ASSIGNMENT TITLE:

SISDB : MySQL ASSIGNMENT 2

SUBMITED BY:

BALAKUMARAN P

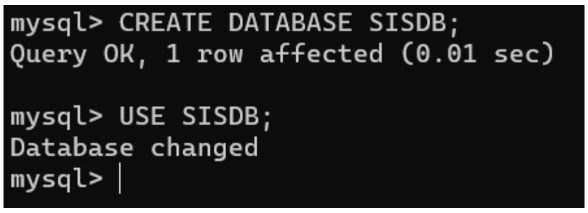
DATE OF SUBMISSION:

22/01/2024

SISDB : MySQL ASSIGNMENT 2

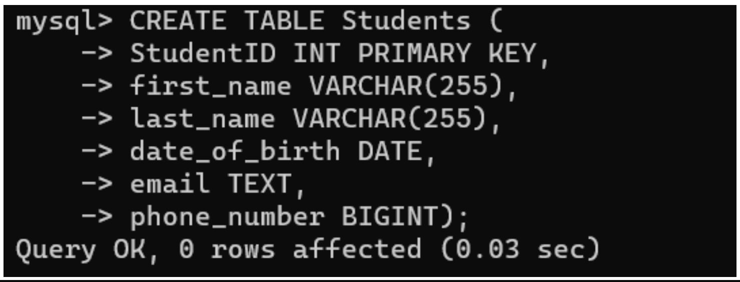
Task 1. Database Design:

1. Create the database named "SISDB"

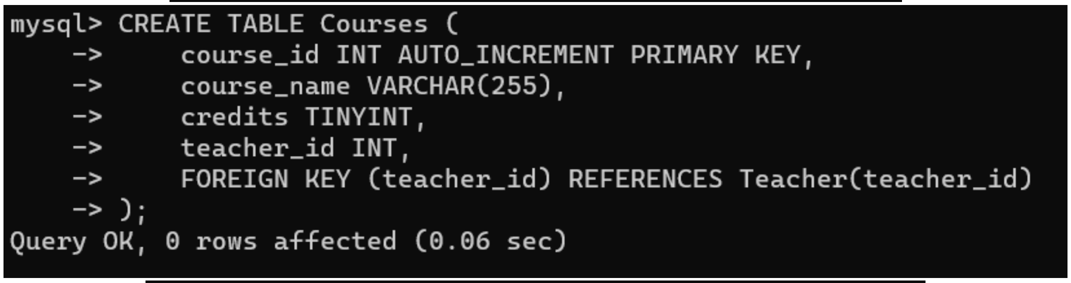


```
mysql> CREATE DATABASE SISDB;
Query OK, 1 row affected (0.01 sec)

mysql> USE SISDB;
Database changed
mysql> |
```

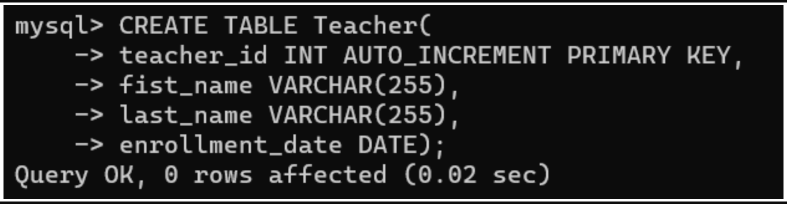
2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships. a. Students b. Courses c. Enrollments d. Teacher e. Payments



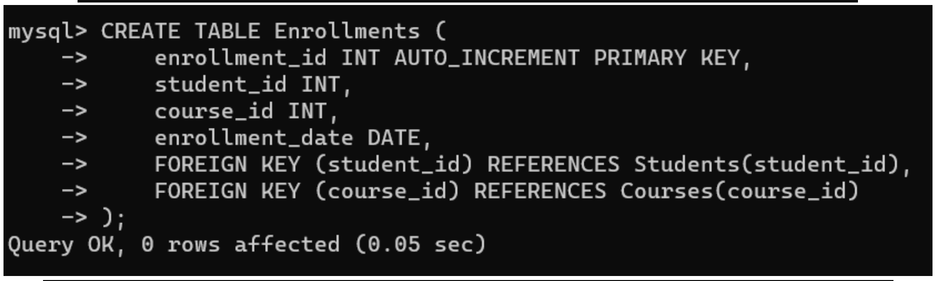
```
mysql> CREATE TABLE Students (
    -> StudentID INT PRIMARY KEY,
    -> first_name VARCHAR(255),
    -> last_name VARCHAR(255),
    -> date_of_birth DATE,
    -> email TEXT,
    -> phone_number BIGINT);
Query OK, 0 rows affected (0.03 sec)
```



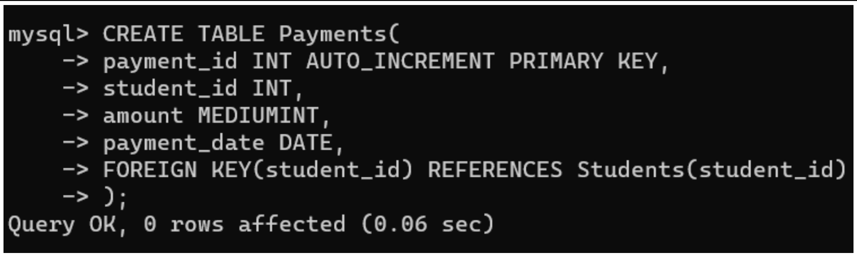
```
mysql> CREATE TABLE Courses (
    ->     course_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     course_name VARCHAR(255),
    ->     credits TINYINT,
    ->     teacher_id INT,
    ->     FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
    -> );
Query OK, 0 rows affected (0.06 sec)
```



```
mysql> CREATE TABLE Teacher(
    -> teacher_id INT AUTO_INCREMENT PRIMARY KEY,
    -> fist_name VARCHAR(255),
    -> last_name VARCHAR(255),
    -> enrollment_date DATE);
Query OK, 0 rows affected (0.02 sec)
```

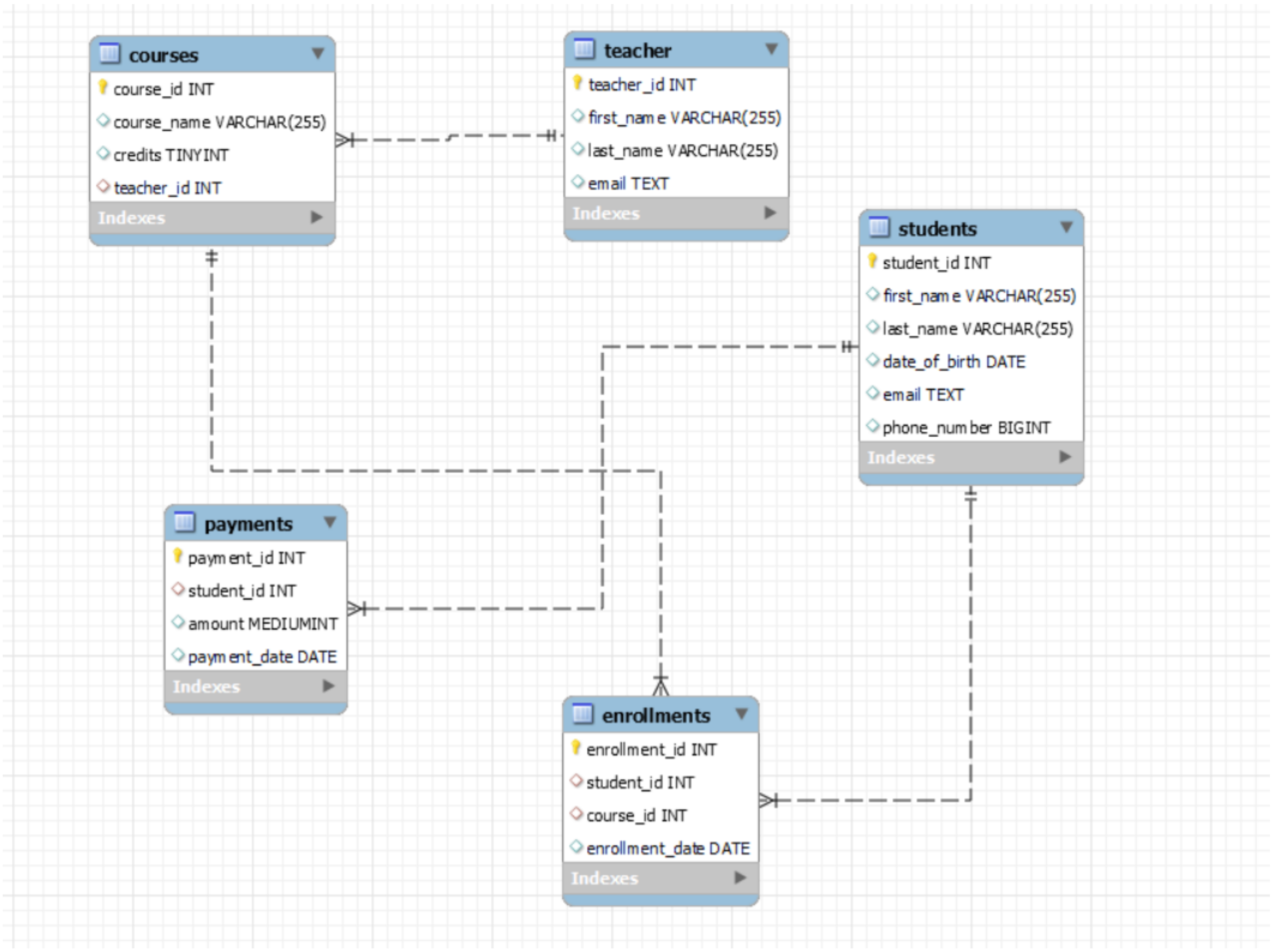


```
mysql> CREATE TABLE Enrollments (
    ->     enrollment_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     student_id INT,
    ->     course_id INT,
    ->     enrollment_date DATE,
    ->     FOREIGN KEY (student_id) REFERENCES Students(student_id),
    ->     FOREIGN KEY (course_id) REFERENCES Courses(course_id)
    -> );
Query OK, 0 rows affected (0.05 sec)
```



```
mysql> CREATE TABLE Payments(
    -> payment_id INT AUTO_INCREMENT PRIMARY KEY,
    -> student_id INT,
    -> amount MEDIUMINT,
    -> payment_date DATE,
    -> FOREIGN KEY(student_id) REFERENCES Students(student_id)
    -> );
Query OK, 0 rows affected (0.06 sec)
```

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



4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.
   a. Students Table:

   student_id: Primary key identifying each student uniquely.

   b. Courses Table:

   course_id: Primary key identifying each course uniquely.
   teacher_id: Foreign key referencing the teacher who is responsible for the course.

   c. Enrollments Table:

   enrollment_id: Primary key identifying each enrollment uniquely.
   student_id: Foreign key referencing the student enrolled.
   course_id: Foreign key referencing the course in which the student is enrolled.

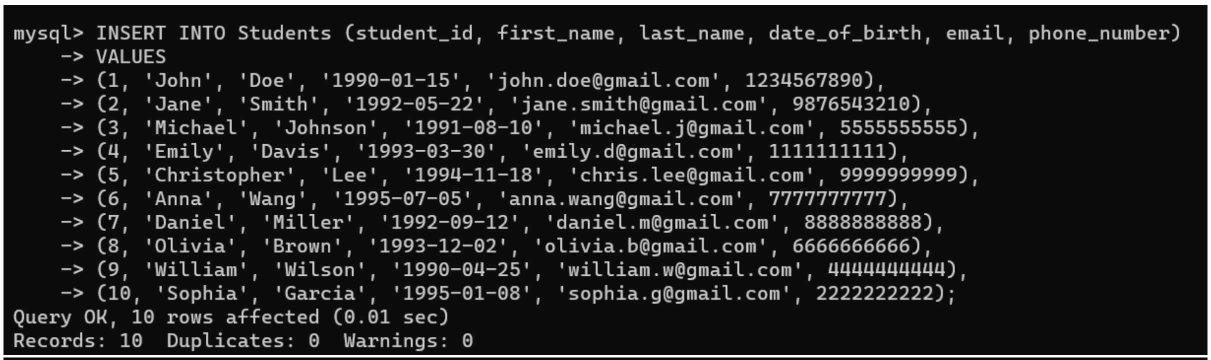
   d. Teacher Table:

   teacher_id: Primary key identifying each teacher uniquely.

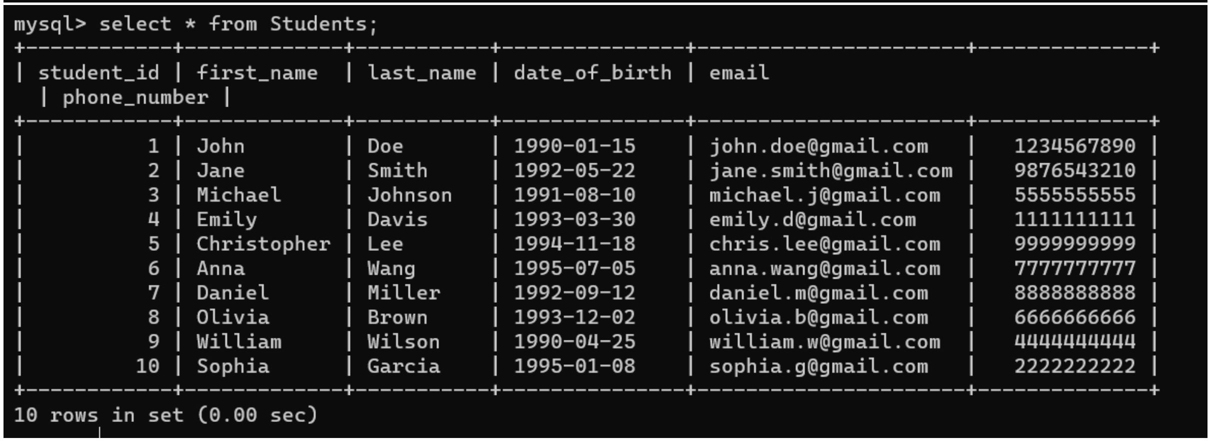
   e. Payments Table:

   payment_id: Primary key identifying each payment uniquely.
   student_id: Foreign key referencing the student making the payment.

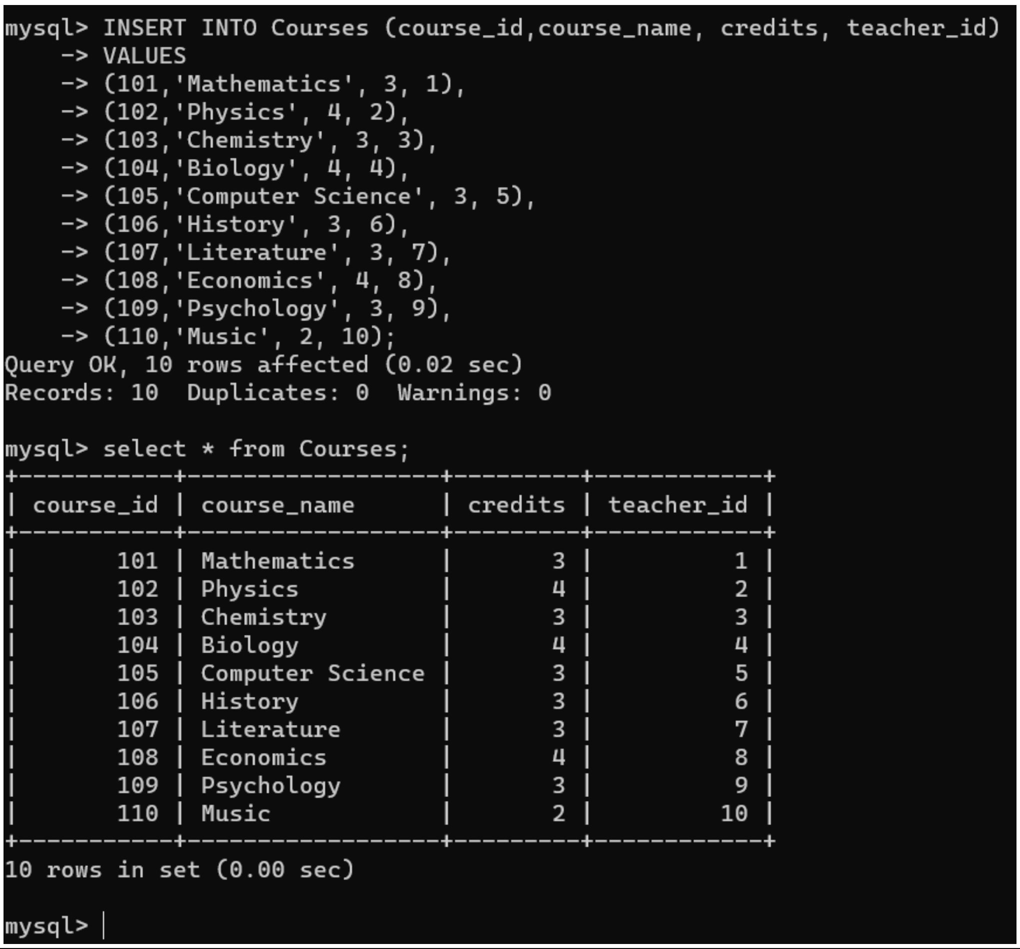
5. Insert at least 10 sample records into each of the following tables. i. Students ii. Courses iii. Enrollments iv. Teacher v. Payments



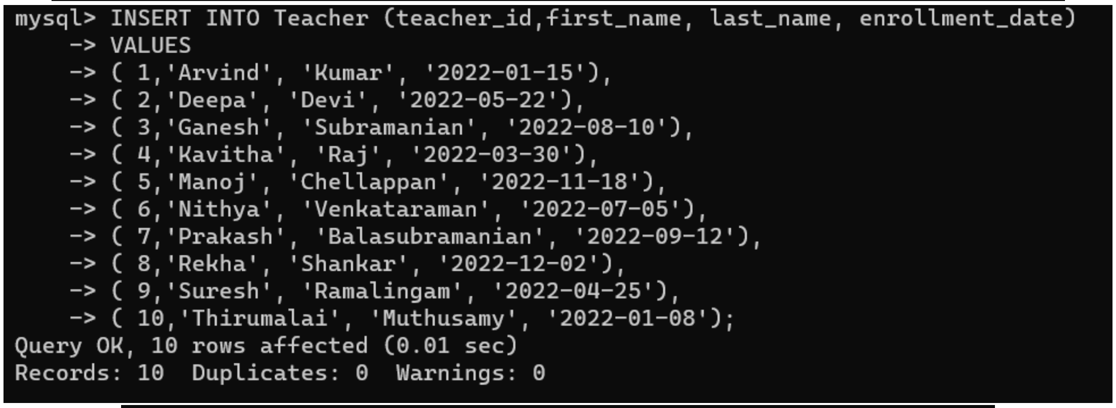
```
mysql> INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number)
    -> VALUES
    -> (1, 'John', 'Doe', '1990-01-15', 'john.doe@gmail.com', 1234567890),
    -> (2, 'Jane', 'Smith', '1992-05-22', 'jane.smith@gmail.com', 9876543210),
    -> (3, 'Michael', 'Johnson', '1991-08-10', 'michael.j@gmail.com', 5555555555),
    -> (4, 'Emily', 'Davis', '1993-03-30', 'emily.d@gmail.com', 1111111111),
    -> (5, 'Christopher', 'Lee', '1994-11-18', 'chris.lee@gmail.com', 9999999999),
    -> (6, 'Anna', 'Wang', '1995-07-05', 'anna.wang@gmail.com', 7777777777),
    -> (7, 'Daniel', 'Miller', '1992-09-12', 'daniel.m@gmail.com', 8888888888),
    -> (8, 'Olivia', 'Brown', '1993-12-02', 'olivia.b@gmail.com', 6666666666),
    -> (9, 'William', 'Wilson', '1990-04-25', 'william.w@gmail.com', 4444444444),
    -> (10, 'Sophia', 'Garcia', '1995-01-08', 'sophia.g@gmail.com', 2222222222);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```



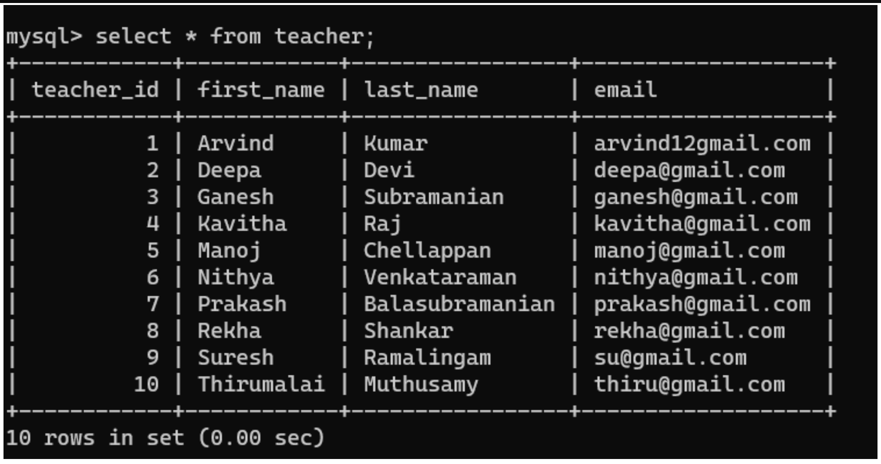
```
mysql> select * from Students;
+------------+-------------+-----------+---------------+----------------------+--------------+
| student_id | first_name  | last_name | date_of_birth | email
  | phone_number |
+------------+-------------+-----------+---------------+----------------------+--------------+
|          1 | John        | Doe       | 1990-01-15    | john.doe@gmail.com   |   1234567890 |
|          2 | Jane        | Smith     | 1992-05-22    | jane.smith@gmail.com |   9876543210 |
|          3 | Michael     | Johnson   | 1991-08-10    | michael.j@gmail.com  |   5555555555 |
|          4 | Emily       | Davis     | 1993-03-30    | emily.d@gmail.com    |   1111111111 |
|          5 | Christopher | Lee       | 1994-11-18    | chris.lee@gmail.com  |   9999999999 |
|          6 | Anna        | Wang      | 1995-07-05    | anna.wang@gmail.com  |   7777777777 |
|          7 | Daniel      | Miller    | 1992-09-12    | daniel.m@gmail.com   |   8888888888 |
|          8 | Olivia      | Brown     | 1993-12-02    | olivia.b@gmail.com   |   6666666666 |
|          9 | William     | Wilson    | 1990-04-25    | william.w@gmail.com  |   4444444444 |
|         10 | Sophia      | Garcia    | 1995-01-08    | sophia.g@gmail.com   |   2222222222 |
+------------+-------------+-----------+---------------+----------------------+--------------+
10 rows in set (0.00 sec)
```



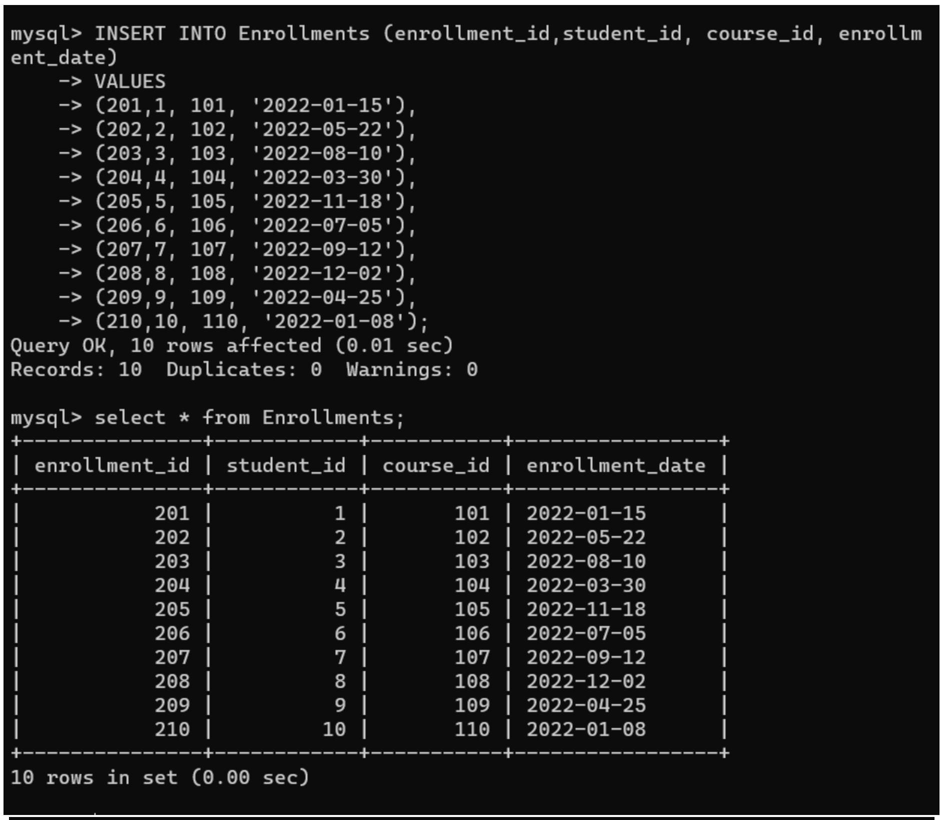
```
mysql> INSERT INTO Courses (course_id,course_name, credits, teacher_id)
    -> VALUES
    -> (101,'Mathematics', 3, 1),
    -> (102,'Physics', 4, 2),
    -> (103,'Chemistry', 3, 3),
    -> (104,'Biology', 4, 4),
    -> (105,'Computer Science', 3, 5),
    -> (106,'History', 3, 6),
    -> (107,'Literature', 3, 7),
    -> (108,'Economics', 4, 8),
    -> (109,'Psychology', 3, 9),
    -> (110,'Music', 2, 10);
Query OK, 10 rows affected (0.02 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Courses;
+-----------+------------------+---------+------------+
| course_id | course_name      | credits | teacher_id |
+-----------+------------------+---------+------------+
|       101 | Mathematics      |       3 |          1 |
|       102 | Physics          |       4 |          2 |
|       103 | Chemistry        |       3 |          3 |
|       104 | Biology          |       4 |          4 |
|       105 | Computer Science |       3 |          5 |
|       106 | History          |       3 |          6 |
|       107 | Literature       |       3 |          7 |
|       108 | Economics        |       4 |          8 |
|       109 | Psychology       |       3 |          9 |
|       110 | Music            |       2 |         10 |
+-----------+------------------+---------+------------+
10 rows in set (0.00 sec)

mysql> |
```



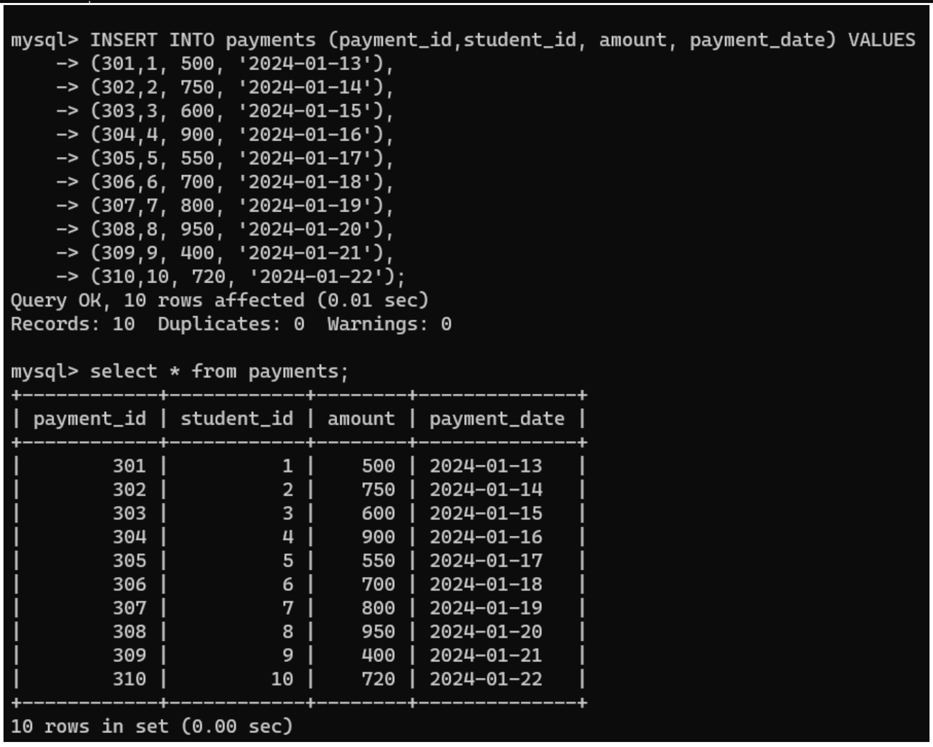
```
mysql> INSERT INTO Teacher (teacher_id,first_name, last_name, enrollment_date)
    -> VALUES
    -> ( 1,'Arvind', 'Kumar', '2022-01-15'),
    -> ( 2,'Deepa', 'Devi', '2022-05-22'),
    -> ( 3,'Ganesh', 'Subramanian', '2022-08-10'),
    -> ( 4,'Kavitha', 'Raj', '2022-03-30'),
    -> ( 5,'Manoj', 'Chellappan', '2022-11-18'),
    -> ( 6,'Nithya', 'Venkataraman', '2022-07-05'),
    -> ( 7,'Prakash', 'Balasubramanian', '2022-09-12'),
    -> ( 8,'Rekha', 'Shankar', '2022-12-02'),
    -> ( 9,'Suresh', 'Ramalingam', '2022-04-25'),
    -> ( 10,'Thirumalai', 'Muthusamy', '2022-01-08');
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             |
+------------+------------+-----------------+-------------------+
|          1 | Arvind     | Kumar           | arvind12gmail.com |
|          2 | Deepa      | Devi            | deepa@gmail.com   |
|          3 | Ganesh     | Subramanian     | ganesh@gmail.com  |
|          4 | Kavitha    | Raj             | kavitha@gmail.com |
|          5 | Manoj      | Chellappan      | manoj@gmail.com   |
|          6 | Nithya     | Venkataraman    | nithya@gmail.com  |
|          7 | Prakash    | Balasubramanian | prakash@gmail.com |
|          8 | Rekha      | Shankar         | rekha@gmail.com   |
|          9 | Suresh     | Ramalingam      | su@gmail.com      |
|         10 | Thirumalai | Muthusamy       | thiru@gmail.com   |
+------------+------------+-----------------+-------------------+
10 rows in set (0.00 sec)
```



```
mysql> INSERT INTO Enrollments (enrollment_id,student_id, course_id, enrollm
ent_date)
    -> VALUES
    -> (201,1, 101, '2022-01-15'),
    -> (202,2, 102, '2022-05-22'),
    -> (203,3, 103, '2022-08-10'),
    -> (204,4, 104, '2022-03-30'),
    -> (205,5, 105, '2022-11-18'),
    -> (206,6, 106, '2022-07-05'),
    -> (207,7, 107, '2022-09-12'),
    -> (208,8, 108, '2022-12-02'),
    -> (209,9, 109, '2022-04-25'),
    -> (210,10, 110, '2022-01-08');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           201 |          1 |       101 | 2022-01-15      |
|           202 |          2 |       102 | 2022-05-22      |
|           203 |          3 |       103 | 2022-08-10      |
|           204 |          4 |       104 | 2022-03-30      |
|           205 |          5 |       105 | 2022-11-18      |
|           206 |          6 |       106 | 2022-07-05      |
|           207 |          7 |       107 | 2022-09-12      |
|           208 |          8 |       108 | 2022-12-02      |
|           209 |          9 |       109 | 2022-04-25      |
|           210 |         10 |       110 | 2022-01-08      |
+---------------+------------+-----------+-----------------+
10 rows in set (0.00 sec)
```



```
mysql> INSERT INTO payments (payment_id,student_id, amount, payment_date) VALUES
    -> (301,1, 500, '2024-01-13'),
    -> (302,2, 750, '2024-01-14'),
    -> (303,3, 600, '2024-01-15'),
    -> (304,4, 900, '2024-01-16'),
    -> (305,5, 550, '2024-01-17'),
    -> (306,6, 700, '2024-01-18'),
    -> (307,7, 800, '2024-01-19'),
    -> (308,8, 950, '2024-01-20'),
    -> (309,9, 400, '2024-01-21'),
    -> (310,10, 720, '2024-01-22');
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 |
+------------+------------+--------+--------------+
|        301 |          1 |    500 | 2024-01-13   |
|        302 |          2 |    750 | 2024-01-14   |
|        303 |          3 |    600 | 2024-01-15   |
|        304 |          4 |    900 | 2024-01-16   |
|        305 |          5 |    550 | 2024-01-17   |
|        306 |          6 |    700 | 2024-01-18   |
|        307 |          7 |    800 | 2024-01-19   |
|        308 |          8 |    950 | 2024-01-20   |
|        309 |          9 |    400 | 2024-01-21   |
|        310 |         10 |    720 | 2024-01-22   |
+------------+------------+--------+--------------+
10 rows in set (0.00 sec)
```

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

1. Write an SQL query to insert a new student into the "Students" table with the following details:

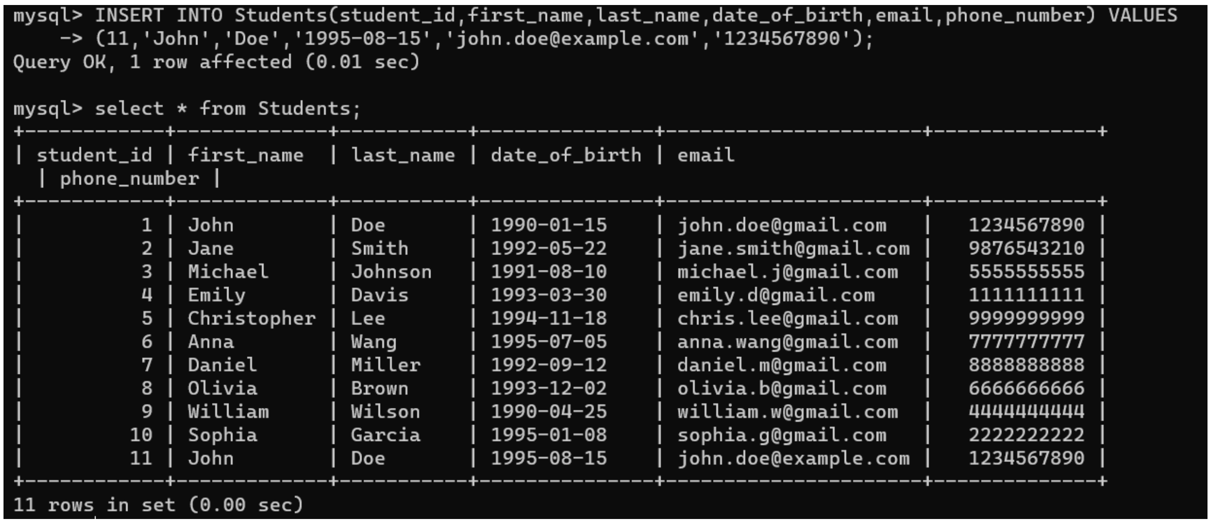
a. First Name: John

b. Last Name: Doe

c. Date of Birth: 1995-08-15

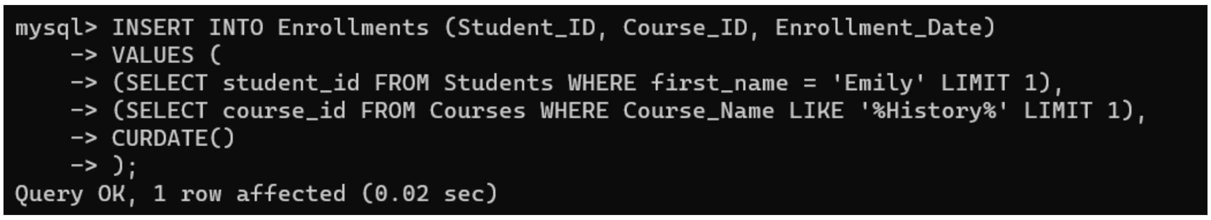
d. Email: john.doe@example.com

e. Phone Number: 1234567890

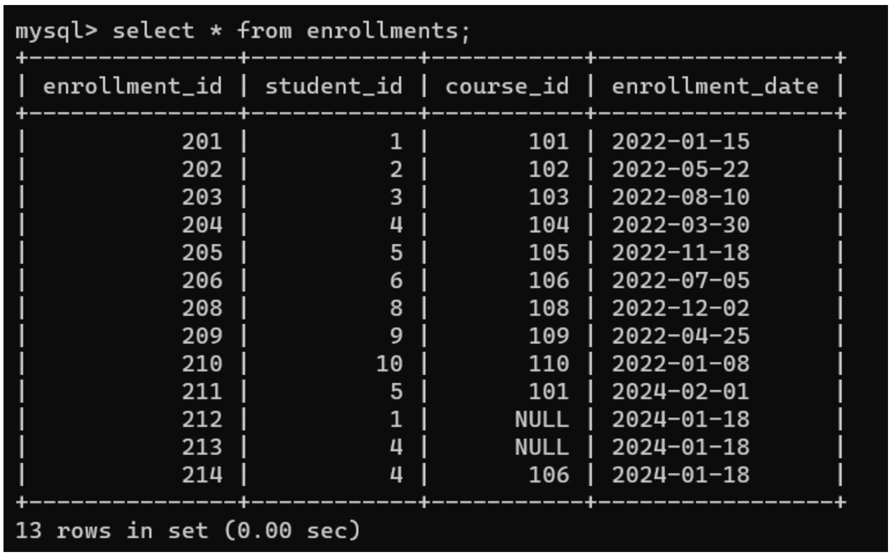


```
mysql> INSERT INTO Students(student_id,first_name,last_name,date_of_birth,email,phone_number) VALUES
    -> (11,'John','Doe','1995-08-15','john.doe@example.com','1234567890');
Query OK, 1 row affected (0.01 sec)

mysql> select * from Students;
+------------+-------------+-----------+---------------+----------------------+--------------+
| student_id | first_name  | last_name | date_of_birth | email
  | phone_number |
+------------+-------------+-----------+---------------+----------------------+--------------+
|          1 | John        | Doe       | 1990-01-15    | john.doe@gmail.com   |   1234567890 |
|          2 | Jane        | Smith     | 1992-05-22    | jane.smith@gmail.com |   9876543210 |
|          3 | Michael     | Johnson   | 1991-08-10    | michael.j@gmail.com  |   5555555555 |
|          4 | Emily       | Davis     | 1993-03-30    | emily.d@gmail.com    |   1111111111 |
|          5 | Christopher | Lee       | 1994-11-18    | chris.lee@gmail.com  |   9999999999 |
|          6 | Anna        | Wang      | 1995-07-05    | anna.wang@gmail.com  |   7777777777 |
|          7 | Daniel      | Miller    | 1992-09-12    | daniel.m@gmail.com   |   8888888888 |
|          8 | Olivia      | Brown     | 1993-12-02    | olivia.b@gmail.com   |   6666666666 |
|          9 | William     | Wilson    | 1990-04-25    | william.w@gmail.com  |   4444444444 |
|         10 | Sophia      | Garcia    | 1995-01-08    | sophia.g@gmail.com   |   2222222222 |
|         11 | John        | Doe       | 1995-08-15    | john.doe@example.com |   1234567890 |
+------------+-------------+-----------+---------------+----------------------+--------------+
11 rows in set (0.00 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.

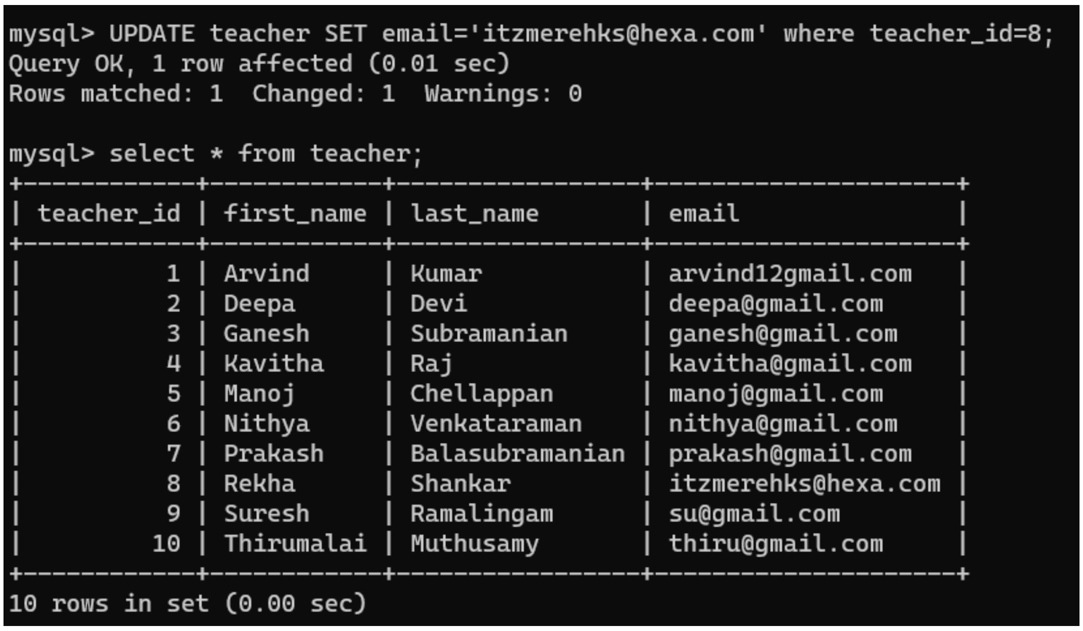


```
mysql> INSERT INTO Enrollments (Student_ID, Course_ID, Enrollment_Date)
    -> VALUES (
    -> (SELECT student_id FROM Students WHERE first_name = 'Emily' LIMIT 1),
    -> (SELECT course_id FROM Courses WHERE Course_Name LIKE '%History%' LIMIT 1),
    -> CURDATE()
    -> );
Query OK, 1 row affected (0.02 sec)
```



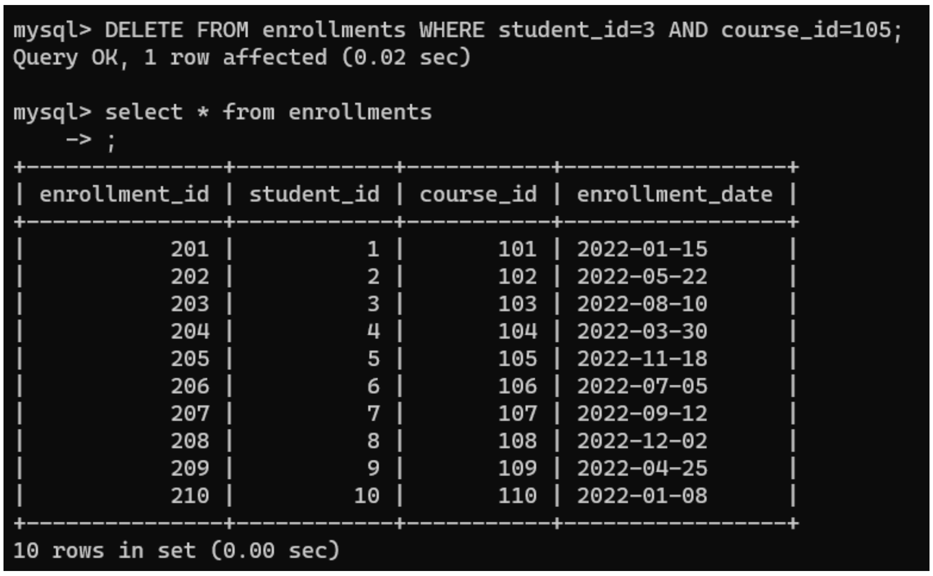
```
mysql> select * from enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           201 |          1 |       101 | 2022-01-15      |
|           202 |          2 |       102 | 2022-05-22      |
|           203 |          3 |       103 | 2022-08-10      |
|           204 |          4 |       104 | 2022-03-30      |
|           205 |          5 |       105 | 2022-11-18      |
|           206 |          6 |       106 | 2022-07-05      |
|           208 |          8 |       108 | 2022-12-02      |
|           209 |          9 |       109 | 2022-04-25      |
|           210 |         10 |       110 | 2022-01-08      |
|           211 |          5 |       101 | 2024-02-01      |
|           212 |          1 |      NULL | 2024-01-18      |
|           213 |          4 |      NULL | 2024-01-18      |
|           214 |          4 |       106 | 2024-01-18      |
+---------------+------------+-----------+-----------------+
13 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.



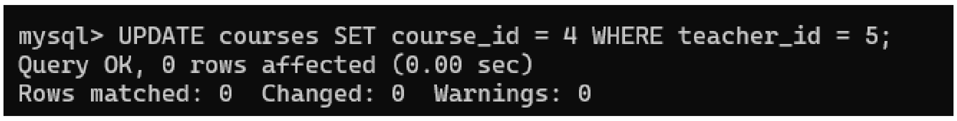
```
mysql> UPDATE teacher SET email='itzmerehks@hexa.com' where teacher_id=8;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from teacher;
+------------+------------+-----------------+---------------------+
| teacher_id | first_name | last_name       | email               |
+------------+------------+-----------------+---------------------+
|          1 | Arvind     | Kumar           | arvind12gmail.com   |
|          2 | Deepa      | Devi            | deepa@gmail.com     |
|          3 | Ganesh     | Subramanian     | ganesh@gmail.com    |
|          4 | Kavitha    | Raj             | kavitha@gmail.com   |
|          5 | Manoj      | Chellappan      | manoj@gmail.com     |
|          6 | Nithya     | Venkataraman    | nithya@gmail.com    |
|          7 | Prakash    | Balasubramanian | prakash@gmail.com   |
|          8 | Rekha      | Shankar         | itzmerehks@hexa.com |
|          9 | Suresh     | Ramalingam      | su@gmail.com        |
|         10 | Thirumalai | Muthusamy       | thiru@gmail.com     |
+------------+------------+-----------------+---------------------+
10 rows 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.



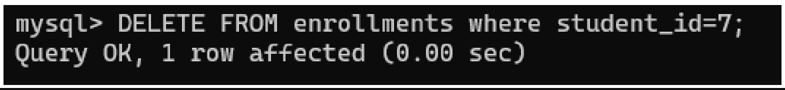
```
mysql> DELETE FROM enrollments WHERE student_id=3 AND course_id=105;
Query OK, 1 row affected (0.02 sec)

mysql> select * from enrollments
    -> ;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           201 |          1 |       101 | 2022-01-15      |
|           202 |          2 |       102 | 2022-05-22      |
|           203 |          3 |       103 | 2022-08-10      |
|           204 |          4 |       104 | 2022-03-30      |
|           205 |          5 |       105 | 2022-11-18      |
|           206 |          6 |       106 | 2022-07-05      |
|           207 |          7 |       107 | 2022-09-12      |
|           208 |          8 |       108 | 2022-12-02      |
|           209 |          9 |       109 | 2022-04-25      |
|           210 |         10 |       110 | 2022-01-08      |
+---------------+------------+-----------+-----------------+
10 rows in set (0.00 sec)
```

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

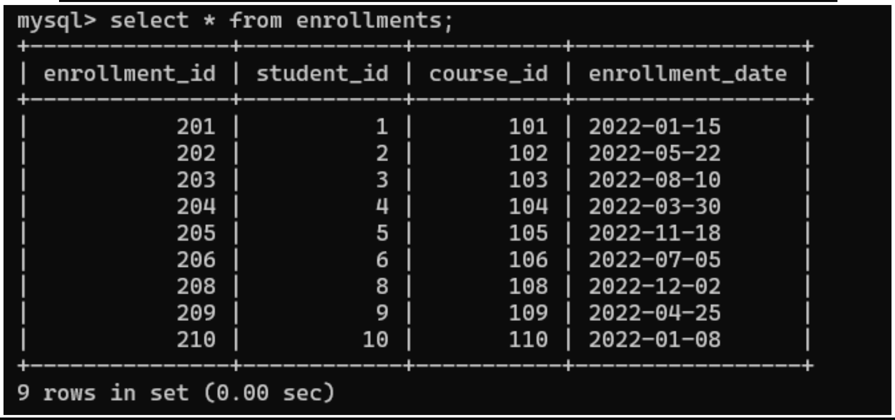


```
mysql> UPDATE courses SET course_id = 4 WHERE teacher_id = 5;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0
```

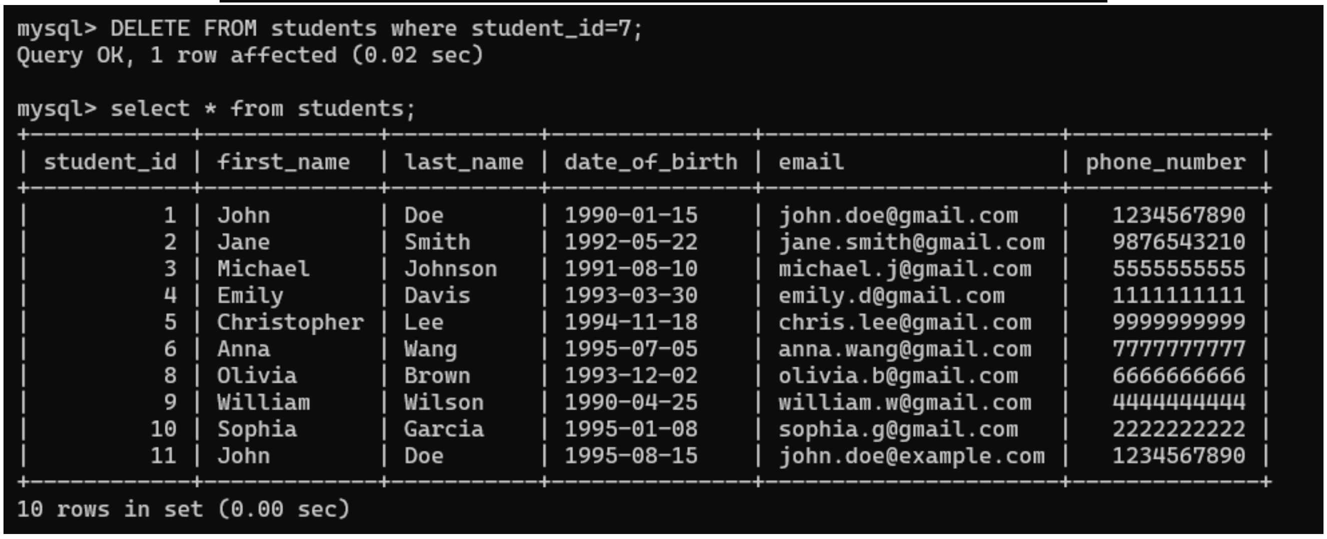
6. Delete a specific student from the "Students" table and remove all their enrollment records from the "Enrollments" table. Be sure to maintain referential integrity.



```
mysql> DELETE FROM enrollments where student_id=7;
Query OK, 1 row affected (0.00 sec)
```

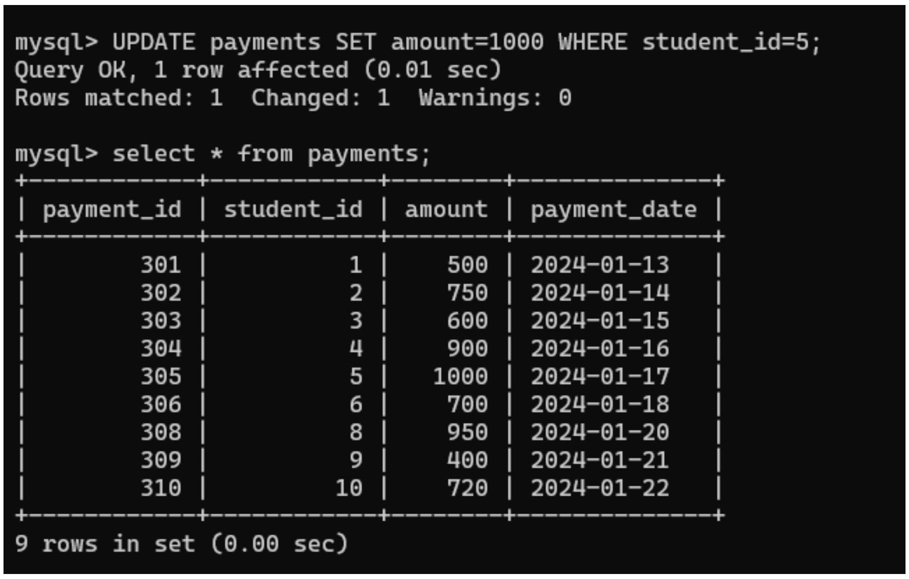


```
mysql> select * from enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           201 |          1 |       101 | 2022-01-15      |
|           202 |          2 |       102 | 2022-05-22      |
|           203 |          3 |       103 | 2022-08-10      |
|           204 |          4 |       104 | 2022-03-30      |
|           205 |          5 |       105 | 2022-11-18      |
|           206 |          6 |       106 | 2022-07-05      |
|           208 |          8 |       108 | 2022-12-02      |
|           209 |          9 |       109 | 2022-04-25      |
|           210 |         10 |       110 | 2022-01-08      |
+---------------+------------+-----------+-----------------+
9 rows in set (0.00 sec)
```



```
mysql> DELETE FROM students where student_id=7;
Query OK, 1 row affected (0.02 sec)

mysql> select * from students;
+------------+-------------+-----------+---------------+----------------------+--------------+
| student_id | first_name  | last_name | date_of_birth | email                | phone_number |
+------------+-------------+-----------+---------------+----------------------+--------------+
|          1 | John        | Doe       | 1990-01-15    | john.doe@gmail.com   |   1234567890 |
|          2 | Jane        | Smith     | 1992-05-22    | jane.smith@gmail.com |   9876543210 |
|          3 | Michael     | Johnson   | 1991-08-10    | michael.j@gmail.com  |   5555555555 |
|          4 | Emily       | Davis     | 1993-03-30    | emily.d@gmail.com    |   1111111111 |
|          5 | Christopher | Lee       | 1994-11-18    | chris.lee@gmail.com  |   9999999999 |
|          6 | Anna        | Wang      | 1995-07-05    | anna.wang@gmail.com  |   7777777777 |
|          8 | Olivia      | Brown     | 1993-12-02    | olivia.b@gmail.com   |   6666666666 |
|          9 | William     | Wilson    | 1990-04-25    | william.w@gmail.com  |   4444444444 |
|         10 | Sophia      | Garcia    | 1995-01-08    | sophia.g@gmail.com   |   2222222222 |
|         11 | John        | Doe       | 1995-08-15    | john.doe@example.com |   1234567890 |
+------------+-------------+-----------+---------------+----------------------+--------------+
10 rows in set (0.00 sec)
```

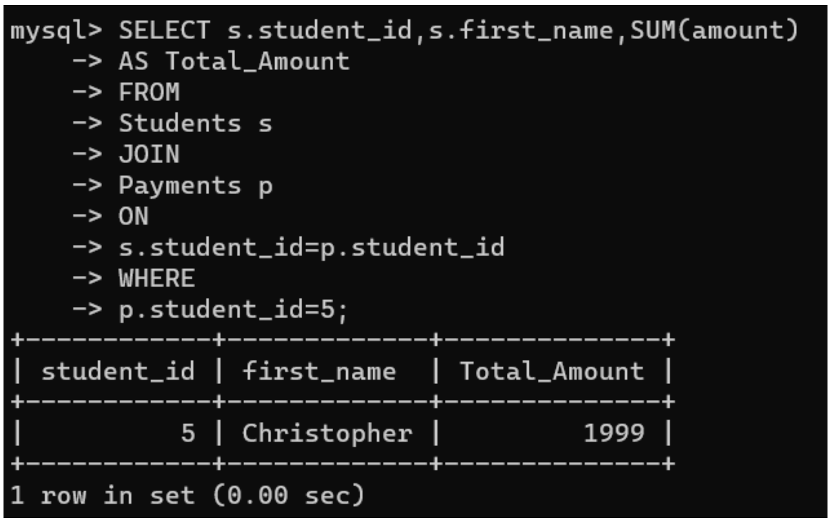
7. Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.



```
mysql> UPDATE payments SET amount=1000 WHERE student_id=5;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from payments;
+------------+------------+--------+--------------+
| payment_id | student_id | amount | payment_date |
+------------+------------+--------+--------------+
|        301 |          1 |    500 | 2024-01-13   |
|        302 |          2 |    750 | 2024-01-14   |
|        303 |          3 |    600 | 2024-01-15   |
|        304 |          4 |    900 | 2024-01-16   |
|        305 |          5 |   1000 | 2024-01-17   |
|        306 |          6 |    700 | 2024-01-18   |
|        308 |          8 |    950 | 2024-01-20   |
|        309 |          9 |    400 | 2024-01-21   |
|        310 |         10 |    720 | 2024-01-22   |
+------------+------------+--------+--------------+
9 rows in set (0.00 sec)
```

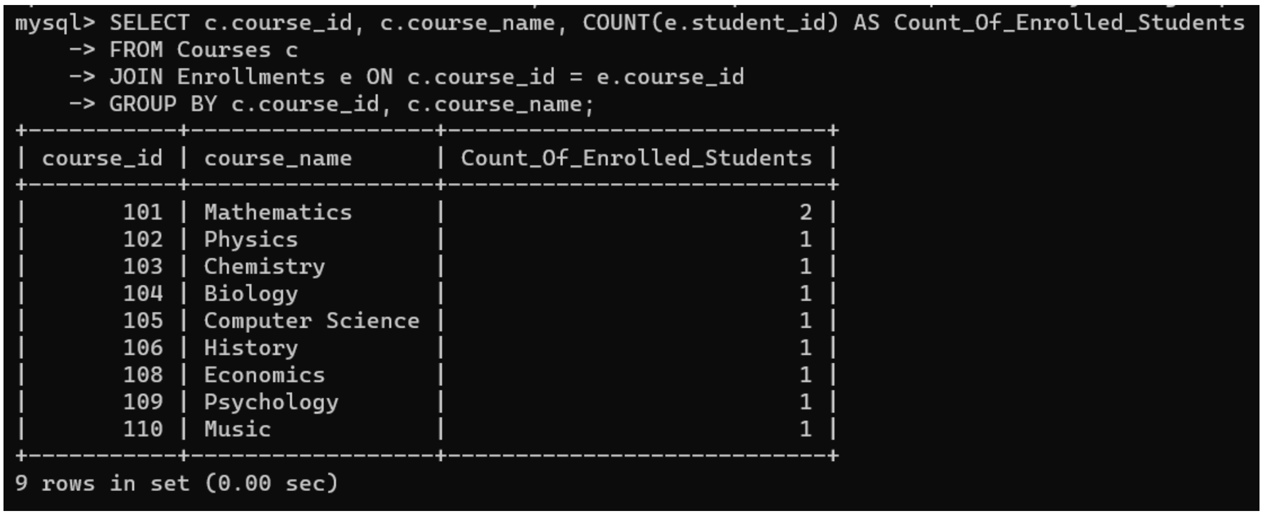
Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:

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.



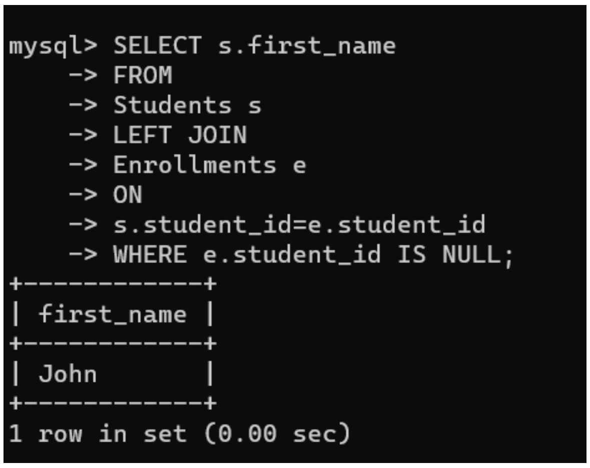
```
mysql> SELECT s.student_id,s.first_name,SUM(amount)
    -> AS Total_Amount
    -> FROM
    -> Students s
    -> JOIN
    -> Payments p
    -> ON
    -> s.student_id=p.student_id
    -> WHERE
    -> p.student_id=5;
+------------+-------------+--------------+
| student_id | first_name  | Total_Amount |
+------------+-------------+--------------+
|          5 | Christopher |         1999 |
+------------+-------------+--------------+
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.



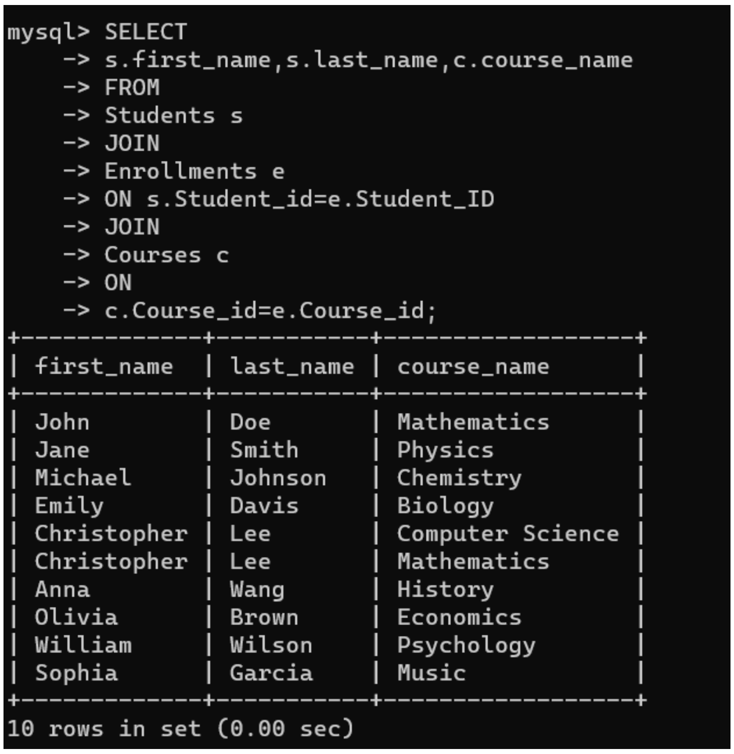
```
mysql> SELECT c.course_id, c.course_name, COUNT(e.student_id) AS Count_Of_Enrolled_Students
    -> FROM Courses c
    -> JOIN Enrollments e ON c.course_id = e.course_id
    -> GROUP BY c.course_id, c.course_name;
+-----------+------------------+----------------------------+
| course_id | course_name      | Count_Of_Enrolled_Students |
+-----------+------------------+----------------------------+
|       101 | Mathematics      |                          2 |
|       102 | Physics          |                          1 |
|       103 | Chemistry        |                          1 |
|       104 | Biology          |                          1 |
|       105 | Computer Science |                          1 |
|       106 | History          |                          1 |
|       108 | Economics        |                          1 |
|       109 | Psychology       |                          1 |
|       110 | Music            |                          1 |
+-----------+------------------+----------------------------+
9 rows in set (0.00 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.



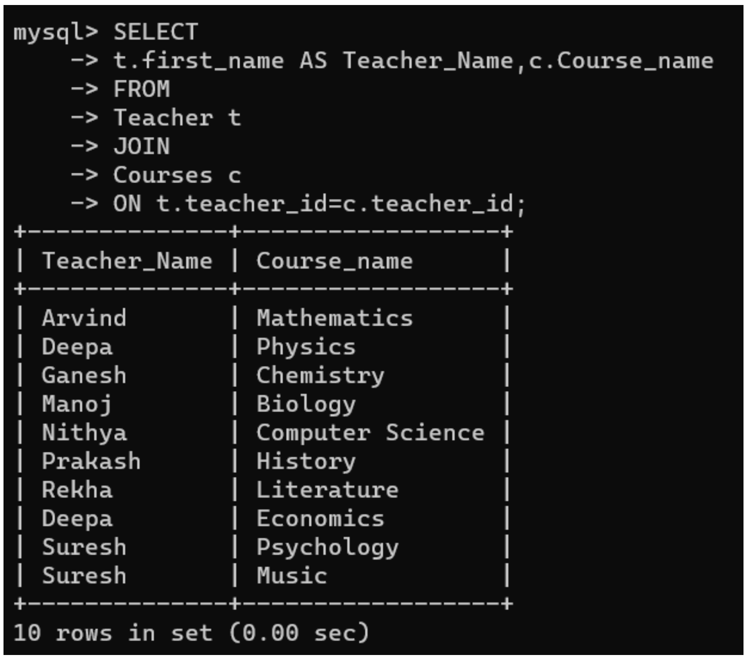
```
mysql> SELECT s.first_name
    -> FROM
    -> Students s
    -> LEFT JOIN
    -> Enrollments e
    -> ON
    -> s.student_id=e.student_id
    -> WHERE e.student_id IS NULL;
+------------+
| first_name |
+------------+
| John       |
+------------+
1 row in set (0.00 sec)
```

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.



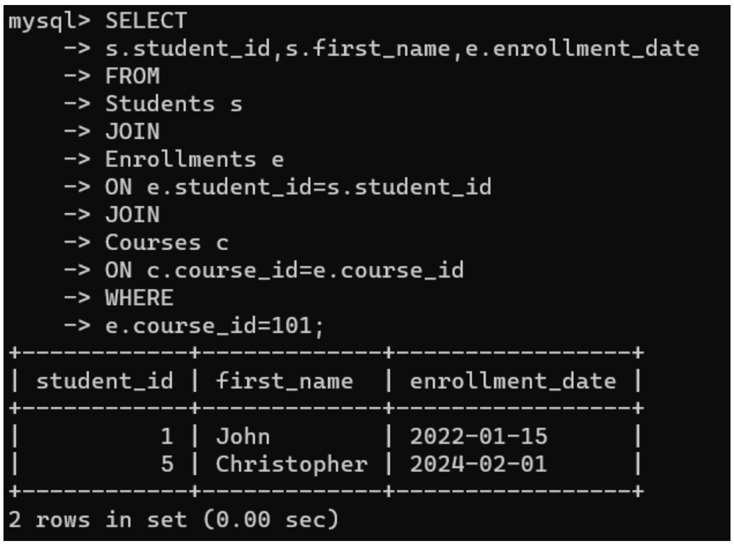
```
mysql> SELECT
    -> s.first_name,s.last_name,c.course_name
    -> FROM
    -> Students s
    -> JOIN
    -> Enrollments e
    -> ON s.Student_id=e.Student_ID
    -> JOIN
    -> Courses c
    -> ON
    -> c.Course_id=e.Course_id;
+-------------+-----------+------------------+
| first_name  | last_name | course_name      |
+-------------+-----------+------------------+
| John        | Doe       | Mathematics      |
| Jane        | Smith     | Physics          |
| Michael     | Johnson   | Chemistry        |
| Emily       | Davis     | Biology          |
| Christopher | Lee       | Computer Science |
| Christopher | Lee       | Mathematics      |
| Anna        | Wang      | History          |
| Olivia      | Brown     | Economics        |
| William     | Wilson    | Psychology       |
| Sophia      | Garcia    | Music            |
+-------------+-----------+------------------+
10 rows in set (0.00 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.



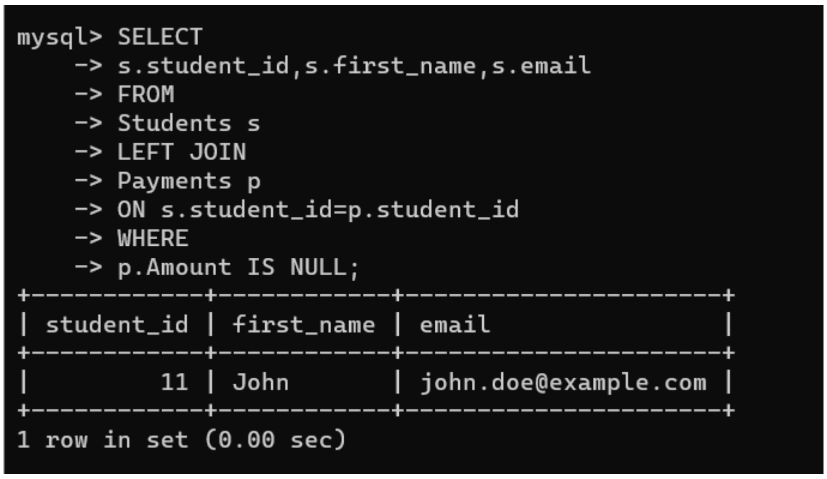
```
mysql> SELECT
    -> t.first_name AS Teacher_Name,c.Course_name
    -> FROM
    -> Teacher t
    -> JOIN
    -> Courses c
    -> ON t.teacher_id=c.teacher_id;
+--------------+------------------+
| Teacher_Name | Course_name      |
+--------------+------------------+
| Arvind       | Mathematics      |
| Deepa        | Physics          |
| Ganesh       | Chemistry        |
| Manoj        | Biology          |
| Nithya       | Computer Science |
| Prakash      | History          |
| Rekha        | Literature       |
| Deepa        | Economics        |
| Suresh       | Psychology       |
| Suresh       | Music            |
+--------------+------------------+
10 rows in set (0.00 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.



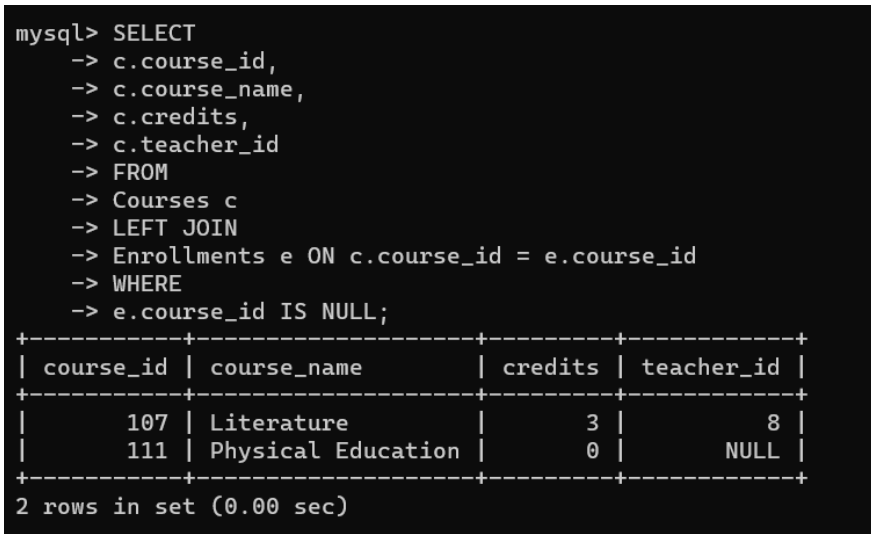
```
mysql> SELECT
    -> s.student_id,s.first_name,e.enrollment_date
    -> FROM
    -> Students s
    -> JOIN
    -> Enrollments e
    -> ON e.student_id=s.student_id
    -> JOIN
    -> Courses c
    -> ON c.course_id=e.course_id
    -> WHERE
    -> e.course_id=101;
+------------+-------------+-----------------+
| student_id | first_name  | enrollment_date |
+------------+-------------+-----------------+
|          1 | John        | 2022-01-15      |
|          5 | Christopher | 2024-02-01      |
+------------+-------------+-----------------+
2 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.



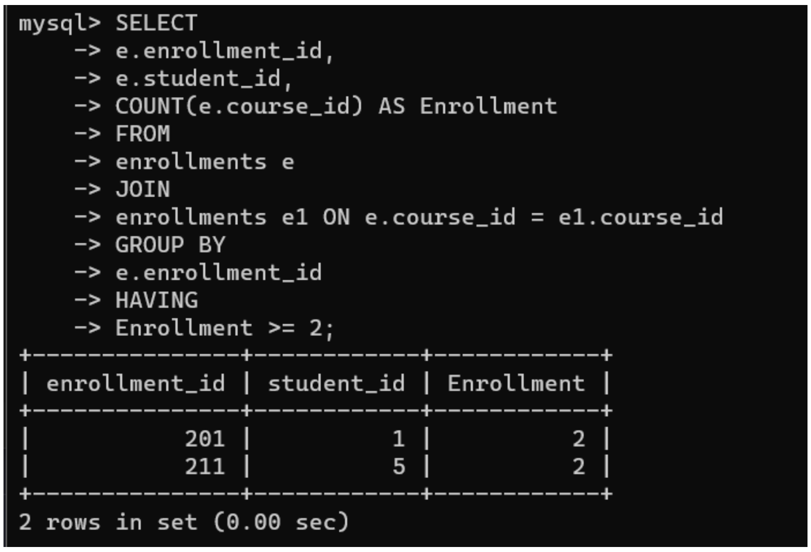
```
mysql> SELECT
    -> s.student_id,s.first_name,s.email
    -> FROM
    -> Students s
    -> LEFT JOIN
    -> Payments p
    -> ON s.student_id=p.student_id
    -> WHERE
    -> p.Amount IS NULL;
+------------+------------+----------------------+
| student_id | first_name | email                |
+------------+------------+----------------------+
|         11 | John       | john.doe@example.com |
+------------+------------+----------------------+
1 row in 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.



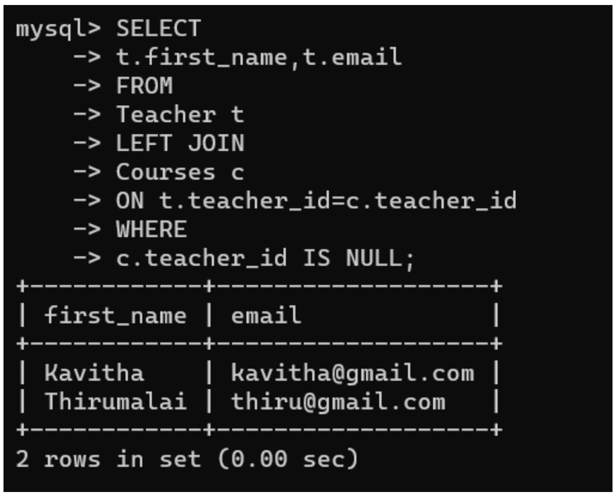
```
mysql> SELECT
    -> c.course_id,
    -> c.course_name,
    -> c.credits,
    -> c.teacher_id
    -> FROM
    -> Courses c
    -> LEFT JOIN
    -> Enrollments e ON c.course_id = e.course_id
    -> WHERE
    -> e.course_id IS NULL;
+-----------+--------------------+---------+------------+
| course_id | course_name        | credits | teacher_id |
+-----------+--------------------+---------+------------+
|       107 | Literature         |       3 |          8 |
|       111 | Physical Education |       0 |       NULL |
+-----------+--------------------+---------+------------+
2 rows 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
    -> e.enrollment_id,
    -> e.student_id,
    -> COUNT(e.course_id) AS Enrollment
    -> FROM
    -> enrollments e
    -> JOIN
    -> enrollments e1 ON e.course_id = e1.course_id
    -> GROUP BY
    -> e.enrollment_id
    -> HAVING
    -> Enrollment >= 2;
+---------------+------------+------------+
| enrollment_id | student_id | Enrollment |
+---------------+------------+------------+
|           201 |          1 |          2 |
|           211 |          5 |          2 |
+---------------+------------+------------+
2 rows in set (0.00 sec)
```

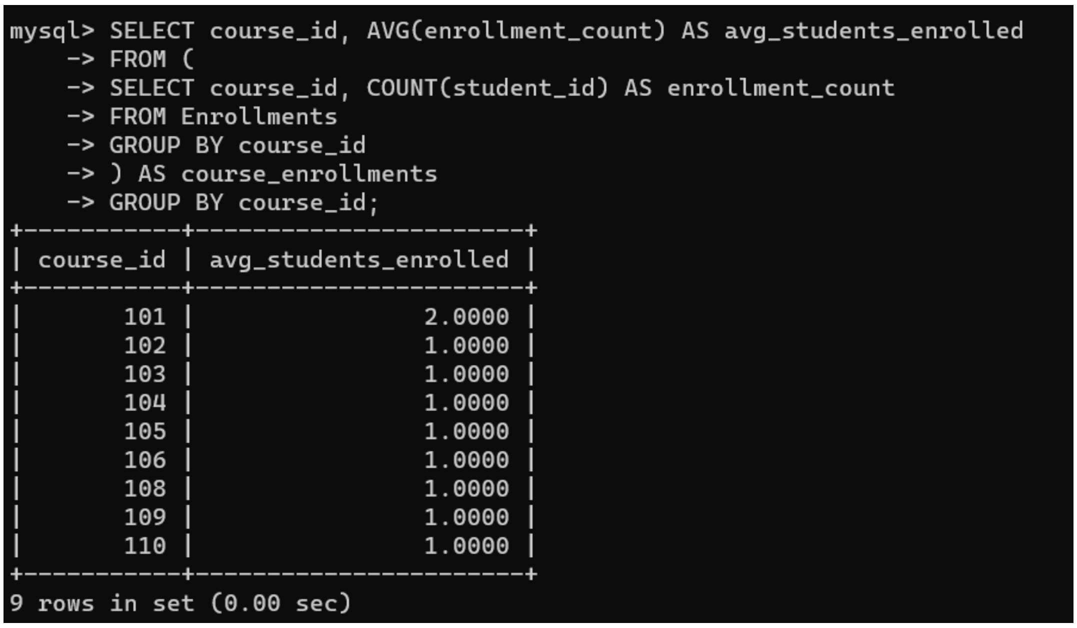
10. Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher" table and the "Courses" table and filter for teachers with NULL course assignments.



```
mysql> SELECT
    -> t.first_name,t.email
    -> FROM
    -> Teacher t
    -> LEFT JOIN
    -> Courses c
    -> ON t.teacher_id=c.teacher_id
    -> WHERE
    -> c.teacher_id IS NULL;
+------------+--------------------+
| first_name | email              |
+------------+--------------------+
| Kavitha    | kavitha@gmail.com |
| Thirumalai | thiru@gmail.com   |
+------------+--------------------+
2 rows in set (0.00 sec)
```

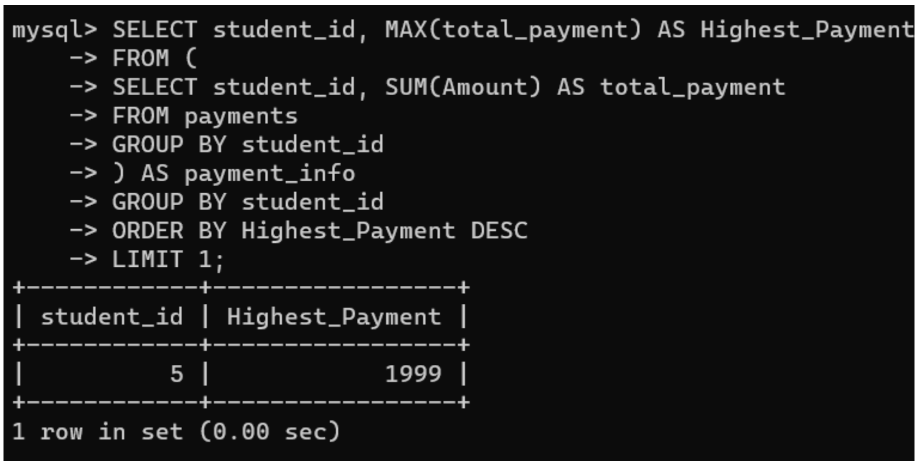
Task 4. Subquery and its type:

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



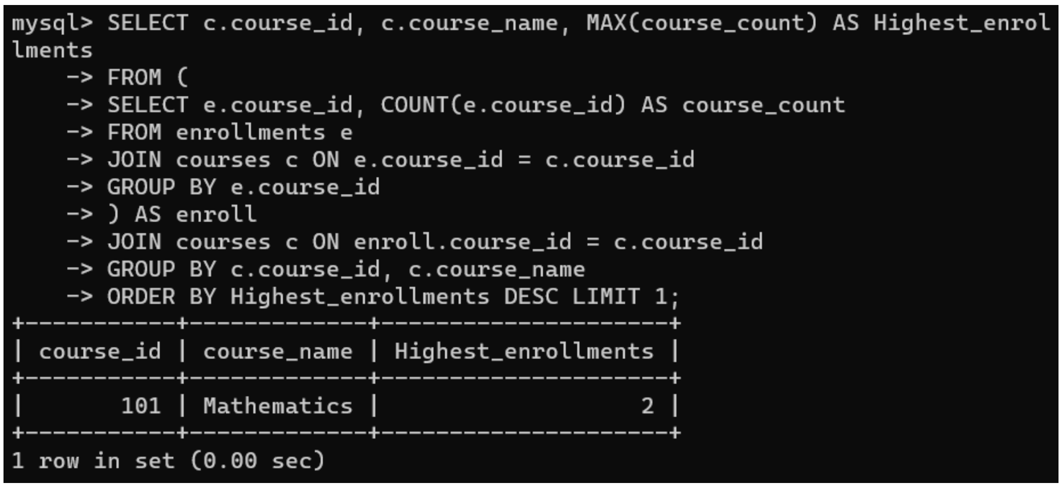
```
mysql> SELECT course_id, AVG(enrollment_count) AS avg_students_enrolled
    -> FROM (
    -> SELECT course_id, COUNT(student_id) AS enrollment_count
    -> FROM Enrollments
    -> GROUP BY course_id
    -> ) AS course_enrollments
    -> GROUP BY course_id;
+-----------+-----------------------+
| course_id | avg_students_enrolled |
+-----------+-----------------------+
|       101 |                2.0000 |
|       102 |                1.0000 |
|       103 |                1.0000 |
|       104 |                1.0000 |
|       105 |                1.0000 |
|       106 |                1.0000 |
|       108 |                1.0000 |
|       109 |                1.0000 |
|       110 |                1.0000 |
+-----------+-----------------------+
9 rows in set (0.00 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.



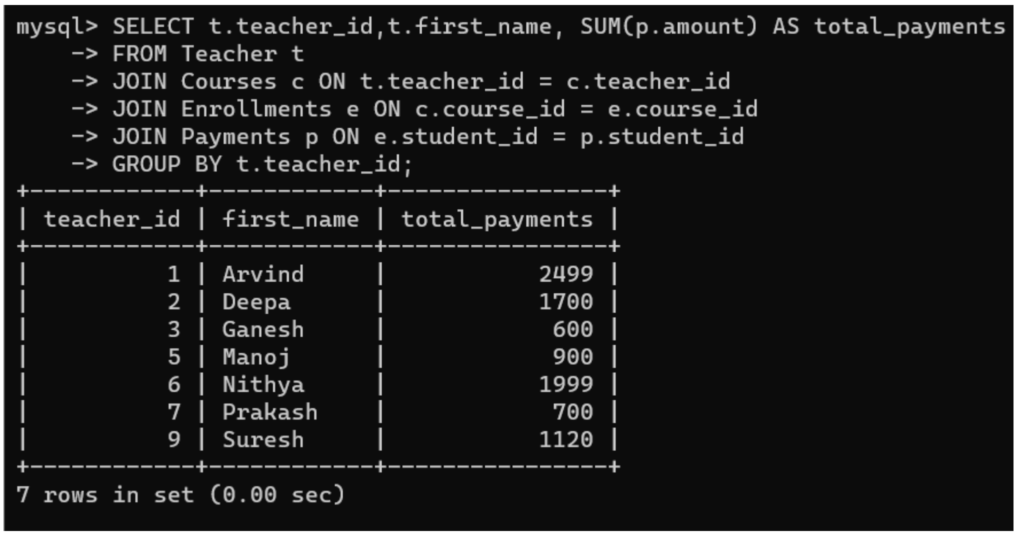
```
mysql> SELECT student_id, MAX(total_payment) AS Highest_Payment
    -> FROM (
    -> SELECT student_id, SUM(Amount) AS total_payment
    -> FROM payments
    -> GROUP BY student_id
    -> ) AS payment_info
    -> GROUP BY student_id
    -> ORDER BY Highest_Payment DESC
    -> LIMIT 1;
+------------+-----------------+
| student_id | Highest_Payment |
+------------+-----------------+
|          5 |            1999 |
+------------+-----------------+
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.



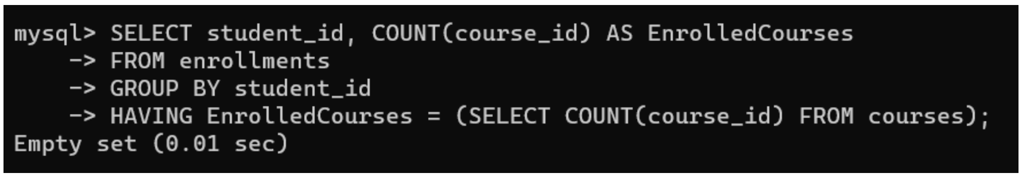
```
mysql> SELECT c.course_id, c.course_name, MAX(course_count) AS Highest_enrol
lments
    -> FROM (
    -> SELECT e.course_id, COUNT(e.course_id) AS course_count
    -> FROM enrollments e
    -> JOIN courses c ON e.course_id = c.course_id
    -> GROUP BY e.course_id
    -> ) AS enroll
    -> JOIN courses c ON enroll.course_id = c.course_id
    -> GROUP BY c.course_id, c.course_name
    -> ORDER BY Highest_enrollments DESC LIMIT 1;
+-----------+-------------+---------------------+
| course_id | course_name | Highest_enrollments |
+-----------+-------------+---------------------+
|       101 | Mathematics |                   2 |
+-----------+-------------+---------------------+
1 row 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.



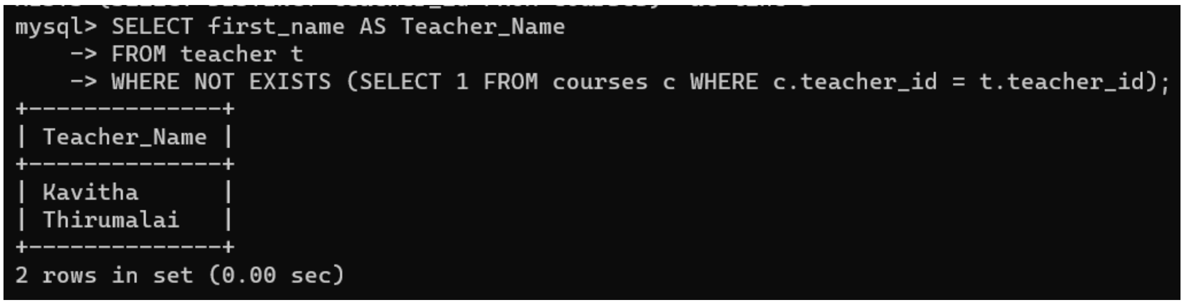
```
mysql> SELECT t.teacher_id,t.first_name, SUM(p.amount) AS total_payments
    -> FROM Teacher t
    -> JOIN Courses c ON t.teacher_id = c.teacher_id
    -> JOIN Enrollments e ON c.course_id = e.course_id
    -> JOIN Payments p ON e.student_id = p.student_id
    -> GROUP BY t.teacher_id;
+------------+------------+----------------+
| teacher_id | first_name | total_payments |
+------------+------------+----------------+
|          1 | Arvind     |           2499 |
|          2 | Deepa      |           1700 |
|          3 | Ganesh     |            600 |
|          5 | Manoj      |            900 |
|          6 | Nithya     |           1999 |
|          7 | Prakash    |            700 |
|          9 | Suresh     |           1120 |
+------------+------------+----------------+
7 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.



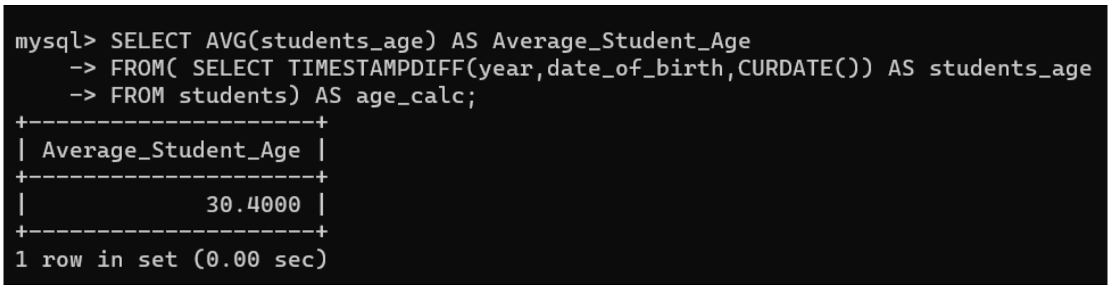
```
mysql> SELECT student_id, COUNT(course_id) AS EnrolledCourses
    -> FROM enrollments
    -> GROUP BY student_id
    -> HAVING EnrolledCourses = (SELECT COUNT(course_id) FROM courses);
Empty set (0.01 sec)
```

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



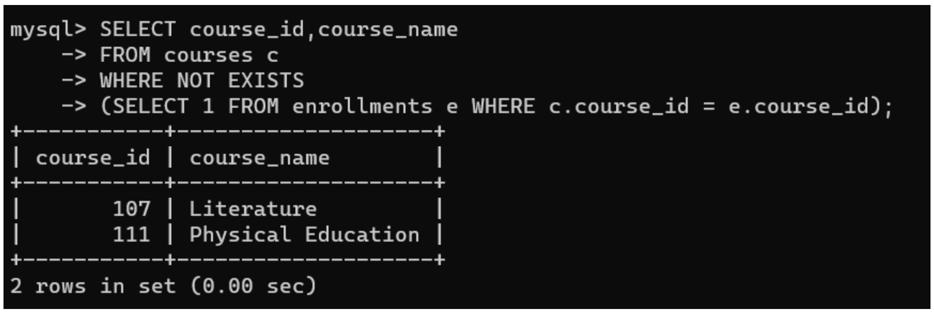
```
mysql> SELECT first_name AS Teacher_Name
    -> FROM teacher t
    -> WHERE NOT EXISTS (SELECT 1 FROM courses c WHERE c.teacher_id = t.teacher_id);
+--------------+
| Teacher_Name |
+--------------+
| Kavitha      |
| Thirumalai   |
+--------------+
2 rows in set (0.00 sec)
```

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



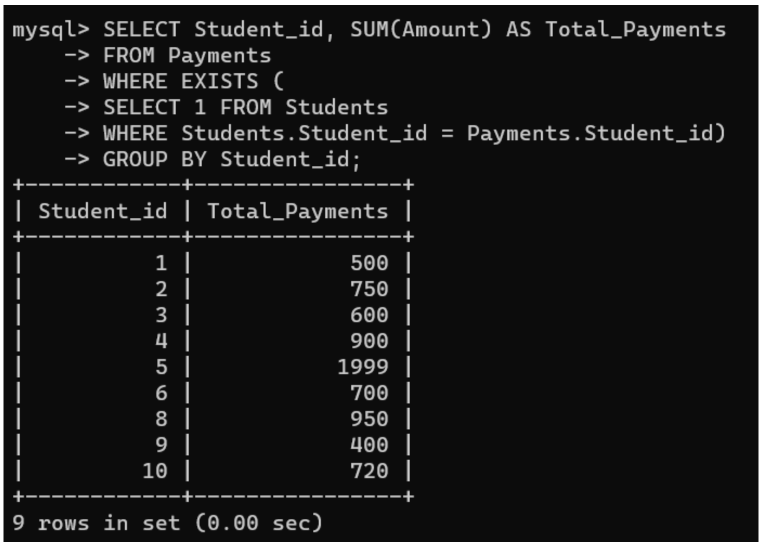
```
mysql> SELECT AVG(students_age) AS Average_Student_Age
    -> FROM( SELECT TIMESTAMPDIFF(year,date_of_birth,CURDATE()) AS students_age
    -> FROM students) AS age_calc;
+---------------------+
| Average_Student_Age |
+---------------------+
|             30.4000 |
+---------------------+
1 row in set (0.00 sec)
```

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



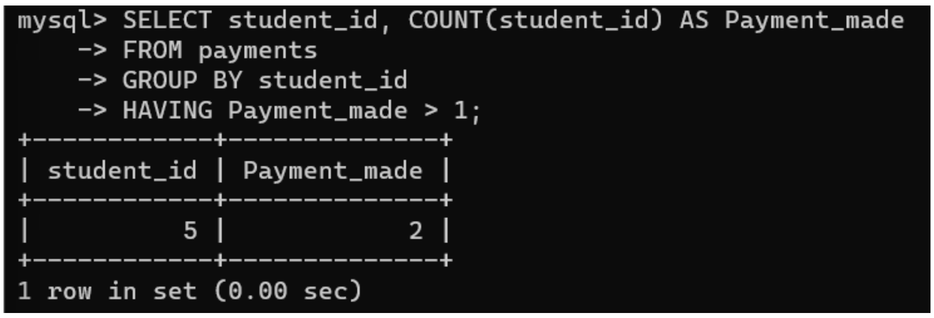
```
mysql> SELECT course_id,course_name
    -> FROM courses c
    -> WHERE NOT EXISTS
    -> (SELECT 1 FROM enrollments e WHERE c.course_id = e.course_id);
+-----------+--------------------+
| course_id | course_name        |
+-----------+--------------------+
|       107 | Literature         |
|       111 | Physical Education |
+-----------+--------------------+
2 rows 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.



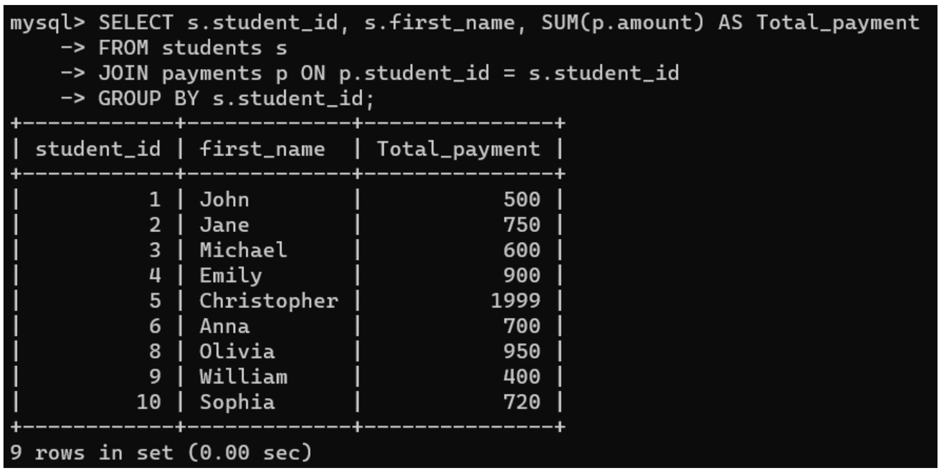
```
mysql> SELECT Student_id, SUM(Amount) AS Total_Payments
    -> FROM Payments
    -> WHERE EXISTS (
    -> SELECT 1 FROM Students
    -> WHERE Students.Student_id = Payments.Student_id)
    -> GROUP BY Student_id;
+------------+----------------+
| Student_id | Total_Payments |
+------------+----------------+
|          1 |            500 |
|          2 |            750 |
|          3 |            600 |
|          4 |            900 |
|          5 |           1999 |
|          6 |            700 |
|          8 |            950 |
|          9 |            400 |
|         10 |            720 |
+------------+----------------+
9 rows in set (0.00 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.



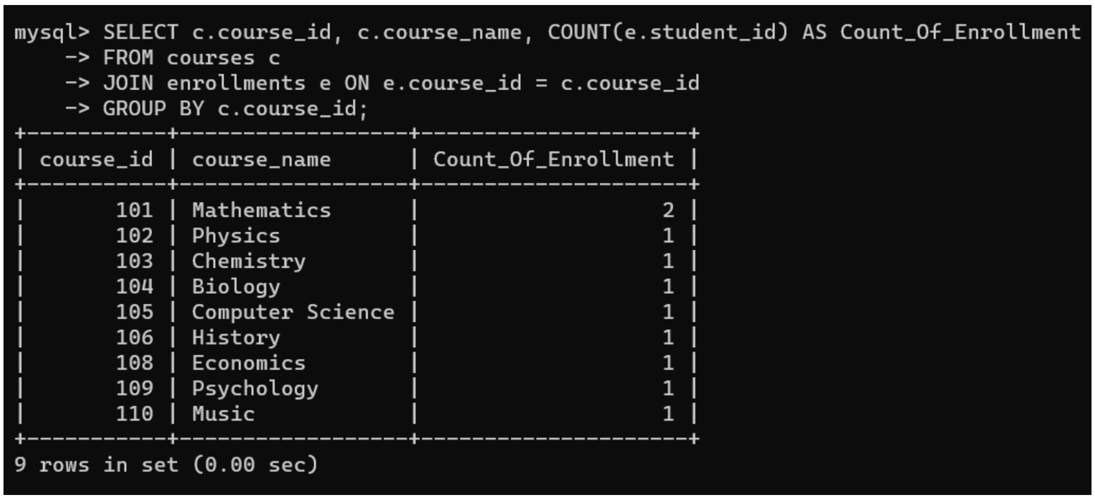
```
mysql> SELECT student_id, COUNT(student_id) AS Payment_made
    -> FROM payments
    -> GROUP BY student_id
    -> HAVING Payment_made > 1;
+------------+--------------+
| student_id | Payment_made |
+------------+--------------+
|          5 |            2 |
+------------+--------------+
1 row in 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.



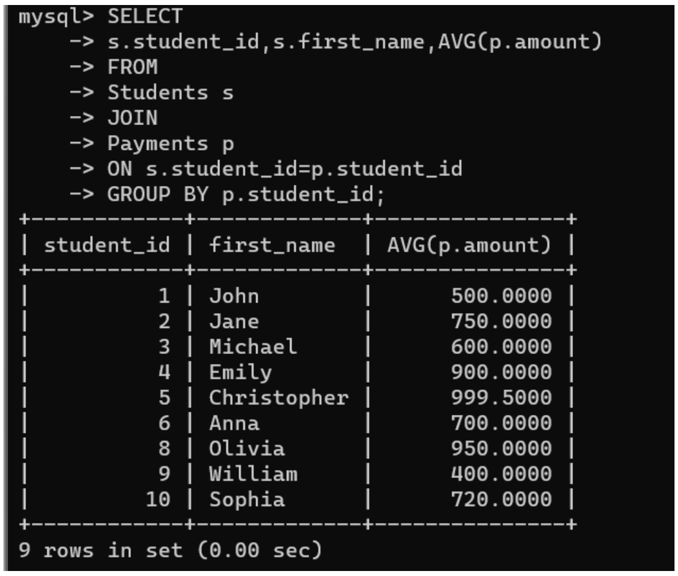
```
mysql> SELECT s.student_id, s.first_name, SUM(p.amount) AS Total_payment
    -> FROM students s
    -> JOIN payments p ON p.student_id = s.student_id
    -> GROUP BY s.student_id;
+------------+-------------+---------------+
| student_id | first_name  | Total_payment |
+------------+-------------+---------------+
|          1 | John        |           500 |
|          2 | Jane        |           750 |
|          3 | Michael     |           600 |
|          4 | Emily       |           900 |
|          5 | Christopher |          1999 |
|          6 | Anna        |           700 |
|          8 | Olivia      |           950 |
|          9 | William     |           400 |
|         10 | Sophia      |           720 |
+------------+-------------+---------------+
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_id, c.course_name, COUNT(e.student_id) AS Count_Of_Enrollment
    -> FROM courses c
    -> JOIN enrollments e ON e.course_id = c.course_id
    -> GROUP BY c.course_id;
+-----------+------------------+---------------------+
| course_id | course_name      | Count_Of_Enrollment |
+-----------+------------------+---------------------+
|       101 | Mathematics      |                   2 |
|       102 | Physics          |                   1 |
|       103 | Chemistry        |                   1 |
|       104 | Biology          |                   1 |
|       105 | Computer Science |                   1 |
|       106 | History          |                   1 |
|       108 | Economics        |                   1 |
|       109 | Psychology       |                   1 |
|       110 | 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,s.first_name,AVG(p.amount)
    -> FROM
    -> Students s
    -> JOIN
    -> Payments p
    -> ON s.student_id=p.student_id
    -> GROUP BY p.student_id;
+------------+-------------+---------------+
| student_id | first_name  | AVG(p.amount) |
+------------+-------------+---------------+
|          1 | John        |      500.0000 |
|          2 | Jane        |      750.0000 |
|          3 | Michael     |      600.0000 |
|          4 | Emily       |      900.0000 |
|          5 | Christopher |      999.5000 |
|          6 | Anna        |      700.0000 |
|          8 | Olivia      |      950.0000 |
|          9 | William     |      400.0000 |
|         10 | Sophia      |      720.0000 |
+------------+-------------+---------------+
9 rows in set (0.00 sec)
```