STUDENT INFORMATION SYSTEM (SIS)

SQL

Task 1: Database Design:

1. Create the database named "SISDB".

```
CREATE DATABASE SISDB;
USE 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

```
CREATE TABLE Students (
    student_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    date_of_birth DATE,
    email VARCHAR(100),
    phone_number VARCHAR(20)
);
```

b. Teacher

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

c. Courses

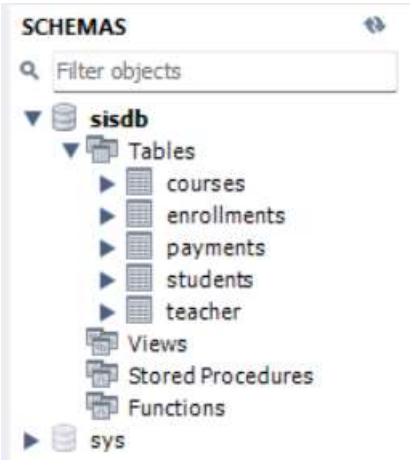
```
CREATE TABLE Courses (
    course_id INT  AUTO_INCREMENT PRIMARY KEY,
    course_name VARCHAR(100),
    credits INT,
    teacher_id INT,
    FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
);
```

d. Enrollments

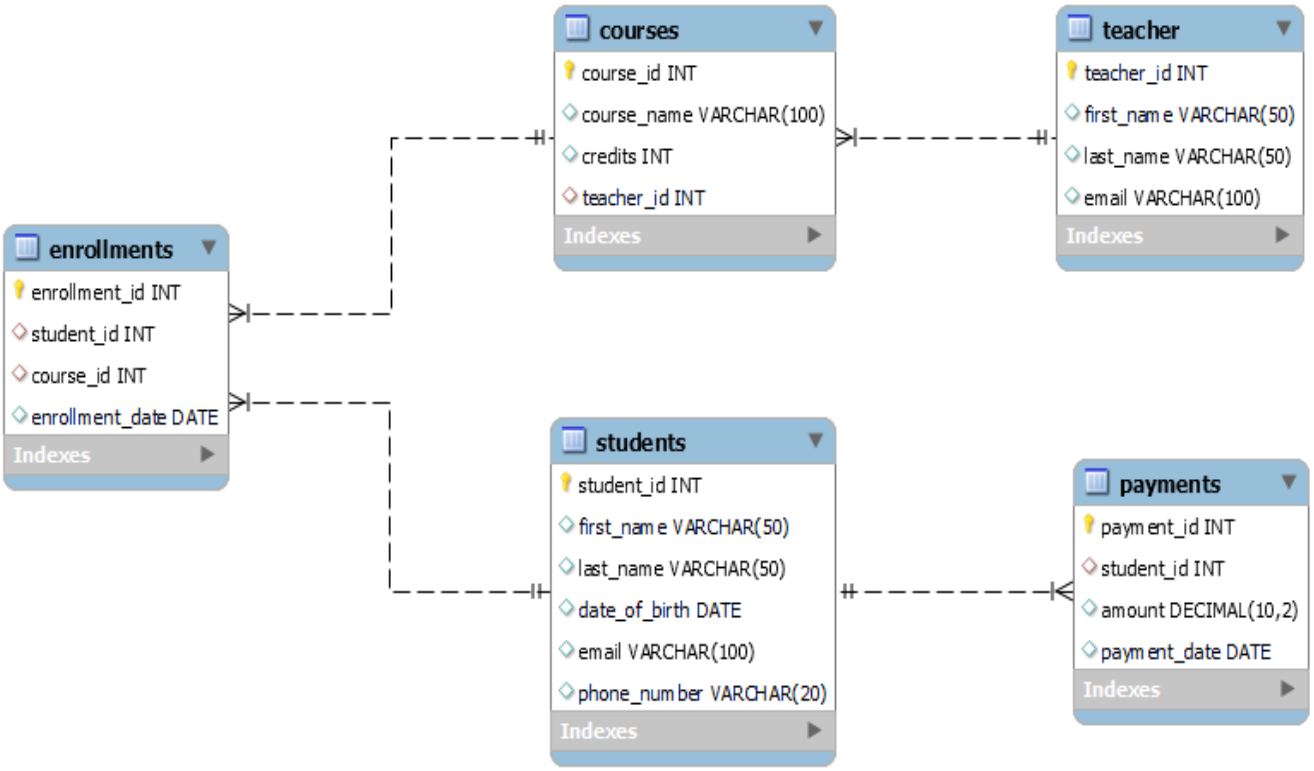
```
CREATE TABLE Enrollments (
    enrollment_id INT  AUTO_INCREMENT PRIMARY KEY,
    student_id INT,
    course_id INT,
    enrollment_date DATE,
    FOREIGN KEY (student_id) REFERENCES Students(student_id),
    FOREIGN KEY (course_id) REFERENCES Courses(course_id)
);
```

e. Payments

```
CREATE TABLE Payments (
    payment_id INT  AUTO_INCREMENT 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

```
INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number)
VALUES
    (1, 'Kavya', 'Kaja', '2002-04-12', 'kavya@gmail.com', '9734567890'),
    (2, 'Ramya', 'Gowda', '2001-06-18', 'ramya@gmail.com', '9876543210'),
    (3, 'Usha', 'Manda', '2006-12-05', 'usha@gmail.com', '8667890123'),
    (4, 'Raju', 'Kalaga', '2005-11-15', 'raju@gmail.com', '7434567890'),
    (5, 'Ravi', 'Mutta', '2009-10-20', 'ravi@gmail.com', '9876543210'),
    (6, 'Divya', 'Varma', '2000-03-21', 'divya@gmail.com', '9467890123'),
    (7, 'Ram', 'Ganta', '2004-04-06', 'ram@gmail.com', '8234567890'),
    (8, 'Mohan', 'Gurram', '2003-11-03', 'mohan@gmail.com', '9876543210'),
    (9, 'Lakshmi', 'Reddy', '2004-01-11', 'lakshmi@gmail.com', '6767890123'),
    (10, 'Vani', 'Koduri', '2006-06-10', 'vani@gmail.com', '7367890123');

SELECT * FROM Students;
```

student_id	first_name	last_name	date_of_birth	email	phone_number
1	Kavya	Kaja	2002-04-12	kavya@gmail.com	9734567890
2	Ramya	Gowda	2001-06-18	ramya@gmail.com	9876543210
3	Usha	Manda	2006-12-05	usha@gmail.com	8667890123
4	Raju	Kalaga	2005-11-15	raju@gmail.com	7434567890
5	Ravi	Mutta	2009-10-20	ravi@gmail.com	9876543210
6	Divya	Varma	2000-03-21	divya@gmail.com	9467890123
7	Ram	Ganta	2004-04-06	ram@gmail.com	8234567890
8	Mohan	Gurram	2003-11-03	mohan@gmail.com	9876543210
9	Lakshmi	Reddy	2004-01-11	lakshmi@gmail.com	6767890123
10	Vani	Koduri	2006-06-10	vani@gmail.com	7367890123

ii. Teacher

```
INSERT INTO Teacher (teacher_id, first_name, last_name, email)
VALUES
    (11, 'Surya' , 'Pendyala', 'surya@gmail.com'),
    (12, 'Ramesh', 'Jasthi', 'ramesh@gmail.com'),
    (13, 'Pooja', 'Palakurthi', 'pooja@gmail.com'),
    (14, 'John', 'Paluri', 'john@gmail.com'),
    (15, 'Sony', 'Konduri', 'sony@gmail.com'),
    (16, 'Mary', 'Dasari', 'mary@gmail.com'),
    (17, 'Chaya', 'Parimi', 'chaya@gmail.com'),
    (18, 'Pragna', 'Pathuri', 'pragna@gmail.com'),
    (19, 'Rohan', 'Katta', 'rohan@gmail.com'),
    (20, 'Rohith', 'Mavuri', 'rohith@gmail.com');

SELECT * FROM Teacher;
```

	teacher_id	first_name	last_name	email
▶	11	Surya	Pendyala	surya@gmail.com
	12	Ramesh	Jasthi	ramesh@gmail.com
	13	Pooja	Palakurthi	pooja@gmail.com
	14	John	Paluri	john@gmail.com
	15	Sony	Konduri	sony@gmail.com
	16	Mary	Dasari	mary@gmail.com
	17	Chaya	Parimi	chaya@gmail.com
	18	Pragna	Pathuri	pragna@gmail.com
	19	Rohan	Katta	rohan@gmail.com
	20	Rohith	Mavuri	rohith@gmail.com

iii. Courses

```
INSERT INTO Courses (course_id, course_name, credits, teacher_id)
VALUES
    (21, 'Introduction to Programming', 3, 11),
    (22, 'Database Management', 4, 12),
    (23, 'Web Development', 3, 13),
    (24, 'Java', 4, 14),
    (25, 'C Programming', 3, 15),
    (26, 'MYSQL', 4, 12),
    (27, 'Full Stack', 4, 14),
    (28, 'Python', 3, 15),
    (29, 'Software Engineering', 4, 19),
    (30, 'data Structures and Algorithms', 3, 20);

SELECT * FROM Courses;
```

	course_id	course_name	credits	teacher_id
▶	21	Introduction to Programming	3	11
	22	Database Management	4	12
	23	Web Development	3	13
	24	Java	4	14
	25	C Programming	3	15
	26	MYSQL	4	12
	27	Full Stack	4	14
	28	Python	3	15
	29	Software Engineering	4	19
	30	data Structures and Algorithms	3	20

iv. Enrollments

```
INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
VALUES
    (31, 1, 21, '2023-01-05'),
    (32, 2, 22, '2023-03-10'),
    (33, 3, 21, '2023-01-05'),
    (34, 4, 24, '2023-06-25'),
    (35, 5, 25, '2023-05-04'),
    (36, 2, 26, '2023-09-15'),
    (37, 4, 27, '2023-12-05'),
    (38, 5, 28, '2023-11-10'),
    (39, 9, 29, '2023-09-06'),
    (40, 10,29, '2023-09-06');

SELECT * FROM Enrollments;
```

enrollment_id	student_id	course_id	enrollment_date
31	1	21	2023-01-05
32	2	22	2023-03-10
33	3	21	2023-01-05
34	4	24	2023-06-25
35	5	25	2023-05-04
36	2	26	2023-09-15
37	4	27	2023-12-05
38	5	28	2023-11-10
39	9	29	2023-09-06
40	10	29	2023-09-06

v. Payments

```
INSERT INTO Payments (payment_id, student_id, amount, payment_date)
VALUES
    (41, 1, 10000, '2023-01-10'),
    (42, 2, 7000, '2023-03-15'),
    (43, 3, 10000, '2023-01-10'),
    (44, 4, 9000, '2023-06-30'),
    (45, 5, 13000, '2023-05-08'),
    (46, 2, 8000, '2023-09-20'),
    (47, 4, 16000, '2023-12-10'),
    (48, 5, 12000, '2023-11-15'),
    (49, 9, 15000, '2023-09-12'),
    (50, 10, 15000, '2023-09-12');

SELECT * FROM Payments;
```

payment_id	student_id	amount	payment_date
41	1	10000.00	2023-01-10
42	2	7000.00	2023-03-15
43	3	10000.00	2023-01-10
44	4	9000.00	2023-06-30
45	5	13000.00	2023-05-08
46	2	8000.00	2023-09-20
47	4	16000.00	2023-12-10
48	5	12000.00	2023-11-15
49	9	15000.00	2023-09-12
50	10	15000.00	2023-09-12

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

SELECT * FROM Students;
```

student_id	first_name	last_name	date_of_birth	email	phone_number
1	Kavya	Kaja	2002-04-12	kavya@gmail.com	9734567890
2	Ramya	Gowda	2001-06-18	ramya@gmail.com	9876543210
3	Usha	Manda	2006-12-05	usha@gmail.com	8667890123
4	Raju	Kalaga	2005-11-15	raju@gmail.com	7434567890
5	Ravi	Mutta	2009-10-20	ravi@gmail.com	9876543210
6	Divya	Varma	2000-03-21	divya@gmail.com	9467890123
7	Ram	Ganta	2004-04-06	ram@gmail.com	8234567890
8	Mohan	Gurram	2003-11-03	mohan@gmail.com	9876543210
9	Lakshmi	Reddy	2004-01-11	lakshmi@gmail.com	6767890123
10	Vani	Koduri	2006-06-10	vani@gmail.com	7367890123
11	John	Doe	1995-08-15	john.doe@exampl...	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 (student_id, course_id, enrollment_date)
VALUES (1 , 26, '2024-01-07');

SELECT * FROM Enrollments;
```

enrollment_id	student_id	course_id	enrollment_date
31	1	21	2023-01-05
32	2	22	2023-03-10
33	3	21	2023-01-05
34	4	24	2023-06-25
35	5	25	2023-05-04
36	2	26	2023-09-15
37	4	27	2023-12-05
38	5	28	2023-11-10
39	9	29	2023-09-06
40	10	29	2023-09-06
41	1	26	2024-01-07

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 = 'Surya123@gmail.com'
WHERE first_name = 'Surya';

SELECT * FROM Teacher;
```

teacher_id	first_name	last_name	email
11	Surya	Pendyala	Surya123@gmail.com
12	Ramesh	Jasthi	ramesh@gmail.com
13	Pooja	Palakurthi	pooja@gmail.com
14	John	Paluri	john@gmail.com
15	Sony	Konduri	sony@gmail.com
16	Mary	Dasari	mary@gmail.com
17	Chaya	Parimi	chaya@gmail.com
18	Pragna	Pathuri	pragna@gmail.com
19	Rohan	Katta	rohan@gmail.com
20	Rohith	Mavuri	rohith@gmail.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 student_id = 2 AND course_id = 22;

SELECT * FROM Enrollments;
```

enrollment_id	student_id	course_id	enrollment_date
31	1	21	2023-01-05
33	3	21	2023-01-05
34	4	24	2023-06-25
35	5	25	2023-05-04
36	2	26	2023-09-15
37	4	27	2023-12-05
38	5	28	2023-11-10
39	9	29	2023-09-06
40	10	29	2023-09-06
41	1	26	2024-01-07

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 = 11 WHERE course_id = 25;

SELECT * FROM courses;
```

course_id	course_name	credits	teacher_id
21	Introduction to Programming	3	11
22	Database Management	4	12
23	Web Development	3	13
24	Java	4	14
25	C Programming	3	11
26	MYSQL	4	12
27	Full Stack	4	14
28	Python	3	15
29	Software Engineering	4	19
30	data Structures and Algorithms	3	20

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 = 10;

SELECT * FROM enrollments;

SET foreign_key_checks=0;
DELETE FROM Students
WHERE student_id = 10;

SELECT * FROM students;
```

enrollment_id	student_id	course_id	enrollment_date
31	1	21	2023-01-05
33	3	21	2023-01-05
34	4	24	2023-06-25
35	5	25	2023-05-04
36	2	26	2023-09-15
37	4	27	2023-12-05
38	5	28	2023-11-10
39	9	29	2023-09-06
41	1	26	2024-01-07

student_id	first_name	last_name	date_of_birth	email	phone_number
1	Kavya	Kaja	2002-04-12	kavya@gmail.com	9734567890
2	Ramya	Gowda	2001-06-18	ramya@gmail.com	9876543210
3	Usha	Manda	2006-12-05	usha@gmail.com	8667890123
4	Raju	Kalaga	2005-11-15	raju@gmail.com	7434567890
5	Ravi	Mutta	2009-10-20	ravi@gmail.com	9876543210
6	Divya	Varma	2000-03-21	divya@gmail.com	9467890123
7	Ram	Ganta	2004-04-06	ram@gmail.com	8234567890
8	Mohan	Gurram	2003-11-03	mohan@gmail.com	9876543210
9	Lakshmi	Reddy	2004-01-11	lakshmi@gmail.com	6767890123
11	John	Doe	1995-08-15	john.doe@example.com	1234567890

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 = 18000
WHERE payment_id = 41;

SELECT * FROM Payments;
```

payment_id	student_id	amount	payment_date
41	1	18000.00	2023-01-10
42	2	7000.00	2023-03-15
43	3	10000.00	2023-01-10
44	4	9000.00	2023-06-30
45	5	13000.00	2023-05-08
46	2	8000.00	2023-09-20
47	4	16000.00	2023-12-10
48	5	12000.00	2023-11-15
49	9	15000.00	2023-09-12
50	10	15000.00	2023-09-12

Tasks 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.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 WHERE s.student_id = 2
GROUP BY s.student_id, s.first_name, s.last_name;
```

student_id	first_name	last_name	total_payments
2	Ramya	Gowda	15000.00

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
21	Introduction to Programming	2
22	Database Management	0
23	Web Development	0
24	Java	1
25	C Programming	1
26	MYSQL	2
27	Full Stack	1
28	Python	1
29	Software Engineering	1
30	data Structures and Algorithms	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
▶	6	Divya	Varma
	7	Ram	Ganta
	8	Mohan	Gurram
	11	John	Doe

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
LEFT OUTER JOIN courses c ON e.course_id = c.course_id;
```

	first_name	last_name	course_name
▶	Kavya	Kaja	Introduction to Programming
	Usha	Manda	Introduction to Programming
	Raju	Kalaga	Java
	Ravi	Mutta	C Programming
	Ramya	Gowda	MYSQL
	Raju	Kalaga	Full Stack
	Ravi	Mutta	Python
	Lakshmi	Reddy	Software Engineering
	Kavya	Kaja	MYSQL

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 AS teacher_first_name, t.last_name AS teacher_last_name, c.course_name
FROM teacher t
JOIN courses c ON t.teacher_id = c.teacher_id;
```

	teacher_first_name	teacher_last_name	course_name
▶	Surya	Pendyala	Introduction to Programming
	Ramesh	Jasthi	Database Management
	Pooja	Palakurthi	Web Development
	John	Paluri	Java
	Surya	Pendyala	C Programming
	Ramesh	Jasthi	MYSQL
	John	Paluri	Full Stack
	Sony	Konduri	Python
	Rohan	Katta	Software Engineering
	Rohith	Mavuri	data Structures and Algorithms

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
JOIN Courses c ON e.course_id = c.course_id
WHERE c.course_name = 'MYSQL';
```

	first_name	last_name	enrollment_date
▶	Ramya	Gowda	2023-09-15
	Kavya	Kaja	2024-01-07

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.student_id IS NULL;
```

	first_name	last_name
▶	Divya	Varma
	Ram	Ganta
	Mohan	Gurram
	John	Doe

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
▶	22	Database Management
	23	Web Development
	30	data Structures and Algorithms

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 e1.student_id, s.first_name, s.last_name, COUNT(e1.course_id) AS enrollments_count
FROM Enrollments e1 JOIN Students s ON e1.student_id = s.student_id
GROUP BY e1.student_id, s.first_name, s.last_name
HAVING COUNT(e1.course_id) > 1;
```

student_id	first_name	last_name	enrollments_count
1	Kavya	Kaja	2
4	Raju	Kalaga	2
5	Ravi	Mutta	2

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.course_id IS NULL;
```

teacher_id	first_name	last_name
16	Mary	Dasari
17	Chaya	Parimi
18	Pragna	Pathuri

Tasks 3: 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 c.course_id, c.course_name, AVG(enrollment_count) AS average_students_enrolled
FROM Courses c
JOIN (
    SELECT course_id, COUNT(DISTINCT student_id) AS enrollment_count
    FROM Enrollments
    GROUP BY course_id
) e ON c.course_id = e.course_id
GROUP BY c.course_id, c.course_name;
```

course_id	course_name	average_students_enrolled
21	Introduction to Programming	2.0000
24	Java	1.0000
25	C Programming	1.0000
26	MYSQL	2.0000
27	Full Stack	1.0000
28	Python	1.0000
29	Software Engineering	1.0000

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 AS highest_payment
FROM Students s
JOIN Payments p ON s.student_id = p.student_id
WHERE p.amount = (SELECT MAX(amount) FROM Payments);
```

student_id	first_name	last_name	highest_payment
1	Kavya	Kaja	18000.00

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, COUNT(e.enrollment_id) AS enrollment_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.enrollment_id) = (
        SELECT MAX(enrollment_count)
        FROM
            (
                SELECT course_id, COUNT(enrollment_id) AS enrollment_count
                FROM Enrollments
                GROUP BY course_id
            ) AS max_enrollments
    );
```

course_id	course_name	enrollment_count
21	Introduction to Programming	2
26	MYSQL	2

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 AS teacher_first_name, t.last_name AS teacher_last_name, COALESCE(SUM(p.amount), 0) AS total_payments
FROM Teacher t
LEFT 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	teacher_first_name	teacher_last_name	total_payments
11	Surya	Pendyala	53000.00
12	Ramesh	Jasthi	33000.00
13	Pooja	Palakurthi	0.00
14	John	Paluri	50000.00
15	Sony	Konduri	25000.00
16	Mary	Dasari	0.00
17	Chaya	Parimi	0.00
18	Pragna	Pathuri	0.00
19	Rohan	Katta	15000.00
20	Rohith	Mavuri	0.00

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
    (
        SELECT COUNT(c.course_id)
        FROM Courses c
        ) = (
        SELECT COUNT(DISTINCT e.course_id)
        FROM Enrollments e
        WHERE e.student_id = s.student_id
    );
```

student_id	first_name	last_name
NULL	NULL	NULL

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

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

	teacher_id	first_name	last_name
▶	16	Mary	Dasari
	17	Chaya	Parimi
	18	Pragna	Pathuri

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 student_ages;
```

	average_age
▶	20.2000

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
    );
```

	course_id	course_name
▶	22	Database Management
	23	Web Development
	30	data Structures and Algorithms

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, e.course_id, COALESCE(SUM(p.amount), 0) AS total_payments
FROM Enrollments e
LEFT JOIN Payments p ON e.student_id = p.student_id
GROUP BY e.student_id, e.course_id;
```

	student_id	course_id	total_payments
▶	1	21	18000.00
	3	21	10000.00
	4	24	25000.00
	5	25	25000.00
	2	26	15000.00
	4	27	25000.00
	5	28	25000.00
	9	29	15000.00
	1	26	18000.00

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

	student_id	first_name	last_name
▶	2	Ramya	Gowda
	4	Raju	Kalaga
	5	Ravi	Mutta

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, COALESCE(SUM(p.amount), 0) 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	Kavya	Kaja	18000.00
2	Ramya	Gowda	15000.00
3	Usha	Manda	10000.00
4	Raju	Kalaga	25000.00
5	Ravi	Mutta	25000.00
6	Divya	Varma	0.00
7	Ram	Ganta	0.00
8	Mohan	Gurram	0.00
9	Lakshmi	Reddy	15000.00
11	John	Doe	0.00

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 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_name	enrolled_students_count
Introduction to Programming	2
Database Management	0
Web Development	0
Java	1
C Programming	1
MYSQL	2
Full Stack	1
Python	1
Software Engineering	1
data Structures and Algorithms	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, AVG(p.amount) AS average_payment_amount
FROM Students s
JOIN Payments p ON s.student_id = p.student_id
GROUP BY s.student_id;
```

student_id	average_payment_amount
1	18000.000000
2	7500.000000
3	10000.000000
4	12500.000000
5	12500.000000
9	15000.000000