Analytics Lab: Use a Relational Database for University Data

Scenario

You are hired as an analytics consultant for Brideshead University. They've brought you on to lay the foundation for data-driven decision-making at the university.

Your task: Build a PostgreSQL database to organize Brideshead's student, instructor, and course data.

In this lab, you'll create a database, insert a few records using the provided SQL code files, and run queries to perform the given tasks. Upon completion, you will be able to:

- Create a database in PostgreSQL.
- Populate the database with data.
- Execute SQL queries to accomplish specific tasks.

The *Database System Concepts* textbook website is a useful further resource.

Code Files

These files are available on the course page, as well as the MSBA GitHub repository, in the Relational Databases course folder.

- ***university_ddl.sql***. This is the Data Definition Language (DDL) SQL code that creates a schema named university and the tables that are part of the schema.
    - Use the above code the first time.
    - If you would like to recreate the schema for some reason, then, to avoid errors, you must first drop the existing schema using the code below:

      ```
      DROP SCHEMA university CASCADE;
      ```

- ***university_data.sql***. This file contains SQL insert statements to load the sample data into all the tables, after first deleting any data the tables currently contain.

Discussion Questions

The provided DDL code file includes various constraints, such as primary keys, foreign keys, and check constraints, to ensure data integrity in the university's database design.

1. Why are these constraints crucial for maintaining accurate and reliable data?
2. What potential issues might arise if these constraints were not implemented correctly in the database design?

Technical Challenges

1. Create a PostgreSQL database using PgAdmin. If you installed PostgreSQL via Docker, ensure Docker is running before connecting to PgAdmin.
2. Run the code in the ***university_ddl.sql*** file in order to set up the database using PgAdmin.
3. To set the university schema as the default schema, set the search path using the code block below:

```
SET search_path TO university;
```

4. Run the code in the ***university_data.sql*** file in order to populate the database with the sample data.
5. Try out some queries, and briefly explain what they do and how many records you see in the results as comments for each query below:

```
SELECT * FROM instructor;
```

```
SELECT name FROM instructor
WHERE dept_name = 'Comp. Sci.' AND salary > 70000;
```

```
SELECT * FROM instructor, department
WHERE instructor.dept_name = department.dept_name;
```

6. Write a SQL query to find the names of all the instructors from the Biology department.

7. Write a SQL query to find the names of courses in the Computer Science department which have 3 credits.
8. For the student with ID 12345, write a SQL query to show the course_id and title of all courses registered for by the student.
9. Write a SQL query to display the IDs of all instructors who have never taught a course.
10. Write a SQL query to find the names of all students who have ever taken any Comp. Sci. course. (There should be no duplicate names.)
11. Write a SQL query to find the maximum and minimum enrollment across all sections. Consider only sections that had some enrollment; don't worry about sections with no enrolled students.
12. Write a SQL query to return the section(s) with the highest enrollment for all courses, and list the enrollment for those sections, using a subquery.
13. Grades are mapped to a grade point as follows: Grades are mapped to a grade point as follows: 'A+':10; A' : 10; 'A-':9; 'B+':8; 'B':7; 'B-':6; 'C+':5; 'C':4; 'C-':3; 'D+':2; 'D': 1; 'D-':0; 'F':0. Create a table to store these mappings, and use it to write a query to find the cumulative grade point average (GPA).
    a. GPA is calculated by averaging the grade points earned by a student in all courses. For example, if a student earns an A (10.0), a B+ (8.0), and a C- (3.0) in three different courses, the average would be (10.0 + 8.0 + 3.0) / 3= 7.0, resulting in a cumulative GPA of 7.0.
    b. To ensure that no student receives a GPA before they have received all their final grades, make sure that, for students who have a null grade in any course they have taken, the cumulative GPA is shown as null.

