Final Project SQL

**Database:**

CREATE DATABASE final\_project;

**Tables and values:**

-- -----------------------------------------------------

-- Table 1: Departments

-- -----------------------------------------------------

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY AUTO\_INCREMENT,

DepartmentName VARCHAR(100) NOT NULL

);

INSERT INTO Departments (DepartmentName)

VALUES

('Computer Science'),

('Mathematics');

-- -----------------------------------------------------

-- Table 2: Students

-- -----------------------------------------------------

CREATE TABLE Students (

StudentID INT PRIMARY KEY AUTO\_INCREMENT,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50) NOT NULL,

Email VARCHAR(100) UNIQUE,

BirthDate DATE,

EnrollmentDate DATE

);

-- Sample Data

INSERT INTO Students (FirstName, LastName, Email, BirthDate, EnrollmentDate)

VALUES

('John', 'Doe', 'john.doe@univ.com', '2002-03-15', '2021-07-20'),

('Emma', 'Smith', 'emma.smith@univ.com', '2001-06-10', '2021-07-25'),

('Liam', 'Brown', 'liam.brown@univ.com', '2003-02-05', '2022-08-01');

-- -----------------------------------------------------

-- Table 3: Courses

-- -----------------------------------------------------

CREATE TABLE Courses (

CourseID INT PRIMARY KEY AUTO\_INCREMENT,

CourseName VARCHAR(100) NOT NULL,

DepartmentID INT,

Credits INT,

FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)

);

-- Sample Data

INSERT INTO Courses (CourseName, DepartmentID, Credits)

VALUES

('Introduction to SQL', 1, 3),

('Data Structures', 2, 4);

-- -----------------------------------------------------

-- Table 4: Instructors

-- -----------------------------------------------------

CREATE TABLE Instructors (

InstructorID INT PRIMARY KEY AUTO\_INCREMENT,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50) NOT NULL,

Email VARCHAR(100),

DepartmentID INT,

FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)

);

-- Sample Data

INSERT INTO Instructors (FirstName, LastName, Email, DepartmentID)

VALUES

('Alice', 'Johnson', 'alice.johnson@univ.com', 1),

('Bob', 'Lee', 'bob.lee@univ.com', 2);

-- -----------------------------------------------------

-- Table 5: Enrollments

-- -----------------------------------------------------

CREATE TABLE Enrollments (

EnrollmentID INT PRIMARY KEY AUTO\_INCREMENT,

StudentID INT,

CourseID INT,

EnrollmentDate DATE,

FOREIGN KEY (StudentID) REFERENCES Students(StudentID),

FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)

);

-- Sample Data

INSERT INTO Enrollments (StudentID, CourseID, EnrollmentDate)

VALUES

(1, 1, '2022-08-01'),

(2, 2, '2021-08-01');

INSERT INTO Departments (DepartmentName)

VALUES

('Physics'),

('Chemistry'),

('Economics');

INSERT INTO Students (FirstName, LastName, Email, BirthDate, EnrollmentDate)

VALUES

('Olivia', 'White', 'olivia.white@univ.com', '2002-05-12', '2022-07-18'),

('Noah', 'Wilson', 'noah.wilson@univ.com', '2001-11-03', '2022-08-05'),

('Sophia', 'Taylor', 'sophia.taylor@univ.com', '2003-01-24', '2023-01-10'),

('James', 'Anderson', 'james.anderson@univ.com', '2002-09-15', '2023-07-15'),

('Ava', 'Martin', 'ava.martin@univ.com', '2003-06-08', '2023-07-30');

INSERT INTO Courses (CourseName, DepartmentID, Credits)

VALUES

('Database Management Systems', 1, 4),

('Linear Algebra', 2, 3),

('Quantum Mechanics', 3, 4),

('Organic Chemistry', 4, 3),

('Microeconomics', 5, 3);

INSERT INTO Instructors (FirstName, LastName, Email, DepartmentID)

VALUES

('Clara', 'Davis', 'clara.davis@univ.com', 1),

('David', 'Miller', 'david.miller@univ.com', 2),

('Evelyn', 'Brown', 'evelyn.brown@univ.com', 3),

('Frank', 'Wilson', 'frank.wilson@univ.com', 4),

('Grace', 'Clark', 'grace.clark@univ.com', 5);

INSERT INTO Enrollments (StudentID, CourseID, EnrollmentDate)

VALUES

(3, 3, '2023-02-01'),

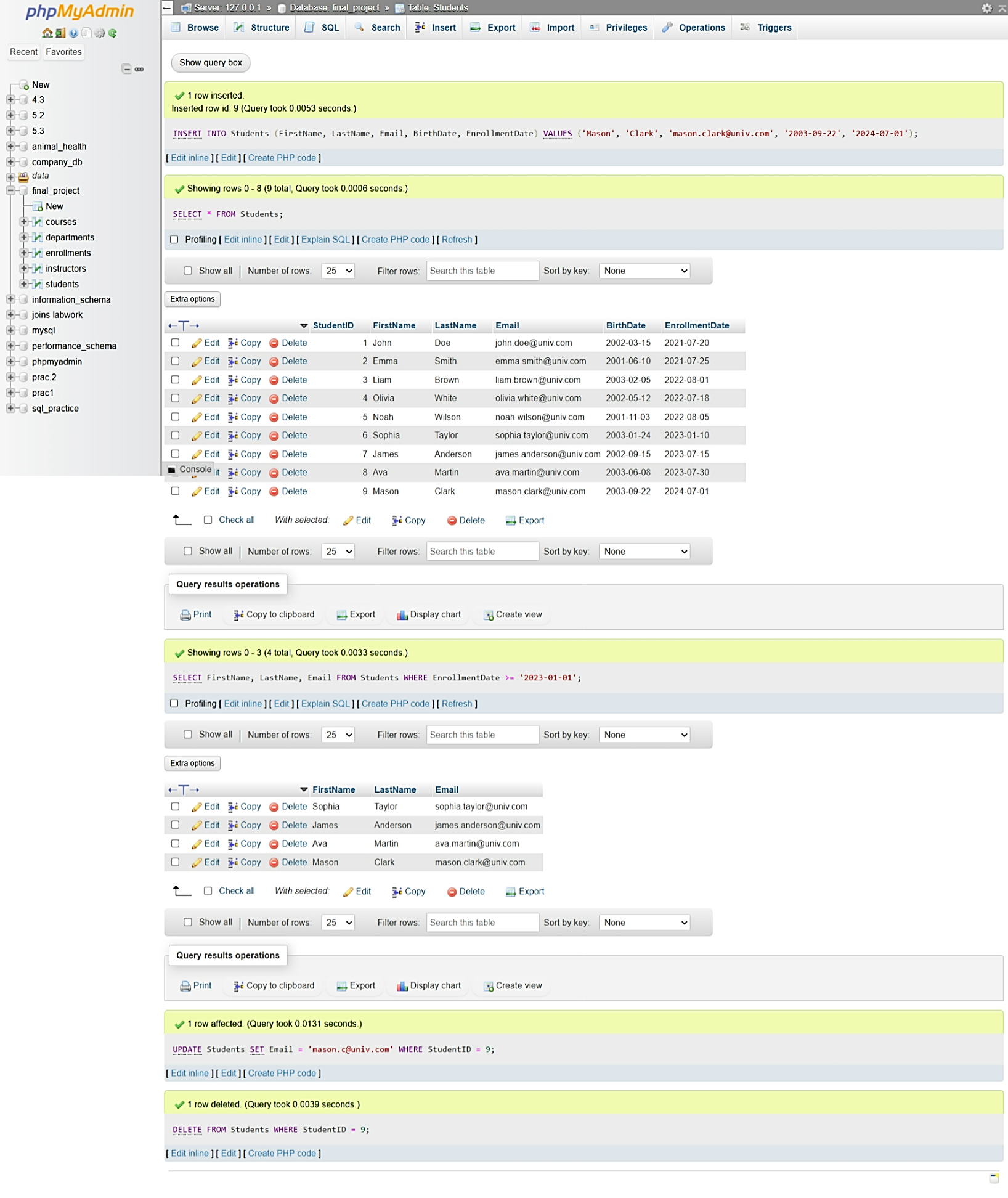
(4, 4, '2023-02-10'),

(5, 5, '2023-07-25'),

(6, 6, '2023-08-01'),

(7, 7, '2023-08-10');

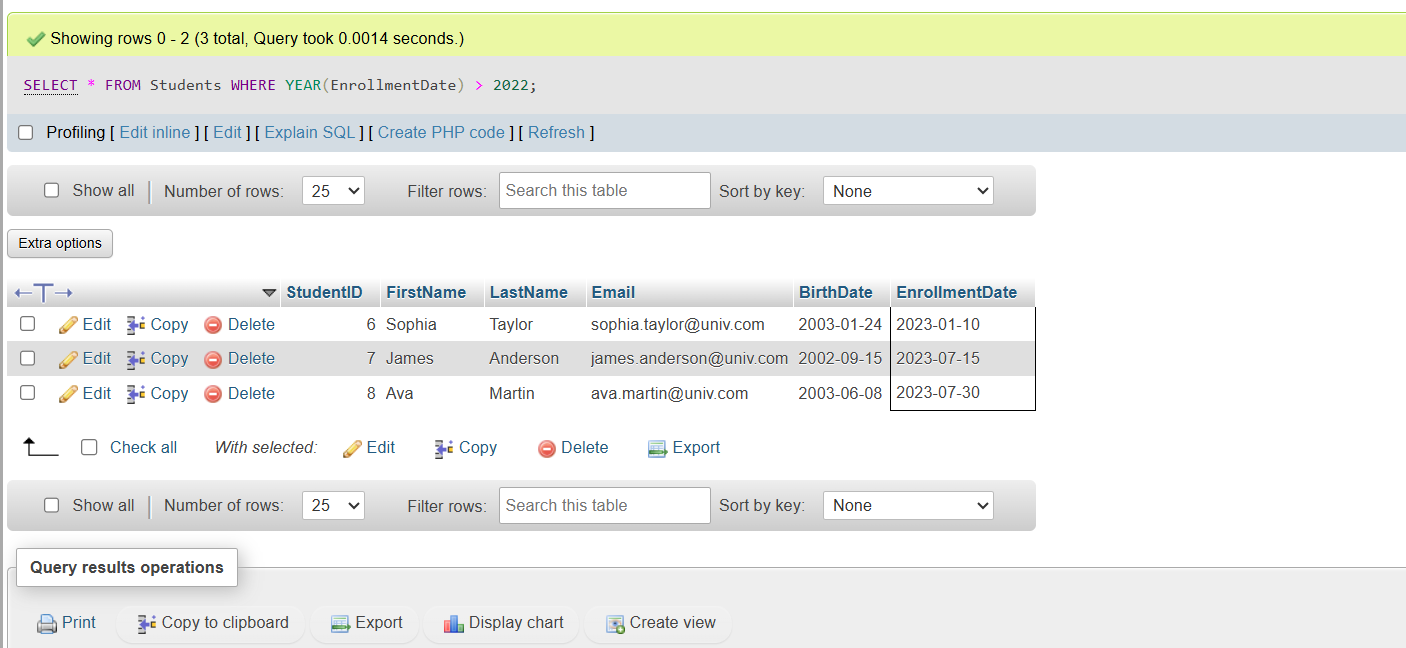
Tasks

1. Perform CURD operation.
2. Retrieve students who enrolled after 2022.

* SELECT \*

FROM Students

WHERE YEAR(EnrollmentDate) > 2022;



1. Retrieve courses offered by the Mathematics department with a limit of 5 courses.

* SELECT

c.CourseID,

c.CourseName,

d.DepartmentName,

c.Credits

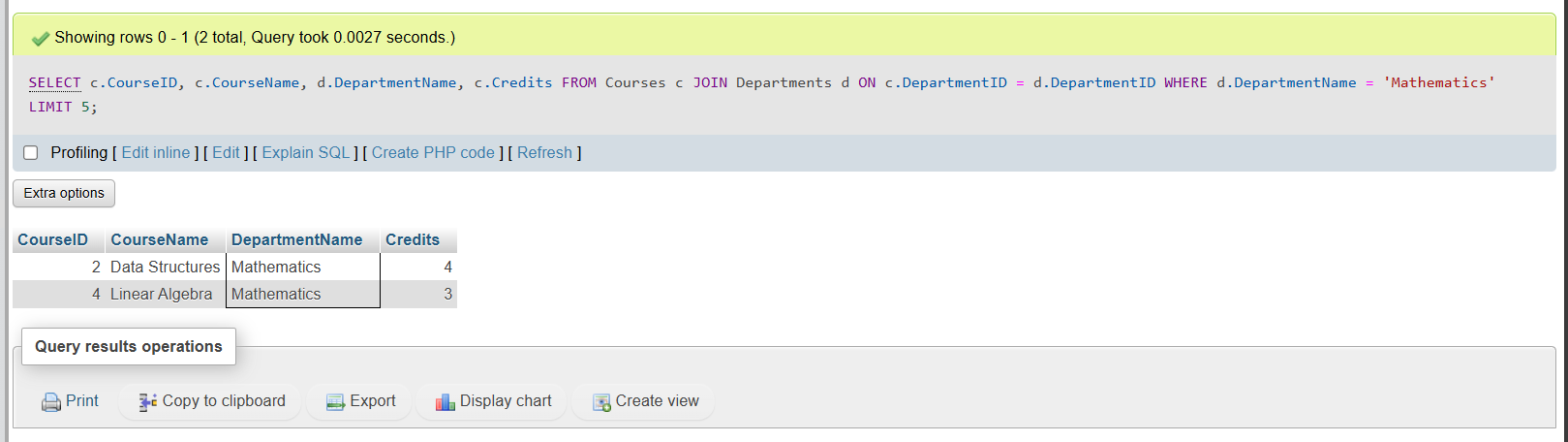
FROM Courses c

JOIN Departments d

ON c.DepartmentID = d.DepartmentID

WHERE d.DepartmentName = 'Mathematics'

LIMIT 5;



1. Get the number of students enrolled in each course, filtering for courses with more than 5 students.

* SELECT

c.CourseID,

c.CourseName,

COUNT(e.StudentID) AS Total\_Students

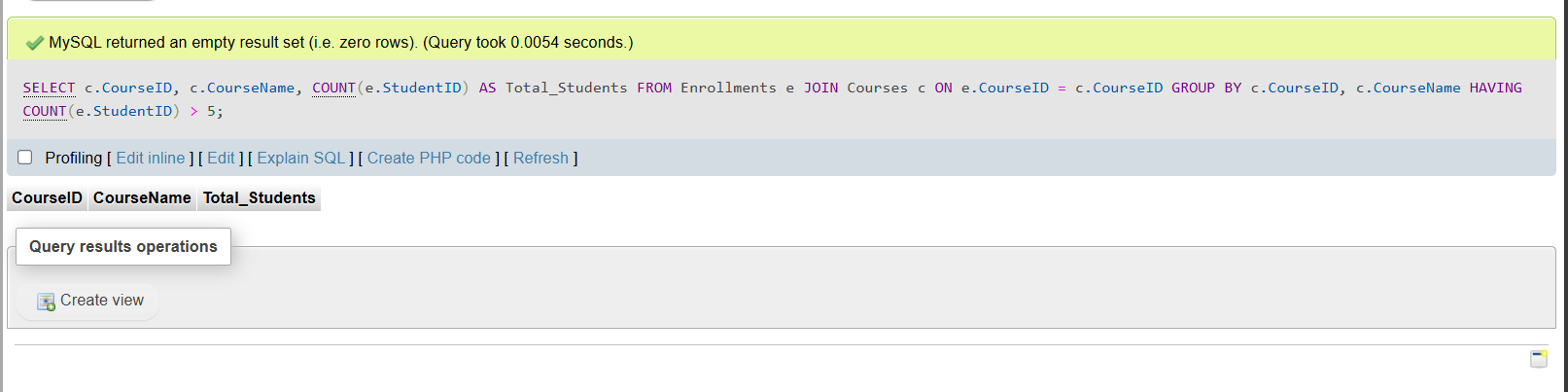
FROM Enrollments e

JOIN Courses c

ON e.CourseID = c.CourseID

GROUP BY c.CourseID, c.CourseName

HAVING COUNT(e.StudentID) > 5;



1. Find students who are enrolled in both Introduction to SQL and Data Structures.

* SELECT

s.StudentID,

s.FirstName,

s.LastName,

s.Email

FROM Students s

WHERE s.StudentID IN (

SELECT e1.StudentID

FROM Enrollments e1

JOIN Courses c1 ON e1.CourseID = c1.CourseID

WHERE c1.CourseName = 'Introduction to SQL'

)

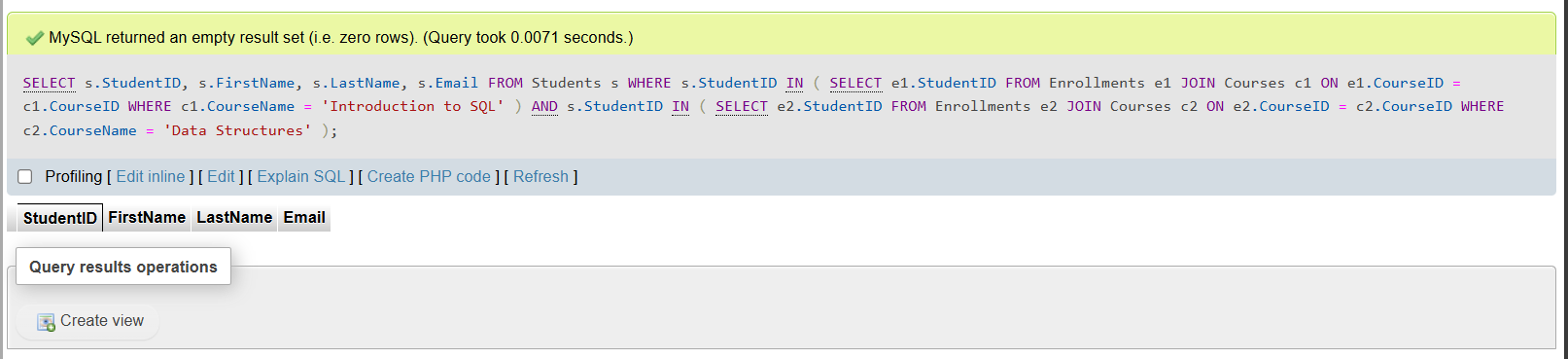
AND s.StudentID IN (

SELECT e2.StudentID

FROM Enrollments e2

JOIN Courses c2 ON e2.CourseID = c2.CourseID

WHERE c2.CourseName = 'Data Structures');



1. Find students who are either enrolled in Introduction to SQL or Data Structures.

* SELECT

s.StudentID,

s.FirstName,

s.LastName,

s.Email

FROM Students s

WHERE s.StudentID IN (

SELECT e1.StudentID

FROM Enrollments e1

JOIN Courses c1 ON e1.CourseID = c1.CourseID

WHERE c1.CourseName = 'Introduction to SQL'

)

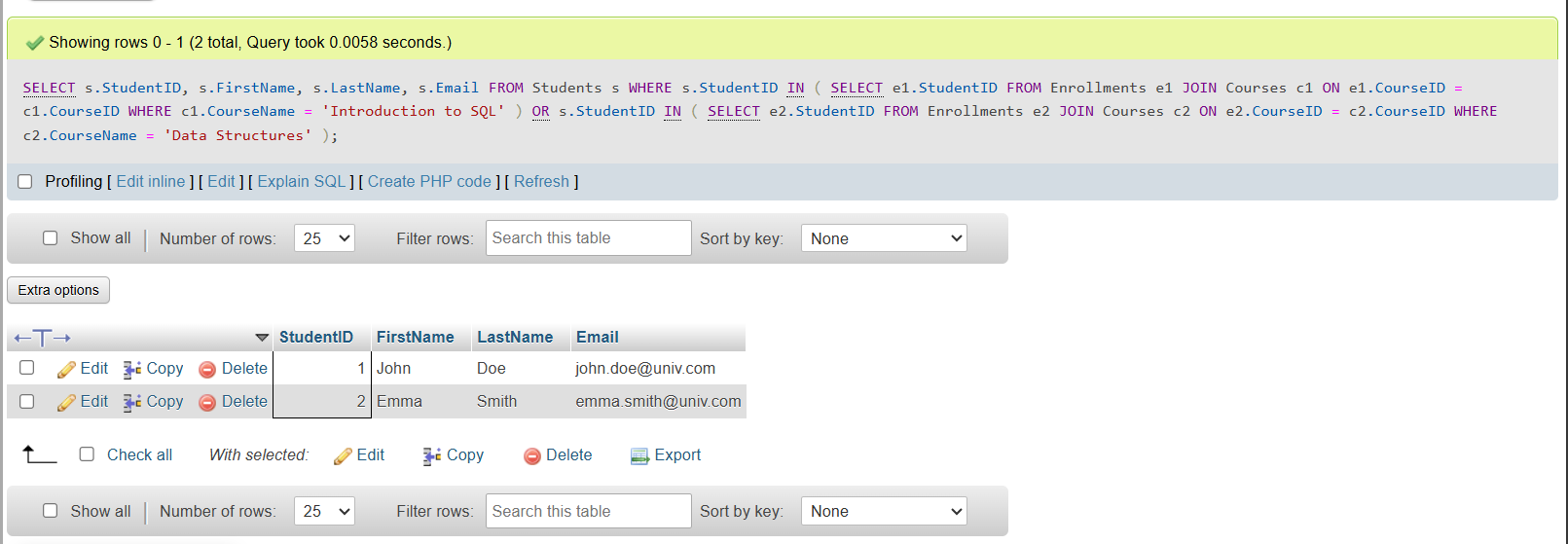
OR s.StudentID IN (

SELECT e2.StudentID

FROM Enrollments e2

JOIN Courses c2 ON e2.CourseID = c2.CourseID

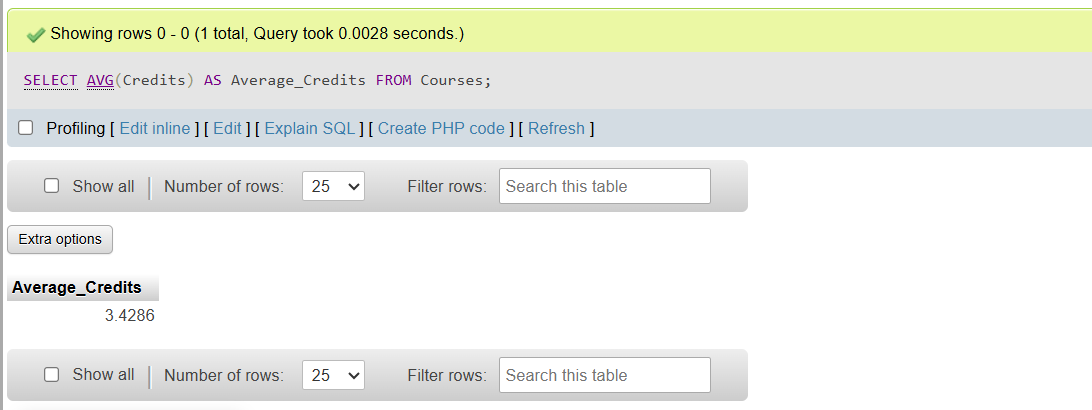
WHERE c2.CourseName = 'Data Structures');



1. Calculate the average number of credits for all courses.

* SELECT AVG(Credits) AS Average\_Credits

FROM Courses;



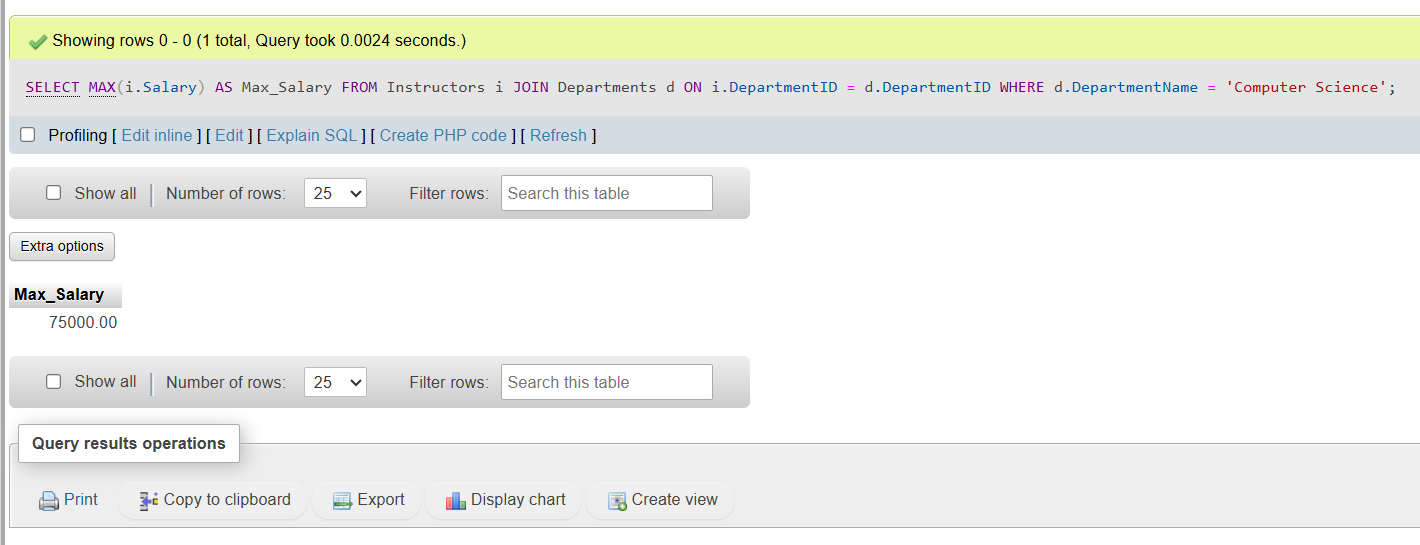
1. Find the maximum salary of instructors in the Computer Science department.

* SELECT MAX(i.Salary) AS Max\_Salary

FROM Instructors i

JOIN Departments d ON i.DepartmentID = d.DepartmentID

WHERE d.DepartmentName = 'Computer Science';



1. Count the number of students enrolled in each department.

* SELECT

d.DepartmentName,

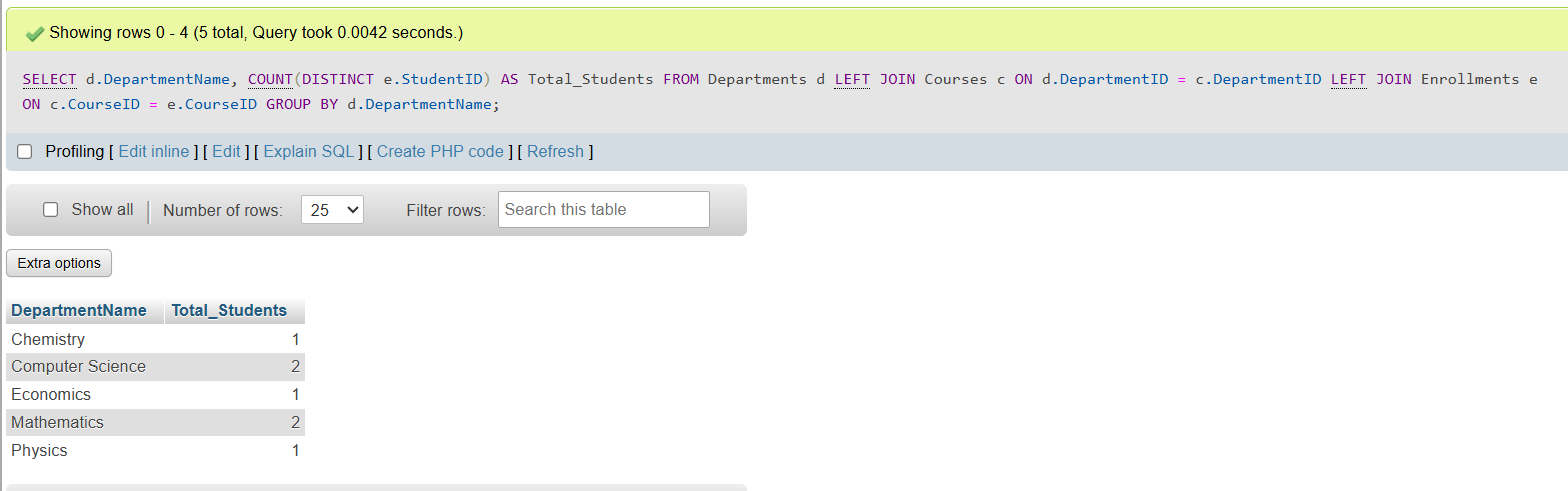
COUNT(DISTINCT e.StudentID) AS Total\_Students

FROM Departments d

LEFT JOIN Courses c ON d.DepartmentID = c.DepartmentID

LEFT JOIN Enrollments e ON c.CourseID = e.CourseID

GROUP BY d.DepartmentName;



1. INNER JOIN: Retrieve students and their corresponding courses.

* SELECT

s.StudentID,

s.FirstName,

s.LastName,

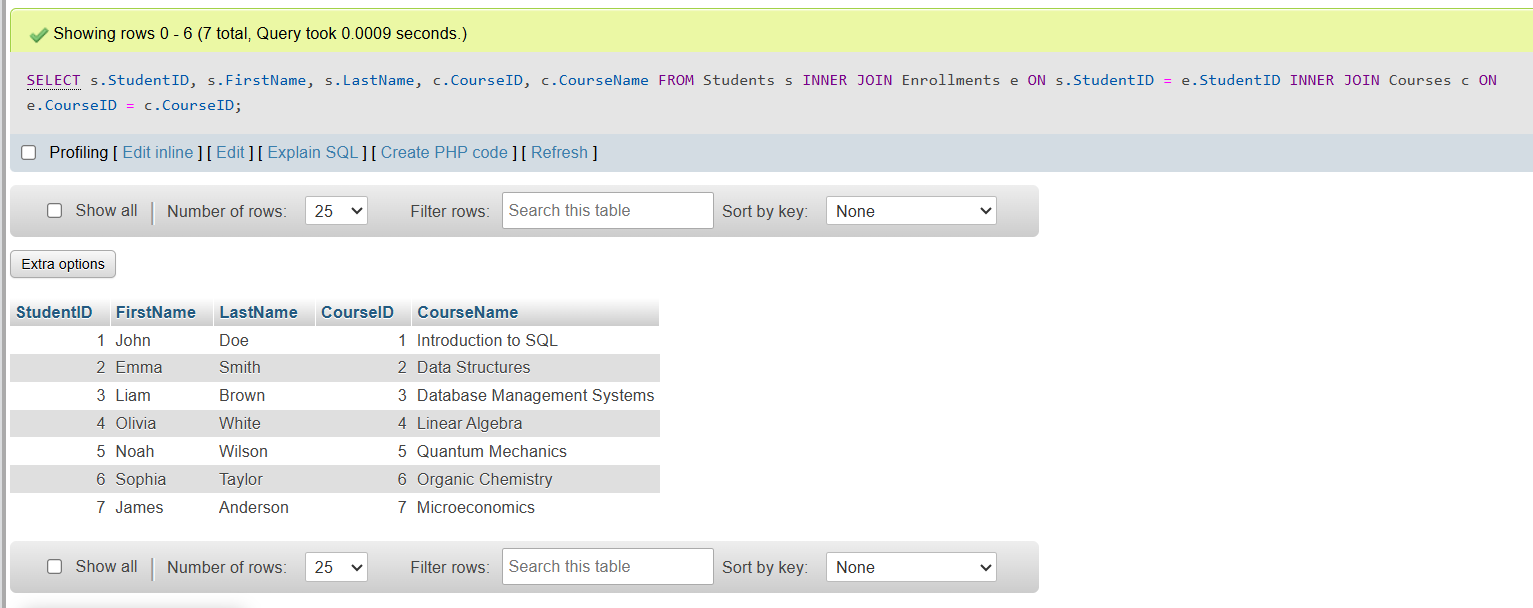
c.CourseID,

c.CourseName

FROM Students s

INNER JOIN Enrollments e ON s.StudentID = e.StudentID

INNER JOIN Courses c ON e.CourseID = c.CourseID;



1. LEFT JOIN: Retrieve all students and their corresponding courses, if any.

* SELECT

s.StudentID,

s.FirstName,

s.LastName,

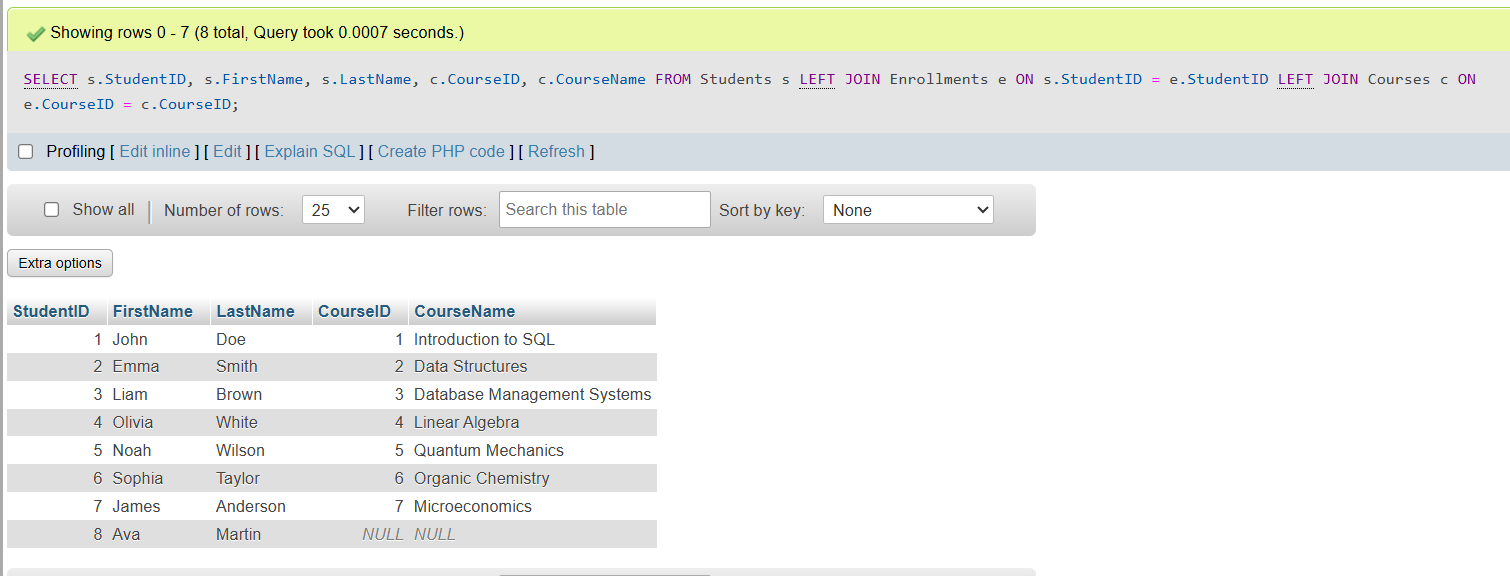
c.CourseID,

c.CourseName

FROM Students s

LEFT JOIN Enrollments e ON s.StudentID = e.StudentID

LEFT JOIN Courses c ON e.CourseID = c.CourseID;



1. Subquery: Find students enrolled in courses that have more than 10 students.

* SELECT

s.StudentID,

s.FirstName,

s.LastName,

c.CourseName

FROM Students s

JOIN Enrollments e ON s.StudentID = e.StudentID

JOIN Courses c ON e.CourseID = c.CourseID

WHERE c.CourseID IN (

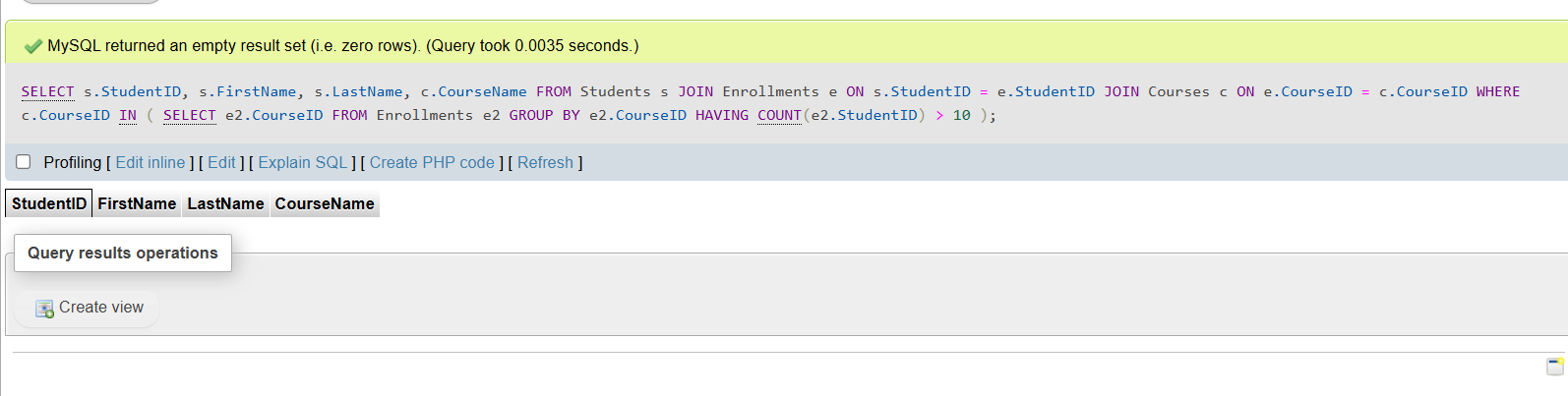
SELECT e2.CourseID

FROM Enrollments e2

GROUP BY e2.CourseID

HAVING COUNT(e2.StudentID) > 10

);



1. Extract the year from the EnrollmentDate of students.

* SELECT

StudentID,

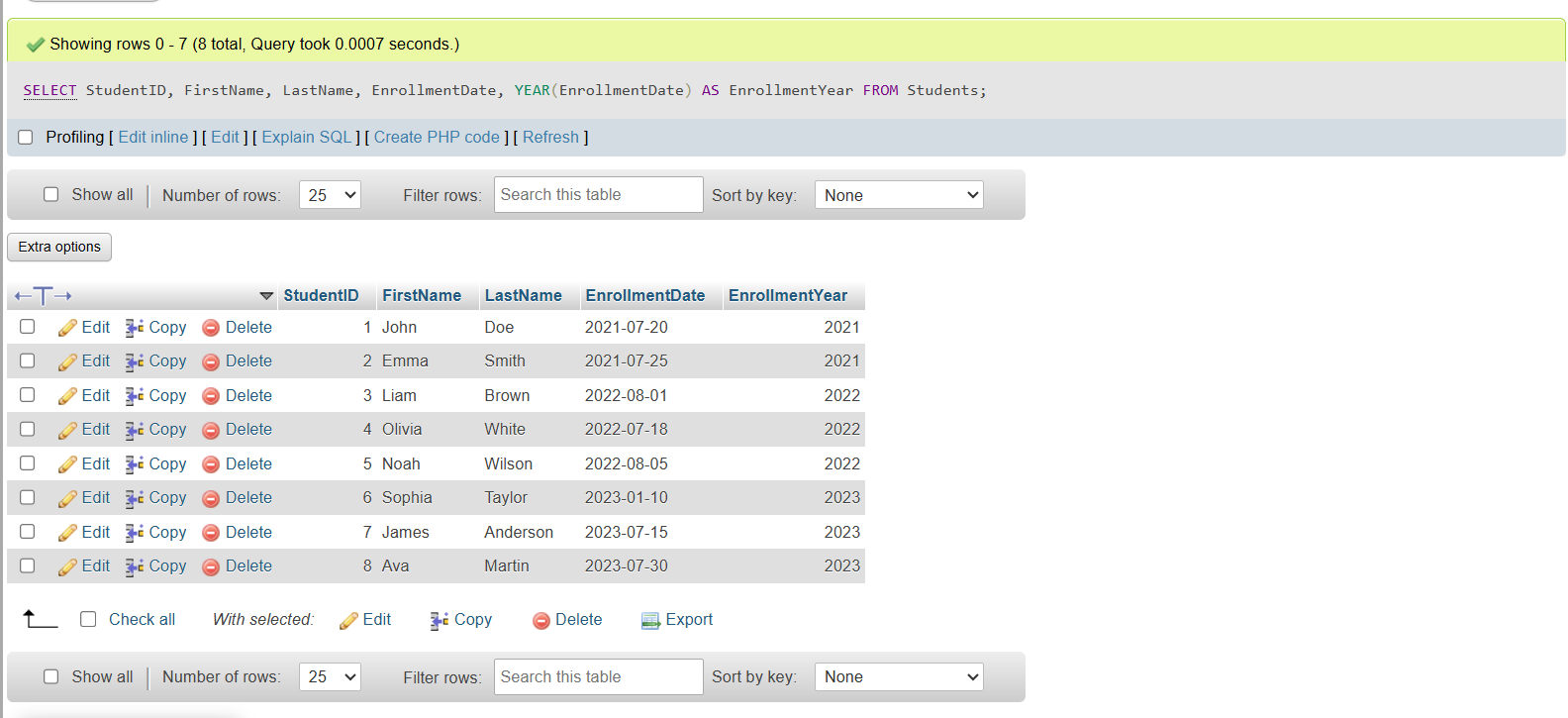
FirstName,

LastName,

EnrollmentDate,

YEAR(EnrollmentDate) AS EnrollmentYear

FROM Students;



1. Concatenate the instructor's first and last name.

* SELECT

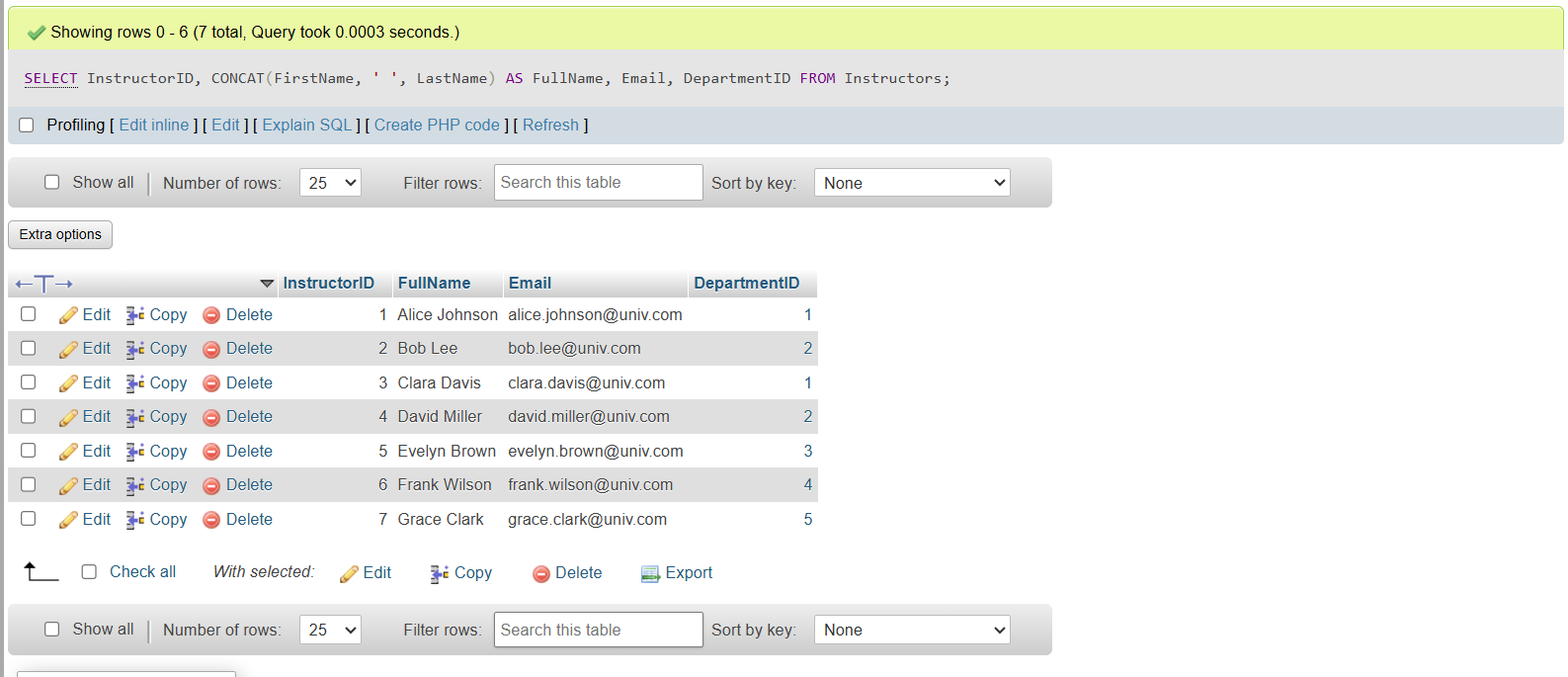
InstructorID,

CONCAT(FirstName, ' ', LastName) AS FullName,

Email,

DepartmentID

FROM Instructors;



1. Calculate the running total of students enrolled in courses.

* SELECT

c.CourseID,

c.CourseName,

COUNT(e.StudentID) AS EnrolledStudents,

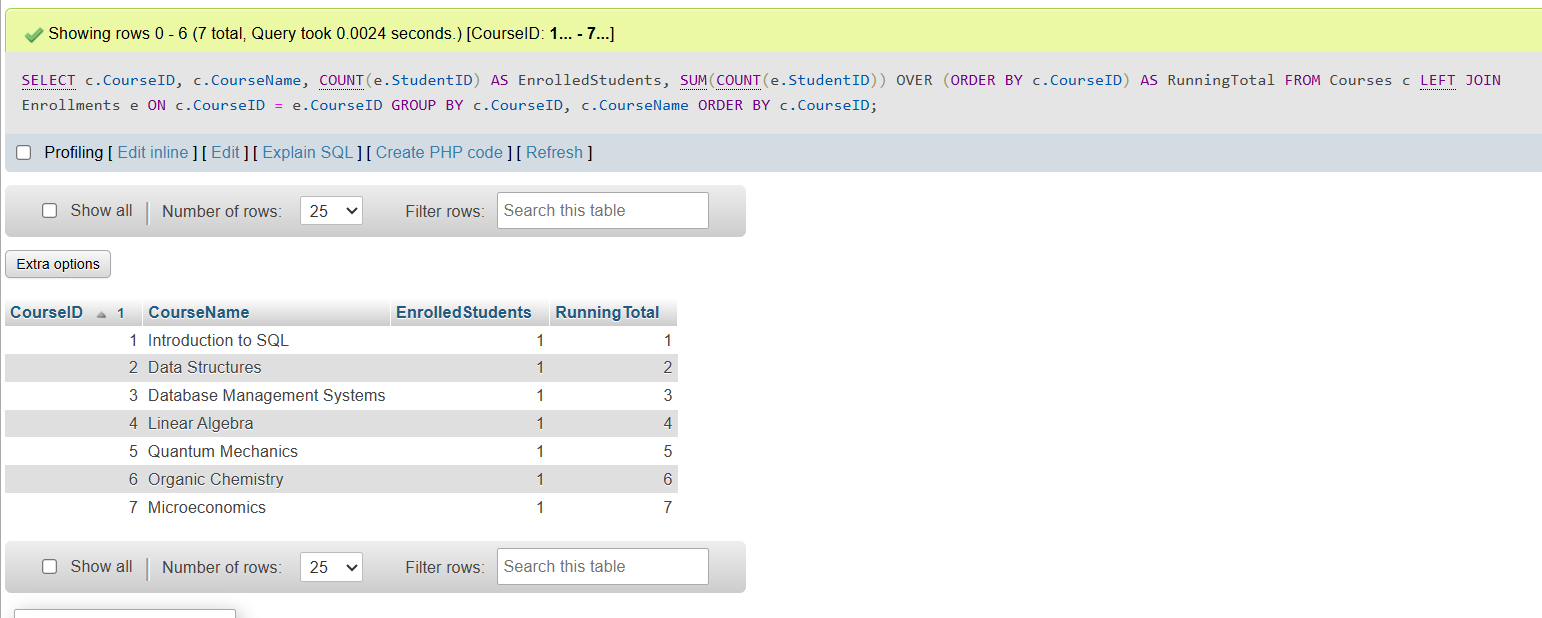
SUM(COUNT(e.StudentID)) OVER (ORDER BY c.CourseID) AS RunningTotal

FROM Courses c

LEFT JOIN Enrollments e ON c.CourseID = e.CourseID

GROUP BY c.CourseID, c.CourseName

ORDER BY c.CourseID;



1. Label students as 'Senior' or 'Junior' based on their year of enrollment. (If the enrollment date is more than 4 years from the currendate, put the label 'Senior' otherwise 'Junior' )

* SELECT

StudentID,

FirstName,

LastName,

EnrollmentDate,

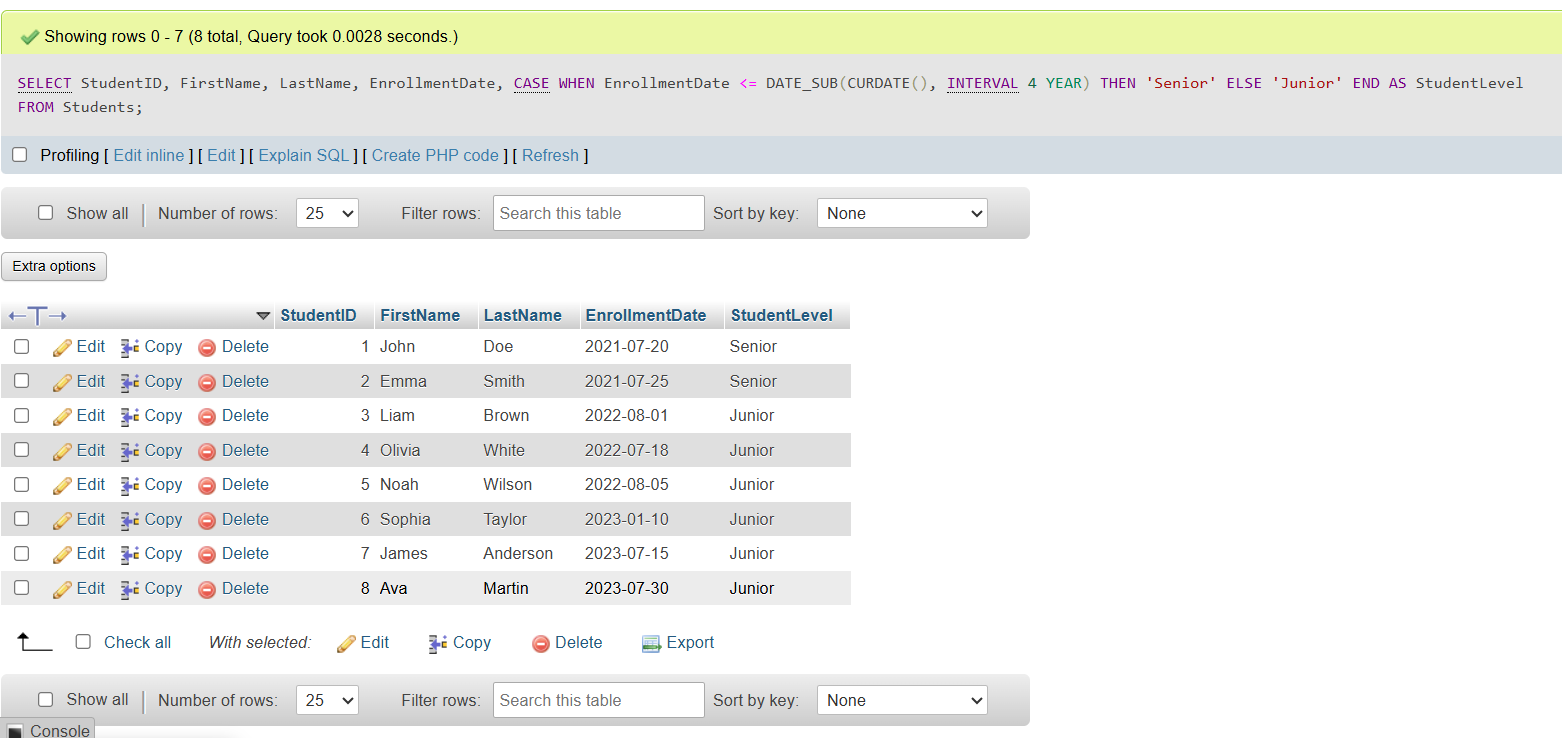
CASE

WHEN EnrollmentDate <= DATE\_SUB(CURDATE(), INTERVAL 4 YEAR) THEN 'Senior'

ELSE 'Junior'

END AS StudentLevel

FROM Students;



Thank You

Meet Patel.