

The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the connection to "LAPTOP-QH4GR396\SQLEXPRESS (SQL Server 16.0.1000 - master)".
- SQL Query Editor:** Title bar: "SQLQuery1.sql... Patel (64)\*".
  - Text area:

```
1 CREATE DATABASE UNIVERSITYDB;
```
  - Messages pane: "No issues found", "Commands completed successfully.", "Completion time: 2025-09-06T22:28:13.7399946+05:30".
  - Status bar: "Ln: 1 Ch: 30 TABS CRLF".
- Task List:** Shows "Query executed successfully."

The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the connection to "LAPTOP-QH4GR396\SQLEXPRESS (SQL Server 16.0.1000 - UNIVERSITYDB)".
- SQL Query Editor:** Title bar: "SQLQuery1... Patel (64)\*".
  - Text area:

```
1 CREATE DATABASE UNIVERSITYDB;
2 USE UniversityDB;
```
  - Messages pane: "No issues found", "Commands completed successfully.", "Completion time: 2025-09-06T22:29:39.8665520+05:30".
  - Status bar: "Ln: 3 Ch: 1 TABS CRLF".
- Task List:** Shows "Query executed successfully."

The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the connection to "LAPTOP-QH4GR396\SQLEXPRESS (SQL Server 16.0.1000 - UNIVERSITYDB)".
- SQL Query Editor:** Title bar: "SQLQuery1... Patel (64)\*".
  - Text area:

```
1 CREATE DATABASE UNIVERSITYDB;
2 USE UniversityDB;
3
4 CREATE TABLE Students (
5     StudentID INT PRIMARY KEY,
6     Name VARCHAR(50),
7     Age INT,
8     Major VARCHAR(50)
9 );
10
11
```
  - Messages pane: "No issues found", "Commands completed successfully.", "Completion time: 2025-09-06T22:30:30.5728419+05:30".
  - Status bar: "Ln: 5 Ch: 1 SPC CRLF".
- Task List:** Shows "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
);
```

No issues found

Commands completed successfully.

Completion time: 2025-09-06T22:31:18.7507832+05:30

Query executed successfully.

```
CREATE TABLE Courses (
    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)
);
```

No issues found

Commands completed successfully.

Completion time: 2025-09-06T22:32:24.5051577+05:30

Query executed successfully.

```
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

Query executed successfully.

```
34 (7, 'George', 24, 'Data Science'),
35 (8, 'Hannah', 22, 'Computer Science'),
36 (9, 'Ian', 21, 'Data Science'),
37 (10, 'Julia', 23, 'Computer Science'),
38 (11, 'Kevin', 25, 'Data Science'),
39 (12, 'Liam', 22, 'Computer Science'),
40 (13, 'Mona', 24, 'Data Science'),
41 (14, 'Nate', 17, 'Undeclared'),
42 (15, 'Olivia', 23, 'Physics');

SELECT * FROM Students;
```

StudentID	Name	Age	Major
1	Alice	20	Computer Science
2	Bob	22	Data Science
3	Charlie	21	Computer Science
4	Diana	23	Data Science
5	Eve	19	Mathematics
6	Fiona	20	Computer Science
7	George	24	Data Science
8	Hannah	22	Computer Science
9	Ian	21	Data Science
10	Jane	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

```
(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);
```

CourseID	CourseName	Credits
1	Database Systems	4
2	Algorithms	4
3	Machine Learning	3
4	Web Development	3

```
(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;
```

CourseID	CourseName	Credits
1	Database Systems	4
2	Algorithms	4
3	Machine Learning	3
4	Web Development	3

Object Explorer

```
54    INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID, Grade) VALUES
55    (1, 1, 101, 'A'),
56    (2, 2, 103, 'B'),
57    (3, 3, 102, 'A'),
58    (4, 4, 103, 'C'),
59    (5, 1, 102, 'B'),
60    (6, 6, 101, 'A-'),
61    (7, 7, 103, 'B+'),
62    (8, 8, 102, 'C+'),
63    (9, 9, 103, 'A'),
64    (10, 2, 101, 'B');

(10 rows affected)

Completion time: 2025-09-06T22:37:32.1849995+05:30
```

Ready

Object Explorer

```
56    (2, 2, 103, 'B'),
57    (3, 3, 102, 'A'),
58    (4, 4, 103, 'C'),
59    (5, 1, 102, 'B'),
60    (6, 6, 101, 'A-'),
61    (7, 7, 103, 'B+'),
62    (8, 8, 102, 'C+'),
63    (9, 9, 103, 'A'),
64    (10, 2, 101, 'B');

65
66    SELECT * FROM Enrollments;
```

EnrollmentID	StudentID	CourseID	Grade
1	1	101	A
2	2	103	B
3	3	102	A
4	4	103	C
5	5	102	B
6	6	101	A-
7	7	103	B+
8	8	102	C+
9	9	103	A
10	10	101	B

Query executed successfully.

Object Explorer

```
64    (10, 2, 101, 'B');

65
66    SELECT * FROM Enrollments;
67
68    =====
69    -- Executing All Provided Queries
70    =====
71
72    -- Question: How do you define a temporary table for departments?
73    CREATE TABLE Departments (
74        DeptID INT PRIMARY KEY,
75        DeptName VARCHAR(50)
76    );
```

Commands completed successfully.

Completion time: 2025-09-06T22:40:53.1882943+05:30

Ready

File Edit View Query Git Project Tools Extensions Window Help Search Solution1

Object Explorer

Connect to... UNIVERSITYDB

Databases System Databases Database Snapshots UNIVERSITYDB Database Diagrams Tables System Tables FileTables External Tables Graph Tables Courses Enrollments Students Views External Resources Synonyms Programmability Query Store Service Broker Storage Security Server Objects Replication Management XEvent Profiler

SQLQuery1.s... Patel (64)\*

```

70
71
72 -- Question: How do you define a temporary table for departments?
73 CREATE TABLE #Departments (
74     DeptID INT PRIMARY KEY,
75     DeptName VARCHAR(50)
76 );
77
78 -- Question: How can you add a new 'Email' column to the Students table?
79 ALTER TABLE Students ADD Email VARCHAR(100);
80
81 SELECT * FROM Students;

```

Results Messages

StudentID	Name	Age	Major	Email
1	Alice	20	Computer Science	NULL
2	Bob	22	Computer 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	Ivan	21	Mathematics	NULL
10	Julia	23	Computer Science	NULL
11	Kevin	25	Data Science	NULL
12	Liam	22	Computer Science	NULL
13	Mia	24	Mathematics	NULL
14	Noel	17	Undeclared	NULL
15	Olivia	23	Physics	NULL

Query executed successfully.

File Edit View Query Git Project Tools Extensions Window Help Search Solution1

Object Explorer

Connect to... UNIVERSITYDB

Databases System Databases Database Snapshots UNIVERSITYDB Database Diagrams Tables System Tables FileTables External Tables Graph Tables Courses Enrollments Students Views External Resources Synonyms Programmability Query Store Service Broker Storage Security Server Objects Replication Management XEvent Profiler

SQLQuery1.s... Patel (64)\*

```

75
76
77
78
79
80
81
82
83
84

```

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;

```

Messages

Commands completed successfully.

Completion time: 2025-09-06T23:13:55.6124042+05:30

Query executed successfully.

File Edit View Query Git Project Tools Extensions Window Help Search Solution1

Object Explorer

Connect to... UNIVERSITYDB

Databases System Databases Database Snapshots UNIVERSITYDB Database Diagrams Tables System Tables FileTables External Tables Graph Tables Courses Enrollments Students Views External Resources Synonyms Programmability Query Store Service Broker Storage Security Server Objects Replication Management XEvent Profiler

SQLQuery1.s... Patel (64)\*

```

81
82
83
84
85
86
87

```

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);

```

Messages

Commands completed successfully.

Completion time: 2025-09-06T23:15:14.3960479+05:30

Query executed successfully.

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The Object Explorer on the left lists the database structure for 'UNIVERSITYDB'. The SQL Query window on the right contains a script for managing the 'Students' table:

```
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;
```

The status bar at the bottom indicates 'Query executed successfully.' and shows connection details: 'LAPTOP-QH4GR390\SQLEXPRESS ... LAPTOP-QH4GR390\Ayan ... UNIVERSITYDB 00:00:00 0 rows'.

-- 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: 2025-09-06 23:17:06.5979158+05:30

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The Object Explorer on the left lists the database structure for 'LAPTOP-QH4GR396\SQLEXPRESS (SQL Server 16.0.1000)'. The 'UniversityDB' database is selected, showing tables like 'Departments', 'Students', 'Enrollments', and 'Courses'. The 'Results' tab in the center displays the output of a query that drops the 'Departments' table, adds a constraint to ensure student age is at least 17, removes the constraint, and updates a student's major. The 'Messages' tab shows a success message. The status bar at the bottom indicates the query was executed successfully.

```
File Edit View Query Git Project Tools Extensions Window Help Search Solution1 Sign in

Object Explorer
Connect
LAPTOP-QH4GR396\SQLEXPRESS (SQL Server 16.0.1000)
  - Databases
    - System Databases
    - Database Snapshots
  - UNIVERSITYDB
    - Database Diagrams
      - Tables
        - System Tables
        - FileTables
        - In-Memory Tables
        - Graph Tables
        - dbo.Tables
        - dbo.Courses
        - dbo.Enrollments
        - dbo.Students
      - Views
      - Extended Events
      - Resource Reservations
      - Synonyms
      - Programmability
      - Query Store
      - Service Broker
      - Storage
      - Security
  - Security
  - Server Objects
  - Replication
  - Management
  - XEvent Profiler

SQLQuery1.s... (Panel (64)) x

84
85
86
87
88
89
90
91
92
93
94
95
96

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;

Results Messages
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 Frank 24 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 Kristin 20 Mathematics NULL
12 Liam 22 Computer Science NULL
13 Mona 24 Data Science NULL
14 Nate 17 Undeclared NULL
15 Olivia 23 Physics NULL

Ln: 96 Ch: 1 SPC CRLF

Query executed successfully. LAPTOP-QH4GR396\SQLEXPRESS | LAPTOP-QH4GR396\Aryan | UNIVERSITYDB 00:00:00 15 rows
```

File Edit View Query Git Project Tools Extensions Window Help Search Solution

Object Explorer

Connect UNIVERSITYDB

SQLQuery1.s\_Patel (64)\*

```
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;
```

Completion time: 2025-09-06T23:19:39.2232677+05:30

Messages

1 row affected

Query executed successfully.

File Edit View Query Git Project Tools Extensions Window Help Search Solution

Object Explorer

Connect UNIVERSITYDB

SQLQuery1.s\_Patel (64)\*

```
-- 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;
```

Results

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	Jane	23	Computer Science	NULL
11	Kevin	25	Data Science	NULL
12	Liam	22	Computer Science	NULL
13	Mona	24	Data Science	NULL
14	Olivia	23	Physics	NULL

Query executed successfully.

File Edit View Query Git Project Tools Extensions Window Help Search Solution

Object Explorer

Connect UNIVERSITYDB

SQLQuery1.s\_Patel (64)\*

```
-- 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;
```

Results

Name	Major
Alice	Data Science
Bob	Data Science
Charlie	Computer Science
Diana	Data Science
Fiona	Computer Science
George	Computer Science
Hannah	Computer Science
Ian	Data Science
Jane	Computer Science
Kevin	Data Science
Liam	Computer Science
Mona	Data Science
Olivia	Physics

Query executed successfully.

Object Explorer

```

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;

```

Results

AvgAge
22

Query executed successfully.

Object Explorer

```

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;

```

Results

Major	StudentCount
Data Science	7

Query executed successfully.

Object Explorer

```

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';

```

Results

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.

Object Explorer

```
-- 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	2
Ian	A	3
Fiona	A-	4
Bob	B	5
Bob	B	6
Alice	B	7
George	B+	8
Diana	C	9
Hannah	C+	10

Query executed successfully.

Object Explorer

```
-- Question: List student names and the courses they are enrolled in.
SELECT s.Name, e.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.

Object Explorer

```
-- Question: Show all students and their courses, including students not enrolled in any course.
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 course.
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
Fiona	Database Systems
George	Machine Learning
Hannah	Algorithms
Ian	Machine Learning
Julia	NULL
Julia	NULL
Kevin	NULL
Liam	NULL
Mona	NULL
Olivia	NULL

Query executed successfully.

File Edit View Query Git Project Tools Extensions Window Help | Search | Solution1

Object Explorer

LAPTOP-QH4GR396\SQLEXPRESS (SQL Server 16.0.1000 - UNIVERSITYDB)

SQLQuery1... Patel (64)\*

```

127     INNER JOIN Enrollments e ON s.StudentID = e.StudentID
128     INNER JOIN Courses c ON e.CourseID = c.CourseID;
129
130     -- Question: Show all students and their courses, including students not enrolled in any
131     <-- SELECT s.Name, c.CourseName
132     FROM Students s
133     LEFT JOIN Enrollments e ON s.StudentID = e.StudentID
134     LEFT JOIN Courses c ON e.CourseID = c.CourseID;
135
136     -- Question: Generate every possible combination of a student paired with a course.
137     <-- SELECT s.Name, c.CourseName
138     FROM Students s CROSS JOIN Courses c;
139

```

Results

Name	CourseName
Alice	Database Systems
Bob	Database Systems
Charlie	Database Systems
Diana	Database Systems
Ethan	Database Systems
Fiona	Database Systems
George	Database Systems
Hannah	Database Systems
Ian	Database Systems
Julia	Database Systems
Karen	Database Systems
Liam	Database Systems
Mona	Database Systems
Olivia	Database Systems
Alice	Algorithms
Bob	Algorithms
Charlie	Algorithms
Diana	Algorithms

Query executed successfully.

File Edit View Query Git Project Tools Extensions Window Help | Search | Solution1

Object Explorer

LAPTOP-QH4GR396\SQLEXPRESS (SQL Server 16.0.1000 - UNIVERSITYDB)

SQLQuery1... Patel (64)\*

```

131     <-- SELECT s.Name, c.CourseName
132     FROM Students s
133     LEFT JOIN Enrollments e ON s.StudentID = e.StudentID
134     LEFT JOIN Courses c ON e.CourseID = c.CourseID;
135
136     -- Question: Generate every possible combination of a student paired with a course.
137     <-- SELECT s.Name, c.CourseName
138     FROM Students s CROSS JOIN Courses c;
139
140     -- Question: Find pairs of different students who are in the same major.
141     <-- SELECT s1.Name AS Student1, s2.Name AS Student2
142     FROM Students s1
143     JOIN Students s2 ON s1.Major = s2.Major AND s1.StudentID <> s2.StudentID;
144

```

Results

Student1	Student2
Alice	Bob
Alice	Diana
Alice	George
Alice	Ian
Alice	Julia
Alice	Mona
Alice	Olivia
Bob	George
Bob	Ian
Bob	Julia
Bob	Mona
Bob	Olivia
Charlie	Fiona
Charlie	George
Charlie	Ian
Charlie	Julia
Charlie	Liam
Diana	Alice
Diana	Bob

Query executed successfully.

File Edit View Query Git Project Tools Extensions Window Help | Search | Solution1

Object Explorer

LAPTOP-QH4GR396\SQLEXPRESS (SQL Server 16.0.1000 - UNIVERSITYDB)

SQLQuery1... Patel (64)\*

```

138     <-- SELECT s1.Name AS Student1, s2.Name AS Student2
139     FROM Students s CROSS JOIN Courses c;
140
141     -- Question: Find pairs of different students who are in the same major.
142     <-- SELECT s1.Name AS Student1, s2.Name AS Student2
143     FROM Students s1
144     JOIN Students s2 ON s1.Major = s2.Major AND s1.StudentID <> s2.StudentID;
145
146     -- Question: For each major, provide a comma-separated list of student names.
147     <-- SELECT Major, STRING_AGG(Name, ', ') AS Students
148     FROM Students
149     GROUP BY Major;

```

Results

Major	Students
Computer Science	Charlie, Fiona, Hannah, Julia, Liam
Data Science	Mona, Kevin, Ian, George, Diana, Alice, Bob
Mathematics	Ethan
Physics	Olivia

Query executed successfully.

Object Explorer

```

140 -- Question: Find pairs of different students who are in the same major.
141
142 SELECT s1.Name AS Student1, s2.Name AS Student2
143 FROM Students s1
144 JOIN Students s2 ON s1.Major = s2.Major AND s1.StudentID <> s2.StudentID;
145
146 -- Question: For each major, provide a comma-separated list of student names.
147 SELECT Major, STRING_AGG(Name, ', ') AS Students
148 FROM Students
149 GROUP BY Major;
150
151 -- Question: Find students who are older than the average student age.
152
153 SELECT Name FROM Students
154 WHERE Age > (SELECT AVG(Age) FROM Students);

```

Results

Name
David
George
Julia
Kevin
Mona
Olivia

Query executed successfully.

Object Explorer

```

147
148
149
150
151
152
153
154
155
156
157
158
159

```

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'

Results

Name
Alice
Charlie
Ian

Query executed successfully.

Object Explorer

```

150
151
152
153
154
155
156
157
158
159
160
161
162
163

```

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;

Results

Major	AvgAge
Computer Science	21
Data Science	22
Mathematics	19
Physics	23

Query executed successfully.

Object Explorer

```

156 WHERE EXISTS (
157     SELECT 1 FROM Enrollments e
158     WHERE e.StudentID = s.StudentID AND e.Grade = 'A'
159 );
160
161 -- Question: Display each major and its calculated average age.
162 SELECT Major, AvgAge
163 FROM (SELECT Major, AVG(Age) AS AvgAge FROM Students GROUP BY Major) AS t;
164
165 -- Question: Create a single list of all student names and course names.
166
167 -- Question: Find student IDs present in both Students and Enrollments tables.
168
169 UNION
170
171
172
173

```

Results

Name	
1	Algorithm
2	Alice
3	Bab
4	Charlie
5	Diana
6	Ethan
7	Fiona
8	Gina
9	Hannah
10	Ian
11	Juli
12	Liam
13	Mia
14	Olivia
15	Machine Learning
16	Mona
17	Web Development

Query executed successfully.

Object Explorer

```

161 -- Question: Display each major and its calculated average age.
162
163 -- Question: Create a single list of all student names and course names.
164
165
166 -- Question: Find student IDs present in both Students and Enrollments tables.
167
168
169
170
171
172
173

```

Results

StudentID
1
2
3
4
5
6
7
8

Query executed successfully.

Object Explorer

```

169
170
171
172
173
174
175
176
177
178
179
180
181

```

Results

Name	
1	Ethan
2	Julia
3	Kevin
4	Liam
5	Mona
6	Olivia

Query executed successfully.

File Edit View Query Git Project Tools Extensions Window Help Search Solution

UNIVERSITYDB Execute

Object Explorer

SQLQuery1.s\_ Patel (64)\*

```

175 -- Question: Find students registered but not enrolled in any course.
176 SELECT Name FROM Students
177 WHERE StudentID IN (
178     SELECT StudentID FROM Students
179     EXCEPT
180     SELECT StudentID FROM Enrollments
181 );
182
183 -- Question: How to perform several changes as a single atomic operation?
184 BEGIN TRANSACTION;
185 INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID, Grade) VALUES (16, 1, 104,
186 UPDATE Students SET Major = 'AI' WHERE StudentID = 1;
187 COMMIT TRANSACTION;

```

(1 row affected)

(1 row affected)

Completion time: 2025-09-06T23:30:16.5925724+05:30

No issues found

Query executed successfully.

Ln: 183 Ch: 1 TABS MIXED

Ready

File Edit View Query Git Project Tools Extensions Window Help Search Solution

UNIVERSITYDB Execute

Object Explorer

SQLQuery1.s\_ Patel (64)\*

```

178 SELECT StudentID FROM Students
179 EXCEPT
180 SELECT StudentID FROM Enrollments
181 ;
182
183 -- Question: How to perform several changes as a single atomic operation?
184 BEGIN TRANSACTION;
185 INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID, Grade) VALUES (16, 1, 104,
186 UPDATE Students SET Major = 'AI' WHERE StudentID = 1;
187 COMMIT TRANSACTION;
188
189 -- Question: How do you create an index to speed up searches on the Major column?
190 CREATE INDEX idx_student_major ON Students(Major);

```

Commands completed successfully.

Completion time: 2025-09-06T23:30:43.5233400+05:30

No issues found

Query executed successfully.

Ln: 190 Ch: 1 TABS MIXED

Ready

File Edit View Query Git Project Tools Extensions Window Help Search Solution

UNIVERSITYDB Execute

Object Explorer

SQLQuery1.s\_ Patel (64)\*

```

182 -- Question: How to perform several changes as a single atomic operation?
183 BEGIN TRANSACTION;
184 INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID, Grade) VALUES (16, 1, 104,
185 UPDATE Students SET Major = 'AI' WHERE StudentID = 1;
186 COMMIT TRANSACTION;
187
188 -- Question: How do you create an index to speed up searches on the Major column?
189 CREATE INDEX idx_student_major ON Students(Major);
190
191 -- Question: How can you check if the index is used?
192 SELECT * FROM Students WHERE Major = 'Data Science';
193

```

StudentID	Name	Age	Major	Email
1	Bob	22	Data Science	NULL
2	Diana	23	Data Science	NULL
3	George	24	Data Science	NULL
4	Ian	21	Data Science	NULL
5	Kevin	25	Data Science	NULL
6	Mona	24	Data Science	NULL

No issues found

Query executed successfully.

Ln: 193 Ch: 1 TABS MIXED

Ready

File Edit View Query Git Project Tools Extensions Window Help Search Solution

UNIVERSITYDB Execute

Object Explorer

LAPTOP-QH4GR396\SQLEXPRESS (SQL Server 16.0.1000-)

Databases

UNIVERSITYDB

Tables

Views

Security

Object Explorer

SQLQuery1.s... Patel (64)\*

```
184 BEGIN TRANSACTION;
185 INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID, Grade) VALUES (16, 1, 104, 'A');
186 UPDATE Students SET Major = 'AI' WHERE StudentID = 1;
187 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;
```

Messages

Commands completed successfully.

Completion time: 2025-09-06T23:31:38.2096472+05:30

Query executed successfully.

Ln: 195 Ch: 1 SPC CRLF

File Edit View Query Git Project Tools Extensions Window Help Search Solution

UNIVERSITYDB Execute

Object Explorer

LAPTOP-QH4GR396\SQLEXPRESS (SQL Server 16.0.1000-)

Databases

UNIVERSITYDB

Tables

Views

Security

Object Explorer

SQLQuery1.s... Patel (64)\*

```
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;
```

Results

Name	Age
Kevin	25
Mona	24
George	24
Diana	23
Olivia	23
Julia	23
Liam	22
Bob	22
Isabella	22
Ian	21
Charlie	21
Alice	20
Fiona	20
Ethan	19

Messages

Query executed successfully.

Ln: 199 Ch: 1 SPC CRLF

File Edit View Query Git Project Tools Extensions Window Help Search Solution

UNIVERSITYDB Execute

Object Explorer

LAPTOP-QH4GR396\SQLEXPRESS (SQL Server 16.0.1000-)

Databases

UNIVERSITYDB

Tables

Views

Security

Object Explorer

SQLQuery1.s... Patel (64)\*

```
-- 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;
```

Results

Name	Age
Kevin	25
George	24
Isabella	24
Diana	23
Julia	23
Olivia	23
Bob	22
Isabella	22
Liam	22
Charlie	21
Ian	21
Alice	20
Fiona	20
Ethan	19

Messages

Query executed successfully.

Ln: 204 Ch: 1 SPC CRLF

File Edit View Query Git Project Tools Extensions Window Help Search Solution1 Sign in

UNIVERSITYDB Execute

Object Explorer

SQLQuery1.s... Patel (64)\*

```
203 -- Question: List students, sorted first by age (descending), then by name (ascending).
204 SELECT Name, Age
205 FROM Students
206 ORDER BY Age DESC, Name ASC;
207
208 -- Question: Show the top 5 oldest students.
209 SELECT TOP 5 Name, Age
210 FROM Students
211 ORDER BY Age DESC;
```

Results Messages

Name	Age
Kevin	25
George	24
Mona	24
Diana	23
Julia	23

Ln: 209 Ch: 1 SPC CRLF

Query executed successfully.

LAPTOP-QH4GR396\SQLEXPRESS ... LAPTOP-QH4GR396\Aryan ... UNIVERSITYDB 00:00:00 5 rows

Ready

File Edit View Query Git Project Tools Extensions Window Help Search Solution1 Sign in

UNIVERSITYDB Execute

Object Explorer

SQLQuery1.s... Patel (64)\*

```
210 SELECT TOP 5 Name, Age
211 FROM Students
212 ORDER BY Age DESC;
213
214 -- Question: Using a CTE, find students who are older than the average age.
215 WITH AvgAgeCTE AS (
216     SELECT AVG(Age) AS AgeValue FROM Students
217 )
218     SELECT Name, Age
219     FROM Students, AvgAgeCTE
220     WHERE Students.Age > AvgAgeCTE.AgeValue;
```

Results Messages

Name	Age
Diana	23
George	24
Julia	23
Kevin	25
Mona	24
Olivia	23

Ln: 214 Ch: 1 SPC CRLF

Query executed successfully.

LAPTOP-QH4GR396\SQLEXPRESS ... LAPTOP-QH4GR396\Aryan ... UNIVERSITYDB 00:00:00 6 rows

Ready