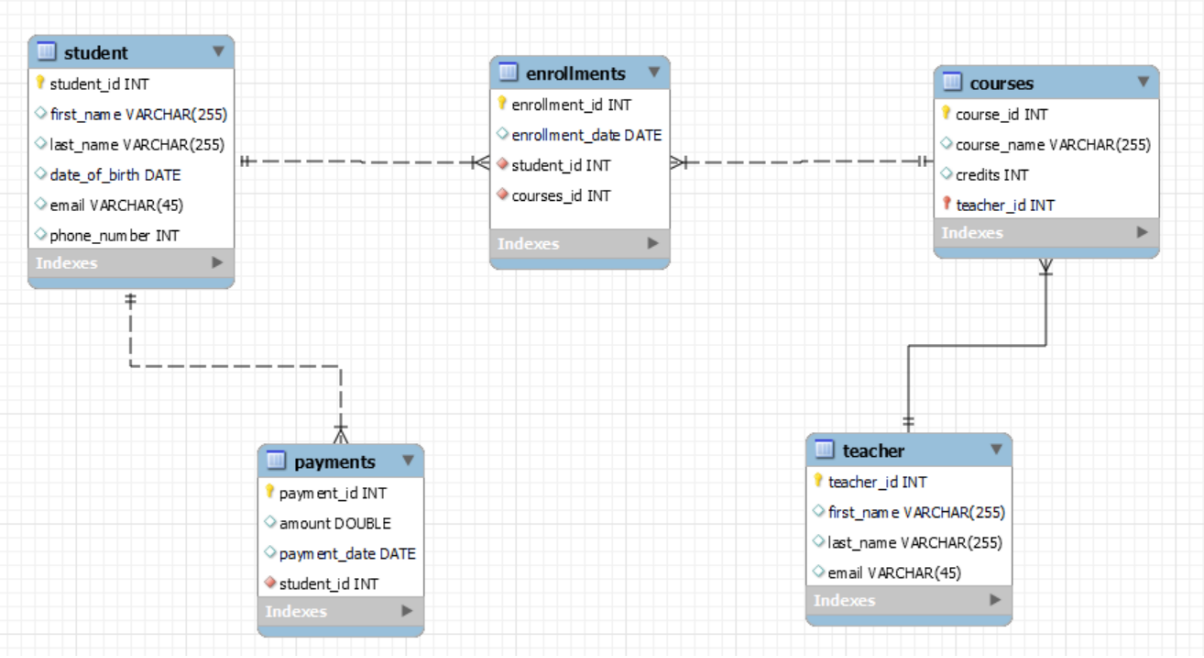
ASSIGNMENT – 2 Student Information System (SIS)

ER DIAGRAM:



CODE:

```
CREATE DATABASE SISDB;
USE SISDB;
-- Create the 'student' table
CREATE TABLE student (
    student_id INT PRIMARY KEY,
    first_name VARCHAR(255),
```

```
    last_name VARCHAR(255),
    birth_date DATE,
    email VARCHAR(255),
    phone_number VARCHAR(255)
);

-- Insert data into the 'student' table
INSERT INTO student VALUES
  (1, 'sethu', 'Deepika','1990-01-01', 'sethu.deepika@email.com', '1234567890'),
  (2, 'Leena', 'Devi', '1992-05-15', 'leena.devi@email.com', '9876543210'),
  (3, 'Mani', 'Malar', '1988-08-20', 'mani.malar@email.com', '5551234567'),
  (4, 'Emily', 'Johnson', '1995-03-10','emily.j@email.com', '3216549870'),
  (5, 'Haritha', 'Karthikeyan','1993-12-05', 'haritha.k@email.com', '7778889999'),
  (6, 'Sophia', 'Miller', '1991-06-18', 'sophia.m@email.com', '4445556666'),
  (7, 'Olivia', 'Davis', '1989-09-22',  'oliver.d@email.com', '1237894560'),
  (8, 'Priyanka', 'Ramakrishna', '1994-11-28','priya.rama@email.com', '1112223333'),
  (9, 'Diya', 'Suraj','1996-04-15', 'diya.suraj@email.com', '9998887777'),
  (10, 'Avanthika', 'Jain','1997-07-12', 'avanthika.jain@email.com', '5557778888');

 select * from student;

-- Create the 'teacher' table
CREATE TABLE teacher (
```

```
    teacher_id INT PRIMARY KEY,
    first_name VARCHAR(255),
    last_name VARCHAR(255),
    email VARCHAR(255)
);

-- Insert data into the 'teacher' table
INSERT INTO teacher VALUES
   (1, 'Professor', 'Johnson', 'prof.johnson@email.com'),
   (2, 'Professor', 'Smith', 'prof.smith@email.com'),
   (3, 'Professor', 'Brown', 'prof.brown@email.com'),
   (4, 'Professor', 'Miller', 'prof.miller@email.com'),
   (5, 'Professor', 'Anderson', 'prof.anderson@email.com'),
   (6, 'Professor', 'Williams', 'prof.williams@email.com'),
   (7, 'Professor', 'Taylor', 'prof.taylor@email.com'),
   (8, 'Professor', 'Davis', 'prof.davis@email.com'),
   (9, 'Professor', 'Harris', 'prof.harris@email.com'),
   (10, 'Professor', 'Clark', 'prof.clark@email.com');

select * from teacher;
-- Create the '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 student(student_id)
);

-- Insert data into the 'payments' table
INSERT INTO payments VALUES
  (1, 1, 500.00, '2024-03-01'),
  (2, 2, 750.50, '2024-03-05'),
  (3, 3, 1200.75, '2024-03-10'),
  (4, 4, 850.25, '2024-03-15'),
  (5, 5, 960.00, '2024-03-20'),
  (6, 6, 620.50, '2024-03-25'),
  (7, 7, 1100.00, '2024-03-30'),
  (8, 8, 480.75, '2024-04-02'),
  (9, 9, 900.00, '2024-04-07'),
  (10, 10, 800.50, '2024-04-12');
        select * from payments;

-- Create the 'courses' table
CREATE TABLE courses (
    course_id INT PRIMARY KEY,
    course_name VARCHAR(255),
    credits INT,
    teacher_id INT,
    FOREIGN KEY (teacher_id) REFERENCES teacher(teacher_id)
);
```

```
-- Insert data into the 'courses' table
INSERT INTO courses VALUES
  (101, 'Mathematics', 3, 1),
  (102, 'History', 4, 2),
  (103, 'Computer Science', 5, 3),
  (104, 'Physics', 4, 4),
  (105, 'Literature', 3, 5),
  (106, 'Chemistry', 4, 6),
  (107, 'Biology', 4, 7),
  (108, 'Art', 3, 8),
  (109, 'Music', 2, 9),
  (110, 'Economics', 5, 10);
        select * from courses;

-- Create the 'enrollments' table
CREATE TABLE enrollments (
   enrollment_id INT PRIMARY KEY,
   student_id INT,
   course_id INT,
   enrollment_date DATE,
   FOREIGN KEY (student_id) REFERENCES student(student_id),
   FOREIGN KEY (course_id) REFERENCES courses(course_id)
);

-- Insert data into the 'enrollments' table
INSERT INTO enrollments VALUES
```

```
  (1, 1, 101, '2024-01-15'),
  (2, 2, 102, '2024-02-20'),
  (3, 3, 103, '2024-03-10'),
  (4, 4, 104, '2024-04-05'),
  (5, 5, 105, '2024-05-12'),
  (6, 6, 106, '2024-06-18'),
  (7, 7, 107, '2024-07-22'),
  (8, 8, 108, '2024-08-28'),
  (9, 9, 109, '2024-09-15'),
  (10, 10, 110, '2024-10-10');

        select * from enrollments;


/*-----------------------task2----------------------*/

-- 1 Write an SQL query to insert a new student into the "Students" table.
INSERT INTO student VALUES
  (11, 'John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');

-- 2 Write an SQL query to enroll a student in a course.
INSERT INTO enrollments VALUES
  (11, 11, 103, '2024-03-15');

-- 3 Update the email address of a specific teacher in the "Teacher" table.
UPDATE teacher SET email = 'prof.smith@email.com' WHERE teacher_id = 1;
```

-- 4 Write an SQL query to delete a specific enrollment record from the "Enrollments" table.

```
DELETE FROM enrollments WHERE student_id = 3 AND course_id = 103;
```

-- 5 Update the "Courses" table to assign a specific teacher to a course.

```
UPDATE courses SET teacher_id = 5 WHERE course_id = 105;
```

-- 6 Delete a specific student from the "Students" table and remove all their enrollment records.

```
DELETE FROM student WHERE student_id = 6;
```

-- Remove enrollments for the deleted student

```
DELETE FROM enrollments WHERE student_id = 6;
```

-- 7 Update the payment amount for a specific payment record in the "Payments" table.

```
UPDATE payments SET amount = 600.50 WHERE payment_id = 2;
```

/*----------task3---------*/

-- 3.1 Write an SQL query to calculate the total payments made by a specific student.

```
SELECT s.first_name, s.last_name, SUM(p.amount) AS total_payments
FROM student s
JOIN payments p ON s.student_id = p.student_id
WHERE s.student_id = 1;
```

-- 3.2 Write an SQL query to retrieve a list of courses along with the count of students enrolled in each course.

```
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_name;
```

-- 3.3 Write an SQL query to find the names of students who have not enrolled in any course.

```
SELECT s.first_name, s.last_name
FROM student s
LEFT JOIN enrollments e ON s.student_id = e.student_id
WHERE e.enrollment_id IS NULL OR s.student_id IS NULL;
```

-- 3.4 Write an SQL query to retrieve the first name, last name of students, and the names of the courses they are enrolled in.

```
SELECT s.first_name, s.last_name, c.course_name
FROM student s
JOIN enrollments e ON s.student_id = e.student_id
JOIN courses c ON e.course_id = c.course_id;
```

-- 3.5 Create a query to list the names 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;

-- 3.6 Retrieve a list of students and their enrollment dates for a specific course.
SELECT s.first_name, s.last_name, e.enrollment_date
FROM student 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;


-- 3.7 Find the names of students who have not made any payments.
SELECT s.first_name, s.last_name
FROM student s
LEFT JOIN payments p ON s.student_id = p.student_id
WHERE p.payment_id IS NULL;


-- 3.8 Write a query to identify courses that have no enrollments.
SELECT c.course_name
FROM courses c
LEFT JOIN enrollments e ON c.course_id = e.course_id
WHERE e.enrollment_id IS NULL;

-- 3.9 Identify students who are enrolled in more than one course.
SELECT s.first_name, s.last_name, COUNT(e.enrollment_id) AS
enrolled_courses_count
```

```
FROM student s
JOIN enrollments e ON s.student_id = e.student_id
GROUP BY s.student_id
HAVING COUNT(e.enrollment_id) > 1;


-- 3.10 Find teachers who are not assigned to any courses.
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;

-- TASK 4

-- 4.1 Write an SQL query to calculate the average number of students enrolled in each course.
SELECT c.course_name, AVG(enrollment_count) AS avg_students_enrolled
FROM courses c
JOIN (SELECT course_id, COUNT(student_id) AS enrollment_count
    FROM enrollments
    GROUP BY course_id) e ON c.course_id = e.course_id
GROUP BY c.course_name;

-- 4.2 Identify the student(s) who made the highest payment.
SELECT s.first_name, s.last_name, p.amount
FROM student s
JOIN payments p ON s.student_id = p.student_id
```

```
WHERE p.amount = (SELECT MAX(amount) FROM payments);

-- 4.3 Retrieve a list of courses with the highest number of enrollments.
SELECT c.course_name, enrollment_count
FROM courses c
JOIN (SELECT course_id, COUNT(student_id) AS enrollment_count
    FROM enrollments
    GROUP BY course_id
    ORDER BY COUNT(student_id) DESC
    LIMIT 1) e ON c.course_id = e.course_id;

-- 4.4 Calculate the total payments made to courses taught by each teacher.
SELECT 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.first_name, t.last_name;

-- 4.5 Identify students who are enrolled in all available courses.
SELECT s.first_name, s.last_name
FROM student s
WHERE (SELECT COUNT(DISTINCT course_id) FROM courses) =
    (SELECT COUNT(DISTINCT course_id) FROM enrollments e WHERE
s.student_id = e.student_id);
```

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

```
SELECT t.first_name, t.last_name

FROM teacher t

WHERE t.teacher_id NOT IN (SELECT DISTINCT teacher_id FROM courses);
```

-- 4.7 Calculate the average age of all students.

```
SELECT AVG(DATEDIFF(CURDATE(), s.birth_date) / 365) AS average_age

FROM student s;
```

-- 4.8 Identify courses with no enrollments.

```
SELECT course_name

FROM courses c

WHERE NOT EXISTS (SELECT 1 FROM enrollments e WHERE c.course_id = e.course_id);
```

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

```
SELECT s.first_name, s.last_name, c.course_name, SUM(p.amount) AS total_payments

FROM student 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.first_name, s.last_name, c.course_name;
```

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

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

-- 4.11 Write an SQL query to calculate the total payments made by each student.
SELECT s.first_name, s.last_name, SUM(p.amount) AS total_payments
FROM student s
JOIN payments p ON s.student_id = p.student_id
GROUP BY s.first_name, s.last_name;

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

-- 4.13 Calculate the average payment amount made by students.
SELECT AVG(amount) AS average_payment_amount
FROM payments;
```