PROJECT - 1

ABSTRACT:

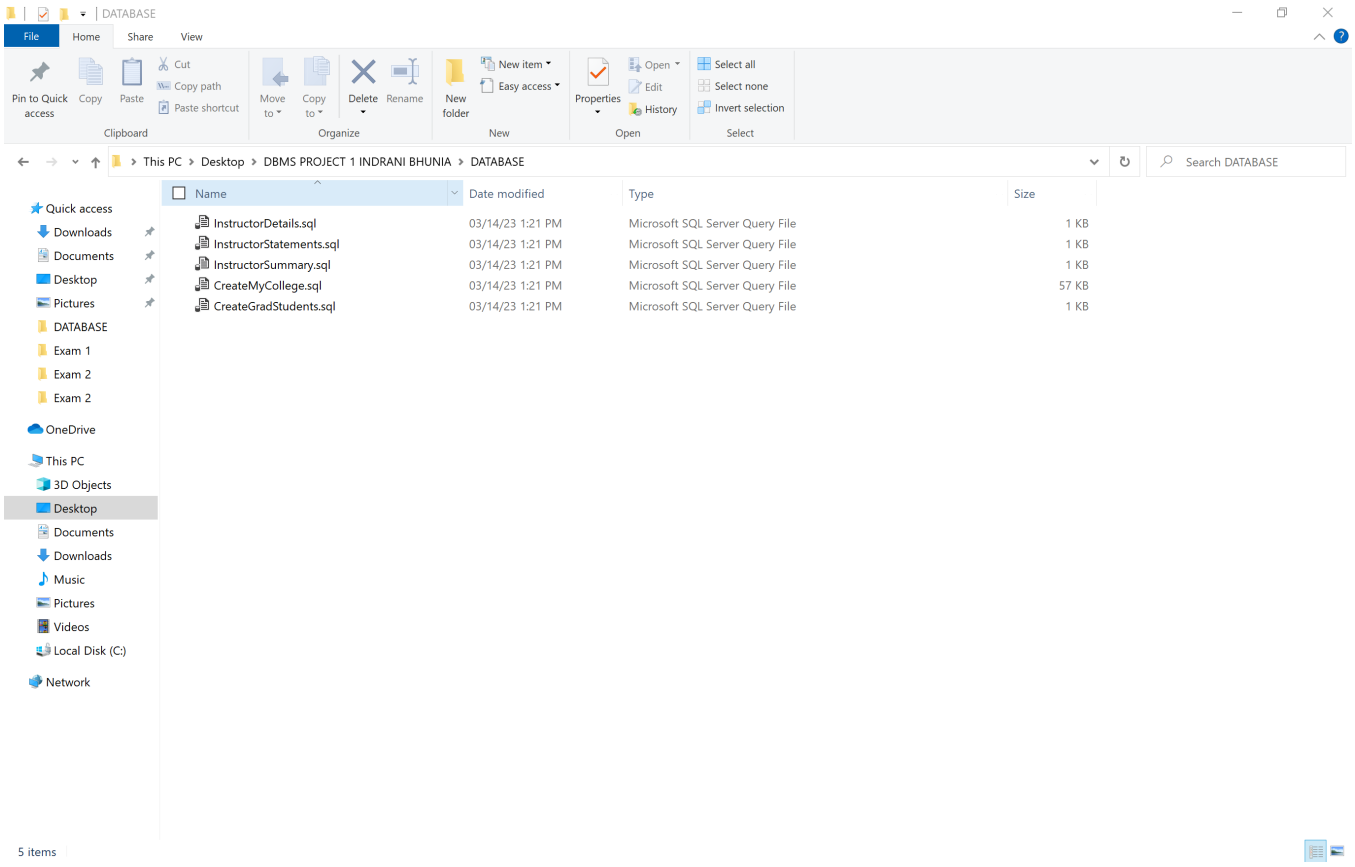
College Database

This project focuses on utilizing SQL Server Management Studio to run SQL queries in a college database's database management system. I ran all of the attached files in various query editors in the first section. SQL Joins, Union, Subqueries, and other operations are covered in the second section. In this part, simple DML queries like SELECT, UPDATE, INSERT, and DELETE are employed. It also includes several tasks that use DDL (Data Definition Language) queries. The questions in the following area are based on advanced SQL techniques and involve writing views, stored procedures, functions, and scripts. The final section, which deals with database design, contains two database designs that I created, one for a college database and the other for a website for a book. Additionally, this report includes comments on the project as well as some of the materials I used. I will establish the MyCollege database for this project, generate and enter SQL statements on it, then execute those statements.

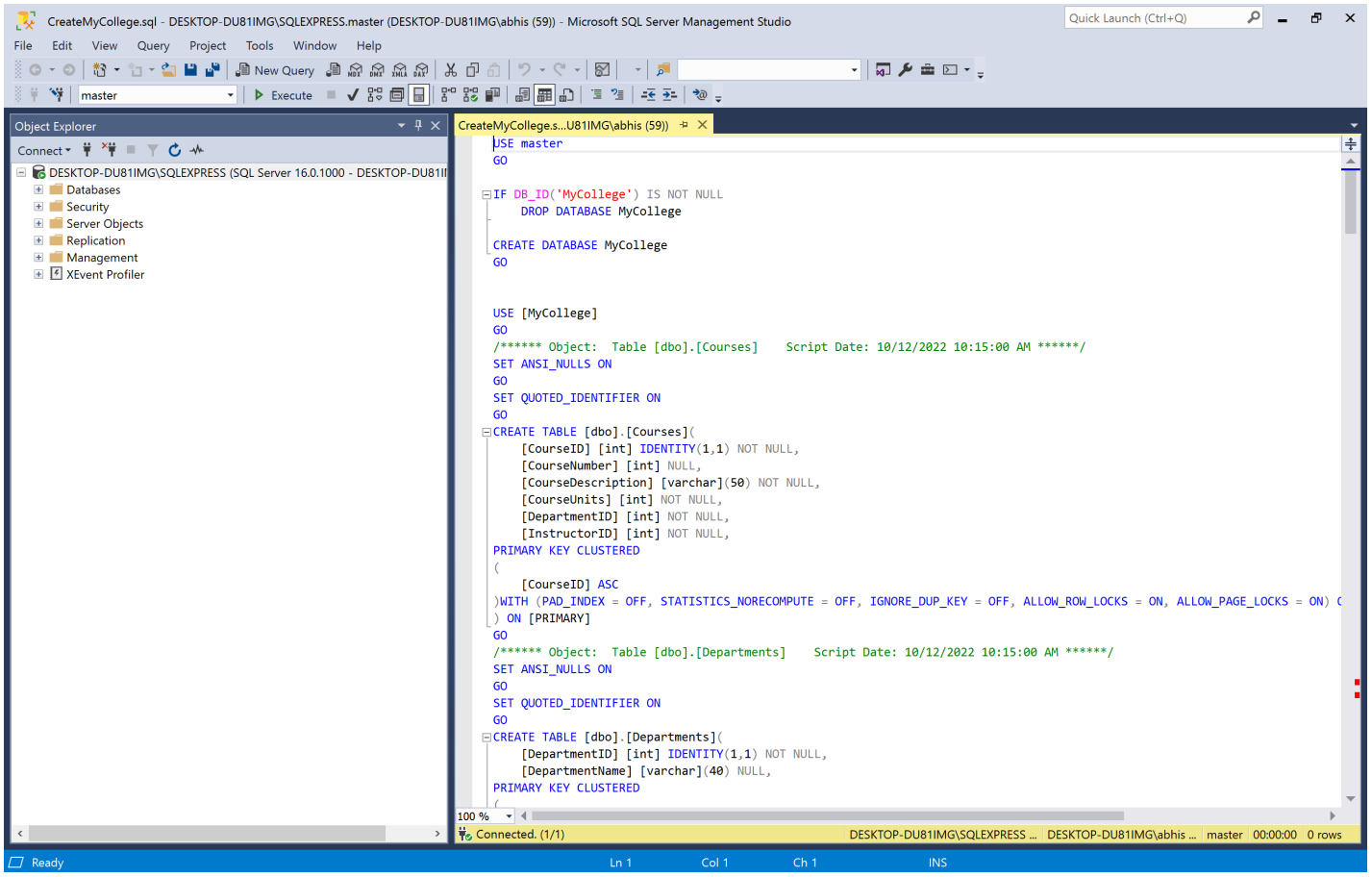
TABLE OF CONTENTS:

- **PART A, Database Setup -**

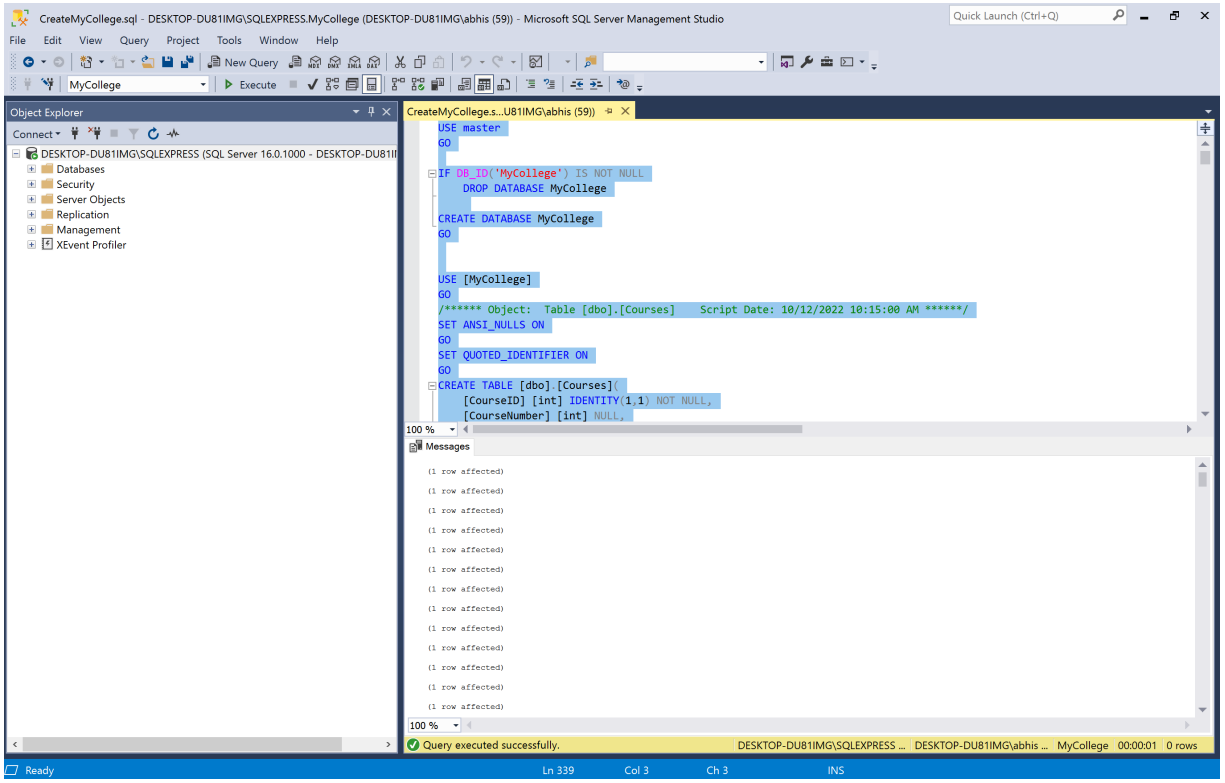
1. Download sql files from this project tab and open CreateMyCollege.sql in SQL server management studio. Execute the entire script and show the message in the Message tab, indicating the script is executed successfully.



Comment : Downloaded the SQL files required for the project

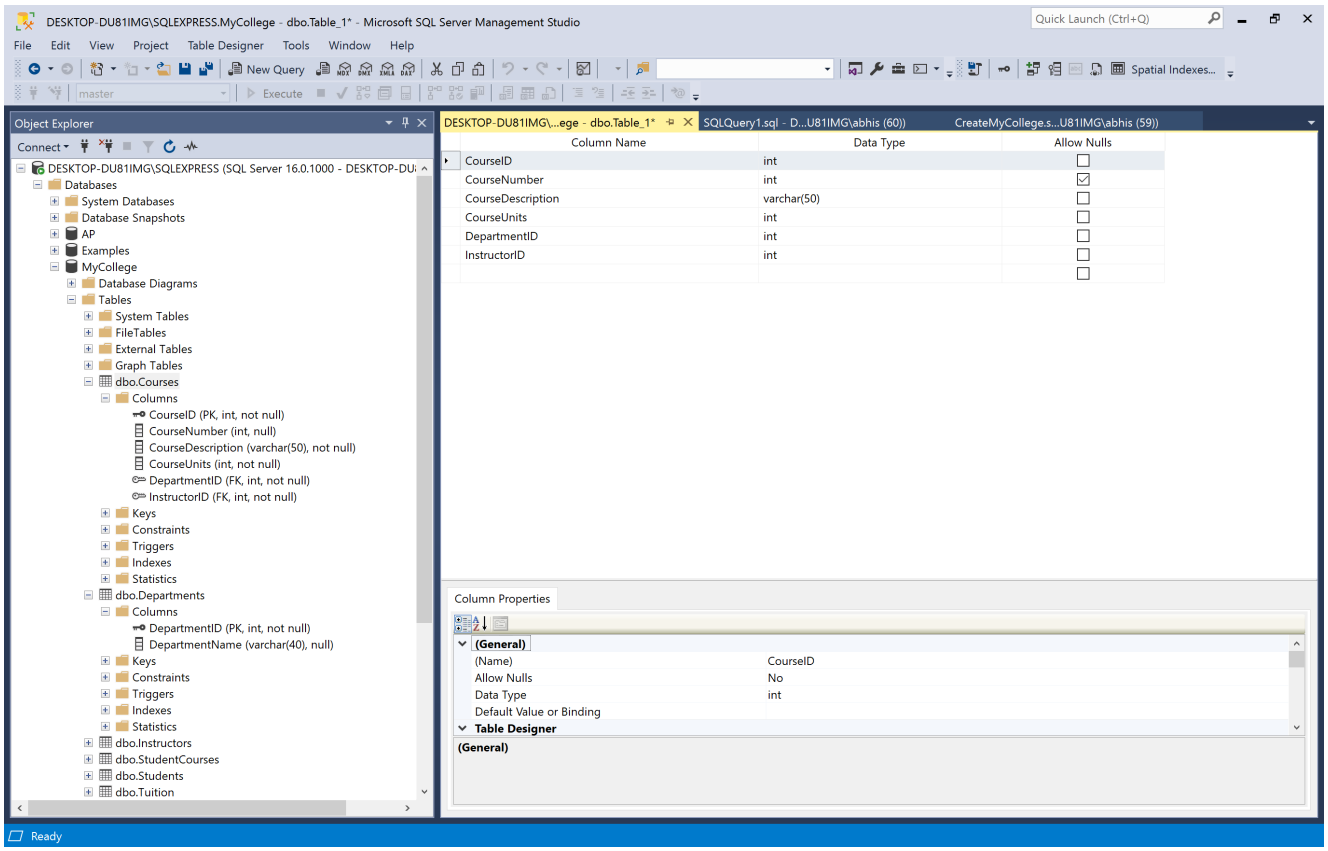


Comment :CreatMyCollege.sql

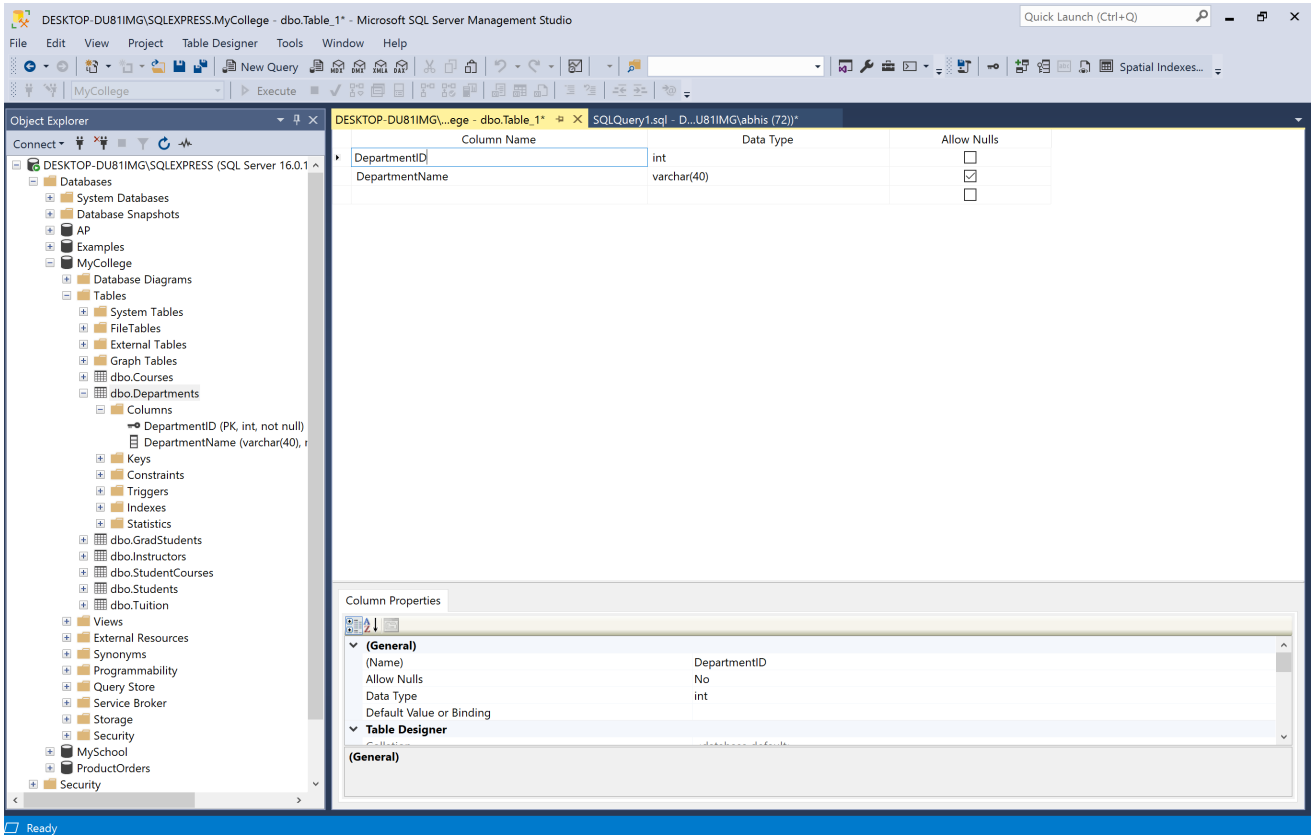


Comment : Entire script executed, with a message indicating successful script execution in the Message tab.

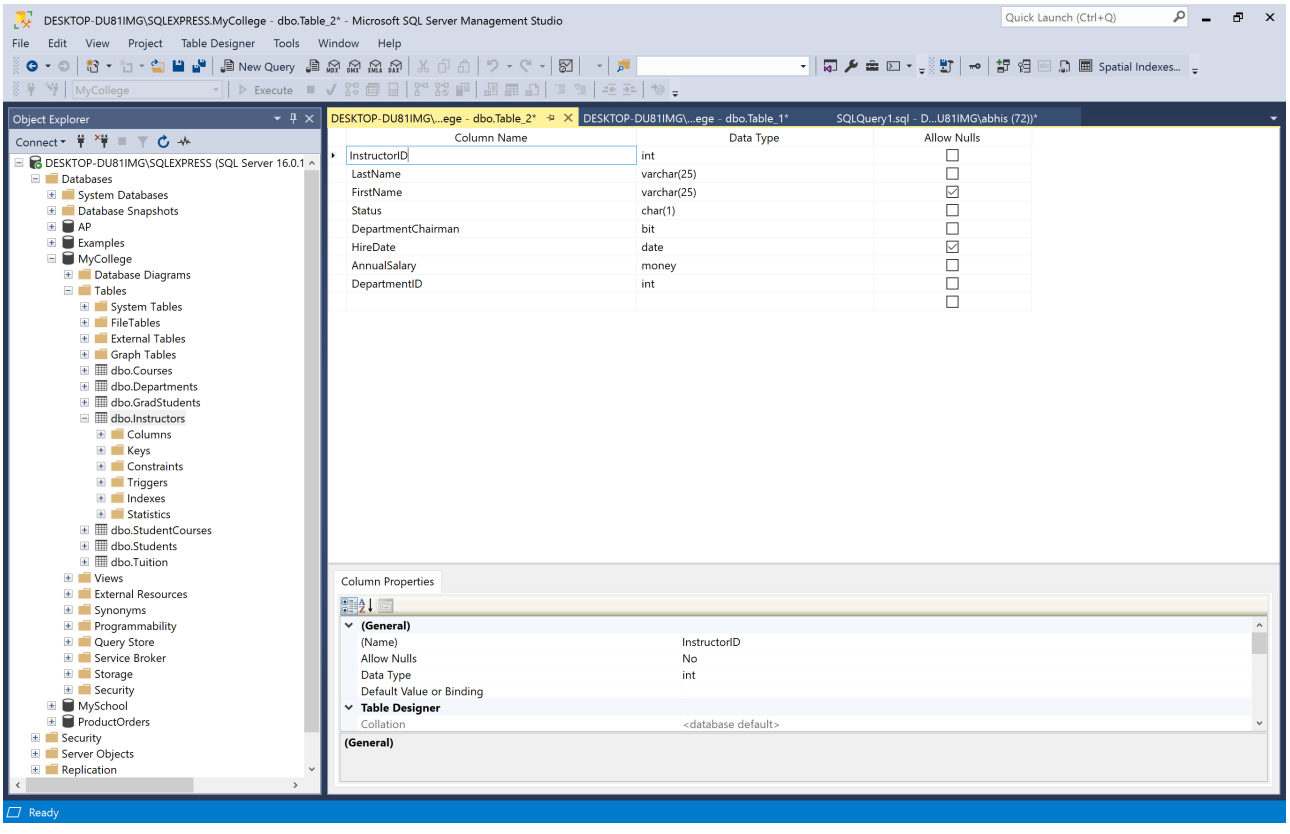
2. Do Navigate through the database objects and view the column definitions for each table. Open a new Query Editor window. Show details in Courses table and Instructors table using SELECT statement.



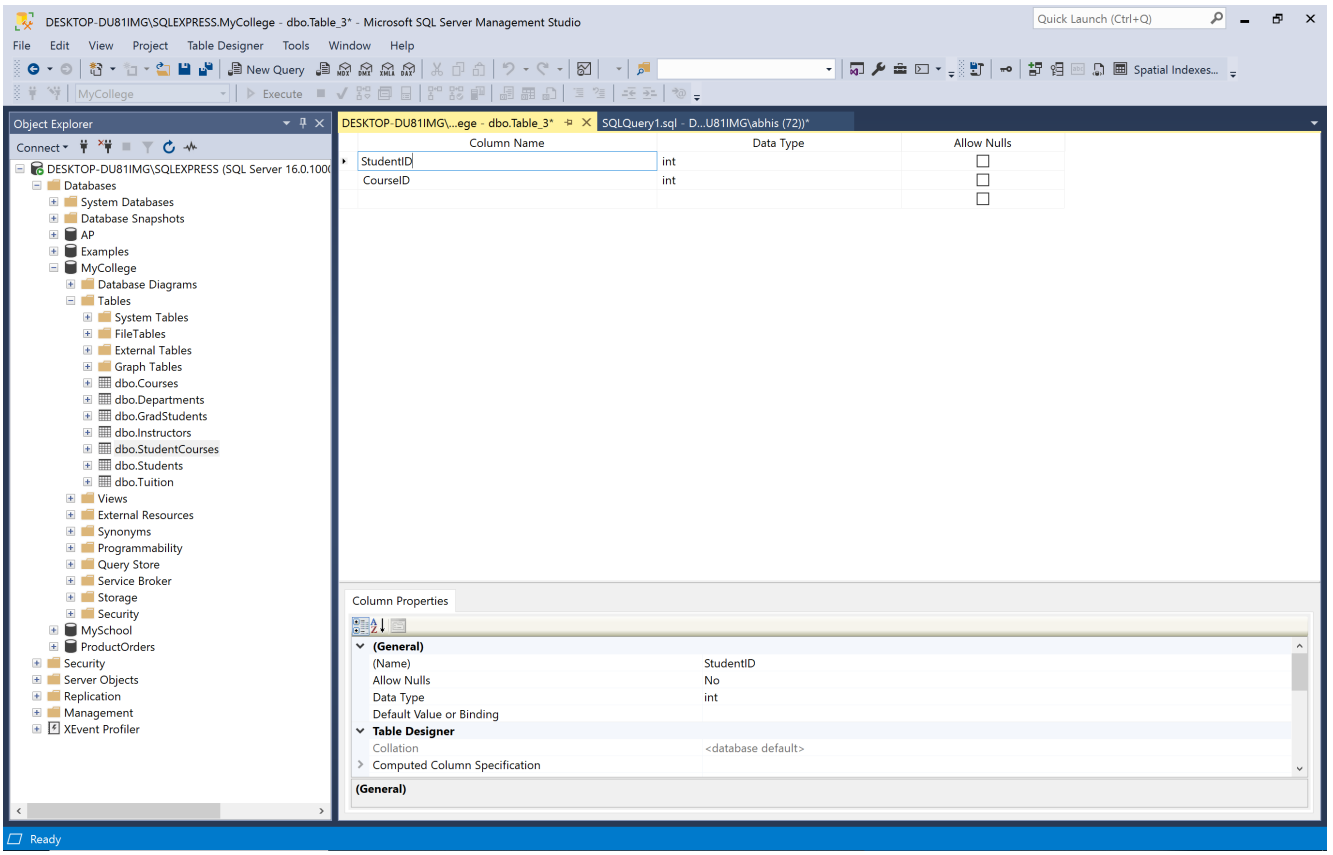
Comment :Column definition of table – ‘Courses’



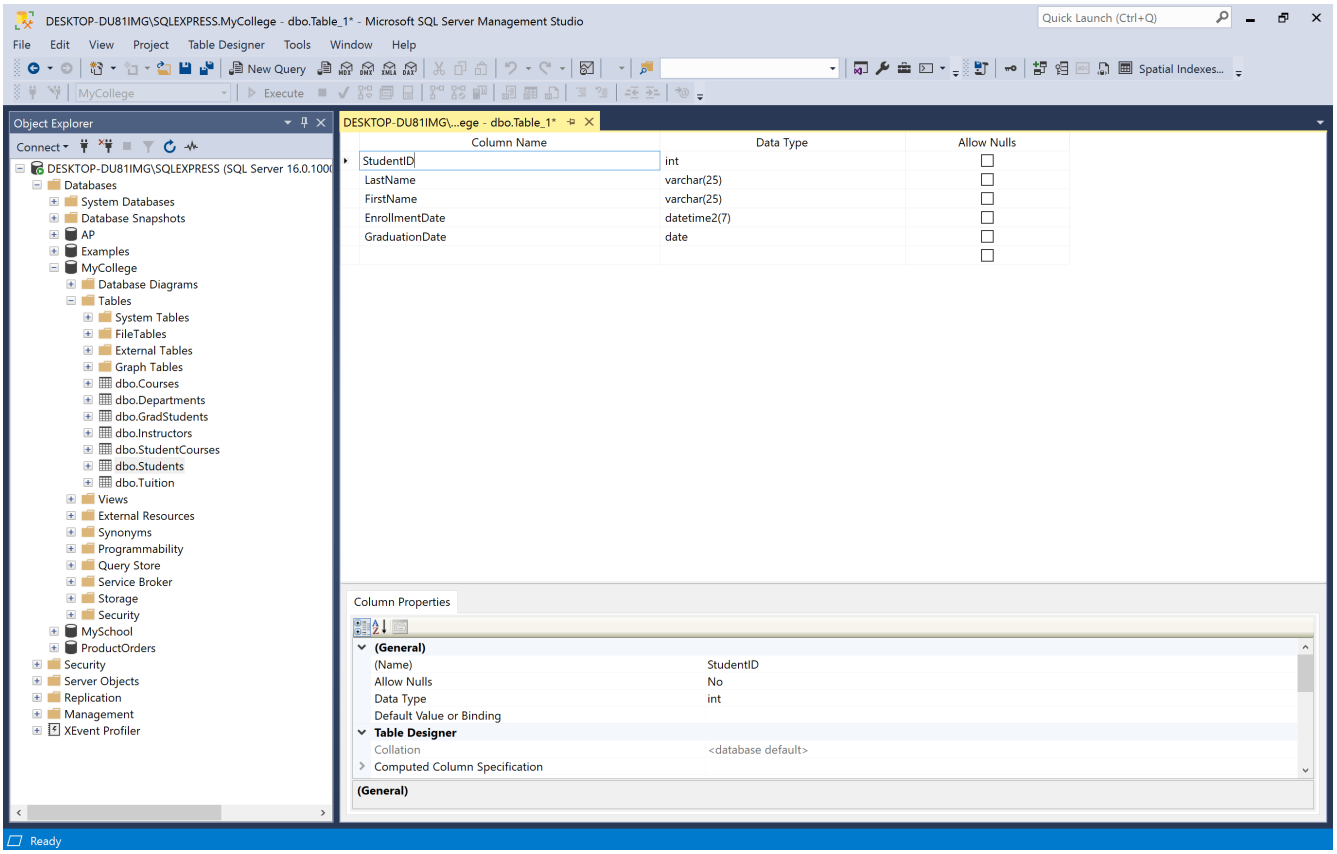
Comment :Column definition of table – ‘Departments’



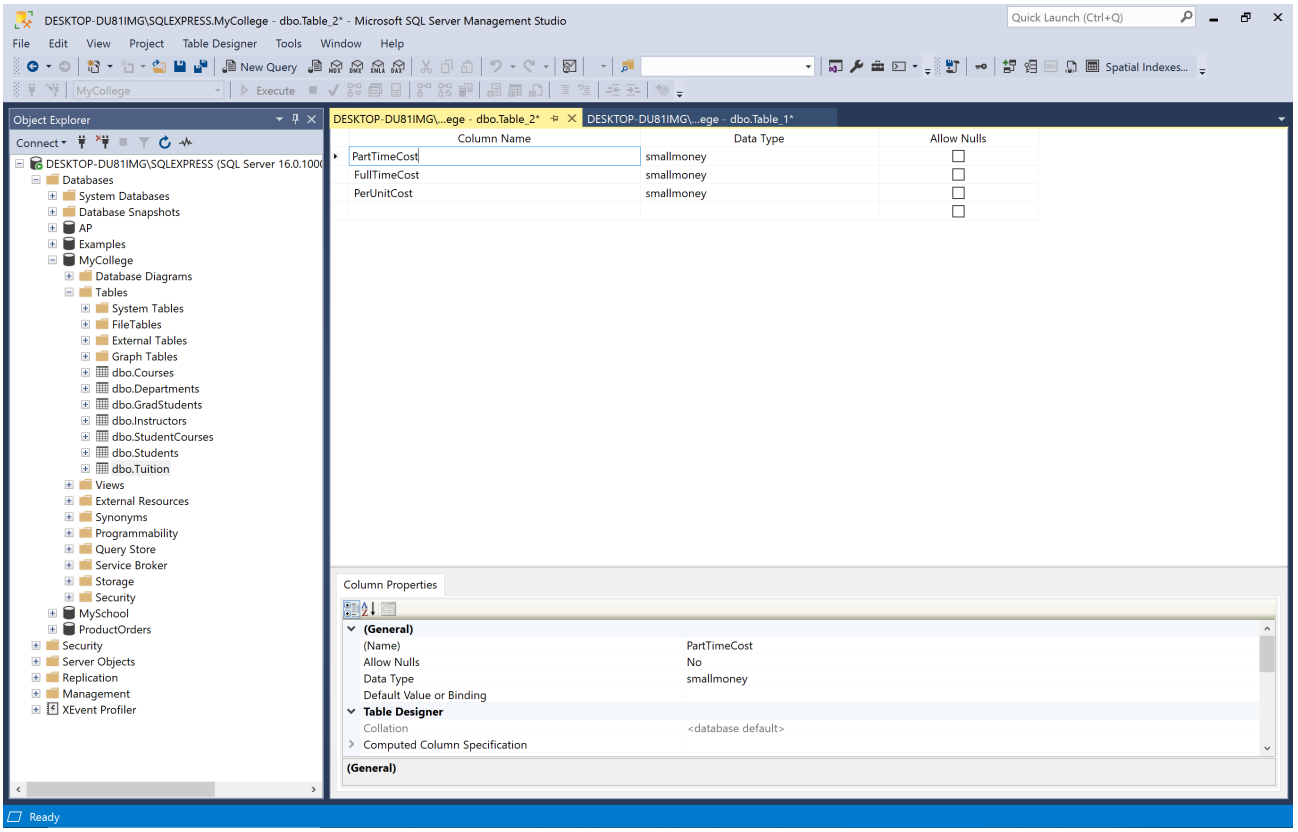
Comment : Column definition of table – ‘Instructors’



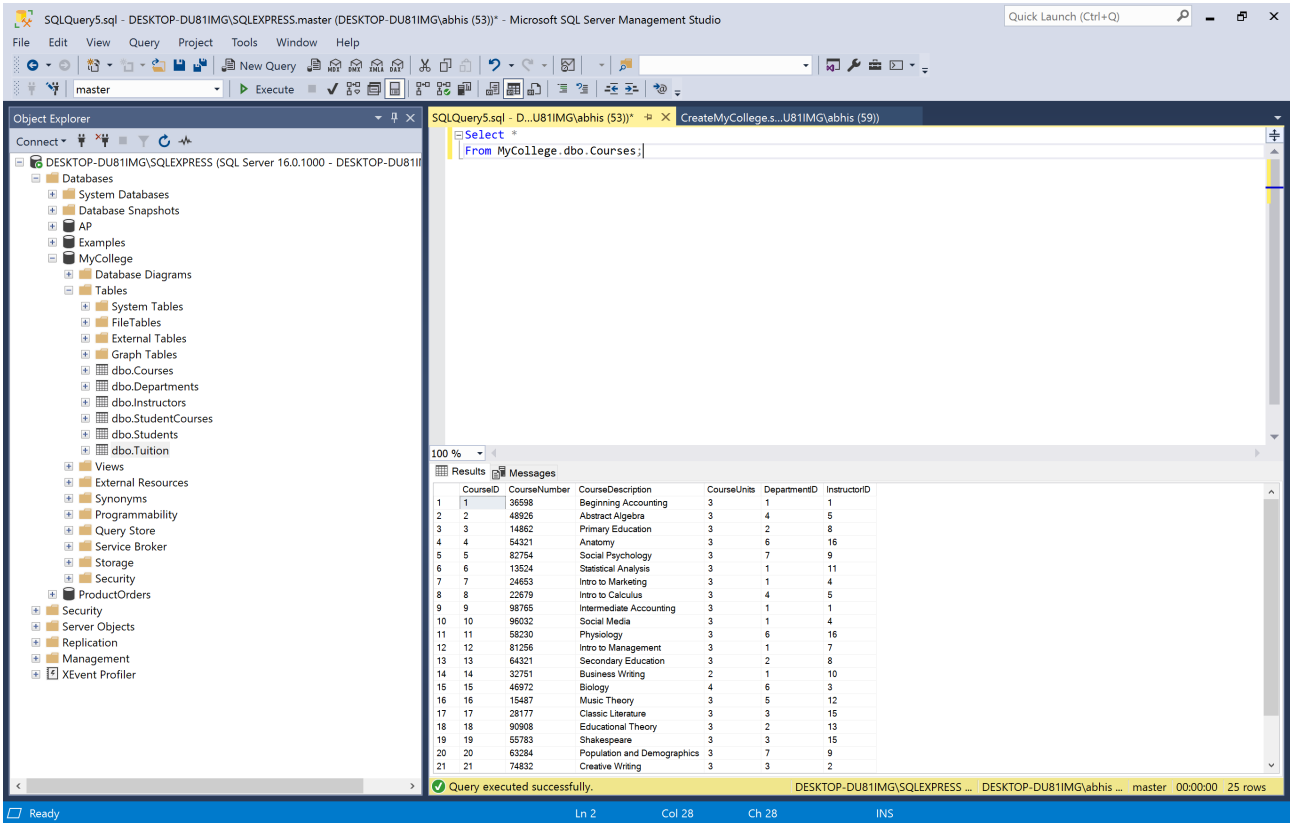
Comment :Column definition of table – ‘StudentCourses’



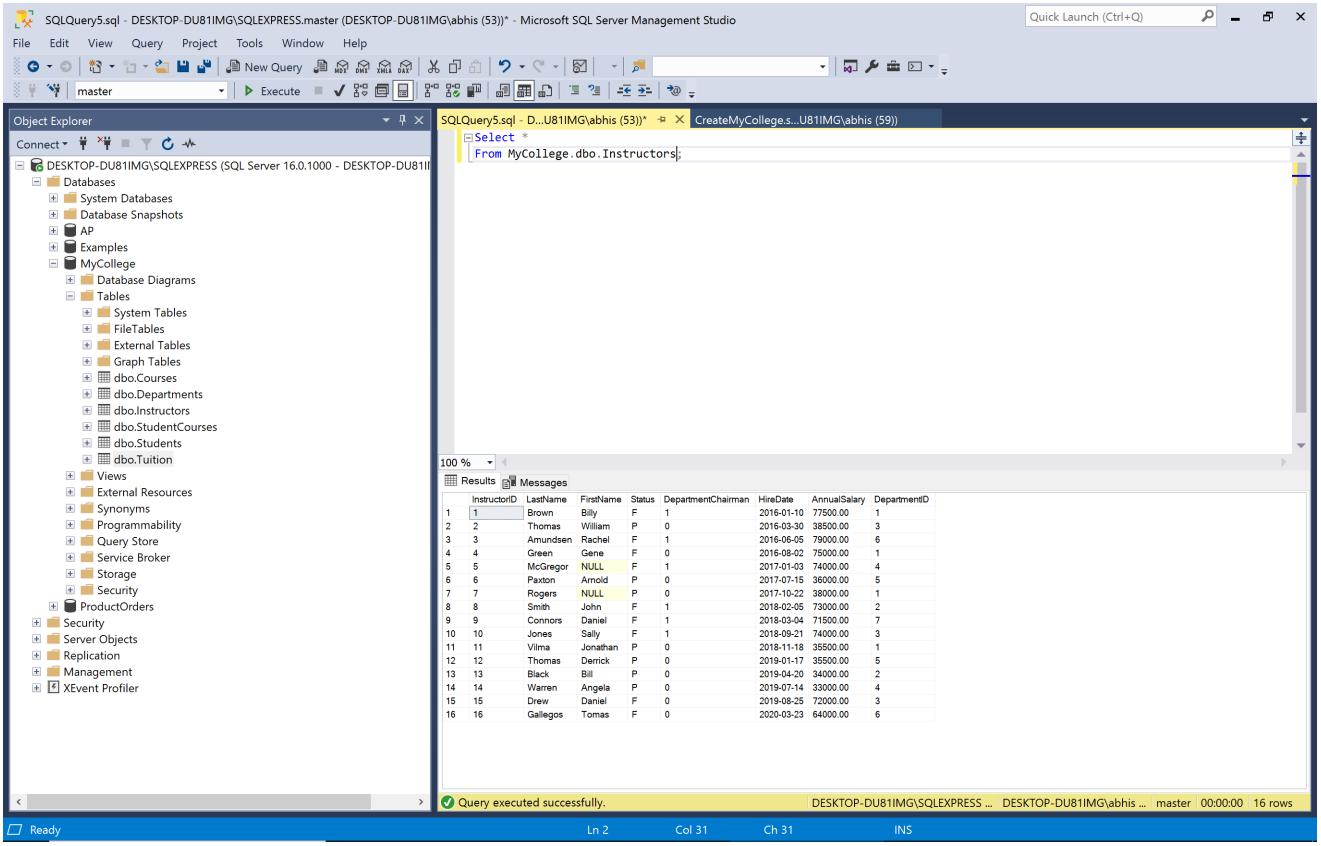
Comment : Column definition of table – ‘Students’



Comment : Column definition of table – ‘Tuition’



Comment : Details in Courses table, table No of rows : 25

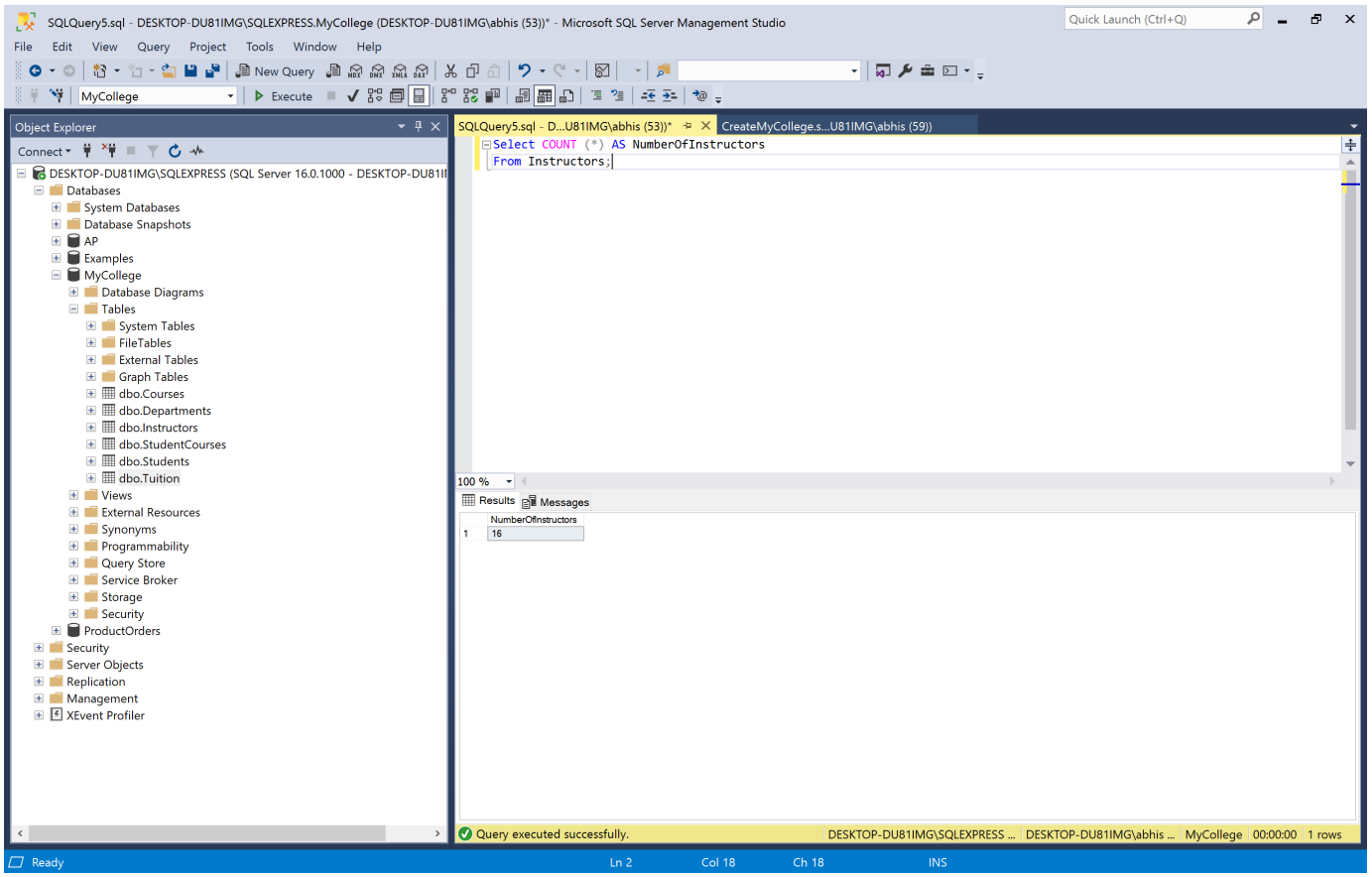


Comment :Details in Instructors, table No of rows : 16

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

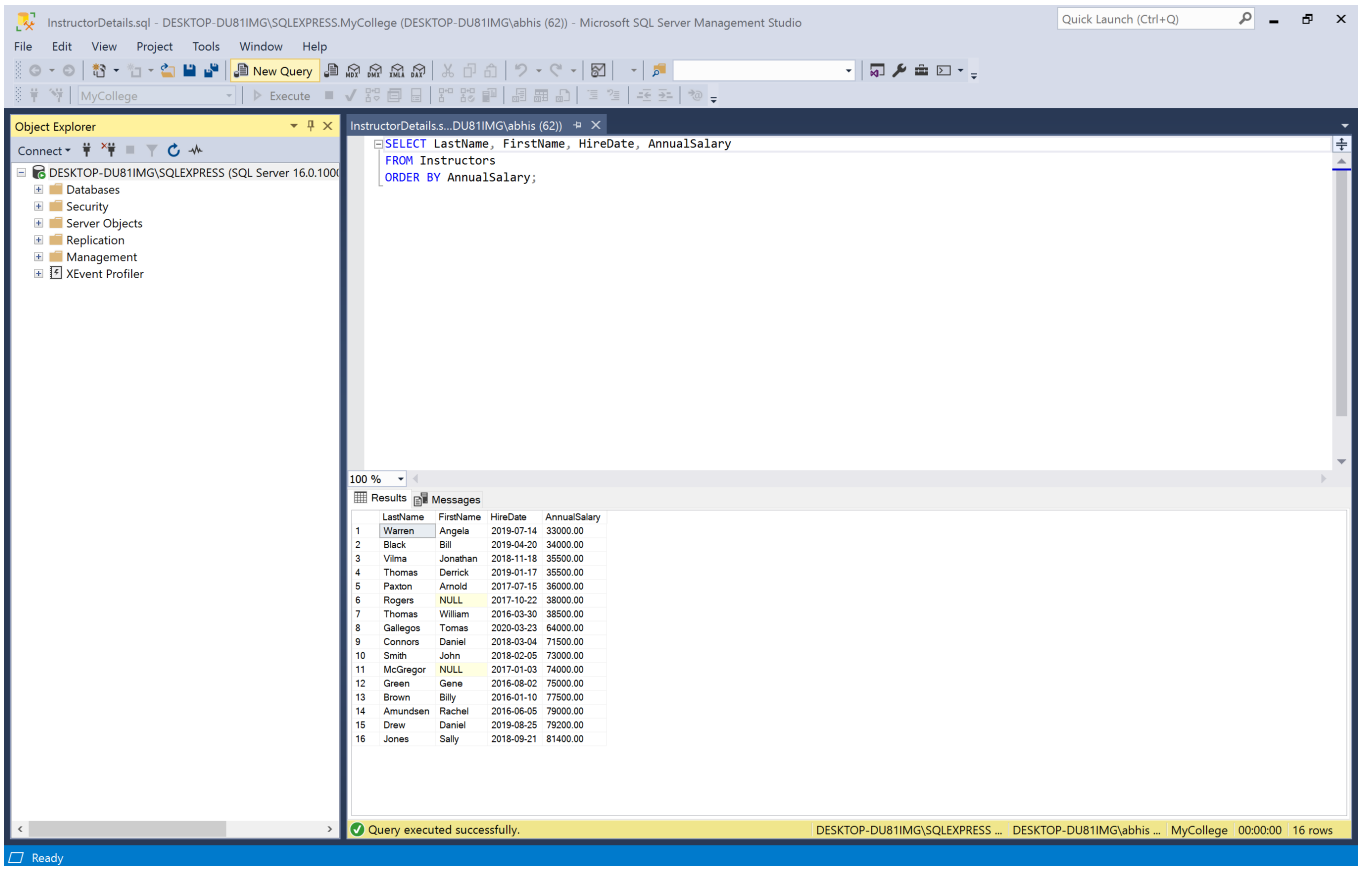
Query –

```
SELECT COUNT(*) AS
NumberOfInstructors FROM
Instructors;
```



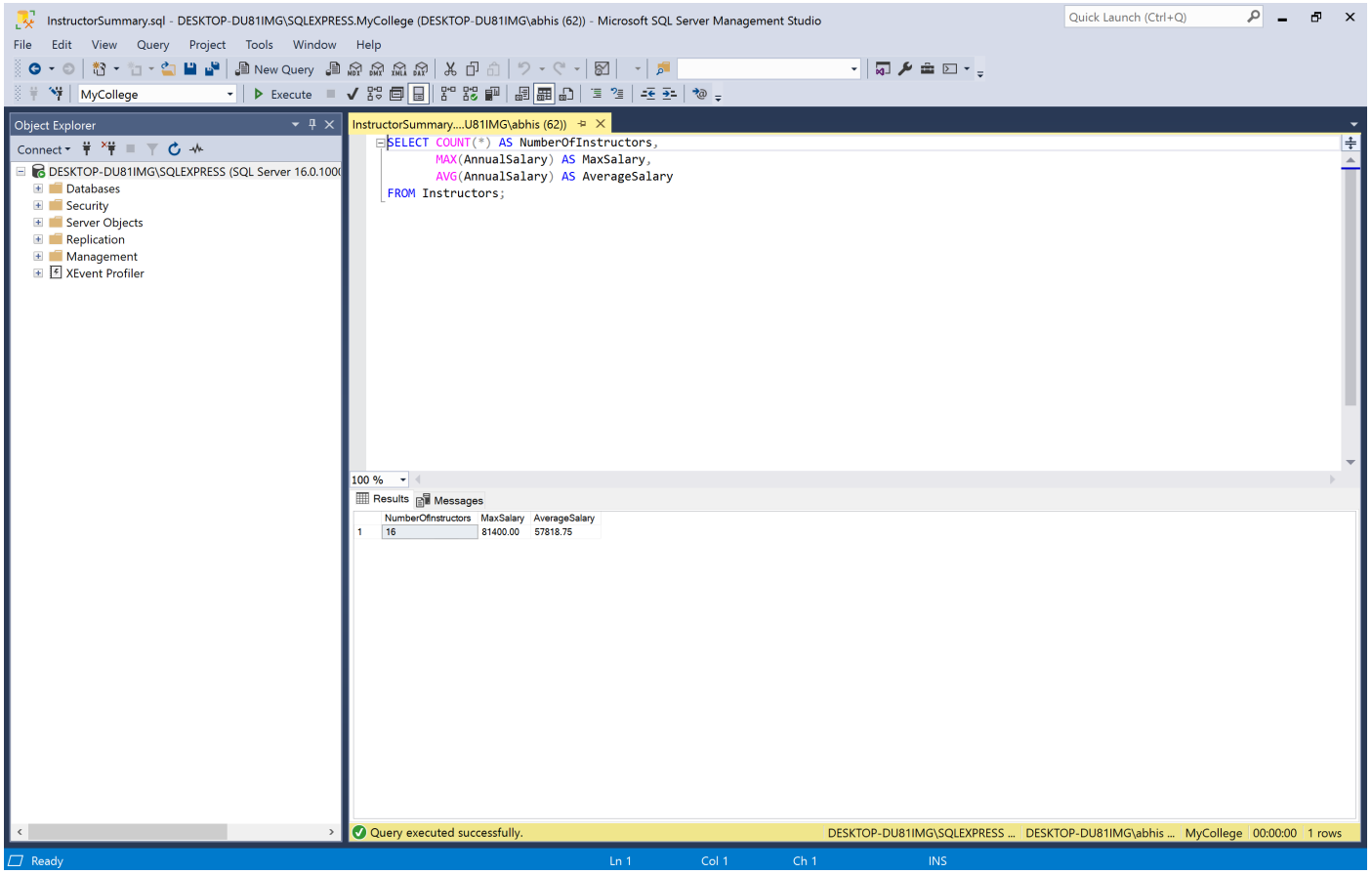
Comment : Query result output, showing the total number of instructors in Instructors table.
No of rows : 1

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



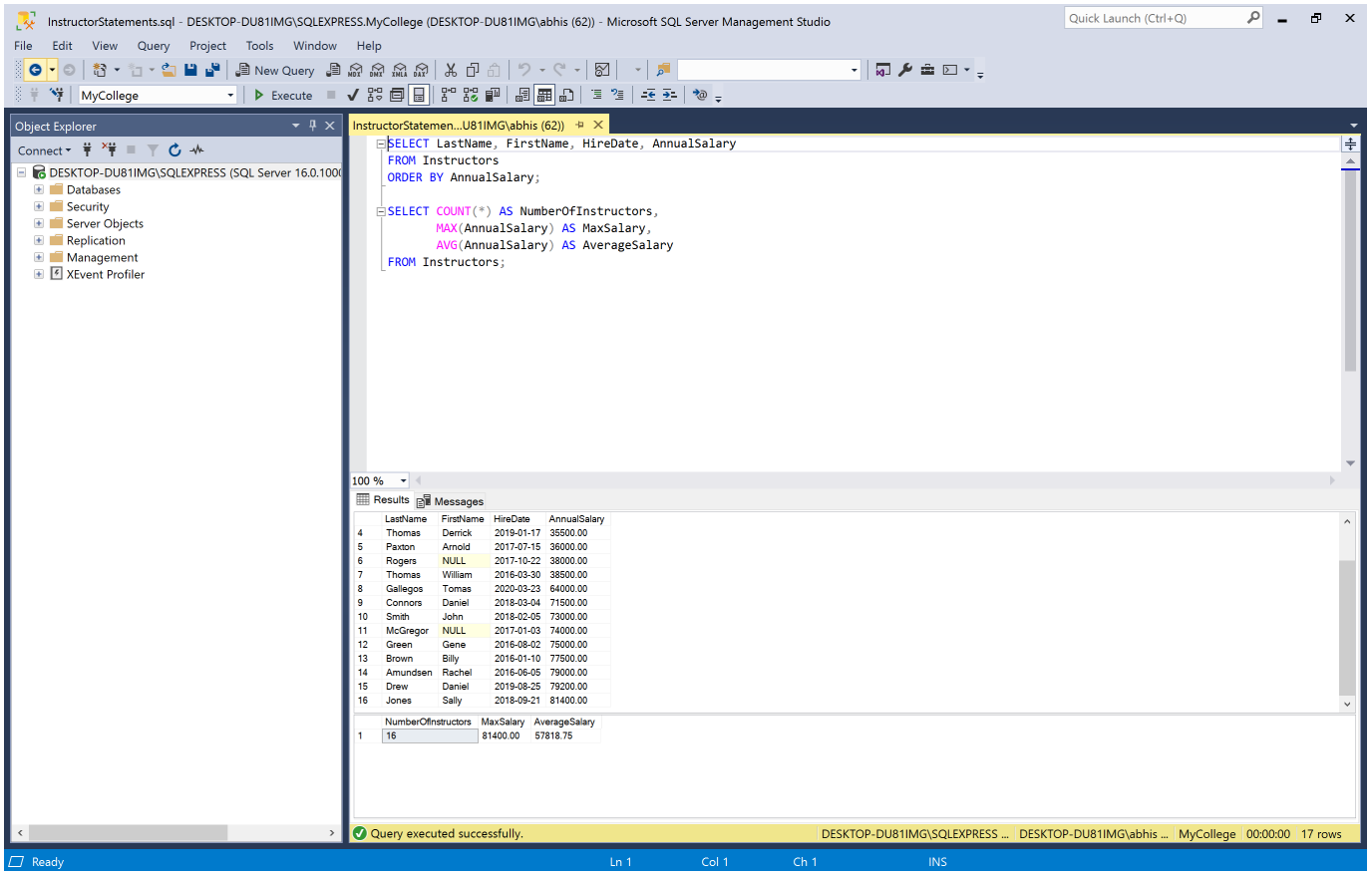
Comment : Query result output, showing the Annual Salary of Instructors & noted the script contains only one SQL statement. No of rows : 16

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



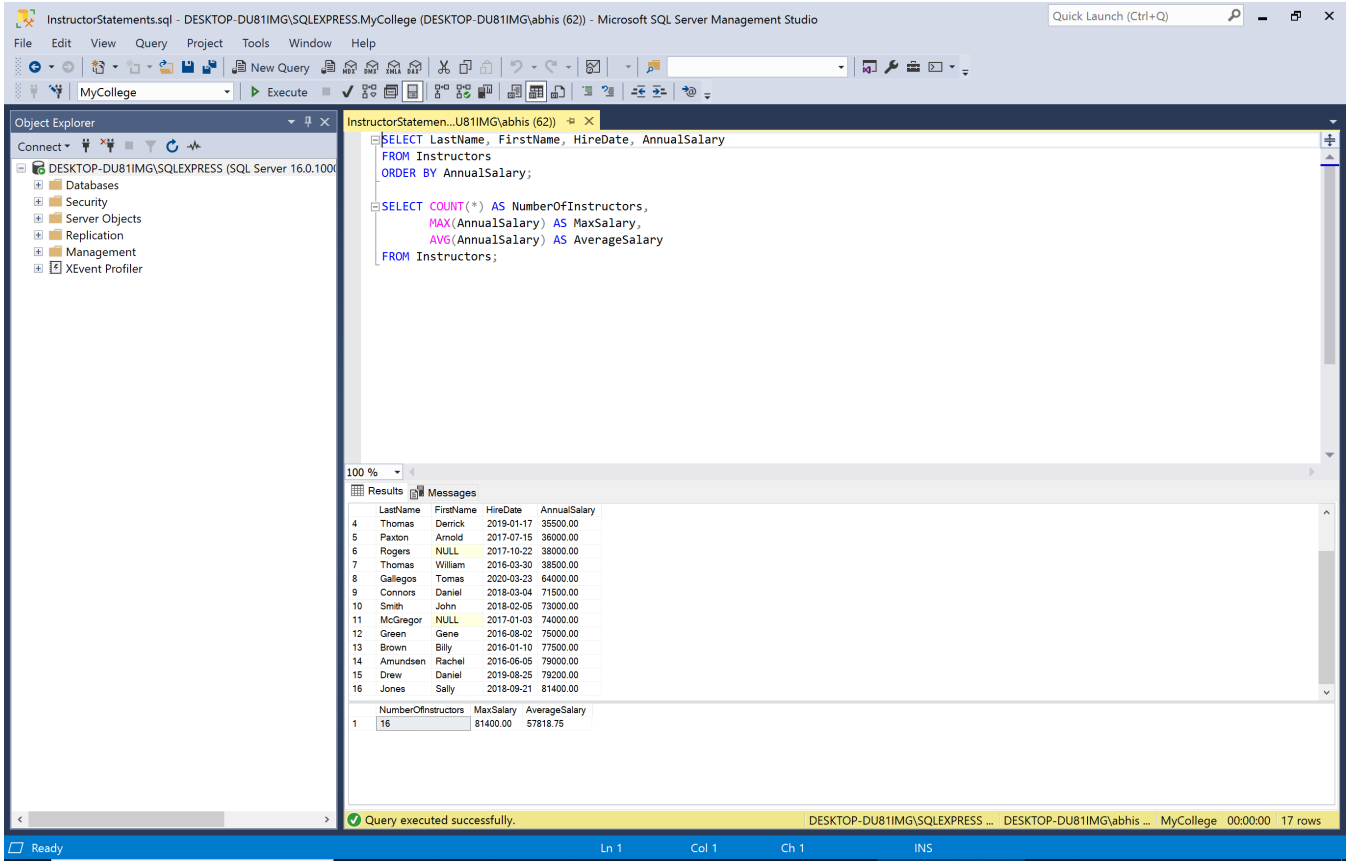
Comment : Query result output, showing the Maximum and Average Salary of Instructors.
No of rows : 1

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



Comment :Opened the script named InstructorStatement.sql and noted that the statements that end with semicolons, which indicated that they are 2 different SQL query statements. No of rows : 16 and for query2 No of rows : 1

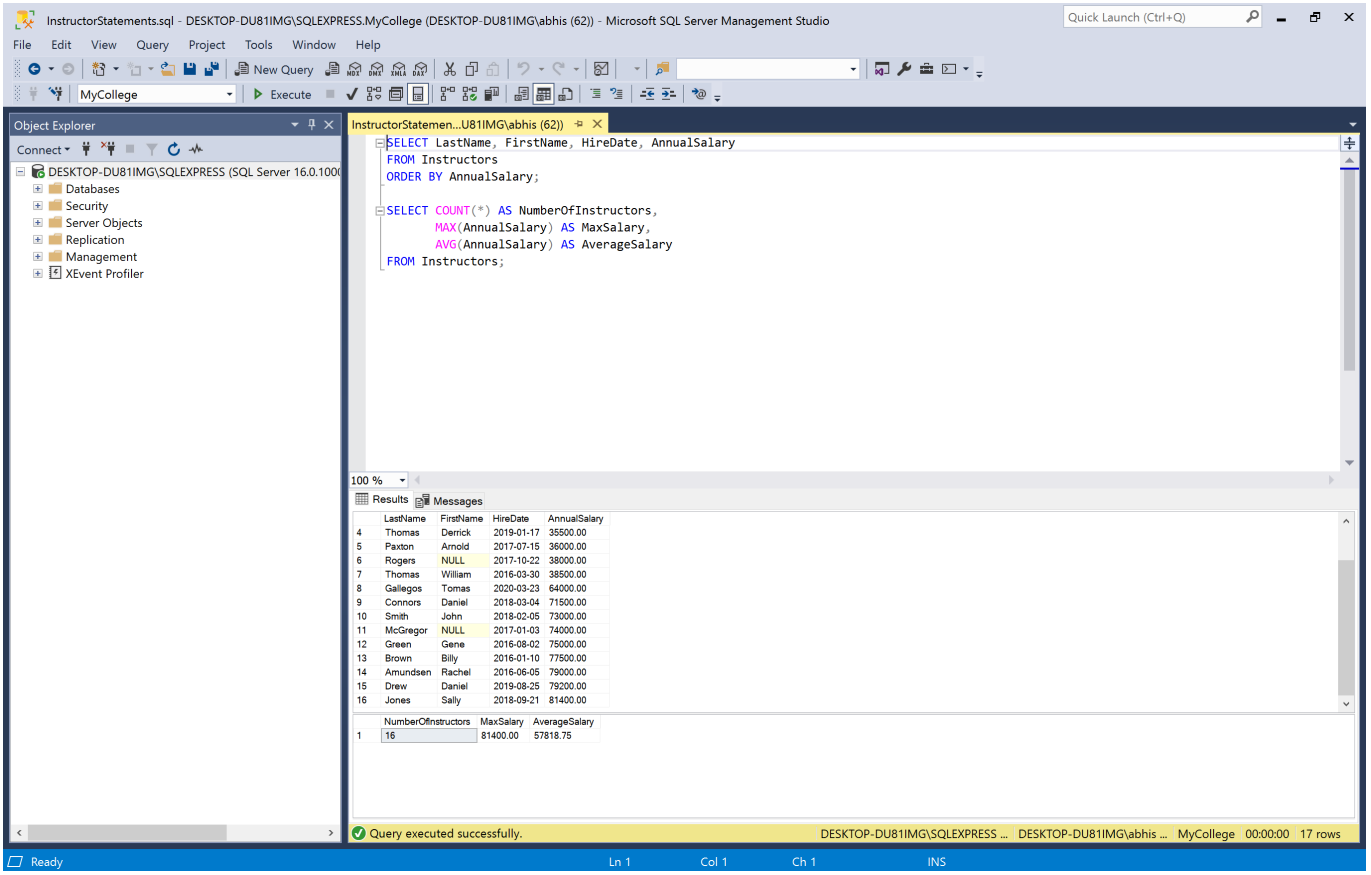
7. Press the F5 key or click the Execute button to run both statements in this script. Note that this displays the results in two Results tabs. Make sure to view the results of both SELECT statements.



Comment : Executed both the statements in the script.

Noted the results in the twoResults tabs, which showcase the Annual Salary, MaxSalary and AverageSalary ofInstructors. No of rows : 17

8. Exit from SQL Server Management Studio.



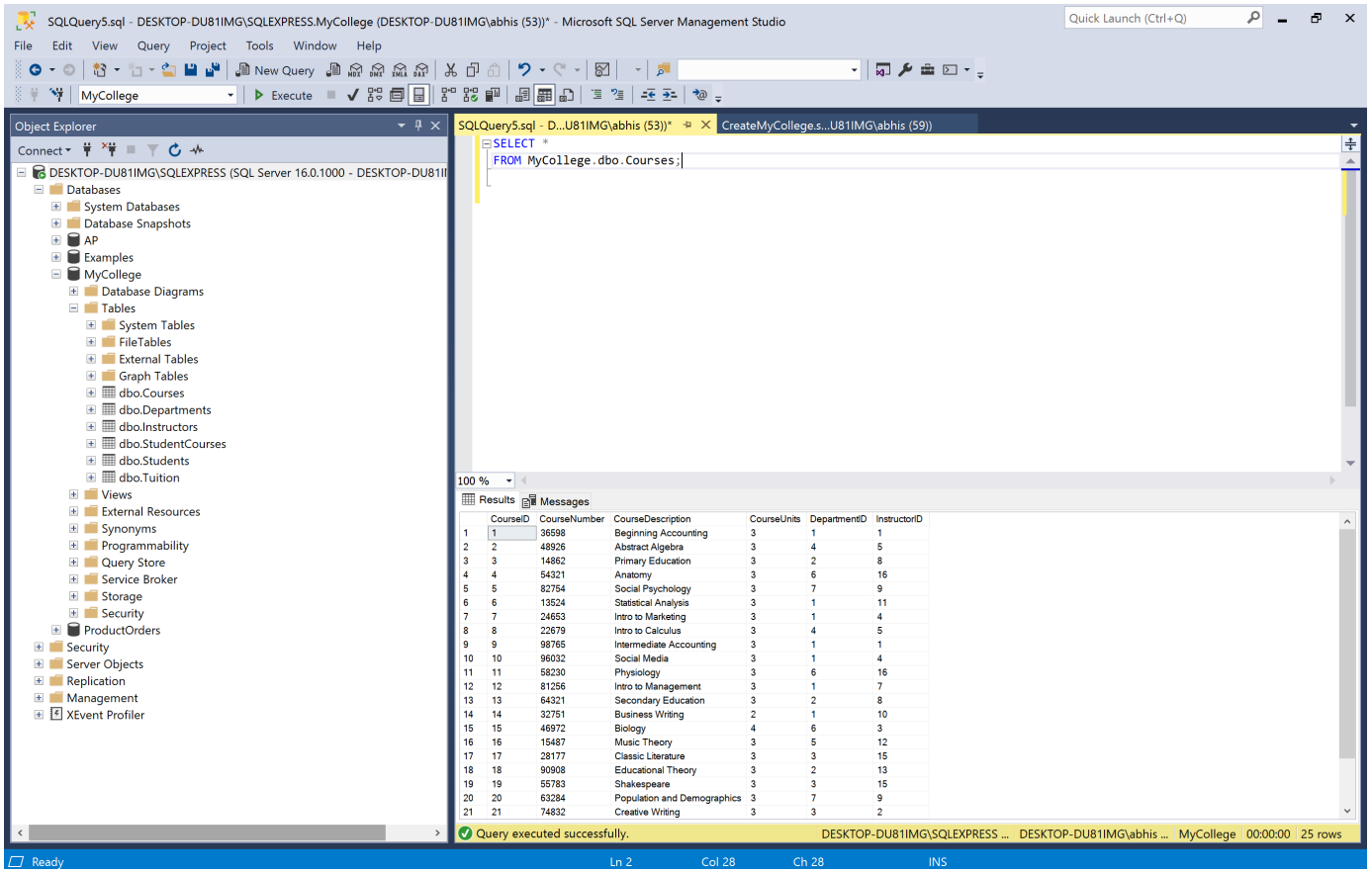
Comment :Closing SQL Server Management Studio. No of rows : 1

- **PART B, Essential SQL Skills –**

1. Write a SELECT statement that returns all of the columns from the Courses table. Then,run this statement to make sure it works correctly.

Query –

```
SELECT *
FROM Courses;
```



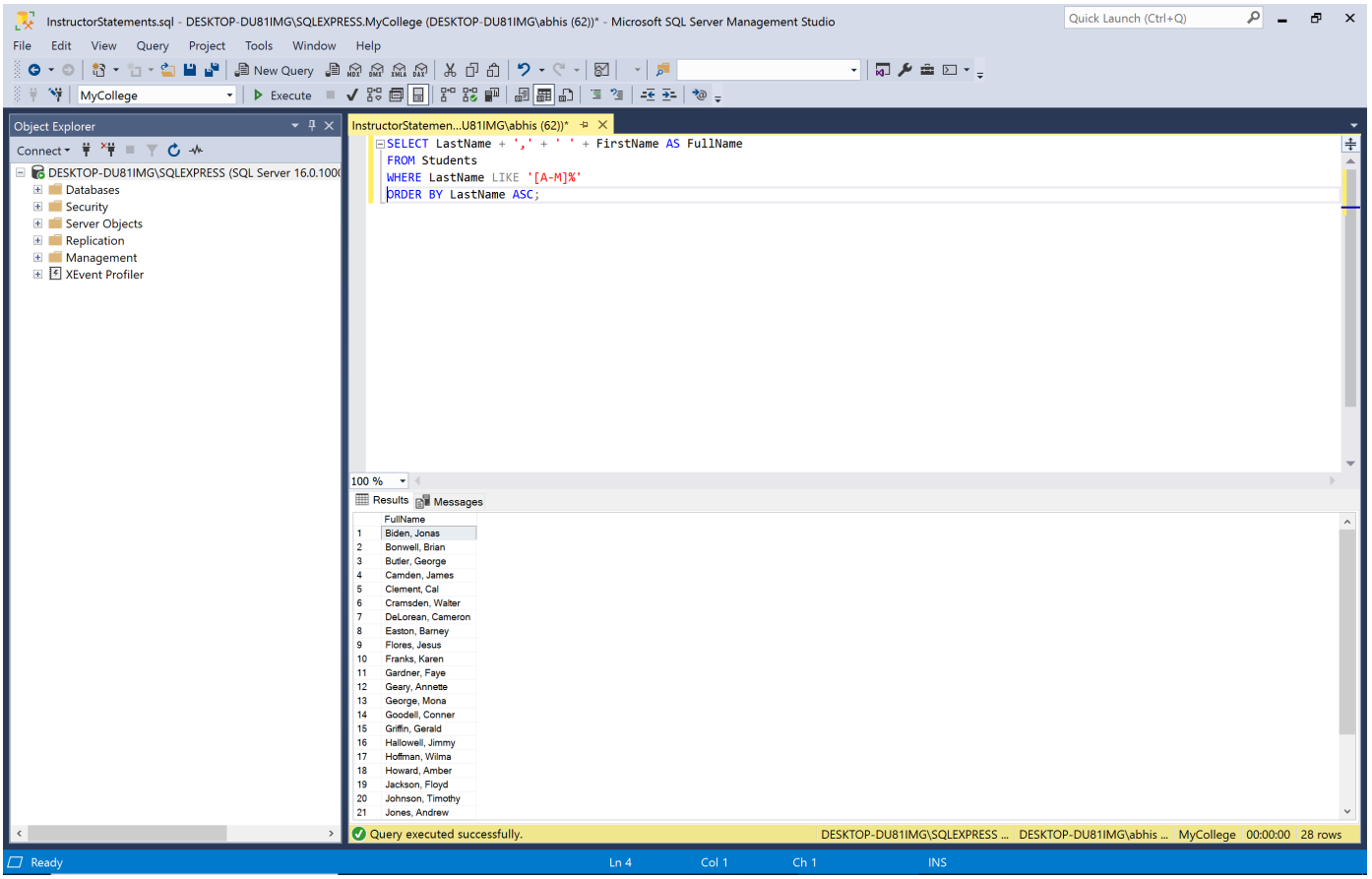
	CourseID	CourseNumber	CourseDescription	CourseUnits	DepartmentID	InstructorID
1	1	36598	Beginning Accounting	3	1	1
2	2	48926	Abstract Algebra	3	4	5
3	3	14862	Primary Education	3	2	8
4	4	54321	Anatomy	3	6	16
5	5	82754	Social Psychology	3	7	9
6	6	13524	Statistical Analysis	3	1	11
7	7	24653	Intro to Marketing	3	1	4
8	8	22679	Intro to Calculus	3	4	5
9	9	98765	Intermediate Accounting	3	1	1
10	10	96032	Social Media	3	1	4
11	11	58230	Physiology	3	6	16
12	12	81256	Intro to Management	3	1	7
13	13	64321	Secondary Education	3	2	8
14	14	32751	Business Writing	2	1	10
15	15	46972	Biology	4	6	3
16	16	15487	Music Theory	3	5	12
17	17	28177	Classic Literature	3	3	15
18	18	90908	Educational Theory	3	2	13
19	19	55783	Shakespeare	3	3	15
20	20	63284	Population and Demographics	3	7	9
21	21	74832	Creative Writing	3	3	2

Comment :Executed the SELECT statement that returns all the columns of Courses table.
No of rows : 25

2. Write a SELECT statement that returns one column from the Students table named FullName that joins the LastName and FirstName columns. Format this column with the last name, a comma, a space, and the first name. Sort the result set by last name in ascending sequence. Return only the students whose last name begins with a letter fromA to M.

Query –

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



Comment :Successfully executed a SELECT Statement that returns a column ‘FullName’ inthe required format using Students table. No of rows : 28

3. Write a SELECT statement that returns these column names and data from theInstructors table:

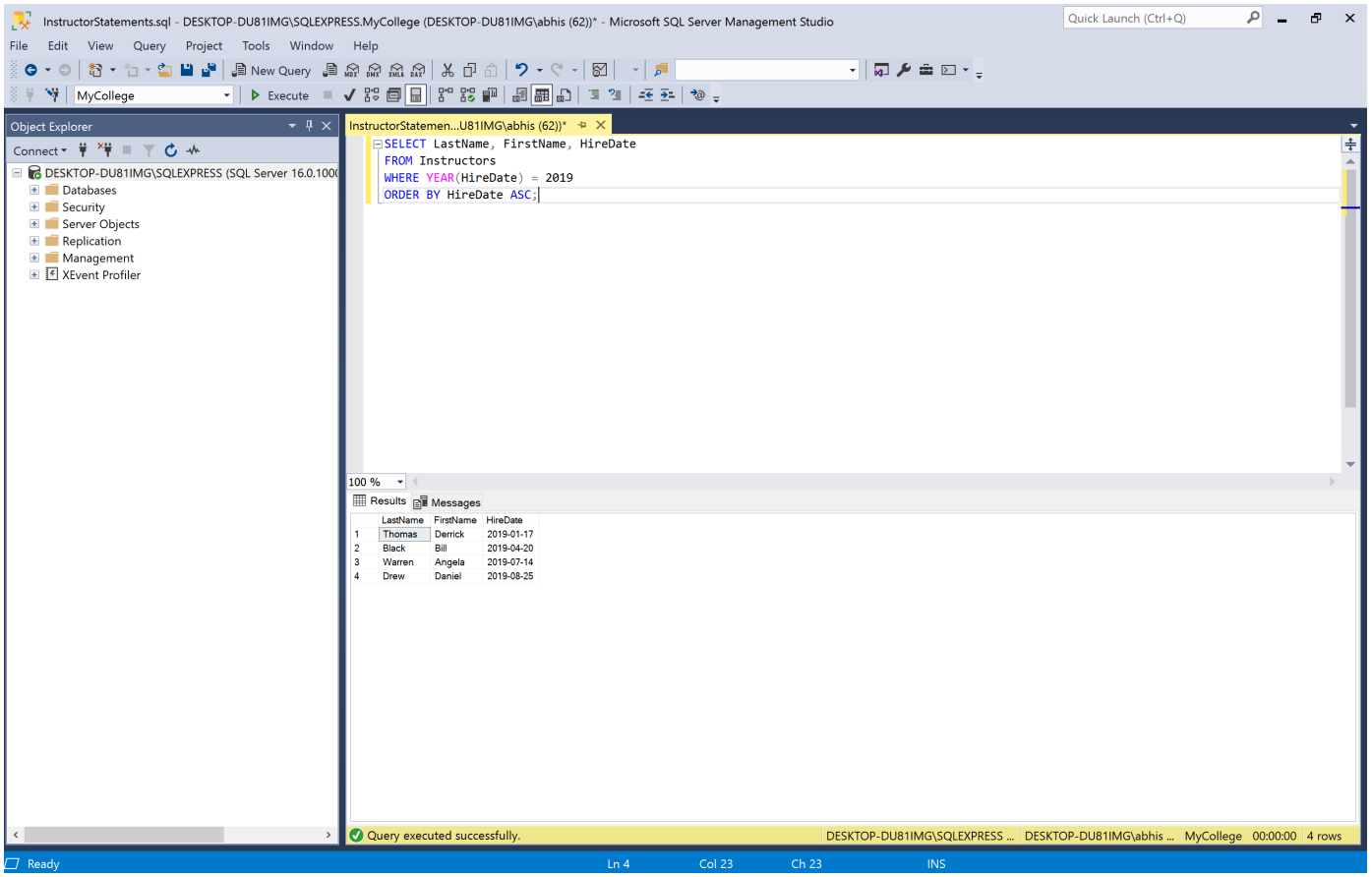
   LastName The LastName column
   FirstName The FirstName column
   HireDate The HireDate column

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

YEAR() – The YEAR() function returns the year part for a specified date.

Query –

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



Comment :SELECT query result output, displaying the column names and data from Instructors table according to the requirements using YEAR function and ORDER BY.
Noof rows : 4

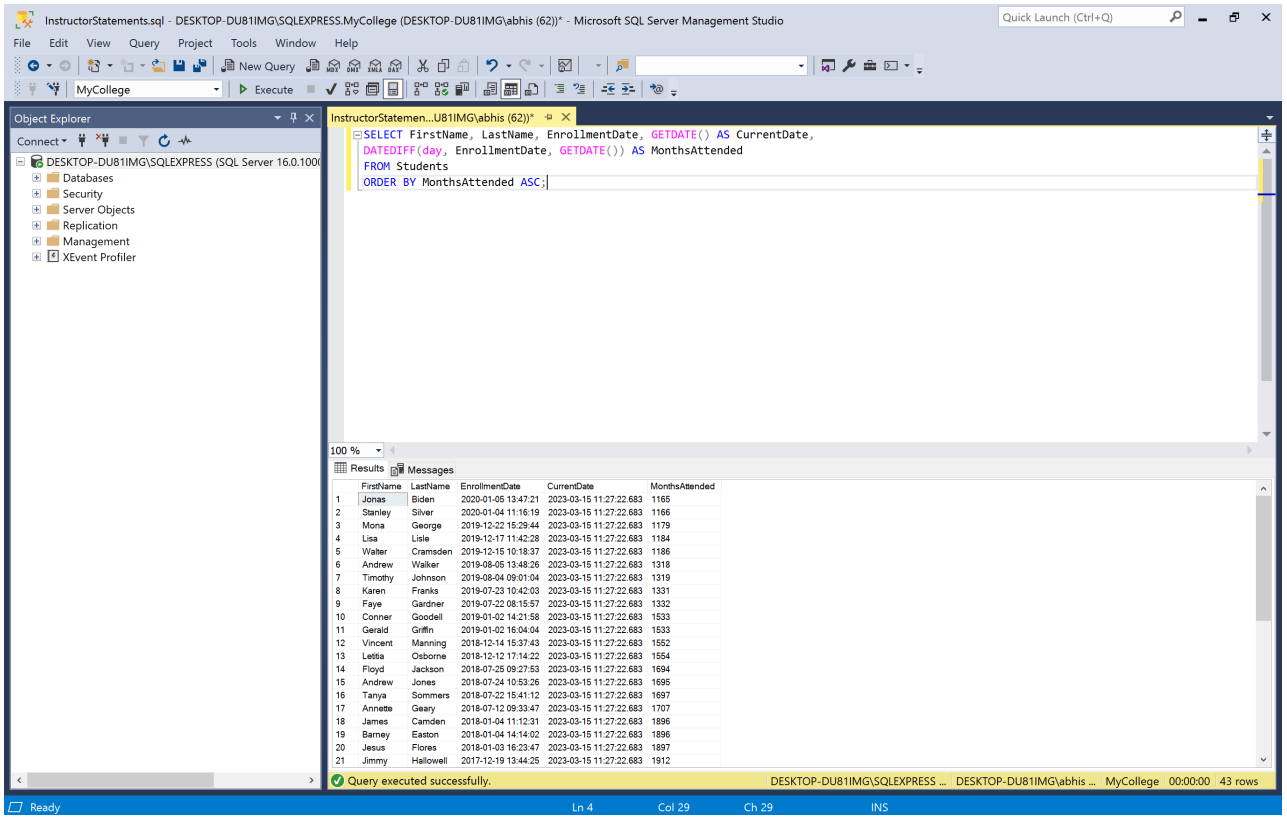
4. Write a SELECT statement that returns these column names and data from the Studentstable:

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

   Sort the result set in ascending sequence by the MonthsAttended column.

Query –

```
SELECT FirstName, LastName, EnrollmentDate, GETDATE() AS
CurrentDate,DATEDIFF(day, EnrollmentDate, GETDATE()) AS
MonthsAttended
FROM Students
ORDER BY MonthsAttended ASC;
```



Comment :Successfully executed a SELECT statement that returns the required column names and data using GETDATE() and DATEDIFF() function; ordered them in ascendingorder using ORDER BY. No of rows : 43

5. Write a SELECT statement that returns these column names and data from theInstructors table:

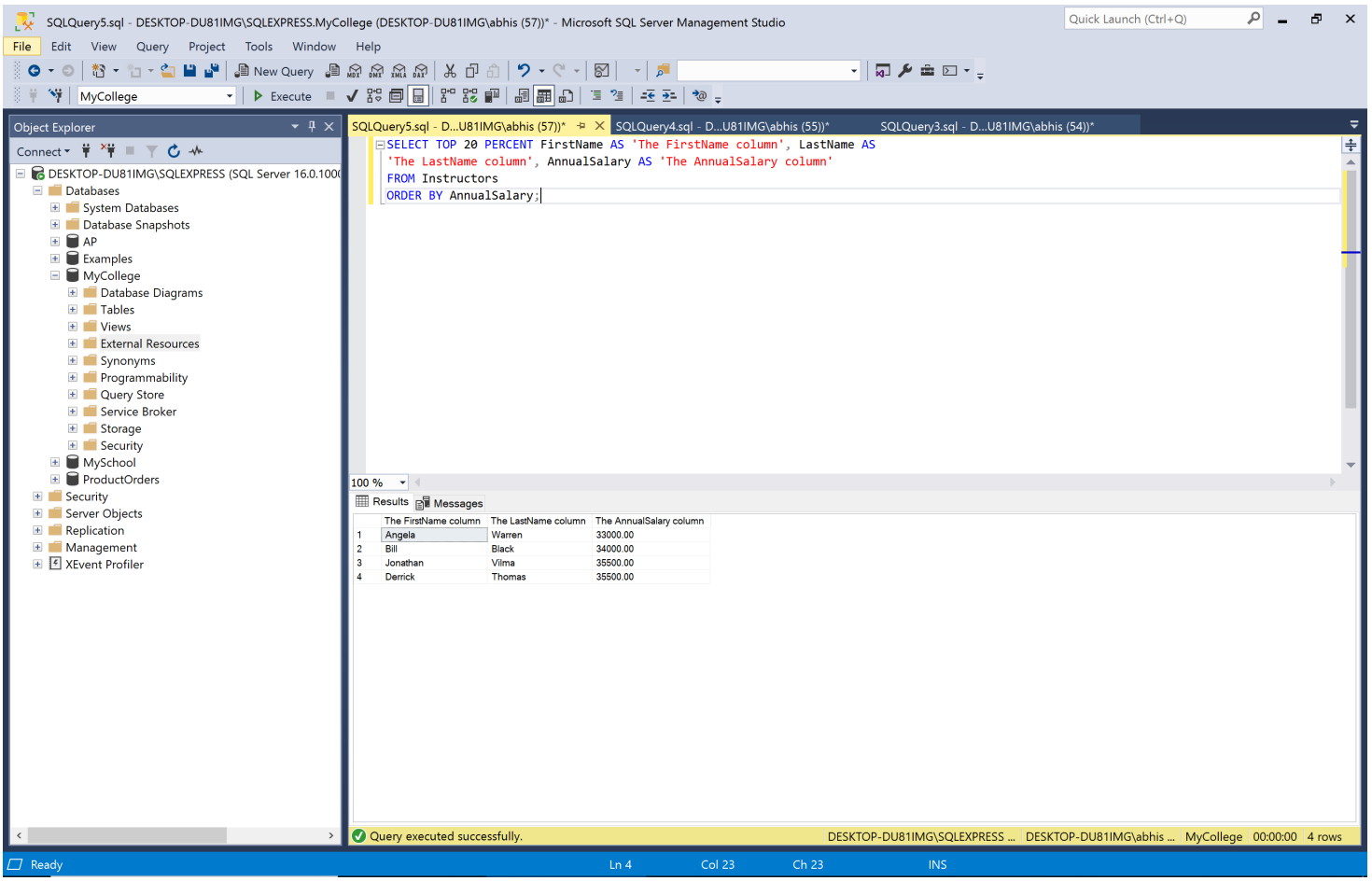
FirstName	The FirstName column
LastName	The LastName column
AnnualSalary	The AnnualSalary column

Return only the top 20 percent of instructors based on annual salary.

TOP clause – The TOP clause is used to specify the number of records to return in both numeric and percentage format.

Query –

```
SELECT TOP 20 PERCENT FirstName AS 'The FirstName column',
LastName AS'The LastName column', AnnualSalary AS 'The
AnnualSalary column'
FROM Instructors
ORDER BY AnnualSalary DESC;
```



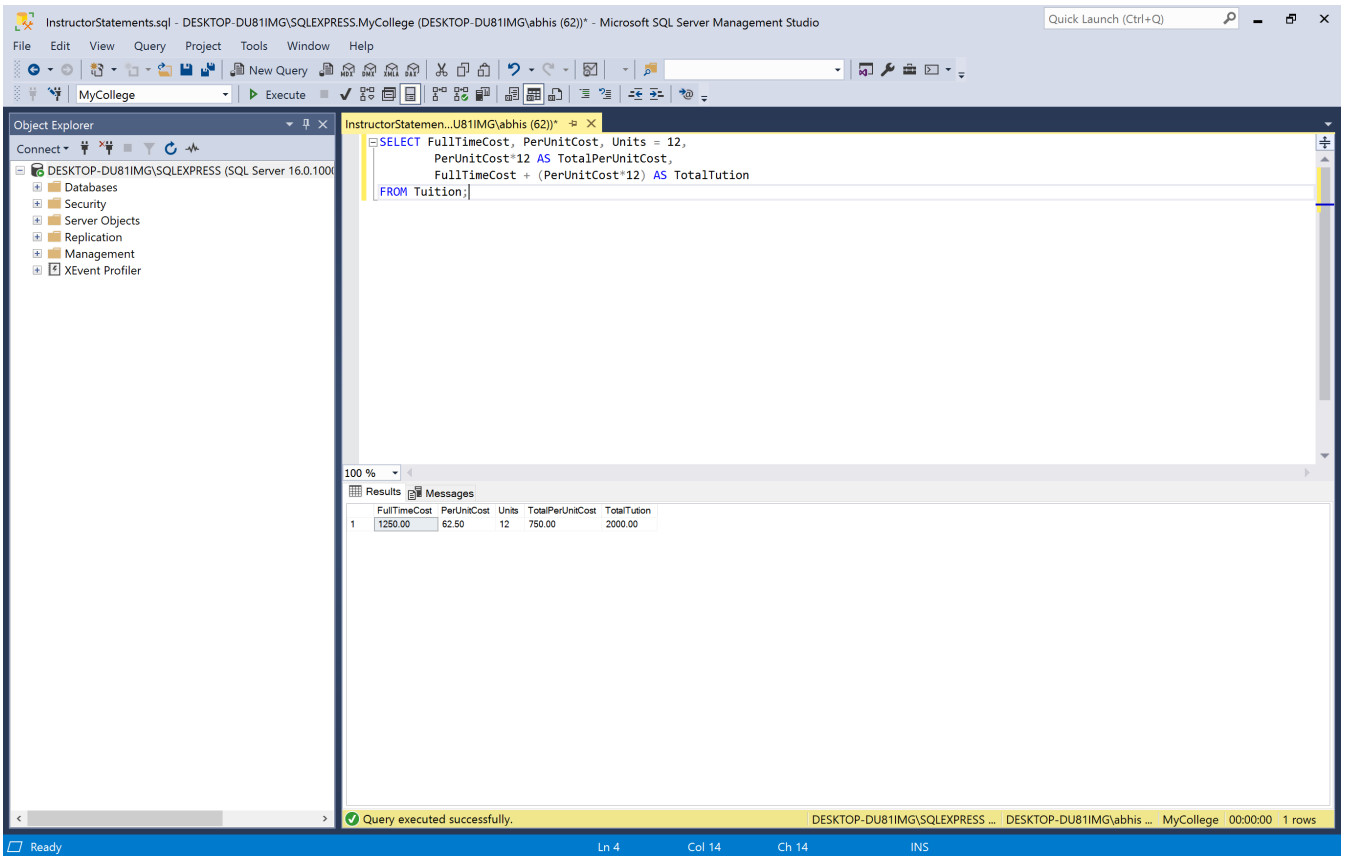
Comment :Successfully executed a SELECT statement that returns the required column names and data using TOP clause and ordered them based on annual salary using ORDER BY. No of rows : 4

6. Write a SELECT statement that returns these columns and data from the Tuition table, along with a constant value and two calculated values:

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

Query –

```
SELECT FullTimeCost, PerUnitCost, Units = 12,
        PerUnitCost*12 AS TotalPerUnitCost,
        FullTimeCost + (PerUnitCost*12) AS
        TotalTution
FROM Tuition;
```

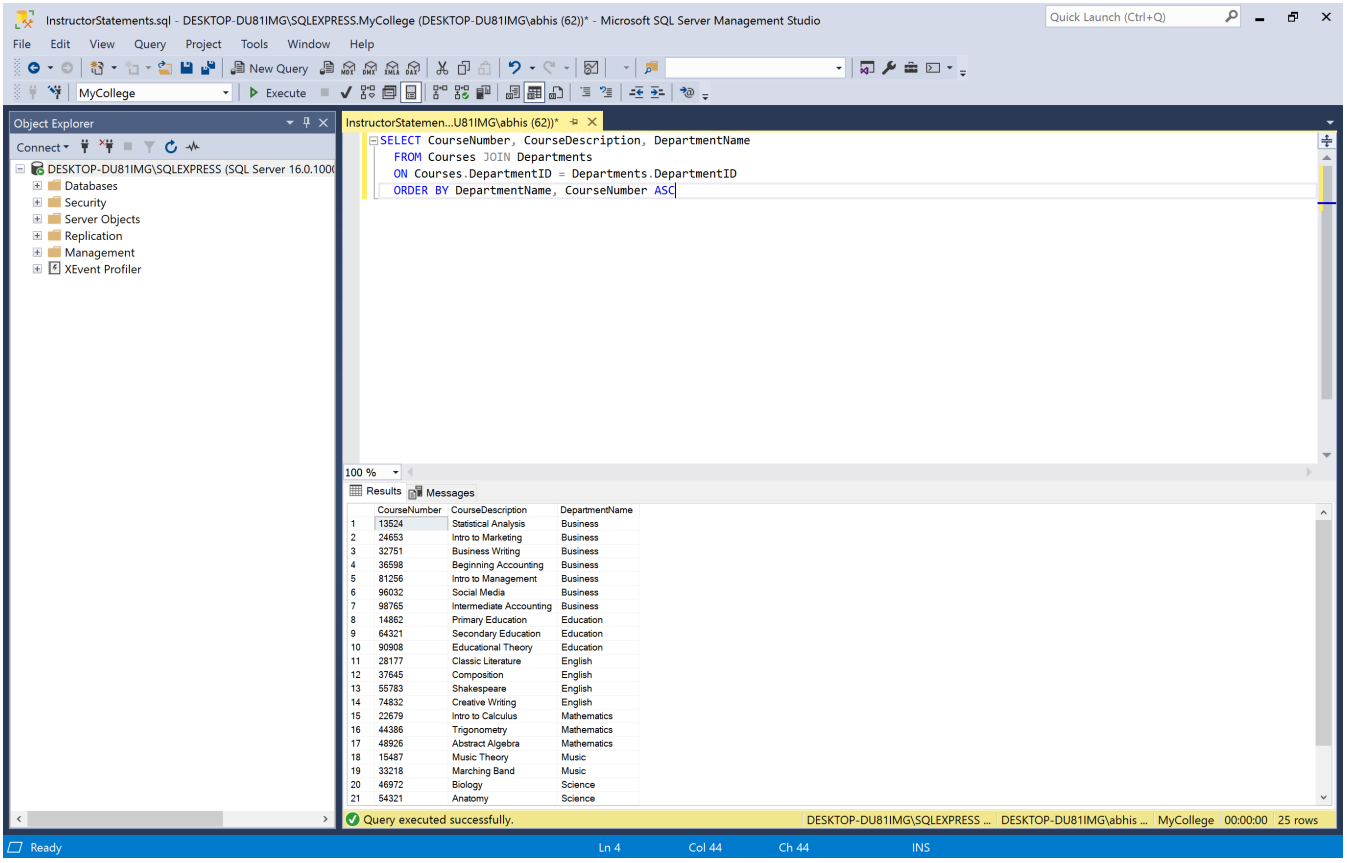


Comment :Successfully executed a SELECT statement that returns the required columns and data from Tuition table using SQL arithmetic operators. No of rows : 1

7. Write a SELECT statement that joins the Courses table to the Departments table and returns these columns: CourseNumber, CourseDescription, DepartmentName. Sort the result set by DepartmentName and then by CourseNumber in ascending order.

Query –

```
SELECT CourseNumber, CourseDescription,
DepartmentNameFROM Courses JOIN Departments
ON Courses.DepartmentID =
Departments.DepartmentIDORDER BY
DepartmentName, CourseNumber ASC
```



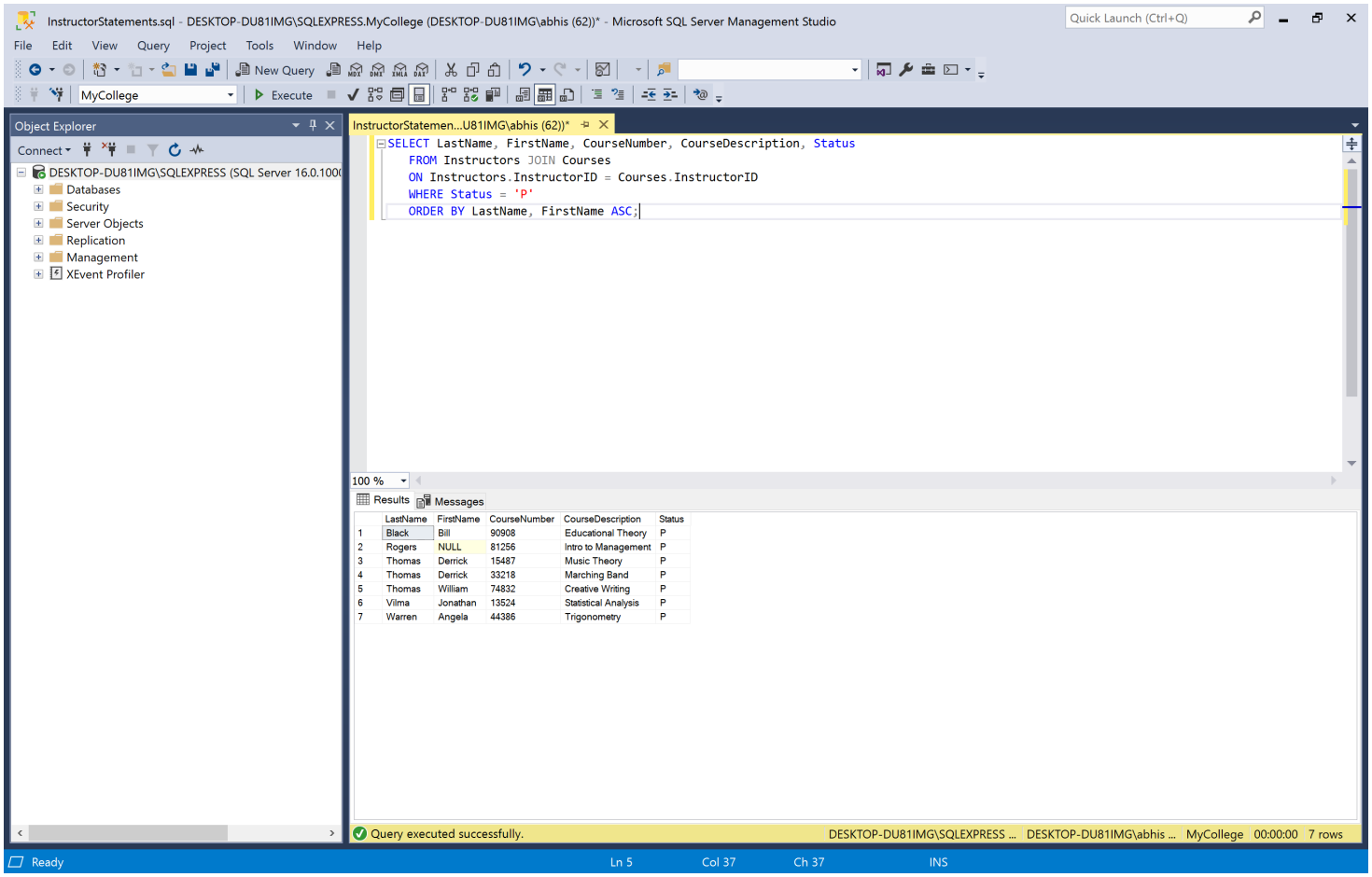
Comment :Successfully executed the SELECT statement that returns the 3 required columns after applying JOIN operation between Courses and Departments table; orderedthem in ascending order using ORDER BY. No of rows : 25

8. Write a SELECT statement that joins the Instructors table to the Courses table and returns these columns: LastName, FirstName, CourseNumber, CourseDescription.

   Return all courses for each instructor with a status of “P” (part time). Sort the result set by LastName and then by FirstName in ascending order.

   Query –

```
SELECT LastName, FirstName, CourseNumber, CourseDescription,
StatusFROM Instructors JOIN Courses
ON Instructors.InstructorID =
Courses.InstructorIDWHERE Status = 'P'
ORDER BY LastName, FirstName ASC;
```



Comment :Successfully executed a SELECT statement that returns only the required rowsof data with status “P” using JOIN operation between Instructors and Courses table; ordered them in the necessary manner using ORDER BY. No of rows : 7

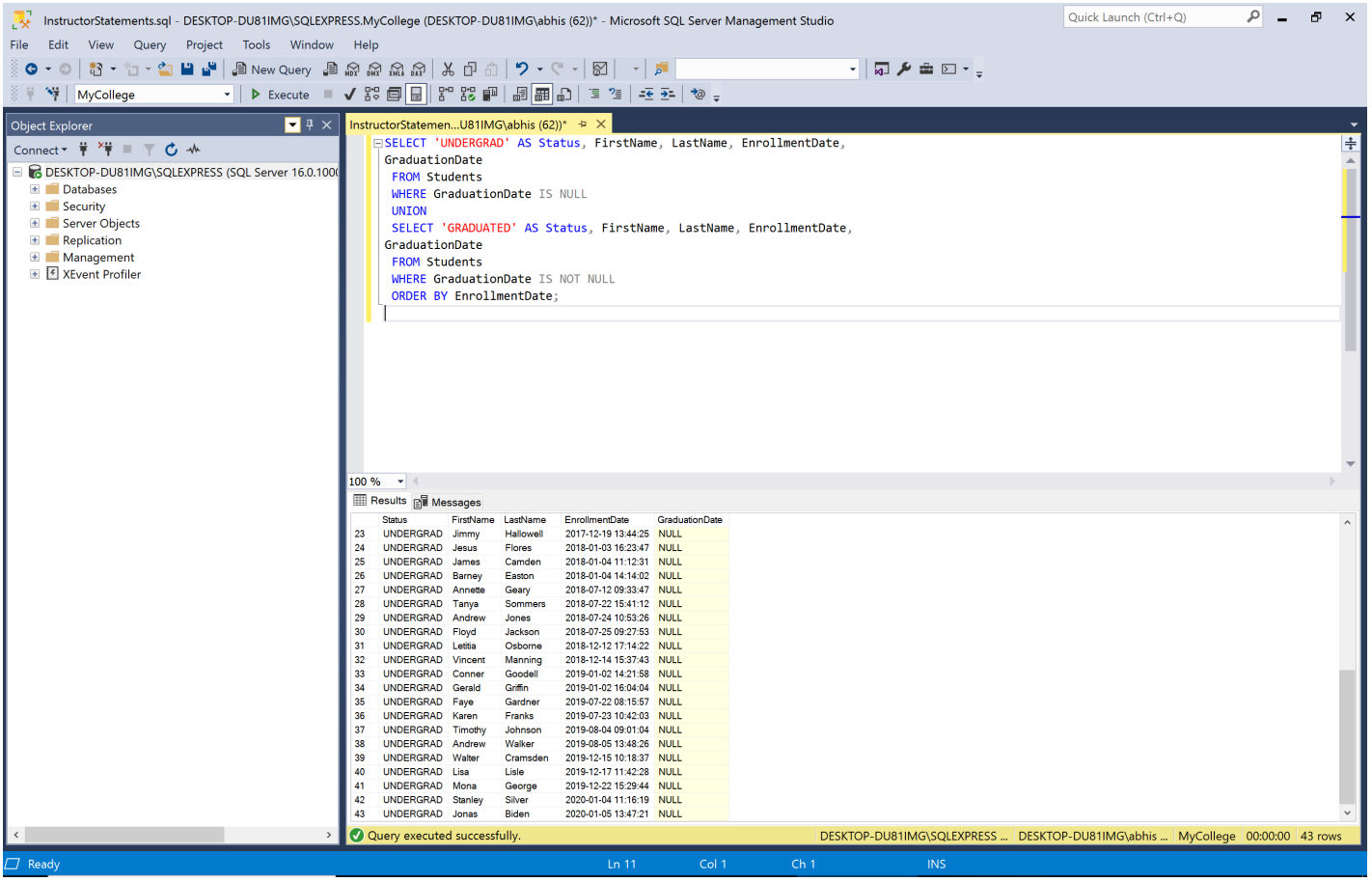
9. Use the UNION operator to generate a result set consisting of five columns from theStudents table:

Status	A calculated column that contains a value of UNDERGRADor 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.Query –

```
SELECT 'UNDERGRAD' AS Status, FirstName, LastName,
EnrollmentDate,GraduationDate
FROM Students
WHERE GraduationDate IS
NULLUNION
SELECT 'GRADUATED' AS Status, FirstName, LastName,
EnrollmentDate,GraduationDate
FROM Students
WHERE GraduationDate IS NOT
NULLORDER BY EnrollmentDate;
```



Comment :Successfully executed the query that returns the required (data) rows of UNDERGRAD and GRADUATED students using UNION and SQL NULL values. No ofrows : 43

10. Write a SELECT statement that returns these columns:

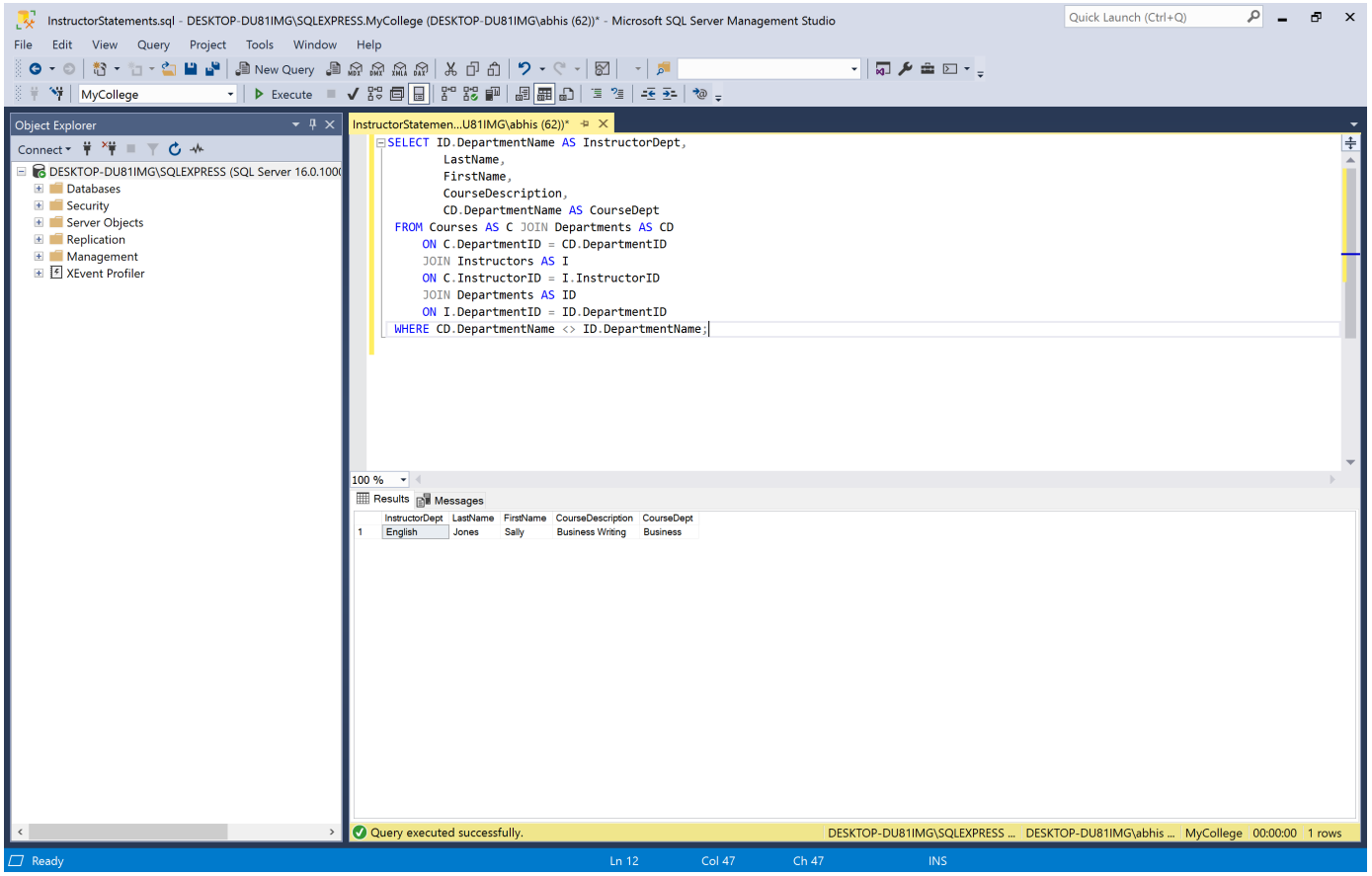
    InstructorDept The DepartmentName column from the Departments tablefor a related instructor

    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.
    (Hint: You will need to join the Departments table to both the Instructors table and the Courses table, which will require you to use table aliases to distinguish the two tables.)

    Query –

```
SELECT ID.DepartmentName AS
       InstructorDept,LastName,
       FirstName,
       CourseDescriptio
       n,
       CD.DepartmentName AS
CourseDept FROM Courses AS C JOIN
Departments AS CD
    ON C.DepartmentID =
    CD.DepartmentIDJOIN Instructors
    AS I
    ON C.InstructorID =
    I.InstructorIDJOIN Departments
    AS ID
    ON I.DepartmentID = ID.DepartmentID
WHERE CD.DepartmentName <>
ID.DepartmentName;
```



Comment :Successfully executed a query that returned one row for each course that's in adifferent department than the department of the instructor assigned to teach that course using JOIN that joined three tables i.e Courses, Department and Instructors. No of rows : 1

11. Write a SELECT statement that returns one row for each instructor that has courses withthese 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 tableThe sum of the course units in the Courses table

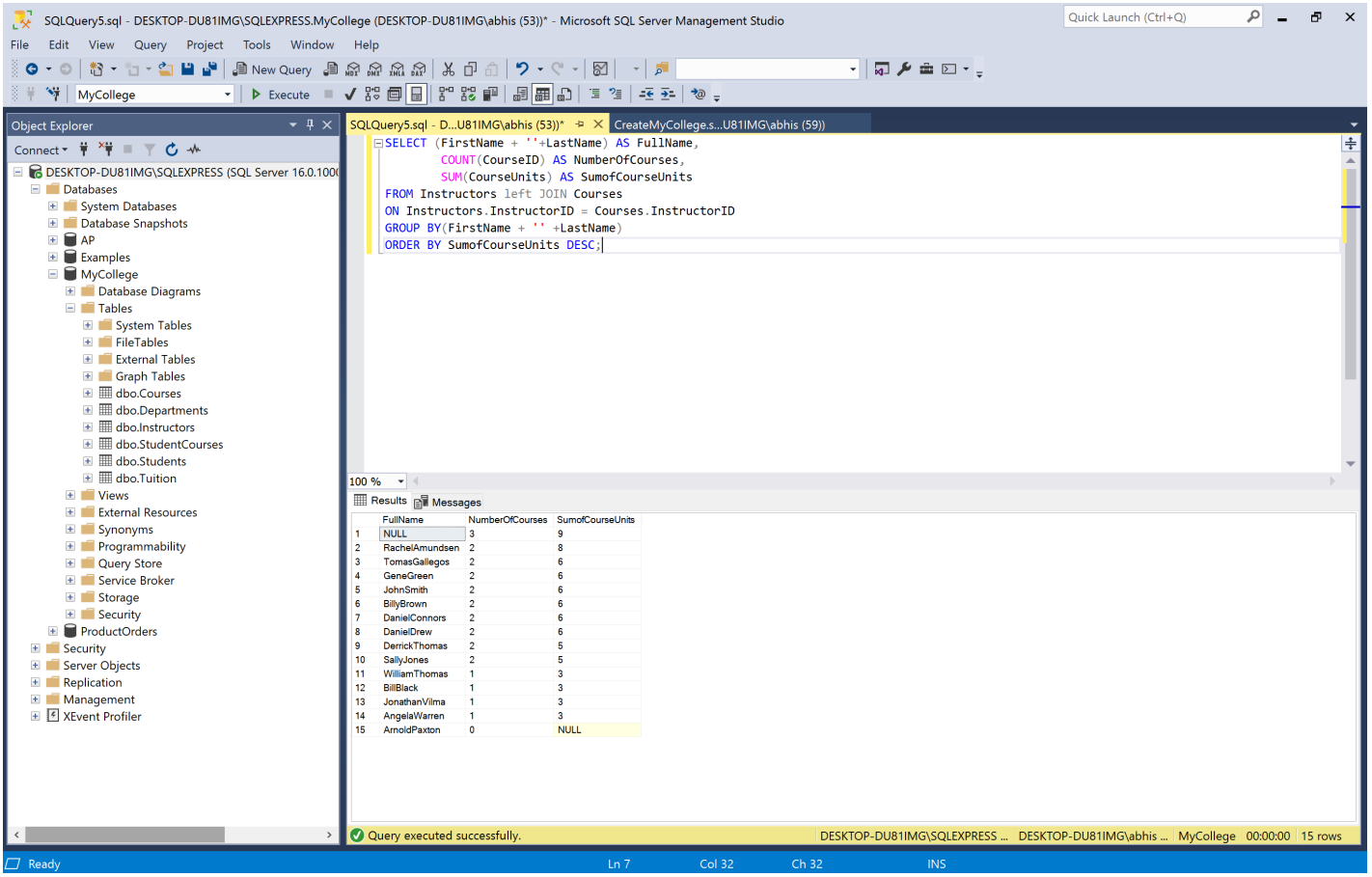
    (Hint: You will need to concatenate the instructor first and last names again in theGROUP BY clause.)

    Sort the result set in descending sequence by the total course units for each instructor.Query –

```
SELECT (FirstName + ' ' + LastName) AS
        FullName,COUNT(CourseID) AS
        NumberofCourses, SUM(CourseUnits) AS
        SumofCourseUnits
FROM Instructors left JOIN Courses
ON Instructors.InstructorID =
Courses.InstructorIDGROUP BY (FirstName + ' '
+ LastName)
ORDER BY SumofCourseUnits DESC;
```

COUNT() Function - The COUNT() function returns the number of rows that matches aspecified criterion.

SUM() Function - The SUM() function returns the total sum of a numeric column.



Comment :Successfully executed a SELECT statement that returns one row for each instructor that has courses using LEFT JOIN, GROUP BY, ORDER BY along with COUNT() and SUM() functions. No of rows : 15 , As, one of the instructor's first name hada null value, the concatenation of the first and last name will resulted in a null value.

12. Write a SELECT statement that answers this question: What is the total number ofcourses taught by parttime instructors? Return these columns:

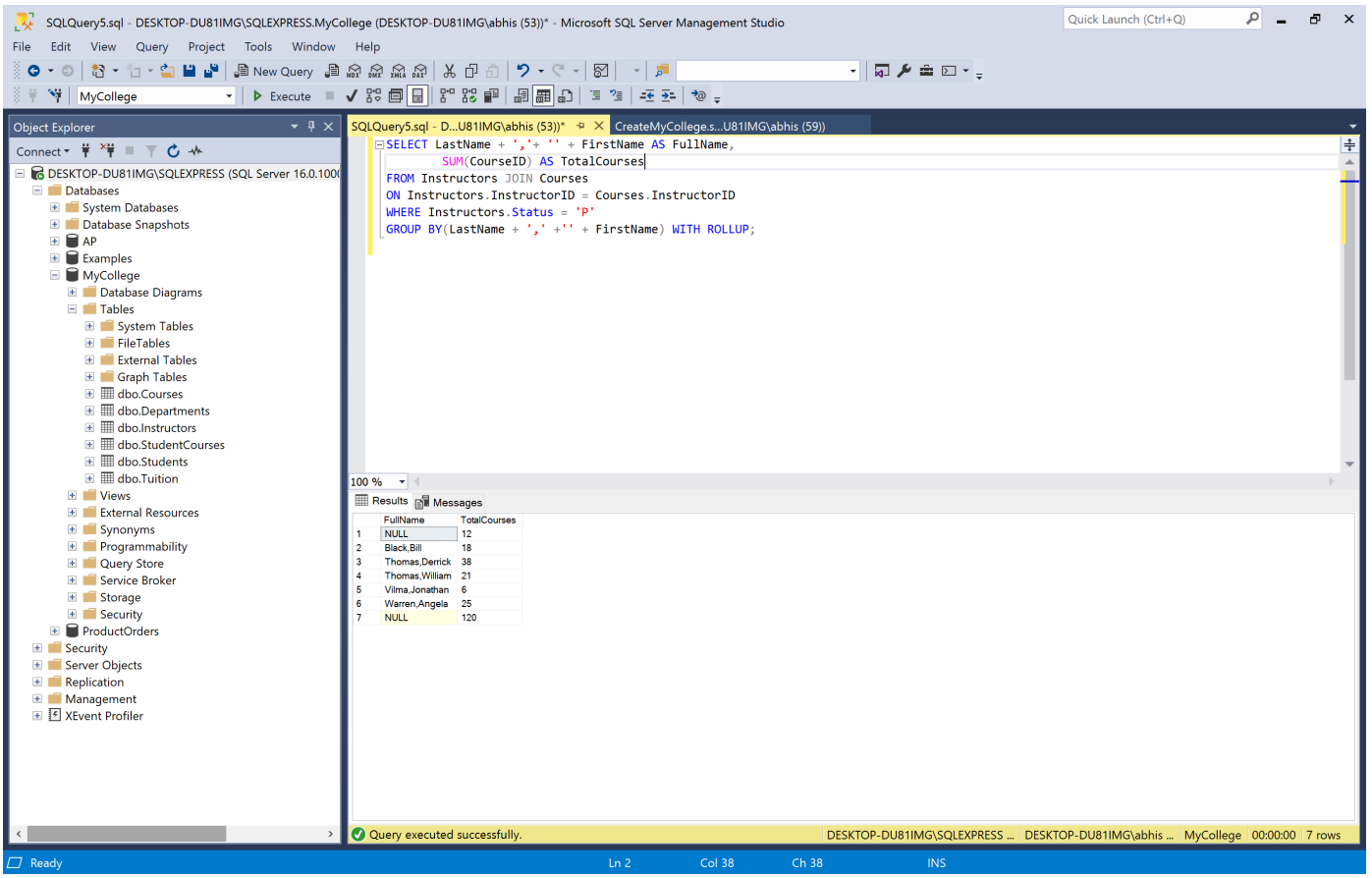
The instructor last name and first name from the Instructors table in this format: Doe, John (Note: If the instructor first name has a null value, the concatenation of thefirst and last name will result in a null value.)

The total number of courses taught for each instructor in the Courses tableUse the ROLLUP operator to include a row that gives the grand total.

Query –

```
SELECT LastName + ',' + ' ' + FirstName AS
        FullName,SUM(CourseID) AS TotalCourses
FROM Instructors JOIN Courses
ON Instructors.InstructorID =
Courses.InstructorID WHERE Instructors.Status
= 'P'
GROUP BY (LastName + ',' + ' ' + FirstName) WITH ROLLUP;
```

ROLLUP operator – Rollup operator helps in providing a summary row for each groupinglevel. It is an extension of GROUP BY Clause.

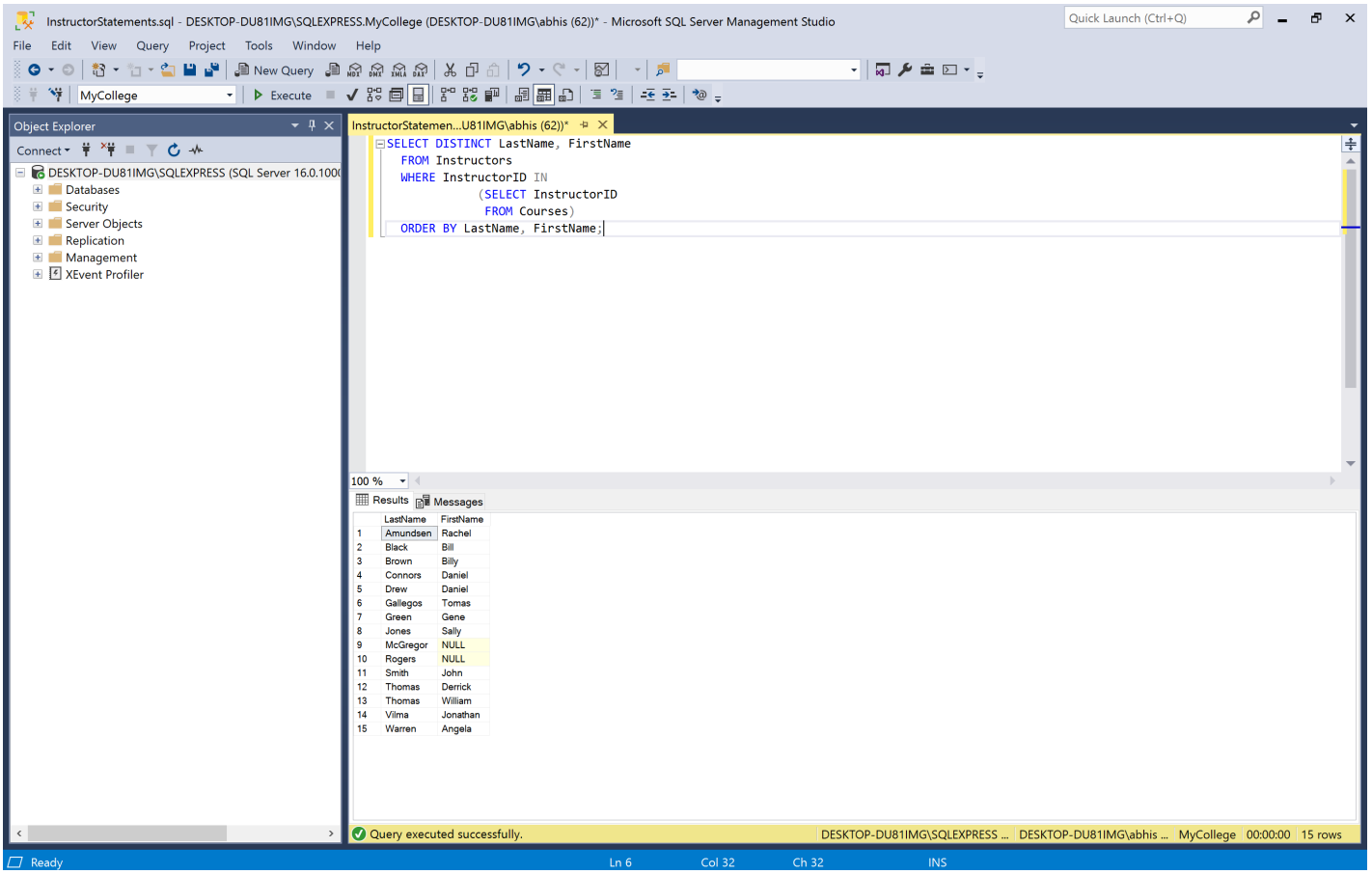


Comment :Successfully executed a displayed the total number of courses taught by part-time instructors as 120, using ROLLUP, GROUPBY, SUM() and JOIN. No of rows : 7

13. Write a SELECT statement that returns the same result set as this SELECT statement,but don't use a join. Instead, use a subquery in a WHERE clause that uses the IN keyword.
SELECT DISTINCT LastName,
FirstNameFROM Instructors i JOIN
Courses c
ON i.InstructorID = c.InstructorID
ORDER BY LastName, FirstName

Query –

```
SELECT DISTINCT LastName,
FirstNameFROM Instructors
WHERE InstructorID IN
          (SELECT
           InstructorIDFROM
           Courses)
ORDER BY LastName, FirstName;
```

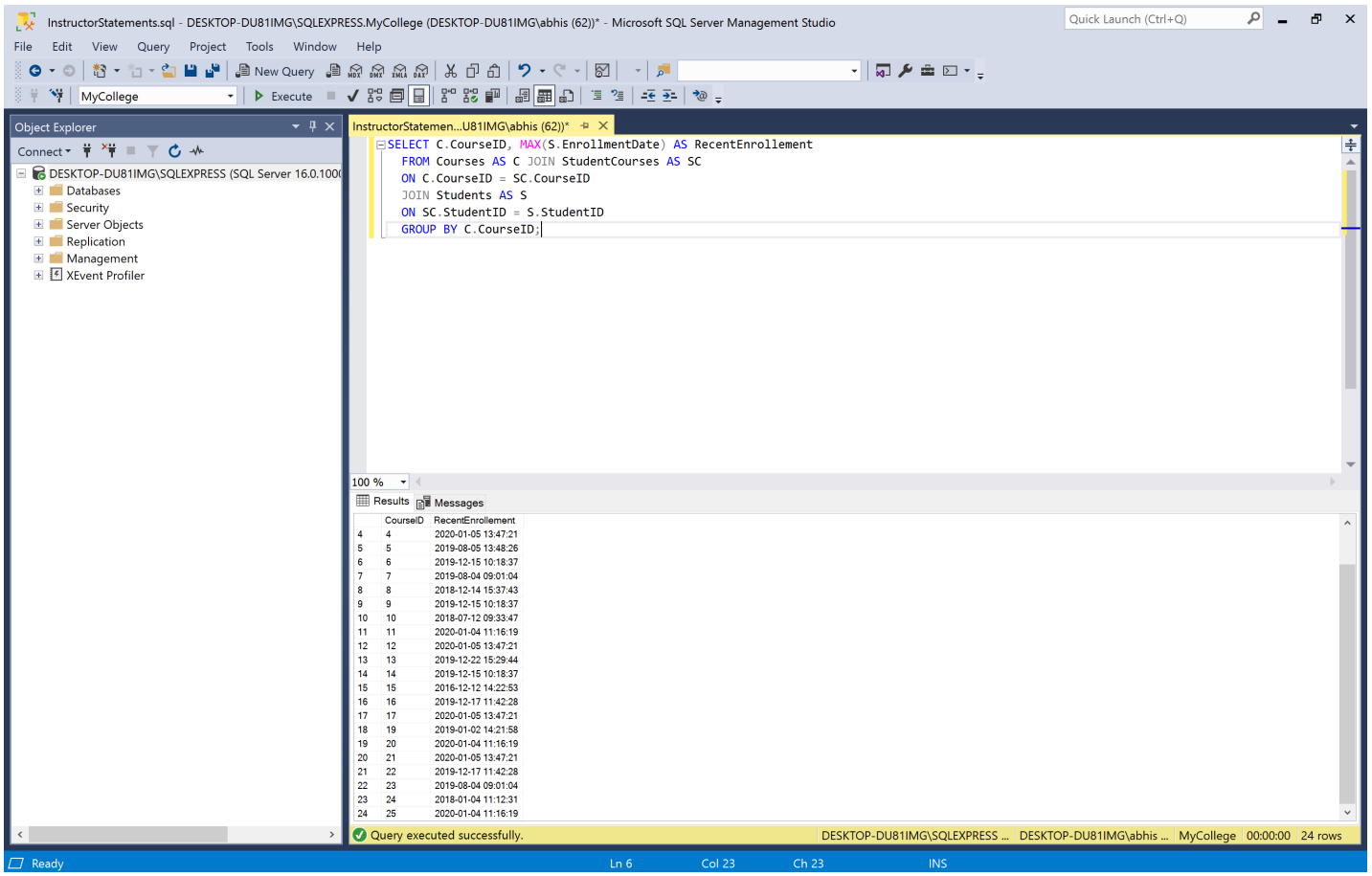


Comment :Successfully executed the SELECT without using JOIN, using logic ofSubquery via 'IN', No of rows : 15

14. Write a SELECT statement that returns one row for each course with these columns:The CourseID column from the Courses table
The most recent enrollment date for that course from the Students table
Change the SELECT statement to a CTE. Then, write a SELECT statement that returns 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.

SELECT Query –

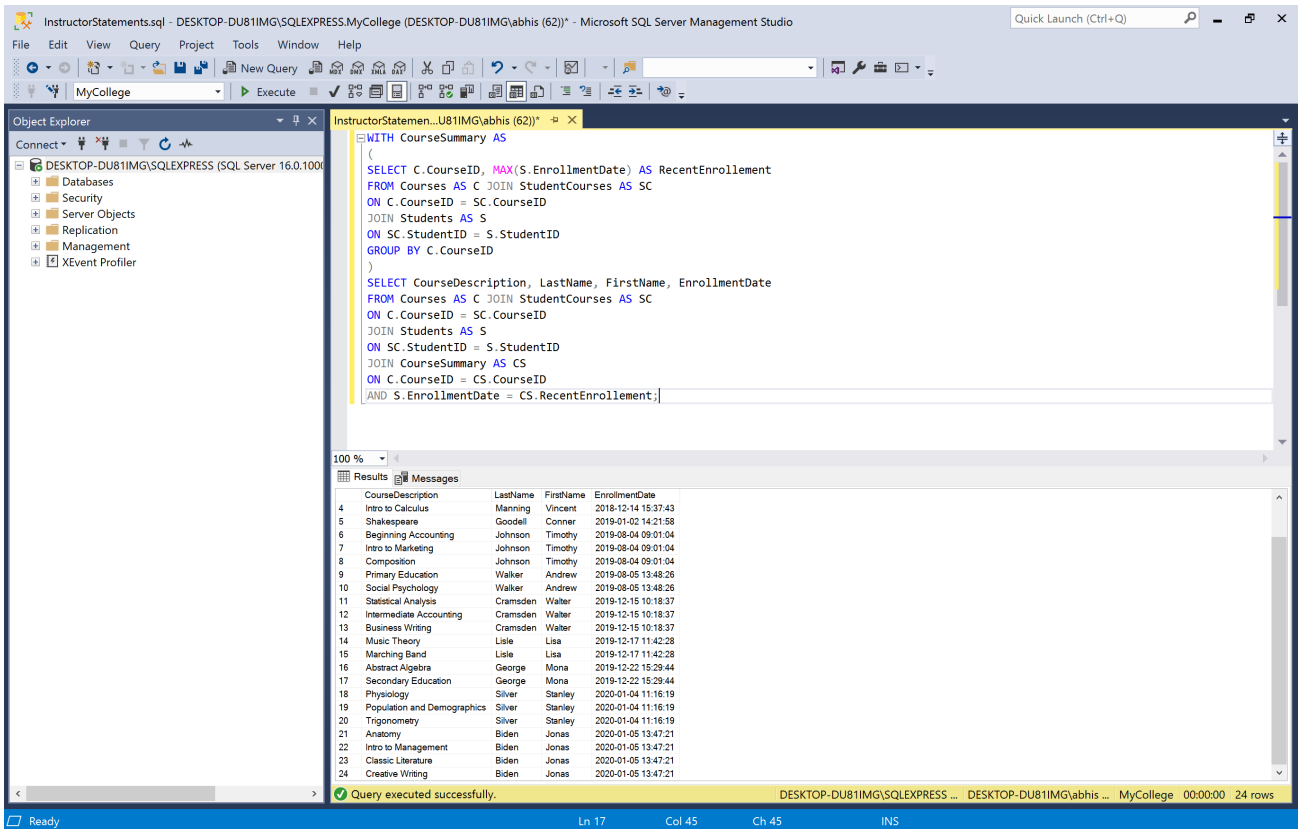
```
SELECT C.CourseID, MAX(S.EnrollmentDate) AS
RecentEnrollement FROM Courses AS C JOIN StudentCourses
AS SC
ON C.CourseID =
SC.CourseIDJOIN Students
AS S
ON SC.StudentID =
S.StudentIDGROUP BY
C.CourseID;
```



Comment : SELECT statement that returns one row for each course. No of rows : 24

CTE Query –

```
WITH CourseSummary
AS(
SELECT C.CourseID, MAX(S.EnrollmentDate) AS
RecentEnrollementFROM Courses AS C JOIN StudentCourses
AS SC
ON C.CourseID =
SC.CourseIDJOIN Students
AS S
ON SC.StudentID =
S.StudentIDGROUP BY
C.CourseID
)
SELECT CourseDescription, LastName, FirstName,
EnrollmentDateFROM Courses AS C JOIN StudentCourses AS
SC
ON C.CourseID =
SC.CourseIDJOIN Students
AS S
ON SC.StudentID =
S.StudentIDJOIN
CourseSummary AS CS
ON C.CourseID = CS.CourseID
AND S.EnrollmentDate = CS.RecentEnrollement;
```



Comment : CTE _ STATEMENT that returns one row for each course. No of rows : 24

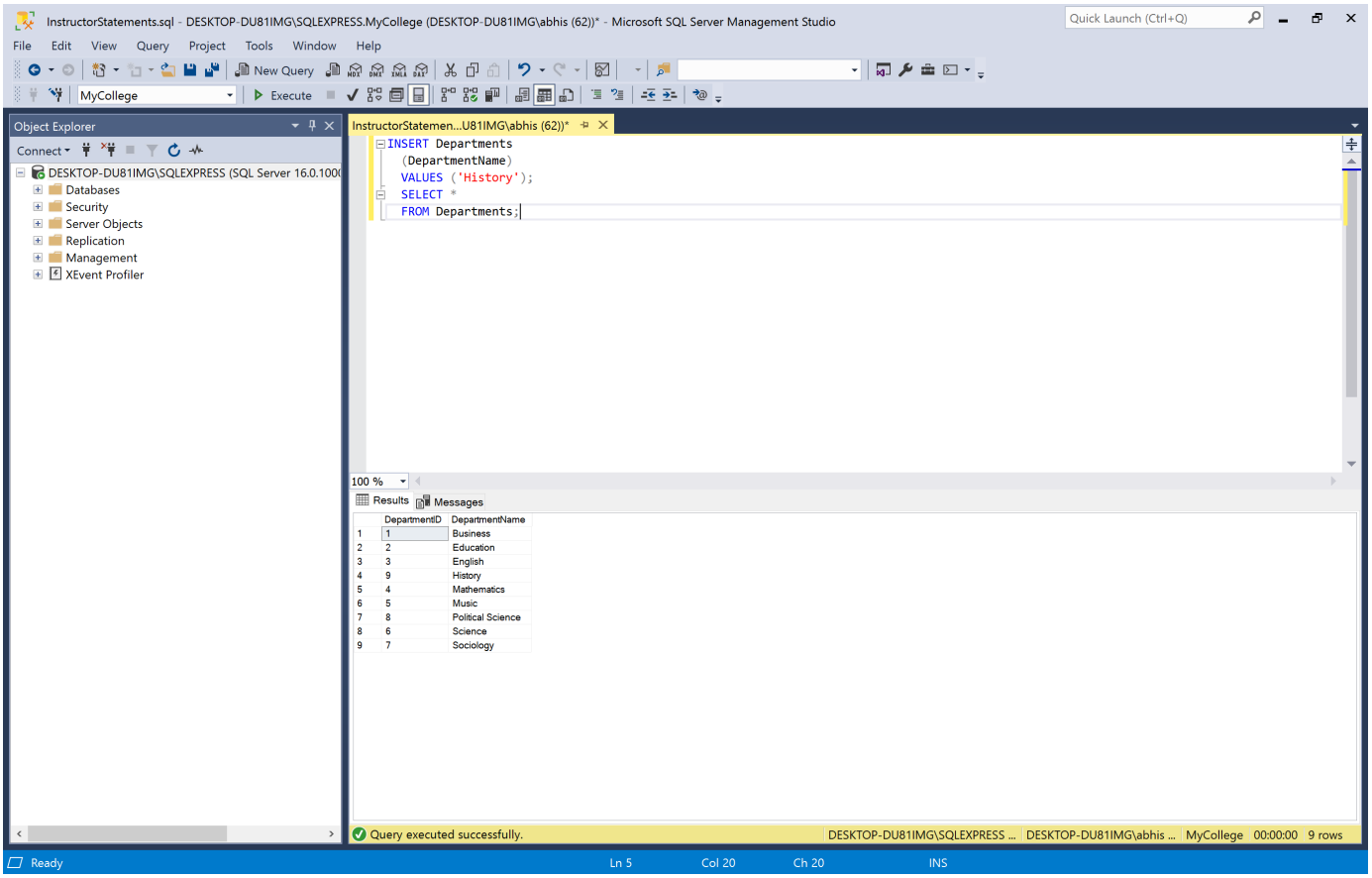
15. Write an INSERT statement that adds this row to the Departments table:DepartmentName: History
Code the INSERT statement so SQL Server automatically generates the value for the DepartmentID column.

Query –

```
INSERT Departments
(DepartmentName)
VALUES ('History');

SELECT *
FROM Departments;
```

InstructorID	The next automatically generated
:LastName:	IDBenedict
FirstName:	Susa
Status:	nP
DepartmentChairman	0
:HireDate:	Today's
AnnualSalary:	date
DepartmentID:	34000.00
InstructorID:	9
LastName:	The next automatically generated
FirstName:	IDAdams
Status:	nul
DepartmentChairman	IF
:HireDate:	1
AnnualSalary:	Today's

Write this statement without using a column list.



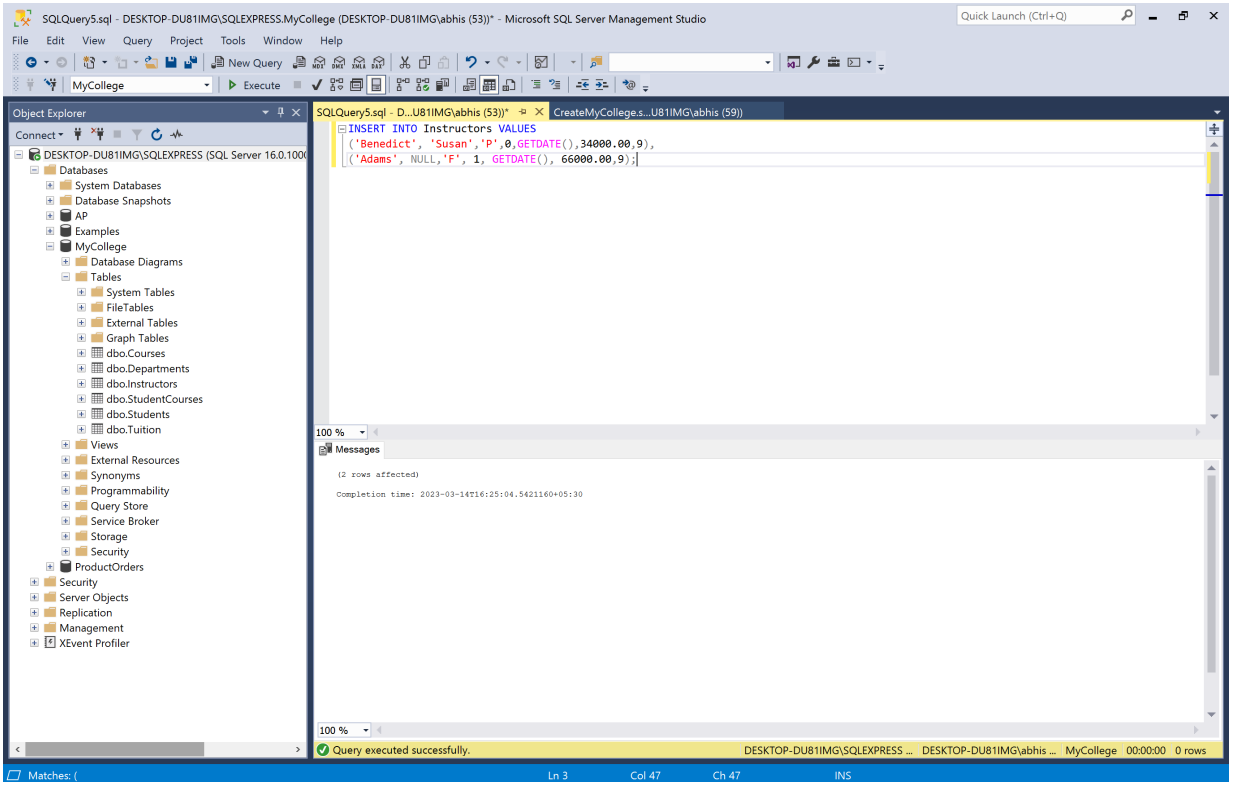
Comment :Successfully Inserted DepartmentName value ‘History’ into Departments tableusing INSERT statement. No of rows : 9

Comments - SQL Server automatically generates a new value for DepartmentID every time you insert a new value into the DepartmentName column of the Departments table.

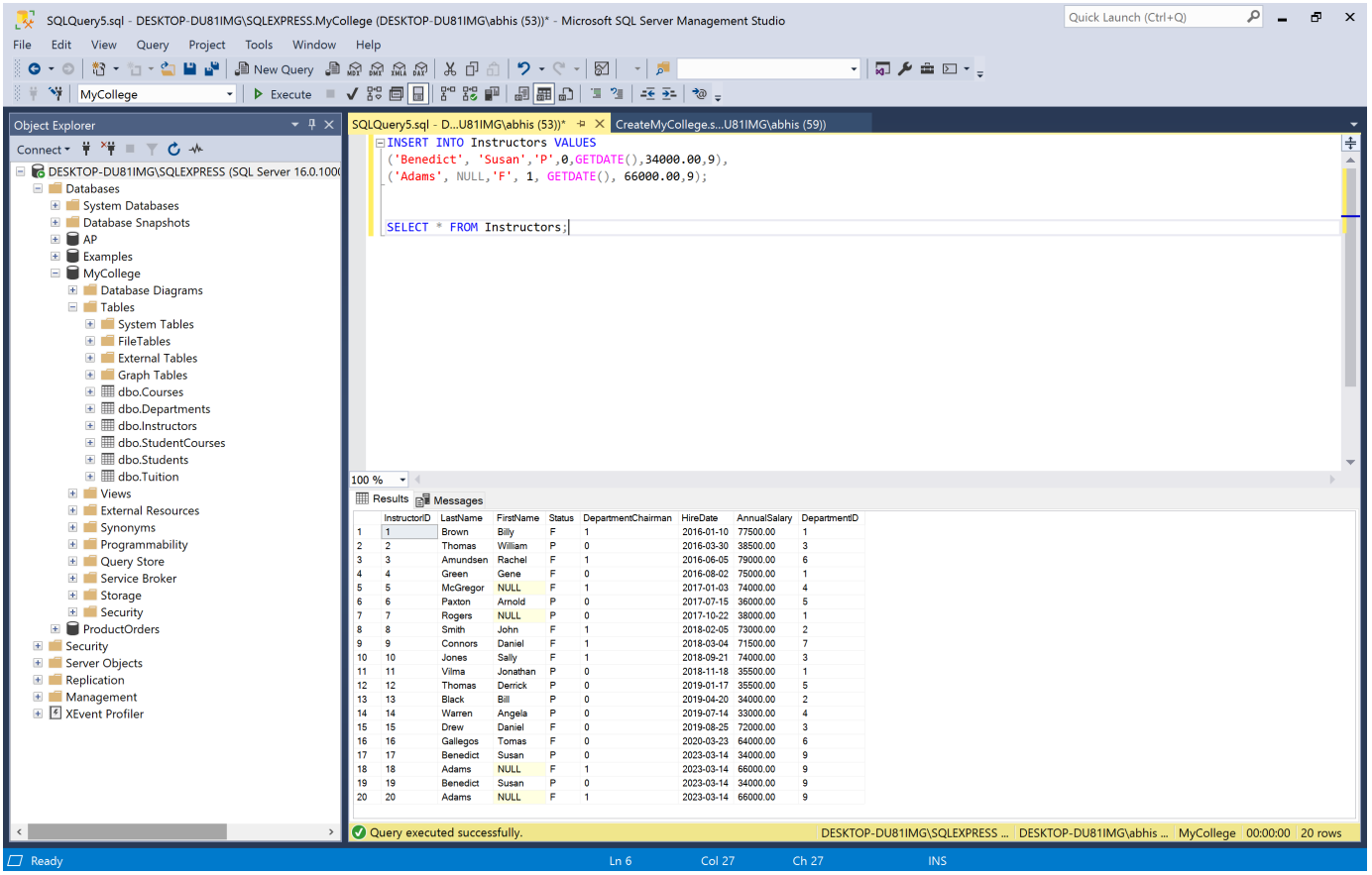
16) Write a single INSERT statement that adds these rows to the Instructors table:

Query :

```
INSERT INTO Instructors VALUES
('Benedict','Susan','P',0,GETDATE(),34000.00,10),
('Adams',NULL,'F',1, GETDATE(),66000.00,10);
SELECT * FROM Instructors;
```



Comment :Wrote a query without using column list. 2 rows affected.



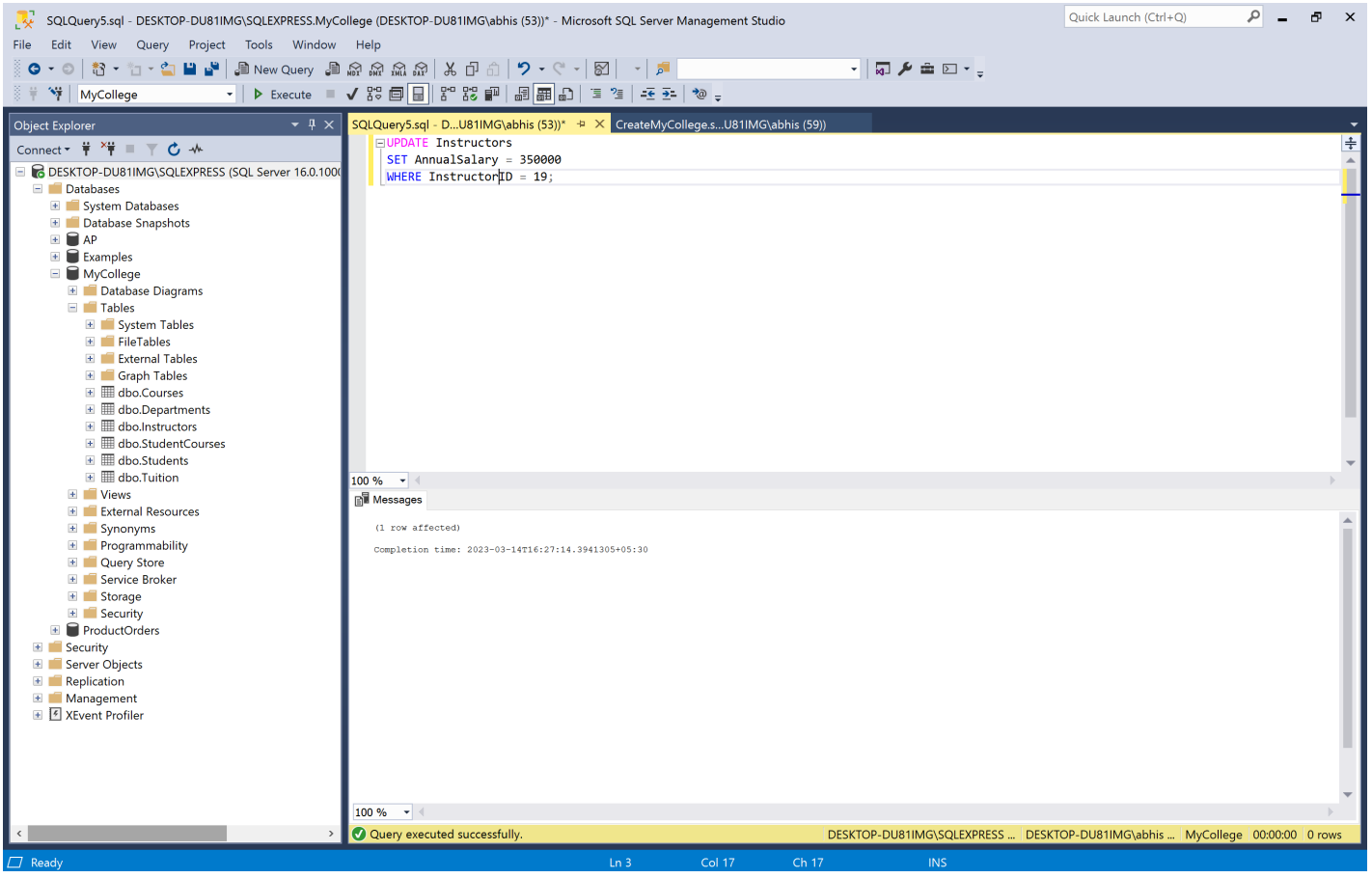
Comments : Successfully inserted the 2 rows of data in Instructions table which arenumbered 34 and 35 in the InstructorID column.

17. Write an UPDATE statement that modifies the first instructor you added in the above question. This statement should change the AnnualSalary column from 34,000 to 35,0000,and it should use the InstructorID column to identify the row.

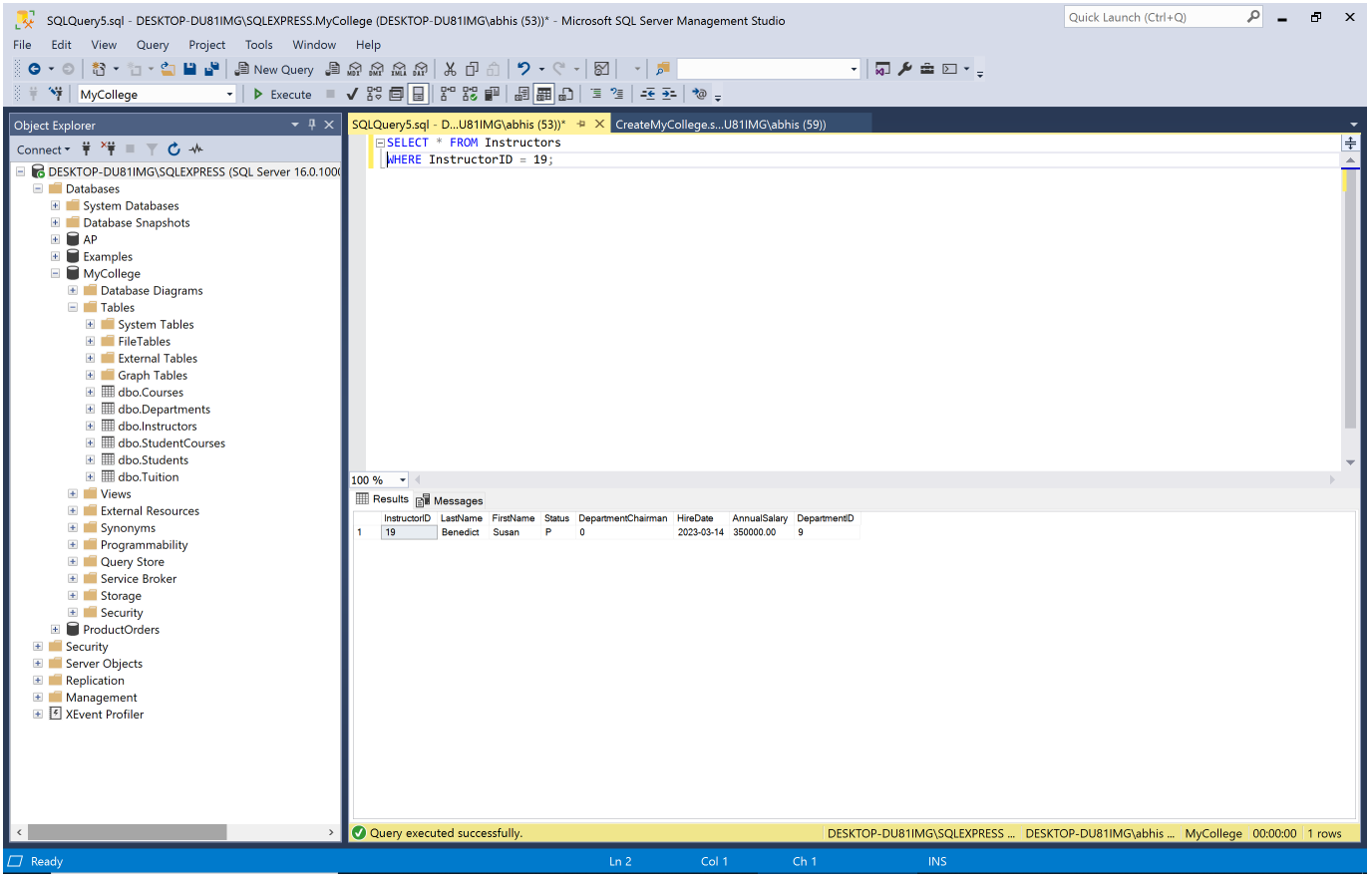
Query :

```
UPDATE Instructors
SET AnnualSalary =
350000WHERE InstructorID
= 20;
```

```
SELECT * FROM
InstructorsWHERE
InstructorID = 20;
```



Comments : Successfully updated the Instructors table with the required value, using UPDATE statement.

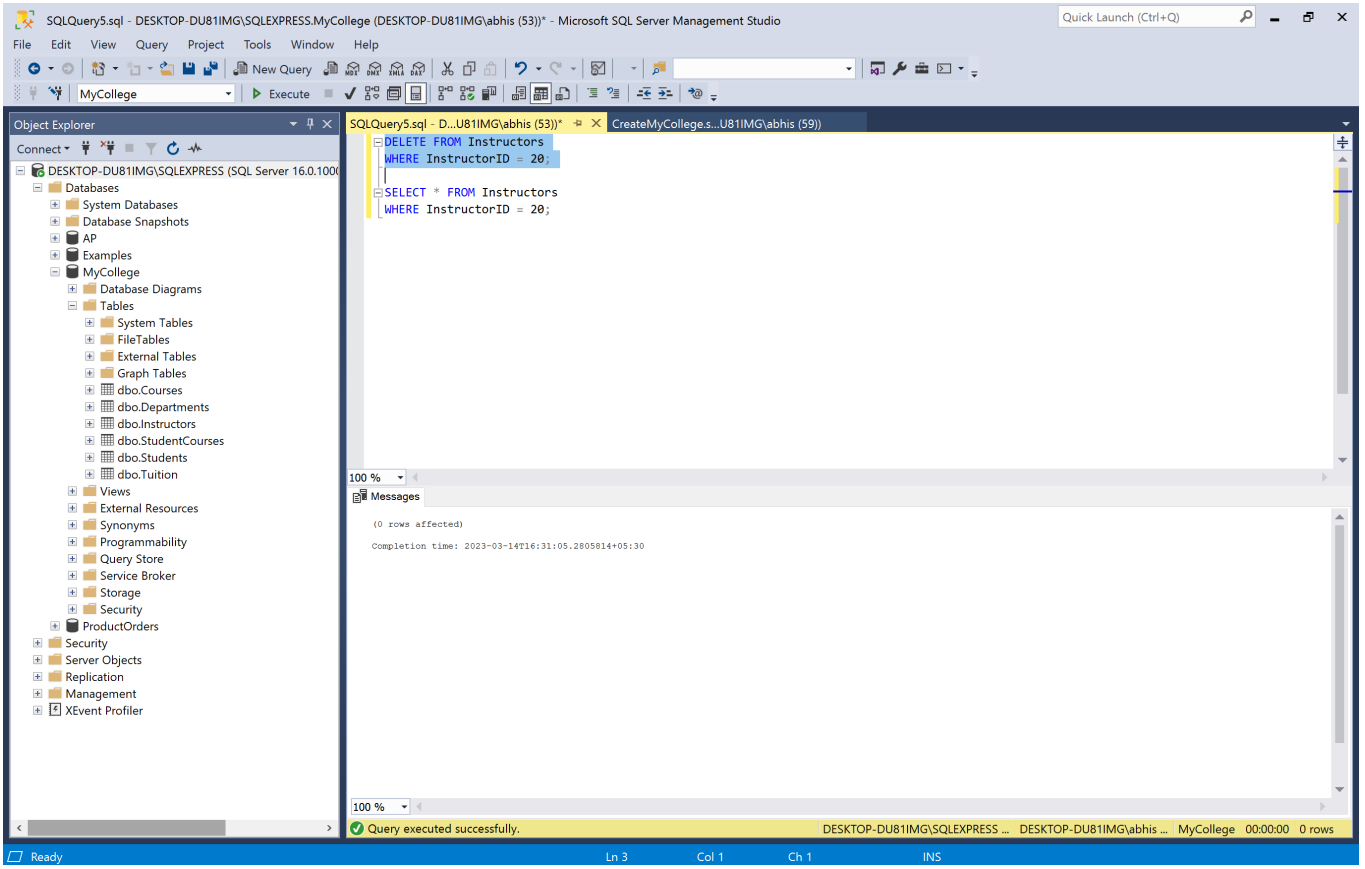


Comments : The updated value, which was raised from 34,000 to 35,0000, should beconfirmed.
No. of rows 1

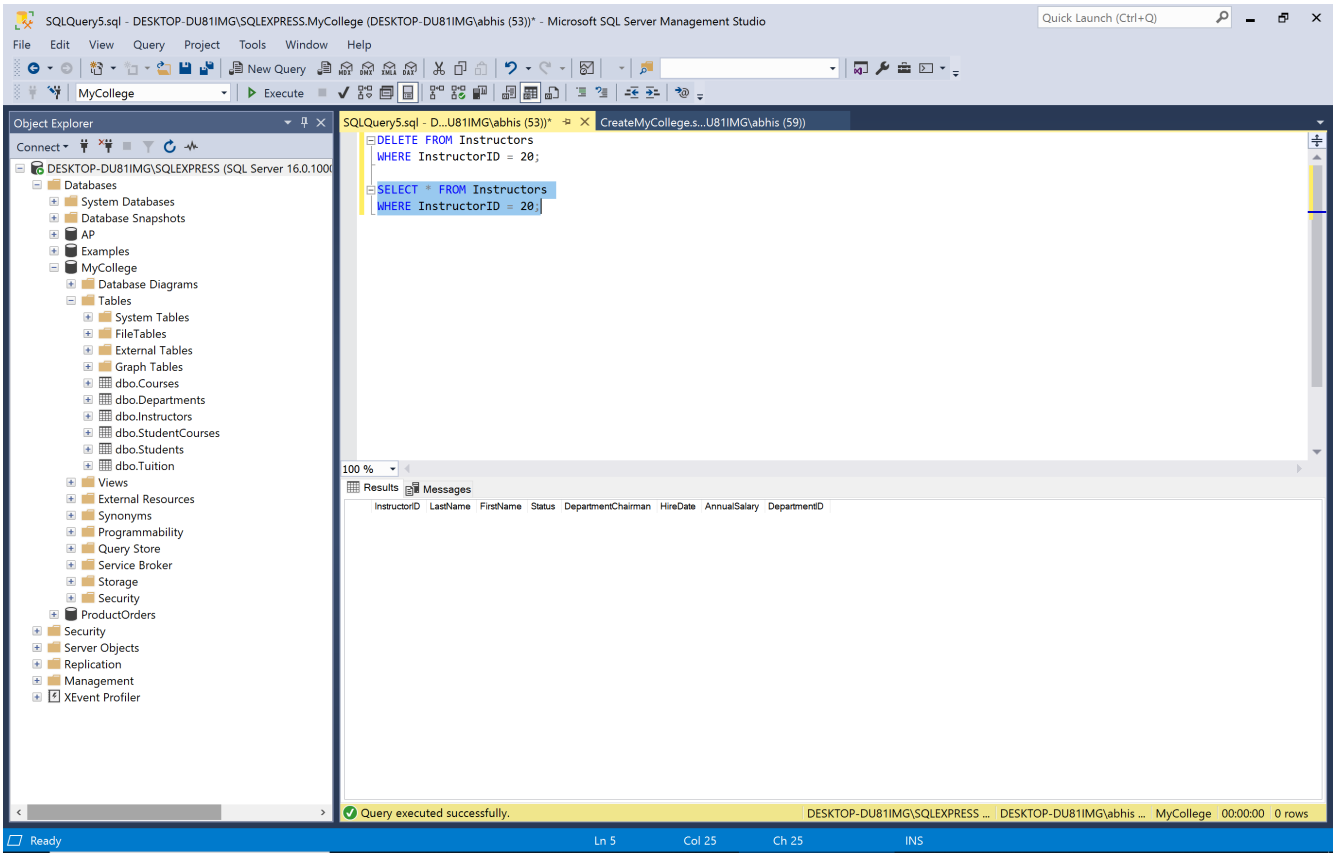
18) Write a DELETE statement that deletes the second instructor you added in question.21 This statement should use the InstructorID column to identify the row.

Query :

```
DELETE FROM
Instructors WHERE
InstructorID = 20;

SELECT * FROM
InstructorsWHERE
InstructorID = 20;
```



Comments : Successfully deleted the second instructor, using DELETE statement.

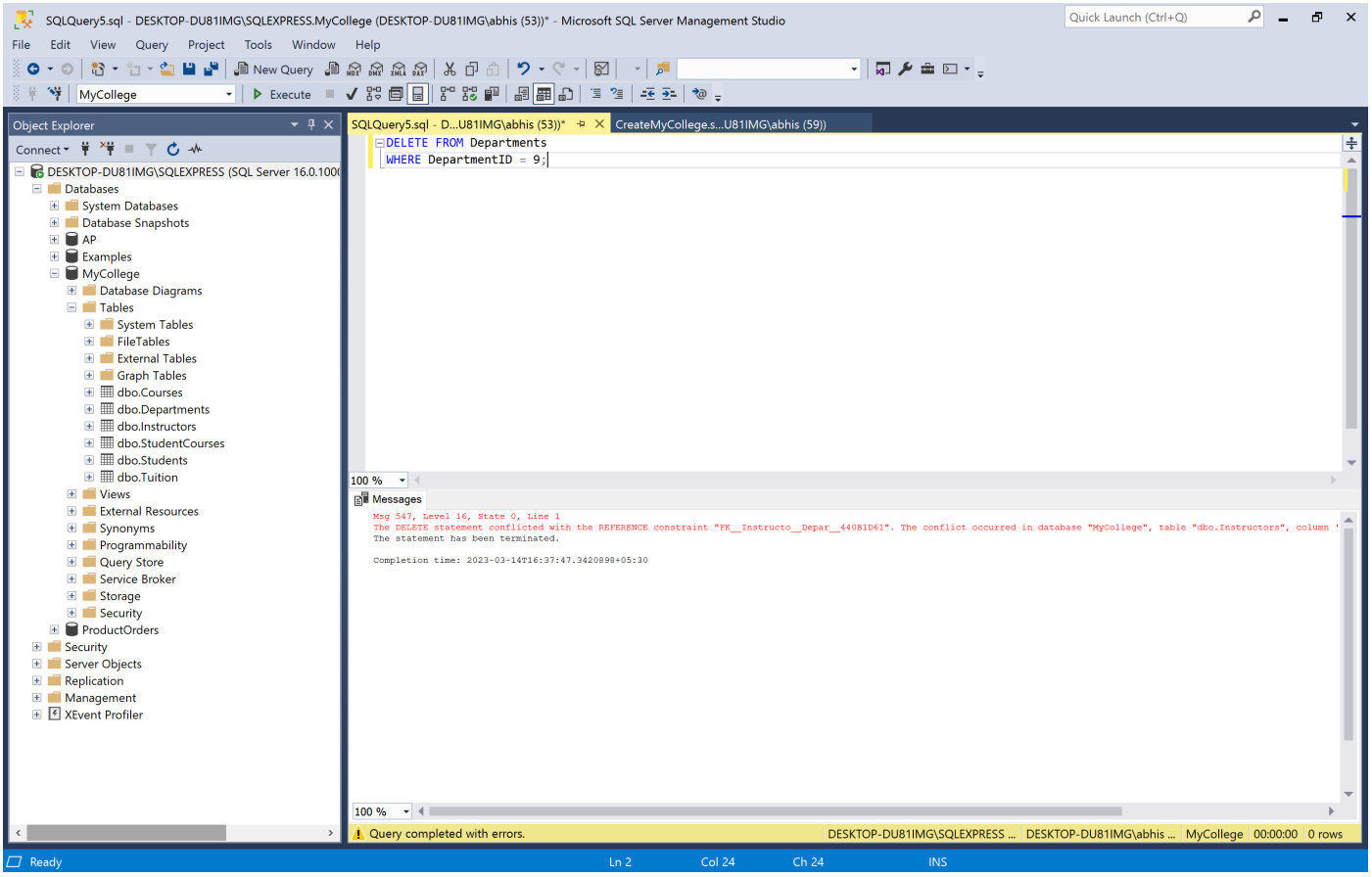


Comments : Instructors table not showing any value, indicating successful deletion ofsecond instructor from Instructors table.

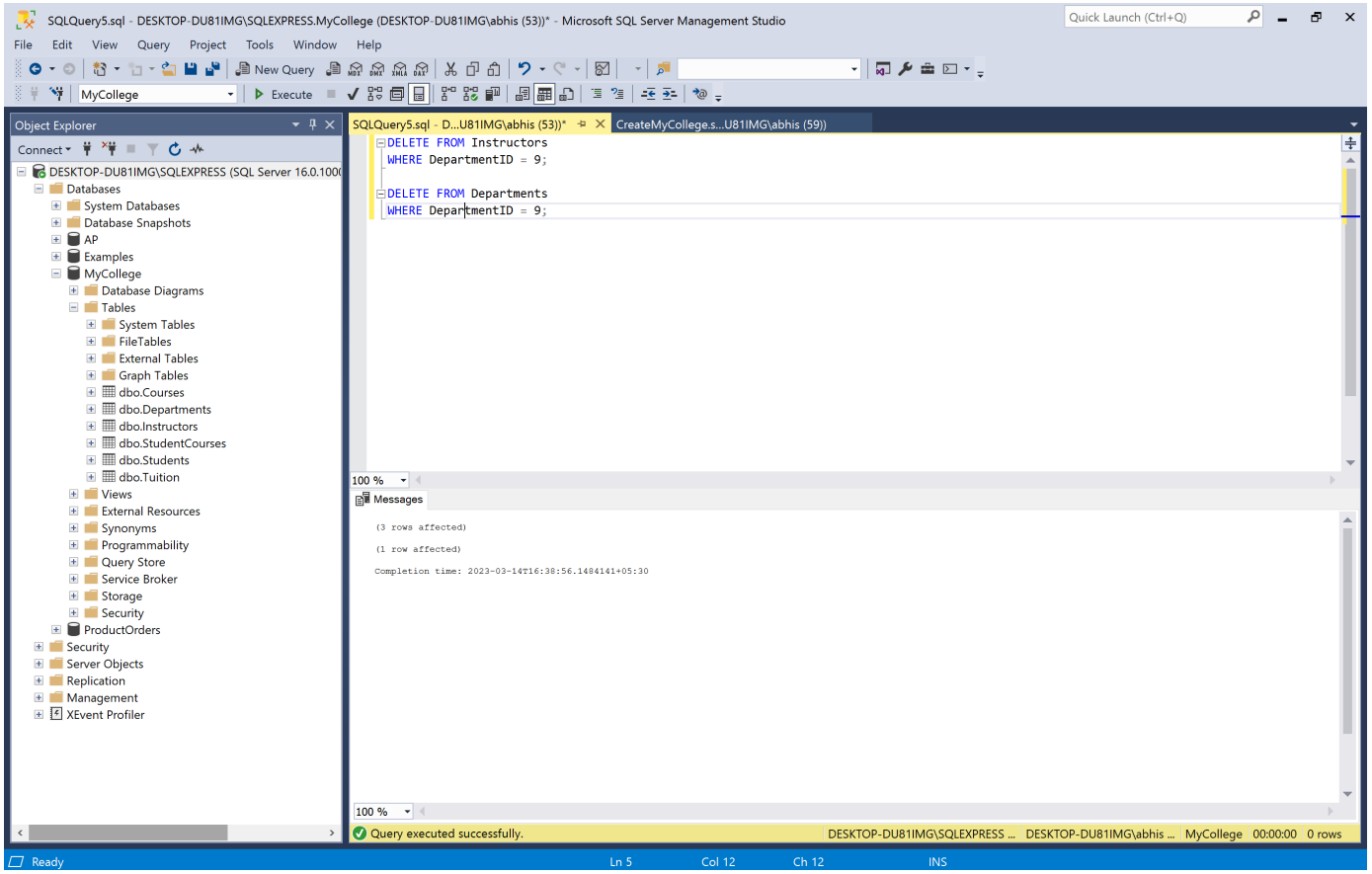
19) Write a DELETE statement that deletes the row in the Departments table that has anID of 9. When you execute this statement, it will produce an error since the department has related rows in the Instructors table. To fix that, precede the DELETE statement withanother DELETE statement that deletes all instructors in this department.

Query :

```
DELETE FROM Instructors
 WHERE DepartmentID = 9;

 DELETE FROM
 DepartmentsWHERE
 DepartmentID = 9;
```



Comments : Error statement since, the department has related rows in the Instructors table. Due to the department's linked rows in the Instructors table, we are experiencing anerror. Prior to deleting them from the Departments dataset, we must first delete those identical needed values from the Instructors table. This is because the Departments and Instructors table and the DepartmentID column have a foreign key–primary key connection.

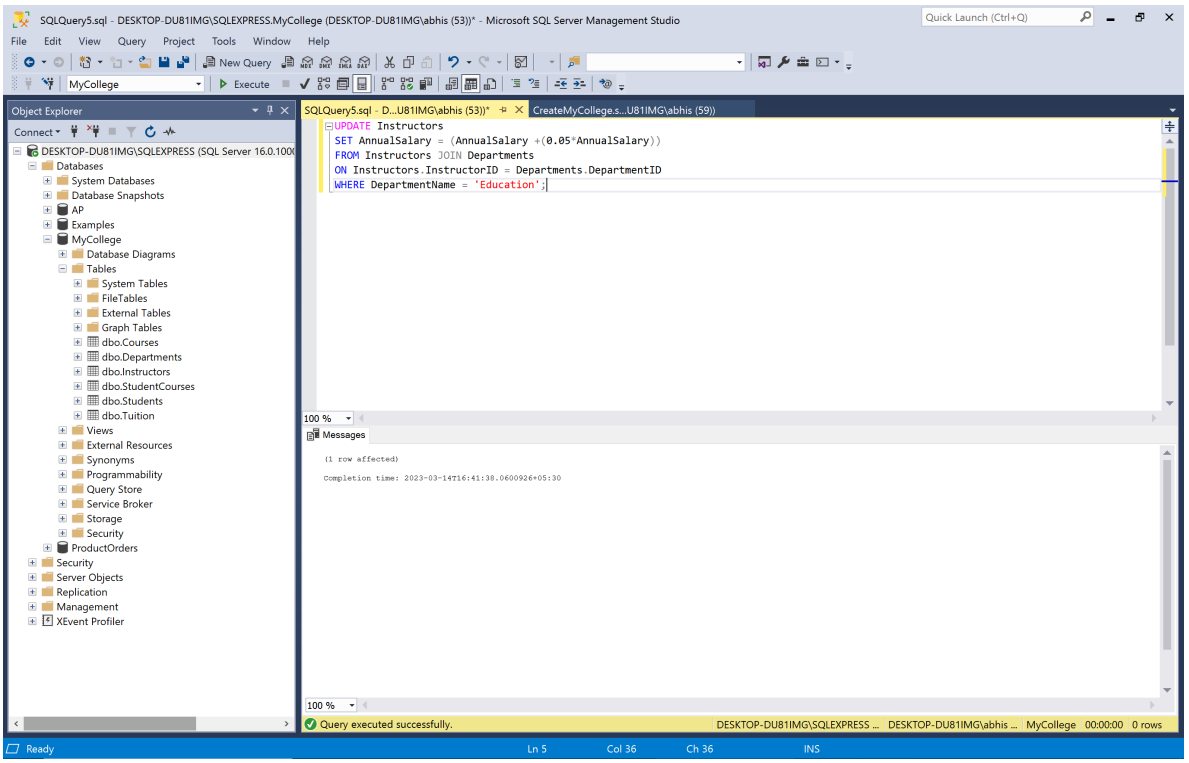


Comments : Successfully deleted the rows from the Departments table with DepartmentID = ‘9’, using DELETE statement.

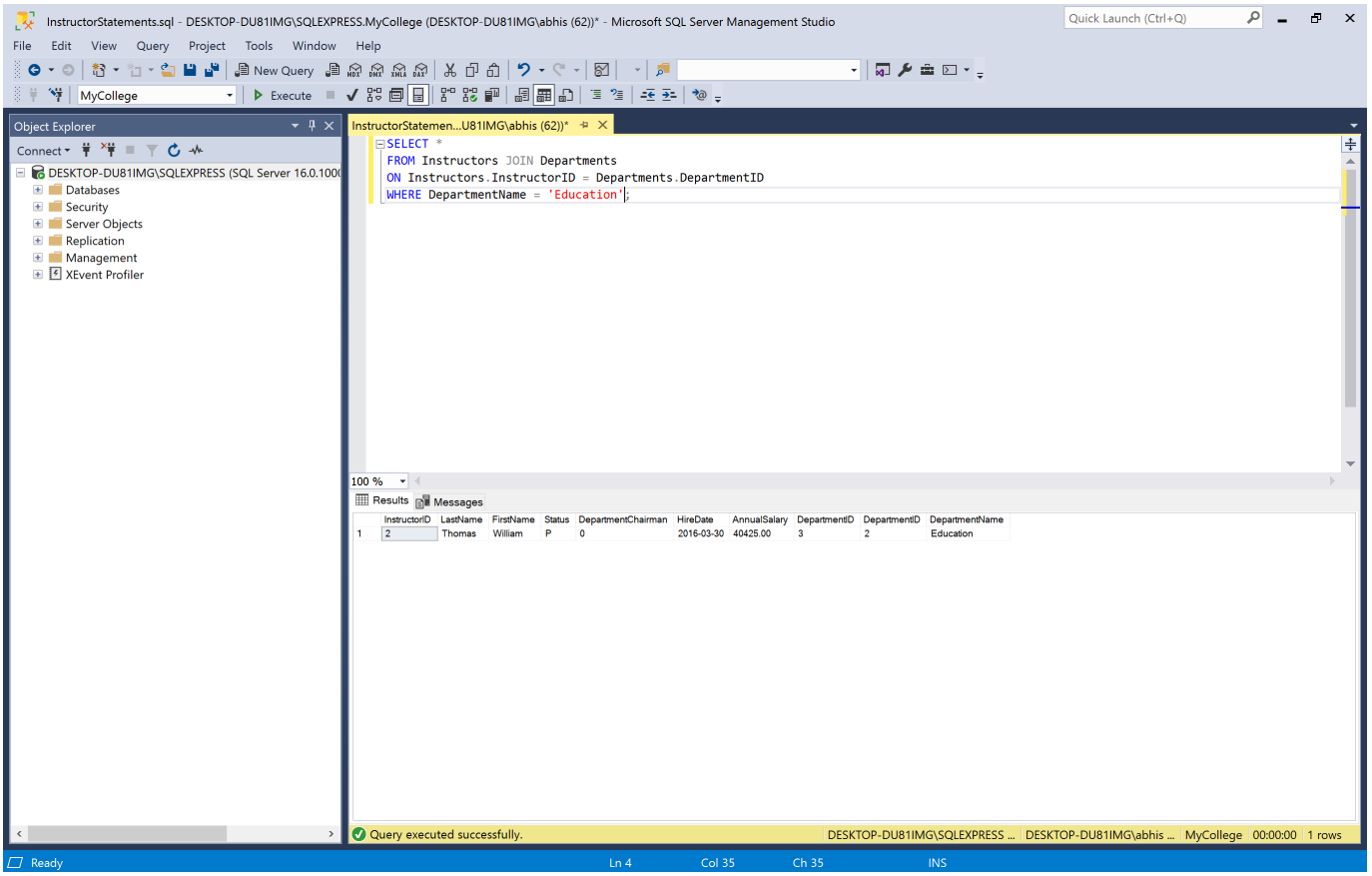
20) Write an UPDATE statement that increases the annual salary for all instructors in the Education department by 5%. To do that, join the Departments and Instructors tables and then filter the rows by the department name

Query :

```
UPDATE Instructors
SET AnnualSalary = (AnnualSalary + (0.05*AnnualSalary))
FROM Instructors JOIN Departments
ON Instructors.InstructorID =
Departments.DepartmentIDWHERE DepartmentName =
‘Education’;

SELECT *
FROM Instructors JOIN Departments
ON Instructors.InstructorID =
Departments.DepartmentIDWHERE DepartmentName =
‘Education’;
```



Comments : Successfully increased the annual salary for all instructors in the Educationdepartment by 5%, using JOIN and UPDATE.

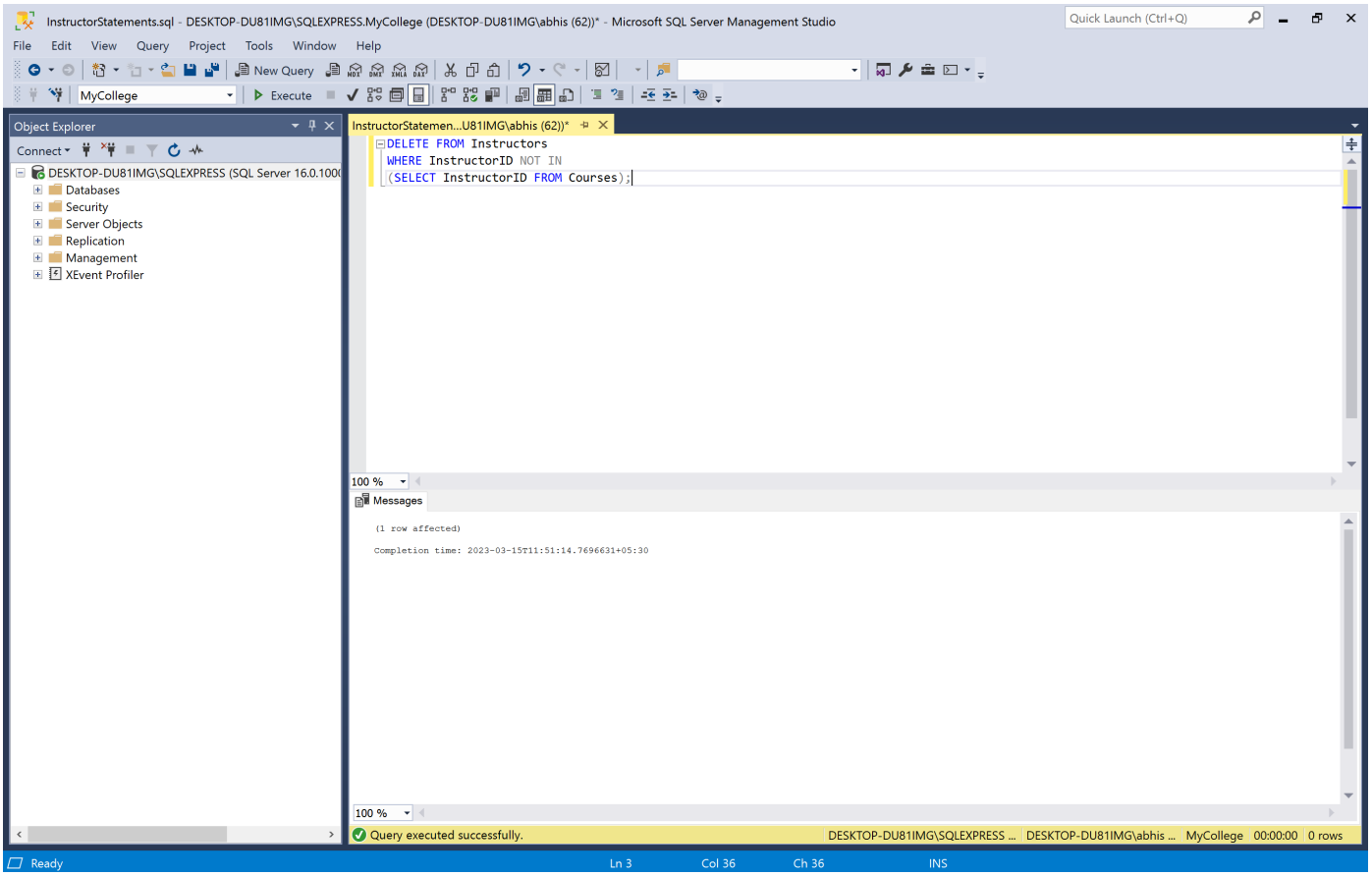


Comments : The updated annual salary of instructors from Education department.

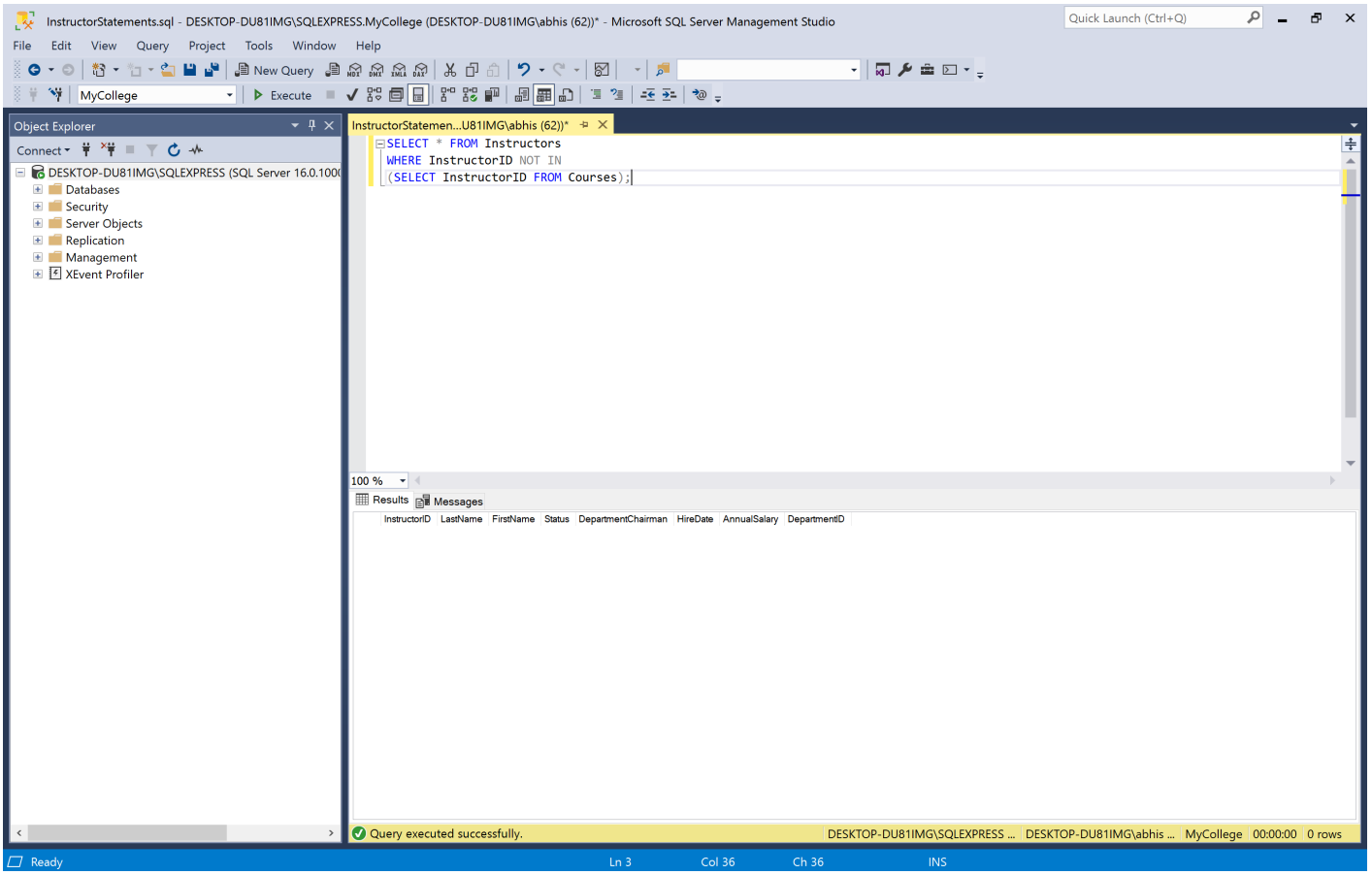
21) Write a DELETE statement that deletes instructors that aren't teaching any courses. To do that, use a subquery in the WHERE clause.

Query :

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

SELECT * FROM
InstructorsWHERE
InstructorID NOT IN
(SELECT InstructorID FROM Courses);
```



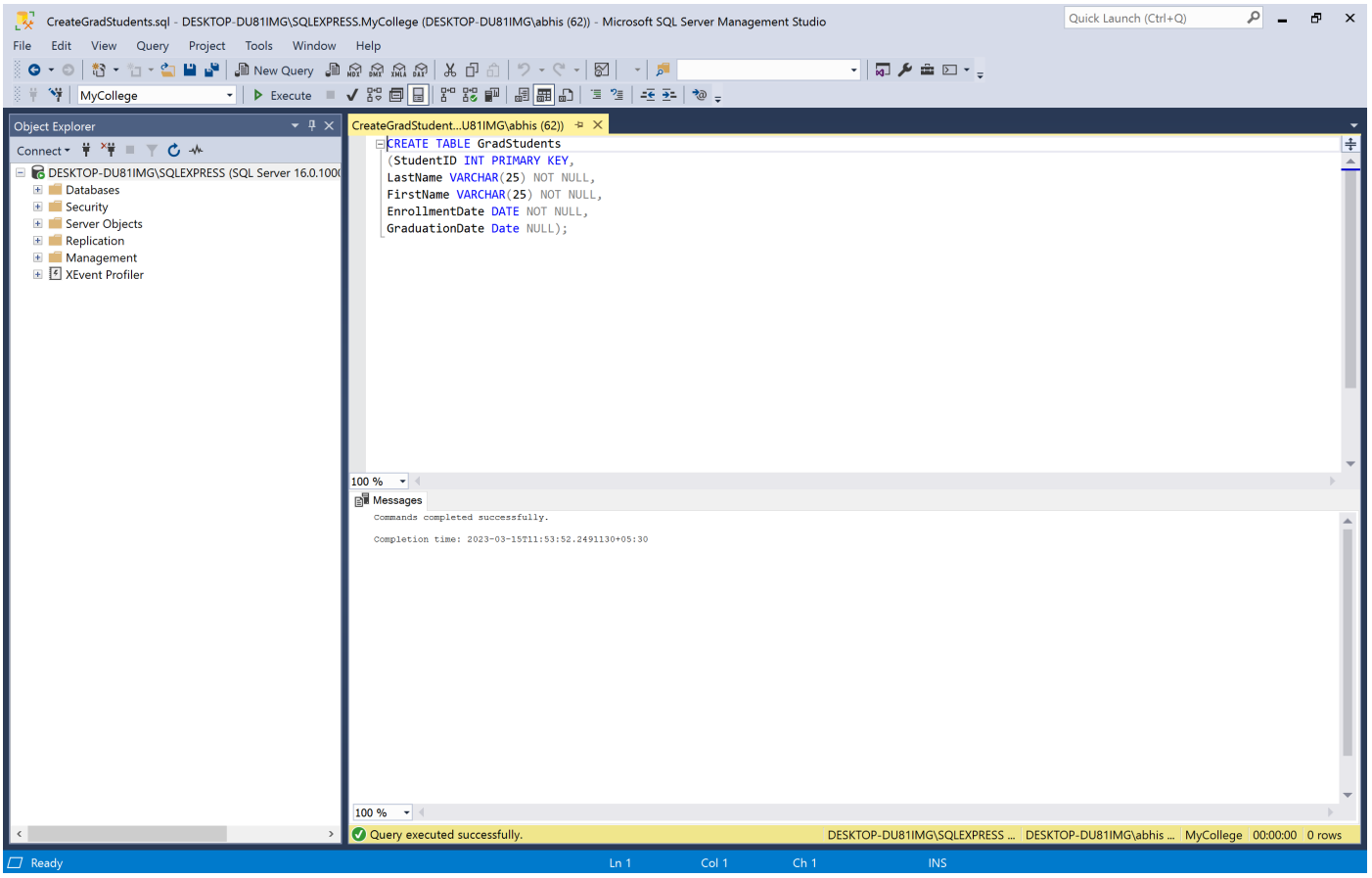
Comments : Successfully deleted 3 rows from Instructors



Comments : Successfully deleted instructors that aren't teaching any courses, usingDELETE and subquery in the WHERE clause.

22) Open the script named CreateGradStudents.sql that's attached above. Run this file tocreate a table named GradStudents. This table has the same columns as the Students table, but the StudentID column isn't defined as an identity column.
Query :

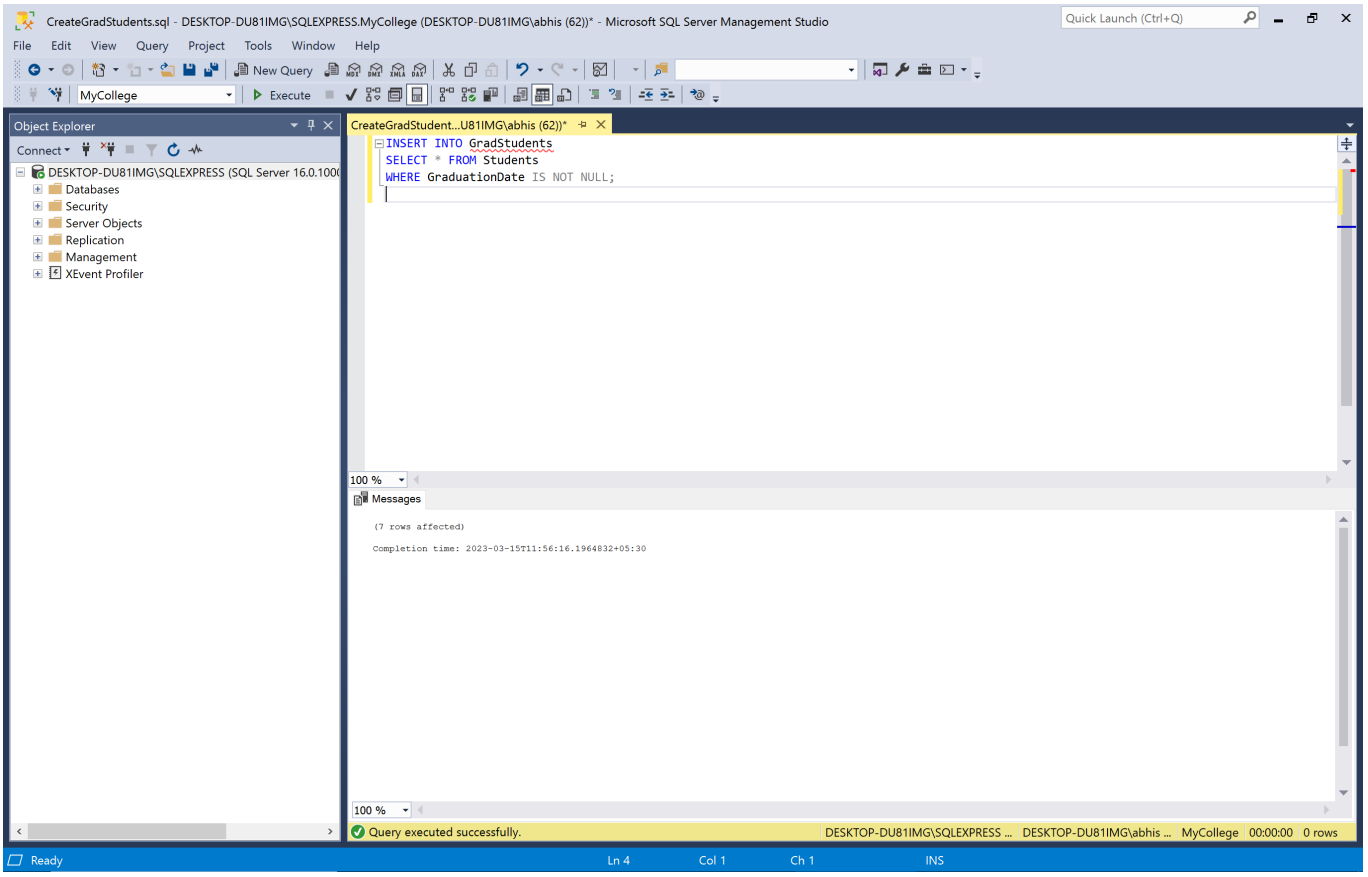
```
CREATE TABLE GradStudents
(StudentID INT PRIMARY KEY,
LastName VARCHAR(25) NOT NULL,
FirstName VARCHAR(25) NOT NULL,
EnrollmentDate DATE NOT NULL,
GraduationDate DATE NULL;
```



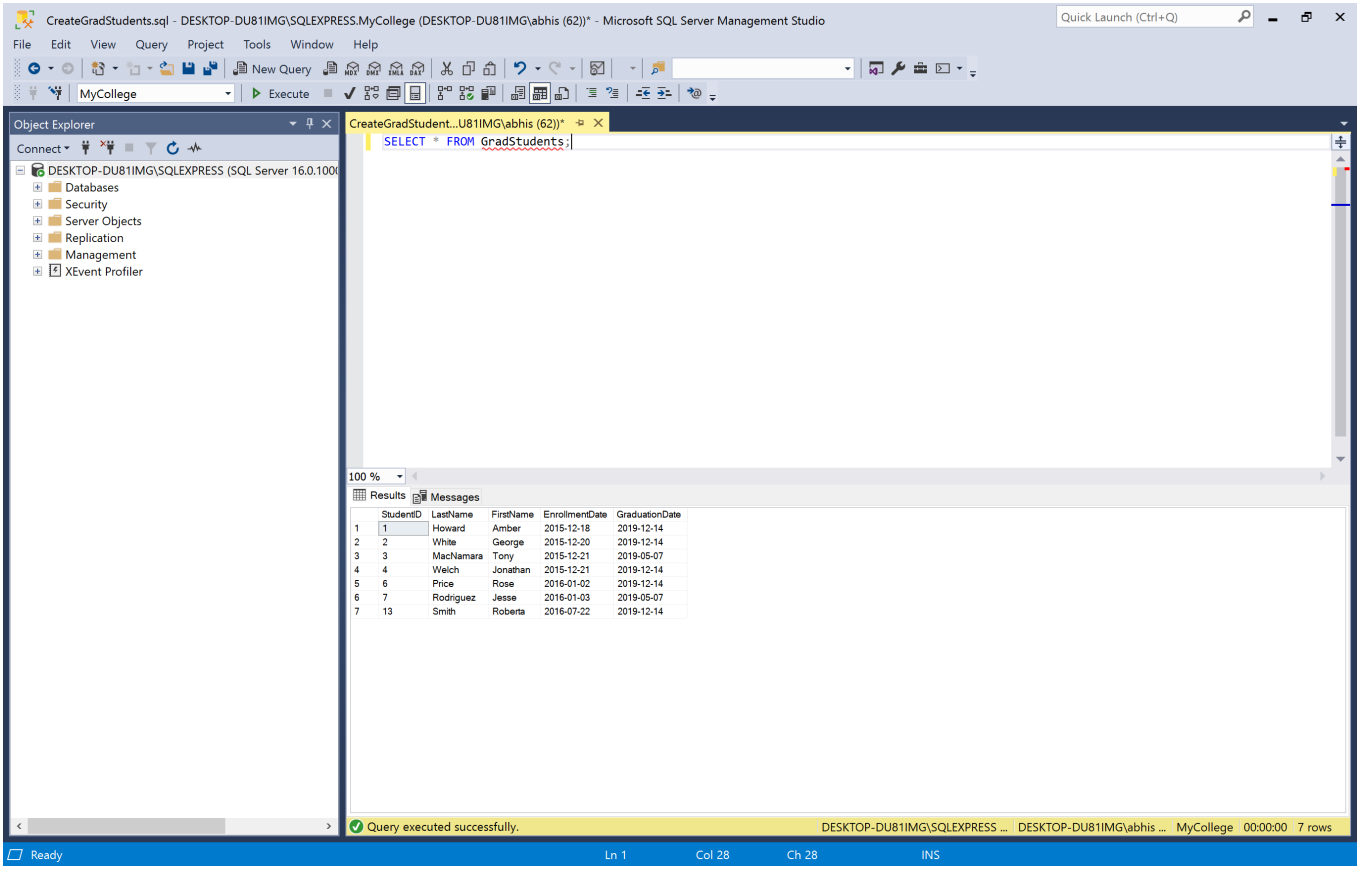
Comments : Successfully created a table named GradStudents, using the attached file given for this project.

23) Write an INSERT statement that inserts rows from the Students table into the GradStudents table. Include only the rows for students that have graduated, and don't use a column list.
Query :

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

SELECT * FROM GradStudents;
```



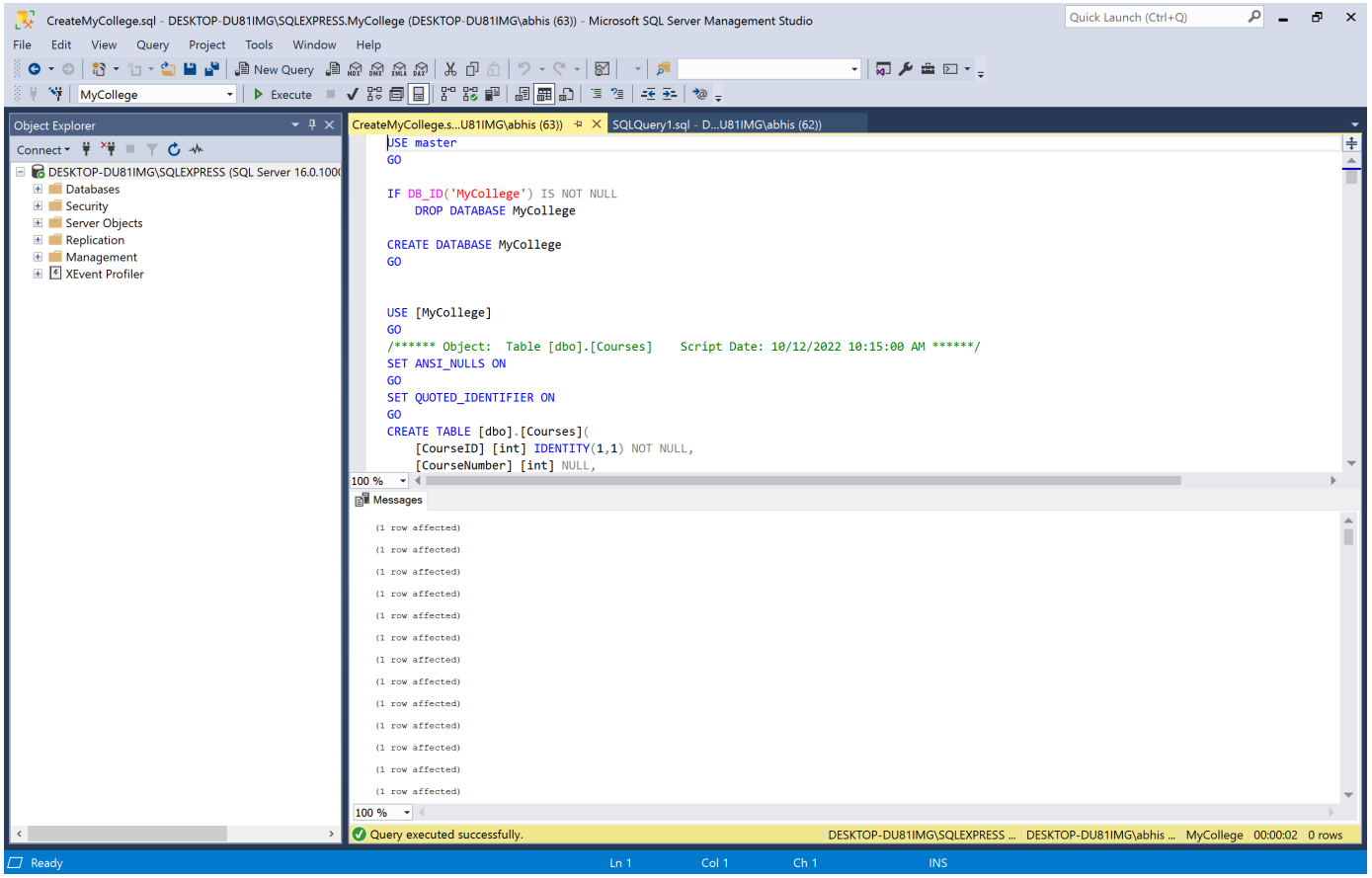
Comments : Successfully inserted rows into the GradStudents table of only students who have graduated, using INSERT statement.



Comments : Verifying the result, No of rows 7

24) Open the script named CreateMyCollege.sql. Then, run this script. That should restore

the data that's in the database. If an error message is displayed indicating that the database is in use, you'll need to close and restart the Management Studio and then run the script again.



Comments : Successfully restored the data in the database named 'MyCollege' by running the given script.

25) Write a SELECT statement that returns these colums from the Students table:

A column that uses the CONVERT function to return the EnrollmentDate column in this format: MM/DD/YYYY. In other words, use 2-digit months and days and a 4-digit year, and separate each date component with slashes.

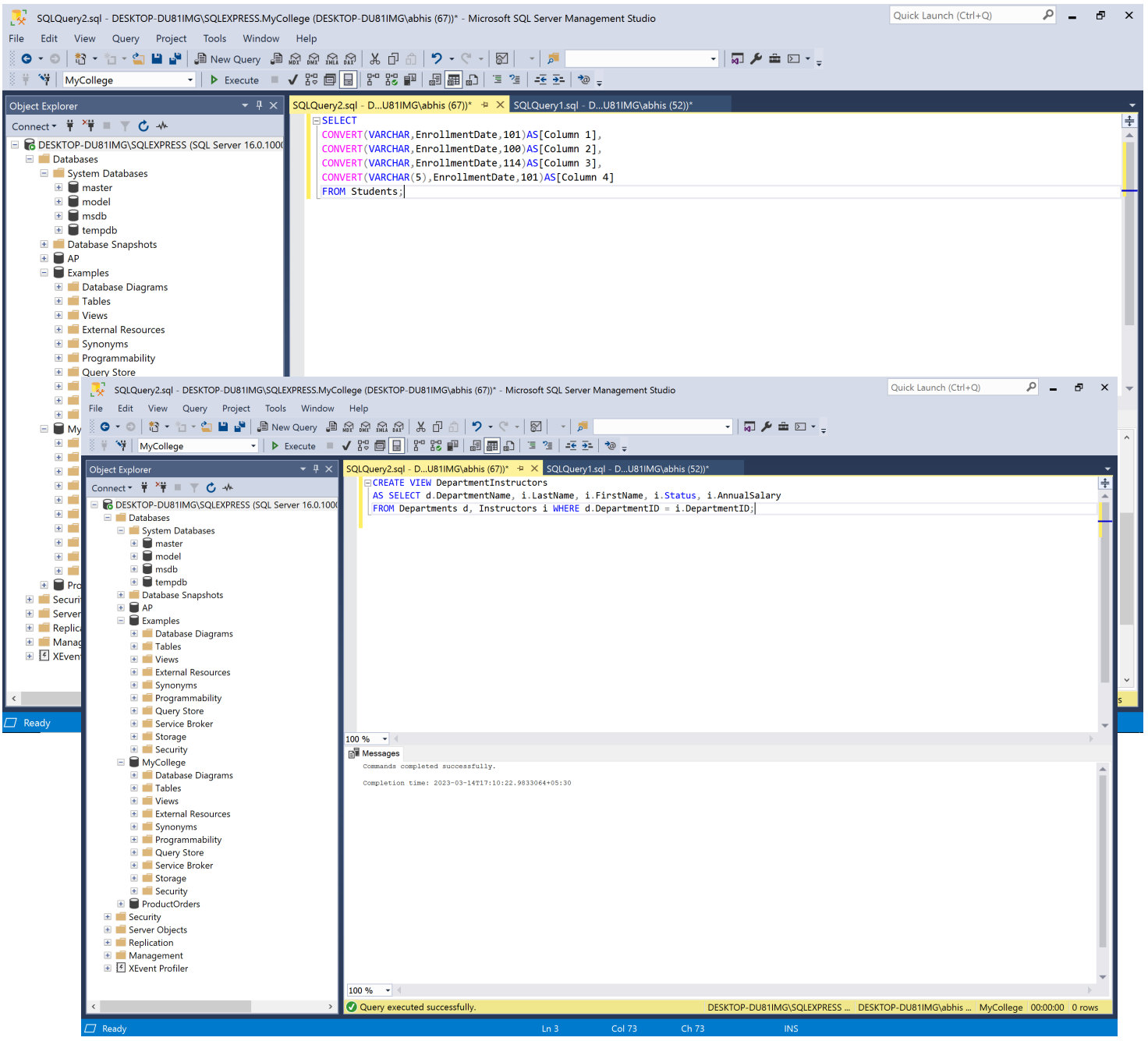
A column that uses the CONVERT function to return the EnrollmentDate column with the date, and the hours and minutes on a 12-hour clock with an am/pm indicator.

A column that uses the CONVERT function to return the EnrollmentDate column with just the time in a 24-hour format, including the milliseconds.

A column that uses the CONVERT function to return the EnrollmentDate column with just the month and day.

Query :

```
SELECT
CONVERT(VARCHAR, EnrollmentDate, 101) AS [Column 1],
CONVERT(VARCHAR, EnrollmentDate, 100) AS [Column 2],
CONVERT(VARCHAR, EnrollmentDate, 114) AS [Column 3],
CONVERT(VARCHAR(5), EnrollmentDate, 101) AS [Column 4]
FROM Students;
```



Comments : Successfully returned 4 columns in the required format as stated in the question statement, using CONVERT() function. No of rows 43

PART C, Advanced SQL Skills (views/stores procedures/ functions /scripts) –

Query :

```
CREATE VIEW DepartmentInstructors
AS SELECT d.DepartmentName, i.LastName, i.FirstName,
i.Status ,i.AnnualSalaryFROM Departments d, Instructors i WHERE
d.DepartmentID= i.DepartmentID;
```

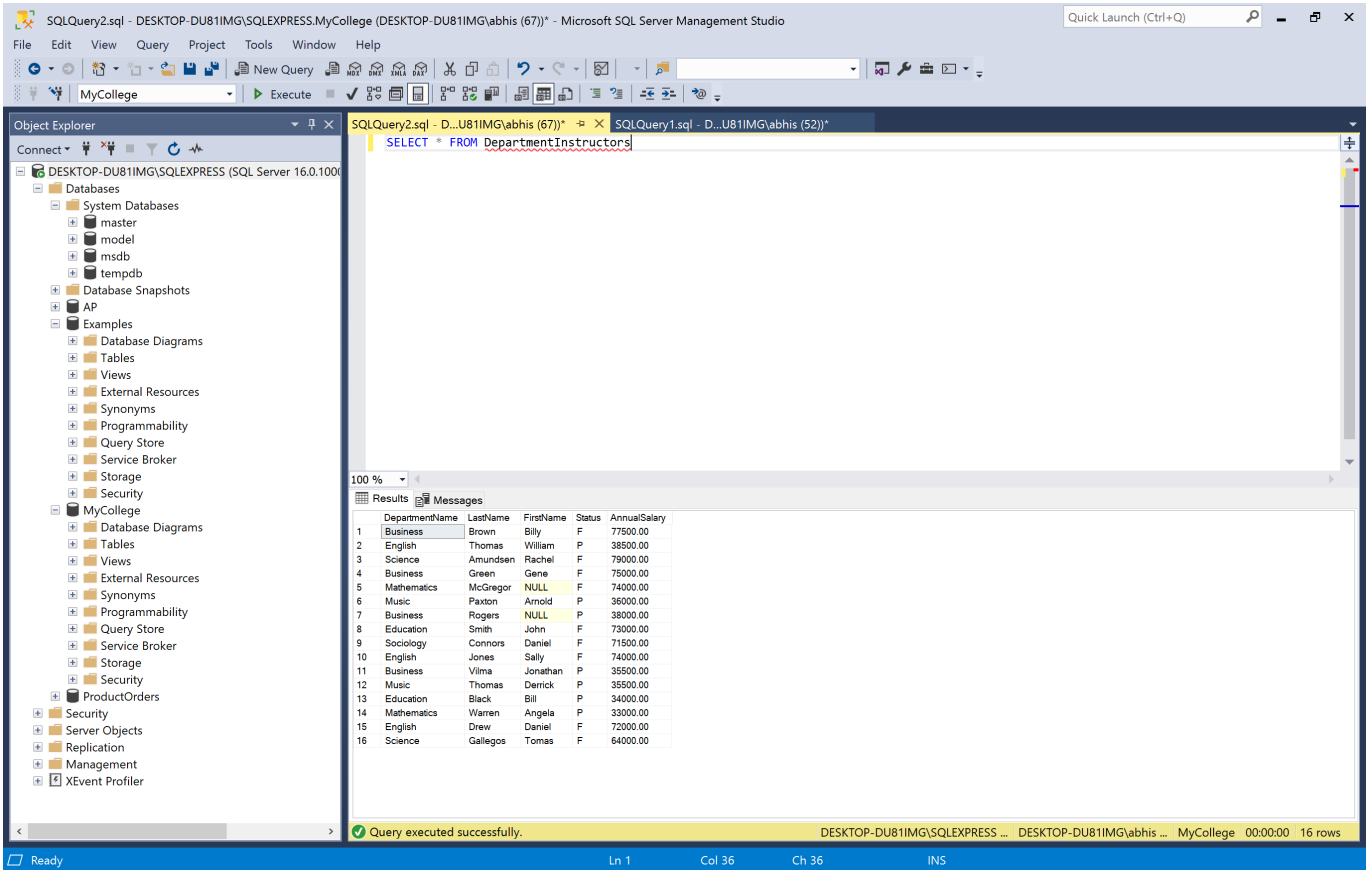
Comments : Successfully created a view named DepartmentIInstructors returning thethree required columns, using CREATE VIEW statement.

2) Write a SELECT statement that returns all the columns from the DepartmentInstructors view that you created in exercise 1.

Query :

```
SELECT *FROM DepartmentInstructors;
```

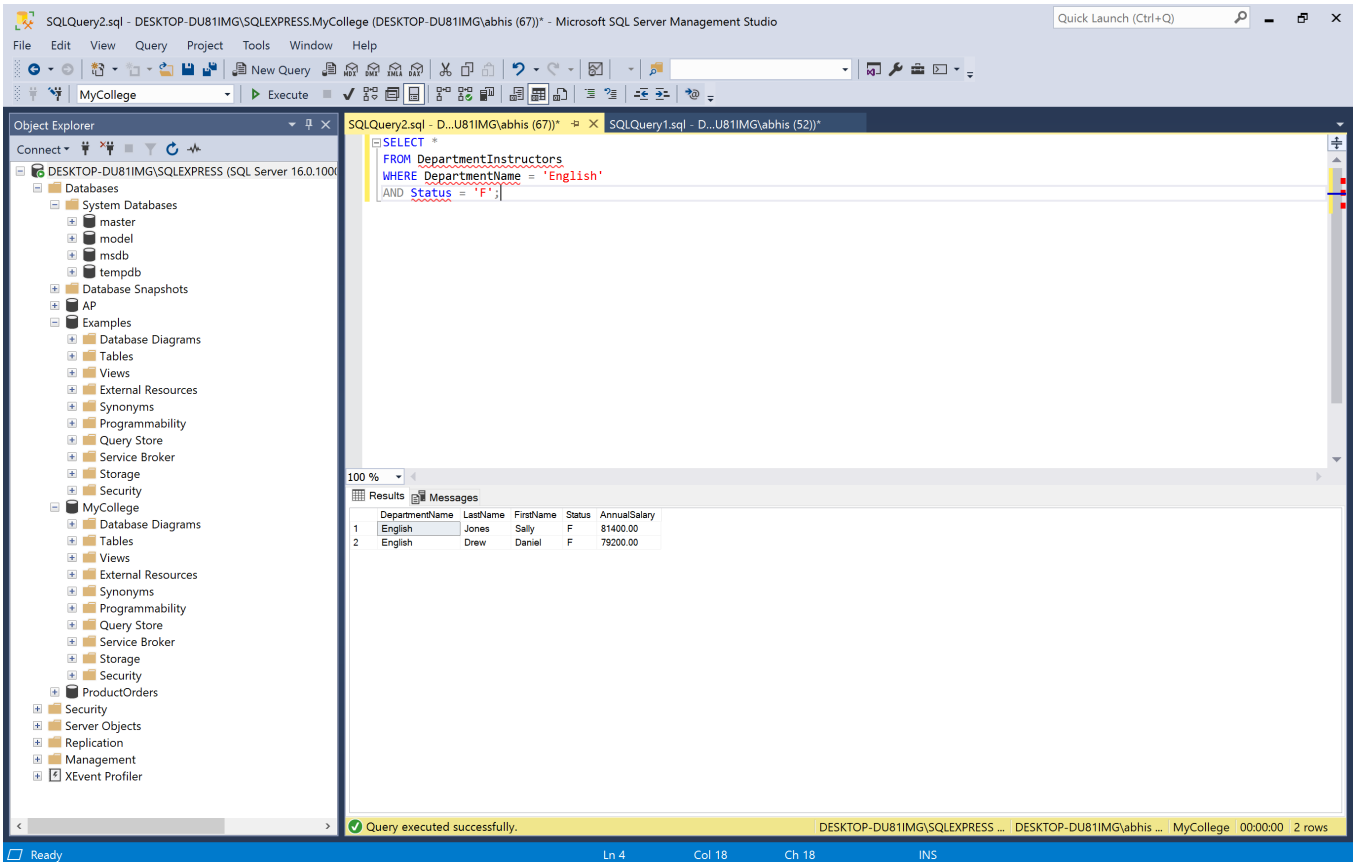
Comments : Successfully displayed query result output, showing all the columns from the DepartmentInstructors view, using SELECT statement.


SQLQuery2.sql - DESKTOP-DU81IMG\SQLEXPRESS.MyCollege (DESKTOP-DU81IMG\abhis (67))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
MyCollege
Execute
Object Explorer
Connect
DESKTOP-DU81IMG\SQLEXPRESS (SQL Server 16.0.1000
Databases
System Databases
master
model
msdb
tempdb
Database Snapshots
AP
Examples
Database Diagrams
Tables
Views
External Resources
Synonyms
Programmability
Query Store
Service Broker
Storage
Security
MyCollege
Database Diagrams
Tables
Views
External Resources
Synonyms
Programmability
Query Store
Service Broker
Storage
Security
ProductOrders
Security
Server Objects
Replication
Management
XEvent Profiler
SQLQuery2.sql - D...U81IMG\abhis (67))*
SQLQuery1.sql - D...U81IMG\abhis (52))*
SELECT * FROM DepartmentInstructors
100 %
Results
Messages
DepartmentName LastName FirstName Status AnnualSalary
1 Business Brown Billy F 77500.00
2 English Thomas William P 38500.00
3 Science Amundsen Rachel P 79000.00
4 Business Green Gene F 75000.00
5 Mathematics McGregor NULL F 74000.00
6 Music Paxton Arnold P 36000.00
7 Business Rogers NULL P 38000.00
8 Education Smith John F 73000.00
9 Sociology Connors Daniel F 71500.00
10 English Jones Sally F 74000.00
11 Business Vilma Jonathan P 35500.00
12 Music Thomas Derrick P 35500.00
13 Education Black Bill P 34000.00
14 Mathematics Warren Angela P 33000.00
15 English Drew Daniel F 72000.00
16 Science Gallegos Tomas F 64000.00
Query executed successfully.
DESKTOP-DU81IMG\SQLEXPRESS ...
DESKTOP-DU81IMG\abhis ...
MyCollege
00:00:00
16 rows
Ready
Ln 1
Col 36
Ch 36
INS

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

Query :

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

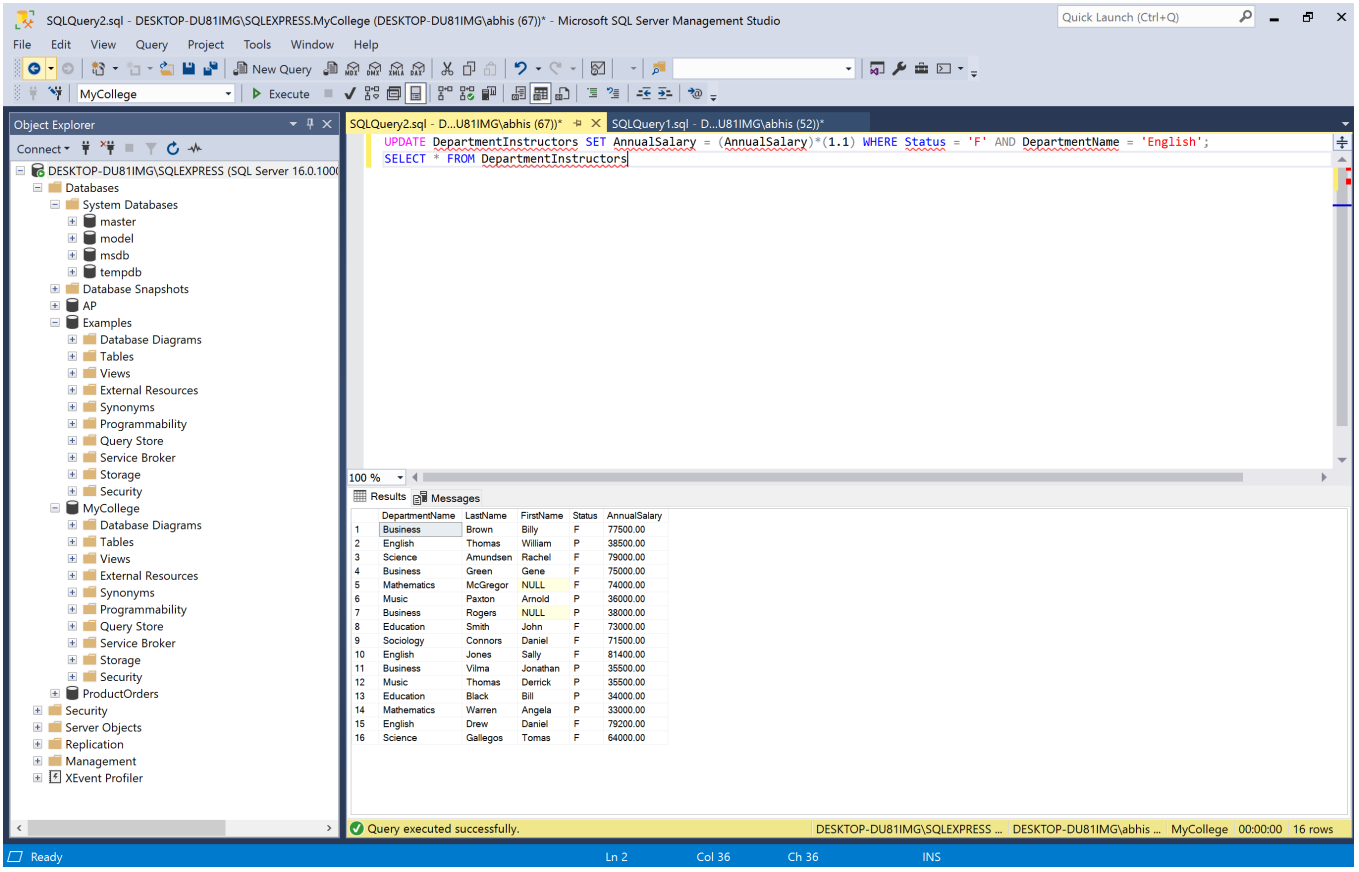


Comments: Successfully executed a SELECT statement that return one row for each full- time instructor in English department, total 9 rows of data.

4) Write an UPDATE statement that updates the DepartmentInstructors view you created in exercise 1 so it increases the annual salary for each fulltime instructor inthe English department by 10%. Then, run the SELECT statement you wrote in exercise 2 to be sure this worked correctly.

Query :

```
UPDATE DepartmentInstructors SET AnnualSalary =(AnnualSalary)*(1.1) WHERE
Status ='F'AND DepartmentName ='English';
SELECT*FROM DepartmentInstructors
```

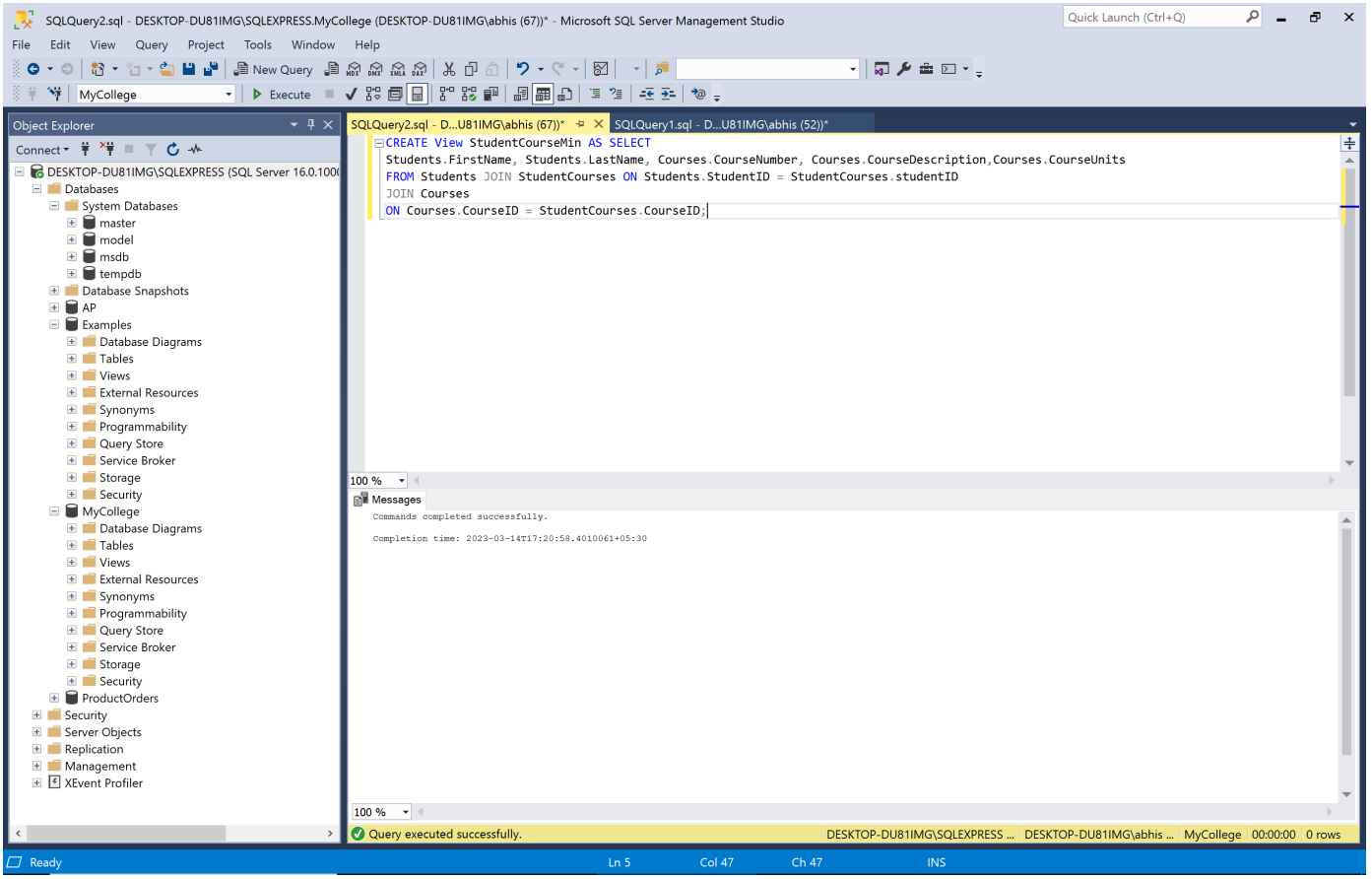


Comments: Successfully updated the DepartmentInstructions view so as to increase the annual salary for each full time instructor in the English department by 10%.

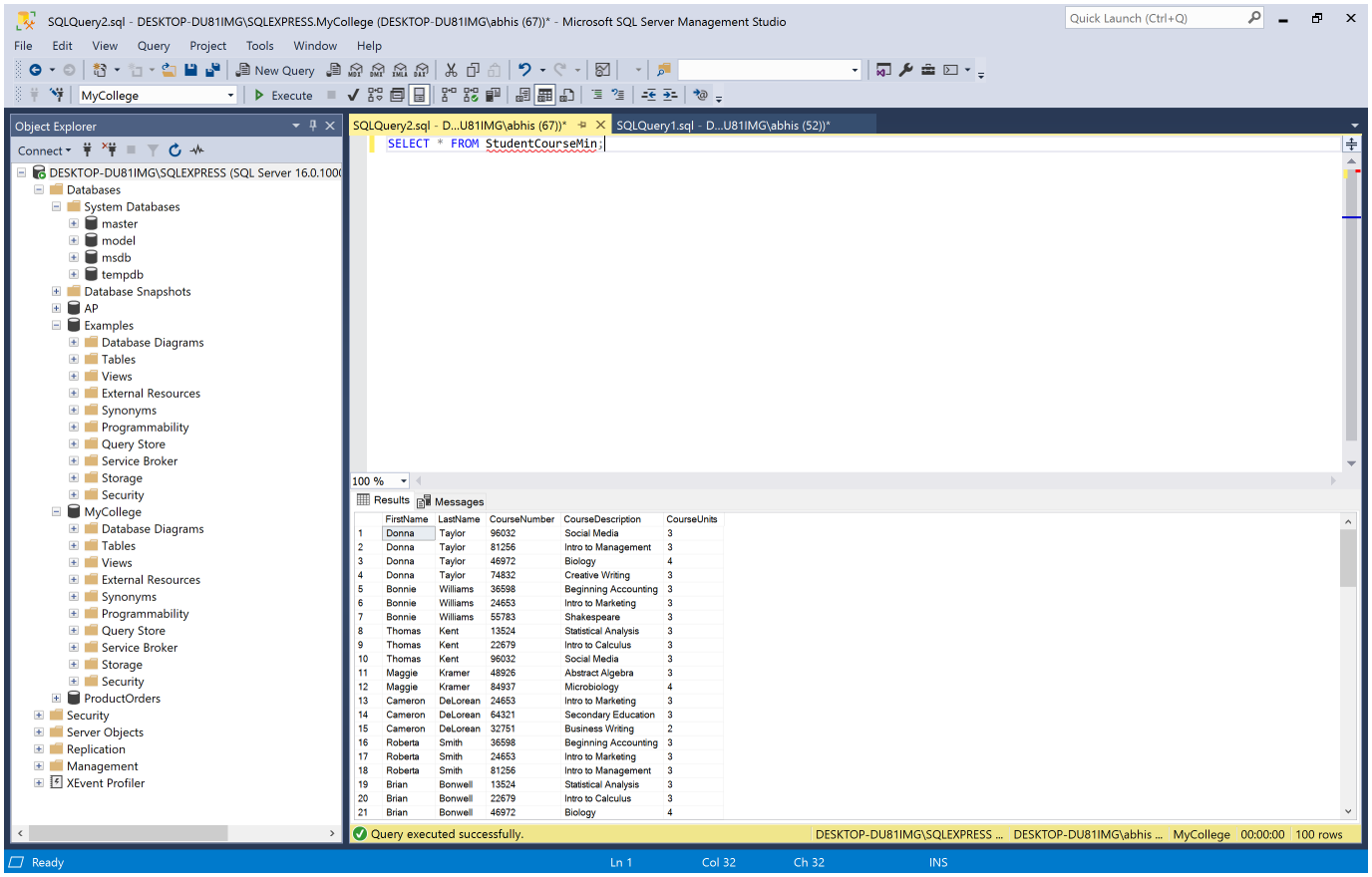
5) Create a view named StudentCoursesMin that returns these columns: the FirstName and LastName from the Students table and the CourseNumber,CourseDescription, and CourseUnits from the Courses

Query :

```
CREATE View StudentCoursesMin AS SELECT
Students.FirstName,Students.LastName,Courses.CourseNumber,Courses.CourseDescri
ption,Courses.CourseUnits
FROM Students JOIN StudentCourses ON Students.StudentID =
StudentCourses.StudentIDJOIN Courses
ON Courses.CourseID =StudentCourses.CourseID
```



Comments: Successfully created a view named StudentCoursesMin with the required columns, using CREATE VIEW statement.

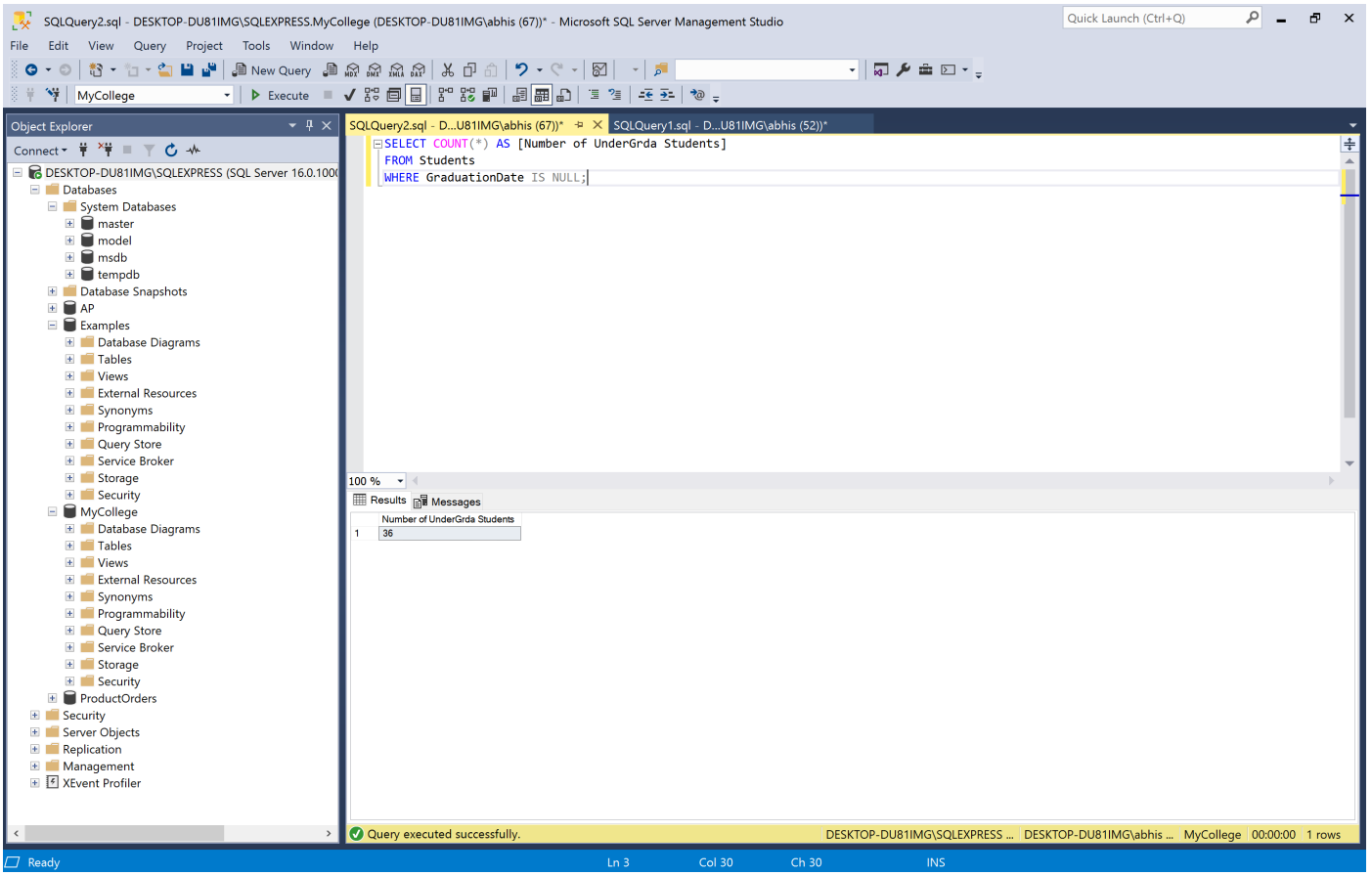


Comments: Verifying the StudentCoursesMin and 100 rows affected.

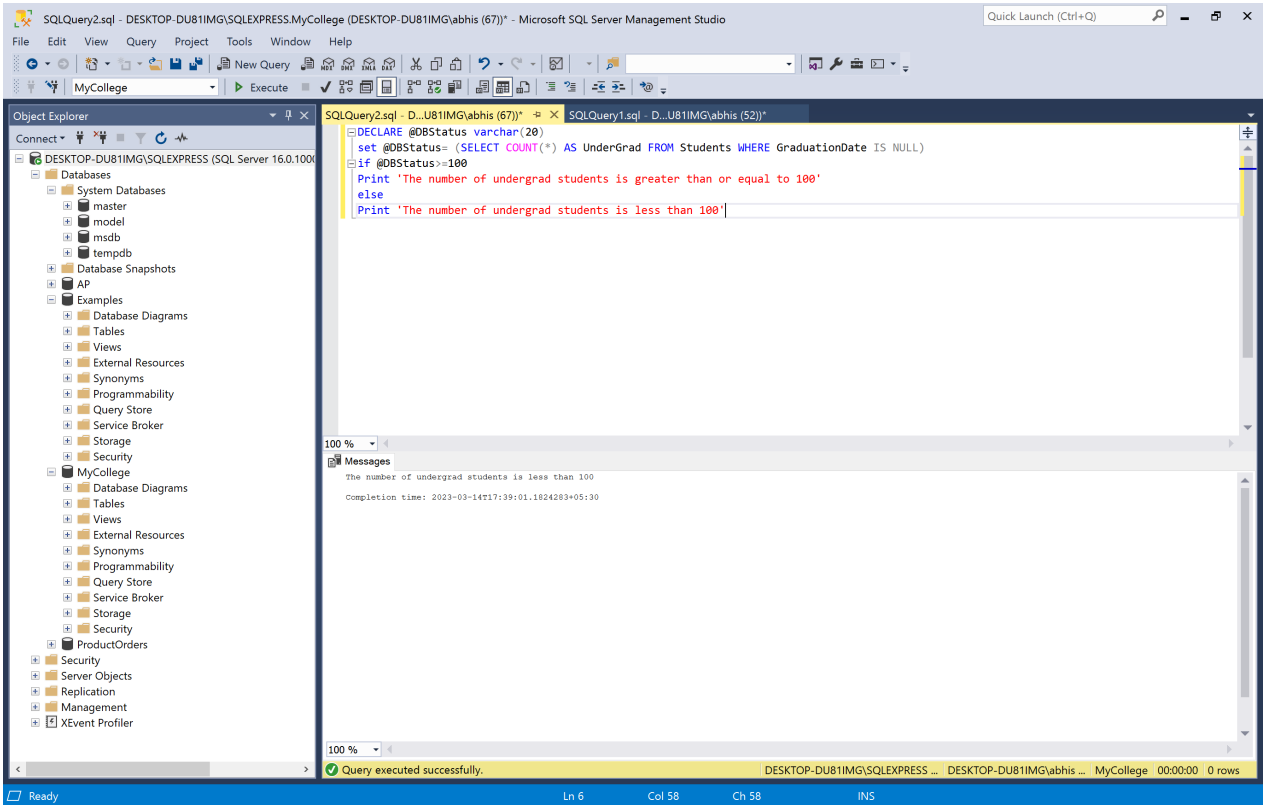
6) Write a script that declares a variable and sets it to the count of all students in the Students table that haven't graduated. If the count is greater than or equal to 100, the script should display a message that says, "The number of undergrad studentsis greater than or equal to 100". Otherwise, it should say, "The number of undergrad students is less than 100".

Query :

```
DECLARE @DBStatus varchar(20)
set @DBStatus=(SELECT COUNT(*) AS UnderGrad FROM Students WHERE
GraduationDate ISNULL)
if @DBStatus>=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'
```



Comments: Number of undergrad students are 36

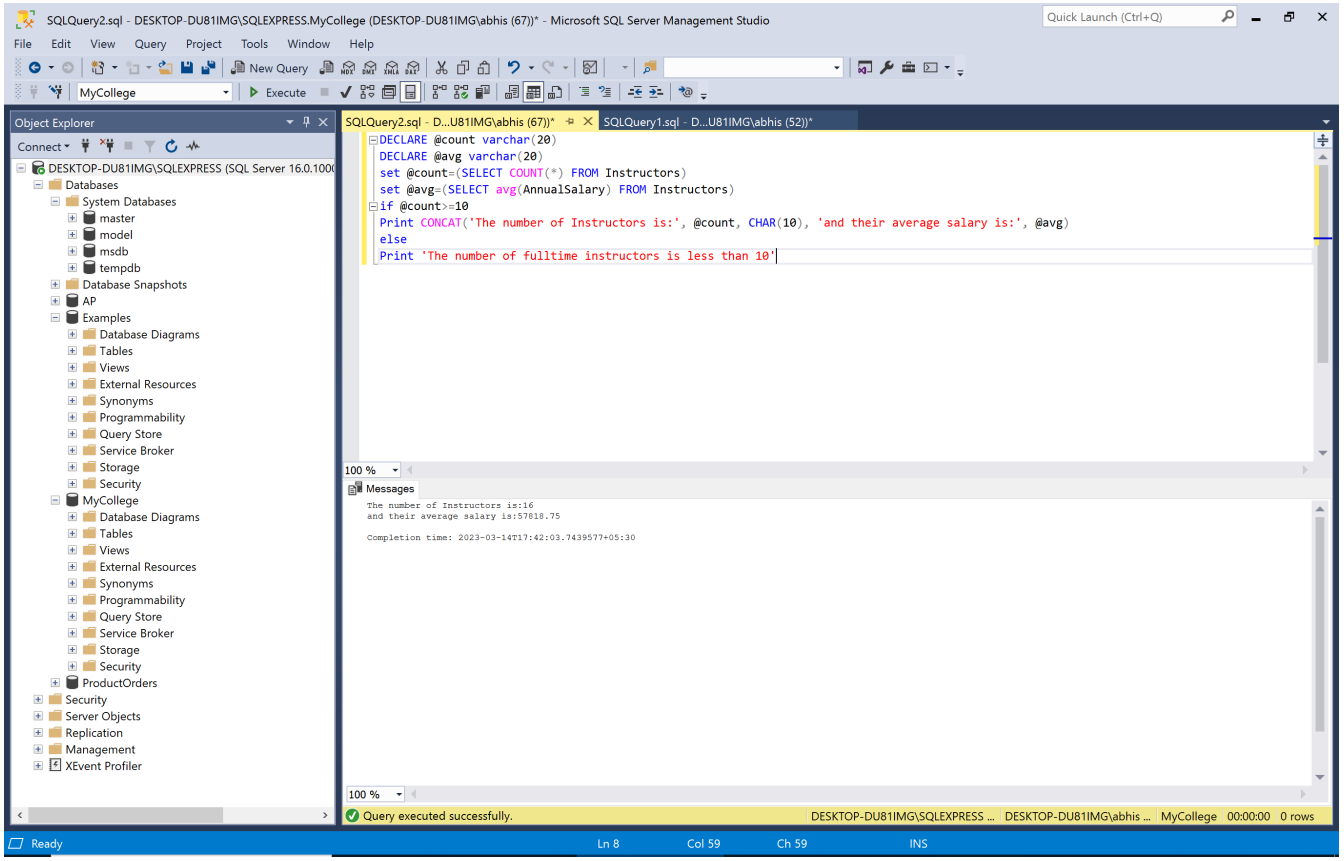


Comments : The number of undergrad students is less than 100

7) Write a script that uses two variables to store (1) the count of all of the instructors in the Instructors table and (2) the average annual salary for those instructors. If the instructor count is greater than or equal to 10, the script should print a message that displays the values of both variables. Otherwise, the script should print a message that says, “The number of fulltime instructors is less than 10”.

Query :

```
DECLARE @count
varchar(20)DECLARE @avg
varchar(20)
set @count=(SELECT COUNT(*) FROM Instructors)
set @avg=(SELECT avg(AnnualSalary) FROM
Instructors)if @count>=10
Print CONCAT('The number of Instructors is:', @count, CHAR(10), 'and their
averagesalary is:', @avg)
else
Print 'The number of fulltime instructors is less than 10'
```

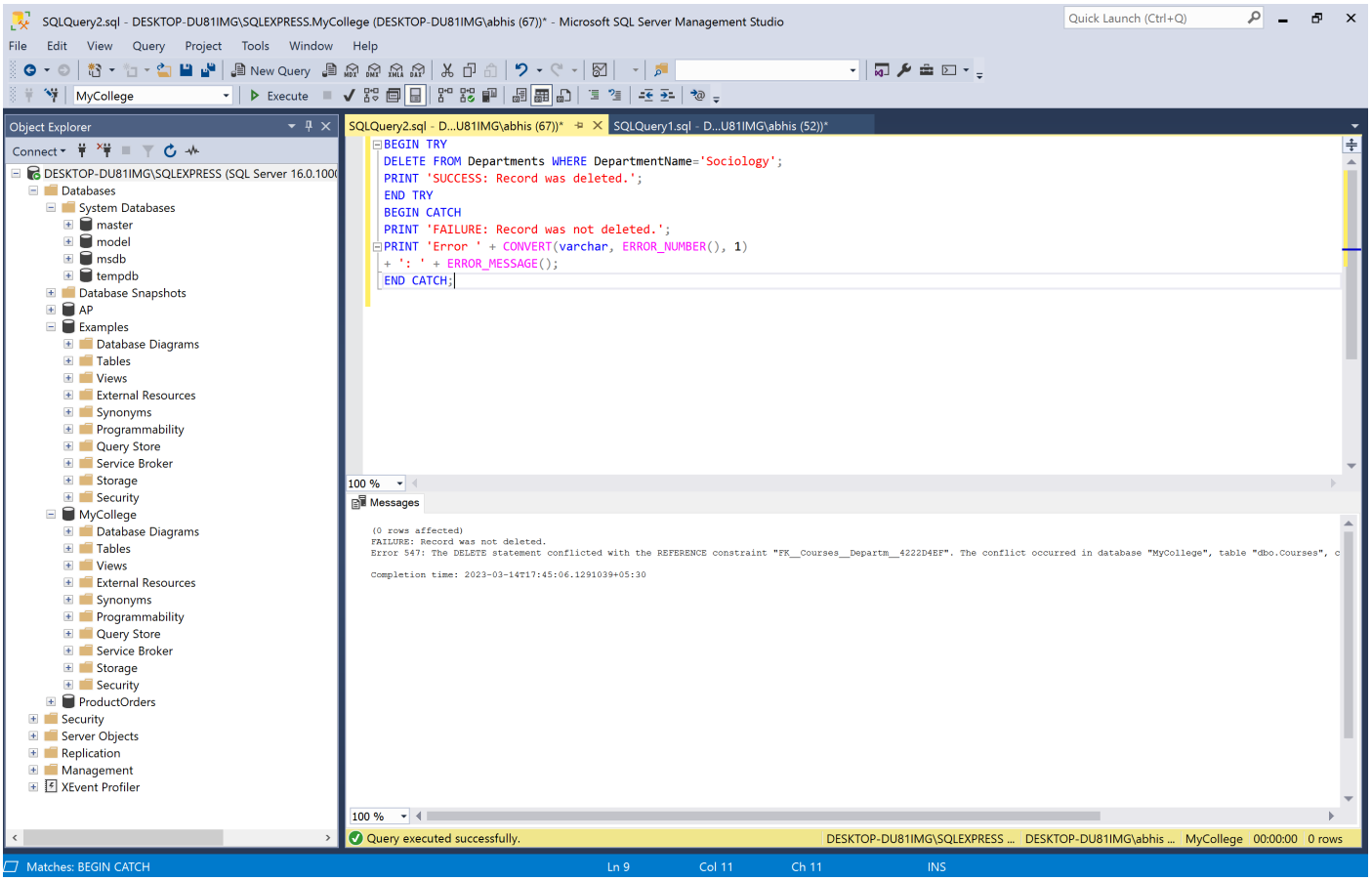


Comments : The number of Instructors is 16 and their average salary is 57818.75

8) Write a script that attempts to delete the department with the name ‘Sociology’ from the Departments table. If the delete is successful, the script should display this message:
SUCCESS: Record was deleted.
If the delete is unsuccessful, the script should display a message something likethis:
FAILURE: Record was not deleted.
Error 547: The DELETE statement conflicted with the REFERENCE constraint"FK__Instructo__Depar__267ABA7A". The conflict occurred in database "MurachCollege", table "dbo.Instructors", column 'DepartmentID'.

Query :

```
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;
```

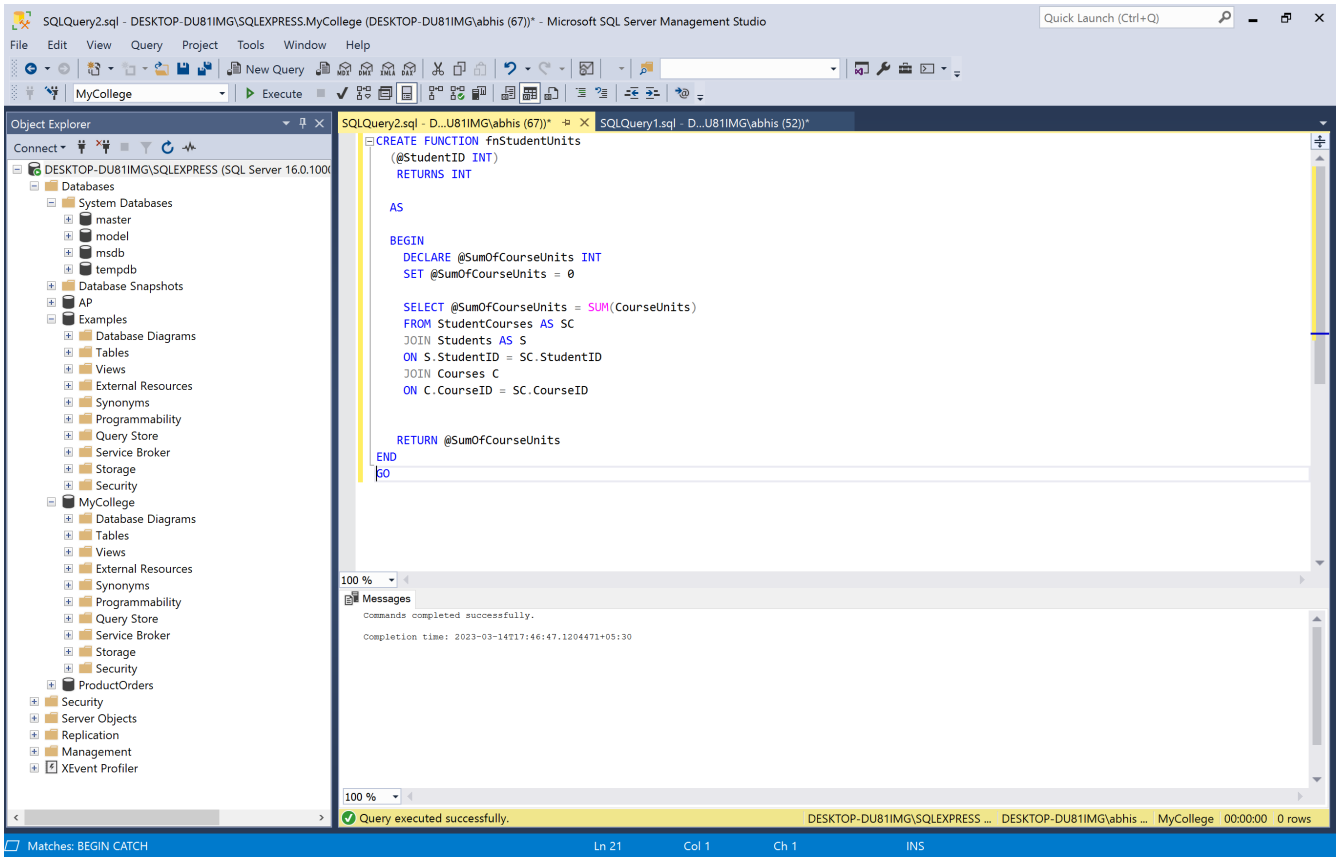


Comments: Sociology department name was successfully deleted with the help of aforementioned query.

9) Write a script that creates and calls a function named fnStudentUnits that calculates the total course units of a student in the StudentCourses table. To dothat, this function should accept one parameter for the student ID, and it shouldreturn the sum of the course units for the student.

Query :

```
CREATE FUNCTION
fnStudentUnits(@StudentID
INT)
RETURNS
INTAS

BEGIN
DECLARE @SumOfCourseUnits
INTSET @SumOfCourseUnits =
0

SELECT @SumOfCourseUnits =
SUM(CourseUnits)FROM StudentCourses AS
SC
JOIN Students AS S
ON S.StudentID =
SC.StudentIDJOIN Courses C
ON C.CourseID = SC.CourseID

RETURN
@SumOfCourseUnitsEND
GO
```

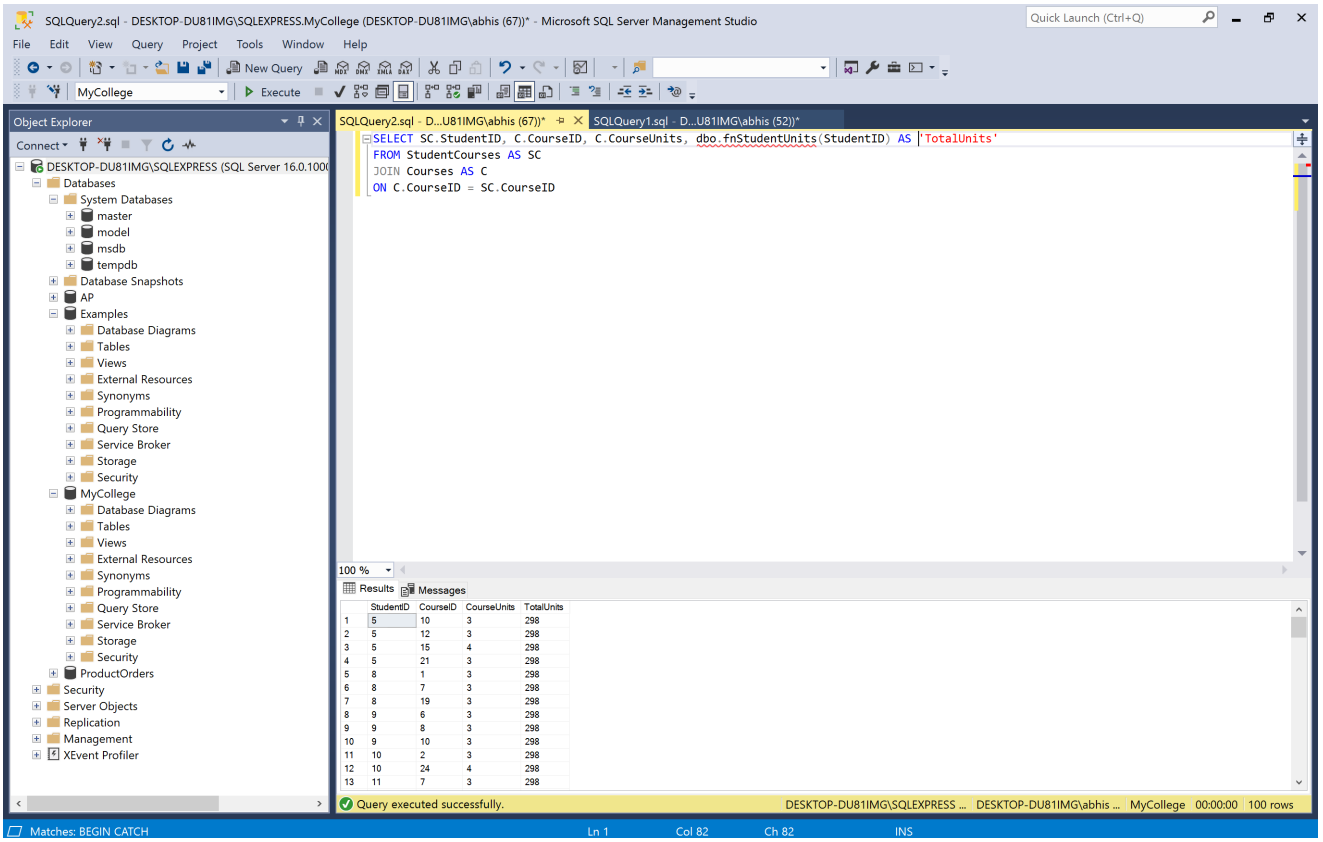


Comments : Successfully wrote the query calls a function named fnStudentUnits that calculates the total course units of a student in the StudentCourses table. To do that, this function should accept one parameter for the student ID, and it should return the sum of the course units for the student.

10)The SELECT statement that calls the function should return the StudentID from theStudentCourses table, the CourseNumber and CourseUnits from the Courses table, and the value return by the fnStudentUnits function for that student.

Query:

```
SELECT SC.StudentID, C.CourseID, C.CourseUnits,
dbo.fnStudentUnits(StudentID) AS'TotalUnits'
FROM StudentCourses AS
SCJOIN Courses AS C
ON C.CourseID = SC.CourseID
```

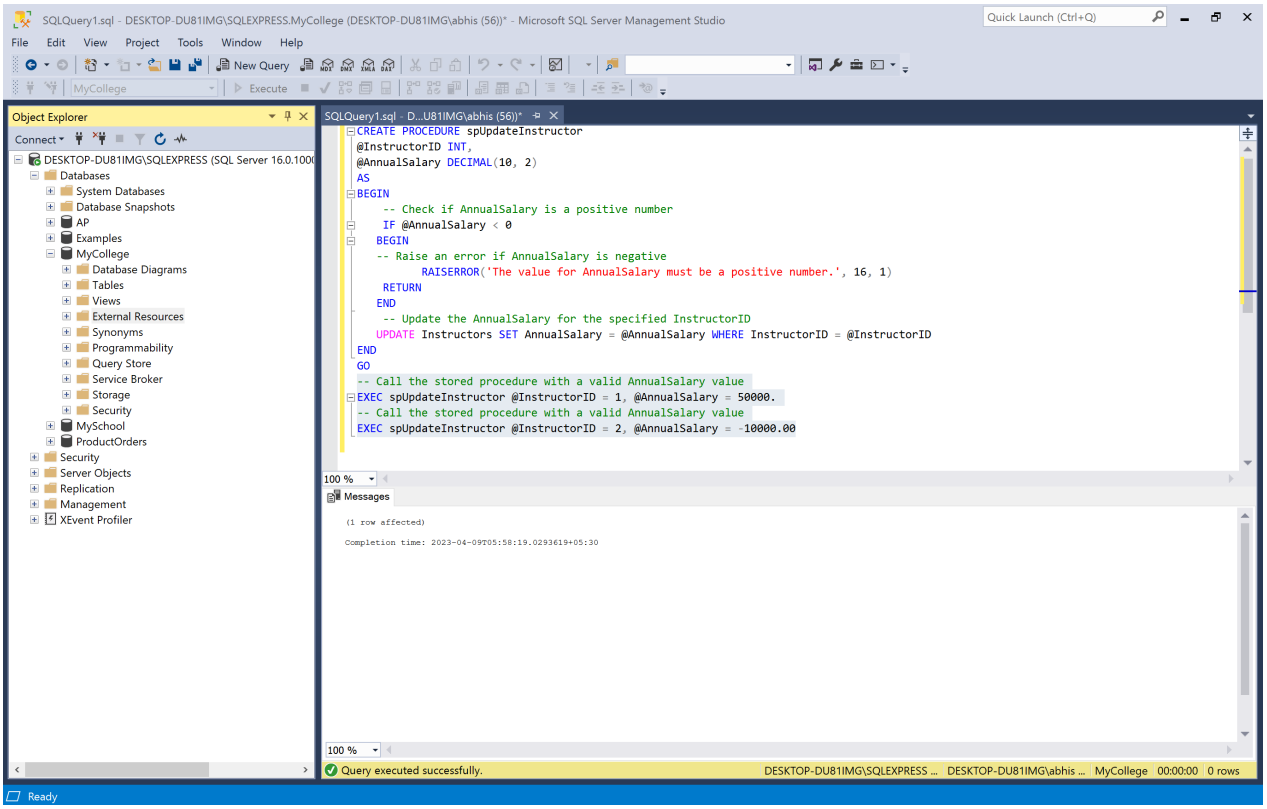


Comments: Successfully wrote the query that calls the function to return the StudentID from the StudentCourses table, the CourseNumber and CourseUnits from the Courses table, and the value return by the fnStudentUnits function for that student.

11. Write a script that creates and calls a stored procedure named spUpdateInstructor that updates the AnnualSalary column in the Instructors table. This procedure should have one parameter for the instructor ID and another for the annual salary. If the value for the AnnualSalary column is a negative number, the stored procedure should raise an error that indicates that the value for this column must be a positive number. Code at least two EXEC statements that test this procedure.

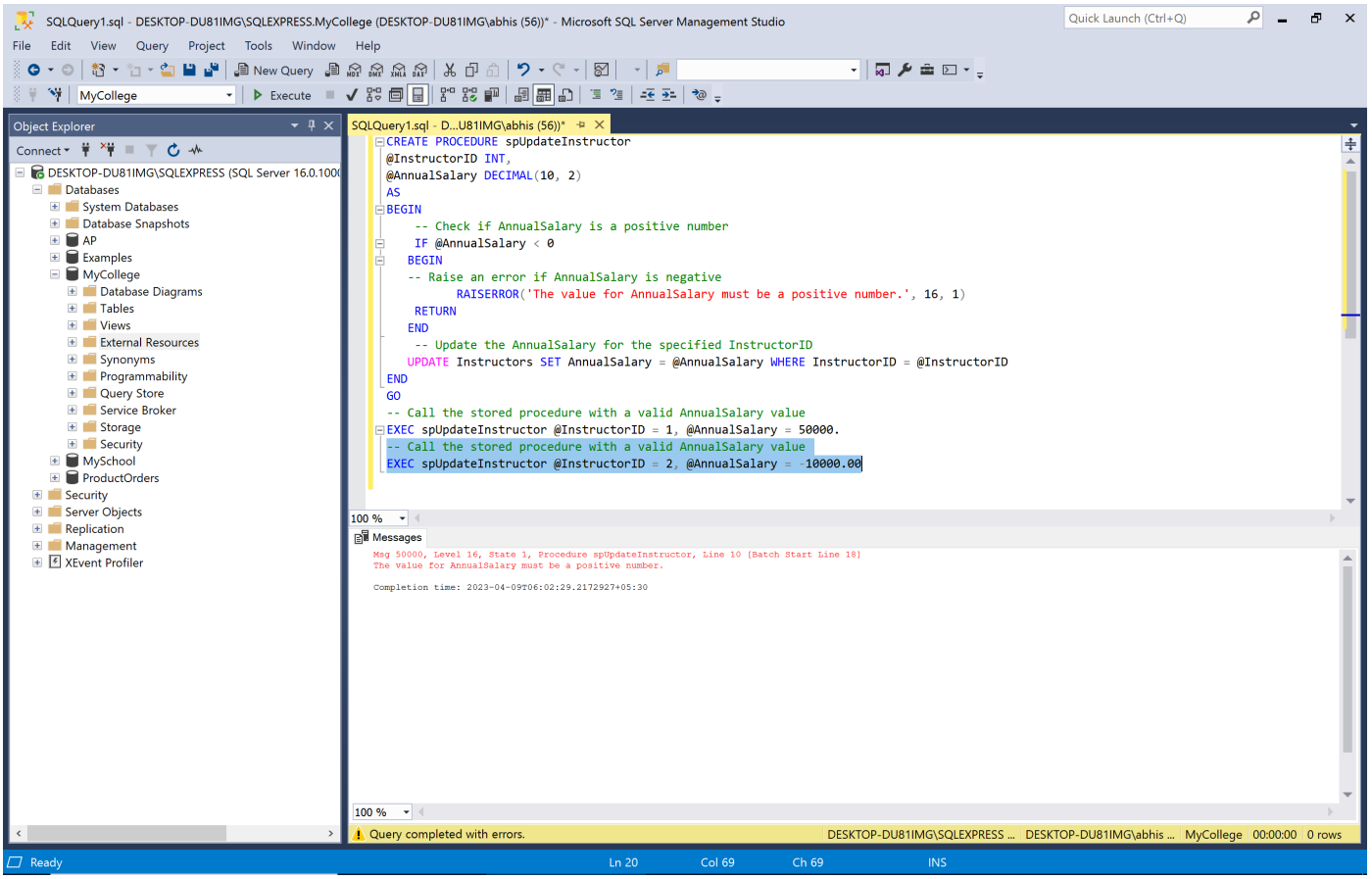
Query:

```
CREATE PROCEDURE spUpdateInstructor
    @InstructorID INT,
    @AnnualSalary DECIMAL(10, 2)
AS
BEGIN
    -- Check if AnnualSalary is a positive number
    IF @AnnualSalary < 0
    BEGIN
        -- Raise an error if AnnualSalary is negative
        RAISERROR('The value for AnnualSalary must be a positive number.', 16, 1)
        RETURN
    END

    -- Update the AnnualSalary for the specified InstructorID
    UPDATE Instructors SET AnnualSalary = @AnnualSalary WHERE InstructorID = @InstructorID
END
GO

-- Call the stored procedure with a valid AnnualSalary value
EXEC spUpdateInstructor @InstructorID = 1, @AnnualSalary = 50000.00
```



Comments : For the first exec call it’s valid and update the record

```
-- Call the stored procedure with an invalid AnnualSalary value
EXEC spUpdateInstructor @InstructorID = 2, @AnnualSalary = -10000.00
```

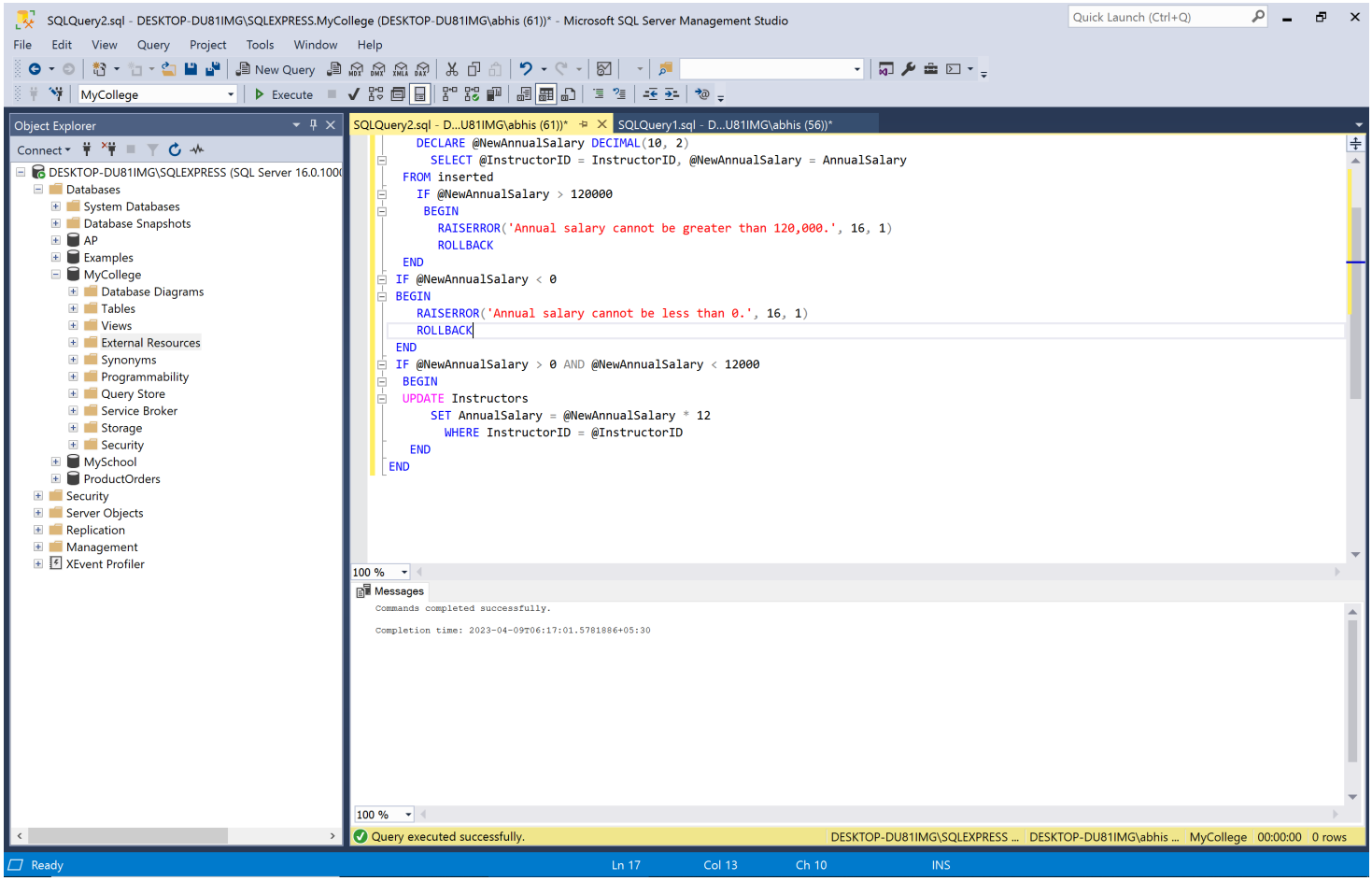


Comments : Throws error as salary is less than 0

12. Create a trigger named Instructors_UPDATE that checks the new value for the AnnualSalary column of the Instructors table. This trigger should raise an appropriate error if the annual salary is greater than 120,000 or less than 0. If the new annual salary is between 0 and 12,000, this trigger should modify the new annual salary by multiplying it by 12. That way, a monthly salary of 5,000 becomes an annual salary of 60,000. Test this trigger with an appropriate UPDATE statement.

Query:

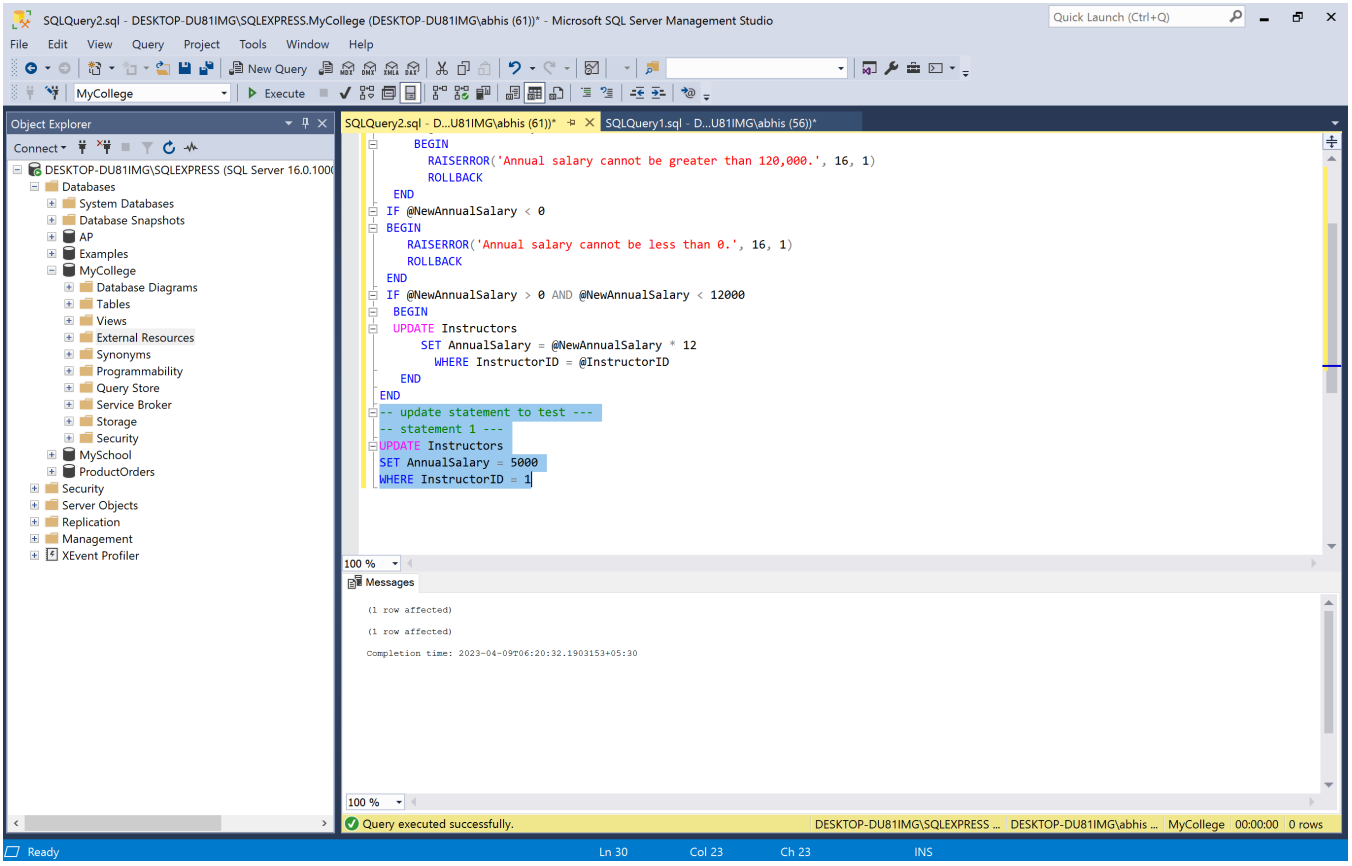
```
CREATE TRIGGER Instructors_UPDATE ON Instructors
AFTER UPDATE
AS
BEGIN
   DECLARE @InstructorID INT
   DECLARE @NewAnnualSalary DECIMAL(10, 2)

   SELECT @InstructorID = InstructorID, @NewAnnualSalary = AnnualSalary
   FROM inserted

   IF @NewAnnualSalary > 120000
   BEGIN
      RAISERROR('Annual salary cannot be greater than 120,000.', 16, 1)
      ROLLBACK
   END

   IF @NewAnnualSalary < 0
   BEGIN
      RAISERROR('Annual salary cannot be less than 0.', 16, 1)
      ROLLBACK
   END

   IF @NewAnnualSalary > 0 AND @NewAnnualSalary < 12000
   BEGIN
      UPDATE Instructors
      SET AnnualSalary = @NewAnnualSalary * 12
      WHERE InstructorID = @InstructorID
   END
END
```



Comments : Commands completed successfully

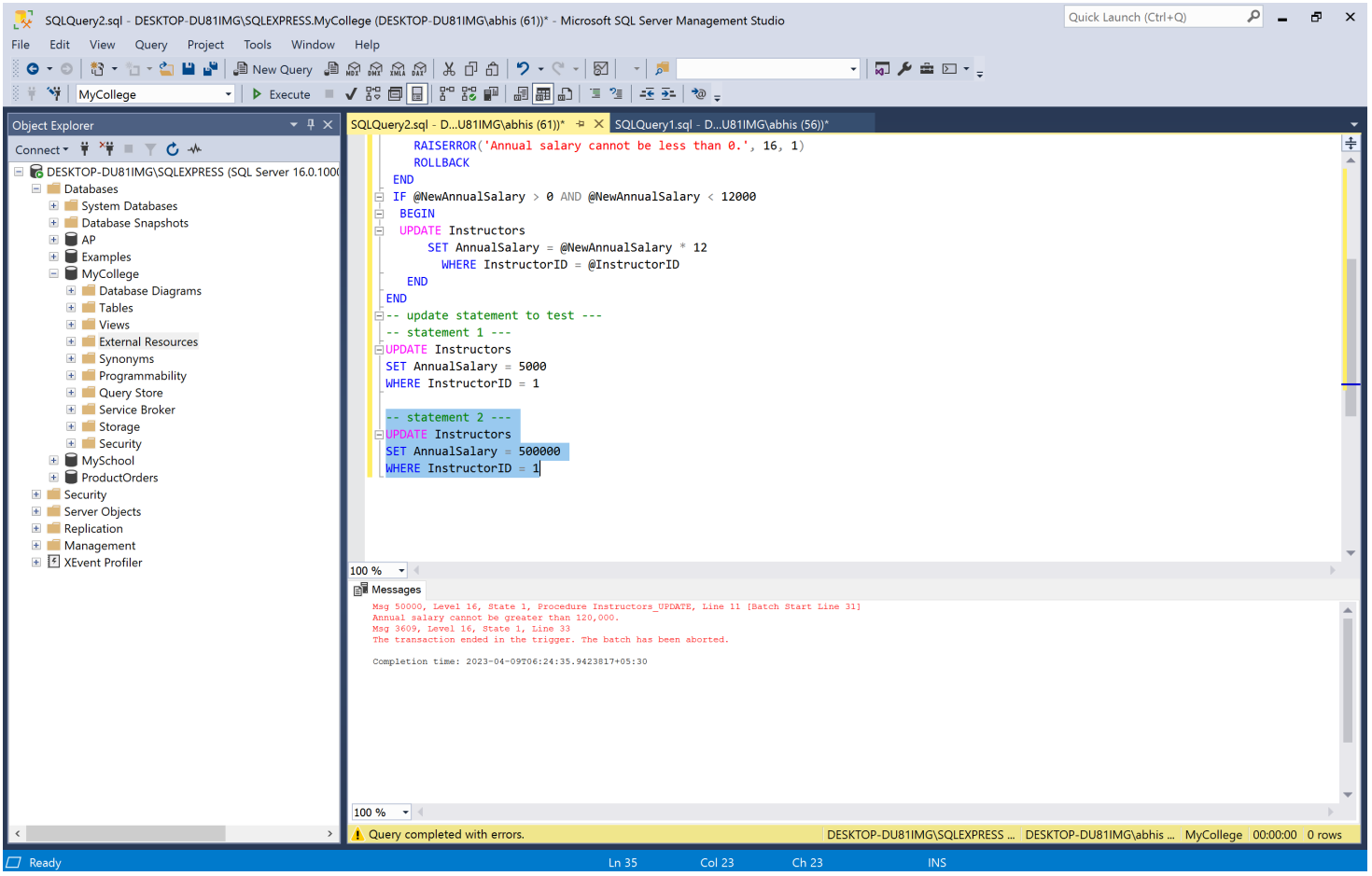
Test the trigger by update statements,

```
-- update statement to test ---
-- statement 1 ---
UPDATE Instructors
SET AnnualSalary = 5000
WHERE InstructorID = 1
```



Comments : This is valid update so it will perform the update operation

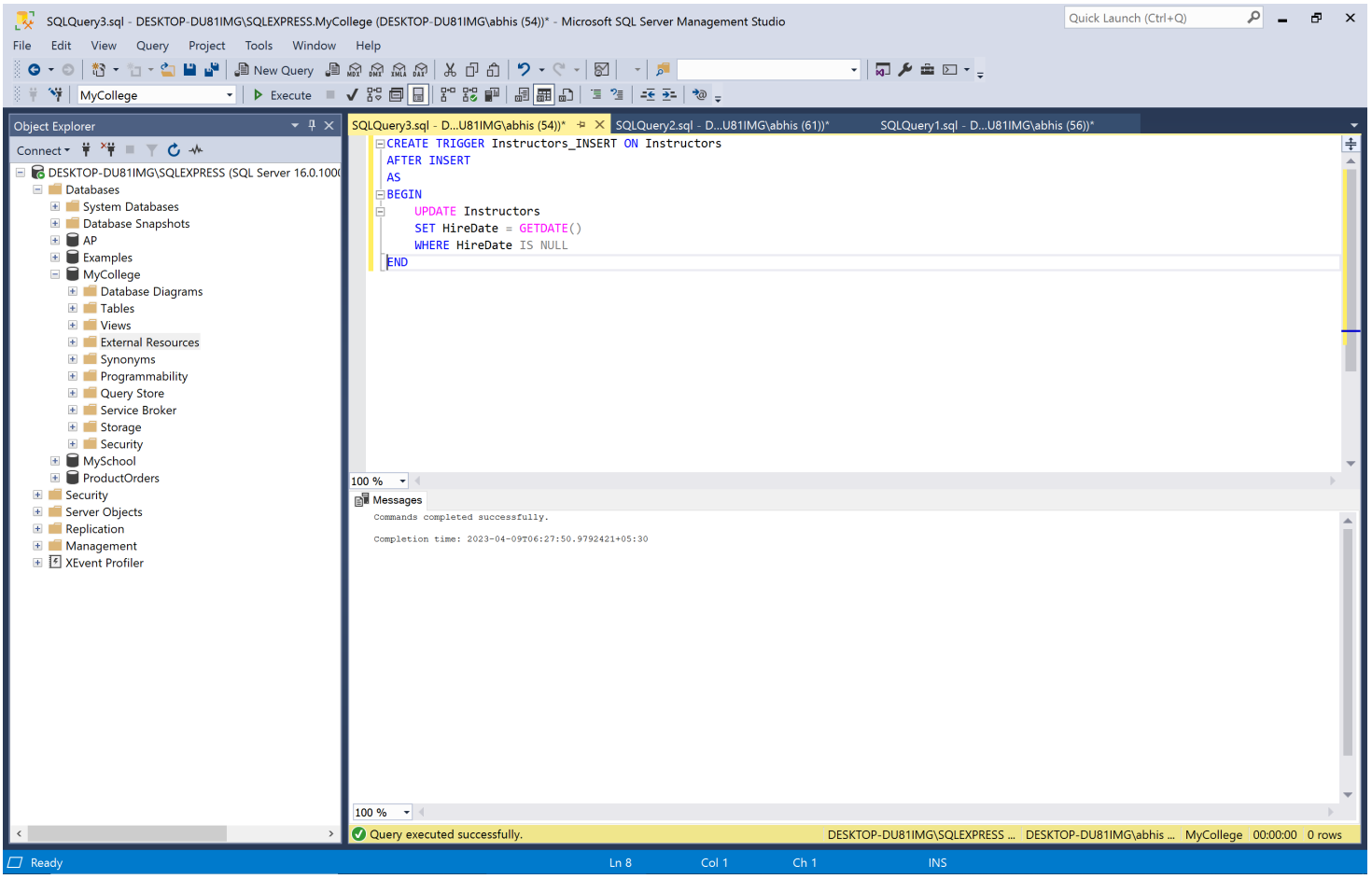
```
-- statement 2 ---

UPDATE Instructors
SET AnnualSalary = 500000
WHERE InstructorID = 1
```



Comment : As the Annual Salary is more than 120,000 it throws error and rollbacks the transaction

13.Create a trigger named Instructors_INSERT that inserts the current date for the HireDate column of the Instructors table if the value for that column is null. Test this trigger with an appropriate INSERT statement.
Query:

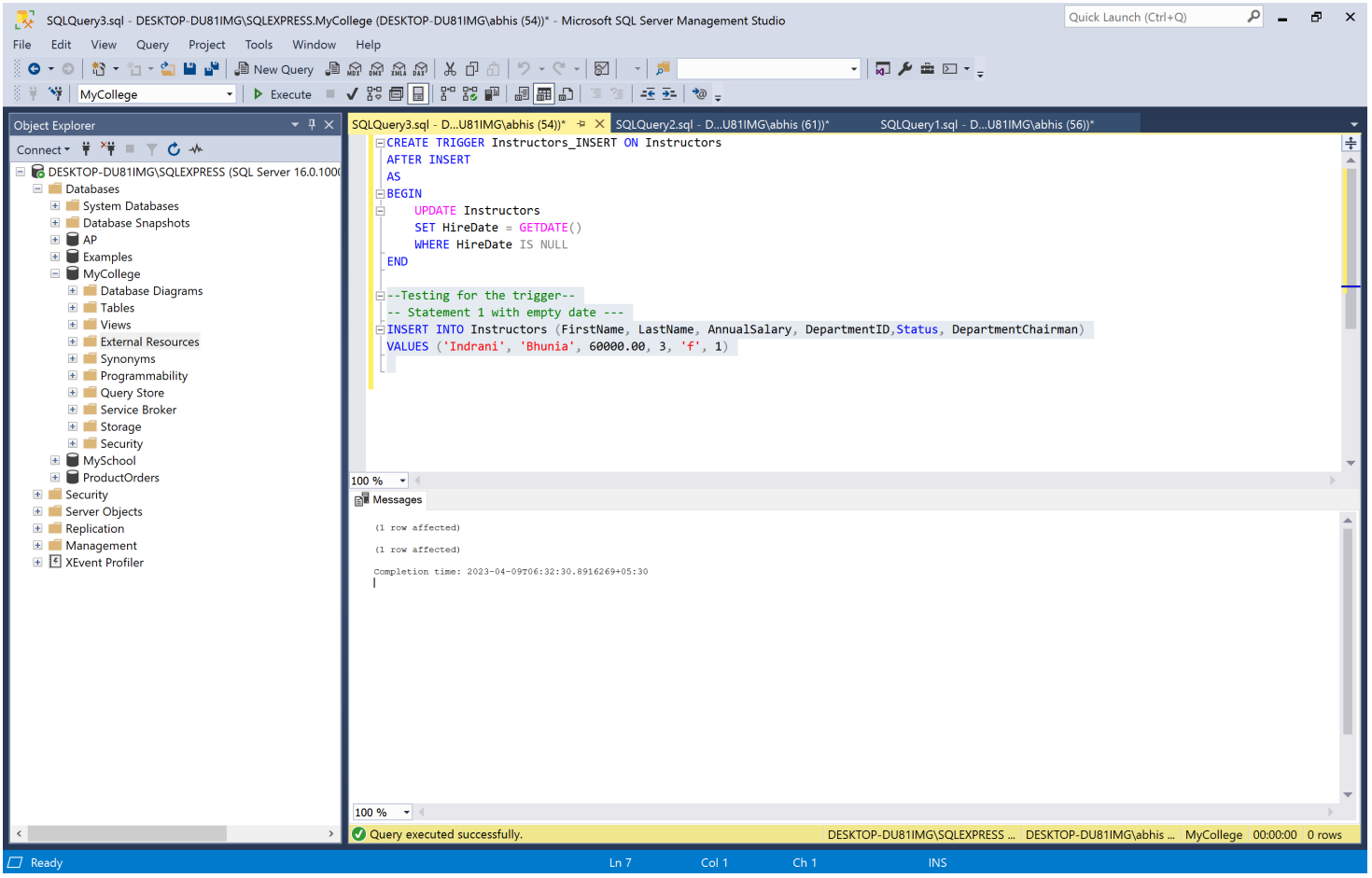
```
CREATE TRIGGER Instructors_INSERT ON Instructors
AFTER INSERT
AS
BEGIN
    UPDATE Instructors
    SET HireDate = GETDATE()
    WHERE HireDate IS NULL
END
```



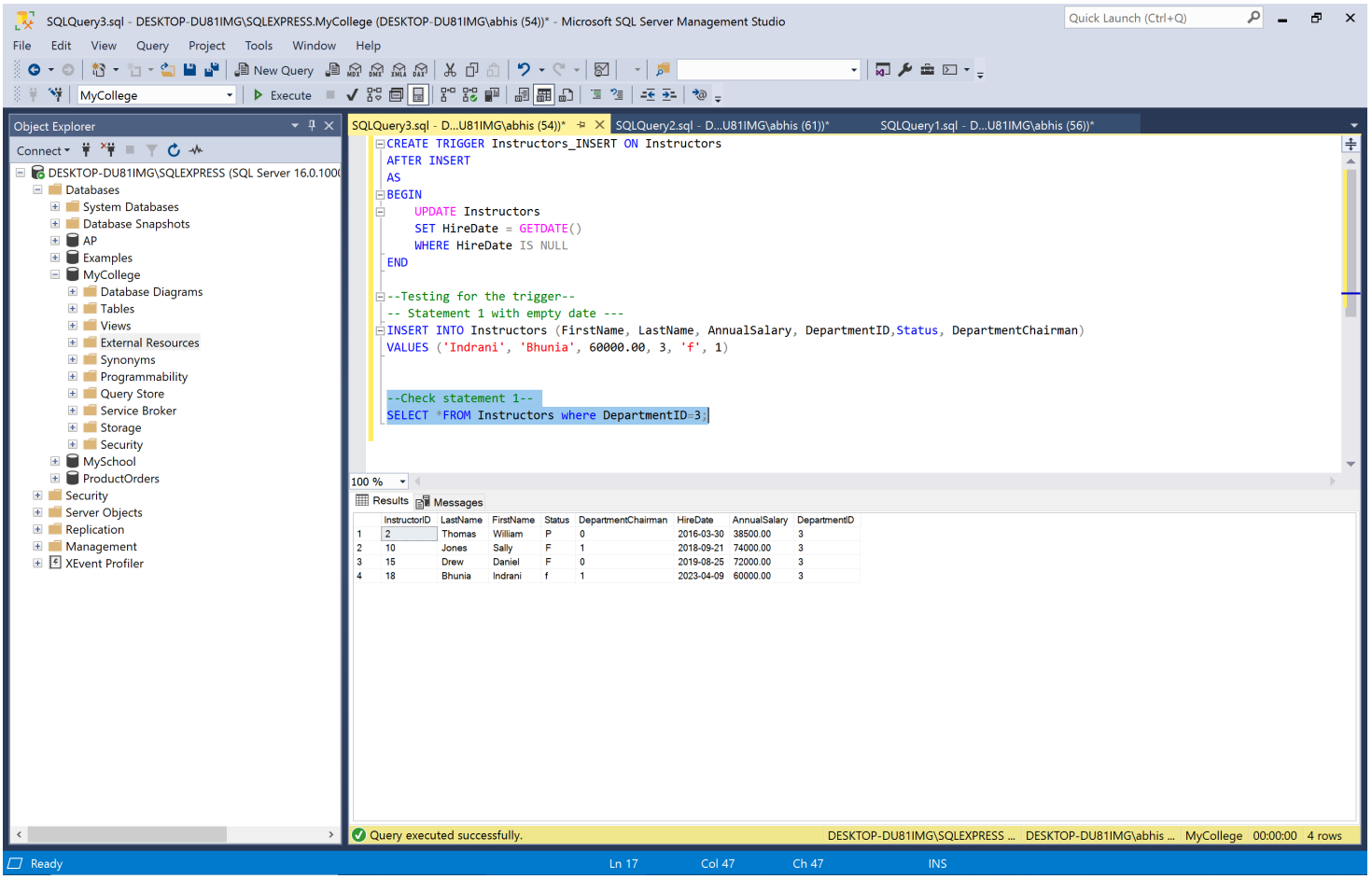
Comments : Trigger created successfully

Test the Trigger:

```
--Testing for the trigger--
-- Statement 1 with empty date ---
INSERT INTO Instructors (FirstName, LastName, AnnualSalary, DepartmentID,Status,
DepartmentChairman)
VALUES ('Indrani', 'Bhunia', 60000.00, 3, 'f', 1)
```

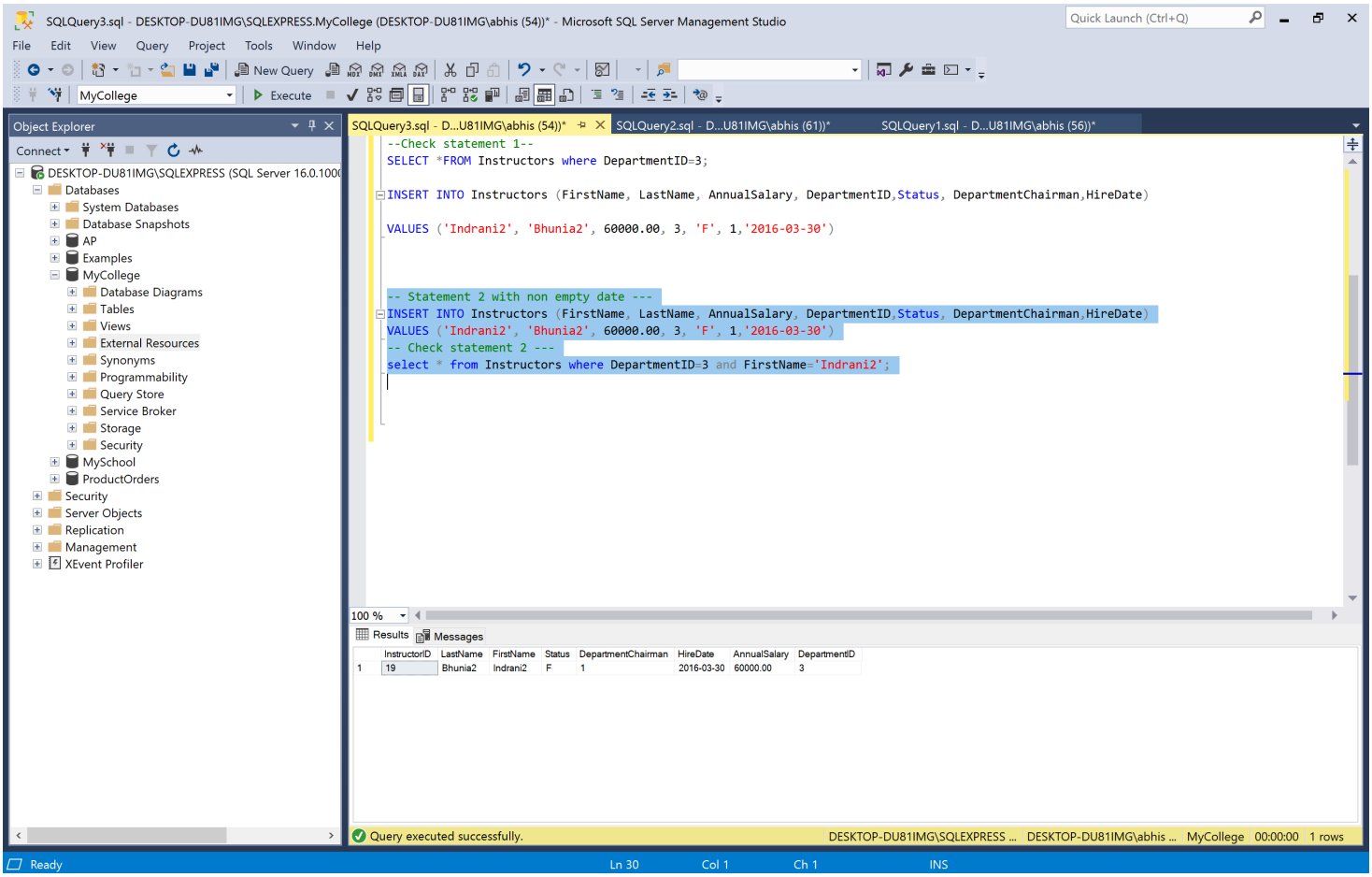


Comments : Inserting Row



Comments :The select query checks for Indrani Bhunia whose hiredate is current date as it wasn't set during insertion

```
-- Statement 2 with non empty date ---
INSERT INTO Instructors (FirstName, LastName, AnnualSalary, DepartmentID,Status,
DepartmentChairman,HireDate)
VALUES ('Indrani2', 'Bhunia2', 60000.00, 3, 'F', 1,'2016-03-30')

-- Check statement 2 ---

select * from Instructors where DepartmentID=3 and FirstName='Indrani2';
```

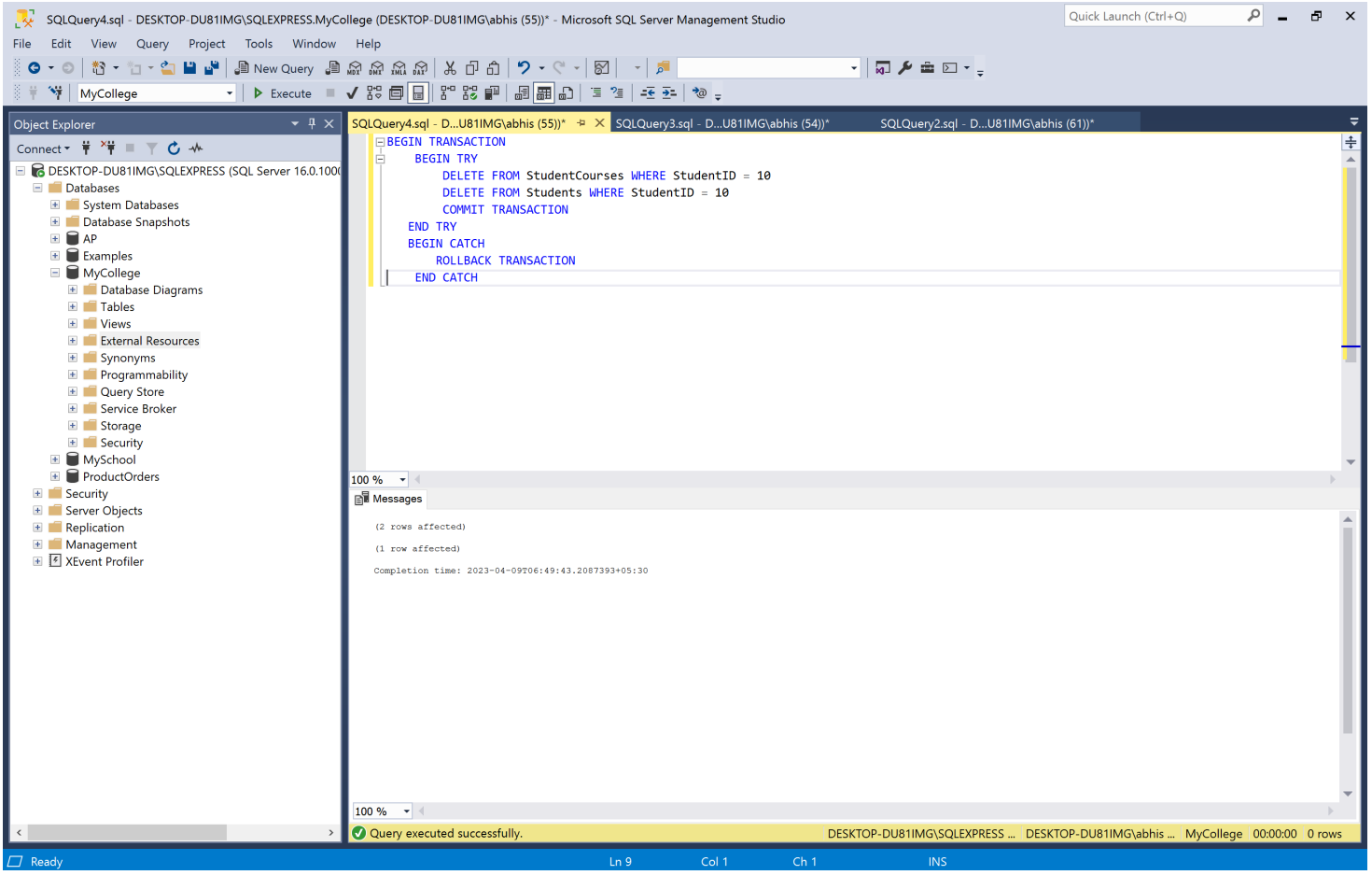


Comments : Testing statement2 where the HireDate is given while insertion

14.Write a script that includes two SQL statements coded as a transaction to delete the row with a student ID of 10 from the Students table. To do this, you must first delete all courses for that student from the StudentCourses table. If these statements execute successfully, commit the changes. Otherwise, roll back the changes.

Query:

```
BEGIN TRANSACTION
   BEGIN TRY
      DELETE FROM StudentCourses WHERE StudentID = 10
      DELETE FROM Students WHERE StudentID = 10
      COMMIT TRANSACTION
   END TRY
   BEGIN CATCH
      ROLLBACK TRANSACTION
   END CATCH
```

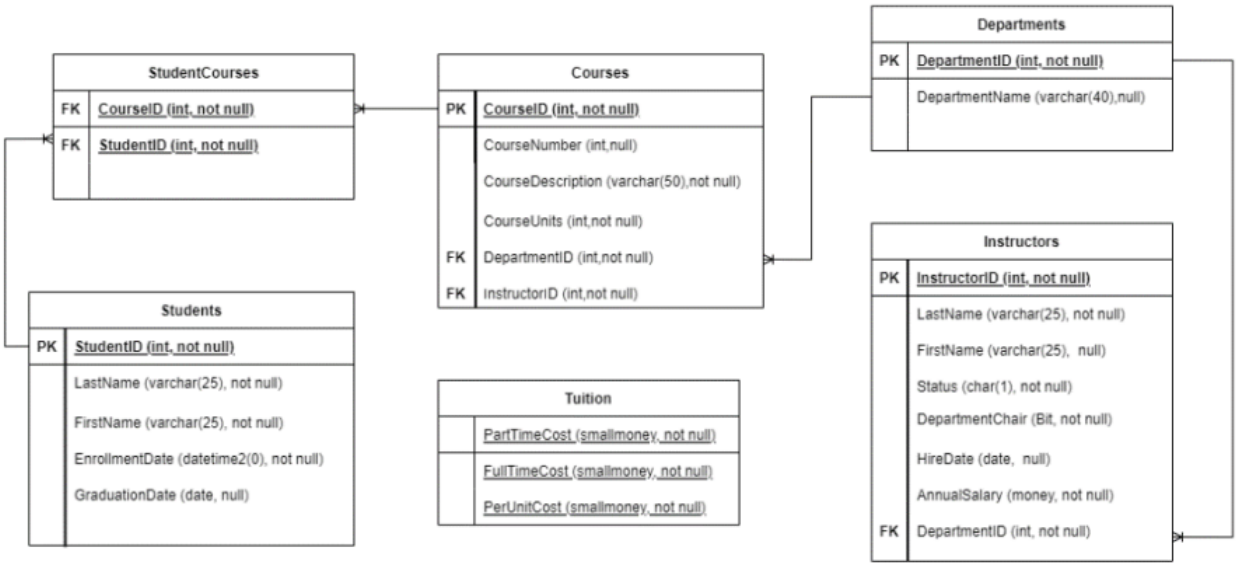


Comments : Executed transaction successfully

Part D :

Database Design

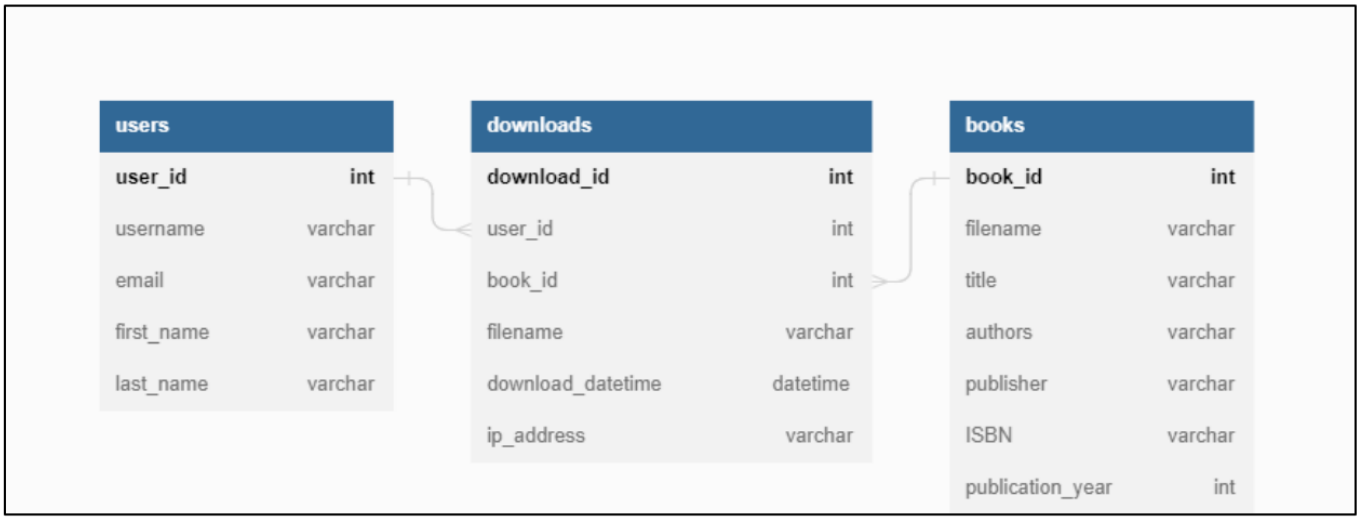
1.Create a database diagram that shows the relationships between the tables in the MyCollege database. You can use any digital tool to draw your diagrams. "Resources and Links" tab on this Blackboard page lists several tools for drawing E/R diagrams.



Comment : ERD for college database

Explanation : The MyCollege Database's schema includes six tables: Departments, Courses, StudentCourses, Instructors, Students, and Tuition. Table StudentCourses has two foreign keys: CourseID and StudentID. Table Courses has a primary key of CourseID and two foreign keys: DepartmentID and InstructorID. Table Instructors has a primary key of InstructorID and a foreign key of DepartmentID. Table Students has a primary key of StudentID. The StudentCourses table acts as a linking table because there is a many-to-many relationship between Students and Courses. The Tuition table is an independent object with no relationships to other tables. The Departments, Instructors, and Courses tables are connected with one-to-many relationships. There is a one-to-many relationship between Departments and Instructors. The above diagram illustrates these connections.

2. Design a database diagram for a database that stores information about the downloads that users make from a book website.
1. Each user must have a user name, email address, first name, and last name.
2. Each user can have zero or more downloads.
3. Each download must have a filename, download date/time, the IP of the downloading site.
4. Each book can be related to zero or more downloads.
5. Each book must have a filename, title, authors, publisher, ISBN, publisher, year.



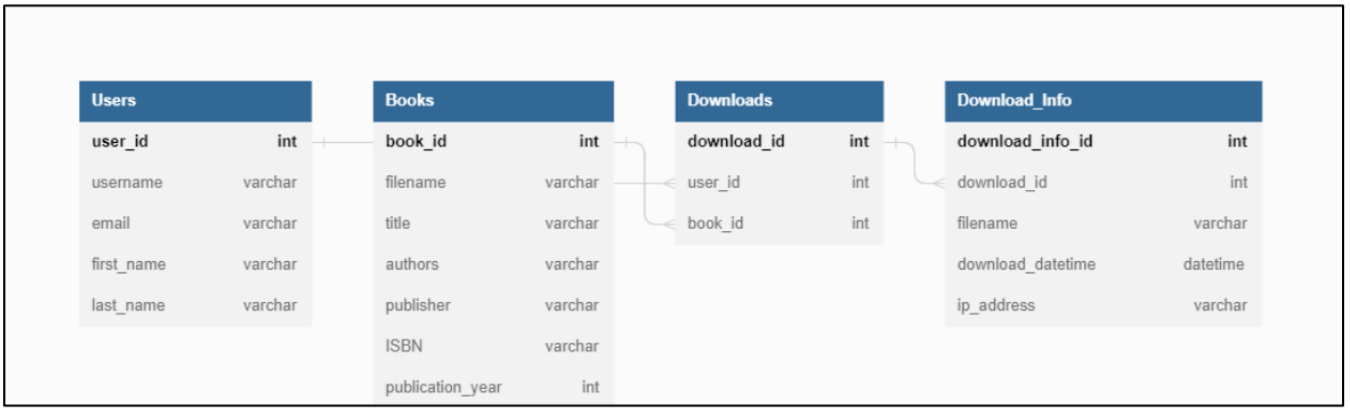
Explanation: This diagram includes three tables: Book, Download, and User. The Book table includes the information about the book, with a unique BookID, filename, title, authors, publisher, ISBN, and year. The User table includes information about the user, with a unique UserID, username, email address, first name, and last name. The Download table is a many-to-many relationship table that connects the Book and User tables. It includes a unique DownloadID, the Book filename that was downloaded (foreign key to Book table), the UserID of the user who downloaded the book (foreign key to User table), the date and time of the download, and the IP address of the site where the download originated. This database diagram can store information about multiple downloads made by different users for different books. It allows for easy tracking of the books that are downloaded, when they were downloaded, and by whom.

3. Analyze your diagram and normalize it to its third-normal form.

The before schema has three tables: Users, Downloads, and Books. The Users table contains information about users such as user_id, username, email, first_name, and last_name. The Downloads table contains download information such as download_id, user_id, book_id, filename, download_datetime, and ip_address. The Books table contains book information such as book_id, filename, title, authors, publisher, ISBN, and publication_year.

This schema is not in 3NF as there are dependencies between non-key attributes and partial dependencies. The Downloads table has a partial dependency on user_id and book_id, meaning that the attributes of the table are not functionally dependent on the entire primary key.

After normalizing the schema, we have four tables: Users, Books, Downloads, and Download_Info. The Users table remains the same as the original schema. The Books table also remains the same. However, we have split the Downloads table into two tables: Downloads and Download_Info. The Downloads table contains only the download_id, user_id, and book_id columns, which form the primary key. The Download_Info table contains the remaining columns from the original Downloads table: filename, download_datetime, and ip_address. This separation eliminates the partial dependency and creates two tables that are both in 3NF.



Explanation:

The normalized database schema includes four tables: Users, Books, Downloads, and Download_Info.

The Users table remains the same as in the original schema and includes columns for user_id, username, email, first_name, and last_name.

The Books table also remains the same as in the original schema and includes columns for book_id, filename, title, authors, publisher, ISBN, and publication_year.

The Downloads table has been split into two tables: Downloads and Download_Info. The Downloads table includes only the columns download_id, user_id, and book_id, which form the primary key of the table. This separation helps to eliminate the partial dependency and creates two tables that are both in 3NF.

The Download_Info table contains the remaining columns from the original Downloads table, including filename, download_datetime, and ip_address. This table is related to the Downloads table by the download_id column.

References

1. Lucidchart: A web-based diagramming tool that offers various templates, including

database diagrams. https://www.lucidchart.com/

2. Draw.io: A free online diagramming tool that allows you to create a variety of diagrams, including ER diagrams. https://app.diagrams.net/

3. SQL tutorials - w3 schools, https://www.w3schools.com/sql

Remarks:

Project 1 provided a comprehensive understanding of DBMS concepts, including Joins, Unions, Aggregate functions, Subqueries, Triggers, and Procedures. It also provided experience in advanced queries and database design models, with practical applications of procedures and views. Through the project, I gained proficiency in executing DDL and DML applications, drawing logical models, and mapping them. Additionally, I learned about SQL data types, constraints, and statement processing, as well as different types of operators and case statements. Overall, the project was a valuable experience for developing a deeper understanding of basic to advanced DBMS concepts.