**APACHE PIG**

Start pig shell on your terminal

For Mapreduce

$pig –x mapreduce

For Local Mode

$pig –x local

grunt>

These are the operator we will use in pig:

**Load Operator:**

This operator used to load data from local/hdfs to pig using load keyword.

**Syntax:**

**Relation\_name** = LOAD**'Input file path'** USING **function** as **schema**;

**Explain:**

|  |  |
| --- | --- |
| **Relation\_name** | Mention relation name in which we want to store the data.(Emp) |
| **Input file path** | Mention the HDFS/local directory where the file is stored.(emp.txt) |
| **Function** | Choose a function from the set of load functions like (BinStorage, JsonLoader, PigStorage, TextLoader).(PigStorage) |
| **Schema** | Define the schema of the data like number as int.(id,name etc) |

**Example:**

Suppose you have a file **Student.txt** and you want to load it into pig so command is:

grunt> stu\_load = load '/user/cloudera/Student.txt’ using PigStorage(',') as (sid:int, sname:chararray, address:chararray);

**C:\Users\viks\Desktop\p1.png**

**Dump operator:**

Using dump operator you can see your relation\_name data in which you stored previous.

**Syntax:**

grunt> Dump **Relation\_Name**;

**Example:**

grunt> dump stu\_load

C:\Users\viks\Desktop\p1.png

### Limit

Used to limit the number of outputs to the desired number.

Syntax:

Alias = LIMIT alias n;

Where;

alias : name of the relation.

n : number of tuples to be displayed.

**Example:**

**stu\_limit = limit stu\_load 2;**

C:\Users\viks\Desktop\p1.png

**Store operator:**

This is use for store data into HDFS from pig which is processed in pig.

**Syntax:**

STORE **Relation\_name** INTO **' required\_directory\_path ' [USING function];**

**Explain:**

|  |  |
| --- | --- |
| **required\_directory\_path** | Directory path Where you want to store your pig data. |

**Example:**

Suppose we processed employee data into pig now we want to store this into another file.

grunt> store stu\_load into '/user/cloudera/output';

C:\Users\viks\Desktop\p1.png

**Describe operator:**

This operator basically use to see schema of a relation.

**Syntax:**

grunt> Describe Relation\_name

**Example:**

grunt> describe stu\_load;

C:\Users\viks\Desktop\p1.png

### Order operator

Sorts a relation based on single or multiple fields.

Syntax:

alias = ORDER alias BY {field\_name [ASC | DESC]

Where;

alias : is the relation

ORDER : is the keyword.

BY : is the keyword.

field\_name : column on which you want to sort the relation.

ASC : sort in ascending order

DESC : sort in descending order.

**Example:**

stu\_order = ORDER stu\_load by sid DESC;

**Explain operator:**

It is used to display MapReduce execution plans of a relation.

**Syntax:**

grunt> explain Relation\_name;

**Example:**

grunt> explain stu\_load;

C:\Users\viks\Desktop\p1.png

**Illustrate operator:**

It gives step-by-step execution of a sequence of statements.

**Syntax:**

grunt> illustrate Relation\_name;

**Example:**

grunt> illustrate stu\_load;

**Group operator:**

It is use to group data in one or more relation.

**Syntax:**

grouped\_data = group **Relation\_name** by **field;**

**Explain:**

|  |  |
| --- | --- |
| Field | Particular column like id, name in student table |

**Example:**

stu\_grouped = group stu\_load by stuid;

**Group by multiple column:**

stu\_group\_multi = group stu\_load by (name, address);

C:\Users\viks\Desktop\p1.png

stu\_grouped\_all = group stu\_load all;

C:\Users\viks\Desktop\p1.png

**Join operator:**

It is use to combine two or more relation

1. Self-join0
2. Inner join
3. Outer joins.

**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

**Order.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

**NOTE**: We are join two different tables /files here.

Please note both the files are comma separated and so while loading the data, you will have to use the command PigStorage(',')

First load the file in HDFS

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

C:\Users\viks\Desktop\p1.png

grunt>orders = load 'user/cloudera/orders.txt' using PigStorage(',')as (oid:int, date:chararray, customer\_id:int, amount:int);

C:\Users\viks\Desktop\p1.png

**JOIN OPERATIONS**

Self - join

**Self-join** is used to join a table with itself as if the table were two relations.

in **self-join**, we have to rename the same relation and then apply the join condition.

**Syntax:**

**Relation3\_name =** join **Relation1\_name** BY key**, Relation2\_name** BY key

Here we are join single table but we change relation name.

grunt> cust\_realation1 = load '/user/cloudera/customers.txt' using PigStorage(',')as (id:int, name:chararray, age:int, address:chararray, salary:int);

grunt> cust\_realation2 = load '/user/cloudera/customers.txt' using PigStorage(',')as (id:int, name:chararray, age:int, address:chararray, salary:

C:\Users\viks\Desktop\p1.png

grunt> customers3 = JOIN cust\_relation1 BY id, cust\_relation2 BY id;

C:\Users\viks\Desktop\p1.png

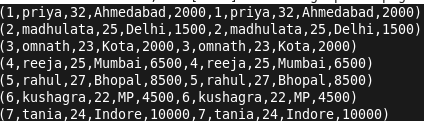
### Verification

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

dump customers3

C:\Users\viks\Desktop\p1.png

OUTPUT



## Inner Join

Inner join returns rows when there is a match in both tables.

### Syntax

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

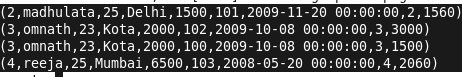
### Example

customer\_orders = JOIN customers BY id, orders BY customer\_id;

### Verification

Verify the relation **customer\_orders** using the **DUMP** operator as shown below.

dump customer\_orders



## 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

**grunt> Relation3\_name = JOIN Relation1\_name BY id LEFT OUTER, Relation2\_name BY customer\_id;**

### Example

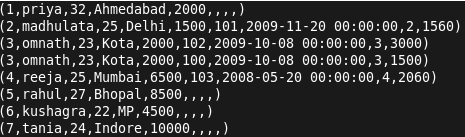
**left\_outer= JOIN customers BY id LEFT OUTER, orders BY customer\_id;**

**C:\Users\viks\Desktop\p1.png**

**Verification**

Verify the relation **left\_outer** using the **DUMP** operator as shown below.

dump left\_outer



## 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.

right\_outer = JOIN customers BY id RIGHT, orders BY customer\_id;

### Example

outer\_right= JOIN customers BY id LEFT RIGHT, orders BY customer\_id;

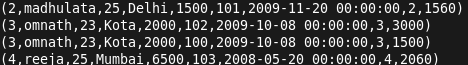
C:\Users\viks\Desktop\p1.png

### Verification

Verify the relation **right\_outer** using the **DUMP** operator as shown below.

dump right\_outer

**OUTPUT**



## Full Outer Join

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

### Syntax

full\_outer = JOIN customers BY id FULL OUTER, orders BY customer\_id;

### Example

full\_order = JOIN customers BY id FULL OUTER, orders BY customer\_id;

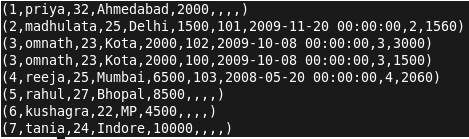
C:\Users\viks\Desktop\p1.png

### Verification

Verify the relation **full\_order** using the **DUMP** operator as shown below.

dump full\_order

**OUTPUT**



## Using Multiple Keys Join

We can perform JOIN operation using multiple keys.

### Syntax:

**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

**LOAD THE FILES IN HDFS**

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

**C:\Users\viks\Desktop\p1.png**

employee\_contact = LOAD '/user/cloudera/employee\_contact.txt' USING PigStorage(',') as (id:int, phone:chararray, email:chararray, city:chararray, jobid:int);

**C:\Users\viks\Desktop\p1.png**

Key\_join = JOIN employee BY (id,jobid), employee\_contact BY (id,jobid);

**output**

### Verification

Verify the relation **Key\_join** using the **DUMP** operator as shown below.

dump Key\_join

