CSE 581 – Intro To Database Management Systems

Project 1

Rachana Mohan Fulsundar

SUID : 512373928

11/11/2022

Abstract :

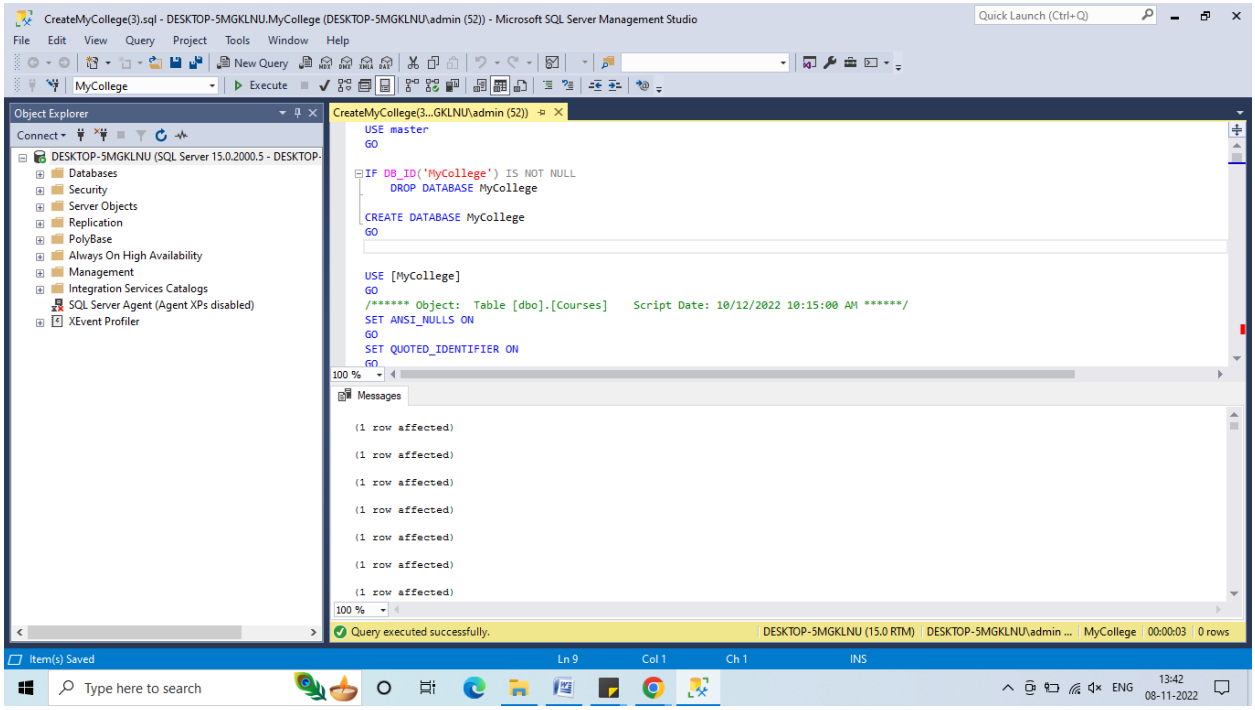
In this project, we'll use SQL Server Management Studio to create MyCollege database and enter SQL statements and run them against this database. This database consists Courses, Departments, Instructors, StudentCourses, Students and Tution tables.

We will first do database setup by executing the scripts provided, then perform DML operations on this database. We'll also write views, scripts, stored procedures, functions and design database diagrams.

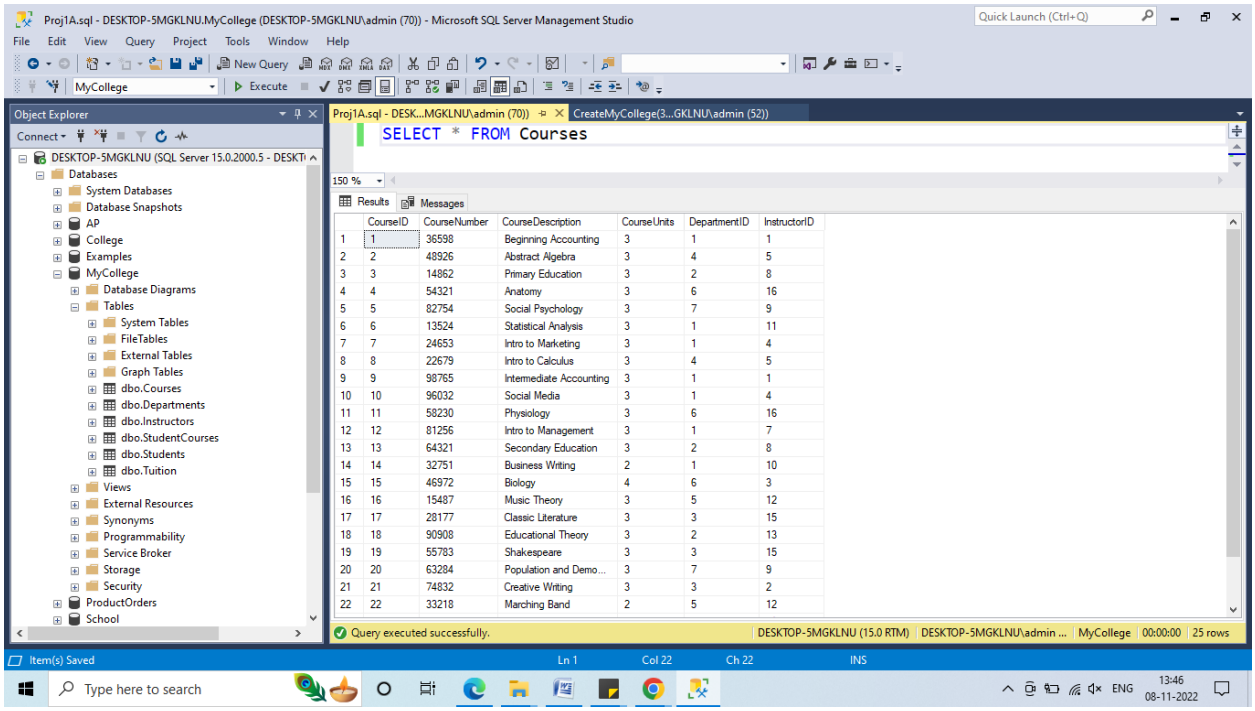
Table of Contents

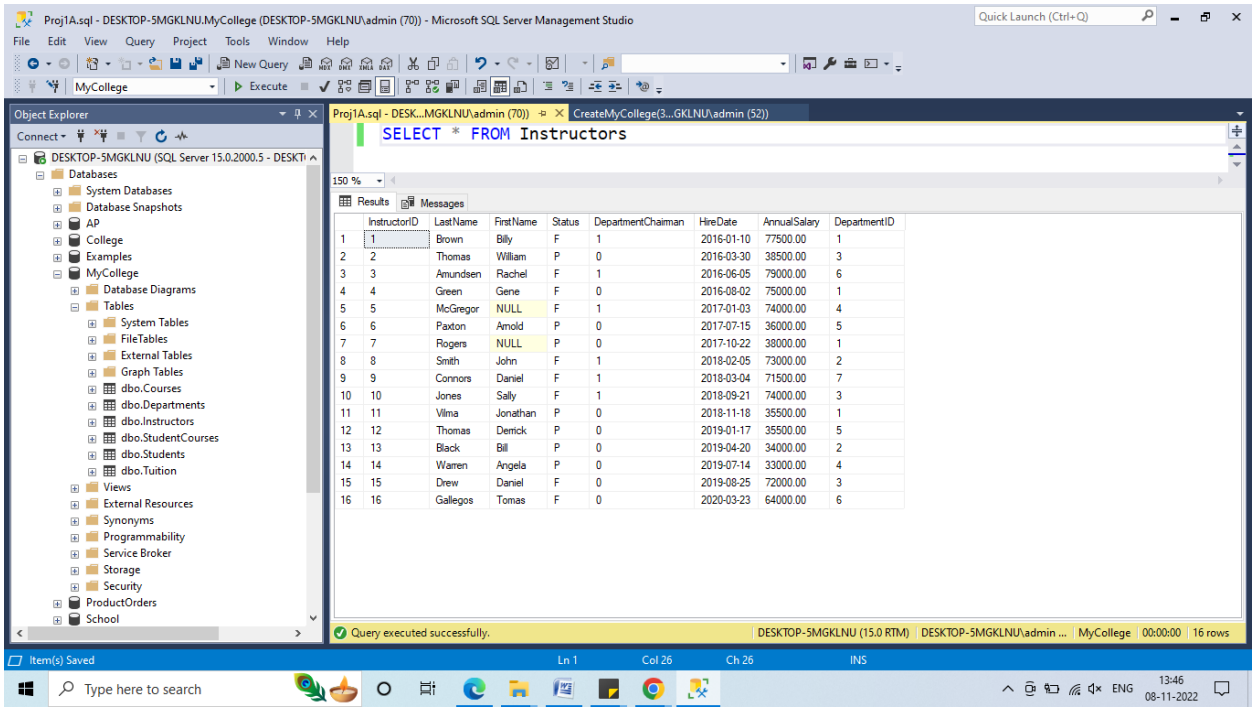
Section A : Database Setup

1. Download sql files from this project tab and open CreatMyCollege.sql in SQL server management studio. Execute the entire script and show the message in the Message tab, indicating the script is executed successfully.

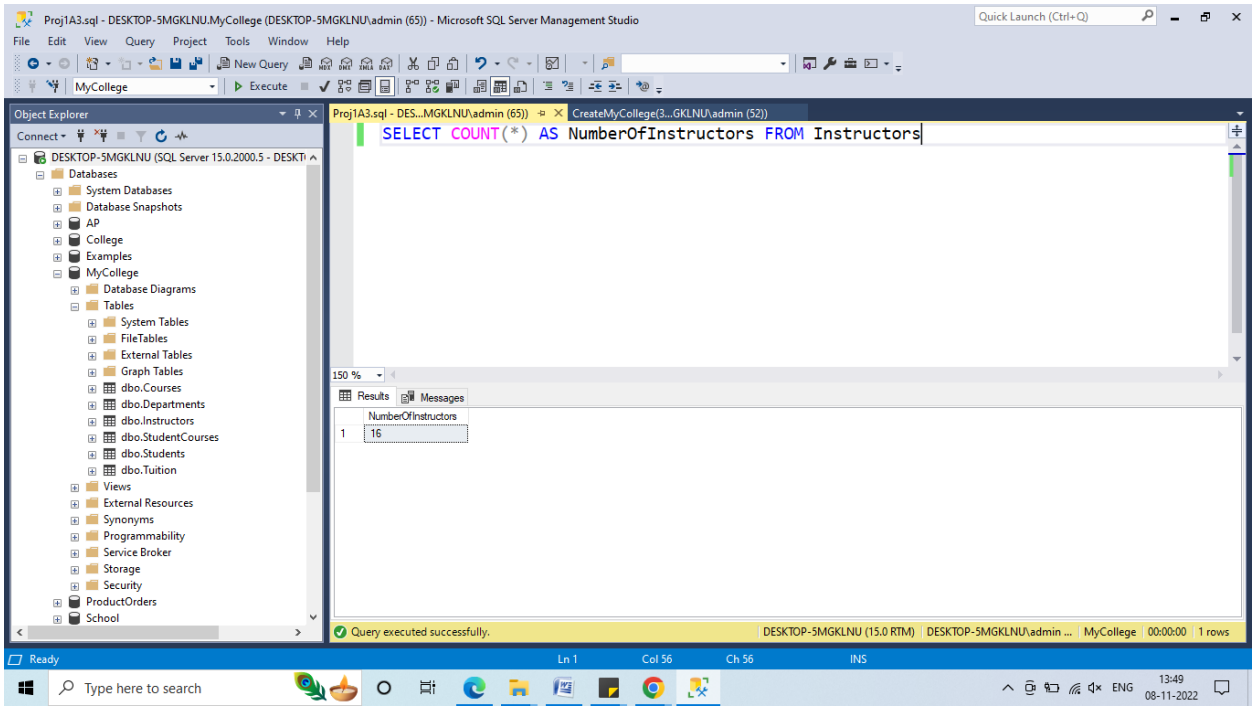


2. Navigate through the database objects and view the column definitions for each table. Open a new Query Editor window. Show details in Courses table and Instructors table using SELECT statement.

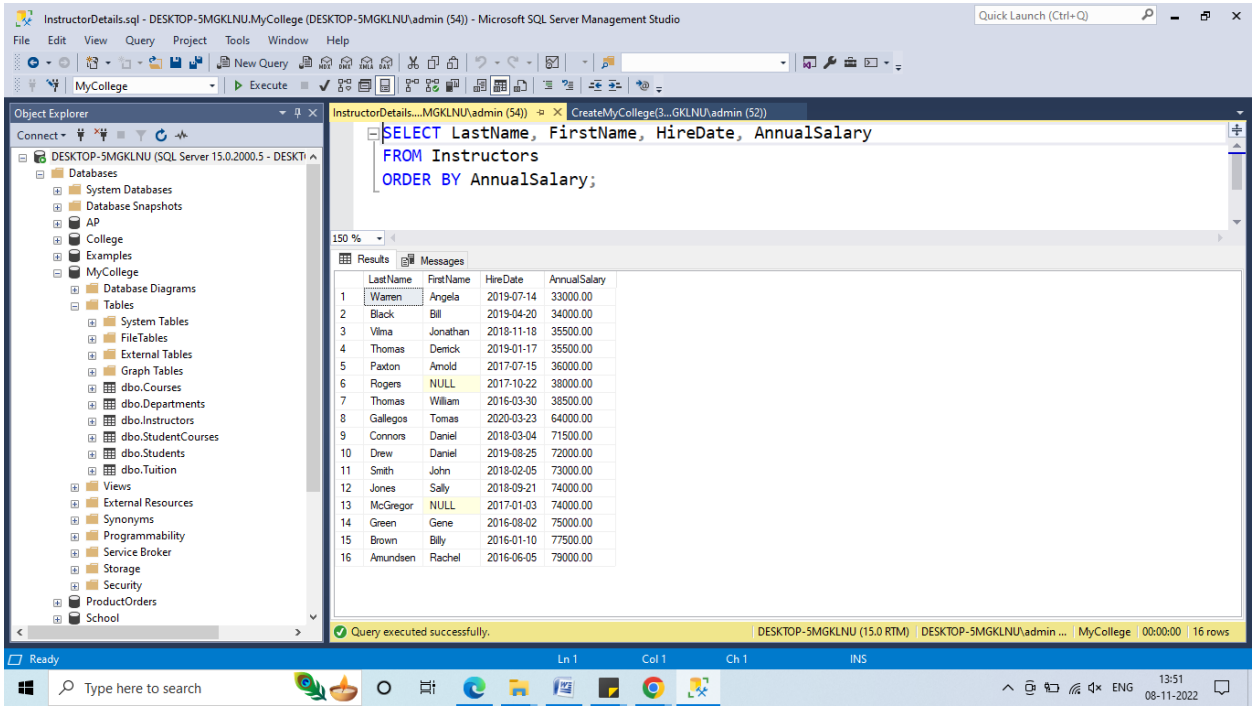




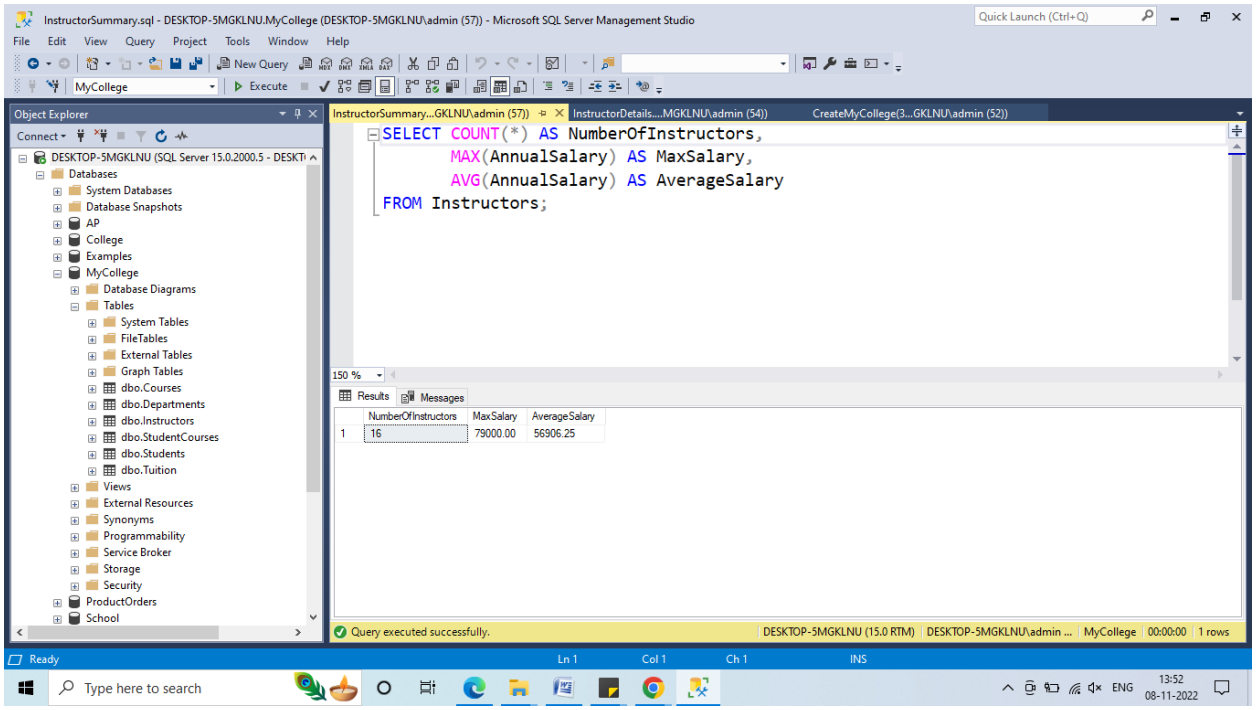
3. Open another Query Editor window and then enter and run this statement
SELECT COUNT(*) AS NumberOfInstructors FROM Instructors



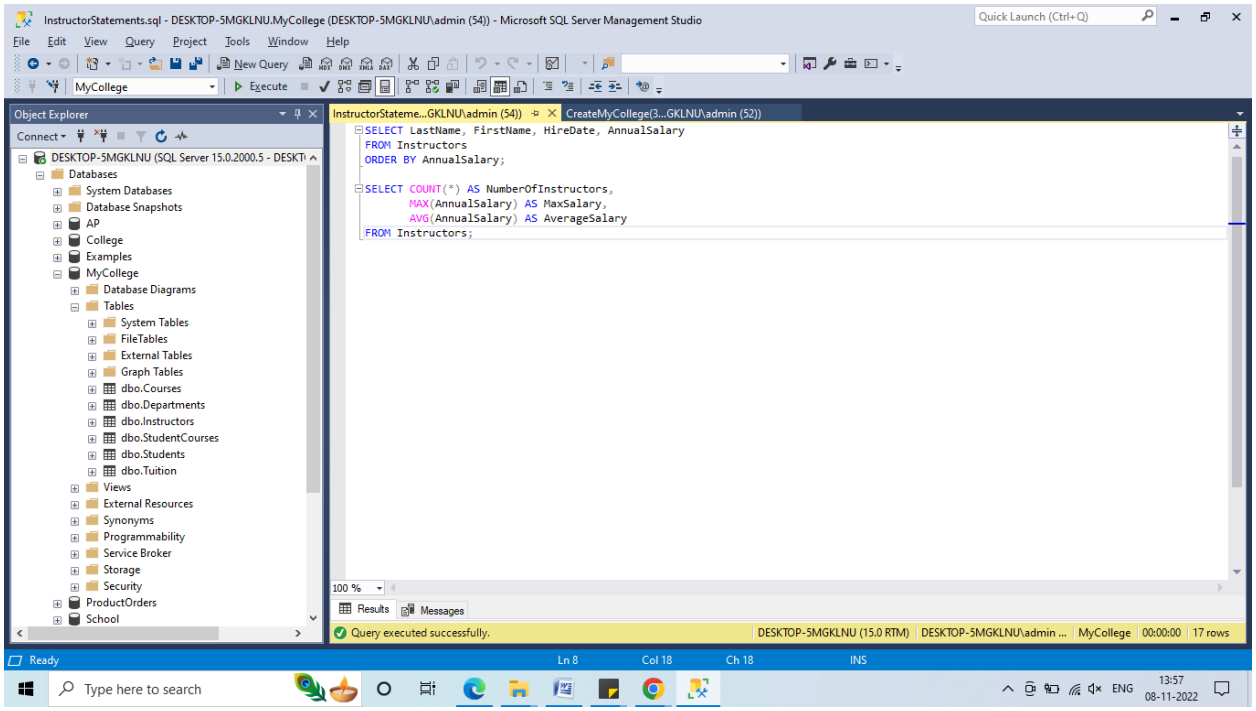
4. Open the script named InstructorDetails.sql. Note that this script contains just one SQL statement. Then, run the statement.



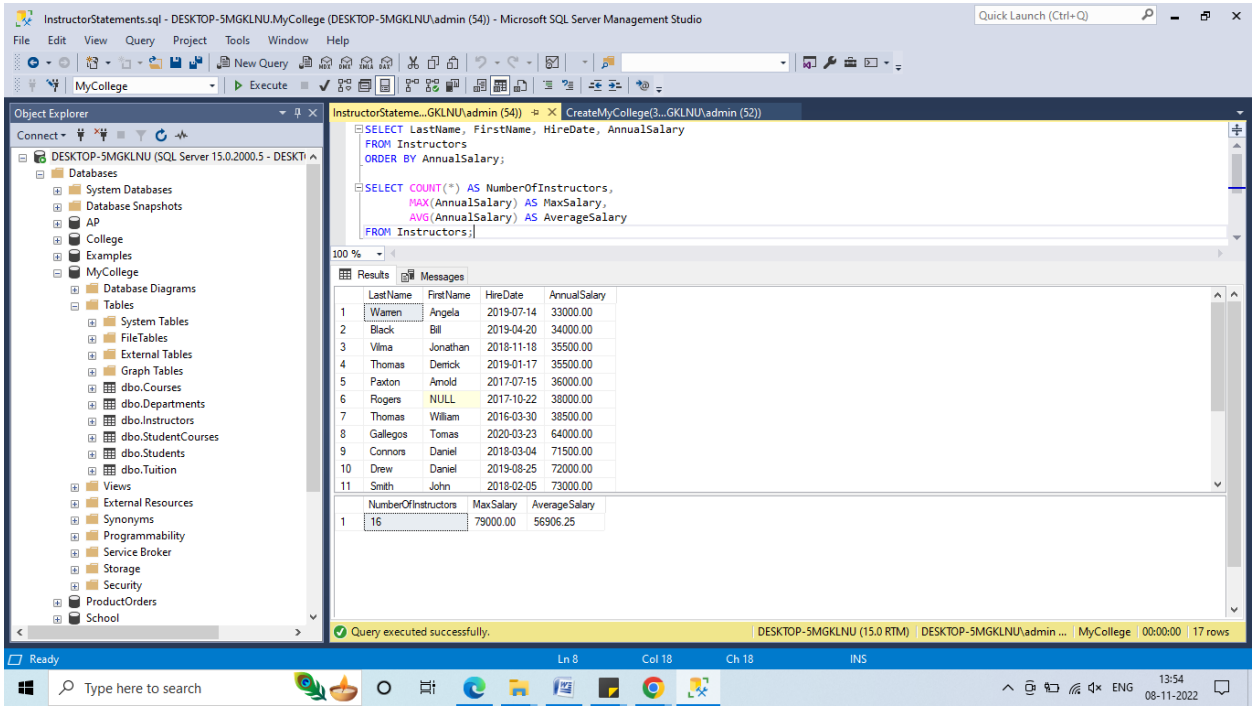
5. Open the script named InstructorSummary.sql. Note that this opens another Query Editor window.



6. Open the script named InstructorStatements.sql. Note that this script contains two SQL statements that end with semicolons.



7. Press the F5 key or click the Execute button to run both statements in this script. Note that this displays the results in two Results tabs. Make sure to view the results of both SELECT statements.

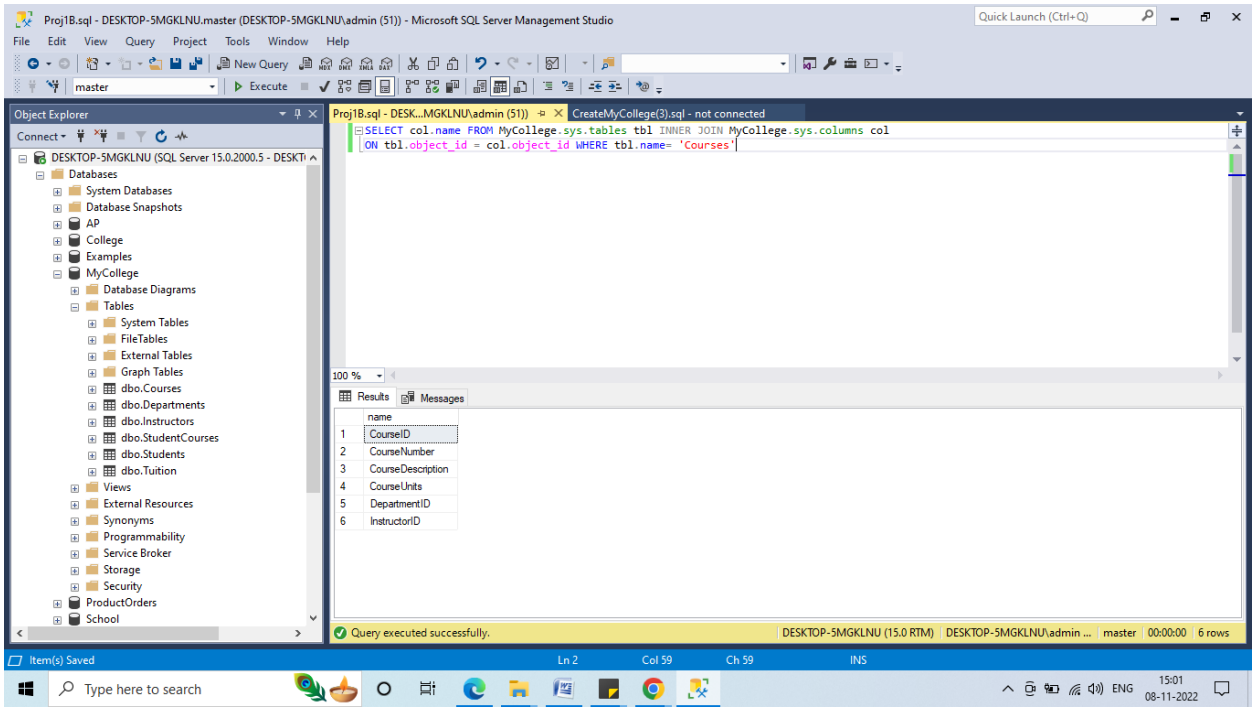


8. Exit from SQL Server Management Studio.

Section B : Essential SQL Skills

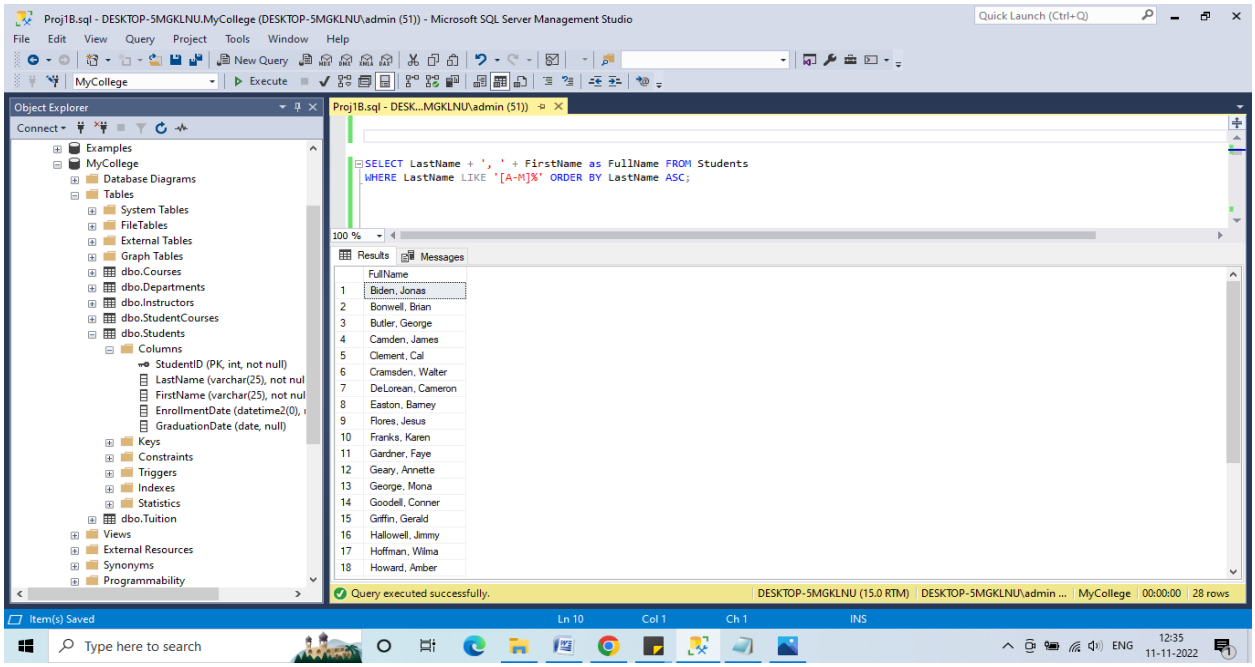
1. Display all the columns from the Courses table

```
SELECT col.name FROM MyCollege.sys.tables tbl INNER JOIN MyCollege.sys.columns col
ON tbl.object_id = col.object_id WHERE tbl.name= 'Courses'
```



2. Display FullName as LastName + ', ' + FirstName. Sort the result set by last name in ascending sequence.Return only the students whose last name begins with a letter from A to M.

```
SELECT LastName + ', ' + FirstName as FullName FROM Students
WHERE LastName LIKE '[A-M]%' ORDER BY LastName ASC;
```



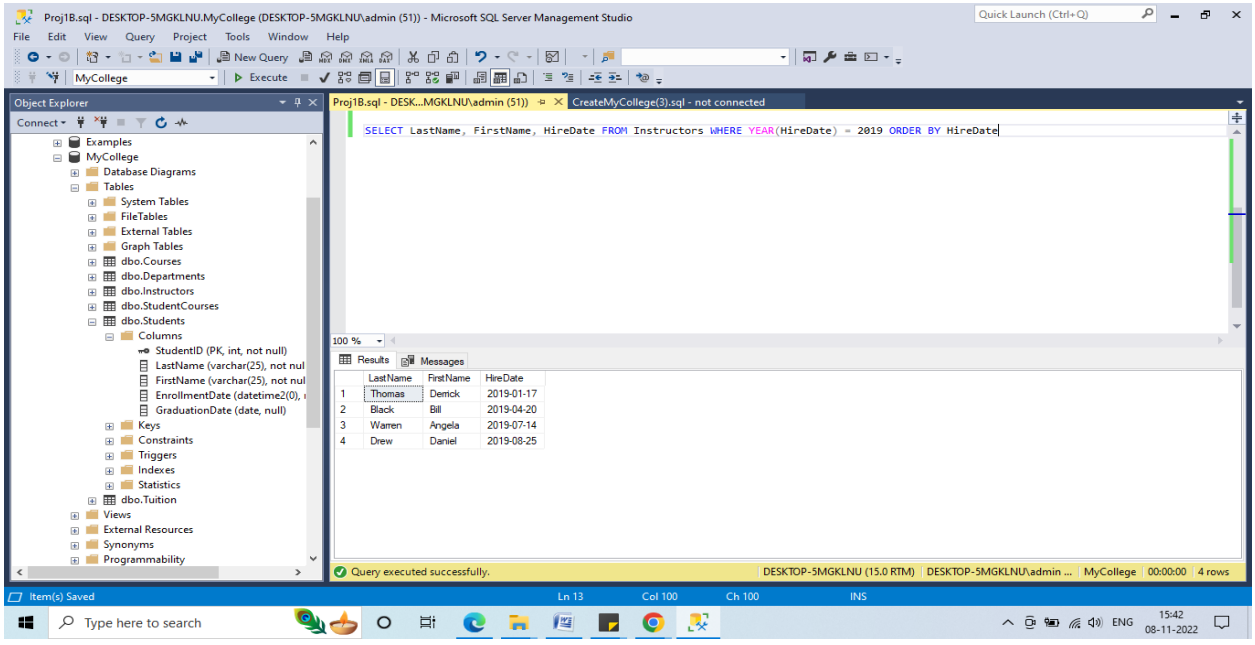
3. Display following column and Return only the rows with a hire date that's in 2019. Sort the result set in ascending sequence by the HireDate column

   LastName The LastName colum

   FirstName The FirstName column

   HireDate The HireDate column

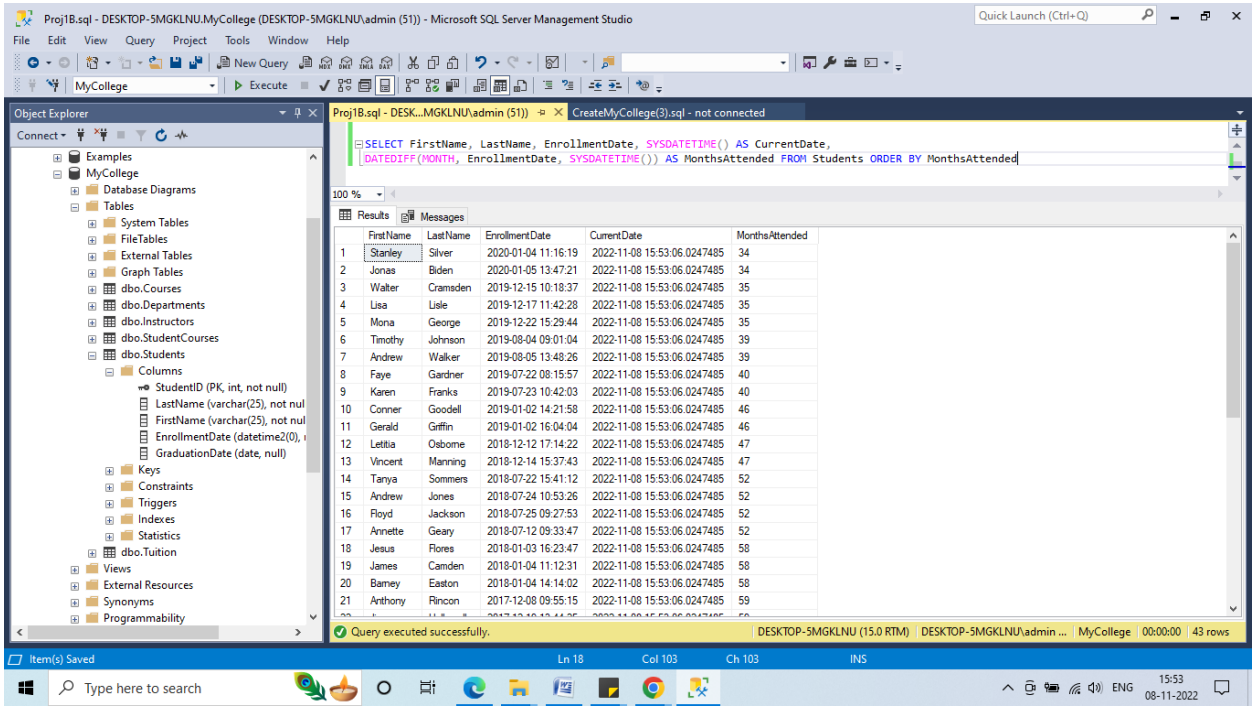
```
SELECT LastName, FirstName, HireDate FROM Instructors WHERE YEAR(HireDate) = 2019
ORDER BY HireDate
```



4. Display below columns and Sort result set in ascending sequence by MonthsAttended column

   FirstName The FirstName column
   LastName The LastName column
   EnrollmentDate The EnrollmentDate column
   CurrentDate The current date
   MonthsAttended A column that's calculated by getting the difference between the enrollment date and the current date

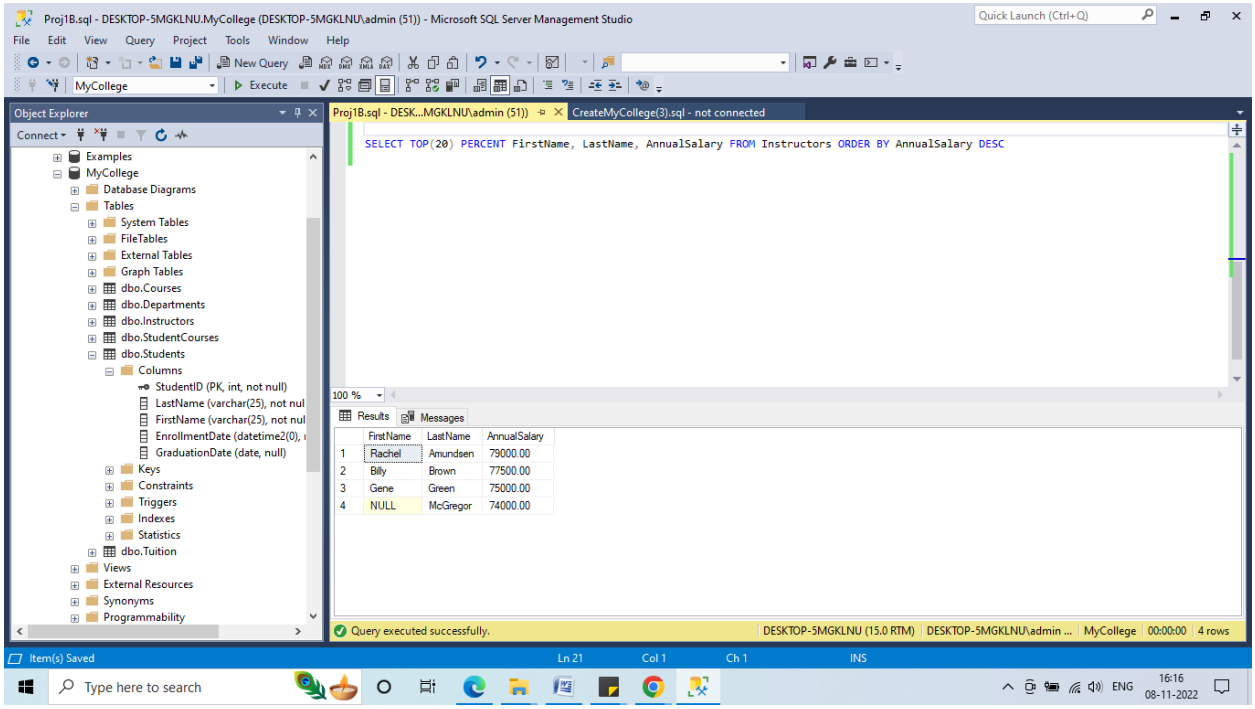
```
SELECT FirstName, LastName, EnrollmentDate, SYSDATETIME() AS CurrentDate,
DATEDIFF(MONTH, EnrollmentDate, SYSDATETIME()) AS MonthsAttended FROM Students
ORDER BY MonthsAttended
```



5. Display below columns and Return only the top 20 percent of instructors based on annual salary

   FirstName The FirstName column
   LastName The LastName column
   AnnualSalary The AnnualSalary column

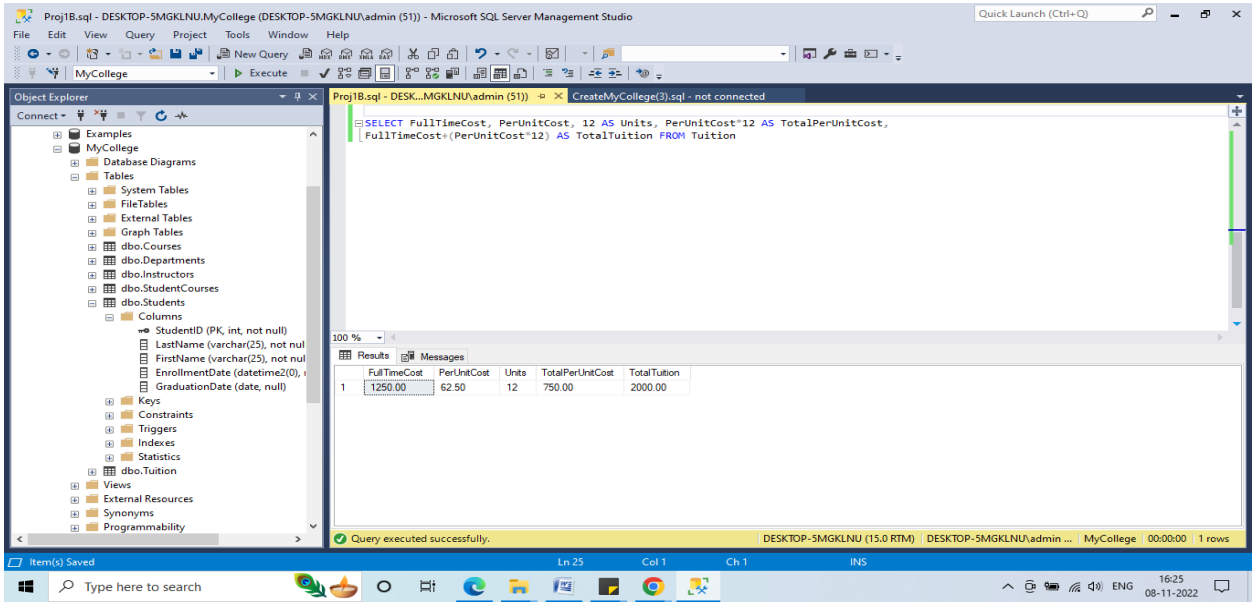
```
SELECT TOP(20) PERCENT FirstName, LastName, AnnualSalary FROM Instructors ORDER BY
AnnualSalary DESC
```



6. Display below columns

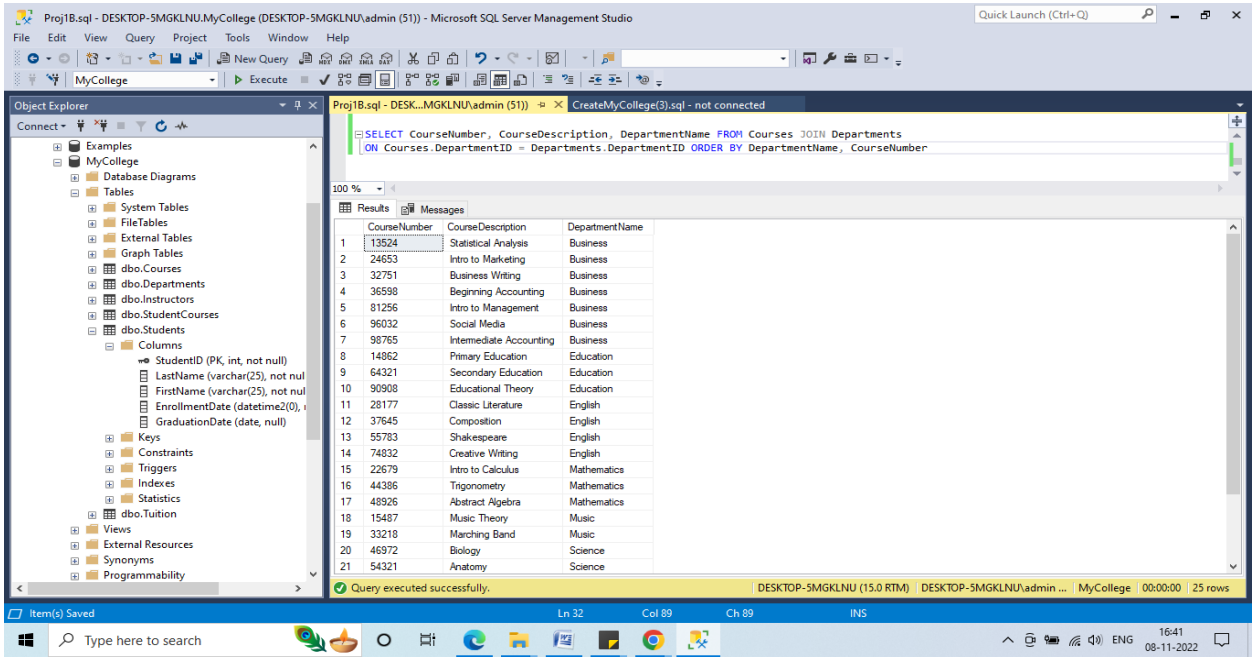
FullTimeCost	The FullTimeCost column
PerUnitCost	The PerUnitCost column
Units	12
TotalPerUnitCost	A column that's calculated by multiplying the per unit cost by the units
TotalTuition	A column that's calculated by adding the full time cost to the total per unit cost

```
SELECT FullTimeCost, PerUnitCost, 12 AS Units, PerUnitCost*12 AS TotalPerUnitCost,
FullTimeCost+(PerUnitCost*12) AS TotalTuition FROM Tuition
```



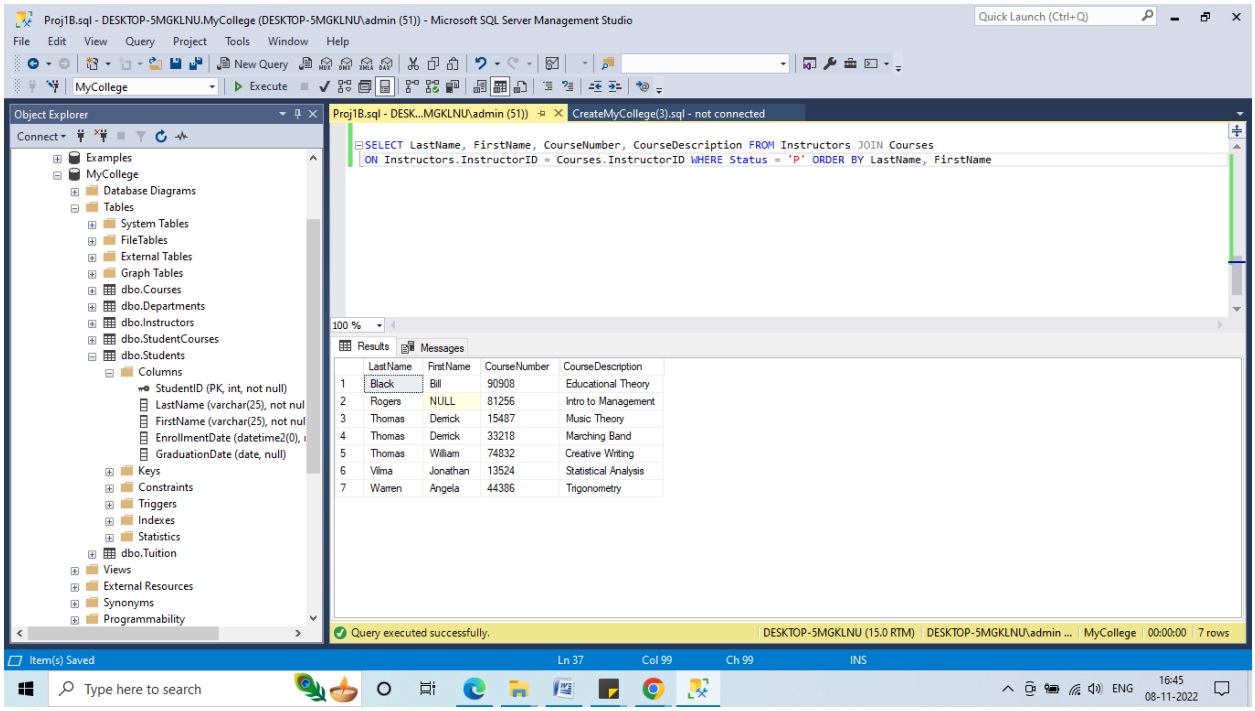
7. Join Courses table to Departments table and return: CourseNumber, CourseDescription, DepartmentName. Sort result set by DepartmentName and CourseNumber in ascending order.

```
SELECT CourseNumber, CourseDescription, DepartmentName FROM Courses JOIN
Departments ON Courses.DepartmentID = Departments.DepartmentID ORDER BY
DepartmentName, CourseNumber
```



8. Join Instructors table to Courses table and returns: LastName, FirstName, CourseNumber, CourseDescription. Return all courses for each instructor with a status of “P” (part time). Sort the result set by LastName and FirstName in ascending order

```
SELECT LastName, FirstName, CourseNumber, CourseDescription FROM Instructors JOIN
Courses ON Instructors.InstructorID = Courses.InstructorID WHERE Status = 'P'
ORDER BY LastName, FirstName
```

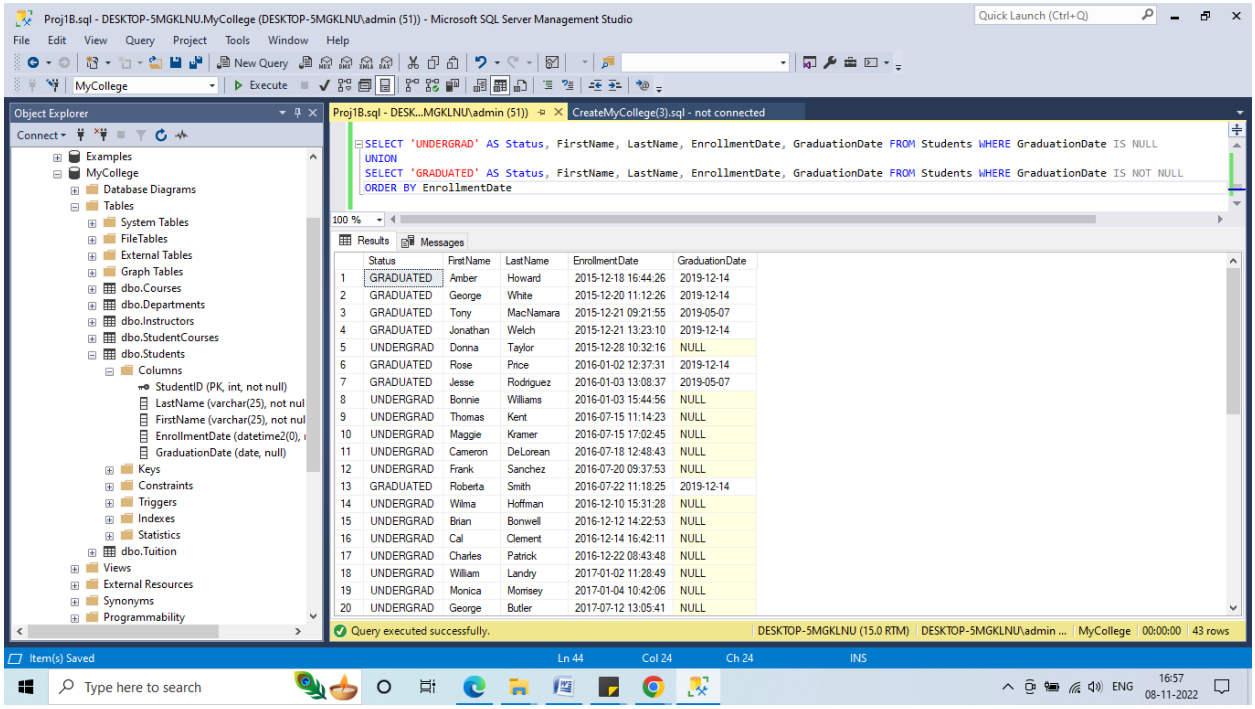


9. Use the UNION operator to generate a result set consisting below columns

Status	A calculated column that contains a value of UNDERGRAD or GRADUATED
FirstName	The FirstName column
LastName	The LastName column
EnrollmentDate	The EnrollmentDate column
GraduationDate	The GraduationDate column

If the student doesn’t have a value in the GraduationDate column, the Status column should contain a value of UNDERGRAD. Otherwise, it should contain a value of GRADUATED.
Sort the final result set by EnrollmentDate.

```
SELECT 'UNDERGRAD' AS Status, FirstName, LastName, EnrollmentDate, GraduationDate
FROM Students WHERE GraduationDate IS NULL
UNION
SELECT 'GRADUATED' AS Status, FirstName, LastName, EnrollmentDate, GraduationDate
FROM Students WHERE GraduationDate IS NOT NULL
ORDER BY EnrollmentDate
```



10. Display below columns:

    InstructorDept The DepartmentName column from the Departments table for a related instructor

    LastName The LastName column from the Instructors table

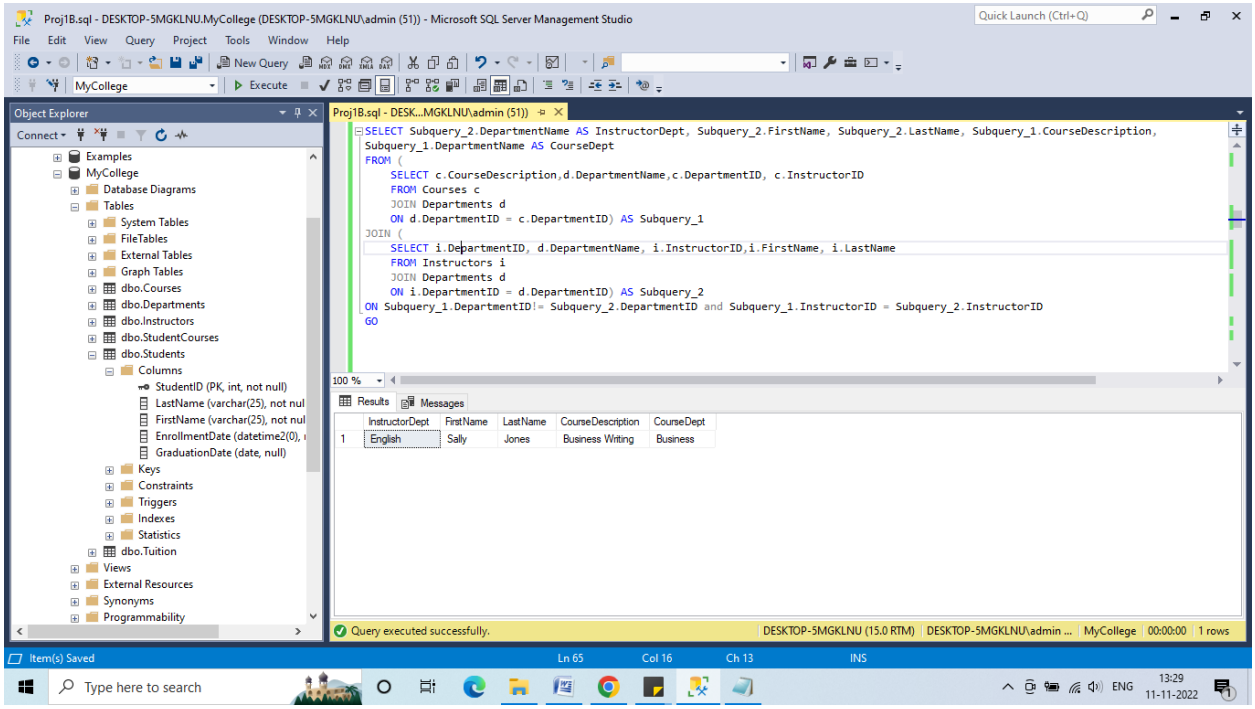
    FirstName The FirstName column from the Instructors table

    CourseDescription The CourseDescription column from the Courses table

    CourseDept The DepartmentName column from the Departments table for a related instructor

    Return one row for each course that's in a different department than the department of the instructor assigned to teach that course

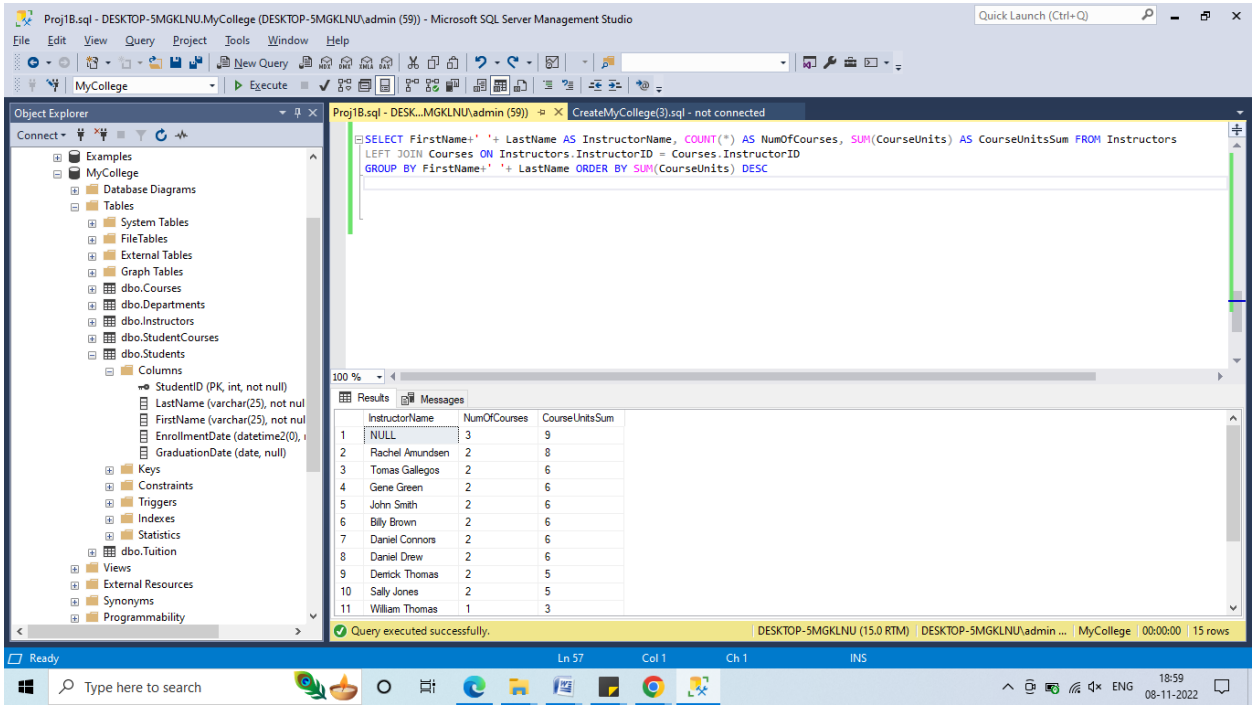
```
SELECT Subquery_2.DepartmentName AS InstructorDept, Subquery_2.FirstName,
Subquery_2.LastName, Subquery_1.CourseDescription,
Subquery_1.DepartmentName AS CourseDept FROM (
SELECT c.CourseDescription,d.DepartmentName,c.DepartmentID, c.InstructorID
FROM Courses c JOIN Departments d
ON d.DepartmentID = c.DepartmentID) AS Subquery_1
JOIN (
SELECT i.DepartmentID, d.DepartmentName, i.InstructorID,i.FirstName, i.LastName
FROM Instructors I JOIN Departments d
ON i.DepartmentID = d.DepartmentID) AS Subquery_2
ON Subquery_1.DepartmentID!= Subquery_2.DepartmentID and Subquery_1.InstructorID =
Subquery_2.InstructorID
GO
```



11. Display one row for each instructor that has courses with these columns:
    The instructor first and last names from the Instructors table in this format: John Doe (Note: If the instructor first name has a null value, the concatenation of the first and last name will result in a null value.)
    A count of the number of courses in the Courses table
    The sum of the course units in the Courses table
    Sort the result set in descending sequence by the total course units for each instructor

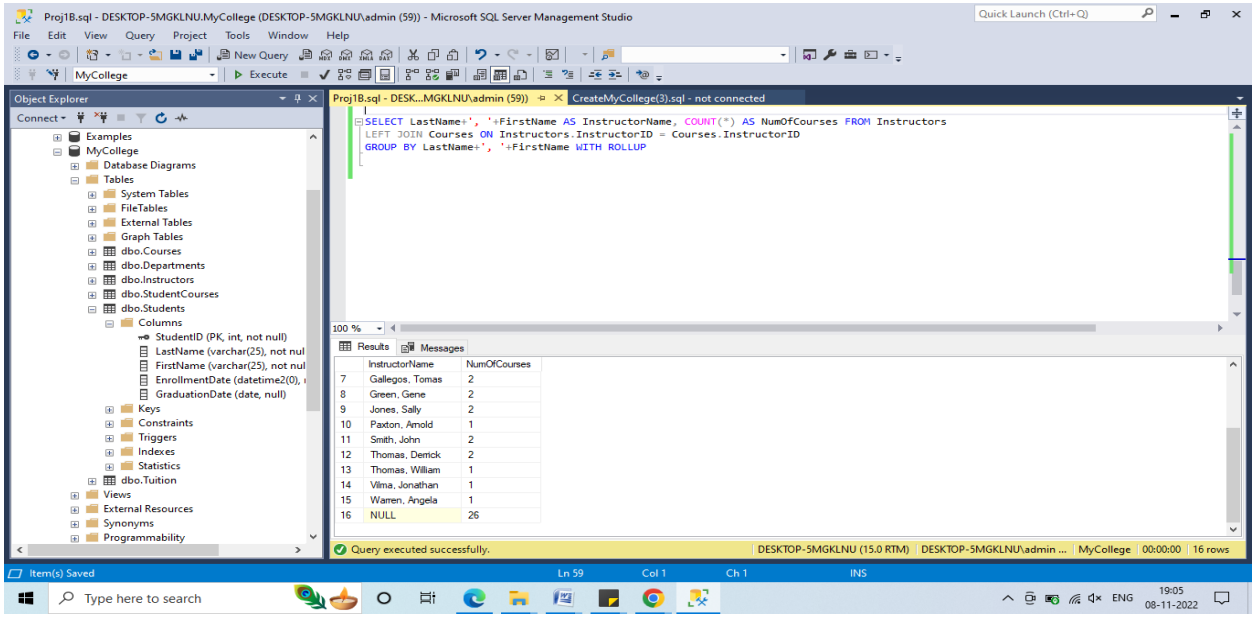
```
SELECT FirstName+' '+ LastName AS InstructorName, COUNT(*) AS NumOfCourses,
SUM(CourseUnits) AS CourseUnitsSum FROM Instructors

LEFT JOIN Courses ON Instructors.InstructorID = Courses.InstructorID
GROUP BY FirstName+' '+ LastName ORDER BY SUM(CourseUnits) DESC
```



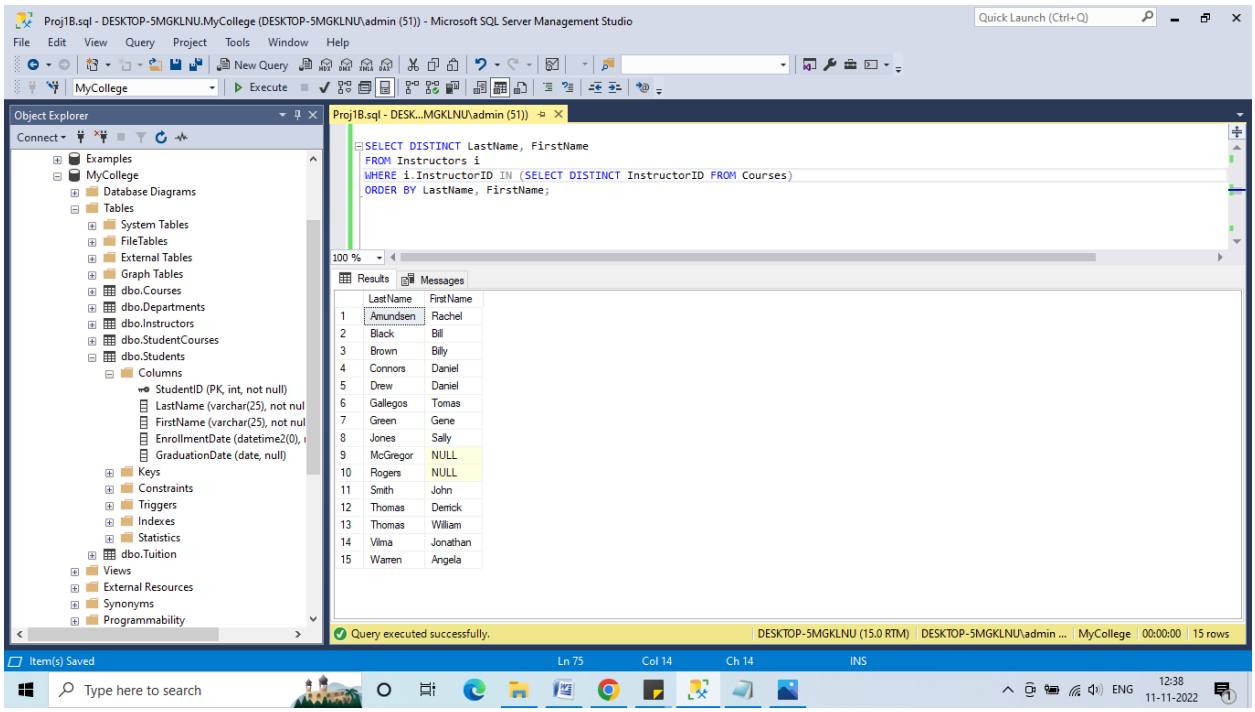
12. Display the total number of courses taught by part time instructors, instructor last name and first name from the Instructors table in this format: Doe, John (Note: If the instructor first name has a null value, the concatenation of the first and last name will result in a null value.)
    The total number of courses taught for each instructor in the Courses table
    Use the ROLLUP operator to include a row that gives the grand total.

```
SELECT LastName+', '+FirstName AS InstructorName, COUNT(*) AS NumOfCourses FROM
Instructors
LEFT JOIN Courses ON Instructors.InstructorID = Courses.InstructorID
GROUP BY LastName+', '+FirstName WITH ROLLUP
```



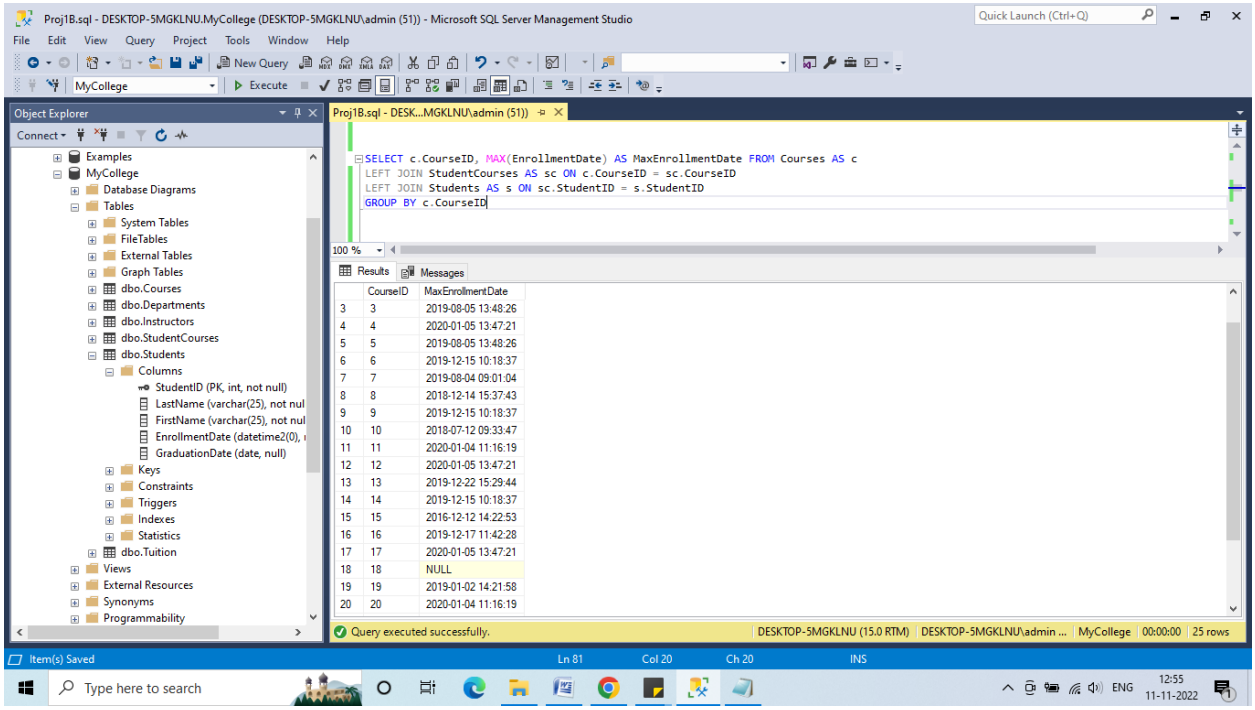
13. use a subquery in a WHERE clause that uses the IN keyword.

```
SELECT DISTINCT LastName, FirstName
FROM Instructors i
WHERE i.InstructorID IN (SELECT DISTINCT InstructorID FROM Courses)
ORDER BY LastName, FirstName;
```



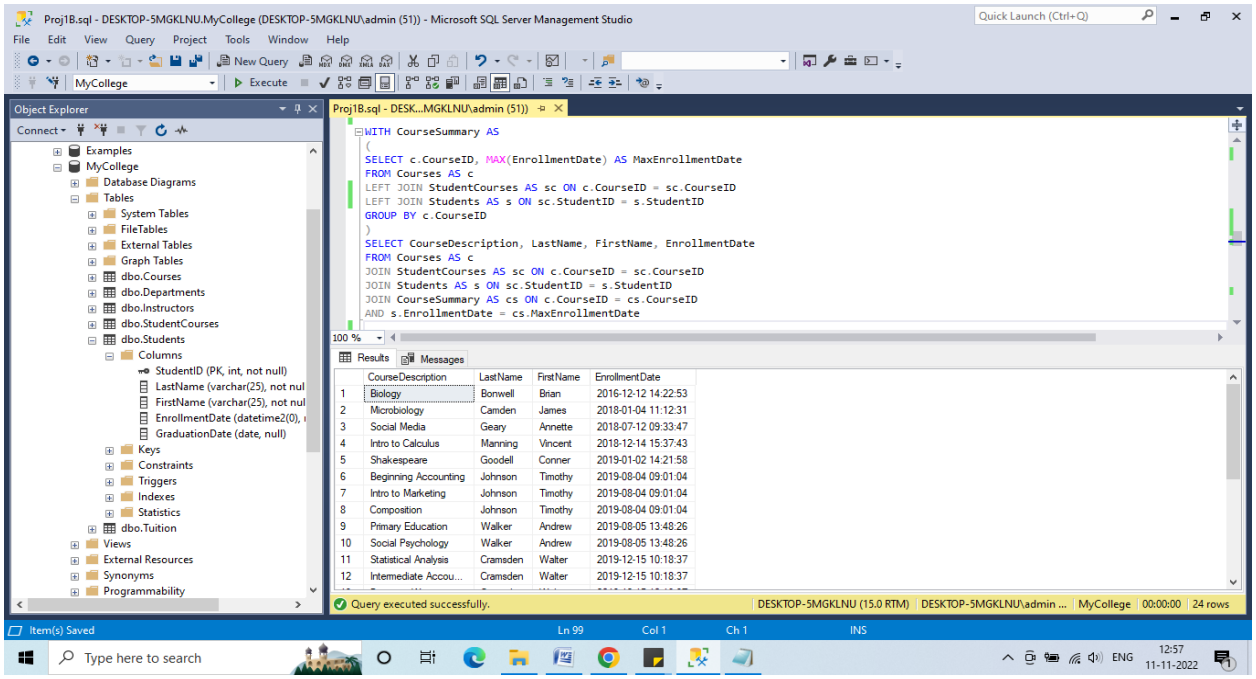
14. Display one row for each course with these columns :CourseID and enrollment date

```
SELECT c.CourseID, MAX(EnrollmentDate) AS MaxEnrollmentDate FROM Courses AS c
LEFT JOIN StudentCourses AS sc ON c.CourseID = sc.CourseID
LEFT JOIN Students AS s ON sc.StudentID = s.StudentID
GROUP BY c.CourseID
```



Use CTE and display one row per course that shows the CourseDescription for the course and the LastName, FirstName, and EnrollmentDate for the student with the most recent enrollment data.

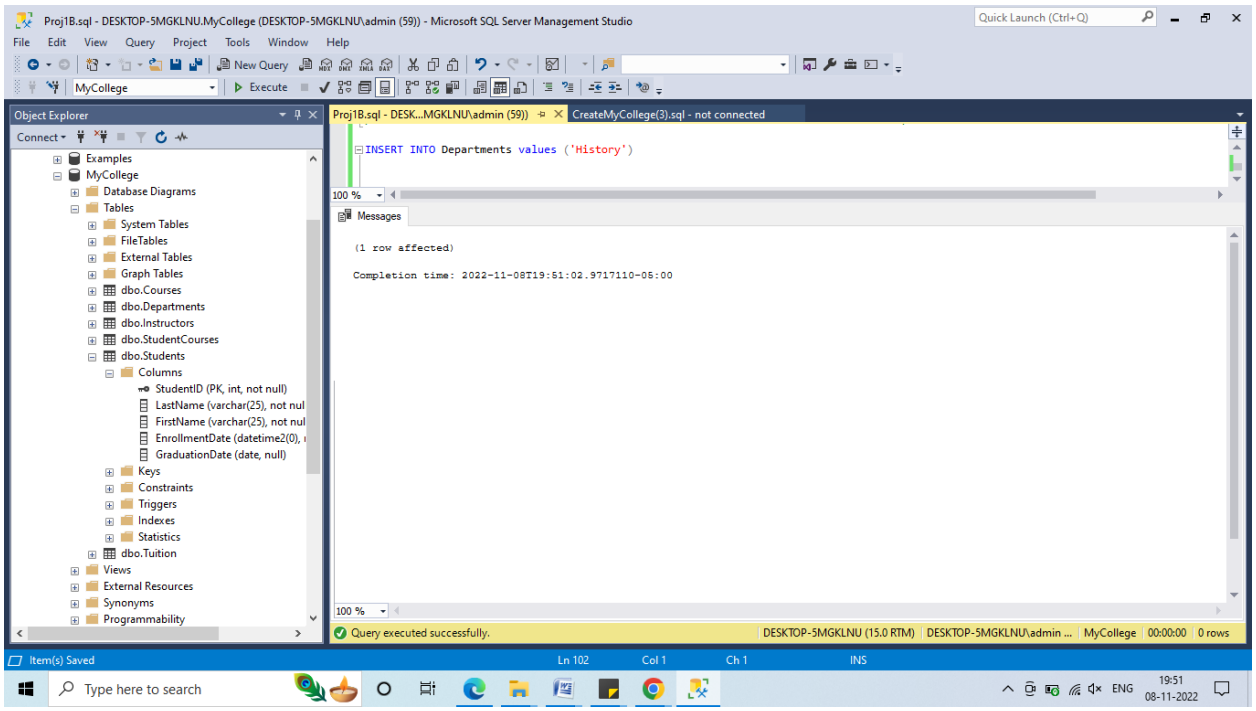
```
WITH CourseSummary AS
(
SELECT c.CourseID, MAX(EnrollmentDate) AS MaxEnrollmentDate
FROM Courses AS c
LEFT JOIN StudentCourses AS sc ON c.CourseID = sc.CourseID
LEFT JOIN Students AS s ON sc.StudentID = s.StudentID
GROUP BY c.CourseID
)
SELECT CourseDescription, LastName, FirstName, EnrollmentDate
FROM Courses AS c
JOIN StudentCourses AS sc ON c.CourseID = sc.CourseID
JOIN Students AS s ON sc.StudentID = s.StudentID
JOIN CourseSummary AS cs ON c.CourseID = cs.CourseID
AND s.EnrollmentDate = cs.MaxEnrollmentDate
```

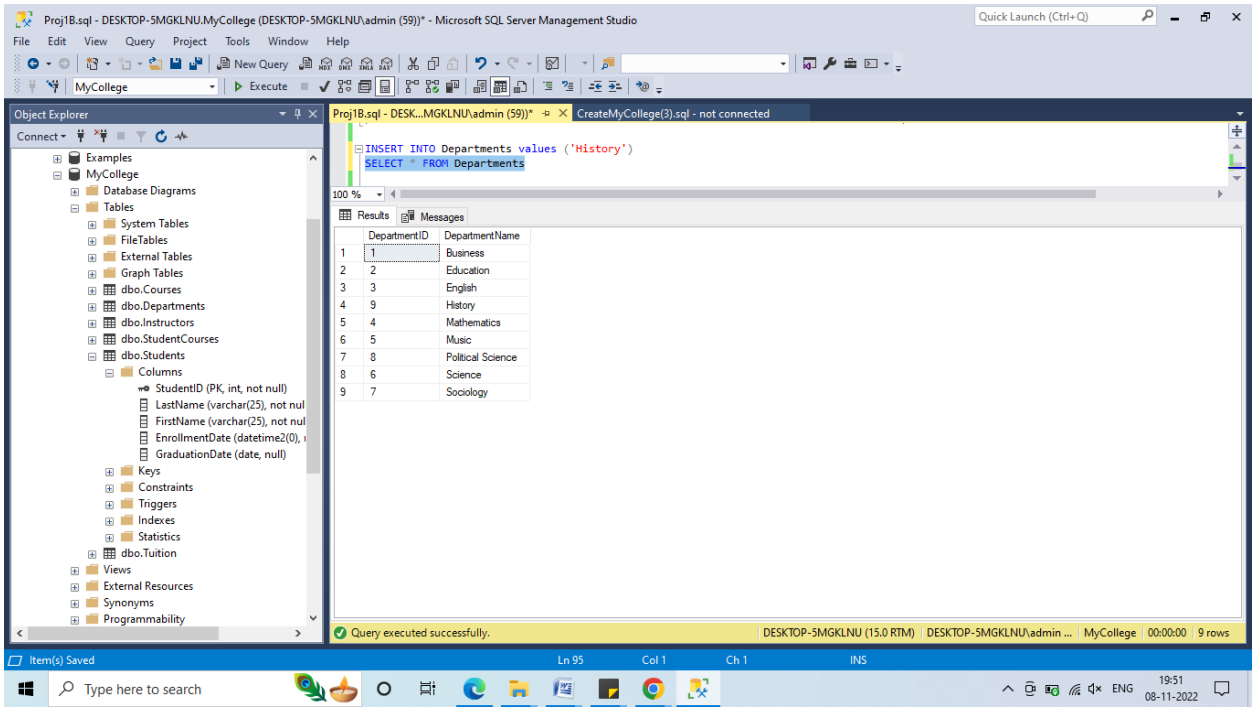


15. Add below row to the Departments table:

    DepartmentName: History

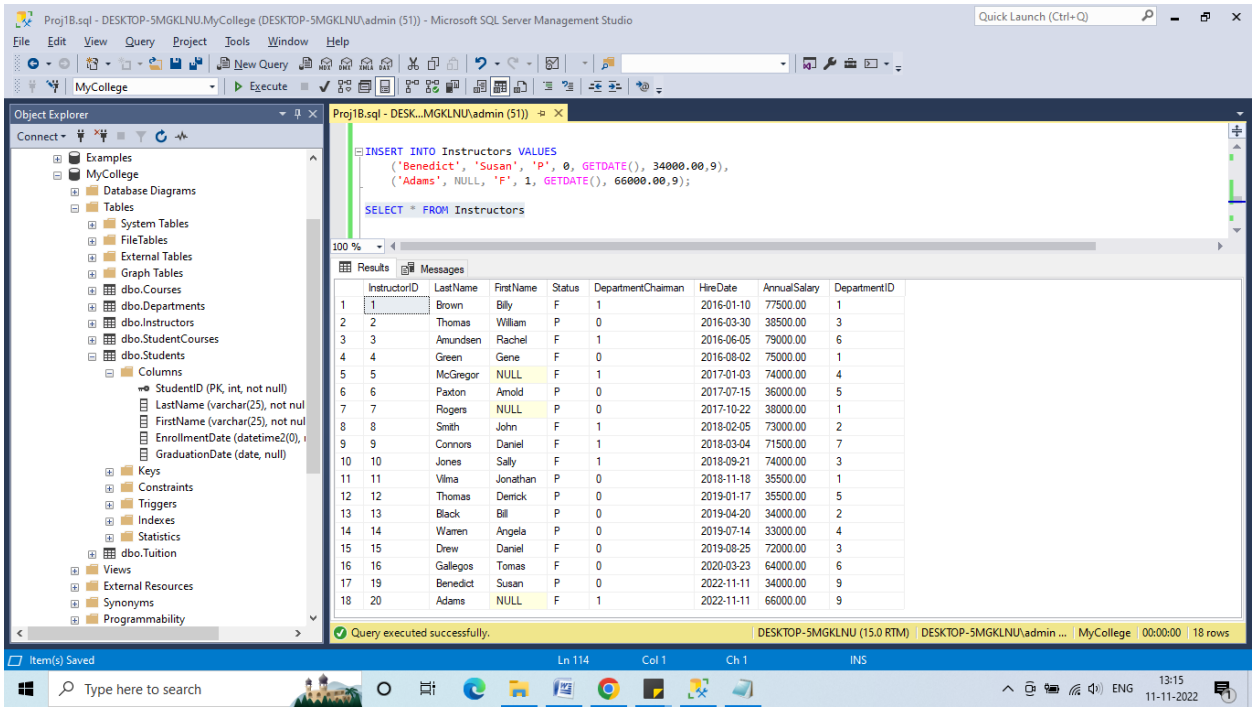
```
INSERT INTO Departments values ('History')
SELECT * FROM Departments
```





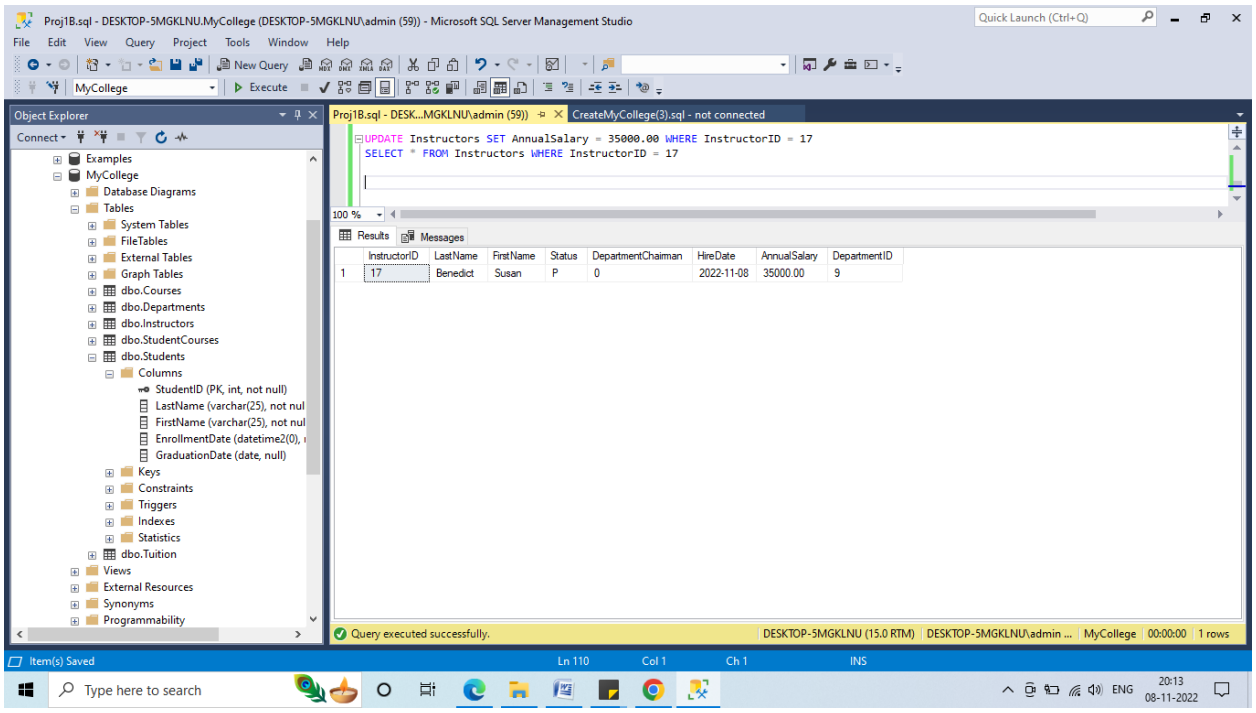
16. Write a single INSERT statement that adds these rows to the Instructors table:

```
INSERT INTO Instructors VALUES
('Benedict', 'Susan', 'P', 0, GETDATE(), 34000.00,9),
('Adams', NULL, 'F', 1, GETDATE(), 66000.00,9);
SELECT * FROM Instructors
```



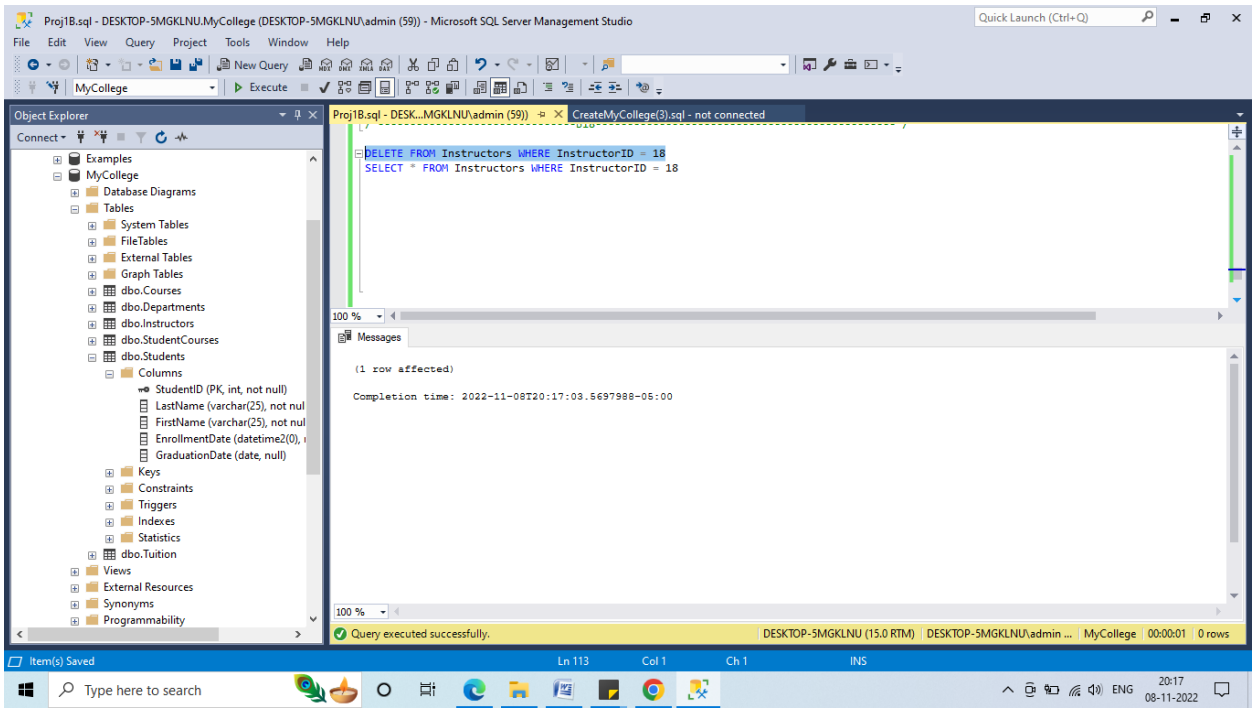
17. UPDATE AnnualSalary column from 34,000 to 35,0000 using InstructorID

```
UPDATE Instructors SET AnnualSalary = 35000.00 WHERE InstructorID = 17
SELECT * FROM Instructors WHERE InstructorID = 17
```

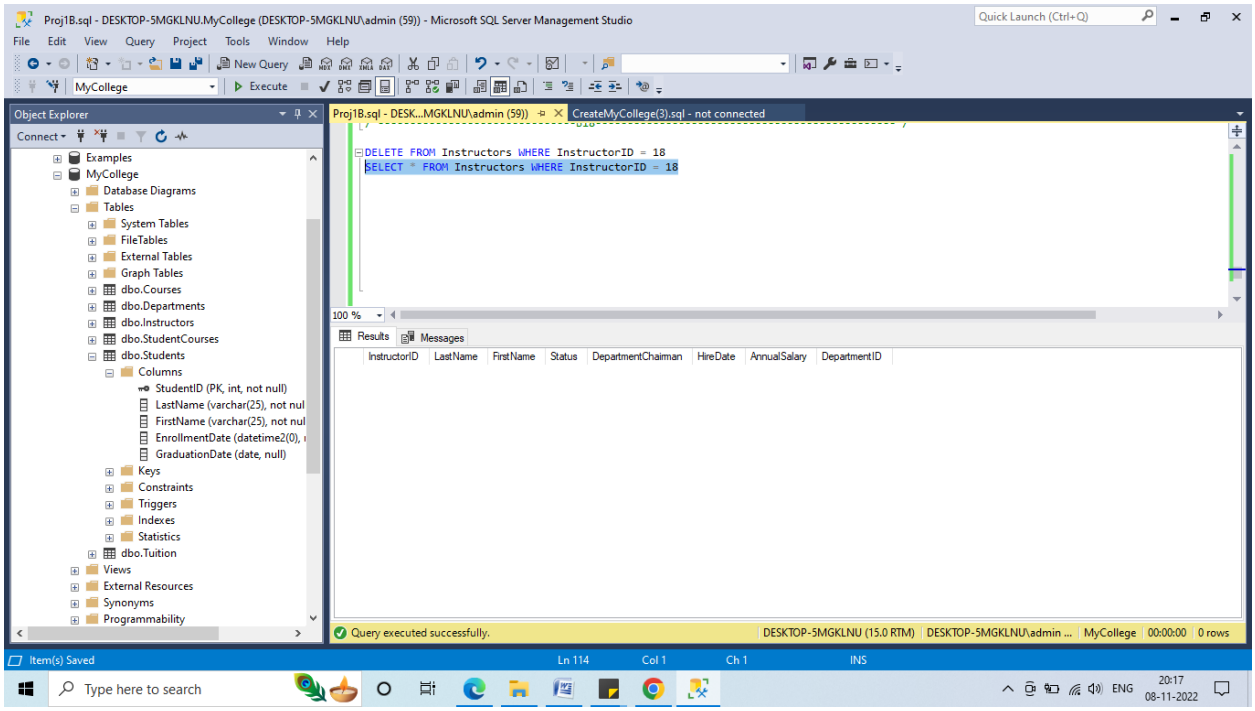


18. Write a DELETE row using InstructorID

```
DELETE FROM Instructors WHERE InstructorID = 18
```

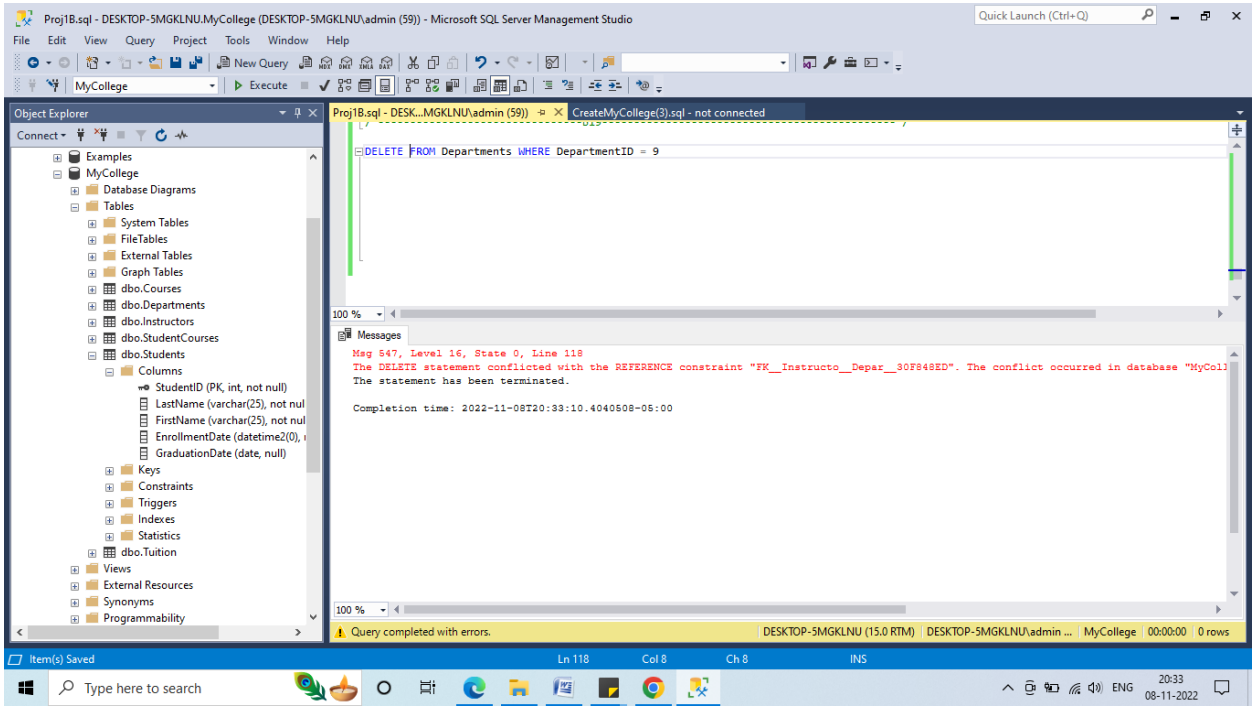


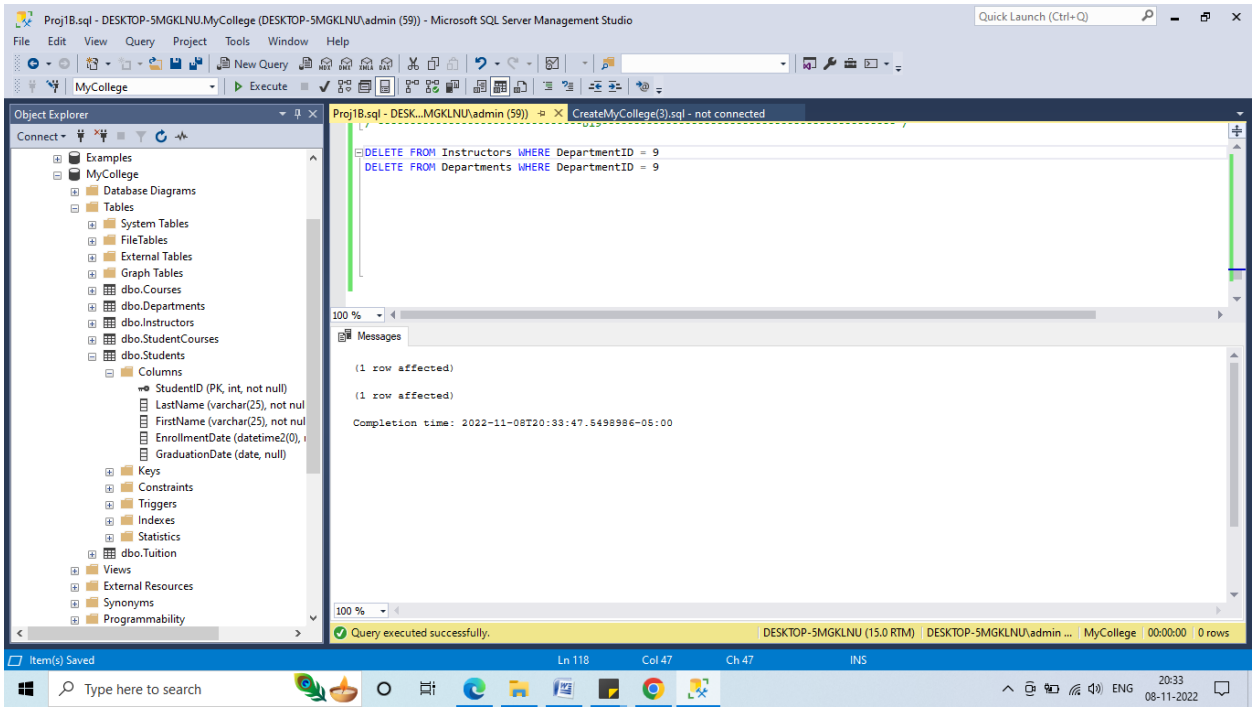

```
SELECT * FROM Instructors WHERE InstructorID = 18
```



19. DELETE the row in the Departments table that has an ID of 9.

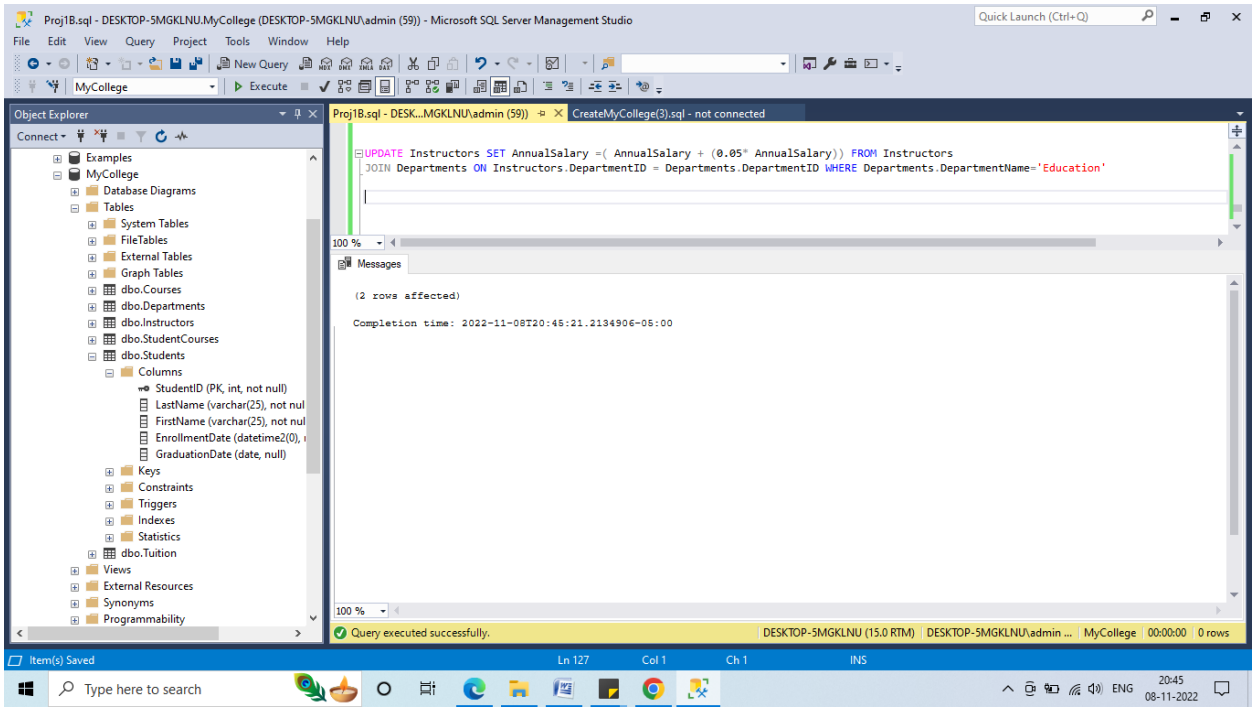
```
DELETE FROM Instructors WHERE DepartmentID = 9
DELETE FROM Departments WHERE DepartmentID = 9
```





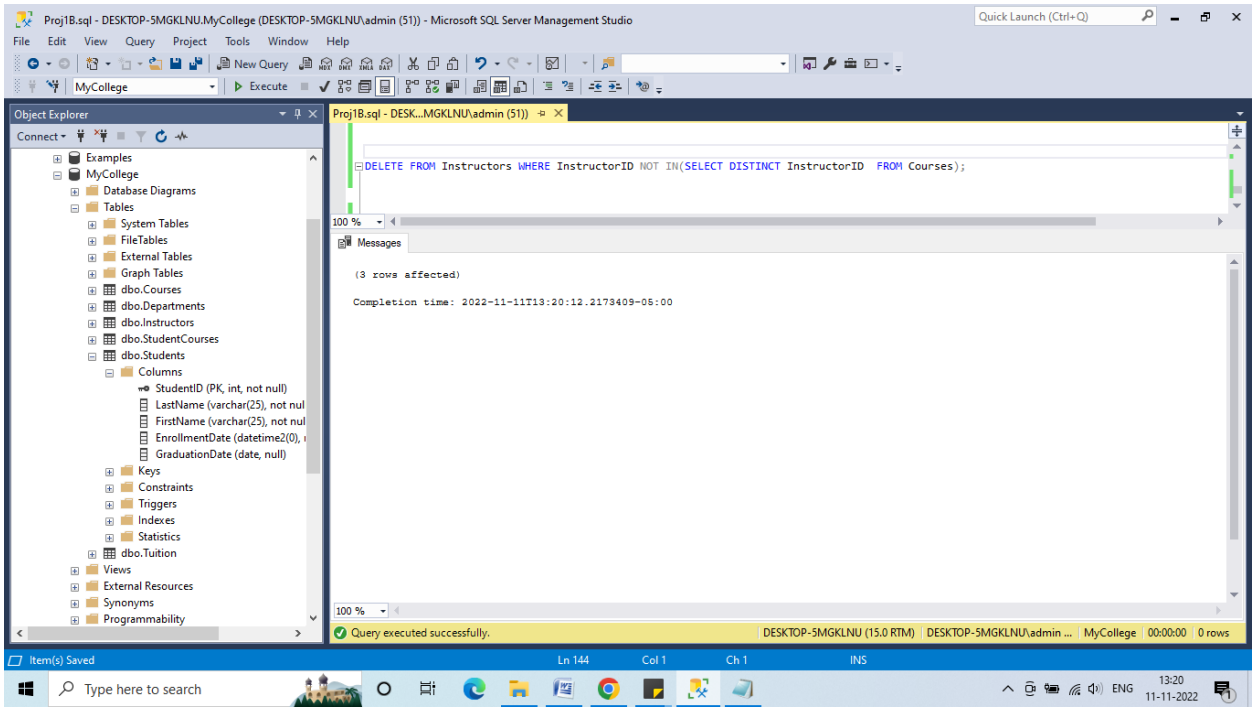
20. UPDATE annual salary for all instructors in the Education department by 5%.

```
UPDATE Instructors SET AnnualSalary =( AnnualSalary + (0.05* AnnualSalary)) FROM
Instructors
JOIN Departments ON Instructors.DepartmentID = Departments.DepartmentID WHERE
Departments.DepartmentName='Education'
```

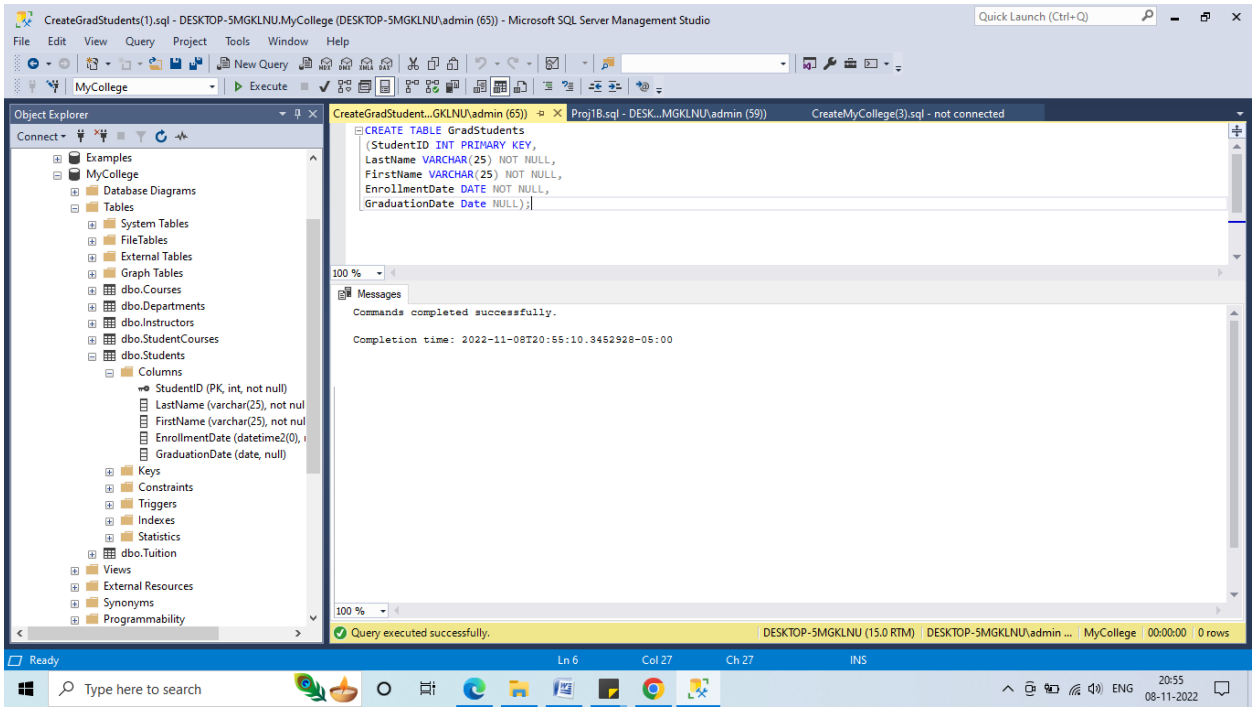


21. Delete instructors that aren't teaching any courses.

```
DELETE FROM Instructors WHERE InstructorID NOT IN(SELECT DISTINCT InstructorID
FROM Courses);
```

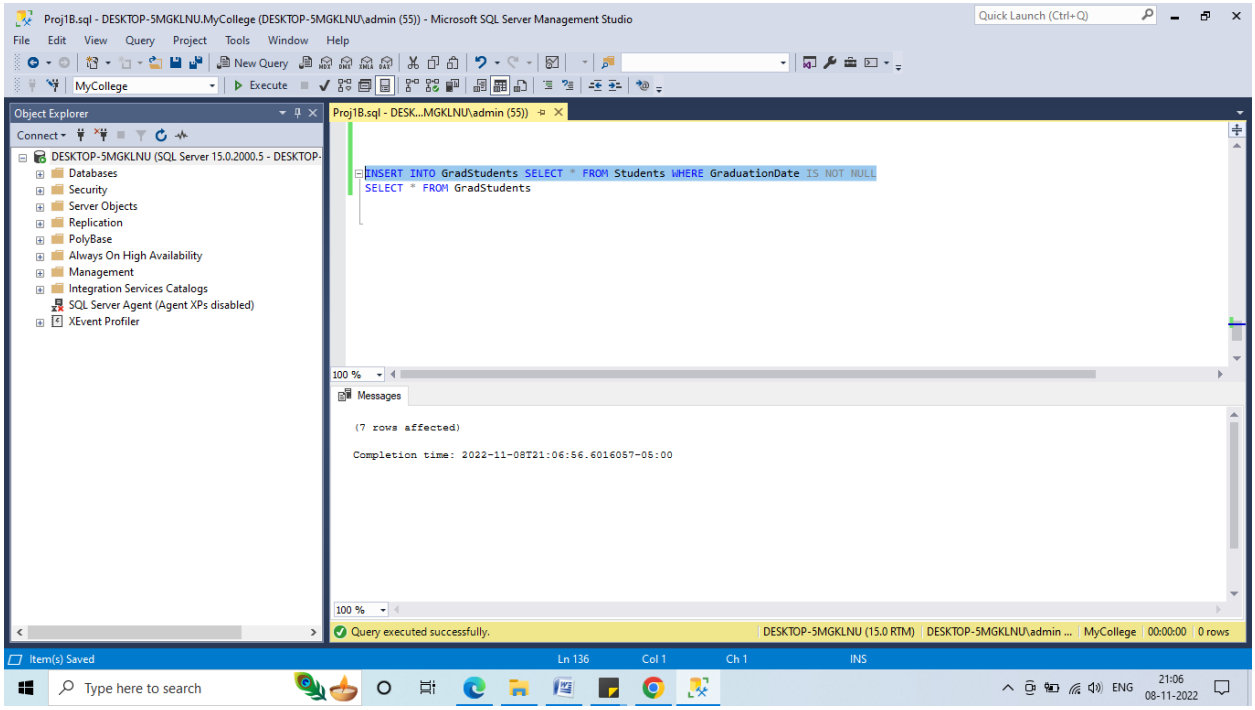


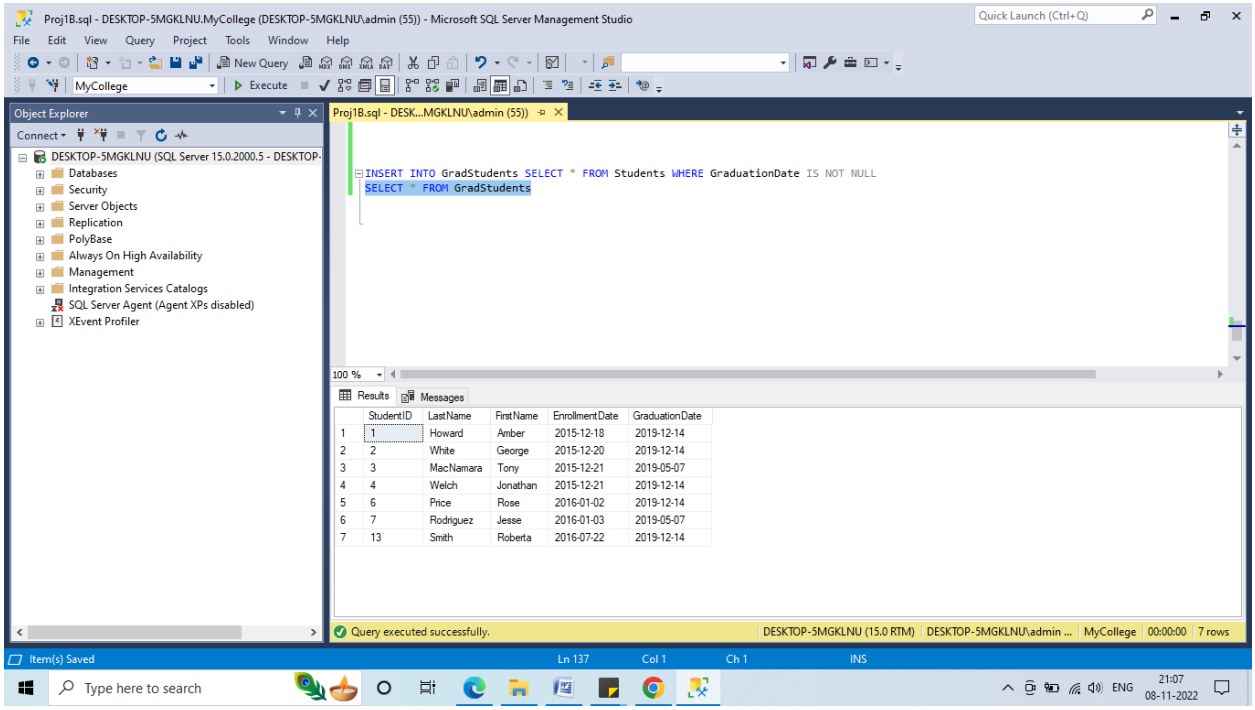
22. Open the script named CreateGradStudents.sql that's attached above. Run this file to create a table named GradStudents.



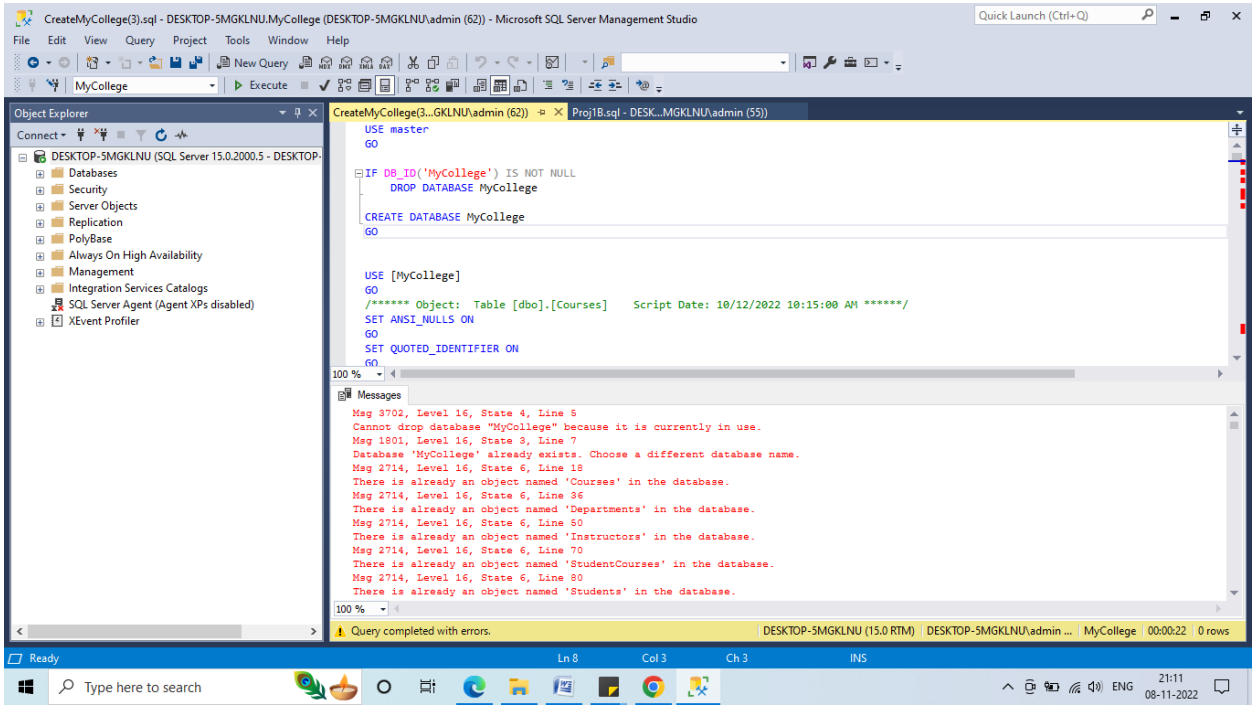
23. Insert rows from Students table GradStudents table Include only rows of graduated students

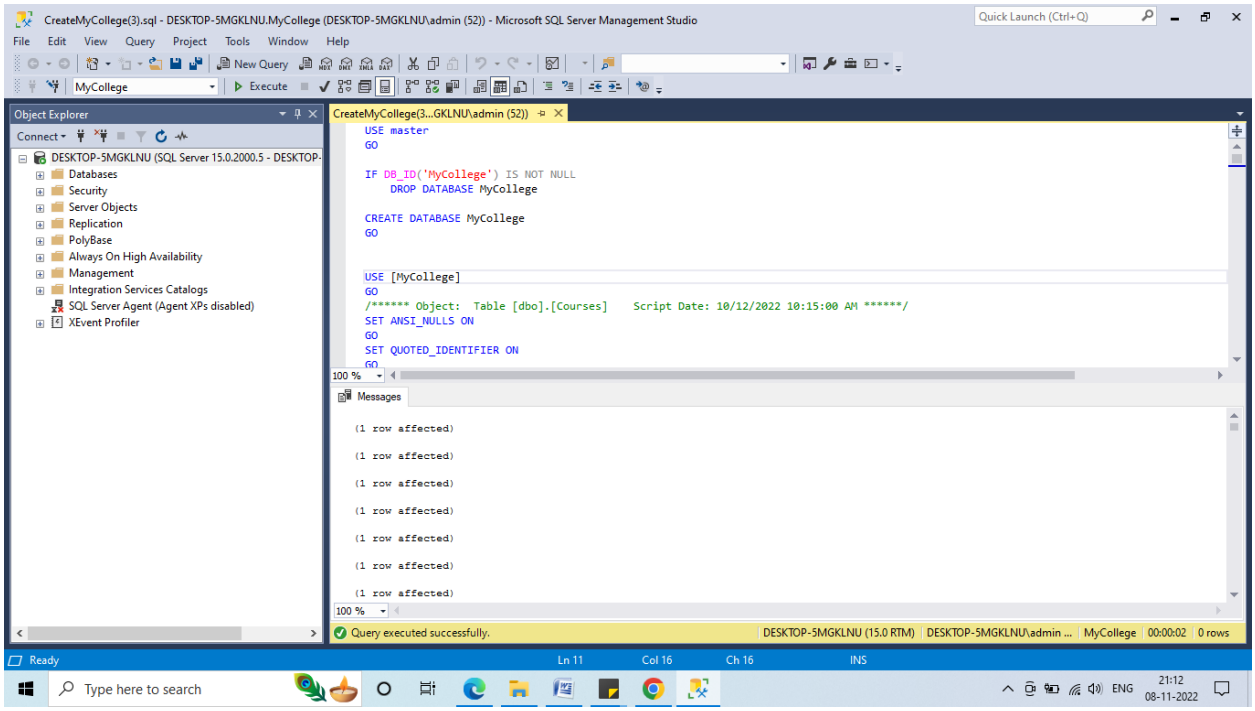
```
INSERT INTO GradStudents SELECT * FROM Students WHERE GraduationDate IS NOT NULL
SELECT * FROM GradStudents
```





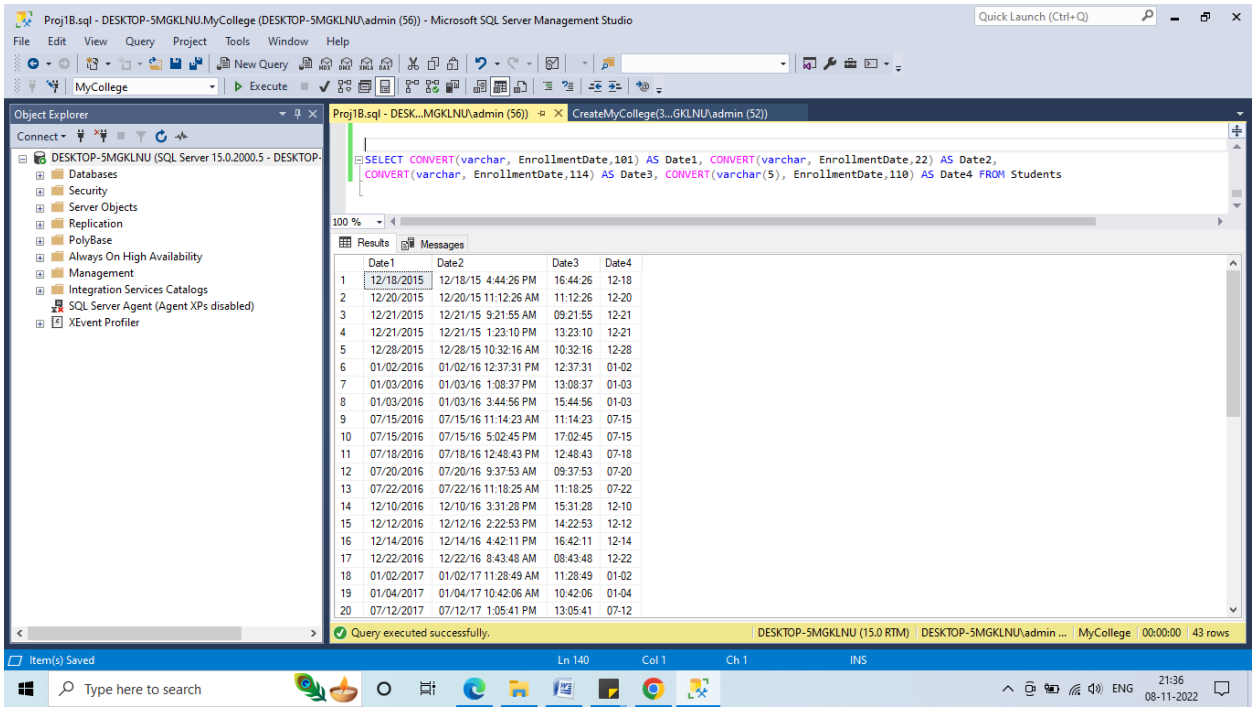
24. Restore database





25. Display columns using CONVERT functions

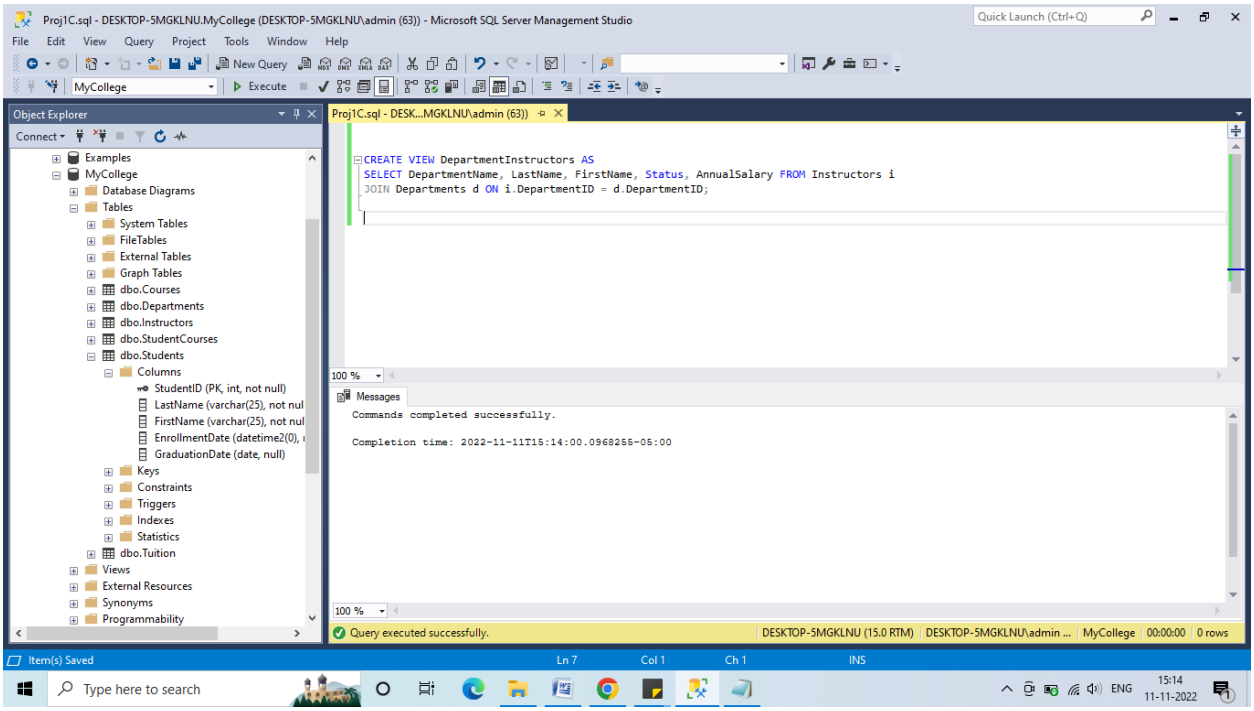
```
SELECT CONVERT(varchar, EnrollmentDate,101) AS Date1, CONVERT(varchar,
EnrollmentDate,22) AS Date2,
CONVERT(varchar, EnrollmentDate,114) AS Date3, CONVERT(varchar(5),
EnrollmentDate,110) AS Date4 FROM Students
```



Section C : Advanced SQL Skills (views/stores procedures/ functions / scripts)

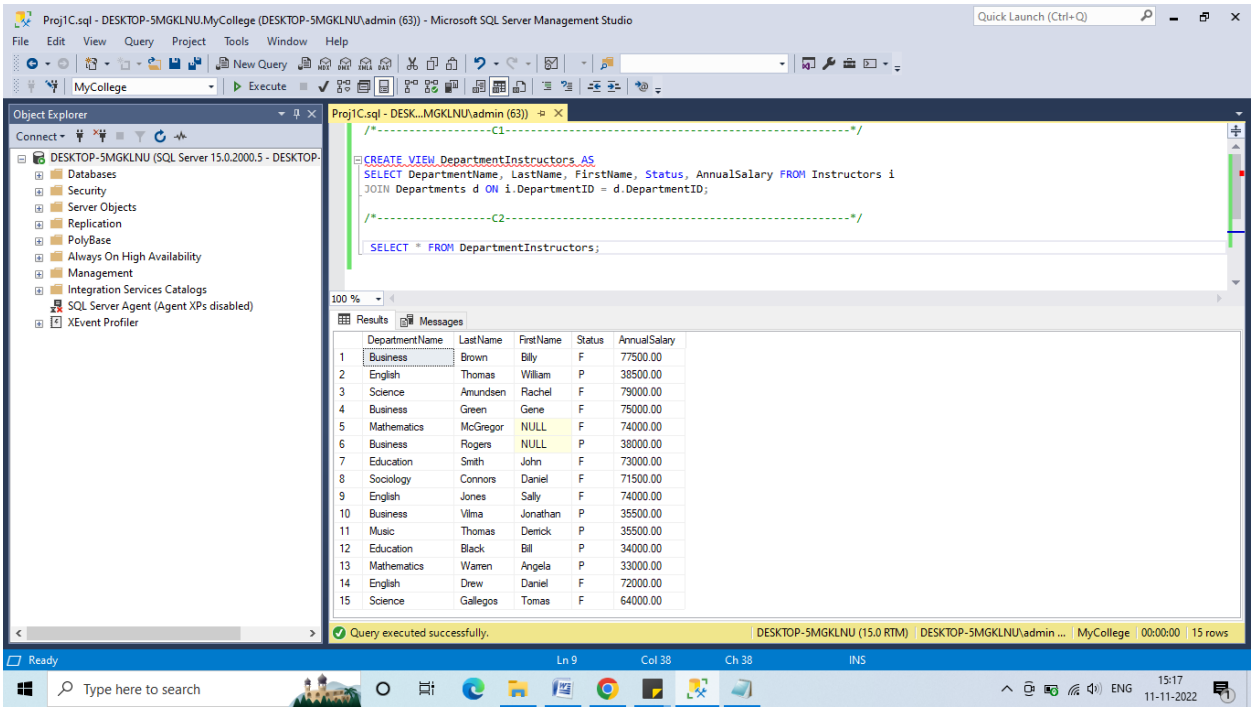
1. Create a view named DepartmentInstructors that returns the DepartmentName ,LastName, FirstName, Status, and AnnualSalary columns

```
CREATE VIEW DepartmentInstructors AS
SELECT DepartmentName, LastName, FirstName, Status, AnnualSalary FROM
Instructors i JOIN Departments d ON i.DepartmentID = d.DepartmentID;
```



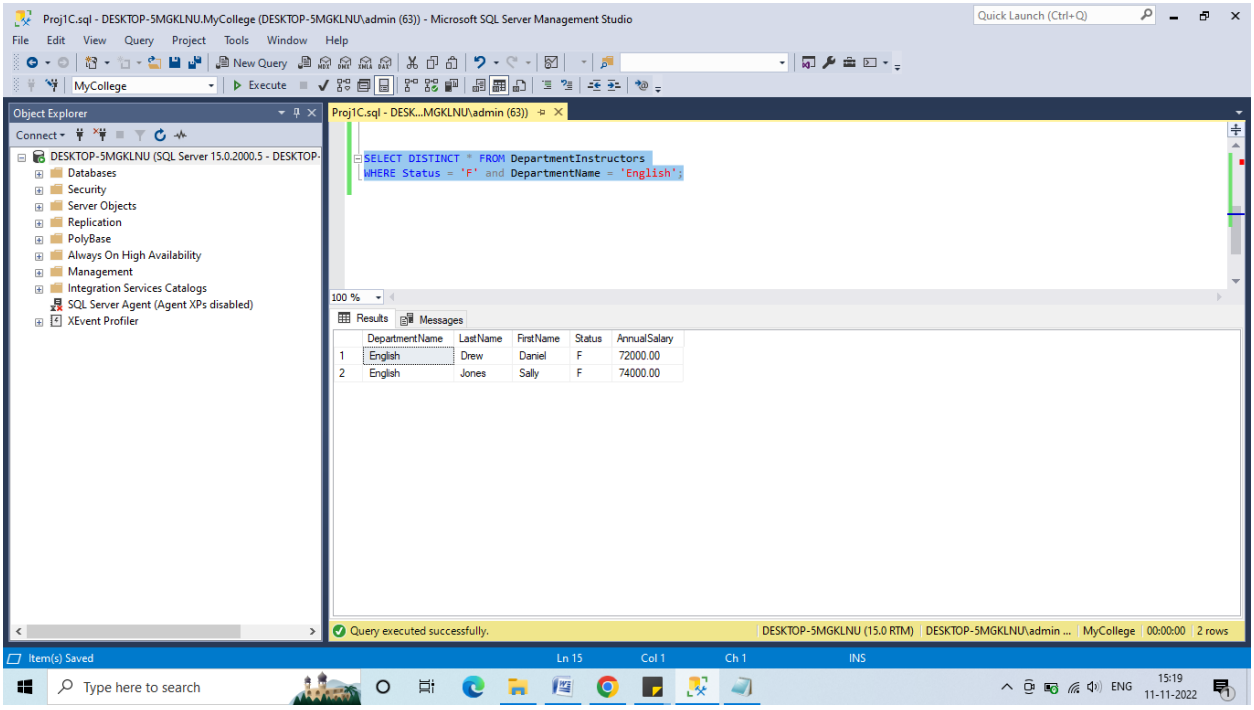
2. Display DepartmentInstructors view

```
SELECT * FROM DepartmentInstructors;
```



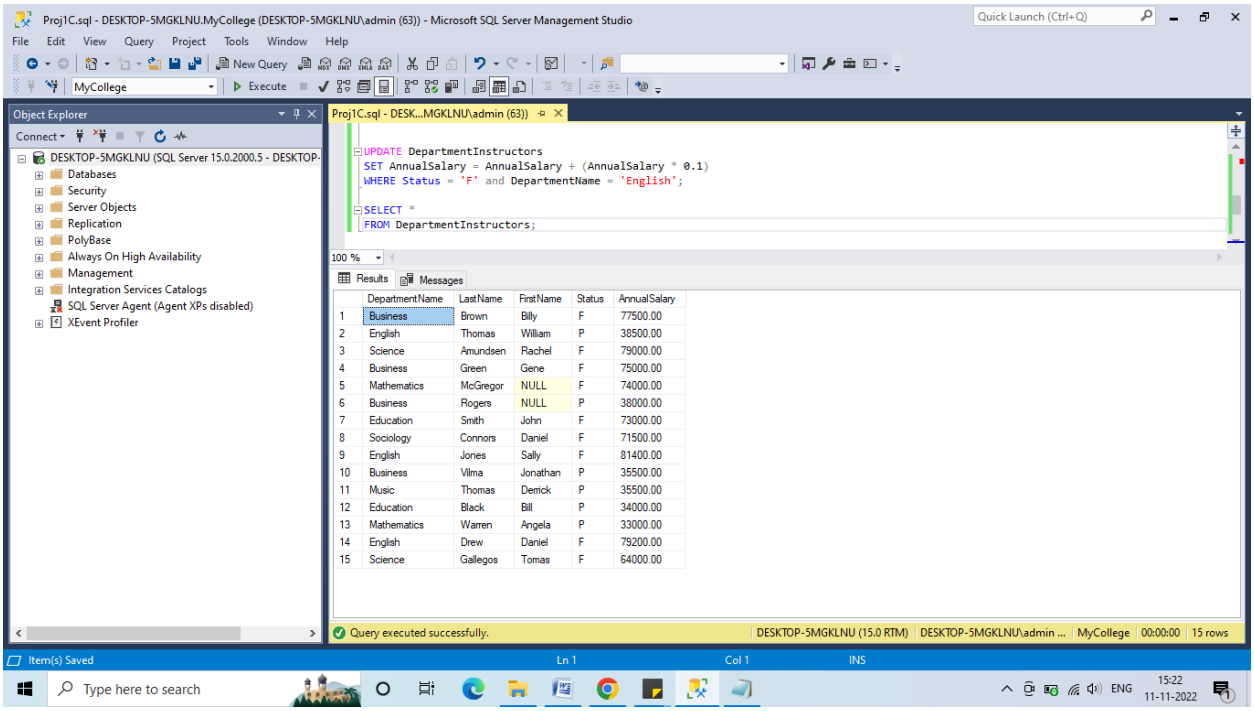
3. Return one row for each fulltime instructor in the English department

```
SELECT DISTINCT * FROM DepartmentInstructors
WHERE Status = 'F' and DepartmentName = 'English';
```



4. Update the annual salary for each fulltime instructor in the English department by 10%

```
UPDATE DepartmentInstructors
SET AnnualSalary = AnnualSalary + (AnnualSalary * 0.1)
WHERE Status = 'F' and DepartmentName = 'English';
SELECT * FROM DepartmentInstructors;
```

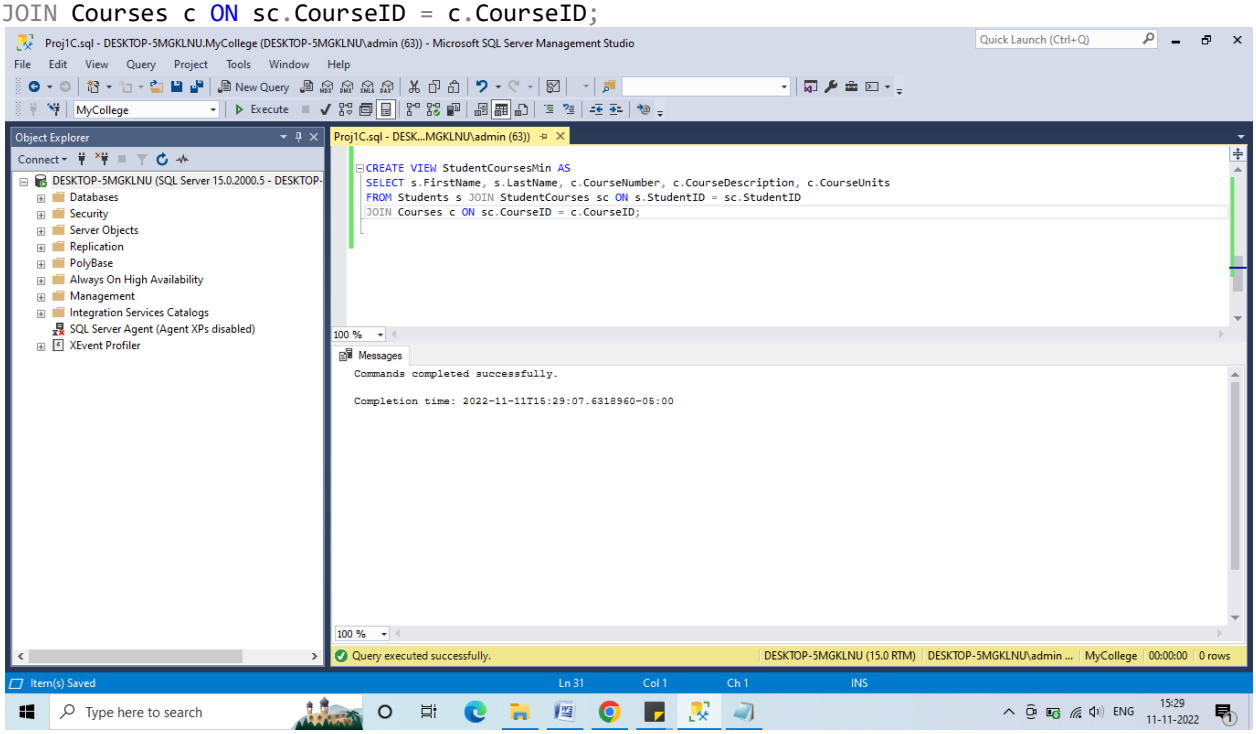


	DepartmentName	LastName	FirstName	Status	AnnualSalary
1	Business	Brown	Billy	F	77500.00
2	English	Thomas	William	P	38500.00
3	Science	Amundsen	Rachel	F	79000.00
4	Business	Green	Gene	F	75000.00
5	Mathematics	McGregor	NULL	F	74000.00
6	Business	Rogers	NULL	P	38000.00
7	Education	Smith	John	F	73000.00
8	Sociology	Connors	Daniel	F	71500.00
9	English	Jones	Sally	F	81400.00
10	Business	Vilma	Jonathan	P	35500.00
11	Music	Thomas	Derrick	P	35500.00
12	Education	Black	Bill	P	34000.00
13	Mathematics	Warren	Angela	P	33000.00
14	English	Drew	Daniel	F	79200.00
15	Science	Gallegos	Tomas	F	64000.00

5. Display FirstName and LastName ,CourseNumber, CourseDescription, and CourseUnits

```
CREATE VIEW StudentCoursesMin AS
SELECT s.FirstName, s.LastName, c.CourseNumber, c.CourseDescription,
c.CourseUnits
FROM Students s JOIN StudentCourses sc ON s.StudentID = sc.StudentID
```

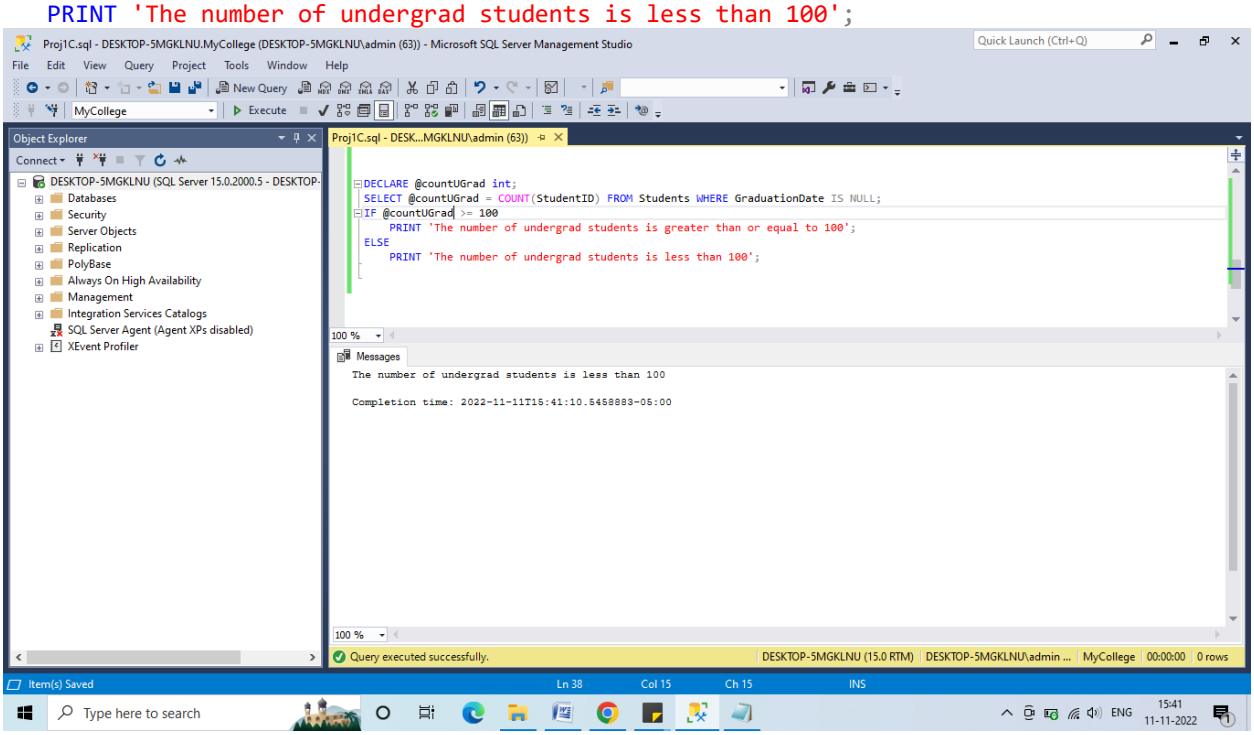
```
JOIN Courses c ON sc.CourseID = c.CourseID;
```



6. Display number of grad students using scripts

```
DECLARE @countUGrad int;
SELECT @countUGrad = COUNT(StudentID) FROM Students WHERE GraduationDate IS
NULL;
IF @countUGrad >= 100
    PRINT 'The number of undergrad students is greater than or equal to 100';
ELSE
```

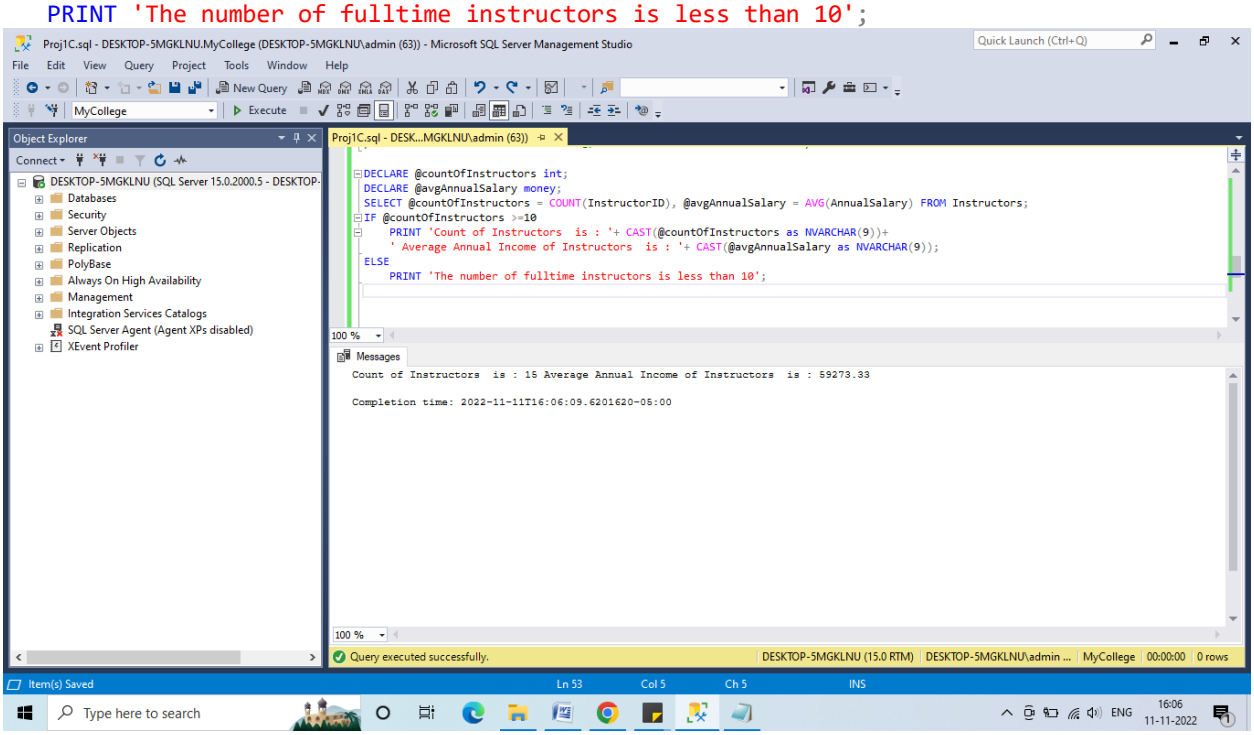
```
    PRINT 'The number of undergrad students is less than 100';
```



7. Count of instructors and their average salary

```
DECLARE @countOfInstructors int;
DECLARE @avgAnnualSalary money;
SELECT @countOfInstructors = COUNT(InstructorID), @avgAnnualSalary =
AVG(AnnualSalary) FROM Instructors;
IF @countOfInstructors >=10
    PRINT 'Count of Instructors  is : '+ CAST(@countOfInstructors as
NVARCHAR(9))+
    ' Average Annual Income of Instructors  is : '+ CAST(@avgAnnualSalary as
NVARCHAR(9));
ELSE
```

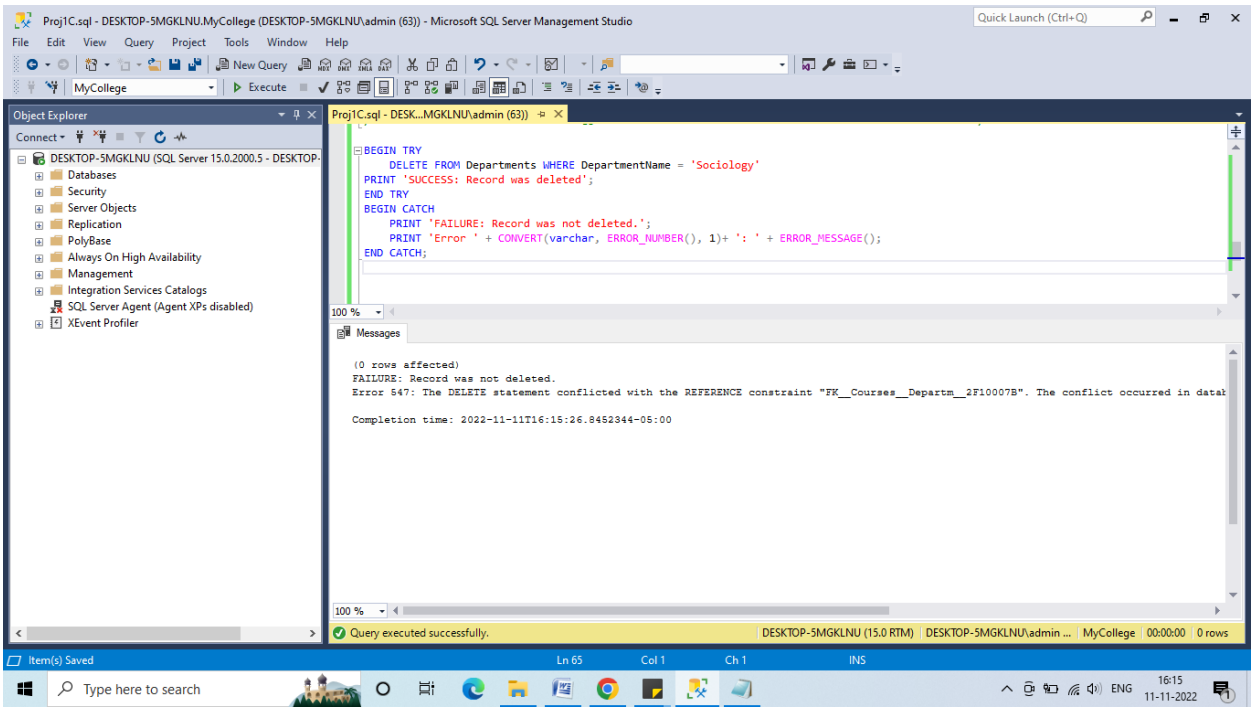
```
PRINT 'The number of fulltime instructors is less than 10';
```



8. Delete the department with the name ‘Sociology’ using scripts

```
BEGIN TRY
DELETE FROM Departments WHERE DepartmentName = 'Sociology'
PRINT 'SUCCESS: Record was deleted';
END TRY
BEGIN CATCH
PRINT 'FAILURE: Record was not deleted.';
PRINT 'Error ' + CONVERT(varchar, ERROR_NUMBER(), 1)+ ': ' +
ERROR_MESSAGE();
```

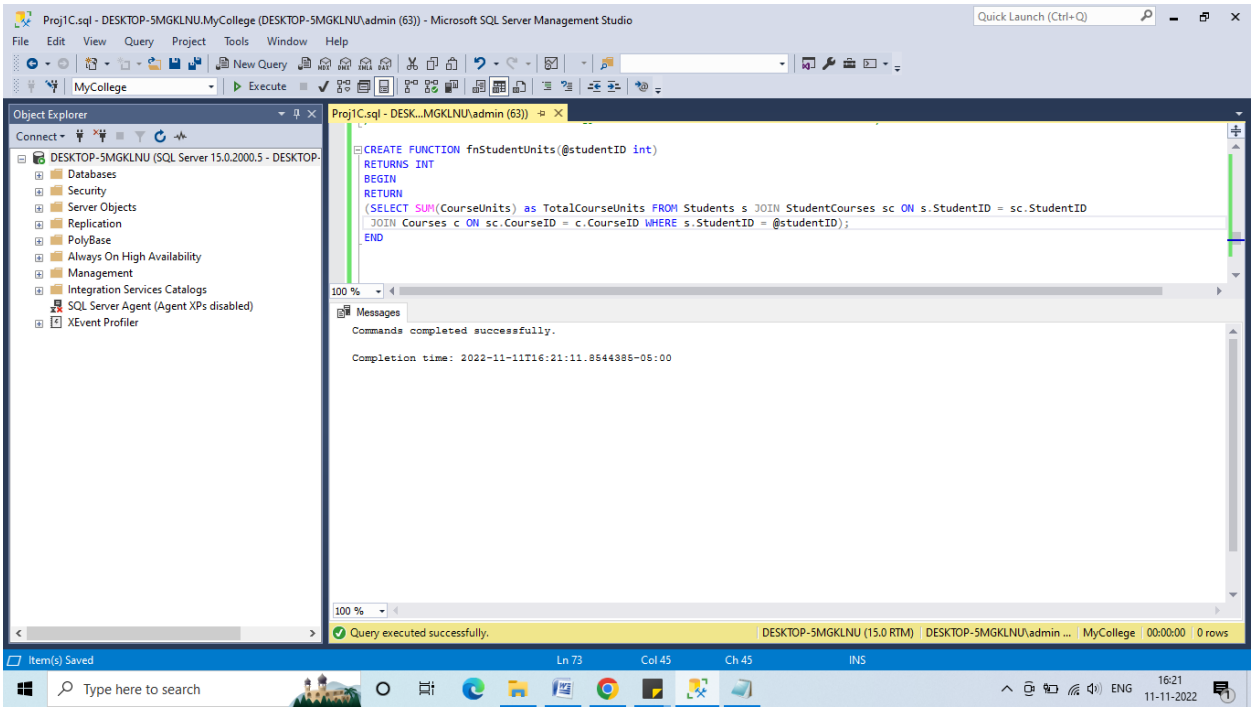
```
END CATCH;
```

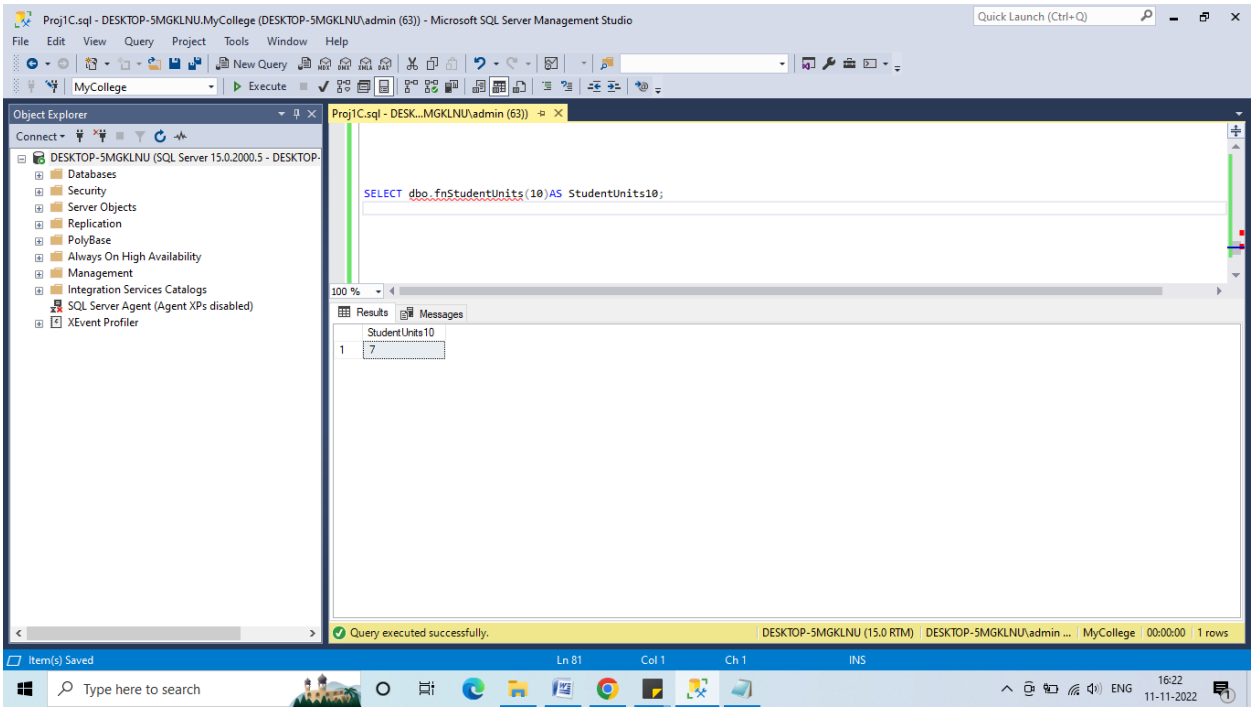


9. Total course units using scripts

```
CREATE FUNCTION fnStudentUnits(@studentID int)
RETURNS INT
BEGIN
RETURN
(SELECT sc.StudentID, SUM(CourseUnits) as TotalCourseUnits FROM Students s JOIN
StudentCourses sc ON s.StudentID = sc.StudentID
JOIN Courses c ON sc.CourseID = c.CourseID WHERE s.StudentID = @studentID GROUP
BY sc.StudentID);
```

```
END
```

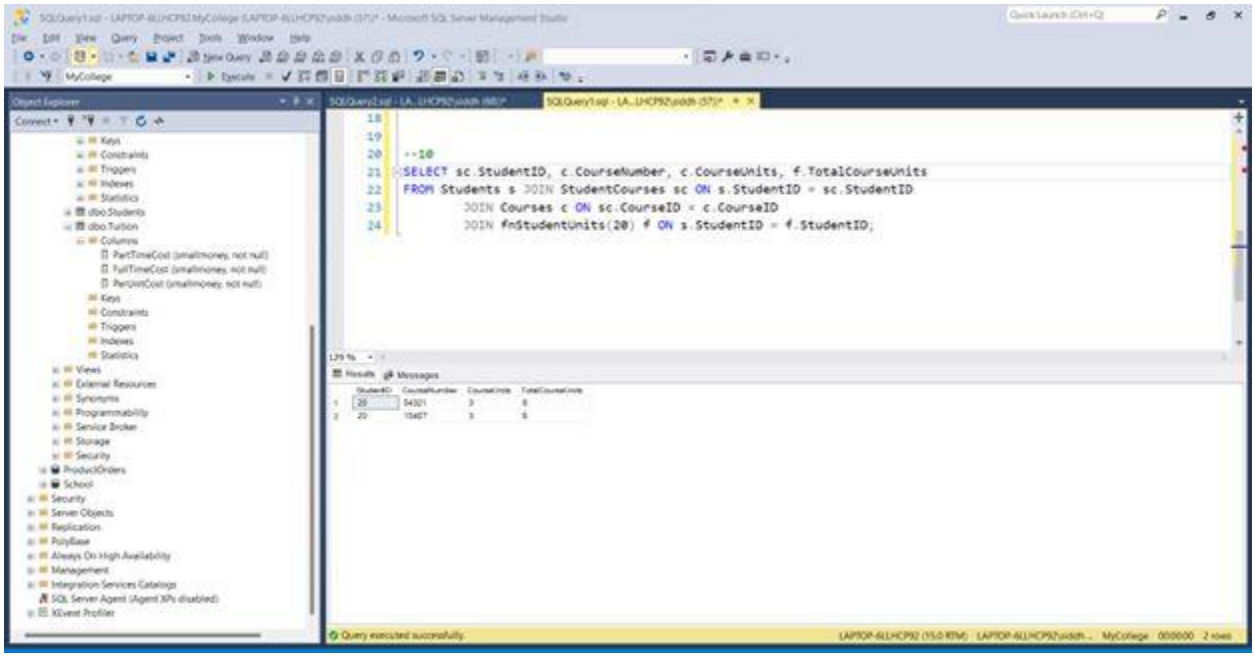




10. return the StudentID from the StudentCourses table, the CourseNumber and CourseUnits from the Courses table, and the value return by the fnStudentUnits function for that student

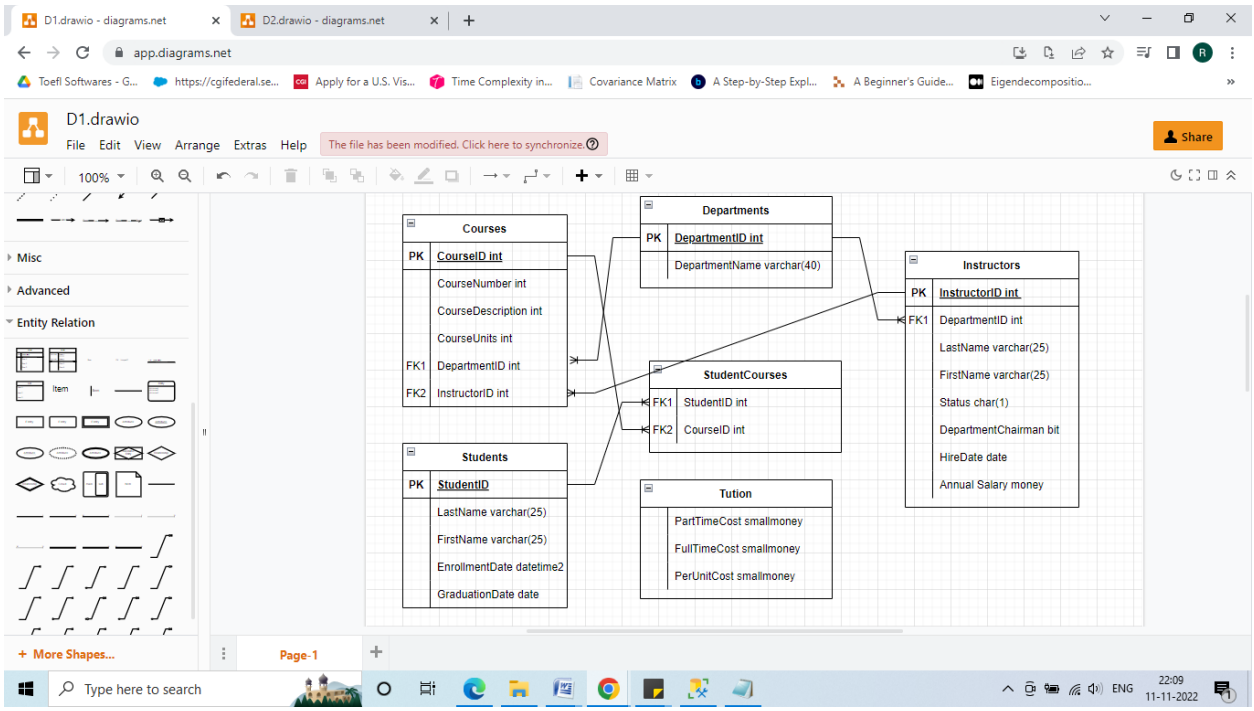
```
CREATE FUNCTION fnStudentUnits(@studentID int)
RETURNS TABLE
RETURN
```

```
(SELECT s.StudentID, SUM(CourseUnits) as TotalCourseUnits
 FROM Students s JOIN StudentCourses sc ON s.StudentID = sc.StudentID
JOIN Courses c ON sc.CourseID = c.CourseID
 WHERE s.StudentID = @studentID
 GROUP BY s.StudentID);
 SELECT * FROM fnStudentUnits(20);
 SELECT sc.StudentID, c.CourseNumber, c.CourseUnits, f.TotalCourseUnits
 FROM Students s JOIN StudentCourses sc ON s.StudentID = sc.StudentID
 JOIN Courses c ON sc.CourseID = c.CourseID
JOIN fnStudentUnits(20) f ON s.StudentID = f.StudentID;
```



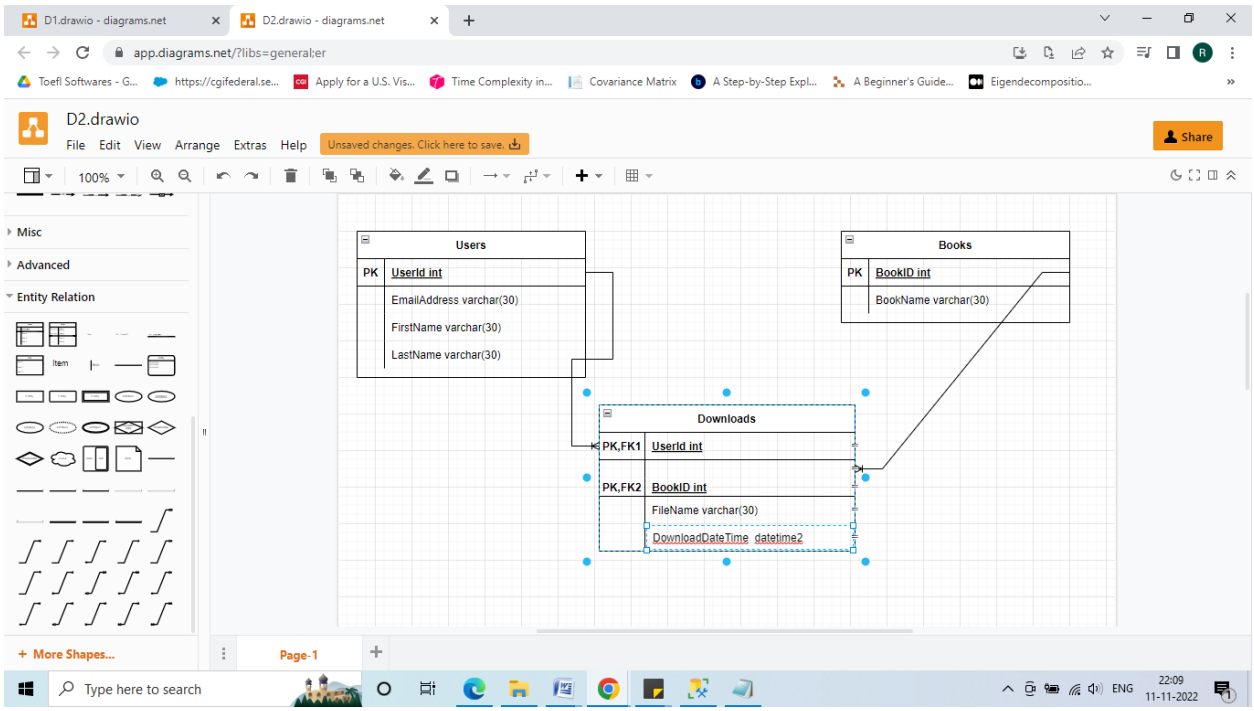
Database Design :

1.



In this database, there is a many to many relationship between Courses and Students table. To eliminate this, we defined a linking table named StudnetsCourses table. Course ID and Student ID are the primary keys of Courses and Students table respectively. One to many relationship between Departments and Courses table, one department can have many courses and many instructors. There is a one to many relationship between Student and StudentCourses table as a student can take many courses. One to many relationship between Instructor and Courses table as an instructor can take many courses.

2.



There will be three tables, Users, Books and Downloads. Users table have UserID as its primary key and BookId as primary key of Books table. There is a many to many relationship Users and Books table. One user can download many books and one book can be downloaded by many users. To eliminate this, we defined a linking table Downloads which will have composite key of UserId and BookId.

Remarks :

In this project, we did database setup, used DML statements .
Used views, scripts, stored procedures and functions on the tables in the database.
Also, we designed database diagrams based on the specifications given.