Experiment 10 - DCL and TCL Statements

A.] DCL:

Query 1.)
This grants the SELECT, INSERT, UPDATE, and DELETE privileges on the customer table to a user with the username username and the password password. The @localhost specifies that the user can only connect from the local machine

Input:
```
GRANT SELECT, INSERT, UPDATE, DELETE
ON customer
TO username@localhost
IDENTIFIED BY 'password';
```

Output: Query OK, 0 rows affected

Query 2.)
This statement revokes the SELECT, INSERT, UPDATE, and DELETE privileges on the customer table from a user with the username username who was previously granted these privileges. The @localhost specifies that the user can only connect from the local machine.

Input:
```
REVOKE SELECT, INSERT, UPDATE, DELETE
ON customer
FROM username@localhost;
```

Output: Query OK, 0 rows affected

Query 3.)
This statement grants the SELECT, INSERT, UPDATE, and DELETE privileges on the customer table to a user with the username username and the password password, and also allows the user to grant these privileges to other users.

Input:
```
GRANT SELECT, INSERT, UPDATE, DELETE
ON customer
TO username@localhost
IDENTIFIED BY 'password'
WITH GRANT OPTION;
```

Output: Query OK, 0 rows affected

Query 4.)

This example modifies the password of a user with the username username who can only connect from the local machine, and sets the new password to newpassword.

Input:

```
ALTER USER username@localhost
IDENTIFIED BY 'newpassword';
```

Output: Query OK, 0 rows affected

B.] TCL:

Query 1.)

The query will update the city value of the customer with a customer_id of 1 to "New York" in the customer table, and the COMMIT statement will commit the changes made in the current transaction, making them permanent.

Input:

```
UPDATE customer
SET city = 'New York'
WHERE customer_id = 1;
COMMIT;
```

Output:1 row(s) updated.
Statement processed.

Query 2.)

This SQL query updates the price of an album with an album_id of 001 to 430, and then rolls back the changes.

Input:

```
UPDATE album
SET price = 430
WHERE album_id = 001;
ROLLBACK;
```

Output:1 row(s) updated.
Statement processed

Query 3.)

This SQL query creates a savepoint named "my_savepoint", and then updates the "phone_no" column in the "customer" table for the customer with a "customer_id" of 2.

Input:

```
SAVEPOINT my_savepoint;
UPDATE customer
```

```
SET phone_no = '555-1234'
WHERE customer_id = 2;
```

Output:1 row(s) updated.
Statement processed

Query 4.)
This SQL query rolls back a transaction to the savepoint named "my_savepoint".

Input:

```
ROLLBACK TO my_savepoint;
```

Query 5.)
This SQL query releases a savepoint named "my_savepoint".

Input:

```
RELEASE SAVEPOINT my_second_savepoint;
```