**Session 4: SCHEDULERS IN YARN & INTRODUCTION TO PIG**

Assignment 4.2

Student Name: Abarajithan SA

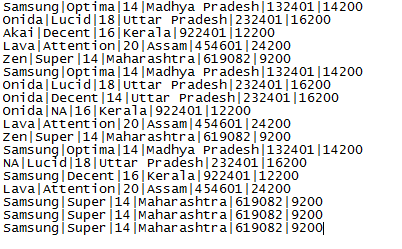
Course: Big Data Hadoop & Spark Training

Start Date:  2017-09-09

End Date:  2017-11-26

**Assignment 4.2** –Create a sample dataset and implement the below Pig commands on the same dataset.

In this assignment we are going to use the below as Input Data set for the first **8** PIG commands.



Contents

[Problem Statement: Create a sample dataset and implement the below Pig commands on the same dataset 3](#_Toc496180664)

[1) Concat(): 4](#_Toc496180665)

[Output: 4](#_Toc496180666)

[2) Tokenize() 4](#_Toc496180667)

[Output: 4](#_Toc496180668)

[3) Sum(): 5](#_Toc496180669)

[Output: 5](#_Toc496180670)

[4) MIN(): 5](#_Toc496180671)

[Output 6](#_Toc496180672)

[5)MAX() 6](#_Toc496180673)

[Output 6](#_Toc496180674)

[6)LIMIT() 6](#_Toc496180675)

[Output 7](#_Toc496180676)

[7)STORE() 7](#_Toc496180677)

[Output 7](#_Toc496180678)

[8)Distinct() 7](#_Toc496180679)

[Output: 8](#_Toc496180680)

[9)Flatten() 8](#_Toc496180681)

[Output 9](#_Toc496180682)

[10)ISEMPTY() 9](#_Toc496180683)

[Output 10](#_Toc496180684)

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

1) Concat

2) Tokenize

3) Sum

4) Min

5) Max

6) Limit

7) Store

8) Distinct

9) Flatten

10) IsEmpty

## 1) Concat():

The CONCAT() function of Pig Latin is used to concatenate two or more expressions of the same type.

Command:

***“baseRelation = LOAD '/home/acadgild/hadoop/television.txt' USING PigStorage('|') AS (company\_name:chararray, unit\_name:chararray, size:int, state:chararray, zip:int, price:int);***

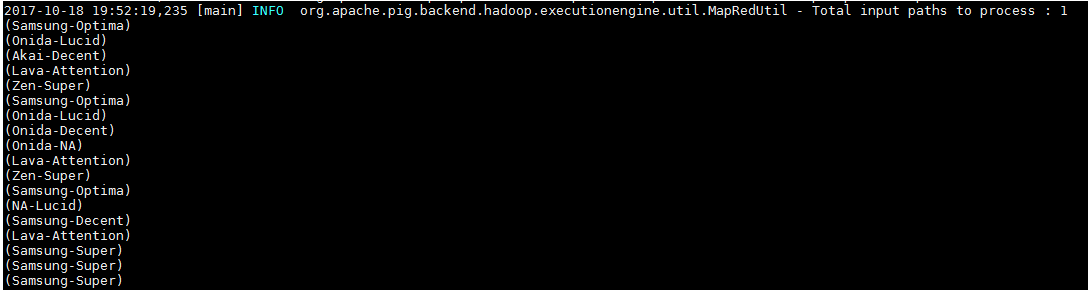
***DUMP baseRelation;***

***DESCRIBE baseRelation;***

***company\_unit\_concat = FOREACH baseRelation GENERATE CONCAT(company\_name, '-', unit\_name);***

***DUMP company\_unit\_concat;”***

### Output:



## 2) Tokenize()

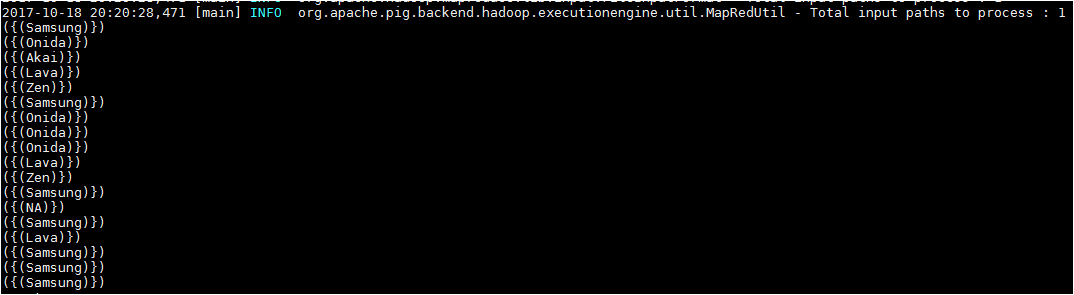
The TOKENIZE() function of Pig Latin is used to split a string (which contains a group of words) in a single tuple and returns a bag which contains the output of the split operation.

Command:

**“company\_unit\_tokenize = FOREACH baseRelation GENERATE TOKENIZE(company\_name);**

**DUMP company\_unit\_tokenize;”**

### Output:



## 3) Sum():

You can use the SUM() function of Pig Latin to get the total of the numeric values of a column in a single-column bag. While computing the total, the SUM() function ignores the NULL values.

Note –

* To get the global sum value, we need to perform a Group All operation, and calculate the sum value using the SUM() function.
* To get the sum value of a group, we need to group it using the Group By operator and proceed with the sum function.

Command:

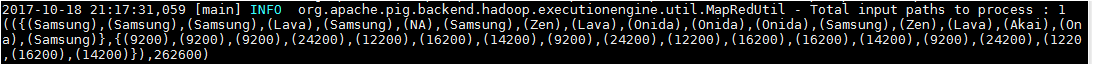
***“company\_price = GROUP baseRelation ALL;”***

### Output:

(all,{(Samsung,Super,14,Maharashtra,619082,9200),(Samsung,Super,14,Maharashtra,619082,9200),(Samsung,Super,14,Maharashtra,619082,9200),(Lava,Attention,20,Assam,454601,24200),(Samsung,Decent,16,Kerala,922401,12200),(NA,Lucid,18,Uttar Pradesh,232401,16200),(Samsung,Optima,14,Madhya Pradesh,132401,14200),(Zen,Super,14,Maharashtra,619082,9200),(Lava,Attention,20,Assam,454601,24200),(Onida,NA,16,Kerala,922401,12200),(Onida,Decent,14,Uttar Pradesh,232401,16200),(Onida,Lucid,18,Uttar Pradesh,232401,16200),(Samsung,Optima,14,Madhya Pradesh,132401,14200),(Zen,Super,14,Maharashtra,619082,9200),(Lava,Attention,20,Assam,454601,24200),(Akai,Decent,16,Kerala,922401,12200),(Onida,Lucid,18,Uttar Pradesh,232401,16200),(Samsung,Optima,14,Madhya Pradesh,132401,14200)})

***“company\_price\_sum = FOREACH company\_price GENERATE (baseRelation.company\_name,baseRelation.price),SUM(baseRelation.price);”***

(({(Samsung),(Samsung),(Samsung),(Lava),(Samsung),(NA),(Samsung),(Zen),(Lava),(Onida),(Onida),(Onida),(Samsung),(Zen),(Lava),(Akai),(Onida),(Samsung)},{(9200),(9200),(9200),(24200),(12200),(16200),(14200),(9200),(24200),(12200),(16200),(16200),(14200),(9200),(24200),(12200),(16200),(14200)}),**262600**)



## 4) MIN():

The MIN() function of Pig Latin is used to get the minimum (lowest) value (numeric or chararray) for a certain column in a single-column bag. While calculating the minimum value, the MIN() function ignores the NULL values.

Note

* To get the global minimum value, we need to perform a Group All operation, and calculate the minimum value using the MIN() function.
* To get the minimum value of a group, we need to group it using the Group By operator and proceed with the minimum function.

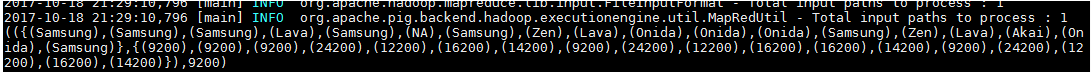
Command:

“***company\_price = GROUP baseRelation ALL;***

***company\_price\_min = FOREACH company\_price GENERATE (baseRelation.company\_name,baseRelation.price),MIN(baseRelation.price);”***

### Output

(({(Samsung),(Samsung),(Samsung),(Lava),(Samsung),(NA),(Samsung),(Zen),(Lava),(Onida),(Onida),(Onida),(Samsung),(Zen),(Lava),(Akai),(Onida),(Samsung)},{(9200),(9200),(9200),(24200),(12200),(16200),(14200),(9200),(24200),(12200),(16200),(16200),(14200),(9200),(24200),(12200),(16200),(14200)}),**9200**)



## 5)MAX()

The Pig Latin MAX() function is used to calculate the highest value for a column (numeric values or chararrays) in a single-column bag. While calculating the maximum value, the Max() function ignores the NULL values.

Note −

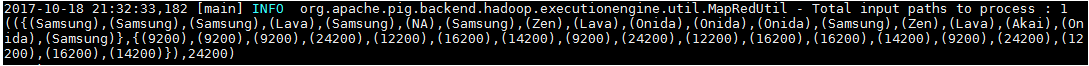
* To get the global maximum value, we need to perform a Group All operation, and calculate the maximum value using the MAX() function.
* To get the maximum value of a group, we need to group it using the Group By operator and proceed with the maximum function.

***“company\_price = GROUP baseRelation ALL;***

***company\_price\_max = FOREACH company\_price GENERATE (baseRelation.company\_name,baseRelation.price),MAX(baseRelation.price);”***

### Output

(({(Samsung),(Samsung),(Samsung),(Lava),(Samsung),(NA),(Samsung),(Zen),(Lava),(Onida),(Onida),(Onida),(Samsung),(Zen),(Lava),(Akai),(Onida),(Samsung)},{(9200),(9200),(9200),(24200),(12200),(16200),(14200),(9200),(24200),(12200),(16200),(16200),(14200),(9200),(24200),(12200),(16200),(14200)}),24200)



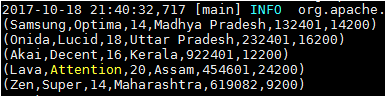
## 6)LIMIT()

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

Command:

***“limit\_data = LIMIT baseRelation 5;”***

### Output



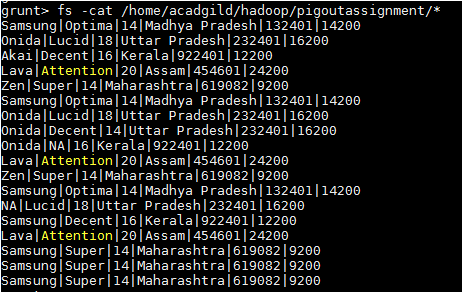
## 7)STORE()

You can store the loaded data in the file system using the **store** operator.

Command:

***“STORE baseRelation INTO '/home/acadgild/hadoop/pigoutassignment' USING PigStorage('|');”***

Output***:***



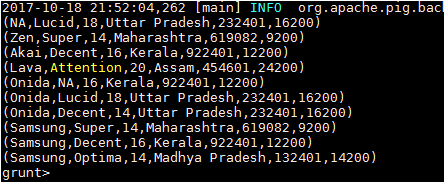
# 8)Distinct()

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

Command:

***“distinct\_data = DISTINCT baseRelation;”***

### Output:



## 9)Flatten()

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

Input Dataset:

A.txt B.txt

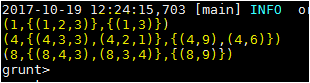
Command:

***“A\_Relation = LOAD '/home/acadgild/hadoop/A.txt' USING PigStorage(',') AS (a1:int,a2:int,a3:int);***

***B\_Relation = LOAD '/home/acadgild/hadoop/B.txt' USING PigStorage(',') AS (b1:int,b2:int);***

***C\_Relation = COGROUP A\_Relation BY a1 inner, B\_Relation BY b1 inner;***

***Dump C\_Relation;***

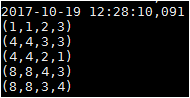
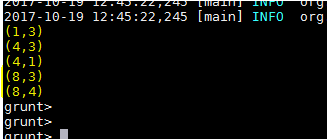


***FLATTEN():***

1. ***X\_Relation = FOREACH C\_Relation GENERATE group, FLATTEN(A\_Relation);***
2. ***X\_Relation = FOREACH C\_Relation GENERATE group, FLATTEN(A\_Relation.a3);***

### Output

In this example the FLATTEN operator is used to eliminate nesting.

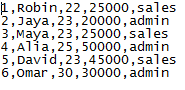
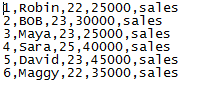
1. 
2. 

## 10)ISEMPTY()

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

Input Data:

**Employee\_Sales.txt** **Employee\_bonus.txt**



Command:

1. ***sales\_relation = LOAD '/home/acadgild/hadoop/Employee\_sales.txt' USING PigStorage(',') AS (id:int, name:chararray, age:int, salary:int, dept:chararray);***

***DUMP sales\_relation;***

1. ***bonus\_relation = LOAD '/home/acadgild/hadoop/Employee\_bonus.txt' USING PigStorage(',') AS (id:int, name:chararray, age:int, salary:int, dept:chararray);***

***DUMP bonus\_relation;***

1. ***cogroup\_data = COGROUP sales\_relation by age, bonus\_relation by age;***

***DUMP cogroup\_data;***

1. ***isempty\_data = FILTER cogroup\_data by IsEmpty(sales\_relation);***

***DUMP isempty\_data;”***

### Output

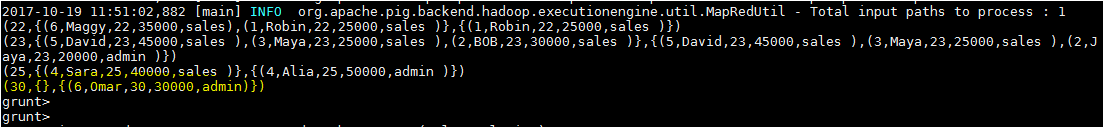
Below is the output of the command number 3.

1. (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)})



Output of the command number 4 **IsEmpty(),**The **sales\_relation** holds the tuples that are not there in the relation **bonus\_relation**.

