Experiment-10

Write Pig Latin scripts to filter, join, and project your data.

1. **FILTER**

Selects tuples from a relation based on some condition.

***Syntax***

alias = FILTER alias  BY expression;

***Terms***

|  |  |
| --- | --- |
| alias | The name of the relation. |
| BY | Required keyword. |
| expression | A boolean expression. |

***Usage***

Use the FILTER operator to work with tuples or rows of data (if you want to work with columns of data, use the FOREACH...GENERATE operation).

FILTER is commonly used to select the data that you want; or, conversely, to filter out (remove) the data you don’t want.

***Examples***

Suppose we have relation A.

A = LOAD 'data' AS (a1:int,a2:int,a3:int);

DUMP A;

(1,2,3)

(4,2,1)

(8,3,4)

(4,3,3)

(7,2,5)

(8,4,3)

In this example the condition states that if the third field equals 3, then include the tuple with relation X.

X = FILTER A BY f3 == 3;

DUMP X;

(1,2,3)

(4,3,3)

(8,4,3)

In this example the condition states that if the first field equals 8 or if the sum of fields f2 and f3 is not greater than first field, then include the tuple relation X.

X = FILTER A BY (f1 == 8) OR (NOT (f2+f3 > f1));

DUMP X;

(4,2,1)

(8,3,4)

(7,2,5)

(8,4,3)

1. **JOIN (inner)**

Performs an inner join of two or more relations based on common field values.

The JOIN operator to perform an inner, equijoin join of two or more relations based on common field values. Inner joins ignore null keys, so it makes sense to filter them out before the join.

Note the following about the GROUP/COGROUP and JOIN operators:

* The GROUP and JOIN operators perform similar functions. GROUP creates a nested set of output tuples while JOIN creates a flat set of output tuples.
* The GROUP/COGROUP and JOIN operators handle null values differently.

#### *Example*

Suppose we have relations A and B.

A = LOAD 'data1' AS (a1:int,a2:int,a3:int);

DUMP A;

(1,2,3)

(4,2,1)

(8,3,4)

(4,3,3)

(7,2,5)

(8,4,3)

B = LOAD 'data2' AS (b1:int,b2:int);

DUMP B;

(2,4)

(8,9)

(1,3)

(2,7)

(2,9)

(4,6)

(4,9)

In this example relations A and B are joined by their first fields.

X = JOIN A BY a1, B BY b1;

DUMP X;

(1,2,3,1,3)

(4,2,1,4,6)

(4,3,3,4,6)

(4,2,1,4,9)

(4,3,3,4,9)

(8,3,4,8,9)

(8,4,3,8,9)

1. **Projection**

You can choose or project some or all fields from a relation using the foreach ..generate statement. The following code chooses all the fields from the emp relation:

All = foreach emp generate \*

The asterisk represents all the fields of a relation. You may have to use the relation name to access a field if you have bag, tuple, or map data types involved, as shown here:

Empnoname= foreach emp generate emptuple.empno,emptuple.empname;

**Foreach Generate**

Foreach is a multipurpose operator used for projection, applying functions, generating a new schema, and performing nested operations. Often, it is used along with the generate keyword, and it operates on one row at a time from the specified relation. Most of its functionality is similar to a SQL SELECT clause. Foreach, also called the transformation operator, performs transformation jobs.

Here’s the syntax:

relname = FOREACH relname1 { block | nested\_block };

relname1 : Relation name to be used

block is used to process outer bags. You can use the foreach .. generate statement to process data. Nested\_block is used to process inner bags.