CS-GY 6083-A, Principles of Database Systems, Fall 2021

Homework 3: SQL
Due at 11:59pm EDT on Thursday, November 4

Description

This assignment covers SQL. When writing SQL queries, keep in mind that your answer should be correct for *any valid instance* of the given database schema. In other words – you are writing programs that should compute correctly on any valid input, not only on the particular instance you are given as an example.

Grading

This assignment is made up of 4 problems, collectively worth 90 points, or 8% of the overall course grade. If this assignment is submitted late, you will receive no credit. This assignment is to be completed individually. Please consult the course syllabus for a description of our academic honesty policy.

Submit the files hw3_part1.sql, hw3_part2.sql, hw3_part3.sql, hw3_part4.sql containing your solution for parts 1 through 4. We will execute the queries to test their correctness on several different database instances. For each part, we encourage you to use a relational database to test your queries. We are providing schema definitions and instances for each problem so you can get started. Keep the schemas as they are, but feel free to modify the instances as you see fit to test your queries. **Importantly**:

- You may not assume anything about the specific instance on which your queries will be executed, beyond what's stated in the questions.
- Do not use the "distinct" keyword unless it's necessary to compute the correct answer.
- Your SQL queries should compile in PostgreSQL on jedi.poly.edu.
- Your queries should produce results in the specified format and in the specified order.

Submission instructions

Submit your assignment on NYU Brightspace.

- Create one file per problem: hw3_part1.sql, hw3_part2.sql, hw3_part3.sql, hw3_part4.sql.
- Submit all files as a single zip archive named <netId>_HW3.zip (e.g., abc123_HW3.zip if your netId is abc123).

Part 1 (30 points): Jobs, Positions, Candidates

Consider relation schemas and instances in the file **Jobs.sql**. In each question below, write a single SQL query that computes the required answer. Each question is worth an equal number of points.

(a) For each candidate, list all (job, grade) pairs for which that candidate has at least the 2 of the required skills, with at least the required number of years for each skill. Results should have the schema (candidate, job, grade). Sort results by candidate, with ties broken by job, and then by grade.

(b) For each candidate, list companies that have positions such that the candidate's salary requirements (Candidates.salary) are no higher than the company's salary budget (Positions.salary). You can assume that a candidate who does not specify a salary requirement is willing to consider any offered position, irrespective of the company's salary budget. Results should have the schema (candidate, company). Sort results by candidate, with ties broken by company.

(c) List candidates who are overqualified for at least one of the skills of the 'DEV' grade 3 position. A candidate is overqualified if they have the required skill (e.g., Linux) and their number of years of experience is no lower than the required number of years (e.g., 3 years for Linux). Result should have the schema (candidate). List each candidate exactly once and sort results by candidate.

(d) List pairs of candidates (candidate1, candidate2) such that:

- Candidate1 and candidate2 have DB and python experience;
- Candidate1 and candiate2 have a stated salary requirement;
- Candidate1 has more years of combined DB and python experience; and
- Candiate1 has a lower salary requirement than candidate 2.

Return each pair of candidates exactly once, with the schema (candidate1, candidate2). Sort results by candidate 1, with ties broken by candidate 2.

(e) For each skill, compute the minimum, maximum and average number of years of experience among the candidates who have that skill, and the minimum, maximum and average salary requirement among the candidates who have that skill. Do not return skills for which no candidate exists in our database. Round averages to 1 digit after the decimal point. Sort results by skill. Results should have the schema (skill, min_years, max_years, avg_years, min_salary, max_salary, avg_salary).

(f) List pairs of positions that both require 'Linux' as a skill. Return each pair exactly once, i.e., do not return both ('DEV', 2, 'IBM', 'DEV', 2, 'SAP') and ('DEV', 2, 'SAP', 'DEV', 2, 'IBM'). Result should have the schema: (job1, grade1, company1, job2, grade2, company2). Sort results by job1, grade1, company1, job2, grade2, company2.

Part 2 (20 points): Tennis Players

Consider relation schemas and instances in the file **Tennis.sql**. In each question below, **except question (c)**, write a single SQL query that computes the required answer. Each question is worth an equal number of points.

(a) List pairs of countries (country1, country2) in which country1 both has fewer players and a lower GDP than country 2. Sort results by country1, break ties by country2.

(b) For players who ranked first for at least two years in a row, list their country name and GDP. Result should have the schema (player_name, country, GDP). List each player exactly once, sort results by player name.

(c) List country name, GDP and population of each country. For countries whose players have ranked first in the past, also list the name of each such player. Result should have the schema (country, GDP, population, top_ranked_player). Sort results by country, with ties broken by player name. **Note**: Consider using a view as part of your solution to this question.

(d) List names of countries that had a top-ranked tennis player either during a non-leap year (leap: any year divisible by 4) or after 2010. Result should have the schema (country). Return each country exactly once and sort results by country name.

Part 3 (20 points): Dishes

Consider relation schemas and instances in the file **Dishes.sql**. In each question below, write a single SQL query that computes the required answer. Each question is worth an equal number of points.

(a) List dishes that contain no fruit, and in which at least one of the ingredients is a meat or a seafood. List each dish exactly once. Result should have the schema (dish), and should be returned in alphabetical order.

(b) List foods of which no fewer than 3 units are used in at least 2 dishes. List each food exactly once. Results should have the schema (food), and should be returned in alphabetical order.

(c) For each dish, compute the number of calories contributed to it by the foods in each category. Results should have the schema (dish, category, calories). Sort results alphabetically by dish, with ties broken by category.

(d) Compute the nutritional value (number of calories) of dishes that contain at most 3 ingredients and have at least 1000 calories. Results should have the schema (dish, total_calories), and should be returned in alphabetical order.

Part 4 (20 points): Cocktails

Consider relation schemas and instances in the file **Cocktails.sql**. In each question below, except (d), write a single SQL query that computes the required answer. Each question is worth an equal number of points.

(a) List names of cocktails that either contain ice or use less than 4 units in total of ingredients. The result should have the schema (cname). Sort results by cname.

(b) List names of cocktails that use ingredients with an average unit cost of less than 3.5. Result should have the schema: (cname, avg_cost). Round the average to 1 digit after the decimal point. Sort results by cname.

(c) List names of cocktails, together with the profit made on the sale of each of them. For a cocktail, we define profit as the difference between its price and the combined cost of its ingredients. Only include information about cocktails that consist of at most 2 ingredients. Result should have the schema (cname, profit). Sort results by cname.

(d) List pairs of cocktails such that the first is stronger, in terms of alcohol content, than the second. To compute the alcohol content of a cocktail, you should assume that the amount of alcohol contained in each unit of a cocktail's ingredient is specified by its ABV (alcohol by volume). Result should have the schema: (cname1, cname2). Sort results alphabetically by cname1, breaking ties by cname2. **Hint**: Consider using subqueries or views to answer this question.