Exercise 10 – Database Security and Permissions

Stewart Johnston
CIS 150 – Intro to Database Administration
NCMC
`johnstons1@student.ncmich.edu`

December 1, 2018

For your convenience, you will find that most numbers are in fact hyperlinked references across the document. Likewise, any other URLs found, such as links to stackoverflow or docs.microsoft, will be hyperlinks. Unfortunately, I've come to find that the syntax highlighting that the listings package provides is limited to a smaller dielect of SQL than is being used here. T-SQL provides many extensions to the language which are not standard, and it is possible that the standard which the syntax highlighting is based on an even older dialect than the most recent standard. Not every keyword is underlined, which is unfortunate, but it still does 99% of what I want it to do, so I am not eager to change anything.

Contents

List of Figures

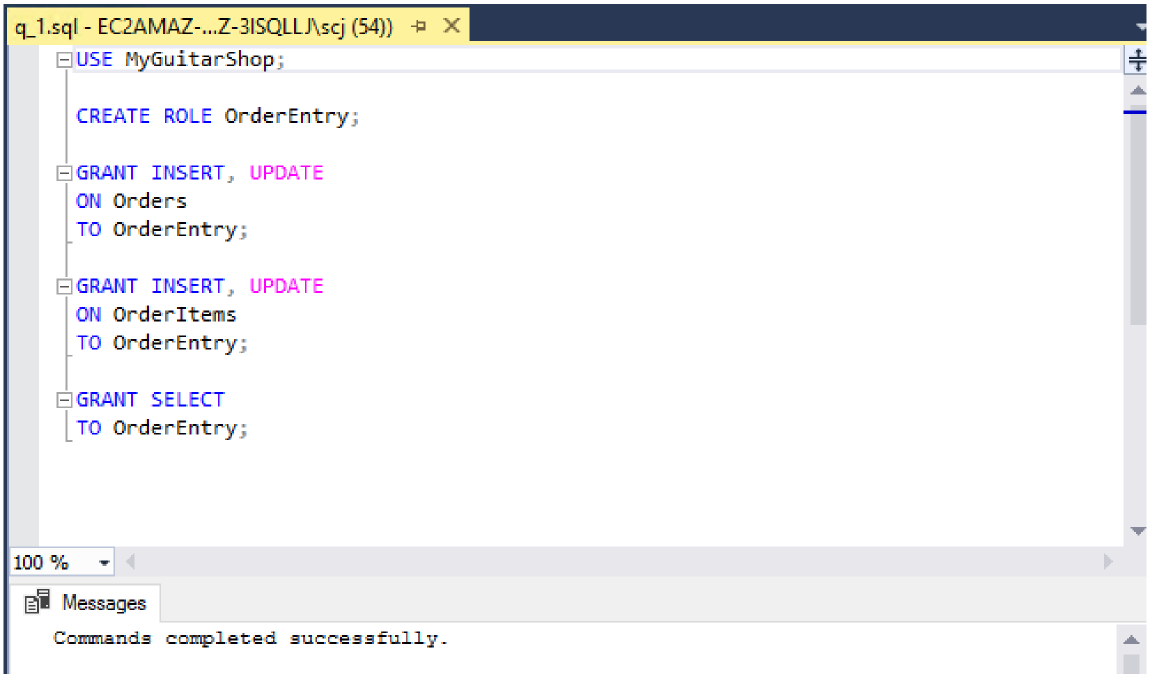
1 Create a role and assign permissions

See listing 1 and accompanying figure 1

Listing 1: Per 1

```
USE MyGuitarShop;

CREATE ROLE OrderEntry;

GRANT INSERT, UPDATE
ON Orders
TO OrderEntry;

GRANT INSERT, UPDATE
ON OrderItems
TO OrderEntry;

GRANT SELECT
TO OrderEntry;
```

Figure 1: Per 1



q_1.sql - EC2AMAZ-...Z-3ISQLLJ\scj (54))

```
USE MyGuitarShop;

CREATE ROLE OrderEntry;

GRANT INSERT, UPDATE
ON Orders
TO OrderEntry;

GRANT INSERT, UPDATE
ON OrderItems
TO OrderEntry;

GRANT SELECT
TO OrderEntry;
```

```
Commands completed successfully.
```

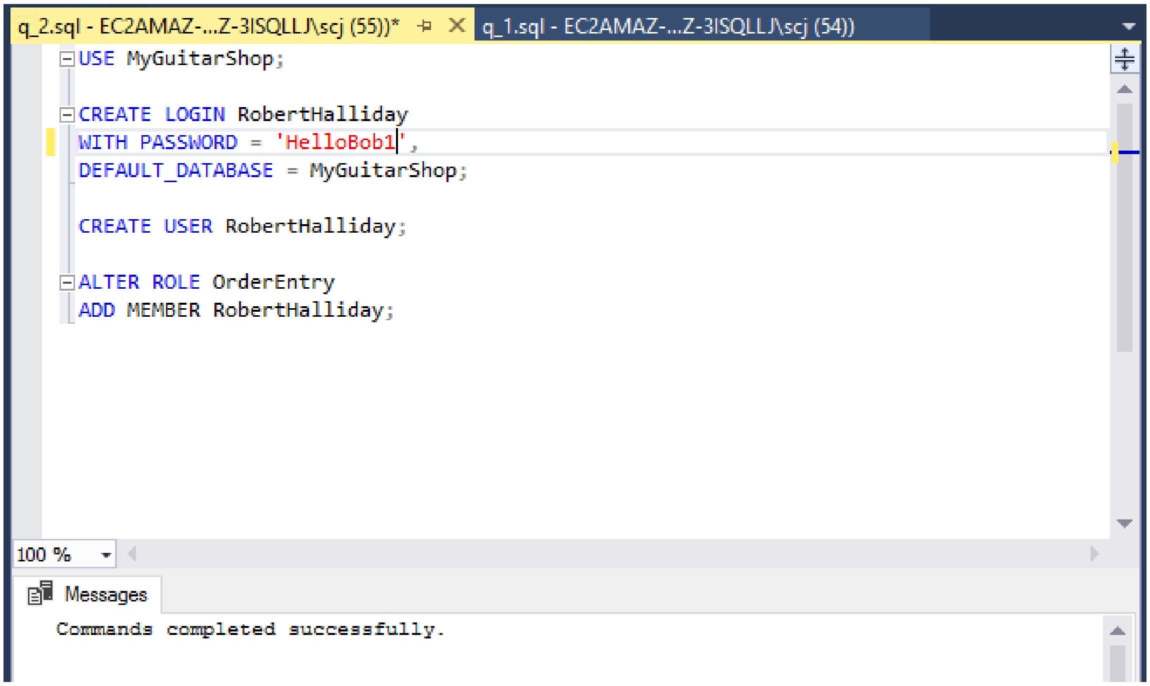
2 Create RobertHalliday login and user

See listing 2 and accompanying figure 2

Listing 2: Per 2

```
USE MyGuitarShop;

CREATE LOGIN RobertHalliday
WITH PASSWORD = 'HelloBob1',
DEFAULT_DATABASE = MyGuitarShop;

CREATE USER RobertHalliday;

ALTER ROLE OrderEntry
ADD MEMBER RobertHalliday;
```

Figure 2: Per 2



q_2.sql - EC2AMAZ-...Z-3ISQLLJ\scj (55))*
q_1.sql - EC2AMAZ-...Z-3ISQLLJ\scj (54))

```
USE MyGuitarShop;

CREATE LOGIN RobertHalliday
WITH PASSWORD = 'HelloBob1',
DEFAULT_DATABASE = MyGuitarShop;

CREATE USER RobertHalliday;

ALTER ROLE OrderEntry
ADD MEMBER RobertHalliday;
```

```
Commands completed successfully.
```

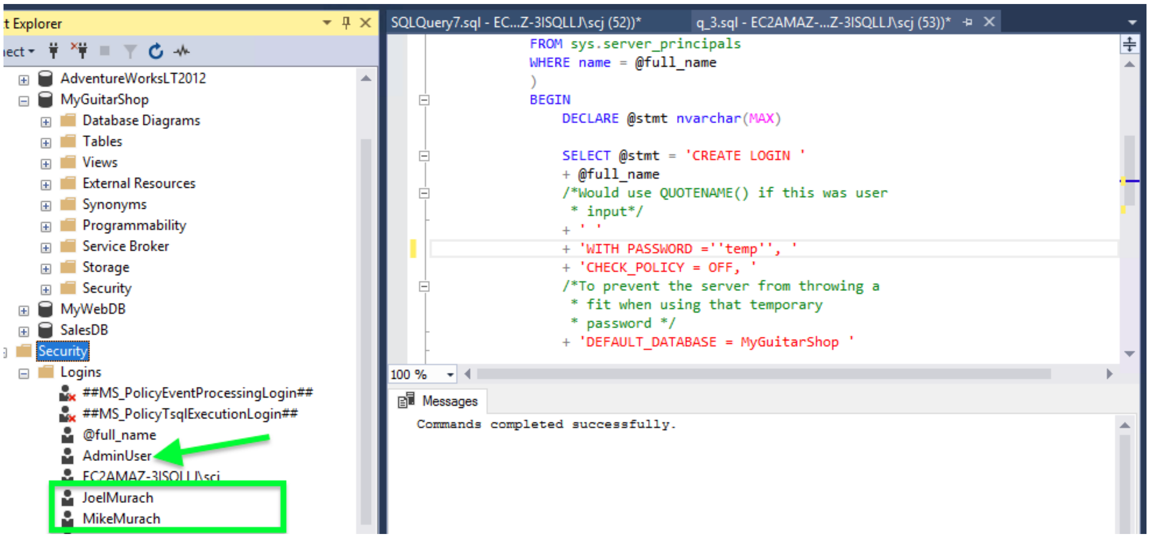
3 Dynamically make logins and users

The task was to programmatically, rather than statically, create logins and users for every person found in the Administrators table, and then associate them with the OrderEntry role created in 1. See listing 3 and accompanying figures 3, 4, 5

Listing 3: Per 3

```
USE MyGuitarShop;

DECLARE @full_name varchar(510)

DECLARE pending_user CURSOR FOR
SELECT (Firstname + Lastname) as name FROM Administrators

OPEN pending_user
FETCH NEXT FROM pending_user INTO @full_name
WHILE @@FETCH_STATUS = 0
      BEGIN
            /*Simple logic, check if a login already exists. Taken
            from https://stackoverflow.com/a/1945219 */
            IF NOT EXISTS (
                        SELECT *
                        FROM sys.server_principals
                        WHERE name = @full_name
                        )
                        BEGIN
                                DECLARE @stmt nvarchar(MAX)

                                SELECT @stmt = 'CREATE␣LOGIN␣'
                                + @full_name
                                /*Would use QUOTENAME() if this was user
                                 * input*/
                                + '␣'
                                + 'WITH␣PASSWORD␣=''temp'',␣'/*Double
                                up on single-quotes to escape them in a
                                string. I don't know why, and I can't
                                find an authoritative source. This was a
                                source of much frustration solved by
                                this answer:
                                https://stackoverflow.com/a/1586588 */
                                + 'CHECK_POLICY␣=␣OFF,␣'
                                /*To prevent the server from throwing a
                                 * fit when using that temporary
                                 * password */
                                + 'DEFAULT_DATABASE␣=␣MyGuitarShop␣'

                                /*Holy nested statements batman.
                                 * Microsoft recently changed their
                                 * documentation website, for some
                                 * reason unbenknownst to me. Having
                                 * done so, they've made it ludicrously
                                 * difficult to find any authoritative
                                 * information on how their dialect of
                                 * T-SQL actually works. Hours wasted
                                 * here: 5-6. */

                                EXEC sp_executesql @stmt

                                /*Absolute nonsense necessary because
                                 * CREATE LOGIN can't use a parameter
                                 * https://stackoverflow.com/a/1379471
                                 * */
END

IF NOT EXISTS (
        SELECT *
        FROM sys.database_principals
        WHERE name = @full_name
)
BEGIN
        DECLARE @creation nvarchar(MAX)
        SELECT @creation = 'CREATE␣USER␣'
        + @full_name
        EXEC sp_executesql @creation
END

DECLARE @role_addition nvarchar(MAX)
SELECT @role_addition = 'ALTER␣ROLE␣OrderEntry␣'
+ 'ADD␣MEMBER␣'
+ @full_name

EXEC sp_executesql @role_addition

FETCH NEXT FROM pending_user INTO @full_name
END

CLOSE pending_user
DEALLOCATE pending_user
```

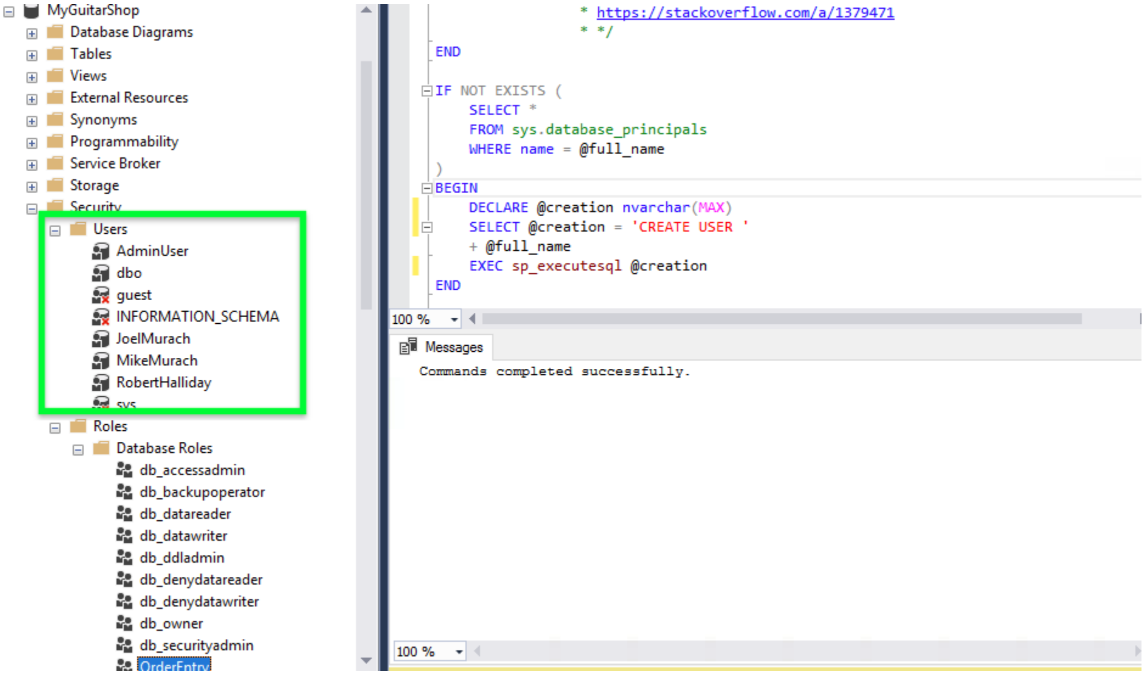
Figure 3: Per 3



```
            FROM sys.server_principals
            WHERE name = @full_name
            )
            BEGIN
                DECLARE @stmt nvarchar(MAX)

                SELECT @stmt = 'CREATE LOGIN '
                + @full_name
                /*Would use QUOTENAME() if this was user
                 * input*/
                + ' '
                + 'WITH PASSWORD =''temp'', '
                + 'CHECK_POLICY = OFF, '
                /*To prevent the server from throwing a
                 * fit when using that temporary
                 * password */
                + 'DEFAULT_DATABASE = MyGuitarShop '
```

```
Commands completed successfully.
```

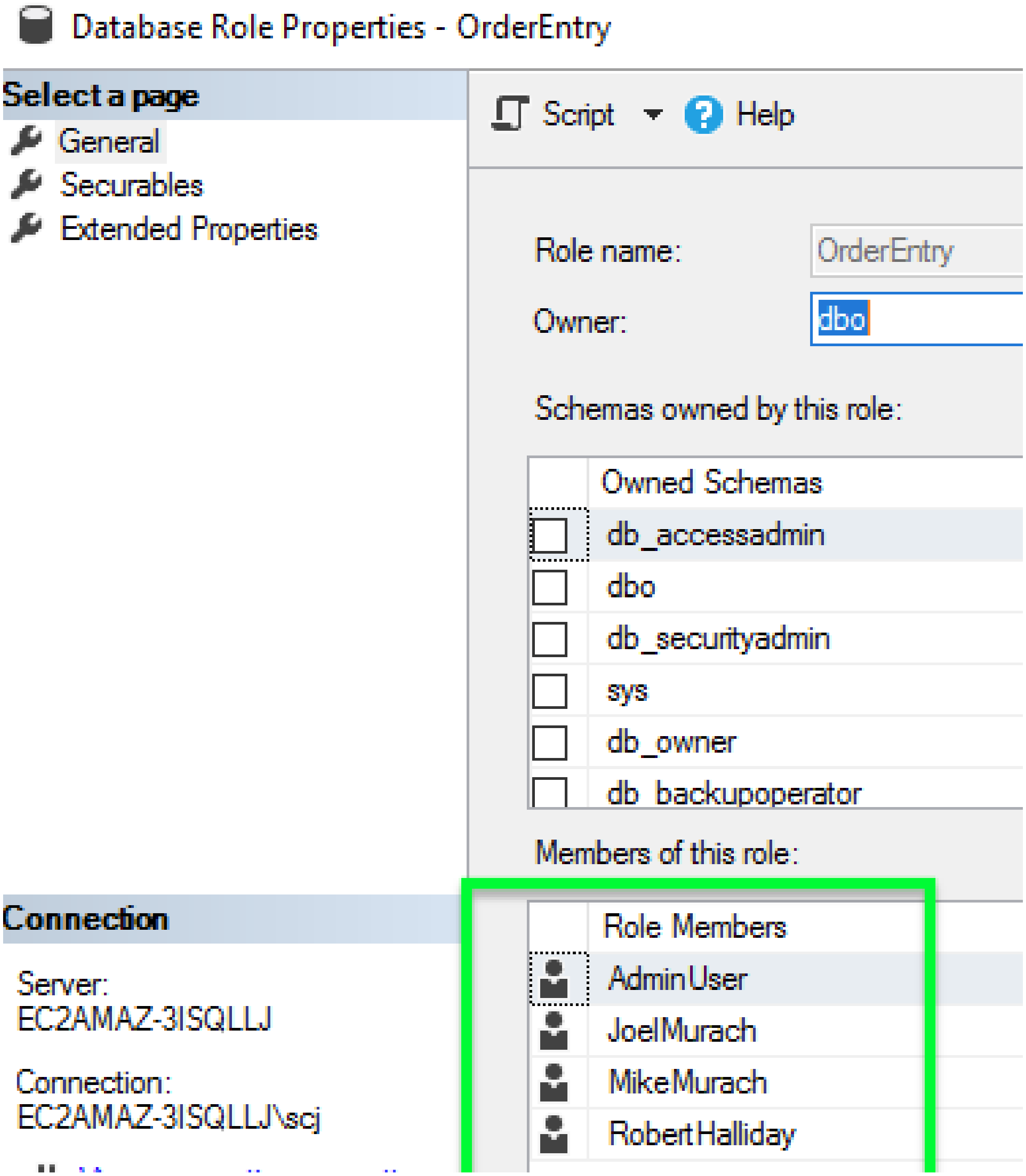
Figure 4: Per 3



```
                 * https://stackoverflow.com/a/1379471
                 * */
END

IF NOT EXISTS (
    SELECT *
    FROM sys.database_principals
    WHERE name = @full_name
)
BEGIN
    DECLARE @creation nvarchar(MAX)
    SELECT @creation = 'CREATE USER '
    + @full_name
    EXEC sp_executesql @creation
END
```

```
Commands completed successfully.
```

Figure 5: Per 3



Database Role Properties - OrderEntry

Select a page

- General
- Securables
- Extended Properties

Role name: OrderEntry

Owner: dbo

Schemas owned by this role:

	Owned Schemas
□	db_accessadmin
□	dbo
□	db_securityadmin
□	sys
□	db_owner
□	db_backupoperator

Members of this role:

	Role Members
	AdminUser
	JoelMurach
	MikeMurach
	RobertHalliday

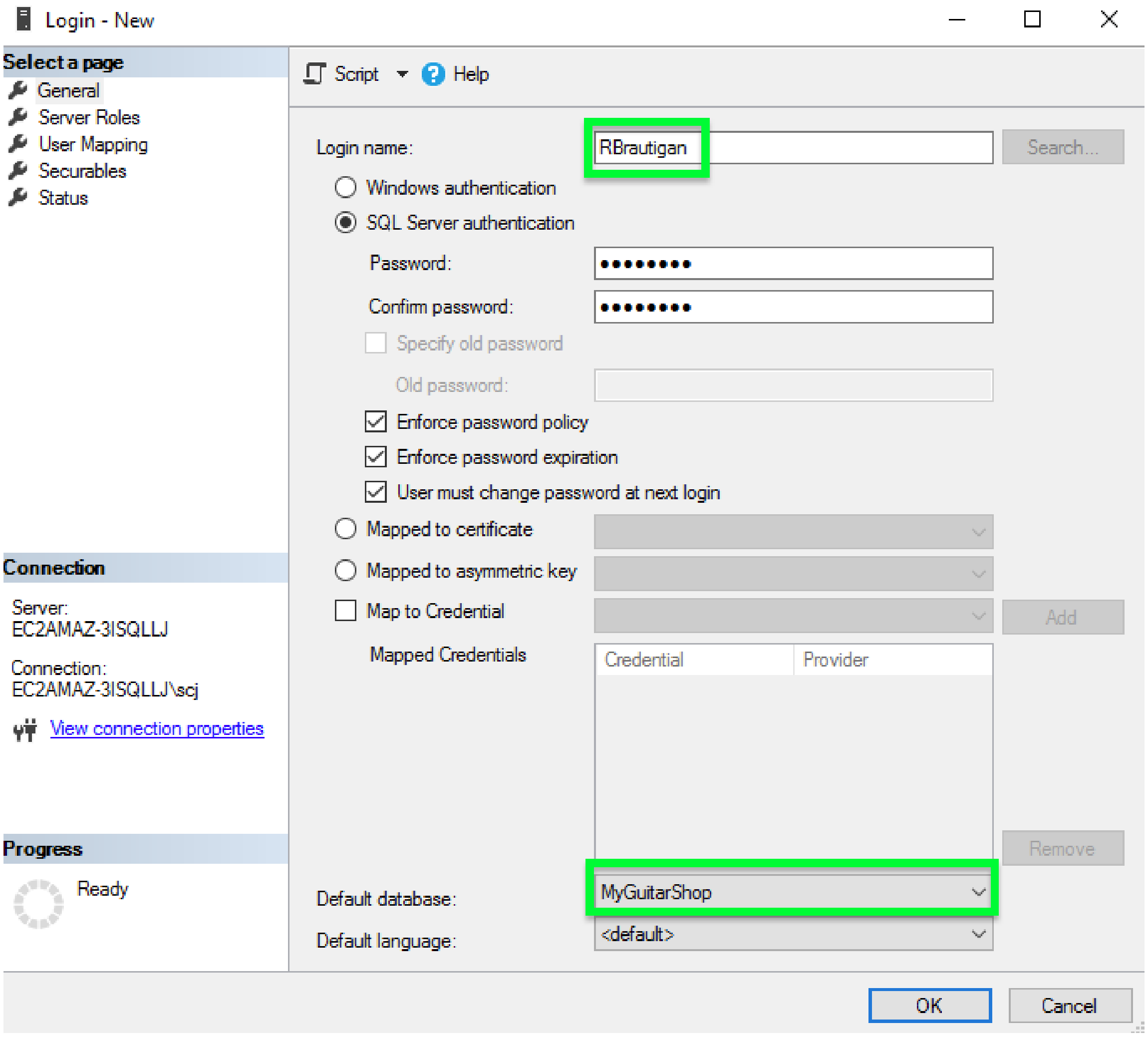
Connection

Server:
EC2AMAZ-3ISQLLJ

Connection:
EC2AMAZ-3ISQLLJ\scj

4 Make a user by hand with the GUI

Figure 6: Per 4



Login - New

Login name: RBrautigan

○ Windows authentication

● SQL Server authentication

Password: ••••••••

Confirm password: ••••••••

□ Specify old password

Old password:

☑ Enforce password policy

☑ Enforce password expiration

☑ User must change password at next login

○ Mapped to certificate

○ Mapped to asymmetric key

□ Map to Credential

Mapped Credentials

Credential	Provider

Default database: MyGuitarShop

Default language: <default>

Connection

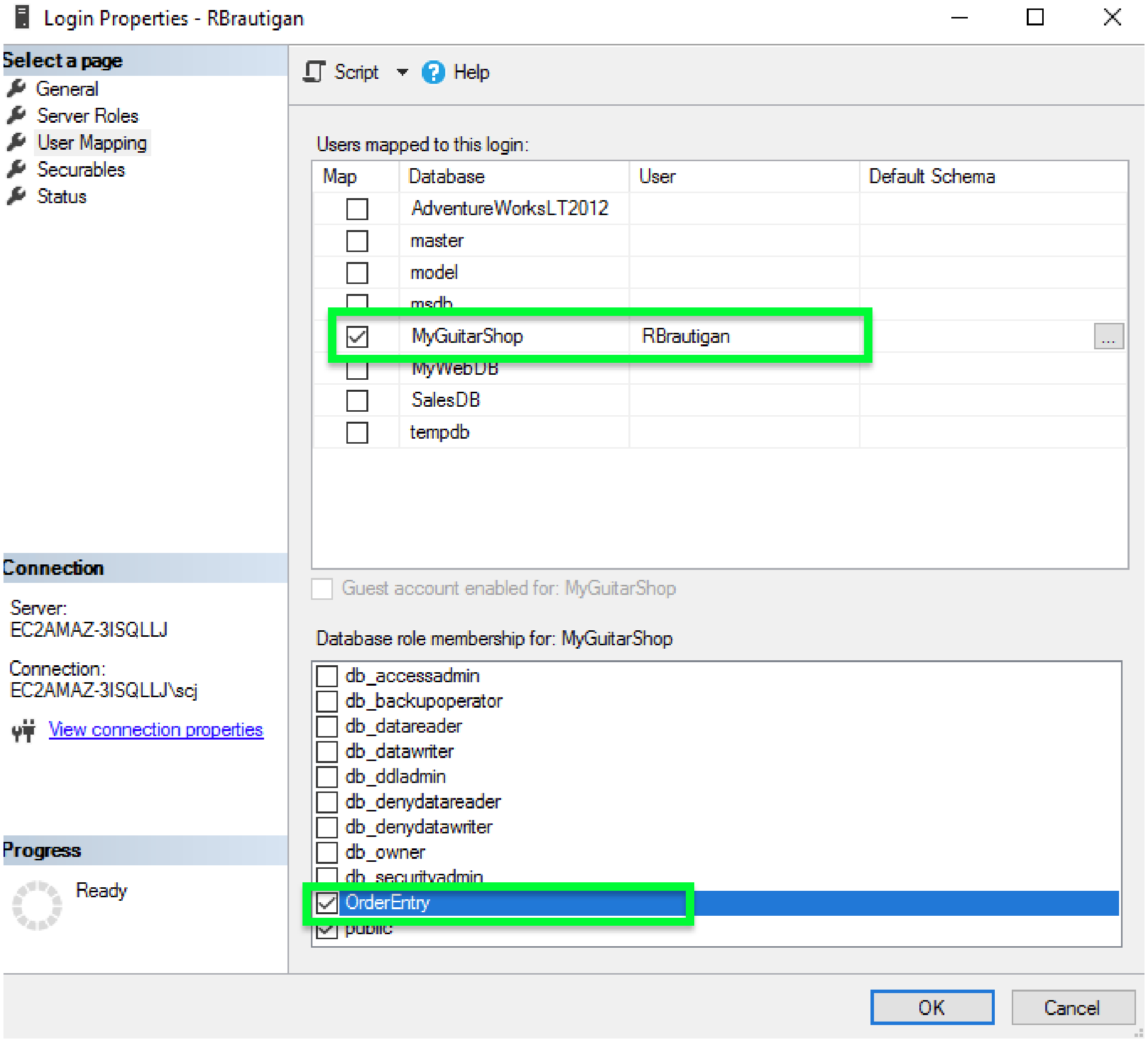
Server:
EC2AMAZ-3ISQLLJ

Connection:
EC2AMAZ-3ISQLLJ\scj

Progress

Ready

Figure 7: Per 4



Login Properties - RBrautigan

Server:
EC2AMAZ-3ISQLLJ

Connection:
EC2AMAZ-3ISQLLJ\scj

Ready

Users mapped to this login:

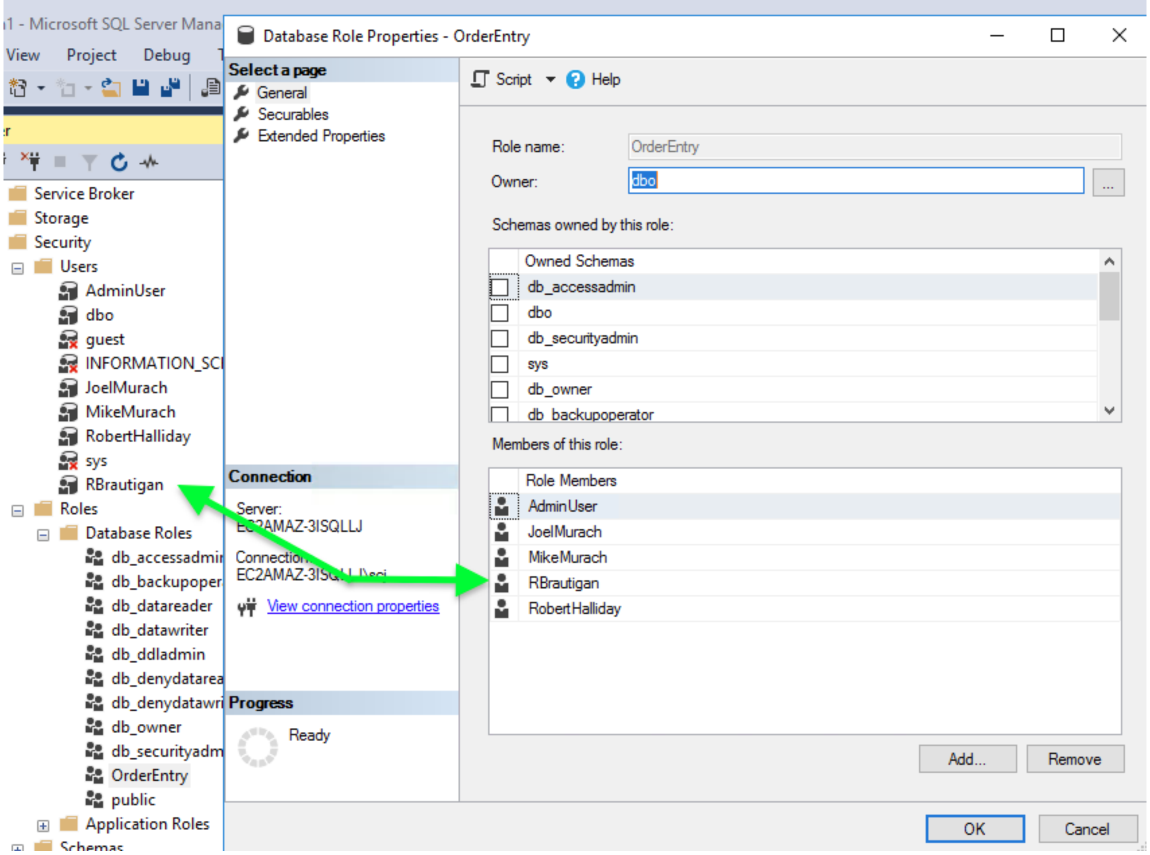
Map	Database	User	Default Schema
□	AdventureWorksLT2012		
□	master		
□	model		
□	msdb		
☑	MyGuitarShop	RBrautigan	
□	MyWebDB		
□	SalesDB		
□	tempdb		

□ Guest account enabled for: MyGuitarShop

Database role membership for: MyGuitarShop

□ db_accessadmin
□ db_backupoperator
□ db_datareader
□ db_datawriter
□ db_ddladmin
□ db_denydatareader
□ db_denydatawriter
□ db_owner
□ db_securityadmin
☑ OrderEntry
☑ public

Figure 8: Per 4



Role name: OrderEntry

Owner: dbo

Schemas owned by this role:

	Owned Schemas
□	db_accessadmin
□	dbo
□	db_securityadmin
□	sys
□	db_owner
□	db_backupoperator

Members of this role:

Role Members
AdminUser
JoelMurach
MikeMurach
RBrautigan
RobertHalliday

5 Remove OrderEntry role

Users need to be dropped from the role first See listing 4 and accompanying figure 9

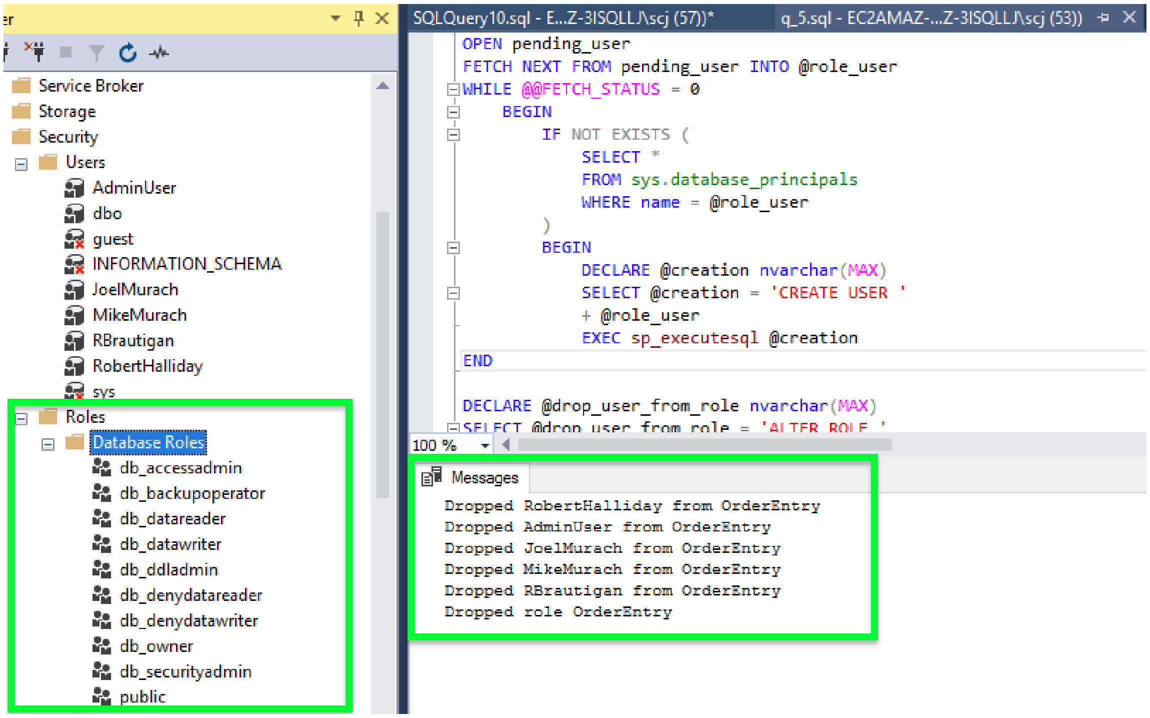
Listing 4: Per 5

```
USE MyGuitarShop;

DECLARE @role_user varchar(510)
DECLARE @role_name char(50)
SET @role_name = 'OrderEntry'

DECLARE pending_user CURSOR FOR
SELECT DP2.name AS DatabaseUserName -- The only value we actually want
 FROM sys.database_role_members AS DRM
 RIGHT OUTER JOIN sys.database_principals AS DP1
```

```
   ON DRM.role_principal_id = DP1.principal_id
 LEFT OUTER JOIN sys.database_principals AS DP2
   ON DRM.member_principal_id = DP2.principal_id
WHERE DP1.type = 'R' --Match only the Role
AND DP1.name = @role_name; --with the name OrderEntry

/*Edited from the query found here:
 * https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/s
 * The purpose is to spit all of the member of the Order Entry role into the
 * cursor. */

OPEN pending_user
FETCH NEXT FROM pending_user INTO @role_user
WHILE @@FETCH_STATUS = 0
        BEGIN
                IF NOT EXISTS (
                        SELECT *
                        FROM sys.database_principals
                        WHERE name = @role_user
                )
                BEGIN
                        DECLARE @creation nvarchar(MAX)
                        SELECT @creation = 'CREATE␣USER␣'
                        + @role_user
                        EXEC sp_executesql @creation
END

DECLARE @drop_user_from_role nvarchar(MAX)
SELECT @drop_user_from_role = 'ALTER␣ROLE␣'
+ @role_name
+ '␣'
+ 'DROP␣MEMBER␣'
+ @role_user

EXEC sp_executesql @drop_user_from_role
PRINT('Dropped␣' + @role_user + '␣from␣' + @role_name)

FETCH NEXT FROM pending_user INTO @role_user
END

CLOSE pending_user
DEALLOCATE pending_user

DECLARE @drop_role nvarchar(MAX)
SELECT @drop_role = 'DROP␣ROLE␣'
+ @role_name
```

```
EXEC sp_executesql @drop_role
PRINT('Dropped␣role␣' + @role_name)
```

Figure 9: Per 4



```
OPEN pending_user
FETCH NEXT FROM pending_user INTO @role_user
WHILE @@FETCH_STATUS = 0
    BEGIN
        IF NOT EXISTS (
            SELECT *
            FROM sys.database_principals
            WHERE name = @role_user
        )
        BEGIN
            DECLARE @creation nvarchar(MAX)
            SELECT @creation = 'CREATE USER '
            + @role_user
            EXEC sp_executesql @creation
END

DECLARE @drop_user_from_role nvarchar(MAX)
SELECT @drop_user_from_role = 'ALTER ROLE '
```

```
Dropped RobertHalliday from OrderEntry
Dropped AdminUser from OrderEntry
Dropped JoelMurach from OrderEntry
Dropped MikeMurach from OrderEntry
Dropped RBrautigan from OrderEntry
Dropped role OrderEntry
```

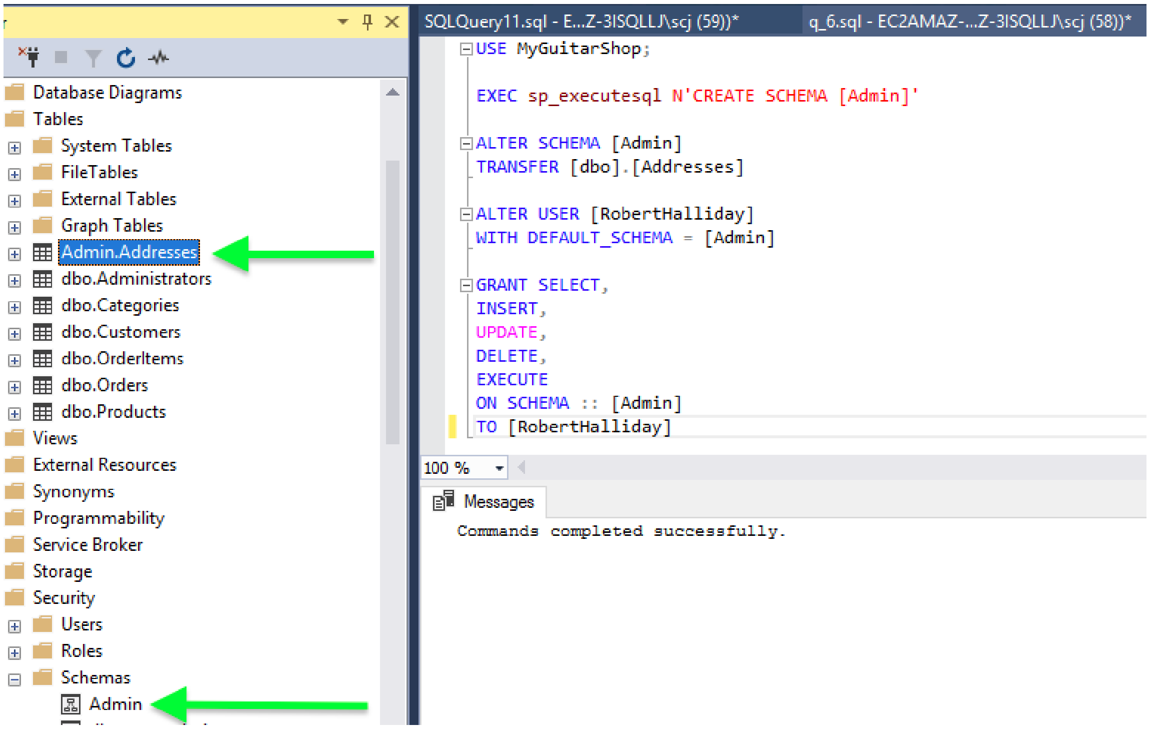
6 Make Admin schema

Also, give Admin schema a table, associate RobertHalliday with it, and give Robert basic operator permissions for the schema. See listing 5 and accompanying figure 10

Listing 5: Per 6

```
USE MyGuitarShop;

EXEC sp_executesql N’CREATE␣SCHEMA␣[Admin]’

ALTER SCHEMA [Admin]
TRANSFER [dbo].[Addresses]

ALTER USER [RobertHalliday]
WITH DEFAULT_SCHEMA = [Admin]

GRANT SELECT,
INSERT,
UPDATE,
DELETE,
EXECUTE
ON SCHEMA :: [Admin]
TO [RobertHalliday]
```

Figure 10: Per 5



```
USE MyGuitarShop;

EXEC sp_executesql N'CREATE SCHEMA [Admin]'

ALTER SCHEMA [Admin]
TRANSFER [dbo].[Addresses]

ALTER USER [RobertHalliday]
WITH DEFAULT_SCHEMA = [Admin]

GRANT SELECT,
INSERT,
UPDATE,
DELETE,
EXECUTE
ON SCHEMA :: [Admin]
TO [RobertHalliday]
```

```
Commands completed successfully.
```