1-Creating the Tables:

SQLQuery9.sql - (l...IR58LH\mmheg (63))*

```sql
-- Create the Students table
CREATE TABLE Students (
    student_id INT IDENTITY(1,1) PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    date_of_birth DATE
);

-- Create the Instructors table
CREATE TABLE Instructors (
    instructor_id INT IDENTITY(1,1) PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100)
);

-- Create the Courses table with an additional instructor_id column
CREATE TABLE Courses (
    course_id INT IDENTITY(1,1) PRIMARY KEY,
    course_name VARCHAR(100),
    course_description TEXT,
    instructor_id INT,
    FOREIGN KEY (instructor_id) REFERENCES Instructors(instructor_id)
);

-- Create the Enrollments table
CREATE TABLE Enrollments (
    enrollment_id INT IDENTITY(1,1) PRIMARY KEY,
    student_id INT FOREIGN KEY REFERENCES Students(student_id),
    course_id INT FOREIGN KEY REFERENCES Courses(course_id),
    enrollment_date DATE
);
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2024-09-02T00:58:33.1930821+03:00
```

100 %

2-inserting students:

```sql
INSERT INTO Students (first_name, last_name, email, date_of_birth) VALUES
('Ahmed', 'Mostafa', 'ahmed.mostafa@example.com', '2000-05-12'),
('Fatma', 'Hassan', 'fatma.hassan@example.com', '2001-08-23'),
('Omar', 'El-Sayed', 'omar.elsayed@example.com', '2002-11-30'),
('Sara', 'Ali', 'sara.ali@example.com', '2003-07-19'),
('Mohamed', 'Khaled', 'mohamed.khaled@example.com', '2004-02-15'),
('Yasmin', 'Ibrahim', 'yasmin.ibrahim@example.com', '2000-09-08'),
('Khaled', 'Abdelrahman', 'khaled.abdelrahman@example.com', '2001-03-25'),
('Aya', 'Gamal', 'aya.gamal@example.com', '2002-06-14'),
('Hassan', 'Mahmoud', 'hassan.mahmoud@example.com', '2005-12-21'),
('Mona', 'Tarek', 'mona.tarek@example.com', '2004-10-10');
```

%

Messages

```
(10 rows affected)

Completion time: 2024-09-02T01:01:29.6650610+03:00
```

3-inserting instructors:

SQLQuery9.sql - (l...IR58LH\mmheg (63))*

```sql
INSERT INTO Instructors (first_name, last_name, email) VALUES
('Ahmed', 'Ezab', 'ahmed.ezab@gmail.com'),
('Mona', 'Shawky', 'mona.shawky@gmail.com'),
('Tarek', 'Adel', 'tarek.adel@gmail.com');
```

100 %

Messages

```
(3 rows affected)

Completion time: 2024-09-02T01:00:05.8437132+03:00
```

4- inserting Courses:

```sql
INSERT INTO Courses (course_name, course_description, instructor_id) VALUES
('Mathematics', 'An introductory course to Mathematics', 1),  -- Assign to Ahmed
('Computer Science', 'Fundamentals of programming, algorithms, and data structures', 2),  -- Assign to Mona
('Physics', 'Basic principles of physics including mechanics and thermodynamics', 3),  -- Assign to Tarek
('History', 'A survey of world history from ancient to modern times', 1),  -- Assign to Ahmed
('Biology', 'Introduction to biological sciences, covering cell biology, genetics, and ecology', 2);  -- Assign to Mona
```

100 %

Messages

```
(5 rows affected)

Completion time: 2024-09-02T01:00:50.2875896+03:00
```

5- inserting enrollments:

```sql
INSERT INTO Enrollments (student_id, course_id, enrollment_date) VALUES
(1, 1, '2023-09-01'),
(2, 1, '2023-09-02'),
(3, 2, '2023-09-03'),
(4, 3, '2023-09-04'),
(5, 4, '2023-09-05'),
(6, 5, '2023-09-06'),
(7, 1, '2023-09-07'),
(8, 2, '2023-09-08'),
(9, 3, '2023-09-09'),
(10, 4, '2023-09-10'),
(1, 5, '2023-09-11'),
(2, 3, '2023-09-12'),
(3, 4, '2023-09-13'),
(4, 5, '2023-09-14'),
(5, 2, '2023-09-15');
```

Messages

```
(15 rows affected)

Completion time: 2024-09-02T01:02:06.1797466+03:00
```

6-Select all students

```sql
--Retrieve a list of all students for administrative purposes or generating reports.
SELECT * FROM dbo.Students;
```

100 %

Results | Messages

	student_id	first_name	last_name	email	date_of_birth
1	1	Ahmed	Mostafa	ahmed.mostafa@example.com	2000-05-12
2	2	Fatma	Hassan	fatma.hassan@example.com	2001-08-23
3	3	Omar	El-Sayed	omar.elsayed@example.com	2002-11-30
4	4	Sara	Ali	sara.ali@example.com	2003-07-19
5	5	Mohamed	Khaled	mohamed.khaled@example.com	2004-02-15
6	6	Yasmin	Ibrahim	yasmin.ibrahim@example.com	2000-09-08
7	7	Khaled	Abdelrahman	khaled.abdelrahman@example.com	2001-03-25
8	8	Aya	Gamal	aya.gamal@example.com	2002-06-14
9	9	Hassan	Mahmoud	hassan.mahmoud@example.com	2005-12-21
10	10	Mona	Tarek	mona.tarek@example.com	2004-10-10

7-Select all courses:

SQLQuery2.sql - (l...IR58LH\mmheg (66))*

```sql
--Retrieve a list of all courses to display in the course catalog or during enrollment periods.
SELECT * FROM dbo.Courses;
```

100 %

Results | Messages

	course_id	course_name	course_description
1	1	Mathematics	An introductory course to Mathematics
2	2	Computer Science	Fundamentals of programming, algorithms, and da...
3	3	Physics	Basic principles of physics including mechanics a...
4	4	History	A survey of world history from ancient to modern ti...
5	5	Biology	Introduction to biological sciences, covering cell ...

8-Select all enrollments with student names and course names:

```sql
--Generate a detailed report showing which students are enrolled in which courses.
SELECT
    e.enrollment_id,
    s.first_name + ' ' + s.last_name AS student_name,
    c.course_name
FROM
    dbo.Enrollments e
    JOIN dbo.Students s ON e.student_id = s.student_id
    JOIN dbo.Courses c ON e.course_id = c.course_id;
```

Results | Messages

	enrollment_id	student_name	course_name
	1	Ahmed Mostafa	Mathematics
	2	Fatma Hassan	Mathematics
	3	Omar El-Sayed	Computer Science
	4	Sara Ali	Physics
	5	Mohamed Khaled	History
	6	Yasmin Ibrahim	Biology
	7	Khaled Abdelrahman	Mathematics
	8	Aya Gamal	Computer Science
	9	Hassan Mahmoud	Physics
0	10	Mona Tarek	History
1	11	Ahmed Mostafa	Biology
2	12	Fatma Hassan	Physics
3	13	Omar El-Sayed	History
4	14	Sara Ali	Biology
5	15	Mohamed Khaled	Computer Science

9-Select students who enrolled in a specific course:

SQLQuery3.sql - (l...IR58LH\mmheg (59))*

```sql
--Identify all students who have enrolled in a particular course, for example, to notify them of updates.
SELECT
    s.student_id,
    s.first_name,
    s.last_name
FROM
    dbo.Students s
    JOIN dbo.Enrollments e ON s.student_id = e.student_id
WHERE
    e.course_id = (SELECT course_id FROM dbo.Courses WHERE course_name = 'Biology');
```

100 %

Results | Messages

	student_id	first_name	last_name
1	6	Yasmin	Ibrahim
2	1	Ahmed	Mostafa
3	4	Sara	Ali

10-Select courses with more than 5 students:

```sql
--Identify popular courses with high enrollment, possibly to allocate additional resources or instructors.
SELECT
    c.course_name,
    COUNT(e.student_id) AS student_count
FROM
    dbo.Courses c
    JOIN dbo.Enrollments e ON c.course_id = e.course_id
GROUP BY
    c.course_name
HAVING
    COUNT(e.student_id) > 5;
```

Results | Messages

course_name	student_count

11-Update a student's email:

```sql
--Update a student's email address, typically after receiving a request for a correction or update.
UPDATE dbo.Students
SET email = 'Ahmed.moustafa@example.com'
WHERE student_id = 1;
```

Messages

```
(1 row affected)

Completion time: 2024-09-01T20:33:11.8361548+03:00
```

12-Delete a course that no students are enrolled in:

```sql
--Clean up the course catalog by removing courses that have no active enrollments.
DELETE FROM dbo.Courses
WHERE course_id NOT IN (SELECT DISTINCT course_id FROM dbo.Enrollments);
```

100 %

Messages

```
(0 rows affected)

Completion time: 2024-09-01T20:34:16.0616398+03:00
```

13-Calculate the average age of students:

```sql
-- Determine the average age of students, possibly for demographic analysis.
SELECT
    AVG(DATEDIFF(year, date_of_birth, GETDATE())) AS average_age
FROM
    dbo.Students;
```

100 %

Results | Messages

	average_age
1	21

14-Find the course with the maximum enrollments:

```sql
--Identify the most popular course, which might require additional resources or sections.
SELECT TOP 1
    c.course_name,
    COUNT(e.student_id) AS enrollment_count
FROM
    dbo.Courses c
    JOIN dbo.Enrollments e ON c.course_id = e.course_id
GROUP BY
    c.course_name
ORDER BY
    enrollment_count DESC;
```

100 %

Results | Messages

	course_name	enrollment_count
1	Computer Science	3

15-List courses along with the number of students enrolled (use GROUP BY):

```sql
--Generate a report showing how many students are enrolled in each course, useful for capacity planning.
SELECT
    c.course_name,
    COUNT(e.student_id) AS student_count
FROM
    dbo.Courses c
    LEFT JOIN dbo.Enrollments e ON c.course_id = e.course_id
GROUP BY
    c.course_name;
```

100 %

Results | Messages

	course_name	student_count
1	Biology	3
2	Computer Science	3
3	History	3
4	Mathematics	3
5	Physics	3

16-Select all students with their enrolled courses (use JOIN):

```sql
--Create a list that shows all students and the courses they are enrolled in, useful for generating student transcripts.
SELECT
    s.student_id,
    s.first_name,
    s.last_name,
    c.course_name
FROM
    dbo.Students s
    LEFT JOIN dbo.Enrollments e ON s.student_id = e.student_id
    LEFT JOIN dbo.Courses c ON e.course_id = c.course_id;
```

100 %

Results | Messages

	student_id	first_name	last_name	course_name
1	1	Ahmed	Mostafa	Mathematics
2	1	Ahmed	Mostafa	Biology
3	2	Fatma	Hassan	Mathematics
4	2	Fatma	Hassan	Physics
5	3	Omar	El-Sayed	Computer Science
6	3	Omar	El-Sayed	History
7	4	Sara	Ali	Physics
8	4	Sara	Ali	Biology
9	5	Mohamed	Khaled	History
10	5	Mohamed	Khaled	Computer Science
11	6	Yasmin	Ibrahim	Biology
12	7	Khaled	Abdelrahman	Mathematics
13	8	Aya	Gamal	Computer Science
14	9	Hassan	Mahmoud	Physics
15	10	Mona	Tarek	History

17-List all instructors and their courses:

```sql
--Generate a report showing which instructors are teaching which courses, useful for scheduling.
SELECT
    i.instructor_id,
    i.first_name,
    i.last_name,
    c.course_name
FROM
    dbo.Instructors i
    JOIN dbo.Courses c ON i.instructor_id = c.instructor_id;
```

%

Results | Messages

instructor_id	first_name	last_name	course_name
1	Ahmed	Ezab	Mathematics
2	Mona	Shawky	Computer Science
3	Tarek	Adel	Physics
1	Ahmed	Ezab	History
2	Mona	Shawky	Biology

18- Find students who are not enrolled in any course:

```sql
--Identify students who have not yet enrolled in any courses, possibly for targeted enrollment campaigns.
SELECT
    s.student_id,
    s.first_name,
    s.last_name
FROM
    dbo.Students s
WHERE
    s.student_id NOT IN (SELECT DISTINCT student_id FROM dbo.Enrollments);
```

100 %

Results | Messages

student_id	first_name	last_name

19-Select students enrolled in more than one course:

```sql
--Identify students who are enrolled in multiple courses, useful for analyzing course load or advising.
SELECT
    s.student_id,
    s.first_name,
    s.last_name
FROM
    dbo.Students s
WHERE
    (SELECT COUNT(*) FROM dbo.Enrollments e WHERE e.student_id = s.student_id) > 1;
```

100 %

Results | Messages

	student_id	first_name	last_name
1	1	Ahmed	Mostafa
2	2	Fatma	Hassan
3	3	Omar	El-Sayed
4	4	Sara	Ali
5	5	Mohamed	Khaled

20-Find courses taught by a specific instructor:

```sql
-- List all courses taught by a specific instructor, useful for planning and workload balancing.
SELECT
    c.course_name
FROM
    dbo.Courses c
WHERE
    c.instructor_id = (SELECT instructor_id FROM dbo.Instructors WHERE last_name = 'Ezab');
```

00 %

Results | Messages

	course_name
1	Mathematics
2	History

21-Select the top 3 students with the most enrollments:

```sql
--Identify the top 3 students with the highest course load, possibly for recognition or advising.
SELECT TOP 3
    s.student_id,
    s.first_name,
    s.last_name,
    COUNT(e.course_id) AS enrollment_count
FROM
    dbo.Students s
    JOIN dbo.Enrollments e ON s.student_id = e.student_id
GROUP BY
    s.student_id, s.first_name, s.last_name
ORDER BY
    enrollment_count DESC;
```

) %

Results | Messages

	student_id	first_name	last_name	enrollment_count
	4	Sara	Ali	2
	3	Omar	El-Sayed	2
	2	Fatma	Hassan	2

22-Use UNION to combine results of two different SELECT queries:

```sql
-- Combine and display results from different queries, such as listing all students and instructors in one result set.
SELECT first_name, last_name, 'Student' AS role FROM dbo.Students
UNION
SELECT first_name, last_name, 'Instructor' AS role FROM dbo.Instructors;
```

100 %

Results | Messages

	first_name	last_name	role
1	Ahmed	Ezab	Instructor
2	Ahmed	Mostafa	Student
3	Aya	Gamal	Student
4	Fatma	Hassan	Student
5	Hassan	Mahmoud	Student
6	Khaled	Abdelrahman	Student
7	Mohamed	Khaled	Student
8	Mona	Shawky	Instructor
9	Mona	Tarek	Student
10	Omar	El-Sayed	Student

23-Create a stored procedure to add a new student:

```sql
--Automate the process of adding a new student to the database, ensuring consistency and reducing manual input.
CREATE PROCEDURE AddStudent
@FirstName VARCHAR(50),
@LastName VARCHAR(50),
@Email VARCHAR(100),
@DateOfBirth DATE
AS
BEGIN
    INSERT INTO dbo.Students (first_name, last_name, email, date_of_birth)
    VALUES (@FirstName, @LastName, @Email, @DateOfBirth);
END;
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2024-09-02T01:28:22.0398204+03:00
```

100 %

24-Create a function to calculate the age of a student based on their date of birth:

```sql
-- Calculate the current age of a student, which can be used in various reports or decision-making processes.
CREATE FUNCTION CalculateAge (@DateOfBirth DATE)
RETURNS INT
AS
BEGIN
    RETURN DATEDIFF(year, @DateOfBirth, GETDATE());
END;
```

Messages

```
Commands completed successfully.

Completion time: 2024-09-02T01:30:09.9537274+03:00
```

25-Calculate the total number of students:

```sql
-- Get the total number of students enrolled, which can be useful for capacity planning and reporting.
SELECT COUNT(*) AS total_students FROM dbo.Students;
```

Results | Messages

	total_students
1	10

26:Calculate the average, minimum, and maximum number of enrollments per course:

```sql
-- Analyze the enrollment patterns across courses to determine course popularity and inform scheduling decisions.
SELECT
    AVG(student_count) AS average_enrollments,
    MIN(student_count) AS min_enrollments,
    MAX(student_count) AS max_enrollments
FROM
(
    SELECT COUNT(e.student_id) AS student_count
    FROM dbo.Courses c
    LEFT JOIN dbo.Enrollments e ON c.course_id = e.course_id
    GROUP BY c.course_id
) AS enrollment_stats;
```

100 %

Results | Messages

	average_enrollments	min_enrollments	max_enrollments
1	3	3	3

27-Use CASE to categorize students based on their age:

```sql
-- Categorize students into different age groups, which can be useful for targeted marketing or educational programs.
SELECT
    first_name,
    last_name,
    CASE
        WHEN DATEDIFF(year, date_of_birth, GETDATE()) < 18 THEN 'Under 18'
        WHEN DATEDIFF(year, date_of_birth, GETDATE()) BETWEEN 18 AND 25 THEN '18-25'
        ELSE '26 and older'
    END AS age_group
FROM
    dbo.Students;
```

Results | Messages

	first_name	last_name	age_group
1	Ahmed	Mostafa	18-25
2	Fatma	Hassan	18-25
3	Omar	El-Sayed	18-25
4	Sara	Ali	18-25
5	Mohamed	Khaled	18-25
6	Yasmin	Ibrahim	18-25
7	Khaled	Abdelrahman	18-25
8	Aya	Gamal	18-25
9	Hassan	Mahmoud	18-25
10	Mona	Tarek	18-25

28-Use EXISTS to find courses with at least one enrolled student:

```sql
--Ensure that courses are only listed if they have active enrollments, useful for determining active course offerings.
SELECT
    course_name
FROM
    dbo.Courses c
WHERE
    EXISTS (SELECT 1 FROM dbo.Enrollments e WHERE e.course_id = c.course_id);
```

Results | Messages

course_name
Mathematics
Computer Science
Physics
History
Biology