Week2Lab_User_Management_and_Access_Control_in_PostgreSQL

Exercise 1: Create New Roles and Grant them Relevant Privileges

In PostgreSQL, users, groups, and roles are all the same entity, with the difference being that users can log in by default.

In this exercise, you will create two new roles: `read_only` and `read_write`, then grant them the relevant privileges.

To create a new role named read_only, enter the following command into the CLI:

```
CREATE ROLE read_only;
```

First, this role needs the privilege to connect to the demo database itself. To grant this privilege, enter the following command into the CLI:

```
GRANT CONNECT ON DATABASE demo TO read_only;
```

Next, the role needs to be able to use the schema in use in this database. In our example, this is the bookings schema. Grant the privilege for the read_only role to use the schema by entering the following:

```
GRANT USAGE ON SCHEMA bookings TO read_only;
```

To access the information in tables in a database, the SELECT command is used. For the read_only role, we want it to be able to access the contents of the database but not to edit or alter it. So for this role, only the SELECT privilege is needed. To grant this privilege, enter the following command:

```
GRANT SELECT ON ALL TABLES IN SCHEMA bookings TO read_only;
```

Task B: Create a read_write role and grant it privileges

Similarly, create a new role called read_write with the following command in the PostgreSQL CLI:

```
CREATE ROLE read_write;
```

As in Task A, this role should first be given the privileges to connect to the demo database. Grant this privilege by entering the following command:

```
GRANT CONNECT ON DATABASE demo TO read_write;
```

Give the role the privileges to use the bookings schema that is used in the demo database with the following:

```
GRANT USAGE ON SCHEMA bookings TO read_write;
```

So far the commands for the read_write role have been essentially the same as for the read_only role. However, the read_write role should have the privileges to not only access the contents of the database, but also to create, delete, and modify entries. The corresponding commands for these actions are SELECT, INSERT, DELETE, and UPDATE, respectively. Grant this role these privileges by entering the following command into the CLI:

```
GRANT SELECT, INSERT, DELETE, UPDATE ON ALL TABLES IN SCHEMA bookings TO read_write;
```

Exercise 2: Add a New User and Assign them a Relevant Role

Suppose you wish to add a new user, user_a, for use by an information and help desk at an airport. In this case, assume that there is no need for this user to modify the contents of the database. As you may have guessed, the appropriate role to assign is the read_only role.

To create a new user named user_a, enter the following command into the PostgreSQL CLI:

```
CREATE USER user_a WITH PASSWORD 'user_a_password';
```

In practice, you would enter a secure password in place of 'userapassword', which will be used to access the database through this user.

Next, assign user_a the read_only role by executing the following command in the CLI:

```
GRANT read_only TO user_a;
```

You can list all the roles and users by typing the following command:

```
\du
```

Exercise 3: Revoke and Deny Access

In this exercise, you will learn how to revoke a user's privilege to access specific tables in a database.

Suppose there is no need for the information and help desk at the airport to access information stored in the aircrafts_data table. In this exercise, you will revoke the SELECT privilege on the aircrafts_data table in the demo database from user_a.

You can use the REVOKE command in the Command Line Interface to remove specific privileges from a role or user in PostgreSQL. Enter the following command into the PostgreSQL CLI to remove the privileges to access the aircrafts_data table from user_a:

```
REVOKE SELECT ON aircrafts_data FROM user_a;
```

Now suppose user_a is transferred departments within the airport and no longer needs to be able to access the demo database at all. You can remove all their SELECT privileges by simply revoking the read_only role you assigned to them earlier. You can do this by entering the following command in the CLI:

```
REVOKE read_only FROM user_a;
```