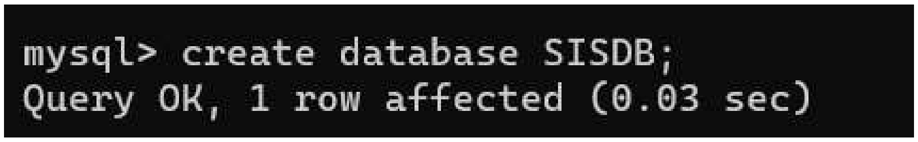
SQL ASSIGNMENT
STUDENT INFORMATION SYSTEM

TASK 1 - Database Design

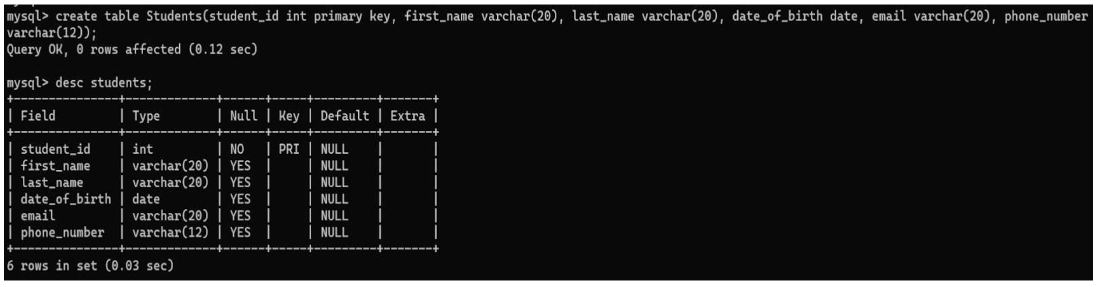
1. ***Create the database named "SISDB"***



```
mysql> create database SISDB;
Query OK, 1 row affected (0.03 sec)
```

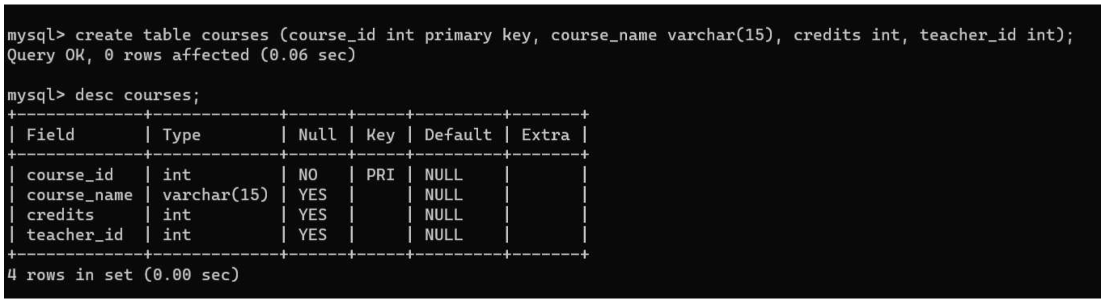
2. ***Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.***

a. Students



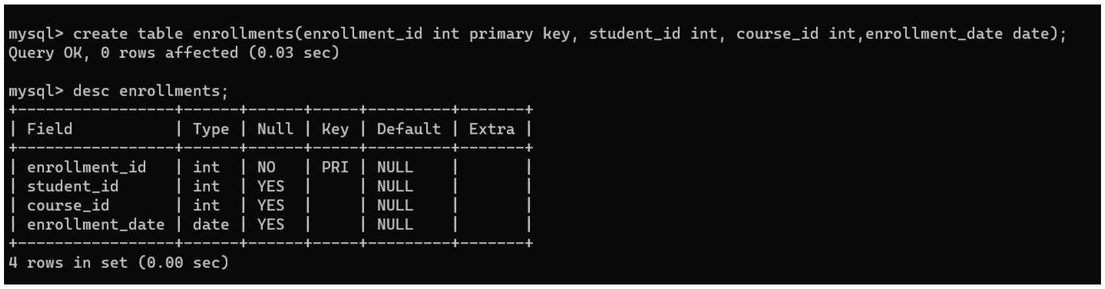
```
mysql> create table Students(student_id int primary key, first_name varchar(20), last_name varchar(20), date_of_birth date, email varchar(20), phone_number
varchar(12));
Query OK, 0 rows affected (0.12 sec)

mysql> desc students;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| student_id    | int         | NO   | PRI | NULL    |       |
| first_name    | varchar(20) | YES  |     | NULL    |       |
| last_name     | varchar(20) | YES  |     | NULL    |       |
| date_of_birth | date        | YES  |     | NULL    |       |
| email         | varchar(20) | YES  |     | NULL    |       |
| phone_number  | varchar(12) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
6 rows in set (0.03 sec)
```

b. Courses



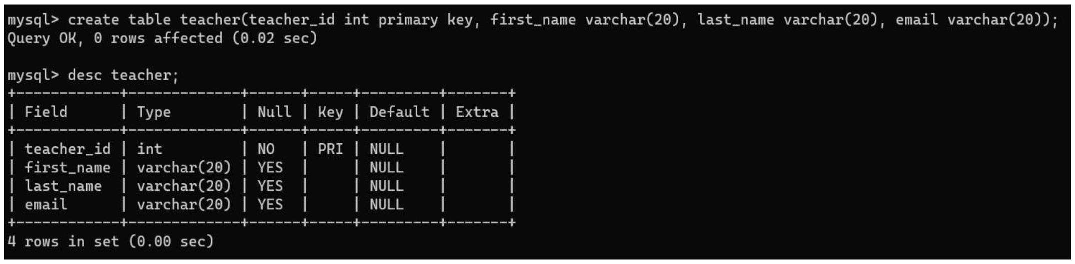
```
mysql> create table courses (course_id int primary key, course_name varchar(15), credits int, teacher_id int);
Query OK, 0 rows affected (0.06 sec)

mysql> desc courses;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| course_id   | int         | NO   | PRI | NULL    |       |
| course_name | varchar(15) | YES  |     | NULL    |       |
| credits     | int         | YES  |     | NULL    |       |
| teacher_id  | int         | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

c. Enrollments



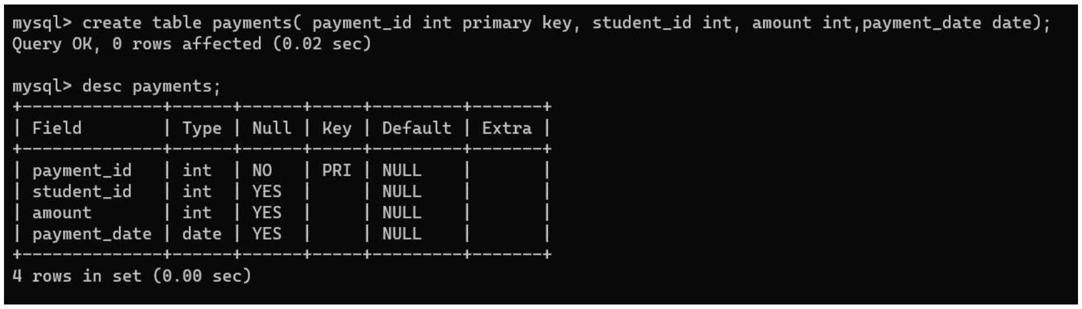
```
mysql> create table enrollments(enrollment_id int primary key, student_id int, course_id int,enrollment_date date);
Query OK, 0 rows affected (0.03 sec)

mysql> desc enrollments;
+-----------------+------+------+-----+---------+-------+
| Field           | Type | Null | Key | Default | Extra |
+-----------------+------+------+-----+---------+-------+
| enrollment_id   | int  | NO   | PRI | NULL    |       |
| student_id      | int  | YES  |     | NULL    |       |
| course_id       | int  | YES  |     | NULL    |       |
| enrollment_date | date | YES  |     | NULL    |       |
+-----------------+------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

d. Teacher



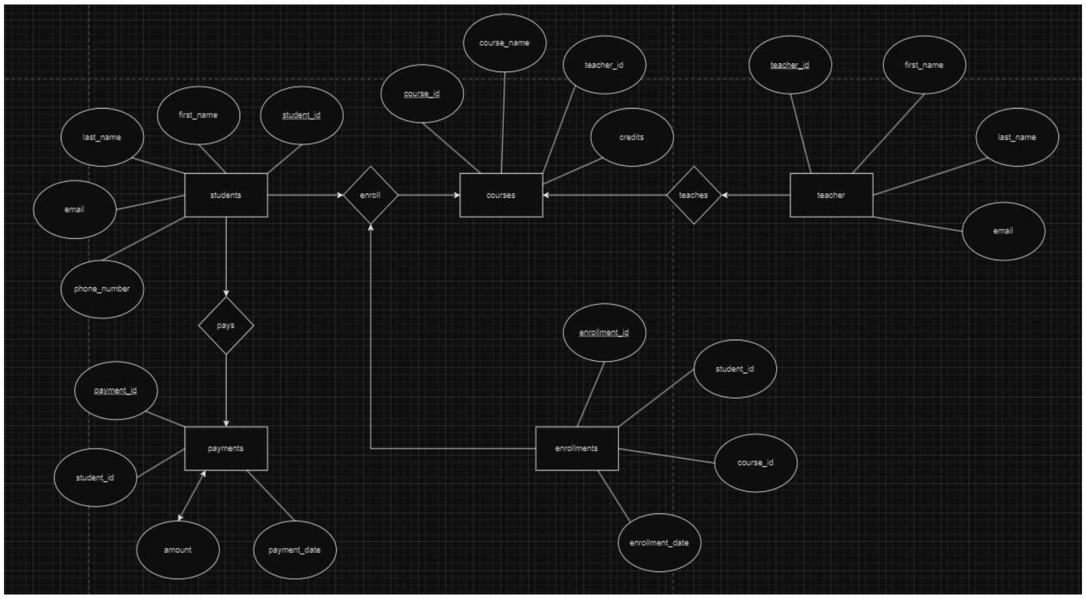
```
mysql> create table teacher(teacher_id int primary key, first_name varchar(20), last_name varchar(20), email varchar(20));
Query OK, 0 rows affected (0.02 sec)

mysql> desc teacher;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| teacher_id | int         | NO   | PRI | NULL    |       |
| first_name | varchar(20) | YES  |     | NULL    |       |
| last_name  | varchar(20) | YES  |     | NULL    |       |
| email      | varchar(20) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

e. Payments



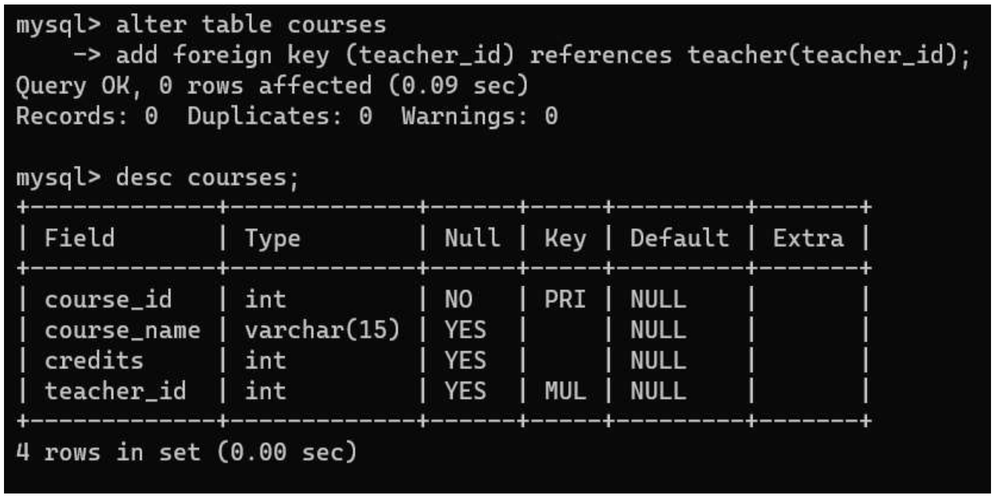
```
mysql> create table payments( payment_id int primary key, student_id int, amount int,payment_date date);
Query OK, 0 rows affected (0.02 sec)

mysql> desc payments;
+--------------+------+------+-----+---------+-------+
| Field        | Type | Null | Key | Default | Extra |
+--------------+------+------+-----+---------+-------+
| payment_id   | int  | NO   | PRI | NULL    |       |
| student_id   | int  | YES  |     | NULL    |       |
| amount       | int  | YES  |     | NULL    |       |
| payment_date | date | YES  |     | NULL    |       |
+--------------+------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

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



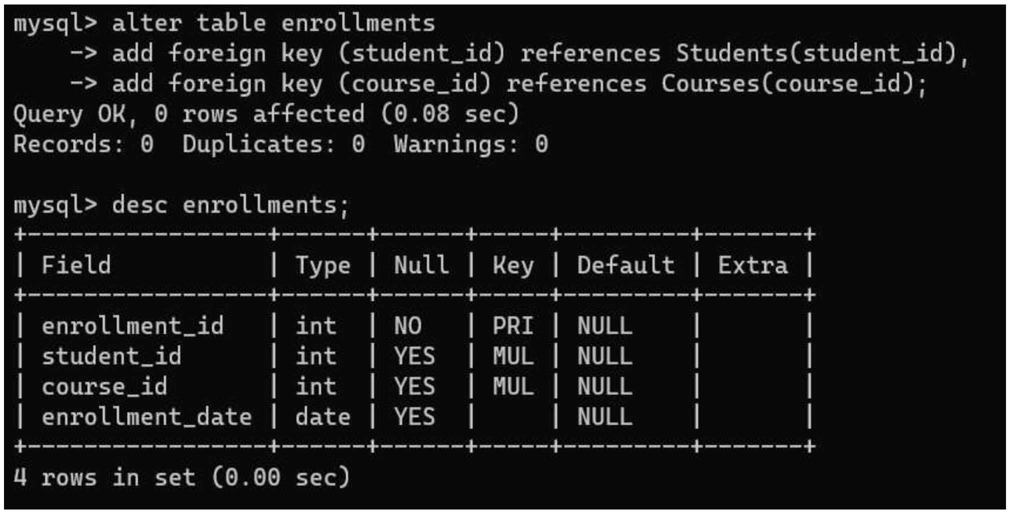
4. ***Create appropriate Primary Key and Foreign Key constraints for referential integrity***

a. Altering Course table



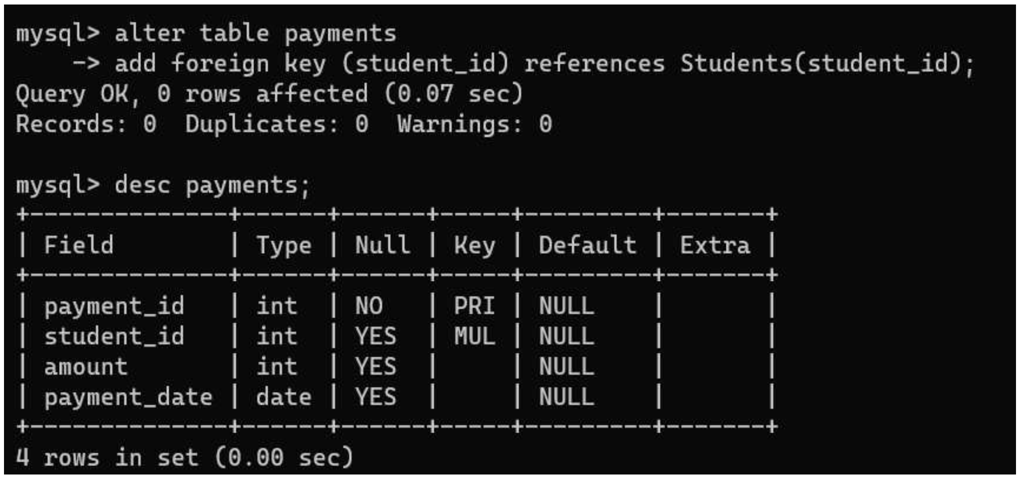
```
mysql> alter table courses
    -> add foreign key (teacher_id) references teacher(teacher_id);
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc courses;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| course_id   | int         | NO   | PRI | NULL    |       |
| course_name | varchar(15) | YES  |     | NULL    |       |
| credits     | int         | YES  |     | NULL    |       |
| teacher_id  | int         | YES  | MUL | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

b. Altering Enrollments table



```
mysql> alter table enrollments
    -> add foreign key (student_id) references Students(student_id),
    -> add foreign key (course_id) references Courses(course_id);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc enrollments;
+-----------------+------+------+-----+---------+-------+
| Field           | Type | Null | Key | Default | Extra |
+-----------------+------+------+-----+---------+-------+
| enrollment_id   | int  | NO   | PRI | NULL    |       |
| student_id      | int  | YES  | MUL | NULL    |       |
| course_id       | int  | YES  | MUL | NULL    |       |
| enrollment_date | date | YES  |     | NULL    |       |
+-----------------+------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

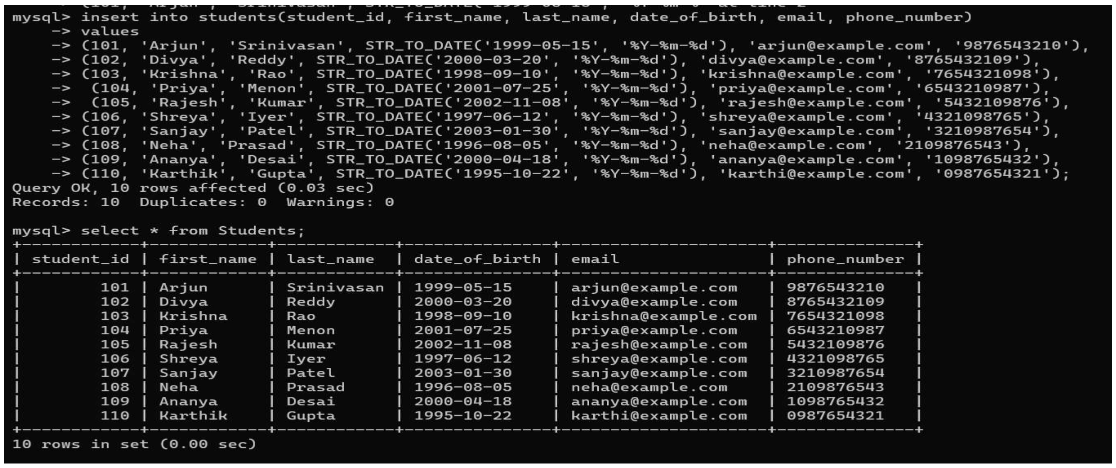
c. Altering payments table



```
mysql> alter table payments
    -> add foreign key (student_id) references Students(student_id);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc payments;
+--------------+------+------+-----+---------+-------+
| Field        | Type | Null | Key | Default | Extra |
+--------------+------+------+-----+---------+-------+
| payment_id   | int  | NO   | PRI | NULL    |       |
| student_id   | int  | YES  | MUL | NULL    |       |
| amount       | int  | YES  |     | NULL    |       |
| payment_date | date | YES  |     | NULL    |       |
+--------------+------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

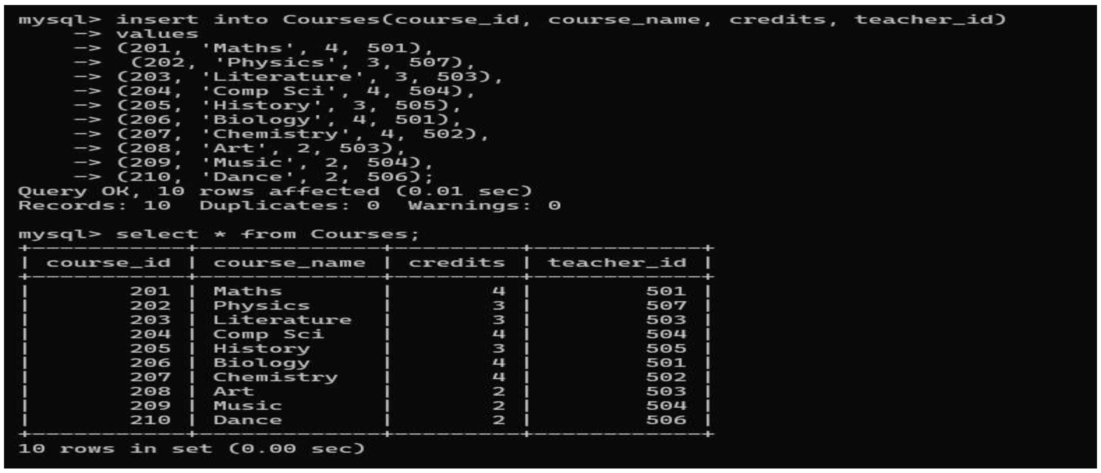
5. Insert at least 10 sample records into each of the following tables.

i. Students



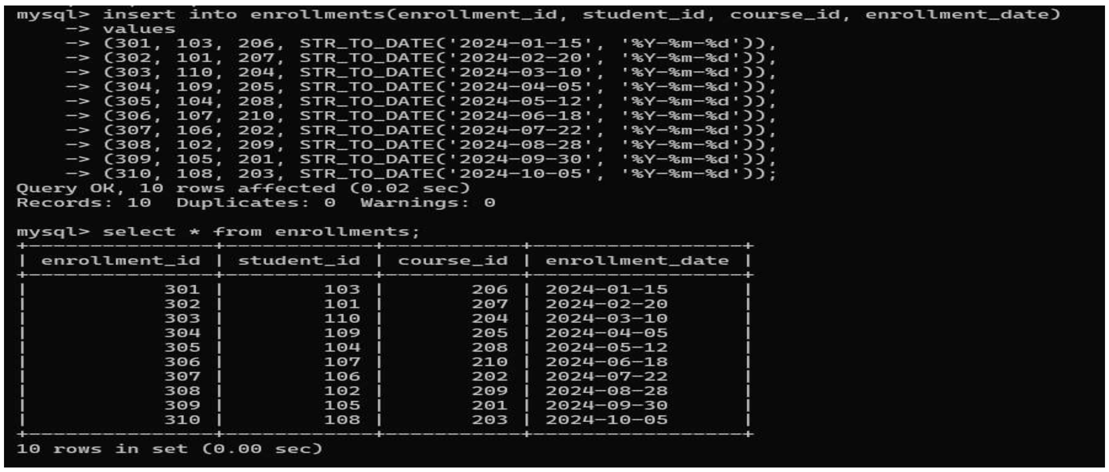
```
mysql> insert into students(student_id, first_name, last_name, date_of_birth, email, phone_number)
    -> values
    -> (101, 'Arjun', 'Srinivasan', STR_TO_DATE('1999-05-15', '%Y-%m-%d'), 'arjun@example.com', '9876543210'),
    -> (102, 'Divya', 'Reddy', STR_TO_DATE('2000-03-20', '%Y-%m-%d'), 'divya@example.com', '8765432109'),
    -> (103, 'Krishna', 'Rao', STR_TO_DATE('1998-09-10', '%Y-%m-%d'), 'krishna@example.com', '7654321098'),
    ->  (104, 'Priya', 'Menon', STR_TO_DATE('2001-07-25', '%Y-%m-%d'), 'priya@example.com', '6543210987'),
    ->  (105, 'Rajesh', 'Kumar', STR_TO_DATE('2002-11-08', '%Y-%m-%d'), 'rajesh@example.com', '5432109876'),
    -> (106, 'Shreya', 'Iyer', STR_TO_DATE('1997-06-12', '%Y-%m-%d'), 'shreya@example.com', '4321098765'),
    -> (107, 'Sanjay', 'Patel', STR_TO_DATE('2003-01-30', '%Y-%m-%d'), 'sanjay@example.com', '3210987654'),
    -> (108, 'Neha', 'Prasad', STR_TO_DATE('1996-08-05', '%Y-%m-%d'), 'neha@example.com', '2109876543'),
    -> (109, 'Ananya', 'Desai', STR_TO_DATE('2000-04-18', '%Y-%m-%d'), 'ananya@example.com', '1098765432'),
    -> (110, 'Karthik', 'Gupta', STR_TO_DATE('1995-10-22', '%Y-%m-%d'), 'karthi@example.com', '0987654321');
Query OK, 10 rows affected (0.03 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Students;
+------------+------------+------------+---------------+---------------------+--------------+
| student_id | first_name | last_name  | date_of_birth | email               | phone_number |
+------------+------------+------------+---------------+---------------------+--------------+
|        101 | Arjun      | Srinivasan | 1999-05-15    | arjun@example.com   | 9876543210   |
|        102 | Divya      | Reddy      | 2000-03-20    | divya@example.com   | 8765432109   |
|        103 | Krishna    | Rao        | 1998-09-10    | krishna@example.com | 7654321098   |
|        104 | Priya      | Menon      | 2001-07-25    | priya@example.com   | 6543210987   |
|        105 | Rajesh     | Kumar      | 2002-11-08    | rajesh@example.com  | 5432109876   |
|        106 | Shreya     | Iyer       | 1997-06-12    | shreya@example.com  | 4321098765   |
|        107 | Sanjay     | Patel      | 2003-01-30    | sanjay@example.com  | 3210987654   |
|        108 | Neha       | Prasad     | 1996-08-05    | neha@example.com    | 2109876543   |
|        109 | Ananya     | Desai      | 2000-04-18    | ananya@example.com  | 1098765432   |
|        110 | Karthik    | Gupta      | 1995-10-22    | karthi@example.com  | 0987654321   |
+------------+------------+------------+---------------+---------------------+--------------+
10 rows in set (0.00 sec)
```

ii. Courses



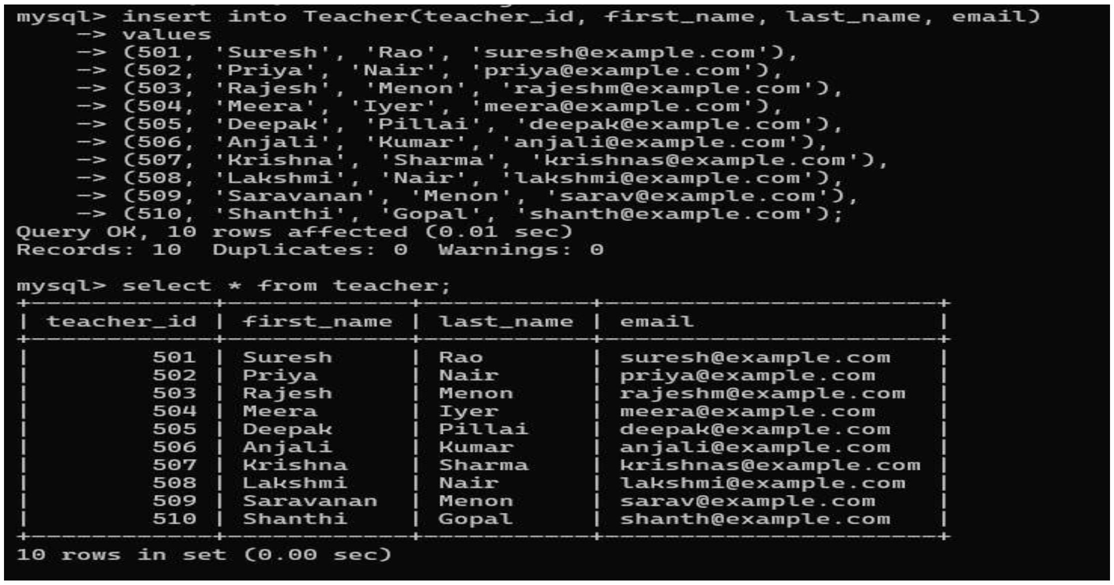
```
mysql> insert into Courses(course_id, course_name, credits, teacher_id)
    -> values
    -> (201, 'Maths', 4, 501),
    ->  (202, 'Physics', 3, 507),
    -> (203, 'Literature', 3, 503),
    -> (204, 'Comp Sci', 4, 504),
    -> (205, 'History', 3, 505),
    -> (206, 'Biology', 4, 501),
    -> (207, 'Chemistry', 4, 502),
    -> (208, 'Art', 2, 503),
    -> (209, 'Music', 2, 504),
    -> (210, 'Dance', 2, 506);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Courses;
+-----------+-------------+---------+------------+
| course_id | course_name | credits | teacher_id |
+-----------+-------------+---------+------------+
|       201 | Maths       |       4 |        501 |
|       202 | Physics     |       3 |        507 |
|       203 | Literature  |       3 |        503 |
|       204 | Comp Sci    |       4 |        504 |
|       205 | History     |       3 |        505 |
|       206 | Biology     |       4 |        501 |
|       207 | Chemistry   |       4 |        502 |
|       208 | Art         |       2 |        503 |
|       209 | Music       |       2 |        504 |
|       210 | Dance       |       2 |        506 |
+-----------+-------------+---------+------------+
10 rows in set (0.00 sec)
```

iii. Enrollments



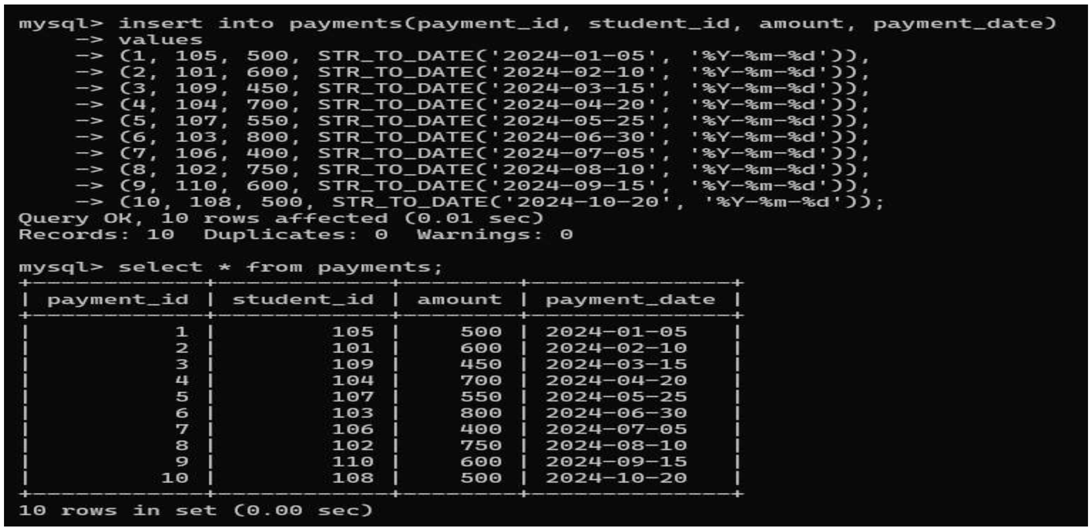
```
mysql> insert into enrollments(enrollment_id, student_id, course_id, enrollment_date)
    -> values
    -> (301, 103, 206, STR_TO_DATE('2024-01-15', '%Y-%m-%d')),
    -> (302, 101, 207, STR_TO_DATE('2024-02-20', '%Y-%m-%d')),
    -> (303, 110, 204, STR_TO_DATE('2024-03-10', '%Y-%m-%d')),
    -> (304, 109, 205, STR_TO_DATE('2024-04-05', '%Y-%m-%d')),
    -> (305, 104, 208, STR_TO_DATE('2024-05-12', '%Y-%m-%d')),
    -> (306, 107, 210, STR_TO_DATE('2024-06-18', '%Y-%m-%d')),
    -> (307, 106, 202, STR_TO_DATE('2024-07-22', '%Y-%m-%d')),
    -> (308, 102, 209, STR_TO_DATE('2024-08-28', '%Y-%m-%d')),
    -> (309, 105, 201, STR_TO_DATE('2024-09-30', '%Y-%m-%d')),
    -> (310, 108, 203, STR_TO_DATE('2024-10-05', '%Y-%m-%d'));
Query OK, 10 rows affected (0.02 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           301 |        103 |       206 | 2024-01-15      |
|           302 |        101 |       207 | 2024-02-20      |
|           303 |        110 |       204 | 2024-03-10      |
|           304 |        109 |       205 | 2024-04-05      |
|           305 |        104 |       208 | 2024-05-12      |
|           306 |        107 |       210 | 2024-06-18      |
|           307 |        106 |       202 | 2024-07-22      |
|           308 |        102 |       209 | 2024-08-28      |
|           309 |        105 |       201 | 2024-09-30      |
|           310 |        108 |       203 | 2024-10-05      |
+---------------+------------+-----------+-----------------+
10 rows in set (0.00 sec)
```

iv. Teacher



```
mysql> insert into Teacher(teacher_id, first_name, last_name, email)
    -> values
    -> (501, 'Suresh', 'Rao', 'suresh@example.com'),
    -> (502, 'Priya', 'Nair', 'priya@example.com'),
    -> (503, 'Rajesh', 'Menon', 'rajeshm@example.com'),
    -> (504, 'Meera', 'Iyer', 'meera@example.com'),
    -> (505, 'Deepak', 'Pillai', 'deepak@example.com'),
    -> (506, 'Anjali', 'Kumar', 'anjali@example.com'),
    -> (507, 'Krishna', 'Sharma', 'krishnas@example.com'),
    -> (508, 'Lakshmi', 'Nair', 'lakshmi@example.com'),
    -> (509, 'Saravanan', 'Menon', 'sarav@example.com'),
    -> (510, 'Shanthi', 'Gopal', 'shanth@example.com');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from teacher;
+------------+------------+-----------+----------------------+
| teacher_id | first_name | last_name | email                |
+------------+------------+-----------+----------------------+
|        501 | Suresh     | Rao       | suresh@example.com   |
|        502 | Priya      | Nair      | priya@example.com    |
|        503 | Rajesh     | Menon     | rajeshm@example.com  |
|        504 | Meera      | Iyer      | meera@example.com    |
|        505 | Deepak     | Pillai    | deepak@example.com   |
|        506 | Anjali     | Kumar     | anjali@example.com   |
|        507 | Krishna    | Sharma    | krishnas@example.com |
|        508 | Lakshmi    | Nair      | lakshmi@example.com  |
|        509 | Saravanan  | Menon     | sarav@example.com    |
|        510 | Shanthi    | Gopal     | shanth@example.com   |
+------------+------------+-----------+----------------------+
10 rows in set (0.00 sec)
```

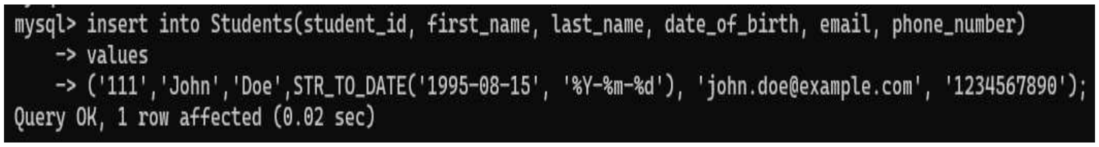
v. Payments



```
mysql> insert into payments(payment_id, student_id, amount, payment_date)
    -> values
    -> (1, 105, 500, STR_TO_DATE('2024-01-05', '%Y-%m-%d')),
    -> (2, 101, 600, STR_TO_DATE('2024-02-10', '%Y-%m-%d')),
    -> (3, 109, 450, STR_TO_DATE('2024-03-15', '%Y-%m-%d')),
    -> (4, 104, 700, STR_TO_DATE('2024-04-20', '%Y-%m-%d')),
    -> (5, 107, 550, STR_TO_DATE('2024-05-25', '%Y-%m-%d')),
    -> (6, 103, 800, STR_TO_DATE('2024-06-30', '%Y-%m-%d')),
    -> (7, 106, 400, STR_TO_DATE('2024-07-05', '%Y-%m-%d')),
    -> (8, 102, 750, STR_TO_DATE('2024-08-10', '%Y-%m-%d')),
    -> (9, 110, 600, STR_TO_DATE('2024-09-15', '%Y-%m-%d')),
    -> (10, 108, 500, STR_TO_DATE('2024-10-20', '%Y-%m-%d'));
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from payments;
+------------+------------+--------+--------------+
| payment_id | student_id | amount | payment_date |
+------------+------------+--------+--------------+
|          1 |        105 |    500 | 2024-01-05   |
|          2 |        101 |    600 | 2024-02-10   |
|          3 |        109 |    450 | 2024-03-15   |
|          4 |        104 |    700 | 2024-04-20   |
|          5 |        107 |    550 | 2024-05-25   |
|          6 |        103 |    800 | 2024-06-30   |
|          7 |        106 |    400 | 2024-07-05   |
|          8 |        102 |    750 | 2024-08-10   |
|          9 |        110 |    600 | 2024-09-15   |
|         10 |        108 |    500 | 2024-10-20   |
+------------+------------+--------+--------------+
10 rows in set (0.00 sec)
```

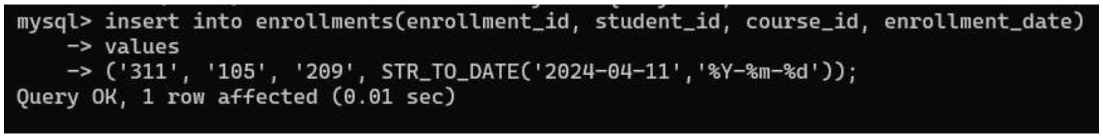
TASK 2 - Select, Where, Between, AND, LIKE:

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

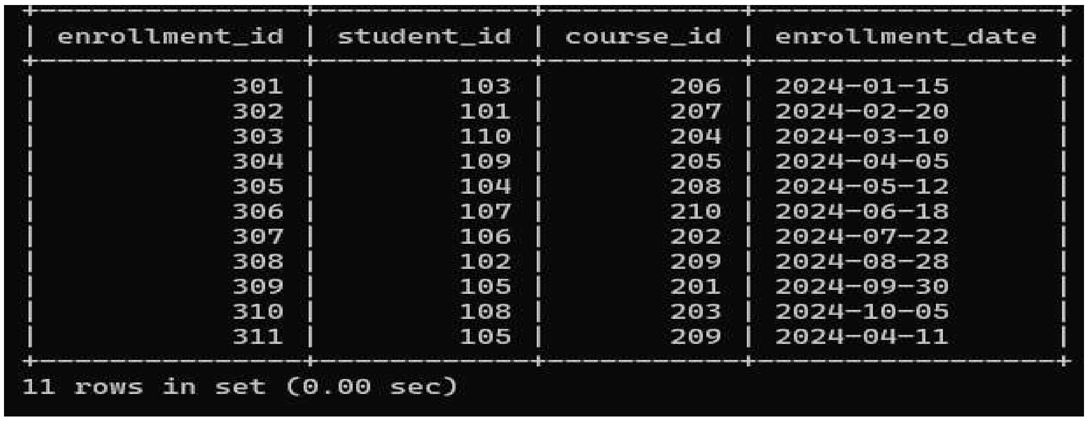


```
mysql> insert into Students(student_id, first_name, last_name, date_of_birth, email, phone_number)
    -> values
    -> ('111','John','Doe',STR_TO_DATE('1995-08-15', '%Y-%m-%d'), 'john.doe@example.com', '1234567890');
Query OK, 1 row affected (0.02 sec)
```

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.***

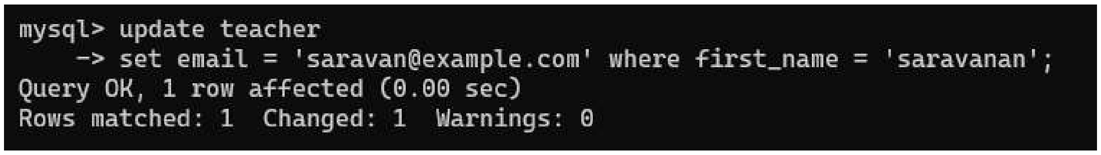


```
mysql> insert into enrollments(enrollment_id, student_id, course_id, enrollment_date)
    -> values
    -> ('311', '105', '209', STR_TO_DATE('2024-04-11','%Y-%m-%d'));
Query OK, 1 row affected (0.01 sec)
```

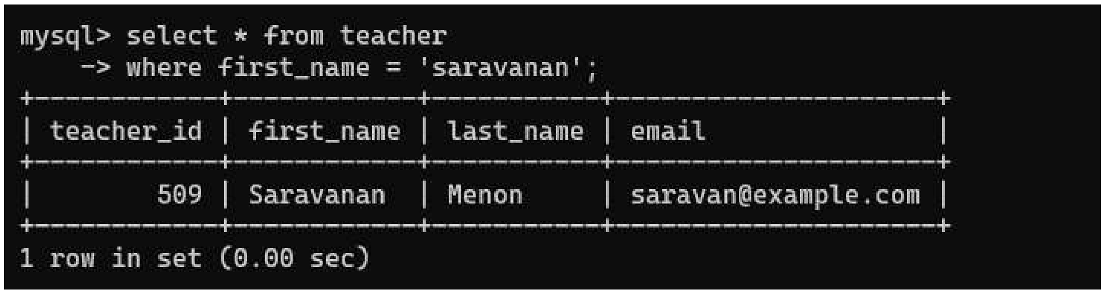


```
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           301 |        103 |       206 | 2024-01-15      |
|           302 |        101 |       207 | 2024-02-20      |
|           303 |        110 |       204 | 2024-03-10      |
|           304 |        109 |       205 | 2024-04-05      |
|           305 |        104 |       208 | 2024-05-12      |
|           306 |        107 |       210 | 2024-06-18      |
|           307 |        106 |       202 | 2024-07-22      |
|           308 |        102 |       209 | 2024-08-28      |
|           309 |        105 |       201 | 2024-09-30      |
|           310 |        108 |       203 | 2024-10-05      |
|           311 |        105 |       209 | 2024-04-11      |
+---------------+------------+-----------+-----------------+
11 rows in set (0.00 sec)
```

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

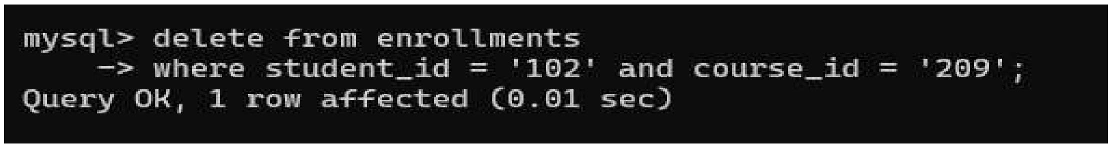


```
mysql> update teacher
    -> set email = 'saravan@example.com' where first_name = 'saravanan';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

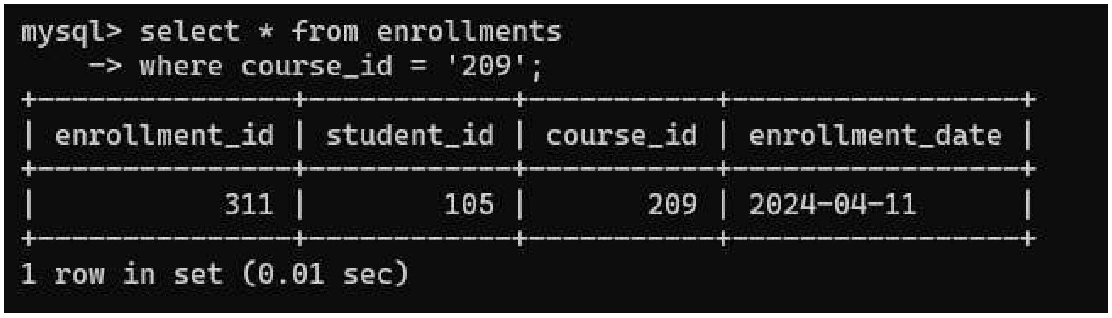


```
mysql> select * from teacher
    -> where first_name = 'saravanan';
+------------+------------+-----------+---------------------+
| teacher_id | first_name | last_name | email               |
+------------+------------+-----------+---------------------+
|        509 | Saravanan  | Menon     | saravan@example.com |
+------------+------------+-----------+---------------------+
1 row in set (0.00 sec)
```

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.***

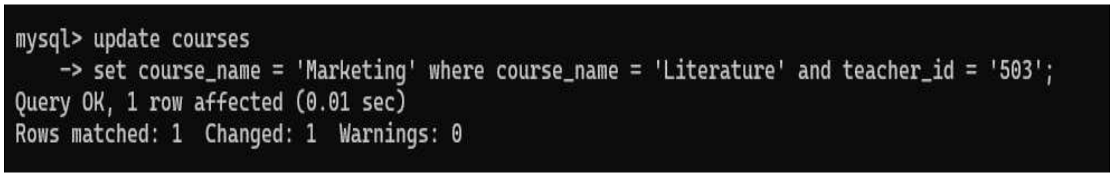


```
mysql> delete from enrollments
    -> where student_id = '102' and course_id = '209';
Query OK, 1 row affected (0.01 sec)
```

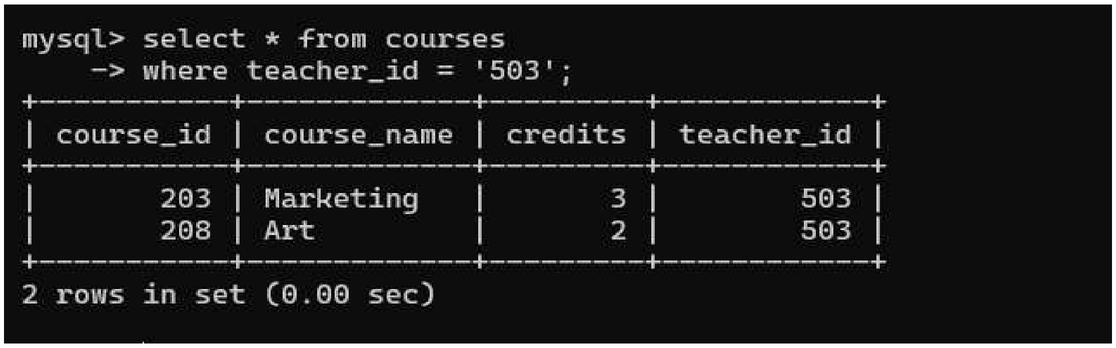


```
mysql> select * from enrollments
    -> where course_id = '209';
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           311 |        105 |       209 | 2024-04-11      |
+---------------+------------+-----------+-----------------+
1 row in set (0.01 sec)
```

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

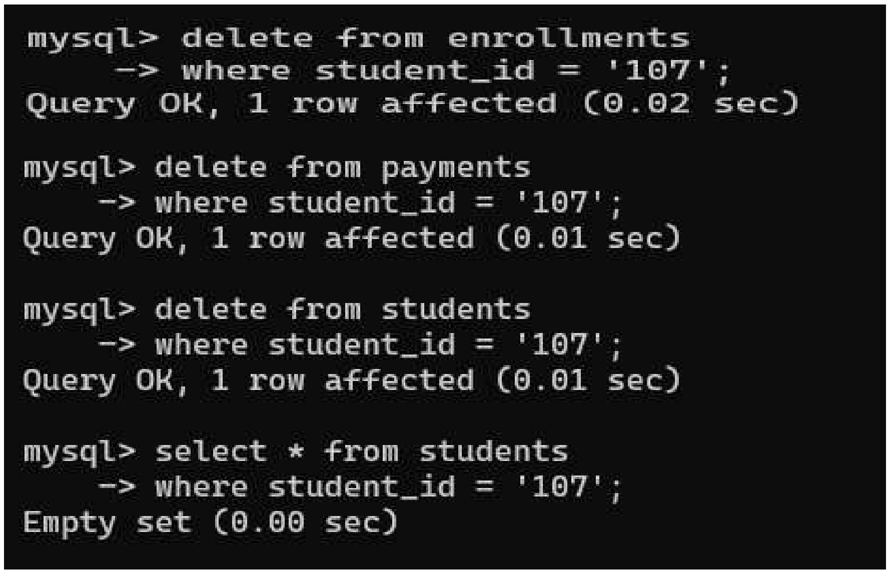


```
mysql> update courses
    -> set course_name = 'Marketing' where course_name = 'Literature' and teacher_id = '503';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```



```
mysql> select * from courses
    -> where teacher_id = '503';
+-----------+-------------+---------+------------+
| course_id | course_name | credits | teacher_id |
+-----------+-------------+---------+------------+
|       203 | Marketing   |       3 |        503 |
|       208 | Art         |       2 |        503 |
+-----------+-------------+---------+------------+
2 rows in set (0.00 sec)
```

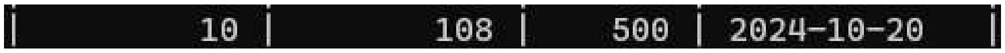
6. ***Delete a specific student from the "Students" table and remove all their enrollment records from the "Enrollments" table. Be sure to maintain referential integrity.***



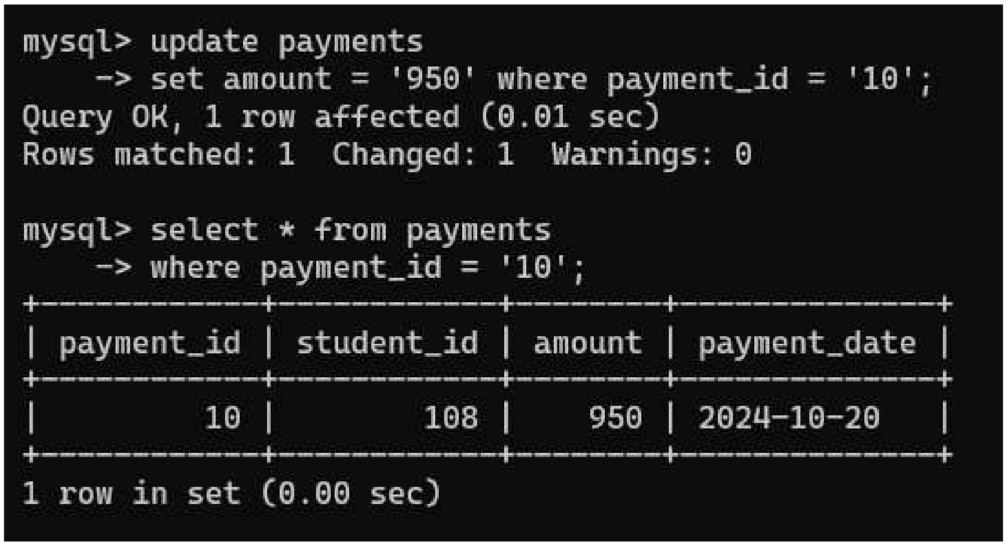
```
mysql> delete from enrollments
    -> where student_id = '107';
Query OK, 1 row affected (0.02 sec)

mysql> delete from payments
    -> where student_id = '107';
Query OK, 1 row affected (0.01 sec)

mysql> delete from students
    -> where student_id = '107';
Query OK, 1 row affected (0.01 sec)

mysql> select * from students
    -> where student_id = '107';
Empty set (0.00 sec)
```

Referential integrity maintained. Student_id has foreign key reference with the enrollments table and payments table.

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



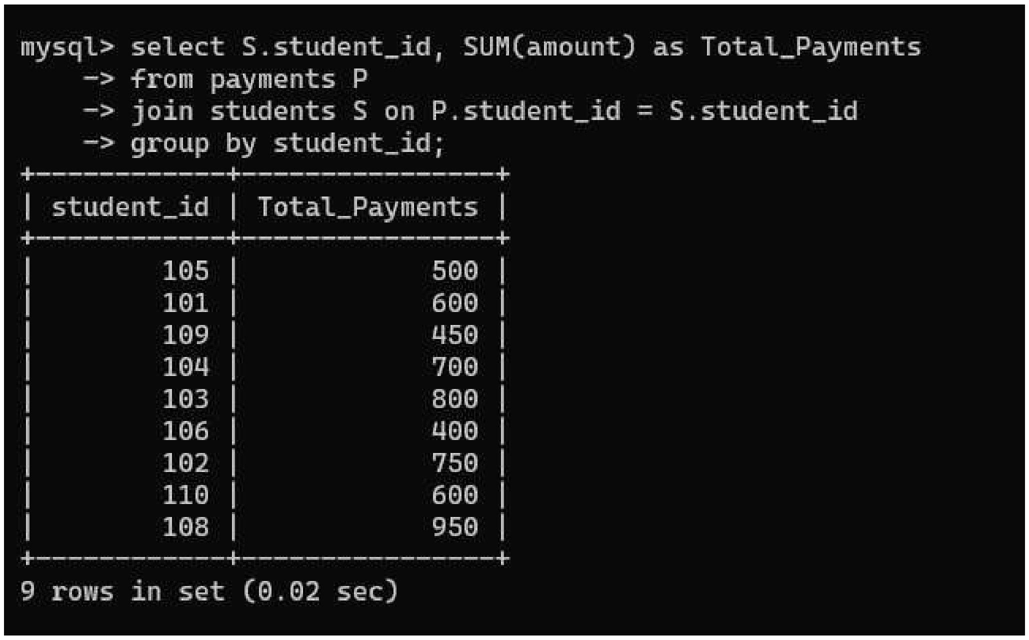
```
|          10 |         108 |    500 | 2024-10-20   |
```



```
mysql> update payments
    -> set amount = '950' where payment_id = '10';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from payments
    -> where payment_id = '10';
+------------+------------+--------+--------------+
| payment_id | student_id | amount | payment_date |
+------------+------------+--------+--------------+
|         10 |        108 |    950 | 2024-10-20   |
+------------+------------+--------+--------------+
1 row in set (0.00 sec)
```

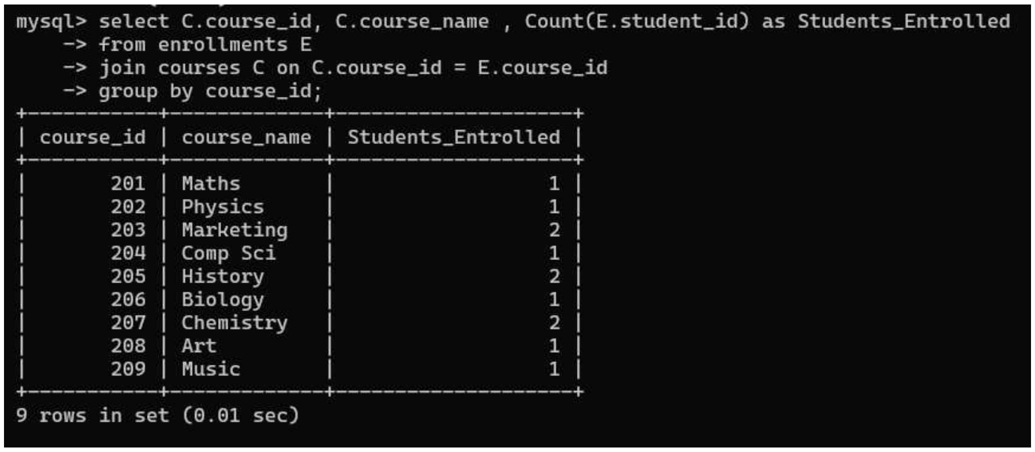
TASK 3 - . Aggregate functions, Having, Order By, GroupBy and Joins:

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.***



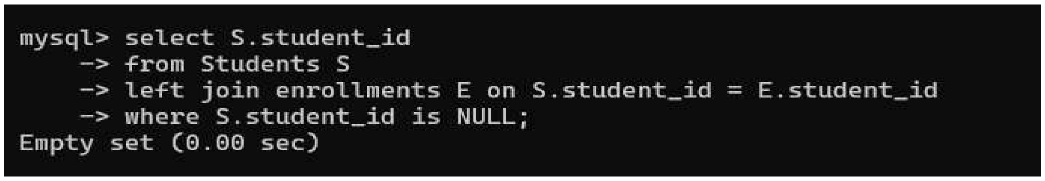
```
mysql> select S.student_id, SUM(amount) as Total_Payments
    -> from payments P
    -> join students S on P.student_id = S.student_id
    -> group by student_id;
+------------+----------------+
| student_id | Total_Payments |
+------------+----------------+
|        105 |            500 |
|        101 |            600 |
|        109 |            450 |
|        104 |            700 |
|        103 |            800 |
|        106 |            400 |
|        102 |            750 |
|        110 |            600 |
|        108 |            950 |
+------------+----------------+
9 rows in set (0.02 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.***



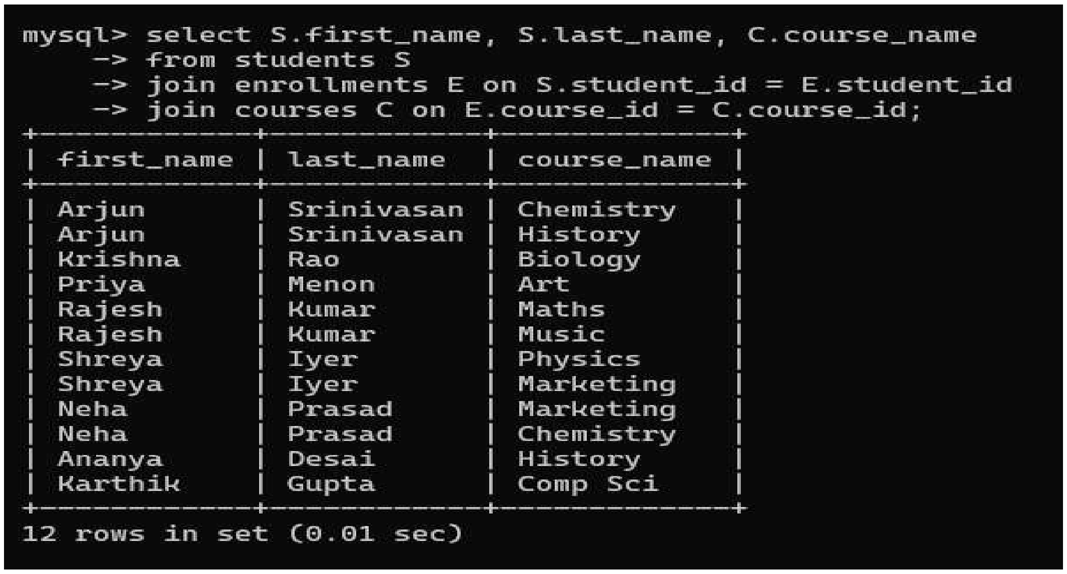
```
mysql> select C.course_id, C.course_name , Count(E.student_id) as Students_Entrolled
    -> from enrollments E
    -> join courses C on C.course_id = E.course_id
    -> group by course_id;
+-----------+-------------+--------------------+
| course_id | course_name | Students_Entrolled |
+-----------+-------------+--------------------+
|       201 | Maths       |                  1 |
|       202 | Physics     |                  1 |
|       203 | Marketing   |                  2 |
|       204 | Comp Sci    |                  1 |
|       205 | History     |                  2 |
|       206 | Biology     |                  1 |
|       207 | Chemistry   |                  2 |
|       208 | Art         |                  1 |
|       209 | Music       |                  1 |
+-----------+-------------+--------------------+
9 rows in set (0.01 sec)
```

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***



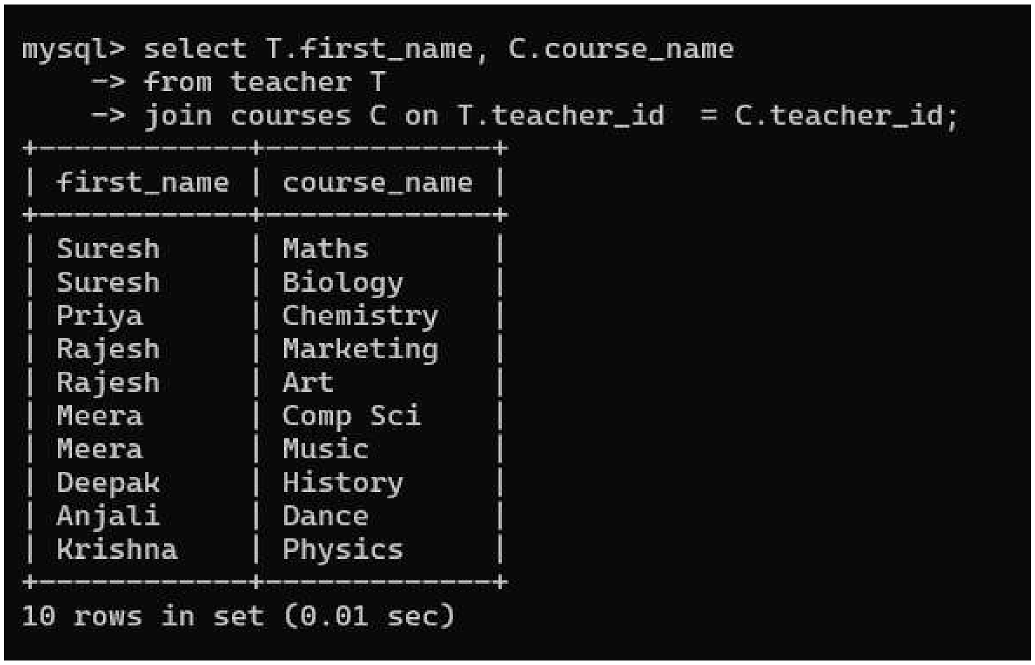
```
mysql> select S.student_id
    -> from Students S
    -> left join enrollments E on S.student_id = E.student_id
    -> where S.student_id is NULL;
Empty set (0.00 sec)
```

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.***



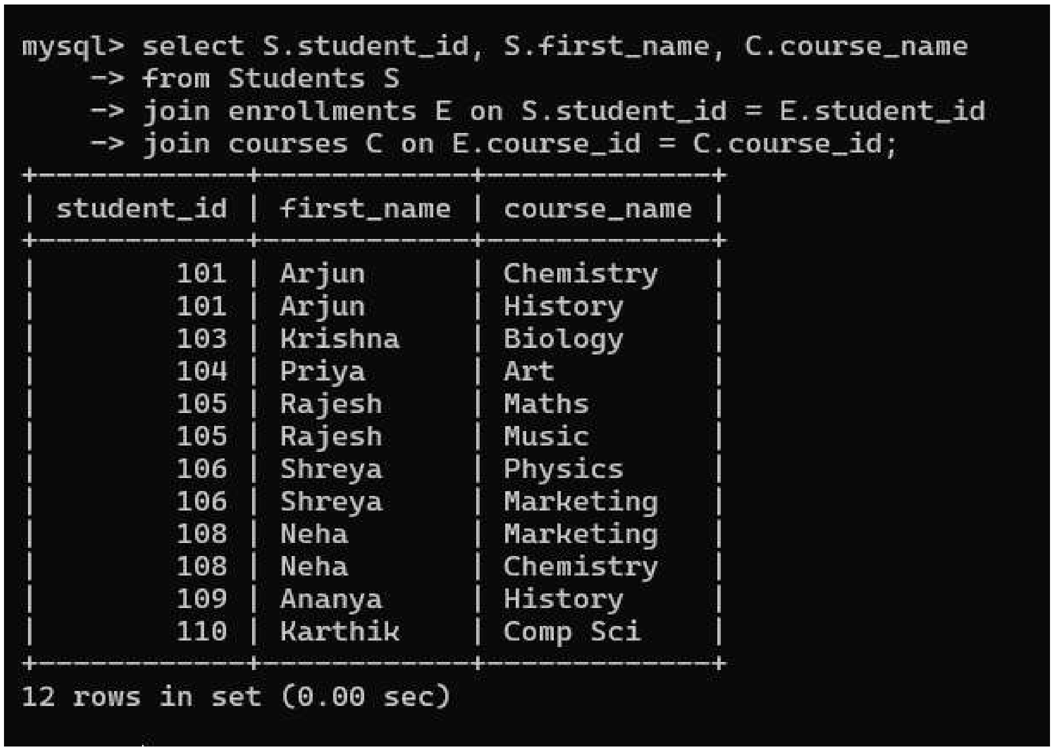
```
mysql> select S.first_name, S.last_name, C.course_name
    -> from students S
    -> join enrollments E on S.student_id = E.student_id
    -> join courses C on E.course_id = C.course_id;
+------------+------------+-------------+
| first_name | last_name  | course_name |
+------------+------------+-------------+
| Arjun      | Srinivasan | Chemistry   |
| Arjun      | Srinivasan | History     |
| Krishna    | Rao        | Biology     |
| Priya      | Menon      | Art         |
| Rajesh     | Kumar      | Maths       |
| Rajesh     | Kumar      | Music       |
| Shreya     | Iyer       | Physics     |
| Shreya     | Iyer       | Marketing   |
| Neha       | Prasad     | Marketing   |
| Neha       | Prasad     | Chemistry   |
| Ananya     | Desai      | History     |
| Karthik    | Gupta      | Comp Sci    |
+------------+------------+-------------+
12 rows in set (0.01 sec)
```

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.***



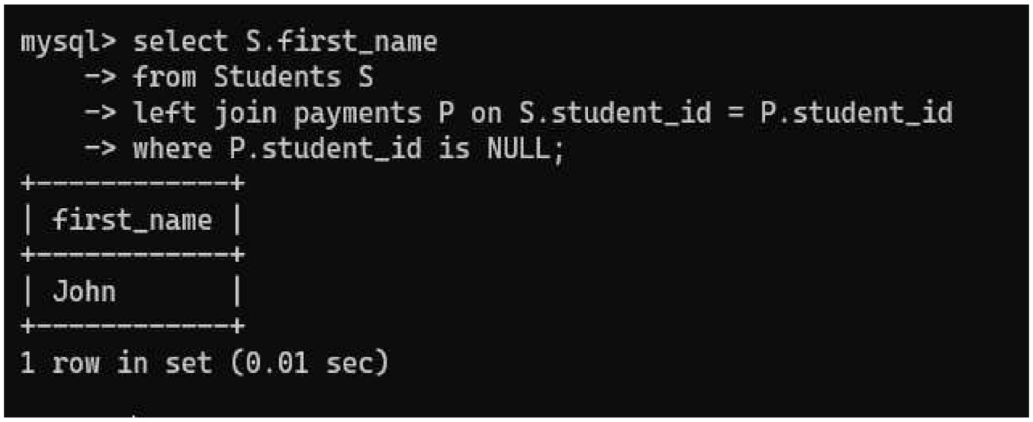
```
mysql> select T.first_name, C.course_name
    -> from teacher T
    -> join courses C on T.teacher_id  = C.teacher_id;
+------------+-------------+
| first_name | course_name |
+------------+-------------+
| Suresh     | Maths       |
| Suresh     | Biology     |
| Priya      | Chemistry   |
| Rajesh     | Marketing   |
| Rajesh     | Art         |
| Meera      | Comp Sci    |
| Meera      | Music       |
| Deepak     | History     |
| Anjali     | Dance       |
| Krishna    | Physics     |
+------------+-------------+
10 rows in set (0.01 sec)
```

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.***



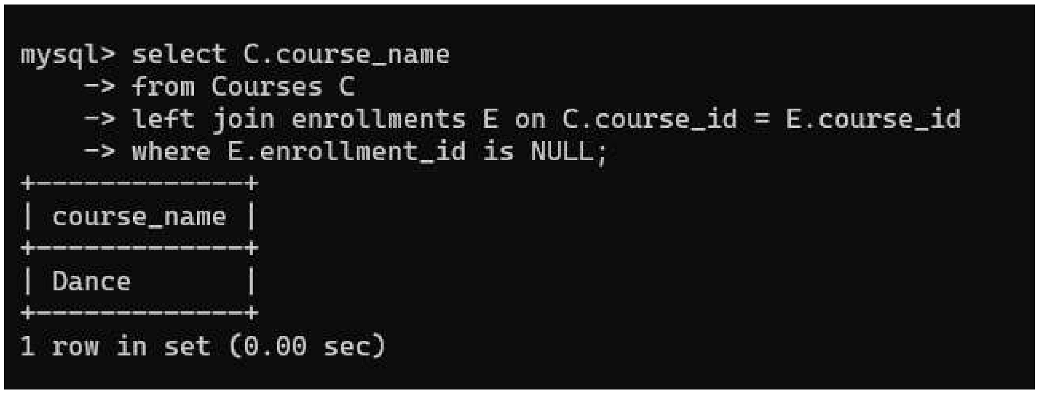
```
mysql> select S.student_id, S.first_name, C.course_name
    -> from Students S
    -> join enrollments E on S.student_id = E.student_id
    -> join courses C on E.course_id = C.course_id;
+------------+------------+-------------+
| student_id | first_name | course_name |
+------------+------------+-------------+
|        101 | Arjun      | Chemistry   |
|        101 | Arjun      | History     |
|        103 | Krishna    | Biology     |
|        104 | Priya      | Art         |
|        105 | Rajesh     | Maths       |
|        105 | Rajesh     | Music       |
|        106 | Shreya     | Physics     |
|        106 | Shreya     | Marketing   |
|        108 | Neha       | Marketing   |
|        108 | Neha       | Chemistry   |
|        109 | Ananya     | History     |
|        110 | Karthik    | Comp Sci    |
+------------+------------+-------------+
12 rows in set (0.00 sec)
```

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.***

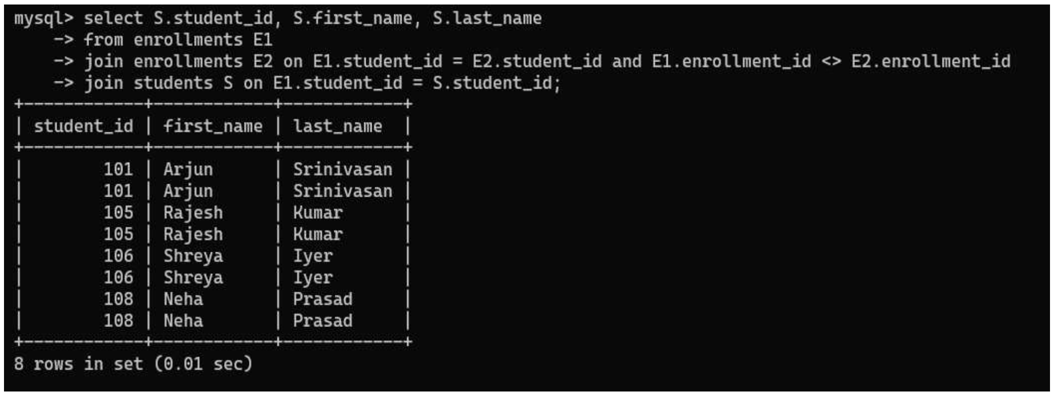


```
mysql> select S.first_name
    -> from Students S
    -> left join payments P on S.student_id = P.student_id
    -> where P.student_id is NULL;
+------------+
| first_name |
+------------+
| John       |
+------------+
1 row in set (0.01 sec)
```

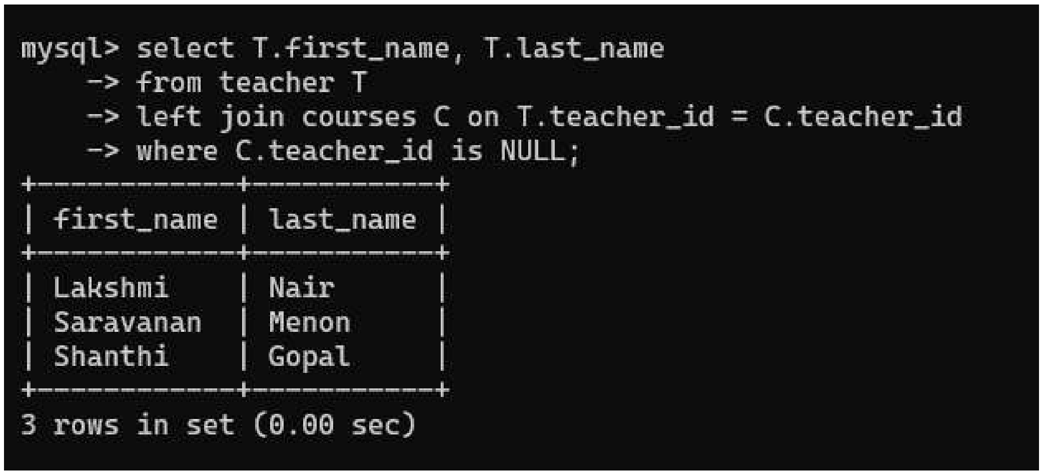
8. ***Write a query to identify courses that have no enrollments. You'll need to use a LEFT JOIN between the "Courses" table and the "Enrollments" table and filter for courses with NULL enrollment records.***


```
mysql> select C.course_name
    -> from Courses C
    -> left join enrollments E on C.course_id = E.course_id
    -> where E.enrollment_id is NULL;
+-------------+
| course_name |
+-------------+
| Dance       |
+-------------+
1 row in 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.***

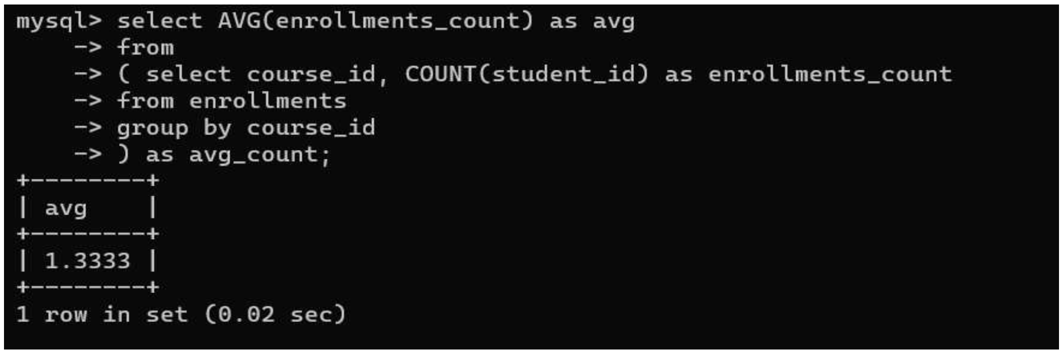

```
mysql> select S.student_id, S.first_name, S.last_name
    -> from enrollments E1
    -> join enrollments E2 on E1.student_id = E2.student_id and E1.enrollment_id <> E2.enrollment_id
    -> join students S on E1.student_id = S.student_id;
+------------+------------+------------+
| student_id | first_name | last_name  |
+------------+------------+------------+
|        101 | Arjun      | Srinivasan |
|        101 | Arjun      | Srinivasan |
|        105 | Rajesh     | Kumar      |
|        105 | Rajesh     | Kumar      |
|        106 | Shreya     | Iyer       |
|        106 | Shreya     | Iyer       |
|        108 | Neha       | Prasad     |
|        108 | Neha       | Prasad     |
+------------+------------+------------+
8 rows in set (0.01 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.***


```
mysql> select T.first_name, T.last_name
    -> from teacher T
    -> left join courses C on T.teacher_id = C.teacher_id
    -> where C.teacher_id is NULL;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| Lakshmi    | Nair      |
| Saravanan  | Menon     |
| Shanthi    | Gopal     |
+------------+-----------+
3 rows in set (0.00 sec)
```

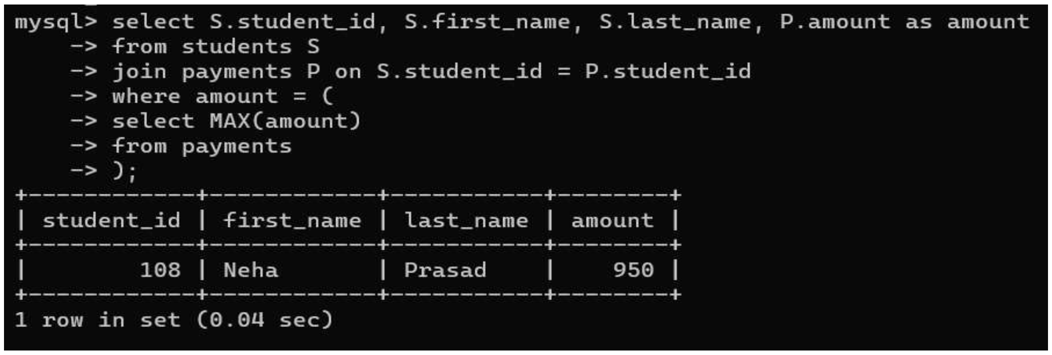
TASK 4 - Subquery and its type:

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



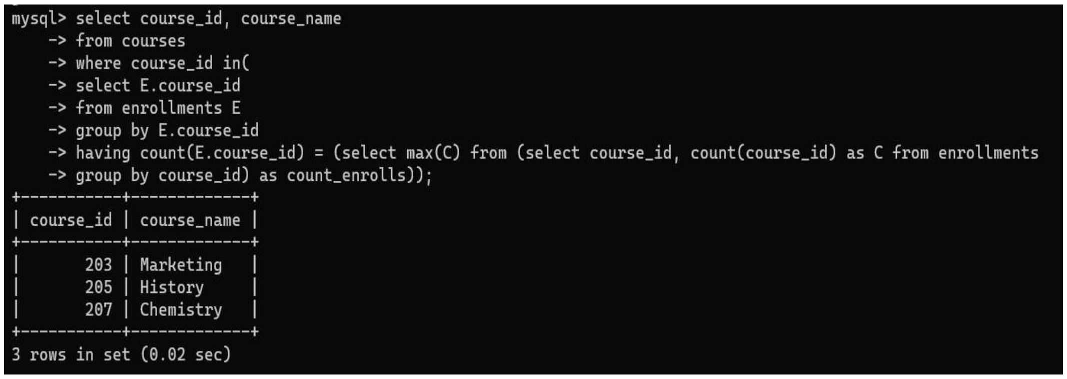
```
mysql> select AVG(enrollments_count) as avg
    -> from
    -> ( select course_id, COUNT(student_id) as enrollments_count
    -> from enrollments
    -> group by course_id
    -> ) as avg_count;
+--------+
| avg    |
+--------+
| 1.3333 |
+--------+
1 row in set (0.02 sec)
```

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.***



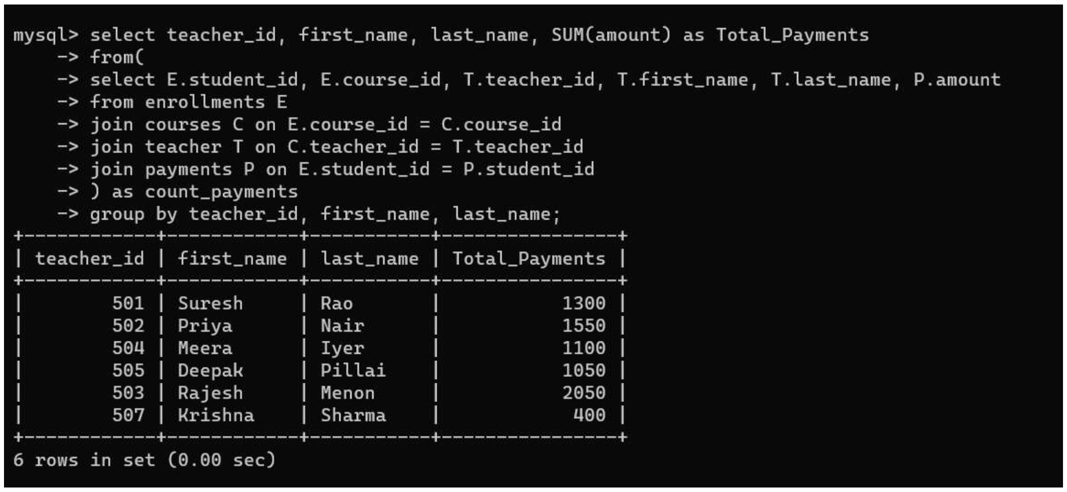
```
mysql> select S.student_id, S.first_name, S.last_name, P.amount as amount
    -> from students S
    -> join payments P on S.student_id = P.student_id
    -> where amount = (
    -> select MAX(amount)
    -> from payments
    -> );
+------------+------------+-----------+--------+
| student_id | first_name | last_name | amount |
+------------+------------+-----------+--------+
|        108 | Neha       | Prasad    |    950 |
+------------+------------+-----------+--------+
1 row in set (0.04 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.***



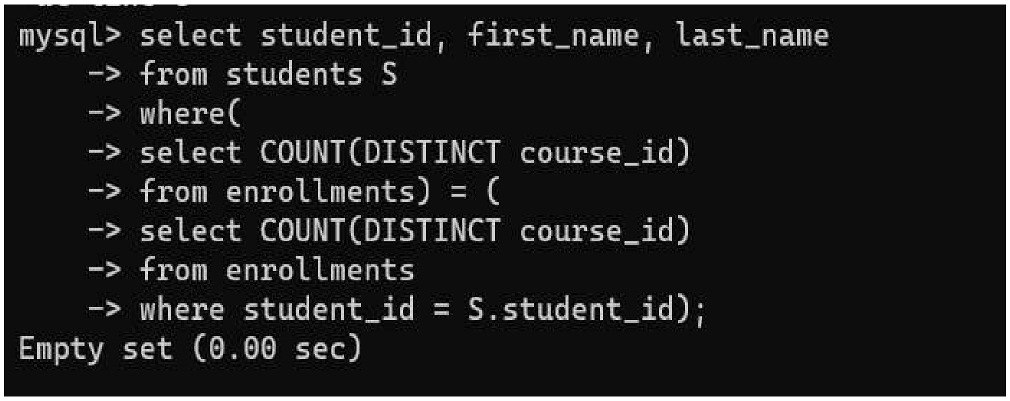
```
mysql> select course_id, course_name
    -> from courses
    -> where course_id in(
    -> select E.course_id
    -> from enrollments E
    -> group by E.course_id
    -> having count(E.course_id) = (select max(C) from (select course_id, count(course_id) as C from enrollments
    -> group by course_id) as count_enrolls));
+-----------+-------------+
| course_id | course_name |
+-----------+-------------+
|       203 | Marketing   |
|       205 | History     |
|       207 | Chemistry   |
+-----------+-------------+
3 rows in set (0.02 sec)
```

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



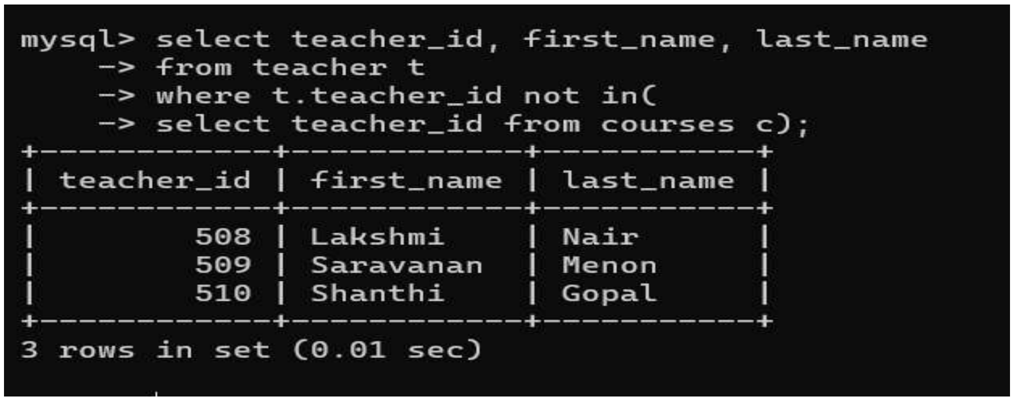
```
mysql> select teacher_id, first_name, last_name, SUM(amount) as Total_Payments
    -> from(
    -> select E.student_id, E.course_id, T.teacher_id, T.first_name, T.last_name, P.amount
    -> from enrollments E
    -> join courses C on E.course_id = C.course_id
    -> join teacher T on C.teacher_id = T.teacher_id
    -> join payments P on E.student_id = P.student_id
    -> ) as count_payments
    -> group by teacher_id, first_name, last_name;
+------------+------------+-----------+----------------+
| teacher_id | first_name | last_name | Total_Payments |
+------------+------------+-----------+----------------+
|        501 | Suresh     | Rao       |           1300 |
|        502 | Priya      | Nair      |           1550 |
|        504 | Meera      | Iyer      |           1100 |
|        505 | Deepak     | Pillai    |           1050 |
|        503 | Rajesh     | Menon     |           2050 |
|        507 | Krishna    | Sharma    |            400 |
+------------+------------+-----------+----------------+
6 rows in set (0.00 sec)
```

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



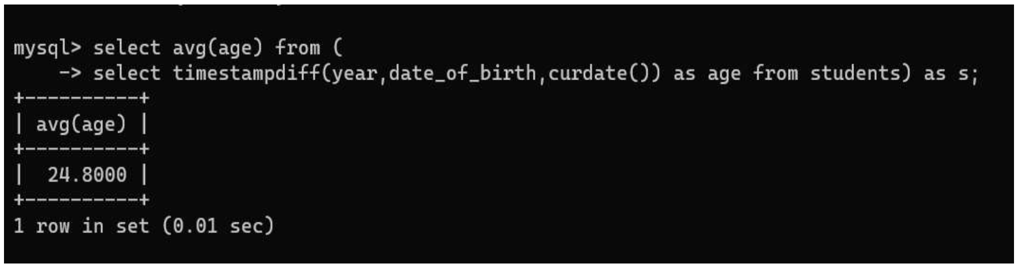
```
mysql> select student_id, first_name, last_name
    -> from students S
    -> where(
    -> select COUNT(DISTINCT course_id)
    -> from enrollments) = (
    -> select COUNT(DISTINCT course_id)
    -> from enrollments
    -> where student_id = S.student_id);
Empty set (0.00 sec)
```

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



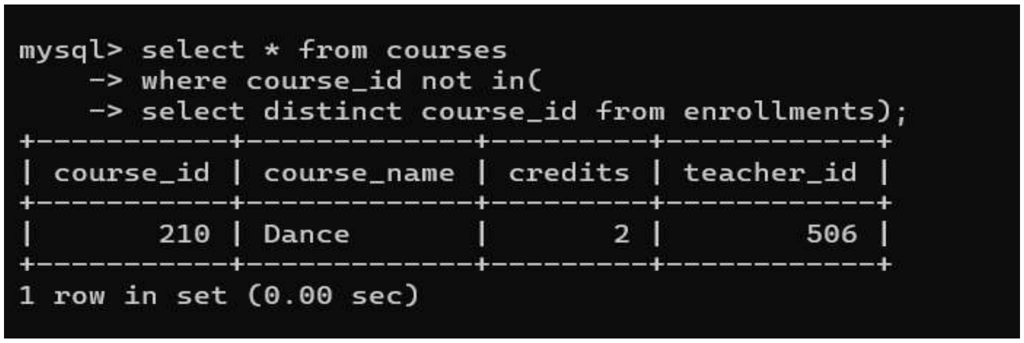
```
mysql> select teacher_id, first_name, last_name
    -> from teacher t
    -> where t.teacher_id not in(
    -> select teacher_id from courses c);
+------------+------------+-----------+
| teacher_id | first_name | last_name |
+------------+------------+-----------+
|        508 | Lakshmi    | Nair      |
|        509 | Saravanan  | Menon     |
|        510 | Shanthi    | Gopal     |
+------------+------------+-----------+
3 rows in set (0.01 sec)
```

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



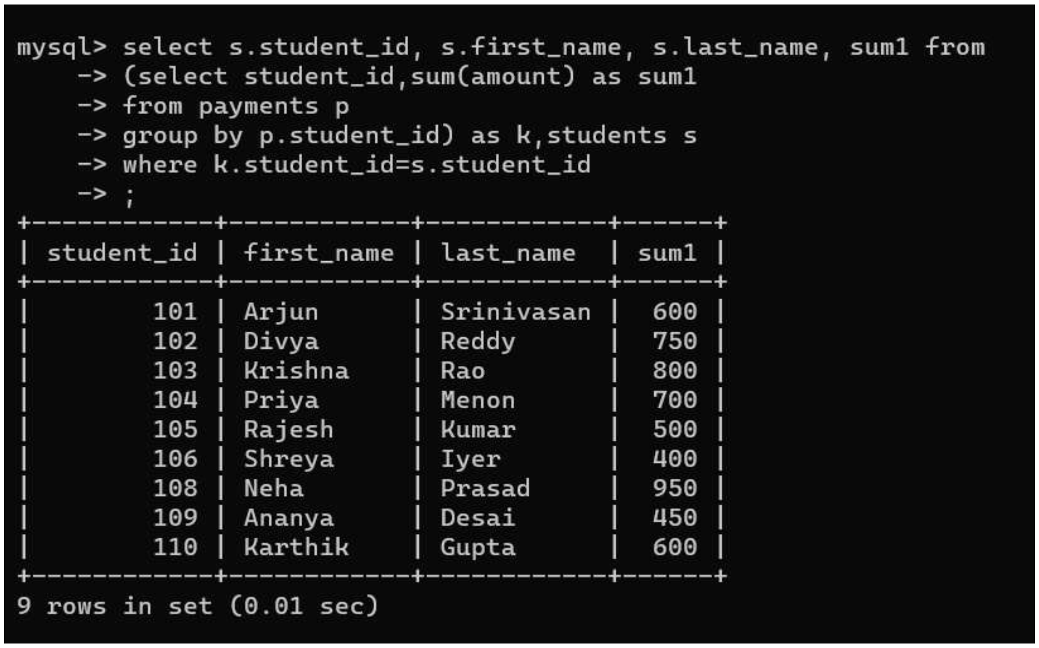
```
mysql> select avg(age) from (
    -> select timestampdiff(year,date_of_birth,curdate()) as age from students) as s;
+----------+
| avg(age) |
+----------+
|  24.8000 |
+----------+
1 row in set (0.01 sec)
```

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



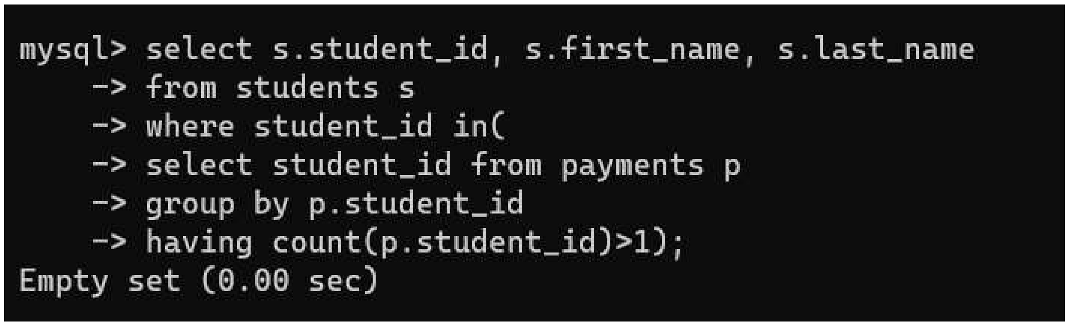
```
mysql> select * from courses
    -> where course_id not in(
    -> select distinct course_id from enrollments);
+-----------+-------------+---------+------------+
| course_id | course_name | credits | teacher_id |
+-----------+-------------+---------+------------+
|       210 | Dance       |       2 |        506 |
+-----------+-------------+---------+------------+
1 row in 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.***



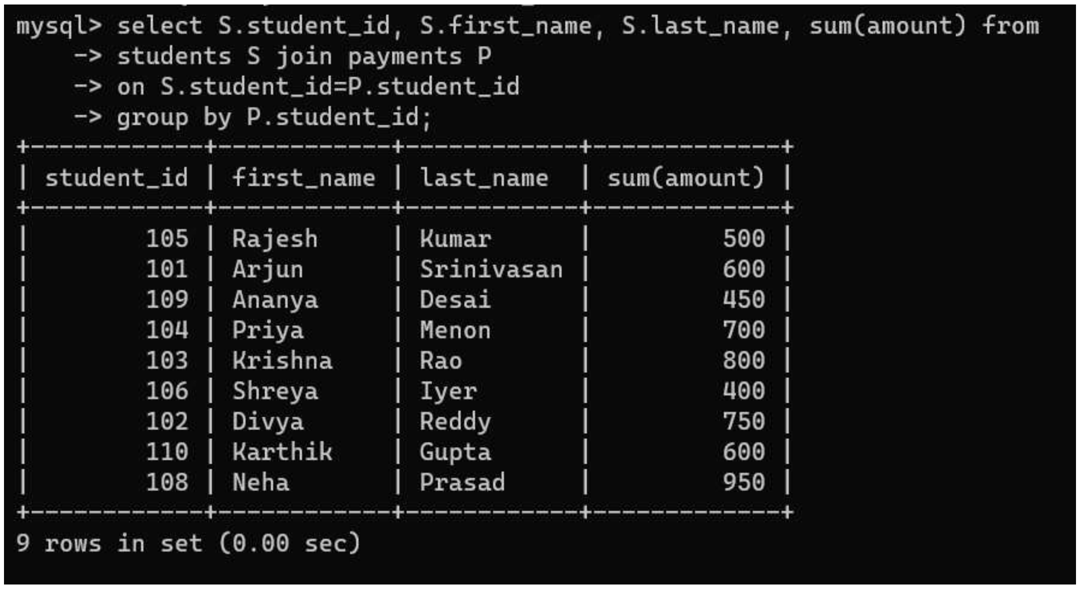
```
mysql> select s.student_id, s.first_name, s.last_name, sum1 from
    -> (select student_id,sum(amount) as sum1
    -> from payments p
    -> group by p.student_id) as k,students s
    -> where k.student_id=s.student_id
    -> ;
+------------+------------+------------+------+
| student_id | first_name | last_name  | sum1 |
+------------+------------+------------+------+
|        101 | Arjun      | Srinivasan |  600 |
|        102 | Divya      | Reddy      |  750 |
|        103 | Krishna    | Rao        |  800 |
|        104 | Priya      | Menon      |  700 |
|        105 | Rajesh     | Kumar      |  500 |
|        106 | Shreya     | Iyer       |  400 |
|        108 | Neha       | Prasad     |  950 |
|        109 | Ananya     | Desai      |  450 |
|        110 | Karthik    | Gupta      |  600 |
+------------+------------+------------+------+
9 rows in set (0.01 sec)
```

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.***



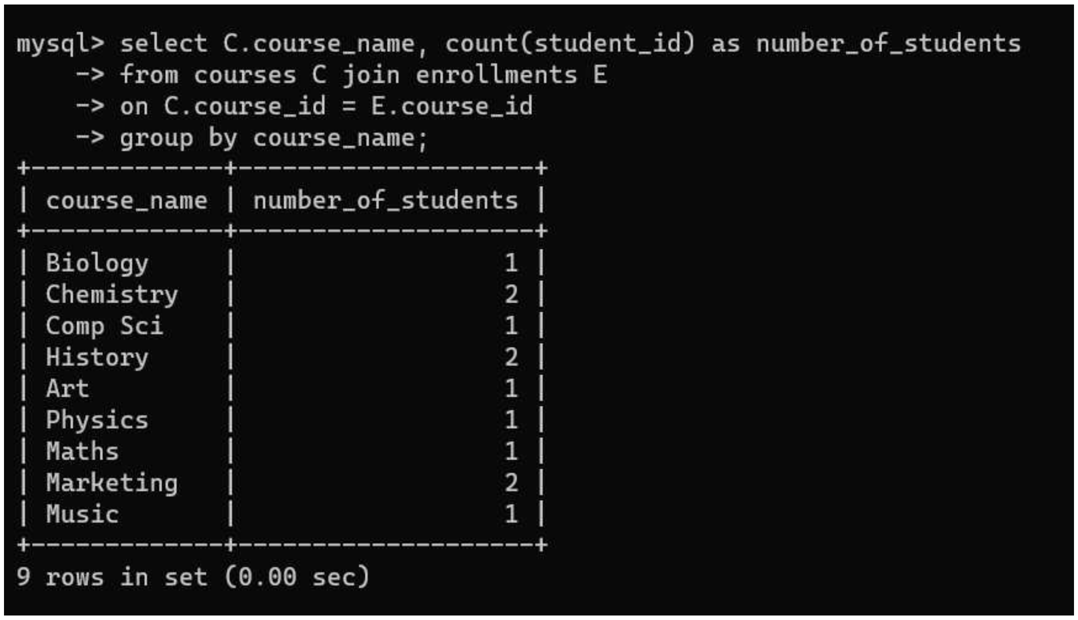
```
mysql> select s.student_id, s.first_name, s.last_name
    -> from students s
    -> where student_id in(
    -> select student_id from payments p
    -> group by p.student_id
    -> having count(p.student_id)>1);
Empty set (0.00 sec)
```

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.***



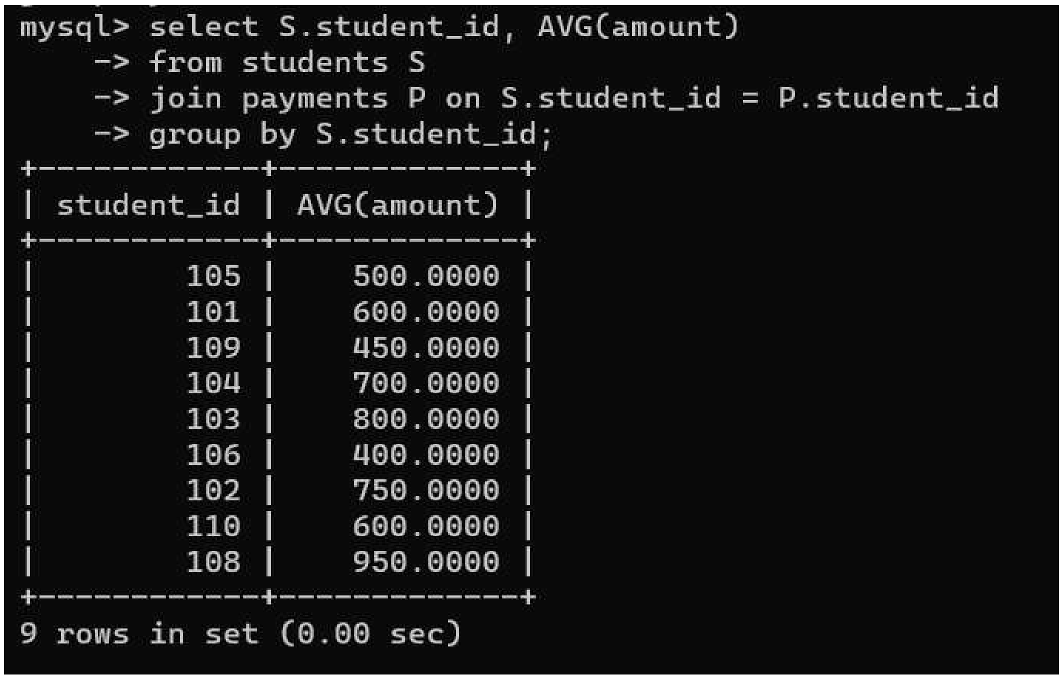
```
mysql> select S.student_id, S.first_name, S.last_name, sum(amount) from
    -> students S join payments P
    -> on S.student_id=P.student_id
    -> group by P.student_id;
+------------+------------+------------+-------------+
| student_id | first_name | last_name  | sum(amount) |
+------------+------------+------------+-------------+
|        105 | Rajesh     | Kumar      |         500 |
|        101 | Arjun      | Srinivasan |         600 |
|        109 | Ananya     | Desai      |         450 |
|        104 | Priya      | Menon      |         700 |
|        103 | Krishna    | Rao        |         800 |
|        106 | Shreya     | Iyer       |         400 |
|        102 | Divya      | Reddy      |         750 |
|        110 | Karthik    | Gupta      |         600 |
|        108 | Neha       | Prasad     |         950 |
+------------+------------+------------+-------------+
9 rows in set (0.00 sec)
```

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.*



```
mysql> select C.course_name, count(student_id) as number_of_students
    -> from courses C join enrollments E
    -> on C.course_id = E.course_id
    -> group by course_name;
+-------------+--------------------+
| course_name | number_of_students |
+-------------+--------------------+
| Biology     |                  1 |
| Chemistry   |                  2 |
| Comp Sci    |                  1 |
| History     |                  2 |
| Art         |                  1 |
| Physics     |                  1 |
| Maths       |                  1 |
| Marketing   |                  2 |
| Music       |                  1 |
+-------------+--------------------+
9 rows in set (0.00 sec)
```

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.*



```
mysql> select S.student_id, AVG(amount)
    -> from students S
    -> join payments P on S.student_id = P.student_id
    -> group by S.student_id;
+------------+-------------+
| student_id | AVG(amount) |
+------------+-------------+
|        105 |    500.0000 |
|        101 |    600.0000 |
|        109 |    450.0000 |
|        104 |    700.0000 |
|        103 |    800.0000 |
|        106 |    400.0000 |
|        102 |    750.0000 |
|        110 |    600.0000 |
|        108 |    950.0000 |
+------------+-------------+
9 rows in set (0.00 sec)
```

*****