TASK 1

Query :1

CREATE DATABASE SISDB;

Query :2

create table Students(student_id bigint primary key, first_name text, last_name text, date_of_birth date, email text, phone_name bigint);

create table Teacher(teacher_id bigint primary key , first_name text, last_name text, email text);

CREATE TABLE Courses (

 course_id BIGINT PRIMARY KEY,

 course_name TEXT,

 credits BIGINT,

 teacher_id BIGINT,

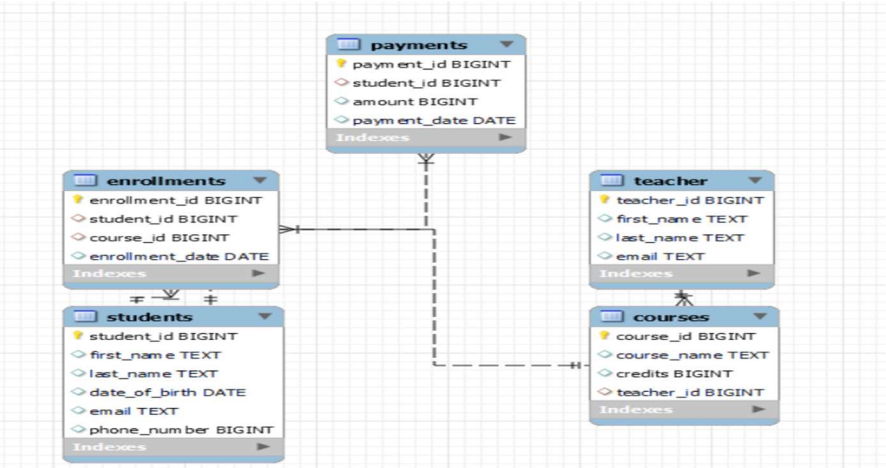
 FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)

);

create table Enrollments (enrollment_id bigint primary key, student_id bigint, course_id bigint, enrollment_date date, foreign key (student_id) REFERENCES Students(student_id),FOREIGN KEY (course_id) REFERENCES Courses(course_id));

create table Payments (payment_id bigint primary key,student_id bigint,amount bigint,payment_date date,foreign key (student_id) REFERENCES Students(student_id));

Query :3



Query :4

```
desc Students;

desc Teacher;

desc Courses;

desc Enrollments;

desc Payments;
```

Query 5:

```
INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number)
VALUES
 (1, 'John', 'Doe', '2000-01-15', 'john.doe@example.com', 1234567890),
 (2, 'Jane', 'Smith', '1999-05-22', 'jane.smith@example.com', 9876543210),
 (3, 'Alice', 'Johnson', '2002-08-10', 'alice.johnson@example.com', 5555555555),
 (4, 'Bob', 'Miller', '2001-03-05', 'bob.miller@example.com', 7777777777),
 (5, 'Eva', 'Brown', '1998-12-03', 'eva.brown@example.com', 9999999999),
 (6, 'Michael', 'Lee', '2003-02-18', 'michael.lee@example.com', 1111111111),
 (7, 'Sophia', 'Clark', '2000-07-27', 'sophia.clark@example.com', 2222222222),
 (8, 'Daniel', 'Taylor', '1999-11-14', 'daniel.taylor@example.com', 3333333333),
 (9, 'Olivia', 'Wright', '2004-04-30', 'olivia.wright@example.com', 4444444444),
 (10, 'Matthew', 'Turner', '2002-09-08', 'matthew.turner@example.com', 6666666666);

INSERT INTO Teacher (teacher_id, first_name, last_name, email)
VALUES
 (1, 'John', 'Doe', 'john.doe@example.com'),
 (2, 'Jane', 'Smith', 'jane.smith@example.com'),
 (3, 'Alice', 'Johnson', 'alice.johnson@example.com'),
 (4, 'Bob', 'Miller', 'bob.miller@example.com'),
 (5, 'Eva', 'Brown', 'eva.brown@example.com'),
 (6, 'Michael', 'Lee', 'michael.lee@example.com'),
 (7, 'Sophia', 'Clark', 'sophia.clark@example.com'),
 (8, 'Daniel', 'Taylor', 'daniel.taylor@example.com'),
```

```
 (9, 'Olivia', 'Wright', 'olivia.wright@example.com'),
 (10, 'Matthew', 'Turner', 'matthew.turner@example.com');

INSERT INTO Courses (course_id, course_name, credits, teacher_id)
VALUES
 (101, 'Mathematics', 3, 1),
 (102, 'Physics', 4, 2),
 (103, 'English Literature', 3, 3),
 (104, 'Computer Science', 4, 4),
 (105, 'History', 3, 5),
 (106, 'Chemistry', 4, 6),
 (107, 'Art', 3, 7),
 (108, 'Music', 2, 8),
 (109, 'Biology', 4, 9),
 (110, 'Economics', 3, 10);
INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
VALUES
 (1, 1, 101, '2023-01-10'),
 (2, 2, 102, '2023-01-15'),
 (3, 3, 103, '2023-01-20'),
 (4, 4, 104, '2023-01-25'),
 (5, 5, 105, '2023-02-01'),
 (6, 6, 106, '2023-02-05'),
 (7, 7, 107, '2023-02-10'),
 (8, 8, 108, '2023-02-15'),
 (9, 9, 109, '2023-02-20'),
 (10, 10, 110, '2023-02-25');

INSERT INTO Payments (payment_id, student_id, amount, payment_date)
VALUES
 (1, 1, 500, '2023-03-05'),
```

(2, 2, 750, '2023-03-10'),

(3, 3, 600, '2023-03-15'),

(4, 4, 900, '2023-03-20'),

(5, 5, 400, '2023-03-25'),

(6, 6, 550, '2023-04-01'),

(7, 7, 700, '2023-04-05'),

(8, 8, 800, '2023-04-10'),

(9, 9, 950, '2023-04-15'),

(10, 10, 600, '2023-04-20');

TASK 2

Query1

insert into Students values('11','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	John	Doe	2000-01-15	john.doe@example.com	1234567890
2	Jane	Smith	1999-05-22	jane.smith@example.com	9876543210
3	Alice	Johnson	2002-08-10	alice.johnson@example.com	5555555555
4	Bob	Miller	2001-03-05	bob.miller@example.com	7777777777
5	Eva	Brown	1998-12-03	eva.brown@example.com	9999999999
6	Michael	Lee	2003-02-18	michael.lee@example.com	1111111111
7	Sophia	Clark	2000-07-27	sophia.clark@example.com	2222222222
8	Daniel	Taylor	1999-11-14	daniel.taylor@example.com	3333333333
9	Olivia	Wright	2004-04-30	olivia.wright@example.com	4444444444
10	Matthew	Turner	2002-09-08	matthew.turner@example.com	6666666666
11	john	doe	1995-08-15	john.doe@example.com	1234567890
NULL	NULL	NULL	NULL	NULL	NULL

Query2

INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)

VALUES (11, 1, 101, CURRENT_DATE);

select * from Enrollments;

enrollment_id	student_id	course_id	enrollment_date
2	2	102	2023-01-15
3	3	103	2023-01-20
4	4	104	2023-01-25
5	5	105	2023-02-01
6	6	106	2023-02-05
7	7	107	2023-02-10
8	8	108	2023-02-15
9	9	109	2023-02-20
10	10	110	2023-02-25
11	1	101	2023-12-09
NULL	NULL	NULL	NULL

Query3

-- Assuming teacher_id = 1 and the new email address is 'new.email@example.com'

UPDATE Teacher

SET email = 'new.email@example.com'

WHERE teacher_id = 1;

select * from Teacher;

teacher_id	first_name	last_name	email
1	John	Doe	new.email@example.com
2	Jane	Smith	jane.smith@example.com
3	Alice	Johnson	alice.johnson@example.com
4	Bob	Miller	bob.miller@example.com
5	Eva	Brown	eva.brown@example.com
6	Michael	Lee	michael.lee@example.com
7	Sophia	Clark	sophia.clark@example.com
8	Daniel	Taylor	daniel.taylor@example.com
9	Olivia	Wright	olivia.wright@example.com
10	Matthew	Turner	matthew.turner@example.com
NULL	NULL	NULL	NULL

Query4

-- Assuming enrollment_id = 1 (replace with the actual enrollment_id)

DELETE FROM Enrollments

WHERE enrollment_id = 1;

select * from Enrollments;

enrollment_id	student_id	course_id	enrollment_date
2	2	102	2023-01-15
3	3	103	2023-01-20
4	4	104	2023-01-25
5	5	105	2023-02-01
6	6	106	2023-02-05
7	7	107	2023-02-10
8	8	108	2023-02-15
9	9	109	2023-02-20
10	10	110	2023-02-25
11	1	101	2023-12-09
NULL	NULL	NULL	NULL

Query5

-- Assuming course_id = 101 and teacher_id = 1

UPDATE Courses

SET teacher_id = 1

WHERE course_id = 109;

select * from Courses;

course_id	course_name	credits	teacher_id
101	Mathematics	3	1
102	Physics	4	2
103	English Literature	3	3
104	Computer Science	4	4
105	History	3	5
106	Chemistry	4	6
107	Art	3	7
108	Music	2	8
109	Biology	4	1
110	Economics	3	10
NULL	NULL	NULL	NULL

Query6

-- Assuming student_id = 1 (replace with the actual student_id)

DELETE FROM Enrollments

WHERE student_id = 1;

DELETE FROM Payments WHERE student_id = 1;

DELETE FROM Students

WHERE student_id = 1;

select * from Enrollments;

select * from Students;

enrollment_id	student_id	course_id	enrollment_date
3	3	103	2023-01-20
4	4	104	2023-01-25
5	5	105	2023-02-01
6	6	106	2023-02-05
7	7	107	2023-02-10
8	8	108	2023-02-15
9	9	109	2023-02-20
10	10	110	2023-02-25
NULL	NULL	NULL	NULL

student_id	first_name	last_name	date_of_birth	email	phone_number
2	Jane	Smith	1999-05-22	jane.smith@example.com	9876543210
3	Alice	Johnson	2002-08-10	alice.johnson@example.com	5555555555
4	Bob	Miller	2001-03-05	bob.miller@example.com	7777777777
5	Eva	Brown	1998-12-03	eva.brown@example.com	9999999999
6	Michael	Lee	2003-02-18	michael.lee@example.com	1111111111
7	Sophia	Clark	2000-07-27	sophia.clark@example.com	2222222222
8	Daniel	Taylor	1999-11-14	daniel.taylor@example.com	3333333333
9	Olivia	Wright	2004-04-30	olivia.wright@example.com	4444444444
10	Matthew	Turner	2002-09-08	matthew.turner@example.com	6666666666
11	john	doe	1995-08-15	john.doe@example.com	1234567890
NULL	NULL	NULL	NULL	NULL	NULL

Query7

-- Assuming payment_id = 1 and the new payment amount is 800

UPDATE Payments

SET amount = 800

WHERE payment_id = 1;

select * from Payments;

payment_id	student_id	amount	payment_date
2	2	750	2023-03-10
3	3	600	2023-03-15
4	4	900	2023-03-20
5	5	400	2023-03-25
6	6	550	2023-04-01
7	7	700	2023-04-05
8	8	800	2023-04-10
9	9	950	2023-04-15
10	10	600	2023-04-20
NULL	NULL	NULL	NULL

-- Task 3: SQL Queries

-- 1. Calculate the total payments made by a specific student

```
SELECT student_id, SUM(amount) AS total_payments
FROM Payments
WHERE student_id = 3
GROUP BY student_id;
```

student_id	total_payments
3	600

-- 2. List of courses with the count of students enrolled in each course

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

course_id	course_name	enrolled_students
101	Mathematics	0
102	Physics	0
103	English Literature	1
104	Computer Science	1
105	History	1
106	Chemistry	1
107	Art	1
108	Music	1
109	Biology	1
110	Economics	1

-- 3. Names of students who have not enrolled in any course

```
SELECT s.student_id, first_name, 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
▶	2	Jane	Smith
	11	john	doe

-- 4. Retrieve the names of students and the names of courses they are enrolled in

```
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;
```

	first_name	last_name	course_name
▶	Alice	Johnson	English Literature
	Bob	Miller	Computer Science
	Eva	Brown	History
	Michael	Lee	Chemistry
	Sophia	Clark	Art
	Daniel	Taylor	Music
	Olivia	Wright	Biology
	Matthew	Turner	Economics

-- 5. List of teachers and the courses they are assigned to

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

	first_name	last_name	course_name
▶	John	Doe	Mathematics
	John	Doe	Biology
	Jane	Smith	Physics
	Alice	Johnson	English Literature
	Bob	Miller	Computer Science
	Eva	Brown	History
	Michael	Lee	Chemistry
	Sophia	Clark	Art
	Daniel	Taylor	Music
	Matthew	Turner	Economics

-- 6. List of students and their enrollment dates for a specific course

```
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_id = 103;
```

	first_name	last_name	enrollment_date
▶	Alice	Johnson	2023-01-20

-- 7. Names of students who have not made any payments

```
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

-- 8. Courses that have no enrollments

```
SELECT c.course_id, 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
▶	101	Mathematics
	102	Physics

-- 9. Students enrolled in more than one course

```
SELECT s.student_id, s.first_name, s.last_name
FROM Enrollments e1
JOIN Enrollments e2 ON e1.student_id = e2.student_id AND e1.course_id <> e2.course_id
JOIN Students s ON e1.student_id = s.student_id;
```

-- 10. Teachers not assigned to any courses

```
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
▶	9	Olivia	Wright

Task 4: Subquery Queries

-- 1. Average number of students enrolled in each course

SELECT AVG(enrolled_students)

FROM (SELECT course_id, COUNT(student_id) AS enrolled_students FROM Enrollments GROUP BY course_id) AS CourseEnrollments;

	AVG(enrolled_students)
▶	1.0000

-- **2. Student(s) who made the highest payment**

SELECT student_id, MAX(amount) AS highest_payment

FROM Payments

GROUP BY student_id

ORDER BY highest_payment DESC

LIMIT 1;

	student_id	highest_payment
▶	9	950

-- 3. Retrieve a list of courses with the highest number of enrollments

SELECT course_id, course_name, MAX(enrollment_count) AS highest_enrollments

FROM (

  SELECT c.course_id, course_name, COUNT(e.student_id) AS enrollment_count

```
    FROM Courses c
    LEFT JOIN Enrollments e ON c.course_id = e.course_id
    GROUP BY c.course_id, course_name
) AS CourseEnrollments
GROUP BY course_id, course_name;
```

course_id	course_name	highest_enrollments
101	Mathematics	0
102	Physics	0
103	English Literature	1
104	Computer Science	1
105	History	1
106	Chemistry	1
107	Art	1
108	Music	1
109	Biology	1
110	Economics	1

-- 4. Calculate the total payments made to courses taught by each teacher

```
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
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;
```

teacher_id	first_name	last_name	total_payments
3	Alice	Johnson	600
4	Bob	Miller	900
5	Eva	Brown	400
6	Michael	Lee	550
7	Sophia	Clark	700
8	Daniel	Taylor	800
1	John	Doe	950
10	Matthew	Turner	600

-- 5. Identify students who are enrolled in all available courses

```
SELECT student_id, first_name, last_name
FROM Students
WHERE (
```

```
   SELECT COUNT(DISTINCT course_id) FROM Courses
) = (
   SELECT COUNT(DISTINCT course_id) FROM Enrollments WHERE student_id = Students.student_id
);
```

-- 6. Retrieve the names of teachers who have not been assigned to any courses

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

teacher_id	first_name	last_name
9	Olivia	Wright
NULL	NULL	NULL

-- 7. Calculate the average age of all students

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

average_age
22.7000

-- 8. Identify courses with no enrollments

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

course_id	course_name
101	Mathematics
102	Physics
NULL	NULL

-- 9. Calculate the total payments made by each student for each course they are enrolled in

```
SELECT s.student_id, s.first_name, s.last_name, c.course_name, SUM(p.amount) AS total_payments
FROM Students s
JOIN Payments p ON s.student_id = p.student_id
```

JOIN Enrollments e ON s.student_id = e.student_id

JOIN Courses c ON e.course_id = c.course_id

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

student_id	first_name	last_name	course_name	total_payments
3	Alice	Johnson	English Literature	600
4	Bob	Miller	Computer Science	900
5	Eva	Brown	History	400
6	Michael	Lee	Chemistry	550
7	Sophia	Clark	Art	700
8	Daniel	Taylor	Music	800
9	Olivia	Wright	Biology	950
10	Matthew	Turner	Economics	600

-- 10. Identify students who have made more than one payment

SELECT student_id, COUNT(payment_id) AS payment_count

FROM Payments

GROUP BY student_id

HAVING payment_count > 1;

-- 11. Write an SQL query to calculate the total payments made by 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
2	Jane	Smith	750
3	Alice	Johnson	600
4	Bob	Miller	900
5	Eva	Brown	400
6	Michael	Lee	550
7	Sophia	Clark	700
8	Daniel	Taylor	800
9	Olivia	Wright	950
10	Matthew	Turner	600
11	john	doe	NULL

-- 12. Retrieve a list of course names along with the count of students enrolled in each course

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

course_id	course_name	enrolled_students
101	Mathematics	0
102	Physics	0
103	English Literature	1
104	Computer Science	1
105	History	1
106	Chemistry	1
107	Art	1
108	Music	1
109	Biology	1
110	Economics	1

-- 13. Calculate the average payment amount made by students

```
SELECT AVG(amount) AS average_payment
FROM Payments;
```

average_payment
694.4444