpose processing.

**Apache Pig Vs MapReduce**

|  |  |
| --- | --- |
| **Apache Pig** | **MapReduce** |
| Data flow language. | Data processing paradigm. |
| High level language. | Low level and rigid. |
| Performing a Join is pretty simple. | It is quite difficult to perform a Join operation between datasets. |
| Uses multi-query approach, thereby reducing the length of the codes. | Require almost 20 times more the number of lines to perform the same task. |
| There is no need for compilation. | MapReduce jobs have a long compilation process. |

**Apache Pig Vs Hive**

|  |  |
| --- | --- |
| **Apache Pig** | **Hive** |
| Apache Pig uses a language called Pig Latin. | Hive uses a language called HiveQL. |
| Pig Latin is a data flow language. | HiveQL is a query processing language. |
| Pig Latin is a procedural language. | HiveQL is a declarative language. |
| Apache Pig can handle structured, unstructured, and semi-structured data. | Hive is mostly for structured data. |

**Apache Pig Mode**

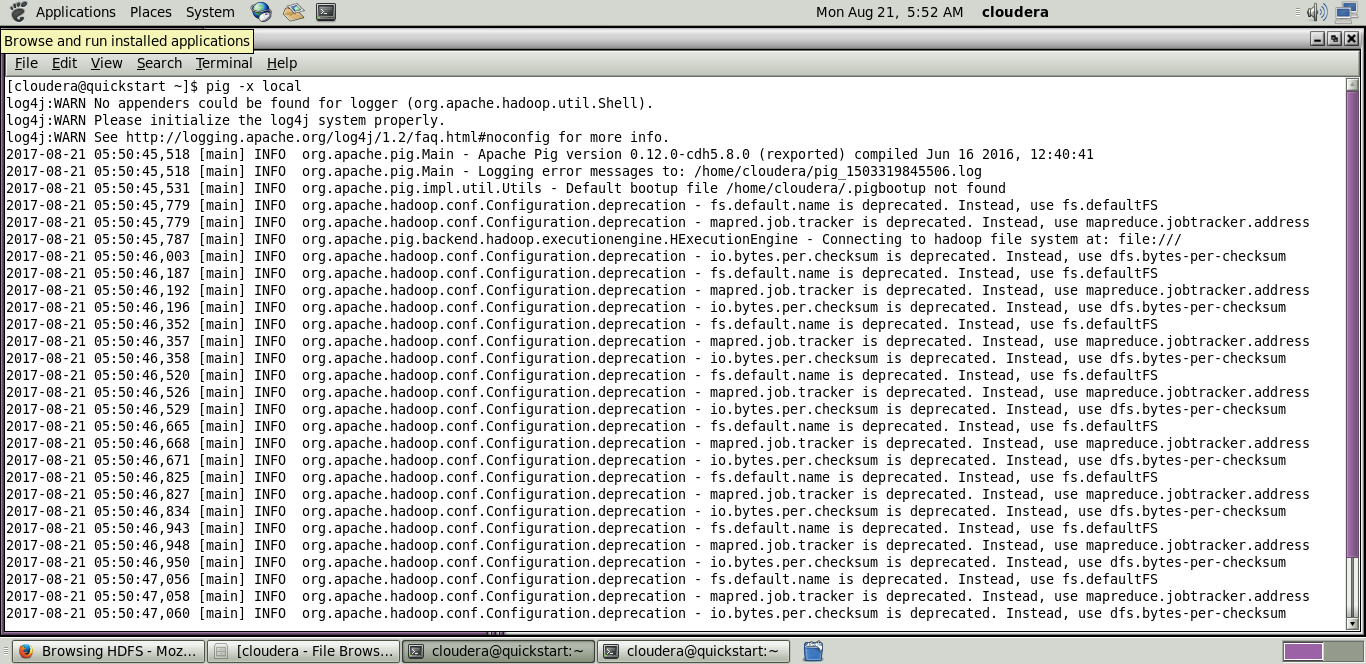
Apache Pig has two modes.

* **Local Mode**  - All the files are installed and run from the local host and local file system.
* **HDFS mode** - MapReduce mode is where we load or process the data that exists in the Hadoop File System (HDFS) using Apache Pig.

**Invoking the Grunt Shell:-**

**Command (Local mode) :** pig -x local

**Command (HDFS) :** pig -x mapreduce



**Apache Pig Execution Mechanisms**

Apache Pig scripts can be executed in three ways, namely:-

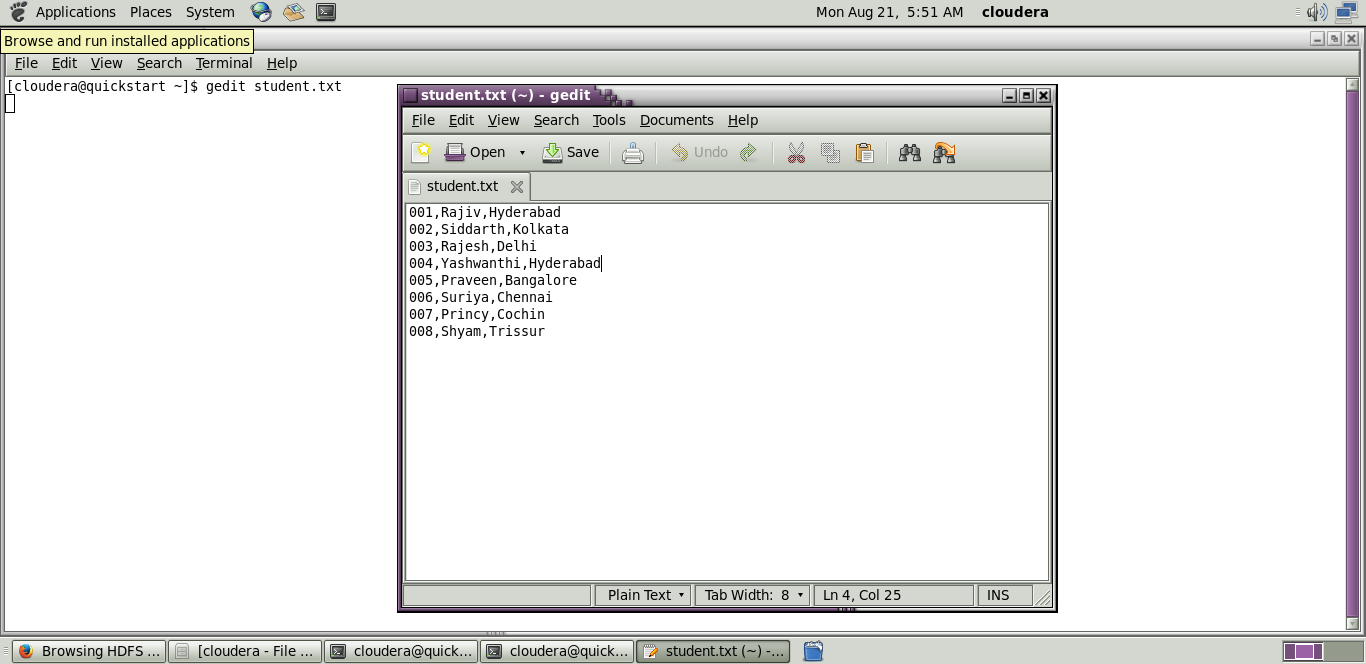
* I**nteractive mode**
* **Batch mode**
* **Embedded mode.**

**Interactive Mode (Grunt shell):-**

After invoking the Grunt shell, you can execute a Pig script by directly entering the Pig Latin statements in it.

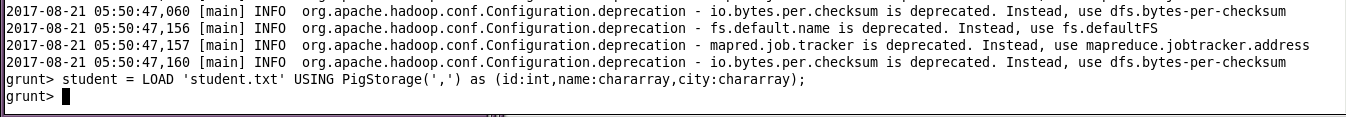
**Step 1:** Create a text file.

**Command:** gedit student.txt



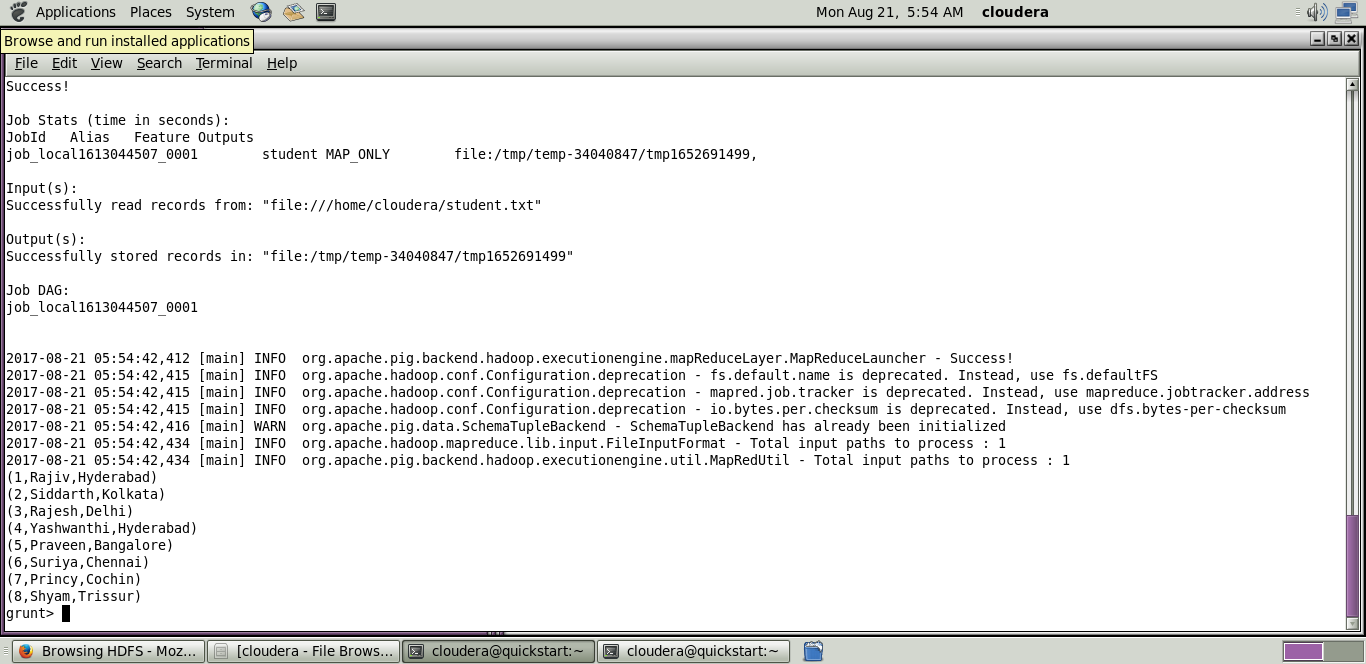
**Step 2:** Load the data into grunt shell

**Command:** student = LOAD 'student.txt' USING PigStorage(',') as (id:int,name:chararray,city:chararray);



**Step 3:** Display the data.

**Command:** Dump student;

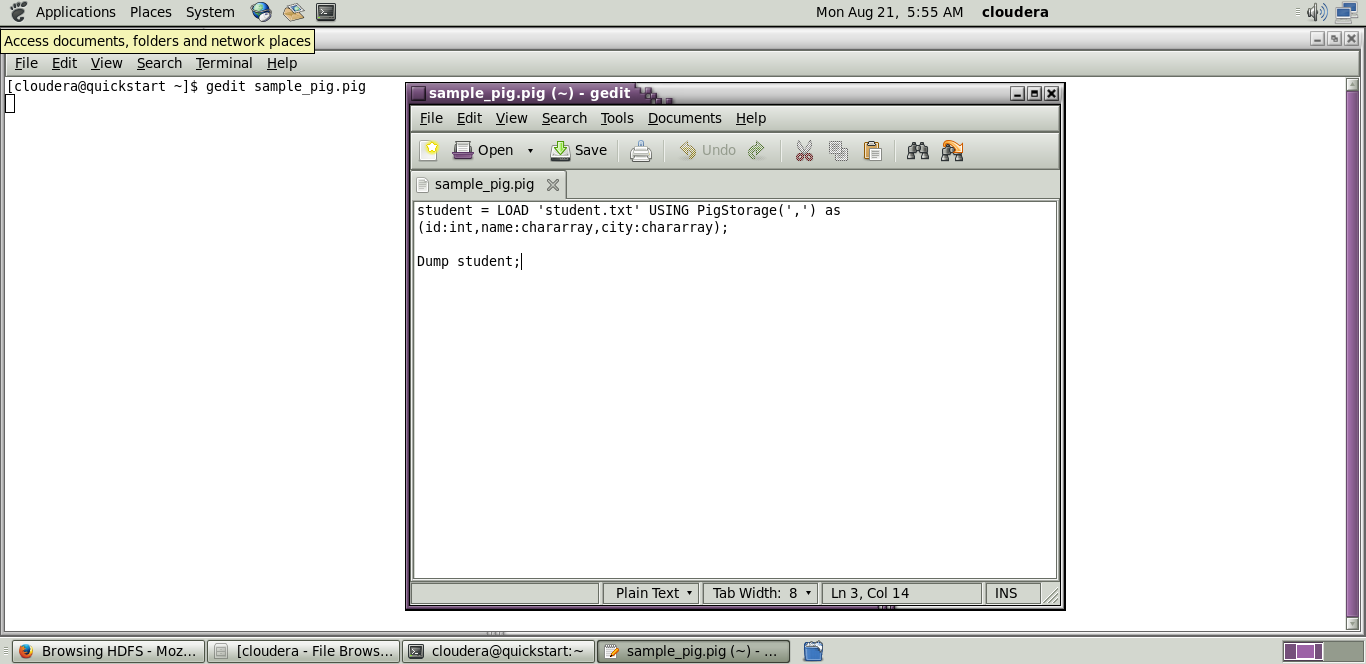


**Batch Mode (Script):-**

We can write an entire Pig Latin script in a file and execute it using the –x command.

**Step 1:** Create a file with .pig extension.

**Command:** gedit sample\_pig.pig



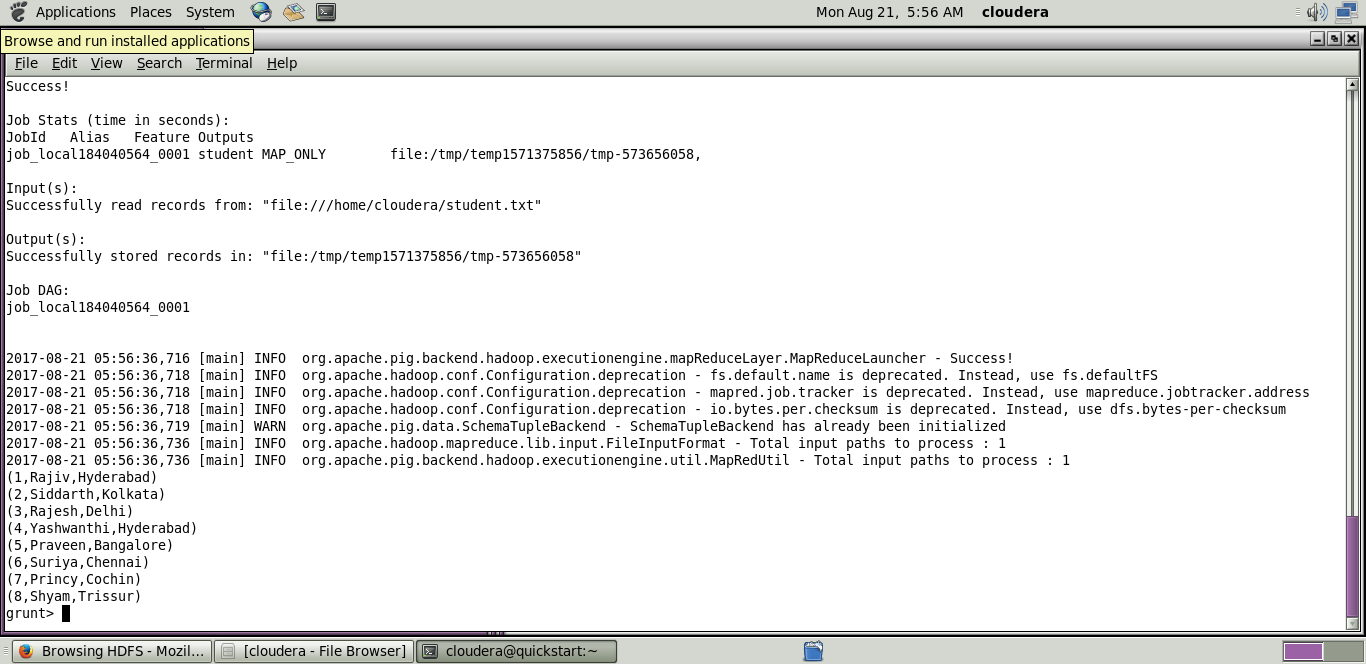
**Step 2:** Paste the following command into that file.

**Command:** student = LOAD 'student.txt' USING PigStorage(',') as (id:int,name:chararray,city:chararray);

Dump student;

**Step 3:** Run from terminal.

**Command:** exec /home/cloudera/sample\_pig.pig;



**Diagnostic Operators**

1. **DUMP** - To print the contents of a relation on the console.

**Command :** DUMP student;

1. **DESCRIBE** - To describe the schema of a relation.

Command : Describe student;

1. **EXPLAIN** - To view the logical, physical, or MapReduce execution plans to compute a relation.

**Command :** Explain student;

1. **ILLUSTRATE** - To view the step-by-step execution of a series of statements.

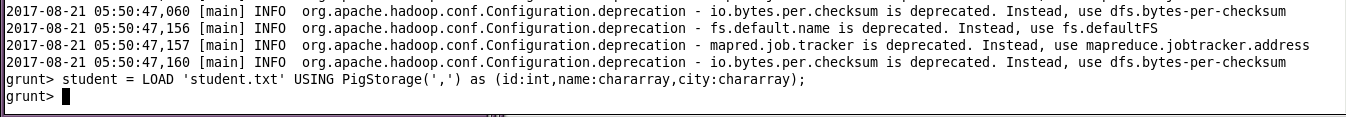
**Command :** Illustrate student;

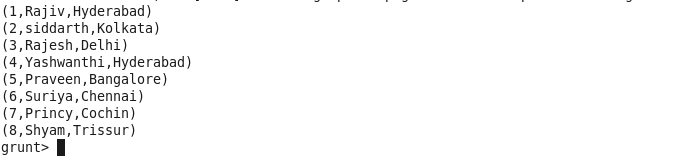
## Pig Latin – Relational Operations

1. **LOAD** - To Load the data from the file system (local/HDFS) into a relation.

**Command** : student = LOAD 'student.txt' USING PigStorage(',') as (id:int,name:chararray,city:chararray);

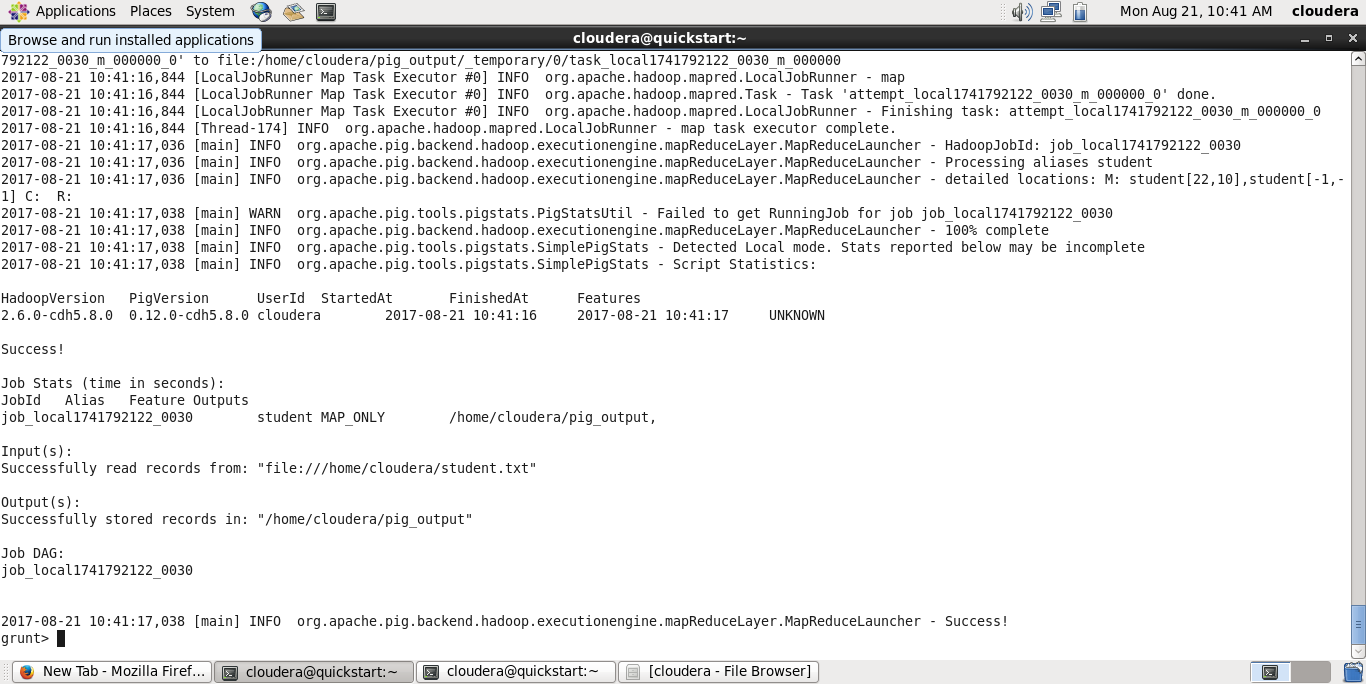
Dump student;





1. **STORE** - To save a relation to the file system (local/HDFS).

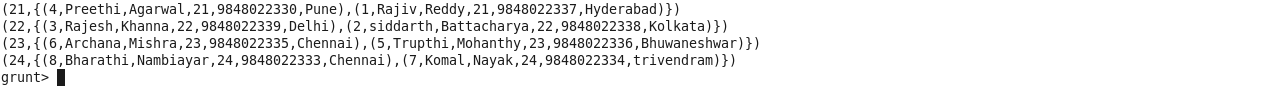
**Command** : STORE student INTO '/home/cloudera/pig\_output' USING PigStorage (',');



1. **GROUP** - To group the data in a single relation.

**Command** : grouping = GROUP student1 by age;

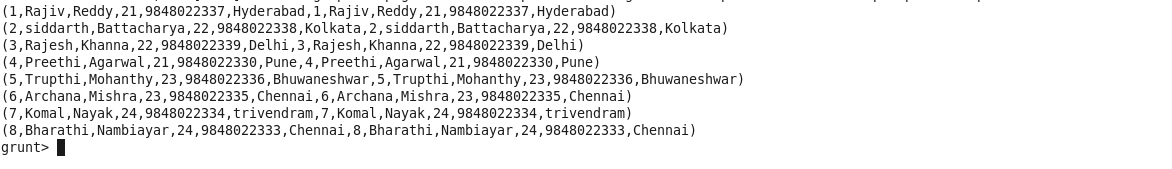
Dump grouping;



1. **JOIN** - To join two or more relations.

**Command** : student3 = JOIN student1 BY id, student2 BY id;

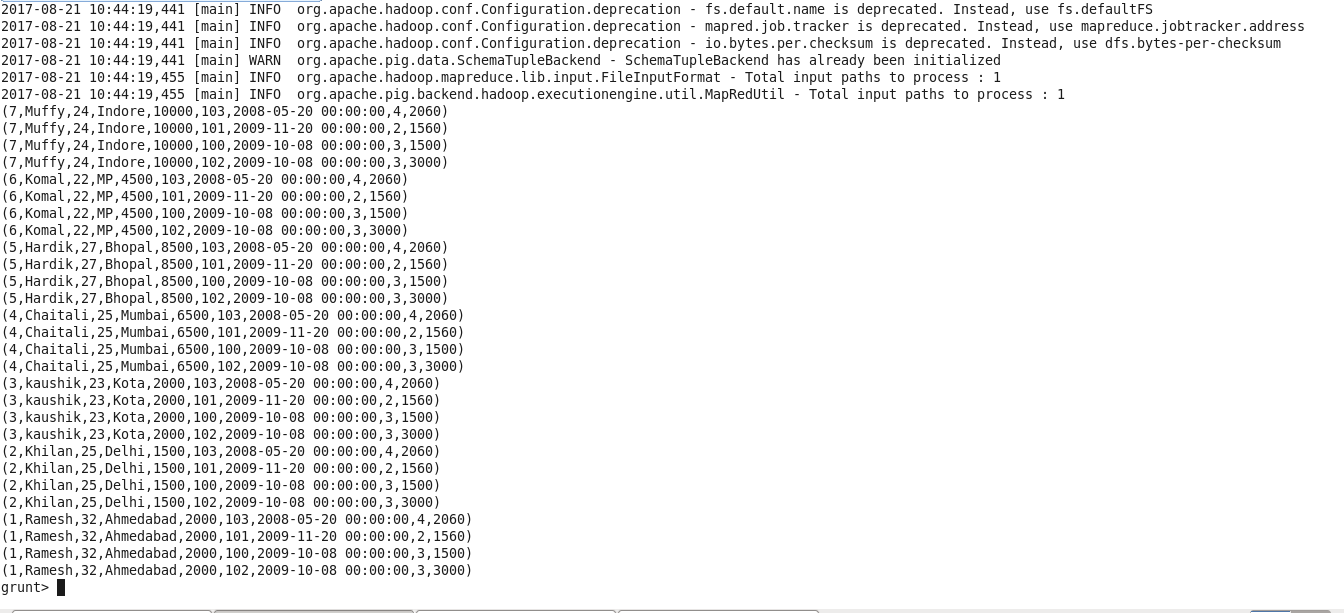
Dump student3;



1. **CROSS** - To create the cross product of two or more relations.

**Command** : cross\_data = CROSS customers, orders;

Dump cross\_data;



1. **UNION** - To combine two or more relations into a single relation.

**Command** : union\_out = UNION student1, student2;

Dump union\_out;

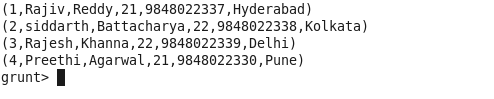


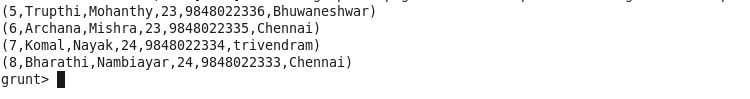
1. **SPLIT** - To split a single relation into two or more relations.

**Command** : SPLIT student1 into student\_details1 if age<23, student\_details2 if (age>=23);

Dump student\_details1;

Dump student\_details2;

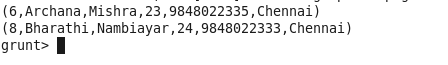




1. **FILTER** - To remove unwanted rows from a relation.

**Command** : filter\_data = FILTER student1 BY city == 'Chennai';

Dump filter\_data;



1. **DISTINCT** - To remove duplicate rows from a relation.

**Command** : student4 = LOAD 'student4.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);

distinct\_data = DISTINCT student4;

Dump distinct\_data;



1. **FOREACH**- To generate data transformations based on columns of data.

**Command** : foreach\_data = FOREACH student1 GENERATE id,age,city;

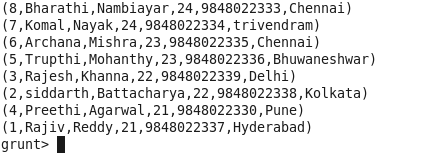
Dump foreach\_data;



1. **ORDER** - To arrange a relation in a sorted order based on one or more fields (ascending or descending).

**Command** : order\_by\_data = ORDER student1 BY age DESC;

Dump order\_by\_data;



1. **LIMIT** - To get a limited number of tuples from a relation.

**Command** : limit\_data = LIMIT student1 4;

Dump limit\_data;

