In-Class Assessment: SQL Views and Roles (15 minutes)

Scenario

You are managing the database for a consulting firm. Different departments need specific views of the data, and you must implement proper security controls to ensure each team sees only what they need.

Prerequisites

Use the existing `employees`, `departments`, and `projects` tables from Lab 6.

Part A: Basic View Creation (5 minutes)

Task 1: Employee Directory View (2.5 minutes)

Create a view named `employee_directory` that shows:

- Employee name
- Department name
- Department location
- Salary
- A column `status` showing:
  - 'High Earner' if salary > 55000
  - 'Standard' otherwise

Requirements:

- Include only employees who are assigned to a department
- Order by department name, then employee name

Test your view:

sql
```
SELECT * FROM employee_directory ORDER BY dept_name;
```

Task 2: Project Summary View (2.5 minutes)

Create a view named `project_summary` that displays:

- Project name
- Budget

- Department name
- Location
- A column `project_size` showing:
  - 'Large' if budget > 80000
  - 'Medium' if budget > 50000
  - 'Small' otherwise

Requirements:

- Include all projects with their departments

Test your view:

sql
```
SELECT * FROM project_summary WHERE project_size = 'Large';
```

Part B: View Modifications (3 minutes)

Task 3: Update and Replace Views (3 minutes)

Step 1: Modify the `employee_directory` view to add one more column:

- `dept_category` showing:
  - 'Technical' if department name contains 'IT' or 'Development'
  - 'Non-Technical' otherwise

Hint: Use `CREATE OR REPLACE VIEW` and the `LIKE` or `ILIKE` operator

Step 2: Rename the `project_summary` view to `project_overview`:

Step 3: Drop the view named `project_overview`:

Part C: Materialized Views (3 minutes)

Task 4: Create and Refresh Materialized View (3 minutes)

Step 1: Create a materialized view named `dept_summary` showing:

- Department name
- Number of employees (use COUNT)
- Number of projects (use COUNT)
- Total project budget (use SUM, show 0 if NULL)

Requirements:

- Create WITH DATA

- Include all departments even if they have no employees or projects

Step 2: Test the refresh process:

1. Insert a new project:

sql
```
INSERT INTO projects (proj_id, proj_name, budget, dept_id)
VALUES (105, 'Security Audit', 45000, 103);
```

2. Query the materialized view:

sql
```
SELECT * FROM dept_summary WHERE dept_name = ‘Operations';
```

3. Refresh the materialized view:

4. Query again:

sql
```
SELECT * FROM dept_summary WHERE dept_name = 'Operations';
```

Part D: Role-Based Access Control (4 minutes)

Task 5: Create Roles and Users (4 minutes)

Step 1 - Create Basic Roles (1.5 minutes):

1. Create role `viewer_role` (no login) with:
   - SELECT privilege on `employee_directory` view
   - SELECT privilege on `departments` table
2. Create role `editor_role` (no login) with:
   - SELECT privilege on ALL three tables (employees, departments, projects)
   - INSERT privilege on `employees` table
   - UPDATE privilege on `employees` table

Step 2 - Create Role Hierarchy (1 minute):

Create a parent role `manager_role` (no login) that:

- Has membership in `editor_role` (inherits all editor privileges)
- Has additional DELETE privilege on `employees` table
- Has additional UPDATE privilege on `projects` table

Step 3 - Create Users (1.5 minutes):

Create the following users:

1. `alice_viewer` with password `'view123'` → assign to `viewer_role`
2. `bob_editor` with password `'edit456'` → assign to `editor_role`
3. `carol_manager` with password `'mgr789'` → assign to `manager_role`

Verification:

sql

```sql
-- Check roles and memberships
SELECT rolname, rolcanlogin FROM pg_roles
WHERE rolname LIKE '%_role' OR rolname LIKE '%_viewer'
  OR rolname LIKE '%_editor' OR rolname LIKE '%_manager'
ORDER BY rolname;
```