**SQL ASSIGNMENT**

**STUDENT INFORMATION SYSTEM**

**Task 1. Database Design:**

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

Query to create a database is:-  **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**

**b. Courses**

**c. Enrollments**

**d. Teacher**

**e. Payments**

**CREATING TABLES FOR THE DATABASE:-**

**1.Students**

• student\_id (Primary Key)

• first\_name

• last\_name

• date\_of\_birth

• email

• phone\_number

Query to create the above table is:-

CREATE TABLE students (

student\_id int NOT NULL,

first\_name varchar(10) NOT NULL,

last\_name varchar(10) NOT NULL,

date\_of\_birth DATE ,

email varchar(15),

phone\_number varchar(10) NOT NULL,

PRIMARY KEY(student\_id)

);

**4. Teacher**

• teacher\_id (Primary Key)

• first\_name

• last\_name

• email

Query to create the above table is:-

CREATE TABLE teacher (

teacher\_id int NOT NULL,

first\_name varchar(10) NOT NULL,

last\_name varchar(10) NOT NULL,

email varchar(15),

PRIMARY KEY(teacher\_id)

);

**2. Courses**

• course\_id (Primary Key)

• course\_name

• credits

• teacher\_id (Foreign Key)

Query to create the above table is:-

CREATE TABLE courses (

course\_id int NOT NULL,

course\_name varchar(10) NOT NULL,

credits int NOT NULL,

teacher\_id int NOT NULL,

PRIMARY KEY(course\_id),

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

);

**3. Enrollments**

• enrollment\_id (Primary Key)

• student\_id (Foreign Key)

• course\_id (Foreign Key)

• enrollment\_date

Query to create the above table is:-

CREATE TABLE enrollments (

enrollment\_id int NOT NULL,

student\_id int NOT NULL,

course\_id int NOT NULL,

enrollment\_date DATE,

PRIMARY KEY(enrollment\_id),

FOREIGN KEY(student\_id) REFERENCES students(student\_id),

FOREIGN KEY(course\_id) REFERENCES courses(course\_id)

);

**5. Payments**

• payment\_id (Primary Key)

• student\_id (Foreign Key)

• amount

• payment\_date

Query to create the above table is:-

CREATE TABLE payments (

payment\_id int NOT NULL,

student\_id int NOT NULL,

amount int NOT NULL,

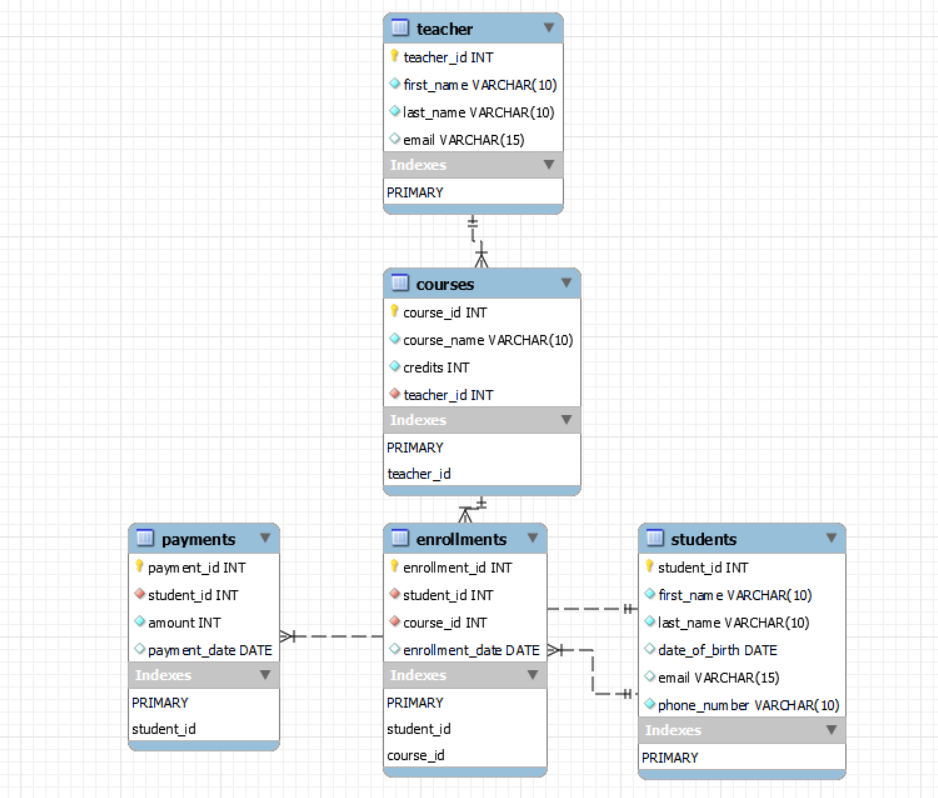
payment\_date DATE ,

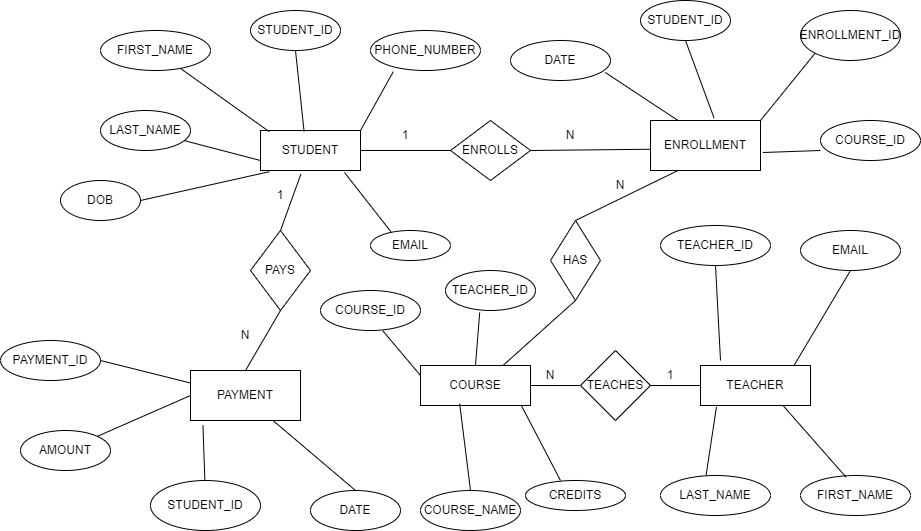
PRIMARY KEY(payment\_id),

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.**

The appropriate primary key and foreign key constraints for referential integrity has been done during the second step only i.e, during table creation only.

For students table the constraint is PRIMARY KEY(student\_id)

For teacher table the constraint is PRIMARY KEY(teacher\_id)

For courses table the constraint is PRIMARY KEY(course\_id),

FOREIGN KEY(teacher\_id) REFERENCES teacher(teacher\_id) here foreign key teacher id refers to the teacher\_id column of the teachers table.

For enrollments table the constraint is PRIMARY KEY(enrollment\_id),

FOREIGN KEY(student\_id) REFERENCES students(student\_id),

FOREIGN KEY(course\_id) REFERENCES courses(course\_id) here the foreign key student\_id refers to the student\_id column of the students table and the foreign key course\_id refers o the course\_id column of the courses table.

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

**i. Students**

**ii. Courses**

**iii. Enrollments**

**iv. Teacher**

**v. Payments**

1. For students table :-

INSERT INTO students VALUES

(1731,'raghu','kumar','1999-08-15','rkum@gmail.com','9876543210'),

(1732,'pavan','kalyan','2001-09-02','pkal@gmail.com','9876543211'),

(1733,'mahesh','babu','2002-07-24','mbob@gmail.com','9876543212'),

(1734,'ram','charan','1999-09-17','rcha@gmail.com','9876543213'),

(1735,'suriya','sivakumar','1998-02-11','ssiv@gmail.com','9876543214'),

(1736,'amar','deep','2000-01-21','aeep@gmail.com','9876543215'),

(1737,'prince','yaseen','2001-06-14','ypri@gmail.com','9876543216'),

(1738,'satya','babu','1997-10-29','sabu@gmail.com','9876543217'),

(1739,'manoj','saravana','2002-03-10','msar@gmail.com','9876543218'),

(1740,'vinod','kumar','1999-09-25','vkum@gmail.com','9876543219')

;

1. For teachers table:-

INSERT INTO teacher VALUES

(1401,'ram','kumar','r1@gmail.com'),

(1402,'shyam','kumar','s1@gmail.com'),

(1403,'kiran','kumar','k1@gmail.com'),

(1404,'siva','kumar','ss1@gmail.com'),

(1405,'sravan','kumar','sss1@gmail.com'),

(1406,'varun','kumar','v1@gmail.com'),

(1407,'trilok','kumar','t1@gmail.com'),

(1408,'praveen','kumar','p1@gmail.com'),

(1409,'vinay','kumar','vv1@gmail.com'),

(1410,'pranay','kumar','pp1@gmail.com')

;

1. For courses table:-

INSERT INTO courses VALUES

(101,'cse1',5,1408),

(102,'ece1',5,1401),

(103,'mec1',4,1410),

(107,'cse2',4,1402),

(109,'ece2',4,1409),

(111,'mec2',5,1403),

(113,'cse3',3,1407),

(119,'ece3',3,1404),

(127,'mec3',3,1406),

(129,'phy',2,1405)

;

1. For enrollments table:-

INSERT INTO enrollments VALUES

(201,1731,101,'2023-06-17'),

(202,1735,111,'2023-06-11'),

(203,1737,127,'2023-07-01'),

(204,1737,129,'2023-06-23'),

(205,1731,107,'2023-06-30'),

(206,1732,101,'2023-06-25'),

(207,1735,109,'2023-06-16'),

(208,1736,113,'2023-06-28'),

(209,1734,127,'2023-06-21'),

(210,1734,102,'2023-06-09')

;

1. For payments table:-

INSERT INTO payments VALUES

(601,1731,6000,'2023-06-17'),

(602,1735,3000,'2023-06-11'),

(603,1737,5000,'2023-07-01'),

(604,1737,4000,'2023-06-23'),

(605,1731,7000,'2023-06-30'),

(606,1732,6000,'2023-06-25'),

(607,1735,7500,'2023-06-16'),

(608,1736,4500,'2023-06-28'),

(609,1734,5000,'2023-06-21'),

(610,1734,5500,'2023-06-09')

;

**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**

Here they havent given the student id . In student table student id is a primary key so it cant be null so I am giving a randomn id and I am entering the value into the table.

Query to insert a new student is:-

INSERT INTO students value

(1741'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.

Query is:-

INSERT INTO enrollments

Value (211,1739,111,CURRENT\_DATE)

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

Query is:-

UPDATE teacher SET email='updtd@gmail.com' where teacher\_id=1403;

1. Write an SQL query to delete a specific enrollment record from the "Enrollments" table.

Query is:-

DELETE FROM enrollments where student\_id=1732;

Select an enrollment record based on the student and course.

Query is:-

SELECT \* FROM enrollments WHERE student\_id=1737 AND course\_id=127;

1. Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.

Query is:-

UPDATE courses SET teacher\_id=1410 WHERE course\_id=119;

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.

Query is:-

DELETE FROM students where student\_id=1740;

DELETE FROM enrollments where student\_id=1740;

1. Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.

Query is:-

UPDATE payments SET amount =9000 where payment\_id=606;

**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

Query is:-

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 = 1734;

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.

Query is:-

SELECT 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;

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.

Query is:-

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

FROM Students s

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

WHERE e.enrollment\_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.

Query is:-

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;

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.

Query is:-

SELECT t.first\_name, t.last\_name, c.course\_name

FROM Teacher t

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

Query is:-

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 = 101;

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.

Query is:-

SELECT 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;

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.

Query is:-

SELECT c.course\_name

FROM Courses c

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

WHERE e.enrollment\_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.

Query is:-

SELECT s.first\_name, s.last\_name, COUNT(e.enrollment\_id) AS course\_count

FROM Students s

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

GROUP BY s.student\_id

HAVING course\_count > 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

Query is:-

SELECT 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

**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.

Query is:-

SELECT AVG(student\_count) AS average\_students\_per\_course

FROM (SELECT course\_id, COUNT(student\_id) AS student\_count FROM Enrollments GROUP BY course\_id) AS counts;

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.

Query is:-

SELECT 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);

1. Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the course(s) with the maximum enrollment count.

Query is:-

SELECT course\_id, course\_name, enrollment\_count

FROM (

SELECT c.course\_id, 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

) AS enrollment\_counts

WHERE enrollment\_count = (

SELECT MAX(enrollment\_count)

FROM (

SELECT c.course\_id, 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

) AS max\_counts

);

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

Query is:-

SELECT t.teacher\_id, t.first\_name, t.last\_name, SUM(p.amount) AS total\_payments

FROM Teacher t

JOIN Courses c ON t.teacher\_id = c.teacher\_id

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

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.

Query is:-

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

FROM

Students s

WHERE

(SELECT COUNT(DISTINCT course\_id) FROM Courses) =

(SELECT COUNT(DISTINCT e.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.

Query is:-

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

FROM Teacher t

WHERE

NOT EXISTS (

SELECT 1

FROM Courses c

WHERE c.teacher\_id = t.teacher\_id

);

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

Query is:-

SELECT AVG(student\_age) AS average\_age

FROM

(

SELECT DATEDIFF(CURRENT\_DATE, date\_of\_birth) / 365 AS student\_age

FROM Students

) AS student\_ages;

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

records.

Query is:-

SELECT course\_id,course\_name

FROM Courses c

WHERE NOT EXISTS (

SELECT 1 FROM Enrollments e

WHERE e.course\_id = c.course\_id

);

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

Query is:-

SELECT s.first\_name, s.last\_name, c.course\_name, SUM(p.amount) AS total\_payments

FROM Students s

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

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

JOIN Courses c ON e.course\_id = c.course\_id

GROUP BY s.student\_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.

Query is:-

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

FROM Students s

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

GROUP BY s.student\_id

HAVING COUNT(p.payment\_id) > 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.

Query is:-

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

FROM Students s

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

GROUP BY s.student\_id;

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

Query is:-

SELECT 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;

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.

Query is:-

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

AVG(p.amount) AS average\_payment\_amount

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;