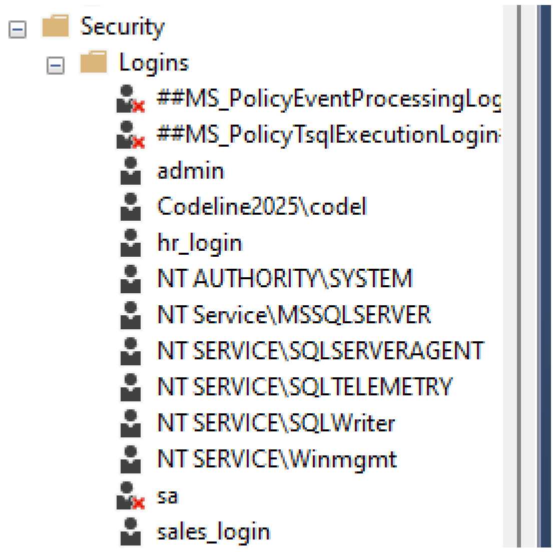
Task Output Checklist Ask the trainees to:

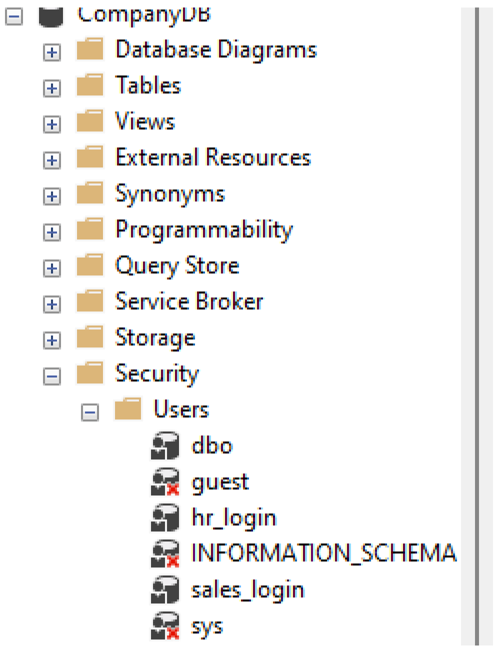
1. Take screenshots of:

o Login creation



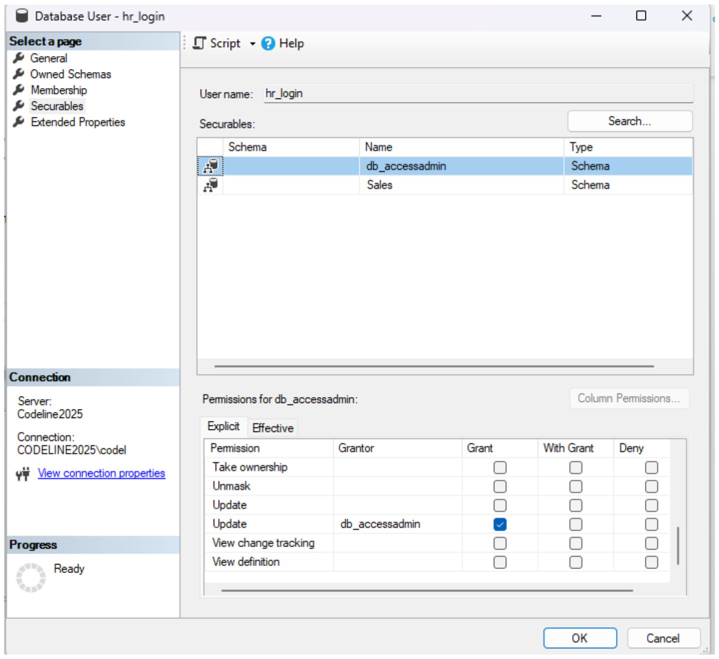
Security
Logins
##MS_PolicyEventProcessingLog
##MS_PolicyTsqlExecutionLogin
admin
Codeline2025\codel
hr_login
NT AUTHORITY\SYSTEM
NT Service\MSSQLSERVER
NT SERVICE\SQLSERVERAGENT
NT SERVICE\SQLTELEMETRY
NT SERVICE\SQLWriter
NT SERVICE\Winmgmt
sa
sales_login

o User creation



```
CompanyDB
  Database Diagrams
  Tables
  Views
  External Resources
  Synonyms
  Programmability
  Query Store
  Service Broker
  Storage
  Security
    Users
      dbo
      guest
      hr_login
      INFORMATION_SCHEMA
      sales_login
      sys
```

o Schema permissions



Database User - hr_login

User name: hr_login

Securables:

	Schema	Name	Type
		db_accessadmin	Schema
		Sales	Schema

Connection

Server:
Codeline2025

Connection:
CODELINE2025\codel

View connection properties

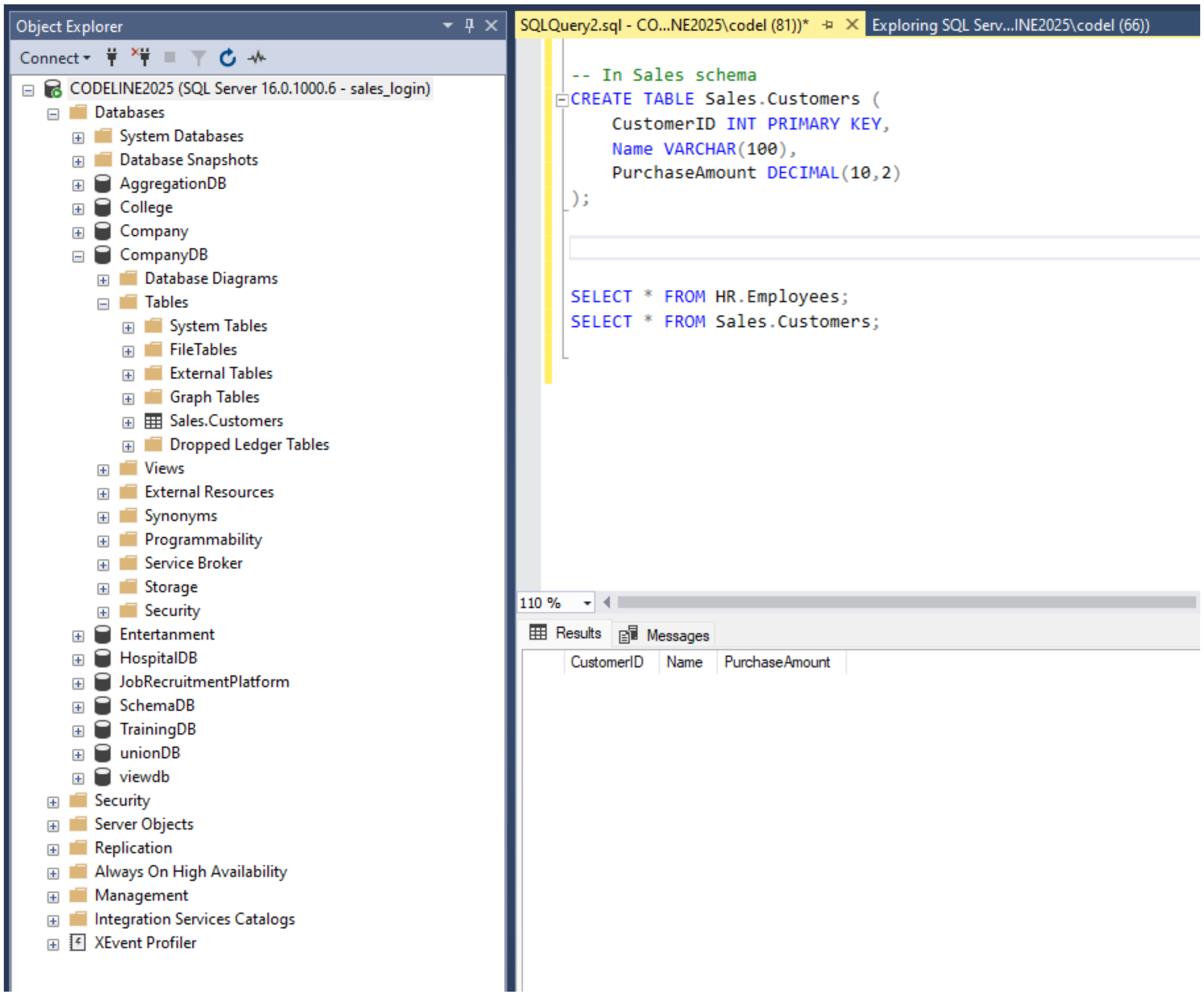
Progress

Ready

Permissions for db_accessadmin:

Permission	Grantor	Grant	With Grant	Deny
Take ownership		☐	☐	☐
Unmask		☐	☐	☐
Update		☐	☐	☐
Update	db_accessadmin	☑	☐	☐
View change tracking		☐	☐	☐
View definition		☐	☐	☐

o Query results showing access works only for their assigned schema



```sql
-- In Sales schema
CREATE TABLE Sales.Customers (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR(100),
    PurchaseAmount DECIMAL(10,2)
);


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

CustomerID	Name	PurchaseAmount

Security Analysis Report

Title: Understanding SQL Security Levels and Real-World Risks
Prepared by: [Your Name]
Date: [Insert Date]

1. Summary of the Problems

The following incidents occurred due to poor access control:

- **Accidental Data Deletion:**
  A developer, Adil, mistakenly ran a `DELETE FROM Employees` command on the **production** database instead of the **test** environment. There was no backup.
- **Salary Data Leaked:**
  Adil created a test report that included sensitive salary information. He accidentally shared this file with an external UI developer.
- **Unauthorized Role Creation:**
  Adil created a new SQL login for a junior developer without informing the DBA. That login was then used to access confidential HR data.
- **Schema Confusion:**
  Adil created tables in the default dbo schema instead of the HR schema, causing permission and access issues for HR users.

2. Root Causes

Several key security flaws led to these issues:

- **No Environment Separation:**
  There was no clear distinction or isolation between **development** and **production** databases.
- **Excessive Permissions:**
  Developers were given **full access** to production systems without any restrictions.
- **No Schema-Level Controls:**
  Data was not organized into department-based schemas with controlled access.
- **Lack of Role-Based Permissions:**
  Permissions were not assigned based on roles (e.g., read-only, data-entry), which led to unnecessary access.

3. Suggested Solutions

To prevent such incidents, the following best practices should be implemented:

- ✅ **Use Schema-Level Permissions:**
  Assign access at the schema level so users can only interact with the data relevant to their department.
- ✅ **Separate Roles Clearly:**
  Create roles like `ReadOnly_Dev`, `DataEntry_Sales`, etc., and assign only required permissions to each role.
- ✅ **Use Views to Protect Sensitive Data:**
  Expose only necessary columns via `VIEW`s and restrict access to underlying tables (e.g., hide salary columns).
- ✅ **Restrict Role and Login Creation:**
  Only DBAs should be allowed to create logins, users, and assign roles. This should be logged and reviewed.
- ✅ **Environment Isolation:**
  Create separate environments for **development**, **testing**, and **production**. Ensure production access is limited and monitored.
- ✅ **Audit Logging:**
  Enable login/audit logging to track changes and access patterns.

4. Lessons Learned

❓ What should developers have access to?

- Only the **development** or **test** environments.
- Data that is **masked** or **non-sensitive**.
- Read-only access to production data if absolutely necessary, and only via views or controlled interfaces.

❓ What should be restricted to DBAs or Admins?

- Production access
- Backup and restore operations

- Creating/modifying logins and permissions
- Sensitive data access (e.g., salaries, HR records)

❓ Why is the "Minimum Privilege" Principle Important?

- It **reduces risk** of accidental or malicious actions.
- **Limits damage** from errors or misuse.
- Helps in **compliance** with regulations (GDPR, HIPAA).
- Makes **auditing and troubleshooting** easier.

🔧 Bonus Activity (Optional Simulation)

1. **Create a Role:**
   a. Name: `ReadOnly_Dev`
   b. Grant only `SELECT` permission on the Sales schema.
2. **Assign Role to Developer Login:**
   a. Map a developer to this role.
3. **Try INSERT or DELETE:**
   a. Attempt data modification — it should **fail** due to insufficient privileges.