EPBI 414 (Fall 2016) - Assignment 7

Advanced SQL

Overview

In this homework, you will put to use the new SQL toolkit that was covered in this unit. In Part 1, you will create some aggregate values based on different filters and grouping variables. In Part 2, you'll do some very basic database administration for a extremely simple table. In both cases, you should feel free to use whatever platform suits you best - MySQL Workbench or the command line.

Logistics

Submit your assignment in a .zip file labeled in the following matter:

<Case ID>_EPBI414_Fall2016_A7.zip

So, if your Case ID is tar9, you would submit the following zip file:

tar9_EPBI414_Fall2016_A7.zip

Part 1

Your supervisor liked the work you did originally about salary differences among male and female employees, based on the data found in `epbi414_employee_data` on `hal`. They want you to expand on that work and produce a few more reports, generally in a bit more detail. To do this, you are going to write a SQL program that has three major queries in it.

Query 1

To start, your supervisor wants you to generate a table that breaks down some basic statistics by gender and title. For each combination of gender and title in the database, they want to know the following information:

- The total number of employees in that group
- The average tenure, or years of employment, of the employees in that group, as of January 1, 2016
- The average age of the employees in that group, as of January 1, 2016
- The average salary of the employees in that group

You should take into consideration the following points:

- Your supervisor is only interested in information about current employees. You should be sure to filter out any employees who no longer work for the company.
- Your supervisor tells you that you can assume that everyone's employment has been continuous since their hire date. In other words, you do not need to parse the title and

salary records to see if you can identify gaps in service.

Write a SQL query that generates the requested data. To help you check your work, you can compare what you get to the sample output, which you can find in the table `assignment7_part1_qry1` in the `epbi414_employee_data` database. [26 points]

Query 2

It turns out that the database you have may not have the best data quality. Your supervisor has asked you to generate a simple report describing how much information you have on people's starting salaries - that is, the salary they had when they were first hired by the company. They would like you to create a table containing four rows, breaking down the number of male and female employees who have, and do not have, a starting salary record. Unlike before, they would like you to ***include*** all employees, including those who no longer work with the company.

A few notes for you:

- In this context, you may find that a left join is useful, since it would let you guarantee that you would get back all of the rows in the employee table if you did a join using it.
- For the purposes of identifying an individual's starting salary, you can assume that their hire date will line up with the "from date" of their first salary.
- This is a case where combining the `UNION` command and a calculated field containing a constant value could be useful!

Write a SQL query that generates the requested data. As with the previous part, you can compare your results to the sample output, which you can find in the table `assignment7_part1_qry2` in the `epbi414_employee_data` database. [20 points]

Query 3

After reviewing the results of your second query, your supervisor thinks you can do one final analysis. They want you to break down the distribution of starting salaries among male and female employees. Specifically, they want to get the total number of employees and the minimum, average, and maximum starting salaries, broken down by gender. As with the second query, they would like you to ***include*** all employees.

A few more notes:

- Some of what you did in Query 2 is likely going to be useful here, but it can be streamlined a bit given that you are not interested in the missing records.
- Consider whether you want a left join or an inner join in this context.

Again, as with the previous parts, the sample output is available in the table `assignment7_part1_qry3` in the `epbi414_employee_data` database. Write a query to generate the requested data. [14 points]

Save all three of your queries, along with any other code (like a `USE` statement at the beginning), into a file named:

```
sql_e414_f2016_A7_P1.sql
```

You can create a file like that using any text editor (like `joe`). Include this file with your homework submission. Remember that your code should be commented and should follow some kind of consistent coding guidelines in order to make it easier to understand! [5 points for style and comments]

Part 2

In this part, you will do some very simple database administration tasks to get a feel for how these things work. To let each of you have an opportunity to try out the MySQL toolkit, a new database has been placed on `hal` for each of you. The database is named epbi414_<your Network ID>. So, for instance, the one set up for me is epbi414_tar9.

Within this database, you have full privileges to `INSERT`, `UPDATE`, `DELETE`, `CREATE`, `ALTER`, `DROP`, and to make temporary tables. This should help prevent anyone from accidentally dropping everyone else's database while playing around with the homework!

To get started, you need to open the SQL file `sql_EPBI414_A7_Part2_Data_Table.sql`. This file is your "setup" button and your "reset" button. You should edit the `USE` statement near the beginning of the file to point it toward your personal database. Once you've done that, you can use the rest of the commands in the file to set up a small example data table that you use for the rest of Part 2.

Note that MySQL's usage of parentheses - the (and) symbols - in the `CREATE TABLE X AS ...` style syntax is sometimes a bit unusual. In certain cases, parentheses are fine, but in others - like the one in the script you are reviewing - it seems to break. Just be aware that, like most things, there are small amounts of variation in how the syntax works on each RDBMS!

Once you have the `patient_tracker` table set up in your personal database, then do the following steps:

1. Using the `CREATE TABLE X AS ...` style syntax, create a backup copy of `patient_tracker` inside your personal database, called `patient_tracker_bak`. All you need to do is create a new table that is identical to the original one. [6 points]

2. Patient 105 was assigned to the wrong gender during data entry. She was falsely set as being male by one of the data entry personnel. Please change her record in `patient_tracker` so that the gender is female. [7 points]

3. Next, please add in this new individual, who for some reason was not entered using the standard data entry tool. Here are the particulars:

   Patient ID: 111

Gender is female, race is American Indian or Alaska Native
Contacted should be 1, all other flags should be 0
[10 points]

4. Finally, it turns out that Patient ID 104 should not have been entered into the system at all - the whole entry was a data error. So, you need to delete the record of Patient ID 104 from the `patient_tracker` table. [7 points]

Save all of the SQL code you used to accomplish this, along with any other code (like a `USE` statement at the beginning), into a file named:

`sql_e414_f2016_A7_P2.sql`

You can create a file like that using any text editor (like `joe`). Include this file with your homework submission. Remember that your code should be commented and should follow some kind of consistent coding guidelines in order to make it easier to understand! [5 points for style and comments]