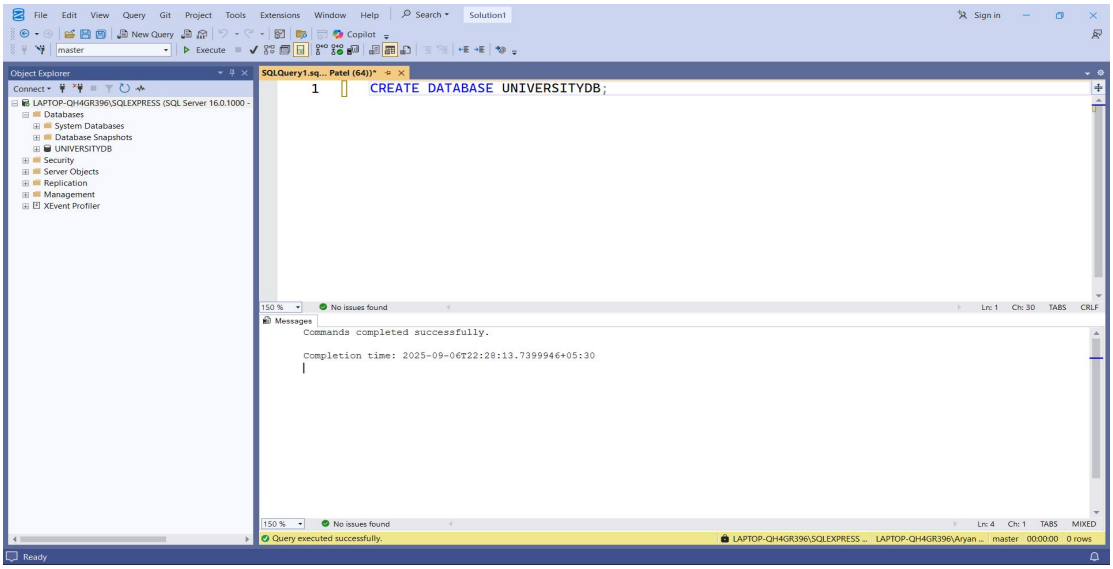
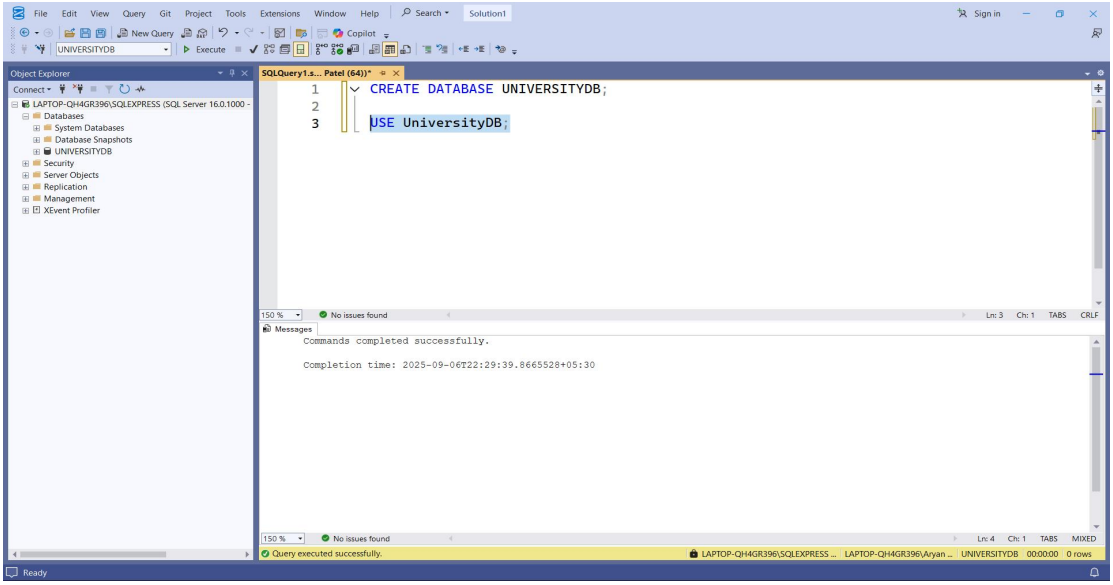
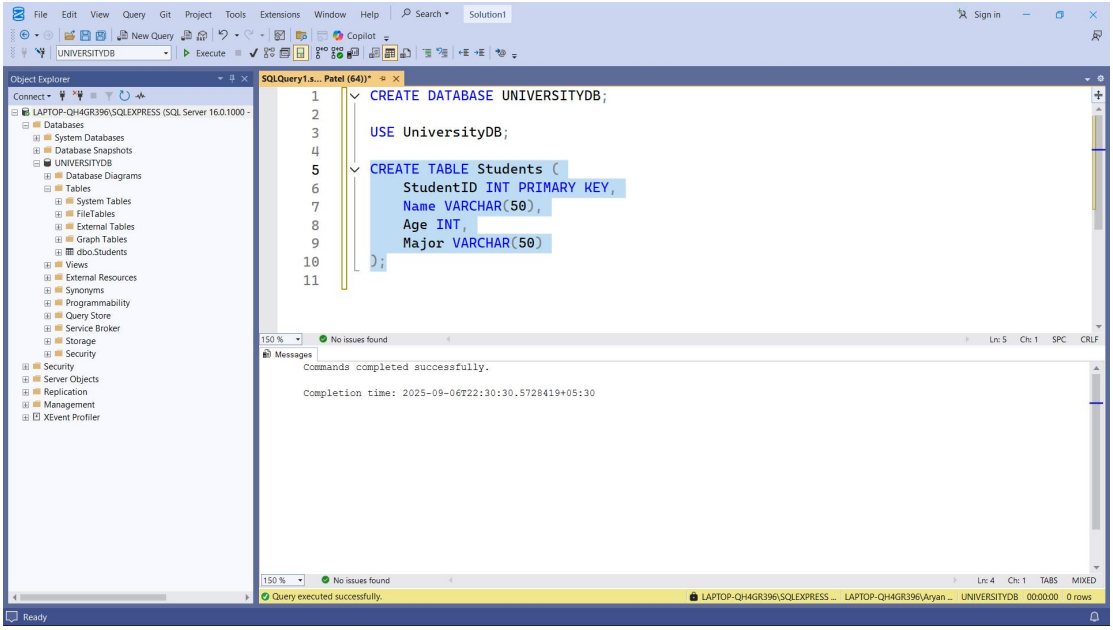
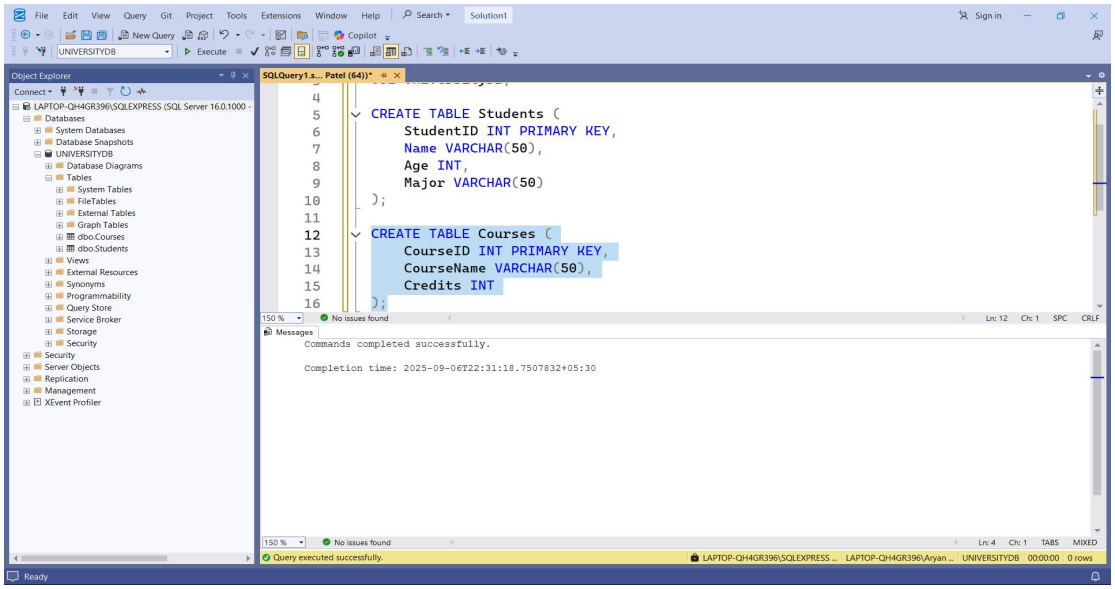
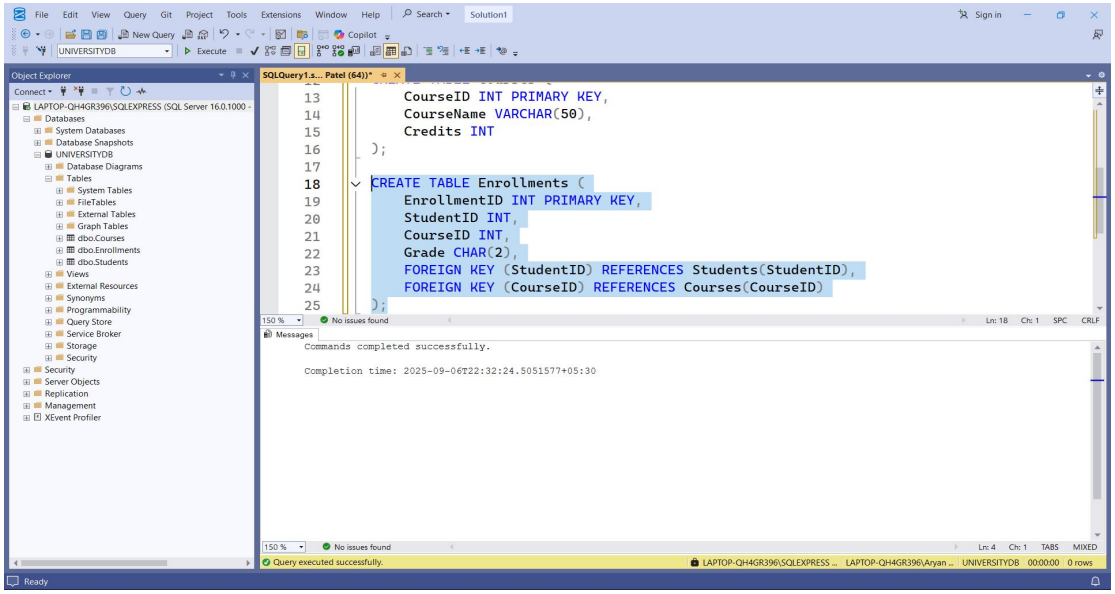
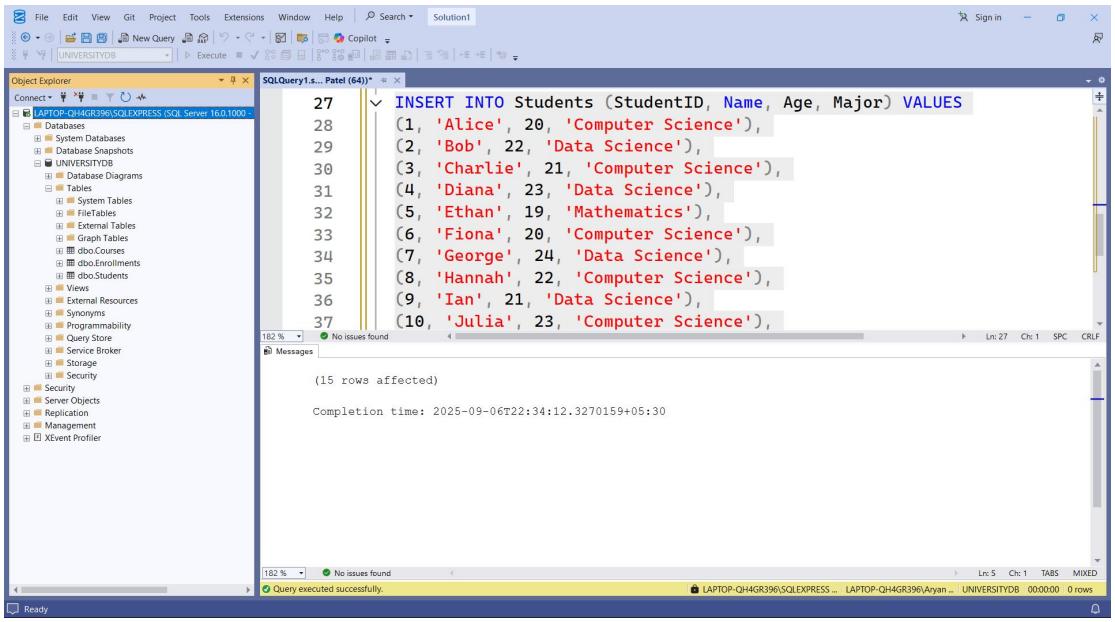

CREATE DATABASE UNIVERSITYDB;
Commands completed successfully.
Completion time: 2025-09-06T22:28:13.7399946+05:30
Query executed successfully.

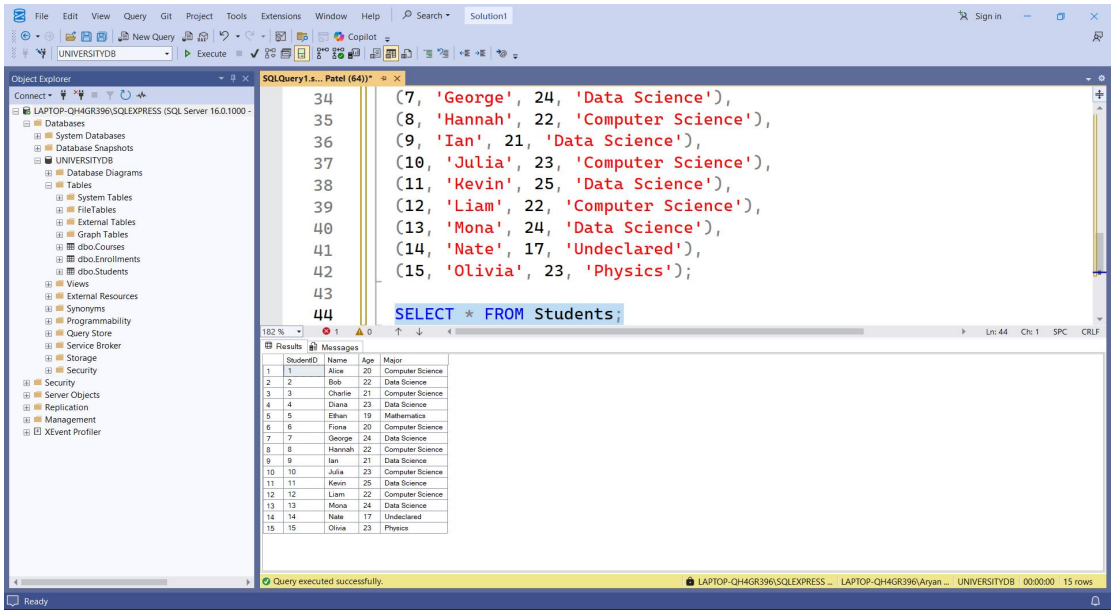

CREATE DATABASE UNIVERSITYDB;
USE UniversityDB;
Commands completed successfully.
Completion time: 2025-09-06T22:29:39.8665528+05:30
Query executed successfully.

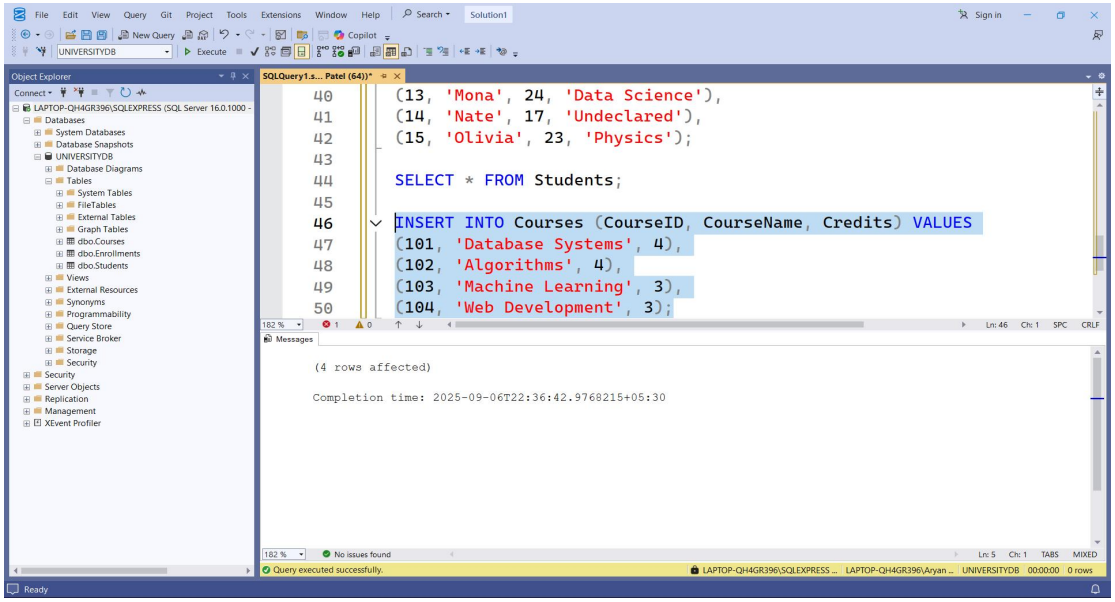

CREATE DATABASE UNIVERSITYDB;
USE UniversityDB;
CREATE TABLE Students (
StudentID INT PRIMARY KEY,
Name VARCHAR(50),
Age INT,
Major VARCHAR(50)
);
Commands completed successfully.
Completion time: 2025-09-06T22:30:30.5728419+05:30
Query executed successfully.

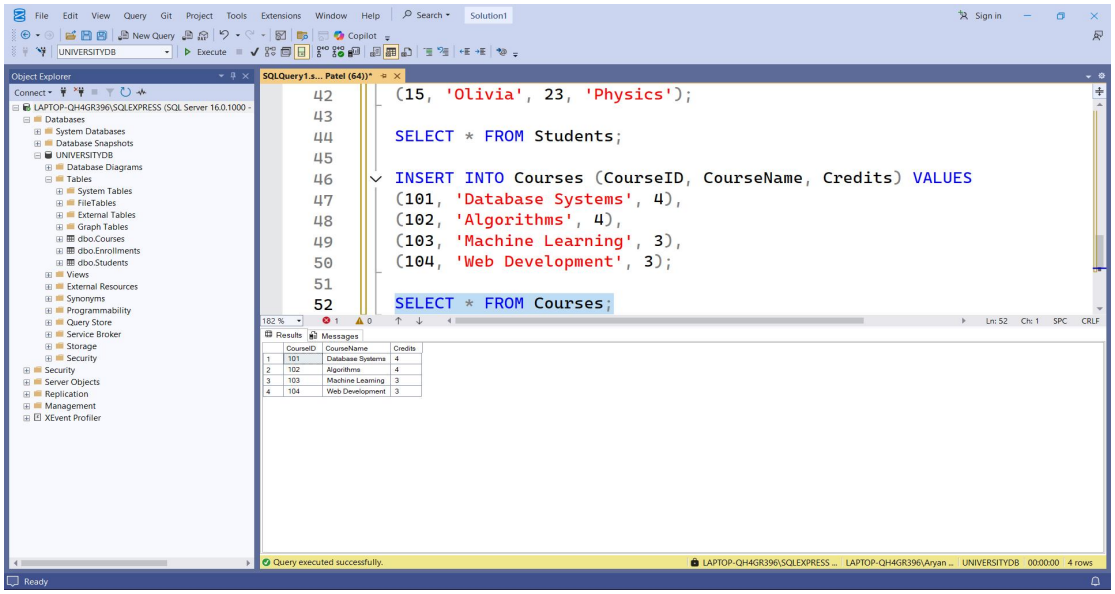

CREATE TABLE Students (
StudentID INT PRIMARY KEY,
Name VARCHAR(50),
Age INT,
Major VARCHAR(50)
);
CREATE TABLE Courses (
CourseID INT PRIMARY KEY,
CourseName VARCHAR(50),
Credits INT
);
Commands completed successfully.
Completion time: 2025-09-06T22:31:18.7507832+05:30

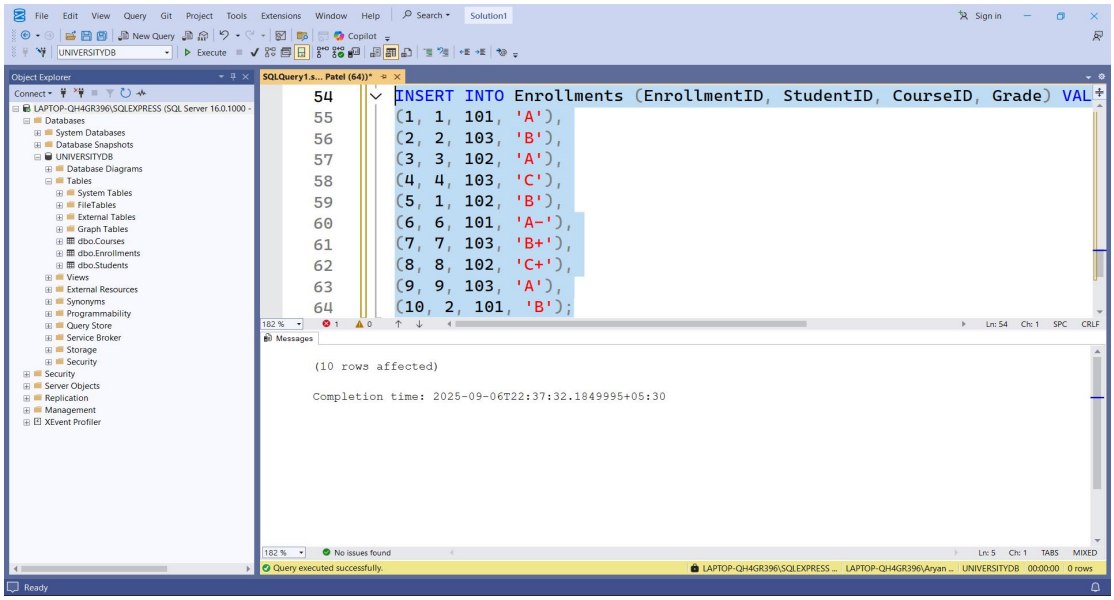

CourseID INT PRIMARY KEY,
CourseName VARCHAR(50),
Credits INT
);
CREATE TABLE Enrollments (
EnrollmentID INT PRIMARY KEY,
StudentID INT,
CourseID INT,
Grade CHAR(2),
FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
);
Commands completed successfully.
Completion time: 2025-09-06T22:32:24.5051577+05:30

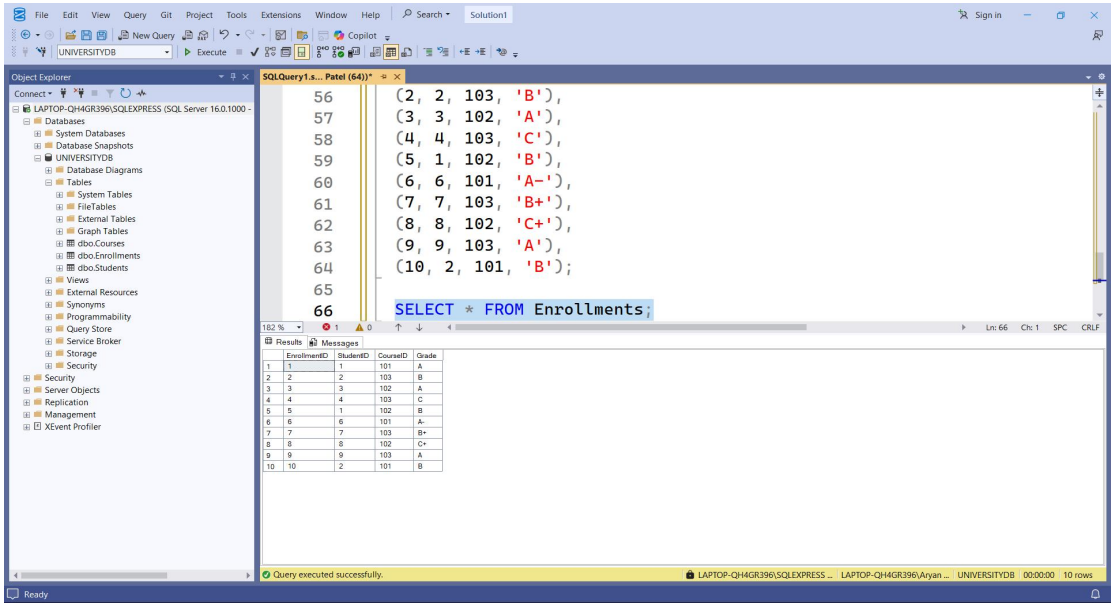

INSERT INTO Students (StudentID, Name, Age, Major) VALUES
(1, 'Alice', 20, 'Computer Science'),
(2, 'Bob', 22, 'Data Science'),
(3, 'Charlie', 21, 'Computer Science'),
(4, 'Diana', 23, 'Data Science'),
(5, 'Ethan', 19, 'Mathematics'),
(6, 'Fiona', 20, 'Computer Science'),
(7, 'George', 24, 'Data Science'),
(8, 'Hannah', 22, 'Computer Science'),
(9, 'Ian', 21, 'Data Science'),
(10, 'Julia', 23, 'Computer Science'),
(15 rows affected)
Completion time: 2025-09-06T22:34:12.3270159+05:30

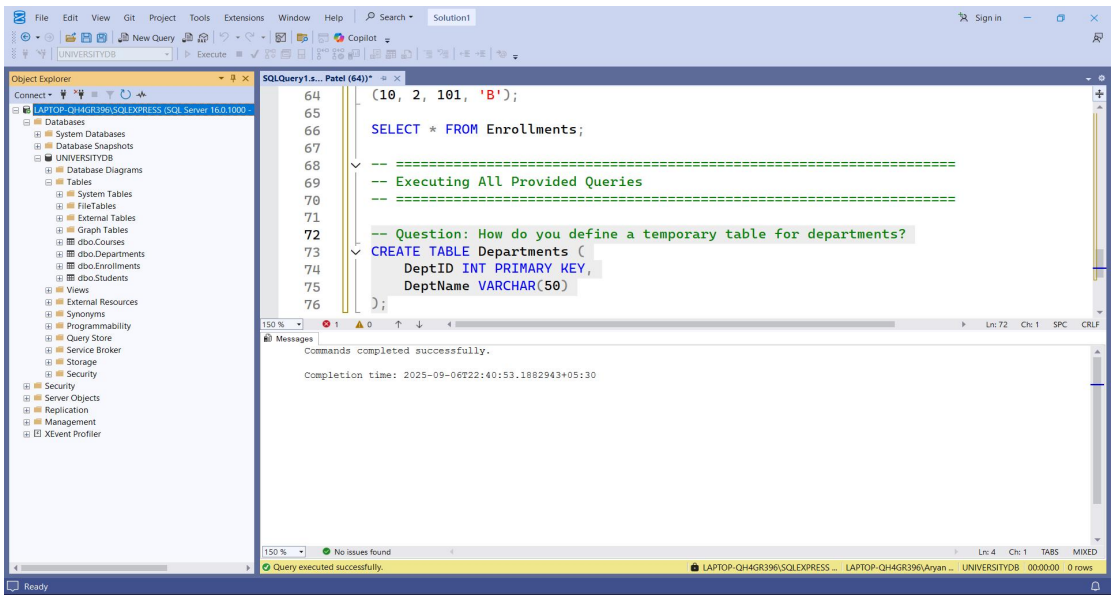

(7, 'George', 24, 'Data Science'),
(8, 'Hannah', 22, 'Computer Science'),
(9, 'Ian', 21, 'Data Science'),
(10, 'Julia', 23, 'Computer Science'),
(11, 'Kevin', 25, 'Data Science'),
(12, 'Liam', 22, 'Computer Science'),
(13, 'Mona', 24, 'Data Science'),
(14, 'Nate', 17, 'Undeclared'),
(15, 'Olivia', 23, 'Physics');
SELECT * FROM Students;
Query executed successfully.

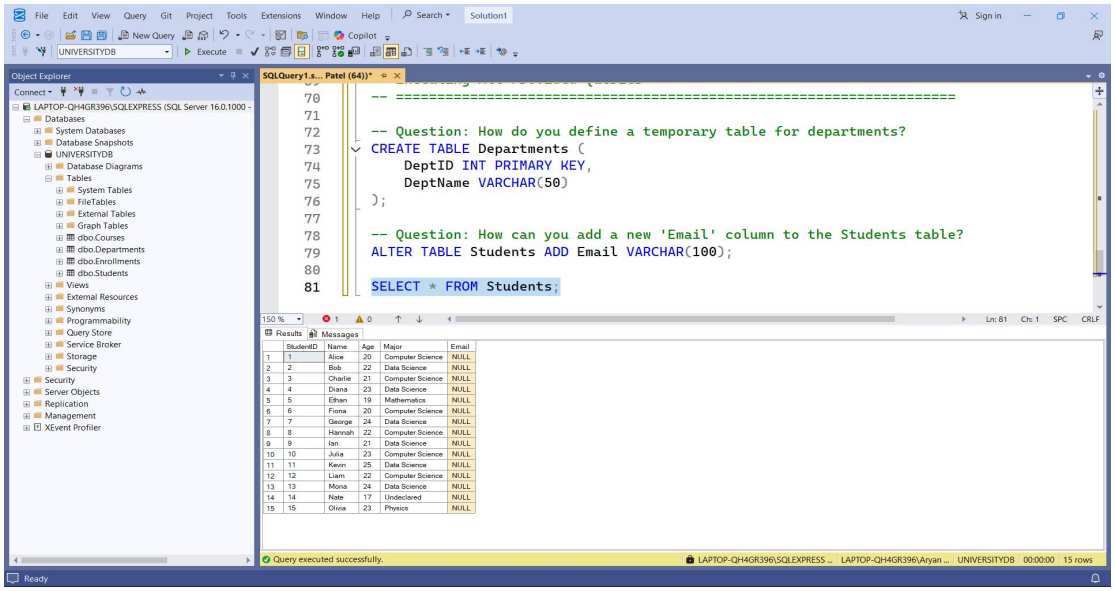

(13, 'Mona', 24, 'Data Science'),
(14, 'Nate', 17, 'Undeclared'),
(15, 'Olivia', 23, 'Physics');
SELECT * FROM Students;
INSERT INTO Courses (CourseID, CourseName, Credits) VALUES
(101, 'Database Systems', 4),
(102, 'Algorithms', 4),
(103, 'Machine Learning', 3),
(104, 'Web Development', 3);
(4 rows affected)
Completion time: 2025-09-06T22:36:42.9768215+05:30
Query executed successfully.

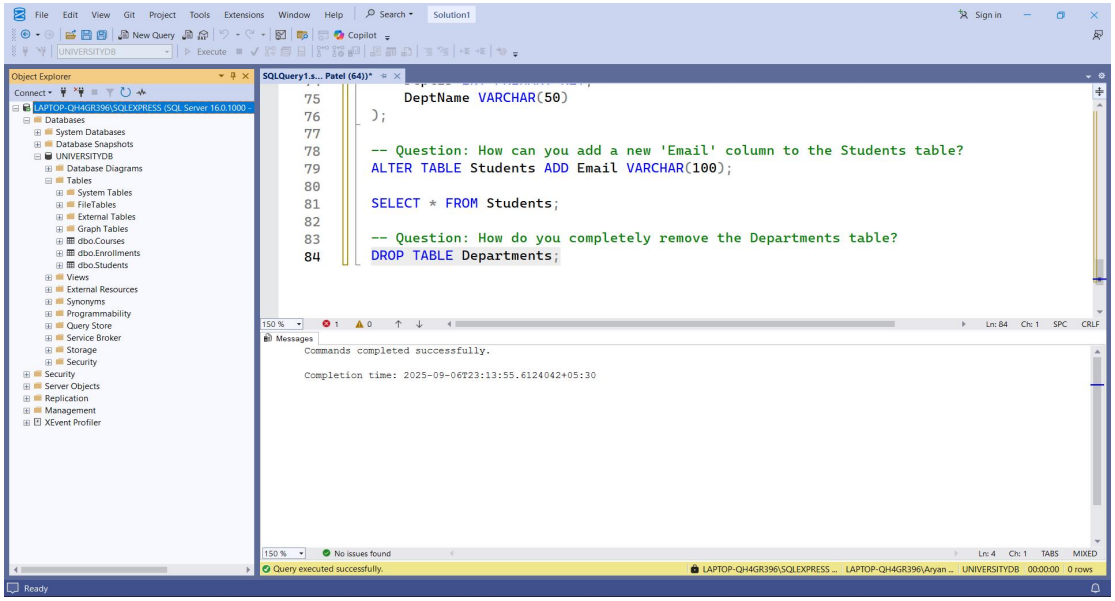

(15, 'Olivia', 23, 'Physics');
SELECT * FROM Students;
INSERT INTO Courses (CourseID, CourseName, Credits) VALUES
(101, 'Database Systems', 4),
(102, 'Algorithms', 4),
(103, 'Machine Learning', 3),
(104, 'Web Development', 3);
SELECT * FROM Courses;
Query executed successfully.

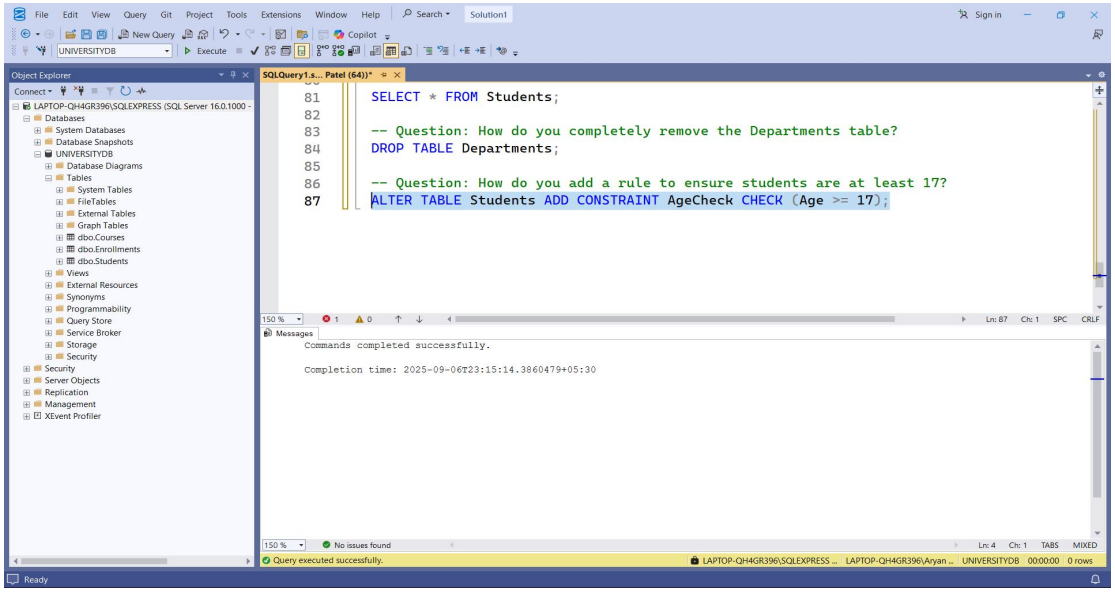

INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID, Grade) VAL
(1, 1, 101, 'A'),
(2, 2, 103, 'B'),
(3, 3, 102, 'A'),
(4, 4, 103, 'C'),
(5, 1, 102, 'B'),
(6, 6, 101, 'A-'),
(7, 7, 103, 'B+'),
(8, 8, 102, 'C+'),
(9, 9, 103, 'A'),
(10, 2, 101, 'B');
(10 rows affected)
Completion time: 2025-09-06T22:37:32.1849995+05:30
Query executed successfully.

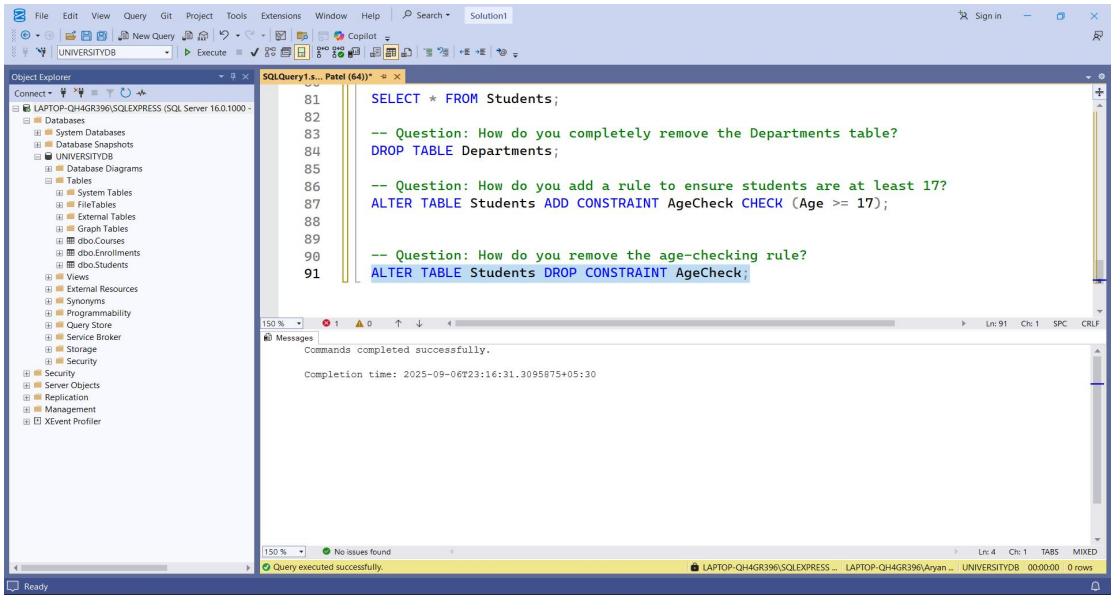

(2, 2, 103, 'B'),
(3, 3, 102, 'A'),
(4, 4, 103, 'C'),
(5, 1, 102, 'B'),
(6, 6, 101, 'A-'),
(7, 7, 103, 'B+'),
(8, 8, 102, 'C+'),
(9, 9, 103, 'A'),
(10, 2, 101, 'B');
SELECT * FROM Enrollments;
EnrollmentID StudentID CourseID Grade
1 1 1 101 A
2 2 2 103 B
3 3 3 102 A
4 4 4 103 C
5 5 1 102 B
6 6 6 101 A-
7 7 7 103 B+
8 8 8 102 C+
9 9 9 103 A
10 10 2 101 B
Query executed successfully.

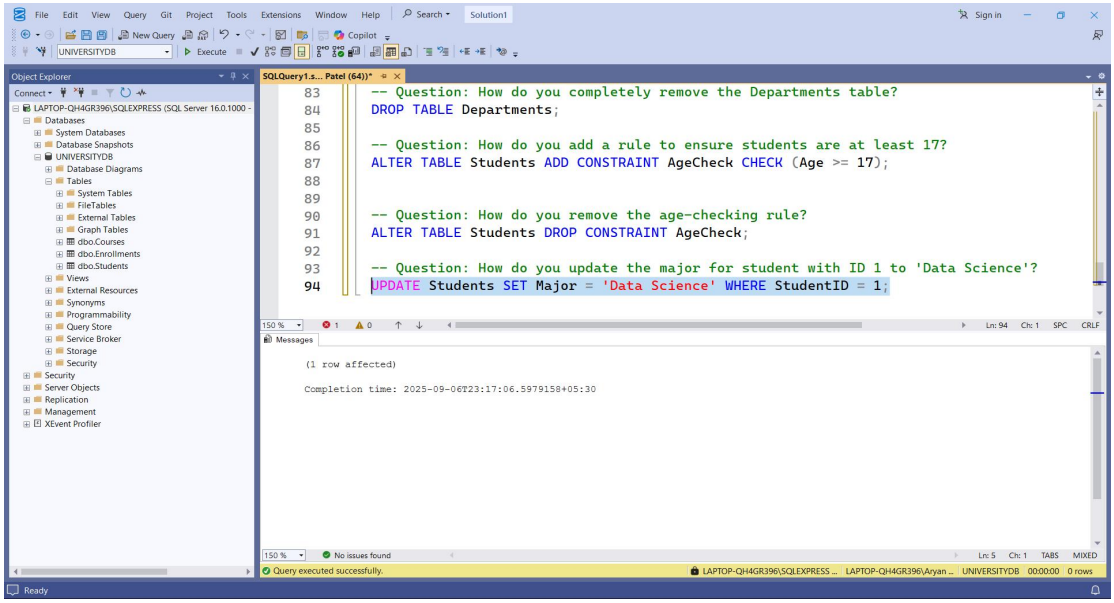

(10, 2, 101, 'B');
SELECT * FROM Enrollments;
-- ======================================================================
-- Executing All Provided Queries
-- ======================================================================
-- Question: How do you define a temporary table for departments?
CREATE TABLE Departments (
DeptID INT PRIMARY KEY,
DeptName VARCHAR(50)
);
Commands completed successfully.
Completion time: 2025-09-06T22:40:53.1882943+05:30
Query executed successfully.

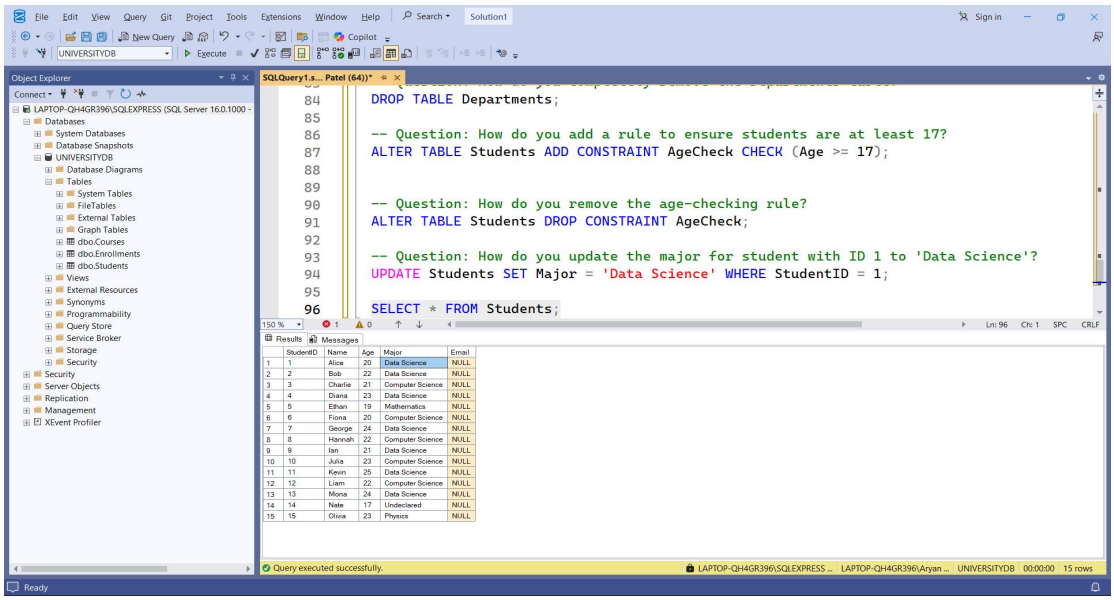

-- Question: How do you define a temporary table for departments?
CREATE TABLE Departments (
DeptID INT PRIMARY KEY,
DeptName VARCHAR(50)
);
-- Question: How can you add a new 'Email' column to the Students table?
ALTER TABLE Students ADD Email VARCHAR(100);
SELECT * FROM Students;
Query executed successfully.

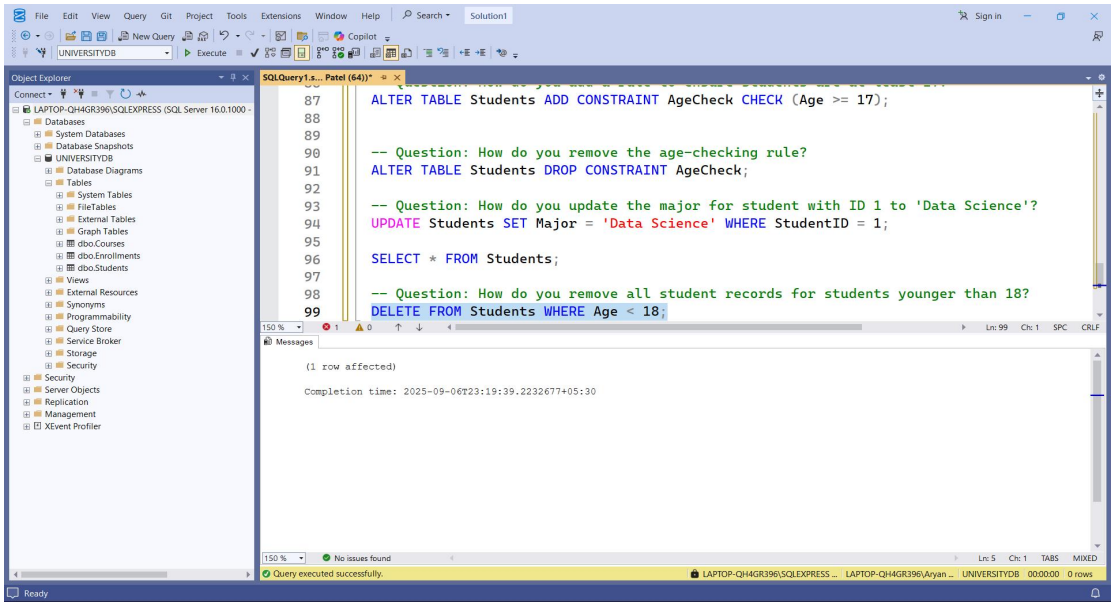

DeptName VARCHAR(50)
);
-- Question: How can you add a new 'Email' column to the Students table?
ALTER TABLE Students ADD Email VARCHAR(100);
SELECT * FROM Students;
-- Question: How do you completely remove the Departments table?
DROP TABLE Departments;
Commands completed successfully.
Completion time: 2025-09-06T23:13:55.6124042+05:30
Query executed successfully.

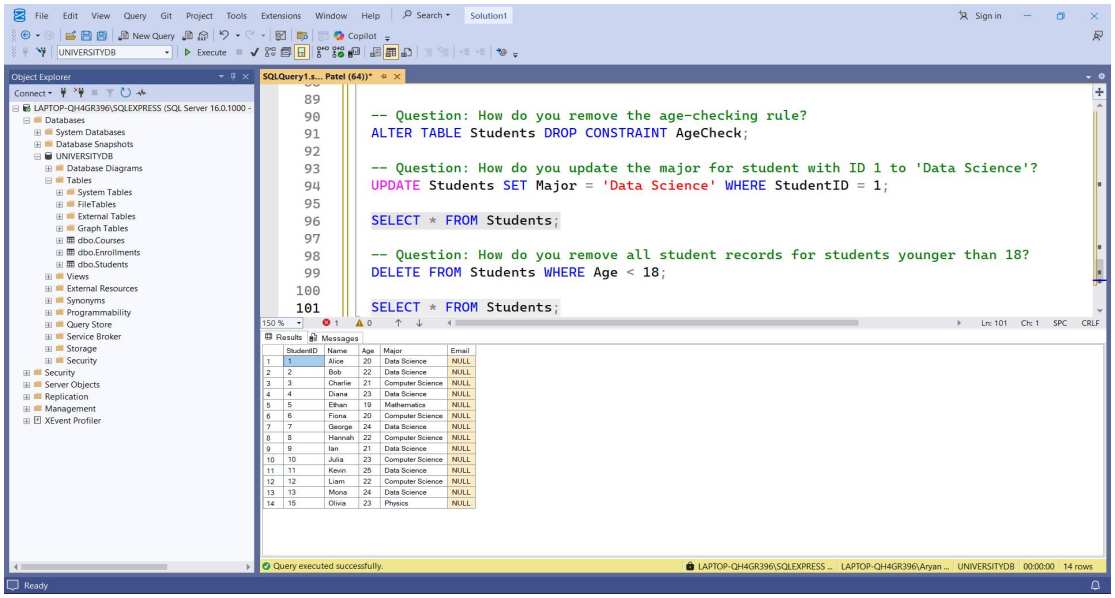

SELECT * FROM Students;
-- Question: How do you completely remove the Departments table?
DROP TABLE Departments;
-- Question: How do you add a rule to ensure students are at least 17?
ALTER TABLE Students ADD CONSTRAINT AgeCheck CHECK (Age >= 17);
Commands completed successfully.
Completion time: 2025-09-06T23:15:14.3860479+05:30
Query executed successfully.

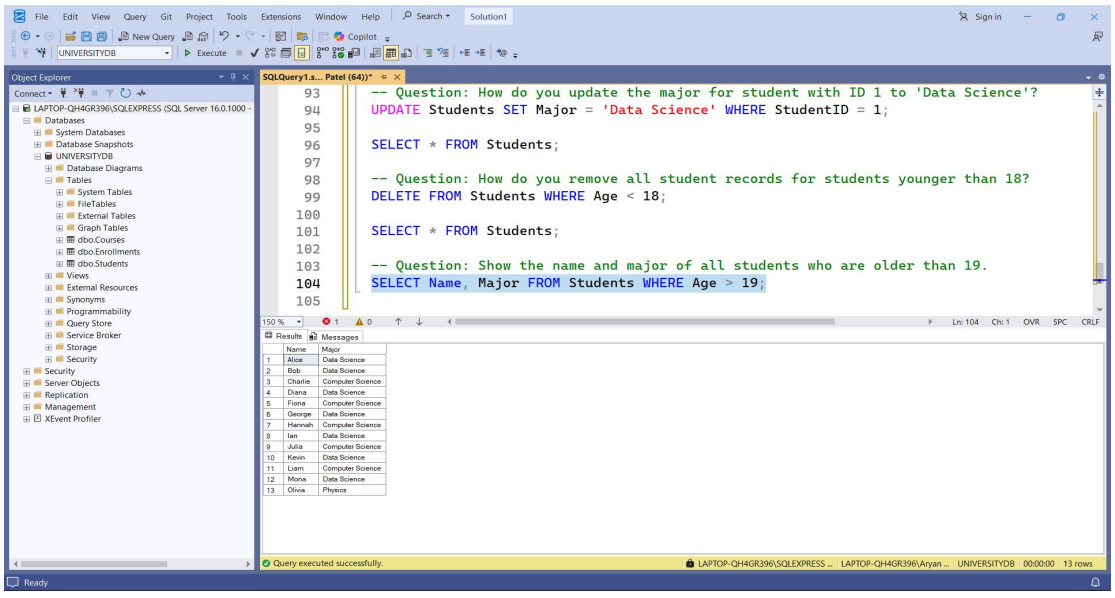

SELECT * FROM Students;
-- Question: How do you completely remove the Departments table?
DROP TABLE Departments;
-- Question: How do you add a rule to ensure students are at least 17?
ALTER TABLE Students ADD CONSTRAINT AgeCheck CHECK (Age >= 17);
-- Question: How do you remove the age-checking rule?
ALTER TABLE Students DROP CONSTRAINT AgeCheck;
Commands completed successfully.
Completion time: 2025-09-06T23:16:31.3095875+05:30

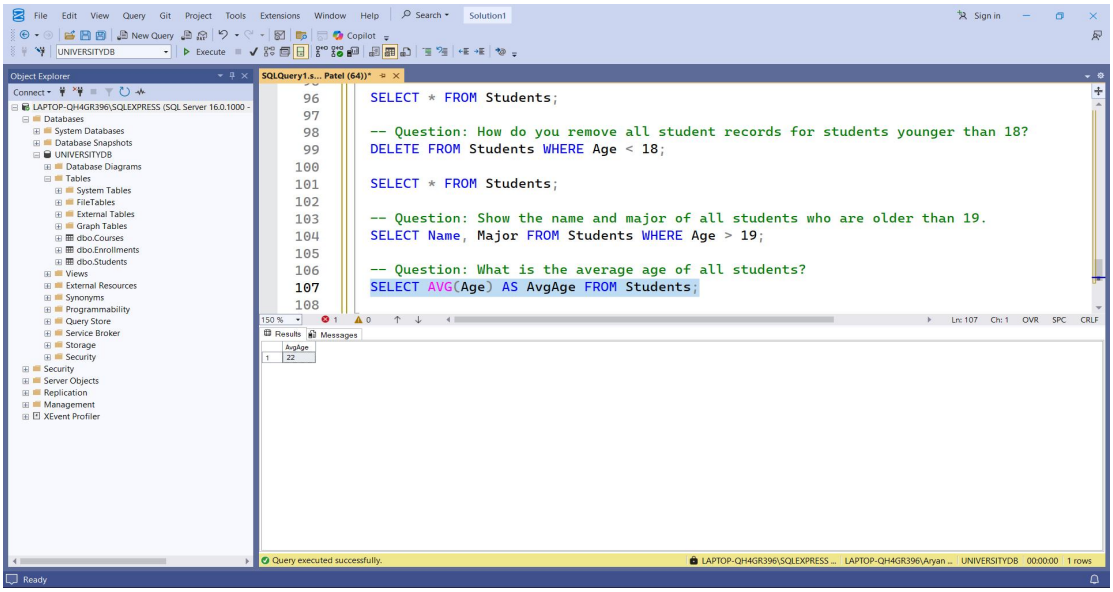

-- Question: How do you completely remove the Departments table?
DROP TABLE Departments;
-- Question: How do you add a rule to ensure students are at least 17?
ALTER TABLE Students ADD CONSTRAINT AgeCheck CHECK (Age >= 17);
-- Question: How do you remove the age-checking rule?
ALTER TABLE Students DROP CONSTRAINT AgeCheck;
-- Question: How do you update the major for student with ID 1 to 'Data Science'?
UPDATE Students SET Major = 'Data Science' WHERE StudentID = 1;
(1 row affected)
Completion time: 2025-09-06T23:17:06.5979158+05:30

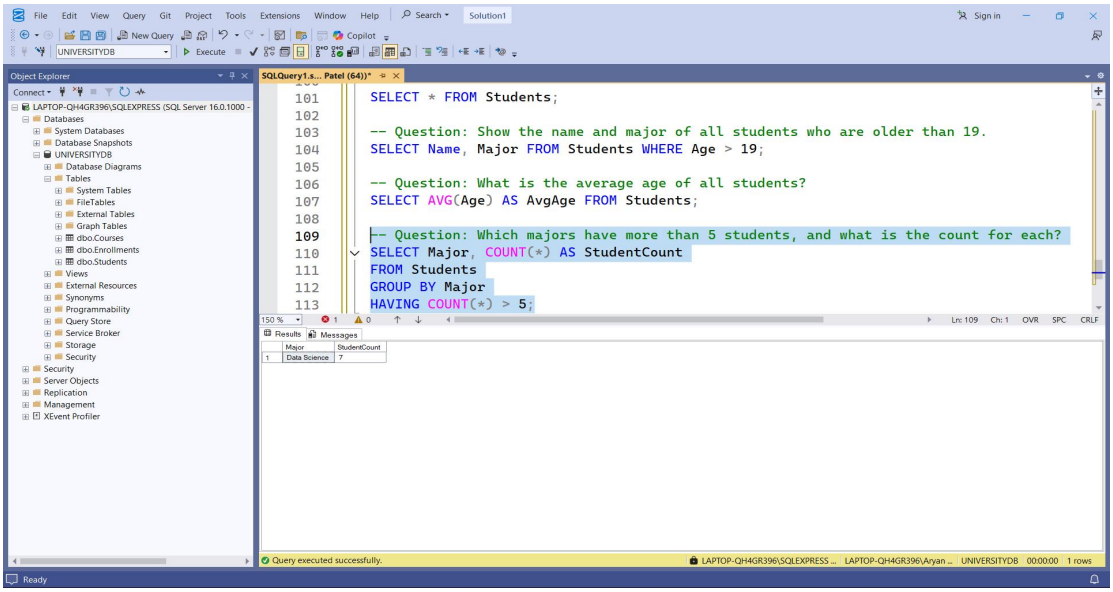

DROP TABLE Departments;
-- Question: How do you add a rule to ensure students are at least 17?
ALTER TABLE Students ADD CONSTRAINT AgeCheck CHECK (Age >= 17);
-- Question: How do you remove the age-checking rule?
ALTER TABLE Students DROP CONSTRAINT AgeCheck;
-- Question: How do you update the major for student with ID 1 to 'Data Science'?
UPDATE Students SET Major = 'Data Science' WHERE StudentID = 1;
SELECT * FROM Students;
StudentID Name Age Major Email
1 Alice 20 Data Science NULL
2 Bob 22 Data Science NULL
3 Charlie 21 Computer Science NULL
4 Diana 23 Data Science NULL
5 Ethan 19 Mathematics NULL
6 Fiona 20 Computer Science NULL
7 George 24 Data Science NULL
8 Hannah 22 Computer Science NULL
9 Ian 21 Data Science NULL
10 Julia 23 Computer Science NULL
11 Kevin 25 Data Science NULL
12 Liam 22 Computer Science NULL
13 Mona 24 Data Science NULL
14 Nate 17 Undeclared NULL
15 Olivia 23 Physics NULL

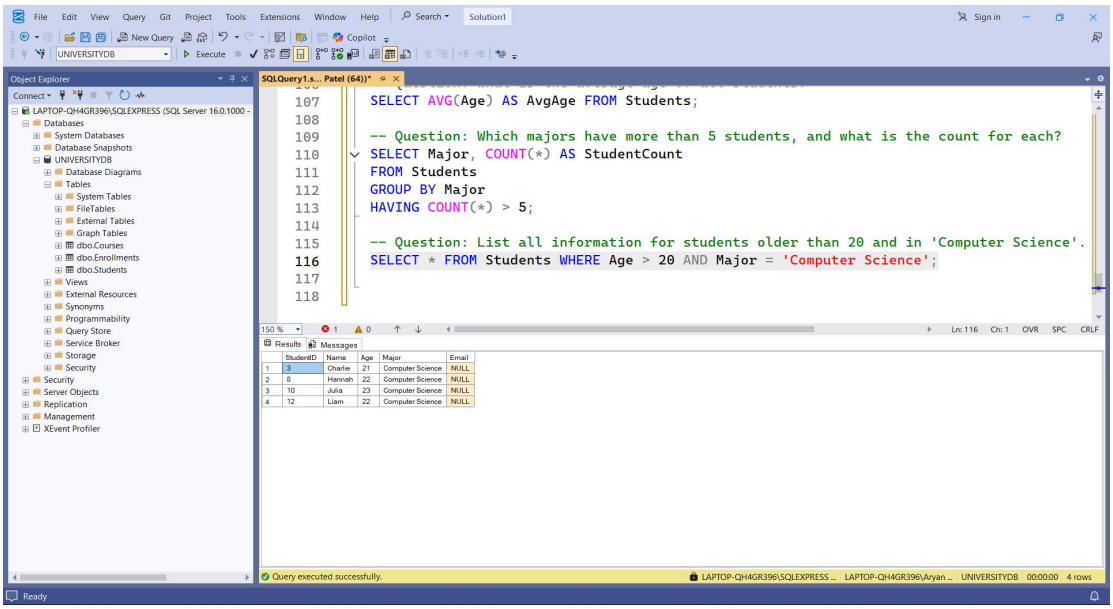

ALTER TABLE Students ADD CONSTRAINT AgeCheck CHECK (Age >= 17);
-- Question: How do you remove the age-checking rule?
ALTER TABLE Students DROP CONSTRAINT AgeCheck;
-- Question: How do you update the major for student with ID 1 to 'Data Science'?
UPDATE Students SET Major = 'Data Science' WHERE StudentID = 1;
SELECT * FROM Students;
-- Question: How do you remove all student records for students younger than 18?
DELETE FROM Students WHERE Age < 18;
(1 row affected)
Completion time: 2025-09-06T23:19:39.2232677+05:30
Query executed successfully.

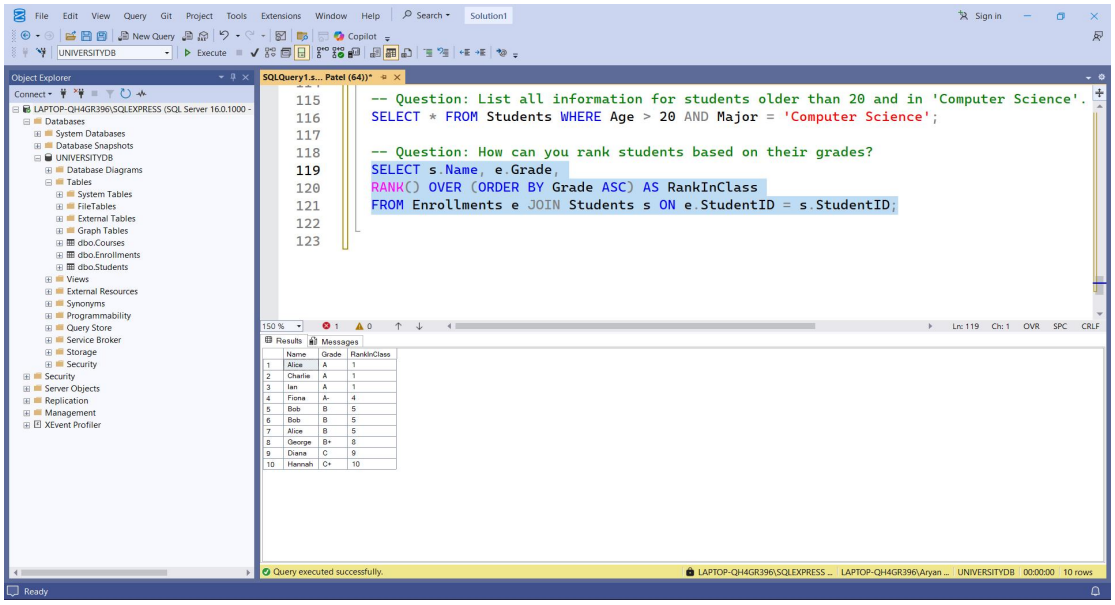

-- Question: How do you remove the age-checking rule?
ALTER TABLE Students DROP CONSTRAINT AgeCheck;
-- Question: How do you update the major for student with ID 1 to 'Data Science'?
UPDATE Students SET Major = 'Data Science' WHERE StudentID = 1;
SELECT * FROM Students;
-- Question: How do you remove all student records for students younger than 18?
DELETE FROM Students WHERE Age < 18;
SELECT * FROM Students;
Query executed successfully.

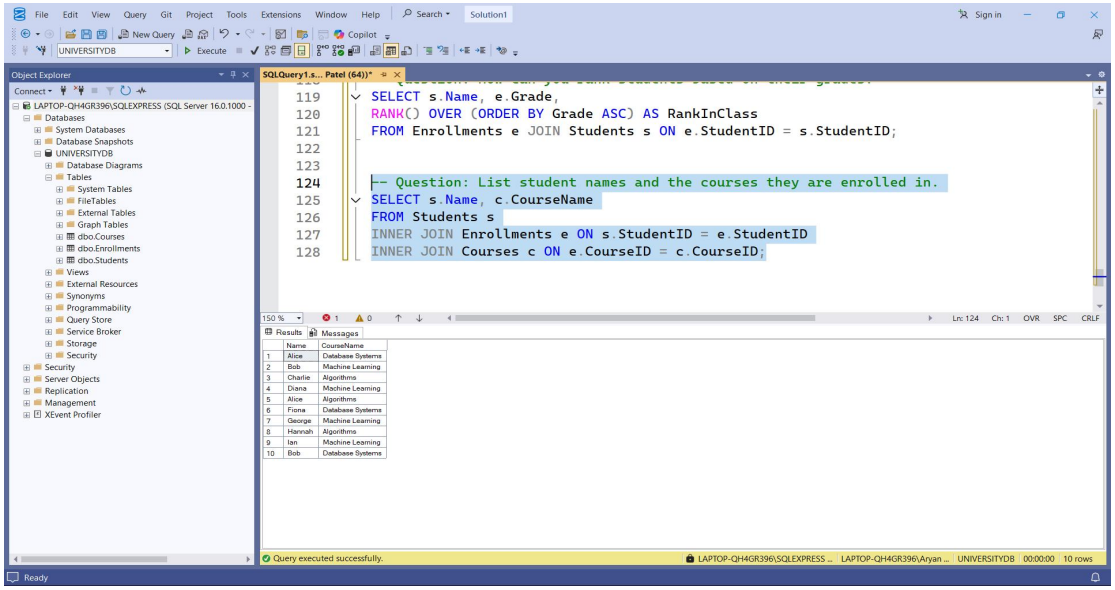

-- Question: How do you update the major for student with ID 1 to 'Data Science'?
UPDATE Students SET Major = 'Data Science' WHERE StudentID = 1;
SELECT * FROM Students;
-- Question: How do you remove all student records for students younger than 18?
DELETE FROM Students WHERE Age < 18;
SELECT * FROM Students;
-- Question: Show the name and major of all students who are older than 19.
SELECT Name, Major FROM Students WHERE Age > 19;
Query executed successfully.

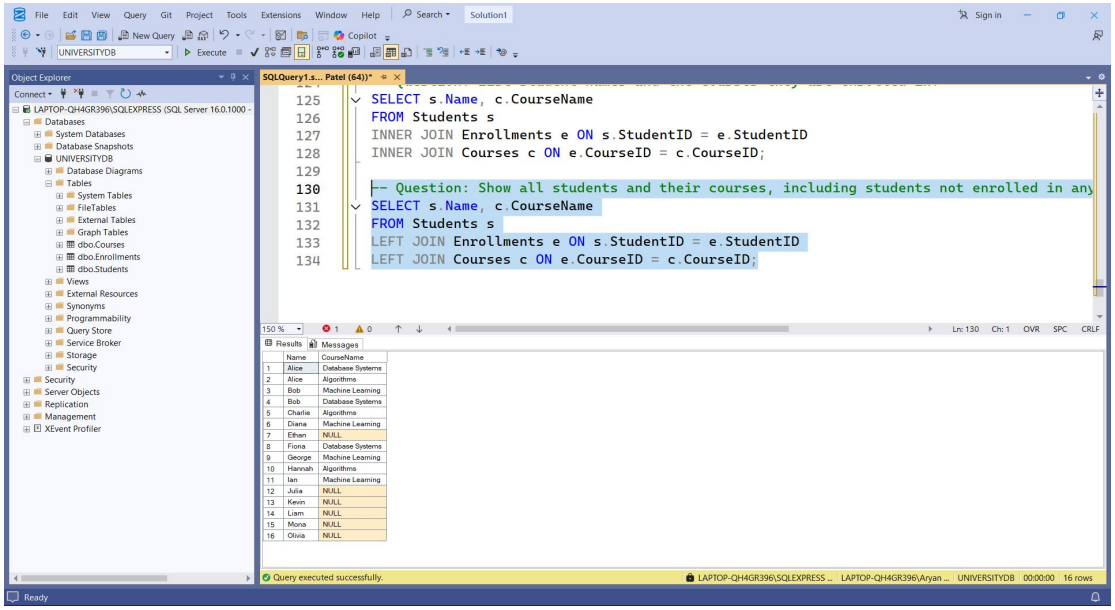

SELECT * FROM Students;
-- Question: How do you remove all student records for students younger than 18?
DELETE FROM Students WHERE Age < 18;
SELECT * FROM Students;
-- Question: Show the name and major of all students who are older than 19.
SELECT Name, Major FROM Students WHERE Age > 19;
-- Question: What is the average age of all students?
SELECT AVG(Age) AS AvgAge FROM Students;
AvgAge
22
Query executed successfully.

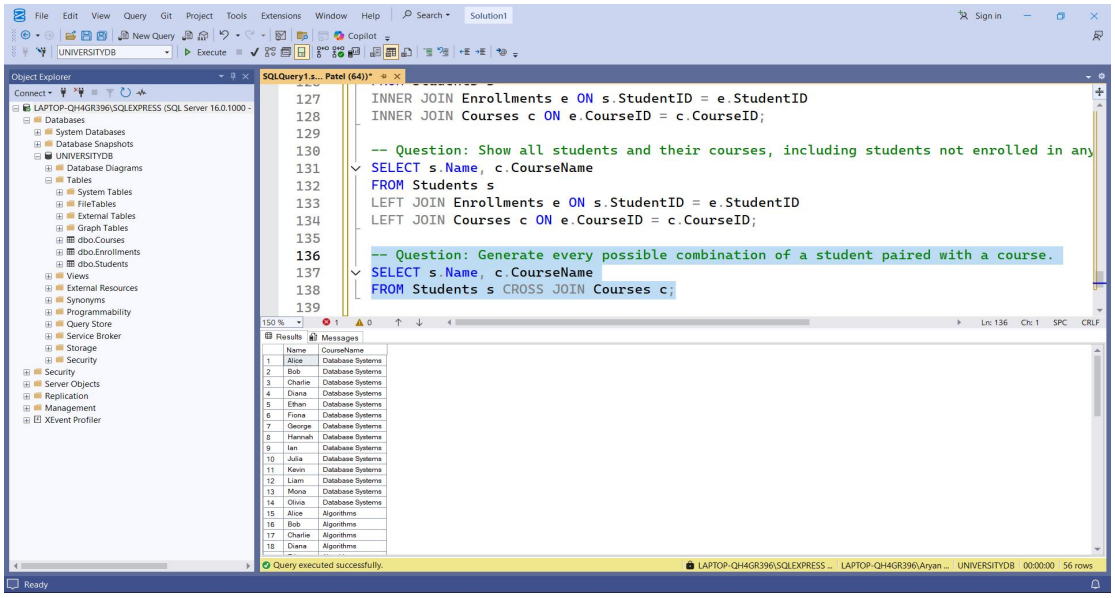

SELECT * FROM Students;
-- Question: Show the name and major of all students who are older than 19.
SELECT Name, Major FROM Students WHERE Age > 19;
-- Question: What is the average age of all students?
SELECT AVG(Age) AS AvgAge FROM Students;
-- Question: Which majors have more than 5 students, and what is the count for each?
SELECT Major, COUNT(*) AS StudentCount
FROM Students
GROUP BY Major
HAVING COUNT(*) > 5;
Major
StudentCount
Data Science
7
Query executed successfully.

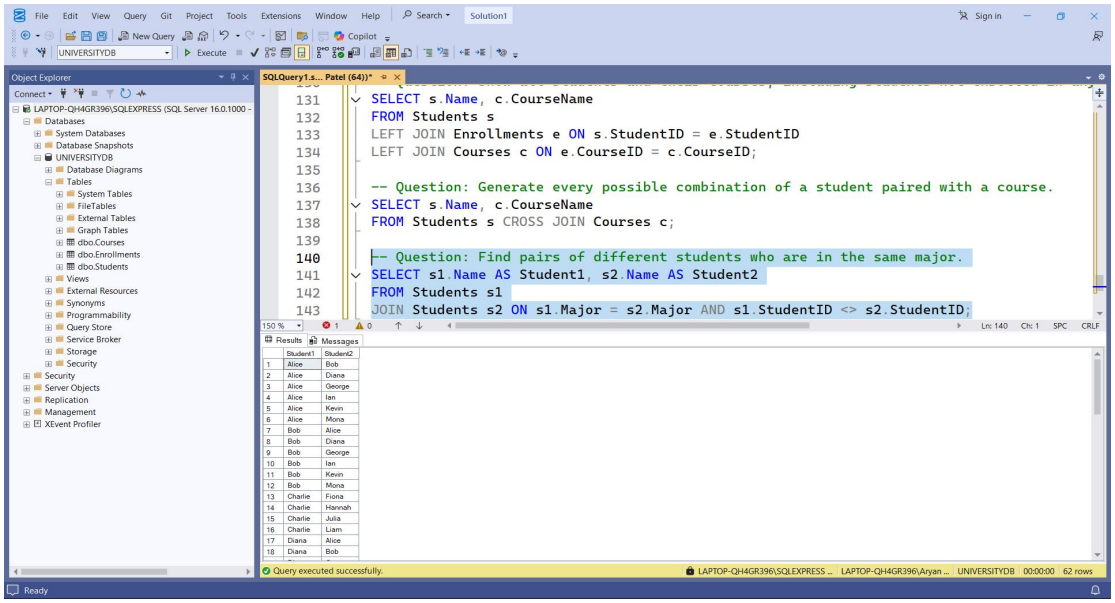

SELECT AVG(Age) AS AvgAge FROM Students;
-- Question: Which majors have more than 5 students, and what is the count for each?
SELECT Major, COUNT(*) AS StudentCount
FROM Students
GROUP BY Major
HAVING COUNT(*) > 5;
-- Question: List all information for students older than 20 and in 'Computer Science'.
SELECT * FROM Students WHERE Age > 20 AND Major = 'Computer Science';
StudentID
Name
Age
Major
Email
3
Charlie
21
Computer Science
NULL
8
Hannah
22
Computer Science
NULL
10
Julia
23
Computer Science
NULL
12
Liam
22
Computer Science
NULL
Query executed successfully.

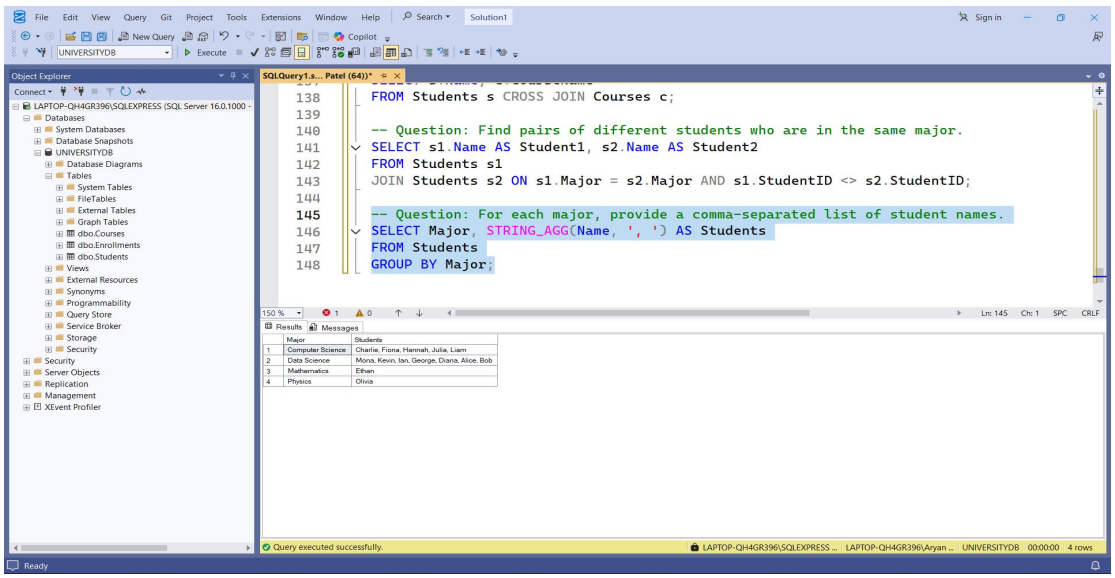

-- Question: List all information for students older than 20 and in 'Computer Science'.
SELECT * FROM Students WHERE Age > 20 AND Major = 'Computer Science';
-- Question: How can you rank students based on their grades?
SELECT s.Name, e.Grade,
RANK() OVER (ORDER BY Grade ASC) AS RankInClass
FROM Enrollments e JOIN Students s ON e.StudentID = s.StudentID;
Name Grade RankInClass
Alice A 1
Charlie A 1
Ian A 1
Fiona A- 4
Bob B 5
Bob B 5
Alice B 5
George B+ 8
Diana C 9
Hannah C+ 10
Query executed successfully.

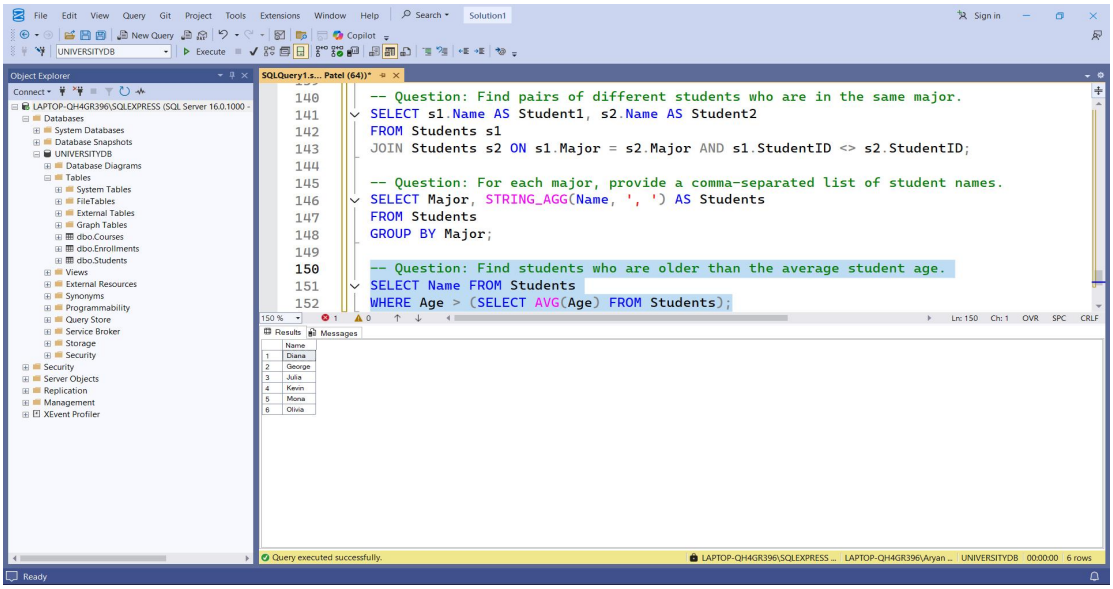

SELECT s.Name, e.Grade,
RANK() OVER (ORDER BY Grade ASC) AS RankInClass
FROM Enrollments e JOIN Students s ON e.StudentID = s.StudentID;
-- Question: List student names and the courses they are enrolled in.
SELECT s.Name, c.CourseName
FROM Students s
INNER JOIN Enrollments e ON s.StudentID = e.StudentID
INNER JOIN Courses c ON e.CourseID = c.CourseID;
Name CourseName
Alice Database Systems
Bob Machine Learning
Charlie Algorithms
Diana Machine Learning
Alice Algorithms
Fiona Database Systems
George Machine Learning
Hannah Algorithms
Ian Machine Learning
Bob Database Systems
Query executed successfully.

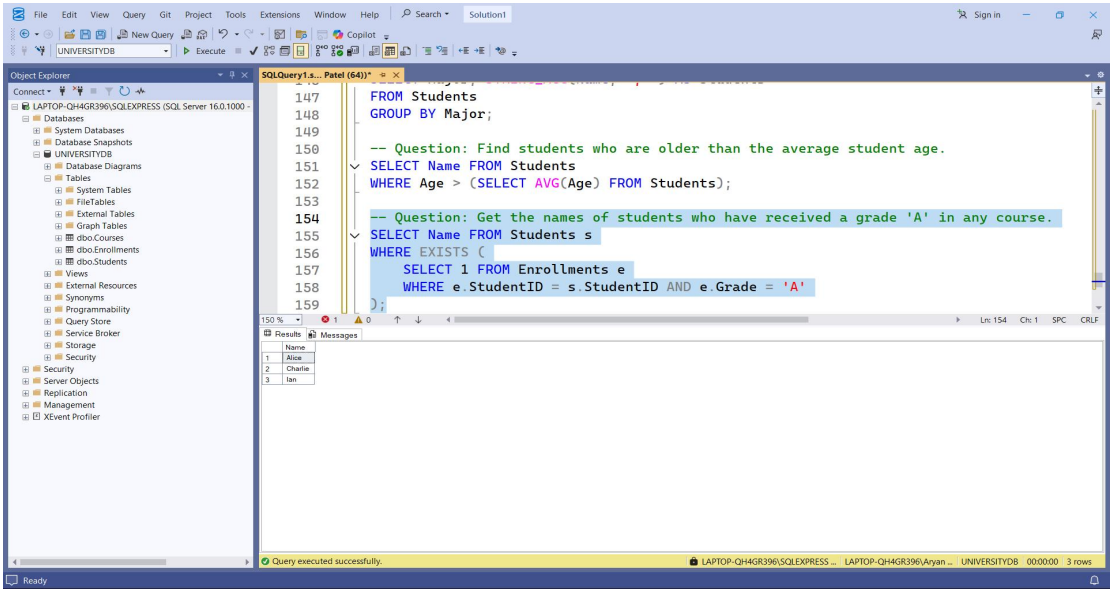

SELECT s.Name, c.CourseName
FROM Students s
INNER JOIN Enrollments e ON s.StudentID = e.StudentID
INNER JOIN Courses c ON e.CourseID = c.CourseID;
-- Question: Show all students and their courses, including students not enrolled in any
SELECT s.Name, c.CourseName
FROM Students s
LEFT JOIN Enrollments e ON s.StudentID = e.StudentID
LEFT JOIN Courses c ON e.CourseID = c.CourseID;
Name CourseName
Alice Database Systems
Alice Algorithms
Bob Machine Learning
Bob Database Systems
Charlie Algorithms
Diana Machine Learning
Ethan NULL
Fiona Database Systems
George Machine Learning
Hannah Algorithms
Ian Machine Learning
Julia NULL
Kevin NULL
Liam NULL
Mona NULL
Olivia NULL
Query executed successfully.

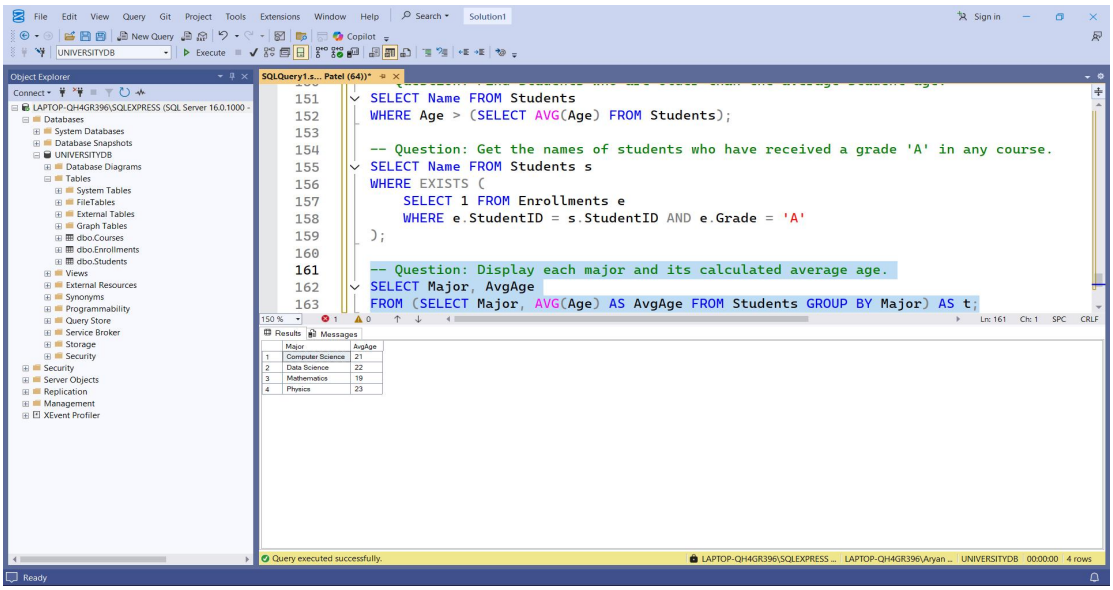

INNER JOIN Enrollments e ON s.StudentID = e.StudentID
INNER JOIN Courses c ON e.CourseID = c.CourseID;
-- Question: Show all students and their courses, including students not enrolled in any
SELECT s.Name, c.CourseName
FROM Students s
LEFT JOIN Enrollments e ON s.StudentID = e.StudentID
LEFT JOIN Courses c ON e.CourseID = c.CourseID;
-- Question: Generate every possible combination of a student paired with a course.
SELECT s.Name, c.CourseName
FROM Students s CROSS JOIN Courses c;
Query executed successfully.

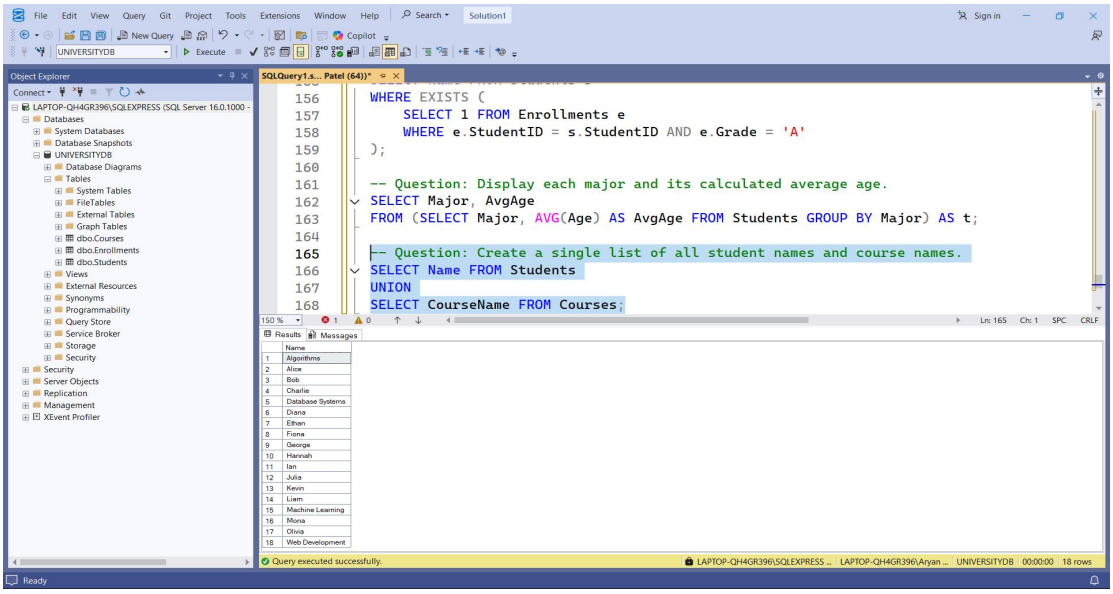

SELECT s.Name, c.CourseName
FROM Students s
LEFT JOIN Enrollments e ON s.StudentID = e.StudentID
LEFT JOIN Courses c ON e.CourseID = c.CourseID;
-- Question: Generate every possible combination of a student paired with a course.
SELECT s.Name, c.CourseName
FROM Students s CROSS JOIN Courses c;
-- Question: Find pairs of different students who are in the same major.
SELECT s1.Name AS Student1, s2.Name AS Student2
FROM Students s1
JOIN Students s2 ON s1.Major = s2.Major AND s1.StudentID <> s2.StudentID;
Query executed successfully.

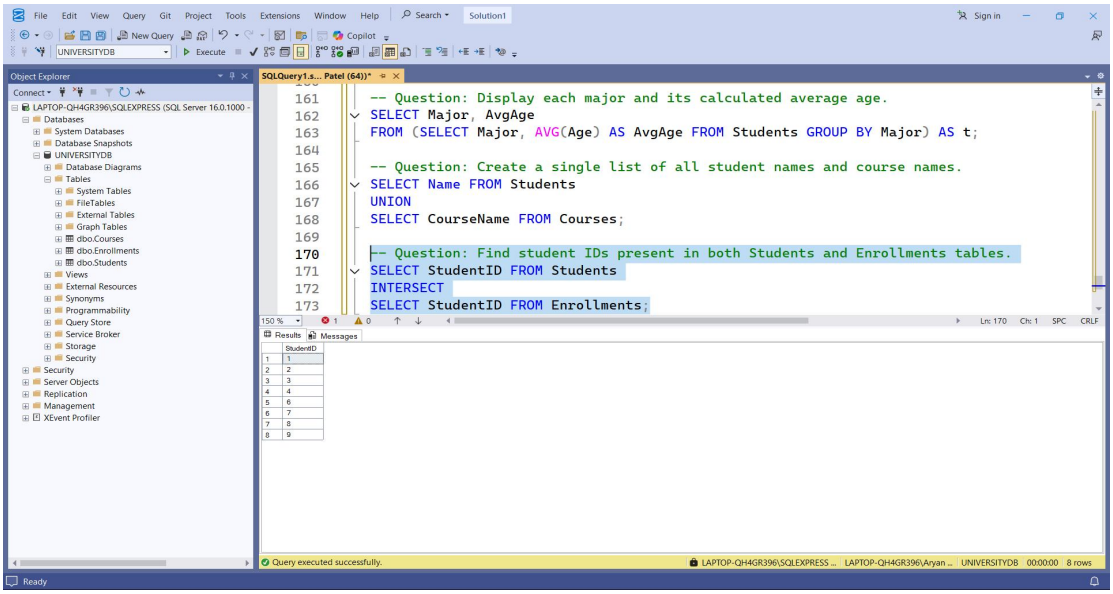

FROM Students s CROSS JOIN Courses c;
-- Question: Find pairs of different students who are in the same major.
SELECT s1.Name AS Student1, s2.Name AS Student2
FROM Students s1
JOIN Students s2 ON s1.Major = s2.Major AND s1.StudentID <> s2.StudentID;
-- Question: For each major, provide a comma-separated list of student names.
SELECT Major, STRING_AGG(Name, ', ') AS Students
FROM Students
GROUP BY Major;
Query executed successfully.

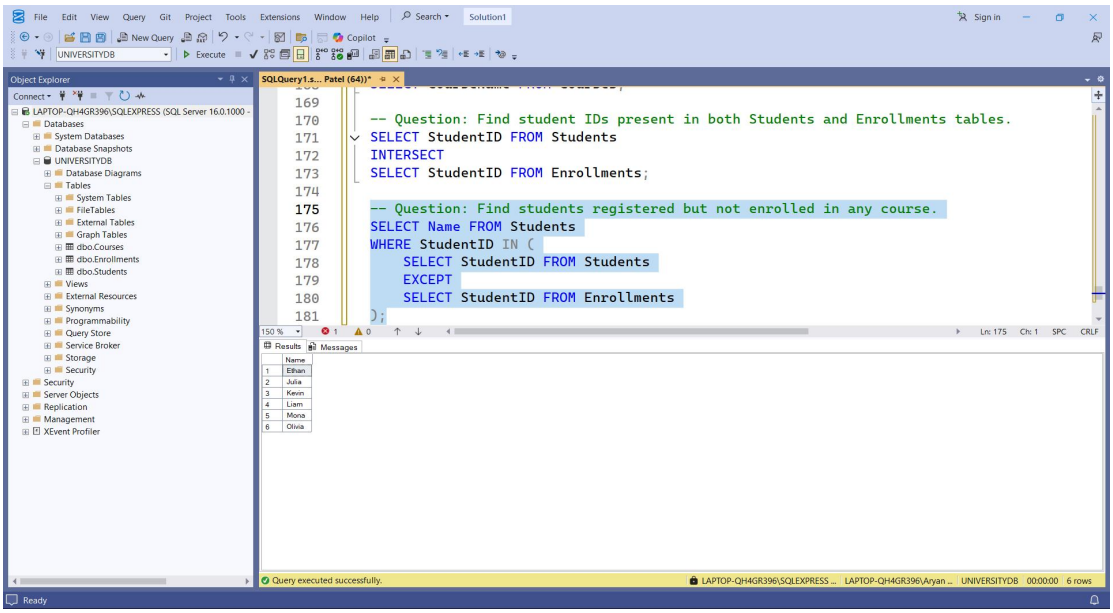

-- Question: Find pairs of different students who are in the same major.
SELECT s1.Name AS Student1, s2.Name AS Student2
FROM Students s1
JOIN Students s2 ON s1.Major = s2.Major AND s1.StudentID <> s2.StudentID;
-- Question: For each major, provide a comma-separated list of student names.
SELECT Major, STRING_AGG(Name, ', ') AS Students
FROM Students
GROUP BY Major;
-- Question: Find students who are older than the average student age.
SELECT Name FROM Students
WHERE Age > (SELECT AVG(Age) FROM Students);
Name
Diana
George
Julia
Kevin
Mona
Olivia
Query executed successfully.
Ready

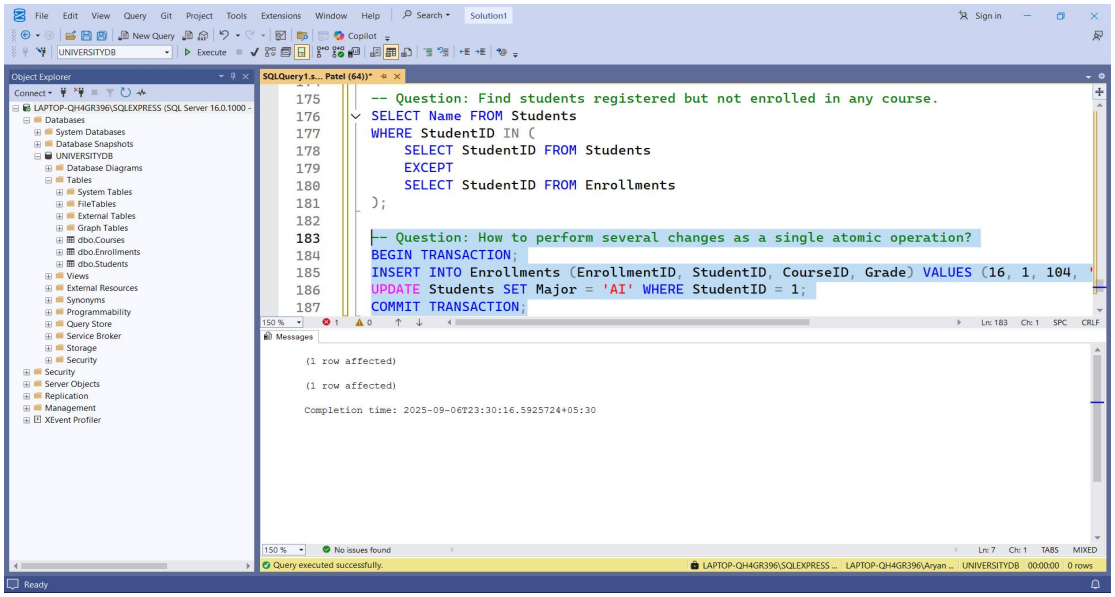

FROM Students
GROUP BY Major;
-- Question: Find students who are older than the average student age.
SELECT Name FROM Students
WHERE Age > (SELECT AVG(Age) FROM Students);
-- Question: Get the names of students who have received a grade 'A' in any course.
SELECT Name FROM Students s
WHERE EXISTS (
SELECT 1 FROM Enrollments e
WHERE e.StudentID = s.StudentID AND e.Grade = 'A'
);
Name
Alice
Charlie
Ian
Query executed successfully.
Ready

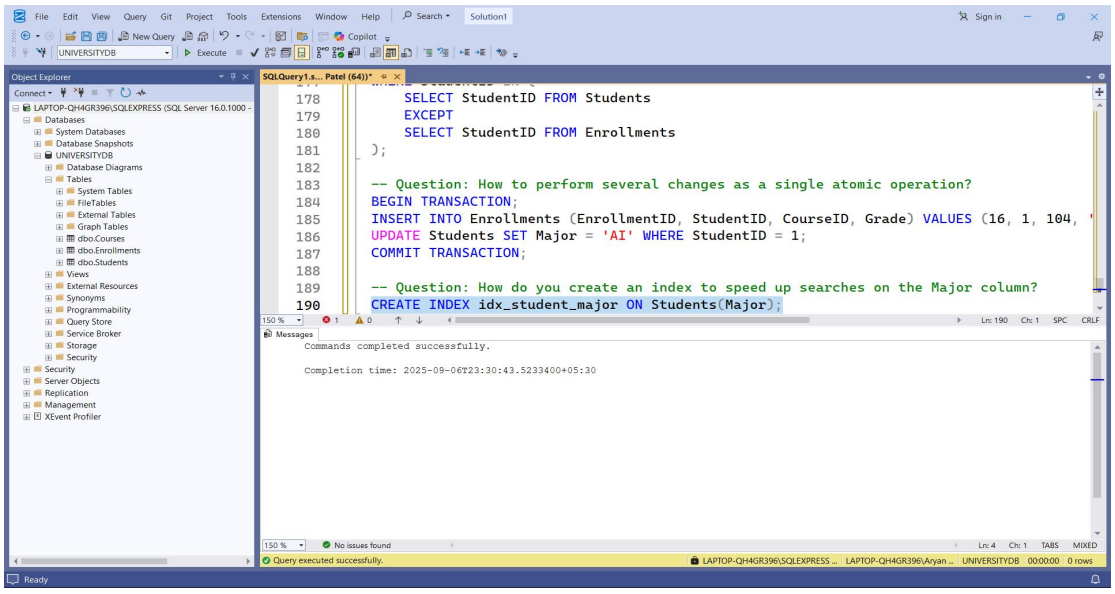

SELECT Name FROM Students
WHERE Age > (SELECT AVG(Age) FROM Students);
-- Question: Get the names of students who have received a grade 'A' in any course.
SELECT Name FROM Students s
WHERE EXISTS (
SELECT 1 FROM Enrollments e
WHERE e.StudentID = s.StudentID AND e.Grade = 'A'
);
-- Question: Display each major and its calculated average age.
SELECT Major, AvgAge
FROM (SELECT Major, AVG(Age) AS AvgAge FROM Students GROUP BY Major) AS t;
Major
AvgAge
Computer Science
21
Data Science
22
Mathematics
19
Physics
23
Query executed successfully.
Ready

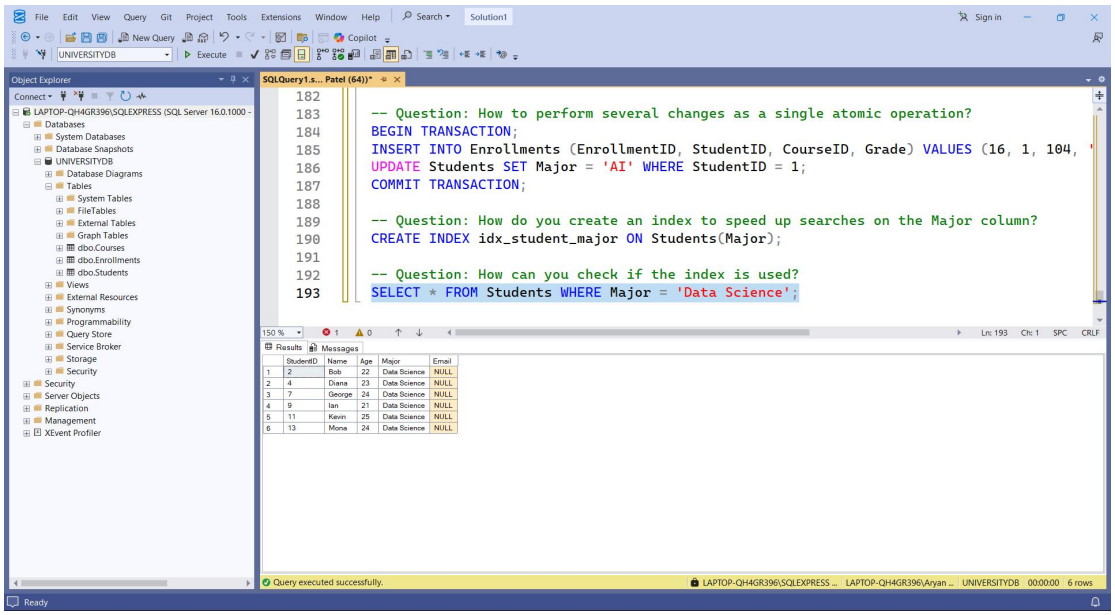

WHERE EXISTS (
SELECT 1 FROM Enrollments e
WHERE e.StudentID = s.StudentID AND e.Grade = 'A'
);
-- Question: Display each major and its calculated average age.
SELECT Major, AvgAge
FROM (SELECT Major, AVG(Age) AS AvgAge FROM Students GROUP BY Major) AS t;
-- Question: Create a single list of all student names and course names.
SELECT Name FROM Students
UNION
SELECT CourseName FROM Courses;
Name
Algorithms
Alice
Bob
Charlie
Database Systems
Diana
Ethan
Fiona
George
Hannah
Ian
Julia
Kevin
Liam
Machine Learning
Mona
Olivia
Web Development
Query executed successfully.
18 rows

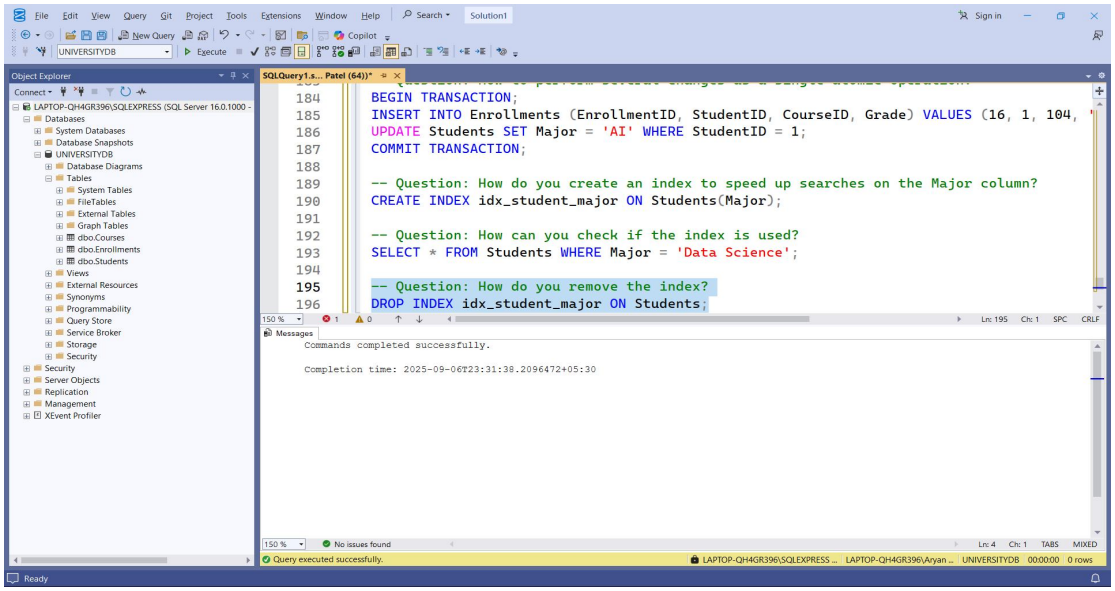

-- Question: Display each major and its calculated average age.
SELECT Major, AvgAge
FROM (SELECT Major, AVG(Age) AS AvgAge FROM Students GROUP BY Major) AS t;
-- Question: Create a single list of all student names and course names.
SELECT Name FROM Students
UNION
SELECT CourseName FROM Courses;
-- Question: Find student IDs present in both Students and Enrollments tables.
SELECT StudentID FROM Students
INTERSECT
SELECT StudentID FROM Enrollments;
StudentID
1
2
3
4
6
7
8
9
Query executed successfully.
8 rows

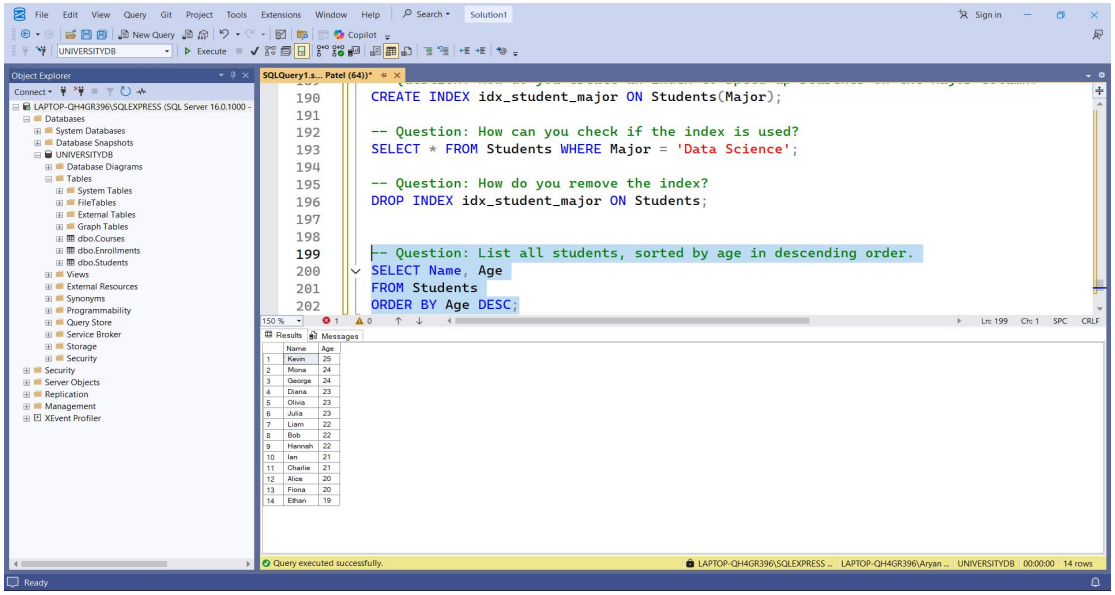

-- Question: Find student IDs present in both Students and Enrollments tables.
SELECT StudentID FROM Students
INTERSECT
SELECT StudentID FROM Enrollments;
-- Question: Find students registered but not enrolled in any course.
SELECT Name FROM Students
WHERE StudentID IN (
SELECT StudentID FROM Students
EXCEPT
SELECT StudentID FROM Enrollments
);
Name
Ethan
Julia
Kevin
Liam
Mona
Olivia
Query executed successfully.
6 rows

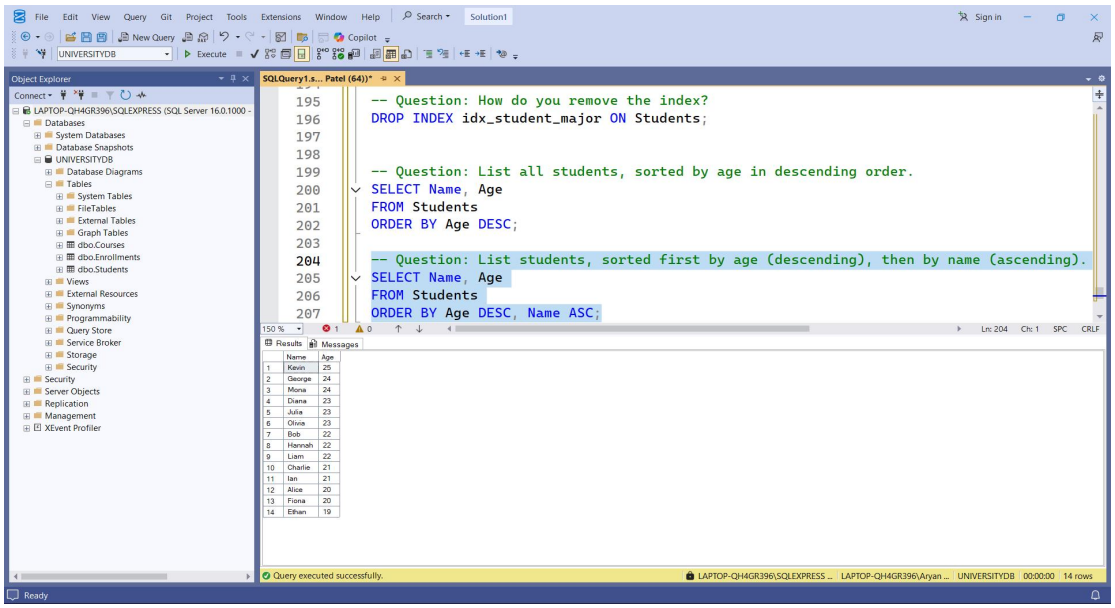

-- Question: Find students registered but not enrolled in any course.
SELECT Name FROM Students
WHERE StudentID IN (
SELECT StudentID FROM Students
EXCEPT
SELECT StudentID FROM Enrollments
);
-- Question: How to perform several changes as a single atomic operation?
BEGIN TRANSACTION;
INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID, Grade) VALUES (16, 1, 104,
UPDATE Students SET Major = 'AI' WHERE StudentID = 1;
COMMIT TRANSACTION;
(1 row affected)
(1 row affected)
Completion time: 2025-09-06T23:30:16.5925724+05:30
Query executed successfully.

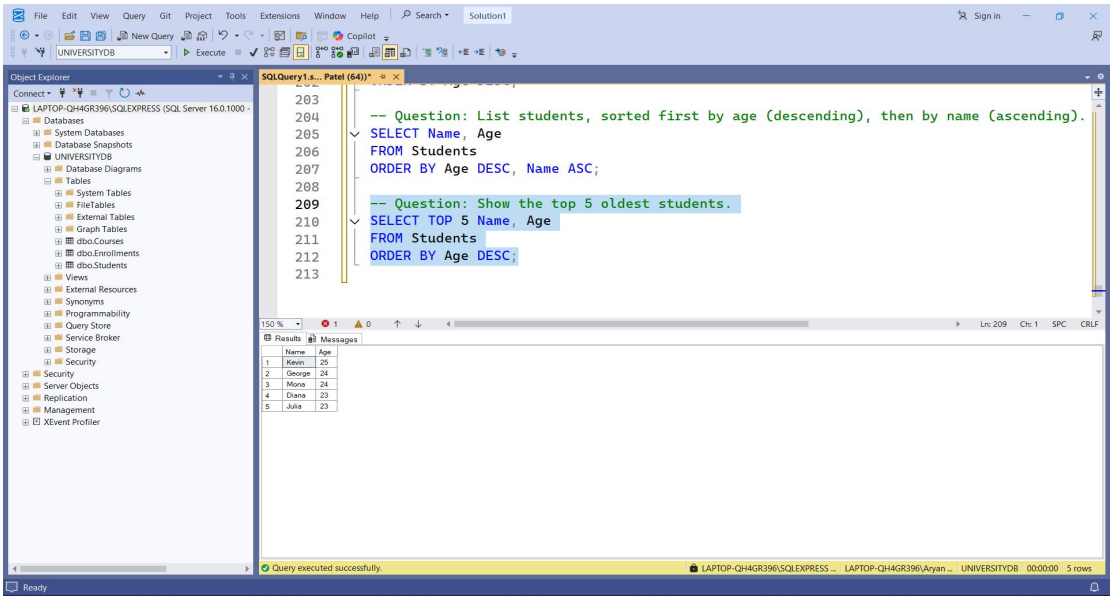

SELECT StudentID FROM Students
EXCEPT
SELECT StudentID FROM Enrollments
);
-- Question: How to perform several changes as a single atomic operation?
BEGIN TRANSACTION;
INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID, Grade) VALUES (16, 1, 104,
UPDATE Students SET Major = 'AI' WHERE StudentID = 1;
COMMIT TRANSACTION;
-- Question: How do you create an index to speed up searches on the Major column?
CREATE INDEX idx_student_major ON Students(Major);
Commands completed successfully.
Completion time: 2025-09-06T23:30:43.5233400+05:30
Query executed successfully.

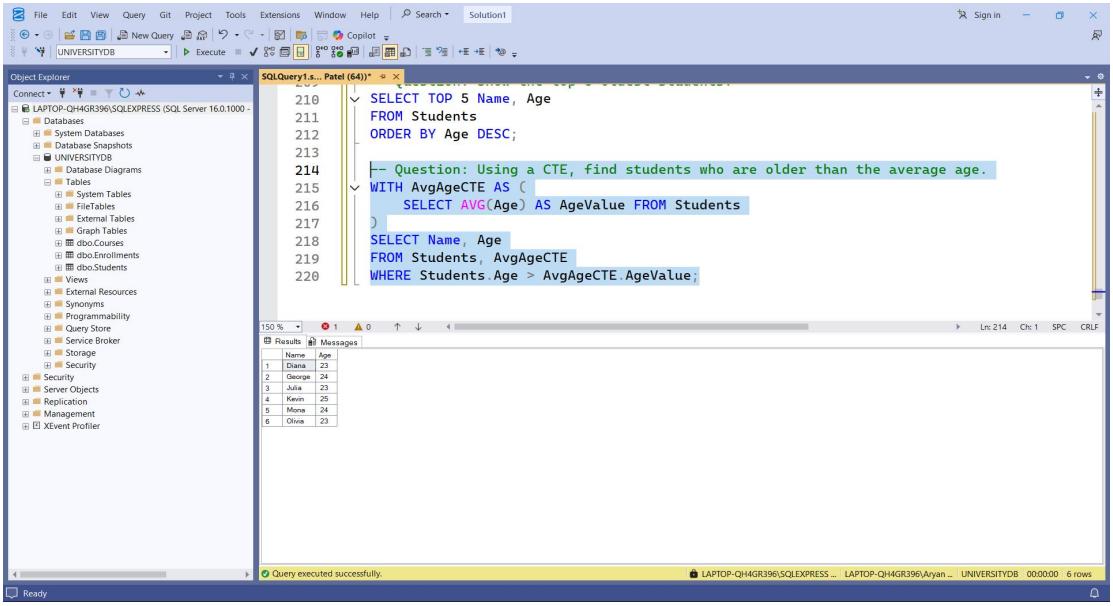

-- Question: How to perform several changes as a single atomic operation?
BEGIN TRANSACTION;
INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID, Grade) VALUES (16, 1, 104,
UPDATE Students SET Major = 'AI' WHERE StudentID = 1;
COMMIT TRANSACTION;
-- Question: How do you create an index to speed up searches on the Major column?
CREATE INDEX idx_student_major ON Students(Major);
-- Question: How can you check if the index is used?
SELECT * FROM Students WHERE Major = 'Data Science';
StudentID Name Age Major Email
1 2 Bob 22 Data Science NULL
2 4 Diana 23 Data Science NULL
3 7 George 24 Data Science NULL
4 9 Ian 21 Data Science NULL
5 11 Kevin 25 Data Science NULL
6 13 Mona 24 Data Science NULL
Query executed successfully.


BEGIN TRANSACTION;
INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID, Grade) VALUES (16, 1, 104,
UPDATE Students SET Major = 'AI' WHERE StudentID = 1;
COMMIT TRANSACTION;
-- Question: How do you create an index to speed up searches on the Major column?
CREATE INDEX idx_student_major ON Students(Major);
-- Question: How can you check if the index is used?
SELECT * FROM Students WHERE Major = 'Data Science';
-- Question: How do you remove the index?
DROP INDEX idx_student_major ON Students;
Commands completed successfully.
Completion time: 2025-09-06T23:31:38.2096472+05:30
Query executed successfully.


CREATE INDEX idx_student_major ON Students(Major);
-- Question: How can you check if the index is used?
SELECT * FROM Students WHERE Major = 'Data Science';
-- Question: How do you remove the index?
DROP INDEX idx_student_major ON Students;
-- Question: List all students, sorted by age in descending order.
SELECT Name, Age
FROM Students
ORDER BY Age DESC;
Query executed successfully.


-- Question: How do you remove the index?
DROP INDEX idx_student_major ON Students;
-- Question: List all students, sorted by age in descending order.
SELECT Name, Age
FROM Students
ORDER BY Age DESC;
-- Question: List students, sorted first by age (descending), then by name (ascending).
SELECT Name, Age
FROM Students
ORDER BY Age DESC, Name ASC;
Query executed successfully.


-- Question: List students, sorted first by age (descending), then by name (ascending).
SELECT Name, Age
FROM Students
ORDER BY Age DESC, Name ASC;
-- Question: Show the top 5 oldest students.
SELECT TOP 5 Name, Age
FROM Students
ORDER BY Age DESC;
Name Age
1 Kevin 25
2 George 24
3 Mona 24
4 Diana 23
5 Julia 23
Query executed successfully.


SELECT TOP 5 Name, Age
FROM Students
ORDER BY Age DESC;
-- Question: Using a CTE, find students who are older than the average age.
WITH AvgAgeCTE AS (
SELECT AVG(Age) AS AgeValue FROM Students
)
SELECT Name, Age
FROM Students, AvgAgeCTE
WHERE Students.Age > AvgAgeCTE.AgeValue;
Name Age
1 Diana 23
2 George 24
3 Julia 23
4 Kevin 25
5 Mona 24
6 Olivia 23
Query executed successfully.