

T.C.

ONDOKUZ MAYIS ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



VERİ TABANI LABORATUVARI
FÖY-2

SORGU ANALİZİ YAPMA VE SORGU OPTİMİZASYONU

SQL SORGULARI

SELECT: Tablo üzerinden gerekli bilgileri elde etmek ve sorgulama yapmak için kullanılır. Tablo adı ile belirtilen tablo içindeki tüm bilgiler koşulsuz olarak listelenecektir. SELECT sözcüğünü izleyen kısımda “*” karakterinin bulunması, ilgili tablodaki bütün alan isimlerinin ve bu alanlardaki bilgilerin listelenmesini sağlayacaktır. FROM ifadesinden sonra ise kullanılacak tablo adı belirtilebilir.

Kullanımı:

SELECT [sütun_listesi] FROM [tablo_listesi]

Örnek: SELECT adı, Soyadı FROM PERSONEL

Bu satır ile Personel tablosunda bulunan adı ve soyadı alanlarını seçersiniz.

adı	Soyadı
ali	ak
veli	kara
ayşe	ay
mehmet	tek
*	

Örnek: SELECT * FROM PERSONEL

Bu satır ile Personel tablosunda bulunan tüm alanlar seçilmiş olur.

Personel_no	adı	Soyadı	Bölümü	İl_Kodu
1	ali	ak	matematik	34
2	veli	kara	matematik	34
30	ayşe	ay	ingilizce	35
44	mehmet	tek	müzik	07
*				

Select komutunun genel kullanımı:

SELECT [DISTINCT | ALL] < sütun (lar) > FROM <tablo adı (lar)>

[WHERE < şart (lar) >]

[GROUP BY < sütun (lar) >]

[HAVING < grup kısıtlaması >]

[ORDER BY <sütun (lar) [ASC | DESC >]]

DISTINCT: Birbirleriyle aynı olan satırların, listeleme esnasında bir kez yazılmasını sağlayan ifadedir.

Kullanımı: `SELECT DISTINCT sütun adı FROM tablo adı`

Örnek: `SELECT DISTINCT adi FROM ogrenci`

ORDER BY: Listelenecek bilgilerin belirli bir alan adına göre sıralanmasını sağlamak için kullanılan komuttur.

Kullanımı: `SELECT * FROM tablo adı ORDER BY sütun adı ASC`

ASC: Sözcüğü nota göre sıralamanın artan olarak yapılmasını sağlar.

DESC: Nota göre sıralamanın azalan olarak yapılmasını sağlar.

Ayrıca tablo içindeki veriler sıralanırken aynı anda birden fazla alana göre sıralama yapmak mümkündür.

Örnek: `SELECT * FROM notlar ORDER BY final ASC`

`SELECT * FROM notlar ORDER BY final ASC, vize DESC`

Karşılaştırma ifadeleri

(NOT, OR, AND), birden fazla koşula göre sıralama işlemlerinde bu ifadeler kullanılır.

İfade	Sembol
>	Büyük
<	Küçük
=	Eşit
<=	Küçük Eşit
>=	Büyük Eşit
<>	Eşit Değil

WHERE:

Veri tabanında veriyi alma işlemi sırasında satırlara birtakım sınırlamalar getirilerek tablonun tüm satırları yerine istenildiği kadarını elde etmek mümkündür. Tabloda belirli kısımları seçme işlemini gerçekleştirmek için WHERE sözcüğü kullanılmaktadır.

Nümerik ifadelerde Kullanımı: `SELECT * FROM tablo adı WHERE final > 60`

Karakterlerde ifadelerde Kullanımı: `SELECT * FROM ogrenci WHERE adi = 'murat'`

Tarih türü ifadelerde Kullanımı: SELECT * FROM ogrenci WHERE Dog_Tarihi > '1979/09/19'

Örnek: SELECT * FROM öğrenci WHERE yasi < 19 and adi='mehmet'

SELECT adi FROM ogrenci WHERE yasi BETWEEN 10 AND 15

BETWEEN: Belirli bir aralık içindeki bilgileri listelemek için kullanılır.

Kullanımı: SELECT * FROM tablo adi WHERE sütun adi BETWEEN baslangıç AND bitis

Örnek: SELECT * FROM notlar WHERE final BETWEEN 40 AND 70 ORDER BY final ASC

LIKE: Karakter türü verilerde, veri içerisinde geçen belirli bir kelime veya verileri bulmak için kullanılır. % sembol A harfinin öncesi ve sonrasında herhangi bir bilgi alabileceğini gösterir. LIKE sözcüğünü, alt tire(_) sembolü ile birlikte kullanmakta mümkündür.

Örnek: SELECT * FROM ogrenci WHERE adi LIKE '%A%'

SELECT * FROM ogrenci WHERE adi LIKE ' __A%'

SELECT * FROM ogrenci WHERE adi LIKE 'A_K_T'

IN ifadesi: Belli bir kolonun kümesi verilerek işlerin daha kolay yapılmasını sağlar.

Örnek: SELECT * FROM notlar WHERE numara IN (SELECT numara FROM ogrenci WHERE bol kod=531)

ANY ifadesi: Aşağıdaki örnek soru çerçevesinde bu sözcüğün SELECT komutu içindeki etkisi açıklanacaktır.

Örnek: Satış (bol_no=1) bölümünde çalışan personelin herhangi birinden daha düşük maaş alan ve mühendislik (bol_no=2) bölümünde çalışan kişileri listeleyiniz.

SELECT * FROM Personel WHERE maas < ANY(SELECT maas FROM Personel WHERE bol_no= 2) AND bol_no=1

ALL ifadesi: "Hepsi, tamamı" anlamdaki bu sözcük, SELECT komutu içerisinde belirli bir koşulu sağlayan bir gruptan tamamı sağlanan koşullarla ilişkili olarak kullanılır.

Örnek: SELECT * FROM notlar WHERE final > ALL (SELECT final FROM notlar WHERE op_kod=421)

İç içe Select: Bazı sorgulamalar, özelliği itibarı ile iç içe SELECT komutlarının kullanılmasını gerektirebilir. İçteki SELECT komutunun bulduğu sonuç, dıştaki SELECT komutunun işlevi yere getirmesi için kullanılır.

Örnek: SELECT proje_ad,yer FROM Proje WHERE bl_no in(SELECT bol_no FROM Personel WHERE adres LIKE '%Fatih%')

Aritmetik İşlemler: SELECT komutu ile veri tabanında mevcut tablolardan listeleme yaparken, tabloda ayrı bir sütun olarak yer almamış ve bir hesaplama sonucunda üretilebilecek bilgileri de liste içine katmak mümkündür.

Örnek: Aşağıdaki SELECT komutu ile bir personel tablosunda personelin şu anda geçerli olan maaşı ile bu maaşın %10 zamlı şekli listelenmektedir.

```
SELECT ad, soyad, maas, maas*1.1 FROM personel;
```

```
SELECT ogrenci_no, ders_adi, vize*0.4 + final*0.6 FROM notlar WHERE ogrenci_no=1;
```

Kümeleme Fonksiyonları: SQL tablo içerisindeki çeşitli matematiksel ifadelerin sonucunu otomatik olarak üretmeyi sağlayan fonksiyonlara sahiptir.

SUM FONKSİYONU: Tablo içerisinde belirli bir sütuna göre toplama işlemi gerçekleştirir.

Kullanımı: Select SUM (sütun_adi) FROM Tablo_adi;

Örnek: SELECT SUM(maas) FROM personel

AVG FONKSİYONU: Belirli bir alan üzerinde aritmetik ortalama (average) hesaplamak için kullanılır.

Kullanımı: Select AVG (sütun_adi) FROM Tablo_adi;

Örnek: SELECT AVG(maas) FROM personel

MAX FONKSİYONU: İçinde, belirlenen sütun içindeki en büyük değeri bulmak için kullanılır.

Kullanımı: Select MAX (sütun_adi) FROM Tablo_adi;

Select MIN (sütun_adi) FROM Tablo_adi;

Örnek: SELECT MAX(maas) FROM personel

COUNT (SAY) FONKSİYONU: Tablo içerisinde herhangi bir alanda sayma işlemi gerçekleştirir.

Kullanımı: Select COUNT (sütun_adi) FROM Tablo_adi;

Örnek: SELECT COUNT(cinsiyet) FROM personel WHERE cinsiyet='bayan'

```
SELECT COUNT(*) FROM personel
```

```
SELECT COUNT(DISTINCT adi) FROM personel
```

GROUP BY: Kümeleme fonksiyonları kullanılırken tablodaki bilgileri, bazı özelliklere göre gruplandırarak bu gruplandırılmış veri üzerinde sorgulama yapmak mümkündür. Bu işlem için, GROUP BY ifadesi kullanılır.

Örnek: SELECT cinsiyet, AVG(ort) FROM ogrenci GROUP BY cinsiyet

HAVING: Gruplandırarak, kümele fonksiyonlarını uygularken, koşul da verilebilir. Bu durumda, grup üzerindeki hesaplamalarla ilişkili koşul belirtirken, HAVING sözcüğünü kullanmak gerekir.

Örnek: SELECT adi FROM ogrenci GROUP BY adi HAVING adi LIKE 'Ali'

Örnek: Finali 50'den yüksek olup ortalama finalleri 60'a eşit ya da büyük olan öğrenci numaralarını ve final ortalamalarını listeleyin.

Select no, AVG(final) from notlar where final>50 group by no having AVG(final)>=60

TABLOLARIN BİRLEŞTİRİLMESİ

Birden fazla tablodan veri almak gerektiği durumlarda tablolar arasında ilişki kurulması gerekmektedir. Bu işleme **Join** (birleştirme) adı verilir. Join işlemi birden fazla tabloyu birbirine bağlayıp bu tablolar üzerinde işlem yapabilmemizi sağlamaktadır. Birleştirme işlemi yapabilmek için tabloların aynı değerlerini içeren sütunlarının kullanılması gerekir. Tablo birleştirme işlemi yapılırken birleştirmek istediğiniz duruma göre, Kartezyen birleşim, eşiti olan birleştirme veya eşiti olmayan birleştirme türlerinden uygun olanını kullanabilirsiniz.

1. Kartezyen Çarpımı: İki tablo arasında birleştirme koşulunun tanımlanmadığı durumlarda Kartezyen çarpımından söz edilir. Soldaki tablonun her kaydı için, sağdaki tablodan bütün kayıtları çeker. Birleştirme koşulunun geçersiz olduğu ve birinci tablodaki tüm satırların ikinci tablodaki tüm satırlarla birleşmediği durumlarda da Kartezyen çarpım elde edilir.

Kullanımı: SELECT sütun1, sütun2, sütun3,..., sütunN FROM tablo1, tablo2, tablo3,..., tabloN

Örnek:

ÜRÜNLER TABLOSU:			REYONLAR TABLOSU:	
Ürün_no	Ürün_adi	Grup_no	Grup_no	Reyon_adi
100	oys hazırlık	5	5	Kitap
101	ales hazırlık	5	10	Sebze
102	domates	10	15	Et
103	kuzu eti	15		

Ürünler tablosundaki kayıtlar ile Reyonlar tablosundaki kayıtlara Kartezyen birleşimi uygulanmış örnek aşağıda verilmiştir.

SELECT * FROM ÜRÜNLER, REYONLAR;

Ürünler tablosunda 4, Reyonlar tablosunda 3 kayıt bulunmaktadır. Bu iki tablonun birleşiminden 4x3=12 satırlık bir birleşim meydana gelecektir. Şekil 1'de 1. Tablodaki her kayıt için 2. Tablodaki tüm kayıtlar listelenmiştir.

Ürün_no	Ürün_adı	ÜRÜNLER.Grup_no	REYONLAR.Grup_no	Reyon_adı
100	oys hazırlık	5	5	Kitap
100	oys hazırlık	5	10	Sebze
100	oys hazırlık	5	15	Et
101	ales hazırlık	5	5	Kitap
101	ales hazırlık	5	10	Sebze
101	ales hazırlık	5	15	Et
102	domates	10	5	Kitap
102	domates	10	10	Sebze
102	domates	10	15	Et
103	kuzu eti	15	5	Kitap
103	kuzu eti	15	10	Sebze
103	kuzu eti	15	15	Et

Şekil 1: Kartezyen çarpımı sonucu tabloların birleştirilmesi

2. Eşiti Olan Birleştirme: İç birleştirme olarak da adlandırılan birleştirme türüdür. İç birleştirme bir sorguya, birleştirilen tabloların birinde yer alan satırların, birleştirilen alanlardaki verileri temel alarak, diğer tablodaki satırlara karşılık geldiğini bildirir. İç birleştirme içeren bir sorgu çalıştırıldığında, sorgu işlemlerine yalnızca, birleştirilen tabloların her ikisinde de bulunan ortak değere sahip olan satırlar eklenir. Birleştirmede yer alan her iki tablodan sadece, birleştirme alanında eşleşen satırlar döndürülmek istenildiği zaman iç birleştirme kullanılır. Eşiti olan birleştirme yapılırken INNER JOIN deyimi kullanılır.

Kullanımı: FROM tablo1 INNER JOIN tablo2 ON tablo1.sütun1 karşılaştırma tablo2.sütun2
Birinci tablodaki tüm kayıtları çekip, bu kayıtlar ile eşleşen ikinci tablodaki kayıtlar listelenir.

Örnek: SELECT * FROM ÜRÜNLER, REYONLAR WHERE
ÜRÜNLER.Grup_no=REYONLAR.Grup_no;

Veya

SELECT * FROM ÜRÜNLER INNER JOIN REYONLAR ON
ÜRÜNLER.Grup_no=REYONLAR.Grup_no;

Yukarıdaki örnekteki her iki yazım türü de tabloları birleştirecek ve ortak değere sahip olan kayıtlar aşağıdaki gibi listelenecektir.

Ürün_no	Ürün_adı	ÜRÜNLER.Grup_no	REYONLAR.Grup_no	Reyon_adı
100	oys hazırlık	5	5	Kitap
101	ales hazırlık	5	5	Kitap
102	domates	10	10	Sebze
103	kuzu eti	15	15	Et

3. Eşiti Olmayan Birleştirme: Eşiti olan birleştirme sırasında bir tablodaki bir sütunun içerdiği değerler diğer tablonun ilgili sütunu ile eşleştirilip sadece eşleşen değerler

birleştiriliyordu. Eşleşmeyen satırlar ise birleştirilemiyordu. Eşleşmeyen satırların da birleştirilip sonuca dâhil edilmesi istenilen durumlarda “Eşiti Olmayan Birleştirme” kullanılmaktadır. Eşiti olmayan birleştirmeler (dış birleştirmeler), eşleşmeyen kayıtların hangi tabloda olduğuna bakarak sol dış birleştirme veya sağ dış birleştirme olmak üzere iki şekilde olabilmektedir.

Kullanımı: FROM tablo1 [LEFT | RIGHT] JOIN tablo2 ON tablo1.sütun1 karşılaştırma tablo2.sütun2 Buradaki tablo1 ve tablo2 kayıtların birleştirileceği tabloların adını, sütun1 ve sütun2 birleştirilen sütunların adlarını, karşılaştırma ise ("=", "<," ">," "<=," ">=" veya "<>.") gibi işleçleri göstermektedir.

Sol dış birleştirme oluşturmak için **LEFT JOIN** kullanılır. Soldaki tablodan tüm kayıtlar alınır, sağdaki tabloda eşleşen kayıtlar yazılır ve eşleşmeyen kayıtlar için NULL değer döndürülür.

Sağ dış birleştirme oluşturmak için **RIGHT JOIN** kullanılır. Sağ dış birleşimler, ilk tablonun (soldaki tablo) kayıtlarında eşleşen değer olmasa bile, iki tablodan ikincisinin (sağdaki tablo) tüm kayıtlarını içerir. Örneğin, Bölümler (sol) ve Personel (sağ) tablolarında bölüme atanmış personel olmasa bile tüm bölümleri seçmek için LEFT JOIN, herhangi bir bölüme atanmamış olanlar da dâhil, tüm personeli seçmek için ise RIGHT JOIN kullanılır.

Örnek 1:

ÜRÜNLER TABLOSU			REYONLAR TABLOSU	
Ürün_no	Ürün_adı	Grup_no	Grup_no	Reyon_adı
100	oys hazırlık	5	5	Kitap
101	ales hazırlık	5	10	Sebze
102	domates	10	15	Et
103	kuzu eti	15	20	Çamaşır suyu
104	BALIK			

Yukarıdaki ÜRÜNLER ve REYONLAR tablosunu RIGHT JOIN kullanarak birleştirmek için aşağıdaki SQL komutu kullanılır.

```
SELECT * FROM ÜRÜNLER RIGHT JOIN REYONLAR ON ÜRÜNLER.Grup_no=REYONLAR.Grup_no;
```

Kod yazılıp çalıştırıldığı zaman görüntü aşağıdaki gibi olacaktır.

Ürün_no	Ürün_adı	ÜRÜNLER.Grup_no	REYONLAR.Grup_no	Reyon_adı
101	ales hazırlık	5	5	Kitap
100	oys hazırlık	5	5	Kitap
102	domates	10	10	Sebze
103	kuzu eti	15	15	Et
			20	Çamaşır suyu

Örnek 2: ÜRÜNLER ve REYONLAR tablosunu LEFT JOIN kullanarak birleştirmek için aşağıdaki SQL kodu yazılır.

```
SELECT * FROM ÜRÜNLER LEFT JOIN REYONLAR ON ÜRÜNLER.Grup_no=REYONLAR.Grup_no;
```


Kod yazılıp çalıştırıldığı zaman görüntü aşağıdaki gibi olacaktır.

Ürün_no	Ürün_adı	ÜRÜNLER.Grup_	REYONLAR.Grup_no	Reyon_adı
100	oys hazırlık	5	5	Kitap
101	ales hazırlık	5	5	Kitap
102	domates	10	10	Sebze
103	kuzu eti	15	15	Et
104	BALIK			

Alt Sorgular: Bazı durumlarda bir sorgudan elde edilen sonuç diğer başka bir sorgu içerisinde kullanılabilir. Bu tür durumlarda iç içe sorgular oluşturulmaktadır. Kullanılan iç içe sorgularda yer alan içteki sorgulara “alt sorgular” adı verilir.

1. Alt Sorgu Düzenleme Kuralları: Alt sorgular düzenlenirken uyulması gereken birtakım kurallar bulunmaktadır. Bunlar;

- FORM sözcüğü içinde tanımlanan sorgular dışında, alt sorgu, ana sorgu içerisindeki karşılaştırma işlecinin sağ tarafında yer almalıdır.
- Alt sorgu, parantezler içerisinde yer almalıdır.
- Alt sorgu ORDER BY sözcüğünü içermemelidir. ORDER BY sadece ana sorgu içerisinde yer alabilmektedir.

2. Alt Sorgunun Tanımlanması: Alt sorgu bir SELECT, SELECT...INTO, INSERT...INTO, DELETE veya UPDATE deyimi içinde veya başka bir alt sorguda SELECT deyiminin kullanılması ile elde edilir.

Kullanımı: SELECT liste FROM tablo WHERE ifade karşılaştırma

(SELECT liste FROM tablo)

Bir alt sorgu aşağıdaki bölümlerden oluşmaktadır:

karşılaştırma [ANY | ALL | SOME] (sqldeyimi): Karşılaştırma, ifadeyi, alt sorgunun sonuçları ile karşılaştırmaya yarayan bir karşılaştırma işlemidir.

ifade [NOT] IN (sqldeyimi): İfade, alt sorgu sonuç kümesinde aranan ifadeye verilen addır.

WHERE iki sorguyu birbirine bağlamak için kullanılmaktadır.

ANY veya SOME, yapılan karşılaştırma sonucunda alt sorgu kayıtlarından herhangi bir tanesi ile eşleşen ana sorgu kayıtlarını almak için kullanılır.

ALL, yapılan karşılaştırma sonucunda alt sorgu kayıtlarının tümüyle eşleşen ana sorgu kayıtlarını almak için kullanılmaktadır.

IN, alt sorgudaki kayıtların değerine eşit olan ana sorgu kayıtlarını almak için kullanılır.

NOT IN ise alt sorgudaki kayıtların değerine eşit olmayan ana sorgu kayıtlarını almak için kullanılır.

3. Çoklu Satır Alt Sorguları: Alt sorgudan bir satır yerine birden fazla satırın elde edildiği durumlar çoklu satır alt sorgusu olarak adlandırılır. Bu tür sorgular IN, ANY, ALL gibi işlemler yardımıyla yapılabilmektedir.

“>ANY” en azdan daha büyük, “<ANY” ise en çoktan daha az anlamına gelmektedir.

“>ALL” en büyükten daha büyük, “<ALL” ise en küçükten daha küçük anlamına gelmektedir.

Örnek IN işleci: İndirim oranı %30’dan daha fazla olan tüm ürünleri gösteren SQL kodu aşağıdaki gibidir.

```
SELECT * FROM Ürünler WHERE U_No IN (SELECT U_No FROM Sipariş WHERE İndirim = .30);
```

Örnek ANY işleci:

Birim fiyatı, % 30 veya daha fazla indirimle satılmış herhangi bir ürünün birim fiyatından yüksek olan tüm ürünleri gösteren SQL kodu aşağıdaki gibidir.

```
SELECT * FROM Ürünler WHERE B_Fiyat > ANY (SELECT B_Fiyat FROM Sipariş WHERE İndirim >= .30);
```

Örnek ALL işleci:

Birim fiyatı, % 30 veya daha fazla indirimle satılmış tüm ürünlerin birim fiyatından yüksek olan ürünleri listelemek için kullanılan SQL kodu aşağıdaki gibidir.

```
SELECT * FROM Ürünler WHERE B_Fiyat > ALL (SELECT B_Fiyat FROM Sipariş WHERE İndirim >= .30);
```

Örnek:

p_no	Adı	Bölüm_no
10	ali	200
11	veli	200
12	ahmet	50
14	mehmet	60

Yukarıdaki PERSONEL tablosuna göre Personel numarası “10” olan personelle aynı bölümde çalışan personelin isimlerini listeleyecek SQL kodu aşağıdaki gibi olacaktır.

```
SELECT Adı, Bölüm_no FROM PERSONEL WHERE Bölüm_no= (SELECT Bölüm_no FROM PERSONEL WHERE p_no=10);
```

Alt sorgu sonucunda “10” numaralı personelin çalıştığı bölümün numarası, yani “200” değeri elde edilecektir. Ana sorguda ise çalıştığı bölüm numarası “200” olan kişileri sorgulamaktadır. Sorguyu yazıp çalıştırdığımızda aşağıdaki sonuç elde edilecektir.

Adı	Bölüm_no
ali	200
veli	200

Veritabanı Temel Sorgular

Tabloya Veri Ekleme (INSERT):

Bir tabloya yeni veri eklemek için **INSERT** deyimi kullanılır. INSERT deyimi, INTO ve VALUES ifadeleri ile birlikte kullanılır ve tabloya yeni veri eklenmesi sağlanır. Veri ekleme sırasında ilk olarak INSERT INTO ile ekleme yapılacak olan tablo veya sütunlar belirlenir. Daha sonra VALUES ifadesi ile eklenecek olan değerler parantez içinde belirtilir. Verilerin sıralanışına dikkat etmek gerekmektedir. Örneğin (Personel_no, Adı, Soyadı, Bölümü) şeklinde sıralanmış olan alanlarda, eklenecek değerlerin de aynı sırada olması gerekmektedir. Tablodaki tüm alanlara değer girilecekse, tablo isminden sonra sütun isimlerinin belirtilmesine gerek yoktur. Çünkü bu tür durumda tablodaki tüm alanlara veri girişi sağlanmış olacaktır.

Sütun ismi belirtmeden kullanımı:

```
INSERT INTO tablo VALUES (değer1, değer2...)
```

Sütun ismi belirterek kullanımı:

```
INSERT INTO tablo (sütun1, sütun2...) VALUES (değer1, değer2...)
```

Mevcut kayıtlarda bulunan tek alanlardaki verileri değiştirmek için ekleme sorgusu yerine güncelleştirme sorgusu kullanılmaktadır. Ekleme sorguları sadece veri satırları eklemek için kullanılır.

Örnek: öğrenci tablosundaki adi, soyadi ve no isimli sütunlara veri ekleyen Sql kod

```
INSERT INTO ogrenci (adi, soyadi, no) VALUES ('Çağlar','Telef',125414);
```

Tablodan veri silmek (DELETE):

Bir tabloda bulunan kayıt veya kayıtların istenildiği zaman silinmesi mümkündür. Tablolarda silme işlemini gerçekleştirmek için DELETE komutu kullanılır. DELETE komutu kullanılırken FROM eki ile birlikte tablo ismi yazılarak hangi tablodan veri silinmesi istendiği belirtilebilir. WHERE deyimi ile de verilerin silinme koşulu belirlenir. Eğer WHERE ifadesi ile bir koşul

belirlenmezse tablodaki tüm kayıtlar silineceğinden WHERE kullanmaya dikkat edilmesi gerekmektedir.

Kullanımı: DELETE FROM Tablo WHERE Koşul

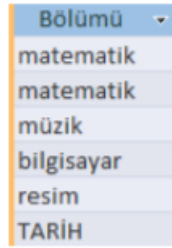
Örnek: Personel tablosunda yer alan 30 numaralı personeli silmek için yazılacak SQL komutu aşağıdaki gibi olacaktır.

```
DELETE FROM PERSONEL WHERE Personel_no = 30
```

Bu kod Personel tablosunda Personel_no=30 olan kaydı silmektedir.

DELETE FROM ogrenci; Öğrenci tablosundaki tüm verileri siler.

Örnek: Personel tablosunda Bölümü “müzik” ve “resim” olanlar dışındaki diğer kayıtların silinmesini istersek; Personel tablosunda silme işlemi uygulamadan önce Bölümü alanında bulunan kayıtlar aşağıda gösterilmiştir.



Bölümü ▾
matematik
matematik
müzik
bilgisayar
resim
TARİH

Aşağıdaki kodu yazıp çalıştırdığımızda personel tablosundaki Bölümü müzik ve resim olan kayıtlar dışındaki tüm kayıtlar silinecektir.

```
DELETE FROM PERSONEL WHERE Bölümü NOT IN ("müzik", "resim") ;
```

Güncelleme İşlemi (UPDATE):

Bir tabloda bulunan kayıt veya kayıtların istenildiği zaman değiştirilmesi mümkündür. Tablolarda güncelleme işlemini gerçekleştirmek için UPDATE komutu kullanılır. SET ifadesi ile güncellenecek alanlar ve bu alanların alacakları yeni değerler belirlenir. WHERE deyimi ile de verilerin güncelleştirilmesi için koşul belirlenir. Eğer WHERE ifadesi ile bir koşul belirlenmezse tablodaki tüm kayıtlar güncellenmiş olacağından WHERE kullanmaya dikkat edilmesi gerekmektedir.

Kullanımı: UPDATE tablo SET sütun1=değer1, sütun2=değer2,... WHERE Koşul

Örnek: Personel tablosunda yer alan 30 numaralı personelin “35” olan İl_Kodu bilgisini “06” olarak değiştirmek isteyelim.

Personel_no	adı	Soyadı	Bölümü	İl_Kodu
1	ali	ak	matematik	34
2	veli	kara	matematik	34
30	ayşe	ay	ingilizce	35

Bu işlem için yazılması gereken SQL kodu aşağıdaki şekilde olacaktır.

```
UPDATE PERSONEL SET İl_Kodu = '06' WHERE Personel_no = 30
```

ALTER KOMUTU

Tablo Yapısında Değişiklik Yapma

ALTER TABLE komutu ile bir tablonun yapısında değişiklik yapmak mümkündür.

Mevcut Bir Tabloya Kolon Ekleme

Tabloya Alan(kolon) eklemek için ALTER TABLE komutuna ADD sözcüğü eklenmelidir.

Örnek: Öğrenci tablosunun içerisine öğrencinin yaş bilgilerini de eklemek istiyorsak;

```
ALTER TABLE ogrenci ADD yas INT NOT NULL;
```

Mevcut Alan Üzerinde Değişiklik Yapmak

Mevcut bir kolon üzerinde değişiklik yapma, değişken uzunluklu bir veri tipine sahip olan kolonun genişliğini arttırma ile sınırlıdır. Bu anlamda, kolon genişliğini azaltma ya da veri tipini değiştirme mümkündür. Bu işlem için ALTER TABLE komutuna MODIFY deyiminin eklenmesi ile gerçekleştirilir.

```
ALTER TABLE ogrenci MODIFY adi CHAR(15);
```

Alan tipini değiştirmek için, ALTER TABLE komutuna ALTER COLUMN deyiminin eklenmesi gerekir.

```
ALTER TABLE ogrenci ALTER COLUMN adi INT
```

Mevcut Bir Tablodan bir kolon Silme

Tablo üzerinden alan silmek için, ALTER TABLE komutuna DROP COLUMN deyimi eklenmesi gerekir.

```
ALTER TABLE personel DROP COLUMN cinsiyet
```

Mevcut Bir Tablonun Bir Kolonunun Adını Değiştirme

Tablonun adını değiştirmek için, ALTER TABLE komutuna RENAME TABLE deyiminin eklenmesi gerekir.

```
ALTER TABLE ogrenci RENAME adi ogr_adi
```

Mevcut Bir Tablonun Tümüyle Silinmesi

Bir tablonun tamamını silmek için DROP TABLE komutu kullanılır.

DROP TABLE öğrenci; Öğrenci tablosunu siler.

Tablodaki Verilerin Silinmesi

TRUNCATE TABLE Delete komutundan bir farkı bulunmaktadır. Örnek olarak tablomuzda 25 kayıt varsa Delete komutu ile bir kayıt silinirse toplam kayıt sayısı 24 olmaktadır. Fakat yeni kayıt eklendiğinde Id si 26 olacaktır. Truncate Table komutu ile silinirse ID'ler sıfırlanmaktadır.

TRUNCATE TABLE öğrenci;

SORULAR

1) PERSONEL ve BÖLÜM tabloları aşağıda verilmiştir.

PERSONEL			BÖLÜM	
p_no	Adı	Bölüm_no	Bölüm_no	Bölüm_adi
10	ali	200	200	Bilgi İşlem
11	veli	200	50	Musasebe
12	ahmet	50	60	Halkla İlişkiler
14	mehmet	60	70	Satış

- Yukarıdaki alan ve verilerden oluşan PERSONEL ve BÖLÜM tablolarını oluşturunuz.
- PERSONEL ve BÖLÜM tablolarını Kartezyen çarpımı kullanarak birleştiriniz.
- PERSONEL ve BÖLÜM tablolarını eşiti olan birleştirme kullanarak birleştirecek SQL kodunu yazınız.
- PERSONEL ve BÖLÜM tablolarını eşiti olmayan birleştirme kullanarak birleştirecek SQL kodunu yazınız.
- PERSONEL tablosuna göre Personel numarası "12" olan personelle aynı bölümde çalışan personelin isimlerini listeleyecek SQL kodunu yazıp çalıştırınız.
- PERSONEL tablosundaki bölüm numaralarını ve personel sayısını bulan SQL kodunu yazıp çalıştırınız.
- PERSONEL tablosundaki "Mehmet" adlı kişinin bölüm numarasını ve adını bulan SQL kodunu yazıp çalıştırınız.
- PERSONEL tablosuna p_no:15, Adı:"Ayşe", Bölüm_no: 70 olan bir personel ekleyen SQL kodunu yazıp çalıştırınız.
- BÖLÜM tablosunda Bölüm_no:50 olan bölümün yanlış yazılmış olan bölüm_adi="muhasebe" olacak şekilde güncelleyen SQL kodunu yazınız.

2) Aşağıdaki tabloları uygun veri tipleri ve anahtarlarını belirleyerek oluşturup uygun veri girişleri yapınız (SQL kullanarak (insert into) birden fazla veri girişini aynı anda yaptırınız).Aşağıda verilen SQL sorgularını hazırlayıp sonuçlarına ilişkin ekran çıktılarını elde ediniz.

NOT: Her sorgu sonucunda en az iki satır veri döndürecek şekilde tablolara veri giriniz.

Tablolar:

Book (book_no, name, first-author, year, price, publisher_no,subject_no)

Student (student_no, name, department_no)

Subject (subject_no, stitle)

Department(department_no,departmentname)

Publisher (publisher_no, name)

Buys (student_no, book_no) : Öğrencilerin almış olduğu kitaplar

Covers (subject_no, book_no): Her bir kitabın hangi konulara ait olduğunu

ifade eder. Bir kitap birden fazla konu altına girebilir.

Studies (student_no, subject_no): Her bir öğrencinin hangi konularda

ilgilendiğini ifade eder.

- En pahalı 10 kitabı adlarına göre listeleyin.
- Son beş yılda her yıl en az bir kitap yayınlayan yayıncıları listeleyin.
- Ortalama kitap fiyatının üzerinde olan kitap isimlerini veriniz.
- Çalıştığı tüm konularla ilgili kitapları satın almış öğrencilerin isimlerini listeleyin.
- Kitaplar için 200 TL den fazla harcama yapan öğrencileri listeleyiniz.

- 3) Laboratuvarı PERSON ve BOLUM tabloları kullanılarak sorular sorulacaktır. Dolayısıyla bu iki tabloyu mutlaka oluşturmanız gerekmektedir.

PERSON

SNO	ADI	SOYADI	BNO	UCRETI
12	MERT	KAYA	50	3000
11	ALİ	AYDIN	50	3500
10	ASLI	AK	50	3300
13	ARİF	YILMAZ	50	5500
14	SUAT	KAYA	50	51000
15	MERAL	BAŞKAYA	50	4900
16	ESRA	ILGAZ	17	3500
17	MERVE	AKSOY	17	5100
18	KAZIM	KAYA	17	6000
19	OZAN	DEMİR	17	5600
20	NAZLI	SOLMAZ	17	6000
21	KEREM	ASLAN	18	6000
22	DURSUN	TASKESEN	18	5000
23	EMEL	MUTLU	18	3500
24	KİRAZ	YALÇIN	18	5500
25	GÜL	KAYA	18	3000
26	SENA	GÜNGÖR	18	4200

BOLUM

BNO	ADI
50	BİLİŞİM
17	MUHASEBE
18	ALT YAPI

SORU isimli bir veritabanı oluşturarak yukarıda verilen PERSON ve BOLUM isimli tabloları oluşturunuz. Verileri INSERT komutunu kullanarak giriniz.

- “Mert Kaya” adlı kişinin çalıştığı bölümün numarasını ve adını bulan SQL kodu yazınız.
- Ücreti 5000 TL’den büyük olan personelin sicil numarasını, adını ve çalıştığı bölümün adını bulan SQL kodunu yazınız.
- Çalışan personel sayısı en az 3 olan bölümlerin numaralarını bulan SQL kodunu yazınız.