SMIT R PATEL

19162121031

SEM 5

Practical 8

Big Data and Analytics

AIM- To understand operators in Apache Pig.

Join Operator

The **JOIN** operator is used to combine records from two or more relations. While performing a join operation, we declare one (or a group of) tuple(s) from each relation, as keys. When these keys match, the two particular tuples are matched, else the records are dropped. Joins can be of the following types −

- Self-join
- Inner-join
- Outer-join − left join, right join, and full join

This chapter explains with examples how to use the join operator in Pig Latin. Assume that we have two files namely **customers.txt** and **orders.txt** in the **/pig_data/** directory of HDFS as shown below.

customers.txt

```
1,Ramesh,32,Ahmedabad,2000.00
2,Khilan,25,Delhi,1500.00
3,kaushik,23,Kota,2000.00
4,Chaitali,25,Mumbai,6500.00
5,Hardik,27,Bhopal,8500.00
6,Komal,22,MP,4500.00
7,Muffy,24,Indore,10000.00
```

```
cloudera@quickstart:~/Desktop
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart Desktop]$ cat customers.txt
cat: customers.txt: No such file or directory
[cloudera@quickstart Desktop]$ cat customers.txt
1,Ramesh,32,Ahmedabad,2000.00
2,Khilan,25,Delhi,1500.00
3,kaushik,23,Kota,2000.00
4,Chaitali,25,Mumbai,6500.00
5,Hardik,27,Bhopal,8500.00
6,Komal,22,MP,4500.00
7,Muffy,24,Indore,10000.00
[cloudera@quickstart Desktop]$ cat orders.txt
102,2009-10-08 00:00:00,3,3000
100,2009-10-08 00:00:00,3,1500
101,2009-11-20 00:00:00,2,1560
103,2008-05-20 00:00:00,4,2060
[cloudera@quickstart Desktop]$
```

orders.txt

102,2009-10-08 00:00:00,3,3000

100,2009-10-08 00:00:00,3,1500

101,2009-11-20 00:00:00,2,1560

103,2008-05-20 00:00:00,4,2060


```
cloudera@quickstart:~/Desktop
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart Desktop]$ cat customers.txt
cat: customers.txt: No such file or directory
[cloudera@quickstart Desktop]$ cat customers.txt
1,Ramesh,32,Ahmedabad,2000.00
2,Khilan,25,Delhi,1500.00
3,kaushik,23,Kota,2000.00
4,Chaitali,25,Mumbai,6500.00
5,Hardik,27,Bhopal,8500.00
6,Komal,22,MP,4500.00
7,Muffy,24,Indore,10000.00
[cloudera@quickstart Desktop]$ cat orders.txt
102,2009-10-08 00:00:00,3,3000
100,2009-10-08 00:00:00,3,1500
101,2009-11-20 00:00:00,2,1560
103,2008-05-20 00:00:00,4,2060
[cloudera@quickstart Desktop]$
```

And we have loaded these two files into Pig with the relations customers and orders as shown below.

```
[cloudera@quickstart Desktop]$ hadoop fs -mkdir smitrpatel
[cloudera@quickstart Desktop]$ hadoop fs -put customers.txt smitrpatel
[cloudera@quickstart Desktop]$ hadoop fs -put orders.txt smitrpatel
[cloudera@quickstart Desktop]$ hadoop fs -ls smitrpatel
Found 2 items
-rw-r--r--   1 cloudera cloudera        194 2021-08-25 17:32 smitrpatel/customer
s.txt
-rw-r--r--   1 cloudera cloudera        124 2021-08-25 17:32 smitrpatel/orders.t
xt
[cloudera@quickstart Desktop]$
```

And we have loaded these two files into Pig with the relations **customers** and **orders** as shown below.

```
grunt> customers = LOAD
'hdfs://localhost:9000/pig_data/customers.txt'
USING  PigStorage(',')
 as (id:int, name:chararray, age:int, address:chararray,
salary:int);


grunt> orders = LOAD
'hdfs://localhost:9000/pig_data/orders.txt' USING
PigStorage(',')
 as (oid:int, date:chararray, customer_id:int, amount:int);
```

Commands:

```
customers = load'smitrpatel/customers.txt' using PigStorage(',')
as (id:int, name:chararray, salary:int);
```

```
grunt> customers = load'smitrpatel/customers.txt'
>> using PigStorage(',') as
>> (id:int, name:chararray, age:int, address:chararray, salary:int);
grunt> orders = load 'smitrpatel/orders.txt'
>> using PigStorage(',') as
>> (id:int, date:chararray, customer_id:int, amount:int);
grunt>
```

Let us now perform various Join operations on these two relations.

Self - join

Self-join is used to join a table with itself as if the table were two relations, temporarily renaming at least one relation.

Generally, in Apache Pig, to perform self-join, we will load the same data multiple times, under different aliases (names). Therefore let us load the contents of the file **customers.txt** as two tables as shown below.

```
grunt> customers1 = LOAD
'hdfs://localhost:9000/pig_data/customers.txt'
USING PigStorage(',')
 as (id:int, name:chararray, age:int, address:chararray,
salary:int);


grunt> customers2 = LOAD
'hdfs://localhost:9000/pig_data/customers.txt'
USING PigStorage(',')
 as (id:int, name:chararray, age:int, address:chararray,
salary:int);
```

Command :

grunt> customers = load 'smitrpatel/customers.txt' using PigStorage(',') as (id:int, name:chararray, age:int, address:chararray, salary:int);

grunt> orders = load 'smitrpatel/orders.txt' using PigStorage(',') as (id:int, date:chararray, customer_id:int, amount:int);

grunt> customers1 = load'smitrpatel/customers.txt' using PigStorage(',') as (id:int, name:chararray, age:int, address:chararray, salary:int);

grunt> customers2 = load'smitrpatel/customers.txt' using PigStorage(',') as (id:int, name:chararray, age:int, address:chararray, salary:int);

```
grunt> customers = load 'smitrpatel/customers.txt' using PigStorage(',') as (id:
int, name:chararray, age:int, address:chararray, salary:int);
grunt> orders = load 'smitrpatel/orders.txt' using PigStorage(',') as (id:int, d
ate:chararray, customer_id:int, amount:int);
grunt> customers1 = load'smitrpatel/customers.txt' using PigStorage(',') as (id:
int, name:chararray, age:int, address:chararray, salary:int);
grunt> customers2 = load'smitrpatel/customers.txt' using PigStorage(',') as (id:
int, name:chararray, age:int, address:chararray, salary:int);
```

Syntax

Given below is the syntax of performing **self-join** operation using the **JOIN** operator.

```
grunt> Relation3_name = JOIN Relation1_name BY key,
Relation2_name BY key ;
```

Example

Let us perform **self-join** operation on the relation **customers**, by joining the two relations **customers1** and **customers2** as shown below.

```
grunt> customers3 = JOIN customers1 BY id, customers2 BY id;
```

Command: customer3 = JOIN customers1 BY id, customers2 BY id;

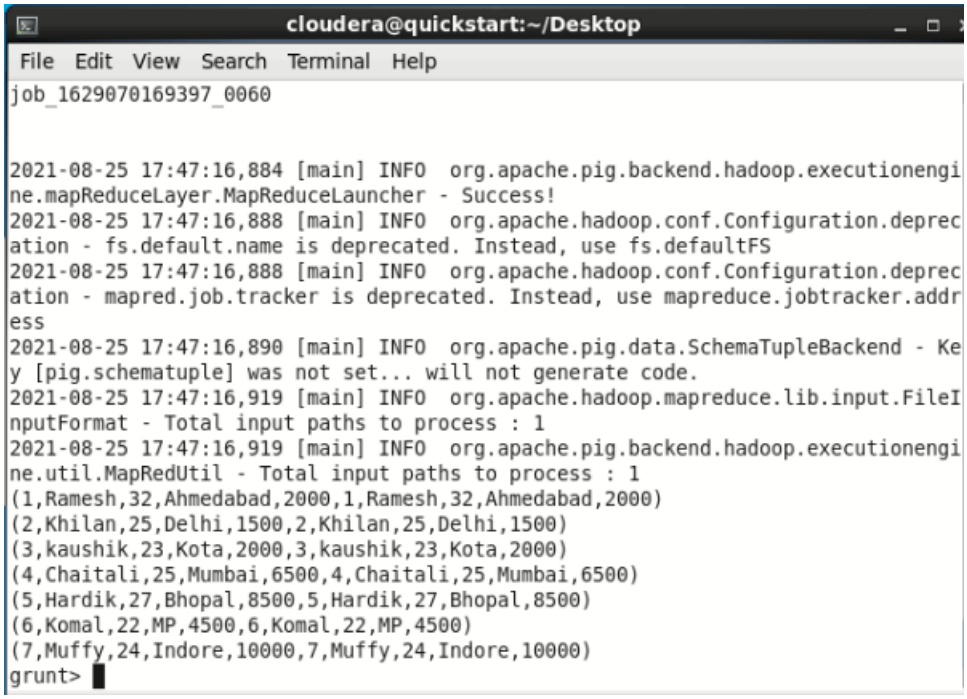
```
grunt> customers = load 'smitrpatel/customers.txt' using PigStorage(',') as (id:
int, name:chararray, age:int, address:chararray, salary:int);
grunt> orders = load 'smitrpatel/orders.txt' using PigStorage(',') as (id:int, d
ate:chararray, customer_id:int, amount:int);
grunt> customers1 = load'smitrpatel/customers.txt' using PigStorage(',') as (id:
int, name:chararray, age:int, address:chararray, salary:int);
grunt> customers2 = load'smitrpatel/customers.txt' using PigStorage(',') as (id:
int, name:chararray, age:int, address:chararray, salary:int);
grunt>
grunt> customer3 = JOIN customers1 BY id, customers2 BY id;
grunt>
```

Verification

Verify the relation **customers3** using the **DUMP** operator as shown below.

```
grunt> Dump customers3;
```

Command : dump customer3



Output

It will produce the following output, displaying the contents of the relation **customers**.

```
(1,Ramesh,32,Ahmedabad,2000,1,Ramesh,32,Ahmed
abad,2000)
(2,Khilan,25,Delhi,1500,2,Khilan,25,Delhi,150
0)

(3,kaushik,23,Kota,2000,3,kaushik,23,Kota,2000)

(4,Chaitali,25,Mumbai,6500,4,Chaitali,25,Mumbai,6500)

(5,Hardik,27,Bhopal,8500,5,Hardik,27,Bhopal,8500)

(6,Komal,22,MP,4500,6,Komal,22,MP,4500)

(7,Muffy,24,Indore,10000,7,Muffy,24,Indore,10000)
```

Inner Join

Inner Join is used quite frequently; it is also referred to as **equijoin**. An inner join returns rows when there is a match in both tables.

It creates a new relation by combining column values of two relations (say A and B) based upon the join-predicate. The query compares each row of A with each row of B to find all pairs of rows which satisfy the join-predicate. When the join-predicate is satisfied, the column values for each matched pair of rows of A and B are combined into a result row.

Syntax

Here is the syntax of performing **inner join** operation using the **JOIN** operator.

```
grunt> result = JOIN relation1 BY columnname,
relation2 BY  columnname;
```

Example

Let us perform **inner join** operation on the two relations **customers** and **orders** as shown below.

```
grunt> coustomer_orders = JOIN customers BY id, orders
BY  customer_id;
```

```
grunt> result = JOIN customers BY id, orders BY customer_id;
grunt>
```

Verification

Verify the relation **coustomer_orders** using the **DUMP** operator as shown below.

```
grunt> Dump coustomer_orders;
```

```
pRedUtil - Total input paths to process : 1
(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)
(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)
grunt>
```

Output

You will get the following output that will the contents of the relation named **coustomer_orders**.

```
(2,Khilan,25,Delhi,1500,101,2009-11-20
00:00:00,2,1560)  (3,kaushik,23,Kota,2000,100,2009-
10-08                              00:00:00,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08
00:00:00,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20
00:00:00,4,2060)
```

Note –

Outer Join: Unlike inner join, **outer join** returns all the rows from at least one of the relations. An outer join operation is carried out in three ways –

- Left outer join
- Right outer join
- Full outer join

Left Outer Join

The **left outer Join** operation returns all rows from the left table, even if there are no matches in the right relation.

Syntax

Given below is the syntax of performing **left outer join** operation using the **JOIN** operator.

```
grunt> Relation3_name = JOIN Relation1_name BY id LEFT
OUTER,  Relation2_name BY customer_id;
```

Example

Let us perform left outer join operation on the two relations customers and orders as shown below.

```
grunt> outer_left = JOIN customers BY id LEFT OUTER, orders BY
customer_id;
```

```
grunt> outer_left = JOIN customers BY id LEFT OUTER, orders BY customer_id;
grunt>
```

Verification

Verify the relation **outer_left** using the **DUMP** operator as shown below.

```
grunt> Dump outer_left;
```

```
pRedUtil - Total input paths to process : 1
(1,Ramesh,32,Ahmedabad,2000,,,,)
(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)
(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)
(5,Hardik,27,Bhopal,8500,,,,)
(6,Komal,22,MP,4500,,,,)
(7,Muffy,24,Indore,10000,,,,)
grunt>
```

Output

It will produce the following output, displaying the contents of the relation **outer_left**.

(1,Ramesh,32,Ahmedabad,2000,,,,)

(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560) (3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)

(5,Hardik,27,Bhopal,8500,,,,)

(6,Komal,22,MP,4500,,,,)

(7,Muffy,24,Indore,10000,,,,)

Right Outer Join

The **right outer join** operation returns all rows from the right table, even if there are no matches in the left table.

Syntax

Given below is the syntax of performing **right outer join** operation using the **JOIN** operator.

```
grunt> outer_right = JOIN customers BY id RIGHT,
orders BY  customer_id;
```

Example

Let us perform **right outer join** operation on the two relations **customers** and **orders** as shown below.

```
grunt> outer_right = JOIN customers BY id RIGHT, orders BY
customer_id;
```

```
cloudera@quickstart:~/Desktop
File  Edit  View  Search  Terminal  Help
grunt> outer_rift = JOIN customers BY id RIGHT, orders BY customer_id;
grunt>
```

Verification

Verify the relation **outer_right** using the **DUMP** operator as shown below.

```
grunt> Dump outer_right
```

```
pRedUtil - Total input paths to process : 1
(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)
(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)
grunt>
```

Output

It will produce the following output, displaying the contents of the relation **outer_right**.

(2,Khilan,25,Delhi,1500,101,2009-11-20
00:00:00,2,1560) (3,kaushik,23,Kota,2000,100,2009-
10-08 00:00:00,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08
00:00:00,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20
00:00:00,4,2060)

Full Outer Join

The **full outer join** operation returns rows when there is a match in one of the relations.

Syntax

Given below is the syntax of performing **full outer join** using the **JOIN** operator.

```
grunt> outer_full = JOIN customers BY id FULL OUTER,
orders BY  customer_id;
```

Example

Let us perform **full outer join** operation on the two relations **customers** and **orders** as shown below.

```
grunt> outer_full = JOIN customers BY id FULL OUTER, orders BY
customer_id;
```

```
cloudera@quickstart:~/Desktop
File  Edit  View  Search  Terminal  Help
grunt> outer_full = JOIN customers BY id FULL OUTER, orders BY customer_id;
grunt>
```

Verification

Verify the relation **outer_full** using the **DUMP** operator as shown below.

```
grun> Dump outer_full;
```

```
pRedUtil - Total input paths to process : 1
(1,Ramesh,32,Ahmedabad,2000,,,,)
(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)
(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)
(5,Hardik,27,Bhopal,8500,,,,)
(6,Komal,22,MP,4500,,,,)
(7,Muffy,24,Indore,10000,,,,)
grunt>
```

Output

It will produce the following output, displaying the contents of the relation **outer_full**.

(1,Ramesh,32,Ahmedabad,2000,,,,)

(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560) (3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)

(5,Hardik,27,Bhopal,8500,,,,)

(6,Komal,22,MP,4500,,,,)

(7,Muffy,24,Indore,10000,,,,)

Using Multiple Keys

We can perform JOIN operation using multiple keys.

Syntax

Here is how you can perform a JOIN operation on two tables using multiple keys.

```
grunt> Relation3_name = JOIN Relation2_name BY (key1,
key2),  Relation3_name BY (key1, key2);
```

Assume that we have two files namely **employee.txt** and **employee_contact.txt** in the **/pig_data/** directory of HDFS as shown below.

employee.txt

```
001,Rajiv,Reddy,21,programmer,003
002,siddarth,Battacharya,22,programmer,003
003,Rajesh,Khanna,22,programmer,003
004,Preethi,Agarwal,21,programmer,003
005,Trupthi,Mohanthy,23,programmer,003
006,Archana,Mishra,23,programmer,003
007,Komal,Nayak,24,teamlead,002
008,Bharathi,Nambiayar,24,manager,001
```

employee_contact.txt

```
001,9848022337,Rajiv@gmail.com,Hyderabad,003
002,9848022338,siddarth@gmail.com,Kolkata,003
003,9848022339,Rajesh@gmail.com,Delhi,003
004,9848022330,Preethi@gmail.com,Pune,003
005,9848022336,Trupthi@gmail.com,Bhuwaneshwar,003
006,9848022335,Archana@gmail.com,Chennai,003
007,9848022334,Komal@gmail.com,trivendram,002
008,9848022333,Bharathi@gmail.com,Chennai,001
```

```
Applications  Places  System
cloudera@quickstart:~/Desktop
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart Desktop]$ cat employee.txt
001,Rajiv,Reddy,21,programmer,003
002,siddarth,Battacharya,22,programmer,003
003,Rajesh,Khanna,22,programmer,003
004,Preethi,Agarwal,21,programmer,003
005,Trupthi,Mohanthy,23,programmer,003
006,Archana,Mishra,23,programmer,003
007,Komal,Nayak,24,teamlead,002
008,Bharathi,Nambiayar,24,manager,001
[cloudera@quickstart Desktop]$ cat employee_contact.txt
001,9848022337,Rajiv@gmail.com,Hyderabad,003
002,9848022338,siddarth@gmail.com,Kolkata,003
003,9848022339,Rajesh@gmail.com,Delhi,003
004,9848022330,Preethi@gmail.com,Pune,003
005,9848022336,Trupthi@gmail.com,Bhuwaneshwar,003
006,9848022335,Archana@gmail.com,Chennai,003
[cloudera@quickstart Desktop]$
```

```
[cloudera@quickstart Desktop]$ hadoop fs -put employee.txt smitrpatel
[cloudera@quickstart Desktop]$ hadoop fs -put employee_contact.txt smitrpatel
[cloudera@quickstart Desktop]$ hadoop fs -ls smitrpatel
Found 4 items
-rw-r--r--   1 cloudera cloudera        194 2021-08-25 17:32 smitrpatel/customers.txt
-rw-r--r--   1 cloudera cloudera        304 2021-08-25 18:08 smitrpatel/employee.txt
-rw-r--r--   1 cloudera cloudera        275 2021-08-25 18:08 smitrpatel/employee_contact.txt
-rw-r--r--   1 cloudera cloudera        124 2021-08-25 17:32 smitrpatel/orders.txt
[cloudera@quickstart Desktop]$
```

And we have loaded these two files into Pig with relations **employee** and **employee_contact** as shown below.

```
grunt> employee = LOAD
'hdfs://localhost:9000/pig_data/employee.txt' USING
PigStorage(',')
 as (id:int, firstname:chararray, lastname:chararray,
age:int, designation:chararray, jobid:int);


grunt> employee_contact = LOAD
'hdfs://localhost:9000/pig_data/employee_contact.t
xt' USING  PigStorage(',')
 as (id:int, phone:chararray, email:chararray,
city:chararray, jobid:int);
```

command :

```
employee = LOAD 'smitrpatel/employee.txt' USING
PigStorage(',') as (id:int, firstname:chararray,
lastname:chararray, age:int, designation:chararray,
jobid:int);
```

```
grunt> employee = LOAD
>> 'smitrpatel/employee.txt' using PigStorage(',') as
>> (id:int, firstname:chararray, lastname:chararray, age:int, designation:chararray, jobid:int);
grunt> employee_contact = LOAD
>> 'smitrpatel/employee_contact.txt' using PigStorage(',') as
>> (id:int, phone:chararray, email:chararray, city:chararray, jobid:int);
grunt>
grunt>
cloudera@quickstart:~...
```

Now, let us join the contents of these two relations using the **JOIN** operator as shown below.

```
grunt> emp = JOIN employee BY (id,jobid), employee_contact
 BY  (id,jobid);
```

```
TRING
Details at logfile: /home/cloudera/Desktop/pig_1629940560239.log
grunt> employee = LOAD
>> 'smitrpatel/employee.txt' using PigStorage(',') as
>> (id:int, firstname:chararray, lastname:chararray, age:int, designation:chararray, jobid:int);
grunt> employee_contact = LOAD
>> 'smitrpatel/employee_contact.txt' using PigStorage(',') as
>> (id:int, phone:chararray, email:chararray, city:chararray, jobid:int);
grunt>
grunt> emp = JOIN employee BY (id,jobid), employee_contact BY (id,jobid);
grunt>
```

Verification

Verify the relation **emp** using the **DUMP** operator as shown below.

```
grunt> Dump emp;
```

```
Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime      MinMapTIme      AvgMapTime      MedianMapTime   MaxReduceTime   MinReduceTime   AvgReduceTime   MedianReducetime    A
lias    Feature Outputs
job_1629070169397_0061  2       1       26      26      26      26      12      12      12      12      emp,employee,employee_contact   HASH_JOIN       hdfs:
//quickstart.cloudera:8020/tmp/temp862422321/tmp1000599393,

Input(s):
Successfully read 6 records from: "hdfs://quickstart.cloudera:8020/user/cloudera/smitrpatel/employee_contact.txt"
Successfully read 8 records from: "hdfs://quickstart.cloudera:8020/user/cloudera/smitrpatel/employee.txt"

Output(s):
Successfully stored 6 records (543 bytes) in: "hdfs://quickstart.cloudera:8020/tmp/temp862422321/tmp1000599393"

Counters:
Total records written : 6
Total bytes written : 543
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1629070169397_0061


2021-08-25 18:24:52,852 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2021-08-25 18:24:52,855 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2021-08-25 18:24:52,855 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.a
ddress
2021-08-25 18:24:52,855 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2021-08-25 18:24:52,878 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2021-08-25 18:24:52,878 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(1,Rajiv,Reddy,21,programmer,3,1,9848022337,Rajiv@gmail.com,Hyderabad,3)
(2,siddarth,Battacharya,22,programmer,3,2,9848022338,siddarth@gmail.com,Kolkata,3)
(3,Rajesh,Khanna,22,programmer,3,3,9848022339,Rajesh@gmail.com,Delhi,3)
(4,Preethi,Agarwal,21,programmer,3,4,9848022330,Preethi@gmail.com,Pune,3)
(5,Trupthi,Mohanthy,23,programmer,3,5,9848022336,Trupthi@gmail.com,Bhuwaneshwar,3)
(6,Archana,Mishra,23,programmer,3,6,9848022335,Archana@gmail.com,Chennai,3)
grunt>
cloudera@quickstart:~...
```

Output

It will produce the following output, displaying the contents of the relation named **emp** as shown below.

2021-08-25 18:24:52,852 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!

2021-08-25 18:24:52,855 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS

2021-08-25 18:24:52,855 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address

2021-08-25 18:24:52,855 [main] INFO org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.

2021-08-25 18:24:52,878 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1

2021-08-25 18:24:52,878 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1

(1,Rajiv,Reddy,21,programmer,3,1,9848022337,Rajiv@gmail.com,Hyderabad,3)

(2,siddarth,Battacharya,22,programmer,3,2,9848022338,siddarth@gmail.com,Kolkata,3)

(3,Rajesh,Khanna,22,programmer,3,3,9848022339,Rajesh@gmail.com,Delhi,3)

(4,Preethi,Agarwal,21,programmer,3,4,9848022330,Preethi@gmail.com,Pune,3)

(5,Trupthi,Mohanthy,23,programmer,3,5,9848022336,Trupthi@gmail.com,Bhuwaneshwar,3
)

(6,Archana,Mishra,23,programmer,3,6,9848022335,Archana@gmail.com,Chennai,3)

CROSS OPERATOR

The CROSS operator computes the cross-product of two or more relations. This chapter explains with example how to use the cross operator in Pig Latin.

Syntax

Given below is the syntax of the CROSS operator.

grunt> Relation3_name = CROSS Relation1_name, Relation2_name;

And we have loaded these two files into Pig with the relations customers and orders.

Let us now get the cross-product of these two relations using the cross operator on these two relations as shown below.

grunt> cross_data = CROSS customers, orders;

```
grunt> cross_data = CROSS customers, orders;
```

```
Applications  Places  System
cloudera@quickstart:~/Desktop
File  Edit  View  Search  Terminal  Help
grunt> cross_data = CROSS customers, orders;
grunt> dump cross_data;
```

Verification

Verify the relation **cross_data** using the **DUMP** operator as shown below.

```
grunt> Dump cross_data;
```

```
 - Total input paths to process : 1
2021-08-17 00:09:12,559 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.Ma
pRedUtil - Total input paths to process : 1
(7,Muffy,24,Indore,10000,103,2008-05-20 00:00:00,4,2060)
(7,Muffy,24,Indore,10000,101,2009-11-20 00:00:00,2,1560)
(7,Muffy,24,Indore,10000,100,2009-10-08 00:00:00,3,1500)
(7,Muffy,24,Indore,10000,102,2009-10-08 00:00:00,3,3000)
(6,Komal,22,MP,4500,103,2008-05-20 00:00:00,4,2060)
(6,Komal,22,MP,4500,101,2009-11-20 00:00:00,2,1560)
(6,Komal,22,MP,4500,100,2009-10-08 00:00:00,3,1500)
(6,Komal,22,MP,4500,102,2009-10-08 00:00:00,3,3000)
(5,Hardik,27,Bhopal,8500,103,2008-05-20 00:00:00,4,2060)
(5,Hardik,27,Bhopal,8500,101,2009-11-20 00:00:00,2,1560)
(5,Hardik,27,Bhopal,8500,100,2009-10-08 00:00:00,3,1500)
(5,Hardik,27,Bhopal,8500,102,2009-10-08 00:00:00,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)
(4,Chaitali,25,Mumbai,6500,101,2009-11-20 00:00:00,2,1560)
(4,Chaitali,25,Mumbai,6500,100,2009-10-08 00:00:00,3,1500)
(4,Chaitali,25,Mumbai,6500,102,2009-10-08 00:00:00,3,3000)
(3,kaushik,23,Kota,2000,103,2008-05-20 00:00:00,4,2060)
(3,kaushik,23,Kota,2000,101,2009-11-20 00:00:00,2,1560)
(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(2,Khilan,25,Delhi,1500,103,2008-05-20 00:00:00,4,2060)
(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)
(2,Khilan,25,Delhi,1500,100,2009-10-08 00:00:00,3,1500)
(2,Khilan,25,Delhi,1500,102,2009-10-08 00:00:00,3,3000)
(1,Ramesh,32,Ahmedabad,2000,103,2008-05-20 00:00:00,4,2060)
(1,Ramesh,32,Ahmedabad,2000,101,2009-11-20 00:00:00,2,1560)
(1,Ramesh,32,Ahmedabad,2000,100,2009-10-08 00:00:00,3,1500)
(1,Ramesh,32,Ahmedabad,2000,102,2009-10-08 00:00:00,3,3000)
grunt>
cloudera@quickstart:...  cloudera@quickstart:~...
```

Output

It will produce the following output, displaying the contents of the relation **cross_data**.

```
(7,Muffy,24,Indore,10000,103,2008-05-20
00:00:00,4,2060)
(7,Muffy,24,Indore,10000,101,2009-11-20
00:00:00,2,1560)
```

(7,Muffy,24,Indore,10000,100,2009-10-08
00:00:00,3,1500)
(7,Muffy,24,Indore,10000,102,2009-10-08
00:00:00,3,3000) (6,Komal,22,MP,4500,103,2008-
05-20 00:00:00,4,2060)

(6,Komal,22,MP,4500,101,2009-11-20 00:00:00,2,1560)

(6,Komal,22,MP,4500,100,2009-10-08 00:00:00,3,1500)

(6,Komal,22,MP,4500,102,2009-10-08 00:00:00,3,3000)

(5,Hardik,27,Bhopal,8500,103,2008-05-20
00:00:00,4,2060)
(5,Hardik,27,Bhopal,8500,101,2009-11-20
00:00:00,2,1560)
(5,Hardik,27,Bhopal,8500,100,2009-10-08
00:00:00,3,1500)
(5,Hardik,27,Bhopal,8500,102,2009-10-08
00:00:00,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20
00:00:00,4,2060)

(4,Chaitali,25,Mumbai,6500,101,2009-20
00:00:00,4,2060)
(2,Khilan,25,Delhi,1500,101,2009-11-20
00:00:00,2,1560)
(2,Khilan,25,Delhi,1500,100,2009-10-08
00:00:00,3,1500)
(2,Khilan,25,Delhi,1500,102,2009-10-08
00:00:00,3,3000)
(1,Ramesh,32,Ahmedabad,2000,103,2008-05-20
00:00:00,4,2060)

(1,Ramesh,32,Ahmedabad,2000,101,2009-11-20
00:00:00,2,1560) (1,Ramesh,32,Ahmedabad,2000,100,2009-
10-08 00:00:00,3,1500)
(1,Ramesh,32,Ahmedabad,2000,102,2009-10-08
00:00:00,3,3000)-11-20 00:00:00,2,1560)

(4,Chaitali,25,Mumbai,6500,100,2009-10-08
00:00:00,3,1500)
(4,Chaitali,25,Mumbai,6500,102,2009-10-08
00:00:00,3,3000)
(3,kaushik,23,Kota,2000,103,2008-05-20
00:00:00,4,2060)
(3,kaushik,23,Kota,2000,101,2009-11-20
00:00:00,2,1560)
(3,kaushik,23,Kota,2000,100,2009-10-08
00:00:00,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08
00:00:00,3,3000)
(2,Khilan,25,Delhi,1500,103,2008-05-20
00:00:00,4,2060)
(2,Khilan,25,Delhi,1500,101,2009-11-20
00:00:00,2,1560)
(2,Khilan,25,Delhi,1500,100,2009-10-08
00:00:00,3,1500) (2,Khilan,25,Delhi,1500,102,2009-
10-08 00:00:00,3,3000)

```
(1,Ramesh,32,Ahmedabad,2000,103,2008-05-20
00:00:00,4,2060)
(1,Ramesh,32,Ahmedabad,2000,101,2009-11-20
00:00:00,2,1560)
(1,Ramesh,32,Ahmedabad,2000,100,2009-10-08
00:00:00,3,1500)
(1,Ramesh,32,Ahmedabad,2000,102,2009-10-08
00:00:00,3,3000)
```

Union Operator

The **UNION** operator of Pig Latin is used to merge the content of two relations. To perform UNION operation on two relations, their columns and domains must be identical.

Syntax

Given below is the syntax of the **UNION** operator.

```
grunt> Relation_name3 = UNION Relation_name1,

Relation_name2;
```

Example

Assume that we have two files namely **student_data1.txt** and **student_data2.txt** in the **/pig_data/** directory of HDFS as shown below.

Student_data1.txt

```
001,Rajiv,Reddy,9848022337,Hyderabad

002,siddarth,Battacharya,9848022338,Kolkata

003,Rajesh,Khanna,9848022339,Delhi

004,Preethi,Agarwal,9848022330,Pune

005,Trupthi,Mohanthy,9848022336,Bhuwaneshwar

006,Archana,Mishra,9848022335,Chennai.
```

Student_data2.txt

7,Komal,Nayak,9848022334,trivendram.

8,Bharathi,Nambiayar,9848022333,Chennai.

```
[cloudera@quickstart Desktop]$ cat student_data1.txt
001,Rajiv,Reddy,9848022337,Hyderabad
002,siddarth,Battacharya,9848022338,Kolkata
003,Rajesh,Khanna,9848022339,Delhi
004,Preethi,Agarwal,9848022330,Pune
005,Trupthi,Mohanthy,9848022336,Bhuwaneshwar
006,Archana,Mishra,9848022335,Chennai
[cloudera@quickstart Desktop]$ cat  student_data2.txt
7,Komal,Nayak,9848022334,trivendram
8,Bharathi,Nambiayar,9848022333,Chennai
[cloudera@quickstart Desktop]$
cloudera@quickstart:~...
```

```
[cloudera@quickstart Desktop]$ hadoop fs -ls smitrpatel
Found 4 items
-rw-r--r--   1 cloudera cloudera        194 2021-08-25 17:32 smitrpatel/customers.txt
-rw-r--r--   1 cloudera cloudera        304 2021-08-25 18:08 smitrpatel/employee.txt
-rw-r--r--   1 cloudera cloudera        275 2021-08-25 18:08 smitrpatel/employee_contact.txt
-rw-r--r--   1 cloudera cloudera        124 2021-08-25 17:32 smitrpatel/orders.txt
[cloudera@quickstart Desktop]$ hadoop fs -put student_data1.txt smitrpatel
[cloudera@quickstart Desktop]$ hadoop fs -put student_data2.txt smitrpatel
[cloudera@quickstart Desktop]$ hadoop fs -ls smitrpatel
Found 6 items
-rw-r--r--   1 cloudera cloudera        194 2021-08-25 17:32 smitrpatel/customers.txt
-rw-r--r--   1 cloudera cloudera        304 2021-08-25 18:08 smitrpatel/employee.txt
-rw-r--r--   1 cloudera cloudera        275 2021-08-25 18:08 smitrpatel/employee_contact.txt
-rw-r--r--   1 cloudera cloudera        124 2021-08-25 17:32 smitrpatel/orders.txt
-rw-r--r--   1 cloudera cloudera        240 2021-08-25 18:33 smitrpatel/student_data1.txt
-rw-r--r--   1 cloudera cloudera         76 2021-08-25 18:33 smitrpatel/student_data2.txt
[cloudera@quickstart Desktop]$
```

And we have loaded these two files into Pig with the relations **student1** and **student2** as shown below.

```
grunt> student1 = LOAD
'hdfs://localhost:9000/pig_data/student_data1.txt'
USING PigStorage(',')
 as (id:int, firstname:chararray, lastname:chararray,
phone:chararray, city:chararray);

grunt> student2 = LOAD
'hdfs://localhost:9000/pig_data/student_data2.txt'
USING PigStorage(',')
 as (id:int, firstname:chararray, lastname:chararray,
phone:chararray, city:chararray);
```

grunt> student1 = load 'smitrpatel/student_data1.txt' using

>> PigStorage(',') as

>> (id:int, firstname:chararray, lastname:chararray, phone:chararray, city:chararray);

grunt> student2 = load 'smitrpatel/student_data2.txt' using

>> PigStorage(',') as

>> (id:int, firstname:chararray, lastname:chararray, phone:chararray, city:chararray);

grunt>

```
ddress
grunt> student1 = load 'smitrpatel/student_data1.txt' using
>> PigStorage(',') as
>> (id:int, firstname:chararray, lastname:chararray, phone:chararray, city:chararray);
grunt> student2 = load 'smitrpatel/student_data2.txt' using
>> PigStorage(',') as
>> (id:int, firstname:chararray, lastname:chararray, phone:chararray, city:chararray);
grunt> [cloudera@quickstart Desktop]$
cloudera@quickstart:~...
```

Let us now merge the contents of these two relations using the **UNION** operator as shown below.

```
grunt> student = UNION student1, student2;
```

```
cloudera@quickstart:~/Desktop
File Edit View Search Terminal Help
grunt> student = UNION student1, student2;
grunt> dump student;
```

Verification

Verify the relation **student** using the **DUMP** operator as shown below.

```
grunt> Dump student;
```

```
pRedUtil - Total input paths to process : 2
(1,Rajiv,Reddy,984802237,Hyderabad)
(2,siddarth,Battacharya,984802238,Kolkata)
(3,Rajesh,Khanna,9848022339,Delhi)
(4,Preethi,Agarwal,9848022330,Pune)
(5,Trupthi,Mohanthy,9848022336,Bhuwaneshwar)
(6,Archana,Mishra,9848022335,Chennai)
(7,Komal,Nayak,9848022334,trivendram)
(8,Bharathi,Nambiayar,9848922333,Chennai)
```

Output

It will display the following output, displaying the contents of the relation **student**.

```
(1,Rajiv,Reddy,9848022337,Hyderabad)
(2,siddarth,Battacharya,9848022338,Kolkata)
(3,Rajesh,Khanna,9848022339,Delhi)
(4,Preethi,Agarwal,9848022330,Pune)
(5,Trupthi,Mohanthy,9848022336,Bhuwaneshwar)
(6,Archana,Mishra,9848022335,Chennai)
(7,Komal,Nayak,9848022334,trivendram)
(8,Bharathi,Nambiayar,9848022333,Chennai)
```

Split Operator

The **SPLIT** operator is used to split a relation into two or more relations. Syntax

Given below is the syntax of the **SPLIT** operator.

```
grunt> SPLIT Relation1_name INTO Relation2_name IF
(condition1),  Relation2_name (condition2),
```

Example

Assume that we have a file named **student_details.txt** in the HDFS directory **/pig_data/** as shown below.

student_details.txt

```
001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddarth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Delhi
004,Preethi,Agarwal,21,9848022330,Pune
005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
006,Archana,Mishra,23,9848022335,Chennai
007,Komal,Nayak,24,9848022334,trivendram
008,Bharathi,Nambiayar,24,9848022333,Chennai
```

```
[cloudera@quickstart Desktop]$ gedit student_details.txt
[cloudera@quickstart Desktop]$ cat student_details.txt
001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddarth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Delhi
004,Preethi,Agarwal,21,9848022330,Pune
005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
006,Archana,Mishra,23,9848022335,Chennai
007,Komal,Nayak,24,9848022334,trivendram
008,Bharathi,Nambiayar,24,9848022333,Chennai
[cloudera@quickstart Desktop]$ hadoop fs -put student_details.txt smitrpatel
[cloudera@quickstart Desktop]$ hadoop fs -ls
Found 6 items
drwxr-xr-x   - cloudera cloudera          0 2021-08-16 11:52 Heet
drwxr-xr-x   - cloudera cloudera          0 2021-08-25 14:54 Prac10
drwxr-xr-x   - cloudera cloudera          0 2021-08-25 15:11 Prac10a
drwxr-xr-x   - cloudera cloudera          0 2021-08-18 15:19 Prac8
drwxr-xr-x   - cloudera cloudera          0 2021-08-18 16:09 Prac9
drwxr-xr-x   - cloudera cloudera          0 2021-08-25 18:44 smitrpatel
[cloudera@quickstart Desktop]$ hadoop fs -ls smitrpatel
Found 7 items
-rw-r--r--   1 cloudera cloudera        194 2021-08-25 17:32 smitrpatel/customers.txt
-rw-r--r--   1 cloudera cloudera        304 2021-08-25 18:08 smitrpatel/employee.txt
-rw-r--r--   1 cloudera cloudera        275 2021-08-25 18:08 smitrpatel/employee_contact.txt
-rw-r--r--   1 cloudera cloudera        124 2021-08-25 17:32 smitrpatel/orders.txt
-rw-r--r--   1 cloudera cloudera        240 2021-08-25 18:33 smitrpatel/student_data1.txt
-rw-r--r--   1 cloudera cloudera         76 2021-08-25 18:33 smitrpatel/student_data2.txt
-rw-r--r--   1 cloudera cloudera        351 2021-08-25 18:44 smitrpatel/student_details.txt
[cloudera@quickstart Desktop]$
```

And we have loaded this file into Pig with the relation name **student_details** as shown below.

```
student_details = LOAD
'hdfs://localhost:9000/pig_data/student_details.txt'
USING PigStorage(',')
 as (id:int, firstname:chararray, lastname:chararray, age:int,
phone:chararray, city:chararray);
```

grunt> student_details = load'smitrpatel/student_details.txt'

> using PigStorage(',') as

>> (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);

grunt> student_details1 = load'smitrpatel/student_details1.txt'

>> using PigStorage(',') as

>> (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);

grunt> student_details2 = load'smitrpatel/student_details2.txt'

>> using PigStorage(',') as

>> (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);

```
grunt> student_details = load'smitrpatel/student_details.txt'
>> using PigStorage(',') as
>> (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);
grunt> student_details1 = load'smitrpatel/student_details1.txt'
>> using PigStorage(',') as
>> (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);
grunt> student_details2 = load'smitrpatel/student_details2.txt'
>> using PigStorage(',') as
>> (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);
grunt> dump student_details1;[cloudera@quickstart Desktop]$ ^C
[cloudera@quickstart Desktop]$
```

cloudera@quickstart:~...

Let us now split the relation into two, one listing the employees of age less than 23, and the other listing the employees having the age between 22 and 25.

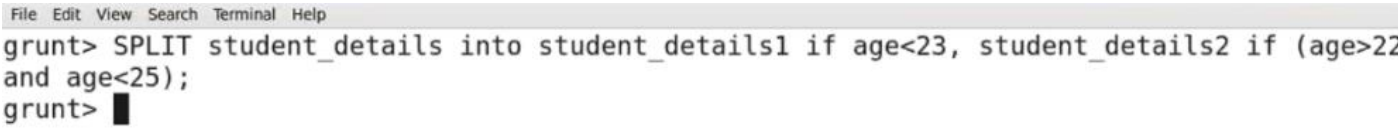
```
SPLIT student_details into student_details1 if
age<23, student_details2 if (22<age and age>25);
```

Command : grunt>SPLIT student_details into student_details1 if age<23, student_details2 if (age>22 and age<25);

Verification

Verify the relations **student_details1** and **student_details2** using the **DUMP** operator as shown below.

```
grunt> Dump student_details1;

grunt> Dump student_details2;
```

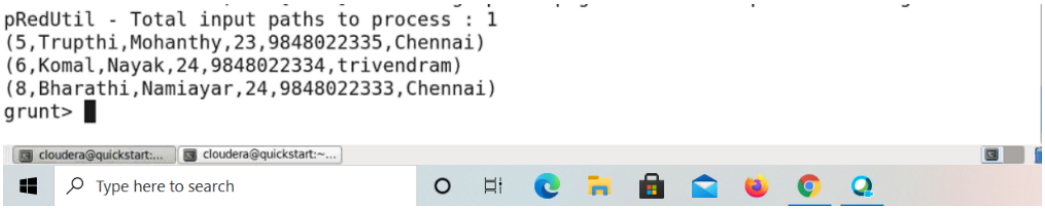


Output

It will produce the following output, displaying the contents of the relations **student_details1** and **student_details2** respectively.

```
grunt> Dump student_details1;

(1,Rajiv,Reddy,21,9848022337,Hyderabad)

(2,siddarth,Battacharya,22,9848022338,Kolkata)

(3,Rajesh,Khanna,22,9848022339,Delhi)

(4,Preethi,Agarwal,21,9848022330,Pune)
```

```
grunt> Dump student_details2;
(5,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar)
(6,Archana,Mishra,23,9848022335,Chennai)
(7,Komal,Nayak,24,9848022334,trivendram)
(8,Bharathi,Nambiayar,24,9848022333,Chennai)
```



Filter Operator

The **FILTER** operator is used to select the required tuples from a relation based on a condition.

Syntax

Given below is the syntax of the **FILTER** operator.

```
grunt> Relation2_name = FILTER Relation1_name BY
(condition);
```

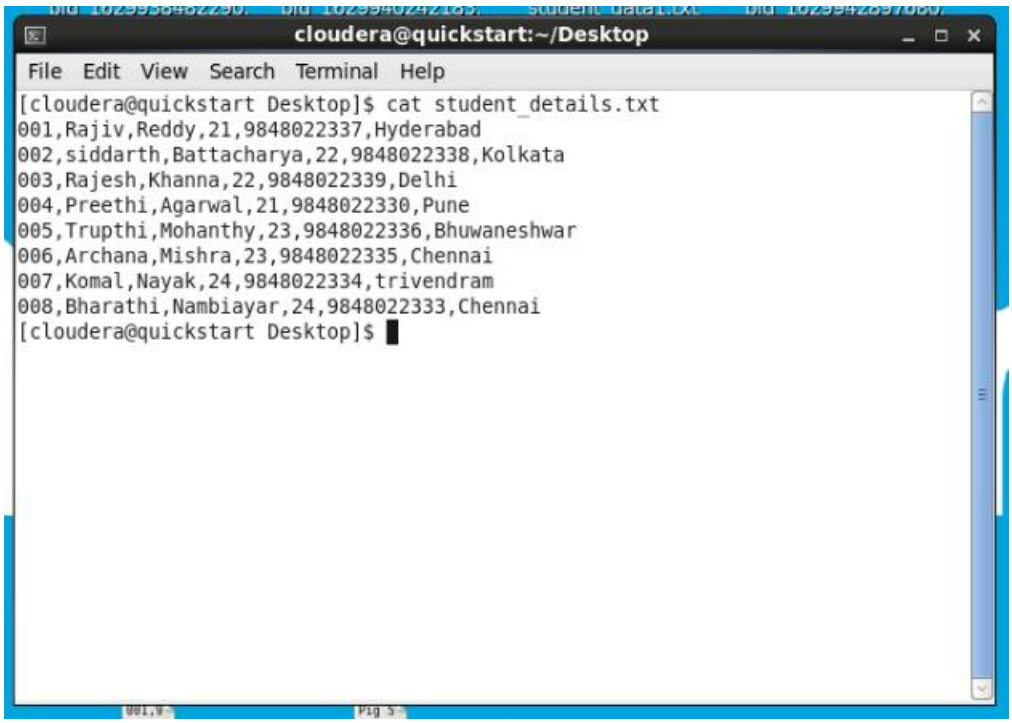
Example

Assume that we have a file named **student_details.txt** in the HDFS directory **/pig_data/** as shown below.

student_details.txt

```
001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddarth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Delhi
004,Preethi,Agarwal,21,9848022330,Pune
```

```
005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
006,Archana,Mishra,23,9848022335,Chennai
007,Komal,Nayak,24,9848022334,trivendram
008,Bharathi,Nambiayar,24,9848022333,Chennai
```



And we have loaded this file into Pig with the relation name **student_details** as shown below.

```
grunt> student_details = LOAD
'hdfs://localhost:9000/pig_data/student_details.txt'
USING  PigStorage(',')
 as (id:int, firstname:chararray, lastname:chararray, age:int,
phone:chararray, city:chararray);
```

```
cloudera@quickstart:~/Desktop
File  Edit  View  Search  Terminal  Help
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2021-08-25 19:21:20,601 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.addr
ess
grunt> student_details = load 'smitrpatel/student_details.txt'
>> using PigStorage(',') as                                                c
iry:)d:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, )    chararray);
grunt> dump student_details;
2021-08-25 19:23:15,899 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: UNKNOWN
2021-08-25 19:23:15,982 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, Duplicate
ForEachColumnRewrite, GroupByConstParallelSetter, ImplicitSplitInserter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NewPartitionFilterO
ptimizer, PushDownForEachFlatten, PushUpFilter, SplitFilter, StreamTypeCastInserter], RULES_DISABLED=[FilterLogicExpressionSimplifier, PartitionFilterOptimiz
er]}
2021-08-25 19:23:16,320 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic?
false
2021-08-25 19:23:16,386 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2021-08-25 19:23:16,387 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
2021-08-25 19:23:16,735 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2021-08-25 19:23:17,252 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig script settings are added to the job
2021-08-25 19:23:17,397 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.reduce.markreset.buffer.percent is deprecated. Instead, us
e mapreduce.reduce.markreset.buffer.percent
2021-08-25 19:23:17,397 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffer.per
cent is not set, set to default 0.3
2021-08-25 19:23:17,397 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.output.compress is deprecated. Instead, use mapreduce.output.f
ileoutputformat.compress
2021-08-25 19:23:18,524 [DataStreamer for file /tmp/temp1335398707/tmp-2132096910/hive-hbase-handler-1.1.0-cdh5.12.0.jar] WARN  org.apache.hadoop.hdfs.DFSCli
ent - Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1281)
        at java.lang.Thread.join(Thread.java:1355)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:952)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:690)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:879)
2021-08-25 19:23:18,598 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - creating jar file Job72630063883140898
82.jar
2021-08-25 19:23:27,868 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - jar file Job7263006388314089882.jar cr
eated
2021-08-25 19:23:27,868 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.jar is deprecated. Instead, use mapreduce.job.jar
cloudera@quickstart:~...
```

Let us now use the Filter operator to get the details of the students who belong to the city Chennai.

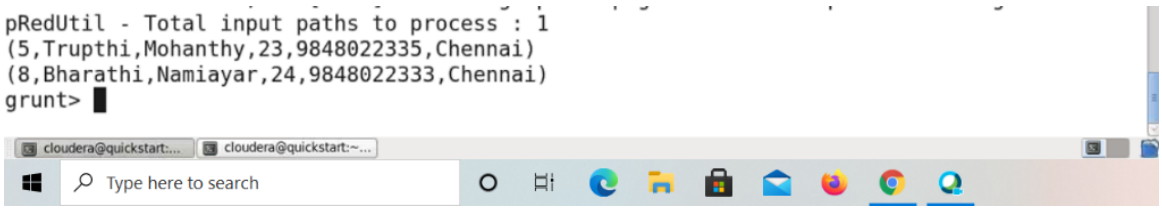
```
filter_data = FILTER student_details BY city == 'Chennai';
```

```
grunt> filter_data = FILTER student_details BY city == 'Chennai';
grunt>
grunt>
```

Verification

Verify the relation **filter_data** using the **DUMP** operator as shown below.

```
grunt> Dump filter_data;
```



Output

It will produce the following output, displaying the contents of the relation **filter_data** as follows.

```
(6,Archana,Mishra,23,9848022335,Chennai)
(8,Bharathi,Nambiayar,24,9848022333,Chennai)
```

Distinct Operator

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

Syntax

Given below is the syntax of the **DISTINCT** operator.

```
grunt> Relation_name2 = DISTINCT Relatin_name1;
```

Example

Assume that we have a file named **student_details.txt** in the HDFS directory **/pig_data/** as shown below.

student_details.txt

```
001,Rajiv,Reddy,9848022337,Hyderabad
002,siddarth,Battacharya,9848022338,Kolkata
002,siddarth,Battacharya,9848022338,Kolkata
003,Rajesh,Khanna,9848022339,Delhi
003,Rajesh,Khanna,9848022339,Delhi
004,Preethi,Agarwal,9848022330,Pune
005,Trupthi,Mohanthy,9848022336,Bhuwaneshwar
006,Archana,Mishra,9848022335,Chennai
006,Archana,Mishra,9848022335,Chennai
```

```
cloudera@quickstart:~/Desktop
File Edit View Search Terminal Help
[cloudera@quickstart Desktop]$ cat student_details2.txt
001,Rajiv,Reddy,9848022337,Hyderabad
002,siddarth,Battacharya,9848022338,Kolkata
003,Rajesh,Khanna,9848022339,Delhi
004,Preethi,Agarwal,9848022330,Pune
005,Trupthi,Mohanthy,9848022336,Bhuwaneshwar
006,Archana,Mishra,9848022335,Chennai
001,Rajiv,Reddy,9848022337,Hyderabad
002,siddarth,Battacharya,9848022338,Kolkata
003,Rajesh,Khanna,9848022339,Delhi
004,Preethi,Agarwal,9848022330,Pune
005,Trupthi,Mohanthy,9848022336,Bhuwaneshwar
006,Archana,Mishra,9848022335,Chennai
[cloudera@quickstart Desktop]$
```

And we have loaded this file into Pig with the relation name **student_details** as shown below.

```
grunt> student_details = LOAD
'hdfs://localhost:9000/pig_data/student_details.txt'
USING PigStorage(',')
as (id:int, firstname:chararray, lastname:chararray,
phone:chararray, city:chararray);
```

grunt> student_details2 = load 'smitrpatel/student_details1.txt' using PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);

grunt> student_details = load 'smitrpatel/student_details2'

>> using PigStorage(',') as

>> (id:int, firstname:chararray, lastname:chararray, phone:chararray, city:chararray);

```
grunt> student_details = load 'smitrpatel/student_details2'
>> using PigStorage(',') as
>> (id:int, firstname:chararray, lastname:chararray, phone:chararray, city:chararray);
grunt>
```

Let us now remove the redundant (duplicate) tuples from the relation named

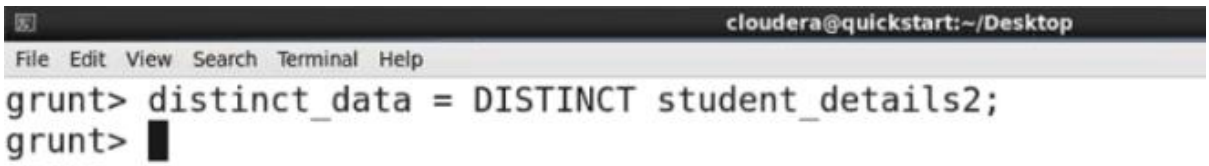
student_details using the **DISTINCT** operator, and store it as another relation named **distinct_data** as shown below.

```
grunt> distinct_data = DISTINCT student_details;
```

Verification

Verify the relation **distinct_data** using the **DUMP** operator as shown below.

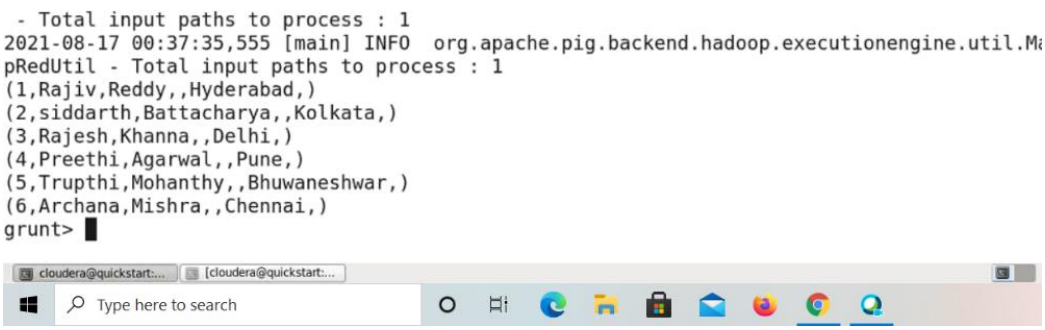
```
grunt> Dump distinct_data;
```



Output

It will produce the following output, displaying the contents of the relation **distinct_data** as follows.

```
(1,Rajiv,Reddy,9848022337,Hyderabad)
(2,siddarth,Battacharya,9848022338,Kolkata)
(3,Rajesh,Khanna,9848022339,Delhi)
(4,Preethi,Agarwal,9848022330,Pune)
(5,Trupthi,Mohanthy,9848022336,Bhuwaneshwar)
(6,Archana,Mishra,9848022335,Chennai)
```



Foreach Operator

The **FOREACH** operator is used to generate specified data transformations based on the column data.

Syntax

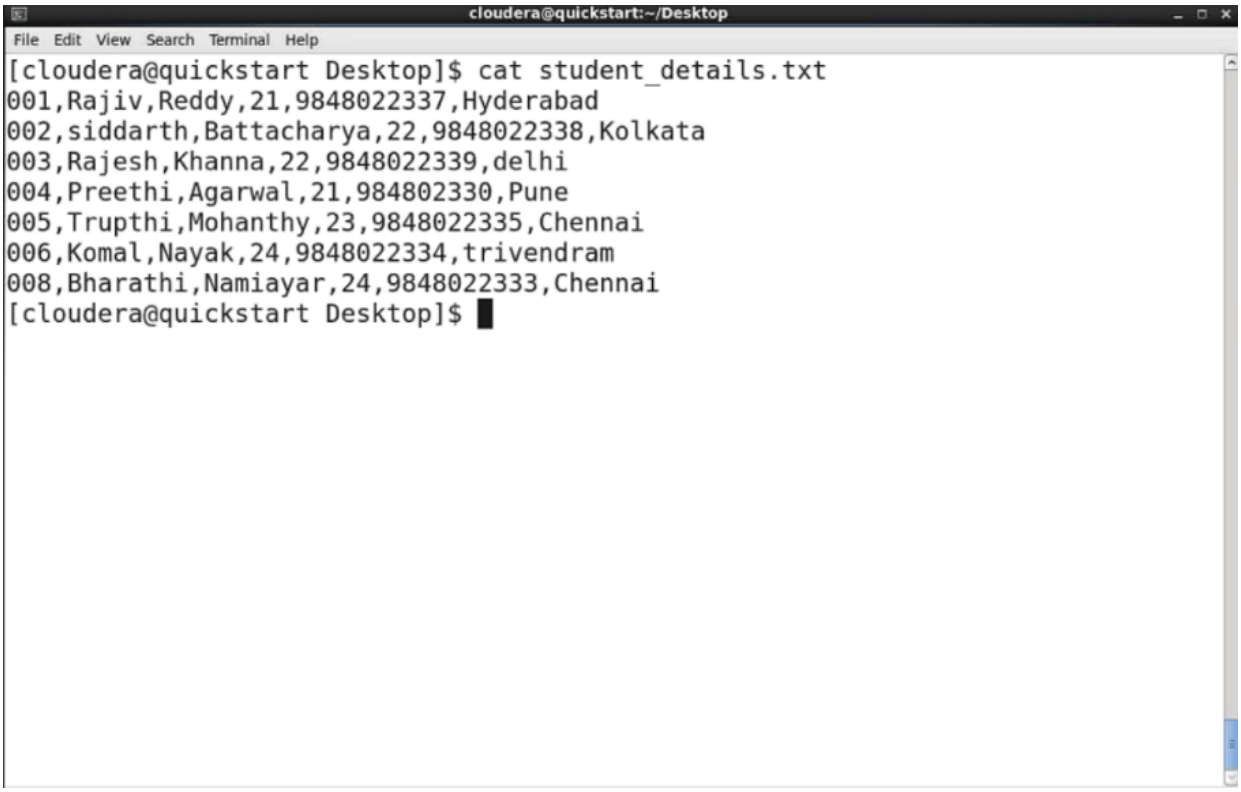
Given below is the syntax of **FOREACH** operator.

```
grunt> Relation_name2 = FOREACH Relatin_name1 GENERATE
(required  data);
```

Example

Assume that we have a file named **student_details.txt** in the HDFS directory **/pig_data/** as shown below.

student_details.txt




```
001,Rajiv,Reddy,21,9848022337,Hyderabad

002,siddarth,Battacharya,22,9848022338,Kolkata

003,Rajesh,Khanna,22,9848022339,Delhi

004,Preethi,Agarwal,21,9848022330,Pune
```

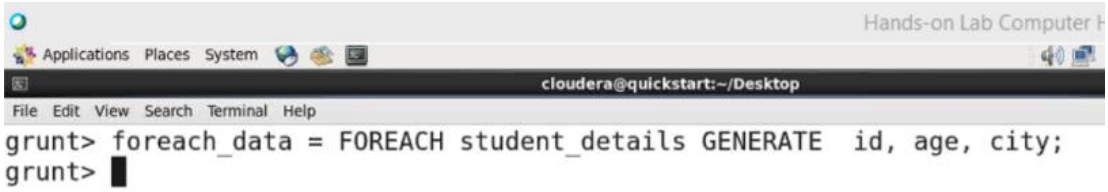
```
005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
006,Archana,Mishra,23,9848022335,Chennai
007,Komal,Nayak,24,9848022334,trivendram
008,Bharathi,Nambiayar,24,9848022333,Chennai
```

And we have loaded this file into Pig with the relation name **student_details** as shown below.

```
grunt> student_details = LOAD
'hdfs://localhost:9000/pig_data/student_details.txt'
USING  PigStorage(',')
 as (id:int, firstname:chararray, lastname:chararray,age:int,
phone:chararray, city:chararray);
```

Let us now get the id, age, and city values of each student from the relation **student_details** and store it into another relation named **foreach_data** using the **foreach** operator as shown below.

```
grunt> foreach_data = FOREACH student_details
GENERATE  id,age,city;
```



Verification

Verify the relation **foreach_data** using the **DUMP** operator as shown below.

```
grunt> Dump foreach_data;
```

```
pRedUtil - Total input paths to process : 1
(1,21,Hyderabad)
(2,22,Kolkata)
(3,22,delhi)
(4,21,Pune)
(5,23,Chennai)
(6,24,trivendram)
(8,24,Chennai)
```

Output

It will produce the following output, displaying the contents of the relation **foreach_data**.

```
(1,21,Hyderabad)
(2,22,Kolkata)
(3,22,Delhi)
(4,21,Pune)
(5,23,Bhuwaneshwar)
(6,23,Chennai)
(7,24,trivendram)
(8,24,Chennai)
```

Order By

The **ORDER BY** operator is used to display the contents of a relation in a sorted order based on one or more fields.

Syntax

Given below is the syntax of the **ORDER BY** operator.

```
grunt> Relation_name2 = ORDER Relatin_name1 BY
(ASC|DESC);
```

Example

Assume that we have a file named **student_details.txt** in the HDFS directory **/pig_data/** as shown below.

student_details.txt

```
cloudera@quickstart:~/Desktop
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart Desktop]$ cat student_details.txt
001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddarth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,delhi
004,Preethi,Agarwal,21,984802330,Pune
005,Trupthi,Mohanthy,23,9848022335,Chennai
006,Komal,Nayak,24,9848022334,trivendram
008,Bharathi,Namiayar,24,9848022333,Chennai
[cloudera@quickstart Desktop]$
```

```
001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddarth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Delhi
004,Preethi,Agarwal,21,9848022330,Pune
005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
006,Archana,Mishra,23,9848022335,Chennai
007,Komal,Nayak,24,9848022334,trivendram
008,Bharathi,Nambiayar,24,9848022333,Chennai
```

And we have loaded this file into Pig with the relation name **student_details** as shown below.

```
grunt> student_details = LOAD
'hdfs://localhost:9000/pig_data/student_details.txt'
USING  PigStorage(',')
 as (id:int, firstname:chararray, lastname:chararray,age:int,
phone:chararray, city:chararray);
```

Let us now sort the relation in a descending order based on the age of the student and store it into another relation named **order_by_data** using the **ORDER BY** operator as shown below.

```
grunt> order_by_data = ORDER student_details BY age DESC;
```

```
grunt> order_by_data =  ORDER student_details BY age DESC;
grunt>
```

Verification

Verify the relation **order_by_data** using the **DUMP** operator as shown below.

```
grunt> Dump order_by_data;
```

Output

It will produce the following output, displaying the contents of the relation **order_by_data**.

```
(8,Bharathi,Nambiayar,24,9848022333,Chennai)
(7,Komal,Nayak,24,9848022334,trivendram)
(6,Archana,Mishra,23,9848022335,Chennai)
(5,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar)
(3,Rajesh,Khanna,22,9848022339,Delhi)
(2,siddarth,Battacharya,22,9848022338,Kolkata)
(4,Preethi,Agarwal,21,9848022330,Pune)
(1,Rajiv,Reddy,21,9848022337,Hyderabad)
```

```
pRedUtil - Total input paths to process : 1
(8,Bharathi,Namiayar,24,9848022333,Chennai)
(6,Komal,Nayak,24,9848022334,trivendram)
(5,Trupthi,Mohanthy,23,9848022335,Chennai)
(3,Rajesh,Khanna,22,9848022339,delhi)
(2,siddarth,Battacharya,22,9848022338,Kolkata)
(4,Preethi,Agarwal,21,984802330,Pune)
(1,Rajiv,Reddy,21,9848022337,Hyderabad)
grunt>
```

2 Items in Trash

Limit Operator

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

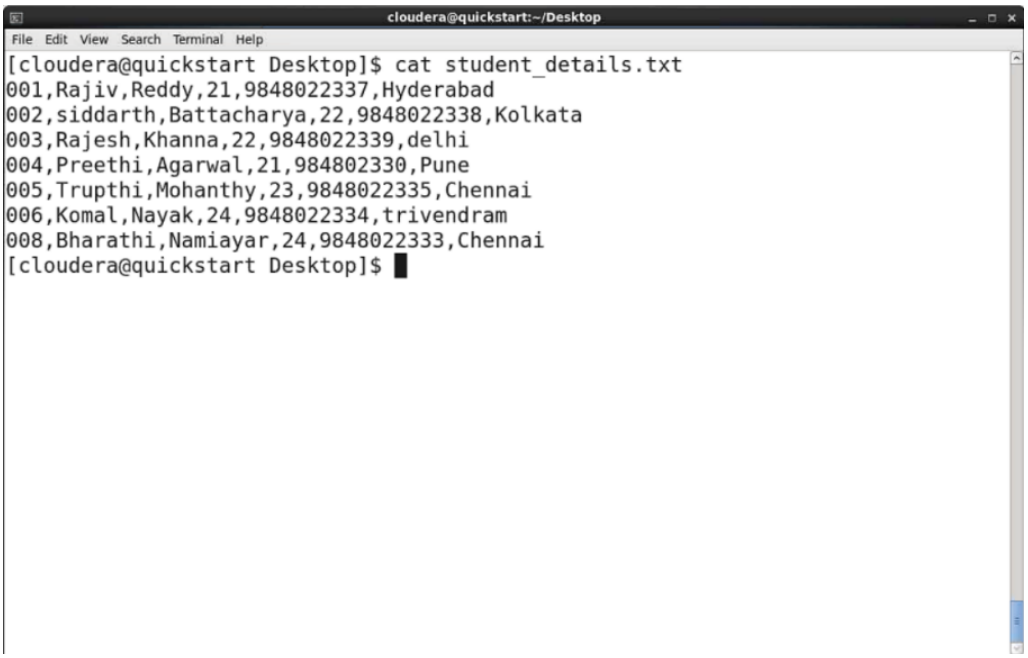
Given below is the syntax of the **LIMIT** operator.

```
grunt> Result = LIMIT Relation_name required number of tuples;
```

Example

Assume that we have a file named **student_details.txt** in the HDFS directory **/pig_data/** as shown below.

student_details.txt




```
001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddarth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Delhi
004,Preethi,Agarwal,21,9848022330,Pune
005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
006,Archana,Mishra,23,9848022335,Chennai
007,Komal,Nayak,24,9848022334,trivendram
008,Bharathi,Nambiayar,24,9848022333,Chennai
```

And we have loaded this file into Pig with the relation name **student_details** as shown below.

```
grunt> student_details = LOAD
'hdfs://localhost:9000/pig_data/student_details.txt'
USING  PigStorage(',')
 as (id:int, firstname:chararray, lastname:chararray,age:int,
phone:chararray, city:chararray);
```

Now, let's sort the relation in descending order based on the age of the student and store it into another relation named **limit_data** using the **ORDER BY** operator as shown below.

```
grunt> limit_data = LIMIT student_details 4;
```

```
File  Edit  View  Search  Terminal  Help
grunt> limit_data = LIMIT student_details 4;
grunt>
```

Verification

Verify the relation **limit_data** using the **DUMP** operator as shown below.

```
grunt> Dump limit_data;
```

```
pRedUtil - Total input paths to process : 1
(1,Rajiv,Reddy,21,9848022337,Hyderabad)
(2,siddarth,Battacharya,22,9848022338,Kolkata)
(3,Rajesh,Khanna,22,9848022339,delhi)
(4,Preethi,Agarwal,21,984802330,Pune)
grunt>
```

Output

It will produce the following output, displaying the contents of the relation **limit_data** as follows.

```
(1,Rajiv,Reddy,21,9848022337,Hyderabad)

(2,siddarth,Battacharya,22,9848022338,Kolkata)

(3,Rajesh,Khanna,22,9848022339,Delhi)
(4,Preethi,Agarwal,21,9848022330,Pune)
```