Mini Project: Student Management System

SECTION I:

Write SQL queries to retrieve the following information:

a. Retrieve the list of all students.

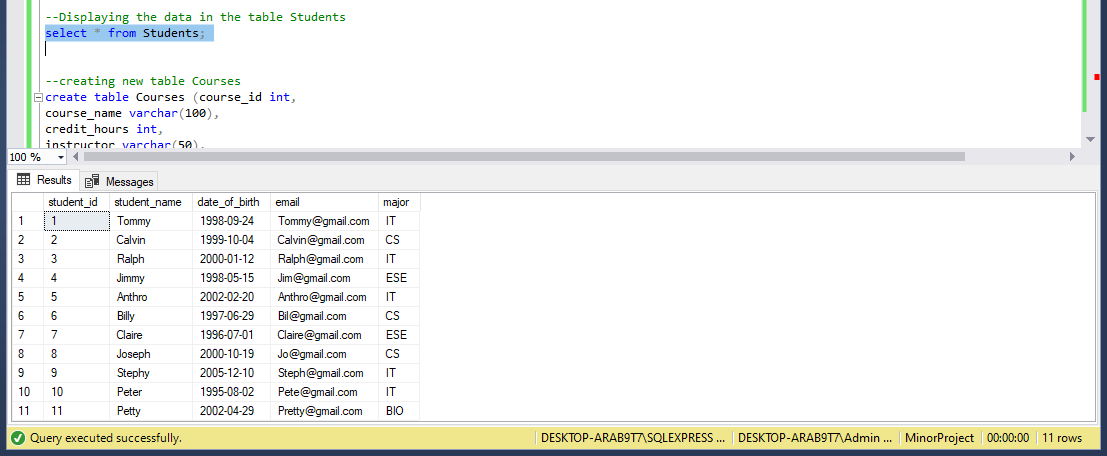
Explanation: Tis query returns all the data records that are stored in the Students table.

Expected output: The query written should return all the student entries available in the Students table created.

Query:

select \* from Students;

Actual output: All the entries made in the Students table are displayed in the output.



b. Retrieve the list of all courses.

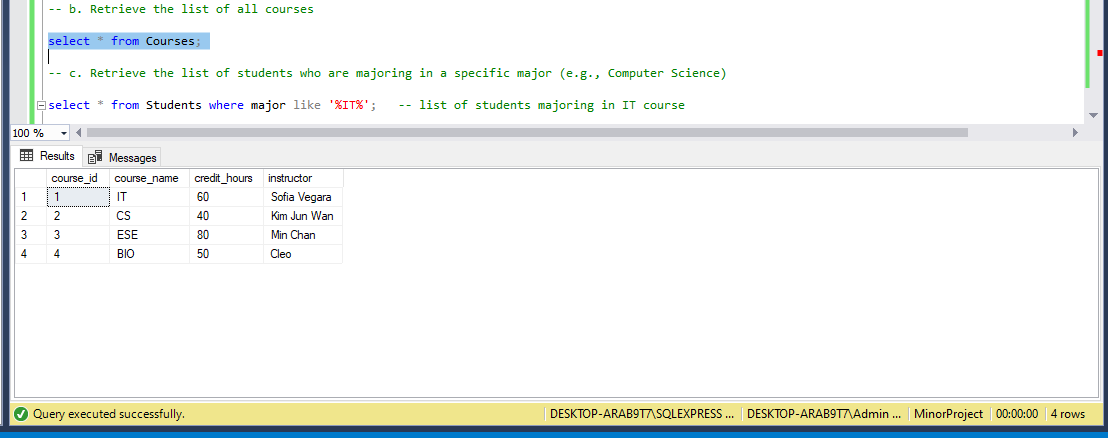
Explanation: This query returns all the data records that are stored in the Courses table.

Expected output: The query written should return all the course entries available in the Courses table created.

Query:

select \* from Courses;

Actual output: All the entries made in the Courses table are displayed in the output.



c. Retrieve the list of students who are majoring in a specific major (e.g., Computer Science).

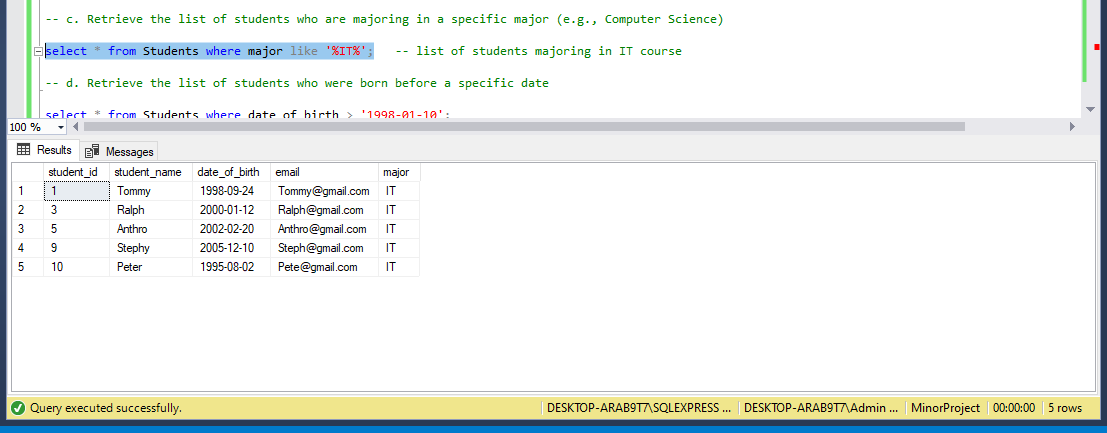
Explanation: The above query requires to retrieve the list of student data for a particular major.

Expected Output: As stated in the question, the list of students majoring in Computer Science should be displayed in the output.

Query:

select \* from Students where major like '%IT%';

Actual output: List of students majoring in the “IT” field is retrieved from the Students table.



d. Retrieve the list of students who were born before a specific date.

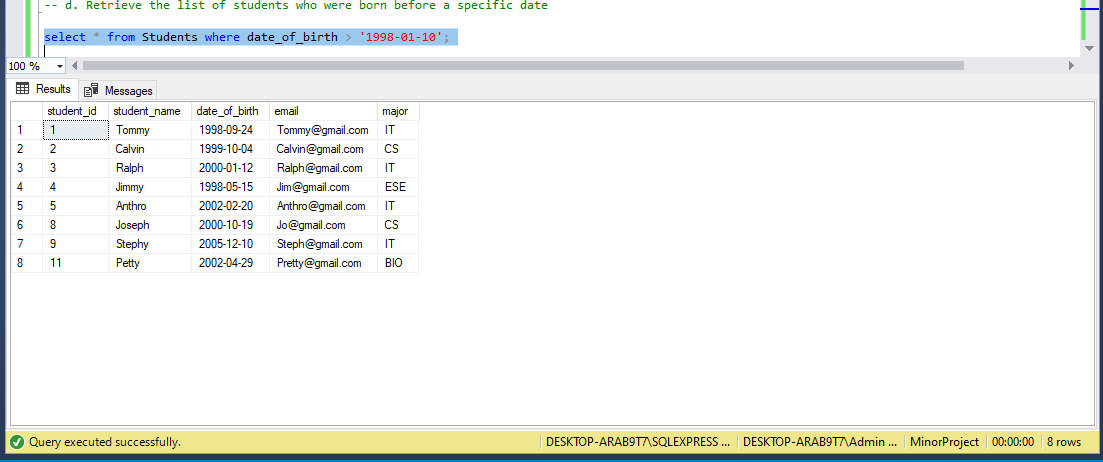
Explanation: This query retrieves the list of students born before a specific date from the Students table using the where clause.

Expected Output: List of students born before a specific date will be displayed as output.

Query:

select \* from Students where date\_of\_birth > '1998-01-10';

Actual Output: All the students born before the date ‘1998-01-10’ are displayed in the output.



e. Retrieve the list of courses taught by a specific instructor.

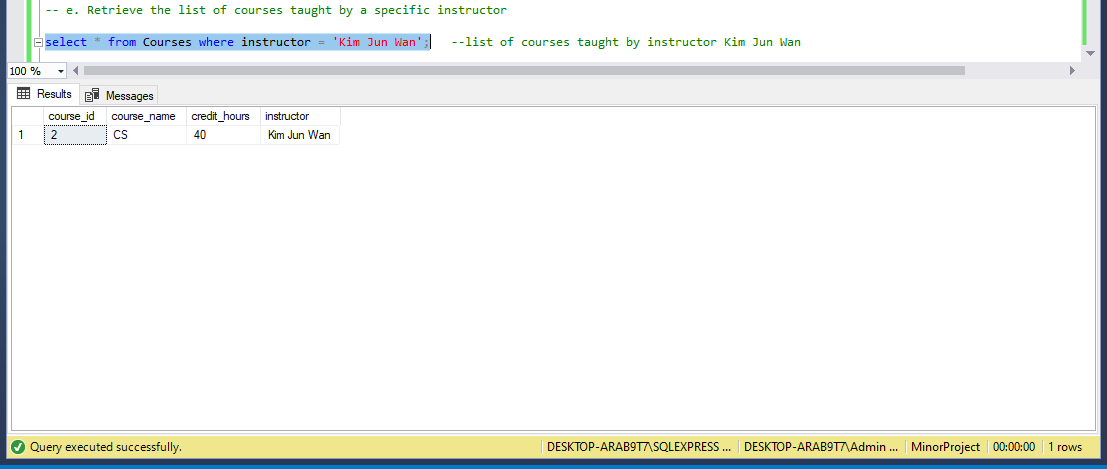
Explanation: To retrieve the list of courses taught by a particular instructor from the Courses table.

Expected Output: Courses taught by the instructor mentioned in the query will be displayed in the output screen.

Query:

select \* from Courses where instructor = 'Kim Jun Wan';

Actual Output: The list of courses taught by instructor ‘Kim Jun Wan’ are displayed on the screen.



f. Retrieve the total number of students enrolled in each major.

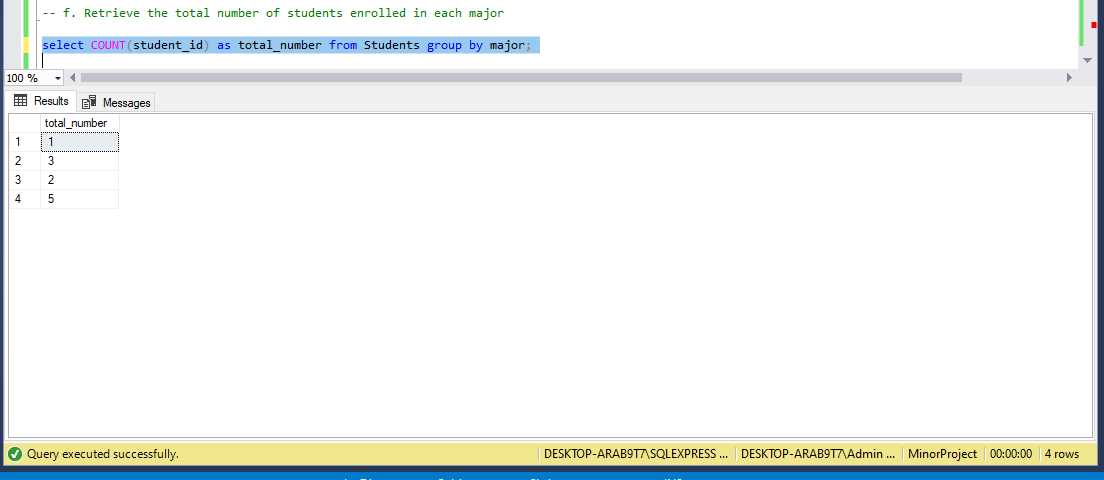
Explanation: This query will calculate the total number of students enrolled for each major available.

Expected Output: Displays the major id with the number of students enrolled for each major.

Query:

select COUNT(student\_id) as total\_number from Students group by major;

Actual Output: Course id and the total number of students per major are displayed on the screen.



g. Retrieve the course with the highest number of credit hours.

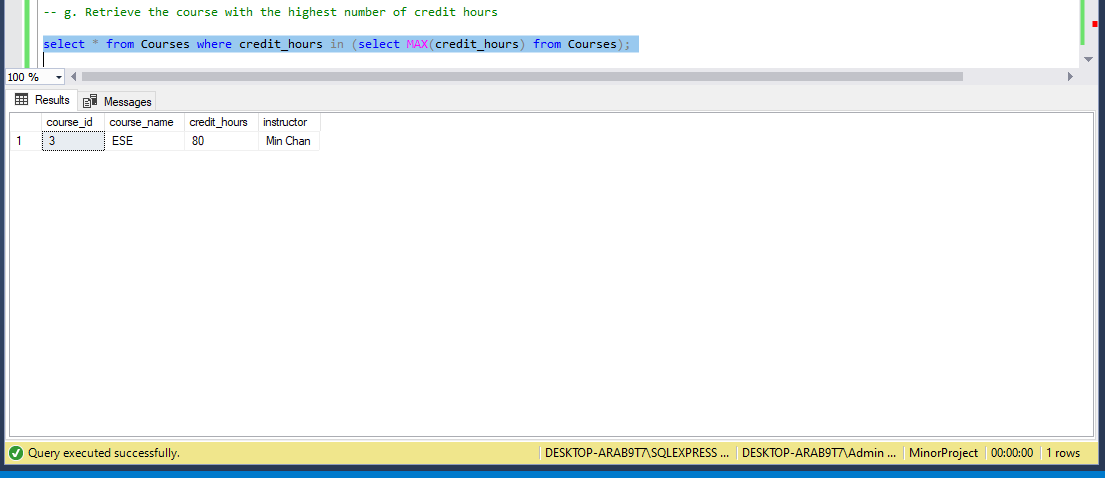
Explanation: Retrieves the course with the maximum number of credit hours assigned to it.

Expected Output: Course with maximum number of credit hours is displayed with it’s course id, credit hours, course name and instructor.

Query:

select \* from Courses where credit\_hours in (select MAX(credit\_hours) from Courses);

Actual Output: Records for the course with highest credit hours are displayed on the output screen.



h. Retrieve the oldest and youngest students in the database.

Explanation: Gets the records of the oldest and the youngest student from the Students table based on the date\_of\_birth.

Expected Output: Displays the record of the students based on their date of birth.

Query:

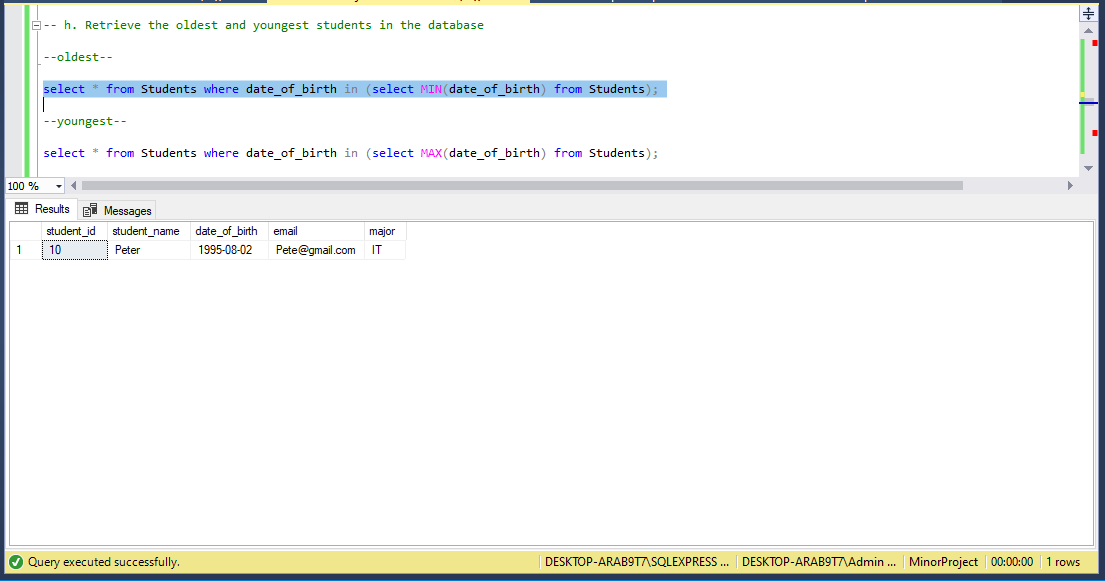
Oldest student:

select \* from Students where date\_of\_birth in (select MIN(date\_of\_birth) from Students);

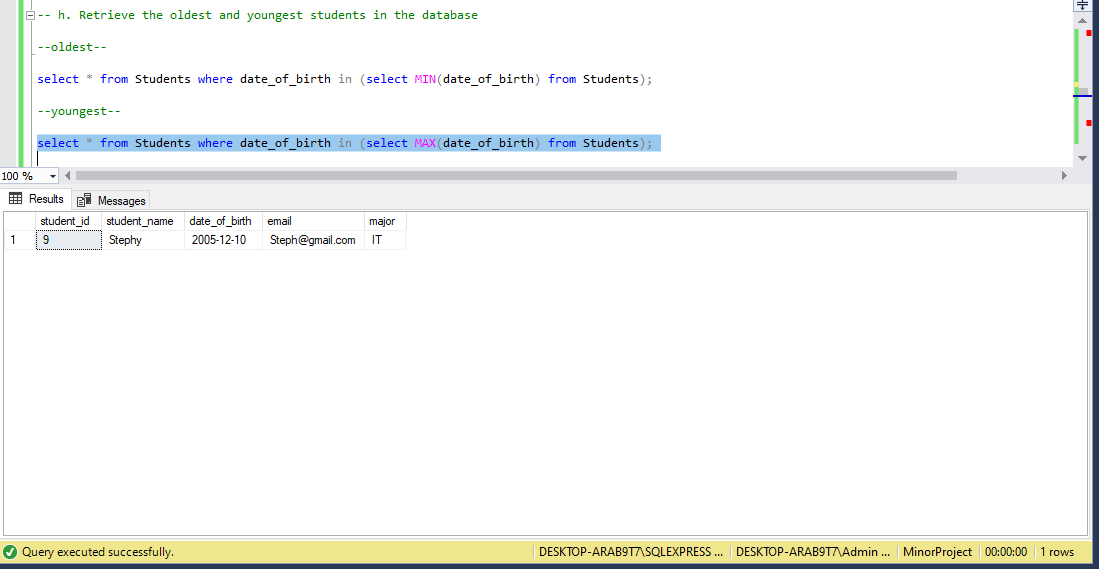
Youngest student:

select \* from Students where date\_of\_birth in (select MAX(date\_of\_birth) from Students);

Oldest student:



Youngest student:



SECTION II:

Write SQL queries to update the database:

a. Add a new course to the Courses table.

Explanation: This query will use the update query to add a new course record to the Courses table.

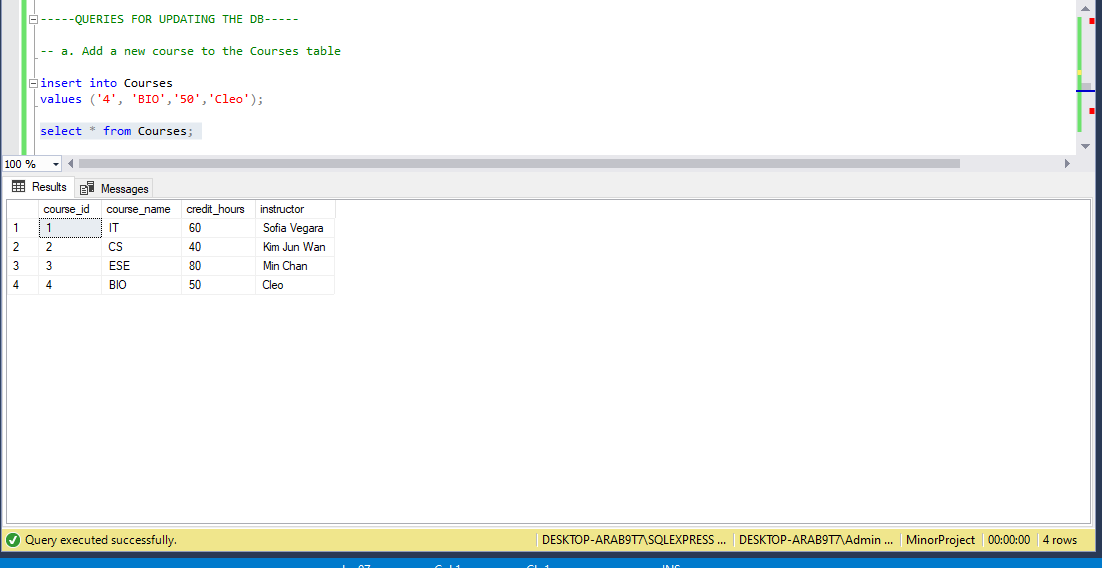
Expected Output: A new course should be added to the table Courses along with all the other column values for that particular record.

Query:

insert into Courses

values ('4', 'BIO','50','Cleo');

Actual Output: Inserts the new course record into the Courses table with values and displays all the courses now available in the table.



b. Enroll a new student in the Students table.

Explanation: To add a new student enrolment record to the Students table.

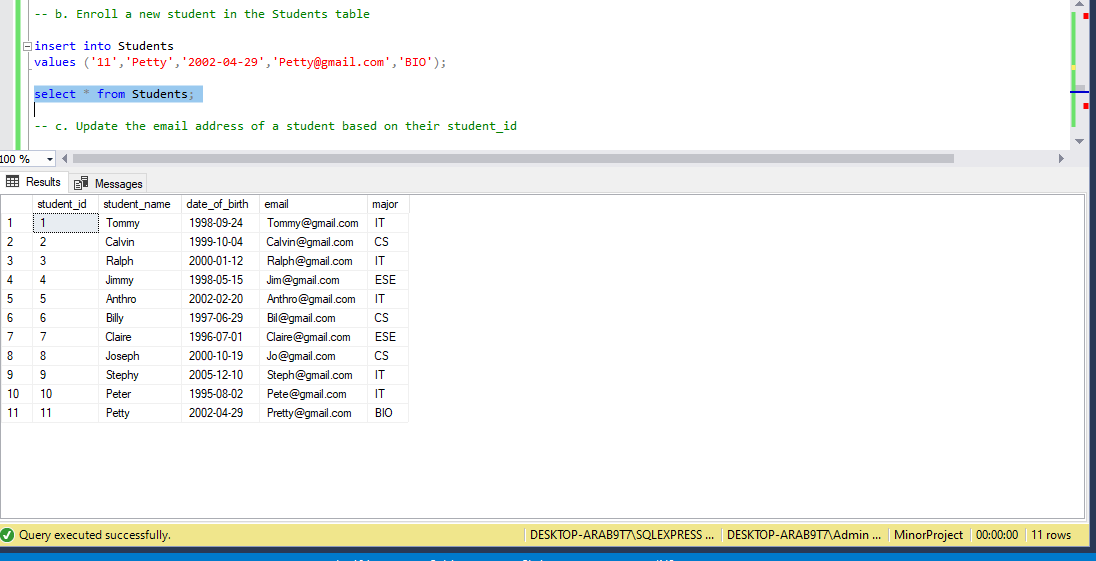
Expected Output: Inserts the new student details into the Students table successfully.

Query:

insert into Students

values ('11','Petty','2002-04-29','Petty@gmail.com','BIO');

Actual Output: Inserts the new student’s record with all the values according to the table columns and display them as the output.



c. Update the email address of a student based on their student\_id.

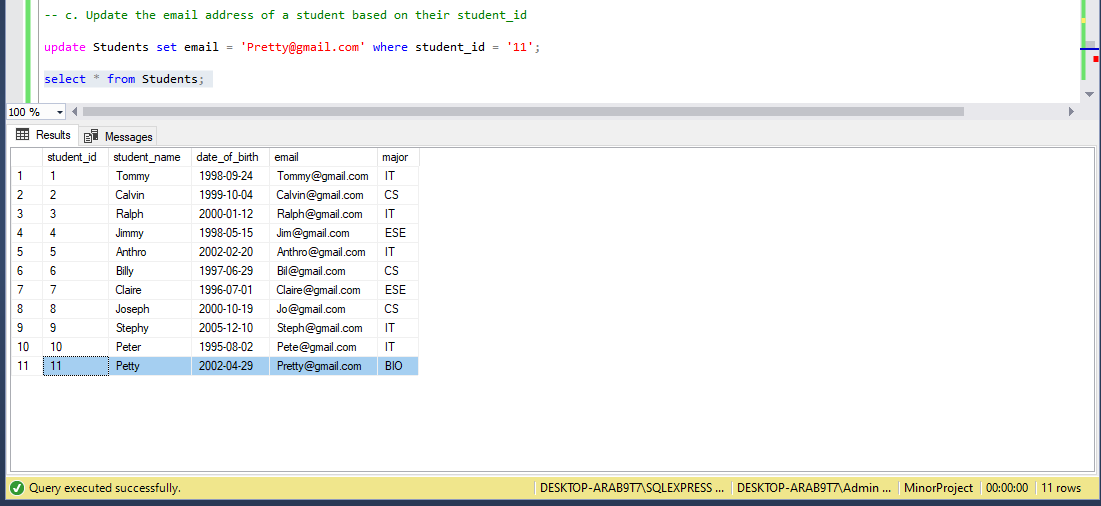
Explanation: To modify the email address of a particular student based on their student id.

Expected Output: Updates the student email id depending on their student id.

Query:

update Students set email = 'Pretty@gmail.com' where student\_id = '11';

Actual Output: Modifies the email id based on the student id of a student and display’s the changes.



SECTION III:

Write SQL queries with nested queries:

a. Retrieve the list of students who are enrolled in a specific course.

Explanation: To get the list of students enrolled for a particular course.

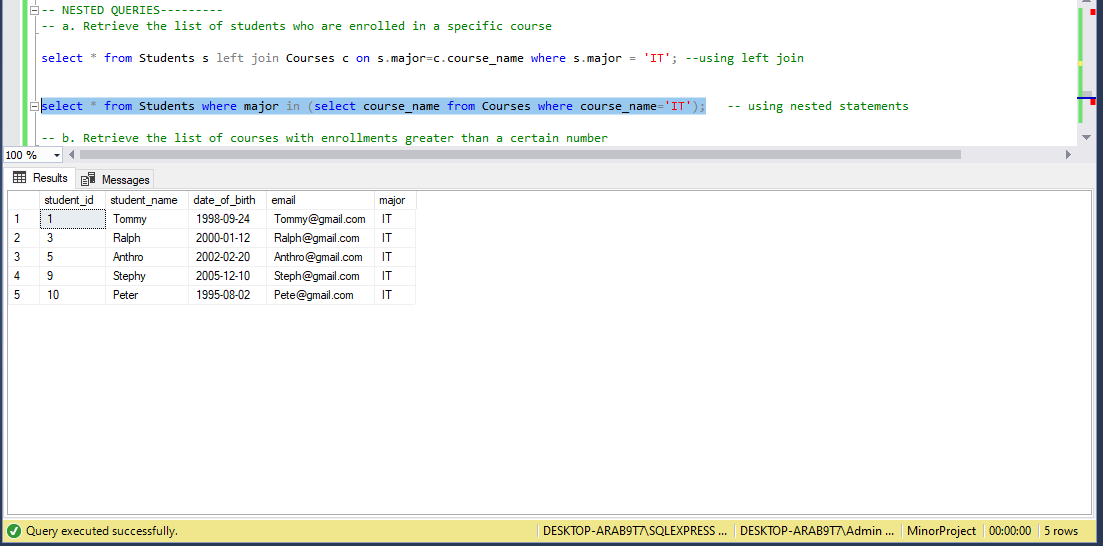
Expected Output: Retrieves the list of students enrolled in a particular course from Students and Courses tables.

Query:

select \* from Students s left join Courses c on s.major=c.course\_name where s.major = 'IT';

select \* from Students where major in (select course\_name from Courses where course\_name='IT');

Actual Output: Returns list of students who are enrolled for “IT” major.



b. Retrieve the list of courses with enrollments greater than a certain number.

Explanation: To get the list of courses with a greater number of enrollments than a specified number.

Expected Output: Display the list of courses having more enrollments than the specified number in the query.

Query:

SELECT c.course\_id, c.course\_name AS enrollment\_count

FROM Courses c

JOIN StudentCourses sc ON c.course\_id = sc.course\_id

GROUP BY c.course\_id, c.course\_name

HAVING COUNT(sc.student\_id) >=1;

Actual output: We need a table where we can store student-course relationship. So, we created another table StudentCourses with the columns student\_id and course\_id. This query joins the Courses and StudentCourses tables using the JOIN operation. It then groups the results by course\_id and course\_name, and counts the number of students enrolled in each course using the COUNT function. The HAVING clause is used to filter the results and only include courses with enrollments greater than the specified number.

