**Creating user in SQL Server**

Creating a user in SQL Server involves a few steps:

1. Connect to SQL Server through SQL Server Management Studio (SSMS)
2. Create a Login: Before creating a user, we need to create a login. A login allows a user to connect to the SQL Server instance. We can create a login using the CREATE LOGIN statement, specifying the login name and password.

Syntax:

CREATE LOGIN YourLoginName WITH PASSWORD = 'YourPassword';

1. Create a User: Once we have a login, we can create a user associated with that login. This user can be mapped to a specific database or set of databases. Use the CREATE USER statement for this.

CREATE USER YourUserName FOR LOGIN YourLoginName;

Login names are at server level and User names exist at Database level.

A database user is not the same as a login. A login provides to a user or application the ability to connect to a SQL Server instance, whereas a database user provides the login rights to access a database.

It's perfectly normal if you have a login name “Robert” and the same “Robert” username in, let's say, “Test″ databse.

Note: we can't have 2 logins with same name; and neither we can have 2 usernames of same name in a database.

Typically, the database user name is the same as the login name, though it does not recommended.

During setting the password, we may face problems and get error messages such as:

Error:

The password does not meet the password policy requirements. Check the minimum password length, password complexity and password history requirements.

To resolve this issue, use one of the following solutions:

* **Adhere to the password complexity requirements**  
  Users that do not have administrative access to the system must adhere to the password complexity requirements defined by Windows.
* **Disable the password complexity requirements policy setting**

Warning:

This solution involves modifying the Windows Operating System privacy setting, and may affect the security and privacy of the applications on the machine.

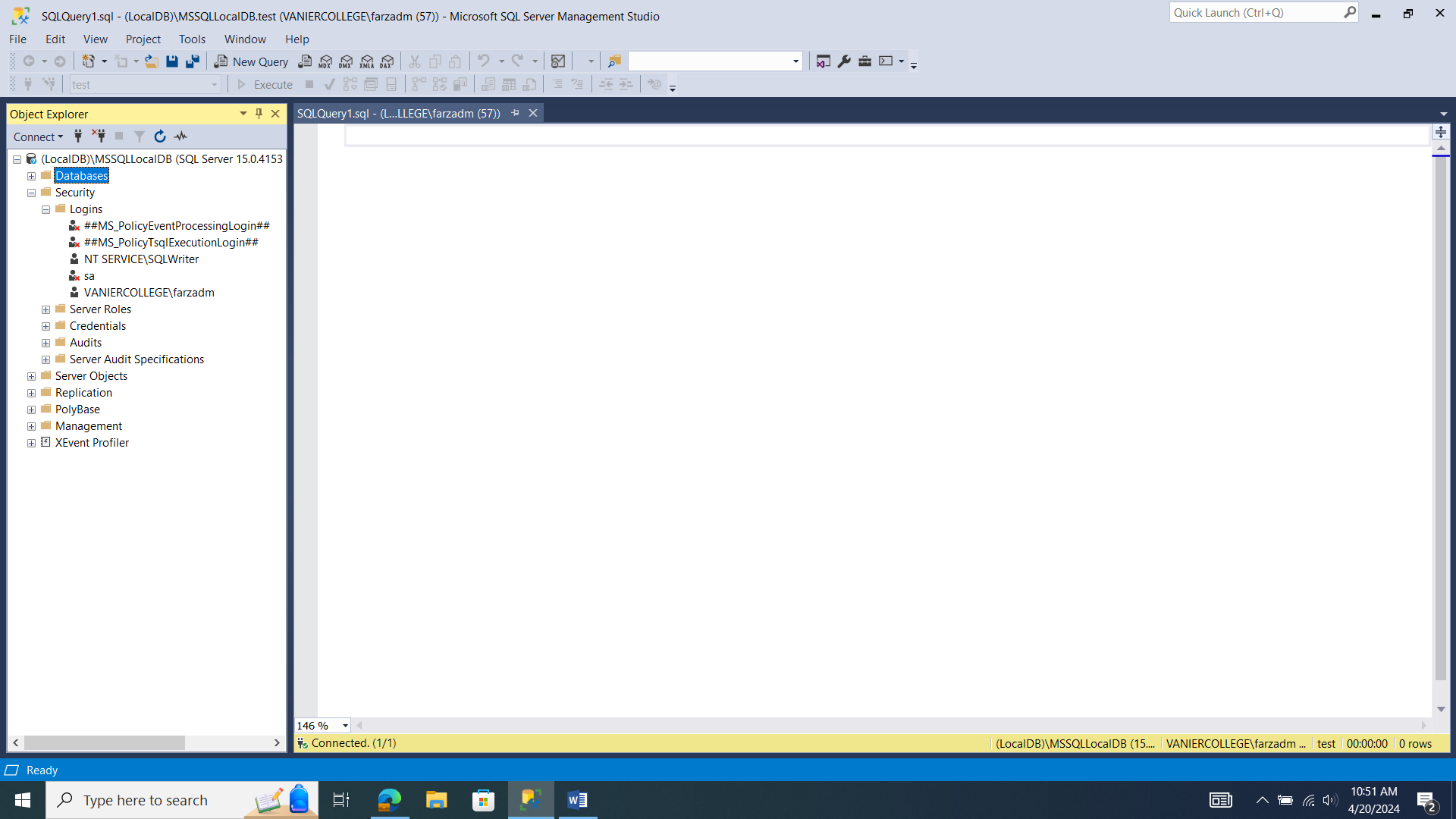
Disabling password setting requirements policy gets done through the Windows Local Security Policy console using the following instructions:

1. To open the Local Security Policy console, click **Start**> type *secpol.msc*
2. In the **Local Security Policy** console, navigate to **Account Policies > Password Policy**.
3. On the right pane, double-click **Password must meet complexity requirements**.
4. Select **Disabled** > click **Apply** > click **OK** and close the Local Security Policy console.

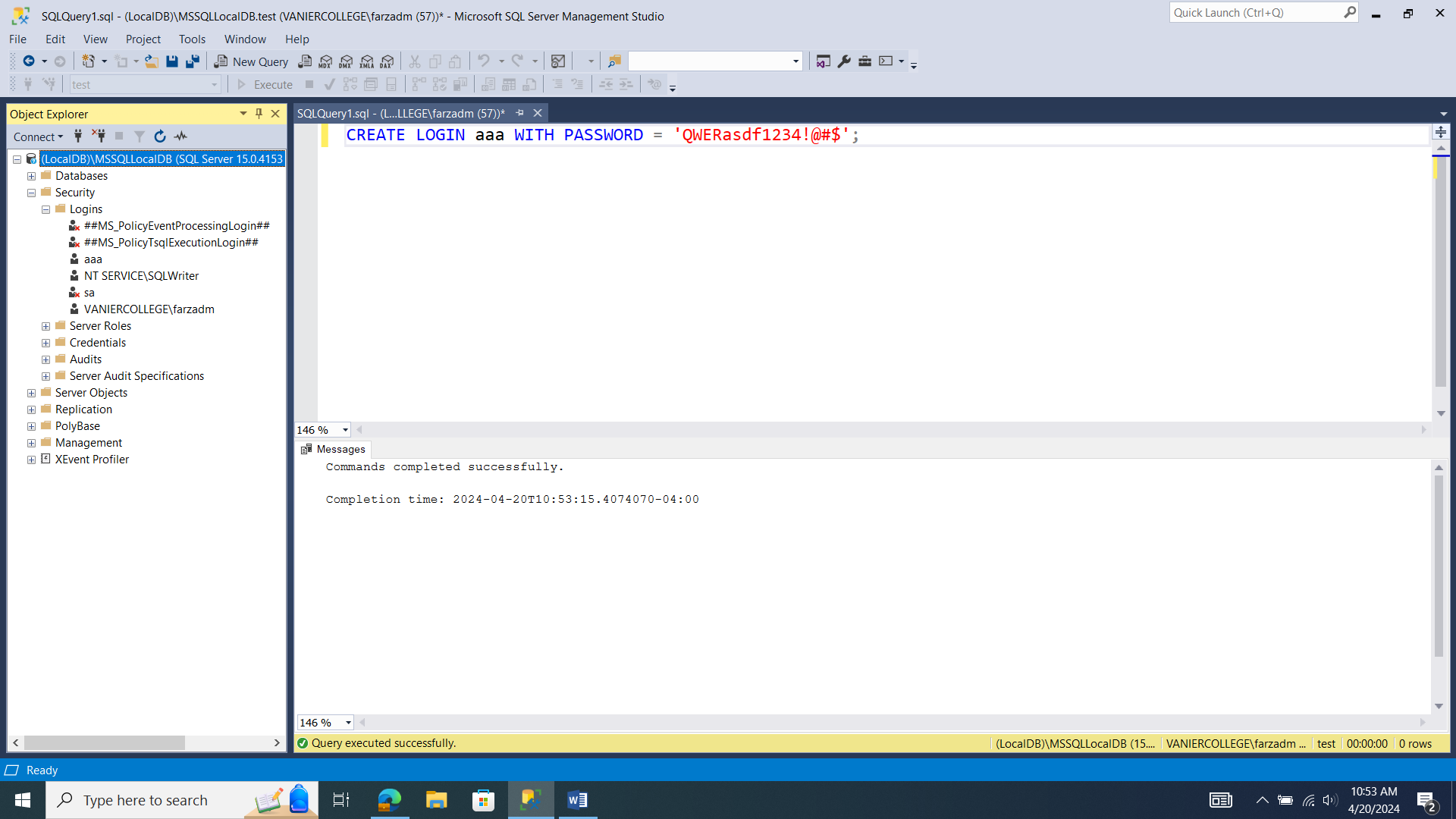
Example for creating the login:

CREATE LOGIN aaa WITH PASSWORD = 'QWERasdf1234!@#$';

Before:



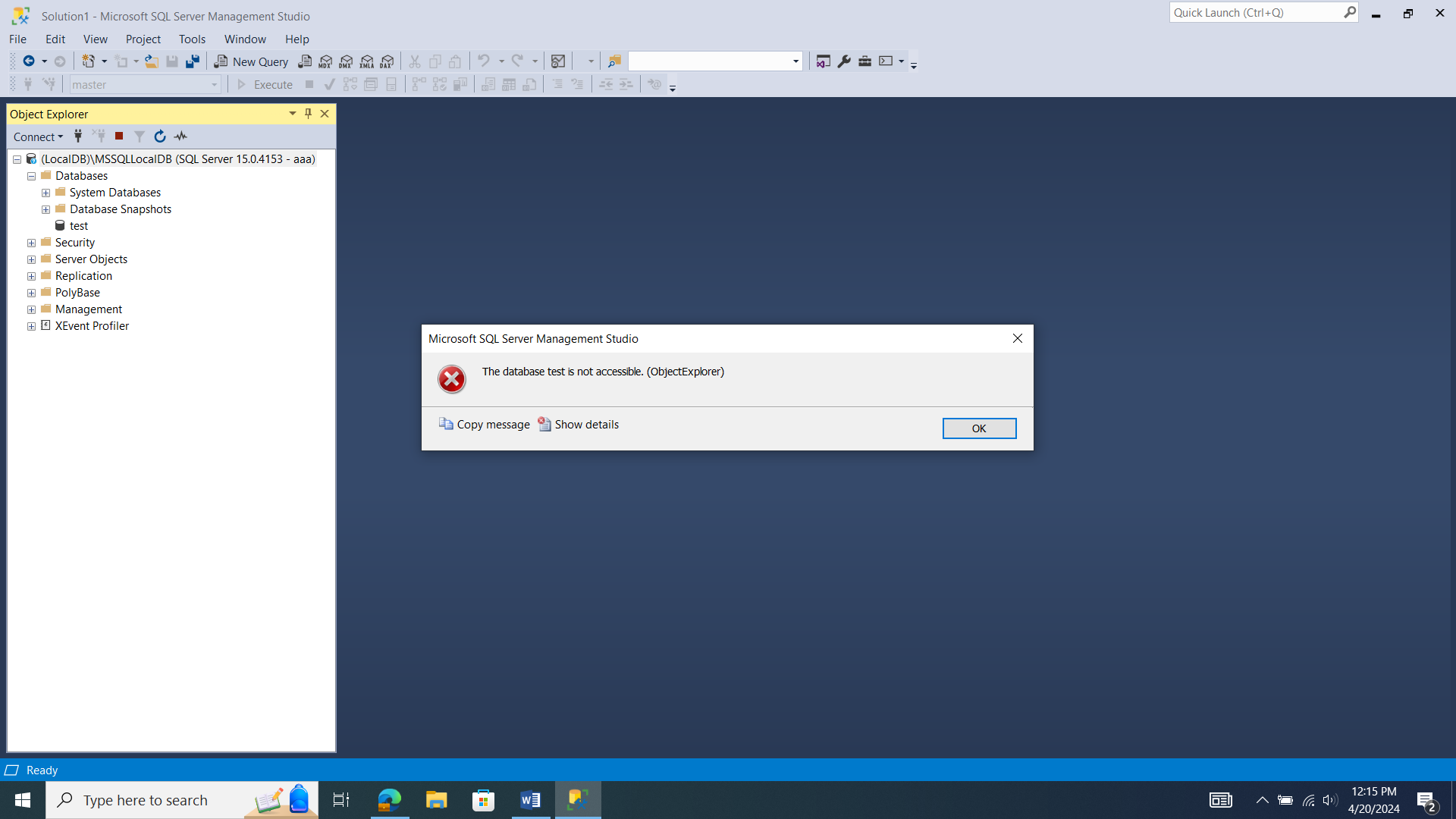
After:



Example for creating the login and user:

CREATE USER aaa FOR LOGIN aaa;

At this moment, if you create a database (using your own user), the new user aaa is not able to open it.



Dropping login and user:

DROP USER aaa;

DROP LOGIN aaa;

Dropping the “user”, yet he is able to login (however can’t have any transactions); while dropping the “login”, he is not able to login from beginning.

Altering the LOGIN: ALTER LOGIN statement is to modify properties of a login.

ALTER LOGIN YourLoginName WITH PASSWORD = 'NewPassword';

Example:

ALTER LOGIN aaa WITH PASSWORD = 'QWERasdf!@#$1234';

Note: In SQL Server, we cannot directly rename a login or a user using a single command like RENAME LOGIN or RENAME USER. However, we can achieve renaming by following a process involving creating a new login or user with the desired name, transferring permissions and properties, and then dropping the old login or user.

**Grant Permissions**: After creating the user, you might want to grant specific permissions to that user. Granting permissions to a user in SQL Server involves giving the user the ability to perform certain actions, such as reading data from tables, modifying data, executing stored procedures, and more. Permissions can be granted at various levels, including server-level, database-level, and object-level.

Granting Database-Level Permissions:

You can grant permissions at the database level using the GRANT statement followed by the permission type and the object on which the permission is being granted.

GRANT SELECT, INSERT, UPDATE, DELETE ON YourTableName TO YourUserName;

This example grants the user YourUserName permissions to perform SELECT, INSERT, UPDATE, and DELETE operations on the table YourTableName.

Example:

Create below table with your own user, for example, inside the Test database:

Use test;

create table city(cityid int, cityname varchar(10))

insert into city values(1, 'montreal'),(2, 'ottawa'),(3, 'toronto');

select \* from city

At this moment, if aaa logs in, he cannot see and select from the city table; but, if you grant permission he will be able to do that:

GRANT SELECT, INSERT, UPDATE, DELETE ON city TO aaa;

And then:

use test

select \* from city

To modify or alter granted database-level permissions in SQL Server, we may revoke the previously granted permissions, and then grant new permissions.

Granting Server-Level Permissions:

Server-level permissions control operations that affect an entire SQL Server instance.

GRANT VIEW ANY DATABASE TO YourUserName;

This example grants the user YourUserName permission to view any database on the server.

To grant the VIEW ANY DATABASE permission to a user, you need to have the necessary permissions yourself, usually either being a member of the sysadmin fixed server role or having the CONTROL SERVER permission.

To have administrative privileges, when opening SQL right click and "Run as Administrator"

Granting Object-Level Permissions:

You can grant permissions on specific objects, such as tables, views, stored procedures, etc.

GRANT EXECUTE ON Object TO YourUserName;

This example grants the user YourUserName permission to execute the stored procedure YourStoredProcedure.

First, create the stored procedure using your own user:

Use test

CREATE PROCEDURE dbo.Stored\_Pro

AS SELECT \* FROM city

EXEC Stored\_Pro

At this moment, you yourself, can execute the created stored procedure; however aaa cannot:

Use test

EXEC Stored\_Pro

But, after granting permission, he can execute the procedure:

GRANT EXECUTE ON dbo.Stored\_Pro TO aaa;

**WITH GRANT OPTION**

The "WITH GRANT OPTION" clause in SQL Server allows a user or role to grant the same permission to other users or roles. When you grant a permission to a user or role with the "WITH GRANT OPTION," it enables that user or role to pass on the permission to others.

GRANT SELECT ON SomeTable TO SomeUser WITH GRANT OPTION;

Here, the SELECT permission is granted on SomeTable to SomeUser with the "WITH GRANT OPTION." This means that SomeUser can not only select data from SomeTable but can also grant the SELECT permission on SomeTable to other users or roles.

Granting the Permission to Another User or Role:

GRANT SELECT ON SomeTable TO AnotherUser;

Because SomeUser was granted the SELECT permission with the "WITH GRANT OPTION," they can now pass on the SELECT permission to AnotherUser without needing additional permissions.

Use Cases:

* Delegating Permissions: You can delegate permission management to certain users or roles by granting permissions with the "WITH GRANT OPTION." This allows them to manage permissions for other users without having to involve the database administrator.
* Hierarchical Permission Structures: You can create hierarchical permission structures where higher-level roles grant permissions to lower-level roles, and those roles can, in turn, grant permissions to other roles or users.

Considerations:

* Security Implications: Be cautious when granting permissions with the "WITH GRANT OPTION," as it allows users to pass on permissions to others. Ensure that users granted with this option understand the implications and responsibilities of managing permissions.
* Principle of Least Privilege: Follow the principle of least privilege and only grant permissions with the "WITH GRANT OPTION" when necessary.
* Auditing: Regularly review permissions and audit trails to monitor the use of permissions granted with the "WITH GRANT OPTION" and ensure compliance with security policies.

By using the "WITH GRANT OPTION" selectively and responsibly, you can empower users to manage permissions effectively while maintaining a secure and compliant database environment.

Example:

With your own user:

Use test

CREATE LOGIN bbb WITH PASSWORD = 'QWERasdf1234!@#$';

CREATE USER bbb FOR LOGIN bbb;

At this moment, bbb cannot see and select from table City.

Grant permission to aaa as:

GRANT SELECT ON city TO aaa WITH GRANT OPTION;

And then, from aaa user:

Use test

GRANT SELECT ON city TO bbb;

Now, bbb can see and select from table City:

use test

SELECT \* from city

Revoking Permissions:

Similarly, you can revoke permissions using the REVOKE statement.

REVOKE SELECT ON YourTableName FROM YourUserName;

This example revokes the SELECT permission on the table YourTableName from the user YourUserName.

Use test

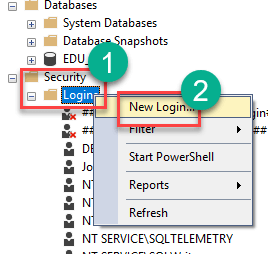
REVOKE SELECT ON city FROM aaa;

And then, aaa cannot select from city anymore; however, yet can insert into.

When granting permissions, it's essential to follow the principle of least privilege, granting only the permissions necessary for the user's intended tasks.

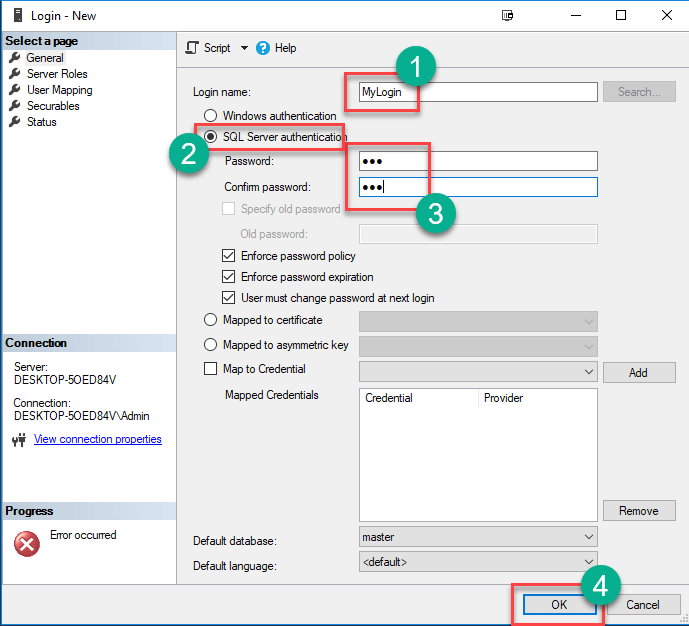
**Creating login in SQL Server through SQL Server Management Studio** **(SSMS):**

**Step 1)**To create login SQL server, Navigate to Security > Logins

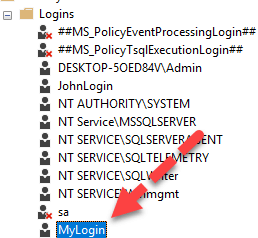
****

Step 2)In the next screen, Enter

1. Login Name
2. Select SQL Server authentication
3. Enter Password for MySQL create user with password
4. Click Ok

****

Step 3)Login is created

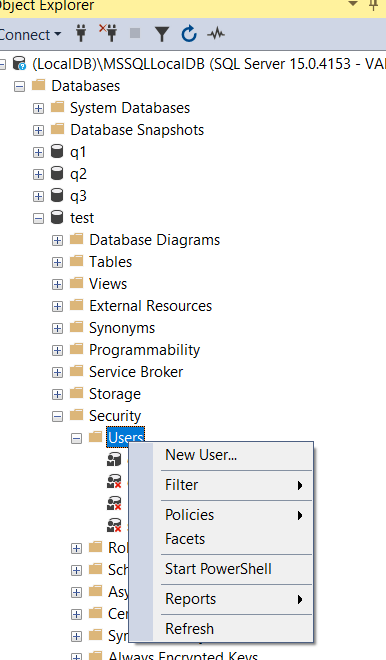


## Creating User in SQL Server Management Studio (SSMS):

We will create a user for the Test database.

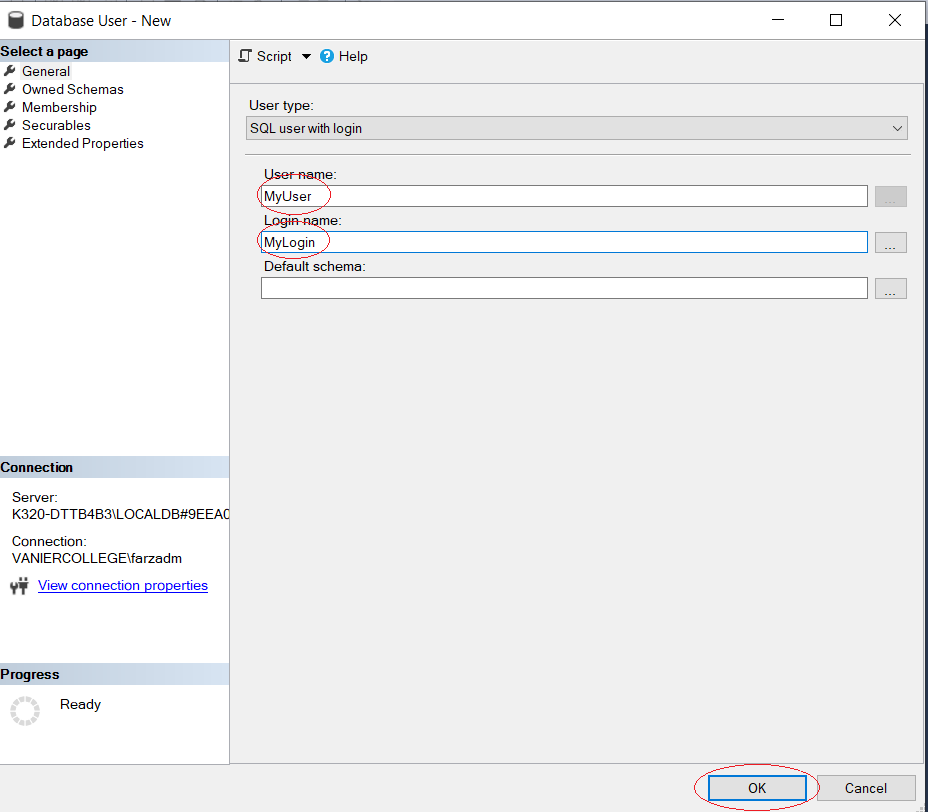
**Step 1) Connect to SQL server to create new user**

1. Connect to SQL Server then expand the Databases folder from the Object Explorer.
2. Identify the database for which you need to create the user and expand it.
3. Expand its Security folder.
4. Right-click the Users folder then choose “New User…”

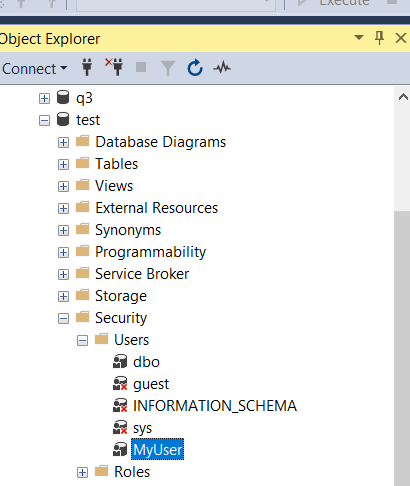


Step 2) Enter User details

1. Enter desired User name
2. Enter the Login name (created earlier)
3. Click OK

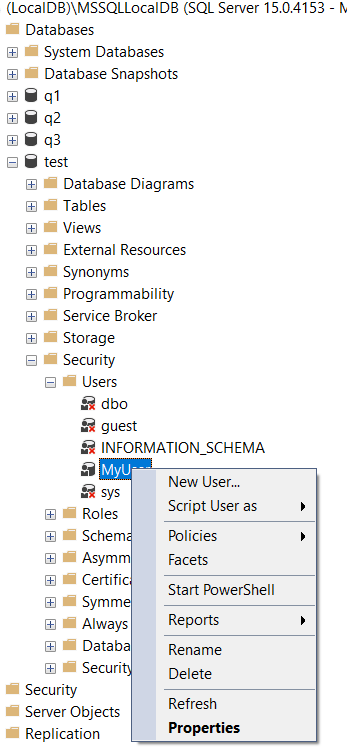


**Step 3) User will be created**



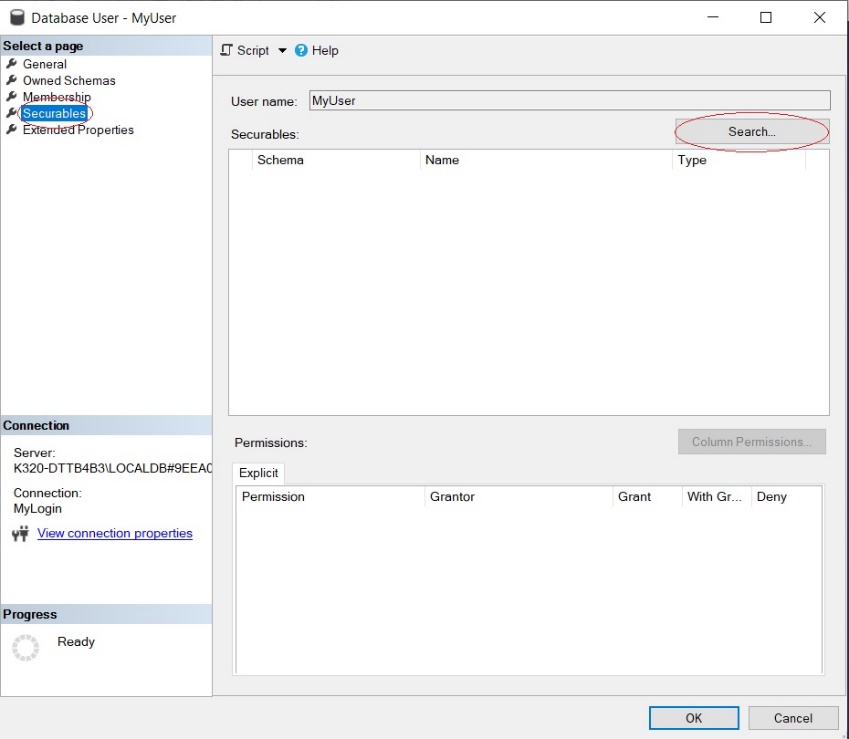
## Assigning Permissions in SQL Server Management Studio (SSMS)

**Step 1)**Connect to your SQL Server instance and expand the folders from the Object Explorer as shown below. Right click on the name of the user, that is, MyUser then choose Properties. (this happens under your own user – the user created the new user)



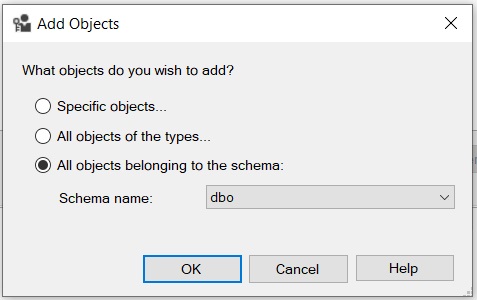
Step 2)In the next screen,

1. Click the Securables option from the left.
2. Click on Search



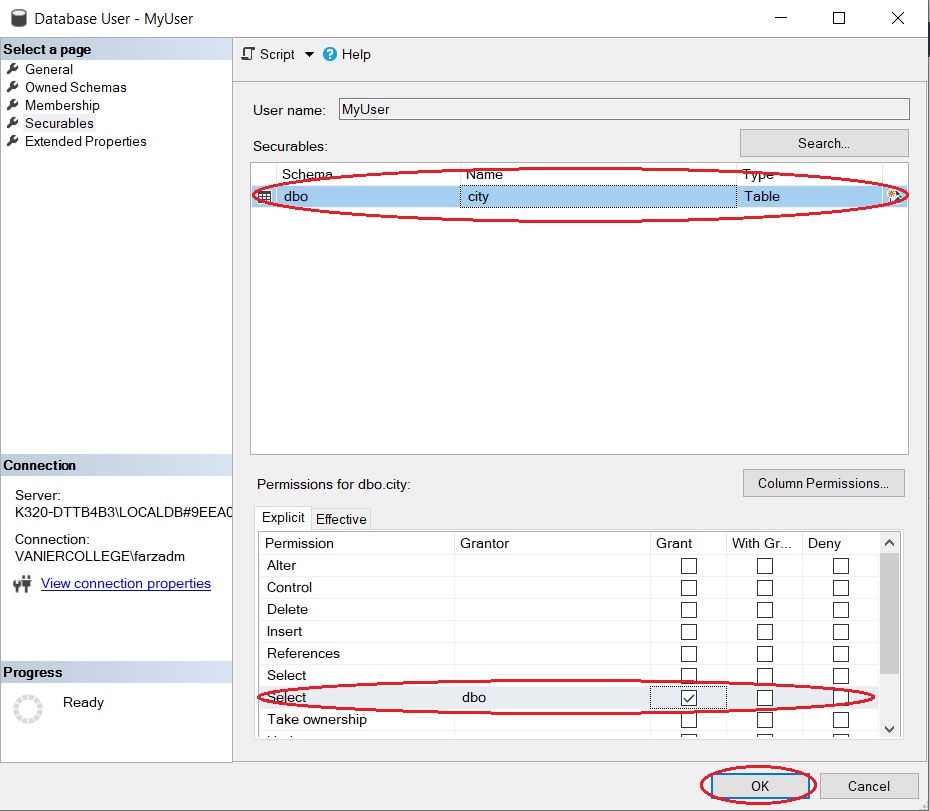
Step 3)In the next window,

1. Select “All Objects belonging to the Schema.”
2. Select Schema name as “dbo”
3. Click OK



Step 4)

1. Identify Table you want to Grant Permission
2. In Explicit Permission select Grant
3. Click Okay



**Step 5)**The user MyUser is granted SELECT permission on table City.

Note: similarly, you may plan to Revoke (Deny), or With Grant the assigned permissions.