

Database project 1

Campus Connection

A College Management Database

Author: Antonios Gerges Nageh

ID: 20221903971

Introduction:

The database is designed to manage student and course information for a university. It includes tables for students, courses, sections, teachers, academic registrations, study timetables, absence reports, questionnaires, programs, results, and emails. The database is designed to support the university's day-to-day operations, such as scheduling courses, registering students for courses, tracking student attendance, and managing academic records.

Entities and Relationships:

The main entities in the database are students, courses, sections, teachers, academic registrations, study timetables, absence reports, questionnaires, programs, results, and emails. These entities are related to each other in various ways, as described below:

A student can have multiple academic registrations. An academic registration belongs to one student.

A student can have multiple study timetables. A study timetable belongs to one student.

A student can have multiple absence reports. An absence report belongs to one student.

A student can have multiple questionnaires. A questionnaire belongs to one student.

A student can have multiple programs. A program belongs to one student.

A course can have multiple sections. A section belongs to one course.

A course can have multiple results. A result belongs to one course.

A teacher can have multiple sections. A section belongs to one teacher.

A teacher can have multiple emails. An email belongs to one teacher.

A student can have multiple emails. An email belongs to one student.

Here's an XML file you can open in this site to see the er diagram clearly

https://drive.google.com/drive/folders/1g8OilCylJX-YGIHejcfL6KwtdEvBwQeb??usp=share_link

and you will open it from this site: diagrams.net

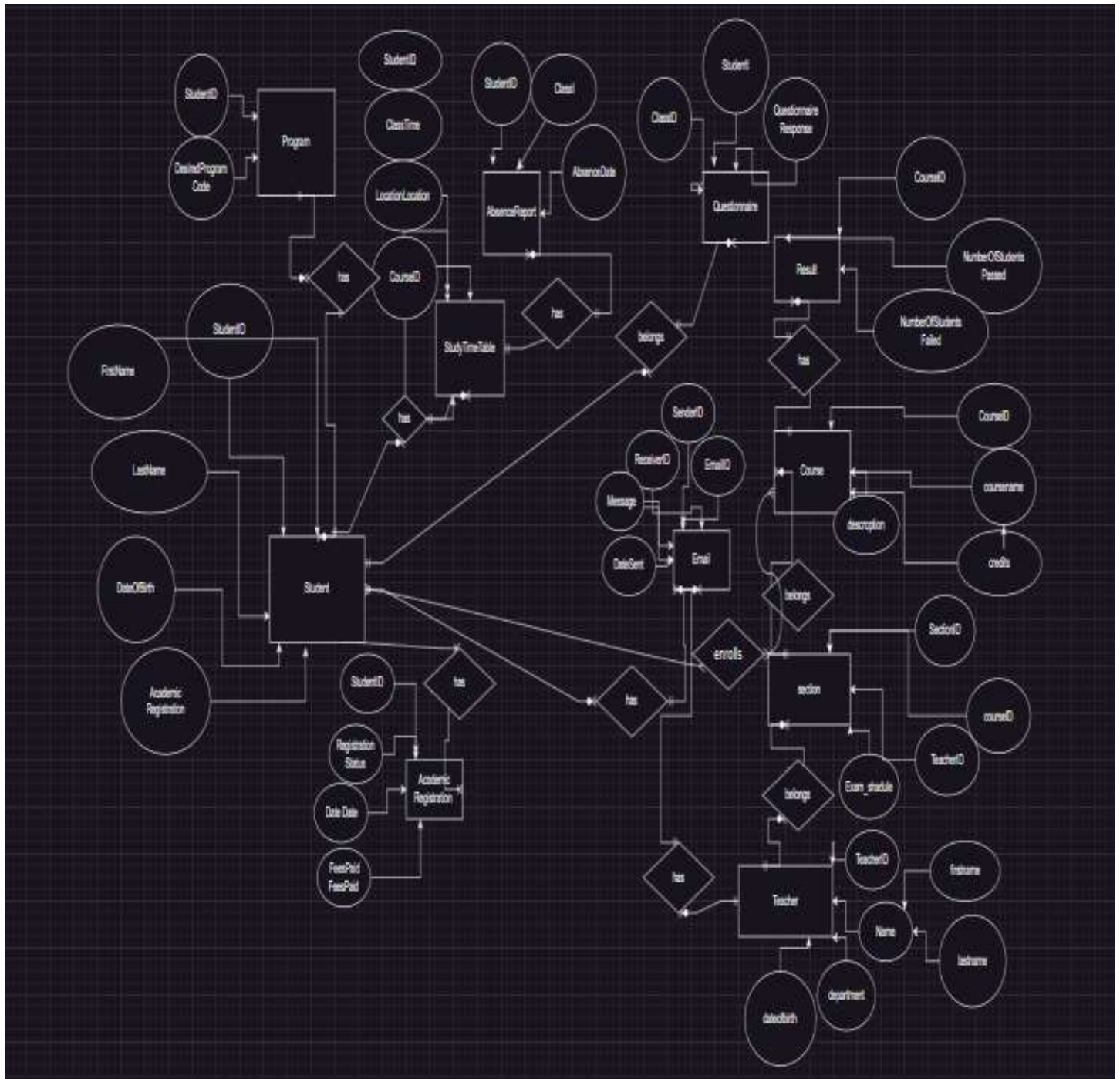
to open the file



1-Click on open Existing diagram.

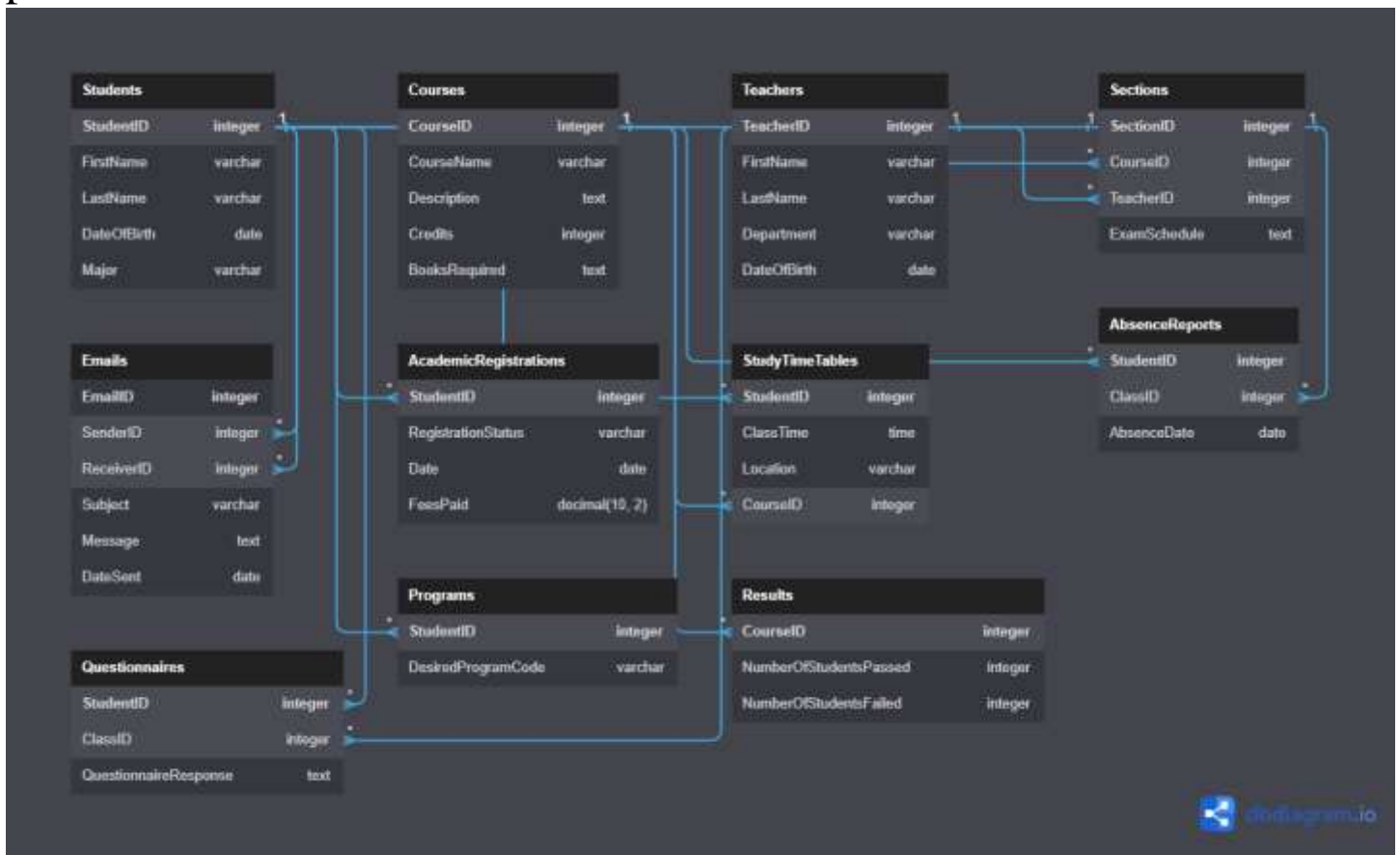
2- Upload the XML file.

here's a screenshot of the ER diagram.



Relational schema

I have created a relational schema using an online tool to look more professional



Also, here's the written schema

Students

- StudentID [PRIMARY KEY]
- FirstName
- LastName
- DateOfBirth
- Major
- AcademicRegistration

Courses

- CourseID [PRIMARY KEY]
- CourseName
- Description
- Credits
- BooksRequired

Teachers

- TeacherID [PRIMARY KEY]
- FirstName
- LastName
- Department
- DateOfBirth

Sections

- SectionID [PRIMARY KEY]
- CourseID [FOREIGN KEY REFERENCES Courses(CourseID)]
- TeacherID [FOREIGN KEY REFERENCES Teachers(TeacherID)]
- ExamSchedule

Emails

- EmailID [PRIMARY KEY]
- SenderID [FOREIGN KEY REFERENCES Students(StudentID)]
- ReceiverID [FOREIGN KEY REFERENCES Students(StudentID)]
- Subject
- Message
- DateSent

AcademicRegistrations

- StudentID [PRIMARY KEY REFERENCES Students(StudentID)]
- RegistrationStatus
- Date
- FeesPaid

StudyTimeTables

- StudentID [FOREIGN KEY REFERENCES Students(StudentID)]
 - ClassTime
 - Location
 - CourseID [FOREIGN KEY REFERENCES Courses(CourseID)]
- [PRIMARY KEY (StudentID, ClassTime)]

AbsenceReports

- StudentID [FOREIGN KEY REFERENCES Students(StudentID)]
 - ClassID [FOREIGN KEY REFERENCES Sections(SectionID)]
 - AbsenceDate
- [PRIMARY KEY (StudentID, ClassID, AbsenceDate)]

Questionnaires

- StudentID [FOREIGN KEY REFERENCES Students(StudentID)]
 - ClassID [FOREIGN KEY REFERENCES Sections(SectionID)]
 - QuestionnaireResponse
- [PRIMARY KEY (StudentID, ClassID)]

Programs

- StudentID [FOREIGN KEY REFERENCES Students(StudentID)]

DesiredProgramCode

Results

CourseID[PRIMARYKEYREFERENCESCourses(CourseID)]

NumberOfStudentsPassed

NumberOfStudentsFailed

Table Descriptions:

The database contains the following tables:

Students: This table stores information about the students, such as their student ID, first name, last name, date of birth, major, and academic registration.

Courses: This table stores information about the courses offered by the university, such as their course ID, course name, description, number of credits, and required books.

Sections: This table stores information about the sections of courses offered by the university, such as their section ID, course ID, teacher ID, and exam schedule.

Teachers: This table stores information about the teachers, such as their teacher ID, first name, last name, department, and date of birth.

Emails: This table stores information about the emails sent and received by the students and teachers, such as the email ID, sender ID, receiver ID, subject, message, and date sent.

AcademicRegistrations: This table stores information about the academic registrations of the students, such as their student ID, registration status, date, and fees paid.

StudyTimeTables: This table stores information about the study timetables of the students, such as their student ID, class time, location, and course ID.

AbsenceReports: This table stores information about the absence reports of the students, such as their student ID, class ID, and absence date.

Questionnaires: This table stores information about the questionnaires filled out by the students, such as their student ID, class ID, and questionnaire response.

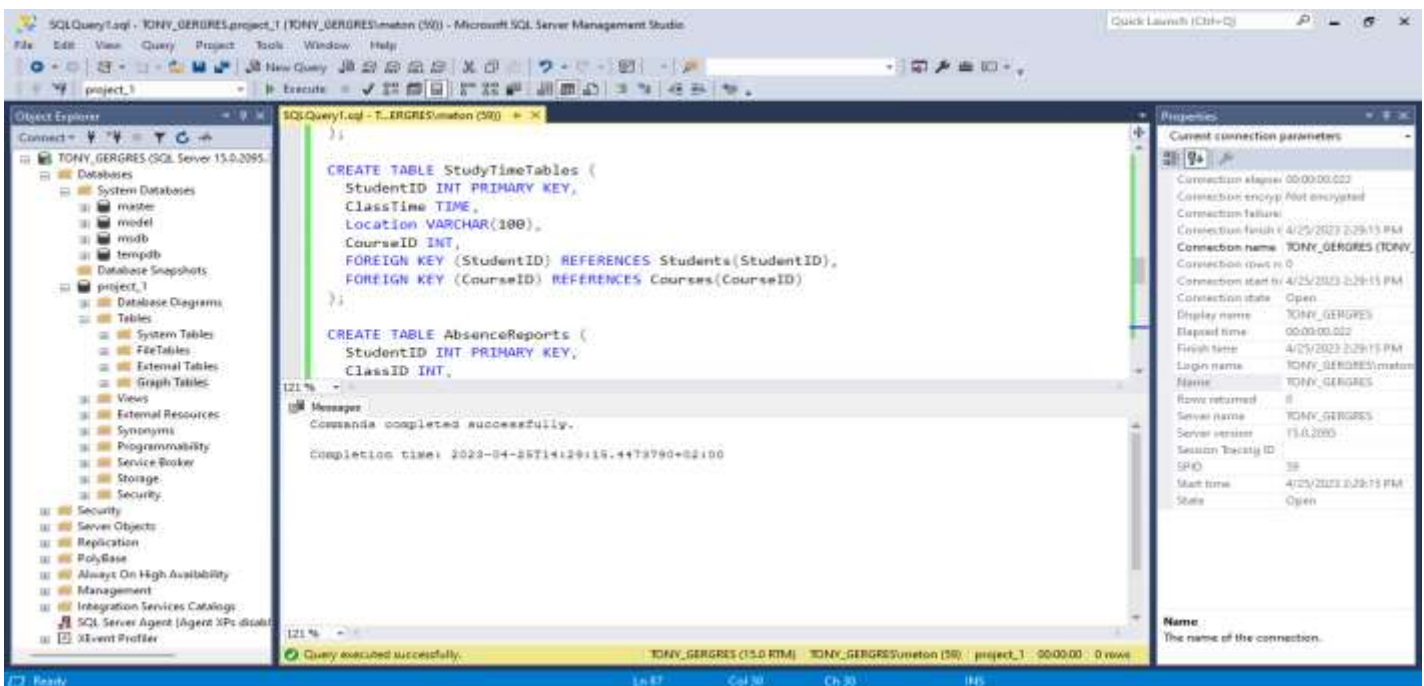
Programs: This table stores information about the programs desired by the students, such as their student ID and desired program code.

Results: This table stores information about the results of the courses, such as their course ID, number of students passed, and number of students failed.

SQL Scripts:

The SQL scripts are used to create the tables and their corresponding columns in the database. Each table has a primary key and foreign keys linking it to other database tables. The SQL scripts can be run on any database management system that supports SQL.

I used a Microsoft SQL server to manage this Database.



Here's the SQL query and the script you will find them all here:

https://drive.google.com/drive/folders/18dv6Gd_BFQw_Gk69mXgGs34MyAiQxxrD?usp=share_link

⇒ You will find 2 queries in the file 1st to create the tables and 2nd to add some samples.

Also here are the 2 queries written:

•First Query

```
CREATE TABLE Students (  
    StudentID INT PRIMARY KEY,  
    FirstName VARCHAR(50),  
    LastName VARCHAR(50),  
    DateOfBirth DATE,  
    Major VARCHAR(50),  
    AcademicRegistration VARCHAR(50));
```

```
CREATE TABLE Courses (  
    CourseID INT PRIMARY KEY,  
    CourseName VARCHAR(100),  
    Description TEXT,  
    Credits INT,  
    BooksRequired TEXT);
```

```
CREATE TABLE Teachers (  
    TeacherID INT PRIMARY KEY,  
    FirstName VARCHAR(50),  
    LastName VARCHAR(50),  
    Department VARCHAR(50),  
    DateOfBirth DATE);
```

```
CREATE TABLE Sections (  
    SectionID INT PRIMARY KEY,  
    CourseID INT,  
    TeacherID INT,  
    ExamSchedule TEXT,  
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID),  
    FOREIGN KEY (TeacherID) REFERENCES Teachers(TeacherID));
```

```
CREATE TABLE Emails (  
    EmailID INT PRIMARY KEY,  
    SenderID INT,
```

```
ReceiverID INT,  
Subject VARCHAR(100),  
Message TEXT,  
DateSent DATE,  
FOREIGN KEY (SenderID) REFERENCES Students(StudentID),  
FOREIGN KEY (ReceiverID) REFERENCES Students(StudentID));
```

```
CREATE TABLE AcademicRegistrations (  
    StudentID INT PRIMARY KEY,  
    RegistrationStatus VARCHAR(50),  
    Date DATE,  
    FeesPaid DECIMAL(10, 2),  
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID));
```

```
CREATE TABLE StudyTimeTables (  
    StudentID INT PRIMARY KEY,  
    ClassTime TIME,  
    Location VARCHAR(100),  
    CourseID INT,  
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),  
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID));
```

```
CREATE TABLE AbsenceReports (  
    StudentID INT PRIMARY KEY,  
    ClassID INT,  
    AbsenceDate DATE,  
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),  
    FOREIGN KEY (ClassID) REFERENCES Sections(SectionID));
```

```
CREATE TABLE Questionnaires (  
    StudentID INT PRIMARY KEY,  
    ClassID INT,  
    QuestionnaireResponse TEXT,  
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
```

```
FOREIGN KEY (ClassID) REFERENCES Sections(SectionID));
```

```
CREATE TABLE Programs (
```

```
StudentID INT PRIMARY KEY,
```

```
DesiredProgramCode VARCHAR(50),
```

```
FOREIGN KEY (StudentID) REFERENCES Students(StudentID));
```

```
CREATE TABLE Results (
```

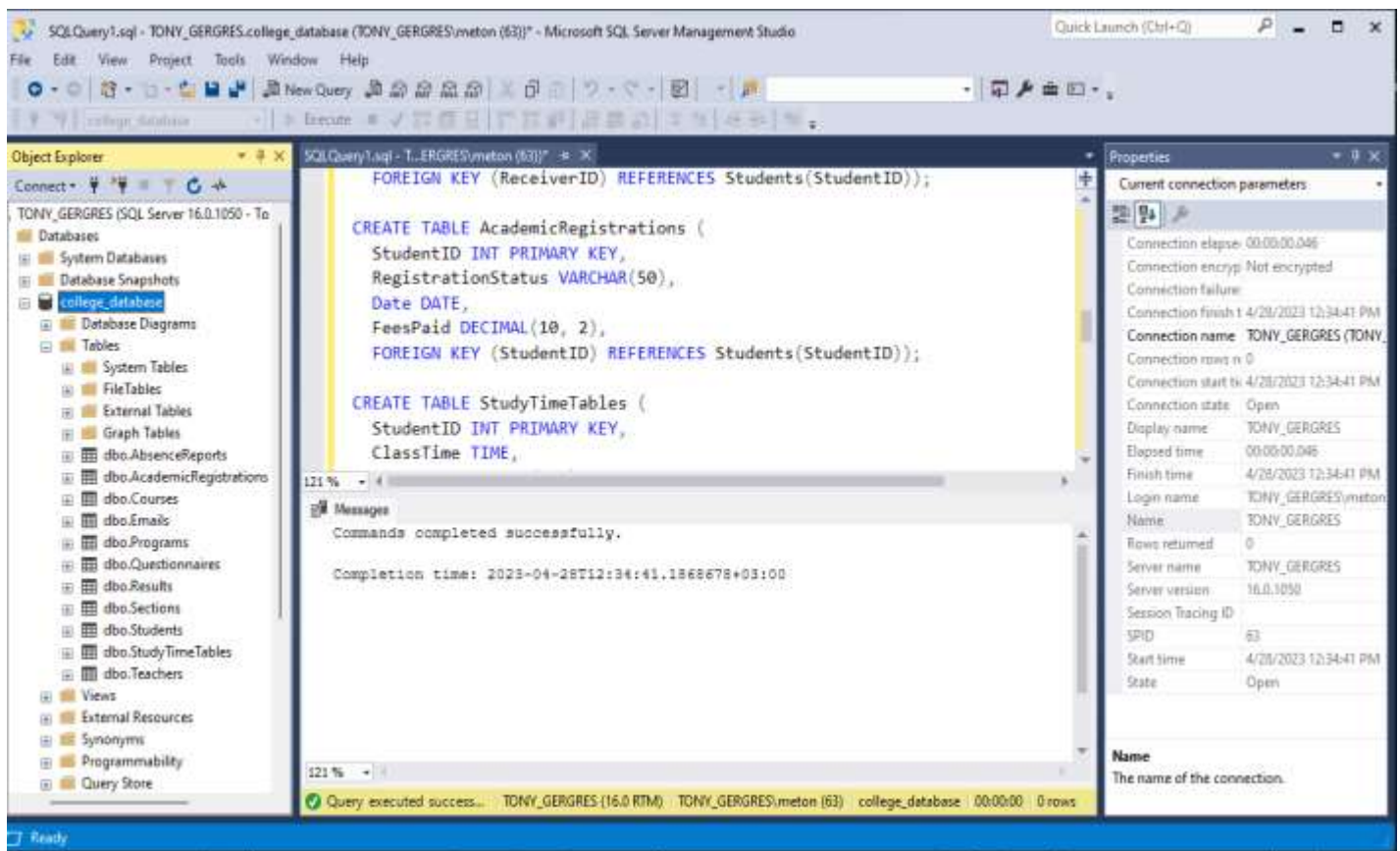
```
CourseID INT PRIMARY KEY,
```

```
NumberOfStudentsPassed INT,
```

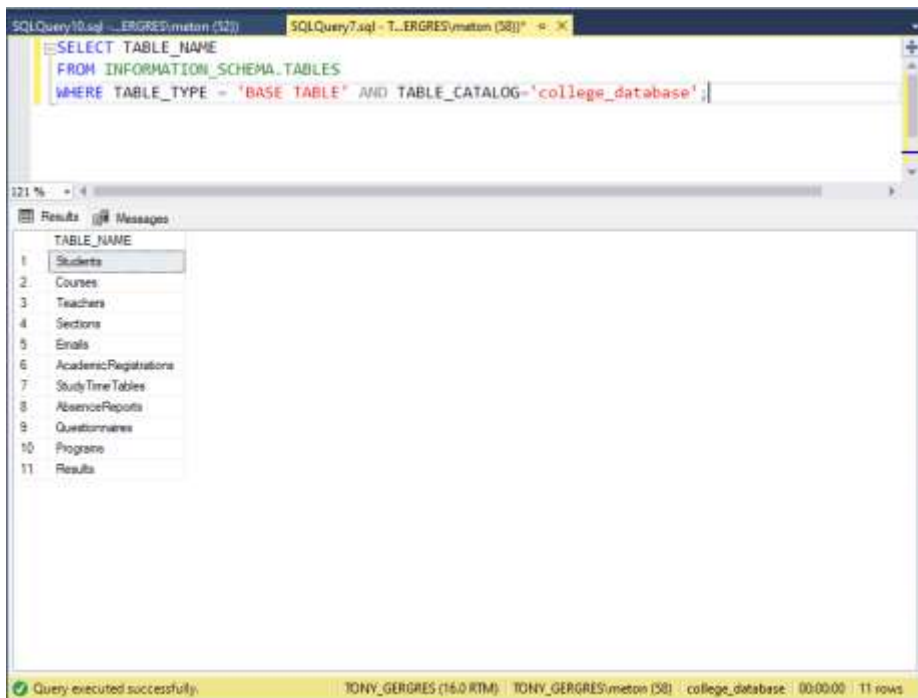
```
NumberOfStudentsFailed INT,
```

```
FOREIGN KEY (CourseID) REFERENCES Courses(CourseID));
```

After executing the query, the tables exist.



Here's a query to show all the tables that I created.



• Adding values query

Note: These data samples were obtained from a free online source and were adapted to fit my database requirements.

```
INSERT INTO Students(StudentID, FirstName, LastName, DateOfBirth, Major,
AcademicRegistration)
```

```
VALUES
```

```
(1, 'John', 'Doe', '1998-05-23', 'intelligent Systems', 'Spring 2023'),
(2, 'Jane', 'Smith', '1999-02-14', 'cyber security', 'Fall 2023'),
(3, 'Mike', 'Johnson', '1997-11-05', 'Media', 'Spring 2023'),
(4, 'Sarah', 'Williams', '2000-08-12', 'health care', 'Fall 2023'),
(5, 'David', 'Lee', '1999-04-01', 'business', 'Spring 2021');
```

```
INSERT INTO Courses (CourseID, CourseName, Description, Credits, BooksRequired)
```

```
VALUES
```

```
(1, 'Introduction to Programming', 'This course introduces students to the basics of
programming using a high-level language such as Python.', 4, 'Programming in Python
by Mark Lutz'),
```

```
(2, 'Calculus I', 'This course covers the fundamentals of calculus including limits,
derivatives, and integrals.', 3, 'Calculus: Early Transcendentals by James Stewart'),
(3, 'probability', 'this course covers all the concepts of probability.', 4,
'probability by Jonathan Claydon'),
(4, 'programming', 'This course covers the basics of programming, bookkeeping, and
accounting systems.', 3, 'programming for everyone'),
(5, 'English Composition', 'This course focuses on writing skills including grammar,
syntax, and composition.', 3, 'The Elements of Style by William Strunk Jr. and E. B.
White);
```

```
INSERT INTO Teachers(TeacherID, FirstName, LastName, Department, DateOfBirth)
VALUES
```

```
(1, 'Mohammed', 'elhabrok', ' Introduction to Programming ', '1980-06-12'),
(2, 'Sarah', 'Smith', 'Mathematics', '1975-12-25'),
(3, 'Marvat', 'Mikhail', 'probability', '1982-03-08'),
(4, 'Emily', 'Davis', 'programming for everyone', '1978-11-14'),
(5, 'Robert', 'Clark', ' Calculus I, '1985-09-01');
```

```
INSERT INTO Sections(SectionID, CourseID, TeacherID, ExamSchedule)
VALUES
```

```
(1, 1, 1, 'May 10th, 2023 9:00am'),
(2, 2, 2, 'December 15th, 2023 1:00pm'),
(3, 3, 3, 'April 25th, 2023 11:00am'),
(4, 4, 4, 'December 10th, 2022 10:00am'),
(5, 5, 5, 'May 5th, 2022 2:00pm');
```

```
INSERT INTO Emails (EmailID, SenderID, ReceiverID, Subject, Message, DateSent)
```

```
VALUES (1, 1, 1, 'Regarding Assignment Submission', 'Dear Sarah, I am writing to
check if you have submitted the assignment. Please let me know if you need any help.
Thanks, John.', '2023-04-20'),
```

```
(2, 2, 2, 'Change in Class Schedule', 'Dear Alex, Due to unforeseen circumstances,
the class on Friday has been rescheduled to Monday at the same time. Please let me
know if you have any questions. Best regards, Sam.', '2023-04-23'),
```

```
(3, 3, 3, 'Invitation to Join the Club', 'Dear Mark, I hope you are doing well. I am
writing to invite you to join our club. We have a lot of exciting activities planned
for this semester. Please let me know if you are interested. Best regards, Alice.',
'2023-04-22'),
```

```
(4, 4, 4, 'Regarding Exam Preparation', 'Dear Tom, I hope you are studying hard for
the exam. Please let me know if you need any study materials or have any questions.
Good luck! Best regards, John.', '2023-04-19'),
```

```
(5, 5, 5, 'Regarding the Project Presentation', 'Dear Sarah, I am writing to discuss
the project presentation. Can we meet sometime next week to discuss our progress?
Best regards, John.', '2023-04-24');
```

--Insert 5 sample records into the AcademicRegistrations table:

```
INSERT INTO AcademicRegistrations (StudentID, RegistrationStatus, Date, FeesPaid)
VALUES (1, 'Registered', '2023-04-01', 5000.00),
(2, 'Registered', '2023-04-02', 5500.00),
(3, 'Registered', '2023-04-03', 6000.00),
(4, 'Registered', '2023-04-04', 4500.00),
(5, 'Registered', '2023-04-05', 4000.00);
```

--Insert 5 sample records into the StudyTimeTables table:

```
INSERT INTO StudyTimeTables (StudentID, ClassTime, Location, CourseID)
VALUES (5, '08:00:00', 'Room 101', 1),
(1, '09:00:00', 'Room 102', 2),
(2, '10:00:00', 'Room 103', 3),
(3, '11:00:00', 'Room 104', 4),
(4, '12:00:00', 'Room 105', 5);
```

--Insert 5 sample records into the AbsenceReports table:

```
INSERT INTO AbsenceReports (StudentID, ClassID, AbsenceDate)
VALUES (1, 1, '2023-04-10'),
(2, 2, '2023-04-11'),
(3, 3, '2023-04-12'),
(4, 4, '2023-04-13'),
(5, 5, '2023-04-14');
```

```
INSERT INTO Questionnaires (StudentID, ClassID, QuestionnaireResponse)
VALUES
(1, 1, 'The class was well organized and informative.'),
(2, 2, 'The professor was very engaging and helpful.'),
(3, 1, 'I found the course material to be challenging but rewarding.'),
(4, 3, 'I struggled to keep up with the pace of the class.'),
(5, 2, 'I enjoyed the group projects and collaboration in this class.');
```

```
INSERT INTO Programs (StudentID, DesiredProgramCode)
```

VALUES

```
(1, 'intelligent Systems'),  
(2, 'business'),  
(3, 'health care'),  
(4, 'cyber security'),  
(5, 'Media');
```

INSERT INTO Results (CourseID, NumberOfStudentsPassed, NumberOfStudentsFailed)

VALUES

```
(1, 20, 5),  
(2, 15, 10),  
(3, 25, 3),  
(4, 18, 7),  
(5, 22, 4);
```

Here's to show the tables after adding the values:

```
SELECT * FROM Students;  
SELECT * FROM Courses;  
SELECT * FROM Teachers;  
SELECT * FROM Sections;  
SELECT * FROM Emails;  
SELECT * FROM AcademicRegistrations;  
SELECT * FROM StudyTimeTables;  
SELECT * FROM AbsenceReports;  
SELECT * FROM Questionnaires;  
SELECT * FROM Programs;  
SELECT * FROM Results;
```

	StudentID	FirstName	LastName	DateOfBirth	Major	AcademicRegistration	
1	1	John	Doe	1998-05-23	intelligent Systems	Spring 2023	
2	2	Jane	Smith	1999-02-14	cyber security	Fall 2023	
3	3	Mike	Johnson	1997-11-05	Media	Spring 2023	
4	4	Sarah	Williams	2000-08-12	health care	Fall 2023	
5	5	David	Lee	1999-04-01	business	Spring 2021	

	CourseID	CourseName	Description	Credits	BooksRequired
1	1	Introduction to Programming	This course introduces students to the basics of ...	4	Programming in Python by Mark Lutz
2	2	Calculus I	This course covers the fundamentals of calculus ...	3	Calculus: Early Transcendentals by James Stewart
3	3	probability	this course covers all the concepts of probability.	4	probability by Jonathan Claydon
4	4	programming	This course covers the basics of programming, b...	3	programming for everyone
5	5	English Composition	This course focuses on writing skills including gra...	3	The Elements of Style by William Strunk Jr. and ...

	TeacherID	FirstName	LastName	Department	DateOfBirth
1	1	Mohammed	elhabrok	Introduction to Programming	1980-06-12
2	2	Sarah	Smith	Mathematics	1975-12-25
3	3	Marvat	Mikhail	probability	1982-03-08
4	4	Emily	Davis	programming for everyone	1978-11-14
5	5	Robert	Clark	Calculus I	1985-09-01

	SectionID	CourseID	TeacherID	ExamSchedule
1	1	1	1	May 10th, 2023 9:00am
2	2	2	2	December 15th, 2023 1:00pm
3	3	3	3	April 25th, 2023 11:00am
4	4	4	4	December 10th, 2022 10:00...
5	5	5	5	May 5th, 2022 2:00pm

	EmailID	SenderID	ReceiverID	Subject	Message	DateSent
1	1	1	1	Regarding Assignment Submission	Dear Sarah, I am writing to check if you have sub...	2023-04-20
2	2	2	2	Change in Class Schedule	Dear Alex, Due to unforeseen circumstances, the ...	2023-04-23
3	3	3	3	Invitation to Join the Club	Dear Mark, I hope you are doing well. I am writing...	2023-04-22
4	4	4	4	Regarding Exam Preparation	Dear Tom, I hope you are studying hard for the ex...	2023-04-19
5	5	5	5	Regarding the Project Presentati...	Dear Sarah, I am writing to discuss the project pre...	2023-04-24

	StudentID	RegistrationStatus	Date	FeesPaid
1	1	Registered	2023-04-01	5000.00
2	2	Registered	2023-04-02	5500.00
3	3	Registered	2023-04-03	6000.00
4	4	Registered	2023-04-04	4500.00
5	5	Registered	2023-04-05	4000.00

	StudentID	ClassTime	Location	CourseID
1	1	09:00:00.0000000	Room 102	2
2	2	10:00:00.0000000	Room 103	3
3	3	11:00:00.0000000	Room 104	4
4	4	12:00:00.0000000	Room 105	5
5	5	08:00:00.0000000	Room 101	1

	StudentID	ClassID	AbsenceDate
1	1	1	2023-04-10
2	2	2	2023-04-11
3	3	3	2023-04-12
4	4	4	2023-04-13
5	5	5	2023-04-14

	StudentID	ClassID	QuestionnaireResponse
1	1	1	The class was well organized and informative.
2	2	2	The professor was very engaging and helpful.
3	3	1	I found the course material to be challengin...
4	4	3	I struggled to keep up with the pace of the ...
5	5	2	I enjoyed the group projects and collaborati...

	StudentID	DesiredProgramCode
1	1	intelligent Systems
2	2	business
3	3	health care
4	4	cyber security
5	5	Media

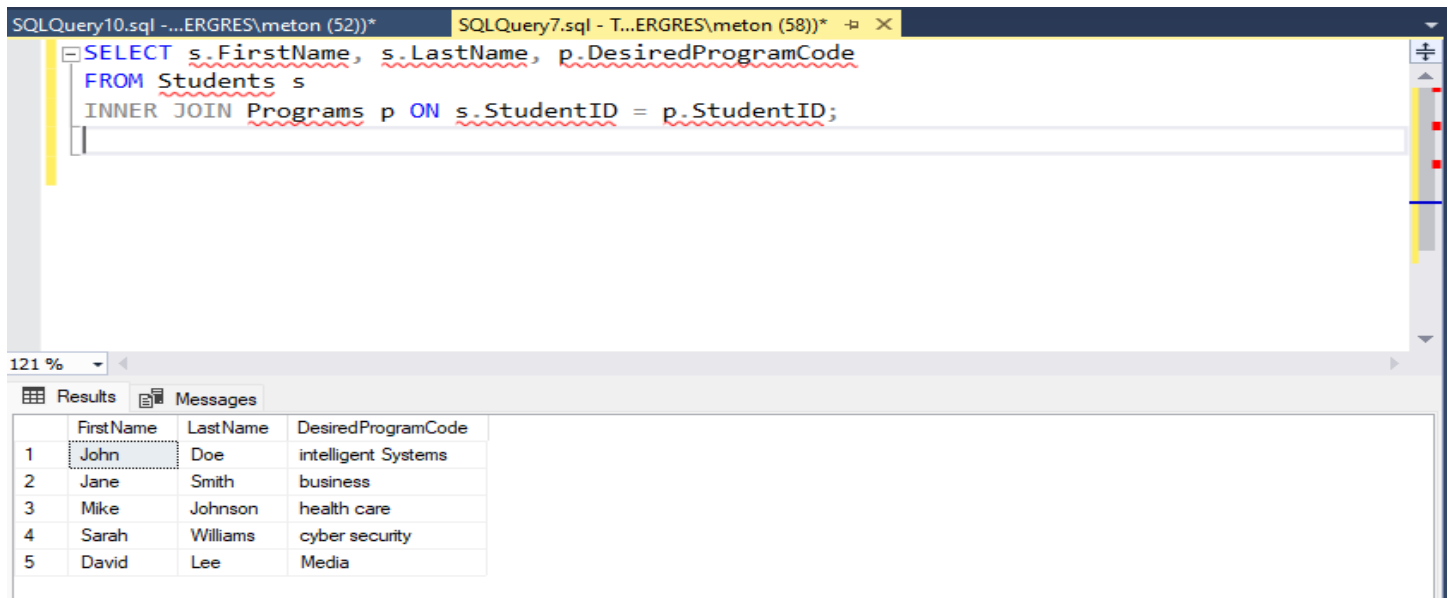
	CourseID	NumberOfStudentsPassed	NumberOfStudentsFailed
1	1	20	5
2	2	15	10
3	3	25	3
4	4	18	7
5	5	22	4

Sample SQL Queries for Common Database Operations

Query #1 - Displaying student information along with their program codes:

```
SELECT s.FirstName, s.LastName, p.DesiredProgramCode
FROM Students s
INNER JOIN Programs p ON s.StudentID = p.StudentID;
```

Explanation: This query uses an INNER JOIN to combine records from the Students table and the Programs table based on the common field StudentID. The resulting output will show each student's first name, last name, and desired program code.



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the SQL query for Query #1. The bottom pane shows the results of the query execution.

SQL Query 10.sql - ...\ERGRES\meton (52))*

```
SELECT s.FirstName, s.LastName, p.DesiredProgramCode
FROM Students s
INNER JOIN Programs p ON s.StudentID = p.StudentID;
```

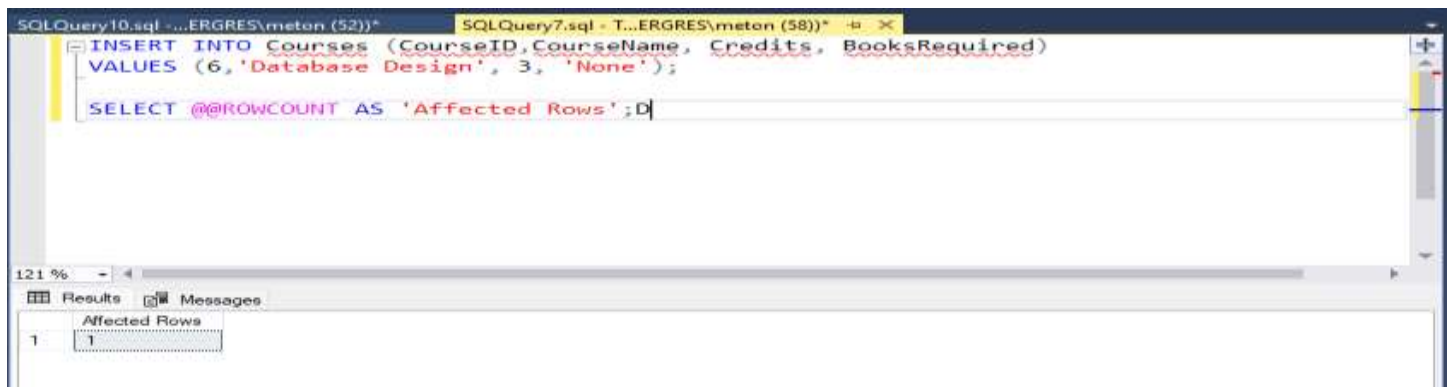
Results

	FirstName	LastName	DesiredProgramCode
1	John	Doe	intelligent Systems
2	Jane	Smith	business
3	Mike	Johnson	health care
4	Sarah	Williams	cyber security
5	David	Lee	Media

Query #2 - Adding a new course to the database:

```
INSERT INTO Courses (course_ID, CourseName, Credits, BooksRequired) VALUES
(6, 'Database Design',
3, 'None');
```

Explanation: This is an INSERT statement used to add a new record to the Courses table. It inserts a new course called "Database Design" which has no books required and awards 3 credits to students who take it.



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the SQL query for Query #2. The bottom pane shows the results of the query execution.

SQL Query 10.sql - ...\ERGRES\meton (52))*

```
INSERT INTO Courses (CourseID, CourseName, Credits, BooksRequired)
VALUES (6, 'Database Design', 3, 'None');
SELECT @@ROWCOUNT AS 'Affected Rows';
```

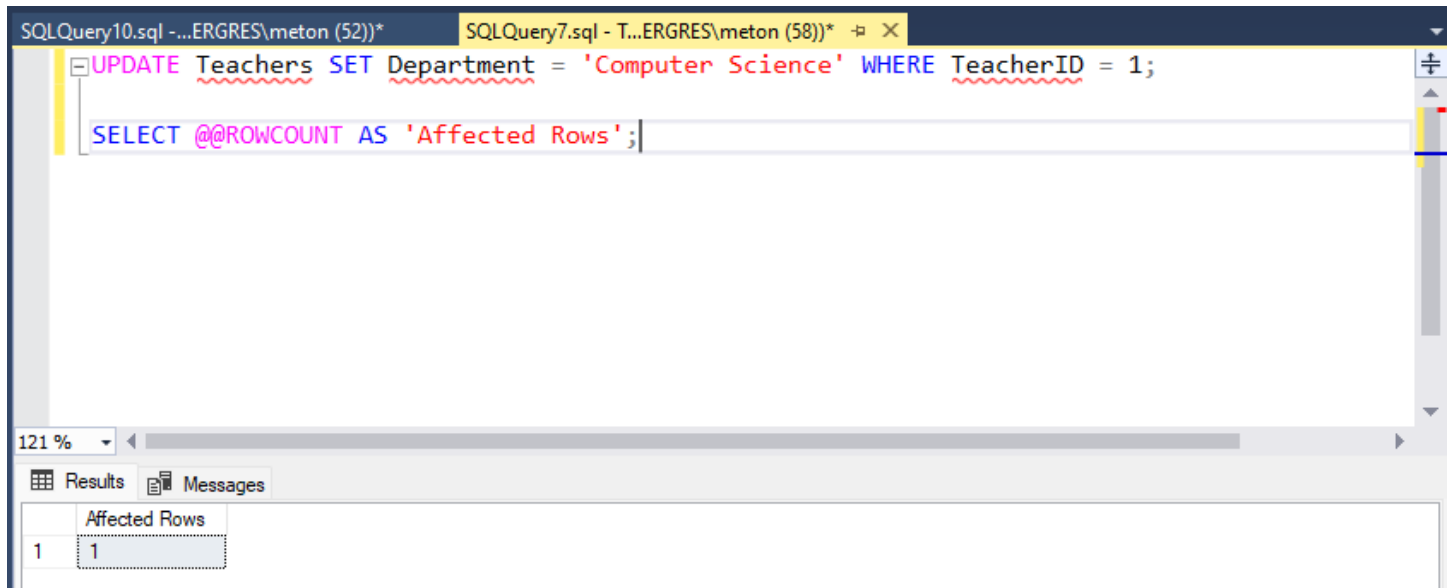
Results

Affected Rows
1

Query #3 - Updating a teacher's department in the system:

```
UPDATE Teachers SET Department = 'Computer Science' WHERE TeacherID = 1;
```

Explanation: This is an UPDATE statement used to modify an existing record in the Teachers table. It updates the department of the teacher whose ID is equal to 1 to "Computer Science".



Query #4 - Retrieving all email messages sent by teachers:

```
SELECT e.Subject, e.Message, t.Department  
FROM Emails e  
INNER JOIN Teachers t ON e.SenderID = t.TeacherID
```

Explanation: This SQL query retrieves data from two tables, Emails and Teachers, using an inner join on the SenderID column in Emails and the TeacherID column in Teachers. The result set includes three columns: Subject and Message columns from the Emails table, and Department column from the Teachers table. The output will show the subject and message of all emails sent by teachers, along with their corresponding department.

The INNER JOIN clause is used to combine data from both tables based on a matching condition. In this case, the matching condition is that the value in the SenderID column of the Emails table must match the value in the TeacherID column of the Teachers table.

The result of the JOIN operation is a new table that contains columns from both tables (Emails and Teachers) where the matching condition is true.

SQLQuery10.sql - ...ERGRES\meton (52))* SQLQuery7.sql - T...ERGRES\meton (58))*

```
SELECT e.Subject, e.Message, t.Department
FROM Emails e
INNER JOIN Teachers t ON e.SenderID = t.TeacherID
```

121 %

Results Messages

	Subject	Message	Department
1	Regarding Assignment Submission	Dear Sarah, I am writing to check if you have sub...	Computer Science
2	Change in Class Schedule	Dear Alex, Due to unforeseen circumstances, the ...	Mathematics
3	Invitation to Join the Club	Dear Mark, I hope you are doing well. I am writing...	probability
4	Regarding Exam Preparation	Dear Tom, I hope you are studying hard for the ex...	programming for everyone
5	Regarding the Project Presentation	Dear Sarah, I am writing to discuss the project pre...	Calculus I

Query #5 - Deleting a section from the system if its exam schedule is missing:

```
DELETE FROM Sections WHERE ExamSchedule IS NULL.
```

Explanation: This query deletes all sections from the Sections table that lack an entry in the corresponding Exams table indicating the exam schedule, as determined by checking for NULL values in the Exam Schedule column.

SQLQuery10.sql - ...ERGRES\meton (52))* SQLQuery7.sql - T...ERGRES\meton (58))*

```
DELETE FROM Sections WHERE ExamSchedule IS NULL
```

121 %

Messages

(0 rows affected)

Completion time: 2023-04-28T13:24:30.6294722+03:00

Conclusion:

The database is designed to manage the student and course information for a university. It contains tables for students, courses, sections, teachers, academic registrations, study timetables, and absences.

Thanks