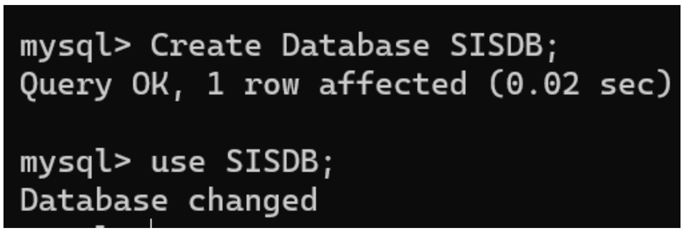
Assignment – 02

Task-1 Database Design:

1. Create the database named "SISDB"



```
mysql> Create Database SISDB;
Query OK, 1 row affected (0.02 sec)

mysql> use SISDB;
Database changed
```

2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema.

➔ **Students Table:**

Column Name	Data Type	Constraints
student_id	INT	PRIMARY KEY
first_name	VARCHAR(50)	NOT NULL
last_name	VARCHAR(50)	NOT NULL
date_of_birth	DATE	NOT NULL
email	VARCHAR(100)	UNIQUE
phone_number	VARCHAR(20)	NOT NULL

Courses Table:

Column Name	Data Type	Constraints
course_id	INT	PRIMARY KEY
course_name	VARCHAR(100)	NOT NULL
credits	INT	NOT NULL
teacher_id	INT	FOREIGN KEY REFERENCES Teacher(teacher_id)

Enrollments Table:

Column Name	Data Type	Constraints
enrollment_id	INT	PRIMARY KEY
student_id	INT	FOREIGN KEY REFERENCES Students(student_id)
course_id	INT	FOREIGN KEY REFERENCES Courses(course_id)
enrollment_date	DATE	NOT NULL

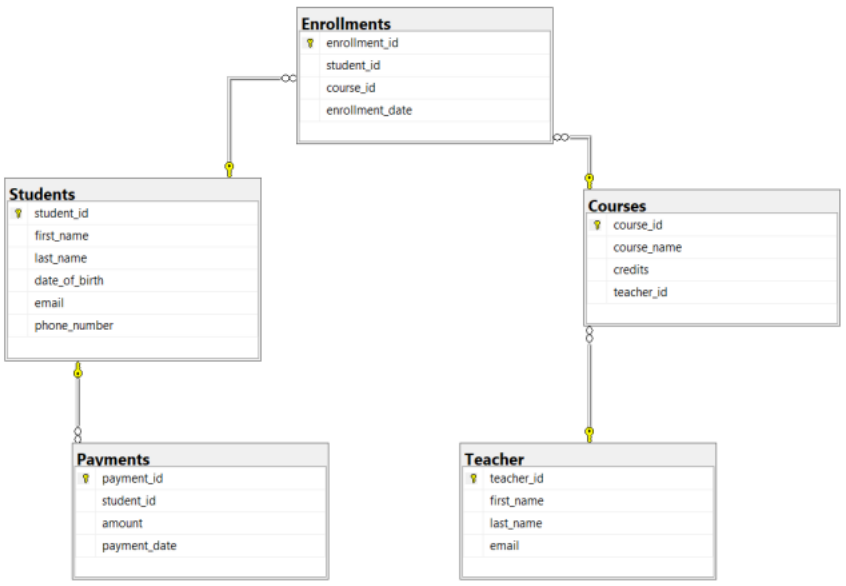
Teacher Table:

Column Name	Data Type	Constraints
teacher_id	INT	PRIMARY KEY
first_name	VARCHAR(50)	NOT NULL
last_name	VARCHAR(50)	NOT NULL
email	VARCHAR(100)	UNIQUE

Payments Table:

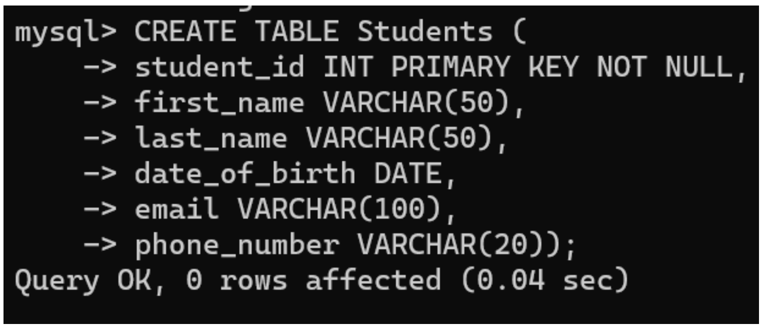
Column Name	Data Type	Constraints
payment_id	INT	PRIMARY KEY
student_id	INT	FOREIGN KEY REFERENCES Students(student_id)
amount	DECIMAL(10,2)	NOT NULL
payment_date	DATE	NOT NULL

3. ERD.



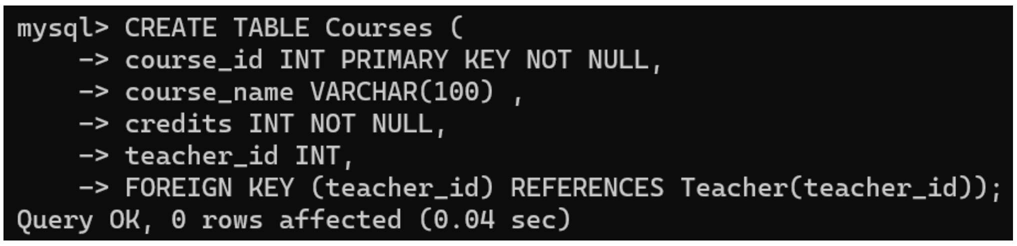
4. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships. Create appropriate Primary Key and Foreign Key constraints for referential integrity.Create appropriate Primary Key and Foreign Key constraints for referential integrity.

- Students:



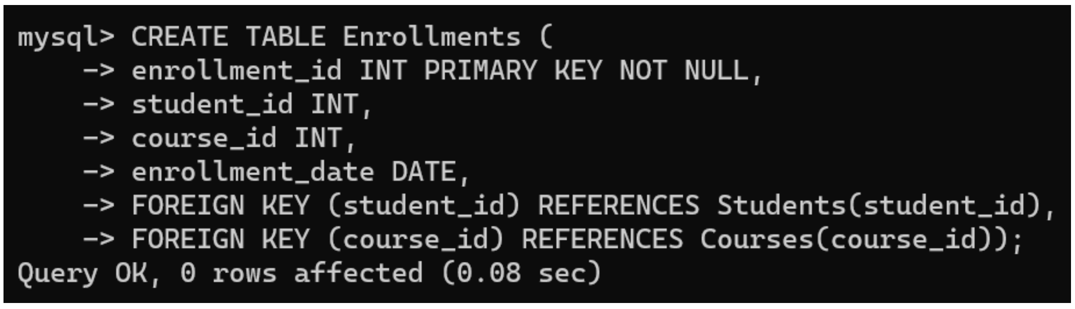
```
mysql> CREATE TABLE Students (
    -> student_id INT PRIMARY KEY NOT NULL,
    -> first_name VARCHAR(50),
    -> last_name VARCHAR(50),
    -> date_of_birth DATE,
    -> email VARCHAR(100),
    -> phone_number VARCHAR(20));
Query OK, 0 rows affected (0.04 sec)
```

- Courses:



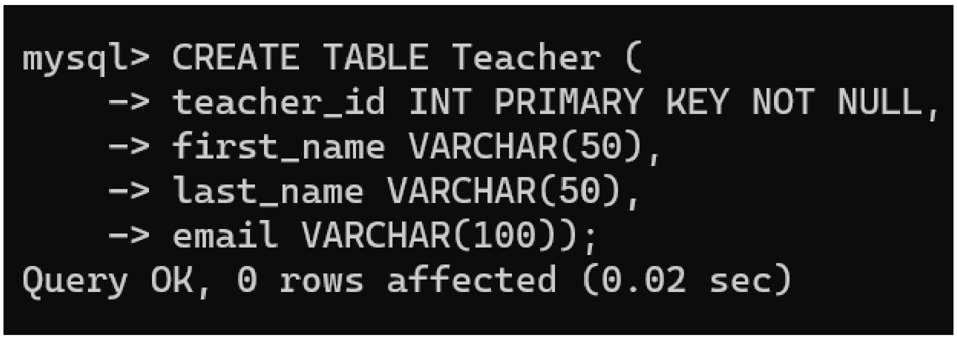
```
mysql> CREATE TABLE Courses (
    -> course_id INT PRIMARY KEY NOT NULL,
    -> course_name VARCHAR(100) ,
    -> credits INT NOT NULL,
    -> teacher_id INT,
    -> FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id));
Query OK, 0 rows affected (0.04 sec)
```

- Enrollments:



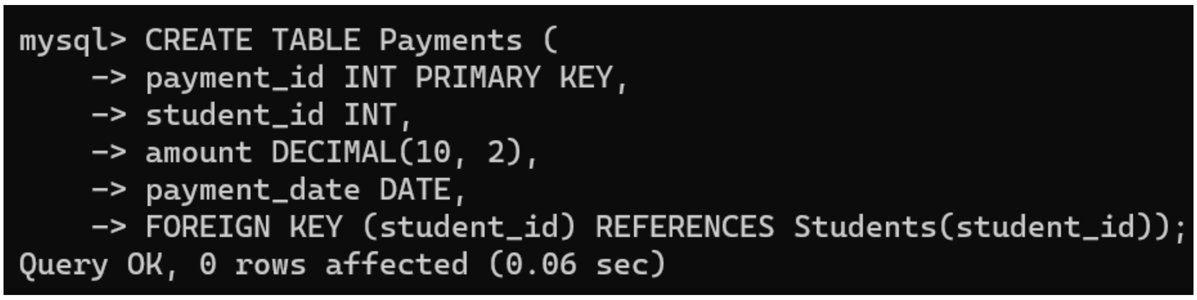
```
mysql> CREATE TABLE Enrollments (
    -> enrollment_id INT PRIMARY KEY NOT NULL,
    -> student_id INT,
    -> course_id INT,
    -> enrollment_date DATE,
    -> FOREIGN KEY (student_id) REFERENCES Students(student_id),
    -> FOREIGN KEY (course_id) REFERENCES Courses(course_id));
Query OK, 0 rows affected (0.08 sec)
```

- Teacher:



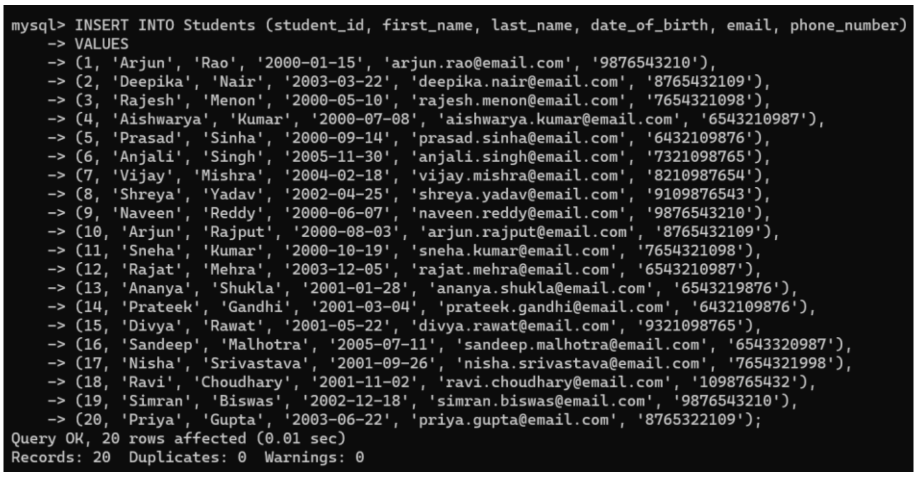
```
mysql> CREATE TABLE Teacher (
    -> teacher_id INT PRIMARY KEY NOT NULL,
    -> first_name VARCHAR(50),
    -> last_name VARCHAR(50),
    -> email VARCHAR(100));
Query OK, 0 rows affected (0.02 sec)
```

- Payments:



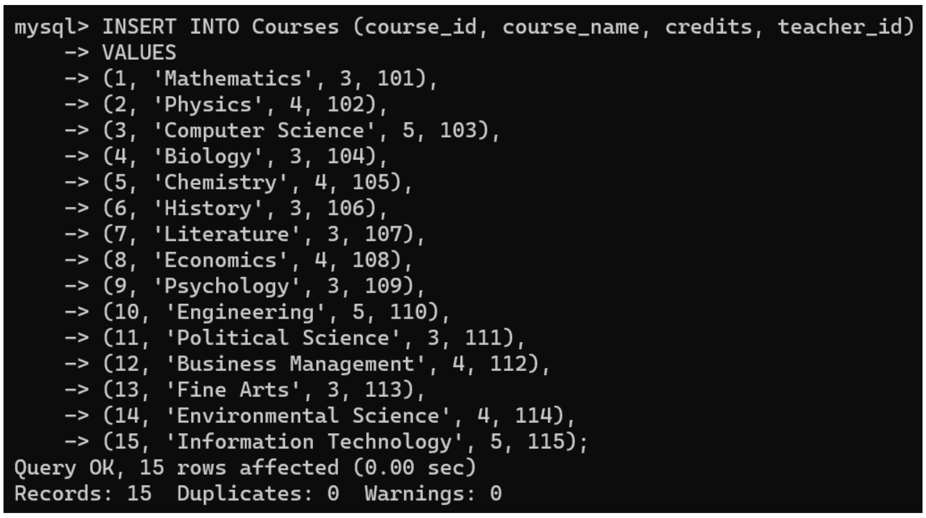
```
mysql> CREATE TABLE Payments (
    -> payment_id INT PRIMARY KEY,
    -> student_id INT,
    -> amount DECIMAL(10, 2),
    -> payment_date DATE,
    -> FOREIGN KEY (student_id) REFERENCES Students(student_id));
Query OK, 0 rows affected (0.06 sec)
```

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



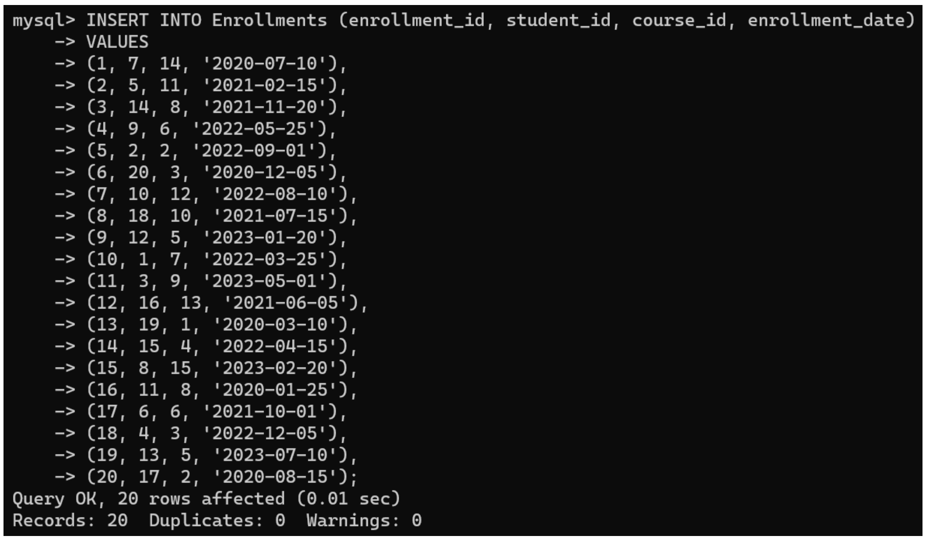
```
mysql> INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number)
    -> VALUES
    -> (1, 'Arjun', 'Rao', '2000-01-15', 'arjun.rao@email.com', '9876543210'),
    -> (2, 'Deepika', 'Nair', '2003-03-22', 'deepika.nair@email.com', '8765432109'),
    -> (3, 'Rajesh', 'Menon', '2000-05-10', 'rajesh.menon@email.com', '7654321098'),
    -> (4, 'Aishwarya', 'Kumar', '2000-07-08', 'aishwarya.kumar@email.com', '6543210987'),
    -> (5, 'Prasad', 'Sinha', '2000-09-14', 'prasad.sinha@email.com', '6432109876'),
    -> (6, 'Anjali', 'Singh', '2005-11-30', 'anjali.singh@email.com', '7321098765'),
    -> (7, 'Vijay', 'Mishra', '2004-02-18', 'vijay.mishra@email.com', '8210987654'),
    -> (8, 'Shreya', 'Yadav', '2002-04-25', 'shreya.yadav@email.com', '9109876543'),
    -> (9, 'Naveen', 'Reddy', '2000-06-07', 'naveen.reddy@email.com', '9876543210'),
    -> (10, 'Arjun', 'Rajput', '2000-08-03', 'arjun.rajput@email.com', '8765432109'),
    -> (11, 'Sneha', 'Kumar', '2000-10-19', 'sneha.kumar@email.com', '7654321098'),
    -> (12, 'Rajat', 'Mehra', '2003-12-05', 'rajat.mehra@email.com', '6543210987'),
    -> (13, 'Ananya', 'Shukla', '2001-01-28', 'ananya.shukla@email.com', '6543219876'),
    -> (14, 'Prateek', 'Gandhi', '2001-03-04', 'prateek.gandhi@email.com', '6432109876'),
    -> (15, 'Divya', 'Rawat', '2001-05-22', 'divya.rawat@email.com', '9321098765'),
    -> (16, 'Sandeep', 'Malhotra', '2005-07-11', 'sandeep.malhotra@email.com', '6543320987'),
    -> (17, 'Nisha', 'Srivastava', '2001-09-26', 'nisha.srivastava@email.com', '7654321998'),
    -> (18, 'Ravi', 'Choudhary', '2001-11-02', 'ravi.choudhary@email.com', '1098765432'),
    -> (19, 'Simran', 'Biswas', '2002-12-18', 'simran.biswas@email.com', '9876543210'),
    -> (20, 'Priya', 'Gupta', '2003-06-22', 'priya.gupta@email.com', '8765322109');
Query OK, 20 rows affected (0.01 sec)
Records: 20  Duplicates: 0  Warnings: 0
```

- Courses:



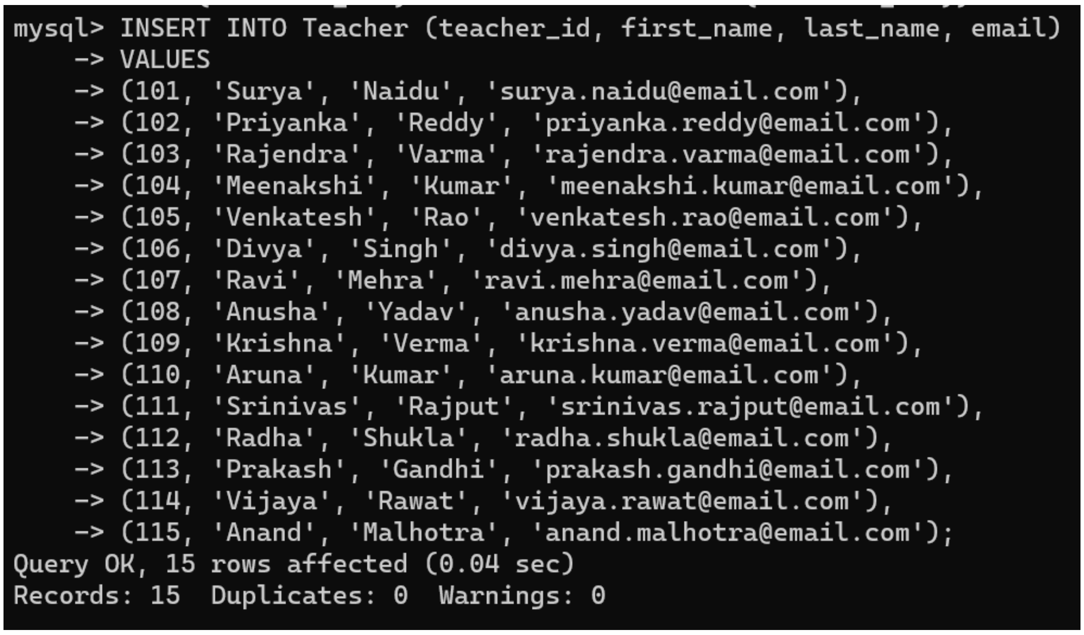
```
mysql> INSERT INTO Courses (course_id, course_name, credits, teacher_id)
    -> VALUES
    -> (1, 'Mathematics', 3, 101),
    -> (2, 'Physics', 4, 102),
    -> (3, 'Computer Science', 5, 103),
    -> (4, 'Biology', 3, 104),
    -> (5, 'Chemistry', 4, 105),
    -> (6, 'History', 3, 106),
    -> (7, 'Literature', 3, 107),
    -> (8, 'Economics', 4, 108),
    -> (9, 'Psychology', 3, 109),
    -> (10, 'Engineering', 5, 110),
    -> (11, 'Political Science', 3, 111),
    -> (12, 'Business Management', 4, 112),
    -> (13, 'Fine Arts', 3, 113),
    -> (14, 'Environmental Science', 4, 114),
    -> (15, 'Information Technology', 5, 115);
Query OK, 15 rows affected (0.00 sec)
Records: 15  Duplicates: 0  Warnings: 0
```

- Enrollments:



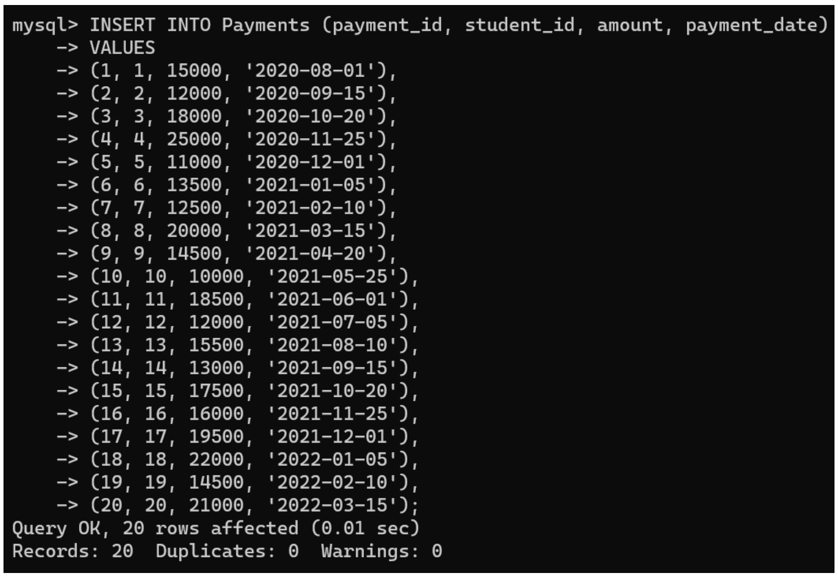
```
mysql> INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
    -> VALUES
    -> (1, 7, 14, '2020-07-10'),
    -> (2, 5, 11, '2021-02-15'),
    -> (3, 14, 8, '2021-11-20'),
    -> (4, 9, 6, '2022-05-25'),
    -> (5, 2, 2, '2022-09-01'),
    -> (6, 20, 3, '2020-12-05'),
    -> (7, 10, 12, '2022-08-10'),
    -> (8, 18, 10, '2021-07-15'),
    -> (9, 12, 5, '2023-01-20'),
    -> (10, 1, 7, '2022-03-25'),
    -> (11, 3, 9, '2023-05-01'),
    -> (12, 16, 13, '2021-06-05'),
    -> (13, 19, 1, '2020-03-10'),
    -> (14, 15, 4, '2022-04-15'),
    -> (15, 8, 15, '2023-02-20'),
    -> (16, 11, 8, '2020-01-25'),
    -> (17, 6, 6, '2021-10-01'),
    -> (18, 4, 3, '2022-12-05'),
    -> (19, 13, 5, '2023-07-10'),
    -> (20, 17, 2, '2020-08-15');
Query OK, 20 rows affected (0.01 sec)
Records: 20  Duplicates: 0  Warnings: 0
```

- Teacher:



```
mysql> INSERT INTO Teacher (teacher_id, first_name, last_name, email)
    -> VALUES
    -> (101, 'Surya', 'Naidu', 'surya.naidu@email.com'),
    -> (102, 'Priyanka', 'Reddy', 'priyanka.reddy@email.com'),
    -> (103, 'Rajendra', 'Varma', 'rajendra.varma@email.com'),
    -> (104, 'Meenakshi', 'Kumar', 'meenakshi.kumar@email.com'),
    -> (105, 'Venkatesh', 'Rao', 'venkatesh.rao@email.com'),
    -> (106, 'Divya', 'Singh', 'divya.singh@email.com'),
    -> (107, 'Ravi', 'Mehra', 'ravi.mehra@email.com'),
    -> (108, 'Anusha', 'Yadav', 'anusha.yadav@email.com'),
    -> (109, 'Krishna', 'Verma', 'krishna.verma@email.com'),
    -> (110, 'Aruna', 'Kumar', 'aruna.kumar@email.com'),
    -> (111, 'Srinivas', 'Rajput', 'srinivas.rajput@email.com'),
    -> (112, 'Radha', 'Shukla', 'radha.shukla@email.com'),
    -> (113, 'Prakash', 'Gandhi', 'prakash.gandhi@email.com'),
    -> (114, 'Vijaya', 'Rawat', 'vijaya.rawat@email.com'),
    -> (115, 'Anand', 'Malhotra', 'anand.malhotra@email.com');
Query OK, 15 rows affected (0.04 sec)
Records: 15  Duplicates: 0  Warnings: 0
```

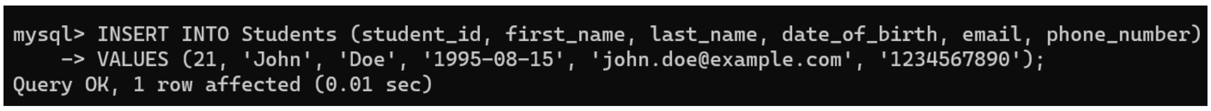
- Payments:



```
mysql> INSERT INTO Payments (payment_id, student_id, amount, payment_date)
    -> VALUES
    -> (1, 1, 15000, '2020-08-01'),
    -> (2, 2, 12000, '2020-09-15'),
    -> (3, 3, 18000, '2020-10-20'),
    -> (4, 4, 25000, '2020-11-25'),
    -> (5, 5, 11000, '2020-12-01'),
    -> (6, 6, 13500, '2021-01-05'),
    -> (7, 7, 12500, '2021-02-10'),
    -> (8, 8, 20000, '2021-03-15'),
    -> (9, 9, 14500, '2021-04-20'),
    -> (10, 10, 10000, '2021-05-25'),
    -> (11, 11, 18500, '2021-06-01'),
    -> (12, 12, 12000, '2021-07-05'),
    -> (13, 13, 15500, '2021-08-10'),
    -> (14, 14, 13000, '2021-09-15'),
    -> (15, 15, 17500, '2021-10-20'),
    -> (16, 16, 16000, '2021-11-25'),
    -> (17, 17, 19500, '2021-12-01'),
    -> (18, 18, 22000, '2022-01-05'),
    -> (19, 19, 14500, '2022-02-10'),
    -> (20, 20, 21000, '2022-03-15');
Query OK, 20 rows affected (0.01 sec)
Records: 20  Duplicates: 0  Warnings: 0
```

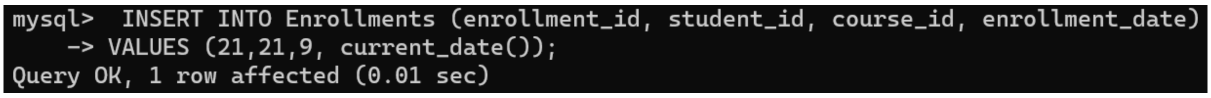
<u>Task-2 Select, Where, Between, AND, Like:</u>

1. Write an SQL query to insert a new student into the "Students" table with the following details:
    - First Name: John
    - Last Name: Doe
    - Date of Birth: 1995-08-15
    - Email: john.doe@example.com
    - Phone Number: 1234567890



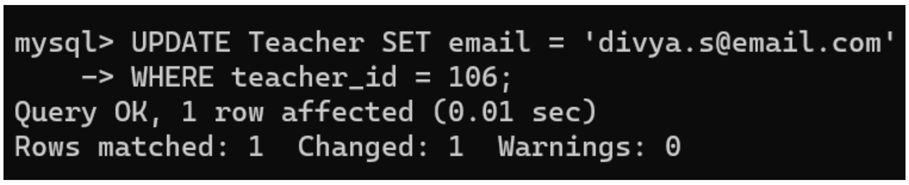
```
mysql> INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number)
    -> VALUES (21, 'John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
Query OK, 1 row affected (0.01 sec)
```

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.



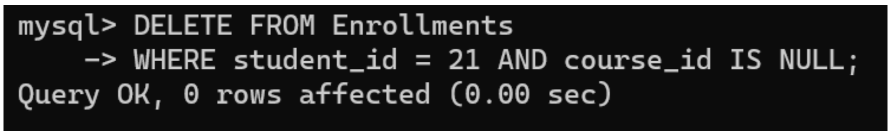
```
mysql>  INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
    -> VALUES (21,21,9, current_date());
Query OK, 1 row affected (0.01 sec)
```

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



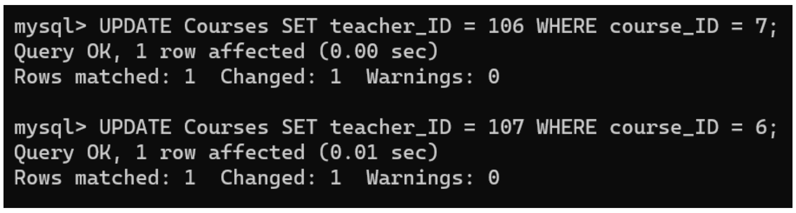
```
mysql> UPDATE Teacher SET email = 'divya.s@email.com'
    -> WHERE teacher_id = 106;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

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



```
mysql> DELETE FROM Enrollments
    -> WHERE student_id = 21 AND course_id IS NULL;
Query OK, 0 rows affected (0.00 sec)
```

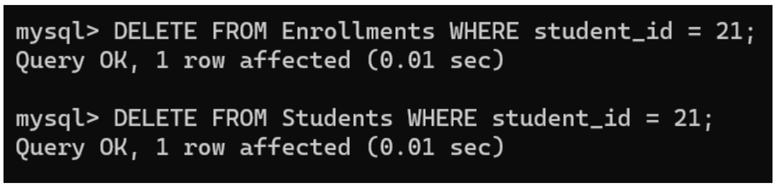
5. Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.



```
mysql> UPDATE Courses SET teacher_ID = 106 WHERE course_ID = 7;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE Courses SET teacher_ID = 107 WHERE course_ID = 6;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

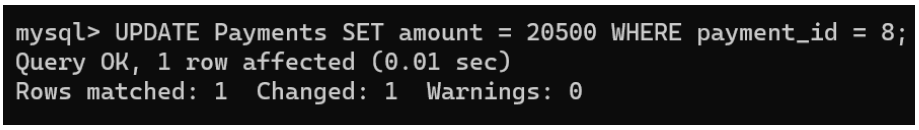
6. Delete a specific student from the "Students" table and remove all their enrollment records from the "Enrollments" table. Be sure to maintain referential integrity.



```
mysql> DELETE FROM Enrollments WHERE student_id = 21;
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM Students WHERE student_id = 21;
Query OK, 1 row affected (0.01 sec)
```

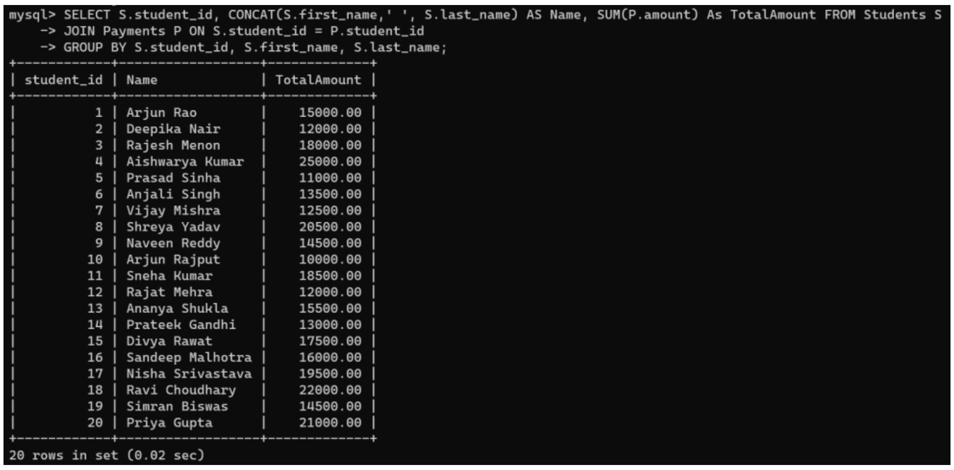
7. Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.



```
mysql> UPDATE Payments SET amount = 20500 WHERE payment_id = 8;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

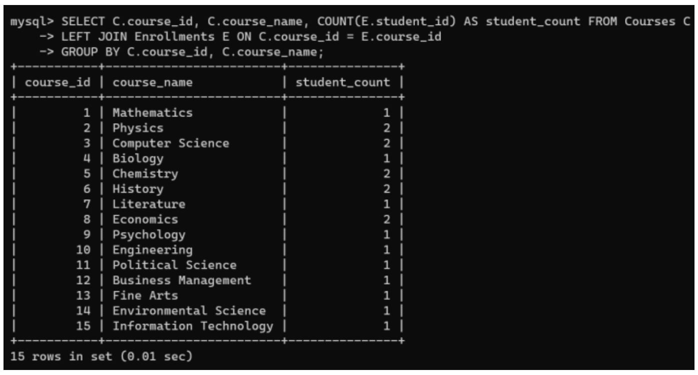
Task-3 Aggregate functions, Having, Order By, Group By and Joins:

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.



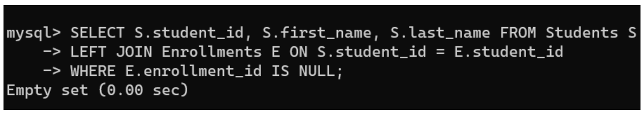
```
mysql> SELECT S.student_id, CONCAT(S.first_name,' ', S.last_name) AS Name, SUM(P.amount) As TotalAmount FROM Students S
    -> JOIN Payments P ON S.student_id = P.student_id
    -> GROUP BY S.student_id, S.first_name, S.last_name;
+------------+-------------------+-------------+
| student_id | Name              | TotalAmount |
+------------+-------------------+-------------+
|          1 | Arjun Rao         |    15000.00 |
|          2 | Deepika Nair      |    12000.00 |
|          3 | Rajesh Menon      |    18000.00 |
|          4 | Aishwarya Kumar   |    25000.00 |
|          5 | Prasad Sinha      |    11000.00 |
|          6 | Anjali Singh      |    13500.00 |
|          7 | Vijay Mishra      |    12500.00 |
|          8 | Shreya Yadav      |    20500.00 |
|          9 | Naveen Reddy      |    14500.00 |
|         10 | Arjun Rajput      |    10000.00 |
|         11 | Sneha Kumar       |    18500.00 |
|         12 | Rajat Mehra       |    12000.00 |
|         13 | Ananya Shukla     |    15500.00 |
|         14 | Prateek Gandhi    |    13000.00 |
|         15 | Divya Rawat       |    17500.00 |
|         16 | Sandeep Malhotra  |    16000.00 |
|         17 | Nisha Srivastava  |    19500.00 |
|         18 | Ravi Choudhary    |    22000.00 |
|         19 | Simran Biswas     |    14500.00 |
|         20 | Priya Gupta       |    21000.00 |
+------------+-------------------+-------------+
20 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.



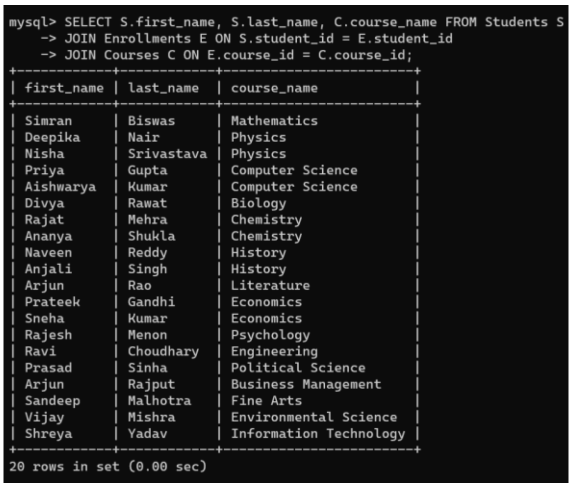
```
mysql> SELECT C.course_id, C.course_name, COUNT(E.student_id) AS student_count FROM Courses C
    -> LEFT JOIN Enrollments E ON C.course_id = E.course_id
    -> GROUP BY C.course_id, C.course_name;
+-----------+------------------------+---------------+
| course_id | course_name            | student_count |
+-----------+------------------------+---------------+
|         1 | Mathematics            |             1 |
|         2 | Physics                |             2 |
|         3 | Computer Science       |             2 |
|         4 | Biology                |             1 |
|         5 | Chemistry              |             2 |
|         6 | History                |             2 |
|         7 | Literature             |             1 |
|         8 | Economics              |             2 |
|         9 | Psychology             |             1 |
|        10 | Engineering            |             1 |
|        11 | Political Science      |             1 |
|        12 | Business Management    |             1 |
|        13 | Fine Arts              |             1 |
|        14 | Environmental Science  |             1 |
|        15 | Information Technology |             1 |
+-----------+------------------------+---------------+
15 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.



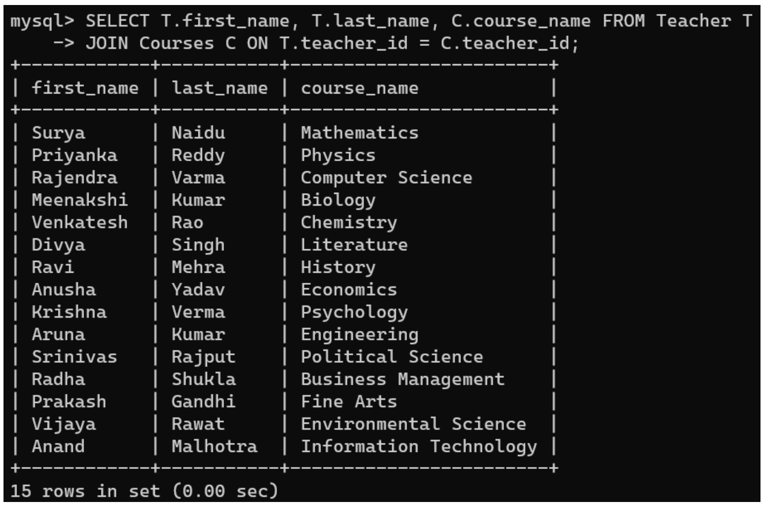
```
mysql> SELECT S.student_id, S.first_name, S.last_name FROM Students S
    -> LEFT JOIN Enrollments E ON S.student_id = E.student_id
    -> WHERE E.enrollment_id IS NULL;
Empty set (0.00 sec)
```

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.



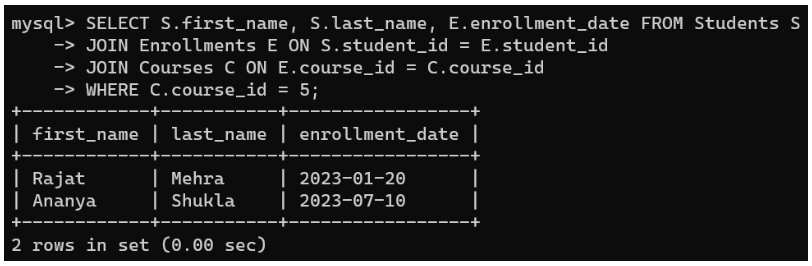
```
mysql> SELECT S.first_name, S.last_name, C.course_name FROM Students S
    -> JOIN Enrollments E ON S.student_id = E.student_id
    -> JOIN Courses C ON E.course_id = C.course_id;
+------------+------------+------------------------+
| first_name | last_name  | course_name            |
+------------+------------+------------------------+
| Simran     | Biswas     | Mathematics            |
| Deepika    | Nair       | Physics                |
| Nisha      | Srivastava | Physics                |
| Priya      | Gupta      | Computer Science       |
| Aishwarya  | Kumar      | Computer Science       |
| Divya      | Rawat      | Biology                |
| Rajat      | Mehra      | Chemistry              |
| Ananya     | Shukla     | Chemistry              |
| Naveen     | Reddy      | History                |
| Anjali     | Singh      | History                |
| Arjun      | Rao        | Literature             |
| Prateek    | Gandhi     | Economics              |
| Sneha      | Kumar      | Economics              |
| Rajesh     | Menon      | Psychology             |
| Ravi       | Choudhary  | Engineering            |
| Prasad     | Sinha      | Political Science      |
| Arjun      | Rajput     | Business Management    |
| Sandeep    | Malhotra   | Fine Arts              |
| Vijay      | Mishra     | Environmental Science  |
| Shreya     | Yadav      | Information Technology |
+------------+------------+------------------------+
20 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.



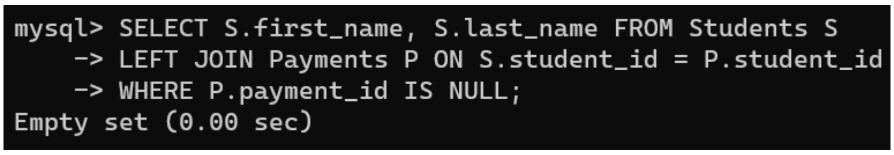
```
mysql> SELECT T.first_name, T.last_name, C.course_name FROM Teacher T
    -> JOIN Courses C ON T.teacher_id = C.teacher_id;
+------------+-----------+------------------------+
| first_name | last_name | course_name            |
+------------+-----------+------------------------+
| Surya      | Naidu     | Mathematics            |
| Priyanka   | Reddy     | Physics                |
| Rajendra   | Varma     | Computer Science       |
| Meenakshi  | Kumar     | Biology                |
| Venkatesh  | Rao       | Chemistry              |
| Divya      | Singh     | Literature             |
| Ravi       | Mehra     | History                |
| Anusha     | Yadav     | Economics              |
| Krishna    | Verma     | Psychology             |
| Aruna      | Kumar     | Engineering            |
| Srinivas   | Rajput    | Political Science      |
| Radha      | Shukla    | Business Management    |
| Prakash    | Gandhi    | Fine Arts              |
| Vijaya     | Rawat     | Environmental Science  |
| Anand      | Malhotra  | Information Technology |
+------------+-----------+------------------------+
15 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.



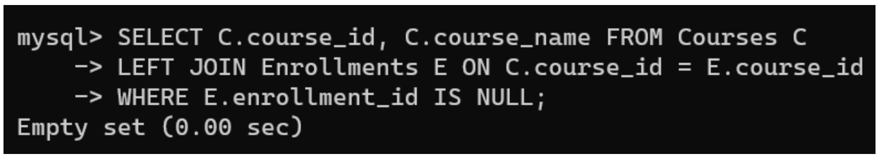
```
mysql> SELECT S.first_name, S.last_name, E.enrollment_date FROM Students S
    -> JOIN Enrollments E ON S.student_id = E.student_id
    -> JOIN Courses C ON E.course_id = C.course_id
    -> WHERE C.course_id = 5;
+------------+-----------+-----------------+
| first_name | last_name | enrollment_date |
+------------+-----------+-----------------+
| Rajat      | Mehra     | 2023-01-20      |
| Ananya     | Shukla    | 2023-07-10      |
+------------+-----------+-----------------+
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.



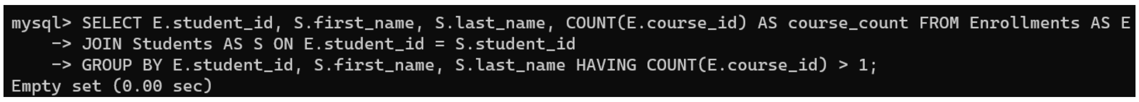
```
mysql> SELECT S.first_name, S.last_name FROM Students S
    -> LEFT JOIN Payments P ON S.student_id = P.student_id
    -> WHERE P.payment_id IS NULL;
Empty set (0.00 sec)
```

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



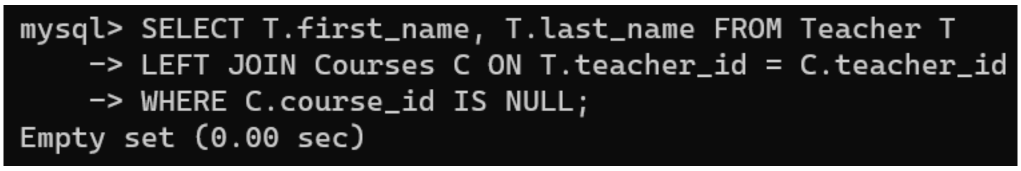
```
mysql> SELECT C.course_id, C.course_name FROM Courses C
    -> LEFT JOIN Enrollments E ON C.course_id = E.course_id
    -> WHERE E.enrollment_id IS NULL;
Empty set (0.00 sec)
```

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



```
mysql> SELECT E.student_id, S.first_name, S.last_name, COUNT(E.course_id) AS course_count FROM Enrollments AS E
    -> JOIN Students AS S ON E.student_id = S.student_id
    -> GROUP BY E.student_id, S.first_name, S.last_name HAVING COUNT(E.course_id) > 1;
Empty set (0.00 sec)
```

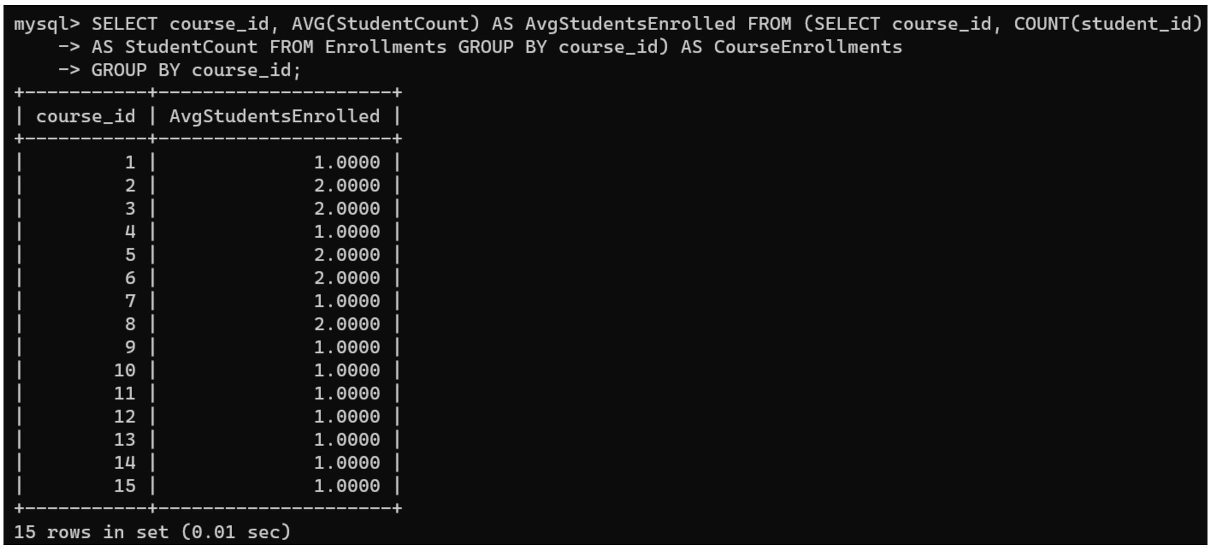
10. Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher" table and the "Courses" table and filter for teachers with NULL course assignments



```
mysql> SELECT T.first_name, T.last_name FROM Teacher T
    -> LEFT JOIN Courses C ON T.teacher_id = C.teacher_id
    -> WHERE C.course_id IS NULL;
Empty set (0.00 sec)
```

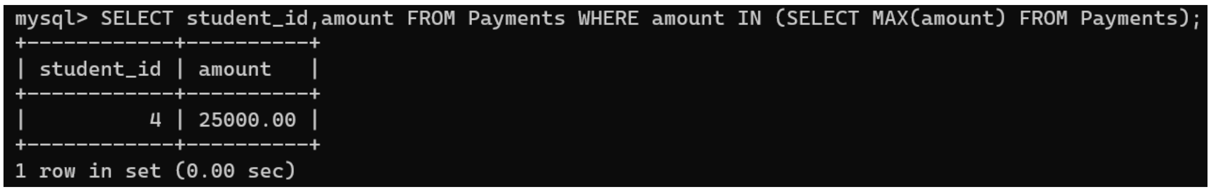
Task-4 Subquery and its type:

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



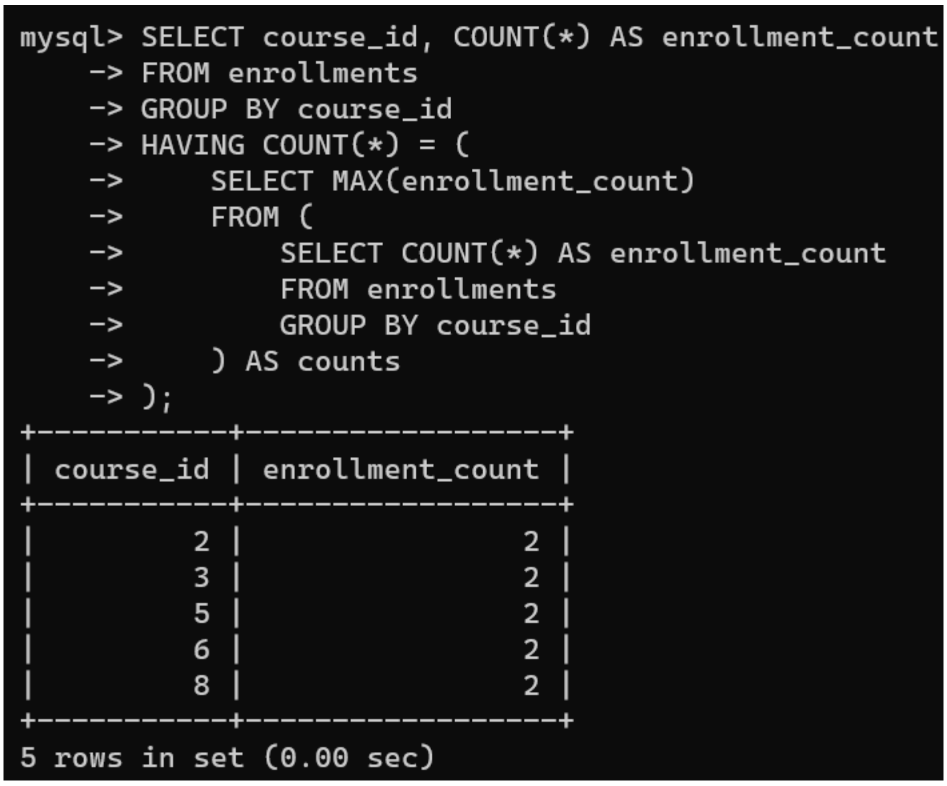
```
mysql> SELECT course_id, AVG(StudentCount) AS AvgStudentsEnrolled FROM (SELECT course_id, COUNT(student_id)
    -> AS StudentCount FROM Enrollments GROUP BY course_id) AS CourseEnrollments
    -> GROUP BY course_id;
+-----------+---------------------+
| course_id | AvgStudentsEnrolled |
+-----------+---------------------+
|         1 |              1.0000 |
|         2 |              2.0000 |
|         3 |              2.0000 |
|         4 |              1.0000 |
|         5 |              2.0000 |
|         6 |              2.0000 |
|         7 |              1.0000 |
|         8 |              2.0000 |
|         9 |              1.0000 |
|        10 |              1.0000 |
|        11 |              1.0000 |
|        12 |              1.0000 |
|        13 |              1.0000 |
|        14 |              1.0000 |
|        15 |              1.0000 |
+-----------+---------------------+
15 rows in set (0.01 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.



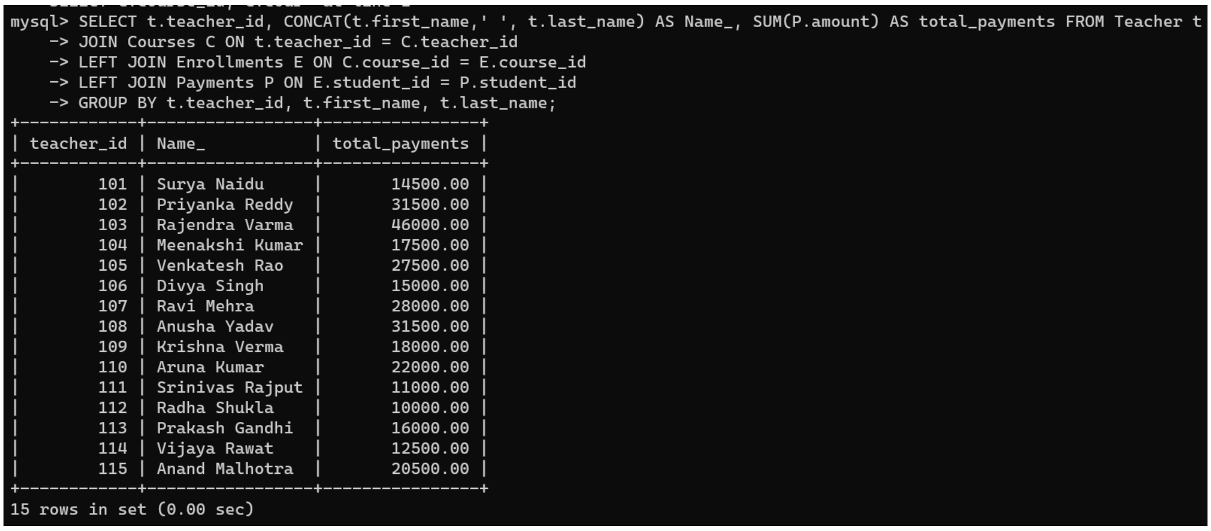
```
mysql> SELECT student_id,amount FROM Payments WHERE amount IN (SELECT MAX(amount) FROM Payments);
+------------+----------+
| student_id | amount   |
+------------+----------+
|          4 | 25000.00 |
+------------+----------+
1 row in set (0.00 sec)
```

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



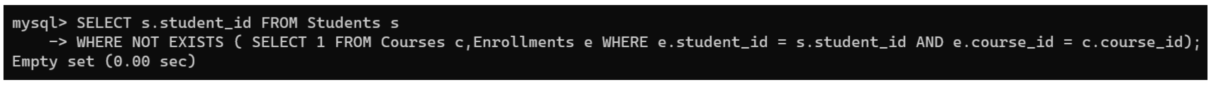
```
mysql> SELECT course_id, COUNT(*) AS enrollment_count
    -> FROM enrollments
    -> GROUP BY course_id
    -> HAVING COUNT(*) = (
    ->     SELECT MAX(enrollment_count)
    ->     FROM (
    ->         SELECT COUNT(*) AS enrollment_count
    ->         FROM enrollments
    ->         GROUP BY course_id
    ->     ) AS counts
    -> );
+-----------+------------------+
| course_id | enrollment_count |
+-----------+------------------+
|         2 |                2 |
|         3 |                2 |
|         5 |                2 |
|         6 |                2 |
|         8 |                2 |
+-----------+------------------+
5 rows in set (0.00 sec)
```

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



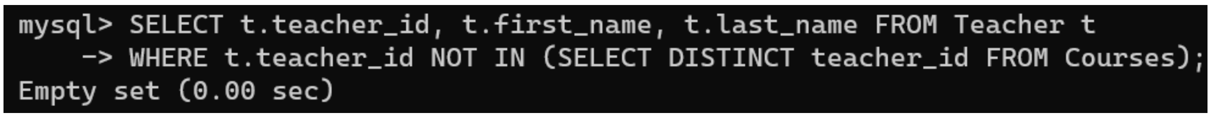
```
mysql> SELECT t.teacher_id, CONCAT(t.first_name,' ', t.last_name) AS Name_, SUM(P.amount) AS total_payments FROM Teacher t
    -> JOIN Courses C ON t.teacher_id = C.teacher_id
    -> LEFT JOIN Enrollments E ON C.course_id = E.course_id
    -> LEFT JOIN Payments P ON E.student_id = P.student_id
    -> GROUP BY t.teacher_id, t.first_name, t.last_name;
+------------+-----------------+----------------+
| teacher_id | Name_           | total_payments |
+------------+-----------------+----------------+
|        101 | Surya Naidu     |       14500.00 |
|        102 | Priyanka Reddy  |       31500.00 |
|        103 | Rajendra Varma  |       46000.00 |
|        104 | Meenakshi Kumar |       17500.00 |
|        105 | Venkatesh Rao   |       27500.00 |
|        106 | Divya Singh     |       15000.00 |
|        107 | Ravi Mehra      |       28000.00 |
|        108 | Anusha Yadav    |       31500.00 |
|        109 | Krishna Verma   |       18000.00 |
|        110 | Aruna Kumar     |       22000.00 |
|        111 | Srinivas Rajput |       11000.00 |
|        112 | Radha Shukla    |       10000.00 |
|        113 | Prakash Gandhi  |       16000.00 |
|        114 | Vijaya Rawat    |       12500.00 |
|        115 | Anand Malhotra  |       20500.00 |
+------------+-----------------+----------------+
15 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.



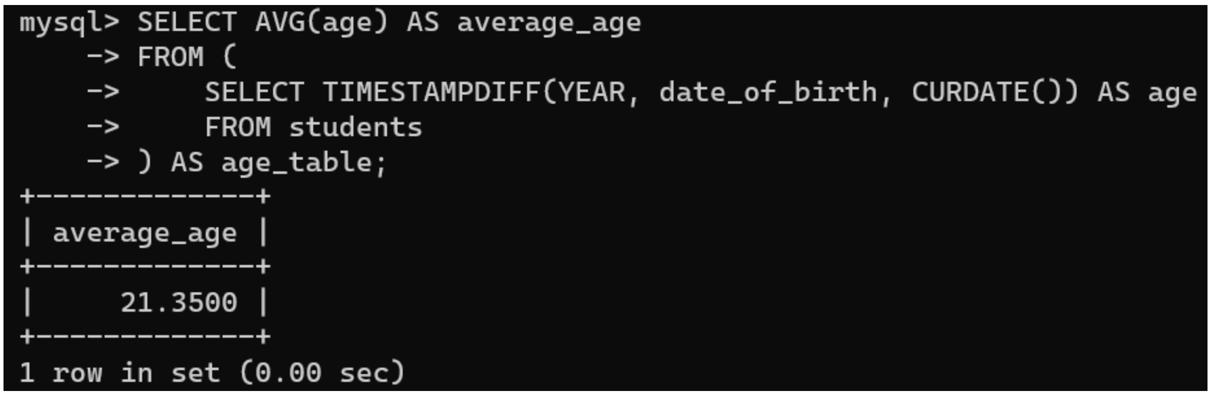
```
mysql> SELECT s.student_id FROM Students s
    -> WHERE NOT EXISTS ( SELECT 1 FROM Courses c,Enrollments e WHERE e.student_id = s.student_id AND e.course_id = c.course_id);
Empty set (0.00 sec)
```

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



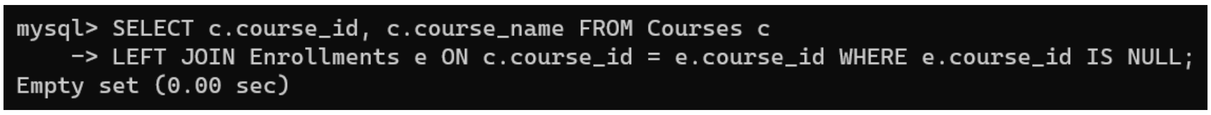
```
mysql> SELECT t.teacher_id, t.first_name, t.last_name FROM Teacher t
    -> WHERE t.teacher_id NOT IN (SELECT DISTINCT teacher_id FROM Courses);
Empty set (0.00 sec)
```

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



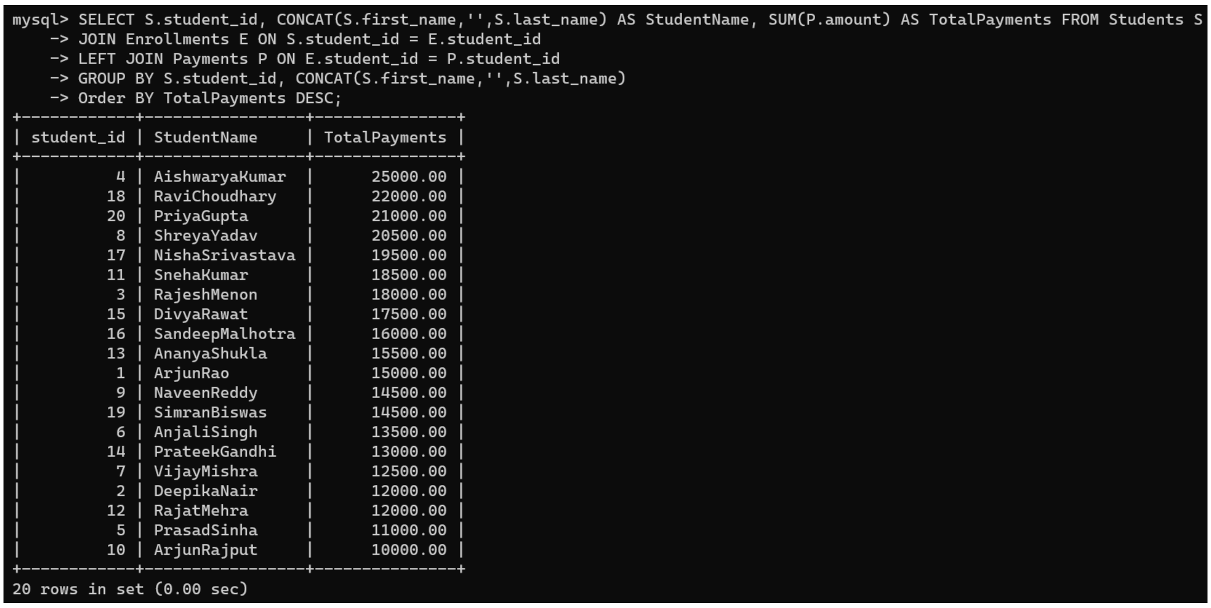
```
mysql> SELECT AVG(age) AS average_age
    -> FROM (
    ->     SELECT TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age
    ->     FROM students
    -> ) AS age_table;
+-------------+
| average_age |
+-------------+
|     21.3500 |
+-------------+
1 row in set (0.00 sec)
```

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



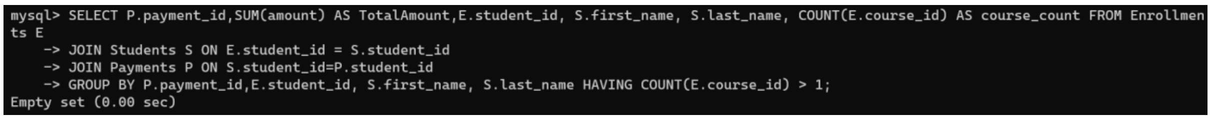
```
mysql> SELECT c.course_id, c.course_name FROM Courses c
    -> LEFT JOIN Enrollments e ON c.course_id = e.course_id WHERE e.course_id IS NULL;
Empty set (0.00 sec)
```

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



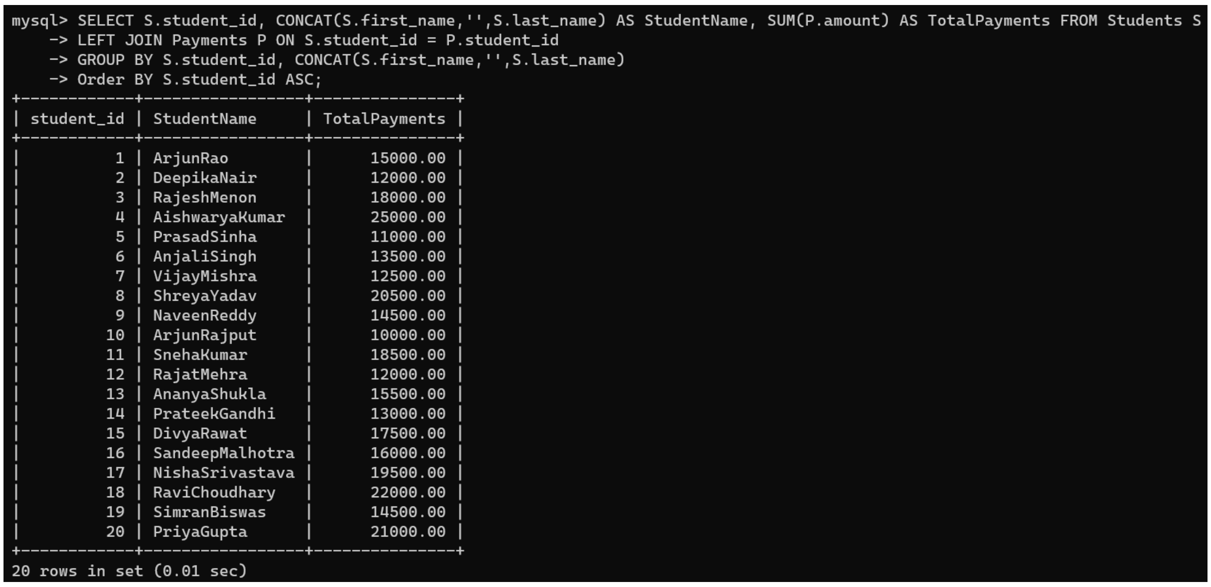
```
mysql> SELECT S.student_id, CONCAT(S.first_name,'',S.last_name) AS StudentName, SUM(P.amount) AS TotalPayments FROM Students S
    -> JOIN Enrollments E ON S.student_id = E.student_id
    -> LEFT JOIN Payments P ON E.student_id = P.student_id
    -> GROUP BY S.student_id, CONCAT(S.first_name,'',S.last_name)
    -> Order BY TotalPayments DESC;
+------------+-----------------+---------------+
| student_id | StudentName     | TotalPayments |
+------------+-----------------+---------------+
|          4 | AishwaryaKumar  |      25000.00 |
|         18 | RaviChoudhary   |      22000.00 |
|         20 | PriyaGupta      |      21000.00 |
|          8 | ShreyaYadav     |      20500.00 |
|         17 | NishaSrivastava |      19500.00 |
|         11 | SnehaKumar      |      18500.00 |
|          3 | RajeshMenon     |      18000.00 |
|         15 | DivyaRawat      |      17500.00 |
|         16 | SandeepMalhotra |      16000.00 |
|         13 | AnanyaShukla    |      15500.00 |
|          1 | ArjunRao        |      15000.00 |
|          9 | NaveenReddy     |      14500.00 |
|         19 | SimranBiswas    |      14500.00 |
|          6 | AnjaliSingh     |      13500.00 |
|         14 | PrateekGandhi   |      13000.00 |
|          7 | VijayMishra     |      12500.00 |
|          2 | DeepikaNair     |      12000.00 |
|         12 | RajatMehra      |      12000.00 |
|          5 | PrasadSinha     |      11000.00 |
|         10 | ArjunRajput     |      10000.00 |
+------------+-----------------+---------------+
20 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.



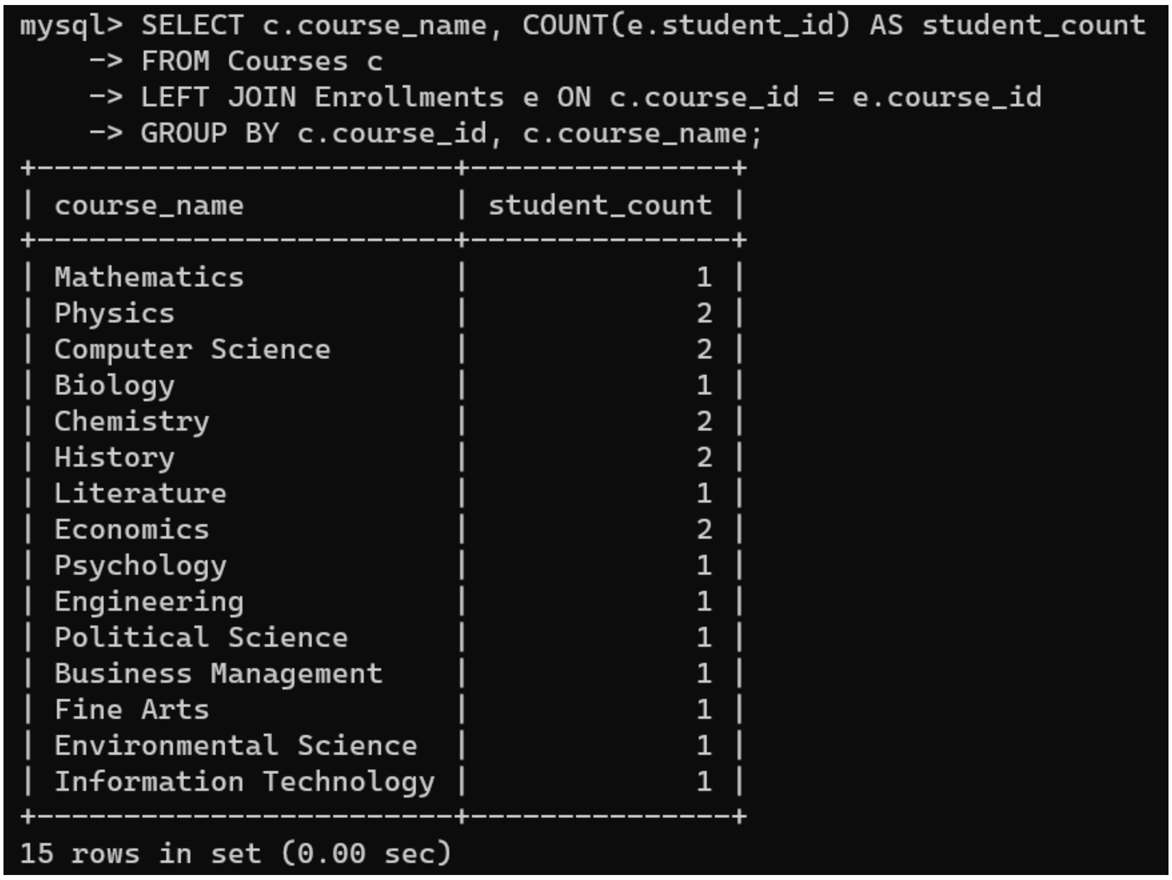
```
mysql> SELECT P.payment_id,SUM(amount) AS TotalAmount,E.student_id, S.first_name, S.last_name, COUNT(E.course_id) AS course_count FROM Enrollmen
ts E
    -> JOIN Students S ON E.student_id = S.student_id
    -> JOIN Payments P ON S.student_id=P.student_id
    -> GROUP BY P.payment_id,E.student_id, S.first_name, S.last_name HAVING COUNT(E.course_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.



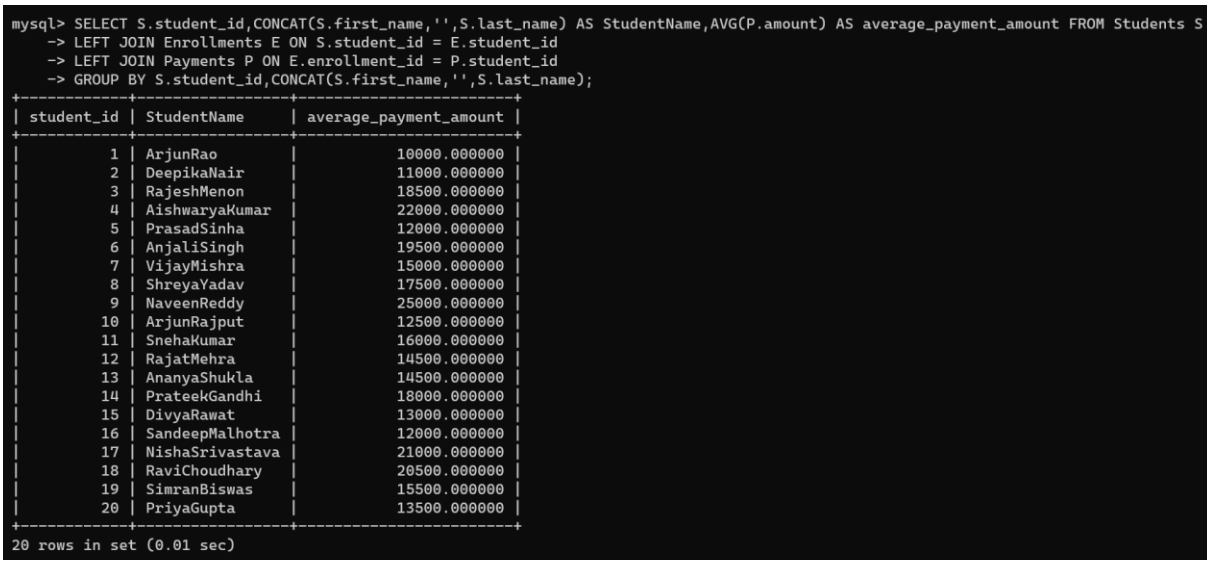
```
mysql> SELECT S.student_id, CONCAT(S.first_name,'',S.last_name) AS StudentName, SUM(P.amount) AS TotalPayments FROM Students S
    -> LEFT JOIN Payments P ON S.student_id = P.student_id
    -> GROUP BY S.student_id, CONCAT(S.first_name,'',S.last_name)
    -> Order BY S.student_id ASC;
+------------+-----------------+---------------+
| student_id | StudentName     | TotalPayments |
+------------+-----------------+---------------+
|          1 | ArjunRao        |      15000.00 |
|          2 | DeepikaNair     |      12000.00 |
|          3 | RajeshMenon     |      18000.00 |
|          4 | AishwaryaKumar  |      25000.00 |
|          5 | PrasadSinha     |      11000.00 |
|          6 | AnjaliSingh     |      13500.00 |
|          7 | VijayMishra     |      12500.00 |
|          8 | ShreyaYadav     |      20500.00 |
|          9 | NaveenReddy     |      14500.00 |
|         10 | ArjunRajput     |      10000.00 |
|         11 | SnehaKumar      |      18500.00 |
|         12 | RajatMehra      |      12000.00 |
|         13 | AnanyaShukla    |      15500.00 |
|         14 | PrateekGandhi   |      13000.00 |
|         15 | DivyaRawat      |      17500.00 |
|         16 | SandeepMalhotra |      16000.00 |
|         17 | NishaSrivastava |      19500.00 |
|         18 | RaviChoudhary   |      22000.00 |
|         19 | SimranBiswas    |      14500.00 |
|         20 | PriyaGupta      |      21000.00 |
+------------+-----------------+---------------+
20 rows in set (0.01 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(e.student_id) AS student_count
    -> FROM Courses c
    -> LEFT JOIN Enrollments e ON c.course_id = e.course_id
    -> GROUP BY c.course_id, c.course_name;
+------------------------+---------------+
| course_name            | student_count |
+------------------------+---------------+
| Mathematics            |             1 |
| Physics                |             2 |
| Computer Science       |             2 |
| Biology                |             1 |
| Chemistry              |             2 |
| History                |             2 |
| Literature             |             1 |
| Economics              |             2 |
| Psychology             |             1 |
| Engineering            |             1 |
| Political Science      |             1 |
| Business Management    |             1 |
| Fine Arts              |             1 |
| Environmental Science  |             1 |
| Information Technology |             1 |
+------------------------+---------------+
15 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,CONCAT(S.first_name,'',S.last_name) AS StudentName,AVG(P.amount) AS average_payment_amount FROM Students S
    -> LEFT JOIN Enrollments E ON S.student_id = E.student_id
    -> LEFT JOIN Payments P ON E.enrollment_id = P.student_id
    -> GROUP BY S.student_id,CONCAT(S.first_name,'',S.last_name);
+------------+-----------------+------------------------+
| student_id | StudentName     | average_payment_amount |
+------------+-----------------+------------------------+
|          1 | ArjunRao        |           10000.000000 |
|          2 | DeepikaNair     |           11000.000000 |
|          3 | RajeshMenon     |           18500.000000 |
|          4 | AishwaryaKumar  |           22000.000000 |
|          5 | PrasadSinha     |           12000.000000 |
|          6 | AnjaliSingh     |           19500.000000 |
|          7 | VijayMishra     |           15000.000000 |
|          8 | ShreyaYadav     |           17500.000000 |
|          9 | NaveenReddy     |           25000.000000 |
|         10 | ArjunRajput     |           12500.000000 |
|         11 | SnehaKumar      |           16000.000000 |
|         12 | RajatMehra      |           14500.000000 |
|         13 | AnanyaShukla    |           14500.000000 |
|         14 | PrateekGandhi   |           18000.000000 |
|         15 | DivyaRawat      |           13000.000000 |
|         16 | SandeepMalhotra |           12000.000000 |
|         17 | NishaSrivastava |           21000.000000 |
|         18 | RaviChoudhary   |           20500.000000 |
|         19 | SimranBiswas    |           15500.000000 |
|         20 | PriyaGupta      |           13500.000000 |
+------------+-----------------+------------------------+
20 rows in set (0.01 sec)
```