STUDENT INFORMATION SYSTEM (SIS)

SQL ASSIGNMENT

TASK 1:

1. Create the database named "SISDB":

```sql
CREATE DATABASE StudentInformationSystemDB;

USE StudentInformationSystemDB;
```

144 %

Messages

```
Commands completed successfully.

Completion time: 2025-04-10T18:12:24.5832729+05:30
```

2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema.

```sql
--Student table

CREATE TABLE Students (
    student_id INT PRIMARY KEY IDENTITY(1,1),
    first_name NVARCHAR(50),
    last_name NVARCHAR(50),
    date_of_birth DATE,
    email NVARCHAR(100),
    phone_number NVARCHAR(20)
);

--Teachers table

CREATE TABLE Teachers (
    teacher_id INT PRIMARY KEY IDENTITY(1,1),
    first_name NVARCHAR(50),
    last_name NVARCHAR(50),
    email NVARCHAR(100)
);
```

```sql
--Course table

CREATE TABLE Courses (
    course_id INT PRIMARY KEY IDENTITY(1,1),
    course_name NVARCHAR(100),
    credits INT,
    teacher_id INT,
    FOREIGN KEY (teacher_id) REFERENCES Teachers(teacher_id)
);


--Enrollments table

CREATE TABLE Enrollments (
    enrollment_id INT PRIMARY KEY IDENTITY(1,1),
    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)
);
--payments table

CREATE TABLE Payments (
    payment_id INT PRIMARY KEY IDENTITY(1,1),
    student_id INT,
    amount DECIMAL(10, 2),
    payment_date DATE,
    FOREIGN KEY (student_id) REFERENCES Students(student_id)
);
```

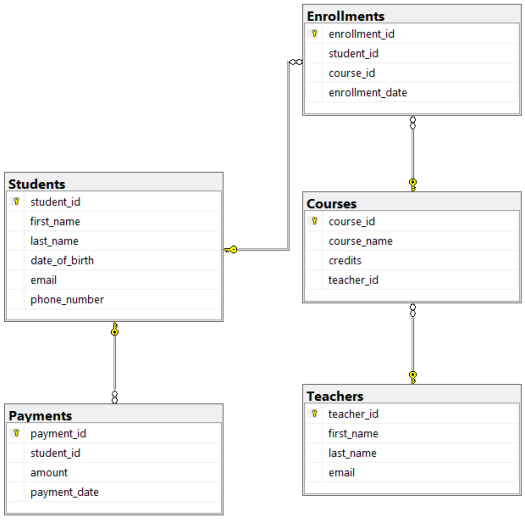
144 %

Messages

```
Commands completed successfully.

Completion time: 2025-04-10T18:26:34.6341463+05:30
```

3. Create an ERD (Entity Relationship Diagram) for the database.



4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.

The Question is already done in Qno:2

5. Insert at least 10 sample records into each of the following tables.

```sql
--Students data--

INSERT INTO Students (first_name, last_name, date_of_birth, email, phone_number)
VALUES
('Rahul', 'Sharma', '2000-08-15', 'rahul.sharma@gmail.com', '9876543210'),
('Priya', 'Mehta', '2001-07-12', 'priya.mehta@yahoo.com', '8765432109'),
('Amit', 'Verma', '2002-01-21', 'amit.verma@outlook.com', '7654321098'),
('Sneha', 'Reddy', '2003-11-11', 'sneha.reddy@gmail.com', '9543210987'),
('Karan', 'Patel', '2000-05-09', 'karan.patel@gmail.com', '9988776655'),
('Neha', 'Singh', '2001-03-28', 'neha.singh@gmail.com', '9123456780'),
('Ravi', 'Kumar', '2002-09-14', 'ravi.kumar@gmail.com', '8899776655'),
('Pooja', 'Nair', '2003-12-19', 'pooja.nair@gmail.com', '9345678901'),
('Anil', 'Yadav', '2001-06-06', 'anil.yadav@gmail.com', '9001234567'),
('Divya', 'Joshi', '2000-02-23', 'divya.joshi@gmail.com', '8888888888');


--Teachers data

INSERT INTO Teachers (first_name, last_name, email)
VALUES
('Anita', 'Deshmukh', 'anita.deshmukh@college.edu.in'),
('Rajeev', 'Nair', 'rajeev.nair@college.edu.in'),
('Sonal', 'Kapoor', 'sonal.kapoor@college.edu.in'),
('Arjun', 'Iyer', 'arjun.iyer@college.edu.in'),
```

```sql
INSERT INTO Courses (course_name, credits, teacher_id)
VALUES
('Introduction to Programming', 4, 1),
('Mathematics 101', 3, 2),
('Computer Science 101', 4, 3),
('English Literature', 3, 4),
('Physics Fundamentals', 4, 5),
('Chemistry Basics', 4, 6),
('Indian History', 3, 7),
('Environmental Studies', 2, 8),
('Web Technologies', 4, 9),
('Database Management Systems', 4, 10);



--enrollement data

INSERT INTO Enrollments (student_id, course_id, enrollment_date)
VALUES
(1, 1, '2024-01-10'),
(2, 2, '2024-01-11'),
(3, 3, '2024-01-12'),
(4, 4, '2024-01-13'),
```

```sql
--payment data

INSERT INTO Payments (student_id, amount, payment_date)
VALUES
(1, 5000.00, '2024-02-01'),
(2, 4500.00, '2024-02-02'),
(3, 6000.00, '2024-02-03'),
(4, 3000.00, '2024-02-04'),
(5, 4000.00, '2024-02-05'),
(6, 7000.00, '2024-02-06'),
(7, 3500.00, '2024-02-07'),
(8, 2000.00, '2024-02-08'),
(9, 5500.00, '2024-02-09'),
(10, 6500.00, '2024-02-10');
```

```
144 %
Messages

(10 rows affected)

(10 rows affected)

(10 rows affected)

(10 rows affected)

(10 rows affected)

Completion time: 2025-04-10T18:36:41.8281533+05:30
```

TASK 2:

1. **Write an SQL query to insert a new student into the "Students" table with the following details:**

- **First Name: John**
- **b. Last Name: Doe**
- **c. Date of Birth: 1995-08-15**
- **d. Email: john.doe@example.com**
- **e. Phone Number: 1234567890**

```sql
--1
INSERT INTO Students (first_name, last_name, date_of_birth, email, phone_number)
VALUES ('John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
```

```
144 %
Messages

(1 row affected)

Completion time: 2025-04-10T18:37:20.1988450+05:30
```

2. **Choose an existing student and course and insert a record into the "Enrollments" table with the enrollment date.**

```sql
--2

INSERT INTO Enrollments (student_id, course_id, enrollment_date)
VALUES (1, 1, '2025-04-09');
```

```
(1 row affected)

Completion time: 2025-04-10T18:40:52.2831193+05:30
```

3. **Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.**

```sql
UPDATE Teachers
SET email = 'anita.deshmukh@updatedcollege.edu.in'
WHERE teacher_id = 1;
```

```
(1 row affected)
Completion time: 2025-04-10T18:40:52.2831193+05:30
```

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.**

```sql
--4

DELETE FROM Enrollments
WHERE student_id = 1 AND course_id = 1;
```

144 %

Messages

```
(2 rows affected)

Completion time: 2025-04-10T18:43:37.5368065+05:30
```

5. **Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.**

```
--5

UPDATE Courses
SET teacher_id = 2
WHERE course_id = 1;
```

144 %

Messages

```
(1 row affected)

Completion time: 2025-04-10T18:45:29.7978536+05:30
```

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 = 3;

DELETE FROM Payments
WHERE student_id = 3;

DELETE FROM Students
WHERE student_id = 3;
```

144 %

Messages

```
(1 row affected)

(1 row affected)

(1 row affected)

Completion time: 2025-04-10T18:48:08.6098349+05:30
```

7. **Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.**

```sql
UPDATE Payments
SET amount = 7000.00
WHERE payment_id = 1;
```

144 %

Messages

```
(1 row affected)

Completion time: 2025-04-10T18:50:17.0690058+05:30
```

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.**

```sql
SELECT s.first_name, s.last_name, SUM(p.amount) AS total_payment
FROM Students s
JOIN Payments p ON s.student_id = p.student_id
WHERE s.student_id = 1
GROUP BY s.first_name, s.last_name;
```

144 %

Results | Messages

	first_name	last_name	total_payment
1	Rahul	Sharma	7000.00

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.**

```sql
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_name;
```

144 %

Results | Messages

	course_name	enrolled_students
1	Chemistry Basics	1
2	Computer Science 101	0
3	Database Management Systems	1
4	English Literature	1
5	Environmental Studies	1
6	Indian History	1
7	Introduction to Programming	0
8	Mathematics 101	1
9	Physics Fundamentals	1
10	Web Technologies	1

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.**

```sql
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;
```

144 %

Results | Messages

	first_name	last_name
1	Rahul	Sharma
2	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.**

```sql
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;
```

144 %

Results | Messages

	first_name	last_name	course_name
1	Priya	Mehta	Mathematics 101
2	Sneha	Reddy	English Literature
3	Karan	Patel	Physics Fundamentals
4	Neha	Singh	Chemistry Basics
5	Ravi	Kumar	Indian History
6	Pooja	Nair	Environmental Studies
7	Anil	Yadav	Web Technologies
8	Divya	Joshi	Database Management Systems

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.**

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

44 %

Results | Messages

	first_name	last_name	course_name
1	Rajeev	Nair	Introduction to Programming
2	Rajeev	Nair	Mathematics 101
3	Sonal	Kapoor	Computer Science 101
4	Arjun	Iyer	English Literature
5	Meena	Rao	Physics Fundamentals
6	Sunil	Mishra	Chemistry Basics
7	Swati	Shah	Indian History
8	Nitin	Chowdhury	Environmental Studies
9	Rekha	Saxena	Web Technologies
10	Vikram	Sen	Database Management Systems

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.**

```sql
SELECT s.first_name, s.last_name, e.enrollment_date
FROM Students s JOIN Enrollments e ON s.student_id = e.student_id WHERE e.course_id = 2;
```

144 %

Results | Messages

	first_name	last_name	enrollment_date
1	Priya	Mehta	2024-01-11

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.**

```sql
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;
```

144 %

Results | Messages

	first_name	last_name
1	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.**

```sql
SELECT c.course_name FROM Courses c LEFT JOIN Enrollments e ON c.course_id = e.course_id
WHERE e.enrollment_id IS NULL;
```

144 %

Results | Messages

	course_name
1	Introduction to Programming
2	Computer Science 101

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.**

```sql
SELECT s.first_name, s.last_name, COUNT(e.course_id) AS course_count FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id GROUP BY s.first_name, s.last_name
HAVING COUNT(e.course_id) > 0;
```

144 %

Results | Messages

	first_name	last_name	course_count
1	Divya	Joshi	1
2	Ravi	Kumar	1
3	Priya	Mehta	1
4	Pooja	Nair	1
5	Karan	Patel	1
6	Sneha	Reddy	1
7	Neha	Singh	1
8	Anil	Yadav	1

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 Teachers t LEFT JOIN Courses c ON t.teacher_id = c.teacher_id
WHERE c.course_id IS NULL;
```

144 %

Results | Messages

	first_name	last_name
1	Anita	Deshmukh

TASK 4 :

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 AVG(student_count) AS avg_enrollments
FROM (
    SELECT COUNT(*) AS student_count
    FROM Enrollments
    GROUP BY course_id
) AS sub;
```

144 %

Results | Messages

	avg_enrollments
1	1

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.first_name, s.last_name, p.amount
FROM Students s JOIN Payments p ON s.student_id = p.student_id
WHERE p.amount = (
    SELECT MAX(amount) FROM Payments
);
```

144 %

Results | Messages

	first_name	last_name	amount
1	Rahul	Sharma	7000.00
2	Neha	Singh	7000.00

3. **Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the course(s) with the maximum enrollment count.**

```sql
SELECT c.course_name, COUNT(e.enrollment_id) AS total_enrollments
FROM Courses c JOIN Enrollments e ON c.course_id = e.course_id
GROUP BY c.course_name HAVING COUNT(e.enrollment_id) = (
    SELECT MAX(enrollment_count)
    FROM (
        SELECT COUNT(*) AS enrollment_count FROM Enrollments GROUP BY course_id
    ) AS sub
);
```

144 %

Results | Messages

	course_name	total_enrollments
1	Chemistry Basics	1
2	Database Management Systems	1
3	English Literature	1
4	Environmental Studies	1
5	Indian History	1
6	Mathematics 101	1
7	Physics Fundamentals	1
8	Web Technologies	1

4. **Calculate the total payments made to courses taught by each teacher. Use subqueries to sum payments for each teacher's courses.**

```sql
SELECT t.first_name, t.last_name, SUM(p.amount) AS total_payment
FROM Teachers 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;
```

144 %

Results | Messages

	first_name	last_name	total_payment
1	Arjun	Iyer	3000.00
2	Meena	Rao	4000.00
3	Nitin	Chowdhury	2000.00
4	Rajeev	Nair	4500.00
5	Rekha	Saxena	5500.00
6	Sunil	Mishra	7000.00
7	Swati	Shah	3500.00
8	Vikram	Sen	6500.00

5. **Identify students who are enrolled in all available courses. Use subqueries to compare a student's enrollments with the total number of courses.**

90 %

Results | Messages

	student_id	first_name	last_name
1	11	John	Doe
2	12	Meera	Kashyap

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

```sql
SELECT first_name, last_name
FROM Teachers
WHERE teacher_id NOT IN (
    SELECT DISTINCT teacher_id FROM Courses WHERE teacher_id IS NOT NULL
);
```

144 %

Results | Messages

	first_name	last_name
1	Anita	Deshmukh

7. **Calculate the average age of all students. Use subqueries to calculate the age of each student based on their date of birth.**

```sql
SELECT AVG(DATEDIFF(YEAR, date_of_birth, GETDATE())) AS average_age
FROM Students;
```

144 %

Results | Messages

	average_age
1	24

8. **Identify courses with no enrollments. Use subqueries to find courses without enrollment records.**

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

144 %

Results | Messages

	course_name
1	Introduction to Programming
2	Computer Science 101

9. **Calculate the total payments made by each student for each course they are enrolled in. Use subqueries and aggregate functions to sum payments.**

```sql
SELECT s.first_name, s.last_name, c.course_name, SUM(p.amount) AS total_payment
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.first_name, s.last_name, c.course_name;
```

144 %

Results | Messages

	first_name	last_name	course_name	total_payment
1	Anil	Yadav	Web Technologies	5500.00
2	Divya	Joshi	Database Management Systems	6500.00
3	Karan	Patel	Physics Fundamentals	4000.00
4	Neha	Singh	Chemistry Basics	7000.00
5	Pooja	Nair	Environmental Studies	2000.00
6	Priya	Mehta	Mathematics 101	4500.00
7	Ravi	Kumar	Indian History	3500.00
8	Sneha	Reddy	English Literature	3000.00

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.**

```sql
SELECT s.first_name, s.last_name, COUNT(p.payment_id) AS payment_count
FROM Students s JOIN Payments p ON s.student_id = p.student_id
GROUP BY s.first_name, s.last_name
HAVING COUNT(p.payment_id) > 2;
```

144 %

Results | Messages

first_name	last_name	payment_count

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.**

```sql
SELECT s.first_name, s.last_name, SUM(p.amount) AS total_payment
FROM Students s JOIN Payments p ON s.student_id = p.student_id
GROUP BY s.first_name, s.last_name;
```

144 %

Results | Messages

	first_name	last_name	total_payment
1	Divya	Joshi	6500.00
2	Ravi	Kumar	3500.00
3	Priya	Mehta	4500.00
4	Pooja	Nair	2000.00
5	Karan	Patel	4000.00
6	Sneha	Reddy	3000.00
7	Rahul	Sharma	7000.00
8	Neha	Singh	7000.00
9	Anil	Yadav	5500.00

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.**

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

144 %

Results | Messages

	course_name	student_count
1	Chemistry Basics	1
2	Computer Science 101	0
3	Database Management Systems	1
4	English Literature	1
5	Environmental Studies	1
6	Indian History	1
7	Introduction to Programming	0
8	Mathematics 101	1
9	Physics Fundamentals	1
10	Web Technologies	1

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.**

```sql
SELECT AVG(p.amount) AS avg_payment
FROM Students s
JOIN Payments p ON s.student_id = p.student_id;
```

144 %

Results | Messages

	avg_payment
1	4777.777777