Pig Latin 3

Operators

continue ..

FOREACH …. GENERATE

- Applies expression on each input record and generate one or multiple records
- Syntax: FOREACH relation_name GENERATE {….}
- Usage example:
  - projection (select one or multiple fields from a relation)

```
X = FOREACH A GENERATE f1;
X = FOREACH A GENERATE a1, a2;
```

FOREACH …. GENERATE

- If the input to the foreach is tuple, bag or map
  - we can make projection on fields inside the bag

```
X = FOREACH C GENERATE group, A.(a1, a2);

DUMP X;
({(1,2)},1)
({(4,3),(4,2)},4)
({(8,4),(8,3)},8)
```

FOREACH …. GENERATE

- Apply functions
- In the following example, group, then apply count function

```
D = group C by word;
E = foreach D generate COUNT(C), group;
```

- nested example

```
X = FOREACH B {
   FA= FILTER A BY outlink == 'www.xyz.org';
   PA = FA.outlink;
   DA = DISTINCT PA;
   GENERATE group, COUNT(DA);}
}

DUMP X;
(www.aaa.com,0)
(www.ccc.com,0)
(www.ddd.com,1)
(www.www.com,1)
```

```
A = LOAD 'data' AS (url:chararray,outlink:chararray);

DUMP A;
```

url outlink

(www.ccc.com,www.hjk.com)

(www.ddd.com,www.xyz.org)

(www.aaa.com,www.cvn.org)

(www.www.com,www.kpt.net)

(www.www.com,www.xyz.org)

(www.ddd.com,www.xyz.org)

```
B = GROUP A BY url;

DUMP B;
```

(www.aaa.com,{(www.aaa.com,www.cvn.org)})

(www.ccc.com,{(www.ccc.com,www.hjk.com)})

(www.ddd.com,{(www.ddd.com,www.xyz.org),(www.ddd.com,www.xyz.org)})

(www.www.com,{(www.www.com,www.kpt.net),(www.www.com,www.xyz.org)})

Order By

- Order the given relation based on the specified field(s)
- Syntax: ORDER relation_name BY field_1 [DESC | ASC], field_2 [DESC | ASC]

```
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)
X = ORDER A BY a3 DESC;


DUMP X;

(7,2,5)

(8,3,4)

(1,2,3)

(4,3,3)

(8,4,3)

(4,2,1)
```

Limit

- is used to limit number of tuples to be taken from a relation
- Syntax: LIMIT relation_name #number
  - this will take the specified number of records from the relation

Built-in Functions

- Eval function
- String functions
- Date-time function
- Math functions

Eval functions

- AVG()
- COUNT()
- MIN()
- MAX()
- SUM()
- CONCAT()
- DIFF()
- SUBTRACT()

Case Sensitive

AVG()

Keyword. Use ALL if you want all tuples to go to a single group; for example, when doing aggregates across entire relations.

- Can be used after grouping
- Syntax: AVG(expression)
- For example: relation students has schema (id , name, gpa)
  - calculate the average of all students
    - use group all operator: this will give two filed; the first is called all(one value), the second is a bag of all tuples
    - then use the built-in function AVG()
    - MIN(), MAX(), SUM() can be used in the same way

```
grunt> student_details = LOAD 'student_details.txt' USING PigStorage(‘,') as (id:int, name:chararray, gpa:int);

grunt> student_group_all = Group student_details All;

grunt> student_gpa_avg = foreach student_group_all  Generate
   (student_details.name, student_details.gpa), AVG(student_details.gpa);
```

CONCAT

- Can be used to concatenate two fields
- Examples:

```
X = FOREACH A GENERATE CONCAT(a,b);
```

String Functions

- ENDSWITH
- STARTSWITH
- SUBSTRING
- EqualIsIgnoreCase
- UPPER
- LOWER
- REPLACE
- TRIM

ENDSWITH STARTSWITH

- ENDSWITH function, takes two string parameters
  - checks whether the first string ends with the second string
- STARTSWITH
  - checks whether the first string starts with the second string

```
grunt> student_details = LOAD 'student_details.txt' USING PigStorage(‘,') as (id:int, name:chararray, gpa:int);
grunt> student_ends_with = foreach student_details  Generate (name ,  id) , ENDSWITH (name , ’n’);

grunt> student_starts_with = foreach student_details  Generate (name ,  id) , STARTSWITH (name , ’R’);
```

SUBSTRING()

- Return part (substring) of a given string
- Syntax: SUBSTRING (string , startIndex, stopIndex)
  - the following example, will generate the first 3 letters from the name

```
grunt> student_details = LOAD 'student_details.txt' USING PigStorage(‘,') as (id:int, name:chararray, gpa:int);
grunt> student_substring = foreach student_details  Generate (name ,  id) , SUBSTRING (name , 0 , 2);
```

EqualIsIgnoreCase()

- Compares if two strings (ignoring the case) are equal
  - returns boolean value
    - true if equal
    - false if not
- Can be used to search for a name

```
grunt> student_details = LOAD 'student_details.txt' USING PigStorage(‘,') as (id:int, name:chararray, gpa:int);
grunt> student_equal = foreach student_details  Generate (name ,  id) , EqualIsIgnoreCase (name , ‘Ali’);
```

UPPER, LOWER

- UPPER
  - convert letters to upper case
- LOWER
  - convert letters to lower case

REPLACE

- replace characters in a given string with new characters
- Syntax: REPLACE (string , regEXP, newString)
  - string represent the string, field on which to apply the replace
  - regEXP: defines part of the string to be replaced, could be substring, could be regular expression
  - newString: new value to be added on the string

```
grunt> student_equal = foreach student_details  Generate (name ,  id) , REPLACE (city , 'Washington', 'WA');
grunt> student_details = LOAD 'student_details.txt' USING PigStorage(',') as (id:int, name:chararray, city:chararray);
```

TRIM, LTRIM, RTRIM

- TRIM reads a string and removes unwanted spaces before and after the string
- to remove unwanted spaces from the right only, use RTRIM
- to remove unwanted spaces from the left only, use LTRIM

Date-time functions

- toDate(): converts the given date to a given format
  - yyyy/MM/dd, yyyy/MM/dd HH:mm:ss
- getDay(): returns the day
- getMonth(): returns month
- getYear(): reruns year

Store

- store data into HDFS after finish processing
- Syntax: STORE relation INTO ‘path’ USING PigStorage(‘,’)