YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR VE ÖĞRETİM TEKNOLOJİLERİ EĞİTİMİ BÖLÜMÜ
EBT – II DERSİ
Öğrt. Gör. Filiz Eyüboğlu

KONU-4
SQL – Structured Query language
Yapısal Sorgu Dili

Kaynak: www.sqlcourse.com

BU NOTLARIN ORİJİNALİ (İNGİLİZCE) www.sqlcourse.com ADRESİNDEDİR.
BU ADRESTE, NOTLARIN YANI SIRA SQL SORGULARINIZI ÇALIŞTIRMANIZ İÇİN BİR SQL YORUMLAYICI (Interpreter) MEVCUTTUR. BU YORUMLAYICIYI, LABORATUARDA BU NOTLARDAKİ UYGULAMA ve ÖRNEKLERİ ÇALIŞTIRMAK İÇİN KULLANACAKSINIZ.

İÇERİK:

1.KISIM

SQL NEDİR?

SQL bir veri tabanıyla iletişim kurmak için kullanılır. ANSI standardına göre ilişkisel veri tabanı yönetim sistemlerinin standart dilidir. SQL cümleleri, bir veri tabanındaki verileri güncellemek, veri tabanından veri çıkarmak, veri silmek vb için kullanılır. Çok bilinen bazı ilişkilsel veri tabanı yönetim sistemleri şunlardır: Oracle, Sybase, MS SQL Server, Access, Ingres... Her ne kadar çoğu veri tabanı yönetim sistemi SQL'i kullansa da çoğunun kendilerine özgü ek özellikleri vardır.

Standart SQL cümleleri, SELECT, INSERT, UPDATE, DELETE, CREATE ve DROP'tur.

TABLO KAVRAMI

Bir ilişkisel veri tabanı sistemi (relational data base management system) bir veya daha fazla TABLO adı verilen nesnelerden meydana gelir. Veriler bu tablolarda saklanır. Tablolar, satır ("row") ve sütunlar ("column") dan meydana gelirler ve her tablonun benzersiz ("unique") bir ismi vardır. Aşağıda örnek bir tablo görüyorsunuz.

DERSLER tablosu

Adı	Ogr. Uyesi	Sınıf	Kredi
Eğitimde Bilgi tekn.	Filiz Eyüboğlu	1	4
Programlama Dilleri	Filiz Eyüboğlu	2	4
Ogretim Tasarımı	Feza Orhan	3	2
Yazarlık Dilleri	Betül Yılmaz	3	4

Tabloda 4 satır, 4 kolon var.

VERİNİN SEÇİLMESİ

SELECT cümlesi bir veri tabanından verileri seçmek ve çıkarmak ("retrieve") için kullanılır. SELECT cümlesinin formatı:

```
SELECT "kolon1" [,kolon2", vb]
 FROM "tablo adı"
 WHERE "koşul"];
```

[ ] : seçimlik anlamına gelir.

WHERE clause'da kullanılabilecek operatörler şunlardır:

= eşittir
> büyüktür
< küçüktür
>= büyük veya eşittir

<= küçük veya eşittir
<> eşit değildir
LIKE ----------→ LIKE çok güçlü bir operatördür. Belirttiğiniz şeye benzeyenleri seçer.
Örneğin

```
SELECT isim, soyadı
   FROM calisan
   WHERE soyadi LIKE 'Er%'
```

Bu sorgu, soyadı 'Er' ile başlayan kayıtları seçer ve getirir.

Diğer bir örnek:

```
SELECT isim, soyadı
   FROM calisan
   WHERE soyadi LIKE '%s';
```

Soyadı 's' ile bitenleri seçer ve getirir.

Örnek tablo (www.sqlcourse.com)

EMPINFO (Çalışan Bilgilerini tutan tablo)

First	Last	Id	Age	City	State
John	Jones	99980	45	Payson	Arizona
Mary	Jones	99982	25	Payson	Arizona
Eric	Edwards	88232	32	San Diego	California
MaryAnn	Edwards	882338	32	Phoenix	Arizona
Ginger	Howell	98002	42	Cottonwood	Arizona
Sebastian	Smith	92001	23	Gila Bend	Arizona
Gus	Gray	22322	35	Bagdad	Arizona
MaryAnn	May	32326	52	Tucson	Arizona
Erica	Williams	32327	60	Show Low	Arizona
Leroy	Brown	32380	22	Pinetop	Arizona
Elroy	Cleaver	32382	22	Globe	Arizona

Yukarıda tablo göz önüne alındğında aşağıdaki SQL sorgularının getireceği sonuçlar nelerdir? Düşününüz. Daha sonra, yukarıda verilen adreste sayfanın sonunda yer alan pencereden SQL Yorumlayıcıya girerek sistemin vereceği sonuçları inceleyiniz.

```
SELECT first, last, city
FROM empinfo;

SELECT last, city, age
FROM empinfo
WHERE yas < 30;

SELECT first, last, city, state
FROM empinfo
```

```
WHERE last LIKE 'J%';


SELECT * FROM empinfo;

SELECT first, last
FROM empinfo
WHERE last LIKE '%s';


SELECT first, last, age
FROM empinfo
WHERE last LIKE '%illia%';


SELECT *
FROM empinfo
WHERE first = 'Eric';
```

SELECT cümlesiyle ilgili sorgular:

Sorguları yazıp SQL Yorumlayıcıya girip çalıştırınız. Sonuçları tabloya bakarak kontrol ediniz.

1. Tablodaki herkesin ilk ismini ve yaşını görüntüleyiniz.
2. Payson'dan olmayan kişilerin ilk adını, soyadını ve şehrini görüntüleyiniz.
3. 40 yaşın üzerinde olan kişilerin tüm bilgilerini (tüm kolonlar) görüntüleyiniz.
4. Soyadı 'ay' ile bitenlerin adını ve soyadını görüntüleyiniz.
5. Adı 'Mary' olanları tüm bilgilerini görüntüleyiniz.
6. Adında 'Mary' geçenlerin tüm bilgilerinin görüntüleyiniz.

Yanıtlar: İngilizce notlar, s. 3-4.

TABLO YARATMA

`CREATE TABLE` cümlesi yeni bir tablo yaratmak için kullanılır. Formatı şöyledir:

```
CREATE TABLE "tablo adı"
   ("kolon1" "veri türü"[constraint],
    "kolon2", "veri türü"[constraint],
  ..... );
```

Örnek:

```
CREATE TABLE calisan
    (adi varchar(15),
    soyadi varchar(20),
    yas number(3),
    adres varchar(30),
    sehir varchar(20) );
```

Tablo ve kolon isimleri bir harf ile başlamalıdır. Devamında ise harfler, rakamlar ve underscore karakteri " _

“ bulunabilir. Uzunluk 30 karakteri geçmemelidir. SQL özel sözcükleri (SELECT, INSERT, CREATE vb gibi) tablo ve kolon adı olarak kullanılamaz.

Bir kolona girilecek verilerle ilgili kurallar'a “**constraint**” denir. Örneğin **“unique”** constraint', tablodaki kayıtlarda bu kolona girilecek değerlerin benzersiz (“unique”) olması gerektiğini yani her hangi iki kolonda aynı değerin olamayacağı kuralını koyar. **“primary key”** constraint'i bulunduğu kolon değerinin tablodaki kayıtlara erişilirken **birincil anahtar** olarak kullanılmasını söyler. Birincil anahtar bildirimi yapıldıysa kayıtlara – sistem sıralı okuma yapmadan - doğrudan erişir. Birincil anahtar değerleri benzersiz olmalıdır.

Tablo Yaratma Uygulaması

Bir şirkette çalışanların bilgilerini tutmak için bir tablo yaratılacak. Tablodaki bilgiler şunlar olacak: adi, soyadi, unvan, yas, maas.

ÖNEMLİ: Tablonuza isim seçerken **herkesin kullanmayacağı bir isim seçiniz. CALISAN_ogrenci numaranız** gibi. Çünkü bu tablolara aynı veri tabanında yer alacaktır ve bir veri tabanındaki tablo isimleri benzersiz olmalıdır.

Yanıt: İng. Notlar, s.6

```
CREATE TABLE CALISAN_FILIZ
  (ADI VARCHAR(15),
   SOYADI VARCHAR(20),
   UNVAN VARCHAR(15),
   YAS NUMBER(2),
   MAAS NUMBER(8) );
```

TABLOYA EKLEME YAPMA (“INSERT”)

INSERT cümlesi tabloya veri eklemek için kullanılır.

```
INSERT INTO “tablo adı”
    (birinci kolon, ........, sonuncu kolon)
    VALUES(ilk değer,.........son değer);
```

Örnek:

```
INSERT INTO calisan
   (ad, soyad, yas, adres, sehir)
  VALUES ('Ayşe', 'Yılmaz', 30, 'Papatya sokak. 25/2 Kızılay', 'Ankara');
```

INSERT uygulamaları:

Yaratmış olduğunuz tabloya aşağıdaki şu 3 çalışanı ekleyiniz.

Ege Erdem, Programcı, 24, 5000
İpek Özgür, Analist, 26, 6000
Berrak Yılmaz, Sekreter, 25, 2000

Bunları ekledikten sonra **kendiniz 5 çalışan daha ekleyiniz** (yukardaki ünvanları kullanarak ve ayrıca birkaç

başka ünvan daha ekleyerek).

Bu eklemelerden sonra şu işleri yapacak SELECT cümleleri giriniz:

1- Tablodaki her çalışana ait tüm kolonları seçiniz.
2- Maaşı 5000 ve üzeri olan çalışanlara ait tüm kolonları seçiniz.
3- Yaşı 25'in üzerinde olanların ad ve soyadlarını seçiniz.
4- Ünvanı programcı olanların ad, soyad ve maaşlarını seçiniz.
5- Soyadı "r" ile bitenlerin tüm kolonlarını seçiniz.
6- Adı "İpek" olanların soyadını seçiniz.
7- Yaşı 80'in üzerinde olanların tüm kolonlarını seçiniz.
8- Adında "e" harfi geçenlerin ad ve soyadlarını seçiniz.

Yanıtlar İng. Notlar s. 7'de

KAYITLARI GÜNCELLEME ("UPDATE")

Kayıt güncelleme için UPDATE cümlesi kullanılır. Formatı şu şekildedir:

```
UPDATE "tablo adı"
SET "kolon adı" = "yeni değer" [, bir sonraki "kolon adı" = yeni
                                değer"......]
WHERE "kolon adı" OPERATOR "değer"
       [AND | OR "kolon" değer" OPERATOR "değer"];
```

Örnekler:

```
UPDATE calisan
SET yas = yas + 1
WHERE isim = 'Ayşe' AND soyadi = 'Yılmaz';

UPDATE telefon_defteri
SET alan_kodu = 212
WHERE posta_kodu = 34340;

UPDATE calisan
SET maas = 7000, unvan = 'veri tabani yöneticisi'
WHERE adi = 'ege' AND soyadi = 'Erdem';
```

UPDATE uygulamaları:

Her UPDATE'den sonra güncellemenizi teyit edecek bir SELECT cümlesi yazıp çalıştırınız.
(daha önce yarattığınız **CALISAN_....tablosu** kullanılacak)

1- İpek Özgür, Ali Yıldırım ile evlendi; soyadını güncelleyiniz.
2- Ege Erdem'in doğum günü oldu, yaşını 1 artırınız.
3- Tüm sekretelerin ünvanı "Yönetici Asistanı" oldu; güncelleyiniz.
4- 4000 ve altında maaş alanlara 500 zam yapıldı; güncelleyiniz.

**** üç update cümlesi de siz yazıp çalıştırınız.**

Yanıtlar İng. Notlar, s.8'de

KAYIT SİLME ("DELETE")

```
DELETE "tablo adı"
    WHERE "kolon adı" OPERATOR "değer"
     [ AND | OR "kolon adı" OPERATOR "değer" ];
```

Örnekler:

```
DELETE FROM calisan;
```
==➔ böyle yazıldığında tablodaki tüm satırlar silinir.

```
DELETE FROM calisan
   WHERE soyadi = 'Yılmaz';
DELETE FROM calisan
    WHERE adi = 'İpek' OR adi = 'Canan';
```

Delete uygulamaları

DELETE cümlelerinizin doğru çalıştığını görmek için SELECT cümlesi kullanınız.

1- Ege Erdem firmadan ayrıldı, kaydını siliniz.
2- 7000'in üzerinde maaş alanlar bütçe kısıtlaması nedeniyle işten çıkarıldı. Bu kişileri tablodan çıkarınız.

**** Kendiniz de iki DELETE cümlesi yazıp çalıştırınız.**

TABLO SİLME

DROP cümlesi tabloyu ve tüm kayıtları silmek için kullanılır.

```
DROP TABLE "tablo adı";
```

Drop uygulaması:

Yaratmış olduğunuz **calisan_...** tablonuzu siliniz. Silindiğinden emin olmak için bir SELECT kullanmaya çalışıp gelen mesajı inceleyiniz.

***** Birinci Kısmın Sonu *****

-

-

2. KISIM: SQL İLERİ KONULAR

items_ordered tablosu

customerid	order_date	item	quantity	price
10330	30-Jun-1999	Pogo stick	1	28.00
10101	30-Jun-1999	Raft	1	58.00
10298	01-Jul-1999	Skateboard	1	33.00
10101	01-Jul-1999	Life Vest	4	125.00
10299	06-Jul-1999	Parachute	1	1250.00
10339	27-Jul-1999	Umbrella	1	4.50
10449	13-Aug-1999	Unicycle	1	180.79
10439	14-Aug-1999	Ski Poles	2	25.50
10101	18-Aug-1999	Rain Coat	1	18.30
10449	01-Sep-1999	Snow Shoes	1	45.00
10439	18-Sep-1999	Tent	1	88.00
10298	19-Sep-1999	Lantern	2	29.00
10410	28-Oct-1999	Sleeping Bag	1	89.22
10438	01-Nov-1999	Umbrella	1	6.75
10438	02-Nov-1999	Pillow	1	8.50
10298	01-Dec-1999	Helmet	1	22.00
10449	15-Dec-1999	Bicycle	1	380.50
10449	22-Dec-1999	Canoe	1	280.00
10101	30-Dec-1999	Hoola Hoop	3	14.75
10330	01-Jan-2000	Flashlight	4	28.00
10101	02-Jan-2000	Lantern	1	16.00
10299	18-Jan-2000	Inflatable Mattress	1	38.00
10438	18-Jan-2000	Tent	1	79.99
10413	19-Jan-2000	Lawnchair	4	32.00
10410	30-Jan-2000	Unicycle	1	192.50
10315	2-Feb-2000	Compass	1	8.00
10449	29-Feb-2000	Flashlight	1	4.50
10101	08-Mar-2000	Sleeping Bag	2	88.70
10298	18-Mar-2000	Pocket Knife	1	22.38
10449	19-Mar-2000	Canoe paddle	2	40.00
10298	01-Apr-2000	Ear Muffs	1	12.50
10330	19-Apr-2000	Shovel	1	16.75

Customers tablosu

customerid	firstname	lastname	city	state
10101	John	Gray	Lynden	Washington
10298	Leroy	Brown	Pinetop	Arizona
10299	Elroy	Keller	Snoqualmie	Washington
10315	Lisa	Jones	Oshkosh	Wisconsin
10325	Ginger	Schultz	Pocatello	Idaho
10329	Kelly	Mendoza	Kailua	Hawaii
10330	Shawn	Dalton	Cannon Beach	Oregon
10338	Michael	Howell	Tillamook	Oregon
10339	Anthony	Sanchez	Winslow	Arizona
10408	Elroy	Cleaver	Globe	Arizona
10410	MaryAnn	Howell	Charleston	South Carolina
10413	Donald	Davids	Gila Bend	Arizona
10419	Linda	Sakahara	Nogales	Arizona
10429	Sarah	Graham	Greensboro	North Carolina
10438	Kevin	Smith	Durango	Colorado
10439	Conrad	Giles	Telluride	Colorado
10449	Isabela	Moore	Yuma	Arizona

SELECT cümlesi

SELECT cümlesinin formatı:

```
SELECT [ALL | DISTINCT] column1[,column2]
FROM table1[,table2]
[WHERE "conditions"]
[GROUP BY "column-list"]
[HAVING "conditions]
[ORDER BY "column-list" [ASC | DESC] ]
```

Örnek:

```
SELECT isim, yas, maas
FROM calisan
WHERE yas > 50;
```

Karşılaştırma operatörleri:

=	Equal
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<> or !=	Not equal to
LIKE	String comparison test

ALL ve DISTINCT anahtar kelimeleri

ALL: tüm kayıtları seçer, "default" olduğu için belirtmemiz gerek yoktur.

DISTINCT: unique (benzersiz) kolon değerlerini seçmek için.

Örnek:

```
SELECT DISTINCT yas
FROM calisan;
```

Alıştırmalar

Alıştırmaları yaparken sistemde mevcut ve yukarıda içerikleri verilmiş **"items-ordered**" ve **"customers"** tablolarını kullanacaksınız.

"items-ordered" tablosundan

1- customerid'si 10449'in satın aldığı ürünleri bulunuz; customerid, item ve price'ı listeleyiniz.
2- "tent" alanların tüm kolonlarını listeleyiniz.
3- "s" ile başlayan item'lara ait customer_id, order_date ve item values listelenecek.
4- distinct item'ları listeleyiniz.

******** Ayrıca 4 SELECT cümlelesi daha yazıp çalıştırınız.**

Yanıtlar: İng. Notlar, s. 14

Kümeleme Fonksiyonları ("Aggregate Functions")

MIN	returns the smallest value in a given column
MAX	returns the largest value in a given column
SUM	returns the sum of the numeric values in a given column
AVG	returns the average value of a given column
COUNT	returns the total number of values in a given column
COUNT(*)	returns the number of rows in a table

Örnekler

```
SELECT AVG(maas)
FROM calisan;
SELECT AVG(maas)
FROM calisan
WHERE unvan = 'programcı';

SELECT COUNT(*)
FROM calisan;
```

Alıştırmalar

1- items-ordered tablosunda fiyatı en yüksek olan eşyayı bulunuz.
2- aralık ayında satılmış eşyaların ortalama fiyatını bulunuz.
3- items_ordered tablosunda kaç satır vardır?
4- sipariş edilen "tent"ler arasında en düşük fiyatı olanı bulunuz (en düşük fiyatı görüntüleyiniz).

Yanıtlar: İng. Notlar, s.15

GROUP BY yantümcesi

Sözdizimi:

```
SELECT column1,
SUM(column2)
FROM "list-of-tables"
GROUP BY "column-list";
```

The GROUP BY clause, belirtilen kolon veya kolonlardaki tüm satırları bir araya toplar ve bu kolonlar üzerinde "aggregate" fonksiyonlarının çalıştırılmasını izin verir. Örneğin, şirkkte her bölümdeki en yüksek maaşı bulmak istiyorsak, bölümlere göre gruplayıp, her bir grubun içinde en yüksek maaşı bulmamız gerekir.

```
SELECT max(salary), dept
FROM employee
GROUP BY dept;
```

This statement will select the maximum salary for the people in each unique department. Basically, the salary for the person who makes the most in each department will be displayed. Their, salary and their department will be returned.

items_ordered tablosunu göz önüne alalım. **Soru**: Quantity'ye göre gruplama yapıp (yani quantity 1 olanlar bir grup, quantity 2 olanlar bir grup, quantity 3 olanlar başka bir grup...) her gruptaki en yüksek maliyeti bulmak istiyoruz.

```
SELECT quantity, MAX(price)
FROM items_ordered
GRUP BY quantity;
```

Bu sorguyu girip sonucuna bakınız, notlarınızdaki tablodan kontrol ediniz.

Alıştırmalar

1- customers tablsounu kullanarak, her bir eyalette kaçar müşteri olduğunu bulunuz. *Hint:* ***count*** *is used to count rows in a column,* ***sum*** *works on numeric data only.*
2- items_ordered tablosundan, her bir eşya için item, max price, min price'ı bulunuz.
3- her bir müşterinin kaç sipariş verdiğini bulunuz. items_ordered tablosunu kullanınız. customerid, bu müşterinin yaptığı sipariş sayısı ve siparişlerinin toplamını bulunuz.

Yanıtlar: İng. Notlar, s. 17

HAVING yantümcesi (" HAVING clause")

HAVING yantümcesi, belirli bir koşula göre grup oluşturmayı sağlar. Bu bakımdan HAVING clause, GROUP BY'dan sonra yer almalıdır.

HAVING yantümcesinin formatı:

```
SELECT column1,
SUM(column2)
FROM "list-of-tables"
GROUP BY "column-list"
HAVING "condition";
```

Bir örnekle açıklamaya çalışalım. Çalışanların ad, bölüm, maaş ve yaşını tutan bir çalışan tablosu olsun. Her bir bölümde çalışan kişilerin ortalama maaşlarını bulmak istiyorsak şöyle yazarız:

```
SELECT dept, avg(salary)
FROM employee
GROUP BY dept;
```

Fakat, gruptaki tüm maaşların ortalamasını almak yerine, maaşı 20000'den fazla olanların ortalamasını istiyorsak HAVING kullanmamız gerekir:

```
SELECT dept, avg(salary)
FROM employee
GROUP BY dept
HAVING avg(salary) > 20000;
```

Uygulamalar (GROUP BY alıştırmalarına benziyor bunlar; ama HAVING kullanılmayı gerektiriyor)

1- customers tablosunu kullanarak her eyalette kaç kişi olduğunu bulmamız isteniyor; ancak kişi sayısı o eyalette 1'den fazla ise.
2- items_ordered tablosunu kullanarak, tablodaki her bir eşya için (max price > 190.00 ise) eşya, max price, min price görüntüleyiniz.
3- Her müşterinin sipariş (order) sayısını bulunuz (eğer birden fazla eşya sipariş ettiyse). items_ordered tablosunu kullanınız. customerid, siparişlerinin sayısı ve siparişlerinin toplamını görüntüleyiniz

Yanıtlar: İng. Notlar, s. 18

ORDER BY yantümvesi

ORDER BY sorgunuzun sonucunun sıralı olarak (büyükten küçüğe veya küçükten büyüğe) görüntülenmesi için kullanılabilecek seçimlik ("optional") bir yantümcedir.

ORDER BY yan cümlesinin formatı:

```
SELECT column1, SUM(column2)
FROM "list-of-tables"
ORDER BY
```

```
"column-list" [ASC | DESC];

[ ] = optional


ASC = Ascending Order - default
DESC = Descending Order
```

Örnek:

Employee_info tablosundan, dept = 'Sales' olan bölümlerde çalışanların employee_id, dept, name, age ve salary bilgilerini maaşa göre küçükten büyüğe doğru sıralayarak görüntüleyiniz.

```
SELECT employee_id, dept, name, age, salary
FROM employee_info
WHERE dept = 'Sales'
ORDER BY salary;
```

Sıralamayı birden fazla kolona göre yapmak isterseniz kolon adlarını virgülle ayırarak yazınız. Örnek olarak:

```
SELECT employee_id, dept, name, age, salary
FROM employee_info
WHERE dept = 'Sales'
ORDER BY salary, age DESC;
```

Alıştırmalar

1- customers tablosundan, tüm müşterilerin lastname, firstname ve city bilgisini seçiniz; soyadına göre küçükten büyüğe doğru sıralayarak görüntüleyiniz.

2- Birinci uygulamanın aynısı...bu sefer soyadını büyükten küçüğe doğru sıralayınız.

3- fiyatı 10.00'den büyük olan eşyaların adını ve fiyatını seçip fiyata göre küçükten büyüğe doğru sıralayınız.

Yanıtlar: İng. Notlar, s.19

Koşulların ve Boolean Operatörlerin Birleştirilmesi

Örnek:

```
SELECT customerid, order_date, item
FROM items_ordered
WHERE (item <> 'Snow shoes') AND (item <> 'Ear muffs');
```

Note: Yes, that is correct, you do want to use an AND here. If you were to use an OR here, then either side of the OR will be true, and EVERY row will be displayed. For example, when it encounters 'Ear muffs', it will evaluate to True since 'Ear muffs' are not equal to 'Snow shoes'.

Örnek:

```
SELECT item, price
FROM items_ordered
WHERE (item LIKE 'S%') OR (item LIKE 'P%') OR (item LIKE 'F%');
```

IN ve BETWEEN Koşul Operatörleri

```
SELECT col1, SUM(col2)
FROM "list-of-tables"
WHERE col3 IN
       (list-of-values);

SELECT col1, SUM(col2)
FROM "list-of-tables"
WHERE col3 BETWEEN value1
AND value2;
```

IN koşul operatörü, küme üyeliğini sınamak için kullanılır. Yani, bir değerin bir kümeye ait olup olup olmadığını sınamak için kullanılır.

Örnek:

```
SELECT employeeid, lastname, salary
FROM employee_info
WHERE lastname IN ('Hernandez', 'Jones', 'Roberts', 'Ruiz');
```

Bu cümle employee_info tablosundan employeeid, lastname, salary bilgilerini listeleyecek eğer isim parantez içindeki isimlerden birine eşitse.

IN yerine eşit ve OR kullanarak aynı sorguyu şu şekilde yazabiliriz.

```
SELECT employeeid, lastname, salary
FROM employee_info
WHERE lastname = 'Hernandez' OR lastname = 'Jones' OR lastname =
'Roberts'
OR lastname = 'Ruiz';
```

Görüldüğü gibi IN kullandığında daha kısa ve anlaşılır yazmak mümkün oluyor.

DİKKAT: Satırları dahil etmemek için de **NOT IN** kullanılır. .

BETWEEN kolul operatörü BETWEEN'den önce belirtilen değerin BETWEEN'den sonra belirtilmiş olan iki değerin arasında olup olmadığını sınamak için kullanılır

Örnek:

```
SELECT employeeid, age, lastname, salary
FROM employee_info
WHERE age BETWEEN 30 AND 40;
```

Bu cümle, employee_info tablosundan – yas 30 ile 40 arasındaysa - employeeid, age, lastname ve salary'yi listeler. Aynı sonucu verecek diğer bir cümle şöyle olabilir:

```
SELECT employeeid, age, lastname, salary
FROM employee_info
WHERE age >= 30 AND age <= 40;
```

DİKKAT: Verdiğimiz aralığın dışında olanları belirtmek için NOT BETWEEN kullanabiliriz.

Alıştırmalar

1- items_ordered tablosunu kullanarak, fiyatı 10.00 ile 80.00 arasında olan eşyaların adını ve fiyatını görüntüleyiniz.
2- customers tablosundan, eyaleti Arizona, Washington, Oklahoma, Colorado veya Hawaii olan müşterilerin ilk adını, şehirini ve eyaletini görüntüleyiniz.

Y. s. 22

Aritmetik Operatörler

ANSI SQL-92 standardı 4 temel aritmetik operatörü destekler.

+	addition
-	subtraction
*	multiplication
/	division
%	modulo

Modulo operatörünü ANSI SQL desteklemez ancak çoğu veri tabanında kullanılır.

Aşağıda çok yararlı bazı matematik fonksiyonlarını bulabilirsiniz. Bunlar ANSI SQL-92 standardında yoktur; kullandığımız VYTS'de yer alabilir, almayabilir.

ABS(x)	returns the absolute value of x
SIGN(x)	returns the sign of input x as -1, 0, or 1 (negative, zero, or positive respectively)
MOD(x,y)	modulo - returns the integer remainder of x divided by y (same as x%y)
FLOOR(x)	returns the largest integer value that is less than or equal to x
CEILING(x) or CEIL(x)	returns the smallest integer value that is greater than or equal to x
POWER(x,y)	returns the value of x raised to the power of y
ROUND(x)	returns the value of x rounded to the nearest whole integer
ROUND(x,d)	returns the value of x rounded to the number of decimal places specified by the value d
SQRT(x)	returns the square-root value of x

Örnek:

```
SELECT round(salary), firstname
FROM employee_info
```

This statement will select the salary rounded to the nearest whole value and the firstname from the **employee_info** table.

Alıştırma

-

1- **items_ordered** tablosoundan her eşya için eşya adını ve eşyanın ortalama **birim** fiyatını görüntüleyiniz. İpucu: Fiyatı miktara bölünüz.

Yanıt:

```
select item, sum(price)/sum(quantity)
from items_ordered
group by item;
```

Tablo Birleştirme İşlemi ("Table Joins") – bir gereklilik

Şu ana kadar yazdığımız tüm sorgularda tek bir tablo kullandık. Şimdi, ilişkisel veri tabanlarının ve SQL'in çok önemli ve yararlı bir özelliğini göreceğiz: Birleştirme ("join") işlemi. İlişkisel veri tabanlarına "ilişkisel" denmesinin nedeni olan işlem. Birleştirme işlemi iki veya daha çok tablo arasında bağlantı ("link") kurarak tek bir tablo oluşturur ve sırgu bu tablo üzerinde çalışır *(Joins allow you to link data from two or more tables together into a single query result--from one single SELECT statement).*

SELECT cümlesinde birden fazla tablo adı geçiyorsa bu sorguda Join işlemi olacağı anlaşılır.

Örneğin:

```
SELECT "list-of-columns"
FROM table1,table2
WHERE "search-condition(s)"
```

Birleştirme işlemi en kolay eğer bir tabloyla çalışırken ve join yeteneği yokken neler olabileceğini göstererek açıklanabilir. Diyelim ki tüm müşterilerinizi ve mağazanızdan neler aldıklarını gösteren tek bir tablonuz var. Tablonun sütunları aşağıdaki gibi olsun:

id	first	last	address	city	state	zip	date	item	price

Tabloya her yeni satır ekleyişte tüm kolonlara bilgi girilmek durumdadır; bu durumda veri tekrarı ("redundant data") oluşacaktır. **Örneğin,** Wolfgang Schultz'un her satınalma yapışında adı, soyadı, adresi şehri vb girilmek zorunda kalacaktır

id	first	last	address	city	state	zip	date	item	price
10982	Wolfgang	Schultz	300 N. 1st Ave	Yuma	AZ	85002	032299	snowboard	45.00
10982	Wolfgang	Schultz	300 N. 1st Ave	Yuma	AZ	85002	082899	snow shovel	35.00
10982	Wolfgang	Schultz	300 N. 1st Ave	Yuma	AZ	85002	091199	gloves	15.00
10982	Wolfgang	Schultz	300 N. 1st Ave	Yuma	AZ	85002	100999	lantern	35.00
10982	Wolfgang	Schultz	300 N. 1st Ave	Yuma	AZ	85002	022900	tent	85.00

Bu mağaza için ideal bir veri tabanında iki tablo olabilir:

1- Bir tablo, müşteri bilgilerini tutmak için
2- Diğeri, müşterilerin neler satınaldıklarını tutmak için.

Bu tabloların kolonları şöyle olacaktır:
"Customer_info" table:

customer_number	firstname	lastname	address	city	state	zip

"Purchases" table:

customer_number	date	item	price

Bu durumda, bir müşteri bir şey satınaldığında sadece "purchases" tablosu güncellenecektir. Ve veri tekrarı önelnemiş olacaktır. Veri tekrarını engelleyecek şekilde tabloların düzenlenmesine **"veri tabanının normalize edilmesi" denir ("normalization").** *Değişik normalizasyon düzeyleri vardır ancak bunlar bu ders kapsamında ele alınmayacaktır.*

Her iki tabloda da ortak bir kolon olduğuna dikkat ediniz: "customer_number" kolonu. Benzersiz müşteri numaralarını içeren bu kolon iki tabloyu birleştirmek (**JOIN**) için kullanılır. Bu iki tabloyu kullanarak müşteri adını ve ne satın aldığını bulmak istediğimizi düşünelim. Aşağıda bu sorguyu başaracak birleştirme cümlesini görüyorsunuz:

```
SELECT customer_info.firstname, customer_info.lastname, purchases.item
FROM customer_info, purchases
WHERE customer_info.customer_number = purchases.customer_number;
```

Bu tarz birleştirmeye "Inner Join" veya "Equijoin" denir. Göreceğiniz ve kullanacağınız birleştirmelerin hemen hepsi bu tip olacaktır.

Kolon isimlerinden önce ait oldukları tablo isimleri ve bir nokta görüyorsunuz. Tablolardaki kolon isimleri benzersiz ise bu yazış şekline (yani tablo adını belirtmeye) gerek yoktur, ancak sorgunun anlaşılabilir olması bakımından tablo isimlerini bildirmek iyi bir alışkanlıktır.
=➔ JOINlerde kolon isimlerinin ait oldukları tabloları belirtmeniz kesinlikle tavsiye edilir.

Not: Yukarda verilmiş olan sözdizimi ("syntax") pek çok veri tabanı sisteminde ve lab'da kullanacağınız yorumlayıcıda bu şekilde çalışır. Ancak başka bir sistemde başka söz dizim kuralları olabilir.

Although the above will probably work, here is the ANSI SQL-92 syntax specification for an Inner Join using the preceding statement above that you might want to try:

```
SELECT customer_info.firstname, customer_info.lastname, purchases.item
FROM customer_info INNER JOIN purchases
ON customer_info.customer_number = purchases.customer_number;
```

Diğer bir örnek:

```
SELECT employee_info.employeeid, employee_info.lastname,
       employee_sales.comission
FROM employee_info, employee_sales
WHERE employee_info.employeeid = employee_sales.employeeid;
```

Bu cümle,
employee_info tablosundaki employeeid = employee_sales tablosundaki employeeid
olan satırlar için

employee_info tablosundan employeeid, lastname

employee_sales tablosundan comission value'yu seçecek ve görüntüleyecektir.

Alıştırmalar

1. Her bir müşteri tarafından sipariş edilmiş eşyaları listeleyecek bir sorgu yazınız. Listede kolon başlıkları: customerid, firstname, lastname, order_date, item ve price olacak.

2. 1. alıştırmayı, görüntülenecek listede customerid, firstname, item ve state (eyalet bilgisi büyükten küçüğe olacak şekilde) tekrarlayınız.

Answers, p. 25

SQL Interpreter

Note: This SQL Interpreter/Database on this site will only support the commands covered on this tutorial - specifically, the SELECT statement. If you are interested in using the other SQL commands, please go to: sqlcourse.com. This site will support all of the advanced SELECT statement options that are not supported on the original sqlcourse.com site.