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)
);
```

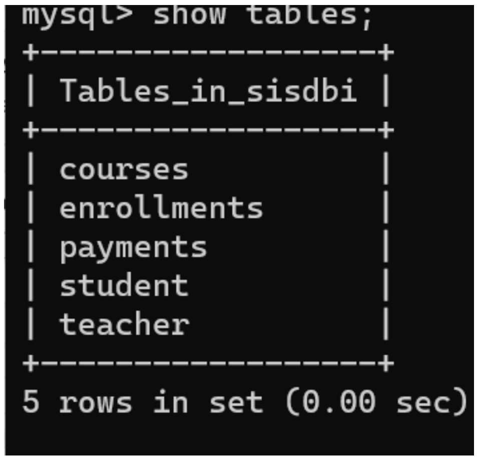
Enrollments table:- create table enrollments(

```
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(

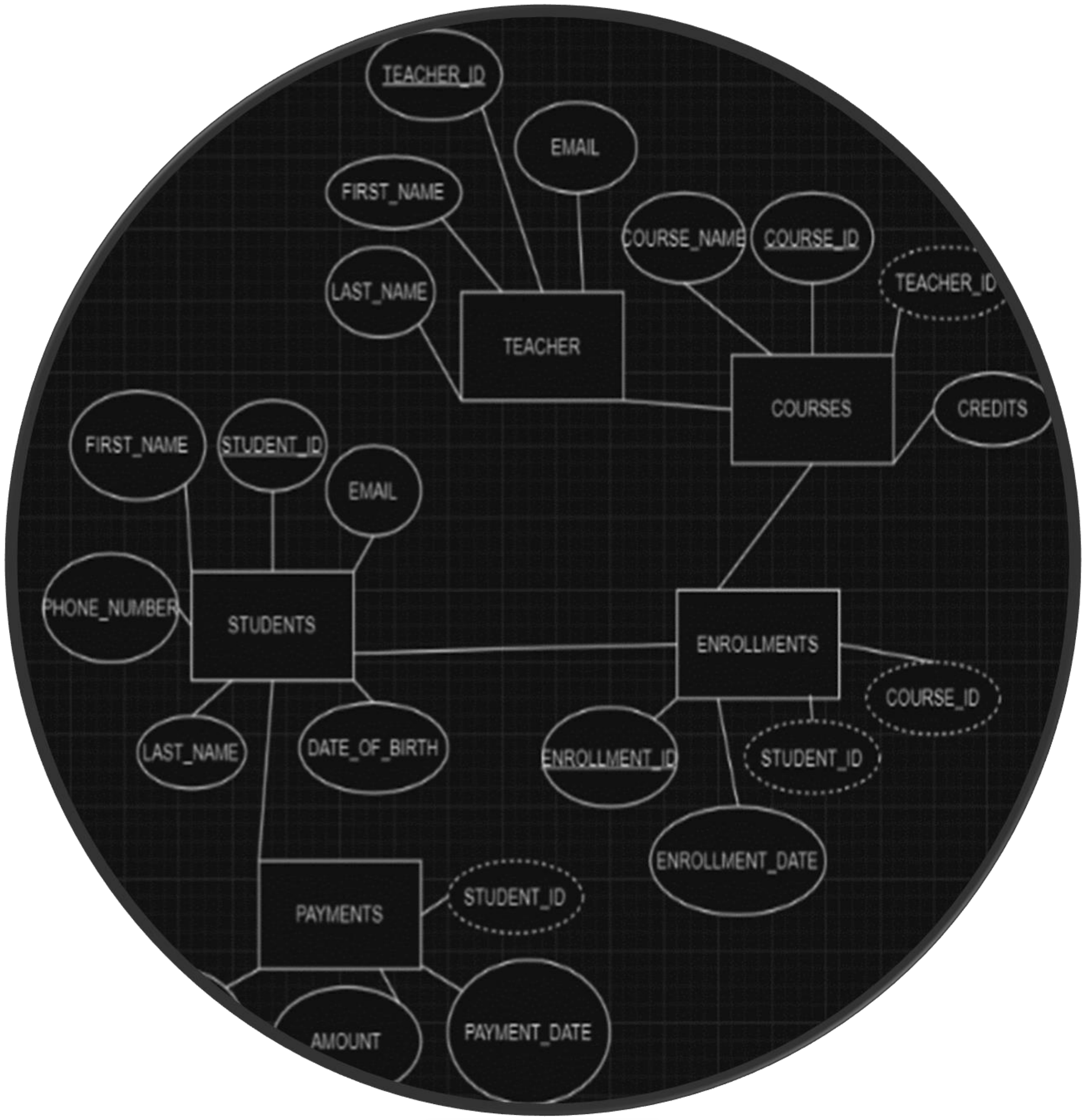
```
payment_id int primary key,
student_id int not null,
amount varchar(15) not null,
payment_date varchar(15),
FOREIGN KEY (student_id) REFERENCES student(student_id)
);
```

Gautam Sharma



```
mysql> show tables;
+------------------+
| Tables_in_sisdbi |
+------------------+
| courses          |
| enrollments      |
| payments         |
| student          |
| teacher          |
+------------------+
5 rows in set (0.00 sec)
```

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),

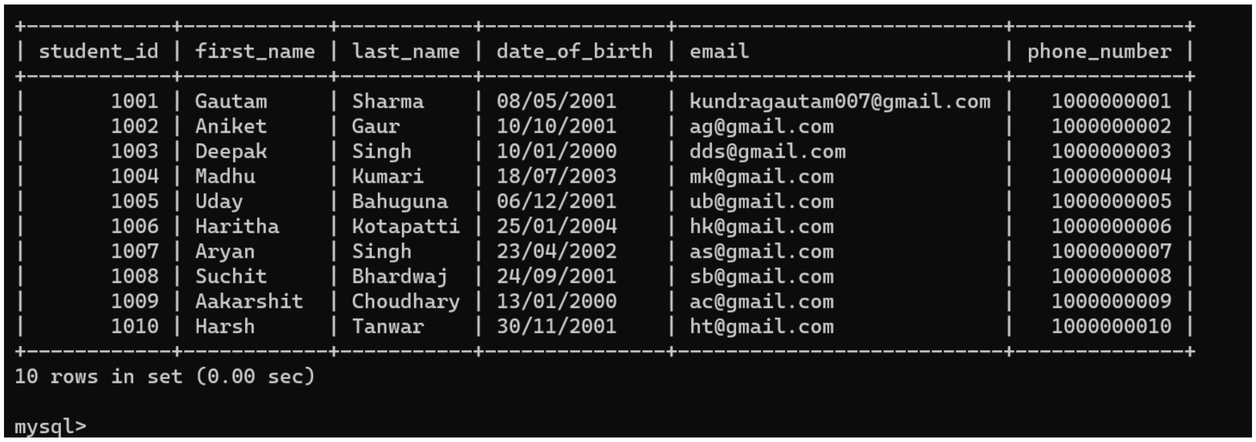
(1005,"Uday","Bahuguna",'06/12/2001',"ub@gmail.com",1000000005),

(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),

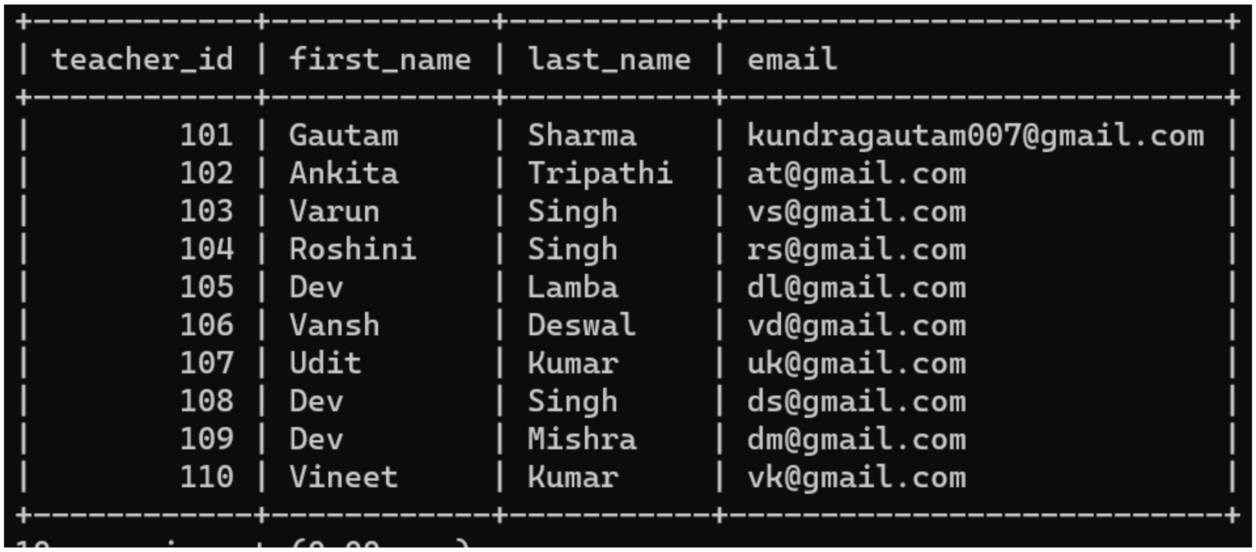
(1009,"Aakarshit","Choudhary",'13/01/2000',"ac@gmail.com",1000000009),
(1010,"Harsh","Tanwar",'30/11/2001',"ht@gmail.com",1000000010);



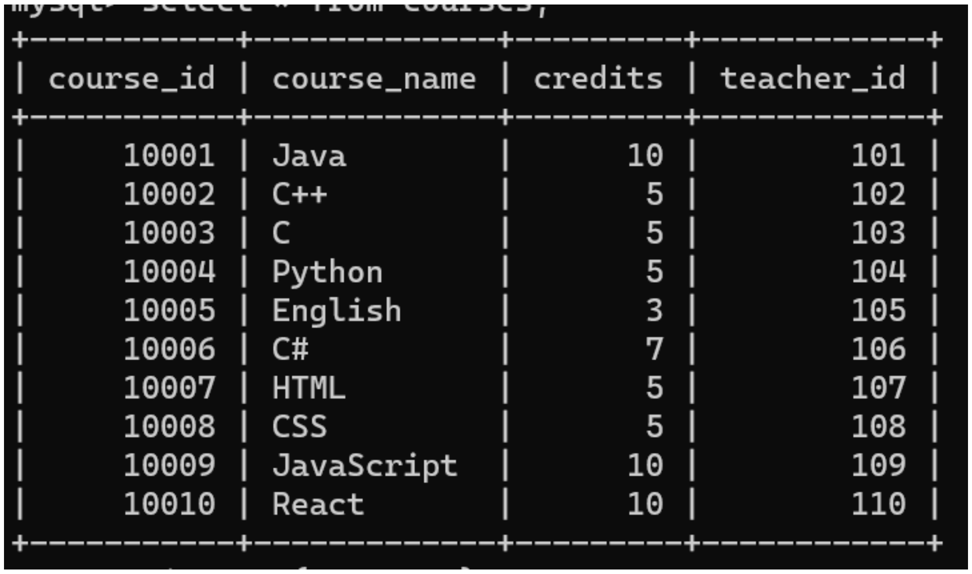
```
+------------+------------+-----------+---------------+---------------------------+--------------+
| student_id | first_name | last_name | date_of_birth | email                     | phone_number |
+------------+------------+-----------+---------------+---------------------------+--------------+
|       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 |
+------------+------------+-----------+---------------+---------------------------+--------------+
10 rows in set (0.00 sec)

mysql>
```

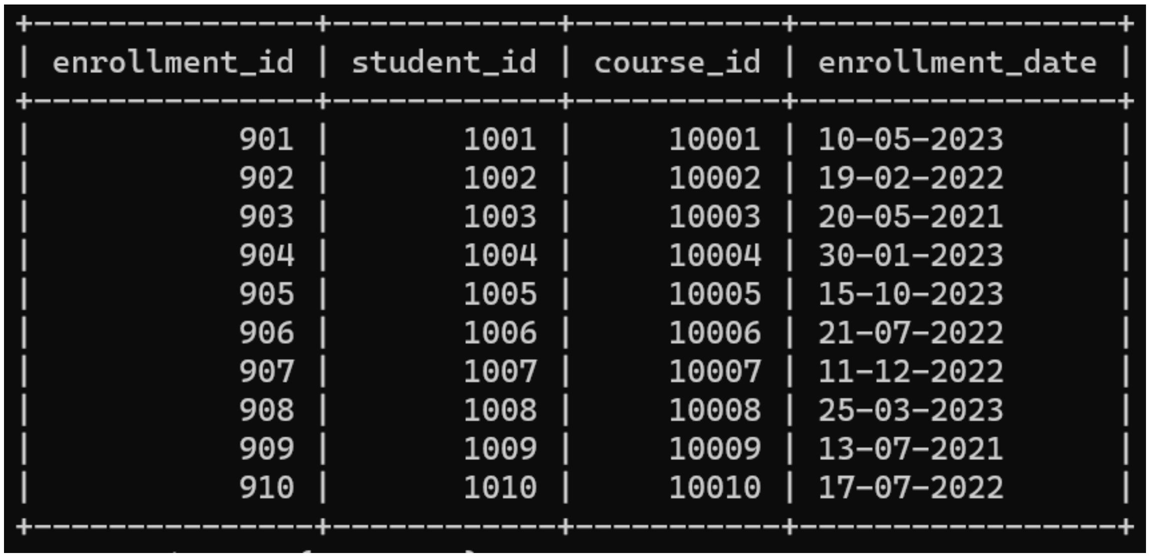
```
insert into teacher(teacher_id,first_name,last_name,email)
values(101,"Gautam","Sharma","kundragautam007@gmail.com"),
(102,"Ankita","Tripathi","at@gmail.com"),
(103,"Varun","Singh","vs@gmail.com"),
(104,"Roshini","Singh","rs@gmail.com"),
(105,"Dev","Lamba","dl@gmail.com"),
(106,"Vansh","Deswal","vd@gmail.com"),
(107,"Udit","Kumar","uk@gmail.com"),
(108,"Dev","Singh","ds@gmail.com"),
(109,"Dev","Mishra","dm@gmail.com"),
(110,"Vineet","Kumar","vk@gmail.com");
```



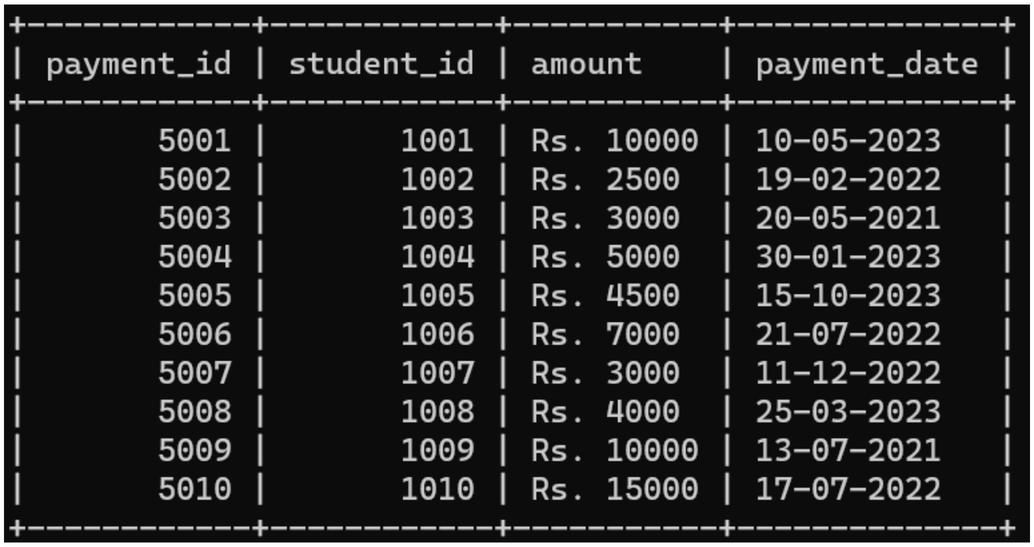
```
+------------+------------+-----------+---------------------------+
| teacher_id | first_name | last_name | email                     |
+------------+------------+-----------+---------------------------+
|        101 | Gautam     | Sharma    | kundragautam007@gmail.com |
|        102 | Ankita     | 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);
```



```
+-----------+-------------+---------+------------+
| course_id | course_name | credits | teacher_id |
+-----------+-------------+---------+------------+
|     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");
```



```
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           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");
```



```
+------------+------------+-----------+--------------+
| payment_id | student_id | amount    | payment_date |
+------------+------------+-----------+--------------+
|       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

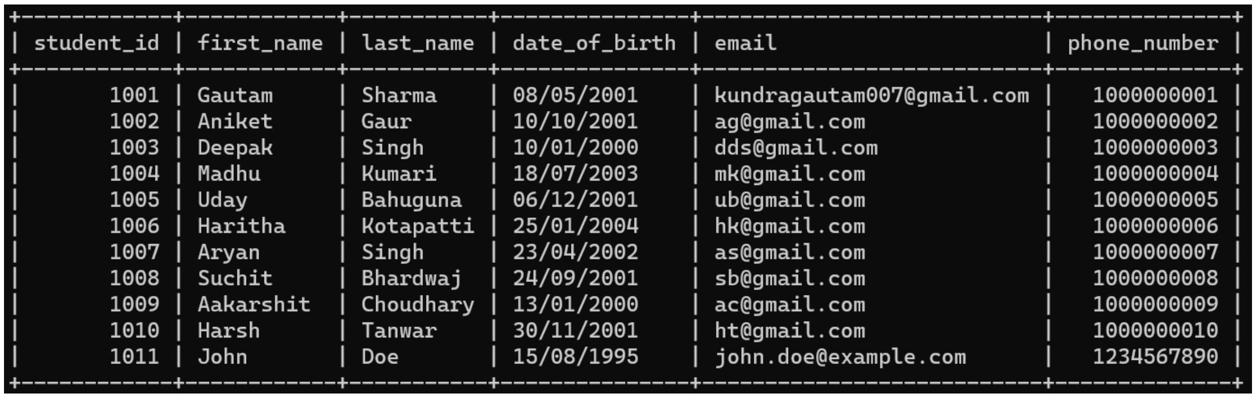
c. Date of Birth: 1995-08-15

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);

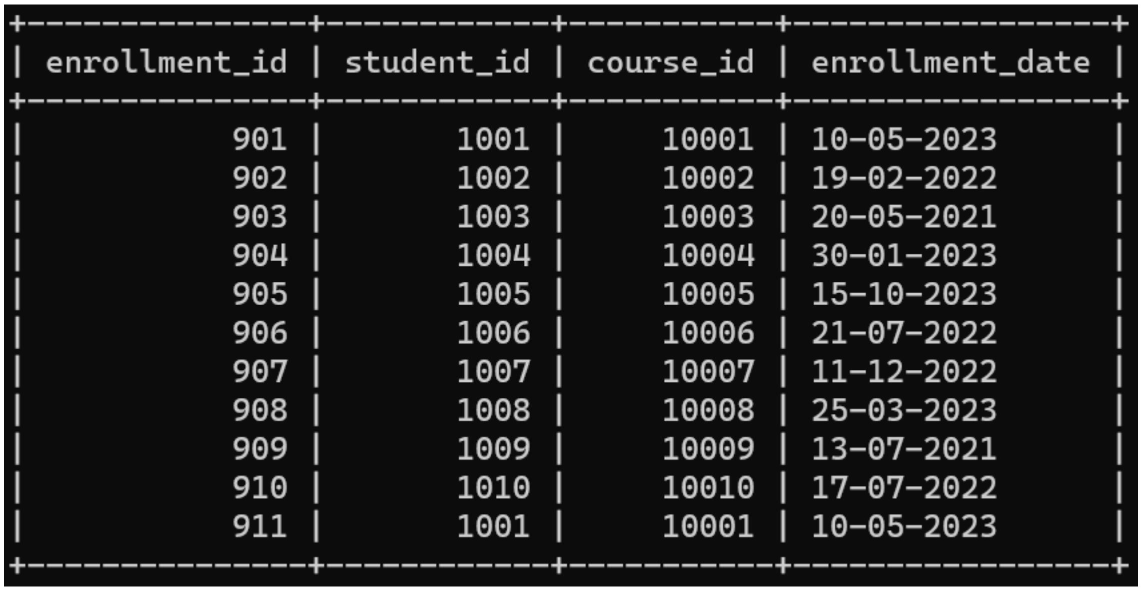


```
+------------+------------+-----------+---------------+---------------------------+--------------+
| student_id | first_name | last_name | date_of_birth | email                     | phone_number |
+------------+------------+-----------+---------------+---------------------------+--------------+
|       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 |
|       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');
```

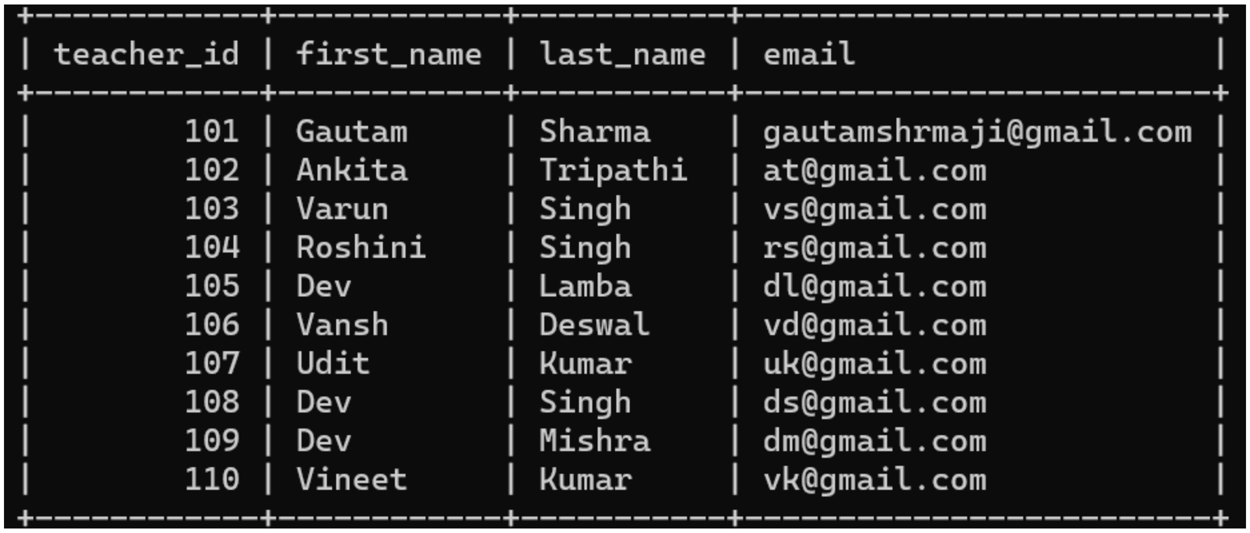


```
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           901 |       1001 |     10001 | 10-05-2023      |
|           902 |       1002 |     10002 | 19-02-2022      |
|           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      |
|           911 |       1001 |     10001 | 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;
```

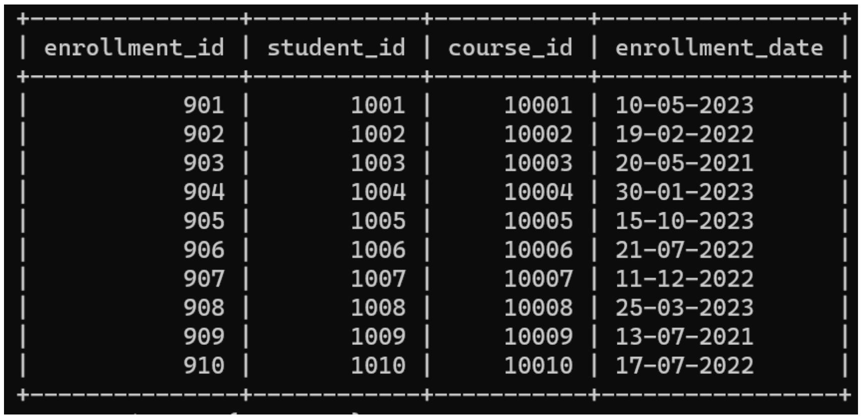


```
+------------+------------+-----------+-------------------------+
| teacher_id | first_name | last_name | email                   |
+------------+------------+-----------+-------------------------+
|        101 | Gautam     | Sharma    | gautamshrmaji@gmail.com |
|        102 | Ankita     | Tripathi  | 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            |
+------------+------------+-----------+-------------------------+
```

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;

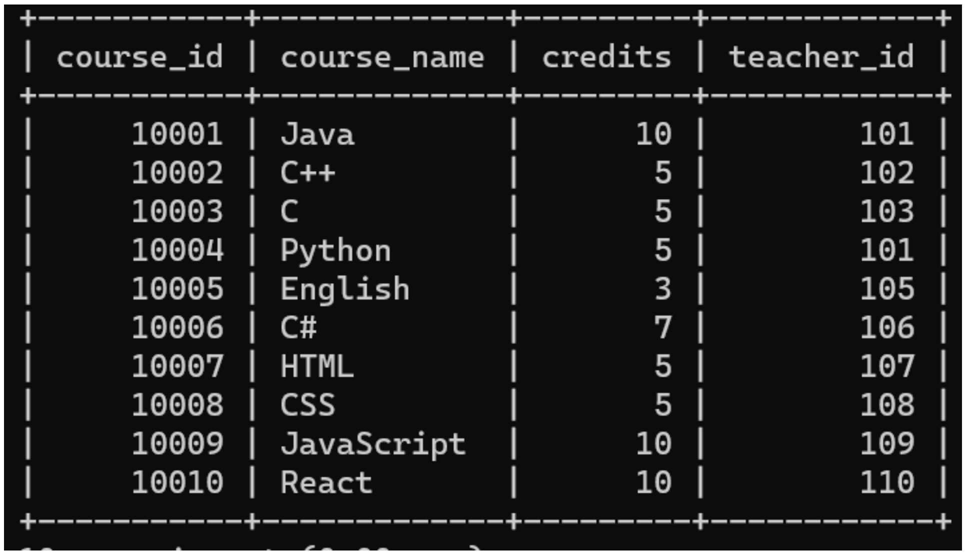


```
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           901 |       1001 |     10001 | 10-05-2023      |
|           902 |       1002 |     10002 | 19-02-2022      |
|           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      |
+---------------+------------+-----------+-----------------+
```

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;



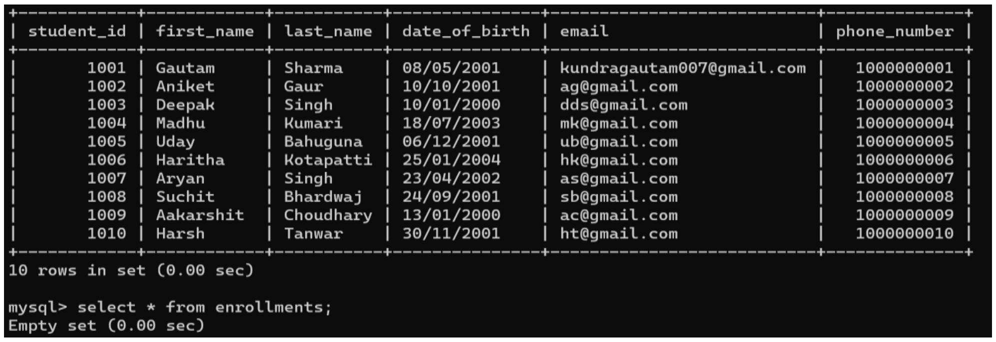
```
+-----------+-------------+---------+------------+
| course_id | course_name | credits | teacher_id |
+-----------+-------------+---------+------------+
|     10001 | Java        |      10 |        101 |
|     10002 | C++         |       5 |        102 |
|     10003 | C           |       5 |        103 |
|     10004 | Python      |       5 |        101 |
|     10005 | English     |       3 |        105 |
|     10006 | C#          |       7 |        106 |
|     10007 | HTML        |       5 |        107 |
|     10008 | CSS         |       5 |        108 |
|     10009 | JavaScript  |      10 |        109 |
|     10010 | React       |      10 |        110 |
+-----------+-------------+---------+------------+
```

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;



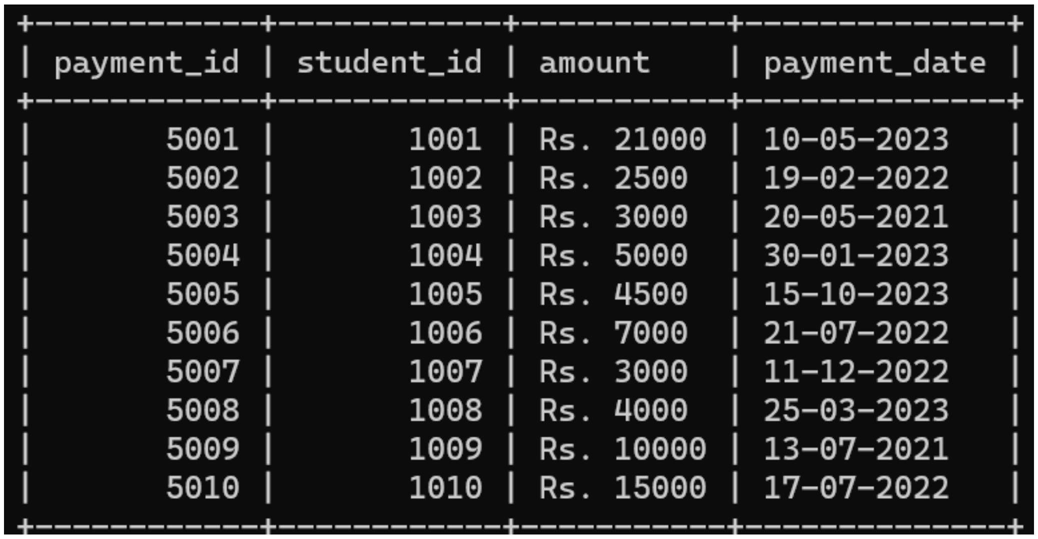
```
+------------+------------+-----------+---------------+---------------------------+--------------+
| student_id | first_name | last_name | date_of_birth | email                     | phone_number |
+------------+------------+-----------+---------------+---------------------------+--------------+
|       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 |
+------------+------------+-----------+---------------+---------------------------+--------------+
10 rows in set (0.00 sec)

mysql> select * from enrollments;
Empty set (0.00 sec)
```

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;



```
+------------+------------+-----------+--------------+
| payment_id | student_id | amount    | payment_date |
+------------+------------+-----------+--------------+
|       5001 |       1001 | Rs. 21000 | 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   |
+------------+------------+-----------+--------------+
```

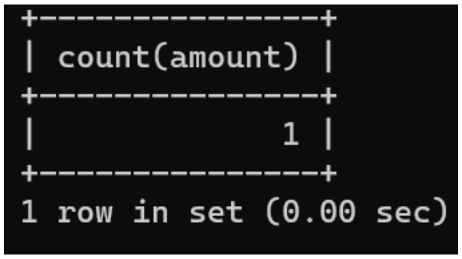
Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:

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;



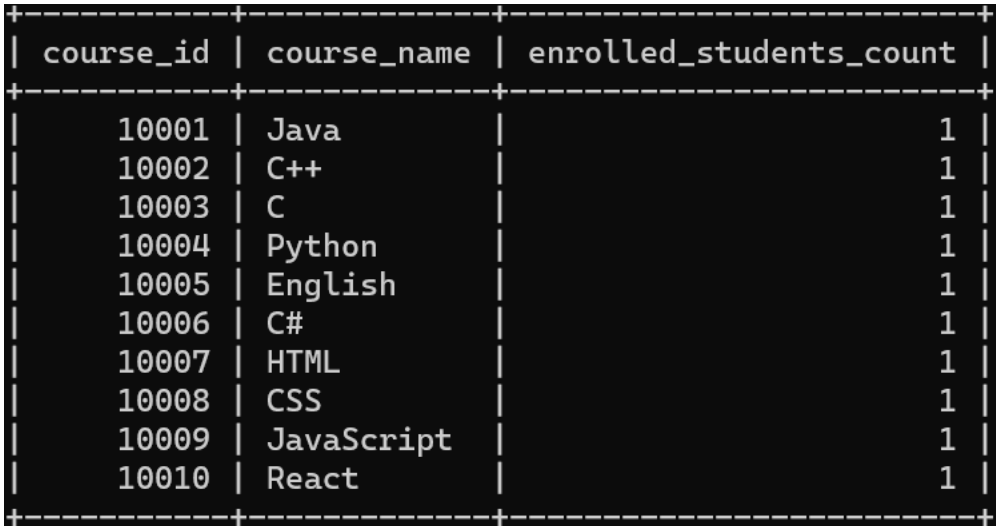
```
+---------------+
| count(amount) |
+---------------+
|             1 |
+---------------+
1 row in set (0.00 sec)
```

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;



```
+-----------+-------------+-------------------------+
| course_id | course_name | enrolled_students_count |
+-----------+-------------+-------------------------+
|     10001 | Java        |                       1 |
|     10002 | C++         |                       1 |
|     10003 | C           |                       1 |
|     10004 | Python      |                       1 |
|     10005 | English     |                       1 |
|     10006 | C#          |                       1 |
|     10007 | HTML        |                       1 |
|     10008 | CSS         |                       1 |
|     10009 | JavaScript  |                       1 |
|     10010 | React       |                       1 |
+-----------+-------------+-------------------------+
```

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

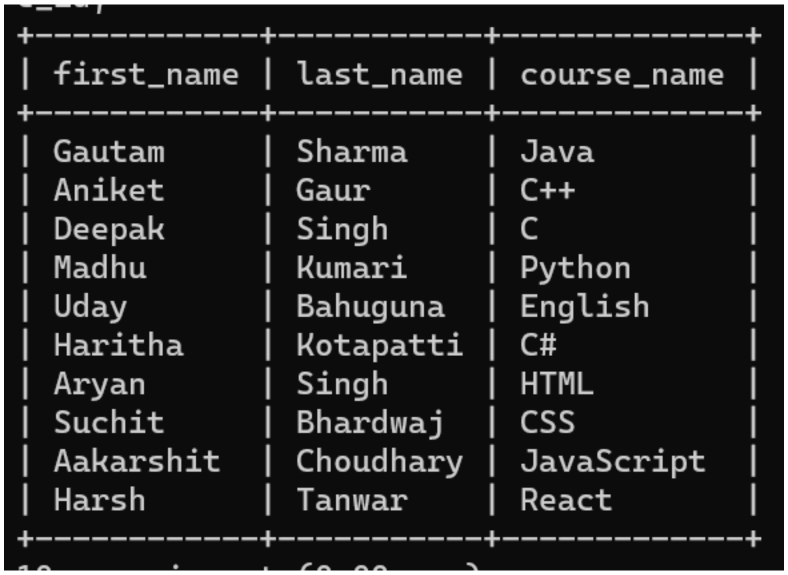
where enrollments.student_id=null;

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;

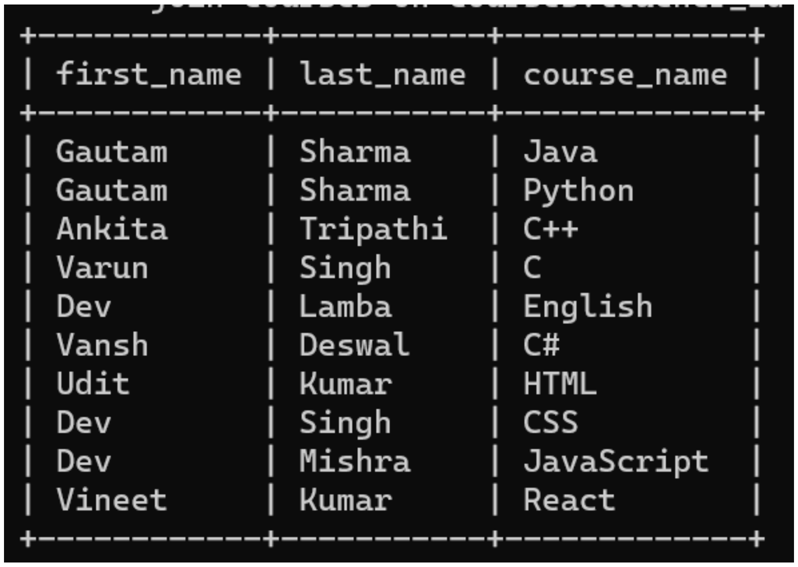


```
+------------+-----------+-------------+
| first_name | last_name | course_name |
+------------+-----------+-------------+
| Gautam     | Sharma    | Java        |
| Aniket     | Gaur      | C++         |
| Deepak     | Singh     | C           |
| Madhu      | Kumari    | Python      |
| Uday       | Bahuguna  | English     |
| Haritha    | Kotapatti | C#          |
| Aryan      | Singh     | HTML        |
| Suchit     | Bhardwaj  | CSS         |
| Aakarshit  | Choudhary | JavaScript  |
| Harsh      | Tanwar    | React       |
+------------+-----------+-------------+
```

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;



```
+------------+-----------+-------------+
| first_name | last_name | course_name |
+------------+-----------+-------------+
| Gautam     | Sharma    | Java        |
| Gautam     | Sharma    | Python      |
| Ankita     | Tripathi  | C++         |
| Varun      | Singh     | C           |
| Dev        | Lamba     | English     |
| Vansh      | Deswal    | C#          |
| Udit       | Kumar     | HTML        |
| Dev        | Singh     | CSS         |
| Dev        | Mishra    | JavaScript  |
| Vineet     | Kumar     | React       |
+------------+-----------+-------------+
```

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.

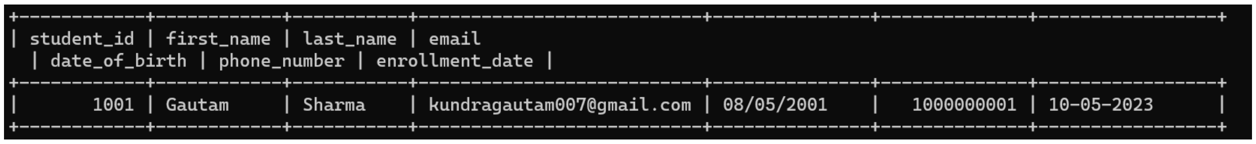
select student.student_id, student.first_name, student.last_name,

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;



```
+------------+------------+-----------+---------------------------+---------------+--------------+-----------------+
| student_id | first_name | last_name | email
  | date_of_birth | phone_number | enrollment_date |
+------------+------------+-----------+---------------------------+---------------+--------------+-----------------+
|       1001 | Gautam     | Sharma    | kundragautam007@gmail.com | 08/05/2001    |   1000000001 | 10-05-2023      |
+------------+------------+-----------+---------------------------+---------------+--------------+-----------------+
```

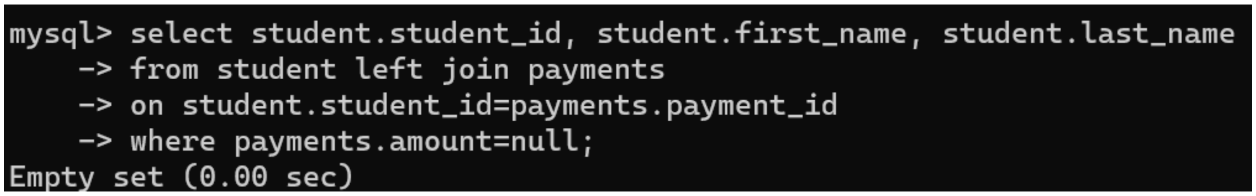
7:- Find the names of students who have not made any payments. Use a LEFT JOIN between the "Students" table and the "Payments" table and filter for students with NULL payment records.

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;



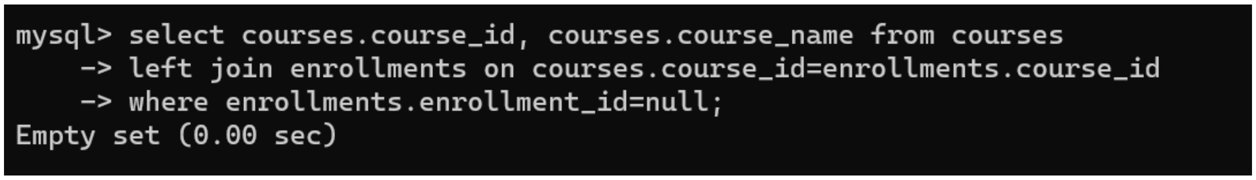
```
mysql> select student.student_id, student.first_name, student.last_name
    -> from student left join payments
    -> on student.student_id=payments.payment_id
    -> where payments.amount=null;
Empty set (0.00 sec)
```

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;



```
mysql> select courses.course_id, courses.course_name from courses
    -> left join enrollments on courses.course_id=enrollments.course_id
    -> where enrollments.enrollment_id=null;
Empty set (0.00 sec)
```

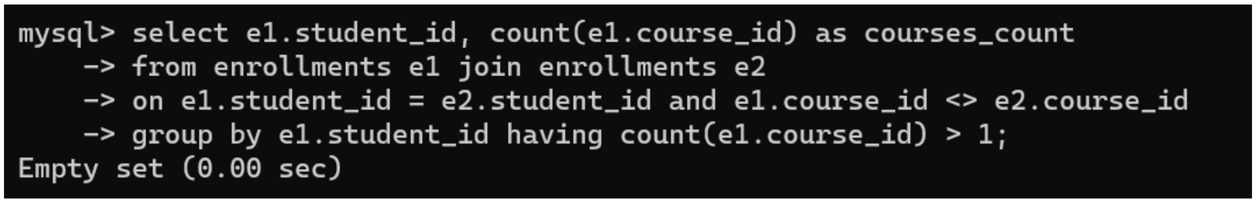
9:- Identify students who are enrolled in more than one course. Use a self-join on the "Enrollments" table to find students with multiple enrollment records.

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;



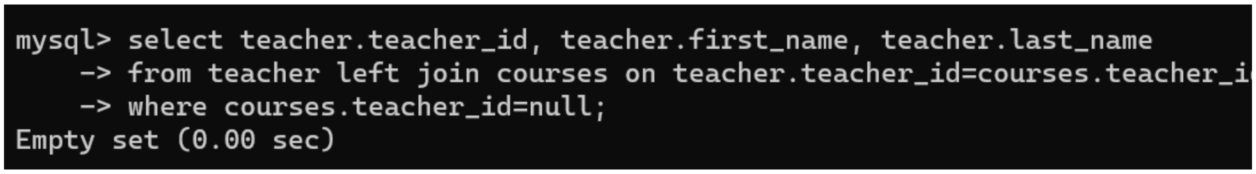
```
mysql> select e1.student_id, count(e1.course_id) as courses_count
    -> from enrollments e1 join enrollments e2
    -> on e1.student_id = e2.student_id and e1.course_id <> e2.course_id
    -> group by e1.student_id having count(e1.course_id) > 1;
Empty set (0.00 sec)
```

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;

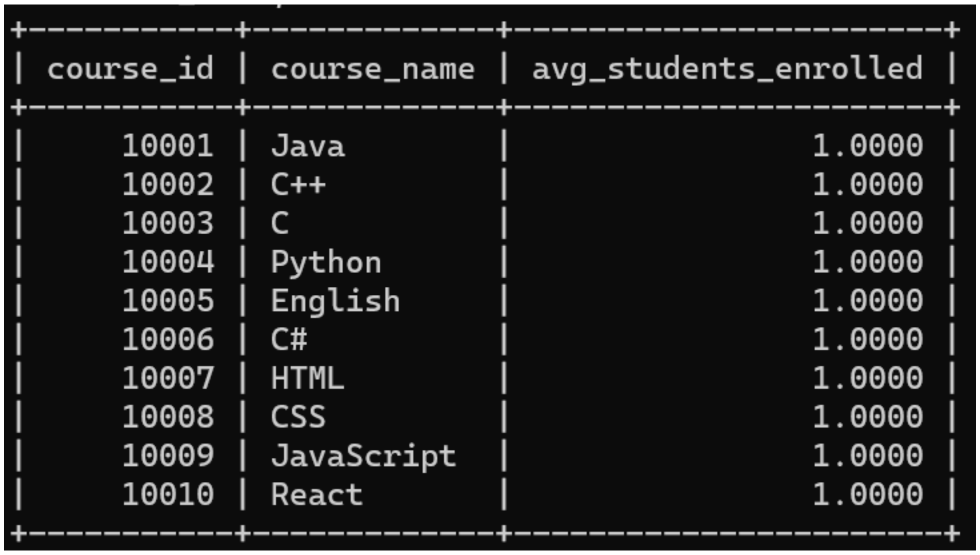


```
mysql> select teacher.teacher_id, teacher.first_name, teacher.last_name
    -> from teacher left join courses on teacher.teacher_id=courses.teacher_i
    -> where courses.teacher_id=null;
Empty set (0.00 sec)
```

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.

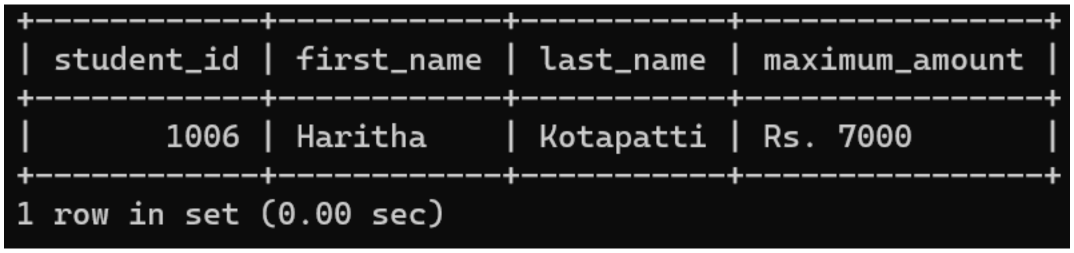
select c.course_id, c.course_name, avg(e.students_enrolled) as avg_students_enrolled from courses c join(select course_id, count(student_id) as students_enrolled from enrollments group by course_id) e on c.course_id=e.course_id group by c.course_id, c.course_name;



```
+-----------+-------------+----------------------+
| course_id | course_name | avg_students_enrolled |
+-----------+-------------+----------------------+
|     10001 | Java        |               1.0000 |
|     10002 | C++         |               1.0000 |
|     10003 | C           |               1.0000 |
|     10004 | Python      |               1.0000 |
|     10005 | English     |               1.0000 |
|     10006 | C#          |               1.0000 |
|     10007 | HTML        |               1.0000 |
|     10008 | CSS         |               1.0000 |
|     10009 | JavaScript  |               1.0000 |
|     10010 | React       |               1.0000 |
+-----------+-------------+----------------------+
```

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.

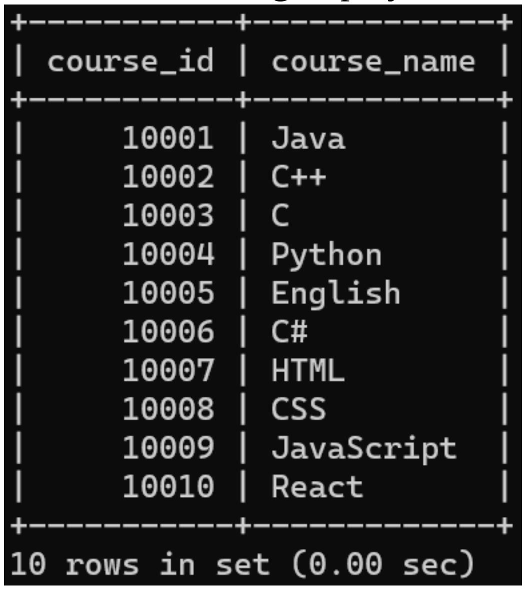
select s.student_id,s.first_name,s.last_name,p.amount as maximum_amount
from student s
join payments p on s.student_id= p.student_id
where p.amount= (select max(amount) from payments);



```
+------------+------------+-----------+----------------+
| student_id | first_name | last_name | maximum_amount |
+------------+------------+-----------+----------------+
|       1006 | Haritha    | Kotapatti | Rs. 7000       |
+------------+------------+-----------+----------------+
1 row in set (0.00 sec)
```

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

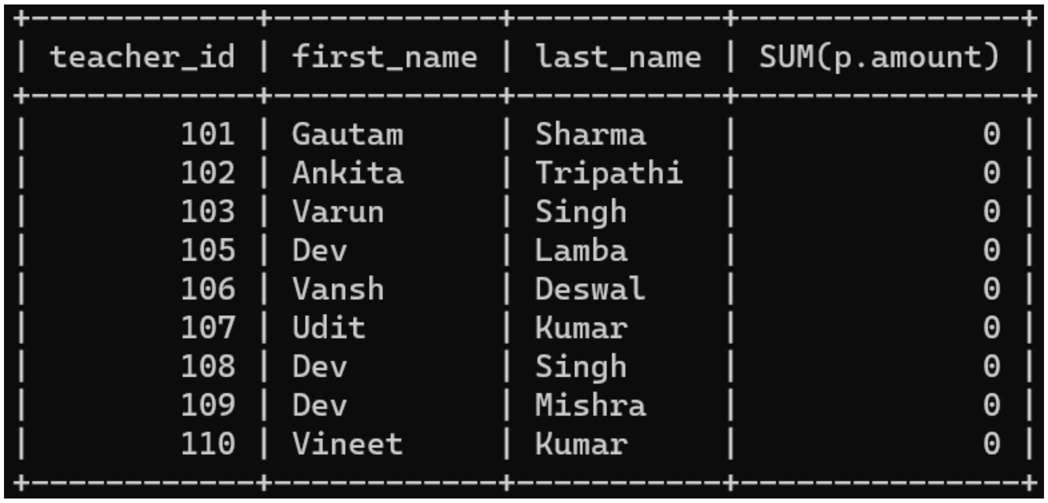
```
select course_id, course_name from courses
where (select max(total_enrollments) from(select course_id, count(student_id)
as total_enrollments
from enrollments group by course_id) as course_enrollments);
```



```
+-----------+-------------+
| course_id | course_name |
+-----------+-------------+
|     10001 | Java        |
|     10002 | C++         |
|     10003 | C           |
|     10004 | Python      |
|     10005 | English     |
|     10006 | C#          |
|     10007 | HTML        |
|     10008 | CSS         |
|     10009 | JavaScript  |
|     10010 | React       |
+-----------+-------------+
10 rows in set (0.00 sec)
```

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

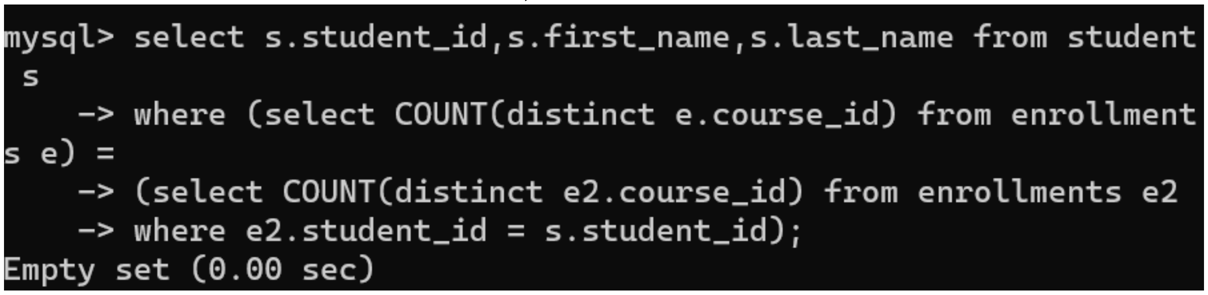
```
select t.teacher_id,t.first_name,t.last_name, SUM(p.amount)
from teacher t
join courses c ON t.teacher_id=c.teacher_id
left join enrollments e ON c.course_id=e.course_id
left join payments p ON e.student_id=p.student_id
group by t.teacher_id,t.first_name,t.last_name;
```



```
+------------+------------+-----------+---------------+
| teacher_id | first_name | last_name | SUM(p.amount) |
+------------+------------+-----------+---------------+
|        101 | Gautam     | Sharma    |             0 |
|        102 | Ankita     | Tripathi  |             0 |
|        103 | Varun      | Singh     |             0 |
|        105 | Dev        | Lamba     |             0 |
|        106 | Vansh      | Deswal    |             0 |
|        107 | Udit       | Kumar     |             0 |
|        108 | Dev        | Singh     |             0 |
|        109 | Dev        | Mishra    |             0 |
|        110 | Vineet     | Kumar     |             0 |
+------------+------------+-----------+---------------+
```

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

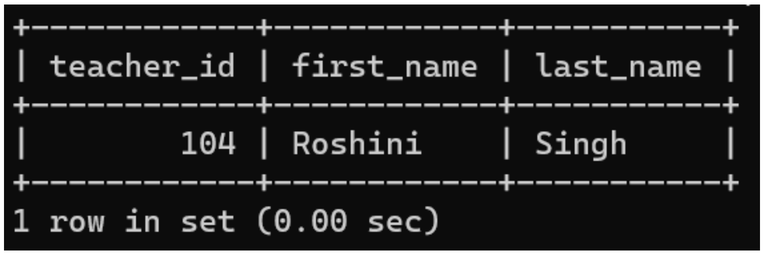
```
select s.student_id,s.first_name,s.last_name from student s
where (select COUNT(distinct e.course_id) from enrollments e) =
(select COUNT(distinct e2.course_id) from enrollments e2
where e2.student_id = s.student_id);
```



```
mysql> select s.student_id,s.first_name,s.last_name from student
 s
    -> where (select COUNT(distinct e.course_id) from enrollment
s e) =
    -> (select COUNT(distinct e2.course_id) from enrollments e2
    -> where e2.student_id = s.student_id);
Empty set (0.00 sec)
```

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

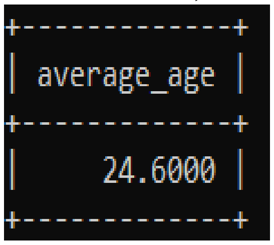
```
select t.teacher_id,t.first_name,t.last_name
from teacher t
where
not exists ( select teacher_id from courses c
where c.teacher_id=t.teacher_id);
```



```
+------------+------------+-----------+
| teacher_id | first_name | last_name |
+------------+------------+-----------+
|        104 | Roshini    | Singh     |
+------------+------------+-----------+
1 row in set (0.00 sec)
```

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

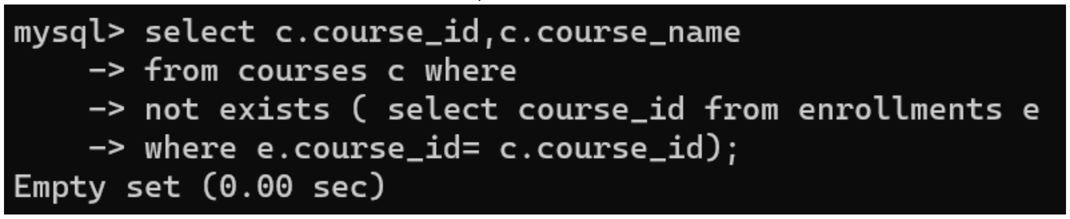
select avg(student_age) as average_age
from (select timestampdiff(year, date_of_birth,curdate()) as student_age
from student) as student_ages;



```
+-------------+
| average_age |
+-------------+
|     24.6000 |
+-------------+
```

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

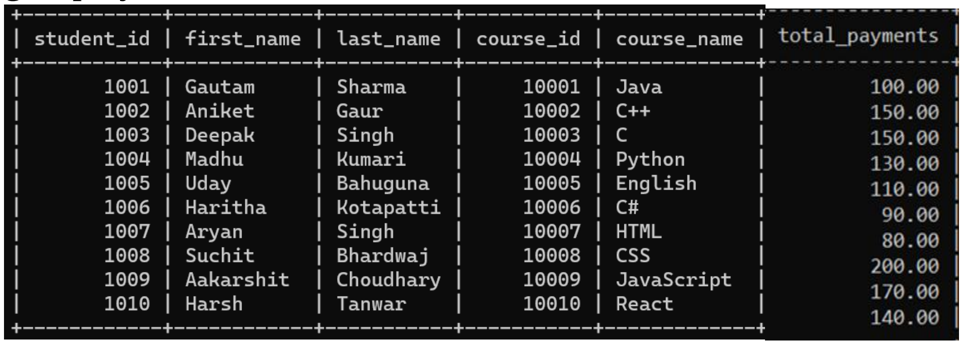
select c.course_id,c.course_name
from courses c where
not exists (select course_id from enrollments e
where e.course_id= c.course_id);



```
mysql> 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);
Empty set (0.00 sec)
```

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

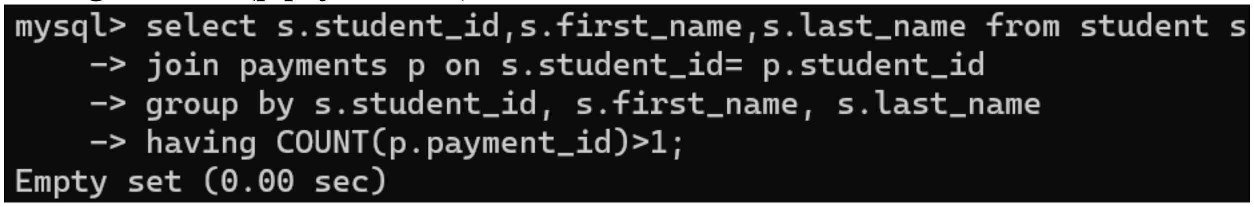
select s.student_id,s.first_name,s.last_name,c.course_id,c.course_name,
SUM(p.amount) as total_payments from student s
join enrollments e on s.student_id= e.student_id
join courses c on e.course_id= c.course_id
left join payments p on s.student_id= p.student_id
group by s.student_id,s.first_name,s.last_name,c.course_id,c.course_name;



```
+------------+------------+-----------+-----------+-------------+----------------+
| student_id | first_name | last_name | course_id | course_name | total_payments |
+------------+------------+-----------+-----------+-------------+----------------+
|       1001 | Gautam     | Sharma    |     10001 | Java        |         100.00 |
|       1002 | Aniket     | Gaur      |     10002 | C++         |         150.00 |
|       1003 | Deepak     | Singh     |     10003 | C           |         150.00 |
|       1004 | Madhu      | Kumari    |     10004 | Python      |         130.00 |
|       1005 | Uday       | Bahuguna  |     10005 | English     |         110.00 |
|       1006 | Haritha    | Kotapatti |     10006 | C#          |          90.00 |
|       1007 | Aryan      | Singh     |     10007 | HTML        |          80.00 |
|       1008 | Suchit     | Bhardwaj  |     10008 | CSS         |         200.00 |
|       1009 | Aakarshit  | Choudhary |     10009 | JavaScript  |         170.00 |
|       1010 | Harsh      | Tanwar    |     10010 | React       |         140.00 |
+------------+------------+-----------+-----------+-------------+----------------+
```

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.

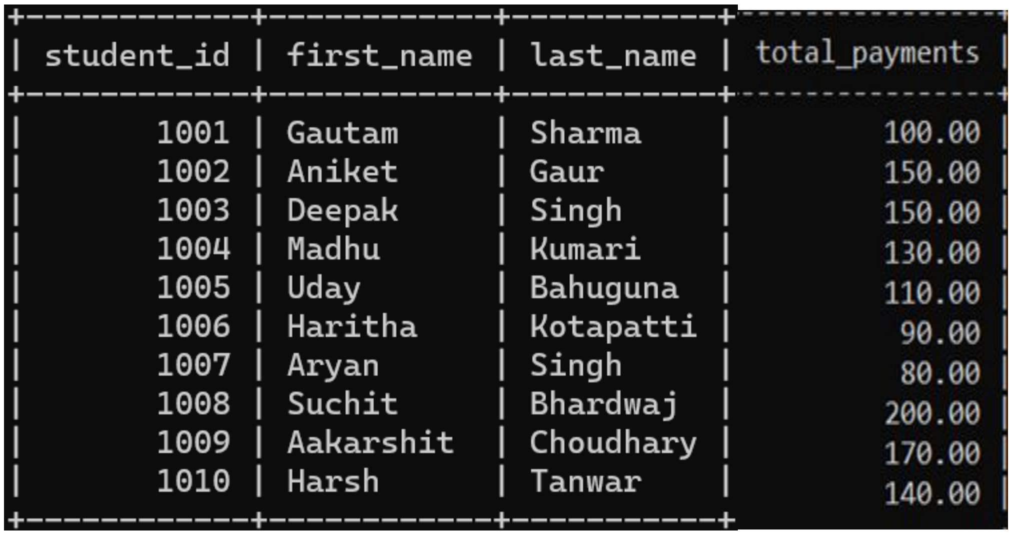
select s.student_id,s.first_name,s.last_name from student s
join payments p on s.student_id= p.student_id
group by s.student_id, s.first_name, s.last_name
having COUNT(p.payment_id)>1;



```
mysql> 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;
Empty set (0.00 sec)
```

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.

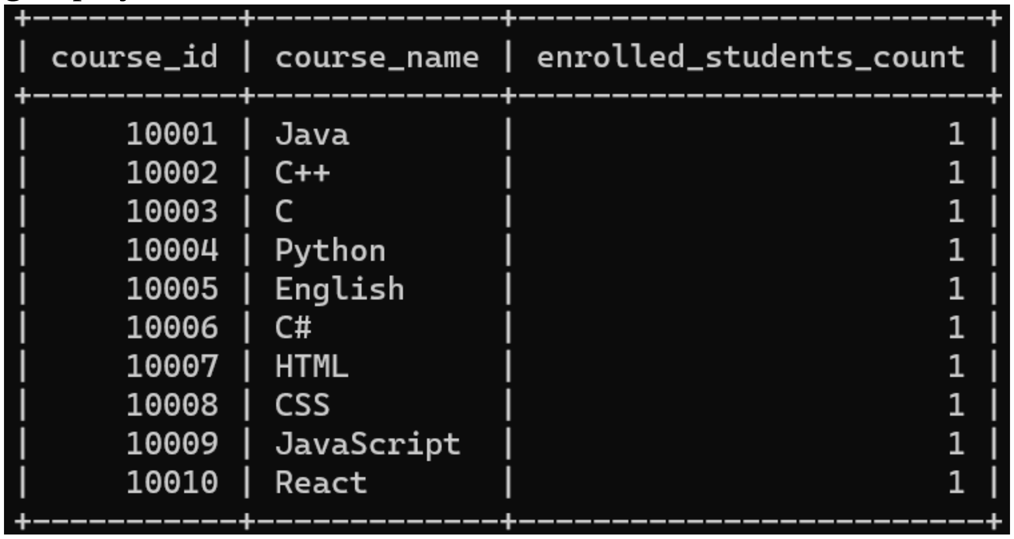
select s.student_id,s.first_name,s.last_name,SUM(p.amount)
as total_payments from student s
left join payments p on s.student_id= p.student_id
group by s.student_id,s.first_name,s.last_name;



```
+------------+------------+-----------+----------------+
| student_id | first_name | last_name | total_payments |
+------------+------------+-----------+----------------+
|       1001 | Gautam     | Sharma    |         100.00 |
|       1002 | Aniket     | Gaur      |         150.00 |
|       1003 | Deepak     | Singh     |         150.00 |
|       1004 | Madhu      | Kumari    |         130.00 |
|       1005 | Uday       | Bahuguna  |         110.00 |
|       1006 | Haritha    | Kotapatti |          90.00 |
|       1007 | Aryan      | Singh     |          80.00 |
|       1008 | Suchit     | Bhardwaj  |         200.00 |
|       1009 | Aakarshit  | Choudhary |         170.00 |
|       1010 | Harsh      | Tanwar    |         140.00 |
+------------+------------+-----------+
```

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;

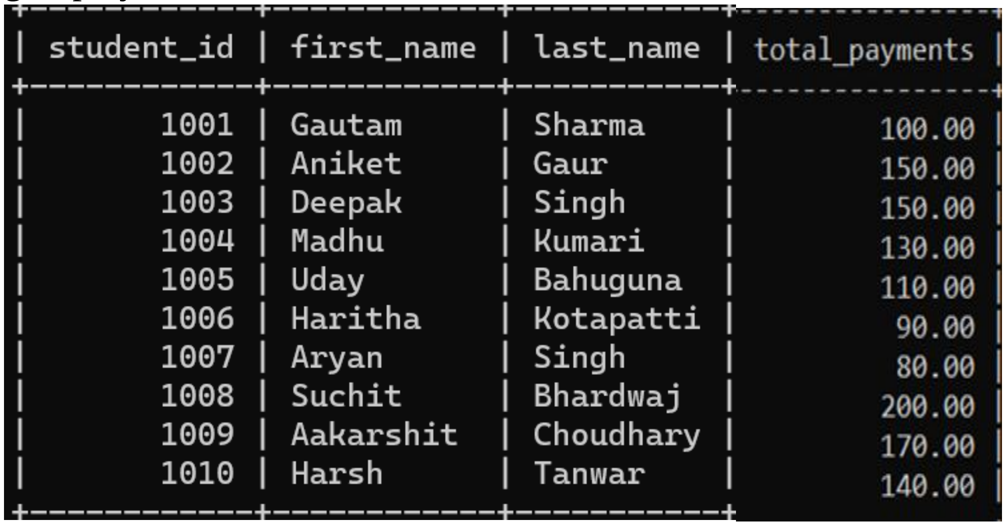


```
+-----------+-------------+-------------------------+
| course_id | course_name | enrolled_students_count |
+-----------+-------------+-------------------------+
|     10001 | Java        |                       1 |
|     10002 | C++         |                       1 |
|     10003 | C           |                       1 |
|     10004 | Python      |                       1 |
|     10005 | English     |                       1 |
|     10006 | C#          |                       1 |
|     10007 | HTML        |                       1 |
|     10008 | CSS         |                       1 |
|     10009 | JavaScript  |                       1 |
|     10010 | React       |                       1 |
+-----------+-------------+-------------------------+
```

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;



```
+------------+------------+-----------+----------------+
| student_id | first_name | last_name | total_payments |
+------------+------------+-----------+----------------+
|       1001 | Gautam     | Sharma    |         100.00 |
|       1002 | Aniket     | Gaur      |         150.00 |
|       1003 | Deepak     | Singh     |         150.00 |
|       1004 | Madhu      | Kumari    |         130.00 |
|       1005 | Uday       | Bahuguna  |         110.00 |
|       1006 | Haritha    | Kotapatti |          90.00 |
|       1007 | Aryan      | Singh     |          80.00 |
|       1008 | Suchit     | Bhardwaj  |         200.00 |
|       1009 | Aakarshit  | Choudhary |         170.00 |
|       1010 | Harsh      | Tanwar    |         140.00 |
+------------+------------+-----------+
```