

6

Amaçlarımız

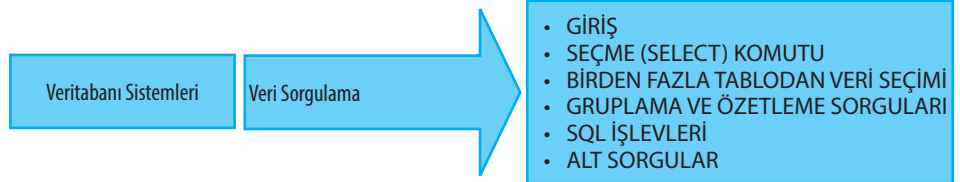
Bu üniteyi tamamladıktan sonra;

- Seçme sorgularının yazım kuralını tanımlayabilecek,
 - Tabloların birbirine bağlanma türlerini sıralayabilecek,
 - Gruplama ve özetleme işlemlerini tanımlayabilecek,
 - SQL işlevlerini sıralayabilecek,
 - Alt Sorgu yapılarını tanımlayabilecek
- bilgi ve becerilere sahip olacaksınız.

Anahtar Kavramlar

- Seçme Sorguları
- Gruplama ve Özetleme
- Tablo Bağlama
- Tablo Birleştirme
- Select, From, Where, Group By, Having
- Alt Sorgu

İçindekiler



Veri Sorgulama

GİRİŞ

SQL dili veritabanı sistemlerinin yönetilmesi için tasarlanmış bir **bildirim** dilidir. Bu dil kullanıcıların ya da istemci yazılımlarının veri ile ilgili isteklerini alarak, depoladığı veriler üzerinde uygulayan bir yapıdadır. Diğer programlama dillerinden farklı olarak işlemleri değil sonuçları tarif eden bir yapısı vardır. Dolayısıyla veritabanı yönetim sistemi tarafından SQL komutları çözümlenerek istenilen işlem yerine getirilir. Bu yapı verinin eklenmesi, işlenmesi ve sorgulanması gibi birçok işlemin istemci ya da kullanıcılar tarafından kolayca yapılmasını sağlamaktadır.

Veritabanı yönetim sistemlerini kullanmamızın temel nedenlerinden biri de verinin farklı görünümünü elde edebilmektir. Organizasyonlarda tüm sistem verilerini barındıran merkezi veritabanları organizasyonda farklı kademe ve görevlerdeki çalışanların veri ihtiyacını karşılamaktadır. Örneğin bir bankada banko çalışanı mudisinin hesap bilgilerine ulaşmakta ve işlem yapmak için bir hesap kaydına ulaşırken bankanın müdürü aylık mevduat ortalaması bilgisine erişmek isteyebilir. Bu ve benzer veri ihtiyaçlarını karşılamak için veritabanı yönetim sistemleri yazılımı ihtiyaca göre veri kümesinin elde etmek üzere tasarlanmıştır. İlişkisel veritabanı yönetim sistemlerinde tablolar hâlinde depolanan verilere ulaşmak için SQL dilinin Veri Sorgulama Dili (DQL- Data Query Language) komutları kullanılmaktadır. Bu ünite de SQL'de veri sorgulama için kullanılan **SELECT** komutu ile ilgilidir. Verinin sadece okunması ile ilgili olan bu komutun çalışma prensibinin öğrenilmesi ilişkisel veritabanı sistemlerinin anlaşılmasında en önemli adımdır.

SELECT komutunun temel yazım şekli verildikten sonra verinin sınırlandırılması, farklı tablolardan verilerin birleştirilmesi, veri kümelerinin birbiri ardı birleştirilmesi, iç-içe sorgular ünite içerisinde aktarılmıştır. SQL sorgu dili komutlarının öğrenilmesinde kullanıcıların bu komutları deneyimlemeleri dilin öğrenilmesi için oldukça önemlidir. Okurların önceki ünitelerde SQL Server Express yazılımını kendi bilgisayarlarına kurarak örnek veritabanı yüklemeleri ve ünite içerisindeki örnek ve sıra sizde sorularını deneyimlemesi konunun öğrenilmesinde fayda sağlayacaktır.

SEÇME (SELECT) KOMUTU

SELECT ifadesi ilişkisel veritabanı yönetim sistemlerinde veri kayıtlarını getirmek için kullanılan komuttur. Bu komut ile bir veri tablosunun tüm satırlarının getirilebileceği gibi bir veya daha çok sayıda tabloda bulunan sütun veya sütunlarının istenilen şartlara uyan kayıtlarını da getirmek mümkündür. **SELECT**, kullanıcıların istedikleri veri kümelerini tarif edebilmeleri için oldukça esnek bir yapıda tasarlanmıştır. Genel yazım kuralı aşağıda verilen komutun köşeli parantez içerisinde belirtilen bölümleri isteğe bağlı seçeneklerdir.

SQL, Yapılandırılmış Sorgu Dili veri ile ilgili yapılacak işlemleri tarif etmek için İngilizce dil yapısında geliştirilmiş bir **bildirim** dilidir (Declarative language).

SELECT komutu VTYS'nin en fazla kullanılan komutudur. Farklı tablolarda depolanan verinin kullanıcıların ihtiyacına cevap verecek şekilde elde edilmesini sağlar.

```
SELECT alan_listesi [INTO yeni_tablo]
[FROM tablo_kaynağı]
[WHERE seçme_koşulları]
[GROUP BY gruplama ifadeleri]
[HAVING Gruplanan_alan_kısıtlamaları]
[ORDER BY sıralama düzeni ASC DESC]
```

“SELECT * FROM Tablo_Adi” seçme sorgusu Tablo_adi tablosundaki tüm alan ve satırları listeler.

Seçme komutu İngilizce dil yapısına göre bir emir cümlesidir ve seç fiili ile başlamaktadır. Yukarıdaki Seçme sorgusunun yazım kuralında “Select alan_listesi” dışında yer alan kısımlar zorunlu değildir. Çalışabilecek en basit seçme sorgusuna örnek olarak “SELECT 'Hello World'” verilebilir. Komutun diğer kısımlarını satırlar hâlinde kısaca aşağıdaki gibi açıklanabilir.

FROM ile başlayan satır seçme işleminin hangi veri kümesinden yapılacağını belirtir. Bu kısımda bir tablo, bir görünüm ya da bir alt sorgu yer alabilir.

WHERE satırı seçme işleminde görüntülenecek ya da hesaba katılacak satırların sınırlandırılması sağlanır. Filtreleme işlemi gibi düşünülebilir.

GROUP BY ifadesi verileri özetleme ya da gruplama işlemleri için kullanılacak bir seçme sorgusu bölümüdür.

HAVING Gruplanan ya da hesaplanan alanların sınırlandırılması için kullanılan bir kısımdır. Where kısmı ile karıştırılmaması gerekir.

ORDER BY sorgunun en sonunda yer alan sıralama işlemidir. Eylem gereği de tüm seçme işlemi tamamlandıktan sonra verinin kullanıcıya hangi sırada gönderileceği bu satıra yazılan komutlarla belirlenir.

Select komutu ile hedeflenen veri kümesini seçmek için komutun yapısında olduğu gibi yukarıdan aşağıya bir tasarım yapılması oldukça önemlidir. Aksi takdirde istenilmeyen sonuçlara ulaşılabilmektedir. Seçme sorgusunun evreni **FROM** sonrasında belirtilen veri kaynakları tarafından oluşturulur. Daha sonraki **WHERE**, **GROUP BY**, **HAVING** ve **ORDER BY** ifadeleri sırasıyla takip eder. Önce **WHERE** ile seçme ve gruplama işlemlerinde hesaba katılmak istenmeyen satırlar çıkarılır sonrasında istenilen kümede gruplama ve özetleme işlemi gerçekleştirilir. Gruplama sonrasında yine görüntülenmek istenmeyen hesaplanmış satırlar **HAVING** komutu ile devre dışı bırakılır. Sürecin en son işlemi ise sıralama olmaktadır. Hedeflenen veri kümelerini elde edilecek sorguların oluşturulmasında bu sıralamanın unutulmaması tasarımın doğru yapılması için önemlidir. **SELECT** ifadesinin mantıksal işleme sırası aşağıdaki gibidir (<https://msdn.microsoft.com/en-us/library/ms189499.aspx>). Bu sıralamada yer alan aşamalar 6. adım dışında izleyen bölümlerde yer alacaktır.

1. FROM
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. WITH CUBE or WITH ROLLUP
7. HAVING
8. SELECT
9. DISTINCT
10. ORDER BY
11. TOP

Bu kitap için kullanılan örnek Northwind veritabanı için örnek bir sorgu ve çalıştırılma sonucu Şekil 6.1’de verilmiştir. Şekil 6.1’de kullanılan SQL Server Management Studio kullanıcıların veritabanı yönetim sistemini yönetmeleri ve tasarımları için oluşturulmuş grafik bir ara yüzdür. Arka planda sunucularda çalışan veritabanı motoru bu arayüze ve bir çok arka plandaki veri istek talebine yanıt vermektedir. Örneğin bir web sunucusunun

Seçme ifadesinin mantıksal işleme sırasının bilinmesi kullanıcıların sorguları daha iyi tasarlamasında yardımcı olmaktadır.

isteği yazılıma farklı ara yazılımlar sayesinde ulaştırılır. Burada SQL dilinin kullanımının öğrenilmesi için bu ara yüzde örnekler gösterilecektir. Şekilde veritabanı yönetim sistemine “**SELECT Ad, Soyadı, [E-posta Adresi] FROM [Çalışanlar]**” sorgusunun sonucu istenmiş ve çalışanlar tablosundaki [Ad], [Soyadı], [E-posta Adresi] alanlarının listelenmesi sağlanmıştır.

Şekil 6.1
 Seçme Sorgularının Yazımı ve Çalıştırılması

1. Yeni boş bir sorgu ekranı açılır.

2. Select Sorgusu yazılır.

3. Yazılan sorgu Execute (F5) komutu ile çalıştırılır.

4. sorgu sonucu görüntülenir.

Ad	Soyadı	E-posta Adresi	
1	Safiye	Göktepe	safiye@northwindtraders.com
2	Atilla	Biber	attila@northwindtraders.com
3	Haluk	Kocak	haluk@northwindtraders.com
4	Meryem	Salah	meriem@northwindtraders.com
5	Serap	Tank	serap@northwindtraders.com
6	Mehmet	Nefer	me
7	Ali	Zare	ali
8	Lale	Güzel	lale
9	Aliye	Türkoglu	aliye

SP1) OFISMACPRO2\Sinan (56) Northwind 00:00:00 9 rows

Select sorgusuyla elde edilen bir veri kümesinin yeni bir tablo olarak kaydedilmesi için **FROM** ifadesinden önce **INTO** tablo_Adı ifadesi yazılmalıdır. Aşağıdaki sorgunun sonucunda kullanıcıya bir tablo listelenmez, bunun yerine “(9 row(s) affected)” iletisi ile yeni bir veritabanı tablosu oluşturulduğu bildirilir.

SELECT Ad, Soyadı, [E-posta Adresi] INTO Çalışanlar_ePosta FROM [Çalışanlar]

Bu ünite de kullanılacak Northwind veritabanı dosyalarına ulaşmak için <https://goo.gl/K839el> adresinde yer alan Northwind.rar dosyasına ulaşabilir ve SQL Server veritabanına ekleyebilirsiniz. Ekleme işlemi bu kitabın 3. Ünitesinde anlatılmıştır. Eğer daha önce kurulmuş ise tekrar kurulması gerekmemeyecektir.



İNTERNET

Benzersiz Değerlerin Elde Edilmesi (DISTINCT)

Seçme sorgularında sıkça kullanılan ifadelerden birisi de bir alandaki verilerin tekrarsız olarak görüntülenmesini sağlayan **DISTINCT** ifadesidir. 9 satır verinin yer aldığı [Çalışanlar] tablosu için yazılan “**SELECT DISTINCT [İş Unvanı] FROM [Çalışanlar]**” sorgusu [İş Unvanı] alanındaki tüm verileri tekrarsız olarak “Genel Müdür Yardımcısı, Satış Koordinatörü, Satış Müdürü, Satış Temsilcisi” şeklinde dört satır ile görüntüler. **DISTINCT** ifadesi kayıtların sayılması ile ilgili işlevlerde de benzersiz satır sayısının sayılması ile ilgili kullanılmaktadır.

Sorgu Sonuçlarının Sıralanması (ORDER BY)

Seçme komutunun mantıksal işleme sırasının son iki adımı olan **ORDER BY** ve **TOP** ifadeleri elde edilen veri kümesinin kullanıcı ya da istemci yazılıma iletilmeden önce sıralanmasını sağlar. **SELECT** komutunun son kısmında yer alan **ORDER BY** ifadesinden sonra veri kümesinin istenilen alanlara göre sıralanması sağlanır. Aşağıda *[Çalışanlar]* tablosunun *[Şehir]*, *[Soyadı]* ve *[Ad]* alanlarını sorgulayarak elde edilen veri kümesinin önce *[Şehir]*'e göre artan, *[Ad]*'a göre azalan ve *[Soyadı]* alanına göre artan sıralayan bir seçme sorgusu yer almaktadır.

```
SELECT Şehir, Soyadı, Ad FROM Çalışanlar
ORDER BY Şehir, Ad DESC, Soyadı
```

Yukarıdaki örnek sorguya dikkat edilirse artan sıralama yapılan alan adlarının yanında herhangi bir ifade yok iken azalan sıralanmak istenen *[Ad]* alanının yanında **DESC** (DESCending: azalan) kelimesi yer almaktadır. Sorgu ifadelerinde sıralama yapılacak alanların sonuç kümesinde bulunma zorunluluğu yoktur.

Bazı durumlarda da **SELECT** komutu ile elde edilen verinin en üst satırlarının görüntülenmesi gerekebilir. Örneğin en fazla satış yapılan on müşteri ya da en fazla ciro yapılan ürün listesi elde edilmek istenebilir. VTYS tüm veri hazırlama sürecini yerine getirdikten sonra en son **TOP** kısmını işleyerek en üstte yer alan satırlar görüntülenir ya da iletilir. **TOP** ifadesinin kullanımında doğal olarak verinin sıralanması da söz konusudur. Aksi durumda en üstte yer alacak verilerin çok bir anlamı olmayacaktır. Northwind veritabanında liste fiyatı en fazla olan üç ürünü bulmak için aşağıdaki sorgu yeterli olacaktır.

```
SELECT TOP (3) [Ürün Adı], [Liste Fiyatı]
FROM Ürünler ORDER BY [Liste Fiyatı] DESC
```

Sorguda **TOP** ifadesinden sonra parantez içinde yazılan '3' ifadesi anlaşılacağı üzere en üstte yer alan üç satırın görüntülenmesini sağlar. Diğer taraftan sorgunun son kısmında yer alan sıralama komutunun yapısına dikkat edilmelidir.

Seçimlerin Sınırlanması (WHERE)

Veri sorgulamasında kayıtların belirli ölçütlere göre sınırlanması ya da diğer bir ifadeyle tablolar içinde aranması için **WHERE** ifadesi kullanılır. Basit bir örnek ile başlanarak **WHERE** komutu ile gerçekleştirilebilecek koşul ifadelerine bu kısımda yer verilecektir. Birçok istemci yazılımı, işlem yapılmak istenen nesnenin (müşteri, hesap, ürün, öğrenci vb.) girişi yapılarak veritabanında sorgular. Bu gibi işlemlerde Select sorgusunun **WHERE** komutu sonrası aranacak bilginin eşleştirilmesi yapılır. *[Ürünler]* tablosunda "Ceviz" adlı ürünün *[Liste Fiyatı]* ve *[Ürün Kodu]* bilgisi sorgulanmak için oluşturulmuş aşağıdaki sorguda **WHERE**'den sonra parantez içinde "[Ürün Adı]='Ceviz'" şeklinde yazılan ifade ilgili ürünün bilgisinin veri tablosu içerisinde bulunarak görüntülenmesini sağlar. Bu ifade de parantez gerekli değildir ancak koşul ifadelerinin sayısının fazla olması parantez kullanımını oldukça önemli hâle getirmektedir.

```
SELECT [Ürün Kodu], [Ürün Adı], [Liste Fiyatı] FROM Ürünler
WHERE ([Ürün Adı] = 'Ceviz')
```

MS SQL Server VTYS'i yazılımında sıkça kullanılan koşul ifadeleri Tablo 6.1'de yer almaktadır. Bazı ifadelerin yanında köşeli parantez içerisindeki *[NOT]* ifadesi ilgili koşulun değilini ifade etmek için de kullanılabilir.

SQL komutlarının tek satır olarak yazılması zorunluluğu yoktur. Derleyici satır gözetmeksizin **SELECT** komutunun yazım kuralına göre ifadeleri dikkate alır.

Operatör	Tanımı	Where sonrası yazım şekli
=	Eşitir	[Ürün Adı] = 'Ceviz'
<>	Eşit Değildir (!=)	[Kategori] <> 'Çerezler'
>	Büyük	[Liste Fiyatı] > 50
<	Küçük	[Liste Fiyatı] < 50
>=	Büyük veya eşit	[Ürün Adı] >= 'H'
<=	Küçük veya eşit	[Liste Fiyatı] <= 10
[NOT] BETWEEN	Belirtilen değerler arasında. (Sınır değerleri listeye dahil edilir)	[Liste Fiyatı] BETWEEN 10 AND 20
[NOT] LIKE	Metin içerisindeki desene göre kısıtlama	[Ürün Adı] LIKE 'C%' (C ile başlayan ürünler)
[NOT] IN	Bir alanın birden fazla değer ile sınırlandırılması	[Ürün Adı] IN ('Ceviz', 'Hint Çayı')
EXISTS	Alt sorgular ile mevcut sorgunun sınırlandırılması sağlanır	EXISTS (Select * From Where)
IS [NOT] NULL	Boş satırların bulunması	[Liste FİYATI] IS NOT NULL
CONTAINS	Metin içerisinde karakter bazlı aranacak ifadeler	CONTAINS (ŞikayetMetni, 'Son Kullanım')
FREE TEXT	Metin içerisinde doğal dil özelliklerine göre ifade arama	FREETEXT (Belge, 'go fast')

Tablo 6.1
SQL Koşul Operatörleri
Listesi

Listede yer alan **CONTAINS** ve **FREETEXT** ifadeleri uzun metin verisi içeren alanlarda ilgili metin ifadelerinin bulunması için kullanılır. Arama yapılacak alanların daha önce Full-Text Indeks olarak tanımlanması gerekmektedir.

Ürünler Tablosunda [Liste fiyatı] 40 ile 60 arasında olan ve [Ürün Adı] içerisinde "A" harfi olan ürünleri bulmak istersek aşağıdaki koşul ifadelerinin yazılması gerekmektedir.

```
SELECT [Ürün Kodu], [Ürün Adı], [Liste Fiyatı] FROM Ürünler
WHERE ([Liste Fiyatı] BETWEEN 20 AND 40) AND ([Ürün Adı]
LIKE 'A%')
```

Sorgu cümlecği incelendiğinde **WHERE** ifadesinden sonra **AND** ile birbirine bağlanmış iki koşulun olduğu görülmektedir. İlk kısım liste fiyatı koşulu ve diğer koşul ise ürün adı ile ilgili koşuldur. Koşullar kayıt bazlı değerlendirildiği için her bir kaydın bu koşulu sağlayıp sağlamadığı kontrol edilecek ve her ikisini birlikte sağlayanlar kullanıcıya görüntülenecektir. İkinci koşuldaki **LIKE**'dan sonra gelen **'A%'** ifadesi, metin içerisindeki % karakteri herhangi bir metin katarını temsil eder. Joker karakter ismi de verilen bu semboller ile metin içerisindeki desenler aranabilmektedir. **LIKE** ile kullanılabilecek ifadeler aşağıda listelenmiştir

- %: Herhangi bir karakter dizisi ya da hiçbir metin için kullanılır. 'A%' ifadesi "A" ile başlayan ya da sadece 'A' içeren alanları sınırlandırabilir.
- _ (Alt çizgi): Herhangi tek bir karakter. '_eri' ifadesi ilk harfi farklı olabilecek "Geri", "Seri" gibi ifadeleri bulabilmektedir.
- []: Köşeli parantez aralığında tanımlanan harfler ile sınırlandırılmaktadır. '[a-f]' veya '[abcdef]' şeklinde yazılabilir. '[C-T]apa' ifadesi ile "Çapa", "Sapa" gibi metinler sınırlandırılabilir.
- [^] Belirtilen aralıkta karakter içermeyen metinler bulunur. 'File[^1]%' ifadesi içerisinde 1 olamayan ancak 'File' ile başlayan metinleri bulabilecektir.

Koşul ifadeleri arasında **AND** ve **OR** ifadeleri kullanılabilir. Mantıksal sına ifadeleri olan bu ifadelerden **AND** birbirine bağlanan ifadelerin her ikisinin de gerçekleşmesini zorunlu kılar. **OR** ifadesi ise koşullardan en az birinin yerine getirilmesi durumunda ilgili satırı veri kümesine dahil eder.

SIRA SİZDE



Northwind veritabanında

SELECT [Ürün Kodu], [Ürün Adı], [Liste Fiyatı] **FROM** Ürünler
WHERE ([Ürün Adı] = 'Ceviz') **AND** ([Ürün Adı] = 'Zeytin Yağı')
sorgusunun nasıl bir çıktı vereceğini bulunuz.

BİRDEN FAZLA TABLODAN VERİ SEÇİMİ

İlişkisel veritabanı yönetim sistemlerinde gerçek dünya varlıklarının tablolar şeklinde gösterimi söz konusudur. Bu yapılarda sistem, verileri birbirine bağlı çok sayıda tablolar hâlinde organize edilir. Daha önceki konularda da bahsedildiği üzere tablolar arasında farklı ilişki türleri kurulabilmektedir. Veritabanı tasarımı esnasında tablolar ve aralarındaki ilişkilere karar verilmektedir. Ancak istenen veri kümelerini elde etme de tablo yapılarını ve aralarındaki ilişki özelliklerini bilmek gerekmektedir.

Basit olarak iki tabloyu birbirine bağlamak için **WHERE** komutu kullanılabilir. Örnek veritabanında hangi müşterinin hangi siparişleri verdiğini listelemek için [Müşteriler] ve [Siparişler] tablosunu **WHERE** komutu ile bağlayarak bir sorgu yazılmak istenirse aşağıdaki sorgu komutu yazılabilir.

```
SELECT Top (3) Müşteriler.Şirket, Müşteriler.Soyadı,
Müşteriler.Ad, Müşteriler.[No], Siparişler.[Müşteri No]
FROM Siparişler, Müşteriler
WHERE Siparişler.[Müşteri No] = Müşteriler.[No]
```

İki tabloyu birbiri ile ilişkilendirmek ya da diğer bir ifadeyle bağlamak için **FROM** cümleciğinden sonra [Siparişler], [Müşteriler] tablolarını yazmak ve **WHERE** ile iki tablo arasındaki bağlantının belirtilmesi gerekir. “Siparişler.[Müşteri No] = Müşteriler.[No]” eşitliği her iki tablodaki bu değerleri eşlemek anlamına gelir. İki tabloyu bağlamak [Siparişler] tablosunda yer alan [Müşteri No] değerini [Müşteriler] tablosundaki [No] alanında karşı gelen satırdaki değerleri birleştirmektedir. Tablo 6.2’de yukarıdaki sorgunun çalıştırılması sonucu elde edilen sonuç tablosu yer almaktadır. Bu tabloda [Müşteri No] ve [No] alanlarındaki değerlerin aynı olduğuna dikkat ediniz.

Tablo 6.2
Where ile
İlişkilendirilmiş İki
Tablo Sorgusunun
Sonucu

Şirket	Soyadı	Ad	No	Müşteri No
Şirket A	Berber	Aliye	1	1
Şirket A	Berber	Aliye	1	1
Şirket C	Aksak	Timur	3	3

JOIN Komutu ile Tabloların Bağlanması

Tabloları bir sorgu içerisinde birbiri ile bağlamanın en kolay ve yaygın yolu **JOIN** komutudur. **JOIN** komutu ile sadece iki değil çok sayıda tablo birbirine bağlanabilmektedir. Ayrıca farklı bağlanma türleri ile kullanıcıların istenilen veri kümelerini elde etmelerine olanak sağlar. Farklı **JOIN** komutları olmasına karşı yazım kuralı genelde benzerdir. Aşağıdaki yazım kuralı **INNER JOIN** komutu için yazılmıştır.

```
SELECT Alan_Adi FROM Tablo1 INNER JOIN Tablo2 ON Tablo1
Alan1=Tablo2.Alan2.
```


FROM ifadesinden sonra bağlanacak tablo adlarının arasında **INNER JOIN** ifadesi yer almakta **ON** ifadesinden sonra da bağlantı eşitliği tanımlanmaktadır. Bu kısımda farklı mantıksal ilişkilerle de tablolar bağlanabilir. Birden fazla tablonun bulunduğu ya da iç-içe sorguların kullanılmasında tablo isimlerini uzun olarak yazmaktansa bunun yerine kısa takma adlar kullanılabilir. Bu durumda sorgular daha kısa ve anlaşılır hâle gelebilir. Örneğin aşağıdaki sorguda *[Müşteri]* tablosu **FROM** ifadesinden sonra “**Müşteriler AS M**” şeklinde yazılarak bu sorgu içerisinde M takma adı ile kullanılabilir. Benzer şekilde *[Siparişler]* tablosu da “**Siparişler AS S**” ifadesi ile artık “S” harfi ile kullanılmıştır.

```
SELECT M.Şirket, M.Soyadı, M.Ad, S.[Müşteri No], M.No
FROM Siparişler AS S INNER JOIN Müşteriler AS M ON S.
[Müşteri No] = M.No
```

JOIN ifadesi ile tabloları bağlama işlemlerini basit iki tablo kullanarak açıklamak anlaşılabilirliğini kolaylaştıracaktır. Aşağıda A ve B adında iki adet tablo olduğunu varsayalım. Bu iki tablonun birbirine **JOIN** komutu ile bağlanmasında **INNER**, **OUTTER** ve **CROSS** olmak üzere üç farklı seçenek bulunmaktadır.

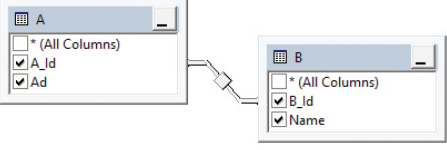
A Tablosu	
A_Id	Ad
1	Bir
2	İki
4	Dört
8	Sekiz

B Tablosu	
B_Id	Name
1	One
2	Two
3	Three
7	Seven

Tablo 6.3

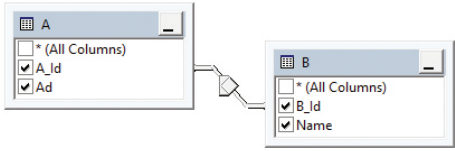
A ve B Tablolarının ve Tablo İçerikleri

Inner Bağlantı: Her iki tablonun da eşleşen alanlarının seçilmesi sağlanır. Diğer bir deyişle kesişim kümesi seçilir. Aşağıda *[A]* ve *[B]* tablosunun **INNER JOIN** komutu ile birleştiren sorgu ve sonucu yer almaktadır. Seçme işleminin sonucunda her iki kümenin bağlantılı olduğu alandaki ortak değerlerin sayısı kadar satır görüntülenir.

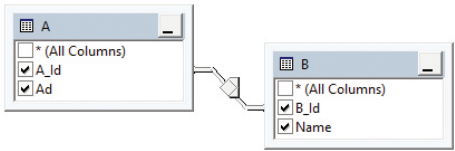
Inner Join	
	Sonuç
	A_Id Ad B_Id Name
	1 Bir 1 One
	2 İki 2 Two
<pre>SELECT A.A_Id, A.Ad, B.B_Id, B.Name FROM A INNER JOIN B ON A.A_Id=B.B_Id</pre>	

Outer Bağlantıları: Outer bağlantıları soldan, sağdan ve tüm (Full) şeklinde üç farklı bağlantı türünden oluşmaktadır.

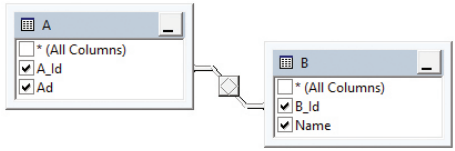
LEFT OUTER JOIN bağlantısı bu ifadenin solunda ve sağında kalan tabloların konumu baz alınarak soldaki tablonun tüm içeriğinin görüntülenmesi sağlar. Aşağıdaki tabloda **LEFT OUTER JOIN** solundaki *[A]* tablosunun tüm verisinin sonuçta yer alması sağlanmıştır. *[B]* tablosunda 4 ve 8 değerlerinin karşılığı olmadığı için **NULL** (boş) olduğu belirtilmiştir. Sonuçta soldaki tablonun eleman sayısı kadar satır görüntülenir.

Left Outer Join		Sonuç			
		A_Id	Ad	B_Id	Name
		1	Bir	1	One
		2	İki	2	Two
		4	Dört	NULL	NULL
		8	Sekiz	NULL	NULL
SELECT A.A_Id, A.Ad, B.B_Id, B.Name FROM A LEFT OUTER JOIN B ON A.A_ Id= B.B_Id					

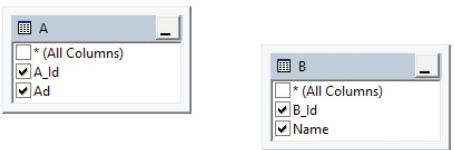
LEFT OUTER JOIN bağlantısının tam tersi olan **RIGHT OUTER JOIN** ise ifadenin sağında kalan tablodaki tüm değerlerin seçilmesini sağlar. Sorgu sonucundaki sağdaki tablonun satır sayısı kadar veri kümesi elde edilir.

Right Outer Join					
	Sonuç				
	A_Id	Ad	B_Id	Name	
	1	Bir	1	One	
	2	İki	2	Two	
	NULL	NULL	3	Three	
NULL	NULL	7	Seven		
<pre>SELECT A.A_Id, A.Ad, B.B_Id, B.Name FROM A RIGHT OUTER JOIN B ON A.A_Id= B.B_Id</pre>					

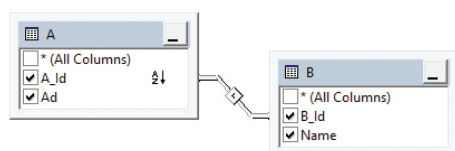
FULL OUTER JOIN ise her iki tabloda yer alan tüm satırların görüntülenmesini sağlar. Aşağıdaki tabloda ortak olan satırların bir kez yer aldığına dikkat ediniz. Bileşim kümesini elde eden bu bağlantıda kesişim değerleri 1 kez yer aldığından eleman sayısı $[A]$ tablosunun $[B]$ den farklı eleman sayısı, $[B]$ tablosunun $[A]$ tablosundan farklı eleman sayısı ve Her iki kümenin ortak elemanlarının sayısının toplamından oluşur.

Full Outer Join																																	
	<table><tr><th colspan="4">Sonuç</th></tr><tr><th>A_Id</th><th>Ad</th><th>B_Id</th><th>Name</th></tr><tr><td>1</td><td>Bir</td><td>1</td><td>One</td></tr><tr><td>2</td><td>İki</td><td>2</td><td>Two</td></tr><tr><td>4</td><td>Dört</td><td>NULL</td><td>NULL</td></tr><tr><td>8</td><td>Sekiz</td><td>NULL</td><td>NULL</td></tr><tr><td>NULL</td><td>NULL</td><td>3</td><td>Three</td></tr><tr><td>NULL</td><td>NULL</td><td>7</td><td>Seven</td></tr></table>	Sonuç				A_Id	Ad	B_Id	Name	1	Bir	1	One	2	İki	2	Two	4	Dört	NULL	NULL	8	Sekiz	NULL	NULL	NULL	NULL	3	Three	NULL	NULL	7	Seven
Sonuç																																	
A_Id	Ad	B_Id	Name																														
1	Bir	1	One																														
2	İki	2	Two																														
4	Dört	NULL	NULL																														
8	Sekiz	NULL	NULL																														
NULL	NULL	3	Three																														
NULL	NULL	7	Seven																														
<pre>SELECT A.A_Id, A.Ad, B.B_Id, B.Name FROM A FULL OUTER JOIN B ON A.A_ Id= B.B_Id</pre>																																	

Cross Bağlantı: Herhangi iki alanın eşleştirilmediği bağlantı türü çapraz (cross) bağlantıdır. Bu bağlantıda $[A]$ daki her bir satır için $[B]$ deki tüm satırlar tekrarlanır. Böylece her iki tablonun eleman sayılarının çarpımı çapraz bağlantı sonucu oluşan listenin satır sayısını verir. Çapraz tablo bağlantısı şablon veri tablolarının üretilmesinde kullanılabilir.

Cross Join				
	A_Id	Ad	B_Id	Name
	1	Bir	1	One
	2	İki	1	One
	4	Dört	1	One
	8	Sekiz	1	One
	1	Bir	2	Two
	2	İki	2	Two
	4	Dört	2	Two
	8	Sekiz	2	Two
	1	Bir	3	Three
	2	İki	3	Three
	4	Dört	3	Three
	8	Sekiz	3	Three
	1	Bir	7	Seven
	2	İki	7	Seven
	4	Dört	7	Seven
	8	Sekiz	7	Seven
<pre>SELECT A.A_Id, A.Ad, B.B_Id, B.Name FROM A CROSS JOIN B</pre>				

Yukarıda örneklerle açıklanan bağlantı türlerinin yanı sıra bağlantı eşitliğinde düzenleme yapılarak farklı kümelere erişilebilir. Aşağıdaki örnekte [A] tablosundaki A_Id değerlerini [B] tablosundaki B_Id değerlerinden küçük olacak şekilde bağlanmıştır.

Inner Join				
	A_Id	Ad	B_Id	Name
	1	Bir	2	Two
	1	Bir	3	Three
	1	Bir	7	Seven
	2	İki	7	Seven
	2	İki	3	Three
	4	Dört	7	Seven
<pre>SELECT A.A_Id, A.Ad, B.B_Id, B.Name FROM A INNER JOIN B ON A.A_Id < B.B_Id</pre>				

Tablo 6.3'de yer alan A ve B tabloları ile hazırlanacak bir sorguda A_Id ve B_Id sütunlarının birbirine eşit olmaması istendiğinde nasıl bir sorgu yazılabilir ve sonucu ne olurdu araştırınız.



SIRA SİZDE

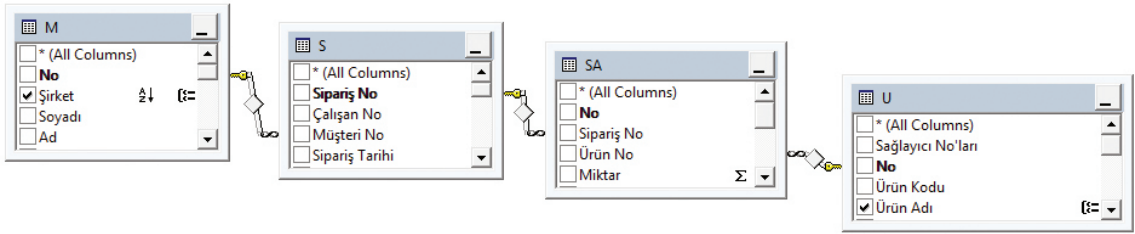
Çok Sayıda Tablonun Bağlanması

Çok sayıda tablonun birbirine bağlanması için farklı bir bağlantı türü yoktur. JOIN komutu aynı yapıyla kullanılır ancak ON kısmında birbiri ile ilişkili alanların doğru tarif edilmesi gerekir. Northwind örnek tablosunda hangi müşterinin hangi üründen kaç adet ürün sipariş ettiğinin sorgulandığını düşünelim. Northwind tablosunun yapısı in-

celendiğinde *[Siparişler]* tablosunda müşteri ve sipariş numarasının bulunduğu, *[Sipariş Ayrıntıları]* tablosunda hangi siparişte hangi üründen ne miktarda sipariş edildiği bilgilerinin yer aldığını görülmektedir. Müşteri ve Ürün bilgileri de *[Müşteriler]* ve *[Ürünler]* tablosunda yer aldığına göre bu dört tablonun birbirine bağlanarak istenilen veri kümesine ulaşılabileceği açıktır. Dört tabloyu birbirine bağlamak için aşağıdaki bağlantı ifadesinin yazılması gerekir.

```
FROM Siparişler AS S
INNER JOIN [Sipariş Ayrıntıları] AS SA ON S.[Sipariş No] =
SA.[Sipariş No]
INNER JOIN Ürünler AS U ON SA.[Ürün No] = U.No
INNER JOIN Müşteriler AS M ON S.[Müşteri No] = M.No
```

Tabloların bağlantılarının oluşturulmasında **INNER JOIN** yapısı kullanıldığından sıralamanın önemi yoktur. Önemli olan **ON** ifadelerinden sonra hangi alanın hangi alanla bağlanacağını belirttiği eşitleme kısmıdır. Bu alanda bir hatanın olması elde edilen veri kümesinin hatalı olmasına neden olur. Bu bağlantı türü grafik sorgu tasarım ekranında aşağıdaki gibi görüntülenir.




Veri Kümelerinin Birleştirilmesi

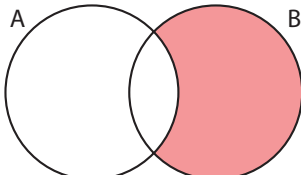
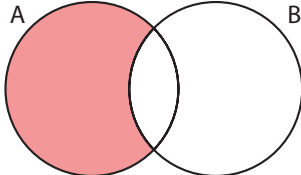
İki veya daha fazla veri kümesinin birleştirilmesi için veri kümelerinin aynı sayıda ve türde alanlarının olması gerekir.

Tabloların birbirine bağlanması ilişkili alanların eşlenerek bağlanan tabloda karşılığının aranması için kullanılır. Veri birleştirme ise birbiri ile aynı sayıda ve türde alanları olan veri kümelerinin alt alta birleştirilmesi, kesişim kümelerinin ve farklarının bulunması işlemidir. Örneğin depolardaki stok miktarının hesaplanmasında depoya giren ve çıkan tablolarının birleştirilerek stok miktarının hesaplanmasında birleştirme sorguları kullanılabilir. Veri kümeleri **INTERSECT**, **EXCEPT**, **UNION**, **UNION ALL** komutları ile birleştirilebilir. Aşağıda tablo 6.2'deki A ve B tabloları için bu komutların kullanımı örneklenmiştir.

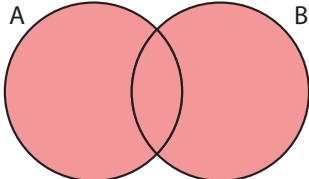
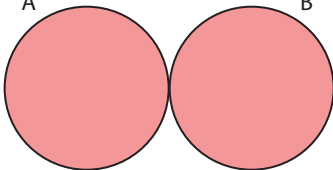
Intersect: İki veya daha çok veri kümesindeki satırların kesişiminin bulunmasında kullanılır. Aşağıdaki sorgu ifadesi dikkatli incelendiğinde iki farklı sorgunun arasına **INTERSECT** ifadesi kullanılarak birleştirildiği görülebilir. Sonuç veri kümesinin alan adının ilk sorgudaki alan adı olduğu görülmektedir.

<pre>SELECT A_Id FROM A INTERSECT SELECT B_Id FROM B</pre>	<table><tr><th>A_Id</th></tr><tr><td>1</td></tr><tr><td>2</td></tr></table>	A_Id	1	2	
A_Id					
1					
2					

Except: Bu ifade ile birleştirilen sorgular veri kümeleri arasındaki farkı listelemek için kullanılırlar. İlk sorgunun aşağıda verilen kümeden farkı belirlenerek görüntülenir.

<pre>SELECT B_Id FROM B EXCEPT SELECT A_Id FROM A</pre>	<table><tr><th>B_Id</th></tr><tr><td>3</td></tr><tr><td>7</td></tr></table>	B_Id	3	7	
B_Id					
3					
7					
<pre>SELECT A_Id FROM A EXCEPT SELECT B_Id FROM B</pre>	<table><tr><th>A_Id</th></tr><tr><td>4</td></tr><tr><td>8</td></tr></table>	A_Id	4	8	
A_Id					
4					
8					

Union ve Union All: **UNION** ve **UNION ALL** komutları kümelerin birleşim kümesini elde etmek için kullanılır. **UNION** ifadesi ile birleştirilmiş veri kümelerinde kesişim kümesi bir kez yer alır. **UNION ALL** ile birleştirilmiş kümelerde ise herhangi bir işlem yapılmadan iki liste alt alta eklenir.

<pre>SELECT A_Id FROM A UNION SELECT B_Id FROM B</pre>	<table><tr><th>A_Id</th></tr><tr><td>1</td></tr><tr><td>2</td></tr><tr><td>3</td></tr><tr><td>4</td></tr><tr><td>7</td></tr><tr><td>8</td></tr></table>	A_Id	1	2	3	4	7	8			
A_Id											
1											
2											
3											
4											
7											
8											
<pre>SELECT A_Id FROM A UNION ALL SELECT B_Id FROM B</pre>	<table><tr><th>A_Id</th></tr><tr><td>1</td></tr><tr><td>2</td></tr><tr><td>4</td></tr><tr><td>8</td></tr><tr><td>1</td></tr><tr><td>2</td></tr><tr><td>3</td></tr><tr><td>7</td></tr></table>	A_Id	1	2	4	8	1	2	3	7	
A_Id											
1											
2											
4											
8											
1											
2											
3											
7											

Tabloların birleştirilmesinde alan sayıları eşleşmediğinde SQL Server **“All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.”** iletilisi ile kullanıcıyı bilgilendirir. Yine benzer şekilde veri tipi uyumsuzluğu durumunda ise **“Conversion failed when converting the nvarchar value ‘One’ to data type tinyint.”** benzeri bir hata üretilir.

GRUPLAMA VE ÖZETLEME SORGULARI

Seçme sorguları ile ilgili buraya kadar uygulanan örnek ve komutlarda tablodaki değerlerin sıralanması, listelenmesi ve birleştirilmesi gibi işlemler gerçekleştirildi. Veritabanı Yönetim Sistemlerinin önemli bir görevi ise kullanıcı isteklerine göre bu verilerin özetlenmesi, gruplanması işlemleridir. Bu sayede veri kümesi işlenerek mevcut sistem hakkında bilgi elde

Veritabanı tablolarında özetleme ve gruplama işlemlerinde **GROUP BY** komutu kullanılır

edilir. Aylık toplam satış tutarları, satışı yapılan ürün sayıları, en çok satış yapan personel listeleri ve benzeri bilgilerin elde edilmesi için kullanılan SQL komutu **GROUP BY**'dir.

GROUP BY komutu bir veri kümesinde belirlenen alanların içerdiği verinin tekrarsız hâle getirerek özetlenmesidir. Örnek veritabanında yer alan *[Siparişler]* tablosunda

“**SELECT [Çalışan No] FROM Siparişler**”

sorgusu çalıştırıldığında siparişi alan çalışan numaraları 47 satır olarak listelenecektir. Diğer taraftan biz sadece hangi çalışanların satış yaptığı sorusunun yanıtını arıyorsak bu sorgu yerine

“**SELECT [Çalışan No] FROM Siparişler GROUP BY [Çalışan No]**”

sorgusunu çalıştırmalıydık. Bu sorguda **GROUP BY** kısmından sonra *[Çalışan No]* alanının yer aldığına dikkat edilmelidir. *[Çalışan No]* alanının tekrarsız olarak elde edilmesi sağlanarak 8 adet veri görüntülenecektir.

Gruplama işlemlerinde hangi alanın gruplanacağı ve hangi alanının görüntüleneceğini belirlemek çok önemlidir. Bu süreçte yapılan hatalar sorgunun çalışmamasına neden olacaktır. Örneğin kullanıcının aşağıdaki sorguyu yazması durumunda neler olabileceğini bir düşünelim.

“**SELECT [Müşteri No] FROM Siparişler GROUP BY [Çalışan No]**”

Kullanıcı *[Siparişler]* tablosunda *[Çalışan No]* alanına göre gruplama yaparak *[Müşteri No]* alanını görüntülemek istemiştir. Sorgunun yapısı irdelenirse tablodaki bir alana göre gruplama yapılırken diğer bir alandaki değerlerinin görüntülenmesi mümkün değildir. Yukarıdaki bu hatalı istek veritabanı sistemi tarafından “**Column ‘Siparişler. Müşteri No’ is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.**” hata mesajı ile geri dönecektir. Çünkü *[Müşteri No]* alanı ile ilgili ya bir özetleme işlemi (sayma gibi) ya da gruplama yapılmalıdır. Şekil 6.2’de yukarıdaki hatalı sorgunun nasıl doğru ve anlamlı bir sorgu hâline getirildiği görülebilir. Şekildeki ilk sorguda Her iki alana göre gruplama yapılmış ancak tek bir alan görüntülenmiştir. Bu bir hata oluşturmaz çünkü gruplanan bir alanın görüntülenmesi zorunlu değildir. Şekildeki ikinci örnekte ise *[Müşteri No]* alanı **Count** fonksiyonu ile sayılmak suretiyle gruplanmadan yer almıştır. Diğer bir deyişle her bir çalışanın kaç satırda yer aldığı hesaplanmıştır.

Şekil 6.2

Group By Örnek Sorgular ve Sonuçları

SQLQuery2.sql - lo...ACPRO2\Sinan (56))

```
SELECT [Müşteri No] FROM Siparişler
Group BY [Çalışan No],[Müşteri No]
```

100 %

Results Messages

	Müşteri No
1	1
2	7
3	10
4	11
5	20

(12.0 SP1) | OFISMACPRO2\Sinan (56) | Northwind | 00:00:00 | 24 rows

SQLQuery2.sql - lo...ACPRO2\Sinan (56))

```
SELECT [Çalışan No], Count([Müşteri No]) As MUSS_AY
FROM Siparişler
Group BY [Çalışan No]
```

100 %

Results Messages

	Çalışan No	MUSS_AY
1	1	12
2	2	3
3	3	6
4	4	8
5	5	4

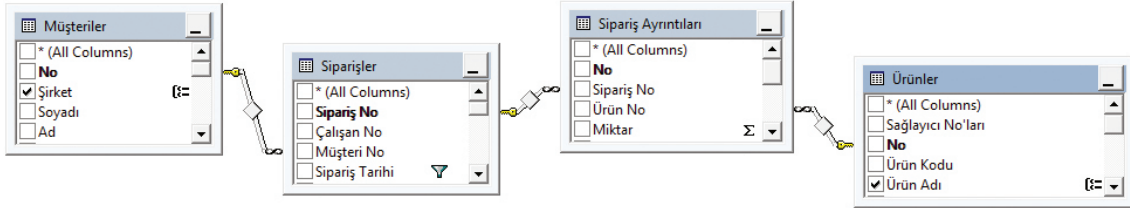
localhost (12.0 SP1) | OFISMACPRO2\Sinan (56) | Northwind | 00:00:00 | 8 rows

Gruplama işlemleri birden fazla tablo üzerinde gerçekleştirilen sorgulamalarda da uygulanabilir. Ünitenin başında verilen seçme sorgularının mantıksal işleme sırasına göre **GROUP BY** işleminden önce tablolardan seçme işlemi ve varsa **WHERE** ile sınırlama işlemi uygulanır.

ÖRNEK 6.1

Northwind veritabanında 2006 Nisan ayında hangi müşterinin hangi üründen kaçar adet sipariş verdiğini elde eden sorguyu yazınız.

İstenilen verinin hazırlanması için öncelikle bu bilginin hangi tablolardan elde edileceğinin tespit edilmesi ve bu tabloların doğru şekilde bağlanması gerekir. Daha sonrasında 2006 Nisan ayı için bir sınırlama ve daha sonra da ilgili alanları seçme ve gruplama işlemi gerçekleştirilir. Müşteri adı [*Müşteriler*] tablosundan, müşterilerin verdiği siparişler [*Siparişler*] tablosundan, siparişlerdeki ürünlerden kaçar adet verildiği [*Sipariş Ayrıntıları*] tablosundan ve Ürün isimleri ise [*Ürünler*] tablosundan elde edilebilir. Bu tablolar ilgili alanlardan birbirine bağlandığı takdirde hedef veri kümesine ulaşılabilir. Bu dört tablo aşağıda grafik olarak gösterildiği gibi **FROM** komutu sonrası birbirine bağlanmıştır. Ancak tablolara kısa takma ad tanımlanarak sorgunun daha kolay yazılması sağlanmıştır.



Tablo bağlanmasından sonra kısıtlama komutu yazılmalıdır. Sipariş tarihi bilgisi [*Siparişler*] tablosunda [*Sipariş Tarihi*] adıyla yer almaktadır. Bu alanın kısıtlanması için tarih türündeki bir alanın ay ve yıl bilgisini elde edebilecek **YEAR()** ve **MONTH()** işlevlerinin kullanılması gerekir. Satışların ürün ve müşteri bazında elde edilmesi istendiğinden **GROUP BY** alanında [*Şirket*] ve [*Ürün Adı*] alanına göre gruplanması gerekir. En son aşamada ise hangi alanlar sorgu sonunda görüntülenecek ya da hesaplanacak bu alanların **SELECT** kısmı sonrasında yazılması tamamlanır. Gruplanan Şirket ve ürün adı alanlarının seçmede kullanılması, oluşan verinin anlaşılması açısından önemlidir. Bir diğer önemli bilgi ise toplam kaçar adet siparişin alındığıdır. Bunun için de [*Sipariş Ayrıntıları*] tablosundaki [*Miktar*] bilgisinin gruplanan alanlara göre toplanması gerekir. **SUM()** komutu sayısal değerlerin toplanması için kullanılan bir özetleme işlevidir. Bu işlevin **SELECT** kısmına yazılması ile sorgu aşağıdaki gibi tamamlanmış olacaktır.

```
SELECT M.Şirket, U.[Ürün Adı], SUM (SA.Miktar) AS Toplam Miktar
FROM Siparişler AS S
  INNER JOIN [Sipariş Ayrıntıları] AS SA ON S.[Sipariş No]
    = SA.[Sipariş No]
  INNER JOIN Müşteriler AS M ON S.[Müşteri No] = M.No
  INNER JOIN Ürünler AS U ON SA.[Ürün No] = U.No
WHERE (YEAR(S.[Sipariş Tarihi]) = 2006) AND (MONTH(S.
[Sipariş Tarihi]) = 4)
GROUP BY M.Şirket, U.[Ürün Adı]
```

Sorgunun çalıştırılması sonucu Tablo 6.4'de verilen tablo elde edilecektir. Birçok tablonun bulunduğu ve karmaşık kısıtlamaların yer aldığı sorguları oluşturmak için kullanıcıların hatasız bir sorgu yazması sistematik bir yaklaşımı gerektirir.

Tablo 6.4
Örnek Uygulama İçin
Yazılan Sorgu Sonucu

Şirket	Ürün Adı	Toplam Miktar
Şirket F	Çikolata	10
Şirket H	Çikolatalı Bisküvi Karışımı	25
....
Şirket BB	Yengeç Eti	50
Şirket Z	Yengeç Eti	30
Şirket Z	Zeytin Yağı	25

SIRA SİZDE



Örnek Uygulama 6.1'de 2006 Nisan ve 2007 Ocak ayındaki siparişlere ilişkin bilgi istenseydi WHERE ifadesi nasıl yazılmalıydı?

Özetleme İşlevleri

Özetleme işlevleri gruplama işlemlerinde sayma, toplama, ortalama ve benzeri işlemleri yerine getirirler.

Gruplama sorgularında sorgulamak istenen veri kümesi **GROUP BY** ifadesinden sonraki alanlara göre gruplanırken diğer alanlar üzerinde de özetleme işlemleri gerçekleştirilir. Sayısal alanlar için sayma, toplama, ortalama gibi matematiksel işlemler, metin türündeki alanlar için ise sayma, en büyük ve en küçük değerlerin alınması gibi işlemler özetleme işlevleri tarafından gerçekleştirilir. SQL Server'da sık olarak kullanılan özetleme işlevleri ve açıklamaları Tablo 6.5'de verilmiştir.

Tablo 6.5
Özetleme İşlevleri ve
Anlamı

İşlev	Açıklaması
COUNT, COUNT_BIG	Bir gruptaki kayıt sayısını hesaplarlar. COUNT Int (tamsayı) veri tipi ile veri döndürürken, COUNT_BIG BigInt(büyük tamsayı) veri tipinde sonuç döndürür.
SUM	Sayısal veriler için kullanılır. NULL olan verileri dikkate almaz. Distinct komutu ile benzersiz değerlerin toplanması sağlanabilir.
AVG	Sayısal değerli alanların ortalamasını hesaplar.
MIN	Belirtilen alanın en küçük değerini getirir. NULL değerleri dikkate alınmaz. Metin değerler için de alfabetik sıraya göre en küçüğünü getirir.
MAX	Belirtilen alanın en büyük değerini getirir. NULL değerleri dikkate alınmaz. Metin değerler için de alfabetik sıraya göre en küçüğünü getirir.
STDEV	Belirtilen alan için standart sapma hesaplar.
STDEVP	Belirtilen alanın tüm değerleri için ana kütle standart sapmasını hesaplar.
VAR	Belirtilen alan için varyans hesaplar.
VARP	Belirtilen alanın tüm değerleri için ana kütle varyansını hesaplar.

Özetleme işlevleri işlem yapılan alandaki NULL olmayan, dolu olan alanlar için işlem yapmaktadır. İşlev adından sonra açılacak parantez içinde alan adından önce **DISTINCT** ifadesi konursa tekrarsız satırlar için işlem yapılmaktadır. Örneğin [Sipariş Ayrıntıları] tablosunda aşağıdaki iki sorgu çalıştırıldığında ilk sorgu 56 sayısını hesaplar. Bu ilgili tabloda [Ürün No] alanı boş olmayan 56 satır olduğu anlamına gelir. İkinci satır ise 24 sonucunu verir ki bu da ilgili tabloda 24 farklı [Ürün No] yer aldığı anlamına gelir. Tekrar eden verilerin bir kere sayılacağı anlamına gelir.

```
SELECT COUNT([Ürün No]) FROM [Sipariş Ayrıntıları]
SELECT COUNT(DISTINCT [Ürün No]) FROM [Sipariş Ayrıntıları]
```

DISTINCT ifadesi diğer özetleme işlevleri ile de kullanılabilir.

Northwind veritabanında sipariş edilen ürünler aylık temelde raporlanmak istenmektedir. Buna göre her ay gerçekleşen siparişlerin toplam sayısı ve ortalama satış miktarları, yine aylık temelde satışlardan elde edilen toplam ciro tutarı ve aylara göre ortalamalarını hesaplayan SQL komutlarını oluşturunuz.

ÖRNEK 6.2

Northwind örnek veritabanında sipariş bilgileri 2006 yılına ait verilerde oluşmuş olsa da sorgunun tasarımıyla yıl bilgisinin de grupta alanı olarak belirlenmesi daha ileride sorgunun doğru çalışmasını sağlayacaktır. Sorgunun hazırlanması için [Siparişler] tablosundaki [Sipariş Tarihi] ve [Sipariş Ayrıntıları] tablosundaki [Miktar] ve [Birim Fiyat] alanlarının bir veri kümesini oluşturmalı ve daha sonrasında yıl ve ay bilgisine göre grupta yapılmalıdır. Aşağıda bu amaçla oluşturulan sorgu incelendiğinde, sorgunun grupta satırında yıl ve ay bilgilerini elde eden fonksiyonlar kullanılarak yapıldığı görülmektedir. Sadece tarih bilgisine göre grupta yapılması durumunda, gün bilgilerini de içeren tarih veri türü veri kümesini günler bazında gruplayacak ve istenilen veri kümesi elde edilemeyecektir.

```
SELECT YEAR(S.[Sipariş Tarihi]) AS YIL,
MONTH(S.[Sipariş Tarihi]) AS AY,
SUM(SA.Miktar) AS Top_Miktar,
AVG(SA.Miktar) AS Ort_Miktar,
SUM(SA.[Birim Fiyat] * SA.Miktar) AS Top_Ciro,
AVG(SA.[Birim Fiyat] * SA.Miktar) AS Ort_Ciro
FROM Siparişler AS S INNER JOIN [Sipariş Ayrıntıları] AS SA
ON S.[Sipariş No] = SA.[Sipariş No]
GROUP BY YEAR(S.[Sipariş Tarihi]), MONTH(S.[Sipariş Tarihi])
```

Tarih alanından elde edilen Ay ve Yıl bilgisine göre grupta yapılan sorgunun seçme kısmında hesap yapılması istenen alanlara göre özetleme işlevleri kullanılmıştır. Toplama için SUM ve ortalama için AVG işlevi kullanılarak oluşan bu yeni alanlar için AS ifadesi ile birer takma ad verilmiştir. Ciro miktarının hesaplanmasında [Sipariş Ayrıntıları] tablosundaki [Birim Fiyat] ve [Miktar] çarpılarak ürünün satışı ile elde edilen tutarlar hesaplanmış ve toplanarak ve ortalaması hesaplanarak özetlenmiştir.

YIL	AY	Top_Miktar	Ort_Miktar	Top_Ciro	Ort_Ciro
2006	1	225	28	3836,00	479,50
2006	2	230	76	2241,50	747,1666
2006	3	1092	84	32609,25	2508,4038
2006	4	985	46	19355,25	921,6785
2006	5	95	23	1788,50	447,125
2006	6	315	35	8306,50	922,9444

Tablo 6.6
Örnek Uygulama 6.2'i
İçin Yazılan Sorgunun
Sonuç Tablosu

Tablo 6.5'de "AVG(SA.Miktar) AS Ort_Miktar" ifadesi ile elde edilen [Ort_Miktar] sütunu verisi incelendiğinde, verinin tamsayı değerlerinden oluştuğu görülmektedir. Toplam değer eleman sayısına bölünerek hesaplanan ortalamanın aslında tam sayı değil kesirli sayı olması beklenir. Bunun nedeni SQL Server'daki özetleme işlevlerinin sonuç olarak üzerinde hesaplama yapılan alanın türünde veri geri döndürmelidirler. "Tinyint", "smallint" ve "int" veri türleri int, bigint veri türü bigint, float ve real veri türleri float türünde veri geri döndürürler. Bu nedenle smallint veri türünde olan [Miktar] alanının orta-

Veri türünün dönüşümü için kullanılan **Cast** komutunun yazım kuralı **Cast(AlanAdı As VeriTürü)** şeklindedir.

lama hesabı tamsayı (int) veri türünde geri dönmüştür. Ancak bu değerin kesirli sayı olarak hesaplanması istenirse veri türlerinin dönüştürülmesinde kullanılan **Cast** ve **Convert** komutları kullanılmalıdır. Ort_Miktar alanının kesirli sayı olarak hesaplanması için **AVG(Cast(SA.Miktar as real))** şeklinde yazmak yeterli olacaktır.

DİKKAT



Cast ve Convert işlevleri Select sorgularında görüntülenecek değerlerin veri türünü değiştirmek için kullanılır. Tablolardaki alanların veri yapılarını kalıcı olarak değiştirmek için veri tanımlama dilindeki ALTER komutu kullanılır.

Özetlenen Değerlerin Sınırlanması

Gruplama sorgularında özetlenen değerlerin belirli koşullara göre sınırlanması için **HAVING** komutu kullanılır. **HAVING** komutunun mantıksal işleme sırası Select ifadesinden önce Gruplama işlevinden sonra yer almasıdır. Dolayısıyla gruplama ve özetleme işlemi tamamlandıktan sonra koşullara uymayan satırlar seçilmez. **WHERE** komutu ise gruplama işleminden önce gerçekleştiği **WHERE** ile belirtilen koşullar Özetleme işlemlerine dahil edilmezler. Örnek uygulama 6.2'de tanımlanan gruplama ve özetleme ile elde edilen sorunun aylık siparişin 50 nin üzerinde olan aylar için hesaplanması istensin. Bu durumda mevcut sorunun sonuna aşağıda da görüleceği gibi **HAVING SUM(SA.Miktar) > 50** satırının eklenmesi yeterli olacaktır.

```
SELECT YEAR(S.[Sipariş Tarihi]) AS YIL,
MONTH(S.[Sipariş Tarihi]) AS AY,
SUM(SA.Miktar) AS Top_Miktat,
AVG(Cast(SA.Miktar as real)) AS Ort_Miktar,
SUM(SA.[Birim Fiyat] * SA.Miktar) AS Top_Ciro,
AVG(SA.[Birim Fiyat] * SA.Miktar) AS Ort_Ciro
FROM Siparişler AS S INNER JOIN [Sipariş Ayrıntıları] AS SA
ON S.[Sipariş No] = SA.[Sipariş No]
GROUP BY YEAR(S.[Sipariş Tarihi]), MONTH(S.[Sipariş Tarihi])
HAVING SUM (SA.Miktar)>50
```

SQL İŞLEVLERİ

SQL sorgularında çeşitli hesaplama ve mantıksal işlemleri gerçekleştirebilmek için işlevler bulunmaktadır. Microsoft SQL Server Transact-SQL dili farklı işlemler için farklı türde işlevler sunmaktadır. Bu ünite Transact-SQL dilinde hazır olarak sunulan ve seçme sorgularında kullanılabilecek işlevlere yer verilecektir. Kullanıcıların ihtiyaçlarına yönelik işlevleri Transact-SQL programlama dili kullanarak oluşturabilmektedir. Ancak bu ünite ve bu kitapta programlamayla ilgili ayrıntıya yer verilmemiştir.

Birçok programlama dilinde farklı yazım şekilleriyle benzer görevleri yerine getiren işlevler mevcuttur. Bu işlevlerin çalışma prensibinin kavranması ve ihtiyaç olduğunda ilgili yardım dosyalarına ulaşarak bu işlevleri etkin kullanmak bir veritabanı kullanıcısı ya da programcısı için oldukça önemlidir.

Mantıksal İşlevler

Mantıksal işlevler **SELECT** sorgularında alanların değerlerini belirli koşullara göre farklı değerlere dönüştürürler. **CHOOSE**, **IIF**, **CASE** işlevlerinin yazım kuralları ve örneklerine aşağıda yer verilmiştir.

CHOOSE: İndeks değeri içeren bir değerin tanımlanan sıralı listede eşleştirerek karşı gelen elemanı görüntüler. **CHOOSE** (İndeks, 'Seç1', 'Seç2',, 'SeçN') olarak yazılır. Northwind veritabanında [Siparişler] tablosundaki [Durum No] alanındaki değer için aşağıdaki sorgu yazılarak indekse karşı gelen metin listelenebilir. İndeks değeri sıfırdan başladığından **CHOOSE** komutunda 1 eklenerek yer almıştır.

```
SELECT [Durum No],
CHOOSE([Durum No]+1, 'Yeni', 'Faturalandı', 'Sevk Edildi', 'Kapatıldı')
AS Durum
FROM Siparişler
```

IIF: Mantıksal bir karşılaştırma sonucu doğru ve yanlış durumları için iki ayrı değer görüntüler. Bu mantıksal işlev **IIF**(Mantıksal_ifade, Doğru_değer, Yanlış_Değer) şeklinde oluşturulur. Northwind veritabanında [Sipariş Ayrıntıları] tablosunda [Satınalma Siparişi No] alanında sipariş alınan ürün için sağlayıcılardan gerçekleştirilen Satınalma numarası yer almaktadır. Bu alandaki değer NULL (boş) ise stoktan karşılandığı anlamına gelmektedir. Bu alanın kullanıcıya daha anlaşılır şekilde sunmak için IIF komutundan yararlanılabilir.

```
SELECT IIF([Satınalma Siparişi No] is NULL, 'Stoktan karşılandı',
cast([Satınalma Siparişi No] as nvarchar(10))+ 'nolu sipariş
ile temin edildi')
As Tedarikdurumu
FROM [Sipariş Ayrıntıları]
```

Seçme sorgusu [Satınalma Siparişi No] alanı boş ise 'stoktan karşılandı', boş değilse ilgili sayısal değeri metine dönüştürerek sipariştan temin edildiğini belirten bir metin oluşturmaktadır. Yukarıdaki sorgunun çalıştırılması durumunda aşağıdakine benzer bir sonuç listelenecektir.

```
Tedarikdurumu
Stoktan karşılandı
Stoktan karşılandı
96 nolu sipariş ile temin edildi
97 nolu sipariş ile temin edildi
```

CASE: **CASE** komutu mantıksal işlevlerden farklı olarak birden çok koşulun tanımlanmasına olanak sağlar. **CASE** komutunun iki farklı yazım kuralı Tablo 6.7'de yazım kuralı ve örnek sorgularla açıklanmıştır.

Tablo 6.7
CASE Komutu Yazılım
Kuralları ve Örnek
Sorguları

Kullanım 1	Kullanım 2																				
<p>Yazım Kuralı: CASE ifade WHEN Değer1 THEN Değer1 WHEN Değer2 THEN Değer2 ELSE 'diğer' END</p>	<p>Yazım Kuralı: CASE WHEN Mantıksal_ifade THEN Değer1 WHEN Mantıksal_ifade THEN Değer2 ELSE 'diğer,' END</p>																				
<p>Örnek Sorgu : SELECT [Durum No], CASE[Durum No] WHEN 0 Then 'Yeni' WHEN 1 Then 'Faturalandı' WHEN 3 Then 'Sevk Edildi' WHEN 9 Then 'Kapatıldı' Else 'Hata' END AS Durum FROM Siparişler</p>	<p>Örnek Sorgu : SELECT Miktar*[Birim Fiyat] AS Ciro, CASE WHEN Miktar*[Birim Fiyat]<1000 THEN 'Az' WHEN Miktar*[Birim Fiyat]<10000 THEN 'Orta' WHEN Miktar*[Birim Fiyat]<100000 THEN 'Yüksek' ELSE 'Çok Yüksek' END AS CiroDurum FROM [Sipariş Ayrıntıları]</p>																				
<table border="1"> <thead> <tr> <th colspan="2">Sonuç:</th></tr> <tr> <th>DurumNo</th><th>Durum</th></tr> </thead> <tbody> <tr> <td>0</td><td>Yeni</td></tr> <tr> <td>1</td><td>Faturalandı</td></tr> <tr> <td>9</td><td>Kapatıldı</td></tr> </tbody> </table>	Sonuç:		DurumNo	Durum	0	Yeni	1	Faturalandı	9	Kapatıldı	<table border="1"> <thead> <tr> <th colspan="2">Sonuç:</th></tr> <tr> <th>Ciro</th><th>CiroDurum</th></tr> </thead> <tbody> <tr> <td>1400,00</td><td>Orta</td></tr> <tr> <td>105,00</td><td>Az</td></tr> <tr> <td>13800,00</td><td>Yüksek</td></tr> </tbody> </table>	Sonuç:		Ciro	CiroDurum	1400,00	Orta	105,00	Az	13800,00	Yüksek
Sonuç:																					
DurumNo	Durum																				
0	Yeni																				
1	Faturalandı																				
9	Kapatıldı																				
Sonuç:																					
Ciro	CiroDurum																				
1400,00	Orta																				
105,00	Az																				
13800,00	Yüksek																				

Tablo incelendiğinde birinci kullanımda tek bir değer aldığı değere göre istenilen sonucun yazdırılması sağlanabilir. İkinci kullanım ise daha esnektir ve her koşul ifadesine farklı bir koşul ifadesi tanımlanabilmektedir. Nortwind veritabanında yer alan tablolarda uygulanan örnek sorgularda istenilen şartların sağlanması durumunda kullanıcıya metin görüntüleyen örneklere yer verilmiştir.

SIRA SİZDE



Case komutunun sıralama işlemlerinde nasıl kullanıldığını araştırınız.

Sayısal İşlevler

Veritabanı tabloları üzerinde gerçekleştirilecek sorgulamalarda gerekli olabilecek hazır matematiksel işlevler bulunmaktadır. Bu işlevler veri tablosunun bir alanı için kullanılabileceği gibi sabit bir değer verilerek de hesaplama yaptırılabilir. Ancak unutulmaması gereken bir konu seçme sorgusunun alan listesinde yar alan bir işlem ya da değer seçimin tüm satırları için tekrar ettiğidir. Tablo 6.8'de seçme sorgularında kullanılabilecek işlevler yer almaktadır.

İşlev	Açıklaması	Örnek
ABS	Belirtilen değerin mutlak değerini veren matematik işlevidir.	ABS (-5) Sonuç : 5
SIGN	Sayısal değerin pozitif, negatif veya işaretsiz olduğunu döndürür	SIGN (-550) Sonuç : -1
ASIN, ACOS, ATAN, ATN2	Belirtilen trigonometrik değerlerin radyan cinsinden açı değerini hesaplar.	4* Atan (1) Sonuç : 3,14
SIN, COS, TAN, COT	Radyan cinsinden belirtilen değerlerin trigonometrik değerlerini hesaplar.	Tan (3.14/4) Sonuç : 1
RADIANS	Derece cinsinden verilen açı değerini radyan cinsine çevirir.	RADIANS (180.0) Sonuç : 3.14
DEGREES	Radyan cinsinden verilen açı değerini derece cinsine çevirir.	Degrees (PI()) Sonuç : 180
PI	Pi (π) katsayısını verir.	PI () Sonuç : 3,14159265358979
CEILING	Kesirli sayıları bir üst tamsayıya yuvarlar.	CEILING (-5.5) Sonuç : -5
FLOOR	Kesirli sayıları bir üst tamsayıya yuvarlar.	FLOOR (-5.5) Sonuç : -6
ROUND	Belirtilen sayı ya da alandaki değeri istenilen basamağa yuvarlar. İlk değer yuvarlanacak değer, ikini değer yuvarlanacak basamak.	ROUND (56.55, -1) Sonuç : 60.00
EXP, LOG, LOG10, POWER	Üssel ve logaritmik işlem yapan işlevlerdir.	POWER (2, 5) Sonuç : 32
RAND	0 ile 1 aralığında düzgün dağılmış rassal sayı üretir.	RAND () Sonuç : 0,5299
SQRT	Belirtilen değerin karekökünü hesaplar.	SQRT (25) Sonuç : 5
SQUARE	Belirtilen değerin karesini hesaplar.	SQRT (25) Sonuç : 625

Tablo 6.8
SQL Server
Matematiksel İşlevleri

Metin İşlevleri

Metin türündeki alanların ya da değerlerin işlenmesine yönelik çok sayıda metin işlevi bulunmaktadır. Tablo 6.9'da sık kullanılan metin işlevleri, gerçekleştirdiği işlem ve örnek kullanımları verilmiştir. Örnekler bir sorgunun tamamını içermeyip sadece işlev ifadeleri için örneklenmiştir. Bu ifadeler bir seçme sorgusunun farklı bölümlerinde kullanılabilir.

SQL Server komut ve işlevlerine ulaşmak için internet kaynaklarına başvurulabilir. İnternet tarayıcısında "String Functions SQL MSDN" arama kelimelerinin kullanılması ile tüm işlev listesine ulaşılabilir.



İNTERNET

Tablo 6.9
SQL Server
Matematiksel İşlevleri

İşlev	Açıklaması	Örnek
SUBSTRING	Metin türündeki bir değerin içerisinde belirli bir bölümün alınmasını sağlayan işlevdir.	SUBSTRING ('YBS206U' ,4,3) Sonuç: 206
PATINDEX	Bir metin içerisinde belirli bir metin ifadesinin arayarak ilk bulunduğu konum bilgisini döndürür.	PATINDEX ('%YBS%', Belge) Sonuç: 50
LEN	Metindeki karakter sayısını döndürür	LEN ('YBS206U') Sonuç: 7
LEFT, RIGHT	Metin türündeki bir alan ya da değerin soldan(LEFT) veya sağdan(RIGHT) istenilen sayıda karakterin seçilmesini sağlar.	LEFT 'YBS206U' ,3) Sonuç: 'YBS'
REPLACE	Bir metin içerisinde belirli bir metni bularak o istenilen metin ile değiştirir.	REPLACE ('YBS206U' , '206' , '304') Sonuç: 'YBS304U'
CONCAT	Bir veya daha fazla metin değeri birleştirir.	CONCAT ('Veri' , 'tabanı' , 'I') Sonuç: 'Veritabanı I'
REPLICATE	Bir metnin değişkenin istenildiği kadar tekrarlanması için kullanılır.	REPLICATE ('0' ,5) Sonuç: '00000'
SPACE	İstenilen sayıda boşluk karakteri üretir.	'A' + SPACE (3) + 'B' Sonuç: 'A A'
NCHAR, CHAR	NCHAR Unicode karakter tablosunun, CHAR ise ASCII kod tablosunun belirtilen rakama karşı gelen karakterini getirir.	NCHAR (65) Sonuç: 'A'
UNICODE, ASCII	UNICODE bir karakterin Unicode, ASCII ise ASCII kod tablosundaki kod karşılığını getirir.	ASCII ('A') Sonuç: 65
STR	Sayısal bir değeri istenilen hassasiyette metne çevirir.	STR (123.45, 6, 1); Sonuç: 123.5

Tarih İşlevleri

Farklı kültürde aynı takvim kullanılsa da tarih bilgilerinin yazılması konusunda farklılıklar bulunmaktadır. Bu nedenle veritabanı yönetim sistemlerinde tarihin gösterilmesi, eklenmesi ile ilgili problemleri ortadan kaldırmak için çeşitli işlev ve formatlar bulunmaktadır. Bilgisayarlarda tarih ve saat bilgileri aslında bir sayı olarak saklanır. Bu sayının tamsayı kısmı gün, ondalıklı kısmı ise 24 saat 1 tamsayıya denk gelecek şekilde depolanmaktadır. SQL Server v veritabanı yönetim sistemi yazılımında Tarih ve saatle ilgili veri türleri Tablo 6.10'da verilmiştir. Tarih ve saat veri türleri ile ilgili temel işlevler aşağıda yer almaktadır.

Tablo 6.10
Transact-SQL Tarih
ve Saat Türleri
(Microsoft MSDN'den
Düzenlenmiştir.)

Veri türü	Biçim	Aralık	Hassasiyet	Depolama (byte)
time	hh:mm:ss [nnnnnnnn]	00:00:00.0000000 23:59:59.9999999	100 nanosaniye	3 to 5
date	YYYY-MM-DD	0001-01-01 9999-12-31	1 gün	3
small datetime	YYYY-MM-DD hh:mm:ss	1900-01-01 2079-06-06	1 dakika	4
datetime	YYYY-MM-DD hh:mm:ss[.nnn]	1753-01-01 9999-12-31	0.00333 saniye	8
datetime2	YYYY-MM-DD hh:mm:ss [nnnnnnnn]	0001-01-01 00:00:00.0000000 9999-12-31 23:59:59.9999999	100 nanosaniye	6-8

DATENAME: Tarih veri türündeki bir değerin bir bölümünü isim olarak geri döndürür. “**DATENAME**(DatePart, Tarih_verisi)” şeklinde yazılan işlev verilen tarih verisinin ilgili kısmının karşılığını İngilizce olarak döndürür. **SELECT DATENAME**(month, '2016/06/19') ya da **SELECT DATENAME**(m, '2016/06/19') şeklinde yazılan komut “June” metin verisini geri döndürür.

DATEPART: Tarih veri türündeki bir değerin istenen bölümünün elde edilmesini sağlar. İşlev “**DATEPART**(DatePart, Tarih_verisi)” şeklinde yazılır. Örneğin bir tarih bilgisinin saniye değerini elde etmek için **SELECT DATENAME**(s, '2016/06/19 12:10:30.123') komutu yazılarak 30 sayısının geri dönmesi sağlanabilir.

DAY, MONTH, YEAR: Ünitede daha önceki örneklerde de kullanılan bu üç işlev sırasıyla bir tarih verisinin yıl, ay ve gün bilgisini sayısal olarak geri döndürür.

DATEFROMPARTS: Yıl, ay ve gün bilgilerinin sayısal olarak girildiği ve ilgili değerlerden tarih veri türünde değişken döndüren bir işlevdir. **SELECT DATEFROMPARTS**(2016, 6, 19) komutunun çalıştırılması ile “2016-06-19” şeklinde **Date** türünde veri döndürür. Bu işlev tarih verilerinin hatalı girilmesini engellemek için istemci yazılımlar tarafından kullanılabilir.

DATEDIFF: İki tarih türündeki verinin arasındaki farkı istenilen tarih birimi olarak hesaplayan bir işlevdir. “**DATEDIFF**(DatePart, BaşlangıçTar,BitisTar)” olarak yazılır. Belirtilen iki tarih arasındaki farkı saat olarak hesaplanması istenen “**SELECT DATEDIFF**(hh, '2015/06/19 12:10', '2016/06/19 00:10')” bu komut 8772 sonucunu verecektir. İki tarih arasında 8772 saat fark olduğu hesaplanmıştır. Bu komut **Int** türünde veri döndüren bir işlevdir. Ancak hesaplamalarda eğer **Int** veri türünden daha büyük bir dönüş olacaksa **DATEDIFF_BIG** işlevi aynı parametrelerle kullanılabilir. Bu işlev **Bigint** veri türünde bir sayı döndürür.

DATEADD: Bir tarih türündeki veriye istenilen tarih birimi olarak ekleme işlemi gerçekleştiren işlevdir. “**DATEADD**(DatePart, Sayı, Tarihverisi)” yazım kuralı ile yazılan bu komutu bir önceki **DATEDIFF** işlevi için verilen örneğin sağlaması için yazdığımızda (**SELECT DATEADD**(hh, 8772, '2015/06/19 12:10')) “2016-06-19 00:10:00.000” tarih bilgisini geri döndürdüğü görülecektir.

ISDATE: Bir metin türündeki verinin içeriğinin tarih bilgisi olup olmadığını kontrol ederek pozitif durumda 1, negatif durumda 0 değerini döndüren işlevdir. “**SELECT ISDATE**('2015/16/19 12:10')” işlevi 0 değeri döndürecektir.

SQL tarih işlevlerinde bir tarih verisinin ilgili bölümünü aşağıdaki tabloda yer alan Datepart sütunundaki ifadeler ya da kısaltma sütunundaki adlar ile ifade edilir.

Bölümler	DatePart	Kısaltma
Yıl	year	yy, yyyy
Çeyrek	quarter	qq, q
Ay	month	mm, m
Yılın günü	dayofyear	dy, y
Gün	day	dd, d
Hafta	week	wk, ww
Saat	hour	hh
Dakika	minute	mi, n
Saniye	second	ss, s
Milisaniye	millisecond	ms
Milisaniye	microsecond	mcs
Nanosaniye	nanosecond	ns

ALT SORGULAR

Konusu seçme sorguları olan bu ünitenin birçok kısmında veri kümesi kavramı kullanıldı. Veritabanı yönetim sisteminde depolanan verilerin kullanıcıların taleplerine göre hazırlanmalarında oldukça karmaşık seçme sorgularını oluşturmak gerekebilmektedir. Bu nedenle oluşturulacak seçme sorguları birden fazla veri kümesi hâlinde hazırlanarak birbirleriyle ilişkilendirmek, sınırlamak ya da birleştirmek suretiyle sistematik bir şekilde yapılandırılabilir. Her bir alt veri kümesine seçme sorgularında alt sorgu ismi verilmektedir. Alt sorgular karmaşık yapıdaki seçme sorgularının hazırlanmasında SQL komutlarının sağladığı esnekliklerden biridir.

Alt sorgular başka bir sorgunun herhangi bir kısmında (seçim, tablo, sınırlayıcı) parantez içerisinde eklenen seçme sorgularıdır. Alt sorgular bir üst sorguya bir değer ya da bir tablo çıktısı verebilirler. Alt sorgu alanları ile üst sorgu alanlarının birbiri ile ilişkilendirmek yoluyla da seçme sorguları hazırlanabilmektedir.

Alt sorgu (subquery) bir seçme sorgusunun bir parçasını oluşturan ve parantez içinde yazılmış sql seçme sorgusuna denilir.

Sorgunun veri kaynağı olarak bir alt sorgunun kullanılması durumunda aşağıdaki yapıda sorgu oluşturulur. Altı çizili olarak gösterilmiş komutlar Alt sorgu olarak adlandırılır.

```
SELECT Ortalama, Alan1
FROM
(Select Alan1, AVG(Alan2) As Ortalama FROM Tablo5 GROUP BY Alan1)
AS Tablo2
```

Yukarıdaki seçme sorgusu bir tablodan değil parantez içerisinde yer alan başka bir veri kümesinin (seçme sorgusunun) ürettiği listeyi kullanmaktadır.

Alt Sorguların Tablo Olarak Kullanılması

Ulaşılmak istenilen bazı veri kümelerinin SQL komutları ile hazırlanmasında birden fazla aşama gerekebilmektedir. Alt sorguların tablo olarak kullanıldığı bir örnek uygulamayı adım adım gerçekleştirerek istenilen veri kümesini elde edelim.

ÖRNEK 6.3

Nortwind veritabanında işletmenin ürünlerine ait mevcut stok miktarlarını hesaplayan sorguyu oluşturunuz.

Northwind veritabanında ürünlere ait stok bilgileri [Stok Hareketleri] tablosunda yer almaktadır. Bu tablodaki [Hareket Türü] alanının değeri 1 ise ürün gelişi 2 ise ürün satışı anlamına gelmektedir. Stok miktarlarının hesaplanabilmesi için gelen ürünlerin pozitif gidenlerin ise negatif değerlerde gösterilmeleri ve daha sonra her ürün için bu değerlerin toplanması gerekmektedir. Bu işlem için iki adımlı bir yapı oluşturmak gerekecektir. İlk adımda [Hareket Türü] değeri 1 olanlar için bir sorgu ve [Hareket Türü] değeri 2 olanlar için [Miktar] alanının değerini -1 ile çarpıldığı ikinci bir sorgu. Bu iki sorgunun alt alta birleştirilmesi için UNION ALL komutu kullanılarak bir veri kümesi hazırlansın. Bu veri kümesini hazırlayacak aşağıdaki sorgu komutları her bir stok hareketi için [Ürün No], [Miktar] verisini listeleyecek ancak ürün çıkışı için miktar değerlerini negatif olarak belirleyecektir.

```
SELECT [Ürün No], Miktar
FROM [Stok Hareketleri]
WHERE ([Hareket Türü] = 1)
UNION ALL
SELECT [Ürün No], Miktar *-1
FROM [Stok Hareketleri]
WHERE ([Hareket Türü] = 2)
```

Bu aşamadan sonra elde edilen veri kümesini [Ürün No] alanına göre gruplamak ve [Miktar] verisini toplamak her ürün için stok miktarını hesaplamak anlamına gelecektir. Bu amaçla yukarıda elde edilen veri kümesinin grupta yapacak bir sorgunun tablosu olarak kullanmak yerinde olacaktır. Alt sorgu olarak yukarıdaki seçme sorgusunun bir başka sorguda parantez içinde kullanılması sağlanabilir. [Stok] adıyla yeni bir tablo gibi yer alan alt sorgunun alanları bir üst sorguda kullanılabilir. Aşağıdaki sorgu incelendiğinde [Stok.Miktar] alanının toplandığı, ürünlere göre grupta yapıldığı görülmektedir. Sorguya ürünler tablosunun eklenmesinin sebebi ise ürün adının da sonuçta görüntülenmek istenmesidir. Aşağıdaki sorgunun çalıştırılması sonucu oluşan 28 kayıtlık listenin ilk 5 satırı Tablo 6.11'de verilmiştir.

```

SELECT Ürünler.[Ürün Adı], Stok.[Ürün No], SUM(Stok.Miktar) AS
Stok_Miktarı
FROM (SELECT [Ürün No], Miktar
      FROM [Stok Hareketleri] WHERE ([Hareket Türü] = 1)
      UNION ALL
      SELECT [Ürün No], Miktar * - 1 AS Expr1
      FROM [Stok Hareketleri] WHERE ([Hareket Türü] = 2)) AS Stok
INNER JOIN Ürünler ON Stok.[Ürün No] = Ürünler.No
GROUP BY Stok.[Ürün No], Ürünler.[Ürün Adı]
ORDER BY Stok_Miktarı DESC

```

Ürün Adı	Ürün No	Stok_Miktarı
Kahve	43	325
Yeşil Çay	81	125
Mantı	56	120
Ravyoli	57	80
Domates Sosu	66	80

Tablo 6.11
Örnek Uygulama
6.3 İçin Oluşturulan
Sorgunun Sonuç
Listesinin İlk Beş Satırı

Alt sorguların içinde de alt sorgu kullanılabilir. Hedef veri kümesine ulaşmak için çok sayıda iç içe sorgu kullanılabilir ancak daha sonra bu sorguların düzenlenmesinde zorluk ortaya çıkabilmektedir. Bu nedenle çok karmaşık sorgular, birbirini izleyen görünüm(ler)(view) olarak tasarlanabilmektedir.

Alt Sorguların Kısıtlayıcı Olarak Kullanılması

Alt sorgular tablo olarak kullanılabilir gibi sınırlayıcı olarak da kullanılabilir. **WHERE** komutu ile bir alandaki değerler başka bir sorgunun içerdiği değerlerle sınırlandırılabilir.

Northwind veritabanında “İçecekler” kategorisindeki ürünlerden sipariş yapmamış müşteri listesini oluşturunuz.

ÖRNEK 6.4

Muhtemelen bu kitap ile ilk kez seçme sorguları ile tanışan okurlar örnek uygulamadaki sorunun olumlu olması durumunda kolayca ilgili sorguyu yazabileceklerdi. Ancak dikkat edilirse bir ürün gurubundan alışveriş yapmayan müşteri listesi isteniyor. Tabloları birbirine bağlayarak ulaşabileceğimiz bir veri kümesi olmadığı açıktır. Bu durumlarda istenilen veri kümesinin değilinden (tam tersinden) hareket edilmelidir. Öncelikle bu kategoride alışveriş yapan müşterileri elde eden bir sorgu oluşturulsun. İlgili üç tablo birbirine bağlanarak **WHERE** (U.Kategori = 'İçecekler') ifadesi ile ilgili müşterilerle sınırlandırılmış, zorunlu olmamakla beraber [Müşteri No] alanına göre gruplanmıştır. Gruplama gereksiz yere müşteri numaralarının tekrarını engellemek için uygulanmıştır. Aşağıdaki sorgu çalıştırıldığında 9 adet müşteri numarasını listeleyecektir.

```

SELECT S.[Müşteri No] AS Müsteriler
FROM Siparişler AS S
INNER JOIN [Sipariş Ayrıntıları] AS SA
ON S.[Sipariş No] = SA.[Sipariş No]
INNER JOIN Ürünler AS U ON SA.[Ürün No] = U.No
WHERE (U.Kategori = 'İçecekler')
GROUP BY S.[Müşteri No]

```

Müşteri listesinden çıkarılacak kümeyi bulduktan sonra **IN** komutunu **NOT** ifadesi ile uygulayarak hedeflenen veri kümesine erişilebilir. Aşağıdaki sorguda dikkat edilmesi gereken **WHERE** komutu sonrasında *[Müşteriler]* tablosundaki *[No]* alanının nasıl sınırlandırıldığıdır.

```
SELECT [No], Şirket
FROM Müşteriler
WHERE NOT ([No] IN
(SELECT S.[Müşteri No] AS Müsteriler
FROM Siparişler AS S INNER JOIN [Sipariş Ayrıntıları] AS SA
ON S.[Sipariş No] = SA.[Sipariş No]
INNER JOIN Ürünler AS U ON SA.[Ürün No] = U.No
WHERE (U.Kategori = 'İçecekler'))
GROUP BY S.[Müşteri No])
```

Sorgunun çalıştırılması durumunda içecekler kategorisinde yer alan ürünlerden hiçbiri alınmış 20 müşteri listelenecektir.

DİKKAT



IN ile yapılan sınırlamalarda alt sorgunun sadece ilgili alanı kısıtlamak için veri türünde tek bir alan seçmelidir. Aksi durumda üst sorgu ilgili alanı hangi değerin içinde arayacağını bilemeyeceğinden hata mesajı oluşturacaktır. Böyle bir hatada kullanıcı “Only one expression can be specified in the select list when the subquery is not introduced with EXISTS.” ifadesi ile uyarılacaktır.

Alt sorgularla yapılan sınırlamalarda tek bir değer üreten alt sorgular da kullanılabilir.

SIRA SİZDE



Bir cümle ile “SELECT [No] FROM [Sipariş Ayrıntıları] WHERE Miktar > (SELECT AVG (Miktar) AS Ort FROM [Sipariş Ayrıntıları])” sorgunun elde ettiği veri kümesini tanımlayınız.

Alt Sorgu Üst Sorguların İlişkilendirilmesi

Alt sorgular bir üst sorgunun alanları ile sınırlanabilmektedir. Biraz karmaşık bir durum gibi görünse de veri kümesi mantığı ile değerlendirildiğinde bu yöntemle istenilen veri kümelerine etkili bir şekilde ulaşılabilirdiği görülecektir.

ÖRNEK 6.5

Müşteriler tablosunda hiç sipariş vermemiş müşteri bulan sorguyu yazınız.

Bu sorgunun bir önceki konuda örnek uygulama 6.4’de uygulanan yöntemle istenilen veri kümesi bulunabilir. Diğer bir yöntem ise **EXISTS** komutudur. **EXISTS** komutu sorguya eklenen alt sorgu ile verilerin ilişkilendirilmesini sağlar. Üst sorguda *[Müşteriler]* tablosuna “M”, Alt sorguda ise *[Siparişler]* tablosuna *[Altsorgu]* adı verilmiştir. Alt sorgunun **WHERE** ifadesinden sonra yer alan “AltSorgu.[Müşteri No]=M.[No]” ifadesi incelendiğinde üst sorgu ile alt sorgunun nasıl bağlantıda olduğu anlaşılabilir. Alt sorguda yer alan müşterilerin üst listeden çıkarılması için **EXISTS** ifadesinden önce **NOT** operatörünün kullanılması önemlidir.

```
SELECT Şirket
FROM Müşteriler AS M
WHERE NOT EXISTS
(SELECT * FROM Siparişler AS AltSorgu
WHERE AltSorgu.[Müşteri No] = M. [No])
```

EXISTS ile yapılan sınırlamalarda alt sorgunun tüm alanlarının “*” sembolü ile seçildiğine dikkat edilmelidir. “*” ile tüm alanları seçilmeyen bir alt sorgu EXISTS ile kullanılamaz.

**DİKKAT**

Alt sorguların bir üst sorgunun tablosu, kısıtlayıcısı olabileceği gibi herhangi bir değer olarak da üst sorgu tarafından kullanılabilir. Bu kullanıma örnek olması açısından aşağıdaki sorguyu dikkatle inceleyiniz.

```
SELECT [Ürün No], Miktar,  
Miktar-(Select AVG(Miktar) FROM [Sipariş Ayrıntıları]  
WHERE [Ürün No]=a.[Ürün No] ) AS Fark  
FROM [Northwind].[dbo].[Sipariş Ayrıntıları] as a
```

Komutlar incelendiğinde seçme alanında bir alt sorgunun yer aldığı görülmektedir. Bu sorgu [Sipariş Ayrıntıları] tablosunun her satırı için satışı yapılan ürünün, o ürün için yapılan tüm satışların ortalama miktardan ne kadar farklı olduğu hesaplanmaktadır. Sorguların satır bazlı oluşturulduğu düşünülürse her bir satıra gelindiğinde alt sorgu ilgili satırın ürün numarası ile bir ortalama hesabı yapıp geri dönecek ve aynı satırın miktar değerinden çıkaracaktır.

Bir sorgunun seçim kısmında yer alan alt sorgular ciddi performans kaybına neden olabilmektedir. Her satır için çalıştırılan sorgunun gerçekleştirdiği işlem, üst sorgu ile ilişkili olması ve sorgu sonucu oluşacak satır sayısı oluşan performans kaybı ile doğru-dan ilişkilidir.

Özet



Seçme sorgularının yazım kuralını tanımlamak

Veri sorgulama dili veritabanı yönetim sistemlerinin en temel işlevlerinden biridir. Depolanan verilerin ihtiyacı olan kişiye ihtiyaç duyduğu ayrıntı ve şekilde ulaştırılması için kullanılan veri sorgulama dilinin veri seçimi ile ilgili komutu **SELECT** ifadesidir. Veri tablolarındaki tabloları birbirleriyle ilişkilendirilerek, birleştirilerek ya da sınırlandırılarak sorgulanması ya da verilerin özetlemesi ve gruplaması ile istenilen veri kümeleri hazırlanabilmektedir. **SELECT** komutunun yazım ifadesi aşağıdaki gibidir.

```
SELECT alan_listesi [INTO yeni_tablo]
[ FROM tablo_kaynağı ]
[ WHERE seçme_koşulları ]
[ GROUP BY gruplama ifadeleri ]
[ HAVING Gruplanan_alan_kısıtlamaları ]
[ ORDER BY sıralama düzeni ASC DESC ]
```

Select komutunu ile veri kümelerini hazırlamada seçme sorgusunun mantıksal işleme sırasının bilinmesi oldukça önemlidir. MS SQL Server yazılımında seçme sorgularında uygulanan işlemler sırasıyla FROM, ON, JOIN, WHERE, GROUP BY, WITH CUBE or WITH ROLLUP, HAVING, SELECT, DISTINCT, ORDER BY, TOP adımlarından oluşur.



Tabloların birbirine bağlanma türlerini sıralamak

Tabloları bir sorgu içerisinde birbiri ile bağlamada **JOIN** komutu kullanılır. **JOIN** komutu ile sadece iki değil çok sayıda tablo birbirine bağlanabilmektedir. Ayrıca farklı bağlanma türleri ile kullanıcıların istenilen veri kümelerini elde etmelerine olanak sağlanır. Tabloları birbirine bağlarken her iki tablodaki ortak alanları birbirine eşleyerek veri kümesini oluşturan **INNER JOIN**, bir tablonun ya da tüm tabloların değerlerinin hepsini kapsayan **OUTER JOIN** ve tablolara arası eşleştirme yapmadan çaprazlama yapan **CROSS JOIN** bağlantı türleri uygulanabilmektedir. Tabloları birbirine eklemek, kesişim ya da birleşim kümelerini elde etmek için **INTERSECT**, **EXCEPT**, **UNION**, **UNION ALL** komutları kullanılmaktadır.



Gruplama ve özetleme işlemlerini tanımlamak

Gruplama bir veri kümesinde belirlenen alanların içerdiği verinin tekrarsız hâle getirerek özetlenmesidir. Bu işlem sırasında gruplamanın yapılacağı alanlar sorguda **GROUP BY** ifadesinden sonra yazılır. Gruplama alanı dışındaki alanlar ise bir özetleme işlevi ile tek bir değer hâline getirilirler. Özetleme işlevleri olarak **COUNT** (Sayma), **SUM** (Toplama), **AVG** (Ortalama), **MIN** (En küçük değer), **MAX** (En büyük değer), **STDEV** (Standart sapma), **VAR** (Varyans) ifadeleri kullanılabilmektedir. Özetleme işlevleri ile elde edilen değerlerin sınırlandırılması için **HAVING** ifadesi kullanılır. Bu komut örneğin Aylık satış ortalamasının belirli değer üstünde ya da altında olan ürünlerin listelenmesi sağlanabilmektedir.



SQL işlevlerini sıralamak

SQL komutlarında belirli işlemleri hızlı olarak yerine getirebilmek için kullanıma sunulmuş çok sayıda işlev bulunmaktadır. Seçme sorgularında sıkça kullanılan mantıksal, sayısal, metin ve tarih işlevleri bu bölümde ele alınmıştır. Mantıksal işlevler seçme sorgularında bir alandaki değerin belirli şartlara uyması durumunda yerine açıklayıcı bir ifadenin görüntülenmesini sağlayabilmektedir. **CHOOSE**, **IF** ve **CASE** işlevleri mantıksal işlevler olarak kullanılmaktadır. Sayısal ve metin işlevleri genellikle diğer programlama dillerinde olduğu gibi matematiksel ve metin işleme işlevlerini içerirler. Tarih bilgilerine yönelik işlevler de bir veritabanı yönetim sisteminin sık kullanılan işlevleridir.



Alt Sorgu yapılarını tanımlamak

Veritabanı yönetim sisteminde depolanan verilerin kullanıcıların taleplerine göre hazırlanmalarında oldukça karmaşık seçme sorgularını oluşturmak gerekebilmektedir. Bu nedenle oluşturulacak seçme sorguları birden fazla veri kümesi hâlinde hazırlanarak birbirleriyle ilişkilendirmek, sınırlamak ya da birleştirmek suretiyle sistematik bir şekilde yapılandırılabilir. Her bir alt veri kümesine seçme sorgularında alt sorgu ismi verilmektedir. Alt sorgular karmaşık yapıdaki seçme sorgularının hazırlanmasında SQL komutlarının sağladığı esnekliklerden biridir. Seçme sorgularında alt sorgular tablo, alan, ya da bir kısıt olarak yer alabilir. Kısıt olarak bir alt sorgunun kullanılmasında **IN** ve **EXISTS** sınırlama yapıları kullanılır. Ayrıca alt sorgu ve üst sorgu alanlarının ilişkilendirilerek veri kümelerinin sınırlandırılması mümkün olabilmektedir.

Kendimizi Sınavalım

1. Aşağıdaki SQL ifadelerinden hangisi sorgunun veri kaynağı ile ilgilidir?
 - a. Select
 - b. Having
 - c. From
 - d. Order by
 - e. Where
2. Aşağıdakilerden hangisi Select komutunun mantıksal işleme sırasının en sonunda yer alır?
 - a. Top
 - b. Join
 - c. Order By
 - d. Having
 - e. Where
3. Aşağıdaki komutlardan hangisi ile bir seçme sorgusunda tablolar birbirine bağlanabilir?
 - a. Order by
 - b. Top
 - c. Group By
 - d. Distinct
 - e. Where
4. INTERSECT ifadesi iki veya daha fazla veri kümesi arasında hangi işlemi gerçekleştirir?
 - a. Birleştirme
 - b. Fark
 - c. Ekleme
 - d. Güncelleme
 - e. Kesişim
5. Distinct ifadesi aşağıdakilerden hangisi ile ilgilidir?
 - a. Ortalama hesaplama
 - b. Toplama işlemi
 - c. Gruplama
 - d. Benzersiz değerlerle işlem
 - e. Standart sapma
6. Aşağıdaki özetleme işlevlerinden hangisi metin türündeki veri ile kullanılabilir?
 - a. AVG
 - b. MAX
 - c. STDEV
 - d. SUM
 - e. VAR
7. Choose işlevi aşağıdakilerden hangi işlev kategorisinde yer alır?
 - a. Metin İşlevleri
 - b. Tarih İşlevleri
 - c. Sistem İşlevleri
 - d. Sayısal İşlevleri
 - e. Mantıksal İşlevler
8. Tarih işlevlerinde aşağıdaki kısaltmalardan hangisi ay bilgisini ifade eder?
 - a. yy
 - b. m
 - c. mi
 - d. ms
 - e. mcs
9. DateADD komutunun işlevi nedir?
 - a. O anki tarih bilgisini getirir.
 - b. Metin bilgisini tarih bilgisine çevirir.
 - c. Tarih verisine istenilen birimde ekleme yapar.
 - d. İki tarih arasındaki süreyi hesaplar.
 - e. İki tarihi birleştirir.
10. Aşağıdakilerden hangisi ile alt sorgu ifadesi **kullanılamaz**?
 - a. Group By
 - b. From
 - c. IN
 - d. Exists
 - e. Select

Kendimizi Sınyalım Yanıt Anahtarı

1. c Yanıtınız yanlış ise “Seçme (Select) Komutu” konusunu yeniden gözden geçiriniz.
2. a Yanıtınız yanlış ise “Seçme (Select) Komutu” konusunu yeniden gözden geçiriniz.
3. e Yanıtınız yanlış ise “Birden Fazla Tablodan Veri Seçimi” konusunu yeniden gözden geçiriniz.
4. e Yanıtınız yanlış ise “Veri Kümelerinin Birleştirilmesi” konusunu yeniden gözden geçiriniz.
5. d Yanıtınız yanlış ise “Özetleme İşlevleri” konusunu yeniden gözden geçiriniz.
6. b Yanıtınız yanlış ise “Özetleme İşlevleri” konusunu yeniden gözden geçiriniz.
7. e Yanıtınız yanlış ise “Mantıksal İşlevler” konusunu yeniden gözden geçiriniz.
8. b Yanıtınız yanlış ise “Tarih İşlevleri” konusunu yeniden gözden geçiriniz.
9. c Yanıtınız yanlış ise “Tarih İşlevleri” konusunu yeniden gözden geçiriniz.
- 10.a Yanıtınız yanlış ise “Alt Sorgular” konusunu yeniden gözden geçiriniz.

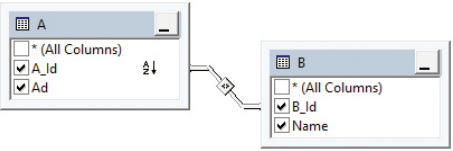
Sıra Sizde Yanıt Anahtarı

Sıra Sizde 1

SQL sorgularındaki koşul satırlarının her bir satırda kontrol edileceği hatırlanırsa “WHERE ([Ürün Adı] = 'Ceviz') AND ([Ürün Adı] = 'Zeytin Yağı’)” koşulunun bir kayıt içerisinde [Ürün Adı] alanının hem Ceviz hem de Zeytin Yağı metnine eşit olması durumu aranmaktadır. Bir alanın bir kayıta farklı iki değere eşit olması mümkün değildir. Dolayısıyla bu sorgu veritabanı verilerinin içeriğine bakılmadan herhangi bir sonuç döndürmeyeceği açıktır. Eğer iki koşul arasında OR ifadesi olsaydı anlamlı bir sorgu ifadesi olurdu.

Sıra Sizde 2

A_Id ve B_Id değerlerinin birbiri ile aynı olmadığı bir veri kümesi elde edilmek istenildiğinde bağlantı kısmına A INNER JOIN B ON A.A_Id <> B.B_Id ifadesinin yazılması gerekirdi. Bu durumda her iki tabloda aynı olan değerler sonuç kümesinde yer almazdı. Aşağıda bağlantı şekli, sorgu ifadesi ve sonuç kümesini bulabilirsiniz.



```
SELECT A.A_Id, A.Ad, B.B_Id, B.Name
FROM A INNER JOIN B ON A.A_Id <> B.B_Id
```

A_Id	Ad	B_Id	Name
1	Bir	2	Two
1	Bir	3	Three
1	Bir	7	Seven
2	İki	7	Seven
2	İki	3	Three
2	İki	1	One
4	Dört	1	One
4	Dört	2	Two
4	Dört	3	Three
4	Dört	7	Seven
8	Sekiz	7	Seven
8	Sekiz	3	Three
8	Sekiz	2	Two
8	Sekiz	1	One

Sıra Sizde 3

Kısıtlama işlemleri için yazılan ifadeler arasında kullanılan mantıksal operatörlerin doğru kullanımı çok önemlidir. Bunun için SQL komutlarının taleplerinin satır bazlı işlendiği unutulmamalıdır. 2006 Nisan ve 2007 Ocak aylarındaki verinin görüntülenmesi için “YEAR(S.[Sipariş Tarihi]) = 2006), (MONTH(S.[Sipariş Tarihi]) = 4, YEAR(S.[Sipariş Tarihi]) = 2007), (MONTH(S.[Sipariş Tarihi]) = 1” kısıtlarının nasıl birleştirileceğini düşünelim. Öncelikle bir kayıttaki tarih bilgisinin yılı 2006 ve ayının 4 olması veya yılı 2007 ve ayının 1 olması gerekir. İşte cümle hâline getirilmiş bu ifadenin kısıt olarak WHERE den sonra yazılmış hâli aşağıdadır. (YEAR(S.[Sipariş Tarihi]) = 2006 AND MONTH(S.[Sipariş Tarihi]) = 4) OR (YEAR(S.[Sipariş Tarihi]) = 2007 AND MONTH(S.[Sipariş Tarihi]) = 1)

Sıra Sizde 4

Sıralama işleminin belirlenen alanlardaki değerlere göre yapılabilmesi **CASE** komutu ile sağlanabilir. Ancak bu dinamik bir süreçtir. Seçme işlem mantık sırasının son adımı olan sıralama işlemi uygulanırken ilgili satırın değerlerine bakılarak sıralama ölçütü değiştirilebilir. Örneğin Aşağıdaki sorguda [*İş Ünvanı*] değeri 'Sahibi' değerini alınca [*İş Ünvanı*] alanına göre diğer durumlarda işe [*Şehir*] alanının değerine göre sıralama yapılır.

```
SELECT *  
FROM [Northwind].[dbo].[Müşteriler]  
Order By  
CASE [İş Ünvanı] when 'Sahibi' THEN [İş Ünvanı]  
ELSE Şehir END
```

Sıra Sizde 5

“Ortalama miktarın üzerinde sipariş edilen sipariş bilgileri” ifadesi sorgunun sözel karşılığıdır.

Yararlanılan ve Başvurulabilecek Kaynaklar

Elmasri, R. ve Navathe, S. B. (2015). **Fundamentals of Database Systems**, United Kingdom: Pearson.

Microsoft SQL Server MSDN [https://msdn.microsoft.com/en-us/library/mt590198\(v=sql.1\).aspx](https://msdn.microsoft.com/en-us/library/mt590198(v=sql.1).aspx)