ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM4522 PROJE RAPORU

Veritabanı Güvenliği ve Erişim Kontrolü

https://github.com/JineenRihawi

Umut Akylbek kyzy 21291003
Cenin Rihavi 21291007

Öğretim Görevlisi Enver Bağcı

23.04.2025

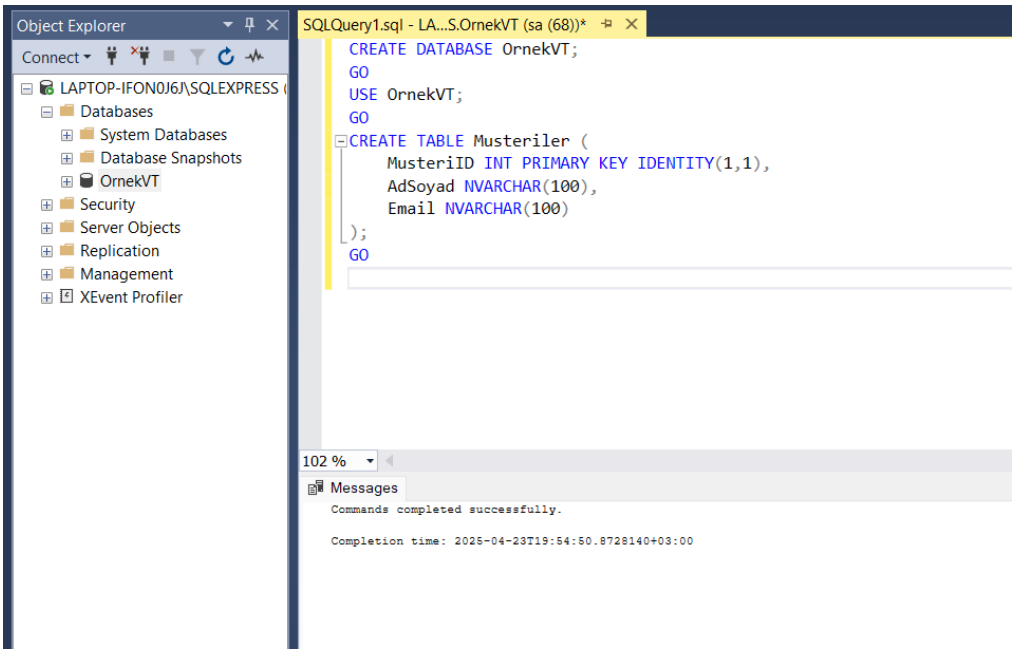
İçindekiler

1. Giriş

Bu projede, Microsoft SQL Server kullanılarak bir veritabanı sisteminin güvenliği uygulamalı olarak ele alınmıştır. Güvenli veritabanı oluşturma, erişim yetkilendirmesi, veri şifreleme, SQL Injection'a karşı koruma ve kullanıcı aktivitelerinin izlenmesi gibi birçok güvenlik başlığı altında işlemler gerçekleştirilmiştir. Ayrıca ekran görüntüleri ve örnek komutlar ile uygulamalar desteklenmiştir.

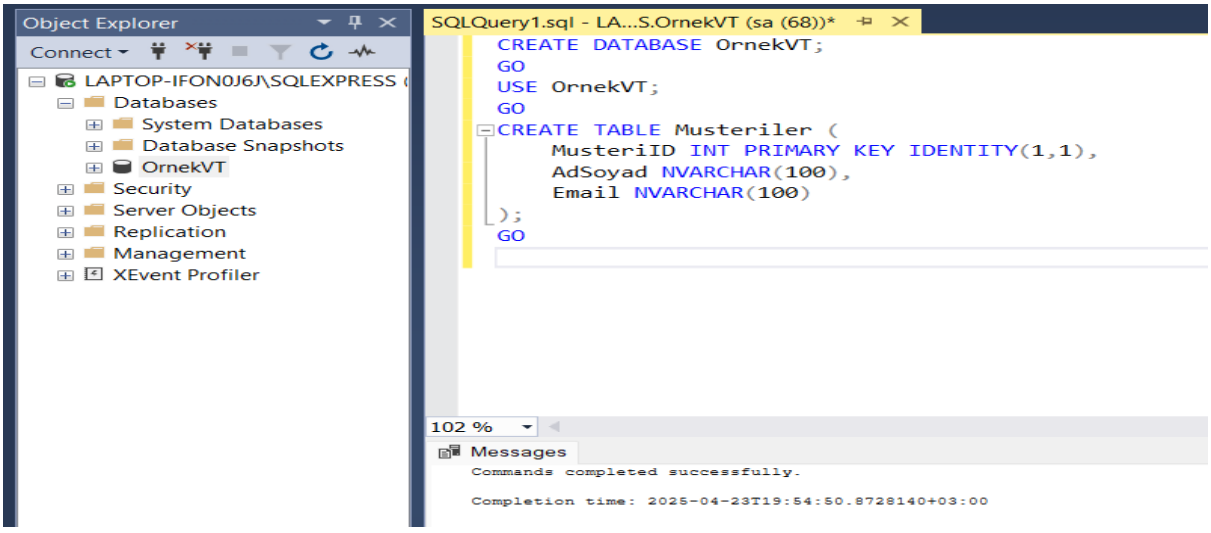
2. ÖRNEK VERİTABANI OLUŞTURMA

2.1. Veri tabanı Oluşturma Komutları



Şekil 1 Veri Tabanı Oluşturma Komutları

2.2. Örnek Tablo Yapıları



Şekil 2Örnek Tablo Yapıları

3. Kimlik Doğrulama ve Yetki Yönetimi

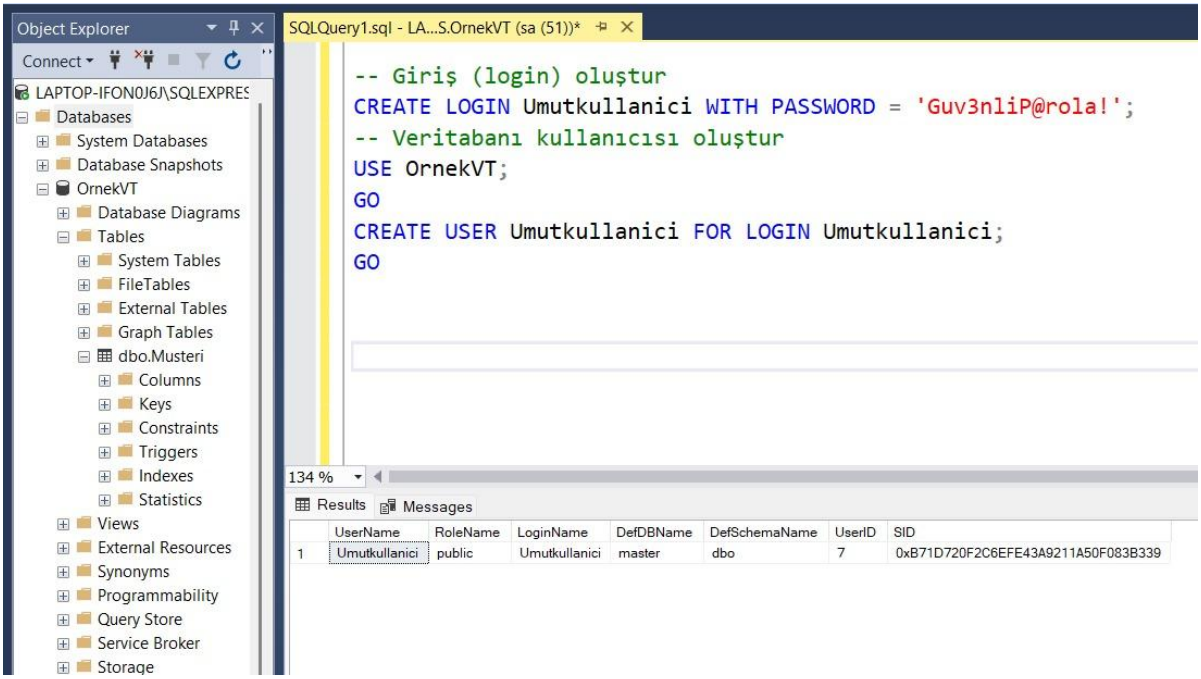
3.1. SQL Server ve Windows Authentication

- **SQL Server Authentication** ve **Windows Authentication** yöntemleriyle kullanıcı oturum açma sistemlerinin karşılaştırılması ve kullanılması.
- Yeni kullanıcı hesapları oluşturulacak (CREATE LOGIN, CREATE USER) ve kullanıcı yetkilendirmesi yapılacak (GRANT, DENY, REVOKE komutları).
- Role-based access control (RBAC) uygulanacak; kullanıcılar özel rollerle sınıflandırılacak.

3.2. SQL Server Authentication Kullanıcısı Oluşturma

SQL Server Authentication kullanabilmek için sunucunun mixed mode (SQL Server and Windows Authentication mode) olarak yapılandırılmış olması gerekir. Bu ayar SQL Server Management Studio (SSMS) üzerinden "Server Properties > Security" kısmından yapılır.

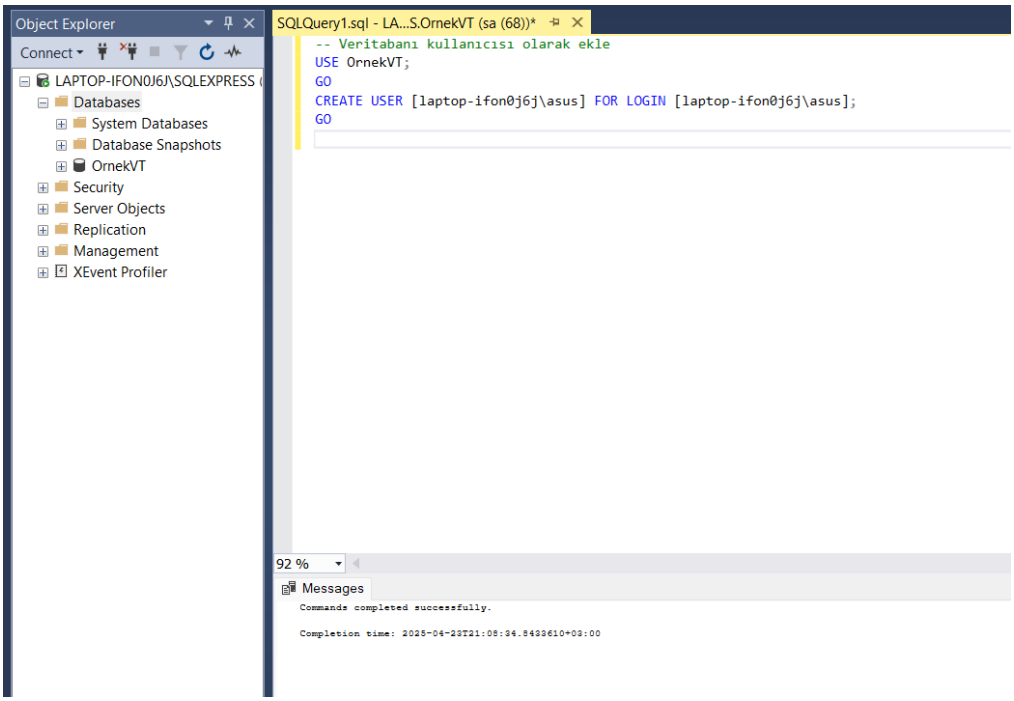
Kullanıcı oluşturmak için aşağıdaki komut kullanılır:



Şekil 3 Veri Tabanı Kullanıcısı Oluşturma (SQL Server Authentication)

3.3. Windows Authentication Kullanıcısı Oluşturma

Windows Authentication yöntemi, kullanıcıların mevcut Windows oturum bilgileriyle SQL Server'a bağlanmasına olanak tanır. Bu yöntem, merkezi kullanıcı yönetimi ve daha güvenli kimlik doğrulama sağladığı için kurumsal ortamlarda yaygın olarak tercih edilir.



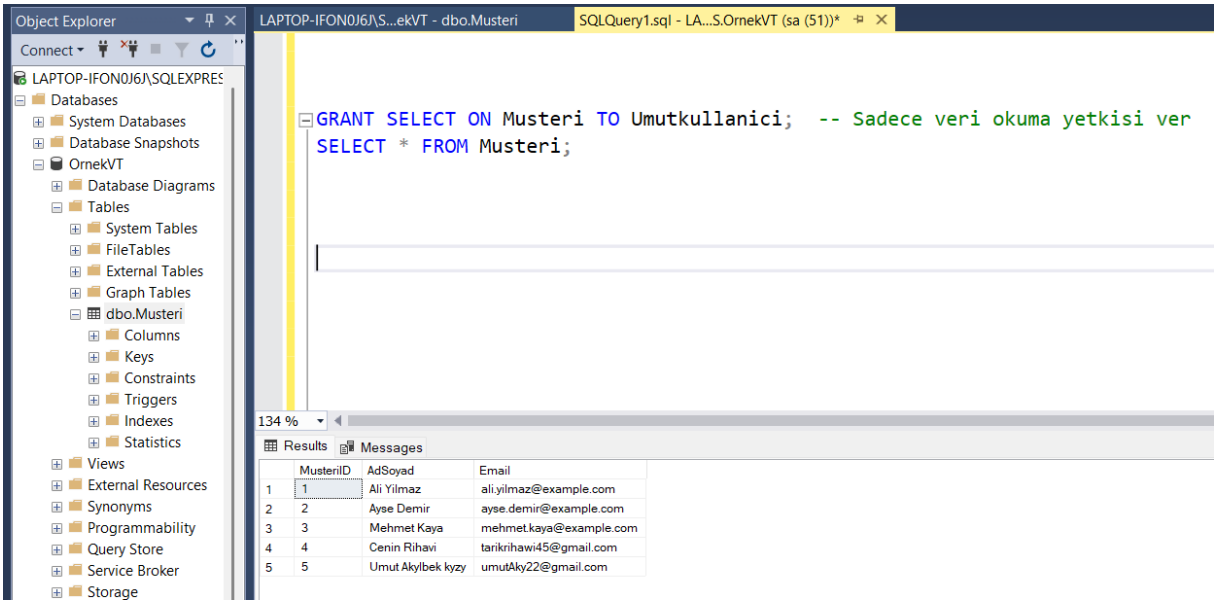
Şekil 4Veri Tabanı Kullanıcısı Oluşturma (Windows Authentication)

3.4. Yetki Verme (GRANT, DENY, REVOKE)

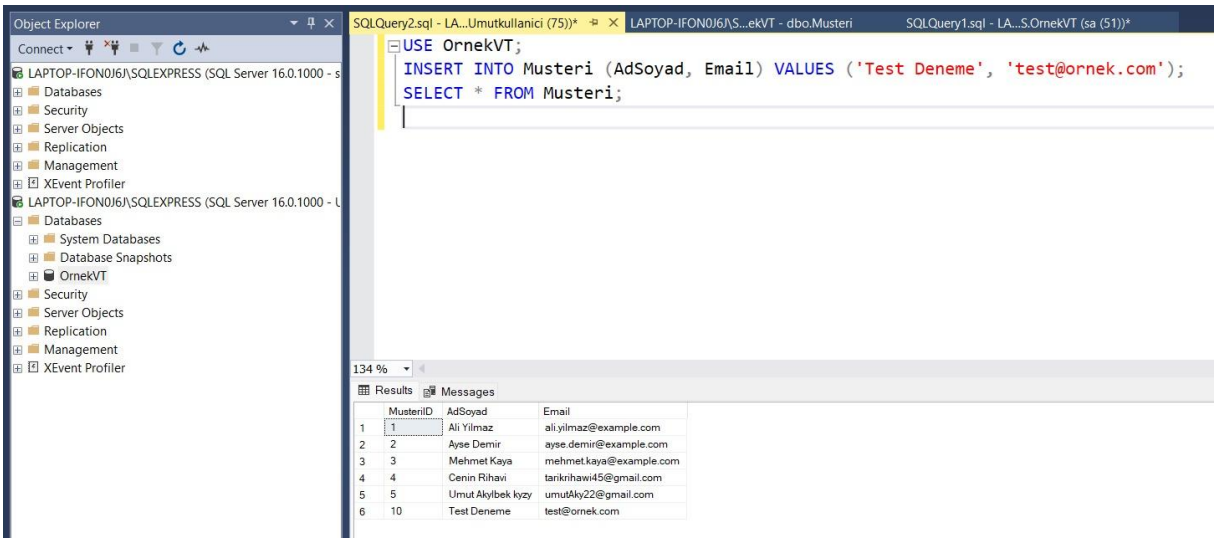
Projede, kullanıcıların sadece yetkili oldukları işlemleri yapabilmesi için **GRANT**, **DENY** ve **REVOKE** komutlarıyla yetki yönetimi sağlanmış; ayrıca kullanıcılar, rollere atanarak toplu yetkilendirme yapılmıştır.

3.4.1. GRANT Komutu:

Belirli bir kullanıcıya veya role bir izin (örneğin SELECT, INSERT) vermek için kullanılır.



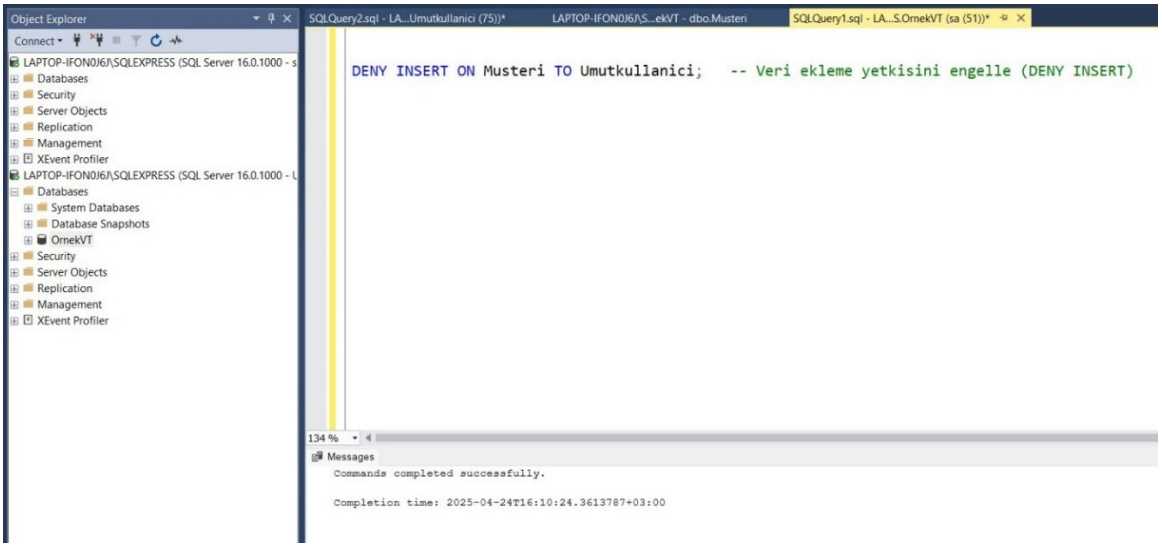
Şekil 5 GRANT Komutu



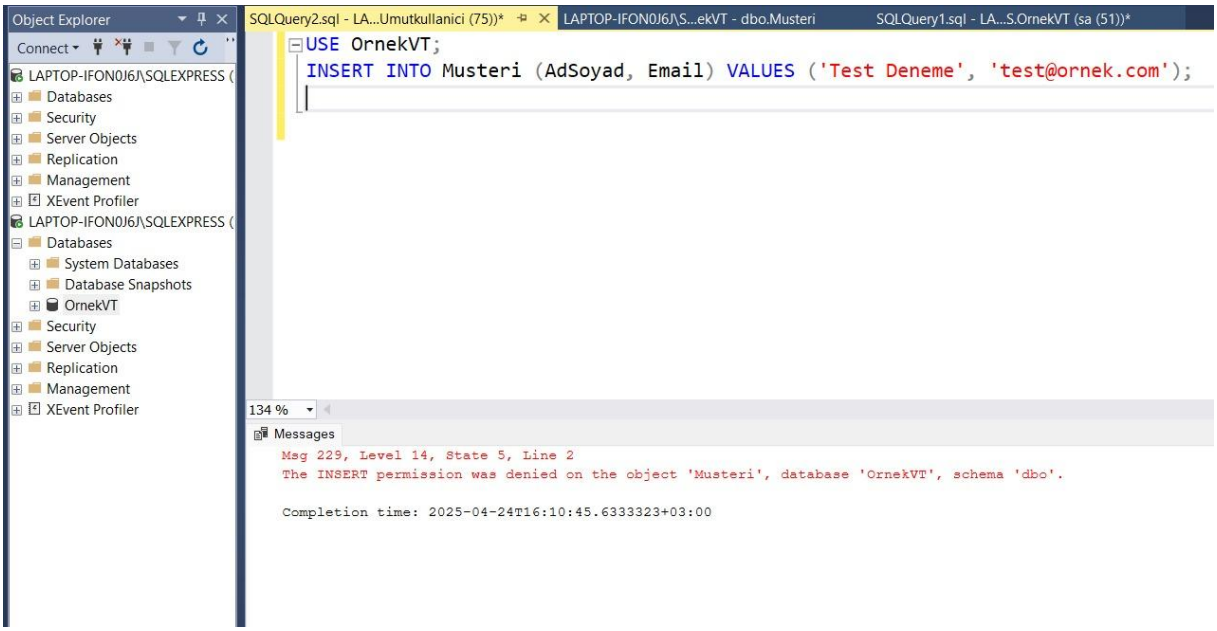
Şekil 6 GRANT Komutunu Test Etme

3.4.2. DENY Komutu:

Belirli bir kullanıcıya veya role bir izni açıkça reddetmek için kullanılır. GRANT'tan üstündür.



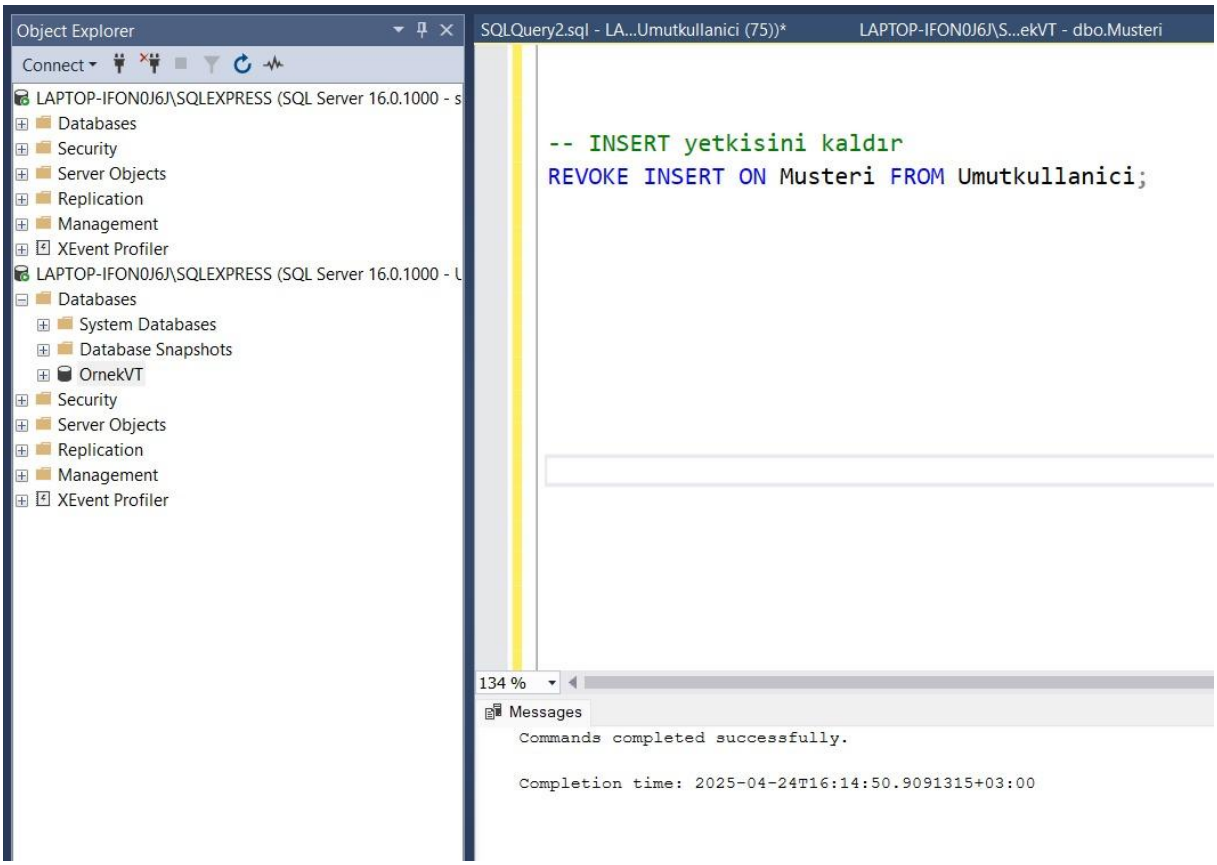
Şekil 7 DENY Komutu



Şekil 8 DENY Komutunu Test Etme

3.4.3. REVOKE Komutu:

Daha önce verilmiş olan bir izni veya reddedilmiş bir izni **geri almak** için kullanılır.

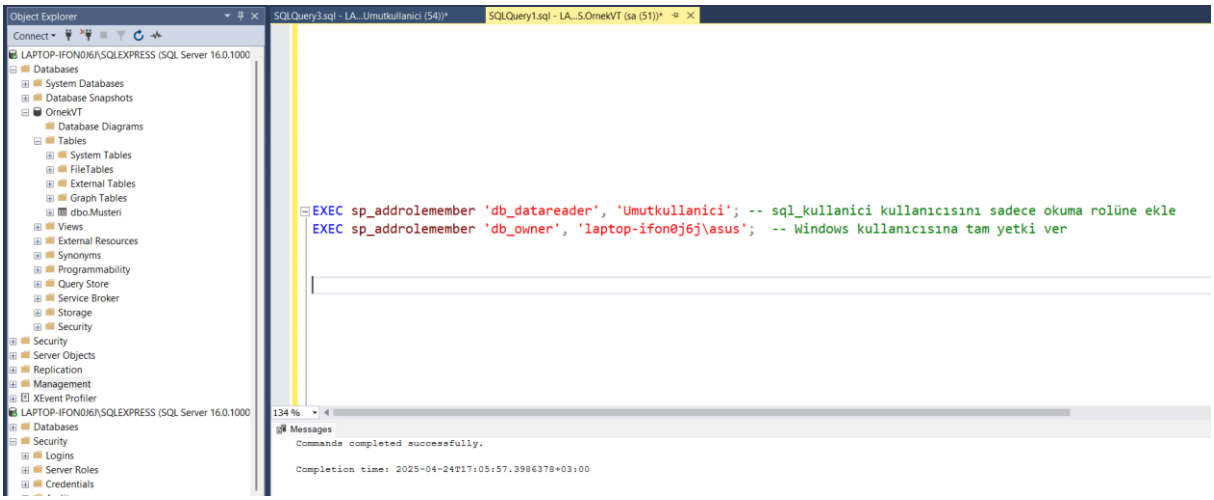


Şekil 9 REVOKE Komutu

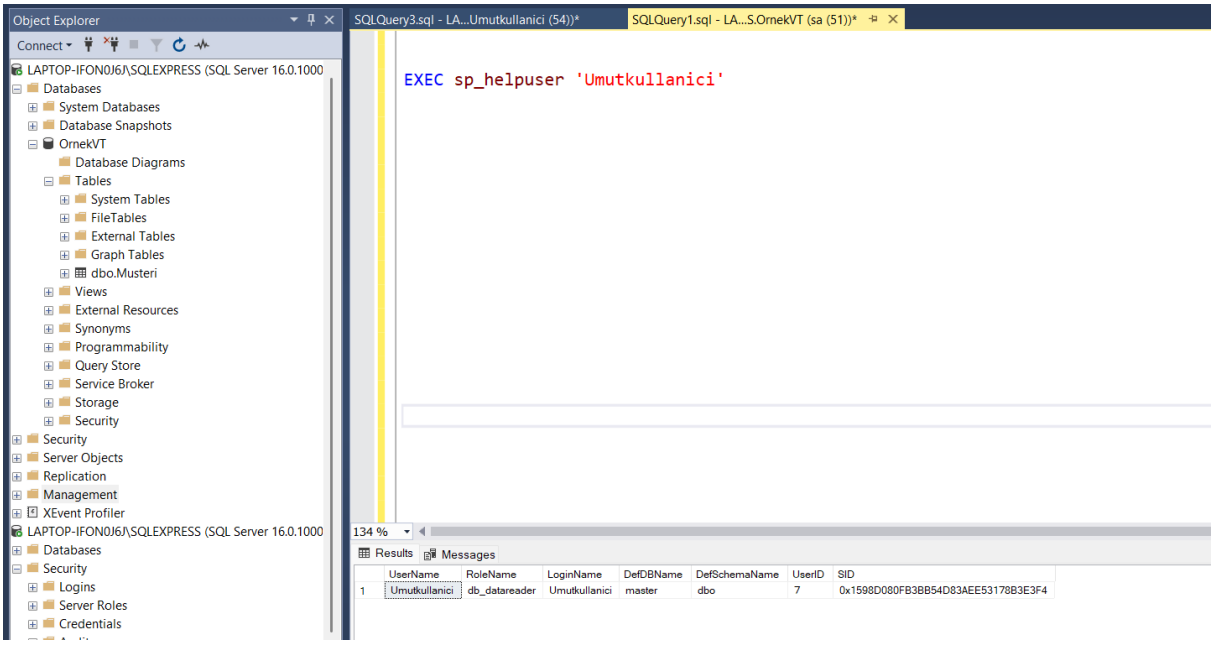
3.5. Role-Based Access Control (RBAC)

SQL Server'daki yerleşik rollerden bazıları:

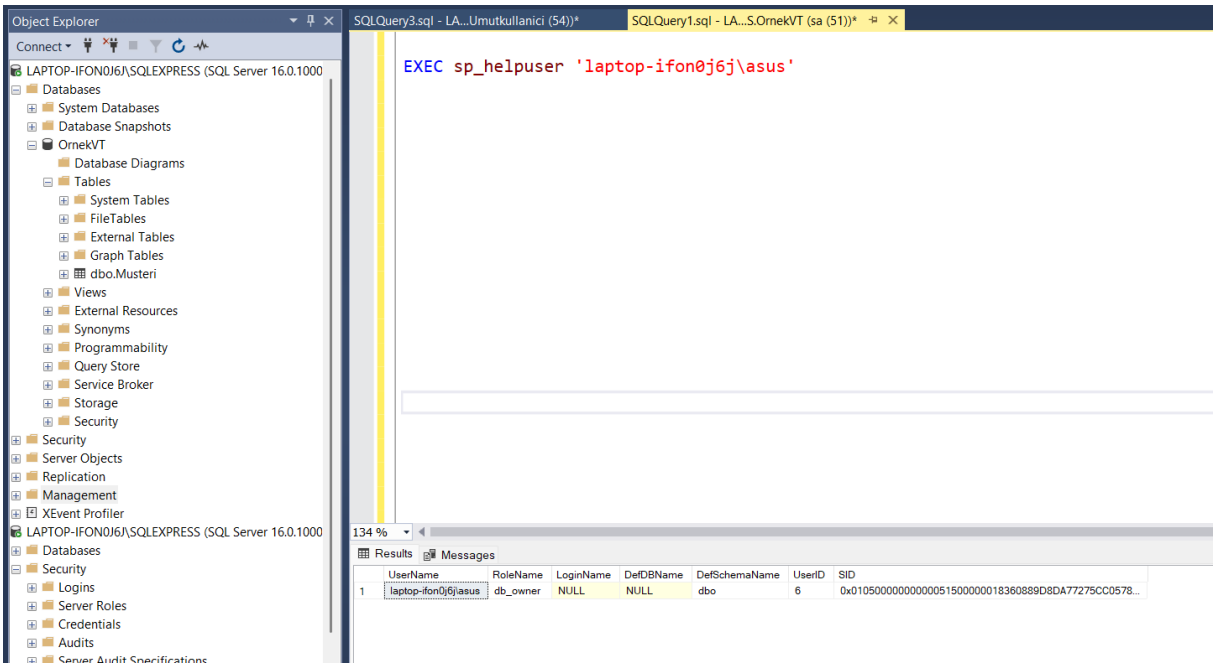
- db_datareader: tüm tabloları okuma yetkisi
- db_datawriter: tüm tablolara yazma yetkisi
- db_owner: tüm işlemleri yapma yetkisi



Şekil 10 Role-Based Access Control (RBAC)



Şekil 11 SQL Server Authentication Kullanıcısı Rol Adı



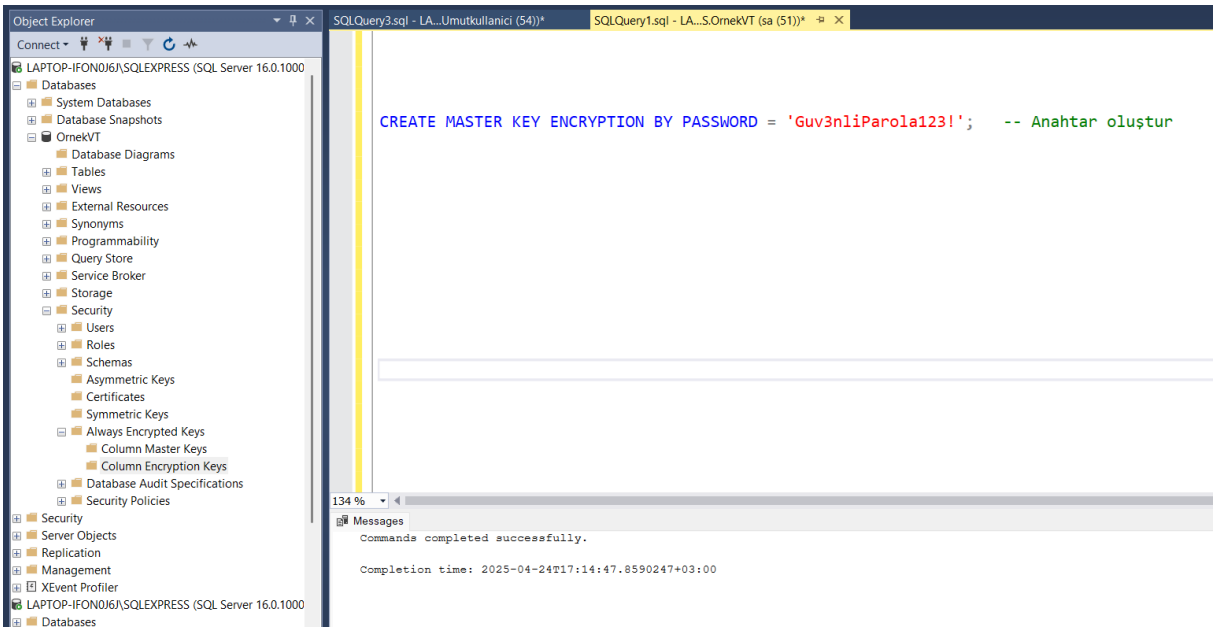
Şekil 12 Windows Authentication Kullanıcısı Rol Adı

4. VERİ ŞİFRELEME

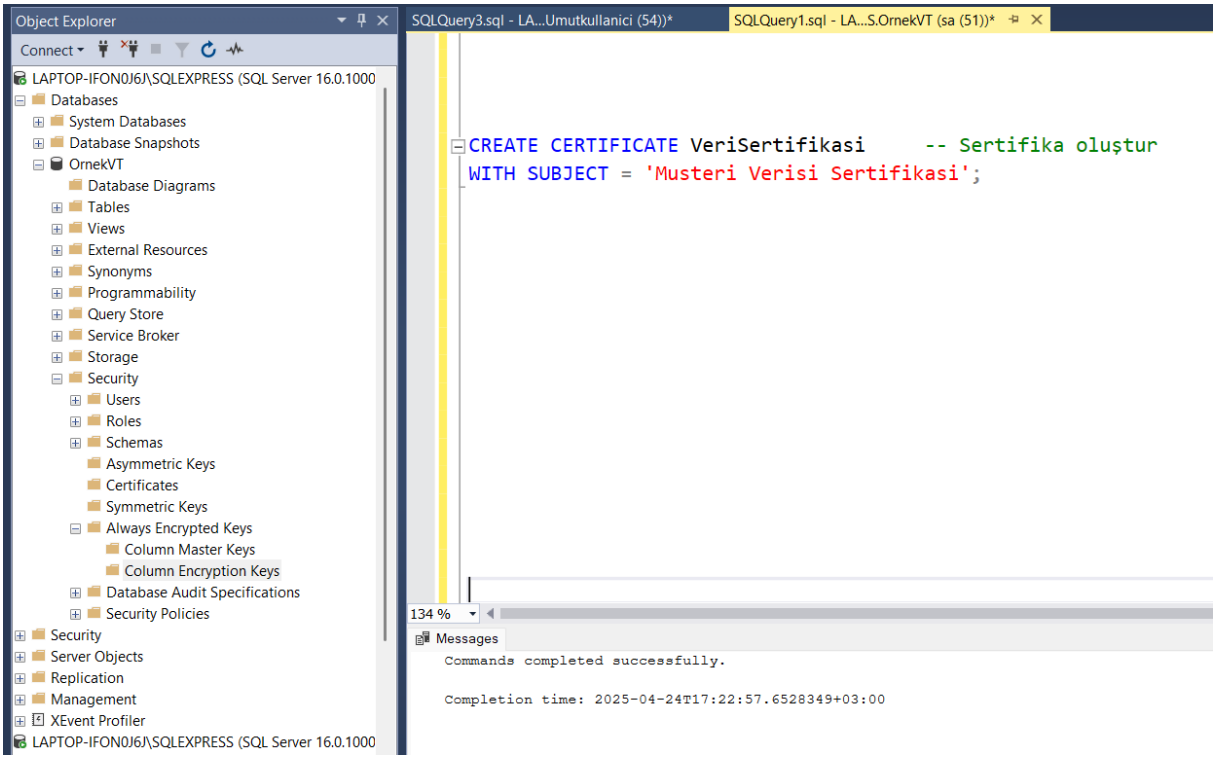
4.1. Şifreleme Yöntemlerinin Karşılaştırılması (TDE vs. Kolon Bazlı)

Projede veri şifreleme için öncelikle **Transparent Data Encryption (TDE)** yöntemi planlanmış olsa da kullanılan SQL Server sürümünün **Standard Edition** olması nedeniyle bu özellik desteklenmemektedir. TDE yalnızca Enterprise, Developer veya Azure gibi gelişmiş sürümlerde aktif olarak kullanılabilir. Bu nedenle, proje kapsamında alternatif bir yöntem olarak **kolon bazlı veri şifreleme** tekniği tercih edilmiştir.

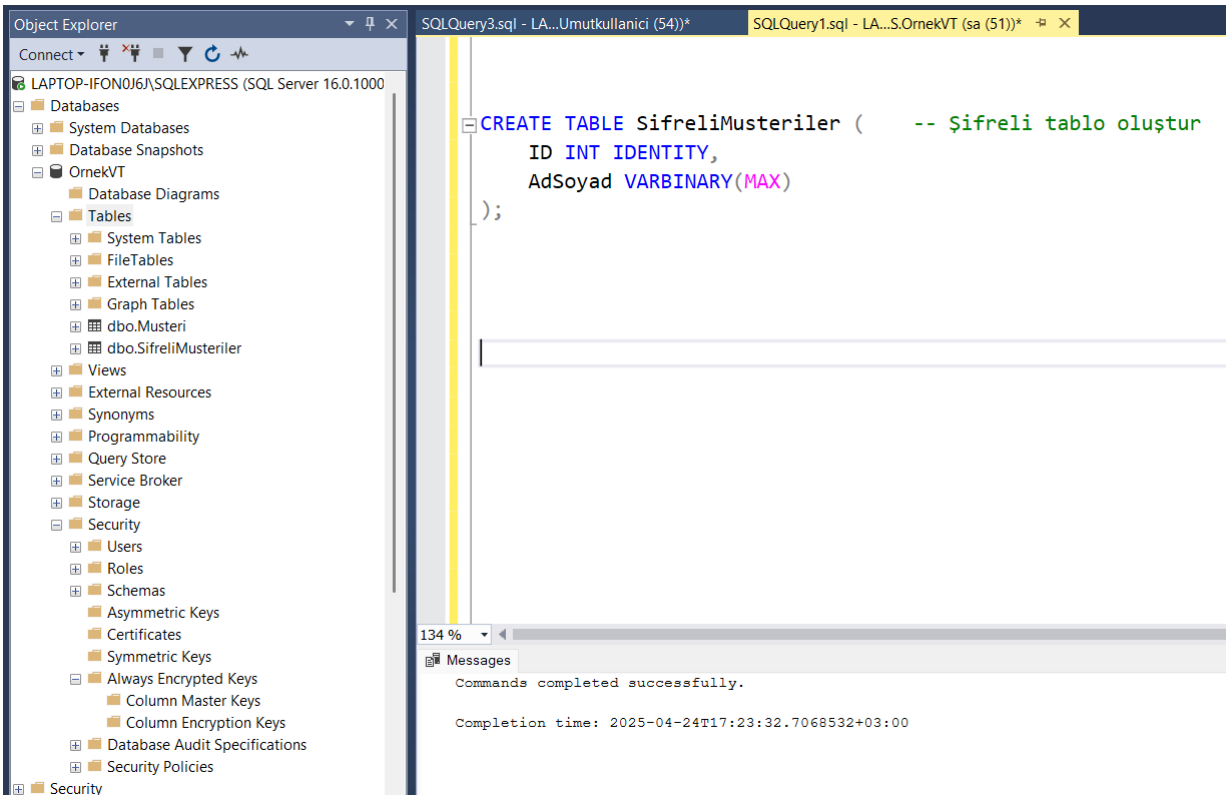
Kolon bazlı şifrelemede, hassas veriler (örneğin müşteri isimleri veya e-posta adresleri) veritabanına kaydedilmeden önce özel bir sertifika ile şifrelenir. Bu amaçla önce bir **Master Key** ve ardından bir **sertifika** oluşturulur. Veriler EncryptByCert fonksiyonu ile şifrelenerek saklanır ve DecryptByCert ile çözümlenebilir. Bu yöntem sayesinde, özellikle TDE desteklenmeyen ortamlarda veri güvenliği sağlanmış olur.



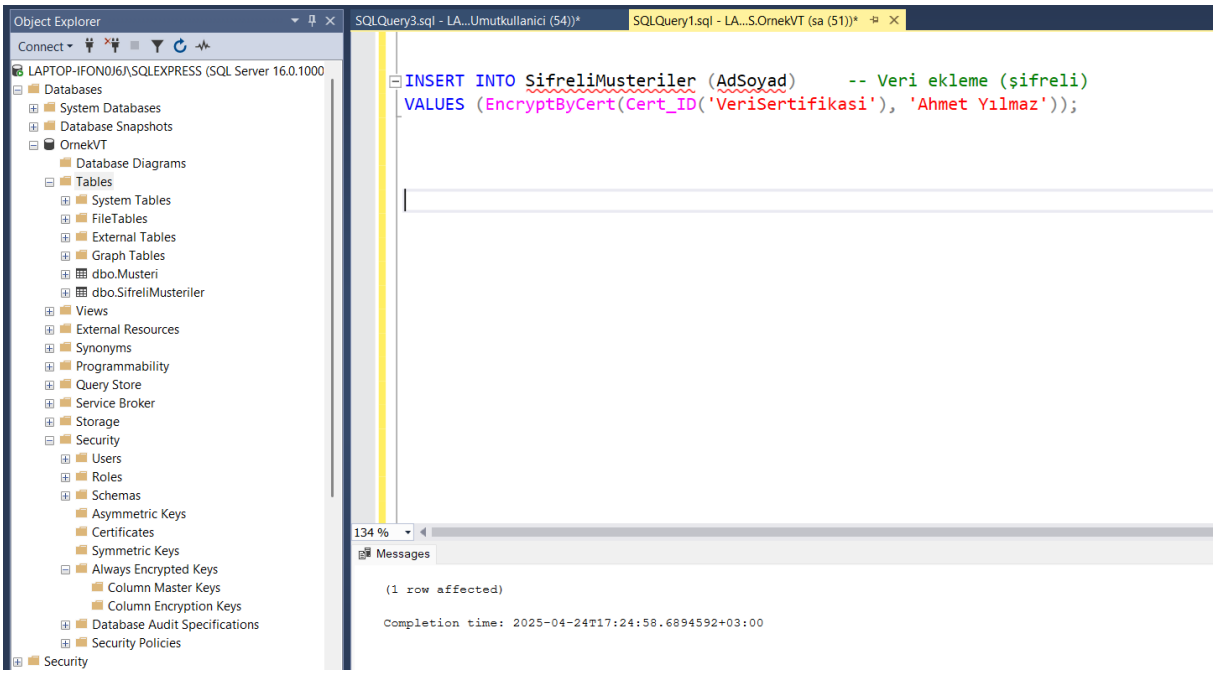
Şekil 13 Anahtar Oluşturma



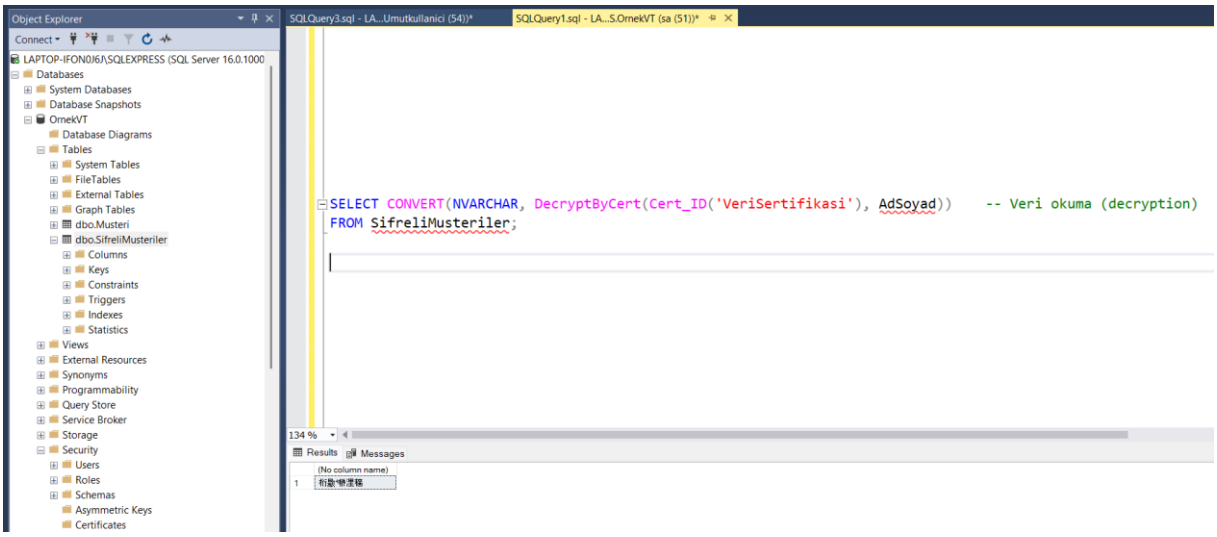
Şekil 14 Sertifika Oluşturma



Şekil 15 Şifreli Tablo Oluşturma



Şekil 16 Şifreli Tabloya Veri Ekleme



Şekil 17 Şifreli Tablodan Veri Okuma

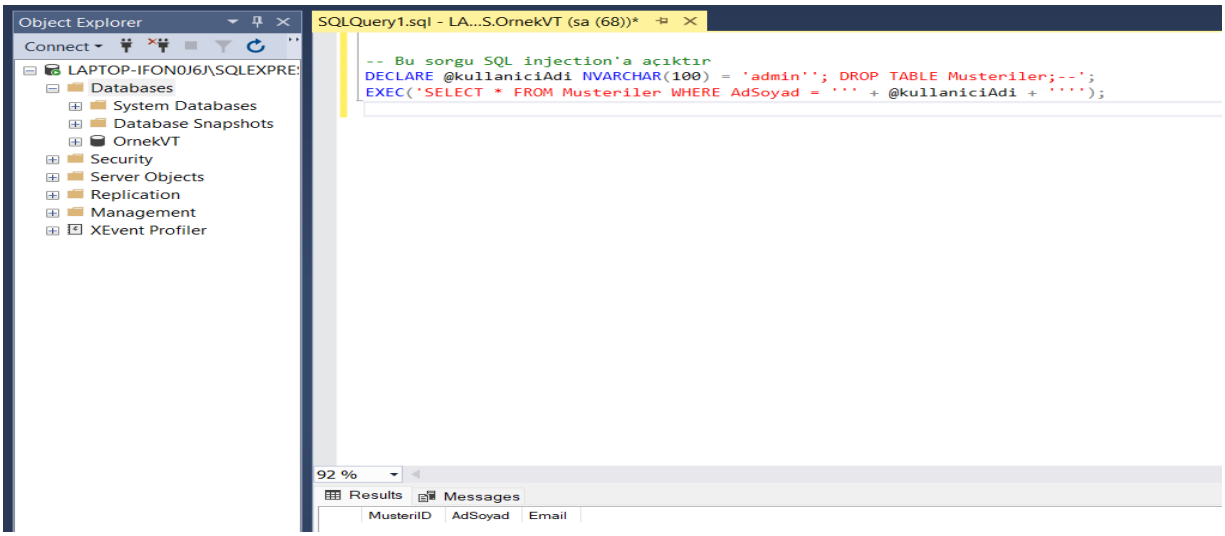
5. SQL INJECTION GÜVENLİĞİ

5.1. SQL Injection Nedir?

SQL Injection, kötü niyetli kullanıcıların, uygulama aracılığıyla veritabanına gönderilen SQL sorgularına zararlı komutlar ekleyerek sisteme izinsiz erişim sağlamaya çalıştığı bir saldırı türüdür. Bu saldırı tekniği, kullanıcı girdilerinin doğru şekilde filtrelenmemesi durumunda veritabanında veri sızdırma, silme veya sistem ele geçirme gibi ciddi güvenlik açıklarına yol açabilir.

5.2. Güvensiz Sorgu Örneği ve Analizi

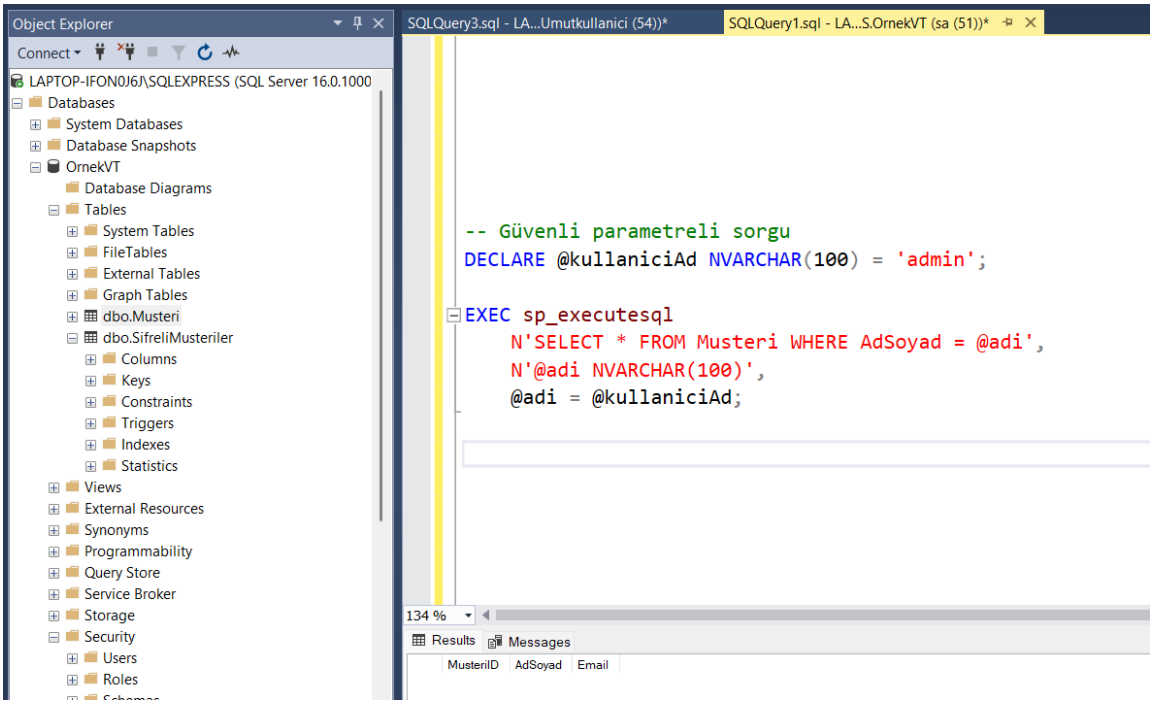
Güvensiz sorgular, kullanıcı girdisinin doğrudan SQL komutlarına eklenmesiyle oluşturulan sorgulardır. Bu yöntem SQL Injection saldırılarına açıktır. Örneğin, kullanıcıdan alınan bir e-posta adresinin doğrudan WHERE koşuluna eklenmesi saldırganların zararlı SQL komutları çalıştırmasına sebep olabilir. Bu yüzden bu yöntem kesinlikle kullanılmamalıdır.



Şekil 18 Güvensiz Sorgu Örneği ve Analizi

5.3. Parametreli Sorgularla Güvenli Kodlama

Güvenli sorgular, kullanıcı girdisinin doğrudan sorguya gömülmesi yerine, parametreler aracılığıyla sorguya iletilmesiyle oluşturulur. Bu yöntem, SQL Injection saldırılarını engeller çünkü veriler otomatik olarak doğrulanır ve komut olarak değil, veri olarak değerlendirilir. SQL Server'da bu amaçla sp_executesql gibi yöntemler veya uygulama tarafında SqlCommand.Parameters kullanılır.

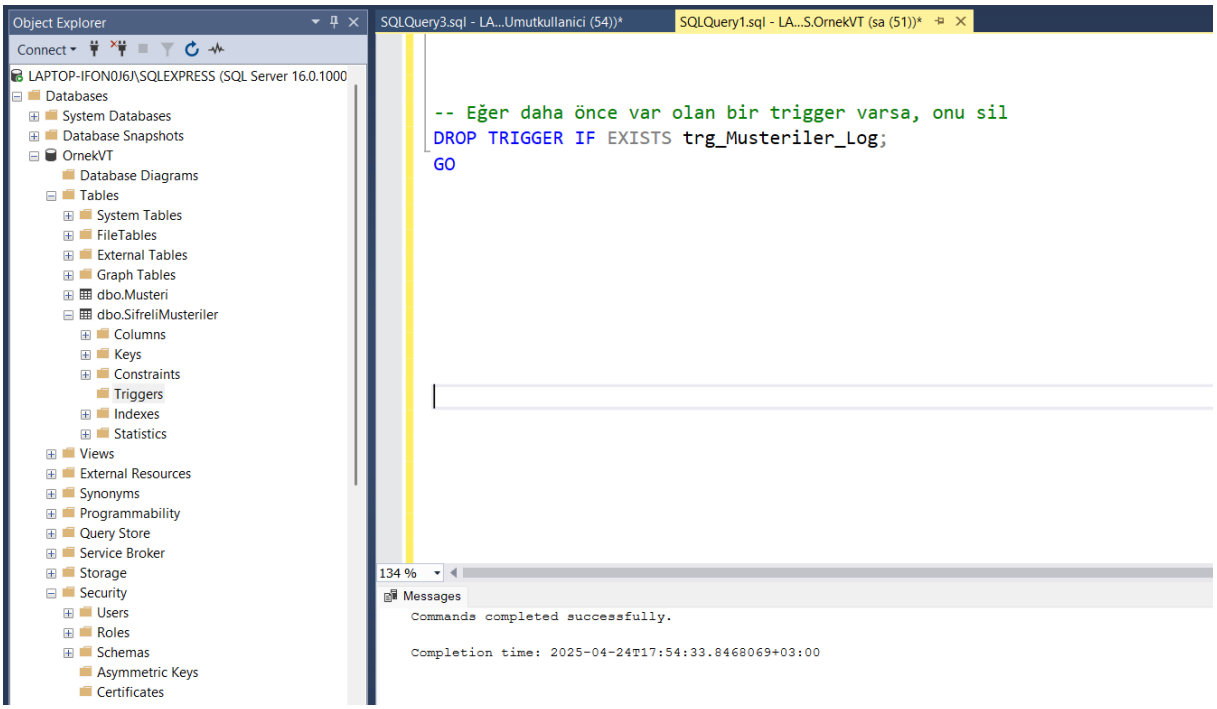


Şekil 19 Güvenli Parametreli Sorgu

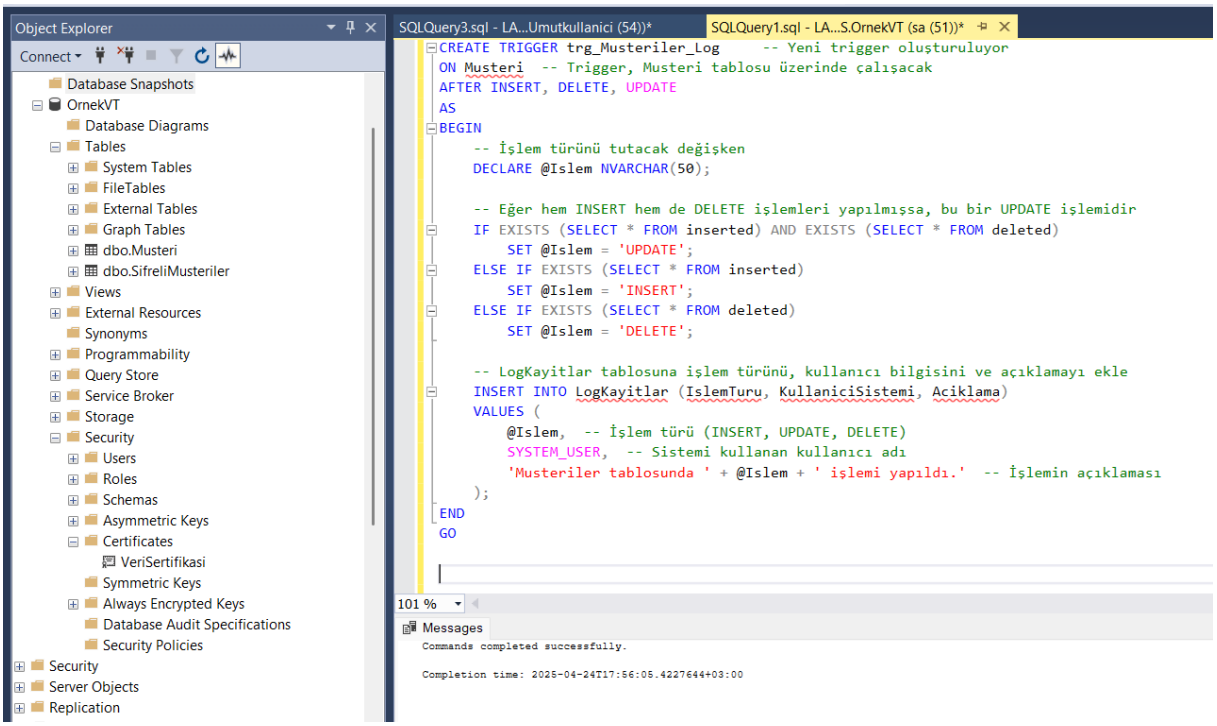
6. KULLANICI AKTİVİTELERİNİ İZLEME

6.1. Trigger ile Loglama Yöntemi

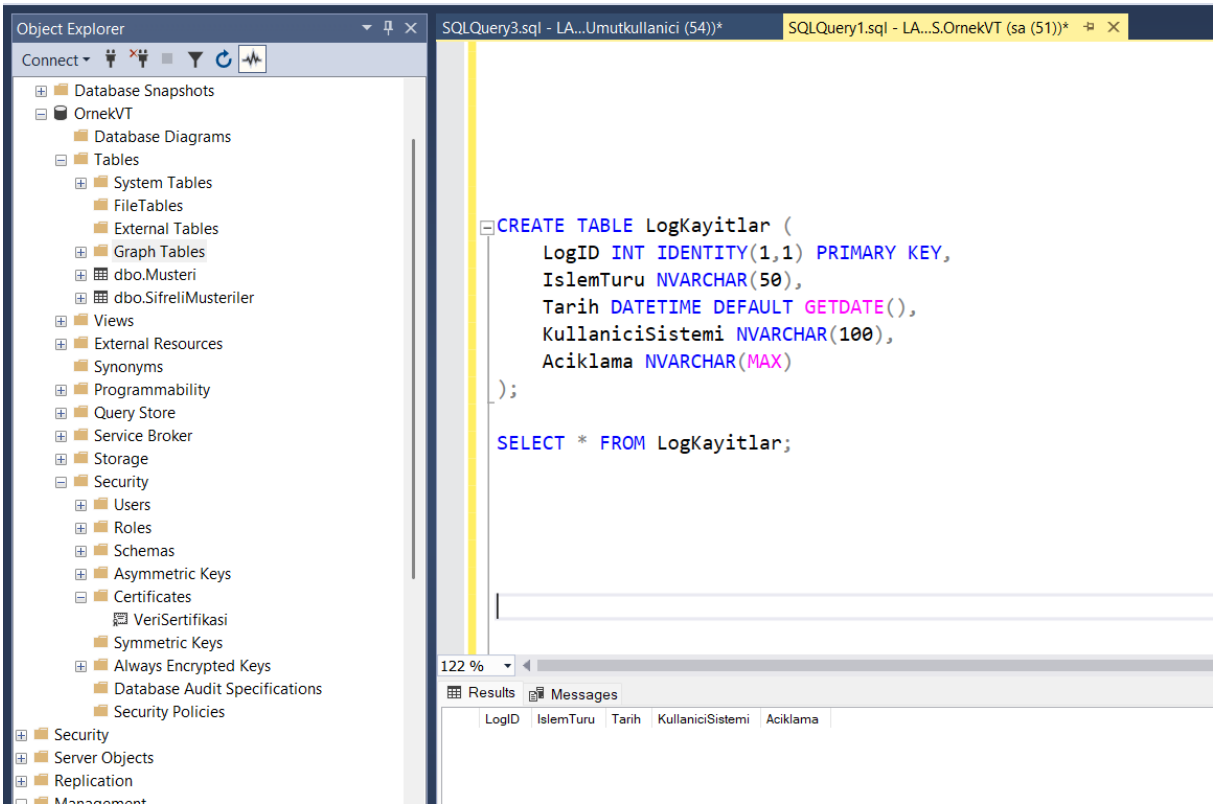
Bu trigger, Musteri tablosu üzerinde yapılan **INSERT**, **DELETE** ve **UPDATE** işlemlerini otomatik olarak LogKayitlar tablosuna kaydeder. Her işlem gerçekleştiğinde, işlem türü (@Islem), işlemi yapan kullanıcı (SYSTEM_USER) ve açıklama bilgisi ile loglanır. Bu sayede, veritabanı üzerinde yapılan değişiklikler izlenebilir hale gelir ve kullanıcı aktiviteleri güvenli bir şekilde takip edilebilir. Audit özelliği kullanılamadığında etkili bir alternatif olarak kullanılabilir.



Şekil 20 Trigger Silme (Eğer Önceden Var İse)

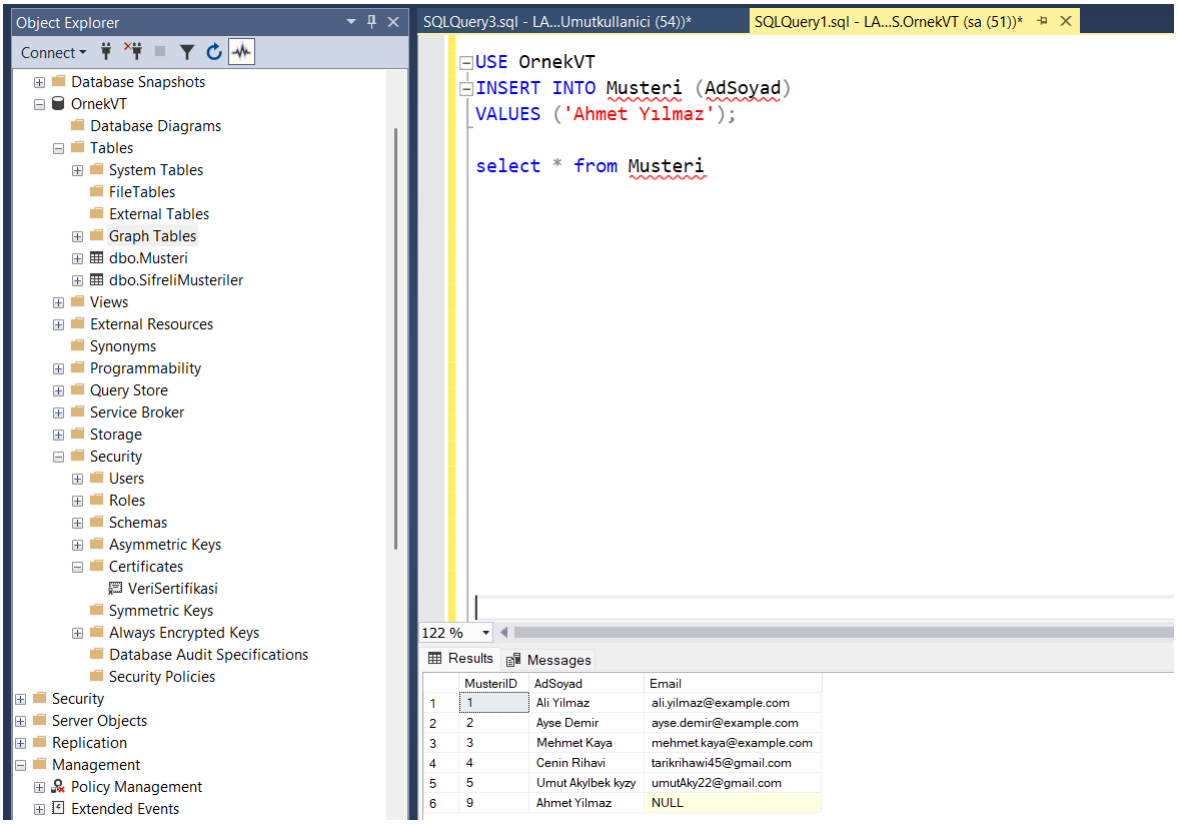


Şekil 21 Trigger Oluşturma

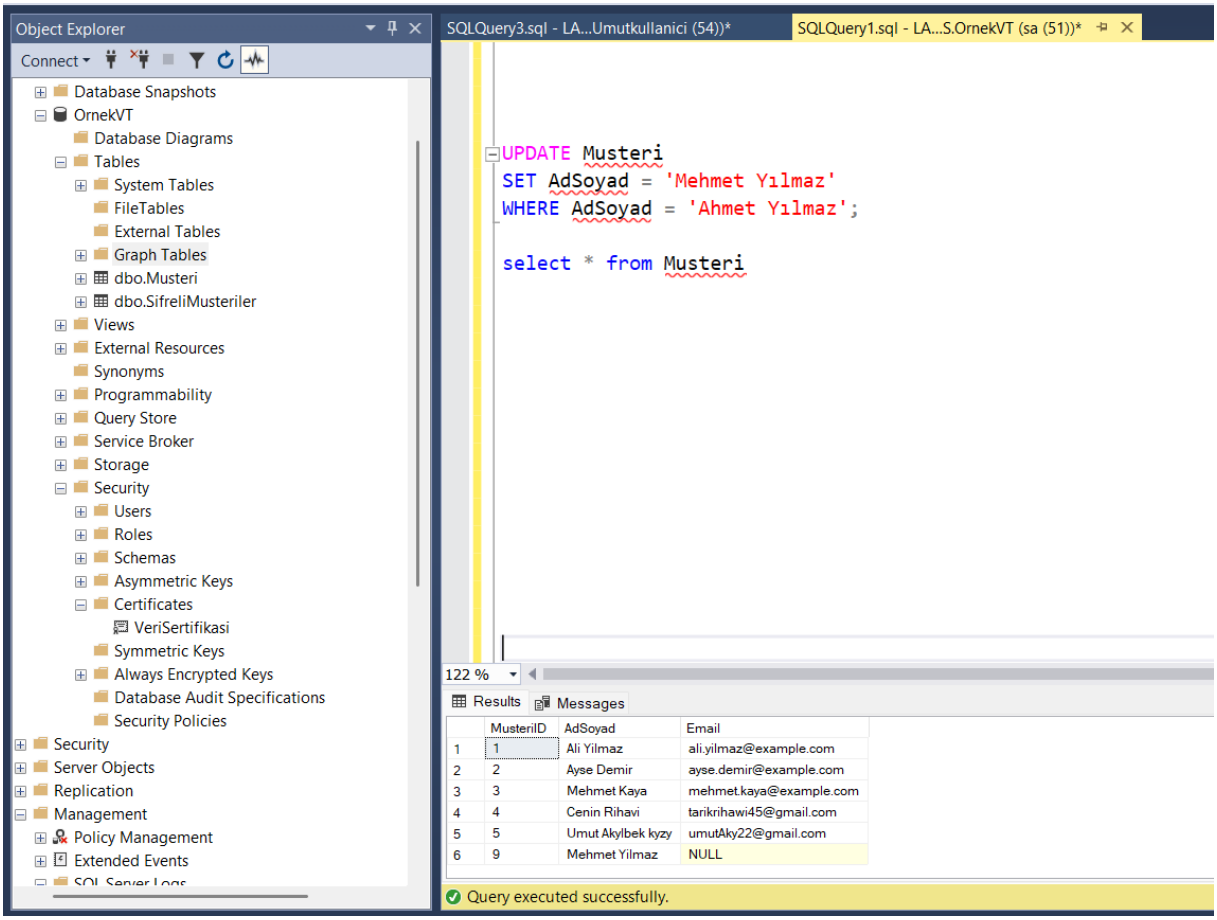


Şekil 22 Log Kayıtları Tablosu Oluşturma

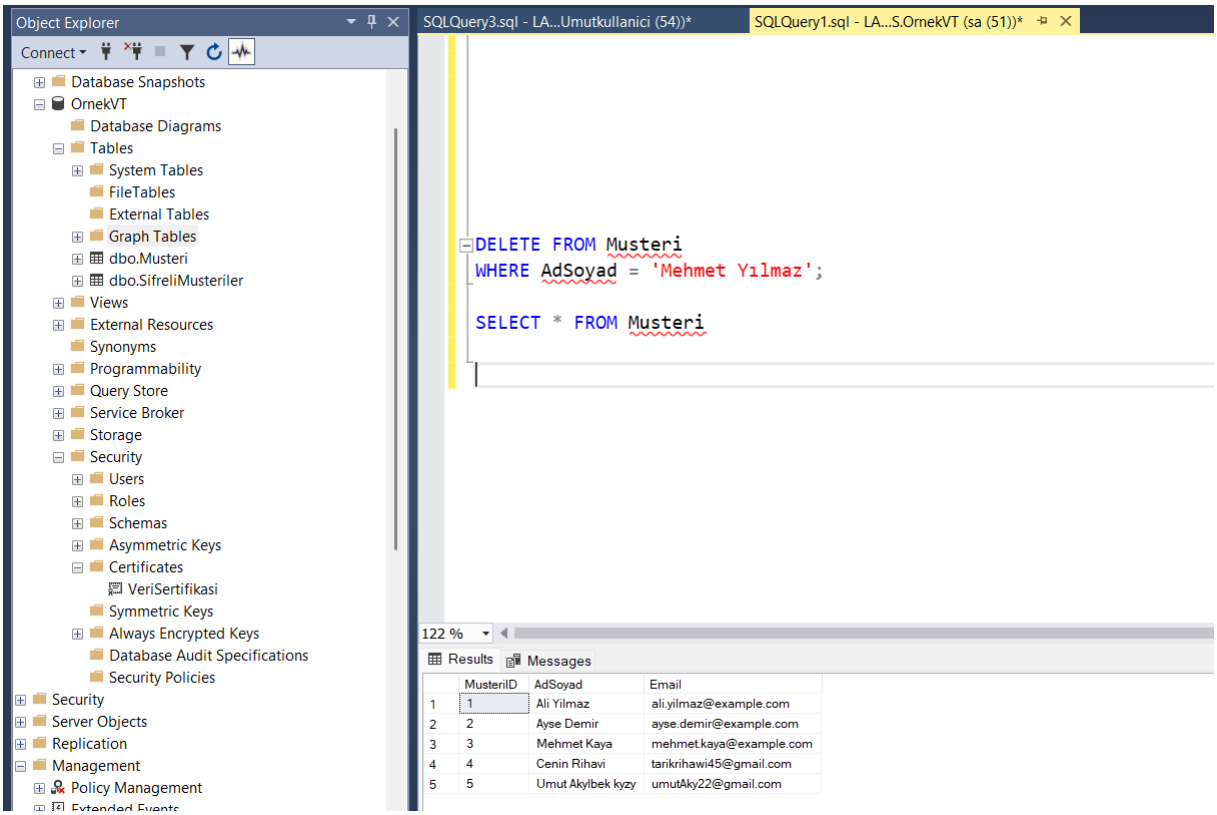
Log kayıtları tablosunun çalışmasını test etmek amacıyla, INSERT, UPDATE ve DELETE komutları kullanılmıştır.



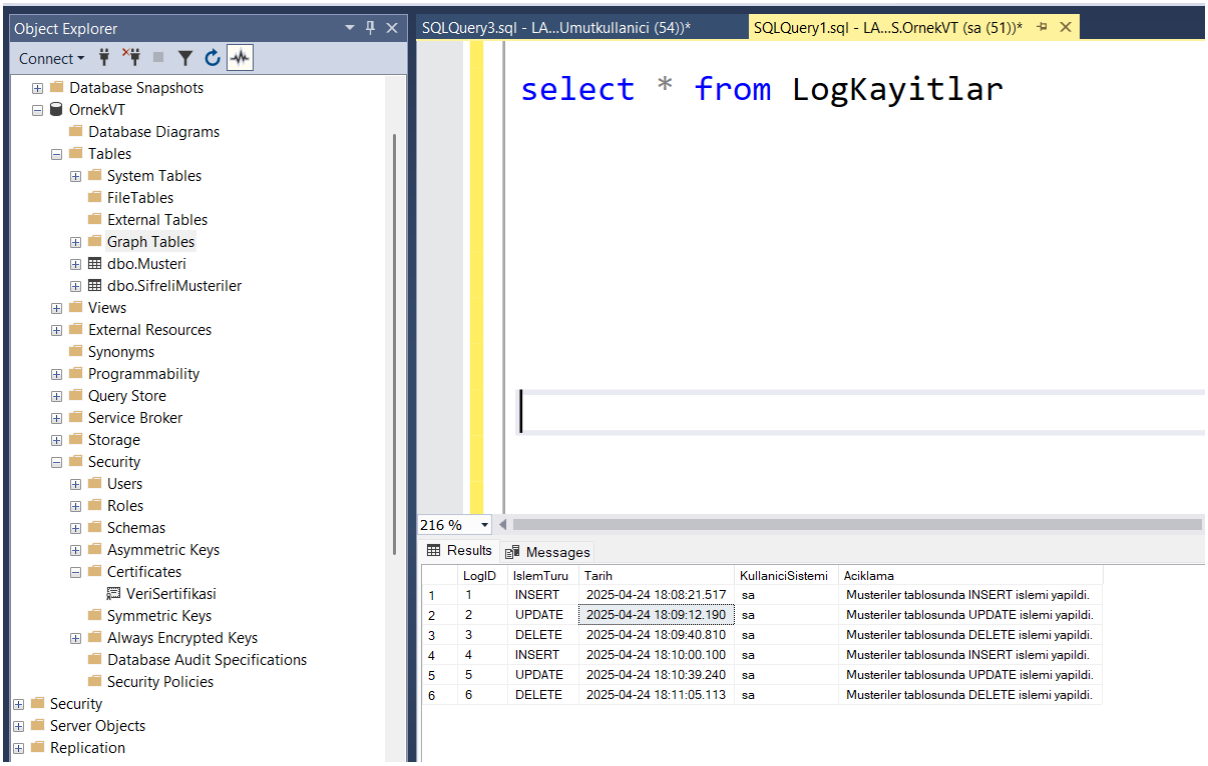
Şekil 23 Müşteri Tablosuna Veri Ekleme



Şekil 24 Müşteri Tablosundaki Veriyi Update Etme



Şekil 25 Müşteri Tablosundan Veri Silme



Şekil 26 Log Kayıtları Tablosu

Notlar:

- SSMS üzerinden **Security> Logins** sekmesinden kullanıcıları görüp, şifreleri değiştirebilirsin.
- Yetkilerde hata yaparsan sp_droprolemember ve DROP USER, DROP LOGIN komutlarını kullanarak temizleyebilirsin.

SONUÇ

Bu proje kapsamında, veritabanı güvenliğini sağlamak için çeşitli yöntemler ve teknikler uygulanarak sistematik bir güvenlik yapısı oluşturulmuştur. Projeyi genel hatlarıyla değerlendirdiğimizde, aşağıdaki sonuçlara ulaşılabilir:

1. **Erişim Kontrolü ve Yetkilendirme**: Veritabanı güvenliğinin temel unsurlarından biri olan erişim kontrolü, SQL Server Authentication ve Windows Authentication yöntemlerinin uygulanmasıyla sağlanmıştır. Kullanıcılar, ihtiyaçlarına göre uygun yetkilerle donatılmış ve Role-Based Access Control (RBAC) sistemi ile her bir kullanıcının veritabanındaki rolü ve yetkileri net bir şekilde belirlenmiştir. Bu sayede, yalnızca yetkili kullanıcıların kritik verilere erişmesi sağlanmıştır.

2. **Veri Şifreleme**: Veritabanında kullanılan veri şifreleme yöntemleri, özellikle hassas verilerin güvenliğini sağlamak için önemli bir rol oynamaktadır. TDE (Transparent Data Encryption) gibi gelişmiş şifreleme yöntemlerinin, kullanılan SQL Server sürümü nedeniyle sınırlı kalması üzerine, alternatif olarak kolon bazlı şifreleme kullanılmıştır. Bu yöntem, verilerin veritabanına şifreli olarak kaydedilmesini sağlayarak, olası güvenlik açıklarını minimize etmiştir.

3. **SQL Injection Güvenliği**: SQL Injection saldırılarına karşı alınan önlemler, proje kapsamında en iyi güvenlik teknikleriyle güçlendirilmiştir. Parametreli sorgular kullanarak SQL Injection'a karşı koruma sağlanmış, güvenli kodlama teknikleriyle saldırı riskleri minimize edilmiştir. Ayrıca, güvensiz sorguların tespiti ve önlenmesi için uygulama seviyesinde gerekli önlemler alınmıştır.

4. **Kullanıcı Aktivite İzleme**: Veritabanı üzerinde gerçekleştirilen işlemler, trigger kullanılarak etkili bir şekilde loglanmıştır. Bu sayede, kullanıcı aktiviteleri takip edilerek, olası güvenlik ihlalleri veya hatalı işlemler kolayca izlenebilmiştir. Ayrıca, loglama süreci, kullanıcı aktivitelerinin güvenli bir şekilde kaydedilmesine olanak tanımıştır.

KAYNAKÇA

- Microsoft, SQL Server Security Best Practices Guide **https://learn.microsoft.com/en-us/sql/relational-databases/security/sql-server-security-best-practices?view=sql-server-ver15**
- OWASP SQL Injection Prevention Cheat Sheet

https://owasp.org/www-community/attacks/SQL_Injection

- **YouTube, 2025, https://youtu.be/PjZH-CO1hX0?si=abEWNOLaOy6ptj4P Erişim Tarihi: 23 Nisan 2025.**
-
- **YouTube, 2025, https://youtu.be/N8xEgSe5RwE?si=InXvlwwv4pc2GUBa Erişim Tarihi: 23 Nisan 2025.**