**ANKARA ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**



**BLM4522 Final Ödevi**

**Ecem Şimşek - 21290553  
Latife Süeda Tuğrul -20290297**

Github: https://github.com/suedatgrl/SQL\_work/

**Vize:**

**1.Proje- Veritabanı Güvenliği ve Erişim Kontrolü (3)**

**2.Proje- Veritabanı Performans Optimizasyonu ve İzleme (1)**

**3.Proje- Veritabanı Yedekleme ve Otomasyon Çalışması (7)**

**Final:**

**4.Proje- Veritabanı Yedekleme ve Felaketten Kurtarma Planı**

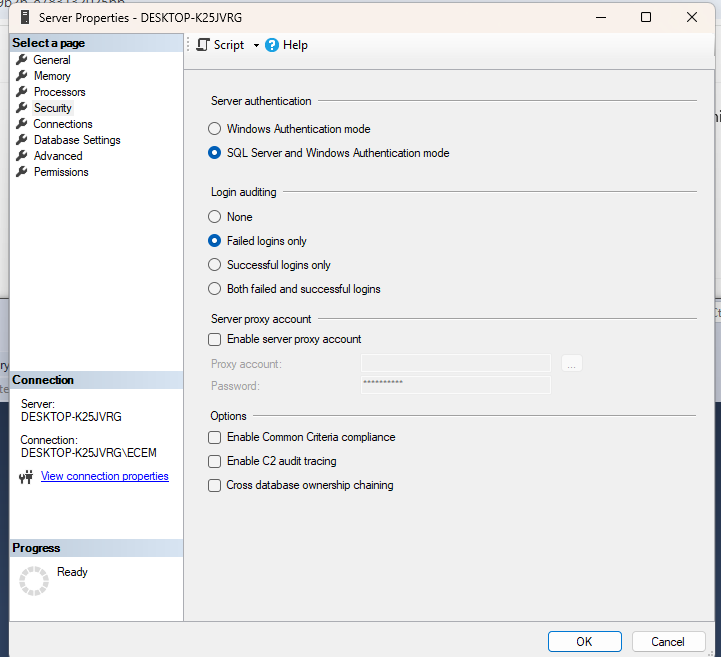
**1.Proje- Veritabanı Güvenliği ve Erişim Kontrolü**

**Erişim Yönetimi**

1. Sol üstteki **Object Explorer** penceresinde sunucu adına **sağ tıklanır** → **Properties** seçilir.

2. Açılan pencerede soldaki menüden **Security** sekmesini seçilir.

3. **Server authentication** kısmında **“SQL Server and Windows Authentication mode”** seçeneğini işaretlenir.

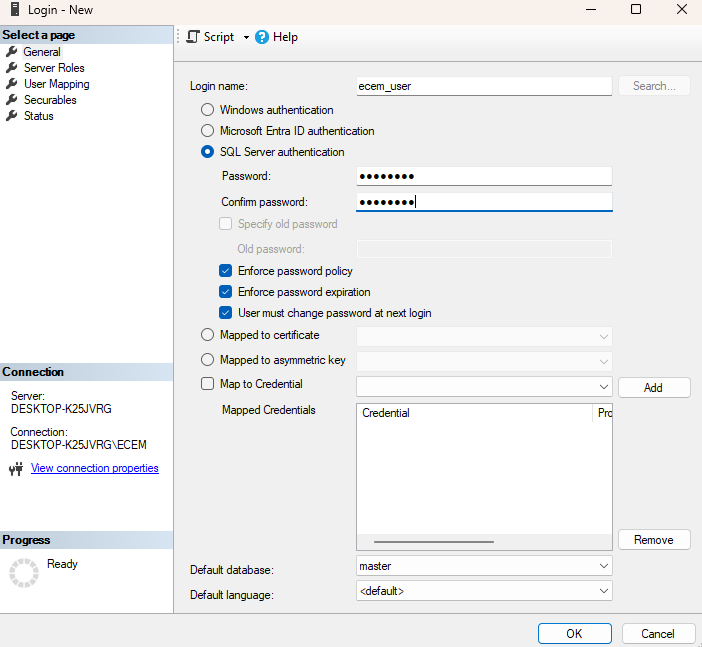


4. Bu değişikliklerin geçerli olması için sunucuyu yenilenir.

5. SSMS’te Object Explorer’dan: Security > Logins > **sağ tıkla** > **New Login...** seçilir.

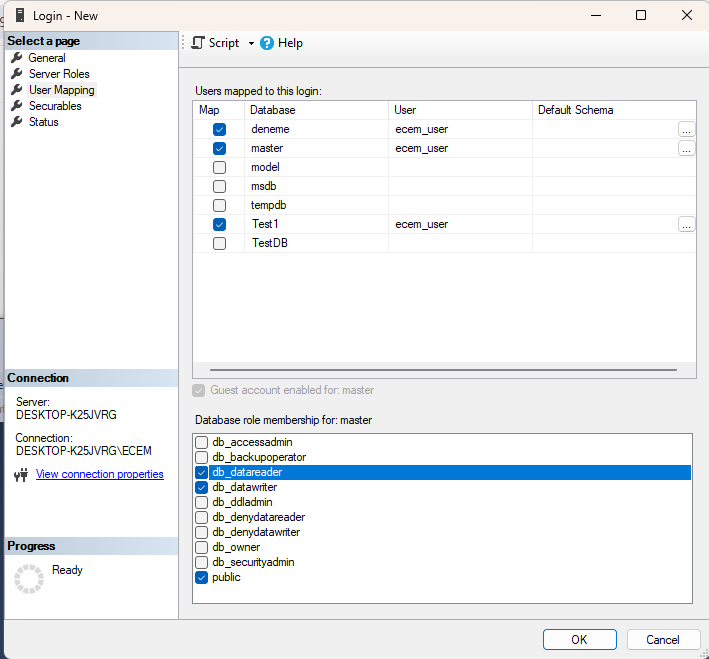
6. Açılan pencerede:

* **Login name** kısmı doldurulur.
* **SQL Server authentication** seçili olsun.
* Güçlü bir şifre girilir.



7. Sol taraftaki **User Mapping** sekmesine gelinir.

* Kullanıcının erişmesini istediğin veritabanını seçilir.
* Alttaki kutulardan db\_datareader, db\_datawriter rolleri işaretlenir (okuma/yazma yetkisi için).



8. Kullanıcı oluşturulmuş olacak.

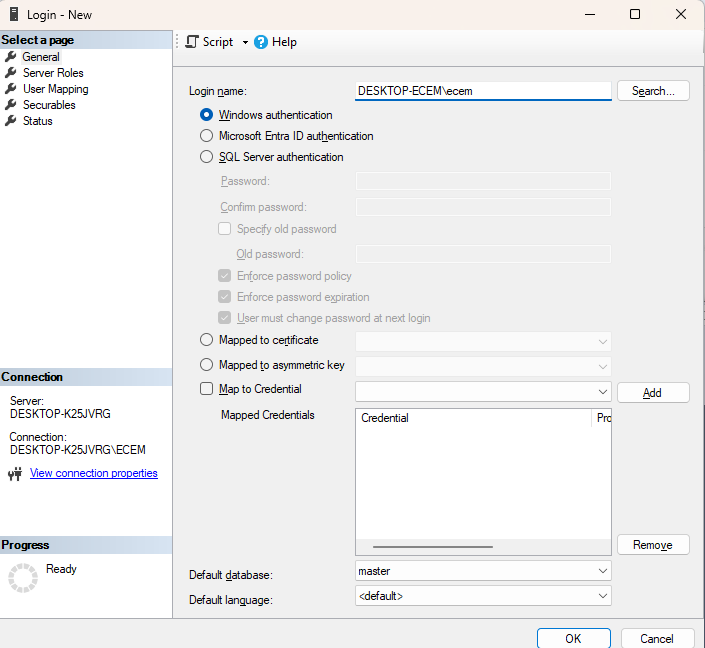
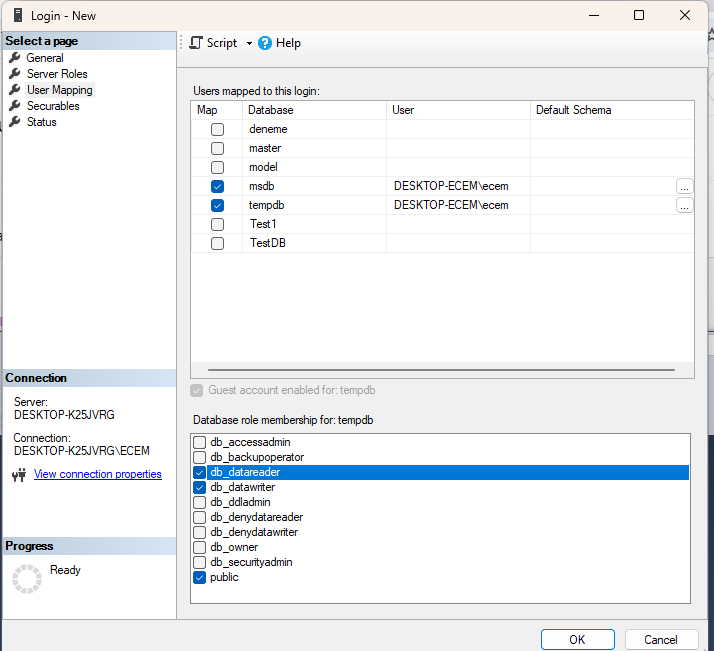
**SQL Server Authentication ve Windows Authentication**

1. Yine: Security > Logins > **sağ tıkla** > **New Login...**

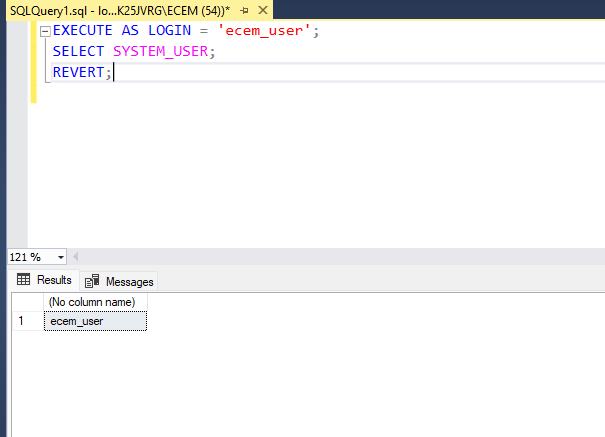
2. Açılan pencerede:

* **Login name**: Windows kullanıcı adı yazılır.
* Authentication kısmında bir değişiklik yapılmasına gerek yok; çünkü bu bir Windows kullanıcısı olacak.

3. **User Mapping** sekmesinden aynı şekilde veritabanını seçilir ve roller tanımlanır.



Giriş yapılan kullanıcıyı test etmek için şu kod çalıştırılır:

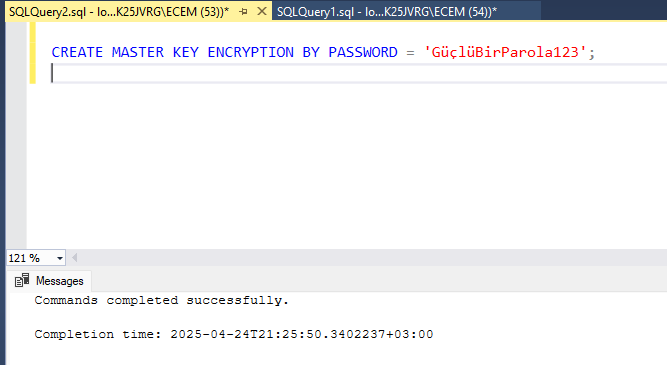


**Veri Şifreleme**

1.TDE'yi etkinleştirmeden önce, **veritabanı şifreleme anahtarı** (Database Encryption Key) için bir **Master Key** oluşturulmalıdır. Master Key, şifreleme anahtarlarını korur.

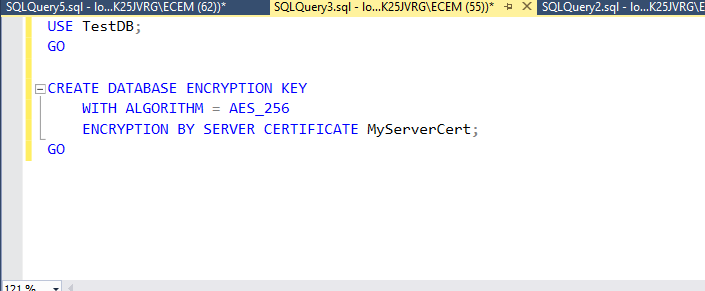
**-Yeni Sorgu** penceresini açılır.

-Aşağıdaki komutla Master Key'i oluşturulur:



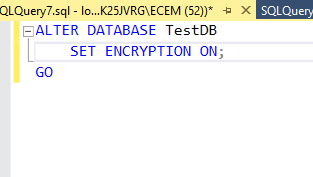
Bu komut, veritabanı şifreleme için gerekli olan Master Key'i oluşturur.

Şimdi **Database Encryption Key (DEK)** oluşturulmalıdır. DEK, veritabanı içindeki verileri şifrelemek için kullanılır.

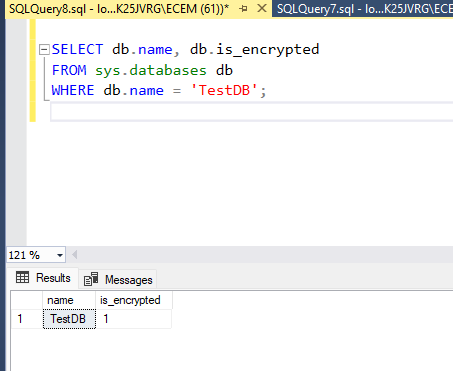


Artık TDE'yi etkinleştirebiliriz. Bu, veritabanındaki tüm verileri şifreler ve diske şifrelenmiş olarak yazılır.

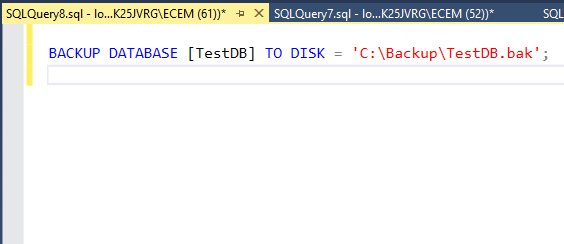
* Aşağıdaki komutla **TDE**'yi etkinleştir:



TDE'nin başarıyla etkinleşip etkinleşmediğini kontrol etmek için şu komutu çalıştırabiliriz.



Veritabanının yedeğini alırken, şifrelenmiş veriler de korunur. Aşağıdaki komutla veritabanının yedeğini alabiliriz.



Artık veritabanındaki hassas bilgiler **TDE ile şifrelenmiş durumda** ve **güvenli bir şekilde korunuyor**.

Bu adımlarla veritabanındaki veriler disk üzerinde şifreli halde saklanacak. Veritabanı şifrelemesi ve yedekleme işlemleri sayesinde verilerin güvenliği artmış olacak.

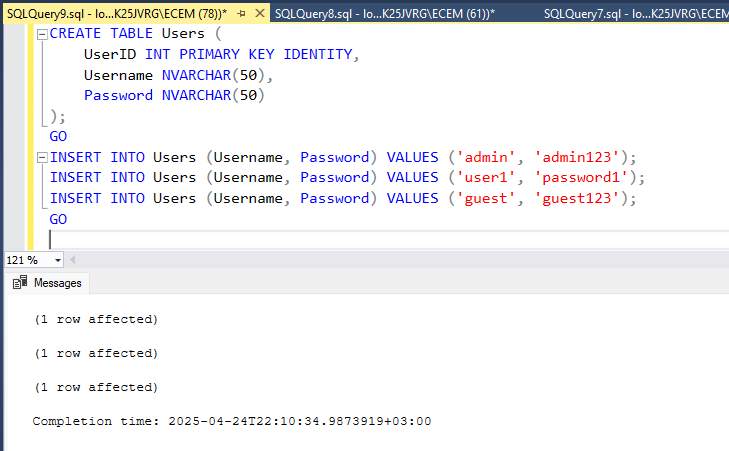
**SQL Injection Testleri**

P**arametreli sorgu (**özellikle **stored procedure)** kullanarak SQL Injection'a karşı nasıl korunacağı ele alınacak.

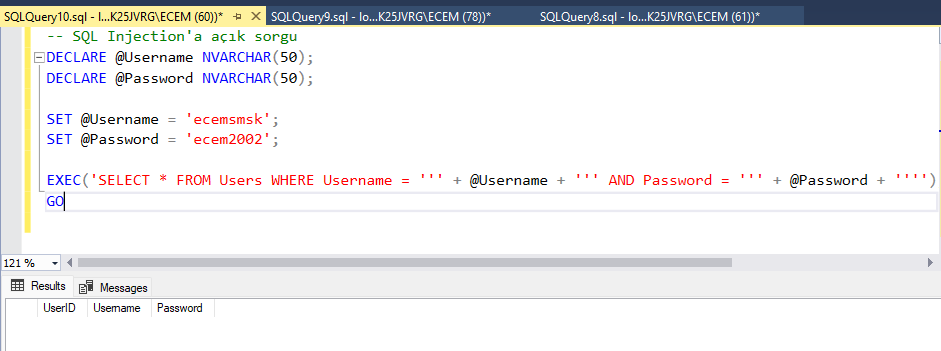
1. İlk olarak, test ortamını kurmak için basit bir veritabanı ve kullanıcılar tablosu oluşturulur. Bu tablonun içine test verisi eklenir.

2. Veritabanı oluşturulduktan sonra, kullanıcılar hakkında bilgi tutacağımız bir tablo oluşturulur. Bu tabloda kullanıcı adı ve şifre bilgilerini tutacağız.

3. Bu tabloya bazı test kullanıcıları eklenir.

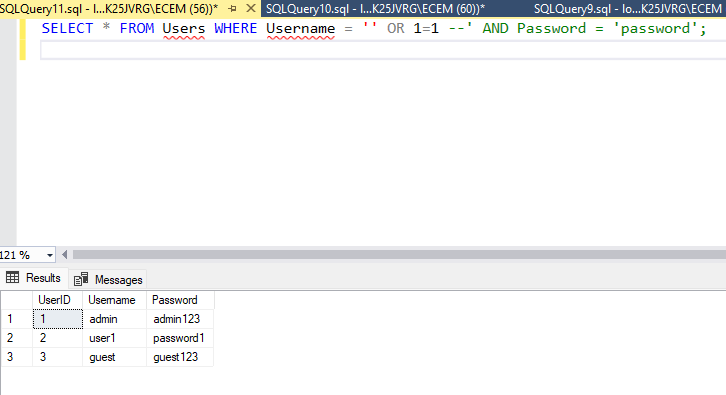


4.Şimdi, SQL Injection'a açık olan bir sorgu yazalım. Bu sorgu, kullanıcı adı ve şifreyi kontrol etmek için gelen girdileri doğrudan SQL sorgusuna ekler. Bu, SQL Injection'a neden olabilir.



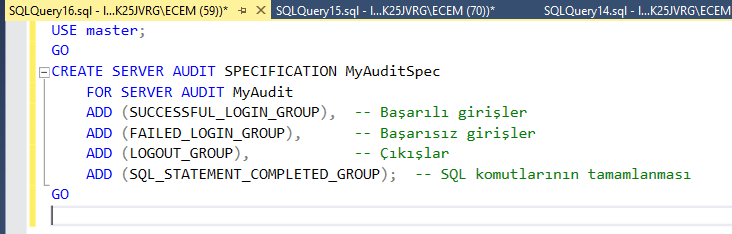
Yukarıdaki kod, **user\_input** yerine kullanıcıdan alınan verileri doğrudan sorguya ekler. Eğer bu sorguya kötü niyetli bir giriş yapılırsa, SQL Injection saldırısı gerçekleştirilebilir.

Eğer kullanıcı adı yerine ' OR 1=1 -- girerse, sorgu şu hale gelir:



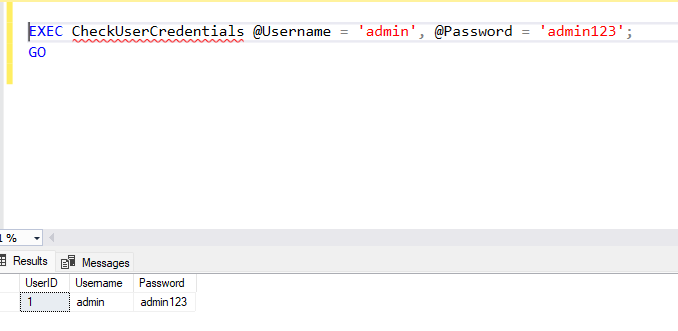
Bu sorgu, şifre kontrolünü geçersiz kılar ve tüm kullanıcıları geri döndürebilir.

Aşağıda, parametreli sorgu ile oluşturulmuş bir **Stored Procedure** örneği bulunmaktadır:



Bu stored procedure, **Username** ve **Password** parametrelerini alır ve bunları doğrudan SQL sorgusunda kullanır. Ancak bu yöntem, SQL Injection'a karşı güvenlidir, çünkü kullanıcı verileri SQL sorgusuna parametre olarak bağlanır ve veritabanı tarafından güvenli bir şekilde işlenir.

Stored Procedure'u çalıştırmak için aşağıdaki gibi bir sorgu yazabiliriz:

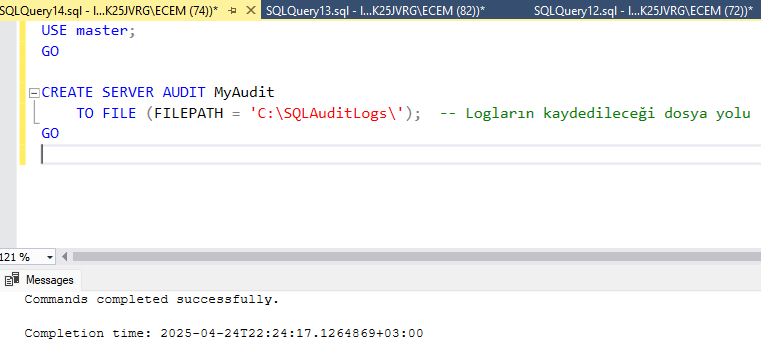


Bu komut, **CheckUserCredentials** prosedürünü çalıştırarak, belirtilen kullanıcı adı ve şifreyi sorgular. Ancak bu prosedür, SQL Injection'a karşı korumalıdır çünkü kullanıcı verileri parametre olarak işlenir.

**Audit Logları**

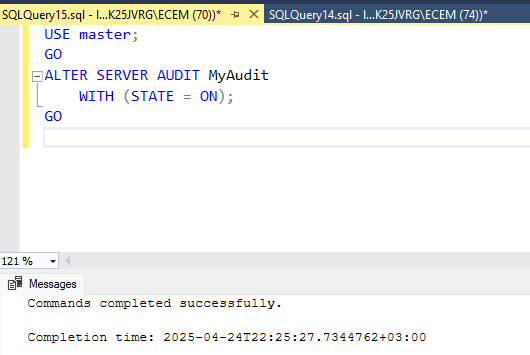
SQL Server Audit özelliğini kullanabilmek için öncelikle veritabanı denetimini başlatmamız gerekir. SQL Server'da Audit, genellikle bir **Audit** nesnesi ve bunun altına bağlı **Audit Specification** nesnelerinden oluşur.

1.Audit nesnesi, veritabanı üzerinde yapılacak aktivitelerin loglanmasını sağlamak için kullanılır.

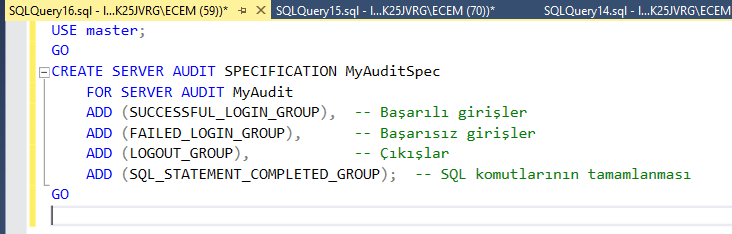


Bu komut, MyAudit adında bir audit nesnesi oluşturur ve logları belirtilen dosya yoluna kaydeder.

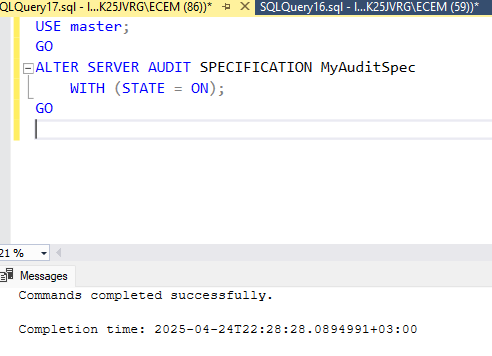
2.Audit nesnesini oluşturduktan sonra, onu başlatmamız gerekir.



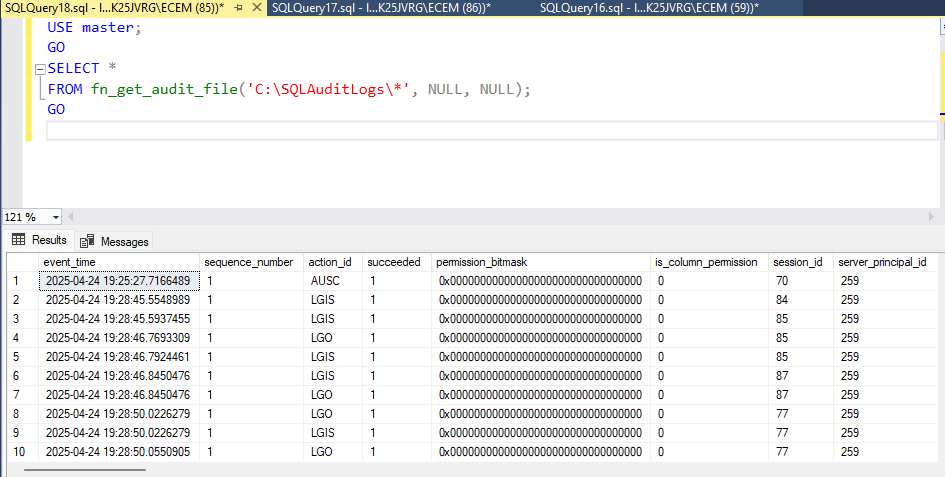
3. Audit'i başlatıp, belirli işlemleri izlemek için bir server-level audit specification oluşturabiliriz. Örneğin, bir kullanıcının giriş yaptığı, çıkış yaptığı, veritabanına bağlandığı ve sorgularını çalıştırdığı aktiviteleri izleyelim.



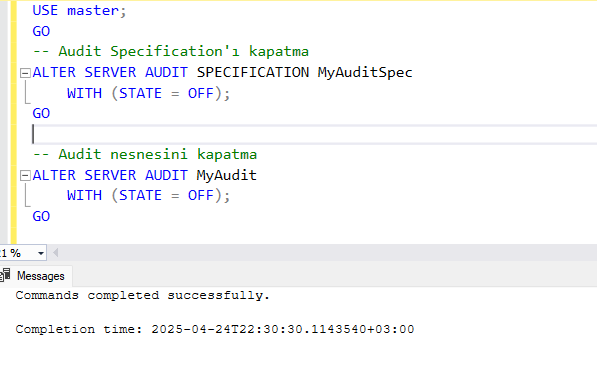
4**.** Oluşturduğumuz **audit specification**'ı başlatmamız gerekir.



**5.** Audit logları SQL Server tarafından **fn\_get\_audit\_file** fonksiyonu ile sorgulanabilir. Bu fonksiyon, belirttiğiniz dosya yolundaki logları okuyabilir.



**6.** Eğer audit'i kapatmak istenirse, aşağıdaki komutlar kullanılabilir.



**2.Proje- Veritabanı Performans Optimizasyonu ve İzleme**

**Veritabanı İzleme**

1. SQL Server Profiler açılır.

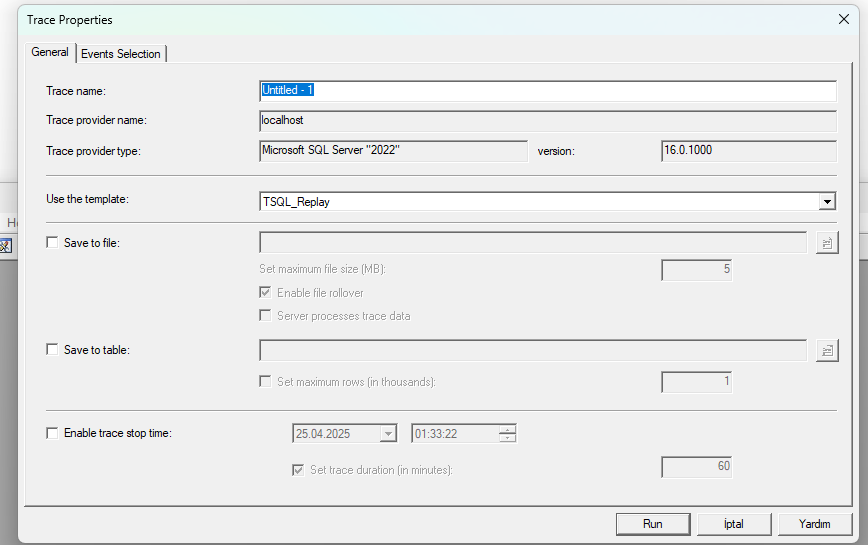
2. “File” → “New Trace…” seçilir.

3. SQL Server’a bağlanılır .

4. Bir **Trace Template** seçilir. (örnek: “TSQL\_Replay”).

5. “Events Selection” sekmesine gelerek aşağıdaki gibi olaylar eklenir:

* SQL:BatchCompleted
* RPC:Completed
* Showplan XML

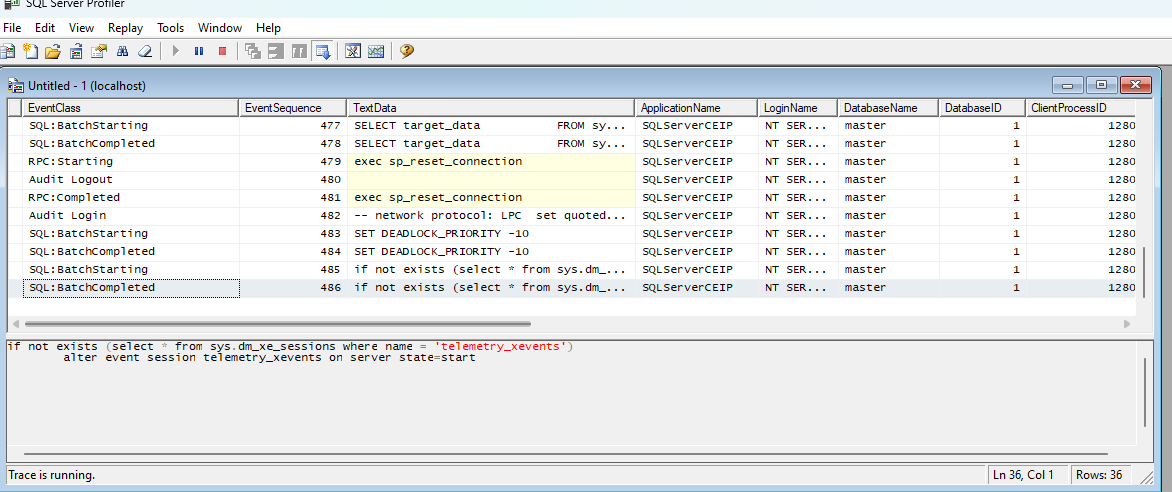


 Trace başlatılır.

 Ağır çalışan veya sürekli tekrarlanan sorguları kaydedilir.

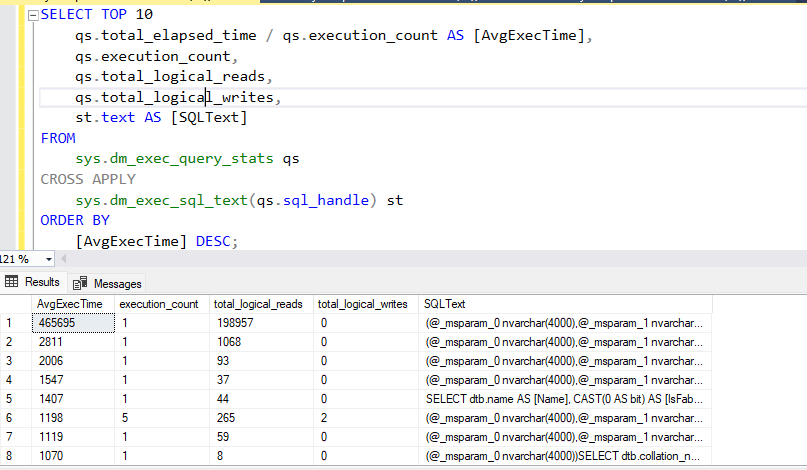
 CPU time, Reads, Writes sütunlarına göre analiz edilebilir.

 Trace'i bitirip .trc dosyası olarak kaydedilebilir..



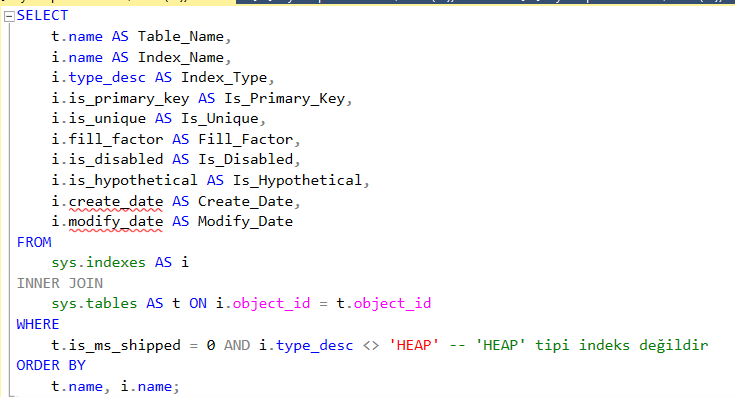
DMV'ler, veritabanının iç durumu hakkında canlı bilgiler sağlar.

Bu sorgu, en çok zaman alan sorguları verir.

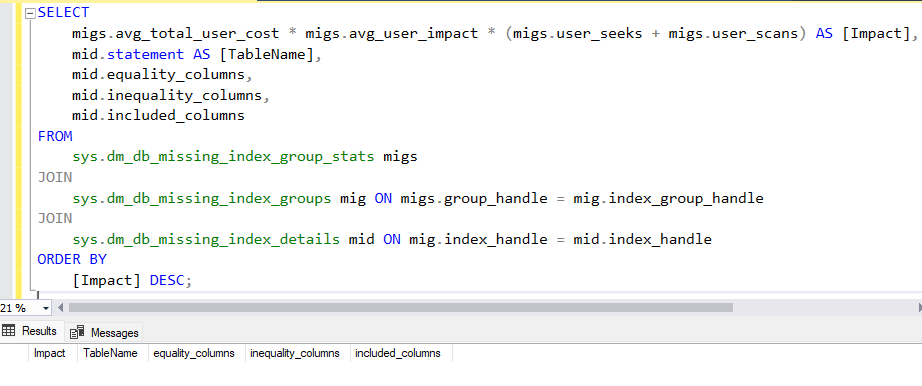


**İndeks Yönetimi**

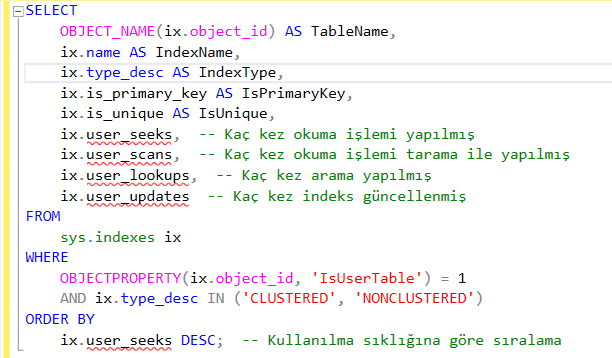
**1.** İlk adımda, veritabanında hangi indekslerin bulunduğunu incelememiz gerekir. Aşağıdaki SQL sorgusu, mevcut tüm indeksleri listeleyecektir**.**



**2.** Bu sorgu, hangi tabloya hangi index’in eklenmesinin faydalı olacağını gösterir.Eksik indeksleri tespit eder.

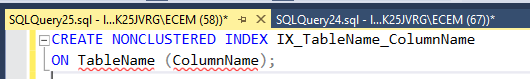


**3.** Gereksiz indeksler, yazma işlemlerini yavaşlatabilir ve disk alanını gereksiz yere doldurabilir. Kullanılmayan veya fazla indeksleri tespit etmek için şu sorguyu kullanabilirsiniz.



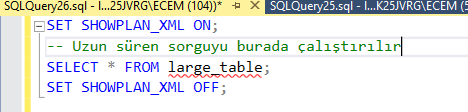
4. Veritabanında en çok kullanılan sorgulara göre yeni indeksler oluşturmak performansı artırabilir. Örneğin, sıkça kullanılan bir sorgu, belirli bir sütuna göre sıralama yapıyorsa, o sütun üzerinde bir indeks oluşturmak faydalı olabilir.

İndeks oluşturma için aşağıdaki komutu kullanılabilir:



**Sorgu İyileştirme**

1. Uzun süren sorguları analiz etmek için, sorgu planlarını incelemek gerekir. Sorgu planı, SQL Server'ın bir sorguyu nasıl çalıştırdığına dair bilgi sağlar. Aşağıdaki sorguyu çalıştırarak bir sorgunun çalışma planı alınabilir.



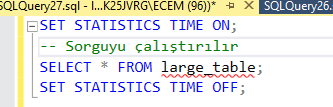
2.Yavaş sorguların bir diğer nedeni, çok sayıda tablonun birleştirilmesidir. Bu durumda, doğru JOIN türünü seçmek önemlidir. INNER JOIN, LEFT JOIN, RIGHT JOIN gibi JOIN türleri, sorguların hızını etkileyebilir.

* INNER JOIN: Sadece her iki tabloda da eşleşen kayıtları getirir.
* LEFT JOIN: Sol tablodaki tüm kayıtları getirir, sağ tablodan eşleşmeyenler NULL olur.

Eğer LEFT JOIN gereksiz yere kullanılıyorsa, sorgu süresi artabilir.

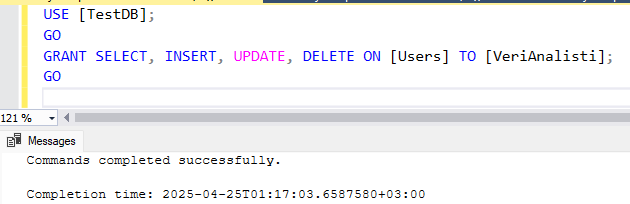
3. İyileştirmeleri uyguladıktan sonra, sorgu performansını test etmek önemlidir. Test etmek için aşağıdaki yöntemleri kullanılabilir.

1. **Execution Plan**: SQL Server Management Studio (SSMS) üzerinden, sorguyu çalıştırırken "Include Actual Execution Plan" seçeneğini etkinleştirerek sorgu planını incelenebilir.
2. **Sorgu Süresi**: Sorgu süresi ile yapılan değişikliklerin etkisini görmek için, SET STATISTICS TIME ON komutunu kullanılabilir. Bu, sorgu süresini gösterir.

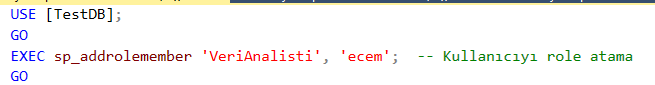


**Veri Yöneticisi Rolleri**

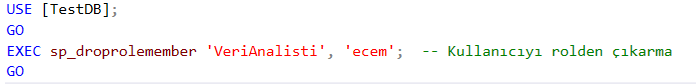
1**.** Rolü oluşturduktan sonra, bu role belirli yetkiler atamamız gerekir. Yetkiler, veritabanında hangi işlemleri yapabileceklerini belirler. Aşağıda, bir role veri okuma ve yazma yetkileri verme örneği yer almaktadır.



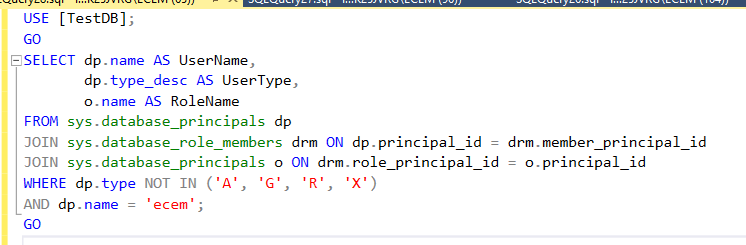
2. Bir rol oluşturduktan ve gerekli yetkileri verdikten sonra, bu rolü bir kullanıcıya atamamız gerekir. Bunun için aşağıdaki komutları kullanılabilir.



3. Bir kullanıcıyı bir rolden çıkarmak için aşağıdaki komutu kullanılabilir.



4. Bir kullanıcının hangi rollerde olduğunu görmek için aşağıdaki sorguyu çalıştırılabilir.



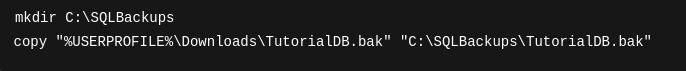
**3.Proje- Veritabanı Yedekleme ve Otomasyon Çalışması:**

1. Ortam Hazırlığı:

SQL Server Developer Edition ve SSMS yüklendi.

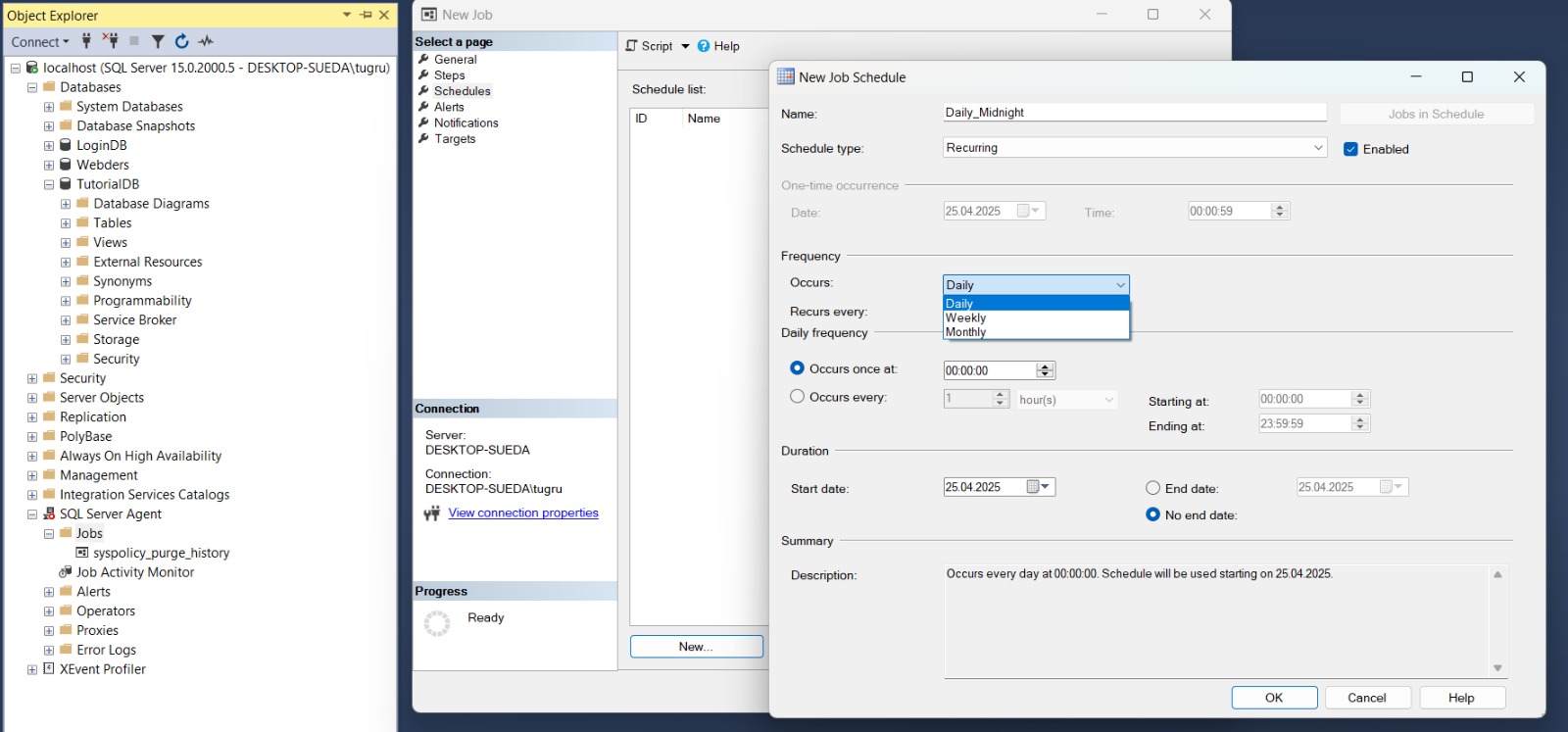
SSMS ile localhost (veya .\SQLEXPRESS) üzerinden “Database Engine”e Windows/sa ile bağlanıldı.

1. Örnek Veritabanı Restore edildi. Burada Db ismimiz TutorialDB olan veritabanımızla çalışmamıza devam edeceğiz.



TutorialDB.bak dosyası C:\SQLBackups yerel klasörüne kopyalandı.

SSMS’te Databases → Restore Database… ile TutorialDB adıyla başarıyla yüklendi.

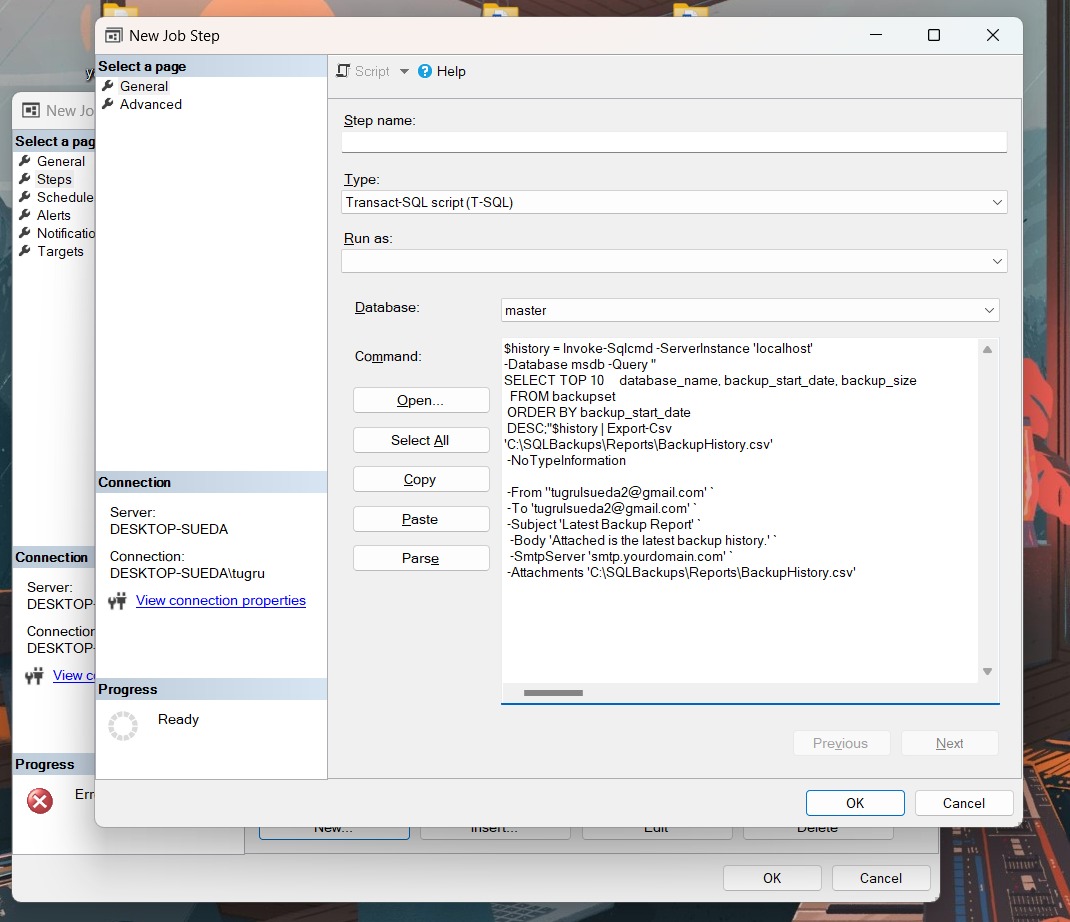
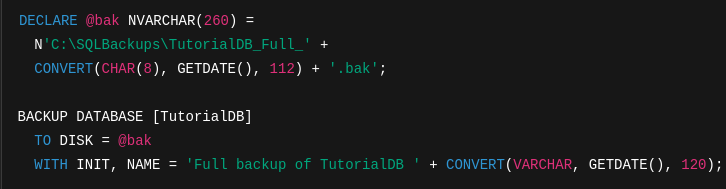


1. Backup Job Oluşturma

SQL Server Agent altında New Job… ile TutorialDB\_FullBackup işi tanımlandı.

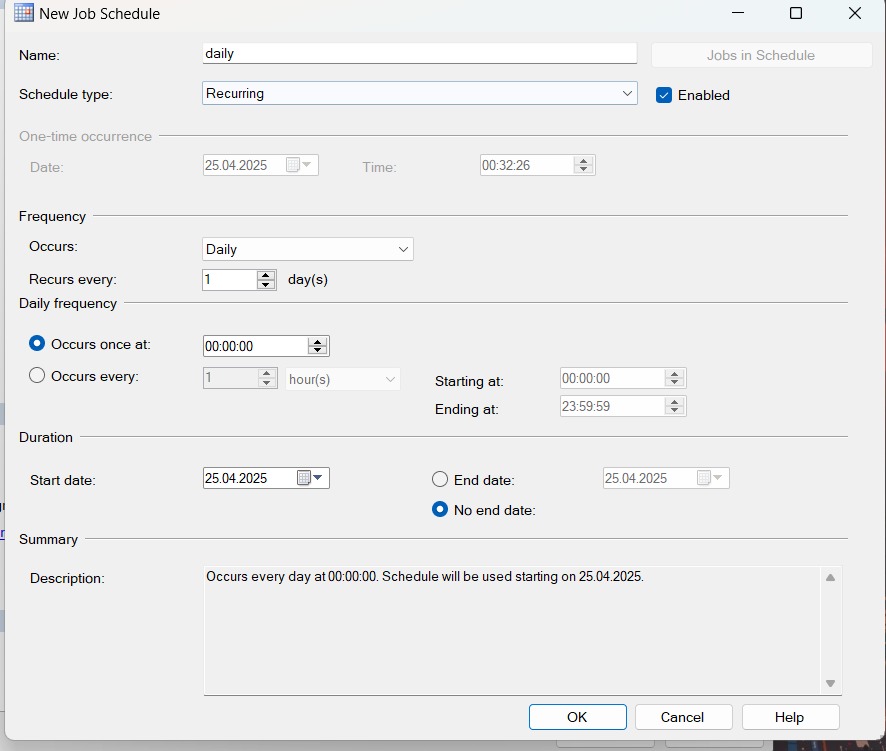
Steps sekmesinde, TutorialDB için günün tarihini isimde kullanan T-SQL yedekleme script’i eklendi. Database olarak master değil kendi TutorialDB isimli veri tabanımızı ekliyoruz.

T-SQL kodu:



1. Zamanlama (Schedule)

Job’un Schedules → New… bölümünde, her gece 00:00’da çalışacak şekilde günlük tetikleyici ayarlandı. Böylece yedekleme günlük olarak sağlanacak.

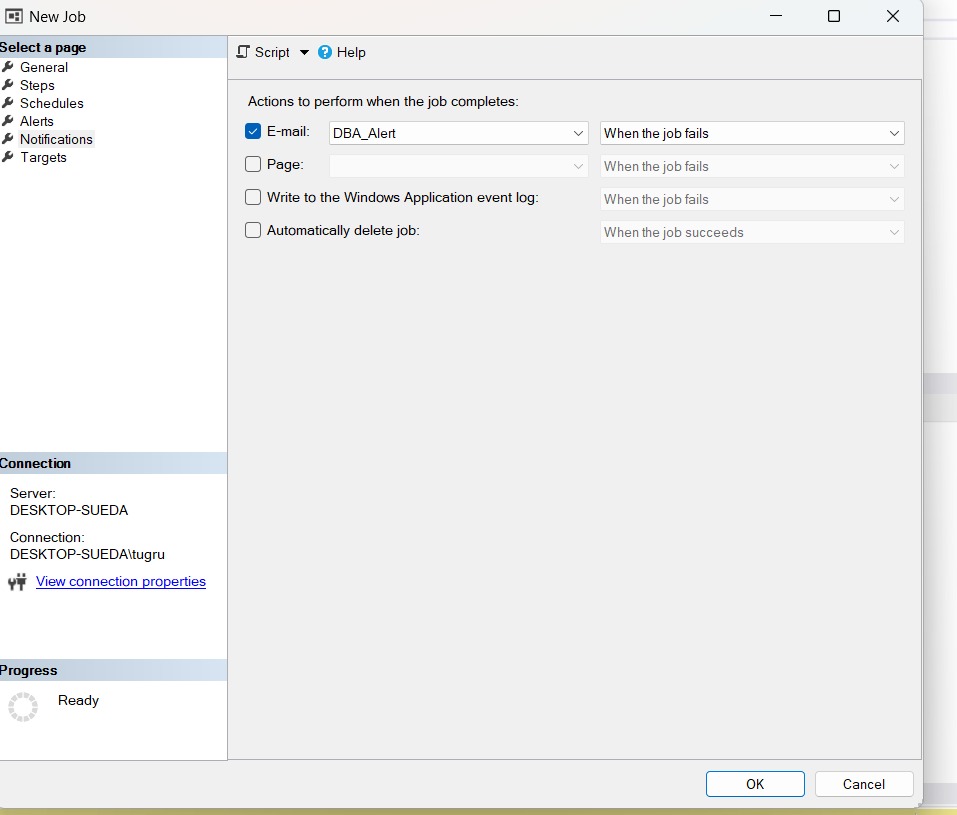
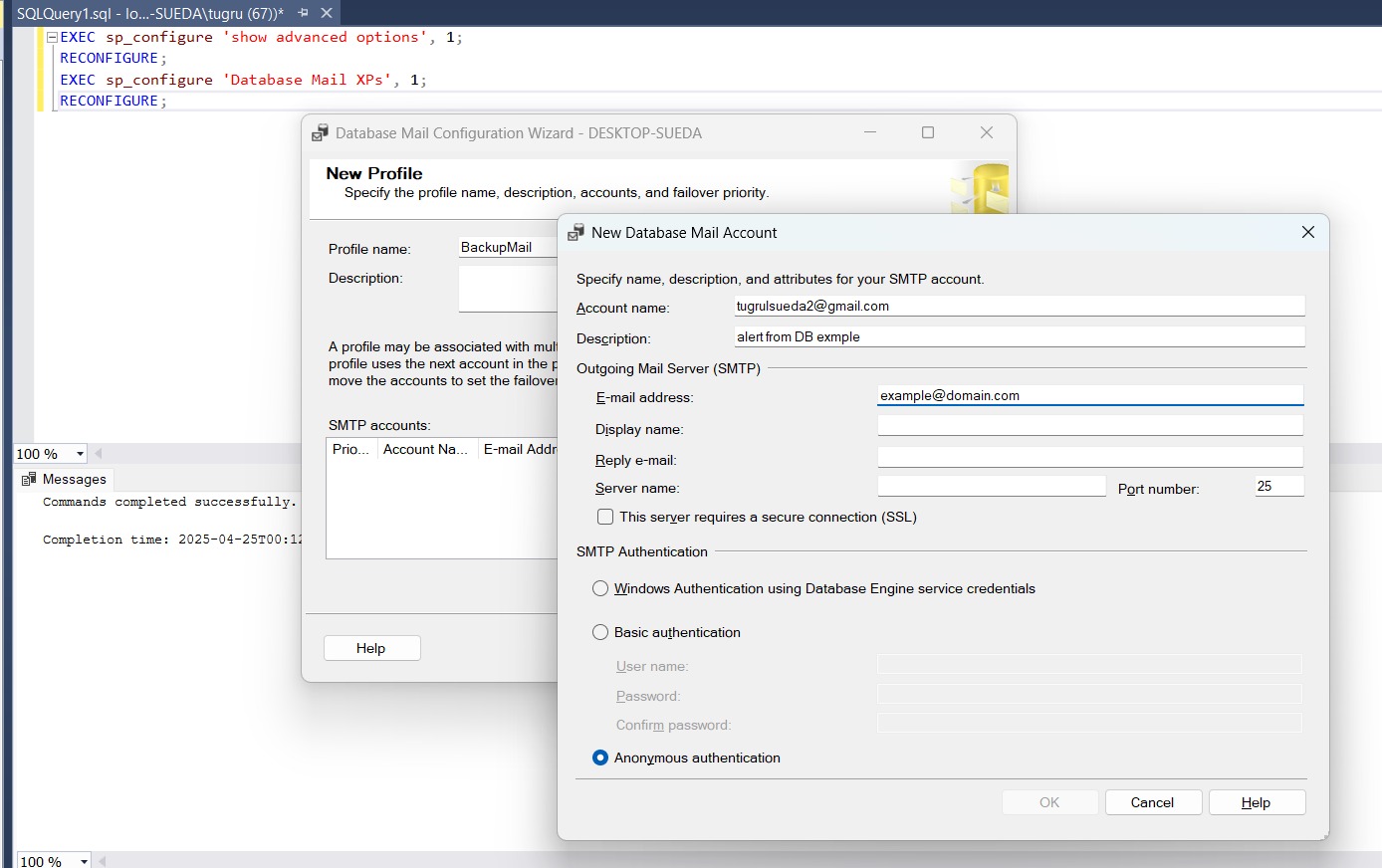
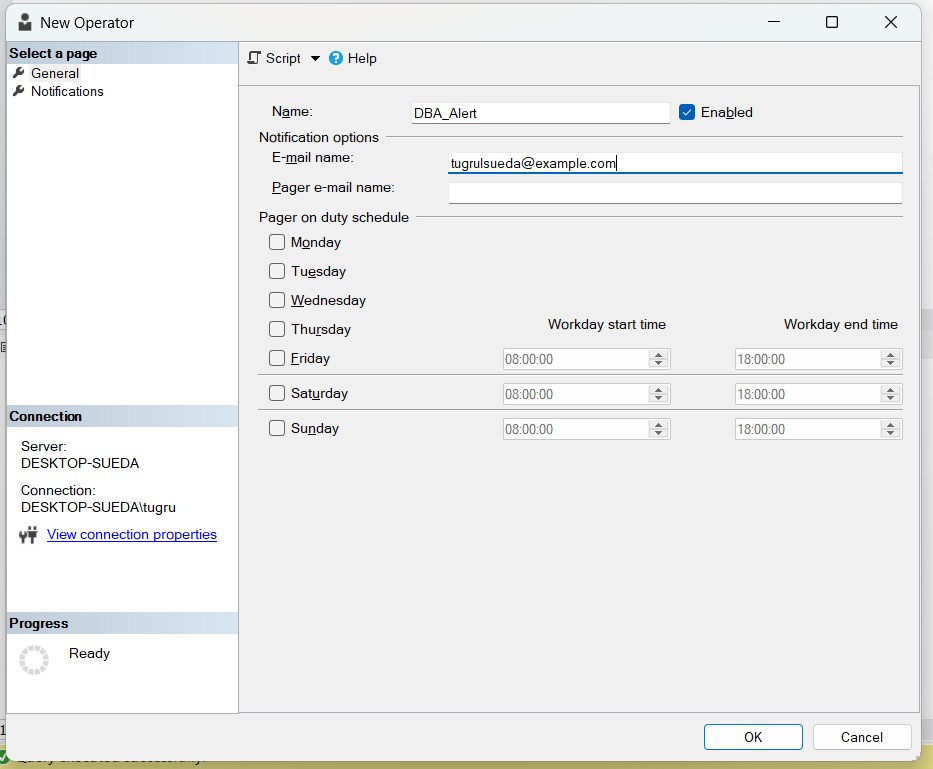
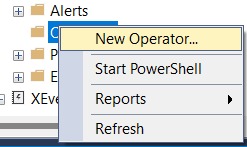


1. E-posta Uyarıları

Management → Database Mail ile BackupMail profili oluşturuldu.

SQL Server Agent Properties’ten bu profil etkinleştirildi.

Operators altında DBA\_Alert tanımlanıp, Job’un Notifications → If job fails → E-mail kısmına atandı.

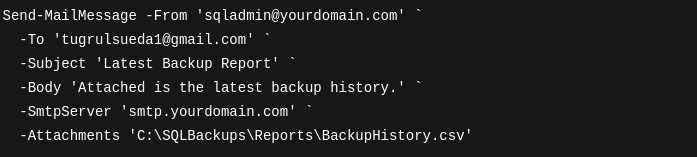
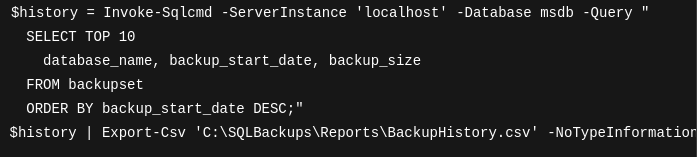
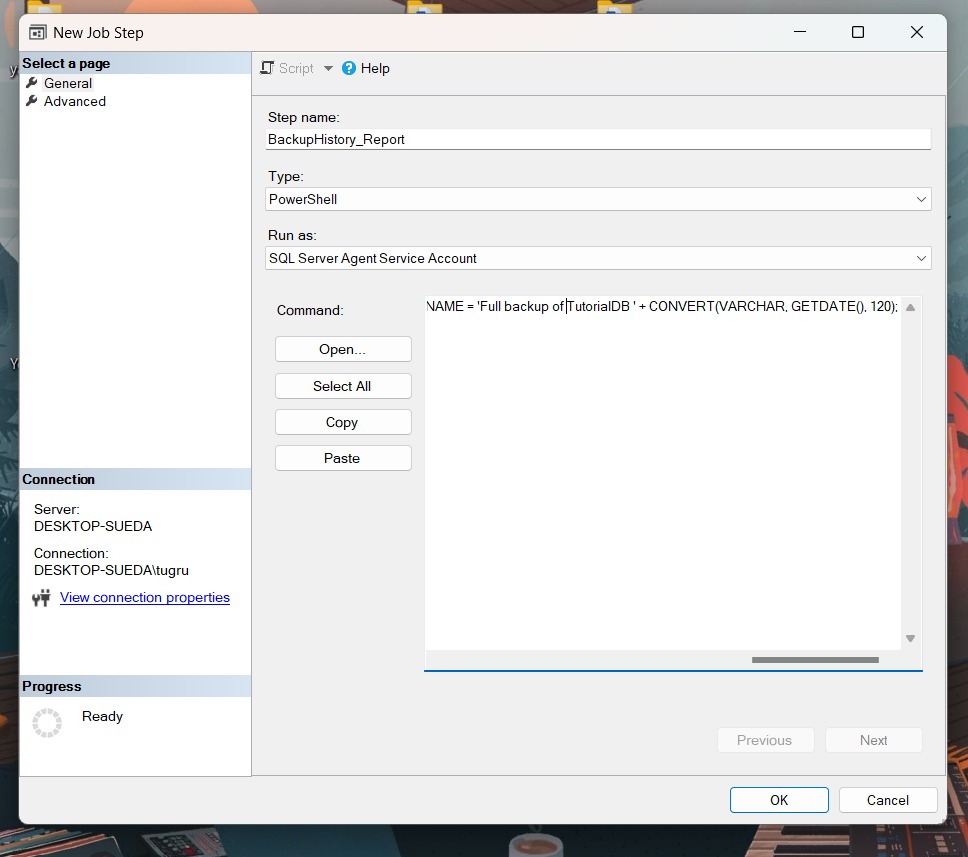


1. Raporlama

Yeni bir PowerShell tabanlı job (BackupHistory\_Report) eklenip,

msdb.backupset’ten son yedek bilgileri CSV’ye yazdırıldı,

Oluşan rapor ilgili adrese e-posta ile gönderildi.



**4.Proje- Veritabanı Yedekleme ve Felaketten Kurtarma Planı**