**Understanding SQL Security Levels and Real-World Risks**

**1. What are SQL Security Levels?**

SQL Server provides layered security controls:

* **Server-Level Login**: Authenticates users at the SQL Server instance level. (e.g., CREATE LOGIN hr\_login ...)
* **Database-Level User**: Maps a server login to a specific database. (e.g., CREATE USER hr\_user FOR LOGIN hr\_login;)
* **Schema-Level Permissions**: Grants access to entire logical sections (schemas) like HR or Sales.
* **Object-Level Permissions**: Grants access to specific tables, views, or stored procedures.

**2. Benefits of Applying Security Levels**

* Restrict sensitive data like salaries from non-HR staff.
* Prevent unauthorized changes to customer data.
* Reduce human errors by limiting access.
* Satisfy compliance and audit requirements.

**3. Real-World Risks Without Security**

* If everyone has full access:
  + Sensitive HR data could be leaked.
  + Accidental or malicious data modifications.
* Developers might unknowingly damage production data.
* Interns or junior developers might access restricted data.

**4. Task Summary: What We Did**

* Created database CompanyDB with two schemas: HR and Sales.
* Created login and users: hr\_login/hr\_user, sales\_login/sales\_user, and User1Login/User1.
* Assigned schema-level permissions:
  + hr\_user can only access HR schema.
  + sales\_user can only access Sales schema.
  + User1 has read-only access to Sales.
* Inserted sample data and verified access using EXECUTE AS USER.
* Sales user could not read HR data, and vice versa.

**5. Security Scenario: "The Overpowered Developer"**

**What Went Wrong:**

1. Adil accidentally deleted all employee records.
2. Salary report was shared externally.
3. A junior developer gained full access.
4. Tables were created in the wrong schema.

**Root Causes:**

* No separation between development and production environments.
* Developers had full access without restriction.
* No schema-level or role-based control.
* No auditing or approval for login creation.

**Suggested Solutions:**

* Use schema-level permissions (as done in this task).
* Assign minimum roles (e.g., ReadOnly, DataEntry).
* Use views to hide sensitive columns.
* Implement audit logs.
* Separate dev and prod environments.

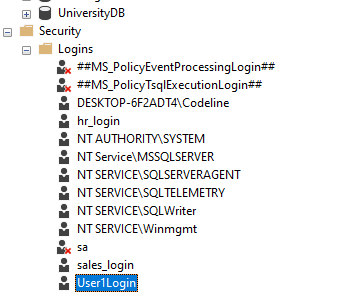
**Lessons Learned:**

* Developers should access only non-sensitive schemas/tables.
* Only DBAs/admins should manage logins and schemas.
* Always apply "least privilege" principle to reduce risk.

**Bonus: Role-Based Security Test**

* Created ReadOnly\_Dev role and granted SELECT only on HR schema.
* Added hr\_user to this role.
* Verified that INSERT/DELETE commands failed.

**Screen shots**

**** **A screenshot of a computer

AI-generated content may be incorrect.** A white background with black text

AI-generated content may be incorrect.

Codes:

-- SQL SECURITY TASK: Schema-Level Access Control

-- Step 1: Create Database

CREATE DATABASE CompanyDB;

GO

USE CompanyDB;

GO

-- Step 2: Create Schemas

CREATE SCHEMA HR;

GO

CREATE SCHEMA Sales;

GO

-- Step 3: Create Tables in Schemas

CREATE TABLE HR.Employees (

EmployeeID INT PRIMARY KEY,

Name NVARCHAR(100),

Position NVARCHAR(100),

Salary DECIMAL(10, 2)

);

GO

CREATE TABLE Sales.Customers (

CustomerID INT PRIMARY KEY,

Name NVARCHAR(100),

PurchaseAmount DECIMAL(10, 2)

);

GO

CREATE TABLE Sales.Orders (

OrderID INT PRIMARY KEY,

CustomerName NVARCHAR(100)

);

GO

CREATE TABLE HR.People (

EmpID INT PRIMARY KEY,

FullName NVARCHAR(100)

);

GO

-- Insert sample data

INSERT INTO Sales.Orders VALUES (1, 'Ali'), (2, 'Fatma');

INSERT INTO HR.People VALUES (101, 'Ahmed'), (102, 'Huda');

GO

-- Step 4: Create Logins and Users

CREATE LOGIN hr\_login WITH PASSWORD = 'Hr@12345';

GO

CREATE USER hr\_user FOR LOGIN hr\_login;

GO

CREATE LOGIN sales\_login WITH PASSWORD = 'Sales@12345';

GO

CREATE USER sales\_user FOR LOGIN sales\_login;

GO

CREATE LOGIN User1Login WITH PASSWORD = 'StrongPass123';

GO

CREATE USER User1 FOR LOGIN User1Login;

GO

-- Step 5: Grant Schema-Level Permissions

-- HR user access only HR schema

GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::HR TO hr\_user;

DENY SELECT ON SCHEMA::Sales TO hr\_user;

GO

-- Sales user access only Sales schema

GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::Sales TO sales\_user;

DENY SELECT ON SCHEMA::HR TO sales\_user;

GO

-- User1 gets read access only on Sales

GRANT SELECT ON SCHEMA::Sales TO User1;

GO

-- Step 6: Optional - Read-Only Role for HR

CREATE ROLE ReadOnly\_Dev;

GO

GRANT SELECT ON SCHEMA::HR TO ReadOnly\_Dev;

EXEC sp\_addrolemember 'ReadOnly\_Dev', 'hr\_user';

GO

-- Step 7: Impersonation Test (User1)

EXECUTE AS USER = 'User1';

GO

-- Should succeed

SELECT \* FROM Sales.Orders;

GO

-- Should fail

SELECT \* FROM HR.People;

GO

-- Revert session

REVERT;

GO