Hands on sessions:

1. Loading data from local file system/hdfs to PIG and vice versa.
2. Diagnostic operator – Dump, Describe, Explain, illustrate
3. Filtering – filter operator; For Each Generate operator: projection, nested projection; Distinct Operator
4. Arithmetic operators, Comparison operator, Boolean Operators

Program – 1: Loading data from local file system/hdfs to PIG and vice versa.

Local ←--→ PIG

Open the terminal

To check the present working directory
[cloudera@localhost ~]$ pwd
/home/cloudera

To create a data file in local system
[cloudera@localhost ~]$ gedit datafile
1,a
2,b
3,c
4,d

Save and close

To verify the file
[cloudera@localhost ~]$ ls
[cloudera@localhost ~]$ cat datafile
1,a
2,b
3,c
4,d

To switch to local execution mode in PIG
[cloudera@localhost ~]$ pig –x local
grunt>

To clear the screen
grunt> clear

To load the data file into PIG
grunt> bag1 = load '/home/cloudera/datafile' using PigStorage(',') as
(id:int,name:chararray);

To see the content of the bag/relation
grunt> dump bag1;

To create alias
grunt> bag2 = bag1;

To see the content of the alias

```
grunt> dump bag2;
```

To store the result of bag into the local system

```
grunt> store  bag1  into  '/home/cloudera/output1'  using  PigStorage(',');

[main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
```

To store the result of bag onto the Desktop

```
grunt> store bag1 into '/home/cloudera/Desktop/output1' using PigStorage(',');
```

To verify the output file in local system

```
grunt> quit
[cloudera@localhost ~]$ ls
[cloudera@localhost ~]$ ls output1
part-m-00000  _SUCCESS
[cloudera@localhost ~]$ cat output1/part*
1,a
2,b
3,c
4,d
```

Also check the desktop for the output file.

HDFS ←--→ PIG

Loading datafile from HDFS to PIG

```
[cloudera@localhost ~]$ gedit datafile2
ramu,bangalore,20
```

raju,chennai,21
john,bangalore,21
sita,hyderabad,22
krish,delhi,20
anu,chennai,19 save and close

To put this file into hadoop
[cloudera@localhost ~]$ hadoop fs -put datafile2 /user/cloudera/

To verify the file in hadoop
[cloudera@localhost ~]$ hadoop fs -ls /user/cloudera/
[cloudera@localhost ~]$ hadoop fs -cat /user/cloudera/datafile2
ramu,bangalore,20
raju,chennai,21
john,bangalore,21
sita,hyderabad,22
krish,delhi,20
anu,chennai,19

To switch to mapreduce mode in PIG
[cloudera@localhost ~]$ pig -x mapreduce
Or
[cloudera@localhost ~]$ pig
grunt > clear

To load the file from hdfs to PIG

```
grunt> student = load '/user/cloudera/datafile2' using PigStorage(',') as
                                        (name:chararray, city:chararray,age:int);
```

To see the content

```
grunt> dump student;
(ramu,bangalore,20)
(raju,chennai,21)
(john,bangalore,21)
(sita,hyderabad,22)
(krish,delhi,20)
(anu,chennai,19)
```

To create alias and see the content

```
grunt> a = student;
grunt> dump a;
```

To store the result into hdfs

```
grunt> store student into '/user/cloudera/output1' using PigStorage(',');

[main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
```

To check the output file in hadoop

```
grunt> fs -ls /user/cloudera/
grunt> fs -ls /user/cloudera/output1
grunt> fs -cat /user/cloudera/output1/part*
ramu,bangalore,20
raju,chennai,21
john,bangalore,21
sita,hyderabad,22
krish,delhi,20
```

anu,chennai,19

We also check the result through the browser
HDFSNamenode-→Browse the file system→ user→ cloudera→Output1→part-m-00000

Program2: Diagnostic operator – Dump, Describe, Explain, illustrate

The **describe** operator is used to view the schema of a relation.
Syntax:

```
grunt> Describe Relation_name
```

grunt> describe student;
student: {name: chararray,city: chararray,age: int}

The **explain** operator is used to display the logical, physical, and MapReduce execution plans of a relation.
Syntax:

```
grunt> explain Relation_name;
```

grunt> explain student;
#New Logical Plan:
#Physical Plan:
Map Reduce Plan

The **illustrate** operator gives you the step-by-step execution of a sequence of statements.

Syntax:

```
grunt> illustrate Relation_name;
```

grunt> illustrate student;
grunt> illustrate student;
grunt> illustrate student;

Program3: Filtering – filter operator; For Each Generate operator: projection, nested projection; Distinct Operator

The **FILTER** operator is used to select the required tuples from a relation based on a condition.
Syntax:

```
grunt> Relation2_name = FILTER Relation1_name BY (condition);
```

grunt> A = filter student by city == 'bangalore';
grunt> dump A;

(ramu,bangalore,20)
(john,bangalore,21)

The **FOREACH** operator is used to generate specified data.
Syntax:

```
grunt> Relation_name2 = FOREACH Relatin_name1 GENERATE (required data);
```

grunt> c = foreach student generate *;
grunt> dump c;

(ramu,bangalore,20)
(raju,chennai,21)
(john,bangalore,21)
(sita,hyderabad,22)
(krish,delhi,20)
(anu,chennai,19)

grunt> c = foreach student generate age;
grunt> dump c;
(20)
(21)
(21)
(22)
(20)
(19)

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

```
grunt> Relation_name2 = DISTINCT Relatin_name1;
```

grunt> a = distinct c;
grunt> dump a;
(19)
(20)
(21)
(22)

The **GROUP** operator is used to group the data a relation. It collects the data having the same key.

Syntax:

```
grunt> Group_data = GROUP Relation_name BY field;
```

grunt> g = group student by city;
grunt> dump g;

(delhi,{(krish,delhi,20)})
(chennai,{(raju,chennai,21),(anu,chennai,19)})
(bangalore,{(ramu,bangalore,20),(john,bangalore,21)})
(hyderabad,{(sita,hyderabad,22)})

grunt> describe g;
g: {group: chararray,student: {(name: chararray,city: chararray,age: int)}}

Program 4: Arithmetic operators, Comparison operator, Boolean Operators

grunt> a = foreach student generate age,age+10;
grunt> dump a;
(20,30)
(21,31)
(21,31)
(22,32)
(20,30)
(19,29)

grunt> A = filter student by age>20;
grunt> dump A;

(raju,chennai,21)
(john,bangalore,21)
(sita,hyderabad,22)

grunt> A = filter student by age>19 and age<22;
grunt> dump A;

(similarly, we can use or,not)

(ramu,bangalore,20)
(raju,chennai,21)
(john,bangalore,21)
(krish,delhi,20)

Problems on module – 2:

Create the following data file

[cloudera@localhost ~]$ gedit doctor

Milan,1001,5,apollo,500
Jay,1002,10,apollo,500
lalit,1003,20,manipal,500
Mohit,1004,15,columbia,600
Chauhan,1005,30,narayana,550
Suraj,1006,25,manipal,650

[cloudera@localhost ~]$ pig -x local

```
grunt> clear


grunt> doc = load '/home/cloudera/doctor' using PigStorage(',') as

        (name:chararray, id:int, exp:int, hosp:chararray, fees:int);

grunt> dump doc;
```

Q) Display the name of the doctor and hospital.

grunt> a = foreach doc generate name,hosp;
grunt> dump a;
(Milan,apollo)
(Jay,apollo)
(lalit,manipal)
(Mohit, columbia)
(Chauhan,narayana)
(Suraj,manipal)

Q) Display the details of doctors who have more than 10 years of experience

grunt> a = filter doc by exp>10;
grunt>dump a;
(lalit,1003,20,manipal,500)
(Mohit,1004,15, columbia,600)
(Chauhan,1005,30,narayana,550)
(Suraj,1006,25,manipal,650)

Q)Display the details of doctor who works in apollo hospital.

grunt> a = filter doc by hosp=='apollo';

```
grunt> dump a;
(Milan,1001,5,apollo,500)
(Jay,1002,10,apollo,500)
```

Q)Display doctor name, docid who works in manipal hospital.

```
grunt> a = filter doc by hosp=='manipal';
grunt> ans = foreach a generate name,id;
grunt> dump ans;
(lalit,1003)
(Suraj,1006)
```

Q) List different fees taken by various doctors.

```
grunt> a = foreach doc generate fees;
grunt> ans = distinct a;
grunt> dump ans;
(500)
(550)
(650)
 (600)
```

Q)Display the details of doctor whose experience is more than 20 yrs but not working in manipal.

```
grunt> a = filter doc by exp>20  and   hosp!='manipal';
grunt> dump a;
(Chauhan,1005,30,narayana,550)
```

(OR)

```
grunt> a = filter doc by exp>20 and not(hosp=='manipal');
grunt> dump a;
(Chauhan,1005,30,narayana,550)
```

Q)Display docname, fees, fees%10

grunt> a = foreach doc generate name, fees,fees%10;
grunt> dump a;
(Milan,500,0)
(Jay,500,0)
(lalit,500,0)
(Mohit,600,0)
(Chauhan,550,50)
(Suraj,650,50)

Q)Display docname,hospname,fees,fees after 10 percentage hike.

grunt> a = foreach doc generate name,hosp, fees,fees*1.1;
grunt> dump a;
(Milan,apollo,500,550.0)
(Jay,apollo,500,550.0)
(lalit,manipal,500,550.0)
(Mohit, columbia ,600,660)
(Chauhan,narayana,550,605.0)
(Suraj,manipal,650,715.0000000000001)

Q) Display true if experience of doctor is above 10; else display false.

grunt> a = foreach doc generate name,exp,(exp>10?'true':'false');
grunt> dump a;
(Milan,5,false)
(Jay,10,false)
(lalit,20,true)
(Mohit,15,true)
(Chauhan,30,true)
(Suraj,25,true)

Q)Display only hospital name, fees by taking into consideration that some concession is going on for all branches of apollo hospital by rs 200. So display new fees for apollo hospitals and same current fees for other hospitals.

```
grunt> a = foreach doc generate hosp,(hosp=='apollo'?fees-200:fees);
grunt> dump a;
(apollo,300)
(apollo,300)
 (manipal,500)
(columbia,600)
(narayana,550)
(manipal,650)
```

Q)Display the details of doctor who is either taking fees above rs 500 or whose experience is above 20 years.

```
grunt> a = filter doc by  fees>500   or   exp>20;                    (OR)
grunt> a = filter doc by  (fees>500)   or  (exp>20);                 (OR)
grunt> a = filter doc by  $4>500   or   $2>20;
grunt> dump a;
(Mohit,1004,15,columbia,600)
(Chauhan,1005,30,narayana,550)
(Suraj,1006,25,manipal,650)
```

Q) Display the details of doctor whose name ends with 'an'

```
grunt> a = filter doc by name matches '.*an';
grunt> dump a;
(Milan,1001,5,apollo,500)
```

(Chauhan,1005,30,narayana,550)

Q) Display the details of doctor whose name begins with ‘S’

grunt> a = filter doc by name matches 'S.*';

(Suraj,1006,25,manipal,650)

Q) Display the details of doctor whose name has ‘a’ as substring

grunt> a = filter doc by name matches '.*a.*';

(Milan,1001,5,apollo,500)
(Jay,1002,10,apollo,500)
(lalit,1003,20,manipal,500)
(Chauhan,1005,30,narayana,550)
(Suraj,1006,25,manipal,650)

Q) Display the details of doctor whose name has substring ‘it’

grunt> a = filter doc by name matches '.*it.*';

(lalit,1003,20,manipal,500)
(Mohit,1004,15,columbia,600)

Q) Display the details of doctor whose hospital name has substring ‘a’

grunt> a = filter doc by hosp matches '.*a.*';

(Milan,1001,5,apollo,500)
(Jay,1002,10,apollo,500)
(lalit,1003,20,manipal,500)
(Mohit,1004,15,columbia,600)
(Chauhan,1005,30,narayana,550)
(Suraj,1006,25,manipal,650)

Q)Perform grouping of tuples wrt hospital name.

grunt> gr = group doc by hosp;
grunt> dump gr;

(apollo,{(Milan,1001,5,apollo,500),(Jay,1002,10,apollo,500)})
(manipal,{(lalit,1003,20,manipal,500),(Suraj,1006,25,manipal,650)})
(columbia,{(Mohit,1004,15,columbia,600)})
(narayana,{(Chauhan,1005,30,narayana,550)})

Q)Display hospital name, doc names on screen for each hospital.

grunt> gr = group doc by hosp;
grunt> a = foreach gr generate group,doc.name;
grunt> dump a;
(apollo,{(Milan),(Jay)})
(manipal,{(lalit),(Suraj)})
(columbia,{(Mohit)})
(narayana,{(Chauhan)})