

Transact-SQL

SQL NEDİR?	4
Transact-SQL	4
SQL Veri İşleme Dili (Data Manipulation Language-DML)	4
SQL Veri Tanımlama Dili (Data Definition Language-DDL).....	4
SQL Veri Kontrol Dili (Data Control Language-DCL).....	4
SELECT İfadesi	5
Belirli Sütunların Görüntülenmesi.....	5
Tüm Sütunları Görüntülemek	5
WHERE Yantümcesi	6
Operatörler (İşleçler).....	6
LIKE ifadesi	7
SQL AND & OR & NOT MANTIKSAL OPERATÖRLERİ	9
SQL Between...And	10
Bir Listedeki Elemanların Aranması – IN.....	11
Boş Değerlerin Görüntülenmesi- NULL Değerler	11
SQL Select Distinct	12
SQL Order By	12
Matematiksel & Metinsel İfadeler	13
SQL Count Fonksiyonları	14
COUNT(*).....	14
COUNT(<sütun adı>)	15
COUNT DISTINCT.....	15
SÜTUNLARIN ve TABLOLARIN YENİDEN ADLANDIRILMASI - AS	16
SQL TOPLAM FONKSİYONLARI (AGGREGATE FUNCTIONS)	16
AVG(<sütun adı>) Fonksiyonu	16
MAX(<sütun adı>) Fonksiyonu	16
MIN(<sütun adı>) Fonksiyonu	16
SUM(<sütun adı>) Fonksiyonu	17
İlk n Kaydın Görüntülenmesi - TOP n	17
WITH TIES	17
GROUP BY.....	19
HAVING YANTÜMCESİ.....	20
GROUP BY İLE ÖZET BİLGİ.....	22
ROLLUP	22
CUBE	23
GROUPING FONKSİYONU	24
COMPUTE ve COMPUTE BY	25
SQL Join	27
INNER JOIN	27
OUTER JOIN	28
LEFT JOIN.....	28
RIGHT JOIN	29
CROSS JOIN	30

İkiden Fazla Tablonun Birleştirilmesi	31
SELF-JOINS	32
UNION.....	33
ALT SORGULAR(SUBQUERIES).....	34
Sonuç Kümesinin Tablo Olarak Kullanımı:.....	34
Sonuç Kümesinin Bir Deyim Olarak Kullanılması:	34
İlişkili Alt Sorgu	35
ALT SORU-JOIN DÖNÜŞÜMÜ	37
ALT SORU – HAVING DÖNÜŞÜMÜ	38
EXISTS ve NOT EXISTS İŞLEVLERİ:	38
INSERT... VALUE İŞLEMİ	39
INSERT...SELECT.....	40
INSERT INTO... VALUES.....	41
Update Deyimi	42
Delete Deyimi	42

SQL NEDİR?

SQL, ingilizce açık adıyla "Structured Query Language"(Yapılandırılmış Sorgulama Dili) bir veritabanı sorgu dilidir. SQL ile veritabanına yeni tablolar, yeni kayıtlar ekleyip, silebilir, var olanlar üzerinde düzenlemeler ve sorgular yapabilirsiniz. SQL ile Oracle, db2, Sybase, Informix, Microsoft SQL Server, MS Access gibi veritabanı yönetim sistemlerinde çalışabilirsiniz. SQL, standard bir veritabanı sorgu dilidir, bütün gelişmiş veritabanı uygulamalarında kullanılır.

Transact-SQL

SQL, düzeltilmesi veya değiştirilmesi istenen bilgileri açıkça belirtmeye izin veren ve yerine getirilebilecek başlıca işlemleri tanımlamamızı sağlayan bir komut takımıdır. Bu komutların oluşturduğu yapıya Transact-SQL dili denir. Transact-SQL ile veri ve sorgulara erişebilir, güncelleyebilir ve ilişkisel veritabanı sistemini yönetebilirsiniz. Transact –SQL komutları kullanım amaçlarına göre üç genel kategoriye ayrılır.

SQL Veri İşleme Dili (Data Manipulation Language-DML)

SQL Veri İşleme Dili veri girmek, değiştirmek, silmek ve verileri almak için kullanılan DML komutlarının tümüdür. En sık kullanılan DML komutları ve kullanım amaçları aşağıdaki gibidir.

SELECT	: Veri seçmek
DELETE	: Veri silmek
UPDATE	: Veri güncellemek
INSERT	: Veri girmek

SQL Veri Tanımlama Dili (Data Definition Language-DDL)

SQL Veri Tanımlama Dili verilerin tutulduğu nesneler olan tabloların yaratılmasını, silinmesini ve bazı temel özelliklerinin düzenlenmesini sağlar. En sık kullanılan bazı DDL komutları ve kullanım amaçları aşağıdaki gibidir.

CREATE TABLE	: Yeni bir tablo yaratmak
ALTER TABLE	: Tabloda değişiklik yapmak
DROP TABLE	: Tabloyu silmek
CREATE INDEX	: Tabloda dizin oluşturmak

SQL Veri Kontrol Dili (Data Control Language-DCL)

SQL Veri Kontrol Dili bir veritabanı kullanıcısı veya rolü ile ilgili izinlerin düzenlenmesini sağlar. Aşağıdaki tablo DCL komutlarını ve fonksiyonlarını göstermektedir.

GRANT	:Kullanıcıya yetki vermek
DENY	:Kullanıcı, grup veya rolü herhangi bir eylem için engeller.
REVOKE	:Daha atanmış olan yetki veya engeli kaldırır.

SELECT İfadesi

Select ifadesi bir tablodan verileri seçmek için kullanılır. Elde edilen veriler sonuç kümesi olarak adlandırılır ve yine bir tablo görüntüsü şeklinde görüntülenir.

Söz dizimi:

SELECT <sütun adı>

FROM <tablo adı>

Belirli Sütunların Görüntülenmesi

Çoğu durumda tablomuzda sakladığımız tüm veriyi görmektense, o an için ihtiyaç duyduğumuz kısmını görüntülemek isteriz. Bunun için Select ifadesini takiben, kullanılacak sütun adları belirtilir.

Örnek: Northwind veritabanındaki Employees tablosunda çalışanlar ile ilgili bilgiler saklanır. Bize personelimizin sadece telefon numaraları gerekli olduğunda aşağıdaki alanları görüntülememiz yeterlidir.

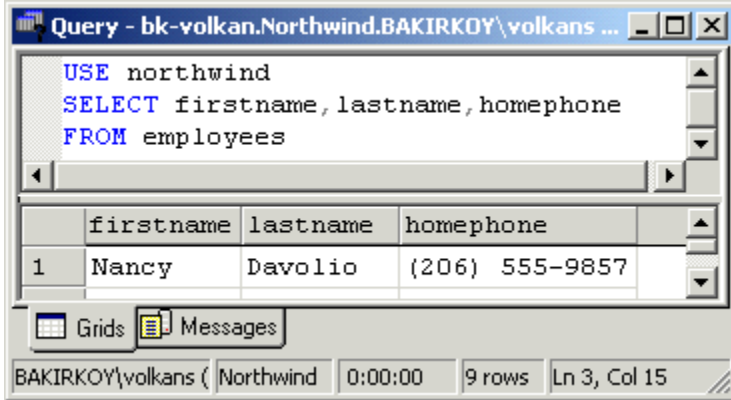
Sorgu:

USE northwind

SELECT firstname, lastname, homephone

FROM employees

Sonuç:

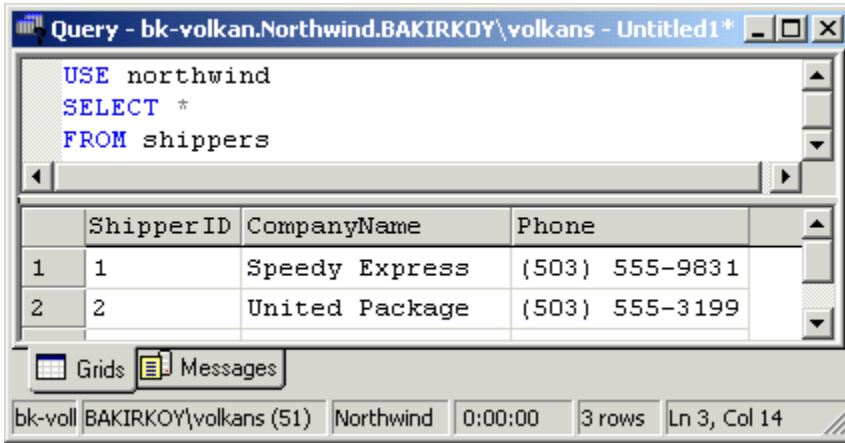


	firstname	lastname	homephone
1	Nancy	Davolio	(206) 555-9857

Tüm Sütunları Görüntülemek

Bir tablodaki tüm alanları görüntülemek için ise tek tek bütün sütun adlarını belirtmekten daha kolay bir yol vardır. Select ifadesinden sonra asterix(*) işaretini yazdığınızda SQL bunu 'tablodaki tüm sütunlar' olarak yorumlayacaktır.

Sonuç:



The screenshot shows a SQL query window titled "Query - bk-volkan.Northwind.BAKIRKOY\volkans - Untitled1*". The query text is:

```
USE northwind
SELECT *
FROM shippers
```

Below the query text, a table of results is displayed with the following columns: ShipperID, CompanyName, and Phone. The table contains two rows of data:

	ShipperID	CompanyName	Phone
1	1	Speedy Express	(503) 555-9831
2	2	United Package	(503) 555-3199

At the bottom of the window, there are tabs for "Grids" and "Messages". The status bar at the bottom shows "bk-voll BAKIRKOY\volkans (51) Northwind 0:00:00 3 rows Ln 3, Col 14".

WHERE Yantümcesi

WHERE yantümcesi görüntülemek istediğimiz verileri belirli bir kritere göre seçebilmemizi sağlar. Bu da, örneğin günlük satış bilgilerinin tutulduğu, 100.000 kayıttan oluşan satışlar tablosundaki sadece son 3 gün içerisindeki satışları görmemizi sağlar.

Where yantümcesinin kullanımı zorunlu değildir. Fakat bir koşula bağlı veri seçmek istediğimizde FROM yantümcesinden sonra eklenebilir.

Söz dizimi: `SELECT <sütun adı> FROM <tablo adı>`
`WHERE <koşul(lar)>`

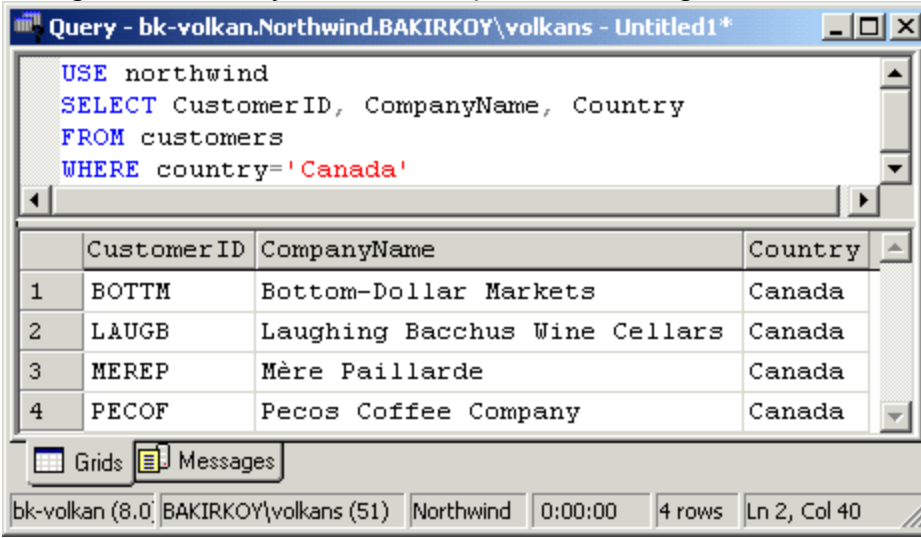
Operatörler (İşleçler)

Aşağıdaki operatörleri WHERE ifadesi ile birlikte sınanmasını istediğiniz koşulları oluştururken kullanabilirsiniz.

Operator	İfade
=	Eşittir
<>	Eşit Değildir, Farklı
>	Büyük
<	Küçük
>=	Büyük veya Eşit
<=	Küçük veya Eşit
BETWEEN	Arasında
LIKE	Metin arama

Not: SQL'in bazı sürümlerinde <> operatörü != şeklinde de kullanılabilir.

Örnek: Northwind veritabanında Customers tablosundan, Kanada daki müşterileri görüntülemek istediğimizde Country alanına bir koşul belirtmemiz gerekecektir.



Aşağıdaki örnek kodları çalıştırıp sonuçlarını inceleyiniz.

Birim fiyatı 20\$ dan büyük olan ürünleri listelemek için Ürünler (Products) tablosunda UnitPrice alanı için bir koşul belirtmemiz gerekir.

```
USE northwind
SELECT ProductID,ProductName,UnitPrice
FROM products
WHERE unitprice>20
```

01.01.1998 tarihinden itibaren verilmiş olan siparişleri görüntülemek için;

```
USE northwind
SELECT OrderID,CustomerID,OrderDate
FROM Orders
WHERE OrderDate>'1/1/1998'
```

LIKE ifadesi

LIKE kelimesi sütundaki değerlerin, joker karakterler kullanılarak oluşturduğumuz bir arama koşulu ile karşılaştırılmasını sağlar.

Söz dizimi: **SELECT <sütun adı> FROM <tablo adı>**
WHERE <aranacak sütun> LIKE <kriter>

Joker Karakter	Anlamı
%	Herhangi uzunlukta karakter
_	Herhangi tek karakter
[]	Belirtilen aralıkta herhangi bir karakter
[^]	Belirtilen aralıkta olmayan herhangi bir karakter

Örnek: Müşterilerimiz içerisinde ilk harfi K olanları listelemek istediğimizde;

Sorgu:

```
USE northwind
SELECT * FROM Customers
WHERE CustomerID LIKE 'K%'
```

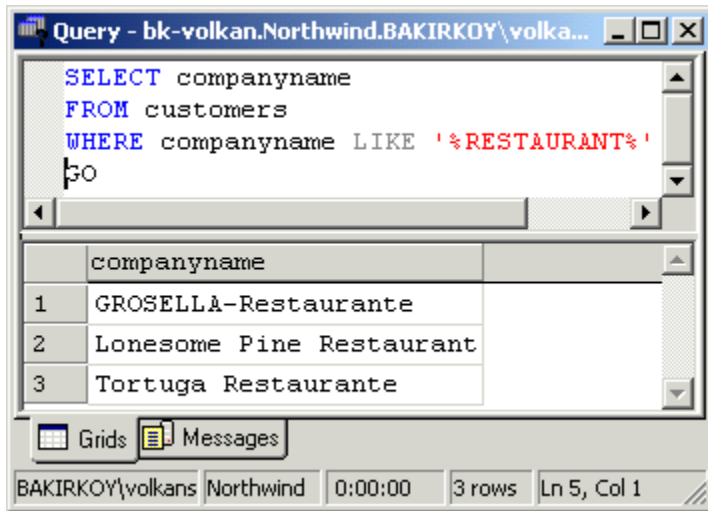
Örnek arama koşulları:

LIKE 'BR%'	İlk iki harfi BR olan tüm kayıtlar
LIKE 'Br%'	İlk iki harfi Br olan tüm kayıtlar
LIKE '%een'	Son üç harfi een olan tüm kayıtlar
LIKE '%en%'	İçerisinde en ifadesi geçen tüm kayıtlar
LIKE '_en'	Son iki harfi en olan üç harften oluşan tüm kayıtlar
LIKE '[CK]%'	C veya K harfleriyle başlayan tüm kayıtlar
LIKE '[S-V]ing'	ing ile biten ve ilk harfi S ile V harfleri arasında olan dört harfli herhangi kelime
LIKE 'M[^c]%'	M ile başlayan ikinci harfi c olmayan tüm kayıtlar

Örnek: Aşağıdaki örnekte firma adı içerisinde “restaurant” kelimesi geçen müşteriler listelenmiştir.

Sorgu:

```
USE northwind
SELECT companyname
FROM customers
WHERE companyname LIKE '%RESTAURANT%'
GO
```



SQL AND & OR & NOT MANTIKSAL OPERATÖRLERİ

AND ve OR Mantıksal operatörlerini birden fazla koşulu birleştirmek için kullanırız.

AND operatörü kullanarak, birleştirilen koşulların tümüne uyan satırlar listelenir.

OR operatörü kullanılarak, birleştirilen koşullardan en az birine uyan satırlar listelenir.

NOT operatörü kendisinden sonra gelen koşulu *sağlamayan* kayıtları listeler.

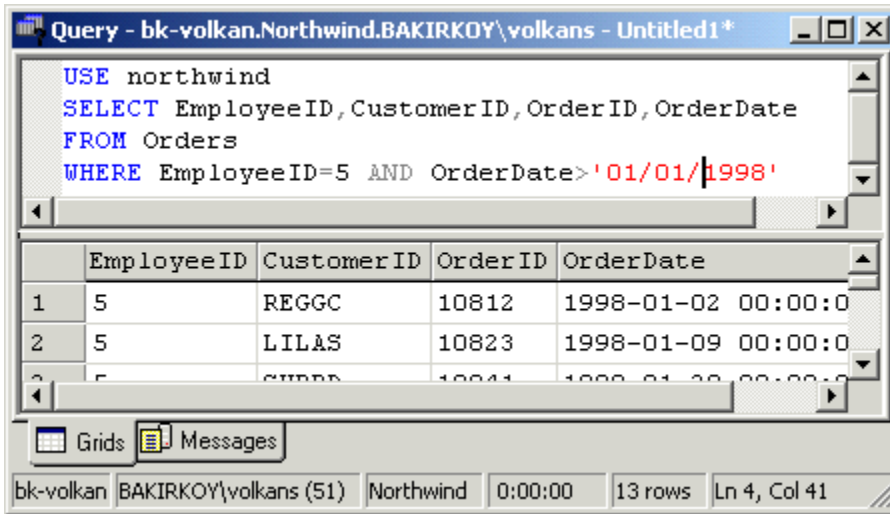
Parantez kullanımı ikiden fazla koşul olması durumunda koşulların öncelik sırasını belirler.

Parantez kullanılmaması durumunda SQL önce NOT, sonra AND ve en son da OR mantıksal operatörünü işler.

Söz dizimi: SELECT <sütun adı> FROM <tablo adı>
WHERE <koşul> AND <koşul>[AND <koşul>...]

SELECT <sütun adı> FROM <tablo adı>
WHERE <koşul> OR <koşul> [OR <koşul>...]

Örnek: Personel nosu 5 olan çalışanın 1998 yılından sonra aldığı siparişleri listelemek için kontrol edilmesi gereken iki koşul vardır.



Örnek: Berlin'deki veya Amerika'daki üreticileri listelemek istediğimizde;

Sorgu:

Sonuç:

```
USE northwind
SELECT
SupplierID,CompanyName,City,C
ountry
FROM Suppliers
WHERE Country='USA' OR
City='Berlin'
ORDER BY city,country
```

	SupplierID	CompanyName	City	Country
1	3	Grandma Kelly's H...	Ann Arbor	USA
2	16	Bigfoot Breweries	Bend	USA
3	11	Heli Süßwaren Gmb...	Berlin	Germany
4	19	New England Seafo...	Boston	USA
5	2	New Orleans Cajun...	New Orleans	USA

Örnek: 1 veya 2 nolu üreticilerin 18\$ dan pahalı ürünlerini listelemek istersek;

Sorgu:

```
USE northwind
SELECT
ProductName,SupplierID,UnitPrice
FROM Products
WHERE (SupplierID=1 OR
SupplierID=2)
AND UnitPrice>18
```

Sonuç:

	ProductName	SupplierID	UnitPrice
1	Chang	1	19.0000
2	Chef Anton's Cajun Seasoning	2	22.0000
3	Chef Anton's Gumbo Mix	2	21.3500
4	Louisiana Fiery Hot Pepper ...	2	21.0500

SQL Between...And

BETWEEN ... AND operatörü 2 değer ile belirtilen aralığı sınır. Bu değerler sayı, metin veya tarih olabilir.

Söz dizimi: SELECT <sütun adı> FROM <tablo adı>

WHERE <sütun adı> BETWEEN <değer 1> AND <değer 2>

Örnek: Alfabetik olarak müşteri kodu CACTU ve DUMON arasında yer alan tüm müşterileri görüntülemek için;

Sorgu:

```
USE northwind
SELECT *
FROM CUSTOMERS
WHERE CustomerID
BETWEEN 'CACTU' AND
'DUMON'
```

Sonuç:

	CustomerID	CompanyName	ContactName
1	CACTU	Cactus Comidas para llevar	Patricio Simpson
2	CENTC	Centro comercial Moctezuma	Francisco Chang
3	CHOPS	Chop-suey Chinese	Yang Wang
4	COMMI	Comércio Mineiro	Pedro Afonso
5	CONSH	Consolidated Holdings	Elizabeth Brown
6	DRACD	Drachenblut Delikatessen	Sven Ottlieb
7	DUMON	Du monde entier	Janine Labrune

Örnek: Sadece 1996 yılı içerisinde, ilk harfi 'V' olan müşterilerin verdiği siparişleri görmek istediğimizde;

Sorgu:

```
USE Northwind
SELECT OrderId, CustomerId, Freight
FROM Orders
WHERE OrderDate BETWEEN '01/01/1996' AND '12/31/1996'
AND CustomerID LIKE 'v%'
```

	OrderID	CustomerID	Freight
1	10248	VINET	32.3800
2	10251	VICTE	41.3400
3	10274	VINET	6.0100
4	10295	VINET	1.1500
5	10334	VICTE	8.5600
6	10367	VAFFE	13.5500
7	10399	VAFFE	27.3600

Bir Listedeki Elemanların Aranması – IN

IN sözcüğünü bir listedeki elemanlardan herhangi biriyle eşleşen satırları görüntülemek için kullanırız.

NOT IN arama kriterini ise listede olmayan değerleri aramak için kullanırız.

NOT koşul bildirimi diğer arama koşullarına göre biraz daha yavaş çalışır. Performans açısından çok sık kullanılması tavsiye edilmez.

IN ile elde ettiğimiz sonuçları OR kullanarak elde edebiliriz.

Örnek: Japonya ve İtalyadaki üreticileri görmek istediğimizde;

Sorgu:

```
USE northwind
SELECT companyname, country
FROM suppliers
WHERE country IN ('Japan', 'Italy')
GO
```

Sonuç:

	companyname	country
1	Tokyo Traders	Japan
2	Mayumi's	Japan
3	Formaggi Fortini s.r.l.	Italy
4	Pasta Buttini s.r.l.	Italy

Boş Değerlerin Görüntülenmesi- NULL Değerler

Veri girişi sırasında alana herhangi bir değer girilmezse ve alan için herhangi bir varsayılan değer atanmamışsa, alanda boş (null) değeri saklanır. Null değeri boşluk(' ') veya sıfır(0) değerinden farklıdır. Belirli bir alanına hiçbir değer girilmemiş kayıtları listelemek için **IS NULL** arama kriteri kullanılır.

Bir tablodaki hangi alanların varsayılan değerleri olacağına veya alanın NULL değere izin verip vermediğine, tablo tasarımı sırasında karar verilir.

Eğer boş olmayan satırları listelemek gerektiğinde **IS NOT NULL** arama kriterini kullanılır.

Örnek: Aşağıdaki örnek fax numarası girilmemiş firmaları listeler.

Sorgu:

```
USE northwind
SELECT companyname, fax
FROM suppliers
WHERE fax IS NULL
GO
```

Sonuç:

	companyname	fax
1	Exotic Liquids	NULL
2	New Orleans Cajun Delights	NULL
3	Tokyo Traders	NULL
4	Cooperativa de Quesos 'Las ...	NULL
5	Mayumi's	NULL
6	Specialty Biscuits Ltd	NULL

Örnek: Aşağıdaki örnek ShipRegion bilgisi girilmiş olan kayıtları listeler.

Sorgu:

```
USE Northwind
SELECT
OrderID, CustomerID, Freight, ShipRegion
FROM Orders
WHERE ShipRegion IS NOT NULL
```

Sonuç:

	OrderID	CustomerID	Freight	ShipRegion
1	10250	HANAR	65.8300	RJ
2	10253	HANAR	58.1700	RJ
3	10256	WELLI	13.9700	SP
4	10257	HILAA	81.9100	Táchira
5	10261	QUEDE	3.0500	RJ
6	10262	RATTC	48.2900	NM
7	10268	GOOSE	66.2900	DF

SQL Select Distinct

DISTINCT kelimesi bir sütundaki benzersiz kayıtları listeler. Bir sütunda belirli bir kelime iki veya daha fazla sayıda tekrarlanıyor olabilir. Distinct anahtar kelimesi ile her tekrarlanan kelime sadece bir kez listelenir.

Söz dizimi: SELECT DISTINCT <sütun adı> FROM <tablo adı> [WHERE <koşul>]

Örnek: Hangi ülkelerdeki üreticilerle çalıştığımızı görmek isteyelim. Bu durumda tek bir ülkeden birden fazla firma ile çalışıyor olabiliriz.

Sorgu:

```
USE northwind
SELECT DISTINCT country
FROM suppliers
ORDER BY country
```

Sonuç:

	country
1	Australia
2	Brazil
3	Canada
4	Denmark
5	Finland

SQL Order By

ORDER BY yantümcesi satırları, belirtilen sütun(lar)a göre sıralamak için kullanılır.

ORDER BY ile hangi sütuna göre sıralayacağımızı ve sıralamanın artan veya azalan şekilde yapılacağını belirleriz. Sıralama yönü belirtilmez ise SQL veriyi artan şekilde sıralar.

Sıralama yapılacak alanlar SELECT ifadesinde yer almak zorunda değildir.

Söz dizimi:

SELECT <sütun ad(lar)ı> FROM <tablo adı>

ORDER BY <sütun adı><sıralama yönü>, <sütun adı><sıralama yönü>

Örnek: Aşağıdaki örnekte sonuç kümesi CategoryID alanına göre azalan, aynı kategorideki ürünler ise UnitPrice alanına göre artan şekilde sıralanmıştır.

Sorgu:

```
USE northwind
SELECT productid, productname,
categoryid, unitprice
FROM products
ORDER BY categoryid DESC, unitprice
ASC
```

Sonuç:

	productid	productname	categoryid	unitprice
1	13	Konbu	8	6.0000
2	45	Rogede sild	8	9.5000
3	41	Jack's N...	8	9.6500
4	46	Spegesild	8	12.0000
5	58	Escargot	8	13.2500

Matematiksel & Metinsel İfadeler

Tablolardaki alanları kullanarak, matematiksel veya metinsel işleçler yardımıyla oluşturduğumuz ifadeler ile sorgularımızın gücünü arttırabiliriz.

İşleç	Anlamı
+	Toplama
-	Çıkarma
*	Çarpma
/	Bölme
%	Mod

Örnek: Aşağıdaki örnekte Birim fiyat (UnitPrice) alanı Miktar(Quantity) alanı ile çarpılarak toplam fiyat hesaplanmış ve "TotalCost" adı ile yeni bir alan olarak görüntülenmiştir.

Sorgu:

```
USE northwind
SELECT OrderID, ProductID,
(UnitPrice * Quantity) as TotalCost
FROM [Order Details]
WHERE (UnitPrice * Quantity) > 10000
GO
```

Sonuç:

	Order ID	Product ID	TotalCost
1	10353	38	10540.0000
2	10417	38	10540.0000
3	10424	38	10329.2000
4	10865	38	15810.0000
5	10889	38	10540.0000
6	10981	38	15810.0000

SQL Count Fonksiyonları

SQL, veritabanındaki kayıtları sayabilmek için yerleşik COUNT fonksiyonunu kullanır.

Söz dizimi: SELECT COUNT(<sütun adı>) FROM <tablo adı>

COUNT(*)

COUNT fonksiyonu (*) ile kullanıldığında FROM ile belirtilen tablodaki toplam satır sayısını verir.

Örnek: Aşağıdaki örnekte Siparişler (Orders) tablosundaki kayıt sayısı listelenmiştir.

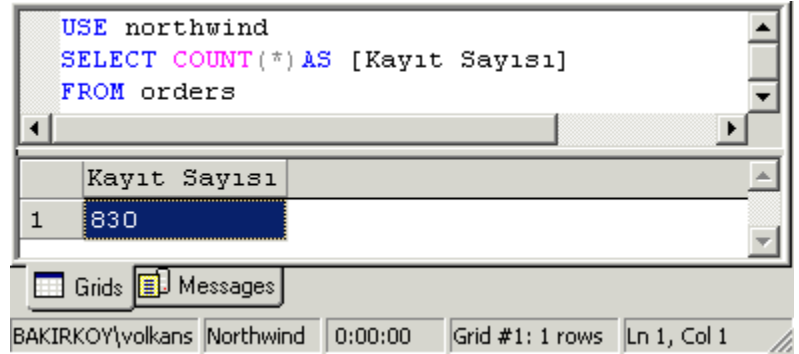
Sorgu:

USE northwind

SELECT COUNT(*) AS [Kayıt Sayısı]

FROM orders

Sonuç:



	Kayıt Sayısı
1	830

Grids Messages

BAKIRKOY\volkans Northwind 0:00:00 Grid #1: 1 rows Ln 1, Col 1

Örnek: Aşağıdaki örnek yaşı 20 den büyük olan çalışanlarınızı(Employees) listeler.

Sorgu:

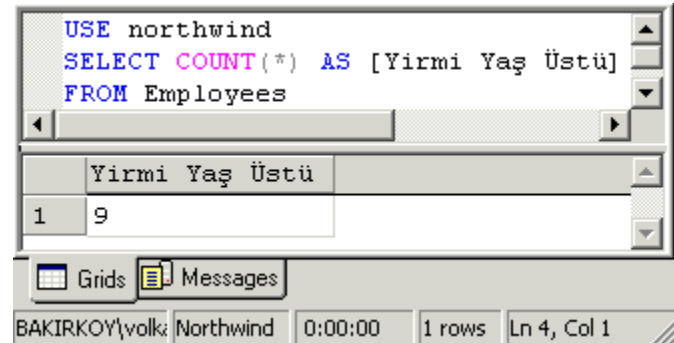
USE northwind

SELECT COUNT(*) AS [Yirmi Yaş Üstü]

FROM Employees

WHERE getdate() - BirthDate > 20

Sonuc:



	Yirmi Yaş Üstü
1	9

Grids Messages

BAKIRKOY\volk Northwind 0:00:00 1 rows Ln 4, Col 1

COUNT(<sütun adı>)

COUNT fonksiyonu bir sütun adı ile birlikte kullanıldığında, o sütundaki boş (NULL) olmayan kayıtların sayısını verir.

Örnek:

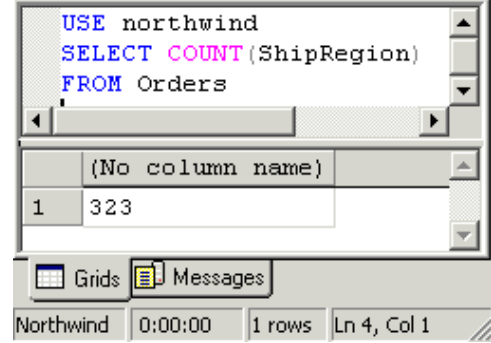
Sorgu:

USE northwind

SELECT COUNT(ShipRegion)

FROM Orders

Sonuç:



{No column name}	
1	323

COUNT DISTINCT

DISTINCT anahtar kelimesi COUNT ile birlikte kullanıldığında, belirtilen sütundaki benzersiz kayıtların sayısını verir.

Söz dizimi: SELECT COUNT(DISTINCT <sütun adı> FROM <tablo adı>

Örnek: Siparişlerin kaç farklı müşteriden alındığını öğrenmek istediğimizde,

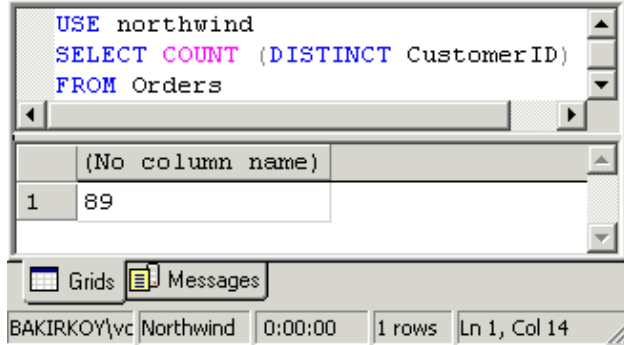
Sorgu:

USE northwind

SELECT COUNT (DISTINCT CustomerID)

FROM Orders

Sonuç:



{No column name}	
1	89

SÜTUNLARIN ve TABLOLARIN YENİDEN ADLANDIRILMASI - AS

Sonuç kümelerindeki sütun adları varsayılan olarak tablodaki alan adlarıdır. AS yardımcı kelimesini kullanarak sütunları farklı adlarla görüntüleyebiliriz.

Aynı şekilde uzun tablo isimlerini de kullanımı daha kolay olacak şekilde değiştirebiliriz.

Söz dizimi: SELECT <sütun adı> AS <yeni ad> FROM <tablo adı> AS <yeni tablo adı>

SQL TOPLAM FONKSİYONLARI (AGGREGATE FUNCTIONS)

AVG(<sütun adı>) Fonksiyonu

AVG fonksiyonu belirtilen sütundaki ortalama değeri verir. Bu hesaplama sırasında boş (NULL) değerler işleme katılmaz.

Örnek: Aşağıda Ürünler(Products) tablosundaki ortalama ürün birim fiyatı(UnitPrice) hesaplanmıştır.

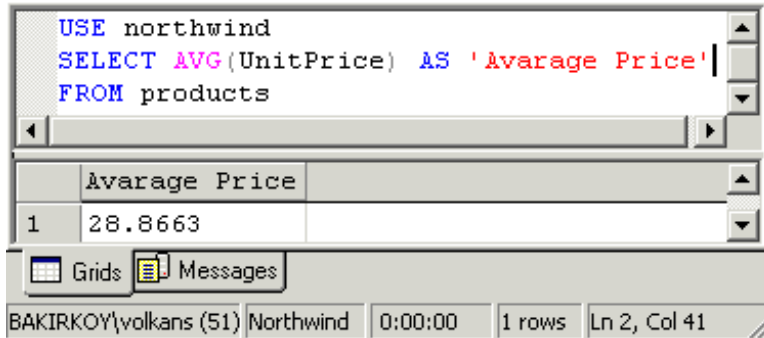
Sorgu:

USE northwind

SELECT AVG(UnitPrice) AS 'Avarage Price'

FROM products

Sonuç:



	Avarage Price
1	28.8663

Grids Messages

BAKIRKOY\volkans (51) Northwind 0:00:00 1 rows Ln 2, Col 41

MAX(<sütun adı>) Fonksiyonu

MAX fonksiyonu, bir sütundaki en yüksek değeri verir. Boş (NULL) değerler işleme katılmaz.

MAX fonksiyonu alfabetik sıralamadaki ilk metinsel ifadeyi de bulmamızı sağlar.

MIN(<sütun adı>) Fonksiyonu

MIN fonksiyonu bir sütundaki kayıtlardanki en küçük değeri verir. Boş (NULL) değerler işleme katılmaz.

MIN fonksiyonu alfabetik sıralamadaki son metinsel ifadeyi de bulmamızı sağlar.

SUM(<sütun adı>) Fonksiyonu

SUM fonksiyonu belirtilen sütundaki sayısal değerlerin bir toplamını geri döndürür.

İlk n Kaydın Görüntülenmesi - TOP n

Bazı durumlarda bir tablo veya sorgudaki tüm kayıtları görmek istemeyebiliriz. Bir sonuç kümesindeki ilk n satırı veya tüm kayıtların belirli bir yüzdesini görüntüleyebilmek için TOP n anahtar kelimesini kullanırız.

TOP n veya TOP n PERCENT anahtar kelimeleri ORDER BY yan tümcesini ile birlikte kullanılmalıdır. Aksi halde listelenen verilerde WHERE ifadesi ile belirtilen koşula uyan kayıtlar rastgele dizileceğinden, istediğimiz sonucu elde edemeyebiliriz.

WITH TIES

WITH TIES yan tümcesi, ORDER BY ile sıralanan sonuç kümesinde son kayıt ile aynı değerde olan kayıtların da listelenmesini sağlar. Bu durumda sonuç kümeniz belirttiğiniz n sayısından daha fazla olabilir.

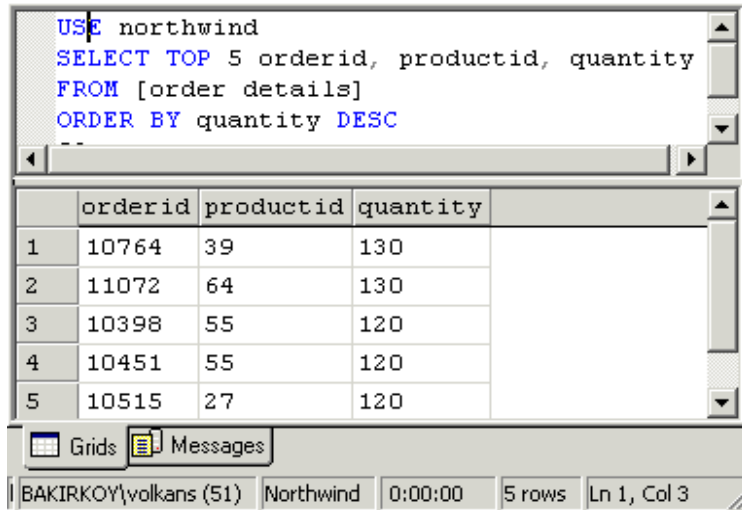
WITH TIES yan tümcesini sadece ORDER BY yan tümcesi ile kullanabilirsiniz.

Örnek: Aşağıda sipariş detayları tablosundan (order details) en yüksek siparişi verilen 5 ürün listelenmek istenmiştir.

Sorgu:

```
USE northwind
SELECT TOP 5 orderid, productid, quantity
FROM [order details]
ORDER BY quantity DESC
GO
```

Sonuç:



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

```
USE northwind
SELECT TOP 5 orderid, productid, quantity
FROM [order details]
ORDER BY quantity DESC
```

The result is displayed as a table with 5 rows and 4 columns: orderid, productid, quantity, and an empty column. The data is as follows:

	orderid	productid	quantity	
1	10764	39	130	
2	11072	64	130	
3	10398	55	120	
4	10451	55	120	
5	10515	27	120	

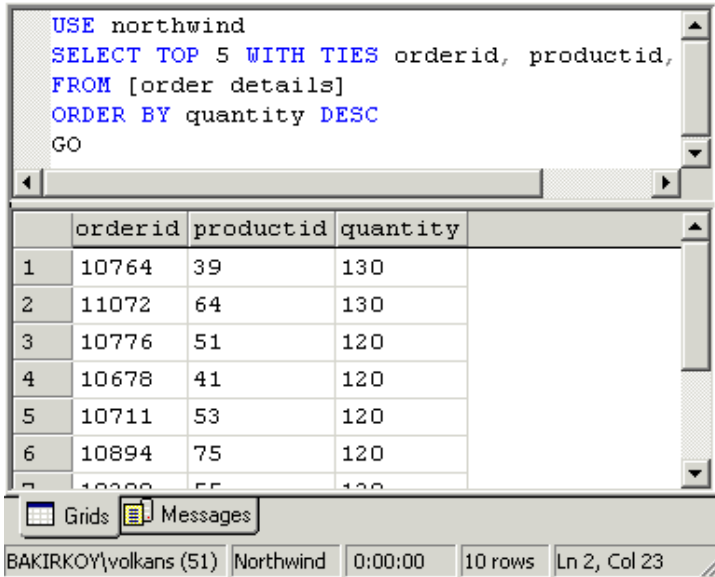
The window also shows a status bar at the bottom with the following information: BAKIRKOY\volkans (51) Northwind 0:00:00 5 rows Ln 1, Col 3

Örnek: Yukarıdaki sorguyu WITH TIES ile birlikte çalıştırırsak sonuç kümesinin bu kez 10 satırdan oluştuğunu görürüz. Bu durumda miktarı son kayıttaki değere eşit olan kayıtlarda listelenmiştir. İki sonucu karşılaştırmız.

Sorgu:

```
USE northwind
SELECT TOP 5 WITH TIES orderid, productid, quantity
FROM [order details]
ORDER BY quantity DESC
GO
```

Sonuç:



	orderid	productid	quantity
1	10764	39	130
2	11072	64	130
3	10776	51	120
4	10678	41	120
5	10711	53	120
6	10894	75	120
7	10888	55	120
8	10776	51	120
9	10678	41	120
10	10711	53	120

GROUP BY

DISTINCT anahtar kelimesi yalnızca benzersiz satırları listelemek için kullanılırken, GROUP BY yantümcesi belirtilen sütun yada sütunlardaki aynı değere sahip satırları tek bir satırda birleştirmeyi sağlar. GROUP BY genellikle toplam fonksiyonlarıyla (Aggregate Function) kullanılır. GROUP BY sorgularında kullanılan en yaygın toplama işlevleri MIN, MAX, SUM ve COUNT'dır.

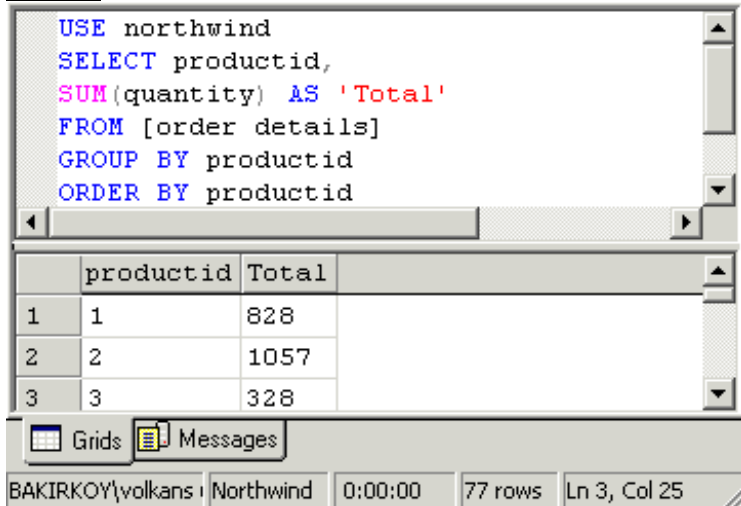
GROUP BY ile belirtilen sütunlar SELECT ifadesinde de yerelmalıdır.

Örnek: Aşağıda Sipariş Detayları(Order Details) tablosu productid alanına göre gruplandırılmış ve her grubun toplam sipariş miktarı hesaplanmıştır.

Sorgu:

```
USE northwind
SELECT productid,
SUM(quantity) AS 'Total'
FROM [order details]
GROUP BY productid
ORDER BY productid
GO
```

Sonuc:



The screenshot shows a SQL query execution window. The query is as follows:

```
USE northwind
SELECT productid,
SUM(quantity) AS 'Total'
FROM [order details]
GROUP BY productid
ORDER BY productid
```

The results are displayed in a table with the following data:

	productid	Total
1	1	828
2	2	1057
3	3	328

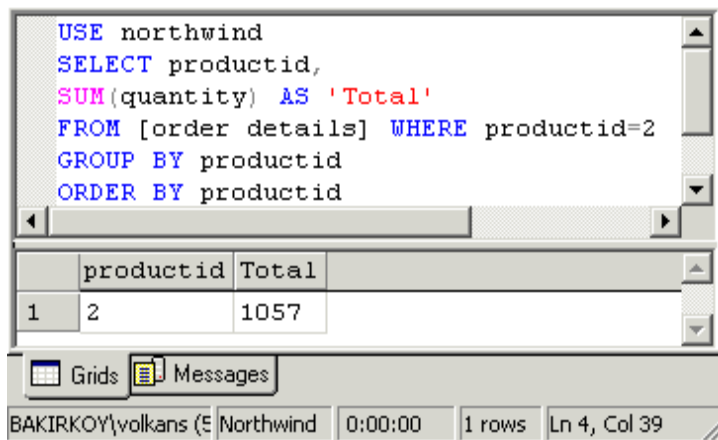
The window also shows a status bar at the bottom with the following information: BAKIRKOY\volkans | Northwind | 0:00:00 | 77 rows | Ln 3, Col 25

Örnek: Aşağıdaki örnekte sadece ürün kodu 2 olan kayıtların toplam miktarı hesaplanmıştır.

Sorgu:

```
USE northwind
SELECT productid,
SUM(quantity) AS 'Total'
FROM [order details] WHERE productid=2
GROUP BY productid
ORDER BY productid
GO
```

Sonuç:



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

```
USE northwind
SELECT productid,
SUM(quantity) AS 'Total'
FROM [order details] WHERE productid=2
GROUP BY productid
ORDER BY productid
```

Below the query, the result is displayed in a table with two columns: productid and Total. The result shows one row with productid 2 and Total 1057.

productid	Total
2	1057

At the bottom of the window, there are tabs for 'Grids' and 'Messages'. The status bar at the bottom indicates 'BAKIRKOY\volkans (S Northwind 0:00:00 1 rows Ln 4, Col 39'.

HAVING YANTÜMCESİ

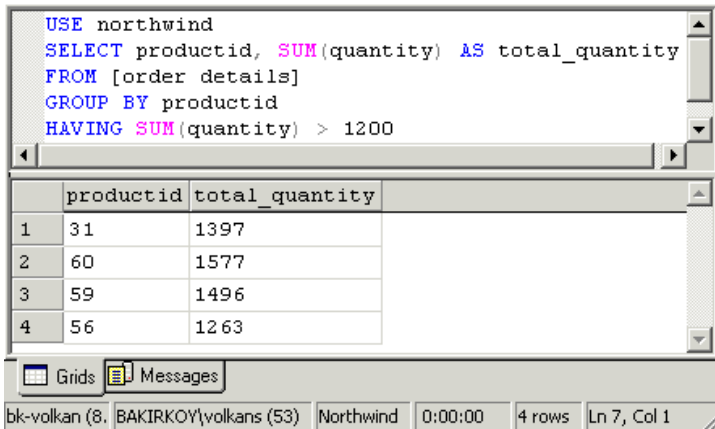
Toplam fonksiyonlarını kullanırken kısıt girilmesi gerektiğinde WHERE yantümcesini kullanamayız. HAVING yantümcesi, GROUP BY ile elde edilecek satırları kısıtlamak için kullanılır. İşlev olarak WHERE yantümcesi gibi çalışır fakat WHERE yantümcesi gruplama işlemlerinden önce, HAVING yantümcesi ise GROUP BY'dan sonra uygulanır.

Örnek: Bu sorguda toplam sipariş miktarı 1200 adetten fazla olan ürün kodları listelenmiştir.

Sorgu:

```
USE northwind
SELECT productid, SUM(quantity) AS total_quantity
FROM [order details]
GROUP BY productid
HAVING SUM(quantity) > 1200
GO
```

Sonuç:



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

```
USE northwind
SELECT productid, SUM(quantity) AS total_quantity
FROM [order details]
GROUP BY productid
HAVING SUM(quantity) > 1200
```

Below the query window, a table displays the results of the query:

	productid	total_quantity
1	31	1397
2	60	1577
3	59	1496
4	56	1263

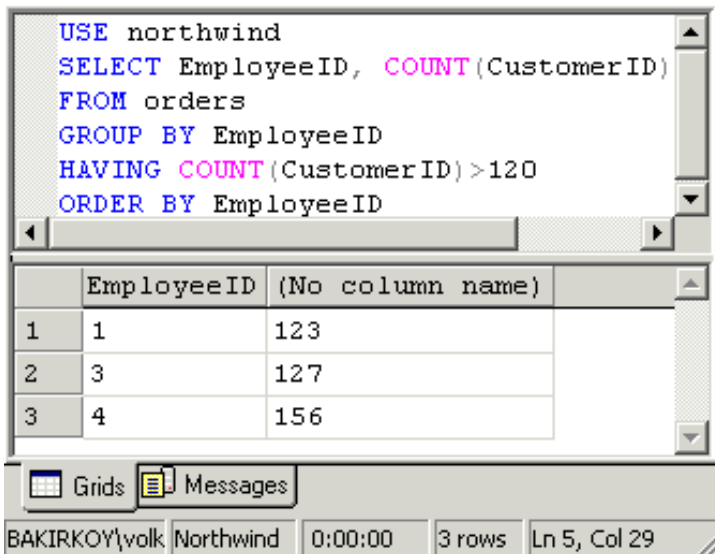
The status bar at the bottom indicates: bk-volkan (8), BAKIRKOY\volkans (53), Northwind, 0:00:00, 4 rows, Ln 7, Col 1.

Örnek: Siparişler (Orders) tablosunu EmployeeID alanına göre gruplandırılmış ve CustomerID alanı her grup için sayılmıştır. Bu şekilde her çalışanın toplam kaç sipariş aldığını görebilirim. Burada saydırılacak alan CustomerID’ den farklı bir alan da olabilirdi.

Sorgu:

```
USE northwind
SELECT EmployeeID, COUNT(CustomerID)
FROM orders
GROUP BY EmployeeID
HAVING COUNT(CustomerID)>120
ORDER BY EmployeeID
```

Sonuç:



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

```
USE northwind
SELECT EmployeeID, COUNT(CustomerID)
FROM orders
GROUP BY EmployeeID
HAVING COUNT(CustomerID)>120
ORDER BY EmployeeID
```

Below the query window, a table displays the results of the query:

	EmployeeID	(No column name)
1	1	123
2	3	127
3	4	156

The status bar at the bottom indicates: BAKIRKOY\volk, Northwind, 0:00:00, 3 rows, Ln 5, Col 29.

GROUP BY İLE ÖZET BİLGİ

GROUP BY ile bir toplama işlevi kullandığımızda her farklı değer için sadece sonuç satırı görüntülenir. Gruplandığımız değerlerin detaylarını farklı yantümceler kullanarak görüntüleyebiliriz.

ROLLUP

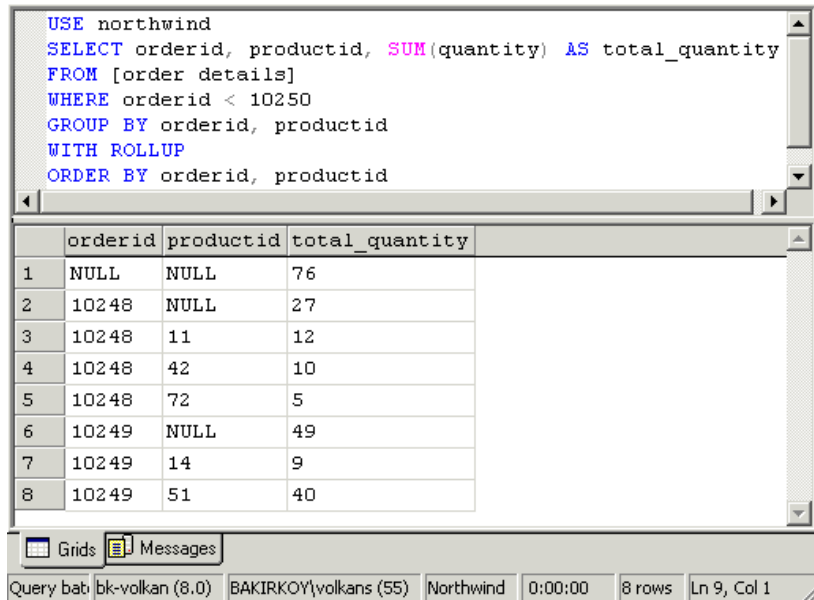
Grup değerlerini özetlemek için kullanılır.

Örnek:

Sorgu:

```
USE northwind
SELECT orderid, productid, SUM(quantity) AS total_quantity
FROM [order details]
WHERE orderid < 10250
GROUP BY orderid, productid
WITH ROLLUP
ORDER BY orderid, productid
GO
```

Sonuç:



The screenshot shows a SQL query execution window. The query is as follows:

```
USE northwind
SELECT orderid, productid, SUM(quantity) AS total_quantity
FROM [order details]
WHERE orderid < 10250
GROUP BY orderid, productid
WITH ROLLUP
ORDER BY orderid, productid
GO
```

The results are displayed in a table with 4 columns: orderid, productid, total_quantity, and an empty column. The table contains 8 rows of data, including a summary row for each orderid.

	orderid	productid	total_quantity	
1	NULL	NULL	76	
2	10248	NULL	27	
3	10248	11	12	
4	10248	42	10	
5	10248	72	5	
6	10249	NULL	49	
7	10249	14	9	
8	10249	51	40	

The bottom of the window shows a status bar with the following information: Query batı: bk-volkan (8.0) BAKIRKOY\volkans (55) Northwind 0:00:00 8 rows Ln 9, Col 1

CUBE

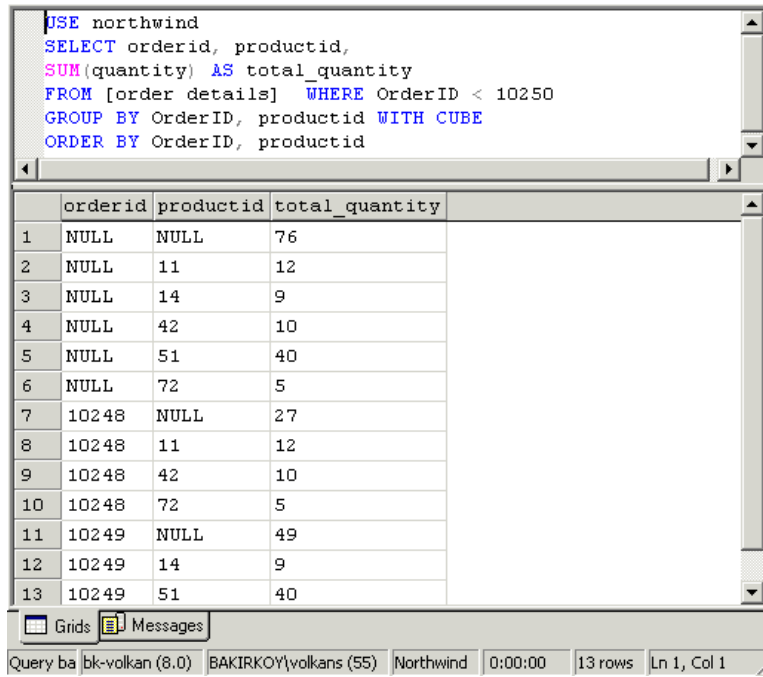
GROUP BY ile belirtilen sütunların tüm olası eşleşmelerini özetler

Örnek: Aşağıdaki örnekte sipariş kodu 10250'den küçük olan kayıtların ve ürünlerin detaylı toplamaları listelenmiştir.

Sorgu:

```
SELECT orderid, productid,  
SUM(quantity) AS total_quantity  
FROM [order details] WHERE OrderID < 10250  
GROUP BY OrderID, productid WITH CUBE  
ORDER BY OrderID, productid
```

Sonuç:



```
USE northwind  
SELECT orderid, productid,  
SUM(quantity) AS total_quantity  
FROM [order details] WHERE OrderID < 10250  
GROUP BY OrderID, productid WITH CUBE  
ORDER BY OrderID, productid
```

	orderid	productid	total_quantity
1	NULL	NULL	76
2	NULL	11	12
3	NULL	14	9
4	NULL	42	10
5	NULL	51	40
6	NULL	72	5
7	10248	NULL	27
8	10248	11	12
9	10248	42	10
10	10248	72	5
11	10249	NULL	49
12	10249	14	9
13	10249	51	40

Query ba bk-volkan (8.0) BAKIRKOY\volkans (55) Northwind 0:00:00 13 rows Ln 1, Col 1

GROUPING FONKSİYONU

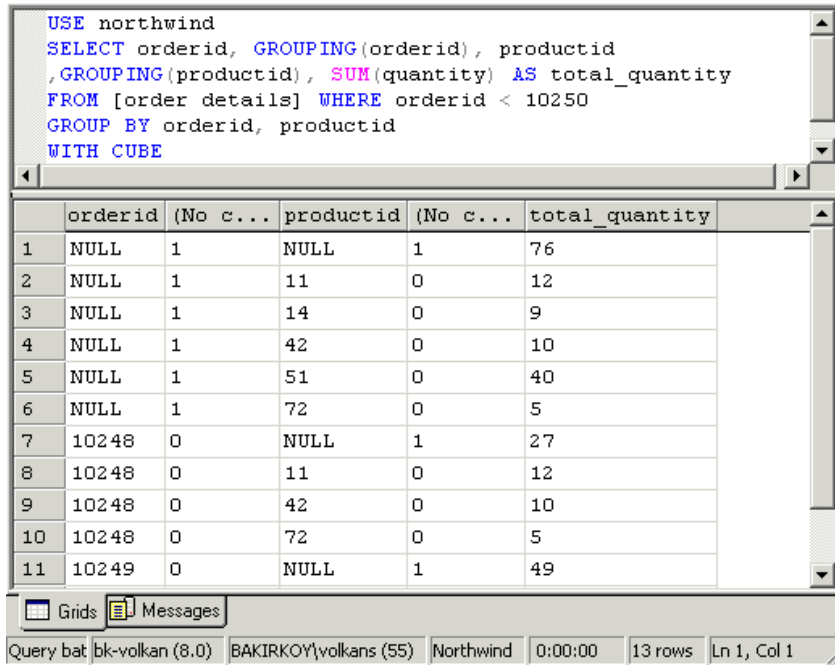
ROLL UP ve CUBE ile elde edilen sonuç kümesini görsel olarak daha kolay yorumlayabilmemizi sağlar. GROUPING ile belirtilen her sütun için yeni bir sütun oluşturulur. Bu sütunlarda özet bilgiler için 1, detay bilgiler için 0 değeri görüntülenir.

Örnek: Yukarıdaki alıştırmadan farklı olarak fazladan eklenmiş sütunlara dikkat edin. OrderID alanı NULL olarak görünen kayıtlar tüm siparişlerdeki ürünlerin toplamlarını göstermektedir.

Sorgu:

```
USE northwind
SELECT orderid, GROUPING(orderid), productid
, GROUPING(productid), SUM(quantity) AS total_quantity
FROM [order details] WHERE orderid < 10250
GROUP BY orderid, productid
WITH CUBE
ORDER BY orderid, productid
```

Sonuc:



	orderid	(No c...	productid	(No c...	total_quantity
1	NULL	1	NULL	1	76
2	NULL	1	11	0	12
3	NULL	1	14	0	9
4	NULL	1	42	0	10
5	NULL	1	51	0	40
6	NULL	1	72	0	5
7	10248	0	NULL	1	27
8	10248	0	11	0	12
9	10248	0	42	0	10
10	10248	0	72	0	5
11	10249	0	NULL	1	49

COMPUTE ve COMPUTE BY

COMPUTE ifadesi tüm sonuç kümesini listeler ve bu listenin altında tek bir detay satırı görüntüler. COMPUTE ile kullandığınız sütun SELECT listesinde de yer almalıdır.

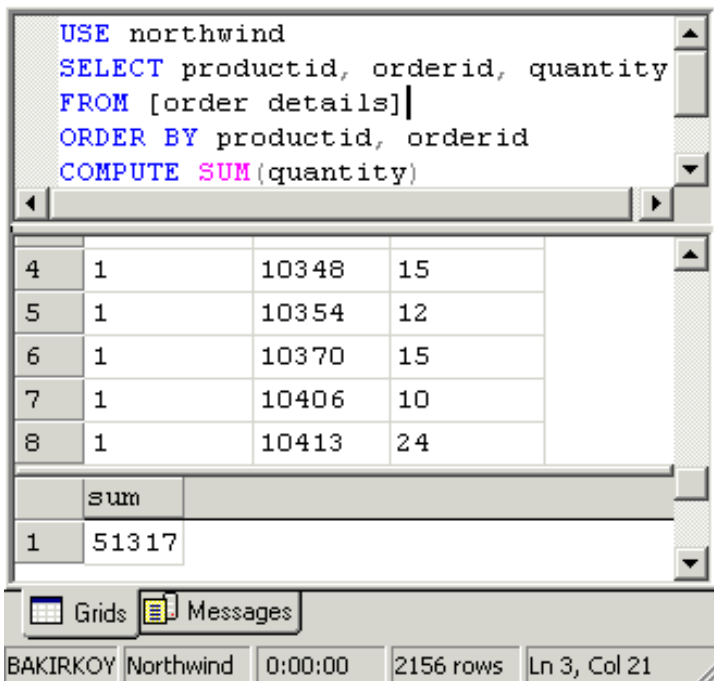
COMPUTE BY kullanarak daha detaylı gruplandırma da yapılabilir.

Örnek: Aşağıdaki sorgu Sipariş Detayları (Order Details) tablosundaki tüm satırları listeler ve kayıt kümesinin sonuna tüm ürünlerin genel toplamını verir.

Sorgu:

```
USE northwind
SELECT productid, orderid, quantity
FROM [Order Details]
ORDER BY productid, orderid
COMPUTE SUM(quantity)
GO
```

Sonuç:



4	1	10348	15	
5	1	10354	12	
6	1	10370	15	
7	1	10406	10	
8	1	10413	24	
	sum			
1		51317		

Grids Messages

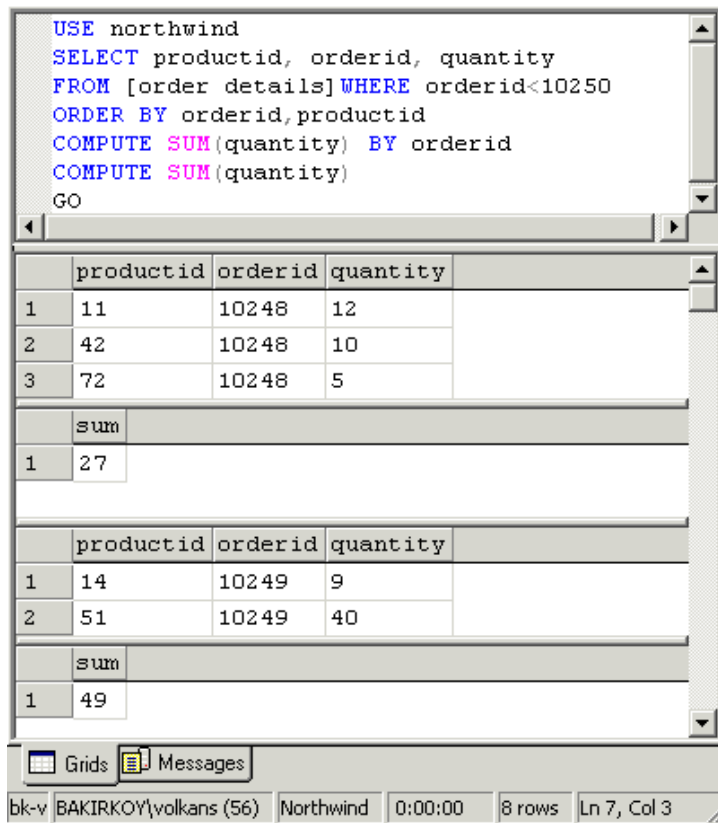
BAKIRKOY Northwind 0:00:00 2156 rows Ln 3, Col 21

Örnek: Aşağıdaki sorgu Sipariş Detayları (Order Details) tablosundaki tüm satırları listeleyecek ve her sipariş altına sipariş genel toplamını verecektir. Son olarak ise tüm siparişlerdeki genel toplam görüntülenecektir.

Sorgu:

```
USE northwind
SELECT productid, orderid, quantity
FROM [order details] WHERE orderid<10250
ORDER BY orderid,productid
COMPUTE SUM(quantity) BY orderid
COMPUTE SUM(quantity)
GO
```

Sonuç:



```
USE northwind
SELECT productid, orderid, quantity
FROM [order details] WHERE orderid<10250
ORDER BY orderid,productid
COMPUTE SUM(quantity) BY orderid
COMPUTE SUM(quantity)
GO
```

	productid	orderid	quantity	
1	11	10248	12	
2	42	10248	10	
3	72	10248	5	
	sum			
1	27			
	productid	orderid	quantity	
1	14	10249	9	
2	51	10249	40	
	sum			
1	49			

Grids Messages

bk-v BAKIRKÖY\volkans (56) Northwind 0:00:00 8 rows Ln 7, Col 3

SQL Join

Şimdiye kadar aynı anda sadece tek bir tablodan veriler ile çalıştık. Bazı durumlarda sonuç kümemizin daha anlamlı olabilmesi için iki farklı tablodan verilere ihtiyacımız olabilir. Bunun için sorgumuzda iki anahtar kelime daha kullanırız;

JOIN ifadesi ile hangi tabloları ve nasıl birleştireceğimizi

ON ifadesi ile tabloların hangi alanlar üzerinden birleşeceğini belirtiriz.

Tablolar genelde Birincil Anahtar ve Yabancı Anahtar alanları üzerinden birleştirilselerde, gerektiğinde diğer herhangi bir alan da bunun için kullanılabilir. Fakat bu alanların aynı tür veri içerdiğinden emin olmalısınız.

Birincil Anahtar(Primary key), her kayıt için benzersiz bir değer taşıyan alandır. Örneğin, öğrenci kayıtlarının tutulduğu bir tabloda, öğrenci numarasının saklandığı alan birincil anahtar olarak seçilebilir. Zira öğrenciyle ilgili tüm bilgiler gerçekte onun numarası ile kodlanmıştır ve her öğrencinin numarası bir birinden farklıdır.

INNER JOIN

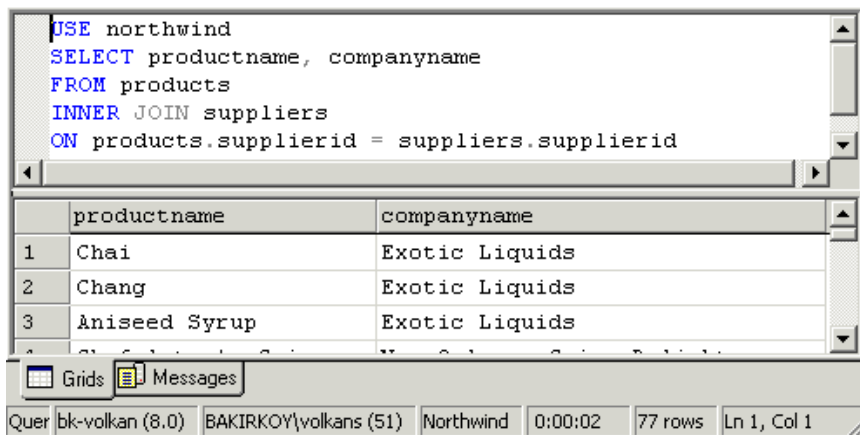
SQL de varsayılan bağlantı türüdür. İlişkili tablolarda sadece JOIN koşulunu sağlayan kayıtlar listelenir.

Örnek:

Sorgu:

```
USE northwind
SELECT productname, companyname
FROM products
INNER JOIN suppliers
ON products.supplierid = suppliers.supplierid
GO
```

Sonuc:



	productname	companyname
1	Chai	Exotic Liquids
2	Chang	Exotic Liquids
3	Aniseed Syrup	Exotic Liquids

Grids Messages

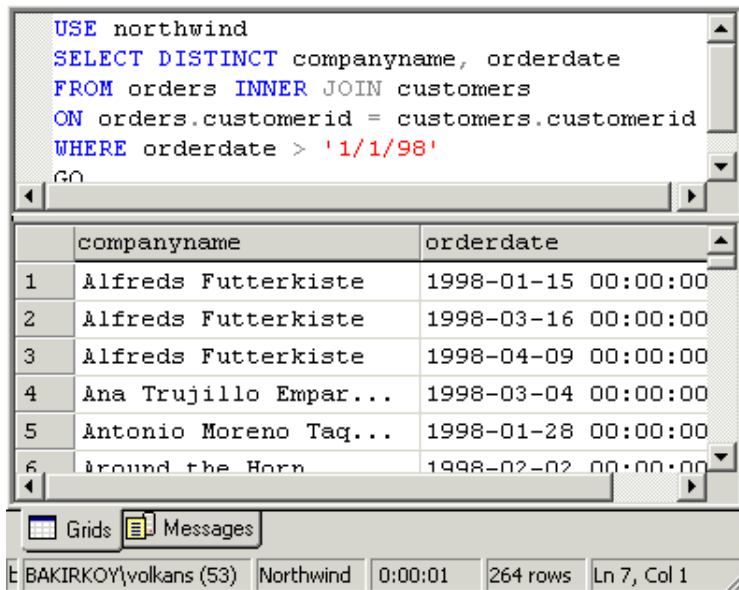
Quer bk-volkan (8.0) BAKIRKOY\volkans (51) Northwind 0:00:02 77 rows Ln 1, Col 1

Örnek: Aşağıdaki örnekte 01.01.1998 tarihinden sonra sipariş vermiş olan müşterilerin isimleri listelenmiştir. Bu iki bilgi farklı tablolarda olduğundan iki tablo arasında, her iki tabloda da olan CustomerID alanı üzerinden bir bağlantı sağlanmıştır.

Sorgu:

```
USE northwind
SELECT DISTINCT companyname, orderdate
FROM orders INNER JOIN customers
ON orders.customerid = customers.customerid
WHERE orderdate > '1/1/98'
GO
```

Sonuç:



	companyname	orderdate
1	Alfreds Futterkiste	1998-01-15 00:00:00
2	Alfreds Futterkiste	1998-03-16 00:00:00
3	Alfreds Futterkiste	1998-04-09 00:00:00
4	Ana Trujillo Empar...	1998-03-04 00:00:00
5	Antonio Moreno Taq...	1998-01-28 00:00:00
6	Around the Horn	1998-02-02 00:00:00

OUTER JOIN

LEFT veya RIGHT OUTER JOIN ifadeleri ilişkili alanda birbiriyle eşleşenlerin yanında eşleşmeyen kayıtları da listeler. JOIN koşuluna uymayan satılar NULL(boş) değer olarak görüntülenirler.

LEFT JOIN

LEFT OUTER JOIN ile yazımdaki ilk tablonun tüm satırları listelenir ve diğer tablo ile eşleşmeyen alanlar NULL(boş) değer ile gösterilir. Eğer tabloların sorgudaki sırası değişirse aynı sonucu elde edebilmek için RIGHT OUTER JOIN kullanılır.

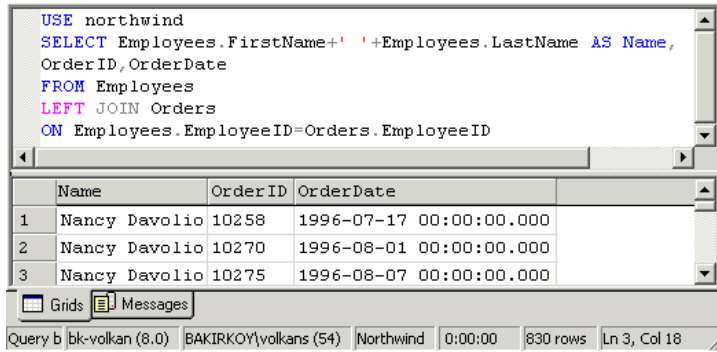
Söz dizimi: SELECT <sütun_ad(lar)> FROM <ilk_tablo>
LEFT JOIN <ikinci_tablo>
ON <ilk_tablo>.<ahantar_alan> = <ikinci_tablo>.<anahtar_alan>

Örnek: Aşağıdaki sorgunun SELECT kısmında iki sütun tekbir sütun adı altında birleştirilmiştir. Çalıştırıldığında çalışanın tam adı ve müşterilerden aldığı siparişler ve tarihleri listelenecektir.

Sorgu:

```
USE northwind
SELECT Employees.FirstName+' '+Employees.LastName AS Name,
OrderID, OrderDate
FROM Employees
LEFT JOIN Orders
ON Employees.EmployeeID=Orders.EmployeeID
```

Sonuç:



	Name	OrderID	OrderDate
1	Nancy Davolio	10258	1996-07-17 00:00:00.000
2	Nancy Davolio	10270	1996-08-01 00:00:00.000
3	Nancy Davolio	10275	1996-08-07 00:00:00.000

RIGHT JOIN

İkinci tablodaki tüm kayıtları listelemek istendiğinde RIGHT JOIN kullanılır. Bu durumda ise birinci tablodaki eşleşmeyen kayıtlar NULL(boş) olarak görüntülenir. Eğer tabloların yazım önceliğini değiştirirseniz, sorgu LEFT OUTER JOIN ile aynı sonucu verecektir.

Söz dizimi:

```
SELECT <sütun ad(lar)>
FROM <ilk_tablo>
RIGHT JOIN <ikinci_tablo>
ON <birinci_tablo>.<anahtar_alan> = <ikinci_tablo>.<anahtar_alan>
```

Örnek: Aşağıdaki örnekte tüm müşteriler ve verdikleri siparişlerin tarihleri listelenmiştir. Hiç siparişi olmayan müşterilerin tarih alanı NULL(boş) olarak görüntülendiğine dikkat edin.

Sorgu:

```
USE northwind
SELECT companyname, customers.customerid, orderdate
FROM customers
LEFT OUTER JOIN orders
ON customers.customerid = orders.customerid
GO
```

Sonuç:

```
USE northwind
SELECT companyname, customers.customerid, orderdate
FROM customers
LEFT OUTER JOIN orders
ON customers.customerid = orders.customerid
```

	companyname	customerid	orderdate
189	Familia Arquibaldo	FAMIA	1997-01-14 00:0
190	Familia Arquibaldo	FAMIA	1996-12-18 00:0
191	FISSA Fabrica Inte...	FISSA	NULL
192	Folies gourmandes	FOLIG	1997-01-08 00:0
193	Folies gourmandes	FOLIG	1997-03-20 00:0
194	Folies gourmandes	FOLIG	1997-08-15 00:0

Grids Messages

bk-volkan (8, BAKIRKOY\volkans (51) Northwind 0:00:00 842 rows Ln 7, Col 1

CROSS JOIN

İlişkili tablolar arasında olası tüm eşleşmeleri listeler. Belirli ortak bir alan belirtilmesine gerek yoktur.

Örnek: Aşağıdaki örnek üreticilerin çalışabilecekleri tüm olası gemi şirketlerini görmemizi sağlar. Shippers tablosu 3 kayıt ve Suppliers(Üreticiler) tablosunda 29 kayıt olduğundan, sonuç kümesinde 87 satır görüntülenecektir.

Sorgu:

```
USE northwind
SELECT suppliers.companyname, shippers.companyname
FROM suppliers
CROSS JOIN shippers
GO
```

Sonuç:

```
USE northwind
SELECT suppliers.companyname, shippers.companyname
FROM suppliers
CROSS JOIN shippers
GO
```

	companyname	companyname
1	Aux joyeux ecclésiastiques	Speedy Express
2	Bigfoot Breweries	Speedy Express
3	Cooperativa de Quesos 'Las ...	Speedy Express
4	Escargots Nouveaux	Speedy Express

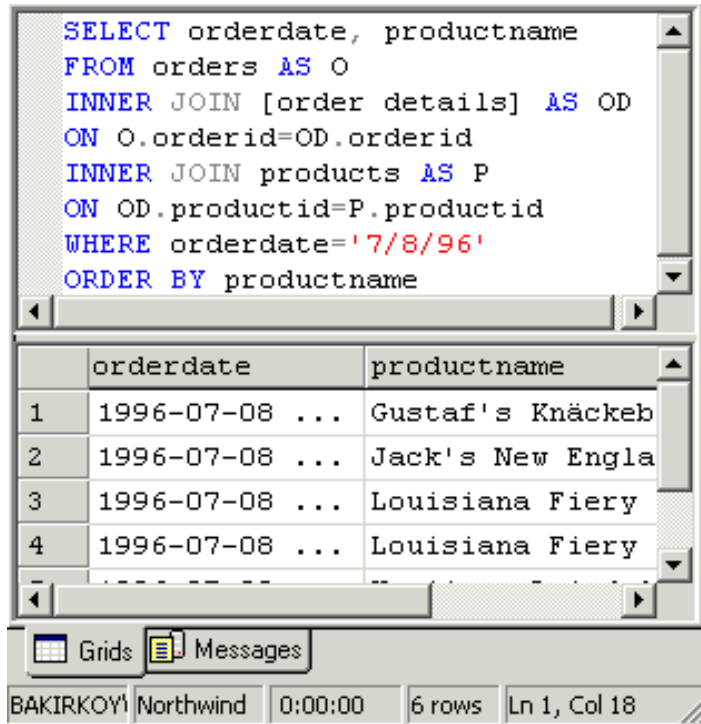
Grids Messages

bk-volkan (8, BAKIRKOY\volkans (53) Northwind 0:00:00 87 rows Ln 6, Col 1

İkiden Fazla Tablonun Birleştirilmesi

Gerektiğinde ikiden fazla tabloyu birleştirmek te mümkündür. Bir JOIN işlemiyle birleştirilen tablolar bir başka tabloya aralarındaki ortak bir sütun üzerinden bağlanabilir.

Örnek: Aşağıdaki örnekte Sipariş Detayları(Order Details) tablosunu ara tablo şeklinde kullanarak, Siparişler (Orders) ve Ürünler(Products) tablosundan gerekli bilgileri görüntülenmiştir. Bir sipariş birden fazla ürünü içerebilir ve bir ürün birden fazla siparişte yer alabilir. Sonuç kümemizde Sipariş Detayları (Order Details) tablosundan hiçbir alan görüntülemesekte sorguda kullanmalıyız.



```
SELECT orderdate, productname
FROM orders AS O
INNER JOIN [order details] AS OD
ON O.orderid=OD.orderid
INNER JOIN products AS P
ON OD.productid=P.productid
WHERE orderdate='7/8/96'
ORDER BY productname
```

	orderdate	productname
1	1996-07-08 ...	Gustaf's Knäckebröd
2	1996-07-08 ...	Jack's New England Soda
3	1996-07-08 ...	Louisiana Fiery Hot Sauce
4	1996-07-08 ...	Louisiana Fiery Hot Sauce

Grids Messages

BAKIRKOY Northwind 0:00:00 6 rows Ln 1, Col 18

SELF-JOINS

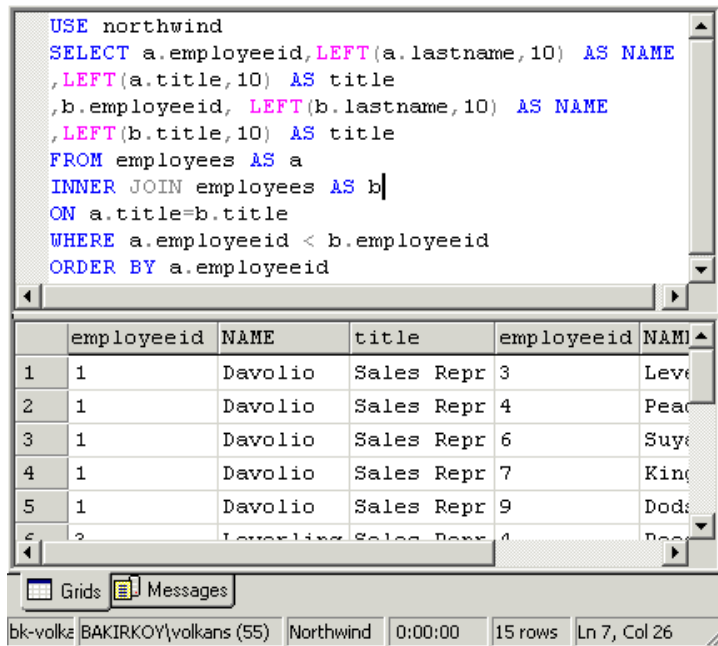
Bir tablodaki aynı değerleri içeren farklı satırları listelemek istediğimizde o tabloyu kendisi ile birleştiririz.

Örnek: Aşağıdaki örnek görevi(job title) aynı olan çalışanları listeler. WHERE ifadesi ile belirtilen koşul kendisiyle eşleşen kayıtların görüntülenmesi engeller.

Sorgu:

```
USE northwind
SELECT a.employeeid,LEFT(a.lastname,10) AS NAME
,LEFT(a.title,10) AS title
,b.employeeid, LEFT(b.lastname,10) AS NAME
,LEFT(b.title,10) AS title
FROM employees AS a
INNER JOIN employees AS b
ON a.title=b.title
WHERE a.employeeid < b.employeeid
ORDER BY a.employeeid
```

Sonuc:



```
USE northwind
SELECT a.employeeid,LEFT(a.lastname,10) AS NAME
,LEFT(a.title,10) AS title
,b.employeeid, LEFT(b.lastname,10) AS NAME
,LEFT(b.title,10) AS title
FROM employees AS a
INNER JOIN employees AS b
ON a.title=b.title
WHERE a.employeeid < b.employeeid
ORDER BY a.employeeid
```

	employeeid	NAME	title	employeeid	NAME
1	1	Davolio	Sales Repr	3	Leve
2	1	Davolio	Sales Repr	4	Peac
3	1	Davolio	Sales Repr	6	Suye
4	1	Davolio	Sales Repr	7	King
5	1	Davolio	Sales Repr	9	Dods
6	2	Fierling	Sales Repr	4	Peac

Grids Messages

bk-volkz BAKIRKOY\volkans (55) Northwind 0:00:00 15 rows Ln 7, Col 26

UNION

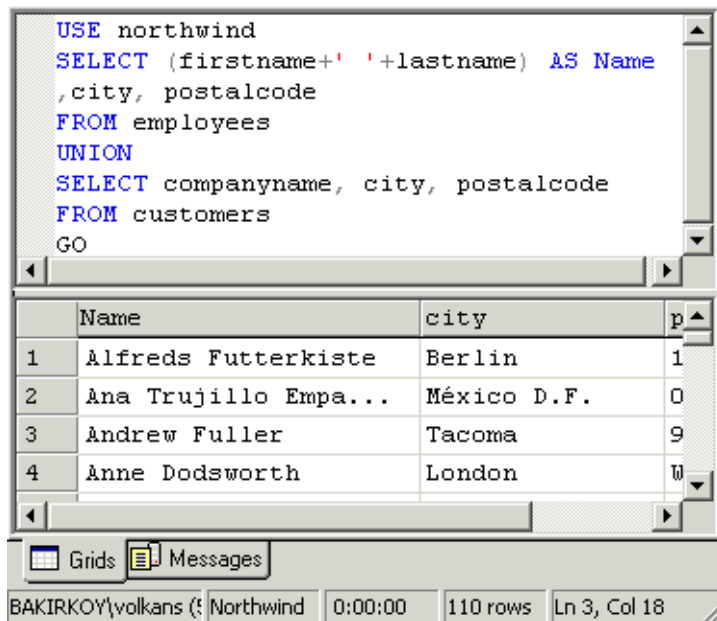
UNION işlevi iki veya daha fazla SELECT sorgusunun sonuçlarını tek bir sonuç kümesinde birleştirir.
Görüntülenecek olan tablolar aynı veritüründe aynı sayıda ve aynı düzendeki sütunlardan oluşmalıdır.

Örnek: Aşağıdaki örnekte Müşteriler(Customers) tablosu ve Çalışanlar (Employees) tablosu verileri birleştirilmiştir. Birinci sorgunun SELECT ifadesindeki takma adın görüntülendiğine dikkat edin.

Sorgu:

```
USE northwind
SELECT (firstname+' '+lastname) AS Name
,city, postalcode
FROM employees
UNION
SELECT companyname, city, postalcode
FROM customers
GO
```

Sonuç:



```
USE northwind
SELECT (firstname+' '+lastname) AS Name
,city, postalcode
FROM employees
UNION
SELECT companyname, city, postalcode
FROM customers
GO
```

	Name	city	p
1	Alfreds Futterkiste	Berlin	1
2	Ana Trujillo Empa...	México D.F.	0
3	Andrew Fuller	Tacoma	9
4	Anne Dodsworth	London	0
5	Banana Bunches O...	London	0
6	Bear Footwear	London	0
7	Berglunds Ska	Berlin	1
8	Bicycle Repair S...	London	0
9	Bicycle Repair S...	London	0
10	Bicycle Repair S...	London	0
11	Bicycle Repair S...	London	0
12	Bicycle Repair S...	London	0
13	Bicycle Repair S...	London	0
14	Bicycle Repair S...	London	0
15	Bicycle Repair S...	London	0
16	Bicycle Repair S...	London	0
17	Bicycle Repair S...	London	0
18	Bicycle Repair S...	London	0
19	Bicycle Repair S...	London	0
20	Bicycle Repair S...	London	0
21	Bicycle Repair S...	London	0
22	Bicycle Repair S...	London	0
23	Bicycle Repair S...	London	0
24	Bicycle Repair S...	London	0
25	Bicycle Repair S...	London	0
26	Bicycle Repair S...	London	0
27	Bicycle Repair S...	London	0
28	Bicycle Repair S...	London	0
29	Bicycle Repair S...	London	0
30	Bicycle Repair S...	London	0
31	Bicycle Repair S...	London	0
32	Bicycle Repair S...	London	0
33	Bicycle Repair S...	London	0
34	Bicycle Repair S...	London	0
35	Bicycle Repair S...	London	0
36	Bicycle Repair S...	London	0
37	Bicycle Repair S...	London	0
38	Bicycle Repair S...	London	0
39	Bicycle Repair S...	London	0
40	Bicycle Repair S...	London	0
41	Bicycle Repair S...	London	0
42	Bicycle Repair S...	London	0
43	Bicycle Repair S...	London	0
44	Bicycle Repair S...	London	0
45	Bicycle Repair S...	London	0
46	Bicycle Repair S...	London	0
47	Bicycle Repair S...	London	0
48	Bicycle Repair S...	London	0
49	Bicycle Repair S...	London	0
50	Bicycle Repair S...	London	0
51	Bicycle Repair S...	London	0
52	Bicycle Repair S...	London	0
53	Bicycle Repair S...	London	0
54	Bicycle Repair S...	London	0
55	Bicycle Repair S...	London	0
56	Bicycle Repair S...	London	0
57	Bicycle Repair S...	London	0
58	Bicycle Repair S...	London	0
59	Bicycle Repair S...	London	0
60	Bicycle Repair S...	London	0
61	Bicycle Repair S...	London	0
62	Bicycle Repair S...	London	0
63	Bicycle Repair S...	London	0
64	Bicycle Repair S...	London	0
65	Bicycle Repair S...	London	0
66	Bicycle Repair S...	London	0
67	Bicycle Repair S...	London	0
68	Bicycle Repair S...	London	0
69	Bicycle Repair S...	London	0
70	Bicycle Repair S...	London	0
71	Bicycle Repair S...	London	0
72	Bicycle Repair S...	London	0
73	Bicycle Repair S...	London	0
74	Bicycle Repair S...	London	0
75	Bicycle Repair S...	London	0
76	Bicycle Repair S...	London	0
77	Bicycle Repair S...	London	0
78	Bicycle Repair S...	London	0
79	Bicycle Repair S...	London	0
80	Bicycle Repair S...	London	0
81	Bicycle Repair S...	London	0
82	Bicycle Repair S...	London	0
83	Bicycle Repair S...	London	0
84	Bicycle Repair S...	London	0
85	Bicycle Repair S...	London	0
86	Bicycle Repair S...	London	0
87	Bicycle Repair S...	London	0
88	Bicycle Repair S...	London	0
89	Bicycle Repair S...	London	0
90	Bicycle Repair S...	London	0
91	Bicycle Repair S...	London	0
92	Bicycle Repair S...	London	0
93	Bicycle Repair S...	London	0
94	Bicycle Repair S...	London	0
95	Bicycle Repair S...	London	0
96	Bicycle Repair S...	London	0
97	Bicycle Repair S...	London	0
98	Bicycle Repair S...	London	0
99	Bicycle Repair S...	London	0
100	Bicycle Repair S...	London	0

Grids Messages

BAKIRKOY\volkans (Northwind) 0:00:00 110 rows Ln 3, Col 18

ALT SORGULAR(SUBQUERIES)

Bir Alt Sorgu başka bir SELECT, INSERT, UPDATE veya DELETE ifadesi içerisine yuvalanmış bir SELECT ifadesidir. Çoğu durumda alt sorguları JOIN ifadelerine dönüştürebilirsiniz. Alt sorgular, karmaşık sorguları biraz daha anlaşılabilir mantıksal bölümlere ayırmak için veya bir başka sorgunun sonuç kümesine dayalı bir sorguyu çalıştırmak için kullanılır. JOIN bağlantıları Alt Sorgulardan daha hızlı işlenirler.

Sonuç Kümesinin Tablo Olarak Kullanımı:

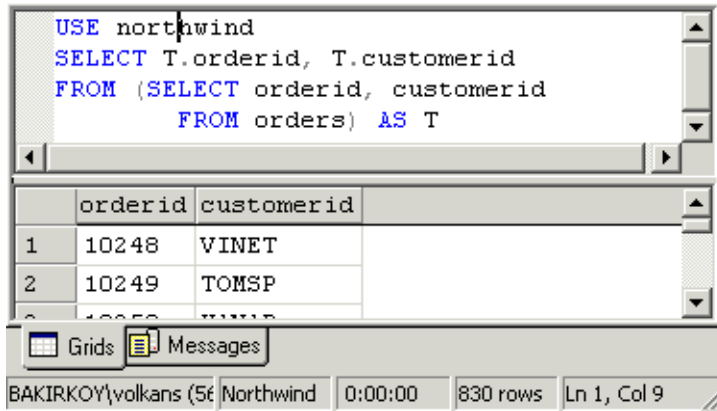
Bir sorgunun FROM ifadesinde kullanılan tablonun yerine bir başka sorguyu kullanabilirsiniz. Sonuç Kümesine herhangi bir tabloya yapabildiğimiz gibi bir geçici takma ad verilebilir.

Örnek: İç sorgu ile üretilen sonuç kümesi dış sorguda kullanılmıştır.

Sorgu:

```
USE northwind
SELECT T.orderid, T.customerid
FROM (SELECT orderid, customerid
      FROM orders) AS T
GO
```

Sonuç:



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

```
USE northwind
SELECT T.orderid, T.customerid
FROM (SELECT orderid, customerid
      FROM orders) AS T
```

Below the query, a table grid displays the results:

	orderid	customerid
1	10248	VINET
2	10249	TOMSP
3	10250	VINET

The status bar at the bottom indicates: BAKIRKOY\volkans (56) Northwind 0:00:00 830 rows Ln 1, Col 9

Sonuç Kümesinin Bir Deyim Olarak Kullanılması:

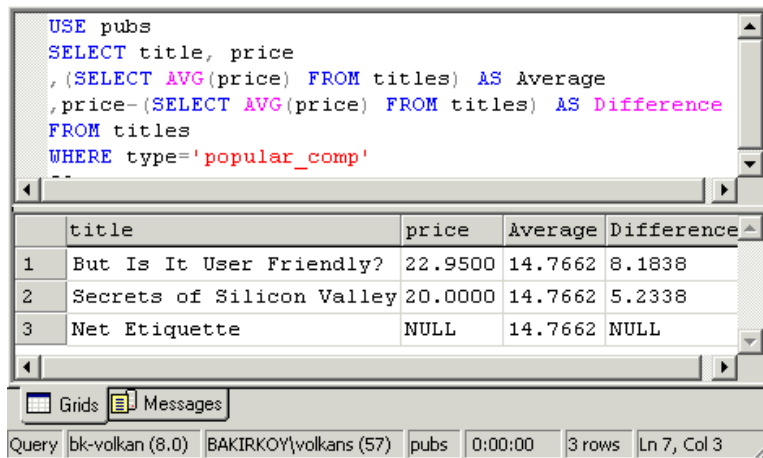
Transact-SQL de bir sorguyu herhangi bir deyimmin yerine kullanabilirsiniz. Bu amaç için kullanılacak bir sorgu tek bir sayısal değer veya tek bir sütun değer listesi döndürmelidir. Bir değerler listesi geri döndüren sorgular IN anahtar kelimesi içeren WHERE yantümcesi yerine kullanılabilir.

Örnek: Aşağıdaki sorgu çok satan bir bilgisayar kitabının fiyatını, bütün kitapların ortalama fiyatlarını ve bilgisayar kitabı ile ortalama fiyat arasındaki farkı geri döndürür.

Sorgu:

```
USE pubs
SELECT title, price
, (SELECT AVG(price) FROM titles) AS Average
, price - (SELECT AVG(price) FROM titles) AS Difference
FROM titles
WHERE type='popular_comp'
GO
```

Sonuç:



```
USE pubs
SELECT title, price
, (SELECT AVG(price) FROM titles) AS Average
, price - (SELECT AVG(price) FROM titles) AS Difference
FROM titles
WHERE type='popular_comp'
```

	title	price	Average	Difference
1	But Is It User Friendly?	22.9500	14.7662	8.1838
2	Secrets of Silicon Valley	20.0000	14.7662	5.2338
3	Net Etiquette	NULL	14.7662	NULL

Query bk-volkan (8.0) BAKIRKOY\volkans (57) pubs 0:00:00 3 rows Ln 7, Col 3

İlişkili Alt Sorgu

İç sorgu ile dış sorgu bir alan üzerinden doğrudan ilişkilidir.

Örnek: Aşağıdaki örnekte 23 nolu üründen 20 den fazla sipariş veren müşterileri listeler.

Sorgu:

```
USE northwind
SELECT orderid, customerid FROM orders AS or1
WHERE 20 < (SELECT quantity FROM [order details] AS od
            WHERE or1.orderid=od.orderid AND
                  od.productid=23)
GO
```

Sonuc:

```
USE northwind
SELECT orderid, customerid FROM orders AS or1
WHERE 20<(SELECT quantity FROM [order details] AS od
WHERE or1.orderid=od.orderid AND
od.productid=23)
```

	orderid	customerid
1	10337	FRANK
2	10348	WANDK
3	10396	FRANK
4	10402	ERNSH

Grids Messages

bk-volkan BAKIRKOY\volkans (57) Northwind 0:00:00 Grid #1: 11 rows Ln 4, Col 1

Örnek: Aşağıdaki sorgu ürünlerin bir listesini ve ürünün o ana kadarki verilmiş en yüksek sipariş miktarını döndürür.

Sorgu:

```
USE northwind
SELECT DISTINCT productid,quantity
FROM [order details] AS ord1
WHERE quantity=( SELECT MAX(quantity)FROM [order details] AS ord2
WHERE ord1.productid=ord2.productid )
GO
```

Sonuc:

```
USE northwind
SELECT DISTINCT productid,quantity
FROM [order details] AS ord1
WHERE quantity=( SELECT MAX(quantity)FROM [order details] AS ord2
WHERE ord1.productid=ord2.productid )
```

	productid	quantity
1	1	80
2	2	100
3	3	60
4	4	50

Grids Messages

Query batch complete bk-volkan (8.0) BAKIRKOY\volkans (57) Northwind 0:00:00 77 rows Ln 2, Col 35

ALT SORGU-JOIN DÖNÜŞÜMÜ

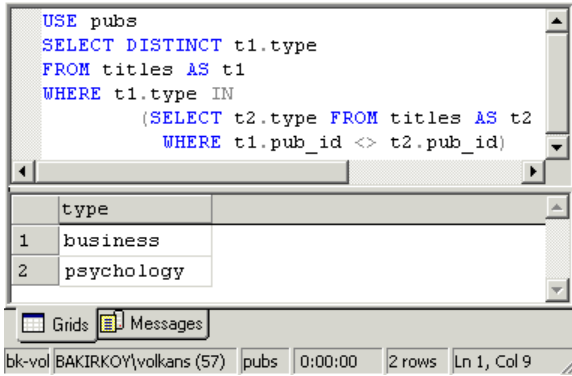
İlişkili alt sorguları bir JOIN ifadesi yerine kullanabilirsiniz.

Örnek: Aşağıdaki iki farklı sorgu aynı sonucu üretmektedir. Birinci örnekte Alt Sorgu kullanılmış, ikinci sorguda aynı sonuca JOIN ile ulaşılmıştır.

Sorgu 1:

```
USE pubs
SELECT DISTINCT t1.type
FROM titles AS t1
WHERE t1.type IN
      (SELECT t2.type FROM titles AS t2
       WHERE t1.pub_id <> t2.pub_id)
GO
```

Sonuç1:



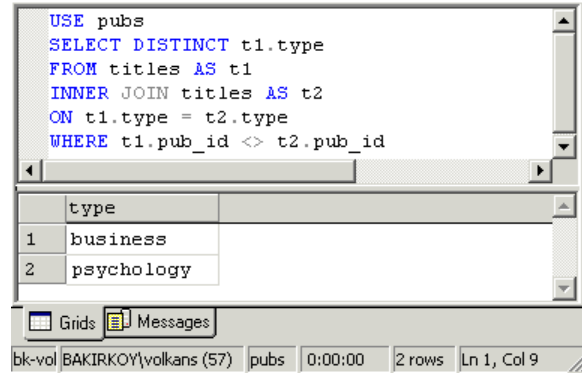
	type
1	business
2	psychology

Grids Messages
bk-vol BAKIRKOY\volkans (57) pubs 0:00:00 2 rows Ln 1, Col 9

Sorgu 2:

```
USE pubs
SELECT DISTINCT t1.type
FROM titles AS t1
INNER JOIN titles AS t2
      ON t1.type = t2.type
WHERE t1.pub_id <> t2.pub_id
GO
```

Sonuç 2:



	type
1	business
2	psychology

Grids Messages
bk-vol BAKIRKOY\volkans (57) pubs 0:00:00 2 rows Ln 1, Col 9

ALT SORGU – HAVING DÖNÜŞÜMÜ

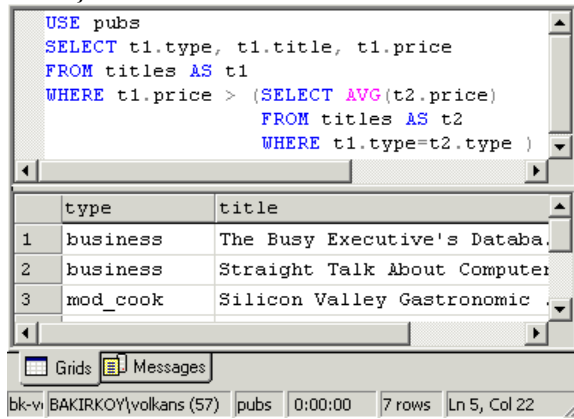
İlişkili bir alt sorgunun ürettiği sonuç kümesini HAVING yantümcesi ile de elde edebiliriz.

Örnek: Aşağıdaki iki örnekte kendi türünün ortalama fiyatından daha pahalı olan ürünleri listeler.

Sorgu 1:

```
USE pubs
SELECT t1.type, t1.title, t1.price
FROM titles AS t1
WHERE t1.price > (SELECT AVG(t2.price)
                  FROM titles AS t2
                  WHERE t1.type=t2.type )
GO
```

Sonuç 1:

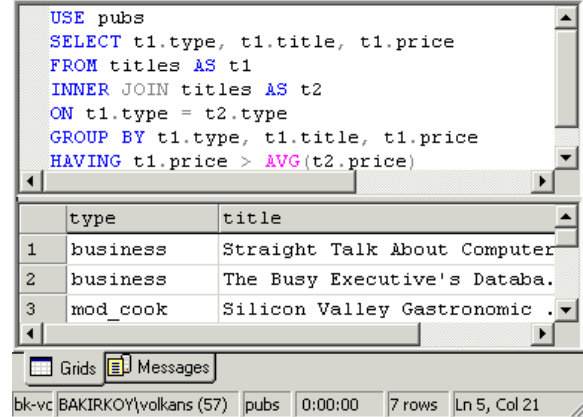


	type	title
1	business	The Busy Executive's Database
2	business	Straight Talk About Computers
3	mod_cook	Silicon Valley Gastronomic

Sorgu 2:

```
USE pubs
SELECT t1.type, t1.title, t1.price
FROM titles AS t1
INNER JOIN titles AS t2
ON t1.type = t2.type
GROUP BY t1.type, t1.title, t1.price
HAVING t1.price > AVG(t2.price)
GO
```

Sonuç 2:



	type	title
1	business	Straight Talk About Computers
2	business	The Busy Executive's Database
3	mod_cook	Silicon Valley Gastronomic

EXISTS ve NOT EXISTS İŞLEVLERİ:

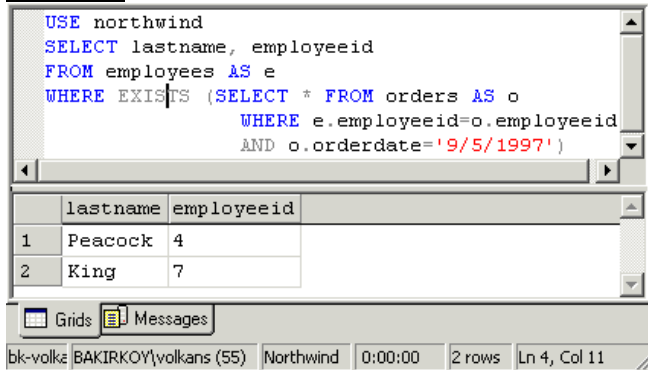
Bazı durumlarda Alt Sorguların ürettikleri değerler ile değil, sorgunun sonucunun var olup olmadığıyla ilgileniriz. EXIST ve NOT EXIST işlevleri, bu sorgulamayı yapmak için kullanılır ve DOĞRU (TRUE) veya YANLIŞ (FALSE) değeri döndürür.

Örnek: Birinci sorguda '09/05/1997' tarihinde sipariş almış olan çalışanlar EXIST işlevi yardımıyla tespit edilip, bilgileri listelenmiştir. Aynı sonuca tabloları JOIN ile birleştirerek de ulaşabiliriz. İkinci sorgudaki DISTINCT ifadesi her çalışanın tek bir kez görüntülenmesini sağlar.

Sorgu 1:

```
USE northwind
SELECT lastname, employeeid
FROM employees AS e
WHERE EXISTS
    (SELECT * FROM orders AS o
     WHERE e.employeeid=o.employeeid
     AND o.orderdate='9/5/1997')
GO
```

Sonuç 1:



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results grid. The query editor contains the following SQL code:

```
USE northwind
SELECT lastname, employeeid
FROM employees AS e
WHERE EXISTS (SELECT * FROM orders AS o
              WHERE e.employeeid=o.employeeid
              AND o.orderdate='9/5/1997')
```

The results grid displays two rows of data:

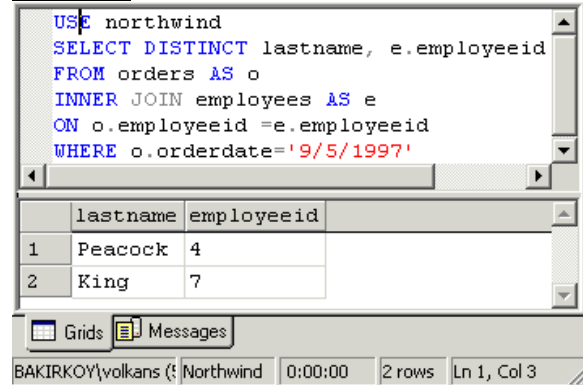
	lastname	employeeid
1	Peacock	4
2	King	7

The status bar at the bottom indicates: bk-volk: BAKIRKOY\volkans (55) Northwind 0:00:00 2 rows Ln 4, Col 11

Sorgu 2:

```
USE northwind
SELECT DISTINCT lastname, e.employeeid
FROM orders AS o
INNER JOIN employees AS e
ON o.employeeid =e.employeeid
WHERE o.orderdate='9/5/1997'
GO
```

Sonuç 2:



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results grid. The query editor contains the following SQL code:

```
USE northwind
SELECT DISTINCT lastname, e.employeeid
FROM orders AS o
INNER JOIN employees AS e
ON o.employeeid =e.employeeid
WHERE o.orderdate='9/5/1997'
```

The results grid displays two rows of data:

	lastname	employeeid
1	Peacock	4
2	King	7

The status bar at the bottom indicates: BAKIRKOY\volkans (55) Northwind 0:00:00 2 rows Ln 1, Col 3

INSERT... VALUE İŞLEMİ

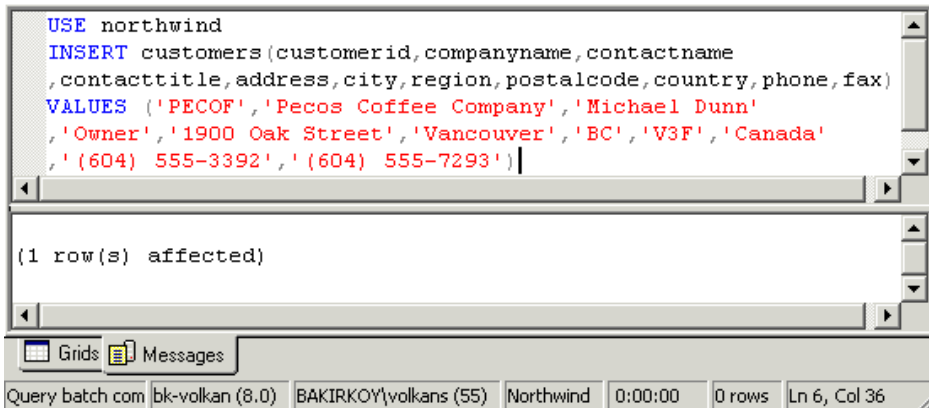
INSERT ifadesini VALUES yantümcesi ile birlikte kullanarak tablolara yeni satırlar eklenebilir. Eklenecek olan verilerde sütun sırası ve veri türü tablonunkilerle uyushmalıdır.

Örnek: Aşağıdaki sorgu 'Pecos Coffe Company' isimli firmayı müşterler tablosuna ekleyecektir.

Sorgu :

```
USE northwind
INSERT customers(customerid,companyname,contactname
,contacttitle,address,city,region,postalcode,country,phone,fax)
VALUES ('PECOF','Pecos Coffee Company','Michael Dunn'
,'Owner','1900 Oak Street','Vancouver','BC','V3F','Canada'
,'(604) 555-3392','(604) 555-7293')
GO
```

Sonuç:



```
USE northwind
INSERT customers (customerid, companyname, contactname
, contacttitle, address, city, region, postalcode, country, phone, fax)
VALUES ('PECOF', 'Pecos Coffee Company', 'Michael Dunn'
, 'Owner', '1900 Oak Street', 'Vancouver', 'BC', 'V3F', 'Canada'
, '(604) 555-3392', '(604) 555-7293')|
```

(1 row(s) affected)

Query batch com bk-volkan (8.0) BAKIRKOY\volkans (55) Northwind 0:00:00 0 rows Ln 6, Col 36

Not: Sorguyu ikinci kez çalıştırdığınızda aşağıdaki hata mesajı ile karşılaşacaksınız.

```
Server: Msg 2627, Level 14, State 1, Line 1
Violation of PRIMARY KEY constraint 'PK_Customers'. Cannot insert
The statement has been terminated.
```

Bunun nedeni Birincil Anahtarın tekrarlanmasına izin vermemesidir.

INSERT...SELECT

Bir sorgunun sonuç kümesini bir tabloya eklemek istediğimizde INSERT – SELECT ifadesini kullanırız. Bu işlem sırasında verilerin yazılacağı tablonun alanlarının boş(NULL) değere izin verip vermediğine veya alanın varsayılan (default) bir değeri olup olmadığına dikkat edilmelidir. Eğer boş (NULL) değere izin verilmiyorsa, bu alanlar için mutlaka bir değer belirtilmelidir.

Örnek: Aşağıdaki sorgu Çalışanlar (Employees) tablosundaki verileri Müşteriler (Customers) tablosuna ekleyecektir. Burada şirketin her çalışanı yine şirket için potansiyel bir müşteri olduğu düşünülmüştür.

Sorgu içerisindeki substring() fonksiyonu, FirstName ve LastName alanlarından sırasıyla 3 ve 2’şer harf alarak yeni bir alan oluşturulmuştur. Bu alan verinin ekleneceği Müşteriler (Customers) tablosunun birincil alanı CustomerID’ ye yazılacaktır

Sorgu:

```
USE northwind
INSERT customers
SELECT substring(firstname,1,2)+substring(lastname,1,3)
,lastname, firstname, title, address, city
,region, postalcode, country, homephone, NULL
FROM employees
GO
```


Sonuç:

```
USE northwind
INSERT customers
SELECT substring(firstname,1,2)+substring(lastname,1,3)
,lastname, firstname, title, address, city
,region, postalcode, country, homephone, NULL
FROM employees
```

(9 row(s) affected)

Que bk-volkan (8.0) BAKIRKOY\volkans (55) Northwind 0:00:00 0 rows Ln 3, Col 55

INSERT INTO... VALUES

INSERT INTO deyimi bir tabloya yeni kayıtlar eklemek için kullanılır. Daha önce gördüğümüz INSERT INTO deyiminden farklı olarak sadece belirli sütunlara da veri girişi yapılabilir.

Söz dizimi 1:

INSERT INTO table_name
VALUES (value1, value2,...)

Söz dizimi 2:

INSERT INTO table_name (column1, column2,...)
VALUES (value1, value2,...)

Örnek: Aşağıdaki sorgu Customers tablosuna sadece müşteri kodu (CustomerID) ve Firma adı (CompanyName) verilerinin girildiği bir kayıt ekler.

Sorgu:

```
USE northwind
INSERT INTO customers(CustomerID,CompanyName)
VALUES('VSEZ','Viol Products')
```

Sonuç:

```
USE northwind
INSERT INTO customers (CustomerID,CompanyName)
VALUES ('VSEZ', 'Viol Products')
```

(1 row(s) affected)

bk-vo BAKIRKOY\volkans (55) Northwind 0:00:00 0 rows Ln 2, Col 1

Update Deyimi

UPDATE deyimi tablodaki veriyi değiştirmek için kullanılır.

Söz dizimi:

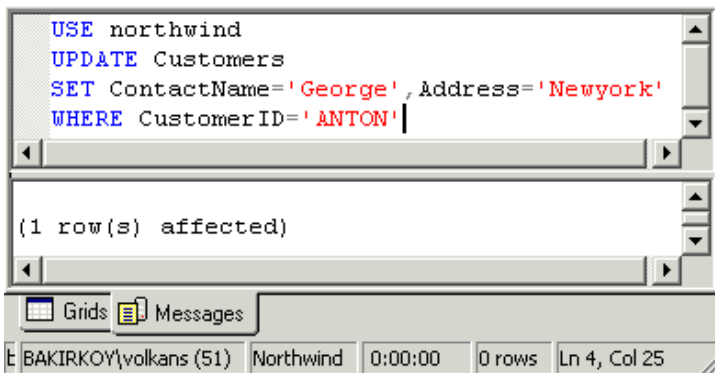
```
UPDATE <tablo adı>  
SET <sütun adı>= <yeni değer>  
WHERE <sütun adı> = <eski değer>
```

Örnek: Aşağıdaki sorgu Müşteri kodu(CustomerID) 'ANTON' olan kayıtların Görüşülecek Kişi (ContactName) ve Adres alanlarını güncelleyecektir. Koşul kısmında belirttiğimiz alan Birincil Alan olduğundan sadece tek bir satır güncellenecektir. Sorgunun sonucunu basit bir SELECT sorgusu ile kontrol edebilirsiniz.

Sorgu:

```
USE northwind  
UPDATE Customers  
SET ContactName='George',Address='Newyork'  
WHERE CustomerID='ANTON'
```

Sonuç:



Delete Deyimi

DELETE deyimi tablodaki satırları silmek için kullanılır.

Söz dizimi:

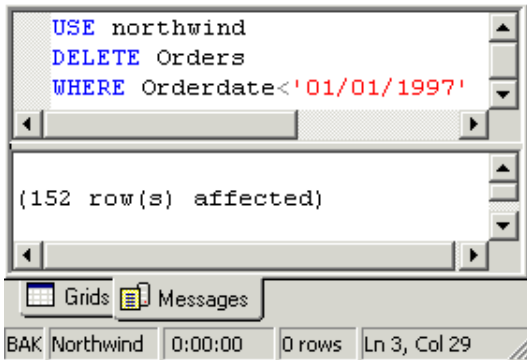
```
DELETE FROM <tablo adı>  
WHERE <sütun adı>=<aranan değer>
```

Örnek: Aşağıdaki örnekte Siparişler (Orders) tablosundan 1997 yılından önceki siparişler silinecektir.

Sorgu:

```
USE northwind  
DELETE Orders  
WHERE Orderdate<'01/01/1997'
```

Sonuç:



Not: Sipariş Detayları (Order Details) tablosunda ilgili kayıtlar olduğundan, sizin sorgunuzda bir hata uyarısı verecektir.

Tablodaki bütün kayıtların silinebilmesi için ;

Söz dizimi:

DELETE FROM <tablo adı>

Bölüm 2

Transact-Sql Dili

Transact-Sql bildiğimiz standart SQL dilinin geliştirilmiş halidir. Bu bölümde sizin bu önemli dili öğrenmenizi sağlayacağız.

Veri Tanımlama Dili (Data Definition Language) deyimleri

DDL deyimleri SQL server veritabanlarının oluşturulmasını ve yönetilmesini sağlar. Bu yapı temel olarak Create, Alter ve Drop deyimlerin içerir ve en çok aşağıdaki deyimler kullanılır.

CREATE DATABASE

CREATE PROCEDURE

ALTER DATABASE

DROP DATABASE

DROP PROCEDURE

Create Database

Bu deyim bir database, Data file gibi veri tabanın içereceği bütün yapıları kullanabileceğiniz bir amaçla kullanılır. Genel yapısı aşağıdaki gibidir.

Söz dizimi:

CREATE DATABASE database_adi

[ON

[< dosyanınAdı > [,...n]]

[, < filegrup > [,...n]]

]

[LOG ON { < dosyanınAdı > [,...n] }]

[COLLATE DilAyarları]

[FOR LOAD | FOR ATTACH]

< dosyanınAdı > ::=

[PRIMARY]

([NAME = DataFileninAdı ,]

FILENAME = 'DosyanınTamAdı'

[, SIZE = Boyut]

[, MAXSIZE = { EnFazlaBüyümesi | UNLIMITED }]

[, FILEGROWTH = BüyümeHızı]) [,...n]

< filegroup > ::=

FILEGROUP filegroup_Adi < DosyanınAdı > [,...n]

bu yapı gözünüzü korkutmasın aslında sadece 'Create Database İsim' deyimini yeterlidir diğerleri opsiyoneldir yani kullanımlarında sql server model veritabanından bu yetenekleri alır ve oluşturur. Aşağıdaki bölümde yukarıda gördüğünüz deyimlerin neler olduğu anlatılmaktadır.

ON — dosyaları ve bu dosyaların harddiskteki yerlerini gösterir.

N — Bu deyim bir çok kez kullanılabileceğini anlatır.

LOG ON—Transaction Log dosyanızın nerede tutulacağını anlatır.

COLLATE—allows you to define the collation for the database. Collation is a set of rules that determines how data is compared, ordered, and presented.

FOR LOAD— oluşturduğunuz databasenin "dbo use only" opsiyonunu kullanmasını sağlar

FOR ATTACH — daha önceden oluşturulmuş data file lerin gösterilmesini yeni oluşturduğunuz database ye eklenmesini sağlar

PRIMARY — databasenin primary data file ını gösterir.

FILEGROUP — yönetim açısından data file ların belli bir grupta toplanmasını sağlar.

Create database deyimini daha iyi anlamak için aşağıdaki örneklerle bir göz atın. İlk örnek çok basit bir tanımlama ile bir database yaratıyor varsayılan ayarları kullanıyor. İkinci örnekte ise bir data file ve bir transaction log dosyasının özellikleri verilerek database yaratılıyor.

Örnek: Create Database örnekDB

İkinci örneğimizde ise bazı gelişmiş parametreler kullanılıyor

Örnek: Create Database örnekDb

On

```
(Name = 'örnekDb_dat',  
    FILENAME = 'c:\data\entSales.mdf',  
    SIZE = 10,  
    MAXSIZE = 50,  
    FILEGROWTH = 5)
```

LOG ON

```
(NAME = 'örnekDb_Log',  
    FILENAME = 'c:\data\entSaleslog.ldf',  
    SIZE = 5,  
    MAXSIZE = 25,  
    FILEGROWTH = 5)
```

Bu örnekte yeni yaratılan database bir primary data file oluşturulsun ve bu dosya C de data isimli klasörün altına konulsun bu dosyanın dosya varsayılan olarak 10 Mb boyutunda olsun maksimum 50Mb olabilsin ve 5 Mb büyüsün, aynı mantıkla Transaction Log dosyasında yorumlayabilirsiniz.

Create Procedure

Bu deyim stored procedure oluşturmakta kullanılır. Yazım kuralı aşağıdaki gibidir.

Söz dizimi:

```
CREATE PROC [ EDURE ] Prosedurİsmi  
    [ { @Prametre VeriTürü }  
    ] [ ,...n ] [ OUTPUT ]  
[ WITH  
    { RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION } ]  
AS SQLDeyimleri [ ...n ]
```

Yukarıda yazılan bazı cümlelerin açıklaması

@parameter — Stored Procedure ye argüman vermekte kullanılır. 2100 taneye kadar argüman verilebilir.

OUTPUT — argümanların geri dönüş değerleri olacaksa kullanılır.

n – birden çok argüman kullanılabileceğini anlatır.

{RECOMPILE|ENCRYPTION|RECOMPILE, ENCRYPTION}—RECOMPILE Sql serverin stored procedürü cache de tutmaması çağarıldığı zaman derlemesi anlamına gelir. ENCRYPTION SQL serverin prosedür içindeki text şeklindeki dataların şifrelenmesini sağlar

Aşağıda örnek bir stored procedüre var.

Örnek:

```
CREATE PROCEDURE musteribilgisi
```

```
    @Adi varchar(30) = 'D%',
```

```
    @Soyadi varchar(18) = '%'
```

```
AS
```

```
SELECT MusteriAdi, MusteriSoyadi, MBilgiSirketi, MBilgiTelefonu
```

```
FROM Musteriler m INNER JOIN MusteriBilgileri mb
```

```
    ON m.MusteriID = mb.MusteriBilgileriID
```

```
WHERE m.MusteriAdi LIKE @Adi
```

```
    AND m.MusteriSoyadi LIKE @Soyadi
```

Bu stored procedur ün ismi MusteriBilgisi olarak tanımlanmıştır. Argüman olarak iki argüman alır. Birincisi @Adi argümanı diğeri @Soyadi Argümanı böylece ben SP imde verdiğim isme ve verdiğim soyisme sahip olan müşterilerimi bulabileceğim. Bu müşterilerinde MusteriBilgileri tablosundan diğer bilgilerini alıp bir kayıt dizisi olarak görüntüleyebileceğim.

Çalıştırılma yöntemi :

```
EXECUTE MusteriBilgisi 'Cenk', 'Çağlar'
```

Alter Database

Bu deyimi Database nız üstünde değişiklikler yapmak için kullanabilirsiniz. Örneğin Data File larınızı, Transaction Log dosyalarınızı düzenleyip silebilir yada yeni bir tane daha oluşturabilirsiniz.

Söz dizimi:

```
ALTER DATABASE database
```

```
{ ADD FILE < DosyaAdı > [ ,...n ] [ TO FILEGROUP FileGrupAdı ]
```

```
| ADD LOG FILE < DosyaAdı > [ ,...n ]
```

```
| REMOVE FILE DosyanınTamAdı _name
```

```
| ADD FILEGROUP FileGrupAdı
```

```
| REMOVE FILEGROUP FileGrupAdı
```

```
| MODIFY FILE < DosyaAdı >
```

```
| MODIFY NAME = YeniDatabaseAdı
```

```
| MODIFY FILEGROUP FileGrupAdı {FileGrupÖzellikleri | NAME = YeniFileGrupAdı }
```

```
| SET < optionAdı > [ ,...n ] [ WITH < deyim > ]
```

```
| COLLATE < collationİsmi >
```

```
}
```

Aşağıdaki örnek database in data file ını 20Mb ye ayarlıyor.

Örnek:

```
ALTER DATABASE OrnekDb
```

```
MODIFY FILE
```

(Name = ornekDb_dat
SIZE = 20MB)

ALTER PROCEDURE

ALTER PROCEDURE deyimini kullanarak stored Procedure nizi değiştirebilirsiniz. Yazım şekli aşağıda örneklenmiştir.

Söz dizimi:

```
ALTER PROC [ EDURE ] ProsedürAdı  
    [ { @parametre VeriTürü }  
      ] [ ,...n ]  
[ WITH  
    { RECOMPILE | ENCRYPTION  
      | RECOMPILE , ENCRYPTION  
    }  
]  
AS  
    SqlDeyimleri [ ...n ]
```

Aşağıda bu konu ile ilgili daha önceden oluşturduğumuz Sp nin Alter Procedure ile düzenlenmesini görüyoruz. @Adi değişkeninin değeri değiştirilmiş ve istenen sütunlar değiştirilmiştir.

Örnek:

```
ALTER PROCEDURE musteribilgisi  
    @Adi varchar(30) = 'A%',  
    @Soyadi varchar(18) = '%'  
AS  
SELECT MusteriAdi, MusteriSoyadi, MBilgiSirketi, MBilgiAlacak  
FROM Musteriler m INNER JOIN MusteriBilgileri mb  
    ON m.MusteriID = mb.MusteriBilgileriID  
WHERE m.MusteriAdi LIKE @Adi  
    AND m.MusteriSoyadi LIKE @Soyadi
```

DROP DATABASE

Drop Database deyimi bir yada birden çok Databasenin SQL Server üzerinden dilinmesi anlamına gelmektedir. Bir Database nin silinmesi demek bu Database nin sahip olduğu Data File ların ve Transaction Log dosyalarının komple Hard Diskten atılması demektir.

Söz dizimi:

```
DROP DATABASE DataBaseAdi [, . . . n]  
n – buraya birden çok Database adı yazılabilir.
```

Aşağıda DROP DATABASE kullanımına bir örnek verilmiştir. Bu örnek SQL server üzerinden OrnekDb ve Pubs Database sini kopmle silmektedir.

Örnek:

```
DROP DATABASE OrnekDb, Pubs
```

DROP PROCEDURE

DROP PROCEDURE deyimi Sql serverdan bir stored procedure yi silmek demektir.

Söz dizimi:

DROP Procedure [ProcedureAdı] [, . . . n]

Bu deyimdeki N yine birden çok Procedure adının yazılabildiğini anlatır.

Aşağıda bir Stored Procedure nin silinmesi örneklenmiştir.

Örnek:

DROP PROCEDURE MusteriBilgisi

Data Control Language Deyimleri

DCL kullanarak SQL server 2000 ortamında güvenlik işlemlerinizi Transact-SQL ile düzenleyebilirsiniz. Her hangi bir nesne oluşturduktan sonra (DataBase, Stored Procedure) bunların güvenlik altında olmasını sağlamak için bu deyimleri kullanabilirsiniz. Aşağıda kullanılan deyimleri görüyorsunuz

GRANT

REVOKE

DENY

GRANT Deyimi

Bu deyim bazı özel kullanıcılara data base objelerini yada datayı çalıştırabilmesi yada okuma yetkisi verir.

GRANT deyimini bir tablonun bazı sütunlarına aktarabilirsiniz. Örneğin müşteriBilgileri tablosundaki MusteriAdi kolonuna sadece SELECT izni verebilirsiniz. Çok basit bir yazımı aşağıda örneklenmiştir

Söz dizimi:

GRANT

{ ALL [PRIVILEGES] }

ON TabloAdi

TO RoleAdi

Aşağıdaki örnekte MusteriBilgileri tablosuna Yöneticiler Rolüne SELECT izni veriliyor

Örnek:

GRANT SELECT

ON MusteriBilgileri

TO Yöneticiler

REVOKE Deyimi

REVOKE deyimi, verilmiş olan izin yada yada kısıtlanmış olunan bir komutun çekimser olarak kullanılmasını sağlar. Eğer SELECT i REVOKE etmişseniz bir rol için o roldeki insanlar SELECT edebilir. Ancak REVOKE çekimser olarak nitelendirilir.

Yani Cenk kullanıcısı hem bir rolünde hemde iki rolünde olabilir. Eğer bir rolüne SELECT e GRANT verilmiş ve iki rolüne SELECT e REVOKE verilmişse Cenk SELECT kullanabilir. Tam tersi olarak bir rolüne SELECT e DENY iki rolüne SELECT e REVOKE verilmişse Cenk SELECT yapamaz

Söz dizimi:

REVOKE


```
{ ALL [ PRIVILEGES ] }
```

```
ON TabloAdi
```

```
TO RoleAdi
```

Aşağıdaki örnekte MusteriBilgileri tablosuna Yoneticiler grubunun SELECT izni iptal ediliyor.

Örnek:

```
REVOKE SELECT
```

```
ON MusteriBilgileri
```

```
TO Yoneticiler
```

DENY Deyimi

DENY bir role yada kullanıcıya bir işin yapılmamasını sağlamak amacıyla o rolü yada kullanıcıyı kısıtlamak için kullanılır. DENY yetkisi diğer bütün yetkileri ezer. Yani Cenk kullanıcı birinci rölde SELECT e GRANT verilmiş ancak ikinci rölde SELECT e DENY verilmiş ise Cenk kullanıcısı SELECT yapamaz. Yazım kuralı aşağıdaki gibidir.

Söz dizimi:

```
DENY
```

```
{ ALL [ PRIVILEGES ] }
```

```
ON TabloAdi
```

```
TO RoleAdi
```

Aşağıdaki örnek MusteriBilgileri tablosundaki yoneticiler rölüne SELECT iznini kaldırmak demektir.

Örnek:

```
DENY SELECT
```

```
ON MusteriBilgileri
```

```
TO Yoneticiler
```

Tanımlayıcılar (Identifiers)

Daha önceki örneklerde gördüğümüz gibi bir çok nesnenin ismini biz aslında tanımlayıcı olarak kullandık. Bu isimler SQL server 2000 tarafından bilinen isimler oldu, aslında bizim bu tanımladığımız isimler temel olarak bir SQL server yapılarından birini anlatan, buna referans gösterilen bir yapıdır. Aşağıdaki örneğe bakarak Tanımlayıcıları daha rahat anlamlandırabilirsiniz.

```
Create Database Musteri
```

Ben bu deyimle Musteri isimli bir tanımlayıcı oluşturdum. Ve bu tanımlayıcının aslında bir Database tanımlamasını sağladım.

İki tip tanımlayıcı vardır. Düzenli ve sınırlandırılmış. Düzenli tanımlayıcılar SQL serverin yapısına uyarlar basittirler. Sınırlanmış tanımlayıcılar SQL serverin yapısının dışındaki tanımlayıcılardır. Bu türdeki tanımlayıcıları kullanmak için köşeli parantezler ([]) içine alınması gerekmektedir. Genellikle SQL server tarafından kullanılan bir keyword, boşluklu bir yazım yada bir rakamla başlanıldığı zaman parantezlerin kullanılması zorunludur. Aşağıda sınırlandırılmış bir tanımlayıcı örneği vardır

Create Database [Musteri Bilgileri]

Düzenli tanımlayıcılar aşağıdaki özelliklere sahip olurlar.

İlk karakterleri mutlaka bir harf, bir alt çizgi, bir @ karakteri yada # karakteri olmalıdır.

İlk karakterden sonra devam eden karakterleri yukarıdakilere ek olarak rakamlardanda oluşabilirler.

Düzenli bir tanımlayıcı bir SQL deyimi yada bir SQL server keyword ü olamaz. Örneğin SELECT

Düzenli bir tanımlayıcı boşluk (Space) karakteri yada diğer özel karakterlerden birini içeremez.

Değişkenler

Transact – SQL deki değişkenler Scriptlerde, fonksiyonlarda nesnelerle kullanılabilecek değerleri tutabilir. Değişkenler T-SQL dili için vazgeçilemez derecede önemlidir. Değişkenler bir fonksiyonun içindeki anlık bilgiyi tutabilir, parametre olarak kullanılabilir yada bir işlemin sonucunda geri dönüş değeri olarak iş görebilirler.

Değişkeni eğer parametre olarak kullanmayacaksanız, değişkeni tanımlamak için DECLARE deyimini kullanmanız gerekmektedir. Bundan sonra yazacağınız tanımlayıcının değişken olduğu anlatmanız için ilk karakterini @ yapmalısınız. Değişkenin adını @ ile başlatarak verdikten sonra değişkenin veri türünü tanımlamanız gerekmektedir. Burada SQL server 2000 ile kullanılabilecek herhangi bir veri türünü kullanabilirsiniz. Daha sonra bir değişkene değer atamak için iki yönteminiz var. Eğer değişkene varsayılan bir değer atayacaksanız SET deyimini, varsayılan bir değer atamayacaksanız veri tabanından aldığınız bir değeri atayacaksanız SELECT deyimini kullanabilirsiniz (SELECT deyimini aynı zamanda SET deyimi gibi varsayılan değer atamak içinde kullanabilirsiniz)

Set kullanımına örnek :

```
DECLARE @Degisken varchar(5)
```

```
SET @Degisken = 'Cenk'
```

```
SELECT @Degisken
```

Select kullanımına örnek :

```
DECLARE @Degisken varchar(10)
```

```
DECLARE @empId int
```

```
SELECT @empId = 5
```

```
SELECT @Degisken = FirstName FROM employees
```

```
WHERE employeeId = @empId
```

```
SELECT @Degisken
```

Fonksiyonlar

Fonksiyonlar işlerinizi kolaylaştırmak için sizin adınıza rutin yaptığınız işlerin bir kerede yapılmasını sağlayan yapılardır. SQL server 2000 ile birlikte gelen fonksiyonlar verileri işlemenizi ve düzenlemenizi

sağlayabilir. SQL server 2000 ile gelen fonksiyonları düzenleyemez ve değiştiremezsiniz. Ancak kendiniz bazı fonksiyonlar yazarak bunları istediğiniz gibi düzenleyip, silmeniz mümkündür.

SQL Server ile gelen fonksiyonlar

SQL server ile gelen fonksiyonları birkaç bölüme ayırabiliriz. Aggregate, scalar vs. gibi

Aggregate fonksiyonları bir kümedeki verilere sizin istediğiniz işlemi yaparak tek bir sonuç döndürür.

Örneğin daha önce kullandığımız AVG fonksiyonu bir tablodaki herhangi bir sütünün ortalamasını verir.

Aşağıdaki tabloda aggregate fonksiyonlarının bir listesini bulacaksınız.

Aggregate Fonksiyonu	Kullanımı
AVG	Seçtiğiniz gruptaki rakamların ortalamalarını verir.
COUNT	Seçtiğiniz gruptaki elemanların kaç tane olduğunu verir.
COUNT_BIG	Count gibi çalışır. Tek farkı saydığınız kolondaki değerlerin sonucunu Bigint olarak verir. (Count int olarak verir)
MAX	Seçtiğiniz grubun içindeki en büyük elemanı verir.
MIN	Seçtiğiniz grubun içindeki en küçük elemanı verir.
SUM	Seçtiğiniz gruptaki elemanların toplamını verir.
STDEV	Verdiğiniz değerlerin Standart sapmasını geri döndürür.
VAR	Verdiğiniz değerlerin Standart Varyansını geri döndürür.

Diğer fonksiyonlar

Sayısal Fonksiyon	Kullanımı
@@DATEFIRST	SET DATEFIRST ile atama yaptığınız değeri geri döndürür bu haftanın ilk günün ingiliz sisteminde kaçınıcı olduğuna göre düşünmeniz gerekir (ingilizlerle haftanın ilk günü Pazar günüdür.)
@@LANGID	Şu anda kullanmakta olduğunuz geçerli dil tanımlayıcısının değerini döndürür.
@@MAX_CONNECTIONS	Aynı anda SQL server e bağlanabilecek maksimum kullanıcı sayısını döndürür.
@@SERVERNAME	SQL Serverin kurulu olduğu makinenin adını gösterir.
@@SPID	Kullanılmakta olan kullanıcının Proses Id sini gösterir.
@@VERSION	Kullanılmakta olan SQL serverin kurulum Tarihi, saati işlemci tipi ve işletim sistemi tipini görüntüler.
DATEADD	Belli miktarda bir tarih yada saat unsurunu arttırmak için kullanılır. (Örnek : DATEADD(month, 2, getdate()))
DATEDIFF	İki tarih arasında sizin verdiğiniz kritere göre geçen tarih yada saat farkını bulur.(örnek : SELECT DATEDIFF(day, '2004/01/01', getdate()))
DATENAME	Verdiğiniz tarihin hangi elemanını istiyorsanız onu string türde döndürür, özellikle ay isimlerinde kullanılabilir (Örnek : SELECT

Sayısal Fonksiyon	Kullanımı
	DATENAME(mm, getdate()))
DATEPART	Verdiğiniz tarihin hangi elemanını istiyorsanız onu döndürür. (Örnek : SELECT DATEPART(yy, getdate()))
DAY	Verdiğiniz tarihin gün bilgisini rakamsal olarak döndürür. (Örnek : SELECT DAY(getdate()))
GETDATE	SQL serverin standart tarih formatında bilgisayarın tarih ve saat bilgisini gösterir.
GETUTCDATE	SQL serverin standart tarih formatında bilgisayarın tarih ve saat bilgisini UTC formatına göre gösterir.
MONTH	Verdiğiniz Tarihin ay bilgisini rakamsal olarak gösterir.
YEAR	Verdiğiniz Tarihin yıl bilgisini rakamsal olarak gösterir.
ABS	Verilen rakamsal bilginin mutlak değerini alır. Yani değer negatif isede pozitif e çevirip gösterir.
RAND	0 ile 1 arasında rastgele sayı üretmek için kullanılır.
ROUND	Küsrallı sayıların yuvarlatılmasını sağlar
SIGN	Pozitif sayılar için +1, 0 için 0, negatif sayılar içinde -1 verir. Matematikteki Tam Değer işlemidir.
DB_ID	Veritabanının SQL serverdaki ID sini verir
OBJECT_ID	Veritabanındaki bir objenin ID sini döndürür.
USER_ID	Veritabanına o anda bağlı olan kullanıcının ID sini verir.
ASCII	Verdiğiniz metinsel verinin en soldaki karakterinin ASCII kodunu verir.
CHAR	ASCII kodunu verdiğiniz değeri gösterir.
LEFT	Verdiğiniz metinsel verinin soldan belli sayıda karakterinin alınıp gösterilmesini sağlar.
LOWER	Büyük harflerle yazılmış metinsel verilerin küçük harflere çevrilmesini sağlar
LTRIM	Verdiğiniz metinsel verinin eğer sol tarafında boşluk karakteri varsa bunlardan temizleyerek görüntüler.
RIGHT	Verdiğiniz metinsel verinin sağdan belli sayıda karakterinin alınıp gösterilmesini sağlar.
RTRIM	Verdiğiniz metinsel verinin eğer sağ tarafında boşluk karakteri varsa bunlardan temizleyerek görüntüler.
STR	Verdiğiniz nümerik değeri metine çevirir.
SUBSTRING	Karakter, binary, resimsel ve metinsel verilerin belli bir bölümünü

Sayısal Fonksiyon	Kullanımı
	almak için kullanılır.
UPPER	Küçük harflerle yazılmış metinsel verilerin büyük harflere çevrilmesini sağlar
@ @ERROR	En son çalıştırdığınız Transact-SQL komutundan oluşan hatanın numarasını döndürür.
@ @IDENTITY	Tabloya en son eklenen IDENTITY kolonun değerini gösterir.
@ @ROWCOUNT	En son yapılan Transact-SQL işleminde kaç datanın döndüğünü gösterir.
@ @TRANCOUNT	SQL server a siz bağlı iken yaptığını aktif Transaction ları gösterir.
@ @CONNECTIONS	SQL Serverin en son başlatılmasından itibaren kaç tane connection yapılmış ise onu gösterir.

Veri Türleri

Veri türleri SQL serverin bir tablosunda tutulacak datanın hangi türden olduğunu anlatmak için kullanılır. Örneğin SQL server da bulunan bir tablonun bir kolonunu aşağıda yer alan SQL server 2000 ile gelen veri türlerinden bir tanesi mesela bigint diye tanımlarsan SQL server o kolona sadece rakam girilmesine izin verecektir. SQL Server 2000 ile gelen veri türlerinin yanında siz kendi veri türünüzde tanımlayabilirsiniz. Ancak kendi tanımlayacağınız veri türü SQL server 2000 ile gelen bir veri türünden türetilmiş olmalıdır.

Veri Türü	Değeri
bigint	-2 ⁶³ (-9223372036854775808)'dan 2 ⁶³ -1 (9223372036854775807). Arası tam sayı değerleri
binary	Sabit uzunlukta 8,000 byte'a kadar binary verileri destekler.
bit	0 veya 1 değerini alan tam sayı veri türü
char	8000 karaktere kadar sabit uzunlukta Uni Code olmayan karakter dizilerini tutar.
cursor	Bir Cursor e referans olarak kullanılır.
datetime	1 Ocak 1753'den 31 Aralık 31, 9999,'a kadar olan tarih ve saat verilerini tutar.
decimal	Sabit hassaslıkta ve -10 ³⁸ +1 ile 10 ³⁸ -1 arasında
float	-1.79E + 308'den 1.79E + 308 'e kadar.
image	Değişken uzunlukta (2,147,483,647) byte'a kadar binary verileri destekler.
int	-2 ³¹ (-2,147,483,648)'dan 2 ³¹ - 1 (2,147,483,647) Arası tam sayı değerleri
money	Çok geniş hassalıktadır. Genelde parasal büyük sayıları kullanmak içindir. -2 ⁶³ (- 922,337,203,685,477.5808) ile 2 ⁶³ - 1 (+922,337,203,685,477.5807) arasında.
nchar	4000 karaktere kadar sabit uzunlukta Unicode karakter dizilerini tutar.

Veri Türü	Değeri
ntext	2,147,483,647 karaktere kadar (Unicode) destekler.
nvarchar	4000 karaktere kadar değişken uzunlukta Unicode karakter dizilerini tutar.
real	-3.40E + 38'den 3.40E + 38. e kadar. Küsüratlı verileri tutar
smalldatetime	1 Ocak 1900'dan 6 Haziran 2079'a kadar.
smallint	2 ¹⁵ (-32,768) through 2 ¹⁵ - 1 (32,767). Arası tam sayı değerleri
smallmoney	money veri tipinin küçük versiyonudur. -214,748.3648 ile +214,748.3647 arasında.
text	2,147,483,647 karaktere kadar destekler.
timestamp	Veritabanındaki rakamsal binary türündeki verilerdir. Her zaman tek olarak otomatik şekilde üretilirler 8 Bayt lık yer kaplarlar
tinyint	0'dan 255 e kadar Tam sayı değerleri
varbinary	Değişken uzunlukta 8,000 byte'a kadar binary verileri destekler.
varchar	8000 karaktere kadar değişken uzunlukta Uni Code olmayan karakter dizilerini tutar.
uniqueidentifier	Global tekil değerlerdir. (GUID) 16 bayt lık yer kaplarlar.

Açıklama Satırları

Açıklama satırları bir Transact-SQL deyiminde bazı satırlara açıklama yazmak bazı satırların işlenmesini engellemek amacı ile kullanılır. Bu açıklama satırları ileride kodunuza baktığınızda size yada daha sonra sizin kodunuzu kullanacaklara yardımcı olması amaçlıda kullanılabilir. İyi tasarlanmış, iyi kodlanmış her programcıkta açıklama satırları kullanılmalıdır.

SQL server da iki türlü açıklama satırı ekleyebilirsiniz. Bunlardan ilki bir satırı açıklama satırı yapan yöntemdir.

Örnek:

```
SELECT * --burası örnek bir açıklama satırı
```

```
FROM Customers
```

```
--burasıda başka bir örnek açıklama satırı
```

Örnek:

```
UPDATE customers
```

```
SET areacode = '508'
```

```
WHERE areacode = '602'
```

```
/* Bu araya birden çok açıklama satırı yazılabilir
```

```
Burasıda bir başka satır. */
```

Not : slash asteriks yöntemi birden çok satırı açıklama satırı yapmak için en uygun yöntemdir.