Student Information System (SIS) Instructions:

- Submitting assignments should be a single file or through git hub link shared with trainer and hexavarsity.
- Each assignment builds upon the previous one, and by the end, you will have a comprehensive application implemented in Java/C#/Python with a strong focus on SQL schema design, control flow statements, loops, arrays, collections, and database interaction.
- Follow object-oriented principles throughout the Java programming assignments. Use classes and objects to model real-world entities, encapsulate data and behavior, and ensure code reusability.
- Throw user defined exception from method and handle in the main method.
- The following Directory structure is to be followed in the application.
    - ○ **entity/model**
        - ✦ Create entity classes in this package. All entity class should not have any business logic. ○ **dao**
        - ✦ Create Service Provider interface/abstract class to showcase functionalities.
        - ✦ Create the implementation class for the above interface/abstract class with db interaction.
    - ○ **exception**
        - Create user defined exceptions in this package and handle exceptions whenever needed.
    - ○ **util**
        - ✦ Create a DBPropertyUtil class with a static function which takes property file name as parameter and returns connection string.
        - ✦ Create a DBConnUtil class which holds static method which takes connection string as parameter file and returns connection object. ○ **main**
        - ✦ Create a class MainModule and demonstrate the functionalities in a menu driven application.

In this assignment, you will work with a simplified Student Information System (SIS) database. The SIS database contains information about students, courses, and enrollments. Your task is to perform various SQL operations on this database to retrieve and manipulate data.

Database Tables:

The SIS database consists of the following tables:

1. Students

- student_id (Primary Key)
- first_name
- last_name
- date_of_birth
- email
- phone_number

2. **Courses**

- course_id (Primary Key)
- course_name
- credits
- teacher_id (Foreign Key)

3. **Enrollments**

- enrollment_id (Primary Key)
- student_id (Foreign Key)
- course_id (Foreign Key)
- enrollment_date

4. **Teacher**

- teacher_id (Primary Key)
- first_name
- last_name
- email

5. **Payments**

- payment_id (Primary Key)
- student_id (Foreign Key)
- amount
- payment_date

Task 1. Database Design:

1. Create the database named "SISDB"

   Query : create database StudentInformationSystem;

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

      Query: create table students(student_id int primary key auto_increment , first_name varchar(60) not null , last_name varchar(60) not null , date_of_birth date , email varchar(50) , phone_number bigint unique);

   b. Courses

      Query: create table courses(course_id int primary key auto_increment , course_name varchar(70) , credits int check (credits > 0) , teacher_id int, fore
      ign key (teacher_id) references teacher(teacher_id));

   c. Enrollments

      Query: create table enrollments (enrollment_id int primary key auto_incremen
      t , student_id int , course_id int , enrollment_date date , foreign key(stud
      ent_id) references students(student_id) on delete cascade, foreign key(cours
      e_id) references courses(course_id) on delete cascade);

d. Teacher

Query: create table teacher(teacher_id int primary key auto_increment , first_name varchar(60), last_name varchar(60), email varchar(60));

e. Payments

Query: create table payments(payment_id int primary key auto_increment ,student_id int, amount int, payment_date date , foreign key(student_id) references students(student_id) on delete cascade);

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

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

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

i. Students

Query: INSERT INTO students (first_name, last_name, date_of_birth, email, phone_number) VALUES
('Arun', 'Subramanian', '2001-02-15', 'arun.subramanian@email.com', 9876543211),
('Deepika', 'Ramachandran', '2000-06-25', 'deepika.ramachandran@email.com', 9876543212),
('Praveen', 'Venkatesan', '1999-10-10', 'praveen.venkatesan@email.com', 9876543213),
('Meena', 'Krishnan', '2002-03-30', 'meena.krishnan@email.com', 9876543214),
('Karthik', 'Srinivasan', '2000-07-08', 'karthik.srinivasan@email.com', 9876543215),
('Sowmya', 'Balasubramaniam', '2001-12-12', 'sowmya.balasubramaniam@email.com', 9876543216),
('Vignesh', 'Rajendran', '1998-05-20', 'vignesh.rajendran@email.com', 9876543217),
('Aishwarya', 'Thirumalai', '2003-09-05', 'aishwarya.thirumalai@email.com', 9876543218),
('Gokul', 'Panneerselvam', '1999-11-22', 'gokul.panneerselvam@email.com', 9876543219),
('Divya', 'Sundararajan', '2002-08-14', 'divya.sundararajan@email.com', 9876543220);

ii. Courses

Query: INSERT INTO courses (course_name, credits, teacher_id) VALUES
('Database Management Systems', 3, 1),
('Computer Networks', 4, 2),
('Artificial Intelligence', 3, 3),
('Software Engineering', 3, 4),
('Machine Learning', 4, 5),
('Cloud Computing', 3, 6),
('Cybersecurity Fundamentals', 3, 7),
('Operating Systems', 4, 8),
('Data Structures and Algorithms', 4, 9),
('Web Development', 3, 10);

iii. Enrollments

Query:INSERT INTO enrollments (student_id, course_id, enrollment_date) VALUES

```
(1, 3, '2025-01-10'),
(2, 5, '2025-02-15'),
(3, 1, '2025-03-05'),
(4, 7, '2025-01-22'),
(5, 2, '2025-02-28'),
(6, 6, '2025-03-10'),
(7, 4, '2025-01-18'),
(8, 9, '2025-02-20'),
(9, 8, '2025-03-15'),
(10, 10, '2025-01-25');
```

iv. Teacher

Query:
```
INSERT INTO teacher (first_name, last_name, email) VALUES
('Murugan', 'Srinivasan', 'murugan.srinivasan@email.com'),
('Lakshmi', 'Balakrishnan', 'lakshmi.balakrishnan@email.com'),
('Arun', 'Chidambaram', 'arun.chidambaram@email.com'),
('Revathi', 'Ramaswamy', 'revathi.ramaswamy@email.com'),
('Venkatesh', 'Subramaniam', 'venkatesh.subramaniam@email.com'),
('Meenakshi', 'Krishnamurthy', 'meenakshi.krishnamurthy@email.com'),
('Karthikeyan', 'Rajagopal', 'karthikeyan.rajagopal@email.com'),
('Sowmya', 'Palanisamy', 'sowmya.palanisamy@email.com'),
('Gopinath', 'Thiruvengadam', 'gopinath.thiruvengadam@email.com'),
('Priya', 'Vasudevan', 'priya.vasudevan@email.com');
```

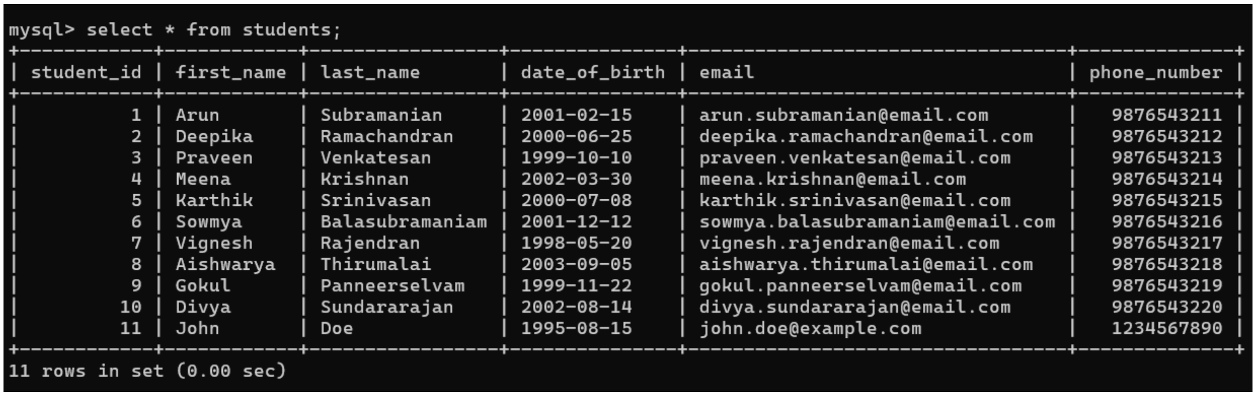
v. Payments

Query:
```
INSERT INTO payments (student_id, amount, payment_date) VALUES
(1, 5000, '2025-01-05'),
(2, 4500, '2025-02-10'),
(3, 6000, '2025-03-15'),
(4, 5200, '2025-01-20'),
(5, 5800, '2025-02-25'),
(6, 4900, '2025-03-05'),
(7, 5500, '2025-01-30'),
(8, 5300, '2025-02-12'),
(9, 6200, '2025-03-18'),
(10, 4700, '2025-01-08');
```

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

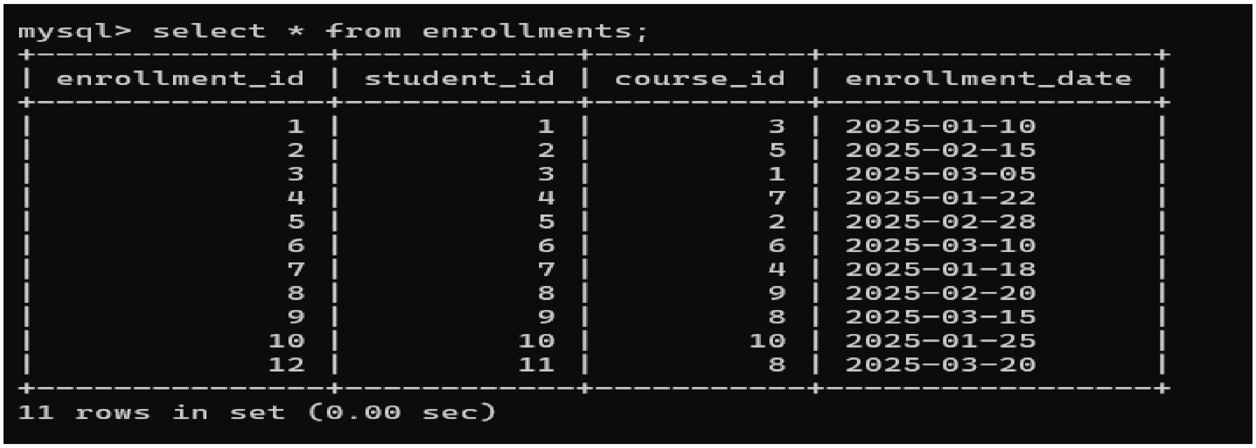
1. Write an SQL query to insert a new student into the "Students" table with the following details:
   a. First Name: John
   b. Last Name: Doe
   c. Date of Birth: 1995-08-15
   d. Email: john.doe@example.com
   e. Phone Number: 1234567890

Query: INSERT INTO students (first_name, last_name, date_of_birth, email, phone_number) VALUES ('John', 'Doe', '1995-08-15', 'john.doe@example.com', 1234567890);



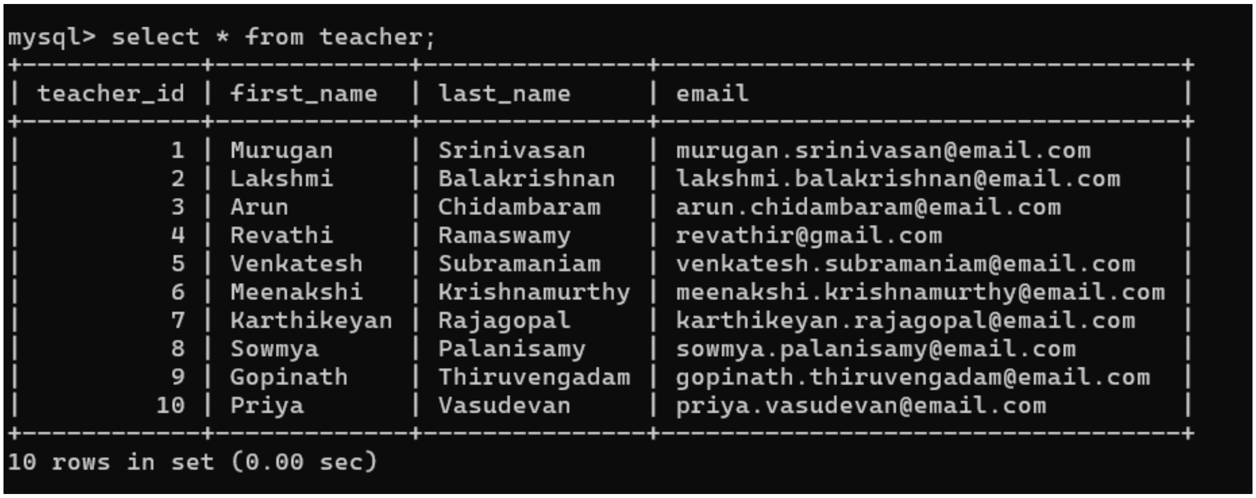
```
mysql> select * from students;
+------------+------------+-----------------+---------------+---------------------------------+--------------+
| student_id | first_name | last_name       | date_of_birth | email                           | phone_number |
+------------+------------+-----------------+---------------+---------------------------------+--------------+
|          1 | Arun       | Subramanian     | 2001-02-15    | arun.subramanian@email.com      |   9876543211 |
|          2 | Deepika    | Ramachandran    | 2000-06-25    | deepika.ramachandran@email.com  |   9876543212 |
|          3 | Praveen    | Venkatesan      | 1999-10-10    | praveen.venkatesan@email.com    |   9876543213 |
|          4 | Meena      | Krishnan        | 2002-03-30    | meena.krishnan@email.com        |   9876543214 |
|          5 | Karthik    | Srinivasan      | 2000-07-08    | karthik.srinivasan@email.com    |   9876543215 |
|          6 | Sowmya     | Balasubramaniam | 2001-12-12    | sowmya.balasubramaniam@email.com |   9876543216 |
|          7 | Vignesh    | Rajendran       | 1998-05-20    | vignesh.rajendran@email.com     |   9876543217 |
|          8 | Aishwarya  | Thirumalai      | 2003-09-05    | aishwarya.thirumalai@email.com  |   9876543218 |
|          9 | Gokul      | Panneerselvam   | 1999-11-22    | gokul.panneerselvam@email.com   |   9876543219 |
|         10 | Divya      | Sundararajan    | 2002-08-14    | divya.sundararajan@email.com    |   9876543220 |
|         11 | John       | Doe             | 1995-08-15    | john.doe@example.com            |   1234567890 |
+------------+------------+-----------------+---------------+---------------------------------+--------------+
11 rows in set (0.00 sec)
```

2. Write an SQL query to enroll a student in a course. Choose an existing student and course and insert a record into the "Enrollments" table with the enrollment date.
   Query: insert into enrollments(student_id,course_id,enrollment_date) values (11 , 8 , '2025-03-20');



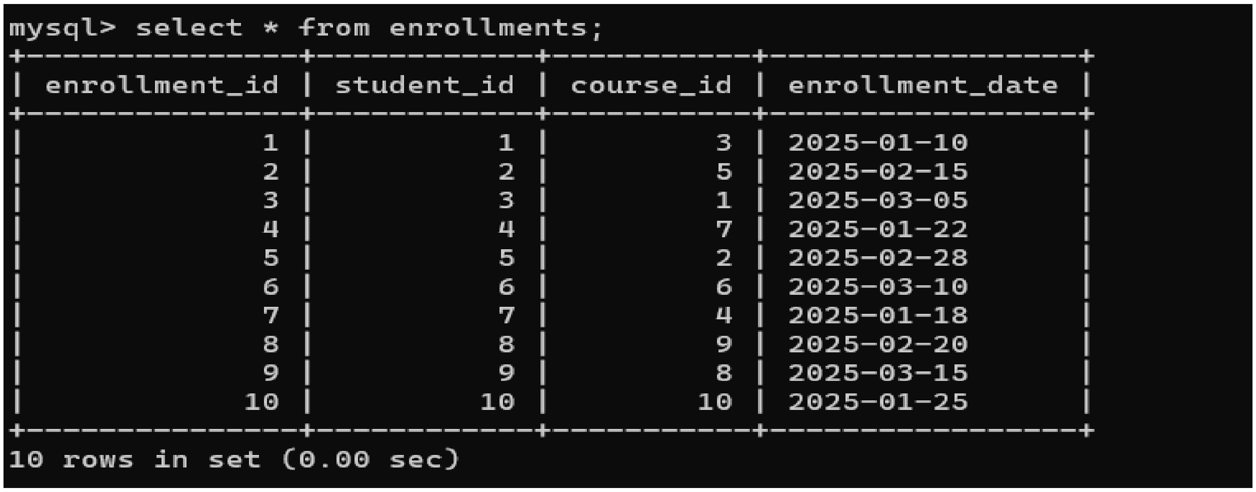
```
mysql> select * from enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|             1 |          1 |         3 | 2025-01-10      |
|             2 |          2 |         5 | 2025-02-15      |
|             3 |          3 |         1 | 2025-03-05      |
|             4 |          4 |         7 | 2025-01-22      |
|             5 |          5 |         2 | 2025-02-28      |
|             6 |          6 |         6 | 2025-03-10      |
|             7 |          7 |         4 | 2025-01-18      |
|             8 |          8 |         9 | 2025-02-20      |
|             9 |          9 |         8 | 2025-03-15      |
|            10 |         10 |        10 | 2025-01-25      |
|            12 |         11 |         8 | 2025-03-20      |
+---------------+------------+-----------+-----------------+
11 rows in set (0.00 sec)
```

3. Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.
   Query: update teacher set email="revathir@gmail.com" where first_name="revathi";



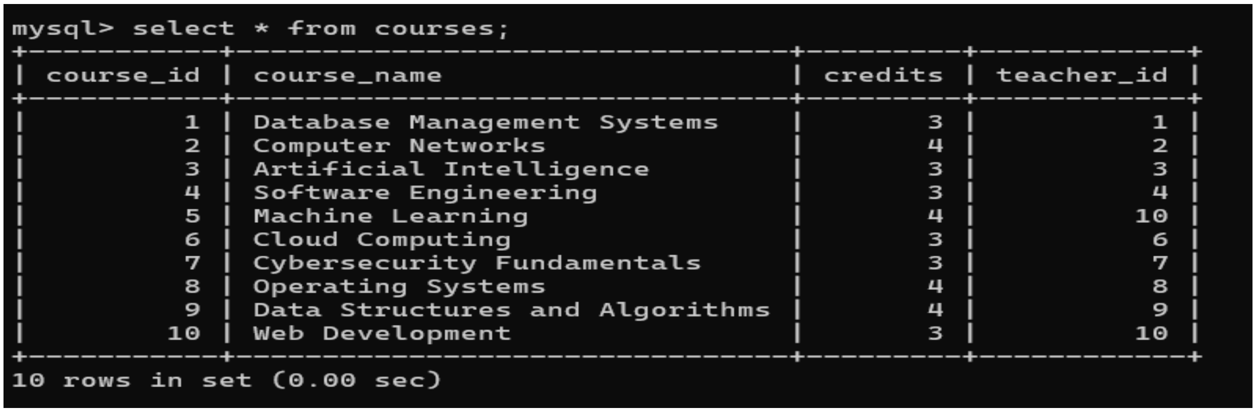
```
mysql> select * from teacher;
+------------+-------------+---------------+-----------------------------------+
| teacher_id | first_name  | last_name     | email                             |
+------------+-------------+---------------+-----------------------------------+
|          1 | Murugan     | Srinivasan    | murugan.srinivasan@email.com      |
|          2 | Lakshmi     | Balakrishnan  | lakshmi.balakrishnan@email.com    |
|          3 | Arun        | Chidambaram   | arun.chidambaram@email.com        |
|          4 | Revathi     | Ramaswamy     | revathir@gmail.com                |
|          5 | Venkatesh   | Subramaniam   | venkatesh.subramaniam@email.com   |
|          6 | Meenakshi   | Krishnamurthy | meenakshi.krishnamurthy@email.com |
|          7 | Karthikeyan | Rajagopal     | karthikeyan.rajagopal@email.com   |
|          8 | Sowmya      | Palanisamy    | sowmya.palanisamy@email.com       |
|          9 | Gopinath    | Thiruvengadam | gopinath.thiruvengadam@email.com  |
|         10 | Priya       | Vasudevan     | priya.vasudevan@email.com         |
+------------+-------------+---------------+-----------------------------------+
10 rows in set (0.00 sec)
```

4. Write an SQL query to delete a specific enrollment record from the "Enrollments" table. Select an enrollment record based on the student and course.
   Query: delete from enrollments where enrollment_id=12;



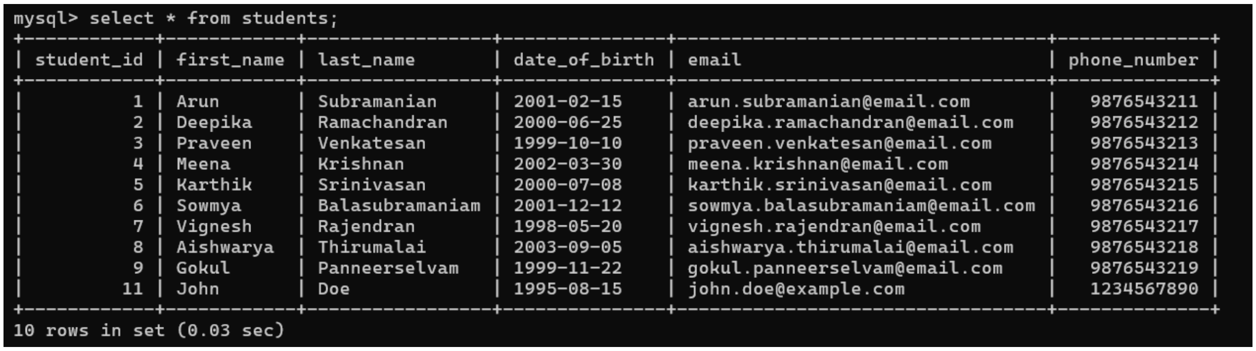
```
mysql> select * from enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|             1 |          1 |         3 | 2025-01-10      |
|             2 |          2 |         5 | 2025-02-15      |
|             3 |          3 |         1 | 2025-03-05      |
|             4 |          4 |         7 | 2025-01-22      |
|             5 |          5 |         2 | 2025-02-28      |
|             6 |          6 |         6 | 2025-03-10      |
|             7 |          7 |         4 | 2025-01-18      |
|             8 |          8 |         9 | 2025-02-20      |
|             9 |          9 |         8 | 2025-03-15      |
|            10 |         10 |        10 | 2025-01-25      |
+---------------+------------+-----------+-----------------+
10 rows in set (0.00 sec)
```

5. Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.
   Query: update courses set teacher_id=10 where course_name="Machine learning";

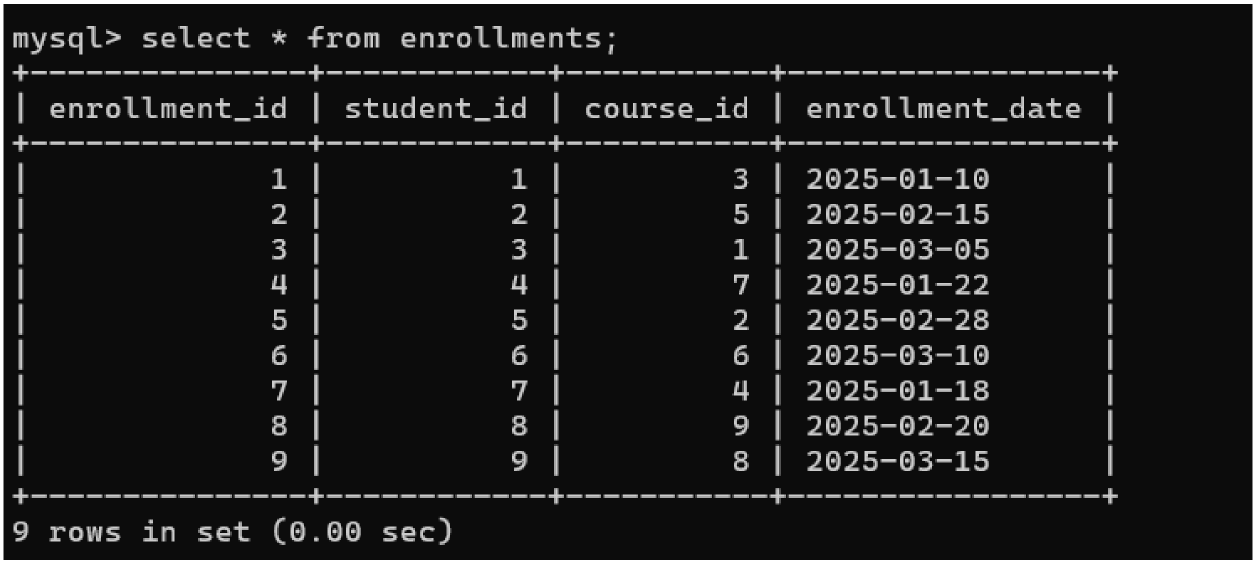


```
mysql> select * from courses;
+-----------+--------------------------------+---------+------------+
| course_id | course_name                    | credits | teacher_id |
+-----------+--------------------------------+---------+------------+
|         1 | Database Management Systems    |       3 |          1 |
|         2 | Computer Networks              |       4 |          2 |
|         3 | Artificial Intelligence        |       3 |          3 |
|         4 | Software Engineering           |       3 |          4 |
|         5 | Machine Learning               |       4 |         10 |
|         6 | Cloud Computing                |       3 |          6 |
|         7 | Cybersecurity Fundamentals     |       3 |          7 |
|         8 | Operating Systems              |       4 |          8 |
|         9 | Data Structures and Algorithms |       4 |          9 |
|        10 | Web Development                |       3 |         10 |
+-----------+--------------------------------+---------+------------+
10 rows in set (0.00 sec)
```

6. Delete a specific student from the "Students" table and remove all their enrollment records from the "Enrollments" table. Be sure to maintain referential integrity.
   Query: delete from students where student_id=10;

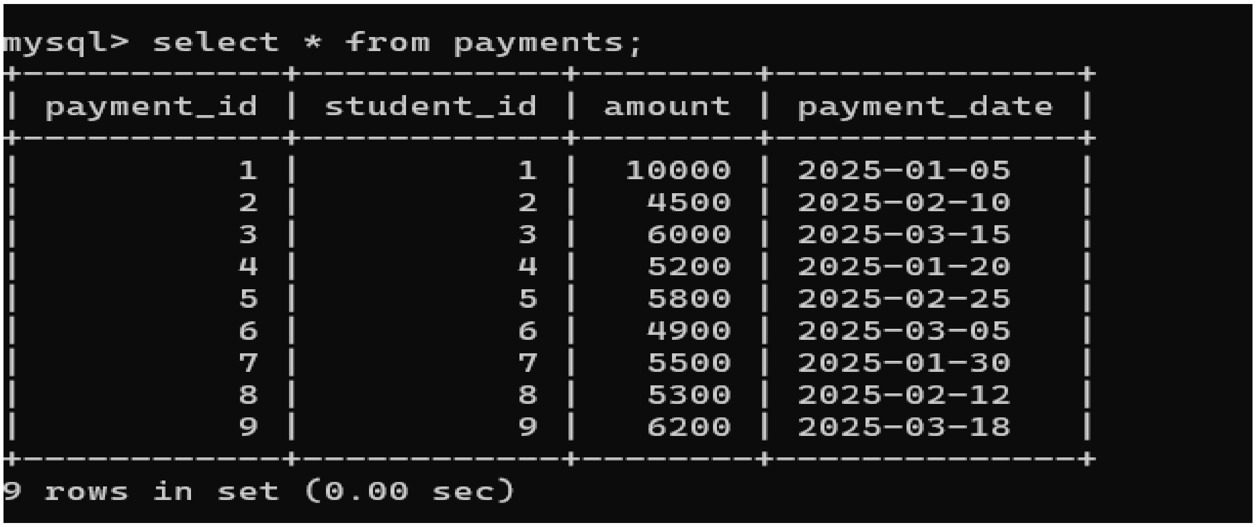


```
mysql> select * from students;
+------------+------------+-----------------+---------------+----------------------------------+--------------+
| student_id | first_name | last_name       | date_of_birth | email                            | phone_number |
+------------+------------+-----------------+---------------+----------------------------------+--------------+
|          1 | Arun       | Subramanian     | 2001-02-15    | arun.subramanian@email.com       |   9876543211 |
|          2 | Deepika    | Ramachandran    | 2000-06-25    | deepika.ramachandran@email.com   |   9876543212 |
|          3 | Praveen    | Venkatesan      | 1999-10-10    | praveen.venkatesan@email.com     |   9876543213 |
|          4 | Meena      | Krishnan        | 2002-03-30    | meena.krishnan@email.com         |   9876543214 |
|          5 | Karthik    | Srinivasan      | 2000-07-08    | karthik.srinivasan@email.com     |   9876543215 |
|          6 | Sowmya     | Balasubramaniam | 2001-12-12    | sowmya.balasubramaniam@email.com |   9876543216 |
|          7 | Vignesh    | Rajendran       | 1998-05-20    | vignesh.rajendran@email.com      |   9876543217 |
|          8 | Aishwarya  | Thirumalai      | 2003-09-05    | aishwarya.thirumalai@email.com   |   9876543218 |
|          9 | Gokul      | Panneerselvam   | 1999-11-22    | gokul.panneerselvam@email.com    |   9876543219 |
|         11 | John       | Doe             | 1995-08-15    | john.doe@example.com             |   1234567890 |
+------------+------------+-----------------+---------------+----------------------------------+--------------+
10 rows in set (0.03 sec)
```



```
mysql> select * from enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|             1 |          1 |         3 | 2025-01-10      |
|             2 |          2 |         5 | 2025-02-15      |
|             3 |          3 |         1 | 2025-03-05      |
|             4 |          4 |         7 | 2025-01-22      |
|             5 |          5 |         2 | 2025-02-28      |
|             6 |          6 |         6 | 2025-03-10      |
|             7 |          7 |         4 | 2025-01-18      |
|             8 |          8 |         9 | 2025-02-20      |
|             9 |          9 |         8 | 2025-03-15      |
+---------------+------------+-----------+-----------------+
9 rows in set (0.00 sec)
```

7. Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.
   Query: update payments set amount=10000 where payment_id=1;

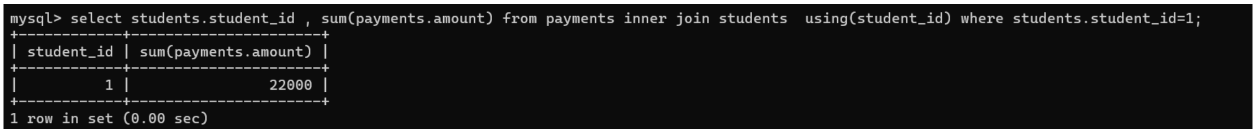


```
mysql> select * from payments;
+------------+------------+--------+--------------+
| payment_id | student_id | amount | payment_date |
+------------+------------+--------+--------------+
|          1 |          1 |  10000 | 2025-01-05   |
|          2 |          2 |   4500 | 2025-02-10   |
|          3 |          3 |   6000 | 2025-03-15   |
|          4 |          4 |   5200 | 2025-01-20   |
|          5 |          5 |   5800 | 2025-02-25   |
|          6 |          6 |   4900 | 2025-03-05   |
|          7 |          7 |   5500 | 2025-01-30   |
|          8 |          8 |   5300 | 2025-02-12   |
|          9 |          9 |   6200 | 2025-03-18   |
+------------+------------+--------+--------------+
9 rows in set (0.00 sec)
```

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

1. Write an SQL query to calculate the total payments made by a specific student. You will need to join the "Payments" table with the "Students" table based on the student's ID.

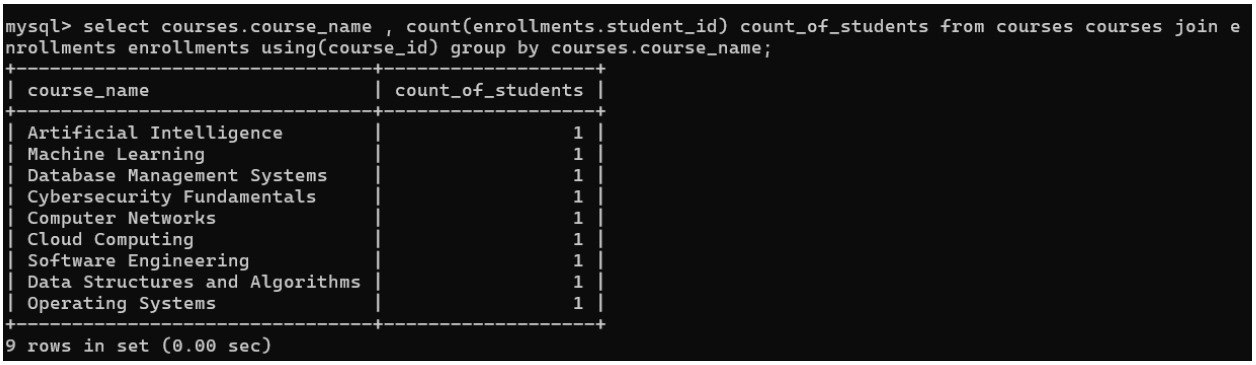
   Query: select students.student_id , sum(payments.amount) from payments inner join students using(student_id) where students.student_id=1;



```
mysql> select students.student_id , sum(payments.amount) from payments inner join students  using(student_id) where students.student_id=1;
+------------+----------------------+
| student_id | sum(payments.amount) |
+------------+----------------------+
|          1 |                22000 |
+------------+----------------------+
1 row in set (0.00 sec)
```

2. Write an SQL query to retrieve a list of courses along with the count of students enrolled in each course. Use a JOIN operation between the "Courses" table and the "Enrollments" table.

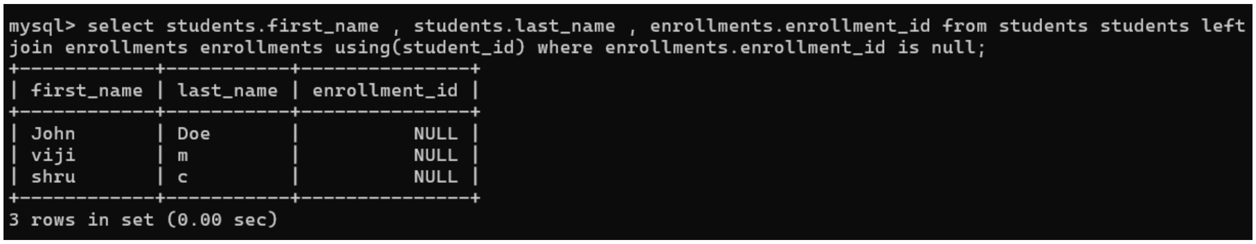
   Query: select courses.course_name , count(enrollments.student_id) count_of_students from courses courses join enrollments enrollments using(course_id) group by courses.course_name;



```
mysql> select courses.course_name , count(enrollments.student_id) count_of_students from courses courses join e
nrollments enrollments using(course_id) group by courses.course_name;
+-------------------------------+-------------------+
| course_name                   | count_of_students |
+-------------------------------+-------------------+
| Artificial Intelligence       |                 1 |
| Machine Learning              |                 1 |
| Database Management Systems   |                 1 |
| Cybersecurity Fundamentals    |                 1 |
| Computer Networks             |                 1 |
| Cloud Computing               |                 1 |
| Software Engineering          |                 1 |
| Data Structures and Algorithms |                1 |
| Operating Systems             |                 1 |
+-------------------------------+-------------------+
9 rows in set (0.00 sec)
```

3. Write an SQL query to find the names of students who have not enrolled in any course. Use a LEFT JOIN between the "Students" table and the "Enrollments" table to identify students without enrollments.

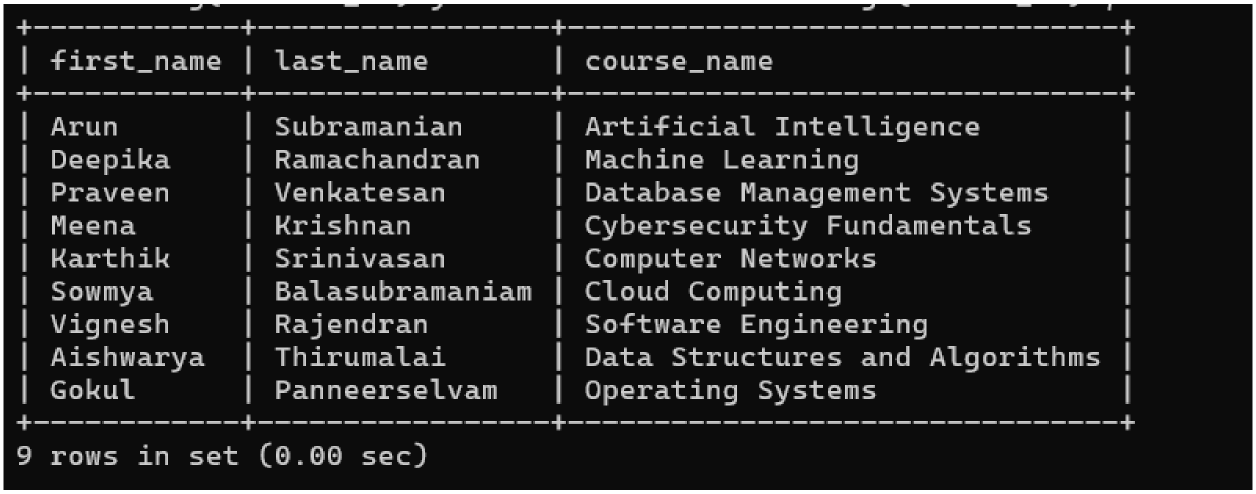
   Query: select students.first_name , students.last_name , enrollments.enrollment_id from students students left join enrollments enrollments using(student_id) where enrollments.enrollment_id is null;



```
mysql> select students.first_name , students.last_name , enrollments.enrollment_id from students students left
join enrollments enrollments using(student_id) where enrollments.enrollment_id is null;
+------------+-----------+---------------+
| first_name | last_name | enrollment_id |
+------------+-----------+---------------+
| John       | Doe       |          NULL |
| viji       | m         |          NULL |
| shru       | c         |          NULL |
+------------+-----------+---------------+
3 rows in set (0.00 sec)
```

4. Write an SQL query to retrieve the first name, last name of students, and the names of the courses they are enrolled in. Use JOIN operations between the "Students" table and the "Enrollments" and "Courses" tables.

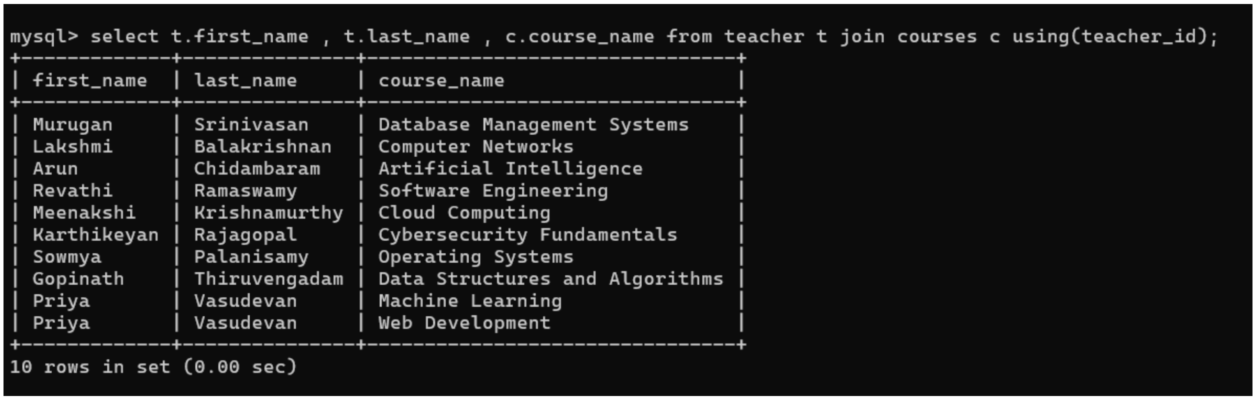
   Query: select students.first_name , students.last_name , courses.course_name from students students join enrollments using(student_id) join courses courses using (course_id) ;



```
+------------+-----------------+--------------------------------+
| first_name | last_name       | course_name                    |
+------------+-----------------+--------------------------------+
| Arun       | Subramanian     | Artificial Intelligence        |
| Deepika    | Ramachandran    | Machine Learning               |
| Praveen    | Venkatesan      | Database Management Systems    |
| Meena      | Krishnan        | Cybersecurity Fundamentals     |
| Karthik    | Srinivasan      | Computer Networks              |
| Sowmya     | Balasubramaniam | Cloud Computing                |
| Vignesh    | Rajendran       | Software Engineering           |
| Aishwarya  | Thirumalai      | Data Structures and Algorithms |
| Gokul      | Panneerselvam   | Operating Systems              |
+------------+-----------------+--------------------------------+
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.

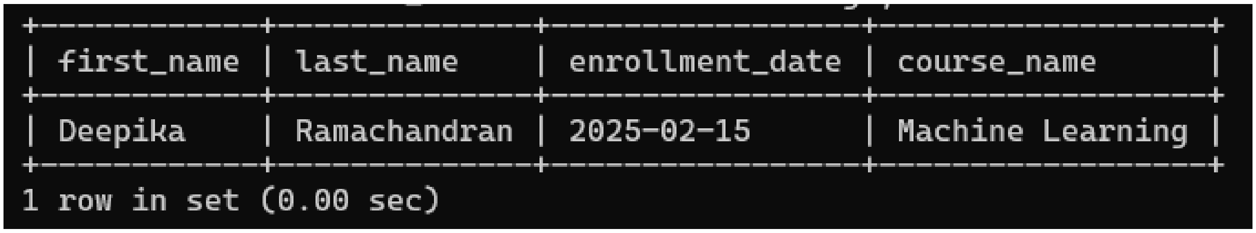
   Query: select t.first_name , t.last_name , c.course_name from teacher t join courses c using(teacher_id);



```
mysql> select t.first_name , t.last_name , c.course_name from teacher t join courses c using(teacher_id);
+-------------+---------------+--------------------------------+
| first_name  | last_name     | course_name                    |
+-------------+---------------+--------------------------------+
| Murugan     | Srinivasan    | Database Management Systems    |
| Lakshmi     | Balakrishnan  | Computer Networks              |
| Arun        | Chidambaram   | Artificial Intelligence        |
| Revathi     | Ramaswamy     | Software Engineering           |
| Meenakshi   | Krishnamurthy | Cloud Computing                |
| Karthikeyan | Rajagopal     | Cybersecurity Fundamentals     |
| Sowmya      | Palanisamy    | Operating Systems              |
| Gopinath    | Thiruvengadam | Data Structures and Algorithms |
| Priya       | Vasudevan     | Machine Learning               |
| Priya       | Vasudevan     | Web Development                |
+-------------+---------------+--------------------------------+
10 rows in set (0.00 sec)
```

6. Retrieve a list of students and their enrollment dates for a specific course. You'll need to join the "Students" table with the "Enrollments" and "Courses" tables.

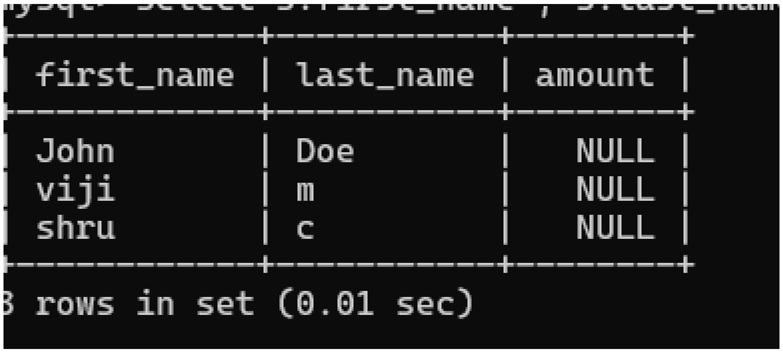
   Query : SELECT s.first_name, s.last_name, e.enrollment_date, c.course_name FROM students s JOIN enrollments e ON s.student_id = e.student_id JOIN courses c ON e.course_id = c.course_id WHERE c.course_name = 'Machine Learning';



```
+------------+--------------+-----------------+------------------+
| first_name | last_name    | enrollment_date | course_name      |
+------------+--------------+-----------------+------------------+
| Deepika    | Ramachandran | 2025-02-15      | Machine Learning |
+------------+--------------+-----------------+------------------+
1 row 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.

   Query: select s.first_name , s.last_name , p.amount from students s left join payments p using(student_id) where p.amount is null;



```
mysql> select s.first_name , s.last_nam
+------------+-----------+--------+
| first_name | last_name | amount |
+------------+-----------+--------+
| John       | Doe       |   NULL |
| viji       | m         |   NULL |
| shru       | c         |   NULL |
+------------+-----------+--------+
3 rows in set (0.01 sec)
```

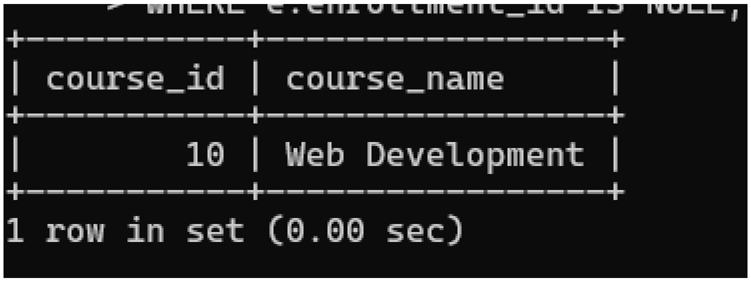
8. Write a query to identify courses that have no enrollments. You'll need to use a LEFT JOIN between the "Courses" table and the "Enrollments" table and filter for courses with NULL enrollment records.

   Query:SELECT c.course_id, c.course_name

   FROM courses c

   LEFT JOIN enrollments e ON c.course_id = e.course_id

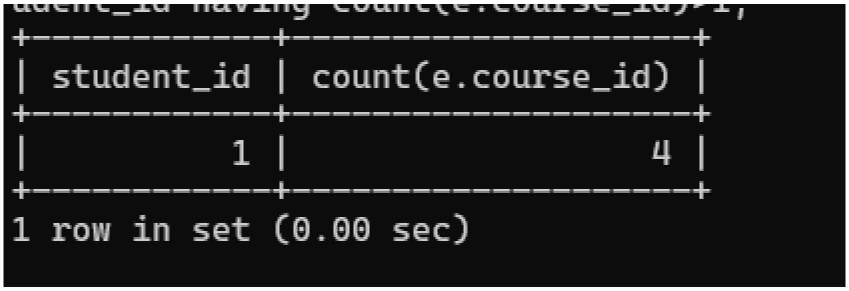
   WHERE e.enrollment_id IS NULL;



```
    -> WHERE e.enrollment_id IS NULL;
+-----------+-----------------+
| course_id | course_name     |
+-----------+-----------------+
|        10 | Web Development |
+-----------+-----------------+
1 row in set (0.00 sec)
```

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

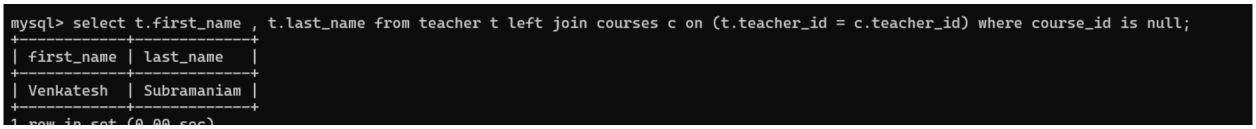
Query: select e.student_id , count(e.course_id) from enrollments e join enrollments e1 on (e.student_id = e1.student_id)group by e.student_id having count(e.course_id)>1;



```
udent_id having count(e.course_id)>1;
+------------+-------------------+
| student_id | count(e.course_id) |
+------------+-------------------+
|          1 |                 4 |
+------------+-------------------+
1 row in set (0.00 sec)
```

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

    Query: select t.first_name , t.last_name from teacher t left join courses c on (t.teacher_id = c.teacher_id) where course_id is null;



```
mysql> select t.first_name , t.last_name from teacher t left join courses c on (t.teacher_id = c.teacher_id) where course_id is null;
+------------+------------+
| first_name | last_name  |
+------------+------------+
| Venkatesh  | Subramaniam |
+------------+------------+
1 row in set (0.00 sec)
```

Task 4. Subquery and its type:

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

   Query: SELECT AVG(student_count) AS avg_students_per_course

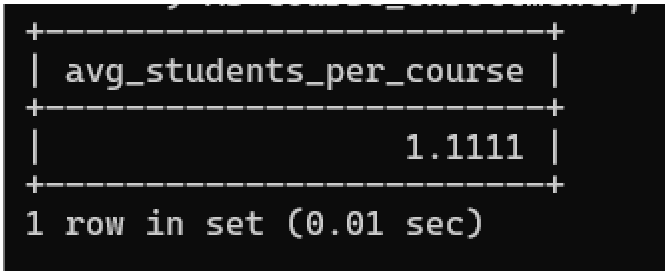
   FROM (

   SELECT course_id, COUNT(student_id) AS student_count

   FROM enrollments

   GROUP BY course_id

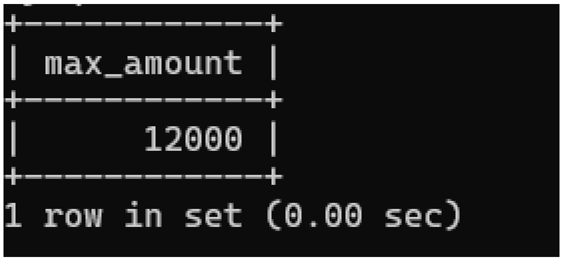
) AS course_enrollments;



```
    ) AS course_enrollments;
+-------------------------+
| avg_students_per_course |
+-------------------------+
|                  1.1111 |
+-------------------------+
1 row 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.

   Query: select max(amount) as max_amount from (select student_id , amount from payments) as maxx;



```
+------------+
| max_amount |
+------------+
|      12000 |
+------------+
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.

   Query : SELECT course_id, COUNT(student_id) AS enrollment_count

   FROM enrollments

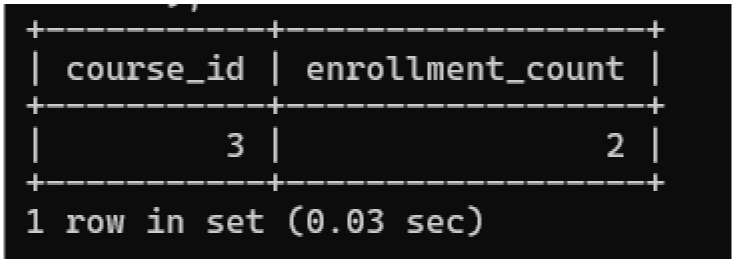
   GROUP BY course_id

   HAVING COUNT(student_id) = (

   SELECT MAX(enrollment_count)

   FROM (SELECT COUNT(student_id) AS enrollment_count FROM enrollments GROUP BY course_id) AS counts

   );



```
+-----------+------------------+
| course_id | enrollment_count |
+-----------+------------------+
|         3 |                2 |
+-----------+------------------+
1 row in set (0.03 sec)
```

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

   Query : SELECT teacher_id,

   (SELECT SUM(p.amount)

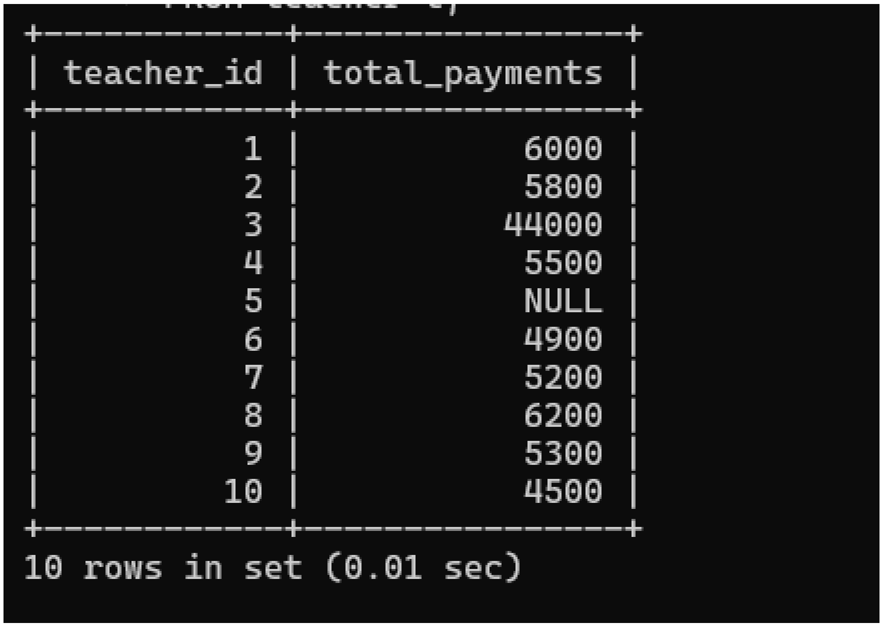
   FROM enrollments e

   JOIN payments p ON e.student_id = p.student_id

   WHERE e.course_id IN (SELECT course_id FROM courses c WHERE c.teacher_id = t.teacher_id)

) AS total_payments

   FROM teacher t;



```
    FROM teacher t)
+------------+----------------+
| teacher_id | total_payments |
+------------+----------------+
|          1 |           6000 |
|          2 |           5800 |
|          3 |          44000 |
|          4 |           5500 |
|          5 |           NULL |
|          6 |           4900 |
|          7 |           5200 |
|          8 |           6200 |
|          9 |           5300 |
|         10 |           4500 |
+------------+----------------+
10 rows in set (0.01 sec)
```

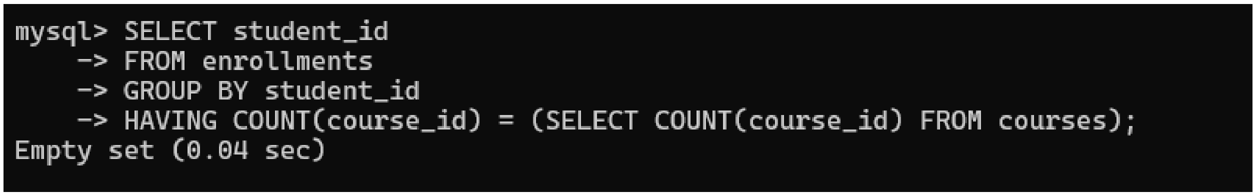
5. Identify students who are enrolled in all available courses. Use subqueries to compare a student's enrollments with the total number of courses.

   SELECT student_id

   FROM enrollments

   GROUP BY student_id

   HAVING COUNT(course_id) = (SELECT COUNT(course_id) FROM courses);



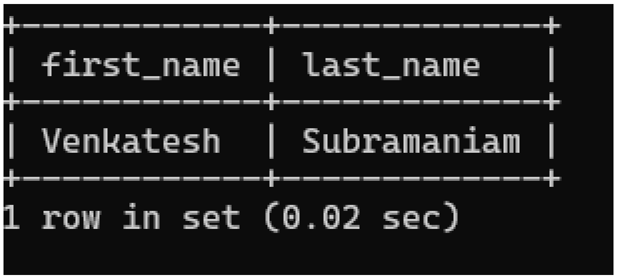
```
mysql> SELECT student_id
    -> FROM enrollments
    -> GROUP BY student_id
    -> HAVING COUNT(course_id) = (SELECT COUNT(course_id) FROM courses);
Empty set (0.04 sec)
```

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

   Query: SELECT first_name, last_name

   FROM teacher

   WHERE teacher_id NOT IN (SELECT DISTINCT teacher_id FROM courses);



```
+------------+-------------+
| first_name | last_name   |
+------------+-------------+
| Venkatesh  | Subramaniam |
+------------+-------------+
1 row in set (0.02 sec)
```

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

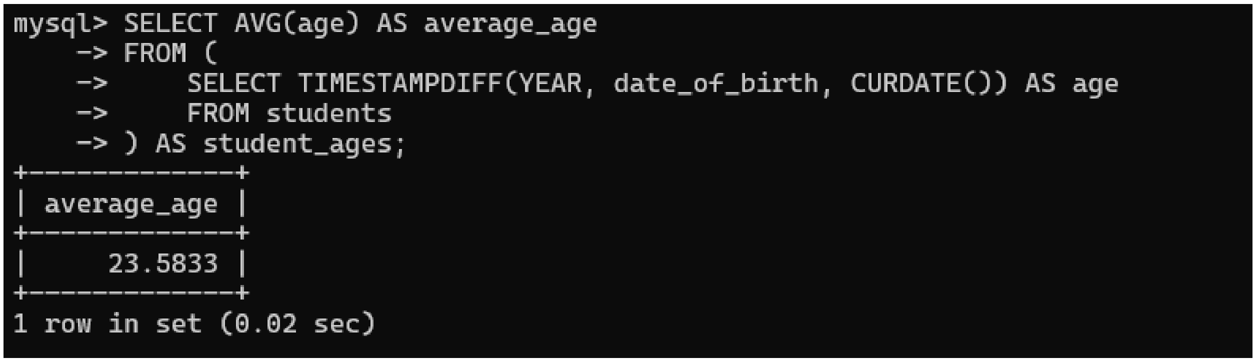
   Query: SELECT AVG(age) AS average_age

   FROM (

     SELECT TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age

     FROM students

) AS student_ages;



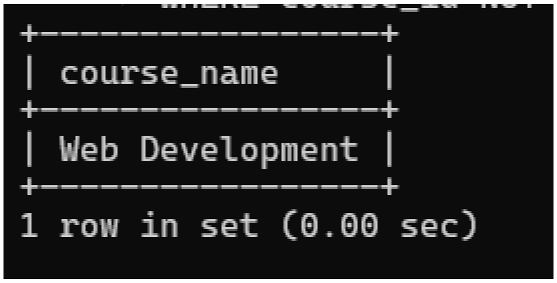
```
mysql> SELECT AVG(age) AS average_age
    -> FROM (
    ->     SELECT TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age
    ->     FROM students
    -> ) AS student_ages;
+-------------+
| average_age |
+-------------+
|     23.5833 |
+-------------+
1 row in set (0.02 sec)
```

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

   Query : SELECT course_name

   FROM courses

   WHERE course_id NOT IN (SELECT DISTINCT course_id FROM enrollments);



```
+-----------------+
| course_name     |
+-----------------+
| Web Development |
+-----------------+
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.

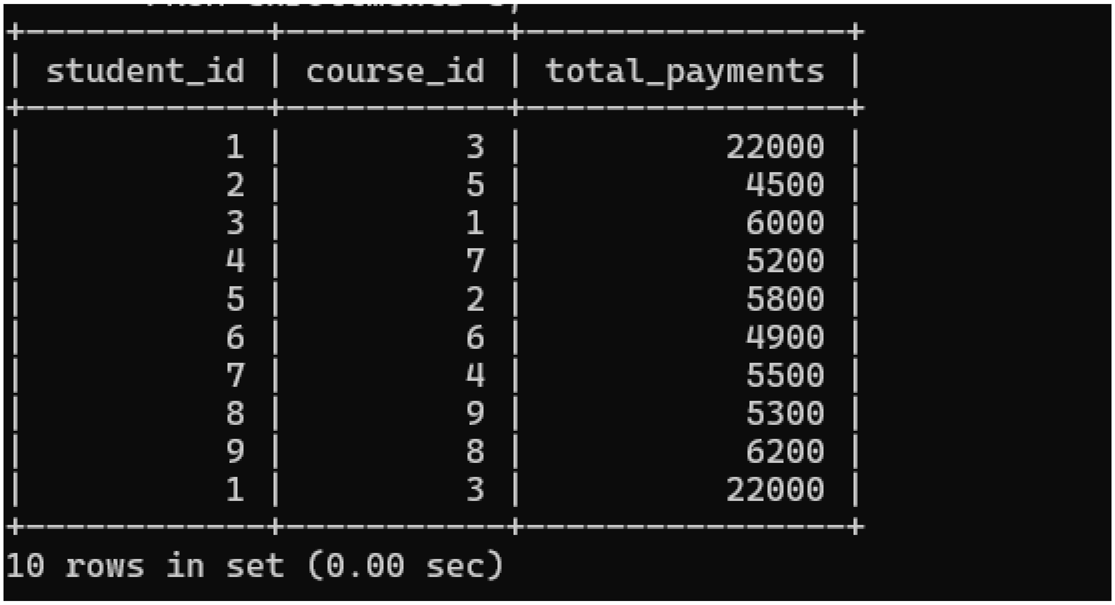
   Query: SELECT student_id, course_id,

       (SELECT SUM(amount)

        FROM payments p

        WHERE p.student_id = e.student_id) AS total_payments

   FROM enrollments e;



```
+------------+-----------+----------------+
| student_id | course_id | total_payments |
+------------+-----------+----------------+
|          1 |         3 |          22000 |
|          2 |         5 |           4500 |
|          3 |         1 |           6000 |
|          4 |         7 |           5200 |
|          5 |         2 |           5800 |
|          6 |         6 |           4900 |
|          7 |         4 |           5500 |
|          8 |         9 |           5300 |
|          9 |         8 |           6200 |
|          1 |         3 |          22000 |
+------------+-----------+----------------+
10 rows in set (0.00 sec)
```

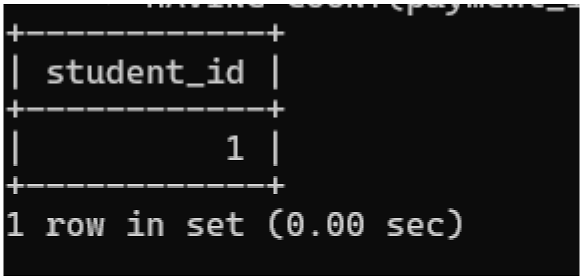
10. Identify students who have made more than one payment. Use subqueries and aggregate functions to count payments per student and filter for those with counts greater than one.

    Query: SELECT student_id

    FROM payments

    GROUP BY student_id

    HAVING COUNT(payment_id) > 1;



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

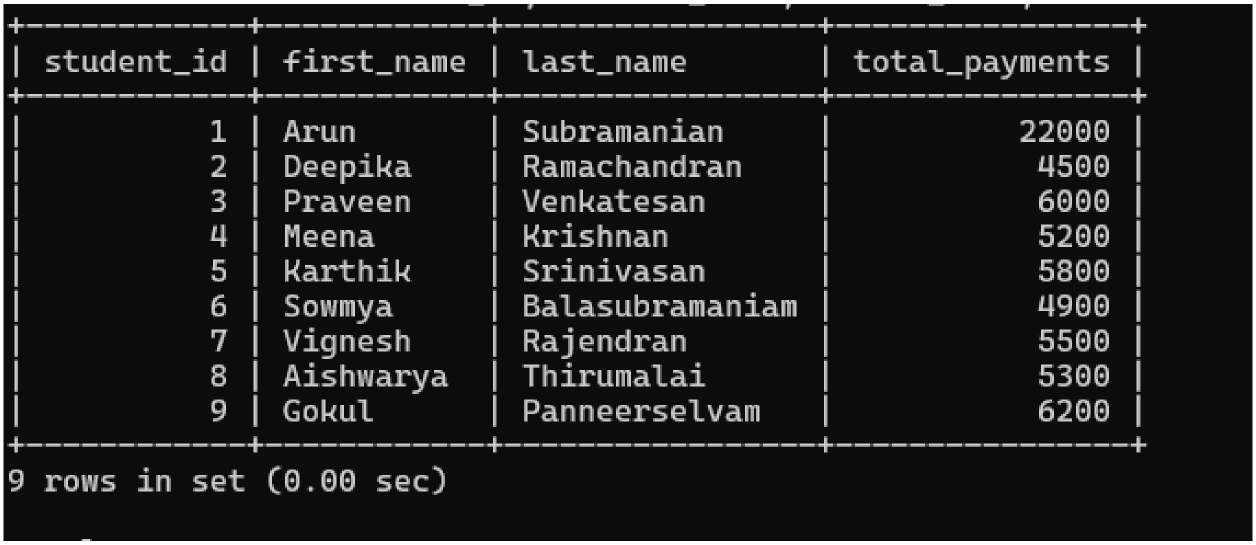
11. Write an SQL query to calculate the total payments made by each student. Join the "Students" table with the "Payments" table and use GROUP BY to calculate the sum of payments for each student.

    Query : SELECT s.student_id, s.first_name, s.last_name, SUM(p.amount) AS total_payments

    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 | first_name | last_name       | total_payments |
+------------+------------+-----------------+----------------+
|          1 | Arun       | Subramanian     |          22000 |
|          2 | Deepika    | Ramachandran    |           4500 |
|          3 | Praveen    | Venkatesan      |           6000 |
|          4 | Meena      | Krishnan        |           5200 |
|          5 | Karthik    | Srinivasan      |           5800 |
|          6 | Sowmya     | Balasubramaniam |           4900 |
|          7 | Vignesh    | Rajendran       |           5500 |
|          8 | Aishwarya  | Thirumalai      |           5300 |
|          9 | Gokul      | Panneerselvam   |           6200 |
+------------+------------+-----------------+----------------+
9 rows in set (0.00 sec)
```

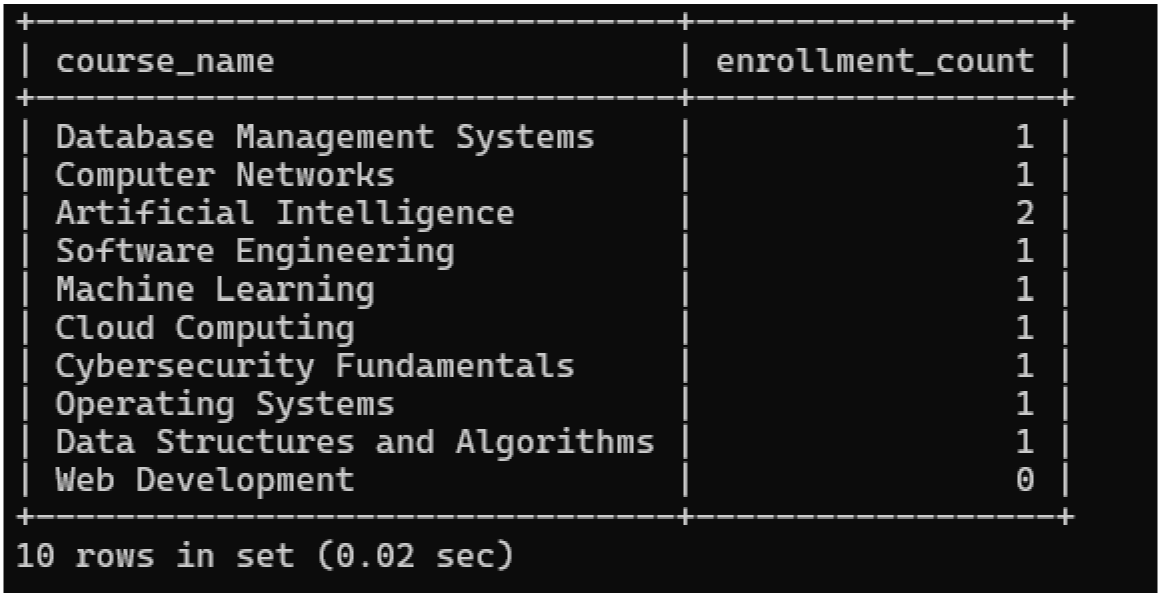
12. Retrieve a list of course names along with the count of students enrolled in each course. Use JOIN operations between the "Courses" table and the "Enrollments" table and GROUP BY to count enrollments.

    Query: SELECT c.course_name, COUNT(e.student_id) AS enrollment_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                    | enrollment_count |
+--------------------------------+------------------+
| Database Management Systems    |                1 |
| Computer Networks              |                1 |
| Artificial Intelligence        |                2 |
| Software Engineering           |                1 |
| Machine Learning               |                1 |
| Cloud Computing                |                1 |
| Cybersecurity Fundamentals     |                1 |
| Operating Systems              |                1 |
| Data Structures and Algorithms |                1 |
| Web Development                |                0 |
+--------------------------------+------------------+
10 rows in set (0.02 sec)
```

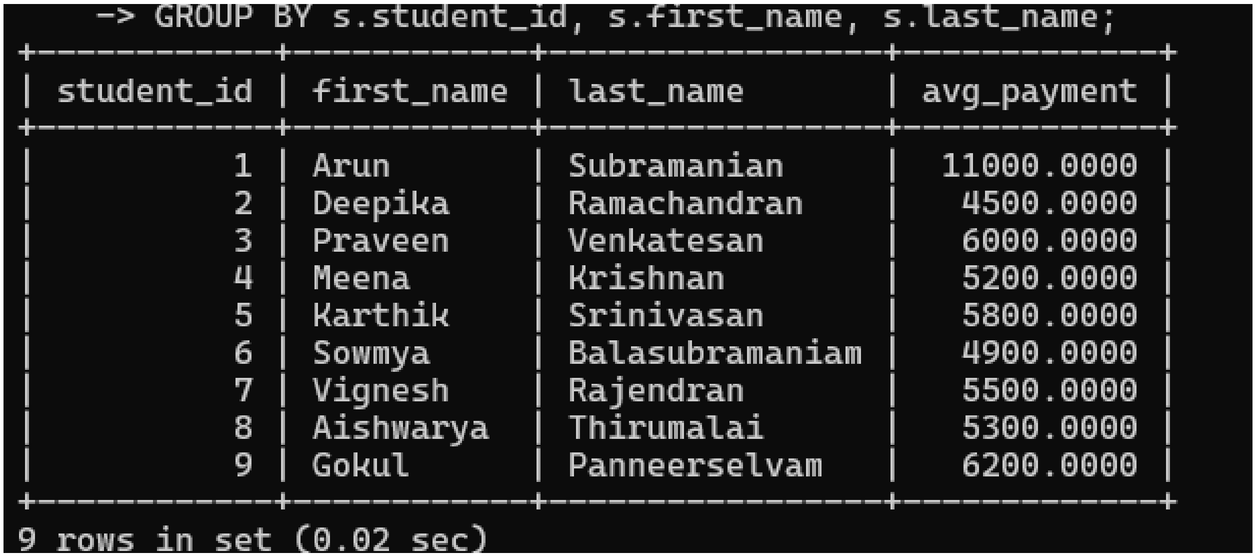
13. Calculate the average payment amount made by students. Use JOIN operations between the "Students" table and the "Payments" table and GROUP BY to calculate the average.

    Query: SELECT s.student_id, s.first_name, s.last_name, AVG(p.amount) AS avg_payment

    FROM students s

JOIN payments p ON s.student_id = p.student_id

GROUP BY s.student_id, s.first_name, s.last_name;



```
    -> GROUP BY s.student_id, s.first_name, s.last_name;
+------------+------------+-----------------+-------------+
| student_id | first_name | last_name       | avg_payment |
+------------+------------+-----------------+-------------+
|          1 | Arun       | Subramanian     |  11000.0000 |
|          2 | Deepika    | Ramachandran    |   4500.0000 |
|          3 | Praveen    | Venkatesan      |   6000.0000 |
|          4 | Meena      | Krishnan        |   5200.0000 |
|          5 | Karthik    | Srinivasan      |   5800.0000 |
|          6 | Sowmya     | Balasubramaniam |   4900.0000 |
|          7 | Vignesh    | Rajendran       |   5500.0000 |
|          8 | Aishwarya  | Thirumalai      |   5300.0000 |
|          9 | Gokul      | Panneerselvam   |   6200.0000 |
+------------+------------+-----------------+-------------+
9 rows in set (0.02 sec)
```