

# Student Management System

Submitted by: Anirudh Suryawanshi

anirudhsuryawanshi759@gmail.com

-----Task 1-----

use sisdb;

describe students;

INSERT INTO students(first\_name, last\_name, date\_of\_birth, email, phone\_number)

VALUES

('Rahul', 'Patel', '1998-05-15', 'rahul.patel@example.com', '+91234567890'),  
('Priya', 'Sharma', '2000-02-28', 'priya.sharma@example.com', '+91987654321'),  
('Aarav', 'Gupta', '1999-09-10', 'aarav.gupta@example.com', '+91122334455'),  
('Neha', 'Verma', '1997-11-20', 'neha.verma@example.com', '+91555666777'),  
('Ananya', 'Singh', '2001-04-03', 'ananya.singh@example.com', '+91444333222'),  
('Vishal', 'Yadav', '1996-08-07', 'vishal.yadav@example.com', '+91777888999'),  
('Swati', 'Kumar', '1998-12-12', 'swati.kumar@example.com', '+91888999000'),  
('Raj', 'Joshi', '2000-06-25', 'raj.joshi@example.com', '+91666777888'),  
('Pooja', 'Mehta', '1999-03-18', 'pooja.mehta@example.com', '+91999888777'),  
('Kiran', 'Chopra', '1997-10-30', 'kiran.chopra@example.com', '+91133444555');

select \* from students;

INSERT INTO teachers(first\_name, last\_name, email)

VALUES ('Rajesh', 'Kumar', 'rajesh.kumar@example.com'),  
('Aarti', 'Sharma', 'aarti.sharma@example.com'),  
('Nikhil', 'Gupta', 'nikhil.gupta@example.com'),  
('Preeti', 'Singh', 'preeti.singh@example.com'),  
('Manish', 'Verma', 'manish.verma@example.com'),  
('Divya', 'Joshi', 'divya.joshi@example.com');

```
('Nisha', 'Patel', 'nisha.patel@example.com'),  
( 'Akash', 'Shah', 'akash.shah@example.com'),  
( 'Sakshi', 'Malhotra', 'sakshi.malhotra@example.com'),  
( 'Karan', 'Chopra', 'karan.chopra@example.com');
```

```
INSERT INTO courses (course_name, credits, teacher_id) VALUES  
( 'Hindi', '4', 5),  
( 'Physics', '4', 2),  
( 'Chemistry', '3', 3),  
( 'Biology', '3', 4),  
( 'Computer Science', '4', 5),  
( 'History', '3', 6),  
( 'Literature', '3', 7),  
( 'Economics', '3', 8),  
( 'Psychology', '3', 9),  
( 'Sociology', '3', 10);
```

```
INSERT INTO enrollments (student_id, course_id, enrollment_date) VALUES  
(1, 1, '2024-04-10'),  
(2, 3, '2024-04-11'),  
(3, 5, '2024-04-12'),  
(4, 7, '2024-04-13'),  
(5, 9, '2024-04-14'),  
(6, 2, '2024-04-15'),  
(7, 4, '2024-04-16'),  
(8, 6, '2024-04-17'),  
(9, 8, '2024-04-18'),  
(10, 10, '2024-04-19');
```

```
INSERT INTO payments (student_id, amount, payment_date) VALUES  
(1, 500, '2024-04-09'),
```

```
(2, 700, '2024-04-10'),  
(3, 1000, '2024-04-11'),  
(4, 800, '2024-04-12'),  
(5, 1200, '2024-04-13'),  
(6, 900, '2024-04-14'),  
(7, 1500, '2024-04-15'),  
(8, 1100, '2024-04-16'),  
(9, 1300, '2024-04-17'),  
(10, 1800, '2024-04-18');
```

```
select * from students;  
select * from courses;  
select * from teachers;  
select * from enrollments;  
select * from payments;
```

-- -----Task 2-----

/\* Q1 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@gmail.com','1234567890');
```

-- Q2 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 (11,2,'2024-04-09');
```

-- Q3 Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.

```
update teachers
```

```
set email='rajeshKA@gmail.com'
```

```
where teacher_id=1;
```

-- Q4 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_date='2024-04-09';
```

```
select *
```

```
from enrollments
```

```
where student_id =7 AND course_id=4;
```

-- Q5 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=3
```

```
where course_name ='mathematics';
```

-- Q6 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 student
```

```
where student_id=2;
```

```
delete from enrollments
```

```
where student_id=2;
```

-- Q7 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=600
where payment_id=1;
```

-- -----Task 3-----

-- Q1 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.*, sum(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.student_id;
```

/\*

```
+-----+-----+-----+-----+-----+-----+
| student_id | first_name | last_name | date_of_birth | email | phone_number |
total_payment |
+-----+-----+-----+-----+-----+-----+
| 1 | Rahul | Patel | 1998-05-15 | rahul.patel@example.com | +91234567890 |
1800 |
+-----+-----+-----+-----+-----+-----+*/
```

-- Q2 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(s.student_id)
from students s
JOIN enrollments e ON s.student_id=e.student_id
JOIN courses c ON c.course_id=e.course_id
group by c.course_id;
```

/\*

```
+-----+-----+
| course_name | count(s.student_id) |
+-----+-----+
```

Mathematics		2	
Physics		1	
Chemistry		1	
Biology		1	
Computer Science		1	
History		1	
Literature		1	
Economics		1	
Psychology		1	
Sociology		1	

```

+-----+-----+*/

```

-- Q3 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 *
```

```
from students s
```

```
LEFT JOIN enrollments e ON s.student_id=e.student_id
```

```
where enrollment_id is null;
```

```
/*
```

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+

```

student_id	first_name	last_name	date_of_birth	email	phone_number
enrollment_id	enrollment_date	course_id	student_id		

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+

```

11	John	Doe	1995-08-15	john.doe@gmail.com	1234567890	NULL
NULL	NULL	NULL				

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+*/

```

-- Q4 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;

```

```

/*

```

```

+-----+-----+-----+
| first_name | last_name | course_name |
+-----+-----+-----+
| Rahul      | Patel     | Mathematics  |
| Tom        | Holland   | Mathematics  |
| Vishal     | Yadav     | Physics      |
| Priya      | Sharma    | Chemistry    |
| Swati      | Kumar     | Biology      |
| Aarav      | Gupta     | Computer Science |
| Raj        | Joshi     | History      |
| Neha       | Verma     | Literature    |
| Pooja      | Mehta     | Economics    |
| Ananya     | Singh     | Psychology    |
| Kiran      | Chopra    | Sociology     |
+-----+-----+-----+*/

```

-- Q5 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 courses c JOIN teachers t ON c.teacher_id=t.teacher_id;

```

```

/*

```

```

+-----+-----+-----+
| first_name | last_name | course_name |
+-----+-----+-----+
| Aarti      | Sharma    | Physics      |
| Nikhil     | Gupta     | Mathematics  |

```

	Nikhil		Gupta		Chemistry	
	Preeti		Singh		Biology	
	Manish		Verma		Computer Science	
	Manish		Verma		Hindi	
	Divya		Joshi		History	
	Nisha		Patel		Literature	
	Akash		Shah		Economics	
	Sakshi		Malhotra		Psychology	
	Karan		Chopra		Sociology	
+-----+-----+-----+*/						

-- Q6 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.*,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='sociology';

/*

```

+-----+-----+-----+-----+-----+-----+-----+						
	student_id		first_name		last_name	
	enrollment_date		date_of_birth		email	
					phone_number	
+-----+-----+-----+-----+-----+-----+-----+						
	10		Kiran		Chopra	
	2024-04-19				1997-10-30	
					kiran.chopra@example.com	
					+91133444555	
+-----+-----+-----+-----+-----+-----+*/						

-- Q7 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.

```

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;

```



```

/*
+-----+-----+
| first_name | last_name |
+-----+-----+
| John      | Doe      |
| Tom       | Holland   |
+-----+-----+*/

```

-- Q8 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 e.course_id=c.course_id
where e.enrollment_id is null;

```

```

/*
+-----+
| course_name |
+-----+
| Hindi       |
+-----+*/

```

-- Q9 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 s.first_name,s.last_name, count(e.enrollment_id) as no_of_courses_enrolled
from students s JOIN enrollments e ON s.student_id=e.student_id
group by s.student_id
having no_of_courses_enrolled>1;

```

```

/*
first_name, last_name, no_of_courses_enrolled ==> no record available*/

```

-- Q10 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 *
from teachers t LEFT JOIN courses c ON c.teacher_id=t.teacher_id
where c.course_id is null;
/*
+-----+-----+-----+-----+-----+-----+-----+
| teacher_id | first_name | last_name | email          | course_id | course_name | credits |
teacher_id |
+-----+-----+-----+-----+-----+-----+-----+
|      1 | Rajesh    | Kumar    | rajeshKA@gmail.com | NULL | NULL      | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+*/

```

#### -- -----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 c.course_name, avg(s.student_id) as average
from students s
JOIN enrollments e ON s.student_id=e.student_id
JOIN courses c ON c.course_id=e.course_id
group by c.course_id;

```

```

/*
+-----+-----+
| course_name | average |
+-----+-----+
| Mathematics | 6.5000 |
| Physics     | 6.0000 |
| Chemistry   | 2.0000 |
| Biology     | 7.0000 |
| Computer Science | 3.0000 |
| History     | 8.0000 |
| Literature   | 4.0000 |
| Economics   | 9.0000 |
| Psychology   | 5.0000 |

```

```
| Sociology      | 10.0000 |
```

```
+-----+-----+*/
```

-- 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.*,p.amount
```

```
from students s JOIN payments p ON s.student_id=p.student_id
```

```
order by p.amount DESC
```

```
limit 1;
```

```
/*
```

```
+-----+-----+
```

```
| course_name    | average |
```

```
+-----+-----+
```

```
| Mathematics    | 6.5000 |
```

```
| Physics        | 6.0000 |
```

```
| Chemistry      | 2.0000 |
```

```
| Biology        | 7.0000 |
```

```
| Computer Science | 3.0000 |
```

```
| History        | 8.0000 |
```

```
| Literature      | 4.0000 |
```

```
| Economics      | 9.0000 |
```

```
| Psychology      | 5.0000 |
```

```
| Sociology      | 10.0000 |
```

```
+-----+-----+*/
```

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

```
select c.course_name, count(e.enrollment_id) as number_of_courses
```

```
from courses c JOIN enrollments e ON c.course_id=e.course_id
```

```
group by c.course_id
```

```
order by number_of_courses DESC
```

```
limit 1;
```

```

/*
+-----+-----+
| course_name | number_of_courses |
+-----+-----+
| Mathematics |          2 |
+-----+-----+*/

```

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

```

select t.*, sum(p.amount) as total_payment
from students s
JOIN enrollments e ON s.student_id=e.student_id
JOIN courses c ON c.course_id=e.course_id
JOIN payments p ON s.student_id=p.student_id
JOIN teachers t ON c.teacher_id=t.teacher_id
group by t.teacher_id;

```

```

/*
+-----+-----+-----+-----+-----+
| teacher_id | first_name | last_name | email | total_payment |
+-----+-----+-----+-----+-----+
| 2 | Aarti | Sharma | aarti.sharma@example.com | 900 |
| 3 | Nikhil | Gupta | nikhil.gupta@example.com | 2500 |
| 4 | Preeti | Singh | preeti.singh@example.com | 1500 |
| 5 | Manish | Verma | manish.verma@example.com | 1000 |
| 6 | Divya | Joshi | divya.joshi@example.com | 1100 |
| 7 | Nisha | Patel | nisha.patel@example.com | 800 |
| 8 | Akash | Shah | akash.shah@example.com | 1300 |
| 9 | Sakshi | Malhotra | sakshi.malhotra@example.com | 1200 |
| 10 | Karan | Chopra | karan.chopra@example.com | 1800 |
+-----+-----+-----+-----+-----+*/

```

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

```
select s.*
from students s
JOIN enrollments e ON s.student_id=e.student_id
JOIN courses c ON c.course_id=e.course_id
where s.student_id = ALL(select course_id from courses);

/*student_id, first_name, last_name, date_of_birth, email, phone_number ==> no record
available*/
```

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

```
select *
from teachers
where teacher_id NOT IN(select t.teacher_id
                        from courses c JOIN teachers t ON
c.teacher_id=t.teacher_id);

/*
+-----+-----+-----+-----+
| teacher_id | first_name | last_name | email          |
+-----+-----+-----+-----+
|      1 | Rajesh    | Kumar    | rajeshKA@gmail.com |
+-----+-----+-----+-----+*/
```

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

```
SELECT avg(TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE())) AS average_age
FROM students;

/*
+-----+-----+-----+-----+
| teacher_id | first_name | last_name | email          |
+-----+-----+-----+-----+
|      1 | Rajesh    | Kumar    | rajeshKA@gmail.com |
+-----+-----+-----+-----+*/
```

```
+-----+-----+-----+-----+*/
```

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

```
select *
from courses
where course_id NOT IN (select c.course_id
                        from courses c JOIN enrollments e ON
                        c.course_id=e.course_id);
```

```
/*
```

```
+-----+-----+-----+-----+
| course_id | course_name | credits | teacher_id |
+-----+-----+-----+-----+
|    11 | Hindi    |    4 |    5 |
+-----+-----+-----+-----+*/
```

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

```
select s.*, sum(p.amount) as total_payment
from students s
JOIN enrollments e ON s.student_id=e.student_id
JOIN courses c ON c.course_id=e.course_id
JOIN payments p ON s.student_id=p.student_id
JOIN teachers t ON c.teacher_id=t.teacher_id
group by s.student_id;
```

```
/*
```

```
+-----+-----+-----+-----+-----+-----+-----+
| student_id | first_name | last_name | date_of_birth | email | phone_number | total_payment |
+-----+-----+-----+-----+-----+-----+-----+
|    1 | Rahul    | Patel    | 1998-05-15 | rahul.patel@example.com | +91234567890 | 1800 |
```

2	Priya	Sharma	2000-02-28	priya.sharma@example.com	+91987654321	700
3	Aarav	Gupta	1999-09-10	aarav.gupta@example.com	+91122334455	1000
4	Neha	Verma	1997-11-20	neha.verma@example.com	+91555666777	800
5	Ananya	Singh	2001-04-03	ananya.singh@example.com	+91444333222	1200
6	Vishal	Yadav	1996-08-07	vishal.yadav@example.com	+91777888999	900
7	Swati	Kumar	1998-12-12	swati.kumar@example.com	+91888999000	1500
8	Raj	Joshi	2000-06-25	raj.joshi@example.com	+91666777888	1100
9	Pooja	Mehta	1999-03-18	pooja.mehta@example.com	+91999888777	1300
10	Kiran	Chopra	1997-10-30	kiran.chopra@example.com	+91133444555	1800

```

+-----+-----+-----+-----+-----+-----+-----+*/

```

-- 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.

```

select s.*
from students s JOIN payments p ON s.student_id=p.student_id
group by s.student_id
having count(p.payment_id)>1;

/*

```

student_id	first_name	last_name	date_of_birth	email	phone_number
1	Rahul	Patel	1998-05-15	rahul.patel@example.com	+91234567890

```

+-----+-----+-----+-----+-----+-----+*/

```

-- 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.

```
select s.*, sum(p.amount) as 'total payment'
```

```
from students s JOIN payments p ON s.student_id=p.student_id
```

```
group by s.student_id;
```

```
/*
```

```
+-----+-----+-----+-----+-----+-----+-----+
| student_id | first_name | last_name | date_of_birth | email | phone_number | total payment |
```

```
+-----+-----+-----+-----+-----+-----+-----+
```

```
| 1 | Rahul | Patel | 1998-05-15 | rahul.patel@example.com | +91234567890 | 1800 |
```

```
| 2 | Priya | Sharma | 2000-02-28 | priya.sharma@example.com | +91987654321 | 700 |
```

```
| 3 | Aarav | Gupta | 1999-09-10 | aarav.gupta@example.com | +91122334455 | 1000 |
```

```
| 4 | Neha | Verma | 1997-11-20 | neha.verma@example.com | +91555666777 | 800 |
```

```
| 5 | Ananya | Singh | 2001-04-03 | ananya.singh@example.com | +91444333222 | 1200 |
```

```
| 6 | Vishal | Yadav | 1996-08-07 | vishal.yadav@example.com | +91777888999 | 900 |
```

```
| 7 | Swati | Kumar | 1998-12-12 | swati.kumar@example.com | +91888999000 | 1500 |
```

```
| 8 | Raj | Joshi | 2000-06-25 | raj.joshi@example.com | +91666777888 | 1100 |
```

```
| 9 | Pooja | Mehta | 1999-03-18 | pooja.mehta@example.com | +91999888777 | 1300 |
```

```
| 10 | Kiran | Chopra | 1997-10-30 | kiran.chopra@example.com | +91133444555 | 1800 |
```

```
+-----+-----+-----+-----+-----+-----+-----+*/
```

-- 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.

```
select c.course_name, count(e.enrollment_id)
```

```
from courses c JOIN enrollments e ON c.course_id=e.course_id
```

```
group by c.course_id;
```



```

/*
+-----+-----+
| course_name | count(e.enrollment_id) |
+-----+-----+
| Mathematics |          2 |
| Physics     |          1 |
| Chemistry   |          1 |
| Biology     |          1 |
| Computer Science |        1 |
| History     |          1 |
| Literature   |          1 |
| Economics   |          1 |
| Psychology  |          1 |
| Sociology   |          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.

```

select s.*, avg(p.amount) as average_payment_amount
from students s JOIN payments p ON s.student_id=p.student_id
group by s.student_id;

```

```

/*
+-----+-----+-----+-----+-----+-----+-----+
| student_id | first_name | last_name | date_of_birth | email | phone_number |
| average_payment_amount |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Rahul | Patel | 1998-05-15 | rahul.patel@example.com | +91234567890 |
900 |
| 2 | Priya | Sharma | 2000-02-28 | priya.sharma@example.com | +91987654321 |
700 |
| 3 | Aarav | Gupta | 1999-09-10 | aarav.gupta@example.com | +91122334455 |
1000 |

```

| 4 | Neha | Verma | 1997-11-20 | neha.verma@example.com | +91555666777 |  
800 |

| 5 | Ananya | Singh | 2001-04-03 | ananya.singh@example.com | +91444333222 |  
1200 |

| 6 | Vishal | Yadav | 1996-08-07 | vishal.yadav@example.com | +91777888999 |  
900 |

| 7 | Swati | Kumar | 1998-12-12 | swati.kumar@example.com | +91888999000 |  
1500 |

| 8 | Raj | Joshi | 2000-06-25 | raj.joshi@example.com | +91666777888 |  
1100 |

| 9 | Pooja | Mehta | 1999-03-18 | pooja.mehta@example.com | +91999888777 |  
1300 |

| 10 | Kiran | Chopra | 1997-10-30 | kiran.chopra@example.com | +91133444555 |  
1800 |

+-----+-----+-----+-----+-----+-----+-----+\*/