NIBM WORLD WIDE

SOFTWARE ENGINEERING

DATABASE MANAGEMENT SYSTEMS

DATA CONTROL LANGUAGE (DCL)

National Institute of Business Management

Lesson 10 – Data Control Language (DCL)

Data Control Language (DCL)

- These statements control the access of the data in the database and determines how, when, and whom can manipulate data.
- They provide commands to specify access rights to tables, and views.
- The Data Control Language (DCL) allows database administrators to configure security access to relational databases.
- DCL is the simplest of the SQL subsets, as it consists of only three commands: GRANT, REVOKE, and DENY.
- Combined, these three commands provide administrators with the flexibility to set and remove database.

GRANT, REVOKE, DENY

Before we use above commands. We need to create new users. To create new users it is always better to login using Sa administrator.

Follow following steps.

Step 01: Login as user **Sa** using SQL Server Authentication. You will get an error as Login Failed or Login disabled.

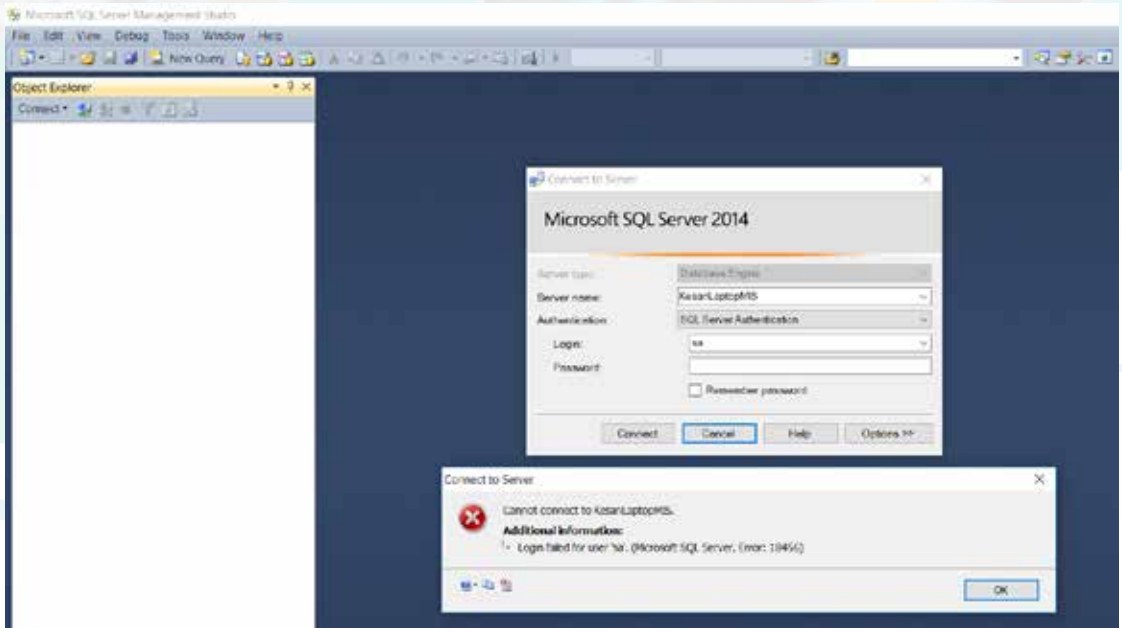


Figure 10.0.1 Error when try to Login as user Sa

Step 02: Login using Windows Authentication mode (Normal Way) and right click on the Server Connection and click Properties.

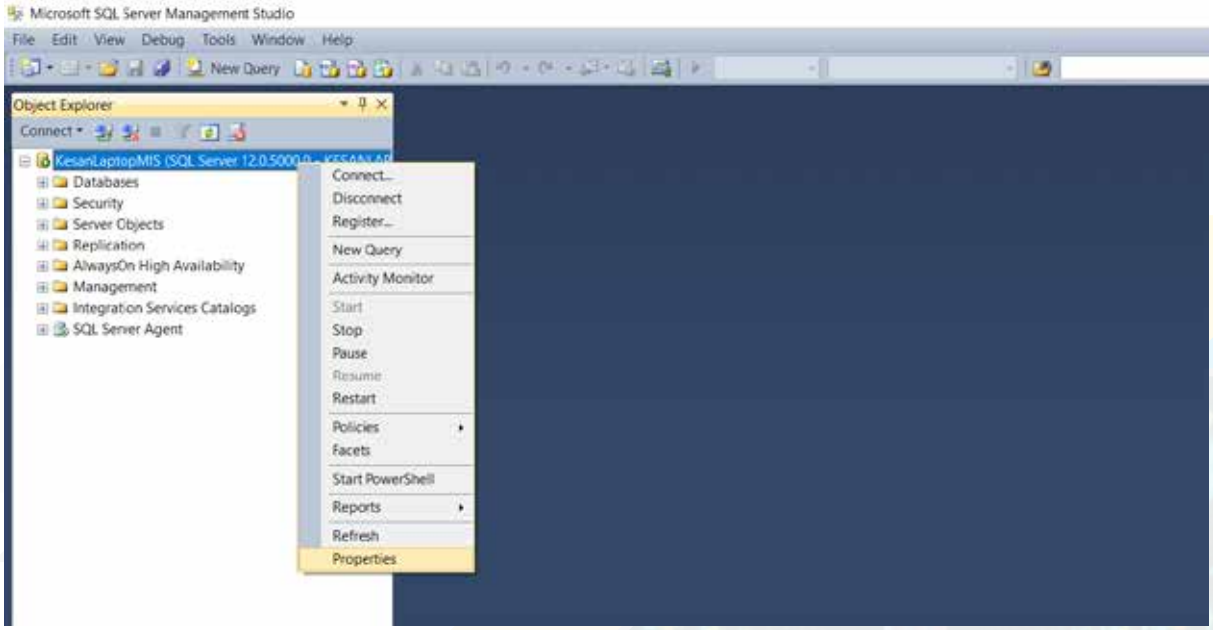


Figure 10.0.2 Login using Windows Authentication mode

Step 03: Under the Properties click Security tab. Then change the Server Authentication mode to SQL Server and Windows Authentication mode and click OK.

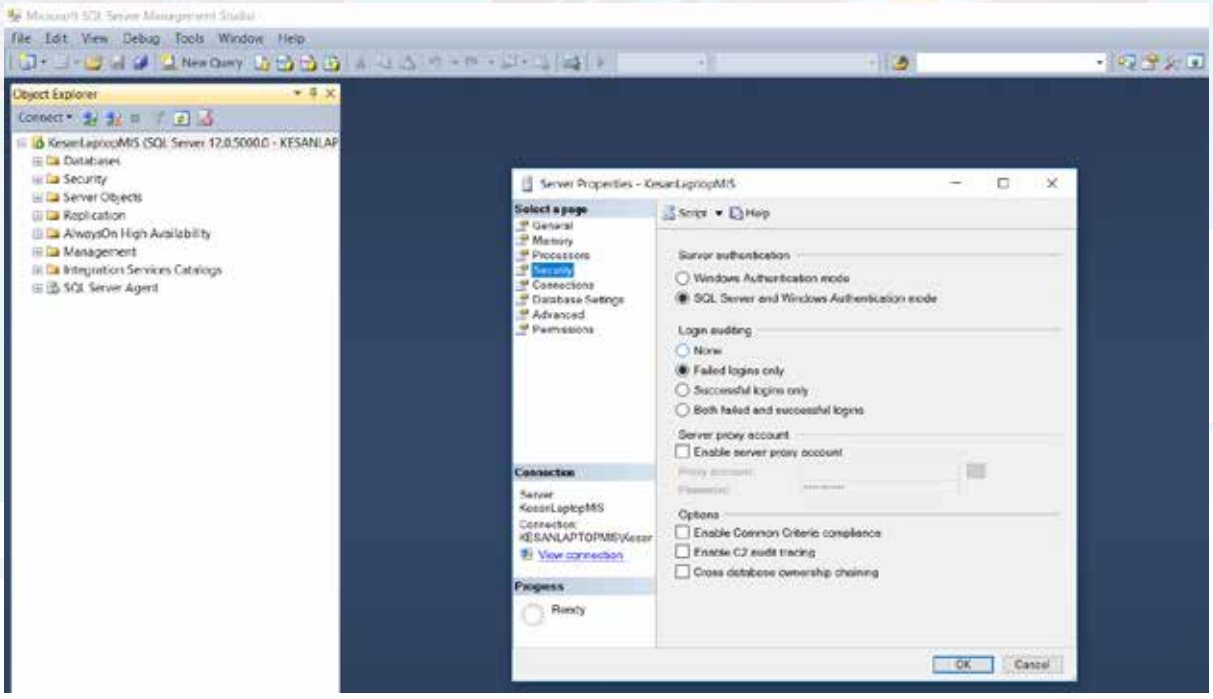


Figure 10.0.3 Change to Server Authentication mode

Step 04: Then you need to restart the SQL Server. Close it and open again. Or else press + R. Then you will get the Run command. Type ervices.msc and click ok. Find SQL Server (MSSQLSERVER) right click then restart.

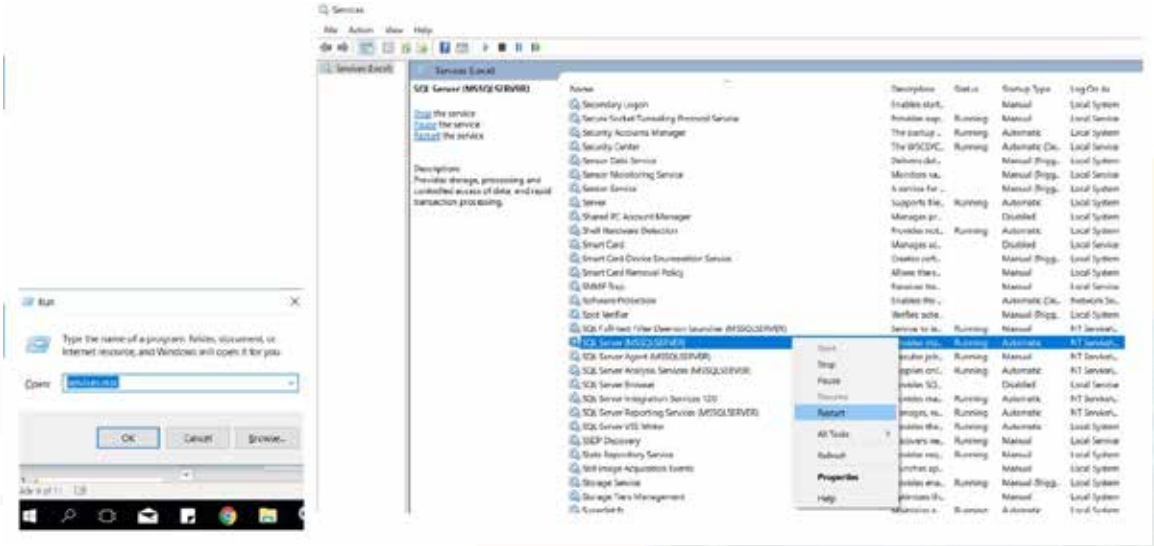


Figure 10.0.4 Restart the SQL Server

Step 05: Login to SQL Server again using the Windows Authentication Mode (Normal Way). Expand Security, Expand Logins go to **Sa** user. Right click on the user and click Properties.

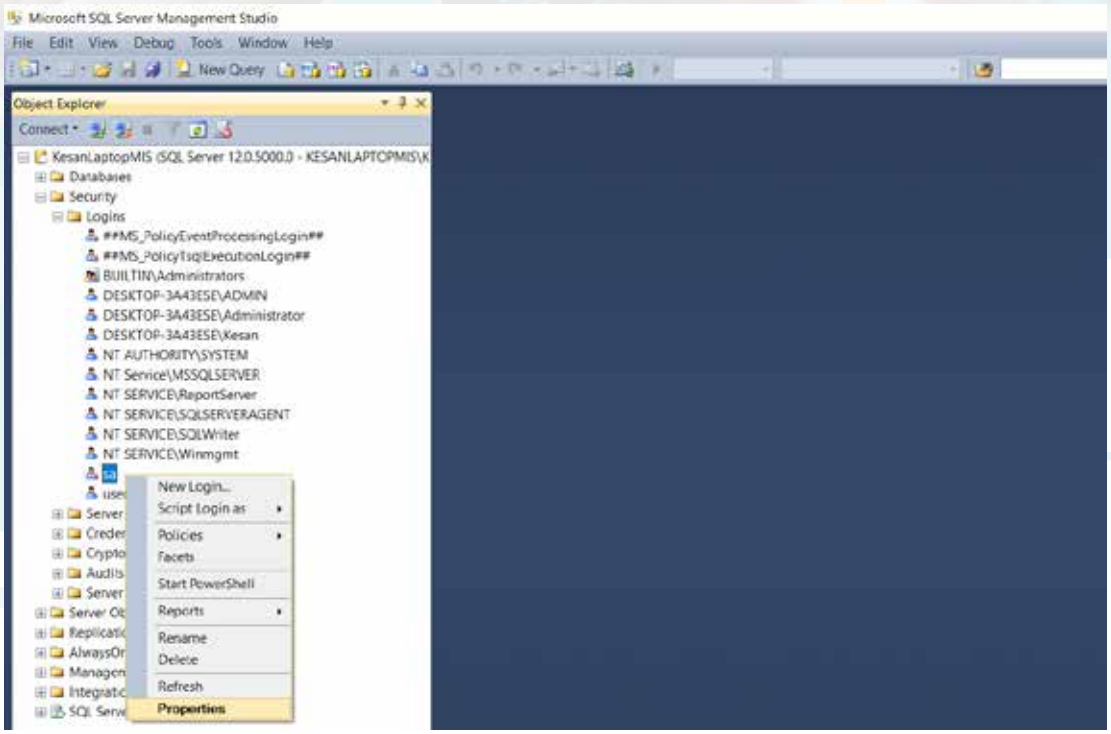


Figure 10.0.5 Expand Logins go to Sa user

Step 06: Go to Status, then Enabled the Login.

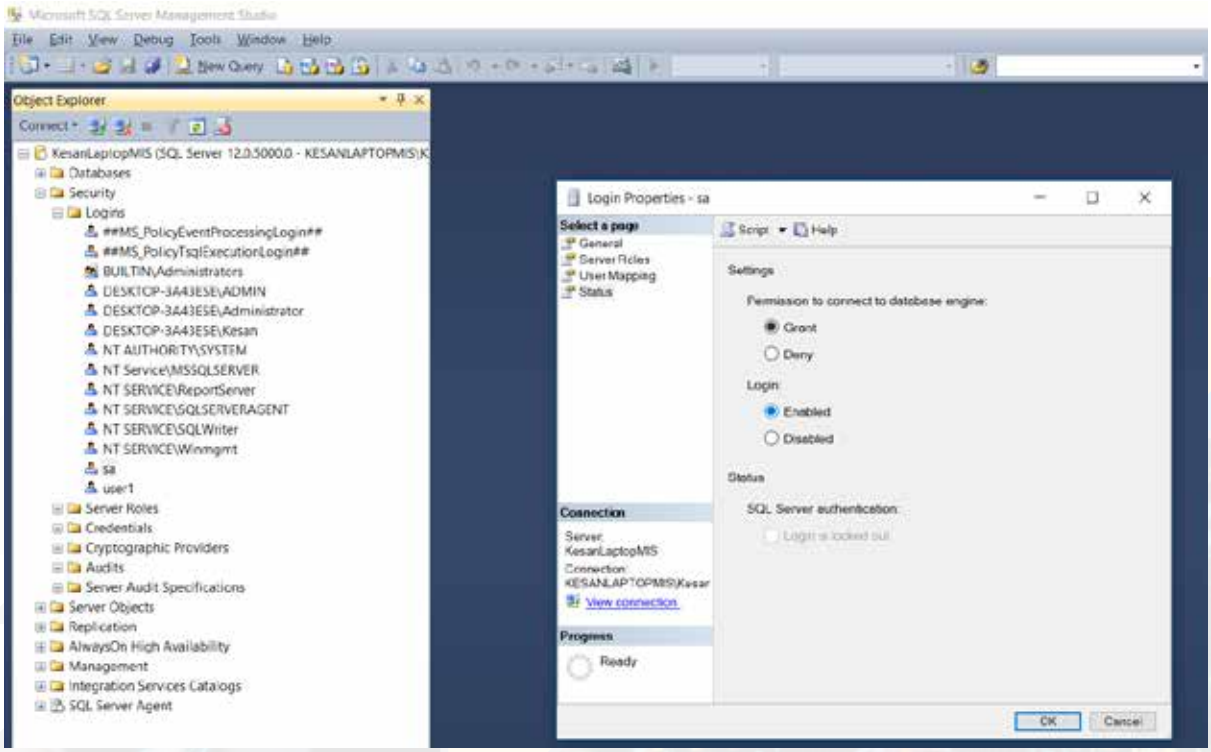


Figure 10.0.6 Enabled the Login

Step 07: Then go to General and change the password and confirm password. Give Simply 123 as the password (If this is SQL 2019 you have to match password policies). The click Ok.

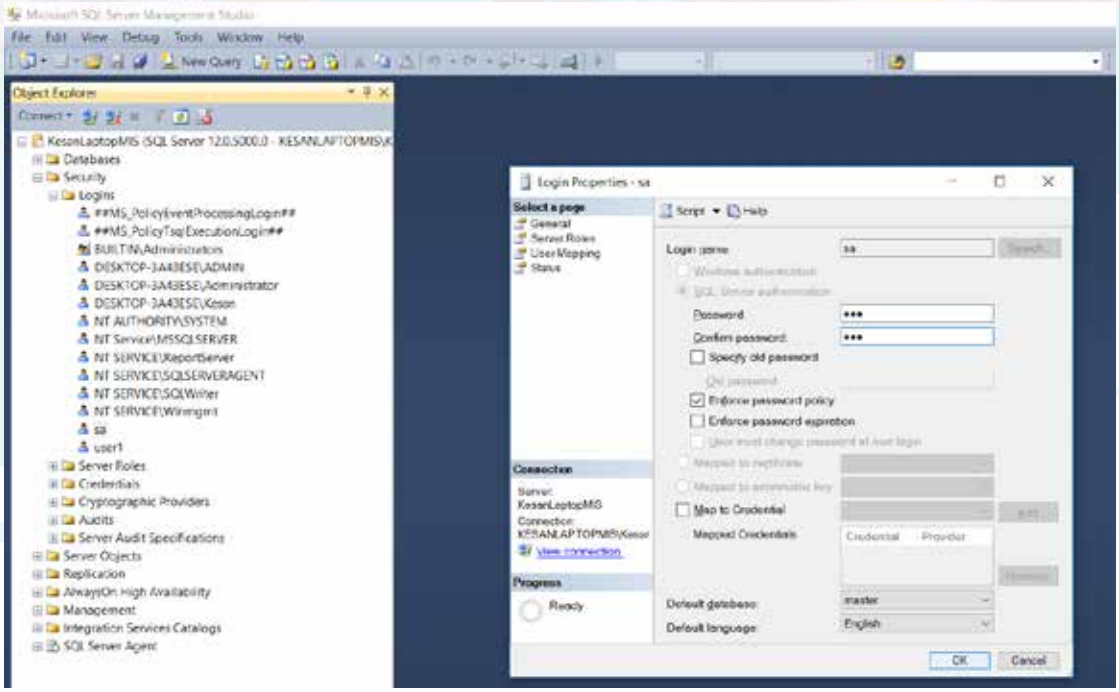


Figure 10.0.7 Change the password

Step 08: Then connect to Database Engine again and Login using SQL Server Authentication Mode.

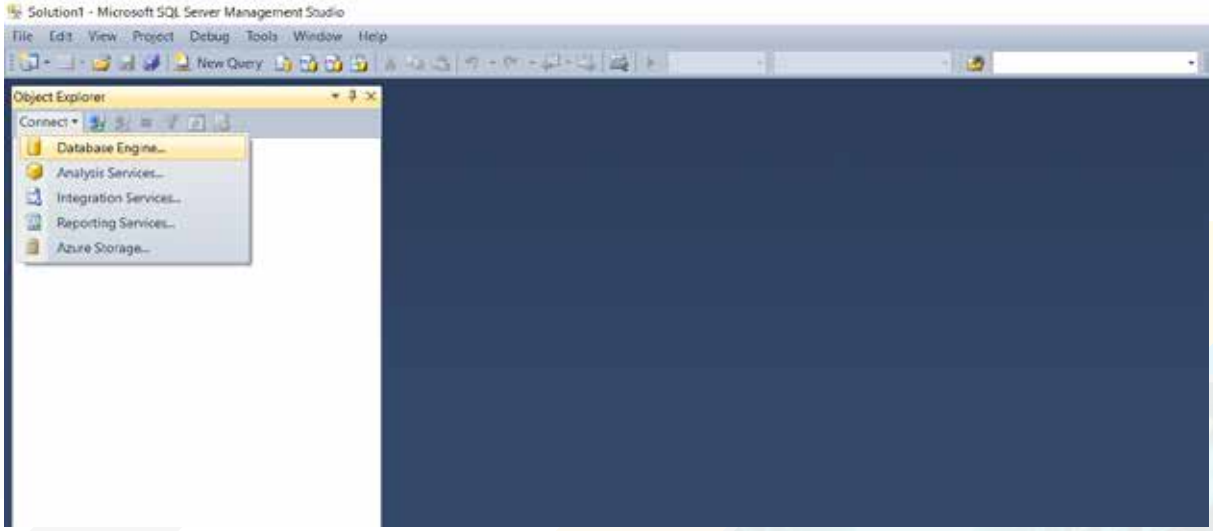


Figure 10.0.8 Connect to Database Engine

Step 09: Select SQL Server Authentication. Give Login Id as **sa** and Password as **123** (Your password). Then click Connect.

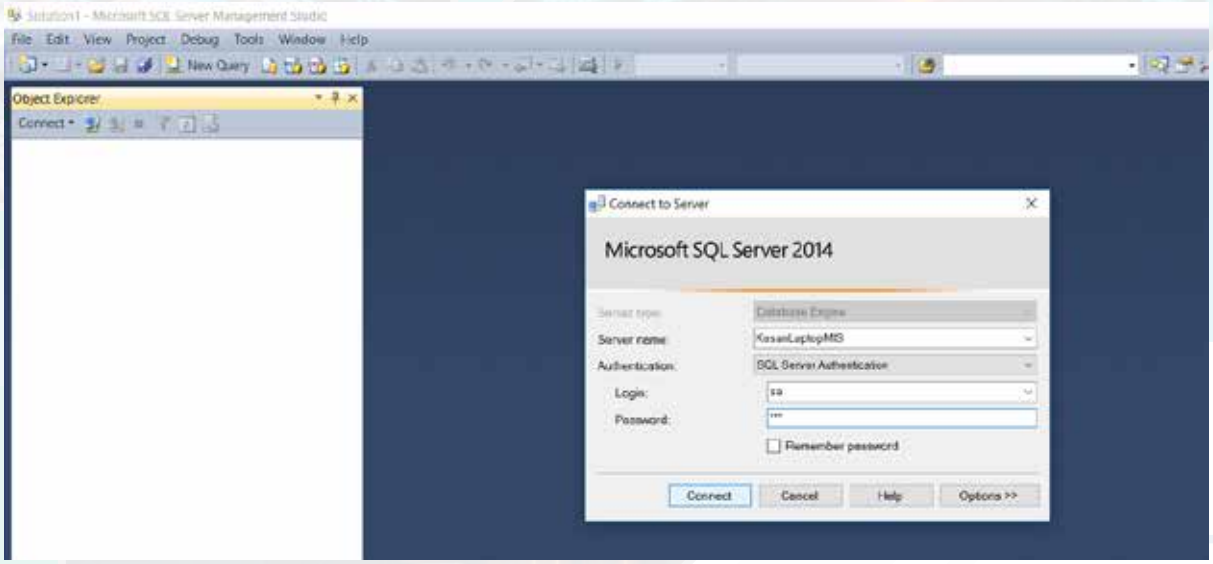


Figure 10.0.9 Login as Sa user

Step 10: Once you connect as **Sa** user create a database and table.

```
Create database Sample;
Use Sample;
Create table Client
(Client_Id varchar(10) Primary Key, Name varchar(20),
Address varchar(20), TP int);
insert into Client values ('C1', 'Gayan', 'Col07', 0772569014);
```

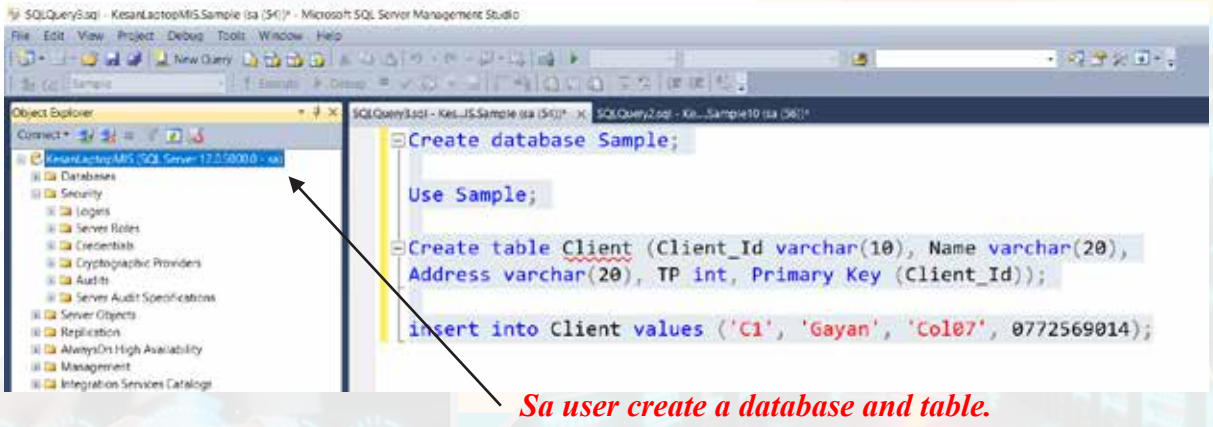


Figure 10.0.10 create a database and table as Sa user

Step 11: Expand the Security, then expand the Logins. Right click on the Logins and click New Login.

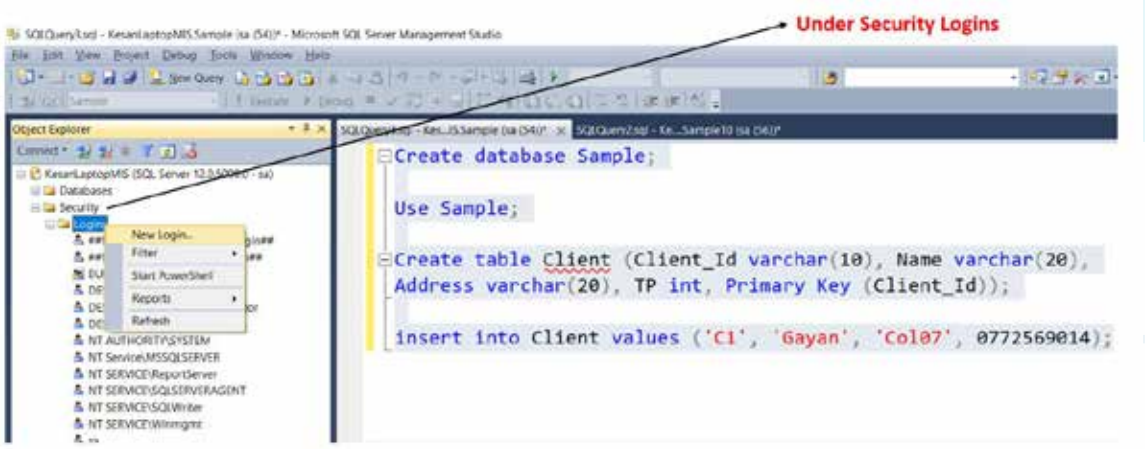


Figure 10.0.11 Create a new Login

Step 12: Change the Mode to SQL Server Authentication mode. Give the Login name as **user1** and password as **123**. (If this is SQL 2019 you have to match password policies).

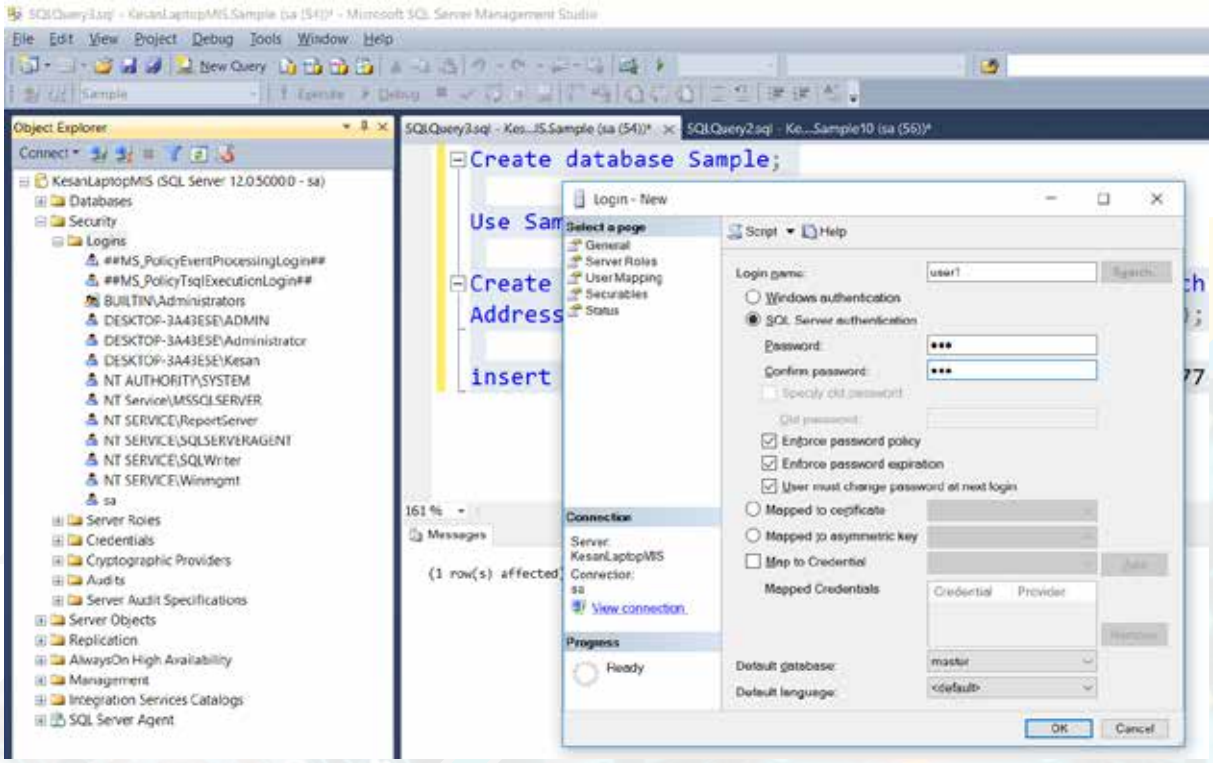


Figure 10.0.12 Create User1

Step 13: Connect to Database Engine again. Login as user1 by giving 123 as password. Once you click connect you have to the change password. Give 456 as the password.

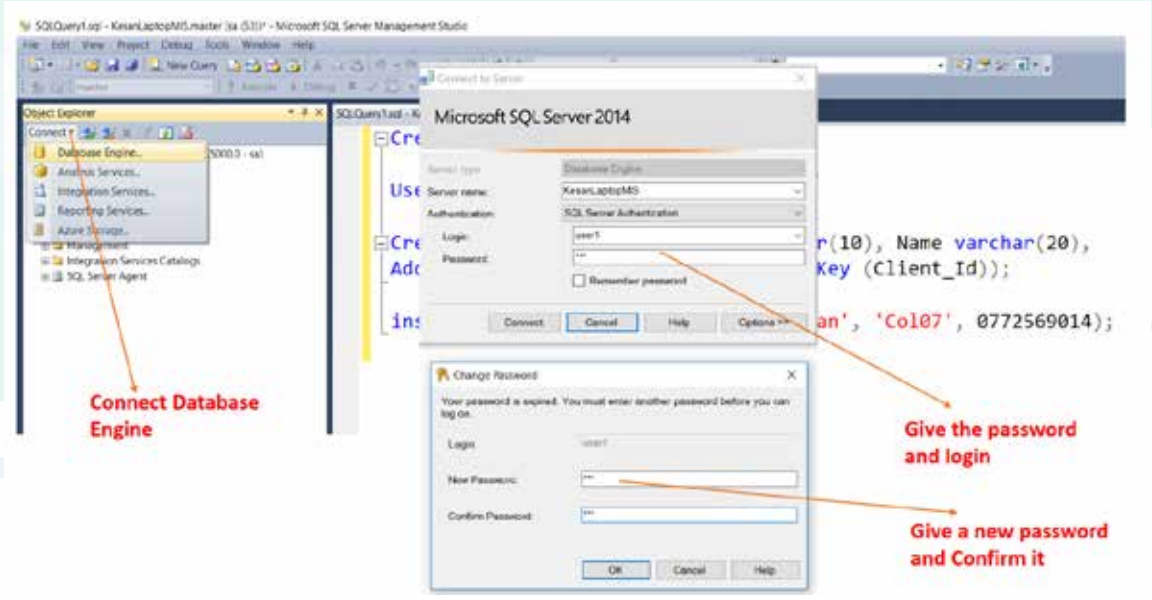


Figure 10.0.13 Login as user1

Step 14: Now you can see both Sa and user1 users are login to the SQL Server. Under the user1 login click the Sample Database. You will get an error as follow because user1 doesn't have any permission to access Sample database.

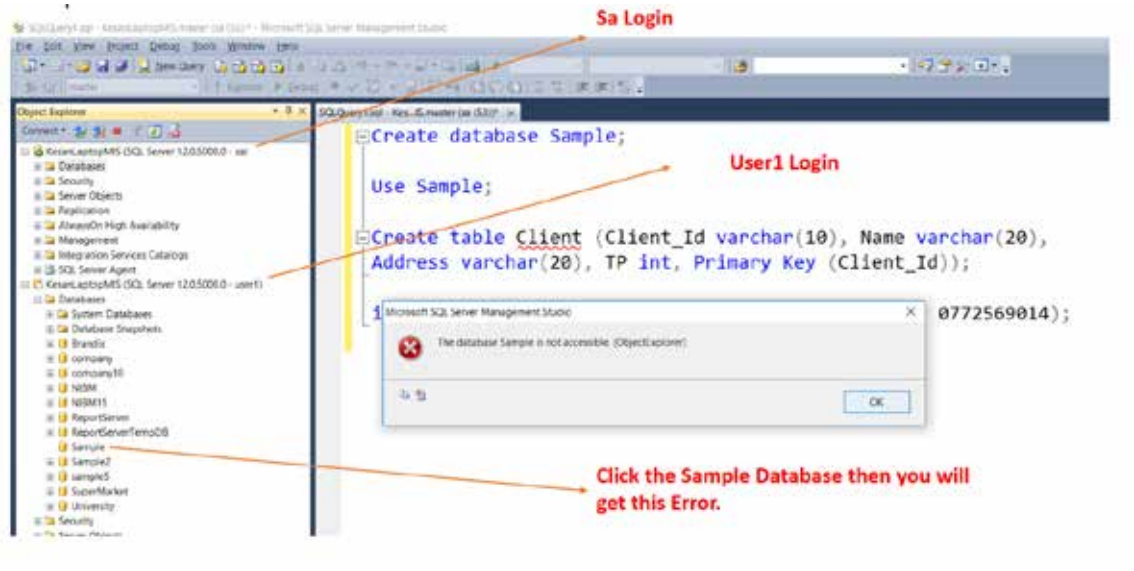


Figure 10.0.14 View both Sa and user1

Step 15: Then under Sa user expand the Databases, go to Sample Database expand it, go to Security under the Sample Database, expand it. Go to Users, Right Click on the Users, click New User.

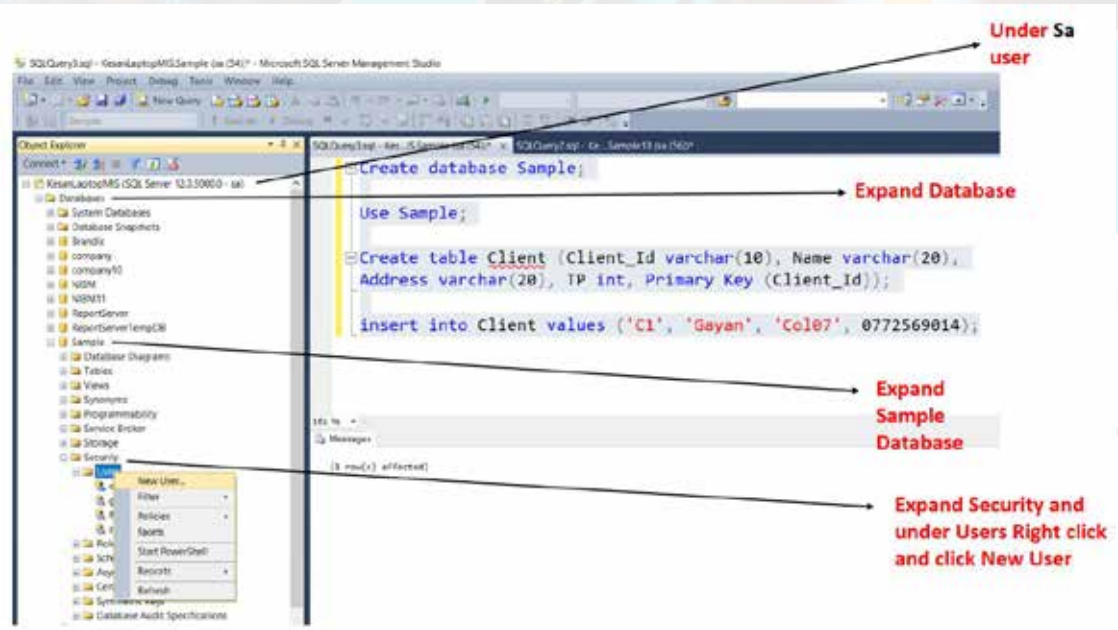


Figure 10.0.15 Create New User

Step 16: Give user1 as User Name. Then click browse on Login name.

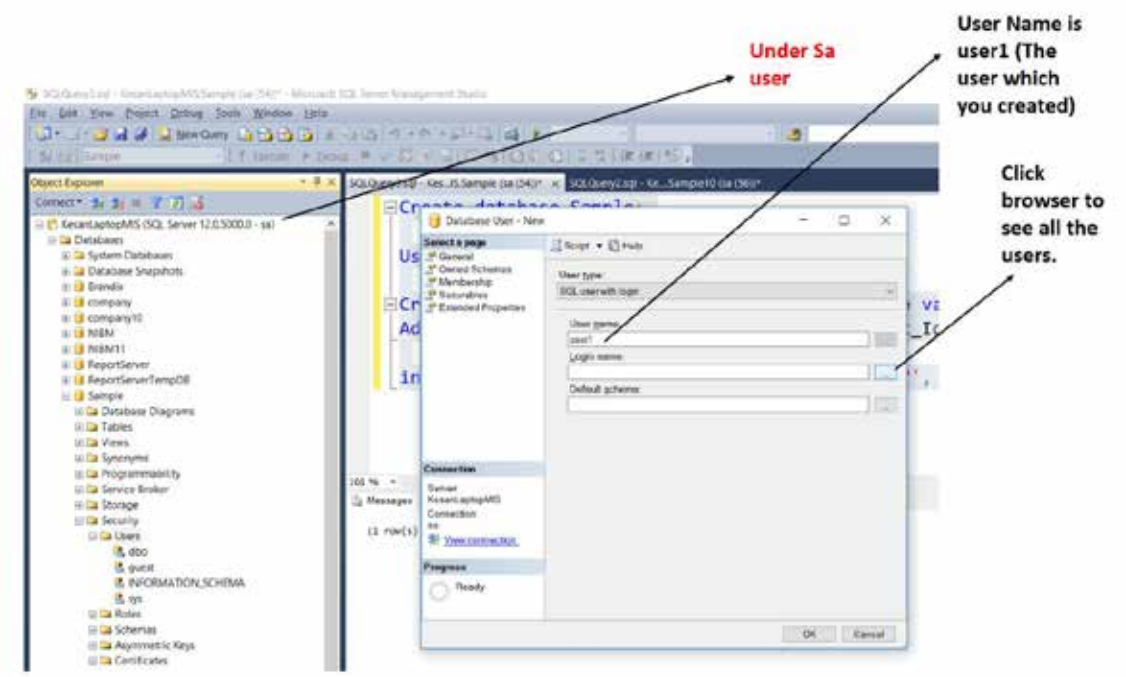


Figure 10.0.16 Create New User by Browsing

Step 17: Click Browse then select user1.

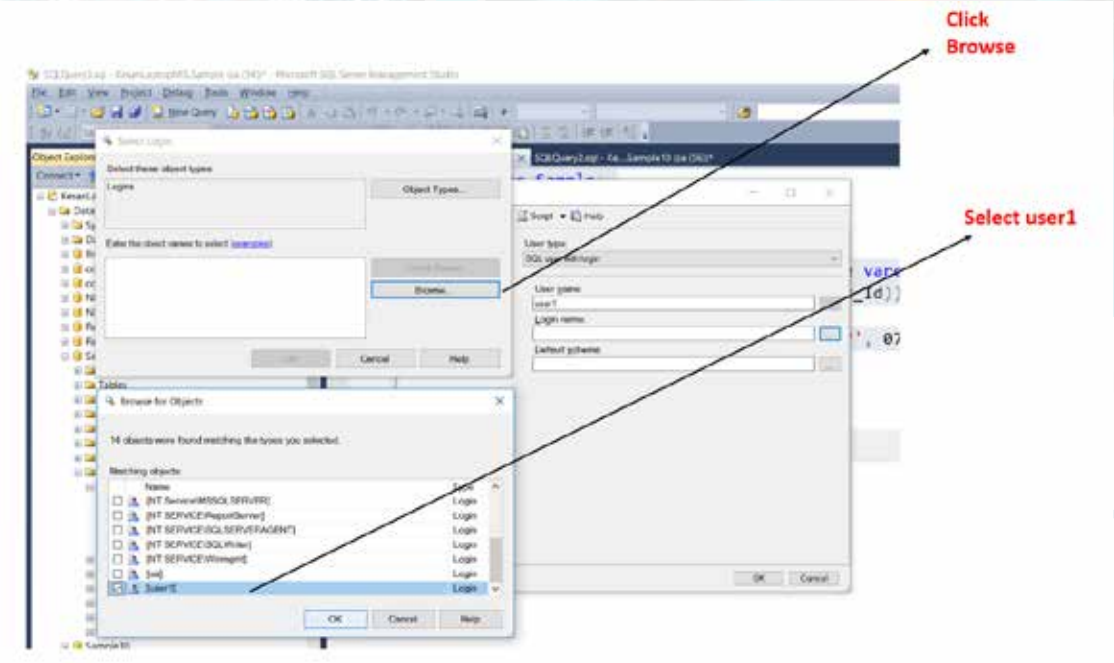


Figure 10.0.17 Select user1

Step 18: Now you can see user1 is added to the Sample Database.

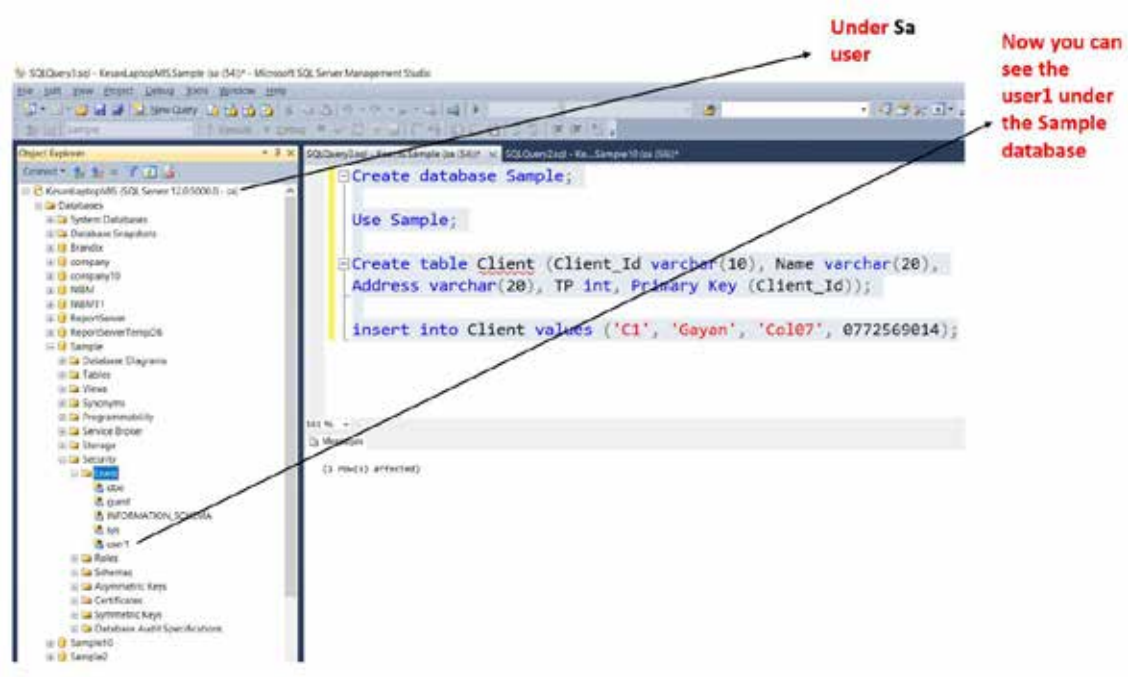


Figure 10.0.18 User1 is added to the Sample Database

Adding Permissions with the GRANT Command

The GRANT command is used by administrators to add new permissions to a database user.

Syntax:

Grant [privilege]
on [object]
to [user]
[with grant option]

Privilege — Can be either the keyword ALL (to grant a wide variety of permissions) or a specific database permission or set of permissions. Examples include CREATE DATABASE, SELECT, INSERT, UPDATE, DELETE, EXECUTE and CREATE VIEW.
Object — Can be any database object. Typically, the object will be either a database, function, stored procedure, table or view.
User — Can be any database user.

If you include the optional **WITH GRANT OPTION** clause at the end of the GRANT command, **you not only grant the specified user the permissions defined in the SQL statement but also give the user the ability to grant those same permissions to other database users. For this reason, use this clause with care.**

Give only Select permission to user1

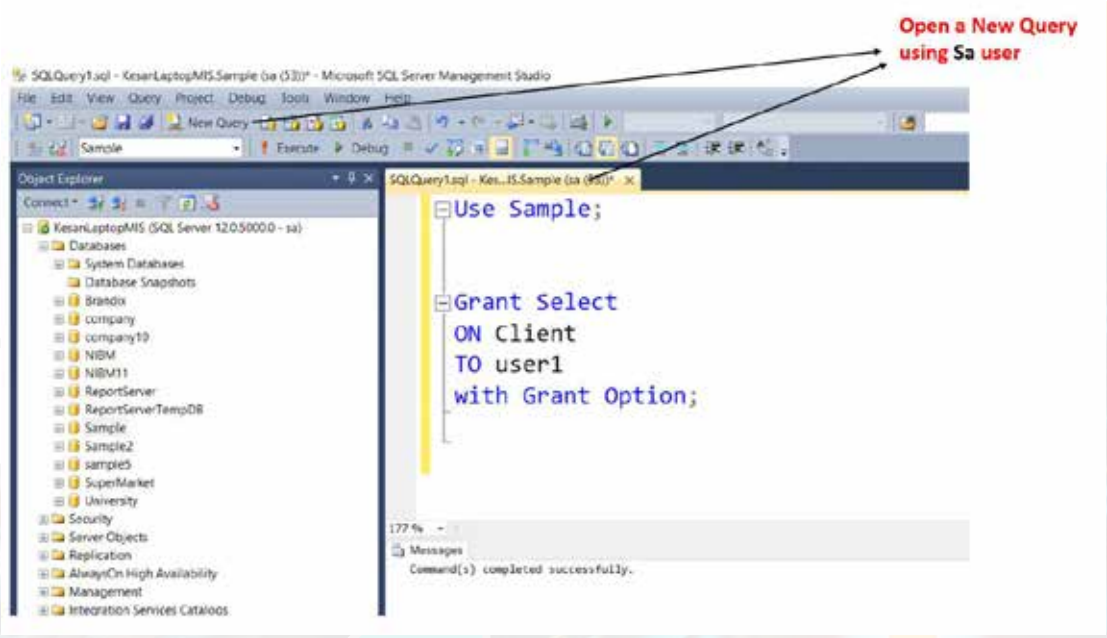
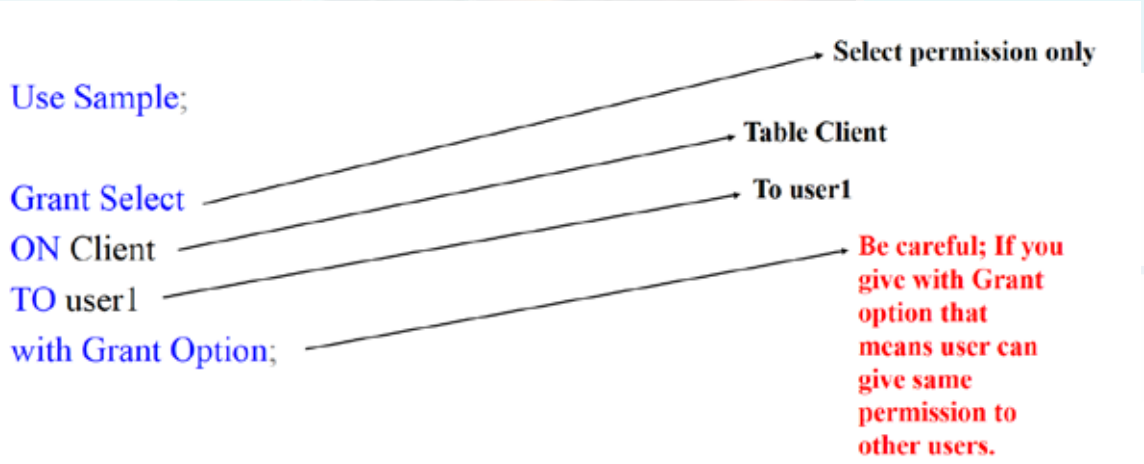


Figure 10.0.19 Give only Select permission to user1

Example 1:



Now click on the user1 connection then click New Query. Check whether you can select the Client table from the Sample database.

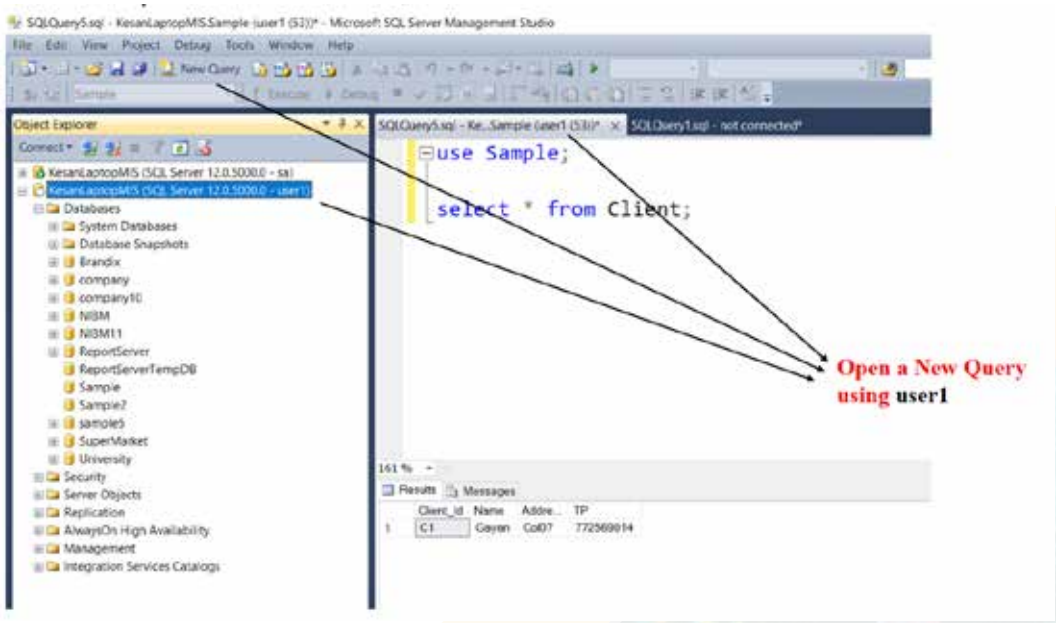


Figure 10.0.20 Open a New Query using user1

Now try to run the Insert command. You will get an error because you don't have insert permission on the Client table.

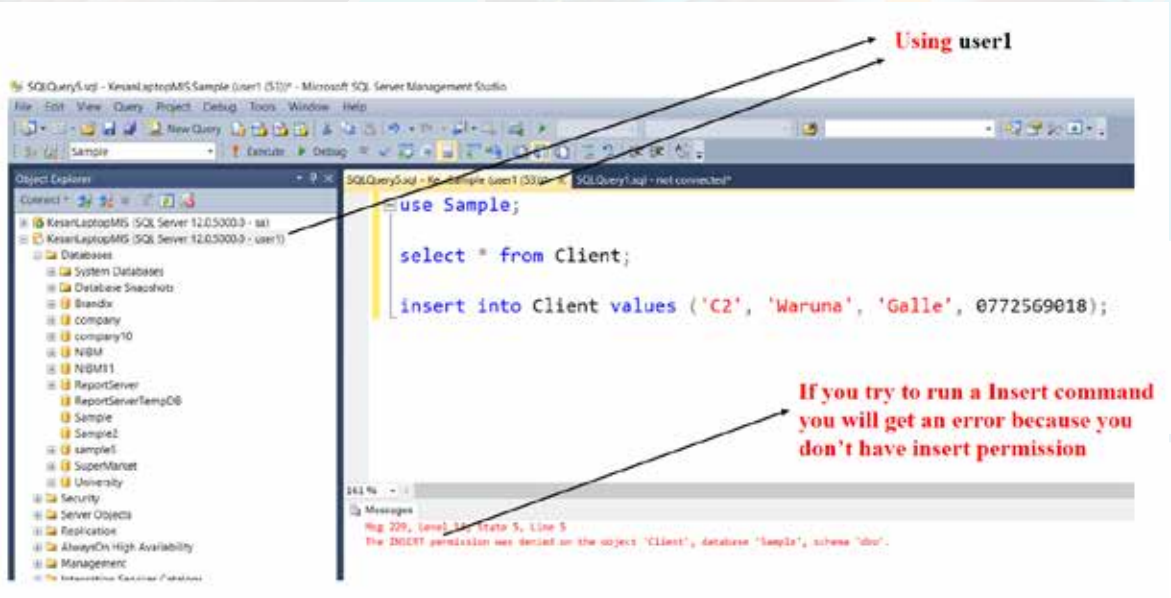


Figure 10.0.21 Try to run the Insert command

Example 2:

```
Use Sample;

Grant Select, Insert, Update
ON Client
TO user1
with Grant Option;
```

Select, Insert, Update permissions only

Table Client

To user1

Be careful; If you give with Grant option that means user can give same permission to other users.

Give Select, Insert and Update permissions to user1

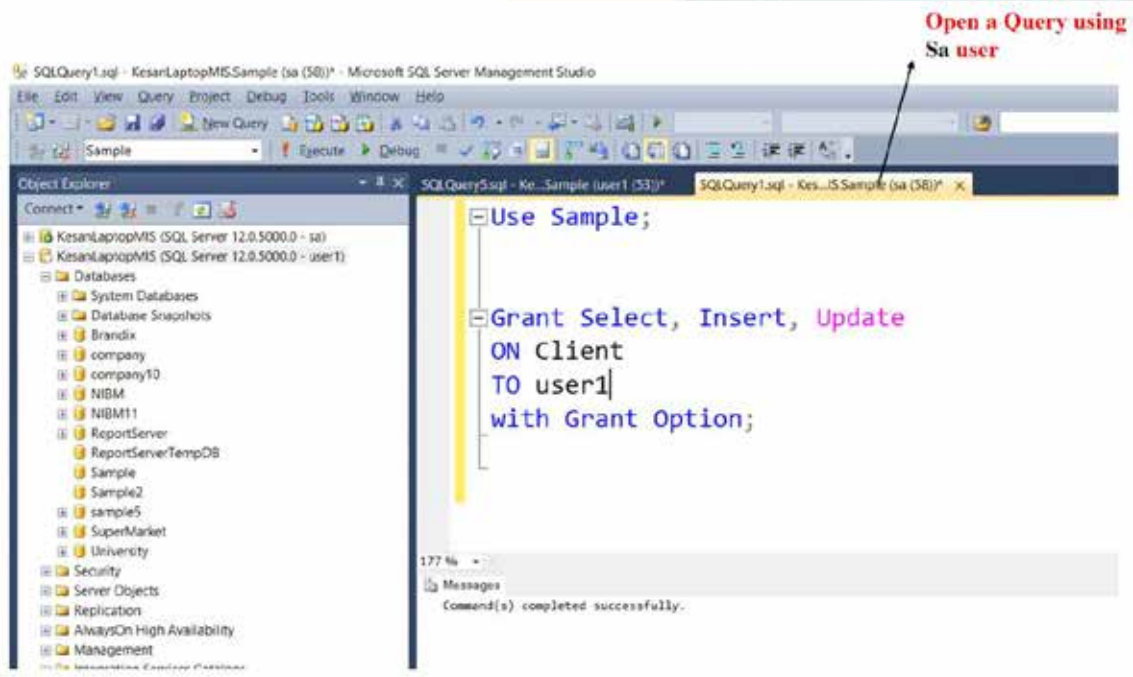


Figure 10.0.22 Give Select, Insert and Update permissions to user1

Now you can Select, Insert and Update the data. But still you can't delete

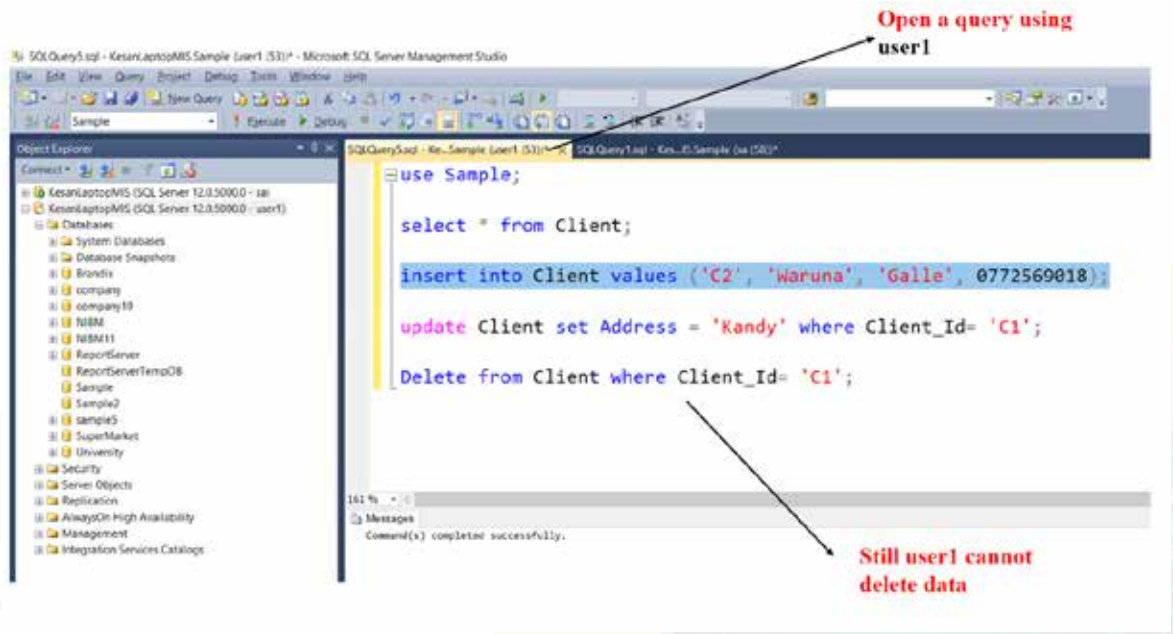
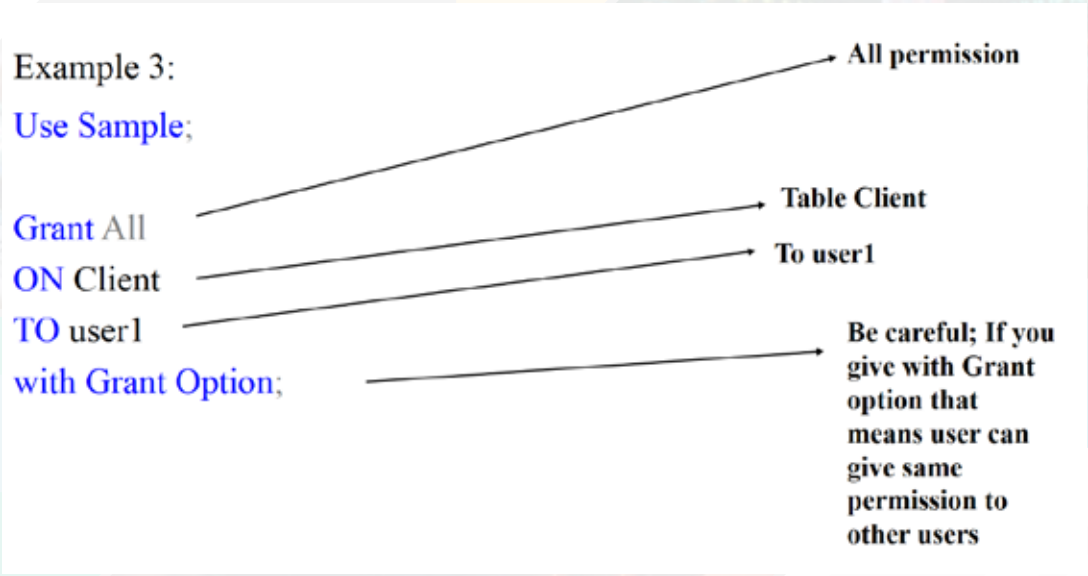


Figure 10.0.23 Select, Insert and Update the data using user1



Now user1 can Select, Insert, Update and Delete.

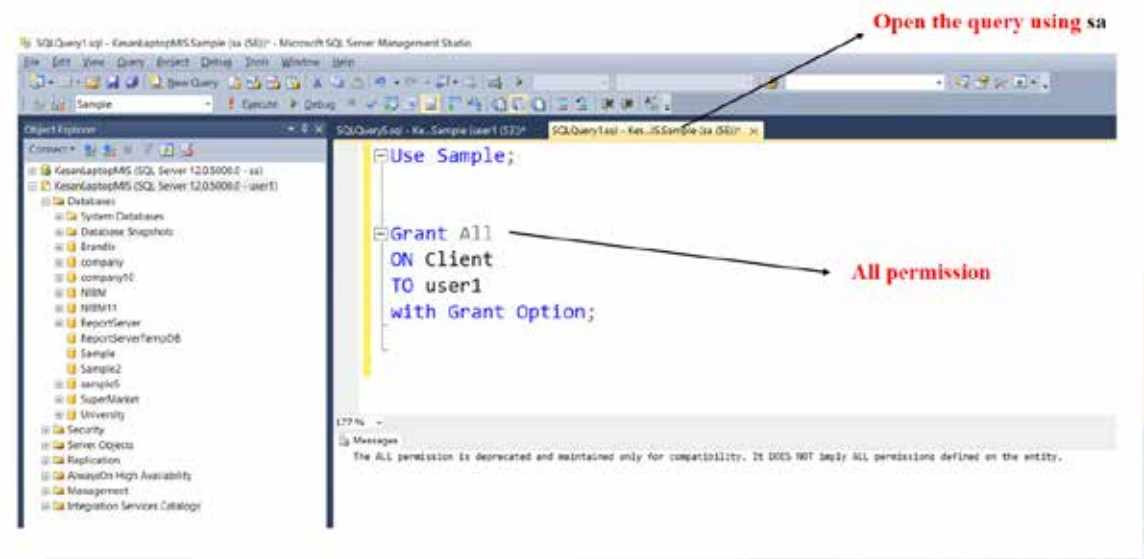


Figure 10.0.24 Select, Insert, Update and Delete using user1

Revoke Permission

The REVOKE command is used to remove database access from a user previously granted such access. The syntax for this command is defined as follows:

Syntax:

REVOKE [GRANT OPTION FOR] [permission]
ON [object]
FROM [user]
[CASCADE]

Permission — Specifies the database permissions to remove from the identified user. The command revokes both GRANT and DENY assertions previously made for the identified permission.

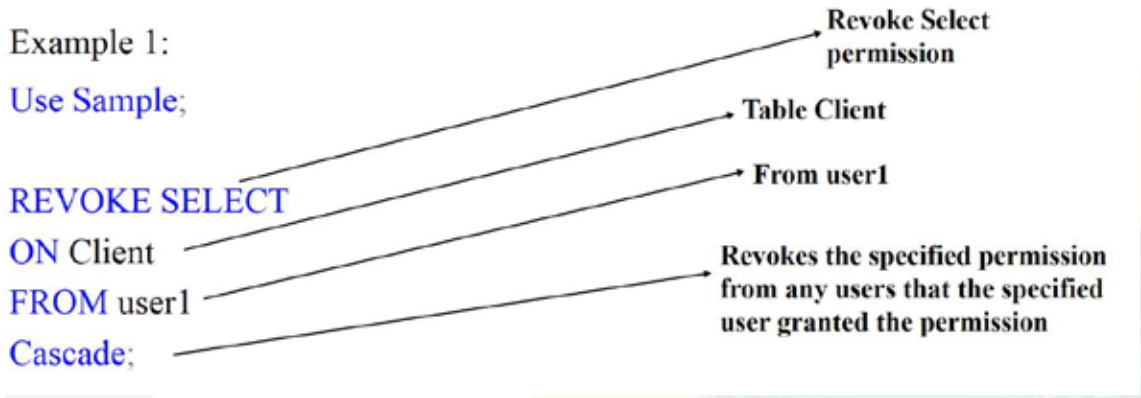
Object — Can be any database object. Typically, the object will be either a database, function, stored procedure, table or view.

User — Can be any database user.

The **GRANT OPTION FOR clause removes the specified user's ability to grant the specified permission to other users.** Note: If you include the GRANT OPTION FOR clause

in a REVOKE statement, the primary permission is not revoked. This clause revokes only the granting ability.

The **CASCADE option also revokes the specified permission from any users that the specified user granted the permission.**



Revoke Select permission from user1

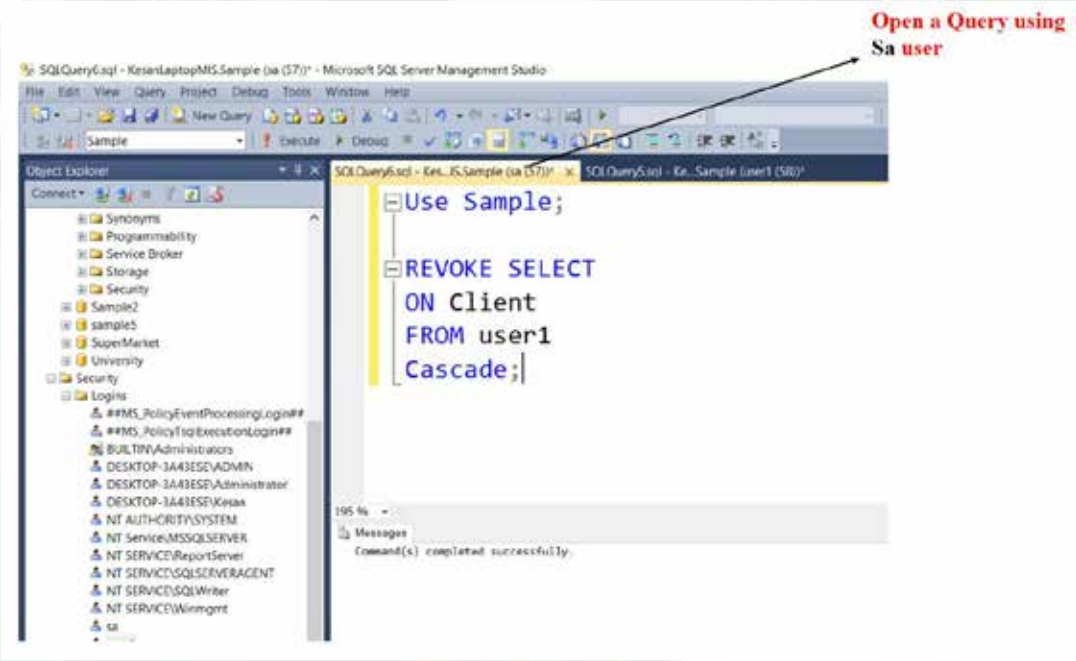


Figure 10.0.25 Revoke Select permission from user1

Run the Select command using user1

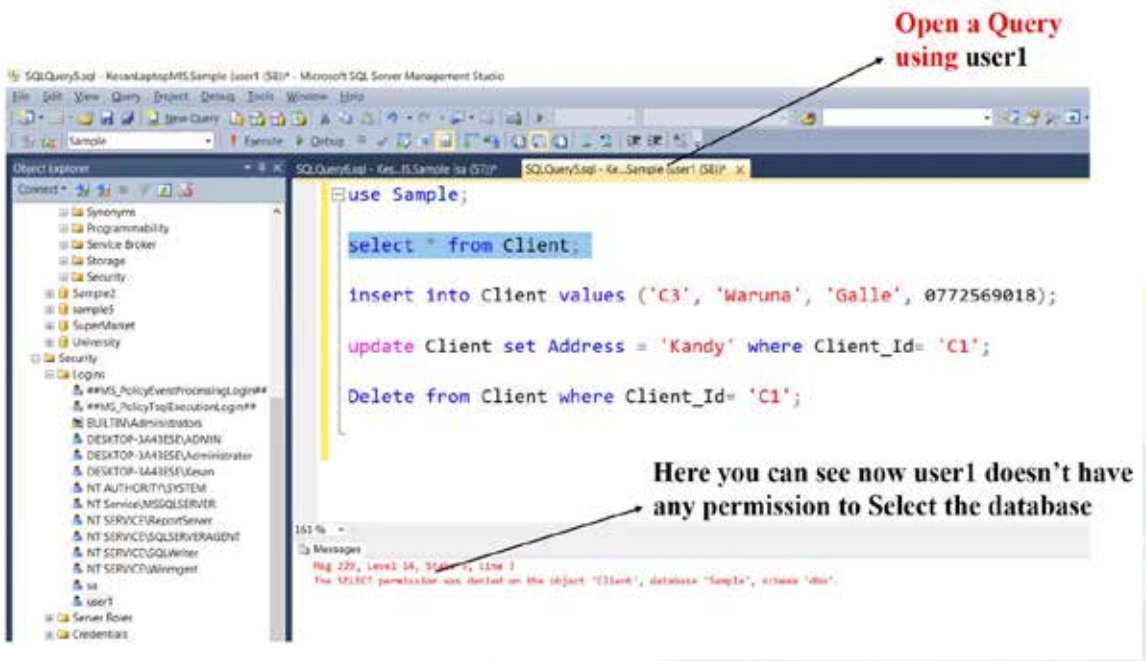
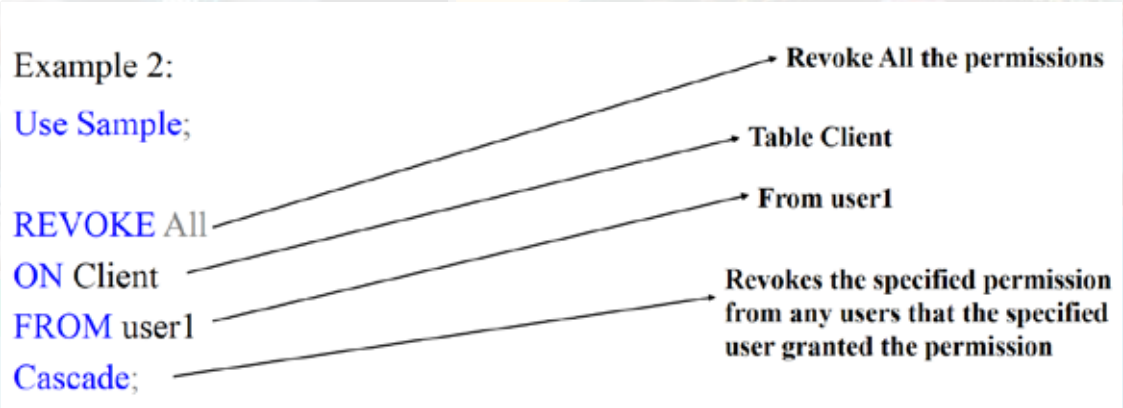


Figure 10.0.26 Run the Select command using user1



Database Schema

What is Schema: A Schema is distinct namespace to facilitate the separation, management, and ownership of database objects. You can think of schema as a namespace or container that holds tables, views, functions etc. Their names can duplicate across different schemas.

Think Database as your House. Inside your house you have Rooms. Rooms are your containers. Inside those containers (Rooms) you have a bed, chair etc. Also in your house there is a default Room (May be the Master Bed Room).

House ← This is your Database

Room 1: Bed, Chair ← This is your Schema (Container)

Room 2: Bed, Chair

Default Room: Bed, Chair ← This is your Default Schema. dbo is the Default Schema name

Whenever we say bring a chair. We always get the chair from the default room. **Like that in SQL Server the default schema is dbo.** If you create a table inside a database, the table will create inside dbo default schema.

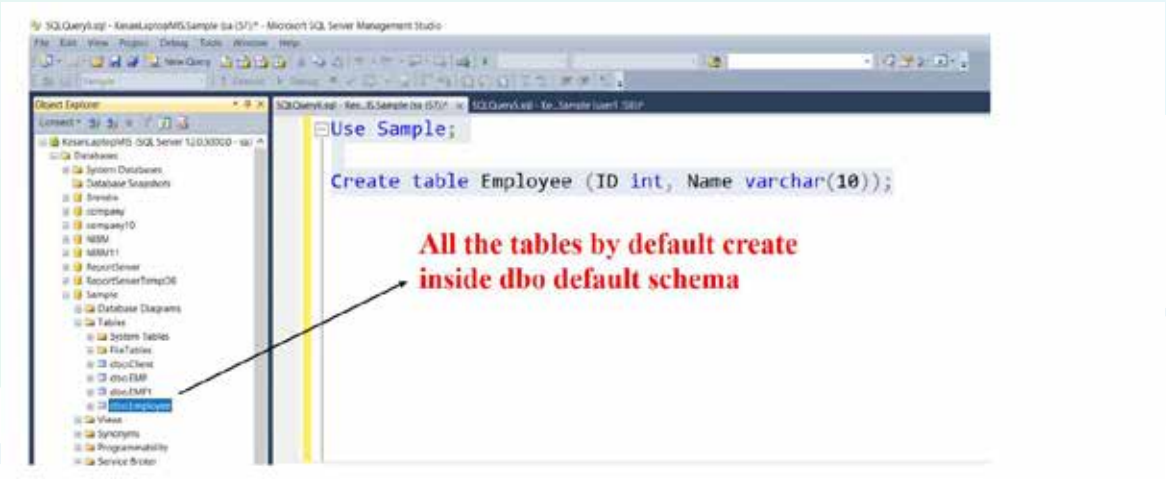


Figure 10.0.27 Default Schema

Expand the Databases - > Sample Database, then Security, then Schemas.

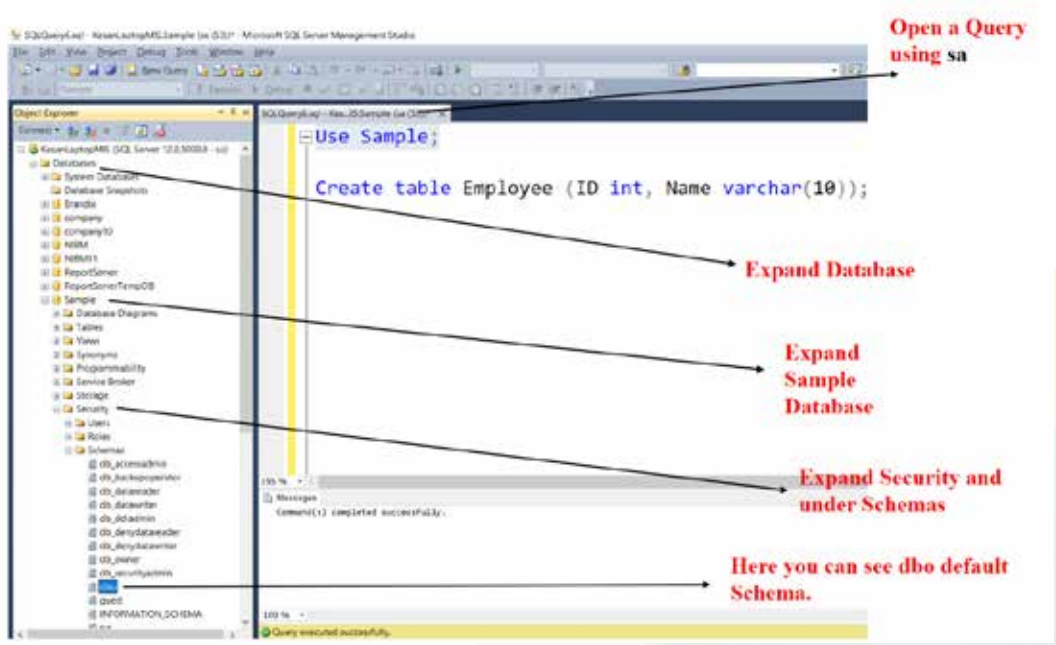


Figure 10.0.28 Default Schema dbo

Create a new database called TESTDB and **create a new schema called TB.**

```
Create database TESTDB;

use TESTDB;

Create Schema TB;
```

Expand the Databases, TESTDB Database, then Security, then Schemas.

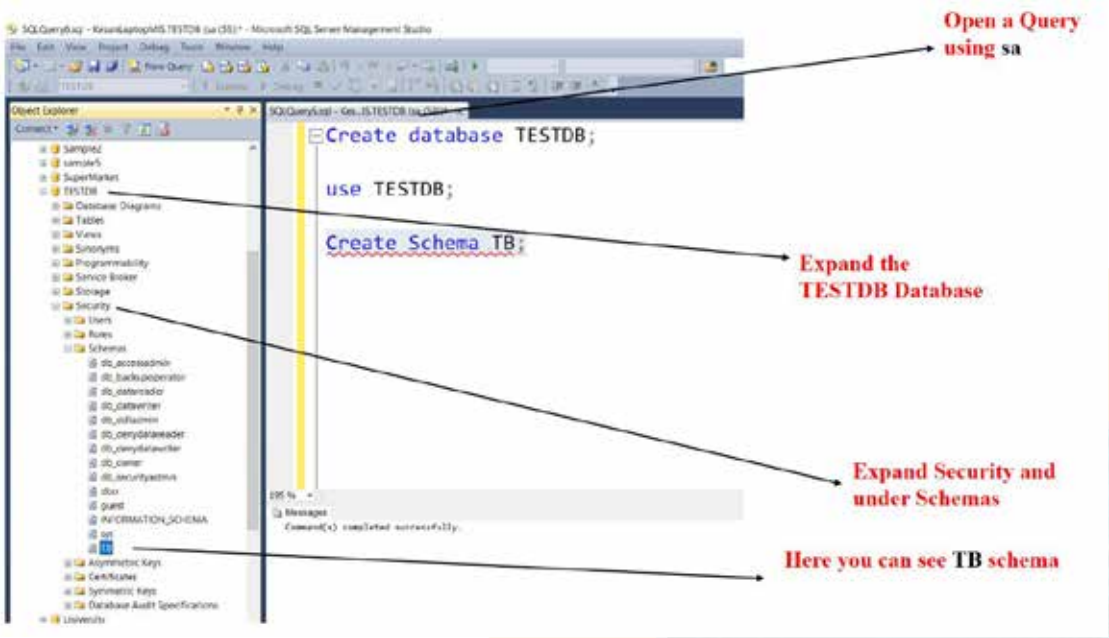


Figure 10.0.29 Create Schema TB

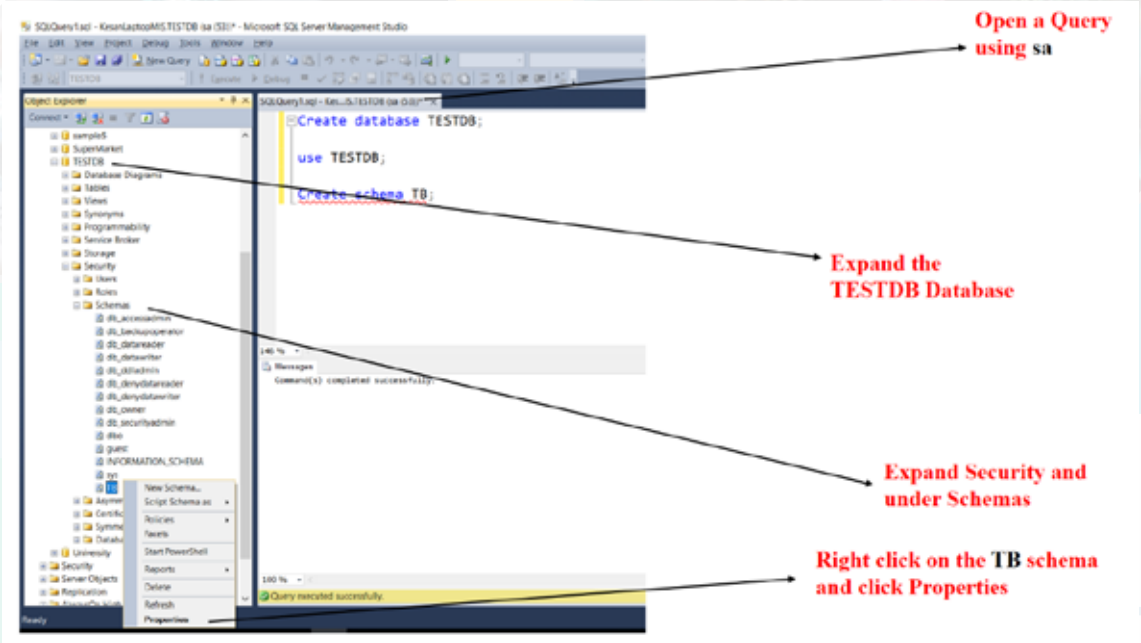


Figure 10.0.30 TB Schema Properties

Once you click Properties you can see the Schema Owner is dbo user.

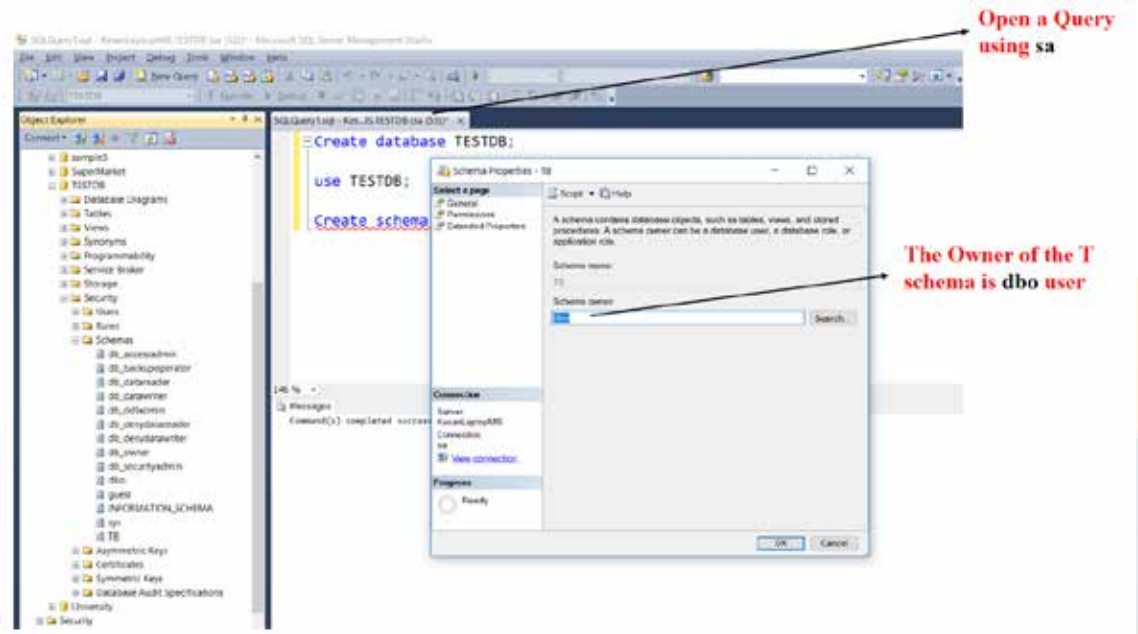


Figure 10.0.31 Schema Owner dbo

Now we are going to add the user1 (the user which we created earlier) to TESTDB then we can add that user1 to TB Schema.

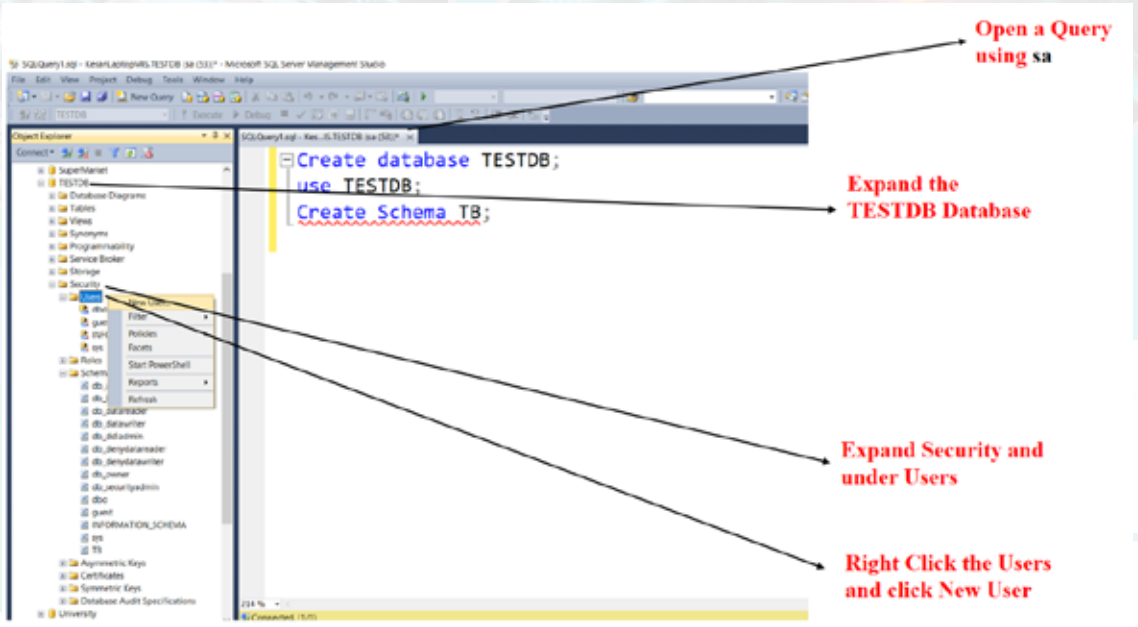


Figure 10.0.32 Add user to Database

Now once you click New User give user name as user1 and login name as user1. Do not select a default schema.

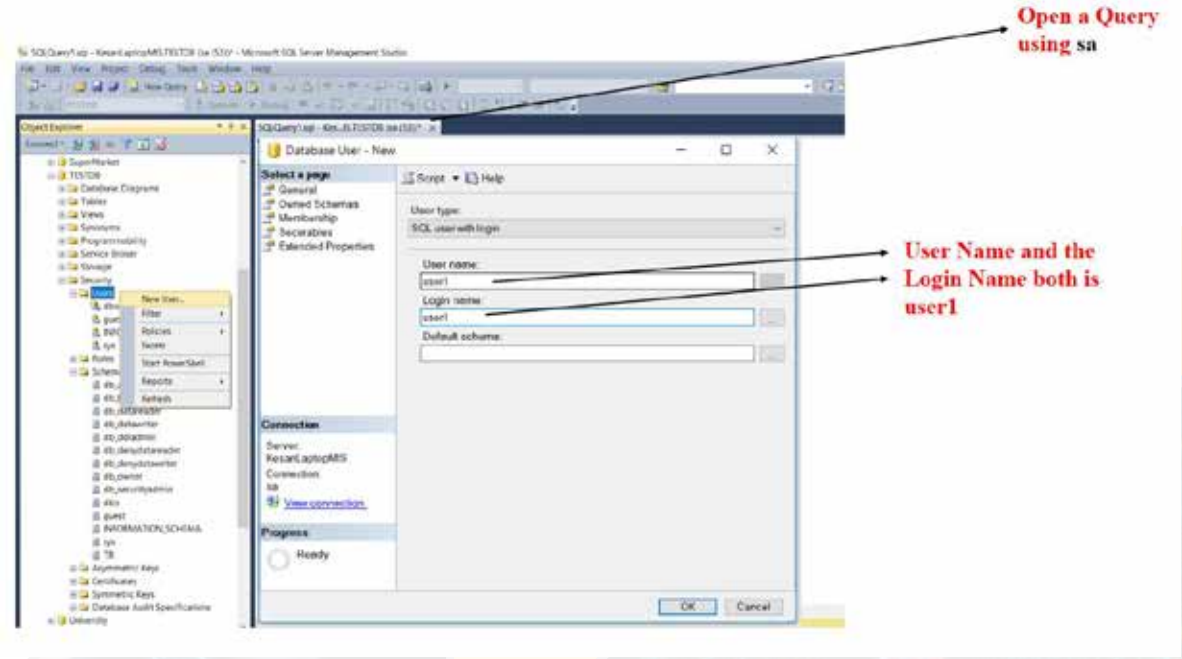


Figure 10.0.33 Add user1 as a new user

Go to Schema select TB schema right click and go to properties.

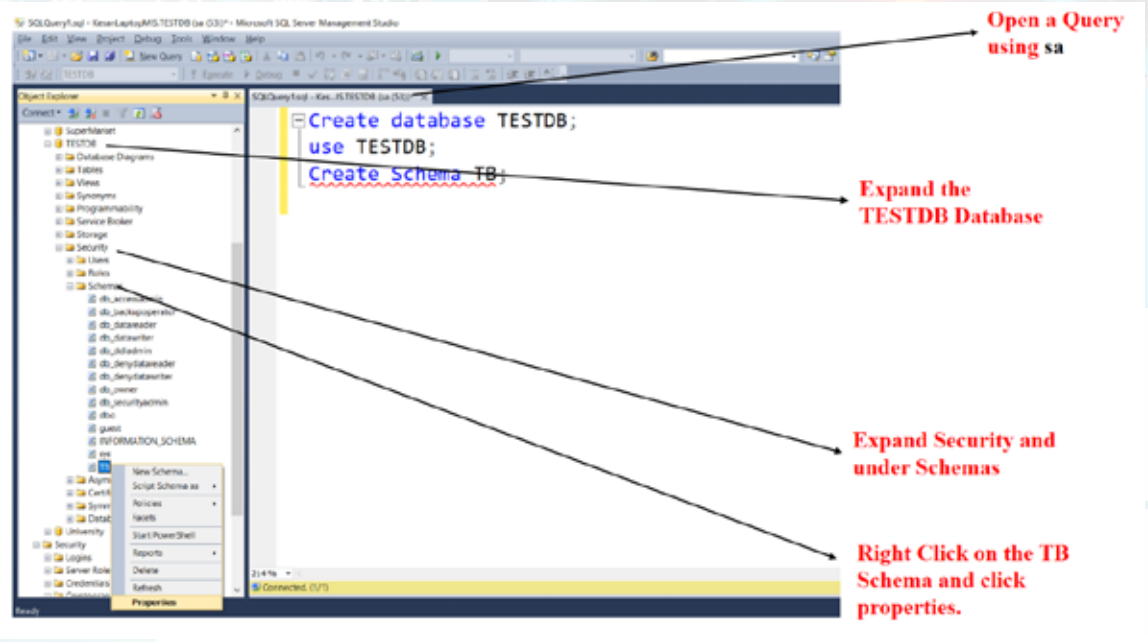


Figure 10.0.34 Select TB schema Properties

Search and Browse user1, select user1 and click Ok.

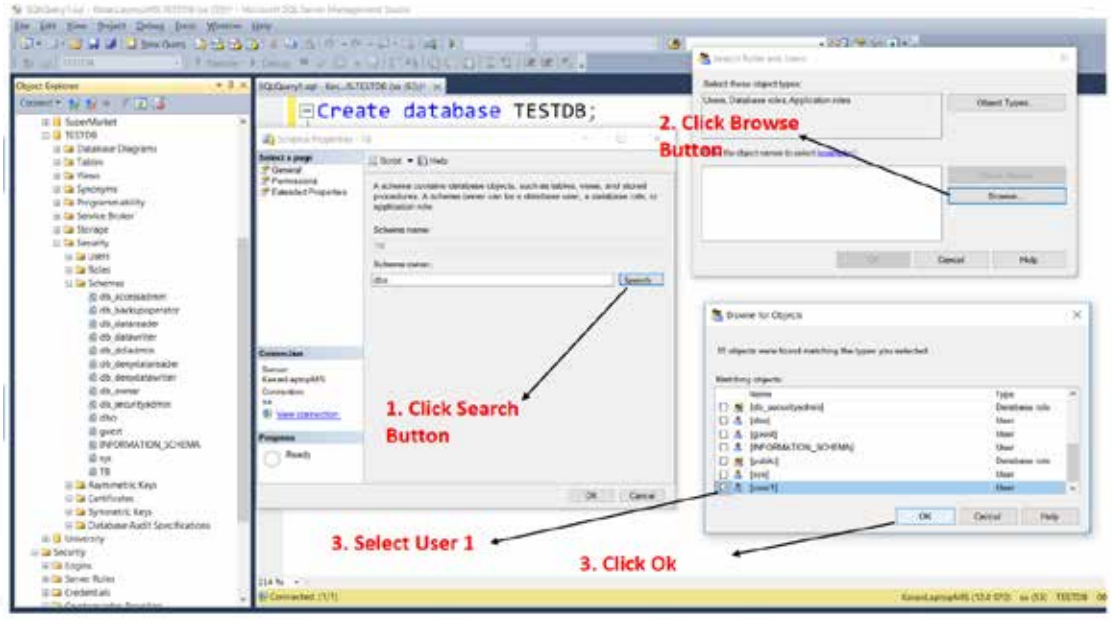


Figure 10.0.35 Select user1

Now TB Schema is owning by user1. Right click on TB schema and go to properties.

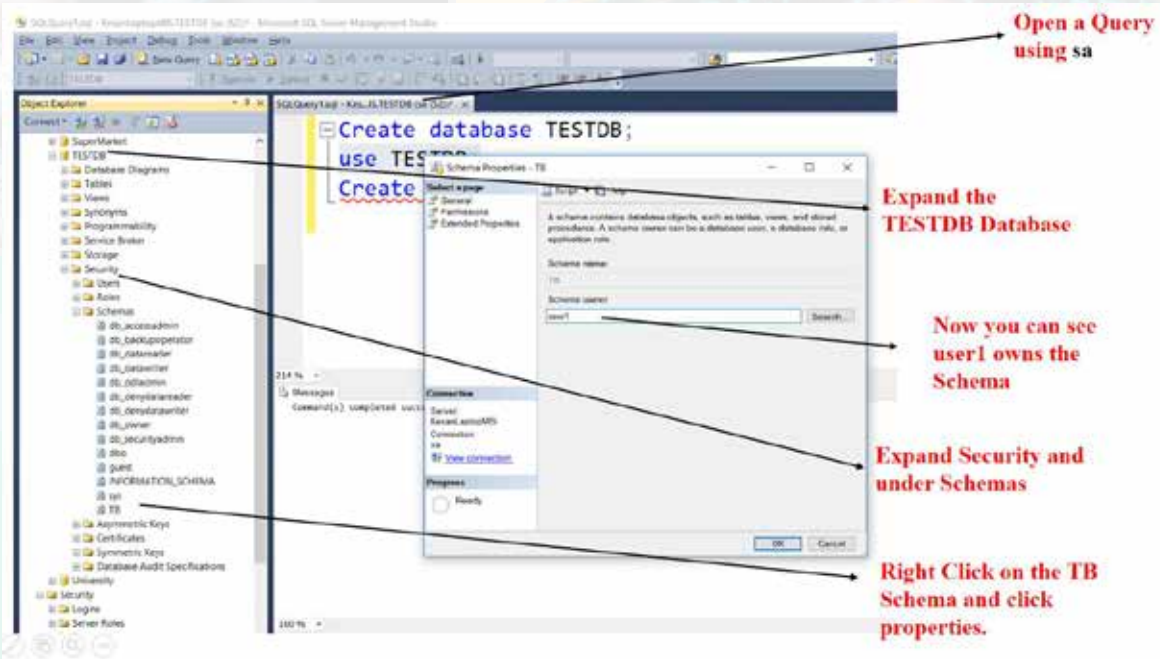


Figure 10.0.36 User1 owns TB Schema

Create table in TB schema under the TESTDB database using user1

Click Connect, then click Database Engine, Login as user1.

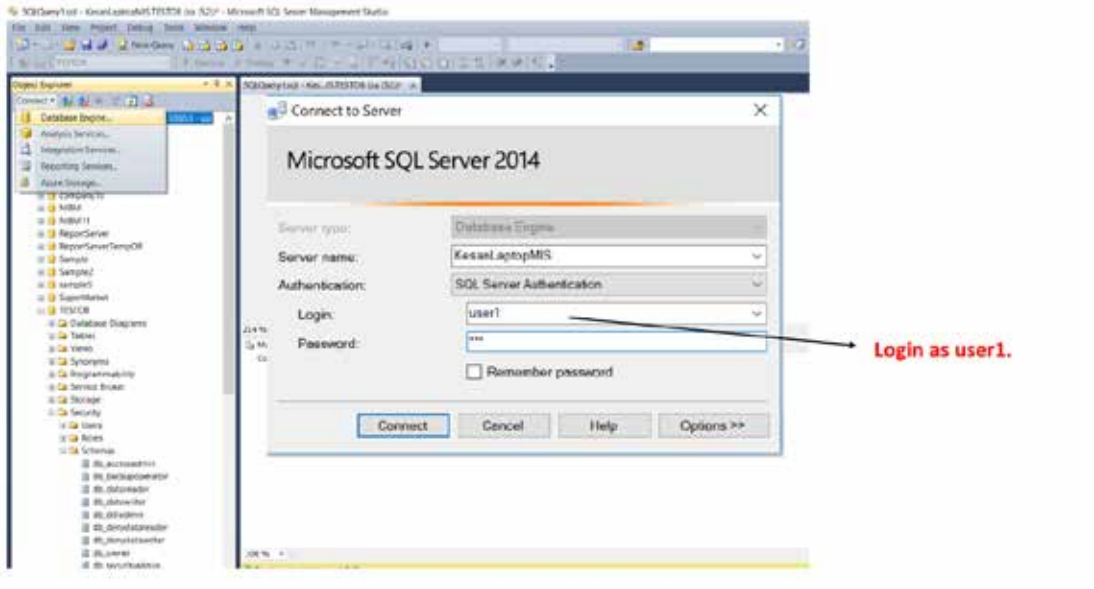


Figure 10.0.37 Login as user1

Open a new query using user1. Then use TESTDB and try to create a table. Now when you create the table you can specify the **Schema TB**.

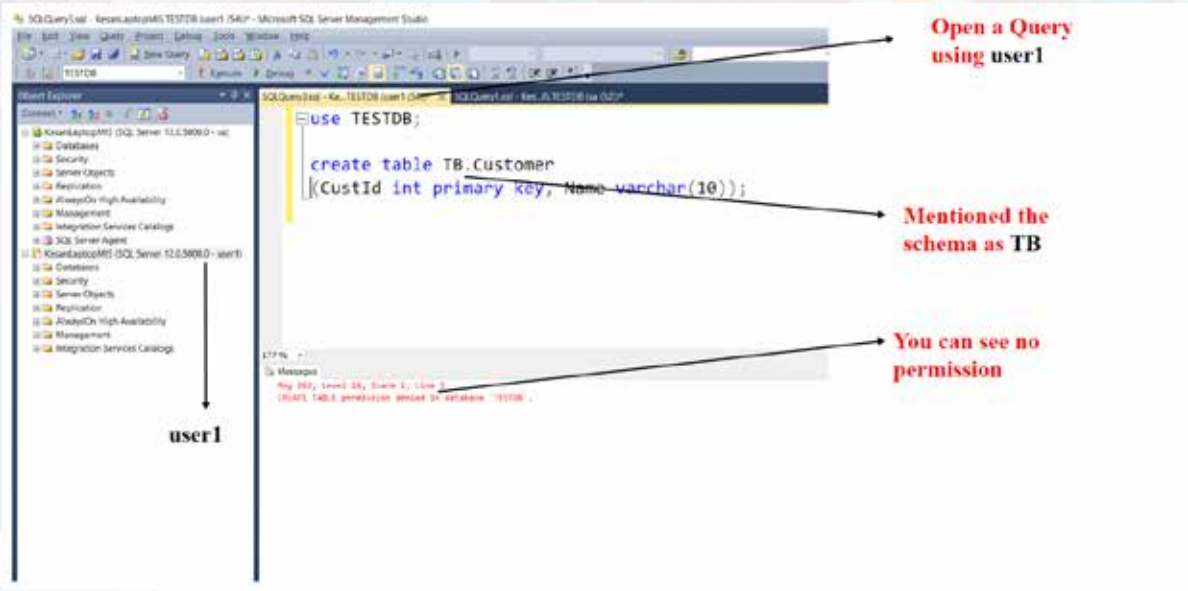


Figure 10.0.38 Create a Database using user1

Therefore, we have to give permission to the user1 from the sa user.

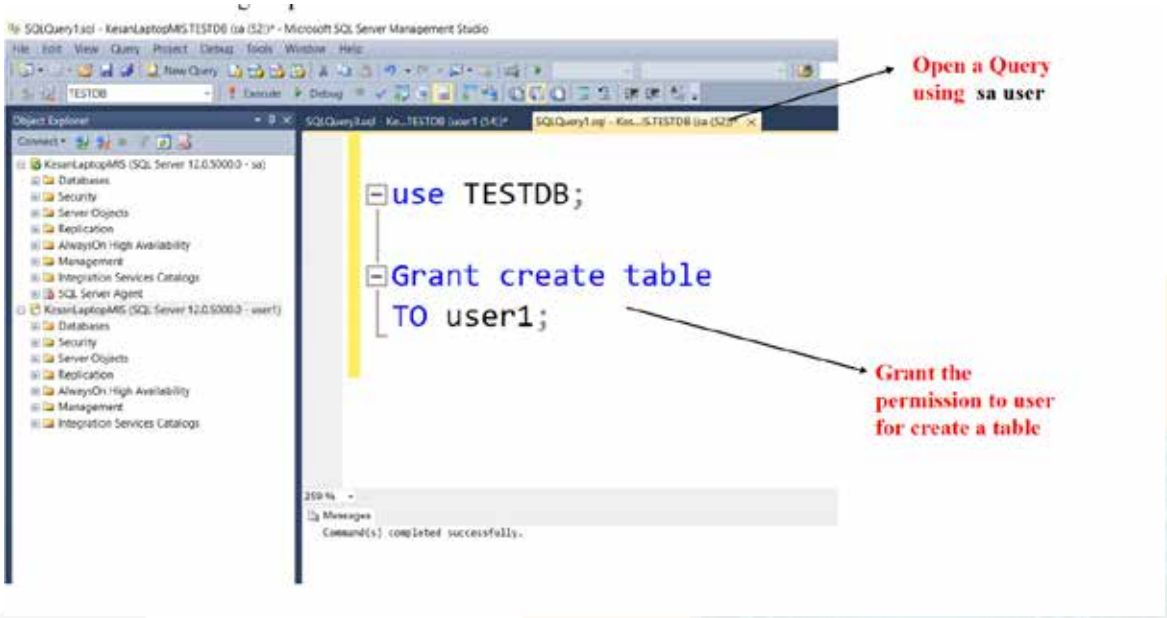


Figure 10.0.39 Give permission to the user1 from the Sa user

Now go to the user1 query and create a table. Now you will be able to create a table.

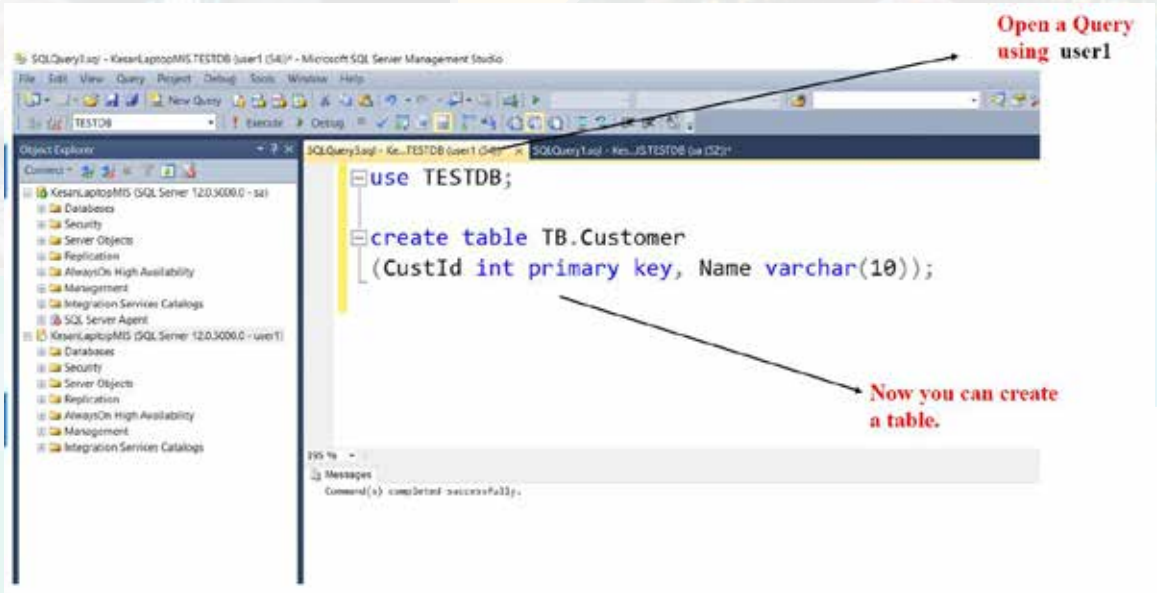
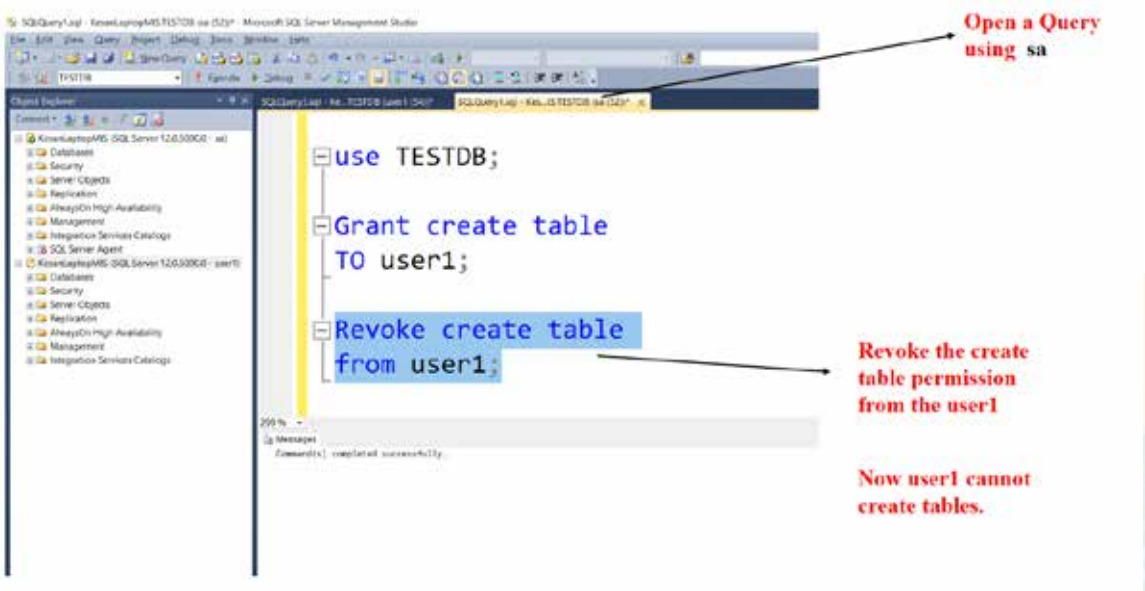


Figure 10.0.40 Create a Table using user1

Revoke create table permission.



Deny Permission

- Explicitly Denying Database Access

- The DENY command is used to explicitly prevent a user from receiving a particular permission. This is helpful when a user is a member of a role or group that is granted a permission, and you want to prevent that individual user from inheriting the permission by creating an exception.

DENY [permission]
ON [object]
TO [user]

Create a Role and add user1 to that Role

Rather than creating a user you can create a Role in SQL server. Once you create a Role you can add different users to that role.

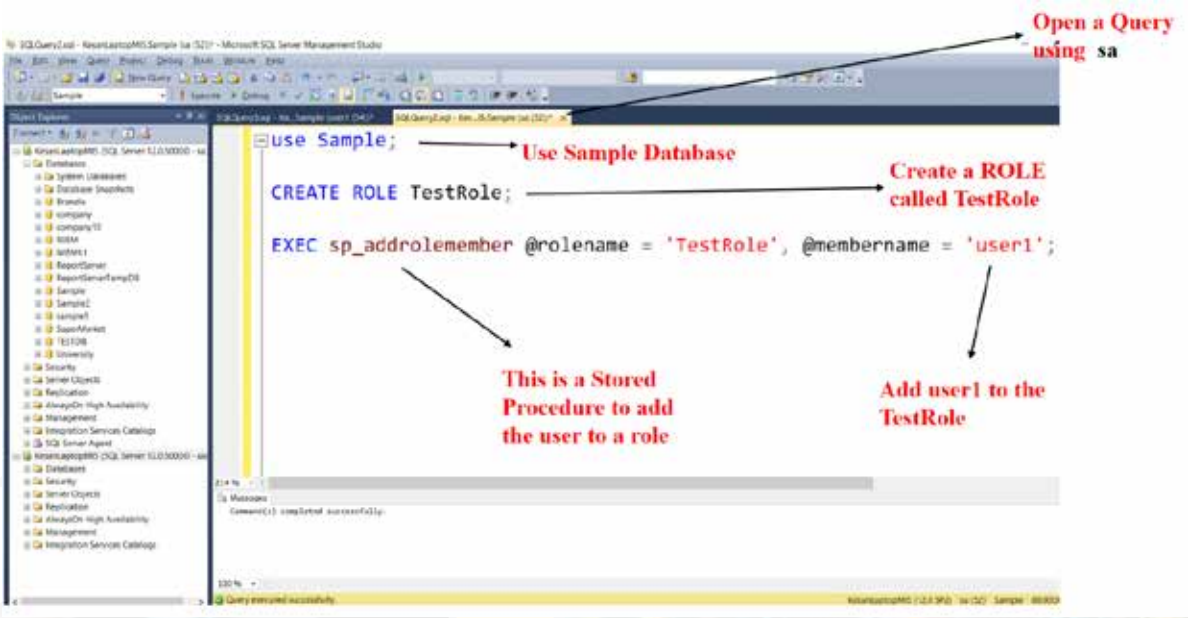


Figure 10.0.41 Create a Role

```
use Sample;

CREATE ROLE TestRole;

EXEC sp_addrolemember @rolename = 'TestRole', @membername = 'user1';
```

After you create a Role and add the user1 to that Role. Create a new Schema called Test and create a simple table under that Test Schema. Then Grant the permission to TestRole for select Test.TestTable.

```
CREATE SCHEMA Test;

CREATE TABLE Test.TestTable (ID int, Name varchar(10));

GRANT SELECT ON Test.TestTable TO TestRole;
```

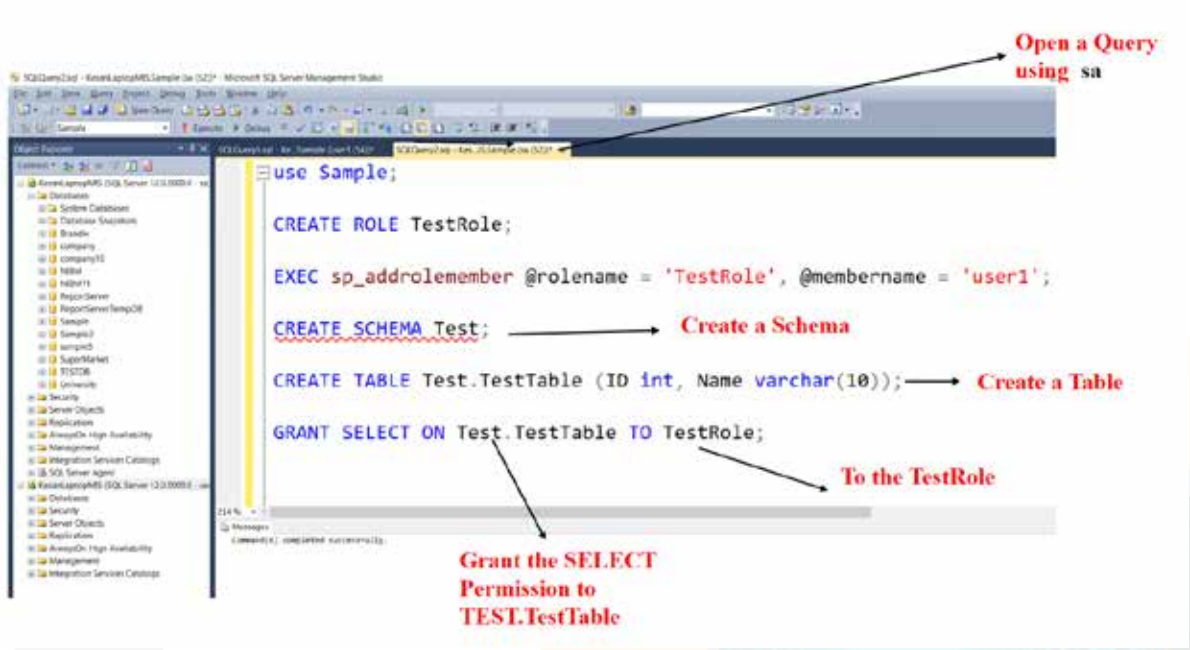


Figure 10.0.42 Grant Permission to Role

Check whether the user1 can select the table Test.TestTable

Open a new query using user1 and run the select command on Test.TestTable to check the permission. **Here you can see user1 can select the table because TestRole has the permission and user1 is a user of TestRole.**

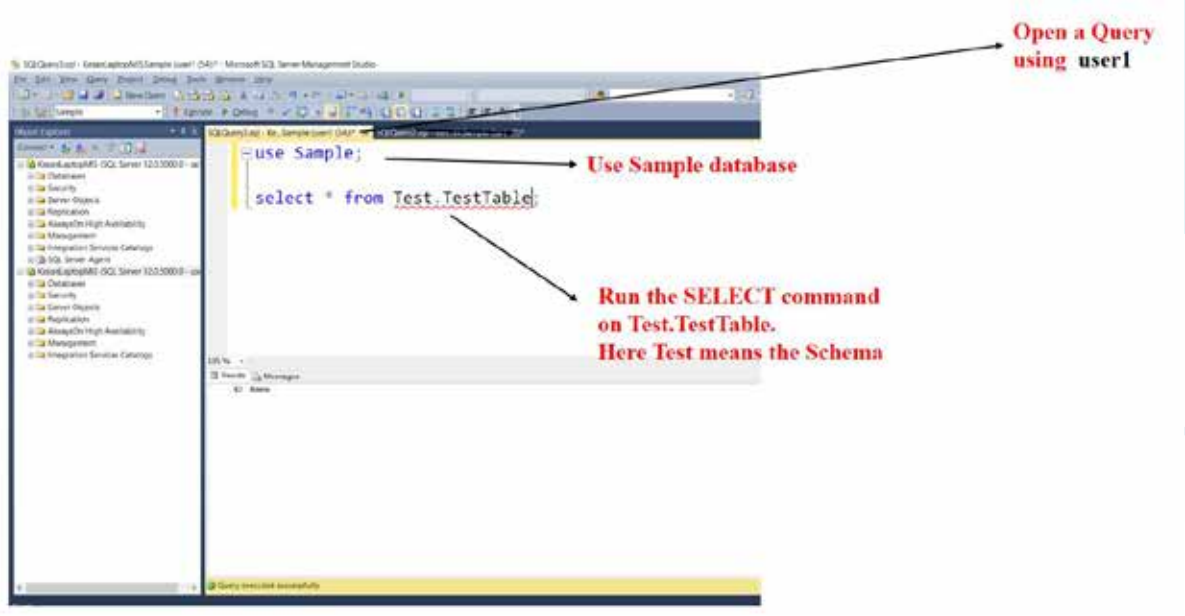


Figure 10.0.43 Select Table using user1

Deny Select permission from the user1

Now we are going to deny the permission from the user1. Here TestRole has the permission to Select the tables and since user1 is a user on that Role user1 can also select the table. Now if we deny the select permission from the user1. User1 cannot select the table even TestRole has select permission.

```
use Sample;

DENY SELECT ON Test.TestTable TO user1;
```

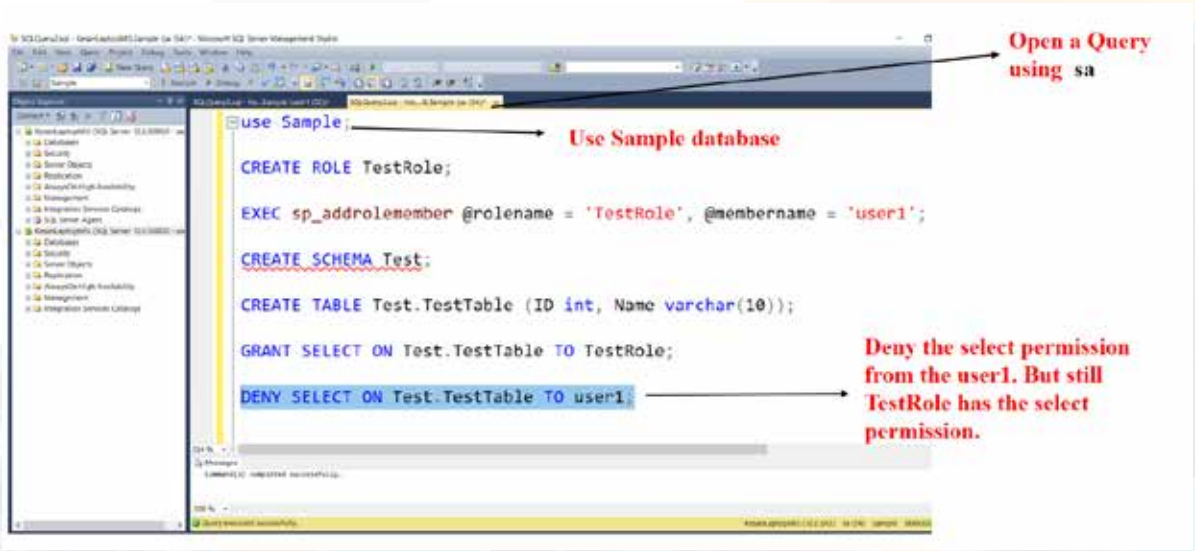


Figure 10.0.44 Deny Select permission from the user1

Now once you run the select command inside user1. You can see user1 doesn't have permission to select the table.

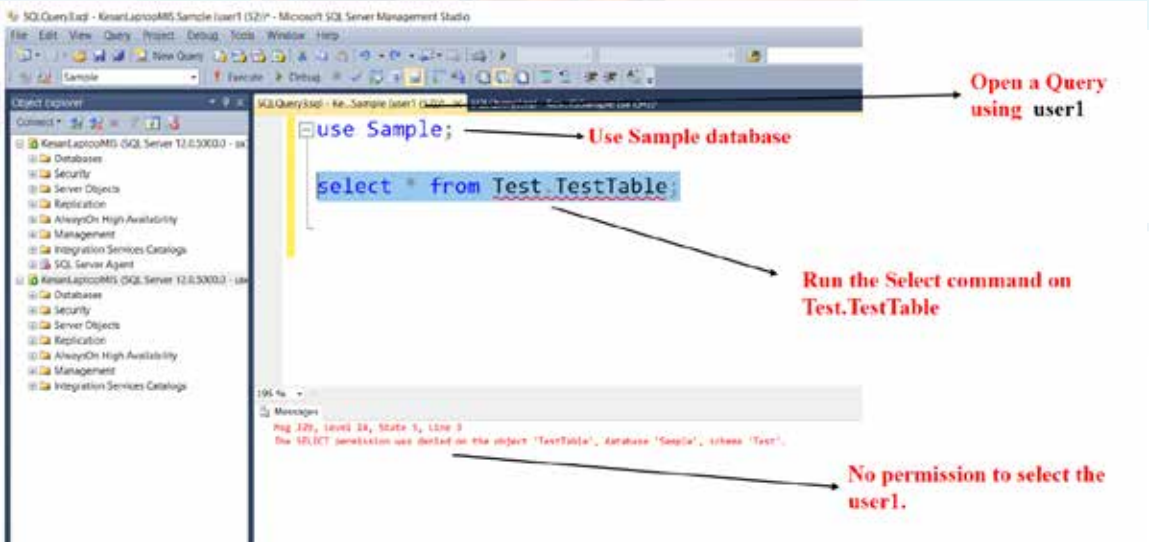


Figure 10.0.45 user1 doesn't have permission to select the table