**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.**

**Students table:-** create table student(

student\_id int primary key,

first\_name varchar(15) not null,

last\_name varchar(15),

date\_of\_birth varchar(15),

email varchar(30),

phone\_number bigint

);

**Teacher table:-** create table teacher(

teacher\_id int primary key,

first\_name varchar(20) not null,

last\_name varchar(20) not null,

email varchar(50));

**Courses table:-** create table courses(

course\_id int primary key,

course\_name varchar(20) not null,

credits int,

teacher\_id int not null,

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

);

**Enrollments table:-** create table enrollments(

enrollment\_id int primary key,

student\_id int not null,

course\_id int not null,

enrollment\_date varchar(15),

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

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

);

**Payments table:-** create table payments(

payment\_id int primary key,

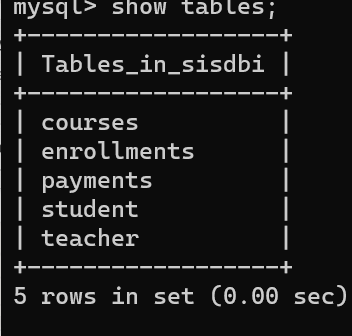
student\_id int not null,

amount varchar(15) not null,

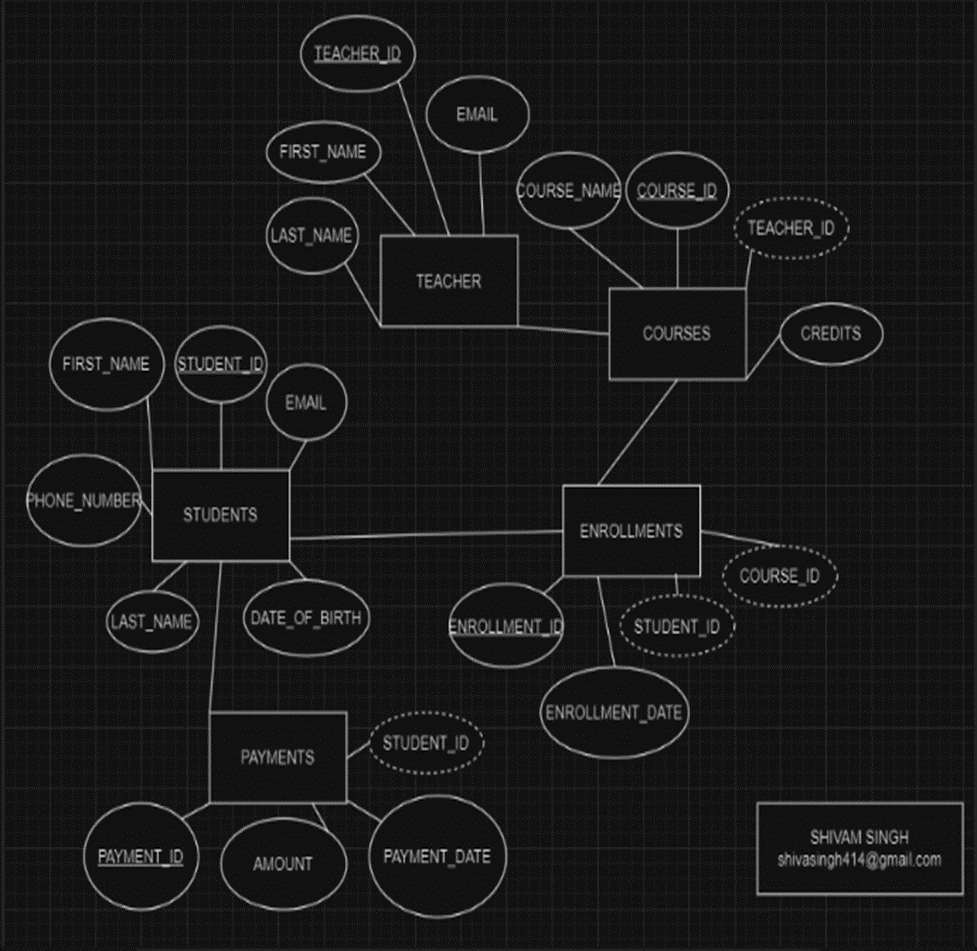
payment\_date varchar(15),

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

);



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



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

insert into student(student\_id,first\_name,last\_name,date\_of\_birth,email,phone\_number)

values(1001,"Gautam","Sharma","08/05/2001","kundragautam007@gmail.com",1000000001),

(1002,"Aniket","Gaur",'10/10/2001',"ag@gmail.com",1000000002),

(1003,"Deepak","Singh",'10/01/2000',"dds@gmail.com",1000000003),

(1004,"Madhu","Kumari",'18/07/2003',"mk@gmail.com",1000000004),

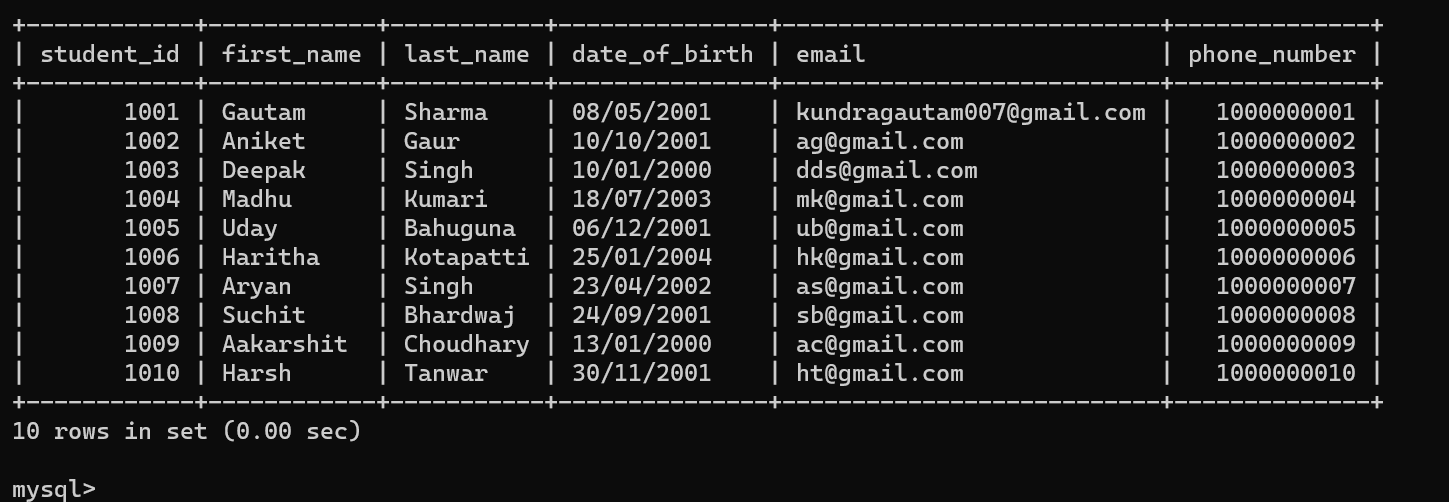
(1005,"Uday","Bahuguna",'06/12/2001',"ub@gmail.com",1000000005),

(1006,"Haritha","Kotapatti",'25/01/2004',"hk@gmail.com",1000000006),

(1007,"Aryan","Singh",'23/04/2002',"as@gmail.com",1000000007),

(1008,"Suchit","Bhardwaj",'24/09/2001',"sb@gmail.com",1000000008),

(1009,"Aakarshit","Choudhary",'13/01/2000',"ac@gmail.com",1000000009), (1010,"Harsh","Tanwar",'30/11/2001',"ht@gmail.com",1000000010);



insert into teacher(teacher\_id,first\_name,last\_name,email)

values(101,"Gautam","Sharma","kundragautam007@gmail.com"),

(102,"Ankita","Tripathi","at@gmail.com"),

(103,"Varun","Singh","vs@gmail.com"),

(104,"Roshini","Singh","rs@gmail.com"),

(105,"Dev","Lamba","dl@gmail.com"),

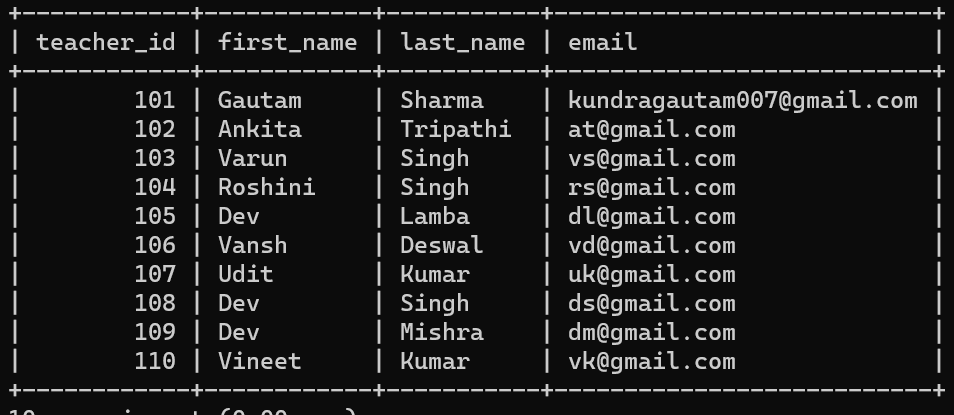
(106,"Vansh","Deswal","vd@gmail.com"),

(107,"Udit","Kumar","uk@gmail.com"),

(108,"Dev","Singh","ds@gmail.com"),

(109,"Dev","Mishra","dm@gmail.com"),

(110,"Vineet","Kumar","vk@gmail.com");



insert into courses(course\_id,course\_name,credits,teacher\_id)

values

(10001,"Java",10,101),

(10002,"C++",5,102),

(10003,"C",5,103),

(10004,"Python",5,104),

(10005,"English",3,105),

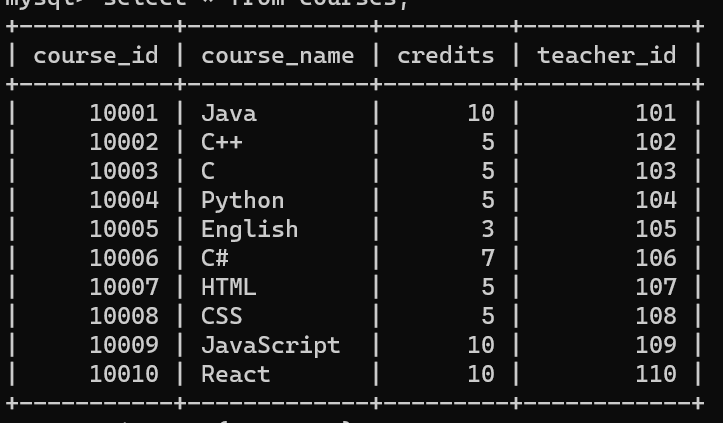
(10006,"C#",7,106),

(10007,"HTML",5,107),

(10008,"CSS",5,108),

(10009,"JavaScript",10,109),

(10010,"React",10,110);



insert into enrollments(enrollment\_id,student\_id,course\_id,enrollment\_date)

values(901,1001,10001,"10-05-2023"),

(902,1002,10002,"19-02-2022"),

(903,1003,10003,"20-05-2021"),

(904,1004,10004,"30-01-2023"),

(905,1005,10005,"15-10-2023"),

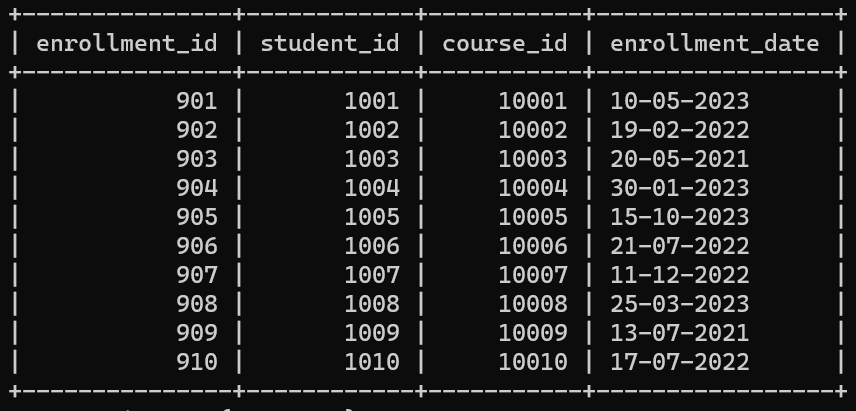
(906,1006,10006,"21-07-2022"),

(907,1007,10007,"11-12-2022"),

(908,1008,10008,"25-03-2023"),

(909,1009,10009,"13-07-2021"),

(910,1010,10010,"17-07-2022");



insert into payments(payment\_id,student\_id,amount,payment\_date)

values(5001,1001,"Rs. 10000","10-05-2023"),

(5002,1002,"Rs. 2500","19-02-2022"),

(5003,1003,"Rs. 3000","20-05-2021"),

(5004,1004,"Rs. 5000","30-01-2023"),

(5005,1005,"Rs. 4500","15-10-2023"),

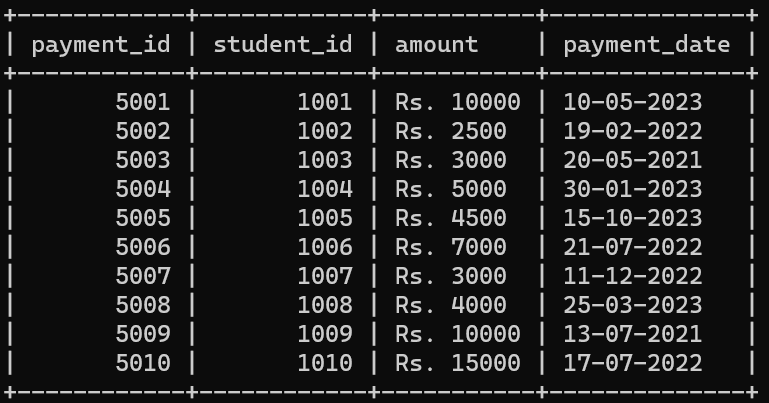
(5006,1006,"Rs. 7000","21-07-2022"),

(5007,1007,"Rs. 3000","11-12-2022"),

(5008,1008,"Rs. 4000","25-03-2023"),

(5009,1009,"Rs. 10000","13-07-2021"),

(5010,1010,"Rs. 15000","17-07-2022");



**Task 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**

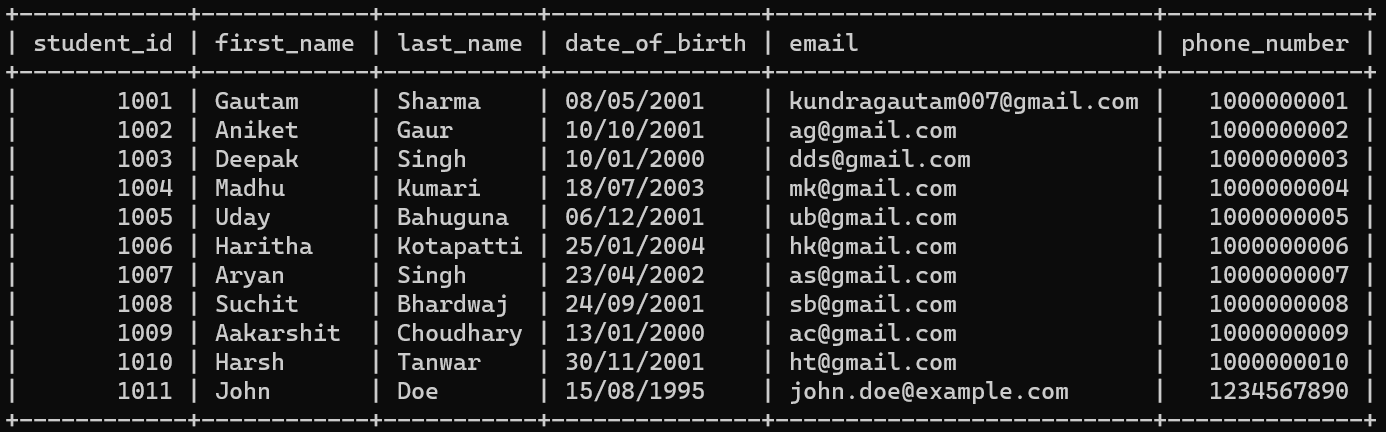
**c. Date of Birth: 1995-08-15**

**d. Email: john.doe@example.com**

**e. Phone Number: 1234567890**

insert into student(student\_id,first\_name,last\_name,date\_of\_birth,email,phone\_number)

values(1011,"John","Doe","15/08/1995","john.doe@example.com",1234567890);



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

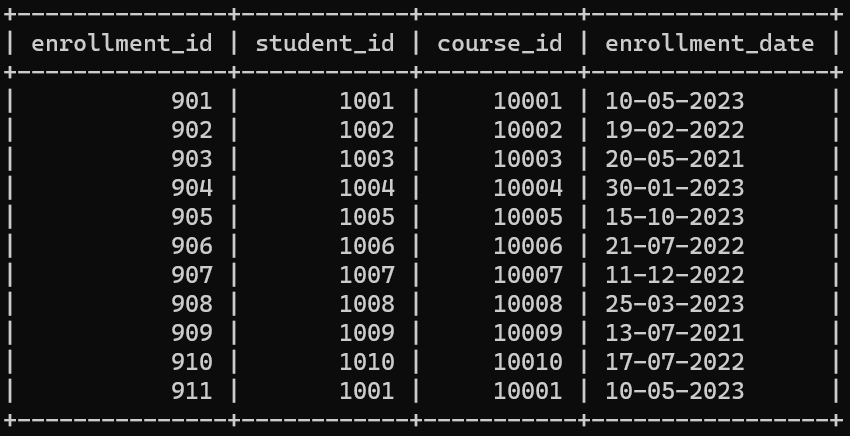
**insert a record into the "Enrollments" table with the enrollment date.**

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

VALUES(911,

(SELECT student\_id FROM Student WHERE first\_name = 'Gautam'),

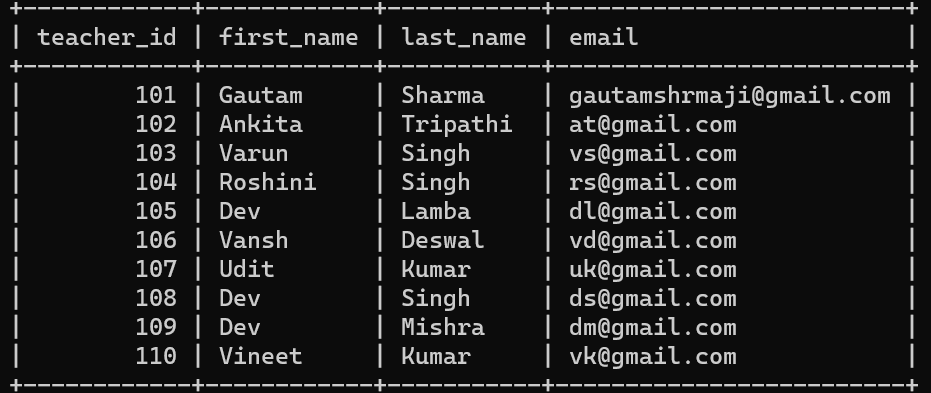
(SELECT course\_id FROM Courses WHERE course\_name = 'Java'),'10-05-2023');



**3:-Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and**

**modify their email address.**

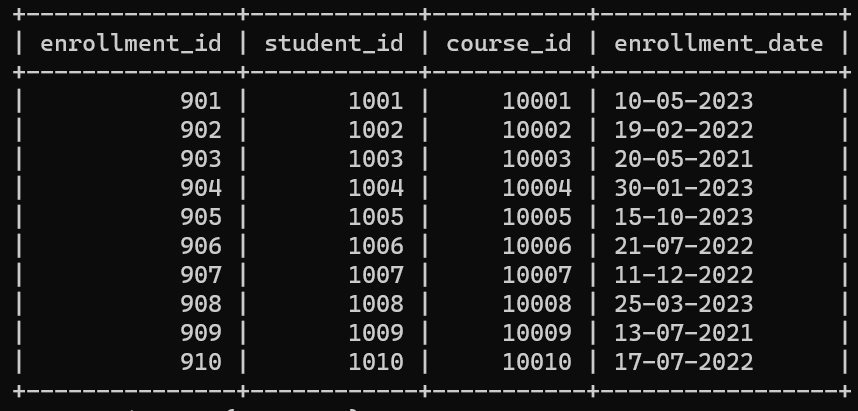
update teacher set email="gautamshrmaji@gmail.com" where teacher\_id=101;



**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.**

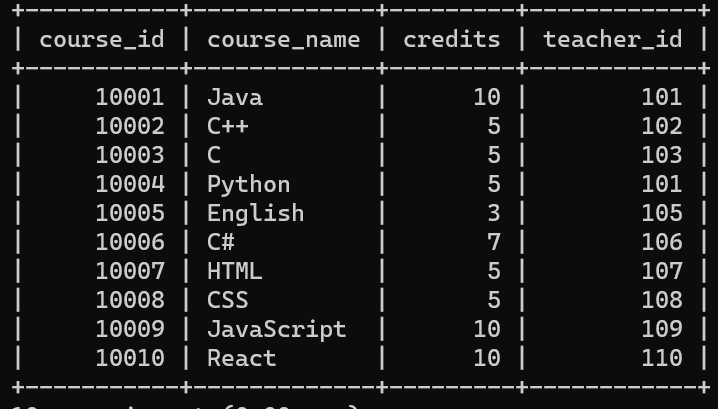
delete from enrollments where enrollment\_id=911;



**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=(select teacher\_id from teacher where first\_name="Gautam")where course\_id=10004;



**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 student where student\_id=1011;

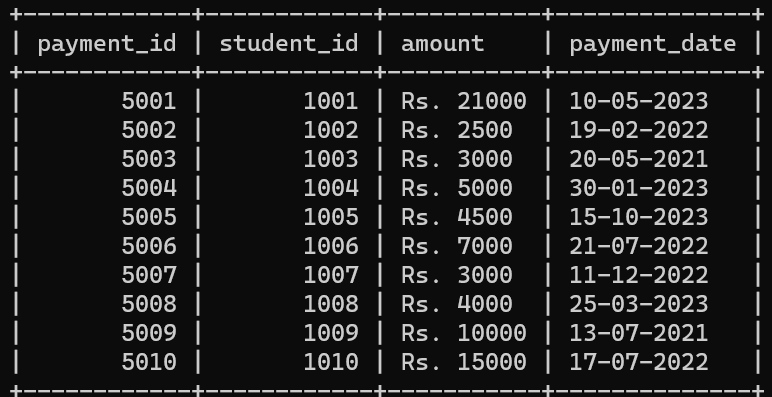
truncate table enrollments;



**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="Rs. 21000" where student\_id=1001;



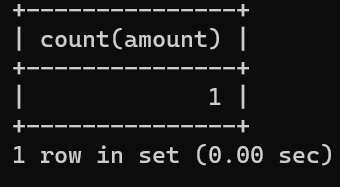
**Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:**

**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.**

select count(amount) from payments inner join student

on payments.student\_id=student.student\_id

where student.student\_id=1001;



**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 courses.course\_id, courses.course\_name, COUNT(enrollments.student\_id) as enrolled\_students\_count from courses join enrollments

on courses.course\_id = enrollments.course\_id

group by courses.course\_id;



**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 student.student\_id, student.first\_name, student.last\_name from student

left join enrollments on student.student\_id=enrollments.student\_id

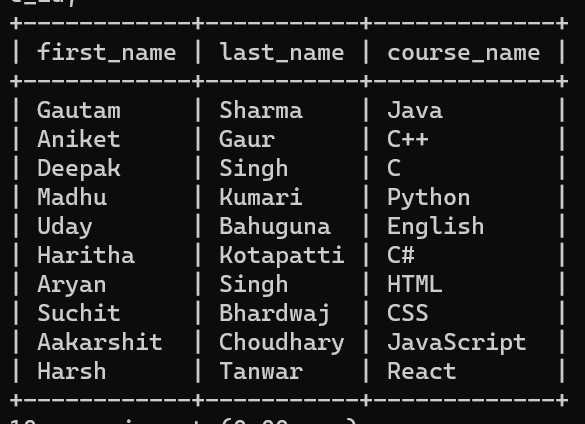
where enrollments.student\_id=null;

**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 student.first\_name, student.last\_name, courses.course\_name from student

inner join enrollments on student.student\_id=enrollments.student\_id

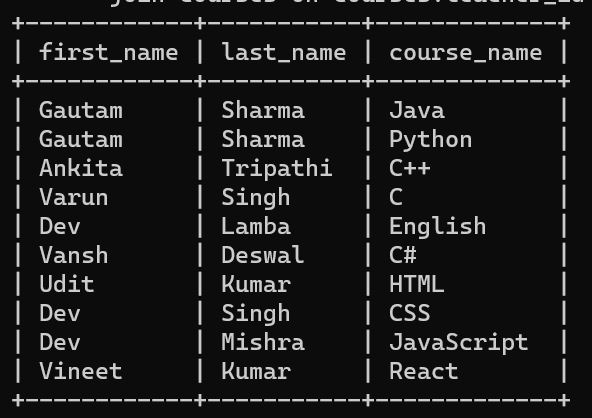
inner join courses on enrollments.course\_id=courses.course\_id;



**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 teacher.first\_name, teacher.last\_name, courses.course\_name from teacher

join courses on courses.teacher\_id=teacher.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.**

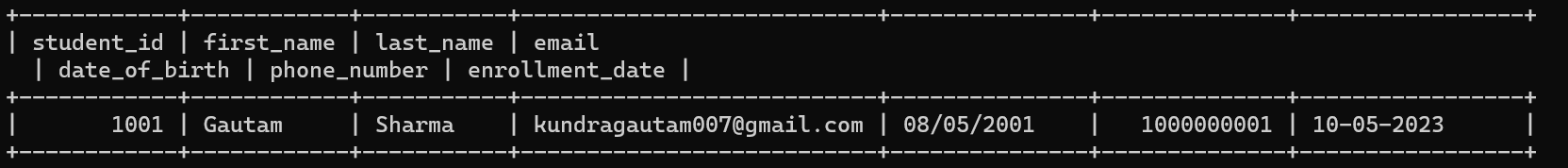
select student.student\_id, student.first\_name, student.last\_name,

student.email, student.date\_of\_birth,student.phone\_number, enrollments.enrollment\_date

from student join enrollments on student.student\_id=enrollments.student\_id

join courses on enrollments.course\_id=courses.course\_id

where enrollments.course\_id=10001;



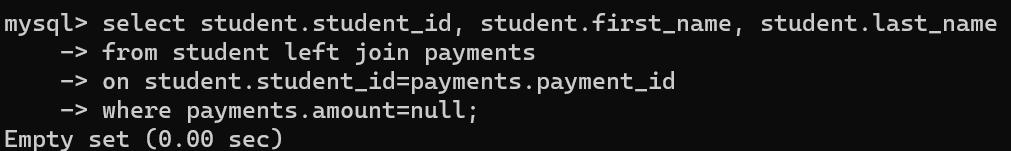
**7:- Find the names of students who have not made any payments. Use a LEFT JOIN between the "Students" table and the "Payments" table and filter for students with NULL payment records.**

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

from student left join payments

on student.student\_id=payments.payment\_id

where payments.amount=null;

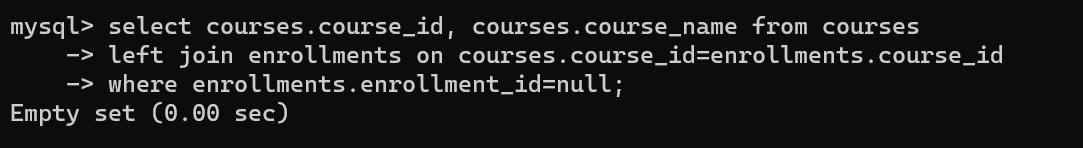


**8:- Write a query to identify courses that have no enrollments. You'll need to use a LEFT JOIN between the "Courses" table and the "Enrollments" table and filter for courses with NULL enrollment records.**

select courses.course\_id, courses.course\_name from courses

left join enrollments on courses.course\_id=enrollments.course\_id

where enrollments.enrollment\_id=null;



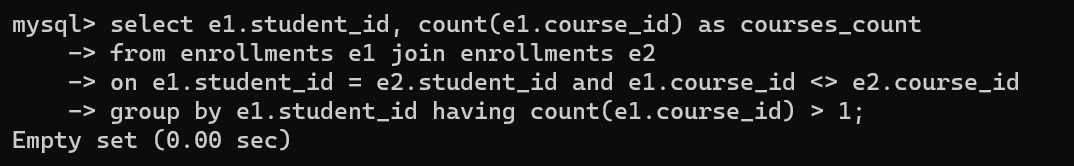
**9:- Identify students who are enrolled in more than one course. Use a self-join on the "Enrollments" table to find students with multiple enrollment records.**

select e1.student\_id, count(e1.course\_id) as courses\_count

from enrollments e1 join enrollments e2

on e1.student\_id = e2.student\_id and e1.course\_id <> e2.course\_id

group by e1.student\_id having count(e1.course\_id) > 1;

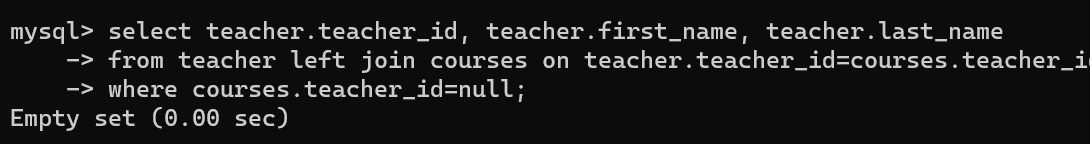


**10:- Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher” table and the "Courses" table and filter for teachers with NULL course assignments.**

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

from teacher left join courses on teacher.teacher\_id=courses.teacher\_id

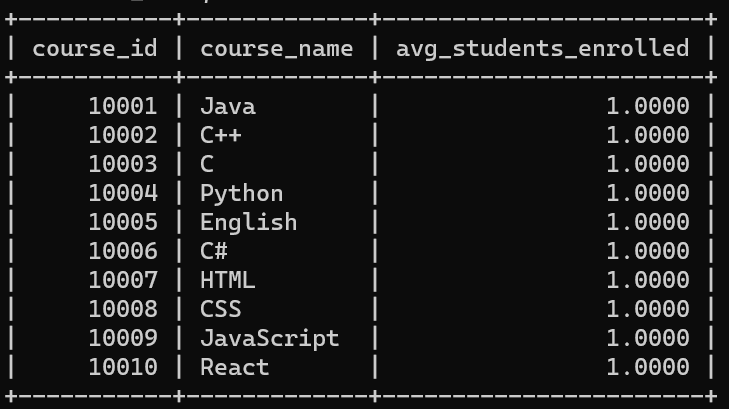
where courses.teacher\_id=null;



**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.**

select c.course\_id, c.course\_name, avg(e.students\_enrolled) as avg\_students\_enrolled from courses c join(select course\_id, count(student\_id) as students\_enrolled from enrollments group by course\_id) e on c.course\_id=e.course\_id group by c.course\_id, c.course\_name;



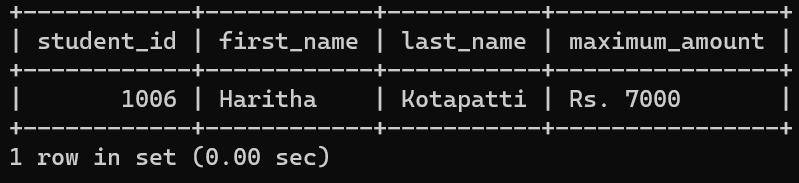
**2:- Identify the student(s) who made the highest payment. Use a subquery to find the maximum payment amount and then retrieve the student(s) associated with that amount.**

select s.student\_id,s.first\_name,s.last\_name,p.amount as maximum\_amount

from student s

join payments p on s.student\_id= p.student\_id

where p.amount= (select max(amount) from payments);



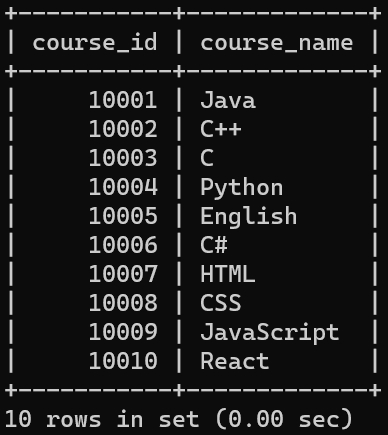
**3:- Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the course(s) with the maximum enrollment count.**

select course\_id, course\_name from courses

where (select max(total\_enrollments) from(select course\_id, count(student\_id)

as total\_enrollments

from enrollments group by course\_id) as course\_enrollments);



**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, SUM(p.amount)

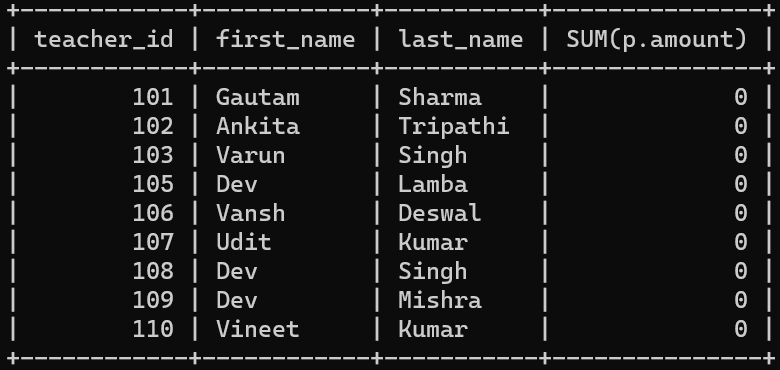
from teacher t

join courses c ON t.teacher\_id=c.teacher\_id

left join enrollments e ON c.course\_id=e.course\_id

left join payments p ON e.student\_id=p.student\_id

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



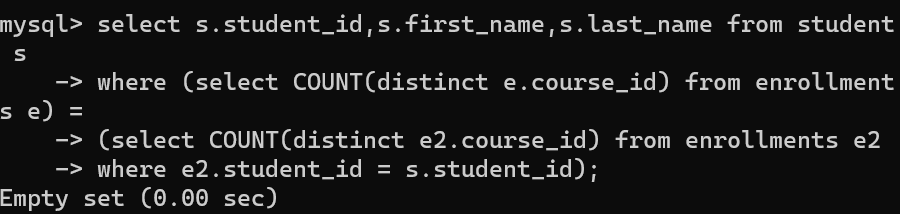
**5:- Identify students who are enrolled in all available courses. Use subqueries to compare a student's enrollments with the total number of courses.**

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

where (select COUNT(distinct e.course\_id) from enrollments e) =

(select COUNT(distinct e2.course\_id) from enrollments e2

where e2.student\_id = s.student\_id);



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

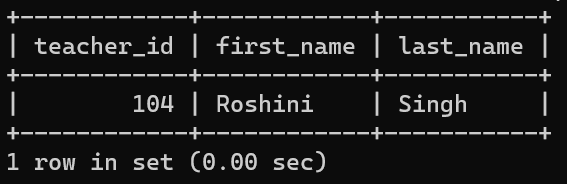
select t.teacher\_id,t.first\_name,t.last\_name

from teacher t

where

not exists ( select teacher\_id 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(student\_age) as average\_age

from ( select timestampdiff( year, date\_of\_birth,curdate()) as student\_age

from student) as student\_ages;

**A computer screen with white text

Description automatically generated**

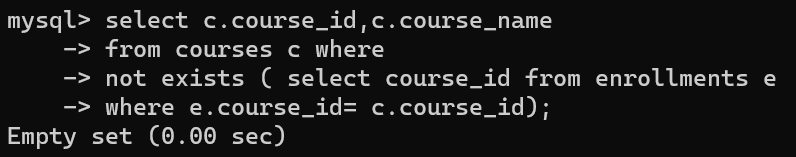
**8:- Identify courses with no enrollments. Use subqueries to find courses without enrollment records.**

select c.course\_id,c.course\_name

from courses c where

not exists ( select course\_id from enrollments e

where e.course\_id= c.course\_id);



**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 s.student\_id,s.first\_name,s.last\_name,c.course\_id,c.course\_name,

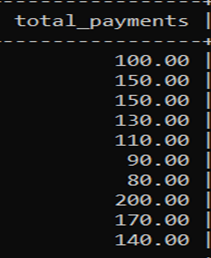
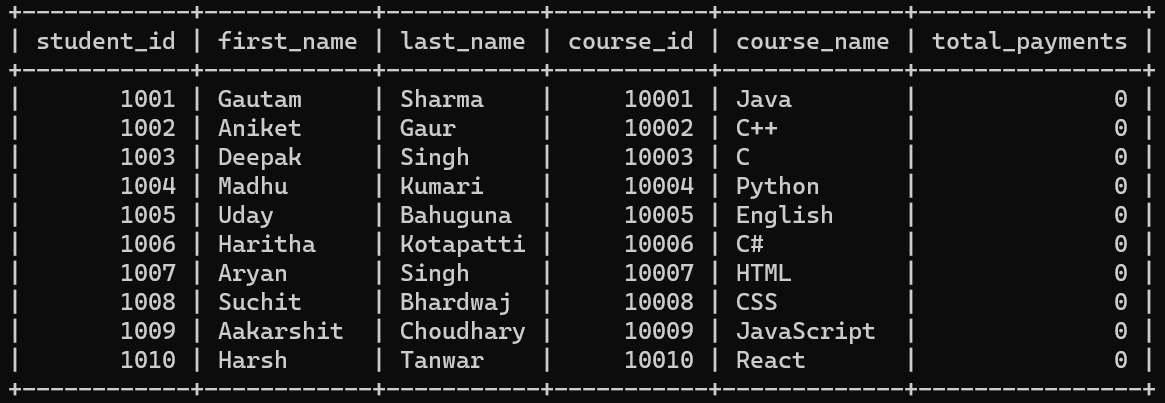
SUM(p.amount) as total\_payments from student s

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

join courses c on e.course\_id= c.course\_id

left join payments p on s.student\_id= p.student\_id

group by s.student\_id,s.first\_name,s.last\_name,c.course\_id,c.course\_name;



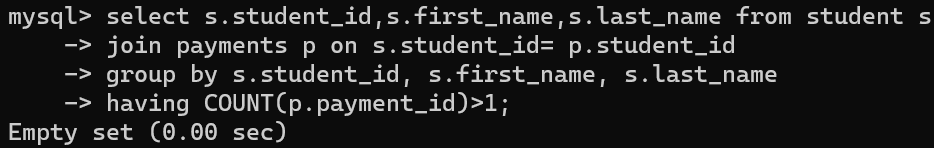
**10:- Identify students who have made more than one payment. Use subqueries and aggregate functions to count payments per student and filter for those with counts greater than one.**

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

join payments p on s.student\_id= p.student\_id

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

having COUNT(p.payment\_id)>1;



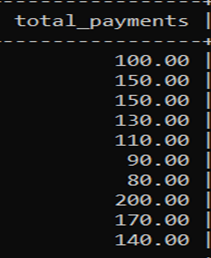
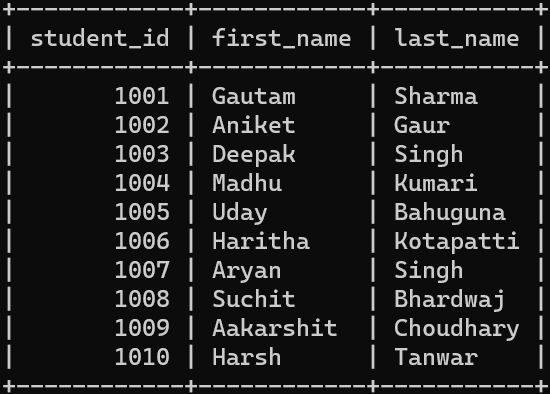
**11:- Write an SQL query to calculate the total payments made by each student. Join the "Students" table with the "Payments" table and use GROUP BY to calculate the sum of payments for each student.**

select s.student\_id,s.first\_name,s.last\_name,SUM(p.amount)

as total\_payments from student s

left join payments p on s.student\_id= p.student\_id

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



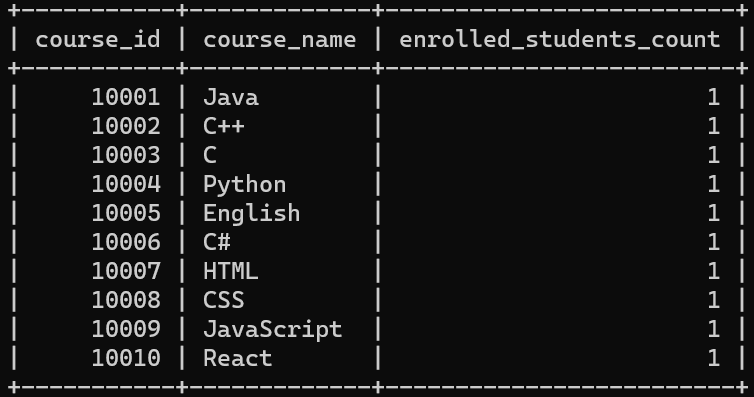
**12:- Retrieve a list of course names along with the count of students enrolled in each course. Use JOIN operations between the "Courses" table and the "Enrollments" table and GROUP BY to count enrollments.**

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

as enrolled\_students\_count from courses c left join

enrollments e on c.course\_id= e.course\_id

group by c.course\_id,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,s.last\_name,sum(p.amount)

as total\_payments from student s

left join payments p on s.student\_id= p.student\_id

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

