companyDB Schema-Level Access & Security Report

Prepared by: Mohammed Yusuf Alkhusaibi  
Date: 2025-05-29

**1. Task Implementation Summary**

SQL logins and users were created for HR and Sales departments. Each department was assigned its own schema.  
Tables were created inside each schema, and permissions were restricted so that each user could only access their own schema.

**2. Schema Permissions Summary**

- HR user has full access to HR schema and no access to Sales.  
- Sales user has full access to Sales schema and no access to HR.  
- Verified by executing SELECT queries as different users.

**3. Why Schema-Level Security is Better**

Schema-level security allows permission to be managed at a logical group level. This reduces complexity and improves maintainability,  
especially when multiple tables or procedures belong to the same functional area. It also ensures clean segregation of access between departments.

**4. Reflection Report: SQL Security Levels**

- Server-Level Login: Grants access to the SQL Server instance.  
- Database-Level User: Maps the login to a specific database.  
- Schema-Level Permission: Assigns access at schema level, simplifying control.  
- Object-Level Permission: Fine-grained control for individual tables or views.

**5. Benefits of Security Levels**

- Prevents unauthorized access to sensitive data.  
- Reduces human error by limiting access scope.  
- Ensures compliance with data protection regulations.  
- Allows scalable security across large systems.

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

- Developers or interns may access or change sensitive data.  
- Salary leaks or accidental data deletion could occur.  
- Auditing and compliance may fail due to poor access control.

**7. Security Scenario: The Overpowered Developer**

Adil, a developer, had full access to production. He:  
1. Accidentally deleted employee records.  
2. Shared salary data externally.  
3. Created unauthorized users.  
4. Caused permission issues by using the wrong schema.

**Root Causes:**

- No dev/prod environment separation.  
- Lack of schema enforcement.  
- No role-based access control.  
- Excessive permissions to non-admins.

**Solutions:**

- Use schema-level permissions to restrict access.  
- Assign roles like ReadOnly\_Dev for safer dev access.  
- Enforce view-based access to sensitive data.  
- Log audit actions and separate environments.

**8. Lessons Learned**

- Developers should have minimal, read-only access in prod.  
- DBAs should handle role creation and schema ownership.  
- Least-privilege principle protects data and systems.

**9. Bonus: Role-Based Security Test**

A role named ReadOnly\_Dev was created with SELECT only access. Attempts to INSERT or DELETE were denied, confirming permission restrictions.