T-SQL CRUD

(SELECT)

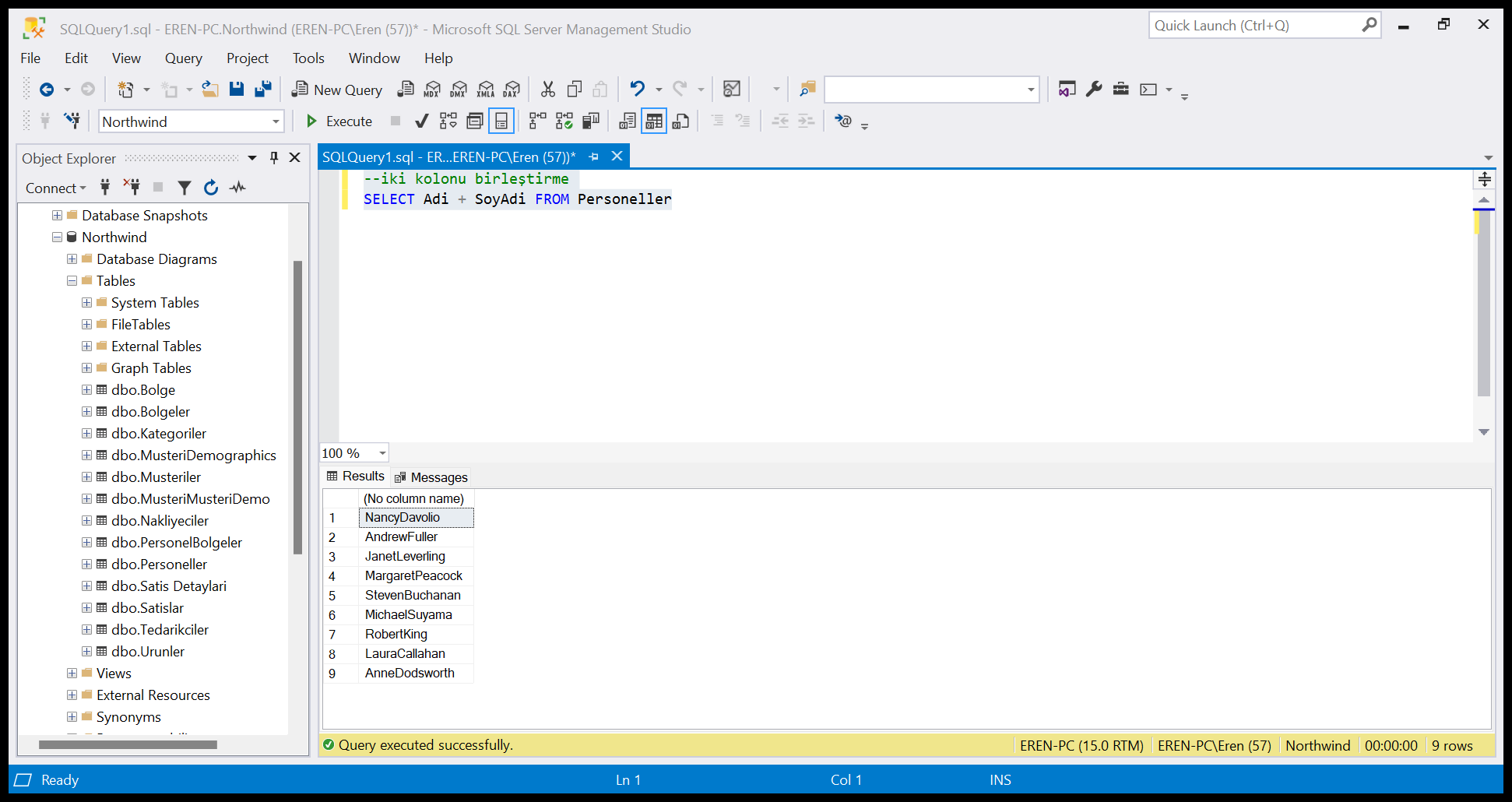
Example

SELECT \* FROM Personeller

Example

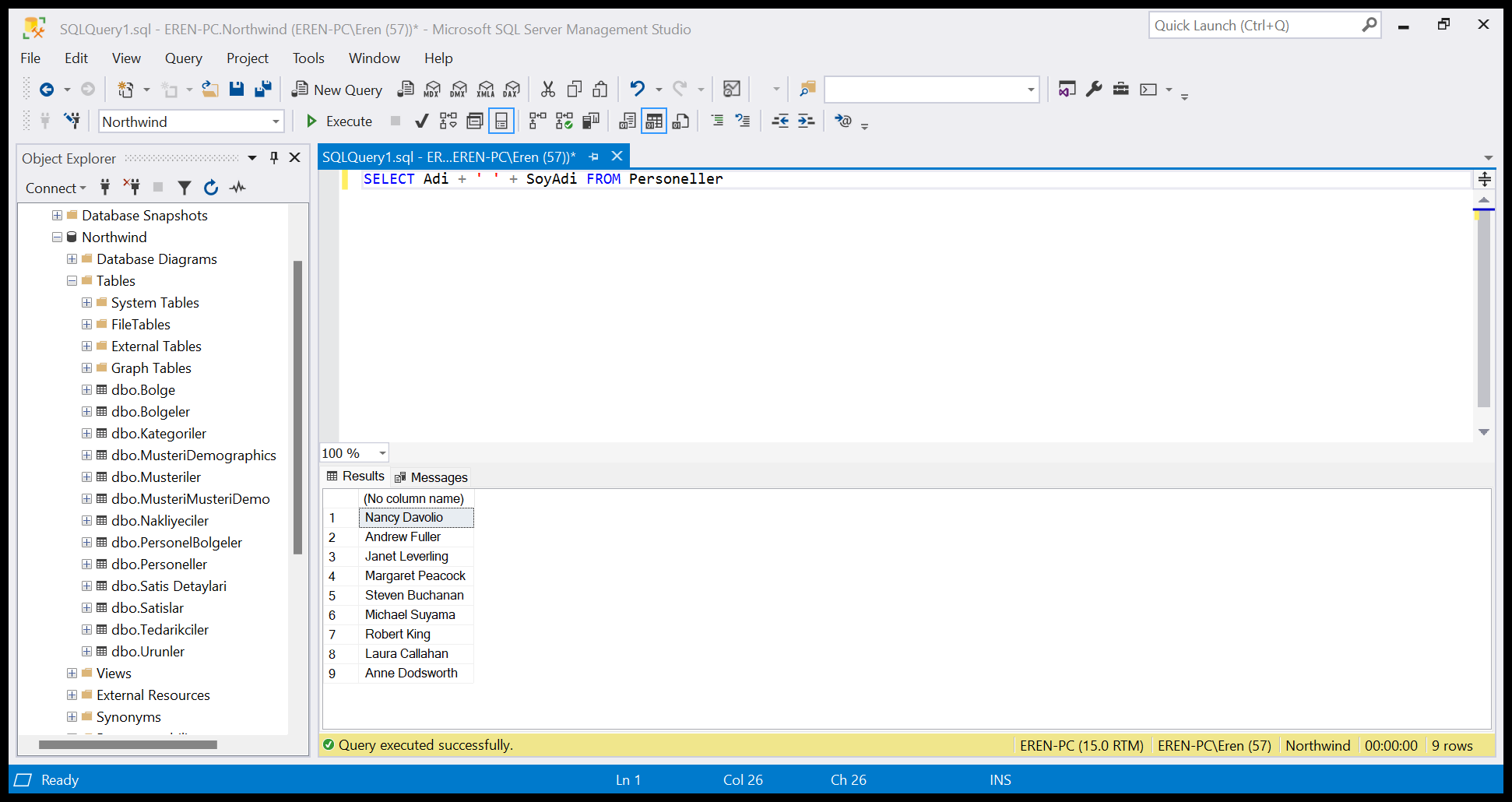
--iki kolonu birleştirme

SELECT Adi + SoyAdi FROM Personeller



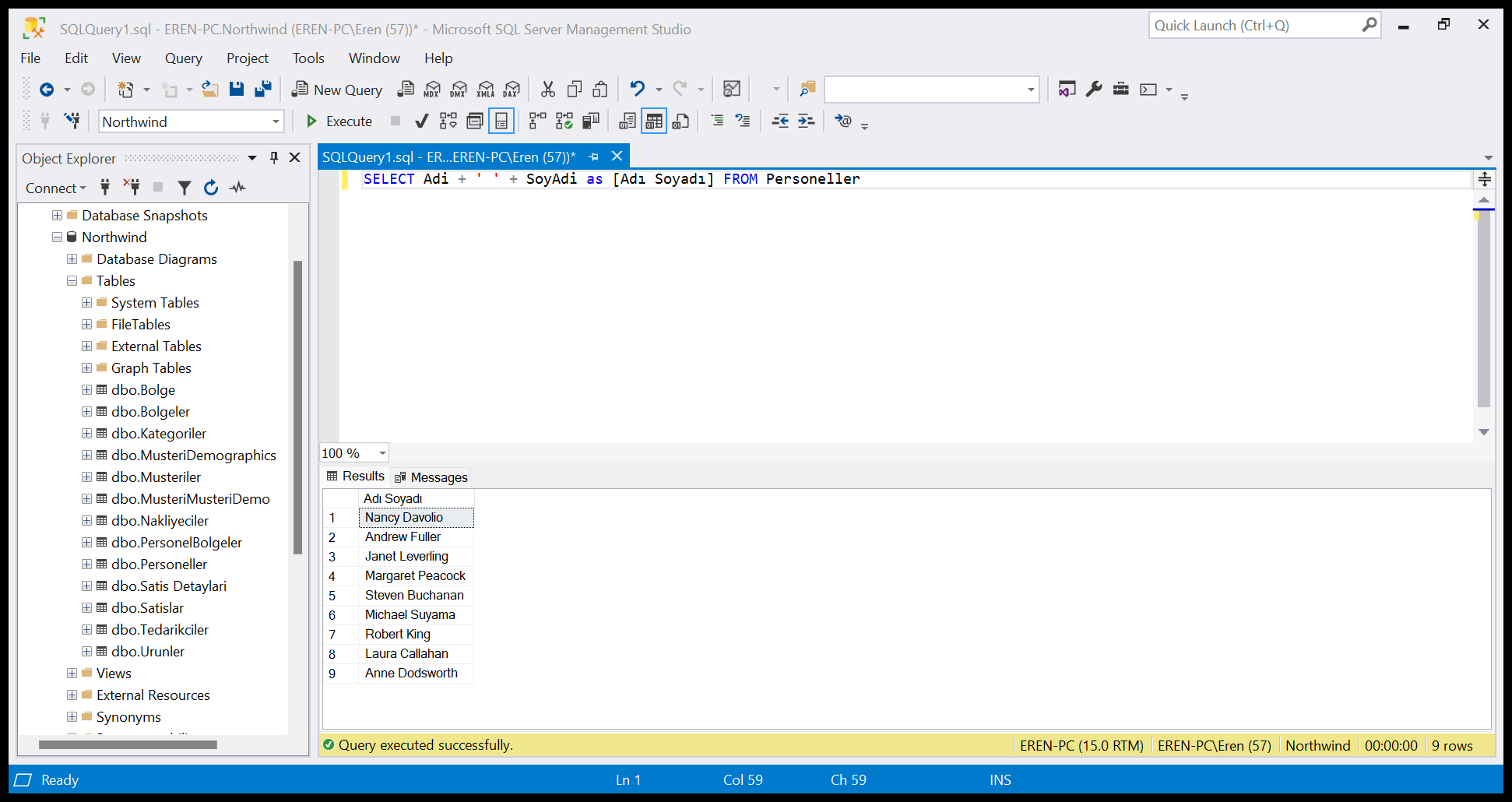
Example

SELECT Adi + ' ' + SoyAdi FROM Personeller



Example

SELECT Adi + ' ' + SoyAdi as [Adı Soyadı] FROM Personeller

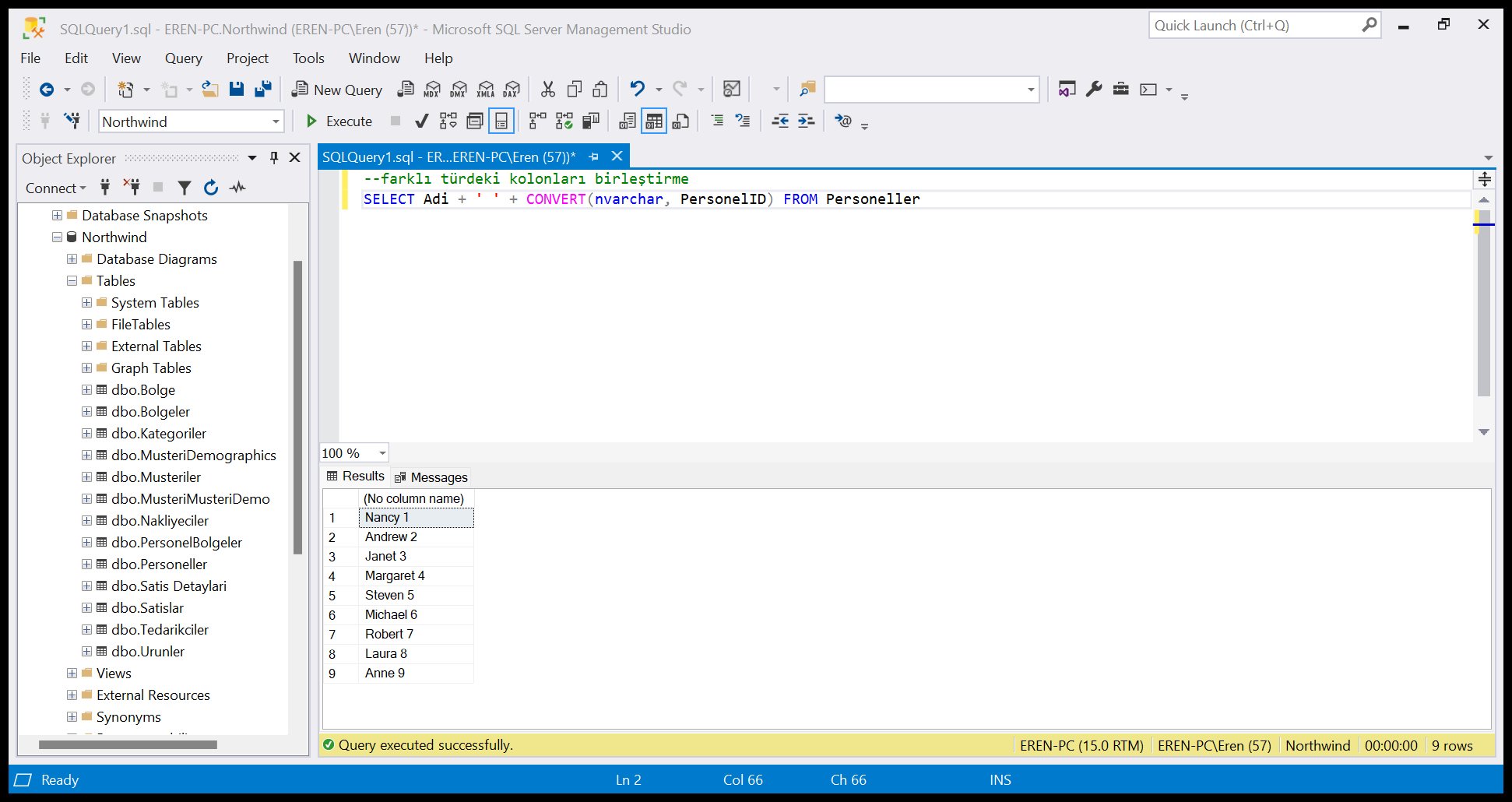


|  |
| --- |
| **NOT**  Kolonlara takma ad(alias) vermek için AS operatörünü kullanabilirsiniz. |

Example

--farklı türdeki kolonları birleştirme

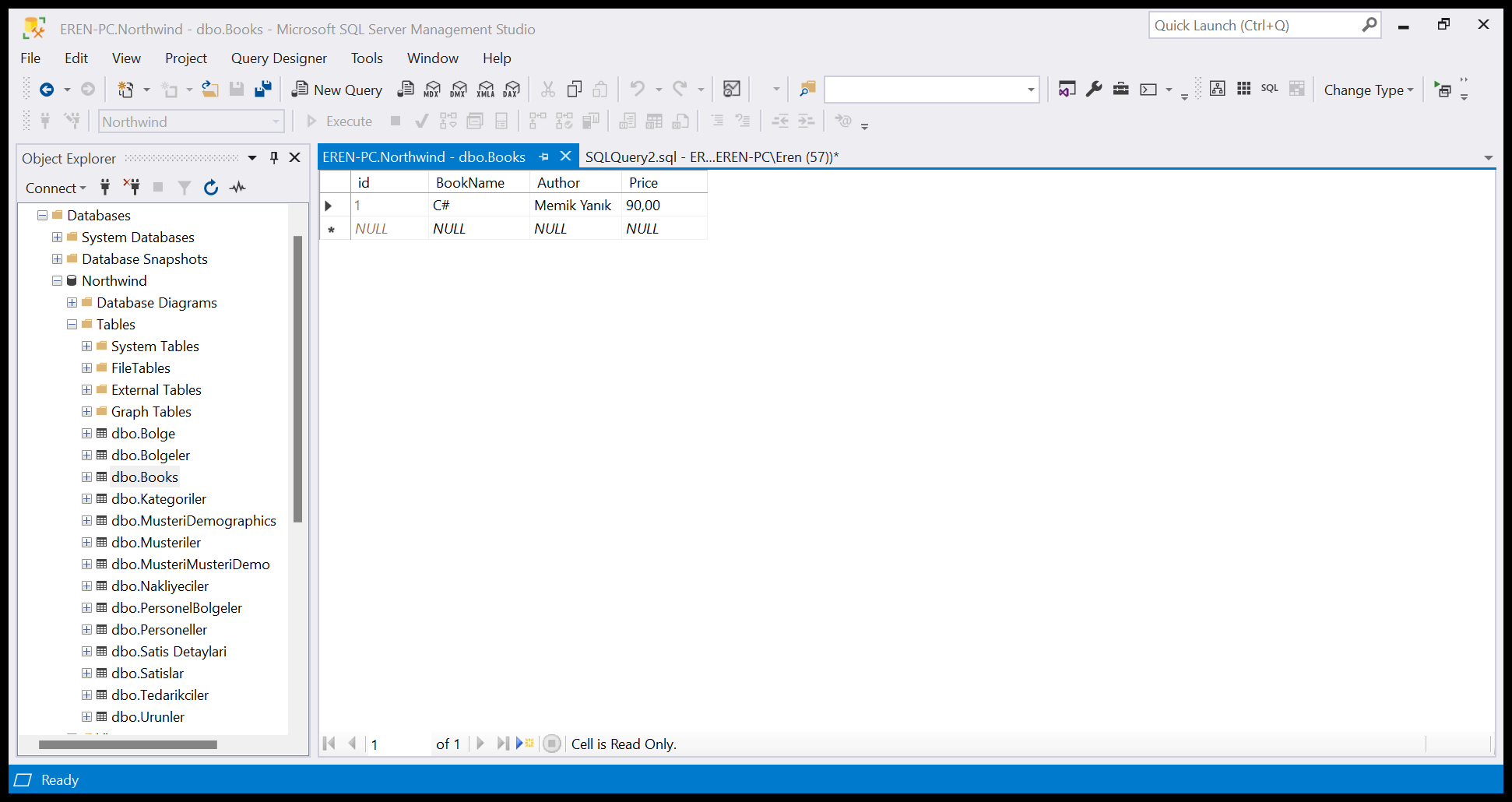
SELECT Adi + ' ' + CONVERT(nvarchar, PersonelID) FROM Personeller



(INSERT)

Example

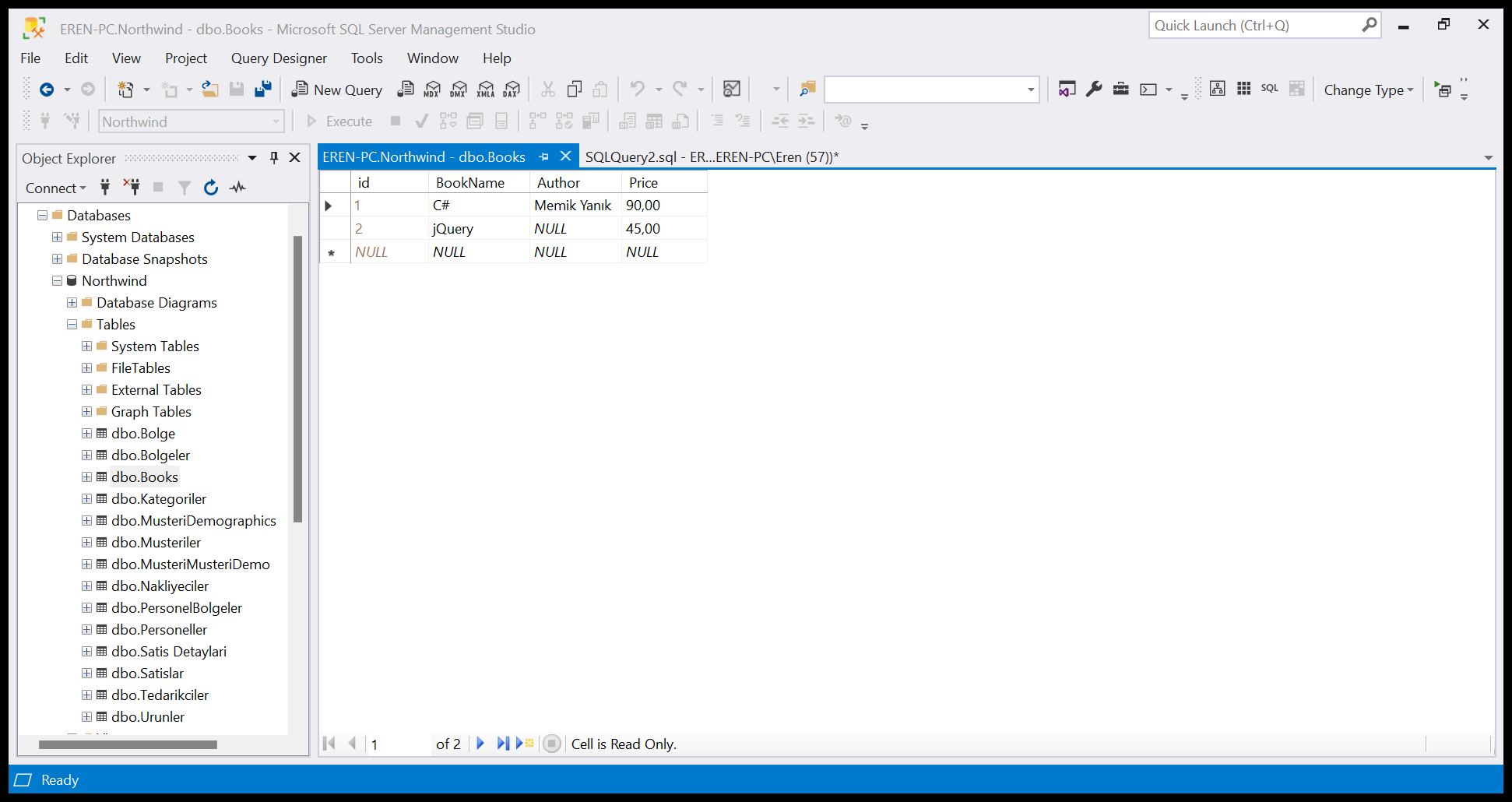
INSERT INTO Books(BookName, Author, Price) VALUES('C#', 'Memik Yanık', 90)



* INSERT komutu ile tabloya kayıt eklerken tüm kolonlara değer aktarmak zorunda değiliz. Aşağıdaki örnekte yazar alanı boş geçilmiştir.

Example

INSERT INTO Books(BookName, Price) VALUES('jQuery', 45.00)



(UPDATE)

Example

UPDATE Books SET Author = 'Salih Baltalı' WHERE id = 2

(DELETE)

Example

DELETE FROM Books WHERE id = 2

Database

(CREATE)

Example

--Varsayılan ayarlar ile bir veritabanı oluşturmaktadır.

CREATE DATABASE store

Example

CREATE DATABASE store ON

(

NAME = 'shop', --Disk üzerindeki adı.

FILENAME = 'C:\Database\shop.mdf', --Konumu

SIZE = 5, --MB cinsinden ilk boyut

FILEGROWTH = 3 --Boyut artış miktarı.

)

* Veritabanı oluştururken log dosyası da oluşturabilirsiniz.

Example

CREATE DATABASE store ON

(

NAME = 'shop',

FILENAME = 'C:\Database\shop.mdf',

SIZE = 5,

FILEGROWTH = 3

)

LOG

ON(

NAME = 'shop\_log',

FILENAME = 'C:\Database\shop\_log.ldf',

SIZE = 5,

FILEGROWTH = 3

)

(ALTER)

Example

--SIZE ve FILEGROWTH değerleri güncellenir.

ALTER DATABASE store

MODIFY FILE

(

NAME = 'shop', --disk üzerindeki adın belirtilmesi gerekmektedir.

SIZE = 10,

FILEGROWTH = 6

)

(DROP)

Example

DROP DATABASE store

Table

(CREATE)

Example

CREATE TABLE Employee

(

Id INT PRIMARY KEY IDENTITY(1, 1) NOT NULL,

FirstName NVARCHAR(255),

LastName NVARCHAR(255),

City NVARCHAR(255),

DepartmentId INT

)

(ALTER)

* Aşağıdaki örnekte tabloya yeni bir kolon eklenmektedir.

Example

ALTER TABLE Employee

ADD Salary money

* Aşağıdaki örnekte tablo içerisindeki bir kolon üzerinde değişiklik yapılmaktadır.

Example

ALTER TABLE Employee

ALTER COLUMN Salary decimal(18,2)

* Aşağıdaki örnekte tablonun bir kolonu silinmektedir.

Example

ALTER TABLE Employee

DROP COLUMN Salary

* Aşağıdaki örnekte tabloya constraint eklenmektedir.

Example

ALTER TABLE Employee

ADD CONSTRAINT defaultConstraint DEFAULT 'None' for City

* Aşağıdaki örnekte tablodaki bir constraint silinmektedir.

Example

ALTER TABLE Employee

DROP CONSTRAINT defaultConstraint

(DROP)

Example

DROP TABLE Employees

(Tablo Adını Değiştirme)

Example

--Tablonun yeni adı Employees olacaktır.

SP\_RENAME 'Employee', 'Employees'

(Kolon Adını Değiştirme)

Example

--Kolon adı DepartmentId olarak değiştirilecektir.

SP\_RENAME 'Employees.DepartmentID', 'DepartmentId', 'column'

(Tablo Kopyalama)

Example

SELECT \* INTO Employee FROM Employees

|  |
| --- |
| **NOT**  Bir tablo yukarıdaki gibi kopyalandığında primary key ve foreign key constraint’ler oluşturulmaz. |

(TRUNCATE TABLE)

Tablonun içeriğini siler ve IDENTITY kolonunu sıfırlar.

Example

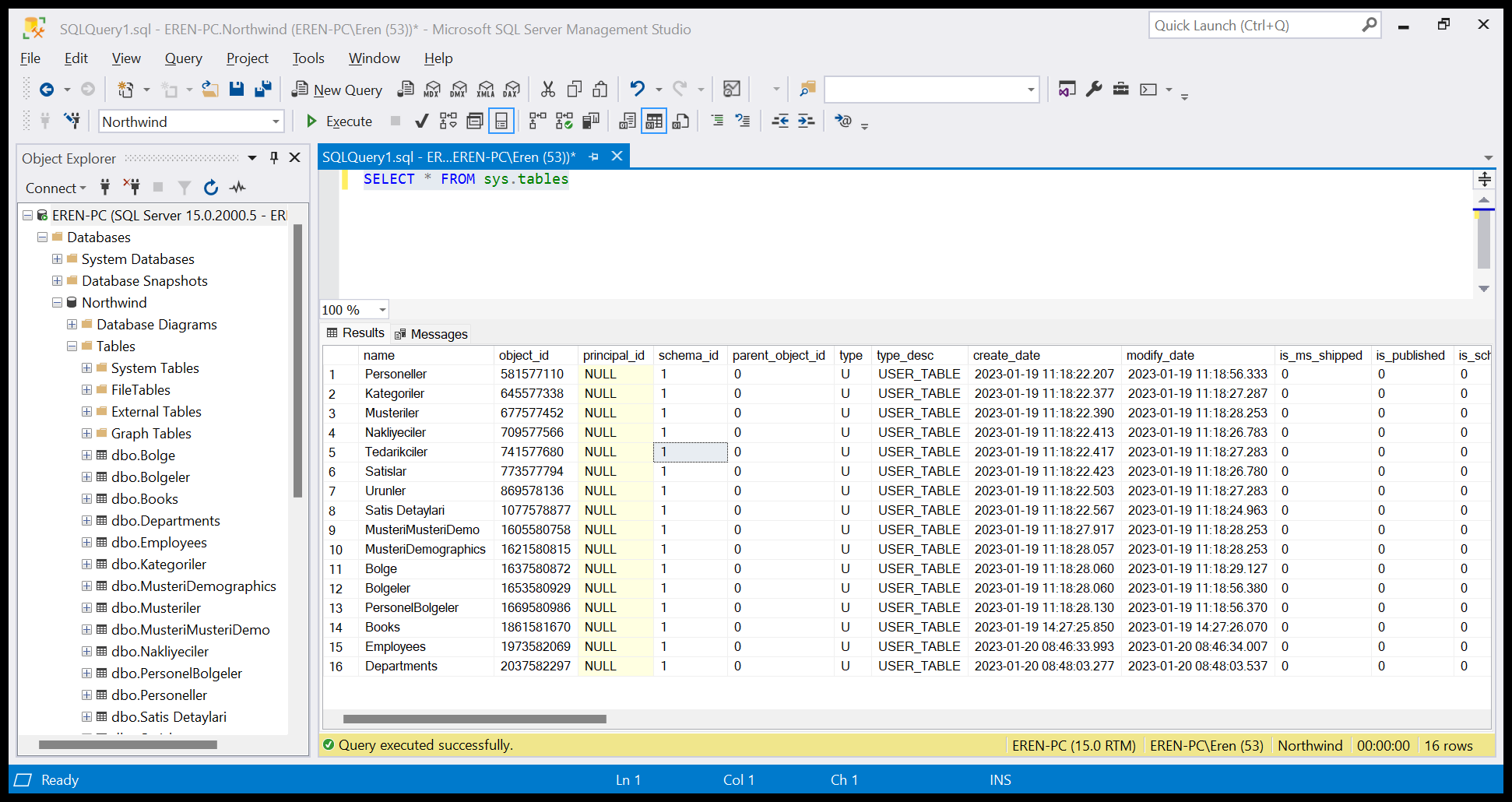
TRUNCATE TABLE Employees

(sys.tables)

Aktif veritabanı içerisindeki tüm tabloları listelemek için kullanılır.

Example

SELECT \* FROM sys.tables



(OBJECTPROPERTY Fonksiyonu)

Bir tablonun primary key içerip içermediğini döndürmektedir.

Example

--1 : Tablo primary key içermektedir.

--0 : Tablo bir primary key'e sahip değildir.

SELECT OBJECTPROPERTY(OBJECT\_ID('Personeller'), 'TableHasPrimaryKey')

SELECT Queries

(WHERE)

Example

--Sehir kolonu London olan kayıtları döndürür.

SELECT \* FROM Personeller where Sehir = 'London'

Example

--BagliCalistigiKisi kolonu 5'den küçük olan kayıtları döndürür.

SELECT \* FROM Personeller where BagliCalistigiKisi < 5

Example

--Hem Sehir kolonu London hem de BagliCalistigiKisi kolonu 5 olan kayıtları döndürür

SELECT \* FROM Personeller where Sehir = 'London' AND BagliCalistigiKisi = 5

Example

--Sehir kolonu London ve BagliCalistigiKisi kolonu 5 olan kayıtları döndürür

SELECT \* FROM Personeller where Sehir = 'London' OR BagliCalistigiKisi = 5

Example

--Sehir kolonu London ve Seattle olan kayıtları döndürür.

SELECT \* from Personeller WHERE Sehir = 'London' OR Sehir = 'Seattle'

Example

--Sehir kolonu London olmayan kayıtları döndürür.

SELECT \* FROM Personeller where NOT Sehir = 'London'

Example

--1993 yılından sonra işe başlayan personelleri döndürür.

SELECT \* from Personeller WHERE YEAR(IseBaslamaTarihi) > 1993

(BETWEEN, NOT BETWEEN)

Example

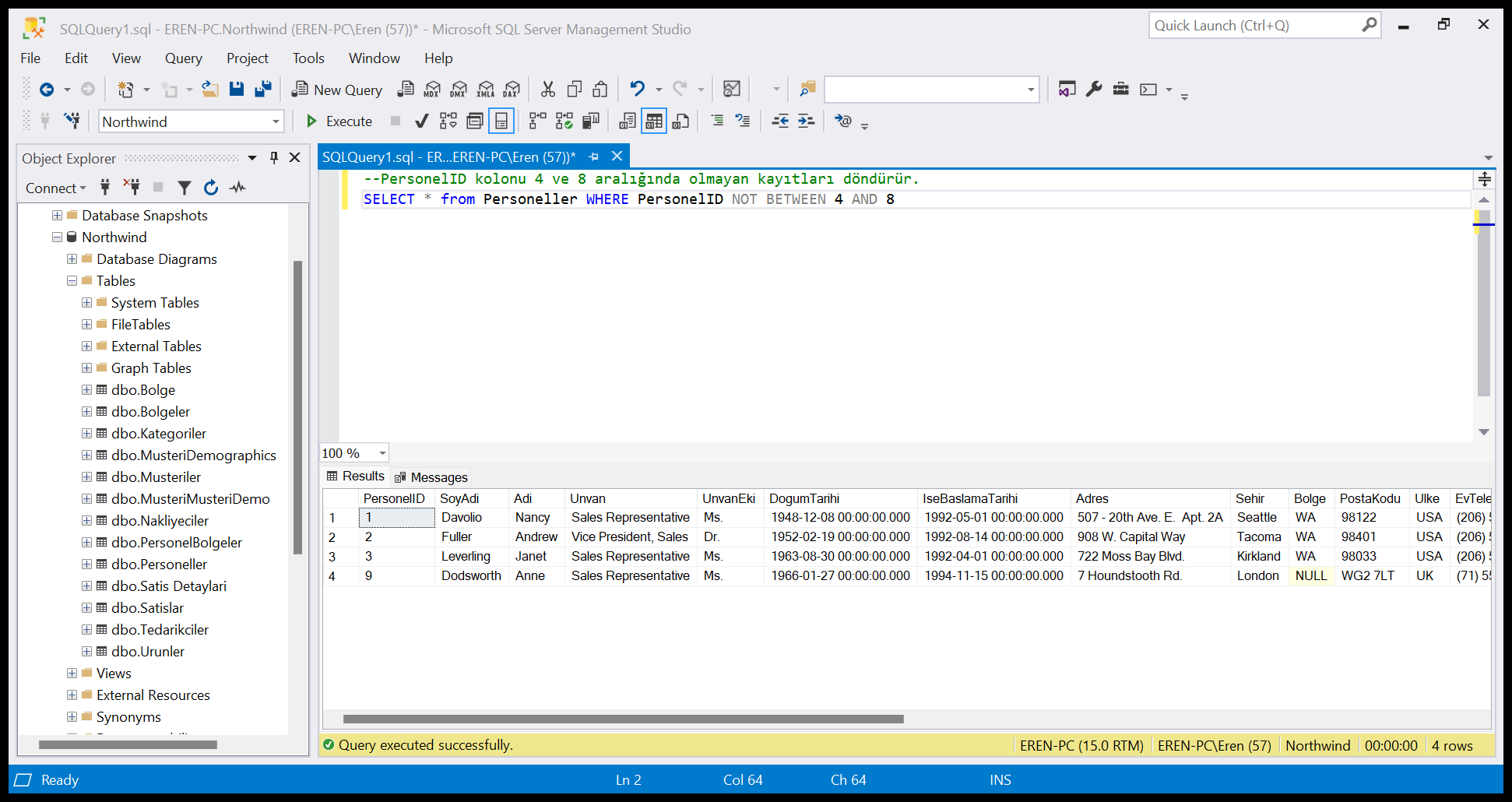
--PersonelID kolonu 4 ve 8 aralığında olan kayıtları döndürür. (4 ve 8 dahil)

SELECT \* from Personeller WHERE PersonelID BETWEEN 4 AND 8

Example

--PersonelID kolonu 4 ve 8 aralığında olmayan kayıtları döndürür.

SELECT \* from Personeller WHERE PersonelID NOT BETWEEN 4 AND 8



Example

--Adi kolonu A ve M(ile başlayan) aralığındaki kayıtları döndürür. (A dahil, M dahil değil)

SELECT \* from Personeller WHERE Adi BETWEEN 'A' AND 'M'

Example

--Adi kolonu A ve M(ile başlayan) aralığında olmayan kayıtları döndürür. (A dahil değil, M dahil)

SELECT \* from Personeller WHERE Adi NOT BETWEEN 'A' AND 'M'

(IS NULL, IS NOT NULL)

Example

--Bolge kolonu NULL olan kayıtları döndürür.

SELECT \* from Personeller WHERE Bolge IS NULL

Example

--Bolge kolonu NULL olmayan kayıtları döndürür.

SELECT \* from Personeller WHERE Bolge IS NOT NULL

(IN, NOT IN)

Example

--Sehir kolonu London ve Seattle olan kayıtları döndürür.

SELECT \* from Personeller WHERE Sehir IN('London', 'Seattle')

Example

--Sehir kolonu London ve Seattle olmayan kayıtları döndürür.

SELECT \* from Personeller WHERE Sehir NOT IN('London', 'Seattle')

(LIKE)

LIKE operatörü ile birlikte kullanılan özel karakterler aşağıda verilmiştir.

[]

%

^

\_

Example

--Adi kolonunda A harfini içeren kayıtları döndürür.

SELECT \* FROM Personeller WHERE Adi LIKE '%A%'

Example

--Başı önemli olmayan ama sonu T ile biten kayıtlar döner.

SELECT \* FROM Personeller WHERE Adi LIKE '%T'

Example

--M harfi ile başlayan kayıtlar döner.

SELECT \* FROM Personeller WHERE Adi LIKE 'M%'

Example

--İlk harfi önemli olmayan ancak ikinci harfi A olan kayıtlar döner.

SELECT \* FROM Personeller WHERE Adi LIKE '\_A%'

Example

--İlk iki harfi önemli olmayan ancak üçüncü harfi N olan kayıtlar döner.

SELECT \* FROM Personeller WHERE Adi LIKE '\_\_N%'

Example

--Son iki harfi önemli olmayan ancak sondan üçüncü harfi N olan kayıtlar döner.

SELECT \* FROM Personeller WHERE Adi LIKE '%N\_\_'

Example

--Adi kolonu M harfi ile başlayan kayıtlar döner.

SELECT \* FROM Personeller WHERE Adi LIKE '[M]%'

Example

--Adi kolonu M ve A harfleri ile başlayan kayıtlar döner.

SELECT \* FROM Personeller WHERE Adi LIKE '[MA]%'

Example

--Adi kolonu M harfi ile başlayan ancak ikinci harfi A olmayan kayıtlar döner.

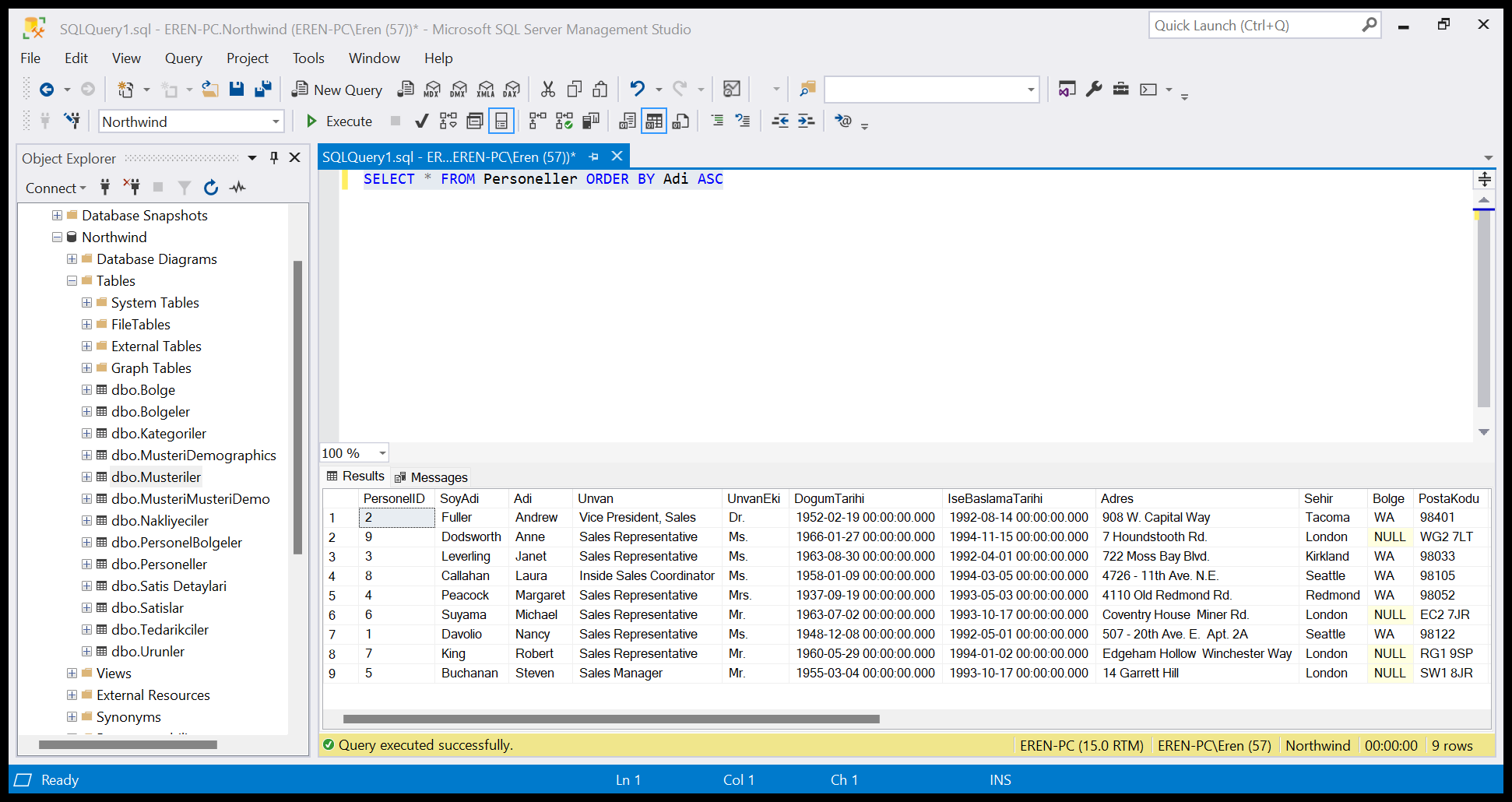
SELECT \* FROM Personeller WHERE Adi LIKE 'M[^A]%'

(ORDER BY)

ORDER BY ile birlikte ASC ve DESC anahtar kelimeleri kullanılmaktadır. ASC artan, DESC azalan bir şekilde sıralama yapar.

Example

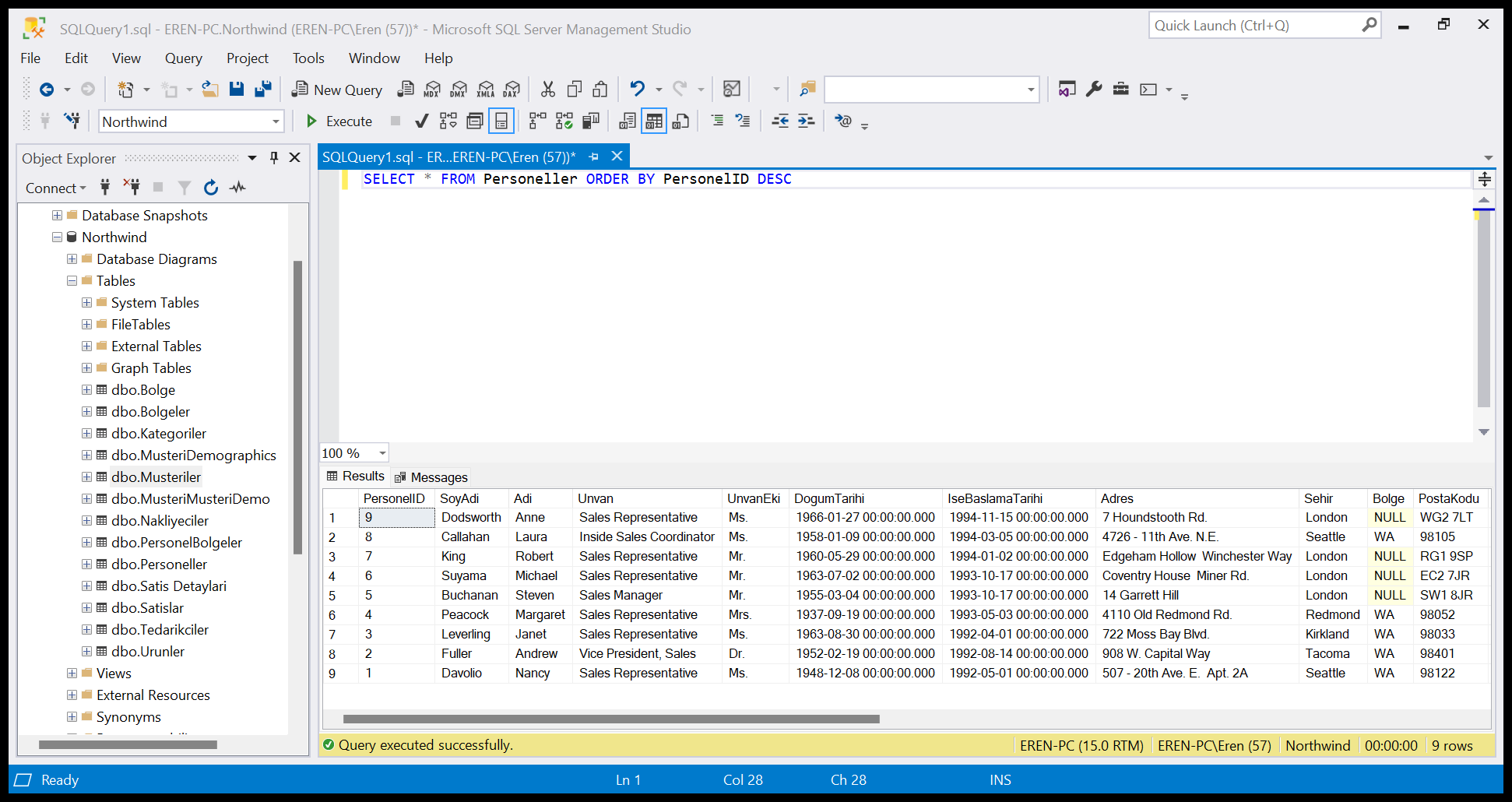
SELECT \* FROM Personeller ORDER BY Adi ASC



* ORDER BY ile nümerik kolonlar da sıralanabilmektedir.

Example

SELECT \* FROM Personeller ORDER BY PersonelID DESC



(TOP)

TOP anahtar kelimesi sınırlı sayıda kayıt döndürmek için kullanılmaktadır.

Example

--En üstteki iki kaydı döndürür.

SELECT TOP 2 \* FROM Personeller

Example

--Adi kolonunu azalan bir şekilde sıralar ve en üstteki kaydı döndürür.

SELECT TOP 1 \* FROM Personeller ORDER BY Adi DESC

(NEWID())

Tablodan rastgele kayıt getirmek için kullanılmaktadır.

Example

SELECT TOP 1 \* FROM Personeller ORDER BY NEWID()

(DISTINCT)

Tablo kolonu içindeki benzer olan verileri teke indirmek için kullanılır.

Example

SELECT DISTINCT Sehir FROM Personeller

(İç İçe Sorgular)

Example

SELECT \* FROM (

SELECT \* FROM Personeller WHERE Sehir = 'London'

) AS TableName WHERE Adi = 'Anne'

Burada SQL Server yeni bir tablo oluşturmaktadır. Parantez içindeki sorgu sonucunda dönen kayıtlar bu tabloya yerleştirilir. Parantez dışına yazılan AS anahtar kelimesi ile SQL Server tarafından oluşturulan tabloya bir isim verilir. Tablo adından sonra da WHERE komutu kullanarak yeni bir koşul belirtebiliyoruz.

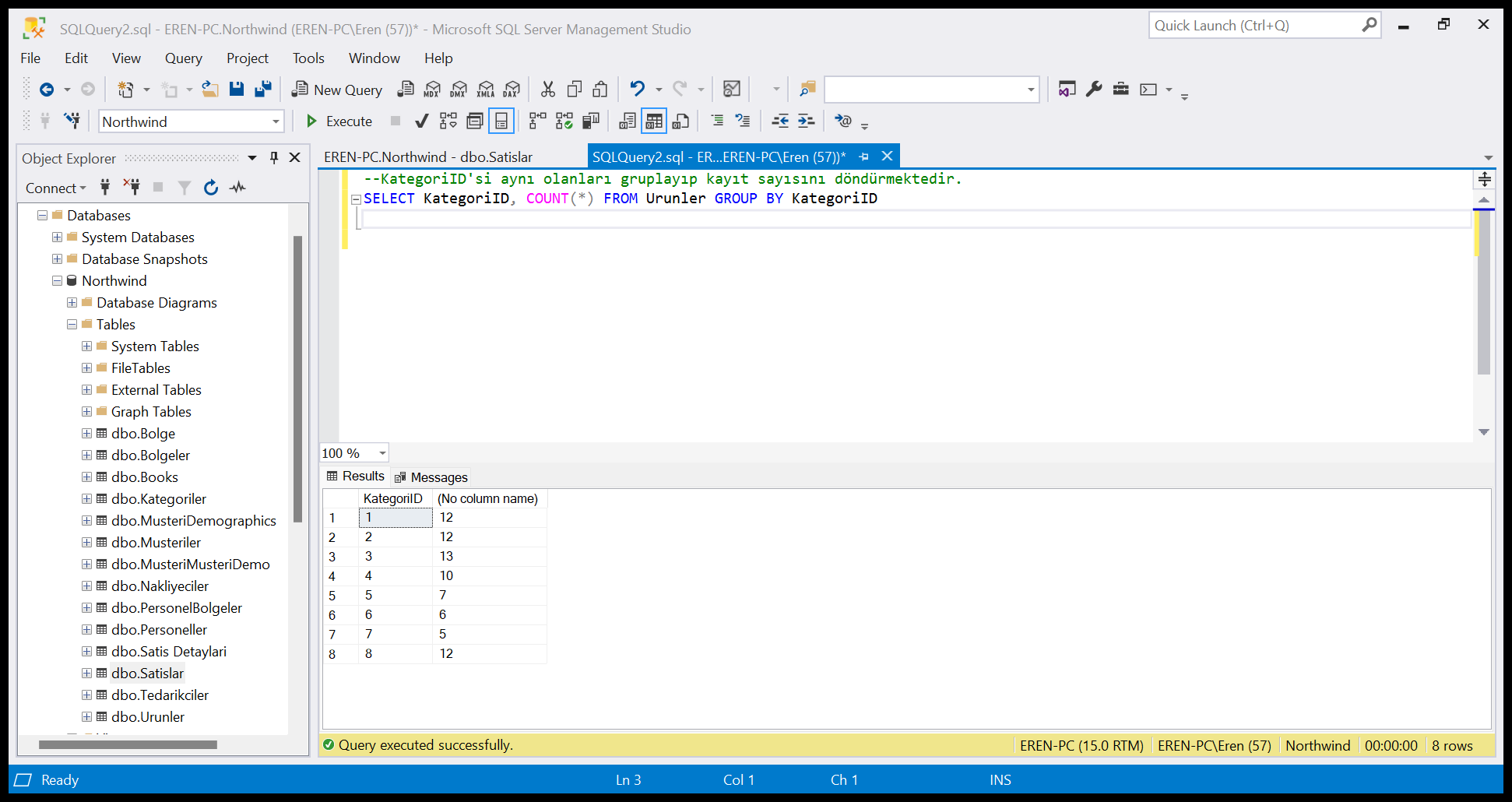
(GROUP BY)

* Sorgu içerisinde normal kolon ve aggregate fonksiyon varsa normal kolonun gruplanması gerekmektedir.

Example

--KategoriID'si aynı olanları gruplayıp kayıt sayısını döndürmektedir.

SELECT KategoriID, COUNT(\*) FROM Urunler GROUP BY KategoriID



Example

--Hangi personelin toplam ne kadar nakliye ücreti ödediğini döndürmektedir.

SELECT PersonelID, SUM(NakliyeUcreti) FROM Satislar GROUP BY PersonelID

* GROUP BY ve WHERE beraber kullanılabilir.

Example

--PersonelID'si 5'den büyük olan personellerin toplam ne kadar nakliye ücreti ödediğini döndürmektedir.

SELECT PersonelID, SUM(NakliyeUcreti) FROM Satislar where PersonelID > 5 GROUP BY PersonelID

(HAVING)

Aggregate fonksiyonlar için koşul belirtmeyi sağlamaktadır.

Example

SELECT KategoriID, COUNT(\*) FROM Urunler GROUP BY KategoriID HAVING COUNT(\*) < 7

(ISNULL())

Parametre olarak verilen değeri NULL değer içeren hücrelere yerleştirmektedir.

Example

SELECT ISNULL(Bolge, 'Bölge yok.') FROM Musteriler

Constraints(Kısıtlayıcılar)

(DEFAULT CONSTRAINT)

Kolona bir değer girilmediğinde varsayılan olarak belirlediğimiz değerin kullanılmasını sağlamaktadır.

Example

ALTER TABLE Employees

ADD CONSTRAINT SalaryConstraint DEFAULT 8500 FOR Salary

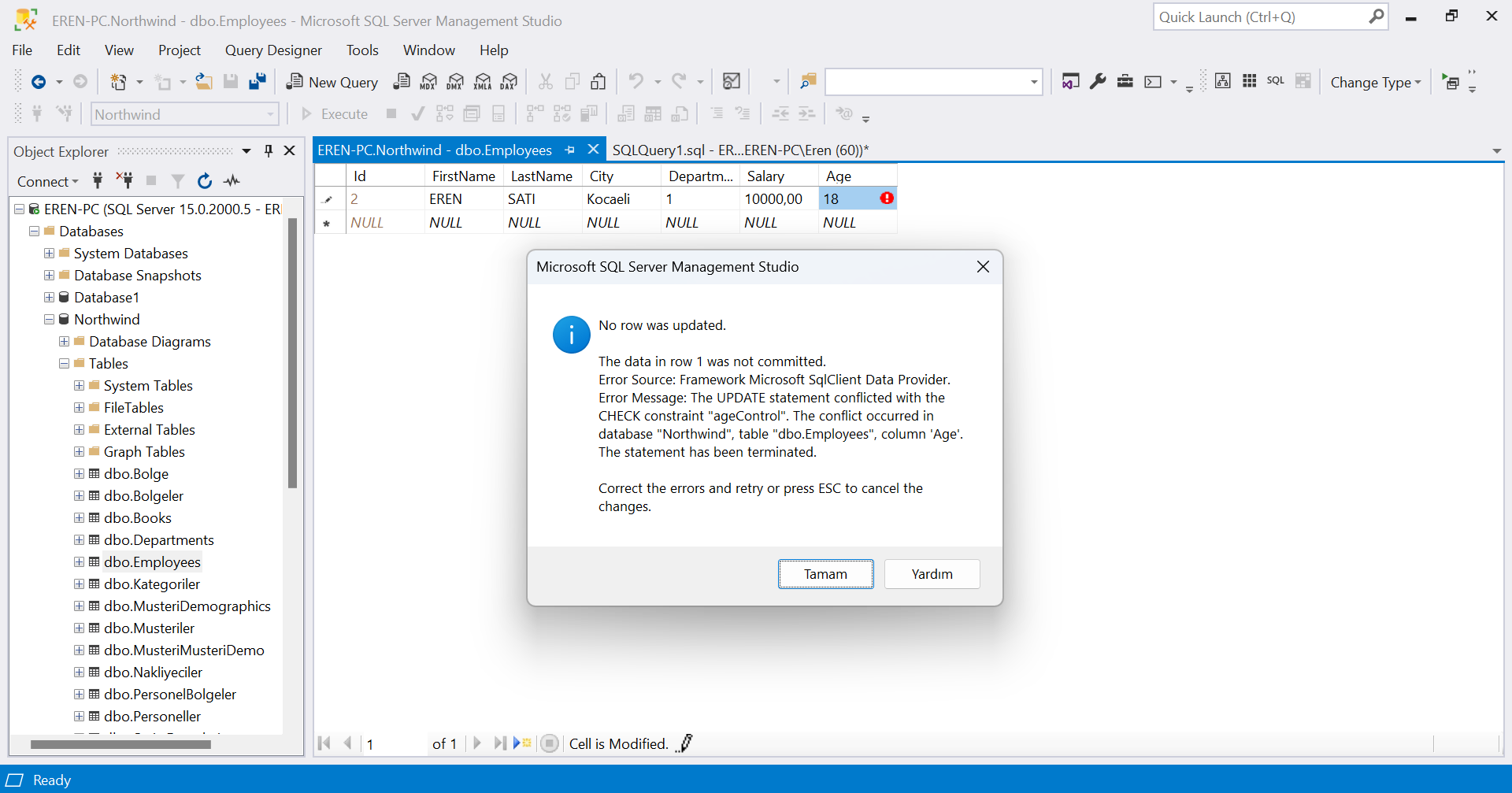
(CHECK CONSTRAINT)

Kolona girilecek olan verinin belirli bir şarta uymasını zorunlu tutmaktadır.

Example

ALTER TABLE Employees

ADD CONSTRAINT ageControl CHECK (Age > 18)



* Check constraint oluşturulmadan önce ilgili tabloda şarta aykırı değerler varsa constraint oluşturulmaz. Aykırı değerlerin görmezden gelinip check constraint’in oluşturulmasını istiyorsanız WITH NOCHECK komutunu kullanabilirsiniz.

Example

ALTER TABLE Employees

WITH NOCHECK ADD CONSTRAINT ageControl CHECK (Age > 18)

(PRIMARY KEY CONSTRAINT)

PRIMARY KEY olarak oluşturulan kolonlar başka tablolardaki FOREIGN KEY kolonlar ile ilişkilendirilebilmektedir. Bunun yanında o kolonun taşıdığı verinin tekil olacağı da garanti edilmiş olunur.

* Bir tablo içerisinde birden fazla PRIMARY KEY kullanılamaz.

Example

ALTER TABLE Employees

ADD CONSTRAINT Id PRIMARY KEY (Id)

(UNIQUE CONSTRAINT)

Belirttiğimiz kolondaki değerlerin tekil olmasını sağlamaktadır.

Example

ALTER TABLE Employees

ADD CONSTRAINT uniqueConstraint UNIQUE (LastName)

(FOREIGN CONSTRAINT)

Tabloların kolonları arasında ilişki kurmamızı sağlamaktadır.

Example

CREATE TABLE Employees

(

EmployeeId INT PRIMARY KEY IDENTITY(1, 1) NOT NULL,

FirstName NVARCHAR(255),

LastName NVARCHAR(255),

City NVARCHAR(255),

DepartmentId int

)

CREATE TABLE Departments

(

DepartmentId INT PRIMARY KEY IDENTITY(1, 1) NOT NULL,

Department NVARCHAR(255)

)

ALTER TABLE Employees

ADD CONSTRAINT foreignEmployeesDepartments FOREIGN KEY (DepartmentId) REFERENCES Departments(DepartmentId)

* CASCADE komutu : Ana tablodaki kayıt silindiğinde ya da güncellendiğinde ilişkili kolonların da silinmesini ve güncellenmesini sağlamaktadır.

Example

ALTER TABLE Employees

ADD CONSTRAINT foreignEmployeesDepartments FOREIGN KEY (DepartmentId) REFERENCES Departments(DepartmentId)

ON DELETE CASCADE

ON UPDATE CASCADE

* SET NULL parametresi : Ana tablodaki kayıt silindiğinde ya da güncellendiğinde ilişkili kolonların NULL olmasını sağlamaktadır.

Example

ALTER TABLE Employees

ADD CONSTRAINT foreignEmployeesDepartments FOREIGN KEY (DepartmentId) REFERENCES Departments(DepartmentId)

ON DELETE CASCADE SET NULL

ON UPDATE CASCADE SET NULL

* SET DEFAULT komutu : Ana tablodaki kayıt silindiğinde ya da güncellendiğinde ilişkili kolonların DEFAULT değeri kullanılmaktadır.

Example

ALTER TABLE Employees

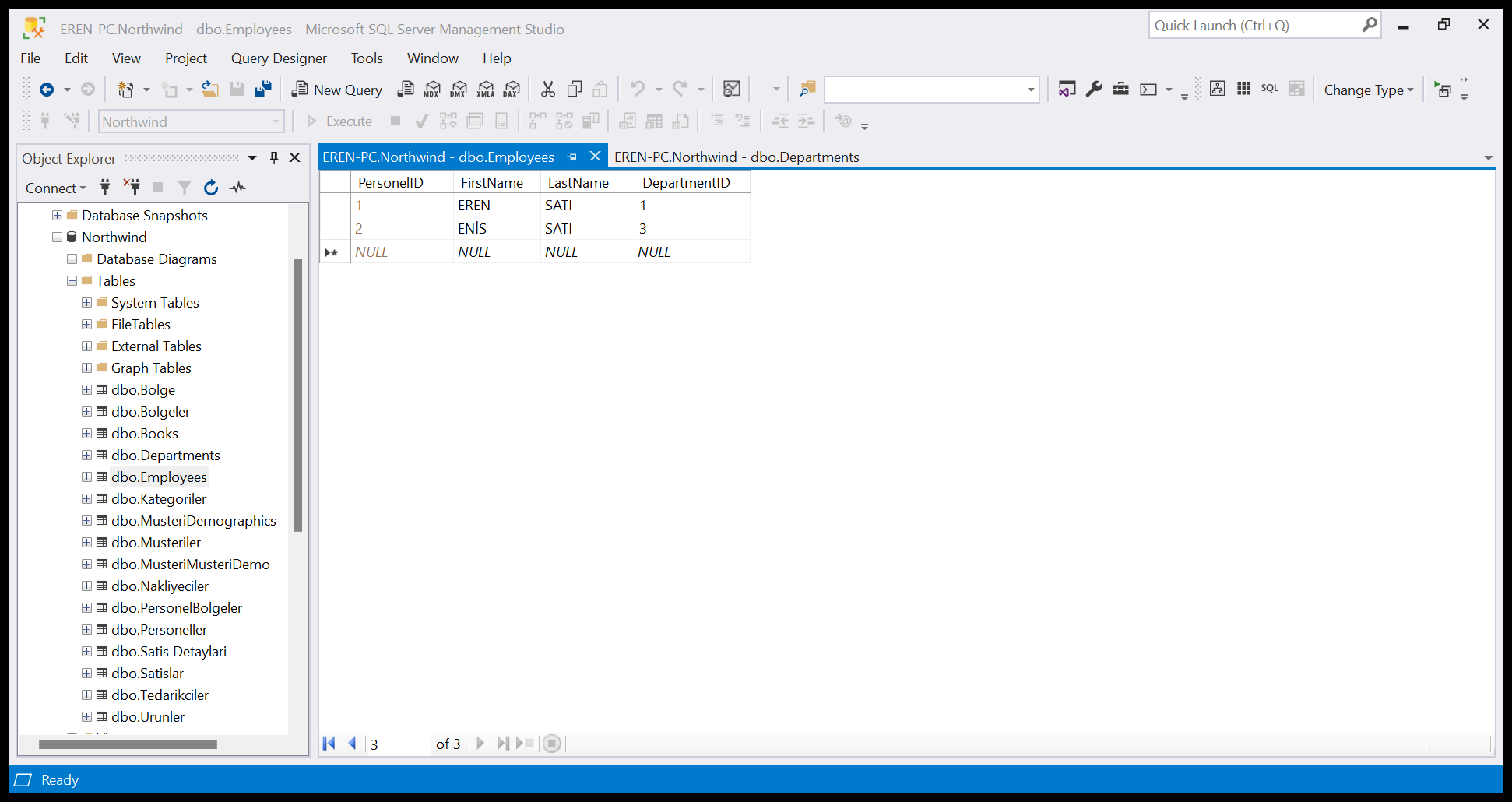
ADD CONSTRAINT foreignEmployeesDepartments FOREIGN KEY (DepartmentId) REFERENCES Departments(DepartmentId)

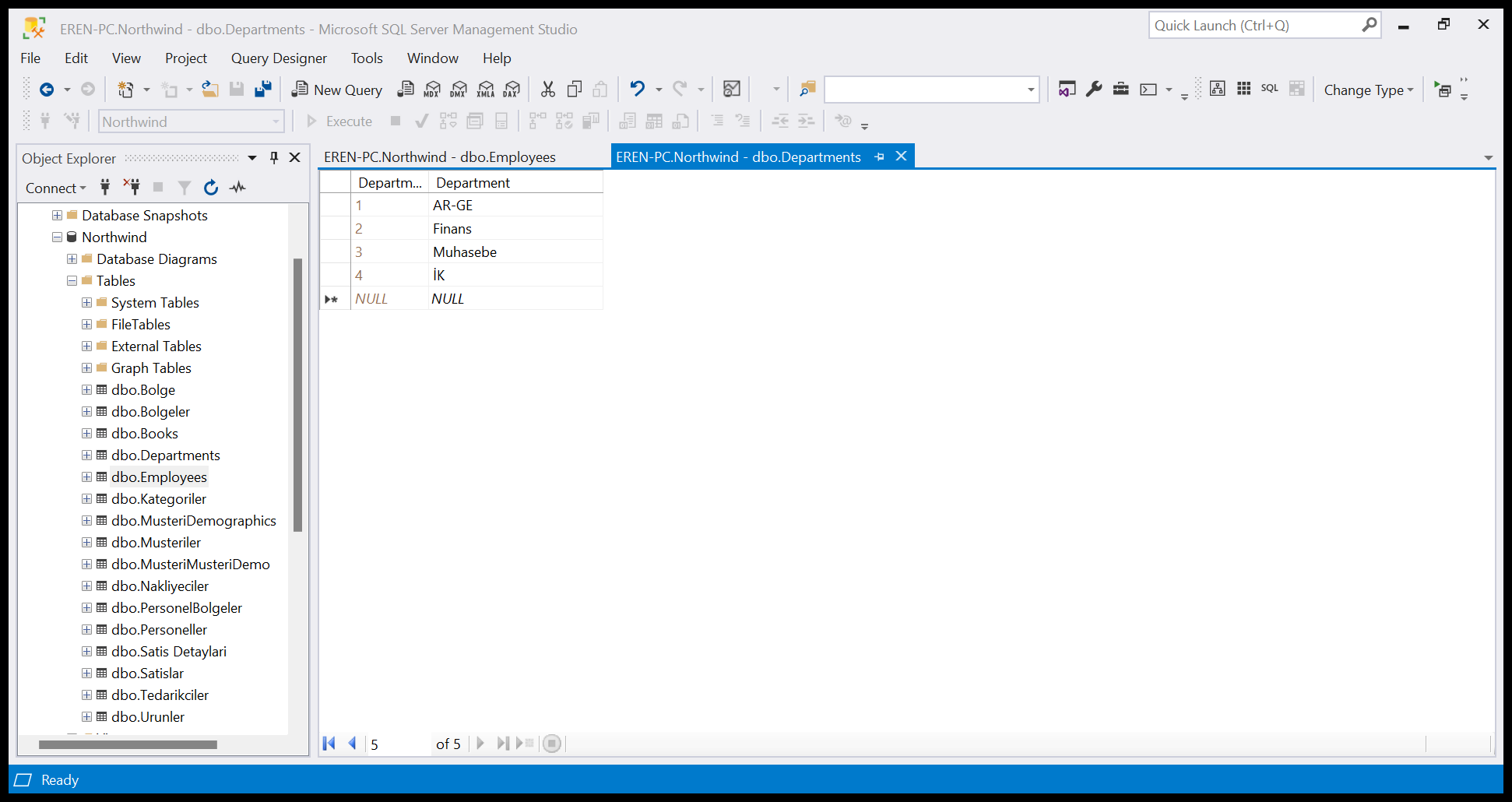
ON DELETE SET DEFAULT

ON UPDATE SET DEFAULT

JOIN

Örneklerimizde aşağıdaki Employees ve Departments tablolarını kullanacağız.

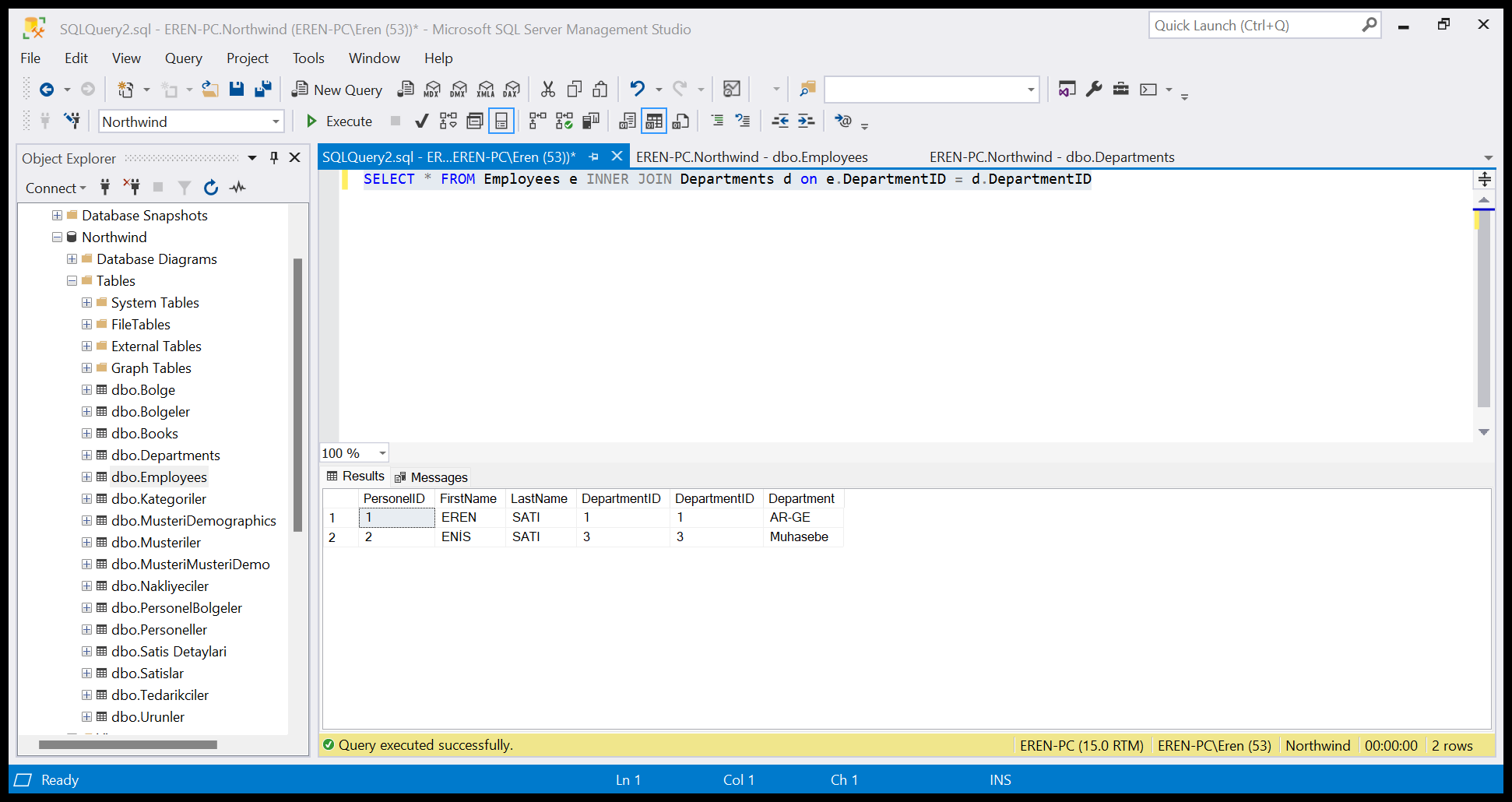




(INNER JOIN)

Example

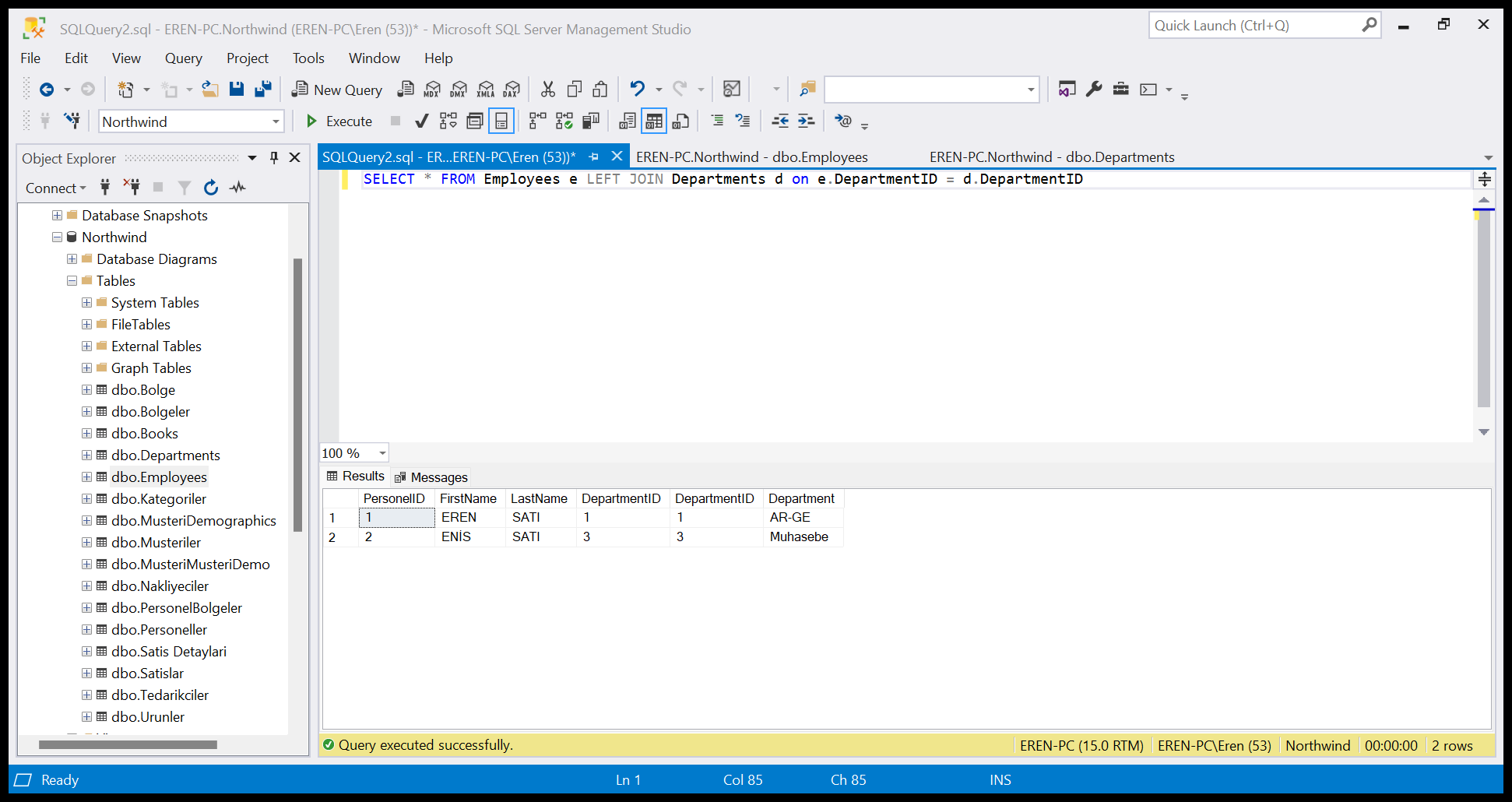
SELECT \* FROM Employees e INNER JOIN Departments d on e.DepartmentID = d.DepartmentID



(LEFT JOIN)

Example

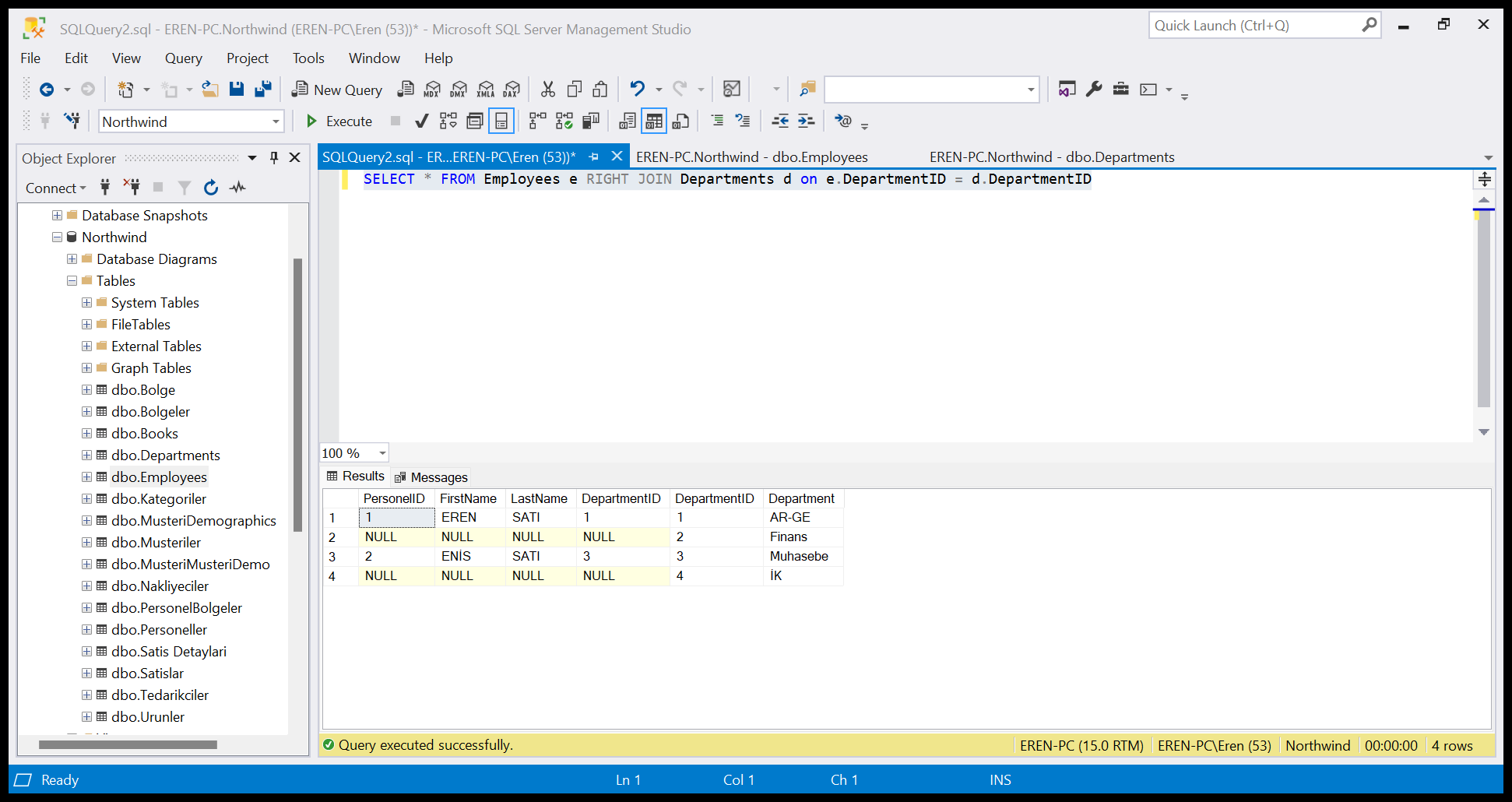
SELECT \* FROM Employees e LEFT JOIN Departments d on e.DepartmentID = d.DepartmentID



(RIGHT JOIN)

Example

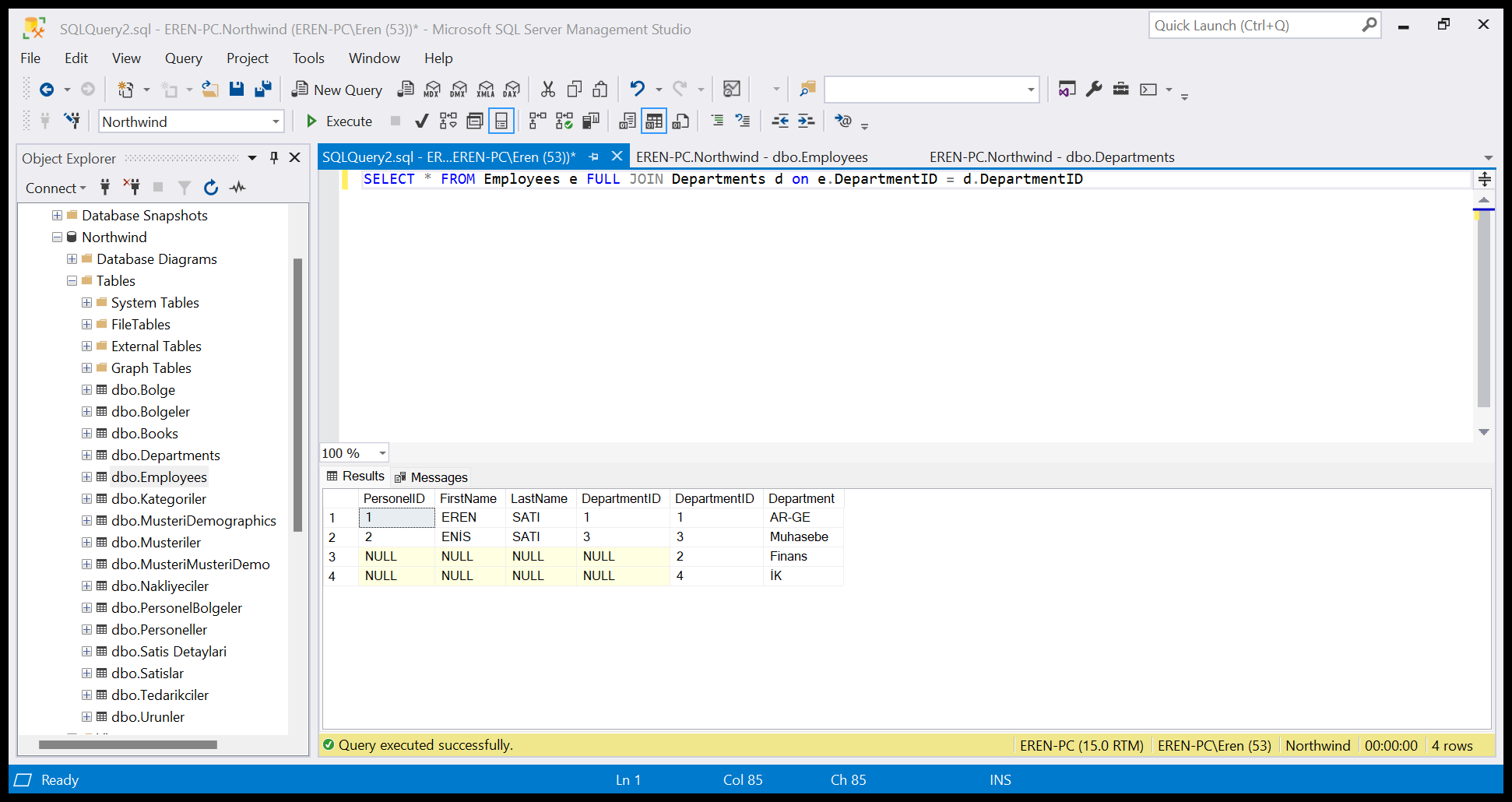
SELECT \* FROM Employees e RIGHT JOIN Departments d on e.DepartmentID = d.DepartmentID



(FULL JOIN)

Example

SELECT \* FROM Employees e FULL JOIN Departments d on e.DepartmentID = d.DepartmentID



Operators

|  |
| --- |
| Comparison Operators(Karşılaştırma) |
| =, !=, <>, >, <, >=, <= |

|  |
| --- |
| Logical Operators(Mantıksal) |
| AND, OR, NOT |

|  |
| --- |
| Arithmetic Operators(Aritmetik) |
| +, -, \*, /, % |

|  |
| --- |
| **NOT**  use komutu aktif database’i değiştirmek için kullanılmaktadır.  Example  use master |

|  |
| --- |
| **NOT**  GO komutu ile sorgularınızı sıra ile çalıştırabilirsiniz. |

Comment Lines

SQL komutları içerisinde yorum satırlarına yer vermek için -- sembolü kullanılmaktadır.

--

Variables

Değişkenler yerel ve global değişkenler olmak üzere ikiye ayrılmaktadır. Kullanıcılar tarafından oluşturulan değişkenler yerel değişken olarak adlandırılmaktadır. Global değişkenler ise SQL Server tarafından tanımlanmıştır ve bu değişkenlerin sadece içerikleri okunabilmektedir. Örnek vermek gerekirse **SELECT @@SERVERNAME** komutu ile ilgili değişkenin değeri okunabilmektedir.

* Management Studio’da New Query butonuna tıklayıp **SELECT @@** yazmanız durumunda global değişkenler listelenmektedir.
* Değişkenler **DECLARE** anahtar kelimesi ile tanımlanmaktadır.

Example

DECLARE @Number INT = 1

SELECT @Number --1

* Değer ataması sonradan yapılabilmektedir.

Example

DECLARE @Name NVARCHAR(255)

SET @Name = 'EREN'

SELECT @Name --EREN

* Aynı satırda birden fazla değişken tanımlanabilmektedir.

Example

DECLARE @i INT = 1, @j INT = 2

* SET anahtar kelimesi ile değer değişikliği yapabilirsiniz.

Example

DECLARE @isUpdated INT = 0

SET @isUpdated = 1

SELECT @isUpdated --1

* Bir sorgu sonucunda dönen kayıtları değişkenlere atayabilirsiniz.

Example

DECLARE @firstName NVARCHAR(MAX), @lastName NVARCHAR(MAX)

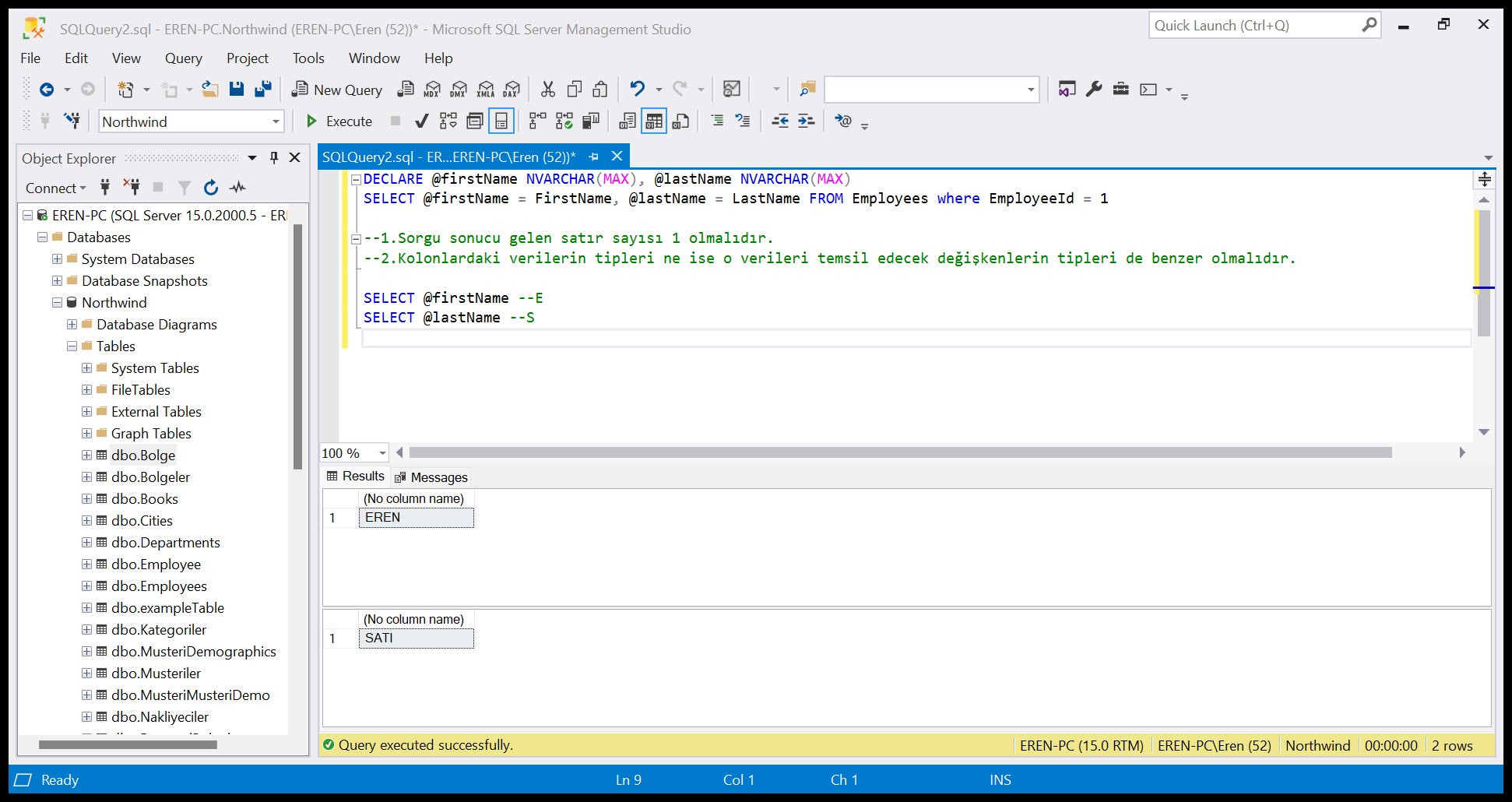
SELECT @firstName = FirstName, @lastName = LastName FROM Employees where EmployeeId = 1

--1.Sorgu sonucu gelen satır sayısı 1 olmalıdır.

--2.Kolonlardaki verilerin tipleri ne ise o verileri temsil edecek değişkenlerin tipleri de benzer olmalıdır.

SELECT @firstName --E

SELECT @lastName --S



Conditional Statements(IF-ELSE IF-ELSE)

Example

DECLARE @Number INT = 1

IF (@Number < 0)

BEGIN

PRINT 'sayı negatif.'

END

ELSE IF (@Number = 0)

BEGIN

PRINT '0'

END

ELSE

BEGIN

PRINT 'sayı pozitif.'

END

EXISTS

Bir sorgu sonucunda dönen kayıt olup olmadığını öğrenmek için kullanılmaktadır.

Example

DECLARE @firstName NVARCHAR(MAX) = 'EREN'

DECLARE @lastName NVARCHAR(MAX) = 'SATI'

IF EXISTS(SELECT \* FROM Employees WHERE FirstName = @firstName AND LastName = @lastName)

BEGIN

PRINT 'kayıt var'

END

ELSE

BEGIN

PRINT 'kayıt yok'

END

Loops(Döngüler)

Example

--1'den 9'a kadar olan sayıların toplamını vermektedir.

DECLARE @total INT = 0, @counter INT = 0

WHILE (@counter < 10)

BEGIN

SET @total += @counter

SET @counter += 1

END

PRINT @total --45

* BREAK deyimi döngüyü sonlandırmak için kullanılmaktadır.
* CONTINUE deyimi döngü içinde belirli bir koşulun kontrolü ile bir sonraki döngü adımına gidilmesini gerçekleştirir.

Example

DECLARE @counter INT = 0

WHILE (@counter < 10)

BEGIN

IF (@counter = 5)

BEGIN

SET @counter += 1

CONTINUE

END

PRINT @counter

SET @counter += 1

END

Views

İlişkili tabloları view üzerinde toplayarak tek bir tablo üzerinde çalışabilirsiniz.

Example

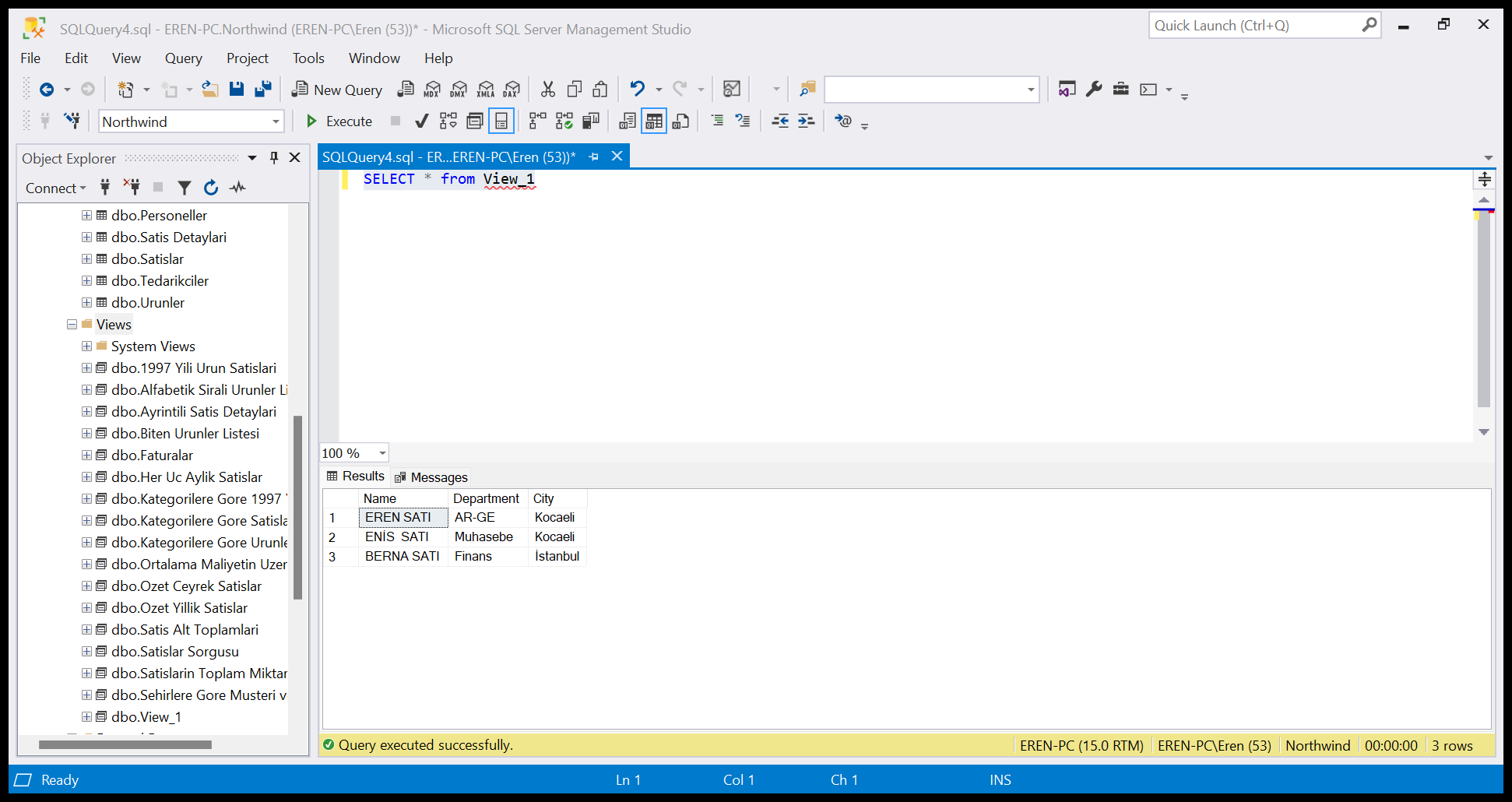
CREATE VIEW View\_1

AS

SELECT e.FirstName + ' ' + e.LastName AS Name, d.Department, c.City FROM Departments d INNER JOIN Employees e ON d.DepartmentId = e.DepartmentId INNER JOIN Cities c ON e.CityId = c.CityId

Example

SELECT \* from View\_1



* View oluşturulurken ORDER BY kullanılamaz. (Sorgu sırasında kullanılabilir)
* View’in kullandığı fiziksel tabloların kolon isimleri değiştirilir, kolonları silinir ya da tablo yapısı bir şekilde değişikliğe uğrar ise view’in çalışması artık mümkün olmayacaktır. Bu durumun önüne geçmek için view’in kullandığı tablolar koruma altına alınabilir. WITH SCHEMABINDING komutu bu amaçla kullanılmaktadır.

Example

CREATE TABLE exampleTable

(

Id INT PRIMARY KEY IDENTITY(1, 1) NOT NULL,

Column1 NVARCHAR(MAX)

)

CREATE VIEW exampleView

WITH SCHEMABINDING

AS

SELECT Id, Column1 FROM dbo.exampleTable --dbo öneki gereklidir.

Bu adımdan sonra artık aşağıdaki değişikliğe izin verilmeyecektir.

ALTER TABLE exampleTable

ALTER COLUMN Column1 INT

Functions

SQL Server’da fonksiyonlar ikiye ayrılmaktadır.

(Scalar Functions)

Geriye belirlediğimiz tipte değer döndürmektedir.

* Scalar fonksiyonlar Management Studio’da **Functions** sekmesi altındaki **Scalar-valued-Functions** klasörü içerisinde tutulmaktadır.

Example

CREATE FUNCTION scalarFunction

(

@Name NVARCHAR(MAX)

)

RETURNS NVARCHAR(MAX) --Geriye dönecek değerin tipi.

AS

BEGIN

RETURN @Name

END

* Scalar fonksiyonlar SELECT ya da PRINT komutları ile çağrılabilmektedir.

Example

SELECT dbo.scalarFunction('EREN') --dbo öneki gereklidir.

* ALTER komutu ile fonksiyonlar yeniden düzenlenebilmektedir.

Example

ALTER FUNCTION scalarFunction

(

@Number1 INT, @Number2 INT

)

RETURNS INT

AS

BEGIN

RETURN @Number1 + @Number2

END

SELECT dbo.scalarFunction(1, 2) --3

* RETURN ifadesinden sonra yazılan komutlar çalışmamaktadır.
* Fonksiyonlar DROP komutu ile silinebilmektedir.

Example

DROP FUNCTION scalarFunction

(Inline Functions)

Geriye bir tablo döndürmektedir.

* Inline fonksiyonlar Management Studio’da **Functions** sekmesi altındaki **Table-valued-Functions** klasörü içerisinde tutulmaktadır.

Example

CREATE FUNCTION inlineFunction

(

@firstName NVARCHAR(MAX), @lastName NVARCHAR(MAX)

)

RETURNS TABLE

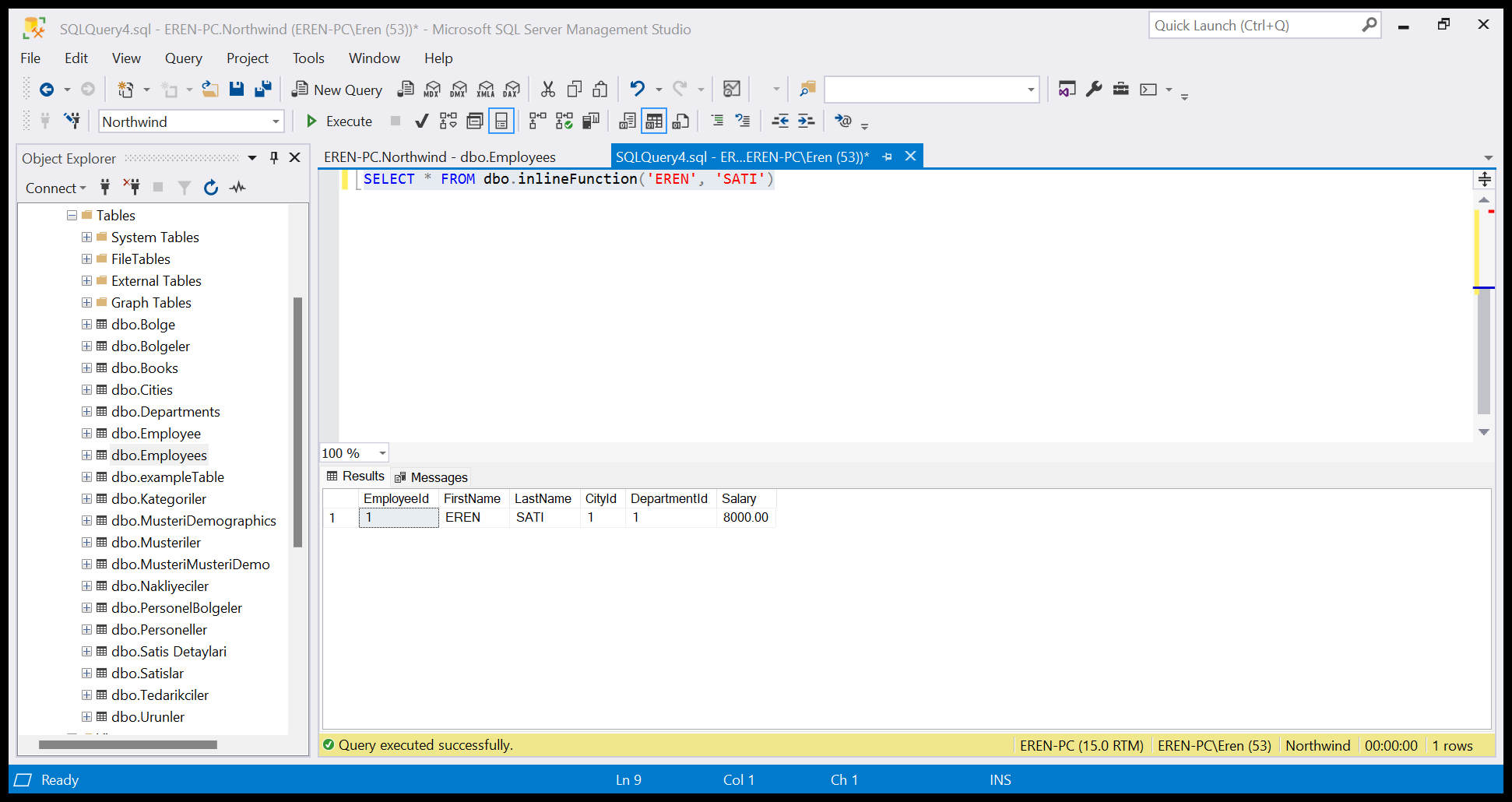
AS

RETURN SELECT \* FROM Employees WHERE FirstName = @firstName AND LastName = @lastName

* Inline fonksiyonlar SELECT komutu ile çağrılmaktadır.

Example

SELECT \* FROM dbo.inlineFunction('EREN', 'SATI')



|  |
| --- |
| **NOT**  Yukarıdaki örnekte fonksiyona parametre olarak verdiğimiz değerler tablo içerisinde olmasaydı geriye boş bir tablo dönerdi. |

(SELECT Sorgusu İçerisinde Fonksiyon Kullanmak)

Example

CREATE FUNCTION addNumbers

(

@Number1 INT, @Number2 INT

)

RETURNS INT

AS

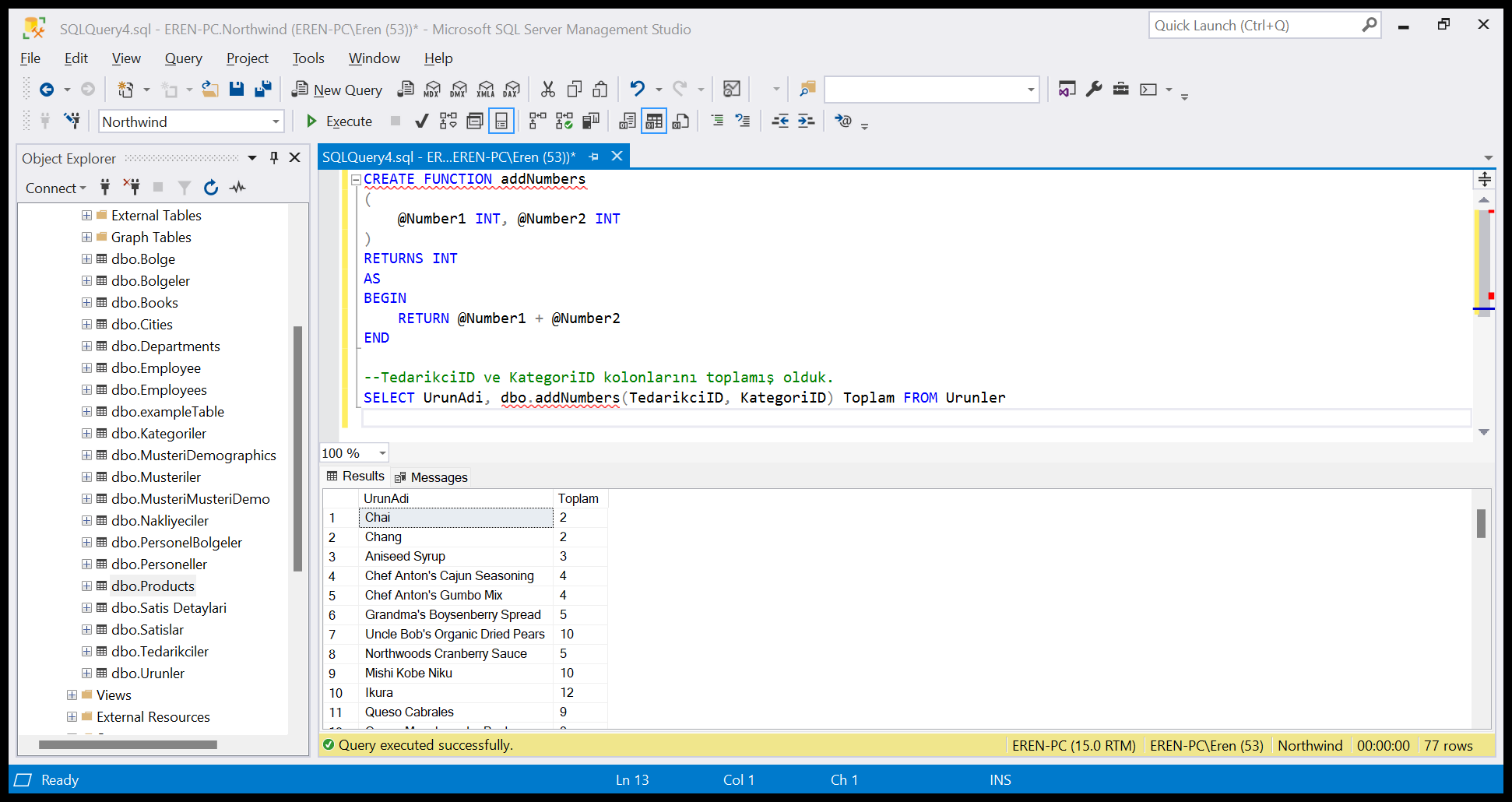
BEGIN

RETURN @Number1 + @Number2

END

--TedarikciID ve KategoriID kolonlarını toplamış olduk.

SELECT UrunAdi, dbo.addNumbers(TedarikciID, KategoriID) Toplam FROM Urunler



Stored Procedure

* Stored procedure’ler normal sorgulardan daha hızlı çalışmaktadır.
* İçerisinde CRUD işlemleri yapılabilmektedir.
* Veritabanımıza ait stored procedure’ler Management Studio’da **Programmability** sekmesi altındaki **Stored Procedures** klasörü içerisinde bulunmaktadır.

Example

CREATE PROC sp\_Test

AS

BEGIN

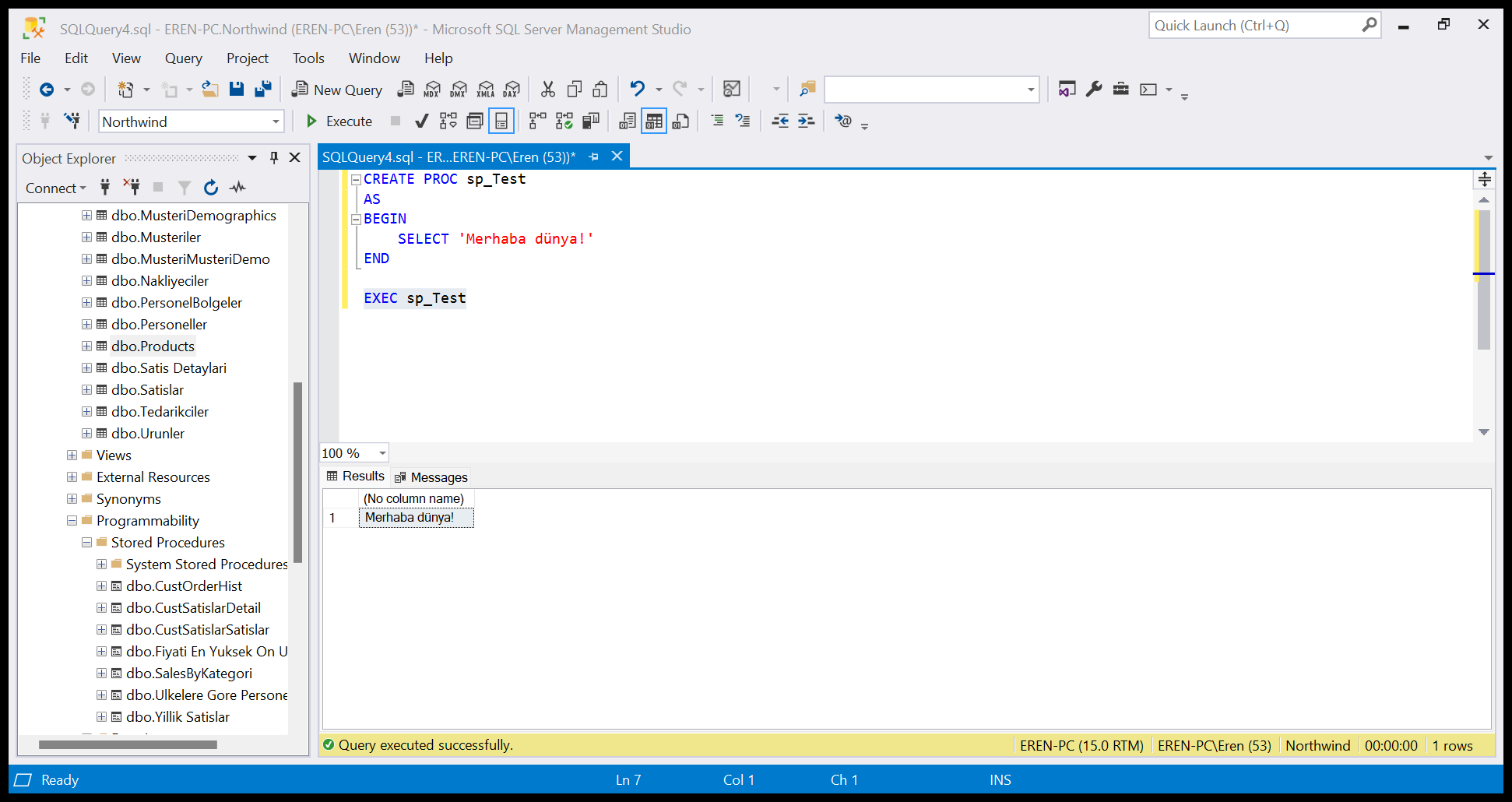
SELECT 'Merhaba dünya!'

END

* Stored procedure’ler EXEC komutu ile çağrılmaktadır.

Example

EXEC sp\_Test



Example

ALTER PROC sp\_Test

(

@Number1 INT,

@Number2 INT

)

AS

BEGIN

SELECT @Number1 + @Number2

END

EXEC sp\_Test 1, 2

* DROP komutu ile stored procedure’leri silebilirsiniz.
* Parametrelere varsayılan değerler verebilirsiniz.

Example

CREATE PROC sp\_addNumbers

(

@Number1 INT = 1,

@Number2 INT = 2

)

AS

BEGIN

SELECT @Number1 + @Number2

END

Aggregate Functions

(AVG)

Ortalama alır.

Example

SELECT AVG(Price) FROM Books

(MAX)

En büyük değeri döndürür.

Example

SELECT MAX(Price) FROM Books

(MIN)

En küçük değeri döndürür.

Example

SELECT MIN(Price) FROM Books

(COUNT)

Toplam kayıt sayısını döndürür.

Example

SELECT COUNT(Price) FROM Books

(SUM)

Toplama işlemi yapar.

Example

SELECT SUM(Price) FROM Books

String Functions

(CHAR())

Parametre olarak verilen ASCII kodun karakterini döndürür. Geriye dönen bilgi CHAR türündedir.

Example

SELECT CHAR(65) --A

(ASCII())

Parametre olarak verilen karakterin ASCII kodunu döndürür. Geriye dönen bilgi INT türündedir.

Example

SELECT ASCII('A') --65

(CHARINDEX())

Metinsel ifadenin içerisinde arama yapmak için kullanılır. Parametre olarak verilen bilginin ilk geçtiği yerin index numarasını INT türünde döndürmektedir. Aranan bilgi bulunamazsa 0 değeri döner.

* SQL’de index numaraları 1’den başlamaktadır.
* Üçüncü parametrenin kullanımı opsiyoneldir başlangıç index’ini belirtmek için kullanılır.

Example

SELECT CHARINDEX('Üniversite', 'Trakya Üniversitesi') --8

Example

SELECT CHARINDEX('Üniversite', 'Trakya Üniversitesi', 9) --0

(LEFT(), RIGHT())

Metinsel ifadenin başından ya da sonundan istediğimiz kadar karakter getirir. Geriye dönen bilgi NVARCHAR türündedir.

Example

DECLARE @Address NVARCHAR(MAX)='YENİŞEHİR MAH. BERKCAN SK. İZMİT'

SELECT LEFT(@Address, 9) --YENİŞEHİR

SELECT RIGHT(@Address, 5) --İZMİT

(LEN())

Metinsel ifadenin uzunluğunu döndürür. Geriye dönen bilgi INT türündedir.

Example

DECLARE @Name NVARCHAR(MAX)='EREN'

SELECT LEN(@Name) --4

(LOWER(), UPPER())

Metnin büyük ya da küçük harflerle gösterilmesini sağlamaktadır.

(LTRIM(), RTRIM())

Metinsel ifadenin solundaki ya da sağındaki boşlukları temizlemek için kullanılır.

(REVERSE())

Parametre olarak verilen dizeyi tersten sıralar.

(SUBSTRING())

Belirlediğimiz konumdan belli sayıda karakter getirir. İkinci parametre başlangıç index’i üçüncü parametre karakter sayısıdır.

(REPLACE())

Example

SELECT REPLACE('eee', 'e', 'a') --aaa

Date Functions

(GETDATE())

Tarihi ve saati verir. Geriye dönen bilgi DATETIME türündedir.

Example

SELECT GETDATE() --2023-01-19 15:54:05.680

(DATEADD())

Bir tarihe ya da saate ekleme veya çıkarma yapmak için kullanılır. Geriye dönen bilgi SMALLDATETIME türündedir.

Example

--Tarihten 1 hafta çıkarır.

SELECT DATEADD(week, -1, GETDATE()) --2023-01-12 15:57:48.040

Example

--Tarihe 3 yıl ekler.

SELECT DATEADD(year, 3, GETDATE()) --2026-01-19 15:59:32.220

Example

--Parametre olarak verilen tarihe 1 ay ekler.

SELECT DATEADD(month, 1, CONVERT(date, '2018-01-01')) --2018-02-01

Yukarıdaki örnekte tırnak içinde yazılan tarih varchar türünde olduğu için tip dönüşümü yaptık.

(DATEDIFF())

İki tarih arasındaki gün, hafta, ay, yıl, saat, dakika ya da saniye sayısını bulmak için kullanılır. Geriye dönen bilgi INT türündedir.

Example

--2015 yılından bugüne kaç gün geçtiğini döndürür.

SELECT DATEDIFF(day, CONVERT(date, '2015-01-01'),GETDATE()) --2940

Example

--1 OCAK 2023 tarihinden bugüne kaç saat geçtiğini döndürür.

SELECT DATEDIFF(hour, CONVERT(datetime, '2023-01-01 00:00:00.000'), GETDATE()) --448

(DATEPART())

Bir tarihten gün, ay, yıl, saat, dakika vb. bilgileri almamızı sağlamaktadır. Geriye dönen bilgi INT türündedir.

Example

SELECT GETDATE() --Bugünün tarihi : 2023-01-19 16:09:55.087

SELECT DATEPART(day, GETDATE()) --19

SELECT DATEPART(month, GETDATE()) --1

SELECT DATEPART(year, GETDATE()) --2023

SELECT DATEPART(hour, GETDATE()) --16

SELECT DATEPART(minute, GETDATE()) --09

SELECT DATEPART(second, GETDATE()) --55

Example

--Yılın kaçıncı gününde olduğumuzu döndürür.

SELECT DATEPART(dayofyear, GETDATE())

Example

--Haftanın kaçıncı gününde olduğumuzu döndürür. Haftanın ilk günü Pazar'dır.

SELECT DATEPART(WEEKDAY, GETDATE())

Example

--Yılın kaçıncı haftasında olduğumuzu döndürür.

SELECT DATEPART(WEEK, GETDATE())

(ISDATE())

Parametre olarak verilen değer tarih ise 1 değilse 0 döndürür. Geriye dönen bilgi INT türündedir.

Example

SELECT ISDATE('2018-01-01') --1

SELECT ISDATE('25-01-2018') --0

SELECT ISDATE('12-25-1993') --1

SELECT ISDATE('25.12.1993') --0

Other Functions

(CONVERT())

Tip dönüşümü yapmak için kullanılmaktadır.

Example

--İkinci parametredeki char bilgiyi money türüne dönüştürmektedir.

SELECT CONVERT(money, '10') --10,00

* Tarih türündeki veriler char türüne dönüştürülürken üçüncü parametre ile tarih formatı ayarlanabilmektedir. Üçüncü parametreye 100 ile 114 arası değer verebilirsiniz.

Example

SELECT CONVERT(CHAR, GETDATE(), 100) --Jan 19 2023 4:24PM

SELECT CONVERT(CHAR, GETDATE(), 101) --01/19/2023

SELECT CONVERT(CHAR, GETDATE(), 102) --2023.01.19

SELECT CONVERT(CHAR, GETDATE(), 103) --19/01/2023

(ISNUMERIC())

Parametre olarak verilen değer sayı ise 1 değilse 0 döner. Geriye dönen bilgi INT türündedir.

Example

SELECT ISNUMERIC(1) --1

SELECT ISNUMERIC('1') --1

SELECT ISNUMERIC('') --0

(NEWID())

Benzersiz bir değer üretmektedir.

Example

SELECT NEWID() --5B731987-CA2C-4C5E-912A-A7AC01E32F1B