In **PostgreSQL**, a **role** is a central concept for managing **users and permissions**.

◇ What is a Role?

A **role** is an entity that can:

- Own database objects (tables, views, etc.)
- Have permissions (SELECT, INSERT, etc.)
- Login into the database (if allowed)

👉 In simple words: **Role = User + Permissions system**

◇ Types of Roles

There are mainly two types:

1. Login Role (User)

- Can log in to the database
- Example: `student1`, `admin`

2. Group Role

- Cannot log in
- Used to **assign permissions to multiple users**

◇ Why We Use Roles?

Roles are used for **security and management**:

☑ 1. Access Control

- Restrict who can access data

☑ 2. Permission Management

- Assign permissions once to a role, then assign role to many users

☑ 3. Better Organization

- Separate admin, teacher, student roles

☑ 4. Data Security

- Prevent unauthorized changes

◇ Syntax: Create Role

Basic Syntax:

```
CREATE ROLE role_name;
```

◇ Create Role with Login (User)

```
CREATE ROLE student1
WITH LOGIN
PASSWORD '1234';
```

◇ Create Role with Common Options

```
CREATE ROLE teacher
WITH
LOGIN
PASSWORD 'securepass'
CREATEDB
CREATEROLE;
```

◇ Important Options

Option	Meaning
LOGIN	Can log in (makes it a user)
PASSWORD	Set password
CREATEDB	Can create databases
CREATEROLE	Can create roles
SUPERUSER	Full access (admin)
NOSUPERUSER	Normal user

◇ Example

```
CREATE ROLE admin_user
WITH LOGIN
PASSWORD 'admin123'
SUPERUSER;
```

◇ Grant Role to Another User

```
GRANT teacher TO student1;
```

🧪 Lab: Understanding Roles using dvdrental Database

🎯 Learning Outcomes

Students will:

- Understand what a **role** is
- Create roles with different privileges
- View and analyze roles
- Change database ownership
- Understand **ownership vs permissions**

◇ Step 1: Create a New Role (User)

Before assigning ownership, the role must exist.

```
CREATE ROLE senior_admin
WITH LOGIN
PASSWORD 'secure_pass123';
```

🧠 Concept:

- `LOGIN` → makes it a **user**
- Without `LOGIN` → it is just a **group role**

👉 **Key Idea:** In PostgreSQL, *there is no separate "user" concept* — everything is a **role**

◇ Step 2: View All Roles

```
SELECT rolname FROM pg_roles;
```

✔ This retrieves all roles in the server.

🔍 Understanding `pg_roles`

- It is a **system catalog view**
- Stores:
  - Users
  - Groups
  - Privileges
  - Role attributes

📊 Better Query (Detailed View)

```
SELECT
    rolname,
    rolsuper,
    rolcreaterole,
    rolcreatedb,
    rolcanlogin
FROM pg_roles;
```

🧠 Column Understanding:

Column	Meaning
rolname	Role name
rolsuper	Superuser or not
rolcreaterole	Can create roles
rolcreatedb	Can create DB
rolcanlogin	Can login

🎯 Filter Only Users

```sql
SELECT rolname
FROM pg_roles
WHERE rolcanlogin = true;
```

💻 Alternative (psql)

```
\du
```

◇ Step 3: Assign Ownership of dvdrental

```sql
ALTER DATABASE dvdrental OWNER TO senior_admin;
```

🧠 Concept: Ownership

Ownership is **very powerful**:

- Owner can:
  - Drop database
  - Rename database
  - Grant permissions

👉 Ownership ≠ Normal permissions

❗ Database Ownership vs Table Ownership

After running:

```sql
ALTER DATABASE dvdrental OWNER TO senior_admin;
```

✔ Only the **database owner changes**

❌ Tables like:

- `film`
- `actor`
- `customer`

👉 **DO NOT change owner automatically**

◇ Step 4: Verify Ownership

In terminal:

```
\l
```

Expected Output:

Database	Owner
dvdrental	senior_admin

⚠ Permissions Requirement

Only:

- Superuser
- OR current database owner

👉 can run `ALTER DATABASE`

Otherwise: ❌ `permission denied`