ROLE-BASED AUTHENTICATION

Objective:
Implement role-based authentication for Ireland DB by identifying and granting admin privileges only to the necessary users, and removing it from others.

Requirements:
1. Access to the Ireland DB instance: at least one superuser or root-level access.
2. List of current users and their roles: for auditing and validation.
3. Stakeholders or team leads: verifying user roles and validating admin needs.
4. Policy or standards: any security guidelines or company policy for database access control.

Plan:
pg_roles is a predefined system catalog view.
It is available in every PostgreSQL database by default.
It’s a read-only view, so you can query it but not modify it directly.

Column Name	**Description**
rolname	Name of the role (user/group)
rolsuper	Boolean - Is the role a superuser?
rolcreaterole	Can the role create other roles?
rolcreatedb	Can the role create databases?
rolcanlogin	Can the role log in?
rolreplication	Can the role initiate replication?
rolbypassrls	Can bypass row-level security policies?

1. Export Current User Roles:
   We need a baseline to know who currently has access and assess if it's appropriate. Query the Ireland DB to get a list of all users and their current roles (especially those with admin access).

   ```
   -- List all roles and their attributes
   SELECT
       rolname AS role_name,
       rolsuper AS is_superuser,
       rolcreaterole AS can_create_roles,
       rolcreatedb AS can_create_db,
       rolreplication AS replication,
       rolbypassrls AS bypass_rls
   FROM pg_roles
   ORDER BY rolname;
   ```

   One way, PgAdmin, after getting results from the query tool, click download.

2. Prepare a Stakeholder Communication Plan:
   People might have admin access for legacy reasons. Direct communication helps validate necessity, avoids confusion, and promotes transparency.
   Notify team leads and users that a role-based access cleanup is in progress. Share the current list with them.
   Make use of mail and normal messages.

3. Collect Admin Requests:
   Ask all users to request admin access if required formally. Include a form or standardized format and convert it into spreadsheet for validation.
   1. Full Name: full legal name
   2. Email / Username: database login ID or official email
   3. Team / Department: e.g., DevOps, Backend, Analytics
   4. Reason for Admin Access: Why do you need admin-level access? Be specific (e.g., DB schema changes, performance tuning, user management, etc.)
   5. Access Duration:
   ☐ Permanent
   ☐ Temporary
   If temporary, specify the end date: __________
   6. Approved By (Team Lead or Manager): Full name and email of the approver
   7. Additional Comments (Optional): Any other relevant info

4. Analyze Requests & Compare With Existing Admins (in Excel):
   - Compare the submitted requests with the current admin users. (Step 1 & 3)
   - Mark users who no longer need admin rights. (Revoke Admin)
   - Mark users who newly requested it and are not on the current list. (Keep Admin, Grant Admin)

5. Validate Requests With Team Leads/Managers:
   Review all requests and role decisions with relevant stakeholders (tech leads, managers). Prevents unauthorized or overprivileged access and provides accountability.

6. Update Roles in the Ireland DB:
   Revoke admin privileges from users not requiring them. Grant admin privileges to approved users only.
   To enforce least-privilege access and reduce security risks.
   ```
   -- Revoke admin:
   REVOKE ALL PRIVILEGES ON DATABASE ireland_db FROM user_xyz;
   -- Grant admin (as per system setup):
   GRANT ALL PRIVILEGES ON DATABASE ireland_db TO user_abc;
   – Grant admin rights to user 'user_abc' only if necessary:
   ALTER ROLE user_abc WITH SUPERUSER CREATEDB CREATEROLE;
   ```

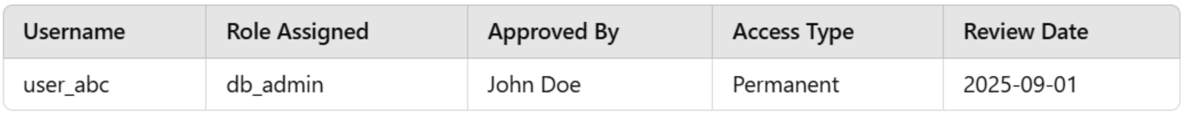
7. Implement Role-Based Groups (Recommended):
   Create roles/groups like db_admin, read_only, analyst, etc., and assign users to those.
   Easier to manage access levels without updating individual users every time.

Create roles:
```
-- Admin Role
CREATE ROLE db_admin NOINHERIT;
-- Read-only Role
CREATE ROLE db_readonly NOINHERIT;
-- Grant privileges to the roles
GRANT ALL PRIVILEGES ON DATABASE ireland_db TO db_admin;
GRANT CONNECT ON DATABASE ireland_db TO db_readonly;
```

Assign Users to Roles:
```
-- Assign user to role
GRANT db_admin TO user_abc;
GRANT db_readonly TO user_xyz;
```

8. Document Role Assignments
   Maintain a record of:
   - Who has what role
   - Approval timestamps
   - Review dates

   For audit trails and future access reviews.



Username	Role Assigned	Approved By	Access Type	Review Date
user_abc	db_admin	John Doe	Permanent	2025-09-01

9. Schedule Regular Access Reviews
   Set up a quarterly review of all user roles: every 3 or 6 months.

10. Monitor & Audit Log Access
    Enable logging and monitor who is accessing what, especially for users with admin roles (postgresql.conf).
```
logging_collector = on
log_connections = on
log_disconnections = on
log_statement = 'ddl'
log_line_prefix = '%t [%p]: [%l-1] user=%u,db=%d,app=%a,client=%h '
```