HW3: Even More SQL queries

361

For the following queries, use the database 361 in Postgres.

(1) Write a query that returns all the numbers that only appear in either in the French table or the English table, but not in both. Your query should return one single column with numbers. In order to answer this query, you will have to use the COALESCE function, for which you will have to look up the documentation online.

(2) Write a query that returns the number of integers that appears only in the English table and the number of integers that appear appears in the French table. Your query should return one row with two columns named (onlyenglish, onlyfrench).

(3) For this question, you need to first add more data into the database

(a) First Create a series of SQL statements that create a new table called Number with one column name id of type integer. Insert all numbers from 1-10. Submit all statements.

(b) Write a query that gives you for each number from 1-10 the translation into English and French (if available in the database). Your query should return 3 columns: id, etext, and ftext. If you don't have the translation of a number in the database, the entry should be NULL.

YOUR OWN VEHICLE REGISTRATION DATABASE

For the following queries, we only provide the intended schema. But you need to create your own database and insert a minimum number of "interesting data" that allows you to verify that the queries you are typine are correct.

Consider the following database of drivers and their insurances:

Vehicle (VIN, model, year)

Driver (license, name, birthday)

Insured (license, VIN, premium)

The key fields are underlined: VIN is the key for Vehicle (an integer), license is the key for Driver (an integer), and VIN and license together form the key for Insured. We do not show the foreign keys. Premium represents the price each driver pays per month.

(4) Creating the database

(a) Create the database with above 3 tables. Choose the appropriate data types and foreign key constraints.

(b) Create a sequence of insert statements that populates your database with "interesting data." "Interesting" here means a set of tuples that helps you later verify if the queries you are going to write over the database are indeed correct and show interesting results. You may want to iteratively populate the database as you progress with your queries.

(5) Querying the database

(a) Find the VINs of vehicles that are insured for a driver between 20 and 30 years of age (as of today).

(b) For each registered driver in the database, find the number of vehicles registered under the driver's license. Hint: if you have registered drivers who currently do not have a vehicle registered in their name (which we assume you do), then the query should return 0.

(c) Find the VINs of vehicles that are insured for some driver under 25 years of age and another driver who is over 50 years of age.

(d) Find drivers who have at least 3 vehicles registered under their name. Your query should return the name of the driver, and the number of registered cars in decreasing order

(e) Some vehicles are operated by more than one driver, and a different premium may be charged for each driver for the same vehicle. Find pairs of license numbers such that the driver with the first license number gets charged a higher premium for the same vehicle compared to the driver with the second license number.

YOUR OWN SOCIAL NETWORK DATABASE

Next you create a social network database about people and their relationships. The database has two relations:

Person (pid, name)
Relationship (pid1, pid2, rel)

The key fields are underlined: Person.pid is the key for Person, and (pid1, pid2) is the composite key for Relationship. You also want Relationship.pid1 and Relationship.pid2 to be foreign keys to Person. rel is a string representing the relationship type, and it can only be 'friend' or 'enemy'. So, a tuple (1,2, 'friend'), means that the person with id 1 considers the person with id 2 a friend. Note that

- the relationship is not symmetric: if Alice is friend with Bob, it does not imply that Bob is friend with Alice
- You want to enforce the constraint that only tuples with a value for rel of strings in {'enemy', 'friend'} can be inserted into the database.

(6) Create a database with appropriate datatypes, primary and foreign keys, and that enforces the domain constraint for the attribute rel. This is an example of a functionality that varies between different databases and we do not expect you to know *the exact syntax *by heart or for the exam. That's something you can also later easily look up. We recommend you search online for "CREATE DOMAIN" and "CHECK CONSTRAINT."

(7) Insert a minimum number of "interesting data" that allows you to verify that the queries you are typine are correct.

(8) Write a SQL query that returns pairs of people (A,B), such that A considers B a friend, and B considers A an enemy. Your query should return results containing pairs of names.

(9) Write a SQL query that returns triples of people (A,B,C), such that A considers B a friend, and B considers C a friend, but C considers A an enemy. Your query should return results containing triples of names.

DELIVERABLES

Put together one DOCX or TXT or SQL file that contains the answers to all questions above.

COLLABORATION POLICY

We randomly assign you to groups of 2 or 3 students. You will have to submit your individual homework on Blackboard, but we allow you to exchange ideas and brainstorm on solution ideas *within* the randomly assigned group of students. If you received help from one of your assigned team mates, then acknowledge the help of your team mate in your homework submission (e.g., "Alice's comments helped with when I was stuck on question 14". Any received help from within the group does not count against you. At the end of the semester, we ask everyone to rate the helpfulness if your team mates.