CHAPTER 3

# Introduction to SQL

## Practice Exercises

**3.1** Write the following queries in SQL, using the university schema. (We suggest you actually run these queries on a database, using the sample data that we provide on the web site of the book, db-book.com. Instructions for setting up a database, and loading sample data, are provided on the above web site.)

a. Find the titles of courses in the Comp. Sci. department that have 3 credits.

b. Find the IDs of all students who were taught by an instructor named Einstein; make sure there are no duplicates in the result.

c. Find the highest salary of any instructor.

d. Find all instructors earning the highest salary (there may be more than one with the same salary).

e. Find the enrollment of each section that was offered in Fall 2017.

f. Find the maximum enrollment, across all sections, in Fall 2017.

g. Find the sections that had the maximum enrollment in Fall 2017.

**3.2** Suppose you are given a relation *grade_points*($\underline{grade}$, *points*) that provides a conversion from letter grades in the *takes* relation to numeric scores; for example, an "A" grade could be specified to correspond to 4 points, an "A−" to 3.7 points, a "B+" to 3.3 points, a "B" to 3 points, and so on. The grade points earned by a student for a course offering (section) is defined as the number of credits for the course multiplied by the numeric points for the grade that the student received.

Given the preceding relation, and our university schema, write each of the following queries in SQL. You may assume for simplicity that no *takes* tuple has the *null* value for *grade*.

a. Find the total grade points earned by the student with ID '12345', across all courses taken by the student.

*person* (*<u>driver_id</u>*, *name*, *address*)
*car* (*<u>license_plate</u>*, *model*, *year*)
*accident* (*<u>report_number</u>*, *year*, *location*)
*owns* (*<u>driver_id</u>*, *<u>license_plate</u>*)
*participated* (*<u>report_number</u>*, *<u>license_plate</u>*, *driver_id*, *damage_amount*)

**Figure 3.17** Insurance database

b. Find the grade point average (*GPA*) for the above student, that is, the total grade points divided by the total credits for the associated courses.

c. Find the ID and the grade-point average of each student.

d. Now reconsider your answers to the earlier parts of this exercise under the assumption that some grades might be null. Explain whether your solutions still work and, if not, provide versions that handle nulls properly.

**3.3** Write the following inserts, deletes, or updates in SQL, using the university schema.

a. Increase the salary of each instructor in the Comp. Sci. department by 10%.

b. Delete all courses that have never been offered (i.e., do not occur in the *section* relation).

c. Insert every student whose *tot_cred* attribute is greater than 100 as an instructor in the same department, with a salary of $10,000.

**3.4** Consider the insurance database of Figure 3.17, where the primary keys are underlined. Construct the following SQL queries for this relational database.

a. Find the total number of people who owned cars that were involved in accidents in 2017.

b. Delete all year-2010 cars belonging to the person whose ID is '12345'.

**3.5** Suppose that we have a relation *marks*(*ID*, *score*) and we wish to assign grades to students based on the score as follows: grade $F$ if $score < 40$, grade $C$ if $40 \leq score < 60$, grade $B$ if $60 \leq score < 80$, and grade $A$ if $80 \leq score$. Write SQL queries to do the following:

a. Display the grade for each student, based on the *marks* relation.

b. Find the number of students with each grade.

*branch*(<u>*branch_name*</u>, *branch_city, assets*)
*customer* (<u>*ID*</u>, *customer_name*, *customer_street, customer_city*)
*loan* (<u>*loan_number*</u>, *branch_name, amount*)
*borrower* (<u>*ID*</u>, <u>*loan_number*</u>)
*account* (<u>*account_number*</u>, *branch_name, balance* )
*depositor* (<u>*ID*</u>, <u>*account_number*</u>)

**Figure 3.18** Banking database.

**3.6** The SQL **like** operator is case sensitive (in most systems), but the lower() function on strings can be used to perform case-insensitive matching. To show how, write a query that finds departments whose names contain the string “sci” as a substring, regardless of the case.

**3.7** Consider the SQL query

> **select** $p.a1$
> **from** $p$, $r1$, $r2$
> **where** $p.a1 = r1.a1$ **or** $p.a1 = r2.a1$

Under what conditions does the preceding query select values of $p.a1$ that are either in $r1$ or in $r2$? Examine carefully the cases where either $r1$ or $r2$ may be empty.

**3.8** Consider the bank database of Figure 3.18, where the primary keys are underlined. Construct the following SQL queries for this relational database.

a. Find the ID of each customer of the bank who has an account but not a loan.

b. Find the ID of each customer who lives on the same street and in the same city as customer '12345'.

c. Find the name of each branch that has at least one customer who has an account in the bank and who lives in “Harrison”.

**3.9** Consider the relational database of Figure 3.19, where the primary keys are underlined. Give an expression in SQL for each of the following queries.

a. Find the ID, name, and city of residence of each employee who works for “First Bank Corporation”.

b. Find the ID, name, and city of residence of each employee who works for “First Bank Corporation” and earns more than $10000.

c. Find the ID of each employee who does not work for “First Bank Corporation”.

d. Find the ID of each employee who earns more than every employee of “Small Bank Corporation”.

e. Assume that companies may be located in several cities. Find the name of each company that is located in every city in which “Small Bank Corporation” is located.

f. Find the name of the company that has the most employees (or companies, in the case where there is a tie for the most).

g. Find the name of each company whose employees earn a higher salary, on average, than the average salary at “First Bank Corporation”.

**3.10** Consider the relational database of Figure 3.19. Give an expression in SQL for each of the following:

a. Modify the database so that the employee whose ID is '12345' now lives in “Newtown”.

b. Give each manager of “First Bank Corporation” a 10 percent raise unless the salary becomes greater than $100000; in such cases, give only a 3 percent raise.

*employee* ($\underline{\textit{ID}}$, *person_name*, *street*, *city*)
*works* ($\underline{\textit{ID}}$, *company_name*, *salary*)
*company* ($\underline{\textit{company\_name}}$, *city*)
*manages* ($\underline{\textit{ID}}$, *manager_id*)

**Figure 3.19** Employee database.