Assignment: Practice Formulating SQL Queries

Avraham Leff

COM3580: Spring 2024

1 Assignment-Specific Packaging

The general packaging is unchanged from the basic "Homework Requirements" (see slides from first lecture and "Homework Policies for COM 3580" on Piazza).

This assignment's "DIR" **must be named** *SQL3*. Your report **must be named** *$DIR/SQL3.pdf*.

Some of the questions require you to put the queries into an execution file named *SQL31.sql*. For those questions, omit the query and screenshot from the report: just be careful that the script name matches the requirements and that it's checked into Git (into the assignments `$DIR`).

2 Motivation

The assignment skips "trivial SQL", dropping you immediately into less-trivial query formulation.

I don't expect you to be able to solve these problems after the first or even the second lecture! I'm releasing it now only to give students with some SQL experience a chance to get started with the assignments and to give students with less SQL experience a sense of my expectations.

Students with less SQL experience are urged to supplement lecture material with self-education from the plethora of on-line SQL material.

3 Database Installation

Some of the exercises in this assignment are based on the Silberschatz textbook's *University database*. Specifically, you'll use the "large database" version of this schema.

Use the scripts from this git repository to create and populate the "large university database".

> The one difference between the "small" and "large" versions of the database is that table names in the small version have an **_s** suffix. For example: *student_s* rather than *student*.

When writing your SQL scripts, make sure to *\connect* to the correct database.

4 Requirements

> Note: in all such exercises, if necessary, you are responsible for (re)initializing the database before executing your SQL commands. All subsequent interactions with the database must execute the specified **in the specified order**!

- If the question calls for you to check in an `sql` script, all you need to do is supply the named script.

  Otherwise: for each of the following *natural language* queries, you must

  1. Show me a formatted SQL "translation" of that query
  2. Show a **clear snapshot** of your POSTGRESQL executing that query and the results therefrom.

> Important: if I ask for fields a and b, the result set **must** include exactly those fields, in the **specified order**.
>
> I will deduct points if you fail to follow the specified format.

- Note: your grade depends only on "correctness": i.e., the result of your query!

1. For the employee database (shown below) *find the ID of each employee with no manager*. Order your results by ascending employee id, return no more than ten tuples.

   employee (*ID*, *person_name*, *street*, *city*)
   works (*ID*, *company_name*, *salary*)
   company (*company_name*, *city*)
   manages (*ID*, *manager_id*)

   An employee database comprised of four tables. Primary key fields are denoted by being underlined.

   This query is straightforward, but note:

   - An employee may simply have no manager listed or may have a manager listed but the value is NULL. Either of these conditions should be treated as "no manager".
   - To encourage a "set approach to SQL", I'm not supplying a sample database for this question. Instead: you'll supply a `psql` script named `SQL31.sql` in the assignment's directory (above), and I will invoke that script against my populated database.

     Your script should not qualify the names of the tables **in any way**: instead refer only to e.g., `employee` and `manages`.

2. This question also refers to the "employee database" (shown above).

   "Return the ID and name of each employee who lives in the same city as the location of the company for which the employee works."

Order the results by ascending ID, no more than 5 tuples in the result set.

3. This question also refers to the "employee database" (shown above).

   "Return the ID and name of each employee who lives in the same city as does her or his manager".

   Order the results by ascending ID, no more than 5 tuples in the result set.

4. This question also refers to the "employee database" (shown above).

   "Return the ID and name of each employee who earns more than the average salary of all employees in their company".

   Order the results by ascending ID, no more than 5 tuples in the result set.

5. This question also refers to the "employee database" (shown above).

   "Return the name of company that has the smallest payroll and that company's payroll."

6. *"Return all instructors, but include only their IDs and the number of sections taught by that instructor.*

   If an instructor has not taught any section, return 0. Order the result set by ascending instructor id, return no more than five results.

7. *"Return all course sections offered in 2008, together with the ID and name of each instructor teaching the section"*.

   If a section has more than one instructor, that section should appear as many times in the result as it has instructors. If a section does not have any instructor, it should still appear in the result with the instructor name set to "—".

   - Each tuple must include: year, semester, course id, section id, instructor id, instructor name
   - Order by ascending year, and, within that ordering, by ascending course id.
   - Return no more than twelve tuples.

8. *"Return all department names together with the total number of instructors in each department."*. The result set must be ordered by ascending number of instructors, and within that ordering, by ascending department name. Return no more than seven tuples.

   Departments that have no instructors and departments that have zero instructors must still be included in the result set