User and Database Management: Takeaways

Syntax

- Connecting to the Postgres server with a secured user:

```
import psycopg2

conn = psycopg2.connect(user="postgres", password="abc123")
```

- Creating a user with a password that can create other users and databases:

```
conn = psycopg2.connect(dbname='dq', user = 'postgres', password='password')

cur = conn.cursor()

cur.execute("CREATE USER data_viewer WITH CREATEUSER CREATEDB PASSWORD 'somepassword'")

conn.commit()
```

- Revoking privileges from a user on a table:

```
conn = psycopg2.connect(dbname="dq", user="dq")

cur = conn.cursor()

cur.execute("REVOKE ALL ON user_accounts FROM data_viewer")

conn.commit()
```

- Grating the privileges to a user on a table:

```
conn = psycopg2.connect(dbname="dq", user="dq")

cur = conn.cursor()

cur.execute("GRANT SELECT ON user_accounts TO data_viewer")

conn.commit()
```

- Creating a group:

```
CREATE GROUP some_group NOLOGIN
```

- Assigning a user to a group:

```
GRANT some_group TO new_user
```

- Creating a database with a specified owner:

```
CREATE DATABASE accounts OWNER postgres
```

Concepts

- A superuser is like an administrator who has full access to the Postgres engine and can issue any command on every database, table, and user. You must enter a password when using a superuser to connect to the Postgres server.
- Privileges are rules that allow a user to run commands such as `SELECT`, `INSERT`, `DELETE TAVLE`. Privileges can either be granted or revoked by the server owner or by database superusers.
- Not revoking certain privileges allow unaware users to issue the wrong command and destroy the entire database.
- The most common practice when creating users is to create them then revoke all privileges, and then choose the privileges you want to grant.

Resources

- User privileges
- Groups