Big Data And Hadoop

Session 16 - Assignment 1

Problem Statement 1:

Get a list of employees who receive a salary less than 100, compared to their immediate employee with higher salary in the same unit Solution:

Input File

The input file is present on the local file system at /home/acadgild/Abhilasha/hive as follows:

```
Σ.
                                            acadgild@localhost:~/Abhilasha/hive
File Edit View Search Terminal Help
[acadgild@localhost hive]$ pwd
/home/acadgild/Abhilasha/hive
[acadgild@localhost hive]$ ls -l
total 588
-rw-rw-r--. 1 acadgild acadgild
                                  5226 Oct 2 18:31 commands
-rw-rw-r--. 1 acadgild acadgild
                                   4901 Oct 1 20:56 commands~
-rw-rw-r--. 1 acadgild acadgild
                                   170 Sep 17 14:17 complexData
-rw-rw-r--. 1 acadgild acadgild
                                   437 Sep 16 19:29 dataset_Session14.txt
-rw-rw-r--. 1 acadgild acadgild
                                   159 Sep 19 08:49 emp_Details
-rw-rw-r--. 1 acadgild acadgild
                                   159 Sep 19 08:24 emp Details~
-rw-rw-r--. 1 acadgild acadgild
                                    84 Sep 17 13:43 empDetails~
                                   107 Sep 18 22:00 employee.csv
-rw-rw-r--. 1 acadgild acadgild
-rw-rw-r-- 1 acadgild acadgild
-rw-rw-r--. 1 acadgild acadgild
                                    107 Sep 18 21:51 employee.csv
                                    282 Oct 2 18:17 Emp Sal
                                     0 Oct
                                             2 18:13 Emp Sal-
-rw-rw-r--. 1 acadgild acadgild
-rw-rw-r--. 1 acadgild acadgild
                                    43 Oct 1 19:36 locations
-rw-rw-r--. 1 acadgild acadgild
                                    43 Oct 1 19:36 locations~
-rw-rw-r--. 1 acadgild acadgild 518669 Sep 19 22:14 olympix data.csv
                                   4096 Sep 19 08:53 output
drwxrwxr-x. 2 acadgild acadgild
drwxrwxr-x. 2 acadgild acadgild
                                   4096 Sep 19 22:59 output-Query3
drwxrwxr-x. 2 acadgild acadgild
                                   4096 Sep 19 22:55 output-Query
-rw-rw-r--. 1 acadgild acadgild
                                   170 Sep 17 14:17 Unsaved Document 1~
-rw-rw-r--. 1 acadgild acadgild
                                    97 Oct 1 19:35 users
-rw-rw-r--. 1 acadgild acadgild
                                    85 Oct 1 19:34 users~
[acadgild@localhost hive]$
```

The content of the input file can be seen using the **cat** command as follows:

```
acadgild@localhost:~/Abhilasha/hive
File Edit View Search Terminal Help
[acadgild@localhost hive]$ cat Emp Sal
001
       Amit
             105
                      Data Minning
002
       Pankaj 85
                      Data Engineer
003
              110
       Kiran
                      Data Scientist
004
       Arpitha 95
                      Data Engineer
005
       Viraj 105
                      Data Mining
006
       Smitha 80
                      Data Analyst
007
       Supriya 90
                      Data Engineer
008
       Vihan 120
                       Data Scientist
               100
                      Data Engineer
       Emma
       Siddharth
                              Data Engineer
[acadgild@localhost hive]$
```

Start hive: We start the hive command line by executing the command hive as shown below:



The above snapshot also shows that hive prompt has started. A pre-requisite to use hive is to start mysql server. This was done using the command sudo service mysqld start.

Step 1: We use **SHOW DATABASES** command to list the databases present. The database we will be using is **custom** as shown below:



Step 2: We use **USE custom** command to make use of custom database, as shown below:



Step 3: We create the table using **CREATE TABLE** command. The fields of the table are: id, name, salary, and department.



Step 4: **SHOW TABLES** command will help us verify that the table is created.



Step 5: **DESCRIBE** command will help us verify the schema of the table as follows:



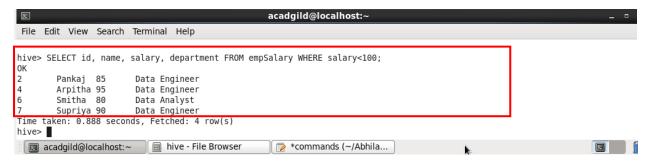
Step 6: Next is to load the data from input file, which is located at **/home/acadgild/Abhilasha/hive** as follows. We use the **LOAD** command and use the keyword **LOCAL** to specify that the file is present in the local file system and not HDFS.



Step 7: Using the **SELECT** * query, we can display the complete data as follows:

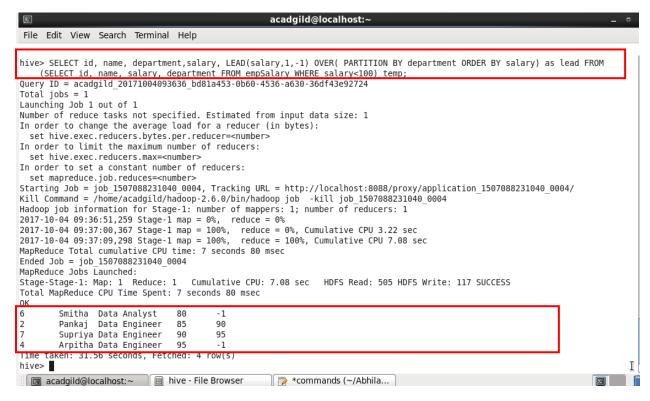


Step 8: We need only those employees that have salary less than 100. Hence, we apply a predicate salary<100 as follows:



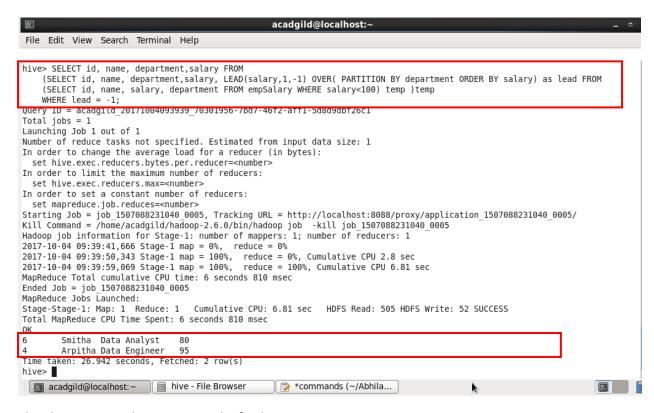
The screenshot also mentions the output of the query executed.

Step 9: We add a use another query on top of the query in step 8, to make use of **LEAD** function. LEAD function will give us the immediate higher salary. If the employee itself has the highest salary in the department, it has no other lead and hence mentioning the default value to be -1. We need this data per department and hence, using the clause **PARTITION BY department** as follows:



The screenshot also mentions the output of the query executed.

Step 10: Now we need only those employees that have highest salary in the department. This can be achieved by using another query on top of the query in step 9. The rows with lead column having value -1 will have highest salary in respective departments. So the final query is as follows:



The above screen shot mentions the final output.

Problem Statement 2:

List of all employees who draw higher salary than the average salary of that department Solution:

In the problem statement 1, we have mentioned of the input file. Same file is being used for this problem statement as well.

Steps 1 to 7 are applicable for this problem statement as well, hence, not repeating them again.

The first part of the solution is to find the average salary per department. If we use GROUP BY and aggregate function AVG, we will get one row per department. However, we need average salary on every row, for every employee. Hence, we use analytic and windowing in hive.

The query used to achieve this is

Select id, name, department, salary as empSalary, AVG (salary) OVER (PARTITION BY department) as avgSalary FROM empSalary;

Here, OVER clause is used to produce results per row, based on the computations of the values of the rows in a particular window. We decide the window based on the department. This is similar to

grouping the rows by department. Then we have made use of the aggregate function AVG to get average salary per department.

The execution of this command and its result are as follows:

```
acadgild@localhost:~
 File Edit View Search Terminal Help
umns. Also check for circular dependencies
Underlying error: org.apache.hadoop.hive.ql.parse.SemanticException: Line 1:79 Invalid table alias or column reference 'depar
tment': (possible column names are:
hive> SELECT id, name,department,salary as empSalary, AVG(salary) OVER (PARTITION BY department) as avgSalary
FROM empSalary;
Query ID = acadgild_20171002183636_f89653b3-8331-4300-b77c-85d7403d14fa
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1506948170615_0001, Tracking URL = http://localhost:8088/proxy/application_1506948170615_0001/Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job -kill job_1506948170615_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-10-02 18:36:24,097 Stage-1 map = 0%, reduce = 0%
2017-10-02 18:36:31,911 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.61 sec
2017-10-02 18:36:41,711 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 5.2 sec
MapReduce Total cumulative CPU time: 5 seconds 200 msec
Ended Job = job 1506948170615 0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 5.2 sec HDFS Read: 505 HDFS Write: 317 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 200 msec
        Smitha Data Analyst
                                          80.0
                                 80
10
        Siddharth
                                                  94.0
                        Data Engineer
                                          100
                Data Engineer
                                          94.0
9
        Emma
                                 100
        Supriya Data Engineer
                                 90
                                          94.0
        Arpitha Data Engineer
                                 95
                                          94.0
        Pankaj Data Engineer
                                 85
                                          94.0
        Virai
                Data Mining
                                 105
                                          105.0
                Data Minning
                                 105
                                          105.0
        Amit
8
        Vihan
                Data Scientist 120
                                          115.0
        Kiran
                Data Scientist
Time taken: 33.984 seconds, Fetched: 10 row(s)
hive>
```

Now, we need list of those employees whose salary is greater than average salary of the department. For this, we will use a nested query and apply the predicate on top of the above query as follows:

To the output of above query, we have applied the predicate empSalary > avgSalary as follows

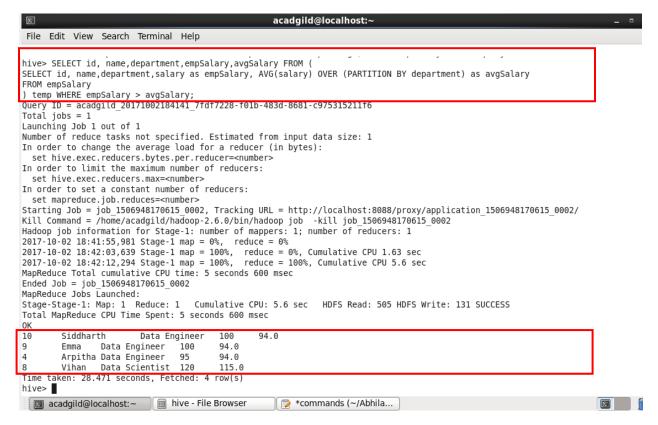
Select id, name, department, empSalary, avgSalary FROM

(

Select id, name, department, salary as empSalary, AVG (salary) OVER (PARTITION BY department) as avgSalary FROM empSalary

) temp

where empSalary>avgSalary;



The output shows the result containing the employees with their salary greater than average salary. The rows of the output are id, name, department, salary of employee and average salary of the department.