ASSIGNMENT2 - PIG COMMANDS
(In MAP REDUCE MODE USING GRUNT SHELL)

<<<<<<<----------- PIG COMMANDS ------------>>>>>>>

Following commands make use of specified data files:

Commands	DataFiles
concat,tokenize,sum,min,max, limit, store, flatten	students_data.txt
distinct	duplicate_data.txt
IsEmpty	emp1.txt and emp2.txt

********** Loading students_data.txt in pig **********

LOAD Command: It loads data from hdfs to pig

Below command loads the "students_data.txt" from hdfs to pig, since file contains ',' as delimiter therefore, "PigStorage" load function is used to specify delimiter, default delimiter is '\t' i.e. tab.

```
grunt> load_std_data = LOAD
'/user/my_pig_stuff/students_data.txt' USING PigStorage(',')
AS
(rollno:int,firstname:chararray,lastname:chararray,marks:int,a
ge:int);
2017-08-12 17:18:36,547 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
mapreduce.job.counters.limit is deprecated. Instead, use
mapreduce.job.counters.max
2017-08-12 17:18:36,552 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-
per-checksum
2017-08-12 17:18:36,552 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
```

DESCRIBE command: It shows schema of relation/alias

Below command describes schema of load_std_data

```
grunt> DESCRIBE load_std_data;
```

```
load_std_data: {rollno: int,firstname: chararray,lastname:
chararray,marks: int,age: int}
```

Dump Command: It displays the result of schema/alias

Below command displays result of load_std_data

```
grunt> DUMP load_std_data;
2017-08-12 17:10:44,760 [main] INFO
org.apache.pig.tools.pigstats.ScriptState - Pig features used
in the script: UNKNOWN
2017-08-12 17:10:45,034 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation - .......
org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.J
obControlCompiler - This job cannot be converted run in-
process ..... (not all processing of DUMP
command is shown here)
(1,Amit,Sharma,86,12)
(2,Sumit,Kumar,70,13)
(3,Ramesh,Verma,65,11)
(4,Suraj,Gupta,69,12)
(5,Ravi,Verma,80,13)
(6,Manav,Gupta,79,11)
(7,Rajesh,Sharma,50,12)
(8,Sameer,Kumar,78,13)
(9,Rajan,Gupta,90,12)
(10,Aman,Sharma,87,11)
```

Output produced by above dump command, shows data is loaded correctly in load_std_data relation/alias

********** CONCAT FUNCTION **********

Purpose: It is used to concatenate two or more expressions/fields of the same type

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

Example: Concatenate two fields i.e. firstname and lastname of load_std_data relation

Below command concatenates firstname and lastname separated by space ' ' and result is stored in concatfirstlastname

```
grunt> concatfirstlastname = FOREACH load_std_data GENERATE
CONCAT(firstname,' ',lastname) as name;
```

Using Describe, schema of concatfirstlastname is shown, which shows, only single field is created from two fields

```
grunt> DESCRIBE concatfirstlastname;
concatfirstlastname: {name: chararray}
```

Using Dump, data stored in concatfirstlastname is displayed

```
grunt> DUMP concatfirstlastname;
2017-08-12 15:44:34,142 [main] INFO
org.apache.pig.tools.pigstats.ScriptState - Pig features used
```

```
in the script: UNKNOWN
io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-
per-checksum .....................................
org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
- Total input paths to process : 1
(Amit Sharma)
(Sumit Kumar)
(Ramesh Verma)
(Suraj Gupta)
(Ravi Verma)
(Manav Gupta)
(Rajesh Sharma)
(Sameer Kumar)
(Rajan Gupta)
(Aman Sharma)
```

Output displayed by above dump command shows that firstname and lastname are concatenated with separator space ' '

********** TOKENIZE FUNCTION **********

Purpose: The TOKENIZE() function 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.

Syntax:

```
grunt> TOKENIZE(expression/field [, 'field_delimiter'])
```

As a delimeter to the TOKENIZE() function, we can pass space [ ], double quote [" "], coma [,], parenthesis [()], star [*].

Example: In concatfirstlastname relation, we have a single field which contains two words separated by space, so to split those fields, tokenize() function is applied.

Below command applies tokenize() function on every row of concatfirstlastname relation on its name field using foreach command

```
grunt> tokenizeName = FOREACH concatfirstlastname GENERATE
TOKENIZE(name);
```

Using describe, schema of tokenizeName relation is displayed which shows bag of tuple is created

```
grunt> DESCRIBE tokenizeName;
tokenizeName: {bag_of_tokenTuples_from_name: {tuple_of_tokens:
(token: chararray)}}
```

Since, alias is not provided with tokenize(name) so bag_of_tokenTuples_from_name appears inside bag, in next command alias is provided for this.

Below command provides alias to tokenize(name)

```
grunt> tokenizeName = FOREACH concatfirstlastname GENERATE
TOKENIZE(name) AS tokenName;
```

Using describe, we can see alias has been provided to bag_of_tokenTuples_from_name

```
grunt> DESCRIBE tokenizeName;
tokenizeName: {tokenName: {tuple_of_tokens: (token:
chararray)}}
```

Using dump, we can see how tokenize() function has done processing on name field of concatfirstlastname relation

```
grunt> DUMP tokenizeName;
2017-08-12 15:48:05,406 [main] INFO
org.apache.pig.tools.pigstats.ScriptState - Pig features used
in the script: UNKNOWN
2017-08-12 15:48:05,573 [main] INFO
2017-08-12 15:48:05,576 [main] INFO  .....................

org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
- Total input paths to process : 1

({(Amit),(Sharma)})
({(Sumit),(Kumar)})
({(Ramesh),(Verma)})
({(Suraj),(Gupta)})
({(Ravi),(Verma)})
({(Manav),(Gupta)})
({(Rajesh),(Sharma)})
({(Sameer),(Kumar)})
({(Rajan),(Gupta)})
({(Aman),(Sharma)})
```

In above output, we can see that, Outer tuple() contains bag{}, which contains two tuples (),()

********** SUM FUNCTION **********

Purpose: To get the total of the numeric values of a column in a single-column bag SUM() function is used . While computing the total, the SUM() function ignores the NULL values.

NOTE: To get the sum value of a group, we need to group it using the Group By operator and proceed with the sum function.

Syntax: `grunt> SUM(expression/field)`

Example: To find sum of marks per age group of students

In order to process above query, two steps are followed:

Step 1: GROUP BY COMMAND IS USED TO get details of students per age

```
grunt> groupByAge = GROUP load_std_data BY age;
```

Below command shows schema of groupByAge relation

```
grunt> DESCRIBE groupByAge;
groupByAge: {group: int,load_std_data: {(rollno:
int,firstname: chararray,lastname: chararray,marks: int,age:
int)}}
```

groupByAge relation contains two fields, first field is "group" i.e. age and second field is "load_std_data relation" itself, on which grouping is done

Below commands shows data inside groupByAge relation

```
grunt> DUMP groupByAge;
2017-08-12 15:50:32,855 [main] INFO
org.apache.pig.tools.pigstats.ScriptState - Pig features used
in the script: GROUP_BY
2017-08-12 15:50:33,015 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2017-08-12 15:50:33,015 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
mapreduce.job.counters.limit is deprecated. Instead, use
mapreduce.job.counters.max
2017-08-12 15:50:33,018 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-
per-checksum.........................
org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
- Total input paths to process : 1

(11,{(10,Aman,Sharma,87,11),(6,Manav,Gupta,79,11),(3,Ramesh,Ve
rma,65,11)})
(12,{(9,Rajan,Gupta,90,12),(7,Rajesh,Sharma,50,12),(4,Suraj,Gu
pta,69,12),(1,Amit,Sharma,86,12)})
(13,{(8,Sameer,Kumar,78,13),(5,Ravi,Verma,80,13),(2,Sumit,Kuma
r,70,13)})
```

Output of above dump command shows that, per age, a bag is associated which contains rows related to that age only

Step 2: In this step SUM function is applied on marks field which is present in load_std_data relation and which is further a bag type field in groupByAge relation

Below statement calculates sum of marks per age group of the students, and result is stored in sumOfMarksGPByAge relation

```
grunt> sumOfMarksGPByAge = FOREACH groupByAge GENERATE
```

```
group,SUM(load_std_data.marks) as sumOfMarks;
```

Below command shows the schema of sumOfMarksGPBYAge relation, which shows two fields group and sumOfMarks

```
grunt> DESCRIBE sumOfMarksGPByAge;
sumOfMarksGPByAge: {group: int,sumOfMarks: long}
```

Below command displays data inside sumOfMarksGPBYAge

```
grunt> DUMP sumOfMarksGPByAge;
2017-08-12 15:54:16,983 [main] INFO
org.apache.pig.tools.pigstats.ScriptState - Pig features used
in the script: GROUP_BY
2017-08-12 15:54:17,050 [main]
INFO ..................................................
org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
- Total input paths to process : 1
(11,231)
(12,295)
(13,228)
```

IN above output, we can see that first column is age and second column is sum of marks per age

********** MIN FUNCTION **********

Purpose: 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 minimum value of a group, we need to group it using the Group By operator and proceed with the minimum function.

Syntax: `grunt> MIN(expression/field)`

Example: To find minimum marks for each group in groupByAge relation

Below command finds minimum marks in each group using foreach and result is stored in minOfMarksGPByAge relation

```
grunt> minOfMarksGPByAge = FOREACH groupByAge GENERATE
group,MIN(load_std_data.marks) as minOfMarks;
```

Describe command shows two fields have been generated in

relation minOfMarksGPByAge, first is group i.e. age and second is minOfMarks

```
grunt> DESCRIBE minOfMarksGPByAge;
minOfMarksGPByAge: {group: int,minOfMarks: int}
```

Dump command shows the data in minOfMarksGPByAge relation

```
grunt> DUMP minOfMarksGPByAge;
2017-08-12 15:57:56,390 [main] INFO
org.apache.pig.tools.pigstats.ScriptState - Pig features used
in the script: GROUP_BY
2017-08-12 15:57:56,555 [main] INFO

org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
- Total input paths to process : 1
(11,65)
(12,50)
(13,70)
```

********** MAX FUNCTION **********

Purpose: 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 maximum value of a group, we need to group it using the Group By operator and proceed with the maximum function.

Syntax: grunt> Max(expression)

Example: To find maximum marks for each group in groupByAge relation

Below command finds maximum marks for each group in groupByAge relation using Foreach command and result is stored in maxOfMarksGPByAge relation

```
grunt> maxOfMarksGPByAge = FOREACH groupByAge GENERATE
group,MAX(load_std_data.marks) as maxOfMarks;
```

Below command shows the schema of maxOfMarksGPByAge relation, where first is group i.e. age and second field is maxOfMarks

```
grunt> DESCRIBE maxOfMarksGPByAge;
maxOfMarksGPByAge: {group: int,maxOfMarks: int}
```

Below commands shows data inside maxOfMarksGPByAge relation

```
grunt> DUMP maxOfMarksGPByAge;
2017-08-12 16:00:05,593 [main] INFO
org.apache.pig.tools.pigstats.ScriptState - Pig features used
in the script: GROUP_BY
2017-08-12          16:00:05,818          [main]          INFO
org.apache.hadoop.conf.Configuration.deprecation -

org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
- Total input paths to process : 1
(11,87)
(12,90)
(13,80)
```

********** LIMIT COMMAND **********

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

Syntax: grunt> Result = LIMIT relation_name no_oftuples;

Example: To work with limited tuples of load_std_data relation

Below command limits tuples of relation load_std_data to 5, however original number of tuples inside it are 10, and final result is stored in relation limitDisplayOfTuples

```
grunt> limitDisplayOfTuples = LIMIT load_std_data 5;
```

Below command shows that limitDisplayOfTuples relation has same schema as that of load_std_data relation

```
grunt> DESCRIBE limitDisplayOfTuples;
limitDisplayOfTuples: {rollno: int,firstname:
chararray,lastname: chararray,marks: int,age: int}
```

Below command shows that limitDisplayOfTuples has only 5 tuples

```
grunt> DUMP limitDisplayOfTuples;
2017-08-12 16:04:56,324 [main] INFO
org.apache.pig.tools.pigstats.ScriptState - Pig features used
in the script: LIMIT
2017-08-12          16:04:56,496          [main]          INFO
org.apache.hadoop.conf.Configuration.deprecation -
org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
- Total input paths to process : 1
(1,Amit,Sharma,86,12)
(2,Sumit,Kumar,70,13)
(3,Ramesh,Verma,65,11)
(4,Suraj,Gupta,69,12)
(5,Ravi,Verma,80,13)
```

********** STORE COMMAND **********

Purpose: Store operator is used to load the data from pig to the file system

Syntax: grunt> STORE relation_name INTO 'required_directory_path' [USING loadFunction];

Example: To load the output of relation concatfirstlastname in hdfs

Below command stores output of concatfirstlastname at location /user/my_pig_stuff/concat_result in hdfs

```
grunt> storeConcatName = STORE concatfirstlastname INTO
'/user/my_pig_stuff/concat_result';
2017-08-12 16:08:08,533 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
mapreduce.job.counters.limit is deprecated. Instead, use
mapreduce.job.counters.max
2017-08-12          16:08:08,936          [main]          INFO
org.apache.....................................................
2017-08-12          16:09:14,235          [main]          INFO
org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.M
apReduceLauncher - Success!
grunt>
```

Check the data inside hdfs -------->>>>>>>>

```
[acadgild@localhost ~]$ hadoop fs -ls /user/my_pig_stuff
17/08/12 16:09:23 WARN util.NativeCodeLoader: Unable to load
native-hadoop library for your platform... using builtin-java
classes where applicable
Found 5 items
drwxr-xr-x   - acadgild supergroup          0 2017-08-12 16:09
/user/my_pig_stuff/concat_result
-rw-r--r--   1 acadgild supergroup         52 2017-08-12 02:40
/user/my_pig_stuff/duplicate_data.txt
-rw-r--r--   1 acadgild supergroup         94 2017-08-12 03:18
/user/my_pig_stuff/emp1.txt
-rw-r--r--   1 acadgild supergroup         98 2017-08-12 03:19
/user/my_pig_stuff/emp2.txt
-rw-r--r--   1 acadgild supergroup        204 2017-08-12 01:48
/user/my_pig_stuff/students_data.txt

[acadgild@localhost ~]$ hadoop fs -ls
/user/my_pig_stuff/concat_result
17/08/12 16:10:15 WARN util.NativeCodeLoader: Unable to load
native-hadoop library for your platform... using builtin-java
classes where applicable
Found 2 items
-rw-r--r--   1 acadgild supergroup          0 2017-08-12 16:09
/user/my_pig_stuff/concat_result/_SUCCESS
-rw-r--r--   1 acadgild supergroup        123 2017-08-12 16:09
/user/my_pig_stuff/concat_result/part-m-00000
```

```
[acadgild@localhost ~]$ hadoop fs -cat
/user/my_pig_stuff/concat_result/part-m-00000
17/08/12 16:10:48 WARN util.NativeCodeLoader: Unable to load
native-hadoop library for your platform... using builtin-java
classes where applicable
Amit Sharma
Sumit Kumar
Ramesh Verma
Suraj Gupta
Ravi Verma
Manav Gupta
Rajesh Sharma
Sameer Kumar
Rajan Gupta
Aman Sharma
```

******* Loading "duplicate_data.txt" file from hdfs to pig*******

```
grunt> load_duplicate_data = LOAD
'/user/my_pig_stuff/duplicate_data.txt' USING PigStorage(',')
AS (rollno:int,name:chararray,marks:int);
2017-08-12 16:13:22,270 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2017-08-12 16:13:22,271 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
mapreduce.job.counters.limit is deprecated. Instead, use
mapreduce.job.counters.max
2017-08-12 16:13:22,277 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-
per-checksum.............................
org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
```

Below commands shows the schema of load_duplicate_data,where three fields, i.e. rollno, name, marks are shown

```
grunt> DESCRIBE load_duplicate_data;
load_duplicate_data: {rollno: int,name: chararray,marks: int}
```

Below commands shows data is correctly loaded from duplicate_data.txt to load_duplicate_data relation

```
grunt> DUMP load_duplicate_data;
2017-08-12 16:14:02,955 [main] INFO
org.apache.pig.tools.pigstats.ScriptState - Pig features used
in the script: UNKNOWN
2017-08-12 16:14:03,133 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2017-08-12 16:14:03,133 [main] INFO  ................
org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
- Total input paths to process : 1
(1,Aman,50)
```

```
(2,Suraj,60)
(3,Ravi,70)
(1,Aman,50)
(2,Suraj,60)
```

Now Below command can work on this duplicate data

********** DISTINCT COMMAND **********

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

Syntax: grunt> relation_name2 = DISTINCT relation_name1;

Example: To work with distinct data inside load_duplicate_data relation

Below command places distinct records in distinctDupData relation from load_duplicate_data relation

```
grunt> distinctDupData = DISTINCT load_duplicate_data;
```

Below command shows that schema does not change

```
grunt> DESCRIBE distinctDupData;
distinctDupData: {rollno: int,name: chararray,marks: int}
```

Below command displays the data inside distinctDupData

```
grunt> DUMP distinctDupData;
2017-08-12 16:17:55,142 [main] INFO
org.apache.pig.tools.pigstats.ScriptState - Pig features used
in the script: DISTINCT
2017-08-12 16:17:55,235 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation - .........
org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
- Total input paths to process : 1
(1,Aman,50)
(2,Suraj,60)
(3,Ravi,70)
```

********** FLATTEN FUNCTION **********

Purpose: The FLATTEN() 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/flattens tuples as well as bags.

Syntax: grunt> flatten(nested_expression)

Example: To flatten the tokenizeName data which appears as below:

```
({(Amit),(Sharma)})
({(Sumit),(Kumar)})
({(Ramesh),(Verma)})
({(Suraj),(Gupta)})
```

```
({(Ravi),(Verma)})
({(Manav),(Gupta)})
({(Rajesh),(Sharma)})
({(Sameer),(Kumar)})
({(Rajan),(Gupta)})
({(Aman),(Sharma)})
```

Below command flattens the tokenName field of tokenizeName relation using FLATTEN function which works on each row using foreach and result is stored in flattendTokenizeName relation

```
grunt> flattenTokenizeName = FOREACH tokenizeName GENERATE
FLATTEN(tokenName) as flattenedTokenizeName;
```

Below command shows the schema of flattenTokenizeName relation having only one field flattenedTokenizeName which is a bag

```
grunt> DESCRIBE flattenTokenizeName;
flattenTokenizeName: {flattenedTokenizeName: chararray}
```

Below command shows how data is flattened, i.e. nested fields are taken and placed on separated line

```
grunt> DUMP flattenTokenizeName;
2017-08-12 16:21:30,751 [main] INFO
org.apache.pig.tools.pigstats.ScriptState - Pig features used
in the script: UNKNOWN
2017-08-12 16:21:30,954 [main] INFO  ..............

org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
- Total input paths to process : 1
(Amit)
(Sharma)
(Sumit)
(Kumar)
(Ramesh)
(Verma)
(Suraj)
(Gupta)
(Ravi)
(Verma)
(Manav)
(Gupta)
(Rajesh)
(Sharma)
(Sameer)
(Kumar)
(Rajan)
(Gupta)
(Aman)
(Sharma)
```

******** Loading emp1.txt from hdfs to pig ********

```
grunt> load_emp1_data = LOAD '/user/my_pig_stuff/emp1.txt'
USING PigStorage(',') AS
(id:int,name:chararray,salary:int,dept:chararray,age:int);
2017-08-12 16:24:33,416 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
mapreduce.job.counters.limit is deprecated. Instead, use
mapreduce.job.counters.max
2017-08-12 16:24:33,416 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-
per-checksum
2017-08-12 16:24:33,417 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
```

From below schema,we can see that, load_emp1_data relation contains 5 fields

```
grunt> DESCRIBE load_emp1_data;
load_emp1_data: {id: int,name: chararray,salary: int,dept:
chararray,age: int}
```

Below command shows data is loaded correctly from emp1.txt file to load_emp1_data relation

```
grunt> DUMP load_emp1_data;
2017-08-12 16:24:55,066 [main] INFO
org.apache.pig.tools.pigstats.ScriptState - Pig features used
in the script: UNKNOWN
2017-08-12 16:24:55,554 [main] INFO  .................
org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
- Total input paths to process : 1
(101,Amit,2000,sales,30)
(102,Suraj,3000,sales,35)
(103,Raman,4000,sales,30)
(104,Ravi,3000,sales,30)
```

****** Loading emp2.txt from hdfs to pig ******

```
grunt> load_emp2_data = LOAD '/user/my_pig_stuff/emp2.txt'
USING PigStorage(',') AS
(id:int,name:chararray,salary:int,age:int,dept:chararray);
2017-08-12 16:26:53,616 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
mapreduce.job.counters.limit is deprecated. Instead, use
mapreduce.job.counters.max
2017-08-12 16:26:53,617 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-
per-checksum
2017-08-12 16:26:53,617 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
```

From below schema,we can see that, load_emp2_data relation contains 5 fields

```
grunt> DESCRIBE load_emp2_data;
load_emp2_data: {id: int,name: chararray,salary: int,age:
```

```
int,dept: chararray}
```

Below command shows data is loaded correctly from emp2.txt file to load_emp2_data relation

```
grunt> DUMP load_emp2_data;
2017-08-12 16:27:05,678 [main] INFO
org.apache.pig.tools.pigstats.ScriptState - Pig features used
in the script: UNKNOWN
2017-08-12 16:27:05,831 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2017-08-12 16:27:05,831 [main] INFO  ..............
org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
- Total input paths to process : 1
(101,Amit,4000,35,sales)
(102,Rajeev,3500,30,admin)
(103,Ravijot,4500,40,sales)
(104,Ratan,5000,42,admin)
```

********** ISEMPTY FUNCTION **********

Purpose: The IsEmpty() function is used to check if a bag or map is empty.

Syntax: `grunt> IsEmpty(expression)`

Example: From a cogrouped data, find empty bags

Above task in completed in two steps:

Step 1: Using cogroup functin, load_emp1_data and load_emp2 _data are cogrouped, on the basis of common field i.e. age

Below command stores the cogrouped data in cogroupEmp1Emp2 relation

```
grunt> cogroupEmp1Emp2 = COGROUP load_emp1_data BY age,
load_emp2_data BY age;
```

Below command shows, cogroupEmp1Emp2 contains an outer bag and two inner bags, where two inners bags are load_emp1_data and load_emp2_data itself, and these two inner bags are associated to group field which is age field

```
grunt> DESCRIBE cogroupEmp1Emp2;
cogroupEmp1Emp2: {group: int,load_emp1_data: {(id: int,name:
chararray,salary: int,dept: chararray,age: int)},load_emp2
_data: {(id: int,name: chararray,salary: int,age: int,dept:
chararray)}}
```

Below command shows the result in cogrouped format i.e. in outer bag and inner bags format

```
grunt> DUMP cogroupEmp1Emp2;
2017-08-12 16:29:31,031 [main] INFO
org.apache.pig.tools.pigstats.ScriptState - Pig features used
in the script: COGROUP
2017-08-12 16:29:31,171 [main] INFO  ............

org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
- Total input paths to process : 1
(30,{(104,Ravi,3000,sales,30),(103,Raman,4000,sales,30),(101,A
mit,2000,sales,30)},{(102,Rajeev,3500,30,admin)})
(35,{(102,Suraj,3000,sales,35)},{(101,Amit,4000,35,sales)})
(40,{},{(103,Ravijot,4500,40,sales)})
(42,{},{(104,Ratan,5000,42,admin)})
```

Step 2 : Using ISEMPTY function, we can check whether load_emp1_data bag is empty or not

Below command filters only those records from cogroupEmp1Emp2 relation where load_emp1_data bag is empty

```
grunt> isEmptyOnCogroup = FILTER cogroupEmp1Emp2 BY
IsEmpty(load_emp1_data);
```

Below command shows filter operation does not change schema

```
grunt> DESCRIBE isEmptyOnCogroup;
isEmptyOnCogroup: {group: int,load_emp1_data: {(id: int,name:
chararray,salary: int,dept: chararray,age: int)},load_emp2
_data: {(id: int,name: chararray,salary: int,age: int,dept:
chararray)}}
```

Below command shows group field i.e. age, empty load_emp1_data bag and non-empty load_emp2_data bag

```
grunt> DUMP isEmptyOnCogroup;
2017-08-12 16:36:56,798 [main] INFO
org.apache.pig.tools.pigstats.ScriptState - Pig features used
in the script: COGROUP,FILTER
2017-08-12 16:36:56,892 [main] INFO
org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2017-08-12 16:36:56,892 [main] INFO  ............

org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
- Total input paths to process : 1
(40,{},{(103,Ravijot,4500,40,sales)})
(42,{},{(104,Ratan,5000,42,admin)})
```