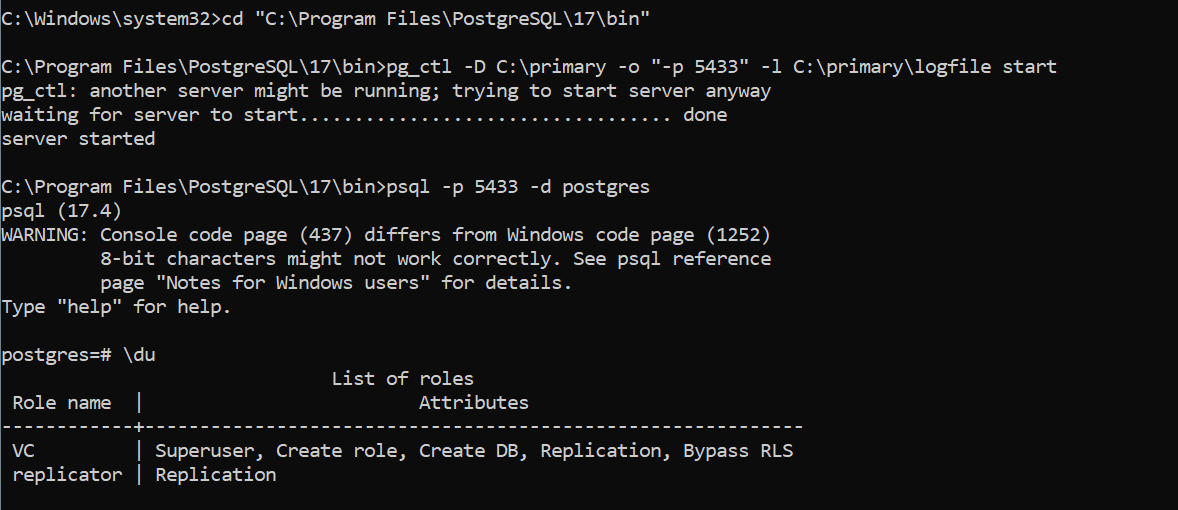
**GRANTING AND REVOKING PERMISSIONS IN POSTGRE SQL**



**Step 1: Create a Role (User)**

CREATE ROLE readonly LOGIN PASSWORD '1234';



This creates a **login-enabled role (user)** named readonly with the password '1234'

**Purpose**: You’ll use this role to test permission controls (e.g., can/can’t read/insert data).

**Step 2: Create a New Database**

CREATE DATABASE dbsample;

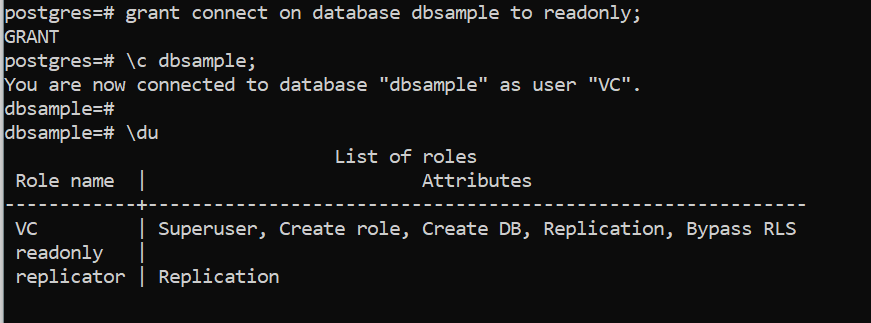


A new database named dbsample is created.

You’ll use this to test role privileges on tables within this database.

**Step 3: Allow the User to Connect to the Database**

GRANT CONNECT ON DATABASE dbsample TO readonly;



This **grants the ability to connect** to the database dbsample for the readonly user.

**Step 4: Grant Privileges on the public Schema**

GRANT USAGE, CREATE ON SCHEMA public TO readonly;



This allows readonly to:

* **USE** the schema (USAGE) → needed for resolving table names
* **CREATE** objects in the schema (CREATE) → e.g., tables, views, etc.

**Step 5: Grant Table-Level Permissions**

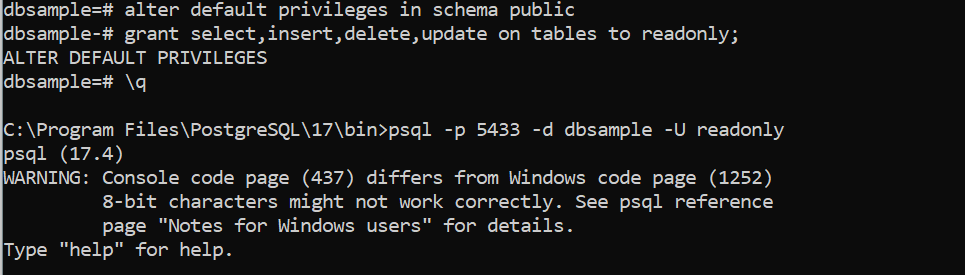
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO readonly;



This gives full access to **read and modify** existing tables:

* SELECT: View table data
* INSERT: Add new rows
* UPDATE: Modify existing rows
* DELETE: Remove rows

**Step 6: Set Default Privileges for Future Tables**



ALTER DEFAULT PRIVILEGES IN SCHEMA public

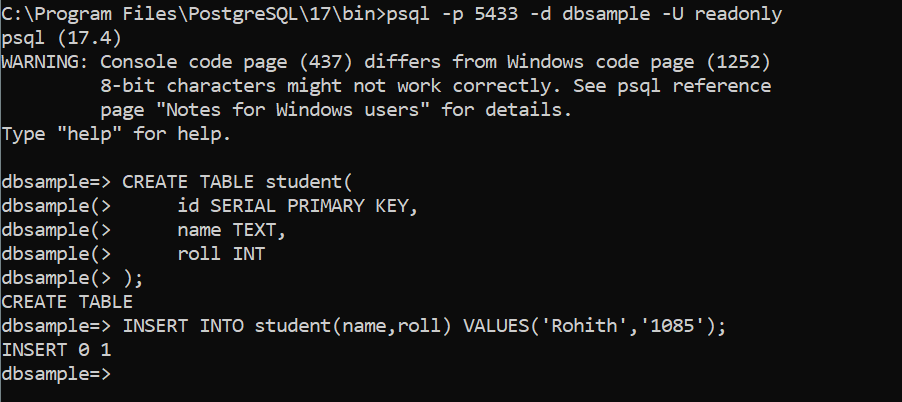
GRANT SELECT, INSERT, DELETE, UPDATE ON TABLES TO readonly;

This ensures that any **new table** created in the public schema will automatically have these permissions granted to readonly.

**Outcome So Far**

At this point, the readonly user can:

* Connect to dbsample
* Access the public schema
* Insert/select/update/delete from tables that exist now and those created in the future

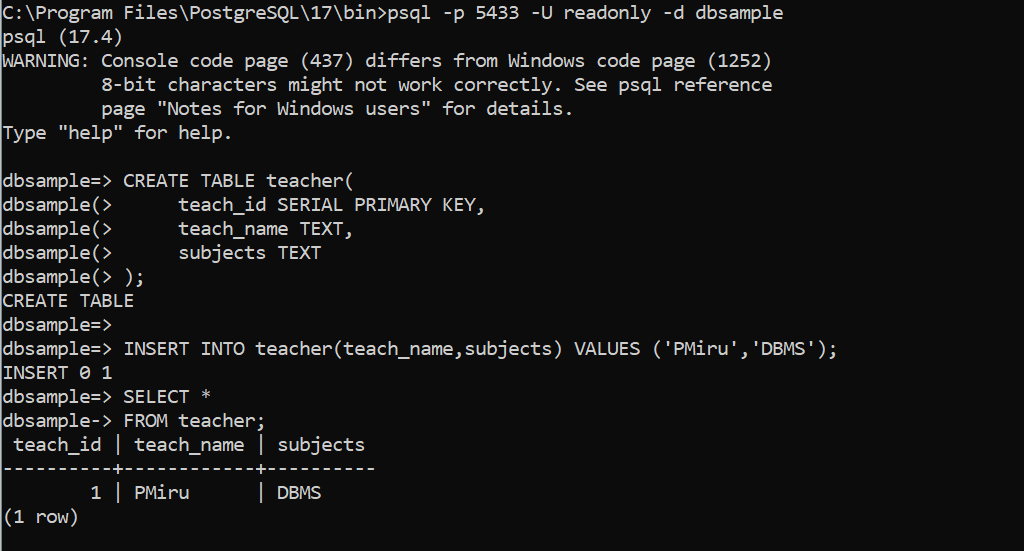


**Step 7: Create and Use Tables**

Then:

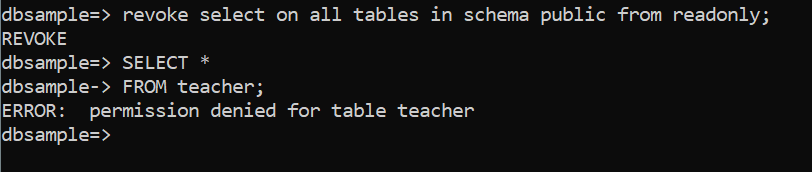
1. **Logged in as readonly** user to the database
2. Created student and teacher tables
3. Inserted values and retrieved them successfully using SELECT \* FROM teacher;

This shows that readonly **had full table access**.



**Step 8: Revoke Access**

REVOKE SELECT ON ALL TABLES IN SCHEMA public FROM readonly;



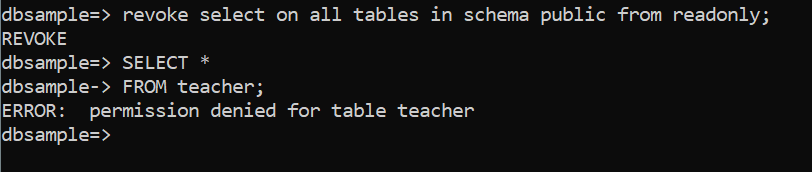
This revokes the SELECT permission only, meaning:

* The user can no longer read (SELECT) data from any tables
* But they may still INSERT, UPDATE, or DELETE depending on whether those privileges are still granted

**Verification: Denied SELECT**

SELECT \* FROM teacher;

-- ERROR: permission denied for table teacher



This confirms that readonly **can no longer view table data** after the SELECT privilege was revoked.

**Summary of PostgreSQL Permission Keywords**

| **Keyword** | **Meaning** |
| --- | --- |
| CONNECT | Permission to connect to a database |
| USAGE | Permission to use a schema |
| CREATE | Permission to create objects in schema |
| SELECT | Read data from tables |
| INSERT | Add new rows to tables |
| UPDATE | Modify existing rows |
| DELETE | Remove rows |
|  |  |

**Final Commands for Documentation**

-- 1. Create role

CREATE ROLE readonly LOGIN PASSWORD '1234';

-- 2. Create database

CREATE DATABASE dbsample;

-- 3. Grant database connection

GRANT CONNECT ON DATABASE dbsample TO readonly;

-- 4. Grant schema access

GRANT USAGE, CREATE ON SCHEMA public TO readonly;

-- 5. Grant table access

GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO readonly;

-- 6. Set default privileges for future tables

ALTER DEFAULT PRIVILEGES IN SCHEMA public

GRANT SELECT, INSERT, DELETE, UPDATE ON TABLES TO readonly;

-- 7. Revoke SELECT permission

REVOKE SELECT ON ALL TABLES IN SCHEMA public FROM readonly;