

**ANKARA ÜNİVERSİTESİ**  
**MÜHENDİSLİK FAKÜLTESİ**  
**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**



**BLM4522 FİNAL ÖDEVİ**

**Ecem Şimşek - 21290553**

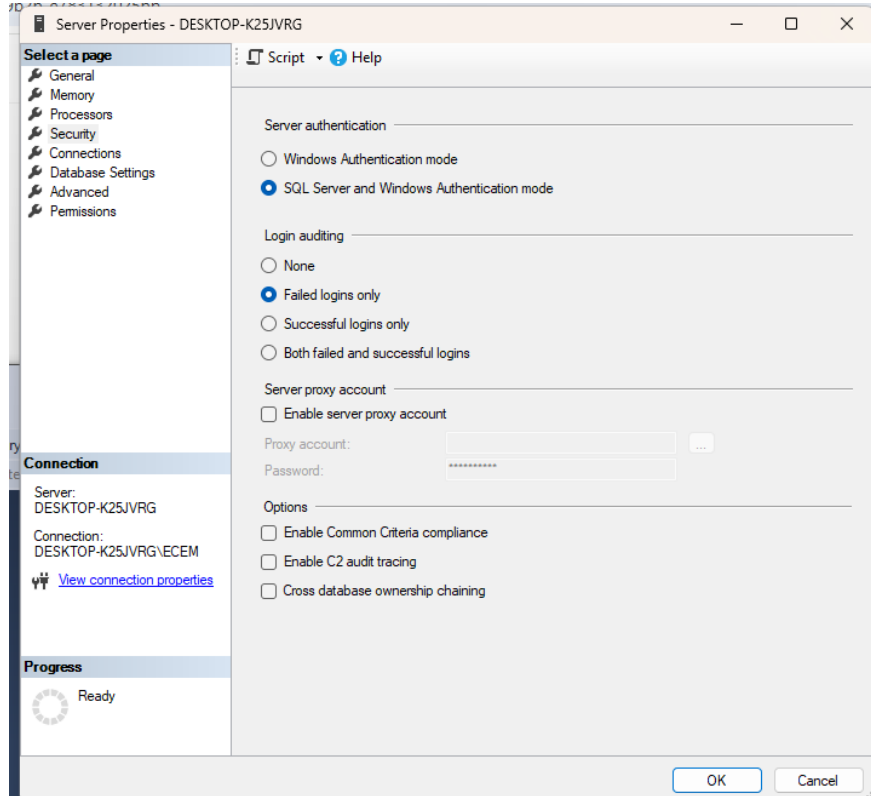
**Latife Süeda Tuğrul -20290297**

**Github : [https://github.com/suedatgrl/SQL\\_work](https://github.com/suedatgrl/SQL_work)**

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

## **Erişim Yönetimi**

1. Sol üstteki **Object Explorer** penceresinde sunucu adına **sağ tıklanır** → **Properties** seçilir.
2. Açılan pencerede soldaki menüden **Security** sekmesini seçilir.
3. **Server authentication** kısmında “**SQL Server and Windows Authentication mode**” seçeneğini işaretlenir.



4. Bu değişikliklerin geçerli olması için sunucuyu yenilenir.
5. SSMS'te Object Explorer'dan: **Security > Logins > sağ tıkla > New Login...** seçilir.
6. Açılan pencerede:
  - **Login name** kısmı doldurulur.
  - **SQL Server authentication** seçili olsun.
  - Güçlü bir şifre girilir.

Login - New

Select a page

- General
- Server Roles
- User Mapping
- Securables
- Status

Connection

Server: DESKTOP-K25JVRG

Connection: DESKTOP-K25JVRG\ECEM

[View connection properties](#)

Progress

Ready

Login name:  Search...

☐ Windows authentication  
☐ Microsoft Entra ID authentication  
☒ SQL Server authentication

Password:

Confirm password:

☐ Specify old password

Old password:

☒ Enforce password policy  
☒ Enforce password expiration  
☒ User must change password at next login

☐ Mapped to certificate  
☐ Mapped to asymmetric key  
☐ Map to Credential

Mapped Credentials

Credential	Pr...

Add Remove

Default database:

Default language:

OK Cancel

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

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

Login - New

Select a page

- General
- Server Roles
- User Mapping
- Securables
- Status

Connection

Server: DESKTOP-K25JVRG

Connection: DESKTOP-K25JVRG\ECEM

[View connection properties](#)

Progress

Ready

Users mapped to this login:

Map	Database	User	Default Schema
<input checked="" type="checkbox"/>	deneme	ecem_user	...
<input checked="" type="checkbox"/>	master	ecem_user	...
<input type="checkbox"/>	model		
<input type="checkbox"/>	msdb		
<input type="checkbox"/>	tempdb		
<input checked="" type="checkbox"/>	Test1	ecem_user	...
<input type="checkbox"/>	TestDB		

☒ Guest account enabled for: master

Database role membership for: master

<input type="checkbox"/>	db_accessadmin
<input type="checkbox"/>	db_backupoperator
<input checked="" type="checkbox"/>	db_datareader
<input checked="" type="checkbox"/>	db_datawriter
<input type="checkbox"/>	db_dtladmin
<input type="checkbox"/>	db_denydatareader
<input type="checkbox"/>	db_denydatawriter
<input type="checkbox"/>	db_owner
<input type="checkbox"/>	db_securityadmin
<input checked="" type="checkbox"/>	public

OK Cancel

8. Kullanıcı oluşturulmuş olacak.

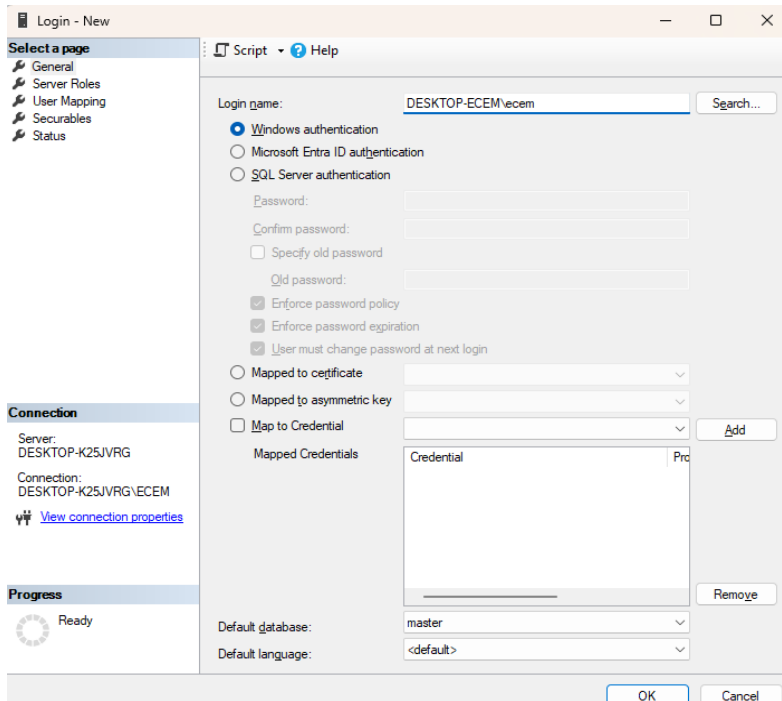
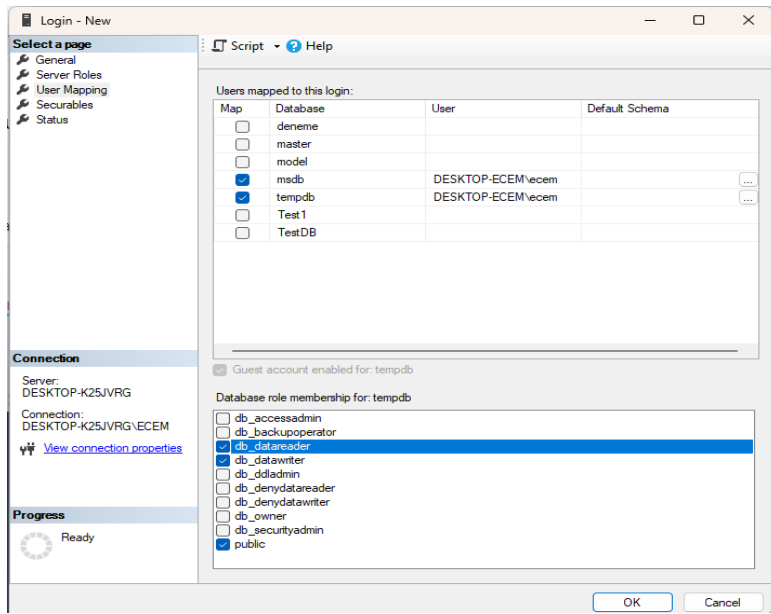
## SQL Server Authentication ve Windows Authentication

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

2. Açılan pencerede:

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

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



Giriş yapılan kullanıcıyı test etmek için şu kod çalıştırılır:

```
SQLQuery1.sql - lo...K25JVRG\ECCEM (54))* X
EXECUTE AS LOGIN = 'ecem_user';
SELECT SYSTEM_USER;
REVERT;
```

121 %

Results Messages

(No column name)
1 ecem_user

## Veri Şifreleme

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

-Yeni Sorgu penceresini açılır.

-Aşağıdaki komutla Master Key'i oluşturulur:

```
SQLQuery2.sql - lo...K25JVRG\ECCEM (53))* X SQLQuery1.sql - lo...K25JVRG\ECCEM (54))*
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'GüçlüBirParola123';
```

121 %

Messages

Commands completed successfully.

Completion time: 2025-04-24T21:25:50.3402237+03:00

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

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

```
SQLQuery5.sql - lo...K25JVRG\ECEM (62))* SQLQuery3.sql - lo...K25JVRG\ECEM (55))* SQLQuery2.sql - lo...K25JVRG\ECEM (54))*
USE TestDB;
GO

CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCPTION BY SERVER CERTIFICATE MyServerCert;
GO
```

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

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

```
SQLQuery7.sql - lo...K25JVRG\ECEM (52))* SQLQuery
ALTER DATABASE TestDB
SET ENCRYPTION ON;
GO
```

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

```
SQLQuery8.sql - lo...K25JVRG\ECEM (61))* SQLQuery7.sql - lo...K25JVRG\ECEM (52))*
SELECT db.name, db.is_encrypted
FROM sys.databases db
WHERE db.name = 'TestDB';
```

	name	is_encrypted
1	TestDB	1

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

```
SQLQuery8.sql - lo...K25JVRG\ECCEM (61)))*  SQLQuery7.sql - lo...K25JVRG\ECCEM (52)))*  SQL
BACKUP DATABASE [TestDB] TO DISK = 'C:\Backup\TestDB.bak';
```

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

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

## SQL Injection Testleri

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

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

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

3. Bu tabloya bazı test kullanıcıları eklenir.

```
SQLQuery9.sql - lo...K25JVRG\ECCEM (78)))*  SQLQuery8.sql - lo...K25JVRG\ECCEM (61)))*  SQLQuery7.sql - lo...K25JVRG\ECCEM
CREATE TABLE Users (
    UserID INT PRIMARY KEY IDENTITY,
    Username NVARCHAR(50),
    Password NVARCHAR(50)
);
GO
INSERT INTO Users (Username, Password) VALUES ('admin', 'admin123');
INSERT INTO Users (Username, Password) VALUES ('user1', 'password1');
INSERT INTO Users (Username, Password) VALUES ('guest', 'guest123');
GO

121 %
Messages

(1 row affected)

(1 row affected)

(1 row affected)

Completion time: 2025-04-24T22:10:34.9873919+03:00
```

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

```
SQLQuery10.sql - I...K25JVRG\ECCEM (60)))* X SQLQuery9.sql - lo...K25JVRG\ECCEM (78)))* SQLQuery8.sql - lo...K25JVRG\ECCEM (61)))*
-- SQL Injection'a açık sorgu
DECLARE @Username NVARCHAR(50);
DECLARE @Password NVARCHAR(50);

SET @Username = 'ecemsmk';
SET @Password = 'ecem2002';

EXEC('SELECT * FROM Users WHERE Username = ''' + @Username + ''' AND Password = ''' + @Password + ''')
GO
```

121 %

Results Messages

UserID	Username	Password
--------	----------	----------

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

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

```
SQLQuery11.sql - I...K25JVRG\ECCEM (56)))* X SQLQuery10.sql - I...K25JVRG\ECCEM (60)))* SQLQuery9.sql - lo...K25JVRG\ECCEM
SELECT * FROM Users WHERE Username = '' OR 1=1 --' AND Password = 'password';
```

121 %

Results Messages

	UserID	Username	Password
1	1	admin	admin123
2	2	user1	password1
3	3	guest	guest123

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

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

```
SQLQuery16.sql - I...K25JVRG\ECCEM (59)))* X SQLQuery15.sql - I...K25JVRG\ECCEM (70)))* SQLQuery14.sql - I...K25JVRG\ECCEM
USE master;
GO
CREATE SERVER AUDIT SPECIFICATION MyAuditSpec
FOR SERVER AUDIT MyAudit
ADD (SUCCESSFUL_LOGIN_GROUP), -- Başarılı girişler
ADD (FAILED_LOGIN_GROUP), -- Başarısız girişler
ADD (LOGOUT_GROUP), -- Çıkışlar
ADD (SQL_STATEMENT_COMPLETED_GROUP); -- SQL komutlarının tamamlanması
GO
```



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

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

```
EXEC CheckUserCredentials @Username = 'admin', @Password = 'admin123';
GO
```

UserID	Username	Password
1	admin	admin123

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

## Audit Logları

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

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

```
USE master;
GO

CREATE SERVER AUDIT MyAudit
TO FILE (FILEPATH = 'C:\SQLAuditLogs\'); -- Logların kaydedileceği dosya yolu
GO
```

Messages

Commands completed successfully.

Completion time: 2025-04-24T22:24:17.1264869+03:00

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

2. Audit nesnesini oluşturduktan sonra, onu başlatmamız gerekir.

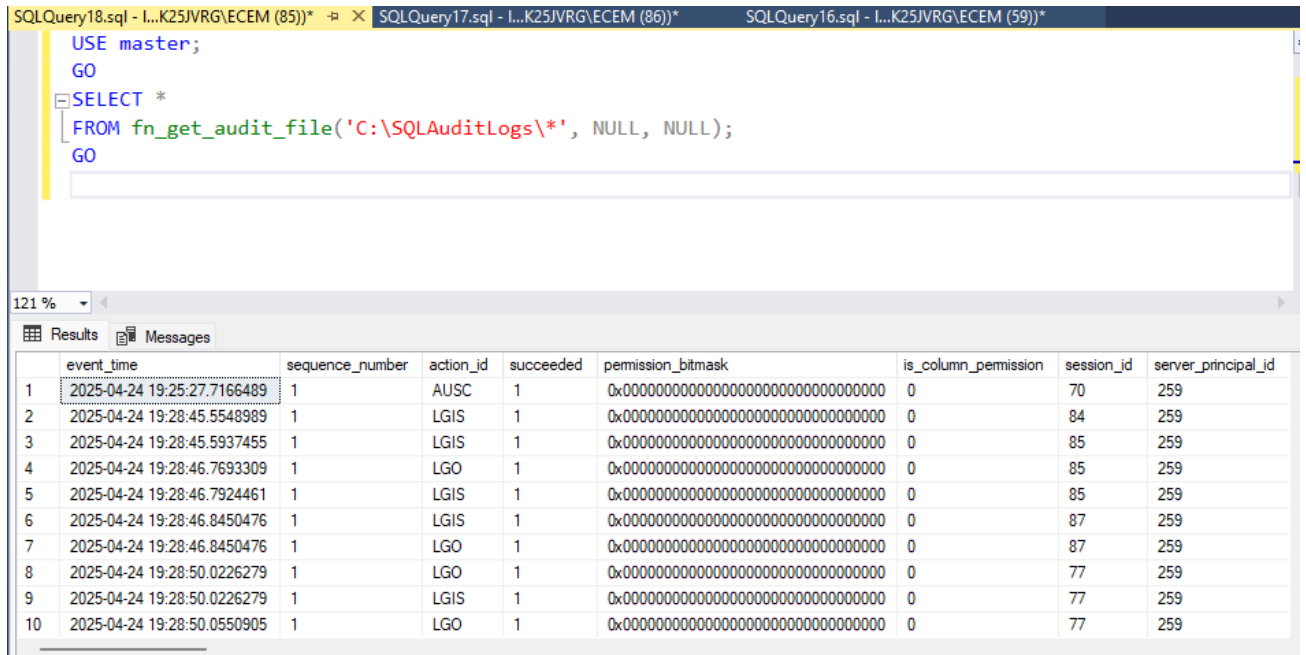
```
SQLQuery15.sql - I...K25JVRG\ECEM (70))*  SQLQuery14.sql - I...K25JVRG\ECEM (74))*  
  
USE master;  
GO  
ALTER SERVER AUDIT MyAudit  
WITH (STATE = ON);  
GO  
  
121 %  
Messages  
Commands completed successfully.  
Completion time: 2025-04-24T22:25:27.7344762+03:00
```

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

```
SQLQuery16.sql - I...K25JVRG\ECEM (59))*  SQLQuery15.sql - I...K25JVRG\ECEM (70))*  SQLQuery14.sql - I...K25JVRG\ECEM  
  
USE master;  
GO  
CREATE SERVER AUDIT SPECIFICATION MyAuditSpec  
FOR SERVER AUDIT MyAudit  
ADD (SUCCESSFUL_LOGIN_GROUP), -- Başarılı girişler  
ADD (FAILED_LOGIN_GROUP), -- Başarısız girişler  
ADD (LOGOUT_GROUP), -- Çıkışlar  
ADD (SQL_STATEMENT_COMPLETED_GROUP); -- SQL komutlarının tamamlanması  
GO  
  
21 %  
Messages  
Commands completed successfully.  
Completion time: 2025-04-24T22:28:28.0894991+03:00
```

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

5. Audit logları SQL Server tarafından `fn_get_audit_file` fonksiyonu ile sorgulanabilir. Bu fonksiyon, belirttiğiniz dosya yolundaki logları okuyabilir.



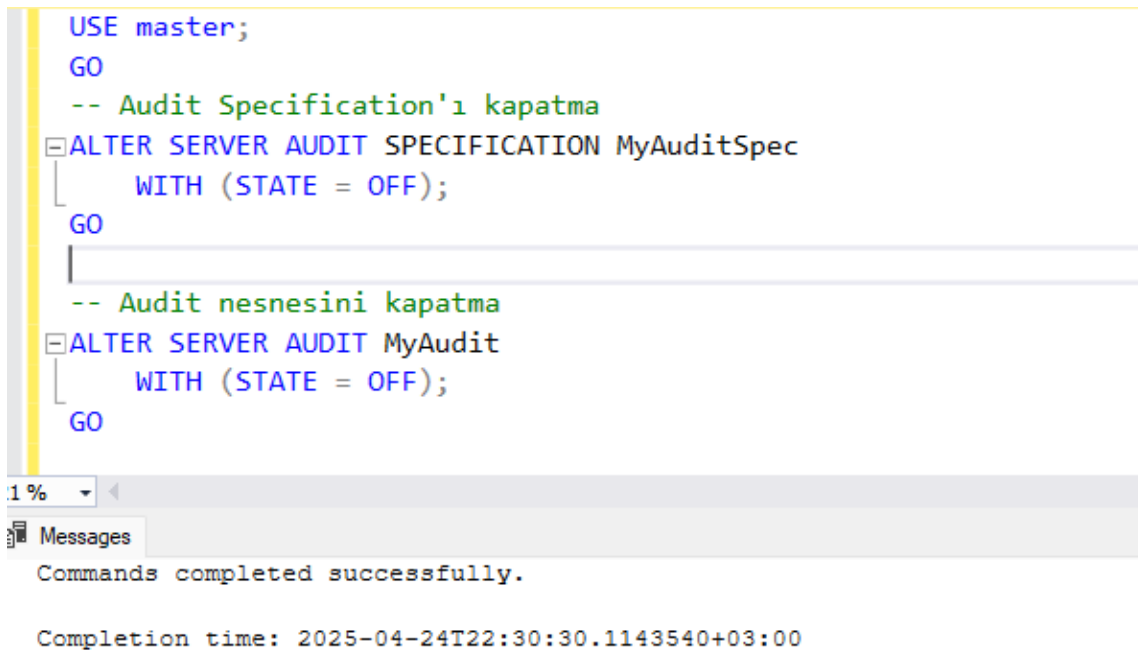
The screenshot shows a SQL Server Enterprise Manager interface. At the top, there are three tabs for SQL queries. The active query window contains the following T-SQL code:

```
USE master;
GO
SELECT *
FROM fn_get_audit_file('C:\SQLAuditLogs\*', NULL, NULL);
GO
```

Below the query window, the 'Results' tab is selected, displaying a table with 10 rows of audit log data. The columns are: event\_time, sequence\_number, action\_id, succeeded, permission\_bitmask, is\_column\_permission, session\_id, and server\_principal\_id.

	event_time	sequence_number	action_id	succeeded	permission_bitmask	is_column_permission	session_id	server_principal_id
1	2025-04-24 19:25:27.7166489	1	AUSC	1	0x00000000000000000000000000000000	0	70	259
2	2025-04-24 19:28:45.5548989	1	LGIS	1	0x00000000000000000000000000000000	0	84	259
3	2025-04-24 19:28:45.5937455	1	LGIS	1	0x00000000000000000000000000000000	0	85	259
4	2025-04-24 19:28:46.7693309	1	LGO	1	0x00000000000000000000000000000000	0	85	259
5	2025-04-24 19:28:46.7924461	1	LGIS	1	0x00000000000000000000000000000000	0	85	259
6	2025-04-24 19:28:46.8450476	1	LGIS	1	0x00000000000000000000000000000000	0	87	259
7	2025-04-24 19:28:46.8450476	1	LGO	1	0x00000000000000000000000000000000	0	87	259
8	2025-04-24 19:28:50.0226279	1	LGO	1	0x00000000000000000000000000000000	0	77	259
9	2025-04-24 19:28:50.0226279	1	LGIS	1	0x00000000000000000000000000000000	0	77	259
10	2025-04-24 19:28:50.0550905	1	LGO	1	0x00000000000000000000000000000000	0	77	259

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



The screenshot shows a SQL Server Enterprise Manager interface with a query window containing the following T-SQL code:

```
USE master;
GO
-- Audit Specification'ı kapatma
ALTER SERVER AUDIT SPECIFICATION MyAuditSpec
WITH (STATE = OFF);
GO
-- Audit nesnesini kapatma
ALTER SERVER AUDIT MyAudit
WITH (STATE = OFF);
GO
```

Below the query window, the 'Messages' tab is selected, displaying the following messages:

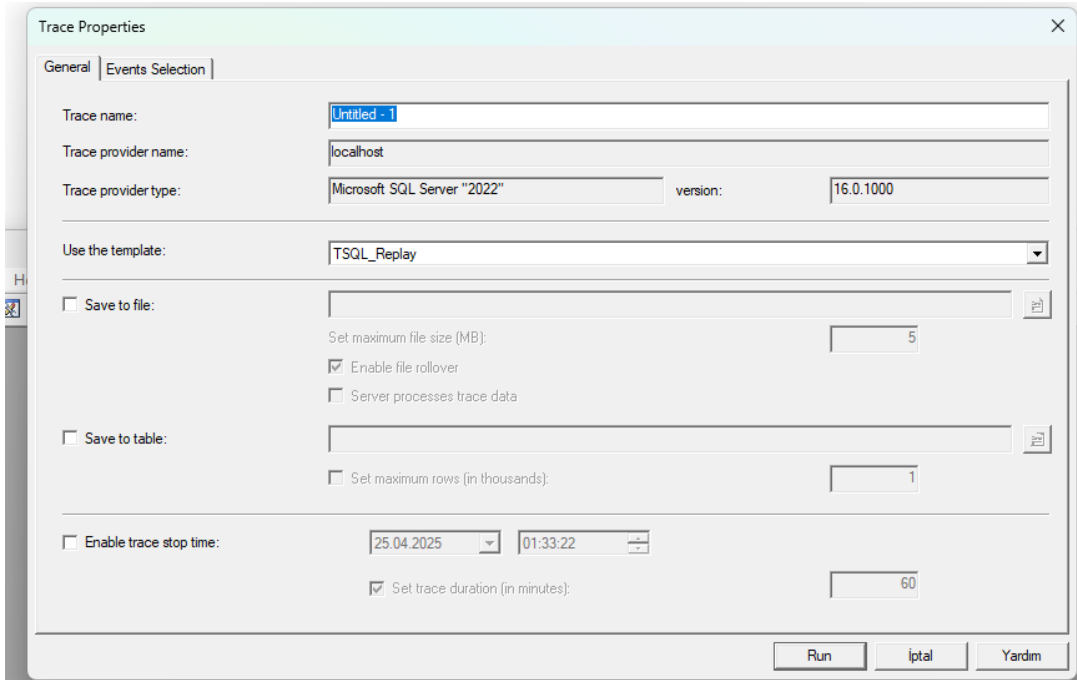
```
Commands completed successfully.

Completion time: 2025-04-24T22:30:30.1143540+03:00
```

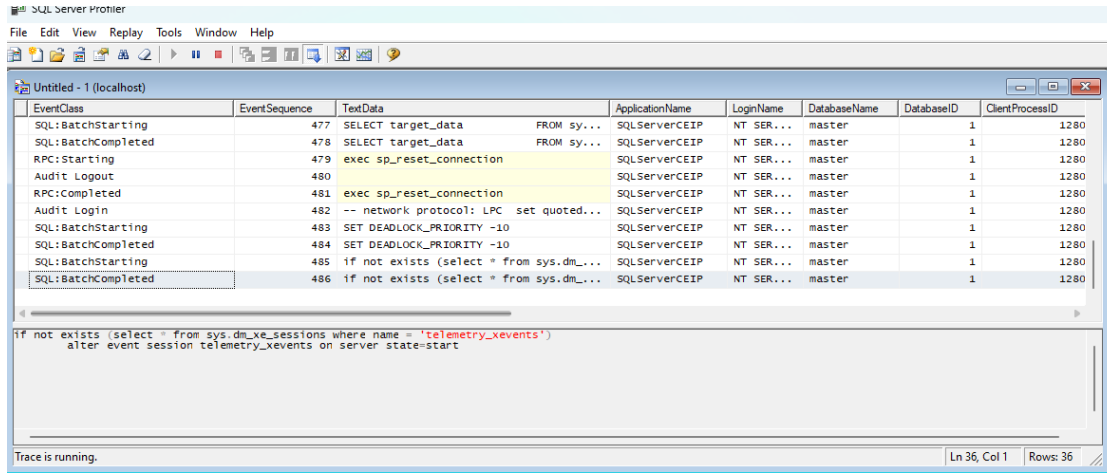
## 2.Proje- Veritabanı Performans Optimizasyonu ve İzleme

### Veritabanı İzleme

1. SQL Server Profiler açılır.
2. “File” → “New Trace...” seçilir.
3. SQL Server’a bağlanılır .
4. Bir **Trace Template** seçilir. (örnek: “TSQL\_Replay”).
5. “Events Selection” sekmesine gelerek aşağıdaki gibi olaylar eklenir:
  - SQL:BatchCompleted
  - RPC:Completed
  - Showplan XML



- Trace başlatılır.
- Ağır çalışan veya sürekli tekrarlanan sorguları kaydedilir.
- CPU time, Reads, Writes sütunlarına göre analiz edilebilir.
- Trace'i bitirip .trc dosyası olarak kaydedilebilir..



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

Bu sorgu, en çok zaman alan sorguları verir.

```

SELECT TOP 10
    qs.total_elapsed_time / qs.execution_count AS [AvgExecTime],
    qs.execution_count,
    qs.total_logical_reads,
    qs.total_logical_writes,
    st.text AS [SQLText]
FROM
    sys.dm_exec_query_stats qs
CROSS APPLY
    sys.dm_exec_sql_text(qs.sql_handle) st
ORDER BY
    [AvgExecTime] DESC;

```

	AvgExecTime	execution_count	total_logical_reads	total_logical_writes	SQLText
1	455695	1	198957	0	(@_msparam_0 nvarchar(4000),@_msparam_1 nvarchar...
2	2811	1	1068	0	(@_msparam_0 nvarchar(4000),@_msparam_1 nvarchar...
3	2006	1	93	0	(@_msparam_0 nvarchar(4000),@_msparam_1 nvarchar...
4	1547	1	37	0	(@_msparam_0 nvarchar(4000),@_msparam_1 nvarchar...
5	1407	1	44	0	SELECT dtb.name AS [Name], CAST(0 AS bit) AS [IsFab...
6	1198	5	265	2	(@_msparam_0 nvarchar(4000),@_msparam_1 nvarchar...
7	1119	1	59	0	(@_msparam_0 nvarchar(4000),@_msparam_1 nvarchar...
8	1070	1	8	0	(@_msparam_0 nvarchar(4000))SELECT dtb.collation_n...

## İndeks Yönetimi

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

```

SELECT
    t.name AS Table_Name,
    i.name AS Index_Name,
    i.type_desc AS Index_Type,
    i.is_primary_key AS Is_Primary_Key,
    i.is_unique AS Is_Unique,
    i.fill_factor AS Fill_Factor,
    i.is_disabled AS Is_Disabled,
    i.is_hypothetical AS Is_Hypothetical,
    i.create_date AS Create_Date,
    i.modify_date AS Modify_Date
FROM
    sys.indexes AS i
INNER JOIN
    sys.tables AS t ON i.object_id = t.object_id
WHERE
    t.is_ms_shipped = 0 AND i.type_desc <> 'HEAP' -- 'HEAP' tipi indeks değildir
ORDER BY
    t.name, i.name;

```

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

```
SELECT
    migs.avg_total_user_cost * migs.avg_user_impact * (migs.user_seeks + migs.user_scans) AS [Impact],
    mid.statement AS [TableName],
    mid.equality_columns,
    mid.inequality_columns,
    mid.included_columns
FROM
    sys.dm_db_missing_index_group_stats migs
JOIN
    sys.dm_db_missing_index_groups mig ON migs.group_handle = mig.index_group_handle
JOIN
    sys.dm_db_missing_index_details mid ON mig.index_handle = mid.index_handle
ORDER BY
    [Impact] DESC;
```

21 %

Results Messages

Impact | TableName | equality\_columns | inequality\_columns | included\_columns

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

```
SELECT
    OBJECT_NAME(ix.object_id) AS TableName,
    ix.name AS IndexName,
    ix.type_desc AS IndexType,
    ix.is_primary_key AS IsPrimaryKey,
    ix.is_unique AS IsUnique,
    ix.user_seeks, -- Kaç kez okuma işlemi yapılmış
    ix.user_scans, -- Kaç kez okuma işlemi tarama ile yapılmış
    ix.user_lookups, -- Kaç kez arama yapılmış
    ix.user_updates -- Kaç kez indeks güncellenmiş
FROM
    sys.indexes ix
WHERE
    OBJECTPROPERTY(ix.object_id, 'IsUserTable') = 1
    AND ix.type_desc IN ('CLUSTERED', 'NONCLUSTERED')
ORDER BY
    ix.user_seeks DESC; -- Kullanılma sıklığına göre sıralama
```

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

İndeks oluşturma için aşağıdaki komutu kullanılabilir:

```
SQLQuery25.sql - I...K25JVRG\ECM (58))* SQLQuery24.sql - I...K25JVRG\ECM (67))*
CREATE NONCLUSTERED INDEX IX_TableName_ColumnName
ON TableName (ColumnName);
```

## Sorgu İyileştirme

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

```
SQLQuery26.sql - I...25JVRG\ECCEM (104))* X SQLQuery25.sql - I...K25JVRG\ECCEM (5
SET SHOWPLAN_XML ON;
-- Uzun süren sorguyu burada çalıştırılır
SELECT * FROM large_table;
SET SHOWPLAN_XML OFF;
```

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

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

Eğer LEFT JOIN gereksiz yere kullanılıyorsa, sorgu süresi artabilir.

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

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

```
SQLQuery27.sql - I...K25JVRG\ECCEM (96))* X SQLQuery26.
SET STATISTICS TIME ON;
-- Sorguyu çalıştırılır
SELECT * FROM large_table;
SET STATISTICS TIME OFF;
```

## Veri Yöneticisi Roller

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

```
USE [TestDB];
GO
GRANT SELECT, INSERT, UPDATE, DELETE ON [Users] TO [VeriAnalisti];
GO
```

121 %

Messages

Commands completed successfully.

Completion time: 2025-04-25T01:17:03.6587580+03:00

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

```
USE [TestDB];
GO
EXEC sp_addrolemember 'VeriAnalisti', 'ecem'; -- Kullanıcıyı role atama
GO
```

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

```
USE [TestDB];
GO
EXEC sp_droprolemember 'VeriAnalisti', 'ecem'; -- Kullanıcıyı rolden çıkarma
GO
```

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

```
USE [TestDB];
GO
SELECT dp.name AS UserName,
       dp.type_desc AS UserType,
       o.name AS RoleName
FROM sys.database_principals dp
JOIN sys.database_role_members drm ON dp.principal_id = drm.member_principal_id
JOIN sys.database_principals o ON drm.role_principal_id = o.principal_id
WHERE dp.type NOT IN ('A', 'G', 'R', 'X')
AND dp.name = 'ecem';
GO
```



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

#### 1. Ortam Hazırlığı:

SQL Server Developer Edition ve SSMS yüklendi.

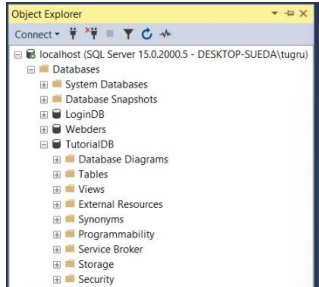
SSMS ile localhost (veya .\SQLEXPRESS) üzerinden "Database Engine"e Windows/sa ile bağlanıldı.

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

```
mkdir C:\SQLBackups  
copy "%USERPROFILE%\Downloads\TutorialDB.bak" "C:\SQLBackups\TutorialDB.bak"
```

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

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



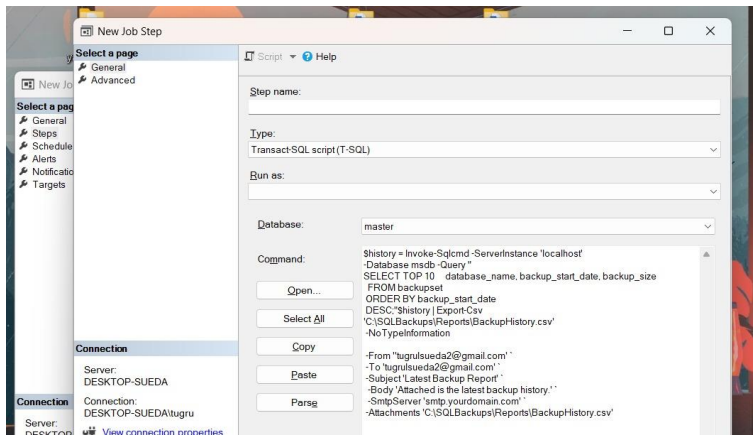
#### 3. Backup Job Oluşturma

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

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

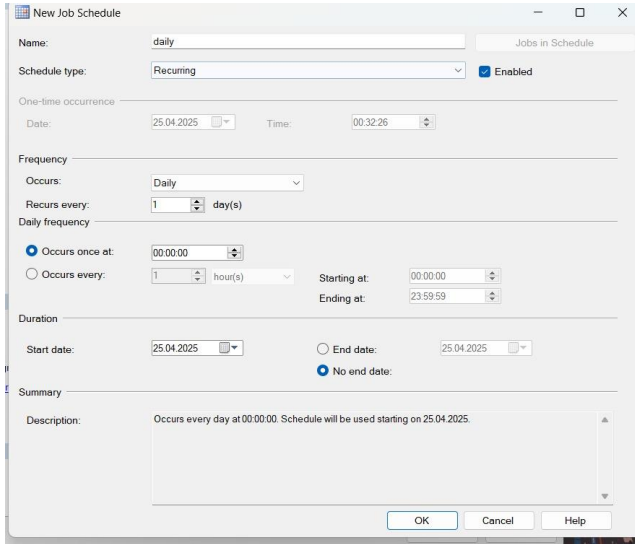
T-SQL kodu:

```
DECLARE @bak NVARCHAR(260) =  
N'C:\SQLBackups\TutorialDB_Full_' +  
CONVERT(CHAR(8), GETDATE(), 112) + '.bak';  
  
BACKUP DATABASE [TutorialDB]  
TO DISK = @bak  
WITH INIT, NAME = 'Full backup of TutorialDB ' + CONVERT(VARCHAR, GETDATE(), 120);
```



#### 4. Zamanlama (Schedule)

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

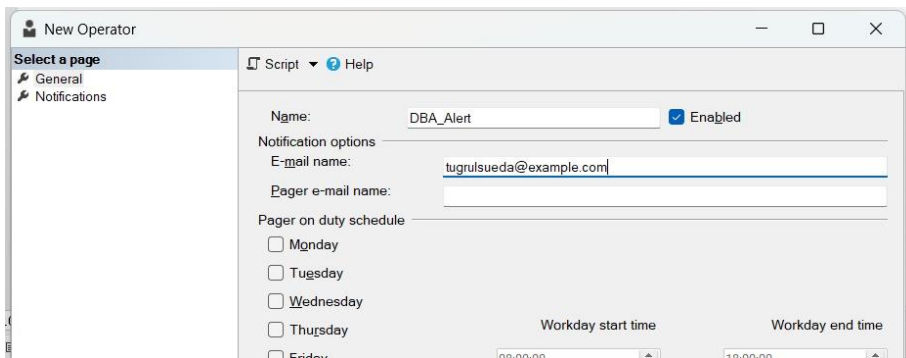


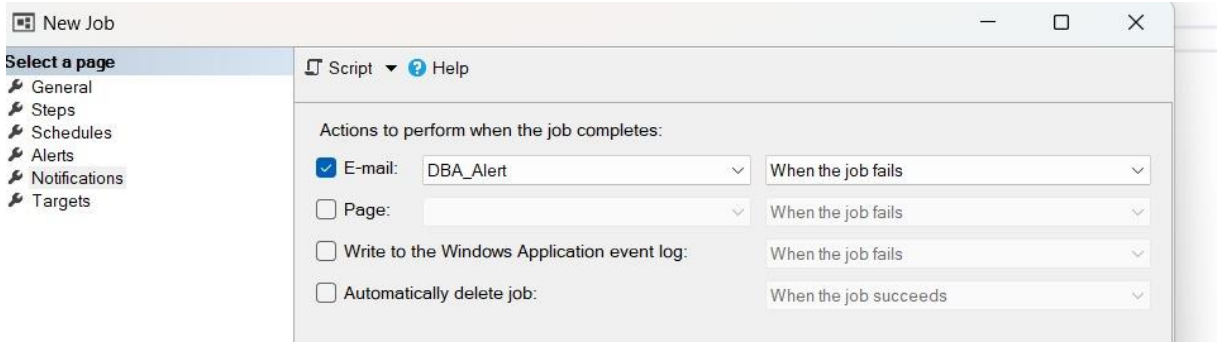
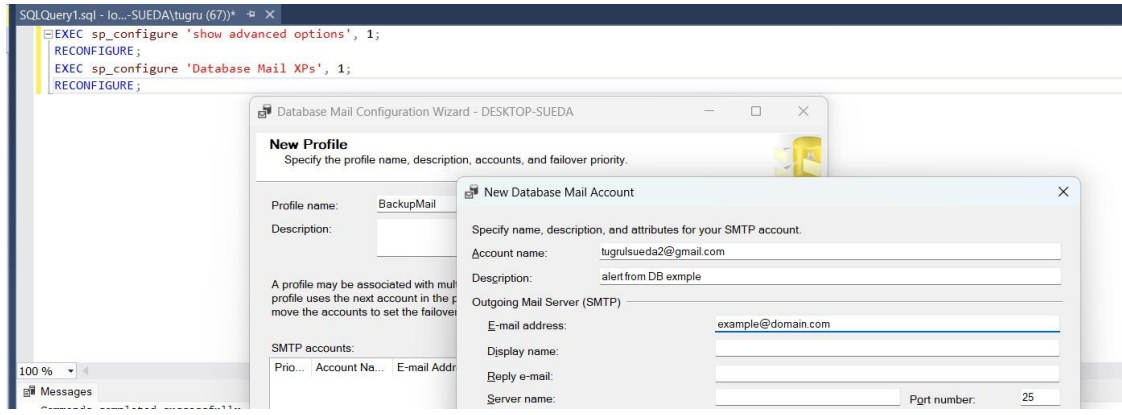
#### 5. E-posta Uyarıları

Management → Database Mail ile BackupMail profili oluşturuldu.

SQL Server Agent Properties'ten bu profil etkinleştirildi.

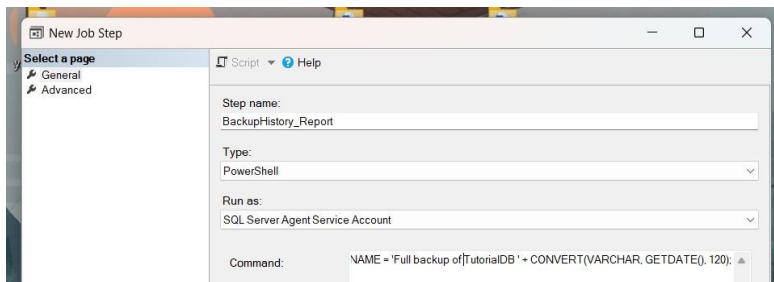
Operators altında DBA\_Alert tanımlanıp, Job'un Notifications → If job fails → E-mail kısmına atandı.





## 6. Raporlama

Yeni bir PowerShell tabanlı job (BackupHistory\_Report) eklenip, msdb.backupset'ten son yedek bilgileri CSV'ye yazdırıldı, Oluşan rapor ilgili adrese e-posta ile gönderildi.



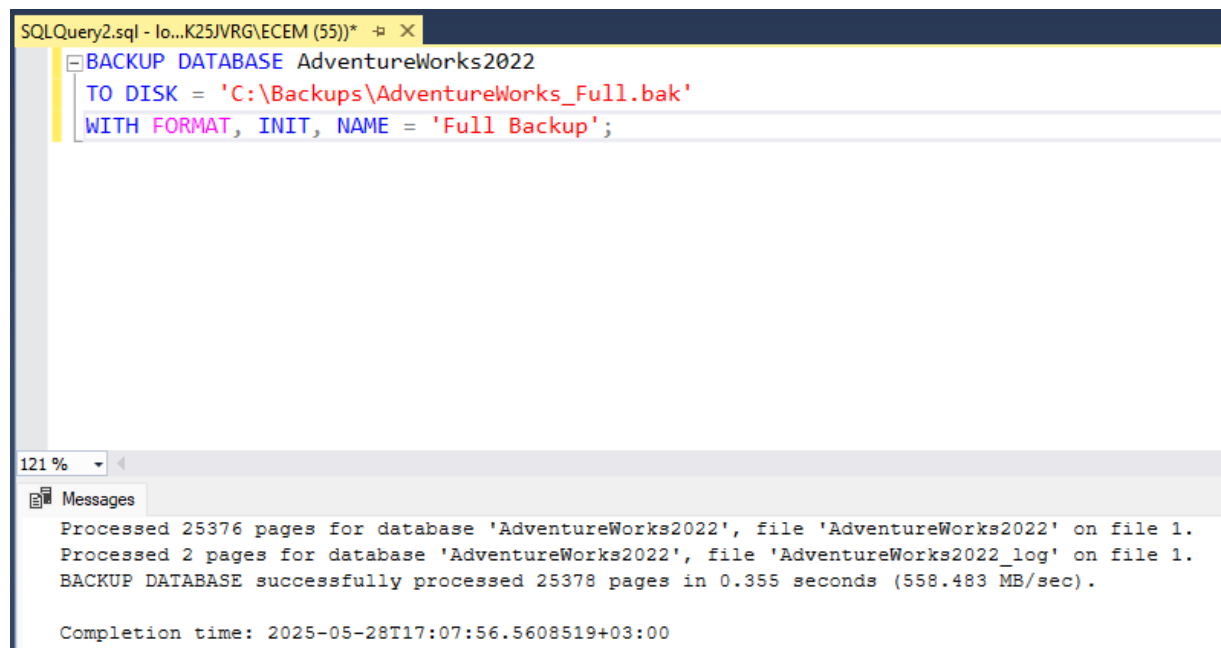
```
$history = Invoke-Sqlcmd -ServerInstance 'localhost' -Database msdb -Query "
SELECT TOP 10
    database_name, backup_start_date, backup_size
FROM backupset
ORDER BY backup_start_date DESC;"
$history | Export-Csv 'C:\SQLBackups\Reports\BackupHistory.csv' -NoTypeInformation
```

```
Send-MailMessage -From 'sqladmin@yourdomain.com' `
-To 'tugrulsueda1@gmail.com' `
-Subject 'Latest Backup Report' `
-Body 'Attached is the latest backup history.' `
-SmtpServer 'smtp.yourdomain.com' `
-Attachments 'C:\SQLBackups\Reports\BackupHistory.csv'
```

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

- Öncellikle internetten bir veritabanı kurulumu yapıldı.

### **1.A. Tam Yedekleme (Full Backup)**



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the SQL query for a full backup of the AdventureWorks2022 database. The bottom pane shows the execution results, indicating that the backup was successful.

```
SQLQuery2.sql - Io...K25\VRG\ECEM (55))* X
BACKUP DATABASE AdventureWorks2022
TO DISK = 'C:\Backups\AdventureWorks_Full.bak'
WITH FORMAT, INIT, NAME = 'Full Backup';
```

121 %

Messages

Processed 25376 pages for database 'AdventureWorks2022', file 'AdventureWorks2022' on file 1.  
Processed 2 pages for database 'AdventureWorks2022', file 'AdventureWorks2022\_log' on file 1.  
BACKUP DATABASE successfully processed 25378 pages in 0.355 seconds (558.483 MB/sec).

Completion time: 2025-05-28T17:07:56.5608519+03:00

### **1.B. Fark Yedekleme (Differential Backup)**

```
SQLQuery2.sql - Io...K25JVRG\ECCEM (55)) *  X
BACKUP DATABASE AdventureWorks2022
TO DISK = 'C:\Backups\AdventureWorks_Diff.bak'
WITH DIFFERENTIAL, NAME = 'Differential Backup';

121 %
Messages
Processed 64 pages for database 'AdventureWorks2022', file 'AdventureWorks2022' on file 1.
Processed 2 pages for database 'AdventureWorks2022', file 'AdventureWorks2022_log' on file 1.
BACKUP DATABASE WITH DIFFERENTIAL successfully processed 66 pages in 0.014 seconds (36.551 MB/sec).

Completion time: 2025-05-28T17:13:32.3536305+03:00
```

### 1.C. Artık Yedekleme (Transaction Log Backup)

```
SQLQuery2.sql - Io...K25JVRG\ECCEM (55)) *  X
BACKUP LOG AdventureWorks2022
TO DISK = 'C:\Backups\AdventureWorks_Log.trn'
WITH NAME = 'Log Backup';

121 %
Messages
Processed 25376 pages for database 'AdventureWorks2022', file 'AdventureWorks2022' on file 1.
Processed 2 pages for database 'AdventureWorks2022', file 'AdventureWorks2022_log' on file 1.
BACKUP DATABASE successfully processed 25378 pages in 0.351 seconds (564.848 MB/sec).
Processed 3 pages for database 'AdventureWorks2022', file 'AdventureWorks2022_log' on file 1.
BACKUP LOG successfully processed 3 pages in 0.004 seconds (5.859 MB/sec).

Completion time: 2025-05-28T17:15:59.1485718+03:00
```

### 1.D. Point-in-Time Restore (Belirli zamana geri döndürme)

- **Transaction Log backup** gerekiyor.
- RESTORE DATABASE ve RESTORE LOG komutları ile birlikte STOPAT parametresi kullanılır. Örnek:

```
RESTORE DATABASE AdventureWorks2022 FROM DISK =
'C:\Backups\AdventureWorks2022_Full.bak' WITH NORECOVERY;
```

```
RESTORE LOG AdventureWorks2022 FROM DISK =
'C:\Backups\AdventureWorks2022_Log.trn'
WITH STOPAT = '2025-05-28 14:30:00', RECOVERY;
```

### 1.E. Database Mirroring

- Database Mirroring için **en az iki SQL Server Instance** gerekir.
- Veritabanı **Full Recovery Moduna** alınır.

```
ALTER DATABASE AdventureWorks SET RECOVERY FULL;  
GO
```

- Tam Yedek ve Log Yedeklerini alınır.

```
-- Full backup (Tam yedek)  
BACKUP DATABASE AdventureWorks TO DISK =  
'C:\Backups\AdventureWorks.bak';  
GO
```

```
-- Transaction Log backup (Log yedeği)  
BACKUP LOG AdventureWorks TO DISK =  
'C:\Backups\AdventureWorks_Log.trn';  
GO
```

- Veritabanını Mirroring için hazırlanır.

```
RESTORE DATABASE AdventureWorks FROM DISK =  
'C:\Backups\AdventureWorks.bak' WITH NORECOVERY;  
GO
```

```
RESTORE LOG AdventureWorks FROM DISK =  
'C:\Backups\AdventureWorks_Log.trn' WITH NORECOVERY;  
GO
```

- Mirroring başlatılır.

## 2. Otomatik Yedekleme Zamanlayıcıları

- SQL Server Agent çalıştırılır.
- SQL Server Agent > Jobs > → New Job...
- Steps sekmesine tıklanır.New butonuna tıklanılır.
- Step name,database,type seçilir.Command alanına yedekleme kodu yazılır.Örneğin:

```
BACKUP DATABASE AdventureWorks2019  
TO DISK = 'C:\Backups\AdventureWorks_Scheduled.bak'  
WITH INIT, NAME = 'Scheduled Full Backup';
```

- Schedules sekmesinden new butonuna basılır.
- Name,Schedule type, frequency ve occurs once at seçenekleri seçilir.
- Zamanlama kaydedilir.

**New Job Schedule**

Name:  Jobs in Schedule

Schedule type:  ☒ Enabled

---

One-time occurrence

Date:  Time:

---

Frequency

Occurs:

Recurs every:  day(s)

---

Daily frequency

☒ Occurs once at:

☐ Occurs every:  hour(s)

Starting at:  Ending at:

---

Duration

Start date:  ☐ End date:

☒ No end date:

---

Summary

Description:

### 3. Felaketten Kurtarma Senaryoları

- Yanlışlıkla tablo silme

```
SQLQuery1.sql - Io...K25JVRG\ECM (57))* X
DROP TABLE Person.Person;
```

121 %

Messages

Commands completed successfully.

Completion time: 2025-05-28T17:41:37.4403198+03:00

- Kurtarma

```
SQLQuery1.sql - Io...K25JVRG\ECM (57))* X
USE master;
GO

RESTORE DATABASE AdventureWorks2022
FROM DISK = 'C:\Backups\AdventureWorks_Full.bak'
WITH REPLACE;
```

121 %

Messages

Processed 25376 pages for database 'AdventureWorks2022', file 'AdventureWorks2022' on file 1.  
Processed 2 pages for database 'AdventureWorks2022', file 'AdventureWorks2022\_log' on file 1.  
RESTORE DATABASE successfully processed 25378 pages in 0.457 seconds (433.833 MB/sec).

Completion time: 2025-05-28T17:48:25.3165111+03:00

### 4. Test Yedekleme Senaryosu

### Yeni bir veritabanı olarak test restore:

```
SQLQuery2.sql - lo...K25JVRG\ECEM (54))*
```

```
USE master;
GO

RESTORE DATABASE AdventureWorks_Test
FROM DISK = 'C:\Backups\AdventureWorks_Full.bak'
WITH MOVE 'AdventureWorks2022' TO 'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\AdventureWorks'
MOVE 'AdventureWorks2022_log' TO 'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\AdventureWorks_log'
REPLACE;
```

121 %

Messages

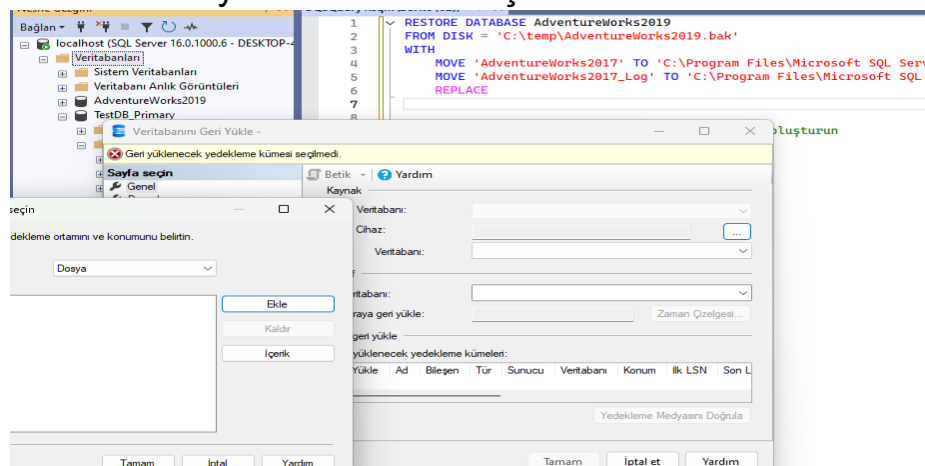
Processed 25376 pages for database 'AdventureWorks\_Test', file 'AdventureWorks2022' on file 1.  
Processed 2 pages for database 'AdventureWorks\_Test', file 'AdventureWorks2022\_log' on file 1.  
RESTORE DATABASE successfully processed 25378 pages in 0.465 seconds (426.369 MB/sec).

Completion time: 2025-05-28T17:57:14.7172421+03:00

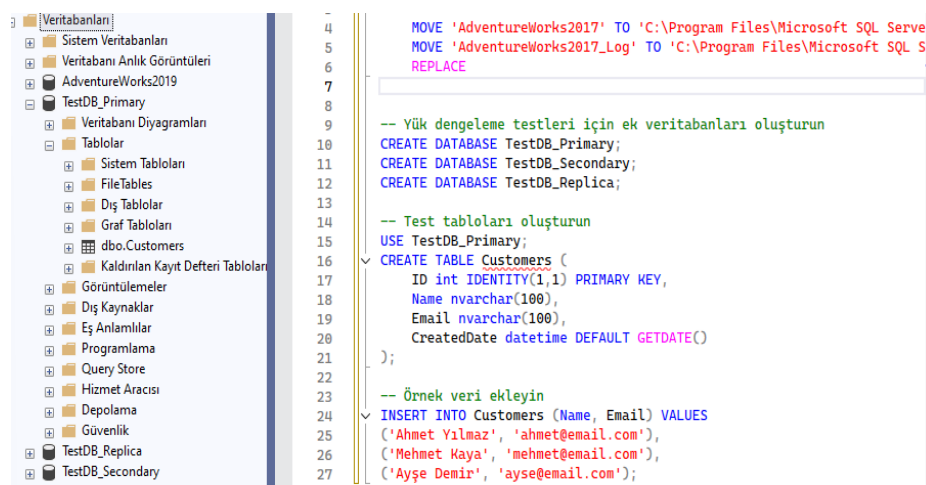
Böylece yedeklerin bozulmadığı test edilir.

## **5.Proje - Veritabanı Yük Dengeleme ve Dağıtık Veritabanı Yapıları**

Bu kısım db yi restore etmek için:

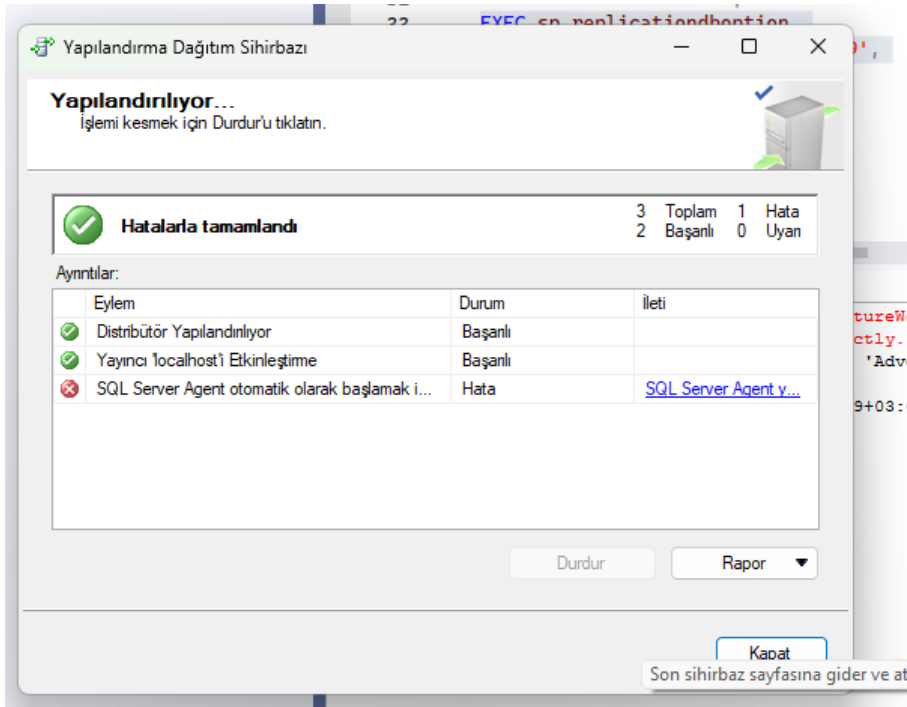


Bu kısım test veritabanı oluşturmak için:

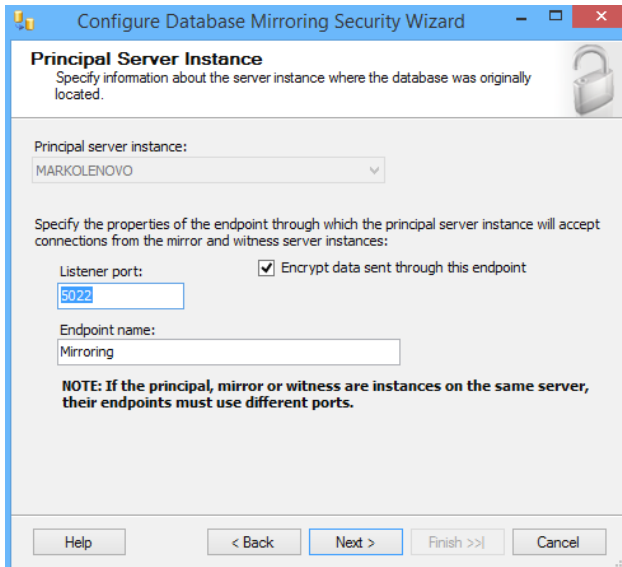


configure distribution:

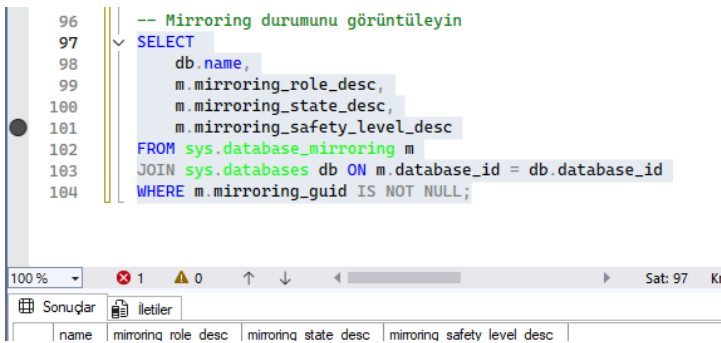




Mirroring:



buradaki mirror durumunu görüntüledik:



failover:

```

-- Failover testi
-- 1. Primary'ye veri insert
USE TestDB_Primary;
INSERT INTO Customers (Name, Email) VALUES ('Test User', 'test@ema:

-- 2. Failover
ALTER AVAILABILITY GROUP AG_TestDB FAILOVER;

-- 3. Yeni primary'de veriyi kontrol
SELECT * FROM Customers ORDER BY ID DESC;

-- Publisher'da veri ekleyin
USE AdventureWorks2019;
INSERT INTO Person.Person (PersonType, FirstName, LastName)
VALUES ('EM', 'Te
şema AdventureWorks2019.Person

-- Subscriber'da kontrol edin
USE AdventureWorks_Replica;
SELECT TOP 5 * FROM Person.Person ORDER BY ModifiedDate DESC;

```

Tüm kod:

```

RESTORE DATABASE AdventureWorks2019
FROM DISK = 'C:\temp\AdventureWorks2019.bak'
WITH
    MOVE 'AdventureWorks2017' TO 'C:\Program Files\Microsoft SQL
Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\AdventureWorks2019.mdf',
    MOVE 'AdventureWorks2017_Log' TO 'C:\Program Files\Microsoft SQL
Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\AdventureWorks2019.ldf',
    REPLACE

```

```

-- Yük dengeleme testleri için ek veritabanları oluşturun
CREATE DATABASE TestDB_Primary;
CREATE DATABASE TestDB_Secondary;
CREATE DATABASE TestDB_Replica;

```

```

-- Test tabloları oluşturun
USE TestDB_Primary;
CREATE TABLE Customers (
    ID int IDENTITY(1,1) PRIMARY KEY,
    Name nvarchar(100),
    Email nvarchar(100),
    CreatedDate datetime DEFAULT GETDATE()
);

```

```

-- Örnek veri ekleyin
INSERT INTO Customers (Name, Email) VALUES
('Ahmet Yılmaz', 'ahmet@email.com'),
('Mehmet Kaya', 'mehmet@email.com'),
('Ayşe Demir', 'ayse@email.com');

```

```

-- Replication için gerekli ayarları yapın

```

```
USE AdventureWorks2019;
EXEC sp_replicationdboption
    @dbname = 'AdventureWorks2019',
    @optname = 'publish',
    @value = 'true';
```

```
--Availability Group Oluşturma
-- Önce veritabanını Full Recovery moduna alın
ALTER DATABASE TestDB_Primary SET RECOVERY FULL;
```

```
-- Full backup alın
BACKUP DATABASE TestDB_Primary
TO DISK = 'C:\temp\TestDB_Primary.bak';
```

```
-- Log backup alın
BACKUP LOG TestDB_Primary
TO DISK = 'C:\temp\TestDB_Primary.trn';
```

```
-- Availability Group Listener oluşturun
ALTER AVAILABILITY GROUP AG_TestDB
ADD LISTENER 'AG_TestDB_Listener' (
    WITH IP ((N'127.0.0.1', N'255.255.255.0')),
    PORT = 1433
);
```

```
--Mirroring db hazırlık
-- Principal veritabanını Full Recovery moduna alın
ALTER DATABASE TestDB_Secondary SET RECOVERY FULL;
```

```
-- Full backup alın
BACKUP DATABASE TestDB_Secondary
TO DISK = 'C:\temp\TestDB_Secondary.bak';
```

```
-- Log backup alın
BACKUP LOG TestDB_Secondary
TO DISK = 'C:\temp\TestDB_Secondary.trn';
```

```
-- Mirror veritabanı oluşturun (NORECOVERY ile)
RESTORE DATABASE TestDB_Secondary_Mirror
FROM DISK = 'C:\temp\TestDB_Secondary.bak'
WITH NORECOVERY,
MOVE 'TestDB_Secondary' TO 'C:\Program Files\Microsoft SQL
Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\TestDB_Secondary_Mirror.mdf',
MOVE 'TestDB_Secondary_Log' TO 'C:\Program Files\Microsoft SQL
```

Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\TestDB\_Secondary\_Mirror.ldf;

-- Log restore

```
RESTORE LOG TestDB_Secondary_Mirror
FROM DISK = 'C:\temp\TestDB_Secondary.trn'
WITH NORECOVERY;
```

--kurulum

-- Mirror veritabanında partner ayarlayın

```
ALTER DATABASE TestDB_Secondary_Mirror
SET PARTNER = 'TCP://localhost:5022';
```

-- Principal veritabanında partner ayarlayın

```
ALTER DATABASE TestDB_Secondary
SET PARTNER = 'TCP://localhost:5023';
```

-- Mirroring durumunu görüntüleyin

```
SELECT
    db.name,
    m.mirroring_role_desc,
    m.mirroring_state_desc,
    m.mirroring_safety_level_desc
FROM sys.database_amirroring m
JOIN sys.databases db ON m.database_id = db.database_id
WHERE m.mirroring_guid IS NOT NULL;
```

--failover

-- Availability Group'ta manual failover

```
ALTER AVAILABILITY GROUP AG_TestDB FAILOVER;
```

-- Failover durumunu kontrol

```
SELECT
    ag.name AS AvailabilityGroup,
    r.replica_server_name,
    r.role_desc,
    rs.is_local,
    rs.role
FROM sys.availability_groups ag
JOIN sys.availability_replicas r ON ag.group_id = r.group_id
JOIN sys.dm_hadr_availability_replica_states rs ON r.replica_id = rs.replica_id;
```

-- Test için connection string

-- Always On için: Server=AG\_TestDB\_Listener;Database=TestDB\_Primary;

```
-- Mirroring için: Server=localhost;Database=TestDB_Secondary;Failover
Partner=localhost;

-- Failover testi
-- 1. Primary'ye veri insert
USE TestDB_Primary;
INSERT INTO Customers (Name, Email) VALUES ('Test User', 'test@email.com');

-- 2. Failover
ALTER AVAILABILITY GROUP AG_TestDB FAILOVER;

-- 3. Yeni primary'de veriyi kontrol
SELECT * FROM Customers ORDER BY ID DESC;

-- Publisher'da veri ekleyin
USE AdventureWorks2019;
INSERT INTO Person.Person (PersonType, FirstName, LastName)
VALUES ('EM', 'Test', 'User');

-- Subscriber'da kontrol edin
USE AdventureWorks_Replica;
SELECT TOP 5 * FROM Person.Person ORDER BY ModifiedDate DESC;
```

## **6.Proje - Veri Temizleme ve ETL Süreçleri Tasarımı**

ETL (Extract, Transform, Load) şu üç adımı ifade eder:

1. Extract (Çıkarma): Veriyi farklı kaynaklardan alırsın.
2. Transform (Dönüştürme): Hatalı veya eksik verileri temizler, dönüştürürsün.
3. Load (Yükleme): Temizlenmiş verileri hedef tabloya yüklersin.

### **1. Veriyi Temizleme ve Dönüştürme**

- Geçerli e-posta adreslerini filtreleme:

SQLQuery5.sql - Io...25JVRG\ECCEM (112))\* X SQLQuery4.sql - Io...K25JVRG\ECCEM (70))\* SQLQuery3.sql - Io...

```
-- Geçerli e-posta için REGEX gibi kullanılabilecek LIKE filtresi
SELECT * FROM Customer_Source1
WHERE Email IS NOT NULL AND Email LIKE '%@%.%';
```

121 %

Results Messages

ID	FullName	Email	BirthDate
1	Ali Veli	ali@example.com	1990-12-31

Bu sorgu, geçerli formatta olan (örneğin @ ve . içeren) e-postaları seçer.

- Geçerli tarihleri dönüştür ve hatalı olanları dışla:

SQLQuery5.sql - Io...25JVRG\ECCEM (112))\* X SQLQuery4.sql - Io...K25JVRG\ECCEM (70))\* SQL

```
SELECT * FROM Customer_Source1
WHERE TRY_CONVERT(DATE, BirthDate, 120) IS NOT NULL;
```

121 %

Results Messages

ID	FullName	Email	BirthDate
1	Ali Veli	ali@example.com	1990-12-31

TRY\_CONVERT(DATE, ...), geçerli tarih formatında olmayanları dışlar.

## 2. Temizlenmiş Veriyi Hedefe Yükleme

SQLQuery5.sql - Io...25JVRG\ECCEM (112))\* X SQLQuery4.sql - Io...K25JVRG\ECCEM (70))\* SQLQuery3.sql - Io...

```
CREATE TABLE Customer_Staging (
    CustomerID INT,
    FullName NVARCHAR(100),
    Email NVARCHAR(100),
    BirthDate DATE,
    Source NVARCHAR(50)
);
-- Kaynak 1'den temizlenmiş veri yüklenir
INSERT INTO Customer_Staging
SELECT ID, FullName, Email, CONVERT(DATE, BirthDate), 'Source1'
FROM Customer_Source1
WHERE Email IS NOT NULL AND Email LIKE '%@%.%'
AND TRY_CONVERT(DATE, BirthDate, 120) IS NOT NULL;
-- Kaynak 2'den temizlenmiş veri yükle
INSERT INTO Customer_Staging
SELECT CustID, Name, Mail, CONVERT(DATE, DOB), 'Source2'
FROM Customer_Source2
WHERE Mail IS NOT NULL AND Mail LIKE '%@%.%'
AND TRY_CONVERT(DATE, DOB, 120) IS NOT NULL;
```

121 %

Messages

(1 row affected)

(2 rows affected)

## 3. Veri Kalitesi Raporu Oluşturma

- Kaç kaydın hatalı olduğunu görme

```
SQLQuery5.sql - Io...25JVRG\ECCEM (112))* X SQLQuery4.sql - Io...K25JVRG\ECCEM (70))* SQLQuery
-- Hatalı e-posta
SELECT COUNT(*) AS InvalidEmails FROM Customer_Source1
WHERE Email IS NULL OR Email NOT LIKE '%@%.%';

-- Geçersiz doğum tarihi
SELECT COUNT(*) AS InvalidBirthDates FROM Customer_Source1
WHERE TRY_CONVERT(DATE, BirthDate, 120) IS NULL;
```

121 %

Results Messages

	InvalidEmails
1	2

	InvalidBirthDates
1	2

- Hedef tabloya yüklenen toplam veri sayısı

```
SQLQuery5.sql - Io...25JVRG\ECCEM (112))* X SQLQuery4.sql - Io...K25JVRG\ECCEM (70))* SQLQuery3.
SELECT COUNT(*) AS LoadedCustomers FROM Customer_Staging;
```

121 %

Results Messages

	LoadedCustomers
1	3

## **7.Proje - Veritabanı Yükseltme ve Sürüm Yönetimi**

### **1. Veritabanı Yükseltme Planı**

- BACKUP DATABASE ile tüm verilerin yedeğini alınır.
- Hedef sunucuya restore edilir (yeni sürüm).
- Uyumluluk seviyesi (compatibility level) kontrol edilir ve gerekirse güncellenir.
- Veri bütünlüğü ve uygulama uyumluluğu testleri yapılır.

```
SQLQuery5.sql - Io...25JVRG\ECCEM (112))* X SQLQuery4.sql - Io...K25JVRG\ECCEM (70))* SQLQuery3.sql - I
RESTORE DATABASE [AdventureWorks2022]
FROM DISK = 'C:\Backup\AdventureWorks2016.bak'
WITH MOVE 'EskiVeritabani_Data' TO 'C:\MSSQL\YeniVeritabani.mdf',
MOVE 'EskiVeritabani_Log' TO 'C:\MSSQL\YeniVeritabani.ldf',
REPLACE;
```

121 %

Messages

Completion time: 2025-05-28T21:40:27.4494186+03:00

Uyumluluk seviyesi yükseltilir.

```
SQLQuery6.sql - Io...K25JVRG\ECEM (51))* X SQLQuery5.sql - Io...25JVRG\ECEM
ALTER DATABASE [Test1]
SET COMPATIBILITY_LEVEL = 160;
```

## 2. Sürüm Yönetimi

**Amaç:** Veritabanı şemasında (tablo, prosedür, view vs.) yapılan tüm değişiklikleri izlemek.

DDL Trigger oluşturulur:

```
SQLQuery6.sql - Io...K25JVRG\ECEM (51))* X SQLQuery5.sql - Io...25JVRG\ECEM (112))* SQLQuery4.sql - Io...K25JVRG\ECEM
CREATE TRIGGER tr_TrackSchemaChanges
ON DATABASE
FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE,
CREATE_PROCEDURE, ALTER_PROCEDURE, DROP_PROCEDURE
AS
BEGIN
    INSERT INTO SchemaChangeLog (EventType, ObjectName, UserName, EventDate)
    VALUES (
        EVENTDATA().value('(/EVENT_INSTANCE/EventType)[1]', 'NVARCHAR(100)'),
        EVENTDATA().value('(/EVENT_INSTANCE/ObjectName)[1]', 'NVARCHAR(100)'),
        SYSTEM_USER,
        GETDATE()
    );
END;
```

121 %

Messages

Commands completed successfully.

Completion time: 2025-05-28T21:47:45.5942237+03:00

Takip tablosu oluşturulur:

```
SQLQuery6.sql - Io...K25JVRG\ECEM (51))* X SQLQuery5.sql - Io...25JVRG\ECEM (112))*
CREATE TABLE SchemaChangeLog (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    EventType NVARCHAR(100),
    ObjectName NVARCHAR(100),
    UserName NVARCHAR(100),
    EventDate DATETIME
);
```

121 %

Messages

(1 row affected)

Completion time: 2025-05-28T21:49:15.3406113+03:00

## 3. Test ve Geri Dönüş Planı

- Tüm sorgular, stored procedure'ler, view'lar kontrol edilir.
- Uygulama veritabanına bağlantı kurabiliyor mu kontrol edilir.
- Performans testleri yapılır.

Geri dönüş planı



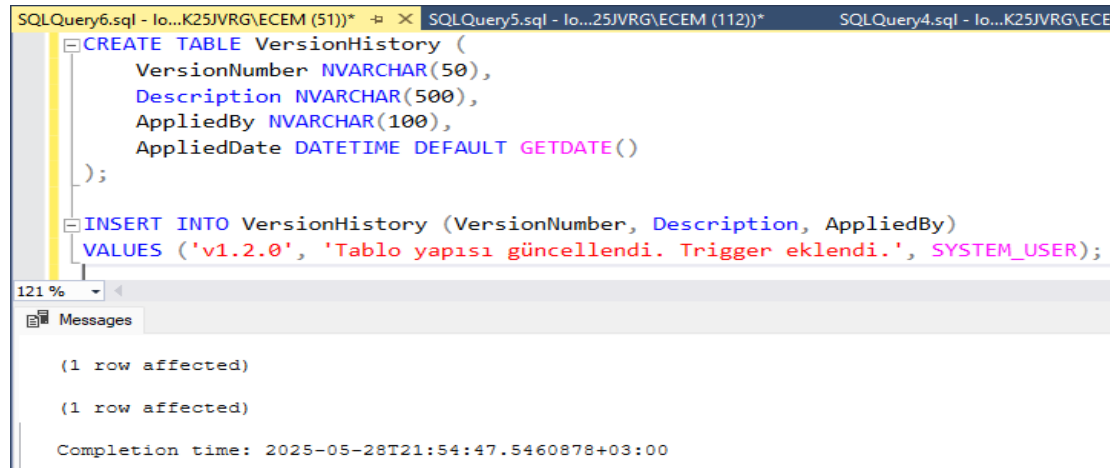
Eğer yükseltme başarısız olursa:

- Yeni veritabanı silinir
- Önceki yedek geri yüklenir

Geri dönüş örneği:

```
RESTORE DATABASE [AdventureWorks2022]
FROM DISK = 'C:\Backup\AdventureWorks2016.bak'
WITH REPLACE;
```

Sürüm değişikliklerini not almak için basit bir versiyon kontrol tablosu da kullanılabilir:



```
SQLQuery6.sql - Io...K25JVRG\ECEM (51)))*  X SQLQuery5.sql - Io...25JVRG\ECEM (112)))* SQLQuery4.sql - Io...K25JVRG\ECEM (112)))*
CREATE TABLE VersionHistory (
    VersionNumber NVARCHAR(50),
    Description NVARCHAR(500),
    AppliedBy NVARCHAR(100),
    AppliedDate DATETIME DEFAULT GETDATE()
);
INSERT INTO VersionHistory (VersionNumber, Description, AppliedBy)
VALUES ('v1.2.0', 'Tablo yapısı güncellendi. Trigger eklendi.', SYSTEM_USER);
```

121 %

Messages

(1 row affected)

(1 row affected)

Completion time: 2025-05-28T21:54:47.5460878+03:00