THURSDAY, AUGUST 31, 2023

USERS & ROLES

DOCUMENTATION

AYMANE SABRI

DATA DEVELOPER

I. SQL Server .

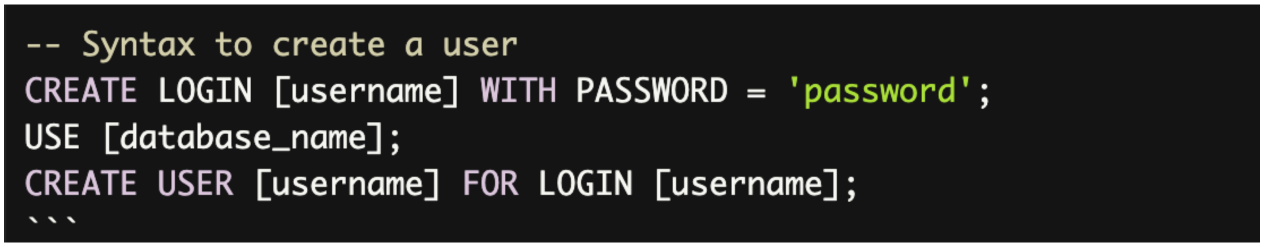
1. Introduction to Users and Roles?

- Explanation of users : Users are individual database accounts that can access and manipulate data within a SQL Server database.
- Explanation of roles: Roles are a way to group multiple users together and assign permissions and privileges to the group as a whole.

2. User Management:

a. Creating users :

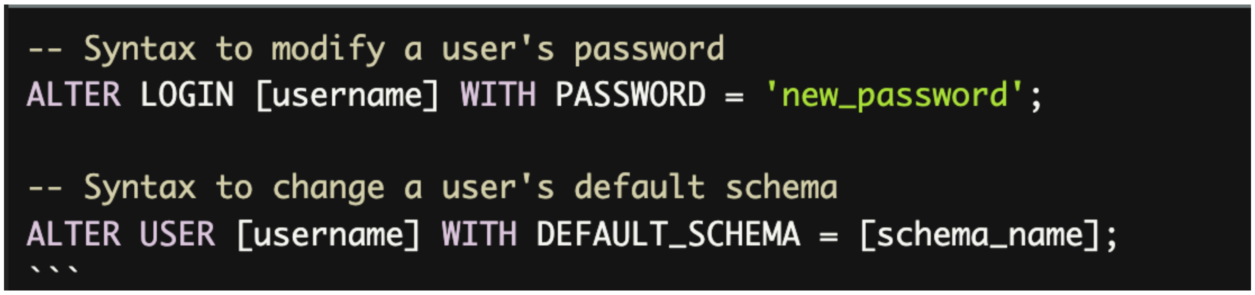
Users are individual database accounts that can access and manipulate data within a SQL Server database.



~~~sql
-- Syntax to create a user
CREATE LOGIN [username] WITH PASSWORD = 'password';
USE [database_name];
CREATE USER [username] FOR LOGIN [username];
```
~~~

b. Modifying users :

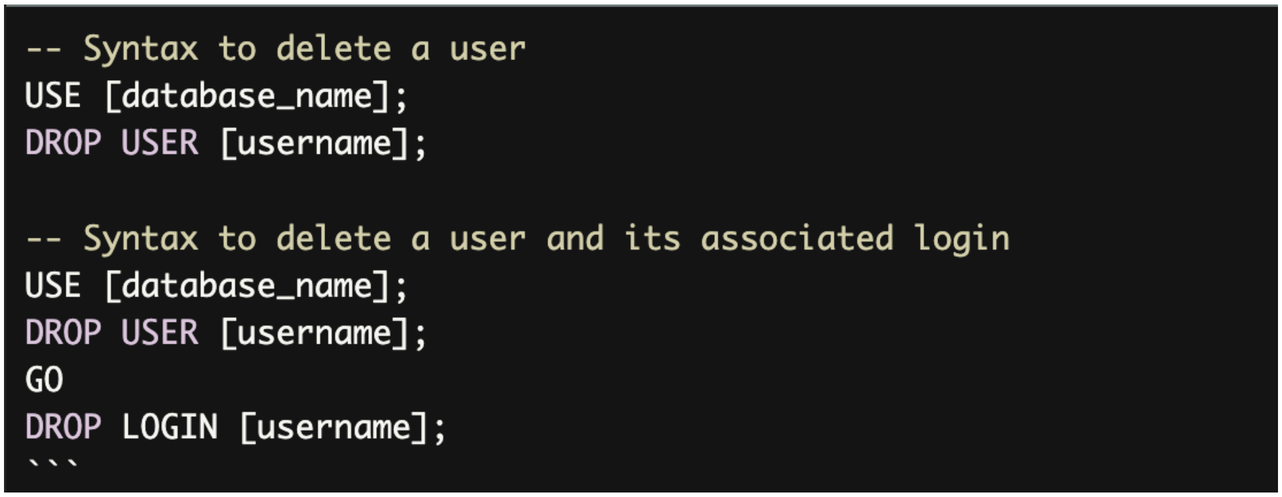
Modifying users involves changing user properties such as password, default schema, etc.

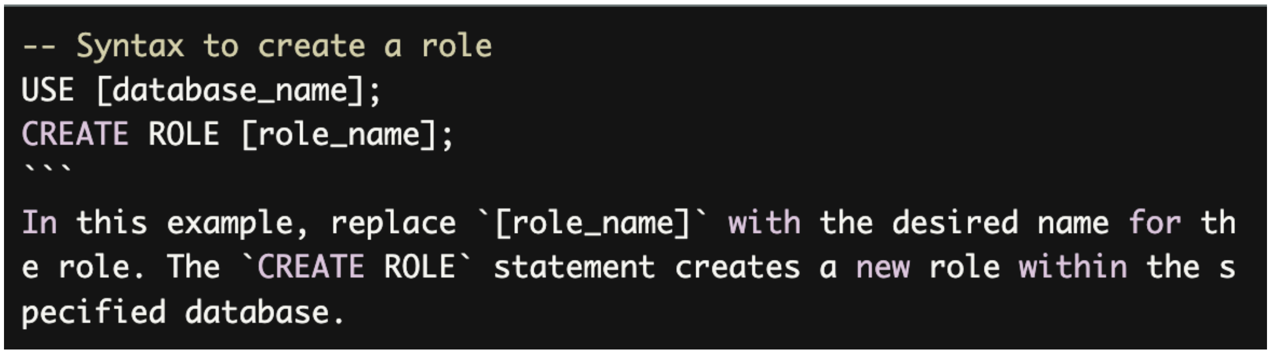


~~~sql
-- Syntax to modify a user's password
ALTER LOGIN [username] WITH PASSWORD = 'new_password';

-- Syntax to change a user's default schema
ALTER USER [username] WITH DEFAULT_SCHEMA = [schema_name];
```
~~~

c. Deleting users :

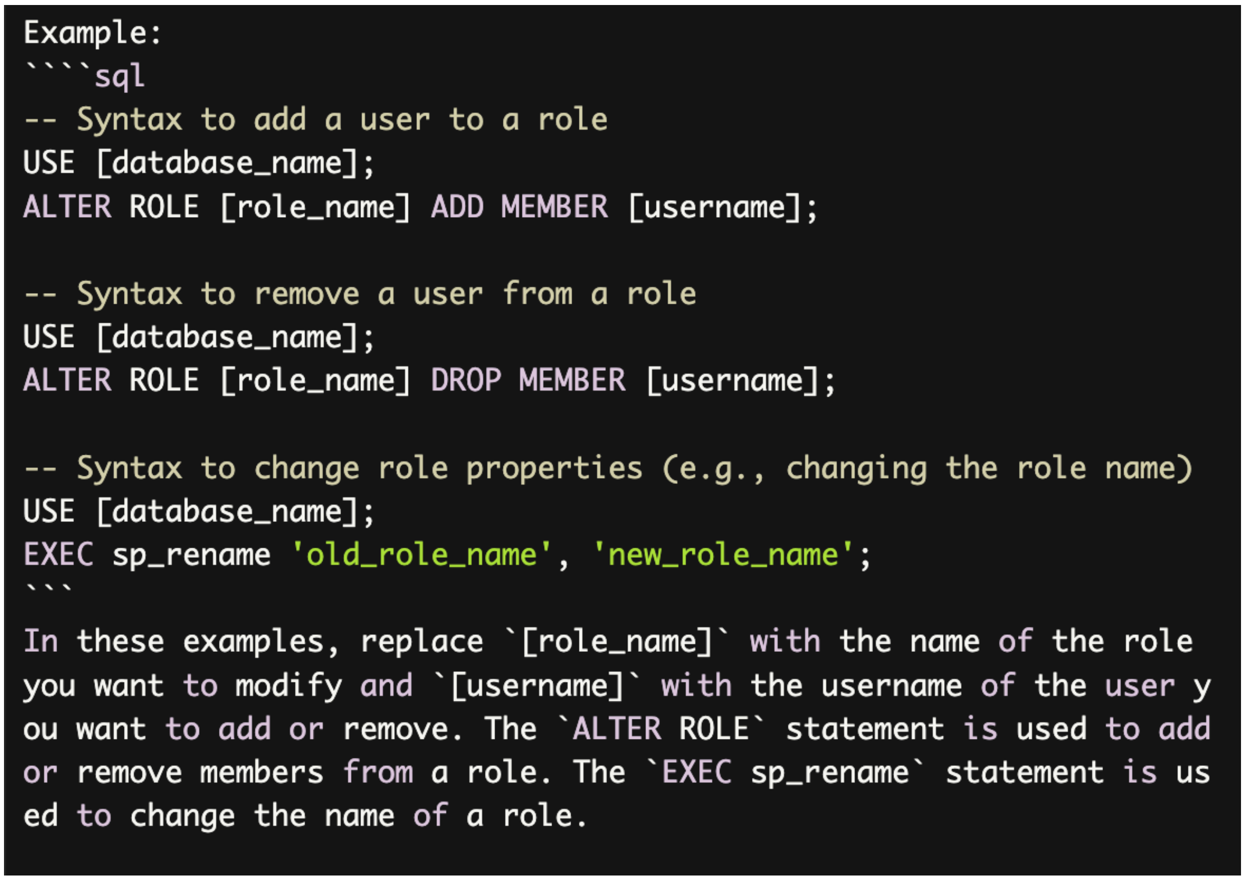
Deleting users involves removing user accounts from the SQL Server.



```
-- Syntax to delete a user
USE [database_name];
DROP USER [username];

-- Syntax to delete a user and its associated login
USE [database_name];
DROP USER [username];
GO
DROP LOGIN [username];
```
```

## 3. Role Management:

### a. Creating Roles:

Creating roles involves creating groups to which multiple users can be assigned, allowing you to assign permissions and privileges to the group as a whole.



```
-- Syntax to create a role
USE [database_name];
CREATE ROLE [role_name];
```
In this example, replace `[role_name]` with the desired name for th
e role. The `CREATE ROLE` statement creates a new role within the s
pecified database.
```

b. Modifying Roles:

Modifying roles involves adding or removing users from roles and changing role properties.



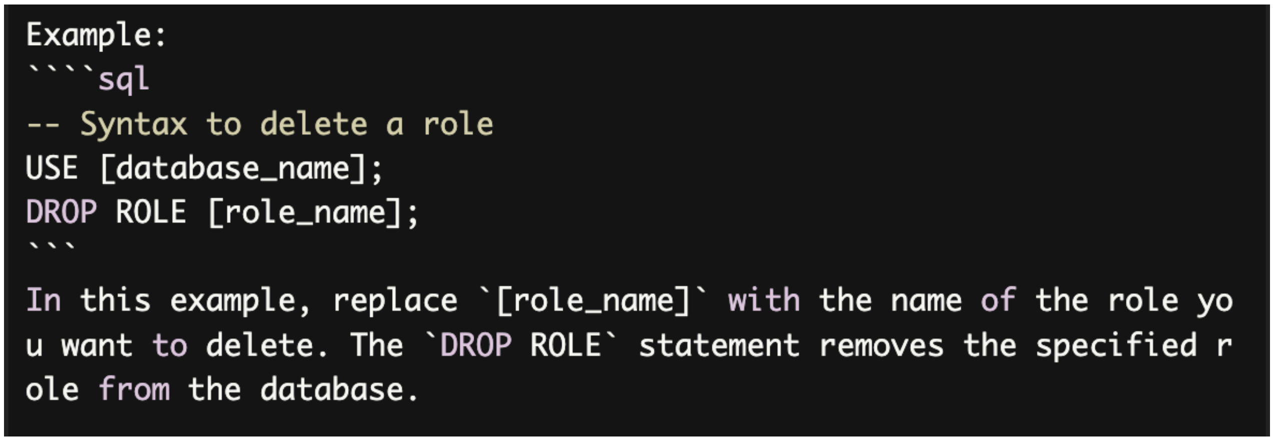
~~~~~
Example:
````sql
-- Syntax to add a user to a role
USE [database_name];
ALTER ROLE [role_name] ADD MEMBER [username];

-- Syntax to remove a user from a role
USE [database_name];
ALTER ROLE [role_name] DROP MEMBER [username];

-- Syntax to change role properties (e.g., changing the role name)
USE [database_name];
EXEC sp_rename 'old_role_name', 'new_role_name';
```
In these examples, replace `[role_name]` with the name of the role
you want to modify and `[username]` with the username of the user y
ou want to add or remove. The `ALTER ROLE` statement is used to add
or remove members from a role. The `EXEC sp_rename` statement is us
ed to change the name of a role.
~~~~~

c. Deleting Roles:

Deleting roles involves removing roles from the SQL Server.



~~~~~
Example:
````sql
-- Syntax to delete a role
USE [database_name];
DROP ROLE [role_name];
```
In this example, replace `[role_name]` with the name of the role yo
u want to delete. The `DROP ROLE` statement removes the specified r
ole from the database.
~~~~~

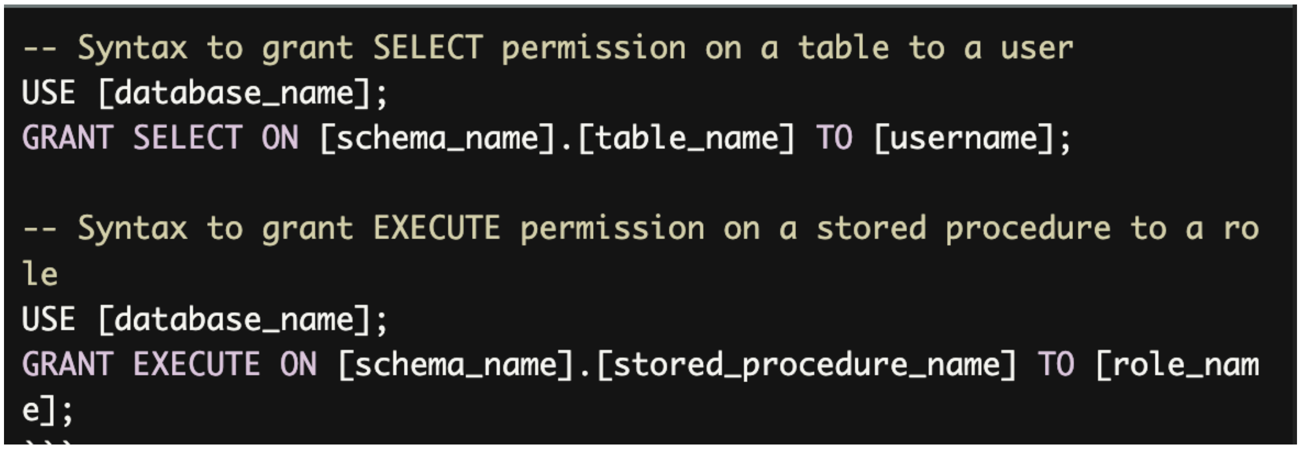
4. Permissions and Privileges:

a. Explanation of Permissions :

Permissions in SQL Server define the actions that a user or role can perform on specific database objects.

a. Granting Permissions :

To grant specific permissions to users or roles, you can use the `GRANT` statement in SQL Server.

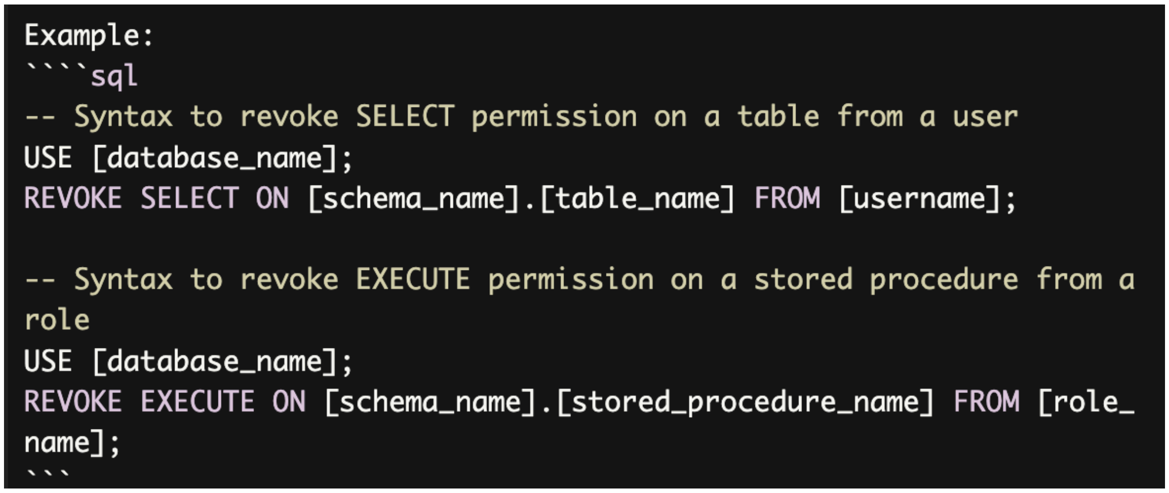


~~~sql
-- Syntax to grant SELECT permission on a table to a user
USE [database_name];
GRANT SELECT ON [schema_name].[table_name] TO [username];

-- Syntax to grant EXECUTE permission on a stored procedure to a role
USE [database_name];
GRANT EXECUTE ON [schema_name].[stored_procedure_name] TO [role_name];
```
~~~

a. Revoking Permissions :

To remove previously granted permissions from users or roles, you can use the `REVOKE` statement in SQL Server.



~~~
Example:
````sql
-- Syntax to revoke SELECT permission on a table from a user
USE [database_name];
REVOKE SELECT ON [schema_name].[table_name] FROM [username];

-- Syntax to revoke EXECUTE permission on a stored procedure from a role
USE [database_name];
REVOKE EXECUTE ON [schema_name].[stored_procedure_name] FROM [role_name];
```
~~~

5. Built-in Roles :

a. Explanation of Built-in Roles:

SQL Server provides several pre-defined roles that come with specific permissions and privileges, and are designed to simplify the process of managing permissions for common database operations. Built-in roles help to ensure security and simplify the assignment of permissions to users.

b. Common Built-in Roles :

- sysadmin: The sysadmin role is the highest level of privilege in SQL Server. Members of this role have full administrative rights and can perform any action within the server and its databases.
- db_owner: The db_owner role is a database-level role that has full control over a specific database. Members of this role can perform all configuration and maintenance tasks, as well as manage other roles and permissions within the database.
- db_datareader: The db_datareader role is a database-level role that grants read-only access to all user tables within a particular database. Members of this role can view data but cannot perform any modifications.
- db_datawriter: The db_datawriter role is a database-level role that grants write access to all user tables within a specific database. Members of this role can modify data but cannot perform any administrative tasks.

6. Security Best Practices :

- Principle of Least Privilege
- Regular Review and Auditing
- Strong Password Policies
- Implementing Database Auditing
- Strong Authentication and Authorization
- Regular Database Backups
- Secure Network Configuration