Laboratory Report

# Laboratory Report

|  |  |
| --- | --- |
| Common Data | |
| Student name | **Kormoua Khongmeng** |
| Neptun code | **I3MLPQ** |
| Department | **Dept. of Automation and Applied Informatics** |
| Instructor name | **AL-Magsoosi Husam Kareem Farhan** |
| Laboratory place | **BME IL206** |
| Laboratory time | **10:15 – 12:00** |
| Title or Sequence number | **3** |

|  |  |
| --- | --- |
| Exercises | |
| Task 1 |  |
| Task 2 |  |
| Task 3 |  |
| Task 4 |  |
| Task 5 |  |
| Task 6 |  |
| Task 7 |  |
| Task 8 |  |
| Task 9 |  |
| Task 10 |  |

# Exercises

### Task #1

**Problem statement:** Find the names of all Juniors (level = JR) who are enrolled in a class taught by I. Teach

**Solution:**

SELECT sname

FROM Student

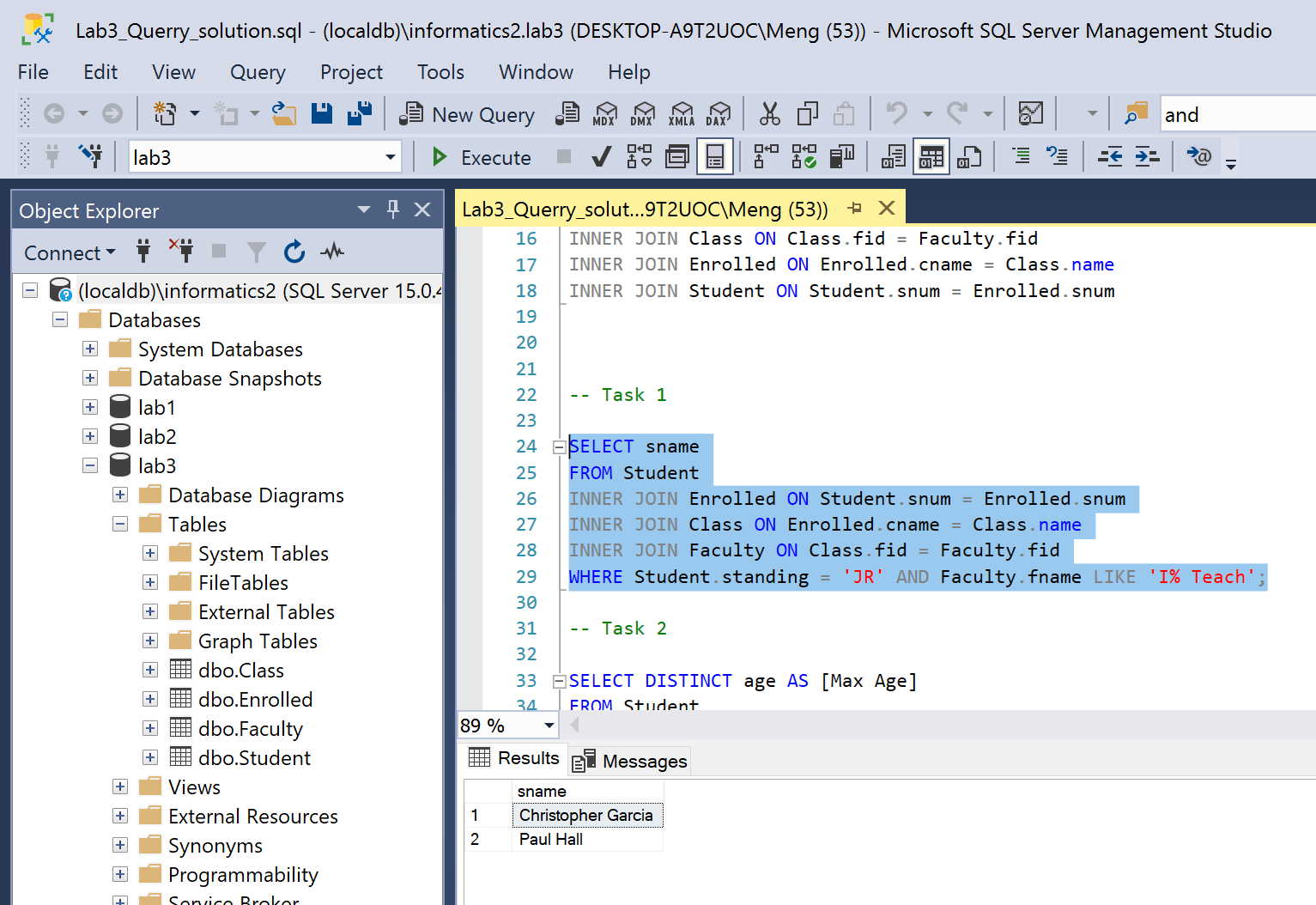
INNER JOIN Enrolled ON Student.snum = Enrolled.snum

INNER JOIN Class ON Enrolled.cname = Class.name

INNER JOIN Faculty ON Class.fid = Faculty.fid

WHERE Student.standing = 'JR' AND Faculty.fname LIKE 'I% Teach';

**Visual result:**



**Reasoning:**

We want a name of students, so I begin from selecting a name of student from student table. However, I needed to INNER JOIN student table to Enrolled table in order to know what class each student registered in then another INNER JOIN to Class table in order to know which faculty is responsible for each Class. Finally, INNER JOIN with Faculty table to be able to know the name from Faculty. Of course, every time that we do INNER JOIN we need to join on foreign key of one table to a primary Key of another table. At this point all the table are joined we can simply check a condition by WHERE to get a student who registered a class that is taught by faculty name I. Teach.

### Task #2

**Problem statement:** Find the age of the oldest student who is either a History major or enrolled in a course taught by I. Teach. Rename the resulting column to “Max Age”. As the 'fname' column of Faculty contains full names, you should take it into account when designing your query

**Solution:**

SELECT DISTINCT age AS [Max Age]

FROM Student

WHERE Student.age = (

SELECT MAX(Student.age)

FROM Student

INNER JOIN Enrolled ON Student.snum = Enrolled.snum

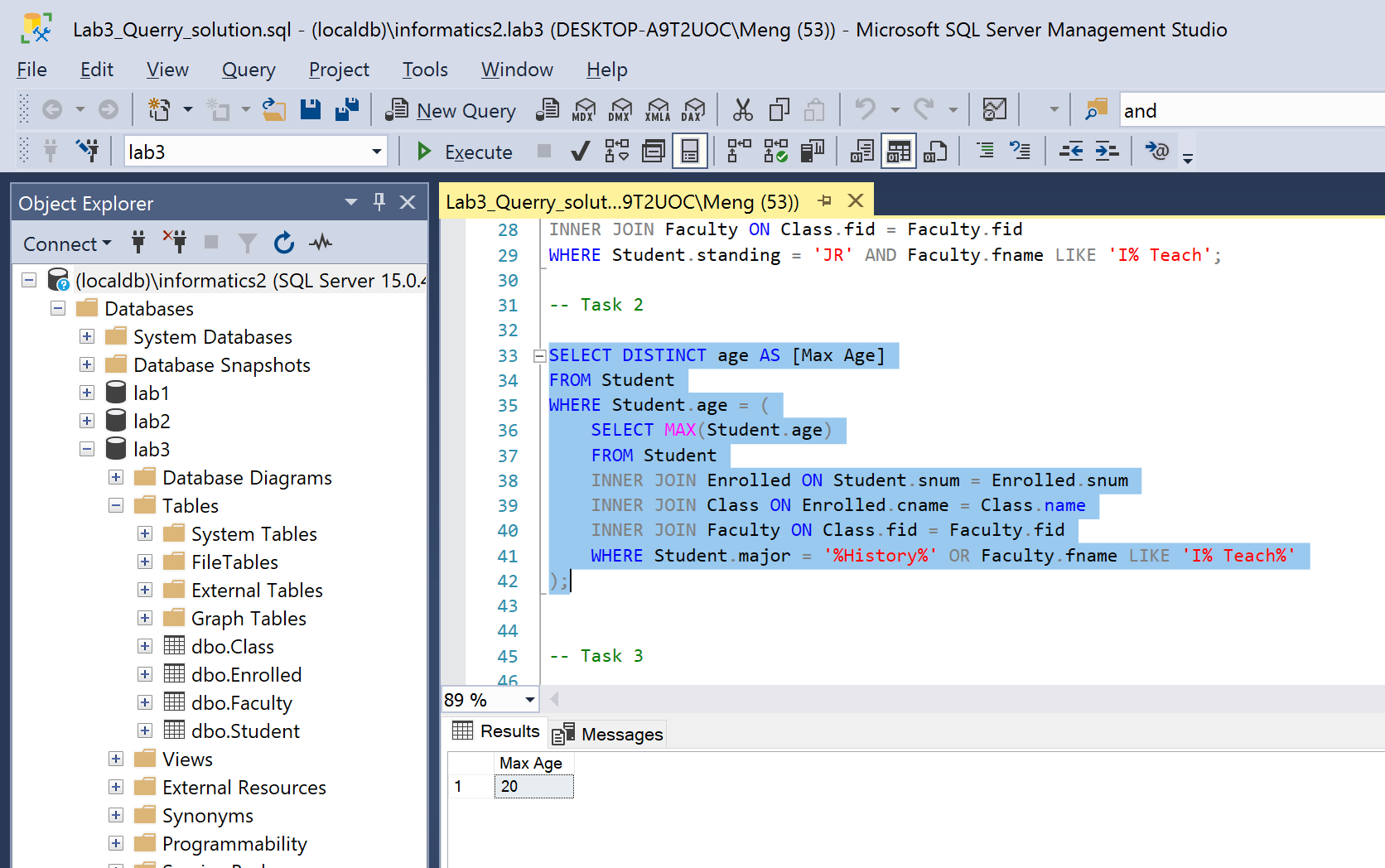
INNER JOIN Class ON Enrolled.cname = Class.name

INNER JOIN Faculty ON Class.fid = Faculty.fid

WHERE Student.major = '%History%' OR Faculty.fname LIKE 'I% Teach%'

);

**Visual result:**



**Reasoning:**

What I like to do in my solution is first simply select what we want to display to the result. In our case in just a column age from student table. Now we can check further who is going to be a good answer for us. We will use a nested select to find and check almost everything that we want then give the age back to the first select which will just show an expected result to the user. In the nested select max(student.age) so in the age column we can get the oldest student. Since we need to know the information of student which is outside student table like faculty that a student is studying with (I. Teach.) so we need to do inner join until we get to a table that we are able to search for this information which is Faculty table. Now we can do a simple condition check based on what we want which is WHERE Student.major = '%History%' OR Faculty.fname LIKE 'I% Teach%'.

### Task #3

**Problem statement:** Find the names of all classes that either meet in room R128 or have five or more students enrolled.

**Solution:**

SELECT name

FROM Class

WHERE Class.room = 'R128' OR Class.name IN (

SELECT name

FROM Class

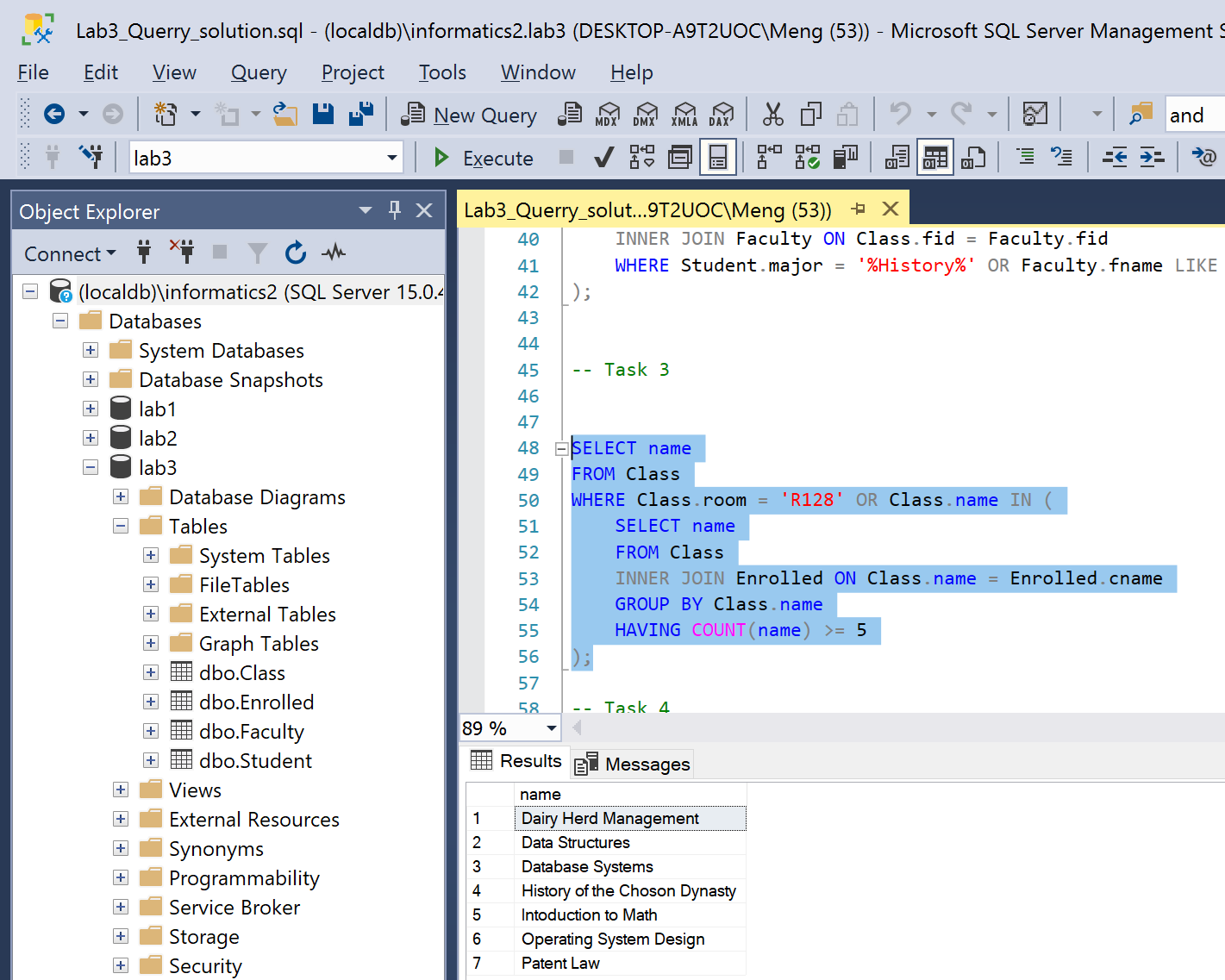
INNER JOIN Enrolled ON Class.name = Enrolled.cname

GROUP BY Class.name

HAVING COUNT(name) >= 5

);

**Visual result:**



**Reasoning:**

This task we want to show a class so we will start from selecting from class table. First condition can be found easily by simple check like WHERE Class.room = 'R128'. However, for second condition we need to know a number of registered students in each class so we need to do INNER join with Enrolled table. We can then group by the name of the class and check COUNT number student > 5. If yes then we will return this class back from to the first select.

### Task #4

**Problem statement:** Find the names of all students who are enrolled in two different classes that meet at the same time. The resulting set should be empty. Usually, it is not allowed to enroll in two parallel classes, hence you will be unable to attend both at the same time.

**Solution:**

SELECT sname

FROM Student

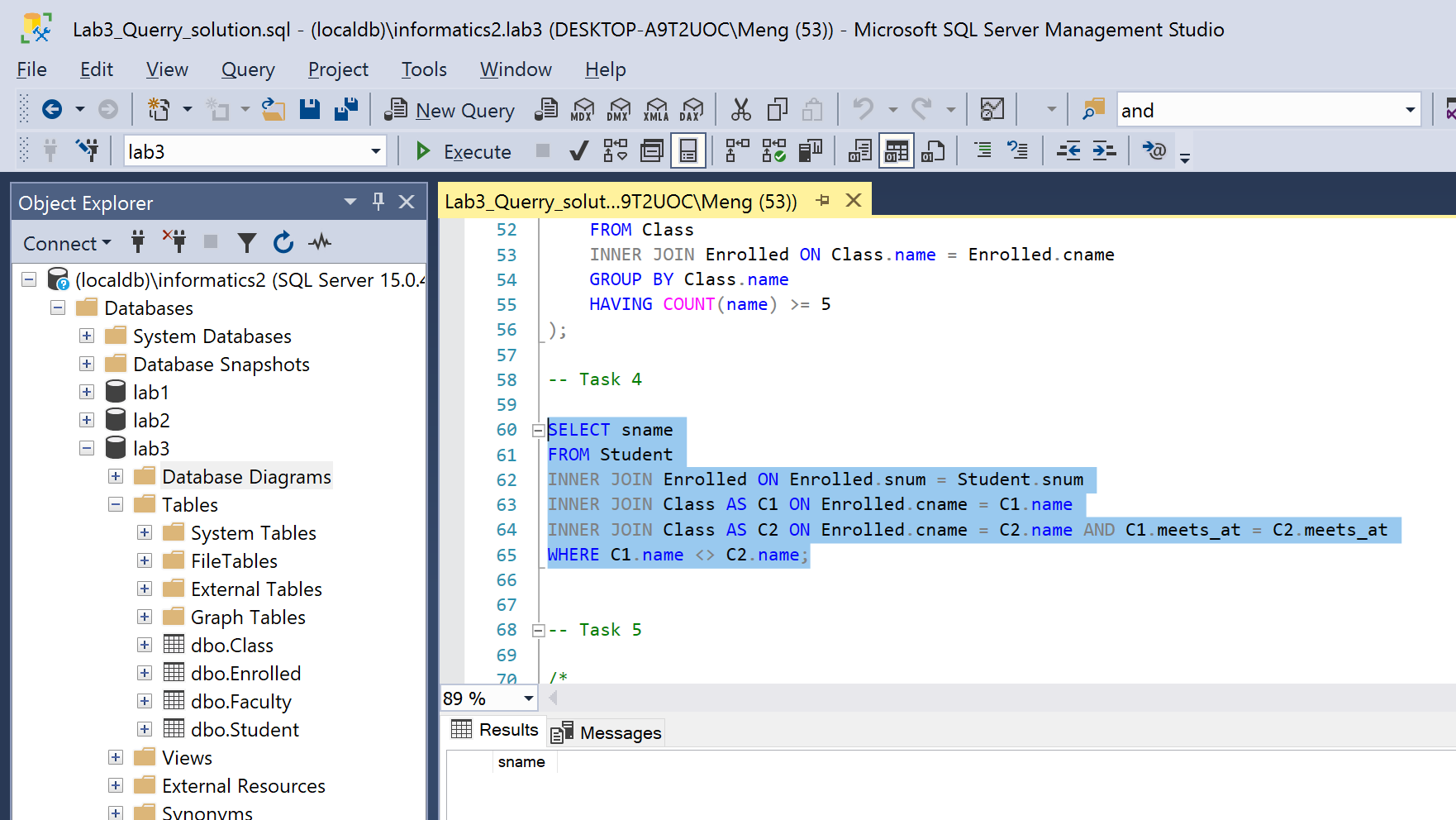
INNER JOIN Enrolled ON Enrolled.snum = Student.snum

INNER JOIN Class AS C1 ON Enrolled.cname = C1.name

INNER JOIN Class AS C2 ON Enrolled.cname = C2.name AND C1.meets\_at = C2.meets\_at

WHERE C1.name <> C2.name;

**Visual result:**



**Reasoning:**

In this task we want a name of student, so we SELECT sname FROM Student table we want to check a class time of each class student enrolled to, so we do INNER JOIN to Enrolled and Class. We INNER JOIN with class 2 times as C1 and C2. This gives us two instances of class C1 and C2 but actually they are same in everything. Bu this give us an ability to check a class from C1 if the name and the time is match with C2 or not. If they have same time but different name then this means this student take different class which occur at same time.

### Task #5

**Problem statement:** Find the names of faculty members who teach in every room some class is taught

**Solution:**

SELECT Faculty.fname

FROM Faculty

INNER JOIN Class ON Faculty.fid = Class.fid

GROUP BY Faculty.fname

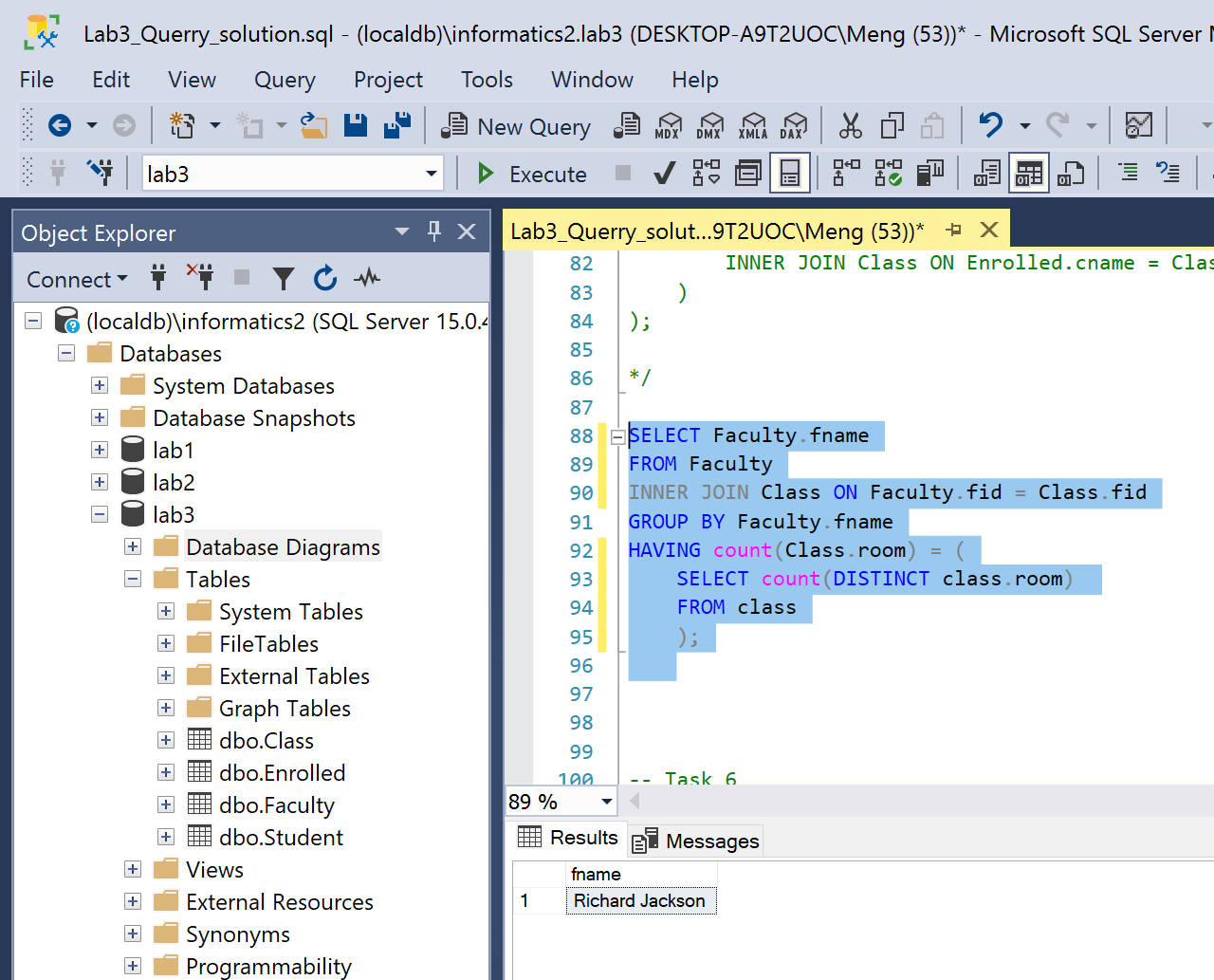
HAVING count(DISTINCT Class.room) = (

SELECT count(DISTINCT class.room)

FROM class

);

**Visual result:**



**Reasoning:**

In this task we need to find a member of a Faculty which is assigned to teach every room that some class is assigned to. So first we need to check how many room is assigned by count DISTINCT room column in Class table SELECT count(DISTINCT class.room) FROM class then compare this to the count distinct of room that faculty member is assigned to which first need to INNER JOIN faculty table with class table first. Then we can find the number of room that is assigned to faculty member by GROUP BY Faculty.fname and HAVING count(DISTINCT Class.room)

### Task #6

**Problem statement:** Find the names of faculty members for whom the combined enrollment of the courses that they teach is less than five

**Solution:**

SELECT Faculty.fname

FROM Faculty

WHERE Faculty.fname IN (

SELECT Faculty.fname

FROM Faculty

LEFT JOIN Class ON Class.fid = Faculty.fid

LEFT JOIN Enrolled ON Enrolled.cname = Class.name

LEFT JOIN Student ON Student.snum = Enrolled.snum

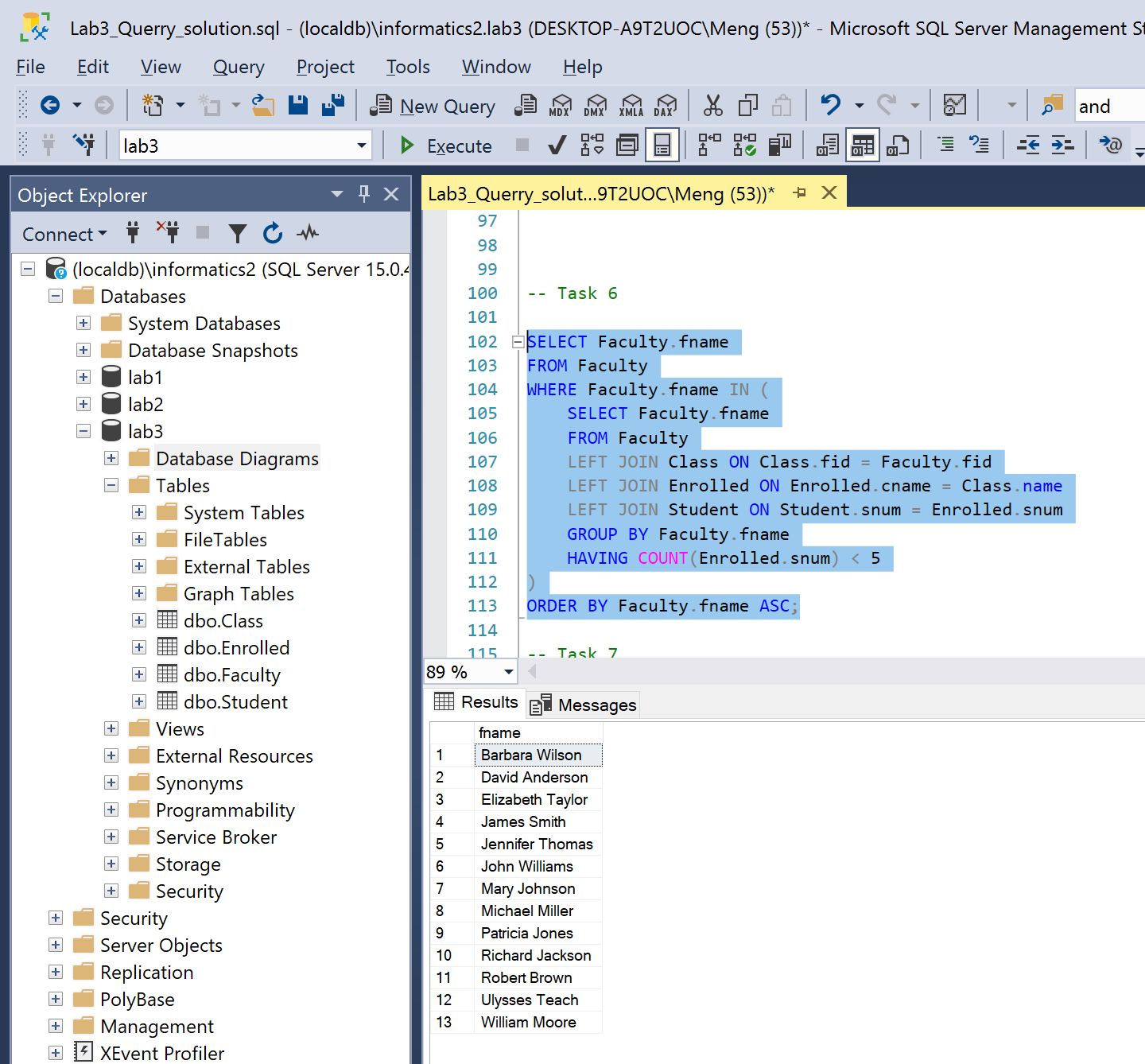
GROUP BY Faculty.fname

HAVING COUNT(Enrolled.snum) < 5

)

ORDER BY Faculty.fname ASC;

**Visual result:**



**Reasoning:**

This task first we can do (from nested select) is do LEFT JOIN faculty member to every other table first then group by faculty member name and count if they have enrolled class more than 5 or not. We can collect a name of those who has no more than 5 enrolled class or open class then return this result back to the select outside nested select. Then the outer select will just take those faculty member and show their properties as a result in ascending order. In this task we need to use LEFT JOIN because we want to get the information for this faculty member who does not need to teach any class as well which they will not be match if we use INNER JOIN.

### Task #7

**Problem statement:** For each level, print the level and the average age of students for that level. For all levels except JR print the level and the average age of students for that level

**Solution:**

SELECT Student.standing, AVG(Student.age) AS [Average age]

FROM Student

GROUP BY Student.standing

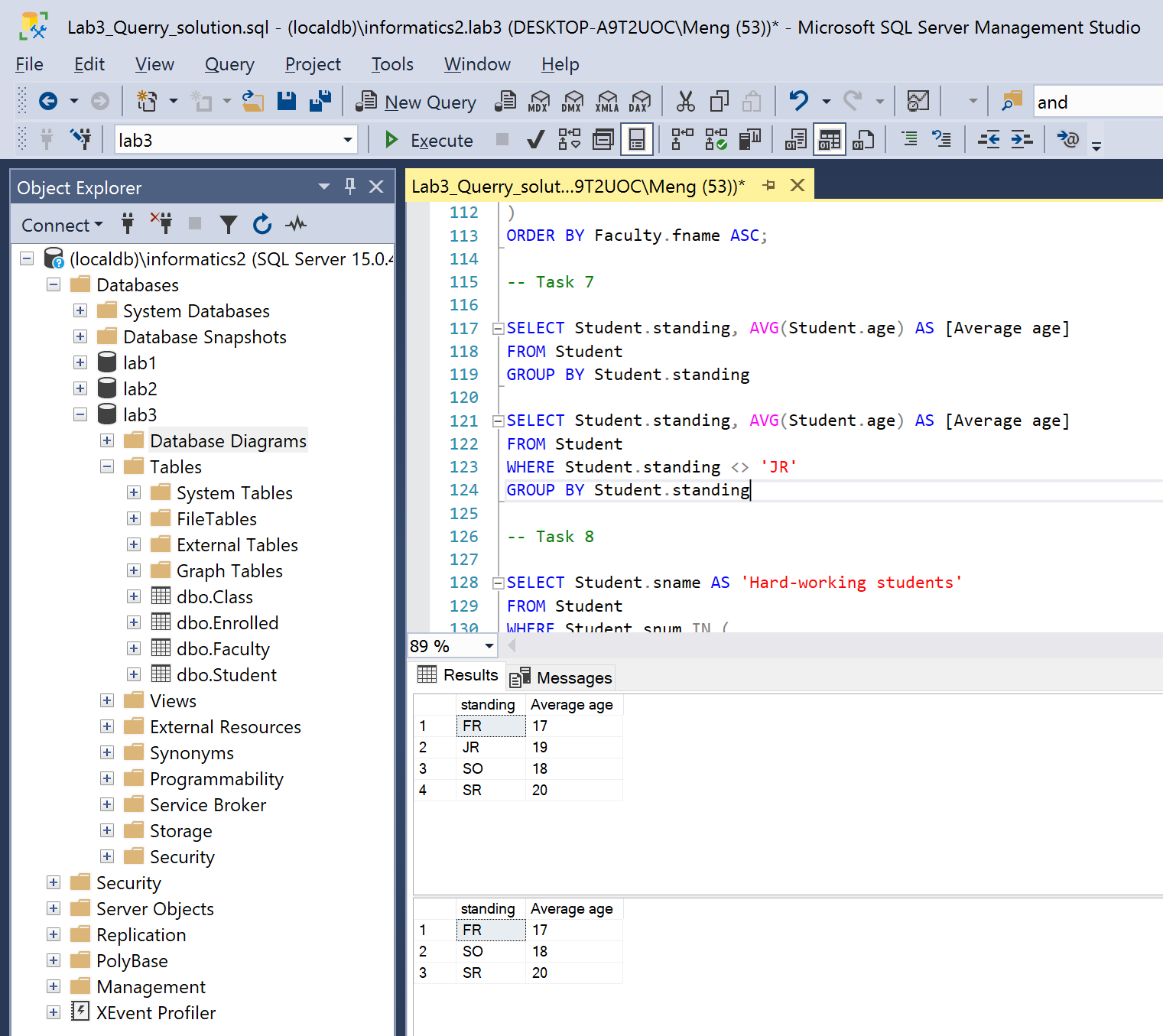
SELECT Student.standing, AVG(Student.age) AS [Average age]

FROM Student

WHERE Student.standing <> 'JR'

GROUP BY Student.standing

**Visual result:**



**Reasoning:**

This task both implementations are super similar, we just need to group a student as their standing and use average function to find average of their age. For second one, we can just add one condition which does not take 'JR'standing into account.

### Task #8

**Problem statement:** Find the names of students enrolled in the maximum number of classes. Rename the resulting column to “Hard-working students”

**Solution:**

SELECT Student.sname AS 'Hard-working students'

FROM Student

WHERE Student.snum IN (

SELECT Enrolled.snum

FROM Enrolled

GROUP BY Enrolled.snum

HAVING COUNT(Enrolled.cname) >= ALL (

SELECT COUNT (EnrolledTemp.cname)

FROM Enrolled AS EnrolledTemp

GROUP BY EnrolledTemp.snum

)

)

ORDER BY Student.sname ASC;

**Visual result:**

Graphical user interface, text, application

Description automatically generated

**Reasoning:**

For this task we first need to find out the number of class that each student enrolled in. this result will be check later we will call this result R1. Now we need to compare each student to the result R1 that we just got. We can do so by only Enrolled table, we group by student id then we count the number of class. In this way we will get the number of class that each student enrolled in then we compare each of these students to each of the tuple result R1 that we got it earlier. In this way we will get the name of student who they have enrolled class as max enrolled class by student. We give this result back to a outer select to just show this students properties as a final result order by their name in ascending order.

### Task #9

**Problem statement:** Find the names of students not enrolled in any class. Rename the resulting column to “Lazy students”

**Solution:**

SELECT Student.sname AS 'Lazy students'

FROM Student

WHERE Student.snum IN (

SELECT Student.snum

FROM Student

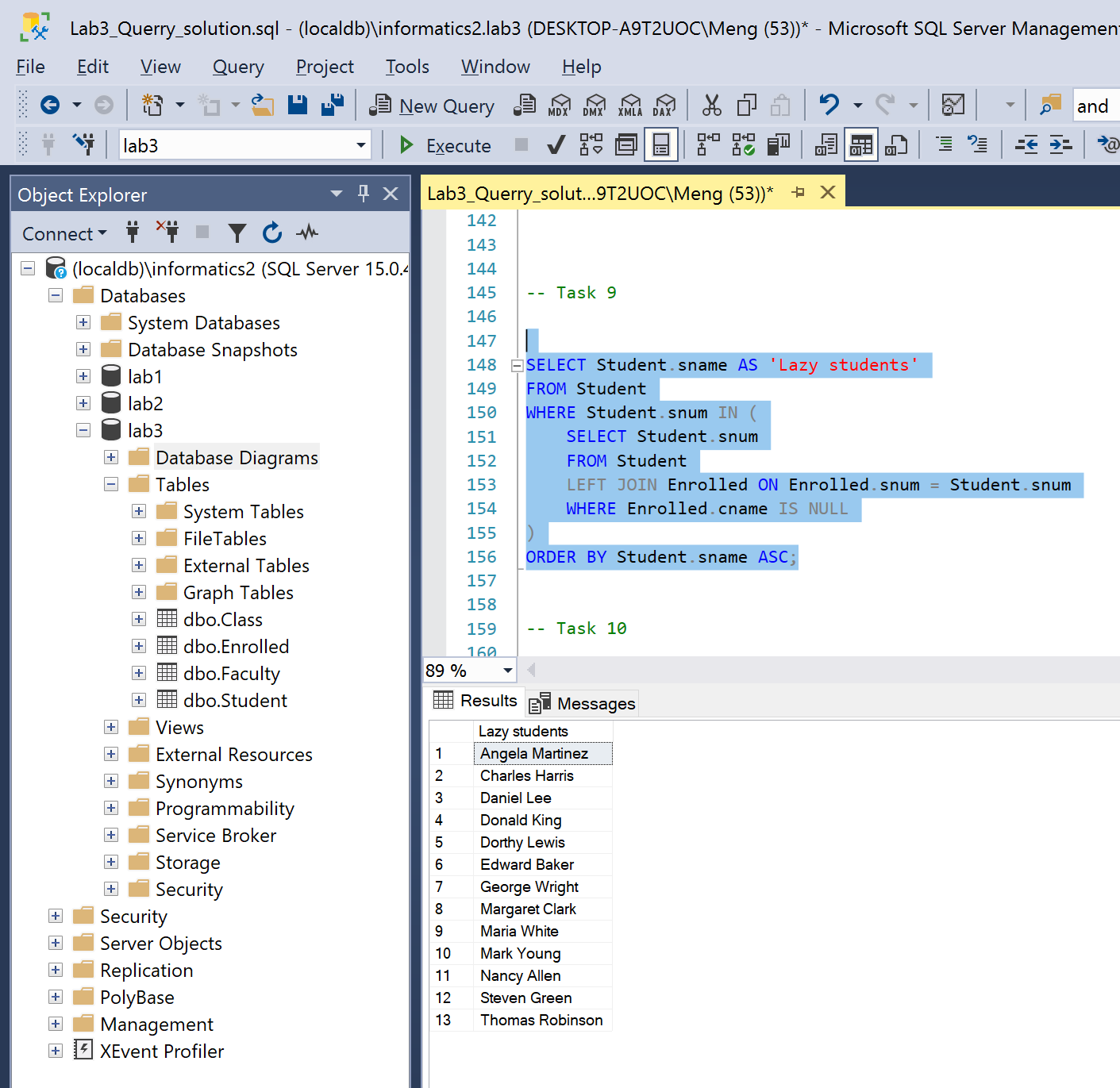
LEFT JOIN Enrolled ON Enrolled.snum = Student.snum

WHERE Enrolled.cname IS NULL

)

ORDER BY Student.sname ASC;

**Visual result:**



**Reasoning:**

We can do this by first LEFT JOIN student table to enrolled table then just select all student who does not have enrolled class or enrolled class IS NULL. Then order the result by name in ascending order.

### Task #10

**Problem statement:** For each age value that appears in Student, find the level value that appears most often. For example, if there are more FR level students aged 18 than SR, JR or SO students aged 18, you should print the pair (18, FR).

**Solution:**

SELECT DISTINCT Student.age, Student.standing

FROM Student

WHERE Student.age IN (

SELECT Student.age

FROM Student

GROUP BY Student.age, Student.standing

HAVING COUNT(Student.standing) >= ALL (

SELECT COUNT(Student.standing)

FROM Student

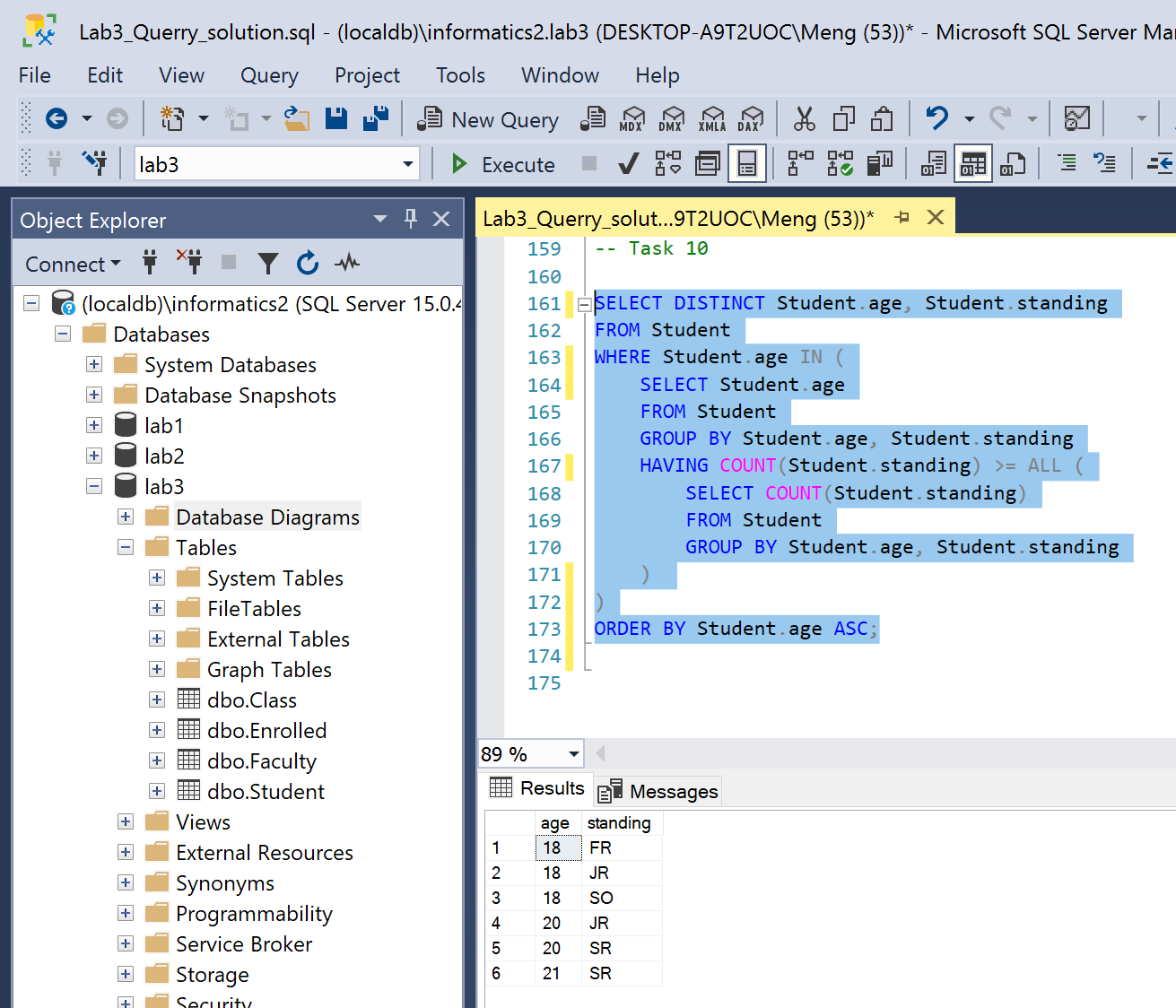
GROUP BY Student.age, Student.standing

)

)

ORDER BY Student.age ASC;

**Visual result:**



**Reasoning:**

In this task, first I tried to group by student age and count the number of standings, this could get me that in each age for each standing, how many student are present for each selection. This will be my result R1. Then, I tried to select age from student group by age and standing and count the number of their standing then compare to each of tuple in R1, this should give a result that for each age a most often appear standing if present.

### Instructions

1. **Problem statement is mandatory.**
2. **A solution without explanation is NOT accepted.**
3. **If you need to copy the source code, you can do it with copy/paste commands. Please do not use screenshots for code listings.**
4. **Other screenshots (figures, graphs, etc.) should be scaled appropriately. Please cut off unnecessary elements on the images.**