

# Analytics Lab: Use a Relational Database for University Data

## Solutions to the Technical Challenges

*(Solutions to challenges 1-4 are not provided. For help setting up your database, reach out to your classmates and instructor.)*

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;
```

**12 records**

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

**2 records**

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

**12 records**

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

```
SELECT name  
FROM instructor  
WHERE dept_name = 'Biology';
```

7. Write a SQL query to find the names of courses in the Computer Science department which have 3 credits.

```
SELECT title
FROM course
WHERE dept_name = 'Comp. Sci.' AND credits = 3;
```

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.

```
SELECT t.course_id, c.title
FROM takes AS t
JOIN course AS c ON t.course_id = c.course_id
WHERE t.ID = '12345';
```

9. Write a SQL query to display the IDs of all instructors who have never taught a course.

```
SELECT ID
FROM instructor AS i
WHERE NOT EXISTS (
    SELECT *
    FROM teaches AS t
    WHERE i.ID = t.ID
);
```

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.)

```
SELECT DISTINCT s.name
FROM student AS s
JOIN takes AS t ON s.ID = t.ID
JOIN course AS c ON t.course_id = c.course_id
WHERE c.dept_name = 'Comp. Sci.';
```

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.

```
SELECT MAX(enrollment_count) AS max_enrollment,
MIN(enrollment_count) AS min_enrollment
FROM (
    SELECT course_id, sec_id, semester, year,
COUNT(*) AS enrollment_count
    FROM takes
    GROUP BY course_id, sec_id, semester, year
) AS section_enrollments;
```

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.

```
SELECT course_id, sec_id, semester, year,
enrollment_count
FROM (
    SELECT course_id, sec_id, semester, year,
COUNT(*) AS enrollment_count,
    RANK() OVER (PARTITION BY course_id
ORDER BY COUNT(*) DESC) AS rank
    FROM takes
    GROUP BY course_id, sec_id, semester, year
) AS section_enrollments
WHERE rank = 1;
```

13. 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).
- Cumulative 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.
  - 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.

```
--Create grade_points table
CREATE TABLE grade_points (
```

```
        grade CHAR(2) PRIMARY KEY,
        grade_point INTEGER
    );

-- Insert the grade mappings
INSERT INTO grade_points (grade, grade_point) VALUES
('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);

SELECT t.ID,
       CASE
           WHEN COUNT(t.grade) FILTER (WHERE t.grade
IS NULL) > 0 THEN NULL
           ELSE ROUND(AVG(gp.grade_point),2)
       END AS cumulative_gpa
FROM takes AS t
LEFT JOIN grade_points AS gp ON t.grade = gp.grade
GROUP BY t.ID;
```