# Case Study for SQL : University Management System

**Scenario:**

You are working with a university's database that stores information about students, courses, enrollments, instructors, and departments. Your task is to write SQL queries to retrieve specific information using joins and sub-queries.

## Database Schema:

1. **Departments** Table:
    - `department_id` (Primary Key)
    - `department_name`
    - `building`
2. **Instructors** Table:
    - `instructor_id` (Primary Key)
    - `instructor_name`
    - `department_id` (Foreign Key referencing `Departments.department_id`)
    - `salary`
3. **Courses** Table:
    - `course_id` (Primary Key)
    - `course_name`
    - `department_id` (Foreign Key referencing `Departments.department_id`)
    - `credits`
4. **Students** Table:
    - `student_id` (Primary Key)
    - `student_name`
    - `department_id` (Foreign Key referencing `Departments.department_id`)
    - `admission_year`
5. **Enrollments** Table:
    - `enrollment_id` (Primary Key)
    - `student_id` (Foreign Key referencing `Students.student_id`)
    - `course_id` (Foreign Key referencing `Courses.course_id`)
    - `enrollment_date`
    - `grade`

## Sample Data:

1. **Departments**:

```
INSERT INTO Departments (department_id, department_name, building) VALUES
(1, 'Computer Science', 'Engineering Building'),
(2, 'Mathematics', 'Science Building'),
(3, 'Physics', 'Science Building');
```

2. **Instructors**:

```
INSERT INTO Instructors (instructor_id, instructor_name, department_id,
salary) VALUES
(1, 'Dr. Smith', 1, 80000),
(2, 'Dr. Johnson', 2, 75000),
(3, 'Dr. Brown', 3, 85000);
```

3. **Courses**:

```
INSERT INTO Courses (course_id, course_name, department_id, credits) VALUES
(1, 'Introduction to Programming', 1, 3),
(2, 'Calculus I', 2, 4),
(3, 'Quantum Mechanics', 3, 3);
```

4. **Students**:

```
INSERT INTO Students (student_id, student_name, department_id,
admission_year) VALUES
(1, 'Alice Johnson', 1, 2020),
(2, 'Bob Smith', 2, 2019),
(3, 'Charlie Brown', 3, 2021);
```

5. **Enrollments**:

```
INSERT INTO Enrollments (enrollment_id, student_id, course_id,
enrollment_date, grade) VALUES
(1, 1, 1, '2023-01-15', 'A'),
(2, 2, 2, '2023-01-16', 'B'),
(3, 3, 3, '2023-01-17', 'A-');
```

---

## SQL Queries to Practice:

1. **Retrieve all courses along with their department names:**

```sql
SELECT c.course_name, d.department_name
FROM Courses c
JOIN Departments d ON c.department_id = d.department_id;
```

2. **Find the total number of students enrolled in each course:**

```sql
SELECT c.course_name, COUNT(e.student_id) AS total_students
FROM Courses c
LEFT JOIN Enrollments e ON c.course_id = e.course_id
GROUP BY c.course_name;
```

3. **List all students who have enrolled in courses from the 'Computer Science' department:**

```sql
SELECT s.student_name
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id
JOIN Departments d ON c.department_id = d.department_id
WHERE d.department_name = 'Computer Science';
```

4. **Find the average salary of instructors in each department:**

```sql
SELECT d.department_name, AVG(i.salary) AS average_salary
FROM Departments d
JOIN Instructors i ON d.department_id = i.department_id
GROUP BY d.department_name;
```

5. **Retrieve the names of students who have not enrolled in any course:**

```sql
SELECT s.student_name
FROM Students s
LEFT JOIN Enrollments e ON s.student_id = e.student_id
WHERE e.enrollment_id IS NULL;
```

6. **Find the course with the highest number of credits:**

```sql
SELECT course_name, credits
FROM Courses
WHERE credits = (SELECT MAX(credits) FROM Courses);
```

7. **List all instructors along with the courses they teach (if any):**

```sql
SELECT i.instructor_name, c.course_name
FROM Instructors i
LEFT JOIN Courses c ON i.department_id = c.department_id;
```

8. **Find the total number of students admitted each year:**

```sql
SELECT admission_year, COUNT(student_id) AS total_students
FROM Students
GROUP BY admission_year;
```

9. **Retrieve the names of students who scored an 'A' in any course:**

```sql
SELECT DISTINCT s.student_name
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
WHERE e.grade = 'A';
```

10. **Find the department with the most courses:**

```sql
SELECT d.department_name, COUNT(c.course_id) AS total_courses
FROM Departments d
JOIN Courses c ON d.department_id = c.department_id
GROUP BY d.department_name
ORDER BY total_courses DESC
LIMIT 1;
```

11. **List all courses that have no enrollments:**

```sql
SELECT c.course_name
FROM Courses c
LEFT JOIN Enrollments e ON c.course_id = e.course_id
WHERE e.enrollment_id IS NULL;
```

12. **Find the student with the highest number of enrollments:**

```sql
SELECT s.student_name, COUNT(e.enrollment_id) AS total_enrollments
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
GROUP BY s.student_name
ORDER BY total_enrollments DESC
LIMIT 1;
```

13. **Retrieve the names of students who enrolled in courses taught by 'Dr. Smith':**

```sql
SELECT DISTINCT s.student_name
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id
JOIN Instructors i ON c.department_id = i.department_id
WHERE i.instructor_name = 'Dr. Smith';
```

14. **Find the average grade of each student:**

```sql
SELECT s.student_name, AVG(e.grade) AS average_grade
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
GROUP BY s.student_name;
```

15. **List all departments and the number of instructors in each:**

```sql
SELECT d.department_name, COUNT(i.instructor_id) AS total_instructors
FROM Departments d
LEFT JOIN Instructors i ON d.department_id = i.department_id
GROUP BY d.department_name;
```

---

## Key Concepts Covered:

- **Joins**: Combining data from multiple tables (`INNER JOIN`, `LEFT JOIN`).
- **Sub-queries**: Using nested queries to filter or calculate data.
- **Aggregation**: Using `COUNT`, `AVG`, `MAX`, and `GROUP BY` to summarize data.
- **Filtering**: Using `WHERE`, `HAVING`, and conditional logic.