Azure SQL Server Dynamic Data Masking

In this tutorial I'm going to show how to:

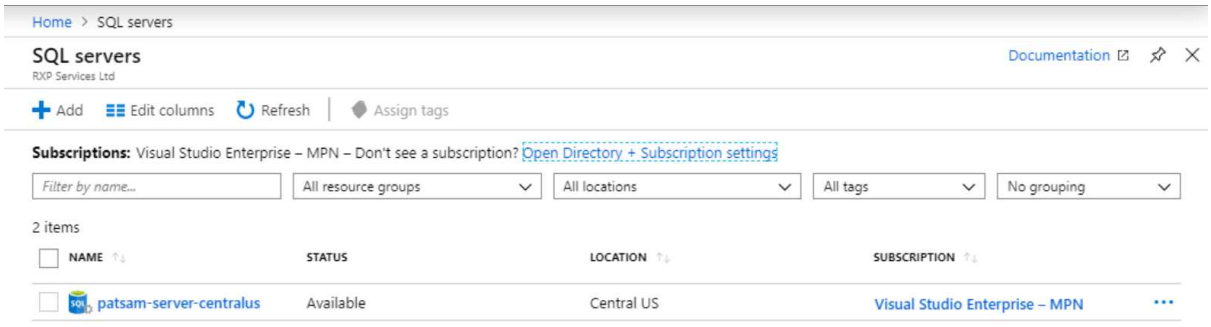
- Setting up Dynamic Data Masking
- Create different SQL Users to test out Dynamic Data Masking
- Test Dynamic Data Masking using different logins

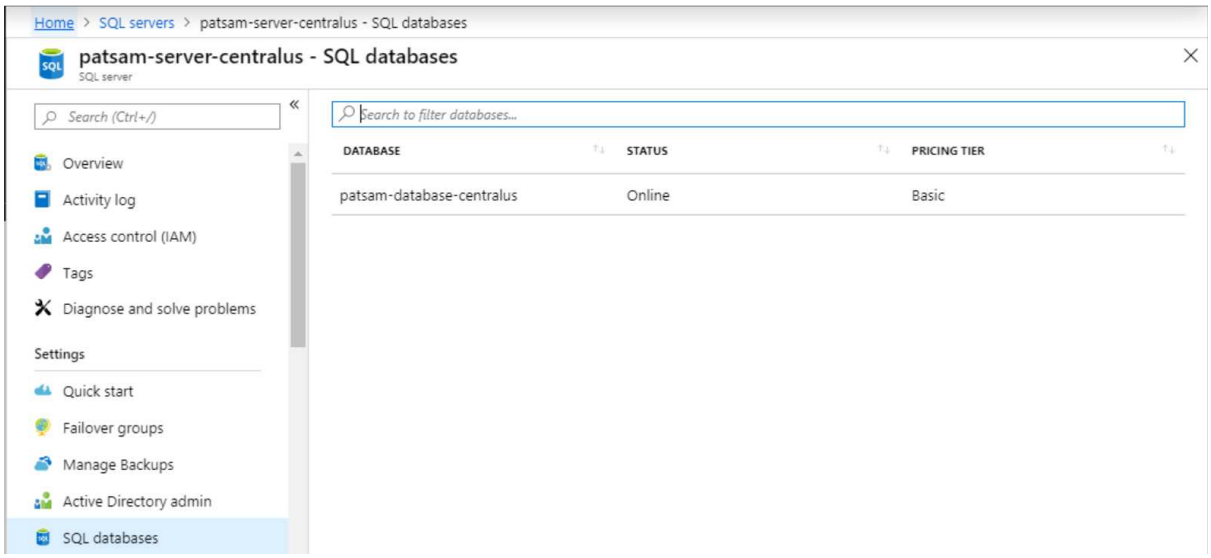
Patrick Sameera Jayalath

Step 01: Setting up Dynamic Data Masking

For this exercise we are using an existing SQL Database.

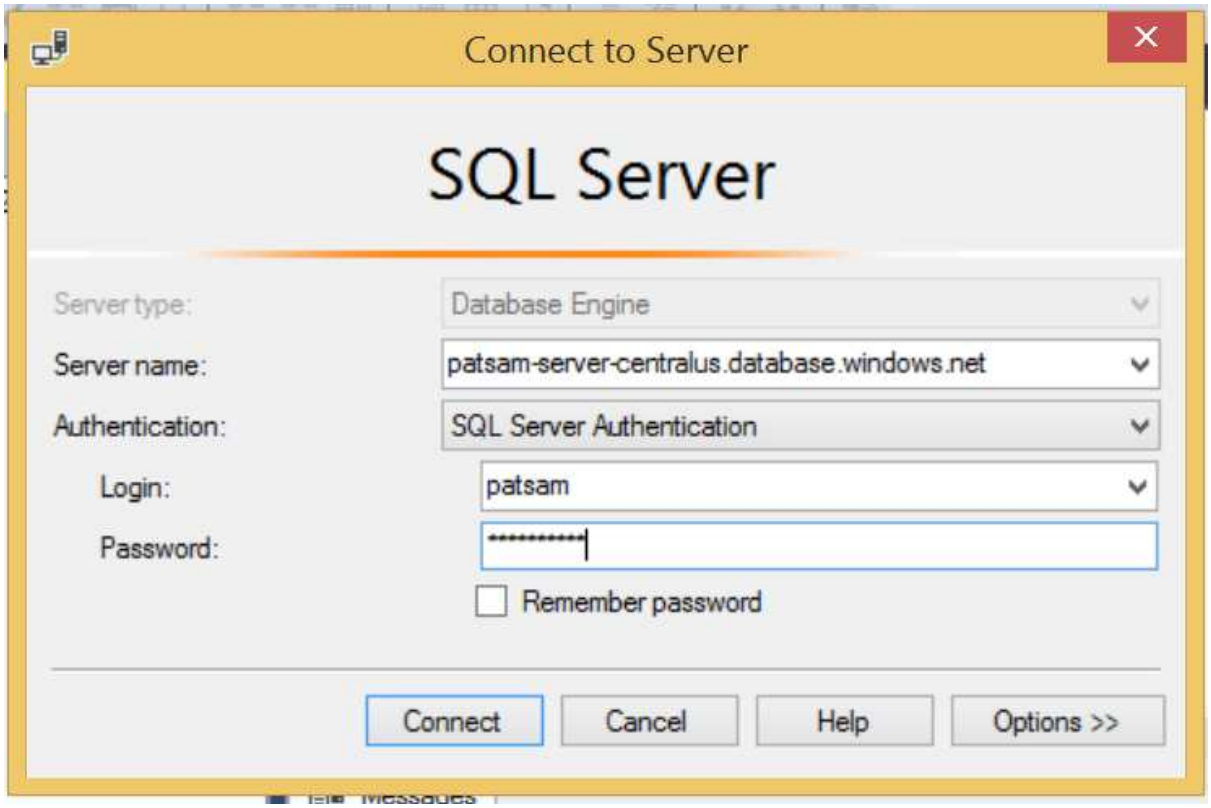
- Resource group: rg-NetworkServices
- Azure SQL Server: patsam-server-centralus
- Azure SQL Database: patsam-database-centralus





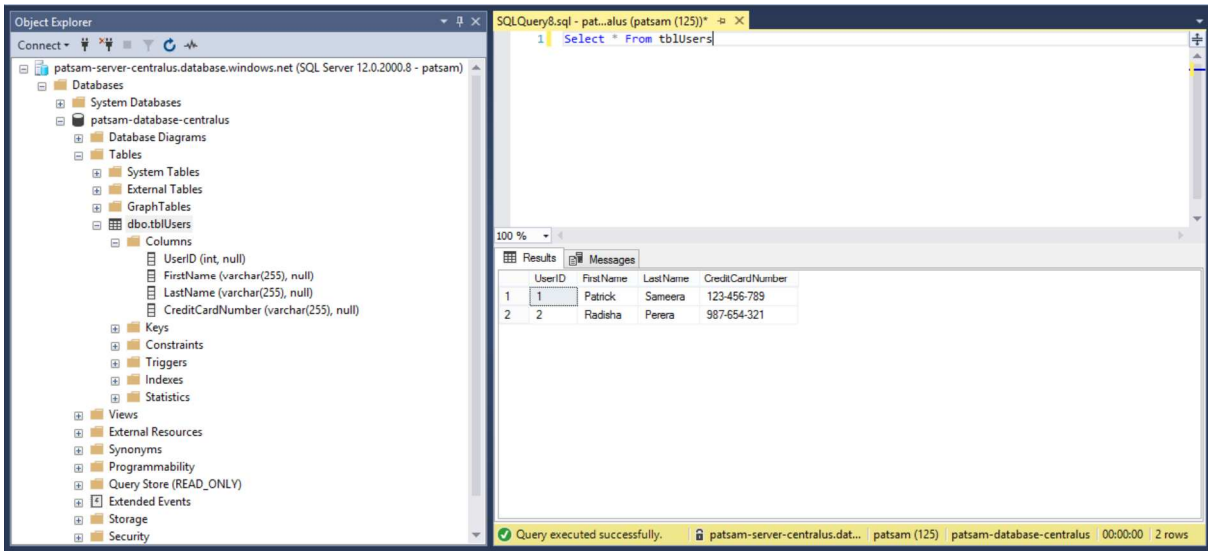
Connect to SQL Database through SQL Server Management to inspect the table structure and data.

Make sure you have relevant Firewall Rules defined that allows us to connect to SQL Server Management.



In the Database we have a table called “tblUsers” with bellow table structure.

- UserId
- FirstName
- LastName
- CreditCardNumber



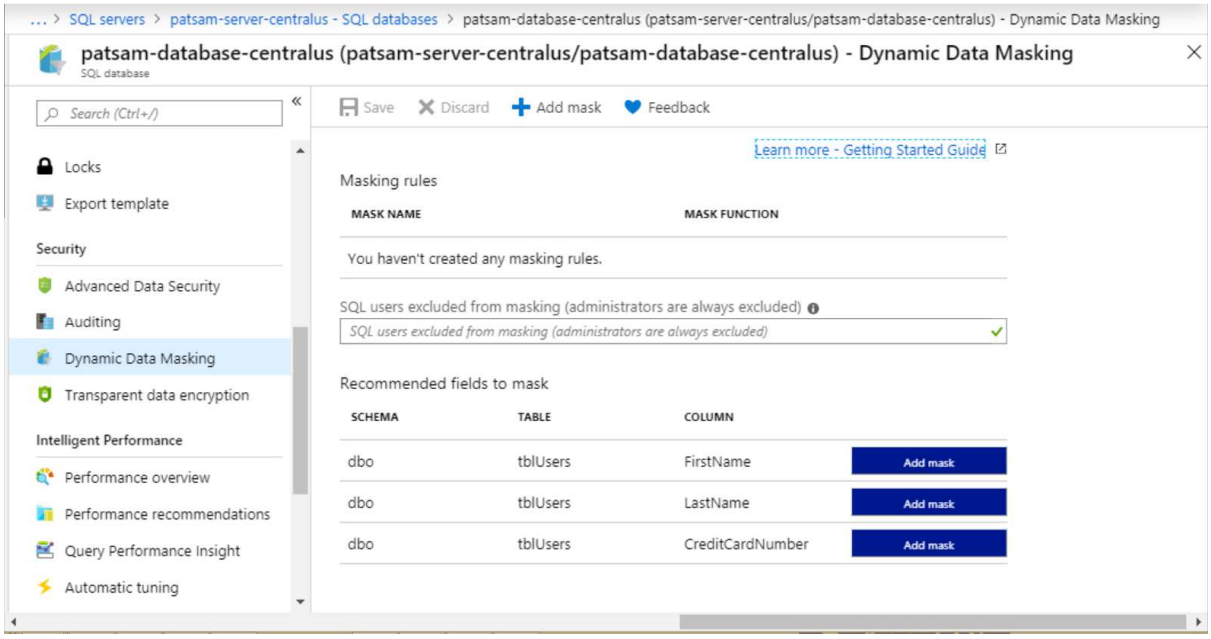
As you can see tblUsers have columns like Credit Card Number details, which are very sensitive information.

When Dynamic Data Masking is applied, the main objective is to limit the exposure of confidential information to users who do not have the necessary privileges to observe confidential information.

Data is not masked and stored in the disk. Masking is applied to the result set returned to the user. The users with “db_owner” permission on “UNMASK” permission can still see the unmasked data when they query the table.

Click on SQL Database → Dynamic Data Masking tab.

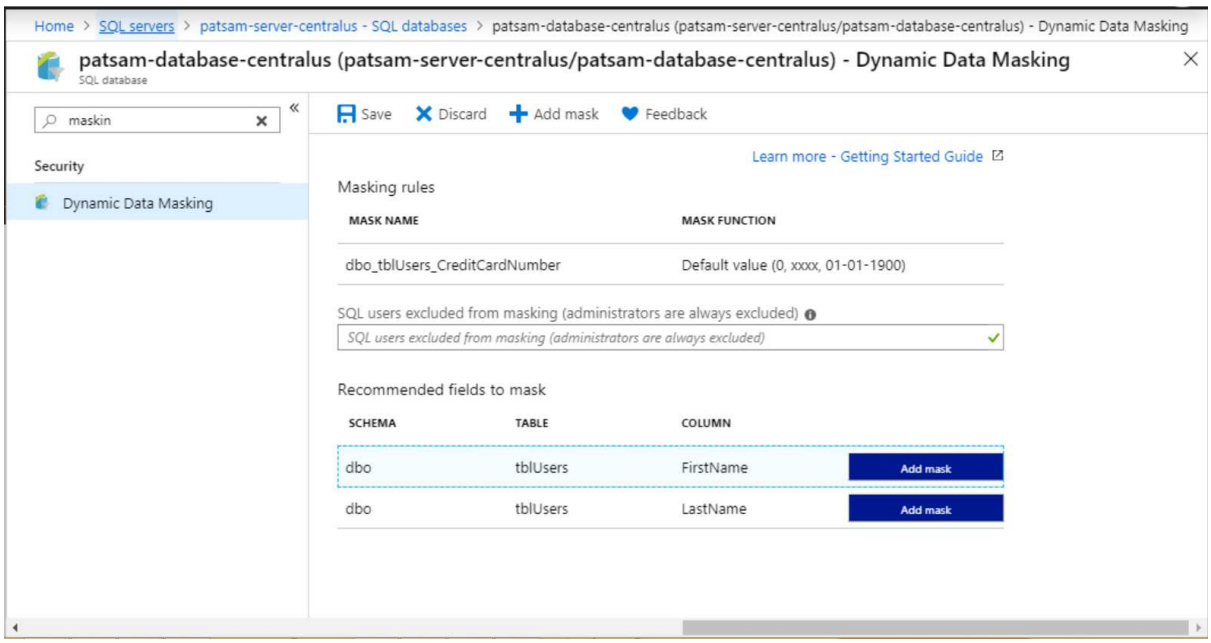
It will show the list of tables and columns in the Database that is recommended for masking.



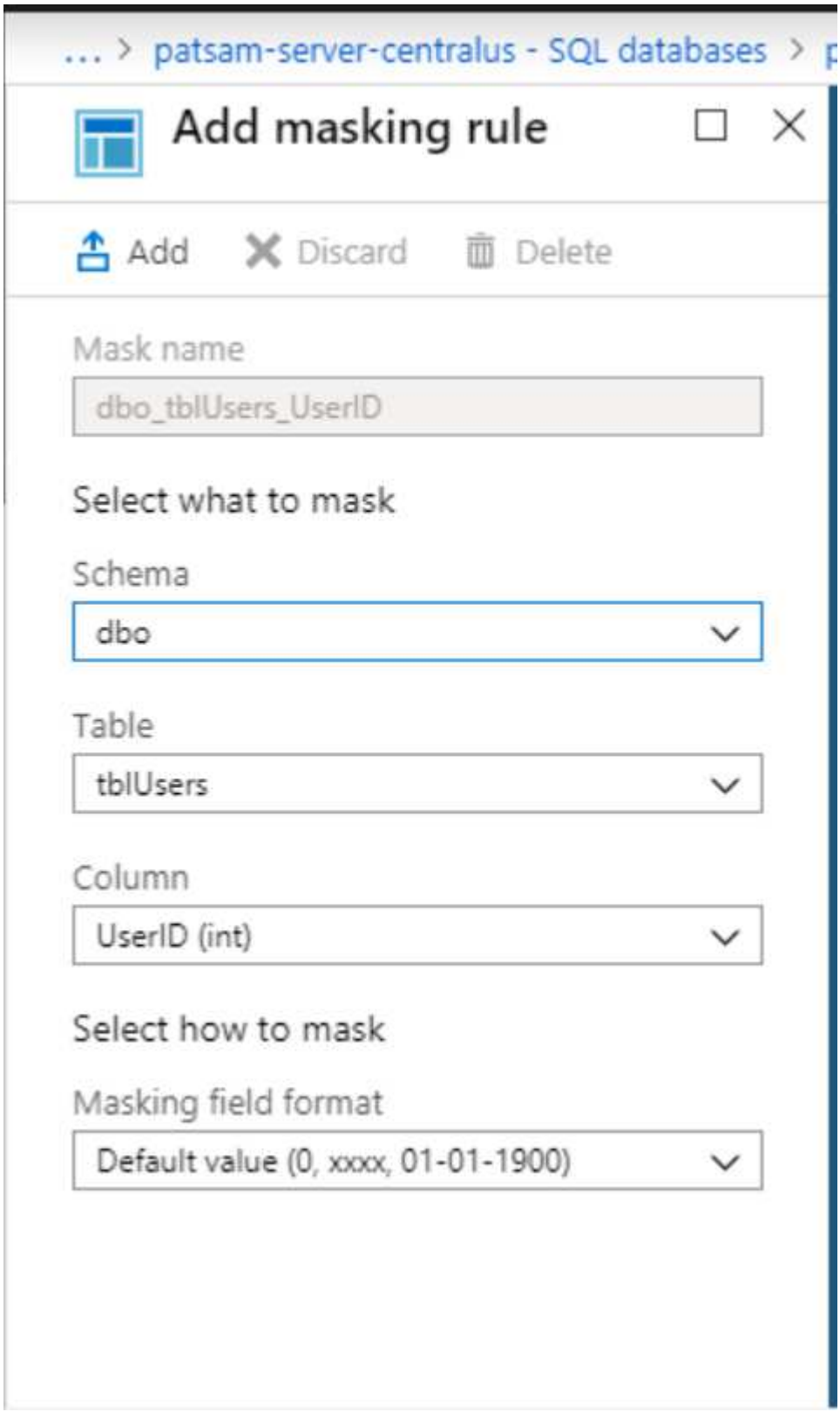
Since we want to do masking for CreditCardNumber column Click on Add Mask button against CreditCardNumber.

It will add new Masking Rule with:

- Masking name: dbo_tblUsers_CreditCardNumber
- Mask function: Default value (0, xxxx, 01-01-1900)



Or you can add new mask from clicking + Add Mask to mask columns that are not shown under recommendations.

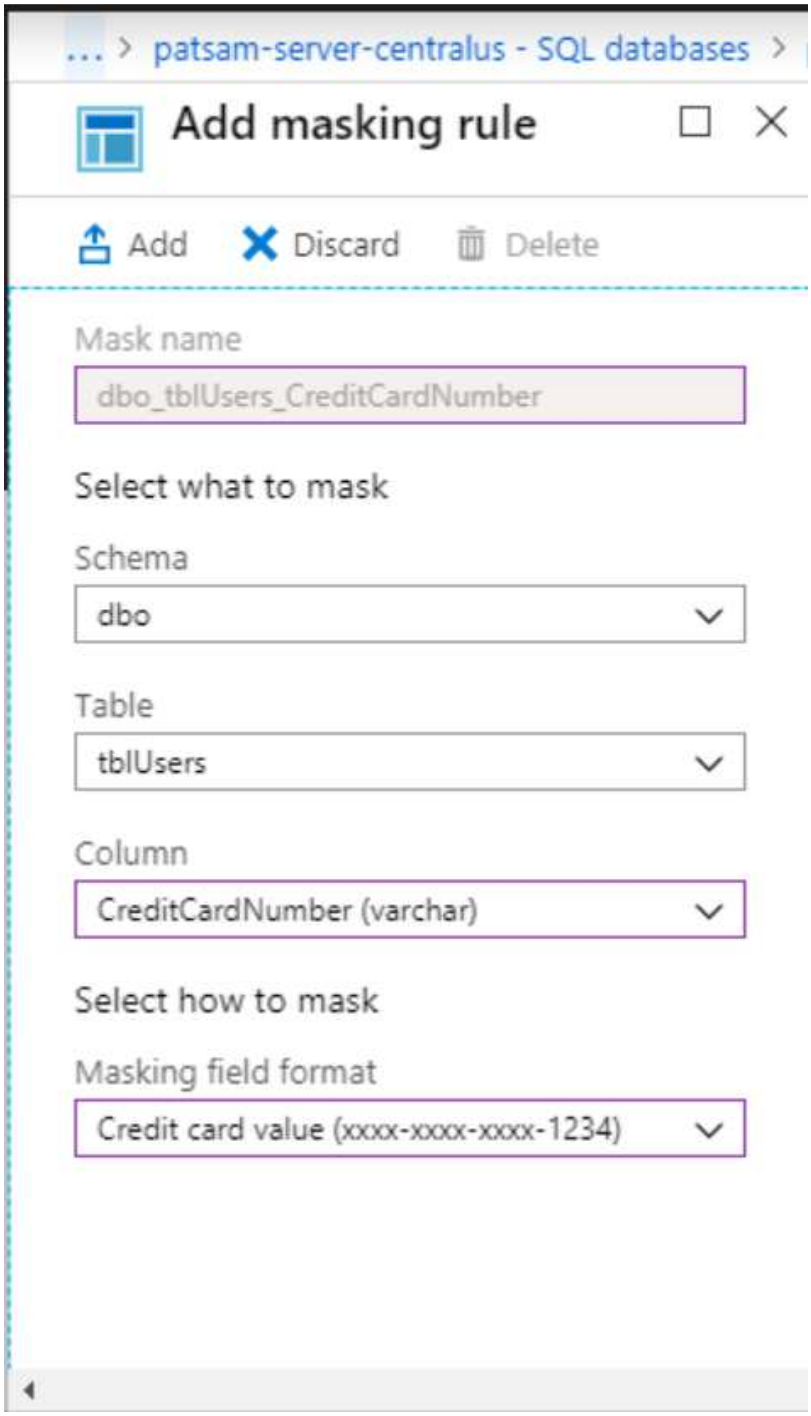


There we can select any Schema/Table/Column we want to Mask.

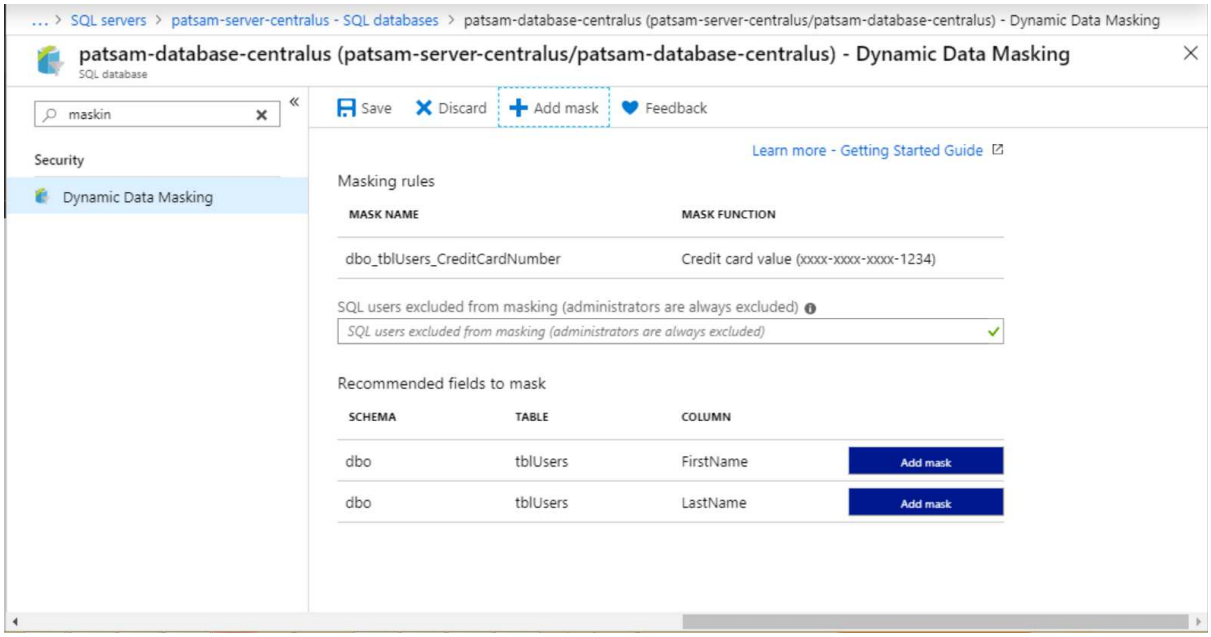
Since we are masking CreditCardNumber column:

- Schema: dbo
- Table: tblUsers
- Column: CreditCardNumber
- Masking format: Credit card value (xxxx-xxxx-xxxx-1234)

Click Add.



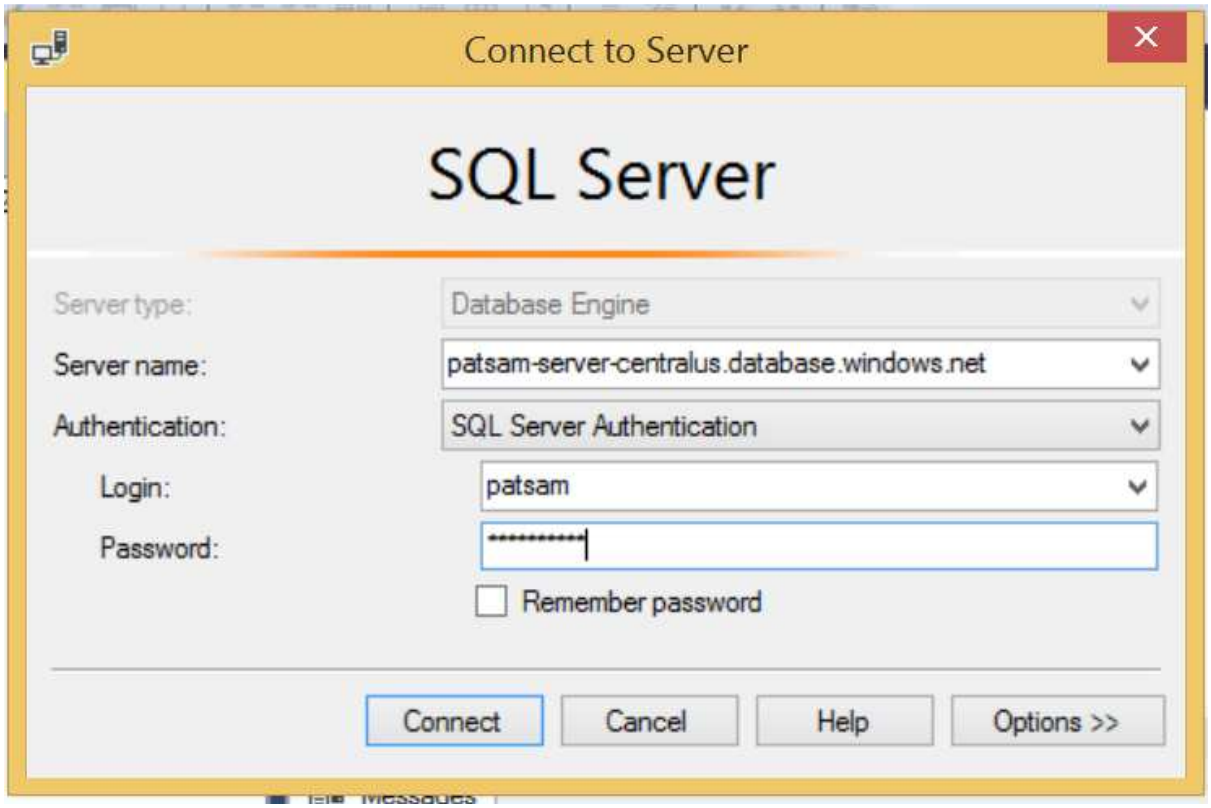
Click Save


... > SQL servers > patsam-server-centralus - SQL databases > patsam-database-centralus (patsam-server-centralus/patsam-database-centralus) - Dynamic Data Masking
patsam-database-centralus (patsam-server-centralus/patsam-database-centralus) - Dynamic Data Masking
SQL database
maskin
Security
Dynamic Data Masking
Save
Discard
Add mask
Feedback
Learn more - Getting Started Guide
Masking rules
MASK NAME
MASK FUNCTION
dbo_tblUsers_CreditCardNumber
Credit card value (xxxx-xxxx-xxxx-1234)
SQL users excluded from masking (administrators are always excluded)
SQL users excluded from masking (administrators are always excluded)
Recommended fields to mask
SCHEMA
TABLE
COLUMN
dbo
tblUsers
FirstName
Add mask
dbo
tblUsers
LastName
Add mask

Step 02: Create different SQL Users to test out Dynamic Data Masking

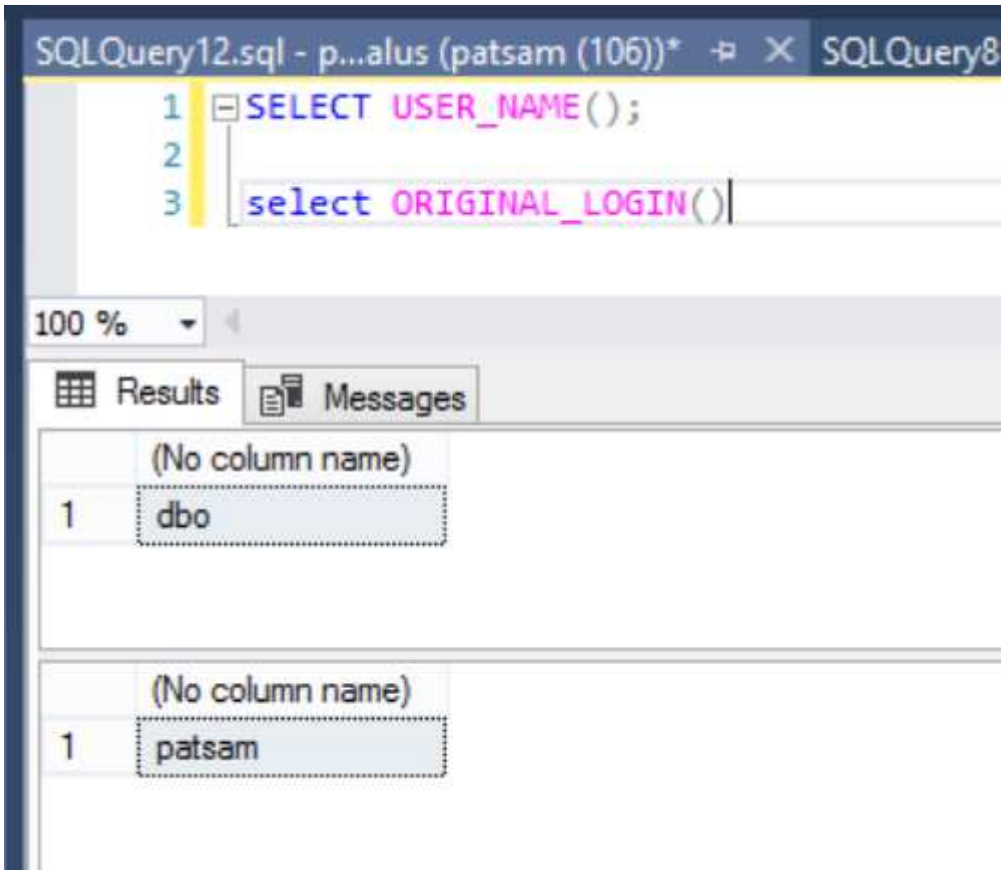
Log in to the Database using SQL Server Management.

- Login: patsam



Run bellow query to find out about login details.

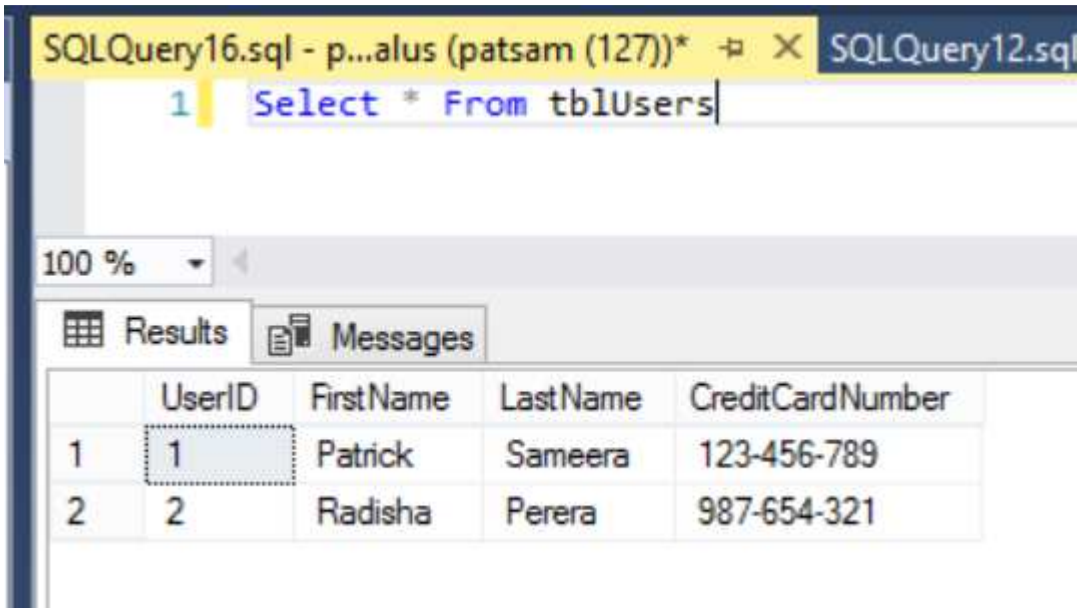
```
select USER_NAME()

select ORIGINAL_LOGIN()
```



Run bellow query to view the data on tblUsers.

```
Select * From tblUsers
```

Notice even though we masked column CreditCardNember column, we still can view the data.



That is because User logged in have the appropriate permissions to view data.

Remember I mentioned that the users with “db_owner” permission on “UNMASK” permission can still see the unmasked data when they query the table.

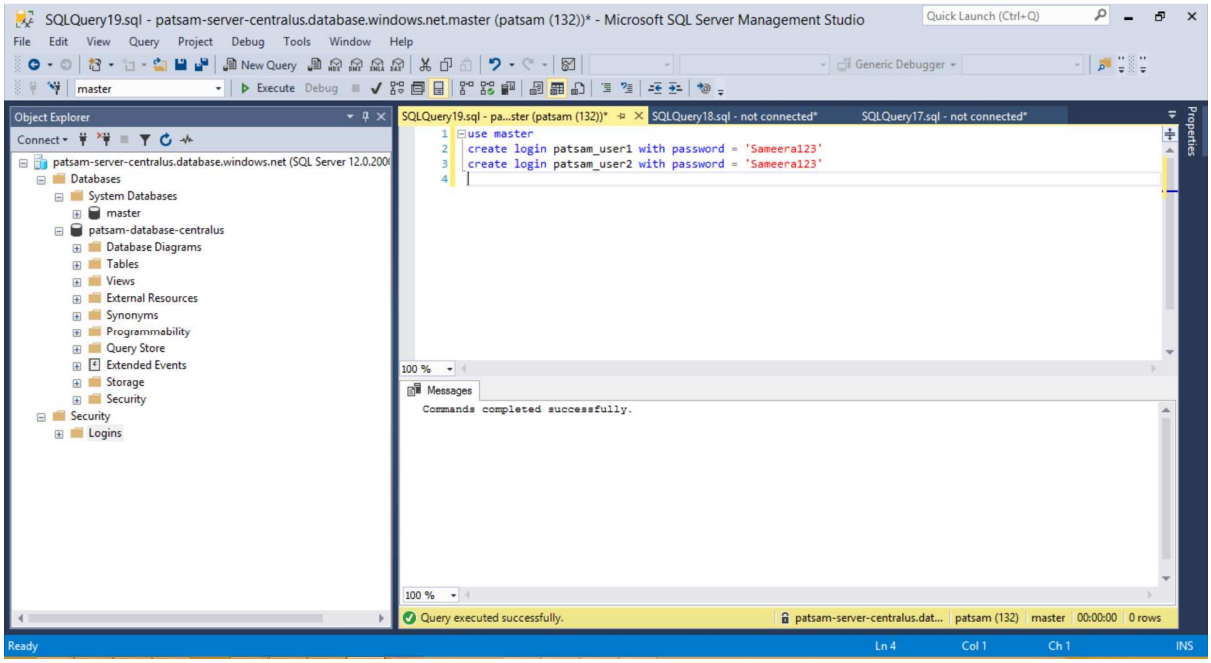
So, we are going to create 2 users:

- patsam_user1
- patsam_user2

<u>Create logins</u>

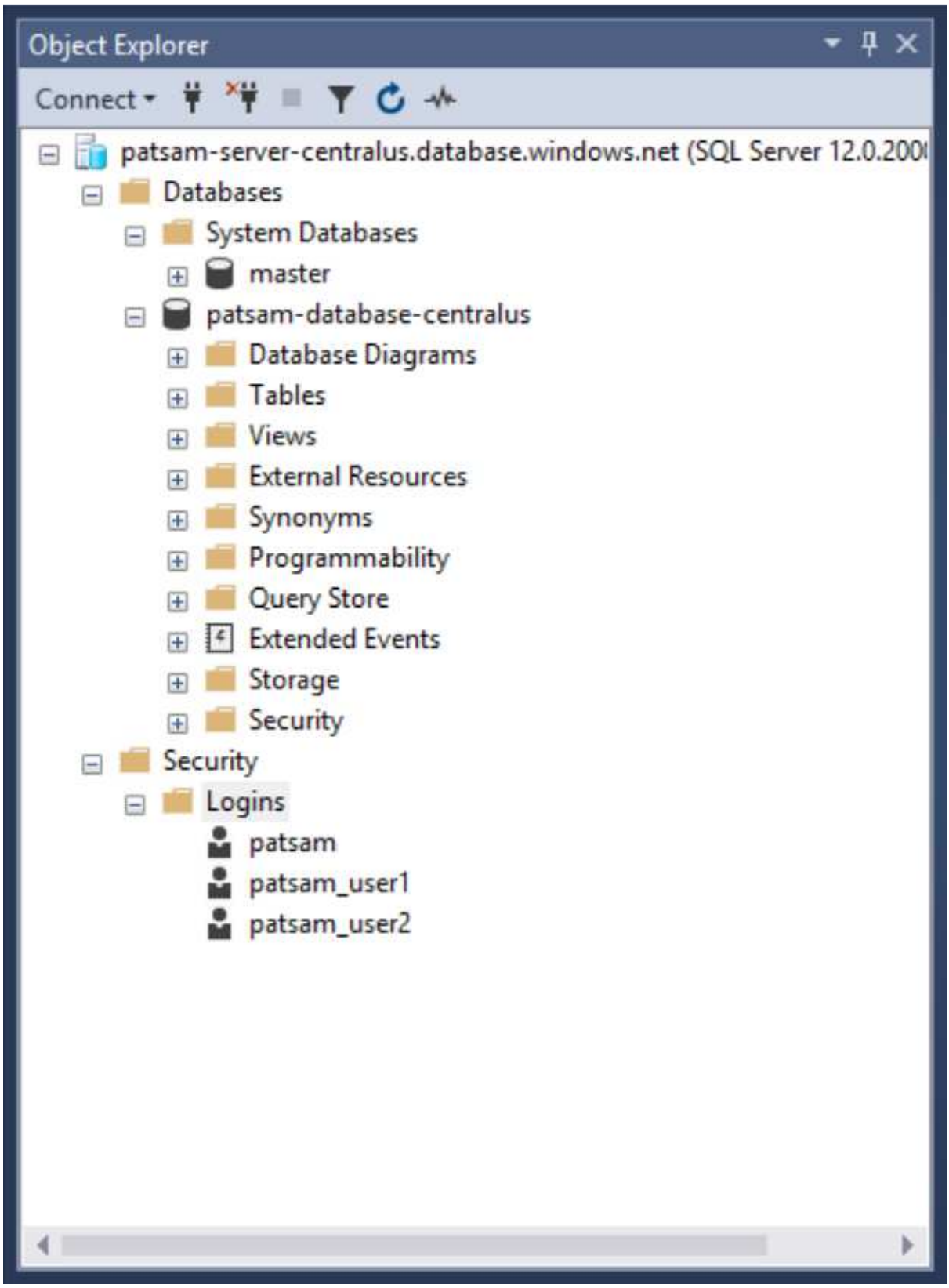
Select “master” Database and run bellow query.

```
use master
create login patsam_user1 with password = 'Sameera123'
create login patsam_user2 with password = 'Sameera123'
```



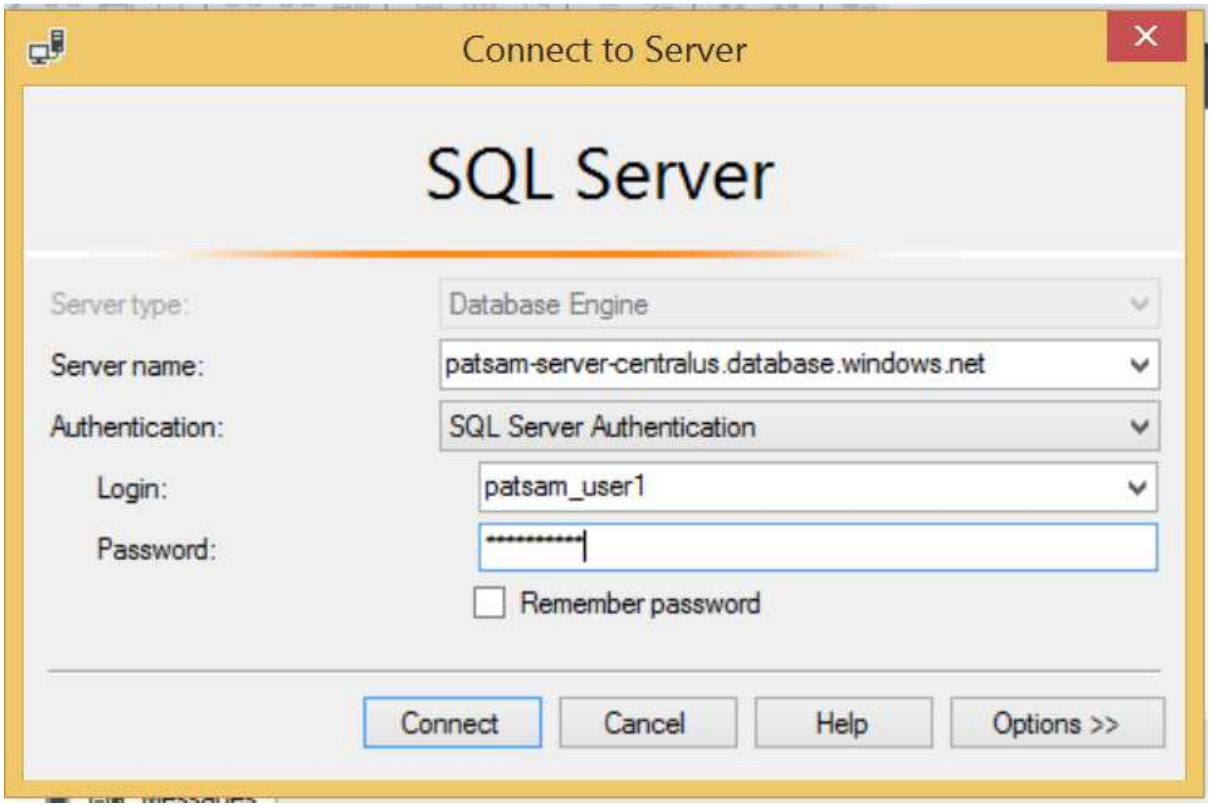
We can see the 2 newly created Logins under Security.

- master → Security → Logins



Now try to login to the Database using these Logins.

- Login: patsam_user1



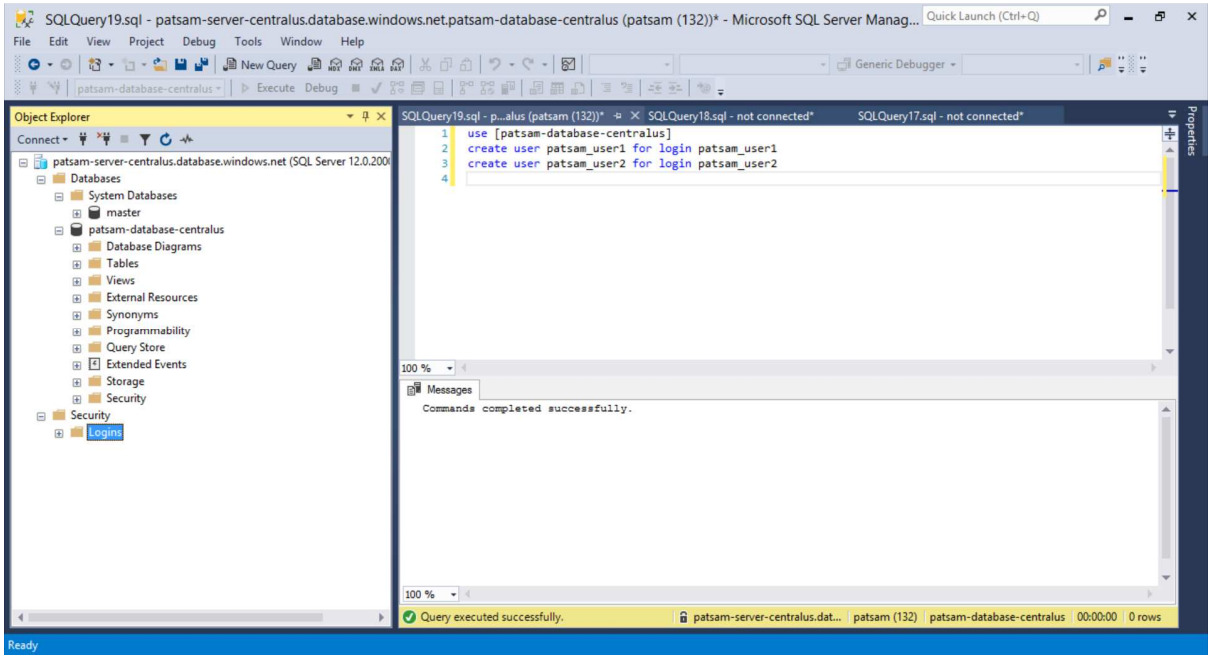
You get bellow error.

That is because we haven't created any User for this Login.

Create users

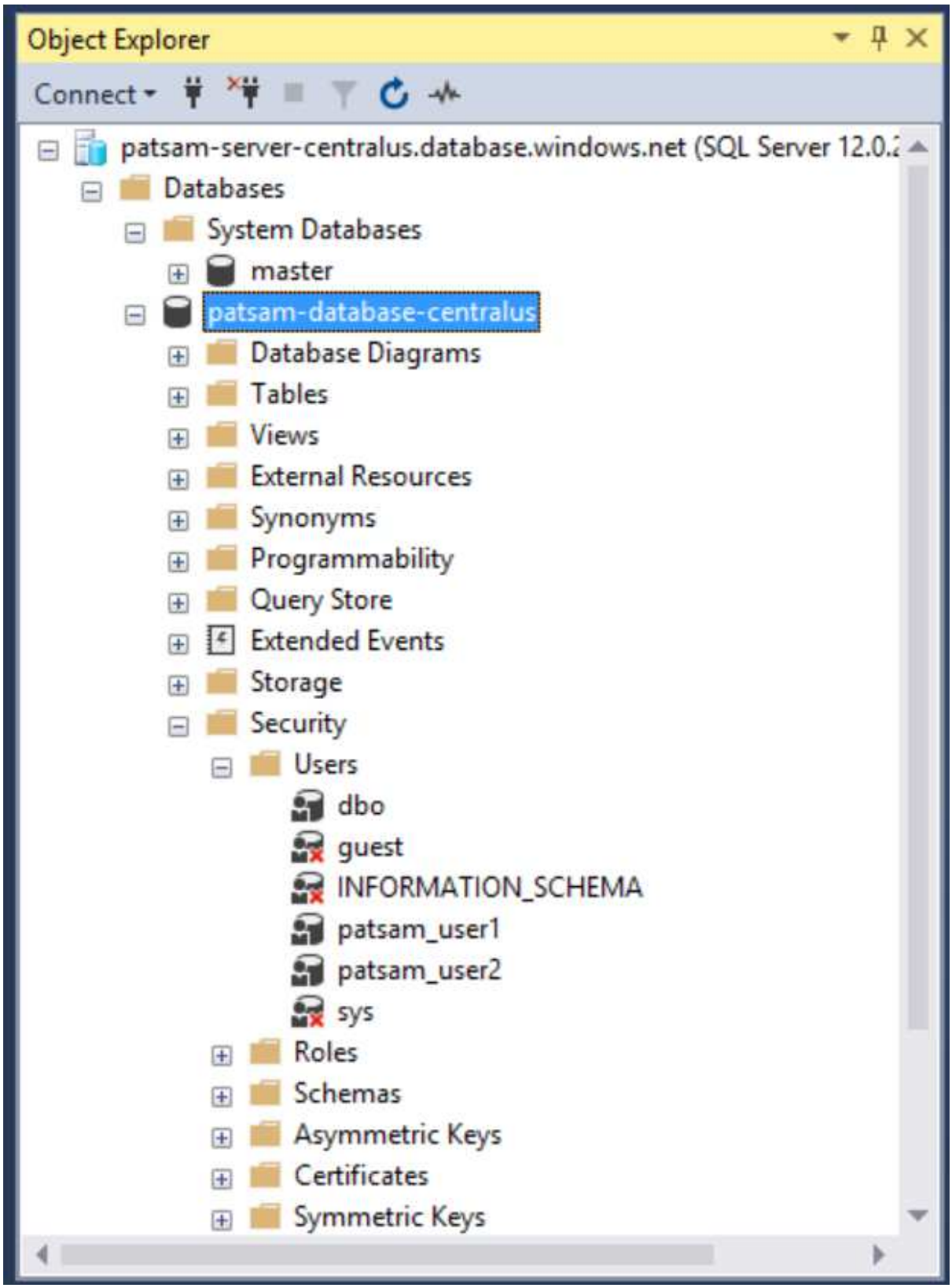
Select “patsam-database-centralus” Database and run bellow query.

```
use [patsam-database-centralus]
create user patsam_user1 for login patsam_user1
create user patsam_user2 for login patsam_user2
```



We can see the 2 newly created Users under Security.

- patsam-database-centralus → Security → Users

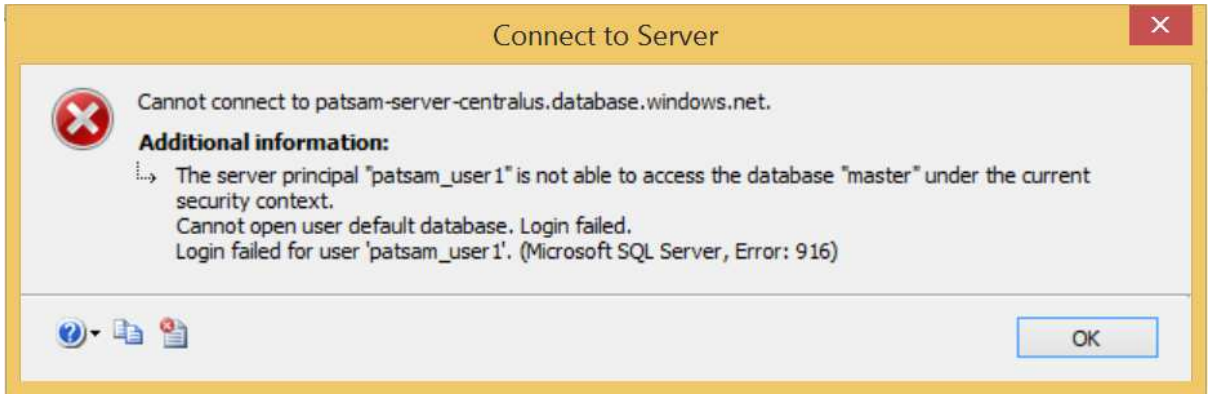


Now try to login to the Database using these Users.

- Login: patsam_user1

You get bellow error.

That is because patsam_user1 trying to connect to “master” database.

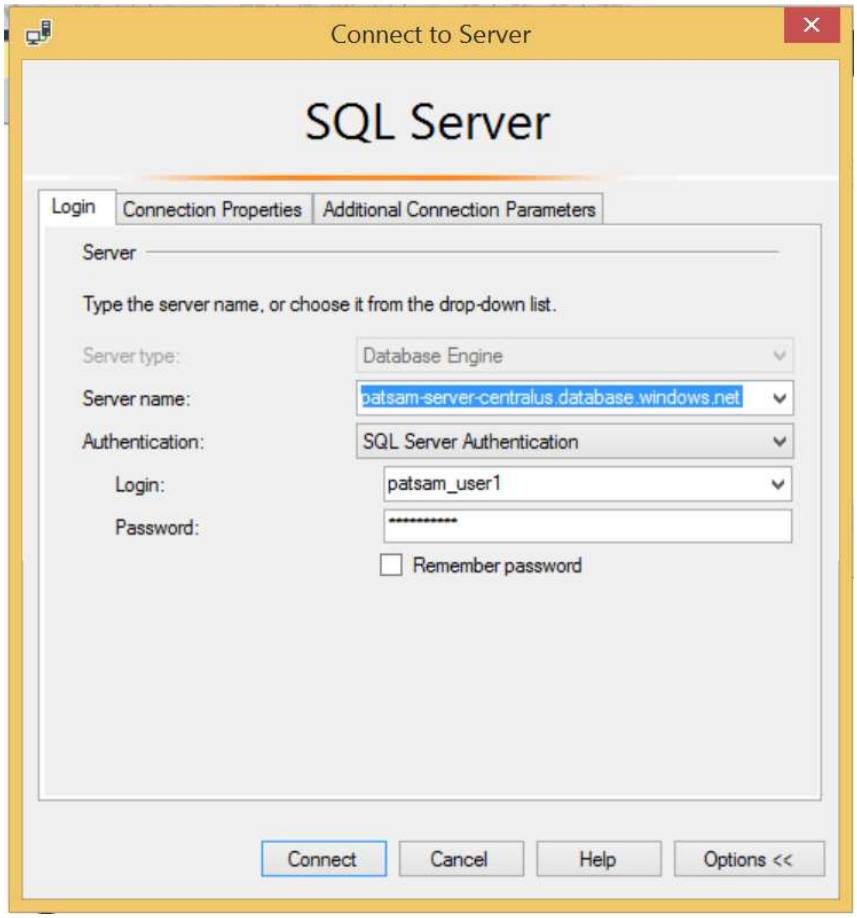


We haven’t created patsam_user1 for master Database. We created patsam_user1 for patsam-database-centralus Database.

Again, try to login to Database using SQL Server Management.

- Login: patsam_user1

Enter login details and before clicking Connect button Click on Options button.

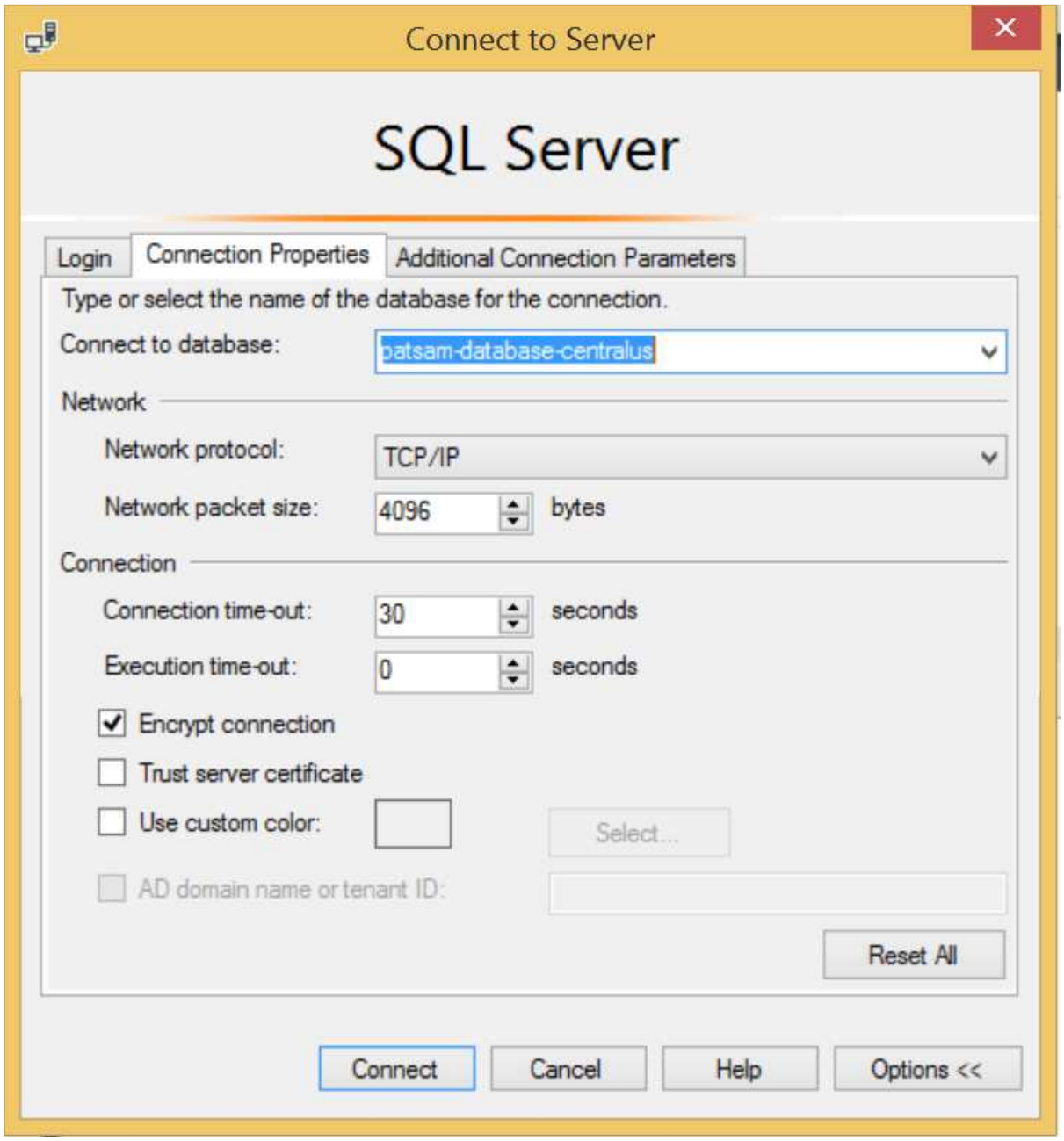


Click on Connection Properties tab.

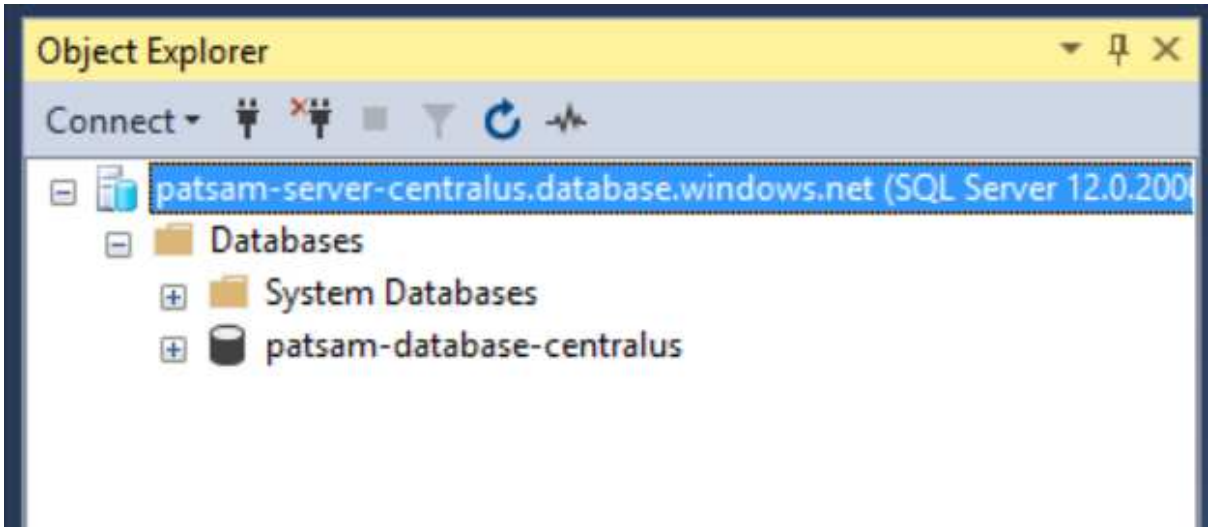
Type in the Database name we are trying to connect.

- Database name: patsam-database-centralus

Click Connect now.

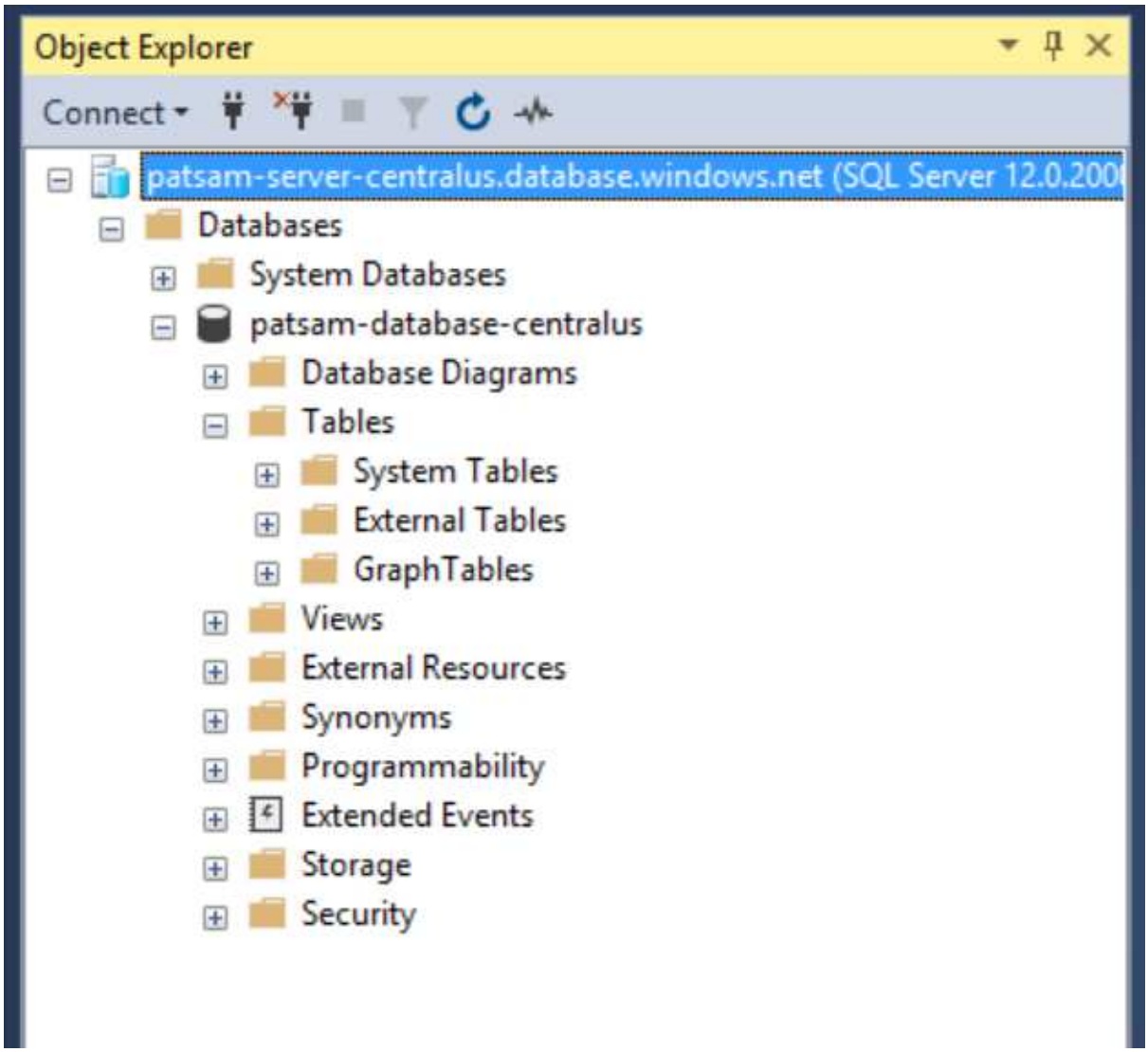


This time it connects.



But if you try to expand Tables and view them, you notice there are no tables listed.

Thant is because I haven't assigned any Roles to the User we created.

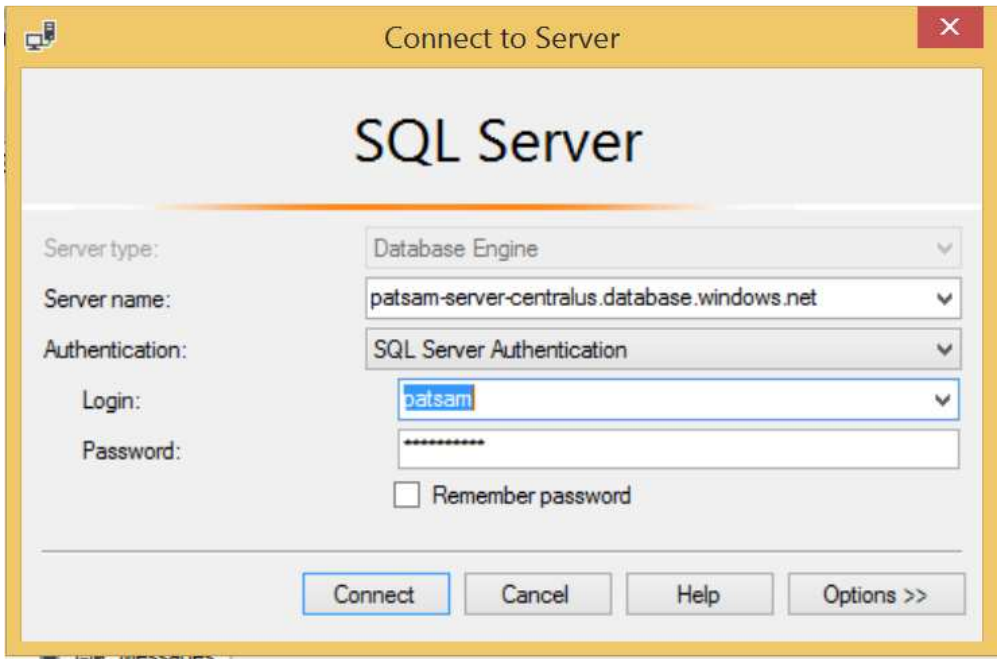


Assign roles

Next, we need to assign Roles to the Users we created earlier.

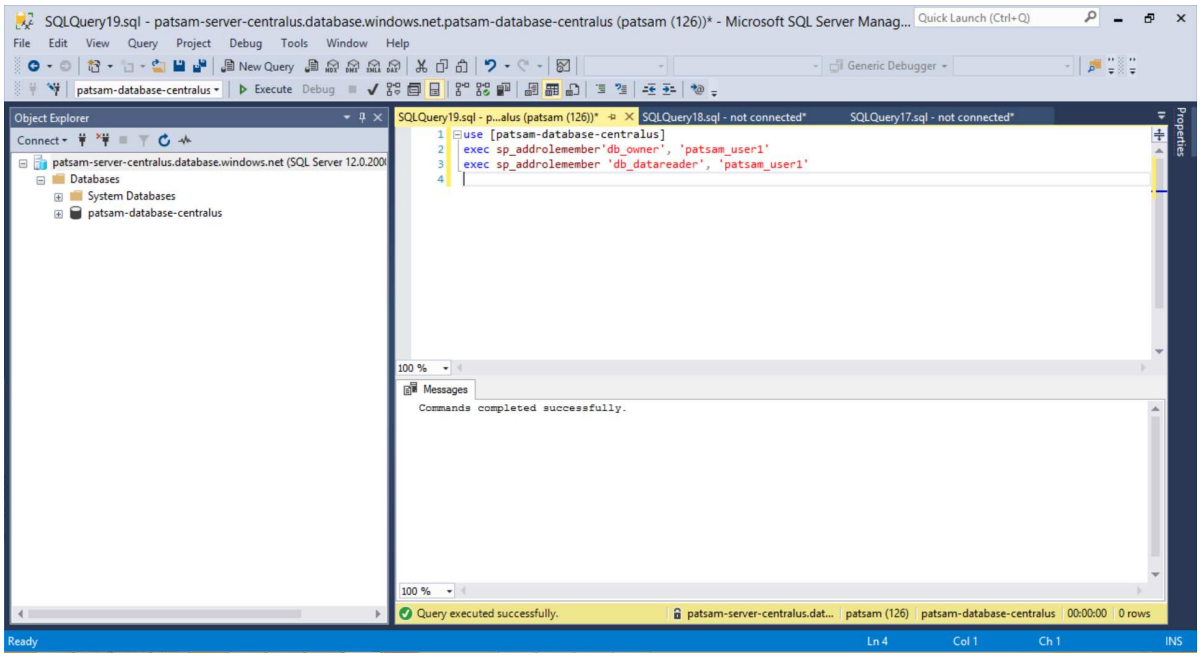
Log in to the Database using SQL Server Management.

- Login: patsam



Run bellow query.

```
use [patsam-database-centralus]
exec sp_addrolemember'db_owner', 'patsam_user1'
exec sp_addrolemember 'db_datareader', 'patsam_user2'
```



Step 03: Test Dynamic Data Masking using different logins

We are going to test out the Dynamic Data Masking using the 2 new Users we created.

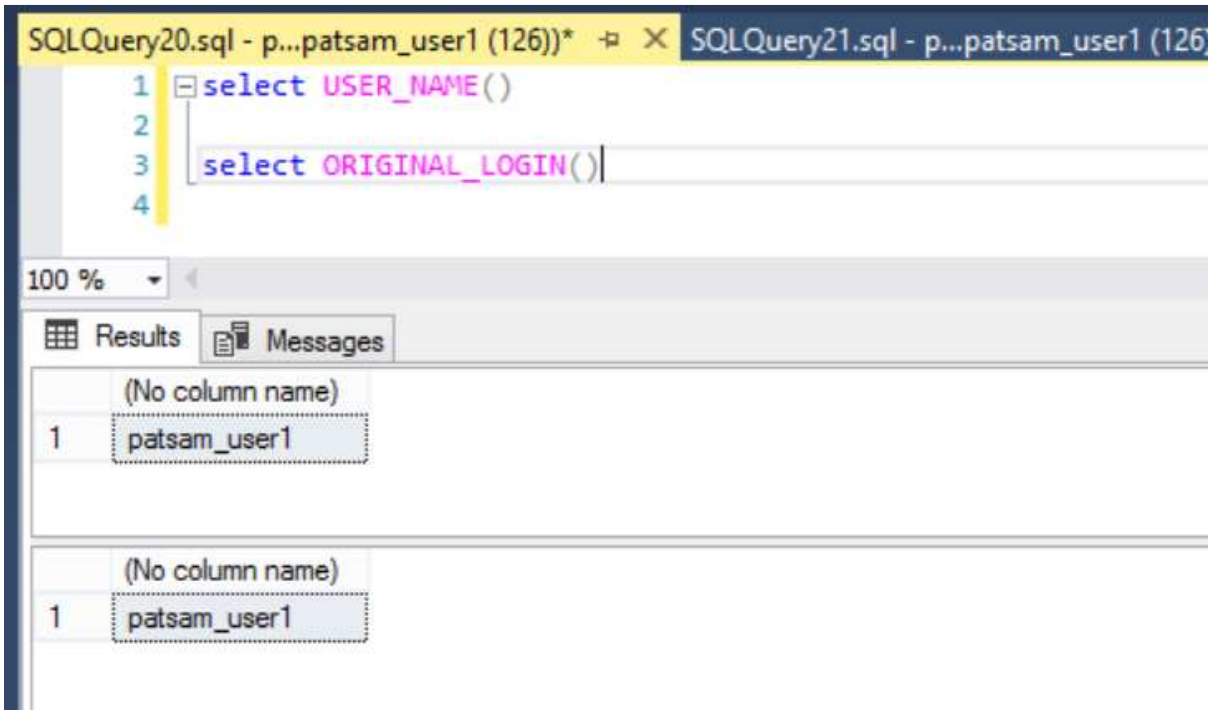
User	Role
patsam_user1	db_owner
patsam_user1	db_datareader

Log in to the SQL Server Management using patsam_user1(db_owner)

- Login: patsam_user1

Just to make sure, run bellow query to find out about login details.

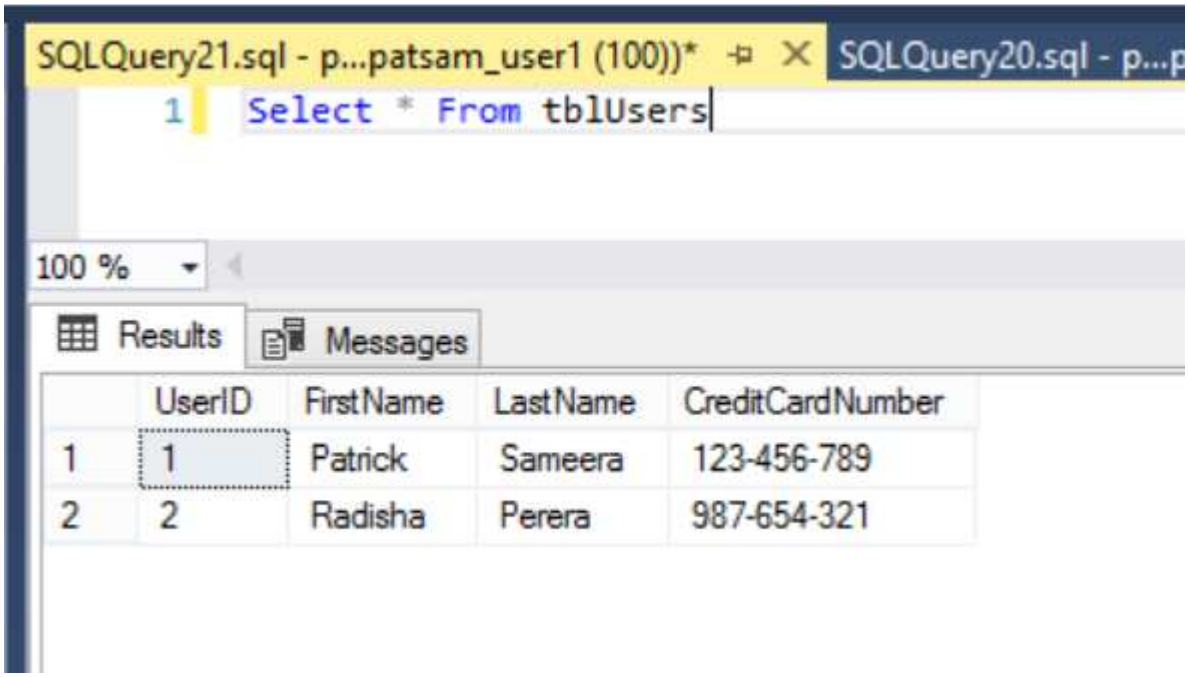
```
select USER_NAME()

select ORIGINAL_LOGIN()
```



Run bellow query to view the data on tblUsers.

```
Select * From tblUsers
```

Notice even though we masked column CreditCardNember column, we still can view the data. That is because User patsam_user1 is in db_owner Role.

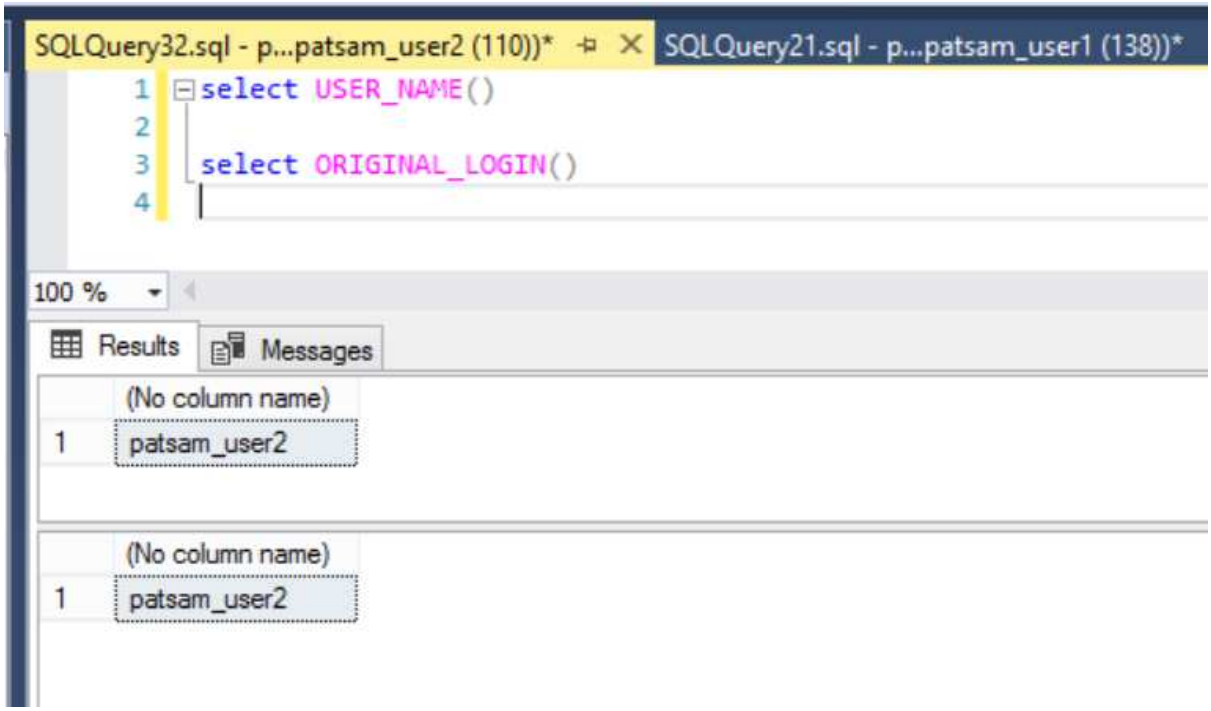


Log in to the SQL Server Management using patsam_user2(db_datareader)

- Login: patsam_user2

Just to make sure, run bellow query to find out about login details.

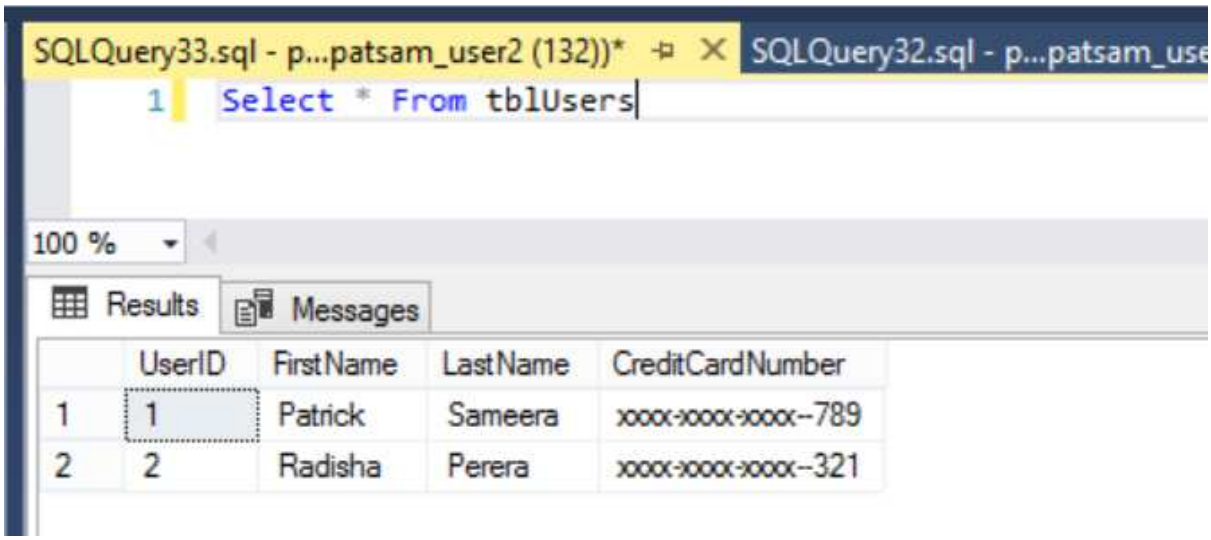
```
select USER_NAME()

select ORIGINAL_LOGIN()
```



Run bellow query to view the data on tblUsers.

```
Select * From tblUsers
```

Notice data in the column CreditCardNember is now masked. That is because User patsam_user2 is in db_datareader Role.



Run bellow query to see which User has what Permission.

```
        --1) List all access provisioned to a SQL user or Windows user/group directly
        SELECT
         [UserType] = CASE princ.[type]
                          WHEN 'S' THEN 'SQL User'
                          WHEN 'U' THEN 'Windows User'
                          WHEN 'G' THEN 'Windows Group'
                      END,
         [DatabaseUserName] = princ.[name],
         [LoginName]        = ulogin.[name],
         [Role]             = NULL,
         [PermissionType]   = perm.[permission_name],
         [PermissionState]  = perm.[state_desc],
         [ObjectType] = CASE perm.[class]
                          WHEN 1 THEN obj.[type_desc]        -- Schema-contained
objects
                          ELSE perm.[class_desc]             -- Higher-level objects
                      END,
         [Schema] = objschem.[name],
         [ObjectName] = CASE perm.[class]
                          WHEN 3 THEN permschem.[name]       -- Schemas
                          WHEN 4 THEN imp.[name]             -- Impersonations
                          ELSE OBJECT_NAME(perm.[major_id])  -- General objects
                      END,
         [ColumnName] = col.[name]
    FROM
         --Database user
         sys.database_principals            AS princ
         --Login accounts
         LEFT JOIN sys.database_principals    AS ulogin    ON ulogin.[sid] =
princ.[sid]
         --Permissions
         LEFT JOIN sys.database_permissions AS perm      ON perm.[grantee_principal_id]
= princ.[principal_id]
         LEFT JOIN sys.schemas              AS permschem ON permschem.[schema_id] =
perm.[major_id]
         LEFT JOIN sys.objects              AS obj       ON obj.[object_id] =
perm.[major_id]
         LEFT JOIN sys.schemas              AS objschem  ON objschem.[schema_id] =
obj.[schema_id]
         --Table columns
         LEFT JOIN sys.columns              AS col       ON col.[object_id] =
perm.[major_id]
                                                            AND col.[column_id] =
perm.[minor_id]
         --Impersonations
         LEFT JOIN sys.database_principals  AS imp       ON imp.[principal_id] =
perm.[major_id]
    WHERE
         princ.[type] IN ('S','U','G')
         -- No need for these system accounts
         AND princ.[name] NOT IN ('sys', 'INFORMATION_SCHEMA')

UNION

    --2) List all access provisioned to a SQL user or Windows user/group through a
database or application role
    SELECT
         [UserType] = CASE membprinc.[type]
                          WHEN 'S' THEN 'SQL User'
                          WHEN 'U' THEN 'Windows User'
```

```
                        WHEN 'G' THEN 'Windows Group'
                    END,
        [DatabaseUserName] = membprinc.[name],
        [LoginName]        = ulogin.[name],
        [Role]             = roleprinc.[name],
        [PermissionType]   = perm.[permission_name],
        [PermissionState]  = perm.[state_desc],
        [ObjectType] = CASE perm.[class]
                        WHEN 1 THEN obj.[type_desc]        -- Schema-contained
objects
                        ELSE perm.[class_desc]             -- Higher-level objects
                    END,
        [Schema] = objschem.[name],
        [ObjectName] = CASE perm.[class]
                        WHEN 3 THEN permschem.[name]       -- Schemas
                        WHEN 4 THEN imp.[name]             -- Impersonations
                        ELSE OBJECT_NAME(perm.[major_id])  -- General objects
                    END,
        [ColumnName] = col.[name]
    FROM
        --Role/member associations
        sys.database_role_members          AS members
        --Roles
        JOIN      sys.database_principals  AS roleprinc ON roleprinc.[principal_id] =
members.[role_principal_id]
        --Role members (database users)
        JOIN      sys.database_principals  AS membprinc ON membprinc.[principal_id] =
members.[member_principal_id]
        --Login accounts
        LEFT JOIN sys.database_principals    AS ulogin    ON ulogin.[sid] =
membprinc.[sid]
        --Permissions
        LEFT JOIN sys.database_permissions AS perm      ON perm.[grantee_principal_id]
= roleprinc.[principal_id]
        LEFT JOIN sys.schemas              AS permschem ON permschem.[schema_id] =
perm.[major_id]
        LEFT JOIN sys.objects              AS obj       ON obj.[object_id] =
perm.[major_id]
        LEFT JOIN sys.schemas              AS objschem  ON objschem.[schema_id] =
obj.[schema_id]
        --Table columns
        LEFT JOIN sys.columns              AS col       ON col.[object_id] =
perm.[major_id]
                                                           AND col.[column_id] =
perm.[minor_id]
        --Impersonations
        LEFT JOIN sys.database_principals  AS imp       ON imp.[principal_id] =
perm.[major_id]
    WHERE
        membprinc.[type] IN ('S','U','G')
        -- No need for these system accounts
        AND membprinc.[name] NOT IN ('sys', 'INFORMATION_SCHEMA')

UNION

    --3) List all access provisioned to the public role, which everyone gets by
default
    SELECT
        [UserType]         = '{All Users}',
        [DatabaseUserName] = '{All Users}',
        [LoginName]        = '{All Users}',
        [Role]             = roleprinc.[name],
```

```
        [PermissionType]    = perm.[permission_name],
        [PermissionState]   = perm.[state_desc],
        [ObjectType] = CASE perm.[class]
                            WHEN 1 THEN obj.[type_desc]        -- Schema-contained
objects
                            ELSE perm.[class_desc]             -- Higher-level objects
                        END,
        [Schema] = objschem.[name],
        [ObjectName] = CASE perm.[class]
                            WHEN 3 THEN permschem.[name]       -- Schemas
                            WHEN 4 THEN imp.[name]             -- Impersonations
                            ELSE OBJECT_NAME(perm.[major_id])  -- General objects
                        END,
        [ColumnName] = col.[name]
    FROM
        --Roles
        sys.database_principals            AS roleprinc
        --Role permissions
        LEFT JOIN sys.database_permissions AS perm      ON perm.[grantee_principal_id]
= roleprinc.[principal_id]
        LEFT JOIN sys.schemas              AS permschem ON permschem.[schema_id] =
perm.[major_id]
        --All objects
        JOIN      sys.objects              AS obj       ON obj.[object_id] =
perm.[major_id]
        LEFT JOIN sys.schemas              AS objschem  ON objschem.[schema_id] =
obj.[schema_id]
        --Table columns
        LEFT JOIN sys.columns              AS col       ON col.[object_id] =
perm.[major_id]
                                                           AND col.[column_id] =
perm.[minor_id]
        --Impersonations
        LEFT JOIN sys.database_principals  AS imp       ON imp.[principal_id] =
perm.[major_id]
    WHERE
        roleprinc.[type] = 'R'
        AND roleprinc.[name] = 'public'
        AND obj.[is_ms_shipped] = 0

ORDER BY
    [UserType],
    [DatabaseUserName],
    [LoginName],
    [Role],
    [Schema],
    [ObjectName],
    [ColumnName],
    [PermissionType],
    [PermissionState],
    [ObjectType]
```

SQLQuery31.sql - p...alus (patsam (132))* | SQLQuery20.sql - p...patsam_user1 (105))* | SQLQuery21.sql - p...patsam_user1 (126))*

```sql
--1) List all access provisioned to a SQL user or Windows user/group directly
SELECT
    [UserType] = CASE princ.[type]
                    WHEN 'S' THEN 'SQL User'
                    WHEN 'U' THEN 'Windows User'
                    WHEN 'G' THEN 'Windows Group'
                 END,
    [DatabaseUserName] = princ.[name],
    [LoginName]        = ulogin.[name],
    [Role]             = NULL,
    [PermissionType]   = perm.[permission_name],
    [PermissionState]  = perm.[state_desc],
    [ObjectType] = CASE perm.[class]
                    WHEN 1 THEN obj.[type_desc]        -- Schema-contained objects
                    ELSE perm.[class_desc]             -- Higher-level objects
                 END
```

100 %

Results | Messages

	UserType	DatabaseUserName	LoginName	Role	PermissionType	PermissionState	ObjectType	Schema	ObjectName	ColumnNam
1	SQL User	dbo	dbo	NULL	CONNECT	GRANT	DATABASE	NULL	NULL	NULL
2	SQL User	dbo	dbo	db_owner	NULL	NULL	NULL	NULL	NULL	NULL
3	SQL User	guest	guest	NULL	NULL	NULL	NULL	NULL	NULL	NULL
4	SQL User	patsam_user1	patsam_user1	NULL	CONNECT	GRANT	DATABASE	NULL	NULL	NULL
5	SQL User	patsam_user1	patsam_user1	db_owner	NULL	NULL	NULL	NULL	NULL	NULL
6	SQL User	patsam_user2	patsam_user2	NULL	CONNECT	GRANT	DATABASE	NULL	NULL	NULL
7	SQL User	patsam_user2	patsam_user2	db_datareader	NULL	NULL	NULL	NULL	NULL	NULL

```sql
select name as username,

       create_date,
       modify_date,
       type_desc as type,
       authentication_type_desc as authentication_type
from sys.database_principals
where type not in ('A', 'G', 'R', 'X')
      and sid is not null
order by username;
```

SQLQuery36.sql - p...patsam_user1 (127))*

```sql
select name as username,
       create_date,
       modify_date,
       type_desc as type,
       authentication_type_desc as authentication_type
from sys.database_principals
where type not in ('A', 'G', 'R', 'X')
      and sid is not null
order by username;
```

100 %

Results | Messages

	username	create_date	modify_date	type	authentication_type
1	dbo	2003-04-08 09:10:42.287	2019-09-19 04:32:26.017	SQL_USER	INSTANCE
2	guest	2003-04-08 09:10:42.317	2003-04-08 09:10:42.317	SQL_USER	NONE
3	patsam_user1	2019-09-20 04:29:57.390	2019-09-20 04:29:57.390	SQL_USER	INSTANCE
4	patsam_user2	2019-09-20 04:29:57.413	2019-09-20 04:29:57.413	SQL_USER	INSTANCE

Query executed successfully. | patsam-server-centralus.dat... | patsam_user1 (127) | patsam-database-centralus | 00:00:00 | 4 rows