

## 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:

```
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~
-rw-rw-r--. 1 acadgild acadgild 107 Sep 18 22:00 employee.csv
-rw-rw-r--. 1 acadgild acadgild 107 Sep 18 21:51 employee.csv~
-rw-rw-r--. 1 acadgild acadgild 282 Oct  2 18:17 Emp_Sal
-rw-rw-r--. 1 acadgild acadgild 0 Oct  2 18:13 Emp_Sal~
-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
drwxrwxr-x. 2 acadgild acadgild 4096 Sep 19 08:53 output
drwxrwxr-x. 2 acadgild acadgild 4096 Sep 19 22:59 output-Query3
drwxrwxr-x. 2 acadgild acadgild 4096 Sep 19 22:55 output-Query4
-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 Mining
002 Pankaj 85 Data Engineer
003 Kiran 110 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
009 Emma 100 Data Engineer
010 Siddharth 100 Data Engineer
[acadgild@localhost hive]$
```

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

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
[acadgild@localhost ~]$ hive  
Logging initialized using configuration in jar:file:/usr/local/hive/lib/hive-common-0.14.0.jar!/hive-log4j.properties  
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/hive-jdbc-0.14.0-standalone.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.  
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]  
hive>
```

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:

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
hive> SHOW DATABASES;  
OK  
acadgild  
h1  
custom  
default  
Time taken: 0.067 seconds, Fetched: 4 row(s)  
hive>
```

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

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
hive> USE custom;  
OK  
Time taken: 0.036 seconds  
hive>  
hive> USE custom;  
OK  
Time taken: 0.036 seconds  
hive>
```

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

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
hive> CREATE TABLE empSalary  
(  
  id INT,  
  name STRING,  
  salary INT,  
  department STRING  
)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY '\t';  
OK  
Time taken: 0.463 seconds  
hive>
```

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

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
hive> SHOW TABLES;  
OK  
empdetails  
empsalary  
olympic  
temperature_data  
temperature_data_vw  
Time taken: 0.074 seconds, Fetched: 5 row(s)  
hive>
```

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

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
hive> DESCRIBE empSalary  
> ;  
OK  
id                int  
name              string  
salary            int  
department         string  
Time taken: 0.204 seconds, Fetched: 4 row(s)  
hive>
```

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.

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
hive> LOAD DATA LOCAL INPATH '/home/acadgild/Abhilasha/hive/Emp_Sal'  
INTO TABLE empSalary;  
Loading data to table custom.empsalary  
Table custom.empsalary stats: [numFiles=1, totalSize=282]  
OK  
Time taken: 1.625 seconds  
hive>
```

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

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
hive> SELECT * FROM empSalary;  
OK  
1      Amit      105      Data Mining  
2      Pankaj     85       Data Engineer  
3      Kiran      110      Data Scientist  
4      Arpitha    95       Data Engineer  
5      Viraj      105      Data Mining  
6      Smitha     80       Data Analyst  
7      Supriya    90       Data Engineer  
8      Vihan      120      Data Scientist  
9      Emma       100      Data Engineer  
10     Siddharth  100      Data Engineer  
Time taken: 0.441 seconds, Fetched: 10 row(s)  
hive>
```

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

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
hive> SELECT id, name, salary, department FROM empSalary WHERE salary<100;  
OK  
2      Pankaj     85       Data Engineer  
4      Arpitha    95       Data Engineer  
6      Smitha     80       Data Analyst  
7      Supriya    90       Data Engineer  
Time taken: 0.888 seconds, Fetched: 4 row(s)  
hive>
```

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:

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
hive> SELECT id, name, department,salary, LEAD(salary,1,-1) OVER( PARTITION BY department ORDER BY salary) as lead FROM  
(SELECT id, name, salary, department FROM empSalary WHERE salary<100) temp;  
Query ID = acadgild_20171004093636_bd81a453-0b60-4536-a630-36df43e92724  
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_1507088231040_0004, Tracking URL = http://localhost:8088/proxy/application_1507088231040_0004/  
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job -kill job_1507088231040_0004  
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1  
2017-10-04 09:36:51,259 Stage-1 map = 0%, reduce = 0%  
2017-10-04 09:37:00,367 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.22 sec  
2017-10-04 09:37:09,298 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 7.08 sec  
MapReduce Total cumulative CPU time: 7 seconds 80 msec  
Ended Job = job_1507088231040_0004  
MapReduce Jobs Launched:  
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 7.08 sec HDFS Read: 505 HDFS Write: 117 SUCCESS  
Total MapReduce CPU Time Spent: 7 seconds 80 msec  
OK  
6      Smitha  Data Analyst      80      -1  
2      Pankaj  Data Engineer    85      90  
7      Supriya Data Engineer    90      95  
4      Arpitha Data Engineer    95      -1  
Time taken: 31.56 seconds, Fetched: 4 row(s)  
hive>
```

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:

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
  
hive> SELECT id, name, department,salary FROM  
  (SELECT id, name, department,salary, LEAD(salary,1,-1) OVER( PARTITION BY department ORDER BY salary) as lead FROM  
  (SELECT id, name, salary, department FROM empSalary WHERE salary<100) temp )temp  
  WHERE lead = -1;  
Query ID = acadgild_20171004093939_70301956-70d7-46f2-af11-5d809dbf26c1  
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_1507088231040_0005, Tracking URL = http://localhost:8088/proxy/application_1507088231040_0005/  
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job -kill job_1507088231040_0005  
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1  
2017-10-04 09:39:41,666 Stage-1 map = 0%, reduce = 0%  
2017-10-04 09:39:50,343 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.8 sec  
2017-10-04 09:39:59,069 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 6.81 sec  
MapReduce Total cumulative CPU time: 6 seconds 810 msec  
Ended Job = job_1507088231040_0005  
MapReduce Jobs Launched:  
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 6.81 sec HDFS Read: 505 HDFS Write: 52 SUCCESS  
Total MapReduce CPU Time Spent: 6 seconds 810 msec  
OK  
6      Smitha Data Analyst      80  
4      Arpitha Data Engineer    95  
Time taken: 26.942 seconds, Fetched: 2 row(s)  
hive>
```

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 'department': (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  
OK  
6 Smitha Data Analyst 80 80.0  
10 Siddharth Data Engineer 100 94.0  
9 Emma Data Engineer 100 94.0  
7 Supriya Data Engineer 90 94.0  
4 Arpitha Data Engineer 95 94.0  
2 Pankaj Data Engineer 85 94.0  
5 Viraj Data Mining 105 105.0  
1 Amit Data Mining 105 105.0  
8 Vihan Data Scientist 120 115.0  
3 Kiran Data Scientist 110 115.0  
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;**

```
acadgild@localhost:~  
File Edit View Search Terminal Help  
hive> 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;  
Query ID = acadgild_20171002184141_7fdf7228-f01b-483d-8681-c975315211f6  
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_0002, Tracking URL = http://localhost:8088/proxy/application_1506948170615_0002/  
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job -kill job_1506948170615_0002  
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1  
2017-10-02 18:41:55,981 Stage-1 map = 0%, reduce = 0%  
2017-10-02 18:42:03,639 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.63 sec  
2017-10-02 18:42:12,294 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 5.6 sec  
MapReduce Total cumulative CPU time: 5 seconds 600 msec  
Ended Job = job_1506948170615_0002  
MapReduce Jobs Launched:  
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 5.6 sec HDFS Read: 505 HDFS Write: 131 SUCCESS  
Total MapReduce CPU Time Spent: 5 seconds 600 msec  
OK  
10 Siddharth Data Engineer 100 94.0  
9 Emma Data Engineer 100 94.0  
4 Arpitha Data Engineer 95 94.0  
8 Vihan Data Scientist 120 115.0  
Time taken: 28.471 seconds, Fetched: 4 row(s)  
hive>
```

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.