**Task 1: Figure out what security precautions are already used in your dvd\_rental database. Prepare description**

*This is an analysis of the security precautions in the dvd\_rental database, focusing on roles, privileges granted to specific roles/users, and table-specific access controls.*

**1. Roles and Their Properties**

To identify all roles in the database, the following query is executed against the pg\_roles system catalog:

SELECT rolname, rolsuper, rolcanlogin, rolcreatedb, rolcreaterole

FROM pg\_roles;

**Findings**:

* **postgres**: A superuser role (rolsuper = true, rolcanlogin = true) with full privileges on all database objects. This is the default administrative role created during PostgreSQL installation.
* **public**: A default role with no inherent privileges unless explicitly granted. All users inherit this role implicitly.
* No additional roles (dvd\_rental\_user) are defined by default in the dvd\_rental database unless added by the application.

**Recommendation:**

* Create application-specific roles with limited privileges (like SELECT on customer for reporting).

**2. Table-Level Privileges**

To list privileges granted to roles for each table, the following query is used:

SELECT grantee, table\_name, privilege\_type

FROM information\_schema.role\_table\_grants

WHERE table\_schema = 'public'

ORDER BY table\_name, grantee;

**Findings**:

* The **postgres** role has full privileges (SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER) on all tables in the public schema, including:
  + customer (sensitive columns: email, first\_name, last\_name)
  + rental (columns: customer\_id, rental\_date)
  + payment (sensitive columns: amount, payment\_date)
  + staff (sensitive columns: password, email)
  + inventory, film, actor, category, language, store, address, city, country
* The **public** role has **no privileges** on any tables, ensuring that new users/roles cannot access tables without explicit grants.
* No other roles have table-level privileges by default, as the dvd\_rental database is a database with minimal role configuration.

**3. Column-Level Privileges**

To check for column-specific grants, the following query is executed:

SELECT table\_name, column\_name, privilege\_type, grantee

FROM information\_schema.column\_privileges

WHERE table\_schema = 'public';

**Findings**:

* No column-level privileges are defined in the dvd\_rental database.
* Users with SELECT on a table (postgres on customer) can access all columns, including sensitive ones like customer.email or staff.password.
* This highlights a lack of fine-grained access control at the column level.

**Recomandation:**

* Grant column-level privileges to hide sensitive columns (like staff.password).

**4. Row-Level Security (RLS)**

To verify RLS status, the following query is used:

SELECT relname, relrowsecurity

FROM pg\_class

WHERE relkind = 'r' AND relnamespace = (SELECT oid FROM pg\_namespace WHERE nspname = 'public');

**Findings**:

* RLS is **disabled** (relrowsecurity = false) for all tables (customer, rental, payment, staff, etc.).
* Without RLS, users with SELECT privileges can access all rows in a table, posing a risk for sensitive data (payment.amount).

**Recomandation:**

* Enable RLS to restrict row access (limit payment to specific customer\_id).

**5. Schema Usage**

To list all tables and their schemas, the following query is executed:

SELECT table\_schema, table\_name

FROM information\_schema.tables

WHERE table\_schema = 'public';

**Findings**:

* All tables (customer, rental, payment, staff, inventory, etc.) reside in the **public** schema.
* No additional schemas are used, limiting schema-based isolation. Roles with access to the public schema can potentially interact with all tables (if granted privileges).

**Recomandation:**

* Use separate schemas for different user groups to enhance isolation.

**6. Authentication and Network Security**

**Authentication Methods**:

* The pg\_hba.conf file defines authentication methods. A typical entry might be:

# Example pg\_hba.conf entry

host all all 0.0.0.0/0 scram-sha-256

* **Findings**: The default dvd\_rental setup likely uses md5 or scram-sha-256 for password-based authentication (scram-sha-256 is recommended to prevent password sniffing and replay attacks). No application-specific roles are defined, so authentication is managed by the postgres role or external application logic.

**SSL Configuration**:

* Check postgresql.conf for SSL settings:

ssl = on

* **Findings**: SSL is not enabled by default but is critical for securing network connections. Enabling ssl = on is recommended to encrypt data in transit.

**Recommendations:**

* Enable SSL and use scram-sha-256 for authentication to secure connections.

**7. Data Integrity and Sensitive Data**

**Constraints**:

* **NOT NULL** constraints (such as customer.customer\_id, rental.rental\_date) ensure data validity.
* **Foreign key** constraints (such as rental.customer\_id references customer.customer\_id) enforce referential integrity, preventing unauthorized data manipulation.

**Sensitive Data**:

* **staff.password**: Stored as text or application-hashed (not managed by PostgreSQL’s pg\_authid). Security depends on application-level hashing (SHA-256) and is not enforced by the database.
* **customer.email**, **payment.amount**: Accessible to postgres without restrictions, highlighting the need for role-based or RLS controls.

***Summary***

*The dvd\_rental database has minimal security configuration:*

* ***Roles****: Only postgres (superuser) and public (no privileges) exist by default.*
* ***Privileges****:* 
  + *postgres has full privileges (SELECT, INSERT, UPDATE, DELETE, etc.) on all tables in the public schema.*
  + *No privileges are granted to public or other roles, ensuring new users have no access by default.*
  + *No column-level privileges or RLS are configured, allowing full table access for roles with SELECT.*
* ***Schema****: All tables are in the public schema, with no isolation.*
* ***Authentication****: Likely uses md5 or scram-sha-256, with SSL recommended but not enabled by default.*
* ***Data Integrity****: Enforced through constraints, but staff.password relies on application-level security.*