Load the sample data into hdfs using

**hadoop fs -put /home/acadgild/Pigdataset.txt /user/acadgild/employee**

**1. CONCAT**

**SYNTAX:**

grunt> CONCAT (expression, expression, [...expression])

**1**. student\_details = LOAD 'hdfs://localhost:9000/user/acadgild/employee/Pigdataset.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray, gpa:int);

**2.** Dump student\_details;

**3.** Describe student\_details;

**4**.grunt> student\_name\_concat = foreach student\_details Generate CONCAT (firstname, lastname);

**5.** Dump student\_name\_concat;

**2. TOKENIZE**

**SYNTAX:**

grunt> TOKENIZE(expression [, 'field\_delimiter'])

We can use the TOKENIZE() function to split a string. As an example let us split the name using this function as shown below.

**1**. student\_details = LOAD 'hdfs://localhost:9000//user/acadgild/employee/Pigdataset.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, city:chararray);

**2**. student\_name\_tokenize = foreach student\_details Generate TOKENIZE(name);

**3**. grunt> Dump student\_name\_tokenize;

**3.SUM**

**SYNTAX:**

grunt> SUM(expression)

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.

**1.** employee\_data = LOAD 'hdfs://localhost:9000//user/acadgild/employee/Pigdataset1.txt' USING PigStorage(',') as (id:int, name:chararray, workdate:chararray, daily\_typing\_pages:int);

Let us group the relation employee\_data using the Group All operator, and store the result in the relation named employee\_group as shown below.

**2**.employee\_group = Group employee\_data all;

**3.** Dump employee\_group;

Let us now calculate the global sum of the pages typed daily.

**4.** student\_workpages\_sum = foreach employee\_group Generate

(employee\_data.name,employee\_data.daily\_typing\_pages),SUM(employee\_data.daily\_typing\_pages);

**5.** Dump student\_workpages\_sum;

**4.MIN**

**SYNTAX:**

grunt> MIN(expression)

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.

**1.** student\_details = LOAD 'hdfs://localhost:9000//user/acadgild/employee/Pigdataset.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray, gpa:int);

Let us group the relation student\_details using the Group All operator, and store the result in the relation named student\_group\_all as shown below.

**2**. student\_group\_all = Group student\_details All;

**3**. Dump student\_group\_all;

Let us now calculate the global minimum of GPA, i.e., minimum among the GPA values of all the students using the MIN() function as shown below.

**4.** student\_gpa\_min = foreach student\_group\_all Generate

(student\_details.firstname, student\_details.gpa), MIN(student\_details.gpa);

**5.** Dump student\_gpa\_min;

**5.MAX**

SYNTAX:

grunt> Max(expression)

The Pig Latin MAX() function is used to calculate the highest value for a column (numeric values or chararrays) in a single-column bag.

**1.**student\_details = LOAD 'hdfs://localhost:9000//user/acadgild/employee/Pigdataset.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray, gpa:int);

Let us group the relation student\_details using the Group All operator, and store the result in the relation named student\_group\_all as shown below.

**2**. student\_group\_all = Group student\_details All;

**3.** Dump student\_group\_all;

Let us now calculate the global maximum of GPA, i.e., maximum among the GPA values of all the students using the MAX() function as shown below.

**4** student\_gpa\_max = foreach student\_group\_all Generate

(student\_details.firstname, student\_details.gpa), MAX(student\_details.gpa);

**5.**Dump student\_gpa\_max;

**6.LIMIT**

**SYNTAX:**

grunt> Result = LIMIT Relation\_name required number of tuples;

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

**1**. student\_details = LOAD 'hdfs://localhost:9000//user/acadgild/employee/Pigdataset.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray,age:int, phone:chararray, city:chararray);

Now, let’s sort the relation in descending order based on the age of the student and store it into another relation named limit\_data using the ORDER BY operator as shown below.

**2.** limit\_data = LIMIT student\_details 4;

**3.** Dump limit\_data;

**7.STORE**

**SYNTAX:**

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

**1.** student = LOAD 'hdfs://localhost:9000//user/acadgild/employee/Pigdataset.txt'

USING PigStorage(',')

**2.** STORE student INTO ' hdfs://localhost:9000/pig\_Output/ ' USING PigStorage (',');

**VERIFICATION:**

$ hdfs dfs -cat 'hdfs://localhost:9000//user/acadgild/employee/Pigoutput/part-m-00000'

**8.DISTINCT**

**SYNTAX:**

grunt> Relation\_name2 = DISTINCT Relatin\_name1;

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

**1**. student\_details = LOAD 'hdfs://localhost:9000//user/acadgild/employee/Pigdataset.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, phone:chararray, city:chararray);

Let us now remove the redundant (duplicate) tuples from the relation named student\_details using the DISTINCT operator, and store it as another relation named distinct\_data as shown below.

**2.** distinct\_data = DISTINCT student\_details;

**3.** Dump distinct\_data;

**9.The FLATTEN**

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.

Ex:

**1.**Details= LOAD 'hdfs://localhost:9000//user/acadgild/employee/Pigdataset.txt' USING PigStorage(',') as (id:int, gpa:int);

**2.**Grp= group Details by id;

**3.**Each = foreach Grp generate group as id, Details as gpa;

**4.**describe Each;

-- Each: {a: int,b: {(a: int,b: int)}}

-- Each is like (1, {(1,2), (1,3), (1,4)} )

**10.ISEMPTY**

**SYNTAX**

grunt> IsEmpty(expression)

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

Create 2 files named **sales** and **bonus.tx**t and load into hdfs

**1**. emp\_sales = LOAD 'hdfs://localhost:9000/pig\_data/emp\_sales.txt' USING PigStorage(',')

as (sno:int, name:chararray, age:int, salary:int, dept:chararray);

**2**. emp\_bonus = LOAD 'hdfs://localhost:9000/pig\_data/emp\_bonus.txt' USING PigStorage(',') as (sno:int, name:chararray, age:int, salary:int, dept:chararray);

Let us now group the records/tuples of the relations emp\_sales and emp\_bonus with the key age, using the cogroup operator as shown below.

**3**. cogroup\_data = COGROUP emp\_sales by age, emp\_bonus by age;

**4.** Dump cogroup\_data;

**5.** grunt> isempty\_data = filter cogroup\_data by IsEmpty(emp\_sales);

**6.** Dump isempty\_data;