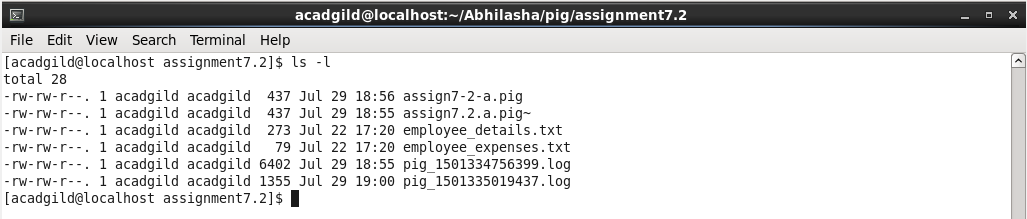
**Big Data And Hadoop**

**Assignment 2 of session7**

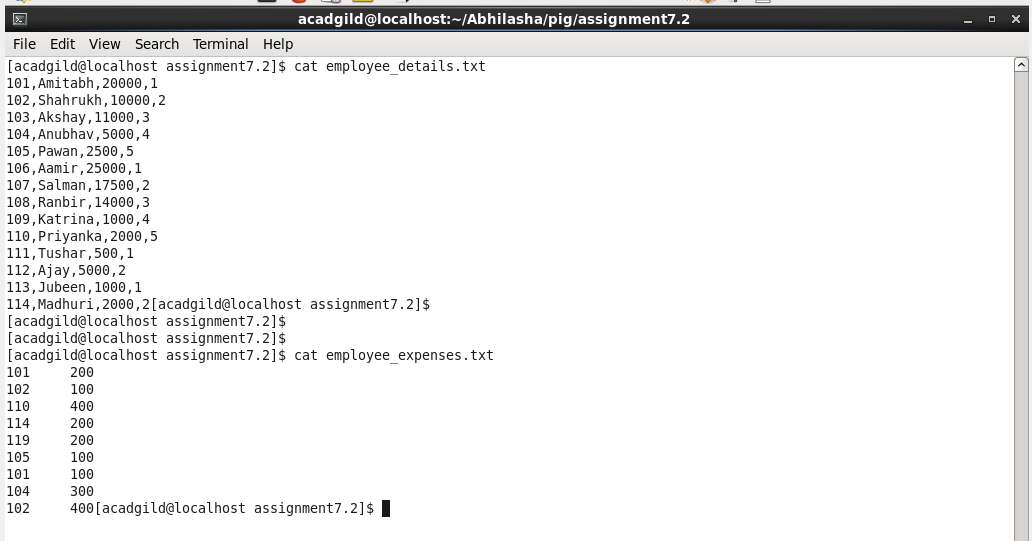
We have employee\_details and employee\_expenses files. Use local mode while running Pig and write Pig Latin script to get below results:

**Input Files:**

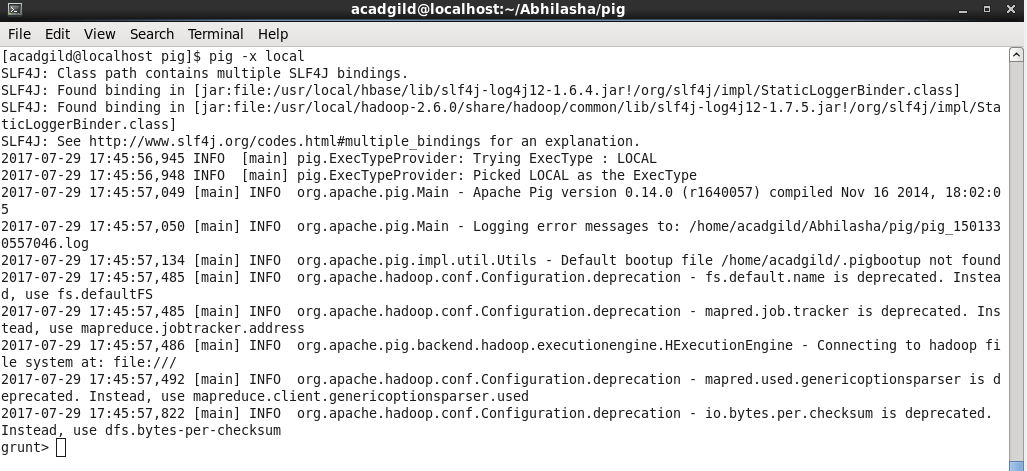
1. Screenshot mentioning the presence of the input files in local directory



1. Content of input files are as follows :



1. Starting pig in local mode. This was needed to test the script, in step by step execution.



This shows that the grunt shell is launched.

**-----------------------------------------------------Problem Statement 1---------------------------------------------------**

**(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)**

**Solution:** The script execution was as follows:



Name of the script executed was assign7-2-a.pig.

The details of script are as follows :

The script was executed in local mode. Hence, **-x local** was used while executing the script.

Step 1: *empDetails = LOAD 'employee\_details.txt' USING PigStorage(',') AS (empId:int, empName:chararray,empSalary:int,empRating:int);*

Load **employee\_details.tx**t in a variable **empDetails**. Using PigStorage operator, we have specified the delimiter for records, i.e., ‘**,**’. Also, we have specified the schema of the data and named the columns as empId, empName, empSalary, empRating that have data-types integer, chararray, integer and integer respectively.

Step 2: sortByRating = ORDER empDetails by empRating DESC, empName;

This is to sort the records based on **empRating** in descending order. Also the records are arranged in dictionary order of names.

Step 3: limitedRecords = LIMIT sortByRating 5;

This is to limit the number of records in output to 5. Hence, used **LIMIT** command.

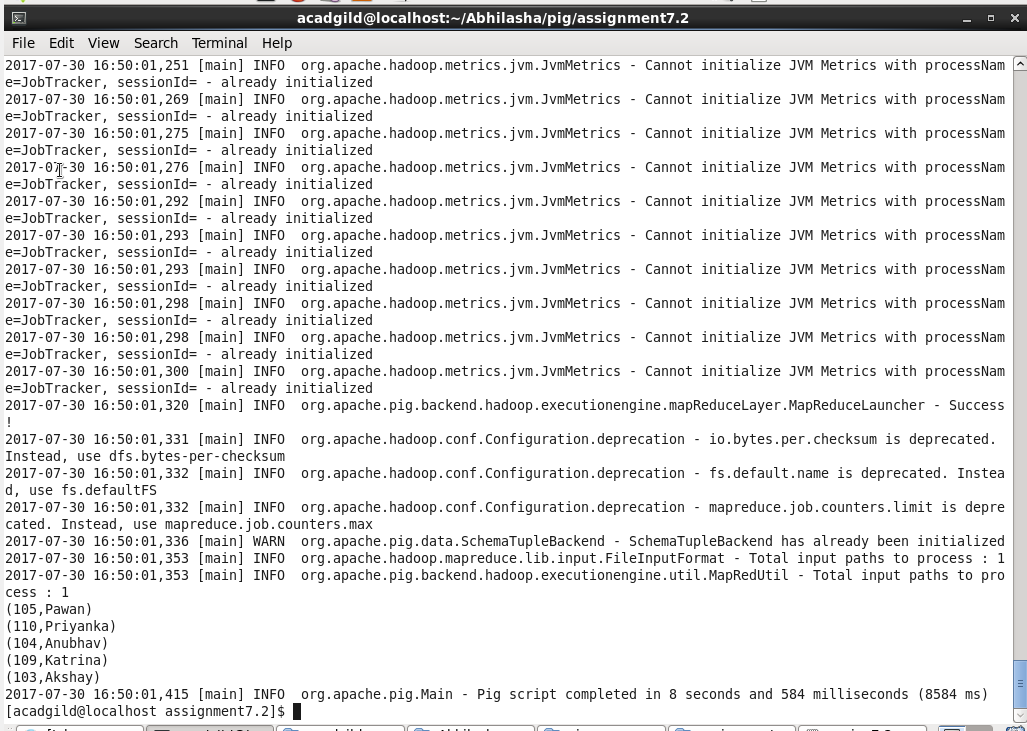
Step 4: requiredEmps = FOREACH limitedRecords generate empId,empName;

This is to fetch only **empId** and **empName** from the resultset.

Step 5: dump requiredEmps;

This is to dump the result on console. The Output is as follows

The output is as follows:

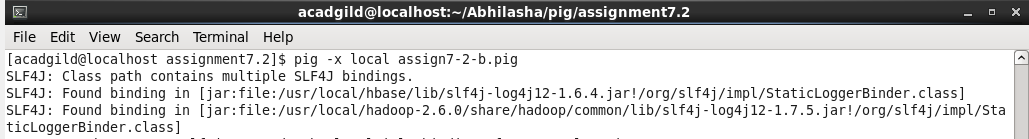


**-----------------------------------------------------Problem Statement 2---------------------------------------------------**

**(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)**

**Solution:** The script execution was as follows:



Name of the script executed was assign7-2-b.pig.

The details of script are as follows:

The script was executed in local mode. Hence, **-x local** was used while executing the script.

Step 1: empDetails = LOAD 'employee\_details.txt' USING PigStorage(',') AS (empId:int, empName:chararray,empSalary:int);

Load **employee\_details**.txt in a variable **empDetails**. Using PigStorage operator, we have specified the delimiter for records, i.e., ‘**,**’. Also, we have specified the schema of the data and named the columns as empId, empName, empSalary that have data-types integer, chararray and integer respectively.

Step 2: filteredEmps = FILTER empDetails BY empId % 2 != 0;

Apply filter to get only those records that have odd **empId**. This is checked by performing a modulo operation, where, if output of modulo is 0 -> even else **empId** is odd.

Step 3: sortBySalary = ORDER filteredEmps by empSalary DESC,empName ;

Sort the records on the basis of **empSalary** in descending order and **empName**.

Step 4: limitedRecords = LIMIT sortBySalary 3;

To fetch only 3 records, we are using **LIMIT** operation.

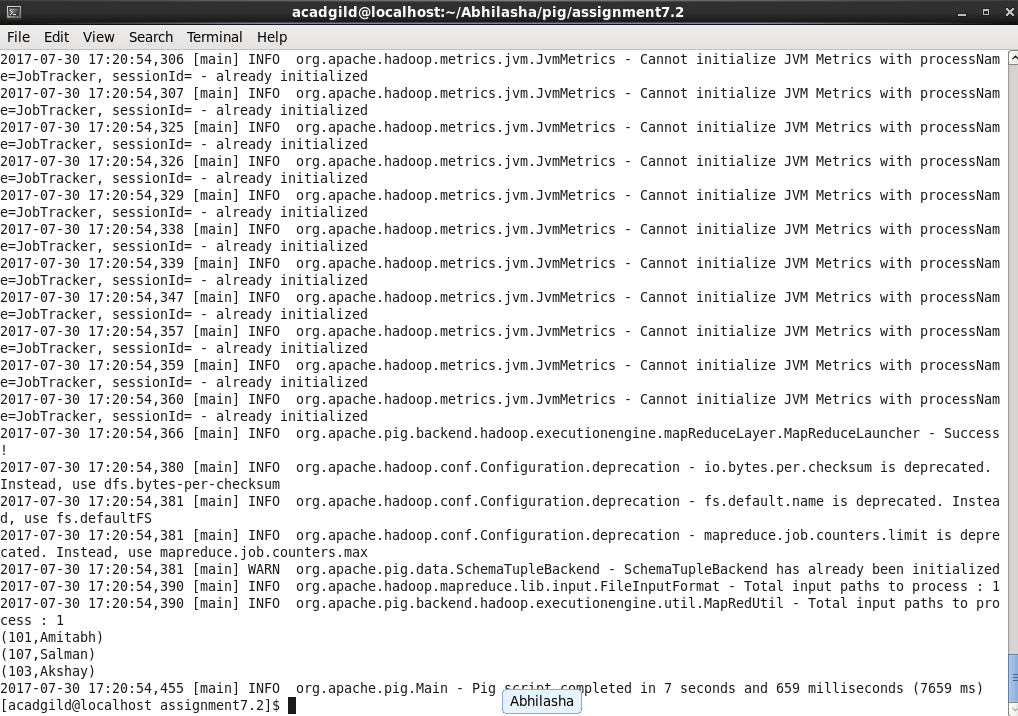
Step 5: requiredEmps = FOREACH limitedRecords generate empId,empName;

To get only empId and empName columns from resultset.

Step 6: dump requiredEmps;

Dump the output on the console.

The output is as follows:

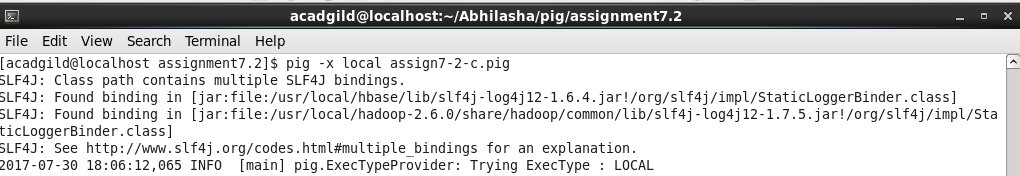


**-----------------------------------------------------Problem Statement 3---------------------------------------------------**

**(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)**

**Solution:** The script execution was as follows:



Name of the script executed was assign7-2-c.pig.

The details of script are as follows:

The script was executed in local mode. Hence, **-x local** was used while executing the script.

Step 1: empDetails = LOAD 'employee\_details.txt' USING PigStorage(',') AS (empId:int, empName:chararray);

Load **employee\_details**.txt in a variable **empDetails**. Using PigStorage operator, we have specified the delimiter for records, i.e., ‘**,**’. Also, we have specified the schema of the data and named the columns as empId, empName that have data-types integer and chararray respectively.

Step 2: empExpenses = LOAD 'employee\_expenses.txt' USING PigStorage(' ') AS (empId:int, expenses:int);

Load **employee\_expenses**.txt in a variable **empExpenses**. Using PigStorage operator, we have specified the delimiter for records, i.e., ‘’. Also, we have specified the schema of the data and named the columns as empId, expenses that have data-types integer and integer respectively.

Step 3: joinData = JOIN empDetails BY empId, empExpenses by empId;

Perform join of empDetails and empExpenses on employee id.

Step 4: sortedData = ORDER joinData by empExpenses::expenses DESC,empDetails::empName ;

Sort the resultset by expenses in descending order and by employee name.

Step 5: firstRecord = LIMIT sortedData 1;

Get first record only. Hence, use **LIMIT** operator.

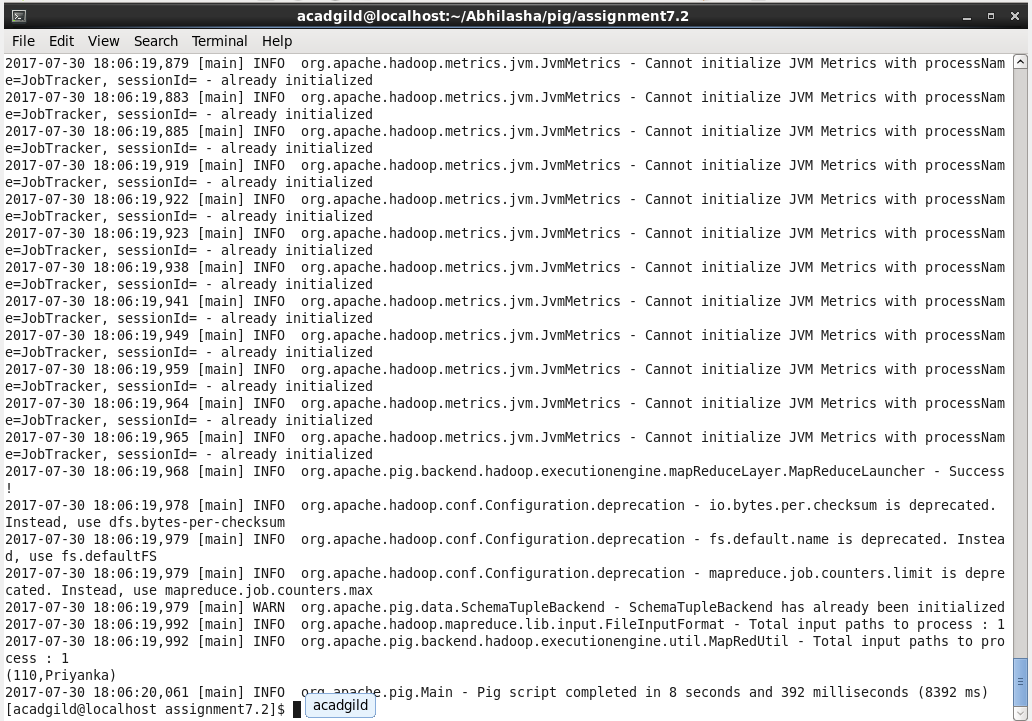
Step 6: requiredEmp = FOREACH firstRecord generate empDetails::empId,empDetails::empName;

To get only empId and empName columns from resultset.

Step 7: dump requiredEmp;

Dump the output on the console.

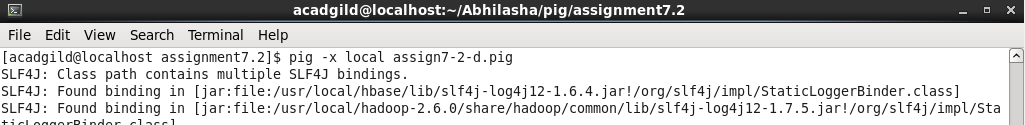
The output is as follows :



**-----------------------------------------------------Problem Statement 4---------------------------------------------------**

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

**Solution:** The script execution was as follows:



Name of the script executed was assign7-2-d.pig.

The details of script are as follows:

The script was executed in local mode. Hence, **-x local** was used while executing the script.

Step 1: empDetails = LOAD 'employee\_details.txt' USING PigStorage(',') AS (empId:int, empName:chararray);

Load **employee\_details**.txt in a variable **empDetails**. Using PigStorage operator, we have specified the delimiter for records, i.e., ‘**,**’. Also, we have specified the schema of the data and named the columns as empId, empName that have data-types integer and chararray respectively.

Step 2: empExpenses = LOAD 'employee\_expenses.txt' USING PigStorage(' ') AS (empId:int);

Load **employee\_expenses**.txt in a variable **empExpenses**. Using PigStorage operator, we have specified the delimiter for records, i.e., ‘’. Also, we have specified the schema of the data and named the column as empId that has data-type integer.

Step 3: joinData = JOIN empDetails BY empId, empExpenses by empId;

Perform join of empDetails and empExpenses on employee id.

Step 4: distinctRecords = DISTINCT joinData;

Data in 'employee\_expenses.txt' has multiple entries for a few empIds. Hence, to remove duplicates, we have used **DISTINCT**.

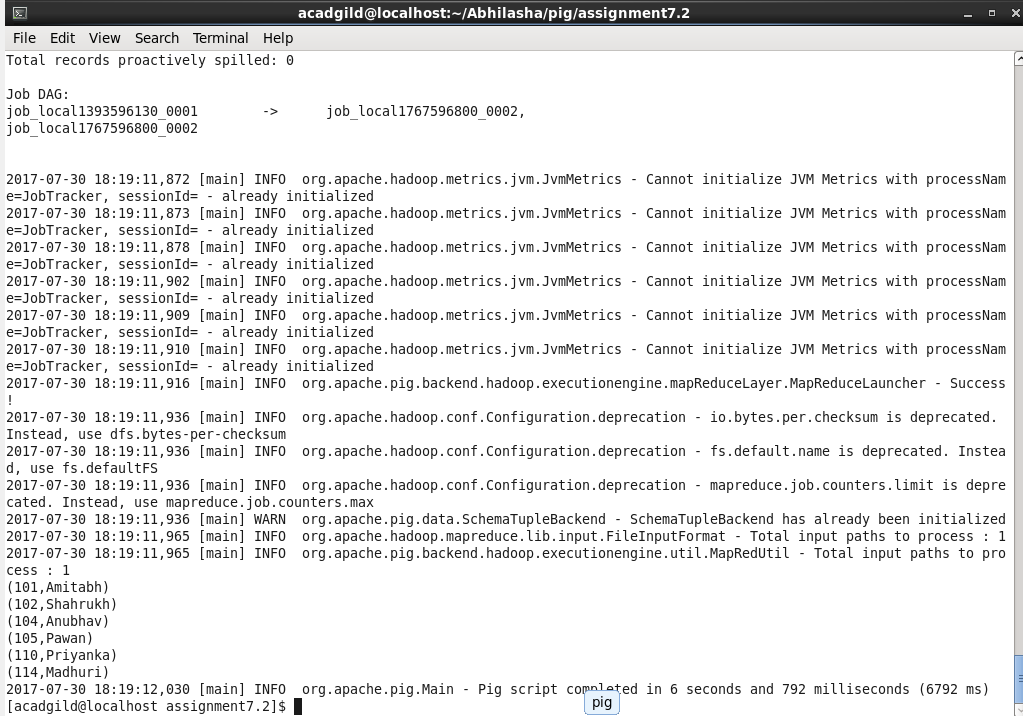
Step 5: requiredEmps = FOREACH distinctRecords generate empDetails::empId,empDetails::empName;

To get only empId and empName columns from resultset.

Step 7: dump requiredEmp;

Dump the output on the console.

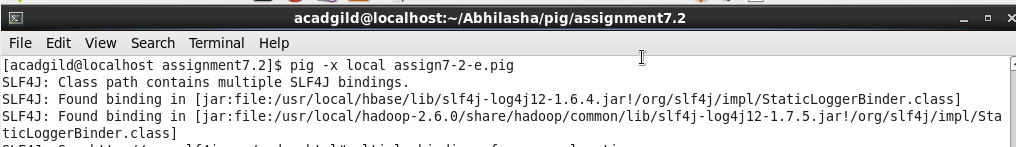
The output is as follows :



**-----------------------------------------------------Problem Statement 5---------------------------------------------------**

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

**Solution:** The script execution was as follows:



Name of the script executed was assign7-2-e.pig.

The details of script are as follows :

The script was executed in local mode. Hence, **-x local** was used while executing the script.

Step 1: empDetails = LOAD 'employee\_details.txt' USING PigStorage(',') AS (empId:int, empName:chararray);

Load **employee\_details**.txt in a variable **empDetails**. Using PigStorage operator, we have specified the delimiter for records, i.e., ‘**,**’. Also, we have specified the schema of the data and named the columns as empId, empName that have data-types integer and chararray respectively.

Step 2: empExpenses = LOAD 'employee\_expenses.txt' USING PigStorage(' ') AS (empId:int);

Load **employee\_expenses**.txt in a variable **empExpenses**. Using PigStorage operator, we have specified the delimiter for records, i.e., ‘’. Also, we have specified the schema of the data and named the column as empId that has data-type integer.

Step 3: coGroupData = COGROUP empDetails BY empId, empExpenses by empId;

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

Step 4: filteredData = FILTER coGroupData BY IsEmpty(empExpenses);

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

Step 5: flattenedData = FOREACH filteredData generate FLATTEN(empDetails);

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

Step 7: dump flattenedData;

Dump the output on the console.

The output is as follows :

