LAB 7

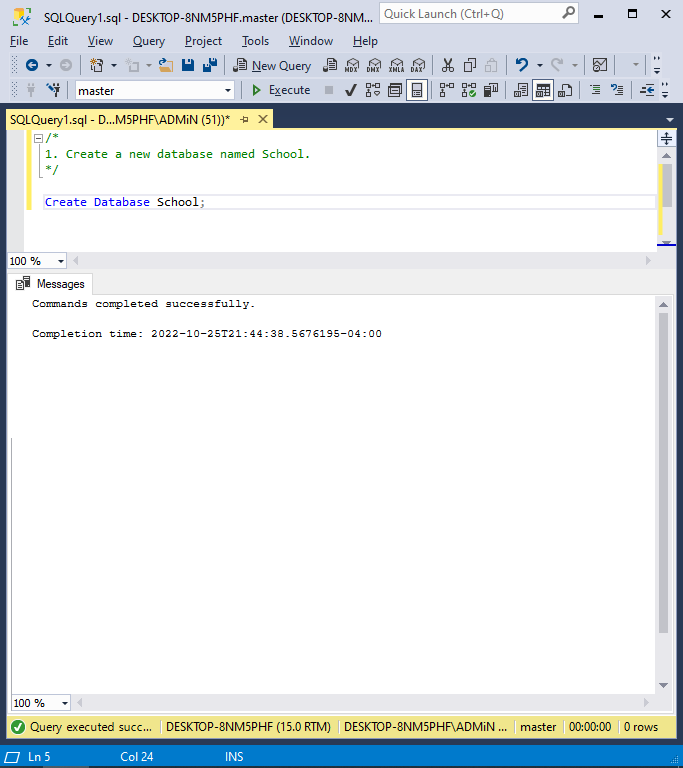
**1.** Create a new database named School.

**Comments:**

* A new table named School is created using CREATE DATABASE statement.

**Query:** Create Database School;

**Output:**

****

**2. (1)**Describe the relationship type shown in figure (one-to-one, one-to-many or many-to-many).

**(2)** Write the CREATE TABLE statements needed to implement the following design in the School database. Include foreign key constraints.

Define StudentID and CourseID as Identity columns. Decide which columns should allow null values, if any, and explain your decision.Define the Course Price column with a defaul zero and a check constraint to allow only positive values.

**Comments:**

* One-to-many relationship exists between Student, Courses and StudyGroup Table as one student can have multiple studygroups and one course can have multiple study groups.
* School Database is used to create three tables namely Students, Courses, and StudyGroups using CREATE DATABASE statement.
* The requirement of the StudentID and CourseID as identity columns has been fulfilled and foreign key constraints has also be defined.
* Null values can be contained by the Phone column of Students table and CoursePrice column present in the Courses table.
* All columns in the table Courses must not be NULL, and the CoursePrice column defaults to 0 with a check constraint that only permits positive values.

**Query:**

* Use School;

/\* CREATE STUDENTS TABLE \*/

Create Table Students (StudentID int NOT NULL IDENTITY PRIMARY KEY,

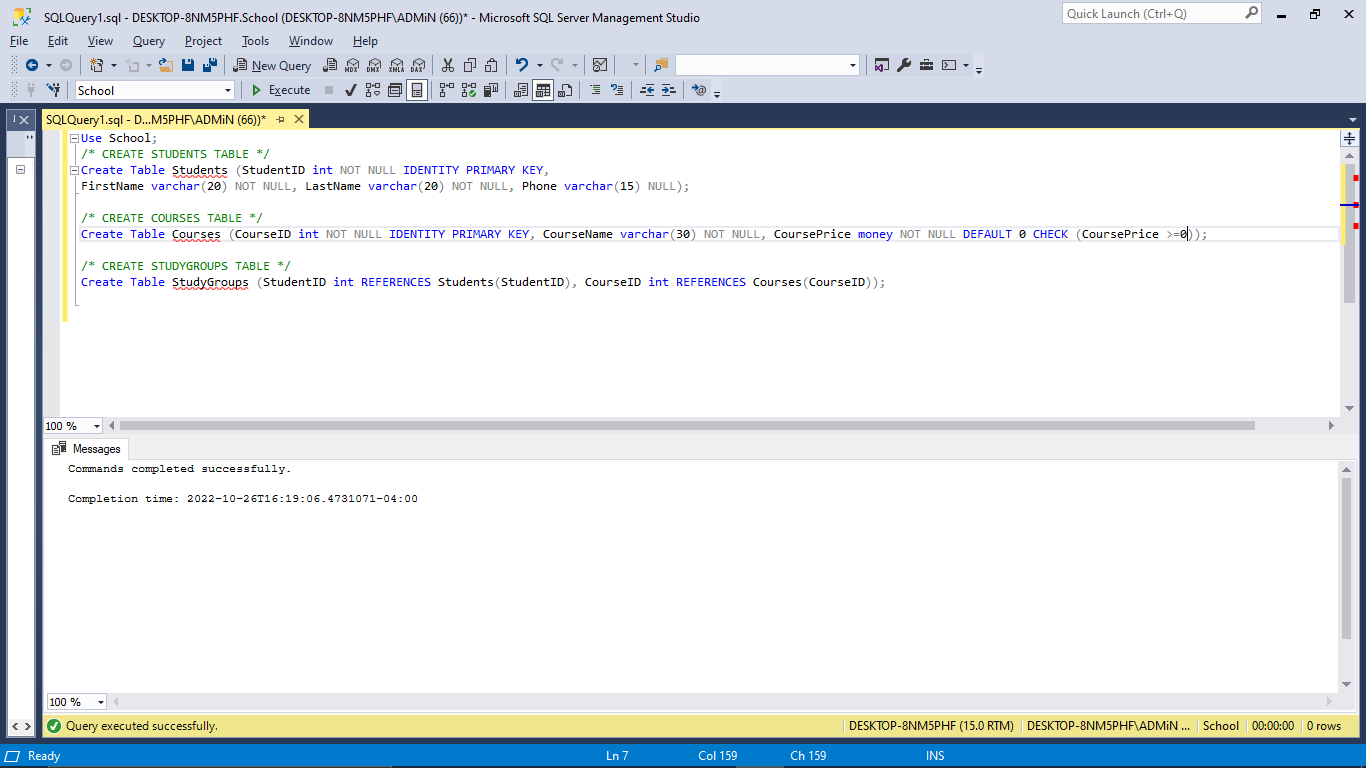
FirstName varchar(20) NOT NULL, LastName varchar(20) NOT NULL, Phone varchar(15) NULL);

/\* CREATE COURSES TABLE \*/

Create Table Courses (CourseID int NOT NULL IDENTITY PRIMARY KEY, CourseName varchar(30) NOT NULL, CoursePrice money NOT NULL DEFAULT 0 CHECK (CoursePrice >=0));

/\* CREATE STUDYGROUPS TABLE \*/

Create Table StudyGroups (StudentID int REFERENCES Students(StudentID), CourseID int REFERENCES Courses(CourseID));



3. Write the CREATE INDEX statements to create a clustered index on the StudentID column and a nonclustered index on the CourseID column of the StudyGroups table.

**Comments:**

* CREATE INDEX statement is used to construct a clustered index on the StudentID column of the StudyGroups table and a nonclustered index on the CourseID column.

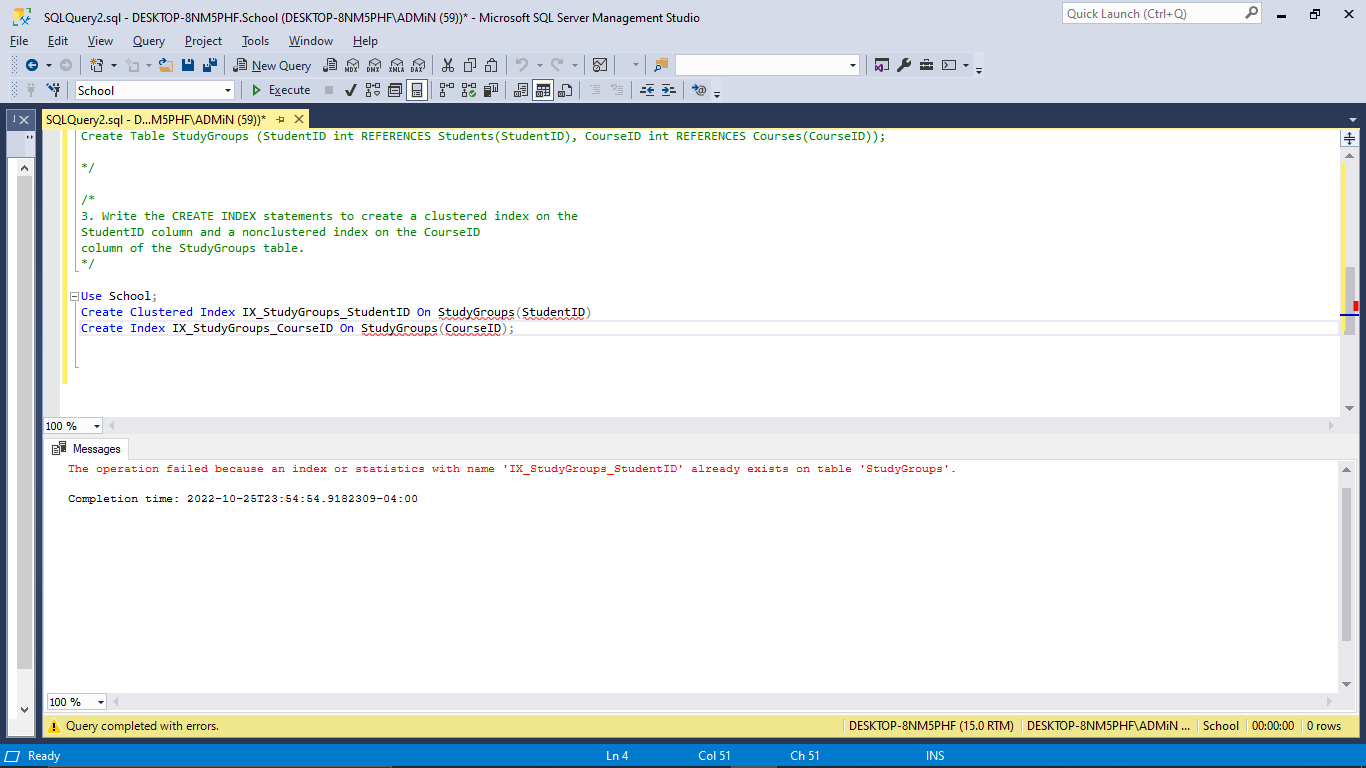
**Query:**

Use School;

Create Clustered Index IX\_StudyGroups\_StudentID On StudyGroups(StudentID)

Create Index IX\_StudyGroups\_CourseID On StudyGroups(CourseID);

**Output:**



4. Write an ALTER TABLE statement that adds a new column, CourseFeePaid, to the Students table. Use the bit data type, disallow null values, and assign a default Boolean value of False.

**Comments:**

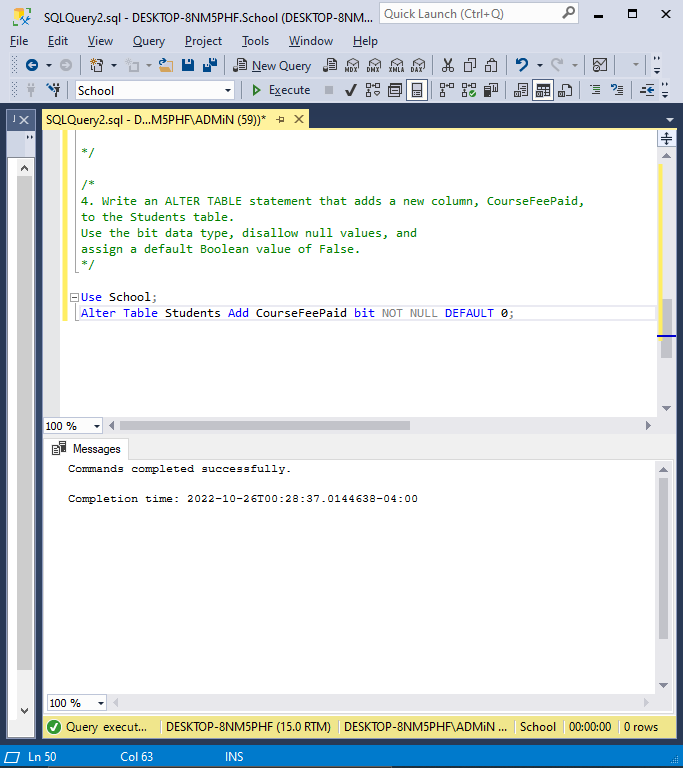
* Created an ALTER TABLE command to expand the Students table's CourseFeePaid column.
* Used a default Boolean value of False, forbid null values, and use the bit data type.

**Query:**

Use School;

Alter Table Students Add CourseFeePaid bit NOT NULL DEFAULT 0;

**Output:**

****

**5.** Write an ALTER TABLE statement that adds two new check constraints to the Invoices table in *the AP database*. The first should allow (1) PaymentDate to be null only if PaymentTotal is zero and (2) PaymentDate to be not null only if PaymentTotal is greater than zero. The second constraint should prevent the sum of PaymentTotal and CreditTotal from being greater than InvoiceTotal.

**Comment:**

* Created an ALTER TABLE command that updates the Invoices table in the AP database with two new check constraints.
* When it comes to the first, PaymentDate should only be able to be null if PaymentTotal is zero and it should only be possible for PaymentDate to be non-null if PaymentTotal is higher than zero.
* The second restriction should prevent InvoiceTotal from exceeding the sum of PaymentTotal and CreditTotal.

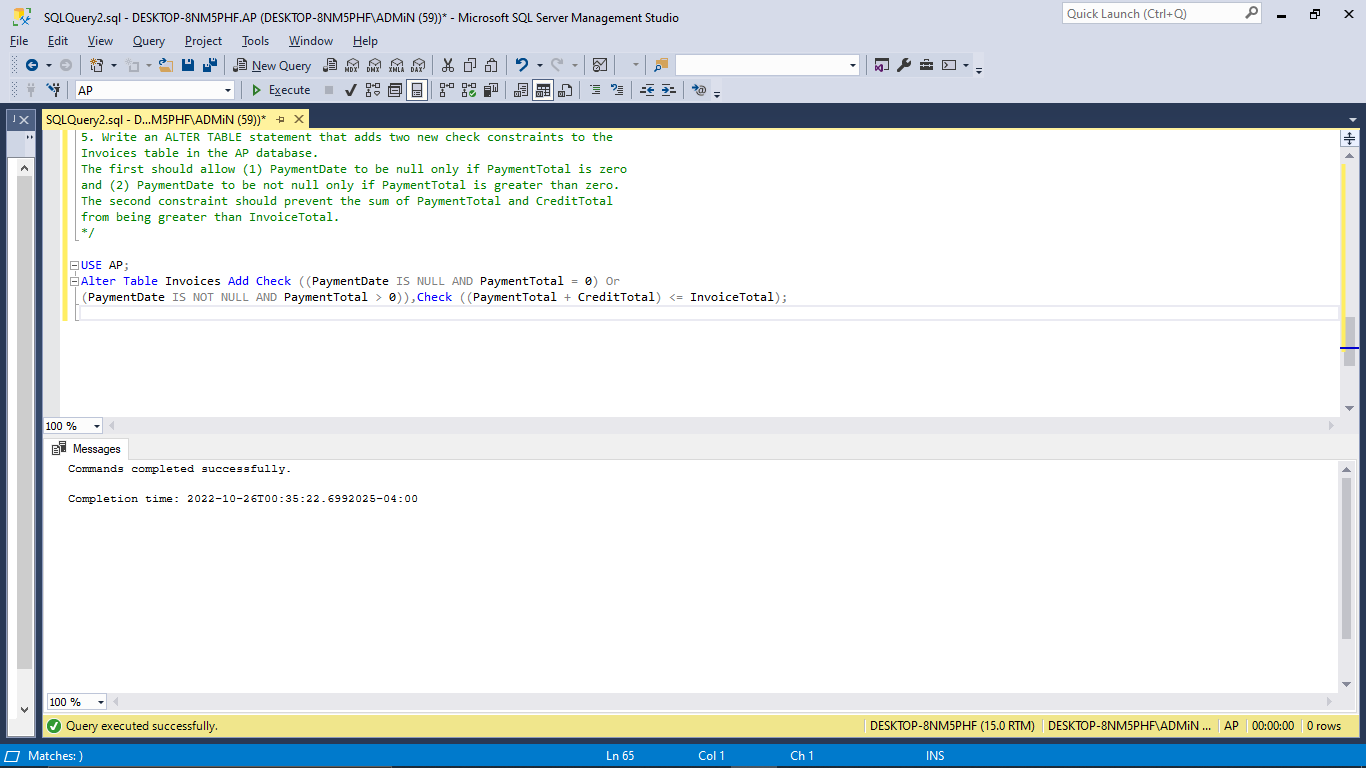
**Query:**

USE AP;

Alter Table Invoices Add Check ((PaymentDate IS NULL AND PaymentTotal = 0) Or

(PaymentDate IS NOT NULL AND PaymentTotal > 0)),Check ((PaymentTotal + CreditTotal) <= InvoiceTotal);

**Output:**

****

**6.** Delete the StudyGroups table from the School database. Then, write a CREATE TABLE statement that recreates the table, this time with a unique constraint that prevents a student from being a study-group member in the same course twice.

**Comment:**

* From the School database, deleted the StudyGroups table.
* Prepared a CREATE TABLE statement that recreates the table with a special constraint that forbids a student from participating in a study group for the same course more than once by using UNIQUE keyword.

**Query:**

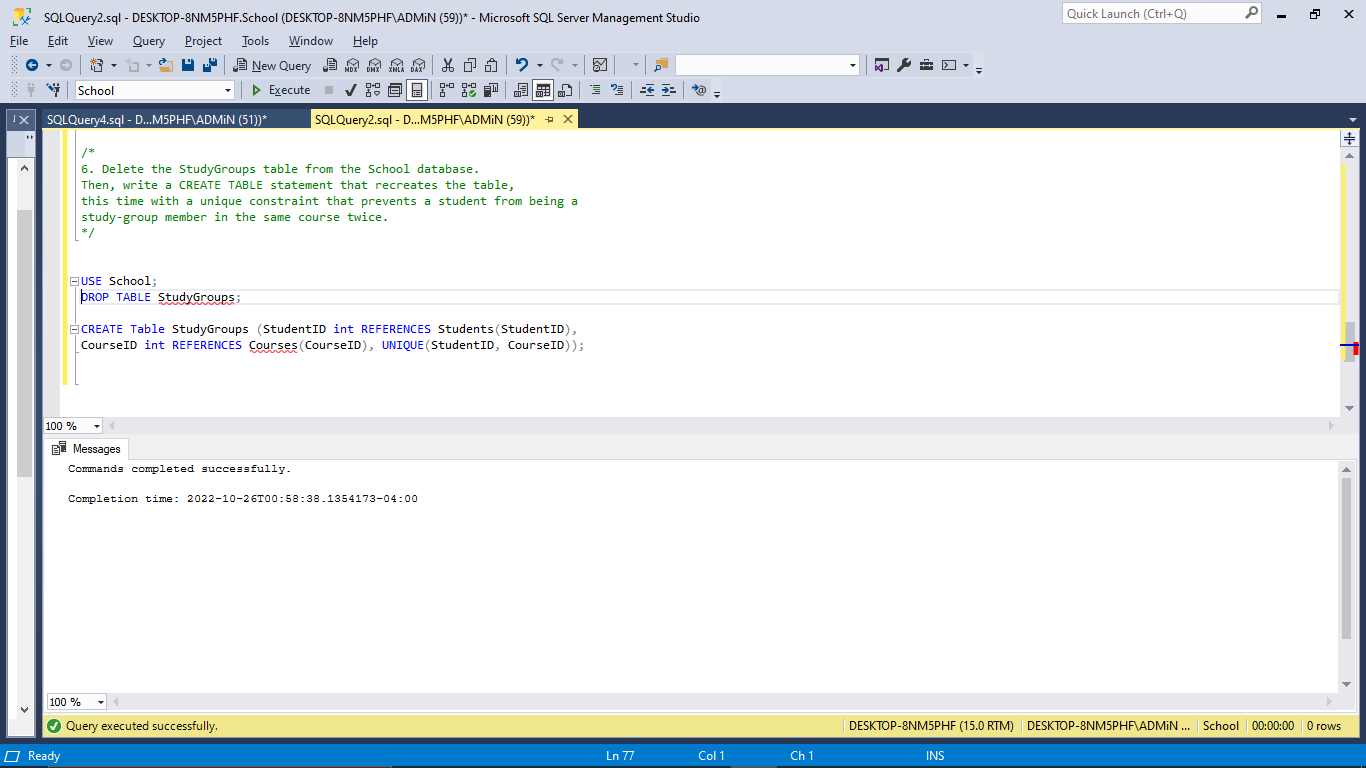
USE School;

DROP TABLE StudyGroups;

CREATE Table StudyGroups (StudentID int REFERENCES Students(StudentID),

CourseID int REFERENCES Courses(CourseID), UNIQUE(StudentID, CourseID));

**Output:**

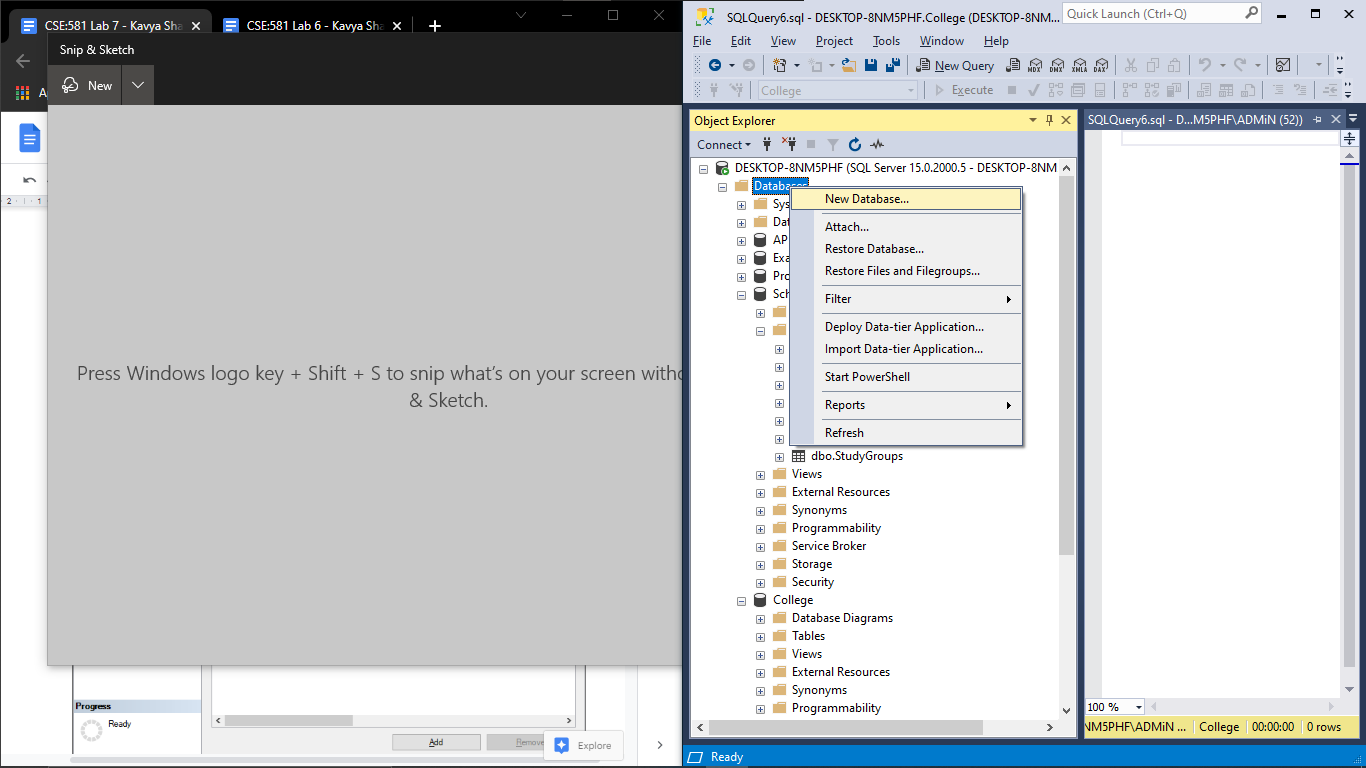
****

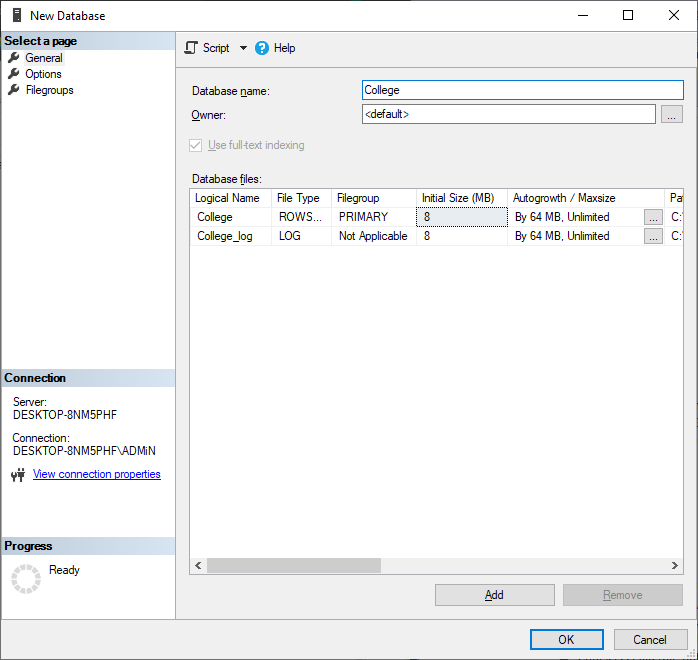
**7.** Use the Management Studio to create a new database called College using the default settings. (Do not use SQL query to do this).

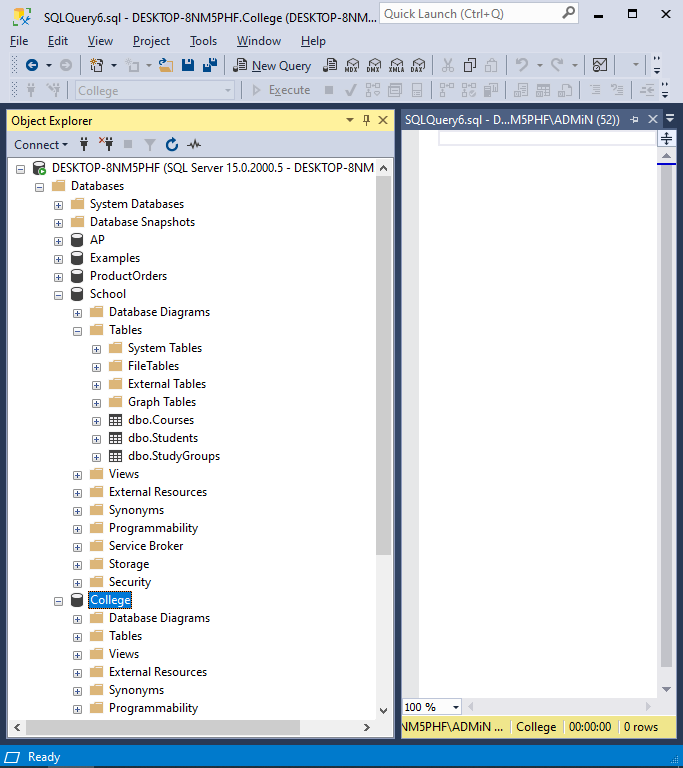
**Comments:**

* Manually made a new database called College by selecting the New Database option from the right-click menu, typing "College," and then clicking OK.

**Output:**

****

****

****

# Remarks for the lab:

The overall lab consisted the designing and implementation of the databases.