**ANKARA ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

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

Açıklama otomatik olarak oluşturuldu**

**BLM4522-Ağ Tabanlı Paralel Dağılım Sistemleri**

## **PROJE AÇIKLAMASI : “Veritabanı Performans Optimizasyonu ve İzleme” **, “Veritabanı Yedekleme ve Felaketten Kurtarma Planı”,** “**Veritabanı Güvenliği ve Erişim Kontrolü” , “**Veritabanı Yük Dengeleme ve Dağıtık Veritabanı Yapıları”, **“Veri Temizleme ve ETL Süreçleri Tasarımı”, “Veritabanı Yükseltme ve Sürüm Yönetimi” ve “Veritabanı Yedekleme ve Otomasyon Çalışması“ konularını uygulayarak gösterme****

**Ders Danışmanı : Enver BAĞCI**

**Elif BELLER - 19290957**

**Github hesabım : ElifBeller-eng**

**Proje Linkim : <https://github.com/ElifBeller-eng/BLM4522-FINAL> (videolar read-me kısmında)**

**29.05.2025**

**İÇİNDEKİLER**

1. **Veritabanı Performans Optimizasyonu ve İzleme**

**Amaç **-**  Temel Başlıklar - Senaryo**  **..........................................4**

### ****Uygulama Adımları ve SQL Kodları****

#### SQL Server Profiler ile izleme başlat (manuel adım) ...............5

**Yavaş Sorgu Örneği - Execution Plan ile Analiz**

**Performans İçin İndeks Oluştur ..............................................6**

**İstatistikleri Güncelle - Fragmentasyon Kontrolü**

**Yeniden Oluştur - Yeniden Düzenle (REBUILD -REORGANIZE) 7**

### ****Sonuç ve Kazanımlar - Ek Notlar .............................................8****

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

**Amaç - Teorik Bilgi - Yedekleme Türleri - Yedekleme Stratejileri .............................................................................9**

**Kurtarma Stratejileri - Uygulama Adımları ve SQL Kodları -**

**Full Yedek Alma ...................................................................10**

**Transaction Log Yedeği Alma - Geri Yükleme (Restore) .........11**

### ****Aynı isimle restore (mevcut veritabanı üzerine) ...................12****

**Yeni isimle restore (orijinali bozmadan test amaçlı).....13**

1. **Veritabanı Güvenliği ve Erişim Kontrolü**

**Amaç - Temel başlıklar .........................................................14**

**Çözüm - Örnek .....................................................................15**

**İÇİNDEKİLER (DEVAM) :**

**Ek notlar -SQL Server Audit Özelliği ......................................16**

**SQL Server Audit Özelliği ......................................................17**

**4. Veritabanı Yük Dengeleme ve Dağıtık Veritabanı Yapıları**

### Konu başlıkları .....................................................................18

**5.Veri Temizleme ve ETL Süreçleri Tasarımı**

**Amaç - Teorik Bilgi - Senaryo ................................................19**

**Uygulama Adımları ve SQL Kodları .......................................20**

**Temiz Verileri Yeni Tabloya Aktarma (LOAD) - Sonuç ...........21**

**Ek Notlar - Örnek .................................................................22**

## ****Veritabanı Yükseltme ve Sürüm Yönetimi****

## **Tek Kullanıcı Moduna Alma-Yedek Alma -Restore İşlemi-Veritabanını Çok Kullanıcılı Hale Getirme ............................**23****

**DDL Trigger Tanımlama -Test ve Geri Dönüş Planı...........24-25**

**7. Veritabanı Yedekleme ve Otomasyon**

**Amaç - Yedekleme Türleri - Senaryo ....................................26**

### ****Yedekleme SQL Komutu - SQL Server Agent ile Otomasyon - Kazanımlar ve Sonuç - Ek Bilgiler ..........................................27****

**Geri Yükleme (Restore) Testi - Otomasyon ........................29**

**Zamanlama (Task Scheduler) ......................................29-30-31**

**Test -Sonuç .....................................................................32-33**

1. **Veritabanı Performans Optimizasyonu ve İzleme**

### ****Amaç****

Veritabanındaki sorguların ve işlemlerin daha hızlı ve verimli çalışmasını sağlamak. Sistem kaynaklarını etkili kullanmak, yavaş sorguları tespit etmek ve optimizasyon önlemleri almak.

Performans izleme ve optimizasyonu birkaç temel başlıkta ele alınır:

**- SQL Server Profiler / Extended Events** :

Sorgu takibi, yavaş işlem tespiti için kullanılır.

****Execution Plan (Yürütme Planı)**:**

Sorgunun nasıl çalıştığını grafiksel olarak gösterir.

**- İndeks Kullanımı** :

Sorguların hızlı çalışması için doğru indeksler tanımlanmalı.

**- Fragmentasyon Kontrolü** :

Sık güncellenen tablolarda zamanla performans düşebilir, yeniden düzenleme gerekebilir.

**- İstatistiklerin Güncel Olması** :

SQL Server, sorgu planı üretmek için istatistiklere bakar.

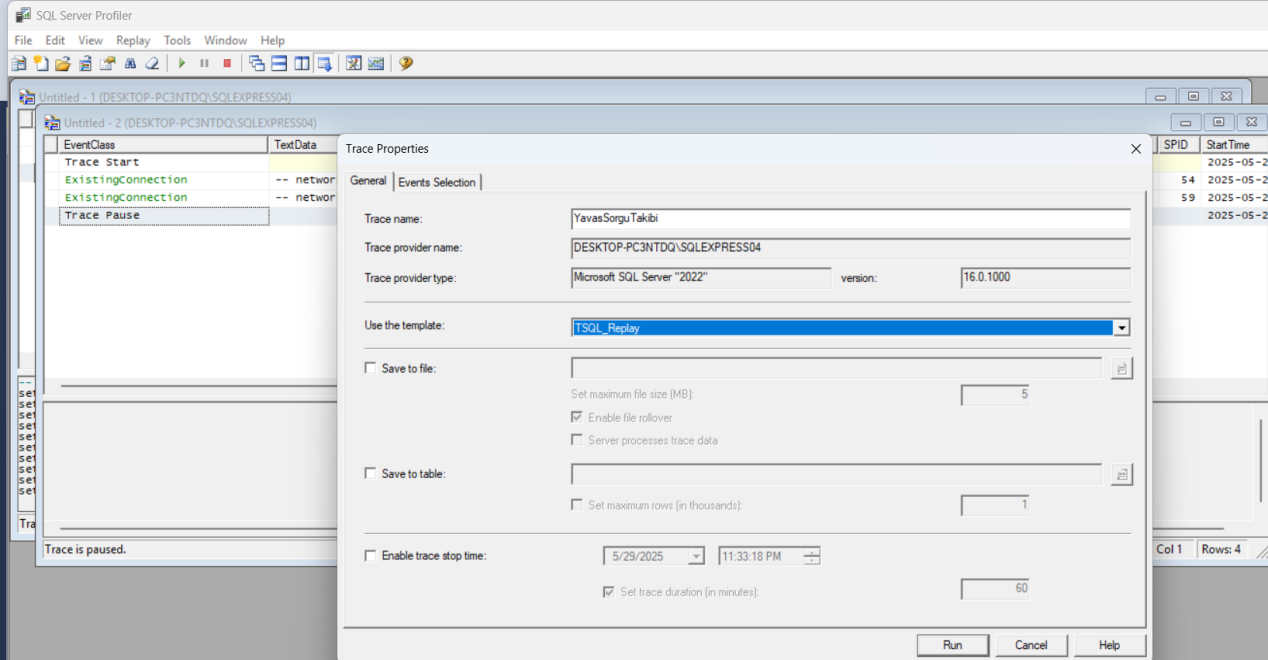
### ****Senaryo :****

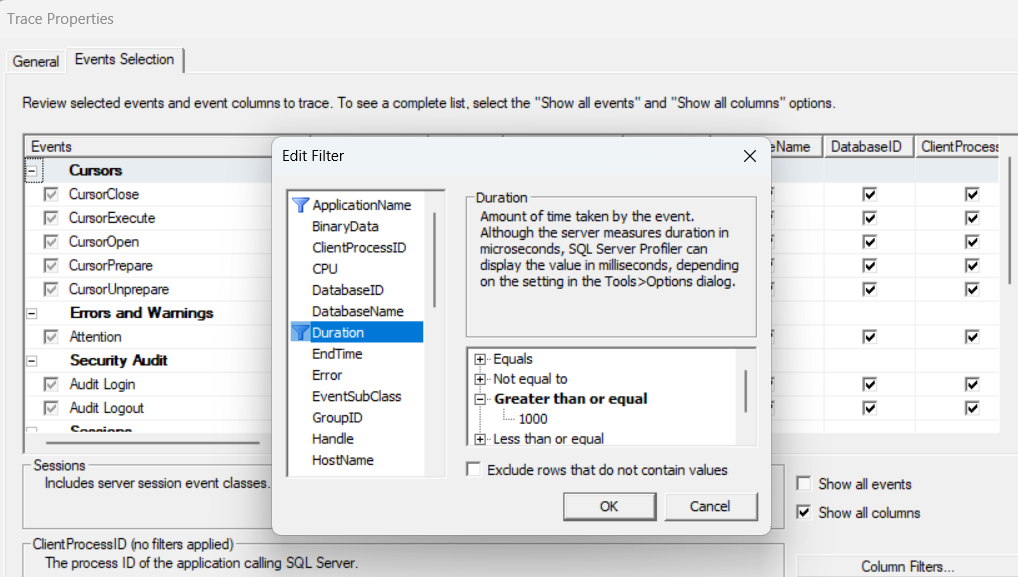
### Örneğin bir ogrenciler tablomuz var. Kullanıcılar sık sık “belirli ad-soyad veya tarih aralığında” arama yapıyor. Ama sorgular yavaş. Bu durumu analiz edip hızlandıracağız.

### ****Uygulama Adımları ve SQL Kodları****

#### 1. SQL Server Profiler ile izleme başlat (manuel adım) :

TSQL\_Replay template yerine Standard da seçilebilirdi.



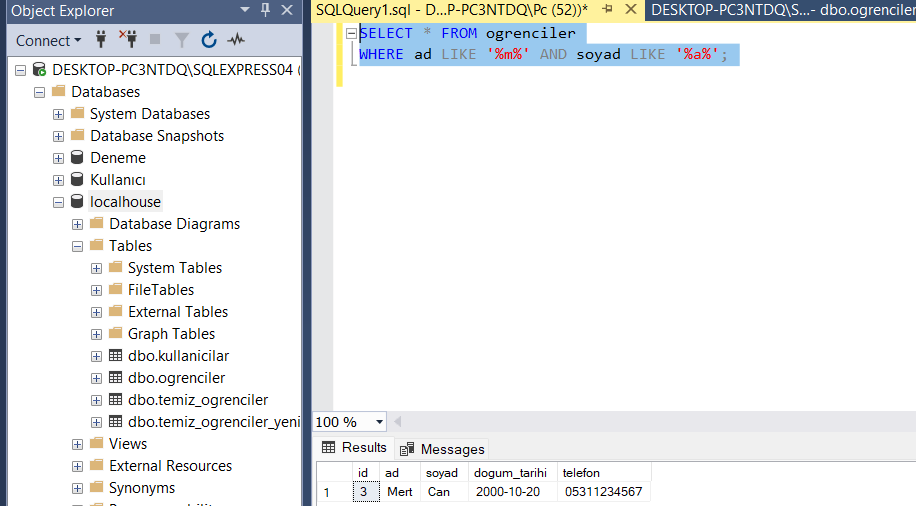


"Sadece **1000 milisaniye ve üzeri** süren sorguları göster."

1000 = 1 saniye

1. **Yavaş Sorgu Örneği :**

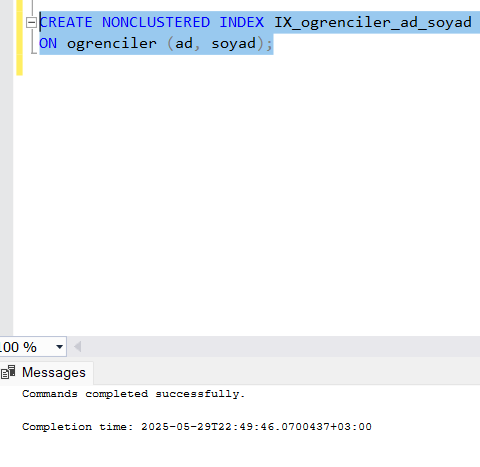
Bu sorgu tarama yapar, indeks yoksa çok yavaş çalışabilir.



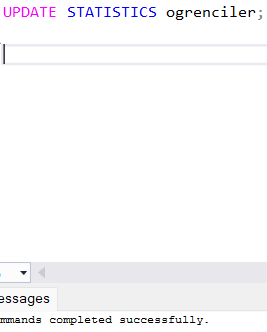
1. **Execution Plan ile Analiz :**

SSMS üzerinde Ctrl + M ile Execution Plan aktif edilir, sonra sorguyu çalıştır.

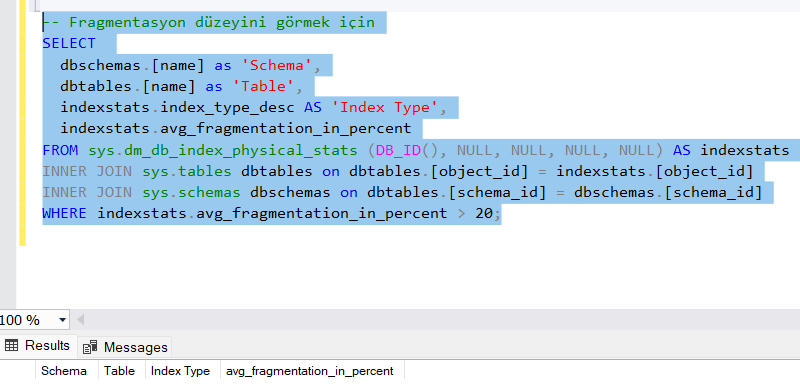
1. **Performans İçin İndeks Oluştur :**



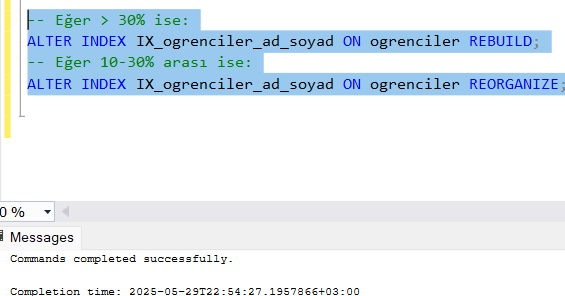
1. **İstatistikleri Güncelle:**



1. **Fragmentasyon Kontrolü:**



**7. Yeniden Oluştur (REBUILD) veya Yeniden Düzenle (REORGANIZE) :**



### ****Sonuç ve Kazanımlar****

- Yavaş çalışan sorgular tespit edildi ve Execution Plan ile analiz edildi.

- Uygun indekslerle sorgu süresi düşürüldü.

- Veritabanı istatistikleri güncellendi.

- Fragmentasyon kontrol edilerek optimize edildi.

### ****Ek Notlar****

- Gerçek sistemlerde SQL Server Profiler yerine "Extended Events" tercih edilir (daha hafif).

- Otomatik bakım görevleri ile indeks ve istatistikler düzenli güncellenebilir.

-Sorguların SET STATISTICS TIME ON ve IO ON ile performansı test edilebilir.

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

### ****Amaç****

Veri kayıplarına karşı önlem almak, sistemin çökmesi veya bozulması durumunda veritabanını güvenli şekilde geri yükleyebilmek.

Bu plan sayesinde :

Veritabanı her durumda kurtarılabilir hale gelir.

Geri dönüş süresi kısalır (RTO).

Veri kaybı minimuma iner (RPO).

### ****Teorik Bilgi****

Felaketten kurtarma (Disaster Recovery - DR) planları genellikle şu bileşenlerden oluşur :

**-- Yedekleme Türleri** :

**- Full Backup (Tam Yedek)** : Veritabanının tümünü içerir.

**- Differential Backup (Fark Yedeği)**: Son tam yedekten sonraki değişiklikler.

**- Transaction Log Backup**: En küçük değişiklikleri bile kaydeder.

**-- Yedekleme Stratejileri** :

- Günlük Full + saatlik Log yedeği

- Farklı fiziksel lokasyonlarda kopya yedekler

- Otomatik zamanlama (SQL Server Agent ya da Task Scheduler)

**-- Kurtarma Stratejileri** :

**- Point-in-Time Restore**: Belirli bir zamandaki duruma geri dönebilme

**- Database Mirroring / Log Shipping / AlwaysOn**: Anlık kopya ile failover

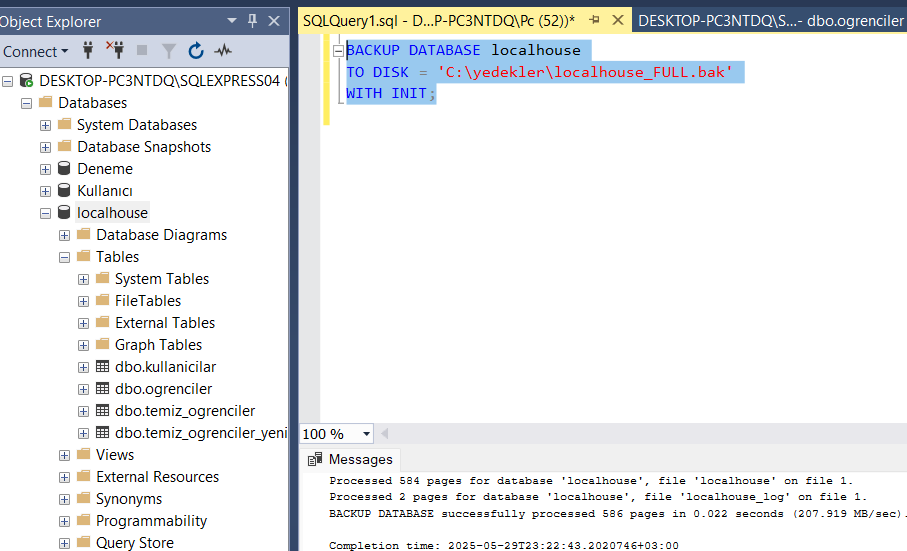
**- Failover Testleri**: Senaryolarla test yapılması

### ****Senaryo :****

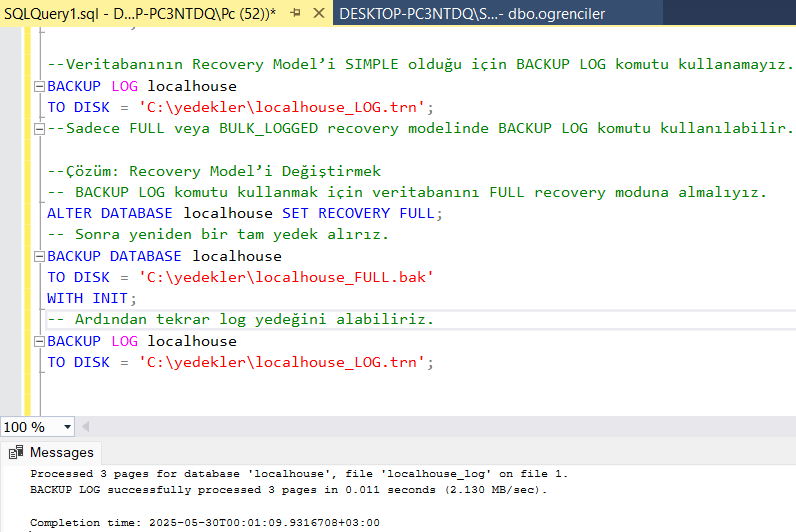
Okul sisteminin veritabanı bozuldu. En son 03:00’te alınan yedeğe geri dönülmek isteniyor. Amaç: Sistem 15 dakika içinde eski haline gelsin ve maksimum 1 saatlik veri kaybı olsun.

**Uygulama Adımları ve SQL Kodları :**

#### **Full Yedek Alma :**



1. **Transaction Log Yedeği Alma :**



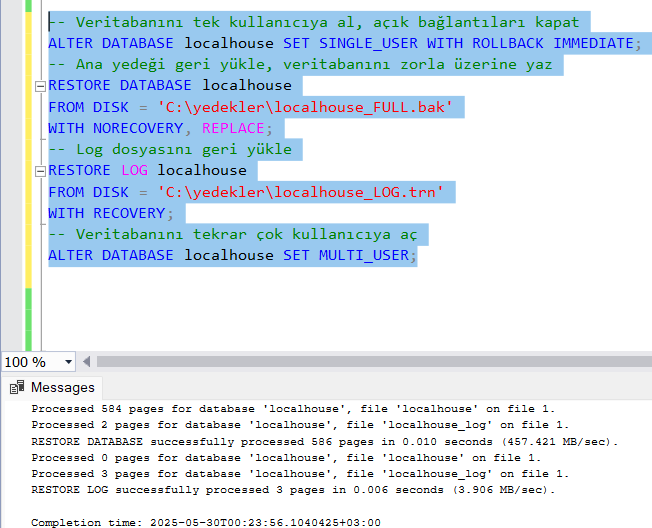
**3. Geri Yükleme (Restore) :**

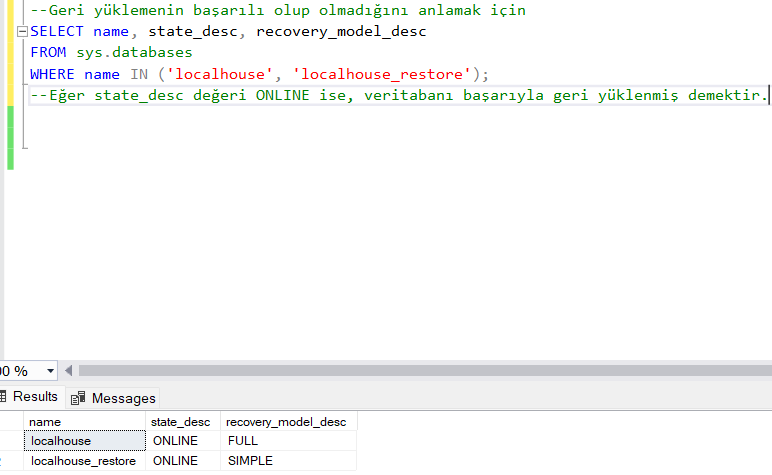
Diyelim ki sistem çöktü ve localhouse veritabanını geri yüklemek istiyoruz veya test yapmak istiyoruz

“Point-in-time” geri dönüşü simüle edeceğiz.

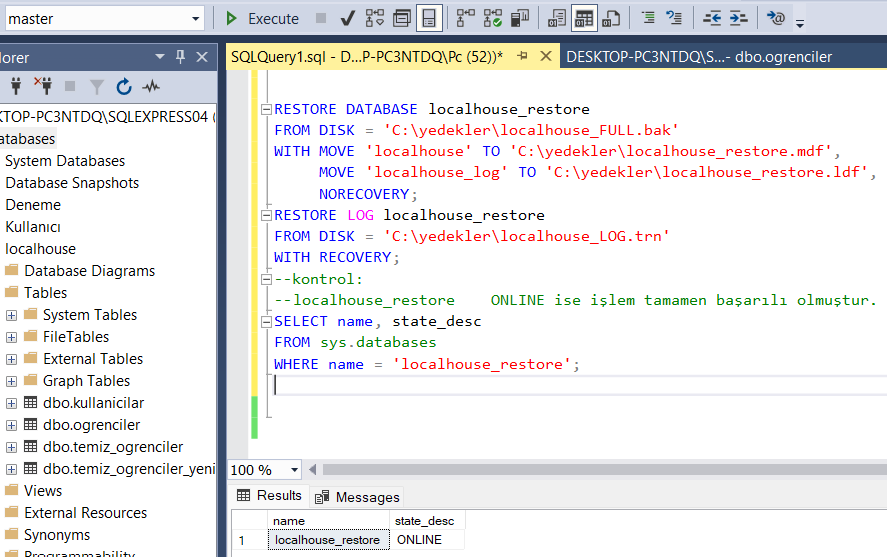
### ****A) Aynı isimle restore (mevcut veritabanı üzerine)****

Bu durumda veritabanını geçici olarak tek kullanıcı moduna alırız.





**B) Yeni isimle restore (orijinali bozmadan test amaçlı)**



1. **Veritabanı Güvenliği ve Erişim Kontrolü**

## ****Amaç :****

Verilerin yetkisiz erişime karşı korunması, kullanıcı rollerinin sınırlandırılması ve olası SQL Injection saldırılarına karşı önlem alınması.

Veritabanı güvenliği, kullanıcı verilerinin gizliliğini ve bütünlüğünü korumak için alınan önlemleri kapsar. En yaygın tehditlerden biri SQL injection saldırılarıdır. Ayrıca kullanıcıların yetkilerini sınırlandırmak ve erişim denetimi sağlamak için rol tabanlı izinler tanımlanmalıdır.

**Temel başlıklar :**

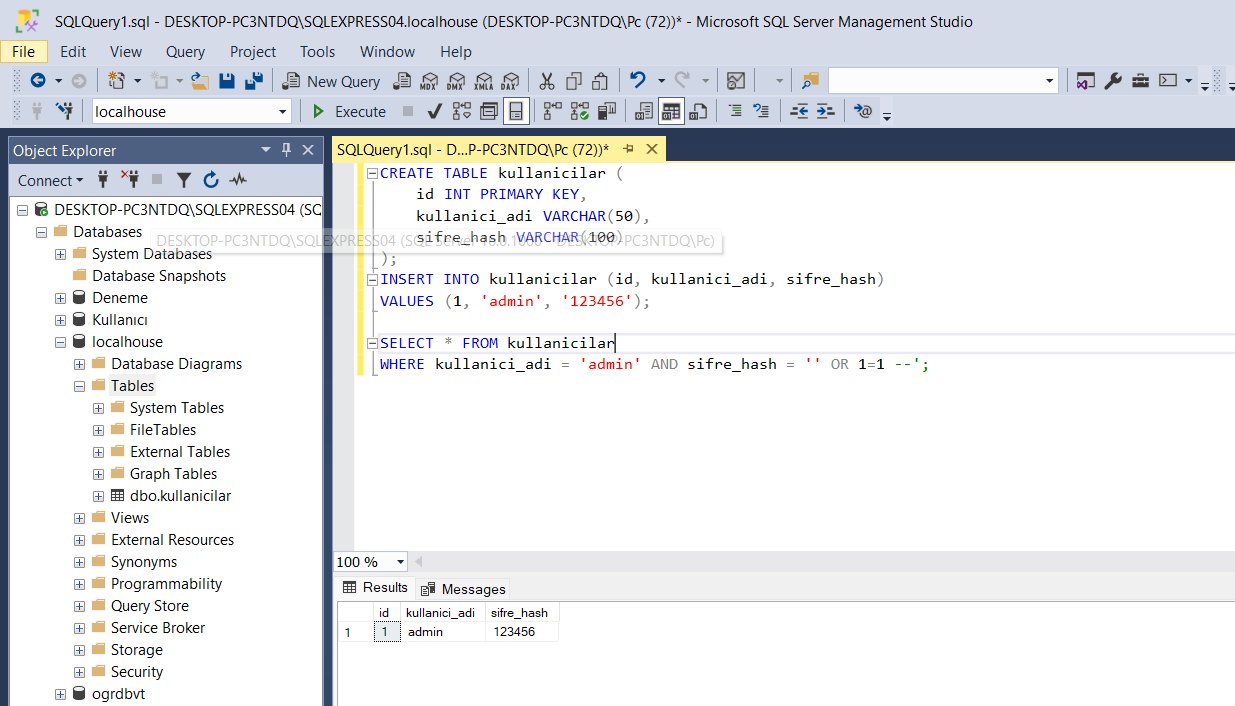
- SQL Injection ve korunma yolları

- Yetkilendirme (GRANT, REVOKE)

- Kullanıcıların rollerle sınırlandırılması

- Hassas verilerin şifrelenmesi (örn. şifre alanları)

(SQL Injection Saldırı Örneği) Tüm kullanıcılar döner, güvenlik açığı oluşur. :

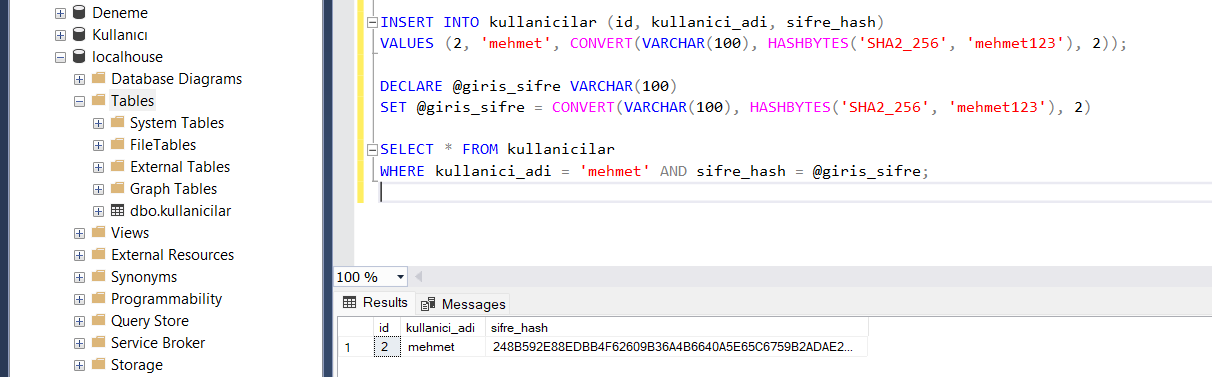


**Çözüm :**

Giriş sistemi, dışarıdan gelen değerleri filtrelemeli. Kodun bu kısmı genelde yazılım tarafından parametreli gönderilir.

Şifreleri güvenli saklama (T-SQL ile hashleme)

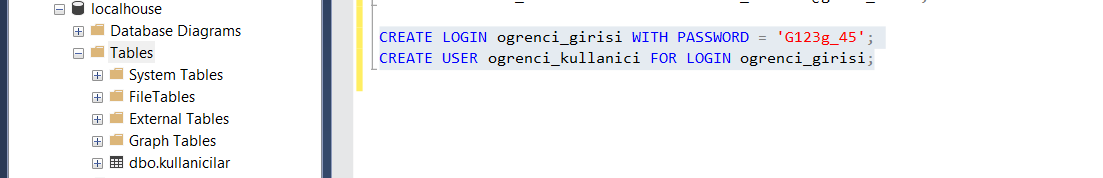
Yeni kullanıcı eklerken HASHBYTES ‘le, şifreyi SHA-256 ‘le saklayalım



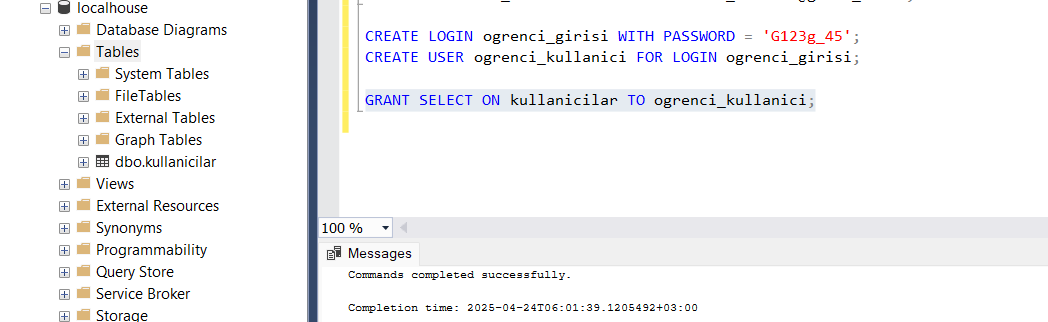
(Hash fonksiyonuyla şifre saklama ve sonrasında da giriş kontrolü)

Bu projede veritabanı güvenliğinin temel prensipleri uygulanarak, hem kullanıcı erişimleri sınırlandırılmış hem de şifrelerin güvenli biçimde saklanması sağlanmıştır. Ayrıca SQL injection gibi saldırı türlerine karşı örnekli çözümler gösterilmiştir.

**Rol Tabanlı Erişim Örneği :**



Sadece veri görme izni verelim :



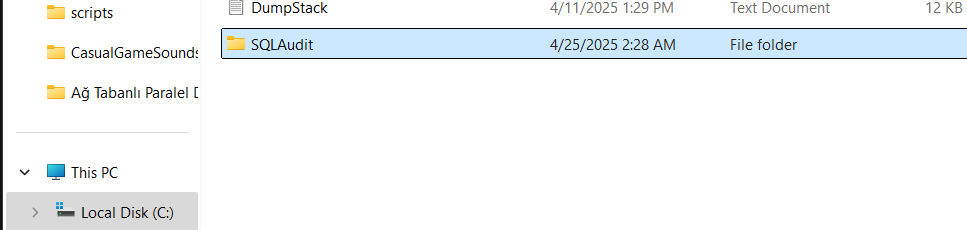
**Ek Notlar :**

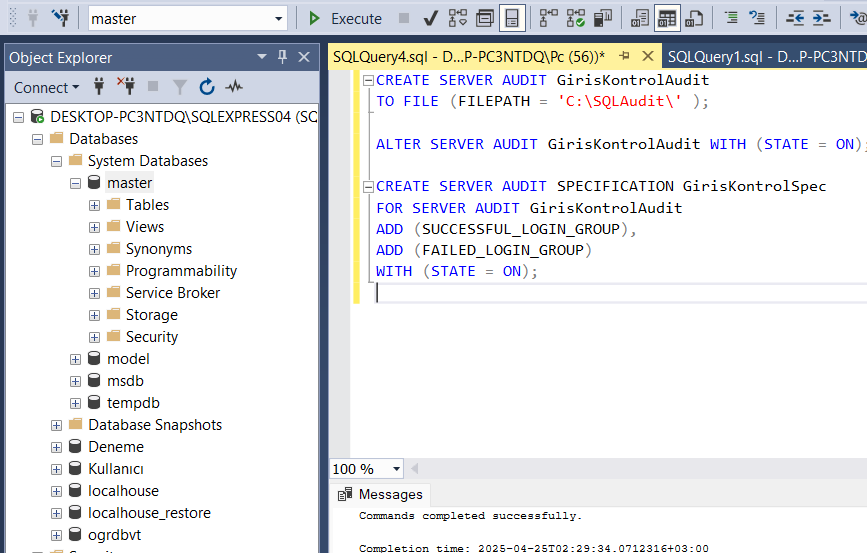
Gerçek projelerde veritabanına erişim öncesi input validation ve parametreli sorgular (prepared statements) kullanılmalıdır.

Şifreleme işlemi tek yönlü hashleme ile yapılmalı, çözülemeyen algoritmalar tercih edilmelidir.

**SQL Server Audit Özelliği**

Eğer güvenlik kısmında kimin ne zaman ne işlem yaptığını izlemek istiyorsak, bunun için SQL Server Audit kullanırız.





Bu kodla :

- Audit nesnesi oluştur.

- Başlat

- Audit Specification (örnek: login işlemlerini izle)

Bu komutlarla başarılı ve başarısız login denemeleri izlenir. Loglar C:\SQLAudit klasörüne kaydedilir. Varsa D sürücüsü de kullanılabilir.

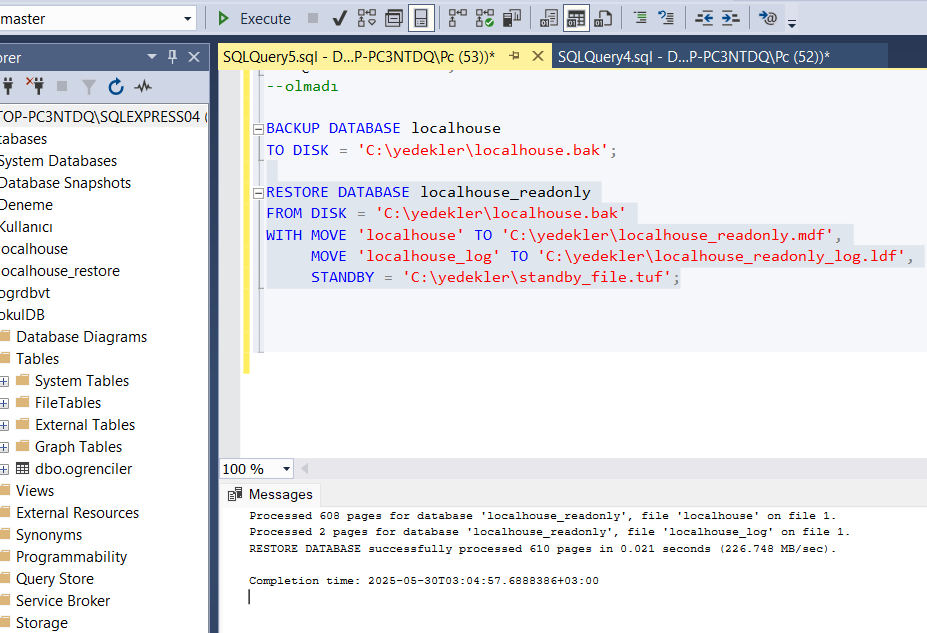
Audit dosyalarını sadece yetkili kullanıcılar görebilir.

1. **Veritabanı Yük Dengeleme ve Dağıtık Veritabanı Yapıları**

**Yapı :**

### Primary Server- Mirror Server - Witness Server (opsiyonel)

Replikasyon gibi **yük dengeleme ve dağıtık veritabanı** uygulamalarını, **SQL Server Express Edition** kullanarak doğrudan yapamayız ama bazı **alternatif çözümler** var.



- Ana veritabanı (localhouse) yedeği alındı.

- Salt okunur olarak localhouse\_readonly adında yedek başka sunucuya (veya kopya olarak) geri yüklendi.

- SELECT işlemleri bu kopyadan yapılabilir.

- Böylece okuma işlemleri dağıtılmış oldu → Yük dengeleme **uygulanmış oldu.**

**5.Veri Temizleme ve ETL Süreçleri Tasarımı**

**Amaç :**  
Dağınık ve eksik veri kaynaklarını temizleyerek anlamlı hale getirmek, dönüşümden geçirip hedef sisteme aktarmak.

**Teorik Bilgi :**  
ETL (Extract - Transform - Load) süreci, veri ambarları ve analiz sistemleri için temel yapı taşlarından biridir:

**- Extract (Çıkarma) :** Verilerin kaynak sistemlerden alınması

**- Transform (Dönüştürme):** Verilerin temizlenmesi, biçimlendirilmesi, dönüştürülmesi

**- Load (Yükleme):** Dönüştürülen verilerin hedef sisteme yüklenmesi

Veri Temizleme işlemleri genellikle aşağıdaki işlemleri içerir:

- Boş alanların tespiti ve temizlenmesi

- Hatalı formatların düzeltilmesi

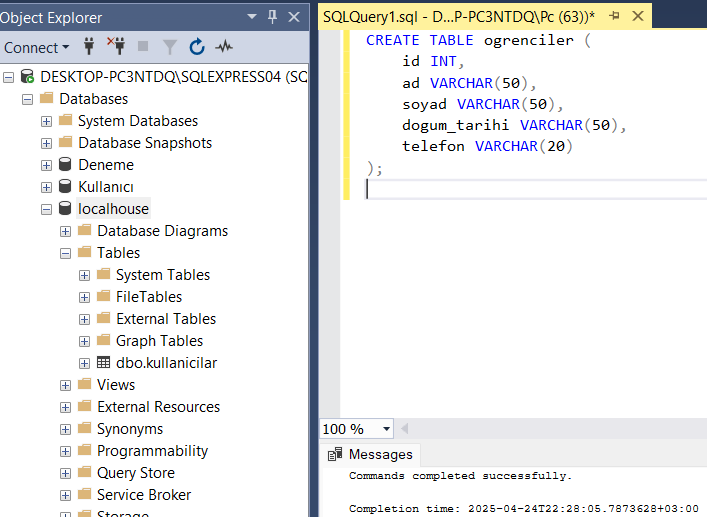
- Tekrarlayan kayıtların ayıklanması

- Uyuşmayan tarih/sayı/veri tiplerinin düzeltilmesi

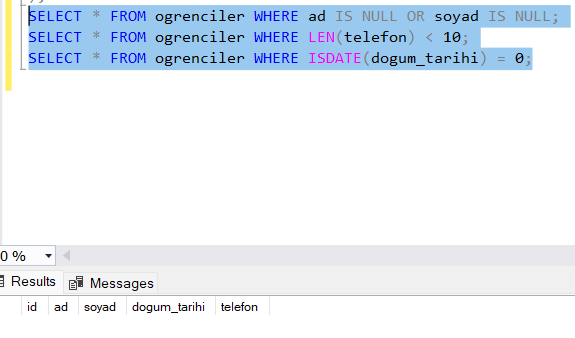
**Senaryo :**  
Bir okul sisteminden alınan öğrenci bilgilerinin yer aldığı tablo hatalı ve eksik veriler içermektedir.  
Bu veriler temizlenip düzgün formata getirilerek raporlama sistemine aktarılacaktır.

**Uygulama Adımları ve SQL Kodları:**

1. **Kaynak Tablo Örneği (ogrenciler)**



1. **Hatalı ve Eksik Verilerin Tespiti**

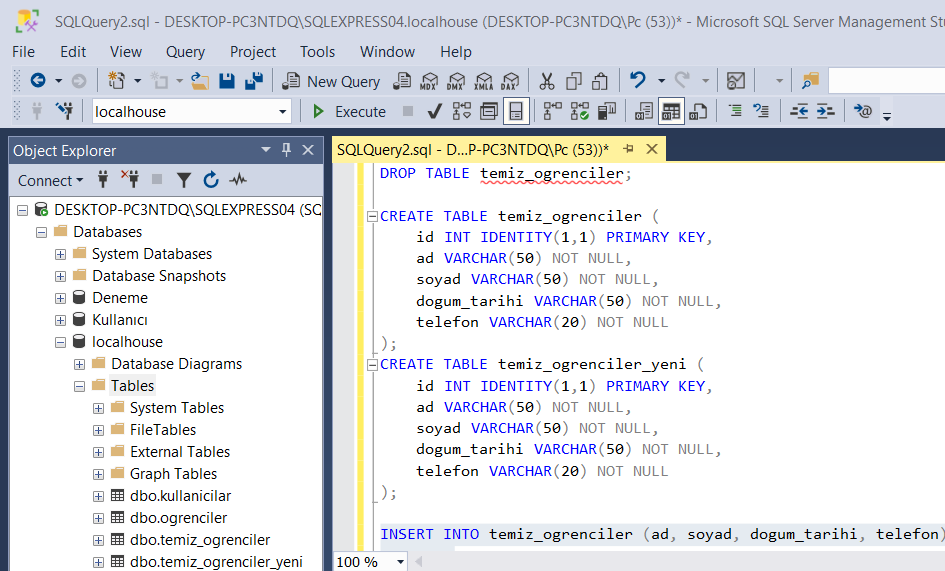


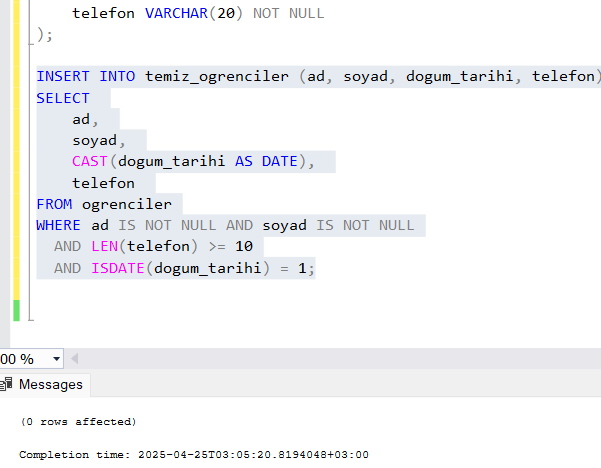
- Boş ad veya soyad

- Geçersiz telefon uzunluğu

- Geçersiz tarih formatı kontrolü (örnek)

1. **Temiz Verileri Yeni Tabloya Aktarma (LOAD Kısmı)**





Geçerli verileri taşıma

**Sonuç :**  
Bu projede veri temizliği gerçekleştirilmiş, hatalı ve eksik kayıtlar filtrelenmiş, düzgün veriler hedef tabloya yüklenmiştir. ETL süreci sayesinde veri analizi için daha anlamlı ve tutarlı bir veri kümesi elde edilmiştir.

**Ek Notlar :**

- Gerçek projelerde veri temizliği SSIS (SQL Server Integration Services) gibi araçlarla otomatikleştirilebilir.

- Temiz veri, doğru raporlamanın temelidir.

- Bu işlemler büyük veri ortamlarında hayati öneme sahiptir.

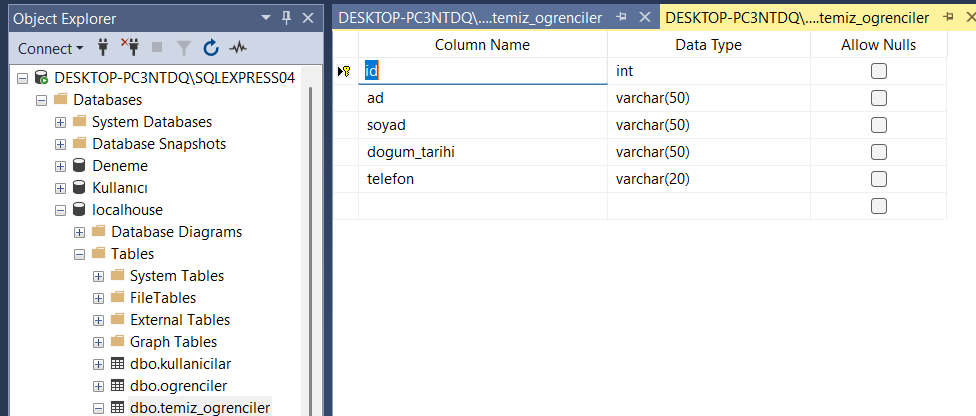
**Örnek :**

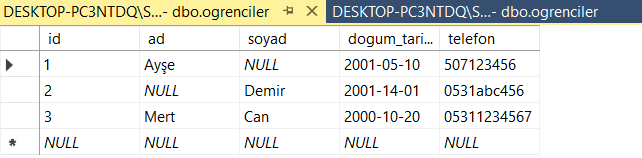
**ETL Öncesi - Önceki Veri (ogrenciler)**

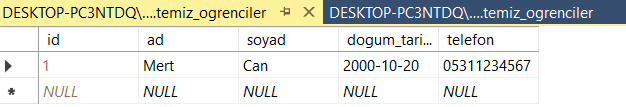
| **id** | **ad** | **soyad** | **dogum\_tarihi** | **telefon** |
| --- | --- | --- | --- | --- |
| 1 | Ayşe | NULL | 2001-05-10 | 507123456 |
| 2 | NULL | Demir | 2001-14-01 | 0531abc456 |
| 3 | Mert | Can | 2000-10-20 | 05311234567 |

**ETL Sonrası - Temizlenmiş Veri (temiz\_ogrenciler)**

| **id** | **ad** | **soyad** | **dogum\_tarihi** | **telefon** |
| --- | --- | --- | --- | --- |
| 3 | Mert | Can | 2000-10-20 | 05311234567 |





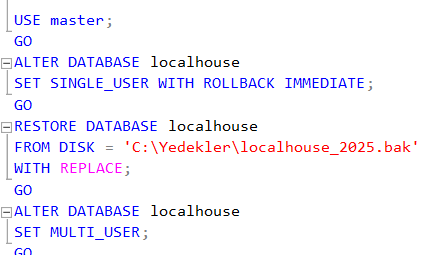
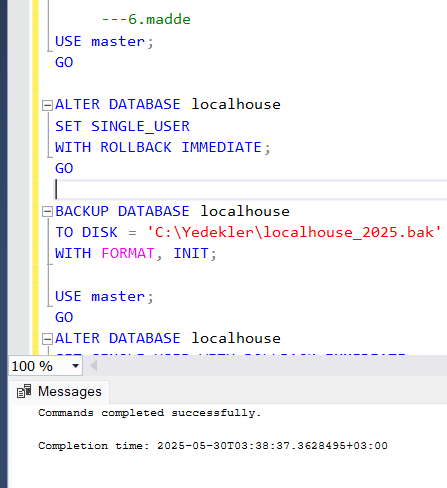


## ****6.Veritabanı Yükseltme ve Sürüm Yönetimi****

### ****Amaç:****

Veritabanı sistemlerinin daha güncel, hızlı ve güvenli hale getirilmesi amacıyla mevcut bir veritabanının yeni bir SQL Server sürümüne taşınması ve bu süreçte oluşabilecek risklere karşı yedekleme ve geri yükleme stratejilerinin uygulanması hedeflenmiştir.

1. Veritabanını Tek Kullanıcı Moduna Alma
2. Yedek Alma (Upgrade öncesi)
3. Restore İşlemi
4. Veritabanını Çok Kullanıcılı Hale Getirme :



### ****Sonuç ve Kazanımlar :****

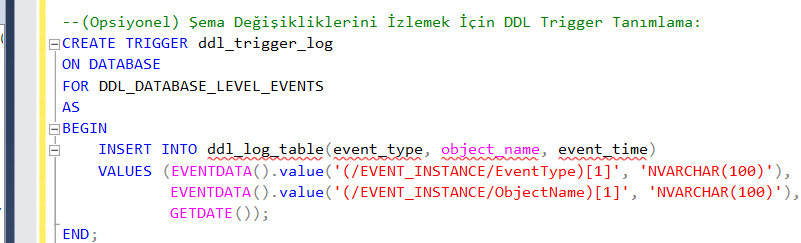
Veritabanı yükseltme süreci sorunsuz olarak tamamlandı.

Herhangi bir veri kaybı yaşanmadan yedekleme ve geri yükleme işlemleri başarıyla uygulandı.

Sistem geçici olarak tek kullanıcı moduna alınarak işlem güvenliği sağlandı.

Restore işlemiyle birlikte yeni sürüm geçişi test edildi ve sistem stabil şekilde çalıştı.

5.(Opsiyonel) Şema Değişikliklerini İzlemek İçin DDL Trigger Tanımlama :



6.Test ve Geri Dönüş Planı :

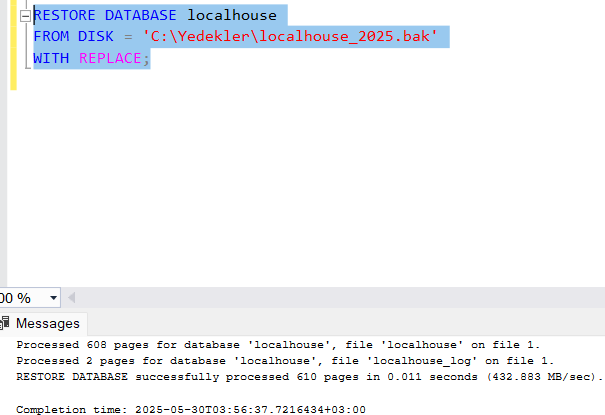
Veritabanını yeni bir sürüme geçirdik veya üzerinde büyük bir değişiklik yaptık. (örneğin tablo eklemek, veri temizlemek, trigger tanımlamak vs.). Bu işlemden **önce** bir **full yedek** almıştık. Şimdi test yaptık ama sistemde sorun çıktı. İşte o zaman "geri dönüş" yaparız.

### ****- İşlem Öncesi Yedek Al:****

Zaten yaptık.

### - ****Yeni Sürümü Test Et:****

Veritabanını restore ederek test ederiz.



### ****- Eğer Sorun Varsa Geri Dön :****

Diyelim ki yapılan yükseltme sonrası hatalar oluştu, bazı tablolar bozuldu.

Yapmamız gereken şey : **önceki yedeği tekrar yüklemek.**  
Örneğin önce şu yedeği almıştık :

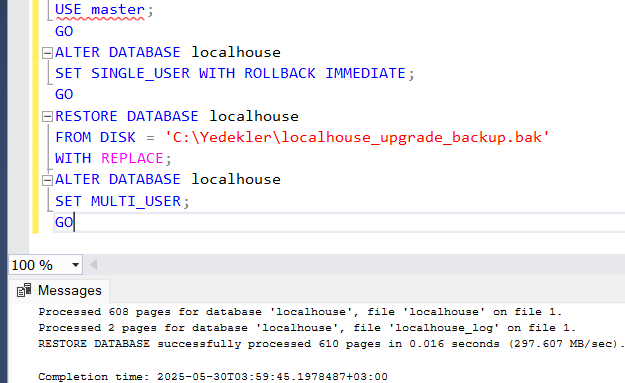
C:\Yedekler\localhouse\_upgrade\_backup.bak

Ona geri dönmek istersek :

-- Veritabanını tek kullanıcı moduna al

-- Önceki sağlam yedeği geri yükle

-- Veritabanını tekrar çok kullanıcıya aç



Yani **eski sağlam yedeğe dönerek sistemi kurtarırız**.

Bu sayede sistem kararlı bir şekilde eski haline döndürülmüş ve veri güvenliği sağlanmıştır.

1. **Veritabanı Yedekleme ve Otomasyon Çalışması**

**Amaç :**

- Verilerin kaybolma riskine karşı korunması

- Yedekleme işlemlerinin otomatik hale getirilmesi

- Sistem yöneticisinin iş yükünün azaltılması

Özetle : Güvenli ve sürdürülebilir veri yönetimi

### ****Yedekleme Türleri :****

**- Tam Yedek (Full) :** Veritabanının tümü

**- Fark Yedeği :** Son tam yedekten sonra değişen kısımlar

**- Transaction Log :** Her değişiklik adım adım izlenir

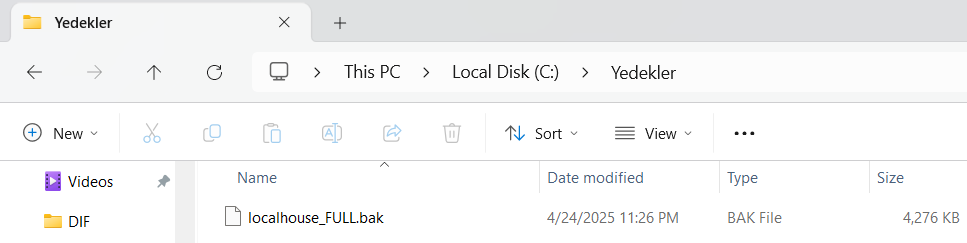
Bu yedekleme türleri birlikte kullanılabilir.

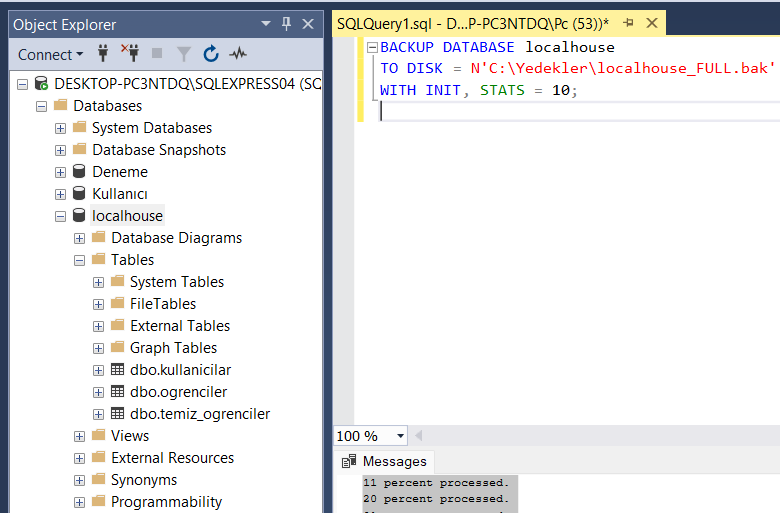
### ****Senaryo :****

“Her gece saat 02:00'de okul veritabanı otomatik olarak yedeklenmelidir.”  
Sistem yöneticisi manuel işlem yerine bir otomasyon kurar.

**Yedekleme SQL Komutu :**

SQL ile doğrudan tam yedek alınır.





### ****SQL Server Agent ile Otomasyon :****

1. SQL Server Agent → New Job
2. Job adı : **Gece\_Yedekleme**
3. Step : BACKUP DATABASE komutu
4. Zamanlama : **02:00** günlük
5. Job aktif hale gelir → Otomatik yedekleme başlar

### ****Kazanımlar ve Sonuç :****

- Veri kaybı riski azalır

- Süreklilik ve güvenlik sağlanır

- Zaman kazancı elde edilir  
Otomasyon, sistem yönetiminde verimliliği artırır.

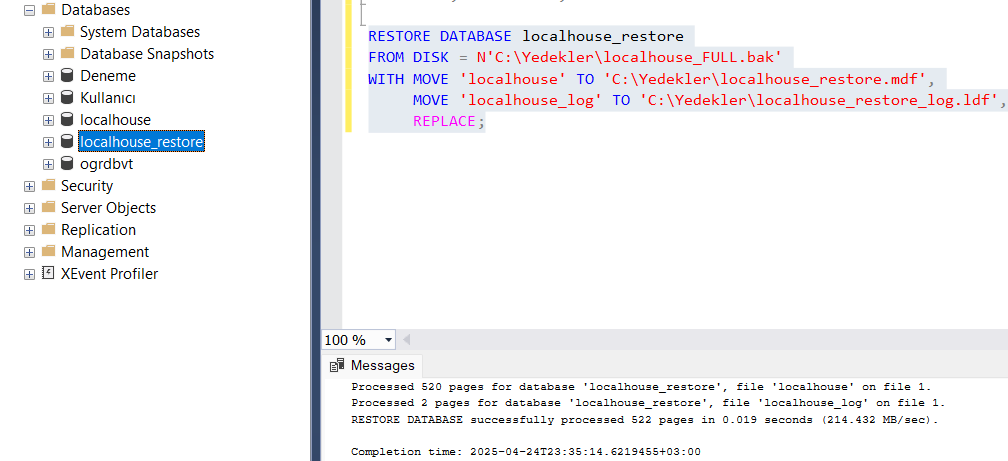
### ****Ek Bilgiler :****

- Yedeklerin başka diskte/uzakta tutulması önerilir

- RESTORE DATABASE işlemi test edilmelidir

- PowerShell veya Batch Script ile gelişmiş otomasyon yapılabilir

-- localhouse\_FULL.bak dosyası varsa, **Geri yükleme (restore) testi** yapabiliriz. :



Veritabanını geri yüklemek için (başka bir isimle istersen, mesela “localhouse\_restore”)

**Not :** Veritabanı açıkken RESTORE işlemi çalışmaz, ya önce bağlantıları kapatmalıyız ya da başka bir adla yüklemeliyiz.

**Otomasyon**

Belirli bir zaman aralığında sistemin kendi kendine bir işi yapmasıdır.

Ve burada o iş: **Veritabanını yedekle.**

**otomasyon kısmı = zamanlama + otomatik çalıştırma** diyebiliriz.

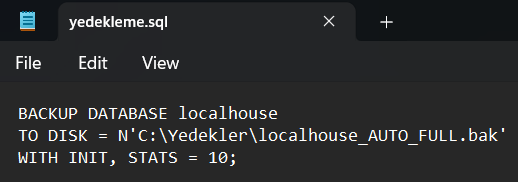
### ****Otomasyonun İçeriği :****

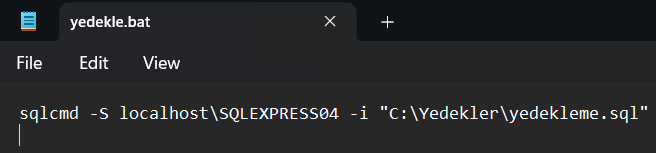
SQL Server Agent'ın kullanılmadığı Express sürümüne göre alternatif çözüm : (Task Scheduler + .bat)

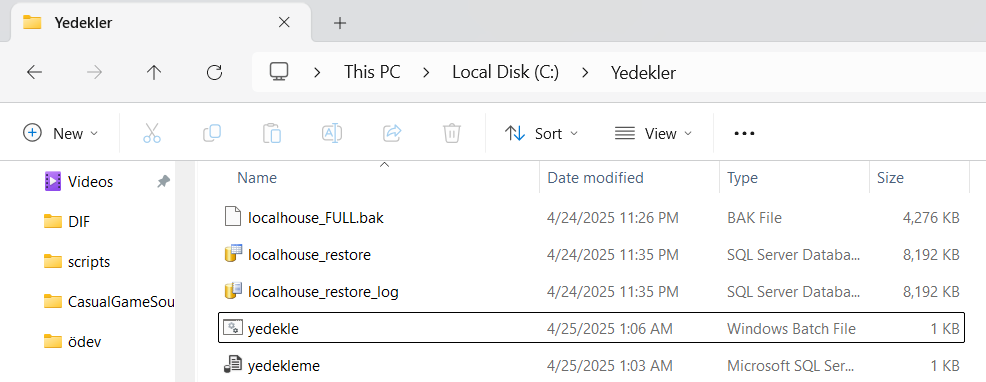
1. **SQL komutunu** bir .sql dosyasında saklamak (.sql dosyasının hazırlanması)  
   Hedef : yedekleme komutunu dışarıdan çağırabilmek

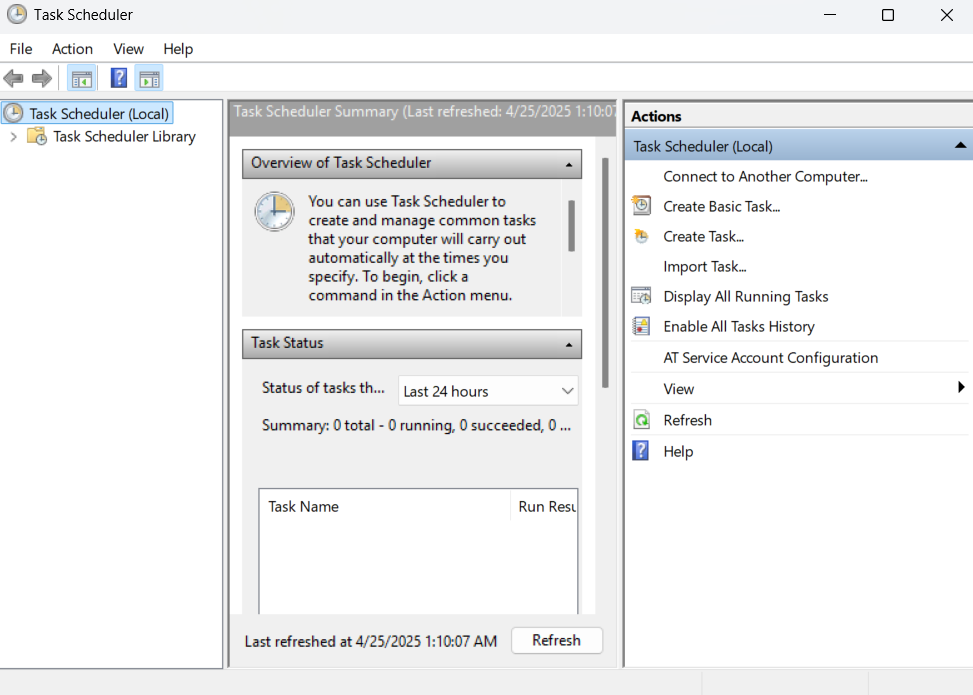
**.bat dosyası** ile bu komutu MSSQL'e göndermek (.bat ile çalıştırılması)  
Hedef : elle SSMS açmadan otomatik çalıştırmak

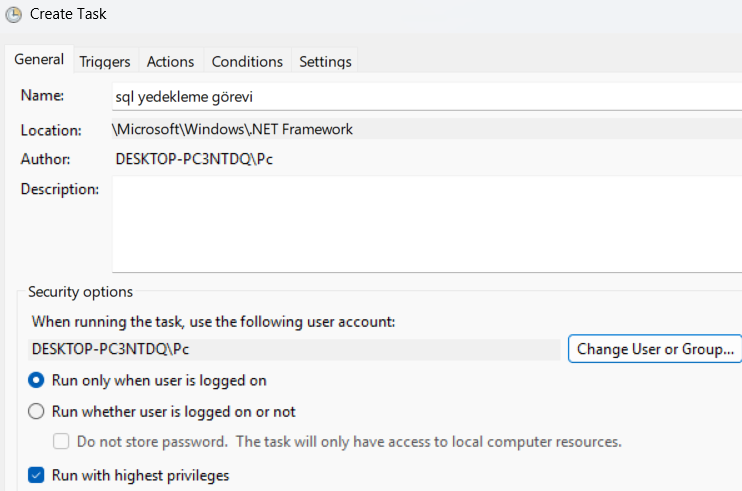
**Zamanlama (Task Scheduler) :** Görev Zamanlayıcı ile **zamanlama (schedule)** yapılmasıdır.  
Hedef : .bat dosyasını **her gün saat 02:00'de** otomatik çalıştırmak

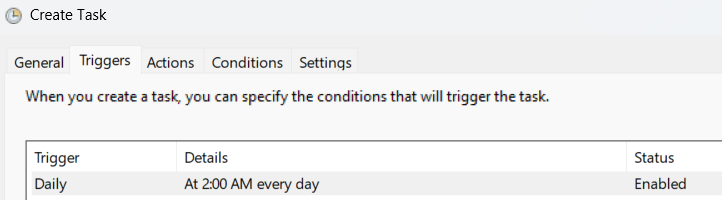


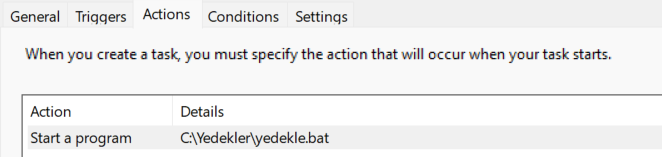


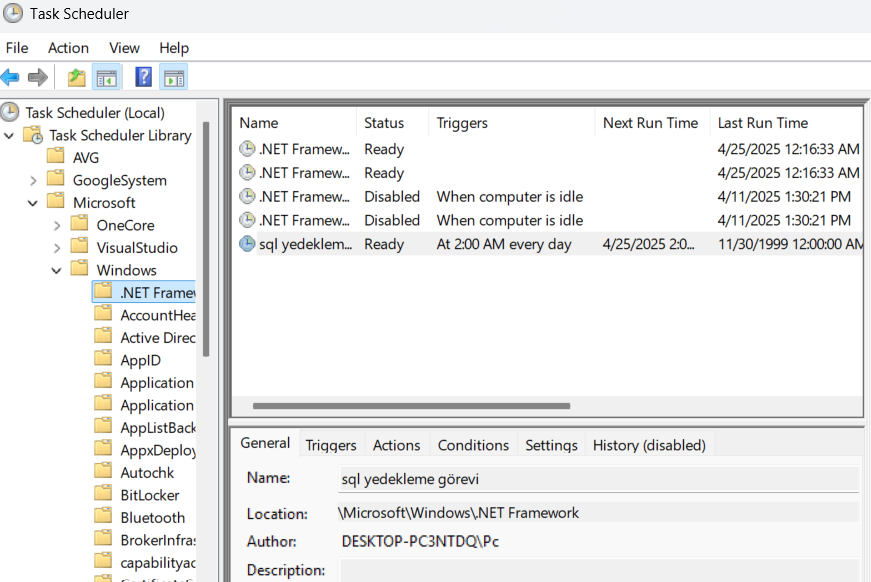






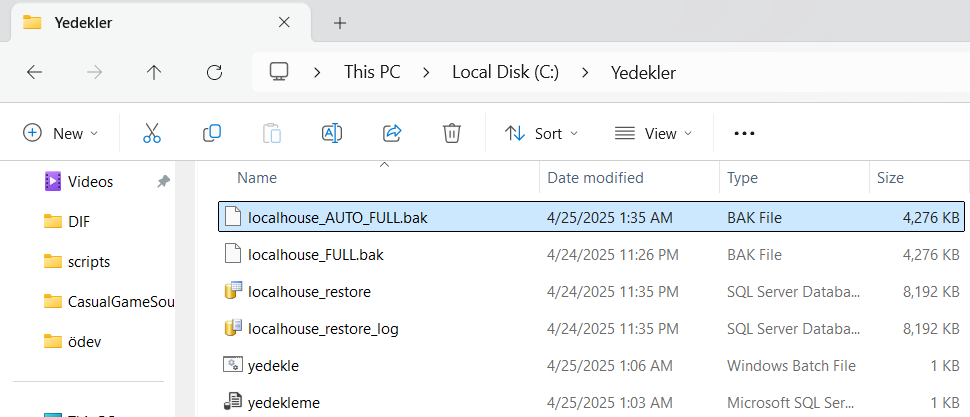


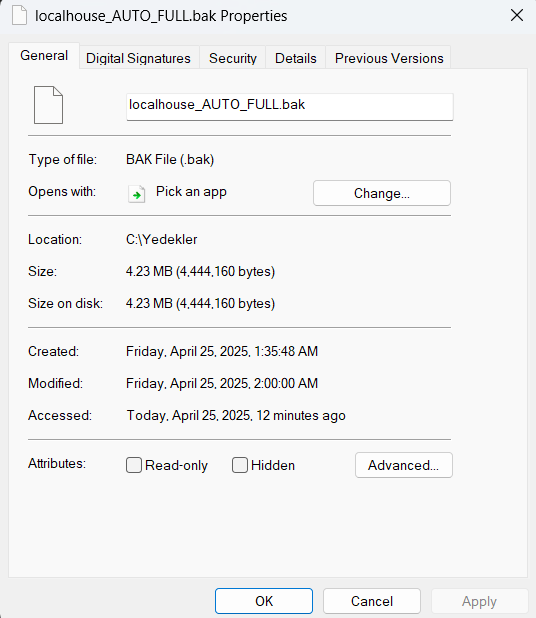




## Test Et :

## Göreve sağ tıklayıp Run (Çalıştır) diyoruz :





**Sonuç :**

Veritabanı yedekleme sürecini otomatikleştirmek amacıyla Windows Görev Zamanlayıcı (Task Scheduler) kullanılmıştır. SQL Server Agent özelliği Express sürümünde bulunmadığı için .sql komutu bir .bat dosyası aracılığıyla tetiklenmiş ve günlük olarak yedek alınması sağlanmıştır.