Name: Manju

Date: 07/05/2025

Batch: May 25th Batch

This document includes the list of SQL queries used for database creation, data insertion, and task completion as part of the assigned Week 2 task.

Each query is accompanied by a brief explanation of its purpose, along with the corresponding output screenshots.

**Table creation queries:**

**Create table Courses**

CREATE TABLE Courses (courseid INT PRIMARY KEY AUTO\_INCREMENT, coursename VARCHAR(100), coursedescription TEXT);

**Purpose:** To create the Courses table which stores course details like ID, name, and description.

**Create table Courses**

CREATE TABLE Enrolments (enrolmentid INT PRIMARY KEY AUTO\_INCREMENT, studentid INT, courseid INT, enrolmentdate DATE, FOREIGN KEY (studentid)REFERENCES Students(StudentID), FOREIGN KEY (courseid) REFERENCES Courses(courseid));

**Purpose**: To create the Enrolments table that links students to courses using foreign keys and stores enrollment dates.

They define structure and enforce referential integrity using **foreign keys**. There is **no output**, but successful execution sets up the database structure required for the tasks.

**Data Insertion**

**Insert Courses**

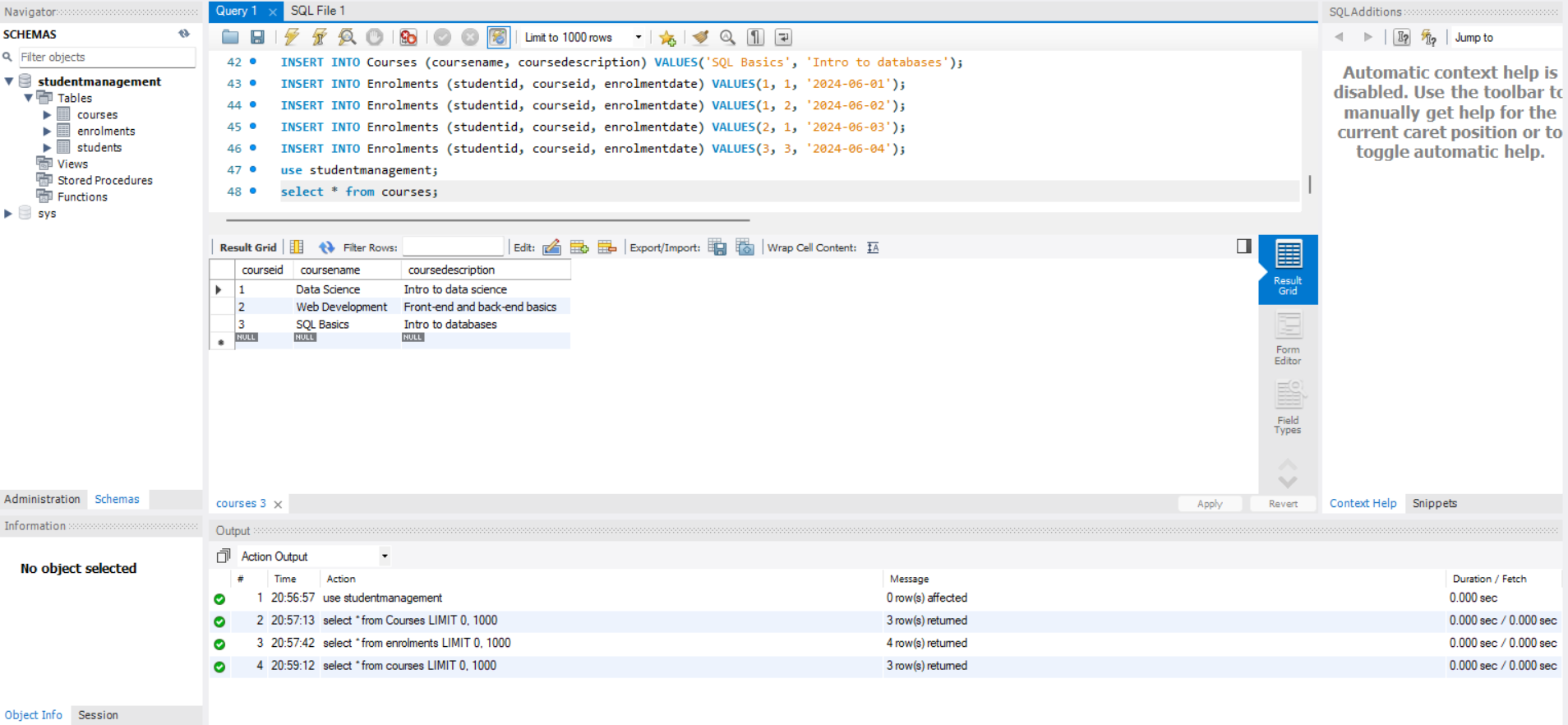
INSERT INTO Courses (coursename, coursedescription) VALUES('Data Science', 'Intro to data science');

INSERT INTO Courses (coursename, coursedescription) VALUES('Web Development', 'Front-end and back-end basics');

INSERT INTO Courses (coursename, coursedescription) VALUES('SQL Basics', 'Intro to databases');

**Purpose**: To insert sample course data into the Courses table for testing and demonstration purposes.

**Output:** select \* from courses;

****

**Insert Enrolments**

INSERT INTO Enrolments (studentid, courseid, enrolmentdate) VALUES(1, 1, '2024-06-01');

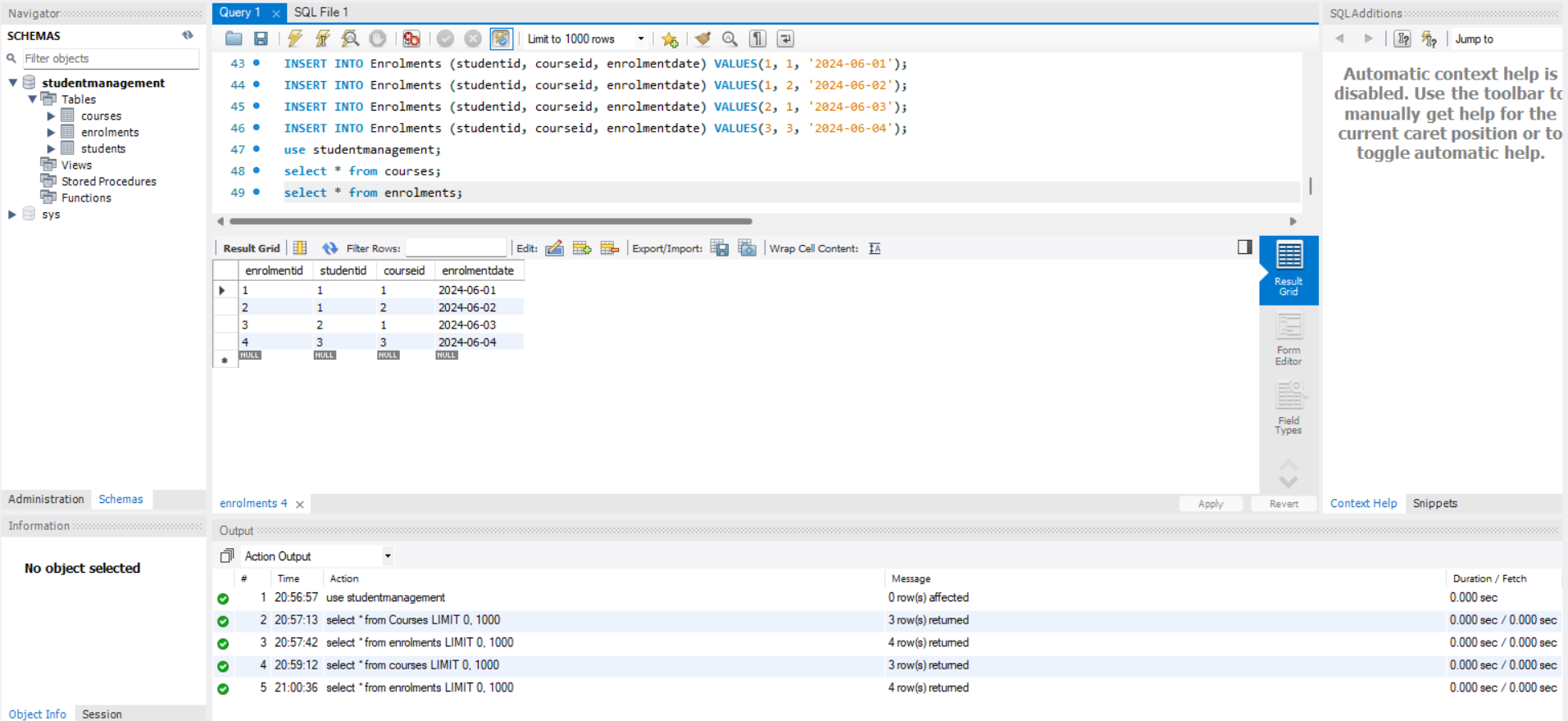
INSERT INTO Enrolments (studentid, courseid, enrolmentdate) VALUES(1, 2, '2024-06-02');

INSERT INTO Enrolments (studentid, courseid, enrolmentdate) VALUES(2, 1, '2024-06-03');

INSERT INTO Enrolments (studentid, courseid, enrolmentdate) VALUES(3, 3, '2024-06-04');

**Purpose**: To insert sample enrollment data that establishes relationships between students and courses.

**Output:** select \* from enrolments;

****

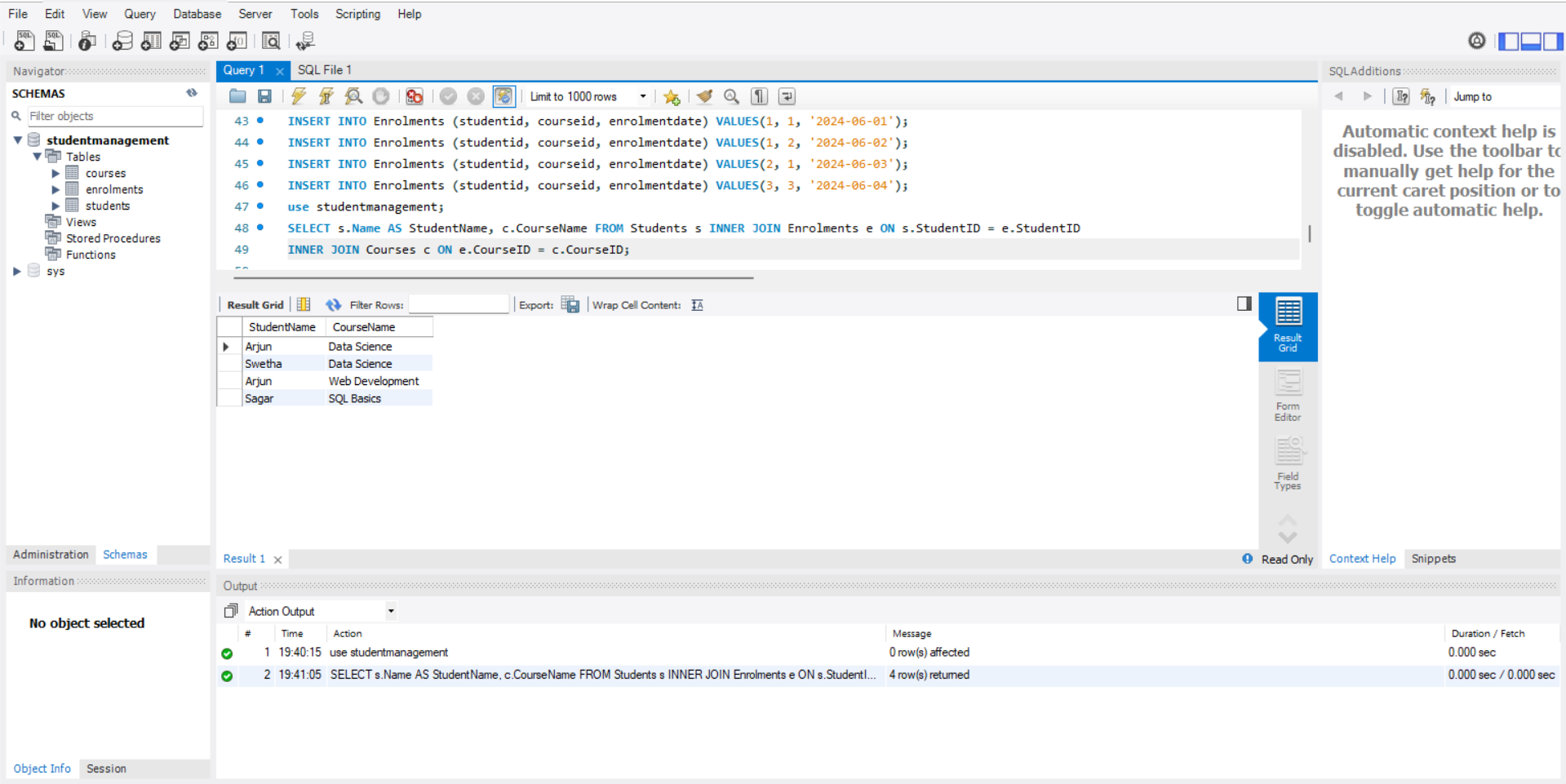
The output is "Query OK", and it confirms data is successfully added — allowing meaningful SELECT queries to be run for analysis.

**Task 1 Query**

SELECT s.Name AS StudentName, c.CourseName FROM Students s INNER JOIN Enrolments e ON s.StudentID = e.StudentID INNER JOIN Courses c ON e.CourseID = c.CourseID;

**Purpose**: To retrieve a list of students along with the names of the courses they are enrolled in.

**Output:**

****

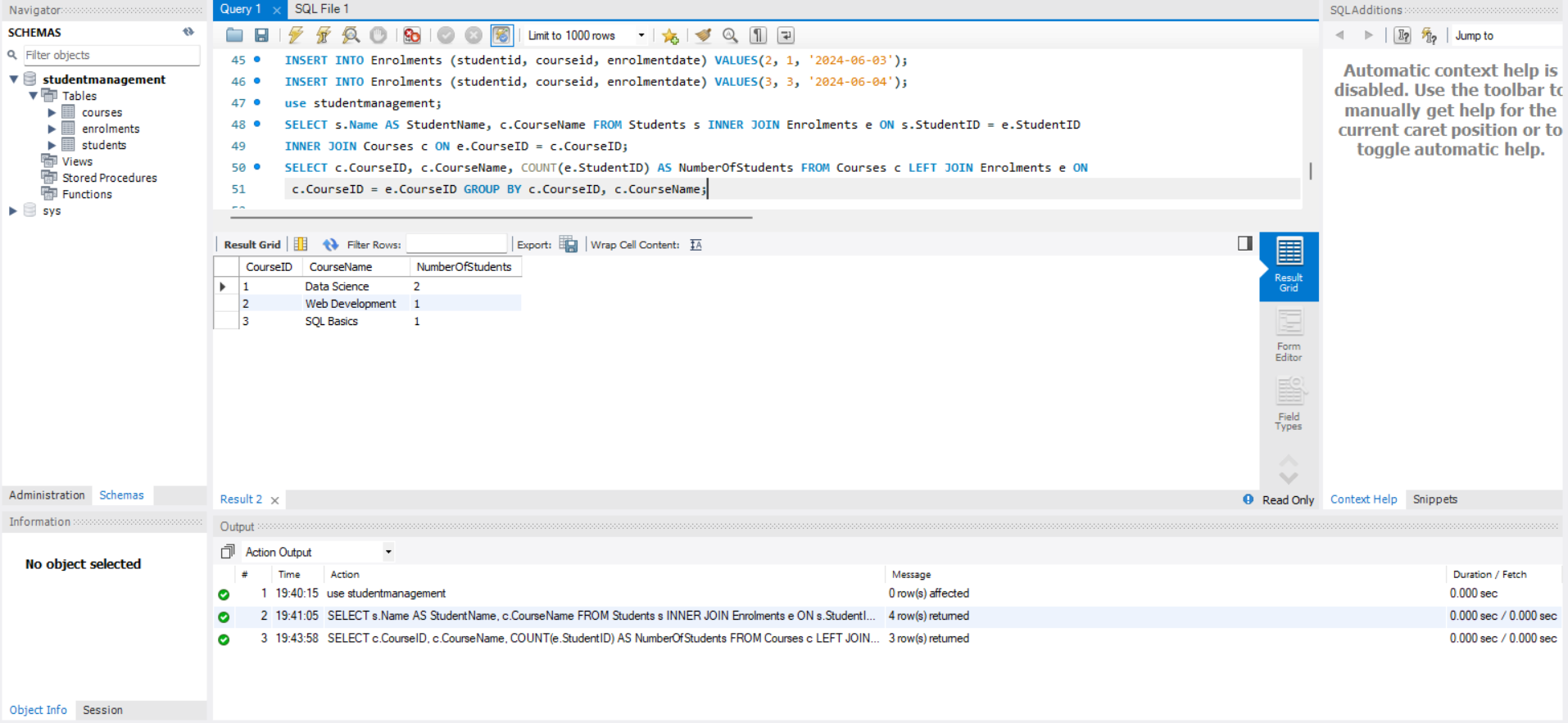
It joins all relevant tables and provides a clear view of enrolments. Directly fulfills the task: list all students and the courses they are enrolled in.

**Task 2 Query**

SELECT c.CourseID, c.CourseName, COUNT(e.StudentID) AS NumberOfStudents FROM Courses c LEFT JOIN Enrolments e ON c.CourseID = e.CourseID GROUP BY c.CourseID, c.CourseName;

**Purpose**: To count the number of students enrolled in each course, including courses with no students.

**Output:**

****

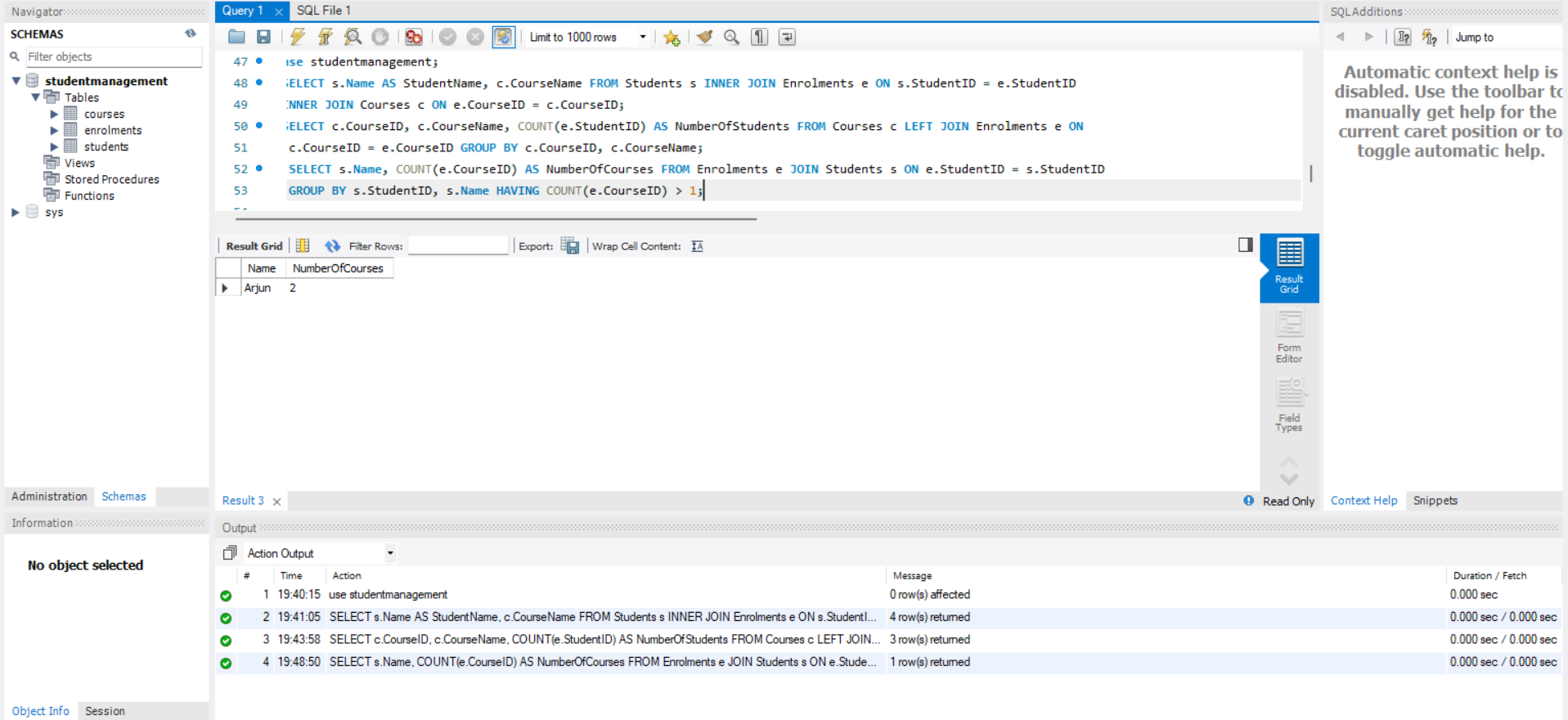
Uses LEFT JOIN to ensure even courses with zero enrolments are shown. Then, COUNT() gives total students per course.

**Task 3 Query**

SELECT s.Name, COUNT(e.CourseID) AS NumberOfCourses FROM Enrolments e JOIN Students s ON e.StudentID = s.StudentID GROUP BY s.StudentID, s.Name HAVING COUNT(e.CourseID) > 1;

**Purpose**: To identify students who are enrolled in more than one course.

**Output:**

****

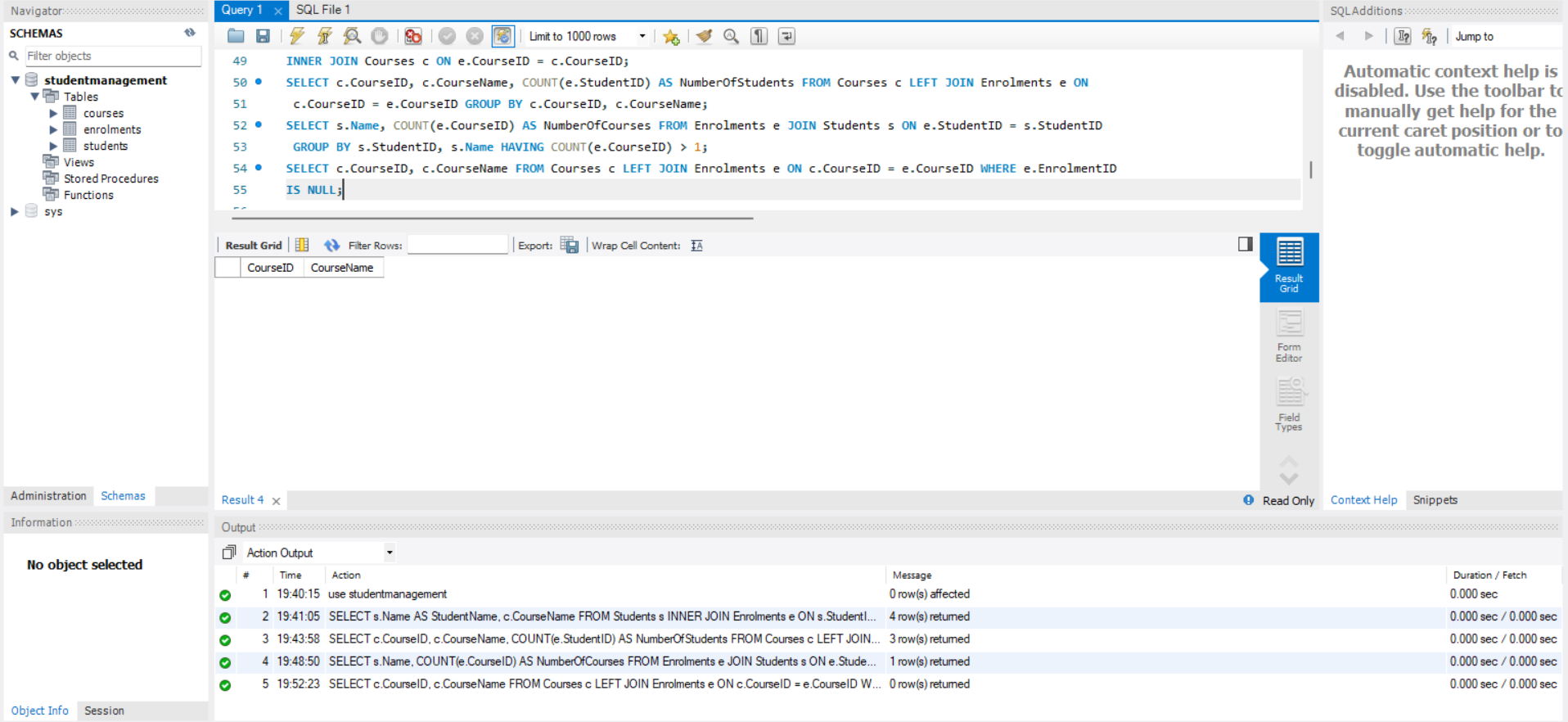
Finds students who enrolled in multiple courses by grouping by student and filtering with HAVING. Accurately identifies multi-course enrolments.

**Task 4 Query**

SELECT c.CourseID, c.CourseName FROM Courses c LEFT JOIN Enrolments e ON c.CourseID = e.CourseID WHERE e.EnrolmentID IS NULL;

**Purpose:** To find courses that have no enrolled students.

**Output:**

****

Finds courses with no matching enrolments by checking for NULL in the joined Enrolments. The empty output means all courses have at least one student — which still satisfies the task by proving no such courses currently exist.

**Summary:**

1. Table creation: The Courses and Enrolments tables were successfully created. The Courses table stores details about each course, including a unique course ID, course name, and description. The Enrolments table establishes a relationship between students and courses using foreign keys, ensuring referential integrity by linking to the Students and Courses tables.
2. Data Insertion: Data was correctly inserted into both the Courses and Enrolments tables. Three courses—Data Science, Web Development, and SQL Basics—were added. Four enrolment records were also inserted, linking specific students to these courses along with the corresponding enrolment dates.
3. Task1: A query using joins displays the students enrolled in each course. This confirms that the relationship between students and courses has been correctly implemented.
4. Task2: All courses have at least one enrolment, with data science being the most popular course. A left join query determine the number of students enrolled in each corse.
5. Task3: A grouped query found students in multiple courses. The output showed one student in several, confirming system support for various student enrollments is strong.
6. Task4: A query was executed to detect any courses without student enrolments. The result returned zero rows, confirming that all available courses have been enrolled in and that there are no inactive or unassigned courses in the system.