STUDENT INFORMATION SYSTEM

Task 1. Database Design:

1. **Create the database named "SISDB"**
   CREATE DATABASE SISDB;
   USE SISDB;

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

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

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

   a. Students Table:

```
CREATE TABLE Students (
 student_id INT PRIMARY KEY,
 first_name VARCHAR(50),
 last_name VARCHAR(50),
 date_of_birth DATE,
 email VARCHAR(100),
 phone_number VARCHAR(15)
 );
```

```
mysql> CREATE TABLE Students (
    ->     student_id INT PRIMARY KEY,
    ->     first_name VARCHAR(50),
    ->     last_name VARCHAR(50),
    ->     date_of_birth DATE,
    ->     email VARCHAR(100),
    ->     phone_number VARCHAR(15)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> desc Students;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| student_id    | int         | NO   | PRI | NULL    |       |
| first_name    | varchar(50) | YES  |     | NULL    |       |
| last_name     | varchar(50) | YES  |     | NULL    |       |
| date_of_birth | date        | YES  |     | NULL    |       |
| email         | varchar(100)| YES  |     | NULL    |       |
| phone_number  | varchar(15) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

b. Teacher Table:

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

```
mysql> CREATE TABLE Teacher (
    ->     teacher_id INT PRIMARY KEY,
    ->     first_name VARCHAR(50),
    ->     last_name VARCHAR(50),
    ->     email VARCHAR(100)
    -> );
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(100)| YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

c. Courses Table:

```
CREATE TABLE Courses (
    course_id INT PRIMARY KEY,
    course_name VARCHAR(100),
    credits INT,
    teacher_id INT,
    FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
);
```

```
mysql> CREATE TABLE Courses (
    ->     course_id INT PRIMARY KEY,
    ->     course_name VARCHAR(100),
    ->     credits INT,
    ->     teacher_id INT,
    ->     FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> desc Courses;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| course_id   | int          | NO   | PRI | NULL    |       |
| course_name | varchar(100) | YES  |     | NULL    |       |
| credits     | int          | YES  |     | NULL    |       |
| teacher_id  | int          | YES  | MUL | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

d. Enrollments Table:

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)
);

```
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 (0.05 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)
```

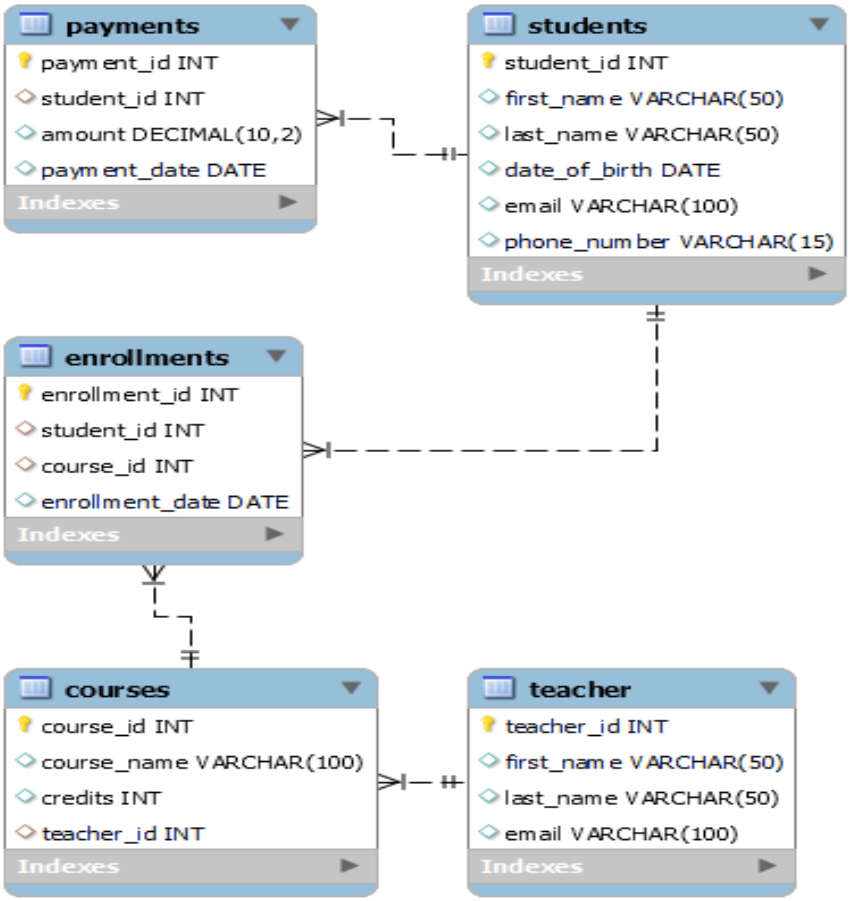
e. **Payments Table:**

```
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)
);
```

```
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 (0.05 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 (0.01 sec)
```

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



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

Primary Keys :

- student_id
- course_id
- teacher_id
- enrollment_id
- payment_id

Foreign Keys:

- Courses.teacher_id → Teacher.teacher_id
- Enrollments.student_id → Students.student_id
- Enrollments.course_id → Courses.course_id
- Payments.student_id → Students.student_id

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

1. Students:

INSERT INTO Students VALUES

(1, 'Amirtha', 'Varshini', '2002-01-01', 'amirtha@example.com', '1234567800'),

(2, 'Dharshini', 'Murugaiyan', '2001-04-15', 'dharshini@example.com', '1234567801'),

(3, 'Ashwin', 'Kumar', '2003-03-10', 'ashwin@example.com', '1234567802'),

(4, 'Sakthi', 'Devi', '2002-09-11', 'sakthi@example.com', '1234567803'),

(5, 'Priya', 'Dharshini', '2001-06-21', 'priya@example.com', '1234567804'),

(6, 'Shamshad', 'Banu', '2000-07-17', 'shamshad@example.com', '1234567805'),

(7, 'Shakeel', 'Ahemd', '2003-12-05', 'shakeel@example.com', '1234567806'),

(8, 'Sri', 'Vidya', '2002-04-03', 'srividya@example.com', '1234567807'),

(9, 'Deekap', 'Kumar', '2004-05-12', 'deekap@example.com', '1234567808'),

(10, 'Thilo', 'Thama', '2003-11-09', 'thilo@example.com', '1234567809');

```
mysql> INSERT INTO Students VALUES
    -> (1, 'Amirtha', 'Varshini', '2002-01-01', 'amirtha@example.com', '1234567800'),
    -> (2, 'Dharshini', 'Murugaiyan', '2001-04-15', 'dharshini@example.com', '1234567801'),
    -> (3, 'Ashwin', 'Kumar', '2003-03-10', 'ashwin@example.com', '1234567802'),
    -> (4, 'Sakthi', 'Devi', '2002-09-11', 'sakthi@example.com', '1234567803'),
    -> (5, 'Priya', 'Dharshini', '2001-06-21', 'priya@example.com', '1234567804'),
    -> (6, 'Shamshad', 'Banu', '2000-07-17', 'shamshad@example.com', '1234567805'),
    -> (7, 'Shakeel', 'Ahemd', '2003-12-05', 'shakeel@example.com', '1234567806'),
    -> (8, 'Sri', 'Vidya', '2002-04-03', 'srividya@example.com', '1234567807'),
    -> (9, 'Deekap', 'Kumar', '2004-05-12', 'deekap@example.com', '1234567808'),
    -> (10, 'Thilo', 'Thama', '2003-11-09', 'thilo@example.com', '1234567809');
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 | Amirtha    | Varshini   | 2002-01-01    | amirtha@example.com  | 1234567800   |
|          2 | Dharshini  | Murugaiyan | 2001-04-15    | dharshini@example.com | 1234567801   |
|          3 | Ashwin     | Kumar      | 2003-03-10    | ashwin@example.com   | 1234567802   |
|          4 | Sakthi     | Devi       | 2002-09-11    | sakthi@example.com   | 1234567803   |
|          5 | Priya      | Dharshini  | 2001-06-21    | priya@example.com    | 1234567804   |
|          6 | Shamshad   | Banu       | 2000-07-17    | shamshad@example.com | 1234567805   |
|          7 | Shakeel    | Ahemd      | 2003-12-05    | shakeel@example.com  | 1234567806   |
|          8 | Sri        | Vidya      | 2002-04-03    | srividya@example.com | 1234567807   |
|          9 | Deekap     | Kumar      | 2004-05-12    | deekap@example.com   | 1234567808   |
|         10 | Thilo      | Thama      | 2003-11-09    | thilo@example.com    | 1234567809   |
+------------+------------+------------+---------------+----------------------+--------------+
10 rows in set (0.00 sec)
```

2. Teacher

INSERT INTO Teacher VALUES

(1, 'Arun', 'Kumar', 'arun.kumar@example.com'),

(2, 'Divya', 'Shree', 'divya.shree@example.com'),

(3, 'Ravi', 'Chandran', 'ravi.chandran@example.com'),

(4, 'Meena', 'Kumari', 'meena.k@example.com'),

(5, 'Karthik', 'Raj', 'karthik.raj@example.com'),

(6, 'Priya', 'Lakshmi', 'priya.l@example.com'),

(7, 'Siva', 'Raj', 'siva.raj@example.com'),

(8, 'Anu', 'Bala', 'anu.b@example.com'),

(9, 'Vijay', 'Anand', 'vijay.anand@example.com'),

(10, 'Latha', 'Devi', 'latha.d@example.com');

```
mysql> INSERT INTO Teacher VALUES
    -> (1, 'Arun', 'Kumar', 'arun.kumar@example.com'),
    -> (2, 'Divya', 'Shree', 'divya.shree@example.com'),
    -> (3, 'Ravi', 'Chandran', 'ravi.chandran@example.com'),
    -> (4, 'Meena', 'Kumari', 'meena.k@example.com'),
    -> (5, 'Karthik', 'Raj', 'karthik.raj@example.com'),
    -> (6, 'Priya', 'Lakshmi', 'priya.l@example.com'),
    -> (7, 'Siva', 'Raj', 'siva.raj@example.com'),
    -> (8, 'Anu', 'Bala', 'anu.b@example.com'),
    -> (9, 'Vijay', 'Anand', 'vijay.anand@example.com'),
    -> (10, 'Latha', 'Devi', 'latha.d@example.com');
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * FROM Teacher;
+------------+------------+-----------+---------------------------+
| teacher_id | first_name | last_name | email                     |
+------------+------------+-----------+---------------------------+
|          1 | Arun       | Kumar     | arun.kumar@example.com    |
|          2 | Divya      | Shree     | divya.shree@example.com   |
|          3 | Ravi       | Chandran  | ravi.chandran@example.com |
|          4 | Meena      | Kumari    | meena.k@example.com       |
|          5 | Karthik    | Raj       | karthik.raj@example.com   |
|          6 | Priya      | Lakshmi   | priya.l@example.com       |
|          7 | Siva       | Raj       | siva.raj@example.com      |
|          8 | Anu        | Bala      | anu.b@example.com         |
|          9 | Vijay      | Anand     | vijay.anand@example.com   |
|         10 | Latha      | Devi      | latha.d@example.com       |
+------------+------------+-----------+---------------------------+
10 rows in set (0.00 sec)
```

3. Courses

INSERT INTO Courses VALUES

(101, 'Mathematics ', 3, 1),

(102, 'Introduction to Programming', 4, 2),

(103, 'Data Structures', 4, 3),

(104, 'Database Systems', 3, 4),

(105, 'Operating Systems', 4, 5),

(106, 'Computer Networks', 3, 6),

(107, 'Web Development', 4, 7),

(108, 'Software Engineering', 3, 8),

(109, 'Machine Learning', 4, 9),

(110, 'Artificial Intelligence', 4, 10),

(120, 'Cloud Computing', 3, 1),

(121, 'Ethical Hacking', 3, 1);

```
mysql> select * FROM Courses;
+-----------+-----------------------------+---------+------------+
| course_id | course_name                 | credits | teacher_id |
+-----------+-----------------------------+---------+------------+
|       101 | Mathematics                 |       3 |          1 |
|       102 | Introduction to Programming |       4 |          2 |
|       103 | Data Structures             |       4 |          3 |
|       104 | Database Systems            |       3 |          4 |
|       105 | Operating Systems           |       4 |          5 |
|       106 | Computer Networks           |       3 |          6 |
|       107 | Web Development             |       4 |          7 |
|       108 | Software Engineering        |       3 |          5 |
|       109 | Machine Learning            |       4 |          9 |
|       110 | Artificial Intelligence     |       4 |         10 |
|       120 | Cloud Computing             |       3 |          1 |
|       121 | Ethical Hacking             |       3 |          1 |
+-----------+-----------------------------+---------+------------+
12 rows in set (0.00 sec)
```

4. Enrollements

INSERT INTO Enrollments VALUES

(201, 1, 101, '2023-08-01'),

(202, 2, 102, '2023-08-01'),

(203, 3, 103, '2023-08-01'),

(204, 4, 104, '2023-08-01'),

(205, 5, 105, '2023-08-01'),

(206, 6, 106, '2023-08-01'),

(207, 7, 107, '2023-08-01'),

(208, 8, 108, '2023-08-01'),

(209, 9, 109, '2023-08-01'),

(210, 10, 110, '2023-08-01'),

(211, 11, 101, '2023-08-01'),

(212, 1, 102, '2023-10-01'),

(213, 1, 103, '2023-10-01'),

(214, 1, 104, '2023-10-01'),

(215, 1, 105, '2023-10-01'),

(216, 1, 106, '2023-10-01'),

(217, 1, 107, '2023-10-01'),

(218, 1, 108, '2023-10-01'),

(219, 1, 109, '2023-10-01'),

(220, 1, 110, '2023-10-01')

(221, 1, 120, '2023-10-12');

```
mysql> Select * FROM Enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           201 |          1 |       101 | 2023-08-01      |
|           202 |          2 |       102 | 2023-08-01      |
|           203 |          3 |       103 | 2023-08-01      |
|           204 |          4 |       104 | 2023-08-01      |
|           205 |          5 |       105 | 2023-08-01      |
|           206 |          6 |       106 | 2023-08-01      |
|           207 |          7 |       107 | 2023-08-01      |
|           208 |          8 |       108 | 2023-08-01      |
|           209 |          9 |       109 | 2023-08-01      |
|           210 |         10 |       110 | 2023-08-01      |
|           211 |         11 |       101 | 2023-10-01      |
|           212 |          1 |       102 | 2023-10-01      |
|           213 |          1 |       103 | 2023-10-01      |
|           214 |          1 |       104 | 2023-10-01      |
|           215 |          1 |       105 | 2023-10-01      |
|           216 |          1 |       106 | 2023-10-01      |
|           217 |          1 |       107 | 2023-10-01      |
|           218 |          1 |       108 | 2023-10-01      |
|           219 |          1 |       109 | 2023-10-01      |
|           220 |          1 |       110 | 2023-10-01      |
|           221 |          1 |       120 | 2023-10-12      |
+---------------+------------+-----------+-----------------+
21 rows in set (0.00 sec)
```

5. Payments

INSERT INTO Payments VALUES
(301, 1, 500.00, '2023-09-01'),
(302, 2, 600.00, '2023-09-02'),
(303, 3, 450.00, '2023-09-03'),
(304, 4, 700.00, '2023-09-04'),
(305, 5, 550.00, '2023-09-05'),
(306, 6, 650.00, '2023-09-06'),
(307, 7, 620.00, '2023-09-07'),
(308, 8, 500.00, '2023-09-08'),
(309, 9, 480.00, '2023-09-09'),
(310, 10, 530.00, '2023-09-10');

```
mysql> INSERT INTO Payments VALUES
    -> (301, 1, 500.00, '2023-09-01'),
    -> (302, 2, 600.00, '2023-09-02'),
    -> (303, 3, 450.00, '2023-09-03'),
    -> (304, 4, 700.00, '2023-09-04'),
    -> (305, 5, 550.00, '2023-09-05'),
    -> (306, 6, 650.00, '2023-09-06'),
    -> (307, 7, 620.00, '2023-09-07'),
    -> (308, 8, 500.00, '2023-09-08'),
    -> (309, 9, 480.00, '2023-09-09'),
    -> (310, 10, 530.00, '2023-09-10');
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 | 500.00 | 2023-09-01   |
|        302 |          2 | 600.00 | 2023-09-02   |
|        303 |          3 | 450.00 | 2023-09-03   |
|        304 |          4 | 700.00 | 2023-09-04   |
|        305 |          5 | 550.00 | 2023-09-05   |
|        306 |          6 | 650.00 | 2023-09-06   |
|        307 |          7 | 620.00 | 2023-09-07   |
|        308 |          8 | 500.00 | 2023-09-08   |
|        309 |          9 | 480.00 | 2023-09-09   |
|        310 |         10 | 530.00 | 2023-09-10   |
+------------+------------+--------+--------------+
10 rows in set (0.00 sec)
```

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

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

a. First Name: John© Hexaware Technologies Limited. All rights www.hexaware.com

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');

```
mysql> INSERT INTO Students 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 | Amirtha    | Varshini   | 2002-01-01    | amirtha@example.com  | 1234567800   |
|          2 | Dharshini  | Murugaiyan | 2001-04-15    | dharshini@example.com | 1234567801   |
|          3 | Ashwin     | Kumar      | 2003-03-10    | ashwin@example.com   | 1234567802   |
|          4 | Sakthi     | Devi       | 2002-09-11    | sakthi@example.com   | 1234567803   |
|          5 | Priya      | Dharshini  | 2001-06-21    | priya@example.com    | 1234567804   |
|          6 | Shamshad   | Banu       | 2000-07-17    | shamshad@example.com | 1234567805   |
|          7 | Shakeel    | Ahemd      | 2003-12-05    | shakeel@example.com  | 1234567806   |
|          8 | Sri        | Vidya      | 2002-04-03    | srividya@example.com | 1234567807   |
|          9 | Deekap     | Kumar      | 2004-05-12    | deekap@example.com   | 1234567808   |
|         10 | Thilo      | Thama      | 2003-11-09    | thilo@example.com    | 1234567809   |
|         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

INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
VALUES (211, 11, 101, '2023-10-01');

```
mysql> SELECT * FROM Enrollments WHERE student_id = 11 AND course_id = 101;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           211 |         11 |       101 | 2023-10-01      |
+---------------+------------+-----------+-----------------+
1 row 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.

UPDATE Teacher
SET email = 'divya.updated@example.com'
WHERE teacher_id = 2;

```
mysql> UPDATE Teacher
    -> SET email = 'divya.updated@example.com'
    -> WHERE teacher_id = 2;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * FROM Teacher;
+------------+------------+-----------+---------------------------+
| teacher_id | first_name | last_name | email                     |
+------------+------------+-----------+---------------------------+
|          1 | Arun       | Kumar     | arun.kumar@example.com    |
|          2 | Divya      | Shree     | divya.updated@example.com |
|          3 | Ravi       | Chandran  | ravi.chandran@example.com |
|          4 | Meena      | Kumari    | meena.k@example.com       |
|          5 | Karthik    | Raj       | karthik.raj@example.com   |
|          6 | Priya      | Lakshmi   | priya.l@example.com       |
|          7 | Siva       | Raj       | siva.raj@example.com      |
|          8 | Anu        | Bala      | anu.b@example.com         |
|          9 | Vijay      | Anand     | vijay.anand@example.com   |
|         10 | Latha      | Devi      | latha.d@example.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.

DELETE FROM Enrollments
WHERE enrollment_id = 210 AND course_id=110;

```
mysql> DELETE FROM Enrollments
    -> WHERE enrollment_id = 210 AND course_id=110;
Query OK, 1 row affected (0.01 sec)

mysql> select * FROM Enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           201 |          1 |       101 | 2023-08-01      |
|           202 |          2 |       102 | 2023-08-01      |
|           203 |          3 |       103 | 2023-08-01      |
|           204 |          4 |       104 | 2023-08-01      |
|           205 |          5 |       105 | 2023-08-01      |
|           206 |          6 |       106 | 2023-08-01      |
|           207 |          7 |       107 | 2023-08-01      |
|           208 |          8 |       108 | 2023-08-01      |
|           209 |          9 |       109 | 2023-08-01      |
|           211 |         11 |       101 | 2023-10-01      |
|           212 |          1 |       102 | 2023-10-01      |
|           213 |          1 |       103 | 2023-10-01      |
|           214 |          1 |       104 | 2023-10-01      |
|           215 |          1 |       105 | 2023-10-01      |
|           216 |          1 |       106 | 2023-10-01      |
|           217 |          1 |       107 | 2023-10-01      |
|           218 |          1 |       108 | 2023-10-01      |
|           219 |          1 |       109 | 2023-10-01      |
|           220 |          1 |       110 | 2023-10-01      |
+---------------+------------+-----------+-----------------+
19 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.**

UPDATE Courses
SET teacher_id = 5
WHERE course_id = 108;

```
mysql> UPDATE Courses
    -> SET teacher_id = 5
    -> WHERE course_id = 108;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * FROM Courses;
+-----------+-----------------------------+---------+------------+
| course_id | course_name                 | credits | teacher_id |
+-----------+-----------------------------+---------+------------+
|       101 | Mathematics                 |       3 |          1 |
|       102 | Introduction to Programming |       4 |          2 |
|       103 | Data Structures             |       4 |          3 |
|       104 | Database Systems            |       3 |          4 |
|       105 | Operating Systems           |       4 |          5 |
|       106 | Computer Networks           |       3 |          6 |
|       107 | Web Development             |       4 |          7 |
|       108 | Software Engineering        |       3 |          5 |
|       109 | Machine Learning            |       4 |          9 |
|       110 | Artificial Intelligence     |       4 |         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**.

DELETE FROM Enrollments WHERE student_id = 9;
DELETE FROM Payments WHERE student_id = 9;
DELETE FROM Students WHERE student_id = 9;

```
mysql> DELETE FROM Enrollments WHERE student_id = 9;
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM Payments WHERE student_id = 9;
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM Students WHERE student_id = 9;
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM Students;
+------------+------------+------------+---------------+-----------------------+--------------+
| student_id | first_name | last_name  | date_of_birth | email                 | phone_number |
+------------+------------+------------+---------------+-----------------------+--------------+
|          1 | Amirtha    | Varshini   | 2002-01-01    | amirtha@example.com   | 1234567800   |
|          2 | Dharshini  | Murugaiyan | 2001-04-15    | dharshini@example.com | 1234567801   |
|          3 | Ashwin     | Kumar      | 2003-03-10    | ashwin@example.com    | 1234567802   |
|          4 | Sakthi     | Devi       | 2002-09-11    | sakthi@example.com    | 1234567803   |
|          5 | Priya      | Dharshini  | 2001-06-21    | priya@example.com     | 1234567804   |
|          6 | Shamshad   | Banu       | 2000-07-17    | shamshad@example.com  | 1234567805   |
|          7 | Shakeel    | Ahemd      | 2003-12-05    | shakeel@example.com   | 1234567806   |
|          8 | Sri        | Vidya      | 2002-04-03    | srividya@example.com  | 1234567807   |
|         10 | Thilo      | Thama      | 2003-11-09    | thilo@example.com     | 1234567809   |
|         11 | John       | Doe        | 1995-08-15    | john.doe@example.com  | 1234567890   |
+------------+------------+------------+---------------+-----------------------+--------------+
10 rows in set (0.00 sec)
```

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

UPDATE Payments
SET amount = 575
WHERE payment_id = 305;

```
mysql> UPDATE Payments
    -> SET amount = 575
    -> WHERE payment_id = 305;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> SELECT * FROM Payments;
+------------+------------+--------+--------------+
| payment_id | student_id | amount | payment_date |
+------------+------------+--------+--------------+
|        301 |          1 | 500.00 | 2023-09-01   |
|        302 |          2 | 600.00 | 2023-09-02   |
|        303 |          3 | 450.00 | 2023-09-03   |
|        304 |          4 | 700.00 | 2023-09-04   |
|        305 |          5 | 575.00 | 2023-09-05   |
|        306 |          6 | 650.00 | 2023-09-06   |
|        307 |          7 | 620.00 | 2023-09-07   |
|        308 |          8 | 500.00 | 2023-09-08   |
|        310 |         10 | 530.00 | 2023-09-10   |
+------------+------------+--------+--------------+
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**.

```
SELECT s.first_name, s.last_name, SUM(p.amount) AS total_payment
FROM Students s
JOIN Payments p ON s.student_id = p.student_id
WHERE s.student_id = 1
GROUP BY s.first_name, s.last_name;
```

```
mysql> SELECT s.first_name, s.last_name, SUM(p.amount) AS total_payment
    -> FROM Students s
    -> JOIN Payments p ON s.student_id = p.student_id
    -> WHERE s.student_id = 1
    -> GROUP BY s.first_name, s.last_name;
+------------+-----------+---------------+
| first_name | last_name | total_payment |
+------------+-----------+---------------+
| Amirtha    | Varshini  |        500.00 |
+------------+-----------+---------------+
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.**

```
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
ORDER BY student_count DESC;
```

```
mysql> SELECT c.course_name, COUNT(e.student_id) AS student_count
    -> FROM Courses c
    -> LEFT JOIN Enrollments e ON c.course_id = e.course_id
    -> GROUP BY c.course_name
    -> ORDER BY student_count DESC;
+----------------------------+---------------+
| course_name                | student_count |
+----------------------------+---------------+
| Mathematics                |             2 |
| Introduction to Programming |             2 |
| Data Structures            |             2 |
| Database Systems           |             2 |
| Operating Systems          |             2 |
| Computer Networks          |             2 |
| Web Development            |             2 |
| Software Engineering       |             2 |
| Artificial Intelligence    |             2 |
| Machine Learning           |             1 |
| Cloud Computing            |             1 |
| Ethical Hacking            |             0 |
+----------------------------+---------------+
12 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.**

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

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

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

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

```
mysql> SELECT s.first_name, s.last_name, c.course_name
    -> FROM Students s
    -> JOIN Enrollments e ON s.student_id = e.student_id
    -> JOIN Courses c ON e.course_id = c.course_id
    -> ORDER BY s.student_id;
+------------+------------+----------------------------+
| first_name | last_name  | course_name                |
+------------+------------+----------------------------+
| Amirtha    | Varshini   | Mathematics                |
| Amirtha    | Varshini   | Introduction to Programming |
| Amirtha    | Varshini   | Data Structures            |
| Amirtha    | Varshini   | Database Systems           |
| Amirtha    | Varshini   | Operating Systems          |
| Amirtha    | Varshini   | Computer Networks          |
| Amirtha    | Varshini   | Web Development            |
| Amirtha    | Varshini   | Software Engineering       |
| Amirtha    | Varshini   | Machine Learning           |
| Amirtha    | Varshini   | Artificial Intelligence    |
| Dharshini  | Murugaiyan | Introduction to Programming |
| Ashwin     | Kumar      | Data Structures            |
| Sakthi     | Devi       | Database Systems           |
| Priya      | Dharshini  | Operating Systems          |
| Shamshad   | Banu       | Computer Networks          |
| Shakeel    | Ahemd      | Web Development            |
| Sri        | Vidya      | Software Engineering       |
| John       | Doe        | Mathematics                |
+------------+------------+----------------------------+
18 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.**

SELECT t.first_name, t.last_name, c.course_name
FROM Teacher t
LEFT JOIN Courses c ON t.teacher_id = c.teacher_id
ORDER BY t.teacher_id;

```
mysql> SELECT t.first_name, t.last_name, c.course_name
    -> FROM Teacher t
    -> LEFT JOIN Courses c ON t.teacher_id = c.teacher_id
    -> ORDER BY t.teacher_id;
+------------+-----------+----------------------------+
| first_name | last_name | course_name                |
+------------+-----------+----------------------------+
| Arun       | Kumar     | Mathematics                |
| Divya      | Shree     | Introduction to Programming |
| Ravi       | Chandran  | Data Structures            |
| Meena      | Kumari    | Database Systems           |
| Karthik    | Raj       | Operating Systems          |
| Karthik    | Raj       | Software Engineering       |
| Priya      | Lakshmi   | Computer Networks          |
| Siva       | Raj       | Web Development            |
| Anu        | Bala      | NULL                       |
| Vijay      | Anand     | Machine Learning           |
| Latha      | Devi      | Artificial Intelligence    |
+------------+-----------+----------------------------+
11 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.

SELECT s.first_name, s.last_name, e.enrollment_date
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id

WHERE e.course_id = 101;

```
mysql> SELECT s.first_name, s.last_name, e.enrollment_date
    -> FROM Students s
    -> JOIN Enrollments e ON s.student_id = e.student_id
    -> WHERE e.course_id = 101;
+------------+-----------+-----------------+
| first_name | last_name | enrollment_date |
+------------+-----------+-----------------+
| Amirtha    | Varshini  | 2023-08-01      |
| John       | Doe       | 2023-10-01      |
+------------+-----------+-----------------+
2 rows in set (0.00 sec)
```

7. Find the names of students who have not made any payments. Use a LEFT JOIN between the "Students" table and the "Payments" table and filter for students with NULL payment records.

SELECT s.first_name, s.last_name
FROM Students s
LEFT JOIN Payments p ON s.student_id = p.student_id
WHERE p.payment_id IS NULL;

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

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

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;

```
mysql> SELECT c.course_id, c.course_name
    -> FROM Courses c
    -> LEFT JOIN Enrollments e ON c.course_id = e.course_id
    -> WHERE e.enrollment_id IS NULL;
+-----------+-----------------+
| course_id | course_name     |
+-----------+-----------------+
|       120 | Cloud Computing |
+-----------+-----------------+
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.

SELECT s.first_name, s.last_name, COUNT(e.course_id) AS course_count
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
GROUP BY s.student_id, s.first_name, s.last_name
HAVING COUNT(e.course_id) > 1;

```
mysql> SELECT s.first_name, s.last_name, COUNT(e.course_id) AS course_count
    -> FROM Students s
    -> JOIN Enrollments e ON s.student_id = e.student_id
    -> GROUP BY s.student_id, s.first_name, s.last_name
    -> HAVING COUNT(e.course_id) > 1;
+------------+-----------+--------------+
| first_name | last_name | course_count |
+------------+-----------+--------------+
| Amirtha    | Varshini  |           10 |
+------------+-----------+--------------+
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.

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

```
mysql> SELECT t.first_name, t.last_name
    -> FROM Teacher t
    -> LEFT JOIN Courses c ON t.teacher_id = c.teacher_id
    -> WHERE c.course_id IS NULL;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| Anu        | Bala      |
+------------+-----------+
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.**

```
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 sub;
```

```
mysql> 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 sub;
+-------------------------+
| avg_students_per_course |
+-------------------------+
|                  1.8000 |
+-------------------------+
1 row in set (0.00 sec)
```

2. **Identify the student(s) who made the highest payment. Use a subquery to find the maximum payment amount and then retrieve the student(s) associated with that amount.**

```
SELECT 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(amount) FROM Payments
);
```

```
mysql> SELECT 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(amount) FROM Payments
    -> );
+------------+-----------+--------+
| first_name | last_name | amount |
+------------+-----------+--------+
| Sakthi     | Devi      | 700.00 |
+------------+-----------+--------+
1 row in set (0.00 sec)
```

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

```
SELECT c.course_name, COUNT(e.student_id) AS student_count
FROM Courses c
JOIN Enrollments e ON c.course_id = e.course_id
GROUP BY c.course_id, c.course_name
HAVING COUNT(e.student_id) = (
SELECT MAX(student_count)
FROM (
SELECT COUNT(student_id) AS student_count
FROM Enrollments
 GROUP BY course_id
 ) AS counts
);
```

```
mysql> SELECT c.course_name, COUNT(e.student_id) AS student_count
    -> FROM Courses c
    -> JOIN Enrollments e ON c.course_id = e.course_id
    -> GROUP BY c.course_id, c.course_name
    -> HAVING COUNT(e.student_id) = (
    -> SELECT MAX(student_count)
    ->           FROM (
    -> SELECT COUNT(student_id) AS student_count
    -> FROM Enrollments
    ->  GROUP BY course_id
    ->  ) AS counts
    -> );
+---------------------------+---------------+
| course_name               | student_count |
+---------------------------+---------------+
| Mathematics               |             2 |
| Introduction to Programming |             2 |
| Data Structures           |             2 |
| Database Systems          |             2 |
| Operating Systems         |             2 |
| Computer Networks         |             2 |
| Web Development           |             2 |
| Software Engineering      |             2 |
+---------------------------+---------------+
8 rows in set (0.00 sec)
```

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

```
SELECT t.first_name, t.last_name, SUM(p.amount) AS total_payment
FROM Teacher t
JOIN Courses c ON t.teacher_id = c.teacher_id
JOIN Enrollments e ON c.course_id = e.course_id
JOIN Payments p ON e.student_id = p.student_id
GROUP BY t.teacher_id, t.first_name, t.last_name;
```

```
mysql> SELECT t.first_name, t.last_name, SUM(p.amount) AS total_payment
    -> FROM Teacher t
    -> JOIN Courses c ON t.teacher_id = c.teacher_id
    -> JOIN Enrollments e ON c.course_id = e.course_id
    -> JOIN Payments p ON e.student_id = p.student_id
    -> GROUP BY t.teacher_id, t.first_name, t.last_name;
+------------+-----------+---------------+
| first_name | last_name | total_payment |
+------------+-----------+---------------+
| Arun       | Kumar     |        500.00 |
| Divya      | Shree     |       1100.00 |
| Ravi       | Chandran  |        950.00 |
| Meena      | Kumari    |       1200.00 |
| Karthik    | Raj       |       2075.00 |
| Priya      | Lakshmi   |       1150.00 |
| Siva       | Raj       |       1120.00 |
| Vijay      | Anand     |        500.00 |
| Latha      | Devi      |        500.00 |
+------------+-----------+---------------+
9 rows in set (0.00 sec)
```

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

```
SELECT s.student_id, s.first_name, s.last_name
FROM Students s
WHERE NOT EXISTS (
 SELECT course_id FROM Courses
 EXCEPT SELECT course_id FROM Enrollments e WHERE e.student_id =
s.student_id
);
```

```
mysql> SELECT s.student_id, s.first_name, s.last_name
    -> FROM Students s
    -> WHERE NOT EXISTS (
    ->  SELECT course_id FROM Courses
    ->  EXCEPT SELECT course_id FROM Enrollments e WHERE e.student_id = s.student_id
    -> );
+------------+------------+-----------+
| student_id | first_name | last_name |
+------------+------------+-----------+
|          1 | Amirtha    | Varshini  |
+------------+------------+-----------+
1 row in set (0.00 sec)
```

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

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

```
mysql> SELECT t.first_name, t.last_name
    -> FROM Teacher t
    -> WHERE t.teacher_id NOT IN (
    -> SELECT DISTINCT teacher_id FROM Courses
    -> );
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| Anu        | Bala      |
+------------+-----------+
1 row in set (0.00 sec)
```

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

```
SELECT AVG(age) AS average_age
FROM (
   SELECT TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age
   FROM Students
) AS ages;
```

```
mysql>
mysql> SELECT AVG(age) AS average_age
    -> FROM (
    ->     SELECT TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age
    ->     FROM Students
    -> ) AS ages;
+-------------+
| average_age |
+-------------+
|     23.2000 |
+-------------+
1 row in set (0.00 sec)
```

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

```
SELECT course_id, course_name
FROM Courses
```

```
WHERE course_id NOT IN (
   SELECT DISTINCT course_id FROM Enrollments
);
```

```
mysql> INSERT INTO Courses (course_id, course_name, credits, teacher_id)
    -> VALUES (121, 'Ethical Hacking', 3, 1);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT course_id, course_name
    -> FROM Courses
    -> WHERE course_id NOT IN (
    ->     SELECT DISTINCT course_id FROM Enrollments
    -> );
+-----------+-----------------+
| course_id | course_name     |
+-----------+-----------------+
|       121 | Ethical Hacking |
+-----------+-----------------+
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**.

```
SELECT s.first_name, s.last_name, c.course_name, SUM(p.amount) AS
total_payment
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id
JOIN Payments p ON s.student_id = p.student_id
GROUP BY s.student_id, c.course_id;
```

```
mysql> SELECT s.first_name, s.last_name, c.course_name, SUM(p.amount) AS total_payment
    -> FROM Students s
    -> JOIN Enrollments e ON s.student_id = e.student_id
    -> JOIN Courses c ON e.course_id = c.course_id
    -> JOIN Payments p ON s.student_id = p.student_id
    -> GROUP BY s.student_id, c.course_id;
+------------+------------+----------------------------+---------------+
| first_name | last_name  | course_name                | total_payment |
+------------+------------+----------------------------+---------------+
| Amirtha    | Varshini   | Mathematics                |        500.00 |
| Amirtha    | Varshini   | Introduction to Programming |        500.00 |
| Amirtha    | Varshini   | Data Structures            |        500.00 |
| Amirtha    | Varshini   | Database Systems           |        500.00 |
| Amirtha    | Varshini   | Operating Systems          |        500.00 |
| Amirtha    | Varshini   | Computer Networks          |        500.00 |
| Amirtha    | Varshini   | Web Development            |        500.00 |
| Amirtha    | Varshini   | Software Engineering       |        500.00 |
| Amirtha    | Varshini   | Machine Learning           |        500.00 |
| Amirtha    | Varshini   | Artificial Intelligence    |        500.00 |
| Amirtha    | Varshini   | Cloud Computing            |        500.00 |
| Dharshini  | Murugaiyan | Introduction to Programming |        600.00 |
| Ashwin     | Kumar      | Data Structures            |        450.00 |
| Sakthi     | Devi       | Database Systems           |        700.00 |
| Priya      | Dharshini  | Operating Systems          |        575.00 |
| Shamshad   | Banu       | Computer Networks          |        650.00 |
| Shakeel    | Ahemd      | Web Development            |        620.00 |
| Sri        | Vidya      | Software Engineering       |        500.00 |
+------------+------------+----------------------------+---------------+
18 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.

SELECT s.first_name, s.last_name, COUNT(p.payment_id) AS payment_count
FROM Students s
JOIN Payments p ON s.student_id = p.student_id
GROUP BY s.student_id
HAVING COUNT(p.payment_id) > 1;

```
mysql> INSERT INTO Payments (payment_id, student_id, amount, payment_date)
    -> VALUES (311, 1, 250.00, '2023-10-15');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT s.first_name, s.last_name, COUNT(p.payment_id) AS payment_count
    -> FROM Students s
    -> JOIN Payments p ON s.student_id = p.student_id
    -> GROUP BY s.student_id
    -> HAVING COUNT(p.payment_id) > 1;
+------------+-----------+---------------+
| first_name | last_name | payment_count |
+------------+-----------+---------------+
| Amirtha    | Varshini  |             2 |
+------------+-----------+---------------+
1 row in set (0.00 sec)
```

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

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

```
mysql> SELECT s.student_id, s.first_name, s.last_name, SUM(p.amount) AS total_payment
    -> FROM Students s
    -> JOIN Payments p ON s.student_id = p.student_id
    -> GROUP BY s.student_id;
+------------+------------+------------+---------------+
| student_id | first_name | last_name  | total_payment |
+------------+------------+------------+---------------+
|          1 | Amirtha    | Varshini   |        750.00 |
|          2 | Dharshini  | Murugaiyan |        600.00 |
|          3 | Ashwin     | Kumar      |        450.00 |
|          4 | Sakthi     | Devi       |        700.00 |
|          5 | Priya      | Dharshini  |        575.00 |
|          6 | Shamshad   | Banu       |        650.00 |
|          7 | Shakeel    | Ahemd      |        620.00 |
|          8 | Sri        | Vidya      |        500.00 |
|         10 | Thilo      | Thama      |        530.00 |
+------------+------------+------------+---------------+
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**

```
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_name;
```

```
mysql> 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_name;
+----------------------------+------------------+
| course_name                | enrollment_count |
+----------------------------+------------------+
| Mathematics                |                2 |
| Introduction to Programming |                2 |
| Data Structures            |                2 |
| Database Systems           |                2 |
| Operating Systems          |                2 |
| Computer Networks          |                2 |
| Web Development            |                2 |
| Software Engineering       |                2 |
| Machine Learning           |                1 |
| Artificial Intelligence    |                1 |
| Cloud Computing            |                1 |
| Ethical Hacking            |                0 |
+----------------------------+------------------+
12 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.**

```
SELECT AVG(student_total) AS average_student_payment
FROM (
SELECT s.student_id, SUM(p.amount) AS student_total
FROM Students s
JOIN Payments p ON s.student_id = p.student_id
GROUP BY s.student_id
) AS totals;
```

```
mysql> SELECT AVG(student_total) AS average_student_payment
    -> FROM (
    -> SELECT s.student_id, SUM(p.amount) AS student_total
    -> FROM Students s
    -> JOIN Payments p ON s.student_id = p.student_id
    -> GROUP BY s.student_id
    -> ) AS totals;
+-------------------------+
| average_student_payment |
+-------------------------+
|              597.222222 |
+-------------------------+
1 row in set (0.00 sec)
```