BİRDEN FAZLA TABLO ÜZERİNDE SORGULAMA

Tablolarda takma isim kullanma

Tablolar birleştirilerek işlem yapıldığında çoğunlukla sorgu içerisinde geçen sütun isimleri tablo isimleriyle beraber kullanılır. Bu durumda uzun tablo isimleri, takma isimler verilerek kısaltılabilir. Tablolar isimlendirilirken FROM deyiminden sonra tablo adı yazılıp AS takma_isim yazılmalıdır.

```
Select ogr.adi_soyadi, bl.bolum_adi
    from ogrenciler AS ogr, bolumler AS bl
    where ogr.bolumkod = bl.bolumkod
```

Birleştirme (JOIN) işlemi

KLASİK JOIN (WHERE ifadesiyle birleştirme)

Birleştirme işlemi ilk başlarda sadece WHERE ifadesi ile gerçekleşiyordu. SQL-92 ve sonrası standartlarda JOIN ifadeleri kullanılmaya başlandı. Bulunla birlikte WHERE ifadesiyle birleştirme kullanımı kalkmamıştır. Bu tür birleştirmede tabloların ortak sütunları koşul ifadesi kısmında eşitlenmelidir.

OGRENCI		
OgrNo	**Adi**	**BKod**
2011601	Ali	1
2011602	Ayşe	2
2011603	Veli	5
2011604	Can	1
2011605	Hatice	3
2011606	Mehmet	1
2011607	Hasan	6
2011608	Neşe	2

BOLUM	
Kod	**Adi**
1	Bilgisayar
2	Bankacılık
3	Muhasebe
4	Büro

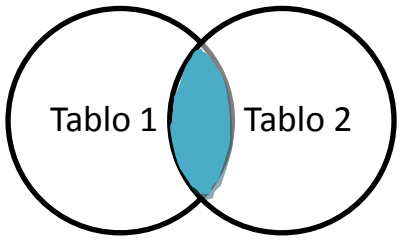
```
Select ogr.adi, bl.adi
    from ogrenci AS ogr, bolum AS bl
    where ogr.bkod = bl.kod
```

Yukarıdaki SQL cümleciği çalıştırıldığında OGRENCI tablosundaki bkod sütunundaki bilginin BOLUM tablosundaki kod sütunundaki bilgi ile eşleştirilebildiği kayıtlar karşımıza gelecektir. Bu INNER JOIN işlemiyle eşdeğerdir. Sonuç olarak aşağıdaki bilgi gelecektir.

```
ogr.adi         bl.adi
Ali             Bilgisayar
Ayşe            Bankacılık
Can             Bilgisayar
Hatice          Muhasebe
Mehmet          Bilgisayar
Neşe            Bankacılık
```

Bu birleştirmede WHERE ifadesinden sonraki koşul ifade kullanılmazsa iki tablodaki satır sayılarının çarpımı kadar satır yeni tabloda listelenir. Bu da ileride anlatacağımız CROSS JOIN işlemiyle eşdeğerdir.

INNER JOIN



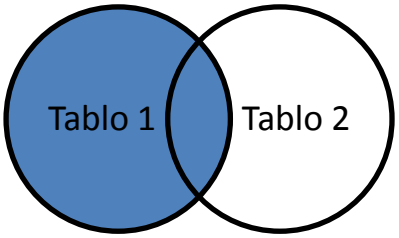
İki veya daha çok tablonun ortak sütunlarının içeriği kontrol edilerek birleştirme işlemi yapılır. Birleştirme yapılan sütunun her iki tabloda da ortak olan verilerini gösterir. INNER JOIN yazmak yerine sadece JOIN yazmak da yeterlidir.

İki tablo arasında birleştirme yaparken, tablolardan her ikisinde de yer alan değerler seçilir, tablolardan sadece birinde yer alıp diğerinde ilişkili değere rastlanmayan satırlar seçilmez.

```
SELECT ogr.adi, bl.adi
    FROM ogrenci AS ogr INNER JOIN bolum AS bl
    ON ogr.bkod = bl.kodu
```

```
ogr.adi         bl.adi
Ali             Bilgisayar
Ayşe            Bankacılık
Can             Bilgisayar
Hatice          Muhasebe
Mehmet          Bilgisayar
Neşe            Bankacılık
```

LEFT OUTER JOIN

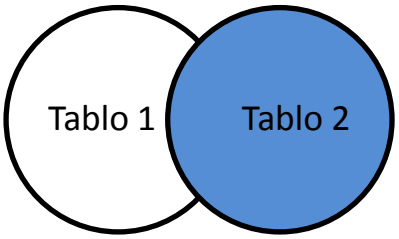


Bu birleştirmede ifadenin solundaki tablo belirleyici tablodur. Bu tablonun diğer tabloyla ilişkisi bulunsun veya bulunmasın tüm satırları listelenecektir. Sağındaki tabloda ise sadece ilişkili satırlar listelenecektir. Belirleyici tablonun içerdiği değerin sağdaki tabloda karşılığı yoksa sağ tablodan gelecek değerler sonuç kümesinde NULL olarak gösterilecektir.

```
SELECT ogr.adi, bl.adi
    FROM ogrenci AS ogr LEFT OUTER JOIN bolum AS bl
    ON ogr.bkod = bl.kodu
```

```
ogr.adi         bl.adi
Ali             Bilgisayar
Ayşe            Bankacılık
Veli            NULL
Can             Bilgisayar
Hatice          Muhasebe
Mehmet          Bilgisayar
Hasan           NULL
Neşe            Bankacılık
```

RIGHT OUTER JOIN

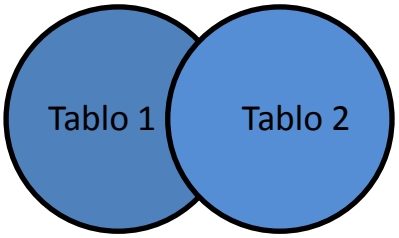


Bu birleştirmede ifadenin sağındaki tablo belirleyici tablodur. Bu tablonun diğer tabloyla ilişkisi bulunsun veya bulunmasın tüm satırları listelenecektir. Solundaki tabloda ise sadece ilişkili satırlar listelenecektir. Belirleyici tablonun içerdiği değerin soldaki tabloda karşılığı yoksa sol tablodan gelecek değerler sonuç kümesinde NULL olarak gösterilecektir.

```
SELECT ogr.adi, bl.adi
    FROM ogrenci AS ogr RIGHT OUTER JOIN bolum AS bl
    ON ogr.bkod = bl.kodu
```

```
ogr.adi         bl.adi
Ali             Bilgisayar
Can             Bilgisayar
Mehmet          Bilgisayar
Ayşe            Bankacılık
Neşe            Bankacılık
Hatice          Muhasebe
NULL            Büro
```

FULL OUTER JOIN



LEFT OUTER JOIN ve RIGHT OUTER JOIN işleminin birleşim kümesidir. Yani iki tablodaki tüm satırlar listelenecek ve ilişkisi olmayanlar NULL değer getirecektir.

```
SELECT ogr.adi, bl.adi
    FROM ogrenci AS ogr FULL OUTER JOIN bolum AS bl
    ON ogr.bkod = bl.kodu
```

ogr.adi	bl.adi
Ali	Bilgisayar
Ayşe	Bankacılık
Veli	NULL
Can	Bilgisayar
Hatice	Muhasebe
Mehmet	Bilgisayar
Hasan	NULL
Neşe	Bankacılık
NULL	Büro

CROSS JOIN

CROSS JOIN birleştirilen tablolardaki tüm satırların listelenmesini sağlar. İki tabloda yer alan satırları çaprazlamak için kullanılır. Sorgu sonucu iki tablonun satırlarının sayısının çarpımı kadar satırdan oluşmaktadır. İlk tablonun her satırı ikinci tablonun satır sayısı kadar tekrar edecektir.

```
SELECT renk.adi, urun.adi
    FROM urun CROSS JOIN renk
```

Ürün Tablosu

Kod	Adi
1	Tişort
2	Gömlek

Renk Tablosu

Kod	Adi
1	Mavi
2	Kırmızı
3	Yeşil

renk.adi	urun.adi
Mavi	Tişort
Kırmızı	Tişort
Yeşil	Tişort
Mavi	Gömlek
Kırmızı	Gömlek
Yeşil	Gömlek

NATURAL JOIN

NATURAL JOIN, elde ettiği sonuç kümesi olarak INNER JOIN türü birleşme ile benzemektedir. Bu tür birleşmede koşul ifadesini belirtmeye gerek yoktur. Sadece tablo isimlerinin yazılması yeterlidir. Ortak sütunlar (isimleri aynı olan) otomatik olarak birleşirler.

```
SELECT ogr.adi, bl.adi
    FROM ogrenci NATURAL JOIN bolum
```

SELF JOIN

Tek bir tablonun kendi kendisi ile birleşmesine SELF JOIN denir. Tablo kendi kendisi ile birleştirilecekse 2 farklı takma isim kullanılır.

Örneğin aşağıdaki tablo personel bilgilerini içermektedir. Her personelin bir yöneticisi vardır. Yönetici de bir personel olduğu için ayrı bir tabloda değil yine personel tablosunda tanımlanmıştır. Personelin yöneticilerinin isimlerini bulmak için tabloyu kendi kendisi ile birleştirmek gerekiyor. Biri personelin kendisi için, biri de yöneticisi için olmak üzere tabloyu 2 farklı takma isimle kullanılmıştır.

Sicil_No	Adi	Yonetici
1	Faruk Toklu	NULL
2	Adnan Gökten	1
3	Ayhan Radavuş	1
4	Erkut Tekeli	2
5	Tufan Özdayı	2
6	Zafer Berikol	3
7	Ömer Faruk Rençber	3
8	Cengiz Aytun	3

```
SELECT p1.sicil_no, p1.adi, p2.adi AS yönetici
    FROM personel AS p1 LEFT OUTER JOIN personel AS p2
        ON p1.yonetici = p2.sicil_no
```

```
Sicil_NO   Adi                  Yonetici
    1      Faruk Toklu          NULL
    2      Adana Gökten         Faruk Toklu
    3      Ayhan Radavuş        Faruk Toklu
    4      Erkut Tekeli         Adnan Gökten
    5      Tufan Özdayı         Adnan Gökten
    6      Zafer Berikol        Ayhan Radavuş
    7      Ömer Faruk Rençber   Ayhan Radavuş
    8      Cengiz Aytun         Ayhan Radavuş
```

ORTAK SÜTUNU BULUNMAYAN TABLOLARI BİRLEŞTİRME

Eğer birleştirilecek tablolarda ortak sütun yoksa birleştirme işleminde koşul olarak eşitleme yerine BETWEEN ifadesi kullanılır. BETWEEN ifadesi yerine "<=" ve ">=" gibi operatörler de kullanılabilir.

Burası yazılacak…

INTERSECT (İki sorgunun kesişimi)

INTERSECT ifadesi aynı sütun isimlerine sahip iki veya daha fazla sorgudan dönen değerlerin kesişim kümesini almak için kullanılır. Her iki tarafta da aynı olan satırlar teke indirgenir.

INTERSECT işleminin yapılabilmesi için her iki sorgu da aynı sütunları içermek zorundadır. Farklı sütunlar bulunduğu takdirde yazılan ifade hata verecektir. Bu yüzden yazılan sorgularda * ifadesi ile tüm sütunları seçmek yerine ortak sütunlar seçilmelidir.

Satis Tablosu

Tarih	Musteri	Urun	Tutar
3.3.2012	ABC Tekstil	Tişort	3000
4.3.2012	Ozler Tekstil	Gömlek	2300
6.3.2012	ABC Tekstil	Gömlek	2100
16.3.2012	Hasan Tekstil	Pantolon	4000
18.3.2012	Ozler Tekstil	Tişort	3400
19.3.2012	ABC Tekstil	Pantolon	1800
23.3.2012	Hasan Tekstil	Pantolon	6200
24.3.2012	Hasan Tekstil	Pantolon	2200

Alis Tablosu

Tarih	Tedarikci	Urun	Tutar
2.3.2012	ABC Tekstil	Kumaş	1400
5.3.2012	İpek Tekstil	İplik	800
9.3.2012	Ömer Tuhafiye	Düğme	600
11.3.2012	Candan İplik	İplik	1100
12.3.2012	Hasan Tekstil	Kumaş	2200
16.3.2012	Ömer Tuhafiye	Düğme	300
20.3.2012	İpek Tekstil	İplik	900
21.3.2012	ABC Tekstil	Kumaş	700

```
SELECT Musteri as Firma FROM Satis
INTERSECT
SELECT Tedarikci as Firma FROM Alis
```

Yukarıda yazılan SQL cümlesinde; hem Satis tablosunda hem de Alis tablosunda olan firmalar sorgulanıyor. Sütun isimleri aynı olmadığı için takma isim verilerek sütun isimleri aynı hale getirilmiştir. SQL Cümlesinin icra edilmesiyle aşağıdaki sonucu elde ederiz.

```
Firma
ABC Tekstil
Hasan Tekstil
```

EXCEPT (İki sorgunun farkı)

Aynı sütun isimlerine sahip iki sorgudan dönen değerlerin farkını almak için kullanılır. Yani bir sorguda olup diğerinde olmayan değerleri alacaktır. Fark alanında aynı olan satırlar teke indirgenir.

```
SELECT Musteri as Firma FROM Satis
EXCEPT
SELECT Tedarikci as Firma FROM Alis
```

Yukarıda yazılan SQL cümlesinde; Satis tablosunda olup da Alis tablosunda olmayan firmalar sorgulanıyor. Sütun isimleri aynı olmadığı için takma isim verilerek sütun isimleri aynı hale getirilmiştir. SQL cümlesinin icra edilmesiyle aşağıdaki sonucu elde ederiz.

```
        Firma
-----------------------
Özler Tekstil
```

UNION (İki sorguyu birleştirme)

Aynı sütun isimlerine sahip iki sorgunun sonuçlarını birleştirmek için kullanılır. İki sorgunun sonuçları birleştirilirken tamamen aynı olan satırlardan yalnızca birisi listelenir.

```
SELECT Musteri as Firma FROM Satis
UNION
SELECT Tedarikci as Firma FROM Alis
```

Yukarıda yazılan SQL cümlesinde; Satis tablosunda olan müşterilere ilave olarak Alis tablosunda olan tedarikçiler sorgulanıyor. Sütun isimleri aynı olmadığı için takma isim verilerek sütun isimleri aynı hale getirilmiştir. Birbirinin aynı olan satırlardan yalnızca birisi listeleniyor. SQL cümlesinin icra edilmesiyle aşağıdaki sonucu elde ederiz.

```
        Firma
-----------------------
ABC Tekstil
Özler Tekstil
Hasan Tekstil
İpek Tekstil
Ömer Tuhafiye
Candan İplik
```

Union deyimi kullanarak ikinci bir örnek yapalım. Yapılan satış ve alış tutarlarının firmalara göre tek bir sorguda alınması için aşağıdaki SQL cümlesini oluşturalım.

```
SELECT 'Satış' AS Turu, Musteri as Firma, Sum(Tutar) as 'Toplam Tutar'
        FROM Satis
        Group By Musteri
UNION
SELECT 'Alış' AS Turu, Tedarikci as Firma, Sum(Tutar) as 'Toplam Tutar'
        FROM Alis
        Group By Tedarikci
```

İlk SQL cümlesi müşteri bazında satış tutarlarının toplamlarını getiriyor. Her satır için Turu isimli bir sütun açılmış ve içeriğine 'Satış' kelimesi girilmiştir. İkinci SQL cümlesi tedarikçi bazında alış tutarlarının toplamlarını getiriyor. Her satır için yine Turu isimli bir sütun açılmış ve içeriğine 'Alış' kelimesi girilmiştir. Sütun isimleri aynı olmadığı için takma isim verilerek sütun isimleri aynı hale getirilmiştir. SQL Cümlesinin icra edilmesiyle aşağıdaki sonucu elde ederiz.

Turu	Firma	Toplam Tutar
Satış	ABC Tekstil	6900
Satış	Özler Tekstil	5700
Satış	Hasan Tekstil	12400
Alış	ABC Tekstil	2100
Alış	İpek Tekstil	1700
Alış	Ömer Tuhafiye	900
Alış	Candan İplik	1100
Alış	Hasan Tekstil	2200

UNION ALL (İki sorguyu birleştirme-tekrarlı)

UNION deyimi gibi aynı sütun isimlerine sahip iki sorgunun sonuçlarını birleştirmek için kullanılır. Farklı olarak iki sorgunun sonuçları birleştirilirken tamamen aynı olan satırların hepsi listelenecektir.

```
SELECT Musteri as Firma FROM Satis
UNION ALL
SELECT Tedarikci as Firma FROM Alis
```

Yukarıda yazılan SQL cümlesinde; Satis tablosunda olan müşterilere ilave olarak Alis tablosunda olan tedarikçiler sorgulanıyor. Sütun isimleri aynı olmadığı için takma isim verilerek sütun isimleri aynı hale getirilmiştir. Birbirinin aynı olan satırların tamamı listelenmiştir. SQL cümlesinin icra edilmesiyle aşağıdaki sonucu elde ederiz.

```
       Firma
-----------------------
ABC Tekstil
Ozler Tekstil
ABC Tekstil
Hasan Tekstil
Ozler Tekstil
ABC Tekstil
Hasan Tekstil
Hasan Tekstil
ABC Tekstil
İpek Tekstil
Ömer Tuhafiye
Candan İplik
Hasan Tekstil
Ömer Tuhafiye
İpek Tekstil
ABC Tekstil
```

INTERSECT ALL (İki sorgunun kesişimi-tekrarlı)

INTERSECT ifadesindeki gibi aynı sütun isimlerine sahip iki veya daha fazla sorgudan dönen değerlerin kesişim kümesini almak için kullanılır. Her iki tarafta da aynı olan satırlar teke indirgenmez. Birden fazla geçen satırlar, daha az geçen taraftaki kadar görünür.

```
SELECT Musteri as Firma FROM Satis
INTERSECT ALL
SELECT Tedarikci as Firma FROM Alis
```

```
       Firma
-----------------------
ABC Tekstil
ABC Tekstil
Hasan Tekstil
```

EXCEPT ALL (İki sorgunun farkı-tekrarlı)

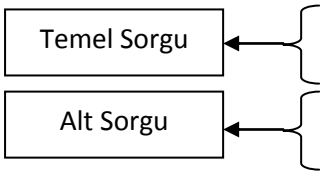
Aynı sütun isimlerine sahip iki sorgudan dönen değerlerin farkını almak için kullanılır. Yani bir sorguda olup diğerinde olmayan değerleri alacaktır. Fark alanında aynı olan satırlar varsa, bu satırlar satır adedi kadar tekrarlanır.

```
SELECT Musteri as Firma FROM Satis
EXCEPT ALL
SELECT Tedarikci as Firma FROM Alis
```

```
       Firma
-----------------------
Özler Tekstil
Özler Tekstil
```

ALT SORGULAR

Alt sorgular, bir SQL sorgusu içerisinde başka bir SQL sorgusu kullanılarak oluşturulan SQL ifadeleridir. İçeride çalışan sorgu, dışarıda çalışan sorguya değer üretebilir. Bu, bazen tekil bir değer olabileceği gibi, bazen birden fazla değer de döndürebilir.




```
SELECT sütunlar FROM tablo_adi
WHERE koşul_ifadesi (
                        SELECT kaynak_sütun_adi
                        FROM kaynak_tablo_adi
                        )
```

Alt sorgularda dikkat edilecek noktalar

- Alt sorgu içerisinde listelenecek sütun sayısı birden fazla olmamalıdır.
- Alt sorgu SELECT, INSERT, UPDATE veya DELETE temel sorgularının içerisinde kullanılabilir.
- Alt sorgular parantez içerisine alınmalıdır.
- Text veya image veri tipleri alt sorguda kullanılmaz.
- Alt sorgu ORDER BY yapısını içermez.
- ORDER BY veya GROUP BY içerisinde alt sorgu kullanılmaz.
- İç içe kullanılacak alt sorgu sayısı sınırlıdır. (MS-SQL de 32'ye kadar, ORACLE'da 255'e kadar alt sorgu iç içe oluşturulabilir.)
- Alt sorguda birleştirme (join) işlemi kullanılmaz.
- Alt sorguda kendi from ifadesi veya diğer from ifadesinde belirtilen sütunları kullanabilir.
- Alt sorgu ve temel sorgu içerisinde sütun isimleri **tablo_adi.sütun_adi** şeklinde veya tabloya takma isimler vererek kullanılabilir.
- Koşul kısmında karşılaştırma operatörleri kullanıldığı zaman alt sorgudan tek bir değer dönmesi gerekmektedir. Alt sorgudan dönen değer sayısı birden fazlaysa koşul kısmında **IN** veya **NOT IN** gibi çoklu değerleri karşılaştırmak için kullanılan ifadeler kullanılmalıdır.
- Bir sorguyu, alt sorgularla çalıştırmak yerine JOIN kullanarak gerçekleştirmeye çalışmanız önerilir. Alt sorgulara, performans açısından "daha sonra düşünülmesi gereken çözümler" olarak bakmakta fayda vardır.

Alt Sorgu Oluşturulması

Aşağıda öğrenci ve bolumler tabloları ve içerdiği kayıtlar verilmektedir.

Ogrenci_No	Adi	Bolum_Kodu
2010689608	Rabia Tavşan	1
2011689001	İslam Anşin	5
2011689002	Erhan Ateş	3
2011689006	Onur Bozkurt	2
2011689008	Gülşah Çetin	1
2011689009	Ayşe Diler	1
2011689010	Yunus Dumlu	2
2011689011	Emine Elhan	4
2011689013	Yusuf Kağan Eroğlu	2
2011689014	Erdem Ersoy	1

Kodu	Adi
1	Bilgisayar
2	Muhasebe
3	Bankacılık
4	Büro
5	Mobilya

Gülşah Çetin ile aynı bölümde olan öğrenciler bulunmak isteniyor. Ama Gülşah Çetin'in Bölüm Kodu bilinmiyor. Böyle bir durumda ilk önce Gülşah Çetin'in bölüm kodu aşağıdaki gibi bulunur.

SELECT Bolum_Kodu FROM ogrenci WHERE Adi = 'Gülşah Çetin'

Sonuç 1 olarak geri dönecektir.

Bölüm Kodunu öğrendikten sonra Bölüm Kodu 1 olan öğrenciler sorgulanır ve aşağıdaki sonuç elde edilir.

SELECT Adi, Bolum_Kodu FROM ogrenci WHERE Bolum_Kodu = 1

```
        Adi          Bolum_Kodu
 Rabia Tavşan             1
 Gülşah Çetin             1
 Ayşe Diler               1
 Erdem Ersoy              1
```

Yalnız tek bir sorguyla aynı sonucu elde etmek için alt sorgular kullanılır. Aşağıda alt sorgu içeren SQL ifadesi verilmiştir.

```
SELECT Adi, Bolum_Kodu FROM ogrenci
   WHERE Bolum_Kodu = (
                              SELECT Bolum_Kodu FROM ogrenci
                              WHERE Adi = 'Gülşah Çetin'
                              )
```

Bilgisayar bölümündeki öğrencilerin numara, adi ve bolum adını bilgisayar_ogr isimli tabloya yeni kayıt olarak eklemek için aşağıdaki SQL ifadesi kullanılır.

```
INSERT INTO bilgisayar_ogr (ogr_no, adi, bolum)
    SELECT ogr_no, adi, 'Bilgisayar' FROM ogrenci WHERE Bolum_Kodu=1
```

Yukarıdaki sorguda bilgisayar bölümünün kodunun bilindiği düşünülmüştür. Eğer bilinmiyorsa ve bölüm tanımları bölümler adında bir tabloda tutuluyorsa aşağıdaki şekilde iki alt sorguyla da işlem gerçekleştirilebilir.

```
INSERT INTO bilgisayar_ogr (ogr_no, adi, bolum)
    SELECT ogr_no, adi, 'Bilgisayar' FROM ogrenci
    WHERE Bolum_Kodu=(
                              SELECT Kodu FROM bolumler
                              WHERE Adi = 'Bilgisayar'
                              )
```

Alt Sorgularda IN ve NOT IN Kullanımı

Alt sorgudan dönen değerlerin birden fazla olduğu durumlarda koşul ifadesi bölümünde **IN** veya **NOT IN** ifadesi kullanılabilir.

Bilgisayar ve Mobilya bölümlerine kayıtlı öğrencileri bulmak için aşağıdaki SQL cümlesi kullanılır.

SELECT Adi, Bolum_Kodu FROM ogrenci WHERE Bolum_Kodu **IN** (1,5)

Adi	Bolum_Kodu
Rabia Tavşan	1
İslam Anşin	5
Gülşah Çetin	1
Ayşe Diler	1
Erdem Ersoy	1

Muhasebe ve Mobilya dışındaki bölümlere kayıtlı öğrencileri bulmak için aşağıdaki SQL cümlesi kullanılır.

SELECT Adi, Bolum_Kodu FROM ogrenci WHERE Bolum_Kodu **NOT IN** (2,5)

Adi	Bolum_Kodu
Rabia Tavşan	1
Erhan Ateş	3
Gülşah Çetin	1
Ayşe Diler	1
Emine Elhan	4
Erdem Ersoy	1

Bölüm kodlarını bilmediğimizi varsayarsak **IN** ve **NOT IN** için değerler kümesini alt sorgu kullanarak oluşturabiliriz.

```
SELECT Adi, Bolum_Kodu FROM ogrenci
WHERE Bolum_Kodu IN (
                        SELECT Kodu FROM bolumler
                        WHERE Adi IN ('Bilgisayar' , 'Mobilya')
                        )
```

```
SELECT Adi, Bolum_Kodu FROM ogrenci
WHERE Bolum_Kodu NOT IN (
                                SELECT Kodu FROM bolumler
                                WHERE Adi IN ('Muhasebe' , 'Mobilya')
                                )
```

Alt Sorgularda ANY ve ALL Kullanımı

ANY : Alt sorgudan dönen bir grup verinin **herhangi birini** sağlayan durumlar için kullanılır.

ALL : Alt sorgudan dönen bir grup verinin **tamamını** sağlayan durumlar için kullanılır.

ANY ve ALL ifadeleri temel sorgunun koşul kısmında karşılaştırma operatörleriyle birlikte kullanılır. Örneğin;

WHERE fiyat > ANY(20,10,50) : Bu ifadede fiyatı 20, 10 veya 50 ‘den en az birinden büyük olanlar anlamına gelir.

WHERE fiyat > ALL(20,10,50) : Bu ifadede fiyatı 20, 10 veya 50 ‘nin tamamından büyük olanlar anlamına gelir.

Adi	**Gorev**	**Maas**
Ali	Tekniker	1200
Mustafa	Tekniker	1400
Ayşe	Hizmetli	900
Fuat	Mühendis	2100
Dursun	Mühendis	1300
Hasan	Mühendis	1100
Mete	Hizmetli	800

Yukarıdaki tabloda Teknikerlerden herhangi birinden daha düşük maaş alan Mühendisleri listeleyen SQL cümlesi aşağıdadır.

```
SELECT * FROM personel
WHERE Maas ANY (
                    SELECT maas FROM Personel
                    WHERE Gorev = ‘Tekniker’)
                    )
                    AND gorev = ‘Mühendis’
```

```
       Adi            Gorev         Maas
-------------------------------------------
Dursun            Mühendis         1300
Hasan             Mühendis         1100
```

Teknikerlerden tamamından daha düşük maaş alan Mühendisleri listeleyen SQL cümlesi aşağıdadır.

```
SELECT * FROM personel
WHERE Maas ALL (
                    SELECT maas FROM Personel
                    WHERE Gorev = ‘Tekniker’
                    )
                    AND gorev = ‘Mühendis’
```

```
       Adi            Gorev         Maas
-------------------------------------------
Hasan             Mühendis         1100
```

Alt Sorguda EXISTS ve NOT EXISTS kullanımı

EXISTS alt sorgudan dönen değer olup olmadığını kontrol etmek için kullanılır. Temel sorgunun koşul kısmında kullanılır. Alt sorgu sonucunda en az bir satır dönerse koşul sağlanmış olur.

NOT EXISTS ifadesi de alt sorgudan dönen değer olup olmadığını kontrol etmek için kullanılır. Temel sorgunun koşul kısmında kullanılır. Alt sorgu sonucunda hiçbir satır dönmezse koşul sağlanmış olur.

Alt sorguları oluştururken tek bir sütun kullanmak gerekirdi. Ama EXISTS ve NOT EXISTS ifadelerinde böyle bir kısıtlama yoktur.

Satıcı

Satno	Adı	Sehir
1	Ali	İstanbul
2	Ayşe	İstanbul
3	Akın	Ankara
4	Can	İzmir
5	Mert	Adana

Satış

Satici_No	Parca_No	Miktar
1	1	200
1	27	300
2	27	250
2	42	115
3	1	500
3	2	55
4	15	800
5	18	900
5	27	400

27 nolu parçayı satan satıcıları listeleyelim.

```
SELECT * FROM satici
WHERE EXISTS (
                SELECT * FROM satis
                WHERE Satici_No = Satno AND Parca_No = 27
                )
```

```
Satno   Adi          Sehir
1       Ali          İstanbul
2       Ayşe         İstanbul
5       Mert         Adana
```

27 nolu parçayı satmayan satıcıları listeleyelim.

```
SELECT * FROM satici
WHERE NOT EXISTS (
                SELECT * FROM satis
                WHERE Satici_No = Satno AND Parca_No = 27
                )
```

Satno	Adi	Sehir
3	Akın	Ankara
4	Can	İzmir

Sütun İsimlerinde Alt Sorgu Kullanımı (İlintili Alt Sorgular)

Bazı durumlarda listelenecek sütunlardan biri olarak alt sorgudan dönen bilgiyi kullanabiliriz. Bu durumda üst sorgunun döndürdüğü her bir satır için alt sorgu tekrar çalıştırılır. Alt sorgu çalıştırılırken, üstteki sorgularda kullanılan tablo isimlerine erişip ilgili satırdaki verileri çekebilir. Ancak bu programlama mantığındaki döngü yapılarına eştir ve çok fazla yük oluşturacaktır.

```
SELECT Ogrenci_No, Adi, (
                        SELECT  bolumler.Adi FROM bolumler
                        WHERE bolumler.Kodu = Ogrenci.Bolum_Kodu
                        ) AS ‘Bölüm Adı’
FROM Ogrenci
```

Ogrenci_No	Adi	Bölüm Adı
2010689608	Rabia Tavşan	Bilgisayar
2011689001	İslam Anşin	Mobilya
2011689002	Erhan Ateş	Bankacılık
2011689006	Onur Bozkurt	Muhasebe
2011689008	Gülşah Çetin	Bilgisayar
2011689009	Ayşe Diler	Bilgisayar
2011689010	Yunus Dumlu	Muhasebe
2011689011	Emine Elhan	Büro
2011689013	Yusuf Kağan Eroğlu	Muhasebe
2011689014	Erdem Ersoy	Bilgisayar

FROM ifadesinde alt sorgu kullanılması (Türetilmiş tablolar)

Bildiğimiz gibi SQL cümlelerinde FROM ifadesinden sonra kullanılacak tablolar belirtilir. Gerektiğinde alt sorgularla üretilen değerleri temel sorgu için bir tablo olarak kullanabiliriz.

Personel tablosunda aynı görevdeki personellerin ortalama maaşından daha fazla maaş alan personeli listeleyelim. Bu tür alt sorgularda, hesaplanmış bütün sütunlara bir takma ad verilmelidir.

```
SELECT p.Gorev, P.Adi, P.Maas, Y.OrtMaas as "Ortalama Maaş"
FROM personel AS p, (
                    SELECT Gorev, AVG(Maas) AS OrtMaas
                    FROM Personel GROUP BY Gorev
                    ) AS y
WHERE p.Gorev = y.Gorev AND p.Maas > Y.OrtMaas
```

Gorev	Adi	Maas	Ortalama Maaş
Hizmetli	Ayşe	900	850
Mühendis	Fuat	2100	1500
Tekniker	Mustafa	1400	1300