**Session 5: EXPLORIN PIG**

Assignment 5.1

Student Name: Abarajithan SA

Course: Big Data Hadoop & Spark Training

Start Date:  2017-09-09

End Date:  2017-11-26

**Assignment 5.1** – We have **employee\_details** and **employee\_expenses** files. Use local mode while running Pig and write Pig Latin script to get below results:

Contents

[Introduction: 2](#_Toc496350699)

[Input Data Sets: 2](#_Toc496350700)

[employee\_details.txt: 2](#_Toc496350701)

[employee\_expenses.txt: 2](#_Toc496350702)

[Load Data into Pig Storage: 2](#_Toc496350703)

[Task a 3](#_Toc496350704)

[Code 3](#_Toc496350705)

[Output 4](#_Toc496350706)

[Task b 4](#_Toc496350707)

[Code 4](#_Toc496350708)

[Output 5](#_Toc496350709)

[Task c 5](#_Toc496350710)

[Code 5](#_Toc496350711)

[Output 6](#_Toc496350712)

[Task D 6](#_Toc496350713)

[Code 6](#_Toc496350714)

[Output 7](#_Toc496350715)

[Task E 7](#_Toc496350716)

[Code 7](#_Toc496350717)

[Output 8](#_Toc496350718)

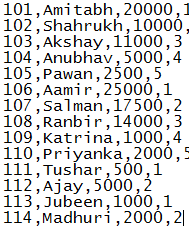
# Introduction:

In this assignment we are going to write PIG Latin codes for the below tasks.

# Input Data Sets:

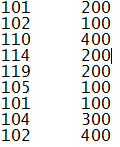
## employee\_details.txt:

<https://github.com/prateekATacadgild/DatasetsForCognizant/blob/master/employee_details.txt>



## employee\_expenses.txt:

[https://github.com/prateekATacadgild/DatasetsForCognizant/blob/master/employee\_expense s.txt](https://github.com/prateekATacadgild/DatasetsForCognizant/blob/master/employee_expense%20s.txt)



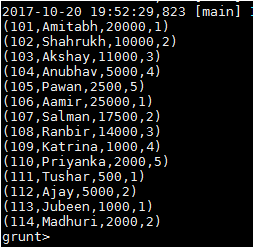
# Load Data into Pig Storage:

Before going into the tasks we are loading the data sets to Pig Storage.

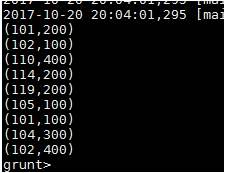
***employee\_details = LOAD '/home/acadgild/hadoop/employee\_details.txt' USING PigStorage(',') AS (id:int, name:chararray, salary:int, rating:int);***

***employee\_expenses = LOAD '/home/acadgild/hadoop/employee\_expenses.txt' USING PigStorage() AS (id:int, expenses:int);***

***DUMP employee\_details;***



***DUMP employee\_expenses;***



# Task a

(a) 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)

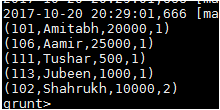
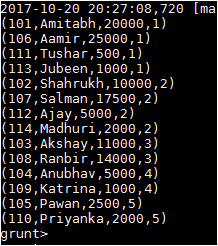
## Code

We are using the function ORDER to order the rating and the ID by ascending order and we are limiting the employee by top 5 with id and the name.

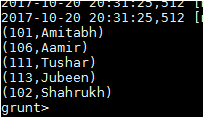
***employee = ORDER employee\_details BY rating asc, id asc;***

***limit\_employee = LIMIT employee 5;***

***top\_employee = FOREACH limit\_employee GENERATE id,name;***



## Output

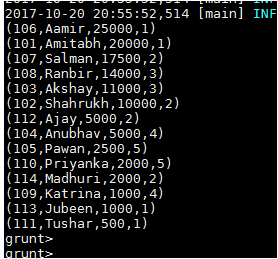
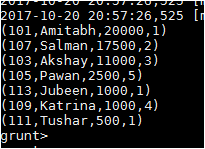
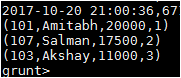


# Task b

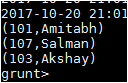
(b) Top 3 employees (employee id and employee name) with highest salary, whose employee id is an odd number. (In case two employees have same salary, employee with name coming first in dictionary should get preference)

## Code

We are using the function ORDER to order the employee\_details by arranging the salary in descending order and filtering the even number using the logic **(id%2)!=0** and limiting the employee by 3 and bringing the result according to the employee ID and Name.

1. ***order\_employee = ORDER employee\_details BY salary DESC;***
2. ***filter\_employee = FILTER order\_employee BY (id%2)!=0;***
3. ***limit\_employee = LIMIT filter\_employee 3;***
4. ***top\_3employee = FOREACH limit\_employee GENERATE id,name;***
5. ***DUMP top\_3employee;***
6. 
7. 
8. 

## Output



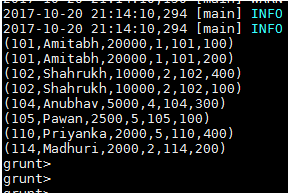
# Task c

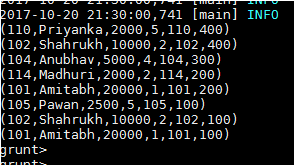
(c) Employee (employee id and employee name) with maximum expense (In case two employees have same expense, employee with name coming first in dictionary should get preference)

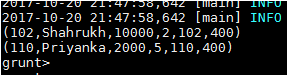
## Code

In this task, we are joining the tables employee\_details and employee\_expenses using the JOIN function and sorting the expenses by Descending order and limiting the employee first 2 rows and bringing the id and name of the employee as the output.

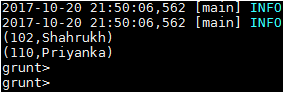
1. ***emp\_det\_exp = JOIN employee\_details BY id, emp\_expenses by id;***
2. ***order\_exp = ORDER emp\_det\_exp BY expenses DESC;***
3. ***limit\_exp = LIMIT order\_exp 2;***
4. ***max\_exp = FOREACH limit\_exp GENERATE $0,$1;***
5. ***DUMP max\_exp;***

1.

2. 

3. 

## Output



# Task D

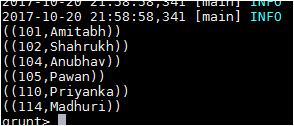
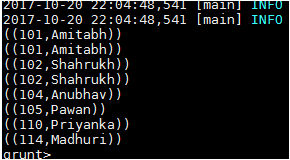
(d) List of employees (employee id and employee name) having entries in **employee\_expenses** file.

## Code

We are joining the table’s employee\_details and employee\_expenses using the JOIN function and removing the redundant tuples from the relation and displaying the ID and name of the employee.

1. ***emp\_det\_exp = JOIN employee\_details BY id, emp\_expenses by id;***
2. ***emp\_names = FOREACH emp\_det\_exp GENERATE($0,$1);***
3. ***names = DISTINCT emp\_names;***
4. ***DUMP names;***

## Output



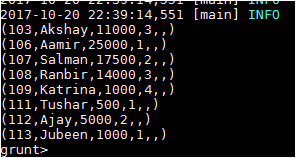
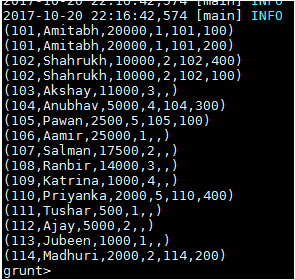
# Task E

(e) List of employees (employee id and employee name) having no entry in **employee\_expenses** file.

## Code

Using the LEFT OUTER JOIN, we are joining the table’s employee\_details and employee\_expenses and filtering the empty values in the columns ($4: employee\_expenses.id and $5: employee\_expenses.expenses) and generating the output by ID and the name of the employee.

1. ***emp\_det\_exp = JOIN employee\_details BY id LEFT OUTER, employee\_expenses BY id;***
2. ***filter\_emp\_det\_exp = FILTER emp\_det\_exp BY $4 Is NULL and $5 Is NULL;***
3. ***gen\_emp\_det\_exp = FOREACH filter\_emp\_det\_exp GENERATE $0,$1;***
4. ***DUMP gen\_emp\_det\_exp;***



## Output

