Assignment 2

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"

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

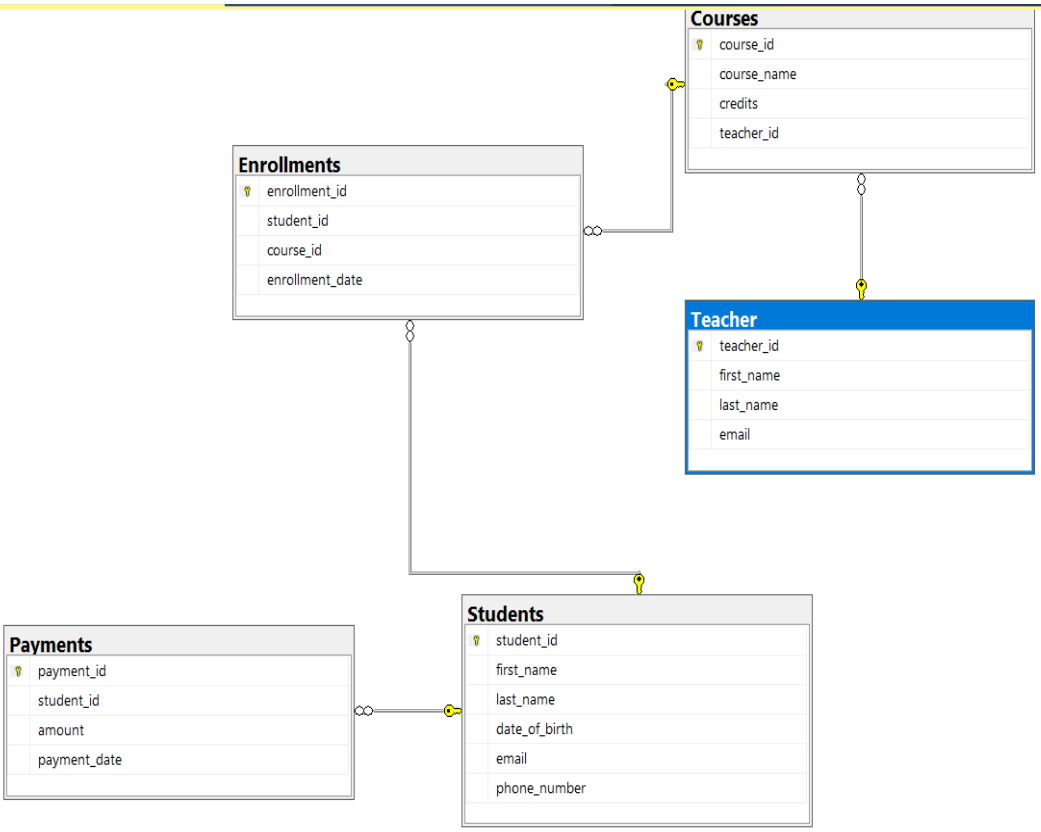
d. Teacher

e. Payments

```
CREATE DATABASE SISDB

CREATE TABLE Students(
student_id INT PRIMARY KEY,
first_name VARCHAR(30),
last_name VARCHAR(30),
date_of_birth DATE,
email VARCHAR(50),
phone_number VARCHAR(10)
);

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)
);

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)
);

CREATE TABLE Teacher (
    teacher_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100)
);

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)
);
```

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

ii. Courses

iii. Enrollments

iv. Teacher

v. Payments

```
INSERT INTO Students VALUES
(1, 'John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890'),
(2, 'Jane', 'Smith', '1998-05-22', 'jane.smith@example.com',
'9876543210'),
(3, 'Alice', 'Johnson', '1997-11-30', 'alice.johnson@example.com',
'5555555555'),
(4, 'Bob', 'Williams', '1996-04-10', 'bob.williams@example.com',
'7777777777'),
(5, 'Eva', 'Brown', '1999-09-18', 'eva.brown@example.com', '8888888888'),
(6, 'Charlie', 'Taylor', '1994-03-05', 'charlie.taylor@example.com',
'6666666666'),
(7, 'Sophia', 'Martin', '1993-01-25', 'sophia.martin@example.com',
'9999999999'),
(8, 'Daniel', 'Clark', '1992-07-12', 'daniel.clark@example.com',
'1111111111'),
(9, 'Olivia', 'Anderson', '1997-12-08', 'olivia.anderson@example.com',
'2222222222'),
(10, 'Michael', 'Davis', '1996-06-20', 'michael.davis@example.com',
'3333333333');

INSERT INTO Teacher VALUES
(1, 'Professor1', 'Johnson', 'professor1.johnson@example.com'),
```

```
(2, 'Professor2', 'Smith', 'professor2.smith@example.com'),
(3, 'Professor3', 'Brown', 'professor3.brown@example.com'),
(4, 'Professor4', 'Clark', 'professor4.clark@example.com'),
(5, 'Professor5', 'Taylor', 'professor5.taylor@example.com'),
(6, 'Professor6', 'Anderson', 'professor6.anderson@example.com'),
(7, 'Professor7', 'Martin', 'professor7.martin@example.com'),
(8, 'Professor8', 'Williams', 'professor8.williams@example.com'),
(9, 'Professor9', 'Davis', 'professor9.davis@example.com'),
(10, 'Professor10', 'White', 'professor10.white@example.com');

INSERT INTO Courses VALUES
(1, 'Mathematics', 3, 1),
(2, 'Physics', 4, 2),
(3, 'Chemistry', 3, 2),
(4, 'History', 3, 3),
(5, 'Computer Science', 4, 4),
(6, 'English Literature', 3, 4),
(7, 'Biology', 4, 5),
(8, 'Art', 2, 6),
(9, 'Economics', 3, 7),
(10, 'Psychology', 3, 8),
(11, 'Algebra', 9, 9),
(12, 'Geography', 2, 10);


INSERT INTO Enrollments VALUES
(101, 1, 1, '2023-01-15'),
(102, 2, 2, '2023-01-16'),
(103, 3, 2, '2023-01-17'),
(104, 4, 2, '2023-01-18'),
(105, 5, 3, '2023-01-19'),
(106, 6, 3, '2023-01-20'),
(107, 7, 4, '2023-01-21'),
(108, 8, 4, '2023-01-22'),
(109, 9, 5, '2023-01-23'),
(110, 10, 5, '2023-01-24');

INSERT INTO Payments VALUES
(1, 1, 100.00, '2023-02-01'),
(2, 2, 120.00, '2023-02-02'),
(3, 3, 90.00, '2023-02-03'),
(4, 4, 110.00, '2023-02-04'),
(5, 5, 80.00, '2023-02-05'),
(6, 6, 130.00, '2023-02-06'),
(7, 7, 95.00, '2023-02-07'),
(8, 8, 105.00, '2023-02-08'),
(9, 9, 75.00, '2023-02-09'),
(10, 10, 85.00, '2023-02-10');
```

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

```
INSERT INTO Students VALUES (11, 'John', 'Doe', '1995-08-15',
'john.doe@example.com', '1234567890');
```

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.

```
INSERT INTO Enrollments VALUES (1, 1, 2, '2023-02-01');
```

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

```
UPDATE Teacher
SET email = 'new.email@example.com'
WHERE teacher_id = 1;
```

	teacher_id	first_name	last_name	email
1	1	Professor1	Johnson	new.email@example.com
2	2	Professor2	Smith	professor2.smith@example.com
3	3	Professor3	Brown	professor3.brown@example.com
4	4	Professor4	Clark	professor4.clark@example.com
5	5	Professor5	Taylor	professor5.taylor@example.com
6	6	Professor6	Anderson	professor6.anderson@example.com
7	7	Professor7	Martin	professor7.martin@example.com
8	8	Professor8	Williams	professor8.williams@example.com
9	9	Professor9	Davis	professor9.davis@example.com
10	10	Professor10	White	professor10.white@example.com

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.

```
DELETE FROM Enrollments
WHERE enrollment_id=110 AND course_id=5;
```

	enrollment_id	student_id	course_id	enrollment_date
1	1	1	2	2023-02-01
2	101	1	1	2023-01-15
3	102	2	2	2023-01-16
4	103	3	2	2023-01-17
5	104	4	2	2023-01-18
6	105	5	3	2023-01-19
7	106	6	3	2023-01-20
8	107	7	4	2023-01-21
9	108	8	4	2023-01-22
10	109	9	5	2023-01-23

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

```
UPDATE Courses
SET teacher_id = 5 WHERE course_id = 11;
```

	course_id	course_name	credits	teacher_id
1	1	Mathematics	3	1
2	2	Physics	4	2
3	3	Chemistry	3	2
4	4	History	3	3
5	5	Computer Science	4	4
6	6	English Literature	3	4
7	7	Biology	4	5
8	8	Art	2	6
9	9	Economics	3	7
10	10	Psychology	3	8
11	11	Algebra	9	5
12	12	Geography	2	10

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.

```
DELETE FROM Enrollments WHERE student_id = 1;

-- Step 1: Delete related records in the Payments table
DELETE FROM Payments WHERE student_id = 1;

-- Step 2: Delete the student from the Students table
DELETE FROM Students WHERE student_id = 1;
```

	course_id	course_name	credits	teacher_id
1	1	Mathematics	3	1
2	2	Physics	4	2
3	3	Chemistry	3	2
4	4	History	3	3
5	5	Computer Science	4	4
6	6	English Literature	3	4
7	7	Biology	4	5
8	8	Art	2	6
9	9	Economics	3	7
10	10	Psychology	3	8
11	11	Algebra	9	5
12	12	Geography	2	10

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

```
UPDATE Payments
SET amount = 300.00
WHERE payment_id=5;
```

	payment_id	student_id	amount	payment_date
1	2	2	120.00	2023-02-02
2	3	3	90.00	2023-02-03
3	4	4	110.00	2023-02-04
4	5	5	300.00	2023-02-05
5	6	6	130.00	2023-02-06
6	7	7	95.00	2023-02-07
7	8	8	105.00	2023-02-08
8	9	9	75.00	2023-02-09
9	10	10	85.00	2023-02-10

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.

```
SELECT s.first_name, s.last_name, SUM(p.amount) AS Total_payments FROM
Students s
JOIN Payments p ON p.student_id = s.student_id
WHERE p.student_id=4
GROUP BY s.first_name,s.last_name;
```

	first_name	last_name	Total_payments
1	Bob	Williams	110.00

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.

```
SELECT C.course_id, C.course_name,
COUNT(E.student_id) AS enrolled_students_count
FROM Courses C
LEFT JOIN
Enrollments E ON C.course_id = E.course_id
GROUP BY C.course_id, C.course_name;
```

	course_id	course_name	enrolled_students_count
1	1	Mathematics	0
2	2	Physics	3
3	3	Chemistry	2
4	4	History	2
5	5	Computer Science	1
6	6	English Literature	0
7	7	Biology	0
8	8	Art	0
9	9	Economics	0
10	10	Psychology	0
11	11	Algebra	0
12	12	Geography	0

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.

```
SELECT s.student_id,s.first_name, s.last_name
FROM Students s
LEFT JOIN Enrollments e
ON s.student_id = e.student_id
WHERE e.student_id IS NULL;
```

	student_id	first_name	last_name
1	10	Michael	Davis
2	11	Rocky	Miller

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.

```
SELECT s.first_name,s.last_name, c.course_name
FROM Students s
JOIN Enrollments e
ON s.student_id = e.student_id
JOIN Courses c ON c.course_id = e.course_id;
```

	first_name	last_name	course_name
1	Jane	Smith	Physics
2	Alice	Johnson	Physics
3	Bob	Williams	Physics
4	Eva	Brown	Chemistry
5	Charlie	Taylor	Chemistry
6	Sophia	Martin	History
7	Daniel	Clark	History
8	Olivia	Anderson	Computer Science

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.

```
SELECT t.teacher_id,c.course_id,t.last_name,c.course_name
FROM Teacher t
JOIN Courses c ON t.teacher_id = c.teacher_id;
```

	teacher_id	course_id	last_name	course_name
1	1	1	Johnson	Mathematics
2	2	2	Smith	Physics
3	2	3	Smith	Chemistry
4	3	4	Brown	History
5	4	5	Clark	Computer Science
6	4	6	Clark	English Literature
7	5	7	Taylor	Biology
8	6	8	Anderson	Art
9	7	9	Martin	Economics
10	8	10	Williams	Psychology
11	5	11	Taylor	Algebra
12	10	12	White	Geography

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.

```
SELECT
s.student_id,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 c.course_id = e.course_id;
```

	student_id	first_name	last_name	enrollment_date	course_name
1	2	Jane	Smith	2023-01-16	Physics
2	3	Alice	Johnson	2023-01-17	Physics
3	4	Bob	Williams	2023-01-18	Physics
4	5	Eva	Brown	2023-01-19	Chemistry
5	6	Charlie	Taylor	2023-01-20	Chemistry
6	7	Sophia	Martin	2023-01-21	History
7	8	Daniel	Clark	2023-01-22	History
8	9	Olivia	Anderson	2023-01-23	Computer Science

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.

```
SELECT s.first_name, s.last_name
FROM Students s
LEFT JOIN
Payments p ON s.student_id = p.student_id
WHERE p.amount is NULL;
```

	first_name	last_name
1	Rocky	Miller

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.

```
SELECT c.course_id,c.course_name
FROM Courses c
LEFT JOIN
```

```
Enrollments e ON c.course_id = e.course_id
WHERE e.course_id is NULL;
```

	course_id	course_name
1	1	Mathematics
2	6	English Literature
3	7	Biology
4	8	Art
5	9	Economics
6	10	Psychology
7	11	Algebra
8	12	Geography

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.

```
SELECT
    E.student_id,
    S.first_name,
    S.last_name
FROM
    Enrollments E
JOIN
    Students S ON E.student_id = S.student_id
GROUP BY
    E.student_id, S.first_name, S.last_name
HAVING
    COUNT(DISTINCT E.course_id) > 1;
```

	student_id	first_name	last_name
1	3	Alice	Johnson
2	7	Sophia	Martin
3	9	Olivia	Anderson

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.

```
SELECT t.teacher_id, t.first_name,t.last_name
FROM Teacher t
LEFT JOIN
Courses c ON t.teacher_id = c.teacher_id
WHERE c.teacher_id is NULL;
```

	teacher_id	first_name	last_name
1	9	Professor9	Davis

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.

```
SELECT course_name, AVG(student_count) AS avg_students_enrolled
FROM (
    SELECT c.course_name, COUNT(e.student_id) AS student_count
    FROM Courses c
    LEFT JOIN Enrollments e ON c.course_id = e.course_id
    GROUP BY c.course_name
) AS CourseStudentCount
GROUP BY course_name;
```

	course_name	avg_students_enrolled
1	Algebra	0
2	Art	0
3	Biology	0
4	Chemistry	2
5	Computer Science	2
6	Economics	0
7	English Literature	0
8	Geography	0
9	History	3
10	Mathematics	0
11	Physics	4
12	Psychology	0

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.

```
SELECT s.student_id, s.first_name,s.last_name, p.amount
FROM Students s
JOIN
Payments p ON s.student_id = p.student_id
WHERE p.amount = (
SELECT MAX(p.amount) FROM Payments p
)
```

	student_id	first_name	last_name	amount
1	5	Eva	Brown	300.00

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

```
 SELECT
    C.course_id,
    C.course_name,
    C.credits,
    C.teacher_id,
    COUNT(E.enrollment_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, C.credits, C.teacher_id
HAVING
    COUNT(E.enrollment_id) = (
        SELECT
            MAX(enrollment_count)
        FROM (
            SELECT
                course_id,
                COUNT(enrollment_id) AS enrollment_count
            FROM
                Enrollments
            GROUP BY
                course_id
        ) AS CourseEnrollmentCounts
    );
```

	course_id	course_name	credits	teacher_id	enrollment_count
1	2	Physics	4	2	4

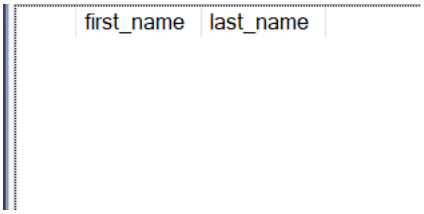
4. Calculate the total payments made to courses taught by each teacher. Use subqueries to sum payments for each teacher's courses.

```
SELECT
    T.teacher_id,
    T.first_name,
    T.last_name,
    SUM(P.amount) AS total_payments
FROM
    Teacher T
JOIN
    Courses C ON T.teacher_id = C.teacher_id
LEFT JOIN
    Enrollments E ON C.course_id = E.course_id
LEFT JOIN
    Payments P ON E.student_id = P.student_id
GROUP BY T.teacher_id, T.first_name, T.last_name;
```

	teacher_id	first_name	last_name	total_payments
1	1	Professor1	Johnson	NULL
2	2	Professor2	Smith	845.00
3	3	Professor3	Brown	275.00
4	4	Professor4	Clark	165.00
5	5	Professor5	Taylor	NULL
6	6	Professor6	Anderson	NULL
7	7	Professor7	Martin	NULL
8	8	Professor8	Williams	NULL
9	10	Professo...	White	NULL

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 s.first_name, s.last_name
FROM Students s
WHERE (SELECT COUNT(DISTINCT e.course_id)
FROM Enrollments e
WHERE e.student_id = s.student_id) = (
SELECT COUNT(DISTINCT course_id) FROM Courses);
```



first_name	last_name

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

```
SELECT t.teacher_id,t.first_name,t.last_name
FROM Teacher t
WHERE t.teacher_id NOT IN
(SELECT DISTINCT c.teacher_id FROM Courses c)
```

	teacher_id	first_name	last_name
1	9	Professor9	Davis

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

```
SELECT
    AVG(age) AS average_age
FROM (
    SELECT
        DATEDIFF(YEAR, date_of_birth, GETDATE()) AS age
    FROM
        Students
) AS StudentAges;
```

Results Messages

	average_age
1	27

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

```
SELECT c.course_id, c.course_name
FROM Courses c
```

```
WHERE c.course_id NOT IN (
SELECT DISTINCT e.course_id FROM Enrollments e)
```

	course_id	course_name
1	1	Mathematics
2	6	English Literature
3	7	Biology
4	8	Art
5	9	Economics
6	10	Psychology
7	11	Algebra
8	12	Geography

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

```
SELECT
    E.student_id,
    C.course_id,
    SUM(P.amount) AS total_payments
FROM
    Enrollments E
JOIN
    Courses C ON E.course_id = C.course_id
LEFT JOIN
    Payments P ON E.student_id = P.student_id
GROUP BY
    E.student_id, C.course_id;
```

	student_id	course_id	total_payments
1	2	2	120.00
2	3	2	90.00
3	4	2	110.00
4	7	2	95.00
5	5	3	300.00
6	6	3	130.00
7	7	4	95.00
8	8	4	105.00
9	9	4	75.00
10	3	5	90.00
11	9	5	75.00

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.

```
SELECT
    student_id
FROM
    Payments
GROUP BY
    student_id
HAVING
```

```
COUNT(payment_id) > 1;
```

student_id

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.

```
SELECT s.student_id,s.first_name,s.last_name, SUM(p.amount) AS
Total_payments
FROM Students s
LEFT 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	2	Jane	Smith	120.00
2	3	Alice	Johnson	90.00
3	4	Bob	Williams	110.00
4	5	Eva	Brown	300.00
5	6	Charlie	Taylor	130.00
6	7	Sophia	Martin	95.00
7	8	Daniel	Clark	105.00
8	9	Olivia	Anderson	75.00
9	10	Michael	Davis	85.00
10	11	Rocky	Miller	NULL

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.

```
SELECT c.course_id,c.course_name, COUNT(e.student_id) AS Student_Count
FROM Courses c
LEFT JOIN Enrollments e ON c.course_id=e.course_id
GROUP BY c.course_id,c.course_name;
```

	course_id	course_name	Student_Count
1	1	Mathematics	0
2	2	Physics	4
3	3	Chemistry	2
4	4	History	3
5	5	Computer Science	2
6	6	English Literature	0
7	7	Biology	0
8	8	Art	0
9	9	Economics	0
10	10	Psychology	0
11	11	Algebra	0
12	12	Geography	0

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.

```
SELECT s.student_id,s.first_name,s.last_name,AVG(p.amount) AS
Averga_payment
FROM Students s
LEFT 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	Averga_payment
1	2	Jane	Smith	120.000000
2	3	Alice	Johnson	90.000000
3	4	Bob	Williams	110.000000
4	5	Eva	Brown	300.000000
5	6	Charlie	Taylor	130.000000
6	7	Sophia	Martin	95.000000
7	8	Daniel	Clark	105.000000
8	9	Olivia	Anderson	75.000000
9	10	Michael	Davis	85.000000
10	11	Rocky	Miller	NULL