**Import Your Database Schema**

If you already have a local database, export the schema and data, then import it into your managed database.

**Export from Local Database**

1. Use pg\_dump to export your local database:

pg\_dump -U postgres -h localhost -d your\_database\_name -F c -b -v -f backup.dump

**Import into Managed Database**

1. Use pg\_restore to import the dump file into your managed database:

pg\_restore -U postgres -h db.your\_project\_id.supabase.co -d postgres -v backup.dump

**Update Environment Variables**

In your hosting platform (e.g., Render, Railway, Heroku), add the **connection string** as an environment variable:

* Key: DATABASE\_URL
* Value: postgresql://postgres:your\_password@db.your\_project\_id.supabase.co:5432/postgres

**STEP 1: Step-by-step guide** to connect to your Render database using psql and query the users table.

**Gather Your Render Database Credentials**

You’ll need the following details to connect to your Render database:

* **Host**: dpg-cv22338gph6c73bbscv0-a.singapore-postgres.render.com
* **Port**: 5432 (default PostgreSQL port)
* **Database Name**: tourism\_db\_9g5v
* **Username**: tourism\_db\_9g5v\_user
* **Password**: Your Render database password.

**STEP 2: Connect to Your Render Database**

Open your terminal and run the following psql command:

psql -h dpg-cv22338gph6c73bbscv0-a.singapore-postgres.render.com -p 5432 -U tourism\_db\_9g5v\_user -d tourism\_db\_9g5v

* **-h**: Specifies the host.
* **-p**: Specifies the port (default is 5432).
* **-U**: Specifies the username.
* **-d**: Specifies the database name.

After running the command, you’ll be prompted to enter the password for the tourism\_db\_9g5v\_user account. Enter the password and press **Enter**.

**STEP 3: Verify the Connection**

If the connection is successful, you’ll see a prompt like this:

psql (14.5, server 13.4)

SSL connection (protocol: TLSv1.3, cipher: TLS\_AES\_256\_GCM\_SHA384, bits: 256, compression: off)

Type "help" for help.

tourism\_db\_9g5v=>

This means you’re now connected to your Render database.

**STEP 4: Query the users Table**

Now that you’re connected, you can run SQL queries to interact with the users table.

**1. Check if the users table exists**

SELECT \* FROM users;

**STEP 5: Check the schema of the users table**

\d users

This will display the table structure, including column names, data types, and constraints.

**STEP 6: Common psql Commands**

Here are some useful psql commands to help you navigate and interact with your database:

| **Command** | **Description** |
| --- | --- |
| \l | List all databases. |
| \c <database\_name> | Connect to a different database. |
| \dt | List all tables in the current database. |
| \d <table\_name> | Describe the structure of a table. |
| \q | Quit psql and return to the terminal. |
| \? | Show help for psql commands. |
| \x | Toggle expanded output (useful for wide tables). |
| \timing | Toggle timing of SQL commands. |

**STEP 7: Disconnect from psql**

To exit psql, simply type:

\q

PROMT: I successfully deployed the Client side and the whole system works including the database, but the only problem is that when I refresh the link, it shows "Not Found"

ANSWER:

The issue you're experiencing is common in **Single Page Applications (SPAs)** like React apps when deployed to a static hosting service. This happens because the server doesn't know how to handle client-side routing (e.g., React Router routes) when you refresh the page or directly access a deep link (e.g., /user/dashboard).

**Why This Happens**

* When you refresh the page or directly access a route like /user/dashboard, the server looks for a file or resource at that path (e.g., user/dashboard/index.html).
* Since your React app is a SPA, all routing is handled on the client side, and there’s no actual file at /user/dashboard. This results in a **404 Not Found** error.

**Solution**

To fix this, you need to configure your server to serve the index.html file for all routes. This way, React Router can handle the routing on the client side.

**Solution**

To fix this, you need to configure your server to serve the index.html file for all routes. This way, React Router can handle the routing on the client side.

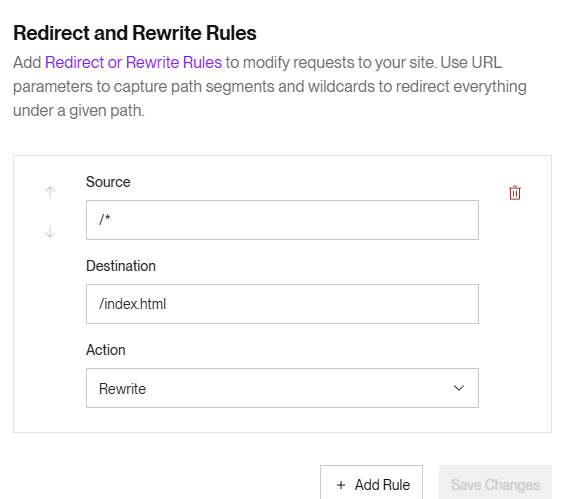
**Step-by-Step Fix**

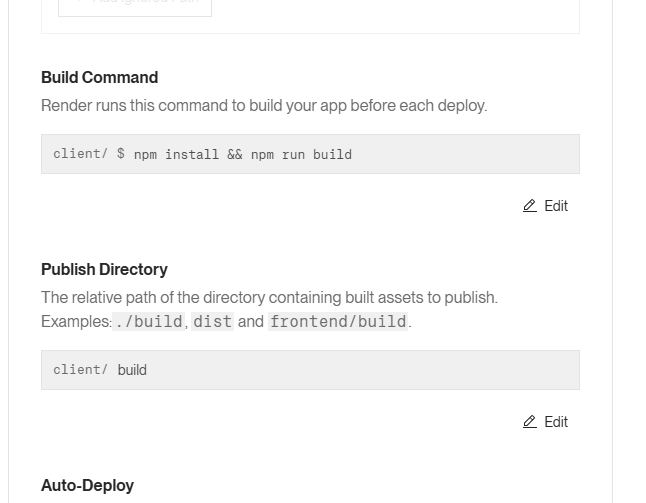
**1. If You're Using Render**

Render allows you to configure custom routes for static sites. Here's how to fix it:

1. Go to your **Render Dashboard**.
2. Select your static site (React app).
3. Go to **Settings > Redirects/Rewrites**.
4. Add a rewrite rule:
   * **Source**: /\*
   * **Destination**: /index.html
   * **Action**: Rewrite
5. Save the changes.

This tells Render to serve index.html for all routes, allowing React Router to handle the routing.





REACT\_APP\_ACCESS\_CODE=panglaotourism@TDMS2025

REACT\_APP\_API\_BASE\_URL=https://tdms-panglao-v1.onrender.com

-- Predefined Admin

INSERT INTO users (username, email, password, role, is\_approved)

VALUES ('TDMS Admin', 'statisticstourismpanglao@gmail.com',

        '$2b$10$MjoygsGSlw.3JSrEqA300.X/Sv.Tv1OZPEdC8pmrAniGrQpPbONia',

        'admin', true);

ALTER TABLE submissions ADD COLUMN number\_of\_rooms INT;

psql "postgresql://tourism\_db\_9g5v\_user:your\_password@dpg-cv22338gph6c73bbscv0-a.singapore-postgres.render.com:5432/tourism\_db\_9g5v?sslmode=require"

psql "postgresql://tourism\_db\_9g5v\_user: am56xLw36KkhUPc2GDqYf0TAhBv06j0n@dpg-cv22338gph6c73bbscv0-a.singapore-postgres.render.com:5432/tourism\_db\_9g5v?sslmode=require"

psql "postgresql://tourism\_db\_9g5v\_user:am56xLw36KkhUPc2GDqYf0TAhBv06j0n@dpg-cv22338gph6c73bbscv0-a.singapore-postgres.render.com: 5432/tourism\_db\_9g5v?sslmode=require"

//First database I use deployed to render

DATABASE\_URL=postgresql://tourism\_db\_9g5v\_user:am56xLw36KkhUPc2GDqYf0TAhBv06j0n@dpg-cv22338gph6c73bbscv0-a/tourism\_db\_9g5v

psql "postgresql://tourism\_db\_qfsw\_user:fZuR14PwbRgZk05f6EcnXCnDgXz1Mo7A@dpg-cvmgs4nfte5s739i37jg-a.oregon-postgres.render.com:5432/tourism\_db\_qfsw?sslmode=require"

APP PASSWORD = https://myaccount.google.com/apppasswords