SQL-Lab05-DBS211-Transaction-Security

# This lab is different than the previous labs. You are to make a script file like the original script you used to load all your data. I will cut and paste the script to my database and run it.

# A suggestion is to build each part separately and then gradually put them together. I believe this will be easier at the start to know that each piece works. Then run and test your whole script.

# Objectives:

The purpose of this lab is to introduce the student to both transactions and security. In the real-world, databases tasks often involve multiple steps and **if any step in the middle fails, the procedure is a failure.** This lab walks the student through a couple transactions and lets them learn how various steps have varying consequences that they need to be aware of.

By the end of this lab, the student will be able to:

* Describe the steps of a transaction, how a transaction begins and ends and walk through live scenarios of a variety of transactions.
* Understand and act appropriately on what needs to be done in the case of transaction failure.
* Grant and revoke permissions to and from other users and public users from the database

# Submission:

***Your submission will be a single text-based SQL file with the solutions provided.***

Your submission needs **to contain a comment header block** and be commented and include your name, student id and oracle id as well as the question number and the solutions. Make sure every SQL statement terminates with a semicolon.

You will use following data to complete the given tasks:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Employee\_id** | **Last\_name** | **First\_name** | **email** | **salary** | **Department\_id** | **Manager\_id** | **Job\_id** |
| 100 | Patel | Ralph | rpatel@mail.com | 8500 | 80 | NULL | Sales Rep |
| 101 | Denis | Betty | [bdenis@mail.com](mailto:bdenis@mail.com) | 8000 | 80 | NULL | Sales Rep |
| 102 | Biri | Ben | [bbirir@mail.com](mailto:bbirir@mail.com) | 9300 | 20 | NULL | Sales Rep |
| 103 | Newman | Chad | [cnewman@mail.com](mailto:cnewman@mail.com) | 5000 | 50 | NULL | Sales Rep |
| 104 | Your Name | Audrey | aropebur@mail.com | 8900 | 60 | NULL | Sales Rep |

* ***SET TRANSACTION READ WRITE*** starts a new transaction.
* ***COMMIT*** commits the current transaction, making its changes permanent.
* ***SAVEPOINT <name>***  sets a pointer to a location that can be rolled back to.
* ***ROLLBACK*** rolls back the current transaction, canceling its changes.
* ***SET autocommit*** disables or enables the default ***autocommit*** mode for the current session.

**Tasks:**

It is very important that these tasks be performed in the order presented here for maximum learning.

## PART A - Transactions

1. List the 4 ways that we know that a transaction can be started
2. Using SQL, create an **empty** table, that is the same as the employees table, and name it ***newEmpXX.***

***Replace the XX with your 2 digits at the end of your Oracle id***

1. Execute the following commands.

SET AUTCOMMIT OFF;  
 SET TRANSACTION READ WRITE;

1. Write an INSERT statement to populate the newEmpXX table with the rows of the sample data. Insert the NULL value for the Manager\_id column. (Write a single INSERT statement to insert all the rows)

AND make employee 104 have your last name.

1. Create a query that shows all the inserted rows from the newEmpXX table with how many rows selected.
2. Execute the rollback command. Display all rows and columns from the newEmpXX table. How many rows are selected?
3. Repeat Task 4. Make the insertion permanent to the table newEmpXX. Display all rows and columns from the newEmpXX table. How many rows are selected?
4. Write an update statement to update the value of column jobTitle to ‘unknown’ for all the employees in the newEmpXX table.
5. Make your changes permanent.
6. Execute the rollback command.
   1. Display all employees from the newEmpXX table whose job title is ‘unknown’. How many rows are still updated?
   2. Was the rollback command effective?
   3. What was the difference between the result of the rollback execution from Task 6 and the result of the rollback execution of this task?
7. Begin a new transaction and then create a statement to delete the employees from the newEmpXX table
8. Create a VIEW, called **vwNewEmpXX**, that queries all the records in the newEmpXX table sorted by last name and then by first name.
9. Perform a rollback to undo the deletion of the employees from step 11
   1. How many employees are now in the newEmpXX table?
   2. Was the rollback effective and why?
10. Begin a new transaction and rerun the data insertion from Task 4 (copy the code down to Task 14 and run it)
11. Set a Savepoint, called ***insertion***, after inserting the data
12. Rerun the update statement from Task 8 and run a query to view the data (copy the code down and run it again)
13. Rollback the transaction to the Savepoint created in task 15 above and run a query to view the data.  
    What does the data look like (i.e. describe what happened)?
14. Use the basic form of the rollback statement and again view the data. Describe what the results look like and what happened.

## Part B - Permissions

1. Write a statement that denies all access to the newEmpXX table for all public users
2. Write a statement that allows a classmate (use their database login) read only access to the newEmpXX table.
3. Write a statement that allows the same classmate to modify (insert, update, and delete) the data of the newEmpXX table.
4. Write a statement the denies all access to the newEmpXX table for the same classmate.

## Part C – Clean up

1. Write statements to permanently remove the view and table created for this lab