**TASK 1 - DATABASE DESIGN**

**1. Create the database named "SISDB"**

CREATE DATABASE SISDB;

**2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.**

**a. Students**

CREATE TABLE Students (

student\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

date\_of\_birth DATE,

email VARCHAR(100),

phone\_number VARCHAR(20)

);

**b. Courses**

CREATE TABLE Courses (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(100),

credits INT,

teacher\_id INT,

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

);

**c. Enrollments**

CREATE TABLE Enrollments (

enrollment\_id INT PRIMARY KEY,

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)

);

**d. Teacher**

CREATE TABLE Teacher (

teacher\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

email VARCHAR(100)

);

**e. Payments**

CREATE TABLE Payments (

payment\_id INT PRIMARY KEY,

student\_id INT,

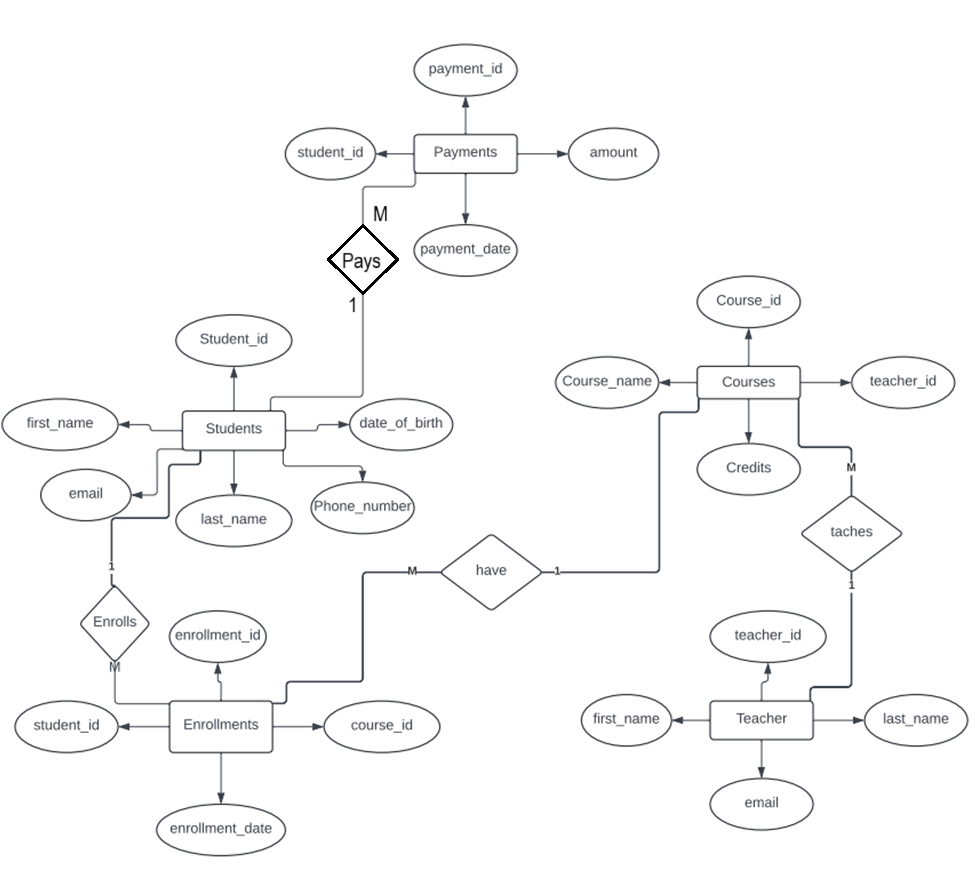
amount INT,

payment\_date DATE,

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

);

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

****

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

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

**i. Students**

INSERT INTO Students (student\_id, first\_name, last\_name, date\_of\_birth, email, phone\_number)

VALUES

(1, 'carl', 'Johnson', '2000-05-15', 'carljohn@example.com', '136-426-7280'),

(2, 'Jane', 'Smith', '1999-12-20', 'janesmith@example.com', '234-567-8901'),

(3, 'Alice', 'Johnson', '2001-03-30', 'alicejohnson@example.com', '345-678-9012'),

(4, 'Bob', 'Brown', '2002-08-10', 'bobbrown@example.com', '456-789-0123'),

(5, 'Charlie', 'Davis', '2000-11-25', 'charliedavis@example.com', '567-890-1234'),

(6, 'Emma', 'Wilson', '2001-07-15', 'emmawilson@example.com', '678-901-2345'),

(7, 'Lucas', 'Garcia', '1998-09-05', 'lucasgarcia@example.com', '789-012-3456'),

(8, 'Liam', 'Martinez', '2003-04-25', 'liammartinez@example.com', '890-123-4567'),

(9, 'Sophia', 'Miller', '2002-06-17', 'sophiamiller@example.com', '901-234-5678'),

(10, 'Olivia', 'Taylor', '2001-01-30', 'oliviataylor@example.com', '012-345-6789');

**ii. Courses**

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

VALUES

(1, 'Introduction to Computer Science', 4, 1),

(2, 'Calculus I', 3, 2),

(3, 'Physics for Scientists and Engineers', 4, 3),

(4, 'Introduction to Psychology', 3, 4),

(5, 'Creative Writing', 3, 5),

(6, 'Principles of Marketing', 3, 6),

(7, 'Microeconomics', 3, 7),

(8, 'Organic Chemistry', 4, 8),

(9, 'Modern Art History', 3, 9),

(10, 'Data Structures and Algorithms', 4, 10);

**iii. Enrollments**

INSERT INTO Enrollments (enrollment\_id, student\_id, course\_id, enrollment\_date)

VALUES

(1, 1, 1, '2023-01-10'),

(2, 2, 2, '2023-01-12'),

(3, 3, 3, '2023-01-15'),

(4, 4, 4, '2023-01-18'),

(5, 5, 5, '2023-01-20'),

(6, 6, 6, '2023-01-22'),

(7, 7, 7, '2023-01-25'),

(8, 8, 8, '2023-01-28'),

(9, 9, 9, '2023-02-01'),

(10, 10, 10, '2023-02-05');

**iv. Teacher**

INSERT INTO Teacher (teacher\_id, first\_name, last\_name, email)

VALUES

(1, 'Mark', 'Johnson', 'mark.johnson@example.com'),

(2, 'Susan', 'Williams', 'susan.williams@example.com'),

(3, 'Robert', 'Brown', 'robert.brown@example.com'),

(4, 'Jessica', 'Davis', 'jessica.davis@example.com'),

(5, 'Emily', 'Miller', 'emily.miller@example.com'),

(6, 'David', 'Wilson', 'david.wilson@example.com'),

(7, 'Michael', 'Moore', 'michael.moore@example.com'),

(8, 'Sarah', 'Taylor', 'sarah.taylor@example.com'),

(9, 'James', 'Anderson', 'james.anderson@example.com'),

(10, 'Patricia', 'Thomas', 'patricia.thomas@example.com');

**v. Payments**

INSERT INTO Payments (payment\_id, student\_id, amount, payment\_date)

VALUES

(1, 1, 500, '2023-01-05'),

(2, 2, 450, '2023-01-07'),

(3, 3, 400, '2023-01-09'),

(4, 4, 600, '2023-01-11'),

(5, 5, 550, '2023-01-13'),

(6, 6, 475, '2023-01-15'),

(7, 7, 520, '2023-01-17'),

(8, 8, 510, '2023-01-19'),

(9, 9, 480, '2023-01-21'),

(10, 10, 530, '2023-01-23');

**TASKS 2 - SELECT, WHERE, BETWEEN, AND, LIKE**

**1. 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 (11, first\_name, last\_name, date\_of\_birth, email, phone\_number)

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

**2. 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 (enrollment\_id, student\_id, course\_id, enrollment\_date)

VALUES (11, 1, 1, '2023-11-15');

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

UPDATE Teacher

SET email = 'new.email@example.com'

WHERE teacher\_id = 1;

**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.**

DELETE FROM Enrollments

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

**5. 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 = 2

WHERE course\_id = 1;

**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 = 1;

DELETE FROM Students

WHERE student\_id = 1;

**7. 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 - AGGREGATE FUNCTIONS, HAVING, ORDER BY, GROUPBY AND JOINS**

**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.**

SELECT first\_name, last\_name, SUM (Payments.amount) AS total\_payments

FROM Payments JOIN Students ON Payments.student\_id = Students.student\_id

WHERE Students.student\_id = 2

**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.**

SELECT courses.course\_id, courses.course\_name, COUNT(student\_id) AS student\_count

FROM Courses LEFT JOIN Enrollments ON Courses.course\_id = Enrollments.course\_id

GROUP BY course\_id, course\_name;

**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.**

SELECT first\_name, last\_name, date\_of\_birth, email, phone\_number

FROM Students LEFT JOIN Enrollments ON Students.student\_id = Enrollments.student\_id

WHERE Enrollments.student\_id IS NULL;

**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.**

SELECT Students.first\_name, Students.last\_name, Courses.course\_name

FROM Enrollments JOIN Students ON Enrollments.student\_id = Students.student\_id

JOIN Courses ON Enrollments.course\_id = Courses.course\_id;

**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.**

SELECT Teacher.first\_name AS teacher\_firstName, Teacher.last\_name AS teacher\_lastName, Courses.course\_name

FROM Courses JOIN Teacher ON Courses.teacher\_id = Teacher.teacher\_id;

**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.**

SELECT Students.first\_name, Students.last\_name, Enrollments.enrollment\_date

FROM Enrollments JOIN Students ON Enrollments.student\_id = Students.student\_id

JOIN Courses ON Enrollments.course\_id = Courses.course\_id

WHERE Courses.course\_id = 3;

**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.**

SELECT first\_name, last\_name, date\_of\_birth, email, phone\_number

FROM Students LEFT JOIN Payments ON Students.student\_id = Payments.student\_id

WHERE Payments.student\_id IS NULL;

**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.**

SELECT courses.course\_id, courses.course\_name, courses.credits, courses.teacher\_id

FROM Courses LEFT JOIN Enrollments ON Courses.course\_id = Enrollments.course\_id

WHERE Enrollments.course\_id IS NULL;

**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.**

SELECT Students.student\_id, Students.first\_name, Students.last\_name, COUNT(Enrollments.course\_id) AS course\_count

FROM Enrollments JOIN Students ON Enrollments.student\_id = Students.student\_id

GROUP BY Students.student\_id

HAVING COUNT(Enrollments.course\_id) > 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 Teacher.teacher\_id, Teacher.first\_name, Teacher.last\_name

FROM Teacher LEFT JOIN Courses ON Teacher.teacher\_id = Courses.teacher\_id

WHERE Courses.teacher\_id IS NULL;

**TASK 4 – SUBQUERY AND ITS TYPE**

**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 course\_id, avg(student\_count) as average\_count from (

select course\_id, count (student\_id) as student\_count

from enrollments group by course\_id) as course\_counts

group by course\_id;

**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.student\_id, s.first\_name, s.last\_name,p.amount from students as s inner join payments as pon s.student\_id = p.student\_id where p.amount = ( select MAX(amount) from payments);

**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, max(e.enroll) as max\_enroll from courses as cinner join (

select course\_id, count(\*) as enrol from enrollments

group by course\_id) as e on c.course\_id=e.course\_id

group by c.course\_name having max(e.enroll) = (

select max(enroll\_count) from (

select count(\*) as enroll\_count from enrollments

group by course\_id

) as subquery

);

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

select t.teacher\_id, t.first\_name as teacher\_first\_name, t.last\_name as teacher\_last\_name,sum(p.amount) as total\_payments from teacher t

inner join courses c on t.teacher\_id = c.teacher\_id

inner join enrollments e on c.course\_id = e.course\_id

inner join payments p on e.student\_id = p.student\_id

group by t.teacher\_id, t.first\_name, t.last\_name;

**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.student\_id, s.first\_name, s.last\_name from

students s where (select count(distinct course\_id) from courses) =

(select count(distinct course\_id)

from enrollments e where e.student\_id = s.student\_id);

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

select t.first\_name from teacher as t where t.teacher\_id not in select c.teacher\_id from courses as c);

**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(age) as average\_age

from (select round(datediff (current\_date(), date\_of\_birth) / 365, 0) as age from students ) as student\_age;

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

select c.course\_id,c.course\_name from courses c

where c.course\_id not in ( select distinct course\_id from enrollments);

**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.first\_name, c.course\_name, (select sum(p.amount)

from payments p where p.student\_id = s.student\_id

and e.course\_id = c.course\_id ) as total\_amount

from students s inner join enrollments e on s.student\_id = e.student\_id

inner join courses con e.course\_id = c.course\_id;

**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 first\_name, last\_name from students

where student\_id in ( select student\_id from payments

group by student\_id having count(\*) > 1);

**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.student\_id, s.first\_name, sum(p.amount) as total\_payments from students s join payments pon s.student\_id group by s.student\_id, s.first\_name;

**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.student\_id) as count\_students

from courses as c left join enrollments as e on c.course\_id = e.course\_id group by course\_name;

**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 avg(p.amount)

from payments p

left join students s on p.student\_ids.student\_id;