Pig Commands

1. Command: LOAD

Usage:To analyze data using Apache Pig, Initially we have to load the data into Apache Pig.

Syntax: Relation_name = LOAD 'Input file path' USING function as schema;

Note: Please make sure that your input file should be in HDFS(when working with pig in cluster mode)

Example: student = LOAD '/usecases/965618/sample_dataset' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, phone:chararray,
city:chararray);

2.**Command:** STORE

Usage:To store the loaded data in the file system using the store operator

Syntax: STORE Relation_name INTO ' required_directory_path ' [USING function];
Note:- Map Reduce code will be generated automatically for STORE command.
Example: STORE student into '/usecases/965618/stu1' using PigStorage('|');

3.**Command:** DUMP

Usage:To run the Pig Latin statements and display the results on the screen.

Syntax: Dump Relation_Name
Note:- Map Reduce code will be generated automatically for DUMP command.
Example: DUMP student;

4.**Command:** Describe

Usage:To view the schema of a relation.

Syntax: Describe Relation_name;

Example: describe student;

Output: student: { id: int,firstname: chararray,lastname: chararray,phone: chararray,city: chararray };

5.**Command:** Explain

Usage: To display the logical, physical, and MapReduce execution plans of a relation.

Syntax: Explain Relation_name;

Example: Explain student;

6.**Command: I**llustrate

Usage: It gives you the step-by-step execution of a sequence of statements.

Syntax: Illustrate Relation_name;

Example: Illustrate student;

Sample Student Dataset :

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

Sample Employee Dataset:

001,Robin,22,newyork
002,BOB,23,Kolkata
003,Maya,23,Tokyo
004,Sara,25,London
005,David,23,Bhuwaneshwar
006,Maggy,22,Chennai

7.**Command:** Group

Usage: The GROUP operator is used to group the data in one or more relations. It collects the data having the same key. We can group the relationship by multiple columns and we can group a relation by all the columns also.

Syntax: Group_data = GROUP Relation_name BY age;

Example: i. group_data = GROUP student by age;
ii. group_date1 = GROUP student by (age,city);
iii. group_data2 = GROUP student ALL;

Output:

i.
(21,{(4,Preethi,Agarwal,21,9848022330,Pune),(1,Rajiv,Reddy,21,9848022337,Hyderabad)})
(22,{(3,Rajesh,Khanna,22,9848022339,Delhi),(2,siddarth,Battacharya,22,9848022338,Kolkata)})

ii.
((21,Pune),{(4,Preethi,Agarwal,21,9848022330,Pune)})
((21,Hyderabad),{(1,Rajiv,Reddy,21,9848022337,Hyderabad)})
((22,Delhi),{(3,Rajesh,Khanna,22,9848022339,Delhi)})

iii.
(all,{(8,Bharathi,Nambiayar,24,9848022333,Chennai),(7,Komal,Nayak,
24,9848022334 ,trivendram),

(6,Archana,Mishra,23,9848022335,Chennai),(5,Trupthi,Mohanthy,23,9848022336,Bhuw
aneshwar)})

8.**Command: COGROUP**

Usage: The COGROUP operator works more or less in the same way as the GROUP operator. The only difference between the two operators is that the group operator is normally used with one relation, while the cogroup operator is used in statements involving two or more relations.

Example: cogroup_data = COGROUP student by age, employee by age;

Output:
(21,{(4,Preethi,Agarwal,21,9848022330,Pune), (1,Rajiv,Reddy,
21,9848022337,Hyderabad)},
{ })
(22,{ (3,Rajesh,Khanna,22,9848022339,Delhi), (2,siddarth,Battacharya,
22,9848022338,Kolkata) },
{ (6,Maggy,22,Chennai),(1,Robin,22,newyork) })
(23,{(6,Archana,Mishra,23,9848022335,Chennai),(5,Trupthi,Mohanthy,
23,9848022336 ,Bhuwaneshwar)},
{(5,David,23,Bhuwaneshwar),(3,Maya,23,Tokyo),(2,BOB,23,Kolkata)})
(24,{(8,Bharathi,Nambiayar,24,9848022333,Chennai),(7,Komal,Nayak,24,9848022334,
trivendram)}, { })
(25,{ }, {(4,Sara,25,London)})

9.**Command:** JOIN

Usage: The JOIN operator is used to combine records from two or more relations. Joins can be of the following types –

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

customers
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
orders
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

i. 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.

Syntax: Relation3_name = JOIN Relation1_name BY key, Relation2_name BY key ;

Example:customers3 = JOIN customers1 BY id, customers2 BY id;

Output:

(1,Ramesh,32,Ahmedabad,2000,1,Ramesh,32,Ahmedabad,2000)
(2,Khilan,25,Delhi,1500,2,Khilan,25,Delhi,1500)
(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)

ii. 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.

-

Syntax:result = JOIN relation1 BY columnname, relation2 BY columnname;

Example: coustomer_orders = JOIN customers BY id, orders BY customer_id;

Output:

(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)

iii.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:
Relation3_name = JOIN Relation1_name BY id LEFT OUTER, Relation2_name BY customer_id;

Example:
outer_left = JOIN customers BY id LEFT OUTER, orders BY customer_id;

Output:

(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:
outer_right = JOIN customers BY id RIGHT, orders BY customer_id;

Example:
outer_right = JOIN customers BY id RIGHT, orders BY customer_id;

Output:
(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:
outer_full = JOIN customers BY id FULL OUTER, orders BY customer_id;

Example:
outer_full = JOIN customers BY id FULL OUTER, orders BY customer_id;

Output:
(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,,,,)

10.**Command:** CROSS

Usage: 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:
Relation3_name = CROSS Relation1_name, Relation2_name;

Example:
cross_data = CROSS customers, orders;

customers
1,Ramesh,32,Ahmedabad,2000.00
2,Khilan,25,Delhi,1500.00

orders
102,2009-10-08 00:00:00,3,3000
100,2009-10-08 00:00:00,3,1500

Output:
(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,2000,100,2009-10-08 00:00:00,3,1500)
(1,Ramesh,32,Ahmedabad,2000,102,2009-10-08 00:00:00,3,3000)

11.**Command:** UNION

Usage: 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:
Relation_name3 = UNION Relation_name1, Relation_name2;

Example:
union_data = UNION Student_data1, Student_data2;

Student_data1
001,Rajiv,Reddy,9848022337,Hyderabad
002,siddarth,Battacharya,9848022338,Kolkata

Student_data2
7,Komal,Nayak,9848022334,trivendram.
8,Bharathi,Nambiayar,9848022333,Chennai.

Output:
001,Rajiv,Reddy,9848022337,Hyderabad
002,siddarth,Battacharya,9848022338,Kolkata
7,Komal,Nayak,9848022334,trivendram.
8,Bharathi,Nambiayar,9848022333,Chennai.

12.**Command:** SPLIT

Usage: The SPLIT operator is used to split a relation into two or more relations.

Syntax:
SPLIT Relation1_name INTO Relation2_name IF (condition1), Relation2_name IF (condition2);

Example:
```
SPLIT student into student_details1 if age<23, student_details2 if (22<age and age>25);
```
Output:
```
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)

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)
```

13.Command: FILTER

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

Syntax:
```
Relation2_name = FILTER Relation1_name BY (condition);
```
Example:
```
filter_data = FILTER student BY city == 'Chennai';
```
Output:
(6,Archana,Mishra,23,9848022335,Chennai)
(8,Bharathi,Nambiayar,24,9848022333,Chennai)

14.Command: DISTINCT

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

Syntax:
```
Relation_name2 = DISTINCT Relatin_name1;
```

Example:
```
distinct_data = DISTINCT student_details;
```

student_details
```
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
```

Output:

```
1,Rajiv,Reddy,9848022337,Hyderabad
2,siddarth,Battacharya,9848022338,Kolkata
3,Rajesh,Khanna,9848022339,Delhi
```

15.Command: FOREACH

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

Syntax:
```
Relation_name2 = FOREACH Relatin_name1 GENERATE (required data);
```

Example:
```
foreach_data = FOREACH student_details GENERATE id,age,city
```

Output:
```
(1,21,Hyderabad)
(2,22,Kolkata)
(3,22,Delhi)
```

16.Command: ORDER BY

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

Syntax:
```
 Relation_name2 = ORDER Relatin_name1 BY (ASC|DESC);
```

Example:
```
order_by_data = ORDER student_details BY age DESC;
```

Output:
```
(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)
```

17.Command: LIMIT

Usage: The LIMIT operator is used to get a limited number of tuples from a relation.

Syntax:
```
 Result = LIMIT Relation_name required number of tuples;
```

Example:
```
 limit_data = LIMIT student_details 2;
```

Output:
1,Rajiv,Reddy,21,9848022337,Hyderabad
2,siddarth,Battacharya,22,9848022338,Kolkata

18. Command: RANK

Usage: RANK operator for extracting top N records.“RANK” operator inserts gaps in between rank positions if ties are observed. However, this behavior can be modified by using the “DENSE” keyword. Below is a sample output if using the DENSE keyword.

Syntax: ranked = RANK input [BY [COL [ASC|DESC]]] [DENSE];

Example:
i.ranked = rank rank_student by GPA desc;
ii.ranked = rank rank_student by GPA desc dense;

Student

NAME	GPA
ANU	5.0
ABI	4.5
UMA	4.0
DEVI	4.5

output i:
(1,ANU,5.0)
(2,DEVI,4.5)
(2,ABI,4.5)
(4,UMA,4.0)

Output ii:
(1,ANU,5.0)
(2,DEVI,4.5)
(2,ABI,4.5)
(3,UMA,4.0)

19. Command: FLATTEN

Usage: FLATTEN is an expression which will eliminate a level of nesting. Given a tuple which contains a bag, FLATTEN will emit several tuples each of which contains one record from the bag.

Syntax:
Relation_name2 = FOREACH Relatin_name1 GENERATE flatten(required data);

Example:
```
 ranked = rank stu_mark by gpa desc dense;
 A = group ranked by rank_stu_mark;
dump A;

(1,{(1,ANU,5.0)})
(2,{(2,ABI,4.5),(2,DEVI,4.5)})
(3,{(3,UMA,4.0)})
B = foreach A generate
flatten(ranked.rank_stu_mark),flatten(ranked.name),flatten(ranked.gpa);
A FLATTEN command would eliminate the inner bags like so:
dump B;
```

```
(1,ANU,5.0)
(2,ABI,4.5)
(2,DEVI,4.5)
(3,UMA,4.0)
```

****

User Defined Functions in Pig

What is Pig UDF ?

Generally Pig having some Built-in functions,we can use that Built-in functions for our Pig Script with out adding any extra code but some times user requirement is not available in that built-in functions at that time user can write some own custom user defined functions called UDF (user defined function).Here is the simple steps of How To Write Pig UDF Example In Java.

Steps to create Pig UDF

sample.txt

```
nivethitha
anitha
savitha
ranjani
poorna
krithika
karthik
sidtharth
```

1. Create simple Udf program in your eclips.

Sample Program to return only the first charecter in UpperCase

```
 package pig_udf;

import java.io.IOException;

import org.apache.pig.EvalFunc;
import org.apache.pig.data.Tuple;
import org.apache.pig.impl.util.WrappedIOException;


public class MyUdf extends EvalFunc<String> {
        public String exec(Tuple input) throws IOException {
                if (input.size() == 0)
                        return null;
                try {
                        String str = (String) input.get(0);
                        char ch = str.toLowerCase().charAt(0);
                        String str1 = String.valueOf(ch);
                        return str1;

                } catch (Exception e) {
                        // TODO: handle exception
                        throw WrappedIOException.wrap(
```

```
                                        "Caught exception processing input row ", e);
                }
        }

}
```

2.Right click on program –> Export –> create Jar

3.Register Jarname;

4.Write The Pig Script

```
Register udf_pig.jar;
A = LOAD 'sample.txt' as (name:chararray);
B = FOREACH A GENERATE pig_udf.MyUdf(name);
DUMP B;
```

Output:

```
(N)
(A)
(S)
(R)
(P)
(K)
(K)
(S)
```