The Learning House
427 S 4th Street #300
Louisville KY 40202

SQL Server Security

.NET Cohort

SOFTWARE GUILD

Lesson Goals

- Learn to create logins and users
- Learn to grant permissions to database objects

The Principle of Least Privilege

We should only give a user account the minimum level of privileges that are essential to the user’s work.

This leads to better stability, better security, and ease of deployment.

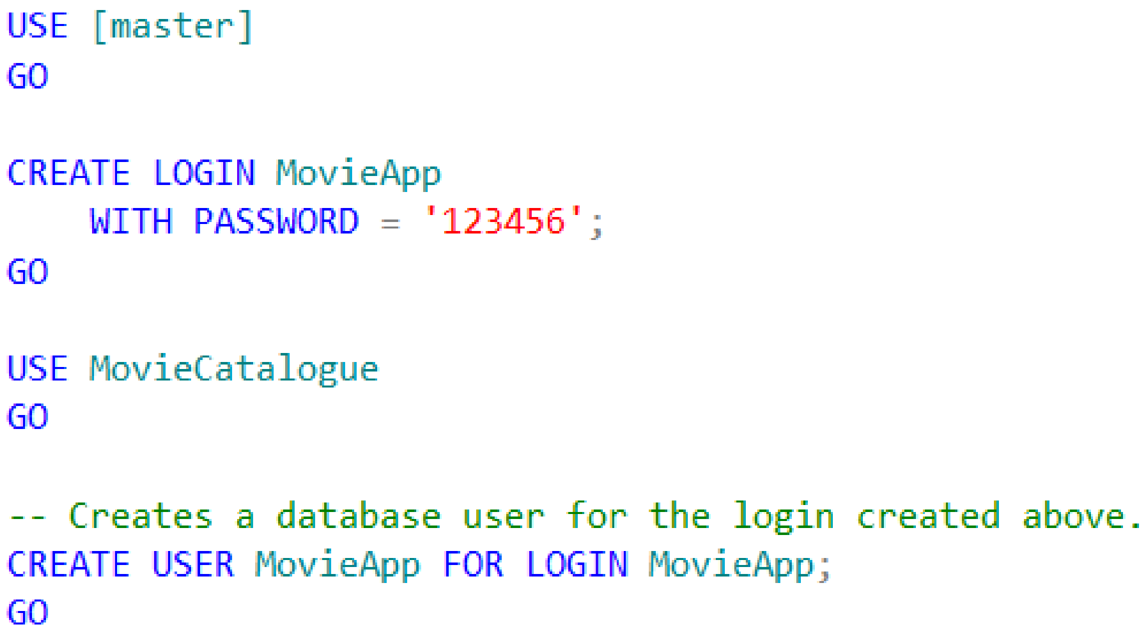
Logins vs. Users

Logins allow a user to connect to SQL Server; users are associations of a login with a database.

Login names and user names do not have to match, but they often do.

Creating a Login and User

Let’s create a login and user for our movie app.



```
USE [master]
GO

CREATE LOGIN MovieApp
    WITH PASSWORD = '123456';
GO

USE MovieCatalogue
GO

-- Creates a database user for the login created above.
CREATE USER MovieApp FOR LOGIN MovieApp;
GO
```

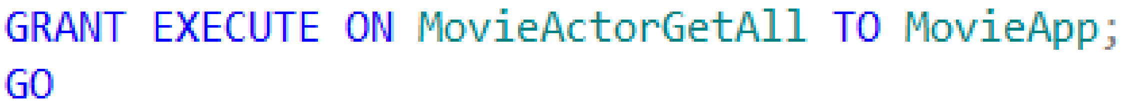
What Can the User Do?

Not much. Open a new instance of Management Studio and log in as the new user.

We haven’t given the user any permissions, so they can’t see anything.

Granting Permissions

We can use the GRANT command to give a user access to SQL Server Objects.



```
GRANT EXECUTE ON MovieActorGetAll TO MovieApp;
GO
```

Other Common Grants

Permission	Description
ALTER	Change objects with the alter command
DELETE	Delete from a table
EXECUTE	Execute a function or stored procedure
INSERT	Insert data into a table
SELECT	Select data from a table or view
UPDATE	Update data in a table

DENY

This is the opposite of grant. Use this in a case where someone has been granted a role by another permission and you want to explicitly override it.

If multiple permissions apply to a user, DENY always beats GRANT.

View

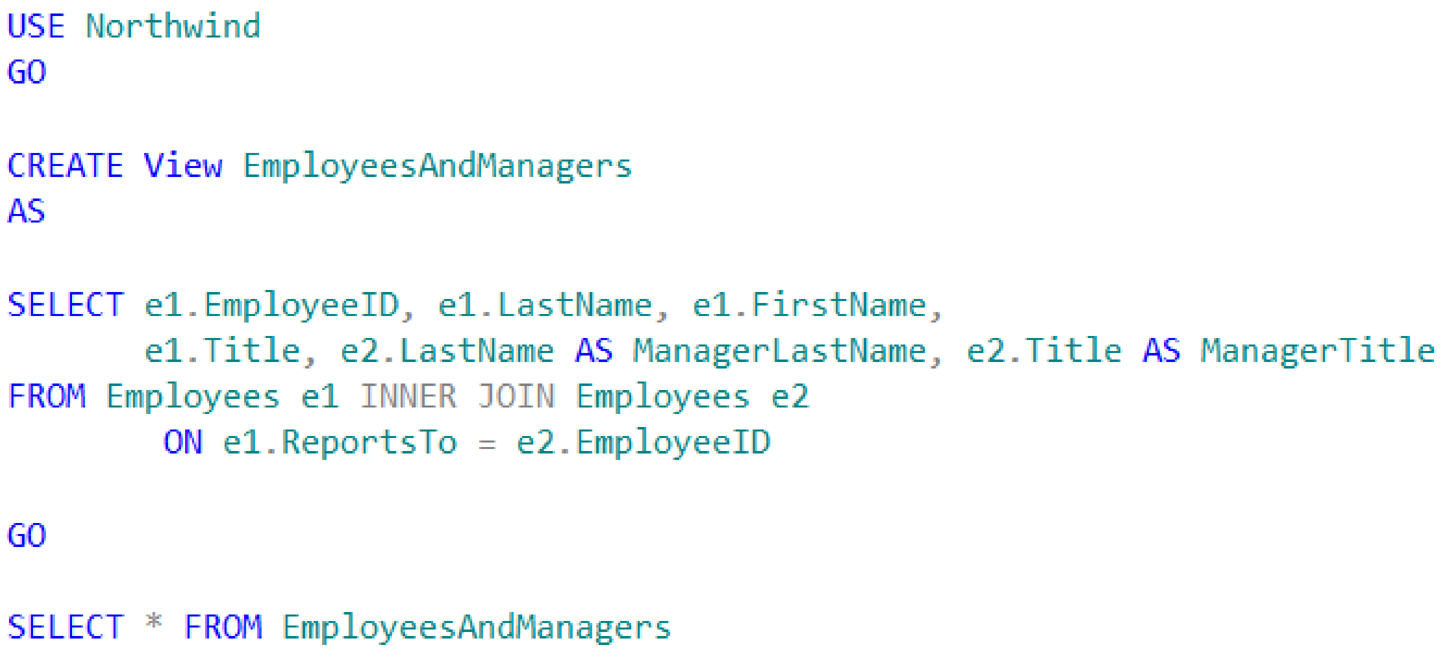
A view is a virtual table whose rows and columns are defined by a query.

We generally use this:

1. To focus or simplify a user’s interface with the database.
2. As a security mechanism: it allows users to access data through a view but does not grant access to the underlying tables.
3. To provide a backwards-compatible interface to a table that has been refactored.

Creating a View (Northwind)

We can use the CREATE VIEW AS statement and then select from it as if it were a table.



```
USE Northwind
GO

CREATE View EmployeesAndManagers
AS

SELECT e1.EmployeeID, e1.LastName, e1.FirstName,
       e1.Title, e2.LastName AS ManagerLastName, e2.Title AS ManagerTitle
FROM Employees e1 INNER JOIN Employees e2
        ON e1.ReportsTo = e2.EmployeeID

GO

SELECT * FROM EmployeesAndManagers
```

Roles

SQL Server has pre-defined roles at the server and database levels.

Think of a role as a group of users that share the same security permissions.

We can create our own roles if we want, though it isn’t common in basic scenarios.

Server Roles

Role Name	Description
sysadmin	“God”
serveradmin	Change server config and shut down server
securityadmin	Can manage logins and properties (GRANT, DENY, REVOKE)
processadmin	Can end processes that are running in SQL Server
setupadmin	Can add and remove linked servers
bulkadmin	Can bulk insert data using BULK INSERT command
diskadmin	Can manage disk usage
dbcreator	Create/Alter/Drop/Restore databases
public	Default for all users, minimum privilege to use the database

Database Roles

Role Name	Description
db_owner	“God”, but only for the database, not the server
db_securityadmin	Manages roles and permissions
db_accessadmin	User management
db_backupoperator	Run backups
db_ddladmin	Create/alter/drop objects
db_datawriter	Add, delete, or change data in user tables
db_datareader	Select data from all user tables