# Faculty Of Computing

**CS220: Database Systems**

**Class: BESE-13AB**

# Lab 10: Correlated Nested Queries in SQL

# Date: November 27, 2023

# Time: 10:00-01:00 & 02:00-05:00

# Instructor: Dr. Bilal Ali & Dr. Shah Khalid

# Lab Engineer: Sundas Dawood

**Name:** Aimen Munawar

**Class:** BESE-13-A

**CMS ID:** 415867

# 

# 

# 

**Lab 10: Correlated Nested Queries in SQL**

# Introduction

* A correlated subquery is one where the inner query depends on values provided by the outer query. This means the inner query is executed repeatedly, once for each row that might be selected by the outer query.

# Objectives

After performing this lab, students should be able to:

1. Understand and formulate Correlated Nested Queries

# Tools/Software Requirement

* MySQL Community Server
* MySQL Workbench

# Helping Material

<https://www.youtube.com/watch?v=0d419Vo2Po4>

# Description

**Nested Queries/Subqueries**

A nested/subquery is a SQL query nested inside a larger query, such inner-outer queries are called nested queries

A subquery may occur in:

* A SELECT clause
* A FROM clause
* A WHERE clause

**Rule of thumb**: avoid writing nested queries when possible; keep in mind that sometimes it’s impossible

**CORRELATED NESTED/SUBQUERIES**

A correlated subquery is one where the inner query depends on values provided by the outer query. This means the inner query is executed repeatedly, once for each row that might be selected by the outer query. For example,

Query: **Find the names of students enrolled in any classes**

select distinct s.sname

from student s

where Exists (select \* e.snum

from enrolled e where s.snum = e.snum)

The subquery in this SELECT statement cannot be resolved independently of the main query. Notice that the outer query specifies that rows are selected from the student table with an alias name of s. The inner query compares the snum ( student name) column of the enrolled table to the same column of the student table. The subquery's results are correlated with each individual row of the main query – thus, the term correlated subquery.

When a subquery uses the EXISTS operator, the subquery functions as an existence test. The WHERE clause of the outer query tests for the existence of rows returned by the inner query.The subquery does not actually produce any data; rather, it returns a value of TRUE or FALSE.

The general format of a subquery WHERE clause with an EXISTS operator as shown below. Note that the NOT operator can also be used to negate the result of the EXISTS operator.

**WHERE [NOT] EXISTS (subquery)**

Subqueries using an EXISTS operator are a bit different from other subqueries, in the following ways:

The keyword EXISTS is not preceded by a column name, constant, or other expression.

The SELECT clause list of a subquery that uses an EXISTS operator almost always consists of an asterisk (\*). This is because there is no real point in listing column names since you are simply testing for the existence of rows that meet the conditions specified in the subquery.

The subquery evaluates to TRUE or FALSE rather than returning any data.

A subquery that uses an EXISTS operator will always be a correlated subquery.

The EXISTS operator is very important, because there is often no alternative to its use. All queries that use the IN operator or a modified comparison operator (=, <, >, etc. modified by ANY or ALL) can be expressed with the EXISTS operator. However, some queries formulated with EXISTS cannot be expressed in any other way. For example,

Query: **Find the names of students enrolled in any classes**

select distinct s.sname

from student s

where s.snum in (select e.snum

from enrolled e);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

select distinct s.sname

from student s

where Exists (select \* e.snum

from enrolled e where s.snum = e.snum)

The NOT EXISTS operator is the mirror-image of the EXISTS operator. A query that uses NOT EXISTS in the WHERE clause is satisfied if the subquery returns no rows.

# Lab Task

**Use the following database schema to fulfill the following information needs:**

**Student (snum: integer, sname: char(30), major: char(25), level: char(2))**

**Faculty (fid: integer, fname: char(30), deptid: integer)**

**Class (cname: char(40), meets\_at: char(20), room: char(10), fid: integer | fid REFS Faculty.fid)**

**Enrolled (snum: integer, cname: char(40) | snum REFS student.snum, cname REFS class.name)**

**Schema already uploaded on LMS you just have to run it .**

**Write SQL expressions for the following queries and execute them:**

1. Find the name of faculty members who do not teach any course.

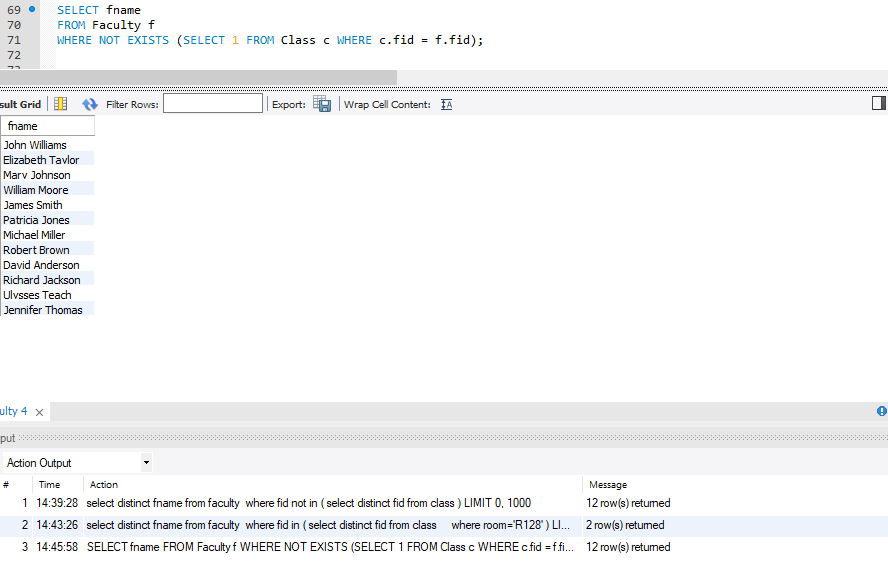
**Code:**

SELECT fname

FROM Faculty f

WHERE NOT EXISTS (SELECT 1 FROM Class c WHERE c.fid = f.fid);

**Output:**

****

1. Find the names of faculty members that have taught classes only in room R128.

**Code:**

SELECT DISTINCT f.fname

FROM Faculty f

WHERE NOT EXISTS (

SELECT 1

FROM Class c

WHERE c.fid = f.fid AND c.room <> 'R128'

) AND EXISTS (

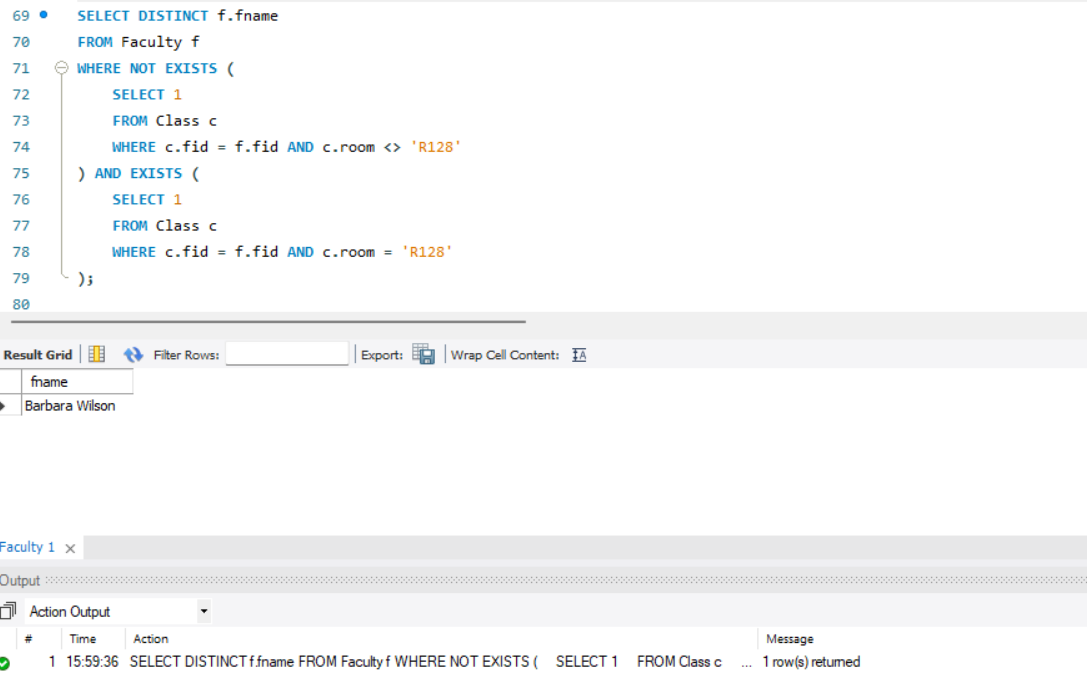
SELECT 1

FROM Class c

WHERE c.fid = f.fid AND c.room = 'R128'

);

**Output:**

****

1. Retrieve the snum and sname of students who have taken classes from both ‘Ivana Teach’ and ‘Linda Davis’.

**Code:**

SELECT s.snum, s.sname

FROM Student s

WHERE EXISTS (

SELECT 1

FROM Enrolled e1

JOIN Class c1 ON e1.cname = c1.cname

JOIN Faculty f1 ON c1.fid = f1.fid

WHERE s.snum = e1.snum AND f1.fname = 'Ivana Teach'

)

AND EXISTS (

SELECT 1

FROM Enrolled e2

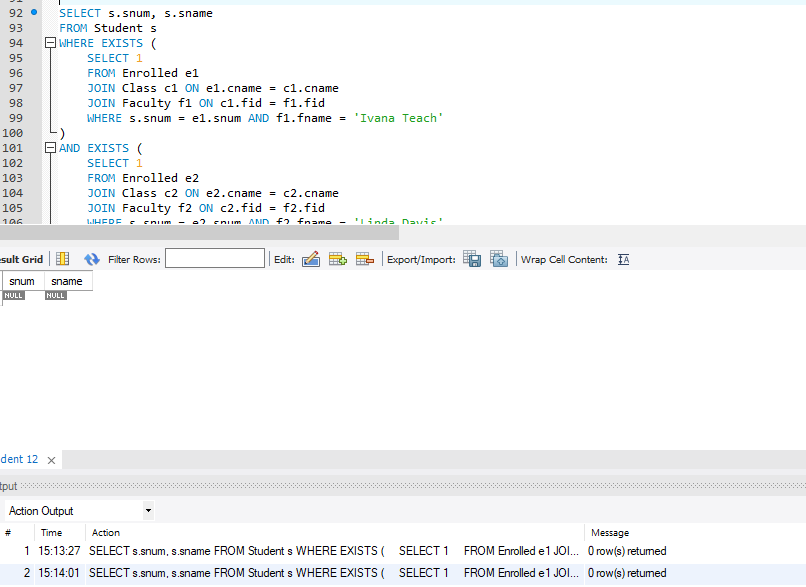
JOIN Class c2 ON e2.cname = c2.cname

JOIN Faculty f2 ON c2.fid = f2.fid

WHERE s.snum = e2.snum AND f2.fname = 'Linda Davis'

);

**Output:**

****

1. Find the name of faculty members that do not teach to class ‘database systems’.

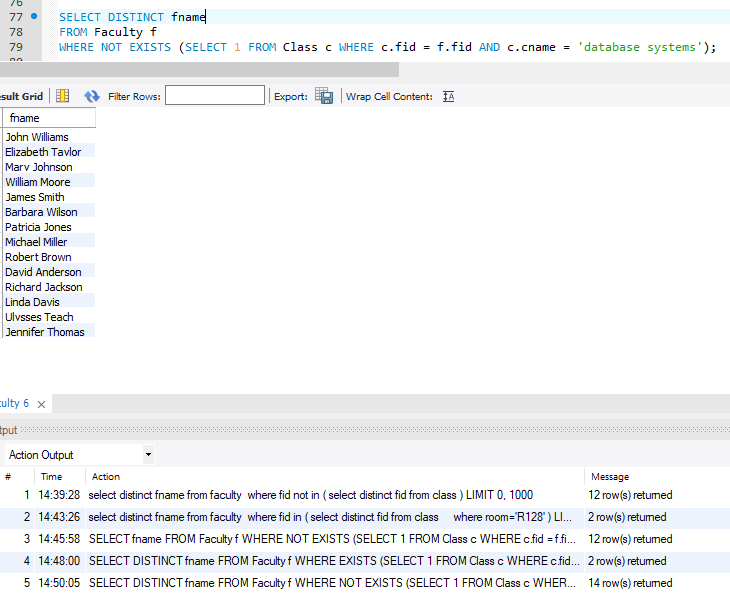
**Code:**

SELECT DISTINCT fname

FROM Faculty f

WHERE NOT EXISTS (SELECT 1 FROM Class c WHERE c.fid = f.fid AND c.cname = 'database systems');

**Output:**

****

1. Find the name of faculty member, department who has taught the maximum number of distinct classes.

**Code:**

SELECT f.fname AS FacultyName, f.dept\_id AS DepartmentID,

(SELECT COUNT(DISTINCT c.cname)

FROM Class c

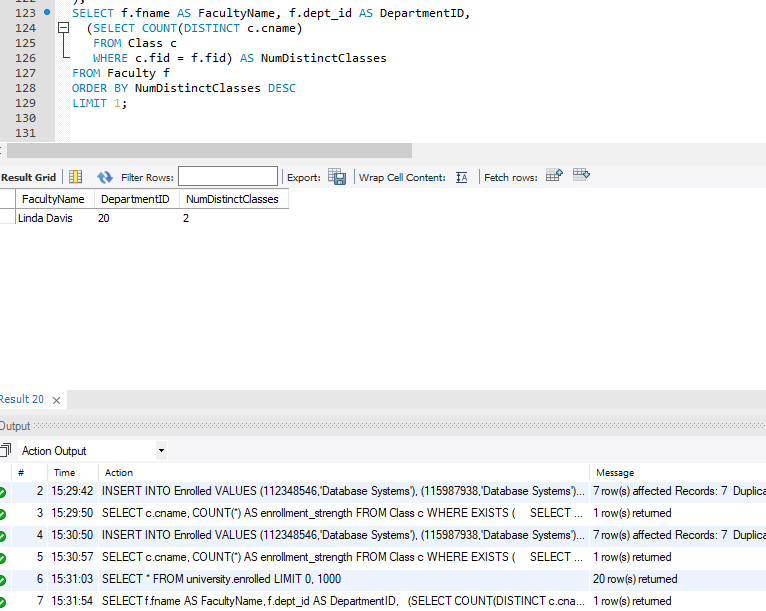
WHERE c.fid = f.fid) AS NumDistinctClasses

FROM Faculty f

ORDER BY NumDistinctClasses DESC

LIMIT 1;

**Output:**

****

1. Find the names of all classes and their enrollment strength that have enrollment greater than 5.

**Code:**

SELECT c.cname, COUNT(\*) AS enrollment\_strength

FROM Class c

WHERE EXISTS (

SELECT 1

FROM Enrolled e

WHERE e.cname = c.cname

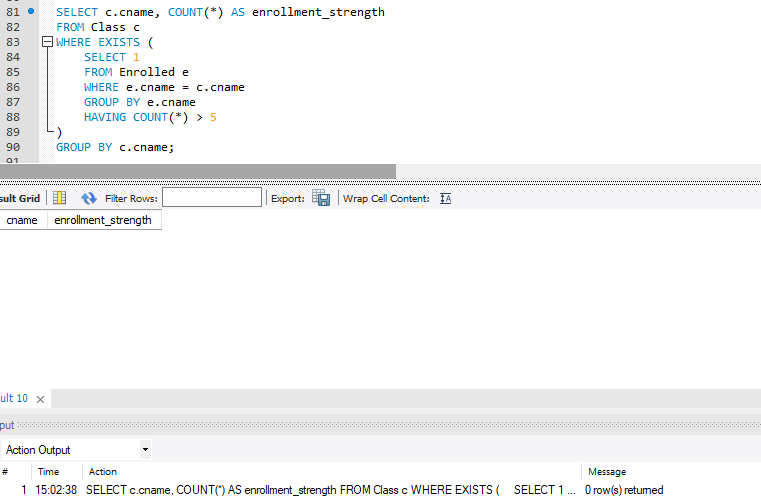
GROUP BY e.cname

HAVING COUNT(\*) > 5

)

GROUP BY c.cname;

**Output:**

****

1. Find the names of all students who are enrolled in two classes that meet at the same time.

**Code:**

SELECT DISTINCT s.sname

FROM Student s

WHERE EXISTS (

SELECT 1

FROM Enrolled e1

JOIN Class c1 ON e1.cname = c1.cname

WHERE s.snum = e1.snum

AND EXISTS (

SELECT 1

FROM Enrolled e2

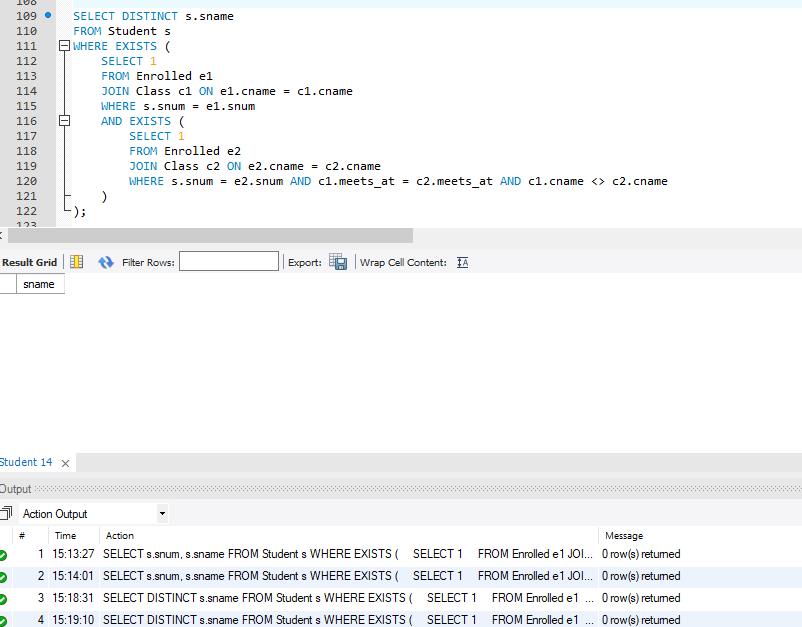
JOIN Class c2 ON e2.cname = c2.cname

WHERE s.snum = e2.snum AND c1.meets\_at = c2.meets\_at AND c1.cname <> c2.cname

)

);

**Output:**



# Deliverables

Complete your lab tasks in SQL workbench and submit a word file in with queries along with the screenshots of the results to all the questions attempted. Upload it on LMS. The marking will be based on viva/lab task submission.