Creating a PostgreSQL Database and User (Modern Best Practices)

After you have successfully installed PostgreSQL on your server (e.g., your Hostinger VPS running Ubuntu), follow these steps. This guide represents the standard, secure procedure for setting up a new database for any web application.

Step 1: Access the PostgreSQL Superuser Shell

PostgreSQL administration is managed through a special Linux user account called postgres, which is created during installation. To perform administrative tasks, you must first switch to this user.

Open your server's terminal and run:

```
sudo -i -u postgres
```

This command logs you in as the postgres user. Your command prompt will change to postgres@your-server-name:~$.

Next, launch the PostgreSQL interactive terminal, psql. This is the primary tool for interacting with your databases.

```
psql
```

You are now inside the psql shell, indicated by the postgres=# prompt. All subsequent commands until you exit are SQL commands.

Step 2: Create a Dedicated, Secure User for Your App

It is a critical security practice to create a unique user for each application. Your backend will use this user's credentials to connect. **Never use the postgres superuser for your application.** This follows the "Principle of Least Privilege," ensuring your app only has access to its own database.

Let's create a user named dentist_admin.

Important: Replace YourVeryStrong!Password123 with a long, unique, and randomly generated password. Use a password manager to generate and store this.

```
CREATE USER dentist_admin WITH PASSWORD 'YourVeryStrong!Password123';
```

PostgreSQL should respond with CREATE ROLE. (USER is an alias for ROLE with login permissions).

Step 3: Create the Application's Database

Now, create the database itself. Let's call it dentist_db. By specifying the OWNER, we link the database to the user we just created. This simplifies permission management.

CREATE DATABASE praxis_db OWNER emad_admin;

You should see the response CREATE DATABASE.

Step 4: Grant All Necessary Privileges

While dentist_admin owns the database, we must explicitly grant it the permissions to create tables, write data, and perform all operations within it.

GRANT ALL PRIVILEGES ON DATABASE praxis_db TO emad_admin;

PostgreSQL will respond with GRANT. This gives your application's user full control over the dentist_db database, but no other databases on the server.

Step 5: Exit and Verify the Connection

You have successfully set up the user and database. Exit the psql shell by typing:

\q

This returns you to the postgres user's Linux prompt. Type exit to return to your normal server user.

To verify that everything works, try connecting directly to your new database as your new user:

This command attempts to connect to the 'dentist_db' database as the 'dentist_admin' user.
The '-h localhost' flag specifies connecting to the server on this machine.
psql -h localhost -d dentist_db -U dentist_admin

It will prompt for the password you set in Step 2. If successful, your prompt will

change to dentist_db=>. This confirms that your credentials and permissions are correct. Type \q to exit.

Using This in Your Backend Application

In your Node.js backend's .env file, you will use the following connection string. This string contains all the information your application needs to connect.

```
# PostgreSQL Connection URL
# Format: postgresql://USER:PASSWORD@HOST:PORT/DATABASE_NAME
DATABASE_URL="postgresql://dentist_admin:YourVeryStrong!Password123@localhost:5432/dentist_db"
```

- **localhost**: This works because your application and database are running on the same VPS.
- **5432**: This is the default port for PostgreSQL.

Your backend will read this URL from the environment variables to establish a secure connection.