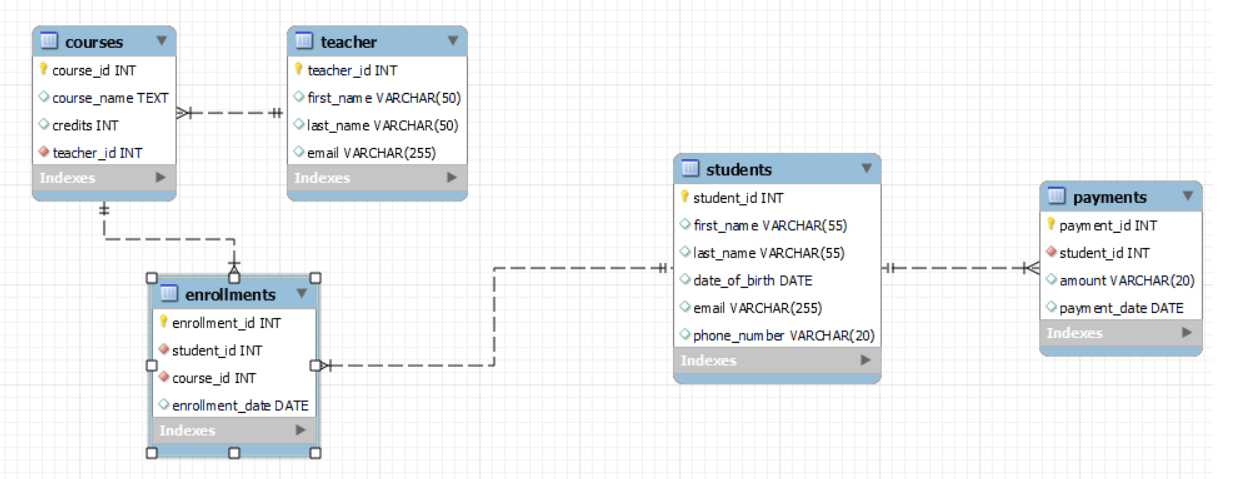
**Tasks 1: Database Design:**

1. Create the database named "SISDB":

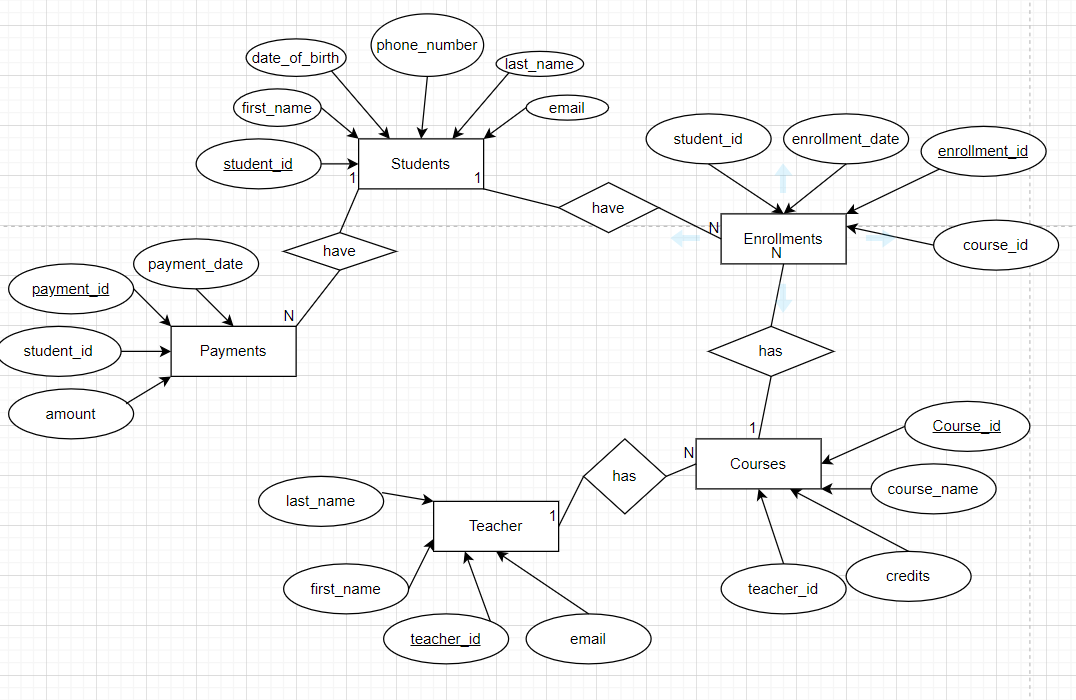
CREATE DATABASE SISDB;

USE SISDB;

1. 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. Payment.



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



1. Create appropriate Primary Key and Foreign Key constraints for referential integrity:

Students Table:

student\_id INT PRIMARY KEY

Teacher Table:

teacher\_id INT PRIMARY KEY

Courses Table:

course\_id INT Primary Key and

FOREIGN KEY(teacher\_id) REFERENCES Teacher(teacher\_id) ON DELETE CASCADE

Enrollments Table:

enrollment\_id INT Primary Key and

FOREIGN KEY(course\_id) REFERENCES Courses(course\_id) ON DELETE CASCADE

Payments Table:

payment\_id INT PRIMARY KEY and

FOREIGN KEY(student\_id) REFERENCES Students(student\_id) ON DELETE CASCADE

1. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

* Students:

CREATE TABLE Students (

student\_id INT PRIMARY KEY,

first\_name VARCHAR(55),

last\_name VARCHAR(55),

date\_of\_birth DATE,

email VARCHAR(255),

phone\_number VARCHAR(20));

* Teacher:

CREATE TABLE Teacher(

teacher\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

email VARCHAR(255)

);

* Courses:

CREATE TABLE Courses(

course\_id INT Primary Key,

course\_name TEXT,

credits INT,

teacher\_id INT NOT NULL,

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

);

* Enrollments:

CREATE TABLE Enrollments(

enrollment\_id INT Primary Key,

student\_id INT NOT NULL,

FOREIGN KEY(student\_id) REFERENCES Students(student\_id) ON DELETE CASCADE,

course\_id INT NOT NULL,

FOREIGN KEY(course\_id) REFERENCES Courses(course\_id) ON DELETE CASCADE,

enrollment\_date DATE

);

* Payments:

CREATE TABLE Payments(

payment\_id INT PRIMARY KEY,

student\_id INT NOT NULL,

FOREIGN KEY(student\_id) REFERENCES Students(student\_id) ON DELETE CASCADE,

amount VARCHAR(20),

payment\_date DATE

);

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

* Students:

INSERT INTO Students values

(1,'Avinash','Kumar','2001-02-10','avi.kr131000@gmail.com', 8210642402),

(2,'Ashish','Kumar','2001-06-26','ashish@gmail.com',8210642401),

(3,'Deepanshu','Patel','2000-02-10','deepanshu@gmail.com',8210642403),

(4,'Shashwat','Shivam','2000-05-28','shashwat@gmail.com',8210642404),

(5,'Prashant','Mahato','2001-02-12','prashant@gmail.com',8210642405),

(6,'Ayush','Pandey','2001-01-15','ayush@gmail.com',8210642406),

(7,'Pragyan','Pandey','2000-12-15','pragyan@gmail.com',8210642474),

(8,'Vibhor','Mishra','1999-05-12','vibhor@gmail.com',8210642407),

(9,'Pranav','Gandhi','1999-06-02','pranav@gmail.com',8210642408),

(10,'Mayank','Agarwal','2000-04-06','mayank@gmail.com',8210642409)

;

* Teacher:

INSERT INTO Teacher values

(1,'Suman','Das','Suman@gmail.com'),

(2,'Lakshay','Kaushik','Lakshay@gmail.com'),

(3,'Arnav','Pandey','arnav@gmail.com'),

(4,'Priyansh','Agarwal','priyansh@gmail.com'),

(5,'Rohit','Sharma','rohit@gmail.com'),

(6,'Virat','Kohli','virat@gmail.com'),

(7,'Steph','Curry','curry@gmail.com'),

(8,'Lebron','James','James@gmail.com'),

(9,'Stuart','Broad','Broad@gmail.com'),

(10,'Jimmy','Anderson','jimmy@gmail.com')

;

* Courses:

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

VALUES

(1, 'Maths', 3,1),

(2, 'Computer Science', 4,2),

(3, 'database', 2,3),

(4, 'Circuits', 4,4),

(5, 'English', 3,5),

(6, 'History', 3,6),

(7, 'Civics', 4,8),

(8, 'Geography', 1,7),

(9, 'Maths', 4,9),

(10, 'Computer Science', 4,10)

;

* Enrollments:

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

VALUES

(1, 2, 1, '2023-12-01'),

(2, 3, 2, '2023-12-02'),

(3, 5, 3, '2023-12-03'),

(4, 1, 4, '2023-12-04'),

(5, 4, 5, '2023-12-05'),

(6, 10, 6, '2023-12-01'),

(7, 9, 7, '2023-11-06'),

(8, 8, 8, '2023-11-03'),

(9, 7, 9, '2023-11-04'),

(10, 6, 10, '2023-11-05')

;

* Payments:

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

VALUES

(1, 2, '1000', '2023-12-06'),

(2, 3, '1500', '2023-02-06'),

(3, 4, '2000', '2023-02-06'),

(4, 1, '2500', '2023-02-05'),

(5, 5, '3000', '2023-02-06'),

(6, 6, '1000', '2023-12-06'),

(7, 7, '1500', '2023-02-06'),

(8, 8, '2000', '2023-02-06'),

(9, 9, '2500', '2023-02-05'),

(10, 10, '3000', '2023-02-06')

;

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

1. **Write SQL queries for the following tasks:**
2. 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](mailto:john.doe@example.com)

e. Phone Number: 1234567890 :

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

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.

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

VALUES ('11','1', '3', '2023-12-07');

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

UPDATE Teacher

SET email= 'kohli@gmail.com' WHERE teacher\_id='6';

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 = '2' AND course\_id = '1'AND enrollment\_date = '2023-12-01' ;

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 = '5' WHERE course\_id = '2;

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 Payments WHERE student\_id = '8';

DELETE FROM Enrollments WHERE student\_id = '8';

DELETE FROM Students WHERE student\_id = '8'

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 = '5000' WHERE payment\_id = 1;

**Tasks 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.student\_id, CONCAT(s.first\_name,' ',s.last\_name) AS FULL\_NAME, SUM(p.amount) AS total\_payments FROM Students s

JOIN Payments p ON s.student\_id = p.student\_id

WHERE s.student\_id = 2

GROUP BY s.student\_id;

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\_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;

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 FROM Students s

LEFT JOIN Enrollments e ON s.student\_id = e.student\_id

WHERE e.student\_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 s.student\_id = e.student\_id

JOIN Courses c ON e.course\_id = c.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 AS Teacher\_FirstName,

t.last\_name AS Teacher\_LastName,

c.course\_name AS Course\_Assigned

FROM Teacher t, Courses c WHERE t.teacher\_id = c.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 AS Student\_FirstName, s.last\_name AS Student\_LastName, e.enrollment\_date AS 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 = 'English';

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 AS Student\_FirstName, s.last\_name AS Student\_LastName

FROM Students s

LEFT JOIN Payments p ON s.student\_id = p.student\_id WHERE p.payment\_id 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\_id, 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 record:

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

COUNT(e.enrollment\_id) AS num\_enrollments 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 num\_enrollments > 1;

1. 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.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;

**Tasks 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 c.course\_id, c.course\_name,

AVG(enrollment\_count) AS average\_students\_enrolled

FROM Courses c LEFT JOIN ( SELECT course\_id,COUNT(DISTINCT student\_id) AS enrollment\_count

FROM Enrollments GROUP BY course\_id) e ON c.course\_id = e.course\_id

GROUP BY c.course\_id, c.course\_name;

1. 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 CONCAT(s.first\_name,s.last\_name) AS Full\_Name, a.amt FROM students s JOIN (

SELECT student\_id,sum(amount) amt FROM Payments

GROUP BY student\_id

ORDER BY amt DESC LIMIT 1) a ON s.student\_id=a.student\_id ;

1. 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\_id, c.course\_name, count(e.enrollment\_id) AS enrolled\_count

FROM Courses c LEFT JOIN Enrollments e ON c.course\_id = e.course\_id GROUP BY c.course\_id, c.course\_name

HAVING enrolled\_count = (SELECT max(enrolled\_count) FROM (SELECT count(enrollment\_id) AS enrolled\_count FROM Enrollments

GROUP BY course\_id) AS maximum\_enrolls);

1. 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,c.course\_id,c.course\_name,CONCAT(t.first\_name ,' ',t.last\_name )AS NAME\_OF\_TEACHER ,SUM(p.amount) AS total\_payments\_IN$

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

LEFT JOIN Enrollments e ON c.course\_id = e.course\_id LEFT JOIN Payments p ON e.student\_id = p.student\_id

GROUP BY c.course\_id,t.teacher\_id;

1. 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 NOT EXISTS (SELECT c.course\_id FROM Courses c WHERE NOT EXISTS ( SELECT e.course\_id FROM Enrollments e WHERE e.student\_id = s.student\_id AND e.course\_id = c.course\_id)

);

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

SELECT t.teacher\_id, t.first\_name, t.last\_name

FROM Teacher t WHERE NOT EXISTS (SELECT 1 FROM Courses c WHERE c.teacher\_id = t.teacher\_id);

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

SELECT AVG(student\_age) AS average\_age

FROM (SELECT student\_id, TIMESTAMPDIFF(YEAR, date\_of\_birth, CURDATE()) AS student\_age FROM Students) AS student\_ages;

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

SELECT course\_id, course\_name

FROM Courses WHERE course\_id NOT IN ( SELECT DISTINCT course\_id FROM Enrollments);

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

SELECT e.student\_id,concat(s.first\_name,' ',s.last\_name) AS FULL\_NAME ,e.course\_id,(SELECT sum(amount) FROM payments p

WHERE p.student\_id = e.student\_id) AS total\_payments

FROM enrollments e

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

GROUP BY e.student\_id,e.course\_id;

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

FROM Students s

JOIN (SELECT student\_id, COUNT(\*) AS payment\_count

FROM Payments

GROUP BY student\_id HAVING payment\_count > 1) p\_counts ON s.student\_id = p\_counts.student\_id;

1. 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, s.last\_name,

IFNULL(SUM(p.amount), 0) AS total\_payments

FROM Students s

LEFT JOIN Payments p ON s.student\_id = p.student\_id

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

1. 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 enrollment\_count

FROM Courses c

LEFT JOIN Enrollments e ON c.course\_id = e.course\_id

GROUP BY c.course\_id, c.course\_name;

1. 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.student\_id, CONCAT(s.first\_name, ' ',s.last\_name) AS FULL\_NAME, 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;