ASSIGNMENT 5.1

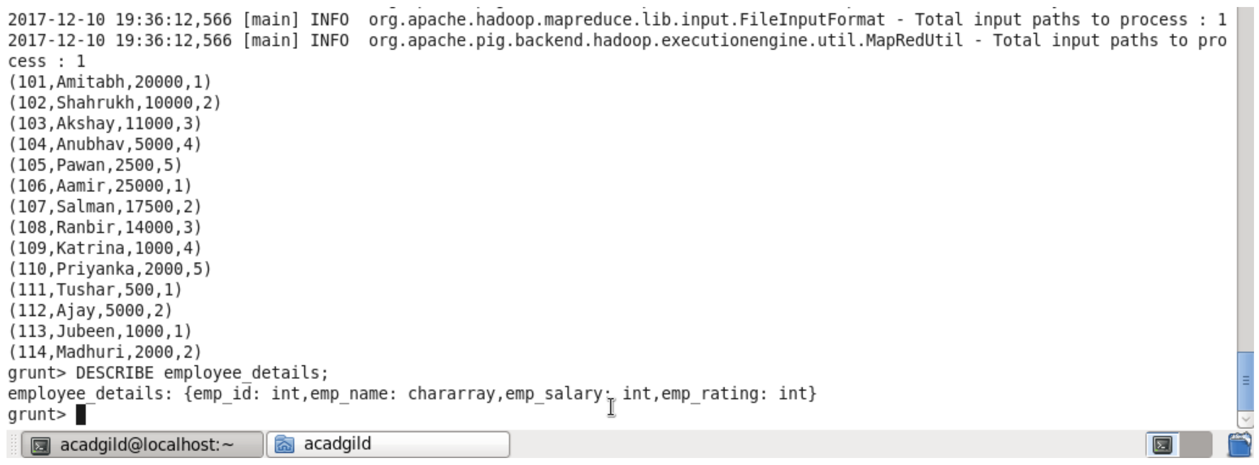
Datasets used:

employee_details.txt

employee_details = LOAD 'employee_details.txt' USING PigStorage(',') AS (emp_id:int, emp_name:chararray, emp_salary:int, emp_rating:int);

Verification:

DUMP 'employee_details';



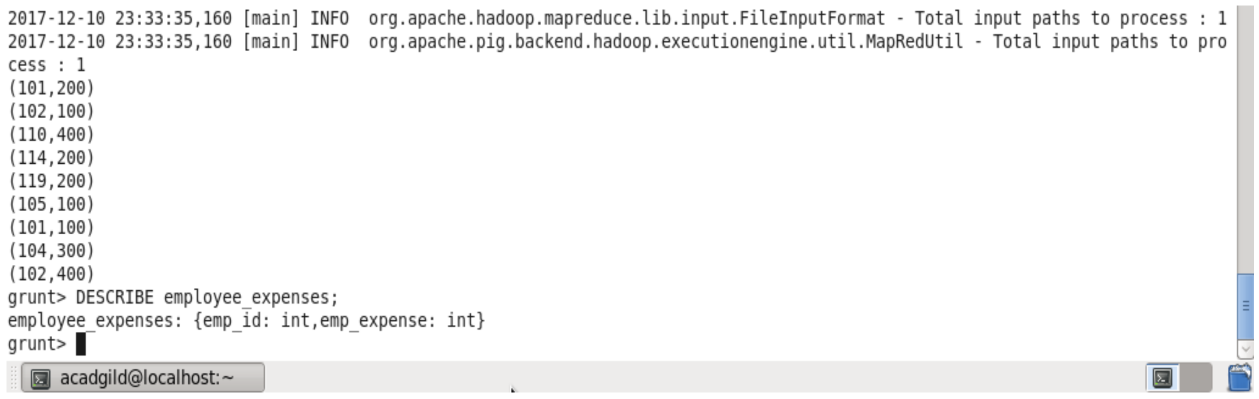
```
2017-12-10 19:36:12,566 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 19:36:12,566 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(101,Amitabh,20000,1)
(102,Shahrukh,10000,2)
(103,Akshay,11000,3)
(104,Anubhav,5000,4)
(105,Pawan,2500,5)
(106,Aamir,25000,1)
(107,Salman,17500,2)
(108,Ranbir,14000,3)
(109,Katrina,1000,4)
(110,Priyanka,2000,5)
(111,Tushar,500,1)
(112,Ajay,5000,2)
(113,Jubeen,1000,1)
(114,Madhuri,2000,2)
grunt> DESCRIBE employee_details;
employee_details: {emp_id: int,emp_name: chararray,emp_salary: int,emp_rating: int}
grunt>
```

employee_expenses.txt

employee_expenses = LOAD 'employee_expenses.txt' AS (emp_id:int, emp_expense:int);

Verification:

DUMP 'employee_ expenses;



```
2017-12-10 23:33:35,160 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 23:33:35,160 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(101,200)
(102,100)
(110,400)
(114,200)
(119,200)
(105,100)
(101,100)
(104,300)
(102,400)
grunt> DESCRIBE employee_expenses;
employee_expenses: {emp_id: int,emp_expense: int}
grunt>
```

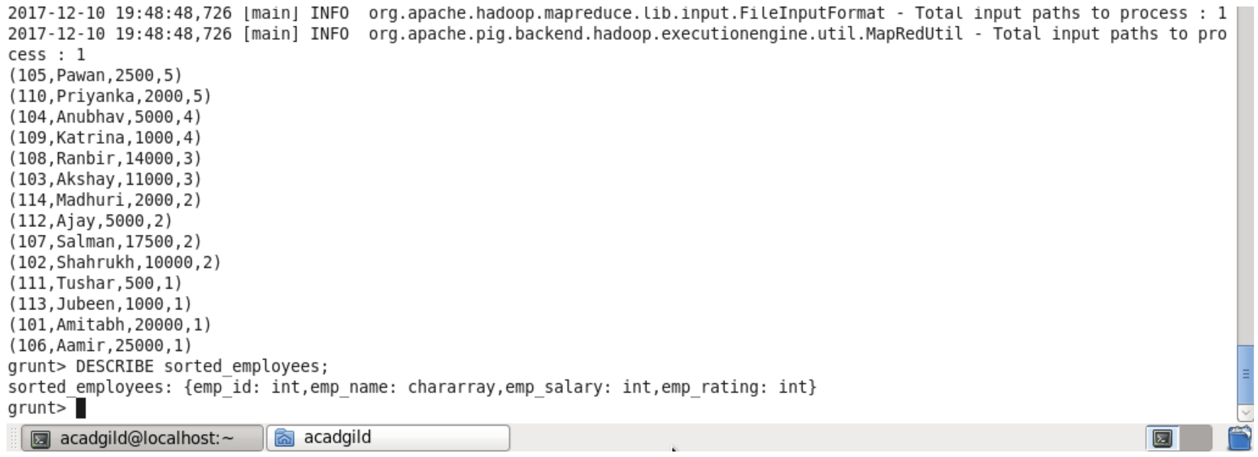
Task 1: Top 5 employees (employee id and employee name) with highest rating. (In case two employees have same rating, employee with name coming first in dictionary should get preference)

Solution:

Step 1: Sort employees by their ratings in descending order.

sorted_employees = ORDER employee_details BY emp_rating DESC;

Output: DUMP sorted_employees;

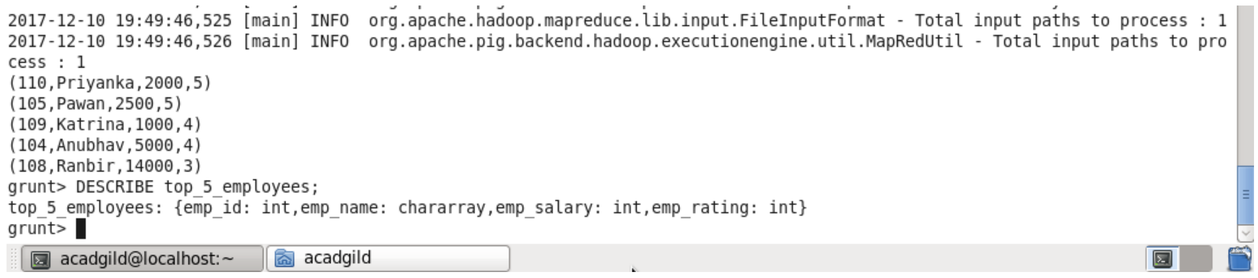


```
2017-12-10 19:48:48,726 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 19:48:48,726 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(105,Pawan,2500,5)
(110,Priyanka,2000,5)
(104,Anubhav,5000,4)
(109,Katrina,1000,4)
(108,Ranbir,14000,3)
(103,Akshay,11000,3)
(114,Madhuri,2000,2)
(112,Ajay,5000,2)
(107,Salman,17500,2)
(102,Shahrukh,10000,2)
(111,Tushar,500,1)
(113,Jubeen,1000,1)
(101,Amitabh,20000,1)
(106,Aamir,25000,1)
grunt> DESCRIBE sorted_employees;
sorted_employees: {emp_id: int,emp_name: chararray,emp_salary: int,emp_rating: int}
grunt>
```

Step 2: Get top 5 employees.

top_5_employees = LIMIT sorted_employees 5;

Output: DUMP top_5_employees;

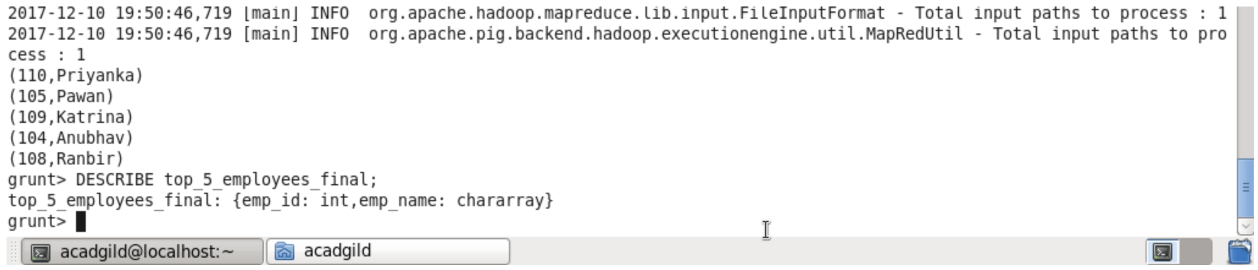


```
2017-12-10 19:49:46,525 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 19:49:46,526 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(110,Priyanka,2000,5)
(105,Pawan,2500,5)
(109,Katrina,1000,4)
(104,Anubhav,5000,4)
(108,Ranbir,14000,3)
grunt> DESCRIBE top_5_employees;
top_5_employees: {emp_id: int,emp_name: chararray,emp_salary: int,emp_rating: int}
grunt>
```

Step 3: Get employee id and employee name from above relation.

top_5_employees_final = FOREACH top_5_employees GENERATE emp_id, emp_name;

Output: DUMP top_5_employees_final;



```
2017-12-10 19:50:46,719 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 19:50:46,719 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(110,Priyanka)
(105,Pawan)
(109,Katrina)
(104,Anubhav)
(108,Ranbir)
grunt> DESCRIBE top_5_employees_final;
top_5_employees_final: {emp_id: int,emp_name: chararray}
grunt>
```

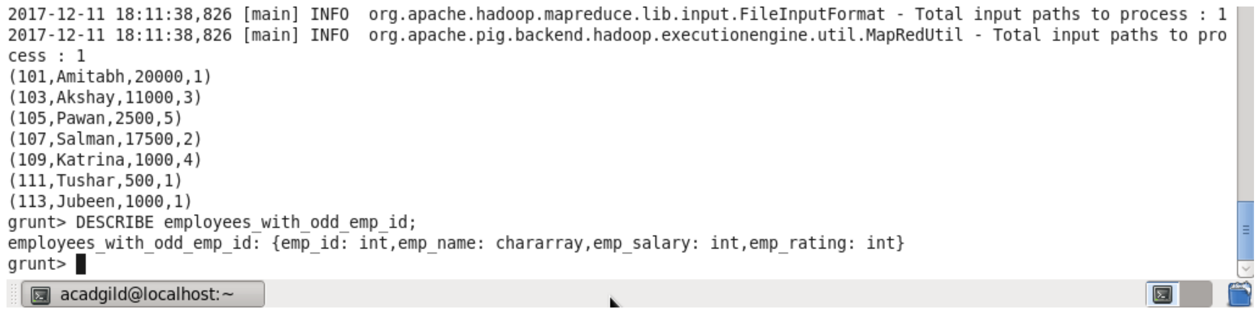
Task 2: Top 3 employees (employee id and employee name) with highest salary, whose employee id is an odd number. (In case two employees have same salary, employee with name coming first in dictionary should get preference).

Solution:

Step 1: Filter out and capture employees with odd employee id.

employees_with_odd_emp_id = FILTER employee_details BY emp_id%2 == 1;

Ouput: DUMP employees_with_odd_emp_id;

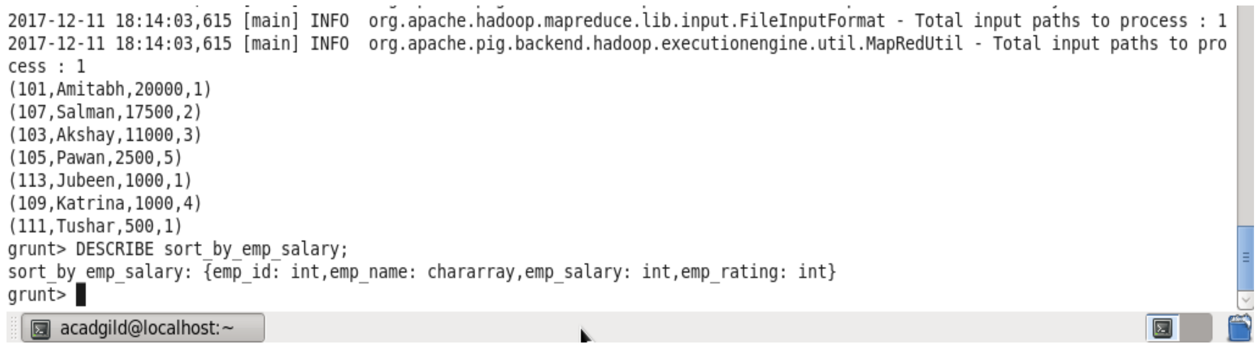


```
2017-12-11 18:11:38,826 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-11 18:11:38,826 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(101,Amitabh,20000,1)
(103,Akshay,11000,3)
(105,Pawan,2500,5)
(107,Salman,17500,2)
(109,Katrina,1000,4)
(111,Tushar,500,1)
(113,Jubeen,1000,1)
grunt> DESCRIBE employees_with_odd_emp_id;
employees_with_odd_emp_id: {emp_id: int,emp_name: chararray,emp_salary: int,emp_rating: int}
grunt>
```

Step 2: Sort employees by their salary in descending order.

sort_by_emp_salary = ORDER employees_with_odd_emp_id BY emp_salary DESC;

Output: DUMP sort_by_emp_salary;

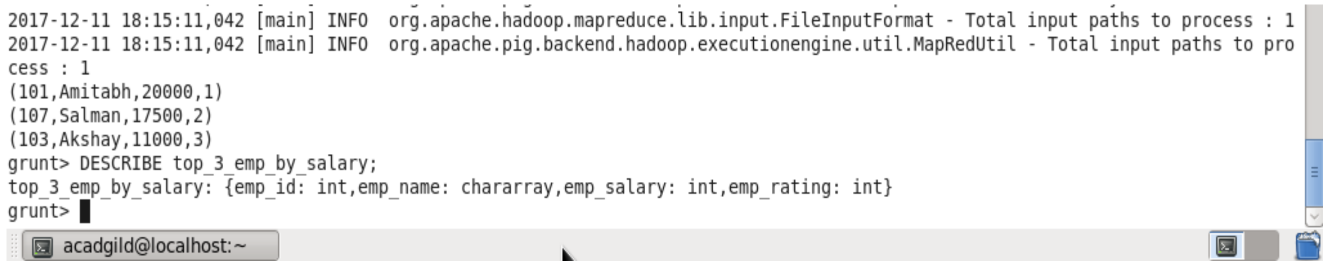


```
2017-12-11 18:14:03,615 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-11 18:14:03,615 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(101,Amitabh,20000,1)
(107,Salman,17500,2)
(103,Akshay,11000,3)
(105,Pawan,2500,5)
(113,Jubeen,1000,1)
(109,Katrina,1000,4)
(111,Tushar,500,1)
grunt> DESCRIBE sort_by_emp_salary;
sort_by_emp_salary: {emp_id: int,emp_name: chararray,emp_salary: int,emp_rating: int}
grunt>
```

Step 3: Get top 3 employees.

top_3_emp_by_salary = LIMIT sort_by_emp_salary 3;

Ouput: DUMP top_3_emp_by_salary;

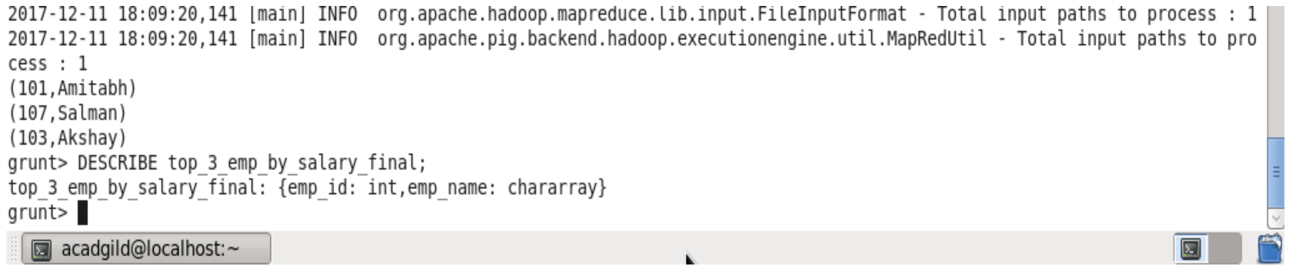


```
2017-12-11 18:15:11,042 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-11 18:15:11,042 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(101,Amitabh,20000,1)
(107,Salman,17500,2)
(103,Akshay,11000,3)
grunt> DESCRIBE top_3_emp_by_salary;
top_3_emp_by_salary: {emp_id: int,emp_name: chararray,emp_salary: int,emp_rating: int}
grunt> 
```

Step 4: Get employee id and employee name from above relation.

top_3_emp_by_salary_final = FOREACH top_3_employees GENERATE emp_id, emp_name;

Output: DUMP top_3_emp_by_salary_final;



```
2017-12-11 18:09:20,141 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-11 18:09:20,141 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(101,Amitabh)
(107,Salman)
(103,Akshay)
grunt> DESCRIBE top_3_emp_by_salary_final;
top_3_emp_by_salary_final: {emp_id: int,emp_name: chararray}
grunt> 
```

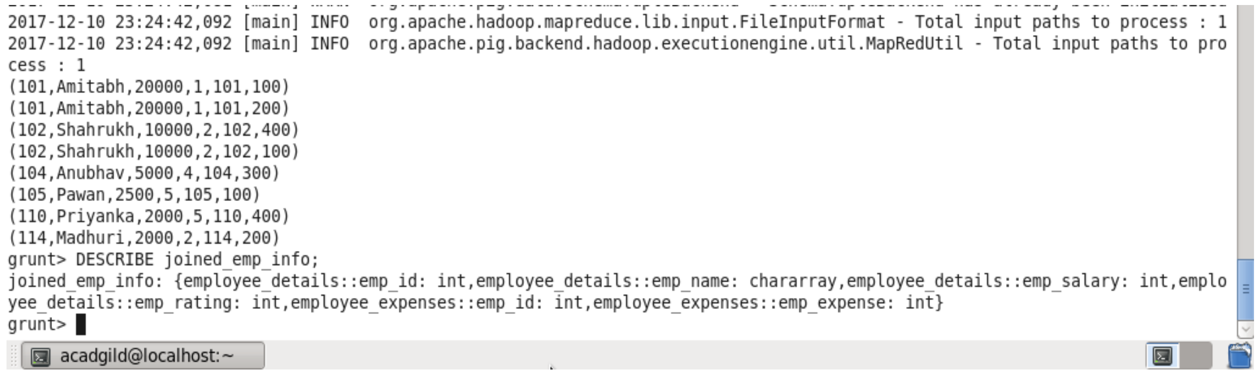
Task 3: Employee (employee id and employee name) with maximum expense (In case two employees have same expense, employee with name coming first should get preference)

Solution:

Step 1: Join relations employee_details and employee_expenses since we require data from both.

joined_emp_info = JOIN employee_details BY emp_id, employee_expenses BY emp_id;

Output: DUMP joined_emp_info;

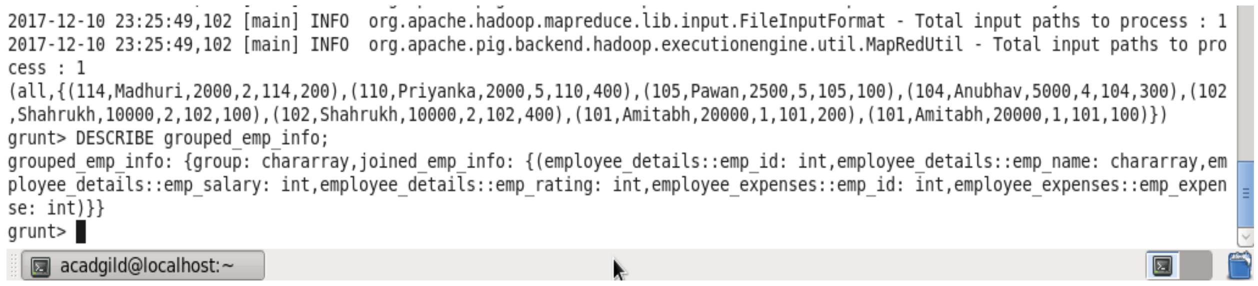


```
2017-12-10 23:24:42,092 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 23:24:42,092 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(101,Amitabh,20000,1,101,100)
(101,Amitabh,20000,1,101,200)
(102,Shahrukh,10000,2,102,400)
(102,Shahrukh,10000,2,102,100)
(104,Anubhav,5000,4,104,300)
(105,Pawan,2500,5,105,100)
(110,Priyanka,2000,5,110,400)
(114,Madhuri,2000,2,114,200)
grunt> DESCRIBE joined_emp_info;
joined_emp_info: {employee_details::emp_id: int,employee_details::emp_name: chararray,employee_details::emp_salary: int,emplo
yee_details::emp_rating: int,employee_expenses::emp_id: int,employee_expenses::emp_expense: int}
grunt> 
```

Step 2: Group data from joined relation by all columns i.e. each tuple/record will be put in separate group.

grouped_emp_info = GROUP joined_emp_info ALL;

Output: DUMP grouped_emp_info;

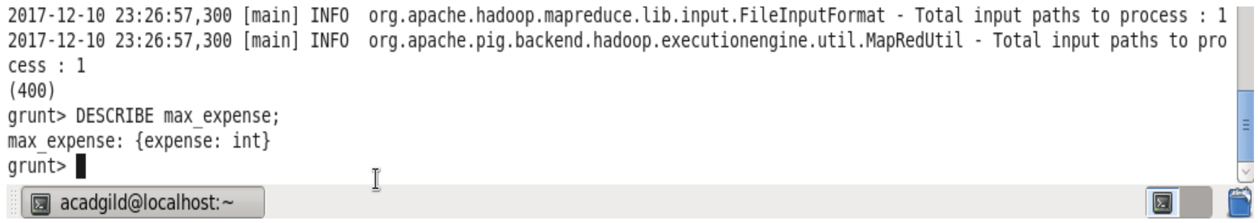


```
2017-12-10 23:25:49,102 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 23:25:49,102 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(all,{(114,Madhuri,2000,2,114,200),(110,Priyanka,2000,5,110,400),(105,Pawan,2500,5,105,100),(104,Anubhav,5000,4,104,300),(102
,Shahrukh,10000,2,102,100),(102,Shahrukh,10000,2,102,400),(101,Amitabh,20000,1,101,200),(101,Amitabh,20000,1,101,100)})
grunt> DESCRIBE grouped_emp_info;
grouped_emp_info: {group: chararray,joined_emp_info: {(employee_details::emp_id: int,employee_details::emp_name: chararray,em
ployee_details::emp_salary: int,employee_details::emp_rating: int,employee_expenses::emp_id: int,employee_expenses::emp_expen
se: int)}}
grunt> 
```

Step 3: Find out the maximum expense value.

max_expense = FOREACH grouped_emp_info GENERATE MAX (joined_emp_info. employee_expenses::emp_expense) AS expense;

Ouput: DUMP max_expense;

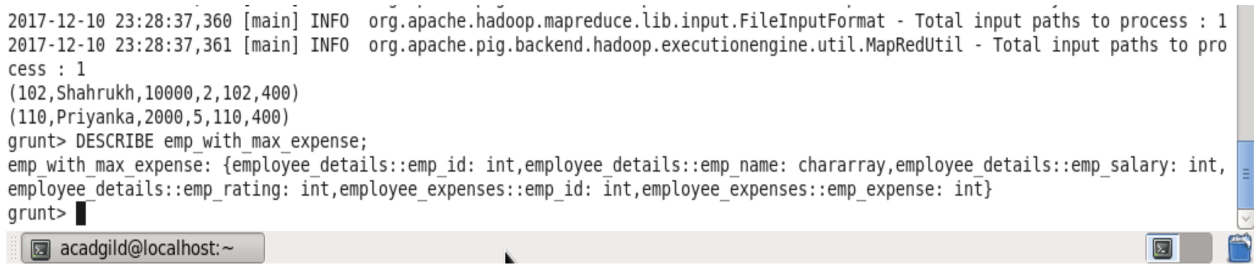


```
2017-12-10 23:26:57,300 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 23:26:57,300 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(400)
grunt> DESCRIBE max_expense;
max_expense: {expense: int}
grunt> 
```

Step 4: Find out the employees who has got expense same as the maximum value.

emp_with_max_expense = FILTER joined_emp_info BY employee_expenses::emp_expense == max_expense.expense;

Output: DUMP emp_with_max_expense;

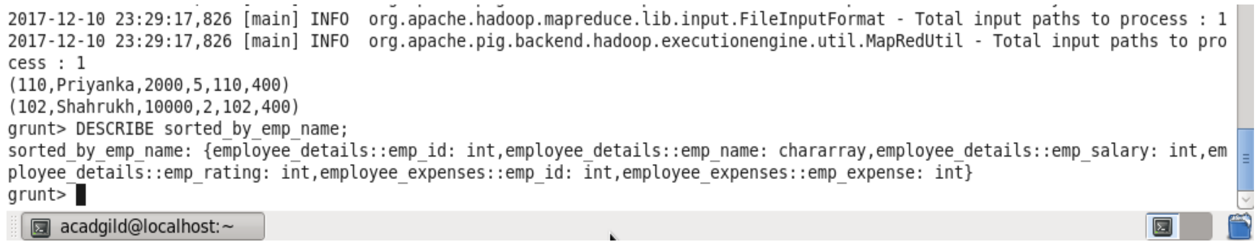


```
2017-12-10 23:28:37,360 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 23:28:37,361 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(102,Shahrukh,10000,2,102,400)
(110,Priyanka,2000,5,110,400)
grunt> DESCRIBE emp_with_max_expense;
emp_with_max_expense: {employee_details::emp_id: int,employee_details::emp_name: chararray,employee_details::emp_salary: int,
employee_details::emp_rating: int,employee_expenses::emp_id: int,employee_expenses::emp_expense: int}
grunt> 
```

Step 5: If there are two or more employees with maximum expenses, sort them out in ascending order.

sorted_by_emp_name = ORDER emp_with_max_expense BY emp_name;

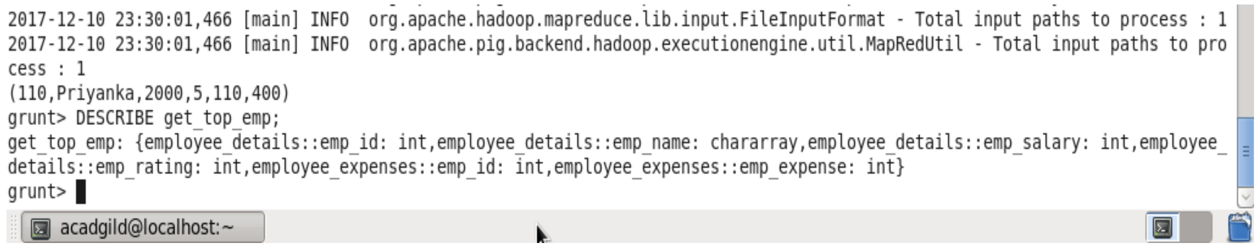
Output: DUMP sorted_by_emp_name;


```
2017-12-10 23:29:17,826 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 23:29:17,826 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(110,Priyanka,2000,5,110,400)
(102,Shahrukh,10000,2,102,400)
grunt> DESCRIBE sorted_by_emp_name;
sorted_by_emp_name: {employee_details::emp_id: int,employee_details::emp_name: chararray,employee_details::emp_salary: int,em
ployee_details::emp_rating: int,employee_expenses::emp_id: int,employee_expenses::emp_expense: int}
grunt> 
```

Step 6: Take first employee record from the above result.

get_top_emp = LIMIT sorted_by_emp_name 1;

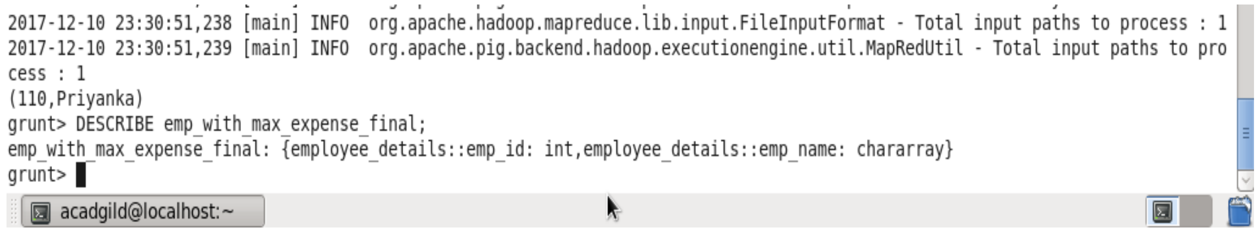
Output: DUMP get_top_emp;


```
2017-12-10 23:30:01,466 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 23:30:01,466 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(110,Priyanka,2000,5,110,400)
grunt> DESCRIBE get_top_emp;
get_top_emp: {employee_details::emp_id: int,employee_details::emp_name: chararray,employee_details::emp_salary: int,employee_
details::emp_rating: int,employee_expenses::emp_id: int,employee_expenses::emp_expense: int}
grunt> 
```

Step 7: Get employee id and employee name from above relation.

emp_with_max_expense_final = FOREACH get_top_emp GENERATE employee_details:: emp_id, employee_details::emp_name;

Output: DUMP emp_with_max_expense_final;


```
2017-12-10 23:30:51,238 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 23:30:51,239 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(110,Priyanka)
grunt> DESCRIBE emp_with_max_expense_final;
emp_with_max_expense_final: {employee_details::emp_id: int,employee_details::emp_name: chararray}
grunt> 
```

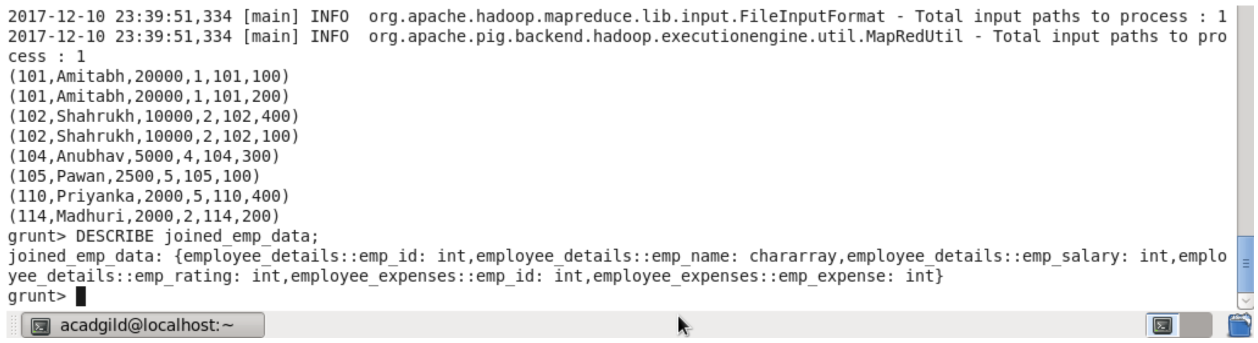
Task 4: List of employees (employee id and employee name) having entries in employee_expenses file.

Solution:

Step 1: Join relations employee_details and employee_expenses since we require data from both.

joined_emp_data = JOIN employee_details BY emp_id, employee_expenses BY emp_id;Output:

Output: DUMP joined_emp_data;

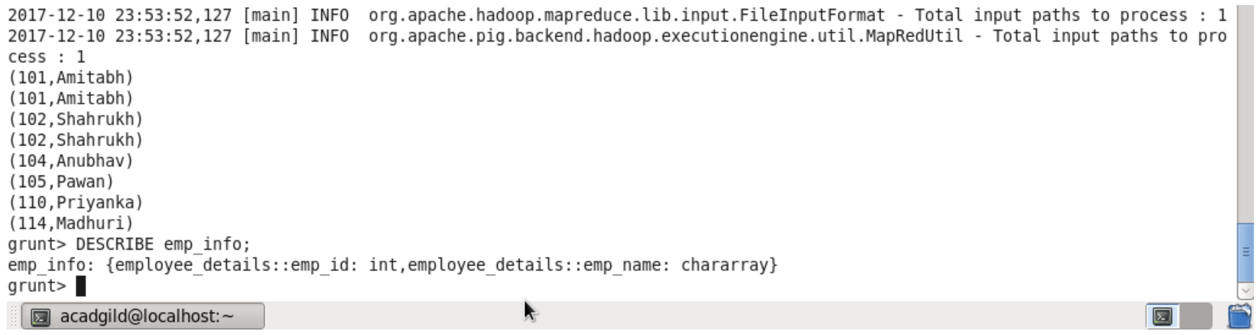


```
2017-12-10 23:39:51,334 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 23:39:51,334 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(101,Amitabh,20000,1,101,100)
(101,Amitabh,20000,1,101,200)
(102,Shahrukh,10000,2,102,400)
(102,Shahrukh,10000,2,102,100)
(104,Anubhav,5000,4,104,300)
(105,Pawan,2500,5,105,100)
(110,Priyanka,2000,5,110,400)
(114,Madhuri,2000,2,114,200)
grunt> DESCRIBE joined_emp_data;
joined_emp_data: {employee_details::emp_id: int,employee_details::emp_name: chararray,employee_details::emp_salary: int,emplo
yee_details::emp_rating: int,employee_expenses::emp_id: int,employee_expenses::emp_expense: int}
grunt> 
```

Step 2: Get all the employee names and their id's from joined relation.

emp_info = FOREACH joined_emp_data GENERATE employee_details::emp_id, employee_details::emp_name;

Output: DUMP emp_info;



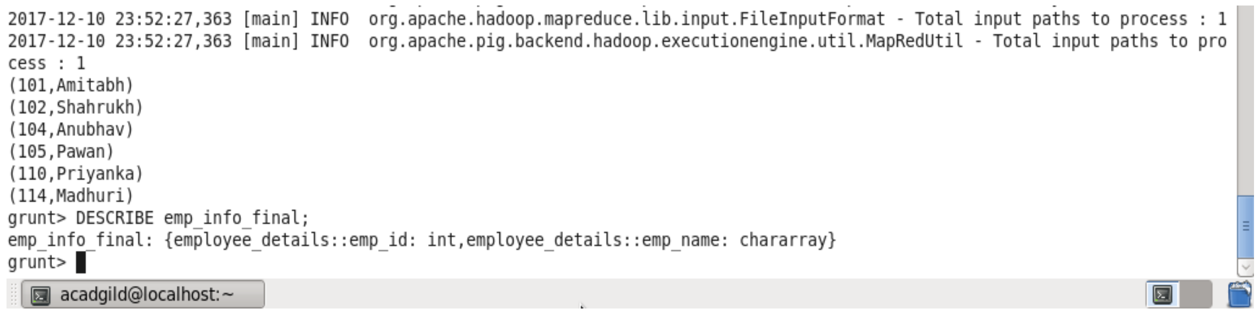
```
2017-12-10 23:53:52,127 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 23:53:52,127 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(101,Amitabh)
(101,Amitabh)
(102,Shahrukh)
(102,Shahrukh)
(104,Anubhav)
(105,Pawan)
(110,Priyanka)
(114,Madhuri)
grunt> DESCRIBE emp_info;
emp_info: {employee_details::emp_id: int,employee_details::emp_name: chararray}
grunt> 
```

Note: There is a tuple in employee_expenses with emp_id = 119 which is not present in employee_details. So I have applied normal join (inner join) on source datasets.

Step 3: As we can see in the above output, there are some duplicate tuples. We need to apply DISTINCT operation to remove duplicates.

emp_info_final = DISTINCT emp_info;

Output: DUMP emp_info_final;



```
2017-12-10 23:52:27,363 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-10 23:52:27,363 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(101,Amitabh)
(102,Shahrukh)
(104,Anubhav)
(105,Pawan)
(110,Priyanka)
(114,Madhuri)
grunt> DESCRIBE emp_info_final;
emp_info_final: {employee_details::emp_id: int,employee_details::emp_name: chararray}
grunt> 
```

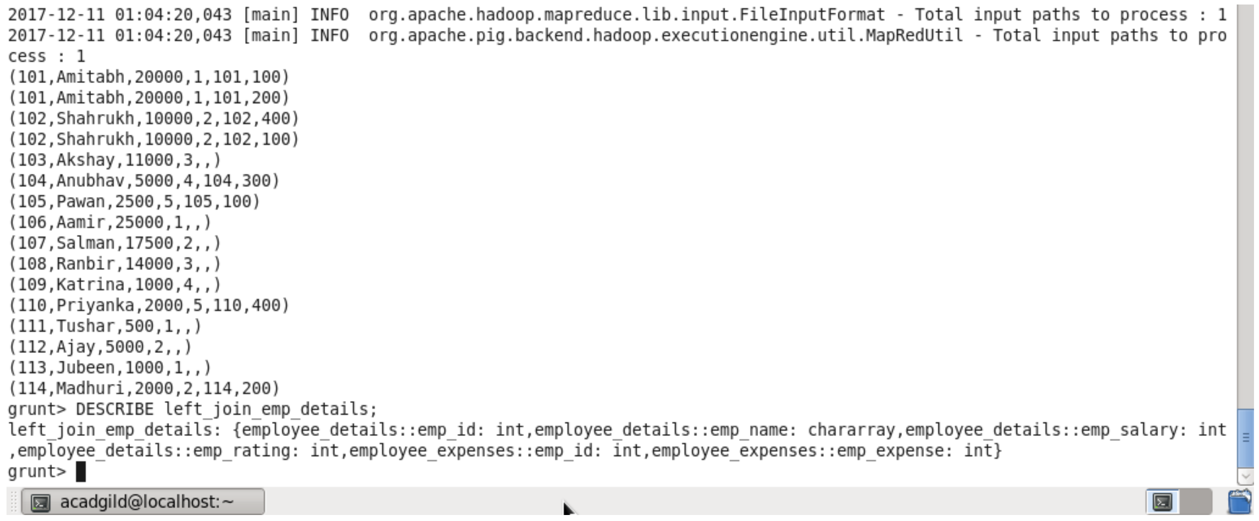
Task 5: List of employees (employee id and employee name) having no entry in employee_expenses file.

Solution:

Step 1: Apply left outer join operation on employee_details and employee_expenses since we require all tuples from the former and none from the latter.

left_join_emp_details = JOIN employee_details BY emp_id LEFT OUTER, employee_expenses BY emp_id;

Output: DUMP left_join_emp_details;

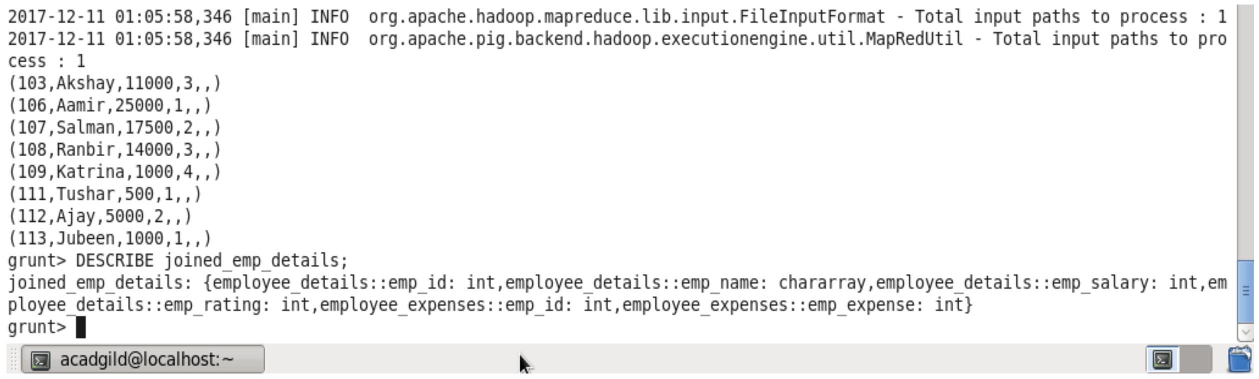


```
2017-12-11 01:04:20,043 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-11 01:04:20,043 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(101,Amitabh,20000,1,101,100)
(101,Amitabh,20000,1,101,200)
(102,Shahrukh,10000,2,102,400)
(102,Shahrukh,10000,2,102,100)
(103,Akshay,11000,3,,)
(104,Anubhav,5000,4,104,300)
(105,Pawan,2500,5,105,100)
(106,Aamir,25000,1,,)
(107,Salman,17500,2,,)
(108,Ranbir,14000,3,,)
(109,Katrina,1000,4,,)
(110,Priyanka,2000,5,110,400)
(111,Tushar,500,1,,)
(112,Ajay,5000,2,,)
(113,Jubeen,1000,1,,)
(114,Madhuri,2000,2,114,200)
grunt> DESCRIBE left_join_emp_details;
left_join_emp_details: {employee_details::emp_id: int,employee_details::emp_name: chararray,employee_details::emp_salary: int
,employee_details::emp_rating: int,employee_expenses::emp_id: int,employee_expenses::emp_expense: int}
grunt> 
```

Step 2: Filter out those records for which emp_id from employee_expenses relation is null.

joined_emp_details = FILTER left_join_emp_details BY employee_expenses::emp_id IS NULL;

Output: DUMP joined_emp_details;

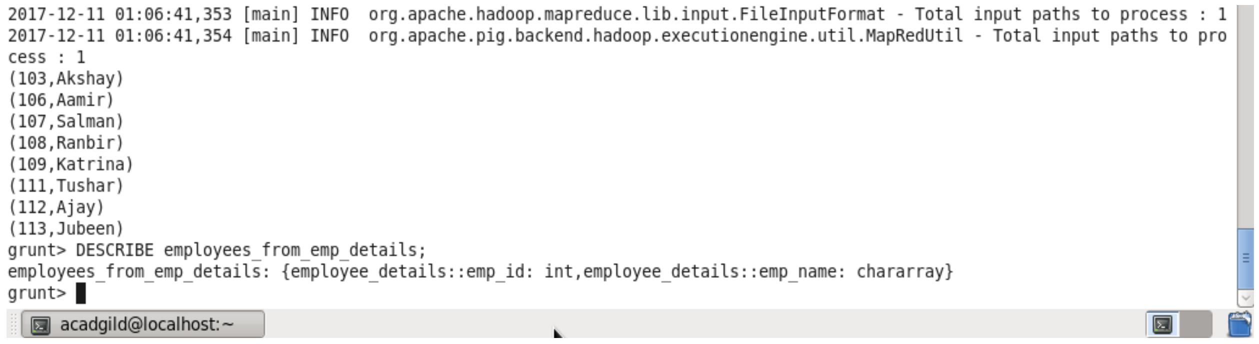


```
2017-12-11 01:05:58,346 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-11 01:05:58,346 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(103,Akshay,11000,3,,)
(106,Aamir,25000,1,,)
(107,Salman,17500,2,,)
(108,Ranbir,14000,3,,)
(109,Katrina,1000,4,,)
(111,Tushar,500,1,,)
(112,Ajay,5000,2,,)
(113,Jubeen,1000,1,,)
grunt> DESCRIBE joined_emp_details;
joined_emp_details: {employee_details::emp_id: int,employee_details::emp_name: chararray,employee_details::emp_salary: int,em
ployee_details::emp_rating: int,employee_expenses::emp_id: int,employee_expenses::emp_expense: int}
grunt> 
```

Step 3: Get employee id and employee name from above relation.

employees_from_emp_details= FOREACH joined_emp_details GENERATE employee_details::emp_id, employee_details::emp_name;

Output: DUMP employees_from_emp_details;



```
2017-12-11 01:06:41,353 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-11 01:06:41,354 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(103,Akshay)
(106,Aamir)
(107,Salman)
(108,Ranbir)
(109,Katrina)
(111,Tushar)
(112,Ajay)
(113,Jubeen)
grunt> DESCRIBE employees_from_emp_details;
employees_from_emp_details: {employee_details::emp_id: int,employee_details::emp_name: chararray}
grunt> 
```