**ASSIGNMENT 2 - Student Information System (SIS)**

**TASK 1**

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.Create Table Students (

student\_id int primary key ,

first\_name varchar(50),

last\_name varchar(50),

date\_of\_birth varchar(15),

email varchar(50),

phone\_number varchar(15)

);

b.Create Table Teacher (

teacher\_id int primary key,

first\_name varchar(50),

last\_name varchar(50),

email varchar(100)

);

c.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)

);

d.Create Table Enrollments (

enrollment\_id int primary key,

student\_id int,

course\_id int,

enrollment\_date varchar(50),

foreign key (student\_id) references students(student\_id),

foreign key (course\_id) references courses(course\_id)

);

e.Create Table Payments (

payment\_id int primary key ,

student\_id int,

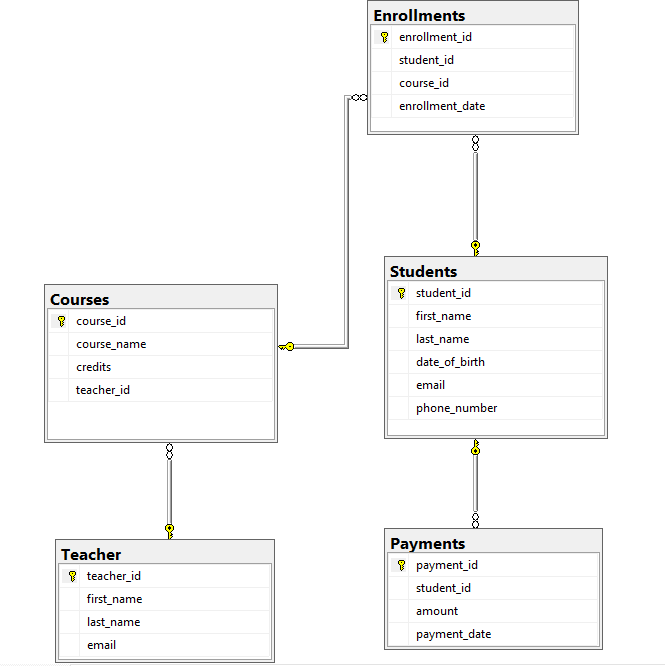
amount decimal(10, 2),

payment\_date varchar(50)

foreign key (student\_id) references students(student\_id)

);

1. **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**

a.Insert into Students (student\_id,first\_name, last\_name, date\_of\_birth, email, phone\_number) values

('101','Varun', 'Kumar', '2001-08-13', 'varunkumar@gmail.com', '9856739210'),

('102','Lakshmi', 'Praba', '2001-09-12', 'lakshmipraba@gmail.com', '9978322991'),

('103','Vanitha', 'Suresh', '2001-08-19', 'vanitha@gmail.com', '9932823135'),

('104','Divya', 'Kannan', '2001-11-08', 'divya@gmail.com', '9973256182'),

('105','Ram', 'Nishanth', '2001-09-19', 'ram@gmail.com', '9976471881'),

('106','Ragavi', 'Ramesh', '2001-09-29', 'ragavir@gmail.com', '9972189881'),

('107','Dhanushree', 'Durai', '2001-07-19', 'dhanushree@gmail.com', '9966471881'),

('108','Bharath', 'Raju', '2001-08-11', 'bharath@gmail.com', '8825871881'),

('109','Raja', 'Sekar', '2001-08-27', 'rajas@gmail.com', '9896479221'),

('110','Naveen', 'Raj', '2001-11-18', 'naveen@gmail.com', '9372821881');

select\*from Students

b.Insert into Teacher (teacher\_id, first\_name, last\_name, email) VALUES

(201, 'Nithish', 'Kumar', 'nithishkumar@gmail.com'),

(202, 'Meena', 'Ravi', 'meenravi@gmail.com'),

(203, 'Suresh', 'Babu', 'sureshbabu@gmail.com'),

(204, 'Priya', 'Venkatesh', 'priyav@gmail.com'),

(205, 'Raj', 'Mohan', 'rajmohan@gmail.com'),

(206, 'Revathi', 'Ramesh', 'revathi@gmail.com'),

(207, 'Karthik', 'Sundar', 'karthik@gmail.com'),

(208, 'Uma', 'Devi', 'umadevi@gmail.com'),

(209, 'Nithya',' Raja','nithya@gmail.com'),

(210, 'Lavanya', 'Natarajan', 'lavanyan@gmail.com');

select\*from Teacher

c. Insert into Courses (course\_id,course\_name, credits, teacher\_id) VALUES

('1001','English Communication', 2, 201),

('1002','Mathematics', 4, 202),

('1003','Compiler Design', 4,203),

('1004','Machin Learning', 2, 206),

('1005','Cybser Security', 3,205),

('1006','Database Management Systems', 4, 204),

('1007','Web Technologies', 3, 209),

('1008','Data Structures', 4, 207),

('1009','Principles of Management',3,208),

('1010','Environmental Studies', 2, 210);

select\*from Courses

d. Insert into Enrollments (enrollment\_id, student\_id, course\_id, enrollment\_date) VALUES

(1, 101, 1001, '2023-06-01'),

(2, 102, 1002, '2023-06-02'),

(3, 103, 1003, '2023-06-03'),

(4, 104, 1004, '2023-06-05'),

(5, 105, 1005, '2023-06-06'),

(6, 106, 1006, '2023-06-07'),

(7, 107, 1007, '2023-06-08'),

(8, 108, 1008, '2023-06-09'),

(9, 109, 1009, '2023-06-10'),

(10, 110, 1010, '2023-06-11');

select\*from Enrollments

e. Insert into Payments (payment\_id, student\_id, amount, payment\_date) VALUES

(501, 101, 10000, '2023-06-01'),

(502, 102, 10000, '2023-06-02'),

(503, 103, 10000, '2023-06-03'),

(504, 104, 13000, '2023-06-05'),

(505, 105, 14000, '2023-06-06'),

(506, 106, 14000, '2023-06-07'),

(507, 107, 12500, '2023-06-08'),

(508, 108, 13500, '2023-06-09'),

(509, 109, 14500, '2023-06-10'),

(510, 110, 9000, '2023-06-11');

Select\*from Payments

**TASK 2**

1. **Write an SQL query to insert a new student into the "Students" table with the following details: a. First Name: John © Hexaware Technologies Limited. All rights www.hexaware.com b. Last Name: Doe c. Date of Birth: 1995-08-15 d. Email: john.doe@example.com e. Phone Number: 1234567890**

Insert into Students (student\_id,first\_name, last\_name, date\_of\_birth, email, phone\_number)values ('111','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, 102, 1002, '2024-06-13');

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 = 'raj.mohan@gmail.com' where teacher\_id = 205;

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 = 102 and course\_id = 1002;

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 = 205 where course\_id = 1002;

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 = 110;

Delete from Payments WHERE student\_id = 110;

Delete from Students WHERE student\_id = 110;

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 = 7500 Where payment\_id = 501;

**TASK 3**

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, 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 =104 group by s.student\_id, 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 total from Courses c JOIN Enrollments e ON c.course\_id = e.course\_id Group by c.course\_name order by total desc;

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.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 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, t.last\_name, c.course\_name from Teacher t LEFT JOIN Courses c ON 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, 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\_name = 'Mathematics';

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 FROM Students sLEFT 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.course\_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.student\_id, s.first\_name, s.last\_name 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;

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 .teacher\_id, t.first\_name, t.last\_name from Teacher t LEFT JOIN Courses c ON t.teacher\_id = c.teacher\_id Wheree c.course\_id is 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 course\_id, Count(student\_id) From Enrollments Group by course\_id;

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 student\_id FROM Payments WHERE amount IN (select MAX(amount) From Payments);

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 course\_id from Enrollments Group By course\_id HAVING COUNT(\*) = ( Selecct MAX(cnt) FROM (SELECT course\_id, COUNT(\*) AS cnt FROM Enrollments Group by course\_id) AS counts);

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

Select teacher\_id, sum(amount) AS total\_payment from Payments

Where course\_id IN ( SELECT course\_id FROM Courses)group by 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 student\_id FROM Enrollments Group by student\_id HAVING COUNT(\*) = (select COUNT(\*) from Courses)

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

Select teacher\_name from Teachers Where teacher\_id NOT IN ( SELECT teacher\_id FROM Courses WHERE teacher\_id Is not null);

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(YEAR(GETDATE()) - YEAR(date\_of\_birth)) AS avg\_age

From Students;

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

Select course\_name from CoursesWHERE course\_id not in (SELECT 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 p.student\_id, e.course\_id, SUM(p.amount) AS total\_payment from Payments p JOIN Enrollments e ON p.student\_id = e.student\_id GROUP BY p.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.first\_name,s.last\_name SUM(p.amount) AS total\_payment from Students s JOIN Payments p ON s.student\_id = p.student\_id group by s.first\_name,s.last\_name;

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.first\_name,s.last\_name, SUM(p.amount) AS total\_payment from Students s Join Payments p ON s.student\_id = p.student\_id group by 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 total\_enrolled from Courses c

JOIN Enrollments e ON c.course\_id = e.course\_id group by 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.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.first\_name,s.last\_name