# ANKARA ÜNİVERSİTESİ

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

****

**FİNAL PROJE RAPORU**

**(BLM4522) AĞ TABANLI PARALEL DAĞITIM SİSTEMLERİ**

**Perizat SAGYNBEKOVA**

**21290895**

**GitHub (https://github.com/Impasbaa/MSSQL.git)**

**30.05.2025**

**PROJE 1. VERİTABANI YEDEKLEME VE FELAKETTEN KURTARMA PLANI**

**PROJE AMACI:** Veritabanı yedekleme ve felaket kurtarma planı, veri kaybını önlemek ve sistem kesintilerinde hızlı kurtarma sağlamak için kritik öneme sahiptir. Bu rapor, bir veritabanı sisteminin güvenliğini sağlamak ve veri kaybını en aza indirgemek amacıyla oluşturulan yedekleme ve felaketten kurtarma planlarını içermektedir. SQL Server kullanılarak tam, artık ve fark yedekleme stratejileri, otomatik yedekleme zamanlayıcıları, felaket senaryoları ve yedek test süreçleri detaylandırılmıştır.

**HEDEF:** Veri kaybını önlemek; Kesinti süresini en aza indirmek; Yedeklerin güvenilirliğini test etmek; Uyumluluk gereksinimlerini karşılamak.

* **Veritabanı Yedekleme,** dosyaların periyodik olarak anlık görüntülerini alarak bir kopyasını oluşturma işlemidir.
* **Felaketten Kurtarma Planı**, bir organizasyona kesintiden sonra (felaket sonrasında) tüm hayati altyapıları, veritabanlarını, uygulamaları ve hizmetleri kurtarma yeteneği kazandıran süreçler, politikalar, prosedürler ve temel ölçümlerin birleşimidir.

**1. GİRİŞ**

* 1. **YEDEKEME STRATEJİLERİ**

**1.1.1. Tam Yedekleme (Full Backup)**

Genellikle günlük veya haftalık periyotlarla büyük veri değişikliklerinden önce veya sonra tüm veritabanının bir kopyasını alır. En hızlı geri yükleme seçeneğidir ve tek başına kullanılabilir fakat fazla depolama alanı gerektirir.

**1.1.2. Artık Yedekleme (Transaction Log Backup)**

Tam yedekten sonra yapılan tüm işlemleri kaydeder. Felaket anında “point-in-time restore” ile belirli bir saniyeye kadar geri dönüş sağlanır ve sıklıkla (örneğin her 15 dakikada bir) yapılması önerilir.

**1.1.3. Fark Yedekleme (Differential Backup)**

Tam yedekleme ile birlikte kullanılmakta ve son tam yedekten sonra değişen verileri yedekler. Avantajı ise disk alanı kullanımı daha azdır.

* 1. **ZAMANLAYICILARLA OTOMATİK YEDEKLEME**

**1.2.1. SQL Server Agent Kullanımı**

SQL Server Agent ile yedekleme işleri zamanlanabilir. Örnek:

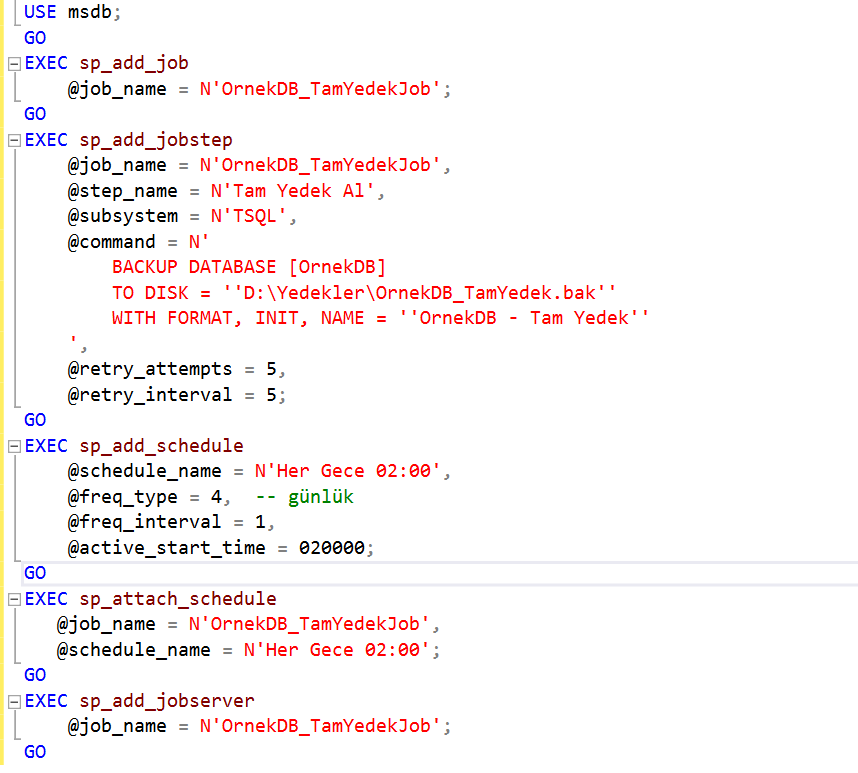
* Tam yedekleme: Her gece saat 02:00
* Artık (ya da artırımlı) yedekleme: Her 15 dakikada bir
* Fark yedekleme: Günde 2 kez (12:00 ve 18:00)

**1.2.2. Script Örneği (Tam Yedekleme)**

metin, ekran görüntüsü, yazılım, web sayfası içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

*Adım adım otomatik tam yedekleme Job’u:*



* 1. **FELAKETTEN KURTARMA SENARYOLARI**

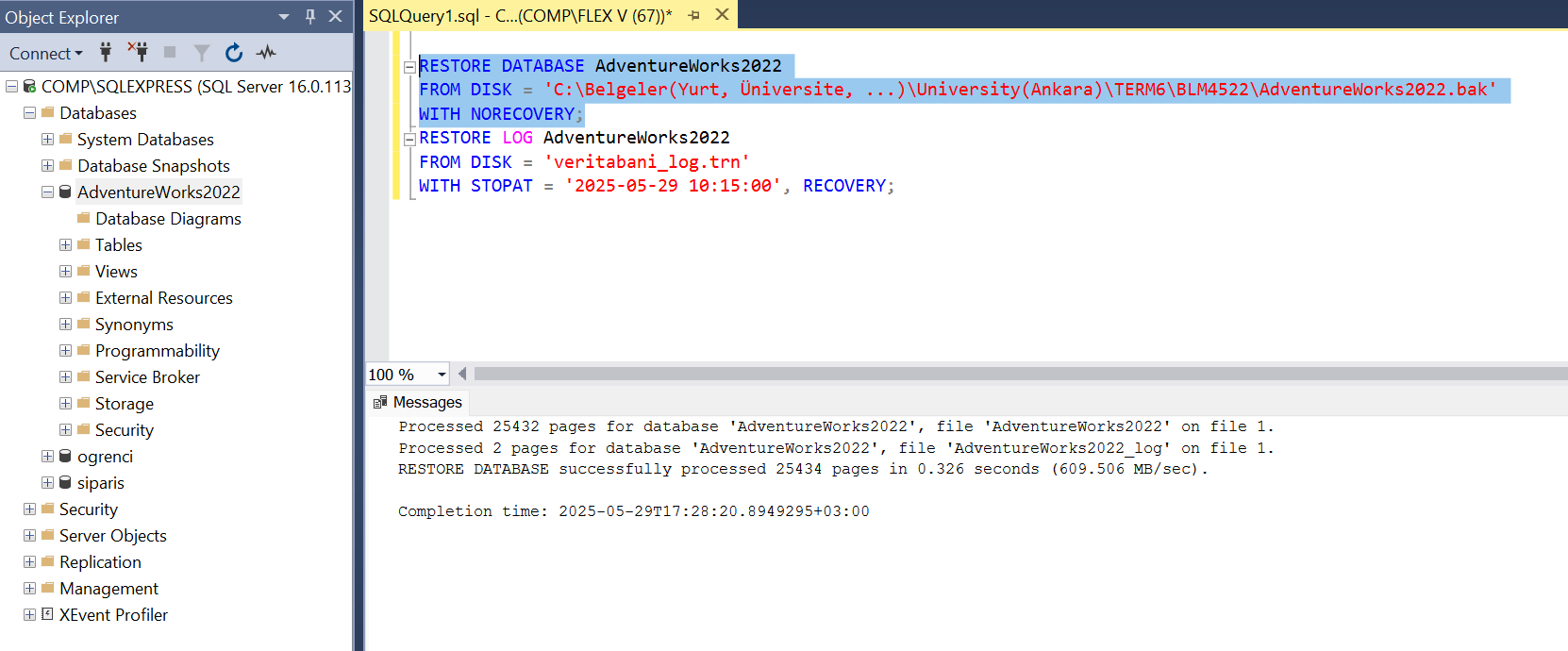
**1.3.1. Kaza ile Silinen Veriler (Veritabanını Belirli Bir Zamana Geri Yükleme – Point-in-Time Restore)**

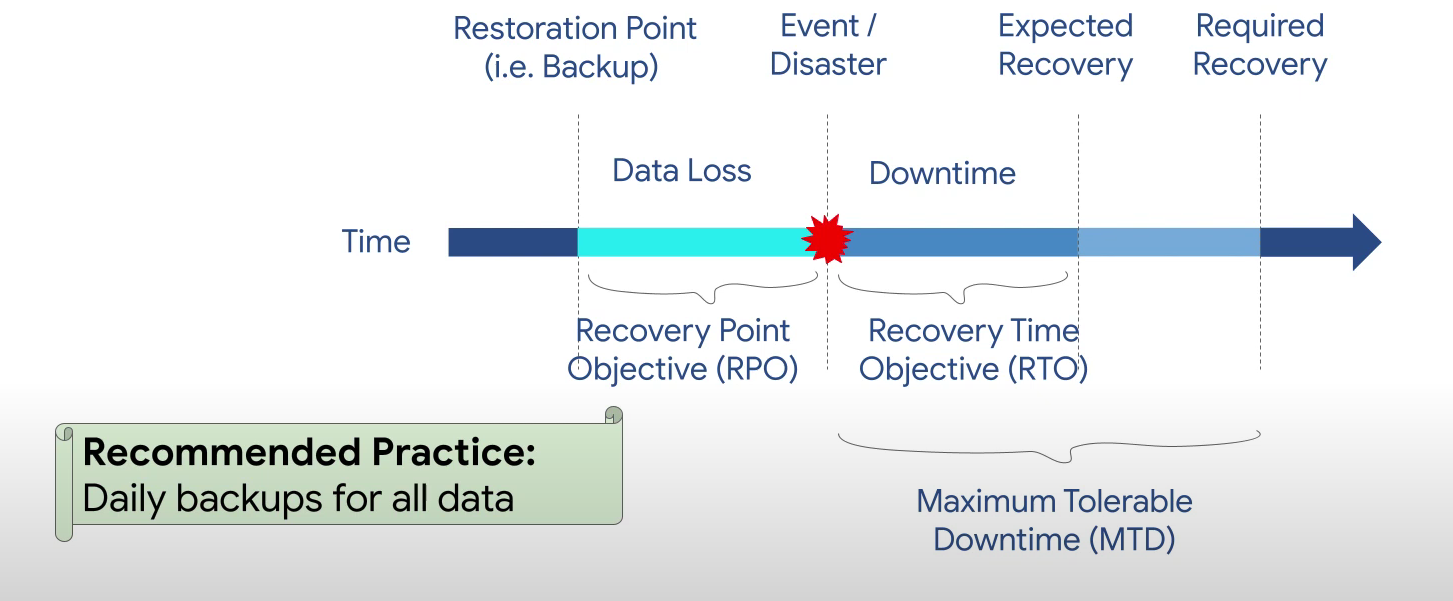
Transaction log yedekleri kullanılarak belirli bir zamana geri dönüş yapılabilir.

Senaryo: Bir kullanıcı yanlışlıkla önemli bir tabloyu sildi.

Çözüm: En son tam yedekten geri yükle. İşlem günlüğünü (log backup) silme işleminden önceki bir zamana kadar uygula.

*Point-in-time restore örneği:*





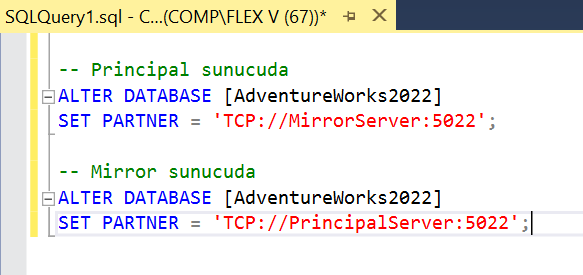
**1.3.2. Donanım Arızası**

Database mirroring veya log shipping gibi çözümlerle yedek sunucuya geçiş sağlanabilir.

* 1. **VERİTABANI YANSITMA (DATABASE MİRRORİNG)**

Senaryo: Sunucu çöktü, hızlı bir şekilde yedek sunucuya geçiş yapılmalı.

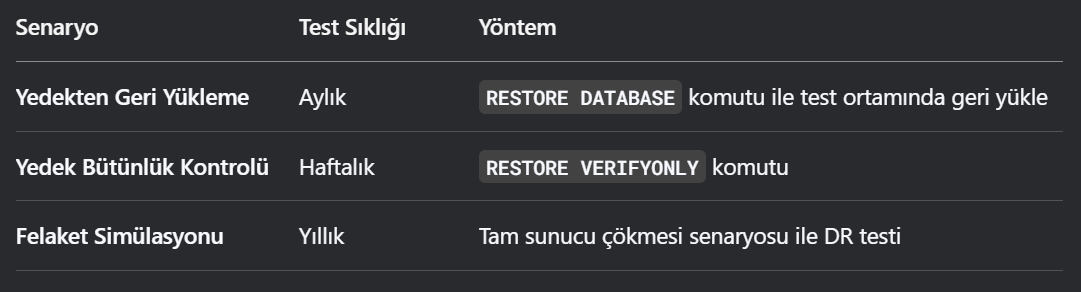
Çözüm: Principal (Ana sunucu) ve Mirror (Yedek sunucu) arasında veritabanı aynalama kurulumu yapılır.



* 1. **TEST YEDEKLEME SENARYOLARI**

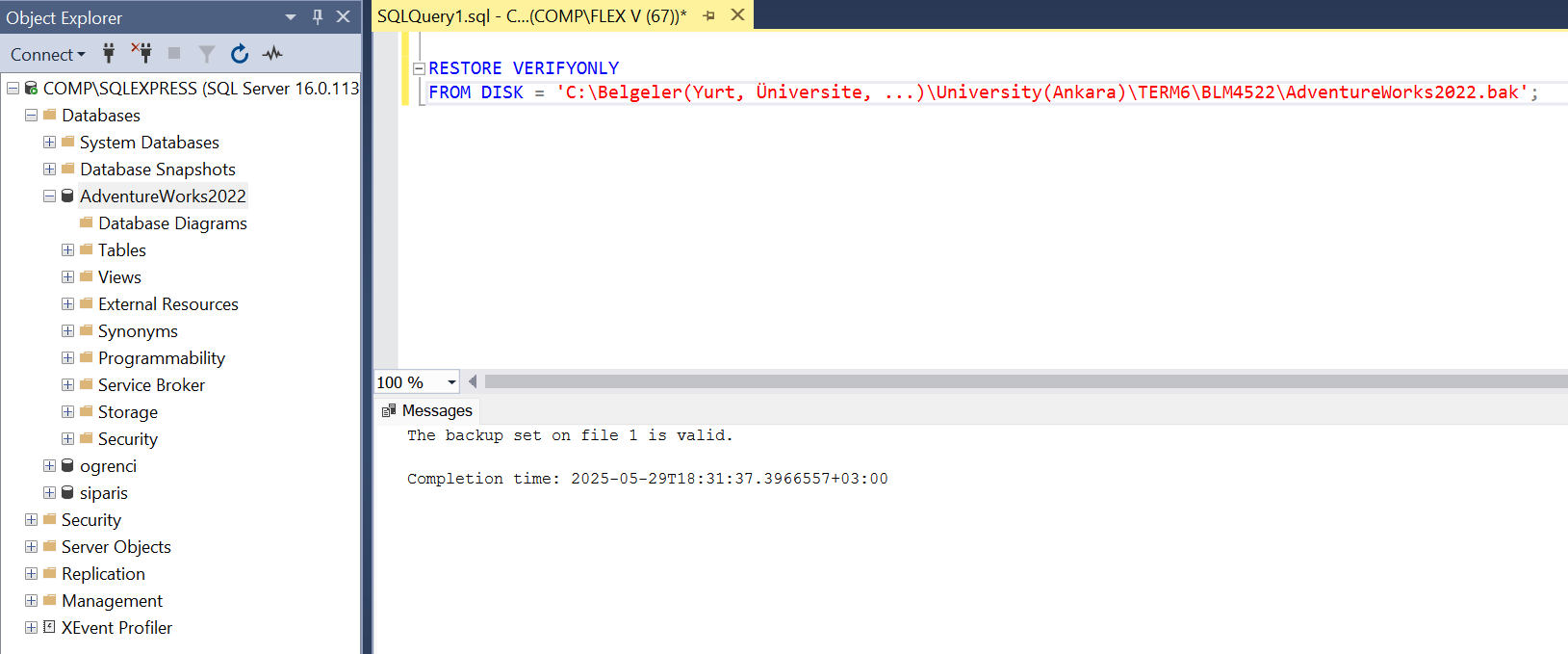
**1.5.1. Yedeklerin Doğruluğunu Test Etme**

Periyodik olarak yedekten geri yükleme testleri yapılmalıdır ve test sunucularında yedeklerin başarıyla yüklenip yüklenmediği kontrol edilmelidir.



**1.5.2. Script Örneği (Test Yükleme)**

*Yedek bütünlük kontrolü:*



**SONUÇ ve ÖNERİLER**

* Her zaman 3-2-1 kuralı uygulanmalı: 3 kopya (birincil + 2 yedek); 2 farklı ortam (örneğin, disk + bulut); 1 off-site yedek (fiziksel felaketlere karşı).
* Büyük veritabanlarında yedekleme çok zaman alıyorsa, **Compressed Backup** kullanılmalı: WITH COMPRESSION.
* Yedekler tek yerdeyse, farklı fiziksel disklere veya bulut ortamına da yedek alınabilir.
* Yetkisiz erişim riski varsa, yedek dosyalarını şifrelenmeli: WITH ENCRYPTION (SQL Server Enterprise)
* Otomatik bildirim için, SQL Agent Job tamamlandığında e-posta ile bilgilendirme ayarlanmalı.

Yedeklerin gerçekten çalıştığından emin olmak önemli ve bunun için de yedekleme testleri mutlaka periyodik olarak yapılmalıdır.

**KAYNAKLAR**

1. <https://www.youtube.com/watch?v=Z-6GkXCTPic&list=TLPQMjkwNTIwMjWm44UozrR5Og&index=4>
2. Kullanılan Veritabanı: <https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms>
3. Anlatım Videosu: https://youtu.be/YV0IEwAlN88?si=U87O8LFiE6ZBXOjd
4. GitHub Linki: <https://github.com/Impasbaa/MSSQL/blob/main/SQLQuery1.sql>

**PROJE 2. VERİTABANI YÜK DENGELEME VE DAĞITIK VERİTABANI YAPILARI**

**PROJE AMACI:** Bu projenin amacı, kurumsal düzeyde bir veritabanı sisteminin yüksek erişilebilirlik, yüksek performans ve kesintisiz hizmet sağlayacak şekilde nasıl yapılandırılabileceğini göstermektir. Bu kapsamda:

* Dağıtık veritabanı yapıları ile verilerin farklı sunucularda/coğrafi lokasyonlarda yönetilmesi,
* Yük dengeleme stratejileri sayesinde sistem kaynaklarının verimli kullanımı,
* Veritabanı replikasyonu ile veri tutarlılığının sağlanması,
* Always On Availability Groups ve Database Mirroring gibi teknolojilerle kesintisiz erişim ve otomatik failover süreçlerinin uygulanması,
* Gerçek dünya senaryoları ile bu yapıların nasıl entegre edileceğinin anlaşılması amaçlanmaktadır.
* **Yük Dengeleme**, yüksek performans, ölçeklenebilirlik ve kesintisiz hizmet vermek amacında gelen veri/iş yükünün birden fazla veritabanı veya sunucuya dengeli şekilde dağıtılmasıdır (bir e-ticaret sitesi örneği düşünelim: Sipariş alma işlemleri PRIMARY sunucudan yapılırken, ürün listeleme, yorum gösterme gibi okuma işlemleri SECONDARY sunucuya yönlendirilir. Bu yönlendirme, uygulama kodunda veya SQL Server'ın “Read-Only Routing” özelliğiyle yapılır).
* **Dağıtık Veritabanında**, veriler veritabanını oluşturan birden fazla bilgisayara dağıtılır. Ve tüm veritabanı tek bir bilgisayarda çalışır (büyük firmalar (Amazon, Trendyol, Hepsiburada gibi), Türkiye ve Avrupa’da farklı veri merkezlerinde aynı veritabanının kopyalarını tutar. Kullanıcı, coğrafi olarak en yakın sunucuya bağlanır. Bu sayede sorgular daha hızlı döner (latency azalır)).

Dağıtılmış veritabanları daha fazla veri depolamamıza; işi bilgisayarlar arasında bölerek büyük miktardaki veriler üzerinde çok daha hızlı sorgular gerçekleştirmemize; donanım veya ağ arızaları durumunda hata toleransını artırarak sistemlerimizi daha dayanıklı hale getirmemize olanak tanır.

metin, ekran görüntüsü, tasarım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**1. GİRİŞ**

**1.1. BİRDEN ÇOK VERİTABANI YÖNETİMİ VE REPLİKASYON TEKNİKLERİ**

* **Veritabanı Replikasyonu (SQL Server Replication)**, bir veritabanındaki verilerin birden fazla sunucuya (subscriber) otomatik olarak çoğaltılması ve senkronize edilmesidir.

Replikasyon türleri:

- Snapshot Replication (belirli aralıklarla verinin tam kopyasını alır)

- Transactional Replication (anlık veri değişimlerini hemen gönderir (gerçek zamanlıya yakın))

- Merge Replication (hem publisher hem subscriber veri değiştirirse birleştirme yapar yani çift yönlü veri senkronizasyonu sağlar)

**1.1.1. Kullanım Senaryosu (Transactional Replication)**

Örnek: İstanbul’daki ana sunucudaki verilerin anlık olarak Ankara’daki bir şubeye aktarılması.

metin, ekran görüntüsü, yazı tipi, web sayfası içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**1.2. YÜK DENGELEME YÖNTEMLERİ: ALWAYS ON AVAİLABİLİTY GROUPS VE DATABASE MİRRORİNG**

**1.2.1. SQL Server Always On Availability Groups**

Çoklu veritabanlarının yüksek erişilebilirlik ve yük dengeleme için yapılandırılmasıdır. Birincil veritabanından ikincil veritabanlarına veri çoğaltma yapar.

Senaryo: Ana sunucu (ServerA) ile yedek sunucu (ServerB) arasında veri senkronizasyonu ve okuma yük dengelemesi sağlanacak.

Adımlar:

* WSFC kümesi oluşturulur.
* SQL Server’da Always On özelliğini aktif edilir.
* Availability Group oluşturulur (SSMS veya T-SQL).
* Replika sunucular belirlenir (Primary & Secondary).
* Listener (bağlantı noktası) ile uygulamalar bağlanır.

*Örnek kod (Availability Group oluşturma – avantajı, uygulamalar okuma işlemlerini otomatik olarak yedek sunuculardan yapabilir. Yük dengelenmiş olur):*

CREATE AVAILABILITY GROUP [OrnekAG]

WITH (

AUTOMATED\_BACKUP\_PREFERENCE = SECONDARY

)

FOR DATABASE [OrnekDB]

REPLICA ON

N'SERVERA' WITH (

ENDPOINT\_URL = 'TCP://SERVERA:5022',

AVAILABILITY\_MODE = SYNCHRONOUS\_COMMIT,

FAILOVER\_MODE = AUTOMATIC,

SECONDARY\_ROLE (ALLOW\_CONNECTIONS = READ\_ONLY)

),

N'SERVERB' WITH (

ENDPOINT\_URL = 'TCP://SERVERB:5022',

AVAILABILITY\_MODE = SYNCHRONOUS\_COMMIT,

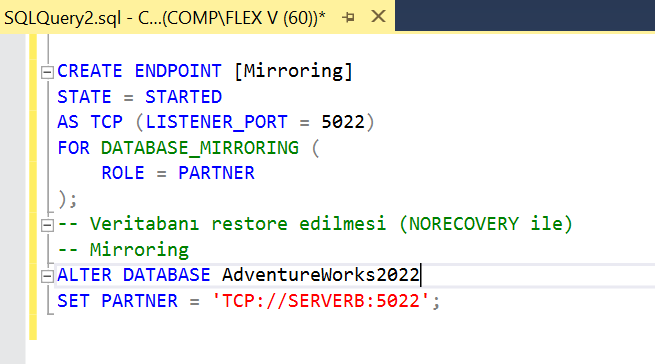
FAILOVER\_MODE = AUTOMATIC,

SECONDARY\_ROLE (ALLOW\_CONNECTIONS = READ\_ONLY)

);

**1.2.2. Database Mirroring (daha eski bir teknoloji)**

Bir veritabanının birincil ve ikincil sunucuya yansıtılması (1 veritabanı için geçerlidir).



**Modlar:**

* High safety (senkron + otomatik geçiş)
* High performance (asenkron)
* High safety without automatic failover

**Gerçek Hayat Senaryosu Örneği:**

📦 **Firma:** TrendyX  
🌐 **Lokasyon:** İstanbul (Ana sunucu), Frankfurt (Yedek sunucu)  
📈 **Uygulama:**

* İstanbul’daki PRIMARY sunucu siparişleri işler (INSERT, UPDATE).
* Almanya’daki SECONDARY sunucu sadece SELECT işlemleri için kullanılır.
* Availability Group ile bağlantılar trendyx-db-listener üzerinden yapılır.
* PRIMARY sunucu çökerse failover otomatik olarak Almanya sunucusuna gerçekleşir.
* Kullanıcılar bu geçişi fark etmez, web sitesi çalışmaya devam eder.

**1.3. FAİLOVER SENARYOLARI**

**1.3.1. Always On Failover Stratejileri**

- Otomatik Failover: Önceden tanımlanmış bir ikincil sunucuya otomatik geçiş sağlar. Genellikle 2-3 düğümlü senkron commit kullanır.

- Planlı Manuel Failover: Bakım veya yükseltme durumlarındadır. Veri kaybı olmaz.,

- Zorunlu Manuel Failover (Force Failover): Birincil sunucu çöktüğünde ve otomatik failover başarısız olduğunda kullanıma geçer. Fakat potansiyel veri kaybı riski vardır.

**1.3.2. Performans Optimizasyonu**

- Okuma İş Yükü Dengeleme: İkincil sunuculara okuma işlemlerini yönlendirme ve Connection string'de ApplicationIntent=ReadOnly kullanımı yapılabilir.

- Yönlendirme Stratejileri: Round-robin DNS; Donanım veya yazılım yük dengeleyicileri; Uygulama katmanında yönlendirme gibi işlemler gerçekleştirilebilir.

**1.3.3. İzleme ve Bakım**

- Sistem DMV'leri: sys.dm\_hadr\_availability\_group\_states; sys.dm\_hadr\_availability\_replica\_states; sys.dm\_hadr\_database\_replica\_states.

- Performans Metrikleri: Veri senkronizasyon gecikmesi; Log send queue size; Redo queue size

Bu teknikler, SQL Server ortamlarında yüksek erişilebilirlik, felaket kurtarma ve performans optimizasyonu sağlamak için kritik öneme sahiptir.

**SONUÇ**

Bu proje kapsamında, yüksek erişilebilirlik, veri tutarlılığı ve sistem performansını maksimize etmeye yönelik olarak **veritabanı yük dengeleme ve dağıtık veritabanı yapıları** incelenmiş ve uygulamaya yönelik senaryolarla desteklenmiştir.

Projede özellikle aşağıdaki başlıklar desteklenmiştir:

* **Always On Availability Groups** kullanılarak, birden fazla sunucu arasında hem veri senkronizasyonu hem de otomatik failover sağlanması.
* **Read-Only Routing** özelliği ile okuma işlemleri (ürün listeleme, yorum gösterme gibi) otomatik olarak **SECONDARY replica**'ya yönlendirilmiş, böylece PRIMARY sunucunun yükünün hafifletilmesi.
* **Replication teknikleri** ile veri çoğaltma ve coğrafi yedekliliğin sağlanması.
* **Failover senaryolarının** test edilmesi, sunucu kesintilerinde sistemin nasıl otomatik olarak yedek sunucuya geçtiğinin gözlemlenmesi.

**KAYNAKLAR**

1. <https://www.youtube.com/watch?v=J-sj3GUrq9k>
2. GitHub Linki: <https://github.com/Impasbaa/MSSQL/blob/main/SQLQuery3.sql>

**PROJE 3. VERİ TEMİZLEME VE ETL SÜREÇLERİ TASARIMI**

**PROJE AMACI:** Bu proje, kurum içindeki dağınık, tutarsız ve kalitesiz verileri standartlaştırarak güvenilir, analize hazır ve iş kararlarını destekleyecek bir veri altyapısı oluşturmayı hedeflemektedir. Temel odak noktaları şunlardır:

* Veri Kalitesinin Artırılması: Eksik, hatalı ve tekrarlanan verilerin tespiti ve düzeltilmesi; Farklı kaynaklardan gelen verilerin standart formata dönüştürülmesi
* ETL (Extract, Transform, Load) Süreçlerinin Otomasyonu: Veri entegrasyonunun manuel süreçlerden kurtarılarak zaman ve maliyet tasarrufu sağlanması; SQL tabanlı veri dönüşümleriyle tutarlı ve ölçeklenebilir bir veri akışı oluşturulması
* Raporlama ve Analiz için Optimize Edilmiş Veri Yapısı: İş zekası (BI) ve makine öğrenimi (ML) modellerine temiz ve yapılandırılmış veri sağlanması; Veri ambarlarında (data warehouse) tarihsel veri takibi ile trend analizlerinin kolaylaştırılması
* Uyumluluk ve Denetim İzlenebilirliği: KVKK, GDPR gibi düzenlemelere uygun veri doğrulama ve loglama mekanizmalarının kurulması: Veri kalite raporlarıyla şeffaf ve denetlenebilir bir süreç yönetimi
* **Veri Temizleme,** hatalı, eksik, yinelenen veya tutarsız verilerin tespit edilmesi ve temizlenmesidir.

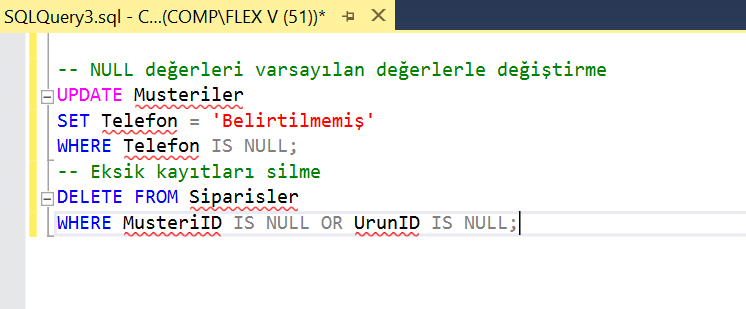
**1. GİRİŞ**

**1.1. Veri Temizleme**

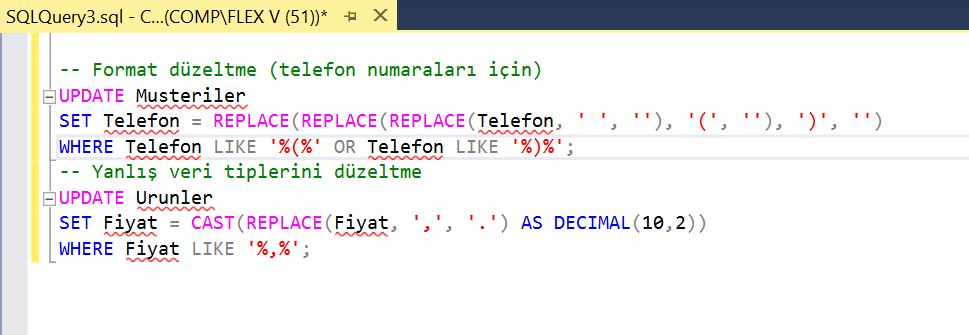
Çözmeye çalıştığımız sorunla ilgili eksik, yanlış veya alakasız yani kirli verilerin eksiksiz, doğru ve alakalı veriler haline getirmektir yani temizlemektir.

**1.1.1. SQL ile Veri Temizleme Teknikleri**

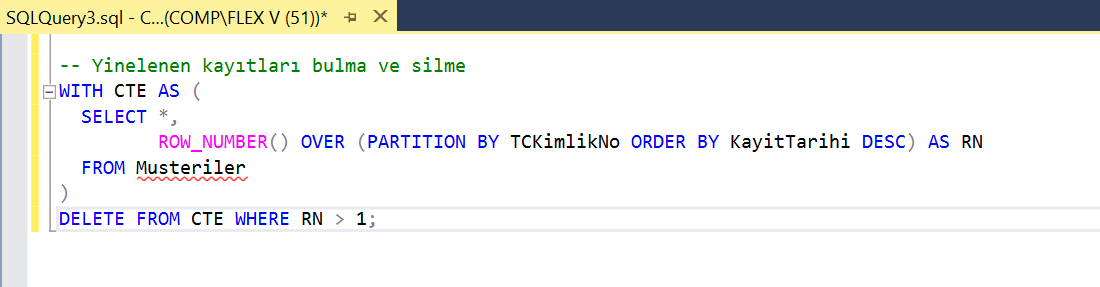
*Eksik verilerin yönetimi:*

****

*Tutarsız verilerin düzeltilmesi:*

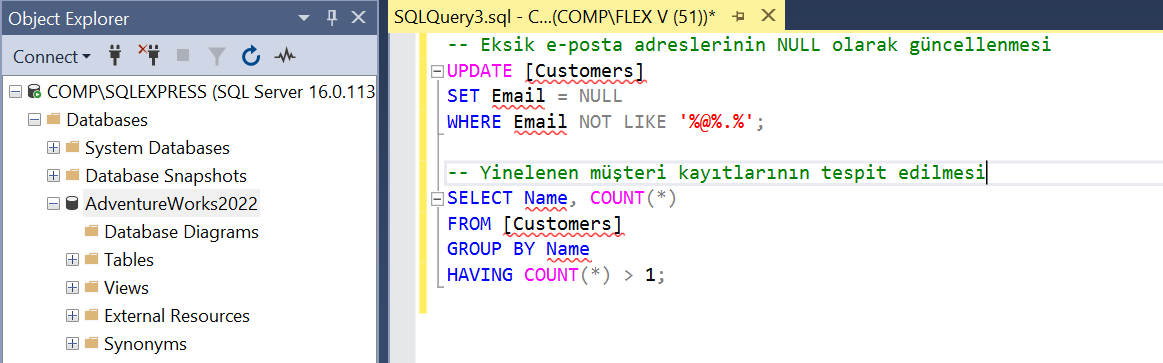
****

*Tekilleştirme (Deduplication):*

****

**Örnek:**

*Eksik veya geçersiz e-posta adreslerinin temizlenmesi (SQL):*

****

**1.2. ETL (Extract, Transform, Load) Süreçleri**

ETL süreçleri, büyük veri kümelerinin işlenmesi, temizlenmesi ve hedef sistemlere yüklenmesi için kritik bir altyapı sağlar.

*ETL Araçları:*

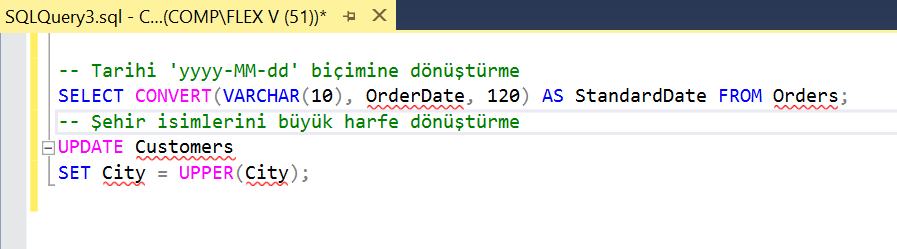
metin, yazı tipi, ekran görüntüsü, logo içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**1.2.1. Veri Dönüştürme**

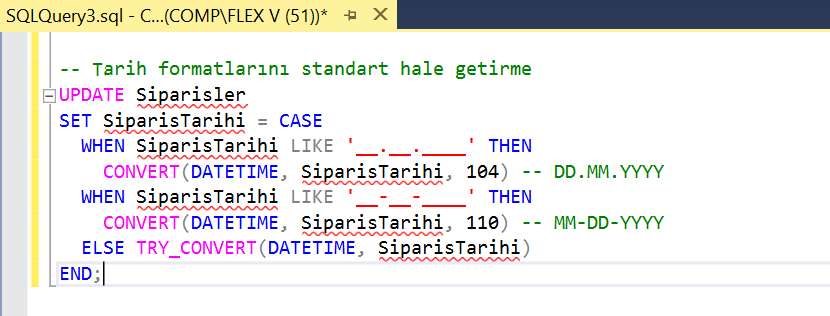
Farklı formatlardaki verileri analiz için tutarlı hale getirmek.

**Örnek:** Tarih formatlarını standardize etmek ve şehir adlarını büyük harfe çevirmek:

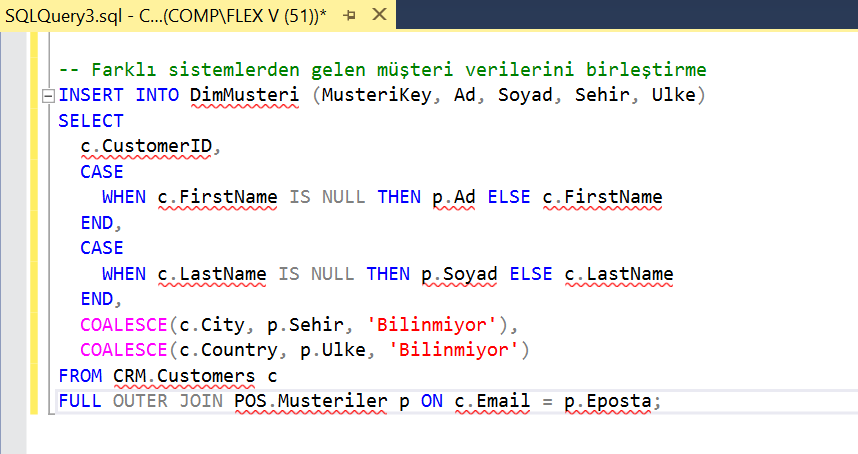


**Standartlaştırma Teknikleri:**

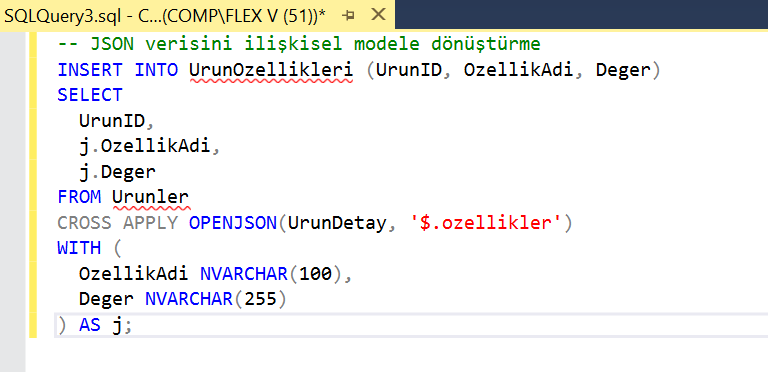
*Veri formatlarını birleştirme:*

****

*Farklı kaynakları entegre etme:*

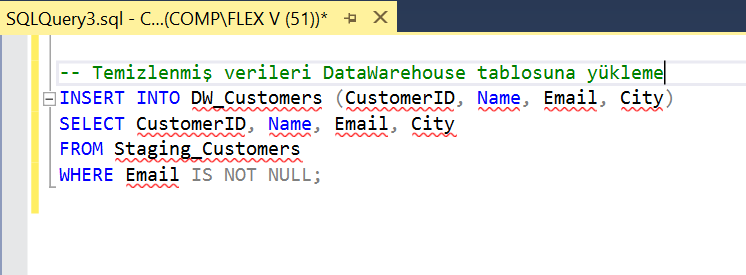


*Hiyerarşik verileri düzleştirme:*



**1.2.2. Veri Yükleme**

Temizlenmiş ve dönüştürülmüş verilerin hedef veritabanına (data warehouse gibi) aktarılması.



**Yükleme Stratejileri:**

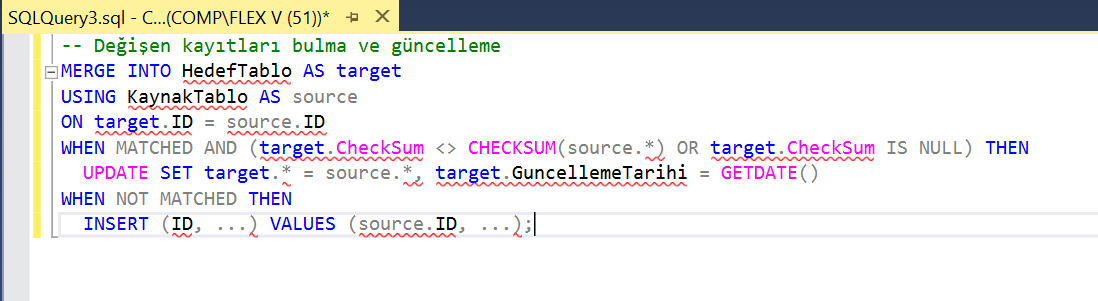
*Tam yükleme:*

TRUNCATE TABLE HedefTablo;

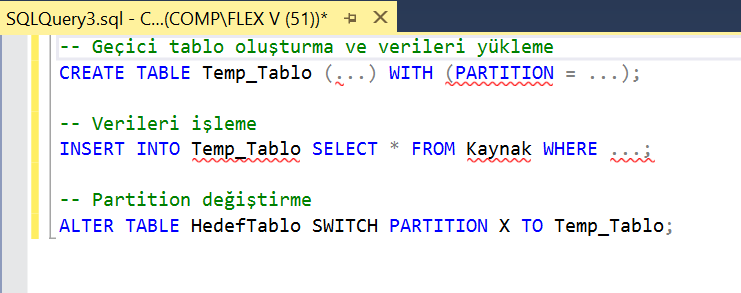
INSERT INTO HedefTablo

SELECT \* FROM KaynakTablo;

*Artımlı yükleme:*



*Partition Switching (büyük veri yüklemeleri için):*

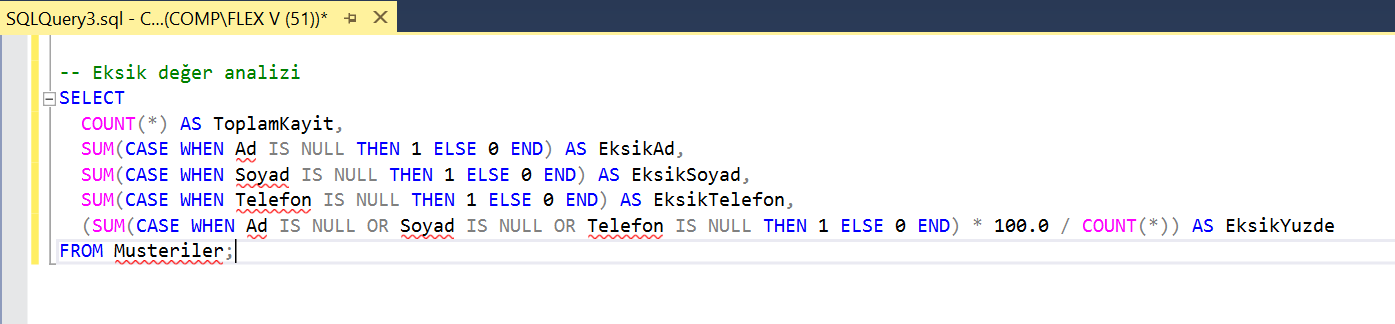


**1.2.3. Veri Kalitesi Raporları**

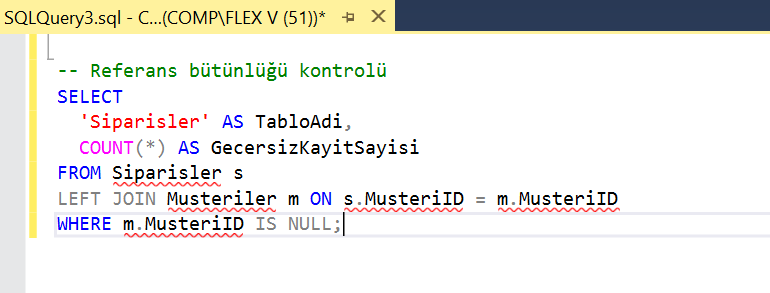
Temizlenen verilerin ne kadar düzeltildiği, hangi tür hataların giderildiği gibi analizler sunmak.

**Veri Kalitesi Metrikleri ve Raporlanması**

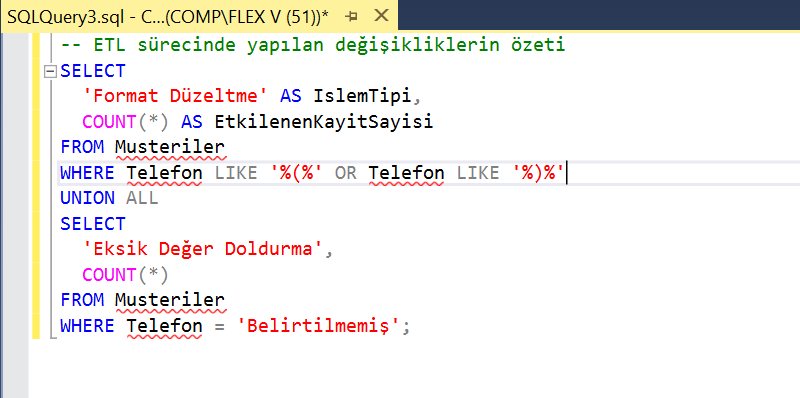
*Temel veri kalitesi ölçümleri:*



*Veri tutarlılık raporları:*

**

*Veri temizleme özet raporu:*



**SONUÇ**

Bu proje kapsamında, büyük ve çeşitli veri kaynaklarının analiz edilebilir hale getirilmesi amacıyla ETL (Extract, Transform, Load) süreçleri detaylı olarak tasarlanmış ve uygulanmıştır. Özellikle verilerin kalitesini artırmak, anlamlı ve tutarlı hale getirmek için veri temizleme, dönüştürme ve yükleme işlemleri sistematik biçimde gerçekleştirilmiştir.

Projede yapılan temel çalışmalar şunlardır:

* **Veri Temizleme**: Hatalı formatta, eksik veya yinelenen verileri tespit etme; SQL sorguları kullanarak bu verileri standartlaştırılmış ve tutarsızlıklar giderilmiş hale dönüştürmek.
* **Veri Dönüştürme**: Farklı kaynaklardan gelen veriler tek bir yapıda ve analiz edilebilir formatta dönüştürülerek işlenebilir hale getirilmiştir. Örneğin, tarih formatları ve adres bilgilerindeki farklar düzeltilmiştir.
* **Veri Yükleme**: Temizlenmiş verilerin, hedef sistemlere (örneğin veri ambarına) başarıyla yüklenmesi ve raporlama sistemlerinin kullanımına sunulması.
* **Veri Kalitesi Raporları**: Temizleme sürecinin başarısının ölçümü, eksik ya da hatalı veri oranlarının raporlanarak sürecin doğruluğu ve etkinliğinin analiz edilmesi.

Bu süreçler sayesinde, sistemdeki veri tutarlılığı ve güvenilirliği önemli ölçüde artırılır. Ayrıca, veri analizi ve raporlama süreçleri daha sağlıklı ve hızlı bir şekilde gerçekleştirilebilir.

Bu projenin çıktıları, özellikle büyük veri setleriyle çalışan kurumlar için karar destek sistemlerinin temelini oluşturacak niteliktedir. ETL süreçlerinin doğru planlanması ve uygulanması sayesinde, veriden maksimum fayda sağlanmasının mümkün olduğu söylenebilir.

**KAYNAKLAR**

1. <https://www.youtube.com/watch?v=rnmoT0P6Oyg>
2. Anlatım Videosu: <https://www.youtube.com/watch?v=7bq16kJ01FY>
3. GitHub Linki: <https://github.com/Impasbaa/MSSQL/blob/main/SQLQuery5.sql>

**PROJE 4. VERİTABANI YEDEKLEME VE OTOMASYON ÇALIŞMASI**

**PROJE AMACI:** Bu projenin amacı, SQL Server ortamında veritabanı yedekleme işlemlerini otomatikleştirerek yönetim süreçlerini kolaylaştırmak, yedek bütünlüğünü düzenli olarak denetlemek ve olası yedekleme hatalarına karşı sistem yöneticilerini zamanında bilgilendirebilecek otomasyon sistemleri kurmaktır.

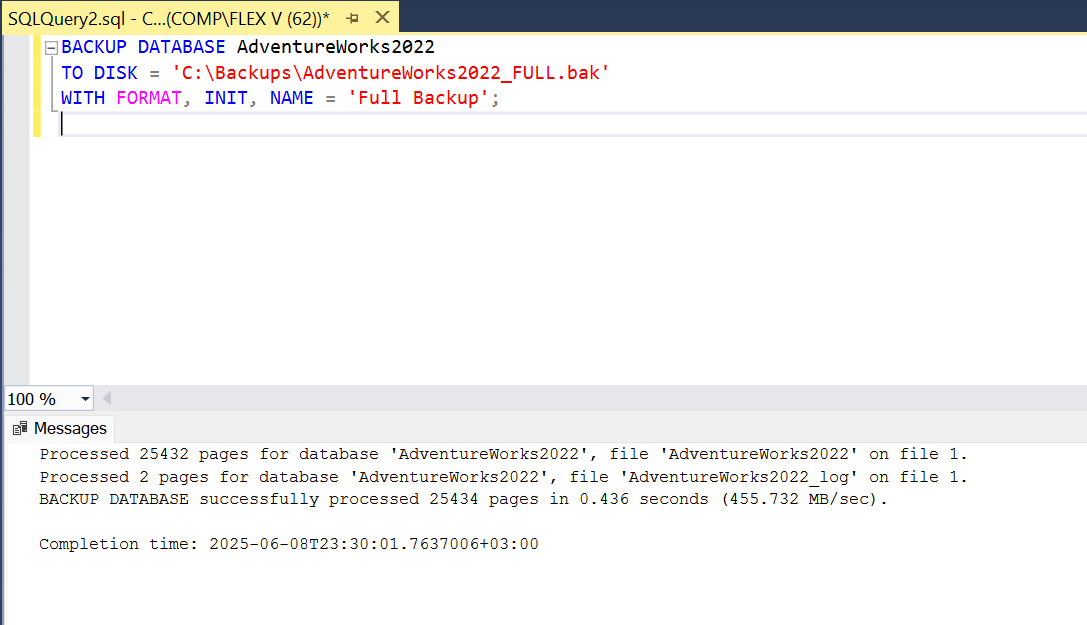
**GEREKEN ARAÇLAR:**

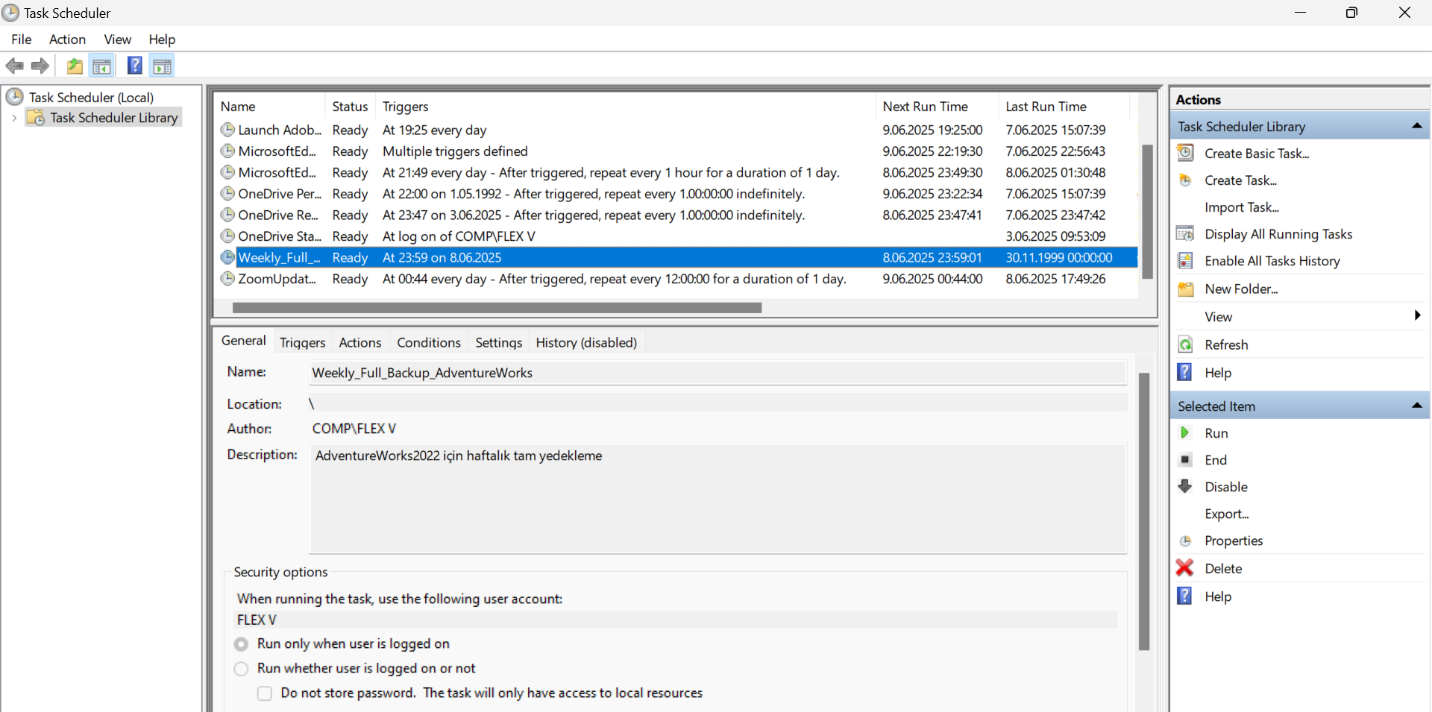
* SQL Server Management Studio (SSMS): Veritabanı yönetimi ve T-SQL scriptleri için.
* SQL Server Agent: Otomatikleştirme ve zamanlanmış yedekleme işleri için.
* PowerShell: Script bazlı otomasyon ve loglama işlemleri için.
* E-posta sunucusu (SMTP): Hatalı işlemlerde ya da Job başarısız olduğunda e-posta uyarıları göndermek için.

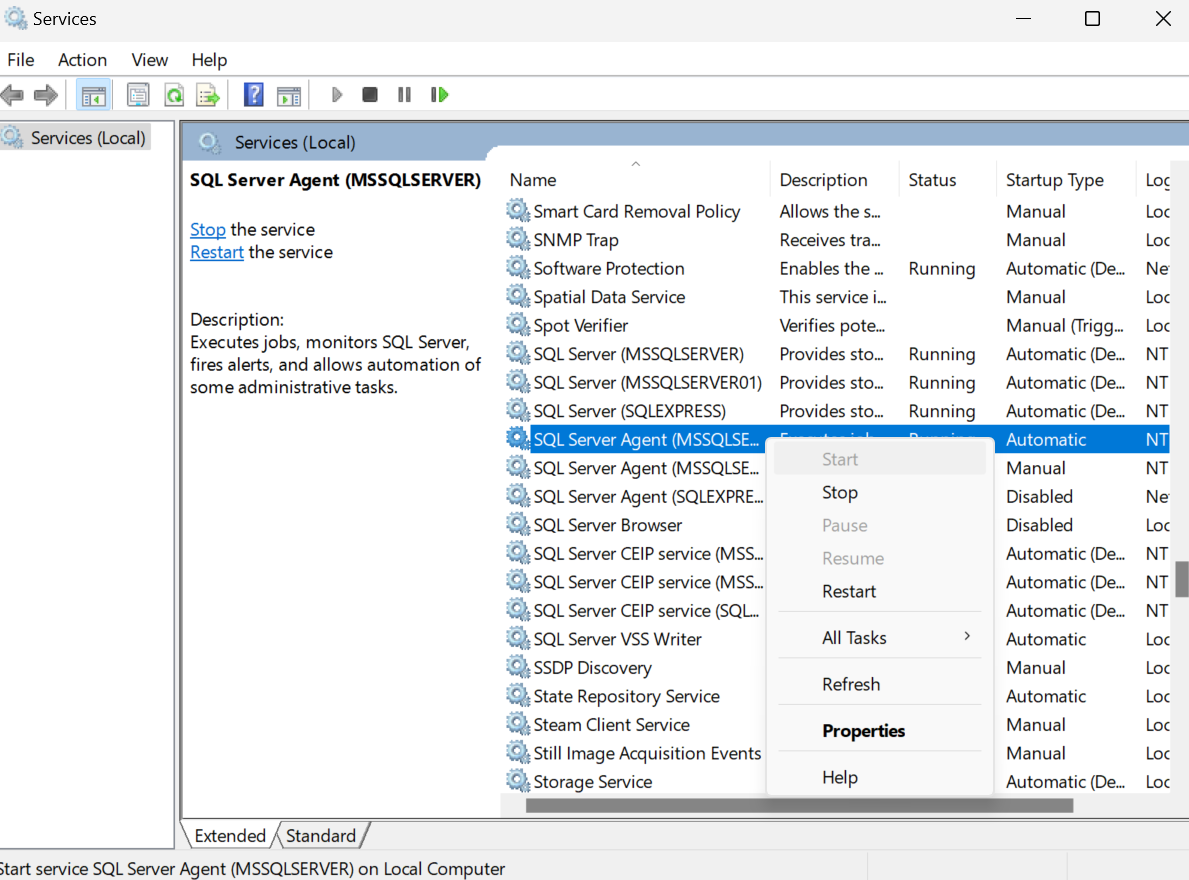
**1. GİRİŞ**

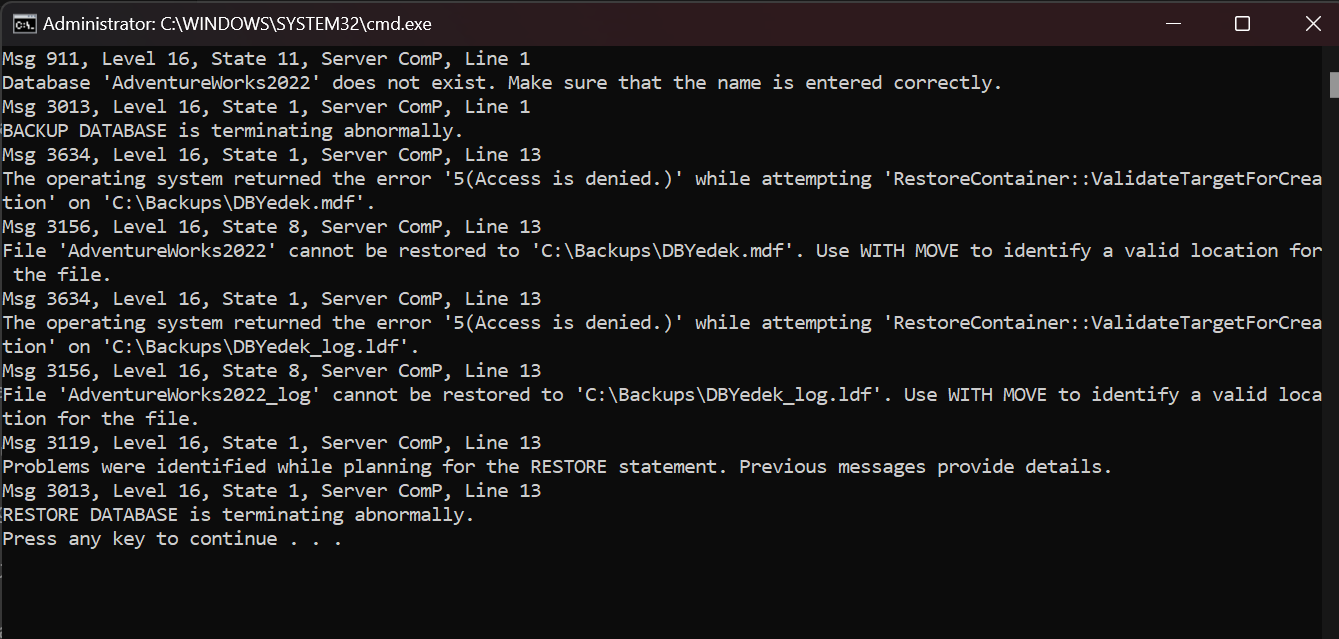
Veritabanı yedekleme, olası donanım arızaları, insan hataları veya siber saldırılar gibi durumlarda veri kaybını önlemek için kritik öneme sahiptir. Özellikle otomatikleştirilmiş yedekleme sistemleri, veritabanı yöneticilerinin manuel işlem yükünü azaltır ve hata olasılığını düşürür.

**1.1. SQL Server Agent ile Otomatik Yedekleme**

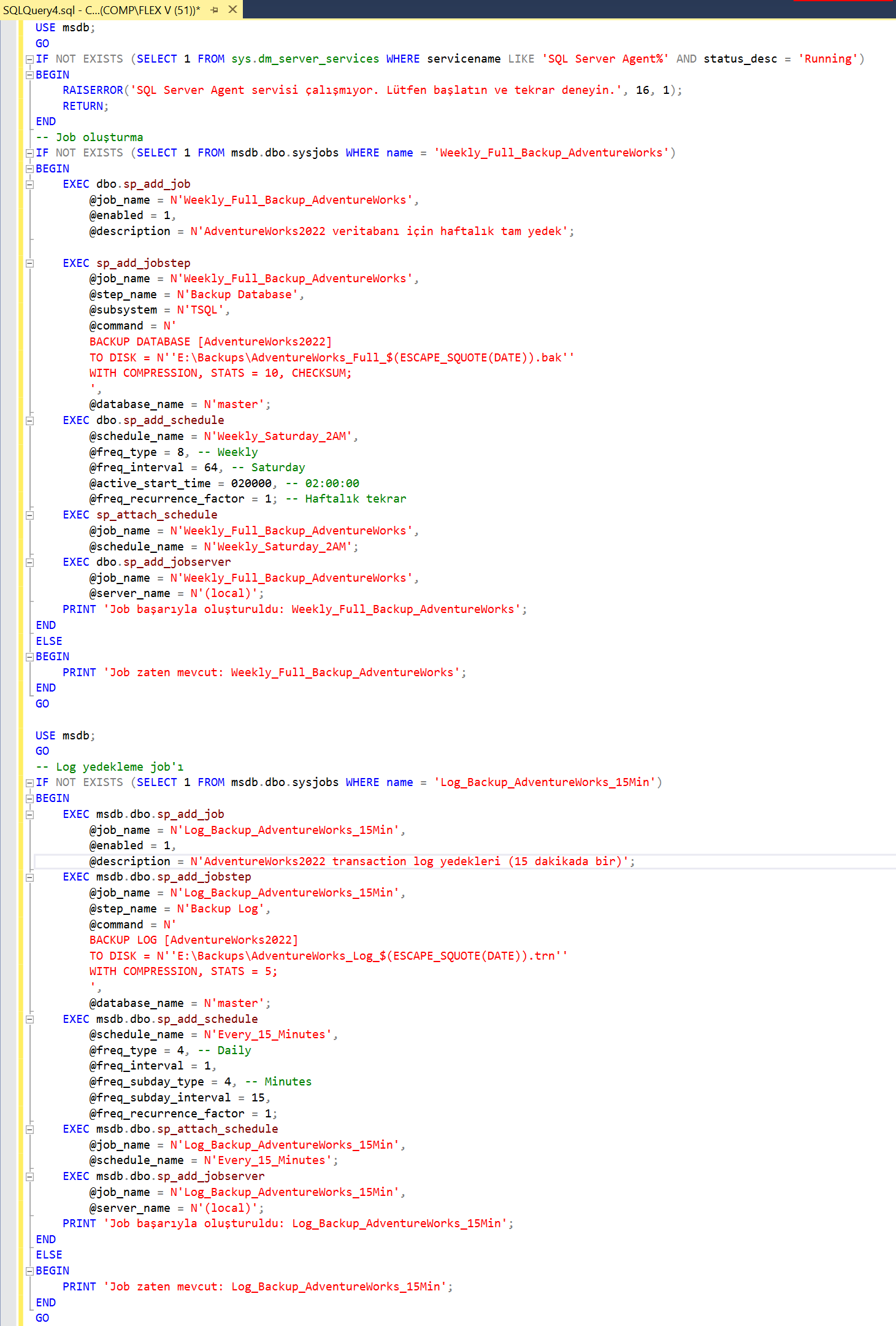


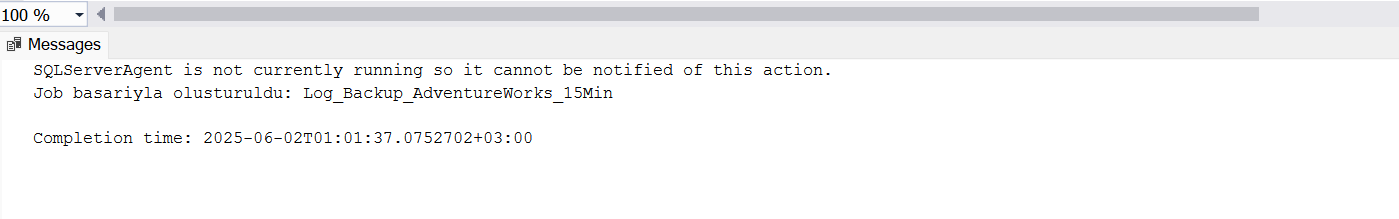






**1.1.1. Tam Yedek Zamanlanmış Görev ve Log Yedekleme – 15 Dakikada Bir**

****

****

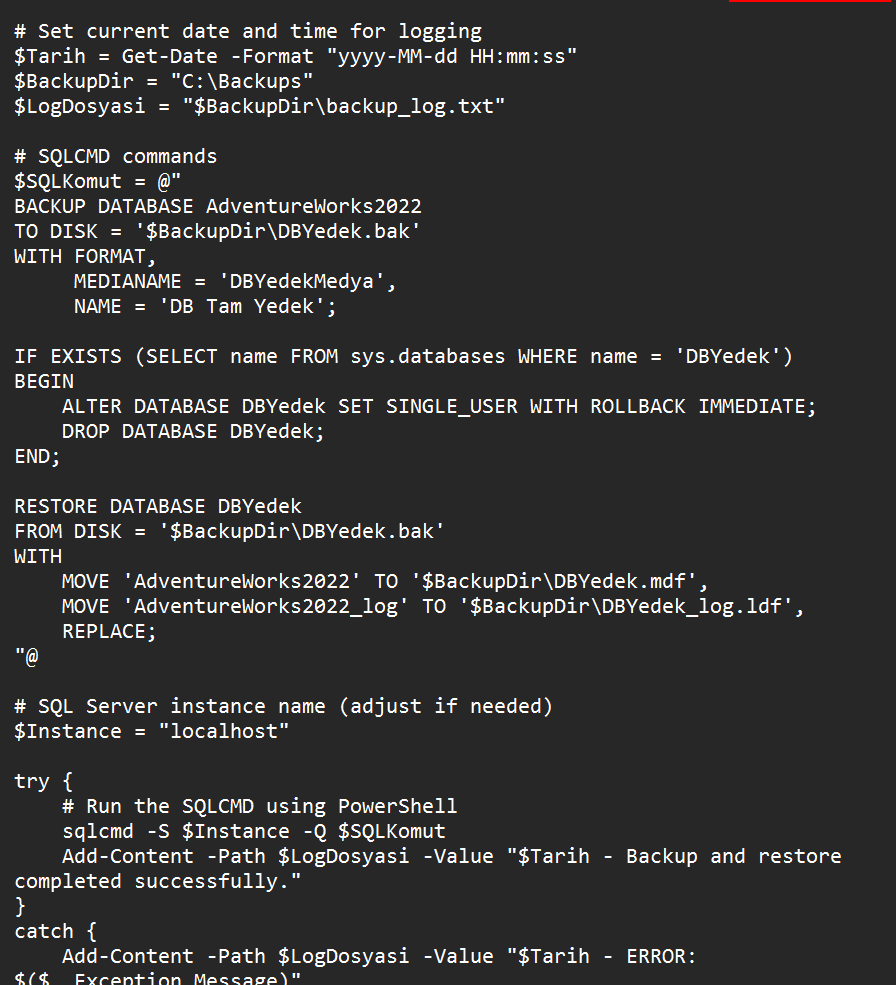
**1.2. PowerShell ile Yedekleme Otomasyonu**

* **BackupRestoreDB.ps1 PowerShell script dosyası**, otomatik işlemleri (backup alma, dosya taşıma, servis kontrolü vs.) yapmak için komutları topluca çalıştırmanı sağlamaktadır.

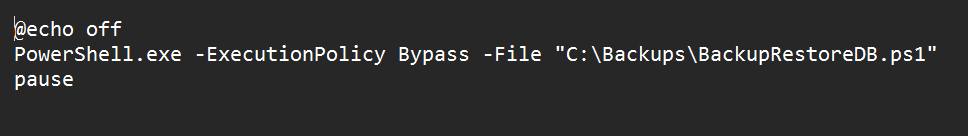
Bu projede .ps1 dosyası SQL Server'dan bir veritabanının (AdventureWorks2022) yedeklemesini almakta ve eğer eski yedek varsa onu silip yenisini yüklemektedir (restore). İşlem sonucunu da bir log dosyasına yazar.

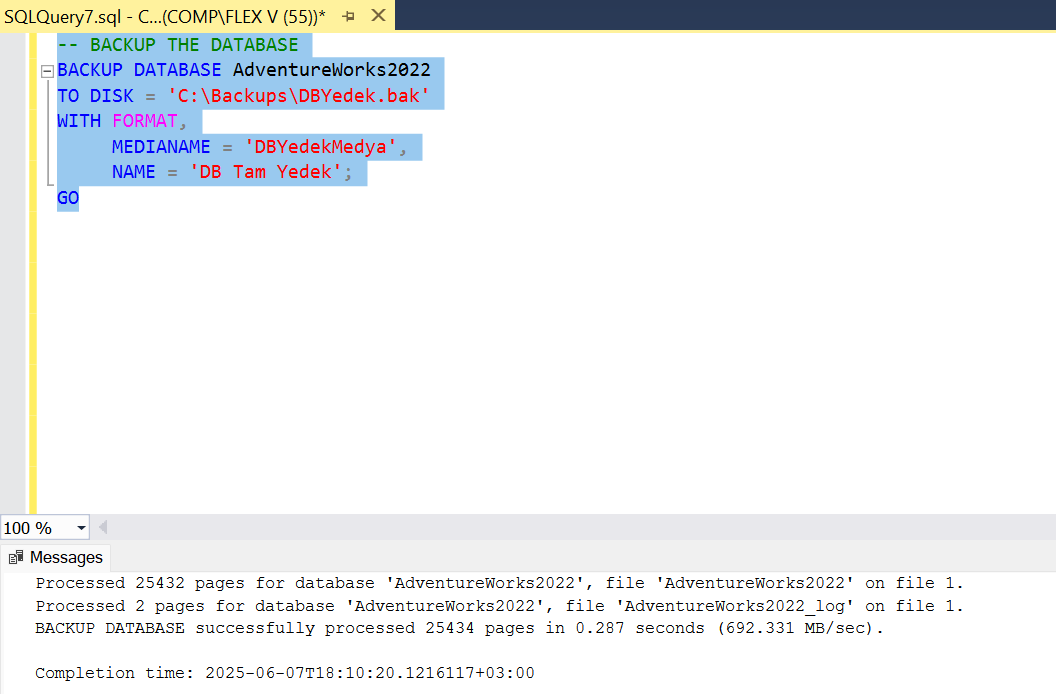
Avantajları şu şekildedir:

* + Tek seferde yedekleme ve geri yükleme işlemini otomatik yapar.
  + El ile komut yazmamıza gerek kalmaz.
  + Hataları log dosyasına yazar, takip etmesi kolaydır.

****

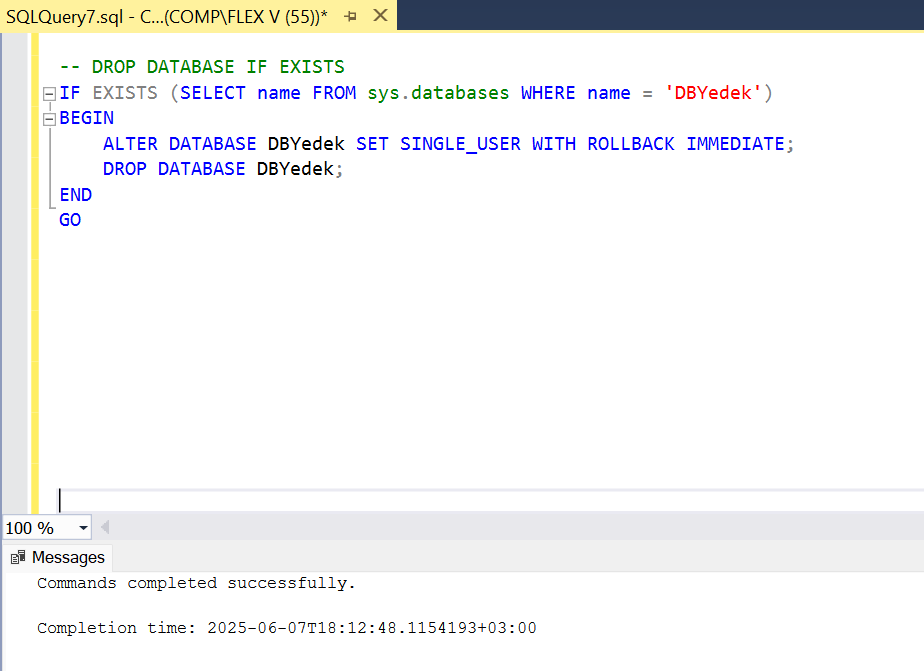
* **BackupRestoreDB.bat Batch dosyası** .ps1 gibi bir PowerShell dosyasını veya başka komutları çift tıklamayla çalıştırır. Bu projede .bat dosyası .ps1 dosyasını çalıştırır.

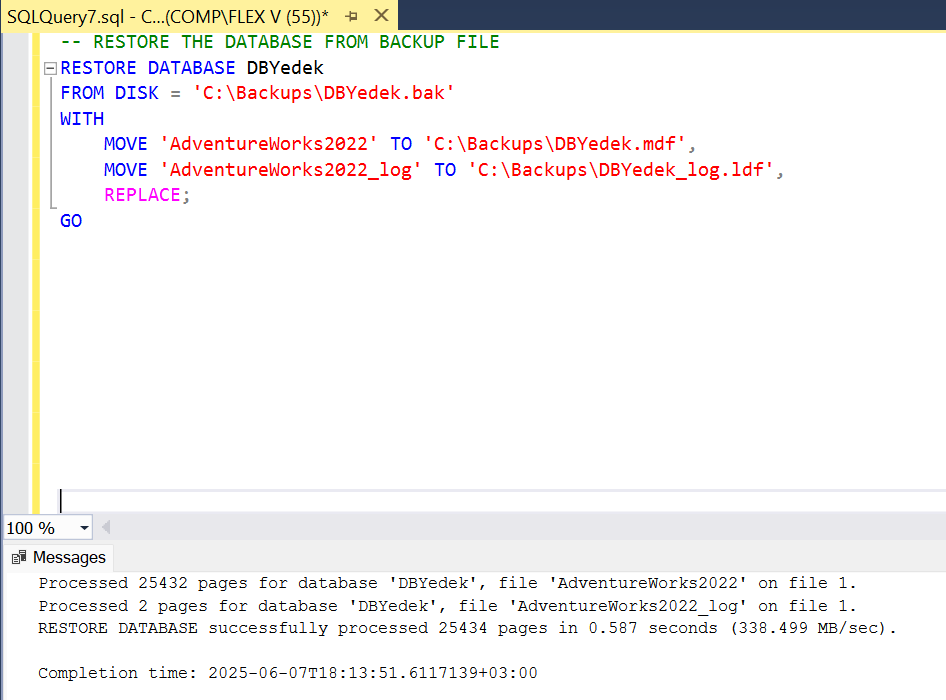


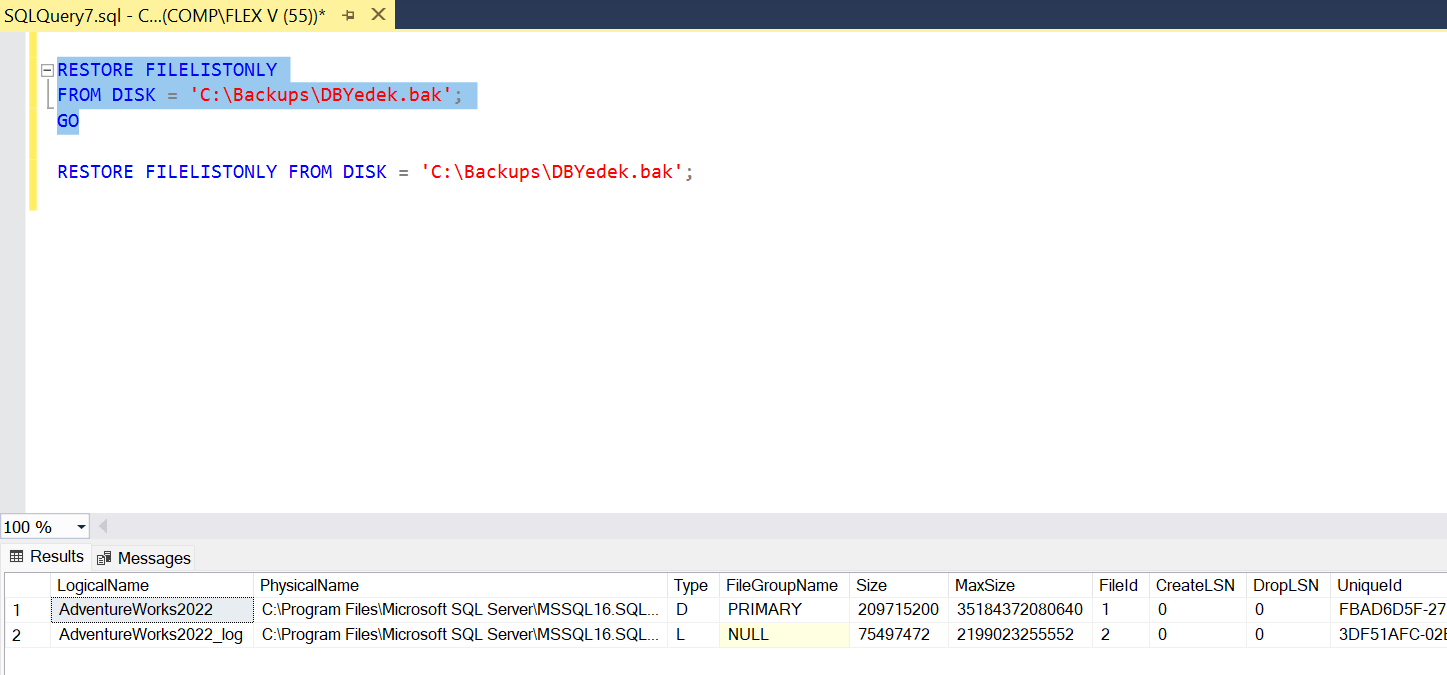


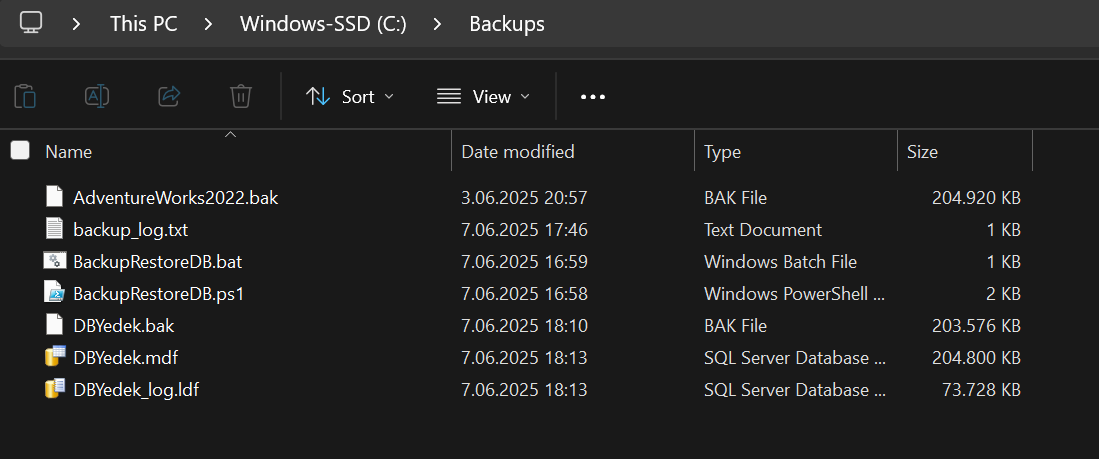
metin, yazılım, web sayfası, web sitesi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.









**1.3. Denetim ve Raporlama**

*Son yedekleri kontrol eden sorgu:*

*metin, ekran görüntüsü, yazılım, web sayfası içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.*

**SONUÇ**

Bu proje kapsamında, SQL Server ortamında yedekleme işlemleri otomatikleştirilmiş, hem tam hem log yedekleme stratejileri uygulanmış, ayrıca PowerShell scripti ile kolay otomasyon ve log takibi sağlanmıştır.

Projenin çıktıları:

* SQL Server Agent ile zamanlanmış tam ve log yedekleme işlerinin kurulumu.
* PowerShell scripti ile manuel komut gereksinimini ortadan kaldıran bir yedekleme otomasyonu.
* Backup başarı/başarısızlık durumlarının loglanması.
* Olası felaket durumlarında geri yükleme (restore) işleminin hızlı yapılabilmesi.

Bu yapı, kurumsal veritabanı yönetimi için sürdürülebilir, güvenli ve kolay yönetilebilir bir yedekleme altyapısı sunmaktadır.

**KAYNAKLAR**

1. GitHub Linki: <https://github.com/Impasbaa/MSSQL/blob/main/SQLQuery7.sql>
2. Microsoft Docs: <https://learn.microsoft.com/en-us/sql/relational-databases/backup-restore/restore-a-database-backup-using-ssms?view=sql-server-ver17>