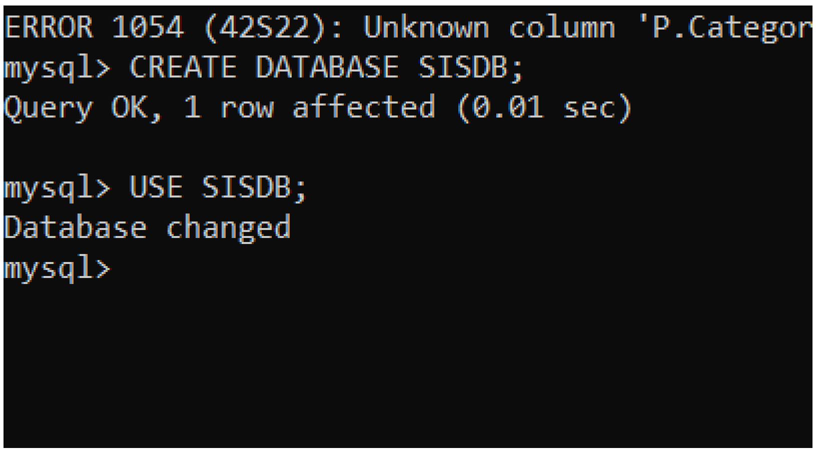
ASSIGNMENT 2

NAME : - SUSHANT KUMAR SINGH
EMAIL :- ssushant886@gmail.com

Task 1. Database Design:

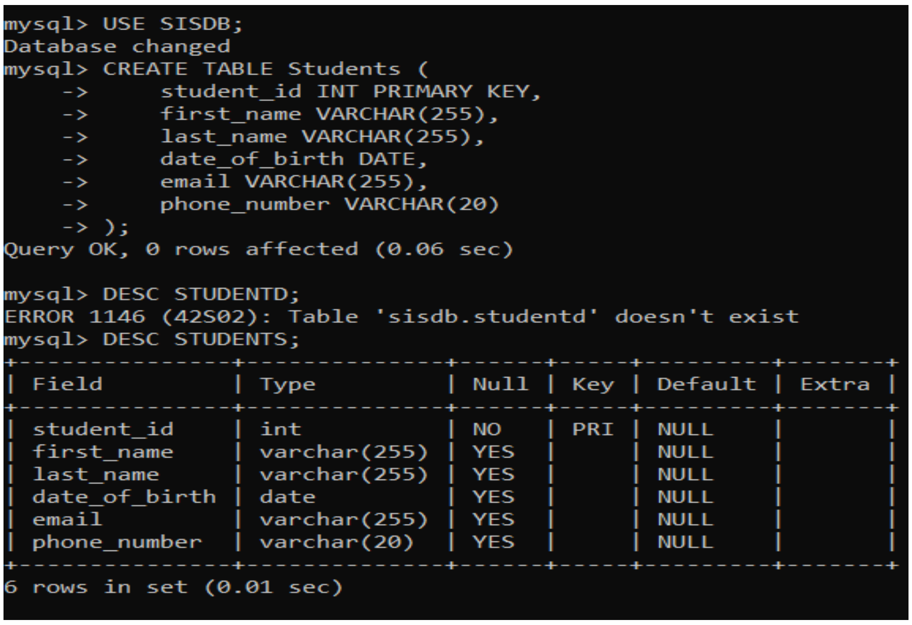
1) **Create the database named "SISDB"**



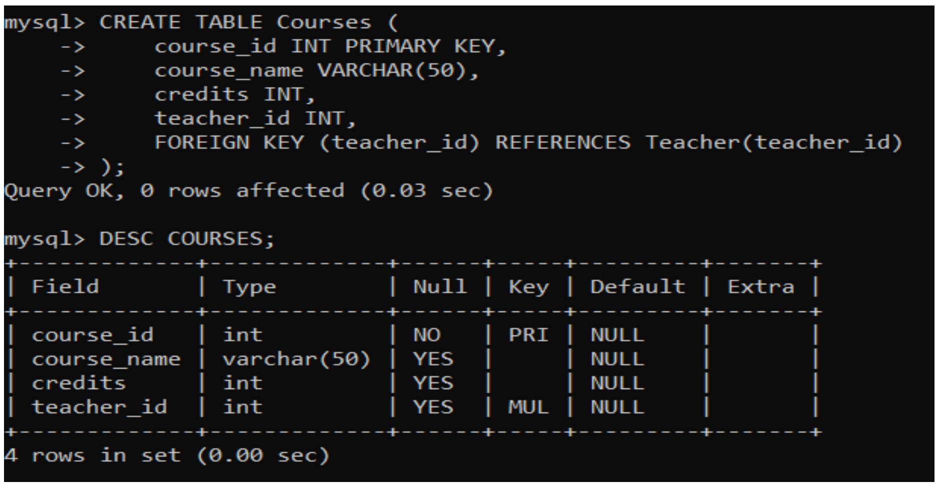
```
ERROR 1054 (42S22): Unknown column 'P.Categor
mysql> CREATE DATABASE SISDB;
Query OK, 1 row affected (0.01 sec)

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

1) STUDENTS

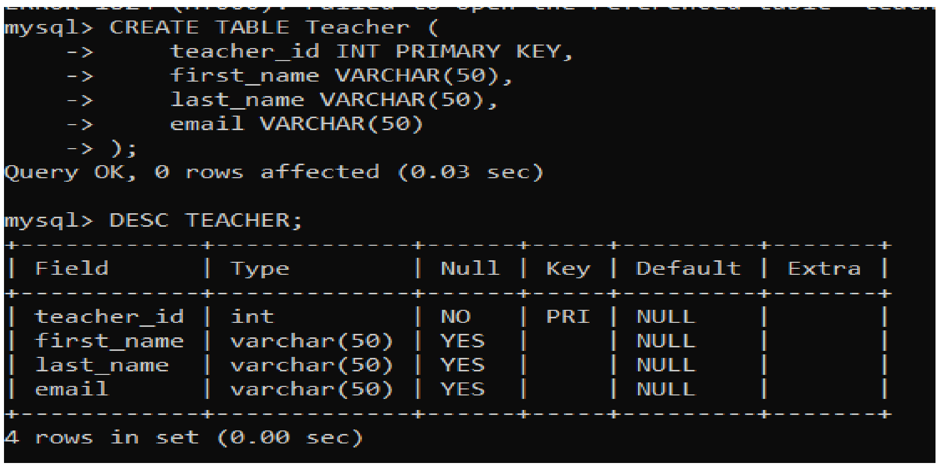


```
mysql> USE SISDB;
Database changed
mysql> CREATE TABLE Students (
    ->     student_id INT PRIMARY KEY,
    ->     first_name VARCHAR(255),
    ->     last_name VARCHAR(255),
    ->     date_of_birth DATE,
    ->     email VARCHAR(255),
    ->     phone_number VARCHAR(20)
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> DESC STUDENTD;
ERROR 1146 (42S02): Table 'sisdb.studentd' doesn't exist
mysql> DESC STUDENTS;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| student_id    | int          | NO   | PRI | NULL    |       |
| first_name    | varchar(255) | YES  |     | NULL    |       |
| last_name     | varchar(255) | YES  |     | NULL    |       |
| date_of_birth | date         | YES  |     | NULL    |       |
| email         | varchar(255) | YES  |     | NULL    |       |
| phone_number  | varchar(20)  | YES  |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```



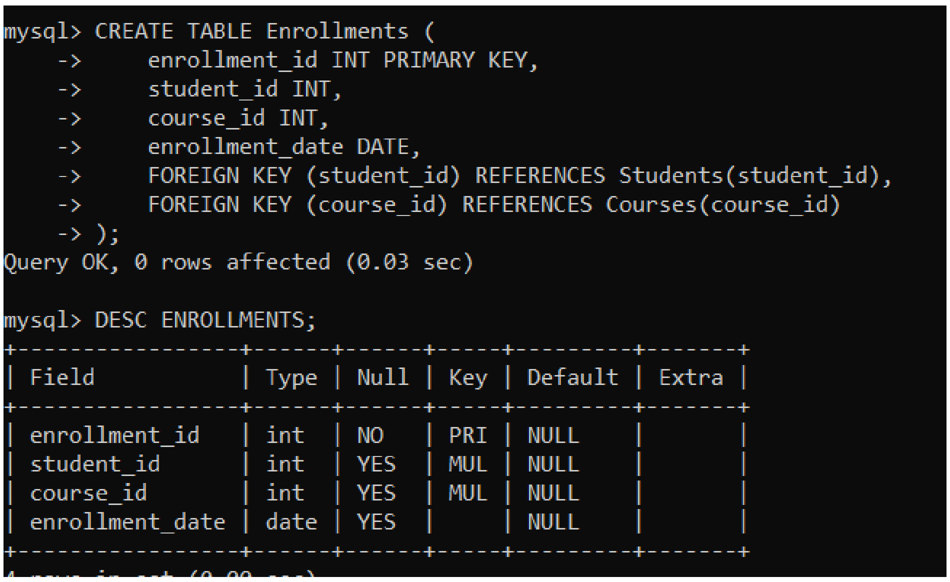
```
mysql> CREATE TABLE Courses (
    ->     course_id INT PRIMARY KEY,
    ->     course_name VARCHAR(50),
    ->     credits INT,
    ->     teacher_id INT,
    ->     FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> DESC COURSES;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| course_id   | int         | NO   | PRI | NULL    |       |
| course_name | varchar(50) | YES  |     | NULL    |       |
| credits     | int         | YES  |     | NULL    |       |
| teacher_id  | int         | YES  | MUL | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

3) TEACHER



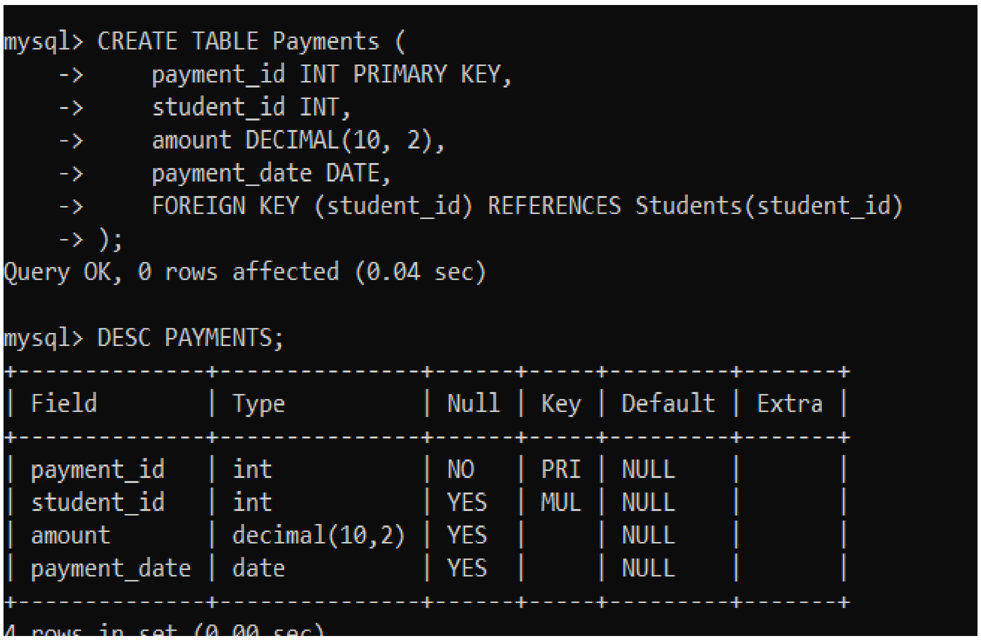
```
ERROR 1824 (HY000): Failed to open the referenced table 'teach
mysql> CREATE TABLE Teacher (
    ->     teacher_id INT PRIMARY KEY,
    ->     first_name VARCHAR(50),
    ->     last_name VARCHAR(50),
    ->     email VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> DESC TEACHER;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| teacher_id | int         | NO   | PRI | NULL    |       |
| first_name | varchar(50) | YES  |     | NULL    |       |
| last_name  | varchar(50) | YES  |     | NULL    |       |
| email      | varchar(50) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

4) ENROLLMENTS



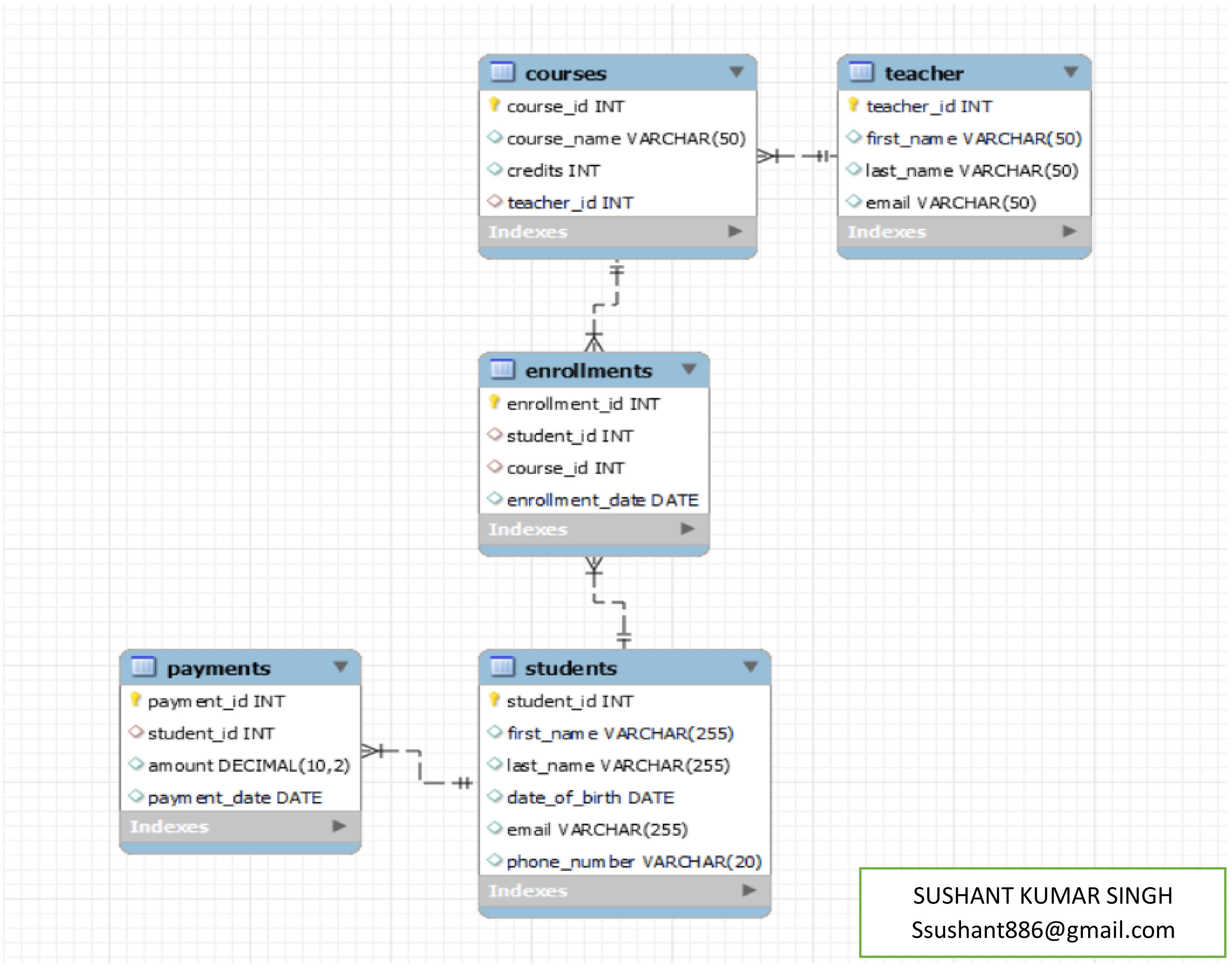
```
mysql> CREATE TABLE Enrollments (
    ->     enrollment_id INT PRIMARY KEY,
    ->     student_id INT,
    ->     course_id INT,
    ->     enrollment_date DATE,
    ->     FOREIGN KEY (student_id) REFERENCES Students(student_id),
    ->     FOREIGN KEY (course_id) REFERENCES Courses(course_id)
    -> );
Query OK, 0 rows affected (0.03 sec)

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)
```

5) 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 (0.04 sec)

mysql> DESC PAYMENTS;
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| payment_id   | int           | NO   | PRI | NULL    |       |
| student_id   | int           | YES  | MUL | NULL    |       |
| amount       | decimal(10,2) | YES  |     | NULL    |       |
| payment_date | date          | YES  |     | NULL    |       |
+--------------+---------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

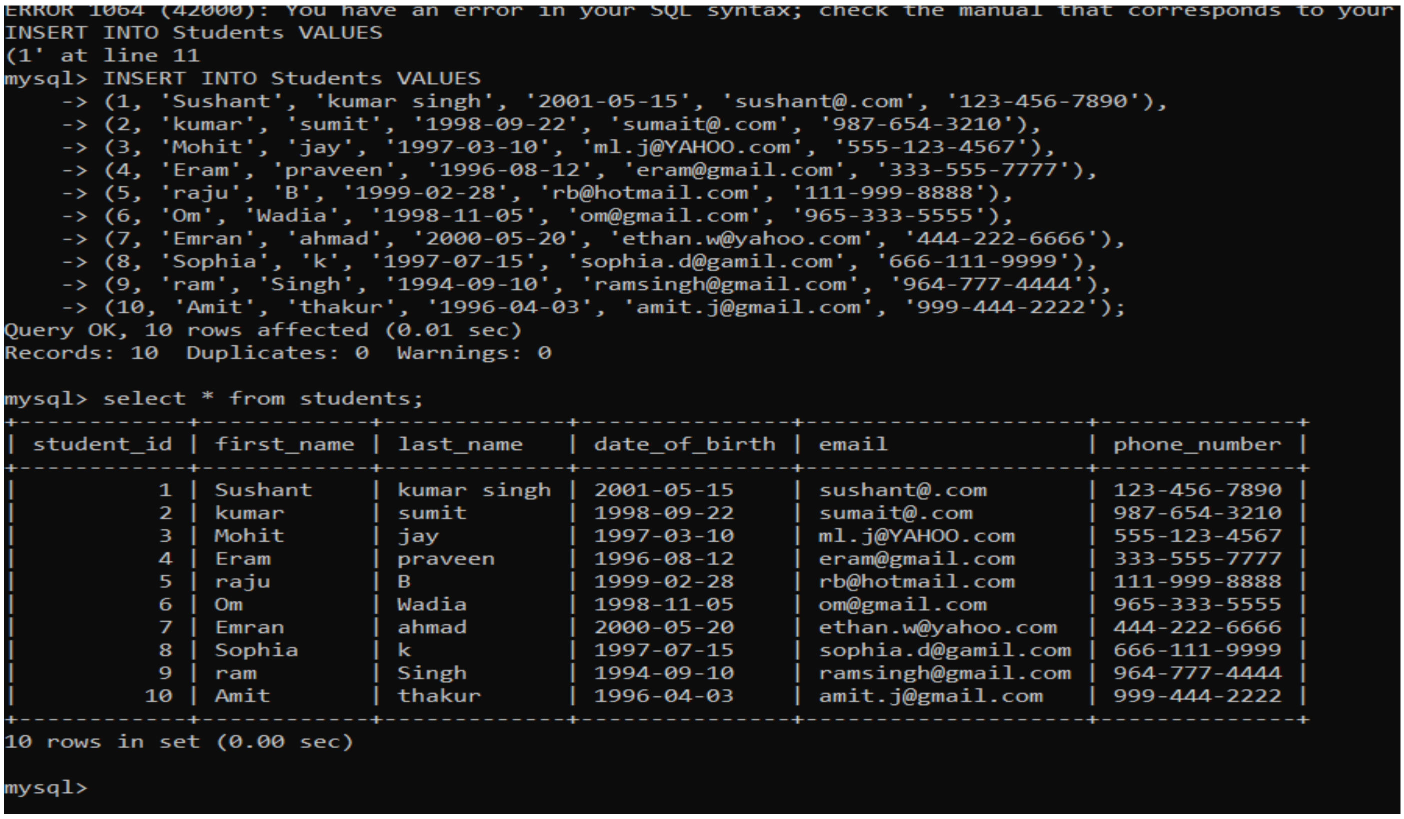
3. Create an ERD (Entity Relationship Diagram) for the database.



SUSHANT KUMAR SINGH
Ssushant886@gmail.com

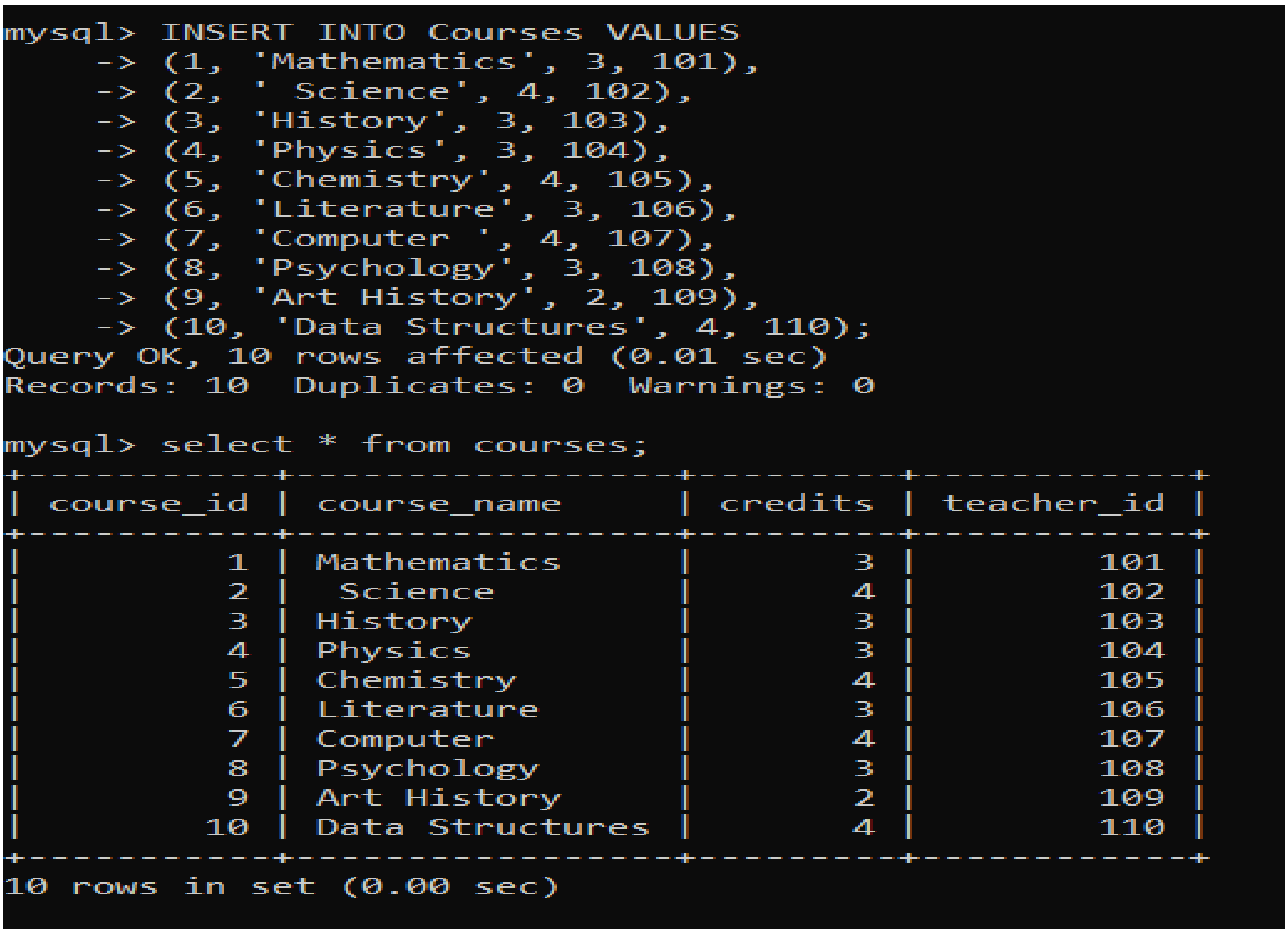
5. Insert at least 10 sample records into each of the following tables. i. Students ii. Courses iii. Enrollments iv. Teacher v. Payments

1) STUDENTS



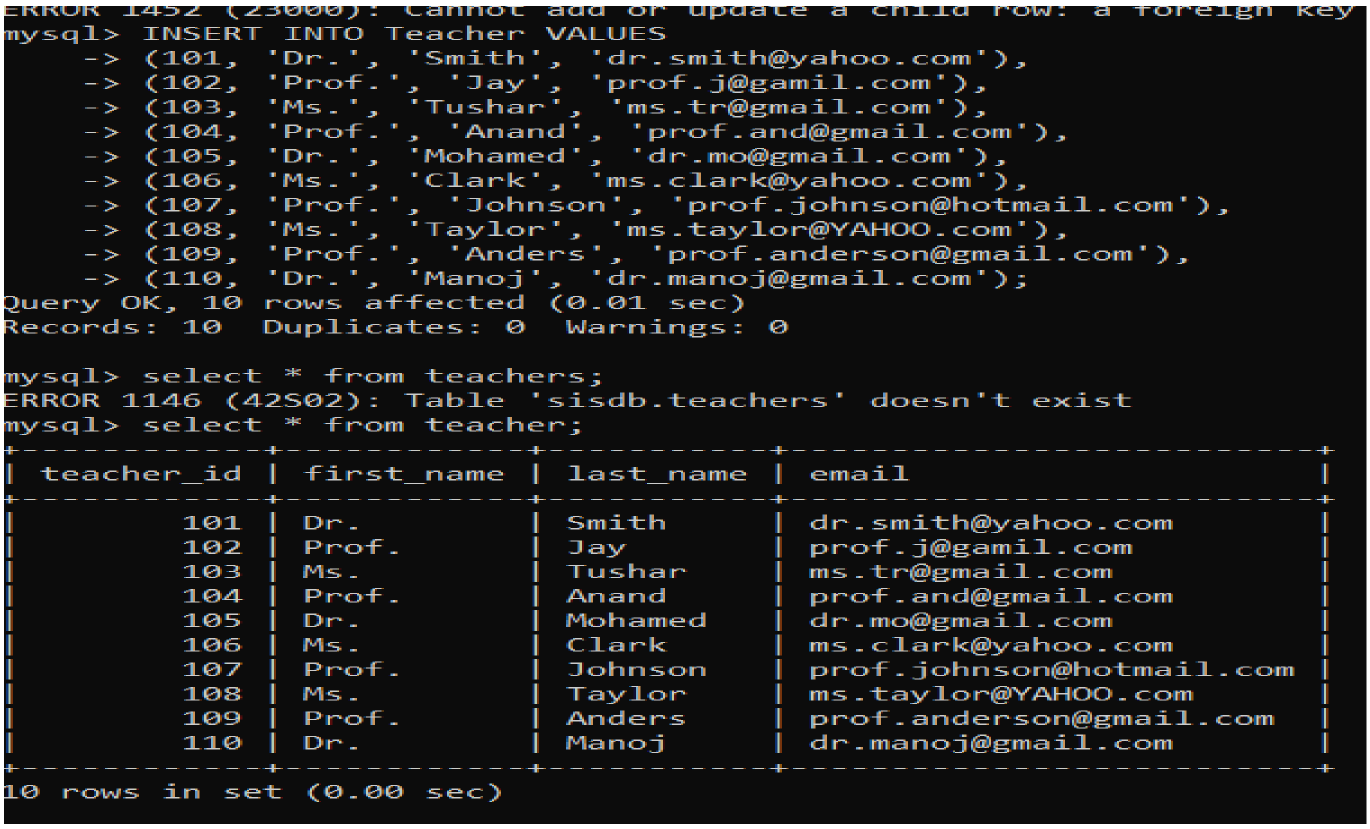
```
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your
INSERT INTO Students VALUES
(1' at line 11
mysql> INSERT INTO Students VALUES
    -> (1, 'Sushant', 'kumar singh', '2001-05-15', 'sushant@.com', '123-456-7890'),
    -> (2, 'kumar', 'sumit', '1998-09-22', 'sumait@.com', '987-654-3210'),
    -> (3, 'Mohit', 'jay', '1997-03-10', 'ml.j@YAHOO.com', '555-123-4567'),
    -> (4, 'Eram', 'praveen', '1996-08-12', 'eram@gmail.com', '333-555-7777'),
    -> (5, 'raju', 'B', '1999-02-28', 'rb@hotmail.com', '111-999-8888'),
    -> (6, 'Om', 'Wadia', '1998-11-05', 'om@gmail.com', '965-333-5555'),
    -> (7, 'Emran', 'ahmad', '2000-05-20', 'ethan.w@yahoo.com', '444-222-6666'),
    -> (8, 'Sophia', 'k', '1997-07-15', 'sophia.d@gamil.com', '666-111-9999'),
    -> (9, 'ram', 'Singh', '1994-09-10', 'ramsingh@gmail.com', '964-777-4444'),
    -> (10, 'Amit', 'thakur', '1996-04-03', 'amit.j@gmail.com', '999-444-2222');
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 | Sushant    | kumar singh | 2001-05-15    | sushant@.com       | 123-456-7890 |
|          2 | kumar      | sumit       | 1998-09-22    | sumait@.com        | 987-654-3210 |
|          3 | Mohit      | jay         | 1997-03-10    | ml.j@YAHOO.com     | 555-123-4567 |
|          4 | Eram       | praveen     | 1996-08-12    | eram@gmail.com     | 333-555-7777 |
|          5 | raju       | B           | 1999-02-28    | rb@hotmail.com     | 111-999-8888 |
|          6 | Om         | Wadia       | 1998-11-05    | om@gmail.com       | 965-333-5555 |
|          7 | Emran      | ahmad       | 2000-05-20    | ethan.w@yahoo.com  | 444-222-6666 |
|          8 | Sophia     | k           | 1997-07-15    | sophia.d@gamil.com | 666-111-9999 |
|          9 | ram        | Singh       | 1994-09-10    | ramsingh@gmail.com | 964-777-4444 |
|         10 | Amit       | thakur      | 1996-04-03    | amit.j@gmail.com   | 999-444-2222 |
+------------+------------+-------------+---------------+--------------------+--------------+
10 rows in set (0.00 sec)

mysql>
```

2) COURSES



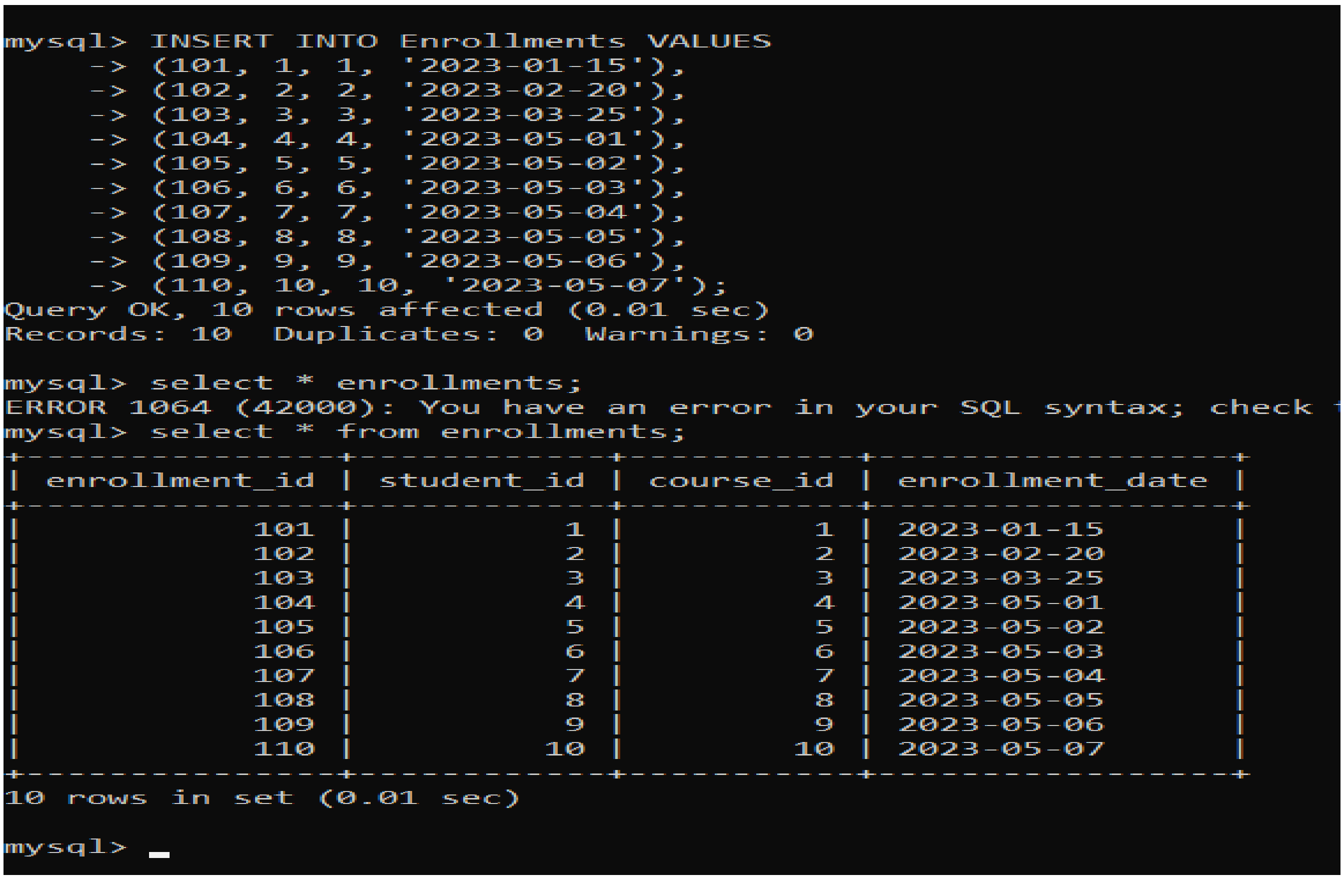
```
mysql> INSERT INTO Courses VALUES
    -> (1, 'Mathematics', 3, 101),
    -> (2, ' Science', 4, 102),
    -> (3, 'History', 3, 103),
    -> (4, 'Physics', 3, 104),
    -> (5, 'Chemistry', 4, 105),
    -> (6, 'Literature', 3, 106),
    -> (7, 'Computer ', 4, 107),
    -> (8, 'Psychology', 3, 108),
    -> (9, 'Art History', 2, 109),
    -> (10, 'Data Structures', 4, 110);
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 |
+-----------+-----------------+---------+------------+
|         1 | Mathematics     |       3 |        101 |
|         2 |  Science        |       4 |        102 |
|         3 | History         |       3 |        103 |
|         4 | Physics         |       3 |        104 |
|         5 | Chemistry       |       4 |        105 |
|         6 | Literature      |       3 |        106 |
|         7 | Computer        |       4 |        107 |
|         8 | Psychology      |       3 |        108 |
|         9 | Art History     |       2 |        109 |
|        10 | Data Structures |       4 |        110 |
+-----------+-----------------+---------+------------+
10 rows in set (0.00 sec)
```

3) TEACHER



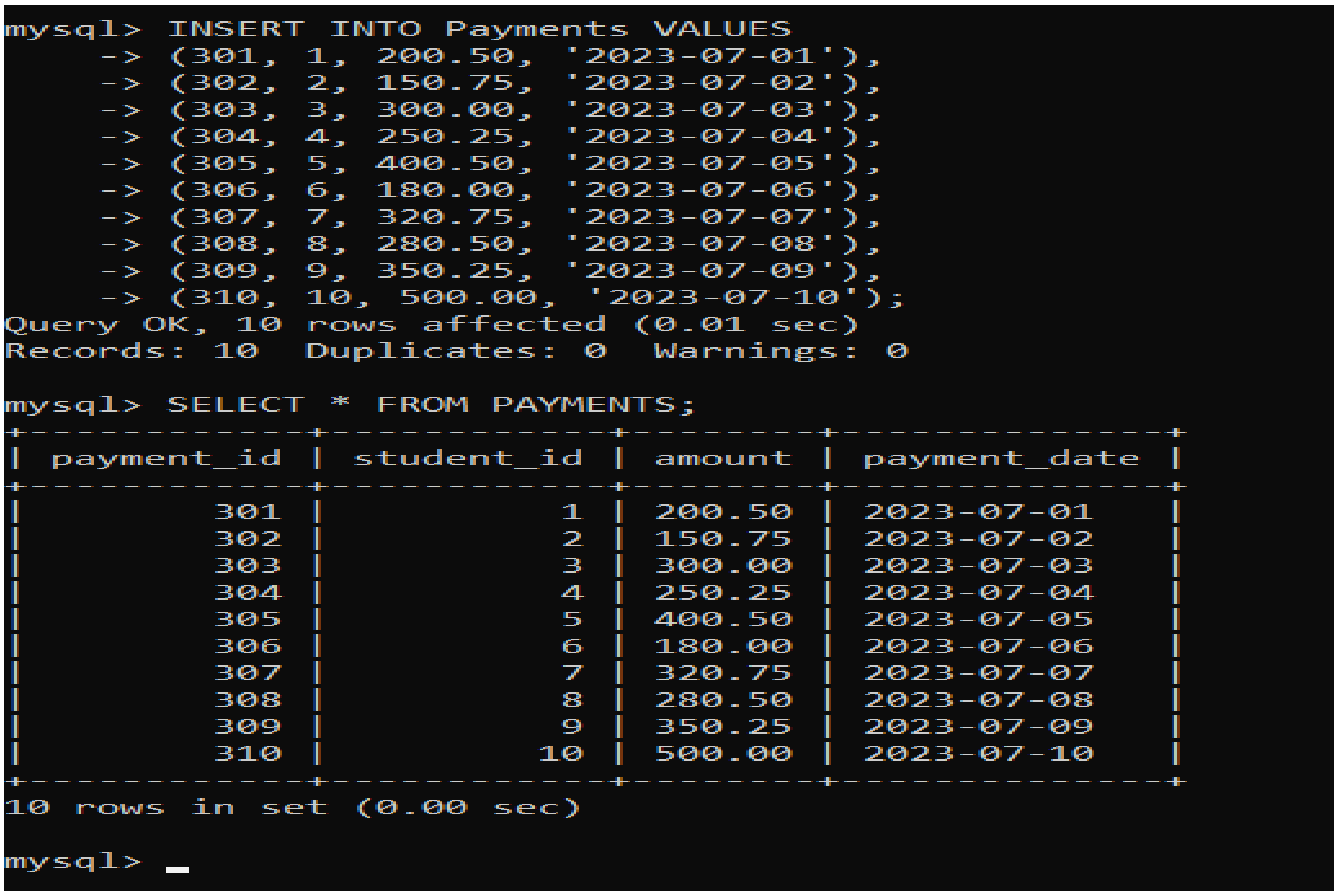
```
ERROR 1452 (23000): Cannot add or update a child row: a foreign key
mysql> INSERT INTO Teacher VALUES
    -> (101, 'Dr.', 'Smith', 'dr.smith@yahoo.com'),
    -> (102, 'Prof.', 'Jay', 'prof.j@gamil.com'),
    -> (103, 'Ms.', 'Tushar', 'ms.tr@gmail.com'),
    -> (104, 'Prof.', 'Anand', 'prof.and@gmail.com'),
    -> (105, 'Dr.', 'Mohamed', 'dr.mo@gmail.com'),
    -> (106, 'Ms.', 'Clark', 'ms.clark@yahoo.com'),
    -> (107, 'Prof.', 'Johnson', 'prof.johnson@hotmail.com'),
    -> (108, 'Ms.', 'Taylor', 'ms.taylor@YAHOO.com'),
    -> (109, 'Prof.', 'Anders', 'prof.anderson@gmail.com'),
    -> (110, 'Dr.', 'Manoj', 'dr.manoj@gmail.com');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from teachers;
ERROR 1146 (42S02): Table 'sisdb.teachers' doesn't exist
mysql> select * from teacher;
+------------+------------+-----------+--------------------------+
| teacher_id | first_name | last_name | email                    |
+------------+------------+-----------+--------------------------+
|        101 | Dr.        | Smith     | dr.smith@yahoo.com       |
|        102 | Prof.      | Jay       | prof.j@gamil.com         |
|        103 | Ms.        | Tushar    | ms.tr@gmail.com          |
|        104 | Prof.      | Anand     | prof.and@gmail.com       |
|        105 | Dr.        | Mohamed   | dr.mo@gmail.com          |
|        106 | Ms.        | Clark     | ms.clark@yahoo.com       |
|        107 | Prof.      | Johnson   | prof.johnson@hotmail.com |
|        108 | Ms.        | Taylor    | ms.taylor@YAHOO.com      |
|        109 | Prof.      | Anders    | prof.anderson@gmail.com  |
|        110 | Dr.        | Manoj     | dr.manoj@gmail.com       |
+------------+------------+-----------+--------------------------+
10 rows in set (0.00 sec)
```

4) ENROLLMENTS



```
mysql> INSERT INTO Enrollments VALUES
    -> (101, 1, 1, '2023-01-15'),
    -> (102, 2, 2, '2023-02-20'),
    -> (103, 3, 3, '2023-03-25'),
    -> (104, 4, 4, '2023-05-01'),
    -> (105, 5, 5, '2023-05-02'),
    -> (106, 6, 6, '2023-05-03'),
    -> (107, 7, 7, '2023-05-04'),
    -> (108, 8, 8, '2023-05-05'),
    -> (109, 9, 9, '2023-05-06'),
    -> (110, 10, 10, '2023-05-07');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * enrollments;
ERROR 1064 (42000): You have an error in your SQL syntax; check
mysql> select * from enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           101 |          1 |         1 | 2023-01-15      |
|           102 |          2 |         2 | 2023-02-20      |
|           103 |          3 |         3 | 2023-03-25      |
|           104 |          4 |         4 | 2023-05-01      |
|           105 |          5 |         5 | 2023-05-02      |
|           106 |          6 |         6 | 2023-05-03      |
|           107 |          7 |         7 | 2023-05-04      |
|           108 |          8 |         8 | 2023-05-05      |
|           109 |          9 |         9 | 2023-05-06      |
|           110 |         10 |        10 | 2023-05-07      |
+---------------+------------+-----------+-----------------+
10 rows in set (0.01 sec)

mysql> _
```

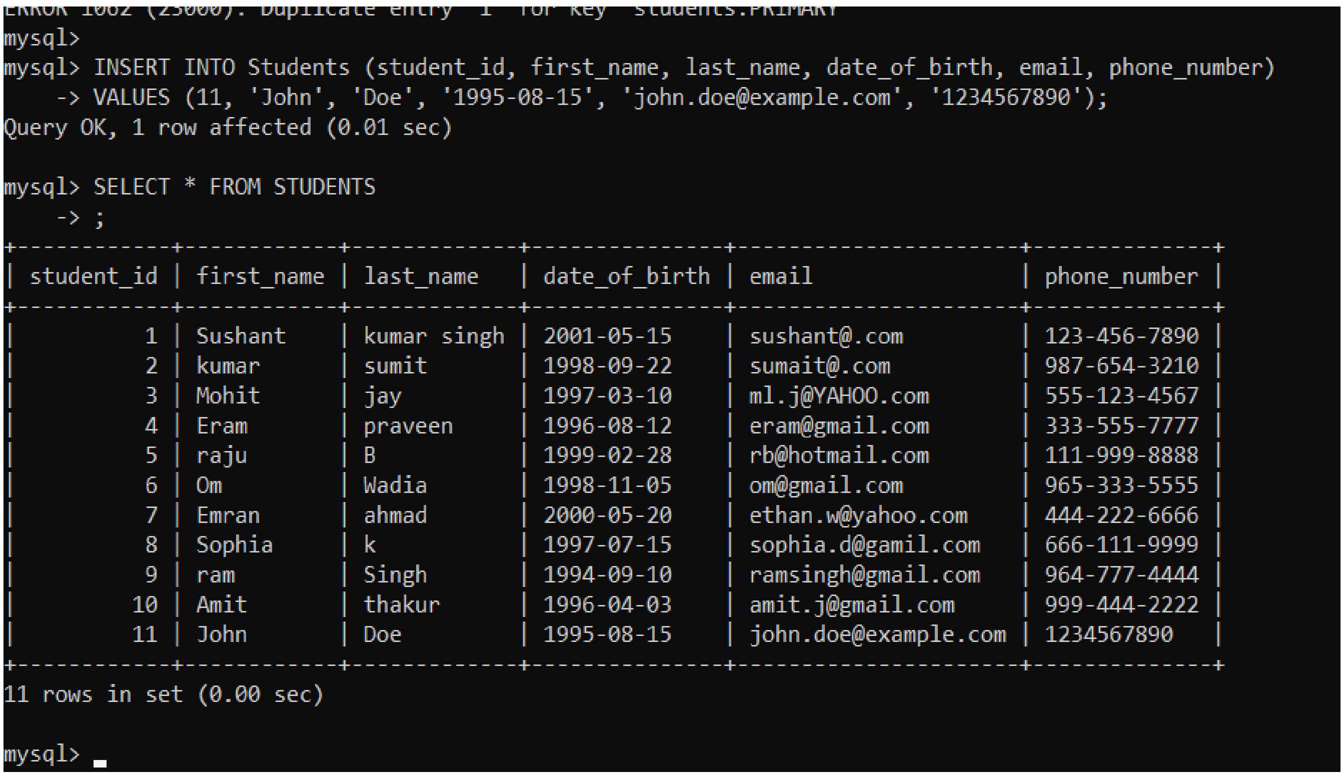
5) PAYMENTS



```
mysql> INSERT INTO Payments VALUES
    -> (301, 1, 200.50, '2023-07-01'),
    -> (302, 2, 150.75, '2023-07-02'),
    -> (303, 3, 300.00, '2023-07-03'),
    -> (304, 4, 250.25, '2023-07-04'),
    -> (305, 5, 400.50, '2023-07-05'),
    -> (306, 6, 180.00, '2023-07-06'),
    -> (307, 7, 320.75, '2023-07-07'),
    -> (308, 8, 280.50, '2023-07-08'),
    -> (309, 9, 350.25, '2023-07-09'),
    -> (310, 10, 500.00, '2023-07-10');
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 | 200.50 | 2023-07-01   |
|        302 |          2 | 150.75 | 2023-07-02   |
|        303 |          3 | 300.00 | 2023-07-03   |
|        304 |          4 | 250.25 | 2023-07-04   |
|        305 |          5 | 400.50 | 2023-07-05   |
|        306 |          6 | 180.00 | 2023-07-06   |
|        307 |          7 | 320.75 | 2023-07-07   |
|        308 |          8 | 280.50 | 2023-07-08   |
|        309 |          9 | 350.25 | 2023-07-09   |
|        310 |         10 | 500.00 | 2023-07-10   |
+------------+------------+--------+--------------+
10 rows in set (0.00 sec)

mysql> _
```

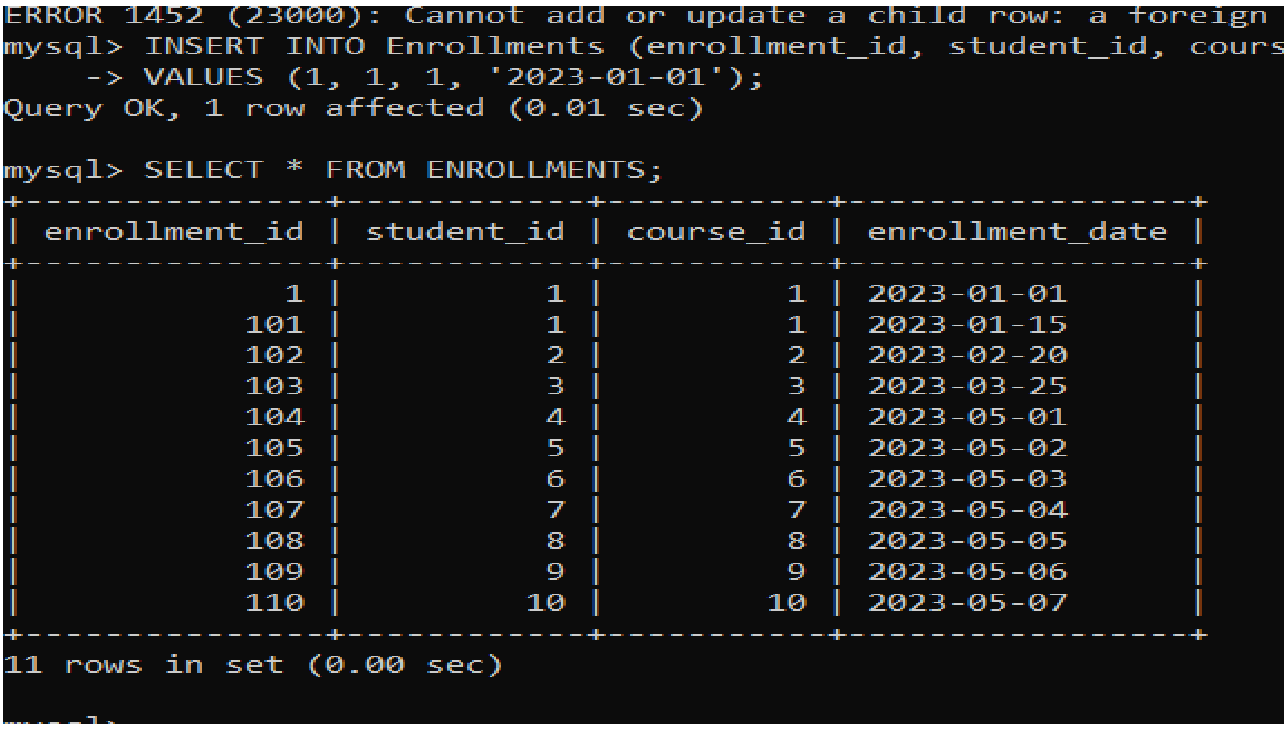
Tasks 2: Select, Where, Between, AND, LIKE:

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**



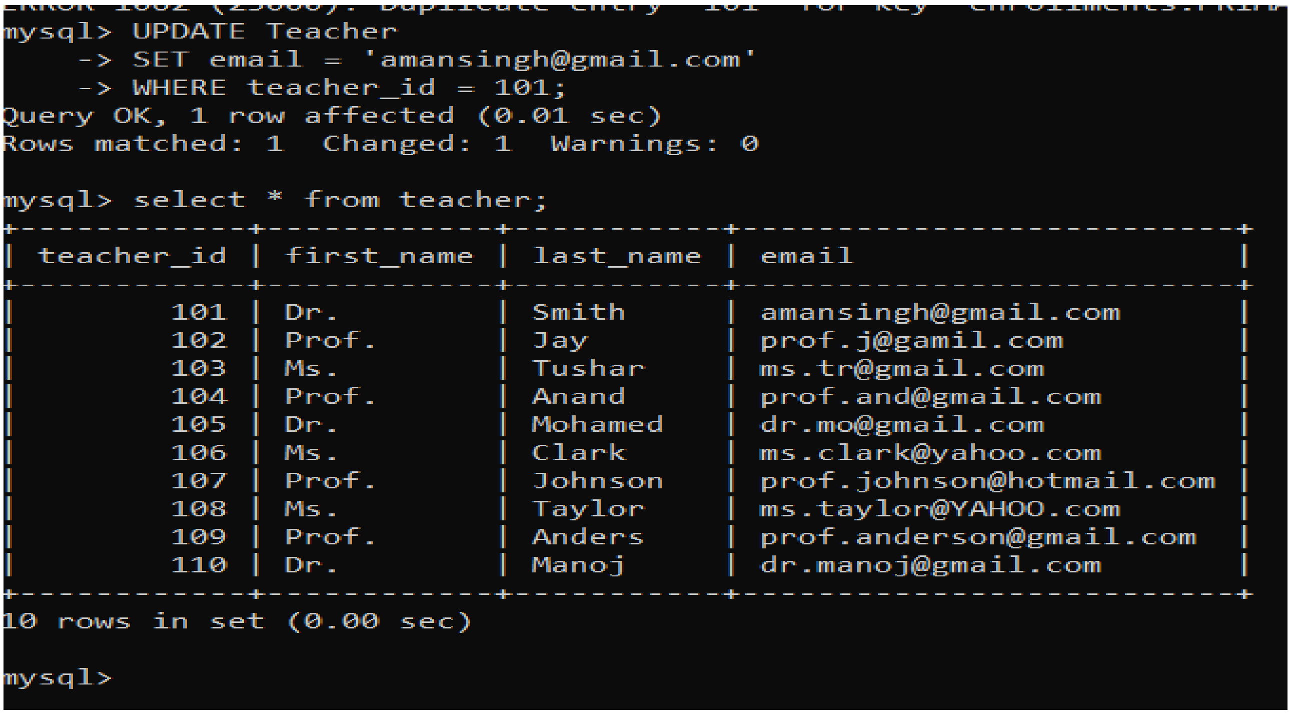
```
ERROR 1062 (23000): Duplicate entry '1' for key 'students.PRIMARY'
mysql>
mysql> INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number)
    -> VALUES (11, 'John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM STUDENTS
    -> ;
+------------+------------+------------+---------------+----------------------+--------------+
| student_id | first_name | last_name  | date_of_birth | email                | phone_number |
+------------+------------+------------+---------------+----------------------+--------------+
|          1 | Sushant    | kumar singh | 2001-05-15   | sushant@.com         | 123-456-7890 |
|          2 | kumar      | sumit      | 1998-09-22    | sumait@.com          | 987-654-3210 |
|          3 | Mohit      | jay        | 1997-03-10    | ml.j@YAHOO.com       | 555-123-4567 |
|          4 | Eram       | praveen    | 1996-08-12    | eram@gmail.com       | 333-555-7777 |
|          5 | raju       | B          | 1999-02-28    | rb@hotmail.com       | 111-999-8888 |
|          6 | Om         | Wadia      | 1998-11-05    | om@gmail.com         | 965-333-5555 |
|          7 | Emran      | ahmad      | 2000-05-20    | ethan.w@yahoo.com    | 444-222-6666 |
|          8 | Sophia     | k          | 1997-07-15    | sophia.d@gamil.com   | 666-111-9999 |
|          9 | ram        | Singh      | 1994-09-10    | ramsingh@gmail.com   | 964-777-4444 |
|         10 | Amit       | thakur     | 1996-04-03    | amit.j@gmail.com     | 999-444-2222 |
|         11 | John       | Doe        | 1995-08-15    | john.doe@example.com | 1234567890   |
+------------+------------+------------+---------------+----------------------+--------------+
11 rows in set (0.00 sec)

mysql> _
```

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.**



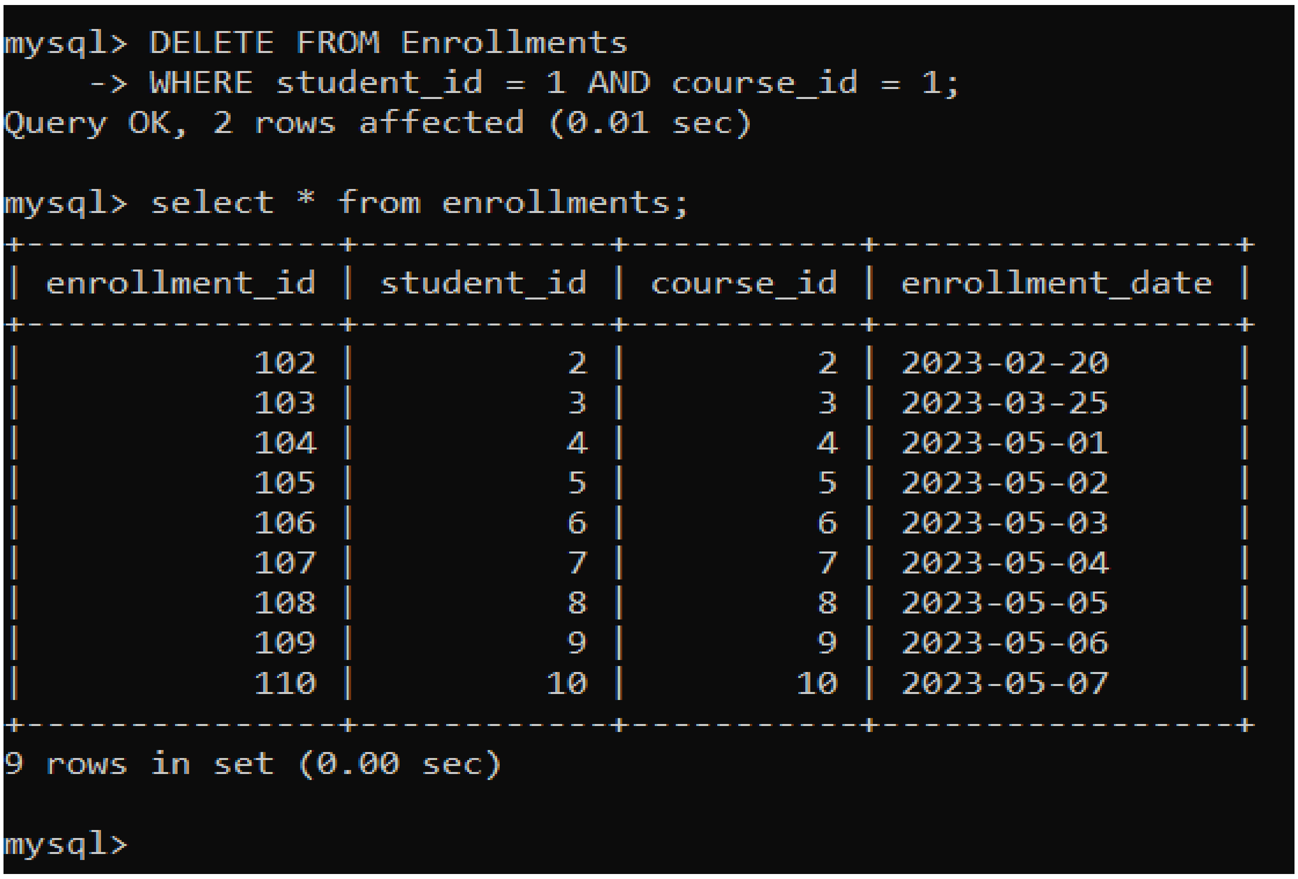
```
ERROR 1452 (23000): Cannot add or update a child row: a foreign
mysql> INSERT INTO Enrollments (enrollment_id, student_id, cours
    -> VALUES (1, 1, 1, '2023-01-01');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM ENROLLMENTS;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|             1 |          1 |         1 | 2023-01-01      |
|           101 |          1 |         1 | 2023-01-15      |
|           102 |          2 |         2 | 2023-02-20      |
|           103 |          3 |         3 | 2023-03-25      |
|           104 |          4 |         4 | 2023-05-01      |
|           105 |          5 |         5 | 2023-05-02      |
|           106 |          6 |         6 | 2023-05-03      |
|           107 |          7 |         7 | 2023-05-04      |
|           108 |          8 |         8 | 2023-05-05      |
|           109 |          9 |         9 | 2023-05-06      |
|           110 |         10 |        10 | 2023-05-07      |
+---------------+------------+-----------+-----------------+
11 rows in set (0.00 sec)

mysql>
```

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



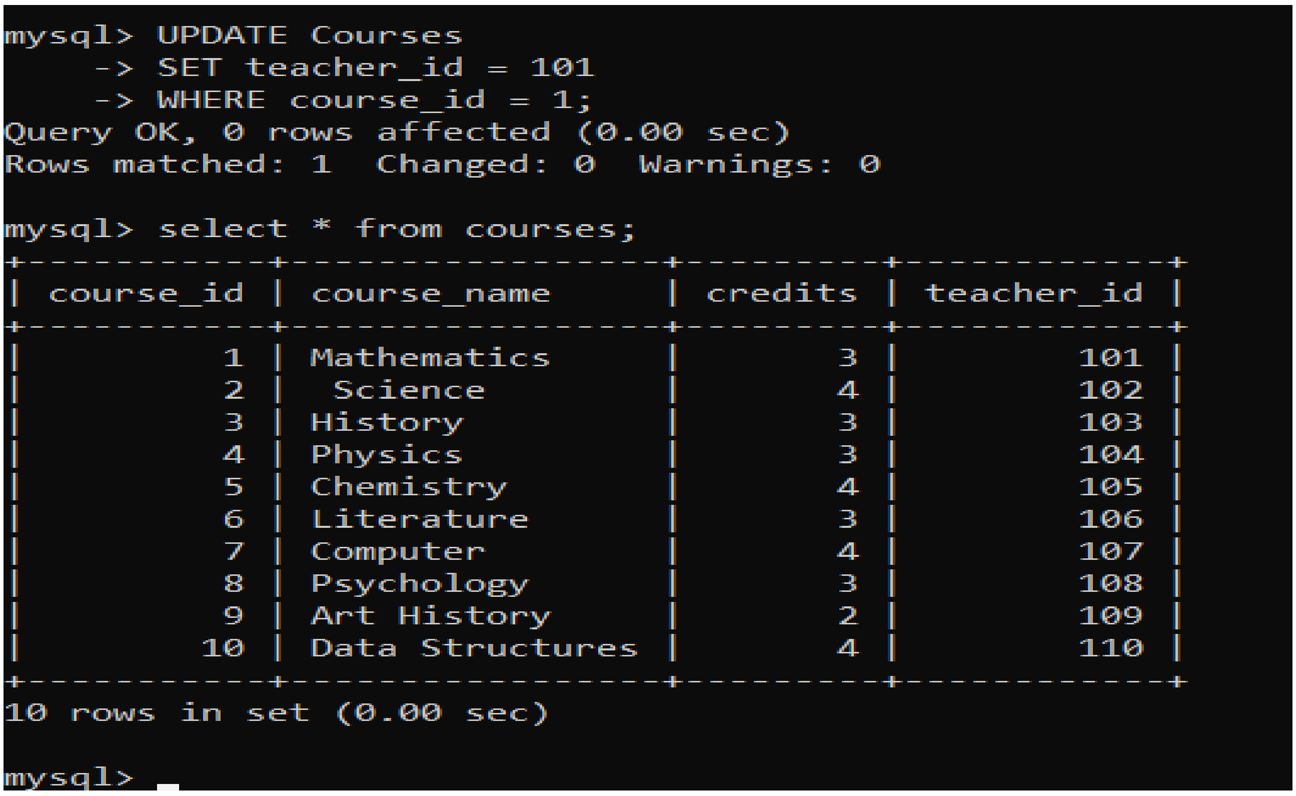
```
ERROR 1062 (23000): Duplicate entry '101' for key 'enrollments.PRIM
mysql> UPDATE Teacher
    -> SET email = 'amansingh@gmail.com'
    -> WHERE teacher_id = 101;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from teacher;
+------------+------------+-----------+---------------------------+
| teacher_id | first_name | last_name | email                     |
+------------+------------+-----------+---------------------------+
|        101 | Dr.        | Smith     | amansingh@gmail.com       |
|        102 | Prof.      | Jay       | prof.j@gamil.com          |
|        103 | Ms.        | Tushar    | ms.tr@gmail.com           |
|        104 | Prof.      | Anand     | prof.and@gmail.com        |
|        105 | Dr.        | Mohamed   | dr.mo@gmail.com           |
|        106 | Ms.        | Clark     | ms.clark@yahoo.com        |
|        107 | Prof.      | Johnson   | prof.johnson@hotmail.com  |
|        108 | Ms.        | Taylor    | ms.taylor@YAHOO.com       |
|        109 | Prof.      | Anders    | prof.anderson@gmail.com   |
|        110 | Dr.        | Manoj     | dr.manoj@gmail.com        |
+------------+------------+-----------+---------------------------+
10 rows in set (0.00 sec)

mysql>
```

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.



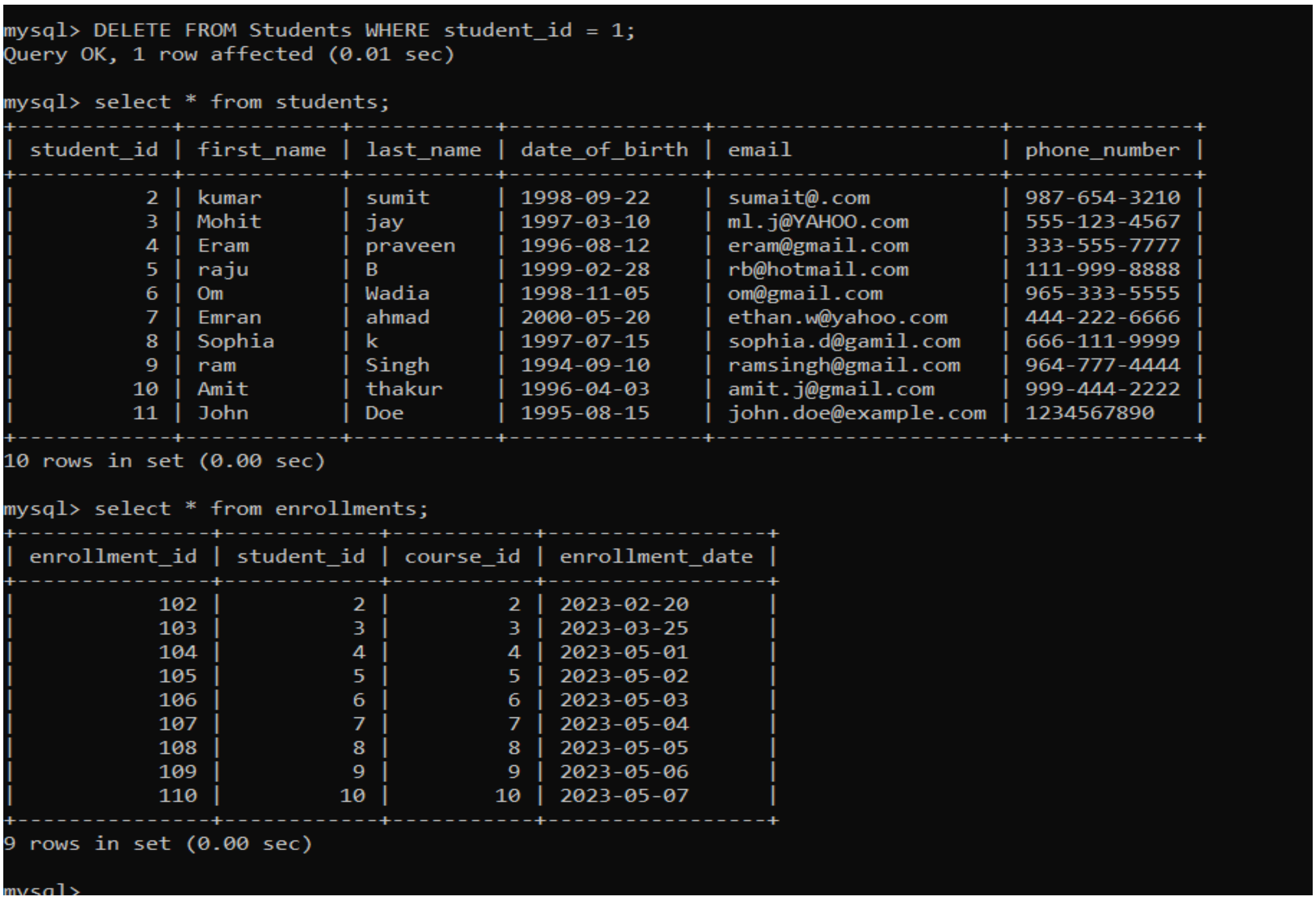
```
mysql> DELETE FROM Enrollments
    -> WHERE student_id = 1 AND course_id = 1;
Query OK, 2 rows affected (0.01 sec)

mysql> select * from enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           102 |          2 |         2 | 2023-02-20      |
|           103 |          3 |         3 | 2023-03-25      |
|           104 |          4 |         4 | 2023-05-01      |
|           105 |          5 |         5 | 2023-05-02      |
|           106 |          6 |         6 | 2023-05-03      |
|           107 |          7 |         7 | 2023-05-04      |
|           108 |          8 |         8 | 2023-05-05      |
|           109 |          9 |         9 | 2023-05-06      |
|           110 |         10 |        10 | 2023-05-07      |
+---------------+------------+-----------+-----------------+
9 rows in set (0.00 sec)

mysql>
```

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



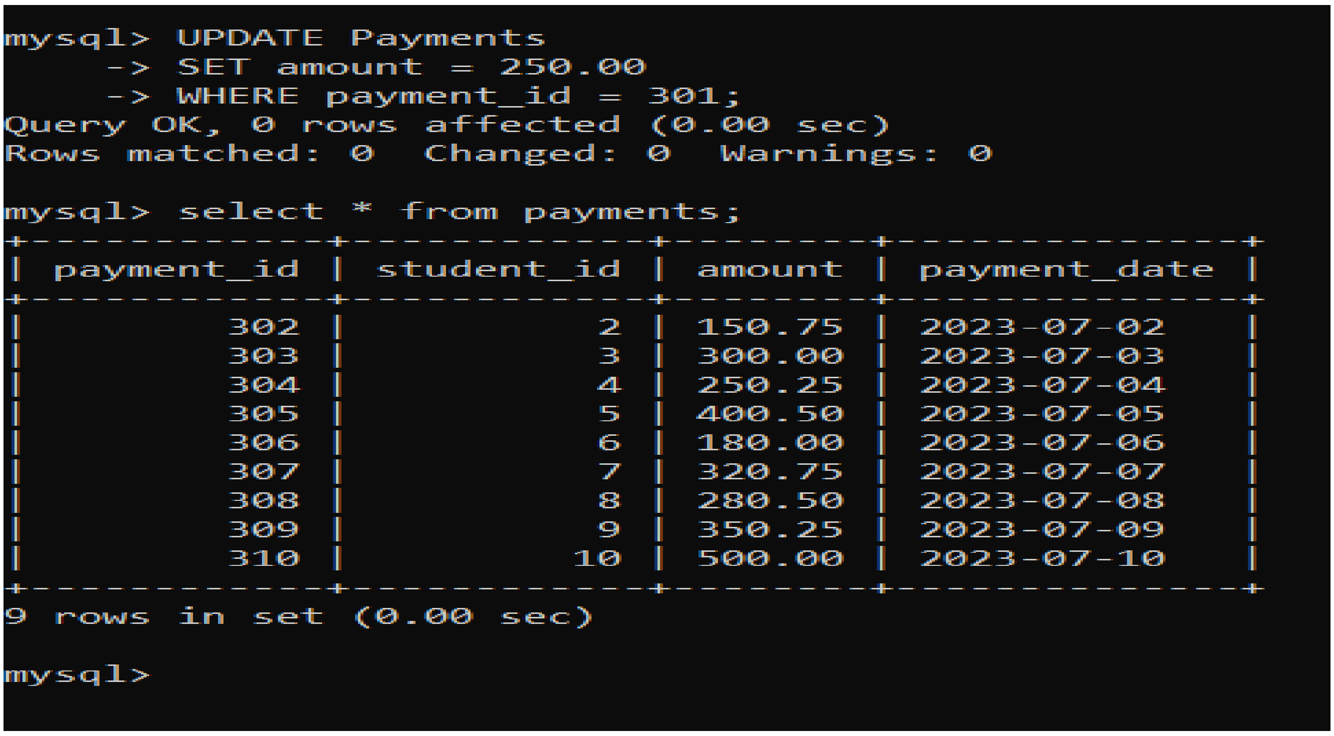
```
mysql> UPDATE Courses
    -> SET teacher_id = 101
    -> WHERE course_id = 1;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

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

mysql> _
```

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 Students WHERE student_id = 1;
Query OK, 1 row affected (0.01 sec)

mysql> select * from students;
+------------+------------+-----------+---------------+----------------------+--------------+
| student_id | first_name | last_name | date_of_birth | email                | phone_number |
+------------+------------+-----------+---------------+----------------------+--------------+
|          2 | kumar      | sumit     | 1998-09-22    | sumait@.com          | 987-654-3210 |
|          3 | Mohit      | jay       | 1997-03-10    | ml.j@YAHOO.com       | 555-123-4567 |
|          4 | Eram       | praveen   | 1996-08-12    | eram@gmail.com       | 333-555-7777 |
|          5 | raju       | B         | 1999-02-28    | rb@hotmail.com       | 111-999-8888 |
|          6 | Om         | Wadia     | 1998-11-05    | om@gmail.com         | 965-333-5555 |
|          7 | Emran      | ahmad     | 2000-05-20    | ethan.w@yahoo.com    | 444-222-6666 |
|          8 | Sophia     | k         | 1997-07-15    | sophia.d@gamil.com   | 666-111-9999 |
|          9 | ram        | Singh     | 1994-09-10    | ramsingh@gmail.com   | 964-777-4444 |
|         10 | Amit       | thakur    | 1996-04-03    | amit.j@gmail.com     | 999-444-2222 |
|         11 | John       | Doe       | 1995-08-15    | john.doe@example.com | 1234567890   |
+------------+------------+-----------+---------------+----------------------+--------------+
10 rows in set (0.00 sec)

mysql> select * from enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           102 |          2 |         2 | 2023-02-20      |
|           103 |          3 |         3 | 2023-03-25      |
|           104 |          4 |         4 | 2023-05-01      |
|           105 |          5 |         5 | 2023-05-02      |
|           106 |          6 |         6 | 2023-05-03      |
|           107 |          7 |         7 | 2023-05-04      |
|           108 |          8 |         8 | 2023-05-05      |
|           109 |          9 |         9 | 2023-05-06      |
|           110 |         10 |        10 | 2023-05-07      |
+---------------+------------+-----------+-----------------+
9 rows in set (0.00 sec)

mysql>
```

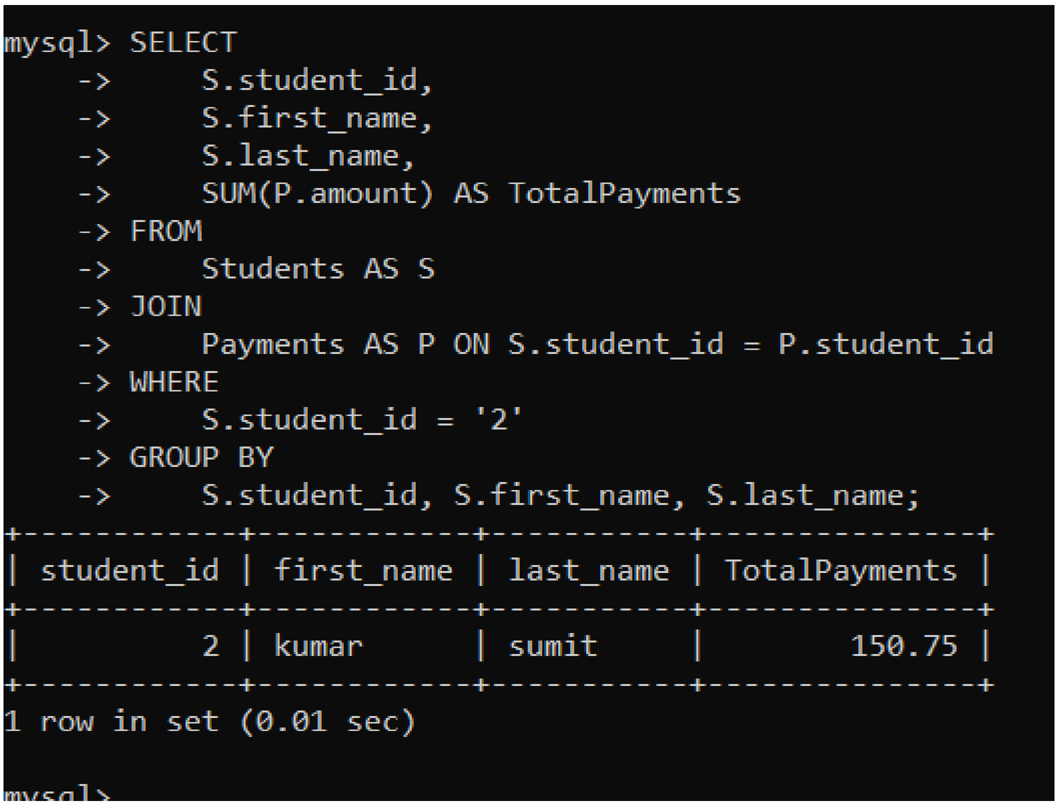
7. Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.



```
mysql> UPDATE Payments
    -> SET amount = 250.00
    -> WHERE payment_id = 301;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0

mysql> select * from payments;
+------------+------------+--------+--------------+
| payment_id | student_id | amount | payment_date |
+------------+------------+--------+--------------+
|        302 |          2 | 150.75 | 2023-07-02   |
|        303 |          3 | 300.00 | 2023-07-03   |
|        304 |          4 | 250.25 | 2023-07-04   |
|        305 |          5 | 400.50 | 2023-07-05   |
|        306 |          6 | 180.00 | 2023-07-06   |
|        307 |          7 | 320.75 | 2023-07-07   |
|        308 |          8 | 280.50 | 2023-07-08   |
|        309 |          9 | 350.25 | 2023-07-09   |
|        310 |         10 | 500.00 | 2023-07-10   |
+------------+------------+--------+--------------+
9 rows in set (0.00 sec)

mysql>
```

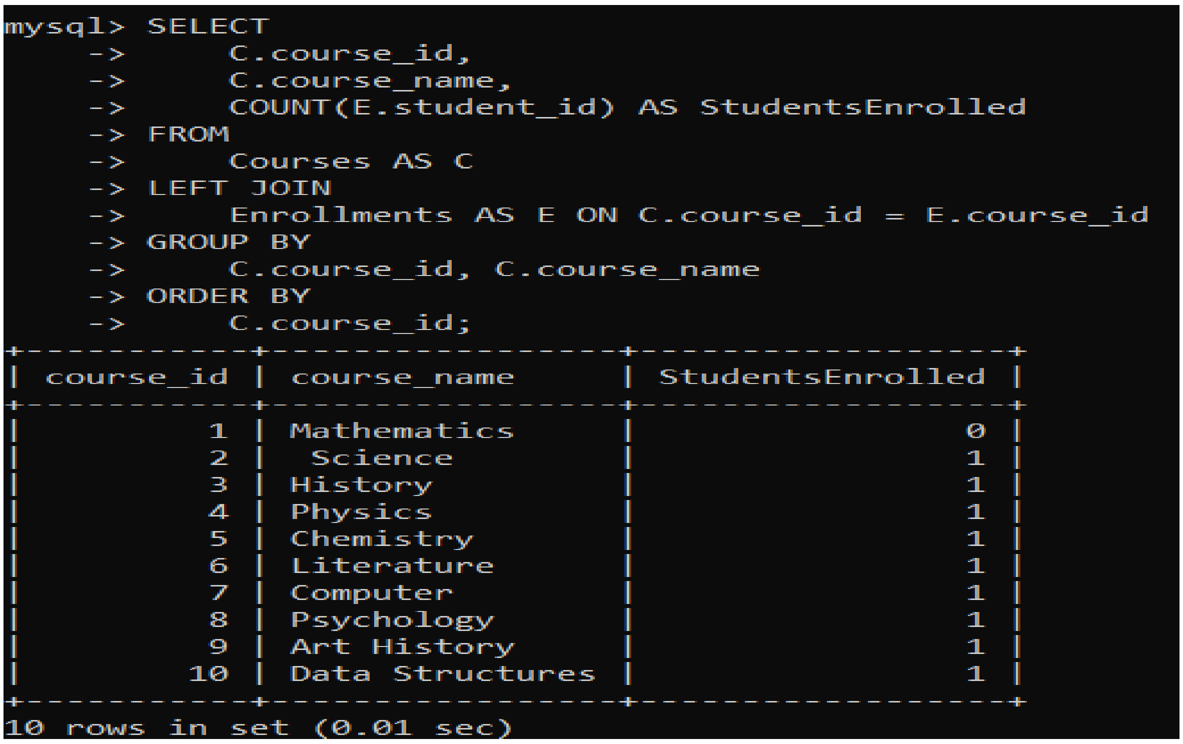
Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:

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.**



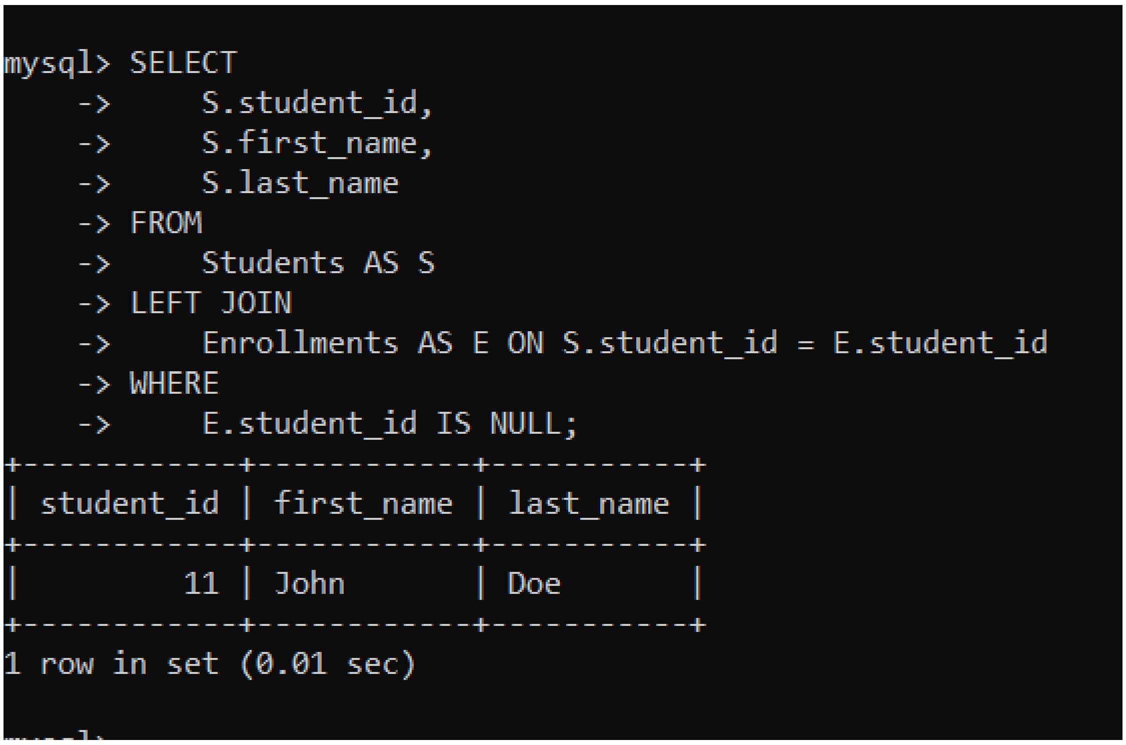
```
mysql> SELECT
    ->     S.student_id,
    ->     S.first_name,
    ->     S.last_name,
    ->     SUM(P.amount) AS TotalPayments
    -> FROM
    ->     Students AS S
    -> JOIN
    ->     Payments AS P ON S.student_id = P.student_id
    -> WHERE
    ->     S.student_id = '2'
    -> GROUP BY
    ->     S.student_id, S.first_name, S.last_name;
+------------+------------+-----------+---------------+
| student_id | first_name | last_name | TotalPayments |
+------------+------------+-----------+---------------+
|          2 | kumar      | sumit     |        150.75 |
+------------+------------+-----------+---------------+
1 row in set (0.01 sec)

mysql>
```

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.



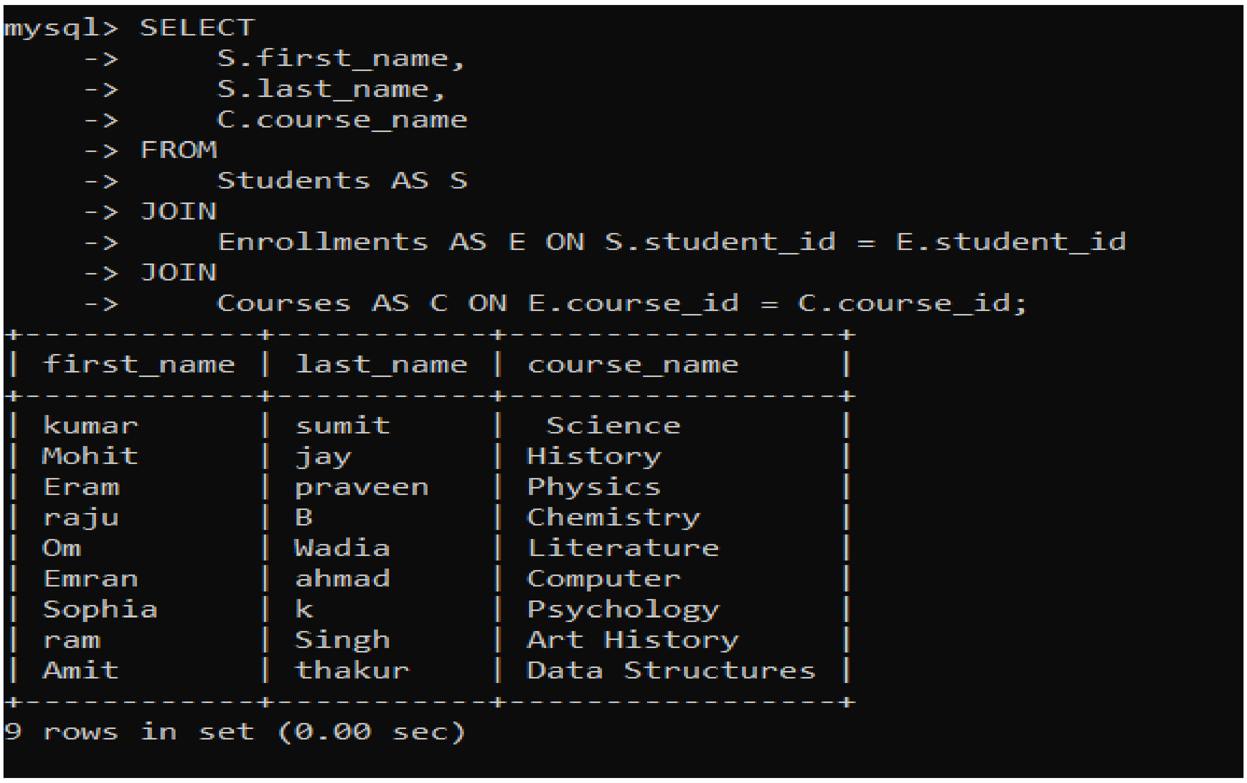
```
mysql> SELECT
    ->     C.course_id,
    ->     C.course_name,
    ->     COUNT(E.student_id) AS StudentsEnrolled
    -> FROM
    ->     Courses AS C
    -> LEFT JOIN
    ->     Enrollments AS E ON C.course_id = E.course_id
    -> GROUP BY
    ->     C.course_id, C.course_name
    -> ORDER BY
    ->     C.course_id;
+-----------+------------------+------------------+
| course_id | course_name      | StudentsEnrolled |
+-----------+------------------+------------------+
|         1 | Mathematics      |                0 |
|         2 |  Science         |                1 |
|         3 | History          |                1 |
|         4 | Physics          |                1 |
|         5 | Chemistry        |                1 |
|         6 | Literature       |                1 |
|         7 | Computer         |                1 |
|         8 | Psychology       |                1 |
|         9 | Art History      |                1 |
|        10 | Data Structures  |                1 |
+-----------+------------------+------------------+
10 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.



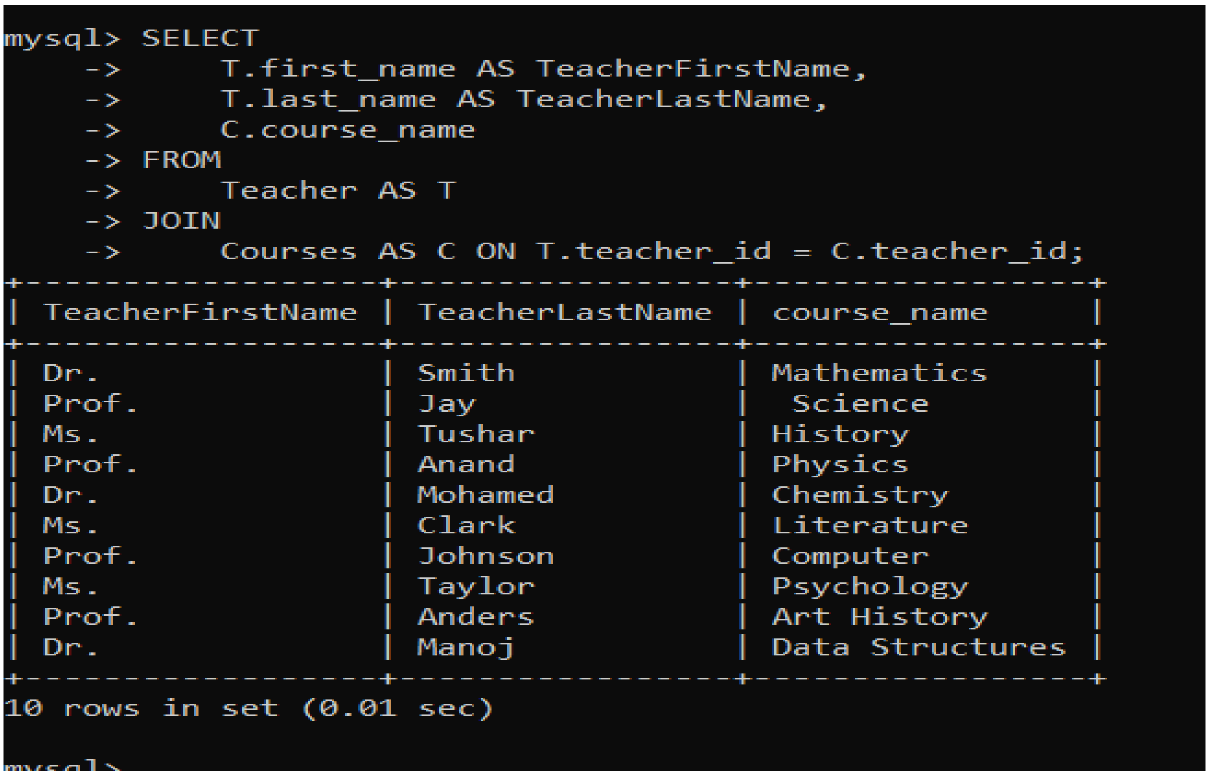
```
mysql> SELECT
    ->     S.student_id,
    ->     S.first_name,
    ->     S.last_name
    -> FROM
    ->     Students AS S
    -> LEFT JOIN
    ->     Enrollments AS E ON S.student_id = E.student_id
    -> WHERE
    ->     E.student_id IS NULL;
+------------+------------+-----------+
| student_id | first_name | last_name |
+------------+------------+-----------+
|         11 | John       | Doe       |
+------------+------------+-----------+
1 row in set (0.01 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.



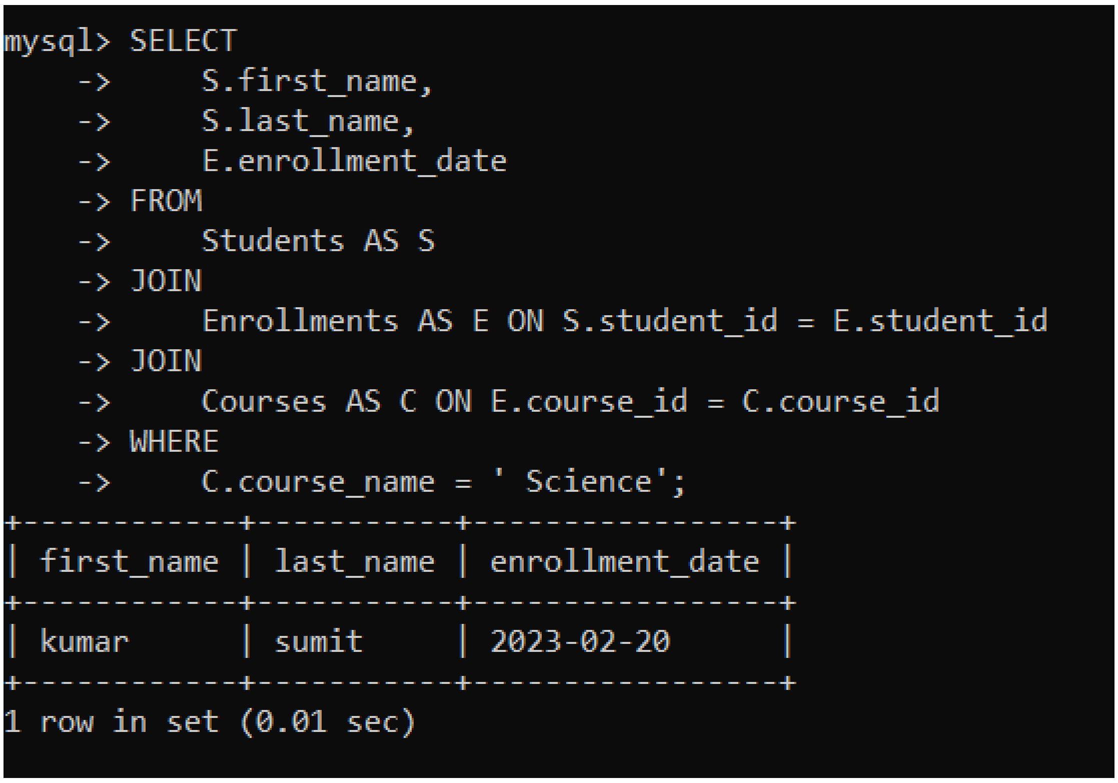
```
mysql> SELECT
    ->     S.first_name,
    ->     S.last_name,
    ->     C.course_name
    -> FROM
    ->     Students AS S
    -> JOIN
    ->     Enrollments AS E ON S.student_id = E.student_id
    -> JOIN
    ->     Courses AS C ON E.course_id = C.course_id;
+------------+-----------+-----------------+
| first_name | last_name | course_name     |
+------------+-----------+-----------------+
| kumar      | sumit     |  Science        |
| Mohit      | jay       | History         |
| Eram       | praveen   | Physics         |
| raju       | B         | Chemistry       |
| Om         | Wadia     | Literature      |
| Emran      | ahmad     | Computer        |
| Sophia     | k         | Psychology      |
| ram        | Singh     | Art History     |
| Amit       | thakur    | Data Structures |
+------------+-----------+-----------------+
9 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.



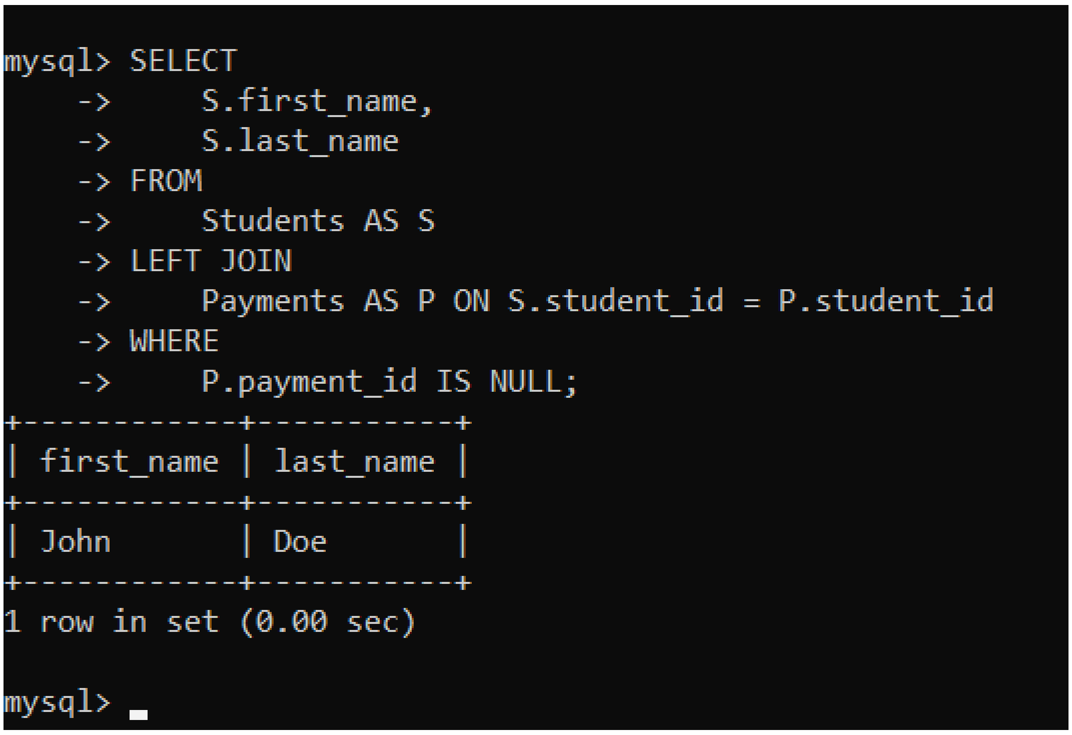
```
mysql> SELECT
    ->     T.first_name AS TeacherFirstName,
    ->     T.last_name AS TeacherLastName,
    ->     C.course_name
    -> FROM
    ->     Teacher AS T
    -> JOIN
    ->     Courses AS C ON T.teacher_id = C.teacher_id;
+------------------+-----------------+------------------+
| TeacherFirstName | TeacherLastName | course_name      |
+------------------+-----------------+------------------+
| Dr.              | Smith           | Mathematics      |
| Prof.            | Jay             |  Science         |
| Ms.              | Tushar          | History          |
| Prof.            | Anand           | Physics          |
| Dr.              | Mohamed         | Chemistry        |
| Ms.              | Clark           | Literature       |
| Prof.            | Johnson         | Computer         |
| Ms.              | Taylor          | Psychology       |
| Prof.            | Anders          | Art History      |
| Dr.              | Manoj           | Data Structures  |
+------------------+-----------------+------------------+
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.



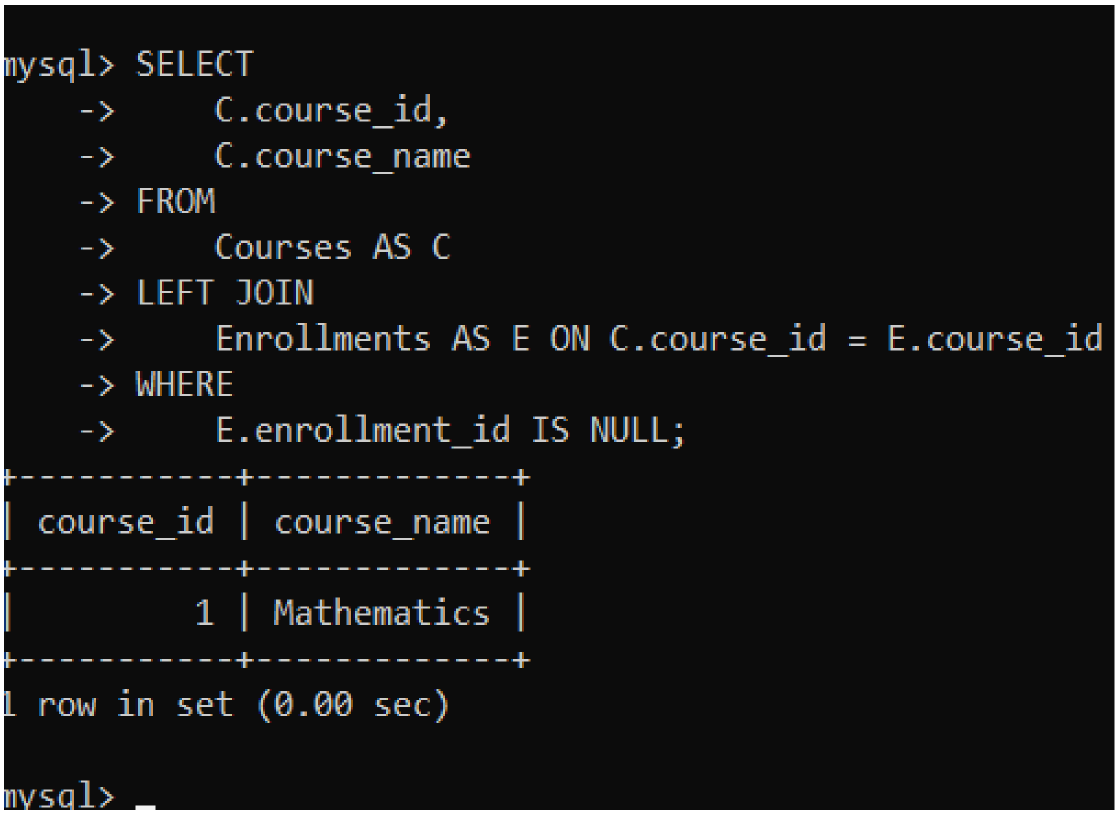
```
mysql> SELECT
    ->     S.first_name,
    ->     S.last_name,
    ->     E.enrollment_date
    -> FROM
    ->     Students AS S
    -> JOIN
    ->     Enrollments AS E ON S.student_id = E.student_id
    -> JOIN
    ->     Courses AS C ON E.course_id = C.course_id
    -> WHERE
    ->     C.course_name = ' Science';
+------------+-----------+-----------------+
| first_name | last_name | enrollment_date |
+------------+-----------+-----------------+
| kumar      | sumit     | 2023-02-20      |
+------------+-----------+-----------------+
1 row in set (0.01 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.



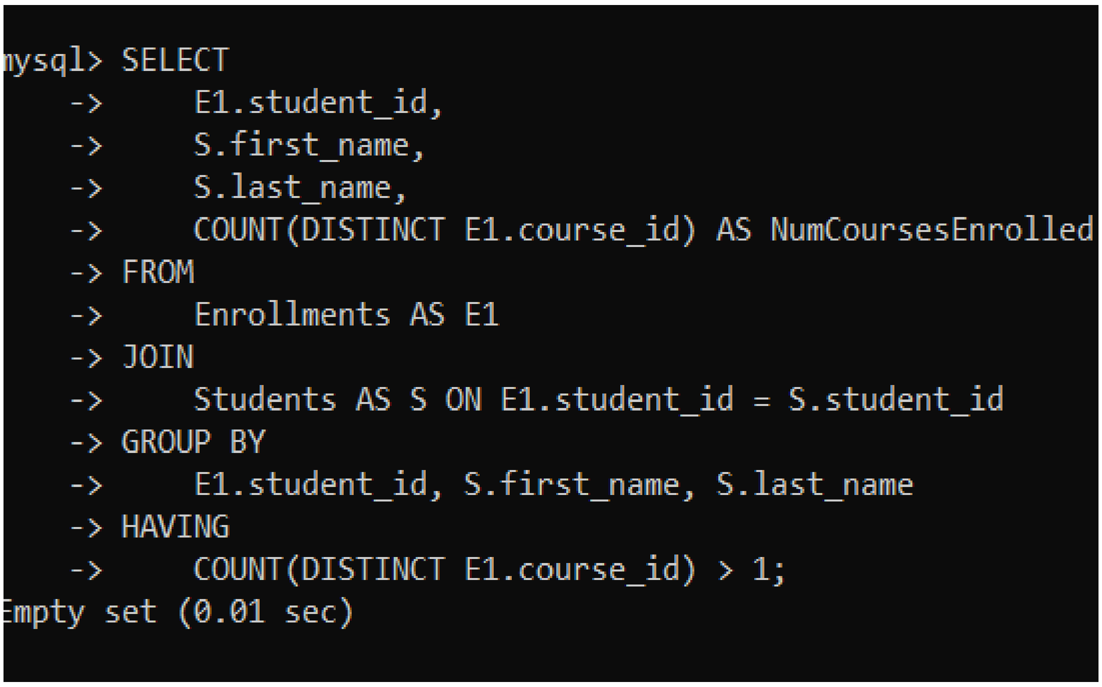
```
mysql> SELECT
    ->     S.first_name,
    ->     S.last_name
    -> FROM
    ->     Students AS S
    -> LEFT JOIN
    ->     Payments AS P ON S.student_id = P.student_id
    -> WHERE
    ->     P.payment_id IS NULL;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| John       | Doe       |
+------------+-----------+
1 row in set (0.00 sec)

mysql> _
```

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.



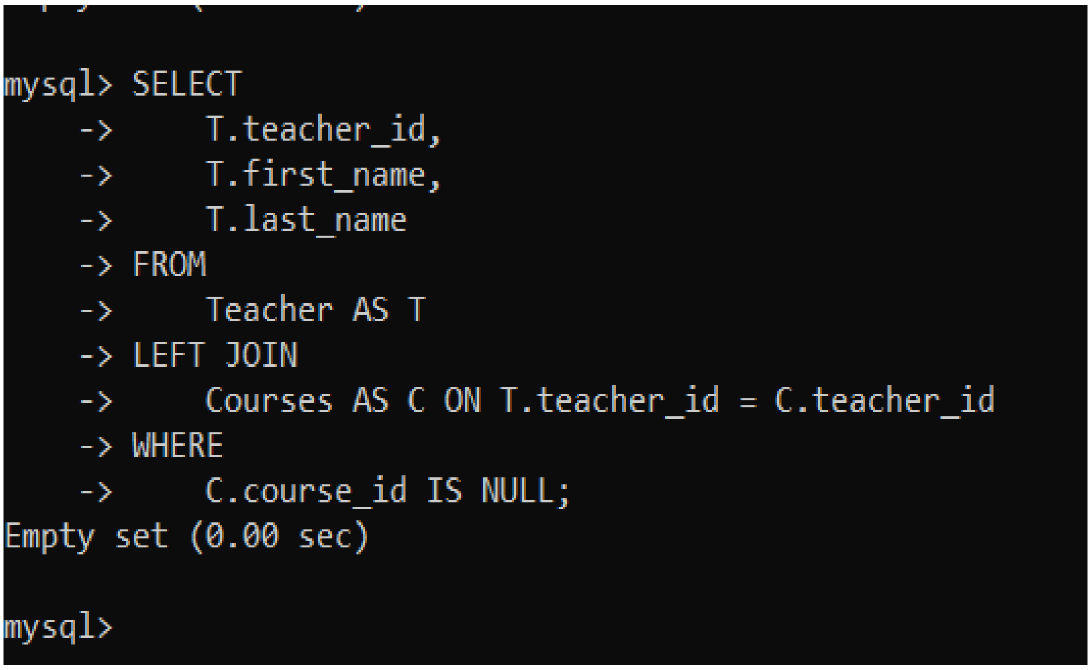
```
mysql> SELECT
    ->     C.course_id,
    ->     C.course_name
    -> FROM
    ->     Courses AS C
    -> LEFT JOIN
    ->     Enrollments AS E ON C.course_id = E.course_id
    -> WHERE
    ->     E.enrollment_id IS NULL;
+-----------+-------------+
| course_id | course_name |
+-----------+-------------+
|         1 | Mathematics |
+-----------+-------------+
1 row in set (0.00 sec)

mysql> _
```

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.



```
ysql> SELECT
    ->     E1.student_id,
    ->     S.first_name,
    ->     S.last_name,
    ->     COUNT(DISTINCT E1.course_id) AS NumCoursesEnrolled
    -> FROM
    ->     Enrollments AS E1
    -> JOIN
    ->     Students AS S ON E1.student_id = S.student_id
    -> GROUP BY
    ->     E1.student_id, S.first_name, S.last_name
    -> HAVING
    ->     COUNT(DISTINCT E1.course_id) > 1;
mpty set (0.01 sec)
```

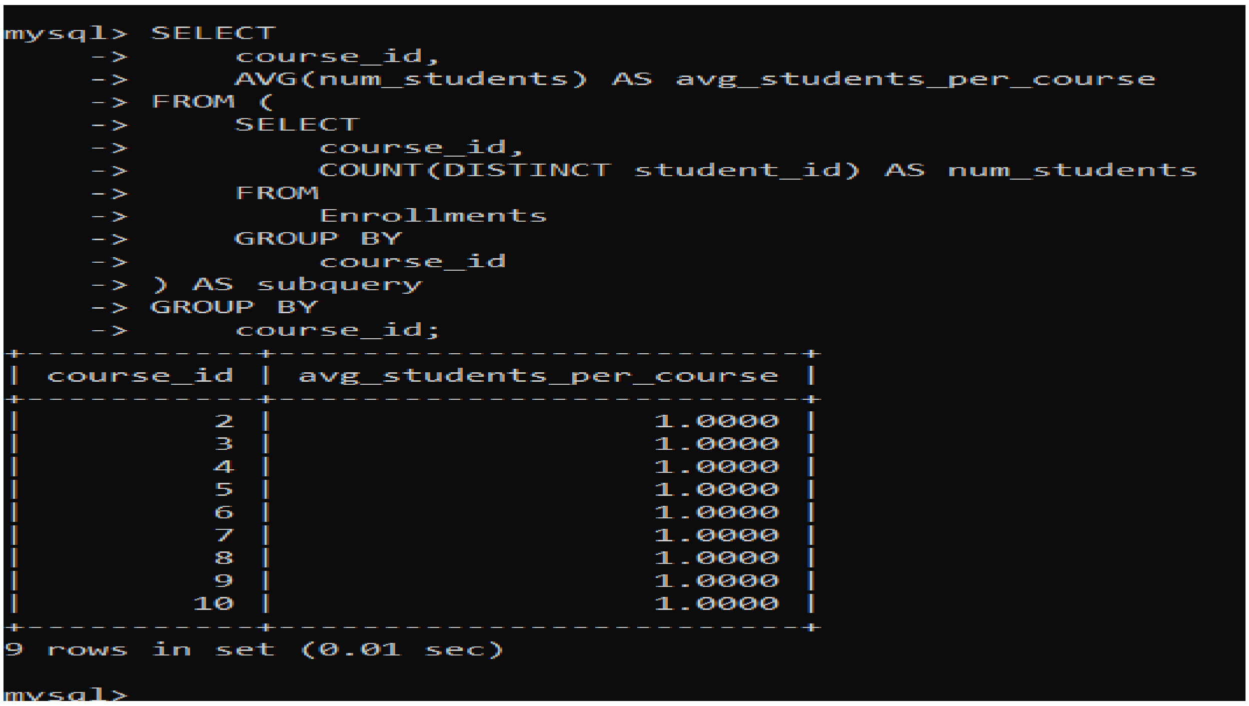
10. Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher" table and the "Courses" table and filter for teachers with NULL course assignments.



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

mysql>
```

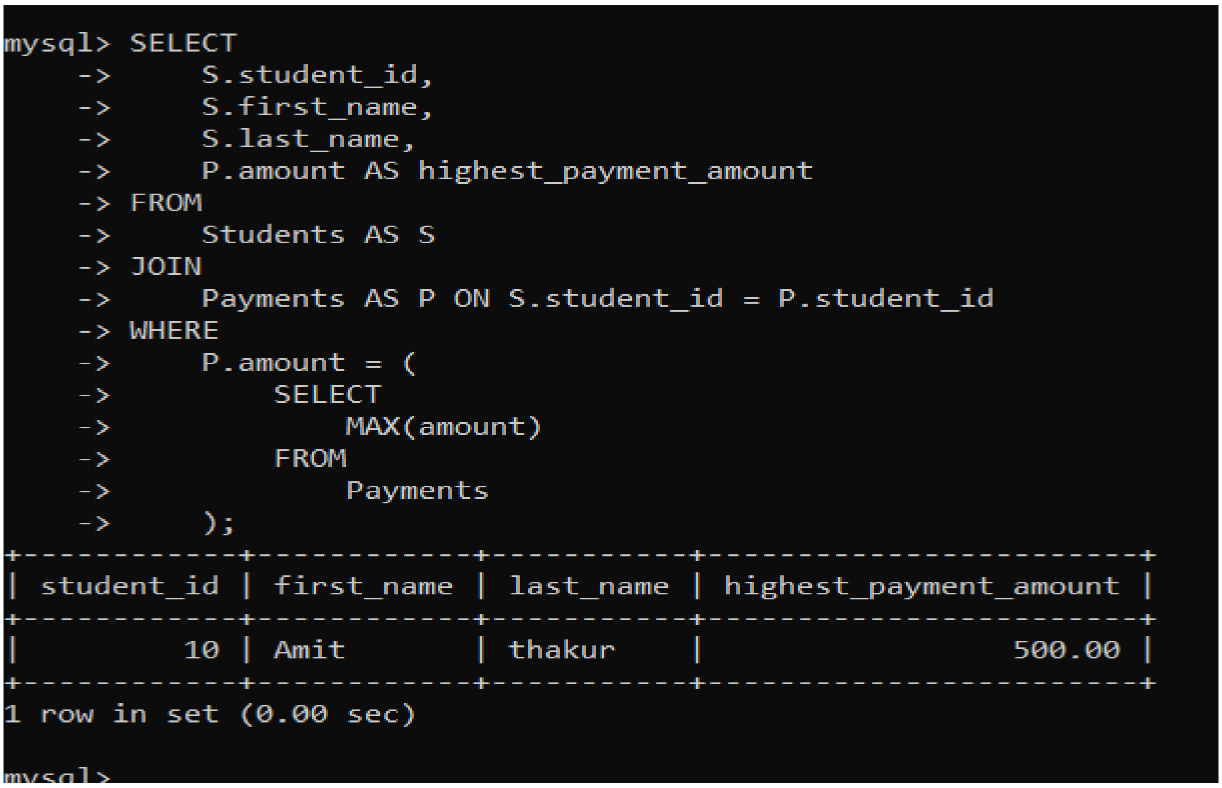
Task 4. Subquery and its type:

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



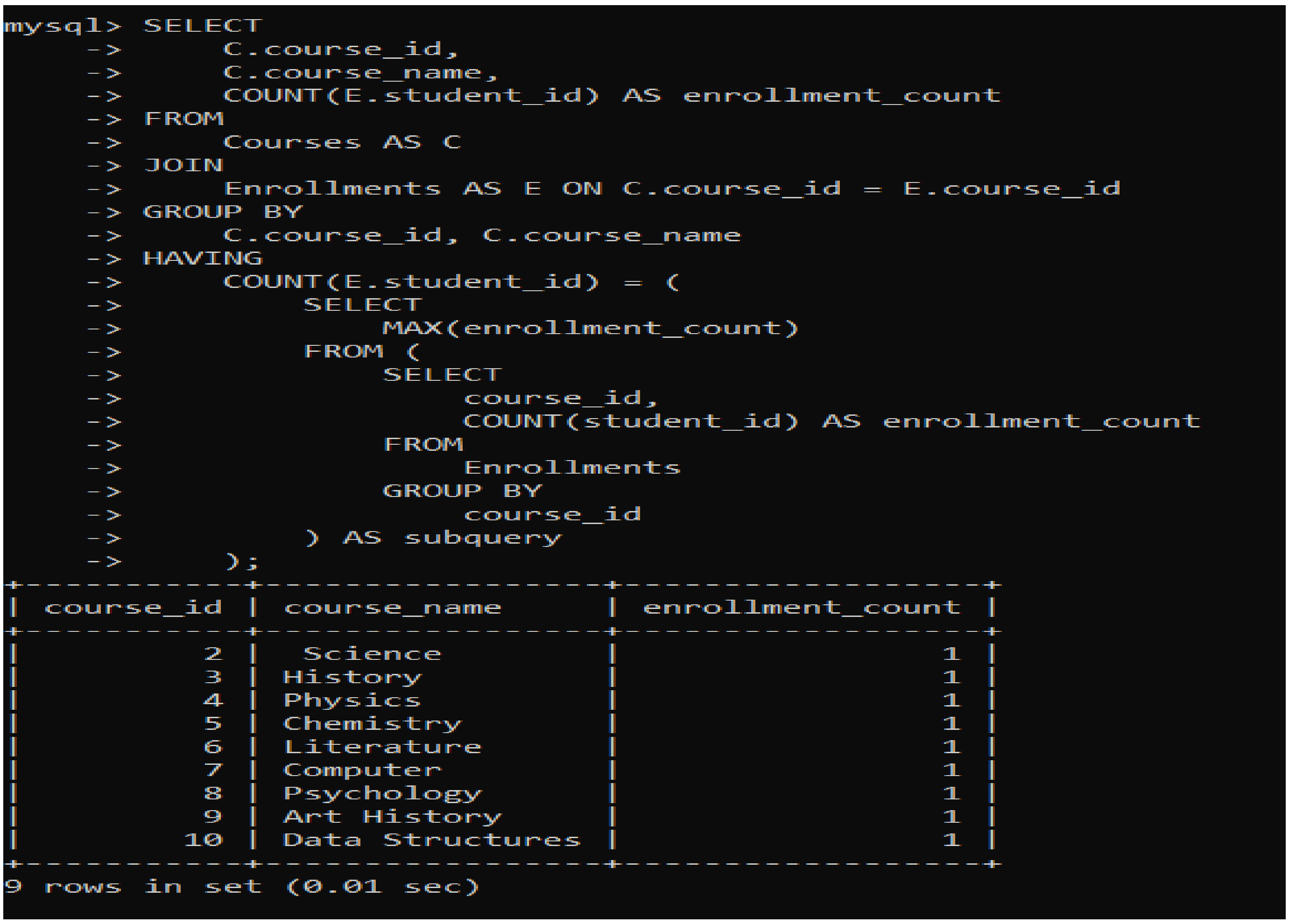
```
mysql> SELECT
    ->     course_id,
    ->     AVG(num_students) AS avg_students_per_course
    -> FROM (
    ->     SELECT
    ->         course_id,
    ->         COUNT(DISTINCT student_id) AS num_students
    ->     FROM
    ->         Enrollments
    ->     GROUP BY
    ->         course_id
    -> ) AS subquery
    -> GROUP BY
    ->     course_id;
+-----------+-------------------------+
| course_id | avg_students_per_course |
+-----------+-------------------------+
|         2 |                  1.0000 |
|         3 |                  1.0000 |
|         4 |                  1.0000 |
|         5 |                  1.0000 |
|         6 |                  1.0000 |
|         7 |                  1.0000 |
|         8 |                  1.0000 |
|         9 |                  1.0000 |
|        10 |                  1.0000 |
+-----------+-------------------------+
9 rows in set (0.01 sec)

mysql>
```

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.**



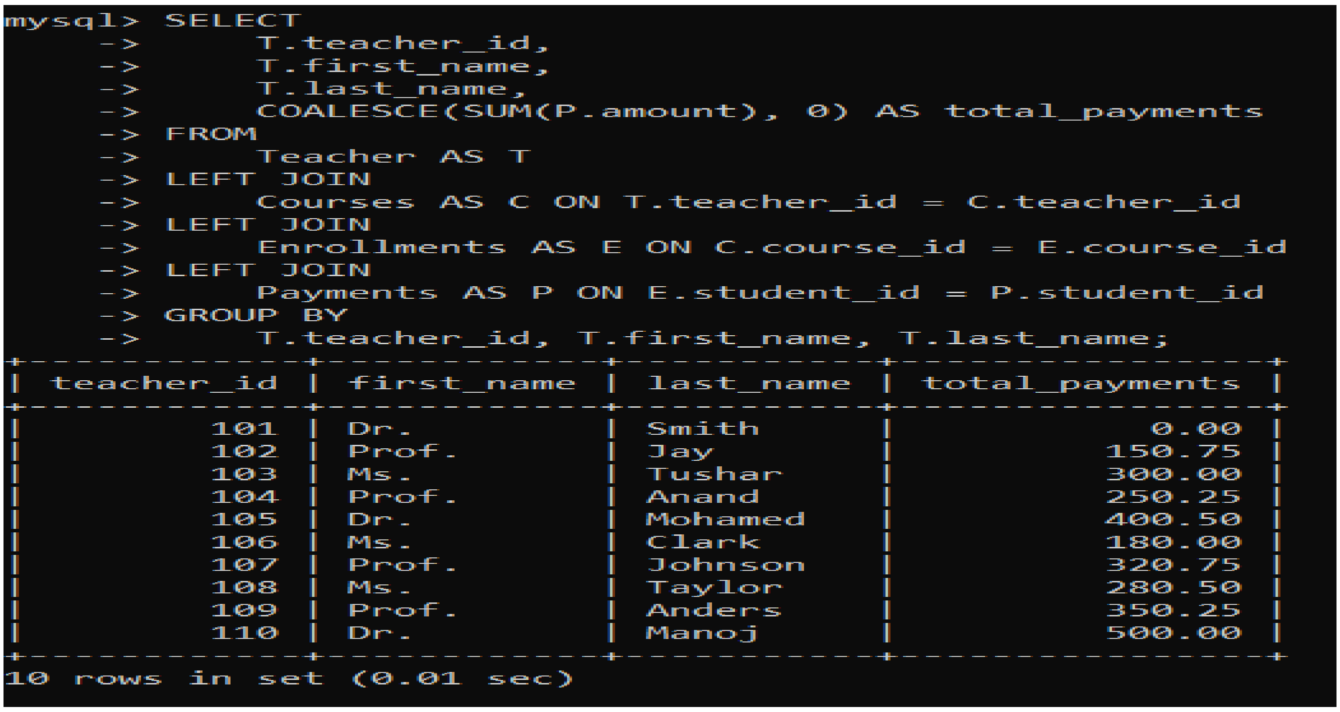
```
mysql> SELECT
    ->     S.student_id,
    ->     S.first_name,
    ->     S.last_name,
    ->     P.amount AS highest_payment_amount
    -> FROM
    ->     Students AS S
    -> JOIN
    ->     Payments AS P ON S.student_id = P.student_id
    -> WHERE
    ->     P.amount = (
    ->         SELECT
    ->             MAX(amount)
    ->         FROM
    ->             Payments
    ->     );
+------------+------------+-----------+------------------------+
| student_id | first_name | last_name | highest_payment_amount |
+------------+------------+-----------+------------------------+
|         10 | Amit       | thakur    |                 500.00 |
+------------+------------+-----------+------------------------+
1 row in set (0.00 sec)

mysql>
```

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



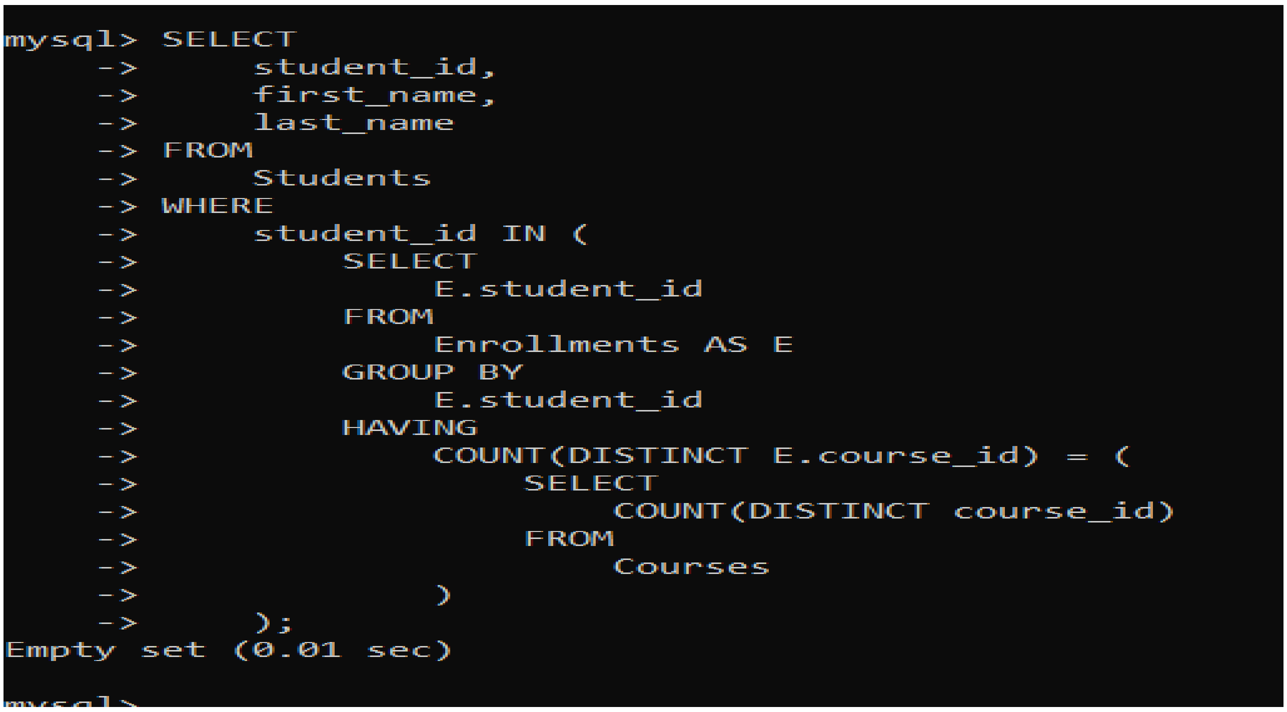
```
mysql> SELECT
    ->     C.course_id,
    ->     C.course_name,
    ->     COUNT(E.student_id) AS enrollment_count
    -> FROM
    ->     Courses AS C
    -> JOIN
    ->     Enrollments AS E ON C.course_id = E.course_id
    -> GROUP BY
    ->     C.course_id, C.course_name
    -> HAVING
    ->     COUNT(E.student_id) = (
    ->         SELECT
    ->             MAX(enrollment_count)
    ->         FROM (
    ->             SELECT
    ->                 course_id,
    ->                 COUNT(student_id) AS enrollment_count
    ->             FROM
    ->                 Enrollments
    ->             GROUP BY
    ->                 course_id
    ->         ) AS subquery
    ->     );
+-----------+-----------------+------------------+
| course_id | course_name     | enrollment_count |
+-----------+-----------------+------------------+
|         2 |  Science        |                1 |
|         3 | History         |                1 |
|         4 | Physics         |                1 |
|         5 | Chemistry       |                1 |
|         6 | Literature      |                1 |
|         7 | Computer        |                1 |
|         8 | Psychology      |                1 |
|         9 | Art History     |                1 |
|        10 | Data Structures |                1 |
+-----------+-----------------+------------------+
9 rows in set (0.01 sec)
```

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



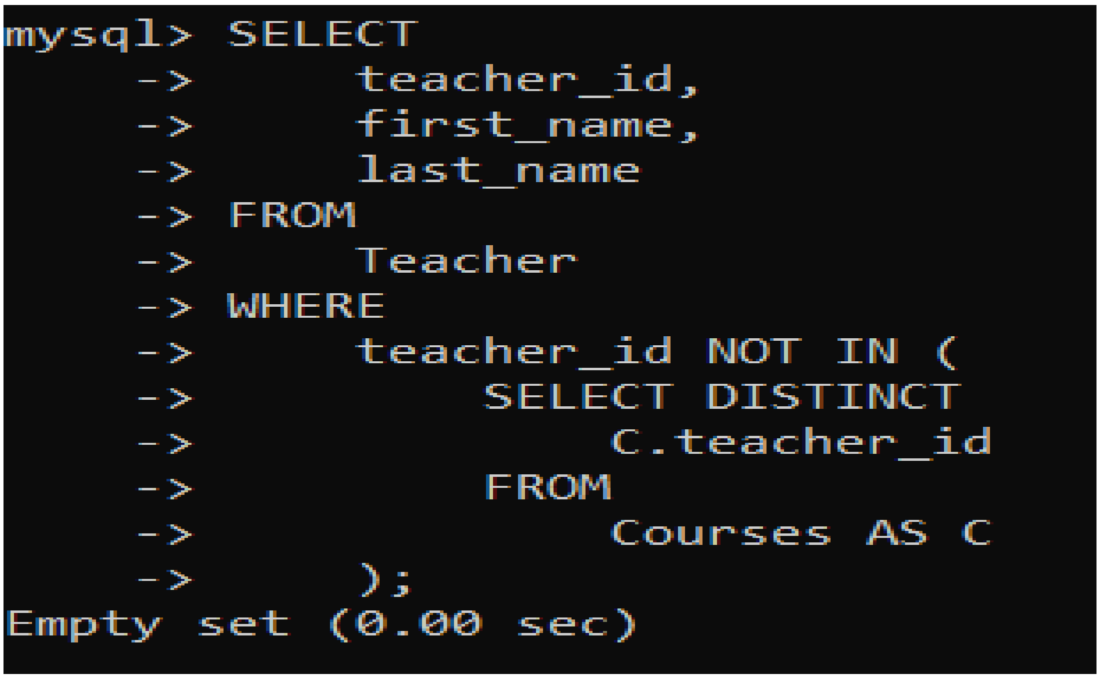
```
mysql> SELECT
    ->     T.teacher_id,
    ->     T.first_name,
    ->     T.last_name,
    ->     COALESCE(SUM(P.amount), 0) AS total_payments
    -> FROM
    ->     Teacher AS T
    -> LEFT JOIN
    ->     Courses AS C ON T.teacher_id = C.teacher_id
    -> LEFT JOIN
    ->     Enrollments AS E ON C.course_id = E.course_id
    -> LEFT JOIN
    ->     Payments AS P ON E.student_id = P.student_id
    -> GROUP BY
    ->     T.teacher_id, T.first_name, T.last_name;
+------------+------------+-----------+----------------+
| teacher_id | first_name | last_name | total_payments |
+------------+------------+-----------+----------------+
|        101 | Dr.        | Smith     |           0.00 |
|        102 | Prof.      | Jay       |         150.75 |
|        103 | Ms.        | Tushar    |         300.00 |
|        104 | Prof.      | Anand     |         250.25 |
|        105 | Dr.        | Mohamed   |         400.50 |
|        106 | Ms.        | Clark     |         180.00 |
|        107 | Prof.      | Johnson   |         320.75 |
|        108 | Ms.        | Taylor    |         280.50 |
|        109 | Prof.      | Anders    |         350.25 |
|        110 | Dr.        | Manoj     |         500.00 |
+------------+------------+-----------+----------------+
10 rows in set (0.01 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.



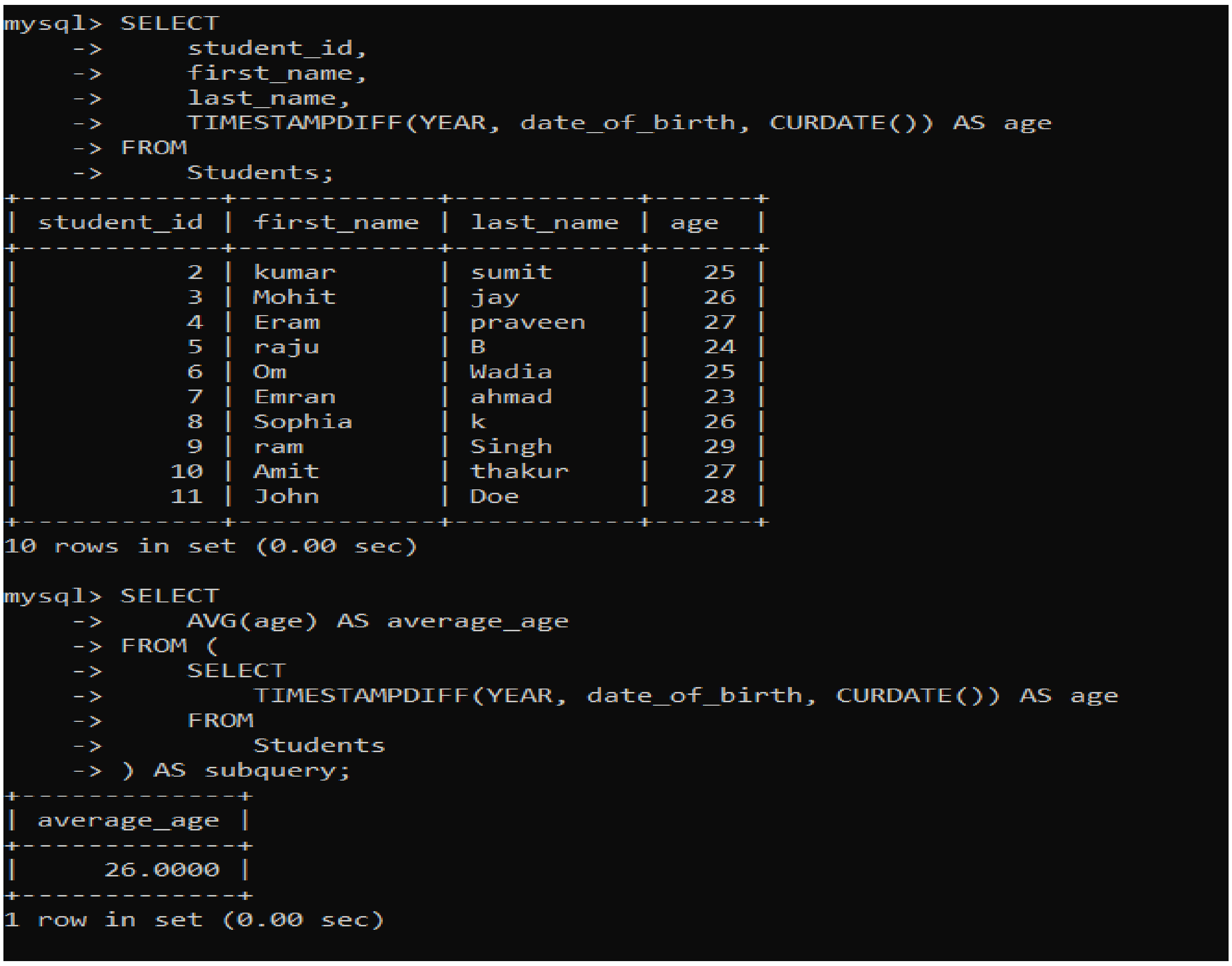
```
mysql> SELECT
    ->     student_id,
    ->     first_name,
    ->     last_name
    -> FROM
    ->     Students
    -> WHERE
    ->     student_id IN (
    ->         SELECT
    ->             E.student_id
    ->         FROM
    ->             Enrollments AS E
    ->         GROUP BY
    ->             E.student_id
    ->         HAVING
    ->             COUNT(DISTINCT E.course_id) = (
    ->                 SELECT
    ->                     COUNT(DISTINCT 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.



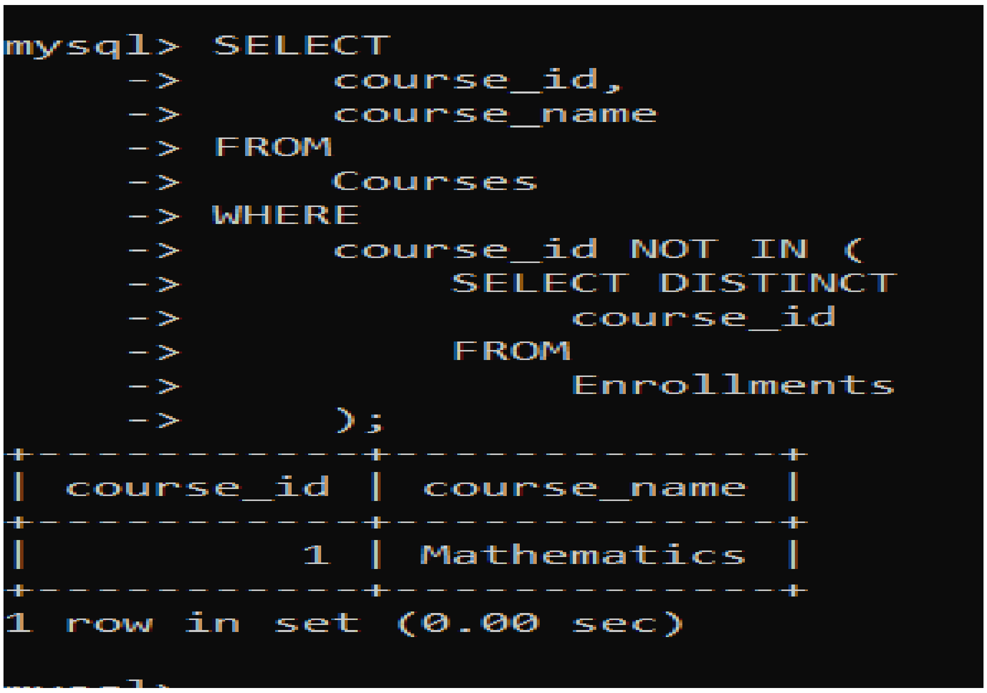
```
mysql> SELECT
    ->     teacher_id,
    ->     first_name,
    ->     last_name
    -> FROM
    ->     Teacher
    -> WHERE
    ->     teacher_id NOT IN (
    ->         SELECT DISTINCT
    ->             C.teacher_id
    ->         FROM
    ->             Courses AS C
    ->     );
Empty set (0.00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

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



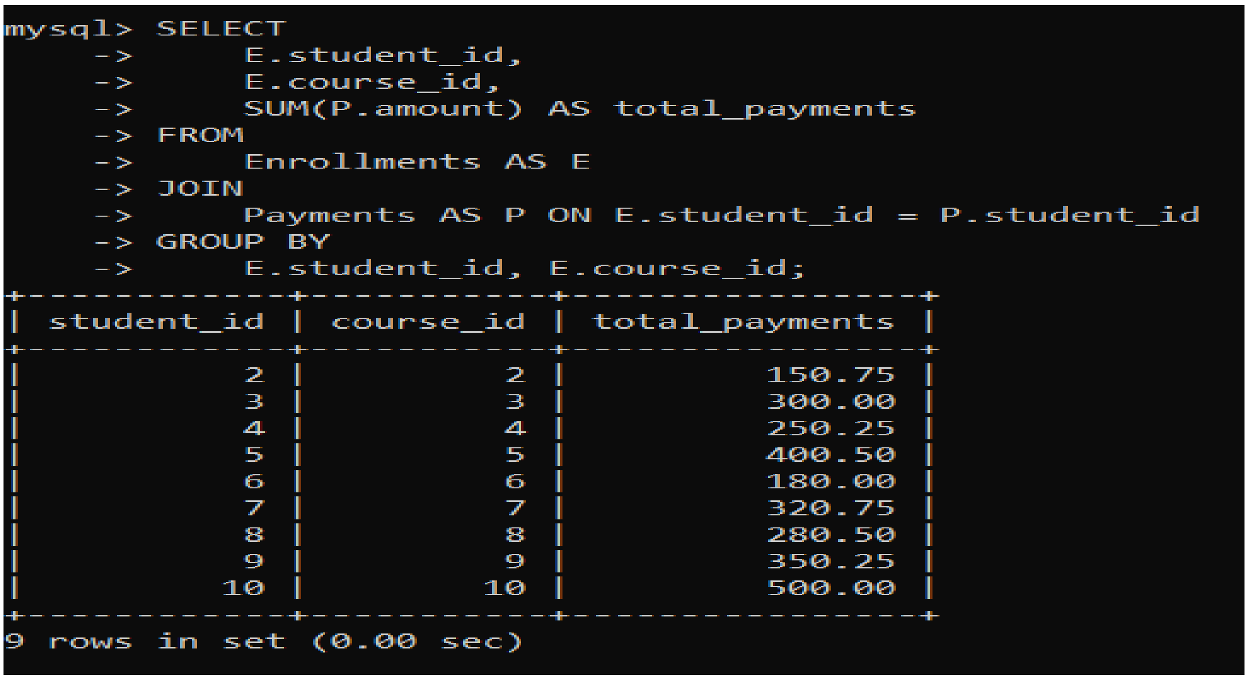
```
mysql> SELECT
    ->     student_id,
    ->     first_name,
    ->     last_name,
    ->     TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age
    -> FROM
    ->     Students;
+------------+------------+-----------+------+
| student_id | first_name | last_name | age  |
+------------+------------+-----------+------+
|          2 | kumar      | sumit     |   25 |
|          3 | Mohit      | jay       |   26 |
|          4 | Eram       | praveen   |   27 |
|          5 | raju       | B         |   24 |
|          6 | Om         | Wadia     |   25 |
|          7 | Emran      | ahmad     |   23 |
|          8 | Sophia     | k         |   26 |
|          9 | ram        | Singh     |   29 |
|         10 | Amit       | thakur    |   27 |
|         11 | John       | Doe       |   28 |
+------------+------------+-----------+------+
10 rows in set (0.00 sec)

mysql> SELECT
    ->     AVG(age) AS average_age
    -> FROM (
    ->     SELECT
    ->         TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age
    ->     FROM
    ->         Students
    -> ) AS subquery;
+-------------+
| average_age |
+-------------+
|     26.0000 |
+-------------+
1 row in set (0.00 sec)
```

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



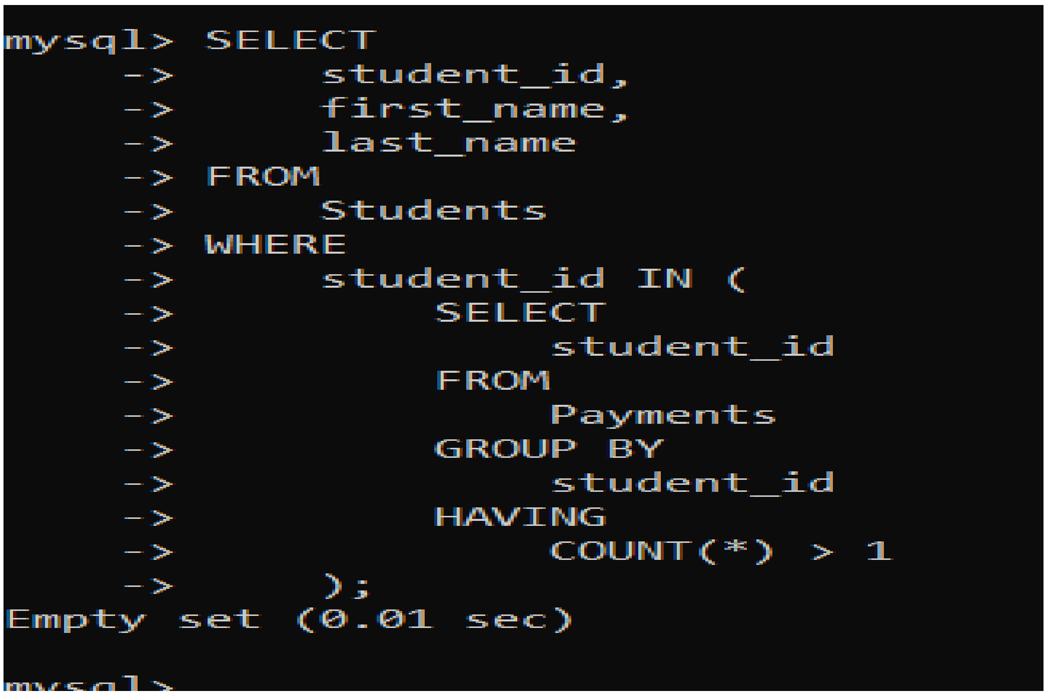
```
mysql> SELECT
    ->     course_id,
    ->     course_name
    -> FROM
    ->     Courses
    -> WHERE
    ->     course_id NOT IN (
    ->         SELECT DISTINCT
    ->             course_id
    ->         FROM
    ->             Enrollments
    ->     );
+-----------+-------------+
| course_id | course_name |
+-----------+-------------+
|         1 | Mathematics |
+-----------+-------------+
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.



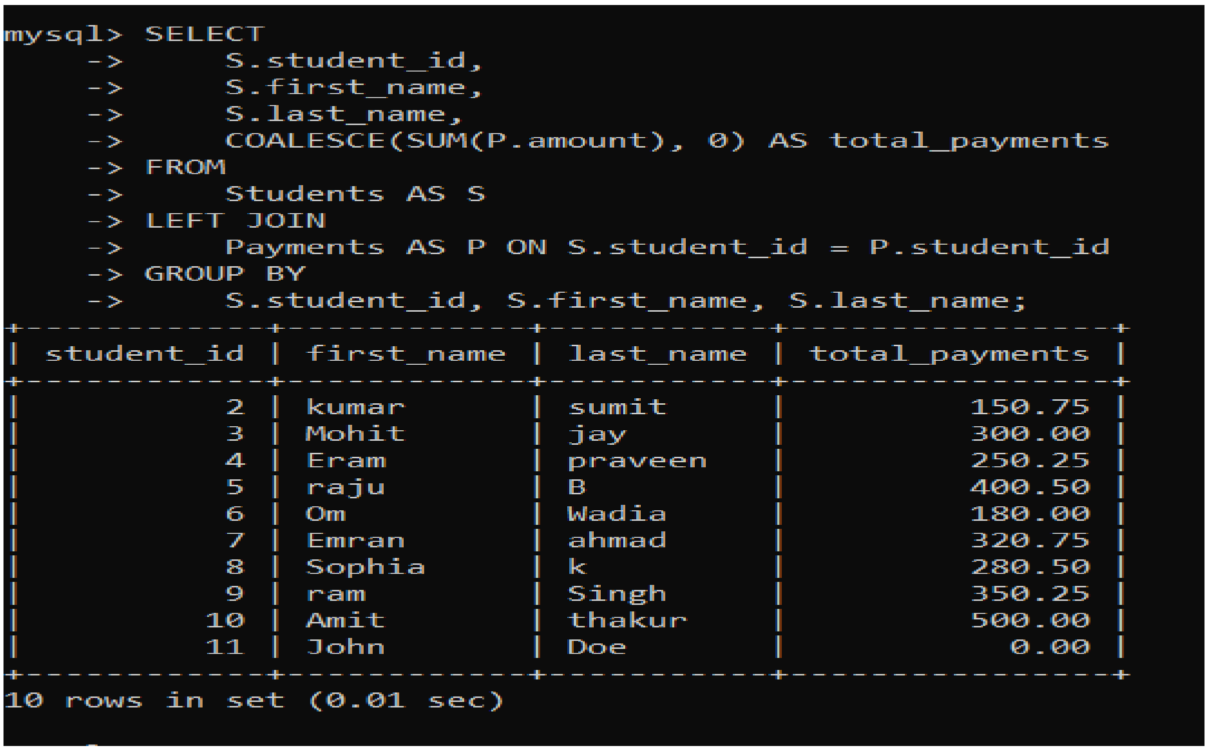
```
mysql> SELECT
    ->     E.student_id,
    ->     E.course_id,
    ->     SUM(P.amount) AS total_payments
    -> FROM
    ->     Enrollments AS E
    -> JOIN
    ->     Payments AS P ON E.student_id = P.student_id
    -> GROUP BY
    ->     E.student_id, E.course_id;
+------------+-----------+----------------+
| student_id | course_id | total_payments |
+------------+-----------+----------------+
|          2 |         2 |         150.75 |
|          3 |         3 |         300.00 |
|          4 |         4 |         250.25 |
|          5 |         5 |         400.50 |
|          6 |         6 |         180.00 |
|          7 |         7 |         320.75 |
|          8 |         8 |         280.50 |
|          9 |         9 |         350.25 |
|         10 |        10 |         500.00 |
+------------+-----------+----------------+
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.



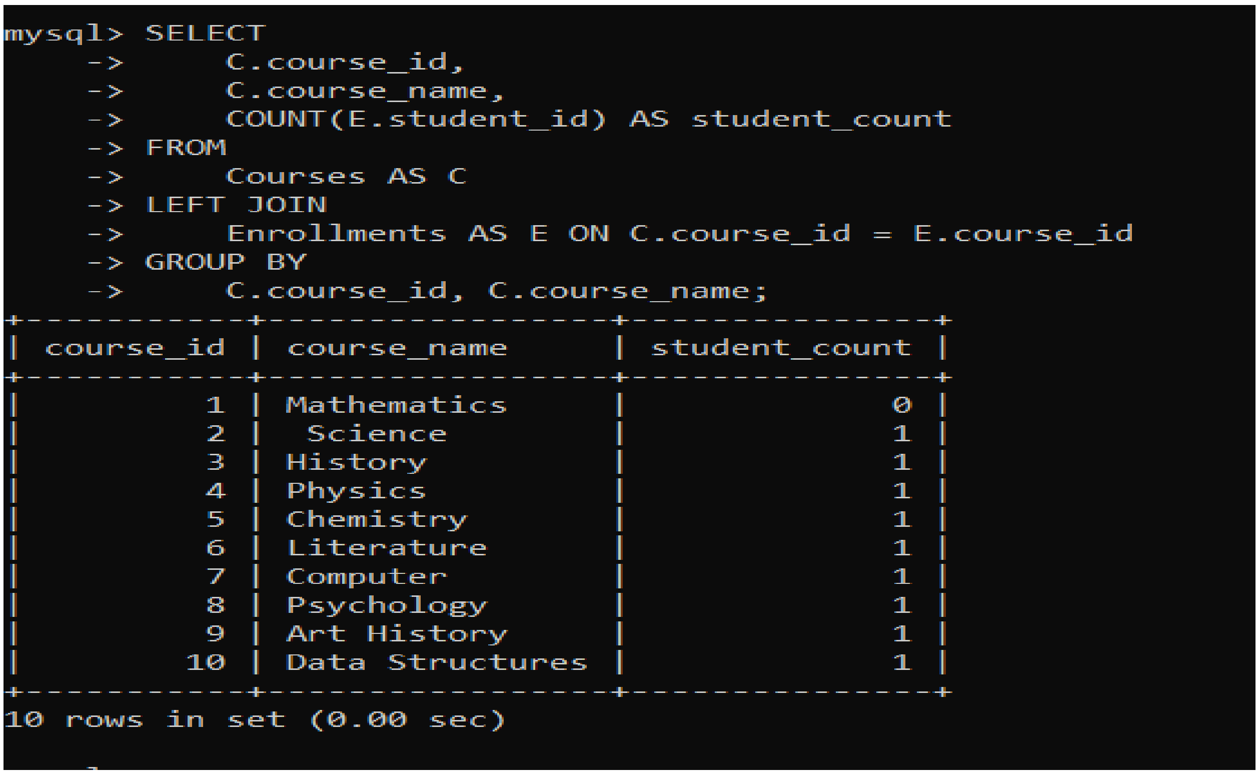
```
mysql> SELECT
    ->     student_id,
    ->     first_name,
    ->     last_name
    -> FROM
    ->     Students
    -> WHERE
    ->     student_id IN (
    ->         SELECT
    ->             student_id
    ->         FROM
    ->             Payments
    ->         GROUP BY
    ->             student_id
    ->         HAVING
    ->             COUNT(*) > 1
    ->     );
Empty set (0.01 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.



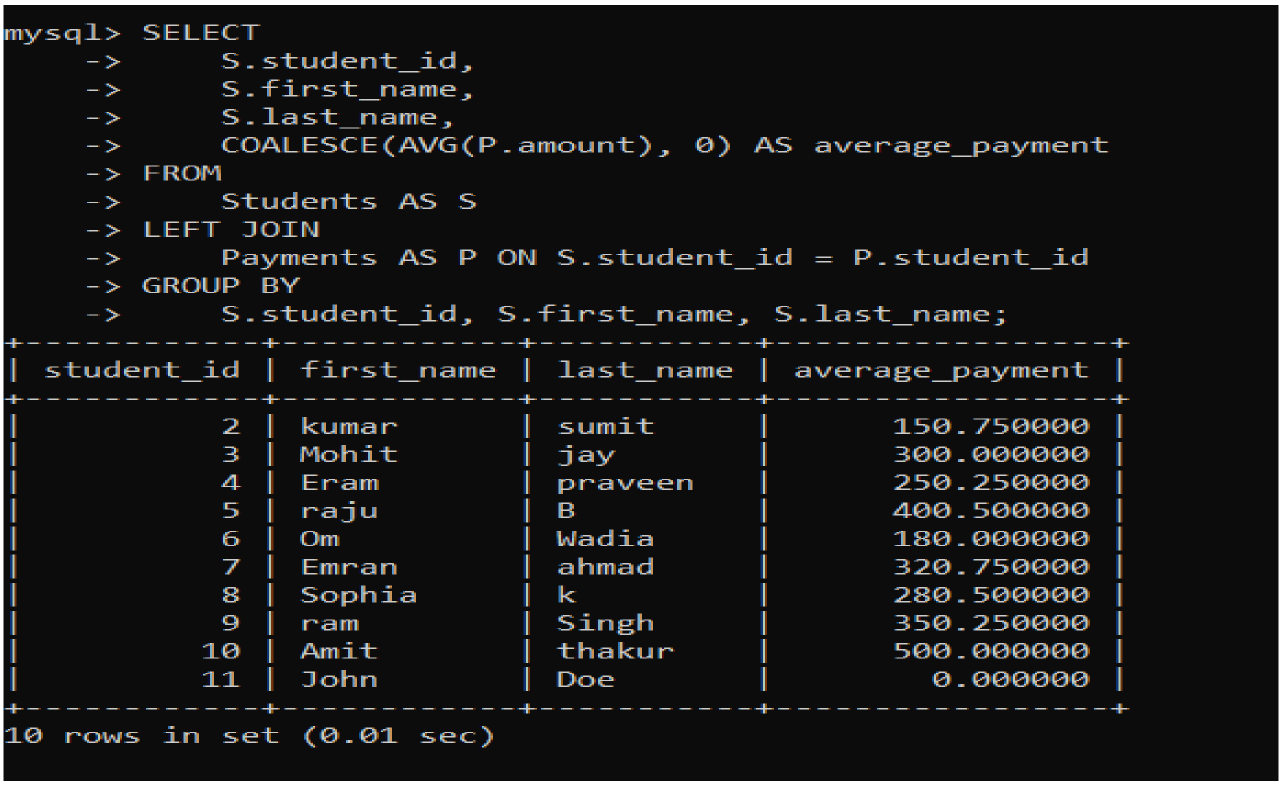
```
mysql> SELECT
    ->     S.student_id,
    ->     S.first_name,
    ->     S.last_name,
    ->     COALESCE(SUM(P.amount), 0) AS total_payments
    -> FROM
    ->     Students AS S
    -> LEFT JOIN
    ->     Payments AS P ON S.student_id = P.student_id
    -> GROUP BY
    ->     S.student_id, S.first_name, S.last_name;
+------------+------------+-----------+----------------+
| student_id | first_name | last_name | total_payments |
+------------+------------+-----------+----------------+
|          2 | kumar      | sumit     |         150.75 |
|          3 | Mohit      | jay       |         300.00 |
|          4 | Eram       | praveen   |         250.25 |
|          5 | raju       | B         |         400.50 |
|          6 | Om         | Wadia     |         180.00 |
|          7 | Emran      | ahmad     |         320.75 |
|          8 | Sophia     | k         |         280.50 |
|          9 | ram        | Singh     |         350.25 |
|         10 | Amit       | thakur    |         500.00 |
|         11 | John       | Doe       |           0.00 |
+------------+------------+-----------+----------------+
10 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_id,
    ->     C.course_name,
    ->     COUNT(E.student_id) AS student_count
    -> FROM
    ->     Courses AS C
    -> LEFT JOIN
    ->     Enrollments AS E ON C.course_id = E.course_id
    -> GROUP BY
    ->     C.course_id, C.course_name;
+-----------+-----------------+---------------+
| course_id | course_name     | student_count |
+-----------+-----------------+---------------+
|         1 | Mathematics     |             0 |
|         2 |  Science        |             1 |
|         3 | History         |             1 |
|         4 | Physics         |             1 |
|         5 | Chemistry       |             1 |
|         6 | Literature      |             1 |
|         7 | Computer        |             1 |
|         8 | Psychology      |             1 |
|         9 | Art History     |             1 |
|        10 | Data Structures |             1 |
+-----------+-----------------+---------------+
10 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,
    ->     S.last_name,
    ->     COALESCE(AVG(P.amount), 0) AS average_payment
    -> FROM
    ->     Students AS S
    -> LEFT JOIN
    ->     Payments AS P ON S.student_id = P.student_id
    -> GROUP BY
    ->     S.student_id, S.first_name, S.last_name;
+------------+------------+-----------+-----------------+
| student_id | first_name | last_name | average_payment |
+------------+------------+-----------+-----------------+
|          2 | kumar      | sumit     |      150.750000 |
|          3 | Mohit      | jay       |      300.000000 |
|          4 | Eram       | praveen   |      250.250000 |
|          5 | raju       | B         |      400.500000 |
|          6 | Om         | Wadia     |      180.000000 |
|          7 | Emran      | ahmad     |      320.750000 |
|          8 | Sophia     | k         |      280.500000 |
|          9 | ram        | Singh     |      350.250000 |
|         10 | Amit       | thakur    |      500.000000 |
|         11 | John       | Doe       |        0.000000 |
+------------+------------+-----------+-----------------+
10 rows in set (0.01 sec)
```