# Build A Student Records Management System

A **Student Records Management System** is a database-driven application designed to manage student information efficiently. It includes tasks like adding students, updating student details, tracking academic performance, and generating reports.

## **Create the Database:**

CREATE DATABASE StudentRecordsDB;

GO

USE StudentRecordsDB;

GO

## **Create the Tables:**

## Students Table: Stores student information

## **Code:**

-- Create the Students table with the following columns:

-- student\_id: unique identifier, auto-incremented

-- name: name of the student, cannot be null

-- email: optional email field for the student

-- phone\_number: optional contact number for the student

-- enrollment\_date: date when the student enrolled, defaults to the current date

CREATE TABLE Students (

student\_id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

name NVARCHAR(255) NOT NULL,

email NVARCHAR(255),

phone\_number NVARCHAR(15),

enrollment\_date DATE DEFAULT GETDATE()

);

GO

## Teachers Table: Stores information about the teachers.

-- Create the Teachers table with the following columns:

-- teacher\_id: unique identifier, auto-incremented

-- name: name of the teacher, cannot be null

-- email: optional email field for the teacher

-- hire\_date: date the teacher was hired, defaults to the current date

CREATE TABLE Teachers (

teacher\_id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

name NVARCHAR(255) NOT NULL,

email NVARCHAR(255),

hire\_date DATE DEFAULT GETDATE()

);

GO

## Courses Table: Stores information about the courses offered.

-- Create the Courses table with the following columns:

-- course\_id: unique identifier for each course, auto-incremented

-- course\_name: name of the course, cannot be null

-- teacher\_id: references the teacher who teaches the course, foreign key from Teachers table

CREATE TABLE Courses (

course\_id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

course\_name NVARCHAR(255) NOT NULL,

teacher\_id INT,

FOREIGN KEY (teacher\_id) REFERENCES Teachers(teacher\_id)

);

GO

## Enrollments Table: Manages student enrollments in courses.

-- Create the Enrollments table to track student enrollment in courses:

-- enrollment\_id: unique identifier, auto-incremented

-- student\_id: foreign key referencing the Students table

-- course\_id: foreign key referencing the Courses table

-- enrollment\_date: date when the student enrolled in the course, defaults to current date

CREATE TABLE Enrollments (

enrollment\_id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

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)

);

GO

## Grades Table: Stores the grades students receive in courses.

-- Create the Grades table to store the grades students receive in courses:

-- grade\_id: unique identifier, auto-incremented

-- student\_id: foreign key referencing the Students table

-- course\_id: foreign key referencing the Courses table

-- grade: letter grade assigned to the student in the course

CREATE TABLE Grades (

grade\_id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

student\_id INT,

course\_id INT,

grade CHAR(2),

FOREIGN KEY (student\_id) REFERENCES Students(student\_id),

FOREIGN KEY (course\_id) REFERENCES Courses(course\_id)

);

GO

## **Inserting Data:**

* **Insert student records into the Students table**

INSERT INTO Students (name, email, phone\_number) VALUES

('Liam Gentry', 'lima@example.com', '1234567890'),

('Lyric Glass', 'lyric@example.com', '0987654321');

GO

* **Insert teacher records into the Teachers table**

INSERT INTO Teachers (name, email) VALUES

('Aiden Nichols', 'alice@example.com'),

('Beau Booth', 'beau@example.com');

GO

* **Insert course records into the Courses table, associating each course with a teacher**

INSERT INTO Courses (course\_name, teacher\_id) VALUES

('Mathematics', 1),

('Physics', 2);

GO

* **Insert enrollment records for students into courses**

INSERT INTO Enrollments (student\_id, course\_id) VALUES

(1, 1), -- Student 1 enrolls in Course 1

(2, 2); -- Student 2 enrolls in Course 2

GO

* **Insert grade records for students in their respective courses**

INSERT INTO Grades (student\_id, course\_id, grade) VALUES

(1, 1, 'A'), -- Student 1 gets grade 'A' in Course 1

(2, 2, 'B'); -- Student 2 gets grade 'B' in Course 2

GO

## **Basic Functionalities:**

* Add new students, courses, and grades.
* Track which students are enrolled in which courses.
* Generate reports on student performance.

## **Writing Queries for Functionality:**

## **Query-1: Add a New Student**

* **Insert a new student record into the Students table with name, email, and phone number**

INSERT INTO Students (name, email, phone\_number) VALUES

('Julio Stokes', 'julio@example.com', '0987654321');

GO

**Explanation:** This SQL statement inserts a new record into the Students table. The student's name is Julio Stokes, with the email julio@example.com and the phone number 0987654321. The INSERT INTO statement is used to add these values to the respective columns in the table.



## **Query-2: Enroll a Student in a Course**

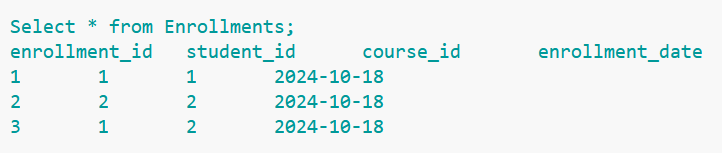
* **Insert a new enrollment record into the Enrollments table**

INSERT INTO Enrollments (student\_id, course\_id) VALUES

(1, 2);

GO

**Explanation: This SQL statement enrolls the student with student\_id = 1 into the course with course\_id = 2. The INSERT INTO statement is used to add a new record to the Enrollments table, linking the student and the course by their respective IDs.**

****

## **Query-3: Assign a Grade to a Student**

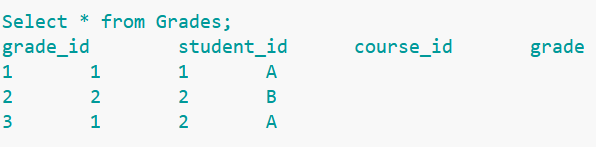
* **Insert a new grade record into the Grades table**

INSERT INTO Grades (student\_id, course\_id, grade) VALUES

(1, 2, 'A');

GO

**Explanation: This SQL statement assigns a grade of 'A' to the student with student\_id = 1 for the course with course\_id = 2. The INSERT INTO statement adds a new record to the Grades table, linking the student's ID, the course ID, and the assigned grade.**



## **Query-4: List All Students Enrolled in a Course**

* **Select student names and course name for students enrolled in course 1**

SELECT s.name, c.course\_name

FROM Enrollments e

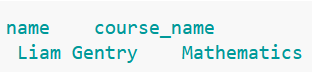
JOIN Students s ON e.student\_id = s.student\_id

JOIN Courses c ON e.course\_id = c.course\_id

WHERE c.course\_id = 1;

GO

**Explanation: This** [**SQL query**](https://www.w3resource.com/projects/sql/sql-projects-on-student-records-management-system.php) **retrieves a list of all students who are enrolled in course 1 along with the course name. It selects the student's name from the Students table and the course name from the Courses table. The Enrollments table acts as a link between the Students and Courses tables, and the query filters for records where the course\_id is 1, meaning it only shows students enrolled in that specific course.**

****

## **Query-5: View Student's Grades**

* **Select student name, course name, and grade for student 1**

SELECT s.name, c.course\_name, g.grade

FROM Grades g

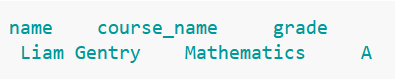
JOIN Students s ON g.student\_id = s.student\_id

JOIN Courses c ON g.course\_id = c.course\_id

WHERE s.student\_id = 1;

GO

**Explanation: This SQL query retrieves the grades of student 1 across all the courses they are enrolled in. It selects the student's name, the course name, and the corresponding grade from the Grades table. The Students and Courses tables are joined to provide the student's details and course names, and the query filters for records where the student\_id is 1, showing only the grades for that specific student.**

****

## **Query-6: Update a Student’s Contact Information**

* **Update the phone number of student 1:** Update the phone number of the student with student\_id 1

UPDATE Students

SET phone\_number = '9876543210'

WHERE student\_id = 1;

GO

**Explanation: This statement updates the phone\_number of the student with student\_id = 1 to '9876543210'. The UPDATE statement modifies the existing record in the Students table where the student\_id matches the specified value. It ensures that only the phone number for student 1 is changed.**

## **Query-7: Remove a Student from a Course**

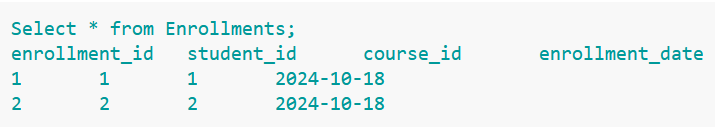
* **Delete the enrollment record for student 1 in course 2**

DELETE FROM Enrollments

WHERE student\_id = 1 AND course\_id = 2;

GO

**Explanation: This SQL statement deletes the enrollment record for the student with student\_id = 1 in the course with course\_id = 2. The DELETE FROM statement removes the matching record from the Enrollments table where both the student and course IDs match the specified values, effectively unenrolling the student from that course.**

****

## **Query-8: View All Courses Taught by a Specific Teacher**

* **Select the names of courses taught by the teacher with teacher\_id 2**

SELECT course\_name

FROM Courses

WHERE teacher\_id = 2;

GO

**Explanation: This SQL query retrieves the names of all courses taught by the teacher with teacher\_id = 2. The SELECT statement is used to get the course\_name from the Courses table, and the query is filtered by teacher\_id, so it only returns courses that are assigned to the specific teacher**

****

## **Query-9: Count the Number of Students in a Course**

* **Count the total number of students enrolled in course 1**

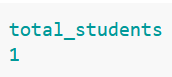
SELECT COUNT(\*) AS total\_students

FROM Enrollments

WHERE course\_id = 1;

GO

**Explanation:** **This SQL query counts the total number of students enrolled in course 1. The COUNT(\*) function returns the total number of records in the Enrollments table that match the condition course\_id = 1. The result is labeled as total\_students to show how many students are currently enrolled in that specific course.**

****

## **Query-10: List Students Who Have Not Yet Been Assigned a Grade**

* **Select the names of students who do not have a grade for course 2**

SELECT s.name

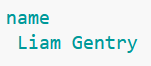
FROM Students s

LEFT JOIN Grades g ON s.student\_id = g.student\_id AND g.course\_id = 2

WHERE g.grade IS NULL;

GO

**Explanation:** **This SQL query retrieves the names of students who are not enrolled in course 2 or who have not received a grade for that course. It uses a LEFT JOIN to combine the Students table with the Grades table based on student\_id and filters for records where the grade is NULL. This means it lists students who either have no corresponding entry in the Grades table for course 2, indicating they haven't received a grade or are not enrolled in the course at all.**

****

## **Query-11: Calculate the Average Grade for a Course**

* **Find the average grade for course 1 (assuming grades are stored as characters, e.g., 'A' = 4, 'B' = 3, etc.).**

SELECT AVG(

CASE

WHEN grade = 'A' THEN 4

WHEN grade = 'B' THEN 3

WHEN grade = 'C' THEN 2

WHEN grade = 'D' THEN 1

ELSE 0

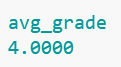
END) AS avg\_grade

FROM Grades

WHERE course\_id = 1;

GO

**Explanation:** **This** [**SQL query**](https://www.w3resource.com/projects/sql/sql-projects-on-student-records-management-system.php) **calculates the average grade point for all students enrolled in course 1. It uses a CASE statement to convert letter grades into numerical values (A = 4, B = 3, C = 2, D = 1, and any other grade is considered as 0). The AVG() function then computes the average of these numerical values. The result is labeled as avg\_grade and represents the overall performance of students in the specified course based on their grades.**



## **Query-12: Find the Highest Grade Assigned in a Course**

* **Retrieve the highest grade for students enrolled in course 2**

SELECT MAX(grade) AS highest\_grade

FROM Grades

WHERE course\_id = 2;

GO

**Explanation:** **This SQL query retrieves the highest grade achieved by students in course 2. It uses the MAX() function to find the maximum value in the grade column from the Grades table, specifically filtering for records where course\_id = 2. The result is labeled as highest\_grade, indicating the top grade received in that particular course.**



## **Query-13: Assign a Teacher to a New Course**

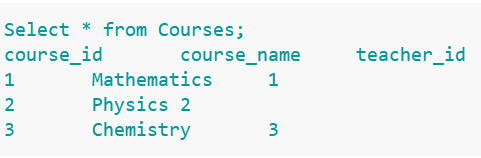
* **Insert a new course record into the Courses table**

INSERT INTO Courses (course\_name, teacher\_id)

VALUES ('Chemistry', 3);

GO

**Explanation:** **This**[**SQL**](https://www.w3resource.com/projects/sql/sql-projects-on-student-records-management-system.php) **code inserts a new record into the Courses table. It specifies that a course named "Chemistry" is being added, and it associates this course with a teacher whose ID is 3. The INSERT INTO statement is used to add new rows to a table, and the VALUES clause provides the data to be inserted into the respective columns of the table.**



## **Query-14: Find the Total Number of Courses Each Student is Enrolled In**

* **Retrieve student names along with the count of courses they are enrolled in**

SELECT s.name, COUNT(e.course\_id) AS total\_courses

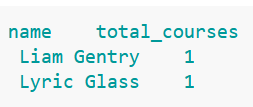
FROM Enrollments e

JOIN Students s ON e.student\_id = s.student\_id

GROUP BY s.name;

GO

**Explanation:** **This SQL query retrieves the names of students along with the total number of courses they are enrolled in. It joins the Enrollments table with the Students table based on the student\_id to associate each enrollment with the corresponding student. The COUNT(e.course\_id) function counts the number of courses for each student, and the results are grouped by the student's name using the GROUP BY clause. This allows for a summary of course enrollments per student.**



## **Query-15: List All Courses a Student is Enrolled In**

* **Retrieve the names of courses that student with ID 1 is enrolled in**

SELECT c.course\_name

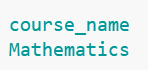
FROM Enrollments e

JOIN Courses c ON e.course\_id = c.course\_id

WHERE e.student\_id = 1;

GO

**Explanation:** **This SQL query retrieves the names of courses in which a specific student, identified by student\_id = 1, is enrolled. It does so by joining the Enrollments table with the Courses table on the course\_id. The join operation connects each enrollment record with its corresponding course record, allowing the query to select the course names associated with the specified student. The WHERE clause filters the results to include only the courses for the specified student.**



## **Query-16: Delete a Course and All Related Data**

* **Delete the course with course\_id = 2 from the Courses table**

DELETE FROM Courses WHERE course\_id = 2;

GO

* **Delete all enrollments associated with the course\_id = 2 from the Enrollments table**

DELETE FROM Enrollments WHERE course\_id = 2;

GO

* **Delete all grades associated with the course\_id = 2 from the Grades table**

DELETE FROM Grades WHERE course\_id = 2;

GO

**Explanation: The provided SQL code consists of three statements that delete data related to a specific course identified by course\_id = 2. The first statement removes the course from the Courses table. The second statement deletes any enrollment records associated with that course from the Enrollments table. Finally, the third statement eliminates any grades recorded for that course from the Grades table. This cascading deletion ensures that all references to the course are removed from the** [**database**](https://www.w3resource.com/projects/sql/sql-projects-on-student-records-management-system.php)**, preventing orphaned records in the related table**

