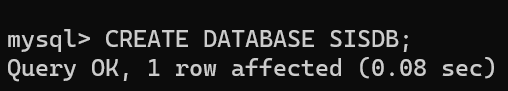
**Task 1. Database Design:**

**1. Create the database named "SISDB"**

CREATE DATABASE SISDB;



**2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships. a. Students b. Courses c. Enrollments d. Teacher e. Payments**

Schema for Students:

CREATE table Students(

student\_id INT PRIMARY KEY,

first\_name VARCHAR(100),

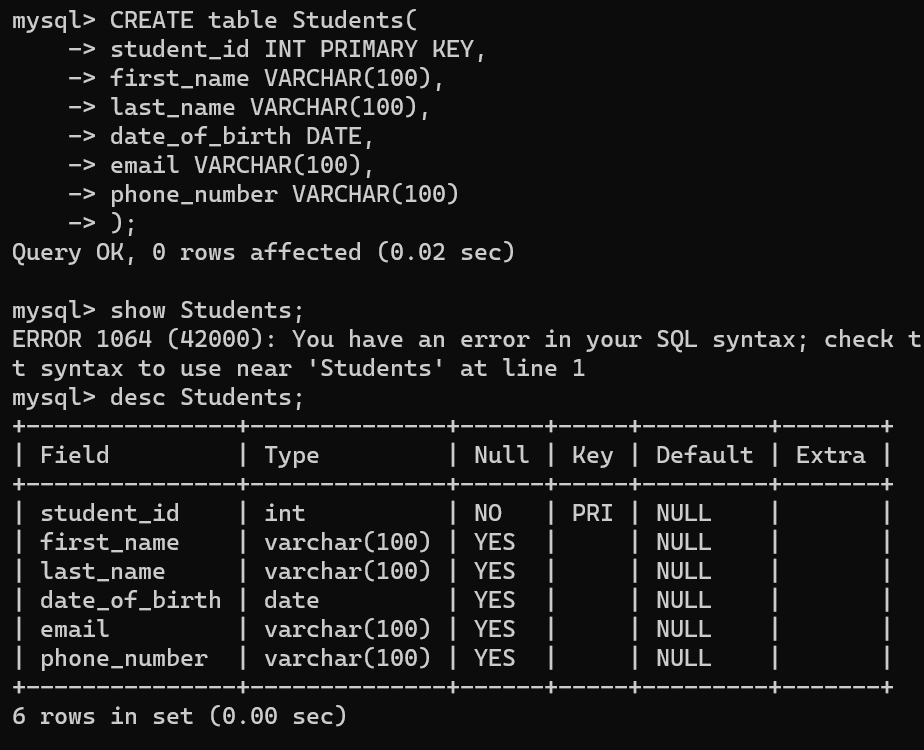
last\_name VARCHAR(100),

date\_of\_birth DATE,

email VARCHAR(100),

phone\_number VARCHAR(100)

);



Schema for Courses:

CREATE table Courses(

course\_id INT PRIMARY KEY,

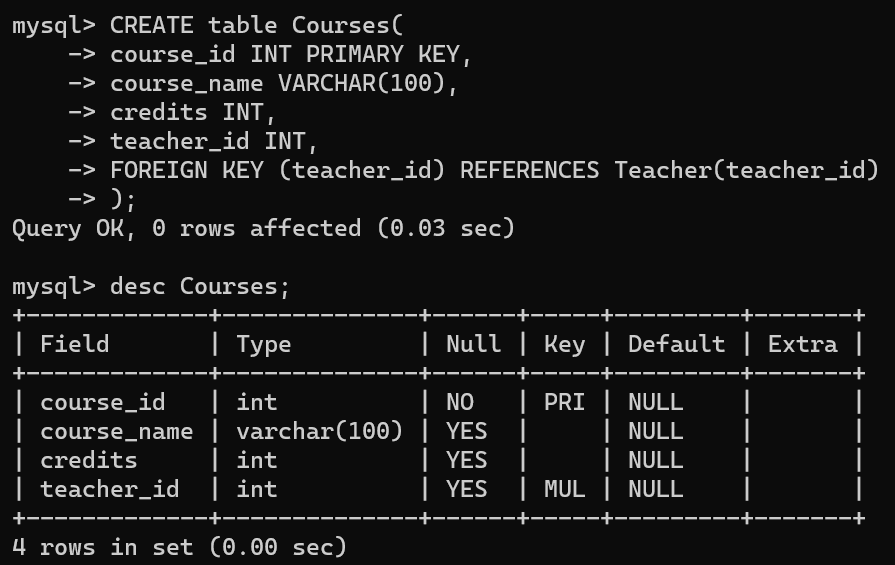
course\_name VARCHAR(100),

credits INT,

teacher\_id INT,

FOREIGN KEY (teacher\_id) REFERENCES Teacher(teacher\_id)

);



Schema for Enrollments:

CREATE table Enrollments (

enrollment\_id INT PRIMARY KEY,

student\_id INT,

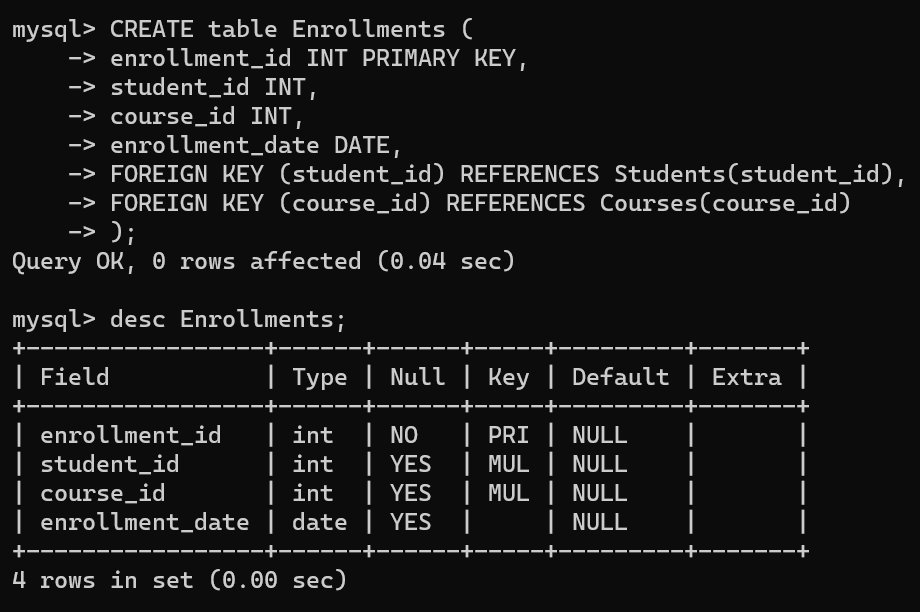
course\_id INT,

enrollment\_date DATE,

FOREIGN KEY (student\_id) REFERENCES Students(student\_id),

FOREIGN KEY (course\_id) REFERENCES Courses(course\_id)

);



Schema for Teacher

CREATE table Teacher(

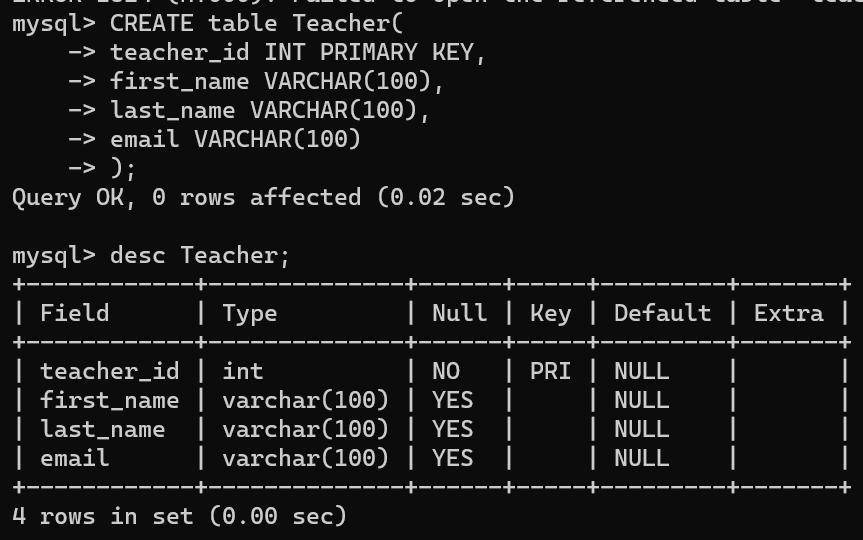
teacher\_id INT PRIMARY KEY,

first\_name VARCHAR(100),

last\_name VARCHAR(100),

email VARCHAR(100)

);



Schema for Payments:

CREATE table Payments(

payment\_id INT PRIMARY KEY,

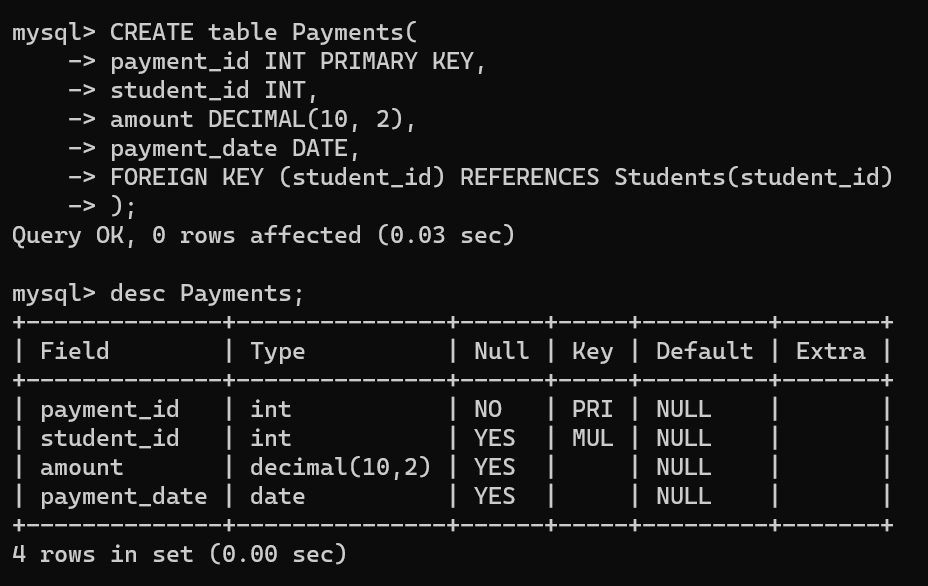
student\_id INT,

amount DECIMAL(10, 2),

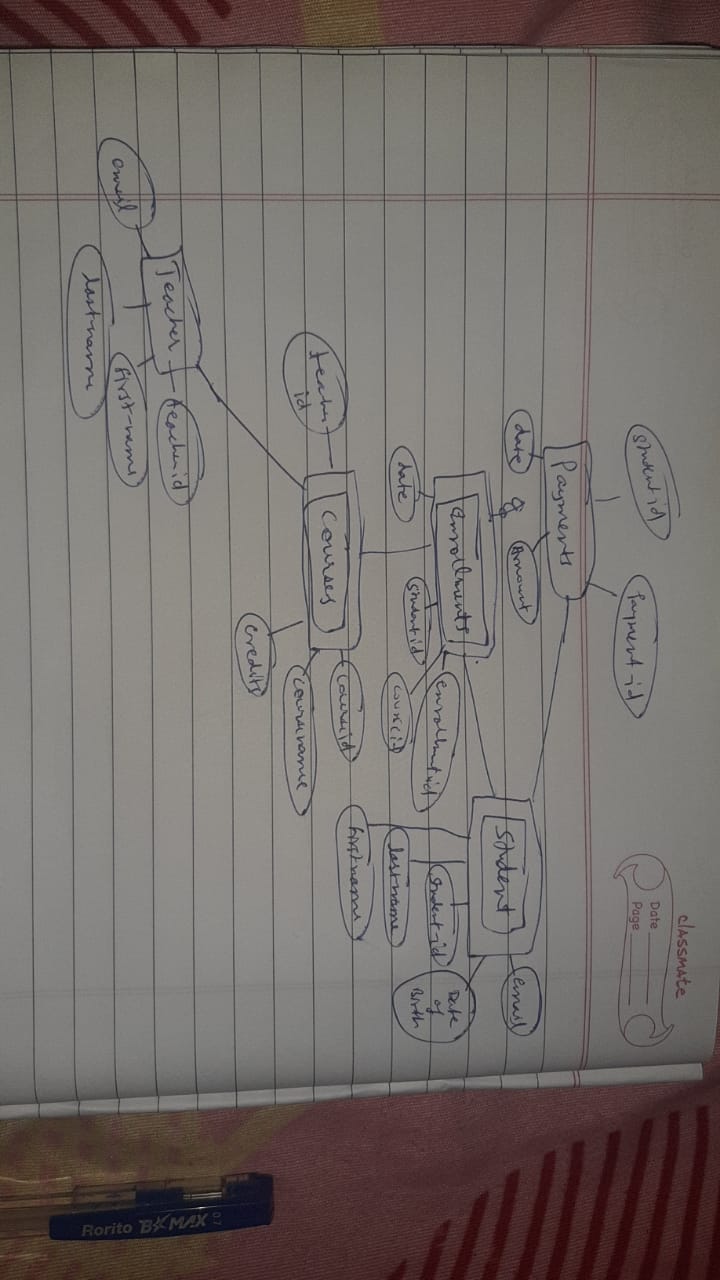
payment\_date DATE,

FOREIGN KEY (student\_id) REFERENCES Students(student\_id)

);



**3. Create an ERD (Entity Relationship Di**

**agram) for the database.**

**4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.**

Primary Key constraint to Students table

ALTER TABLE Students

ADD CONSTRAINT PK\_Students PRIMARY KEY (student\_id);

Primary Key constraint to Courses table

ALTER TABLE Courses

ADD CONSTRAINT PK\_Courses PRIMARY KEY (course\_id);

Primary Key constraint to Enrollments table

ALTER TABLE Enrollments

ADD CONSTRAINT PK\_Enrollments PRIMARY KEY (enrollment\_id);

Foreign Key constraint to Enrollments table referencing Students table

ALTER TABLE Enrollments

ADD CONSTRAINT FK\_Enrollments\_Students FOREIGN KEY (student\_id) REFERENCES Students(student\_id);

Foreign Key constraint to Enrollments table referencing Courses table

ALTER TABLE Enrollments

ADD CONSTRAINT FK\_Enrollments\_Courses FOREIGN KEY (course\_id) REFERENCES Courses(course\_id);

Primary Key constraint to Teacher table

ALTER TABLE Teacher

ADD CONSTRAINT PK\_Teacher PRIMARY KEY (teacher\_id);

Foreign Key constraint to Courses table referencing Teacher table

ALTER TABLE Courses

ADD CONSTRAINT FK\_Courses\_Teacher FOREIGN KEY (teacher\_id) REFERENCES Teacher(teacher\_id);

Primary Key constraint to Payments table

ALTER TABLE Payments

ADD CONSTRAINT PK\_Payments PRIMARY KEY (payment\_id);

Foreign Key constraint to Payments table referencing Students table

ALTER TABLE Payments

ADD CONSTRAINT FK\_Payments\_Students FOREIGN KEY (student\_id) REFERENCES Students(student\_id);

**5. Insert at least 10 sample records into each of the following tables. i. Students ii. Courses iii. Enrollments iv. Teacher v. Payments**

INSERT INTO Students(student\_id, first\_name, last\_name, date\_of\_birth, email, phone\_number) VALUES

(1, 'Pujan', 'Rastogi', '2002-08-23', 'pujan1@gmail.com', '1234567890'),

(2, 'Abhi', 'Ag', '2000-09-20', 'abhi@gmail.com', '9876543210'),

(3, 'Abhishek', 'Chaursaiya', '1999-12-10', 'abhich1@gmail.com', '9812347650'),

(4, 'Raghav', 'Aggarwal', '1993-01-11', 'raghav@gmail.com', '1237890456'),

(5, 'Ishan', 'Sharma', '1995-07-17', 'ishan@gmail.com', '2345678901'),

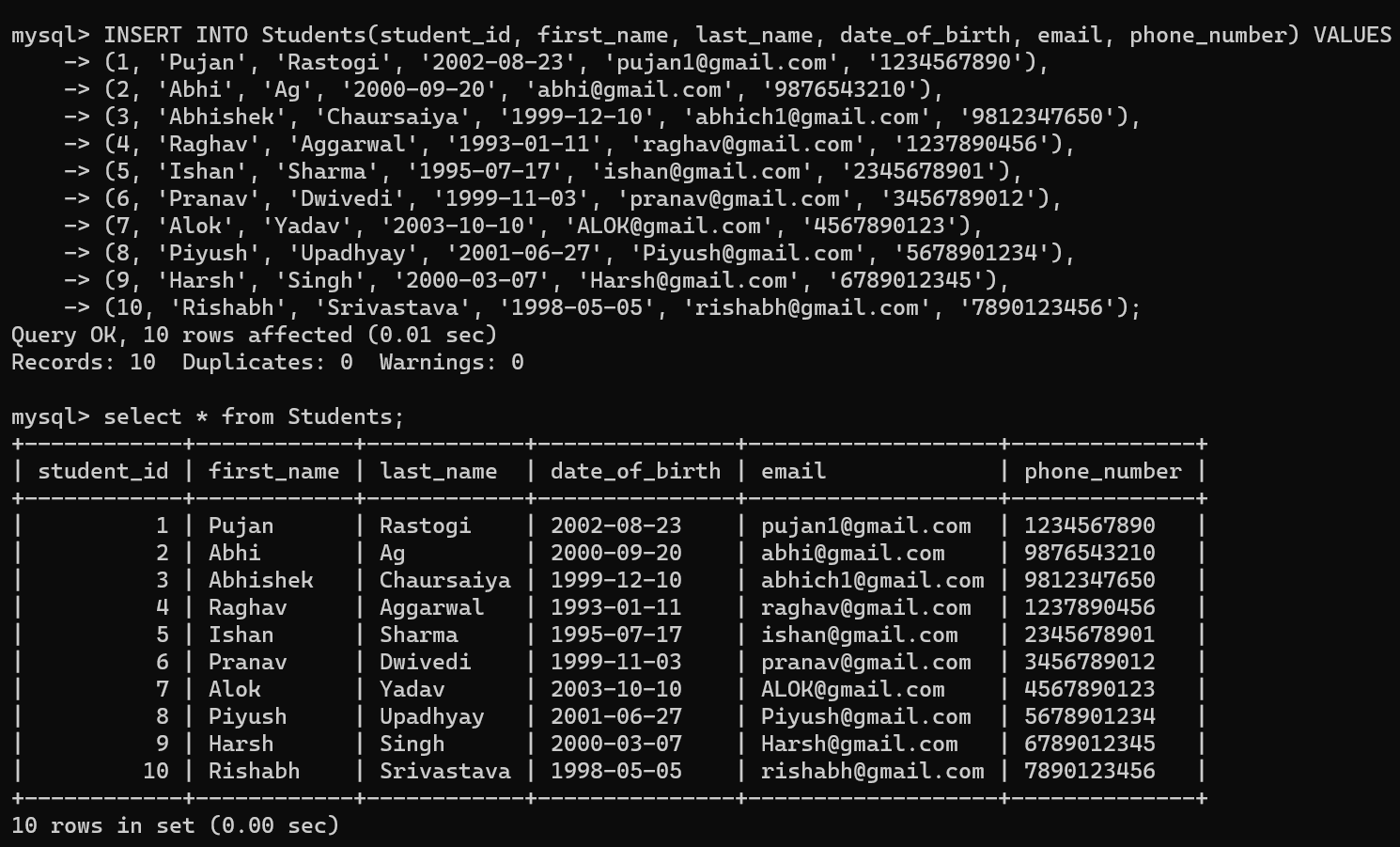
(6, 'Pranav', 'Dwivedi', '1999-11-03', 'pranav@gmail.com', '3456789012'),

(7, 'Alok', 'Yadav', '2003-10-10', 'ALOK@gmail.com', '4567890123'),

(8, 'Piyush', 'Upadhyay', '2001-06-27', 'Piyush@gmail.com', '5678901234'),

(9, 'Harsh', 'Singh', '2000-03-07', 'Harsh@gmail.com', '6789012345'),

(10, 'Rishabh', 'Srivastava', '1998-05-05', 'rishabh@gmail.com', '7890123456');



INSERT INTO Courses(course\_id, course\_name, credits, teacher\_id) VALUES

(1, 'Maths’, 10, 1),

(2, ‘English’, 10, 2),

(3, ‘Hindi’, 20, 3),

(4, ‘Social Studies’, 20, 4),

(5, ‘Chemistry’, 10, 5),

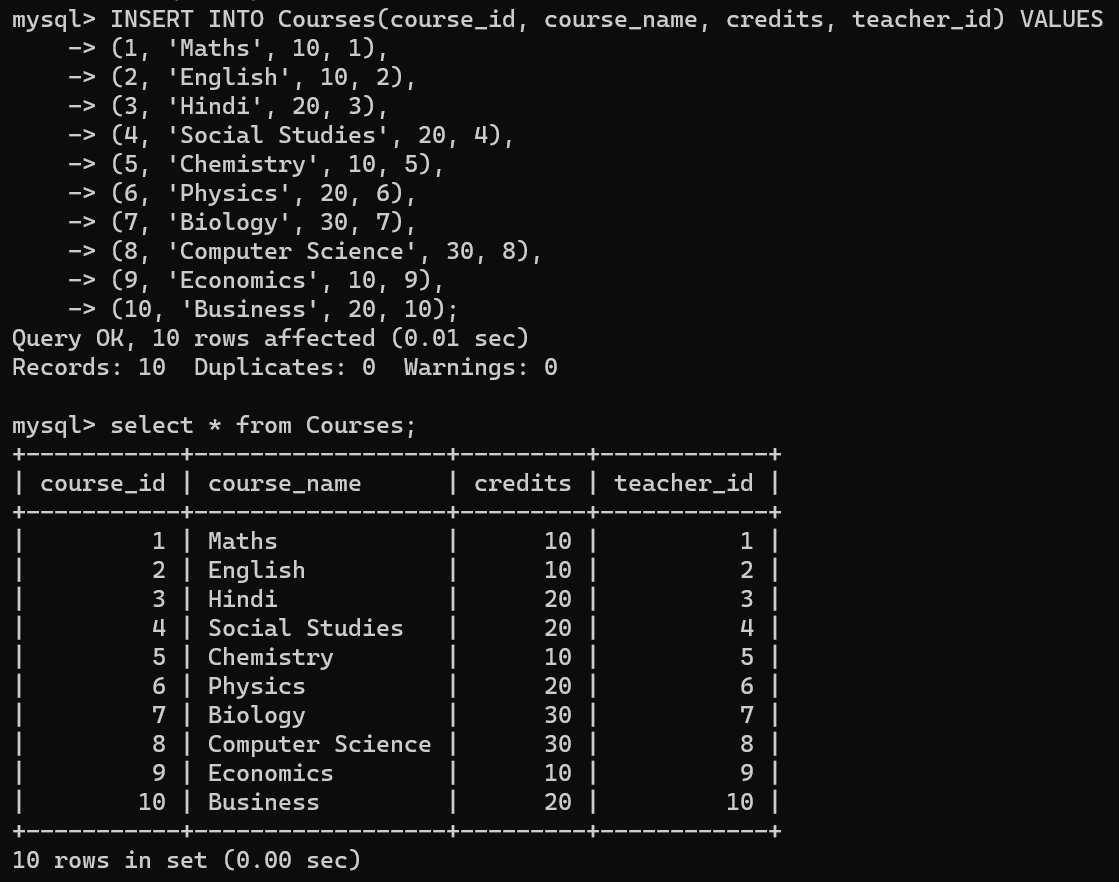
(6, ‘Physics’, 20, 6),

(7, ‘Biology’, 30, 7),

(8, ‘Computer Science’, 30, 8),

(9, ‘Economics’, 10, 9),

(10, ‘Business’, 20, 10);



INSERT INTO Enrollments (enrollment\_id, student\_id, course\_id, enrollment\_date) VALUES

(1, 1, 6, ‘2023-01-10’),

(2, 2, 1, ‘2023-02-11’),

(3, 3, 5, ‘2023-02-15’),

(4, 4, 2, ‘2023-02-18’),

(5, 5, 7, ‘2023-03-21’),

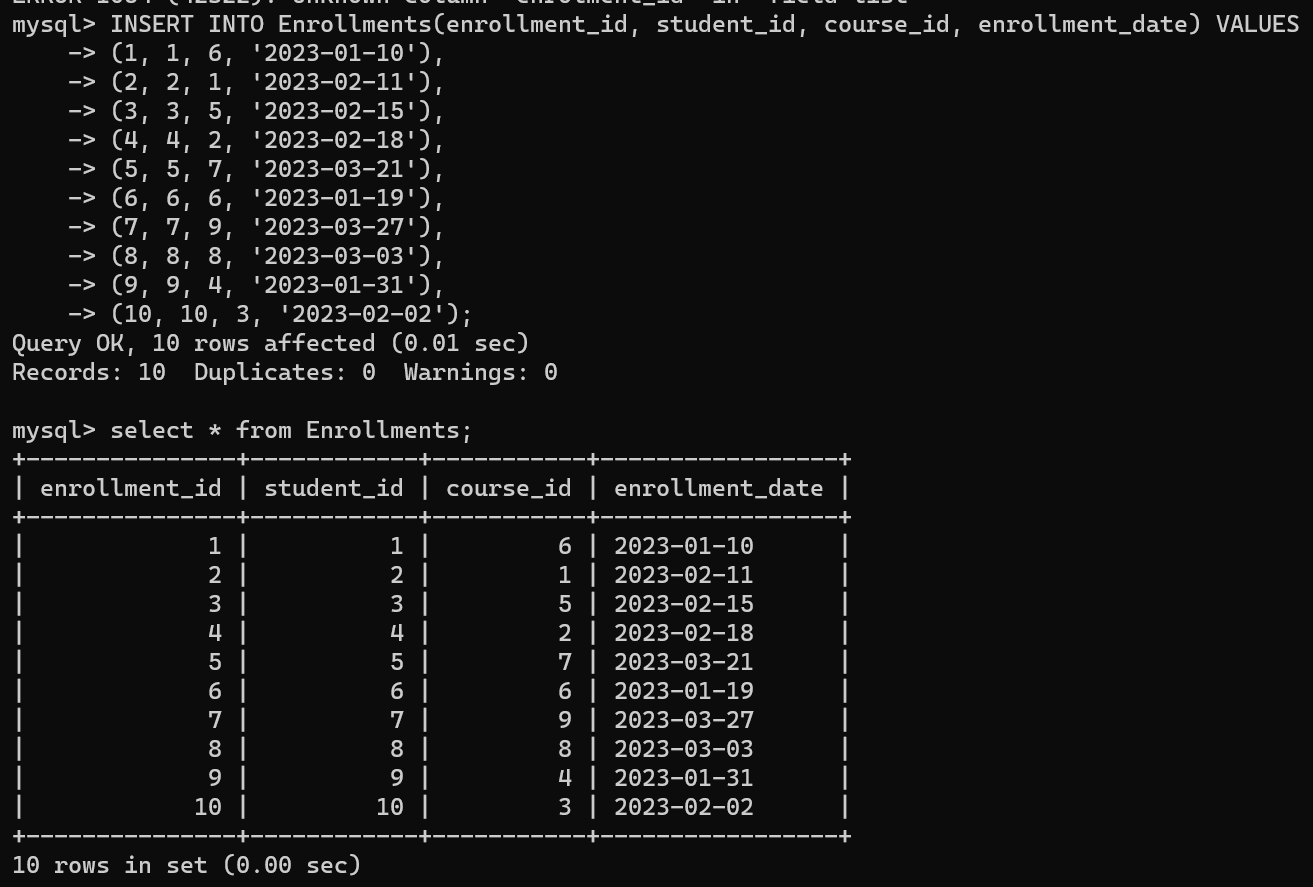
(6, 6, 6, ‘2023-01-19’),

(7, 7, 9, ‘2023-03-27’),

(8, 8, 8, ‘2023-03-03’),

(9, 9, 4, ‘2023-01-31’),

(10, 10, 3, ‘2023-02-02’);



INSERT INTO Teacher (teacher\_id, first\_name, last\_name, email) VALUES

(1, 'Anuradha', 'Ag', 'anuradha@gmail.com'),

(2, 'Pankaj', 'Goel', 'goelpankaj@gmail.com'),

(3, 'Ankur', 'Sharma', 'ankur@gmail.com'),

(4, 'Meenakshi', 'Agg', 'meenakshi@gmail.com'),

(5, 'Rajeev', 'Gupta', 'rajeev@gmail.com'),

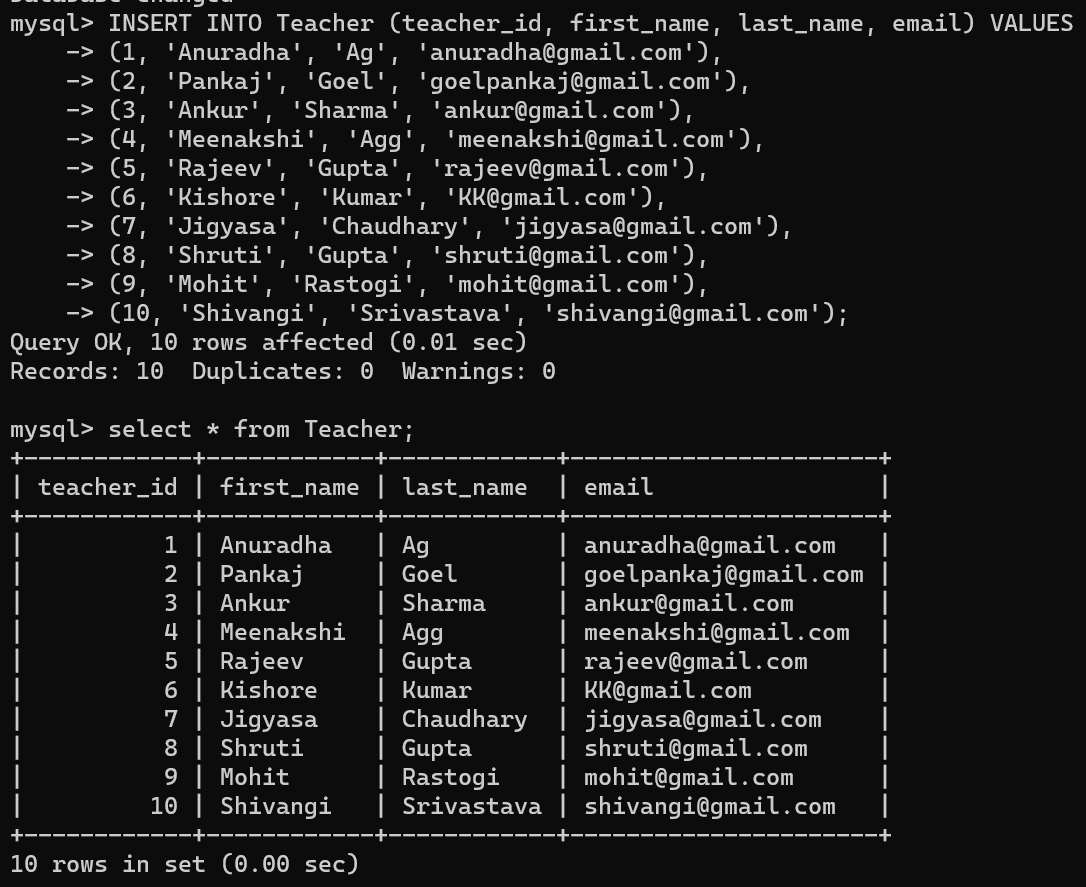
(6, 'Kishore', 'Kumar', 'KK@gmail.com'),

(7, 'Jigyasa', 'Chaudhary', 'jigyasa@gmail.com'),

(8, 'Shruti', 'Gupta', 'shruti@gmail.com'),

(9, 'Mohit', 'Rastogi', 'mohit@gmail.com'),

(10, 'Shivangi', 'Srivastava', 'shivangi@gmail.com');



INSERT INTO Payments(payment\_id, student\_id, amount, payment\_date) VALUES

(1, 1, 1000.00, '2023-01-05'),

(2, 2, 2000.00, '2023-02-02'),

(3, 3, 1500.00, '2023-01-19'),

(4, 4, 3000.00, '2023-01-10'),

(5, 5, 4000.00, '2023-02-23'),

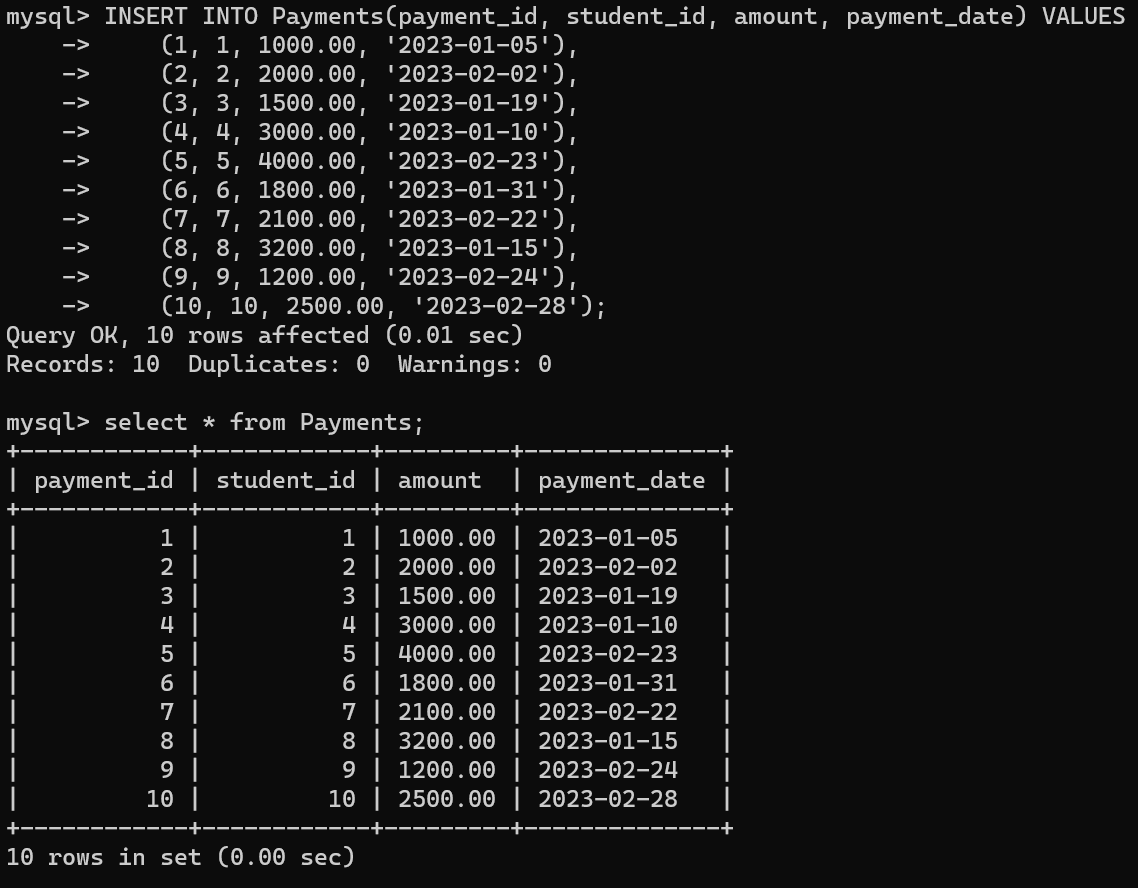
(6, 6, 1800.00, '2023-01-31'),

(7, 7, 2100.00, '2023-02-22'),

(8, 8, 3200.00, '2023-01-15'),

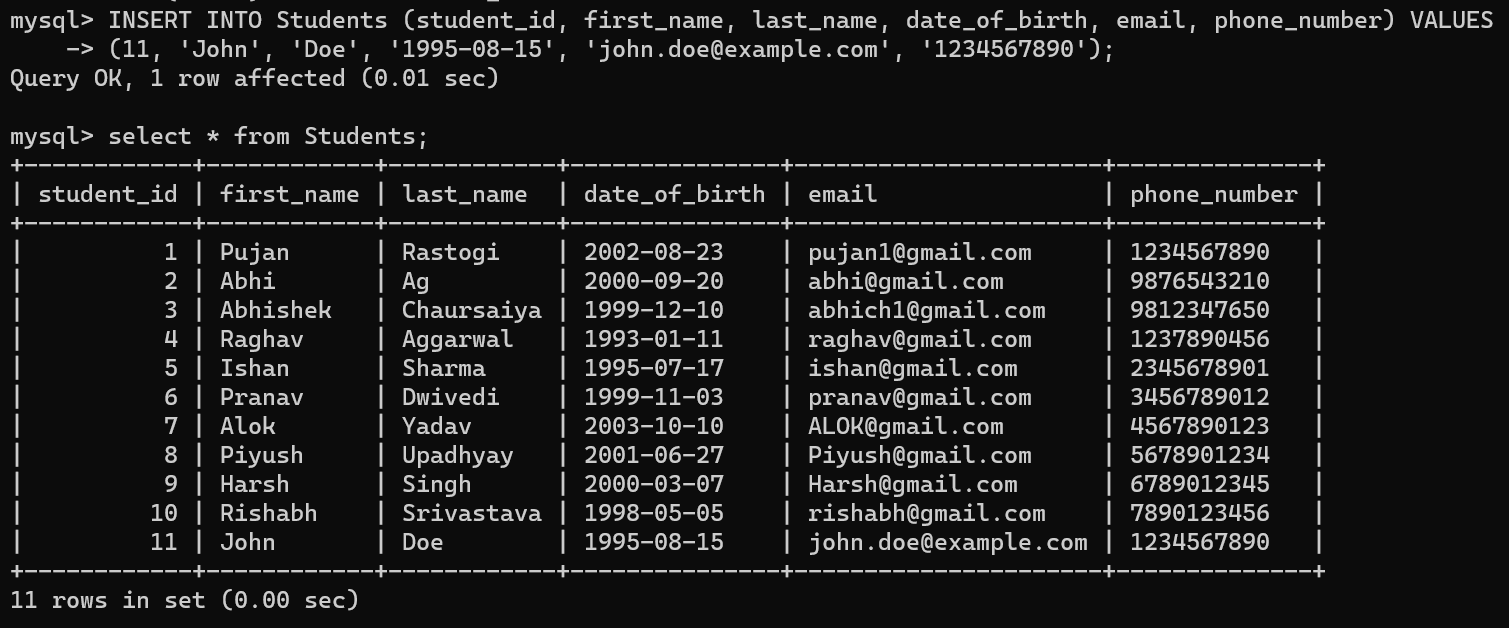
(9, 9, 1200.00, '2023-02-24'),

(10, 10, 2500.00, '2023-02-28');

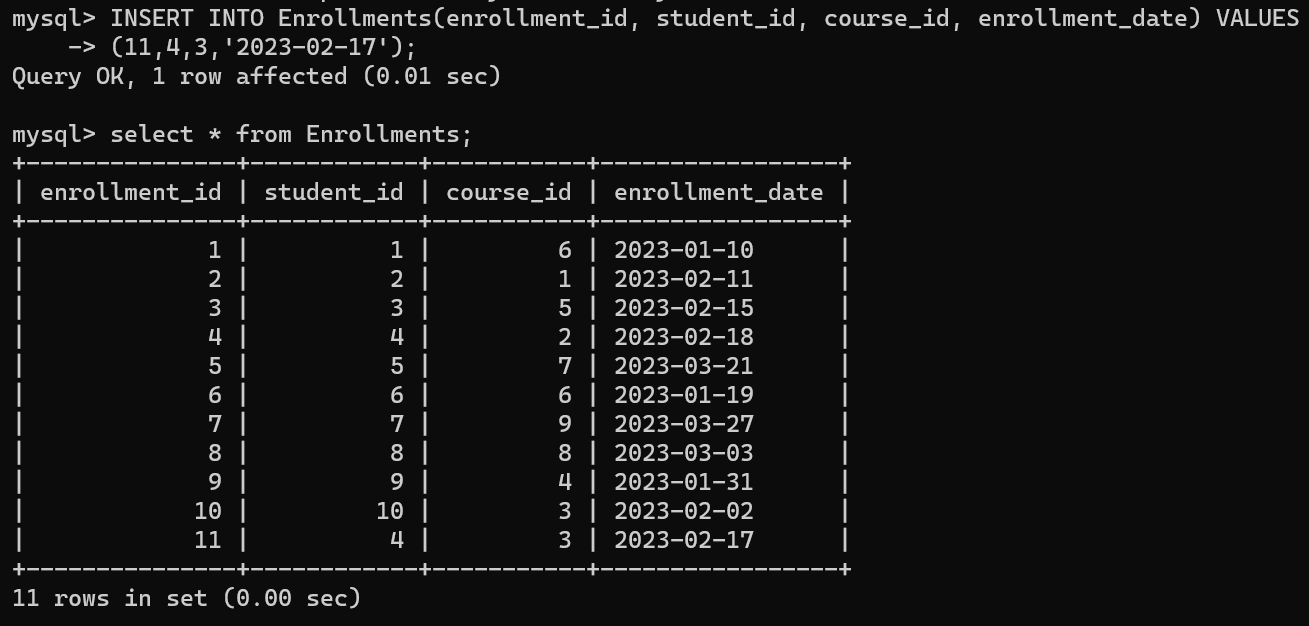


**Tasks 2: Select, Where, Between, AND, LIKE:**

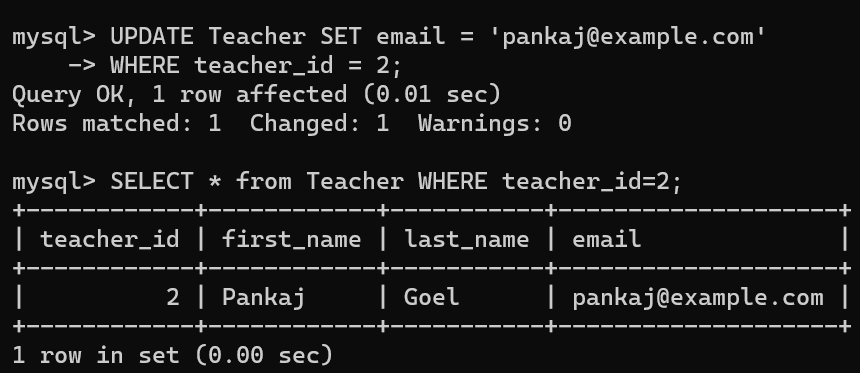
1. **Write an SQL query to insert a new student into the "Students" table with the following details: a. First Name: John b. Last Name: Doe c. Date of Birth: 1995-08-15 d. Email: john.doe@example.com e. Phone Number: 1234567890**

****

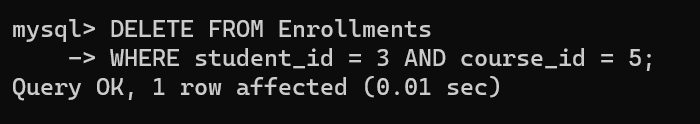
1. **Write an SQL query to enroll a student in a course. Choose an existing student and course and insert a record into the "Enrollments" table with the enrollment date.**

****

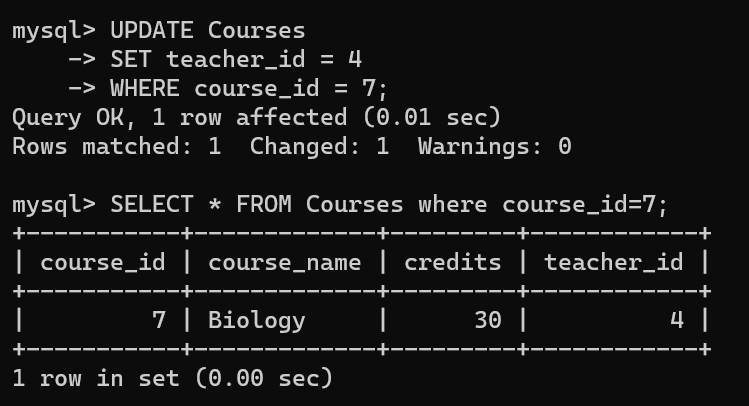
1. **Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.**

****

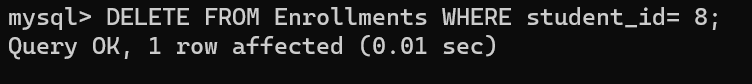
1. **Write an SQL query to delete a specific enrollment record from the "Enrollments" table. Select an enrollment record based on the student and course.**

****

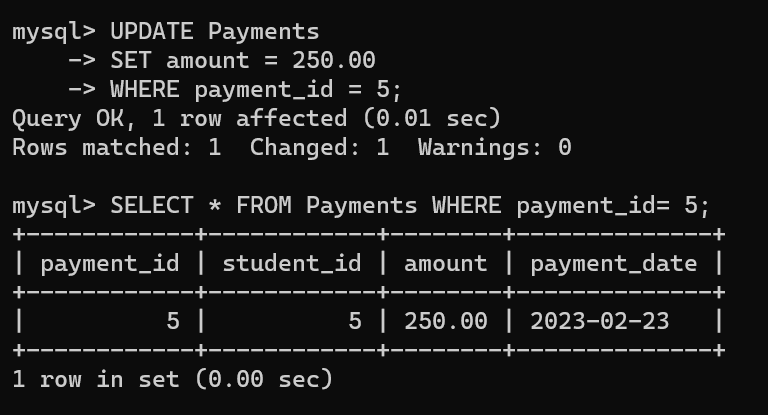
1. **Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.**

****

1. **Delete a specific student from the "Students" table and remove all their enrollment records from the "Enrollments" table. Be sure to maintain referential integrity.**

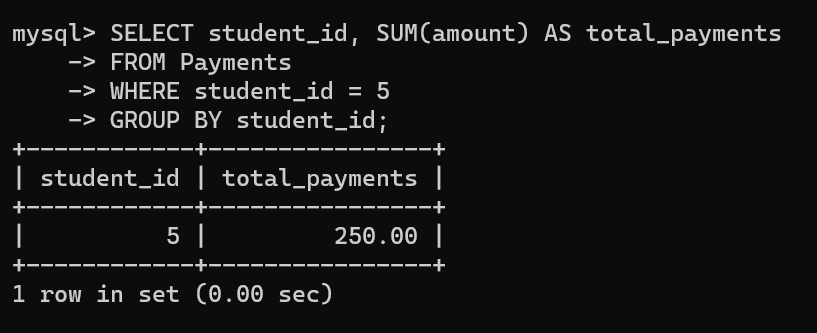
****

1. **Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.**

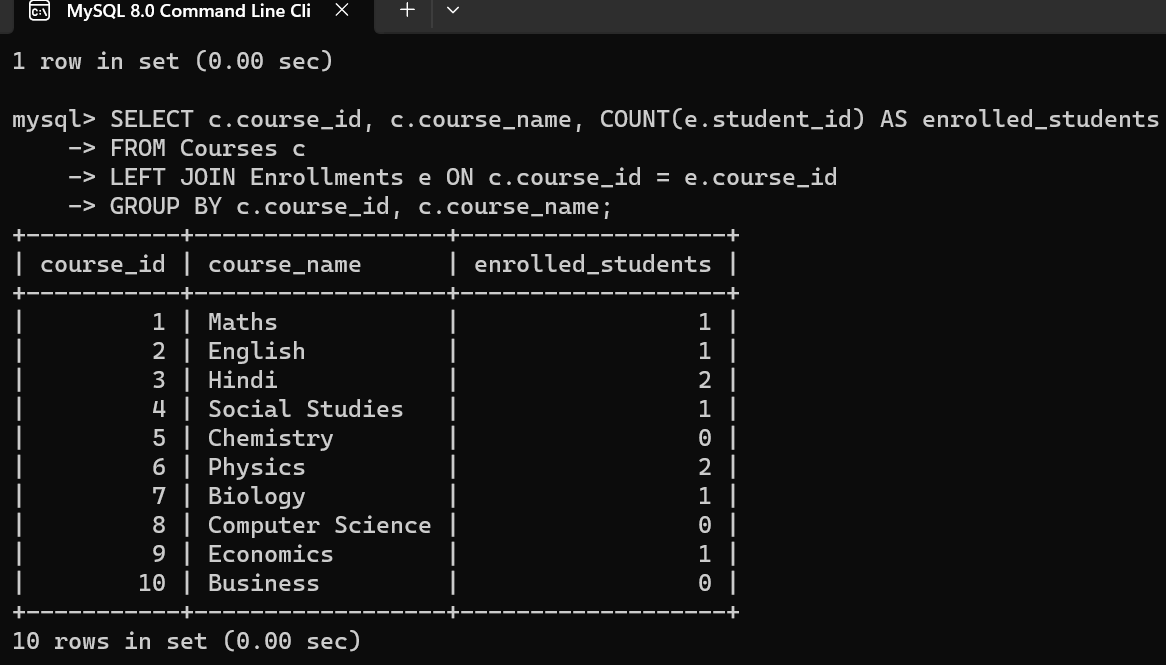
****

**Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:**

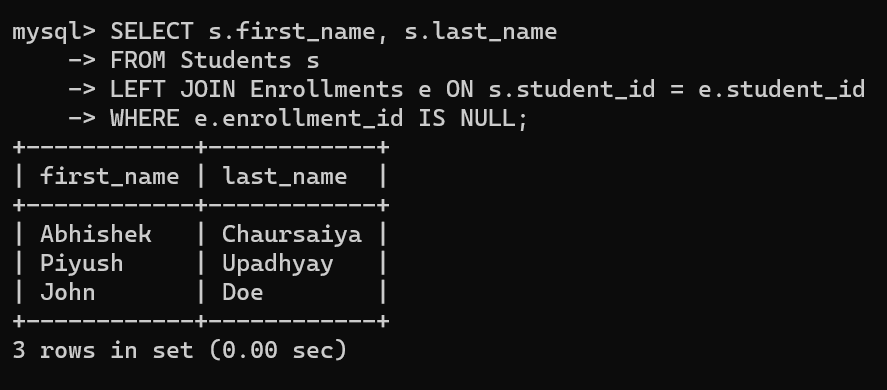
1. **Write an SQL query to calculate the total payments made by a specific student. You will need to join the "Payments" table with the "Students" table based on the student's ID.**

****

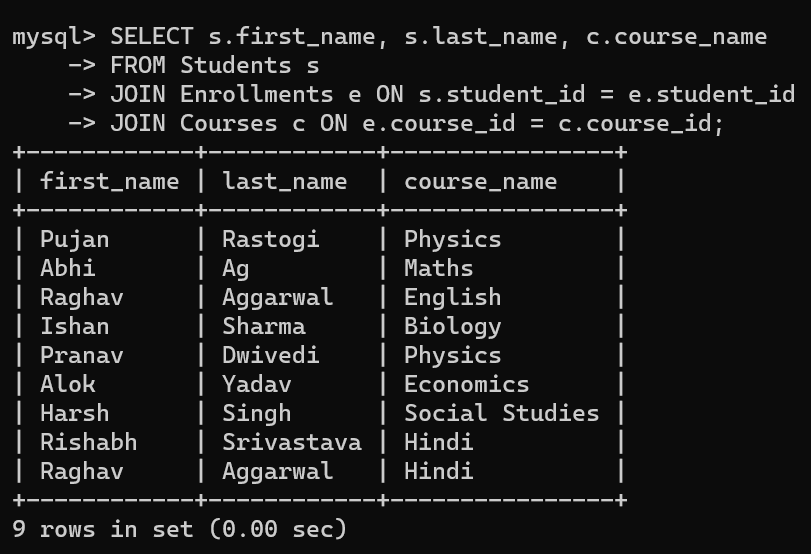
1. **Write an SQL query to retrieve a list of courses along with the count of students enrolled in each course. Use a JOIN operation between the "Courses" table and the "Enrollments" table.**

****

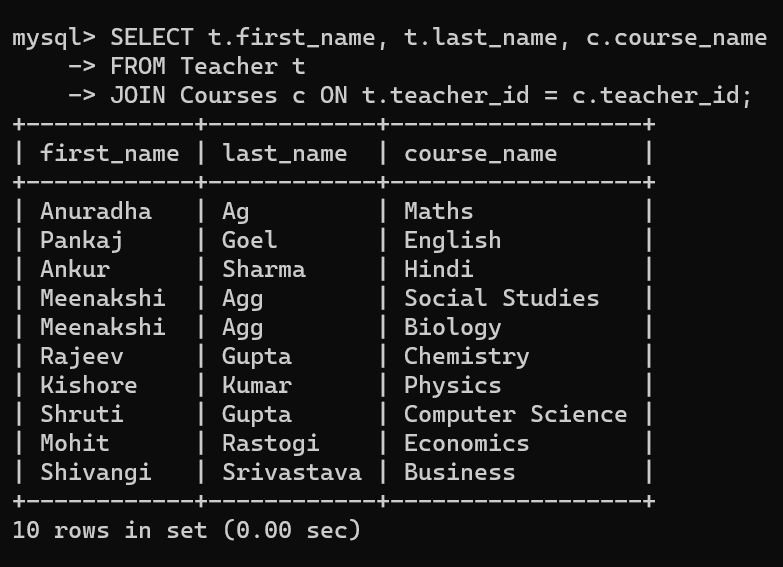
1. **Write an SQL query to find the names of students who have not enrolled in any course. Use a LEFT JOIN between the "Students" table and the "Enrollments" table to identify students without enrollments.**

****

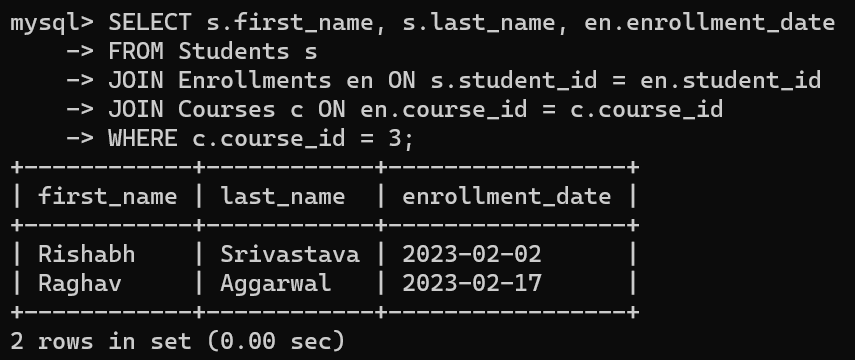
1. **Write an SQL query to retrieve the first name, last name of students, and the names of the courses they are enrolled in. Use JOIN operations between the "Students" table and the "Enrollments" and "Courses" tables.**

****

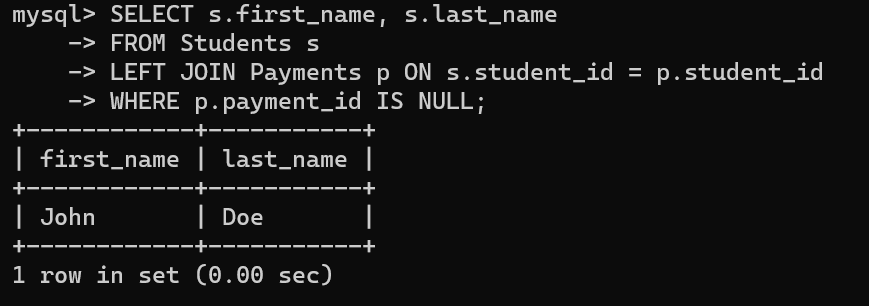
1. **Create a query to list the names of teachers and the courses they are assigned to. Join the "Teacher" table with the "Courses" table.**

****

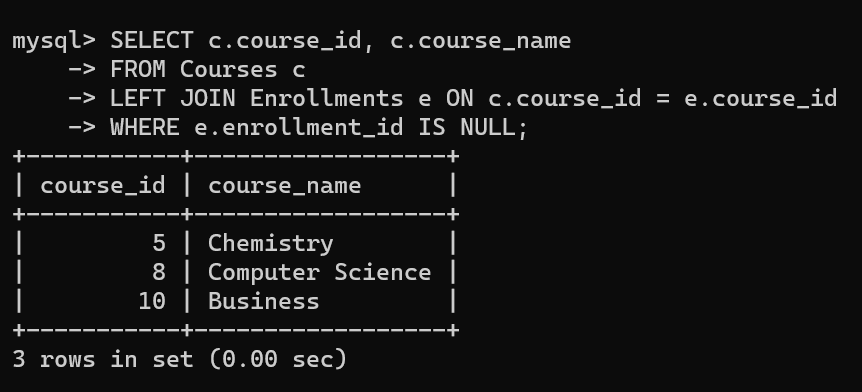
1. **Retrieve a list of students and their enrollment dates for a specific course. You'll need to join the "Students" table with the "Enrollments" and "Courses" tables.**

****

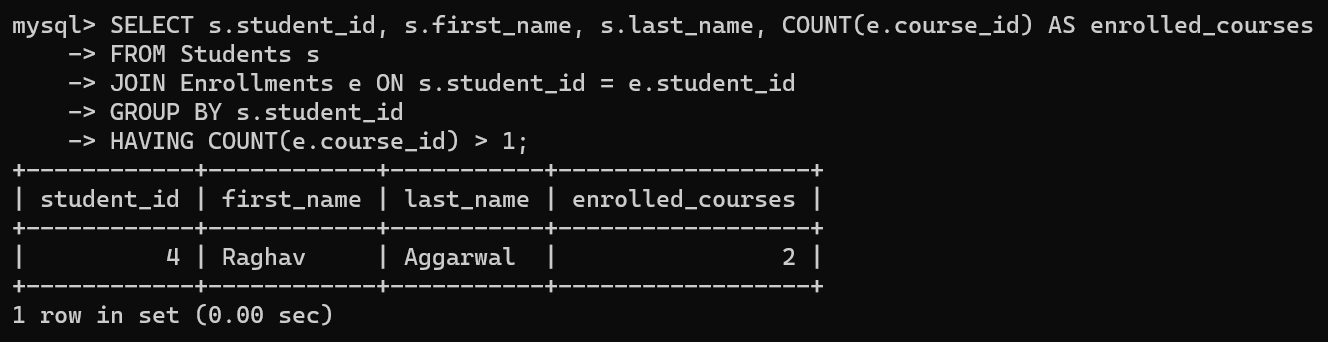
**7. Find the names of students who have not made any payments. Use a LEFT JOIN between the "Students" table and the "Payments" table and filter for students with NULL payment records.**

****

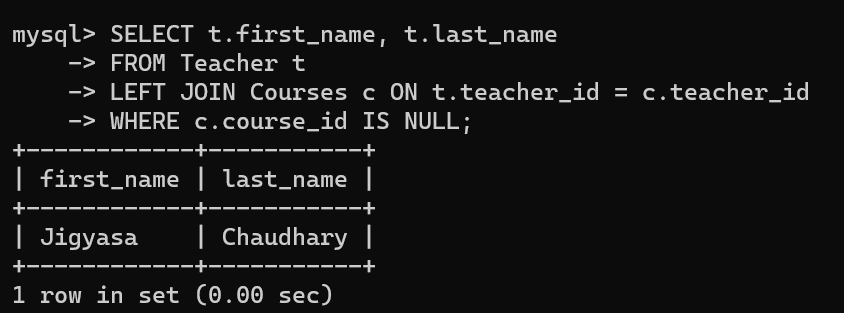
1. **Write a query to identify courses that have no enrollments. You'll need to use a LEFT JOIN between the "Courses" table and the "Enrollments" table and filter for courses with NULL enrollment records.**

****

1. **Identify students who are enrolled in more than one course. Use a self-join on the "Enrollments" table to find students with multiple enrollment records.**

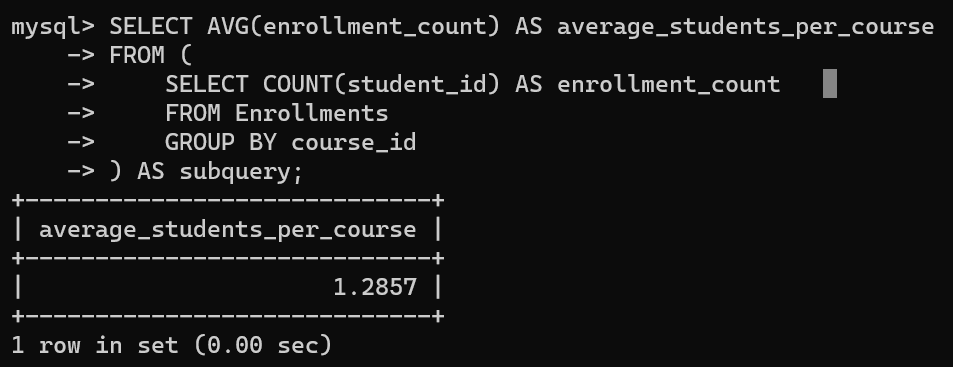
****

1. **Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher" table and the "Courses" table and filter for teachers with NULL course assignments.**

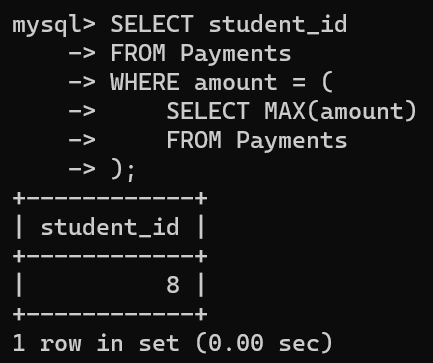
****

**Task 4. Subquery and its type:**

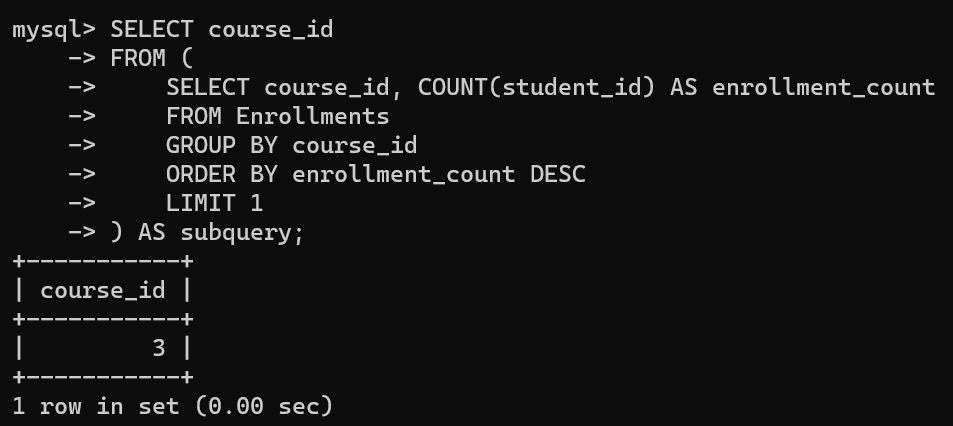
1. **Write an SQL query to calculate the average number of students enrolled in each course. Use aggregate functions and subqueries to achieve this.**

****

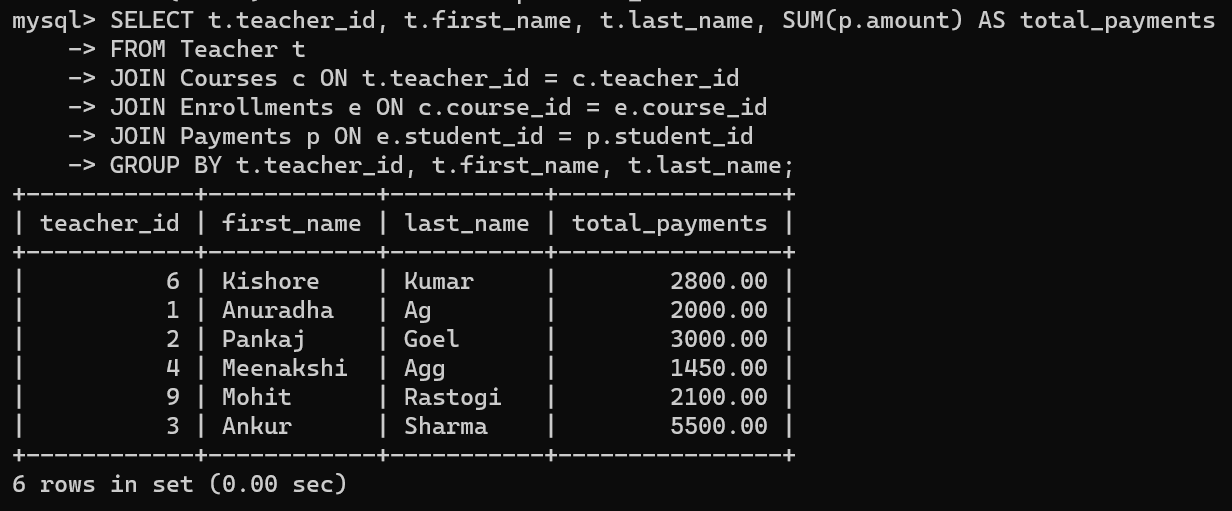
1. **Identify the student(s) who made the highest payment. Use a subquery to find the maximum payment amount and then retrieve the student(s) associated with that amount.**

****

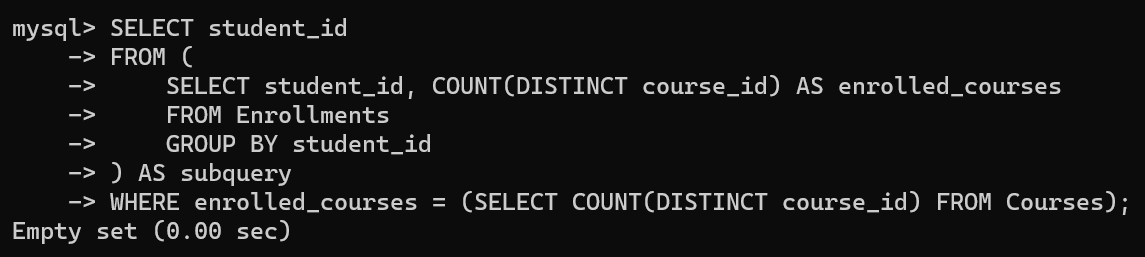
1. **Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the course(s) with the maximum enrollment count.**

****

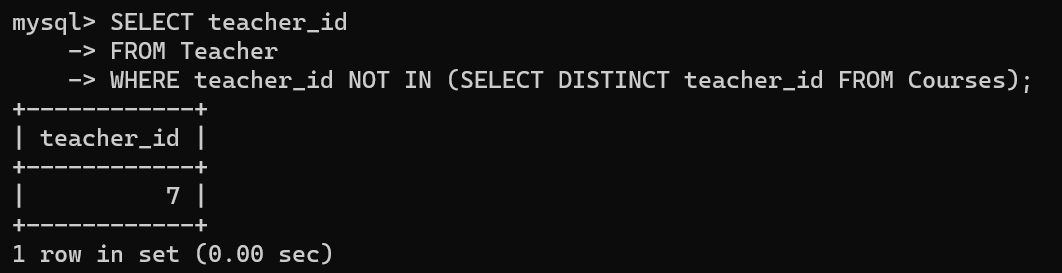
1. **Calculate the total payments made to courses taught by each teacher. Use subqueries to sum payments for each teacher's courses.**

****

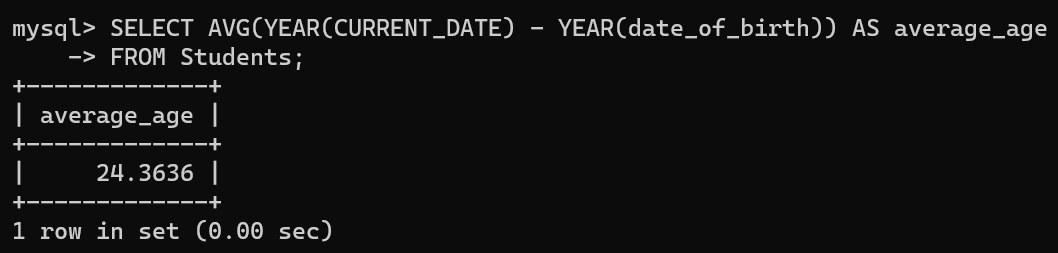
1. **Identify students who are enrolled in all available courses. Use subqueries to compare a student's enrollments with the total number of courses.**

****

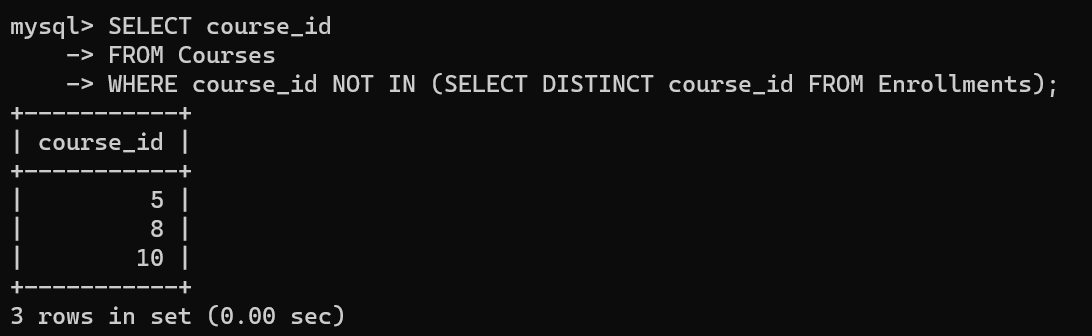
1. **Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to find teachers with no course assignments.**

****

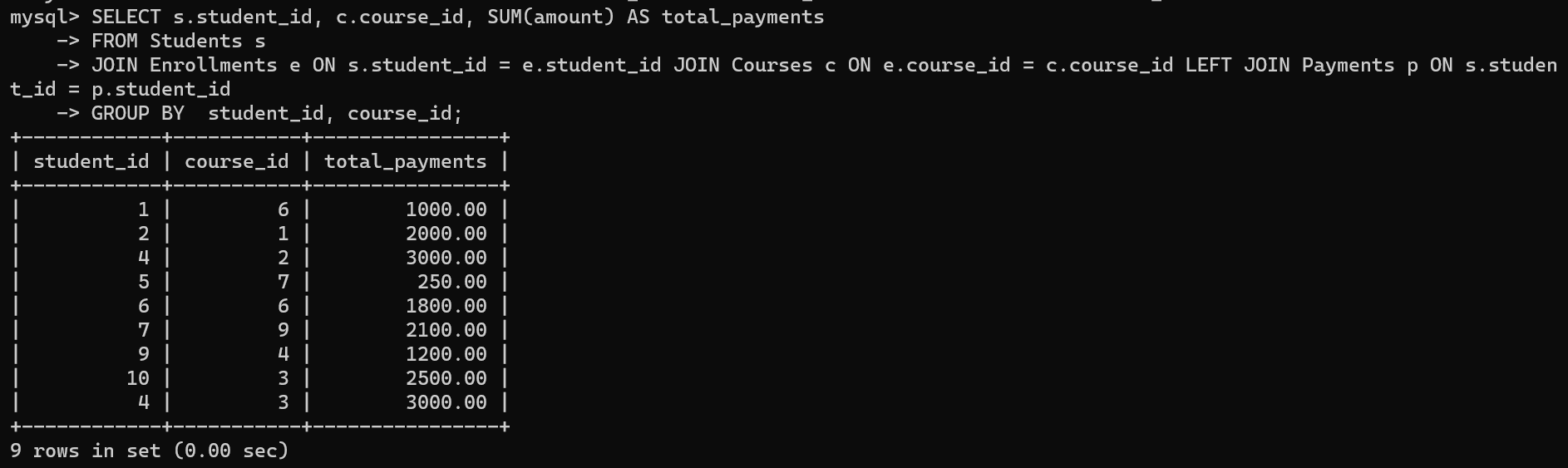
1. **Calculate the average age of all students. Use subqueries to calculate the age of each student based on their date of birth.**

****

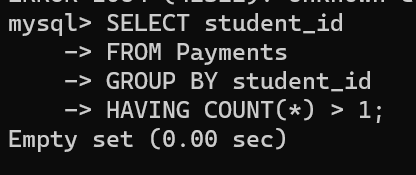
1. **Identify courses with no enrollments. Use subqueries to find courses without enrollment records.**

****

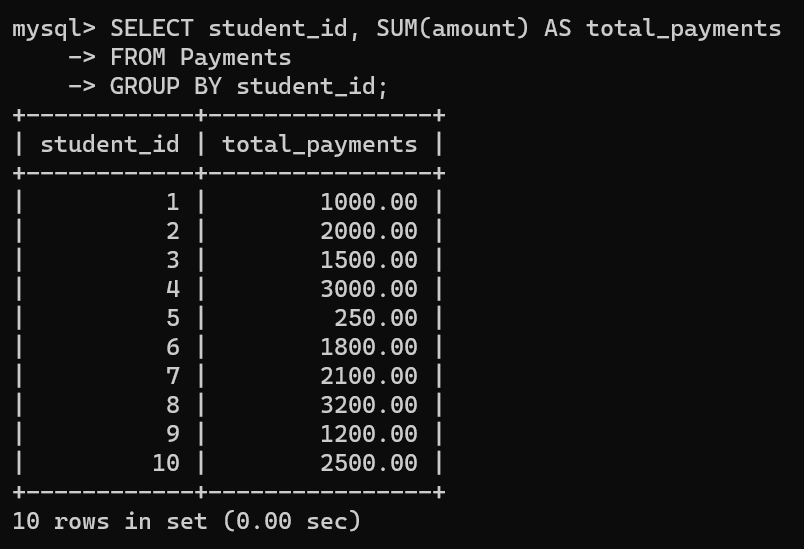
1. **Calculate the total payments made by each student for each course they are enrolled in. Use subqueries and aggregate functions to sum payments.**

****

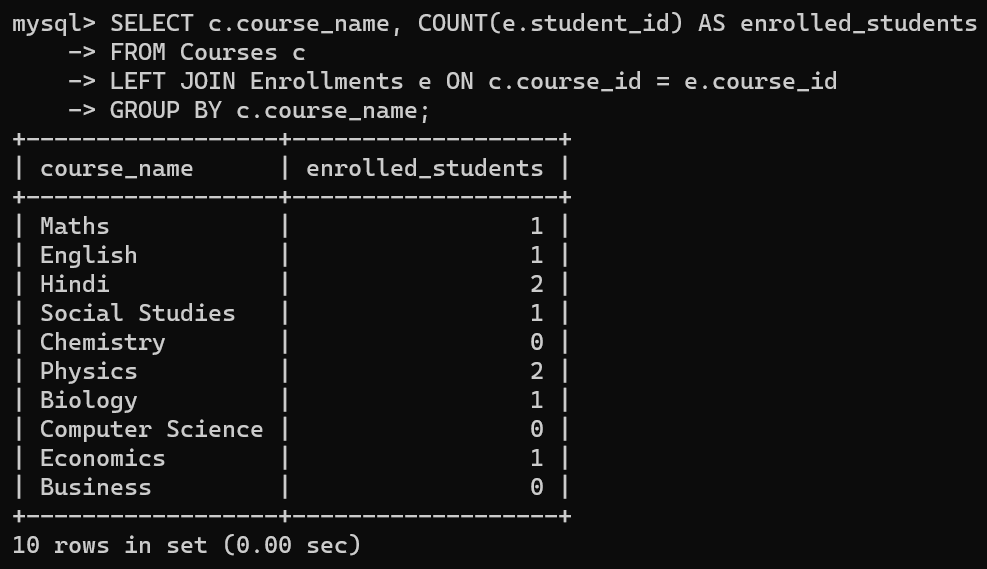
1. **Identify students who have made more than one payment. Use subqueries and aggregate functions to count payments per student and filter for those with counts greater than one.**

****

1. **Write an SQL query to calculate the total payments made by each student. Join the "Students" table with the "Payments" table and use GROUP BY to calculate the sum of payments for each student.**

****

1. **Retrieve a list of course names along with the count of students enrolled in each course. Use JOIN operations between the "Courses" table and the "Enrollments" table and GROUP BY to count enrollments.**

****

1. **Calculate the average payment amount made by students. Use JOIN operations between the "Students" table and the "Payments" table and GROUP BY to calculate the average.**

