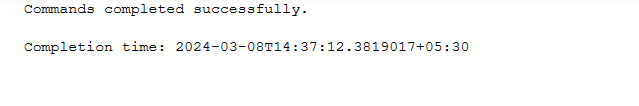
**Assignment(Student Information System (SIS)) – MS SQL**

- UDAYSANSTOSHKUMAR BURLU

**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

--b. Courses

--c. Enrollments

--d. Teacher

--e. Payments

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)

);

CREATE TABLE Teacher (

teacher\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

email VARCHAR(100)

);

CREATE TABLE Courses (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(100),

credits INT,

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

);

CREATE TABLE Enrollments (

enrollment\_id INT PRIMARY KEY,

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

course\_id INT FOREIGN KEY REFERENCES Courses(course\_id),

enrollment\_date DATE

);

CREATE TABLE Payments (

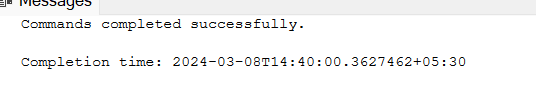
payment\_id INT PRIMARY KEY,

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

amount DECIMAL(10, 2),

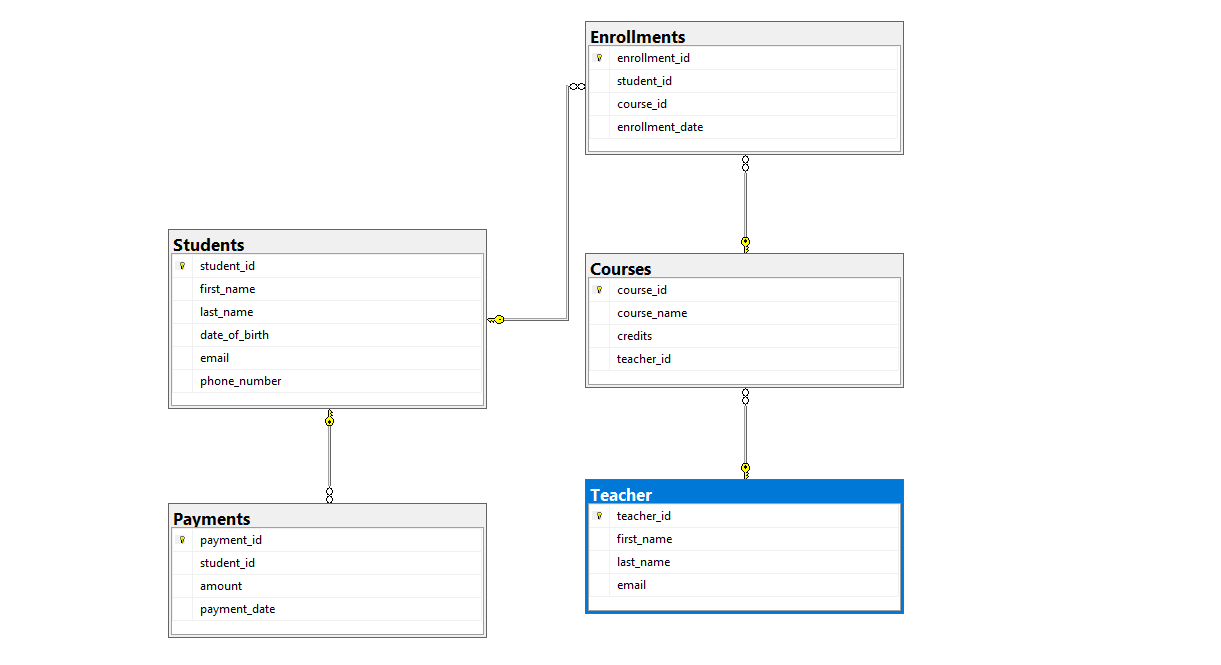
payment\_date DATE

);

****

--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.

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

VALUES

(1, 'Udaysk', 'Lastname1', '1990-01-15', 'udaysk@example.com', '555-1234'),

(2, 'Harsha', 'Patnaik', '1992-05-20', 'harsha.patnaik@example.com', '555-5678'),

(3, 'Hemanth', 'Kumar', '1988-09-10', 'hemanth.kumar@example.com', '555-9876'),

(4, 'Prabhas', 'Lastname4', '1995-03-25', 'prabhas@example.com', '555-4321'),

(5, 'Deepak', 'Lastname5', '1993-11-08', 'deepak@example.com', '555-8765'),

(6, 'Sudheer', 'Lastname6', '1997-07-12', 'sudheer@example.com', '555-2109'),

(7, 'Sunil', 'Lastname7', '1991-02-18', 'sunil@example.com', '555-6543'),

(8, 'Anil', 'Lastname8', '1994-06-30', 'anil@example.com', '555-1098'),

(9, 'Raju', 'Lastname9', '1989-12-05', 'raju@example.com', '555-5432'),

(10, 'Ravi', 'Lastname10', '1996-08-22', 'ravi@example.com', '555-9870'),

(11, 'Vinay', 'Lastname11', '1998-04-14', 'vinay@example.com', '555-1230'),

(12, 'Sekhar', 'Lastname12', '1997-11-02', 'sekhar@example.com', '555-4567');

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

VALUES

(1, 'Introduction to Programming', 3, 101),

(2, 'Database Management', 4, 102),

(3, 'Web Development', 3, 103),

(4, 'Data Science Fundamentals', 4, 104),

(5, 'Software Engineering Principles', 3, 105);

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

VALUES

(1, 1, 1, '2024-02-27'),

(2, 2, 3, '2024-02-28'),

(3, 3, 2, '2024-02-29'),

(4, 4, 4, '2024-03-01');

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

VALUES

(101, 'John', 'Smith', 'john.smith@example.com'),

(102, 'Jane', 'Doe', 'jane.doe@example.com'),

(103, 'Michael', 'Johnson', 'michael.j@example.com'),

(104, 'Emily', 'Williams', 'emily.w@example.com'),

(105, 'David', 'Brown', 'david.b@example.com');

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

VALUES

(1, 1, 100.00, '2024-03-03'),

(2, 2, 75.50, '2024-03-04'),

(3, 3, 120.00, '2024-03-05'),

(4, 4, 90.25, '2024-03-06'),

(5, 5, 150.50, '2024-03-07');

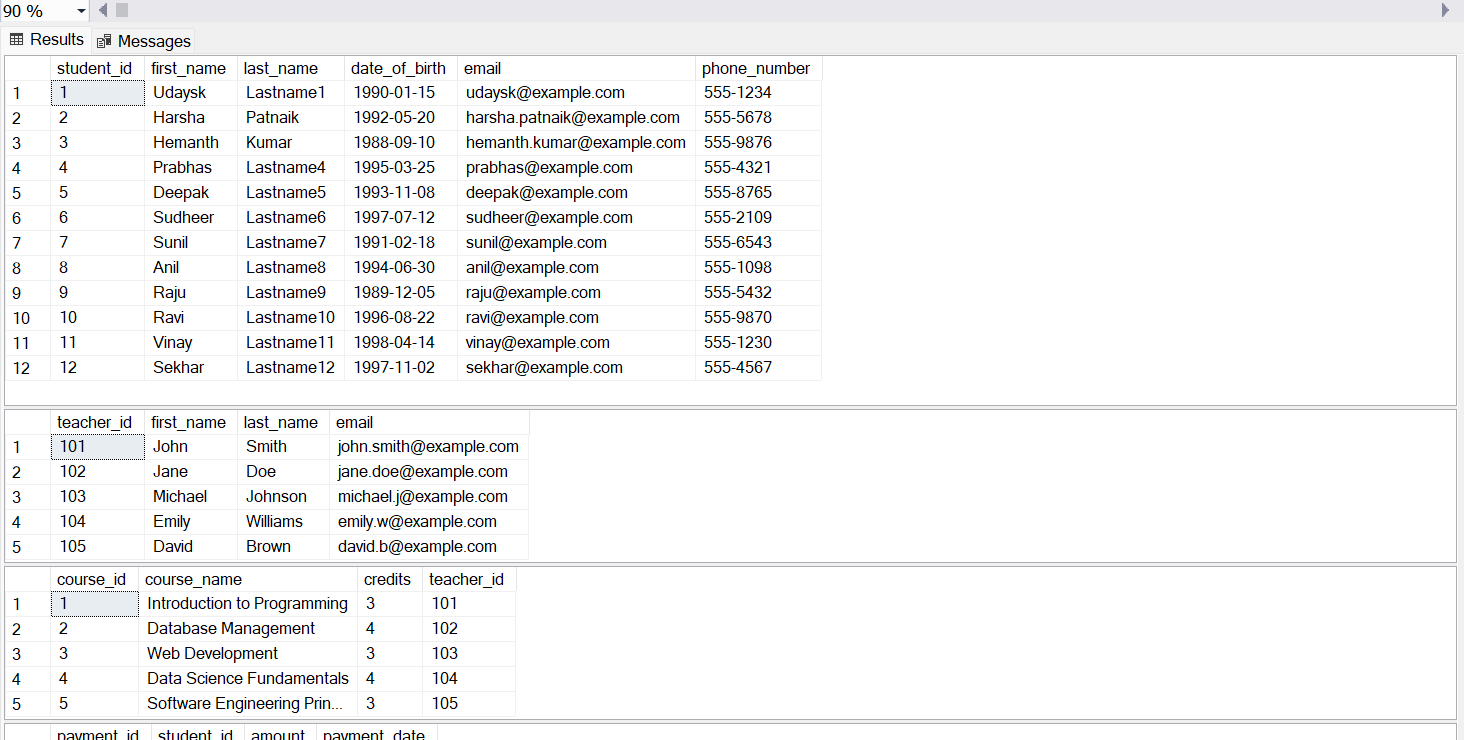
SELECT \* FROM STUDENTS;

SELECT \* FROM Teacher;

SELECT \* FROM Courses;

SELECT \* FROM Payments;

SELECT \* FROM Enrollments;

****

**A screenshot of a computer

Description automatically generated**

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

--Task 2

--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

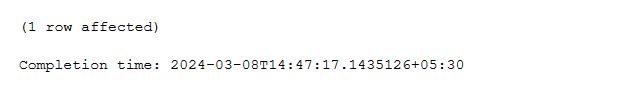
--c. Date of Birth: 1995-08-15

--d. Email: john.doe@example.com

--e. Phone Number: 1234567890

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

VALUES (13, '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 (7, 1, 2, '2024-03-01');

**A screenshot of a computer

Description automatically generated**

--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 = 'TEACHER@GMAIL.COM'

WHERE teacher\_id = 105

**A number and numbers on a white background

Description automatically generated**

--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

**A number and numbers on a white background

Description automatically generated**

--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 = 104

WHERE course\_id = 3

A number and date on a white background

Description automatically generated

--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.

BEGIN TRANSACTION;

DELETE FROM Enrollments

WHERE student\_id = 5;

DELETE FROM Payments

WHERE student\_id = 5;

DELETE FROM Students

WHERE student\_id = 5;

COMMIT;

**A number on a white background

Description automatically generated**

--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 = 300

WHERE payment\_date BETWEEN '2024-03-04' AND '2024-03-06'

**A number and time on a white background

Description automatically generated**

**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 student\_id, SUM(amount) AS total\_payments

FROM Payments

WHERE student\_id = 1

GROUP BY student\_id;

****

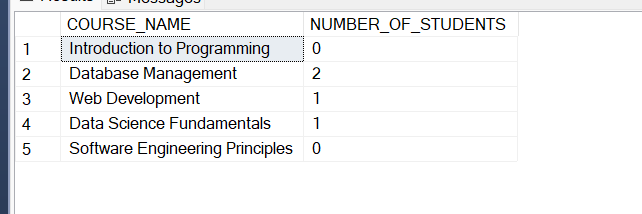
--2 Write an SQL query to retrieve a list of courses along with the count of students enrolled in each course. Use a JOIN operation between the "Courses" table and the "Enrollments" table.

SELECT COURSES.COURSE\_NAME, COUNT(ENROLLMENTS.STUDENT\_ID) NUMBER\_OF\_STUDENTS

FROM COURSES LEFT JOIN ENROLLMENTS

ON COURSES.COURSE\_ID = ENROLLMENTS.COURSE\_ID

GROUP BY COURSES.COURSE\_ID, COURSES.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 S.STUDENT\_ID, 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;

**A screenshot of a computer

Description automatically generated**

--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 s.student\_id, s.first\_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

**A screenshot of a computer

Description automatically generated**

--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 T.FIRST\_NAME TEACHER, C.COURSE\_NAME

FROM Courses C

JOIN Teacher T

ON C.teacher\_id = T.teacher\_id

SELECT T.FIRST\_NAME TEACHER, C.COURSE\_NAME

FROM tEACHER T

JOIN COURSES C

ON C.teacher\_id = T.teacher\_id

**A screen shot of a computer

Description automatically generated**

--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 s.FIRST\_NAME, E.enrollment\_date, 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

**A screenshot of a computer

Description automatically generated**

--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 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;

**A screenshot of a computer

Description automatically generated**

--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 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;

**A screenshot of a computer

Description automatically generated**

--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.

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

VALUES (8, 1, 2, '2024-03-01');

SELECT DISTINCT E.STUDENT\_ID, COUNT(E.COURSE\_ID) COUNTOFCOURSES

FROM ENROLLMENTS E

JOIN ENROLLMENTS C

ON E.STUDENT\_ID = C.STUDENT\_ID

GROUP BY E.STUDENT\_ID, E.COURSE\_ID

HAVING COUNT(E.COURSE\_ID) > 1;

**A screenshot of a computer

Description automatically generated**

--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 assignment

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.TEACHER\_ID IS NULL;

**A screenshot of a computer

Description automatically generated**

**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 E.COURSE\_ID, COUNT(E.STUDENT\_id)

FROM Enrollments E

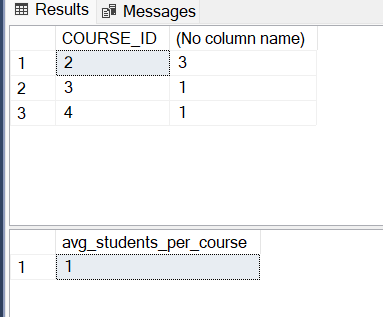
GROUP BY E.course\_id

SELECT AVG(enrollment\_count) AS avg\_students\_per\_course

FROM (SELECT course\_id, COUNT(DISTINCT student\_id) AS enrollment\_count

FROM Enrollments

GROUP BY course\_id) AS CourseEnrollmentCounts;

****

--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 STUDENT\_ID

FROM PAYMENTS

WHERE AMOUNT = (

SELECT MAX(AMOUNT) FROM Payments)

**A screenshot of a computer

Description automatically generated**

--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 COURSE\_ID, COUNT(student\_id) FROM Enrollments

group by course\_id

having COUNT(student\_id) = (

SELECT MAX(ENROLLMENTCOUNT) FROM (

SELECT COURSE\_ID, COUNT(student\_id) ENROLLMENTCOUNT

FROM Enrollments

group by course\_id

) ENROLLMENTCOUNT);

**A screenshot of a computer

Description automatically generated**

--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, 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;

**A screenshot of a computer

Description automatically generated**

--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 STUDENT\_ID, COUNT(COURSE\_ID)

FROM Enrollments

GROUP BY student\_id

HAVING COUNT(COURSE\_ID) = (SELECT COUNT(DISTINCT COURSE\_ID) FROM Courses)

**A screenshot of a computer

Description automatically generated**

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

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

FROM Teacher

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

**A screenshot of a computer

Description automatically generated**

--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(DATEDIFF(YEAR, date\_of\_birth, GETDATE())) AS average\_age

FROM Students;

**A screenshot of a computer

Description automatically generated**

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

SELECT COURSE\_ID

FROM COURSES

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

**A screenshot of a computer

Description automatically generated**

--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 e.student\_id, e.course\_id,

(SELECT SUM(amount) FROM Payments p WHERE p.student\_id = e.student\_id) AS total\_payments

FROM Enrollments e;

**A screenshot of a computer

Description automatically generated**

--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 STUDENT\_ID, FIRST\_NAME, LAST\_NAME

FROM STUDENTS

WHERE STUDENT\_ID IN (

SELECT student\_id

FROM Payments

GROUP BY student\_id

HAVING COUNT(payment\_id) > 1

);

**A screenshot of a computer

Description automatically generated**

--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, SUM(P.AMOUNT)

FROM Students S

JOIN Payments P

ON S.student\_id = P.payment\_id

GROUP BY S.student\_id

**A screenshot of a computer

Description automatically generated**

--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\_ID, C.course\_NAME, COUNT(E.STUDENT\_ID) STUDENTCOUNT

FROM Courses C

JOIN ENROLLMENTS E

ON C.course\_id = E.course\_id

GROUP BY C.COURSE\_ID, C.course\_NAME

**A screenshot of a computer

Description automatically generated**

--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 S.STUDENT\_ID, AVG(P.AMOUNT) AVGAMOUNT

FROM Students S

JOIN Payments P

ON S.student\_id = P.payment\_id

GROUP BY S.student\_id

SELECT AVG(amount) AS average\_payment\_amount

FROM Payments;

**A screenshot of a computer

Description automatically generated**