****

**School of Engineering & Technology**

**Asian Institute of Technology**

**AT82.02 - Data Modeling and Management**

****

**Date: 2 SEPTEMBER 2022**

**Lab Report: 4**

**Submitted To Submitted By:**

**Dr. Chutiporn Anutariya Todsavad Tantortan st123012**

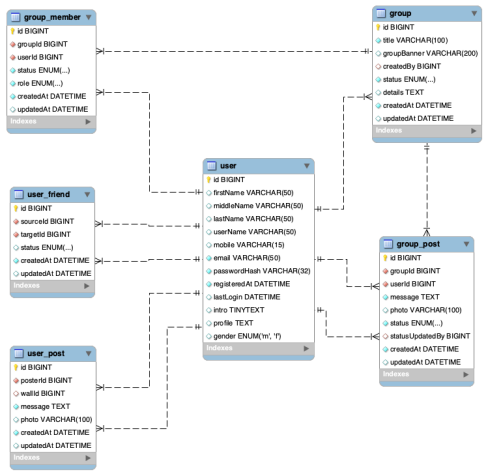
# Lab04: SQL (Part III)

**Objective:** To study and practice the SQL commands for data query and retrieval (SQL DML)

**Estimated Time:** 2 hours

**Number of Tasks: 10**

**Due:** Thursday 8 September 2022, 11:00 P.M.

**Description: Simple Social Network System (SNS) **

In this take home exercise, you are required to show how you query data from the Social Network System database to achieve each data requirement. Many querying techniques you have learned until this fourth laboratory will be employed.

**Reference for SQL commands:** https://www.w3schools.com/sql/

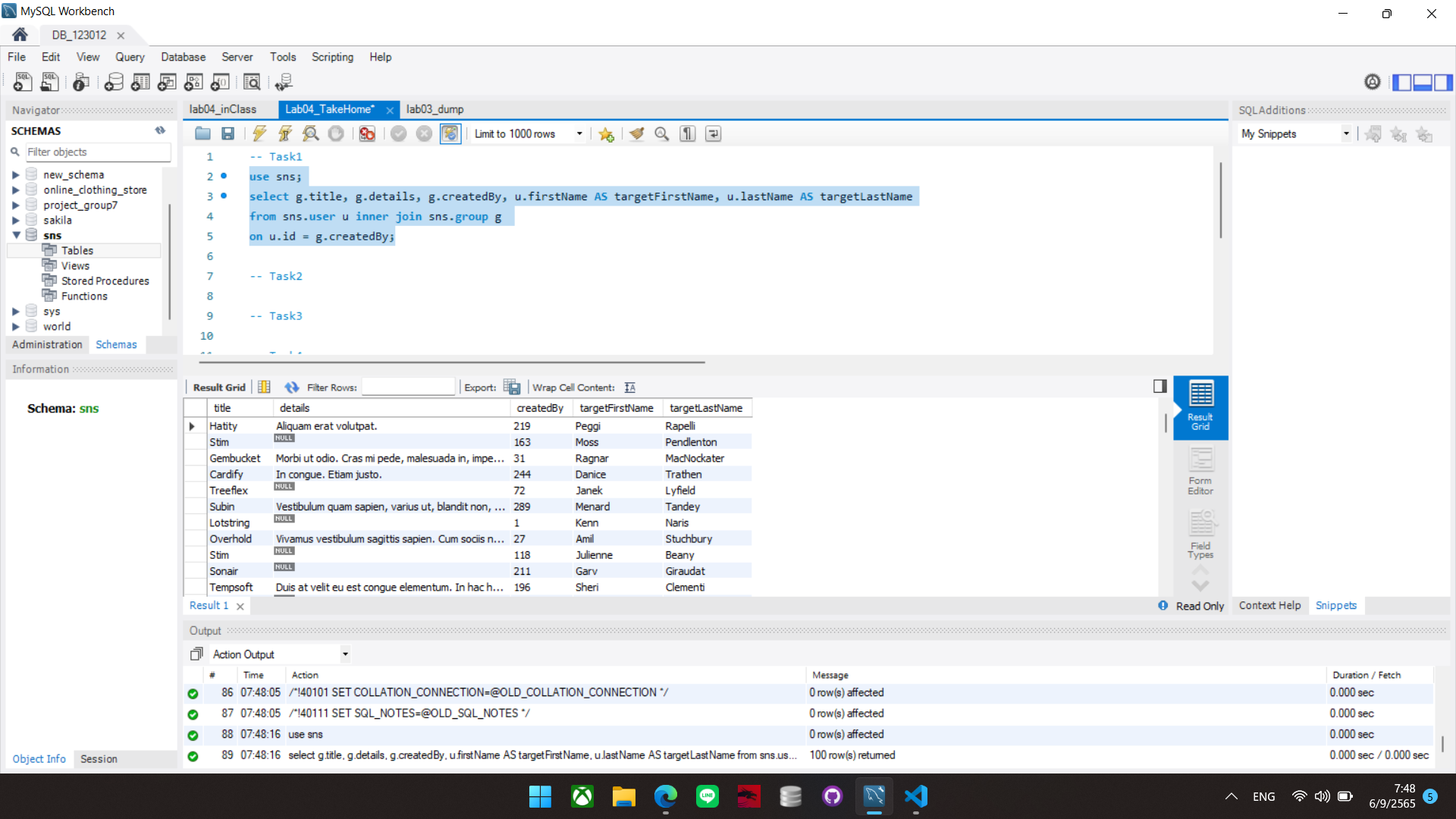
**Task 1:** Show all group information and the creator of the group.

[Paste your code and screenshots here]

use sns;

select g.title, g.details, g.createdBy, u.firstName AS targetFirstName, u.lastName AS targetLastName from sns.user u

inner join sns.group g on u.id = g.createdBy;



**Task 2:** Suppose there is a user (id= 55), who needs to know who has rejected their friend request. Show all names and their contact information (mobile and email). Answer using both JOIN keyword and subquery (total of 2 commands for this task).

[Put your both SQL commands and screenshots here]

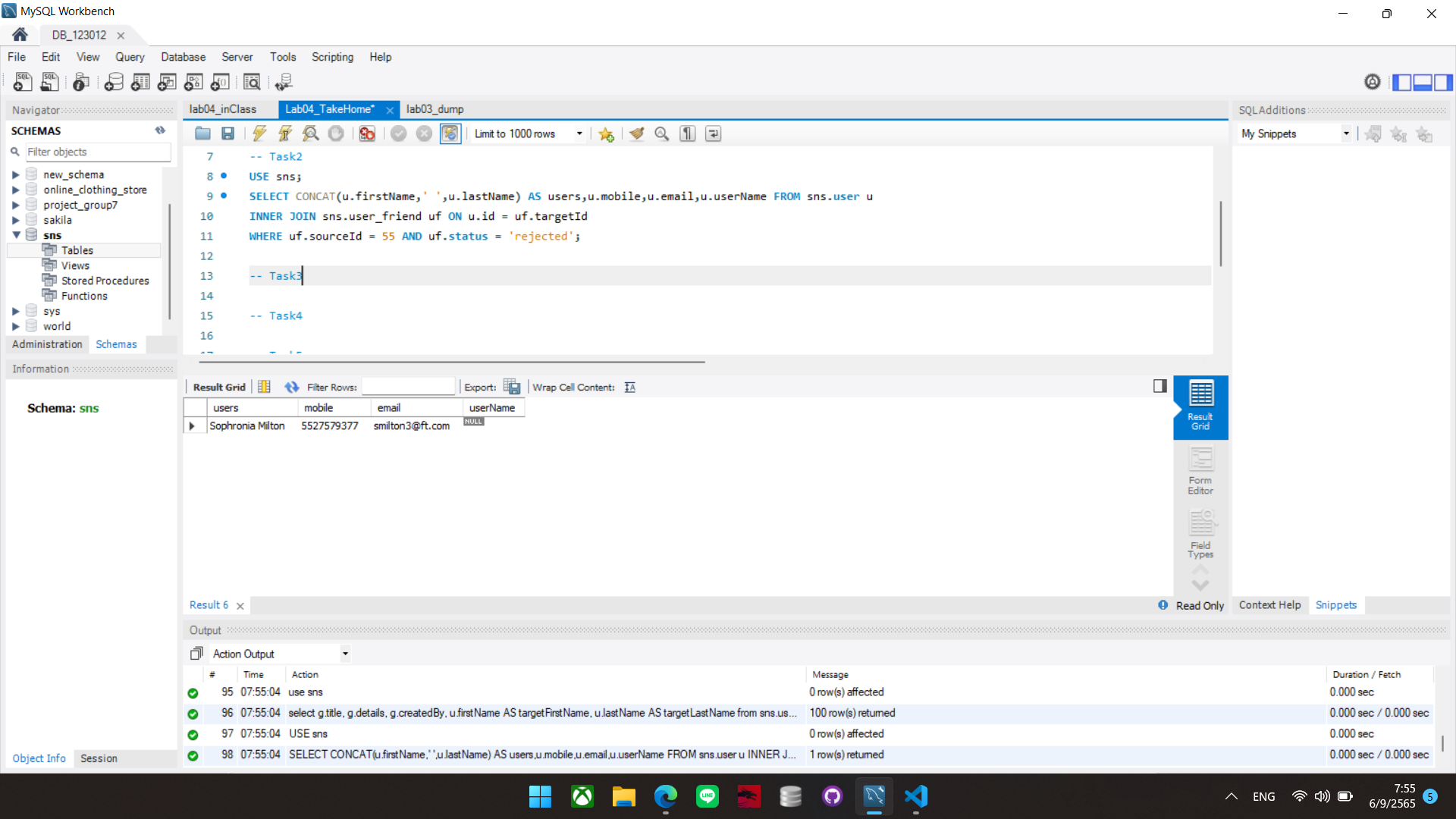
USE sns;

SELECT CONCAT(u.firstName,' ',u.lastName) AS users,u.mobile,u.email

FROM sns.user u

INNER JOIN sns.user\_friend uf ON u.id = uf.targetId

WHERE uf.sourceId = 55 AND uf.status = 'rejected';



**Task 3:** Show the first approved post of each group with the full name of the group, full name of the poster and the date the post was created.

[Paste your code and screenshots here]

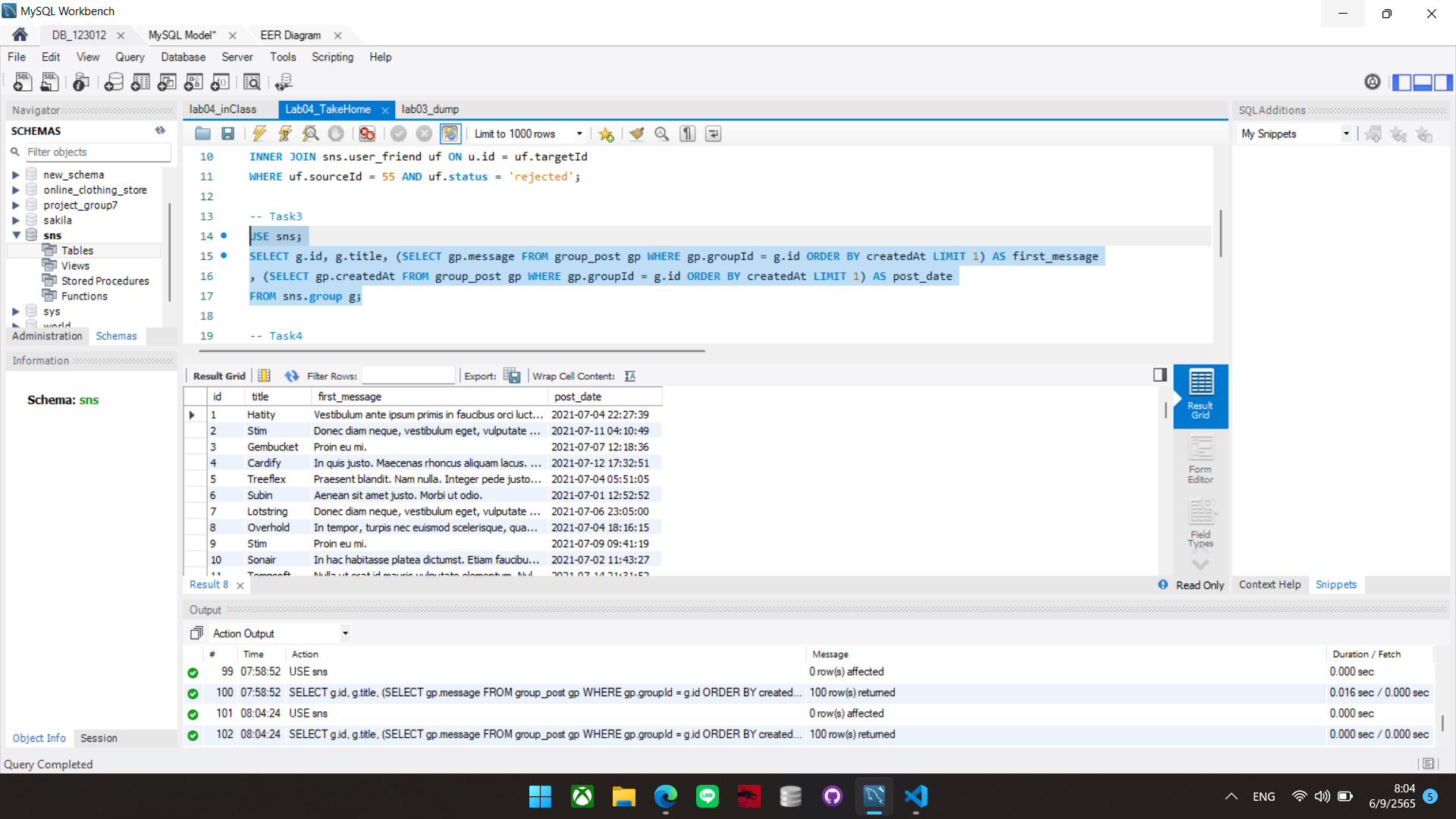
USE sns;

SELECT g.id, g.title,

(SELECT gp.message FROM group\_post gp WHERE gp.groupId = g.id ORDER BY createdAt LIMIT 1) AS first\_message,

(SELECT gp.createdAt FROM group\_post gp WHERE gp.groupId = g.id ORDER BY createdAt LIMIT 1) AS post\_date

FROM sns.group g;



**Task 4:** Summarize how many posts are created on each group wall. Separate columns by group name and the total number of the posters for each gender. Hint: MalePosts | FemalePosts | UnidentifiedPosts

[Paste your code and screenshots here]

SELECT g.title ,

(SELECT COUNT(gm.id) FROM group\_member gm

INNER JOIN sns.user u ON u.id = gm.userId

WHERE gm.groupId = g.id AND u.gender = 'm')

AS MalePost,

(SELECT COUNT(gm.id) FROM group\_member gm

INNER JOIN sns.user u ON u.id = gm.userId

WHERE gm.groupId = g.id AND u.gender = 'f')

AS FemalePost,

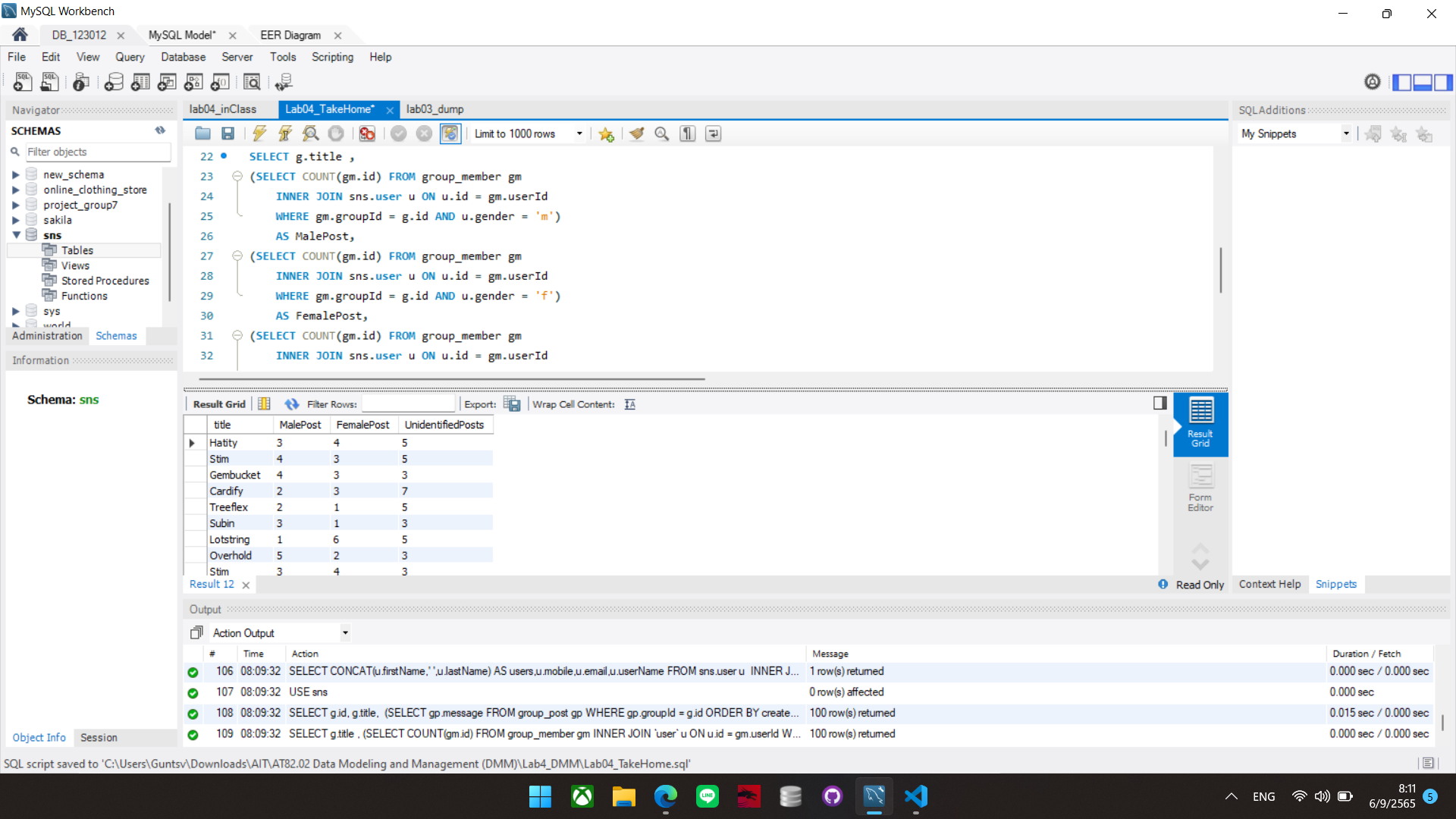
(SELECT COUNT(gm.id) FROM group\_member gm

INNER JOIN sns.user u ON u.id = gm.userId

WHERE gm.groupId = g.id AND u.gender IS NULL )

AS UnidentifiedPosts

FROM sns.group g;



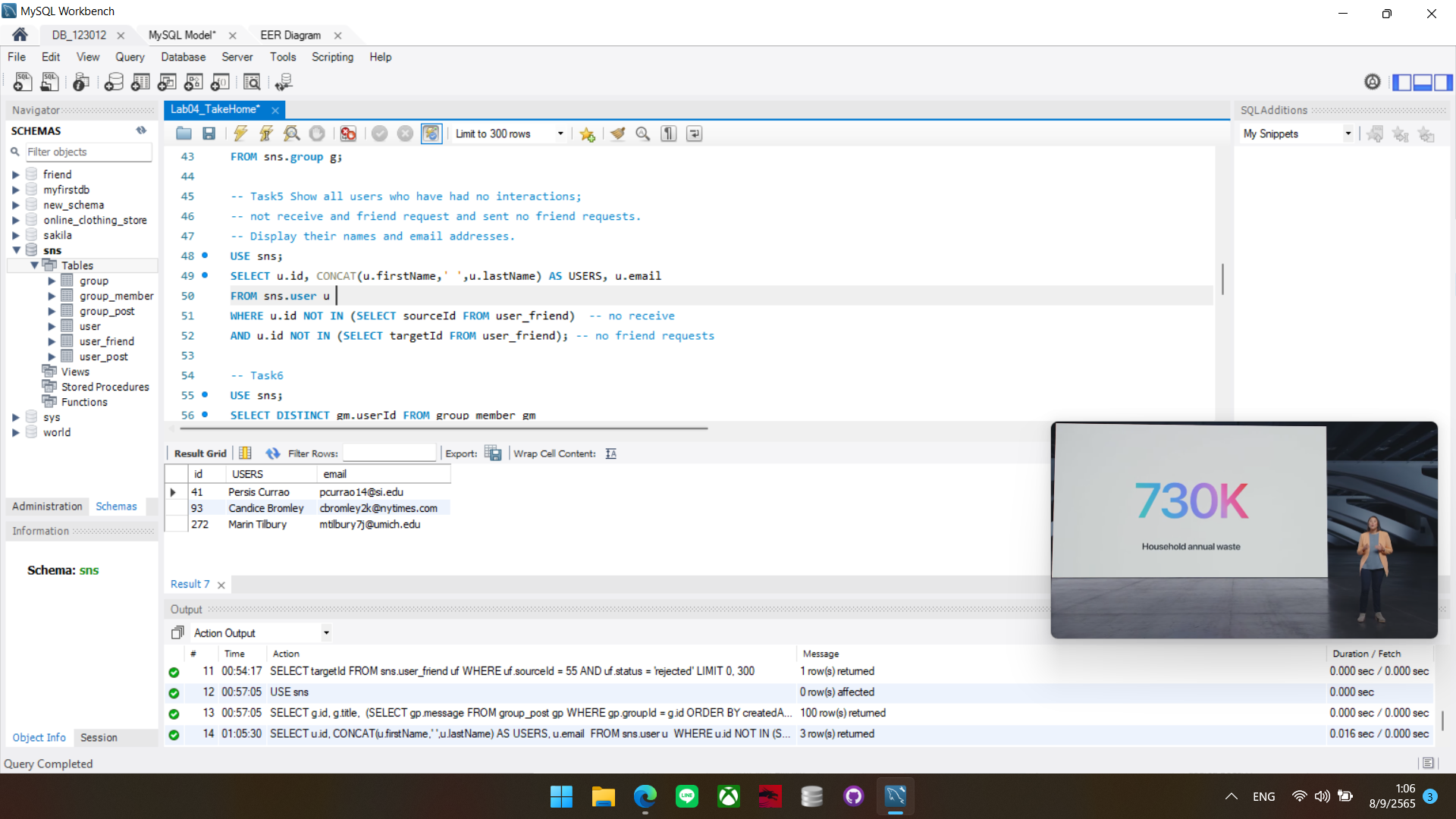
**Task 5:** Show all users who have had no interactions; not receive and friend request and sent no friend requests. Display their names and email addresses.

[Paste your code and screenshots here]

SELECT u.id, CONCAT(u.firstName,' ',u.lastName) AS USERS, u.email FROM sns.user u

WHERE u.id NOT IN (SELECT sourceId FROM user\_friend) -- no receive

AND u.id NOT IN (SELECT targetId FROM user\_friend); -- no friend requests



**Task 6:** Show the inactive members (members with no posts) of the group Id 15. List the full name, user id along with the name of the group.

[Paste your code and screenshots here]

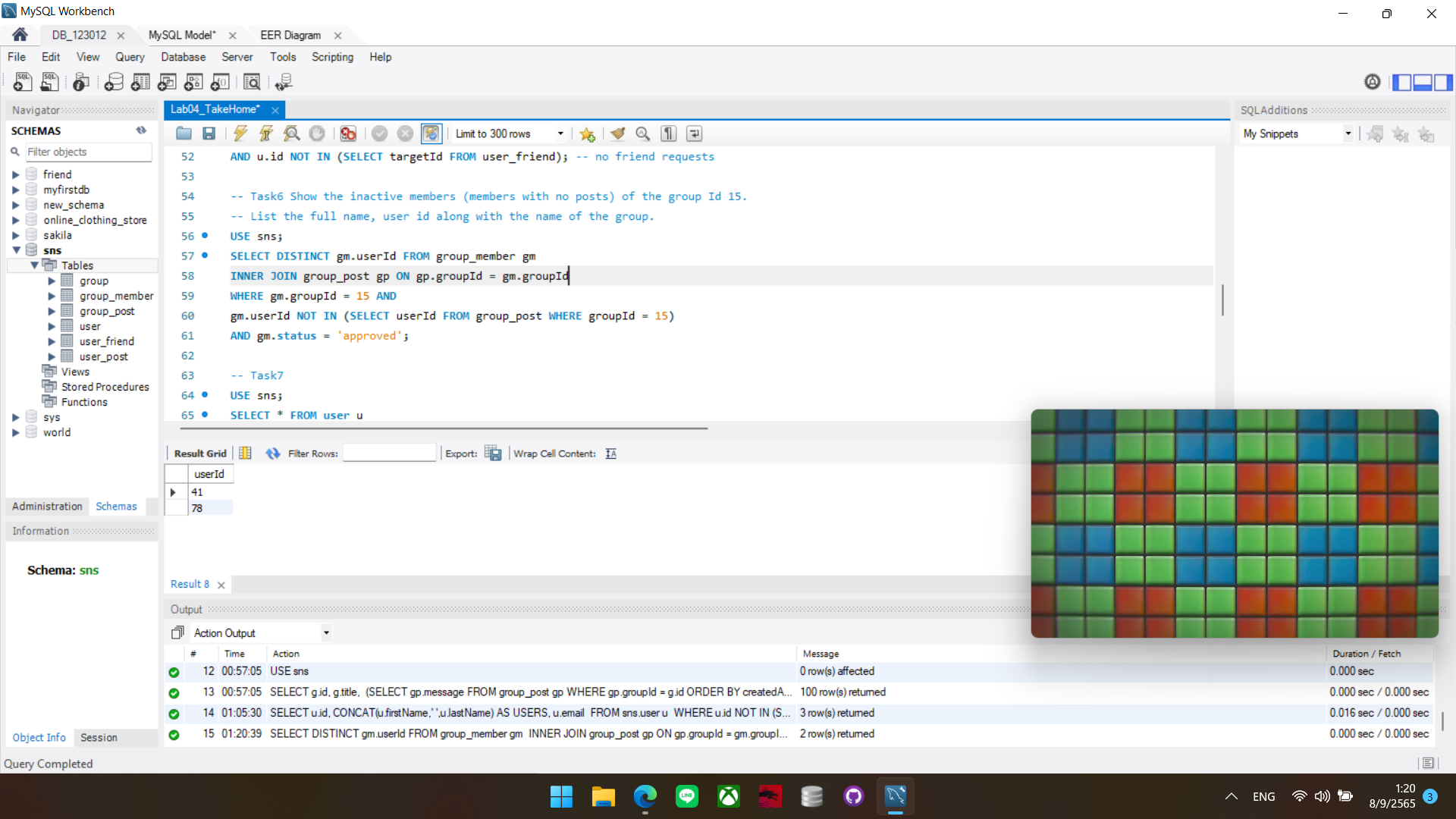
SELECT DISTINCT gm.userId FROM group\_member gm

INNER JOIN group\_post gp ON gp.groupId = gm.groupId

WHERE gm.groupId = 15 AND

gm.userId NOT IN (SELECT userId FROM group\_post WHERE groupId = 15)

AND gm.status = 'approved';



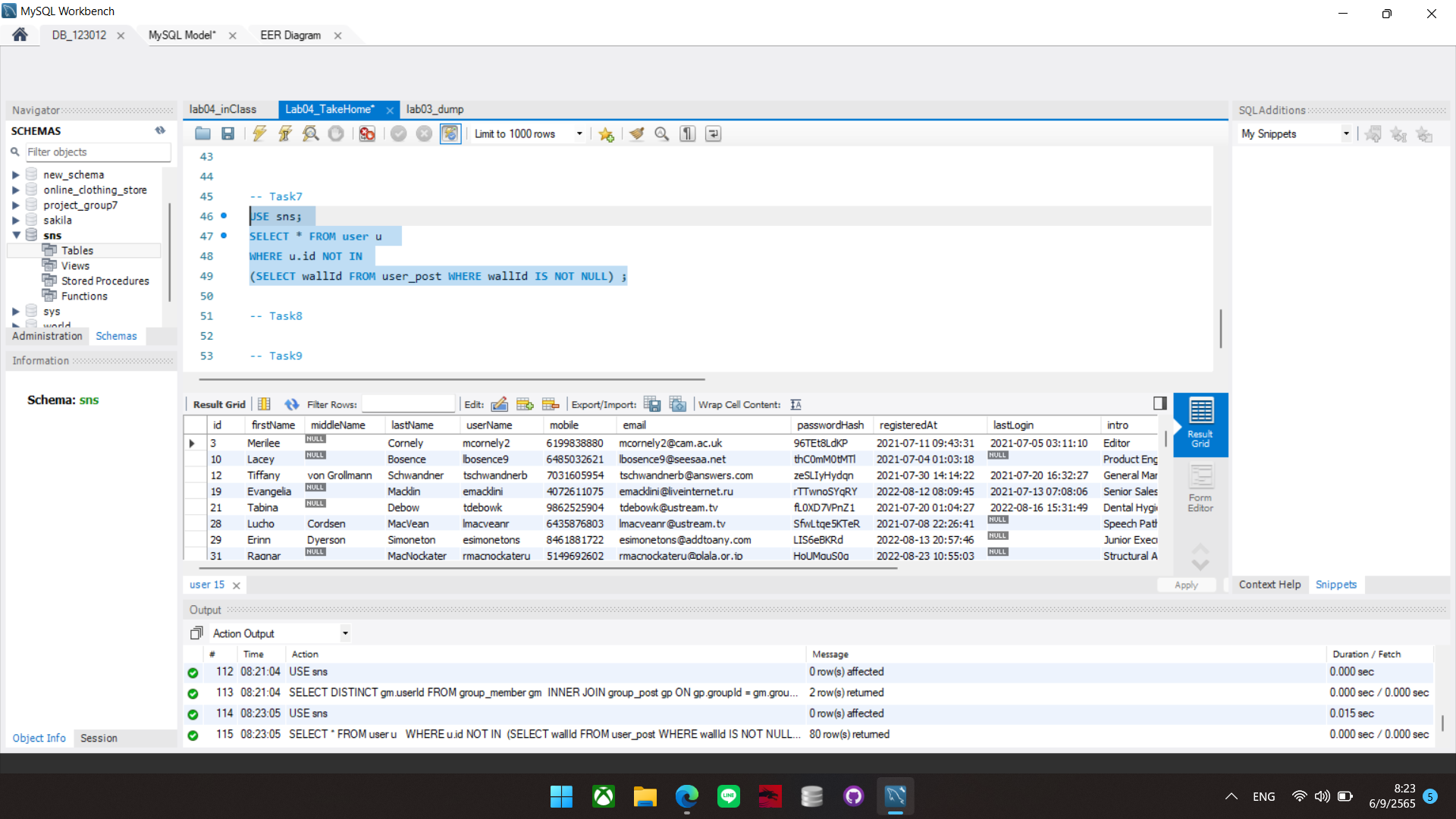
**Task 7:** Show all users who have their message wall empty.

[Paste your code and screenshots here]

SELECT \* FROM user u

WHERE u.id NOT IN

(SELECT wallId FROM user\_post WHERE wallId IS NOT NULL) ;



**Task 8:** Show the user information with the number of posts created by each user (both on friends, public, and group wall). Display as the following.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Id** | **Firstname** | **Email** | **PublicPosts** | **PostsToFriend** | **GroupPosts** | **TotalPosts** |
| XXX | PPPPPPP | ppp@test.com | XXX | XXX | XXX | XXX |
| ... | ... | ... ... | …… | …. | ... ... | ... |

Where

**PublicPosts**: The number of public posts

**PostsToFriend**: The number of posted on the other users’ wall

**GroupPosts**: The number of approved posted on any groups

**TotalPosts**: The total number of posts of the previous three columns HINT: Use subqueries to count the number of posts for each column

[Paste your code and screenshots here]

SELECT u.id, u.firstName, u.email ,

(SELECT COUNT(up.id) FROM user\_post up WHERE up.posterId = u.id AND up.wallId IS NULL) AS PublicPosts,

(SELECT COUNT(up.id) FROM user\_post up WHERE up.posterId = u.id AND up.wallId IS NOT NULL) AS PostsToFriend,

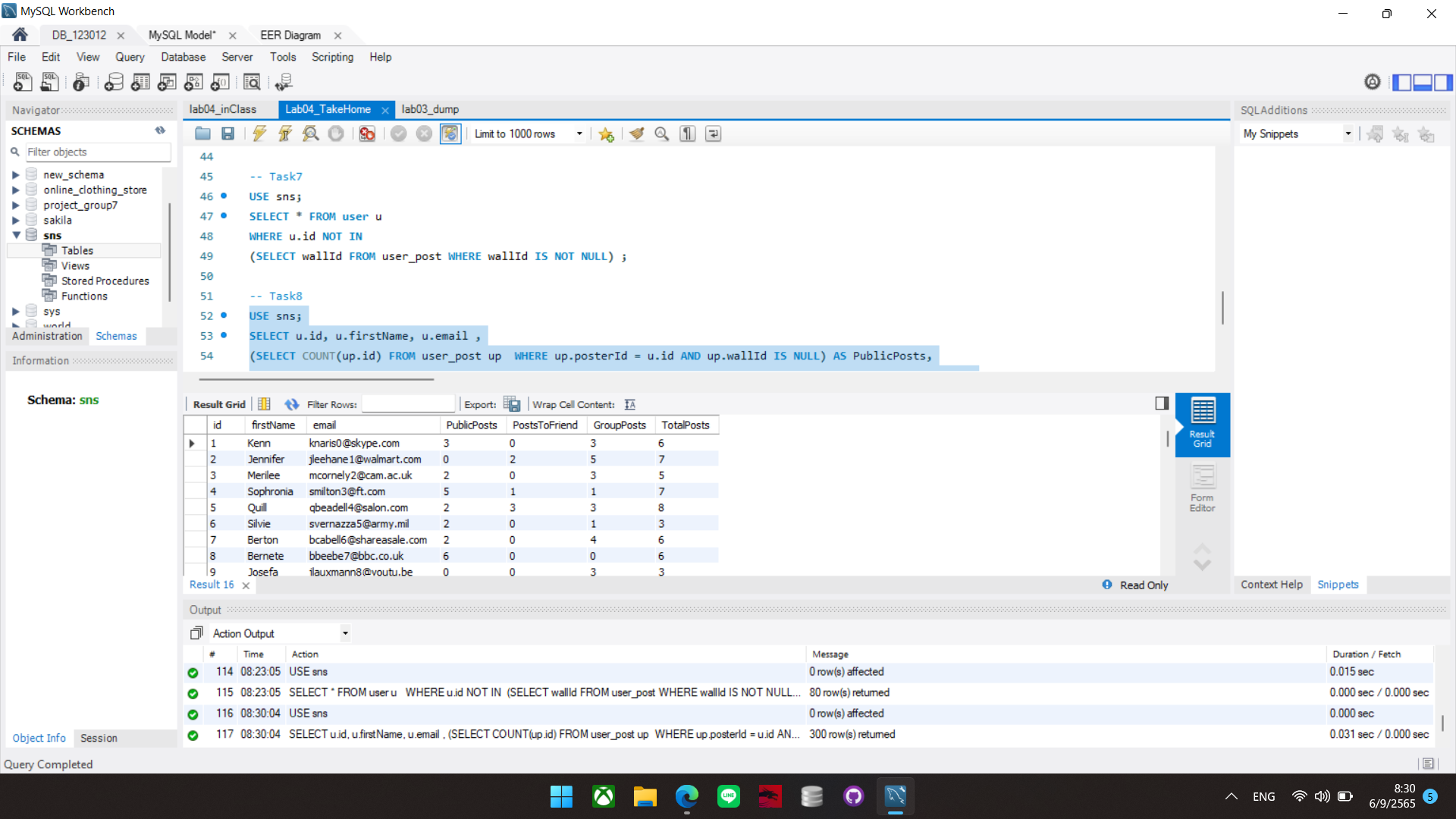
(SELECT COUNT(gp.id) FROM group\_post gp WHERE gp.userId = u.id ) AS GroupPosts,

((SELECT COUNT(up.id) FROM user\_post up WHERE up.posterId = u.id AND up.wallId IS NULL)

+(SELECT COUNT(up.id) FROM user\_post up WHERE up.posterId = u.id AND up.wallId IS NOT NULL)

+(SELECT COUNT(gp.id) FROM group\_post gp WHERE gp.userId = u.id ) ) AS TotalPosts

FROM user u;



**Task 9:** Summarize the group information by the number of members. Display the following columns.

|  |  |  |
| --- | --- | --- |
| **LightGroup** | **MediumGroup** | **DenseGroup** |
| XXX | XXX | XXX |

When

**LightGroup**: The number of groups having less than 5 members

**MediumGroup**: The number of groups having between 5 and 10 members **DenseGroup**: The number of groups having more than 10 members

HINT: Use 3 subqueries for each column filtered by the number of groups depending on the number of members.

[Paste your code and screenshots here]

USE sns;

SELECT

(SELECT COUNT(a.id) FROM

(SELECT g.id,COUNT(gm.userId) AS members FROM group\_member gm

INNER JOIN `group` g ON gm.groupId = g.id WHERE gm.status = 'approved' GROUP BY g.id ORDER BY g.id)

a WHERE a.members < 5) AS LightGroup ,

(SELECT COUNT(a.id) FROM

(SELECT g.id,COUNT(gm.userId) AS members FROM group\_member gm

INNER JOIN `group` g ON gm.groupId = g.id WHERE gm.status = 'approved' GROUP BY g.id ORDER BY g.id)

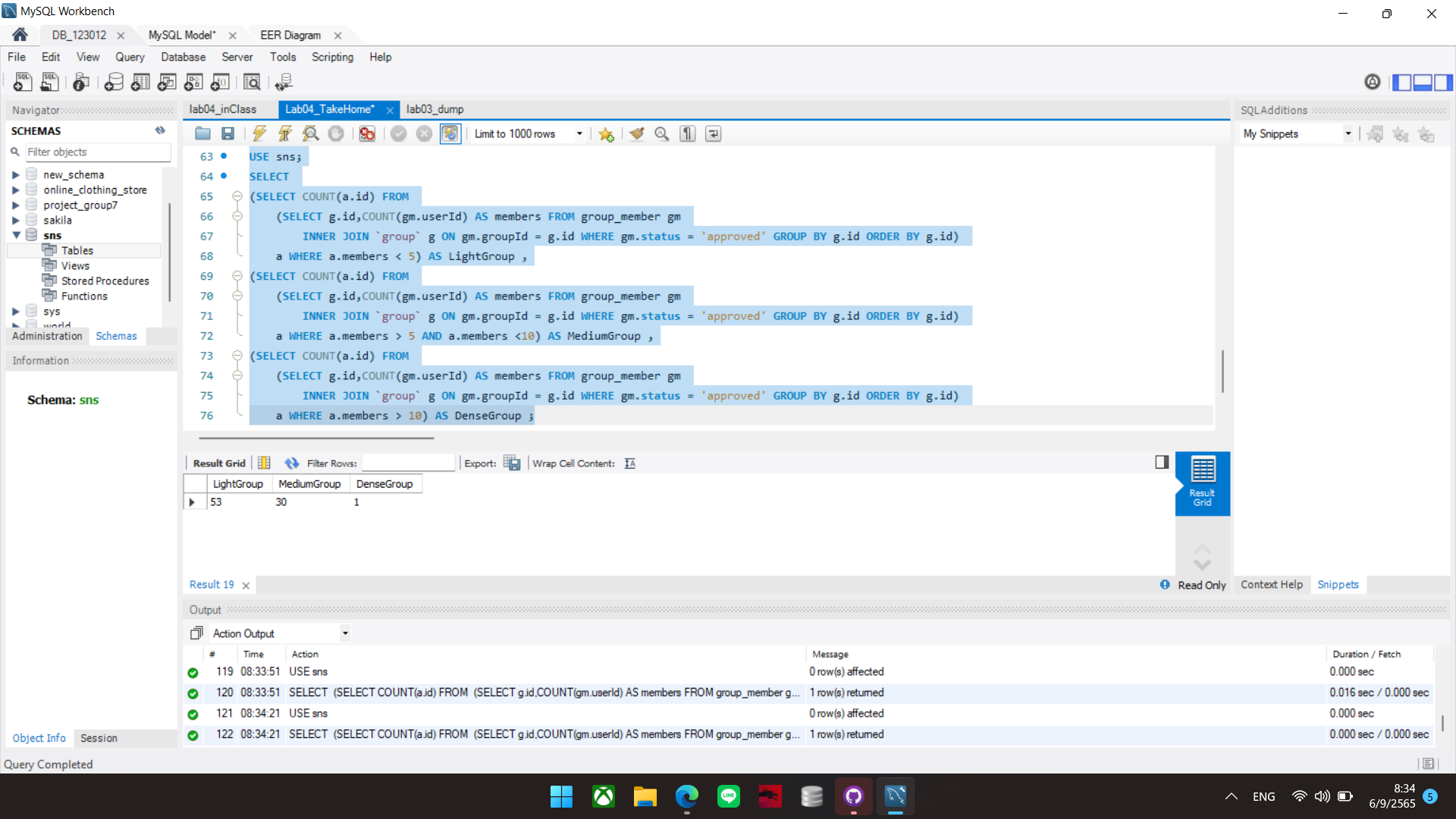
a WHERE a.members > 5 AND a.members <10) AS MediumGroup,

(SELECT COUNT(a.id) FROM

(SELECT g.id,COUNT(gm.userId) AS members FROM group\_member gm

INNER JOIN `group` g ON gm.groupId = g.id WHERE gm.status = 'approved' GROUP BY g.id ORDER BY g.id)

a WHERE a.members > 10) AS DenseGroup ;

****

**Task 10:** Summarize the user table by the range of membership duration (in DAYs). Display the following columns.

|  |  |  |
| --- | --- | --- |
| **Newbie** | **Pros** | **Veterans** |
| XXX | XXX | XXX |

Where

**Newbie**: The number of users who have registered less than 14 days

**Pros**: The number of users who have registered for 14 - 20 days

**Veterans**: The number of users who have registered longer than 20 days

HINT: Use 3 subqueries like the previous task on the **VALUE** of **TIMESTAMPDIFF** of the registeredDate

[Paste your code and screenshots here]

SELECT DISTINCT

(SELECT COUNT(a.id) FROM (SELECT u.id,timestampdiff(DAY,u.registeredAt,now()) AS days FROM user u) a WHERE days<14) AS Newbie ,

(SELECT COUNT(a.id) FROM (SELECT u.id,timestampdiff(DAY,u.registeredAt,now()) AS days FROM user u) a WHERE days BETWEEN 14 and 20) AS Pros,

(SELECT COUNT(a.id) FROM (SELECT u.id,timestampdiff(DAY,u.registeredAt,now()) AS days FROM user u) a WHERE days>20) AS Veterans

FROM sns.user ;

A screenshot of a computer

Description automatically generated