

**dba.SetPassword (technical page)**

**Purpose:**

The dba.SetPassword function is a secure way to change a PostgreSQL user’s password, with built-in auditing and safety features. It is designed for use in environments where tracking changes and avoiding accidental modifications is critical.

**🔧 What It Does**

- Changes the password for a non-superuser PostgreSQL role.  
- Logs the change to an audit table.  
- Performs input validation to avoid common security issues or user errors.

**🔐 Security and Validation**

This function includes safeguards such as:  
- Disallows NULL or blank values for username/password.  
- Requires password to be at least 12 characters long.  
- Blocks changes to the "postgres" superuser.  
- Only changes password if the user exists.

**📥 Usage Syntax**

***SELECT dba.SetPassword('target\_username', 'new\_secure\_password');***

***✅ Example:  
SELECT dba.SetPassword('employee123', 'MyN3wSecureP@ssword!');***

**🧾 What Gets Logged**

Every successful password change is logged into the dba.password\_change\_audit table with:  
- Who changed it (changed\_by)  
- Whose password was changed (changed\_user)  
- Timestamp (changed\_at)  
- Optional note field (currently hardcoded)

**❓ Troubleshooting**

|  |  |
| --- | --- |
| Error Message | Meaning |
| Username cannot be null or empty | You didn’t pass a valid username. |
| Password must be at least 12 characters long | Password is too short. |
| Changing password for superuser "postgres" is not allowed | Security policy restriction. |
| User "xyz" does not exist | PostgreSQL role not found. |

**🧪 Pro Tip**

***To test safely, create a dummy user first:  
CREATE ROLE test\_user LOGIN;  
SELECT dba.SetPassword('test\_user', 'DummyP@ssword123');***

**🛠️ Technical Information**

**⚠️ Important Notes**

- Cannot change the "postgres" account password. (Built-in security control)  
- You must have appropriate privileges to run this function.  
- The function runs as a SECURITY DEFINER, meaning it executes with the privileges of its creator—make sure only trusted users can call it.

**🔄 How It Works (Behind the Scenes)**

1. Schema Setup: Ensures the dba schema exists.  
2. Audit Table: Creates the password\_change\_audit table if it doesn't exist.  
3. Function Logic:  
 - Validates inputs.  
 - Checks user existence.  
 - Executes ALTER ROLE ... WITH PASSWORD.  
 - Logs the action in the audit table.

**✅ Setup Checklist (One-Time)**

Make sure these steps are done first (included in the .sql file):

* - [x] dba schema exists.
* - [x] Ensure groups have access to the schema.
* - [x] Ensure the group has execute rights on the function
* - [x] dba.password\_change\_audit table is created.
* - [x] dba.SetPassword function is deployed.

**🧹 Optional Cleanup**

***If you need to update or remove the function:  
DROP FUNCTION dba.SetPassword(text, text);***