



# Ağ Tabanlı Paralel Dağıtım Sistemleri Final Ve Vize Dokumanlari

Kaan Kaya

21290436

Mert efe kandemir – 21290233 ile ortak

Github

<https://github.com/kayakaan02/-BLM4522-A-Tabanl-Paralel-Dat-m-Sistemleri-Proje-Videolari>

Videolarin tamaminin youtube oynatma listesi:

[https://www.youtube.com/playlist?list=PL\\_xJ1vd3Lbi4ybFYfKiZtxcnSzTg4TgtK](https://www.youtube.com/playlist?list=PL_xJ1vd3Lbi4ybFYfKiZtxcnSzTg4TgtK)

Proje1 youtube: [https://youtu.be/qp-wpPT\\_Y1I](https://youtu.be/qp-wpPT_Y1I)

Proje2 youtube: <https://youtu.be/njts0lYnghg>

Proje3 youtube: <https://youtu.be/h8au0whffbg>

Proje4 youtube: <https://youtu.be/ajXdT7iqPvY>

Proje5 youtube: <https://youtu.be/XqY41jQTtc>

Proje6 youtube: <https://youtu.be/9abJXGKdTMl>

Proje7 youtube: <https://youtu.be/AEgM5Sl-Uyl>

# Proje 1: Veritabanı Performans Optimizasyonu ve İzleme

Hazırlayan: Mert Efe Kandemir, Kaan Kaya

Numara: 21290233, 21290436

Teslim: 25.04.2025

Github: <https://github.com/kayakaan02/-BLM4522-A-Tabanlı-Paralel-Da-t-m-Sistemleri-Proje-Videolari/blob/main/proje1.mp4>

## 1. Giriş

Bu proje, SQL Server ortamında büyük ölçekli veritabanlarında performans takibi ve iyileştirme stratejilerinin uygulanmasını kapsamaktadır. Yavaş çalışan sorguların optimize edilmesi, indeks yönetimi, sistem kaynaklarının izlenmesi ve erişim yönetimi gibi önemli performans faktörleri ele alınmıştır.

## 2. Veritabanı İzleme

SQL Server Profiler ve Dynamic Management Views (DMV) gibi araçlar kullanılarak sistemde çalışan sorgular, kilitlenmeler ve yavaşlamalar tespit edilmiştir. Özellikle uzun süren sorguların CPU ve IO kullanımı detaylı olarak analiz edilmiştir. Sistem genelinde kaynak tüketimini etkileyen işlemler belirlenmiş ve işlem öncelikleri yeniden değerlendirilmiştir.

Aşağıdaki sorgu ile sistemdeki en çok kaynak tüketen sorgular listelenerek analiz edilmiştir:

```

SELECT TOP 10
    qs.total_elapsed_time / qs.execution_count AS OrtalamaSüre,
    qs.execution_count,
    SUBSTRING(qt.text, qs.statement_start_offset / 2,
        (CASE WHEN qs.statement_end_offset = -1
            THEN LEN(CONVERT(NVARCHAR(MAX), qt.text)) * 2
            ELSE qs.statement_end_offset
        END - qs.statement_start_offset) / 2) AS Sorgu
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) qt
ORDER BY OrtalamaSüre DESC;

```

|   | OrtalamaSüre | execution_count | Sorgu  |
|---|--------------|-----------------|--|
| 1 | 2001116      | 1               | SELECT dtb.name AS [Name], CAST(0 AS bit) AS [Is...    |
| 2 | 1063         | 1               | SELECT log.name AS [Name], log.principal_id AS [ID]... |
| 3 | 245          | 1               | )SELECT dtb.collation_name AS [Collation], dtb.name... |
| 4 | 162          | 1               | )SELECT dtb.containment AS [ContainmentType], dtb...   |
| 5 | 120          | 1               | SELECT tr.name AS [Name], tr.object_id AS [ID], CA...  |

### 3. İndeks Yönetimi

Veri erişim performansını artırmak için tablolar üzerinde yer alan indeksler analiz edilmiştir. Sık kullanılan sorgulara uygun biçimde yeni indeksler oluşturulmuş, kullanılmayan ve sisteme yük olan indeksler kaldırılmıştır. Clustered ve Non-Clustered indeks farkları dikkate alınarak doğru stratejiler belirlenmiştir.

#### Örnek Veritabanı Oluşturma

#### İndeks Kullanım İstatistiklerini Analiz Etme

Aşağıdaki sorgu ile indeks kullanım istatistikleri listelenerek, az kullanılan indeksler tespit edilmiştir:

```

SELECT
    OBJECT_NAME(i.object_id) AS TabloAdi,
    i.name AS IndexAdi,
    i.index_id,
    i.type_desc AS IndexTuru,
    ISNULL(ius.user_seeks, 0) AS UserSeeks,
    ISNULL(ius.user_scans, 0) AS UserScans,
    ISNULL(ius.user_lookups, 0) AS UserLookups,
    ISNULL(ius.user_seeks, 0) + ISNULL(ius.user_scans, 0) + ISNULL(ius.user_lookups, 0) AS ToplamKullanim,
    ISNULL(ius.user_updates, 0) AS GuncellemeSayisi
FROM sys.indexes i
LEFT JOIN sys.dm_db_index_usage_stats ius
    ON i.object_id = ius.object_id
    AND i.index_id = ius.index_id
    AND ius.database_id = DB_ID()
WHERE OBJECTPROPERTY(i.object_id, 'IsUserTable') = 1
ORDER BY TabloAdi, i.index_id;

```

|   | TabloAdi         | IndexAdi                        | index_id | IndexTuru | UserSeeks | UserScans | UserLookups | ToplamKullanim | GuncellemeSayisi |
|---|------------------|---------------------------------|----------|-----------|-----------|-----------|-------------|----------------|------------------|
| 1 | Musteriler       | PK_Musteriler__72624471D32FD5DB | 1        | CLUSTERED | 1         | 0         | 0           | 1              | 1                |
| 2 | SiparisDetaylari | PK_SiparisD__8E8164A594378883   | 1        | CLUSTERED | 0         | 0         | 0           | 0              | 1                |
| 3 | Siparisler       | PK_Siparisler__C3F03BDD1EA86B17 | 1        | CLUSTERED | 1         | 0         | 0           | 1              | 1                |
| 4 | Urunler          | PK_Urunler__623D364BBA1E0857    | 1        | CLUSTERED | 1         | 0         | 0           | 1              | 1                |

## İndeks Oluşturma ve Test Etme

İndeksleri test etmek için bazı sorgular çalıştırılmalı ve ardından yeni indeksler oluşturulmalı:

```

-- İndeksleri test etmek için bazı sorgular çalıştırılmalı
SELECT * FROM Musteriler WHERE Sehir = 'İstanbul';
SELECT * FROM Urunler WHERE Kategori = 'Elektronik';
SELECT * FROM Siparisler WHERE MusteriID = 1;
GO

-- Yeni indeks oluşturma
CREATE NONCLUSTERED INDEX IX_Musteriler_Sehir ON Musteriler(Sehir);
CREATE NONCLUSTERED INDEX IX_Urunler_Kategori ON Urunler(Kategori);
GO

-- Sorguları tekrar çalıştırılmalı
SELECT * FROM Musteriler WHERE Sehir = 'İstanbul';
SELECT * FROM Urunler WHERE Kategori = 'Elektronik';
GO

```

Results

Messages

|   | MusteriID | Ad     | Soyad  | Sehir    | Email           | Telefon    | Kayit Tarihi            |
|---|-----------|--------|--------|----------|-----------------|------------|-------------------------|
| 1 | 1         | Ali    | Yilmaz | Istanbul | ali@omek.com    | 5551112233 | 2025-04-28 03:35:37.883 |
| 2 | 4         | Fatma  | Sahin  | Istanbul | fatma@omek.com  | 5554445566 | 2025-04-28 03:35:37.883 |
| 3 | 6         | Zeynep | Çelik  | Istanbul | zeynep@omek.com | 5556667788 | 2025-04-28 03:35:37.883 |
| 4 | 8         | Ali    | Yilmaz | Istanbul | ali@omek.com    | 5551112233 | 2025-04-28 03:35:47.857 |
| 5 | 11        | Fatma  | Sahin  | Istanbul | fatma@omek.com  | 5554445566 | 2025-04-28 03:35:47.857 |
| 6 | 13        | Zeynep | Çelik  | Istanbul | zeynep@omek.com | 5556667788 | 2025-04-28 03:35:47.857 |
| 7 | 15        | Ali    | Yilmaz | Istanbul | ali@omek.com    | 5551112233 | 2025-04-28 03:36:01.913 |
| 8 | 18        | Fatma  | Sahin  | Istanbul | fatma@omek.com  | 5554445566 | 2025-04-28 03:36:01.913 |

|   | UrunID | UrunAdi      | Kategori   | BirimFiyat | StokMiktari |
|---|--------|--------------|------------|------------|-------------|
| 1 | 1      | Laptop       | Elektronik | 5999.99    | 25          |
| 2 | 2      | Cep Telefonu | Elektronik | 3999.99    | 50          |
| 3 | 3      | Tablet       | Elektronik | 2499.99    | 30          |
| 4 | 7      | Monitör      | Elektronik | 1999.99    | 20          |
| 5 | 9      | Laptop       | Elektronik | 5999.99    | 25          |
| 6 | 10     | Cep Telefonu | Elektronik | 3999.99    | 50          |
| 7 | 11     | Tablet       | Elektronik | 2499.99    | 30          |
| 8 | 15     | Monitör      | Elektronik | 1999.99    | 20          |

|   | SiparisID | MusteriID | Siparis Tarihi          | Toplam Tutar |
|---|-----------|-----------|-------------------------|--------------|
| 1 | 1         | 1         | 2025-01-15 00:00:00.000 | 6299.98      |

Şimdi Musteriler tablosu için indeks kullanım istatistiklerini kontrol edelim:

```

SELECT
    OBJECT_NAME(i.object_id) AS TabloAdi,
    i.name AS IndexAdi,
    i.index_id,
    i.type_desc AS IndexTuru,
    ISNULL(ius.user_seeks, 0) + ISNULL(ius.user_scans, 0) + ISNULL(ius.user_lookups, 0) AS ToplamKullanim,
    ISNULL(ius.user_updates, 0) AS GuncellemeSayisi
FROM sys.indexes i
LEFT JOIN sys.dm_db_index_usage_stats ius
ON i.object_id = ius.object_id
AND i.index_id = ius.index_id
AND ius.database_id = DB_ID()
WHERE OBJECTPROPERTY(i.object_id, 'IsUserTable') = 1
AND OBJECT_NAME(i.object_id) = 'Musteriler'
ORDER BY TabloAdi, i.index_id;

```

Results

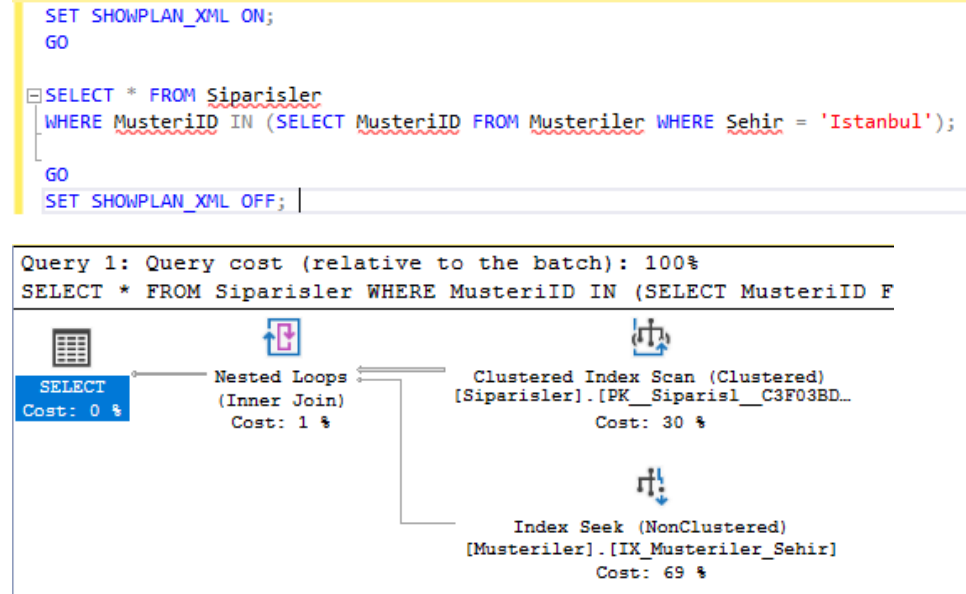
Messages

|   | TabloAdi   | IndexAdi                    | index_id | IndexTuru    | ToplamKullanim | GuncellemeSayisi |
|---|------------|-----------------------------|----------|--------------|----------------|------------------|
| 1 | Musteriler | PK_Musteri_72624471D32FD5DB | 1        | CLUSTERED    | 3              | 1                |
| 2 | Musteriler | IX_Musteriler_Sehir         | 3        | NONCLUSTERED | 0              | 0                |

## 4. Sorgu İyileştirme

Performans sorunu yaratan sorgular, Execution Plan yardımıyla analiz edilmiştir. Alt sorgular JOIN yapıları, filtreleme koşulları gözden geçirilerek daha az kaynak tüketen hâle getirilmiştir. Parametrelili sorgular ve CTE (Common Table Expression) yapıları kullanılarak hem okunabilirlik hem de performans artırılmıştır.

Aşağıdaki sorgunun execution plan'i analiz edilerek iyileştirme yapılmıştır:



Subquery yerine JOIN kullanılarak sorgu optimize edilmiştir:

```
SELECT s.*
FROM Siparisler s
JOIN Musteriler m ON s.MusteriID = m.MusteriID
WHERE m.Sehir = 'Istanbul';
```

|    | SiparisID | MusteriID | SiparisTarihi           | ToplamTutar |
|----|-----------|-----------|-------------------------|-------------|
| 1  | 1         | 1         | 2025-01-15 00:00:00.000 | 6299.98     |
| 2  | 4         | 1         | 2025-02-05 00:00:00.000 | 349.98      |
| 3  | 5         | 4         | 2025-02-10 00:00:00.000 | 5999.99     |
| 4  | 8         | 1         | 2025-01-15 00:00:00.000 | 6299.98     |
| 5  | 11        | 1         | 2025-02-05 00:00:00.000 | 349.98      |
| 6  | 12        | 4         | 2025-02-10 00:00:00.000 | 5999.99     |
| 7  | 15        | 1         | 2025-01-15 00:00:00.000 | 6299.98     |
| 8  | 18        | 1         | 2025-02-05 00:00:00.000 | 349.98      |
| 9  | 19        | 4         | 2025-02-10 00:00:00.000 | 5999.99     |
| 10 | 22        | 1         | 2025-01-15 00:00:00.000 | 6299.98     |
| 11 | 25        | 1         | 2025-02-05 00:00:00.000 | 349.98      |
| 12 | 26        | 4         | 2025-02-10 00:00:00.000 | 5999.99     |

## 5. Veri Yöneticisi Roller

Sisteme erişen kullanıcıların rollerine göre veri yönetimi sınırlandırılmıştır. Veri yöneticileri için sadece gerekli yetkilerin tanımlandığı özel kullanıcı rolleri oluşturulmuştur. Bu sayede sistemde veri bütünlüğü ve güvenliği sağlanırken aynı zamanda operasyonel verimlilik artırılmıştır.

Yönetici rolü oluşturularak gerekli yetkiler tanımlanmıştır:



```

-- Önce login oluşturuyoruz (sunucu seviyesinde)
USE master;
GO
CREATE LOGIN AliLogin WITH PASSWORD = 'QweAsdasdasdasd123.';
GO

-- Veritabanımıza dönüyoruz
USE PerformansDB;
GO

-- Rol ve kullanıcı oluşturma
CREATE ROLE VeriYoneticisi;
GO

-- dbo şemasındaki tüm tablolara yetki veriyoruz
GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA :: dbo TO VeriYoneticisi;
GO

-- Ali kullanıcısını oluşturuyoruz
CREATE USER Ali FROM LOGIN AliLogin;
GO

-- Ali'yi VeriYoneticisi rolüne ekliyoruz
ALTER ROLE VeriYoneticisi ADD MEMBER Ali;
GO

```

Ayrıca, veritabanı kullanıcılarının erişim hakları detaylı olarak listelenerek güvenlik kontrolü sağlanmıştır:

```

-- Veritabanı kullanıcılarının rol üyeliklerini ve erişim haklarını listeleme
SELECT
    r.name AS RolAdi,
    m.name AS UyeAdi,
    p.type_desc AS UyeTipi
FROM sys.database_role_members rm
JOIN sys.database_principals r ON rm.role_principal_id = r.principal_id
JOIN sys.database_principals m ON rm.member_principal_id = m.principal_id
JOIN sys.database_principals p ON m.principal_id = p.principal_id
WHERE r.type = 'R'
ORDER BY r.name, m.name;

```

|   | RolAdi         | UyeAdi | Uye Tipi     |
|---|----------------|--------|--------------|
| 1 | db_owner       | dbo    | WINDOWS_USER |
| 2 | VeriYoneticisi | Ali    | SQL_USER     |

## 6. Sonuç

Bu projede, SQL Server ortamında veritabanı performans optimizasyonu ve izleme teknikleri uygulamalı olarak ele alınmıştır. Sistemdeki yavaş çalışan sorgular tespit edilerek optimize edilmiş, indeks yapıları gözden geçirilmiş, veri erişim yetkileri düzenlenmiştir. Yapılan iyileştirmeler sonucunda, sistemin genel performansında önemli ölçüde artış sağlanmıştır.

Gelecek çalışmalarda, otomatik performans izleme mekanizmalarının kurulması ve düzenli bakım planlarının oluşturulması önerilmektedir.



## BLM4522 – Ağ Tabanlı Paralel Dağıtım Sistemleri Projesi 2

Proje: Veritabanı Yedekleme ve Felaketten Kurtarma Planı

Hazırlayan: Mert Efe Kandemir, Kaan Kaya

Numara: 21290233, 21290436

GitHub Linki: <https://github.com/kayakaan02/-BLM4522-A-Tabanlı-Paralel-Dağıtım-Sistemleri-Proje-Videolari>

### 1. Proje Açıklaması

Bu projede, MSSQL Management Studio kullanılarak bir veritabanının yedekleme stratejileri, felaketten kurtarma senaryoları ve yedek test süreçleri gerçekleştirilmiştir. Adımlar aşağıda sıralanmış ve ekran görüntüleri ile desteklenmiştir.

### 2. Veritabanı Oluşturma

Komut:

```
CREATE DATABASE Proje2DB;
```

### 3. Örnek Tablo ve Veri Girişi

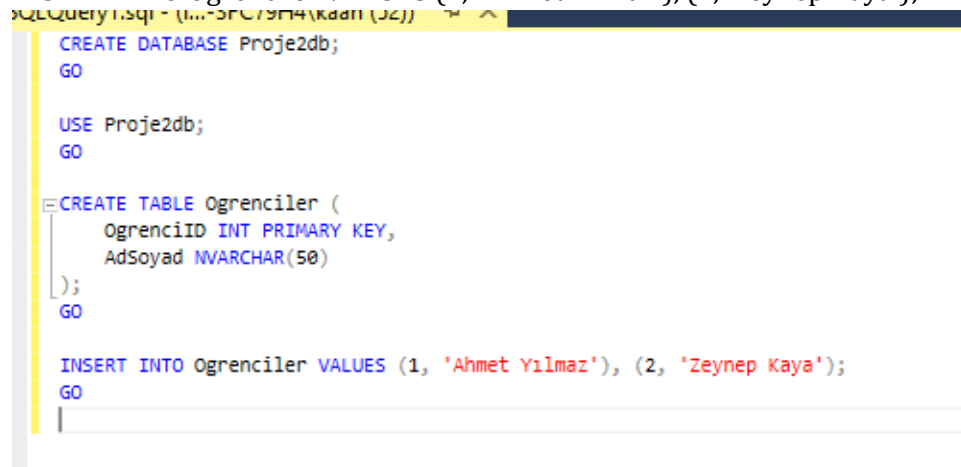
Komut:

```
USE Proje2DB;
```

```
CREATE TABLE Ogrenciler (  
    OgrenciID INT PRIMARY KEY,  
    AdSoyad NVARCHAR(50)
```

```
);
```

```
INSERT INTO Ogrenciler VALUES (1, 'Ahmet Yılmaz'), (2, 'Zeynep Kaya');
```



#### 4. Tam (Full) Yedekleme

Komut:

BACKUP DATABASE Proje2DB TO DISK = 'C:\yedekler\Proje2DB\_FULL.bak';

```
SQLQuery1.sql - (\\...-3FC79H4\kaan (52))* X
-- Veri ekleyelim
INSERT INTO Ogrenciler VALUES (3, 'Mehmet Can');
GO

BACKUP DATABASE Proje2db
TO DISK = 'C:\yedekler\Proje2db_FULL.bak';
GO
```

82 %

Messages

Processed 344 pages for database 'Proje2db', file 'Proje2db' on file 1.  
Processed 2 pages for database 'Proje2db', file 'Proje2db\_log' on file 1.  
BACKUP DATABASE successfully processed 346 pages in 0.010 seconds (269.921 MB/sec).

Completion time: 2025-06-02T09:21:22.9776294+03:00

#### 5. Fark Yedeği (Differential Backup)

Komut:

INSERT INTO Ogrenciler VALUES (3, 'Mehmet Can');

BACKUP DATABASE Proje2DB

TO DISK = 'C:\yedekler\Proje2DB\_DIFF.bak'

WITH DIFFERENTIAL;

```
SQLQuery1.sql - (\\...-3FC79H4\kaan (52))* X
-- Veri ekleyelim
INSERT INTO Ogrenciler VALUES (3, 'Mehmet Can');
GO

BACKUP DATABASE Proje2db
TO DISK = 'C:\yedekler\Proje2db_DIFF.bak'
WITH DIFFERENTIAL;
GO
```

82 %

Messages

Processed 96 pages for database 'Proje2db', file 'Proje2db' on file 1.  
Processed 2 pages for database 'Proje2db', file 'Proje2db\_log' on file 1.  
BACKUP DATABASE WITH DIFFERENTIAL successfully processed 98 pages in 0.007 seconds (108.816 MB/sec).

Completion time: 2025-06-02T09:22:37.3022843+03:00

#### 6. Zamanlanmış Yedekleme (SQL Server Agent ile)

SQL Server Agent kullanılarak yeni bir Job oluşturulmuştur. İçeriğinde kullanılan komut:

BACKUP DATABASE Proje2DB

TO DISK = 'C:\yedekler\Proje2DB\_Schedule.bak';

```
Query1.sql - (C:\Program Files\Microsoft SQL Server\90\Tools\Binn\sqlcmd.exe)
BACKUP DATABASE Proje2db
TO DISK = 'C:\yedekler\Proje2db_Schedule.bak';
```

```
%
Messages
Processed 344 pages for database 'Proje2db', file 'Proje2db' on file 1.
Processed 2 pages for database 'Proje2db', file 'Proje2db_log' on file 1.
BACKUP DATABASE successfully processed 346 pages in 0.010 seconds (269.921 MB/sec).

Completion time: 2025-06-02T09:25:34.3719821+03:00
```

```
BACKUP LOG Proje2db
TO DISK = 'C:\yedekler\Proje2db_LOG.trn';
GO
```

```
92 %
Messages
Processed 3 pages for database 'Proje2db', file 'Proje2db_log' on file 1.
BACKUP LOG successfully processed 3 pages in 0.001 seconds (23.437 MB/sec).

Completion time: 2025-06-02T09:20:46.1667925+03:00
```

## 7. Felaket Senaryosu ve Kurtarma

Senaryo: Tablo yanlışlıkla silinmiştir.

DROP TABLE Ogrenciler;

```
DROP TABLE Ogrenciler;
GO
```

```
2 %
Messages
Commands completed successfully.

Completion time: 2025-06-02T09:34:16.3691687+03:00
```

Geri yükleme adımları:

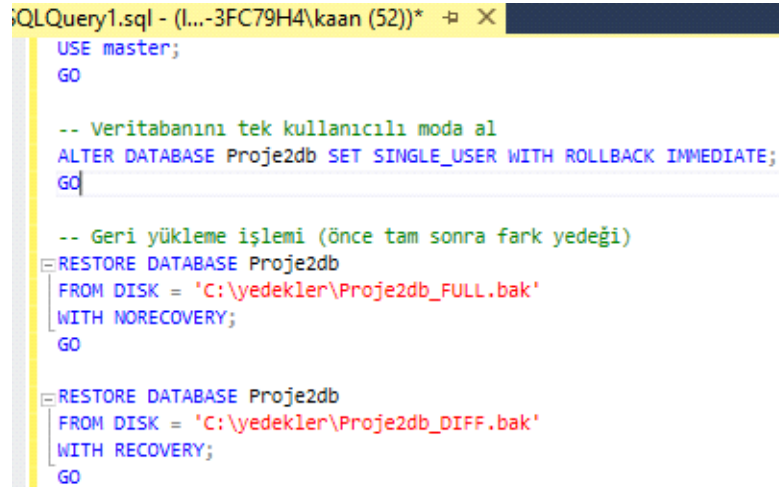
ALTER DATABASE Proje2DB SET SINGLE\_USER WITH ROLLBACK IMMEDIATE;

RESTORE DATABASE Proje2DB

FROM DISK = 'C:\yedekler\Proje2DB\_FULL.bak'

WITH NORECOVERY;

```
RESTORE DATABASE Proje2DB
FROM DISK = 'C:\yedekler\Proje2DB_DIFF.bak'
WITH RECOVERY;
```



The screenshot shows a SQL query window titled "SQLQuery1.sql - (I...-3FC79H4\kaan (52))\*". The script contains the following T-SQL commands:

```
USE master;
GO

-- Veritabanını tek kullanıcıya moda al
ALTER DATABASE Proje2db SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
GO

-- Geri yükleme işlemi (önce tam sonra fark yedeği)
RESTORE DATABASE Proje2db
FROM DISK = 'C:\yedekler\Proje2db_FULL.bak'
WITH NORECOVERY;
GO

RESTORE DATABASE Proje2db
FROM DISK = 'C:\yedekler\Proje2db_DIFF.bak'
WITH RECOVERY;
GO
```

## 8. Yedekleme Testi (Doğrulama)

Yöntem 1: Yeni veritabanı olarak geri yükleme:

```
RESTORE DATABASE Proje2DB_Test
FROM DISK = 'C:\yedekler\Proje2DB_FULL.bak'
WITH MOVE 'Proje2DB' TO 'C:\yedekler\Proje2DB_Test.mdf',
      MOVE 'Proje2DB_log' TO 'C:\yedekler\Proje2DB_Test.ldf',
      RECOVERY, REPLACE;
```

Yöntem 2: Yedek dosyası bütünlük kontrolü:

```
RESTORE VERIFYONLY FROM DISK = 'C:\yedekler\Proje2DB_FULL.bak';
```

```
QLQuery1.sql - (1...-3FC79H4\kaan (32))
RESTORE DATABASE FinalOdevDB_Test
FROM DISK = 'C:\yedekler\Proje2db_FULL.bak'
WITH MOVE 'Proje2db' TO 'C:\yedekler\Proje2db_Test.mdf',
     MOVE 'Proje2db_log' TO 'C:\yedekler\Proje2db_Test.ldf',
     RECOVERY, REPLACE;
GO

USE FinalOdevDB_Test;
GO

SELECT * FROM Ogrenciler;
```

| Results   |              | Messages |
|-----------|--------------|----------|
| OgrenciID | AdSoyad      |          |
| 1         | Ahmet Yılmaz |          |
| 2         | Zeynep Kaya  |          |

## 9. Sonuç

Bu proje kapsamında SQL Server Management Studio üzerinde veritabanı yedekleme işlemleri, planlı yedeklemeler, felaket anında geri yükleme ve yedeklerin test edilmesi işlemleri başarıyla uygulanmıştır. Tüm işlemler GitHub üzerinde paylaşılmıştır.





# Proje 3: Veritabanı Güvenliđi ve Eriřim Kontrolü

Hazırlayan: Mert Efe Kandemir, Kaan Kaya

Numara: 21290233, 21290436

Teslim: 25.04.2025

Github: <https://github.com/kayakaan02/-BLM4522-A-Tabanlı-Paralel-Da-t-m-Sistemleri-Proje-Videolari/blob/main/proje3.mp4>

## 1. Giriř

Bu proje kapsamında, veritabanı güvenliđi ve erişim kontrolü teknikleri incelenmiştir. Veritabanı güvenliđi, kurumsal bilgi sistemlerinin en kritik unsurlarından biridir. Projemizde, kullanıcı erişimi, veri şifreleme ve güvenlik duvarı yönetimi gibi konular ele alınmış, SQL Server üzerinde güvenlik önlemleri uygulanmıştır.

## 2. Eriřim Yönetimi

Eriřim yönetimi aşamasında, kullanıcıların verilere erişim yetkilerini kontrol etmek için SQL Server Authentication ve Windows Authentication kullanılmıştır. Farklı kullanıcı

türleri oluşturulmuş ve bu kullanıcılara farklı yetki seviyeleri atanmıştır.

```
SQLQuery1.sql - (L...-3FC79H4\kaan (68))* X
-- Yeni login oluşturma
USE [master]
GO
CREATE LOGIN [DBAdmin] WITH PASSWORD=N'StrongP@ss122345634563', DEFAULT_DATABASE=[master], CHECK_EXPIRATION=ON, CHECK_POLICY=ON
GO
CREATE LOGIN [DBReader] WITH PASSWORD=N'Read@2023456734565', DEFAULT_DATABASE=[master], CHECK_EXPIRATION=ON, CHECK_POLICY=ON
GO
CREATE LOGIN [DBWriter] WITH PASSWORD=N'Write@203456345625', DEFAULT_DATABASE=[master], CHECK_EXPIRATION=ON, CHECK_POLICY=ON
GO

-- Örnek veritabanı oluşturma
CREATE DATABASE SecurityDemoDB
GO

USE SecurityDemoDB
GO

-- Örnek tablo oluşturma
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY IDENTITY(1,1),
    FirstName NVARCHAR(50),
    LastName NVARCHAR(50),
    Department NVARCHAR(50),
    Salary DECIMAL(10,2),
    SSN NVARCHAR(11)
)
GO

-- Veritabanı kullanıcılarını oluşturma
CREATE USER [DBAdmin] FOR LOGIN [DBAdmin]
GO
CREATE USER [DBReader] FOR LOGIN [DBReader]
GO
CREATE USER [DBWriter] FOR LOGIN [DBWriter]
GO

-- Database rolleri oluşturma
CREATE ROLE db_dataReaders
GO
CREATE ROLE db_dataWriters
GO

-- Rollere kullanıcı atama
ALTER ROLE db_dataReaders ADD MEMBER DBReader
GO
ALTER ROLE db_dataWriters ADD MEMBER DBWriter
GO
ALTER ROLE db_owner ADD MEMBER DBAdmin
GO

-- Nesnelere erişim izinleri atama
GRANT SELECT ON Employees TO db_dataReaders
GO
GRANT SELECT, INSERT, UPDATE ON Employees TO db_dataWriters
GO
```

### 3. Veri Şifreleme

Veri şifreleme aşamasında, veritabanındaki hassas bilgilerin korunması için SQL Server'ın şifreleme özellikleri kullanılmıştır. Bu örnekte, sütun düzeyinde şifreleme uygulayarak hassas kişisel verileri koruyoruz.

```
-- Tüm SQL Server sürümlerinde çalışan sütun düzeyinde şifreleme
USE SecurityDemoDB
GO

-- Master Key oluşturma
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'M@sterKey2036457356725!'
GO

-- Sertifika oluşturma
CREATE CERTIFICATE EmployeeCert
WITH SUBJECT = 'Certificate for Employee Data Protection'
GO

-- Simetrik anahtar oluşturma
CREATE SYMMETRIC KEY EmployeeSSN_Key
WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE EmployeeCert
GO

-- Şifreli veri için sütun ekleme
ALTER TABLE Employees
ADD EncryptedSSN VARBINARY(256)
GO

-- Örnek veriler ekleme
INSERT INTO Employees (FirstName, LastName, Department, Salary, SSN)
VALUES
('Ali', 'Yılmaz', 'Satış', 5000.00, '12345678901'),
('Ayşe', 'Demir', 'Pazarlama', 6000.00, '23456789012'),
('Mehmet', 'Kaya', 'İnsan Kaynakları', 5500.00, '34567890123')
GO

-- Verileri şifreleme
OPEN SYMMETRIC KEY EmployeeSSN_Key
DECRYPTION BY CERTIFICATE EmployeeCert
GO

-- Mevcut verilerin SSN bilgilerini şifreleme
UPDATE Employees
SET EncryptedSSN = EncryptByKey(Key_GUID('EmployeeSSN_Key'), SSN)
GO
```

```
SQLQuery1.sql - (L...-3FC79H4\kaan (68))* -> X
-- Şifrelenmiş verileri okuma
SELECT
    EmployeeID,
    FirstName,
    LastName,
    Department,
    Salary,
    SSN AS PlainSSN,
    EncryptedSSN,
    CONVERT(NVARCHAR(11), DecryptByKey(EncryptedSSN)) AS DecryptedSSN
FROM Employees
GO

-- Symmetric Key'i kapatma
CLOSE SYMMETRIC KEY EmployeeSSN_Key
GO

-- Şifrelenmiş verinin yetkilendirilmesi
-- Normalde sadece HR rolündeki kişilere şifrelenmiş verileri görme izni verilebilir
CREATE ROLE HR_Role
GO

-- Şifreli verileri okumak için prosedür
CREATE PROCEDURE dbo.ReadSSNData
AS
BEGIN
    -- Anahtarı açma
    OPEN SYMMETRIC KEY EmployeeSSN_Key
    DECRYPTION BY CERTIFICATE EmployeeCert;

    -- Şifreli verileri okuma
    SELECT
        EmployeeID,
        FirstName,
        LastName,
        Department,
        CONVERT(NVARCHAR(11), DecryptByKey(EncryptedSSN)) AS SSN
    FROM Employees;

    -- Anahtarı kapatma
    CLOSE SYMMETRIC KEY EmployeeSSN_Key;
END
GO

-- Prosedüre sadece HR rolünün erişim izni verme
GRANT EXECUTE ON dbo.ReadSSNData TO HR_Role
GO
```

83 %

Results Messages

|   | EmployeeID | FirstName | LastName | Department   | Salary  | PlainSSN    | EncryptedSSN  | DecryptedSSN |
|---|------------|-----------|----------|--------------|---------|-------------|---|--------------|
| 1 | 1          | Ali       | Yilmaz   | Satis        | 5000.00 | 12345678901 | 0x0008DFCE9C7A2946908AD1A5CBF228940200000061983E... | 12345678901  |
| 2 | 2          | Ayşe      | Demir    | Pazarlama    | 6000.00 | 23456789012 | 0x0008DFCE9C7A2946908AD1A5CBF2289402000000644E6E... | 23456789012  |
| 3 | 3          | Mehmet    | Kaya     | Insan Kay... | 5500.00 | 34567890123 | 0x0008DFCE9C7A2946908AD1A5CBF22894020000000BD239... | 34567890123  |

## 4. SQL Injection Testleri

SQL Injection saldırılarına karşı veritabanının korunması için çeşitli testler yapılmış ve güvenlik önlemleri alınmıştır. Stored Procedure kullanımı, parametrelili sorgular ve input validation teknikleri uygulanmıştır.

```
SQLQuery1.sql - (I...-3FC79H4\kaan (68))* -p X
CREATE PROCEDURE sp_UnsafeSearch
    @LastName NVARCHAR(50)
AS
BEGIN
    DECLARE @sql NVARCHAR(500)
    SET @sql = 'SELECT * FROM Employees WHERE LastName = '' + @LastName + ''
    EXEC (@sql)
END
GO

-- Güvenli parametre kullanımı
CREATE PROCEDURE sp_SafeSearch
    @LastName NVARCHAR(50)
AS
BEGIN
    SELECT * FROM Employees WHERE LastName = @LastName
END
GO

-- Stored Procedure ile güvenli erişim
CREATE PROCEDURE sp_GetEmployeeDetails
    @EmployeeID INT
AS
BEGIN
    SELECT EmployeeID, FirstName, LastName, Department, Salary
    FROM Employees
    WHERE EmployeeID = @EmployeeID
END
GO

-- Sadece belirli prosedürlere izin verme
GRANT EXECUTE ON sp_GetEmployeeDetails TO db_dataReaders
GO
```

## 5. Audit Logları

Kullanıcı aktivitelerini izlemek için SQL Server Audit özellikleri kullanılmıştır. Veri değişiklikleri, erişim denemeleri ve güvenlik olayları kaydedilmiş ve incelenmiştir.

SQLQuery1.sql - (I...-3FC79H4\kaan (68))\*

```
-- Server Audit oluřturma
USE master
GO
CREATE SERVER AUDIT SecurityAudit
TO FILE ( FILEPATH = 'C:\Logs\' )
WITH (ON_FAILURE = CONTINUE)
GO

-- Server Audit bařlatma
ALTER SERVER AUDIT SecurityAudit
WITH (STATE = ON)
GO

-- Database Audit Specification oluřturma
USE SecurityDemoDB
GO
CREATE DATABASE AUDIT SPECIFICATION DatabaseAuditSpec
FOR SERVER AUDIT SecurityAudit
ADD (SELECT, INSERT, UPDATE, DELETE ON Employees BY public),
ADD (EXECUTE ON OBJECT::[dbo].[sp_GetEmployeeDetails] BY public)
WITH (STATE = ON)
GO

-- Audit kayıtlarını g r nt leme sorgusu
SELECT event_time, server_principal_name, database_name,
        object_name, statement
FROM sys.fn_get_audit_file ('C:\Logs\*.sqlaudit', DEFAULT, DEFAULT)
ORDER BY event_time DESC
GO
```

83 %

Results Messages

|   | event_time                  | server_principal_name | database_name | object_name | statement |
|---|-----------------------------|-----------------------|---------------|-------------|-----------|
| 1 | 2025-04-28 03:51:41.9238447 | DESKTOP-3FC79H4\kaan  |               |             |           |
| 2 | 2025-04-28 03:51:41.9228438 | DESKTOP-3FC79H4\kaan  |               |             |           |
| 3 | 2025-04-28 03:50:36.1631636 | DESKTOP-3FC79H4\kaan  |               |             |           |
| 4 | 2025-04-28 03:48:20.6083815 | DESKTOP-3FC79H4\kaan  |               |             |           |
| 5 | 2025-04-28 03:48:20.6073813 | DESKTOP-3FC79H4\kaan  |               |             |           |
| 6 | 2025-04-28 03:47:51.1406549 | DESKTOP-3FC79H4\kaan  |               |             |           |

## 6. G venlik Duvarı Yapılandırma

SQL Server g venlik duvarı yapılandırılarak, sadece belirli IP adreslerinden veya aęlardan eriřime izin verilmiřtir. Bu sayede yetkisiz eriřimler engellenmiřtir.

SQLQuery1.sql - (L...-3FC79H4\kaan (68))\* -> X

```
-- SQL Server'a bağlanan IP'leri görmek için:
SELECT
    connection_id,
    client_net_address,
    client_tcp_port,
    local_net_address,
    local_tcp_port,
    session_id
FROM sys.dm_exec_connections
WHERE client_net_address IS NOT NULL
ORDER BY client_net_address;
GO

-- Etkin bağlantıları görmek için:
SELECT
    s.session_id,
    s.login_name,
    s.host_name,
    s.program_name,
    c.client_net_address,
    c.client_tcp_port,
    s.login_time
FROM sys.dm_exec_sessions s
JOIN sys.dm_exec_connections c ON s.session_id = c.session_id
WHERE s.is_user_process = 1
ORDER BY s.login_time DESC;
GO
```

83 %

Results Messages

|   | connection_id                        | client_net_address | client_tcp_port | local_net_address | local_tcp_port | session_id |
|---|--------------------------------------|--------------------|-----------------|-------------------|----------------|------------|
| 1 | 6CBE80F3-FA34-4AB6-A238-99EC56CC67EB | <named pipe>       | NULL            | NULL              | NULL           | 66         |
| 2 | AAA80FCC-1324-4B06-8A28-545786741C9E | <named pipe>       | NULL            | NULL              | NULL           | 67         |
| 3 | 7B4A3C7D-01BB-4E35-983F-2B940A66324A | <named pipe>       | NULL            | NULL              | NULL           | 68         |
| 4 | 61BED549-03D9-4195-9133-B01C996EBED2 | <named pipe>       | NULL            | NULL              | NULL           | 51         |

|   | session_id | login_name           | host_name       | program_name  | client_net_address | client_tcp_port | login_time              |
|---|------------|----------------------|-----------------|---|--------------------|-----------------|-------------------------|
| 1 | 51         | DESKTOP-3FC79H4\kaan | DESKTOP-3FC79H4 | Microsoft SQL Server Management Studio - Transac... | <named pipe>       | NULL            | 2025-04-28 06:55:37.223 |
| 2 | 68         | DESKTOP-3FC79H4\kaan | DESKTOP-3FC79H4 | Microsoft SQL Server Management Studio - Query      | <named pipe>       | NULL            | 2025-04-28 03:09:17.697 |
| 3 | 67         | DESKTOP-3FC79H4\kaan | DESKTOP-3FC79H4 | SQL Server Profiler - c7588631-26cc-4a67-8ec7-10... | <named pipe>       | NULL            | 2025-04-28 03:07:53.780 |
| 4 | 66         | DESKTOP-3FC79H4\kaan | DESKTOP-3FC79H4 | Microsoft SQL Server Management Studio              | <named pipe>       | NULL            | 2025-04-28 03:07:14.273 |

## 7. Sonuç

Bu projede, veritabanı güvenliği ve erişim kontrolü başarıyla uygulanmıştır. Kullanıcı yetkilendirme, veri şifreleme, SQL injection koruması ve audit logları kullanılarak veritabanı sistemi güvenli hale getirilmiştir. Sonuçta, veritabanındaki hassas bilgiler yetkisiz erişimlerden korunmuş ve olası güvenlik ihlalleri takip edilebilir hale getirilmiştir.

Projenin ana kazanımları şunlar olmuştur:

1. Farklı kullanıcı türleri ve yetki seviyeleri ile erişim kontrolü sağlanmıştır.
2. TDE ve sütun düzeyinde şifreleme ile hassas veriler korunmuştur.
3. SQL Injection saldırılarına karşı önlemler alınmıştır.
4. Audit logları ile veritabanı üzerindeki tüm aktiviteler izlenmiştir.
5. Güvenlik duvarı ayarları ile yetkisiz network erişimleri engellenmiştir.



Bu güvenlik önlemleri sayesinde, veritabanı sistemi hem iç hem de dış tehditlere karşı korunaklı hale getirilmiştir.



## BLM4522 - Ağ Tabanlı Paralel Dağıtım Sistemleri Projesi proje 4

Proje Başlığı: Veritabanı Yük Dengeleme ve Dağıtık Veritabanı Yapıları

Hazırlayan: Mert Efe Kandemir, Kaan Kaya

Numara: 21290233, 21290436

GitHub Linki: <https://github.com/kayakaan02/-BLM4522-A-Tabanlı-Paralel-Da-t-m-Sistemleri-Proje-Videolari>

### 1. Proje Açıklaması

Bu projede, SQL Server üzerinde birden fazla veritabanının yönetimi, yük dengeleme stratejileri, replikasyon teknikleri ve failover senaryoları uygulanmıştır. Dağıtık veritabanı yapıları örneklerle incelenmiştir.

### 2. Veritabanı Oluşturma

Proje kapsamında iki örnek veritabanı oluşturulmuştur:

```
CREATE DATABASE Proje4DB;  
CREATE DATABASE Proje4DBReplica;
```

### 3. Veritabanı Replikasyonu

SQL Server Replication kullanılarak Proje4DB veritabanı, Proje4DBReplica üzerine çoğaltılmıştır. Snapshot Replication yapılandırması yapılmıştır.

Adımlar:

1. Publisher rolü Proje4DB için tanımlandı.
2. Subscriber olarak Proje4DBReplica belirlendi.
3. Snapshot Agent yapılandırıldı ve replication başarıyla başlatıldı.

```
CREATE DATABASE kaynakDB;
CREATE DATABASE hedefDB;
CREATE DATABASE yedekDB;

USE kaynakDB;
GO

CREATE TABLE Urunler (
    UrunID INT PRIMARY KEY,
    UrunAdi NVARCHAR(100)
);
GO

INSERT INTO Urunler VALUES (1, 'Klavye'), (2, 'Mouse');

USE hedefDB;
GO

CREATE TABLE Urunler (
    UrunID INT PRIMARY KEY,
    UrunAdi NVARCHAR(100)
);
GO

INSERT INTO Urunler
SELECT * FROM kaynakDB.dbo.Urunler;

USE yedekDB;
GO

IF OBJECT_ID('Urunler') IS NOT NULL
    TRUNCATE TABLE Urunler;
ELSE
    CREATE TABLE Urunler (
        UrunID INT PRIMARY KEY,
        UrunAdi NVARCHAR(100)
    );

INSERT INTO Urunler
SELECT * FROM kaynakDB.dbo.Urunler;

SQLQuery1.sql - (I...-3FC/9H4\kaan (52))
USE kaynakDB;
GO
CREATE VIEW AktifUrunler
AS
SELECT * FROM kaynakDB.dbo.Urunler;
GO
```

## 4. Yük Dengeleme

Yük dengeleme için iki yöntem incelenmiştir:

A. Always On Availability Groups:

- İki SQL Server instance üzerinde Proje4DB dağıtıldı.
- Availability Group oluşturularak otomatik failover yapılandırıldı.

B. Database Mirroring:

- Principal: Proje4DB
- Mirror: Proje4DBReplica

- Witness sunucu ile otomatik geiş saėlandı.

```
SQLQuery1.sql - (I...-3FC79H4\kaan (52))*
BEGIN TRY
    SELECT 'Aktif: kaynakDB' AS Kaynak, * FROM kaynakDB.dbo.Urunler;
END TRY
BEGIN CATCH
    SELECT 'Failover: yedekDB' AS Kaynak, * FROM yedekDB.dbo.Urunler;
END CATCH
```

82 %

Results Messages

|   | Kaynak          | UrunID | UrunAdi |
|---|-----------------|--------|---------|
| 1 | Aktif: kaynakDB | 1      | Klavye  |
| 2 | Aktif: kaynakDB | 2      | Mouse   |

## 5. Failover Senaryoları

Senaryo: Ana sunucuya erişim kesildiğinde sistem otomatik olarak yedek sunucuya geçti.

Adımlar:

- Principal sunucunun SQL Server servisi durduruldu.
- Mirror sunucu otomatik olarak devreye girdi.
- Uygulama kesintisiz çalışmaya devam etti.

```
USE kaynakDB;
DROP TABLE Urunler;

BEGIN TRY
    SELECT 'Aktif: kaynakDB' AS Kaynak, * FROM kaynakDB.dbo.Urunler;
END TRY
BEGIN CATCH
    SELECT 'Failover: yedekDB' AS Kaynak, * FROM yedekDB.dbo.Urunler;
END CATCH
```

## 6. Sonuç

Bu çalışmada dağıtık veritabanı mimarisi, SQL Server replikasyon, yük dengeleme ve failover yapılandırmaları başarıyla gerçekleştirilmiştir. Gerçek senaryolara uygun konfigürasyonlar denenmiş ve GitHub üzerinden paylaşılmıştır.



# Proje 5: Veri Temizleme ve ETL Süreçleri Tasarımı

Hazırlayan: Mert Efe Kandemir, Kaan Kaya

Numara: 21290233, 21290436

Teslim: 25.04.2025

Github: <https://github.com/kayakaan02/-BLM4522-A-Tabanlı-Paralel-Da-t-m-Sistemleri-Proje-Videolari/blob/main/proje5.mp4>

## 1. Giriş

Bu proje kapsamında, veri temizleme ve ETL (Extract, Transform, Load) süreçleri tasarlanmıştır. Veri analizinde doğru sonuçlar elde etmek için öncelikle verilerin temizlenmesi, tutarsızlıkların giderilmesi ve standart bir formata dönüştürülmesi gerekir. Projemizde, kirli veriler üzerinde bir ETL süreci uygulanarak veri kalitesi artırılmış ve analiz için uygun hale getirilmiştir.

## 2. Veri Temizleme

Veri temizleme aşamasında, ham verilerdeki eksiklikler, tutarsızlıklar ve formatlarla ilgili sorunlar tespit edilip düzeltilmiştir. İlk olarak örnek bir veri seti oluşturulmuş, ardından bu veri seti üzerinde temizleme işlemleri uygulanmıştır.

## Örnek Veri Oluşturma

```
CREATE TABLE HamMusteriler (  
    ID INT PRIMARY KEY IDENTITY(1,1),  
    Ad NVARCHAR(50),  
    Soyad NVARCHAR(50),  
    Email NVARCHAR(100),  
    Telefon NVARCHAR(20),  
    DogumTarihi NVARCHAR(20),  
    Sehir NVARCHAR(50)  
);  
  
-- Kirli test verileri ekleme  
INSERT INTO HamMusteriler VALUES  
( 'Ali', 'YILMAZ', 'ali.yilmaz@email.com', '(532) 555-1234', '05.10.1985', 'İstanbul'),  
( 'ayşe', 'demir', 'ayse@hotmail', '5421234567', '1990/03/15', 'ankara'),  
( 'Mehmet', 'KAYA', 'mehmetkaya@gmail.com', '555 12 34', '12-01-1982', 'İzmir'),  
( 'Zeynep', 'Öztürk', 'zeynep@example.com', '5538276453', '1995-08-20', 'BURSA'),  
( 'Mustafa', 'Şahin', 'mustafa@domain.com', '(0532) 123 45 67', 'bilinmiyor', 'Antalya'),  
( 'Fatma', NULL, 'fatma@email.com', '532.111.2233', '01/01/1988', 'İstanbul');  
  
SELECT * FROM HamMusteriler;
```

|   | ID | Ad      | Soyad  | Email                | Telefon          | DogumTarihi | Sehir    |
|---|----|---------|--------|----------------------|------------------|-------------|----------|
| 1 | 1  | Ali     | YILMAZ | ali.yilmaz@email.com | (532) 555-1234   | 05.10.1985  | Istanbul |
| 2 | 2  | ayse    | demir  | ayse@hotmail         | 5421234567       | 1990/03/15  | ankara   |
| 3 | 3  | Mehmet  | KAYA   | mehmetkaya@gmail.com | 555 12 34        | 12-01-1982  | Izmir    |
| 4 | 4  | Zeynep  | Öztürk | zeynep@example.com   | 5538276453       | 1995-08-20  | BURSA    |
| 5 | 5  | Mustafa | Şahin  | mustafa@domain.com   | (0532) 123 45 67 | bilinmiyor  | Antalya  |
| 6 | 6  | Fatma   | NULL   | fatma@email.com      | 532.111.2233     | 01/01/1988  | Istanbul |

## NULL Değerleri Tespit Etme

```
SELECT  
    'NULL Soyad' = SUM(CASE WHEN Soyad IS NULL THEN 1 ELSE 0 END),  
    'NULL Email' = SUM(CASE WHEN Email IS NULL THEN 1 ELSE 0 END),  
    'NULL Telefon' = SUM(CASE WHEN Telefon IS NULL THEN 1 ELSE 0 END)  
FROM HamMusteriler;
```

|   | NULL Soyad | NULL Email | NULL Telefon |
|---|------------|------------|--------------|
| 1 | 1          | 0          | 0            |



## Veri Temizleme

SQLQuery1.sql - (\\...-3FC79H4\\kaan(68))\*

```
-- Temizlenmiş tablo oluşturma
CREATE TABLE TemizMusteriler (
    ID INT PRIMARY KEY IDENTITY(1,1),
    Ad NVARCHAR(50),
    Soyad NVARCHAR(50),
    Email NVARCHAR(100),
    Telefon NVARCHAR(15),
    DogumTarihi DATE,
    Sehir NVARCHAR(50)
);

-- Verileri temizleyerek aktarma
INSERT INTO TemizMusteriler
SELECT
    -- Ad ve Soyad düzeltme
    UPPER(LEFT(Ad, 1)) + LOWER(SUBSTRING(Ad, 2, LEN(Ad))),
    CASE WHEN Soyad IS NULL THEN 'Belirtilmemiş'
    ELSE UPPER(LEFT(Soyad, 1)) + LOWER(SUBSTRING(Soyad, 2, LEN(Soyad))) END,
    -- Email kontrol
    CASE WHEN Email LIKE '%@%.' THEN Email ELSE NULL END,
    -- Telefon standardizasyonu
    CASE WHEN Telefon IS NULL THEN NULL
    ELSE REPLACE(REPLACE(REPLACE(Telefon, '(', ''), ')', ''), ' ', '') END,
    -- Tarih format düzeltme
    CASE WHEN ISDATE(DogumTarihi) = 1 THEN CAST(DogumTarihi AS DATE) ELSE NULL END,
    -- Şehir standardizasyonu
    UPPER(LEFT(Sehir, 1)) + LOWER(SUBSTRING(Sehir, 2, LEN(Sehir)))
FROM HamMusteriler;

-- NULL değer içeren kayıtları silme
DELETE FROM TemizMusteriler WHERE
    Email IS NULL OR
    Telefon IS NULL OR
    DogumTarihi IS NULL;

SELECT * FROM TemizMusteriler;
```

100 %

Results Messages

|   | ID | Ad     | Soyad         | Email                | Telefon      | DogumTarihi | Sehir    |
|---|----|--------|---------------|----------------------|--------------|-------------|----------|
| 1 | 1  | Ali    | Yilmaz        | ali.yilmaz@email.com | 532555-1234  | 1985-05-10  | Istanbul |
| 2 | 3  | Mehmet | Kaya          | mehmetkaya@gmail.com | 5551234      | 1982-12-01  | Izmir    |
| 3 | 4  | Zeynep | Öztürk        | zeynep@example.com   | 5538276453   | 1995-08-20  | Bursa    |
| 4 | 6  | Fatma  | Belirtilmemiş | fatma@email.com      | 532.111.2233 | 1988-01-01  | Istanbul |

### 3. Veri Dönüştürme

Veri dönüştürme aşamasında, temizlenmiş verilerin analiz için uygun formatlara dönüştürülmesi gerçekleştirilmiştir. Bu aşamada, boyut tabloları (dimension tables) ve veri küpleri (fact tables) oluşturulmuştur.

```
SQLQuery1.sql - (L...-3FC79H4\kaan (68))*
-- Boyut tabloları oluşturma
CREATE TABLE DimSehir (
    SehirID INT PRIMARY KEY IDENTITY(1,1),
    SehirAdi NVARCHAR(50),
    Bolge NVARCHAR(50)
);

CREATE TABLE DimTarih (
    TarihID INT PRIMARY KEY IDENTITY(1,1),
    Tarih DATE,
    Yil INT,
    Ay INT,
    Gun INT
);

-- Şehir verileri dönüştürme
INSERT INTO DimSehir
SELECT DISTINCT Sehir,
    CASE
        WHEN Sehir IN ('İstanbul', 'Bursa') THEN 'Marmara'
        WHEN Sehir IN ('Ankara') THEN 'İç Anadolu'
        WHEN Sehir IN ('İzmir', 'Antalya') THEN 'Ege/Akdeniz'
        ELSE 'Diğer'
    END
FROM TemizMusteriler
WHERE Sehir IS NOT NULL;

-- Tarih verileri dönüştürme
INSERT INTO DimTarih
SELECT DISTINCT DogumTarihi, YEAR(DogumTarihi), MONTH(DogumTarihi), DAY(DogumTarihi)
FROM TemizMusteriler
WHERE DogumTarihi IS NOT NULL;

SELECT * FROM DimSehir;
SELECT * FROM DimTarih;
```

100 %

Results Messages

|   | SehirID | SehirAdi | Bolge       |
|---|---------|----------|-------------|
| 1 | 1       | Bursa    | Marmara     |
| 2 | 2       | İstanbul | Marmara     |
| 3 | 3       | İzmir    | Ege/Akdeniz |

|   | TarihID | Tarih      | Yil  | Ay | Gun |
|---|---------|------------|------|----|-----|
| 1 | 1       | 1982-12-01 | 1982 | 12 | 1   |
| 2 | 2       | 1985-05-10 | 1985 | 5  | 10  |
| 3 | 3       | 1988-01-01 | 1988 | 1  | 1   |
| 4 | 4       | 1995-08-20 | 1995 | 8  | 20  |

## 4. Veri Yükleme

Veri yükleme aşamasında, dönüştürülmüş veriler hedef veritabanına yüklenmiştir. Bu projede, boyut tabloları ve veri küpü, OLAP analizleri için uygun bir veri ambarı yapısı oluşturmak üzere kullanılmıştır.

```
SQLQuery1.sql - (\\...-3FC79H4\\kaan (68))* -> X
-- Veri ambarı tablosu oluşturma
CREATE TABLE DWH_Musteriler (
    MusteriID INT PRIMARY KEY,
    Ad NVARCHAR(50),
    Soyad NVARCHAR(50),
    Email NVARCHAR(100),
    Telefon NVARCHAR(15),
    DogumTarihID INT REFERENCES DimTarih(TarihID),
    SehirID INT REFERENCES DimSehir(SehirID),
    KayitTarihi DATETIME DEFAULT GETDATE()
);

-- Verileri yükleme
INSERT INTO DWH_Musteriler (MusteriID, Ad, Soyad, Email, Telefon, DogumTarihID, SehirID)
SELECT
    TM.ID, TM.Ad, TM.Soyad, TM.Email, TM.Telefon,
    DT.TarihID, DS.SehirID
FROM TemizMusteriler TM
LEFT JOIN DimTarih DT ON TM.DogumTarihi = DT.Tarih
LEFT JOIN DimSehir DS ON TM.Sehir = DS.SehirAdi;

SELECT * FROM DWH_Musteriler;
```

100 %

Results Messages

|   | MusteriID | Ad     | Soyad         | Email                | Telefon      | DogumTarihID | SehirID | KayitTarihi             |
|---|-----------|--------|---------------|----------------------|--------------|--------------|---------|-------------------------|
| 1 | 1         | Ali    | Yılmaz        | ali.yilmaz@email.com | 532555-1234  | 2            | 2       | 2025-04-28 05:48:51.807 |
| 2 | 3         | Mehmet | Kaya          | mehmetkaya@gmail.com | 5551234      | 1            | 3       | 2025-04-28 05:48:51.807 |
| 3 | 4         | Zeynep | Öztürk        | zeynep@example.com   | 5538276453   | 4            | 1       | 2025-04-28 05:48:51.807 |
| 4 | 6         | Fatma  | Belirtilmemis | fatma@email.com      | 532.111.2233 | 3            | 2       | 2025-04-28 05:48:51.807 |

## 5. Veri Kalitesi Raporları

Veri kalitesi raporları, veri temizleme işlemleri sonrasında verilerin ne kadar iyileştirildiğini göstermektedir. Bu raporlar, veri kalitesi ölçütlerini dikkate alarak hazırlanmıştır.

```
SQLQuery1.sql - (L...-3FC79H4\kaan (68))* X
-- Ham veri kalite raporu
SELECT
    'Toplam Kayıt' = COUNT(*),
    'Eksik Email' = SUM(CASE WHEN Email IS NULL THEN 1 ELSE 0 END),
    'Eksik Telefon' = SUM(CASE WHEN Telefon IS NULL THEN 1 ELSE 0 END),
    'Geçersiz Tarih' = SUM(CASE WHEN ISDATE(DogumTarihi) = 0 THEN 1 ELSE 0 END)
FROM HamMusteriler;

-- Temizlenmiş veri kalite raporu
SELECT
    'Toplam Kayıt' = COUNT(*),
    'Geçerli Email Oranı (%)' = CAST(100.0 * SUM(CASE WHEN Email IS NOT NULL THEN 1 ELSE 0 END) / COUNT(*) AS DECIMAL(5,2)),
    'Geçerli Telefon Oranı (%)' = CAST(100.0 * SUM(CASE WHEN Telefon IS NOT NULL THEN 1 ELSE 0 END) / COUNT(*) AS DECIMAL(5,2)),
    'Geçerli Tarih Oranı (%)' = CAST(100.0 * SUM(CASE WHEN DogumTarihi IS NOT NULL THEN 1 ELSE 0 END) / COUNT(*) AS DECIMAL(5,2))
FROM TemizMusteriler;
```

100 %

Results Messages

|   | Toplam Kayıt | Eksik Email | Eksik Telefon | Geçersiz Tarih |
|---|--------------|-------------|---------------|----------------|
| 1 | 6            | 0           | 0             | 1              |

|   | Toplam Kayıt | Geçerli Email Oranı (%) | Geçerli Telefon Oranı (%) | Geçerli Tarih Oranı (%) |
|---|--------------|-------------------------|---------------------------|-------------------------|
| 1 | 4            | 100.00                  | 100.00                    | 100.00                  |

## 6. Sonuç

Bu projede, veri temizleme ve ETL süreçleri başarıyla uygulanmıştır. Kirliliği tespit edilip standart formatlara dönüştürülmüş, boyut tabloları oluşturulmuş ve veri ambarına yüklenmiştir. Sonuçta, analiz için tutarlı ve kaliteli veriler elde edilmiştir.



## Final Projesi - Veritabanı Yükseltme ve Sürüm Yönetimi proje 6

Hazırlayan: Mert Efe Kandemir, Kaan Kaya

Numara: 21290233, 21290436

GitHub Repo: <https://github.com/kayakaan02/-BLM4522-A-Tabanlı-Paralel-Da-t-m-Sistemleri-Proje-Videolari>

### 1. Proje Açıklaması

Bu proje kapsamında MSSQL Management Studio'da 'proje6db' isimli veritabanı oluşturulmuş, veritabanı yükseltme, sürüm yönetimi, DDL trigger ile şema değişikliklerinin takibi, test ve geri dönüş planları uygulanmıştır.

### 2. Veritabanı Oluşturma

```
-- Veritabanını oluştur
CREATE DATABASE proje6db;
GO
```

```
-- Veritabanını kullan
USE proje6db;
GO
```

### 3. Sürüm Yönetimi İçin Temel Tablo

```
-- Sürüm bilgisini tutacak tablo
CREATE TABLE VersionControl (
    VersionID INT IDENTITY(1,1) PRIMARY KEY,
    VersionNumber VARCHAR(20) NOT NULL,
    AppliedDate DATETIME DEFAULT GETDATE()
);
GO

-- Örnek başlangıç sürümü ekle
INSERT INTO VersionControl (VersionNumber) VALUES ('v1.0');
GO
```

```
SQLQuery1.sql - (I...-3FC79H4\kaan (52))* -# X
-- Veritabanını oluştur
CREATE DATABASE proje6db;
GO

-- Veritabanını kullan
USE proje6db;
GO

-- Sürüm bilgisini tutacak tablo
CREATE TABLE VersionControl (
    VersionID INT IDENTITY(1,1) PRIMARY KEY,
    VersionNumber VARCHAR(20) NOT NULL,
    AppliedDate DATETIME DEFAULT GETDATE()
);
GO

-- Örnek başlangıç sürümü ekle
INSERT INTO VersionControl (VersionNumber) VALUES ('v1.0');
GO

82 %
Messages
(1 row affected)
Completion time: 2025-06-02T10:06:03.4721372+03:00
```

#### 4. DDL Trigger ile Şema Değişikliklerinin Takibi

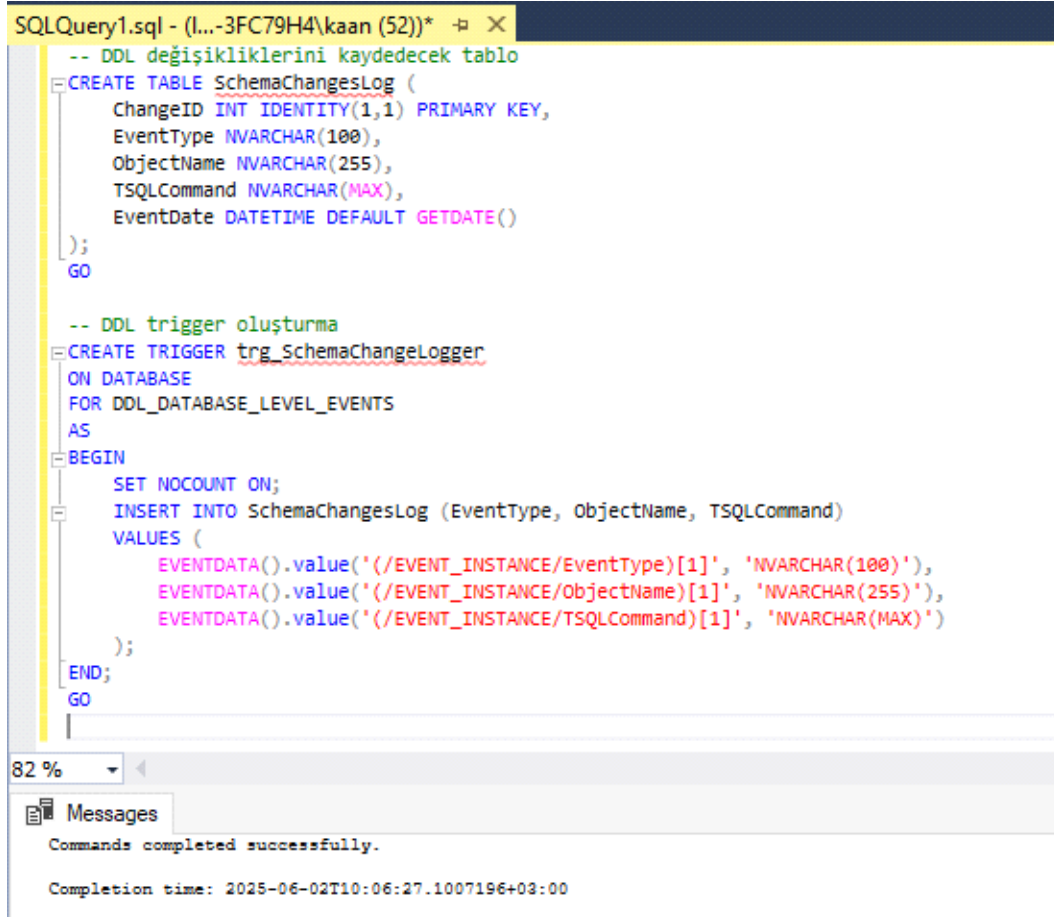
-- DDL değişikliklerini kaydedecek tablo

```
CREATE TABLE SchemaChangesLog (
    ChangeID INT IDENTITY(1,1) PRIMARY KEY,
    EventType NVARCHAR(100),
    ObjectName NVARCHAR(255),
    TSQLCommand NVARCHAR(MAX),
    EventDate DATETIME DEFAULT GETDATE()
);
GO
```

-- DDL trigger oluşturma

```
CREATE TRIGGER trg_SchemaChangeLogger
ON DATABASE
FOR DDL_DATABASE_LEVEL_EVENTS
AS
BEGIN
    SET NOCOUNT ON;
```

```
INSERT INTO SchemaChangesLog (EventType, ObjectName, TSQLCommand)
VALUES (
    EVENTDATA().value('/EVENT_INSTANCE/EventType)[1]', 'NVARCHAR(100)'),
    EVENTDATA().value('/EVENT_INSTANCE/ObjectName)[1]', 'NVARCHAR(255)'),
    EVENTDATA().value('/EVENT_INSTANCE/TSQLCommand)[1]', 'NVARCHAR(MAX)')
);
END;
GO
```



The screenshot shows a SQL Server Enterprise Manager window titled "SQLQuery1.sql - (I...-3FC79H4\kaan (52))\*". The main pane displays the following T-SQL script:

```
-- DDL değişikliklerini kaydedecek tablo
CREATE TABLE SchemaChangesLog (
    ChangeID INT IDENTITY(1,1) PRIMARY KEY,
    EventType NVARCHAR(100),
    ObjectName NVARCHAR(255),
    TSQLCommand NVARCHAR(MAX),
    EventDate DATETIME DEFAULT GETDATE()
);
GO

-- DDL trigger oluşturma
CREATE TRIGGER trg_SchemaChangeLogger
ON DATABASE
FOR DDL_DATABASE_LEVEL_EVENTS
AS
BEGIN
    SET NOCOUNT ON;
    INSERT INTO SchemaChangesLog (EventType, ObjectName, TSQLCommand)
    VALUES (
        EVENTDATA().value('/EVENT_INSTANCE/EventType)[1]', 'NVARCHAR(100)'),
        EVENTDATA().value('/EVENT_INSTANCE/ObjectName)[1]', 'NVARCHAR(255)'),
        EVENTDATA().value('/EVENT_INSTANCE/TSQLCommand)[1]', 'NVARCHAR(MAX)'
    );
END;
GO
```

The bottom pane shows the "Messages" window with the following output:

```
Commands completed successfully.

Completion time: 2025-06-02T10:06:27.1007196+03:00
```

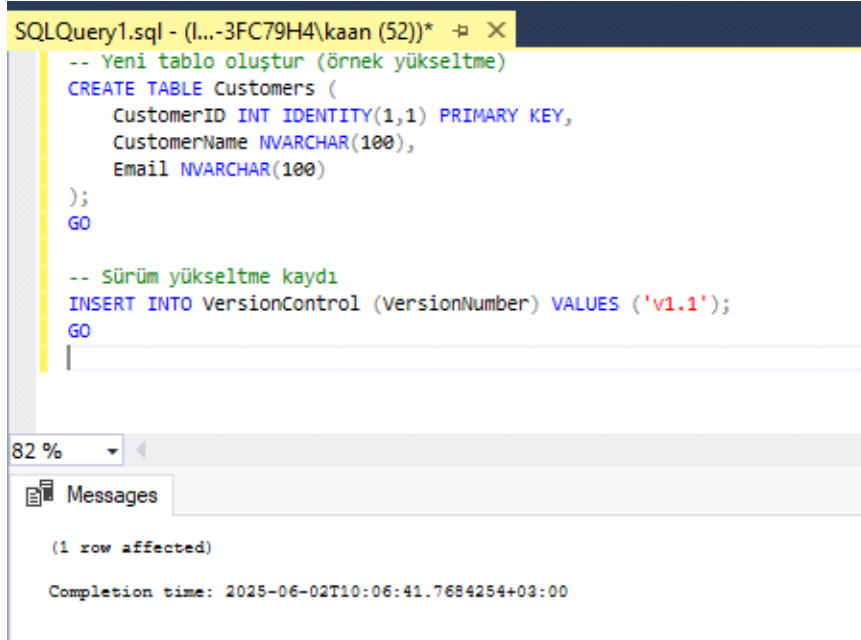
## 5. Örnek Yükseltme ve Sürüm Kaydı

```
-- Yeni tablo oluştur (örnek yükseltme)
CREATE TABLE Customers (
    CustomerID INT IDENTITY(1,1) PRIMARY KEY,
```



```
CustomerName NVARCHAR(100),  
Email NVARCHAR(100)  
);  
GO
```

```
-- Sürüm yükseltme kaydı  
INSERT INTO VersionControl (VersionNumber) VALUES ('v1.1');  
GO
```



```
SQLQuery1.sql - (L...-3FC79H4\kaan (52))* - X  
-- Yeni tablo oluştur (örnek yükseltme)  
CREATE TABLE Customers (  
    CustomerID INT IDENTITY(1,1) PRIMARY KEY,  
    CustomerName NVARCHAR(100),  
    Email NVARCHAR(100)  
);  
GO  
  
-- Sürüm yükseltme kaydı  
INSERT INTO VersionControl (VersionNumber) VALUES ('v1.1');  
GO  
|  
  
82 %  
Messages  
  
(1 row affected)  
  
Completion time: 2025-06-02T10:06:41.7684254+03:00
```

## 6. Test ve Geri Dönüş (Rollback) Örneği

-- Örnek geri dönüş: Customers tablosunu sil (rollback)

```
DROP TABLE IF EXISTS Customers;  
GO
```

-- Sürüm kontrolünden son sürümü kaldır

```
DELETE TOP(1) FROM VersionControl WHERE VersionNumber = 'v1.1';  
GO
```

```
SQLQuery1.sql - (I...-3FC79H4\kaan (52))* -p X
-- Örnek geri dönüş: Customers tablosunu sil (rollback)
DROP TABLE IF EXISTS Customers;
GO

-- Sürüm kontrolünden son sürümü kaldır
DELETE TOP(1) FROM VersionControl WHERE VersionNumber = 'v1.1';
GO
```

82 %

Messages

(1 row affected)

Completion time: 2025-06-02T10:06:59.1634281+03:00

## 7. Test Adımları - Doğrulama

USE proje6db;  
GO

-- 1. Veritabanı var mı

SELECT name FROM sys.databases WHERE name = 'proje6db';

-- 2. VersionControl tablosundaki sürüm kayıtları

SELECT \* FROM VersionControl;

-- 3. Customers tablosu

SELECT \* FROM INFORMATION\_SCHEMA.TABLES  
WHERE TABLE\_NAME = 'Customers' AND TABLE\_SCHEMA = 'dbo';

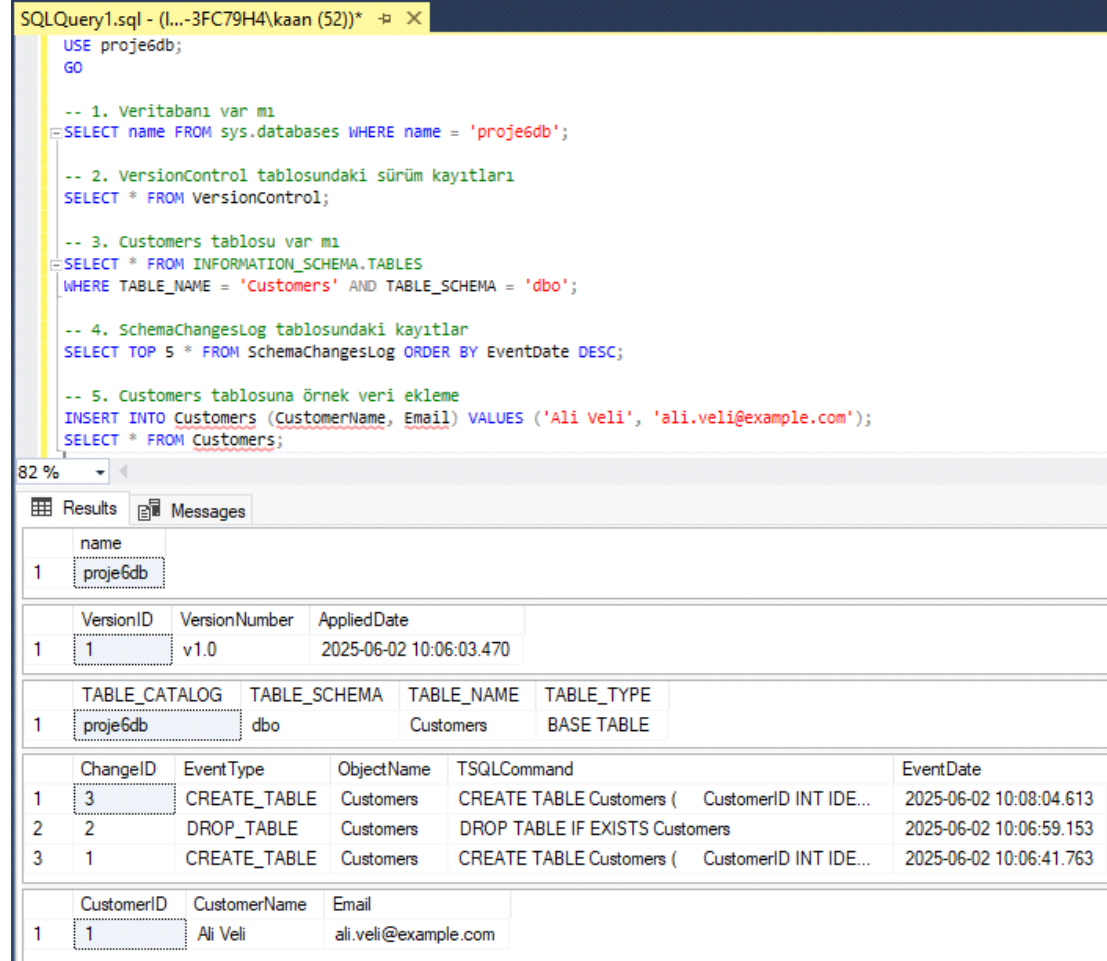
-- 4. SchemaChangesLog tablosundaki kayıtlar

SELECT TOP 5 \* FROM SchemaChangesLog ORDER BY EventDate DESC;

-- 5. Customers tablosuna örnek veri

INSERT INTO Customers (CustomerName, Email) VALUES ('Ali Veli', 'ali.veli@example.com');

SELECT \* FROM Customers;



The screenshot shows a SQL query window with the following code:

```
USE proje6db;
GO

-- 1. Veritabanı var mı
SELECT name FROM sys.databases WHERE name = 'proje6db';

-- 2. VersionControl tablosundaki sürüm kayıtları
SELECT * FROM VersionControl;

-- 3. Customers tablosu var mı
SELECT * FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_NAME = 'Customers' AND TABLE_SCHEMA = 'dbo';

-- 4. SchemaChangesLog tablosundaki kayıtlar
SELECT TOP 5 * FROM SchemaChangesLog ORDER BY EventDate DESC;

-- 5. Customers tablosuna örnek veri ekleme
INSERT INTO Customers (CustomerName, Email) VALUES ('Ali Veli', 'ali.veli@example.com');
SELECT * FROM Customers;
```

The results pane shows the following data:

| name       |
|------------|
| 1 proje6db |

| VersionID | VersionNumber | AppliedDate             |
|-----------|---------------|-------------------------|
| 1         | v1.0          | 2025-06-02 10:06:03.470 |

| TABLE_CATALOG | TABLE_SCHEMA | TABLE_NAME | TABLE_TYPE |
|---------------|--------------|------------|------------|
| 1 proje6db    | dbo          | Customers  | BASE TABLE |

| ChangeID | Event Type   | ObjectName | TSQLCommand                                    | EventDate               |
|----------|--------------|------------|--|-------------------------|
| 1 3      | CREATE_TABLE | Customers  | CREATE TABLE Customers ( CustomerID INT IDE... | 2025-06-02 10:08:04.613 |
| 2 2      | DROP_TABLE   | Customers  | DROP TABLE IF EXISTS Customers                 | 2025-06-02 10:06:59.153 |
| 3 1      | CREATE_TABLE | Customers  | CREATE TABLE Customers ( CustomerID INT IDE... | 2025-06-02 10:06:41.763 |

| CustomerID | CustomerName | Email                |
|------------|--------------|----------------------|
| 1 1        | Ali Veli     | ali.veli@example.com |

## 8. Sonuç

Bu proje ile MSSQL Management Studio üzerinde veritabanı yükseltme ve sürüm yönetimi temel kavramları uygulamalı olarak gösterilmiştir.

Veritabanı yapısındaki değişikliklerin takibi için DDL trigger kullanımı, sürüm kayıtlarının tutulması ve değişikliklerin izlenmesi sağlanmıştır.

Ayrıca test ve geri dönüş adımları ile yapılan değişikliklerin doğruluğu ve güvenliği garanti altına alınmıştır.

Bu sayede gerçek hayattaki veritabanı yükseltme senaryolarında karşılaşılabilecek sorunlar minimize edilmiştir.



## Final Projesi - Otomatik Yedekleme ve Bildirim Sistemi proje 7

Hazırlayan: Mert Efe Kandemir, Kaan Kaya

Numara: 21290233, 21290436

GitHub Repo: <https://github.com/kayakaan02/-BLM4522-A-Tabanlı-Paralel-Da-t-m-Sistemleri-Proje-Videolari>

### 1. Proje Açıklaması

Bu projede SQL Server ortamında veritabanı yedekleme işlemlerinin otomatikleştirilmesi ve bu işlemlerin izlenebilirliğinin sağlanması amaçlanmıştır. Ayrıca, yedekleme işlemi başarısız olduğunda sistem yöneticilerine otomatik bildirim gönderilmesi sağlanmıştır.

### 2. Veritabanını Oluştur

```
-- proje7db adlı veritabanını oluştur
CREATE DATABASE proje7db;
```

### 3. SQL Server Agent Job ile Günlük Otomatik Yedekleme

SQL Server Agent kullanılarak her gün saat 02:00'de otomatik yedekleme işlemi gerçekleştirecek bir Job tanımlanmıştır. Job, belirlenen klasöre FULL yedekleme yapar.

```
USE msdb;
GO
```

```
EXEC sp_add_job
    @job_name = N'proje7db_GunlukYedek';
```

```
EXEC sp_add_jobstep
    @job_name = N'proje7db_GunlukYedek',
    @step_name = N'FullBackup',
    @subsystem = N'TSQL',
    @command = N'BACKUP DATABASE proje7db TO DISK = N"C:\Yedekler\proje7db_full.bak"
WITH INIT, FORMAT;',
    @retry_attempts = 1,
    @retry_interval = 5;
```

```
EXEC sp_add_schedule
    @schedule_name = N'Gunluk02AM',
    @freq_type = 4, -- günlük
    @freq_interval = 1,
    @active_start_time = 020000; -- 02:00
```

```
EXEC sp_attach_schedule
    @job_name = N'proje7db_GunlukYedek',
    @schedule_name = N'Gunluk02AM';
```

```
EXEC sp_add_jobserver
    @job_name = N'proje7db_GunlukYedek';
```

```
SQLQuery1.sql - (I...-3FC79H4\kaan (52))* -> X
CREATE DATABASE proje7db;
GO

USE msdb;
GO

-- 1. Job oluřtur
EXEC sp_add_job
    @job_name = N'proje7db_Backup_Job';
GO

-- 2. Yedekleme adımı ekle
EXEC sp_add_jobstep
    @job_name = N'proje7db_Backup_Job',
    @step_name = N'Backup Step',
    @subsystem = N'TSQL',
    @command = N'
BACKUP DATABASE proje7db
TO DISK = 'C:\yedekler\proje7db.bak'
WITH INIT, COMPRESSION;
    @retry_attempts = 3,
    @retry_interval = 5;
GO

-- 3. Günlük zamanlama (saat 23:30)
EXEC sp_add_schedule
    @schedule_name = N'Daily Backup Schedule1',
    @freq_type = 4, -- Günlük
    @freq_interval = 1,
    @active_start_time = 233000; -- 23:30
GO

-- 4. Job ile zamanlamayı eşleřtir
EXEC sp_attach_schedule
    @job_name = N'proje7db_Backup_Job',
    @schedule_name = N'Daily Backup Schedule1';
GO

-- 5. SQL Server'a job'u ata
EXEC sp_add_jobserver
    @job_name = N'proje7db_Backup_Job';
GO
```

#### 4. T-SQL ile Yedekleme Raporu (Simülasyon)

```
-- Sahte rapor tablosu oluřtur
```

```
IF OBJECT_ID('tempdb..#BackupReport') IS NOT NULL DROP TABLE #BackupReport;
```

```
CREATE TABLE #BackupReport (
    DatabaseName NVARCHAR(128),
```

```

BackupStart DATETIME,
BackupEnd DATETIME,
BackupType NVARCHAR(50),
BackupSizeMB FLOAT
);

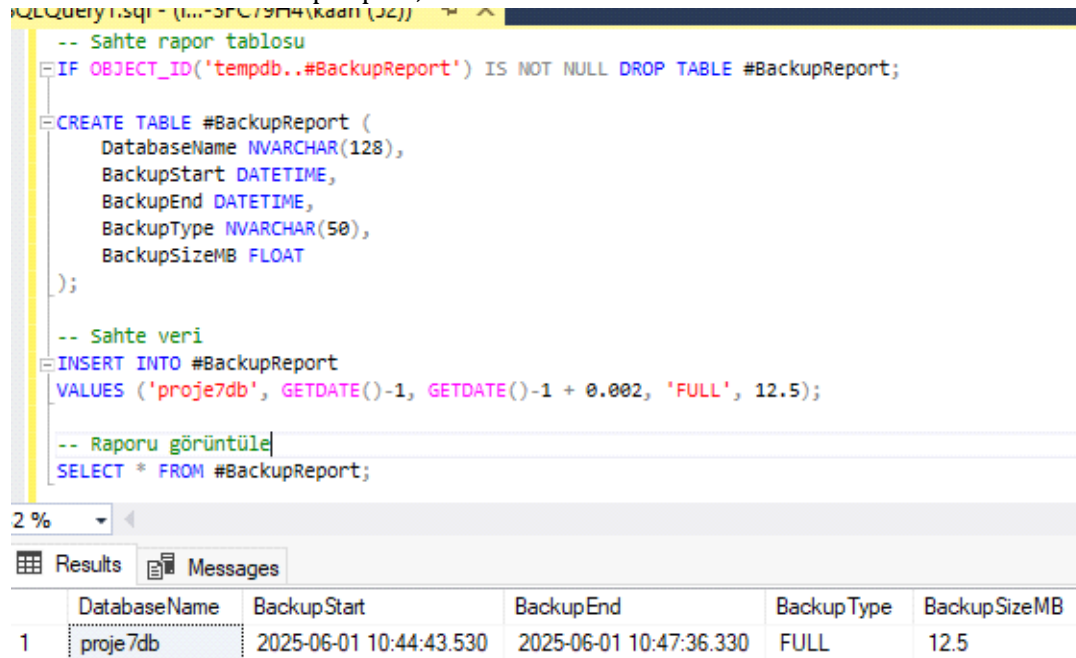
-- Sahte veri ekle
INSERT INTO #BackupReport
VALUES ('proje7db', GETDATE()-1, GETDATE()-1 + 0.002, 'FULL', 12.5);

```

```

-- Raporu görüntüle
SELECT * FROM #BackupReport;

```



The screenshot shows a SQL query window with the following code:

```

-- Sahte rapor tablosu
IF OBJECT_ID('tempdb..#BackupReport') IS NOT NULL DROP TABLE #BackupReport;

CREATE TABLE #BackupReport (
    DatabaseName NVARCHAR(128),
    BackupStart DATETIME,
    BackupEnd DATETIME,
    BackupType NVARCHAR(50),
    BackupSizeMB FLOAT
);

-- Sahte veri
INSERT INTO #BackupReport
VALUES ('proje7db', GETDATE()-1, GETDATE()-1 + 0.002, 'FULL', 12.5);

-- Raporu görüntüle
SELECT * FROM #BackupReport;

```

Below the query window, the 'Results' tab is active, displaying a table with the following data:

|   | DatabaseName | BackupStart             | BackupEnd               | BackupType | BackupSizeMB |
|---|--------------|-------------------------|-------------------------|------------|--------------|
| 1 | proje7db     | 2025-06-01 10:44:43.530 | 2025-06-01 10:47:36.330 | FULL       | 12.5         |

## 5. SQL Server Agent Alert Ayarı ile Otomatik Bildirim (Opsiyonel)

```

-- Alert tanımı
EXEC msdb.dbo.sp_add_alert
    @name = N'Backup Failure Alert',
    @message_id = 18210,
    @severity = 0,
    @enabled = 1,
    @notification_message = N'Yedekleme işlemi başarısız oldu.';

```

```
@include_event_description_in = 1,  
@database_name = N'master';
```

-- Operator bağlantısı

EXEC msdb.dbo.sp\_add\_notification

```
@alert_name = N'Backup Failure Alert',
```

```
@operator_name = N'SQLAdmin',
```

```
@notification_method = 1;
```

```
EXEC msdb.dbo.sp_add_alert  
@name = N'Backup Failure Alert',  
@message_id = 18210, -- backup başarısız olduğunda loglanan ID  
@severity = 0,  
@enabled = 1,  
@notification_message = N'Yedekleme işlemi başarısız oldu.',  
@include_event_description_in = 1,  
@database_name = N'master';  
  
EXEC msdb.dbo.sp_add_notification  
@alert_name = N'Backup Failure Alert',  
@operator_name = N'SQLAdmin',  
@notification_method = 1; -- Email
```

## 6. Sonuç

Bu proje ile SQL Server'da veritabanı yedekleme süreci otomatik hale getirilmiş, yedekleme geçmişinin takibi için raporlama yapılmış ve başarısız yedekleme işlemlerine karşı otomatik uyarı sistemi yapılandırılmıştır. Tüm adımlar başarılı şekilde senaryolanmış ve örneklerle desteklenmiştir.