

Veri Tabanı Yönetim Sistemleri

Giriş

SQL Server Veri Tipleri

- Decimal
- Date
- Datetime2
- Time
- Unicode ve Non-Unicode Verilerin Farkları
- Nchar
- Nvarchar
- Nchar, nvarchar ve varchar Verilerin Farkları

Decimal Veri Tipi

Kesinliği ve ölçeği sabit olan sayıları depolamak için DECIMAL veri türü kullanılır.

Yazımı **DECIMAL(p,s)** veya **DEC(p,s)** veya **NUMERIC(p,s)** şeklinde olup;

p, ondalık noktasının hem solunda hem de sağında saklanacak olan ondalık basamakların maksimum toplam sayısı olan hassasiyettir. Hassasiyet 1 ile 38 arasındadır. Varsayılan hassasiyet 38'dir.

s, ondalık noktasının sağında saklanacak olan ondalık basamak sayısı olan ölçektir. Ölçek 0 ile p (hassasiyet) arasındadır. Varsayılan olarak ölçek sıfırdır.

Decimal Veri Tipi

Örneğin;

```
CREATE TABLE test.sql_server_decimal (  
    dec_col DECIMAL (4, 2),  
    num_col NUMERIC (4, 2)  
);
```

Komutu ile bir tablo oluşturulur. Aşağıdaki komutlar ile de veri eklemesi yapılabilir.

```
INSERT INTO test.sql_server_decimal (dec_col, num_col)  
VALUES  
    (10.05, 20.05);
```

```
INSERT INTO test.sql_server_decimal (dec_col, num_col)  
VALUES  
    (99.999, 12.345);
```

Date Veri Tipi

Tarihleri depolamak için DATE veri türü kullanılır.

Yazımı **DATE** şeklinde olup;

DATE değeri,

1 Ocak 1 MS (0001-01-01) ile

31 Aralık 9999 MS (9999-12-31) arasındadır.

Yazım formatı YYYY-MM-DD şeklindedir.

Date Veri Tipi

BikeStore veri tabanı üzerinde aşağıdaki komut `order_date < '2016-01-05'` koşulu ile çalıştırılır.

```
SELECT
    order_id,
    customer_id,
    order_status,
    order_date
FROM
    sales.orders
WHERE order_date < '2016-01-05'
ORDER BY
    order_date DESC;
```

Date Veri Tipi

Komutun sonucunda aşağıdaki çıktı elde edilir.

order_id	customer_id	order_status	order_date
6	94	4	2016-01-04
7	324	4	2016-01-04
8	1204	4	2016-01-04
4	175	4	2016-01-03
5	1324	4	2016-01-03
3	523	4	2016-01-02
1	259	4	2016-01-01
2	1212	4	2016-01-01

Date Veri Tipi

Aşağıdaki ifade ile iki adet DATE sütununa sahip sales.list_prices adlı bir tablo oluşturur.

```
CREATE TABLE sales.list_prices (  
    product_id INT NOT NULL,  
    valid_from DATE NOT NULL,  
    valid_to DATE NOT NULL,  
    amount DEC (10, 2) NOT NULL,  
    PRIMARY KEY (  
        product_id,  
        valid_from,  
        valid_to  
    ),  
    FOREIGN KEY (product_id)  
        REFERENCES production.products (product_id)  
);
```


Date Veri Tipi

Aşağıdaki INSERT ifadesi ile tabloya tarih değerleri içeren bir veri eklenir.

```
INSERT INTO sales.list_prices (  
    product_id,  
    valid_from,  
    valid_to,  
    amount  
)  
VALUES  
    (  
        1,  
        '2019-01-01',  
        '2019-12-31',  
        400  
    );
```

Time Veri Tipi

Gün içerisindeki saatleri depolamak için TIME veri türü kullanılır.

Yazımı **TIME[(fractional second scale)]** şeklinde olup;

Kesirli saniye ölçeği **(fractional second scale)**, saniyelerin kesirli kısmı için basamak sayısını belirtir. Kesirli saniye ölçeği 0 ile 7 arasında değişir.

Varsayılan olarak, açıkça belirtmezseniz kesirli saniye ölçeği 7'dir.

TIME değeri 00:00:00 ile 23:59:59.9999999 arasındadır.

Yazım formatı hh:mm:ss[.nnnnnnnn] şeklindedir.

```
CREATE TABLE table_name(  
    ...,  
    start_at TIME(0),  
    ...  
);
```

Time Veri Tipi

Aşağıdaki ifade müşterilerin belirli bir mağazayı ziyaret etme sürelerini kaydeden iki TIME sütununa sahip sales.visits adlı bir tablo oluşturur.

```
CREATE TABLE sales.visits (  
    visit_id INT PRIMARY KEY IDENTITY,  
    customer_name VARCHAR (50) NOT NULL,  
    phone VARCHAR (25),  
    store_id INT NOT NULL,  
    visit_on DATE NOT NULL,  
    start_at TIME (0) NOT NULL,  
    end_at TIME (0) NOT NULL,  
    FOREIGN KEY (store_id) REFERENCES sales.stores (store_id)  
);
```

Time Veri Tipi

Aşağıdaki INSERT ifadesi sales.visits tablosuna time değerleri içeren bir veri eklenir.

```
INSERT INTO sales.visits (  
    customer_name,  
    phone,  
    store_id,  
    visit_on,  
    start_at,  
    end_at  
)  
VALUES  
(  
    'John Doe',  
    '(408)-993-3853',  
    1,  
    '2018-06-23',  
    '09:10:00',  
    '09:30:00'  
);
```

Datetime2 Veri Tipi

Hem tarihi hem de saati veritabanında saklamak için DATETIME2 veri türü kullanılır.

Yazımı **DATETIME2(fractional seconds precision)** şeklinde olup;

Kesirli saniye hassasiyeti(**fractional seconds precision**) isteğe bağlıdır. 0 ile 7 arasında değişir. Aşağıdaki komut yapısı ile tabloda oluşturulurken kullanılabilir.

```
CREATE TABLE table_name (  
    ...  
    column_name DATETIME2(3),  
    ...  
);
```

Datetime2 Veri Tipi

DATETIME2 date ve time olmak üzere iki bileşenden oluşmaktadır.

date değeri, 1 Ocak 1 MS (0001-01-01) ile

31 Aralık 9999 MS (9999-12-31) arasındadır.

time değeri 00:00:00 ile 23:59:59.9999999 arasındadır.

Yazım formatı YYYY-MM-DD hh:mm:ss şeklindedir.

Datetime2 Veri Tipi

Aşağıdaki ifade ile datetime2 türünde bir kolona sahip olan bir tablo oluşturur.

```
CREATE TABLE production.product_colors (  
    color_id INT PRIMARY KEY IDENTITY,  
    color_name VARCHAR (50) NOT NULL,  
    created_at DATETIME2  
);
```

Aşağıdaki ifadeler ile de tabloya veri eklemesi gerçekleştirilebilir.

```
INSERT INTO production.product_colors (color_name, created_at)  
VALUES  
    ('Red', GETDATE());
```

```
INSERT INTO production.product_colors (color_name, created_at)  
VALUES  
    ('Green', '2018-06-23 07:30:20');
```

Datetime2 Veri Tipi

created_at sütununun varsayılan değerini şuanki tarih ve saate ayarlamak isteniyorsa aşağıdaki ALTER TABLE ifadesi kullanılabilir.

```
ALTER TABLE production.product_colors  
ADD CONSTRAINT df_current_time  
DEFAULT CURRENT_TIMESTAMP FOR created_at;
```

Bu durum için datetime2 değerini girmeden aşağıdaki şekilde veri eklemesi yapılabilir.

```
INSERT INTO production.product_colors (color_name  
VALUES  
    ('Blue');
```


Nchar Veri Tipi

Sabit uzunluklu Unicode karakter dizilerini depolamak için NCHAR veri türü kullanılır.

Yazımı **NCHAR(n)**, **NATIONAL CHAR(n)** veya **NATIONAL CHARACTER(n)** şeklinde olup;

Buradaki n değeri dizi uzunluğunu belirtir ve 1 ile 4000 arasında değer alır.

Char veri türüne benzemekte olup Nchar sadece sabit uzunluklu karakter dizileri için kullanılabilir.

Veri değerlerinin uzunluklarının değişken olması durumunda ise varchar veya nvarchar veri türü kullanılabilir.

Nchar Veri Tipi

NCHAR ve CHAR veri türünün kıyaslaması aşağıdaki tabloda görüldüğü gibidir.

CHAR	NCHAR
Unicode olmayan karakterler depolanır.	Unicode UCS-2 formunda olan karakterler depolanır.
Her karakter için 1 byte alana ihtiyaç vardır.	Her karakter için 2 byte alana ihtiyaç vardır.
Depolama boyutu başlangıçta tanımlanan boyuta eşittir.	Depolama boyutu başlangıçta tanımlanan boyutun 2 katına eşittir.
8000 karaktere kadar depolama sağlanabilir.	4000 karaktere kadar depolama sağlanabilir.

Nchar Veri Tipi

Aşağıdaki ifade ile NCHAR türünde bir kolona sahip olan bir tablo oluşturur.

```
CREATE TABLE test.sql_server_nchar (  
    val NCHAR(1) NOT NULL  
);
```

Aşağıdaki ifadeler ile de tabloya veri eklemesi gerçekleştirilebilir.

```
INSERT INTO test.sql_server_nchar (val)  
VALUES  
    (N'あ');  
  
INSERT INTO test.sql_server_nchar (val)  
VALUES  
    (N'いえ');
```

Nchar Veri Tipi

Aşağıdaki ifade ile boyutu birden fazla olan bir karakter dizisi girilmek istendiğinde hata oluşmaktadır.

```
INSERT INTO test.sql_server_nchar (val)
VALUES
    (N'いえ');
```

Verilerin karakter sayısını ve boyutu görebilmek için aşağıdaki komut kullanılabilir.

```
SELECT
    val,
    len(val) length,
    DATALENGTH(val) data_length
FROM
    test.sql_server_nchar;
```

Çıktı:

val	length	data_length
あ	1	2

Unicode ve Non-Unicode Verilerin Farkları

Unicode ve Non-Unicode veriler metin karakterlerinin nasıl temsil edildiği ve kodlandığı açısından farklılık göstermektedir. Bu farklar aşağıda görüldüğü gibidir:

- Karakter Kodlaması
- Karakter Kümesi Aralığı
- Kodlama Biçimleri
- Uyumluluk ve Standartlaşma
- Kullanım ve Modernlik

Unicode, diller ve semboller arasında neredeyse her karakterin standart bir şekilde temsil edilmesine olanak tanırken, Non-Unicode kodlamalar genellikle sınırlı, bölgeye özel ve farklı sistemler arasında uyumsuzdur.

SQL Server VARCHAR

- SQL Server VARCHAR veri türü, değişken uzunlukta, Unicode olmayan dize verilerini depolamak için kullanılır.
- Sözdizimi aşağıda gösterilmiştir:
 - VARCHAR(n)
 - Bu sözdiziminde, n, 1 ile 8.000 arasında değişen dize uzunluğunu tanımlar. n'yi belirtmezseniz, varsayılan değeri 1'dir.
- VARCHAR sütununu bildirmenin bir başka yolu da aşağıdaki sözdizimini kullanmaktır:
 - VARCHAR(max)
 - Bu sözdiziminde, max, 2³¹-1 bayt (2 GB) olan maksimum depolama boyutunu tanımlar.
- Genel olarak, bir VARCHAR değerinin depolama boyutu, depolanan verilerin gerçek uzunluğu artı 2 bayttır.
- VARCHAR'ın ISO eşanlamlıları CHARVARYING veya CHARACTERVARYING'dir, bu nedenle bunları birbirinin yerine kullanabilirsiniz.

SQL Server VARCHAR örneği

- Aşağıdaki ifade, bir VARCHAR sütunu içeren yeni bir tablo oluşturur:

```
CREATE TABLE test.sql_server_varchar (  
    val VARCHAR NOT NULL  
);
```

- val sütununun dize uzunluğunu belirtmediğimiz için varsayılan olarak birdir.
- val sütununun dize uzunluğunu değiştirmek için ALTER TABLE ALTER COLUMN ifadesini kullanırsınız:

```
ALTER TABLE test.sql_server_varchar  
ALTER COLUMN val VARCHAR (10) NOT NULL;
```

- Aşağıdaki ifade, test.sql_server_varchar tablosunun val sütununa yeni bir dize ekler:

```
INSERT INTO test.sql_server_varchar (val)
VALUES
    ('SQL Server');
```

- İfade, dize değerinin sütun tanımında tanımlananla aynı uzunlukta olması nedeniyle beklendiği gibi çalıştı.
- Aşağıdaki ifade, uzunluğu sütunun dize uzunluğundan daha büyük olan yeni bir dize verisi eklemeyi dener:

```
INSERT INTO test.sql_server_varchar (val)
VALUES
    ('SQL Server VARCHAR');
```

- SQL Server bir hata verdi ve ifadeyi sonlandırdı:

```
String or binary data would be truncated.
The statement has been terminated.
```


- VARCHAR sütununda depolanan karakter sayısını ve değerlerin bayt sayısını bulmak için, aşağıdaki sorguda gösterildiği gibi LEN ve DATALENGTH işlevlerini kullanırsınız:

```
SELECT
    val,
    LEN(val) len,
    DATALENGTH(val) data_length
FROM
    test.sql_server_varchar;
```

val	len	data_length
SQL Server	10	10

SQL Server NVARCHAR

- SQL Server NVARCHAR veri türü, değişken uzunluktaki Unicode dize verilerini depolamak için kullanılır.
- Aşağıda NVARCHAR sözdizimi gösterilmektedir:
 - NVARCHAR(n)
 - Bu sözdiziminde, n 1 ile 4.000 arasında değişen dize uzunluğunu tanımlar. Dize uzunluğunu belirtmezseniz, varsayılan değeri 1'dir.
- NVARCHAR sütununu bildirmenin bir başka yolu da aşağıdaki sözdizimini kullanmaktır:
 - NVARCHAR(max)
 - Bu sözdiziminde, max, $2^{31}-1$ bayt (2 GB) olan bayt cinsinden maksimum depolama boyutudur.
 - Genel olarak, bir NVARCHAR değerinin bayt cinsinden gerçek depolama boyutu, girilen karakter sayısının iki katı artı 2 bayttır.
- NVARCHAR'ın ISO eş anlamlıları NATIONAL CHAR VARYING veya NATIONAL CHARACTER VARYING'dir, bu nedenle bunları değişken bildiriminde veya sütun veri tanımında birbirinin yerine kullanabilirsiniz.

VARCHAR ve NVARCHAR

	VARCHAR	NVARCHAR
Karakter Veri Türü	Değişken uzunlukta, Unicode olmayan karakterler	Değişken uzunlukta, Unicode ve Unicode olmayan karakterler; örneğin Japonca, Korece ve Çince.
Maksimum Uzunluk	8.000 karaktere kadar	4.000 karaktere kadar
Karakter Boyutu	Karakter başına 1 bayt yer kaplar	Unicode/Unicode olmayan karakter başına 2 bayt kaplar
Depolama Boyutu	Gerçek Uzunluk (bayt cinsinden)	Gerçek Uzunluğun 2 katı (bayt cinsinden)
Kullanım	Veri uzunluğu değişken olduğunda veya değişken uzunlukta sütunlar olduğunda ve gerçek veriler her zaman kapasiteden çok daha az olduğunda kullanılır	Sadece depolama amaçlı olduğundan, yalnızca Japon Kanji veya Kore Hangul karakterleri gibi Unicode desteğine ihtiyacınız varsa kullanılır.

SQL Server NVARCHAR örneği

- Aşağıdaki ifade, bir NVARCHAR sütunu içeren yeni bir tablo oluşturur:

```
CREATE TABLE test.sql_server_nvarchar (  
    val NVARCHAR NOT NULL  
);
```

- Bu örnekte, NVARCHAR sütununun dize uzunluğu varsayılan olarak birdir.
- val sütununun dize uzunluğunu değiştirmek için ALTER TABLE ALTER COLUMN ifadesini kullanırsınız:

```
ALTER TABLE test.sql_server_Nvarchar  
ALTER COLUMN val NVARCHAR (10) NOT NULL;
```

- Aşağıdaki ifade, test.sql_server_nvarchar tablosunun val sütununa yeni bir dize ekler:

```
INSERT INTO test.sql_server_varchar (val)
VALUES
    (N'こんにちは');
```

- Dize değeri, sütun tanımında tanımlanan dize uzunluğundan daha kısa bir uzunluğa sahip olduğundan ifade beklendiği gibi çalıştı.
- Aşağıdaki ifade, uzunluğu val sütununun dize uzunluğundan daha büyük olan yeni bir dize verisi eklemeye çalışır:

```
INSERT INTO test.sql_server_nvarchar (val)
VALUES
    (N'ありがとうございました');
```

- SQL Server bir hata verdi ve ifadeyi sonlandırdı:

```
String or binary data would be truncated.
The statement has been terminated.
```

- NVARCHAR sütununda depolanan değerlerin karakter sayısını ve bayt cinsinden depolama boyutunu bulmak için, LEN ve DATALENGTH işlevlerini aşağıdaki gibi kullanırsınız:

```
SELECT
    val,
    LEN(val) len,
    DATALENGTH(val) data_length
FROM
    test.sql_server_nvarchar;
```

nchar, nvarchar ve varchar arasındaki farklar

1. varchar (Değişken Karakter)

- Kullanım: Unicode olmayan karakter verilerini depolar.
- Depolama: Karakter başına 1 bayt kullanır.
- Uzunluk: 8.000 karaktere kadar (veya varchar(max) ile SQL Server'da daha fazla) depolayabilir.
- En iyisi: İngilizce ve standart ASCII karakterleri kullanan diğer diller.

2. nchar (Ulusal Karakter)

- Kullanım: Sabit uzunlukta Unicode karakter verilerini depolar.
- Depolama: Karakter setinden bağımsız olarak karakter başına 2 bayt kullanır.
- Uzunluk: 4.000 karaktere kadar depolayabilir.
- En iyisi: Sabit uzunluğa ihtiyaç duyduğunuz ve birden fazla dili veya özel karakteri (Çince, Arapça vb. gibi) desteklemek istediğiniz senaryolar.

3. nvarchar (Ulusal Değişken Karakter)

- Kullanım: Değişken uzunlukta Unicode karakter verilerini depolar. Depolama: Karakter başına 2 bayt kullanır.
- Uzunluk: 4.000 karaktere kadar (veya nvarchar(max) ile daha fazla) depolayabilir.
- En iyisi: nchar'a benzer şekilde, çoklu dil desteği için idealdir, ancak değişken uzunluğa izin vererek farklı boyutlardaki metinler için daha verimli hale getirir.

- varchar standart karakter kümeleri için en iyisidir, nchar ve nvarchar ise Unicode verileri için uygundur. Sabit uzunluktaki veriler için nchar'ı ve değişken uzunluktaki veriler için nvarchar'ı kullanın.

SQL Server SELECT

- SQL Server'da tablolar, bir veritabanındaki tüm verileri depolayan nesnelerdir. Verileri, bir elektronik tabloya benzer şekilde satır ve sütun biçiminde düzenlerler.
- Her satır bir tablodaki benzersiz bir kaydı ve her sütun o kayıttaki bir alanı temsil eder. Örneğin, aşağıdaki müşteriler tablosu müşteri kimliği, ad, soyad, telefon, e-posta ve adres gibi müşteri verilerini içerir:

customer_id	first_name	last_name	phone	email	street	city	state	zip_code
1	Debra	Burks	NULL	debra.burks@yahoo.com	9273 Thome Ave.	Orchard Park	NY	14127
2	Kasha	Todd	NULL	kasha.todd@yahoo.com	910 Vine Street	Campbell	CA	95008
3	Tameka	Fisher	NULL	tameka.fisher@aol.com	769C Honey Creek St.	Redondo Beach	CA	90278
4	Daryl	Spence	NULL	daryl.spence@aol.com	988 Pearl Lane	Uniondale	NY	11553
5	Charolette	Rice	(916) 381-6003	charolette.rice@msn.com	107 River Dr.	Sacramento	CA	95820
6	Lyndsey	Bean	NULL	lyndsey.bean@hotmail.com	769 West Road	Fairport	NY	14450
7	Latasha	Hays	(716) 986-3359	latasha.hays@hotmail.com	7014 Manor Station Rd.	Buffalo	NY	14215
8	Jacqueline	Duncan	NULL	jacqueline.duncan@yahoo.com	15 Brown St.	Jackson Heights	NY	11372
9	Genoveva	Baldwin	NULL	genoveva.baldwin@msn.com	8550 Spruce Drive	Port Washington	NY	11050
10	Pamelia	Newman	NULL	pamelia.newman@gmail.com	476 Chestnut Ave.	Monroe	NY	10950

- SQL Server, tabloları ve diğer veritabanı nesnelerini mantıksal olarak gruplamak için şemaları kullanır. Örneğin, örnek veritabanımızda iki şema vardır: satış ve üretim.
- Satış şeması satışla ilgili tüm tabloları içerirken, üretim şeması üretimle ilgili tüm tabloları gruplandırır.

- Bir tablodan veri almak için, aşağıdaki sözdizimiyle SELECT ifadesini kullanırsınız:

```
SELECT  
    select_list  
FROM  
    schema_name.table_name;
```

- Bu sözdiziminde:
 - Öncelikle, SELECT ifadesinde veri sorgulamak istediğiniz virgülle ayrılmış sütunların bir listesini belirtin.
 - İkinci olarak, FROM ifadesinde tablo adını ve şemasını belirtin.
- SELECT ifadesini işlerken, SQL Server önce FROM ifadesini, ardından SELECT ifadesini işler, SELECT ifadesi FROM ifadesinden önce görünse bile:



SQL Server SELECT ifadesi örnekleri

- Örnek veritabanındaki müşteriler tablosunu gösteri için kullanalım.

sales.customers
* customer_id
first_name
last_name
phone
email
street
city
state
zip_code

1) Temel SQL Server SELECT ifadesi örneği

- Aşağıdaki sorgu, tüm müşterilerin adlarını ve soyadlarını almak için bir SELECT ifadesi kullanır:

```
SELECT
    first_name,
    last_name
FROM
    sales.customers;
```

İşte sonuç:

first_name	last_name
Debra	Burks
Kasha	Todd
Tameka	Fisher
Daryl	Spence
Charolette	Rice
Lyndsey	Bean
Latasha	Hays
Jacquiline	Duncan
Genoveva	Baldwin
Pamelia	Newman
Deshawn	Mendoza

- Bir sorgunun sonucuna genellikle sonuç kümesi denir.
- Aşağıdaki ifade, tüm müşterilerin adlarını, soyadlarını ve e-postalarını almak için SELECT ifadesini kullanır:

```
SELECT  
    first_name,  
    last_name,  
    email  
FROM  
    sales.customers;
```

Çıktı:

first_name	last_name	email
Debra	Burks	debra.burks@yahoo.com
Kasha	Todd	kasha.todd@yahoo.com
Tameka	Fisher	tameka.fisher@aol.com
Daryl	Spence	daryl.spence@aol.com
Charolette	Rice	charolette.rice@msn.com
Lyndsey	Bean	lyndsey.bean@hotmail.com
Latasha	Hays	latasha.hays@hotmail.com
Jacqueline	Duncan	jacqueline.duncan@yahoo.com
Genoveva	Baldwin	genoveva.baldwin@msn.com
Pamelia	Newman	pamelia.newman@gmail.com
Deshawn	Mendoza	deshawn.mendoza@yahoo.com
Robby	Sykes	robby.sykes@hotmail.com

2) Bir tablonun tüm sütunlarını almak için SQL Server SELECT'i kullanma

- Tüm tablo sütunlarından veri almak için, SELECT listesindeki tüm sütunları belirtebilirsiniz.
- Alternatif olarak, tüm sütunları seçmek için SELECT *'i kısayol olarak da kullanabilirsiniz:

```
SELECT * FROM sales.customers;
```

customer_id	first_name	last_name	phone	email	street	city	state	zip_code
1	Debra	Burks	NULL	debra.burks@yahoo.com	9273 Thome Ave.	Orchard Park	NY	14127
2	Kasha	Todd	NULL	kasha.todd@yahoo.com	910 Vine Street	Campbell	CA	95008
3	Tameka	Fisher	NULL	tameka.fisher@aol.com	763C Honey Creek St.	Redondo Beach	CA	90278
4	Daryl	Spence	NULL	daryl.spence@aol.com	988 Pearl Lane	Uniondale	NY	11553
5	Charolette	Rice	(916) 381-6003	charolette.rice@msn.com	107 River Dr.	Sacramento	CA	95820
6	Lyndsey	Bean	NULL	lyndsey.bean@hotmail.com	769 West Road	Fairport	NY	14450
7	Latasha	Hays	(716) 986-3359	latasha.hays@hotmail.com	7014 Manor Station Rd.	Buffalo	NY	14215
8	Jacqueline	Duncan	NULL	jacqueline.duncan@yahoo.com	15 Brown St.	Jackson Heights	NY	11372
9	Genoveva	Baldwin	NULL	genoveva.baldwin@msn.com	8550 Spruce Drive	Port Washington	NY	11050
10	Pamela	Newman	NULL	pamela.newman@gmail.com	476 Chestnut Ave.	Monroe	NY	10950
11	Deshawn	Mendoza	NULL	deshawn.mendoza@yahoo.com	8790 Cobblestone Street	Monsey	NY	10952
12	Robby	Sykes	(516) 583-7761	robby.sykes@hotmail.com	486 Rock Maple Street	Hempstead	NY	11550

- SELECT * kullanmak, aşına olmadığınız tabloyu incelemek için kullanışlıdır ve özellikle geçici sorgular için faydalıdır.
- Ancak, aşağıdaki temel nedenlerden dolayı üretim kodunda SELECT * kullanmamalısınız:
 - Birincisi, SELECT * kullanmak genellikle uygulamanızın ihtiyaç duyduğundan daha fazla veri alır. Bu gereksiz verilerin veritabanı sunucusundan uygulamaya aktarılması daha uzun sürer ve uygulamayı yavaşlatır.
 - İkincisi, tabloya yeni sütunlar eklenirse, SELECT * uygulamanızın beklemediği yeni sütunlar da dahil olmak üzere tüm sütunları alır. Bu, uygulamanın beklenmedik şekilde davranmasına neden olabilir.
- Aşağıdaki bölümde, SELECT ifadesinin ek maddelerini kısaca tanıtacağız:
 - WHERE : sonuç kümesindeki satırları filtrele.
 - ORDER BY: sonuç kümesindeki satırları bir veya daha fazla sütuna göre sırala.
 - GROUP BY: satırları gruplara ayır.
 - HAVING: grupları filtrele.

3) WHERE ifadesini kullanarak satırları filtreleme

- Satırları bir veya daha fazla koşula göre filtrelemek için WHERE ifadesini kullanırsınız. Örneğin, aşağıdaki SELECT ifadesi Kaliforniya'da bulunan müşterileri bulmak için bir WHERE ifadesi kullanır:

```
SELECT
    *
FROM
    sales.customers
WHERE
    state = 'CA';
```

customer_id	first_name	last_name	phone	email	street	city	state	zip_code
2	Kasha	Todd	NULL	kasha.todd@yahoo.com	910 Vine Street	Campbell	CA	95008
3	Tameka	Fisher	NULL	tameka.fisher@aol.com	769C Honey Creek St.	Redondo Beach	CA	90278
5	Charolette	Rice	(916) 381-6003	charolette.rice@msn.com	107 River Dr.	Sacramento	CA	95820
24	Corene	Wall	NULL	corene.wall@msn.com	9601 Ocean Rd.	Atwater	CA	95301
30	Jamaal	Albert	NULL	jamaal.albert@gmail.com	853 Stonybrook Street	Torrance	CA	90505
31	Williamae	Holloway	(510) 246-8375	williamae.holloway@msn.com	69 Cypress St.	Oakland	CA	94603
32	Araceli	Golden	NULL	araceli.golden@msn.com	12 Ridgeview Ave.	Fullerton	CA	92831
33	Deloris	Burke	NULL	deloris.burke@hotmail.com	895 Edgemont Drive	Palos Verdes Peninsula	CA	90274
40	Ronna	Butler	NULL	ronna.butler@gmail.com	9438 Plymouth Court	Encino	CA	91316
46	Monika	Berg	NULL	monika.berg@gmail.com	369 Vernon Dr.	Encino	CA	91316
47	Bridgette	Guerra	NULL	bridgette.guerra@hotmail.com	9982 Manor Drive	San Lorenzo	CA	94580

- SELECT ifadesi hem WHERE hem de FROM ifadelerini içeriyorsa, SQL Server bunları şu sırayla işler: FROM, WHERE ve SELECT.

FROM



WHERE



SELECT

4) ORDER BY ifadesini kullanarak satırları sıralama

- Bir sonuç kümesindeki satırları sıralamak için ORDER BY ifadesini kullanırsınız. Örneğin, aşağıdaki sorgu müşterileri artan düzende ilk adlarına göre sıralamak için ORDER BY ifadesini kullanır.

```
SELECT
    *
FROM
    sales.customers
WHERE
    state = 'CA'
ORDER BY
    first_name;
```

Çıktı:

customer_id	first_name	last_name	phone	email	street	city	state	zip_code
673	Adam	Henderson	NULL	adam.henderson@hotmail.com	167 James St.	Los Banos	CA	93635
1261	Adelaida	Hancock	NULL	adelaida.hancock@aol.com	669 S. Gortner Street	San Pablo	CA	94806
574	Adriene	Rivera	NULL	adriene.rivera@hotmail.com	973 Yukon Avenue	Encino	CA	91316
1353	Agatha	Daniels	NULL	agatha.daniels@yahoo.com	125 Canal St.	South El Monte	CA	91733
735	Aide	Franco	NULL	aide.franco@msn.com	8017 Lake Forest St.	Atwater	CA	95301
952	Aileen	Marquez	NULL	aileen.marquez@msn.com	8899 Newbridge Street	Torrance	CA	90505
697	Alane	Mccarty	(619) 377-8586	alane.mccarty@hotmail.com	8254 Hildale Street	San Diego	CA	92111
562	Alejandro	Norman	NULL	alejandro.norman@yahoo.com	8918 Marsh Lane	Upland	CA	91784
1288	Allie	Conley	NULL	allie.conley@msn.com	96 High Point Road	Lawndale	CA	90260
701	Alycia	Nicholson	(805) 493-7311	alysia.nicholson@hotmail.com	868 Trusel St.	Oxnard	CA	93035
619	Ana	Palmer	(657) 323-8684	ana.palmer@yahoo.com	7 Buckingham St.	Anaheim	CA	92806
947	Angele	Castro	NULL	angele.castro@yahoo.com	15 Acacia Drive	Palos Verdes Peninsula	CA	90274

- SELECT ifadesi FROM, WHERE ve ORDER BY ifadesini içerdiğinde, SQL Server bunları şu sırayla işler: FROM, WHERE, SELECT ve ORDER BY:

FROM



WHERE



SELECT



ORDER BY

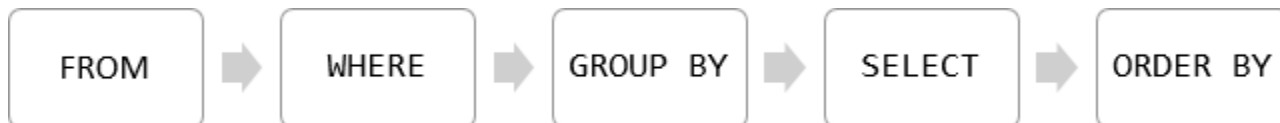
5) Satırları gruplara ayırma

- Satırları gruplara ayırmak için bu GROUP BY ifadeyi kullanırsınız. Örneğin, aşağıda California'da bulunan tüm şehirlerini ve onun şehirdeki müşterisi döner.

```
SELECT
    city,
    COUNT (*)
FROM
    sales.customers
WHERE
    state = 'CA'
GROUP BY
    city
ORDER BY
    city;
```

city	(No column name)
Anaheim	11
Apple Valley	11
Atwater	5
Bakersfield	5
Banning	7
Campbell	10
Canyon Country	12
Coachella	6
Duarte	9
Encino	8
Fresno	5
Fullerton	6
Glendora	8

- Bu durumda SQL Server, ifadeleri şu sırayla işler: FROM, WHERE, GROUP BY, SELECT ve ORDER BY.



6) HAVING ifadesini kullanarak grupları filtreleme

- Grupları bir veya daha fazla koşula göre filtrelemek için HAVING ifadesini kullanırsınız. Örneğin, aşağıdaki ifade, ondan fazla müşterisi olan Kaliforniya'daki şehri döndürmek için HAVING ifadesini kullanır:

```
SELECT
    city,
    COUNT (*)
FROM
    sales.customers
WHERE
    state = 'CA'
GROUP BY
    city
HAVING
    COUNT (*) > 10
ORDER BY
    city;
```

city	(No column name)
Anaheim	11
Apple Valley	11
Canyon Country	12
South El Monte	11
Upland	11

- WHERE ifadesinin satırları filtrelerken HAVING ifadesinin grupları filtrelediğine dikkat edin.

SQL Server Hesaplanan Sütunlar

- Gösterimler için persons adlı yeni bir tablo oluşturalım:

```
CREATE TABLE persons
(
    person_id INT PRIMARY KEY IDENTITY,
    first_name NVARCHAR(100) NOT NULL,
    last_name NVARCHAR(100) NOT NULL,
    dob DATE
);
```

- Ve persons tablosuna iki satır ekleyelim:

```
INSERT INTO
    persons(first_name, last_name, dob)
VALUES
    ('John', 'Doe', '1990-05-01'),
    ('Jane', 'Doe', '1995-03-01');
```

- Persons tablosundaki kişilerin tam adlarını sorgulamak için normalde CONCAT() fonksiyonunu veya + operatörünü aşağıdaki gibi kullanırsınız:

```
SELECT
    person_id,
    first_name + ' ' + last_name AS full_name,
    dob
FROM
    persons
ORDER BY
    full_name;
```

person_id	full_name	dob
2	Jane Doe	1995-03-01
1	John Doe	1990-05-01

- Her sorguya full_name ifadesi first_name + ' ' + last_name eklemek uygun değildir.
- Neyse ki SQL Server bize hesaplanan sütunlar adlı bir özellik sunuyor ve bu özellik aynı tablodaki diğer sütunların değerlerinden türetilen değere sahip bir tabloya yeni bir sütun eklemenize olanak sağlıyor.

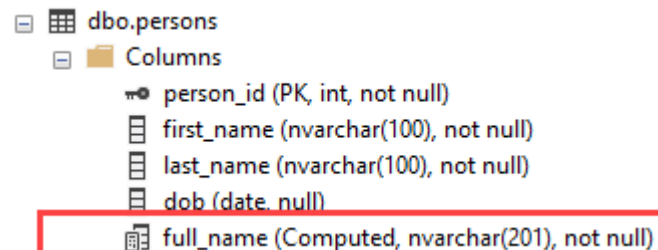
- Örneğin, full_name sütununu ALTER TABLE ADD sütununu aşağıdaki gibi kullanarak persons tablosuna ekleyebilirsiniz:

```
ALTER TABLE persons  
ADD full_name AS (first_name + ' ' + last_name);
```

- Persons tablosundan her veri sorguladığınızda, SQL Server full_name sütununun değerini first_name + ' ' + last_name ifadesine göre hesaplar ve sonucu döndürür. Daha kompakt olan yeni sorgu şu şekildedir:

```
SELECT  
    person_id,  
    full_name,  
    dob  
FROM  
    persons  
ORDER BY  
    full_name;
```

- Persons tablosunu incelerseniz, yeni full_name sütununun sütun listesinde görüldüğünü görebilirsiniz:



dbo.persons
Columns
person_id (PK, int, not null)
first_name (nvarchar(100), not null)
last_name (nvarchar(100), not null)
dob (date, null)
full_name (Computed, nvarchar(201), not null)

Yeni bir tablo oluştururken hesaplanan sütunları tanımlamak için sözdizimi

- Bir tablo oluştururken hesaplanan bir sütunu tanımlamak için aşağıdaki sözdizimini kullanırsınız:

```
CREATE TABLE table_name(  
    ...,  
    column_name AS expression [PERSISTED],  
    ...  
);
```