SQL ASSIGNMENT

STUDENT MANAGEMENT SYSTEM

Task 1. Database Design:

create database sisdb;

use sisdb;

1 . Students Table

create table students (

student_id int primary key auto_increment,

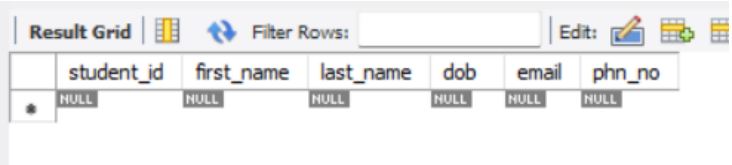
first_name varchar(50) not null,

last_name varchar(50) not null,

dob date not null,

email varchar(100) unique not null,

phn_no varchar(15) unique not null);



2 . Teacher Table

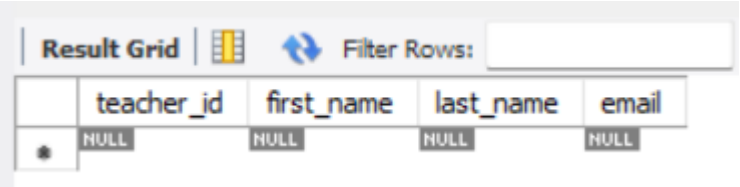
create table teacher (

teacher_id int primary key auto_increment,

first_name varchar(50) not null,

last_name varchar(50) not null,

email varchar(100) unique not null);



3. Courses Table

create table courses (

course_id int primary key auto_increment,

course_name varchar(100) not null,

credits int check (credits > 0),

teacher_id int,

foreign key (teacher_id) references teacher(teacher_id) on delete set null);

Result Grid | Filter Rows:

course_id	course_name	credits	teacher_id
NULL	NULL	NULL	NULL

4 . Enrollments Table

create table enrollments (

enrollment_id int primary key auto_increment,

student_id int,

course_id int,

enrollment_date date not null,

foreign key (student_id) references students(student_id) on delete cascade,

foreign key (course_id) references courses(course_id) on delete cascade);

Result Grid | Filter Rows: | Edit:

enrollment_id	student_id	course_id	enrollment_date
NULL	NULL	NULL	NULL

5. Payments Table

create table payments (

payment_id int primary key auto_increment,

student_id int,

amount decimal(10,2) not null check (amount >= 0),

payment_date date not null,

foreign key (student_id) references students(student_id) on delete cascad);

Result Grid | Filter Rows: | Edit:

payment_id	student_id	amount	payment_date
NULL	NULL	NULL	NULL

Inserting values:

Students:

INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number) VALUES (1, 'John', 'Doe', '2000-05-15', 'john.doe@example.com', '9876543210'),

(2, 'Jane', 'Smith', '2001-08-22', 'jane.smith@example.com', '9876543221'),

(3, 'Mike', 'Johnson', '2000-09-10', 'mike.johnson@example.com', '9876543232'),

(4, 'Emily', 'Davis', '1999-11-05', 'emily.davis@example.com', '9876543243'),

(5, 'Robert', 'Brown', '2001-06-30', 'robert.brown@example.com', '9876543254'),

(6, 'Laura', 'Wilson', '2002-02-18', 'laura.wilson@example.com', '9876543265'),

(7, 'David', 'Clark', '1998-07-25', 'david.clark@example.com', '9876543276'),

(8, 'Sophia', 'Lopez', '2000-12-12', 'sophia.lopez@example.com', '9876543287'),

(9, 'Daniel', 'White', '2001-03-09', 'daniel.white@example.com', '9876543298'),

(10, 'Olivia', 'Martin', '1999-10-22', 'olivia.martin@example.com', '9876543309');

student_id	first_name	last_name	dob	email	phn_no
1	John	Doe	2000-05-15	john.doe@example.com	9876543210
2	Jane	Smith	2001-08-22	jane.smith@example.com	9876543221
3	Mike	Johnson	2000-09-10	mike.johnson@example.com	9876543232
4	Emily	Davis	1999-11-05	emily.davis@example.com	9876543243
5	Robert	Brown	2001-06-30	robert.brown@example.com	9876543254
6	Laura	Wilson	2002-02-18	laura.wilson@example.com	9876543265
7	David	Clark	1998-07-25	david.clark@example.com	9876543276
8	Sophia	Lopez	2000-12-12	sophia.lopez@example.com	9876543287
9	Daniel	White	2001-03-09	daniel.white@example.com	9876543298
10	Olivia	Martin	1999-10-22	olivia.martin@example.com	9876543309
NULL	NULL	NULL	NULL	NULL	NULL

Teachers:

INSERT INTO Teacher (teacher_id, first_name, last_name, email) VALUES

(1, 'Alice', 'Johnson', 'alice.johnson@example.com'),

(2, 'Bob', 'Williams', 'bob.williams@example.com'),

(3, 'Charlie', 'Brown', 'charlie.brown@example.com'),

(4, 'Diana', 'Taylor', 'diana.taylor@example.com'),

(5, 'Edward', 'Harris', 'edward.harris@example.com'),

(6, 'Fiona', 'Clark', 'fiona.clark@example.com'),

(7, 'George', 'Lewis', 'george.lewis@example.com'),

(8, 'Hannah', 'Walker', 'hannah.walker@example.com'),

(9, 'Ian', 'Scott', 'ian.scott@example.com'),

(10, 'Julia', 'Evans', 'julia.evans@example.com');

teacher_id	first_name	last_name	email
1	Alice	Johnson	alice.johnson@example.com
2	Bob	Williams	bob.williams@example.com
3	Charlie	Brown	charlie.brown@example.com
4	Diana	Taylor	diana.taylor@example.com
5	Edward	Harris	edward.harris@example.com
6	Fiona	Clark	fiona.clark@example.com
7	George	Lewis	george.lewis@example.com
8	Hannah	Walker	hannah.walker@example.com
9	Ian	Scott	ian.scott@example.com
10	Julia	Evans	julia.evans@example.com
NULL	NULL	NULL	NULL

Courses:

INSERT INTO Courses (course_id, course_name, credits, teacher_id) VALUES

(1, 'Database Systems', 4, 1),

(2, 'Operating Systems', 3, 2),

(3, 'Data Structures', 4, 3),

(4, 'Machine Learning', 3, 4),

(5, 'Networking', 3, 5),

(6, 'Cyber Security', 4, 6),

(7, 'Software Engineering', 3, 7),

(8, 'Cloud Computing', 4, 8),

(9, 'Artificial Intelligence', 3, 9),

(10, 'Big Data Analytics', 4, 10);

Result Grid | Filter Rows: | Edit:

course_id	course_name	credits	teacher_id
1	Database Systems	4	1
2	Operating Systems	3	2
3	Data Structures	4	3
4	Machine Learning	3	4
5	Networking	3	5
6	Cyber Security	4	6
7	Software Engineering	3	7
8	Cloud Computing	4	8
9	Artificial Intelligence	3	9
10	Big Data Analytics	4	10
NULL	NULL	NULL	NULL

Enrollments:

INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date) VALUES

(1, 1, 1, '2024-01-10'),

(2, 2, 2, '2024-01-12'),

(3, 3, 3, '2024-01-14'),

(4, 4, 4, '2024-01-16'),

(5, 5, 5, '2024-01-18'),

(6, 6, 6, '2024-01-20'),

(7, 7, 7, '2024-01-22'),

(8, 8, 8, '2024-01-24'),

(9, 9, 9, '2024-01-26'),

(10, 10, 10, '2024-01-28');

enrollment_id	student_id	course_id	enrollment_date
1	1	1	2024-01-10
2	2	2	2024-01-12
3	3	3	2024-01-14
4	4	4	2024-01-16
5	5	5	2024-01-18
6	6	6	2024-01-20
7	7	7	2024-01-22
8	8	8	2024-01-24
9	9	9	2024-01-26
10	10	10	2024-01-28
NULL	NULL	NULL	NULL

Payments:

INSERT INTO Payments (payment_id, student_id, amount, payment_date) VALUES

(1, 1, 500.00, '2024-02-01'),

(2, 2, 700.00, '2024-02-03'),

(3, 3, 600.00, '2024-02-05'),

(4, 4, 800.00, '2024-02-07'),

(5, 5, 550.00, '2024-02-09'),

(6, 6, 900.00, '2024-02-11'),

(7, 7, 750.00, '2024-02-13'),

(8, 8, 650.00, '2024-02-15'),

(9, 9, 500.00, '2024-02-17'),

(10, 10, 720.00, '2024-02-19');

payment_id	student_id	amount	payment_date
1	1	500.00	2024-02-01
2	2	700.00	2024-02-03
3	3	600.00	2024-02-05
4	4	800.00	2024-02-07
5	5	550.00	2024-02-09
6	6	900.00	2024-02-11
7	7	750.00	2024-02-13
8	8	650.00	2024-02-15
9	9	500.00	2024-02-17
10	10	720.00	2024-02-19
NULL	NULL	NULL	NULL

Tasks 2: Select, Where, Between, AND, LIKE:

1. Insert a new student into the "Students" table:

INSERT INTO Students (student_id, first_name, last_name, dob, email, phn_no)

VALUES ('11','John', 'Doe', '1995-08-15', 'john.does@example.com', '1234567890');

student_id	first_name	last_name	dob	email	phn_no
1	John	Doe	2000-05-15	john.doe@example.com	9876543210
2	Jane	Smith	2001-08-22	jane.smith@example.com	9876543221
3	Mike	Johnson	2000-09-10	mike.johnson@example.com	9876543232
4	Emily	Davis	1999-11-05	emily.davis@example.com	9876543243
5	Robert	Brown	2001-06-30	robert.brown@example.com	9876543254
6	Laura	Wilson	2002-02-18	laura.wilson@example.com	9876543265
7	David	Clark	1998-07-25	david.clark@example.com	9876543276
8	Sophia	Lopez	2000-12-12	sophia.lopez@example.com	9876543287
9	Daniel	White	2001-03-09	daniel.white@example.com	9876543298
10	Olivia	Martin	1999-10-22	olivia.martin@example.com	9876543309
11	John	Doe	1995-08-15	john.does@example.com	1234567890
NULL	NULL	NULL	NULL	NULL	NULL

2. Enroll a student in a course:

INSERT INTO Enrollments (student_id, course_id, enrollment_date)

VALUES (1, 3, CURDATE());

enrollment_id	student_id	course_id	enrollment_date
1	1	1	2024-01-10
2	2	2	2024-01-12
3	3	3	2024-01-14
4	4	4	2024-01-16
5	5	5	2024-01-18
6	6	6	2024-01-20
7	7	7	2024-01-22
8	8	8	2024-01-24
9	9	9	2024-01-26
10	10	10	2024-01-28
11	1	3	2025-03-31
NULL	NULL	NULL	NULL

3. Update a teacher's email address:

update teacher set email = 'new.email@example.com' where teacher_id = 2;

teacher_id	first_name	last_name	email
1	Alice	Johnson	alice.johnson@example.com
2	Bob	Williams	new.email@example.com
3	Charlie	Brown	charlie.brown@example.com
4	Diana	Taylor	diana.taylor@example.com
5	Edward	Harris	edward.harris@example.com
6	Fiona	Clark	fiona.clark@example.com
7	George	Lewis	george.lewis@example.com
8	Hannah	Walker	hannah.walker@example.com
9	Ian	Scott	ian.scott@example.com
10	Julia	Evans	julia.evans@example.com
NULL	NULL	NULL	NULL

4. Delete a specific enrollment record:

delete from enrollments where student_id = 1 and course_id = 3;

Result Grid | Filter Rows: | Edit:

enrollment_id	student_id	course_id	enrollment_date
1	1	1	2024-01-10
2	2	2	2024-01-12
3	3	3	2024-01-14
4	4	4	2024-01-16
5	5	5	2024-01-18
6	6	6	2024-01-20
7	7	7	2024-01-22
8	8	8	2024-01-24
9	9	9	2024-01-26
10	10	10	2024-01-28
NULL	NULL	NULL	NULL

5. Assign a teacher to a course:

update courses set teacher_id = 5 where course_id = 4;

course_id	course_name	credits	teacher_id
1	Database Systems	4	1
2	Operating Systems	3	2
3	Data Structures	4	3
4	Machine Learning	3	5
5	Networking	3	5
6	Cyber Security	4	6
7	Software Engineering	3	7
8	Cloud Computing	4	8
9	Artificial Intelligence	3	9
10	Big Data Analytics	4	10
NULL	NULL	NULL	NULL

6. Delete a specific student and their enrollments:

delete from students where student_id = 1;

student_id	first_name	last_name	dob	email	phn_no
2	Jane	Smith	2001-08-22	jane.smith@example.com	9876543221
3	Mike	Johnson	2000-09-10	mike.johnson@example.com	9876543232
4	Emily	Davis	1999-11-05	emily.davis@example.com	9876543243
5	Robert	Brown	2001-06-30	robert.brown@example.com	9876543254
6	Laura	Wilson	2002-02-18	laura.wilson@example.com	9876543265
7	David	Clark	1998-07-25	david.clark@example.com	9876543276
8	Sophia	Lopez	2000-12-12	sophia.lopez@example.com	9876543287
9	Daniel	White	2001-03-09	daniel.white@example.com	9876543298
10	Olivia	Martin	1999-10-22	olivia.martin@example.com	9876543309
11	John	Doe	1995-08-15	john.does@example.com	1234567890
NULL	NULL	NULL	NULL	NULL	NULL

7. Update the payment amount for a specific payment record:

update payments set amount = 750.00 where payment_id = 3;

payment_id	student_id	amount	payment_date
2	2	700.00	2024-02-03
3	3	750.00	2024-02-05
4	4	800.00	2024-02-07
5	5	550.00	2024-02-09
6	6	900.00	2024-02-11
7	7	750.00	2024-02-13
8	8	650.00	2024-02-15
9	9	500.00	2024-02-17
10	10	720.00	2024-02-19
NULL	NULL	NULL	NULL

Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:

1. Calculate total payments made by a specific student

select s.student_id, 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 group by s.student_id, s.first_name, s.last_name;

student_id	first_name	last_name	total_payments

2. Retrieve courses along with the count of students enrolled in each course

SELECT c.course_id, 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_id, c.course_name;

course_id	course_name	student_count
1	Database Systems	0
2	Operating Systems	1
3	Data Structures	1
4	Machine Learning	1
5	Networking	1
6	Cyber Security	1
7	Software Engineering	1
8	Cloud Computing	1
9	Artificial Intelligence	1
10	Big Data Analytics	1

3. Find names of students who have not enrolled in any course

```
SELECT s.student_id, s.first_name, s.last_name
FROM Students s
LEFT JOIN Enrollments e ON s.student_id = e.student_id
WHERE e.course_id IS NULL;
```

student_id	first_name	last_name
11	John	Doe

4. Retrieve student names along with courses they are enrolled in

```
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
Jane	Smith	Operating Systems
Mike	Johnson	Data Structures
Emily	Davis	Machine Learning
Robert	Brown	Networking
Laura	Wilson	Cyber Security
David	Clark	Software Engineering
Sophia	Lopez	Cloud Computing
Daniel	White	Artificial Intelligence
Olivia	Martin	Big Data Analytics

5. 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;
```

	first_name	last_name	course_name
▶	Alice	Johnson	Database Systems
	Bob	Williams	Operating Systems
	Charlie	Brown	Data Structures
	Edward	Harris	Machine Learning
	Edward	Harris	Networking
	Fiona	Clark	Cyber Security
	George	Lewis	Software Engineering
	Hannah	Walker	Cloud Computing
	Ian	Scott	Artificial Intelligence
	Julia	Evans	Big Data Analytics

6. Retrieve students and their enrollment dates for a specific course

```
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_id = 2;
```

	first_name	last_name	enrollment_date
▶	Jane	Smith	2024-01-12

7. Find students who have not made any payments

```
SELECT s.student_id, 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;
```

	student_id	first_name	last_name
▶	11	John	Doe

8. Identify courses that have no enrollments

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

	course_id	course_name
▶	1	Database Systems

9. Identify students who are enrolled in more than one course

```
SELECT s.student_id, 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.student_id, s.first_name, s.last_name
HAVING COUNT(e.course_id) > 1;
```

	student_id	first_name	last_name	course_count

10. Find teachers who are not assigned to any courses

```
SELECT t.teacher_id, 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;
```

	teacher_id	first_name	last_name
▶	4	Diana	Taylor

Task 4. Subquery and its type:

1. Calculate the average number of students enrolled in each course

```
SELECT AVG(student_count) AS avg_students_per_course
FROM ( SELECT course_id, COUNT(student_id) AS student_count
 FROM Enrollments  GROUP BY course_id
) AS course_enrollments;
```

	avg_students_per_course
▶	1.0000

2. Identify the student(s) who made the highest payment

```
SELECT s.student_id, 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);
```

student_id	first_name	last_name	amount
6	Laura	Wilson	900.00

3. Retrieve courses with the highest number of enrollments

```
SELECT c.course_id, c.course_name, COUNT(e.student_id) AS enrollment_count
FROM Courses c JOIN Enrollments e ON c.course_id = e.course_id
GROUP BY c.course_id, c.course_name
HAVING COUNT(e.student_id) = (SELECT MAX(student_count)
FROM (SELECT course_id, COUNT(student_id) AS student_count FROM Enrollments
GROUP BY course_id) AS enroll_counts );
```

course_id	course_name	enrollment_count
2	Operating Systems	1
3	Data Structures	1
4	Machine Learning	1
5	Networking	1
6	Cyber Security	1
7	Software Engineering	1
8	Cloud Computing	1
9	Artificial Intelligence	1
10	Big Data Analytics	1

4. Calculate total payments made to courses taught by each teacher

```
SELECT t.teacher_id, t.first_name, t.last_name,
    (SELECT SUM(p.amount)
     FROM Payments p
     JOIN Enrollments e ON p.student_id = e.student_id
     JOIN Courses c ON e.course_id = c.course_id
     WHERE c.teacher_id = t.teacher_id) AS total_payments FROM Teacher t;
```

teacher_id	first_name	last_name	total_payments
1	Alice	Johnson	NULL
2	Bob	Williams	700.00
3	Charlie	Brown	750.00
4	Diana	Taylor	NULL
5	Edward	Harris	1350.00
6	Fiona	Clark	900.00
7	George	Lewis	750.00
8	Hannah	Walker	650.00
9	Ian	Scott	500.00
10	Julia	Evans	720.00

5. Identify students who are enrolled in all available courses

SELECT student_id, first_name, last_name

FROM Students WHERE (SELECT COUNT(course_id) FROM Enrollments WHERE Students.student_id = Enrollments.student_id) = (SELECT COUNT(course_id) FROM Courses);

	student_id	first_name	last_name
*	NULL	NULL	NULL

6. Retrieve teachers with no assigned courses

SELECT teacher_id, first_name, last_name

FROM Teacher WHERE teacher_id NOT IN (SELECT teacher_id FROM Courses);

	teacher_id	first_name	last_name
▶	4	Diana	Taylor
*	NULL	NULL	NULL

7. Calculate the average age of students

SELECT AVG(YEAR(CURDATE()) - YEAR(dob)) AS avg_age FROM Students;

	avg_age
▶	25.4000

8. Identify courses with no enrollments

SELECT course_id, course_name

FROM Courses

WHERE course_id NOT IN (SELECT DISTINCT course_id FROM Enrollments);

	course_id	course_name
▶	1	Database Systems
*	NULL	NULL

9. Calculate total payments made by each student for each course

SELECT s.student_id, s.first_name, s.last_name, c.course_name,

(SELECT SUM(p.amount)

FROM Payments p JOIN Enrollments e ON p.student_id = e.student_id

WHERE e.course_id = c.course_id AND e.student_id = s.student_id) AS total_payments

FROM Students s JOIN Enrollments e ON s.student_id = e.student_id

JOIN Courses c ON e.course_id = c.course_id;

student_id	first_name	last_name	course_name	total_payments
2	Jane	Smith	Operating Systems	700.00
3	Mike	Johnson	Data Structures	750.00
4	Emily	Davis	Machine Learning	800.00
5	Robert	Brown	Networking	550.00
6	Laura	Wilson	Cyber Security	900.00
7	David	Clark	Software Engineering	750.00
8	Sophia	Lopez	Cloud Computing	650.00
9	Daniel	White	Artificial Intelligence	500.00
10	Olivia	Martin	Big Data Analytics	720.00

10. Identify students who have made more than one payment

SELECT student_id, first_name, last_name

FROM Students WHERE student_id IN (

SELECT student_id FROM Payments GROUP BY student_id HAVING

COUNT(payment_id) > 1);

student_id	first_name	last_name
NULL	NULL	NULL

11. Calculate total payments made by each student

SELECT s.student_id, 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

GROUP BY s.student_id, s.first_name, s.last_name;

student_id	first_name	last_name	total_payments
2	Jane	Smith	700.00
3	Mike	Johnson	750.00
4	Emily	Davis	800.00
5	Robert	Brown	550.00
6	Laura	Wilson	900.00
7	David	Clark	750.00
8	Sophia	Lopez	650.00
9	Daniel	White	500.00
10	Olivia	Martin	720.00

12. Retrieve course names along with student enrollments

SELECT c.course_id, 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_id, c.course_name;

course_id	course_name	student_count
1	Database Systems	0
2	Operating Systems	1
3	Data Structures	1
4	Machine Learning	1
5	Networking	1
6	Cyber Security	1
7	Software Engineering	1
8	Cloud Computing	1
9	Artificial Intelligence	1
10	Big Data Analytics	1

13. Calculate the average payment amount made by students

SELECT s.student_id, s.first_name, s.last_name, AVG(p.amount) AS avg_payment

FROM Students s JOIN Payments p ON s.student_id = p.student_id

GROUP BY s.student_id, s.first_name, s.last_name;

student_id	first_name	last_name	avg_payment
2	Jane	Smith	700.000000
3	Mike	Johnson	750.000000
4	Emily	Davis	800.000000
5	Robert	Brown	550.000000
6	Laura	Wilson	900.000000
7	David	Clark	750.000000
8	Sophia	Lopez	650.000000
9	Daniel	White	500.000000
10	Olivia	Martin	720.000000