🛡️ SQL Server Security Level Task Report

☑️ Part 1: SQL Server Security Implementation

1. Create Logins (Server Level)

```
CREATE LOGIN hr_login WITH PASSWORD = 'HR@123';
CREATE LOGIN sales_login WITH PASSWORD = 'Sales@123';
```

2. Create Users in the Database

```
USE CompanyDB;

CREATE USER hr_user FOR LOGIN hr_login;
CREATE USER sales_user FOR LOGIN sales_login;
```

3. Create Schemas

```
CREATE SCHEMA HR;
GO
CREATE SCHEMA Sales;
GO
```

4. Create Tables Under Schemas

```
-- HR Schema Table
CREATE TABLE HR.Employees (
  EmployeeID INT PRIMARY KEY,
  FullName NVARCHAR(100),
  Position NVARCHAR(50),
  Salary DECIMAL(10, 2)
);
GO
```

```
-- Sales Schema Table
CREATE TABLE Sales.Customers (
  CustomerID INT PRIMARY KEY,
  FullName NVARCHAR(100),
  Phone NVARCHAR(20)
);
GO
```

5. Grant and Deny Schema Permissions

```
-- HR User: Access only HR schema
GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::HR TO hr_user;
DENY SELECT, INSERT, UPDATE, DELETE 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, INSERT, UPDATE, DELETE ON SCHEMA::HR TO sales_user;
GO
```

☑️ Part 2: Practical Testing Steps

◇ As `hr_login`:

- ✅ Able to `SELECT * FROM HR.Employees`
- ❌ Access denied to `Sales.Customers`

◇ As `sales_login`:

- ✅ Able to `SELECT * FROM Sales.Customers`
- ❌ Access denied to `HR.Employees`

✅ Confirmed: Schema permissions worked correctly.

📸 Screenshots Checklist

✅ Take screenshots of:

- Login creation in SSMS

```
-- Create login for HR department
CREATE LOGIN hr_login WITH PASSWORD = 'HR@123';

-- Create login for Sales department
CREATE LOGIN sales_login WITH PASSWORD = 'Sales@123';
```

- User creation

```
-- Create users mapped to the logins
CREATE USER hr_user FOR LOGIN hr_login;
CREATE USER sales_user FOR LOGIN sales_login;
```

- Schema permissions being applied

```
--Apply Schema-Level Permissions
-- Grant HR user access to HR schema only
GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::HR TO hr_user;

-- Deny HR user access to Sales
DENY SELECT, INSERT, UPDATE, DELETE ON SCHEMA::Sales TO hr_user;

-- Grant Sales user access to Sales schema only
GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::Sales TO sales_user;

-- Deny Sales user access to HR
DENY SELECT, INSERT, UPDATE, DELETE ON SCHEMA::HR TO sales_user;
```

  - Query results showing access works only for their assigned schema

```
SELECT * FROM HR.Employees;
```

121 %

Results | Messages

	EmployeeID	FullName	Position	Salary
1	101	Aisha Al Zadjali	HR Specialist	850.00
2	102	Faisal Al Habsi	Recruiter	900.00
3	103	Layla Al Busaidi	HR Manager	1200.00

```
SELECT * FROM Sales.Customers;
```

121 %

Results | Messages

	CustomerID	FullName	Phone
1	201	Ahmed Al Nabhani	91234567
2	202	Muna Al Rawahi	93345678
3	203	Hamad Al Kindi	94456789

- Query results:
  - SELECT from assigned schema (✅ success)
  - SELECT from unauthorized schema (❌ failure)

Explanation

◇ Why Schema-Level Security is Better than Table-by-Table Permissions

- Easier to manage: you assign permission once for a whole schema.
- Reduces error: no need to manually grant access to each table.
- Cleaner roles: HR team gets access to all HR tables with one rule.

◇ How This Setup Supports Data Segregation

- Keeps departments' data **separate and secure**.
- Ensures users only see what they need.

- Prevents mistakes and data leaks across teams.
- Follows the principle of **least privilege**.

🧠 Summary

- ✅ Created secure logins and users
- ✅ Organized tables into HR and Sales schemas
- ✅ Applied schema-level access controls
- ✅ Tested access by connecting as both users
- ✅ Validated security rules using queries
- ✅ Wrote reflection on real-world security benefit