**Task 1. Database Design:**

1. Create the database named "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

b. Courses

c. Enrollments

d. Teacher

e. Payments

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

ii. Courses

iii. Enrollments

iv. Teacher

v. Payments

CREATE TABLE Students (

student\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

date\_of\_birth DATE,

email VARCHAR(50),

phone\_number INT

);

CREATE TABLE Courses (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(50),

credits INT,

teacher\_id INT,

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

);

CREATE TABLE Enrollments (

enrollment\_id INT PRIMARY KEY, -- Primary Key

student\_id INT, -- Foreign Key references Students table

course\_id INT, -- Foreign Key references Courses table

enrollment\_date DATE,

CONSTRAINT fk\_student FOREIGN KEY (student\_id) REFERENCES Students(student\_id),

CONSTRAINT fk\_course FOREIGN KEY (course\_id) REFERENCES Courses(course\_id)

);

CREATE TABLE Teachers (

teacher\_id INT PRIMARY KEY, -- Primary Key

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

email VARCHAR(100)

);

CREATE TABLE Payments (

payment\_id INT PRIMARY KEY, -- Primary Key

student\_id INT, -- Foreign Key references Students table

amount DECIMAL(10, 2) NOT NULL,

payment\_date DATE,

CONSTRAINT fk\_student\_payment FOREIGN KEY (student\_id) REFERENCES Students(student\_id)

);

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

VALUES

(1, 'Raj', 'M', '1995-08-15', 'raj.m1995@gmail.com', 1234567890),

(2, 'Anu', 'Sharma', '1997-05-21', 'anu.sharma1997@gmail.com', 1234567891),

(3, 'Priya', 'Dharshini', '1998-03-14', 'priya.dharshini1998@gmail.com', 1234567892),

(4, 'Kamal', 'S', '1996-12-11', 'kamal.s1996@gmail.com', 1234567893),

(5, 'Mike', 'Ross', '1994-07-18', 'mike.ross1994@gmail.com', 1234567894),

(6, 'Raja', 'Singh', '1999-02-23', 'raja.singh1999@gmail.com', 1234567895),

(7, 'Anush', 'Sharma', '1995-10-30', 'anush.sharma1995@gmail.com', 1234567896),

(8, 'Shree', 'Raj', '1996-01-17', 'shree.raj1996@gmail.com', 1234567897),

(9, 'David', 'Miller', '1997-09-24', 'david.miller1997@gmail.com', 1234567898),

(10, 'MS', 'Dhoni', '1998-11-28', 'ms.dhoni1998@gmail.com', 1234567899);

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

VALUES

(1, 'Sachin', 'Tendulkar', 'sachin.tendulkar@gmail.com'),

(2, 'Virat', 'Kohli', 'virat.kohli@gmail.com'),

(3, 'MS', 'Dhoni', 'ms.dhoni@gmail.com'),

(4, 'Rohit', 'Sharma', 'rohit.sharma@gmail.com'),

(5, 'Shikhar', 'Dhawan', 'shikhar.dhawan@gmail.com'),

(6, 'KL', 'Rahul', 'kl.rahul@gmail.com'),

(7, 'Ravindra', 'Jadeja', 'ravindra.jadeja@gmail.com'),

(8, 'Jasprit', 'Bumrah', 'jasprit.bumrah@gmail.com'),

(9, 'Suresh', 'Raina', 'suresh.raina@gmail.com'),

(10, 'Yuvraj', 'Singh', 'yuvraj.singh@gmail.com');

**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 values(11,'John', 'Doe', '1995-08-15', 'john.doe@example.com', 1234567890);

1. 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
2. Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.

update teacher set email = 'ms.dhoni.newemail@gmail.com' where first\_name = 'MS' and last\_name = 'Dhoni';

1. 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 = 3 and course\_id = 3;

1. 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 = 6 where course\_id = 3;

1. 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 Students where student\_id = 3;

1. 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 = 1500.00 where payment\_id = 5;

**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 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 = 5

GROUP BY s.first\_name, s.last\_name;

1. 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(e.student\_id) as enrolled\_students

from courses c

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

group by c.course\_name

1. 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 s.first\_name, s.last\_name, s.student\_id

from students s

left join enrollments e on s.student\_id = e.student\_id

WHERE e.course\_id IS NULL;

1. 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 e.student\_id = s.student\_id

join courses c on c.course\_id = e.course\_id

1. 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 teachers t

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

1. 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.first\_name, s.last\_name, c.course\_name, e.enrollment\_date

from students s

join enrollments e on e.student\_id = s.student\_id

join courses c on c.course\_id = e.course\_id

1. 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, p.amount as total\_payments

from students s

join payments p on p.student\_id = s.student\_id

where p.amount is null

1. 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 c.course\_id = e.course\_id

WHERE e.enrollment\_id IS NULL;

1. 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.course\_id) AS course\_count, GROUP\_CONCAT(c.course\_name ORDER BY c.course\_name SEPARATOR ', ') AS courses

FROM Students s

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

HAVING COUNT(e.course\_id)> 1;

Or

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) > 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, t.teacher\_id

from teachers t

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

where c.course\_id is null