BIG DATA TECHNOLOGIES LAB

(ACADEMIC YEAR : 2017-2018)

I SEMESTER

ASSIGNMENT 4

TOPIC : PIG PRACTICE SESSION II

ASHISH CHANDRAKANT DUSANE

M. TECH. (ACDS)

COMPUTER ENGG. DEPARTMENT

{ PRN : 170101261004 }

~: Practice Session II :~

Filter Operator :~

```
student_details3 = LOAD '/home/student/Documents/Pig/student_details.txt' USING
PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int,
phone:chararray, city:chararray);
```

Output :

(1,Rajiv,Reddy,21,9848022337,Hyderabad)
(2,siddarth,Battacharya,22,9848022338,Kolkata)
(3,Rajesh,Khanna,22,9848022339,Delhi)
(4,Preethi,Agarwal,21,9848022330,Pune)
(5,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar)
(6,Archana,Mishra,23,9848022335,Chennai)
(7,Komal,Nayak,24,9848022334,trivendram)
(8,Bharathi,Nambiayar,24,9848022333,Chennai)

```
filter_data = FILTER student_details3 BY city == 'Chennai';
Dump filter_data;
```

Output :

(8,Bharathi,Nambiayar,24,9848022333,Chennai)

Distinct Operator :~

```
distinct_data = DISTINCT student_details3;
Dump distinct_data;
```

(Duplicate Recors deleted)

Output :

(1,Rajiv,Reddy,21,9848022337,Hyderabad)
(2,siddarth,Battacharya,22,9848022338,Kolkata)
(3,Rajesh,Khanna,22,9848022339,Delhi)
(4,Preethi,Agarwal,21,9848022330,Pune)
(5,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar)
(6,Archana,Mishra,23,9848022335,Chennai)
(7,Komal,Nayak,24,9848022334,trivendram)
(8,Bharathi,Nambiayar,24,9848022333,Chennai)

Foreach Operator :~

```
foreach_data = FOREACH student_details GENERATE id,age,city;
Dump foreach_data;
```

Output :
(1,21,Hyderabad)
(2,22,Kolkata)
(3,22,Delhi)
(4,21,Pune)
(5,23,Bhuwaneshwar)
(6,23,Chennai)
(7,24,trivendram)
(8,24,Chennai)

# Order Operator :~
order_by_data = ORDER student_details BY age DESC;
Dump order_by_data;

Output :
(8,Bharathi,Nambiayar,24,9848022333,Chennai)
(7,Komal,Nayak,24,9848022334,trivendram)
(6,Archana,Mishra,23,9848022335,Chennai)
(5,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar)
(3,Rajesh,Khanna,22,9848022339,Delhi)
(2,siddarth,Battacharya,22,9848022338,Kolkata)
(4,Preethi,Agarwal,21,9848022330,Pune)
(1,Rajiv,Reddy,21,9848022337,Hyderabad)

# Limit Operator :~
limit_data = LIMIT student_details 4;
Dump limit_data;

Output :
(1,Rajiv,Reddy,21,9848022337,Hyderabad)
(2,siddarth,Battacharya,22,9848022338,Kolkata)
(3,Rajesh,Khanna,22,9848022339,Delhi)
(4,Preethi,Agarwal,21,9848022330,Pune)

# Average Function Operator :~
student_details = LOAD '/home/student/Documents/Pig/student_details.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray, gpa:int);

Output :
(1,Rajiv,Reddy,21,9848022337,Hyderabad,89)
(2,siddarth,Battacharya,22,9848022338,Kolkata,78)
(3,Rajesh,Khanna,22,9848022339,Delhi,90)
(4,Preethi,Agarwal,21,9848022330,Pune,93)

(5,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar,75)
(6,Archana,Mishra,23,9848022335,Chennai,87)
(7,Komal,Nayak,24,9848022334,trivendram,83)
(8,Bharathi,Nambiayar,24,9848022333,Chennai,72)

student_group_all = Group student_details All;
student_gpa_avg = foreach student_group_all GENERATE
(student_details.firstname, student_details.gpa), AVG(student_details.gpa);
Dump student_gpa_avg;

Output :
(({(Bharathi),(Komal),(Archana),(Trupthi),(Preethi),(Rajesh),(siddarth),(Rajiv)}
,{(72),(83),(87),(75),(93),(90),(78),(89)}),83.375)

# BagToString Function :~
# Syntax : BagToString(vals:bag [, delimiter:chararray])
dob = LOAD '/home/student/Documents/Pig/dob.txt' USING PigStorage(',') as
(day:int,month:int, year:int);
dump dob;

Output :
(22,3,1990)
(23,11,1989)
(1,3,1998)
(2,6,1980)
(26,9,1989)

group_dob = Group dob All;
Dump group_dob;

Output :
(all,{(26,9,1989),(2,6,1980),(1,3,1998),(23,11,1989),(22,3,1990)})

dob_string = foreach group_dob Generate BagToString(dob);
Dump dob_string;

Output
(26_9_1989_2_6_1980_1_3_1998_23_11_1989_22_3_1990)

# Concat Function :~
student_name_concat = foreach student_details Generate CONCAT (firstname,
lastname);

Output :
(RajivReddy)

(siddarthBattacharya)
(RajeshKhanna)
(PreethiAgarwal)
(TrupthiMohanthy)
(ArchanaMishra)
(KomalNayak)
(BharathiNambiayar)

student_name_concat = foreach student_details Generate CONCAT(firstname, '_',lastname);
Dump student_name_concat;

Output :
(Rajiv_Reddy)
(siddarth_Battacharya)
(Rajesh_Khanna)
(Preethi_Agarwal)
(Trupthi_Mohanthy)
(Archana_Mishra)
(Komal_Nayak)
(Bharathi_Nambiayar)

# Count Function :~
student_details1 = LOAD '/home/student/Documents/Pig/student_cgpa.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray, gpa:int);

Output :
(1,Rajiv,Reddy,21,9848022337,Hyderabad,89)
(2,siddarth,Battacharya,22,9848022338,Kolkata,78)
(3,Rajesh,Khanna,22,9848022339,Delhi,90)
(4,Preethi,Agarwal,21,9848022330,Pune,93)
(5,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar,75)
(6,Archana,Mishra,23,9848022335,Chennai,87)
(7,Komal,Nayak,24,9848022334,trivendram,83)
(8,Bharathi,Nambiayar,24,9848022333,Chennai,72)

student_group_all = Group student_details1 All;
student_count = foreach student_group_all Generate COUNT(student_details1.gpa);

Output :
(8)

Count_Star Function :~

student_count = foreach student_group_all Generate
COUNT_STAR(student_details1.gpa);
Dump student_count;
count NULL values also

Output :
(8)

Diff Function :~
hdfs dfs -put emp_sales.txt /pig
hdfs dfs -put emp_bonus.txt /pig

emp_sales = LOAD '/home/student/Documents/Pig/emp_sales.txt' USING
PigStorage(',') as (sno:int, name:chararray, age:int, salary:int, dept:chararray);

emp_bonus = LOAD '/home/student/Documents/Pig/emp_bonus.txt' USING
PigStorage(',') as (sno:int, name:chararray, age:int, salary:int, dept:chararray);

cogroup_data = COGROUP emp_sales by sno, emp_bonus by sno;
dump cogroup_data;

Output :
(1,{(1,Robin,22,25000,sales)},{(1,Robin,22,25000,sales)})
(2,{(2,BOB,23,30000,sales)},{(2,Jaya,23,20000,admin)})
(3,{(3,Maya,23,25000,sales)},{(3,Maya,23,25000,sales)})
(4,{(4,Sara,25,40000,sales)},{(4,Alia,25,50000,admin)})
(5,{(5,David,23,45000,sales)},{(5,David,23,45000,sales)})
(6,{(6,Maggy,22,35000,sales)},{(6,Omar,30,30000,admin)})

diff_data = FOREACH cogroup_data GENERATE DIFF(emp_sales,emp_bonus);
Dump diff_data;

Output :
({})
({(2,BOB,23,30000,sales),(2,Jaya,23,20000,admin)})
({})
({(4,Sara,25,40000,sales),(4,Alia,25,50000,admin)})
({})
({(6,Maggy,22,35000,sales),(6,Omar,30,30000,admin)})

# isEmpty Function :~
cogroup_data = COGROUP emp_sales by age, emp_bonus by age;
dump cogroup_data;

Output :
(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)})

isempty_data = filter cogroup_data by IsEmpty(emp_sales);
Dump isempty_data;

Output :
(30,{},{(6,Omar,30,30000,admin)})

# Max Function and Min Function :~
student_cgpa = LOAD '/home/student/Documents/Pig/student_cgpa.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray, gpa:int);
student_group_all = Group student_cgpa All;
student_gpa_max = foreach student_group_all Generate (student_details.firstname, student_details.gpa), MAX(student_details.gpa);
student_gpa_min = foreach student_group_all Generate (student_details.firstname, student_details.gpa), MIN(student_details.gpa);

Dump student_gpa_max;
Output :
(({(Bharathi),(Komal),(Archana),(Trupthi),(Preethi),(Rajesh),(siddarth),(Rajiv)} ,{ (72) , (83) , (87) , (75) , (93) , (90) , (78) , (89) }) ,93)

Dump student_gpa_min;
Output :
(({(Bharathi),(Komal),(Archana),(Trupthi),(Preethi),(Rajesh),(siddarth),(Rajiv)} ,{ (72) , (83) , (87) , (75) , (93) , (90) , (78) , (89) }) ,72)

Size :~
Syntax
employee_data = LOAD '/home/student/Documents/Pig/employee.txt' USING PigStorage(',') as (id:int, name:chararray, workdate:chararray, aily_typing_pages:int);

size = FOREACH employee_data GENERATE SIZE(name);
Dump size;
Output :
(4)
(3)
(4)
(4)

(4)
(4)
(4)

# Subtract Function :~
emp_sales = LOAD '/home/student/Documents/Pig/emp_sales.txt' USING PigStorage(',') as (sno:int, name:chararray, age:int, salary:int, dept:chararray);

emp_bonus = LOAD '/home/student/Documents/Pig/emp_bonus.txt' USING PigStorage(',') as (sno:int, name:chararray, age:int, salary:int, dept:chararray);

cogroup_data = COGROUP emp_sales by sno, emp_bonus by sno;
Dump cogroup_data;

Output :
(1,{(1,Robin,22,25000,sales)},{(1,Robin,22,25000,sales)})
(2,{(2,BOB,23,30000,sales)},{(2,Jaya,23,20000,admin)})
(3,{(3,Maya,23,25000,sales)},{(3,Maya,23,25000,sales)})
(4,{(4,Sara,25,40000,sales)},{(4,Alia,25,50000,admin)})
(5,{(5,David,23,45000,sales)},{(5,David,23,45000,sales)})
(6,{(6,Maggy,22,35000,sales)},{(6,Omar,30,30000,admin)})

sub_data1 = FOREACH cogroup_data GENERATE SUBTRACT(emp_sales, emp_bonus);
sub_data2 = FOREACH cogroup_data GENERATE SUBTRACT(emp_bonus, emp_sales);

Output :
Dump sub_data1;
({})
({(2,BOB,23,30000,sales)})
({})
({(4,Sara,25,40000,sales)})
({})
({(6,Maggy,22,35000,sales)})

Dump sub_data2;
({})
({(2,Jaya,23,20000,admin)})
({})
({(4,Alia,25,50000,admin)})
({})
({(6,Omar,30,30000,admin)})

SUM Function :~

```
employee_data = LOAD '/home/student/Documents/Pig/employee.txt' USING
PigStorage(',') as (id:int, name:chararray, workdate:chararray,
daily_typing_pages:int);
```

Output :

(1,John,2007-01-24,250)
(2,Ram,2007-05-27,220)
(3,Jack,2007-05-06,170)
(3,Jack,2007-04-06,100)
(4,Jill,2007-04-06,220)
(5,Zara,2007-06-06,300)
(5,Zara,2007-02-06,350)

```
employee_group = Group employee_data ALL;
student_workpages_sum = foreach employee_group Generate
(employee_data.name,employee_data.daily_typing_pages),SUM(employee_data.dail
y_typing_pages);
Dump student_workpages_sum;
```

Output :

(({(Zara),(Zara),(Jill),(Jack),(Jack),(Ram),(John)},{(350),(300),(220),(100),(170),(220),(250)}),1610)

TextLoader Function :~

```
details = LOAD '/home/student/Documents/Pig/hadoop_logs.txt' USING
TextLoader();
dump details;
```

Output :

(2014-10-20 14:50:08,367 INFO org.apache.hadoop.hdfs.server.namenode.NNStorageRetentionManager: Going to retain 2 images with txid >= 0)
(2014-10-20 14:50:23,090 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Rescanning after 30000 milliseconds)
(2014-10-20 14:50:23,090 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Scanned 0 directive(s) and 0 block(s) in 0 millisecond(s).)
(2014-10-20 14:50:53,090 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Rescanning after 30000 milliseconds)
(2014-10-20 14:50:53,091 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Scanned 0 directive(s) and 0 block(s) in 1 millisecond(s).)

(2014-10-20 14:51:23,090 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Rescanning after 30000 milliseconds)
(2014-10-20 14:51:23,090 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Scanned 0 directive(s) and 0 block(s) in 1 millisecond(s).)
(2014-10-20 14:51:53,090 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Rescanning after 30000 milliseconds)
(2014-10-20 14:51:53,090 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Scanned 0 directive(s) and 0 block(s) in 0 millisecond(s).)
(2014-10-20 14:52:23,090 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Rescanning after 30000 milliseconds)
(2014-10-20 14:52:23,091 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Scanned 0 directive(s) and 0 block(s) in 1 millisecond(s).)
(2014-10-20 14:52:53,090 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Rescanning after 30000 milliseconds)
(2014-10-20 14:52:53,111 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Scanned 0 directive(s) and 0 block(s) in 21 millisecond(s).)
(2014-10-20 14:53:23,090 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Rescanning after 30000 milliseconds)
(2014-10-20 14:53:23,090 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Scanned 0 directive(s) and 0 block(s) in 0 millisecond(s).)
(2014-10-20 14:53:53,090 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Rescanning after 30001 milliseconds)
(2014-10-20 14:53:53,091 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Scanned 0 directive(s) and 0 block(s) in 0 millisecond(s).)
(2014-10-20 14:54:23,091 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Rescanning after 30000 milliseconds)
(2014-10-20 14:54:23,184 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor: Scanned 0 directive(s) and 0 block(s) in 94 millisecond(s).)
(2014-10-20 14:54:53,091 INFO org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor:

Rescanning after 30000 milliseconds)
(2014-10-20 14:54:53,142 INFO
org.apache.hadoop.hdfs.server.blockmanagement.CacheReplicationMonitor:
Scanned 0 directive(s) and 0 block(s) in 51 millisecond(s).)
(2014-10-20 20:26:49,988 INFO
org.apache.hadoop.hdfs.server.namenode.NameNode: STARTUP_MSG:)

# TOBAG Function :~
emp_data = LOAD '/home/student/Documents/Pig/employee_details.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, city:chararray);

Output :
(1,Robin,22,newyork)
(2,BOB,23,Kolkata)
(3,Maya,23,Tokyo)
(4,Sara,25,London)
(5,David,23,Bhuwaneshwar)
(6,Maggy,22,Chennai)

tobag = FOREACH emp_data GENERATE TOBAG (id,name,age,city);
Dump tobag;

Output :
({(1),(Robin),(22),(newyork)})
({(2),(BOB),(23),(Kolkata)})
({(3),(Maya),(23),(Tokyo)})
({(4),(Sara),(25),(London)})
({(5),(David),(23),(Bhuwaneshwar)})
({(6),(Maggy),(22),(Chennai)})

TOP Function :~
emp_data = LOAD '/home/student/Documents/Pig/employee_details.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, city:chararray);
emp_group = Group emp_data BY age;
Dump emp_group;

Output :
(22,{(6,Maggy,22,Chennai),(1,Robin,22,newyork)})
(23,{(5,David,23,Bhuwaneshwar),(3,Maya,23,Tokyo),(2,BOB,23,Kolkata)})
(25,{(4,Sara,25,London)})

data_top = FOREACH emp_group
{top = TOP(2, 0, emp_data);
GENERATE top;}
In the above example we are retrieving the top 2 tuples of a group having greater id.

Since we are retrieving top 2 tuples based on the id, we are passing the index of the column name "id" as second parameter of TOP() function. (index of id is 0)

#TOTUPLE Function :~

```
emp_data = LOAD '/home/student/Documents/Pig/employee_details.txt' USING
PigStorage(',') as (id:int, name:chararray, age:int, city:chararray);

totuple = FOREACH emp_data GENERATE TOTUPLE (id,name,age);
Dump totuple;
```

Output :

((1,Robin,22))
((2,BOB,23))
((3,Maya,23))
((4,Sara,25))
((5,David,23))
((6,Maggy,22))

TOMAP Function :~

```
emp_data = LOAD '/home/student/Documents/Pig/employee_details.txt' USING
PigStorage(',') as (id:int, name:chararray, age:int, city:chararray);
tomap = FOREACH emp_data GENERATE TOMAP(name, age);
dump tomap;
```

Output :

([Robin#22])
([BOB#23])
([Maya#23])
([Sara#25])
([David#23])
([Maggy#22])

ENDSWITH and STARTSWITH Function :~

```
emp_data = LOAD '/home/student/Documents/Pig/employee_details.txt' USING
PigStorage(',') as (id:int, name:chararray, age:int, city:chararray);

emp_endswith = FOREACH emp_data GENERATE (id,name),ENDSWITH ( name, 'n' );
dump emp_endswith;
```

Output :

((1,Robin),true)
((2,BOB),false)
((3,Maya),false)

((4,Sara),false)
((5,David),false)
((6,Maggy),false)

startswith_data = FOREACH emp_data GENERATE (id,name), STARTSWITH (name,'Ro');
dump startswith_data;

Output :
((1,Robin),true)
((2,BOB),false)
((3,Maya),false)
((4,Sara),false)
((5,David),false)
((6,Maggy),false)

# SUBSTRING Function :~
substring_data = FOREACH emp_data GENERATE (id,name), SUBSTRING (name, 0, 2);
Dump substring_data;

Output :
((1,Robin),Ro)
((2,BOB),BO)
((3,Maya),Ma)
((4,Sara),Sa)
((5,David),Da)
((6,Maggy),Ma)

substring_data = FOREACH emp_data GENERATE (id,name), SUBSTRING (name, 0, 3);
Dump substring_data;

Output :
((1,Robin),Rob)
((2,BOB),BOB)
((3,Maya),May)
((4,Sara),Sar)
((5,David),Dav)
((6,Maggy),Mag)

# EqualsIgnoreCase Function :~
equals_data = FOREACH emp_data GENERATE (id,name), EqualsIgnoreCase(name, 'Robin');
Dump equals_data;

Output :
((1,Robin),true)
((2,BOB),false)
((3,Maya),false)
((4,Sara),false)
((5,David),false)
((6,Maggy),false)

# IndexOf Function :~
indexof_data = FOREACH emp_data GENERATE (id,name), INDEXOF(name,'r',0);
Dump indexof_data;

Output :
((1,Robin),-1)
((2,BOB),-1)
((3,Maya),-1)
((4,Sara),2)
((5,David),-1)
((6,Maggy),-1)

The above statement parses the name of each employee and returns the index value at which the letter ‘r’ occurred for the first time. If the name doesn’t contain the letter ‘r’ it returns the value -1

# Last_Index_of :~
last_index_data = FOREACH emp_data GENERATE (id,name),
LAST_INDEX_OF(name, 'g');
Dump last_index_data;

Output :
((1,Robin),-1)
((2,BOB),-1)
((3,Maya),-1)
((4,Sara),-1)
((5,David),-1)
((6,Maggy),3)

The above statement parses the name of each employee from the end and returns the index value at which the letter ‘g’ occurred for the first time. If the name doesn’t contain the letter ‘g’ it returns the value −1

# LCFIRST, UCFIRST, LOWER, UPPER :~
Lcfirst_data = FOREACH emp_data GENERATE (id,name), LCFIRST(name);
ucfirst_data = FOREACH emp_data GENERATE (id,city), UCFIRST(city);

```
upper_data = FOREACH emp_data GENERATE (id,name), UPPER(name);
lower_data = FOREACH emp_data GENERATE (id,name), LOWER(name);
Dump Lcfirst_data;
```

Output :

((1,Robin),robin)
((2,BOB),bOB)
((3,Maya),maya)
((4,Sara),sara)
((5,David),david)
((6,Maggy),maggy)

Dump ucfirst_data;
Output :

((1,newyork),Newyork)
((2,Kolkata),Kolkata)
((3,Tokyo),Tokyo)
((4,London),London)
((5,Bhuwaneshwar),Bhuwaneshwar)
((6,Chennai),Chennai)

Dump lower_data;
Output :

((1,Robin),robin)
((2,BOB),bob)
((3,Maya),maya)
((4,Sara),sara)
((5,David),david)
((6,Maggy),maggy)

Dump upper_data;
Output :

((1,Robin),ROBIN)
((2,BOB),BOB)
((3,Maya),MAYA)
((4,Sara),SARA)
((5,David),DAVID)
((6,Maggy),MAGGY)

Replace Function :~

```
emp_data = LOAD '/home/student/Documents/Pig/employee_details.txt' USING
PigStorage(',') as (id:int, name:chararray, age:int, city:chararray);
replace_data = FOREACH emp_data GENERATE
(id,city),REPLACE(city,'Bhuwaneshwar','Bhuw');
```

Dump replace_data;

Output :
((1,newyork),newyork)
((2,Kolkata),Kolkata)
((3,Tokyo),Tokyo)
((4,London),London)
((5,Bhuwaneshwar),Bhuw)
((6,Chennai),Chennai)

# STRSPLIT Function :~
emp_data = LOAD '/home/student/Documents/Pig/emp_split.txt USING PigStorage(',') as (id:int, name:chararray, age:int, city:chararray);
strsplit_data = FOREACH emp_data GENERATE (id,name), STRSPLIT (name,'_',2);

Output :
((1,Robin),(Robin))
((2,BOB),(BOB))
((3,Maya),(Maya))
((4,Sara),(Sara))
((5,David),(David))
((6,Maggy),(Maggy))

# Date Functions :~
date_data = LOAD '/home/student/Documents/Pig/date.txt' USING PigStorage(',') as (id:int,date:chararray);
todate_data = foreach date_data generate ToDate(date,'yyyy/MM/dd HH:mm:ss') as (date_time:DateTime >);
Dump todate_data;

currenttime_data = foreach todate_data generate CurrentTime();
Dump currenttime_data;
todate_data = foreach date_data generate ToDate(date,'yyyy/MM/dd HH:mm:ss') as (date_time:DateTime);
Dump todate_data;

getday_data = foreach todate_data generate(date_time), GetDay(date_time);
Dump getday_data;
todate_data = foreach date_data generate ToDate(date,'yyyy/MM/dd HH:mm:ss') as (date_time:DateTime);
gethour_data = foreach todate_data generate (date_time), GetHour(date_time);

Mathematical Functions :~
ABS Function :

math_data = LOAD '/home/student/Documents/Pig/math.txt' USING PigStorage(',') as (data:float);

Output :
(5.0)
(16.0)
(9.0)
(2.5)
(5.9)
(3.1)

abs_data = foreach math_data generate (data), ABS(data);
Dump abs_data;

Output :
(5.0,5.0)
(16.0,16.0)
(9.0,9.0)
(2.5,2.5)
(5.9,5.9)
(3.1,3.1)

# Cube and Square Root Function :~
cbrt_data = foreach math_data generate (data), CBRT(data);

Output :
(5.0,1.709975946676697)
(16.0,2.5198420997897464)
(9.0,2.080083823051904)
(2.5,1.3572088082974532)
(5.9,1.8069688790571206)
(3.1,1.4580997208745365)

sqrt_data = foreach math_data generate (data), SQRT(data);

Output :
(5.0,2.23606797749979)
(16.0,4.0)
(9.0,3.0)
(2.5,1.5811388300841898)
(5.9,2.4289915799292987)
(3.1,1.76068165908337)

# Trigometric Functions :~
acos_data = foreach math_data generate (data), ACOS(data);

Output :
(5.0,NaN)
(16.0,NaN)
(9.0,NaN)
(2.5,NaN)
(5.9,NaN)
(3.1,NaN)

asin_data = foreach math_data generate (data), ASIN(data);
# Output :
(5.0,NaN)
(16.0,NaN)
(9.0,NaN)
(2.5,NaN)
(5.9,NaN)
(3.1,NaN)

atan_data = foreach math_data generate (data), ATAN(data);

Output :
(5.0,1.373400766945016)
(16.0,1.5083775167989393)
(9.0,1.460139105621001)
(2.5,1.1902899496825317)
(5.9,1.4029004062076729)
(3.1,1.2587541962439153)

cos_data = foreach math_data generate (data), COS(data);

Output :
(5.0,0.28366218546322625)
(16.0,-0.9576594803233847)
(9.0,-0.9111302618846769)
(2.5,-0.8011436155469337)
(5.9,0.9274784663996888)
(3.1,-0.999135146307834)

cosh_data = foreach math_data generate (data), COSH(data);

Output :
(5.0,74.20994852478785)
(16.0,4443055.260253992)
(9.0,4051.5420254925943)
(2.5,6.132289479663686)
(5.9,182.52012106128686)

(3.1,11.121499185584959)

sin_data = foreach math_data generate (data), SIN(data);

Output :
(5.0,-0.9589242746631385)
(16.0,-0.2879033166650653)
(9.0,0.4121184852417566)
(2.5,0.5984721441039564)
(5.9,-0.3738765763789988)
(3.1,0.04158075771824354)

sinh_data = foreach math_data generate (data), SINH(data);

Output :
(5.0,74.20321057778875)
(16.0,4443055.26025388)
(9.0,4051.54190208279)
(2.5,6.0502044810397875)
(5.9,182.51738161672935)
(3.1,11.076449978895173)

tan_data = foreach math_data generate (data), TAN(data);

Output :
(5.0,-3.380515006246586)
(16.0,0.3006322420239034)
(9.0,-0.45231565944180985)
(2.5,-0.7470222972386603)
(5.9,-0.4031107890087444)
(3.1,-0.041616750118239246)

tanh_data = foreach math_data generate (data), TANH(data);

Output :
(5.0,0.9999092042625951)
(16.0,0.9999999999999747)
(9.0,0.999999969540041)
(2.5,0.9866142981514303)
(5.9,0.9999849909996685)
(3.1,0.9959493584508665)

ceil_data = foreach math_data generate (data), CEIL(data);

Output :
(5.0,5.0)
(16.0,16.0)
(9.0,9.0)
(2.5,3.0)
(5.9,6.0)
(3.1,4.0)

floor_data = foreach math_data generate (data), FLOOR(data);

Output :
(5.0,5.0)
(16.0,16.0)
(9.0,9.0)
(2.5,2.0)
(5.9,5.0)
(3.1,3.0)

round_data = foreach math_data generate (data), ROUND(data);

Output :
(5.0,5)
(16.0,16)
(9.0,9)
(2.5,3)
(5.9,6)
(3.1,3)

Logarithmic Functions :~

log_data = foreach math_data generate (data),LOG(data);

Output :

(5.0,1.6094379124341003)
(16.0,2.772588722239781)
(9.0,2.1972245773362196)
(2.5,0.9162907318741551)
(5.9,1.774952367075645)
(3.1,1.1314020807274126)

log_data1 = foreach math_data generate (data),LOG10(data);

Output :

(5.0,0.6989700043360189)
(16.0,1.2041199826559248)
(9.0,0.9542425094393249)
(2.5,0.3979400086720376)
(5.9,0.7708520186620678)
(3.1,0.4913616804737727)