SQL case-based assignment using a University Database schema. This assignment will involve queries related to students, courses, departments, professors, and enrollments. I'll walk through the case, describe the database schema, and then provide 10 SQL queries related to university data analysis.

Database Schema

Student Table

```
CREATE DATABASE UNIVERSITY_RECORDS;


USE UNIVERSITY_RECORDS;
```

DEPARTMENTS TABLE

```
CREATE TABLE DEPARTMENTS (

DEPARTMENT_ID INT PRIMARY KEY,

DEPARTMENT_NAME VARCHAR(100) ) ;


INSERT INTO DEPARTMENTS (DEPARTMENT_ID,

DEPARTMENT_NAME) VALUES  (1, 'Computer Science'),

(2, 'Mechanical Engineering'),

(3, 'Electrical Engineering'),

(4, 'Civil Engineering');
```

PROFESSORE TABLE

```
CREATE TABLE PROFESSORS (
PROFESSOR_ID INT PRIMARY KEY,

FIRST_NAME VARCHAR(100),

LAST_NAME VARCHAR(100),

EMAIL VARCHAR(100),

PHONE VARCHAR(20) );


INSERT INTO PROFESSORS (PROFESSOR_ID, FIRST_NAME, LAST_NAME,

EMAIL, PHONE) VALUES  (201, 'Rajesh', 'Khanna', 'rajesh.khanna@university.edu',

'9123456780'),

(202, 'Sunita', 'Mehra', 'sunita.mehra@university.edu', '9123456781'),

(203, 'Amit', 'Singh', 'amit.singh@university.edu', '9123456782'),

(204, 'Neha', 'Joshi', 'neha.joshi@university.edu', '9123456783');
```

STUDENT TABLE

```
CREATE TABLE STUDENTS (

STUDENT_ID INT PRIMARY KEY,

FIRST_NAME VARCHAR(100),

LAST_NAME VARCHAR(100),

EMAIL VARCHAR(100),

PHONE VARCHAR(20),

DATE_OF_BIRTH DATE,

ENROLLMENT_DATE DATE,

DEPARTMENT_ID INT,

FOREIGN KEY (DEPARTMENT_ID) REFERENCES DEPARTMENTS(DEPARTMENT_ID) );
```

```
INSERT INTO STUDENTS (STUDENT_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE,
DATE_OF_BIRTH,  ENROLLMENT_DATE, DEPARTMENT_ID) VALUES
INSERT INTO STUDENTS (STUDENT_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE,

DATE_OF_BIRTH, ENROLLMENT_DATE, DEPARTMENT_ID) VALUES

(101, 'Ankit', 'Sharma', 'ankit.sharma@example.com', '9876543210', '2000-05-15',

'2020-08-01', 1),

(102, 'Priya', 'Verma', 'priya.verma@example.com', '9876543211', '2001-03-22',

'2021-08-01', 2),

(103, 'Ravi', 'Kumar', 'ravi.kumar@example.com', '9876543212', '1999-12-10', '2019-08-01',

1),

(104, 'Sneha', 'Mishra', 'sneha.mishra@example.com', '9876543213', '2000-11-30',

'2020-08-01', 3),

(105, 'Deepak', 'Yadav', 'deepak.yadav@example.com', '9876543214', '2002-01-20',

'2022-08-01', 4);
```

COURSES TABLE

```
CREATE TABLE COURSES (

COURSE_ID INT PRIMARY KEY,

COURSE_NAME VARCHAR(100),

DEPARTMENT_ID INT,

PROFESSOR_ID INT,

CREDITS INT,

FOREIGN KEY (DEPARTMENT_ID) REFERENCES
DEPARTMENTS(DEPARTMENT_ID),  FOREIGN KEY (PROFESSOR_ID)

REFERENCES PROFESSORS(PROFESSOR_ID) );


INSERT INTO COURSES (COURSE_ID, COURSE_NAME, DEPARTMENT_ID,
PROFESSOR_ID, CREDITS)  VALUES

(301, 'Data Structures', 1, 201, 4),

(302, 'Thermodynamics', 2, 202, 3),

(303, 'Digital Circuits', 3, 203, 4),

(304, 'Structural Analysis', 4, 204, 3),

(305, 'Operating Systems', 1, 201, 4);
```

ENROLLMENTS TABLE

```
CREATE TABLE ENROLLMENTS (
ENROLLMENT_ID INT PRIMARY KEY,

STUDENT_ID INT,

COURSE_ID INT,

ENROLLMENT_DATE DATE,

GRADE VARCHAR(5),

FOREIGN KEY (STUDENT_ID) REFERENCES STUDENTS(STUDENT_ID),

FOREIGN KEY (COURSE_ID) REFERENCES COURSES(COURSE_ID) );

INSERT INTO ENROLLMENTS (ENROLLMENT_ID, STUDENT_ID, COURSE_ID,
ENROLLMENT_DATE, GRADE) VALUES

(401, 101, 301, '2023-08-01', 'A'),

(402, 102, 302, '2023-08-01', 'B+'),

(403, 103, 301, '2023-08-01', 'A-'),

(404, 104, 303, '2023-08-01', 'B'),
(405, 105, 304, '2023-08-01', 'A'),

(406, 101, 305, '2023-08-01', 'A');
```

SQL Queries for the Case Study

1.Find the Total Number of Students in Each Department

```
SELECT

   d.DEPARTMENT_NAME,

   COUNT(s.STUDENT_ID) AS TOTAL_STUDENTS

FROM

   DEPARTMENTS d

LEFT JOIN

   STUDENTS s ON d.DEPARTMENT_ID = s.DEPARTMENT_ID

GROUP BY

   d.DEPARTMENT_NAME;
```

2.List All Courses Taught by a Specific Professor

```
SELECT

   c.COURSE_ID,

   c.COURSE_NAME,

   d.DEPARTMENT_NAME,

   c.CREDITS

FROM

   COURSES c

JOIN
```

```
   PROFESSORS p ON c.PROFESSOR_ID = p.PROFESSOR_ID
JOIN
   DEPARTMENTS d ON c.DEPARTMENT_ID = d.DEPARTMENT_ID
WHERE
   p.FIRST_NAME = 'Rajesh' AND p.LAST_NAME = 'Khanna';
```

3. Find the Average Grade of Students in Each Course

```
SELECT
   c.COURSE_ID,
   c.COURSE_NAME,
   d.DEPARTMENT_NAME,
   c.CREDITS
FROM
   COURSES c
JOIN
   PROFESSORS p ON c.PROFESSOR_ID = p.PROFESSOR_ID
JOIN
   DEPARTMENTS d ON c.DEPARTMENT_ID = d.DEPARTMENT_ID
WHERE
   p.FIRST_NAME = 'Rajesh' AND p.LAST_NAME = 'Khanna';
```

4. List All Students Who Have Not Enrolled in Any Course

```
SELECT
  c.COURSE_NAME,
  ROUND(AVG(
    CASE e.GRADE
      WHEN 'A' THEN 4.0
      WHEN 'A-' THEN 3.7
      WHEN 'B+' THEN 3.3
      WHEN 'B' THEN 3.0
      WHEN 'B-' THEN 2.7
      WHEN 'C+' THEN 2.3
      WHEN 'C' THEN 2.0
      WHEN 'D' THEN 1.0
      WHEN 'F' THEN 0.0
      ELSE NULL
    END
  ), 2) AS AVERAGE_GRADE_POINT
FROM
  ENROLLMENTS e
JOIN
  COURSES c ON e.COURSE_ID = c.COURSE_ID
GROUP BY
  c.COURSE_NAME;
```

5. Find the Number of Courses Offered by Each Department

```
SELECT
```

```
    d.DEPARTMENT_NAME,

    COUNT(c.COURSE_ID) AS TOTAL_COURSES

FROM

    DEPARTMENTS d

LEFT JOIN

    COURSES c ON d.DEPARTMENT_ID = c.DEPARTMENT_ID

GROUP BY

    d.DEPARTMENT_NAME;
```

6. List All Students Who Have Taken a Specific Course (e.g., 'Database Systems')

```
SELECT

    s.STUDENT_ID,

    s.FIRST_NAME,

    s.LAST_NAME,

    s.EMAIL,

    c.COURSE_NAME

FROM

    ENROLLMENTS e

JOIN

    STUDENTS s ON e.STUDENT_ID = s.STUDENT_ID

JOIN

    COURSES c ON e.COURSE_ID = c.COURSE_ID

WHERE

    c.COURSE_NAME = 'Database Systems';
```

7.Find the Most Popular Course Based on Enrollment Numbers

```
SELECT

   c.COURSE_NAME,

   COUNT(e.STUDENT_ID) AS ENROLLMENT_COUNT

FROM

   ENROLLMENTS e

JOIN

   COURSES c ON e.COURSE_ID = c.COURSE_ID

GROUP BY

   c.COURSE_NAME

ORDER BY

   ENROLLMENT_COUNT DESC

LIMIT 1;
```

8.Find the Average Number of Credits Per Student in a Department

```
SELECT

   d.DEPARTMENT_NAME,

   ROUND(AVG(student_total_credits), 2) AS AVG_CREDITS_PER_STUDENT

FROM (

   SELECT

      s.STUDENT_ID,

      s.DEPARTMENT_ID,

      SUM(c.CREDITS) AS student_total_credits

   FROM

      STUDENTS s
```

```
    JOIN
        ENROLLMENTS e ON s.STUDENT_ID = e.STUDENT_ID
    JOIN
        COURSES c ON e.COURSE_ID = c.COURSE_ID
    GROUP BY
        s.STUDENT_ID, s.DEPARTMENT_ID
) AS student_credit_summary
JOIN
    DEPARTMENTS d ON student_credit_summary.DEPARTMENT_ID = d.DEPARTMENT_ID
GROUP BY
    d.DEPARTMENT_NAME;
```

9. List All Professors Who Teach in More Than One Department

```
SELECT
    p.PROFESSOR_ID,
    p.FIRST_NAME,
    p.LAST_NAME,
    COUNT(DISTINCT c.DEPARTMENT_ID) AS DEPARTMENTS_TAUGHT
FROM
    PROFESSORS p
JOIN
    COURSES c ON p.PROFESSOR_ID = c.PROFESSOR_ID
GROUP BY
    p.PROFESSOR_ID, p.FIRST_NAME, p.LAST_NAME
```

```
HAVING

    COUNT(DISTINCT c.DEPARTMENT_ID) > 1;
```

10.Get the Highest and Lowest Grade in a Specific Course (e.g., 'Operating Systems')

```
SELECT

    c.COURSE_NAME,

    MAX(

        CASE e.GRADE

            WHEN 'A' THEN 4.0

            WHEN 'A-' THEN 3.7

            WHEN 'B+' THEN 3.3

            WHEN 'B' THEN 3.0

            WHEN 'B-' THEN 2.7

            WHEN 'C+' THEN 2.3

            WHEN 'C' THEN 2.0

            WHEN 'D' THEN 1.0

            WHEN 'F' THEN 0.0

            ELSE NULL

        END

    ) AS HIGHEST_GRADE_POINT,

    MIN(

        CASE e.GRADE

            WHEN 'A' THEN 4.0

            WHEN 'A-' THEN 3.7

            WHEN 'B+' THEN 3.3
            WHEN 'B' THEN 3.0

            WHEN 'B-' THEN 2.7

            WHEN 'C+' THEN 2.3

            WHEN 'C' THEN 2.0

            WHEN 'D' THEN 1.0

            WHEN 'F' THEN 0.0

            ELSE NULL

        END

    ) AS LOWEST_GRADE_POINT

FROM

    ENROLLMENTS e

JOIN

    COURSES c ON e.COURSE_ID = c.COURSE_ID

WHERE

    c.COURSE_NAME = 'Operating Systems';
```