**Name: Lavanya Mala A**

**Batch no: 6**

**Domain: Mysql**

**Assignment title: Assignment no 2 – Student Information System(SIS)**

1. Create the database named "SISDB

**create database SISDB;**

**use 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

**create table students(student\_ID int primary key,first\_name varchar(20),last\_name varchar(20),DOB date,email varchar(20),phone decimal(10,0));**

**create table courses(courseID int primary key,course\_name varchar(20),credits varchar(20),TeacherID int);**

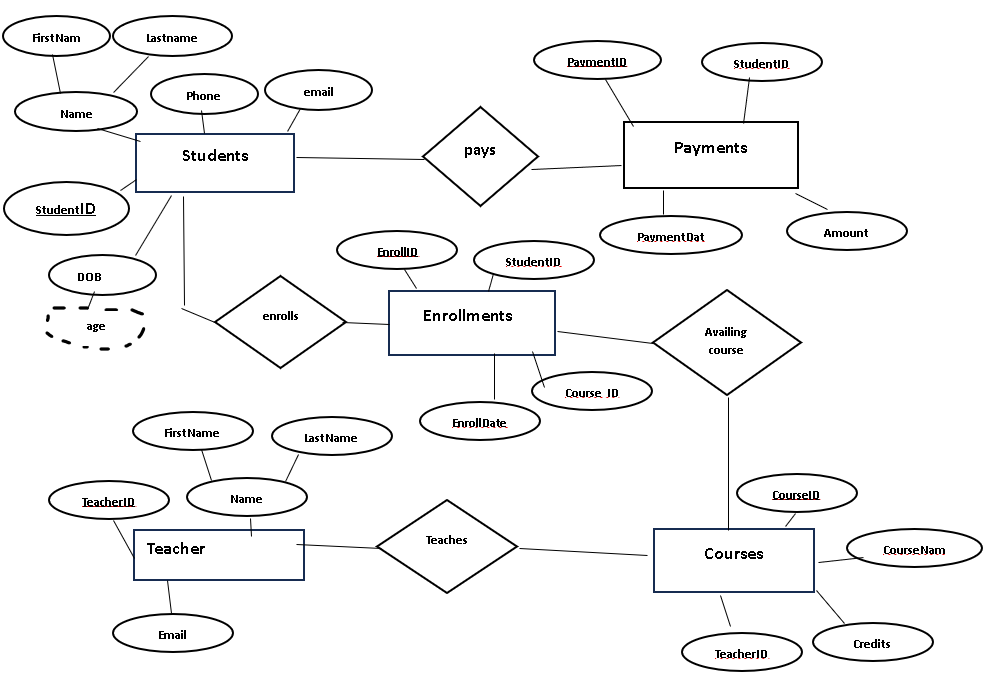
**create table enrollments ( enrollment\_id int primary key,student\_id int,courseid int,enrollment\_date date,foreign key (student\_id) references students(student\_id),foreign key (courseid) references courses(courseid) );**

**create table teacher(teacher\_id int primary key,first\_name varchar(20),last\_name varchar(20), email varchar(20));**

**alter table courses add constraint foreign key(teacher\_ID) references teacher(teacher\_ID);**

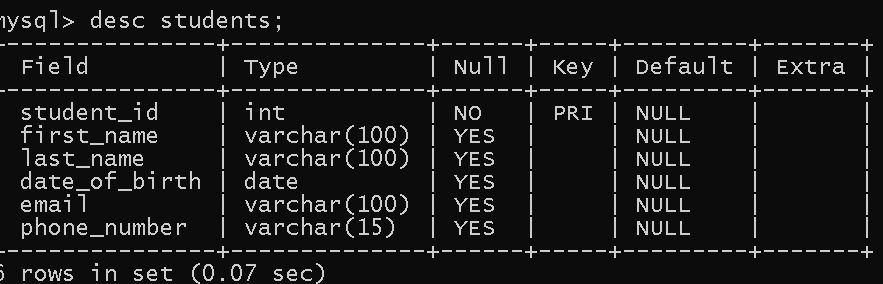
**create table payments(payment\_ID int primary key,student\_ID int,amount decimal(10,2),payment\_date date,foreign key(student\_ID) references students(student\_ID));**

3. Create an ERD (Entity Relationship Diagram) for the database

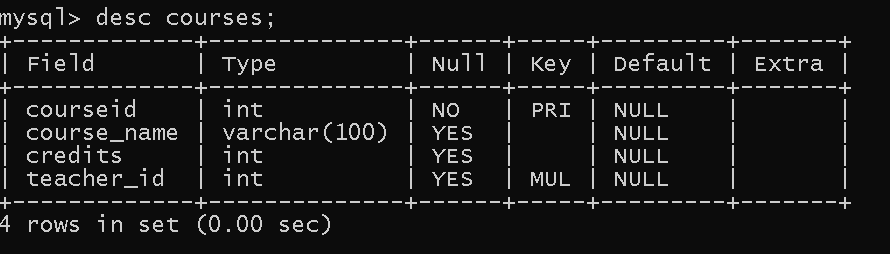
****

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

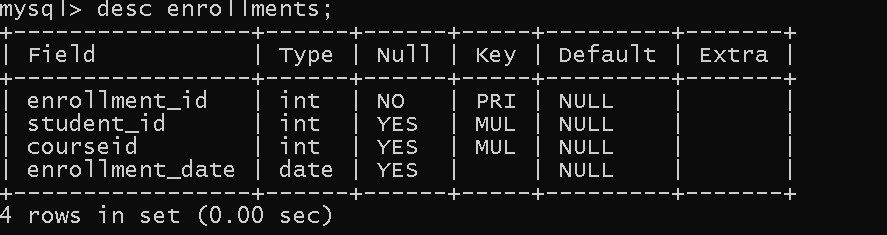
**desc students;**



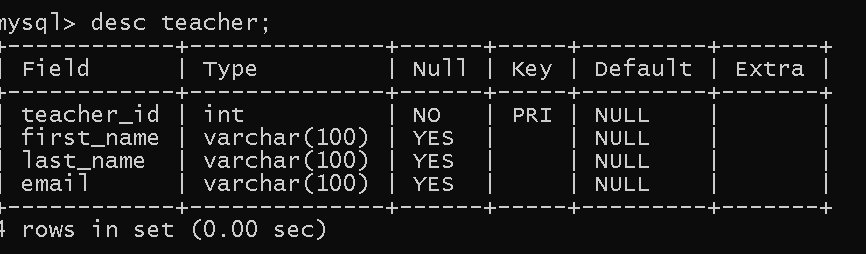
**mysql> desc courses;**



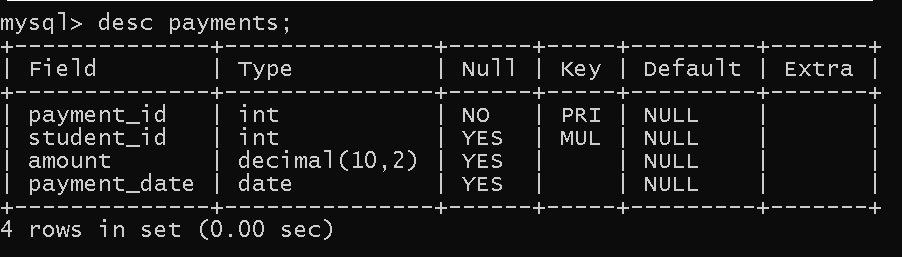
**mysql> desc enrollments;**



**mysql> desc teacher;**



**mysql> desc payments;**



5. Insert at least 10 sample records into each of the following tables.

i. Students

ii. Courses

iii. Enrollments

iv. Teacher

v. Payments

**mysql> INSERT INTO Students (student\_id, first\_name, last\_name, DOB, email, phone) VALUES**

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

**(101, 'John', 'Smith', '2000-01-01', 'john.smith@gmail.com', '8765489701'),**

**(102, 'Jane', 'Doe', '2000-02-02', 'jane.doe@gmail.com', '8765489702'),**

**(103, 'Alice', 'Johnson', '2000-03-03', 'alice.johnson@gmail.com', '8765489703'),**

**(104, 'Bob', 'Brown', '2000-04-04', 'bob.brown@gmail.com', '8765489704'),**

**(105, 'Mary', 'Davis', '2000-05-05', 'mary.davis@gmail.com', '8765489705'),**

**(106, 'Michael', 'Wilson', '2000-06-06', 'michael.wilson@gmail.com', '8765489706'),**

**(107, 'Elizabeth', 'Taylor', '2000-07-07', 'elizabeth.taylor@gmail.com', '8765489707'),**

**(108, 'David', 'Moore', '2000-08-08', 'david.moore@gmail.com', '8765489708'),**

**(109, 'Emma', 'White', '2000-09-09', 'emma.white@gmail.com', '8765489709'),(110, 'James', 'Anderson', '2000-10-10', 'james.anderson@gmail.com', '8765489710');**

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

**-> (401, 'Sarah', 'Connor', 'sarah.connor@gmail.com'),**

**-> (402, 'Bruce', 'Wayne', 'bruce.wayne@gmail.com'),**

**-> (403, 'Diana', 'Prince', 'diana.prince@gmail.com'),**

**-> (404, 'Clark', 'Kent', 'clark.kent@gmail.com'),**

**-> (405, 'Peter', 'Parker', 'peter.parker@gmail.com'),**

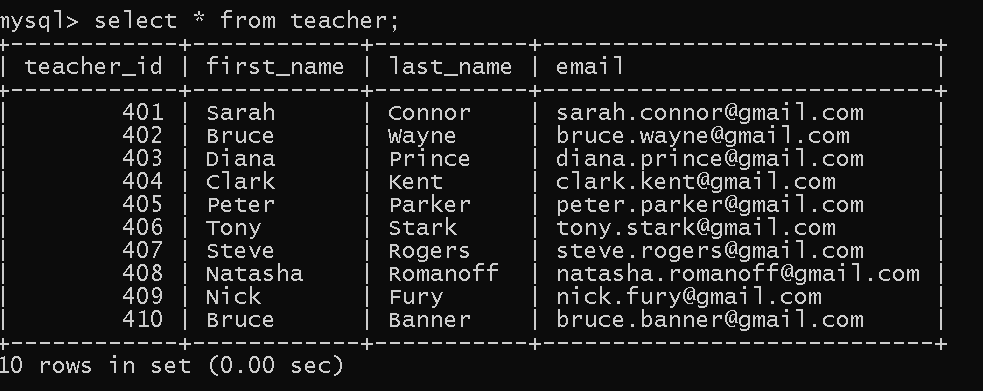
**-> (406, 'Tony', 'Stark', 'tony.stark@gmail.com'),**

**-> (407, 'Steve', 'Rogers', 'steve.rogers@gmail.com'),**

**-> (408, 'Natasha', 'Romanoff', 'natasha.romanoff@gmail.com'),**

**-> (409, 'Nick', 'Fury', 'nick.fury@gmail.com'),**

**-> (410, 'Bruce', 'Banner', 'bruce.banner@gmail.com');**



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

**-> (201, 'Mathematics', 3, 401),**

**-> (202, 'English Literature', 4, 402),**

**-> (203, 'Physics', 4, 403),**

**-> (204, 'Chemistry', 3, 404),**

**-> (205, 'Biology', 4, 405),**

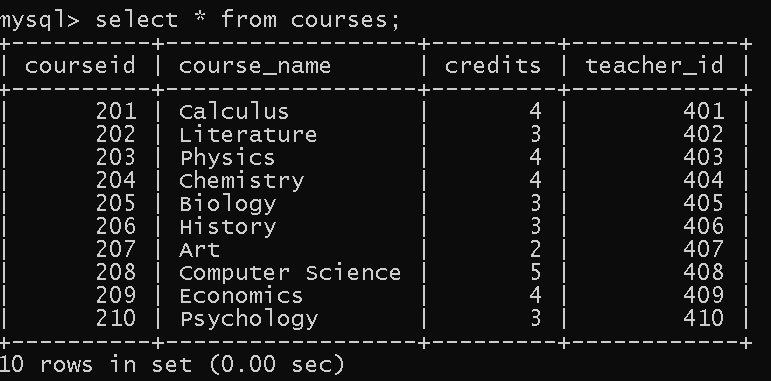
**-> (206, 'History', 3, 401),**

**-> (207, 'Art', 2, 402),**

**-> (208, 'Computer Science', 5, 403),**

**-> (209, 'Economics', 4, 404),**

**-> (210, 'Psychology', 3, 405);**



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

-> (301, 101, 201, '2023-01-20'),

-> (302, 102, 202, '2023-01-21'),

-> (303, 103, 203, '2023-01-22'),

-> (304, 104, 204, '2023-01-23'),

-> (305, 105, 205, '2023-01-24'),

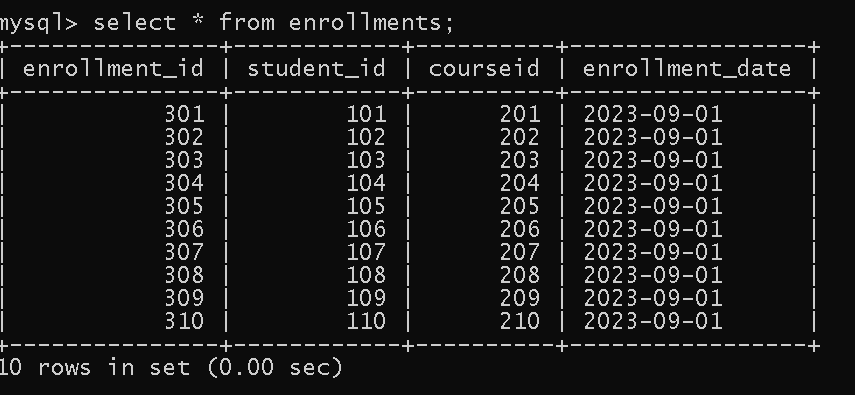
-> (306, 106, 206, '2023-01-25'),

-> (307, 107, 207, '2023-01-26'),

-> (308, 108, 208, '2023-01-27'),

-> (309, 109, 209, '2023-01-28'),

-> (310, 110, 210, '2023-01-29');



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

**-> (501, 101, 2000, '2023-02-01'),**

**-> (502, 102, 1200, '2023-02-02'),**

**-> (503, 103, 1500, '2023-02-03'),**

**-> (504, 104, 1800, '2023-02-04'),**

**-> (505, 105, 1000, '2023-02-05'),**

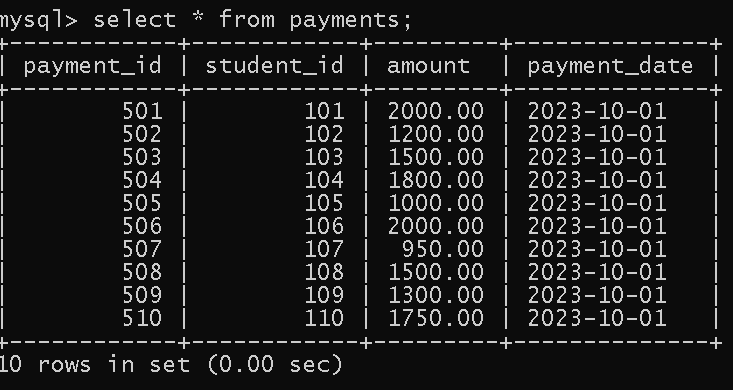
**-> (506, 106, 2000, '2023-02-06'),**

**-> (507, 107, 950, '2023-02-07'),**

**-> (508, 108, 1500, '2023-02-08'),**

**-> (509, 109, 1300, '2023-02-09'),**

**-> (510, 110, 1750, '2023-02-10');**



Task 2:

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@gmail.com

e. Phone Number: 1234567890

**insert into students values(111,"John","Doe","1995-08-15","john.doe@gmail.com",1234567890);**

**select \* from students;**

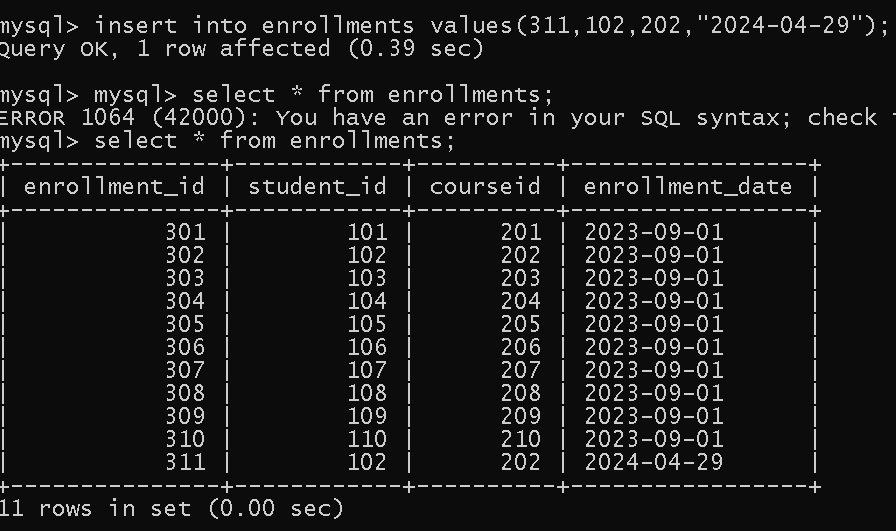


2.Write an SQL query to enroll a student in a course. Choose an existing student and course and

insert a record into the "Enrollments" table with the enrollment date.

**insert into enrollments values(311,102,202,"2024-04-29");**

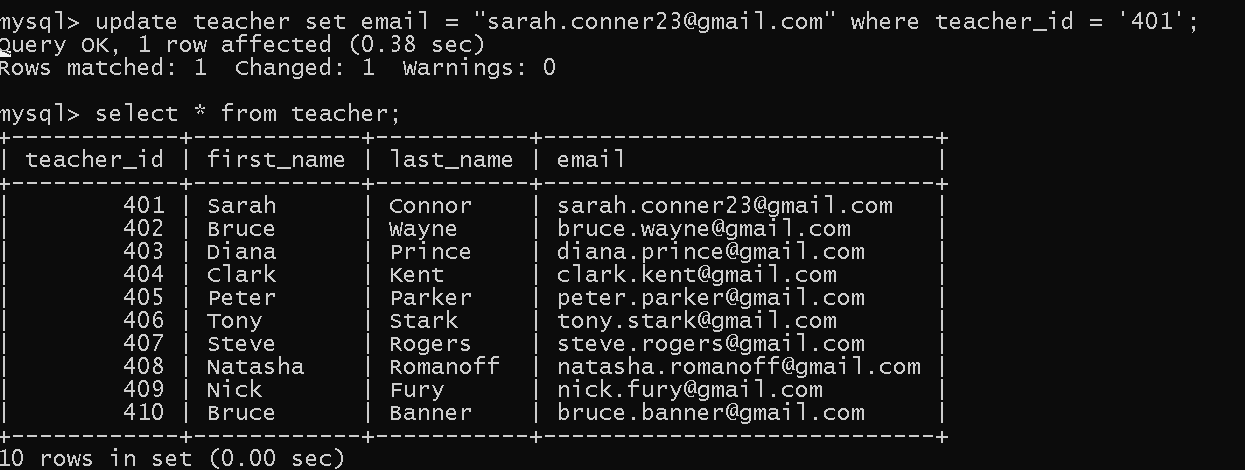
**mysql> select \* from enrollments;**



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

**update teacher set email = "sarah.conner23@gmail.com" where teacher\_id = '401';**

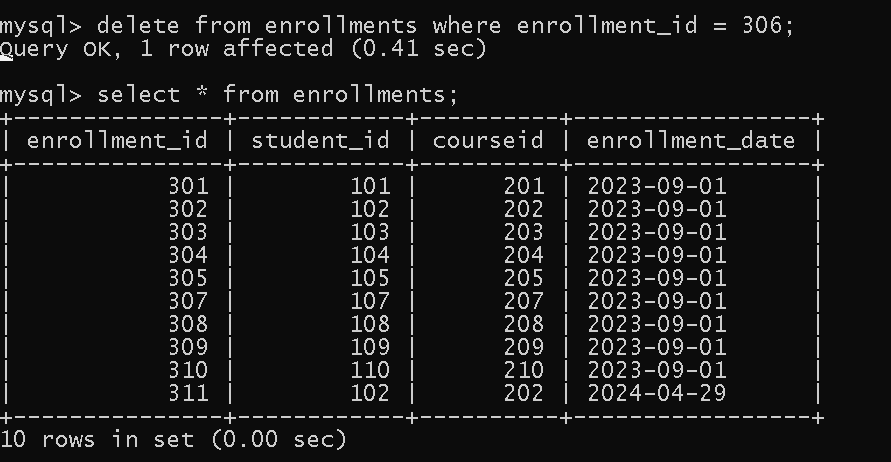
**select \* from teacher;**



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

**delete from enrollments where enrollment\_id = 306;**

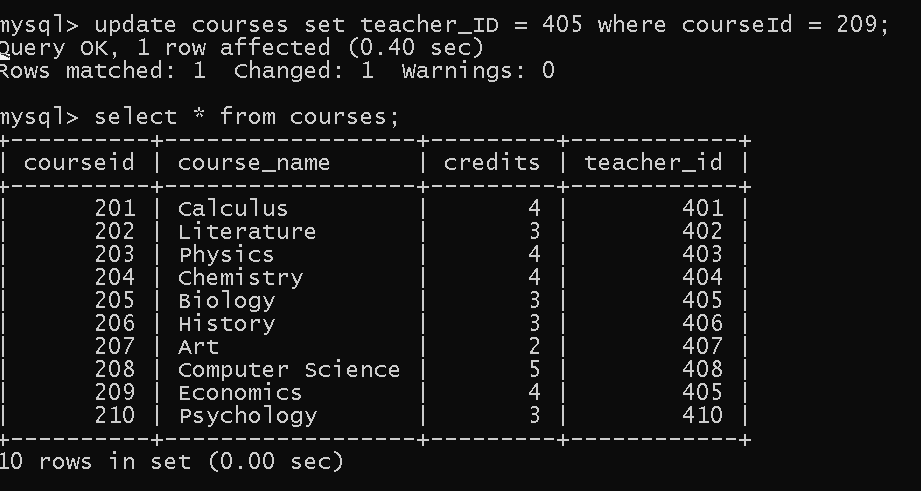
**select \* from enrollments;**



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

update courses set teacher\_ID = 405 where courseId = 209;

select \* from courses;



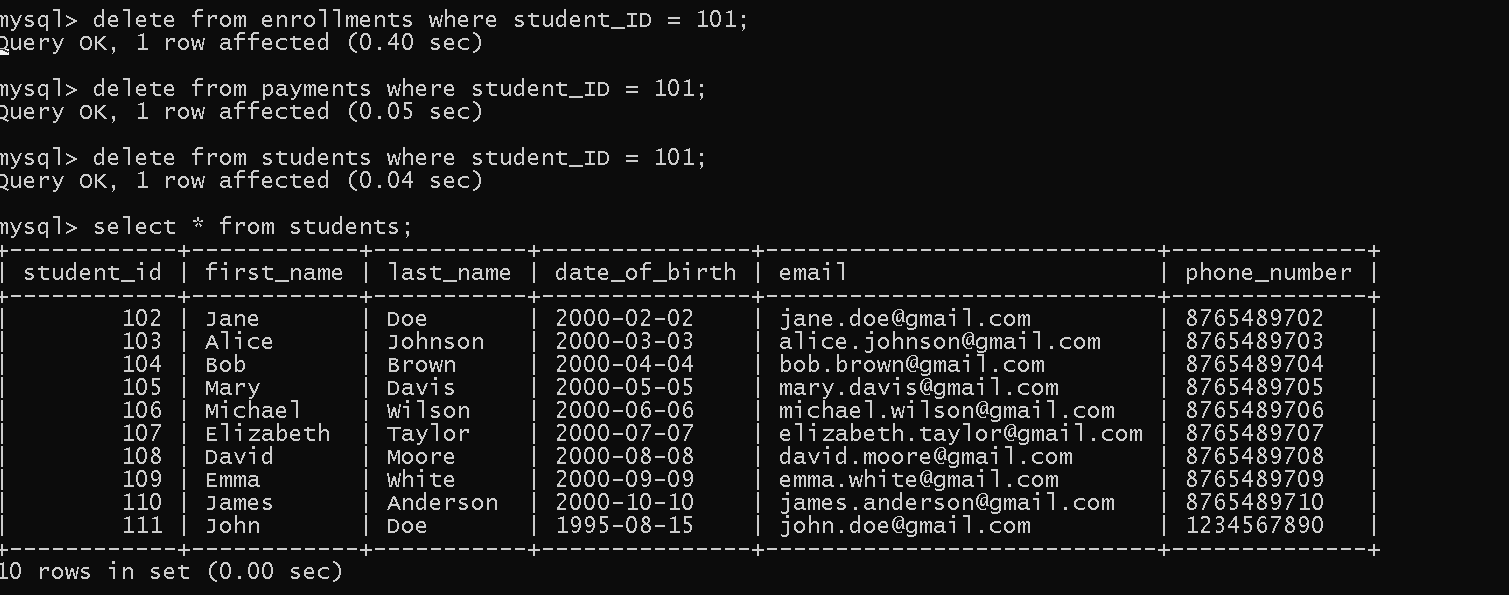
6. Delete a specific student from the "Students" table and remove all their enrollment records from the "Enrollments" table. Be sure to maintain referential integrity

**delete from enrollments where student\_ID = 101;**

**delete from payments where student\_ID = 101;**

**delete from students where student\_ID = 101;**

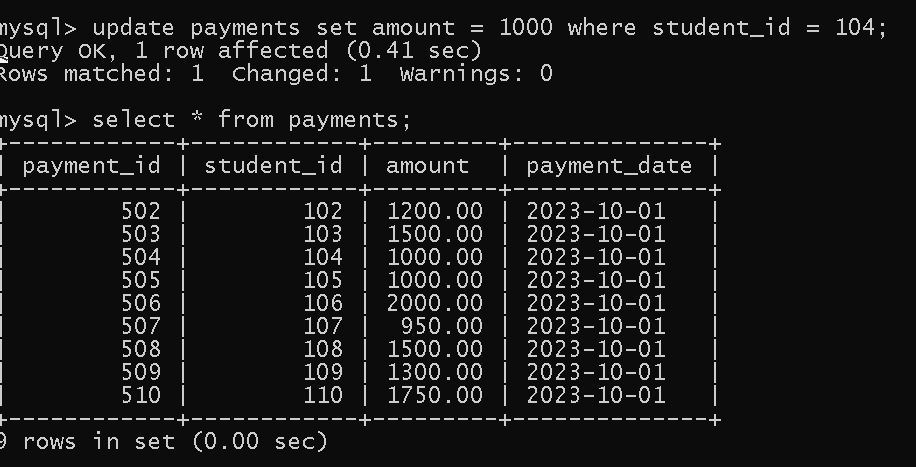
**select \* from students;**



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

**update payments set amount = 1000 where student\_id = 104;**

**select \* from payments;**



**Task 3:**

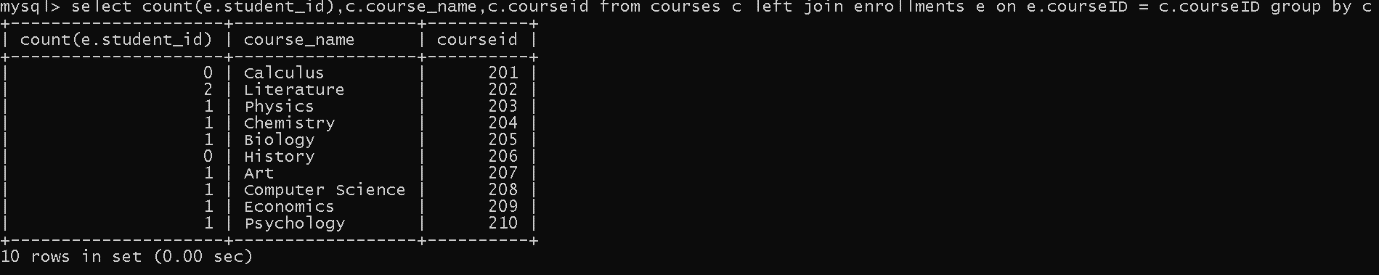
1. Write an SQL query to calculate the total payments made by a specific student. You will need to join the "Payments" table with the "Students" table based on the student's ID.

**select s.student\_ID,sum(p.amount),s.first\_name from payments p inner join students s on s.student\_id = p.student\_id group by s.student\_Id,s.first\_name;**



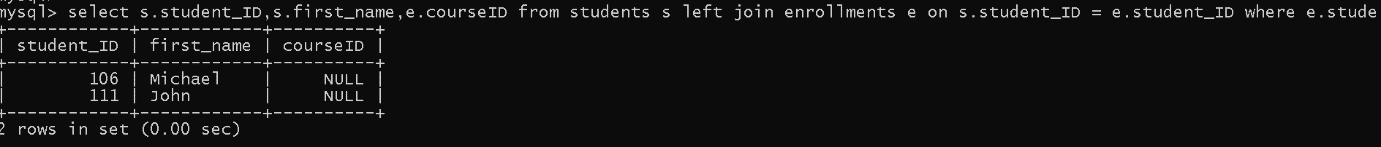
2.Write an SQL query to retrieve a list of courses along with the count of students enrolled in each course. Use a JOIN operation between the "Courses" table and the "Enrollments" table.

**select count(e.student\_id),c.course\_name,c.courseid from courses c left join enrollments e on e.courseID = c.courseID group by courseID;**

****

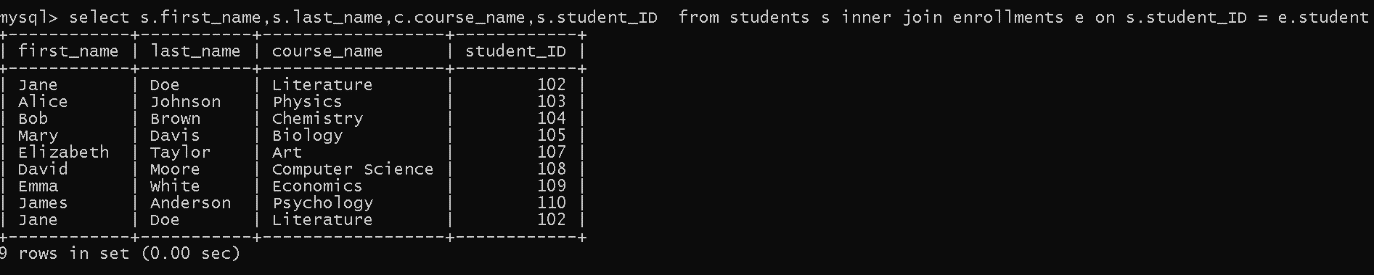
3.Write an SQL query to find the names of students who have not enrolled in any course. Use a LEFT JOIN between the "Students" table and the "Enrollments" table to identify students without enrollments

**select s.student\_ID,s.first\_name,e.courseID from students s left join enrollments e on s.student\_ID = e.student\_ID where e.student\_id is null;**

****

4.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.

**select s.first\_name,s.last\_name,c.course\_name,s.student\_ID from students s inner join enrollments e on s.student\_ID = e.student\_ID inner join courses c on e.courseID = c.courseID;**

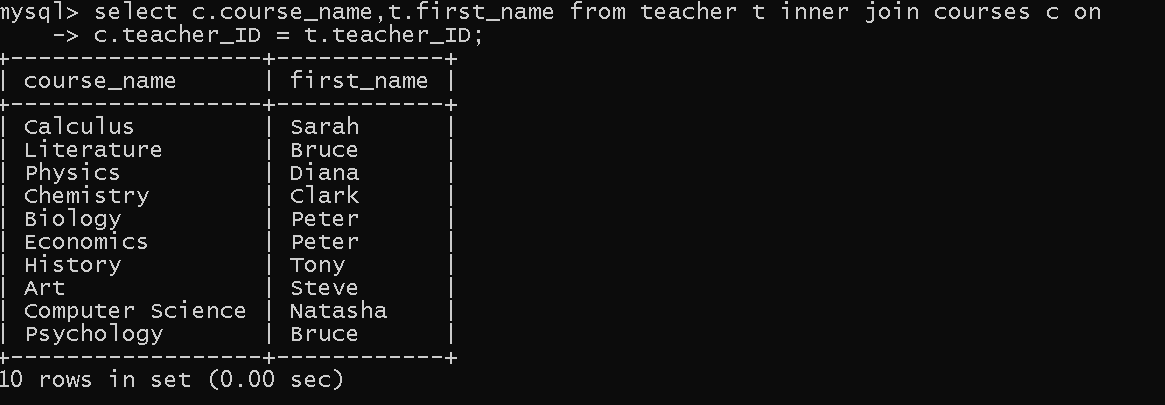


5.Create a query to list the names of teachers and the courses they are assigned to. Join the

"Teacher" table with the "Courses" table.

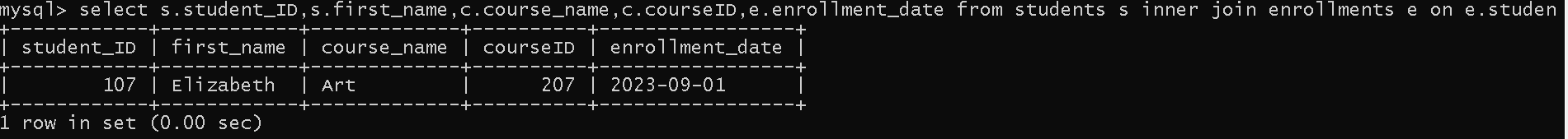
**select c.course\_name,t.first\_name from teacher t inner join courses c on**

**c.teacher\_ID = t.teacher\_ID;**



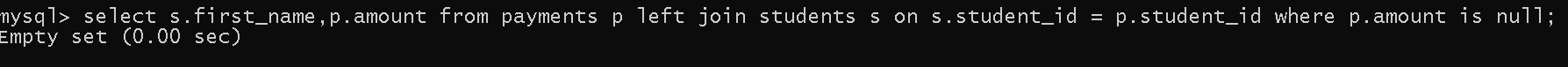
6. Retrieve a list of students and their enrollment dates for a specific course. You'll need to join the "Students" table with the "Enrollments" and "Courses" tables.

**select s.student\_ID,s.first\_name,c.course\_name,c.courseID,e.enrollment\_date from students s inner join enrollments e on e.student\_id = s.student\_id inner join courses c on e.courseID = c.courseID where course\_name = "Art";**



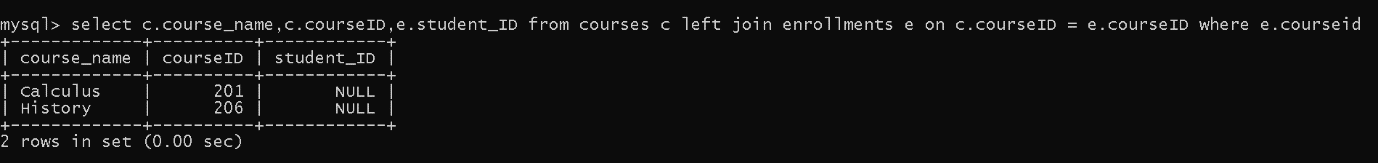
7. Find the names of students who have not made any payments. Use a LEFT JOIN between the "Students" table and the "Payments" table and filter for students with NULL payment records.

**select s.first\_name,p.amount from payments p left join students s on s.student\_id = p.student\_id where p.amount is null;**



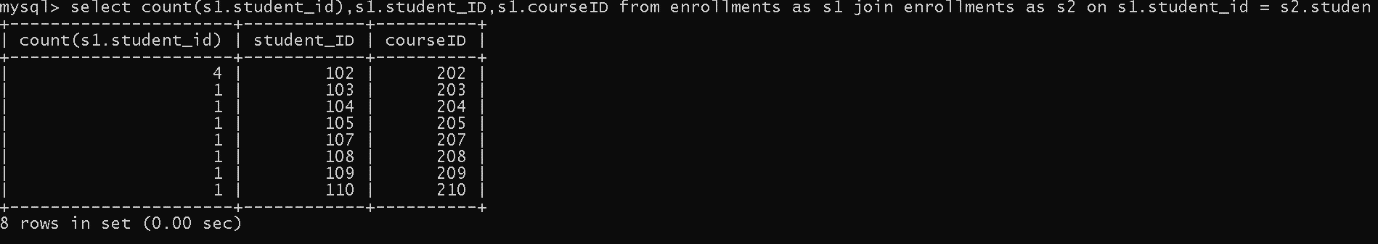
8. Write a query to identify courses that have no enrollments. You'll need to use a LEFT JOIN between the "Courses" table and the "Enrollments" table and filter for courses with NULL enrollment records.

**select c.course\_name,c.courseID,e.student\_ID from courses c left join enrollments e on c.courseID = e.courseID where e.courseid is null;**



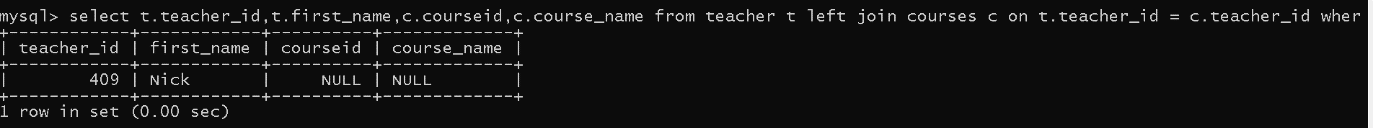
9. Identify students who are enrolled in more than one course. Use a self-join on the "Enrollments" table to find students with multiple enrollment records.

**select count(s1.student\_id),s1.student\_ID,s1.courseID from enrollments as s1 join enrollments as s2 on s1.student\_id = s2.student\_id group by s1.courseID,s1.student\_ID;**



10. 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.

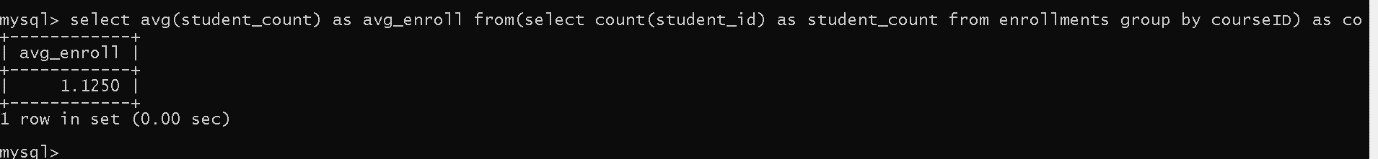
**select t.teacher\_id,t.first\_name,c.courseid,c.course\_name from teacher t left join courses c on t.teacher\_id = c.teacher\_id where courseid is null;**



**Task 4:**

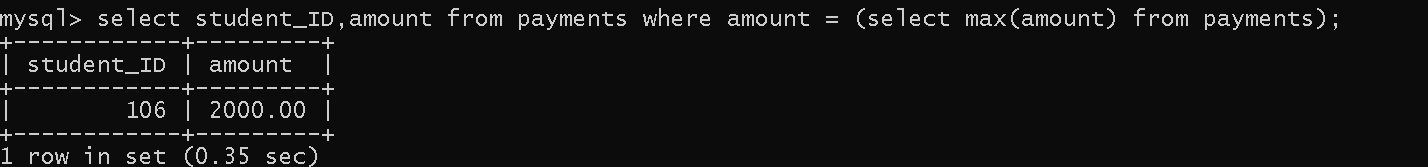
1.Write an SQL query to calculate the average number of students enrolled in each course. Use aggregate functions and subqueries to achieve this.

**select avg(student\_count) as avg\_enroll from(select count(student\_id) as student\_count from enrollments group by courseID) as coursegroup;**



2.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.

**select student\_ID,amount from payments where amount = (select max(amount) from payments);**



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

**select c.courseid, c.course\_name, count(e.student\_id) as numberofenrollments**

**from courses c**

**join enrollments e on c.courseid = e.courseid**

**group by c.courseid, c.course\_name having count(e.student\_id) = (select max(totalenrollments) from (**

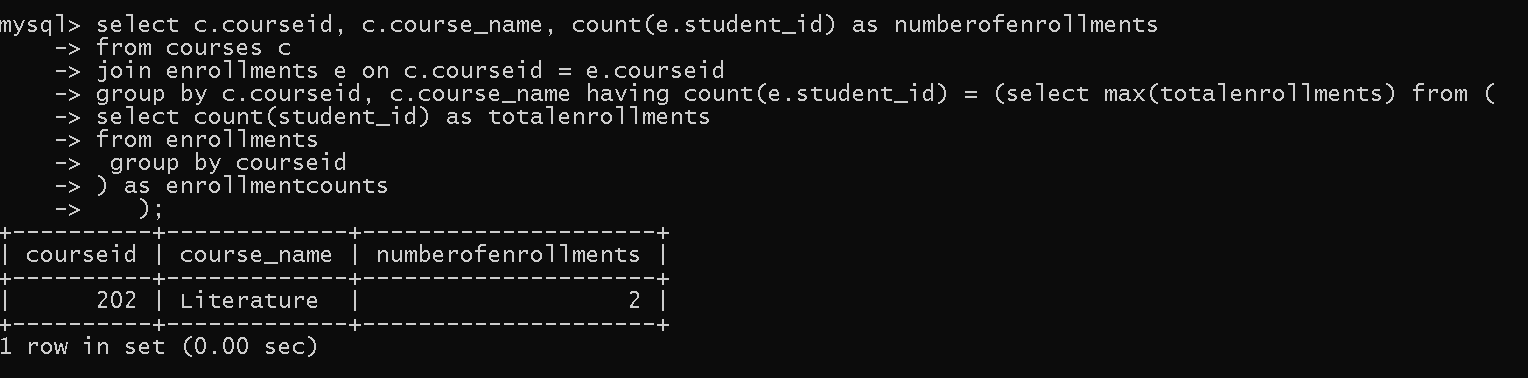
**select count(student\_id) as totalenrollments**

**from enrollments**

**group by courseid**

**) as enrollmentcounts**

**);**



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

**select t.teacher\_id,t.first\_name,t.last\_name,**

**coalesce(sum(p.amount), 0) as totalpayments from**

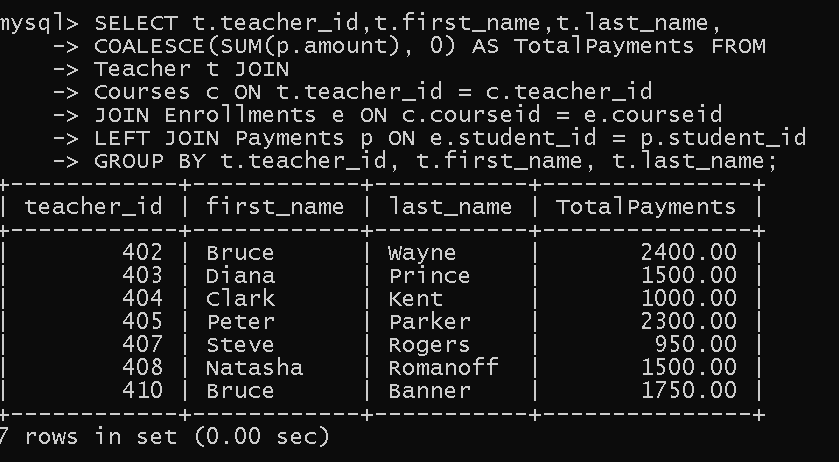
**teacher t join**

**courses c on t.teacher\_id = c.teacher\_id**

**join enrollments e on c.courseid = e.courseid**

**left join payments p on e.student\_id = p.student\_id**

**group by t.teacher\_id, t.first\_name, t.last\_name;**



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

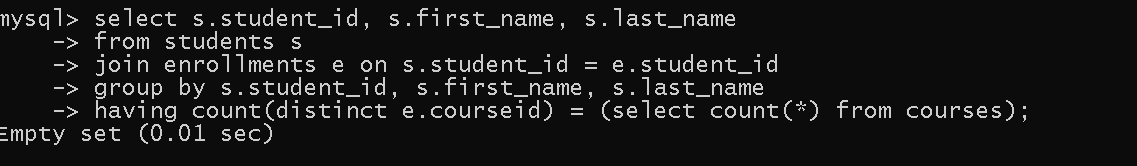
**select s.student\_id, s.first\_name, s.last\_name**

**from students s**

**join enrollments e on s.student\_id = e.student\_id**

**group by s.student\_id, s.first\_name, s.last\_name**

**having count(distinct e.courseid) = (select count(\*) from courses);**

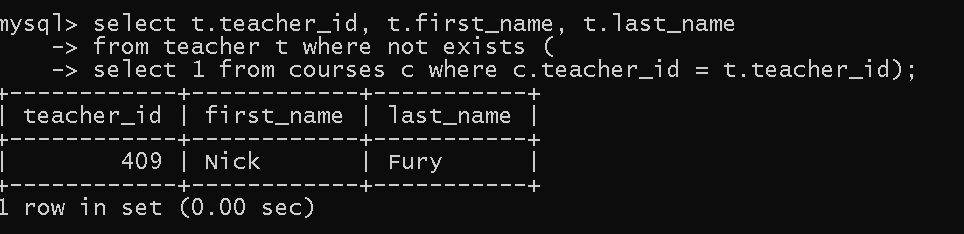


6.Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to find teachers with no course assignments.

**select t.teacher\_id, t.first\_name, t.last\_name**

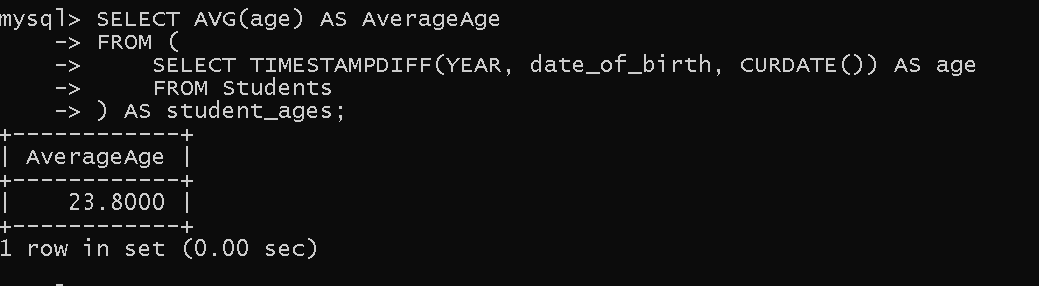
**from teacher t where not exists (**

**select 1 from courses c where c.teacher\_id = t.teacher\_id);**



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

**select avg(age) as averageage from (select timestampdiff(year, date\_of\_birth, curdate()) as age from students ) as student\_ages;**

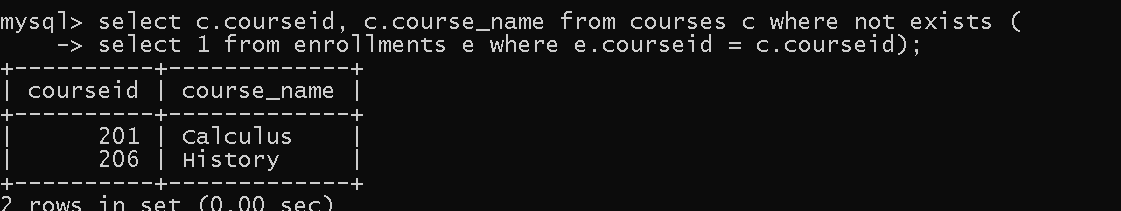


8.Identify courses with no enrollments. Use subqueries to find courses without enrollment

records.

**select c.courseid, c.course\_name from courses c where not exists (**

**select 1 from enrollments e where e.courseid = c.courseid);**

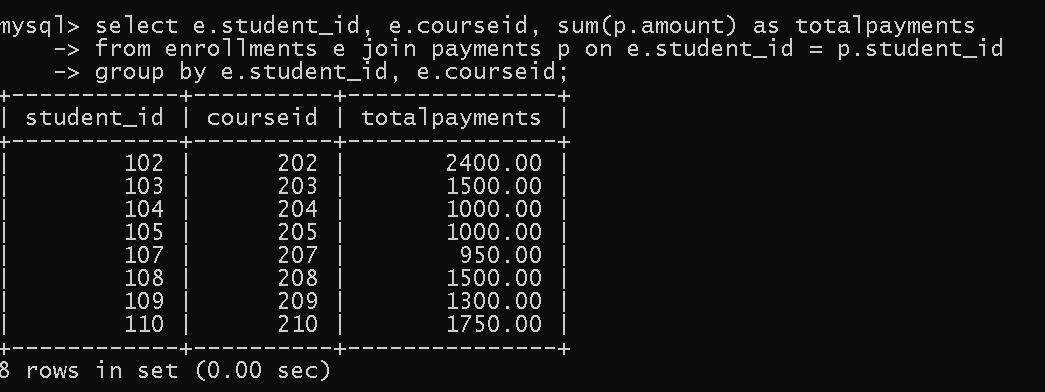


9.Calculate the total payments made by each student for each course they are enrolled in. Use subqueries and aggregate functions to sum payments.

**select e.student\_id, e.courseid, sum(p.amount) as totalpayments**

**from enrollments e join payments p on e.student\_id = p.student\_id**

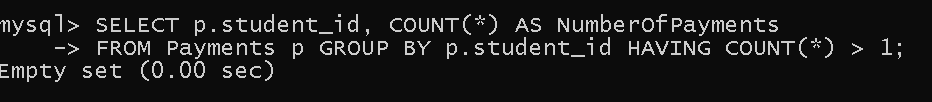
**group by e.student\_id, e.courseid;**



10.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.

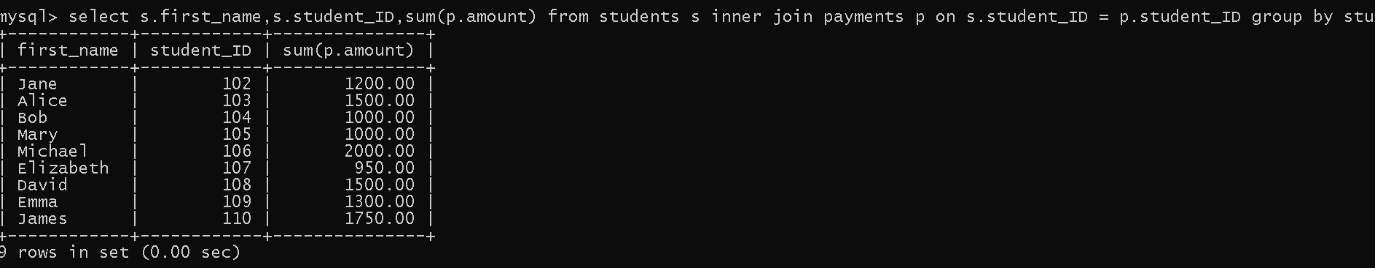
**select p.student\_id, count(\*) as numberofpayments**

**from payments p group by p.student\_id having count(\*) > 1;**



11.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

**select s.first\_name,s.student\_ID,sum(p.amount) from students s inner join payments p on s.student\_ID = p.student\_ID group by student\_id;**

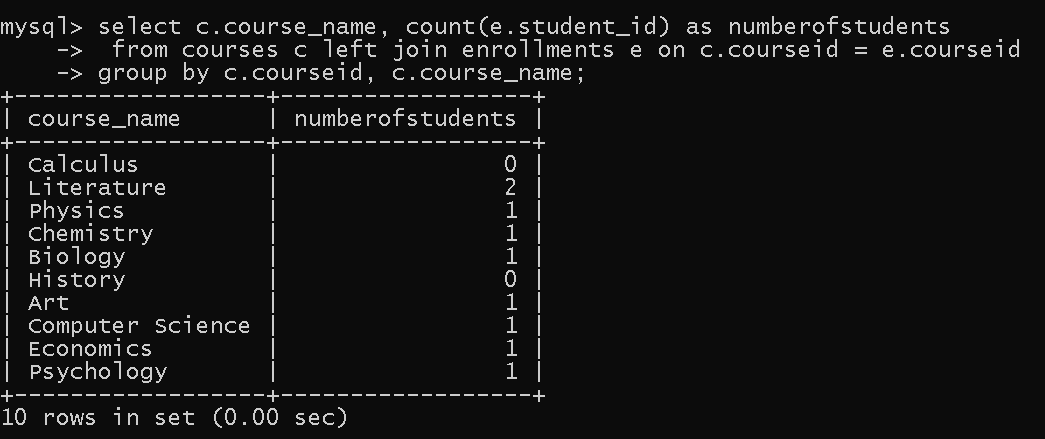


12. Retrieve a list of course names along with the count of students enrolled in each course. Use JOIN operations between the "Courses" table and the "Enrollments" table and GROUP BY to count enrollments.

**select c.course\_name, count(e.student\_id) as numberofstudents**

**from courses c left join enrollments e on c.courseid = e.courseid**

**group by c.courseid, c.course\_name;**



13. 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

**select s.student\_id, s.first\_name, avg(p.amount) from payments p inner join students s on p.student\_id = s.student\_id group by student\_id;**

