Session 5

Assignment 1

|  |  |
| --- | --- |
| **Prepared For:** | AcadGild |
|  |  |
| **Document Approval:** | **AcadGild** |
|  |  |
|  |  |
|  |  |
|  |  |
| **Project Title:** | Session 5 – Assignment 1 |
|  |  |
| **Prepared By:** | Duncan Burgess |
|  |  |
|  | dburgess@duncb.com |
|  |  |
| **Primary Engineer:** | Duncan Burgess |
|  |  |
| **Document Reference:** | **Session 5 – Assignment 1** |
|  |  |
| **Start Date:** | 17/09/2017 |
|  |  |
|  |  |

# 

# Contents

[Contents 2](#_Toc493406454)

[Change History 3](#_Toc493406455)

[1. Problem Statement 4](#_Toc493406456)

[2. Solutions 5](#_Toc493406457)

[2.1. Top 5 Rating 6](#_Toc493406458)

[2.2. Top Odd Wage 6](#_Toc493406459)

[2.3. Maximum Expenses 7](#_Toc493406460)

[2.4. Employees with Expenses 8](#_Toc493406461)

[2.5. Employees with no Expenses 9](#_Toc493406462)

# Change History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Document Revision** | **Date** | **Authored By** | **Authorised By** | **Sections Affected** | **Reason for Change** |
| Rev 01 | 17/09/2017 | Duncan Burgess |  | All | Initial release. |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Problem Statement

We have employee\_details and employee\_expenses files. Use local mode while running Pig and

write Pig Latin script to get below results:

employee\_details.txt

employee\_expenses.txt

(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)

(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)

(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)

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

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

# Solutions

The 2 files used for this assignment.

**employee\_details.txt**

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

**employee\_expenses.txt**

101 200

102 100

110 400

114 200

119 200

105 100

101 100

104 300

102 400

## Top 5 Rating

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)

*grunt> EmpInfo = LOAD 'employee\_details.txt' USING PigStorage(',') AS (empId:int, empName:chararray,empSalary:int,empRating:int);*

*grunt> RatedEmp = ORDER EmpInfo by empRating DESC,empName ASC;*

*grunt> Top5emp = LIMIT RatedEmp 5;*

**Results**

(105,Pawan,2500,5)

(110,Priyanka,2000,5)

(104,Anubhav,5000,4)

(109,Katrina,1000,4)

(103,Akshay,11000,3)

## Top Odd Wage

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)

*grunt> EmpInfo = LOAD 'employee\_details.txt' USING PigStorage(',') AS (empId:int, empName:chararray,empSalary:int,empRating:int);*

% modulo operation, where, if output of modulo is 0 = even no remainders != not equal **empId** is odd as there is a remainder.

*oddEmps = FILTER EmpInfo BY empId % 2 != 0;*

Order by salary highest first then name first alphabetically.

*sortEmpBySalary = ORDER oddEmps by empSalary DESC,empName ;*

Retrieve top 3 wage earners from the odd employee ID.

*Top3OddEmps = LIMIT sortEmpBySalary 3;*

**Results**

(101,Amitabh,20000,1)

(107,Salman,17500,2)

(103,Akshay,11000,3)

## Maximum Expenses

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)

**Note** I have worked out total expenses from the expense file as some employees had multiple entries.

*grunt> ind\_expenses = LOAD 'employee\_expenses.txt' USING PigStorage('\t') AS (empId:int, expense:int);*

*grunt> grouped\_expenses = GROUP ind\_expenses BY empId;*

*grunt> total\_expenses = FOREACH grouped\_expenses GENERATE $0 as empID, SUM(ind\_expenses.expense)as totExp;*

**Results**

(101,300)

(102,500)

(104,300)

(105,100)

(110,400)

(114,200)

Join total\_expenses with employee file

*grunt> joinedData = JOIN EmpInfo BY empId, total\_expenses by group;*

**Result**

(101,Amitabh,20000,1,101,300)

(102,Shahrukh,10000,2,102,500)

(104,Anubhav,5000,4,104,300)

(105,Pawan,2500,5,105,100)

(110,Priyanka,2000,5,110,400)

(114,Madhuri,2000,2,114,200)

Sort the results by expense

*grunt> sortedexp = ORDER joinedData by $5 DESC, $0 ASC;*

**Result**

(102,Shahrukh,10000,2,102,500)

(110,Priyanka,2000,5,110,400)

(101,Amitabh,20000,1,101,300)

(104,Anubhav,5000,4,104,300)

(114,Madhuri,2000,2,114,200)

(105,Pawan,2500,5,105,100)

Obtain the highest Expense

maxexpense = LIMIT sortedexp 1;

**Result**

(102,Shahrukh,10000,2,102,500)

## Employees with Expenses

This was calculated in 2.3 as I created a total\_expenses therefore ID 101 has 2 entries 200 + 100 =300, ID 102 has 2 entries 100 + 400 = 500.

101 200

102 100

110 400

114 200

119 200

105 100

101 100

104 300

102 400

Therefore there is no need to perform a DISTINCT as they are there are no duplicates.

Result if just a simple join using original data

grunt> joinedData2 = JOIN empDetails BY empId, ind\_expenses by empId;

**Result**

(101,Amitabh,101,100)

(101,Amitabh,101,200)

(102,Shahrukh,102,400)

(102,Shahrukh,102,100)

(104,Anubhav,104,300)

(105,Pawan,105,100)

(110,Priyanka,110,400)

(114,Madhuri,114,200)

Creating total expense for emplyees

**Result**

(101,Amitabh,20000,1,101,300)

(102,Shahrukh,10000,2,102,500)

(104,Anubhav,5000,4,104,300)

(105,Pawan,2500,5,105,100)

(110,Priyanka,2000,5,110,400)

(114,Madhuri,2000,2,114,200)

## Employees with no Expenses

*grunt> empDetails = LOAD 'employee\_details.txt' USING PigStorage(',') AS (empId:int, empName:chararray);*

Load employee\_details.txt in a variable empDetails. Using PigStorage operator, with delimiter (,) also specified the schema of the data and named the columns as (empId, empName that have data-types integer and chararray respectively).

**Create total\_ expenses**

*grunt> ind\_expenses = LOAD 'employee\_expenses.txt' USING PigStorage('\t') AS (empId:int, expense:int);*

*grunt> grouped\_expenses = GROUP ind\_expenses BY empId;*

*grunt> total\_expenses = FOREACH grouped\_expenses GENERATE $0 as empID, SUM(ind\_expenses.expense)as totExp;*

*grunt> coGroupData = COGROUP empDetails BY empId, total\_expensess by empId;*

COGROUP is used to achieve cross-product join as well as group by. This is to get columns of both the relations in a record.

*grunt> filteredEmp = FILTER coGroupData BY IsEmpty(empExpenses);*

Filter the records to get only those that have no data for columns from employee expenses data. Using IsEmpty().

grunt> EmpNoexpense = FOREACH filteredEmp generate FLATTEN(empDetails);

To flatten the data, i.e., convert bag of tuples into distinct tuples.

**Result from dump**

(103,Akshay)

(106,Aamir)

(107,Salman)

(108,Ranbir)

(109,Katrina)

(111,Tushar)

(112,Ajay)

(113,Jubeen)