**Veri Tabanı Yönetim Sistemleri**

**-**

**Kütüphane Otomasyonu**

**Hazırlayan**: Muhammed Faruk Tutkus **210707072**

[**Otomasyonun Videolu Anlatımı**](https://drive.google.com/file/d/1CGsuSHJcxZFMcycUOcfFD55pBh6R16nn/view?usp=sharing)

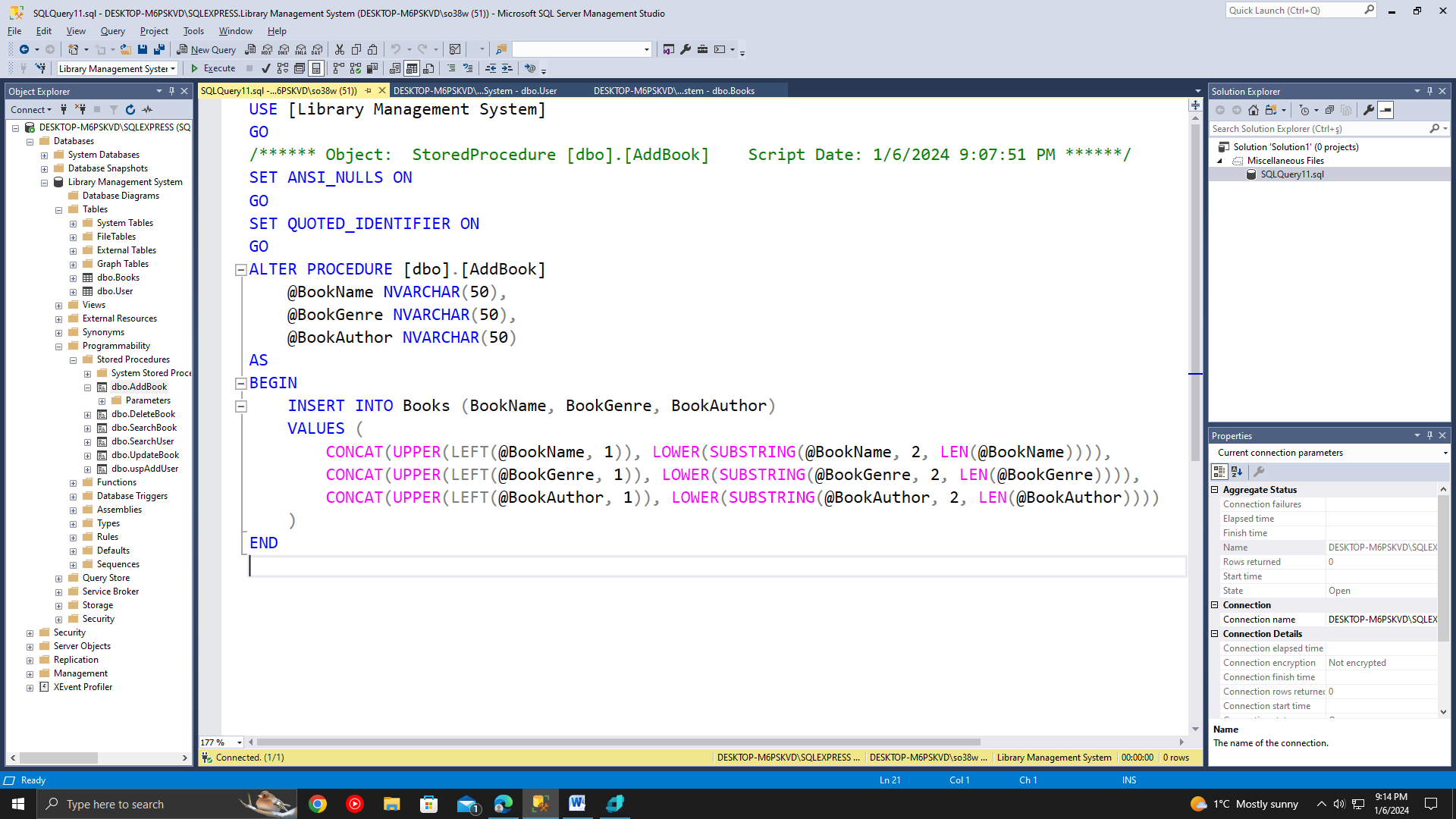
**Database Tanıtımı**

* **Kullanılan Tablolar**

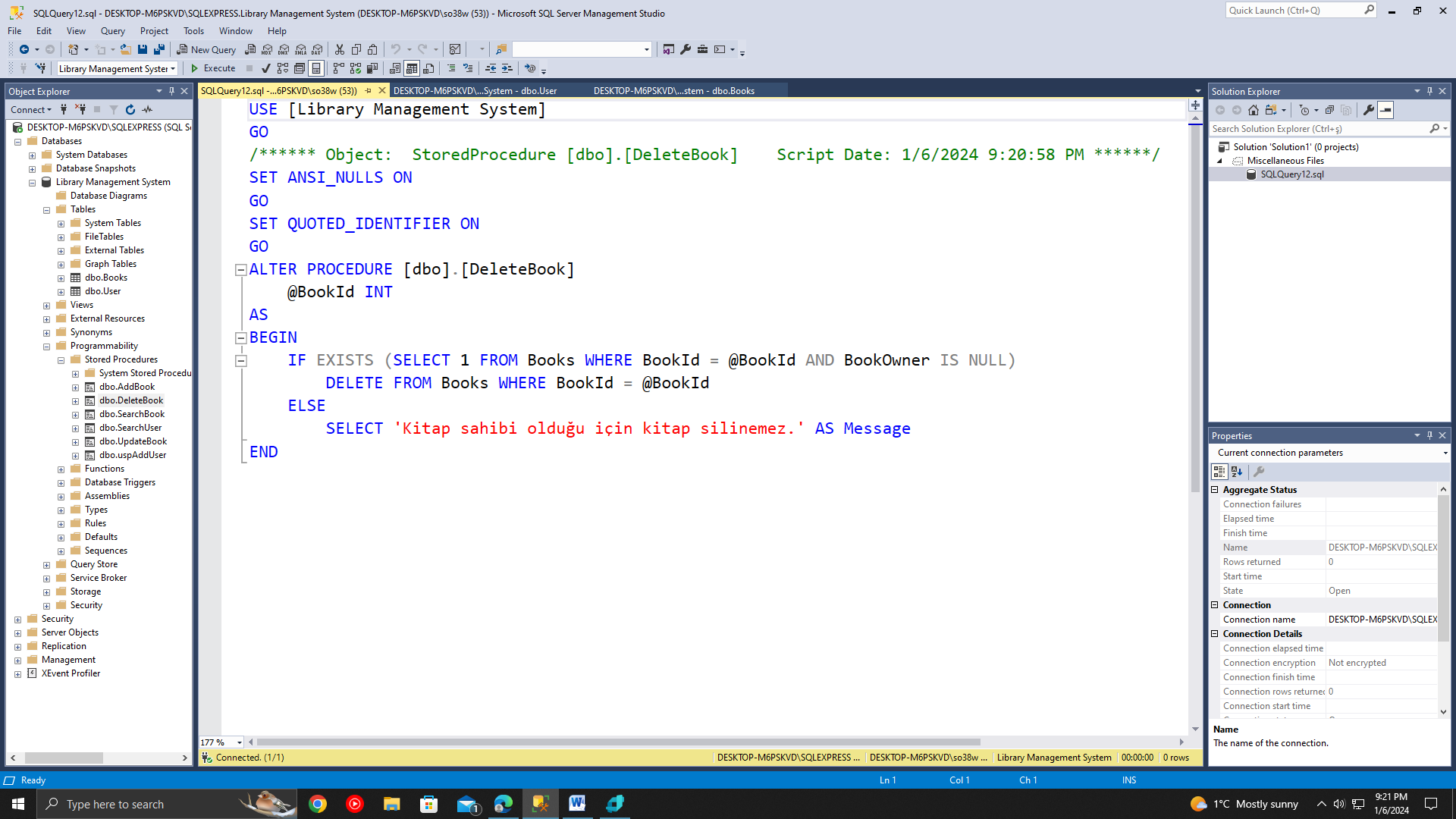
1. **User Tablosu**:
   1. UserID (int)
   2. LoginName (nvarchar(40))
   3. PasswordHash (binary(64))
   4. BookOwner (nvarchar(40))
   5. Role (bit)
2. **Books Tablosu**:
   1. BooksID (int)
   2. BookName (nchar(15))
   3. BookGenre (nchar(15))
   4. BookAuthor (nchar(15))
   5. BookOwner (nchar(10))
   6. RegisterDate (date)
   7. UpdateDate (date)

* **Kullanılan Stored Procedure**

1. **AddBook**

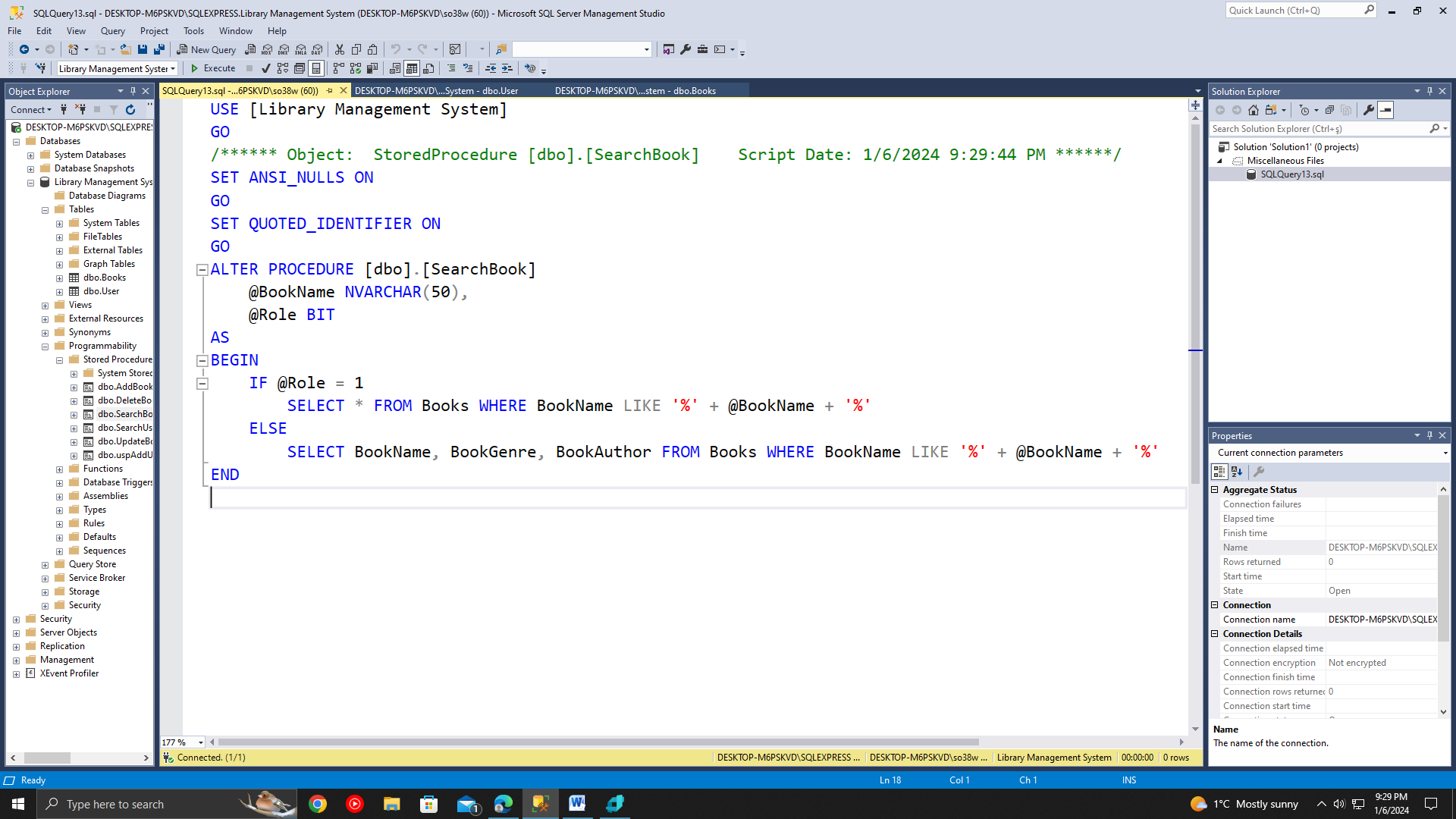


* **ALTER PROCEDURE [dbo].[AddBook]:** Bu satır, AddBook adında bir stored procedure oluşturur veya mevcut birini değiştirir.
* **@BookName NVARCHAR(50), @BookGenre NVARCHAR(50), @BookAuthor NVARCHAR(50):** Bu satır, stored procedure’a giriş parametrelerini tanımlar. Bu durumda, BookName, BookGenre ve BookAuthor adında üç parametre vardır ve her biri NVARCHAR(50) tipindedir, yani 50 karaktere kadar Unicode karakter dizisi saklayabilirler.
* **BEGIN ... END:** Bu satırlar arasındaki kod bloğu, stored procedure’ın yapacağı işlemleri tanımlar.
* **INSERT INTO Books (BookName, BookGenre, BookAuthor):** Bu satır, Books tablosuna yeni bir satır ekler. Eklenen satırın BookName, BookGenre ve BookAuthor sütunlarına değer atanacaktır.
* **VALUES (...):** Bu satır, eklenen satırın hangi değerlere sahip olacağını belirtir. Burada, her bir değer **CONCAT** ve **UPPER**, **LOWER**, **LEFT**, **SUBSTRING**, **LEN** gibi string fonksiyonları kullanılarak oluşturulur. Bu fonksiyonlar, kitap adı, türü ve yazarının ilk harfini büyük, geri kalanını küçük harfle yazmayı sağlar.

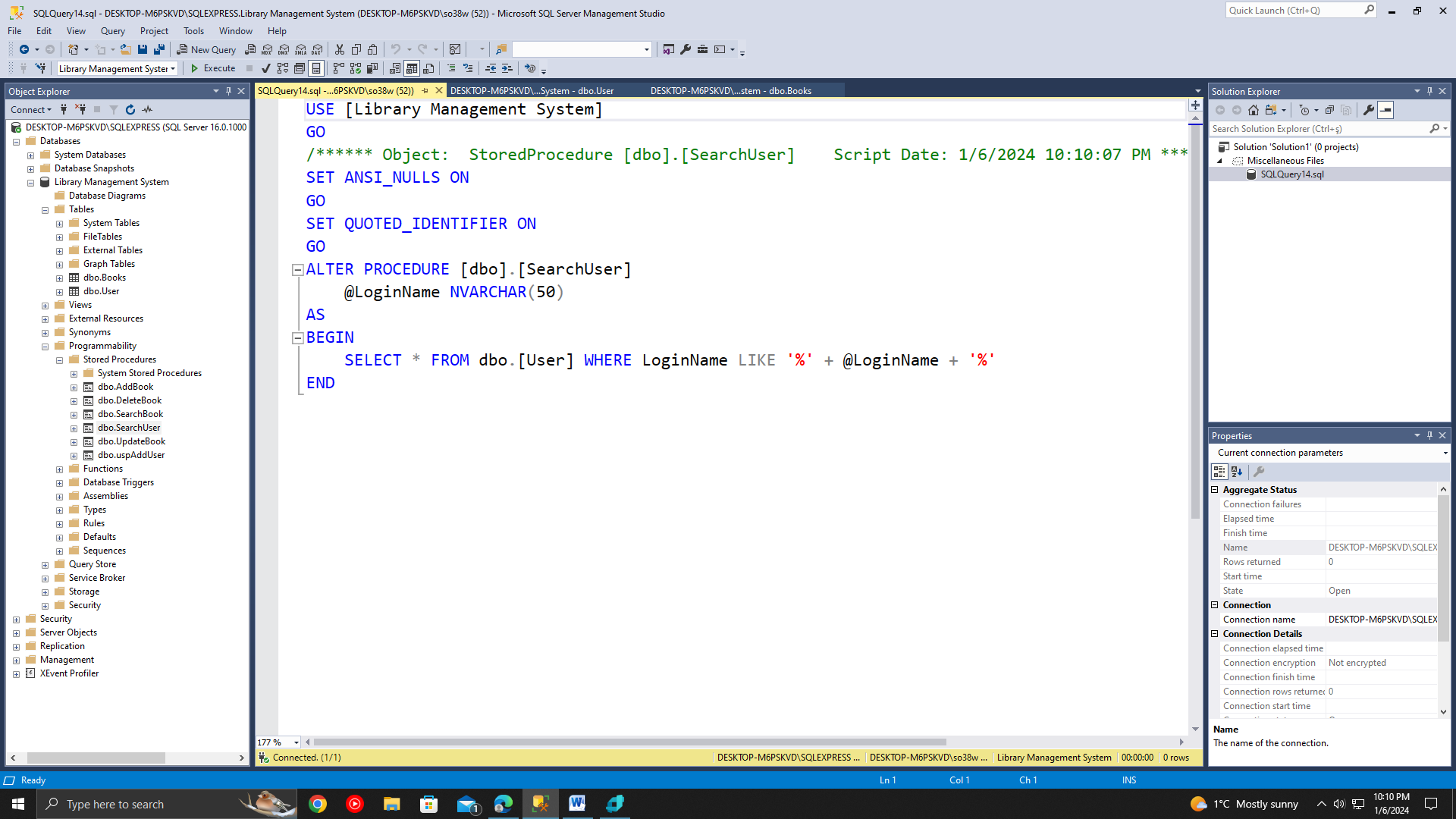
1. **DeleteBook** 

* **ALTER PROCEDURE [dbo].[DeleteBook] @BookId INT:** Bu satır, DeleteBook adında bir stored procedure oluşturur veya mevcut birini değiştirir. Bu stored procedure, bir BookId parametresi alır.
* **IF EXISTS (SELECT 1 FROM Books WHERE BookId = @BookId AND BookOwner IS NULL):** Bu satır, belirli bir BookId’ye sahip ve BookOwner’ı NULL olan bir kitabın var olup olmadığını kontrol eder.
* **DELETE FROM Books WHERE BookId = @BookId:** Eğer belirli bir BookId’ye sahip ve BookOwner’ı NULL olan bir kitap varsa, bu kitabı Books tablosundan siler.

**SELECT 'Kitap sahibi olduğu için kitap silinemez.' AS Message: Eğer belirli bir BookId’ye sahip bir kitap var ancak bu kitabın bir sahibi varsa (yani BookOwner NULL değilse), bu mesajı döndürür.**

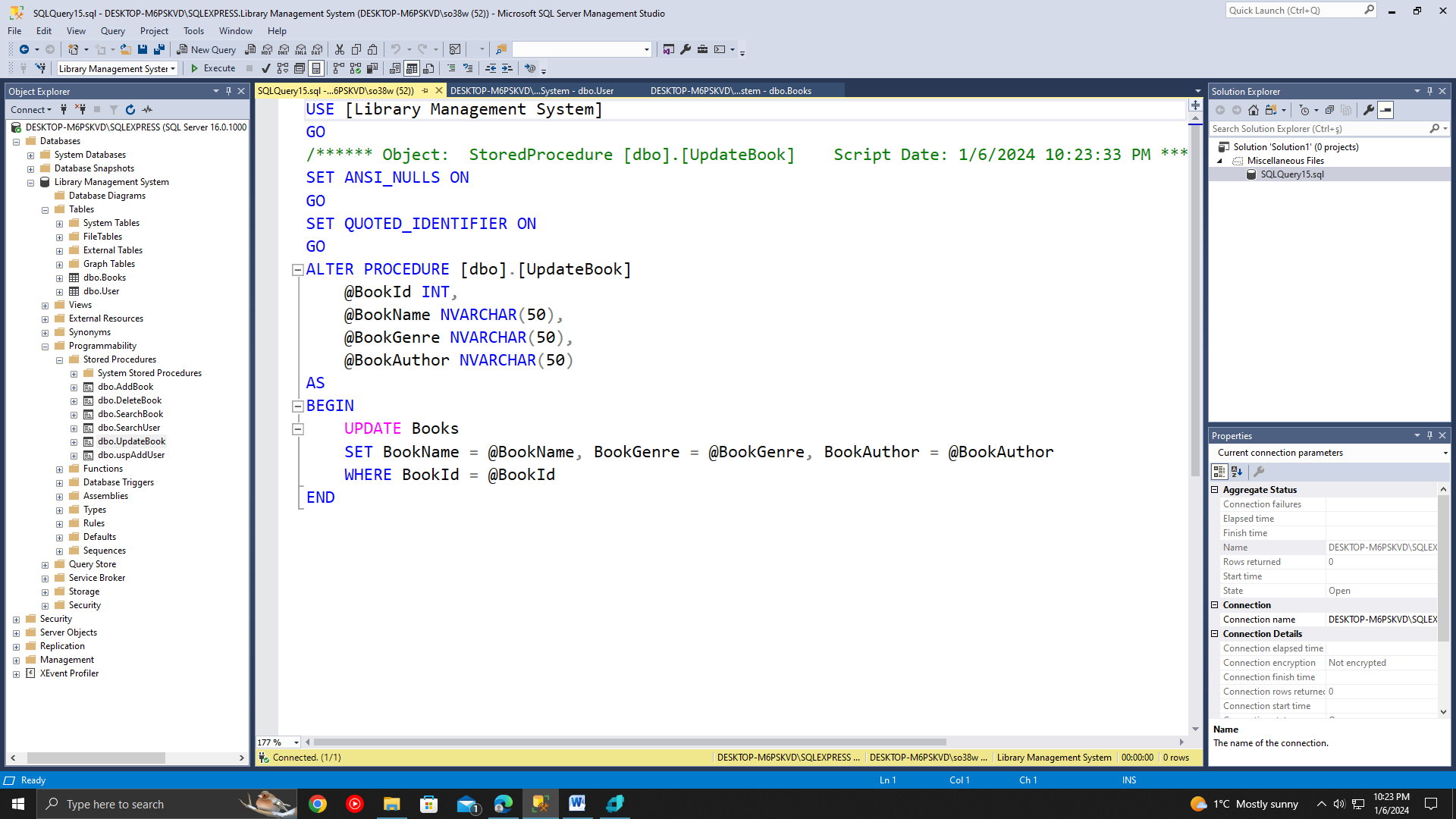
1. **SearchBook** 

* **ALTER PROCEDURE [dbo].[SearchBook] @BookName NVARCHAR(50), @Role BIT:** Bu satır, SearchBook adında bir stored procedure oluşturur veya mevcut birini değiştirir. Bu stored procedure, bir BookName ve bir Role parametresi alır.
* **IF @Role = 1 ... ELSE ... END:** Bu satırlar arasındaki kod bloğu, stored procedure’ın yapacağı işlemleri tanımlar. Eğer Role parametresi 1 ise, BookName’e sahip tüm kitapların tüm bilgilerini döndürür. Eğer Role parametresi 1 değilse, yalnızca BookName, BookGenre ve BookAuthor bilgilerini döndürür.

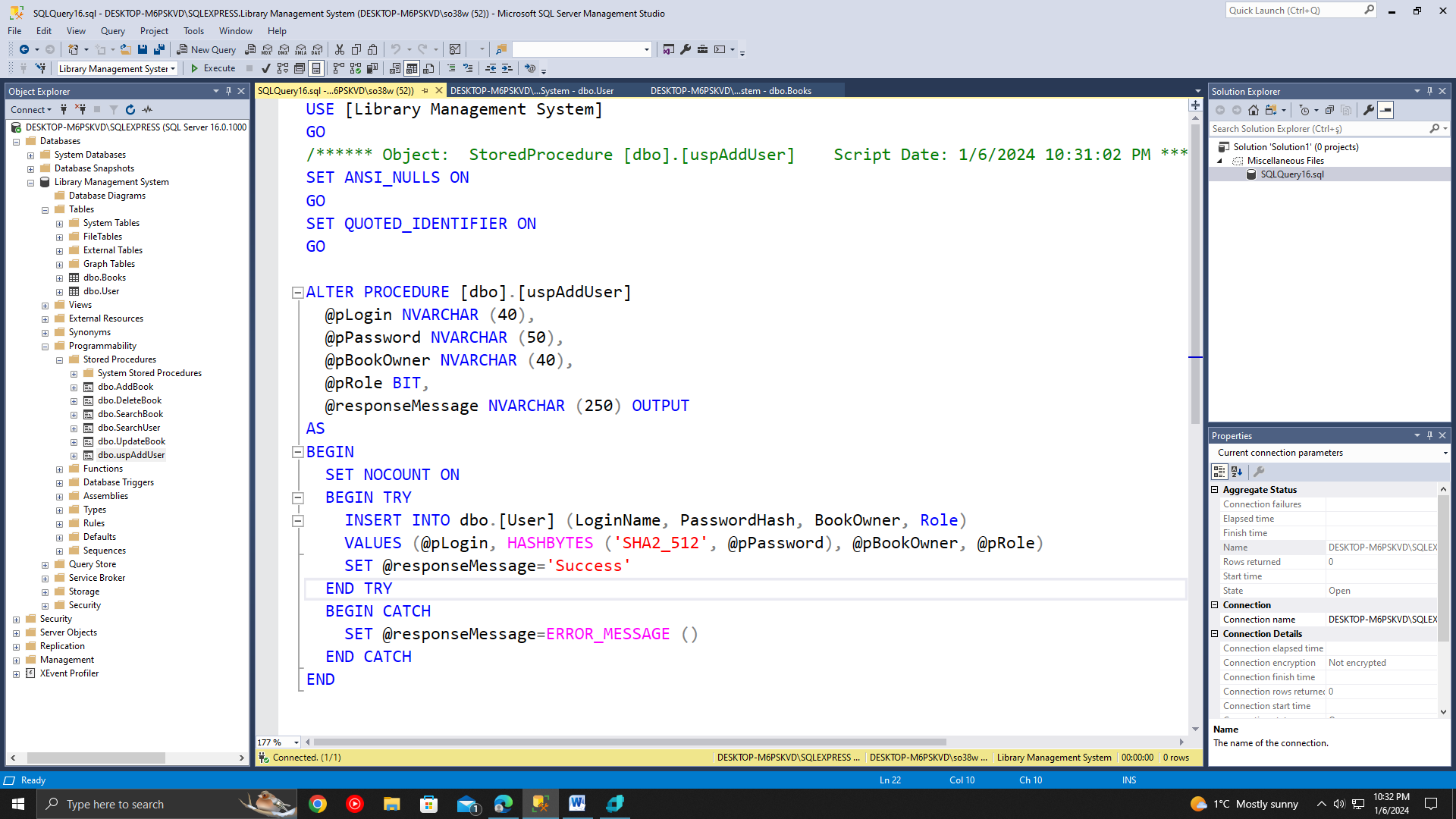
1. **SearchUser**

* **ALTER PROCEDURE [dbo].[SearchUser] @LoginName NVARCHAR(50):** Bu satır, SearchUser adında bir stored procedure oluşturur veya mevcut birini değiştirir. Bu stored procedure, bir LoginName parametresi alır.
* **SELECT \* FROM dbo.[User] WHERE LoginName LIKE '%' + @LoginName + '%':** Bu satır, LoginName’e sahip tüm kullanıcıların tüm bilgilerini döndürür.

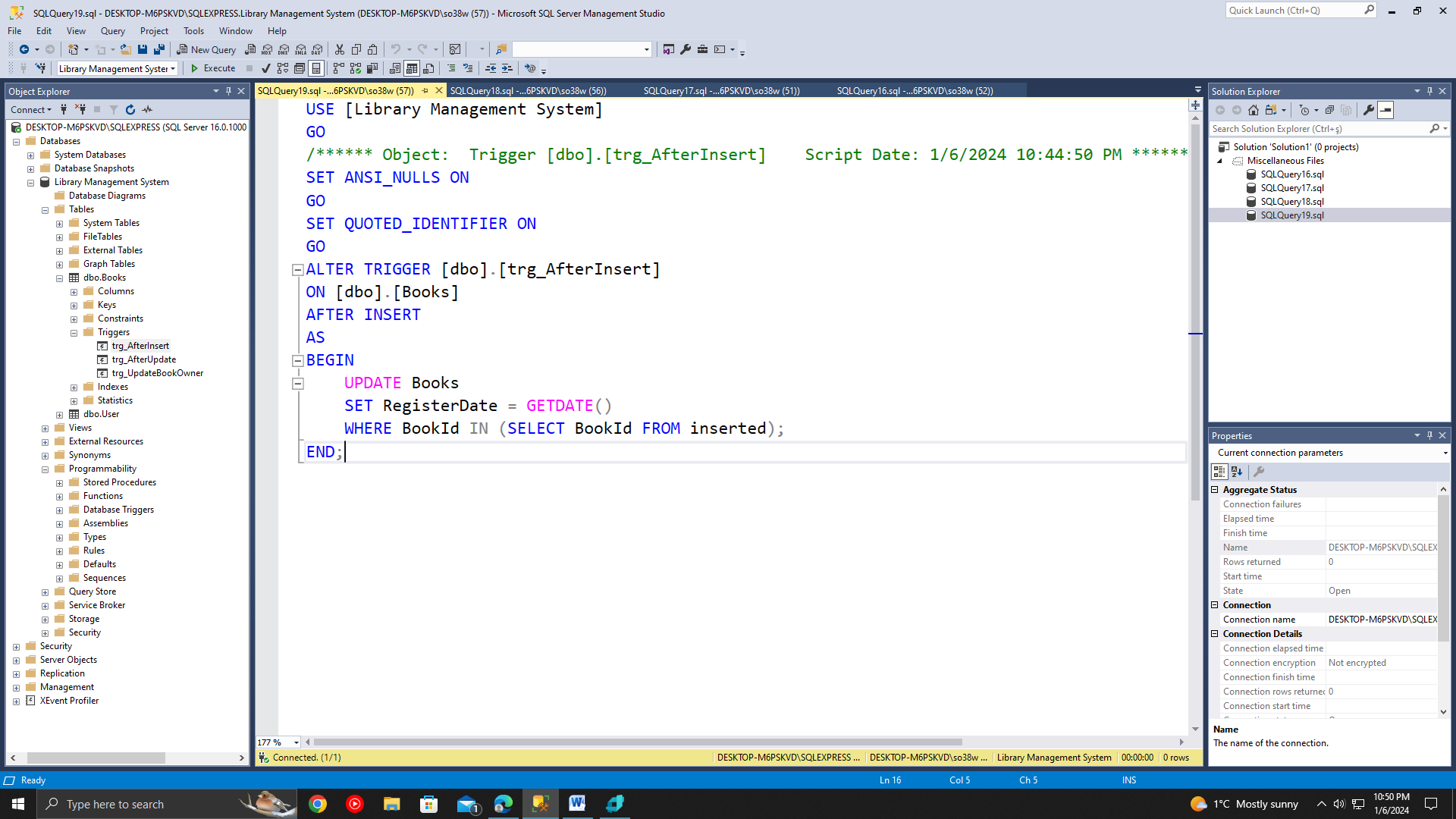
1. **UpdateBook**



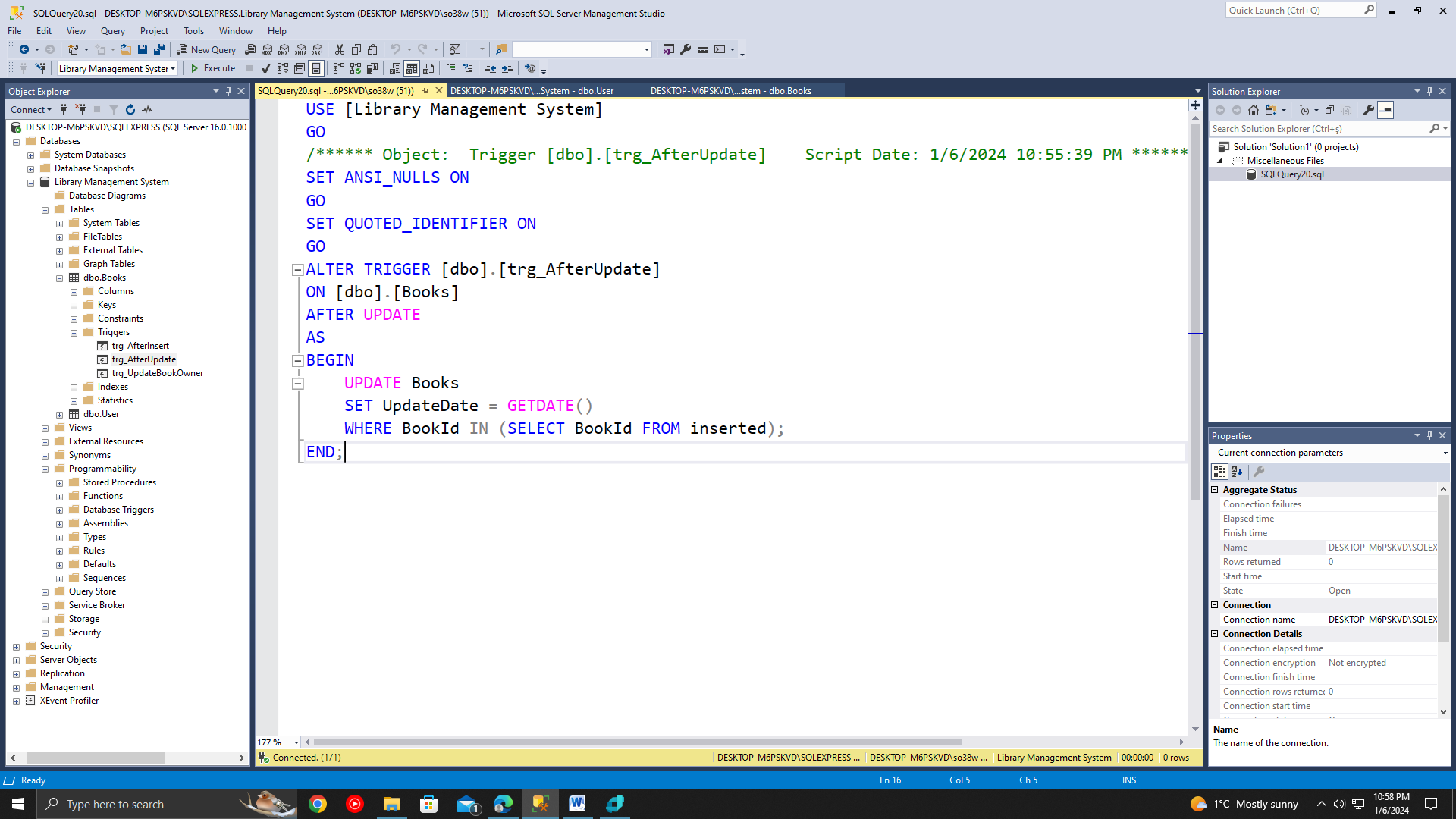
* **ALTER PROCEDURE [dbo].[UpdateBook] @BookId INT, @BookName NVARCHAR(50), @BookGenre NVARCHAR(50), @BookAuthor NVARCHAR(50):** Bu satır, UpdateBook adında bir stored procedure oluşturur veya mevcut birini değiştirir. Bu stored procedure, bir BookId, BookName, BookGenre ve BookAuthor parametresi alır.
* **UPDATE Books SET BookName = @BookName, BookGenre = @BookGenre, BookAuthor = @BookAuthor WHERE BookId = @BookId:** Bu satır, belirli bir BookId’ye sahip kitabın BookName, BookGenre ve BookAuthor bilgilerini günceller.

1. **uspAddUser** 

* **ALTER PROCEDURE [dbo].[uspAddUser] @pLogin NVARCHAR (40), @pPassword NVARCHAR (50), @pBookOwner NVARCHAR (40), @pRole BIT, @responseMessage NVARCHAR (250) OUTPUT:** Bu satır, uspAddUser adında bir stored procedure oluşturur veya mevcut birini değiştirir. Bu stored procedure, bir LoginName, Password, BookOwner, Role ve bir çıktı parametresi olan responseMessage parametresi alır.
* **SET NOCOUNT ON:** Bu satır, SQL Server’ın her bir işlem için bir satır sayısı döndürmesini engeller.
* **BEGIN TRY ... END TRY:** Bu blok, stored procedure’ın ana işlemlerini içerir. Eğer bu blokta bir hata oluşursa, kontrol CATCH blokuna geçer.
* **INSERT INTO dbo.[User] (LoginName, PasswordHash, BookOwner, Role) VALUES (@pLogin, HASHBYTES ('SHA2\_512', @pPassword), @pBookOwner, @pRole):** Bu satır, User tablosuna yeni bir kullanıcı ekler. Kullanıcının şifresi, HASHBYTES fonksiyonu kullanılarak hashlenir.
* **SET @responseMessage='Success':** Eğer INSERT işlemi başarılı olursa, responseMessage parametresine 'Success' değeri atanır.
* **BEGIN CATCH ... END CATCH:** Bu blok, TRY blokunda bir hata oluşursa çalışır. Bu blokta, responseMessage parametresine hata mesajı atanır.
* **Kullanılan Triggerler**

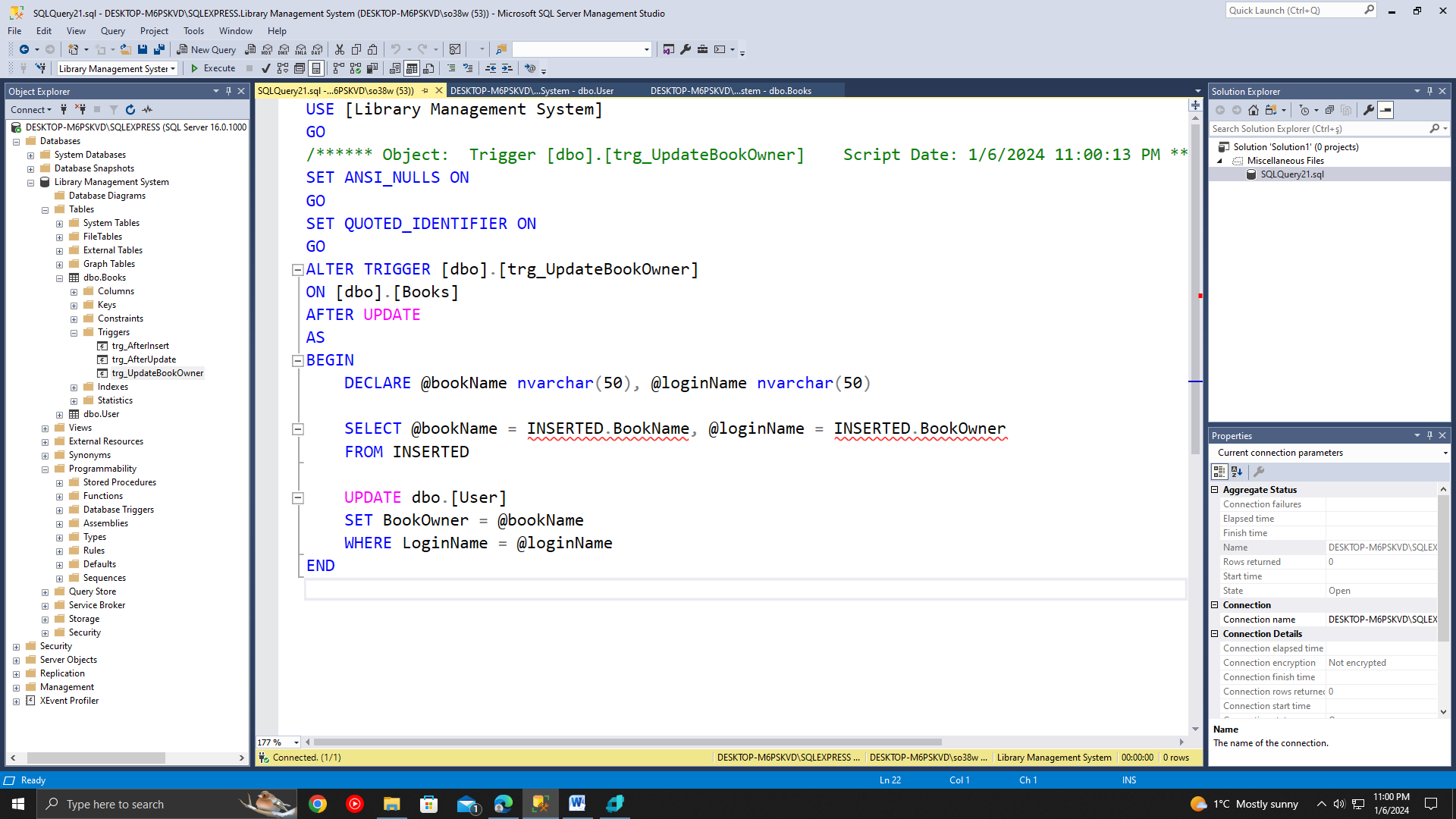
1. **trg\_AfterInsert(Books)**

* **ALTER TRIGGER [dbo].[trg\_AfterInsert] ON [dbo].[Books] AFTER INSERT:** Bu satır, trg\_AfterInsert adında bir tetikleyici oluşturur veya mevcut birini değiştirir. Bu tetikleyici, Books tablosuna yeni bir satır eklendikten sonra çalışır.
* **UPDATE Books SET RegisterDate = GETDATE() WHERE BookId IN (SELECT BookId FROM inserted):** Bu satır, yeni eklenen kitabın RegisterDate sütununu günceller. **GETDATE()** fonksiyonu, mevcut tarihi ve saati döndürür. inserted tablosu, bir INSERT işlemi sırasında eklenen satırları içerir.

1. **trg\_AfterUpdate(Books)**

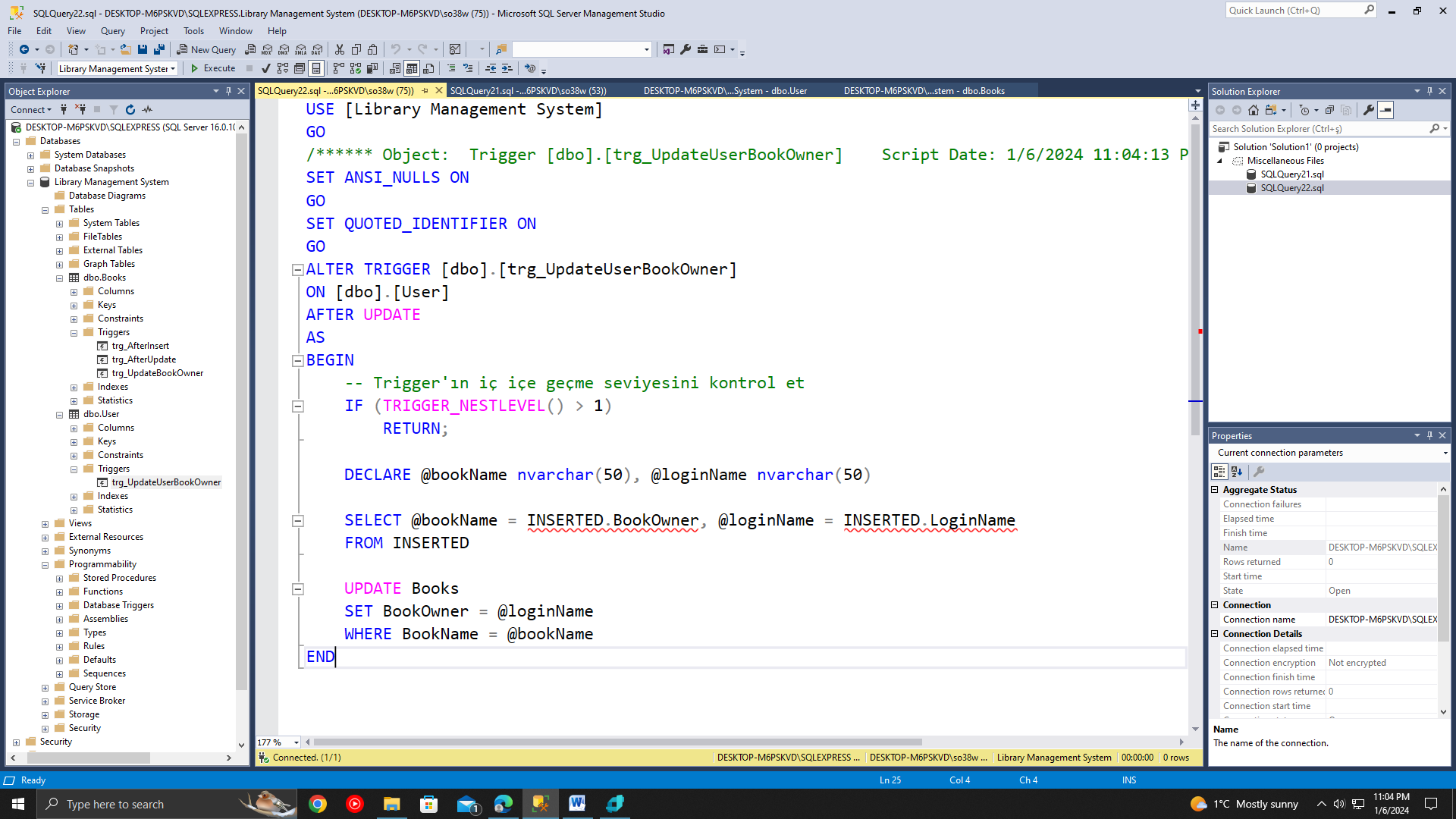
* **ALTER TRIGGER [dbo].[trg\_AfterUpdate] ON [dbo].[Books] AFTER UPDATE:** Bu satır, trg\_AfterUpdate adında bir tetikleyici oluşturur veya mevcut birini değiştirir. Bu tetikleyici, Books tablosunda bir satır güncellendikten sonra çalışır.
* **UPDATE Books SET UpdateDate = GETDATE() WHERE BookId IN (SELECT BookId FROM inserted):** Bu satır, güncellenen kitabın UpdateDate sütununu günceller. GETDATE() fonksiyonu, mevcut tarihi ve saati döndürür. inserted tablosu, bir UPDATE işlemi sırasında güncellenen satırları içerir.

1. **trg\_UpdateBookOwner(Books)**



* **ALTER TRIGGER [dbo].[trg\_UpdateBookOwner] ON [dbo].[Books] AFTER UPDATE:** Bu satır, trg\_UpdateBookOwner adında bir tetikleyici oluşturur veya mevcut birini değiştirir. Bu tetikleyici, Books tablosunda bir satır güncellendikten sonra çalışır.
* **DECLARE @bookName nvarchar(50), @loginName nvarchar(50):** Bu satır, @bookName ve @loginName adında iki değişken tanımlar.
* **SELECT @bookName = INSERTED.BookName, @loginName = INSERTED.BookOwner FROM INSERTED:** Bu satır, güncellenen kitabın BookName ve BookOwner bilgilerini alır ve bu bilgileri @bookName ve @loginName değişkenlerine atar. inserted tablosu, bir UPDATE işlemi sırasında güncellenen satırları içerir.
* **UPDATE dbo.[User] SET BookOwner = @bookName WHERE LoginName = @loginName:** Bu satır, User tablosunda LoginName’i @loginName olan kullanıcının BookOwner bilgisini @bookName ile günceller.

1. **trg\_UpdateUserBookOwner (Books)**



* **ALTER TRIGGER [dbo].[trg\_UpdateUserBookOwner] ON [dbo].[User] AFTER UPDATE:** Bu satır, trg\_UpdateUserBookOwner adında bir tetikleyici oluşturur veya mevcut birini değiştirir. Bu tetikleyici, User tablosunda bir satır güncellendikten sonra çalışır.
* **IF (TRIGGER\_NESTLEVEL() > 1) RETURN;:** Bu satır, tetikleyicinin iç içe geçme seviyesini kontrol eder. Eğer tetikleyici başka bir tetikleyici tarafından tetiklendi ise (yani iç içe geçme seviyesi 1’den büyükse), tetikleyiciyi sonlandırır.
* **DECLARE @bookName nvarchar(50), @loginName nvarchar(50):** Bu satır, @bookName ve @loginName adında iki değişken tanımlar.
* **SELECT @bookName = INSERTED.BookOwner, @loginName = INSERTED.LoginName FROM INSERTED:** Bu satır, güncellenen kullanıcının BookOwner ve LoginName bilgilerini alır ve bu bilgileri @bookName ve @loginName değişkenlerine atar. inserted tablosu, bir UPDATE işlemi sırasında güncellenen satırları içerir.
* **UPDATE Books SET BookOwner = @loginName WHERE BookName = @bookName:** Bu satır, BookName’i @bookName olan kitabın BookOwner bilgisini @loginName ile günceller.
* **Kullanılan Fonksiyonlar**
* **CONCAT:** İki veya daha fazla dizeyi birleştirir.
* **UPPER:** Bir dizenin tüm karakterlerini büyük harfe çevirir.
* **LOWER:** Bir dizenin tüm karakterlerini küçük harfe çevirir.
* **LEFT:** Bir dizenin solundan belirli sayıda karakteri döndürür.
* **SUBSTRING:** Bir dizenin belirli bir bölümünü döndürür.
* **LEN:** Bir dizenin uzunluğunu döndürür.
* **HASHBYTES:** Belirli bir algoritmayı kullanarak bir dizenin hash değerini döndürür.
* **GETDATE:** Mevcut tarih ve saati döndürür.
* **TRIGGER\_NESTLEVEL:** Tetikleyicinin iç içe geçme seviyesini döndürür.
* **ERROR\_MESSAGE:** Son hata mesajını döndürür.
* **ERD(Entity Relationship Diagram)**

