

Create a sample dataset and implement the below Pig commands on the same dataset.

1) concat

The concat () function of pig latin is used to concatenate two or more expressions of same type

Assume that we have a file named student_details.txt with fields like

001,Rajiv,Reddy,21,9875627192,Hyderabad,89

```
(1,Rajiv,Reddy,21,9848022337,Hyderabad,89)
(2,Siddharth,Bhatacaharya,22,9848022338,Kolkatta,78)
(3,Rajesh,Khanna,22,9848022339,Delhi,90)
(4,Preeti,Agarwal,21,9848022330,Pune,93)
(5,Trupti,Desai,23,9848022336,Bhuvaneshwar,75)
(6,Archana,Mishra,23,9848022335,Chennai,87)
(7,Komal,Nayak,24,9848022334,Trivendrum, )
grunt> █
```

Syntax:

Student_name_concat = foreach student_deatils Generate CONCAT
(firstname,lastname);

```
(RajivReddy)
(SiddharthBhatacaharya)
(RajeshKhanna)
(PreetiAgarwal)
(TruptiDesai)
(ArchanaMishra)
(KomalNayak)
grunt> █
```

2) Tokenize

Tokenize function can be used to split a string. For example, let us split the above student_details.txt using the following command.

```
(1,Rajiv,Reddy,21,9848022337,Hyderabad,89)
(2,Siddharth,Bhatacaharya,22,9848022338,Kolkatta,78)
(3,Rajesh,Khanna,22,9848022339,Delhi,90)
(4,Preeti,Agarwal,21,9848022330,Pune,93)
(5,Trupti,Desai,23,9848022336,Bhuvaneshwar,75)
(6,Archana,Mishra,23,9848022335,Chennai,87)
(7,Komal,NAYak,24,9848022334,Trivendrum,)
grunt> █
```

Syntax:

```
student_name_tokenize = foreach student_details Generate
TOKENIZE(firstname);
```

```
dump student_name_tokenize
```

```
paths to process : 1
({(Rajiv)})
({(Siddharth)})
({(Rajesh)})
({(Preeti)})
({(Trupti)})
({(Archana)})
({(Komal)})
grunt> █
```

3) Sum

We have a file name employee.txt under /home/acadgild/pig. Running dump on employee.txt will give the following output

```
paths to process : 1
(1,John,2007-01-24,250)
(2,Ram,2007-05-27,220)
(3,Jack,2007-05-06,170)
(3,Jack,2007-04-06,100)
(4,Jill,2007-04-06,220)
grunt> █
```

Load Syntax:

```
employee_data = LOAD '/home/acadgild/pig/employee.txt' USING
PigStorage(',') as (id:int, name:chararray, workdate:chararray,
daily_typing_pages:int);
```

Group Syntax:

```
employee_group = Group employee_data all;
```

Sum Syntax:

```
student_workpages_sum = foreach employee_group Generate
(employee_data.name,employee_data.daily_typing_pages),SUM(employee_da
ta.daily_typing_pages);
```

```
paths to process : 1
(((Jill),(Jack),(Jack),(Ram),(John)),{(220),(100),(170),(220),(250)}),960)
grunt> █
```

4) Min

The min function of pig Latin is used to get the minimum value for a certain column in a single column bag. When calculating the min value NULL is ignored.

Student Details dataset:

```
(1,Rajiv,Reddy,21,9848022337,Hyderabad,89)
(2,Siddharth,Bhatacharya,22,9848022338,Kolkatta,78)
(3,Rajesh,Khanna,22,9848022339,Delhi,90)
(4,Preeti,Agarwal,21,9848022330,Pune,93)
(5,Trupti,Desai,23,9848022336,Bhuvaneshwar,75)
(6,Archana,Mishra,23,9848022335,Chennai,87)
(7,Komal,NAYak,24,9848022334,Trivendrum,)
grunt> █
```

Min() Syntax:

student_group_all = Group student_details All;

student_gpa_min = foreach student_group_all Generate

(student_details.firstname, student_details.gpa), MIN(student_details.gpa);

```
paths to process : 1
(((Komal),(Archana),(Trupti),(Preeti),(Rajesh),(Siddharth),(Rajiv)),{(),(87),(75),(93),(90),(78),(89)}),75)
grunt> █
```

5. Max

The max function of pig Latin is used to get the maximum value for a certain column in a single column bag. When calculating the max value NULL is ignored.

Student Details dataset:

```
(1,Rajiv,Reddy,21,9848022337,Hyderabad,89)
(2,Siddharth,Bhatacaharya,22,9848022338,Kolkatta,78)
(3,Rajesh,Khanna,22,9848022339,Delhi,90)
(4,Preeti,Agarwal,21,9848022330,Pune,93)
(5,Trupti,Desai,23,9848022336,Bhuvaneshwar,75)
(6,Archana,Mishra,23,9848022335,Chennai,87)
(7,Komal,NAYak,24,9848022334,Trivendrum,)
grunt> █
```

Max() Syntax:

student_group_all = Group student_details All;

student_gpa_max = foreach student_group_all Generate

(student_details.firstname, student_details.gpa), MAX(student_details.gpa);

```
paths to process : 1
(((Komal),(Archana),(Trupti),(Preeti),(Rajesh),(Siddharth),(Rajiv)),{},{(87),(75),(93),(90),(78),(89)}),93)
grunt> █
```

6. Limit

The limit operator is used to get a limited number of tuples from a relation.

Syntax:

Result = LIMIT Relation_name required number of tuples;

Student Details dataset:

```
(1,Rajiv,Reddy,21,9848022337,Hyderabad,89)
(2,Siddharth,Bhatacaharya,22,9848022338,Kolkatta,78)
(3,Rajesh,Khanna,22,9848022339,Delhi,90)
(4,Preeti,Agarwal,21,9848022330,Pune,93)
(5,Trupti,Desai,23,9848022336,Bhuvaneshwar,75)
(6,Archana,Mishra,23,9848022335,Chennai,87)
(7,Komal,Nayak,24,9848022334,Trivendrum,)
grunt> █
```

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

```
paths to process : 1
(1,Rajiv,Reddy,21,9848022337,Hyderabad,89)
(2,Siddharth,Bhatacaharya,22,9848022338,Kolkatta,78)
(3,Rajesh,Khanna,22,9848022339,Delhi,90)
(4,Preeti,Agarwal,21,9848022330,Pune,93)
grunt> █
```

7. Store

We can store the loaded data in the file system using the store operator.

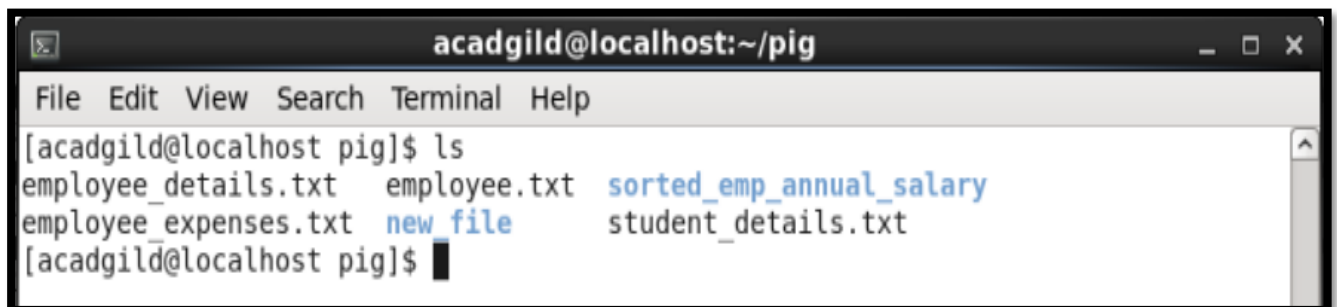
Given below is the syntax of store operator

STORE Relation_name INTO 'required directory path' [using function];

Student_details.txt dataset

```
(1,Rajiv,Reddy,21,9848022337,Hyderabad,89)
(2,Siddharth,Bhatacaharya,22,9848022338,Kolkatta,78)
(3,Rajesh,Khanna,22,9848022339,Delhi,90)
(4,Preeti,Agarwal,21,9848022330,Pune,93)
(5,Trupti,Desai,23,9848022336,Bhuvaneshwar,75)
(6,Archana,Mishra,23,9848022335,Chennai,87)
(7,Komal,Nayak,24,9848022334,Trivendrum,)
grunt> █
```

STORE student_details INTO ' /home/acadgild/new_file ' USING PigStorage (',');

A screenshot of a terminal window titled 'acadgild@localhost:~/pig'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the command '[acadgild@localhost pig]\$ ls' and its output: 'employee_details.txt', 'employee.txt', 'sorted_emp_annual_salary', 'employee_expenses.txt', 'new_file', and 'student_details.txt'. The prompt '[acadgild@localhost pig]\$' is followed by a cursor.

```
acadgild@localhost:~/pig
File Edit View Search Terminal Help
[acadgild@localhost pig]$ ls
employee_details.txt  employee.txt  sorted_emp_annual_salary
employee_expenses.txt new_file      student_details.txt
[acadgild@localhost pig]$ █
```

8. Distinct

The distinct operator is used to remove redundant information (duplicate) tuples from a relation.

Syntax:

Relation_name2 = DISTINCT Relation_name1;

We have a file named student_details

```
(1,Rajiv,Reddy,21,9848022337,Hyderabad,89)
(2,Siddharth,Bhatacharya,22,9848022338,Kolkatta,78)
(3,Rajesh,Khanna,22,9848022339,Delhi,90)
(4,Preeti,Agarwal,21,9848022330,Pune,93)
(5,Trupti,Desai,23,9848022336,Bhubaneswar,75)
(6,Archana,Mishra,23,9848022335,Chennai,87)
(7,Komal,Nayak,24,9848022334,Trivendrum,)
grunt> █
```

```
grunt> distinct_students = DISTINCT student_details;█
```

Since there are no duplicate entries in the relation all the rows/entries are displayed.

```
(1,Rajiv,Reddy,21,9848022337,Hyderabad,89)
(2,Siddharth,Bhatacharya,22,9848022338,Kolkatta,78)
(3,Rajesh,Khanna,22,9848022339,Delhi,90)
(4,Preeti,Agarwal,21,9848022330,Pune,93)
(5,Trupti,Desai,23,9848022336,Bhubaneswar,75)
(6,Archana,Mishra,23,9848022335,Chennai,87)
(7,Komal,Nayak,24,9848022334,Trivendrum,)
grunt> █
```


9. Flatten

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

```
A = load 'input1' USING PigStorage(',') as (x, y);
(x,y) --> (1,2)(1,3)(2,3)
B = load 'input2' USING PigStorage(',') as (x, z);`
(x,z) --> (1,4)(1,2)(3,2)*
C = cogroup A by x, B by x;`
```

result:

```
(1,{(1,2),(1,3)},{(1,4),(1,2)})
(2,{(2,3)},{})
(3,{},{(3,2)})
```

```
D = foreach C generate group, flatten(A), flatten(B);`
```

when both bags flattened, the cross product of tuples are returned.

result:

```
(1,1,2,1,4)
(1,1,2,1,2)
(1,1,3,1,4)
(1,1,3,1,2)
```

```
E = group D by A::x`
```

here your are grouping with x column of relation A.

10. IsEmpty

The IsEmpty() function of pig latin is used to check if a bag or map is empty.

Syntax:

IsEmpty(Expression)

Emp_sales.txt

```
1,Robin,22,25000,sales
2,BOB,23,30000,sales
3,Maya,23,25000,sales
4,Sara,25,40000,sales
5,David,23,45000,sales
6,Maggy,22,35000,sales
```

Emp_bonus.txt

```
1,Robin,22,25000,sales
2,Jaya,23,20000,admin
3,Maya,23,25000,sales
4,Alia,25,50000,admin
5,David,23,45000,sales
6,Omar,30,30000,admin
```

cogroup_data = COGROUP emp_sales by age, emp_bonus by age;

```
grunt> Dump cogroup_data;
```

```
(22, {(6, Maggy, 22, 35000, sales), (1, Robin, 22, 25000, sales)}, {(1, Robin, 22, 25000, sales)})  
(23, {(5, David, 23, 45000, sales), (3, Maya, 23, 25000, sales), (2, BOB, 23, 30000, sales)},  
      {(5, David, 23, 45000, sales), (3, Maya, 23, 25000, sales), (2, Jaya, 23, 20000, admin)})  
(25, {(4, Sara, 25, 40000, sales)}, {(4, Alia, 25, 50000, admin)})  
(30, {}, {(6, Omar, 30, 30000, admin)})
```

grunt> isempty_data = filter cogroup_data by isEmpty(emp_sales);

```
grunt> Dump isempty_data;
```

```
(30, {}, {(6, Omar, 30, 30000, admin)})
```