Student Information System (SIS)

Task 1. Database Design

1. Create the database named "SISDB"

```
create database 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 TABLE Students (
    student_id INT PRIMARY KEY IDENTITY(1,1),
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    date_of_birth DATE,
    email VARCHAR(100) UNIQUE,
    phone_number VARCHAR(15)
);

CREATE TABLE Courses (
    course_id INT PRIMARY KEY IDENTITY(1,1),
    course_name VARCHAR(100),
    credits INT CHECK (credits > 0),
    teacher_id INT,
    FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
```

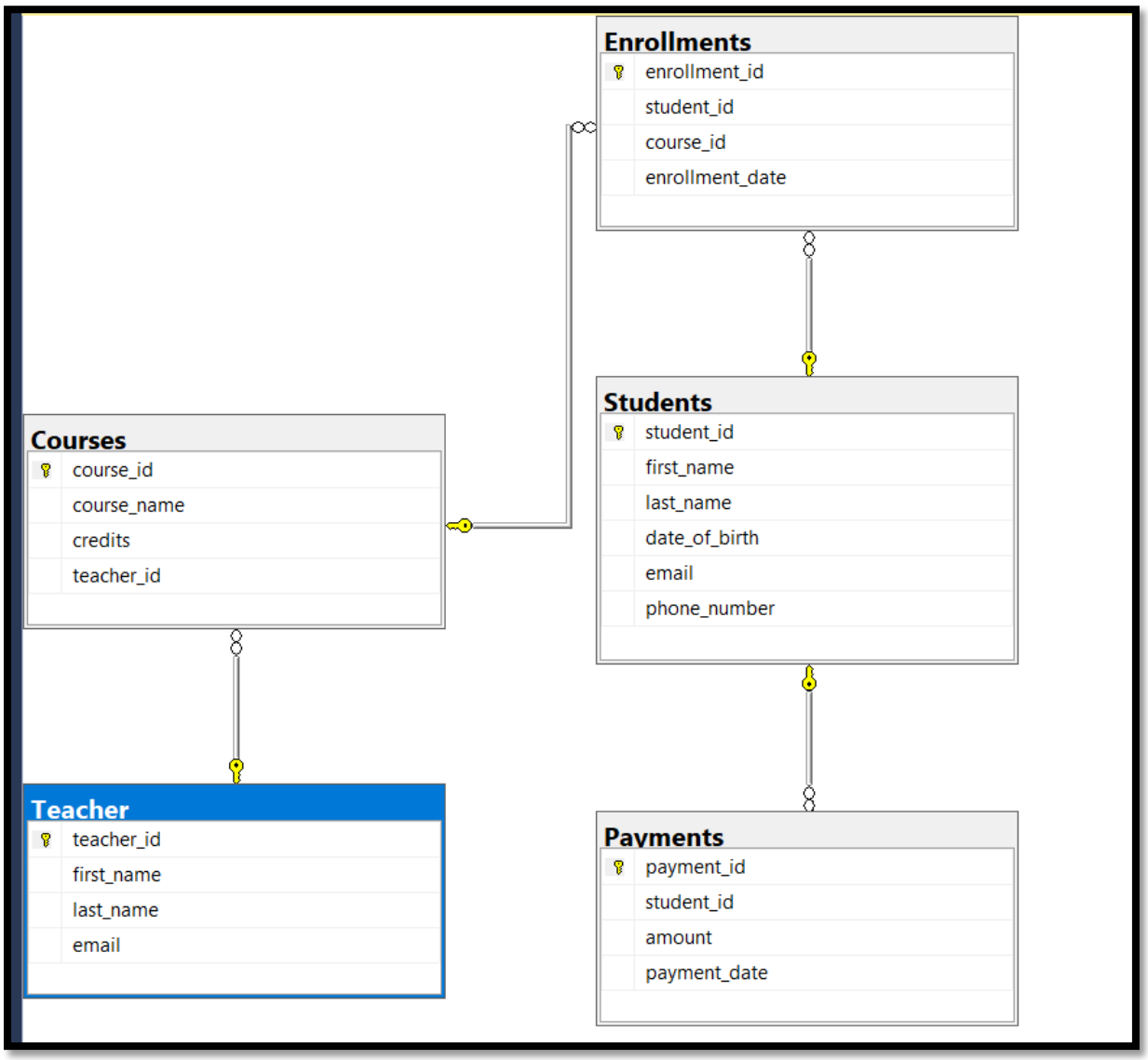
```
);

CREATE TABLE Enrollments (
    enrollment_id INT PRIMARY KEY IDENTITY(1,1),
    student_id INT,
    course_id INT,
    enrollment_date DATE DEFAULT GETDATE(),
    FOREIGN KEY (student_id) REFERENCES Students(student_id),
    FOREIGN KEY (course_id) REFERENCES Courses(course_id)
);

CREATE TABLE Teacher (
    teacher_id INT PRIMARY KEY IDENTITY(1,1),
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100) UNIQUE
);

CREATE TABLE Payments (
    payment_id INT PRIMARY KEY IDENTITY(1,1),
    student_id INT NOT NULL,
    amount DECIMAL(10, 2) NOT NULL,
    payment_date DATE NOT NULL,
    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 (first_name, last_name, date_of_birth, email, phone_number)
VALUES
('Amit', 'Sharma', '2000-01-10', 'amit.sharma@example.com', '9876543210'),
('Sneha', 'Patil', '1999-05-15', 'sneha.patil@example.com', '9876543211'),
('Raj', 'Mehta', '2001-02-20', 'raj.mehta@example.com', '9876543212'),
('Anjali', 'Kumar', '1998-07-25', 'anjali.kumar@example.com', '9876543213'),
('Karan', 'Joshi', '2000-12-30', 'karan.joshi@example.com', '9876543214'),
('Neha', 'Deshmukh', '2002-04-10', 'neha.deshmukh@example.com', '9876543215'),
('Ravi', 'Yadav', '2001-06-18', 'ravi.yadav@example.com', '9876543216'),
('Pooja', 'Nair', '1999-09-09', 'pooja.nair@example.com', '9876543217'),
('Arjun', 'Rao', '2000-11-11', 'arjun.rao@example.com', '9876543218'),
('Divya', 'Iyer', '2001-03-03', 'divya.iyer@example.com', '9876543219');
```

```
INSERT INTO Courses (course_name, credits, teacher_id) VALUES
('Data Structures', 3, 1),
('Operating Systems', 4, 2),
('Database Management', 3, 3),
('Computer Networks', 4, 4),
('Machine Learning', 3, 5),
('Web Development', 2, 1),
('Software Engineering', 3, 2),
('AI Fundamentals', 3, 3),
('Cloud Computing', 3, 4),
```

```
('Cybersecurity', 3, 5);

INSERT INTO Enrollments (student_id, course_id) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(10, 10);

INSERT INTO Teacher (first_name, last_name, email) VALUES
('Suresh', 'Pillai', 'suresh.pillai@college.edu'),
('Meena', 'Bhat', 'meena.bhat@college.edu'),
('Rahul', 'Khanna', 'rahul.khanna@college.edu'),
('Geeta', 'Singh', 'geeta.singh@college.edu'),
('Anil', 'Verma', 'anil.verma@college.edu');

INSERT INTO Payments (student_id, amount, payment_date) VALUES
(1, 5000.00, '2025-01-15'),
(2, 5200.00, '2025-01-18'),
(3, 4800.00, '2025-02-01'),
(4, 5300.00, '2025-02-12'),
(5, 5000.00, '2025-03-05'),
(6, 5100.00, '2025-03-20'),
(7, 4950.00, '2025-04-10'),
(8, 5050.00, '2025-04-18'),
```

```
(9, 5000.00, '2025-05-02'),
(10, 4900.00, '2025-05-15');

INSERT INTO Payments (student_id, amount) VALUES
(1, 5000.00),
(2, 5200.00),
(3, 4800.00),
(4, 5300.00),
(5, 5000.00),
(6, 5100.00),
(7, 4950.00),
(8, 5050.00),
(9, 5000.00),
(10, 4900.00);

INSERT INTO Payments (student_id, amount) VALUES
(1, 5000.00),
(2, 5200.00),
(3, 4800.00),
(4, 5300.00),
(5, 5000.00),
(6, 5100.00),
(7, 4950.00),
(8, 5050.00),
(9, 5000.00),
(10, 4900.00);
```

Task 2

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);
```

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 (3, 2, '2025-06-13');
```

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 = 'anil.verma2@college.edu' where teacher_id = 5;
```

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 = 10
```

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 = 4 where course_id = 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 payments
where student_id = 5;

delete from enrollments
where student_id = 5;

delete from students
where student_id = 5;
```

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 = 5500.00 where payment_id = 4;
```

Task 3

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 payments p join students s on p.student_id = s.student_id
where s.student_id = 3
group by s.first_name, s.last_name;
```

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(student_id) as 'Total_enrolled' from courses c join enrollments e on c.course_id = e.course_id group by c.course_name
```

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
```

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
```

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 join courses c on t.teacher_id = c.teacher_id;
```

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 = 'data structures';
```

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_name
from courses c
left join enrollments e on c.course_id = e.course_id
where e.enrollment_id is null;
```

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 distinct 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. 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;
```