

In **PostgreSQL**, **DROP ROLE** is used to **remove (delete) a role/user from the system**.

---

## ◆ What is **DROP ROLE**?

---

👉 It is a command used to:

- Delete a **role (user or group)**
  - Remove its access to the database system
- 

## ◆ Why Do We Use **DROP ROLE**?

---

### ✓ 1. Remove Unused Users

- When a user leaves (e.g., student graduates, employee resigns)

### ✓ 2. Improve Security

- Remove unnecessary accounts to prevent misuse

### ✓ 3. Clean Database System

- Keep roles organized and manageable
- 

## ◆ Basic Syntax

---

```
DROP ROLE role_name;
```

---

## ◆ Simple Example (Beginner)

---

Step 1: Create a Role

```
CREATE ROLE test_user WITH LOGIN PASSWORD '123';
```

Step 2: Drop the Role

```
DROP ROLE test_user;
```

✓ The role is now removed from PostgreSQL

---

## ⚠ Important Rules (Very Important)

---

You **CANNOT drop a role if it:**

- ✗ 1. Owns a database

Example:

```
ALTER DATABASE dvdrental OWNER TO test_user;
```

Then:

```
DROP ROLE test_user; -- ✗ ERROR
```

---

- ✗ 2. Owns tables or objects
  - ✗ 3. Has active connections
- 

## ◆ How to Safely Drop a Role

---

Step 1: Reassign ownership

```
REASSIGN OWNED BY test_user TO postgres;
```

Step 2: Remove permissions

```
DROP OWNED BY test_user;
```

Step 3: Drop role

```
DROP ROLE test_user;
```

---

## ◆ Example with dvdrental

---

```
-- Create role  
CREATE ROLE trainee WITH LOGIN PASSWORD '123';  
  
-- Give access
```

```
GRANT CONNECT ON DATABASE dvdrental TO trainee;

-- Remove role safely
REASSIGN OWNED BY trainee TO postgres;
DROP OWNED BY trainee;
DROP ROLE trainee;
```

---

## Key Concept

---

👉 A role cannot be deleted if it is still **linked to database objects**

---

## Teaching Tip

---

Ask students: 👉 "Why did DROP ROLE fail?"

This helps them understand:

- Ownership
  - Dependencies
  - Security concepts
- 

## Bonus (Check Roles Before Deleting)

---

```
SELECT rolname FROM pg_roles;
```

---

## Lab: Understanding DROP ROLE in PostgreSQL

### Learning Objectives

Students will learn:

- How to safely delete roles
  - Why **DROP ROLE** may fail
  - How to resolve dependency errors
  - Difference between **ownership and permissions**
- 

### ◆ Scenario

You are managing a system where temporary users (trainees, interns, students) must be removed after use.

---

### ◆ Step 1: Create a Test Role

```
CREATE ROLE trainee WITH LOGIN PASSWORD '123';
```

Verify:

```
SELECT rolname FROM pg_roles;
```

---

### ◆ Step 2: Drop the Role (Basic Case)

```
DROP ROLE trainee;
```

✓ Works because:

- No ownership
- No permissions
- No dependencies

---

### ◆ Step 3: Create Role Again

```
CREATE ROLE trainee WITH LOGIN PASSWORD '123';
```

---

### ◆ Step 4: Assign Permissions (dvdrental)

```
GRANT CONNECT ON DATABASE dvdrental TO trainee;
```

Try dropping:

```
DROP ROLE trainee;
```

👉 ? Ask students:

- Does it work or fail?

✓ Usually it **fails** due to dependencies

---

### ◆ Step 5: Fix the Error

---

```
DROP OWNED BY trainee;  
DROP ROLE trainee;
```

---

## Critical Learning Section

---

### ◆ Step 6: Ownership Problem (Very Important)

#### Create role again:

```
CREATE ROLE trainee WITH LOGIN PASSWORD '123';
```

#### Assign ownership:

```
ALTER DATABASE dvdrental OWNER TO trainee;
```

Now try:

```
DROP ROLE trainee;
```

✗ This will FAIL

---

## Why It Fails?

---

👉 Because:

- Role **owns the database**
- PostgreSQL prevents deletion to avoid data loss

---

## ◆ Step 7: Resolve Ownership Issue

---

```
REASSIGN OWNED BY trainee TO postgres;
```

Then:

```
DROP OWNED BY trainee;
```

**“Remove everything that this role (trainee) owns in the current database.”**

This includes:

- Tables
- Views
- Sequences
- Functions
- Permissions (grants)

**Finally:**

```
DROP ROLE trainee;
```

✓ Now it works

---

## Step 8: Investigate Ownership

---

```
SELECT datname, pg_catalog.pg_get_userbyid(datdba) AS owner  
FROM pg_database;
```

---

## Student Tasks

---

### ✓ Task 1: Basic Deletion

- Create role `temp_user`
  - Drop it immediately
- 

### ✓ Task 2: Permission Dependency

- Create role `intern`
  - Grant SELECT on a table in `dvdrental`
  - Try dropping → observe error
  - Fix and drop
- 

### ✓ Task 3: Ownership Dependency

- Create role `manager`
  - Assign database ownership
  - Try dropping → observe error
  - Fix using `REASSIGN OWNED`
-

## ✓ Task 4: Table Ownership Test

```
ALTER TABLE film OWNER TO trainee;
```

Try:

```
DROP ROLE trainee;
```

👉 Fix it properly

---

## Key Concepts Summary

---

Concept	Explanation
DROP ROLE	Deletes a user/role
Dependency	Role linked to objects
REASSIGN OWNED	Transfer ownership
DROP OWNED	Remove permissions
Ownership	Strongest control

---

## Common Errors (Teach Students)

---

Error	Reason
cannot drop role	still owns objects
permission denied	not superuser
role is in use	active connections

---