**ANKARA ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**amblem, daire, logo, simge, sembol içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**Ağ Tabanlı Paralel Dağıtım Sistemleri**

**1. Veritabanı Performans Optimizasyonu ve İzleme**

**2. Veritabanı Yedekleme ve Felaketten Kurtarma Planı**

**3. Veritabanı Güvenliği ve Erişim Kontrolü**

**Hüseyin Göksu Hacıoğlu**

**21290766**

[**https://github.com/goksu9**](https://github.com/goksu9)

**25.04.2025**

İçindekiler

[1. Proje Başlığı 3](#_Toc196315332)

[2. Projenin Amacı 3](#_Toc196315333)

[3. Kullanılan Veri Seti 3](#_Toc196315334)

[4. Planlanan Aşamalar 3](#_Toc196315335)

[5. Geliştirme Ortamı 4](#_Toc196315336)

[PROJE 1 5](#_Toc196315337)

[Veritabanı İzleme: SQL Profiler ve DMV Kullanımı 5](#_Toc196315338)

[Aşama 1: Sorgu Hızlandırmak İçin İndeks Yönetimi 7](#_Toc196315339)

[Aşama 2: Sorgu İyileştirme 8](#_Toc196315340)

[PROJE 2 10](#_Toc196315341)

[Veritabanı Yedekleme ve Felaketten Kurtarma Planı 10](#_Toc196315342)

[Aşama 1: Tam Yedekleme (Full Backup) 10](#_Toc196315343)

[Aşama 2: Artık (Transaction Log) ve Fark (Differential) Yedekleme 10](#_Toc196315344)

[Aşama 3: Fark (Differential) Yedekleme (GUI Üzerinden) 11](#_Toc196315345)

[Aşama 4: (Log) Yedekleme (GUI Üzerinden) 12](#_Toc196315346)

[Aşama 5: Otomatik T-SQL ile Kullanıcıları Kapatarak Geri Yükleme 14](#_Toc196315347)

[Aşama 6: Test Yedekleme Senaryoları – Yedeklerin Doğruluğunu Test Etme 15](#_Toc196315348)

[PROJE 3 18](#_Toc196315349)

[Veritabanı Güvenliği ve Erişim Kontrolü 18](#_Toc196315350)

[Aşama 1 Amaç: Yeni kullanıcı oluşturmak ve sadece belirli tablolara/verilere erişmesini sağlamak 18](#_Toc196315351)

[Aşama 2: Veri Şifreleme – TDE (Transparent Data Encryption 20](#_Toc196315352)

[Aşama 3: SQL Injection Testleri – Güvenlik Açıklarını Simüle Etme 23](#_Toc196315353)

[Aşama 4: Audit Loglama – Kullanıcı Aktivitelerini İzleme 24](#_Toc196315354)

# 1. Proje Başlığı

Veritabanı Performans Optimizasyonu ve İzleme

# 2. Projenin Amacı

Bu projenin amacı, MSSQL Server üzerinde örnek bir veri tabanı kullanarak veritabanı performansını izlemek, yavaş çalışan sorguları analiz etmek, indeks yönetimi uygulamak ve genel performansı artıracak optimizasyon tekniklerini kullanmaktır.

# 3. Kullanılan Veri Seti

**Dosya Adı :** vgsales.csv

**Kaynak :** İnternet üzerinden temin edilmiştir.

**İçerik :** Video oyunlarının adı, platformu, çıkış yılı, türü, yayıncısı ve farklı bölgelerdeki satış miktarlarını içermektedir.

|  |  |
| --- | --- |
| Sütun Adı | Açıklama |
| Rank | Oyunun sıralaması |
| Name | Oyunun adı |
| Platform | Yayınlandığı platform |
| Year | Çıkış yılı |
| Genre | Türü (Aksiyon, Spor, RPG, vs.) |
| Publisher | Yayıncı firma |
| NA\_Sales | Kuzey Amerika satışları (milyon adet) |
| EU\_Sales | Avrupa satışları |
| JP\_Sales | Japonya satışları |
| Other\_Sales | Diğer bölge satışları |
| Global\_Sales | Tüm dünyadaki toplam satış |

# 4. Planlanan Aşamalar

|  |  |
| --- | --- |
| **Adım No** | **Açıklama** |
| 1 | Veri setinin MSSQL’e aktarılması (Import) |
| 2 | SQL Server Profiler veya DMV kullanılarak sorgu performanslarının izlenmesi |
| 3 | Geciken sorguların analiz edilmesi |
| 4 | Doğru indekslerin oluşturulması |
| 5 | Gereksiz indekslerin kaldırılması |
| 6 | Uzun süren sorgular için yeniden yazım (sorgu optimizasyonu) |
| 7 | Kullanıcı rolleri ve yetkilendirme ile veri yöneticisi erişimlerinin belirlenmesi |
| 8 | Elde edilen performans sonuçlarının karşılaştırılması |

# 

# 5. Geliştirme Ortamı

|  |  |
| --- | --- |
| **Araç / Yazılım** | **Açıklama** |
| MSSQL Management Studio | Veritabanı yönetimi ve sorgular |
| Microsoft SQL Server | Veritabanı altyapısı |
| Excel / Not Defteri | CSV verisinin ön görüntülenmesi |
| GitHub | Projenin versiyon takibi |
| Clipchamp | Süreçlerin video kaydı |

# PROJE 1

# Veritabanı İzleme: SQL Profiler ve DMV Kullanımı

Bu aşamada, SQL Server’da çalışan sorguların performansını izleyebilmek amacıyla iki farklı yöntem kullanılmıştır:

1. SQL Server Profiler aracı ile canlı sorgu trafiği izlenmiş, yavaş sorgular belirlenmiştir.
2. Dynamic Management Views (DMV) yardımıyla geçmişte çalıştırılmış sorguların çalışma süresi, tekrar sayısı gibi metrikleri analiz edilmiştir.

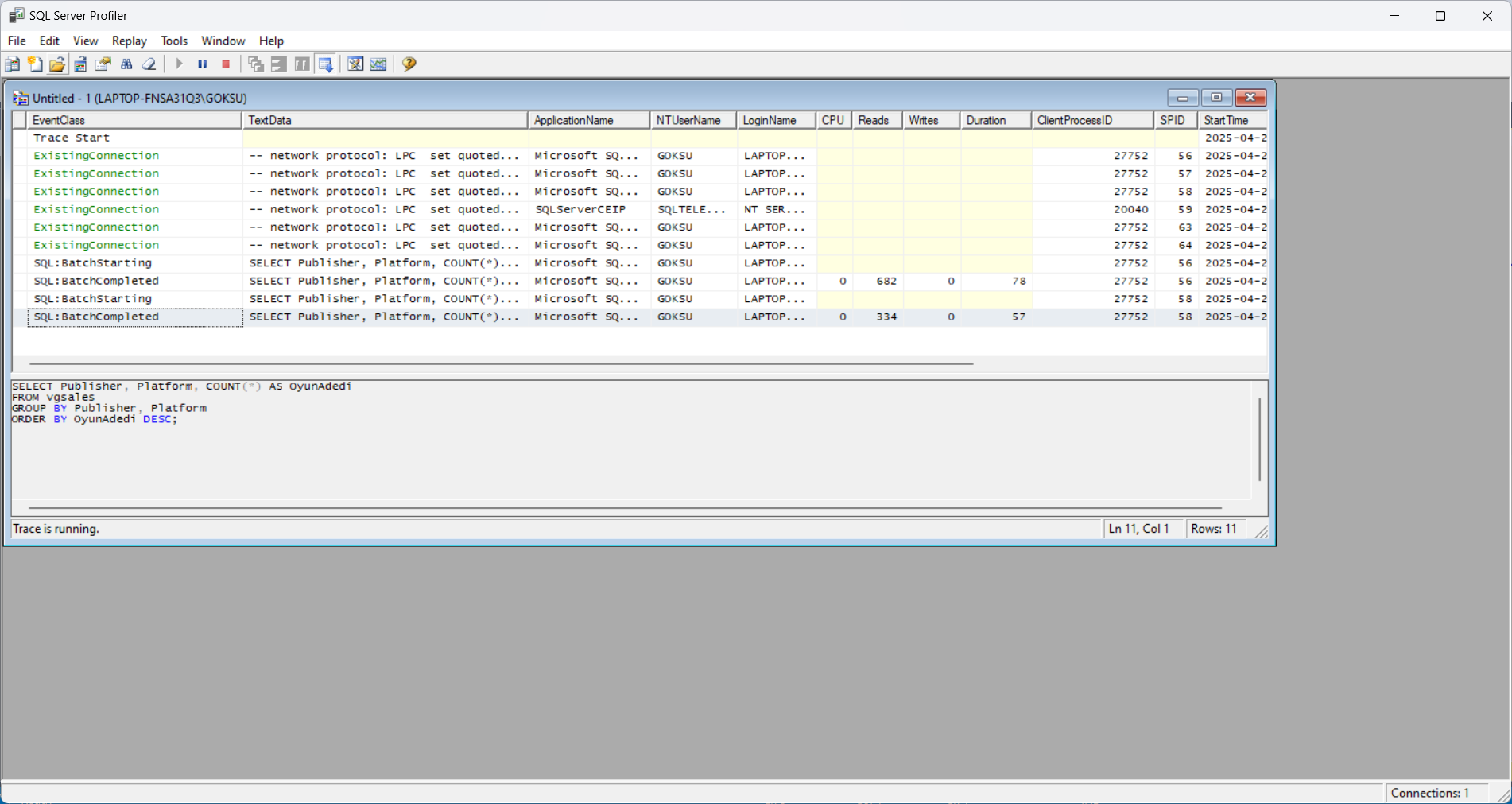
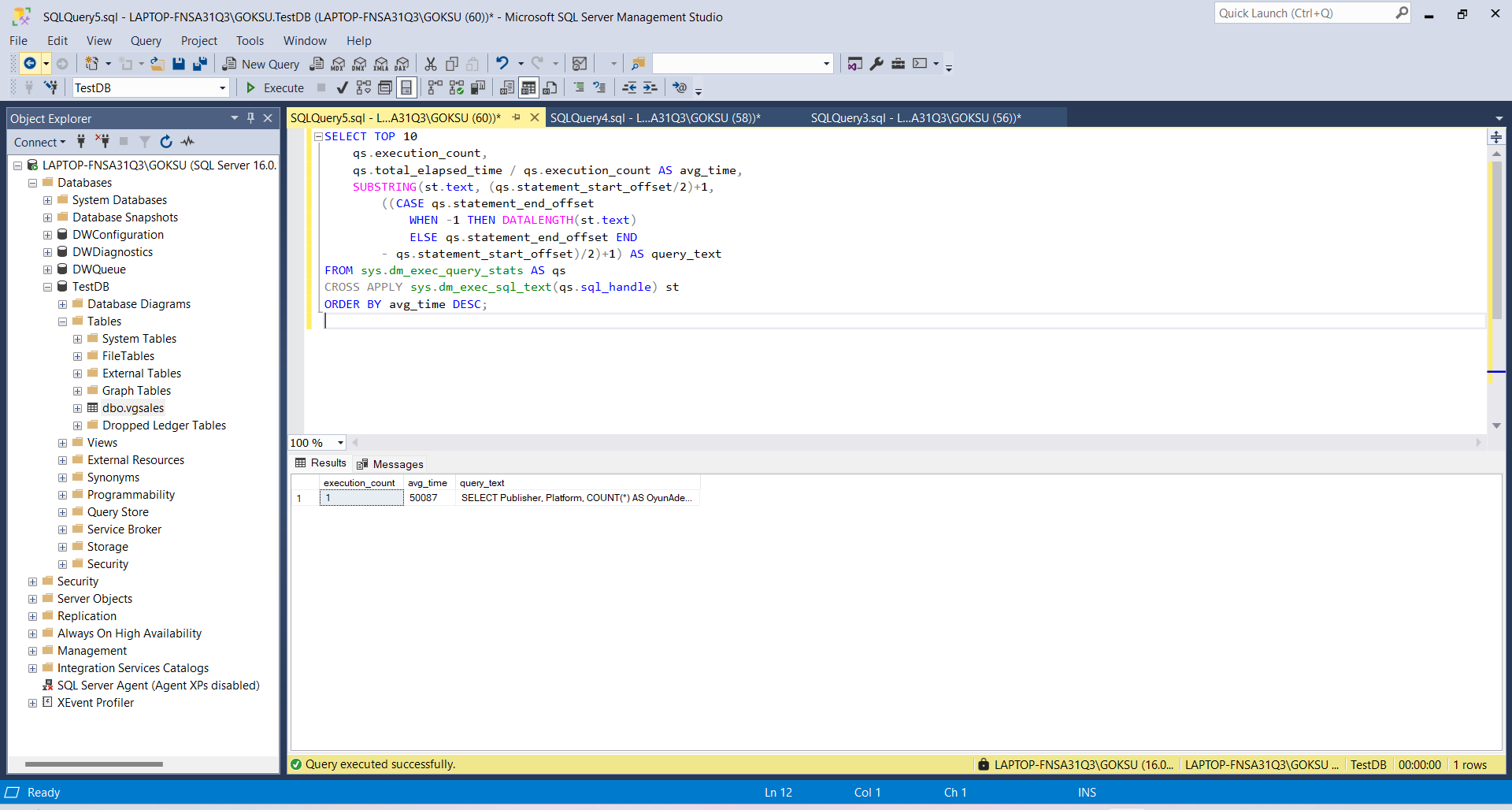


Figure 1 Server Profiler



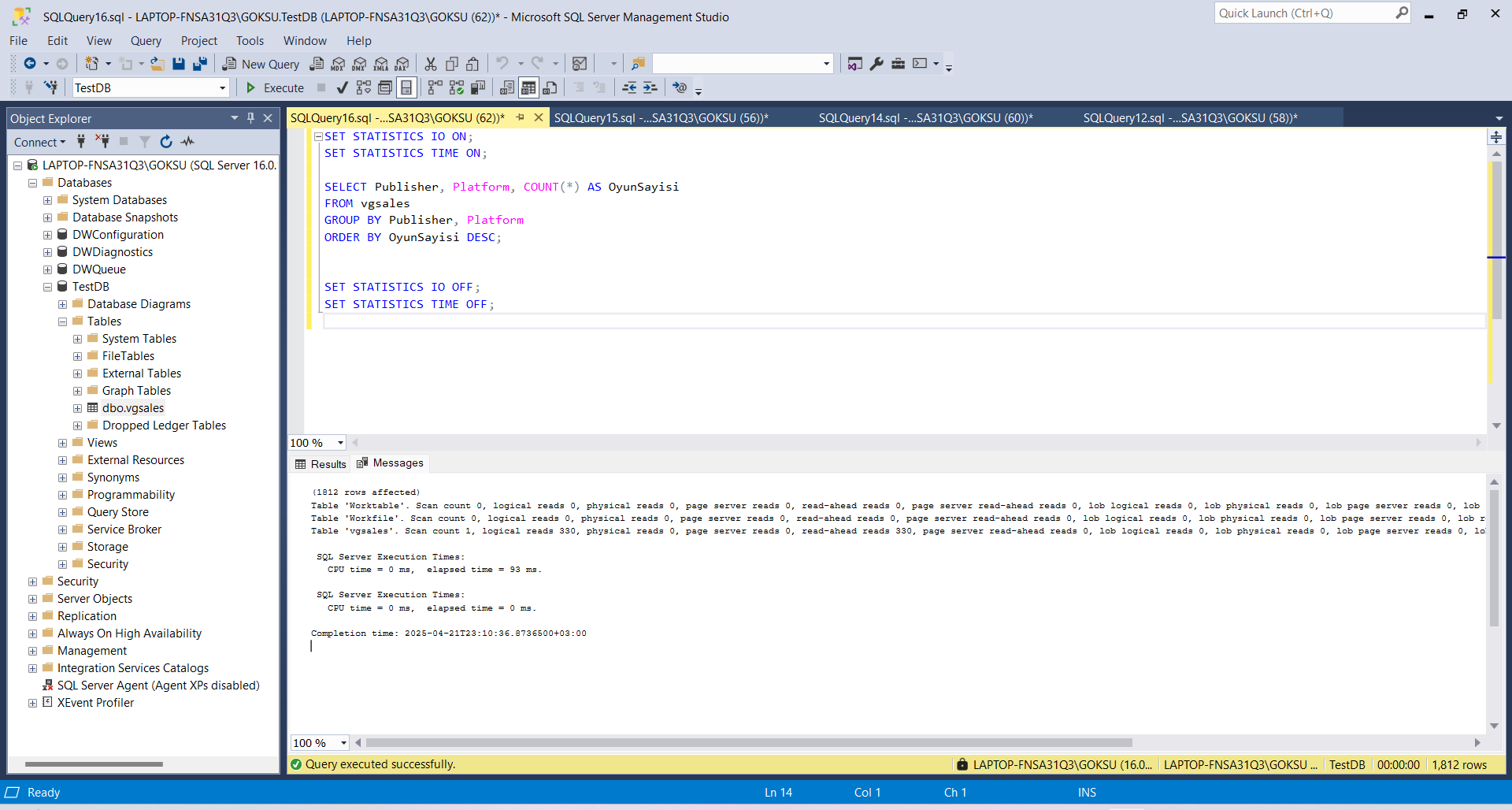


Figure 2 DMV

## Aşama 1: Sorgu Hızlandırmak İçin İndeks Yönetimi

Bu aşamada veritabanı sorgu performansını artırmak amacıyla `vgsales` tablosundaki bazı sütunlara indeksler eklenmiştir. Özellikle WHERE, GROUP BY veya ORDER BY gibi ifadelerde sıkça kullanılan sütunlar hedef alınmıştır. Ayrıca `sys.indexes` görünümü kullanılarak mevcut indeksler listelenmiş, gereksiz olanların kaldırılmasına karar verilmiştir.

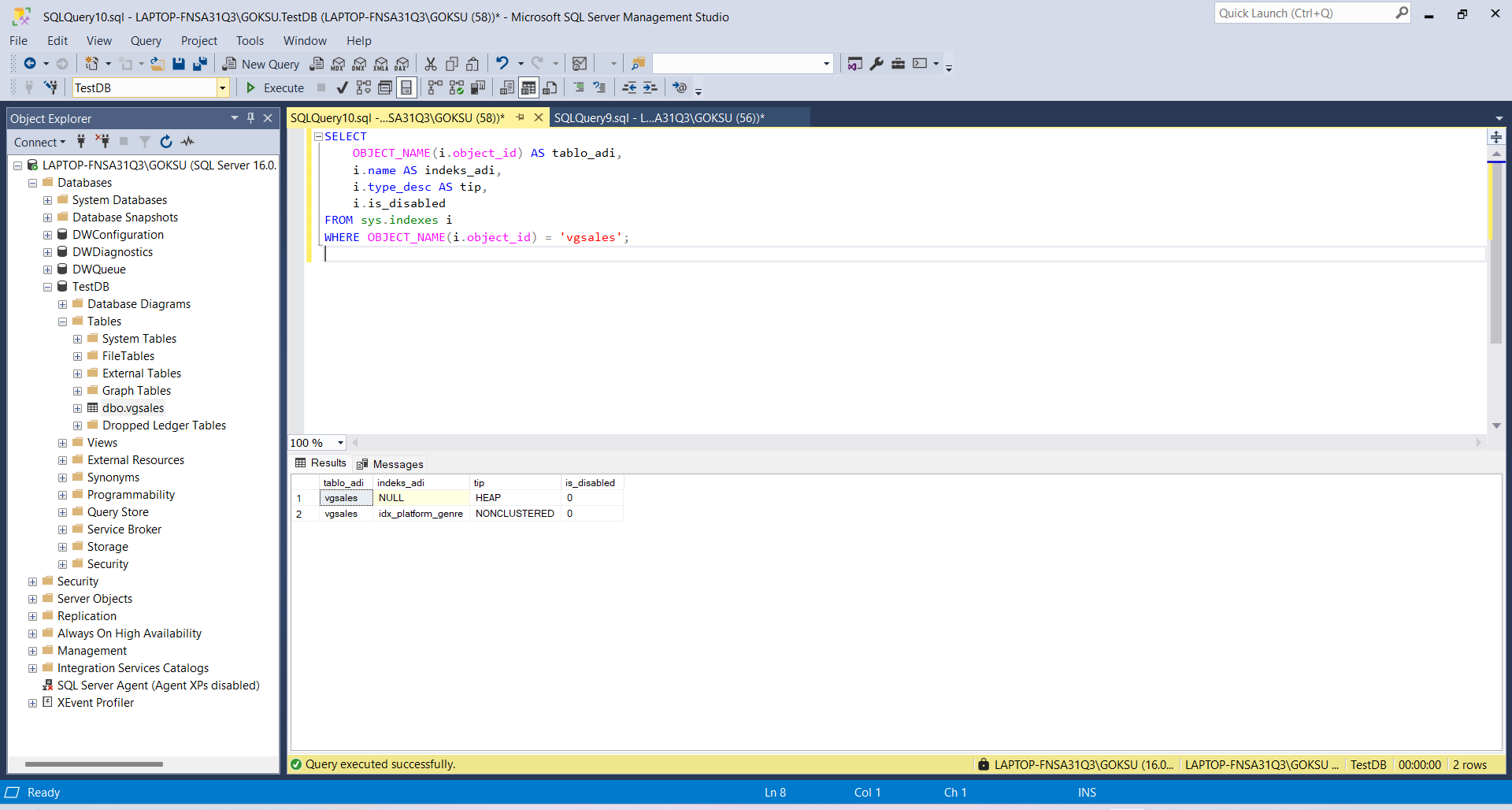


Figure 3 Index ekleme

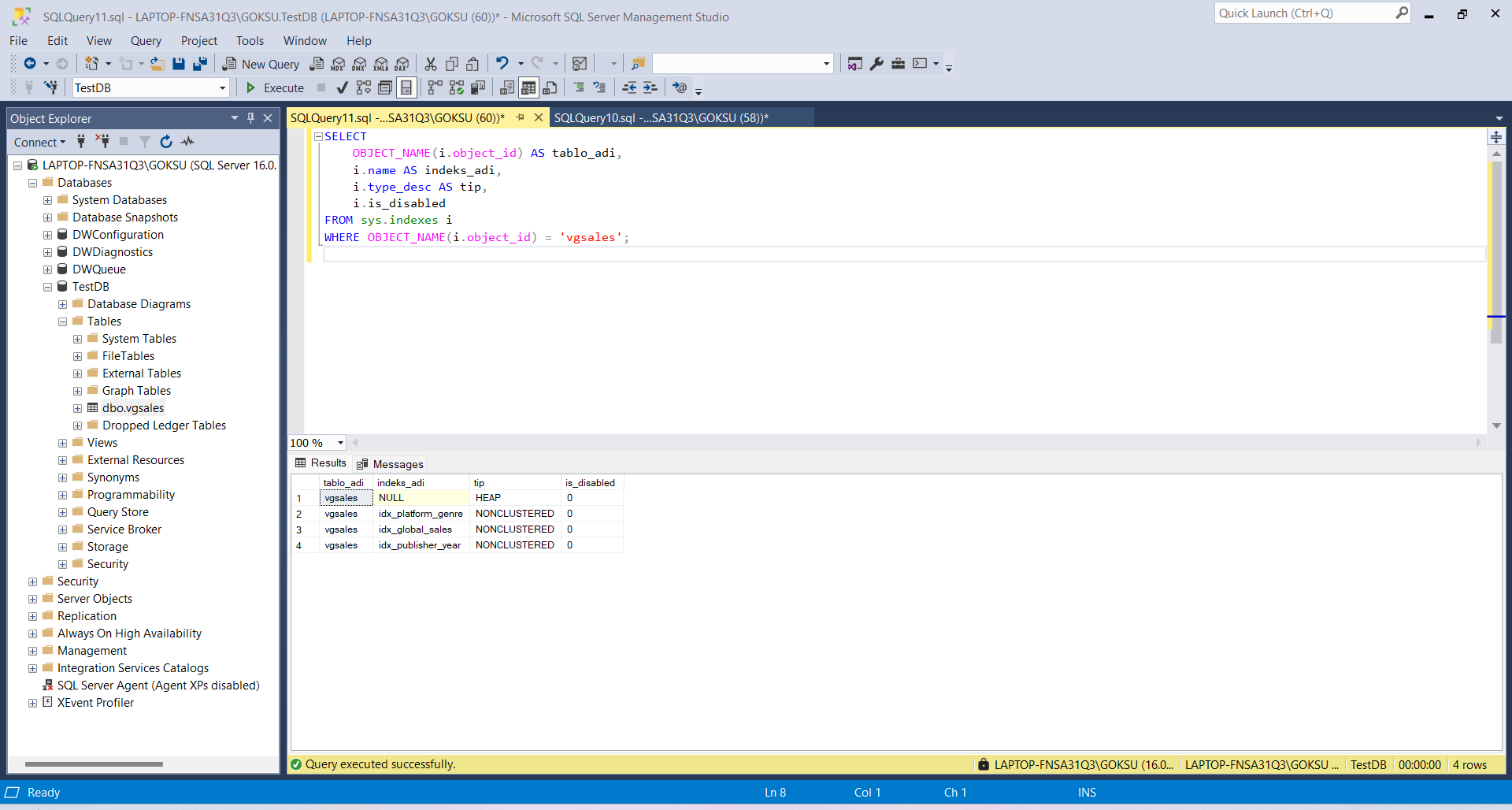


Figure 4 Eklenen indexler

## Aşama 2: Sorgu İyileştirme

Bu aşamada, SQL Server üzerinde çalışan ve ortalama süresi yüksek olan sorgular DMV aracılığıyla belirlenmiş, daha verimli hâle getirilmek üzere optimize edilmiştir. Optimize edilen sorgular öncesinde ve sonrasında `SET STATISTICS IO` ve `SET STATISTICS TIME` komutları kullanılarak kaynak kullanımı ve süre açısından karşılaştırılmıştır.

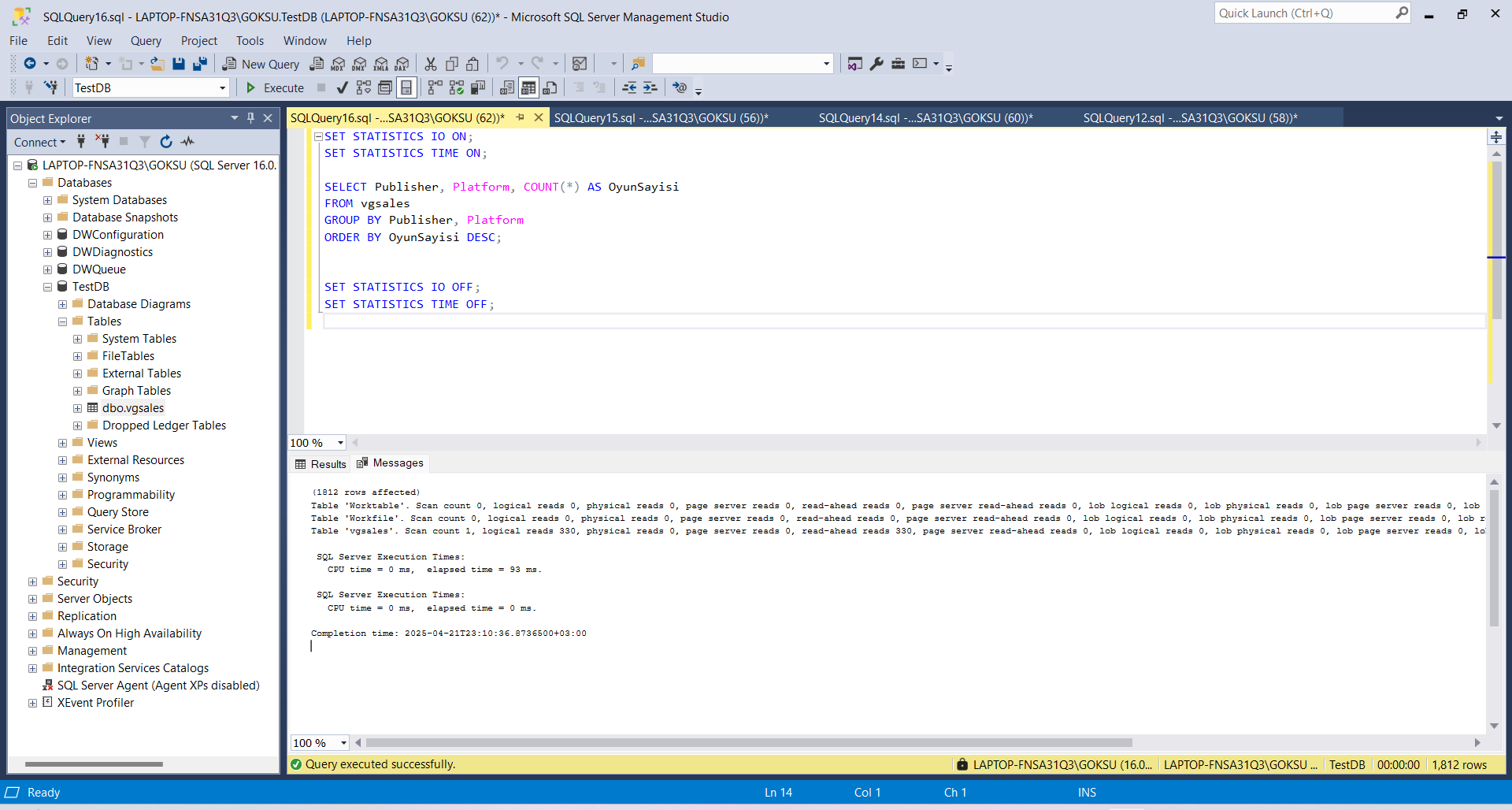


Figure 5 Index kullanmadan önceki hız

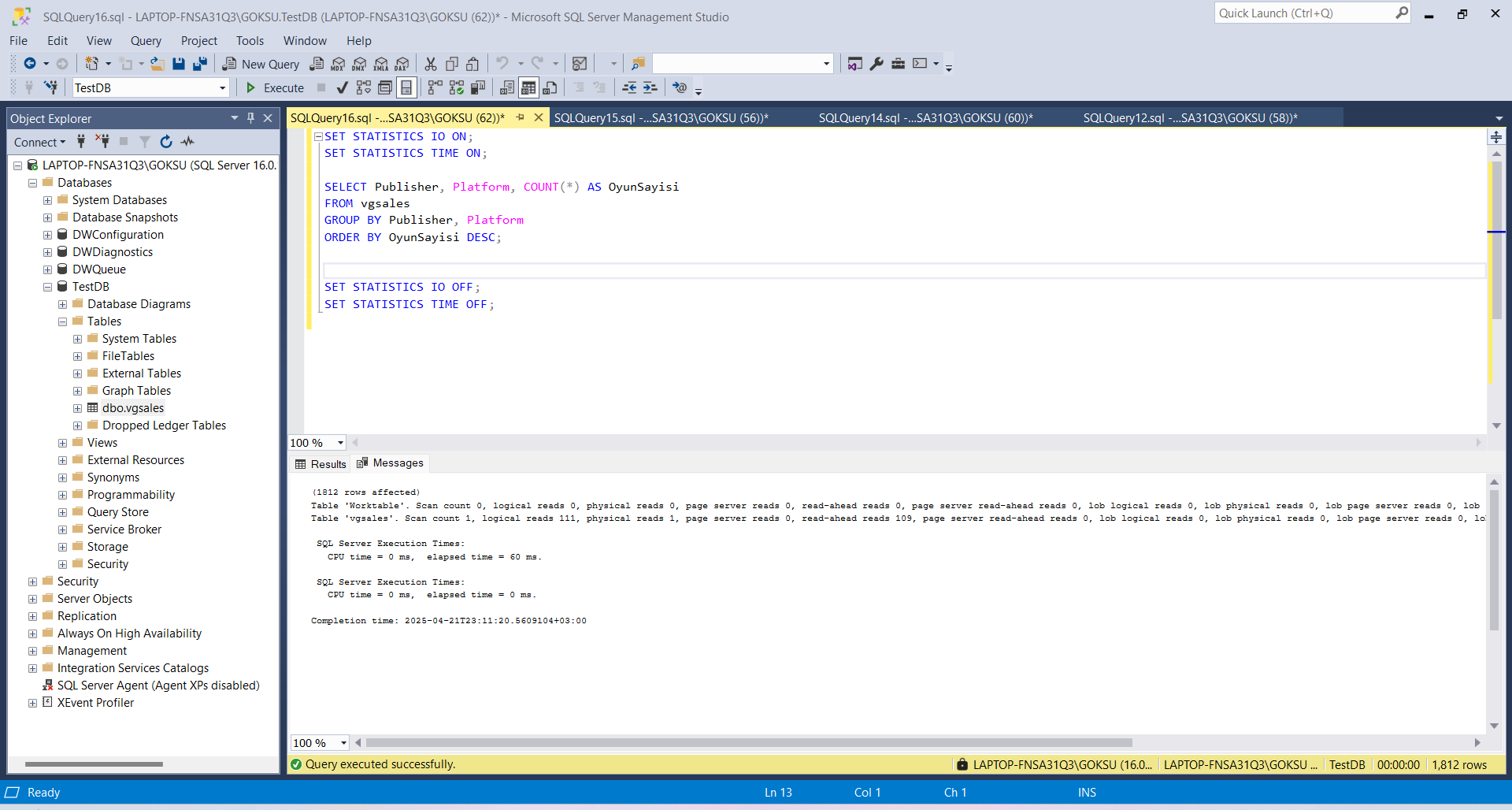


Figure 6: Indexten sonraki hız

# PROJE 2

# Veritabanı Yedekleme ve Felaketten Kurtarma Planı

Bu projenin amacı, MSSQL Server üzerinde bir veritabanının yedekleme stratejilerini belirlemek, düzenli yedekleme işlemlerini gerçekleştirmek ve olası veri kayıplarına karşı kurtarma senaryoları oluşturmaktır. Ayrıca, yedeklerin doğruluğunu test ederek felaket durumlarında verinin geri kazanılabilirliğini güvence altına almak hedeflenmiştir.

| **Araç / Özellik** | **Açıklama** |
| --- | --- |
| SQL Server Backup | Veritabanının yedeğini almak için |
| SQL Server Agent | Otomatik yedekleme zamanlama aracı |
| T-SQL Scripting | Yedekleme ve geri yükleme komutları |
| SSMS GUI | Görsel arayüz ile işlemleri takip |

## Aşama 1: Tam Yedekleme (Full Backup)

SSMS arayüzü aracığılıyla full back up dosyası alındı (Recordings kısmına eklendi).

Tasks > Back Up daha sonrasında çıkan pencerede Back Up type Full olarak seçildi ve Back up component Database seçili olarak .bak uzantılı back up dosyası alındı.

## Aşama 2: Artık (Transaction Log) ve Fark (Differential) Yedekleme

Bu adımda, Full yedeğe ek olarak:

* **Fark yedeği**: Full yedekten sonra değişen tüm verileri içerir
* **Artık (Log) yedeği**: Her işlem sonrası yazılan transaction log’ları içerir (point-in-time restore için kullanılır)

Bu işlemler için veritabanı **FULL Recovery Model**'de olmalıdır. Bu yüzden system Full Recovery Model’e alındı.

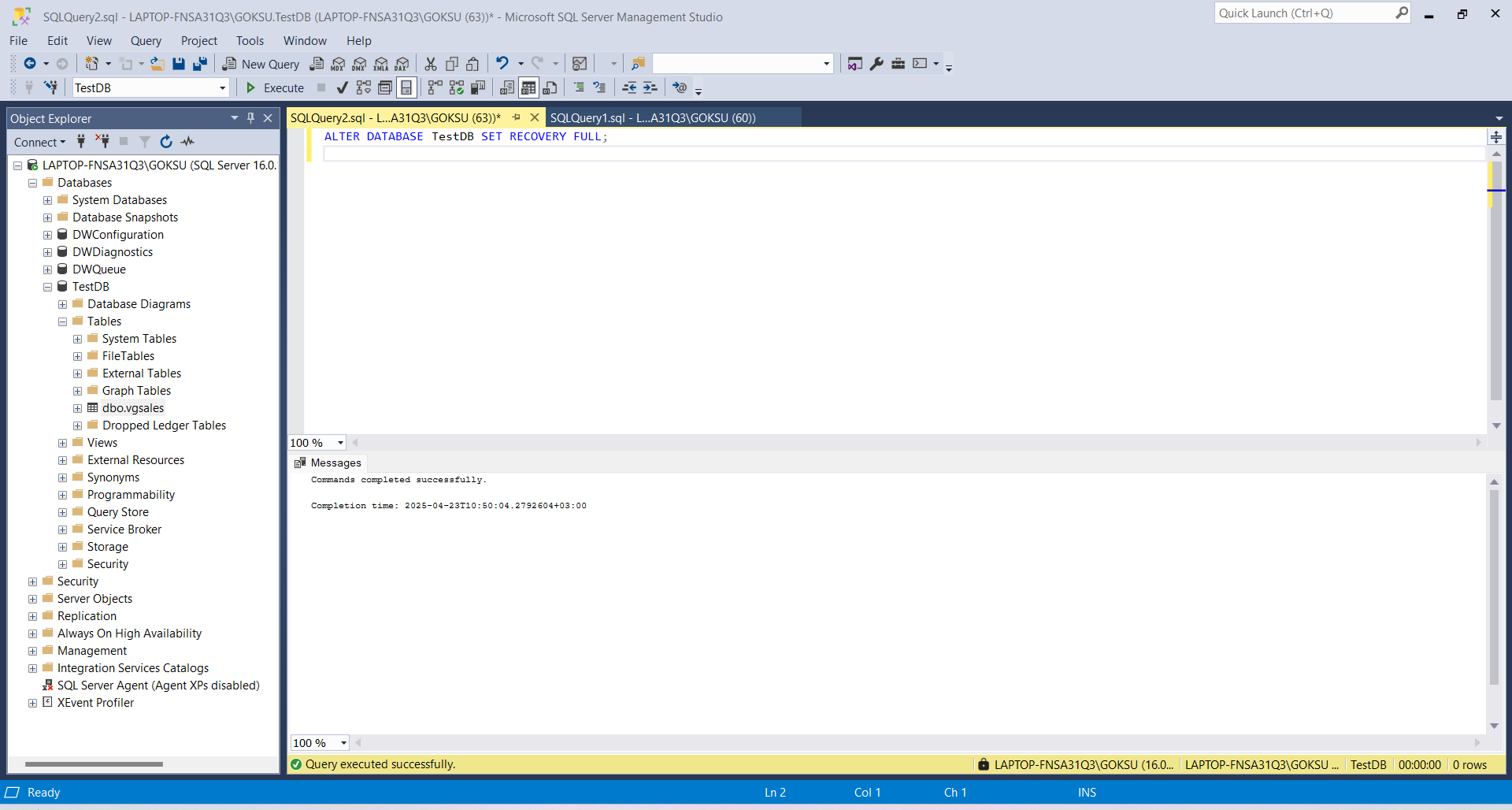


Figure 7 Full Recovery Modele alma scripti

## Aşama 3: Fark (Differential) Yedekleme (GUI Üzerinden)

Yine database üzerinden sağa tıklayarak Back Up sekmesini açıyoruz çıkan pencerede Figure 8 de belirtilen ekrandaki seçenekleri kullanarak Differential back up dosyasını verilen klasörde .bak uzantısı ile oluşturuyoruz.

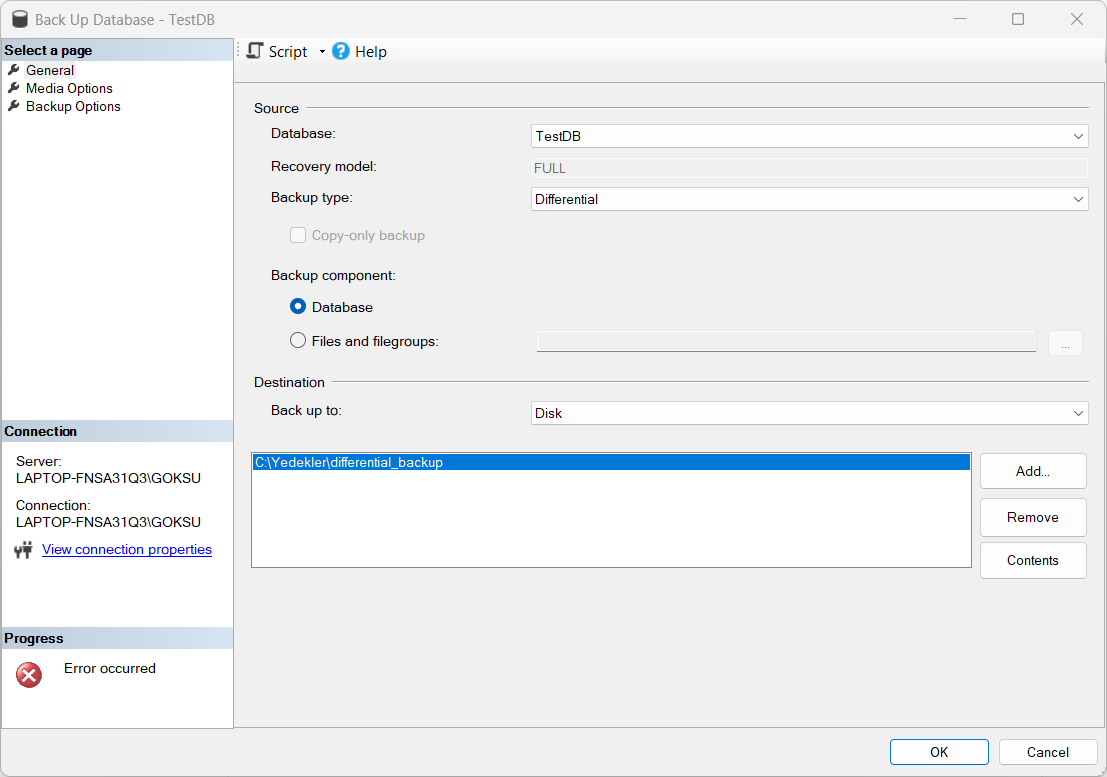


Figure 8 Differential Back up penceresi

## Aşama 4: (Log) Yedekleme (GUI Üzerinden)

Yine aynı şekilde Graphic User Interface üzerinden database’i seçip back up sekmesine bağlanıp Back up type’ ı transaction olarak belirledik.

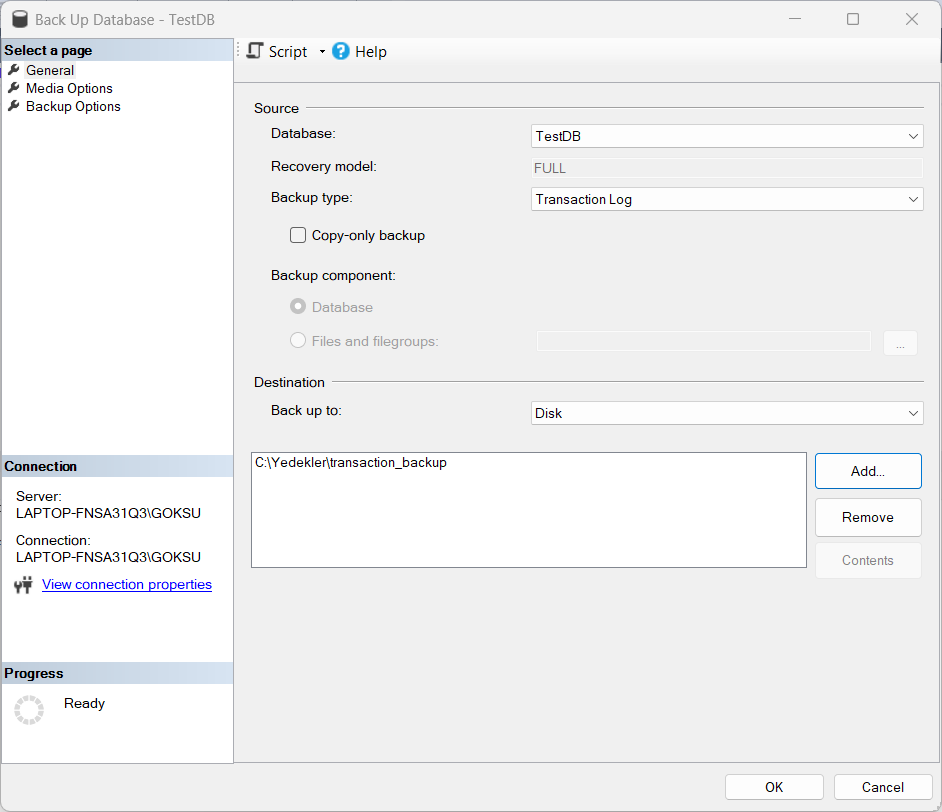


Figure 9 Transaction back up

| **Yedek Türü** | **İçeriği** | **Ne zaman kullanılır?** |
| --- | --- | --- |
| Full | Tüm veritabanı | Haftalık / ana yedek olarak |
| Differential | Full'den sonra değişen veriler | Günlük, hızlı geri yükleme için |
| Transaction Log | Her işlemi tek tek kaydeder | Felaketten sonra “noktaya geri dönüş” |

Grafik arayüzü kullanarak restore back up komutunu çalıştırırken hata aldım bu yüzden T-SQL ile kullanıcıları kapatıp tek Kullanıcı ile daha önce yedekledigim database’den restore komutunu uyguladım.

## Aşama 5: Otomatik T-SQL ile Kullanıcıları Kapatarak Geri Yükleme

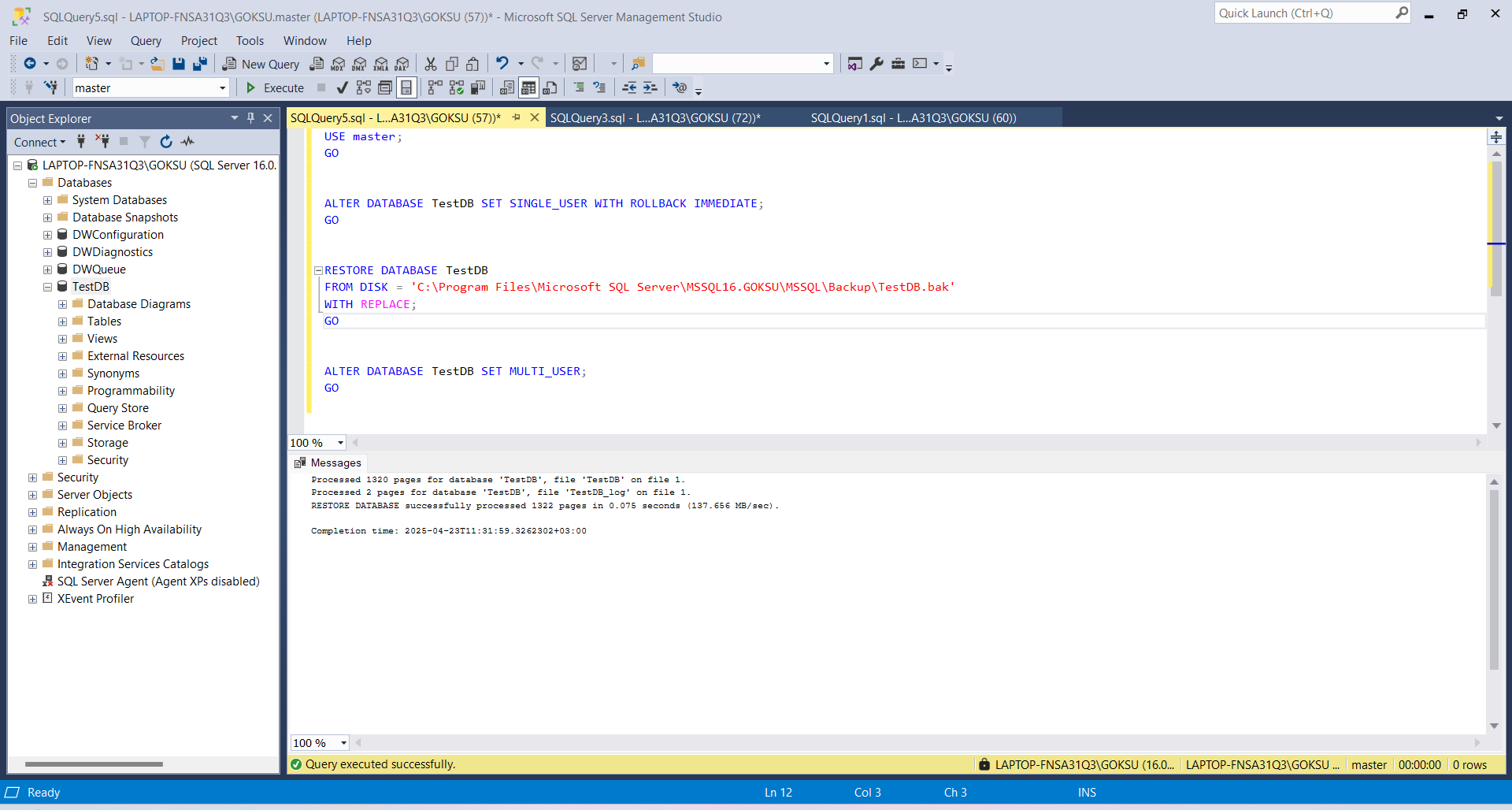


Figure 10 T-SQL restore back up

Restore işlemini göstermek için önce

“DELETE FROM vgsales

WHERE Genre = 'Sports'; “ sorgusu ile Sports tipli verileri silip daha sonrasında ise restore back up ile geri getirdik.

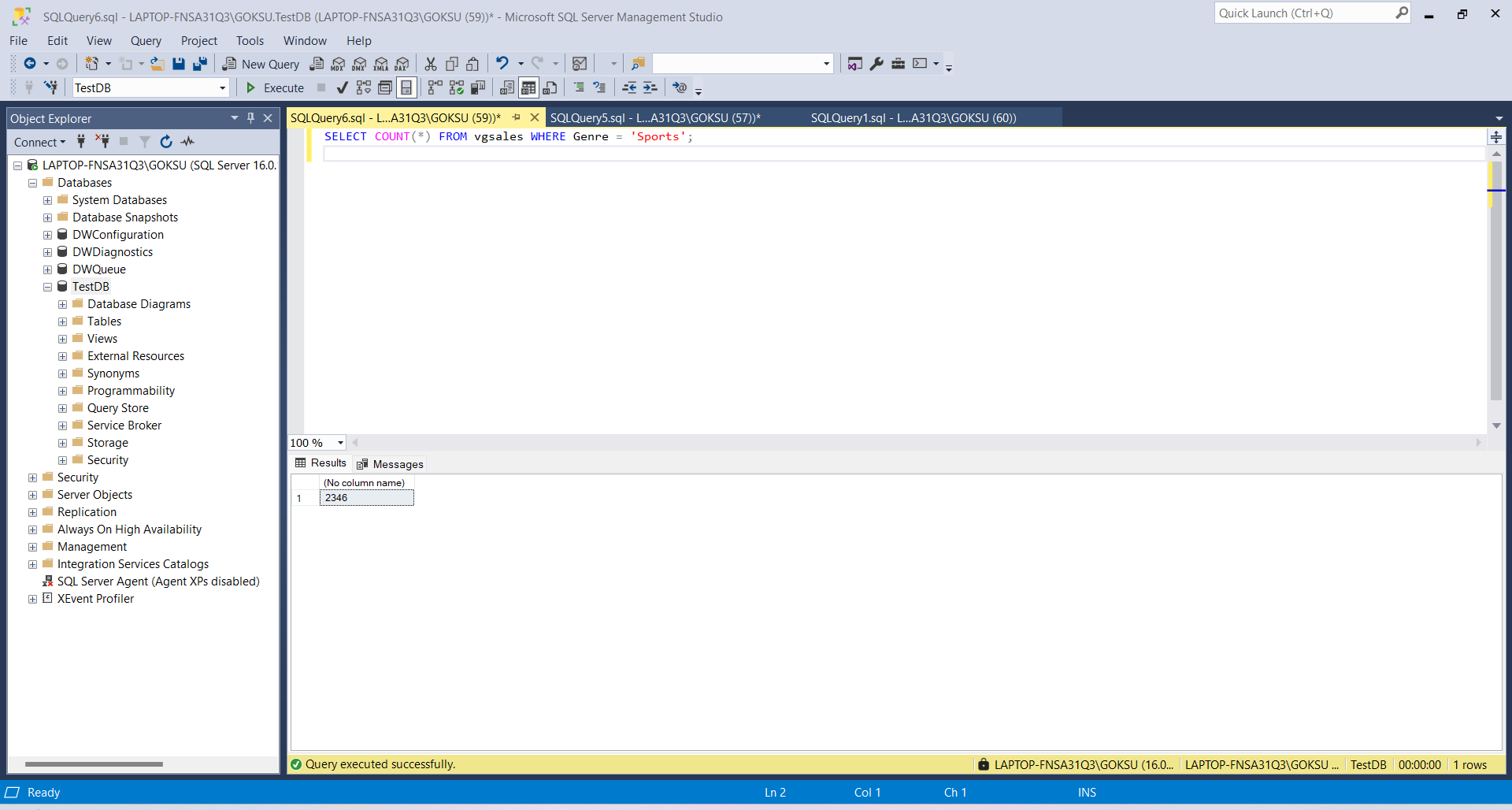
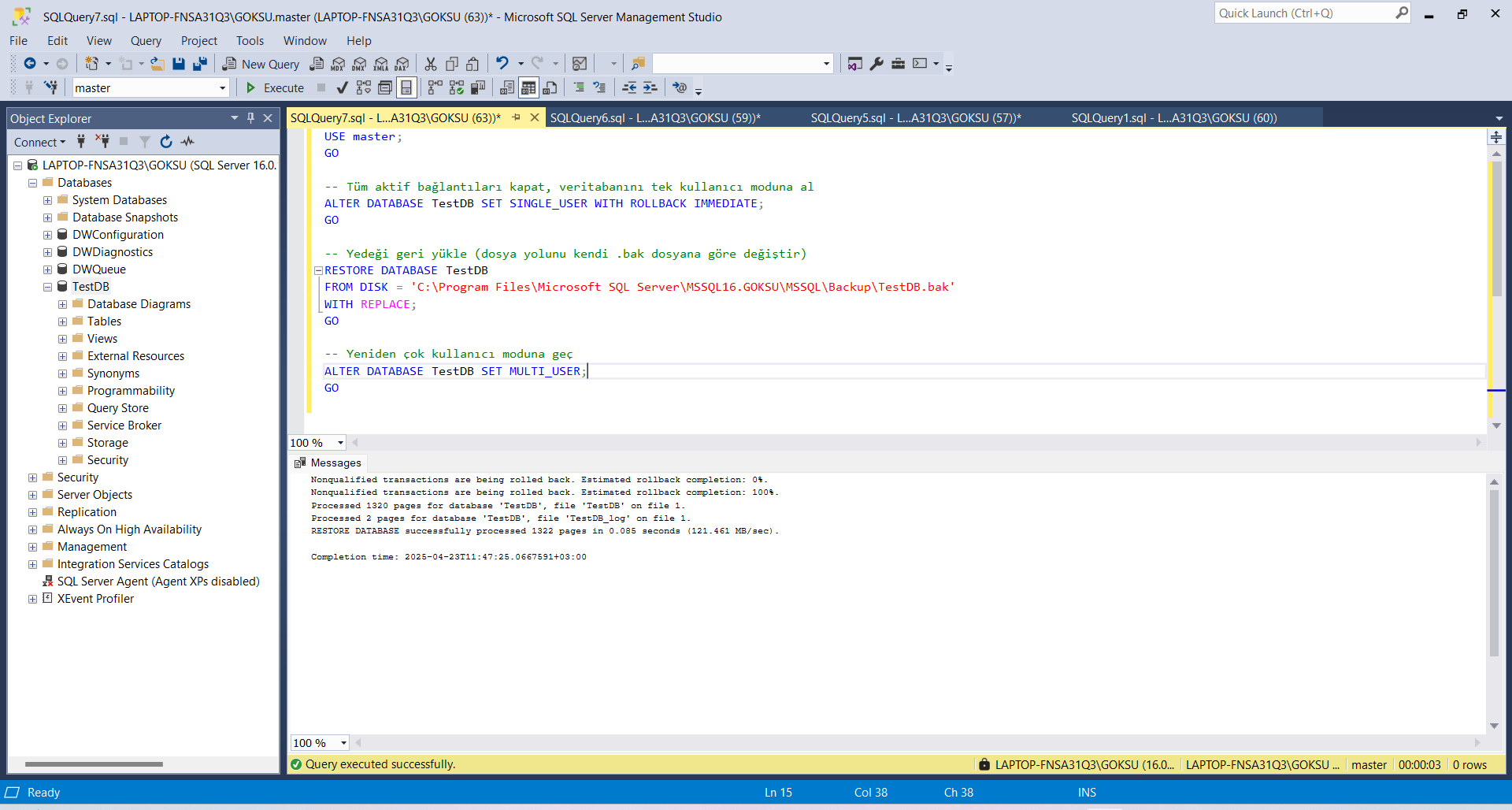


Figure 11 Restore işleminden sonra Sports tipindeki oyunların sayısını gösteren sorgu

## Aşama 6: Test Yedekleme Senaryoları – Yedeklerin Doğruluğunu Test Etme

Bu adımda yedek dosyasının gerçekten **çalışır durumda olduğunu**, geri yüklenebildiğini ve içinde doğru veri olduğunu test edeceğiz.

Bu aşamada alınan yedeklerin bütünlüğü test edilmiştir. Geri yükleme işlemi mevcut veritabanı üzerine değil, aynı yedekten yeni bir veritabanı oluşturularak gerçekleştirilmiştir. Böylece orijinal veritabanına zarar vermeden yedekleme dosyasının geçerli olup olmadığı test edilmiştir.



Bu aşamada, yanlışlıkla silinen verilerin geri getirilebilmesi için daha önce alınan tam yedek (full backup) dosyası kullanılarak `TestDB` veritabanı geri yüklenmiştir.

Geri yükleme sırasında veritabanının başka bağlantılar tarafından kullanılması nedeniyle "Exclusive access could not be obtained..." hatası alınmış, bu durum SQL Server'da sık karşılaşılan bir durumdur.

Sorunun çözümü olarak veritabanı tek kullanıcı moduna alınmış, tüm bağlantılar sonlandırılmış ve `WITH REPLACE` komutu kullanılarak yedek başarıyla geri yüklenmiştir.

Bu işlem sonucunda `Genre = 'Sports'` olan, daha önce silinmiş veriler yeniden erişilebilir hâle getirilmiştir.

Bu adımda, alınan yedeğin geçerliliği test edilmiştir. Mevcut veritabanı üzerine yazmadan, aynı yedek dosyası kullanılarak `TestDB\_YedekKontrol` adında yeni bir veritabanı oluşturulmuştur.

Yapılan sorgular sonucunda daha önce silinen verilerin bu yeni veritabanında hâlâ mevcut olduğu görülmüş ve yedeğin başarılı şekilde geri yüklenebilir olduğu doğrulanmıştır.

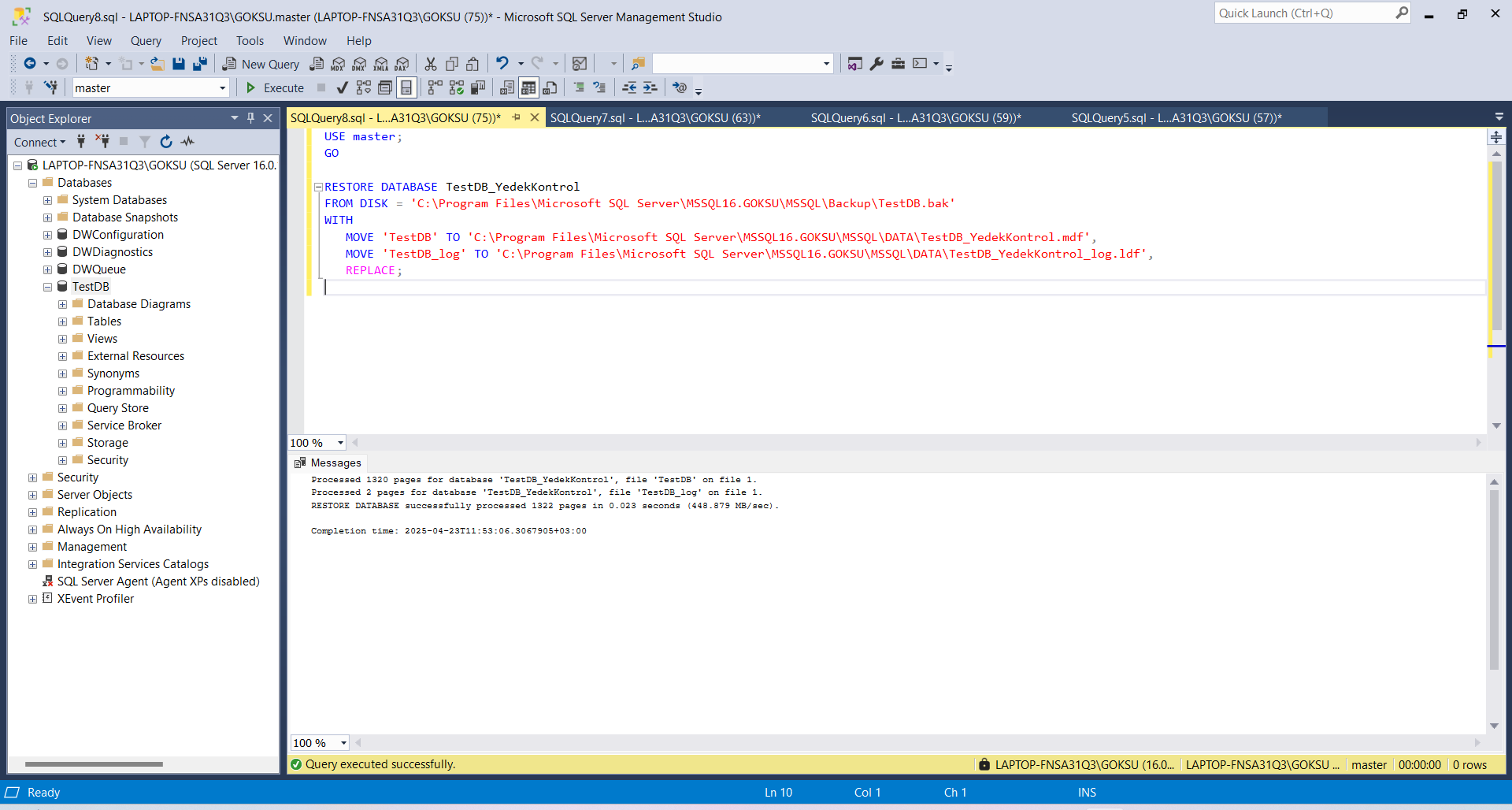


Figure 12 TestDB yedek control

Bu ekran görüntüsü, daha önce alınan `TestDB.bak` yedeğinin T-SQL komutları kullanılarak `TestDB\_YedekKontrol` adında yeni bir veritabanı olarak başarıyla geri yüklendiğini göstermektedir.

Bu işlem, yedeğin geçerli ve kullanılabilir olduğunu doğrulamak için yapılmıştır. Böylece orijinal veritabanına zarar vermeden yedek dosyasının bütünlüğü test edilmiştir.

# PROJE 3

# Veritabanı Güvenliği ve Erişim Kontrolü

Bu projenin amacı, SQL Server veritabanı üzerinde kullanıcı erişim kontrolü, hassas verilerin korunması, veri şifreleme tekniklerinin uygulanması ve kötü amaçlı sorgulara (SQL Injection) karşı güvenlik önlemlerinin test edilmesidir. Aynı zamanda, kullanıcı aktivitelerinin kaydedilmesi için denetim (audit) mekanizmaları uygulanmıştır.

| **Adım No** | **Aşama Başlığı** | **Konu** |
| --- | --- | --- |
| 1 | Kullanıcı Hesabı ve Erişim Yönetimi | SQL Server & Windows Auth |
| 2 | Veri Şifreleme (TDE) | Transparent Data Encryption |
| 3 | SQL Injection Saldırısı Simülasyonu | Güvenlik açığı testi |
| 4 | Kullanıcı Aktivitelerini İzleme | SQL Server Audit ile loglama |

## Aşama 1 Amaç: Yeni kullanıcı oluşturmak ve sadece belirli tablolara/verilere erişmesini sağlamak

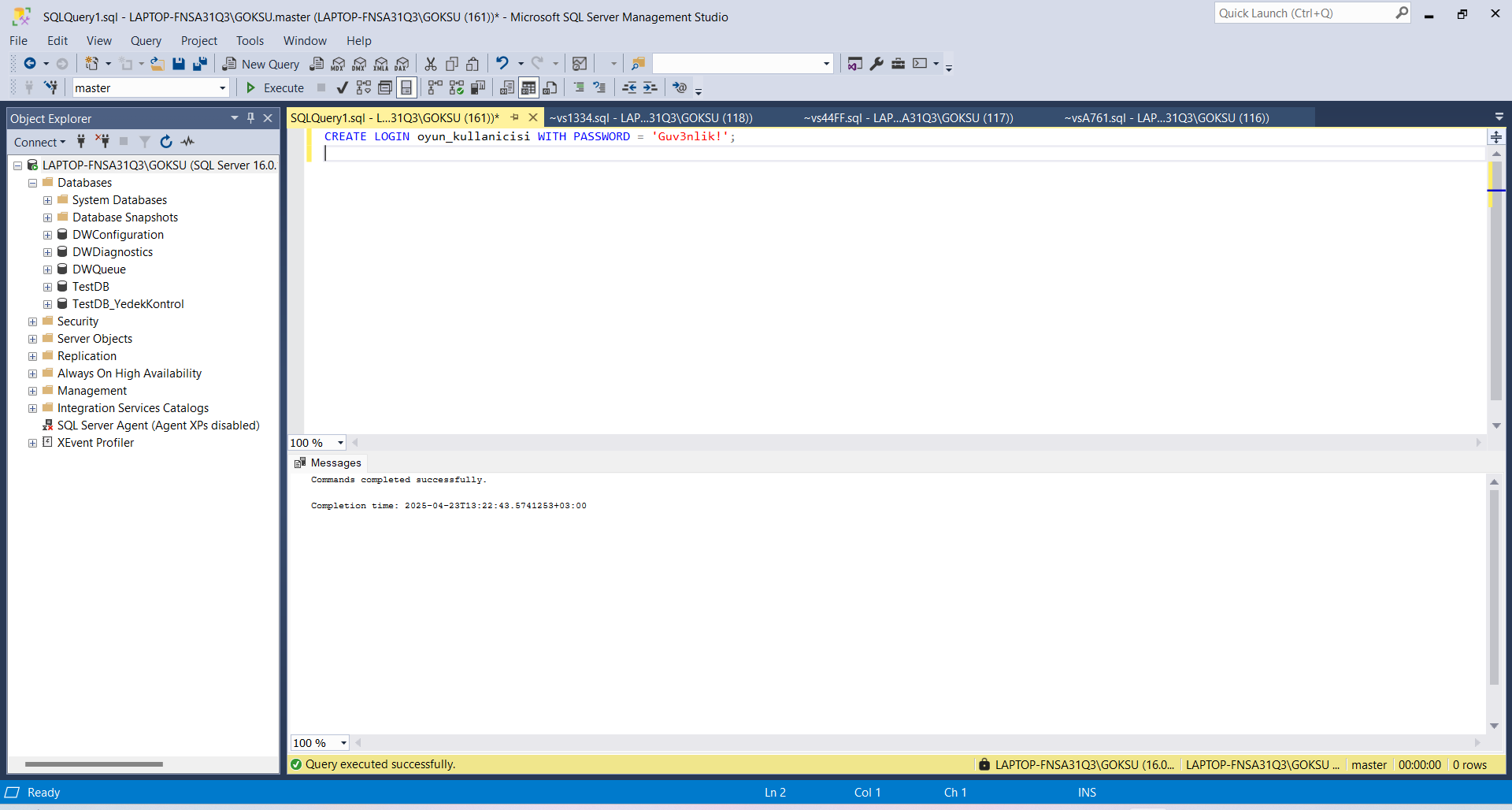


Figure 13 Login Oluştur (SQL Server seviyesi)

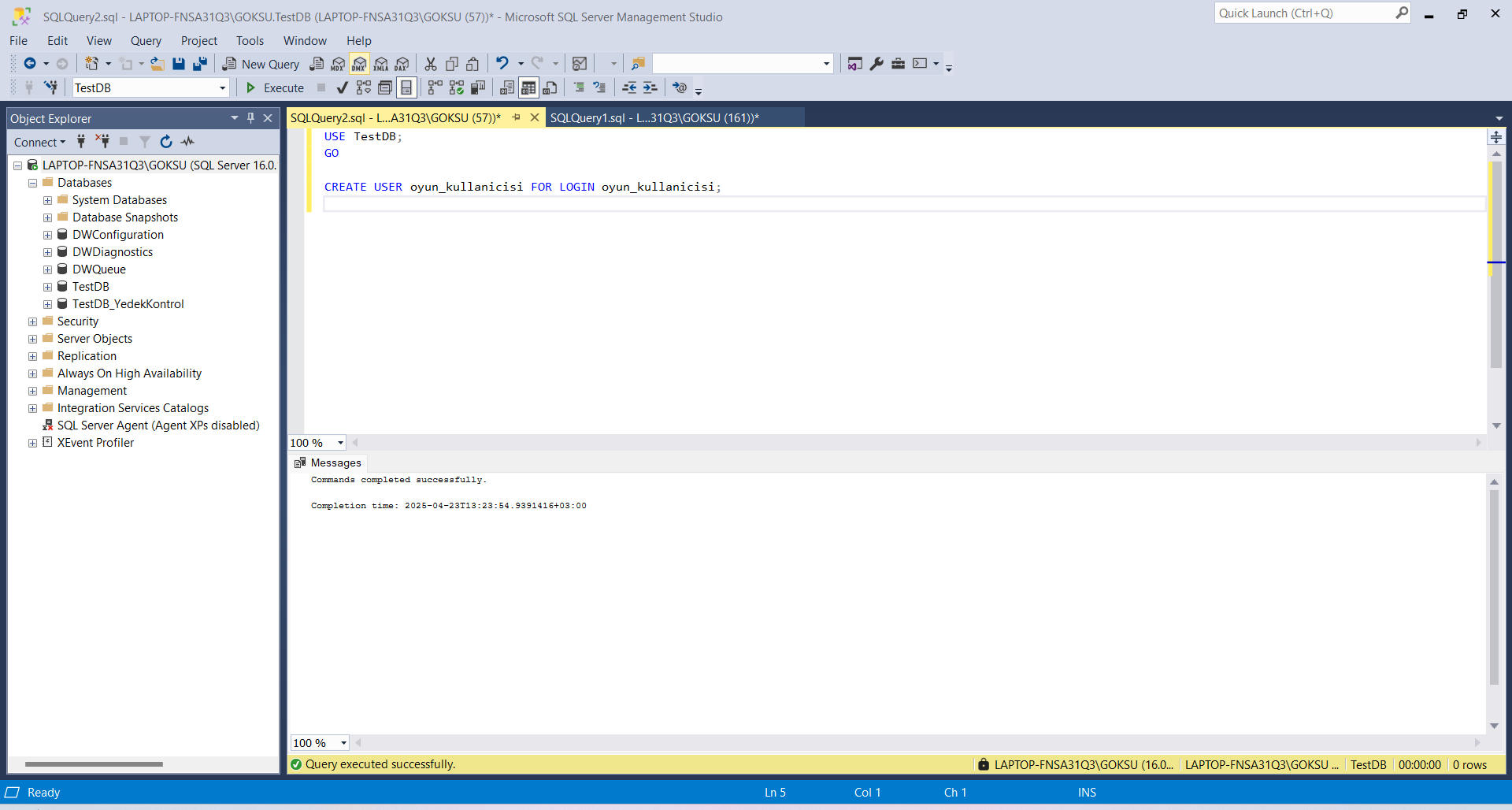


Figure 14 Login’i TestDB veritabanında kullanıcıya dönüştür

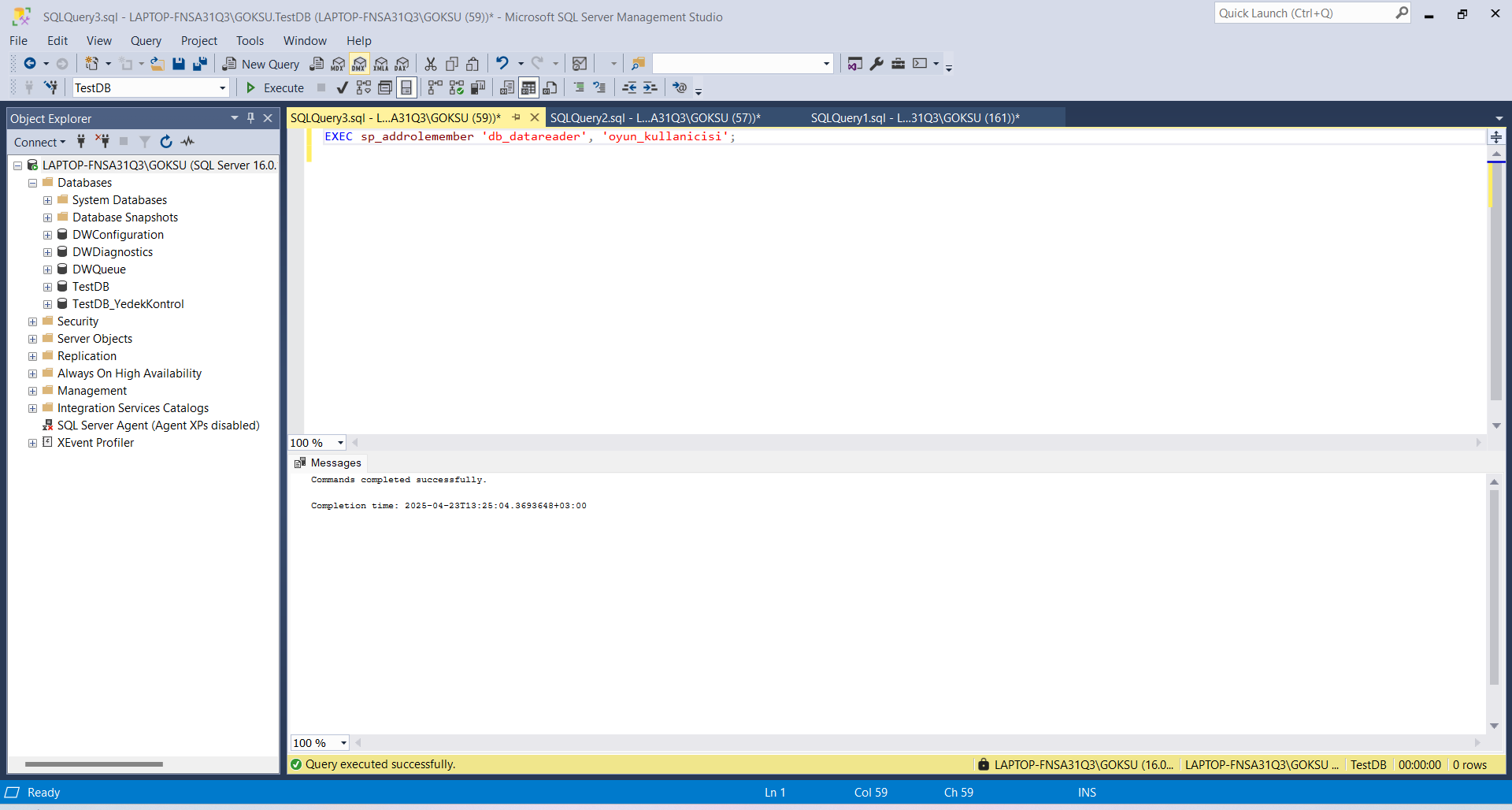


Figure 15 Sadece okuma yetkisi verme

Bu adımda, SQL Server üzerinde `oyun\_kullanicisi` adında bir kullanıcı oluşturulmuş ve bu kullanıcıya sadece veri okuma yetkisi verilmiştir. Kullanıcının veritabanı üzerinde izinsiz silme ve yazma işlemleri yapamayacağı test edilmiştir. Böylece temel erişim kontrolü sağlanmıştır.

Aşama 2: Veri Şifreleme – TDE (Transparent Data Encryption**)**

🎯 ***Amaç:*** Veritabanındaki verilerin disk seviyesinde şifrelenmesini sağlamak. Böylece yetkisiz biri .mdf veya .bak dosyasını ele geçirse bile içeriği okuyamaz.

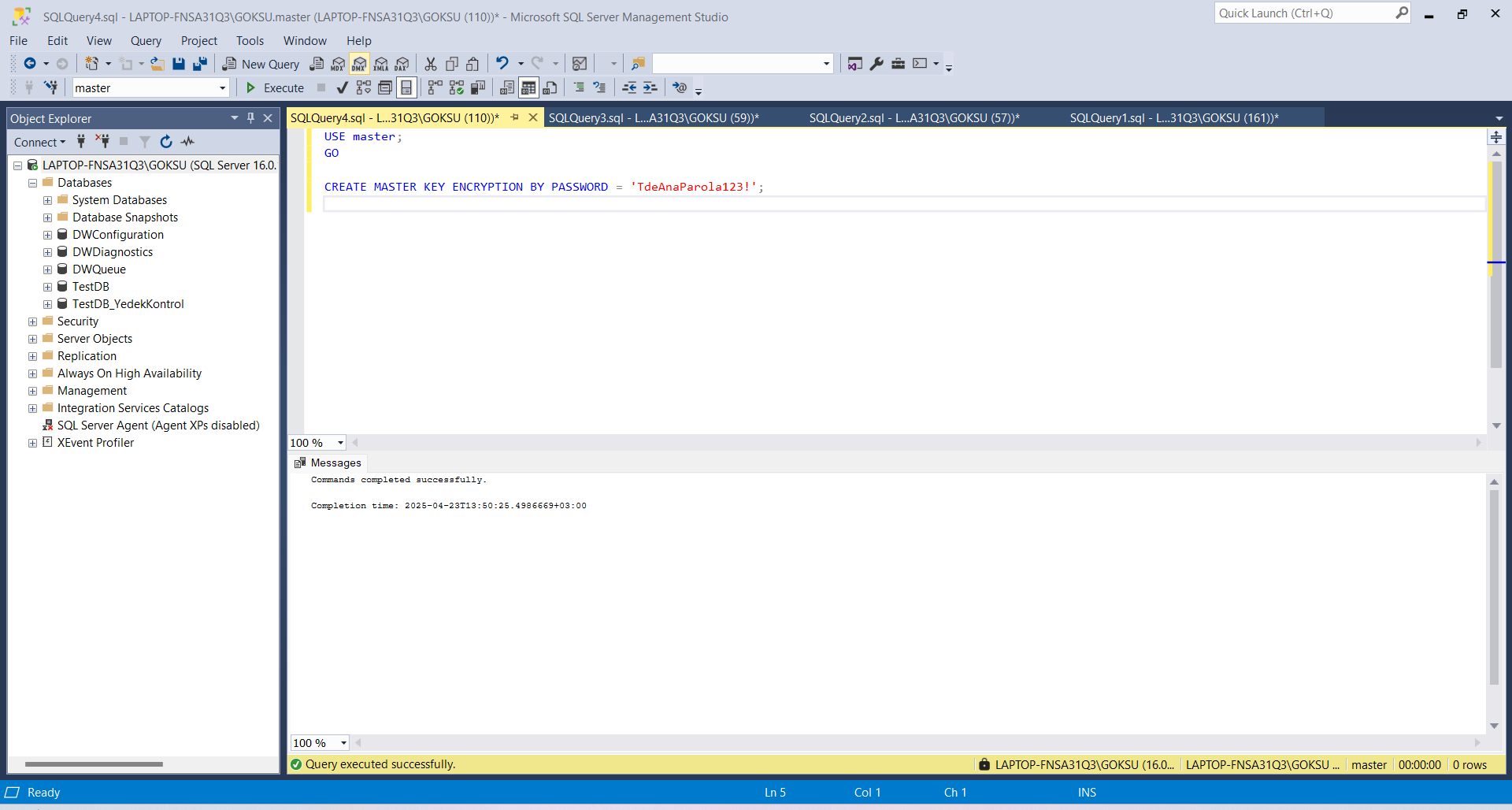


Figure 16 Sunucu ana anahtarı olusturma

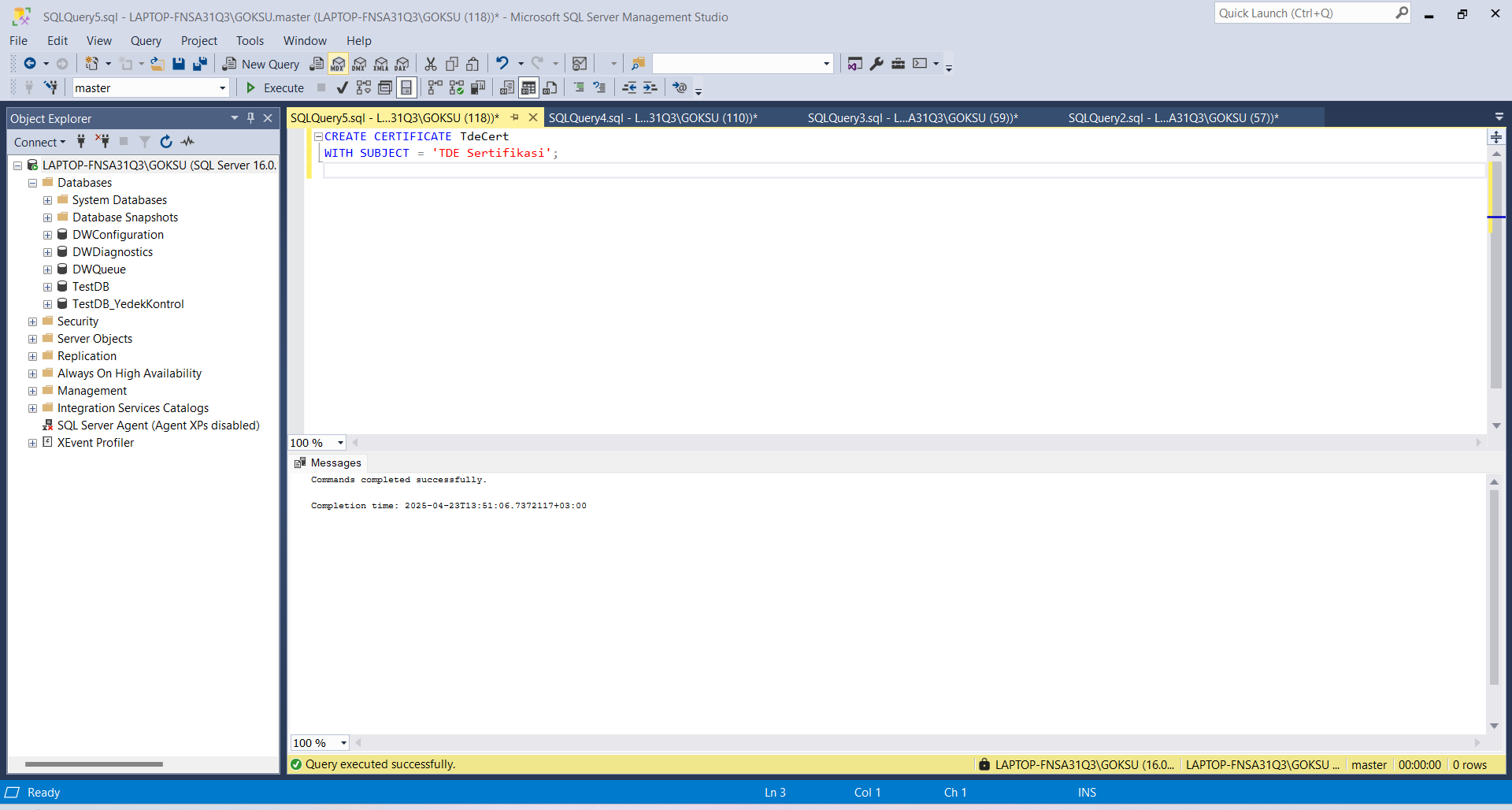


Figure 17 tde sertifika oluşturma

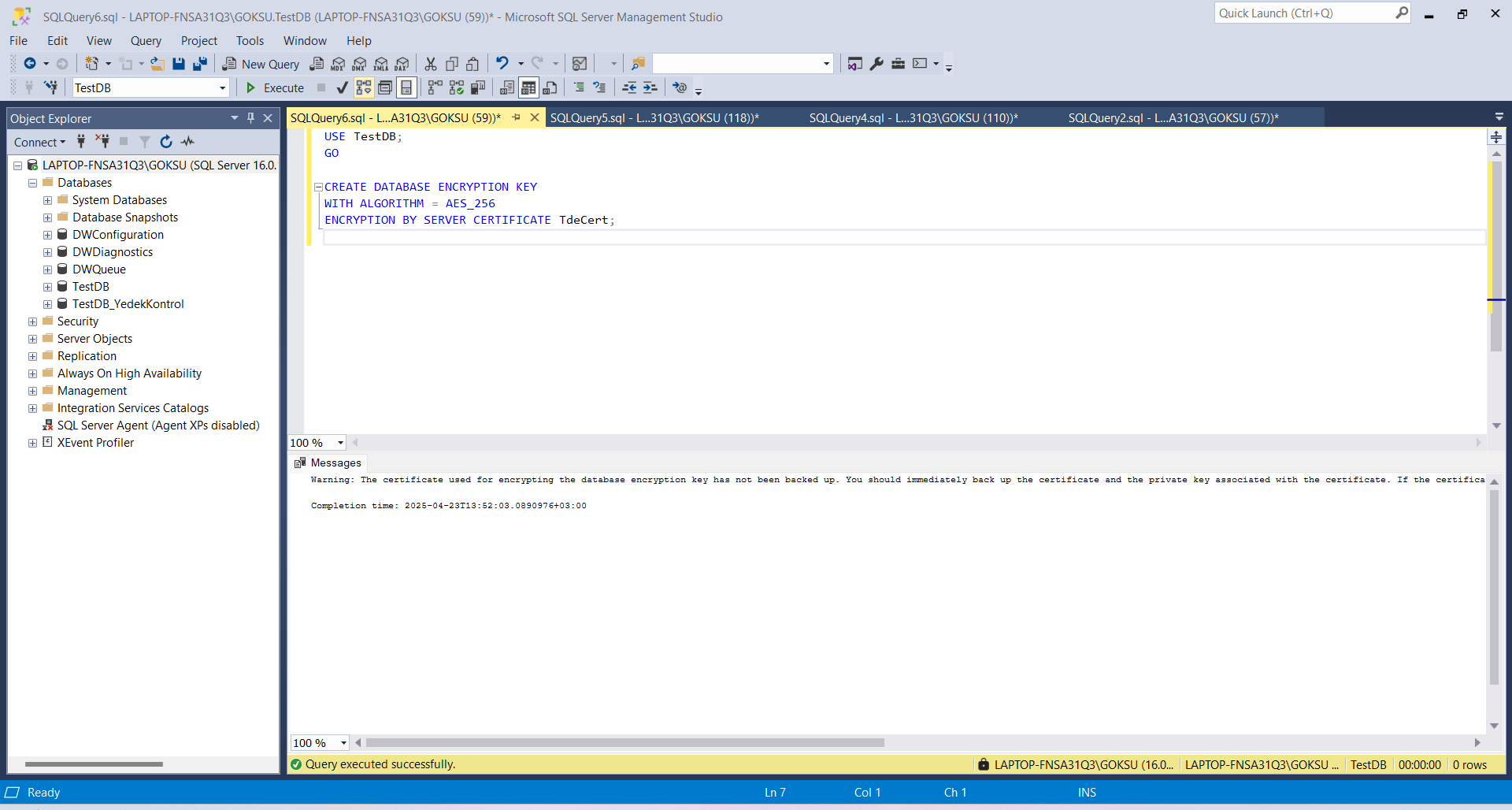


Figure 18 Şifreleme için veritabanı anahtarı

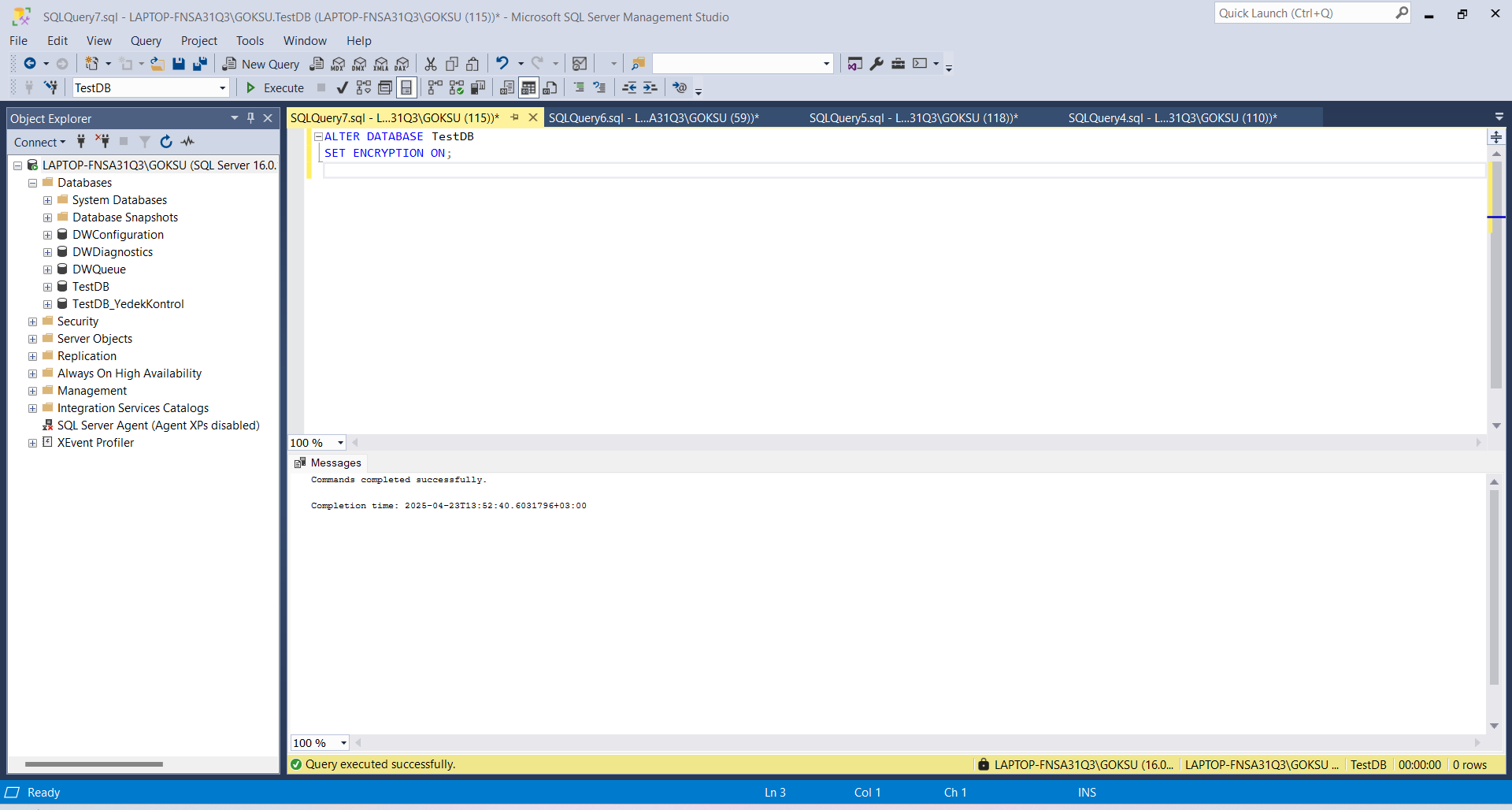


Figure 19 Şifrelemeyi aktif etme

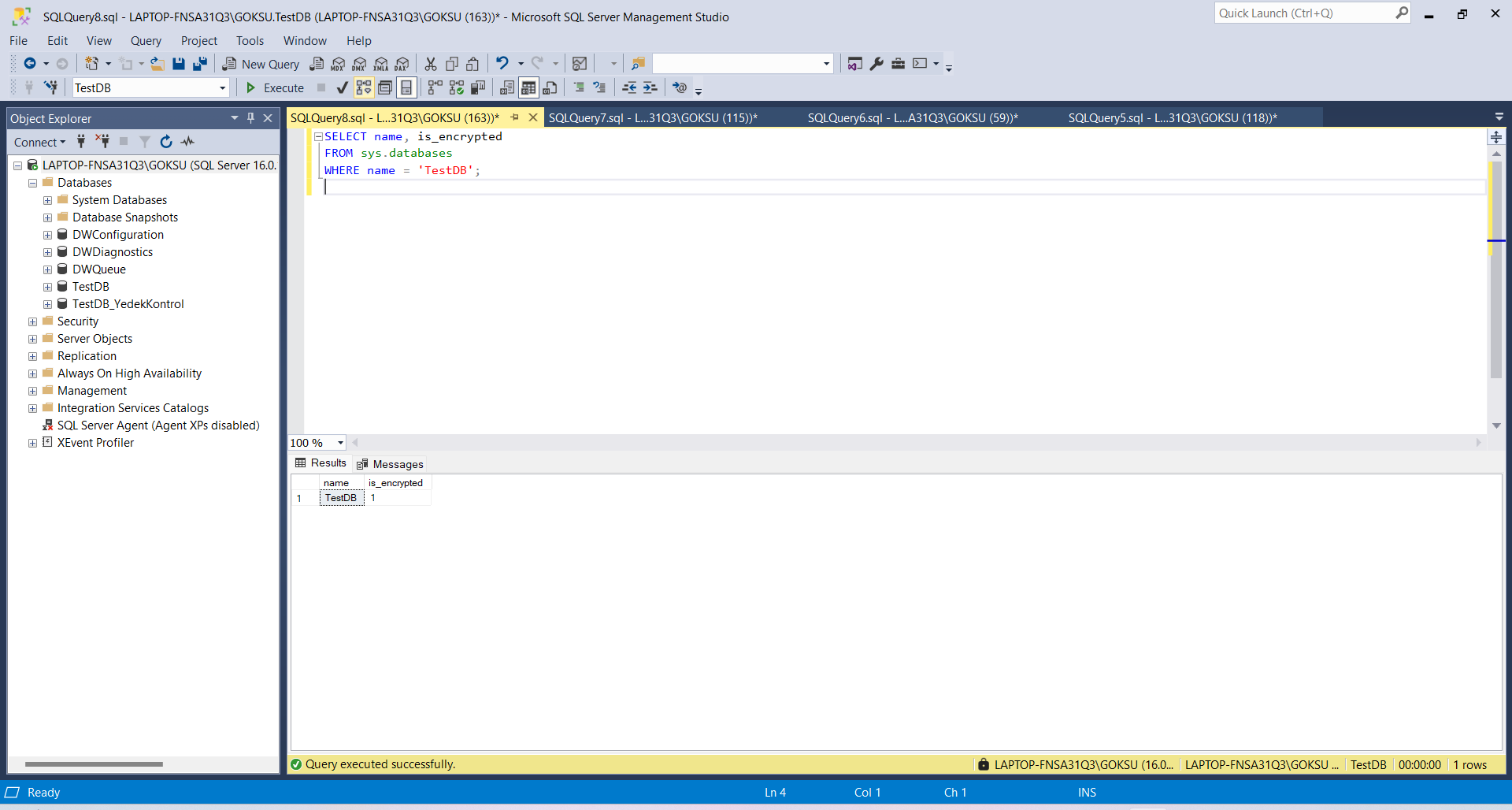


Figure 20 Şifreleme durum

Bu adımda SQL Server üzerinde Transparent Data Encryption (TDE) özelliği etkinleştirilmiştir. Gerekli anahtar ve sertifikalar oluşturularak `TestDB` veritabanı disk seviyesinde şifrelenmiştir. Bu işlem, yedek dosyaların ya da fiziksel veri dosyalarının çalınması durumunda içeriğin okunmasını engeller.

## Aşama 3: SQL Injection Testleri – Güvenlik Açıklarını Simüle Etme

🎯 *Amaç:* SQL sorgularına kötü niyetli girişler enjekte edilerek sistemin açık verip vermediğini test etmek ve bu riskleri önlemek.

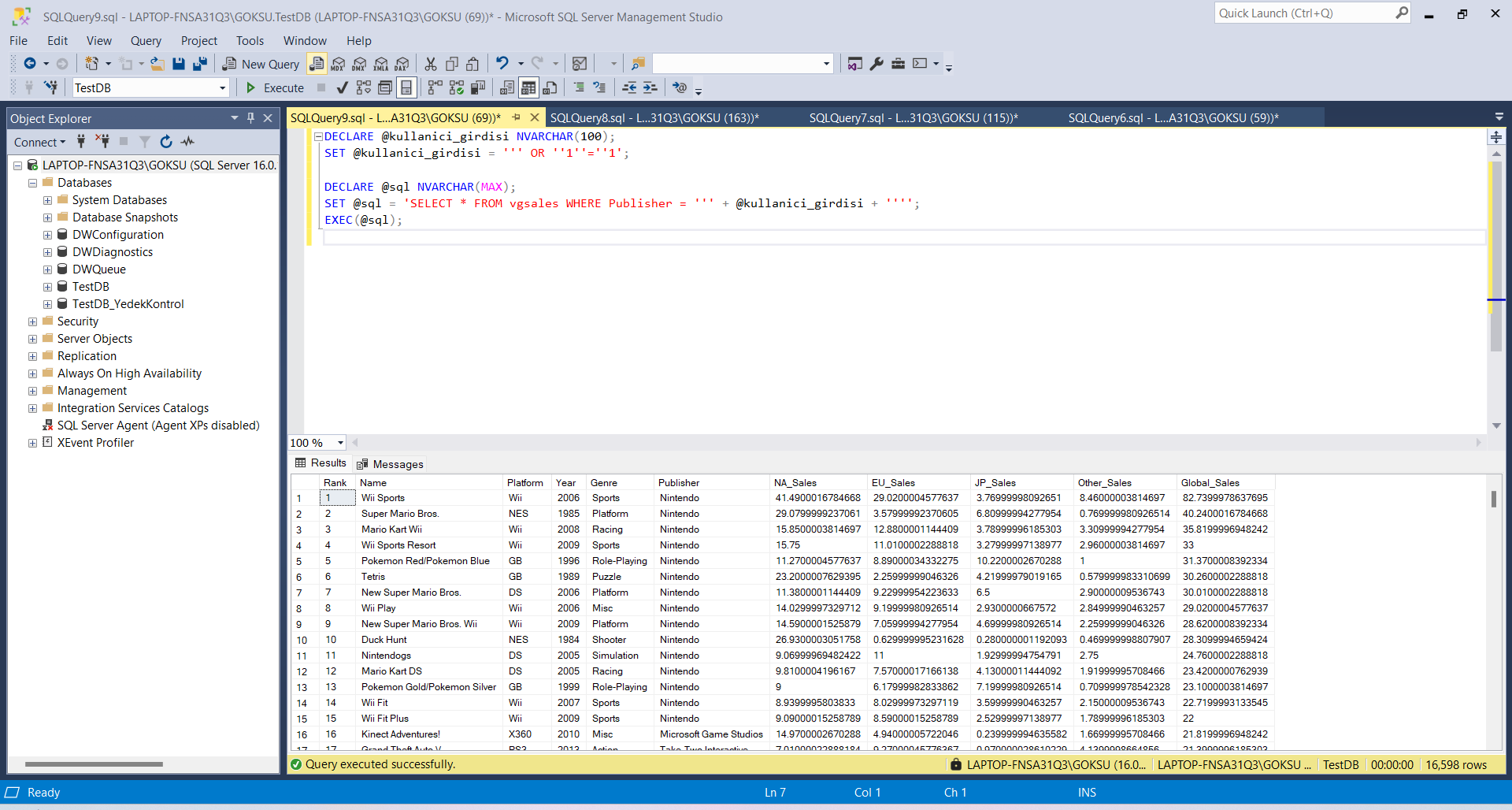


Figure 21 Savunmasız sorgu örneği

Bu giriş → ' OR '1'='1  
Tüm Publisher kayıtlarını çeker, çünkü koşul her zaman doğru olur ➤ **Veri sızdırılır.**

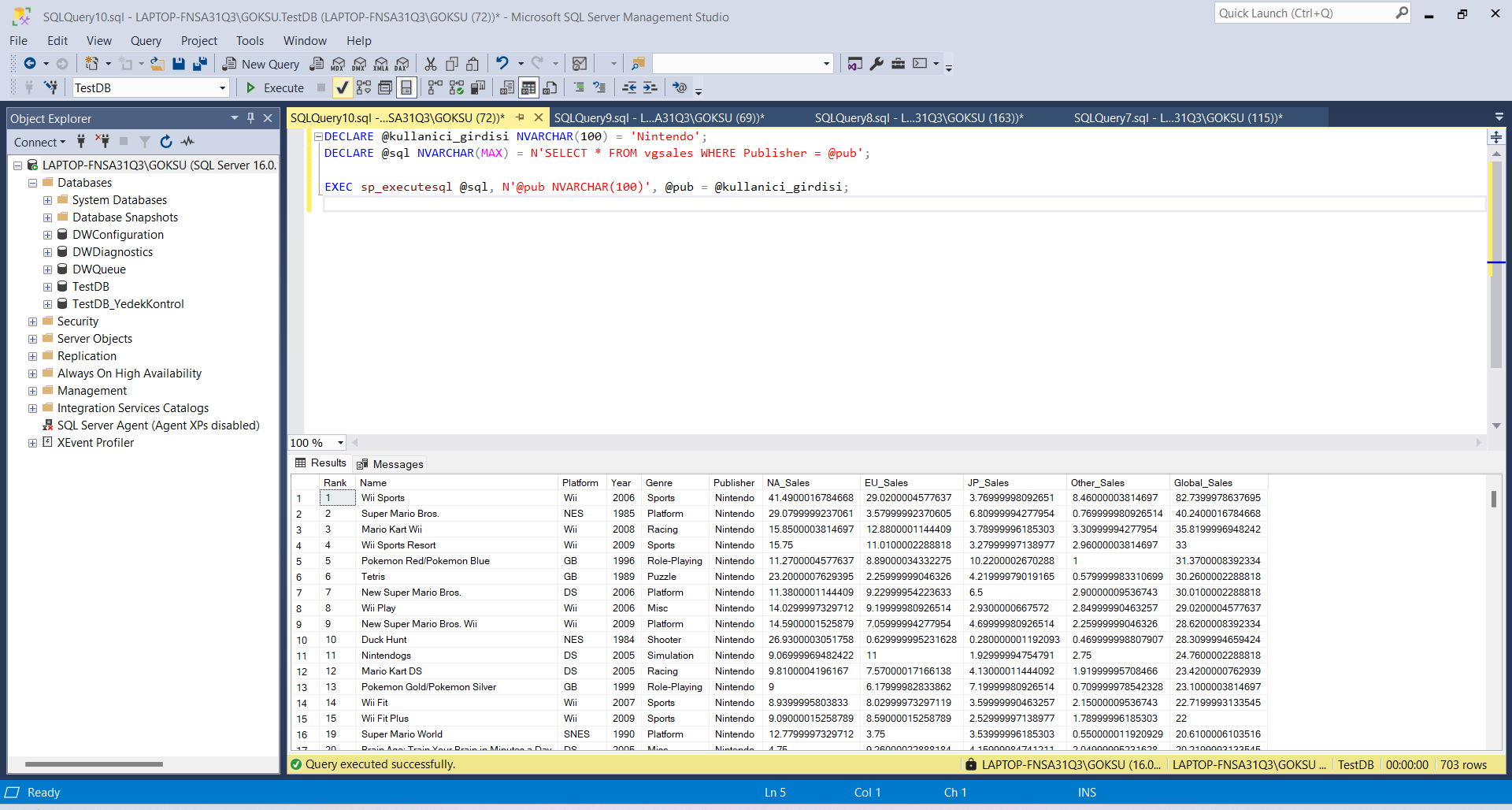


Figure 22 Parametreli Sorgu örneği

Bu yöntemle ' OR '1'='1 gibi girişler **komut olarak değil, veri olarak** algılanır → güvenlik sağlanır.

Bu adımda, SQL Injection saldırılarına karşı veritabanı savunması test edilmiştir. Kötü niyetli bir giriş ile sorgunun mantığının bozulabildiği gösterilmiştir. Ardından, parametreli sorgu kullanılarak bu tür saldırıların önüne geçilmiştir. Bu senaryo, veritabanı uygulamalarında güvenli kodlamanın ne kadar kritik olduğunu ortaya koymuştur.

## Aşama 4: Audit Loglama – Kullanıcı Aktivitelerini İzleme

🎯 *Amaç:* SQL Server'da yapılan işlemleri (login, SELECT, DELETE, UPDATE gibi) **kimin, ne zaman, hangi veritabanında yaptığını** kaydeden denetim (audit) mekanizması kurmak.

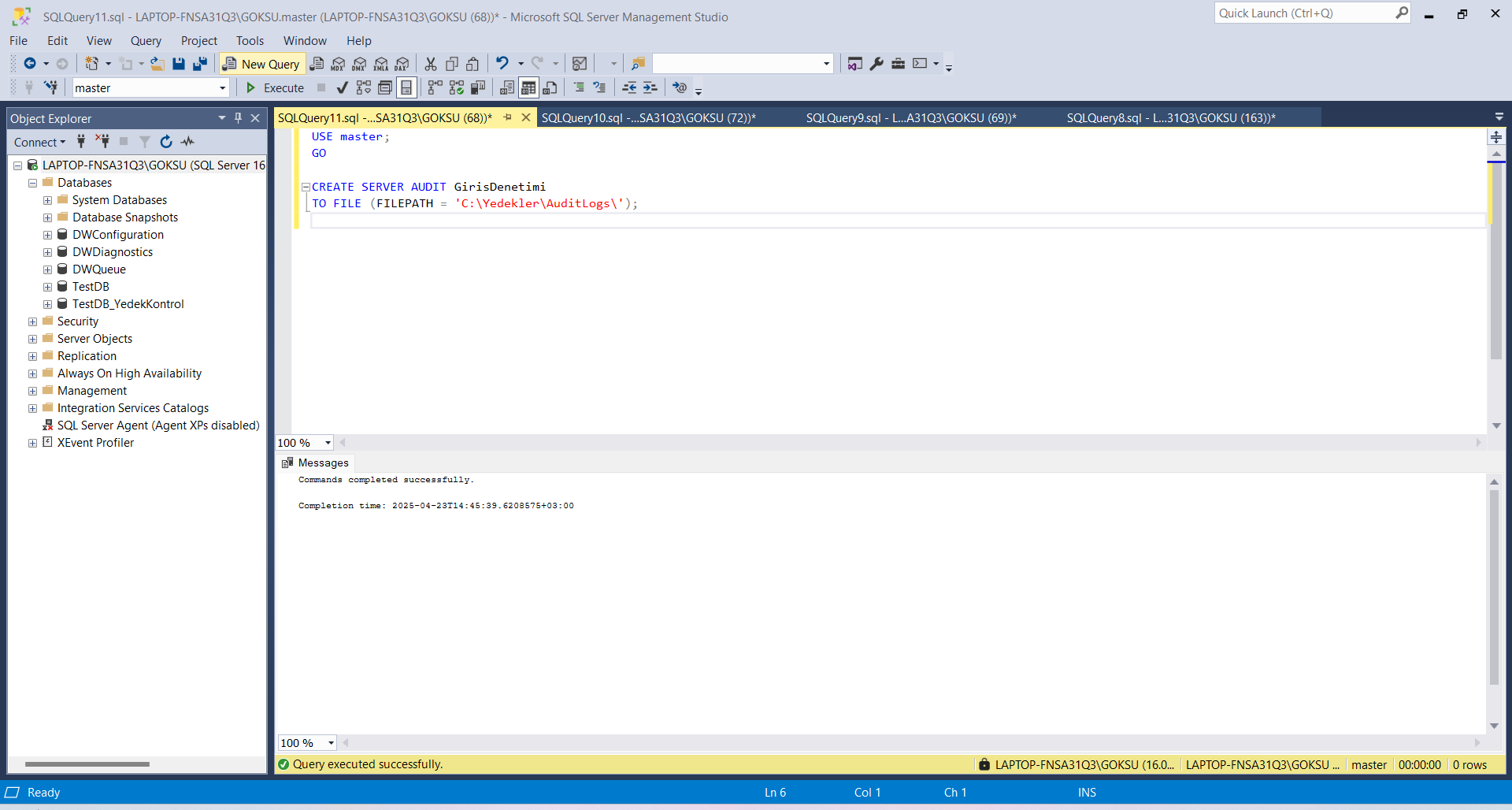


Figure 23 Audit dosyası oluştur

'C:\Yedekler\AuditLogs\' lokasyonunda ilgili klasör oluşturulde ve SQL Server için yazma yetkisi verildi.

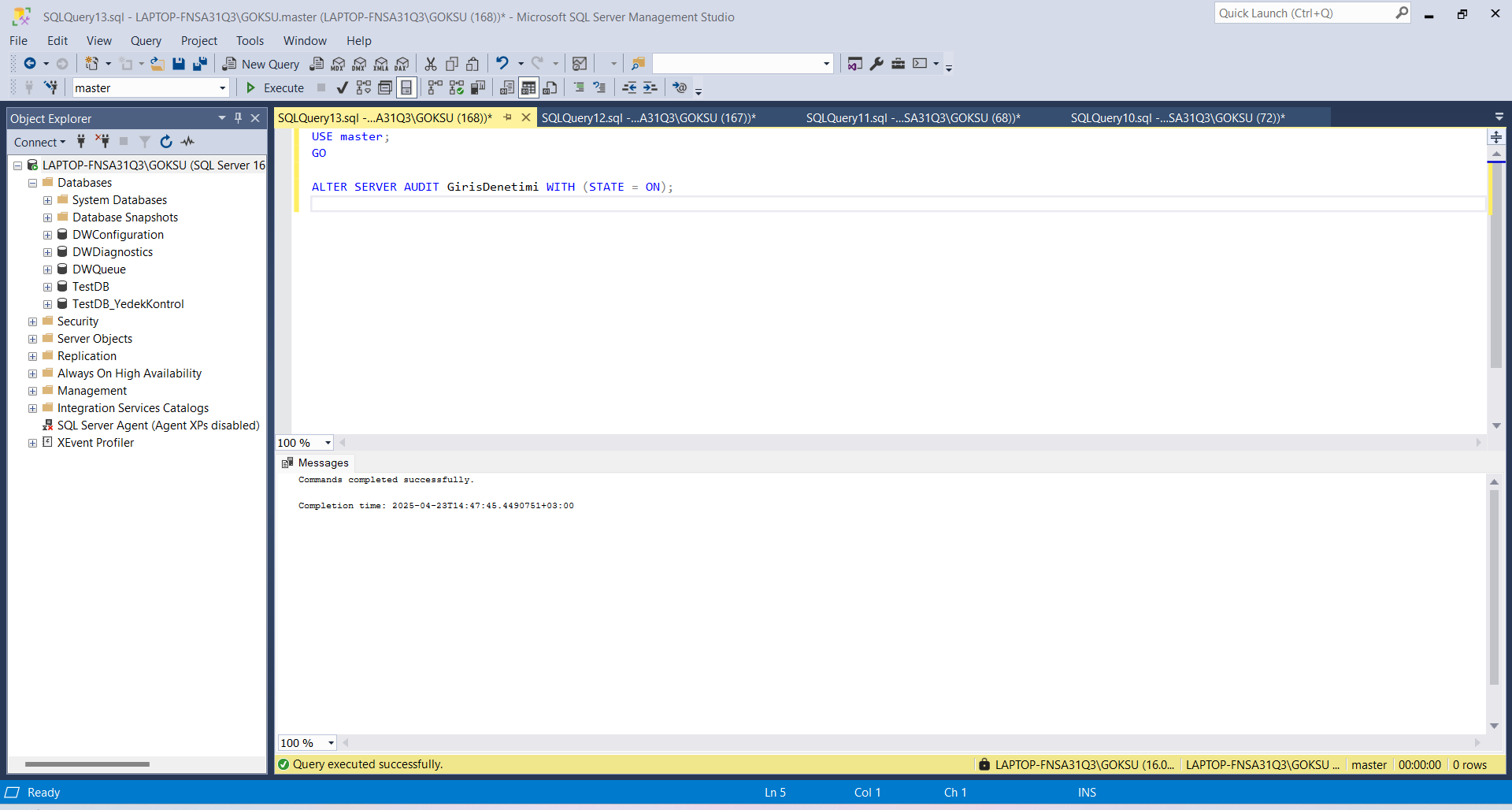


Figure 24 Audit aktive etme

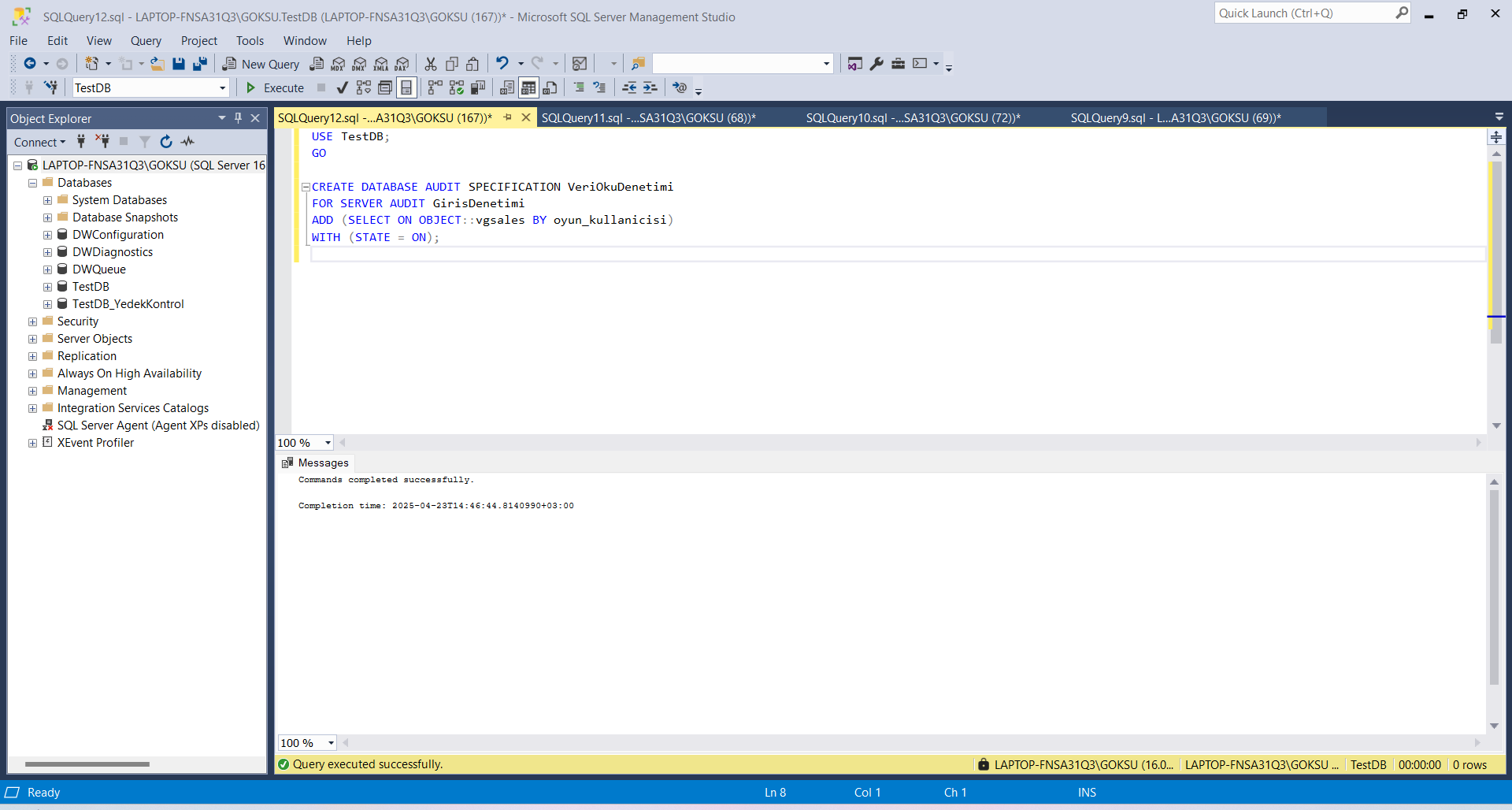


Figure 25 Audit specifications oluşturma

Bu komut, oyun\_kullanicisi kullanıcısı vgsales tablosunda SELECT yaptığında loglanmasını sağlar.

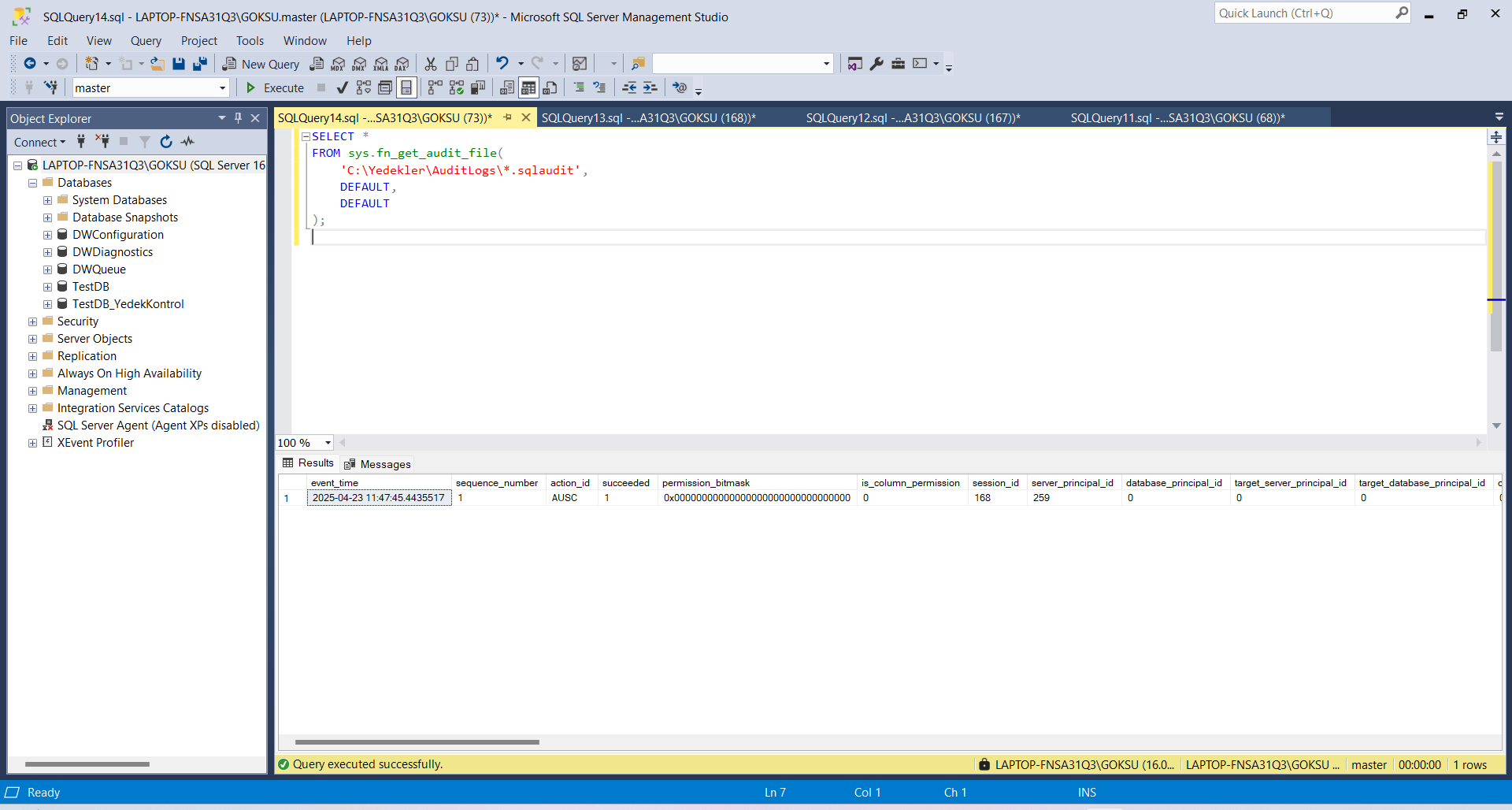


Figure 26 Audit loglarını okuma

Bu adımda SQL Server Audit özelliği kullanılarak, belirli kullanıcıların veritabanı aktiviteleri izlenmiş ve loglanmıştır. Özellikle `oyun\_kullanicisi` adlı kullanıcının `vgsales` tablosunda yaptığı SELECT işlemleri kayıt altına alınmıştır. Böylece hem izinsiz erişimler tespit edilebilir hâle gelmiş hem de sistemin şeffaflığı sağlanmıştır.