

ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



Ağ Tabanlı Paralel Dağıtım Sistemleri
(BLM4522)

Sümeyye TEKİN – 20290296
(Github: github.com/sumeyye4/BLM4522)

VERİTABANI YEDEKLEME VE FELAKETTEN KURTARMA PLANI

AdventureWorks2022 veritabanı için yedekleme ve felaketten kurtarma planı hazırlanmış ve test edilmiştir. Amaç, veritabanı bütünlüğünü sağlamak, veri kaybını önlemek ve acil durumlarda hızlı bir şekilde kurtarma işlemleri gerçekleştirebilmektir.

1. Yedekleme Stratejileri

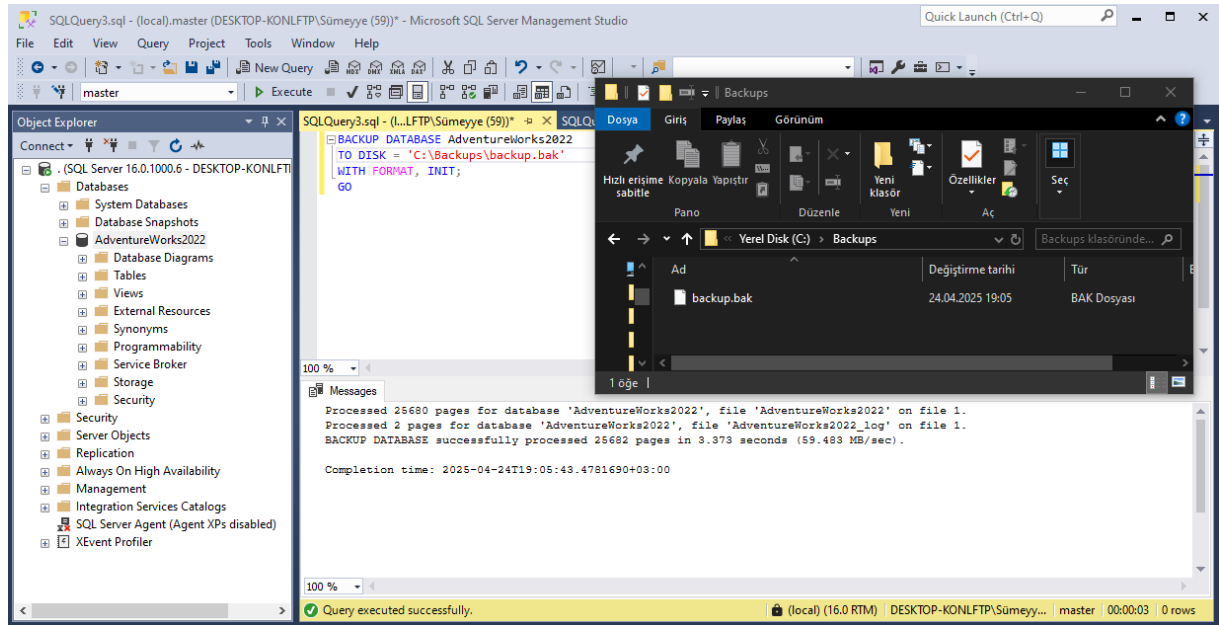
1.1. Tam Yedekleme (Full Backup)

Veritabanının o andaki tam kopyası alındı. BACKUP DATABASE komutu kullanılarak backup.bak dosyası oluşturuldu.

BACKUP DATABASE AdventureWorks2022

TO DISK = 'C:\Backups\backup.bak'

WITH FORMAT, INIT;



Şekil 1: backup.bak dosyası başarıyla oluşturulması ve yedek alınması.

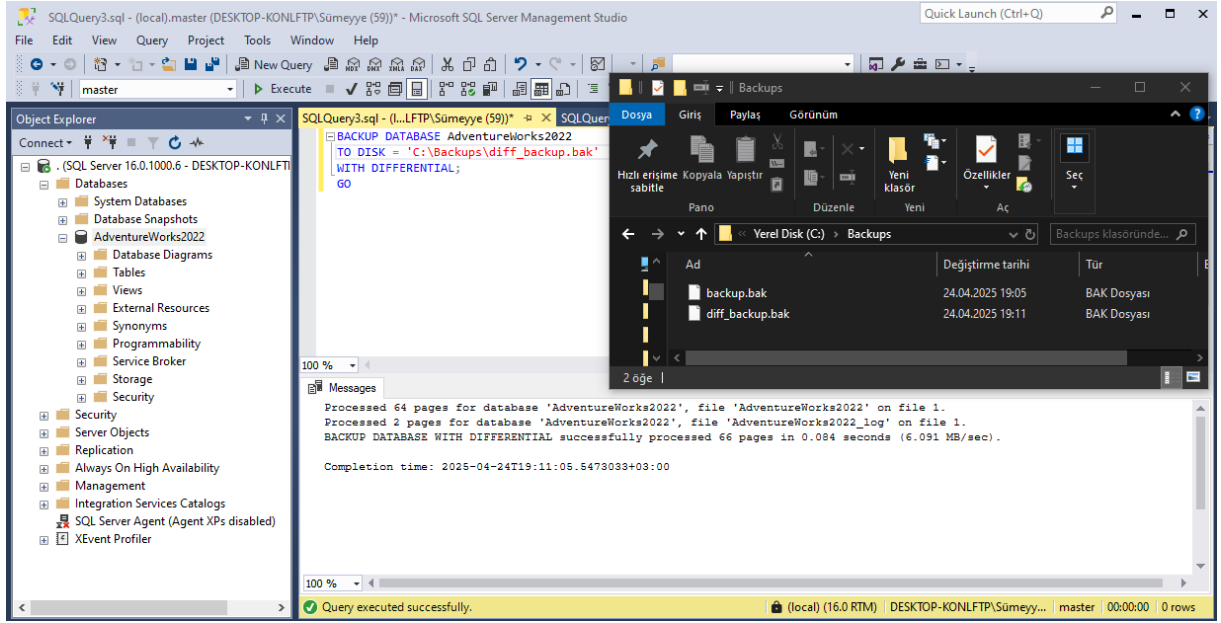
1.2. Artımlı Yedekleme (Differential Backup)

Full yedekten sonra değişen veriler hızlıca yedeklendi. BACKUP DATABASE ... WITH DIFFERENTIAL komutu çalıştırıldı ve diff_backup.bak oluşturuldu.

BACKUP DATABASE AdventureWorks2022

TO DISK = 'C:\Backups\diff_backup.bak'

WITH DIFFERENTIAL;



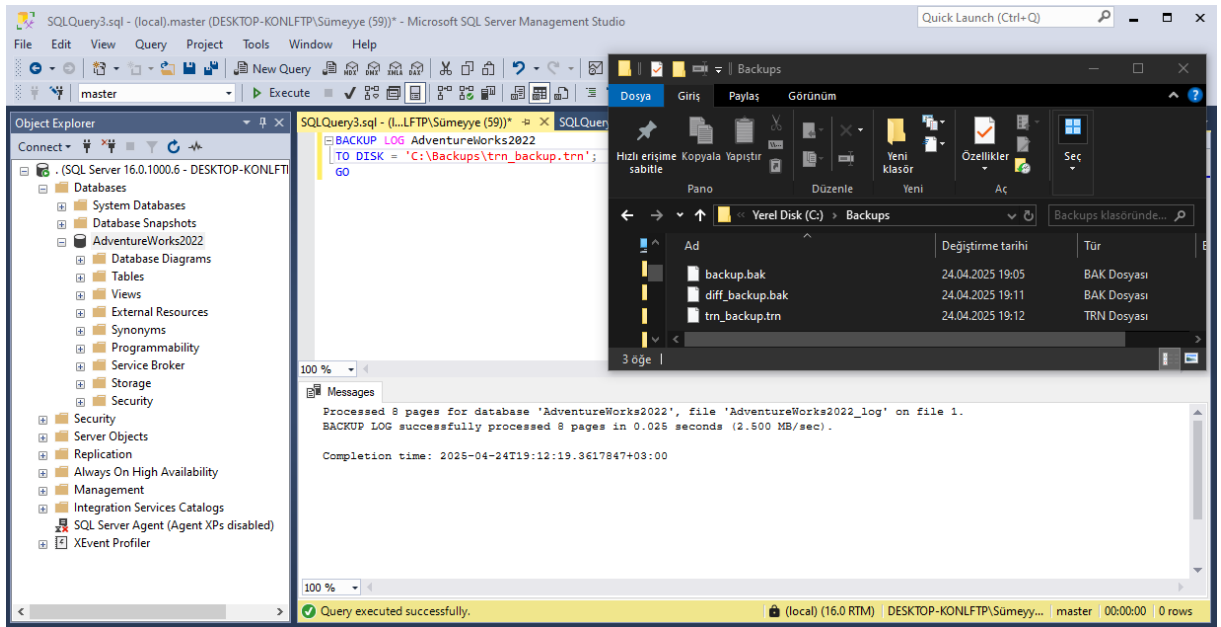
Şekil 2: Değişiklikleri içeren artımlı yedek dosyasının elde edilmesi.

1.3. Transaction Log Yedekleme

Point-in-time kurtarma yeteneği sağlandı. BACKUP LOG komutu ile trn_backup.trn dosyası oluşturuldu.

BACKUP LOG AdventureWorks2022

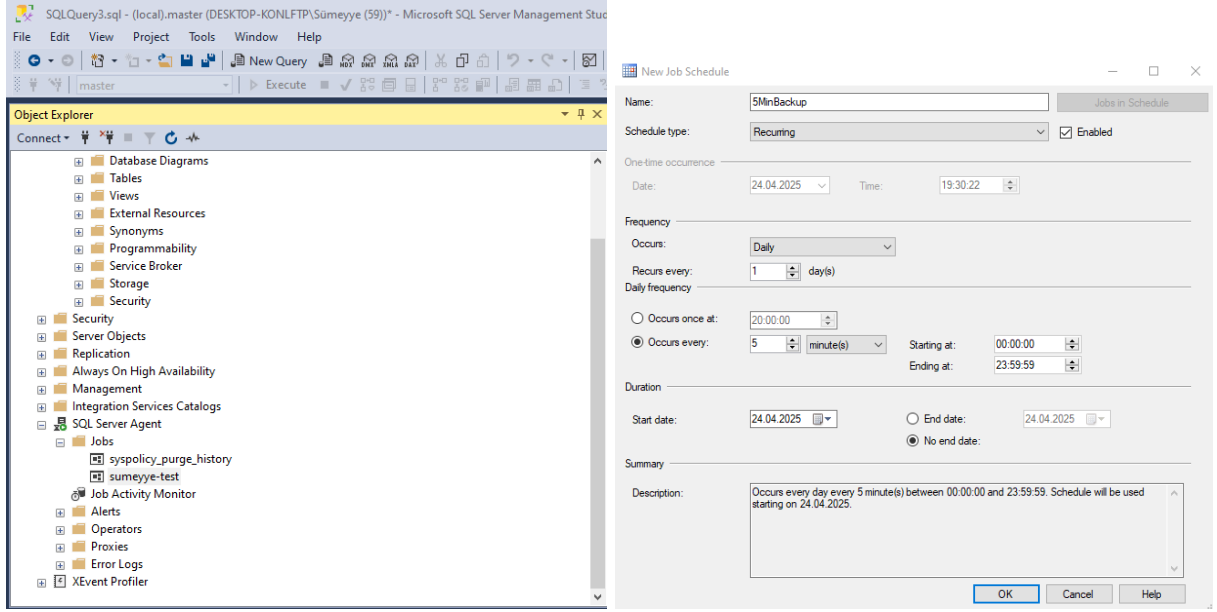
TO DISK = 'C:\Backups\trn_backup.trn';



Şekil 3: Transaction log yedeği başarıyla alınması.

1.4. Yedek Otomasyonu (SQL Server Agent Job)

Yedek işlemleri düzenli ve otomatik olarak yapıldı. sumeyye-test adında bir job oluşturuldu; her 5 dakikada bir full backup alacak şekilde schedule (5MinBackup) tanımlandı.



Şekil 4: Otomatik full yedek job'ın başarıyla oluşturulması.

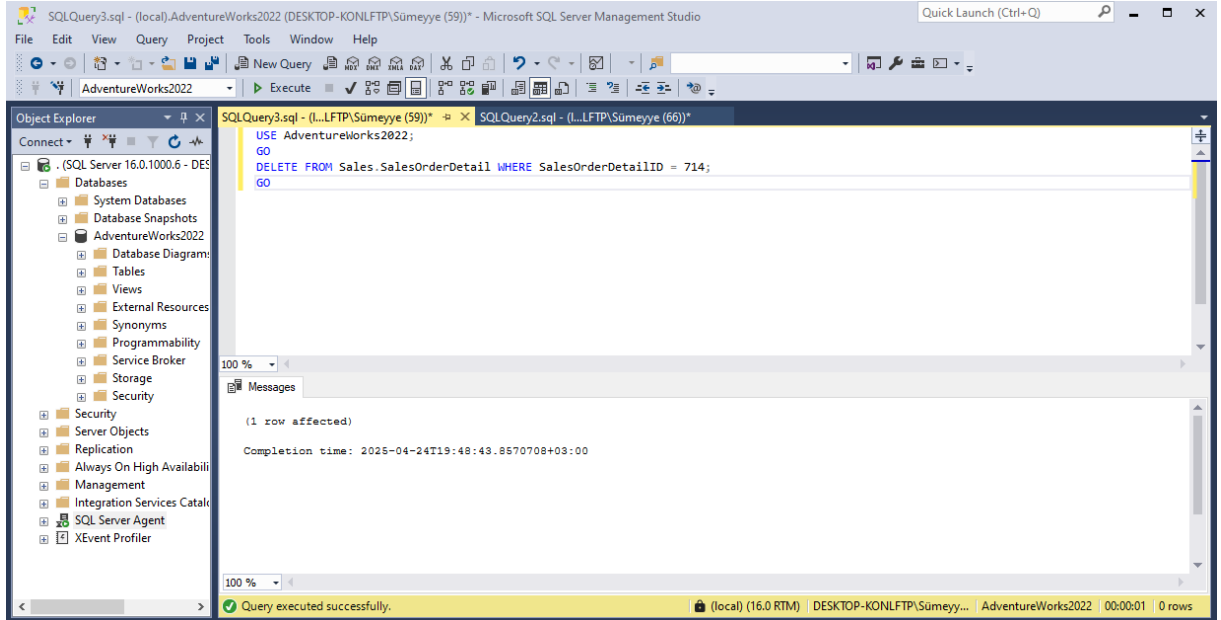
2. Felaketten Kurtarma Senaryoları

2.1. Veri Silme Simülasyonu

Kurtarma adımlarının testi için veri kaybı simüle edildi. SalesOrderDetailID = 714 satırı silindi.

USE AdventureWorks2022;

DELETE FROM Sales.SalesOrderDetail WHERE SalesOrderDetailID = 714;



Şekil 5: Kayıt silinmesi.

2.2. Tam Restore (Full Restore)

Full yedekten veritabanını tamamen geri yüklendi. RESTORE DATABASE komutuyla backup.bak dosyası kullanılarak veritabanı üzerine yazıldı (WITH REPLACE).

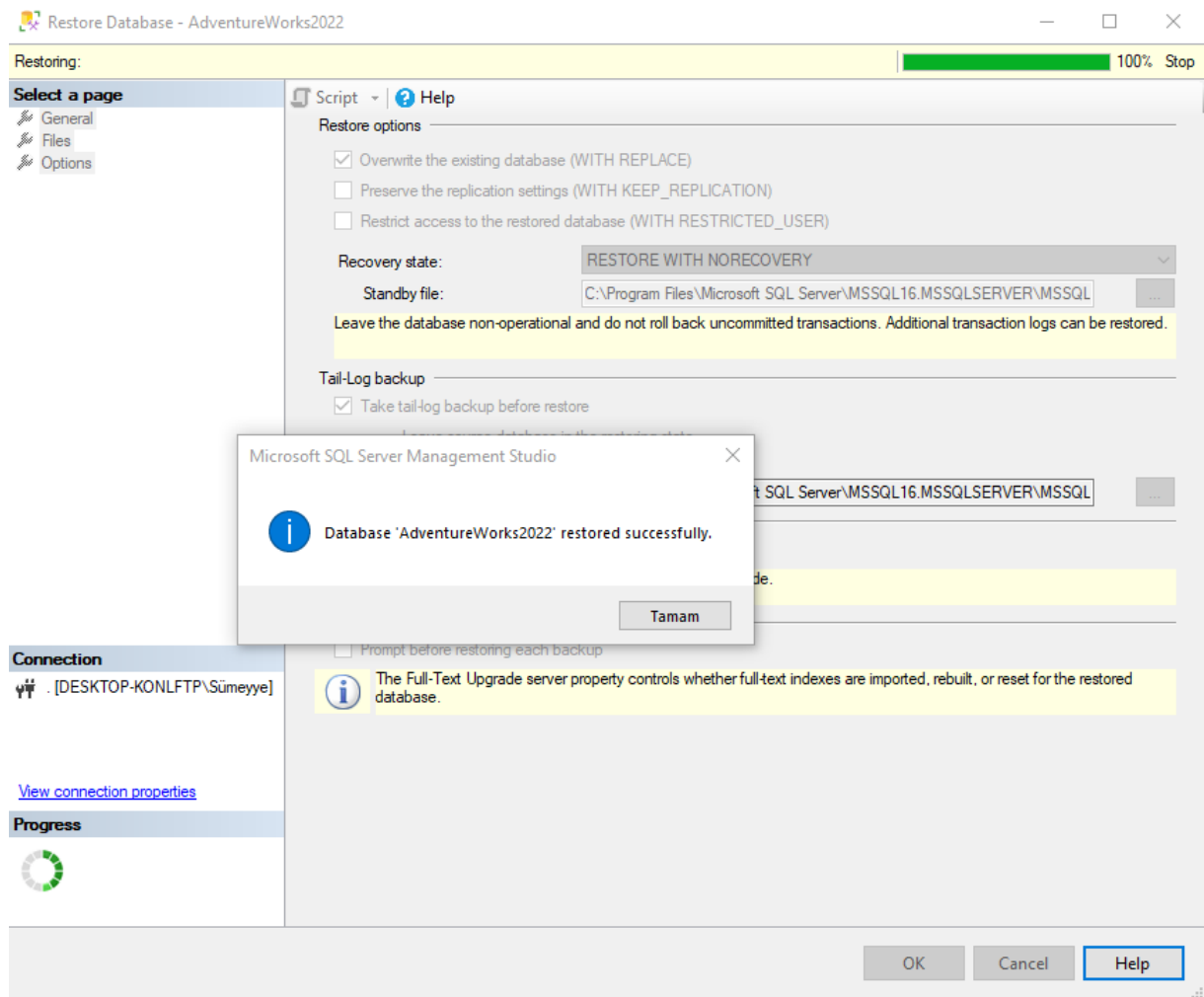
```
ALTER DATABASE AdventureWorks2022 SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
```

```
RESTORE DATABASE AdventureWorks2022
```

```
FROM DISK = 'C:\Backups\backup.bak'
```

```
WITH REPLACE;
```

```
ALTER DATABASE AdventureWorks2022 SET MULTI_USER;
```



Şekil 6: Veritabanının tam olarak geri yüklenmesi.

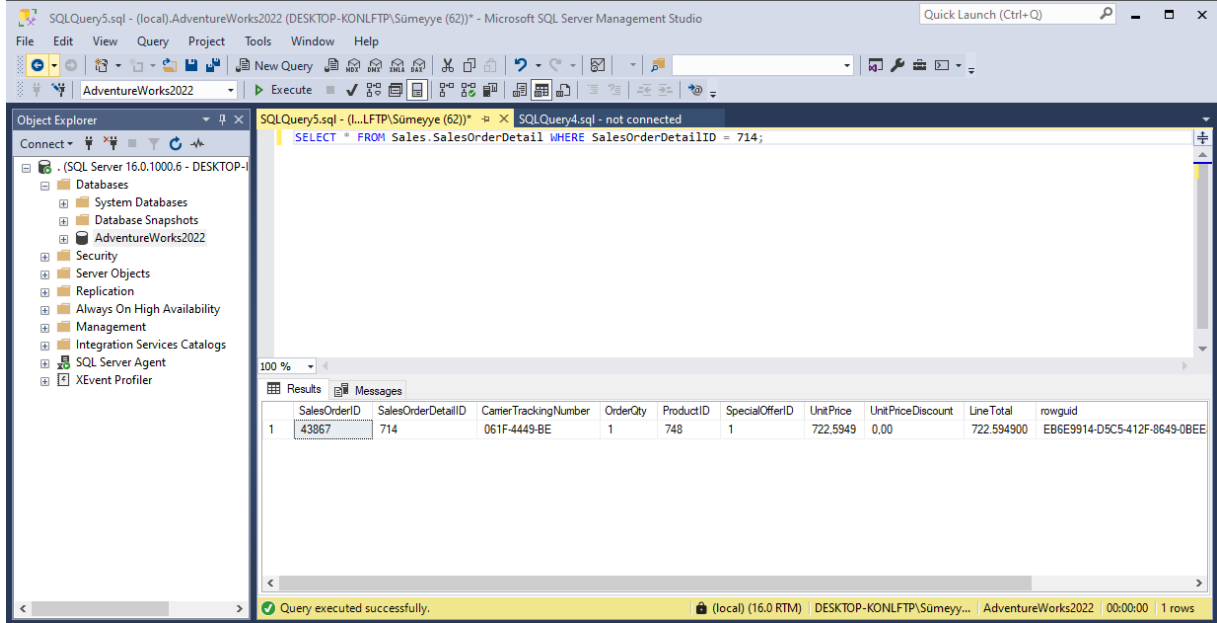
2.3. Zaman Noktası Kurtarma (Point-in-Time Restore)

Belirli bir tarih-saat dilimine dönerek silinen veri kurtarıldı. Transaction log yedeği trn_backup.trn kullanılarak, STOPAT parametresiyle geri yükleme yapıldı ve silinen verinin kurtarıldığı görüldü.

RESTORE LOG AdventureWorks2022

FROM DISK = 'C:\Backups\trn_backup.trn'

WITH STOPAT = '2025-04-24 14:30:00', RECOVERY;



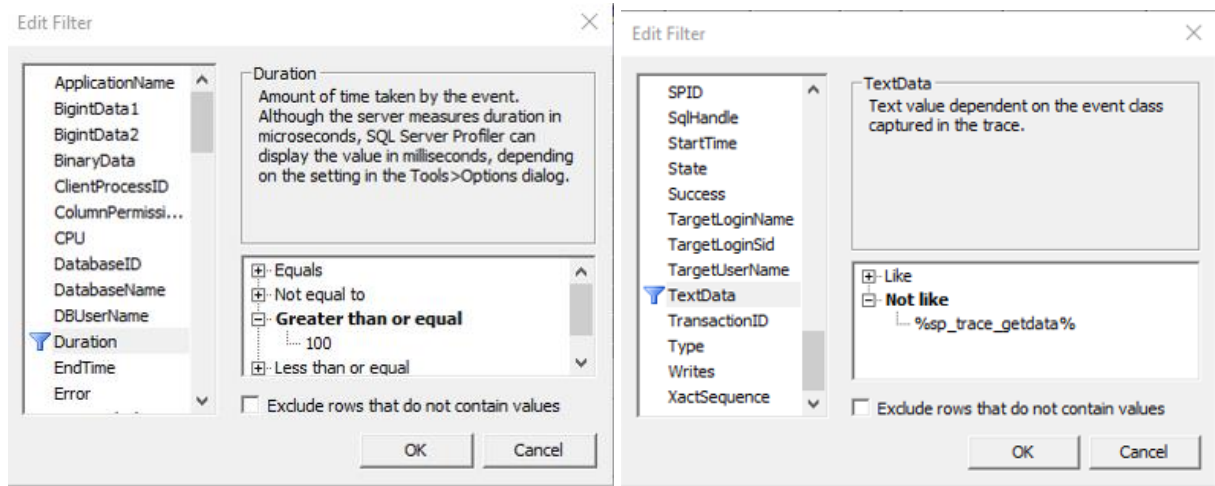
Şekil 7: Söz konusu kayıt geri yüklenerek veri kurtarmanın başarıyla gerçekleştirilmesi.

VERİTABANI PERFORMANS OPTİMİZASYONU VE İZLEME

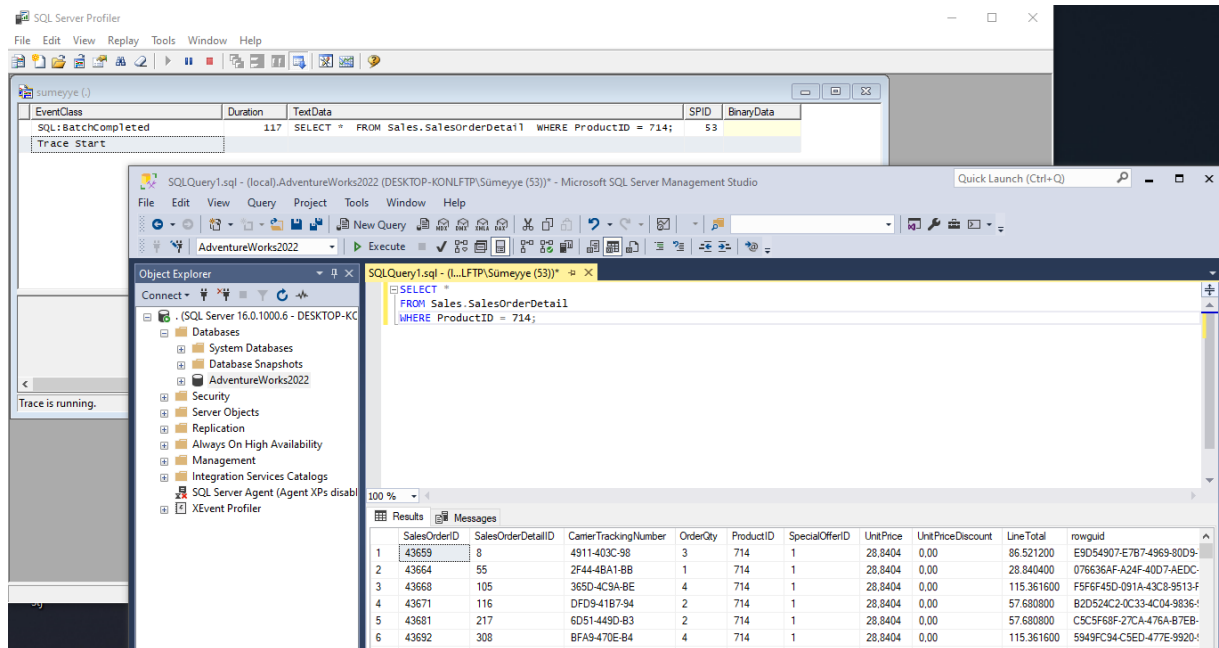
AdventureWorks2022 veritabanındaki yavaş sorgular tespit edilmiş, analiz edilmiş ve indeks optimizasyonu ile performans iyileştirmesi yapılmıştır. Amaç, sorgu sürelerini kısaltmak ve sistem kaynak kullanımını azaltmaktır.

1. SQL Server Profiler ile İzleme

100 milisaniyeden uzun süren sorgular gerçek zamanlı yakalandı. Profiler açıldı, **TSQL_Duration** şablonu seçildi; **Events Selection** → **Column Filters** bölümünde Duration ≥ 100 ms ve TextData NOT LIKE '%sp_trace_getdata%' filtreleri uygulandı ve **Run** ile izleme başlatıldı.



Şekil 8: Events Selection → Column Filters bölümünde Duration ve TextData filtreleri.

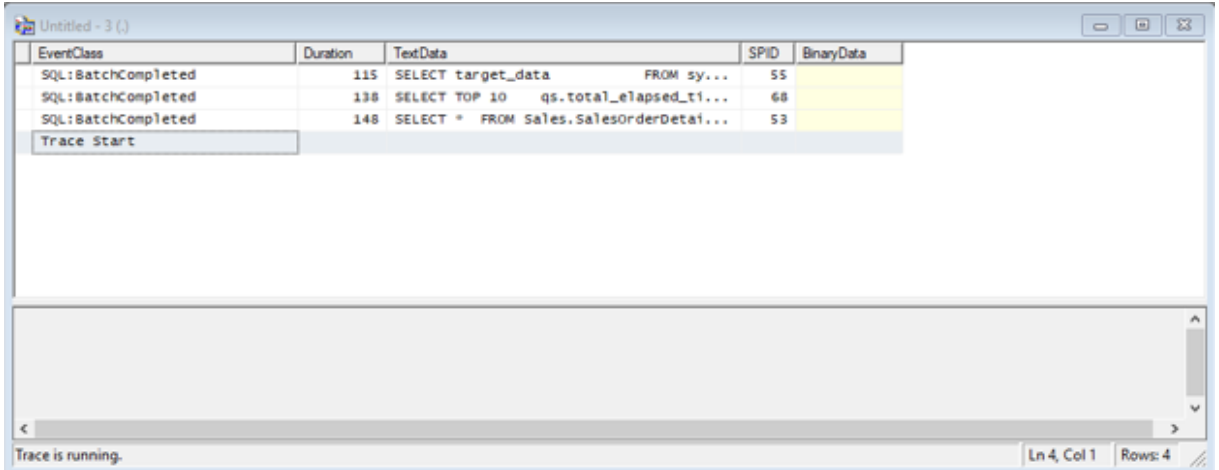


Şekil 9: SSMS üzerinden örnek sorgular gönderilerek slow query'lerin Profiler'da listelenmesi.

2. Dynamic Management Views (DMV) ile Sorgu İstatistikleri

Cache’de tutulan yavaş sorgular sorgulanarak en yüksek ortalama süreye sahip olanı belirlendi. Aşağıdaki script çalıştırıldı:

```
SELECT TOP 10  
    qs.total_elapsed_time/qs.execution_count AS avg_duration_ms,  
    qs.execution_count,  
    qt.text AS query_text  
FROM sys.dm_exec_query_stats qs  
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) qt  
WHERE qt.text NOT LIKE 'sys.%'  
ORDER BY avg_duration_ms DESC;
```



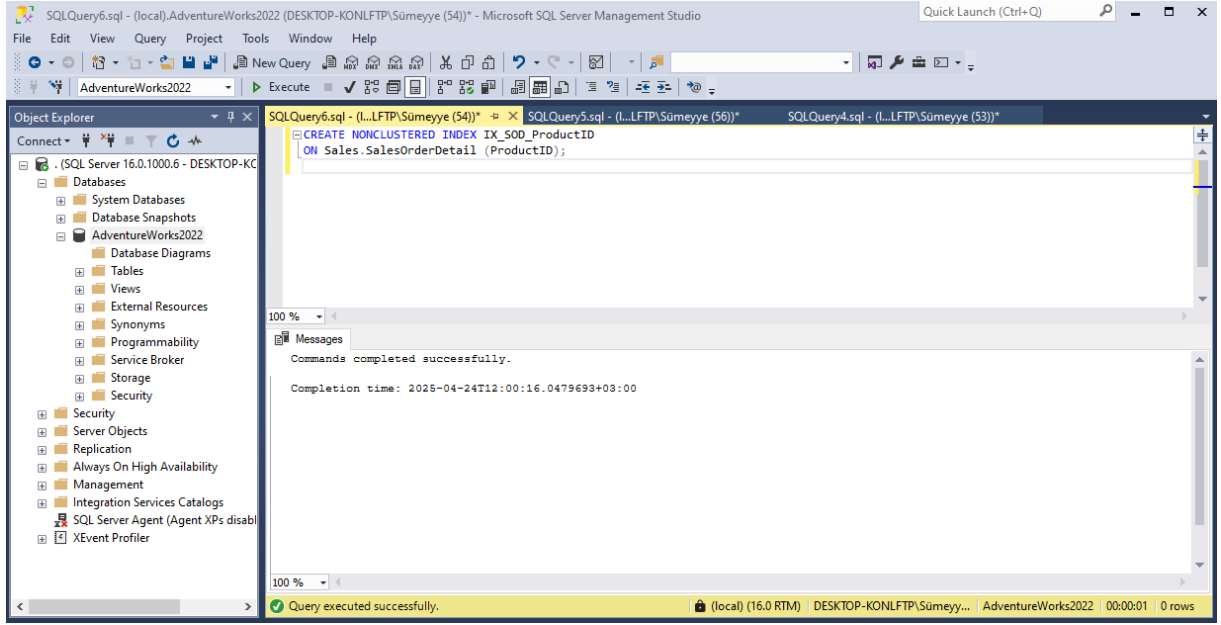
EventClass	Duration	TextData	SPID	BinaryData
SQL:BatchCompleted	115	SELECT target_data FROM sy...	55	
SQL:BatchCompleted	138	SELECT TOP 10 qs.total_elapsed_ti...	68	
SQL:BatchCompleted	148	SELECT * FROM Sales.SalesOrderDetai...	53	
Trace Start				

Şekil 10: En yavaş sorgunun SELECT * FROM Sales.SalesOrderDetail WHERE ProductID = 714; olarak saptanması.

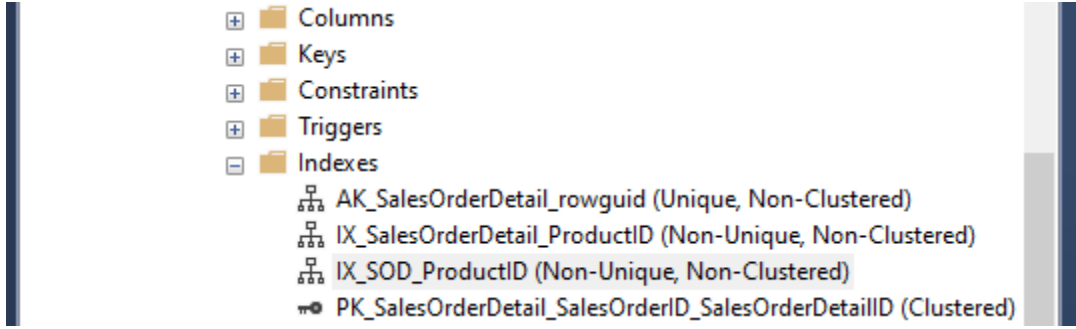
3. İndeks Oluşturma

Belirlenen sorgunun çalıştırma maliyeti düşürüldü. Aşağıdaki komut çalıştırıldı:

```
CREATE NONCLUSTERED INDEX IX_SOD_ProductID  
ON Sales.SalesOrderDetail (ProductID);
```

Şekil 11: T-SQL ile indeks oluşturma script'i ve başarı mesajı.

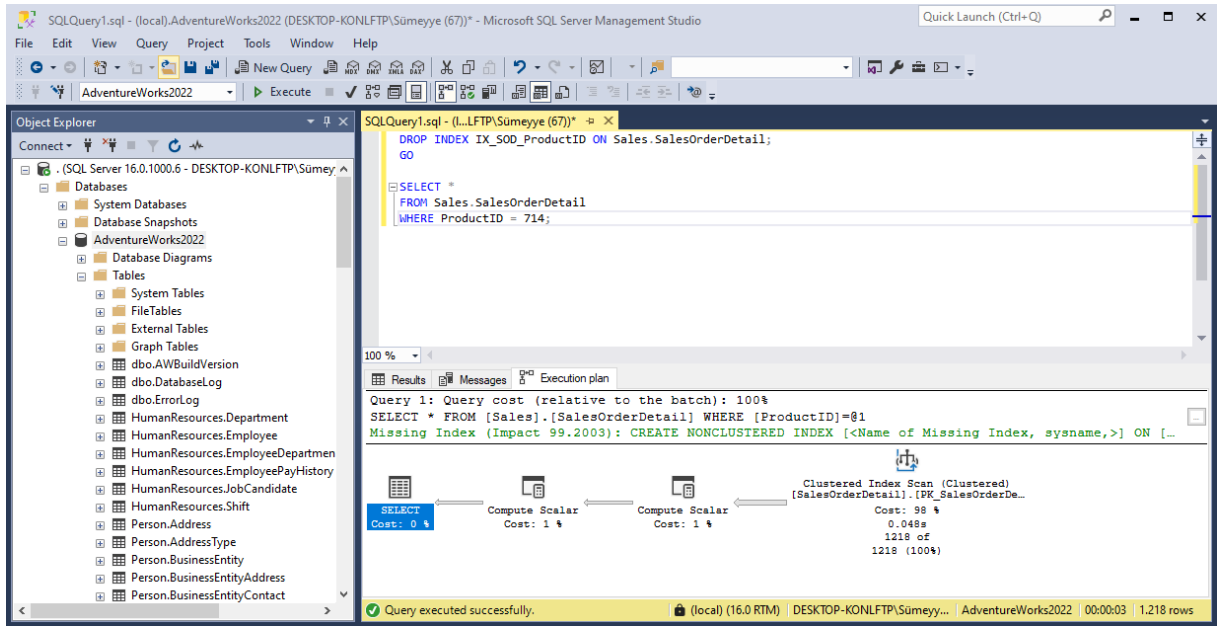


Şekil 12: IX_SOD_ProductID indeksinin başarıyla eklenmesi.

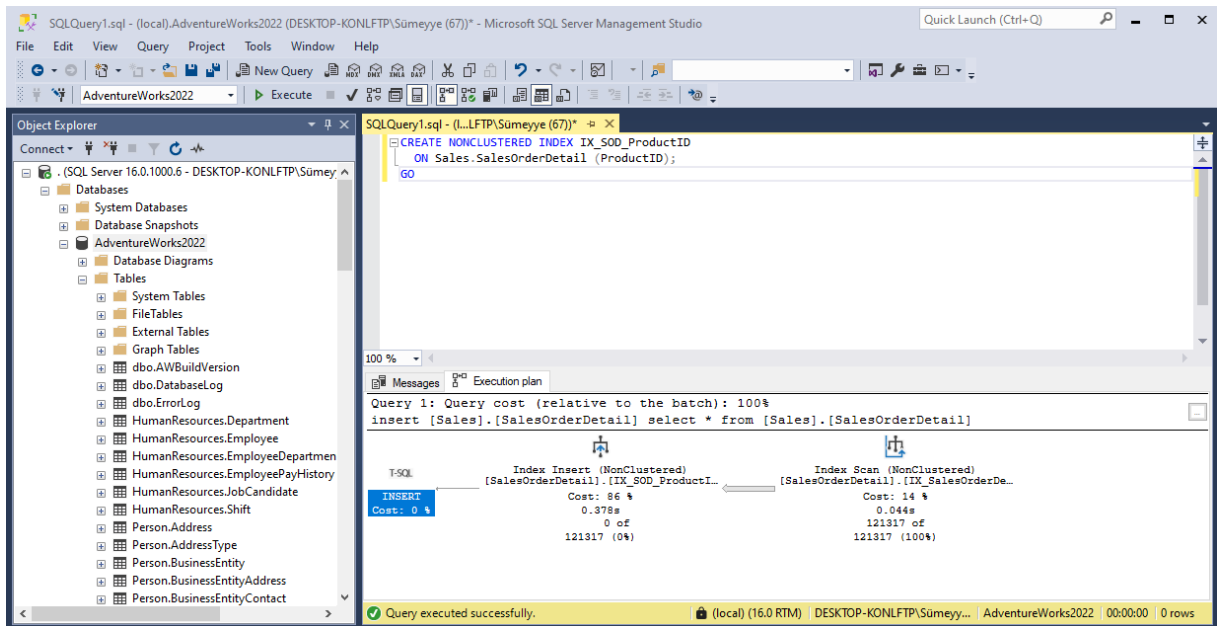
4. Execution Plan Testi

İndeksin sorgu planına etkisi görsel ve metrik olarak doğrulandı. **Include Actual Execution Plan** aktif edilerek sorgu önce ve sonra çalıştırıldı.

- Önce: **SELECT * FROM Sales.SalesOrderDetail WHERE ProductID = 714;** → **Index Scan**
- Sonra: Aynı sorgu → **Index Seek**



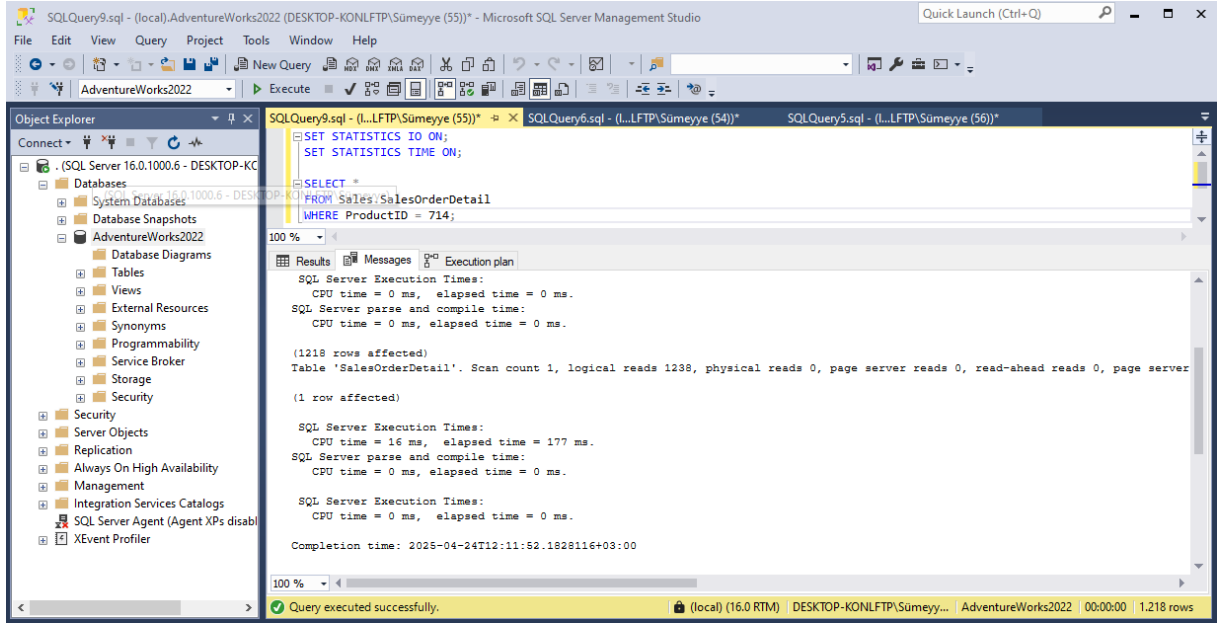
Şekil 13: İndeks öncesi Execution Plan – Index Scan.



Şekil 14: İndeks sonrası Execution Plan – Index Seek.

4. STATISTICS TIME & IO

Disk okuma ve işlem sürelerindeki iyileşme sayısal olarak belgelendi. **SET STATISTICS IO ON; SET STATISTICS TIME ON;** komutlarıyla sorgu çalıştırıldı.



Şekil 15: STATISTICS IO/TIME sonuçları.

VERİTABANI GÜVENLİĞİ VE ERİŞİM KONTROLÜ

AdventureWorks2022 veritabanı üzerinde güvenlik politikasının uygulanması, kullanıcı erişimlerinin kontrolü, kolon bazlı şifreleme ve etkinlik denetimi (audit) adımları yapılmıştır. Amaç, yetkisiz erişimi engellemek, hassas verinin korunmasını sağlamak ve kullanıcı etkinliklerini izlemektir.

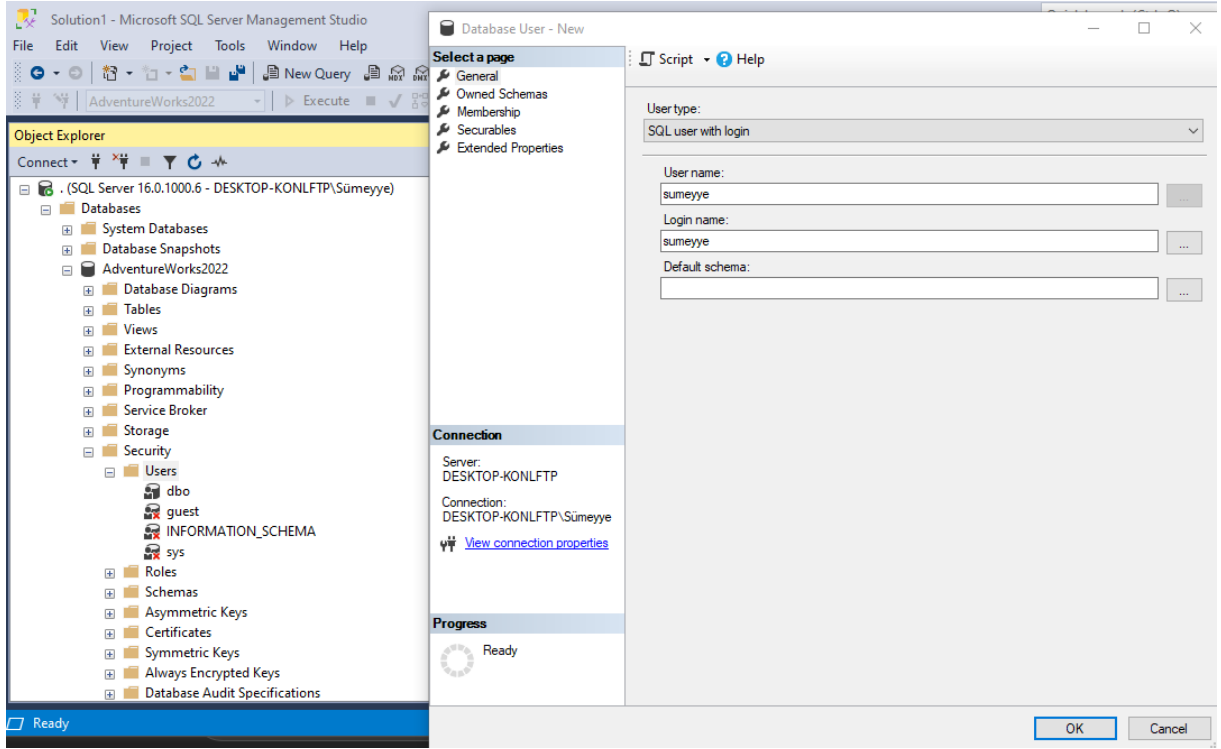
1. Login ve Database User Oluşturma

Uygulama kullanıcıları için ayrı SQL login ve veritabanı kullanıcı hesabı tanımlandı..

- Security → Logins → New Login... sihirbazında **sumeyye** SQL Server login'i oluşturuldu, parola atandı.
- AdventureWorks2022 → Security → Users → New User... sihirbazı ile **sumeyye** veritabanı kullanıcısı eklendi.

The screenshot shows the 'Login - New' dialog box in SQL Server Enterprise Manager. The dialog is for creating a new login named 'sumeyye'. The 'Login name' field is filled with 'sumeyye'. The 'Authentication' section has 'SQL Server authentication' selected. The 'Password' and 'Confirm password' fields are filled with masked characters. The 'Enforce password policy', 'Enforce password expiration', and 'User must change password at next login' checkboxes are all checked. The 'Mapped to certificate', 'Mapped to asymmetric key', and 'Map to Credential' options are not selected. The 'Default database' is set to 'AdventureWorks2022' and the 'Default language' is set to '<default>'. The 'Progress' section shows 'Ready'. The 'OK' and 'Cancel' buttons are at the bottom right.

Şekil 16: New Login penceresi.

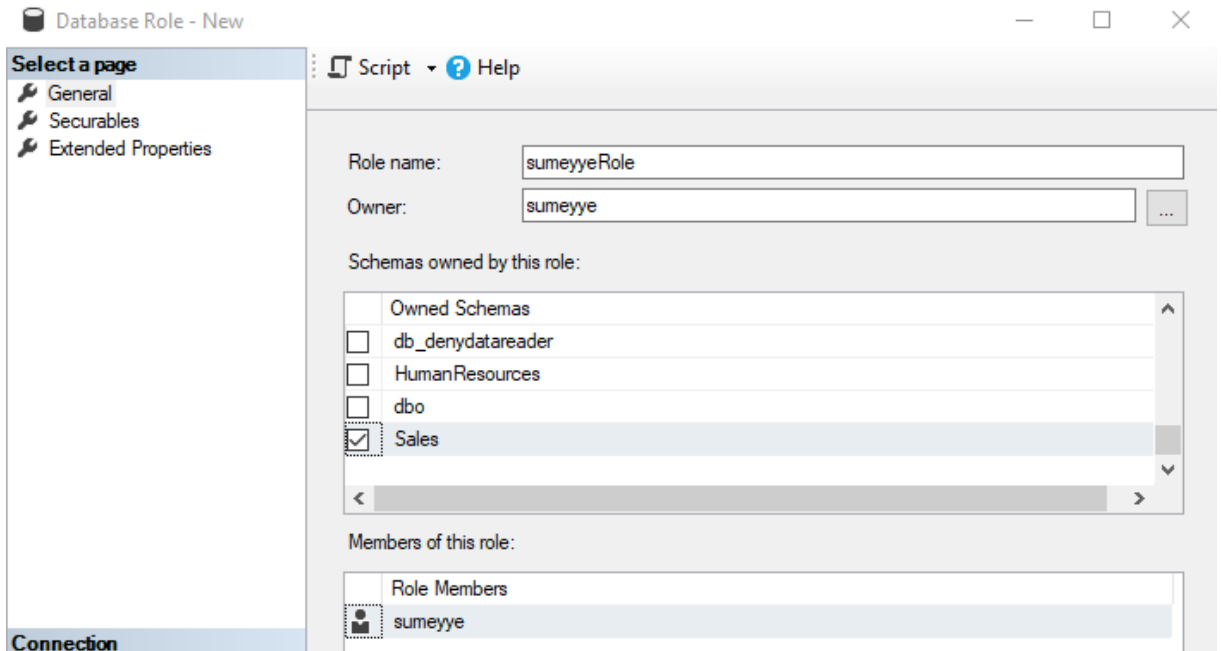


Şekil 17: New User penceresi.

2. Rol Tabanlı Erişim Kontrolü

Belirli tablolar için sadece SELECT yetkisine sahip bir rol oluşturmak.

- AdventureWorks2022 → Security → Roles → Database Roles → New Database Role... ile **sumeyyeRole** rolü oluşturuldu.
- Members sekmesinden sumeyye rolün üyesi yapıldı.

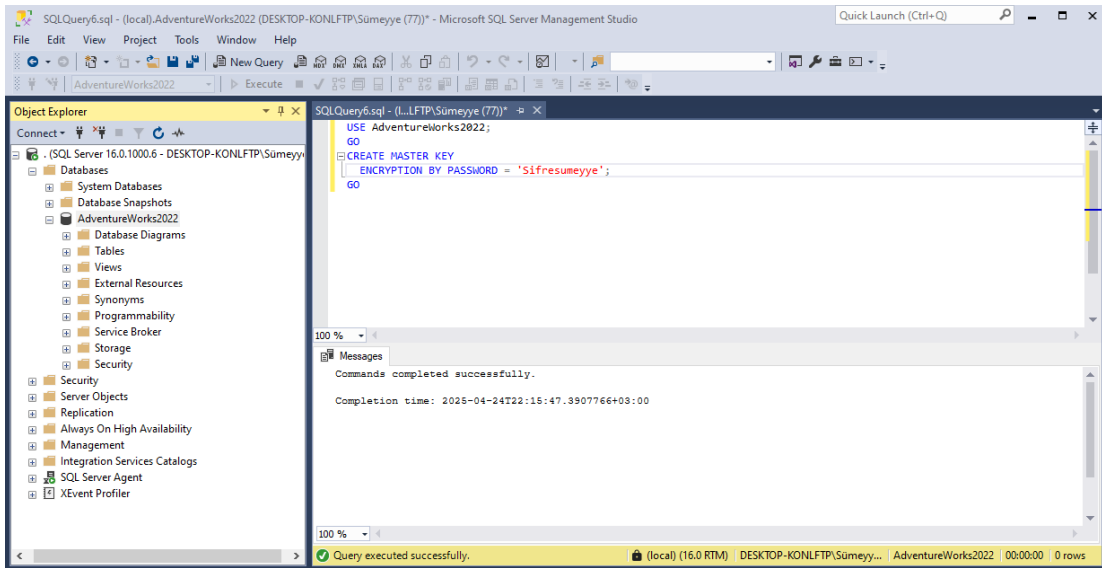


Şekil 18: New Database Role penceresi.

3. Şifreleme (T-SQL Symmetric Key)

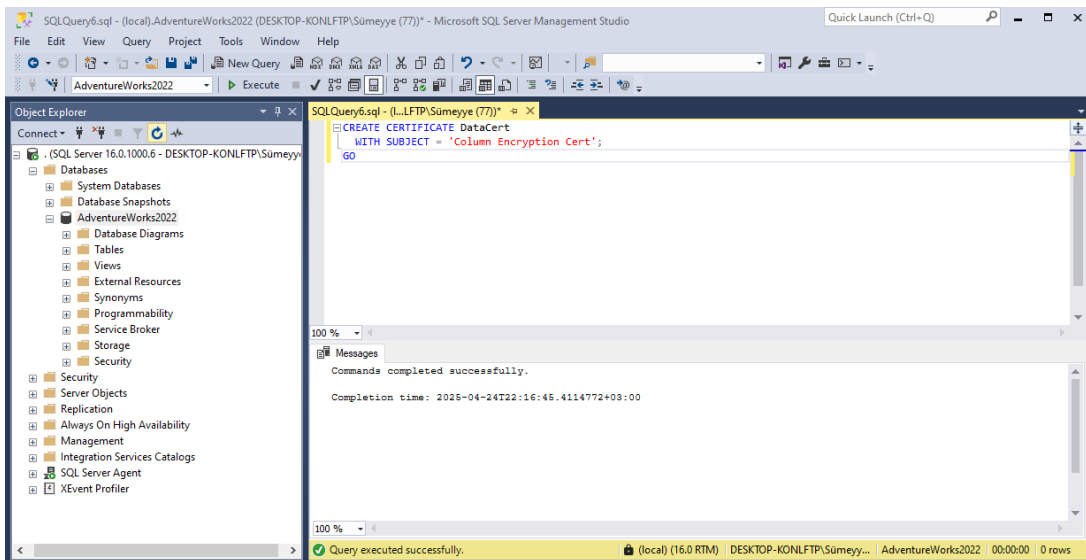
Hassas LineTotal sütunu şifrelendi ve sadece yetkili anahtar açıldığında okunabilir hale getirildi.

- Master Key oluşturuldu:
USE AdventureWorks2022;
GO
CREATE MASTER KEY
ENCRYPTION BY PASSWORD = 'Sifresumeyye';
GO



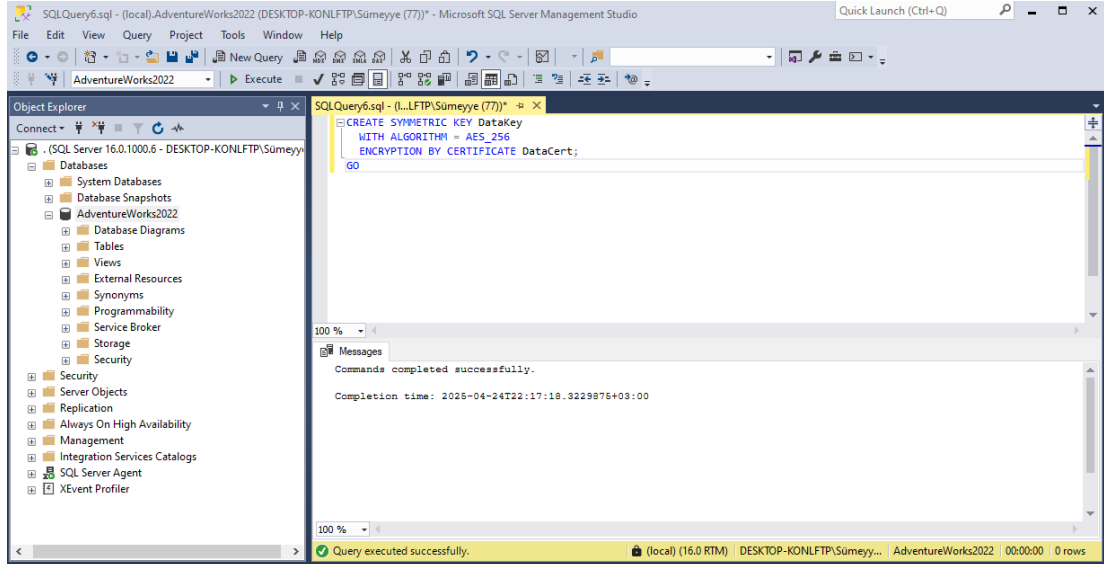
Şekil 19: Master key oluşturulması.

- Sertifika oluşturuldu:
CREATE CERTIFICATE DataCert
WITH SUBJECT = 'Column Encryption Cert';
GO



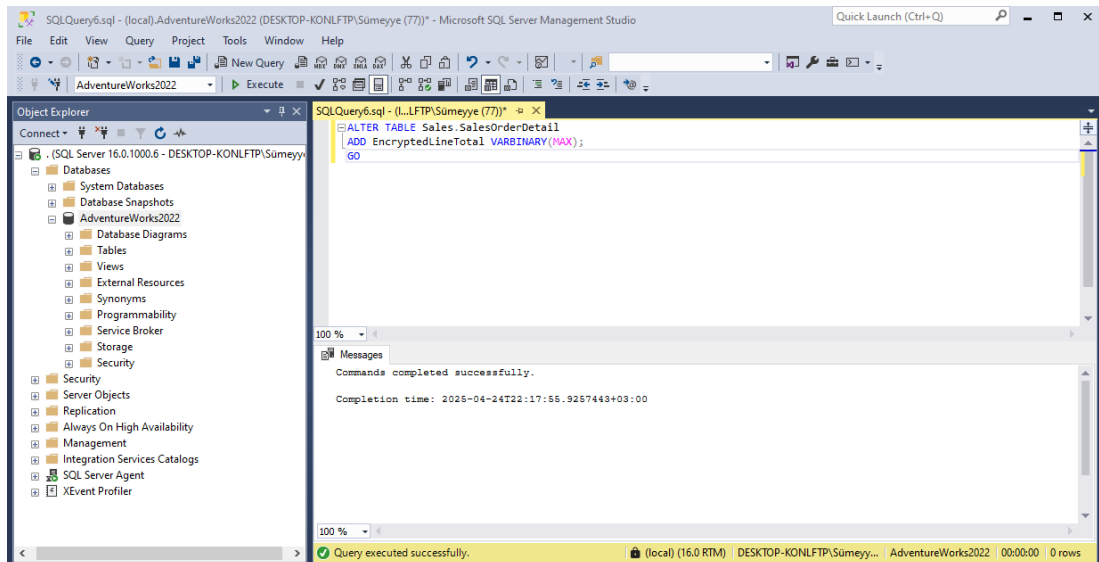
Şekil 20: Sertifika oluşturulması.

- Symmetric Key oluşturuldu:
CREATE SYMMETRIC KEY DataKey
WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE DataCert;
GO



Şekil 21: Symmetric key oluşturulması.

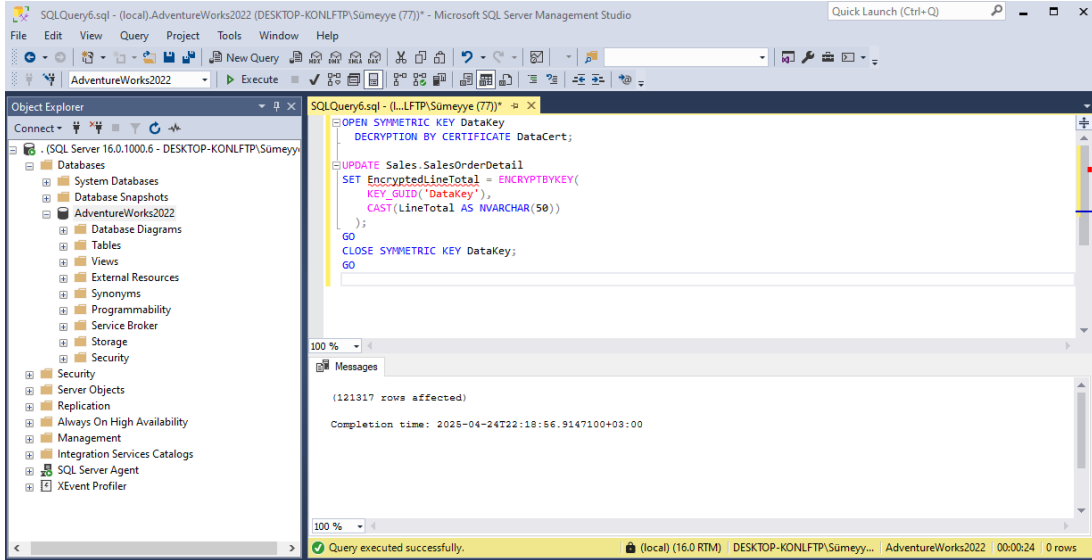
- Yeni kolon eklendi:
ALTER TABLE Sales.SalesOrderDetail
ADD EncryptedLineTotal VARBINARY(MAX);
GO



Şekil 22: Yeni kolon eklenmesi.

- Veri şifreleme işlemi gerçekleştirildi:
OPEN SYMMETRIC KEY DataKey
DECRYPTION BY CERTIFICATE DataCert;
UPDATE Sales.SalesOrderDetail

```
SET EncryptedLineTotal = ENCRYPTBYKEY(  
    KEY_GUID('DataKey'),  
    CAST(LineTotal AS NVARCHAR(50))  
);  
CLOSE SYMMETRIC KEY DataKey;  
GO
```

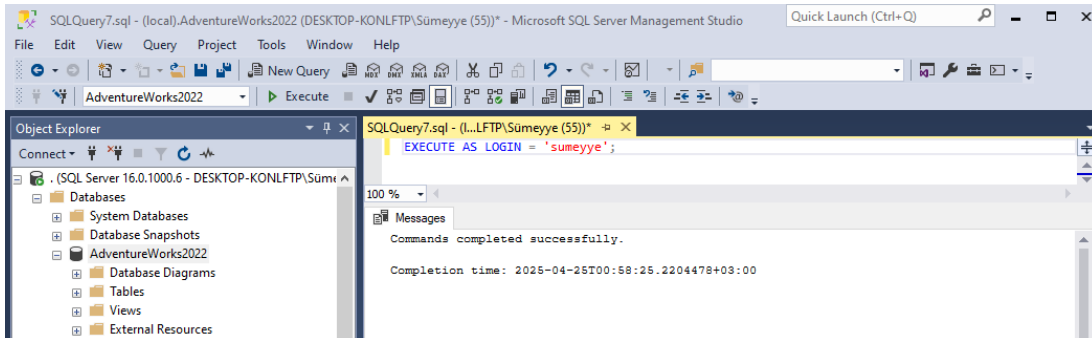


Şekil 23: Veri şifreleme.

4. SQL Injection Testi

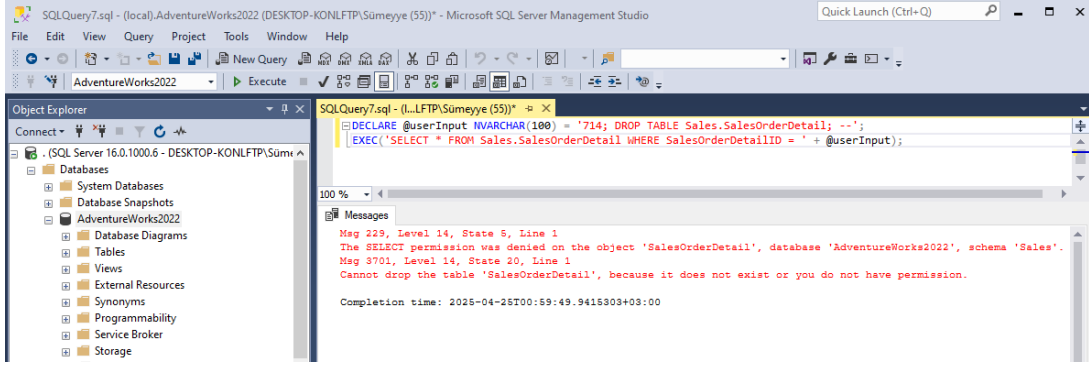
Kullanıcı hakkı kısıtlamasının doğru çalıştığı doğrulandı..

- **EXECUTE AS LOGIN = 'sumeyye';** ile sumeyye kimliğine geçildi.



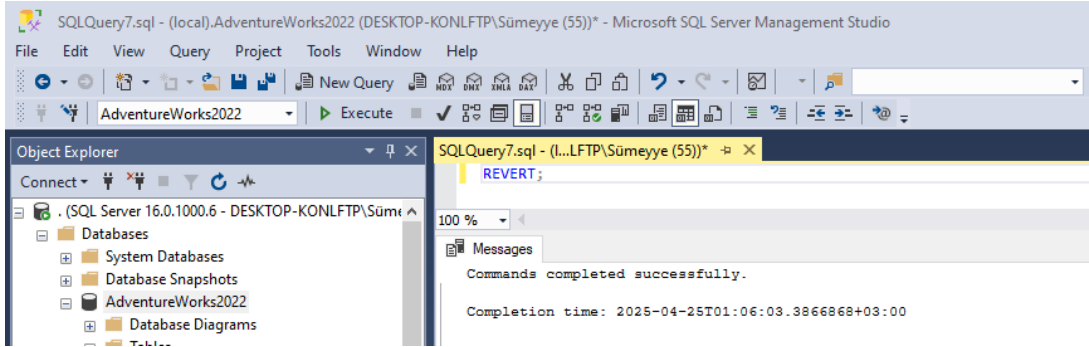
Şekil 24: sumeyye kimliğine geçilmesi.

- Zararlı input simülasyonu:
DECLARE @userInput NVARCHAR(100) = '714; DROP TABLE Sales.SalesOrderDetail; --';
EXEC('SELECT * FROM Sales.SalesOrderDetail WHERE SalesOrderDetailID = ' + @userInput);



Şekil 25: SQL Injection denemesinin reddedilmesi

- Tablonun silinmesi **REVERT**; komutu ile engellendi.

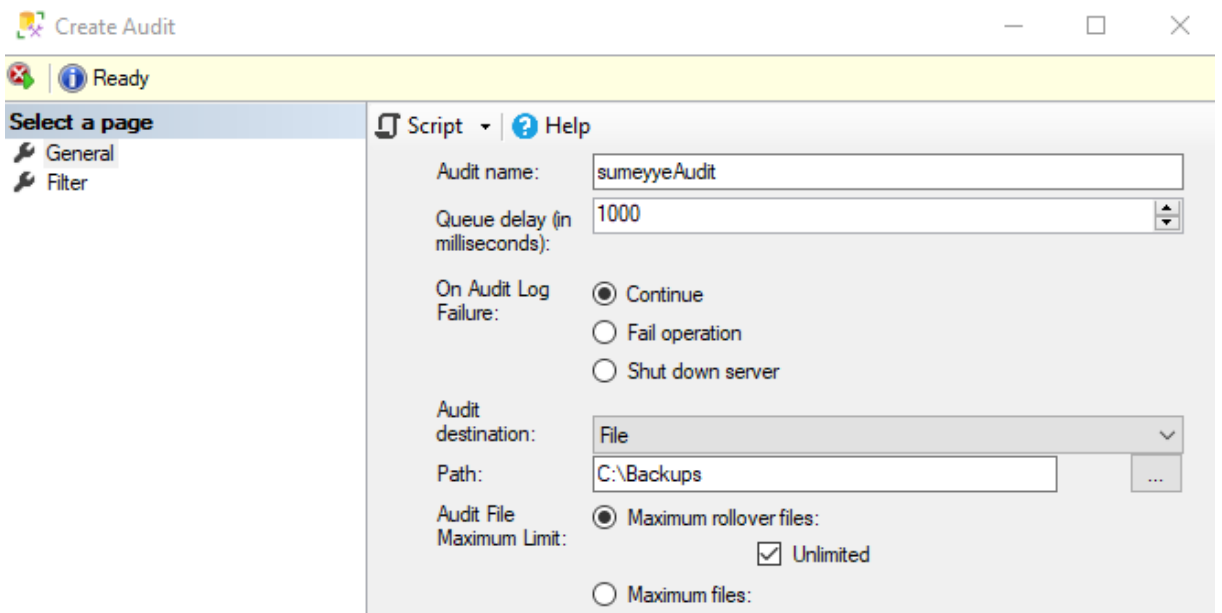


Şekil 26: revert komutu ile tablonun silinmesinin engellenmesi.

5. SQL Server Audit ile Activity Loglama

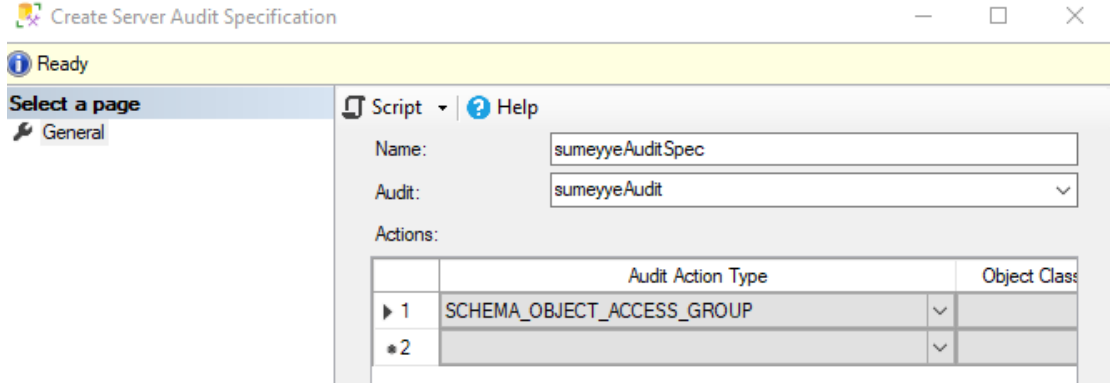
Kullanıcı SELECT işlemleri ve hatalı erişim denemeleri denetlenerek loglandı.

- Server Audit oluşturuldu:
Security → Audits → New Audit... ile **sumeyyeAudit** tanımlandı. **Audit destination:** File seçildi, **File Path:** C:\Backups olarak belirtildi.



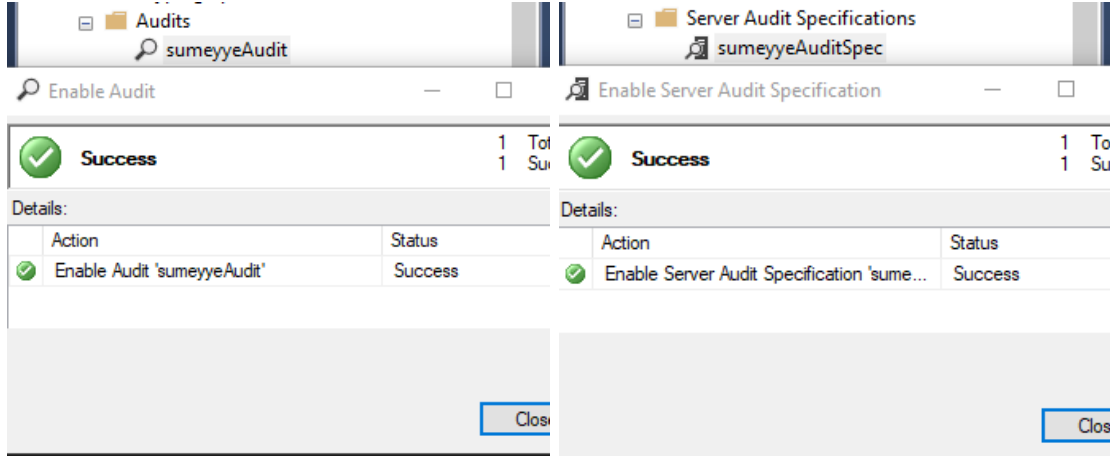
Şekil 27: sumeyyeAudit tanımlanması.

- Server Audit Specification oluşturuldu:
Security → Server Audit Specifications → New Server Audit Specification... ile **SelectAuditSpec** tanımlandı. **Audit:** sumeyyeAuditSpec seçildi. **Action Type:** SCHEMA_OBJECT_ACCESS_GROUP seçildi. **Object Class** alanı boş bırakıldı.



Şekil 28: Server Audit Specification tanımlanması.

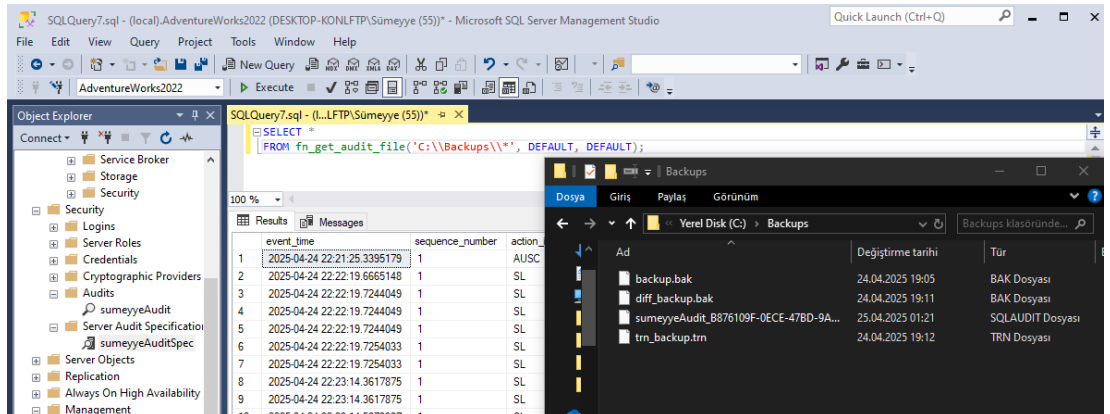
- sumeyyeAudit ve sumeyyeAuditSpec sağ tık → Enable ile etkinleştirildi.



Şekil 29: sumeyyeAudit ve sumeyyeAuditSpec aktifleştirilmesi.

- Audit log kayıtları aşağıdaki komutla görüntülendi:

SELECT *
FROM fn_get_audit_file('C:\\Backups\\', DEFAULT, DEFAULT);



Şekil 30: Audit log kayıtlarının görüntülenmesi.

VERİTABANI YÜK DENGELEME VE DAĞITIK VERİTABANI YAPILARI

AdventureWorks2022 veritabanı için yük dengeleme ve dağıtık veritabanı yapıları oluşturulmuş ve test edilmiştir. Amaç, veritabanı yükünü dağıtmak, veri replikasyonu sağlamak ve failover senaryolarını simüle etmektir.

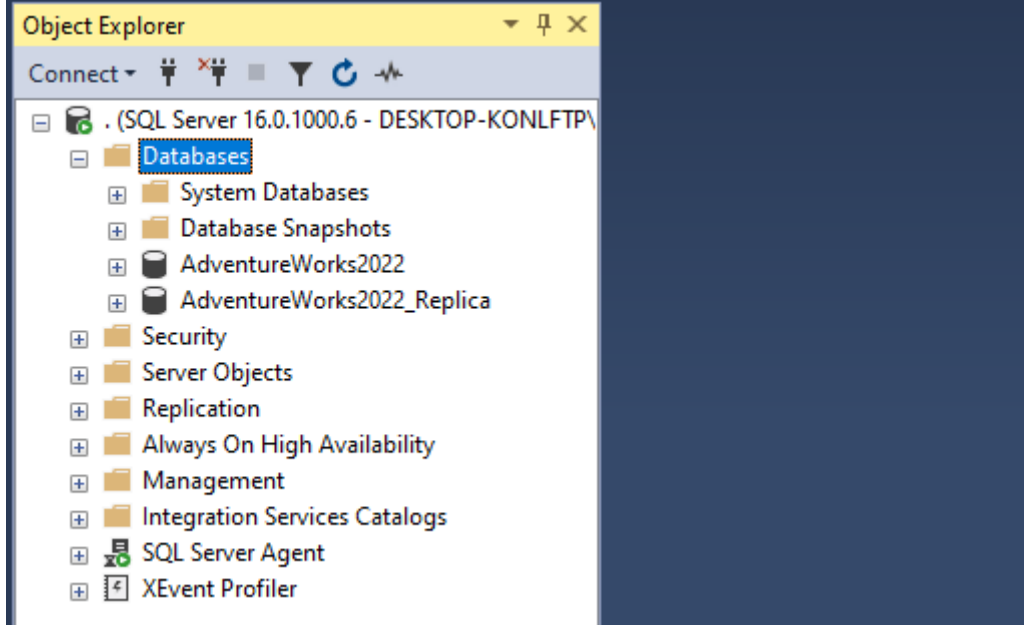
1. Veritabanı Replikasyonu

1.1. Replica Veritabanı Oluşturma

SQL Server Express'te enterprise replication özellikleri desteklenmediği için, manuel veri senkronizasyonu ile replikasyon simüle edilmiştir. AdventureWorks2022_Replica veritabanı oluşturuldu.

```
CREATE DATABASE AdventureWorks2022_Replica;
```

```
GO
```



Şekil 1: Replica veritabanının başarıyla oluşturulması.

1.2. Test Tablosu ve Veri Oluşturma

Primary veritabanında test verileri için TestReplication tablosu oluşturuldu ve örnek veriler eklendi.

```
USE AdventureWorks2022;
```

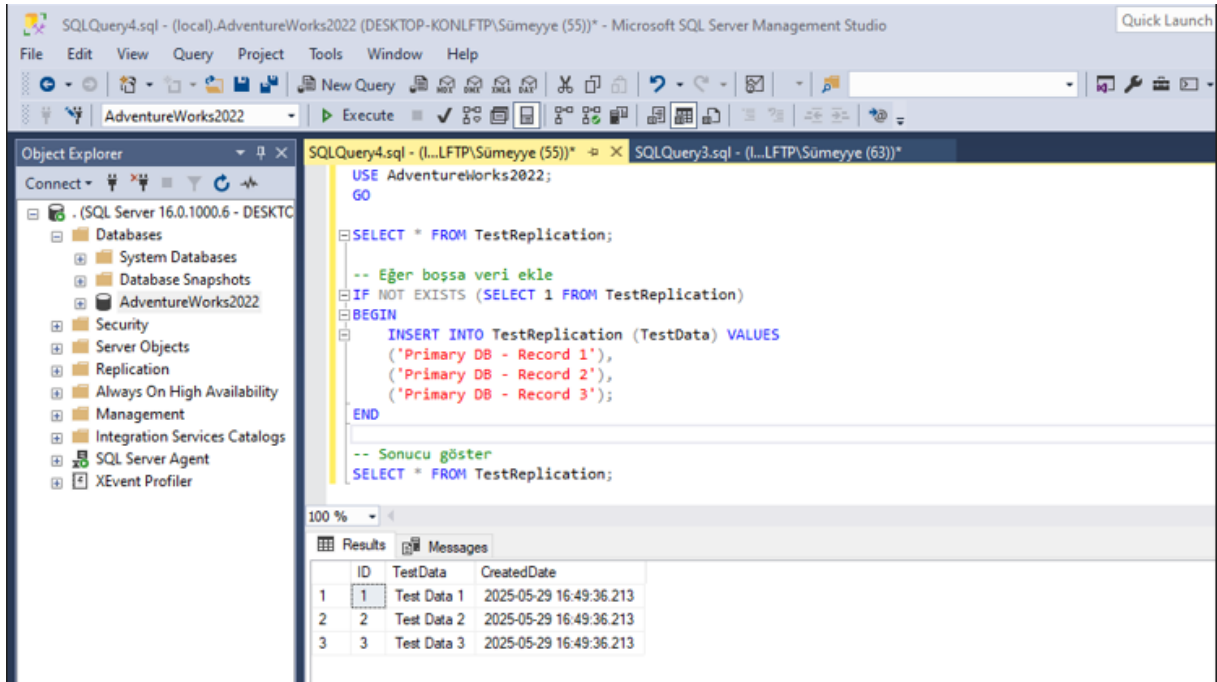
```
GO
```

```
INSERT INTO TestReplication (TestData) VALUES
```

```
('Data 1'),
```

('Data 2'),

('Data 3');



Şekil 2: Primary veritabanında test verilerinin oluşturulması.

1.3. Manuel Veri Senkronizasyonu

Enterprise replication yerine manuel senkronizasyon algoritması geliştirildi. Primary veritabanından Replica veritabanına veri transferi gerçekleştirildi.

USE AdventureWorks2022_Replica;

GO

TRUNCATE TABLE TestReplication;

INSERT INTO TestReplication (ID, TestData, CreatedDate, ServerName, ReplicatedDate)

SELECT

ID,

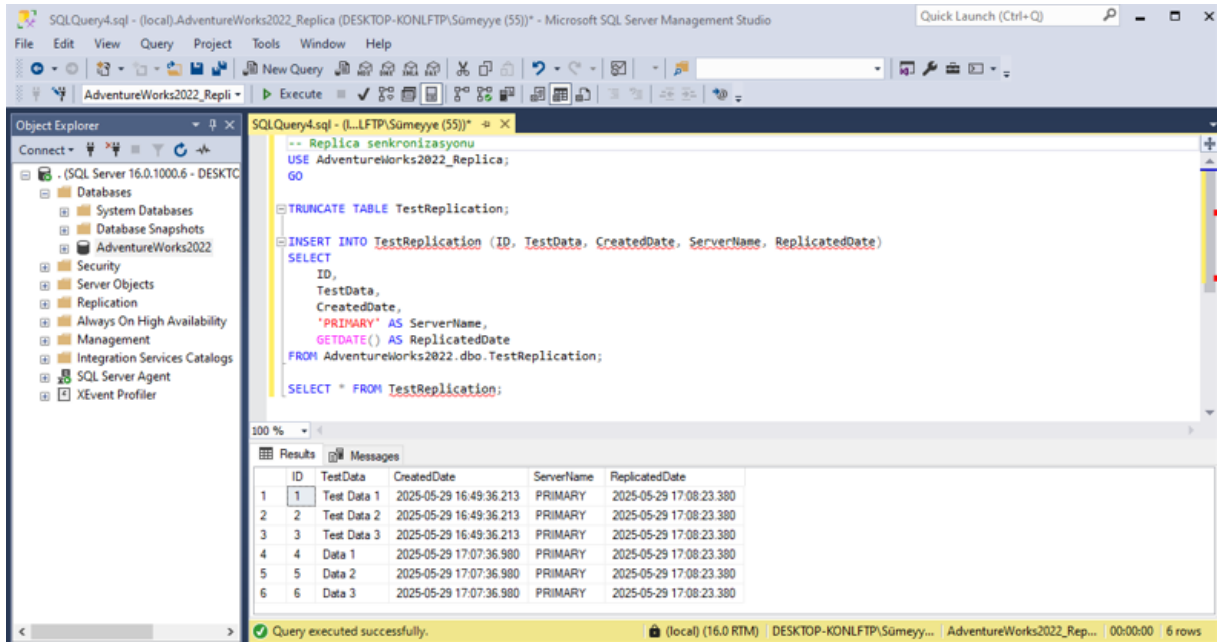
TestData,

CreatedDate,

'PRIMARY' AS ServerName,

GETDATE() AS ReplicatedDate

FROM AdventureWorks2022.dbo.TestReplication;



Şekil 3: Manuel senkronizasyon işleminin başarıyla tamamlanması.

2. Yük Dengeleme Stratejileri

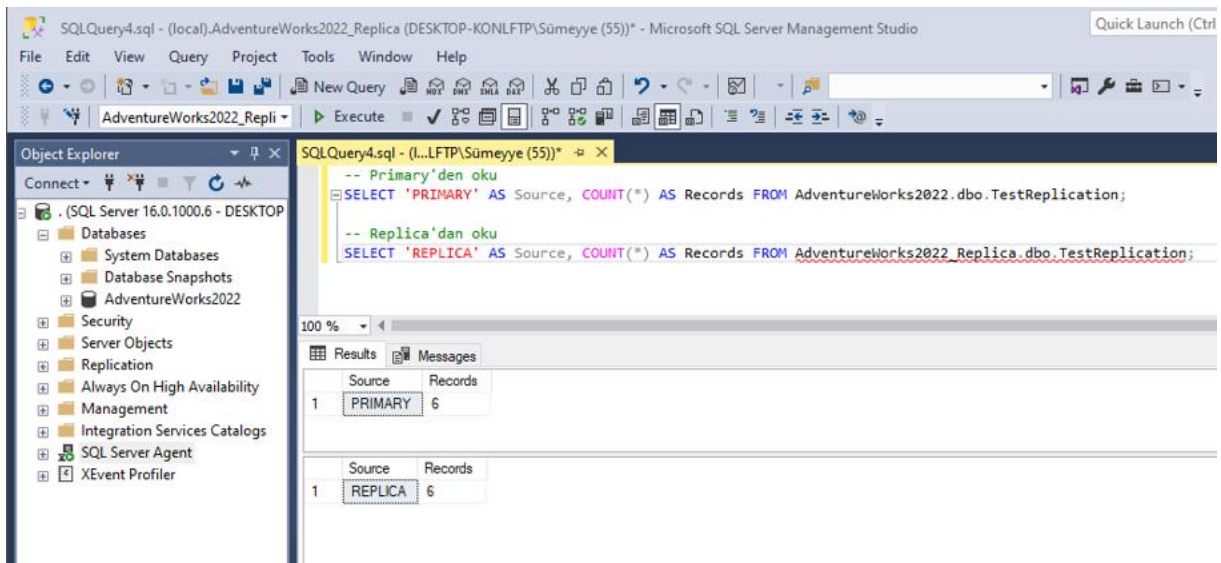
Read operations için yük dengeleme simülasyonu gerçekleştirildi. Hem Primary hem de Replica veritabanından okuma işlemleri test edildi.

SELECT 'PRIMARY' AS Source, COUNT(*) AS Records

FROM AdventureWorks2022.dbo.TestReplication;

SELECT 'REPLICA' AS Source, COUNT(*) AS Records

FROM AdventureWorks2022_Replica.dbo.TestReplication;

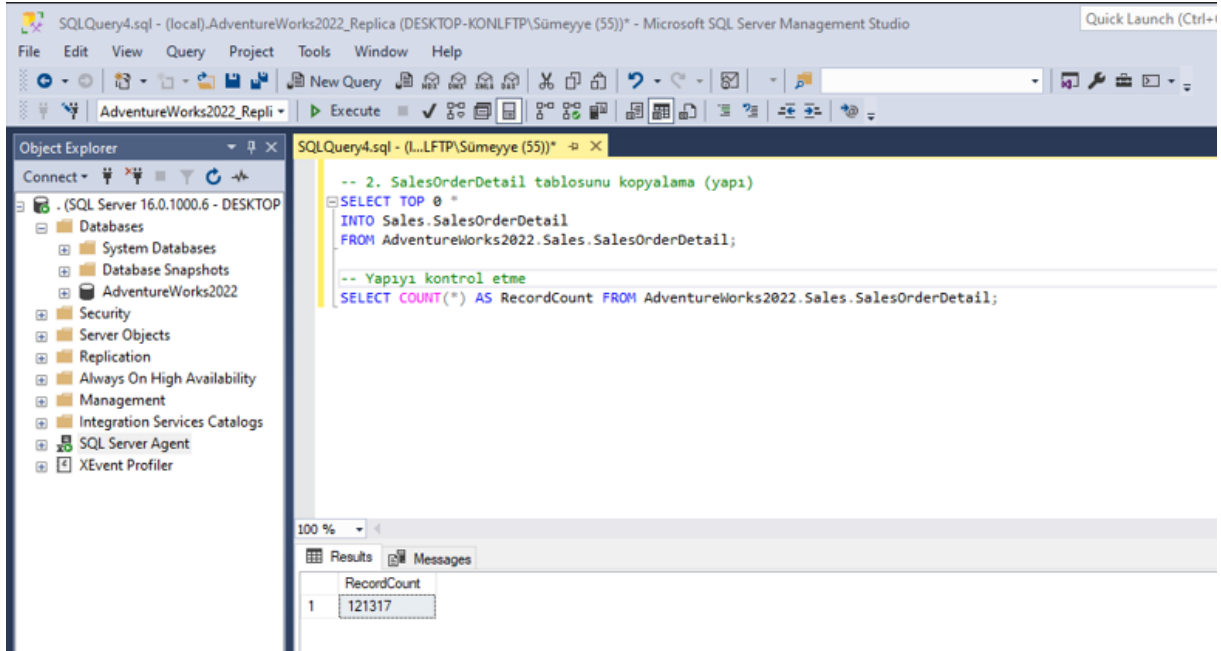


Şekil 4: Load balancing test sonuçları - Primary ve Replica kayıt sayıları.

3. Failover Senaryoları

3.1. Veri Tutarlılığı Kontrolü

Failover senaryolarında veri tutarlılığının korunması test edildi. Primary ve Replica veritabanlarındaki kayıt sayıları karşılaştırıldı ve tutarlılık sağlandığı görüldü.



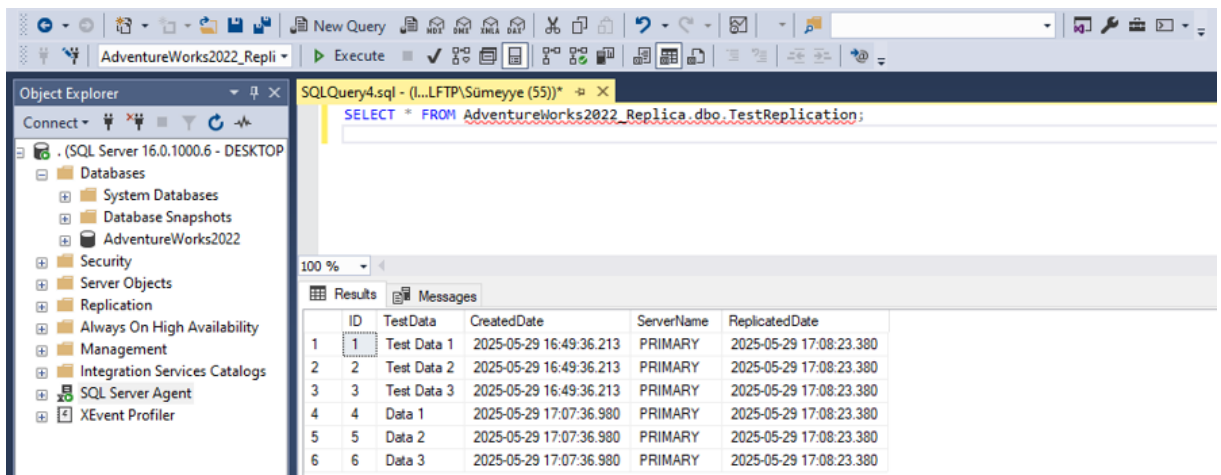
Şekil 5: Failover öncesi veri tutarlılığı kontrolü.

3.2. Read-Only Failover Simülasyonu

Primary veritabanının kullanılamaz durumda olduğu varsayılarak, tüm okuma işlemleri Replica veritabanından gerçekleştirildi.

-- Failover durumunda sadece replica'dan okuma

SELECT * FROM AdventureWorks2022_Replica.dbo.TestReplication;



Şekil 6: Failover durumunda Replica'dan veri okuma.

VERİ TEMİZLEME VE ETL SÜREÇLERİ TASARIMI

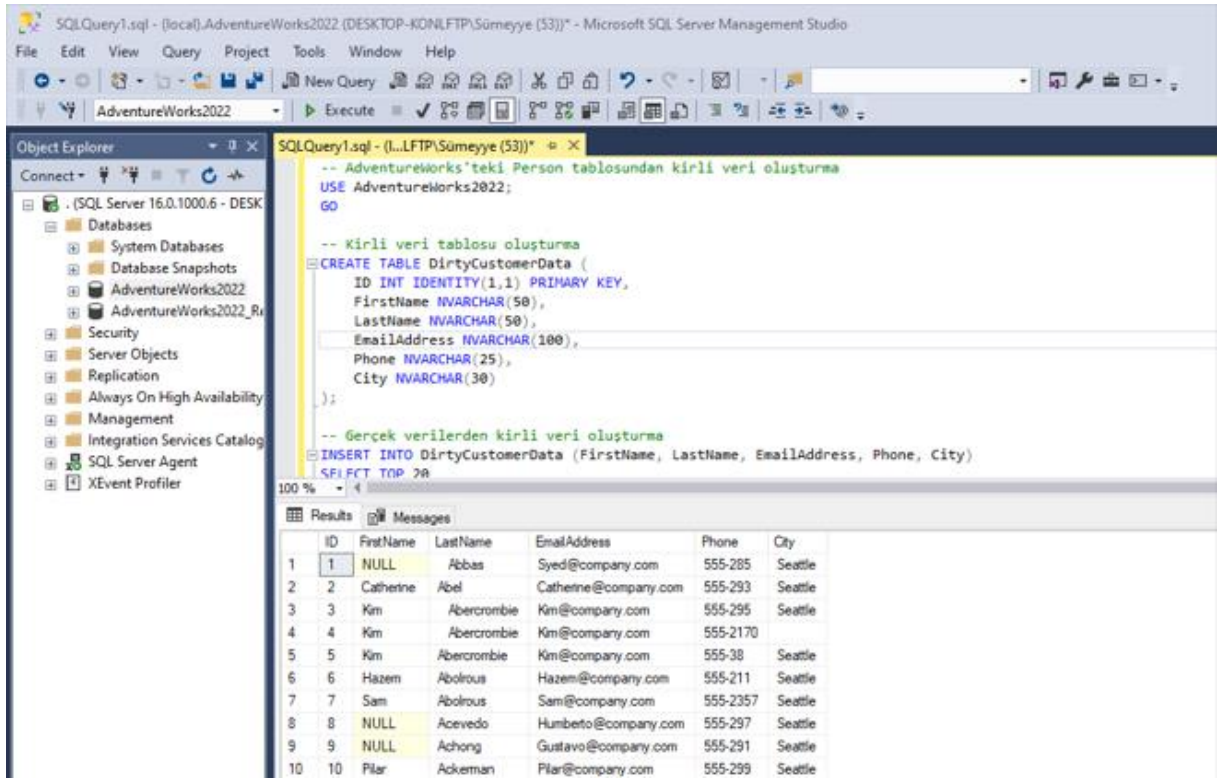
AdventureWorks2022 veritabanı üzerinde veri temizleme ve ETL (Extract, Transform, Load) süreçleri tasarlanmış ve uygulanmıştır. Amaç, gerçek verilerden kirli veri setleri oluşturmak, veri hatalarını tespit etmek, temizlemek ve standart formatlara dönüştürmektir.

1. Veri Temizleme

1.1. Kirli Veri Tablosu Oluşturma

AdventureWorks2022 veritabanındaki Person.Person tablosundan gerçek veriler kullanılarak kirli veri simülasyonu oluşturuldu. Çeşitli veri kalitesi problemleri eklenmiştir.

```
CREATE TABLE DirtyCustomerData (  
    ID INT IDENTITY(1,1) PRIMARY KEY,  
    FirstName NVARCHAR(50),  
    LastName NVARCHAR(50),  
    EmailAddress NVARCHAR(100),  
    Phone NVARCHAR(25),  
    City NVARCHAR(30)  
);  
  
INSERT INTO DirtyCustomerData (FirstName, LastName, EmailAddress, Phone, City)  
SELECT TOP 20  
    CASE WHEN BusinessEntityID % 3 = 0 THEN NULL ELSE FirstName END,  
    CASE WHEN BusinessEntityID % 5 = 0 THEN ' ' + LastName + ' ' ELSE LastName  
    END,  
    CASE WHEN BusinessEntityID % 4 = 0 THEN 'INVALID_EMAIL' ELSE FirstName +  
    '@company.com' END,  
    CASE WHEN BusinessEntityID % 6 = 0 THEN '000-000-0000' ELSE '555-' +  
    CAST(BusinessEntityID AS VARCHAR) END,  
    CASE WHEN BusinessEntityID % 7 = 0 THEN '' ELSE 'Seattle' END  
FROM Person.Person  
WHERE FirstName IS NOT NULL AND LastName IS NOT NULL;
```

Şekil 1: Kirli veri tablosunun oluşturulması ve içeriği.

1.2. Veri Kalitesi Analizi

Kirli veri setinde bulunan veri kalitesi problemleri tespit edilmiştir. NULL değerler, boşluklar, geçersiz email adresleri ve telefon numaraları analiz edilmiştir.

SELECT

'Data Quality Analysis' AS ReportType,

COUNT(*) AS TotalRecords,

SUM(CASE WHEN FirstName IS NULL THEN 1 ELSE 0 END) AS NullFirstNames,

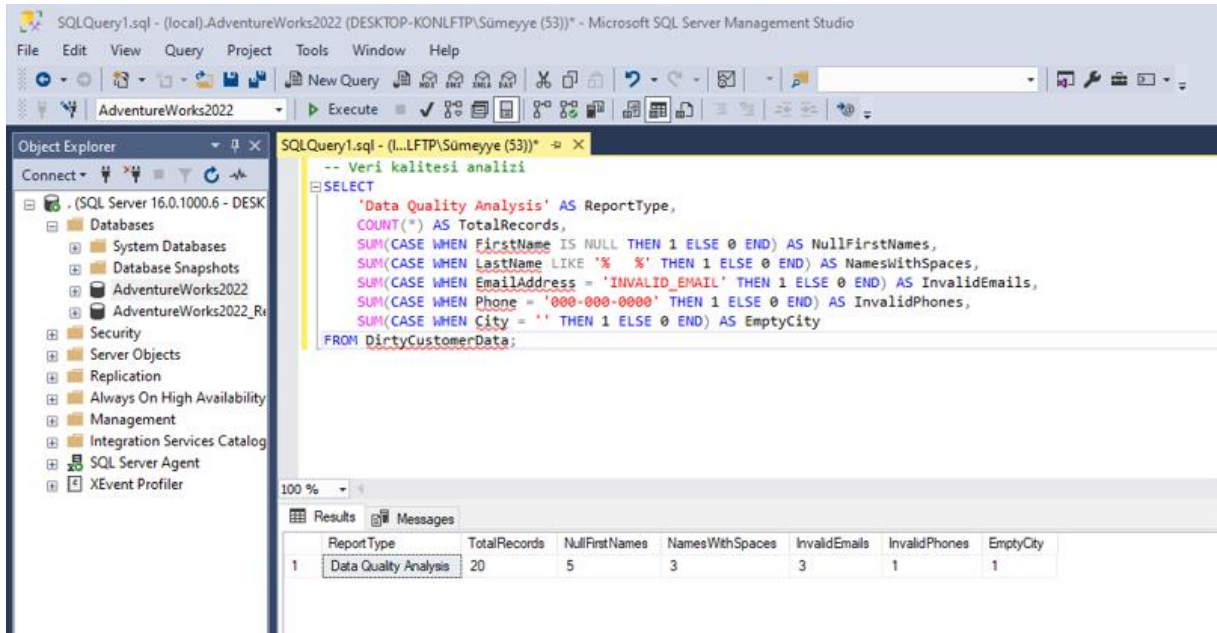
SUM(CASE WHEN LastName LIKE '% %' THEN 1 ELSE 0 END) AS
NamesWithSpaces,

SUM(CASE WHEN EmailAddress = 'INVALID_EMAIL' THEN 1 ELSE 0 END) AS
InvalidEmails,

SUM(CASE WHEN Phone = '000-000-0000' THEN 1 ELSE 0 END) AS InvalidPhones,

SUM(CASE WHEN City = '' THEN 1 ELSE 0 END) AS EmptyCity

FROM DirtyCustomerData;



Şekil 2: Veri kalitesi analiz sonuçları ve problem alanlarının tespiti.

1.3. Veri Temizleme İşlemi

Tespit edilen veri kalitesi problemleri çözülmüş ve temiz veri tablosu oluşturulmuştur. NULL değerler doldurulmuş, boşluklar temizlenmiş ve geçersiz değerler düzeltilmiştir.

```
CREATE TABLE CleanCustomerData (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    FirstName NVARCHAR(50),
    LastName NVARCHAR(50),
    EmailAddress NVARCHAR(100),
    Phone NVARCHAR(25),
    City NVARCHAR(30),
    CleanedDate DATETIME DEFAULT GETDATE()
);
```

```
INSERT INTO CleanCustomerData (FirstName, LastName, EmailAddress, Phone, City)
SELECT
    CASE
        WHEN FirstName IS NULL THEN 'Unknown'
```

```

ELSE LTRIM(RTRIM(FirstName))

END AS FirstName,

LTRIM(RTRIM(LastName)) AS LastName,

CASE

    WHEN EmailAddress = 'INVALID_EMAIL' THEN FirstName +
'.cleaned@company.com'

    ELSE EmailAddress

END AS EmailAddress,

CASE

    WHEN Phone = '000-000-0000' THEN '555-CLEANED'

    ELSE Phone

END AS Phone,

CASE

    WHEN City = '' THEN 'Unknown City'

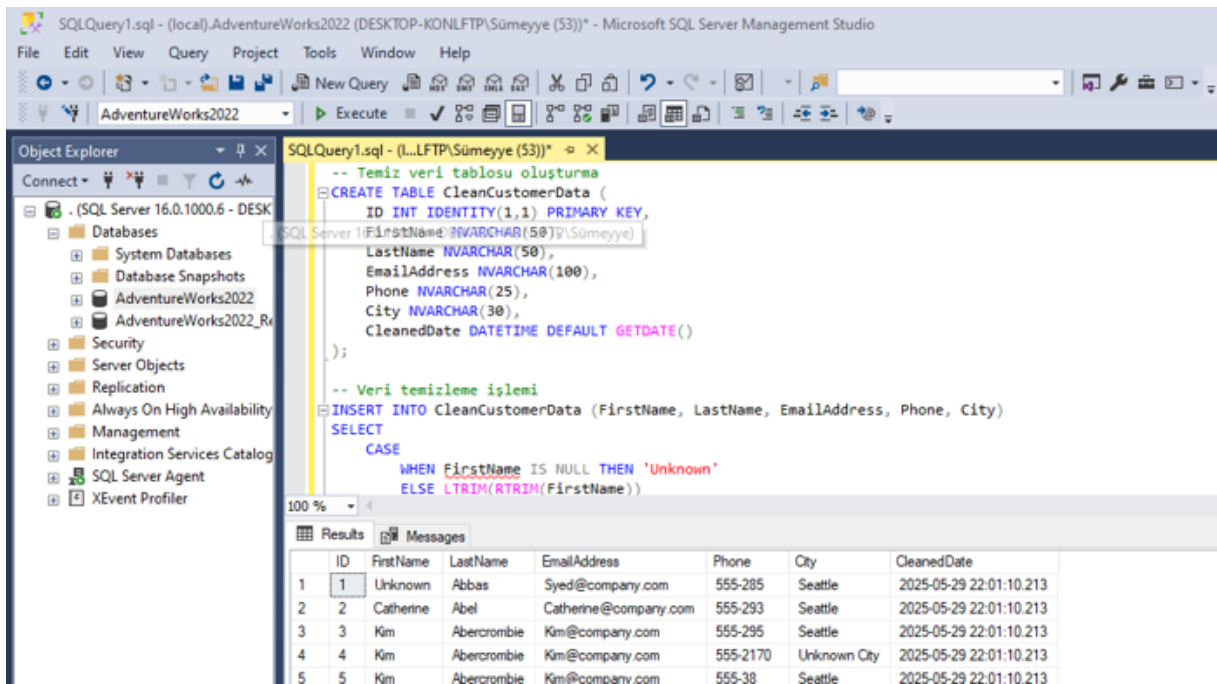
    ELSE City

END AS City

FROM DirtyCustomerData

WHERE LastName IS NOT NULL;

```



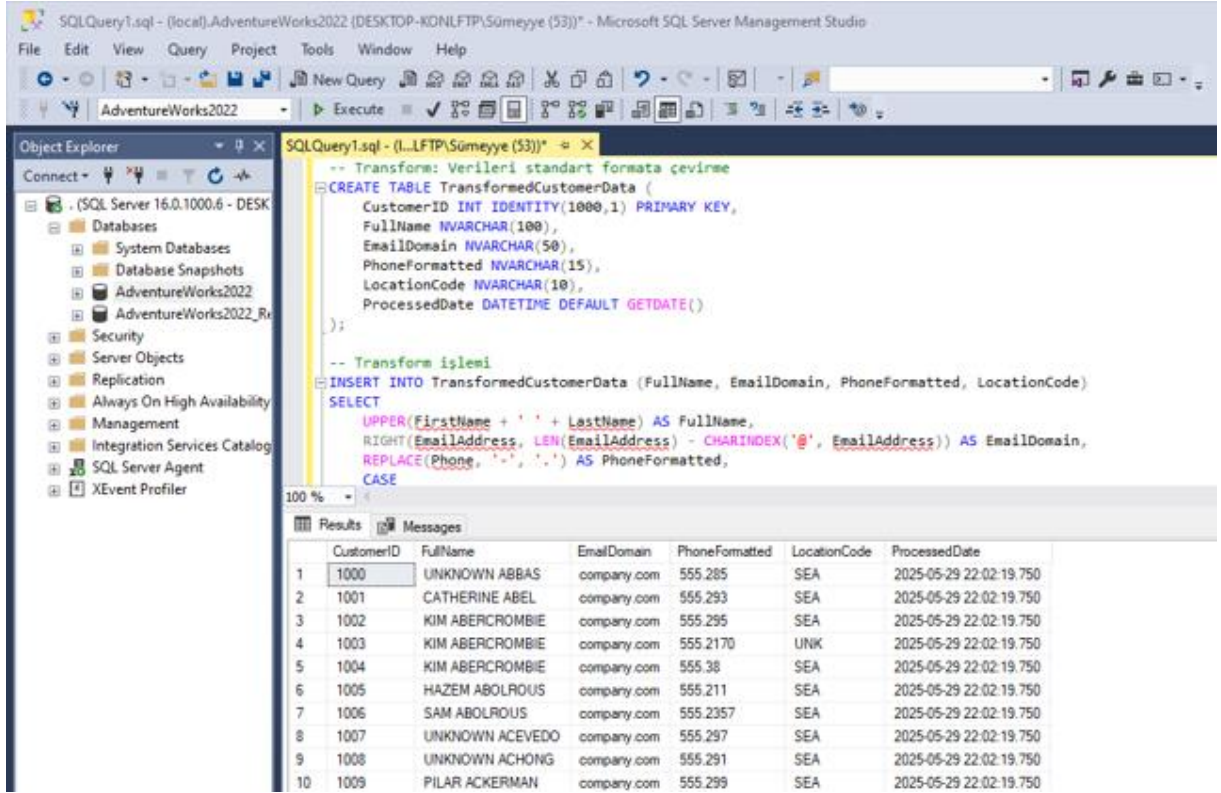
Şekil 3: Veri temizleme işlemi sonucunda elde edilen temiz veriler.

2. Veri Dönüştürme (Transform)

2.1. Veri Standardizasyonu

Temizlenmiş veriler iş gereksinimlerine uygun standart formatlara dönüştürülmüştür. Tam isim birleştirme, email domain çıkarma ve konum kodlama işlemleri yapılmıştır.

```
CREATE TABLE TransformedCustomerData (  
    CustomerID INT IDENTITY(1000,1) PRIMARY KEY,  
    FullName NVARCHAR(100),  
    EmailDomain NVARCHAR(50),  
    PhoneFormatted NVARCHAR(15),  
    LocationCode NVARCHAR(10),  
    ProcessedDate DATETIME DEFAULT GETDATE()  
);  
  
INSERT INTO TransformedCustomerData (FullName, EmailDomain, PhoneFormatted,  
LocationCode)  
SELECT  
    UPPER(FirstName + ' ' + LastName) AS FullName,  
    RIGHT(EmailAddress, LEN(EmailAddress) - CHARINDEX('@', EmailAddress)) AS  
EmailDomain,  
    REPLACE(Phone, '-', '.') AS PhoneFormatted,  
    CASE  
        WHEN City = 'Seattle' THEN 'SEA'  
        WHEN City = 'Unknown City' THEN 'UNK'  
        ELSE 'OTH'  
    END AS LocationCode  
FROM CleanCustomerData;
```



Şekil 4: Transform işlemi sonucunda standardize edilmiş veriler.

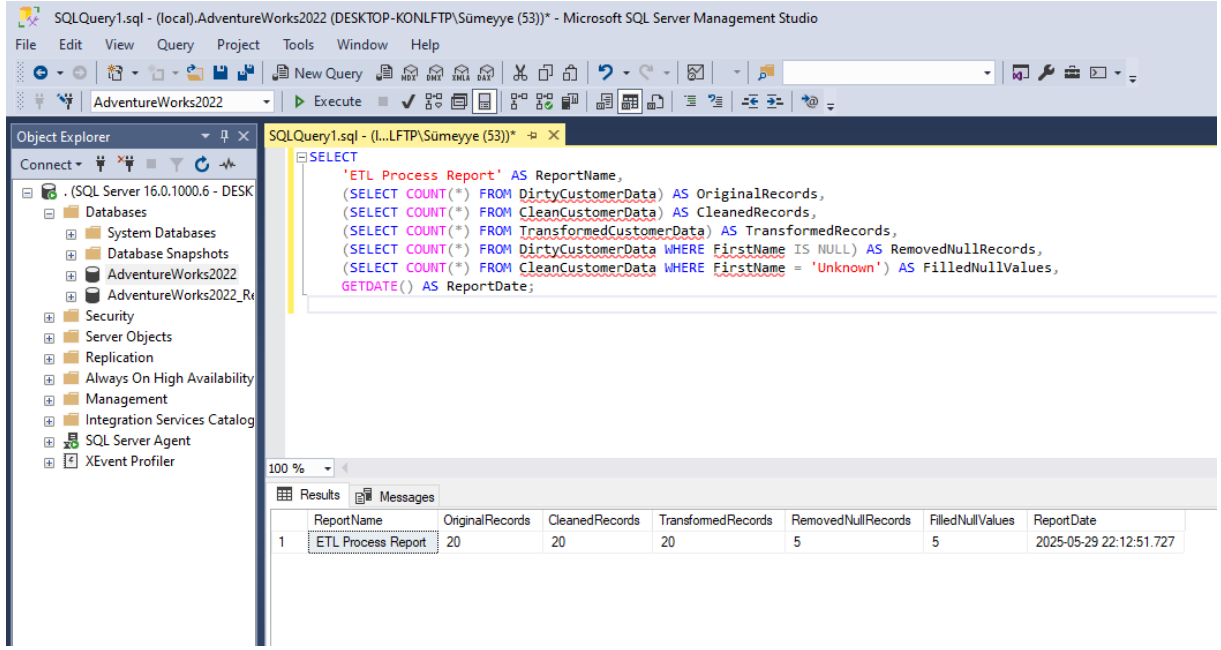
3. Veri Kalitesi Raporları

3.1. ETL Süreç Raporu

ETL sürecinin her aşamasında veri kalitesi metrikleri ölçülmüş ve kapsamlı rapor oluşturulmuştur.

SELECT

'ETL Process Report' AS ReportName,
 (SELECT COUNT(*) FROM DirtyCustomerData) AS OriginalRecords,
 (SELECT COUNT(*) FROM CleanCustomerData) AS CleanedRecords,
 (SELECT COUNT(*) FROM TransformedCustomerData) AS TransformedRecords,
 (SELECT COUNT(*) FROM DirtyCustomerData WHERE FirstName IS NULL) AS
 RemovedNullRecords,
 (SELECT COUNT(*) FROM CleanCustomerData WHERE FirstName = 'Unknown') AS
 FilledNullValues,
 GETDATE() AS ReportDate;



Şekil 6: ETL süreç kalite raporu ve veri akış metrikleri.

3.2. Önce/Sonra Karşılaştırması

Veri temizleme öncesi ve sonrası durumu karşılaştırılarak işlem başarısı ölçülmüştür.

SELECT

'Before Cleaning' AS Stage,

AVG(LEN(FirstName)) AS AvgFirstNameLength,

COUNT(DISTINCT EmailAddress) AS UniqueEmails

FROM DirtyCustomerData

WHERE FirstName IS NOT NULL

UNION ALL

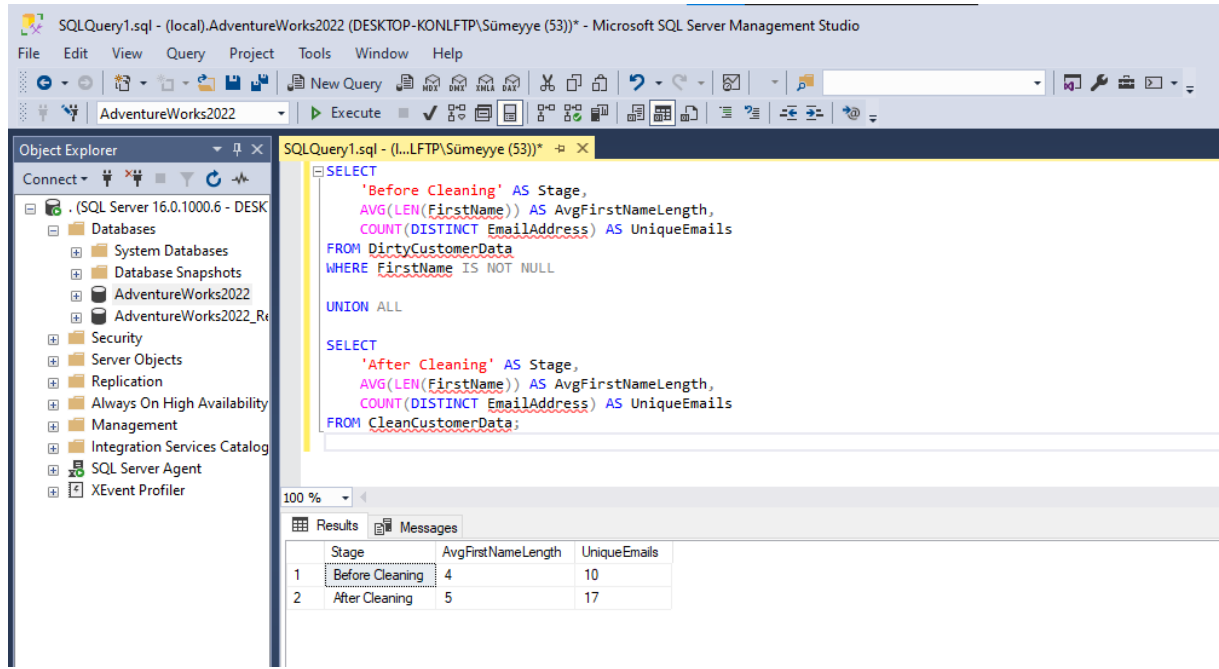
SELECT

'After Cleaning' AS Stage,

AVG(LEN(FirstName)) AS AvgFirstNameLength,

COUNT(DISTINCT EmailAddress) AS UniqueEmails

FROM CleanCustomerData;



Şekil 7: Temizleme öncesi ve sonrası veri kalitesi karşılaştırması.

VERİTABANI YÜKSELTME VE SÜRÜM YÖNETİMİ

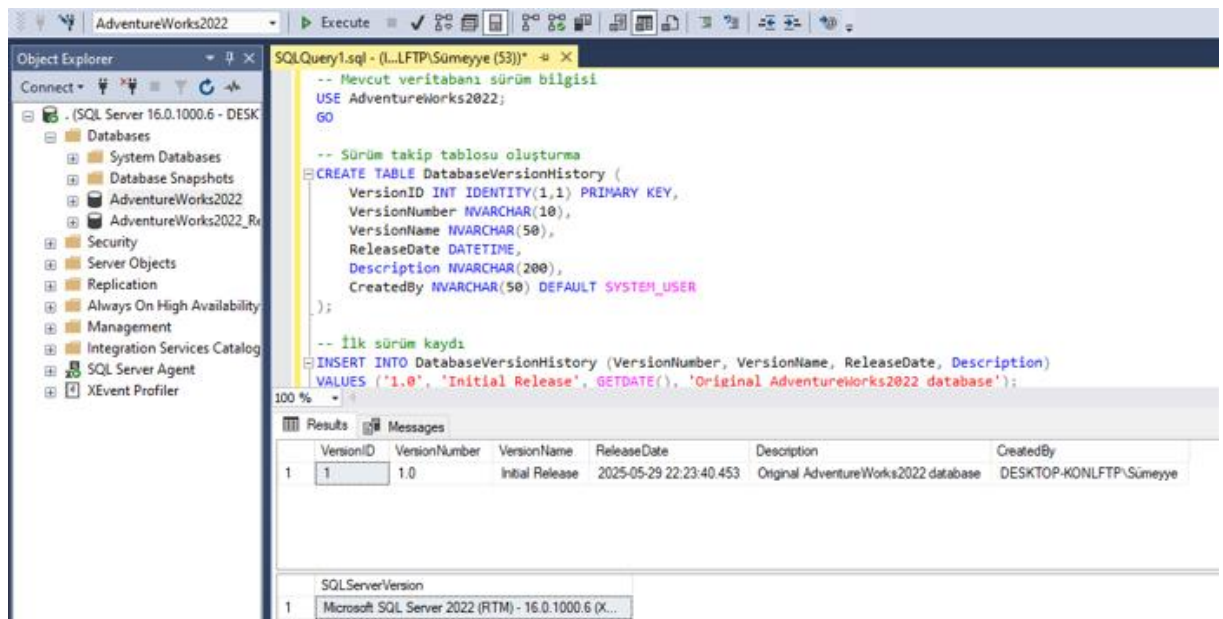
AdventureWorks2022 veritabanı için sürüm yönetimi ve yükseltme süreçleri tasarlanmış ve uygulanmıştır. Amaç, veritabanı yapısındaki değişiklikleri izlemek, güvenli yükseltme planları oluşturmak ve gerektiğinde geri dönüş işlemlerini gerçekleştirebilmektir.

1. Veritabanı Yükseltme Planı

1.1. Sürüm Takip Sistemi Oluşturma

Veritabanı sürümlerini takip etmek için DatabaseVersionHistory tablosu oluşturulmuş ve ilk sürüm kaydı eklenmiştir.

```
CREATE TABLE DatabaseVersionHistory (  
    VersionID INT IDENTITY(1,1) PRIMARY KEY,  
    VersionNumber NVARCHAR(20),  
    VersionName NVARCHAR(50),  
    ReleaseDate DATETIME,  
    Description NVARCHAR(200),  
    CreatedBy NVARCHAR(50) DEFAULT SYSTEM_USER  
);  
  
INSERT INTO DatabaseVersionHistory (VersionNumber, VersionName, ReleaseDate,  
Description)  
VALUES ('1.0', 'Initial Release', GETDATE(), 'Original AdventureWorks2022 database');
```



Şekil 1: İlk sürüm kaydının oluşturulması ve SQL Server versiyon bilgileri.

1.2. Yükseltme Öncesi Güvenlik Yedeklemesi

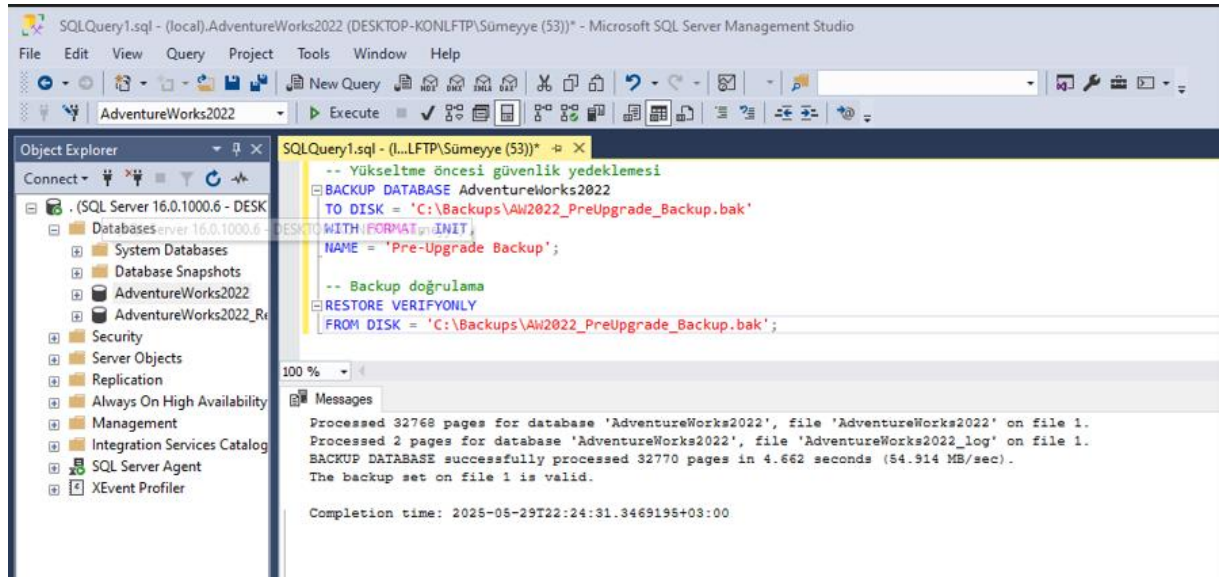
Yükseltme işlemleri öncesinde veri güvenliği için tam yedekleme alınmıştır.

BACKUP DATABASE AdventureWorks2022

TO DISK = 'C:\Backups\AW2022_PreUpgrade_Backup.bak'

WITH FORMAT, INIT,

NAME = 'Pre-Upgrade Backup';



Şekil 2: Yükseltme öncesi backup işleminin başarıyla tamamlanması.

2. Sürüm Yönetimi

2.1. Şema Değişikliklerini İzleme (DDL Triggers)

Veritabanı yapısındaki değişiklikleri otomatik olarak takip etmek için DDL Trigger oluşturulmuştur.

CREATE TABLE SchemaChangeLog (

LogID INT IDENTITY(1,1) PRIMARY KEY,

EventType NVARCHAR(50),

ObjectName NVARCHAR(100),

SQLCommand NVARCHAR(MAX),

ChangeDate DATETIME DEFAULT GETDATE(),

ChangedBy NVARCHAR(50)

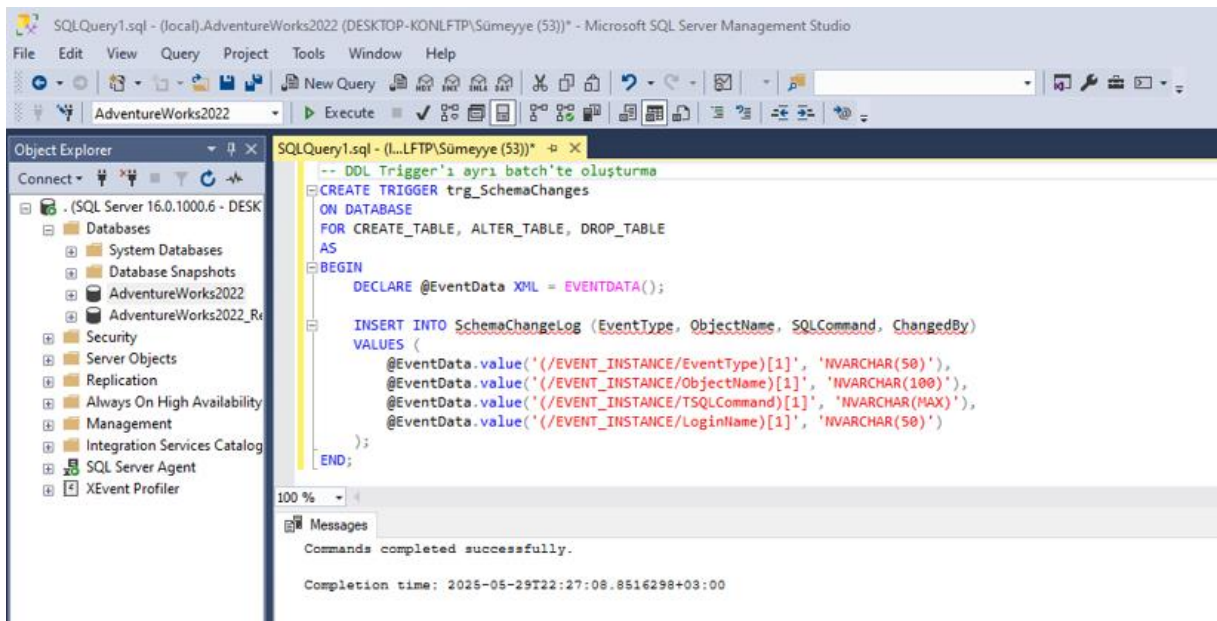

```

);

CREATE TRIGGER trg_SchemaChanges
ON DATABASE
FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
AS
BEGIN
    DECLARE @EventData XML = EVENTDATA();

    INSERT INTO SchemaChangeLog (EventType, ObjectName, SQLCommand,
    ChangedBy)
    VALUES (
        @EventData.value('/EVENT_INSTANCE/EventType)[1]', 'NVARCHAR(50)'),
        @EventData.value('/EVENT_INSTANCE/ObjectName)[1]', 'NVARCHAR(100)'),
        @EventData.value('/EVENT_INSTANCE/TSQLCommand)[1]',
        'NVARCHAR(MAX)'),
        @EventData.value('/EVENT_INSTANCE/LoginName)[1]', 'NVARCHAR(50)')
    );
END;

```



Şekil 3: DDL Trigger'ın başarıyla oluşturulması.

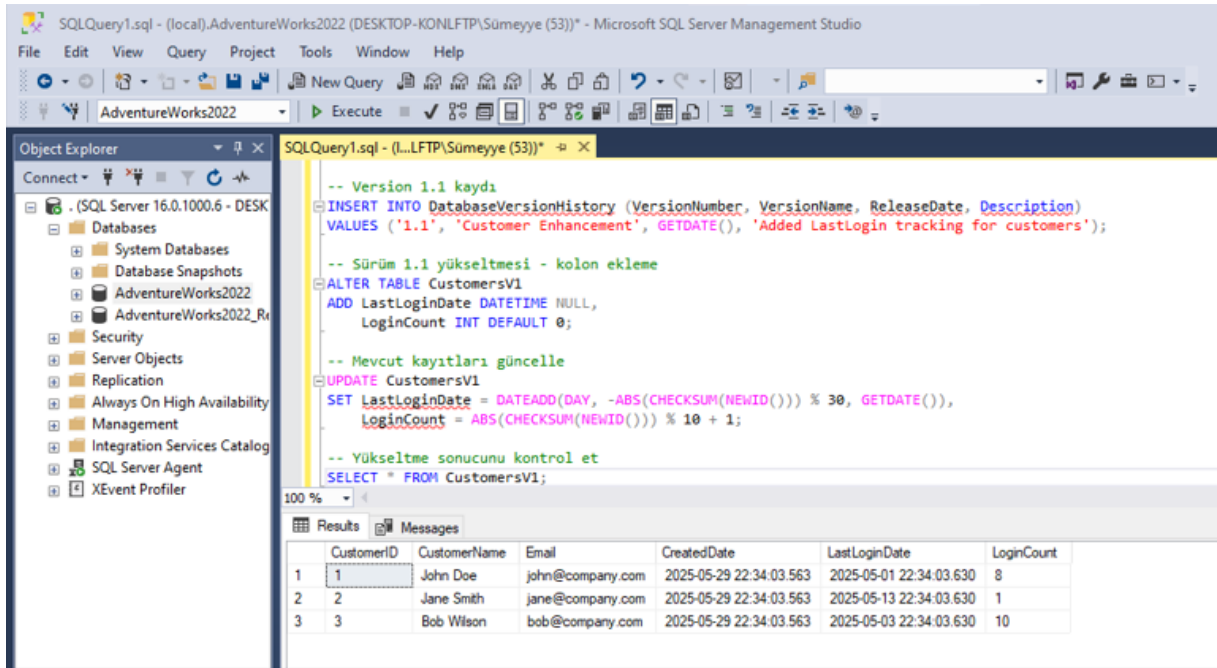
2.2. Sürüm 1.1 Yükseltme İşlemi

Müşteri verilerini geliştirmek amacıyla yeni özellikler eklenmiştir. Test amaçlı CustomersV1 tablosu oluşturularak yükseltme simüle edilmiştir.

```
CREATE TABLE CustomersV1 (  
    CustomerID INT IDENTITY(1,1) PRIMARY KEY,  
    CustomerName NVARCHAR(100),  
    Email NVARCHAR(100),  
    CreatedDate DATETIME DEFAULT GETDATE()  
);
```

```
INSERT INTO DatabaseVersionHistory (VersionNumber, VersionName, ReleaseDate,  
Description)  
VALUES ('1.1', 'Customer Enhancement', GETDATE(), 'Added LastLogin tracking for  
customers');
```

```
ALTER TABLE CustomersV1  
ADD LastLoginDate DATETIME NULL,  
LoginCount INT DEFAULT 0;
```



Şekil 4: Sürüm 1.1 yükseltme işleminin başarıyla tamamlanması.

3. Test ve Geri Dönüş Planı

3.1. Sürüm 1.2 Performans Yükseltmesi

Sistem performansını artırmak amacıyla indeks optimizasyonları yapılmıştır.

```
INSERT INTO DatabaseVersionHistory (VersionNumber, VersionName, ReleaseDate, Description)
```

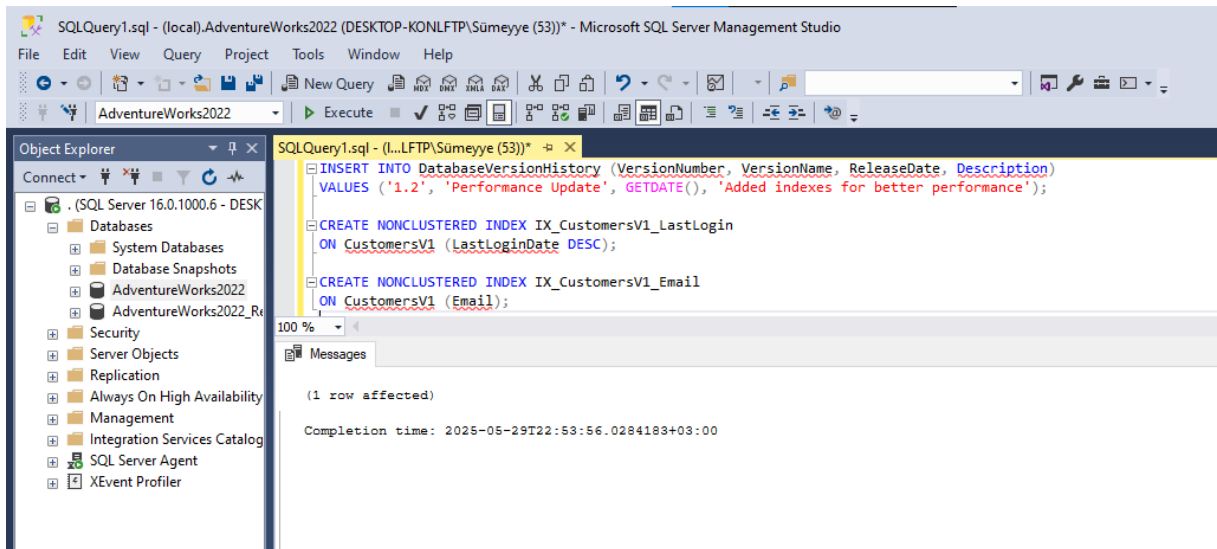
```
VALUES ('1.2', 'Performance Update', GETDATE(), 'Added indexes for better performance');
```

```
CREATE NONCLUSTERED INDEX IX_CustomersV1_LastLogin
```

```
ON CustomersV1 (LastLoginDate DESC);
```

```
CREATE NONCLUSTERED INDEX IX_CustomersV1_Email
```

```
ON CustomersV1 (Email);
```



Şekil 5: İndeks oluşturma ve kontrol işlemlerinin sonuçları.

3.2. Rollback Senaryosu Uygulaması

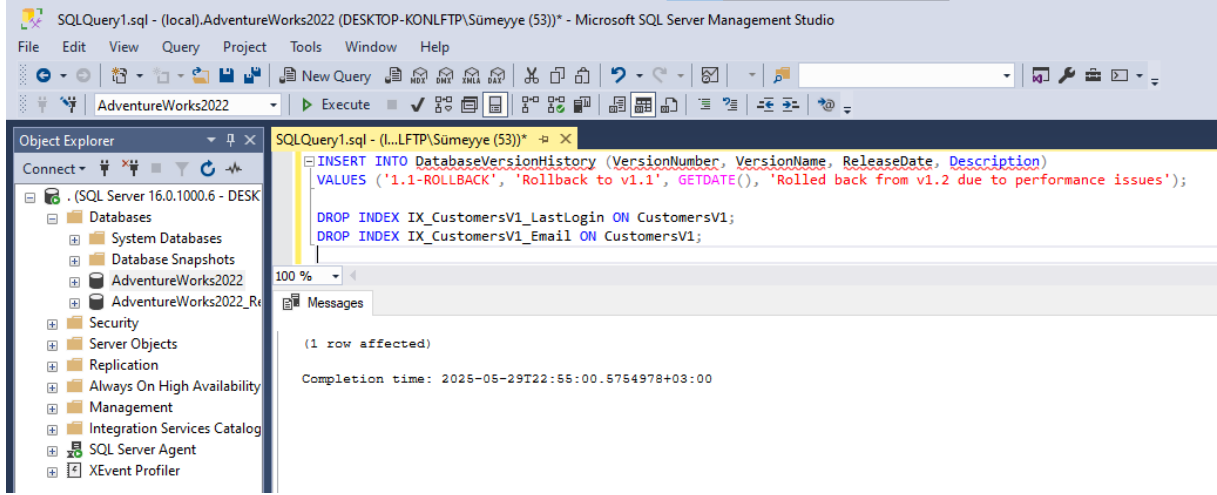
Performans sorunları nedeniyle sürüm 1.2'den sürüm 1.1'e geri dönüş işlemi gerçekleştirilmiştir.

```
INSERT INTO DatabaseVersionHistory (VersionNumber, VersionName, ReleaseDate, Description)
```

```
VALUES ('1.1-ROLLBACK', 'Rollback to v1.1', GETDATE(), 'Rolled back from v1.2 due to performance issues');
```

DROP INDEX IX_CustomersV1_LastLogin ON CustomersV1;

DROP INDEX IX_CustomersV1_Email ON CustomersV1;

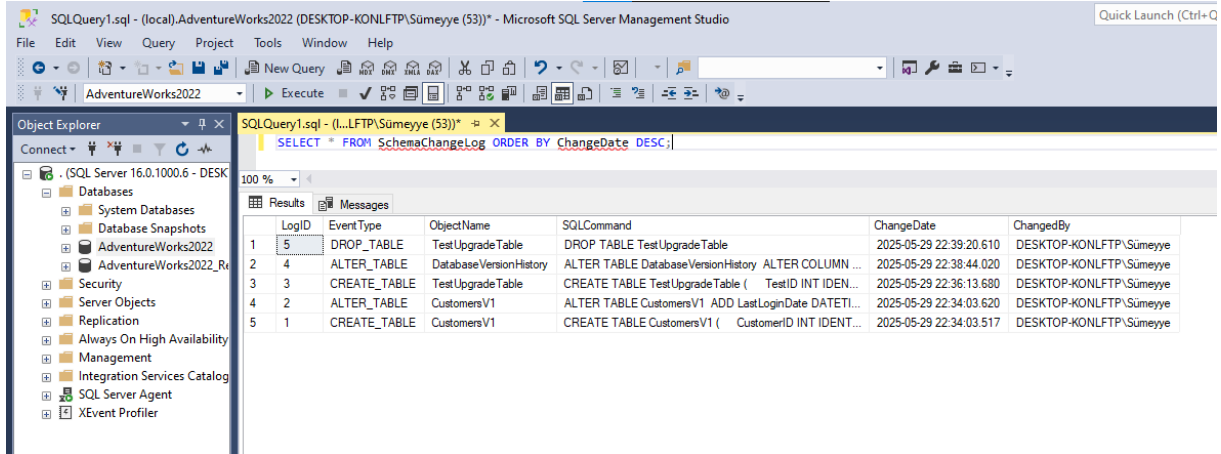


Şekil 6: Rollback işlemi sonrası durum ve sürüm geçmişi.

3.3. Şema Değişiklik Takibi

DDL Trigger'ın çalışması test edilmiş ve şema değişiklikleri başarıyla loglanmıştır.

SELECT * FROM SchemaChangeLog ORDER BY ChangeDate DESC;



Şekil 7: Şema değişiklik loglarının izlenmesi ve kayıt artışı.

VERİTABANI YEDEKLEME VE OTOMASYON ÇALIŞMASI

AdventureWorks2022 veritabanı için otomatik yedekleme sistemi kurulmuş ve test edilmiştir. Amaç, veritabanı yedekleme işlemlerini otomatikleştirmek, düzenli yedeklerin alındığını doğrulamak ve yedekleme süreçlerini izlemektir.

1. SQL Server Agent ile Yedekleme Otomasyonu

1.1. Otomatik Yedekleme Job'ı Oluşturma

SQL Server Agent kullanarak otomatik yedekleme job'ı oluşturulmuştur. AutoBackup_Sumeyye adıyla tanımlanan job, günlük olarak çalışacak şekilde yapılandırılmıştır.

Job Konfigürasyonu:

- **Name:** AutoBackup_Sumeyye
- **Description:** Automatic backup job for AdventureWorks2022
- **Step Name:** FullBackup
- **Command:**

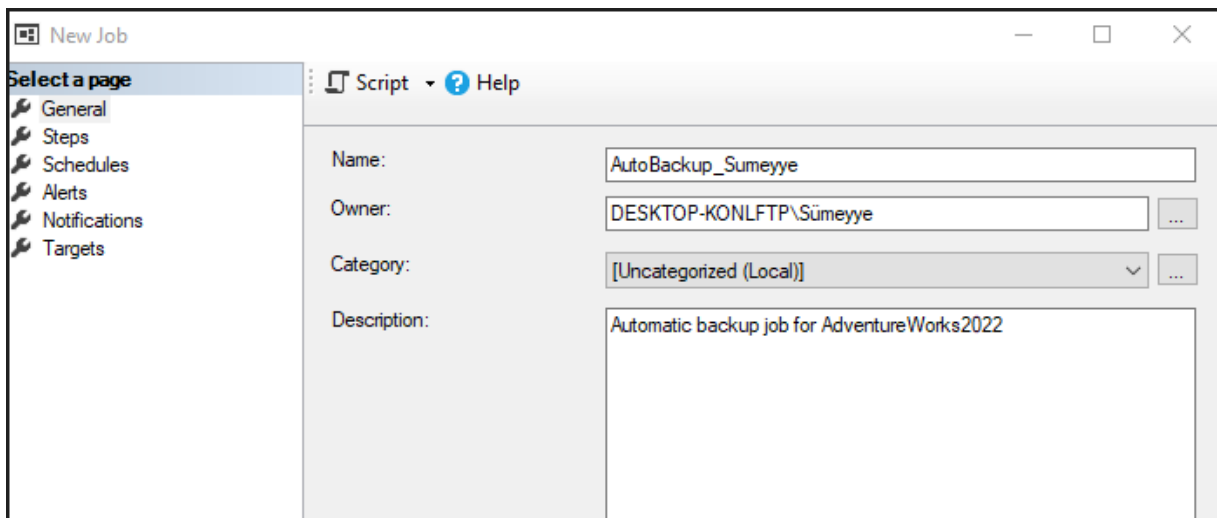
BACKUP DATABASE AdventureWorks2022

TO DISK = 'C:\Backups\AW2022_Auto_' +

FORMAT(GETDATE(), 'yyyyMMdd_HH:mm:ss') + '.bak'

WITH FORMAT, INIT,

NAME = 'AdventureWorks2022 Auto Backup';



The screenshot shows the 'New Job' dialog box in SQL Server Enterprise Manager. The 'General' tab is active. The 'Name' field contains 'AutoBackup_Sumeyye'. The 'Owner' field contains 'DESKTOP-KONLFTP\Sümeyye'. The 'Category' dropdown is set to '[Uncategorized (Local)]'. The 'Description' text box contains 'Automatic backup job for AdventureWorks2022'. The left sidebar shows a tree view with 'General', 'Steps', 'Schedules', 'Alerts', 'Notifications', and 'Targets'.

Şekil 1: SQL Server Agent Job oluşturma ekranları (Job)

New Job Step

Select a page: General, Advanced

Script Help

Step name: Sumeyye_FullBackup

Type: Transact-SQL script (T-SQL)

Run as:

Database: AdventureWorks2022

Command: BACKUP DATABASE AdventureWorks2022 TO DISK = 'C:\Backups\AW2022_Auto_' + FORMAT(GETDATE(), 'yyyyMMdd_HH:mm:ss') + '.bak' WITH FORMAT, INIT, NAME = 'AdventureWorks2022 Auto Backup';

Open... Select All Copy

Şekil 2: SQL Server Agent Job oluşturma ekranları

1.2. Schedule Konfigürasyonu

Job'ın düzenli çalışması için schedule ayarları yapılmıştır:

- **Schedule Name:** DailyBackup
- **Frequency:** Daily (Günlük)
- **Time:** 02:00 (Her gün saat 02:00)

New Job Schedule

Name: Sumeyye_DailyBackup Jobs in Schedule

Schedule type: Recurring Enabled

One-time occurrence

Date: 29.05.2025 Time: 23:00:53

Frequency

Occurs: Daily

Recurrs every: 1 day(s)

Daily frequency

Occurs once at: 02:00:00

Occurs every: 1 hour(s) Starting at: 00:00:00 Ending at: 23:59:59

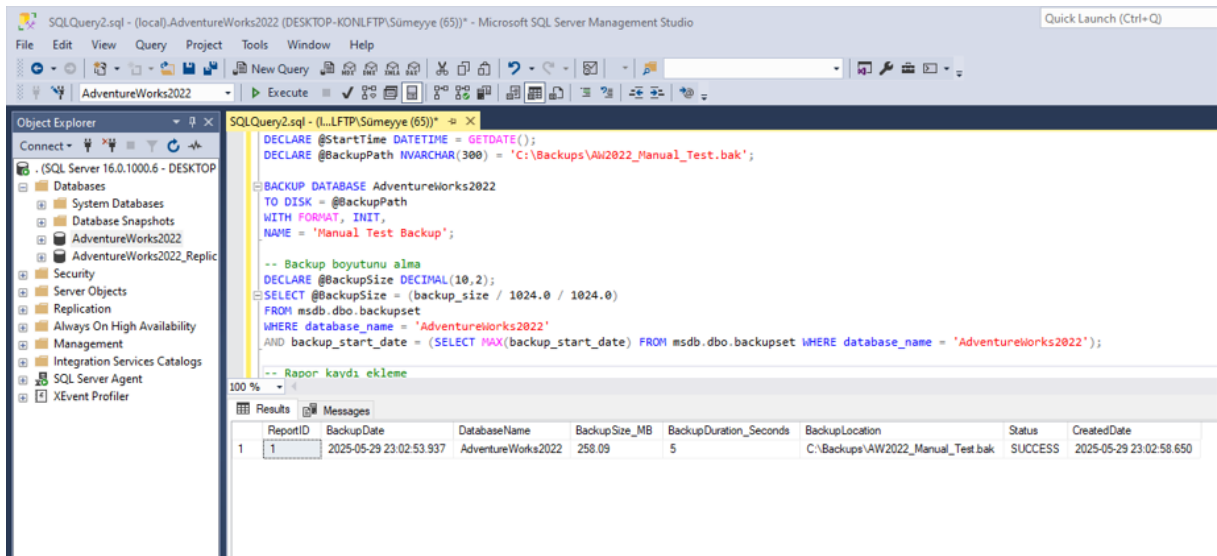
Şekil 3: Job schedule ayarları ve otomatik çalışma konfigürasyonu.

2. PowerShell ve T-SQL Scripting ile Yedekleme Raporları

2.1. Yedekleme Rapor Sistemi

Yedekleme işlemlerinin izlenmesi için BackupReports tablosu oluşturulmuş ve otomatik rapor sistemi kurulmuştur.

```
CREATE TABLE BackupReports (  
    ReportID INT IDENTITY(1,1) PRIMARY KEY,  
    BackupDate DATETIME,  
    DatabaseName NVARCHAR(100),  
    BackupSize_MB DECIMAL(10,2),  
    BackupDuration_Seconds INT,  
    BackupLocation NVARCHAR(300),  
    Status NVARCHAR(20),  
    CreatedDate DATETIME DEFAULT GETDATE()  
);
```



Şekil 4: BackupReports tablosunun oluşturulması ve yapısı.

2.2. Manuel Yedekleme Testi

Otomatik sistem test edilmeden önce manuel yedekleme işlemi gerçekleştirilmiş ve rapor kaydı oluşturulmuştur.

```
DECLARE @StartTime DATETIME = GETDATE();  
DECLARE @BackupPath NVARCHAR(300) = 'C:\Backups\AW2022_Manual_Test.bak';
```

BACKUP DATABASE AdventureWorks2022

TO DISK = @BackupPath

WITH FORMAT, INIT,

NAME = 'Manual Test Backup';

INSERT INTO BackupReports (BackupDate, DatabaseName, BackupSize_MB,
BackupDuration_Seconds, BackupLocation, Status)

VALUES (

@StartTime,

'AdventureWorks2022',

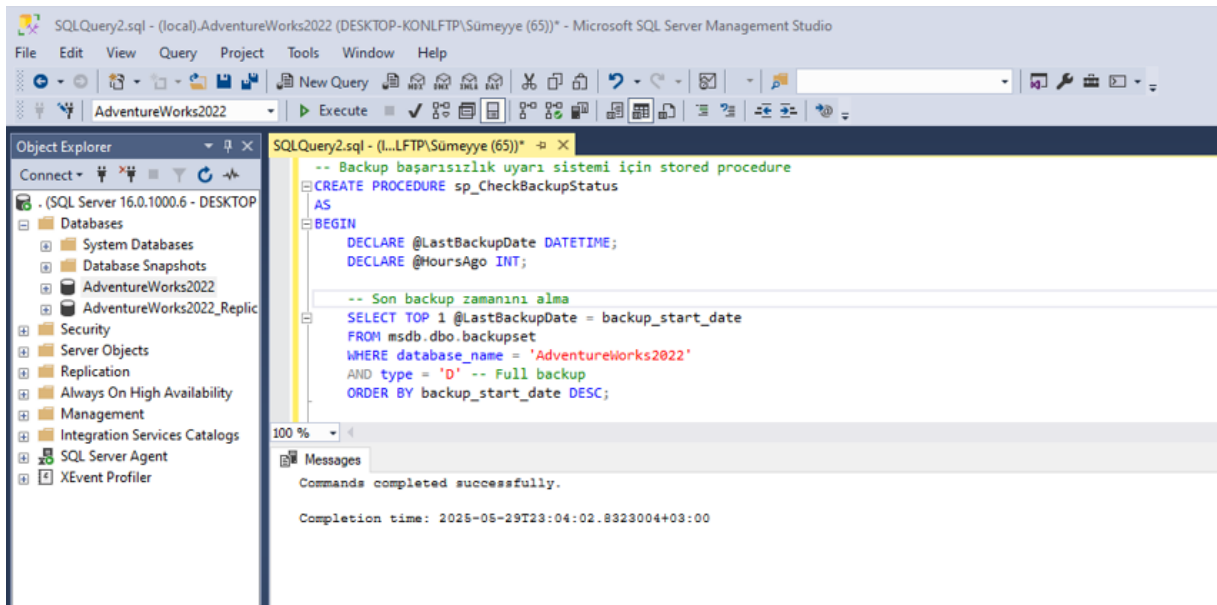
@BackupSize,

DATEDIFF(SECOND, @StartTime, GETDATE()),

@BackupPath,

'SUCCESS'

);



Şekil 5: Manuel backup testi sonuçları ve rapor kaydının oluşturulması.

3. Otomatik Yedekleme Uyarıları

3.1. Backup Status Monitoring

Yedekleme işlemlerinin düzenli takibi için stored procedure oluşturulmuş ve uyarı sistemi kurulmuştur.

```
CREATE PROCEDURE sp_CheckBackupStatus
AS
BEGIN
    DECLARE @LastBackupDate DATETIME;
    DECLARE @HoursAgo INT;

    SELECT TOP 1 @LastBackupDate = backup_start_date
    FROM msdb.dbo.backupset
    WHERE database_name = 'AdventureWorks2022'
    AND type = 'D'
    ORDER BY backup_start_date DESC;

    SET @HoursAgo = DATEDIFF(HOUR, @LastBackupDate, GETDATE());

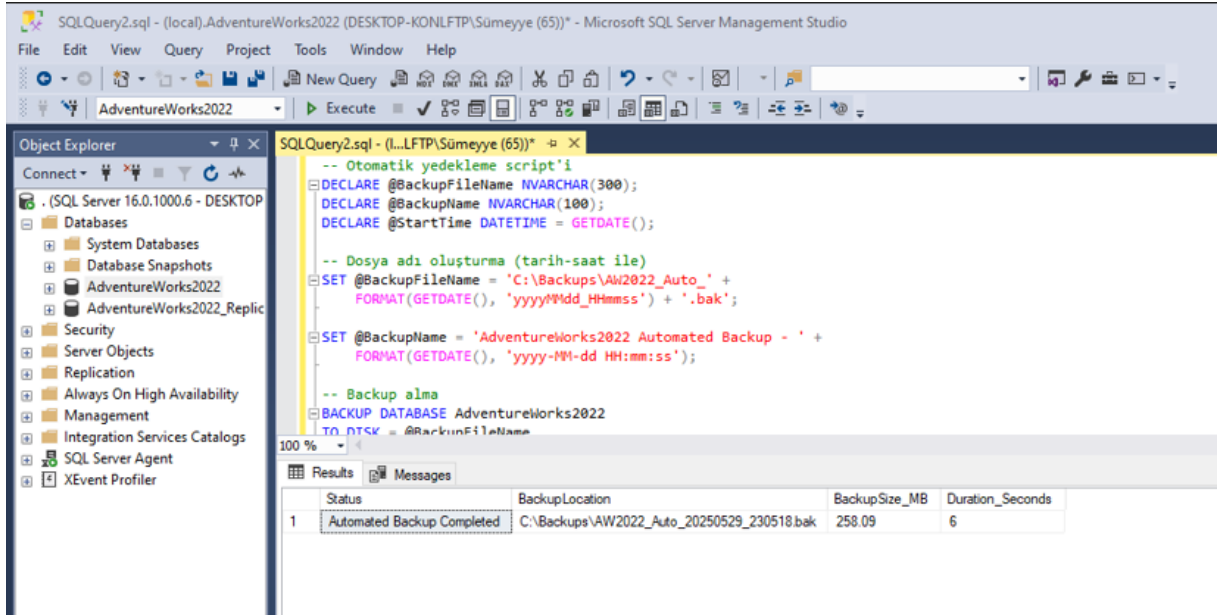
    IF @HoursAgo > 25
    BEGIN
        INSERT INTO BackupReports (BackupDate, DatabaseName, BackupSize_MB,
        BackupDuration_Seconds, BackupLocation, Status)
        VALUES (GETDATE(), 'AdventureWorks2022', 0, 0, 'N/A', 'WARNING: No backup in
        24+ hours');

        PRINT 'WARNING: Last backup was ' + CAST(@HoursAgo AS VARCHAR) + ' hours
        ago!';
    END
    ELSE
    BEGIN
```

```
PRINT 'Backup status OK. Last backup: ' + CAST(@HoursAgo AS VARCHAR) + '
hours ago';
```

```
END
```

```
END;
```



Şekil 6: Backup status check stored procedure'ı ve test sonuçları.

3.2. Otomatik Yedekleme Script'i

Job'da kullanılmak üzere kapsamlı otomatik yedekleme script'i geliştirilmiştir.

```
DECLARE @BackupFileName NVARCHAR(300);
```

```
DECLARE @BackupName NVARCHAR(100);
```

```
DECLARE @StartTime DATETIME = GETDATE();
```

```
SET @BackupFileName = 'C:\Backups\AW2022_Auto_' +  
    FORMAT(GETDATE(), 'yyyyMMdd_HH:mm:ss') + '.bak';
```

```
SET @BackupName = 'AdventureWorks2022 Automated Backup - ' +  
    FORMAT(GETDATE(), 'yyyy-MM-dd HH:mm:ss');
```

```
BACKUP DATABASE AdventureWorks2022
```

```
TO DISK = @BackupFileName
```

WITH FORMAT, INIT,

NAME = @BackupName,

DESCRIPTION = 'Automated daily backup';

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'AdventureWorks2022'. The central pane shows a SQL query titled 'Yedekleme izleme dashboard'u'. The query is a SELECT statement that aggregates backup data from the 'BackupReports' table, filtering for 'SUCCESS' and 'AUTO_SUCCESS' statuses. The results pane on the right displays three tables: a summary of backup statistics, a detailed view of the last 24 hours, and a status log.

ReportTitle	TotalBackups	AvgBackupSize_MB	AvgDuration_Seconds	FirstBackup	LastBackup
Backup Monitoring Dashboard	2	258.090000	5	2025-05-29 23:02:53.937	2025-05-29 23:05:18.243

Period	BackupCount	TotalSize_MB	LastBackupTime
Last 24 Hours Status	2	516.18	2025-05-29 23:05:18.243

Status	Count	LastOccurence
AUTO_SUCCESS	1	2025-05-29 23:05:24.767
SUCCESS	1	2025-05-29 23:02:58.650

Şekil 7: Otomatik backup script'inin çalıştırılması ve sonuçları.

4. Job Test ve Doğrulama

4.1. SQL Server Agent Job Testi

Oluşturulan job test edilmiş ve çalışma durumu doğrulanmıştır.

The screenshot shows the 'Start Jobs' dialog box in SQL Server Agent. It displays the results of a job test for 'AutoBackup_Sumeyye'. The dialog indicates that the 'Start Job' action was successful, while the 'Execute job' action failed. The 'Details' section provides a table of the actions and their outcomes.

Action	Status	Message
Start Job 'AutoBackup_Sumeyye'	Success	
Execute job 'AutoBackup_Sumeyye'	Error	Execution of job

Şekil 7: Job test sonuçları - Start Job başarılı, Execute job durumu.

4.2. Backup Geçmişi Kontrolü

Yedekleme işlemlerinin geçmişi sorgulanarak sistem durumu kontrol edilmiştir.

SELECT TOP 5

database_name,

backup_start_date,

backup_finish_date,

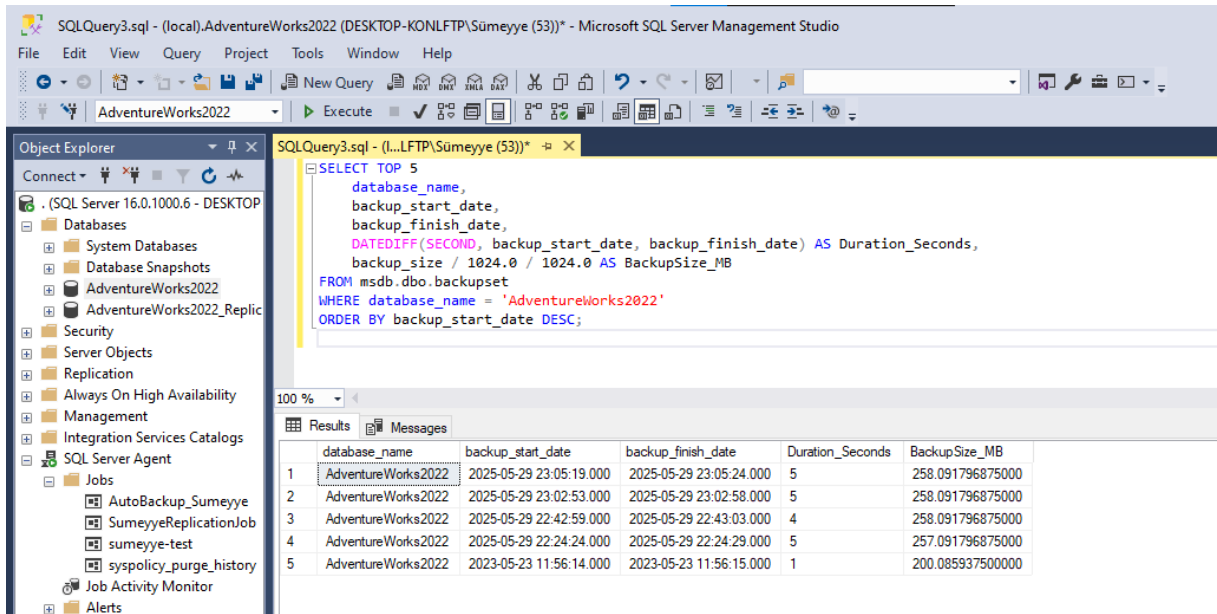
DATEDIFF(SECOND, backup_start_date, backup_finish_date) AS Duration_Seconds,

backup_size / 1024.0 / 1024.0 AS BackupSize_MB

FROM msdb.dbo.backupset

WHERE database_name = 'AdventureWorks2022'

ORDER BY backup_start_date DESC;



The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor on the right contains the following SQL query:

```
SELECT TOP 5
    database_name,
    backup_start_date,
    backup_finish_date,
    DATEDIFF(SECOND, backup_start_date, backup_finish_date) AS Duration_Seconds,
    backup_size / 1024.0 / 1024.0 AS BackupSize_MB
FROM msdb.dbo.backupset
WHERE database_name = 'AdventureWorks2022'
ORDER BY backup_start_date DESC;
```

The Results pane at the bottom displays the output of the query as a table with 5 rows and 5 columns:

	database_name	backup_start_date	backup_finish_date	Duration_Seconds	BackupSize_MB
1	AdventureWorks2022	2025-05-29 23:05:19.000	2025-05-29 23:05:24.000	5	258.091796875000
2	AdventureWorks2022	2025-05-29 23:02:53.000	2025-05-29 23:02:58.000	5	258.091796875000
3	AdventureWorks2022	2025-05-29 22:42:59.000	2025-05-29 22:43:03.000	4	258.091796875000
4	AdventureWorks2022	2025-05-29 22:24:24.000	2025-05-29 22:24:29.000	5	257.091796875000
5	AdventureWorks2022	2023-05-23 11:56:14.000	2023-05-23 11:56:15.000	1	200.085937500000

Şekil 8: Backup geçmişi ve işlem detayları.

5. Yedekleme İzleme Dashboard'u

5.1. Monitoring Dashboard

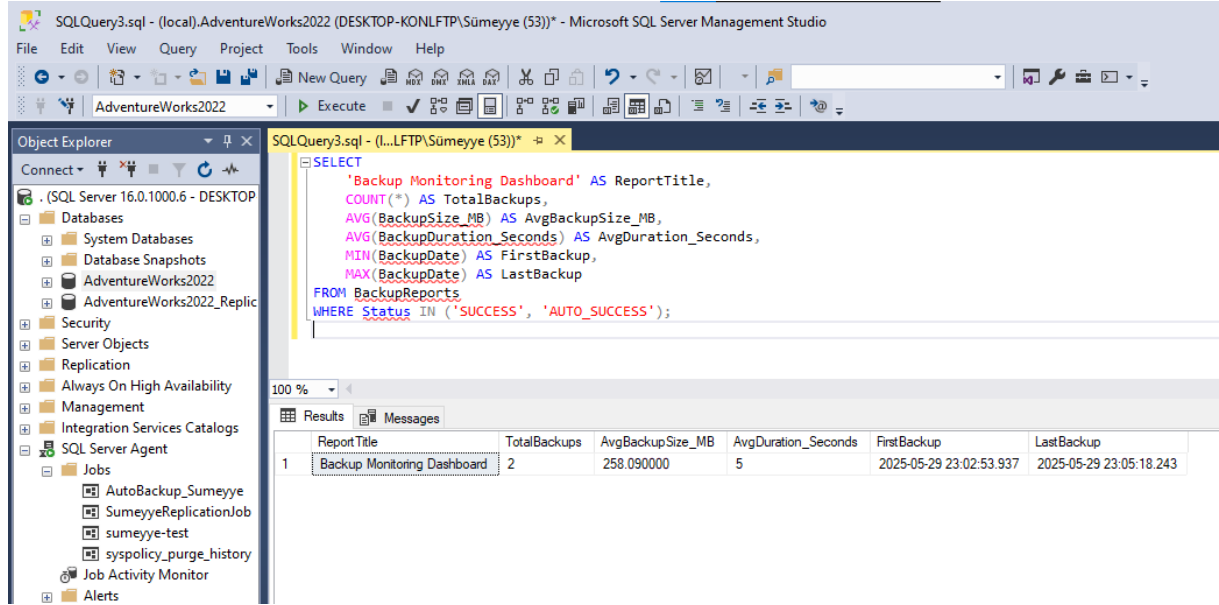
Yedekleme süreçlerinin kapsamlı izlenmesi için dashboard oluşturulmuştur.

SELECT

'Backup Monitoring Dashboard' AS ReportTitle,

COUNT(*) AS TotalBackups,

AVG(BackupSize_MB) AS AvgBackupSize_MB,
AVG(BackupDuration_Seconds) AS AvgDuration_Seconds,
MIN(BackupDate) AS FirstBackup,
MAX(BackupDate) AS LastBackup
FROM BackupReports
WHERE Status IN ('SUCCESS', 'AUTO_SUCCESS');



The screenshot displays the Microsoft SQL Server Management Studio interface. The 'Object Explorer' on the left shows the database structure for 'AdventureWorks2022'. The 'SQL Query3.sql' window in the center contains the following query:

```
SELECT  
    'Backup Monitoring Dashboard' AS ReportTitle,  
    COUNT(*) AS TotalBackups,  
    AVG(BackupSize_MB) AS AvgBackupSize_MB,  
    AVG(BackupDuration_Seconds) AS AvgDuration_Seconds,  
    MIN(BackupDate) AS FirstBackup,  
    MAX(BackupDate) AS LastBackup  
FROM BackupReports  
WHERE Status IN ('SUCCESS', 'AUTO_SUCCESS');
```

The 'Results' pane at the bottom shows the output of the query as a table with 6 columns: ReportTitle, TotalBackups, AvgBackupSize_MB, AvgDuration_Seconds, FirstBackup, and LastBackup. The table contains one row of data.

ReportTitle	TotalBackups	AvgBackupSize_MB	AvgDuration_Seconds	FirstBackup	LastBackup
Backup Monitoring Dashboard	2	258.090000	5	2025-05-29 23:02:53.937	2025-05-29 23:05:18.243

Şekil 9: Backup monitoring dashboard ve performans metrikleri.