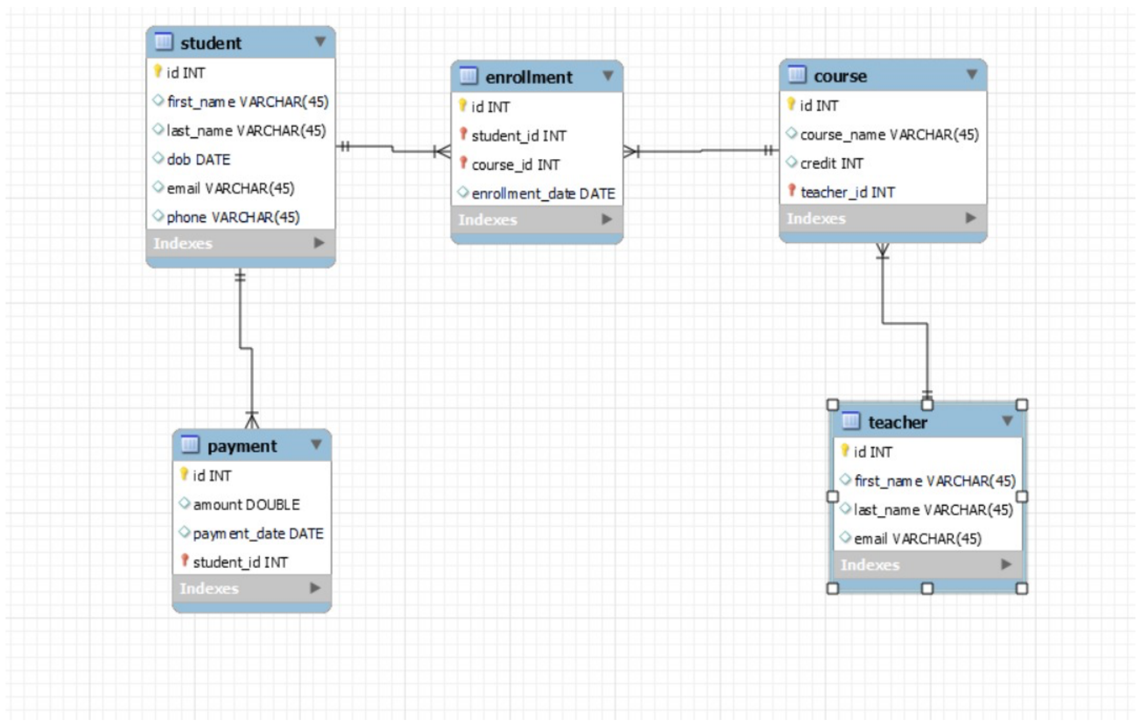
Project- Student Information System

ER Diagram:



Code:

```
create database learnings2;
use learnings2;

CREATE TABLE IF NOT EXISTS Students (
   student_id INT PRIMARY KEY AUTO_INCREMENT,
   first_name VARCHAR(50),
   last_name VARCHAR(50),
   date_of_birth DATE,
   email VARCHAR(100),
   phone_number VARCHAR(15)
);

CREATE TABLE IF NOT EXISTS Teacher (
   teacher_id INT PRIMARY KEY AUTO_INCREMENT,
   first_name VARCHAR(50),
   last_name VARCHAR(50),
   email VARCHAR(100)
);
CREATE TABLE IF NOT EXISTS Courses (
   course_id INT PRIMARY KEY AUTO_INCREMENT,
   course_name VARCHAR(100),
   credits INT,
   teacher_id INT,
   FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
);


CREATE TABLE IF NOT EXISTS Enrollments (
   enrollment_id INT PRIMARY KEY AUTO_INCREMENT,
   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)
);

CREATE TABLE IF NOT EXISTS Payments (
   payment_id INT PRIMARY KEY AUTO_INCREMENT,
   student_id INT,
   amount DECIMAL(10, 2),
   payment_date DATE,
   FOREIGN KEY (student_id) REFERENCES Students(student_id)
);

INSERT INTO Students (first_name, last_name, date_of_birth, email, phone_number)
VALUES
('Arjun', 'Kumar', '1998-05-15', 'arjun.kumar@example.com', '9876543210'),
('Divya', 'Srinivasan', '1999-08-20', 'divya.s@example.com', '8765432109'),
('Rajesh', 'Naidu', '1997-03-10', 'rajesh.n@example.com', '7654321098'),
('Priya', 'Rajendran', '1996-11-25', 'priya.r@example.com', '6543210987'),
('Ganesh', 'Iyer', '1998-09-30', 'ganesh.i@example.com', '5432109876'),
('Sneha', 'Reddy', '1997-06-12', 'sneha.r@example.com', '4321098765'),
('Karthik', 'Menon', '1999-02-28', 'karthik.m@example.com', '3210987654'),
('Meera', 'Gopal', '1998-07-05', 'meera.g@example.com', '2109876543'),
('Ananya', 'Chowdhury', '1997-04-18', 'ananya.c@example.com', '1098765432'),
('Rahul', 'Venkatesh', '1996-12-03', 'rahul.v@example.com', '0987654321');


INSERT INTO Teacher (first_name, last_name, email)
VALUES
('Suresh', 'Kumar', 'suresh.k@example.com'),
('Meenakshi', 'Nair', 'meenakshi.n@example.com'),
```

```sql
('Prakash', 'Raj', 'prakash.r@example.com'),
('Divya', 'Menon', 'divya.m@example.com'),
('Rajesh', 'Srinivasan', 'rajesh.s@example.com');


INSERT INTO Courses (course_name, credits, teacher_id)
VALUES
('Mathematics', 4, 1),
('Physics', 3, 2),
('Computer Science', 3, 3),
('Chemistry', 4, 4),
('Biology', 3, 5),
('English Literature', 3, 1),
('History', 4, 2),
('Economics', 3, 3),
('Geography', 3, 4),
('Political Science', 4, 5);


-- Notes:
-- I have given course id as auto increment immediately after student id. hence it started from
11.

INSERT INTO Enrollments (student_id, course_id, enrollment_date)
VALUES
(1, 11, '2024-01-10'),
(2, 12, '2024-01-10'),
(3, 13, '2024-01-11'),
(4, 14, '2024-01-11'),
(5, 15, '2024-01-12'),
(6, 16, '2024-01-12'),
(7, 17, '2024-01-13'),
(8, 18, '2024-01-13'),
(9, 19, '2024-01-14'),
(10, 20, '2024-01-14');



-- basic CRUD Operations

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

UPDATE Teacher
SET email = 'new_email@example.com'
WHERE teacher_id = 3;
```

```sql
DELETE FROM Enrollments
WHERE student_id = 5 AND course_id = 5;

UPDATE Courses
SET teacher_id = 5
WHERE course_id = 5;

UPDATE Payments
SET amount = 100000
WHERE payment_id = 3;




-- Output: Total payments made by student with student_id = 1.
SELECT 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 = 1;

-- Output: Count of enrolled students for each course.
SELECT c.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;

-- Output: Students who are not enrolled in any courses.
SELECT 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;

-- Output: Students and their enrolled courses.
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;

-- Output: Teachers and their assigned courses.
SELECT t.first_name, t.last_name, c.course_name
FROM Teacher t
JOIN Courses c ON t.teacher_id = c.teacher_id;

-- Output: Students enrolled in the 'Biology' course along with their enrollment date.
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 = 'Biology';

-- Output: 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.student_id IS NULL;

-- Output: Courses with no enrollments.
SELECT c.course_name
FROM Courses c
LEFT JOIN Enrollments e ON c.course_id = e.course_id
WHERE e.course_id IS NULL;

-- Output: Students enrolled in more than one course.
SELECT s.first_name, s.last_name
FROM Students s
JOIN Enrollments e1 ON s.student_id = e1.student_id
JOIN Enrollments e2 ON s.student_id = e2.student_id AND e1.enrollment_id <>
e2.enrollment_id;

-- Output: Teachers with no assigned courses.
SELECT t.first_name, t.last_name
FROM Teacher t
LEFT JOIN Courses c ON t.teacher_id = c.teacher_id
WHERE c.teacher_id IS NULL;




-- Writing Subqueries




-- Output: Average enrollment count for courses.
SELECT AVG(enrollment_count) AS average_enrollment
FROM (
   SELECT COUNT(*) AS enrollment_count
   FROM Enrollments
   GROUP BY course_id
) AS course_enrollments;

-- Output: Student ID(s) with the highest payment amount.
SELECT student_id
FROM Payments
WHERE amount = (SELECT MAX(amount) FROM Payments);
```

```sql
-- Output: Course ID(s) with the maximum number of enrollments.
SELECT course_id
FROM Enrollments
GROUP BY course_id
HAVING COUNT(*) = (SELECT MAX(enrollment_count) FROM (SELECT COUNT(*) AS
enrollment_count FROM Enrollments GROUP BY course_id) AS course_enrollments);

-- Output: Total payments made to each teacher.
SELECT teacher_id, SUM(amount) AS total_payments
FROM (
   SELECT e.course_id, p.amount
   FROM Enrollments e
   JOIN Payments p ON e.student_id = p.student_id
) AS student_payments
JOIN Courses c ON student_payments.course_id = c.course_id
GROUP BY teacher_id;

-- Output: Student ID(s) enrolled in all courses.
SELECT student_id
FROM (
   SELECT student_id, COUNT(DISTINCT course_id) AS course_count
   FROM Enrollments
   GROUP BY student_id
) AS student_courses
WHERE course_count = (SELECT COUNT(*) FROM Courses);

-- Output: Teacher(s) who are not assigned to any courses.
SELECT teacher_id, first_name, last_name
FROM Teacher
WHERE teacher_id NOT IN (SELECT DISTINCT teacher_id FROM Courses);

-- Output: Average age of students.
SELECT AVG(DATEDIFF(NOW(), date_of_birth) / 365) AS average_age
FROM Students;

-- Output: Course ID(s) with no enrollments.
SELECT course_id
FROM Courses
WHERE course_id NOT IN (SELECT DISTINCT course_id FROM Enrollments);

-- Output: Student ID(s) who made multiple payments.
SELECT student_id
FROM Payments
GROUP BY student_id
HAVING COUNT(*) > 1;
```

```sql
-- Output: Total payments made by each student.
SELECT student_id, SUM(amount) AS total_payments
FROM Payments
GROUP BY student_id;

-- Output: Count of enrolled students for each course.
SELECT c.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;
```