## Programming Exercises

### 1. Population Database

VideoNote
Getting Started with the Population Database Problem

Make sure you have downloaded the book's source code from the companion website at www.pearson.com/gaddis. In this chapter's source code folder, you will find a program named `create_cities_db.py`. Run the program. The program will create a database named `cities.db`. The `cities.db` database will have a table named `Cities`, with the following columns:

| Column Name | Data Type |
|---|---|
| `CityID` | `INTEGER PRIMARY KEY` |
| `CityName` | `TEXT` |
| `Population` | `REAL` |

The `CityName` column stores the name of a city and the `Population` column stores the population of that city. After you run the `create_cities_db.py` program, the `Cities` table will contain 20 rows with various cities and their populations.

Next, write a program that connects to the `cities.db` database, and allows the user to select any of the following operations:

- Display a list of cities sorted by population, in ascending order.
- Display a list of cities sorted by population, in descending order.
- Display a list of cities sorted by name.
- Display the total population of all the cities.
- Display the average population of all the cities.
- Display the city with the highest population.
- Display the city with the lowest population.

### 2. Phone Book Database

Write a program that creates a database named `phonebook.db`. The database should have a table named `Entries`, with columns for a person's name and phone number. Next, write a CRUD application that lets the user add rows to the `Entries` table, look up a person's phone number, change a person's phone number, and delete specified rows.

### 3. Relational Database Project

In this assignment, you will create a database named `student_info.db` that holds the following information about students at a college:

- The student's name
- The student's major
- The department in which the student is enrolled

The database should have the following tables:

`Majors` table

| Column Name | Type |
|---|---|
| `MajorID` | `INTEGER PRIMARY KEY` |
| `Name` | `TEXT` |

`Departments` table

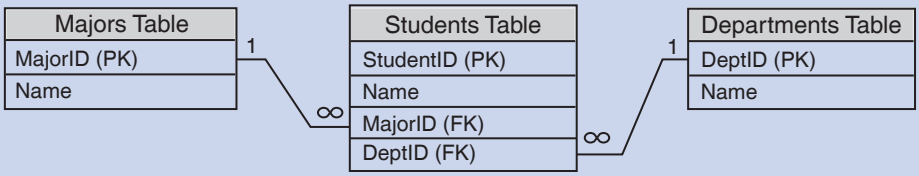
| Column Name | Type |
|---|---|
| `DeptID` | `INTEGER PRIMARY KEY` |
| `Name` | `TEXT` |

`Students` table

| Column Name | Type |
|---|---|
| `StudentID` | `INTEGER PRIMARY KEY` |
| `Name` | `TEXT` |
| `MajorID` | `INTEGER` (*Foreign key that references the* `MajorID` *column in the* `Majors` *table*) |
| `DeptID` | `INTEGER` (*Foreign key that references the* `DeptID` *column in the* `Departments` *table*) |

Figure 14-12 shows an entity relationship diagram for the database.

**Figure 14-12** Entity relationship diagram for the `student_info.db` database




- Write a program that creates the database and the tables.
- Write a program that performs CRUD operations on the `Majors` table. Specifically, the program should allow the user to do the following:

  ___ Add a new major
  ___ Search for an existing major
  ___ Update an existing major
  ___ Delete an existing major
  ___ Show a list of all majors

- Write a program that performs CRUD operations on the `Departments` table. Specifically, the program should allow the user to do the following:

  ___ Add a new department
  ___ Search for an existing department
  ___ Update an existing department
  ___ Delete an existing department
  ___ Show a list of all departments

- Write a program that performs CRUD operations on the `Students` table. Specifically, the program should allow the user to do the following:

  ___ Add a new student
  ___ Search for an existing student
  ___ Update an existing student
  ___ Delete an existing student
  ___ Show a list of all students

  When adding, updating, and deleting rows, be sure to enable foreign key enforcement. When adding a new student to the `Students` table, the user should only be allowed to select an existing major from the `Departments` table, and an existing department from the `Departments` table.