SQL NEDİR?

***SQL*, İngilizce adıyla "Structured Query Language" (Yapılandırılmış Sorgulama Dili) bir veri tabanı sorgulama dilidir. SQL ile veri tabanına yeni tablolar, kayıtlar ekleyip silebilir, mevcut veriler üzerinde düzenlemeler ve sorgular yapılabilir.** SQL ile ORACLE, db2, Sybase, Informix, MS SQL Server, MS Access gibi veri tabanı yönetim sistemlerinde çalışılabilir. SQL, standart bir veri tabanı sorgu dilidir, bütün gelişmiş veri tabanı uygulamalarında kullanılır.

Transact-SQL (T-SQL)

SQL, düzeltilmesi veya değiştirilmesi istenen bilgileri açıkça belirtmeye izin veren ve yerine getirilebilecek başlıca işlemleri tanımlamamızı sağlayan bir komut takımıdır. Bu komutların oluşturduğu yapıya *T-SQL* dili denir. T-SQL ile veri ve sorgulara erişebilir, güncelleyebilir ve ilişkisel veri tabanı sistemi yönetilebilir. T-SQL komutları kullanım amaçlarına göre üç genel kategoriye ayrılır.

Microsoft'un veri tabanı sorgulama dilidir. Transact-SQL, SQL Server ve istemci (client) arasında iletişimi sağlayan SQL sorgulama dilinin gelişmiş bir versiyonudur. Transact Structured Query Language kelimelerinin kısaltmasıdır. T-SQL kullanarak veri tabanına kayıt eklenebilir, silinebilir, güncellenebilir ya da sorgulama ve raporlama yapılabilir. T-SQL ile döngü veya mantıksal işlemler yapmak için bir derleyiciye gerek yoktur.

Bu komutlar, işlevlerine göre şu şekilde ayrılır;
**** DDL (Data Definition Language): Veri tanımlama dili**
**** DML (Data Manipulation Language) : Veri işleme dili**
**** DCL (Data Control Language): Veri kontrol dili**

SQL Veri Tanımlama Dili (Data Definition Language – DDL)

SQL Veri tanımlama dili verilerin tutulduğu nesneler olan tabloların yaratılmasını, silinmesini ve bazı temel özelliklerinin düzenlenmesini sağlar. En sık kullanılan bazı DDL komutları ve kullanım amaçları aşağıdaki gibidir:
CREATE TABLE:Yeni bir tablo yaratmak
ALTER TABLE:Tabloda değişiklik yapmak
DROP TABLE:Tabloyu silmek
CREATE INDEX:Tabloda dizin oluşturmak

SQL Veri İşleme Dili (Data Manipulation Language – DML)

SQL veri işleme dili veri girmek, değiştirmek, silmek ve verileri almak için kullanılan DML komutlarının tümüdür. En sık kullanılan DML komutları ve kullanım amaçları aşağıdaki gibidir:
SELECT:Veri seçmek
DELETE:Veri silmek
UPDATE:Veri güncellemek
INSERT:Veri girmek

SQL Veri Kontrol Dili (Data Control Language – DCL)

SQL veri kontrol dili bir veri tabanı kulalnıcısı veya rolü ile ilgili izinlerin düzenlenmesini sağlar. DCL komuları ve fonksiyonları şöyledir:
GRANT:Kullanıcıya yetki verir.
DENY:Kullanıcı, grup veya rolü herhangi bir eylem için eneller.
REVOKE:Daha önce atanmış olan yetki veya engeli kaldırır.

SQL Kullanımı Hakkında

SQL dili veritabanlarındaki tablolara bağlanıp veri eklemek, daha önceden eklenen verileri okumak ve veriler üzerinde işlem yapmak (güncelleme, silme) için kullanılan bir dildir. Sorgulama yapabilmek için öncelikli olarak veritabanımızda ez azından bir tablo olması gerekmektedir. Birden fazla tablo da olabilir. Ancak ister bir tablo olsun isterse birden çok tablo olsun neticede üzerinde işlem yapmak istediğimiz tabloyu belirtmemiz gerekmektedir. Hangi tablo üzerinde çalışacağımız ***FROM*** ifadesi ile belirtiriz. Bu ifade SQL sorgulama dilinin temelidir ve hemen hemen bütün SQL sorgularında kullanılmaktadır. Hemen hemen dedik çünkü bazı sorgu ifadelerinde kullanılmaz. Mesela yeni bir veritabanı veya tablo oluştururken FROM ifadesi kullanılmaz. FROM ifadesi sadece var olan verilerde sorgulama yaparken kullanılır.

Tek başına FROM ifadesi tabiki yetmez. FROM ile hangi tablo üzerinde işlem yapacağımız belirtiriz ancak ilgili tabloda ekleme işlemi mi, silme işlemi mi veya güncelleme işlemi mi yaptığımızı da belirtmemiz gerekmektedir. Bunun için belli başlı işlemler için hangi özel komutun kullanıldığını aşağıdaki tabloda görebilirsiniz.

Komut	**İşlemi**
Select	Veritabanından verileri okumak için kullanılır.
Update	Veritabanındaki var olan bir kaydın bilgilerini güncellemede kullanılır. Örneğin yaşadığı şehir alanı İstanbul olanları Ankara'ya çeviremek gibi.
Delete	Veritabanındaki bir kaydı silme için kullanılır.
Insert Into	Veritabanına yeni bir kayıt eklemek için kullanılır.
Create Database	Yeni bir veritabanı oluşturmak için kullanılır.
Alter Database	Mevcut veritabanı özelliklerinde değişiklik yapmak için kullanılır.
Create Table	Veritabanına yeni bir tablo eklemek için kullanılır.
Alter Table	Mevcut tablo özelliklerinde işlem yapmak için kullanılır. Bu komut ile mevcut kayıtlar üzerinde işlem yapılmaz.
Drop Table	Veritabanı içindeki bir tabloyu silmek için kullanılır. Bu işlem ile ilgili tablodaki kayıtlar da silinir.

Örnek1:

Select * From Personel (Personel tablosundaki bütün kayıtları çeker.)
Select Ad, Sehir From Personel (Personel tablosunda çok sayıda alan olabilir. Ancak bu komutla sadece Ad ve Sehir alanlarındaki kayıtları çekeriz.)

Yukarıdaki örnekte görüleceği üzere öncelikli olarak hangi işlemin yapılacağı belirtiliyor. Daha sonra ise hangi alanlar için yapılacağı belirtiliyor ve son olarak hangi tablo üzerinde yapılacağı belirtiliyor.

Örnek2:

```
Select * From Personel Where Sehir='İstanbul'
Select * From Personel Where Sehir='İstanbul' and Bolum='Bilgisayar'
```

Birinci önekte Personel tablosundan Sehir alanında İstanbul yazan kayıtlar seçilmektedir. Tabloda 100 tane kayıt olabilir. Ancak bunlardan 20 tanesinin Sehir alanında İstanbul yazıyorsa, yazılan bu komut ile sadece 20 tane kayıt seçilip ekrana yazdırılabilir. İkinci örnekte ise Perosnel tablosundaki kayıtlarda Sehir alaninda İstanbul yazan ve Bolum alanında Bilgisayar yazan kayitler seçilip ekrana yazdırılabilir.

Komutların detaylı kullanımları ile ilgili bilgileri sol taraftaki menüden seçerek görebilirsiniz. Bu bölümde temel olarak SQL yapısının nasıl işlediğini anlamanız yeterli olacaktır.

SQL Genel Data Tipleri

Tablo üzerindeki her bir alanın bir veri tipi vardır. Veritabanını tasarlarken hangi alanın hangi veri tipinde olduğunu belirlemek oldukça önemlidir. Daha sonra bir alandan veri çekmek istenirse alanın tipine göre veri çekmek işlemi gerekecektir.

Aşağıdaki listede genel olarak kullanılan data tiplerini görebilirsiniz.

Data Tipi	Açıklama
CHARACTER(n)	Metin tipinde n ile belirtilen sayı kadar karakter depolar. n ile belirtilenden az karakter girilmiş olsa bile n kadar yer kullanılır.
VARCHAR(n) or CHARACTER VARYING(n)	En fazla n ile belirtilen kadar karakteryer kaplar. Girilen veri n değerinden az ise sadece girildiği kadar alan kaplar.
BINARY(n)	Metinsel olarak Binary değerleri n ile belirtilen kadar depolar. n değerinden az karakter girilse de n kadar yer kullanılır.
BOOLEAN	TRUE - FALSE , VAR - YOK gibi sadece iki durumun varlığında kullanılır.
VARBINARY(n) or BINARY VARYING(n)	Metinsel olarak Binary değerleri en fazla n ile belirtilen kadar depolar. Veri n değerinden az ise sadece girildiği kadar alan kaplar.
INTEGER	Tam sayı olarak numarasal değerleri depolar. Virgüllü değerleri kabul etmez. 4 byte alan kaplar. -2.147.483.648 ile 2.147.438.647 arası değerlerde kullanılır.
SMALLINT	Küçük tam sayı değerlerini depolar. Virgüllü değerleri kabul etmez. 2 byte yer kaplar. -32.768 ile 32.767 arası değerlerde kullanılır.
TINYINT	Mini tam sayı değerlerini depolar. Virgüllü değerleri kabul etmez. 1 byte yer kaplar. 0-255 arası değerlerde kullanılır.
BIGINT	Büyük tam sayıları depolamak için kullanılır. Virgüllü değerleri kabul etmez. 8 byte alan kaplar. -2.147.483.648 ile 2.147.438.647 arası değerlerde kullanılır. -9.223.372.036.854.775.808 ile -9.223.372.036.854.775.807 arası değerlerde kullanılır.
DECIMAL(p,s)	Virgüllü değerlerde kullanılır.. Örneğin: decimal(5,2) ifadesinde toplam 5 tane rakam olduğunu söylenmiş. Bunun 2 tanesi virgülden sonra olaak belirtilmiş. Tam sayı kısmına ise 3 rakam girilebilir.
NUMERIC(p,s)	DECIMAL veri tipi ile aynı özelliklere sahiptir. Bazı sistemler sadece Numeric alanı kullanır.
FLOAT(p)	Virgüllü değerleri yuvarlayarak kayıt eder. p ile hafızada tutulmak istenen byte değeri belirtilir. girilen değer 5 byte'tan fazla ise sadece 5 byte'lık kısmı yuvarlanarak kaydedilir.
REAL	4 byte yer tutar. Aynı zamanda Float(24) ile aynı işi yapar. Ancak bu tipte virgüllü sayıları aklayabilirsiniz.
DATE	Yıl, ay, gün olarak tarih depolar
TIME	Saat, dakika ve saniye olarak veri depolar

TIMESTAMP	Yıl, ay, gün, saat, dakika ve saniye olarak veri depolar
MULTISET	A variable-length and unordered collection of elements
XML	XML tipinde verileri depolar.

Çeşitli Veritabanlarında Kullanılan Veri Tipleri

Microsoft Access, MySQL ve SQL Server'da kullanılan veri tipleri

Microsoft Access Data Types

Data Tipi	Açıklama	Boyut
Text	Metinsel verileri depolar. Girilen sayıları da metinsel olarak depoalr. En fazla 255 karakter alır.	
Memo	255 karakterden fazla bilgi depoalamak istenildiğinde kullanılır. 65,536 karakter alır. **Not:** Bu alanda alfabetik sıralama yapılamaz. Ancak arama işlemleri yapılabilir.	
Byte	0 ile 255 arası tam sayılar için kullanılır. Virgüllü sayılar olmaz.	1 byte
Integer	-32,768 ve 32,767 arasındaki tam sayılar için kullanılır. Virgüllü sayılar olmaz.	2 bytes
Long	-2,147,483,648 and 2,147,483,647 arasındaki tam sayılar için kullanılır. Virgüllü sayılar olmaz.	4 bytes
Single	Tek basamaklı virgüllü ifadeler için kullanılır.	4 bytes
Double	Çok basamakalı virgüllü ifadeler için kullanılır.	8 bytes
Currency	Para birimi için kullanılır. 15 basamaklı tam sayı ve 4 basamaklı kuruş verisi saklanabilir. **Not:**Hangi ülke para birimini kullanacağınızı seçebilirsiniz.	8 bytes
AutoNumber	Otomatik numara alanıdır. Her kayıt eklendiğinde numara 1 artar. Genel olarak 1'den başlar	4 bytes
Date/Time	Tarih ve Saat verilerini saklamak için kullanılır.	8 bytes
Yes/No	Evet/Hayır, Var/Yok, veya Açık/Kapalı gibi mantıksal verilerde kullanılır. Kodlamada True veya False ifadeleri ile sorgulanabilir. **Not:** Bu alan boş olarak geçilemez.	1 bit
Ole Object	Resim, ses, video gibi iki taban dosyalarını depolamak için kullanılır.	1 GB'a kadar
Hyperlink	Web sayfaları dahil olmak üzere diğer dosyalara bağlantılar içerir.	

SQL COMMANDS :

Select

Distinct

Where

And Or

Order by

Insert into

Update

Delete

Select Top

Like

In

Between

As

Inner Join

Left Join

Union

Select Into

Insert Into

Create Database

Create Table

Constraints

Nut Null

Unique

Primary Key

Defaults

Isnull , Ifnull , Coalesce

Avg

Count

First

Last

Max

Min

Sum

Group by

SQL Matematiksel Fonksiyonlar

ABS()

CEILING()

FLOOR()

PI()

POWER()

RAND()

ROUND()

SIGN()

SQRT()

SQUARE()

ISNUMERIC()

SIN

COS

TAN

COT

EXP

LOG , LOG10

4 işlem : + , - , *, /

SQL SELECT Kullanımı

SELECT ifadesi veritbanından verileri okumak için kullanılır. Veritbanındaki tablo üzerindeki bütün alanlardan veri çekilebileceği gibi belirteceğimiz bir kaç alandan da veri çekebiliriz. En azındna bir alan belirtmek gerekmektedir.

Select Kullanım Biçimi
Eğer bütün alanlardan veri çekeceksek, Select ifadesinden sonra * işareti konulur. Böylece bütün alanlardan veri çekileceği belirtilmiş olunur.

```
SELECT *
FROM tablo_adi
```

Eğer birkaç tane alanlardan veri çekeceksek, Select ifadesinden sonra alanları virgüller ayırarak yazarız.

```
SELECT alan_adi1, alan_adi2
FROM tablo_adi
```

Örnek Tablo Uygulaması:
Örnek olarak aşağıdaki gibi Personel isimli tablomuz olsun.

id	Adi_soyadi	Sehir	Bolum	Sicil No
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	Erzincan	Bilgi İşlem Sorumlusu	3456789

Örnek1:
SELECT * FROM Personel (Personel tablosundaki bütün kayıtları çeker.)
Bu kod ile tablodaki bütün alanlarda bulunan kayıtları seçmiş oluruz. Daha sonra bunu kullandığımız programlama ortamında ekrana yazdırırız.

Örnek2:
SELECT Adi_soyadi, Bolum FROM Personel
Bu örnekte Personel tabosundan sadece adı soyadı ve bölüm bilgisinin tutulduğu alanlar seçilmektedir. Diğer alanlar işleme alınmaz. Kullandığımız programlama ortamına kayıtları aktarırken sadece bu iki alan dikkate alınacaktır. Eğer kullandığımız programlama ortamında Şehir bilgisine erişmek istersek hata alırız. Bütün programlama ortamlarında (ASP, PHP, .Net vb.) sorgu sonucu elde edilen veriler üzerinde ilk kayda gitme, önceki kayda gitme, sonraki kayda gitme ve en son kayda gitme gibi özel tanımlı komutlar vardır. Bu komutlar sayesinde Select ile elde edilen veriler üzerinde işlem yapılabilir.

SQL DISTINCT Kullanımı

DISTINCT ifadesi tablodaki belirtilen alanda bulunan kayıtlardan birer örnek alır. Yani tekrar eden kayıtlardan bir tane alır ve bunun yanına da tekrar etmeyen kayıtları koyarak bir veri kümesi oluşturur. Ne işimize yarar veya nerede kullanabiliriz sorusu akla gelebilir.

Mesela bir üyelerinizin depolandığı bir tablo olduğunu düşünün. Bu tablodan mesela İstanbul'da kaç tane üyeniz olduğunu bulmak istediğinizi düşünün. Yaptığınız programa bir açılır liste kutusu koyup illerin isimlerini tek tek yazabilirsiniz. Ancak bu durum hem bir zaman kaybı olur hemde sistemde kayıtlı olmayan ileri de listeleyeceği için, açılır liste kutusu açıldığında uzun bir liste olacaktır. Bunun yerine Distinct komutu kullanarak tablodaki Şehir alanında yazan kayıtlar tek düşürülür ve bir açılır liste kutusuna aktarılabilir. Böylece ilgili şehir seçilerek o şehirde kaç tane üyenizin olduğunu görebilirsiniz. Listede görünmeyen şehirden üyeniz olmadığı anlamını rahatlıkla çıkarabilirsiniz.

Distinct Kullanım Biçimi

```
SELECT DISTINCT alan_adi1,alan_adi2
FROM tablo_adi
```

Distinct kelimesinden sonra yazılacak olan alanlara otomatik olarak uygulanır. Yani birden fazla alan üzerinde Distinct yapılacaksa alanların başına tek tek yazılmaz. Ayrıca Distinct komutu tek başına kullanılamaz. Mutlaka SELECT ifadesi ile kullanılmalıdır.

Burada dikkat edilmesi gereken nokta çoklu distinct kullanımında belirtilen alanlardaki verileri bir bütün olarak ele alır ve diğer kayırlarda benzersiz alanları bulmaya çalışır. (örnek2 ve örnek3'e bakabilirisiniz.)

Örnek Tablo Uygulaması:
Örnek olarak aşağıdaki gibi Personel isimli tablomuz olsun.

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	Erzincan	Muhasebe	3456789
4	İlhan ÖZLÜ	İstanbul	BİlgiİşlemSorumlusu	2345678

Örnek1:

SELECT DISTINCT Sehir FROM Personel

Bu kod ile tablodaki Sehir alanında bulunan kayıtları birer defa alır.

Çıktısı:

Sehir
İstanbul
Kocaeli
Erzincan

Örnek2:

SELECT DISTNICT Sehir, Bolum FROM Personel

Bu örnekte Personel tabosundan adı soyadı ve bölüm bilgisinin tutulduğu alanlar seçilmektedir. Ancak burada dikkat edilmesi gerekn nokta iki alanın sanki tek bir alanmş gibi değelendirilmesidir.

Çıktısı:

Sehir	Bolum
İstanbul	Bilgi İşlem Sorumlusu
Kocaeli	İdari İşler Yöneticisi
Erzincan	Muhasebe

Burada dikkat edeceğiniz üzere tablomuzdaki son satırı almadı. Çünkü Sehir ve Bolum alanlarını tek bir alanmış gibi düşündüğümüz zaman "İstanbul Bilgi İşlem Sorumlusu" ifadesi ortaya çıkar. Son satırdaki kayıtta aynı ifadeye denk gelmektedir. Bu sebeple dikkate alınmadı.

Örnek3:

SELECT DISTNICT Sehir, Bolum, Meslek_Kodu FROM Personel

Bu örnekte Personel tabosundan adı soyadı ve bölüm bilgisinin tutulduğu alanlar seçilmektedir. Ancak burada dikkat edilmesi gerekn nokta iki alanın sanki tek bir alanmş gibi değelendirilmesidir.

Çıktısı:

Sehir	Bolum	Meslek_Kodu
İstanbul	Bilgi İşlem Sorumlusu	1234567
Kocaeli	İdari İşler Yöneticisi	2345678
Erzincan	Muhasebe	3456789
İstanbul	BİlgiİşlemSorumlusu	2345678

Bu örnekte ise bütün kayıtlar gelmiş oldu. Çünkü Sehir, Bolum ve Meslek_kodu alanlarını tek bir alanmış gibi düşündüğümüz zaman; ilk satırı örnek verecek olursak "İstanbul Bilgi İşlem Sorumlusu 1234567" ifadesi ortaya çıkar. Sehir alanında iki tane İstanbul olmasına rağmen ikiside listelenmiştir. Çünkü iki kaydın Meslek_Kodu alanında yazan değer farklıdır.

Aynı tabloyu aşağıdaki kod ile çalıştırdığımız zaman:
Select Distinct Bolum, Meslek_kodu FROM Personel

Çıktısı:

Bolum	Meslek_Kodu
Bilgi İşlem Sorumlusu	1234567
İdari İşler Yöneticisi	2345678
Muhasebe	3456789
Bilgi İşlem Sorumlusu	2345678

Dikkat edileceği üzere Bilgi İşlem Sorumlusu alanı iki defa gelmiş oldu. Aynı mantıktan yola çıakrak alanların birleştirilmiş olduğunu düşünürsek, Bilgi İşlem Sorumlusu kayıtlarında Meslek_Kodu alanında 1234567 ve 2345678 verileri vardır. Dolayısı ile bu iki satır benzersiz değildir.

SQL WHERE Kullanımı

WHERE ifadesi tablodaki alanlarda okuma, güncelleme, silme gibi işlemleri yaparken belli kriterlere sahip kayıtlar üzerinde işlem yapmamızı sağlar. Where ifadesi belirtilmezse uygulanan komut bütün kayıtlar üzerinde geçerli olur. Mesela bir kaydı silmek istediğimiz zaman Where ifadesini kullanmazsak tablodaki bütün kayıtları silecektir.

Where Kullanım Biçimi

```
SELECT  alan_adi1,alan_adi2
FROM tablo_adi
WHERE alan_adi=sorgu_degeri
```

İlk bakışta kod biraz karışık görünebilir. Açıklayacak olursak;
SELECT ifadesi ile sorgunun bir seçme yani veritabanından okuma işlemi yapacağımız belirtmiş olduk. Sonrasında ise hangi alanlarda okuma yapacağımızı belirtiyoruz. Eğer silme işlemi yapacaksak select ifadesi yerine Delete ifadesi kullanılmalıdır.
FROM ifadesi ile veritabanı içindeki hangi tabloda işlem yapılacağı belirtiliyor.
WHERE ifadesi ile yapmak istediğimiz işlem (seçme, silme vs.) için gerekli parametreleri belirteceğimiz bölüm başlıyor.
Where ifadesinin hemen ardından hangi alandaki kritere göre işlem yapacaksan o alan adını yazıyoruz. sorgu_degeri ifadesi ile seçtiğimiz alandaki veri değerini giriyoruz.

Burada eşittir işareti yerine başka operatör işaretleride kullanılabilir. Mesele belli bir sayıdan büyük veya küçük olması durumlarına göre sorgulama yapılabilir. Veya iki tarih arası sorgulama yapılabilir. Aşağıdaki örneklerde detaylı olarak anlatılmıştır. Sayfanın sonunda eşittir dışındaki operatörleri ve ne işe yaradıkları tabloyu inceleyebilirsiniz.

Örnek Tablo Uygulaması:
Örnek olarak aşağıdaki gibi Personel isimli tablomuz olsun.

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	Erzincan	Muhasebe	3456789
4	İlhan ÖZLÜ	İstanbul	BİIgi İşlem Sorumlusu	2345678

Örnek1:

SELECT * FROM Personel WHERE Sehir='İstanbul'

Bu kod ile tablodaki Sehir alanında İstanbul bulunan kayıtları alır. Burada dikkat edilmesi gerek nokta Bolum alanı metinsel alan olduğu için eşittir işaretinden sonra aranmak istenen ifade tek tırnak işareti içinde yazılmıştır. Bazı veritabanı editör uygulamaları çift tırnak işareti kullanabilir.

Çıktısı:

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
4	İlhan ÖZLÜ	İstanbul	BİIgi İşlem Sorumlusu	2345678

Örnek2:

Delete FROM Personel Where id=1

Bu örnekte Personel tabosundan silme işlemi yapılmaktadır. Normalde Select ifadesinden sonra hangi alanlarda işlem yapacağımızı belirtirdik. Ancak Delete komutu alanlarda işlem yapmaz ve direk olarak ilgili kaydı siler. Bu örnekte id alanında 1 yazan kaydın silinmesini belirttik. Genellikle kullanım kolaylığı açısından id alanları tabloda sayısal alan olarak işaretlenir. Sayısal alanlarda işlemler tırnak işareti olmadan yapılır. Aynı şekilde delete yerine select kullanarak kaydı seçtiğimizi varsaysak bile id alanı sayısal alan olduğu için tırnak işaretleri kullanılmaz.

Operatör işaretleri tablosu:

Operatör	Açıklaması	Kullanım Örneği	Örnek A
=	Belirtilen değeri belirtilen adanda arar	Where Bolum='Bilgisayar'	Bolum a
<>	Belirtilen değer dışındaki kayıtları arar	Where Bolum<>'Bilgisayar'	Bolum a
>	Belirtilen değerden büyük kayıtları arar.	Where Maas>1000	Maaşı 1
>=	Belirtilen değere eşit ve büyük olanları arar.	Where Maas>=1000	Maaşı 1
<	Belitilen değerden küçük olanları arar	Where Maas<750	Maaşı 7
<=	Belirtilen değere eşit ve küçük olanları arar	Whee Maas<=750	Maaşı 7
Between	Belli bir aralıkta olan değerleri arar	Where Maas Between 750 and 1000	Maaşı 7
Like	Birkaç karakteri bilinen kayıtları arar	Where Sehir Like 'S%'	Sehir ala
In	Birden fazla değerleri tek alanda arar.	Where Sehir in ('İstanbul','Ankara')	Sehir ala

Not: Between, Like ve IN operatörlerinin kullanımını detaylı olarak konu başlıklarında incelenmiştir.

SQL AND ve OR Kullanımı

AND ve OR ifadeleri birden fazla alanda işlem yapılacaksa kullanılan operatörlerdir. AND operatörü birinci durumla beraber ikinci durumunda olduğu zaman kullanılır. OR operatörü ise birinci durum veya ikinci durumun gerçekleşmesi durumunda kullanılır.

ANd ve OR Kullanım Biçimi

```
SELECT  alan_adi1,alan_adi2
FROM tablo_adi
WHERE alan_adi1=sorgu_degeri AND alan_adi2=sorgu_degeri
```

```
SELECT  alan_adi1,alan_adi2
FROM tablo_adi
WHERE alan_adi1=sorgu_degeri OR alan_adi2=sorgu_degeri
```

Örnek Tablo Uygulaması:
Örnek olarak aşağıdaki gibi Personel isimli tablomuz olsun.

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	Erzincan	Muhasebe	3456789
4	İlhan ÖZLÜ	İstanbul	BİIgi İşlem Sorumlusu	2345678

Örnek1:

SELECT * FROM Personel WHERE Sehir='İstanbul' AND Bolum='Bilgi İşlem Sorumlusu'

Bu kod tablodaki Sehir alanında İstanbul ve Bolum alanında Bilgi İşlem Sorumlusu yazan kayıtları alır.

Çıktısı:

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567

4	İlhan ÖZLÜ	İstanbul	BİIgi İşlem Sorumlusu	2345678

Örnek2:

SELECT * FROM Personel WHERE Sehir='İstanbul' AND Sehir='Kocaeli'

Bu kod Sehir alanında İstanbul ve Kocaeli yazan kayıtları alır.

Çıktısı:

id	**Adi_soyadi**	**Sehir**	**Bolum**	**Meslek_Kodu**
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678
4	İlhan ÖZLÜ	İstanbul	BİIgi İşlem Sorumlusu	2345678

SQL ORDER BY Kullanımı

ORDER BY ifadesi kayıtları belirtilen alanda büyükten küçüğe veya küçükten büyüğe göre sıralar. ASC (ascending) parametresi ile küçükten büyüğe, DESC (descending) parametresi ile büyükten küçüğe göre sıralar. Burada sadece sayısal alanlar değil metinsel alanlarda alfabetik olarak sıralanabilir.

ORDER BY Kullanım Biçimi

```
SELECT  alan_adi1,alan_adi2
FROM tablo_adi
ORDER BY alan_adi2 ASC
```

```
SELECT  alan_adi1,alan_adi2
FROM tablo_adi
ORDER BY alan_adi2 DESC
```

Örnek Tablo Uygulaması:

Örnek olarak aşağıdaki gibi Personel isimli tablomuz olsun.

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	Erzincan	Muhasebe	3456789
4	İlhan ÖZLÜ	İstanbul	BİIgi İşlem Sorumlusu	2345678

Örnek1:

SELECT * FROM Personel ORDER BY Meslek_Kosu ASC

Bu kod tablodaki Meslek_Kodu alanına göre kayıtları küçükten büyüğe doğru alır.

Çıktısı:

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567

2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678
4	İlhan ÖZLÜ	İstanbul	BİIgi İşlem Sorumlusu	2345678
3	Serkan ÖZGÜREL	Erzincan	Muhasebe	3456789

Örnek2:

SELECT * FROM Personel ORDER BY Adi_soyadi DESC

Bu kod Adi_soyadi alanına göre kayıtları büyükten küçüğe yani Z harfinden A harfine doğru dizer. .

Çıktısı:

id	**Adi_soyadi**	**Sehir**	**Bolum**	**Meslek_Kodu**
3	Serkan ÖZGÜREL	Erzincan	Muhasebe	3456789
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
4	İlhan ÖZLÜ	İstanbul	BİIgi İşlem Sorumlusu	2345678
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678

Örnek3:

SELECT * FROM Personel Where Sehir='İstanbul' ORDER BY Meslek_kodu ASC

Bu kod Sehir alanında İstanbul yazan kayıtları seçer. ORDER BY ise sadece bu seçili olan kayıtlar üzerinde Meslek_kodu alanını baz alarak küçükten büyüğe doğru sıralama yapar.

Çıktısı

id	**Adi_soyadi**	**Sehir**	**Bolum**	**Meslek_Kodu**
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
4	İlhan ÖZLÜ	İstanbul	BİIgi İşlem Sorumlusu	2345678

SQL INSERT INTO Kullanımı

INSERT INTO ifadesi tablomuza yeni bir kayıt eklemek için kullanılır.

INSERT INTO Kullanım Biçimi

Insert Into kodu iki türlü kullanılabilir.

Birinci yöntem: Bu yöntemde direk tablo adı belirterek sadece değerleri yazmak surtiyle kayıt ekleyebiliriz. Ancak burada dikkat edeceğimiz nokta eklenecek değer tablomuzdaki alan sırasına göre olmalıdır. Mesele tablomuzdaki alan sıralaması Ad, Soyad, ve Dogum_yili seklinde olsun. Values ifadesinden yazılacak değerler sırası ile işlenir. Karışık yazdığımız zaman, Dogum_yili alanı sayısal bir alan ise metinsel veri girilemeyeceği için programımız hata verecektir. Veya sıralamya dikkat etmezsek bilgilerimiz olması gerek alana yazılmaz.

```
INSERT INTO  tablo_adi
VALUES (deger1, deger2, ...)
```

İkinci yöntem: Bu yöntemde ise eklenecek alanları ve değerleri kendimiz belirtiriz. Burada dikkat edilmesi gereken şey; yazdığımız alan adının sırasına göre değerleri eklememiz olacaktır.

```
INSERT INTO  tablo_adi (alan_adi1, alan_adi2, alan_adi3)
VALUES (deger1, deger2, deger3)
```

Örnek Tablo Uygulaması:

Örnek olarak aşağıdaki gibi Personel isimli tablomuz olsun.

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678

Örnek1:

```
INSERT INTO Personel
VALUES (3, 'Serkan ÖZGÜREL', 'Erzincan', 'Muhasebe', 3456789)
```

Yukarıda görüldüğü gibi tablomuza yeni bir kayıt ekleme kodunu yazdık. Alan adlarını sırası

ile kontrol ettik ve değerlerimizi sıraya dikkat ederek girdik. Metin karakterli alanlara veri eklenirken tek tırnak işareti kullanılır. Sayısal alanlara veri eklerken ifade direk olarak yazılır. Bazı veritbanı editörleri sayısal alana veri girerken de te tırnak işareti içinde yazımı kabul etmektedir.

Çıktısı:

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	Erzincan	Muhasebe	3456789

Örnek2:

```
INSERT INTO Personel (id, adi_soyadi, sehir)
VALUES (3, 'Serkan ÖZGÜREL', 'Erzincan')
```

Bu kod ile tablomuza sadece 3 alan için yeni kayıt eklenir .

Çıktısı:

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	Erzincan		

SQL UPDATE Kullanımı

UPDATE ifadesi tablomuzda bulunan kayıtları güncellemek yani değiştirmek için kullanılır.

UPDATE Kullanım Biçimi

```
UPDATE  tablo_adi
SET alan_adi1=deger1, alan_adi2=deger2, alan_adi3=deger3, ...)
WHERE secilen_alan_adi=alan_degeri
```

Burada dikkat edilecek nokta WHERE ifadesi ile belli bir kayıt veya kayıtlar seçilip değiştirilmek istenilen alanlardaki değerler değiştirilir. Eğer WHERE ifadesini kullanmadan yaparsak tablodaki bütün kayıtları değiştirmiş oluruz.

Örnek Tablo Uygulaması:

Örnek olarak aşağıdaki gibi Personel isimli tablomuz olsun.

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	Erzincan	Muhasebe	3456789

Örnek1:

```
UPDATE Personel
SET  Sehir='Ankara',Meslek_kodu=5555555
WHERE id=3
```

Tablomuzda bulunan kayıtlarda WHERE ifadesi ile id alanında 3 yazan kaydı seçmiş olduk. İlgili kaydın Sehir alanını Ankara ve Meslek_kodu alanını da 5555555 olarak değiştirdik. Metin karakterli alanlara tek tırnak işareti kullanılır. Sayısal alanlarda direk olarak yazılır. Bazı veritbanı editörleri sayısal alana veri girerken de te tırnak işareti içinde yazımı kabul etmektedir.

Çıktısı:

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567

2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	**Ankara**	Muhasebe	**5555555**

Örnek2:

UPDATE Personel
SET Meslek_kodu=1111111

Bu kodda WHERE ifadesi olmadığı için mevcut olan bütün kayıtların Meslek_kodu alanını 1111111 olarak değiştirir..

Çıktısı:

id	**Adi_soyadi**	**Sehir**	**Bolum**	**Meslek_Kodu**
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1111111
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	1111111
3	Serkan ÖZGÜREL	Erzincan	Muhasebe	1111111

SQL DELETE Kullanımı

DELETE ifadesi tablomuzda bulunan kayıtları silmek için kullanılır.

DELETE Kullanım Biçimi

```
DELETE  FROM tablo_adi
WHERE secilen_alan_adi=alan_degeri
```

Burada dikkat edilecek nokta WHERE ifadesi ile belli bir kayıt seçilip silinir. Eğer WHERE ifadesini kullanmadan yaparsak tablodaki bütün kayıtları silmiş oluruz.

Örnek Tablo Uygulaması:

Örnek olarak aşağıdaki gibi Personel isimli tablomuz olsun.

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	İstanbul	Muhasebe	3456789

Örnek1:

```
DELETE FROM Personel
WHERE id=3
```

Tablomuzda bulunan kayıtlarda WHERE ifadesi ile id alanında 3 yazan kaydı silmiş olduk. Metin karakterli alanlara tek tırnak işareti kullanılır. Sayısal alanlarda direk olarak yazılır. Bazı veritbanı editörleri sayısal alana veri girerken de te tırnak işareti içinde yazımı kabul etmektedir.

Çıktısı:

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678

Örnek2:

```
DELETE FROM Personel
WHERE Sehir='İstanbul'
```

Bu kodda WHERE ifadesi ile Sehir alanında İstanbul yazan kayıtları silmiş olduk.

Çıktısı:

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	1111111

SQL SELECT TOP Kullanımı

SELECT ifadesi veritbanından verileri okumak için kullanılır. Select ifadesi tek başına kullanıldığında veritabanındaki tablo üzerinde bulunan bütün verileri seçer. Ancak binlerce hatta on binlerce kaydın olduğu bir tablodan bütün verileri çekmek veritabanını zorlayacak ve uygulamanın kitlenmesine neden olacaktır. **Bunun yerine SELECT TOP ile belirtilen kadar kayıt seçilir.** Select Top ifadesi ile kayıt adedi veya yüzdesi belirtildikten sonra alan adları mutlaka yazılmalıdır. Yazılan alan adlarındaki kayıtlar ekrana gelir. * işareti konulursa bütün alanlar seçilir.

NOT: SELECT TOP ifadesi her veritabanında desteklenmez. Faklı veritabanlarında aynı işi yapan başka ifadeler kullanılır.

Select Top SQL SERVER ve ACCESS Veritabanlarında Kullanım Biçimi

Eğer sadece ilk bir kaç kaydı seçecek isek SELECT TOP ifadesinden sonra seçeceğimiz kayıt adedini belirtiriz.

```
SELECT TOP kayit_adedi   alan_adi1,alan_adi2,...
FROM tablo_adi
```

Eğer bütün kayıtlar içinden yüzdesel olarak seçim yapacaksakz.

```
SELECT TOP yuzde_degeri PERCENT alan_adi1, alan_adi2,...
FROM tablo_adi
```

Örnek Tablo Uygulaması:

Örnek olarak aşağıdaki gibi Personel isimli tablomuz olsun.

id	Adi_soyadi	Sehir	Bolum	Sicil No
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	Erzincan	Bilgi İşlem Sorumlusu	3456789
4	İlhan ÖZLÜ	İstanbul	Muhasebe	7677667

Örnek1:

SELECT TOP 2 Adi_soyadi, Sehir FROM Personel

Bu kod ile tablodaki Adi_soyadi ve Sehir alanlarındaki kayıtlardan sadece ilk iki tanesi seçilir.

Çıktısı:

Adi_soyadi	Sehir
Salih ESKİOĞLU	İstanbul
Ayhan ÇETİNKAYA	Kocaeli
Serkan ÖZGÜREL	Erzincan
İlhan ÖZLÜ	İstanbul

Örnek2:

SELECT TOP 25 Percent * FROM Personel

Bu örnekte Personel tabosunda bütün alanlar seçilmektedir. Ancak bütün kayıtlardan sadece yüzde 25'i seçilmiştir. Tablomuzda toplam 4 kayıt olduğu için bu kod ile sadece 1 kayıt seçilmiş olur.

Çıktısı:

id	Adi_soyadi	Sehir	Bolum	Sicil No
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567

Select Top MySQL ve ORACLE Veritabanlarında Kullanım Biçimi

MySQL Kullanım Biçimi: MySQL veritabanlarında ilgili seçim LIMIT ifadesi ile gerçekleştirilmektedir.

```
SELECT alan_adi1,alan_adi2,...
FROM tablo_adi
LIMIT kayit_adedi
```

Oracle Kullanım Biçimi: ORACLE veritabanlarında WHERE ROWNUM ifadesi ile birlikte kullanılır.

```
SELECT alan_adi1,alan_adi2,...
FROM tablo_adi
WHERE ROWNUM <= kayit_adedi
```

Örnek3:
SELECT Adi_soyadi FROM Pesonel LIMIT 3

Bu örnekte sadece Adi_soyadi alanındaki ilk 3 kayıt seçilmketedir.

Çıktısı:

Adi_soyadi
Salih ESKİOĞLU
Ayhan ÇETİNKAYA
Serkan ÖZGÜREL

Örnek4:

SELECT Bolum FROM Pesonel WHERE ROWNUM <= 3

Bu örnekte sadece Bolum alanındaki ilk 3 kayıt seçilmketedir.

Çıktısı:

Bolum
Bilgi İşlem Sorumlusu
İdari İşler Yöneticisi
Bilgi İşlem Sorumlusu

SQL LIKE Kullanımı

LIKE operatörü tablomuzda bulunan kayıtlardan belirttiğimiz kriterler uygun olanları seçmek için kullanılır.

LIKE Kullanım Biçimi

```
SELECT alan_ad(lari)
FROM tablo_adi
WHERE sorgulanacak_alan_adi LIKE sorgulama_degeri
```

LIKE bir operatördür ve WHERE ile kullanılır. Yani eşittir, büyüktür veya küçüktür işareti yerine kullanılır.

Örnek Tablo Uygulaması:

Örnek olarak aşağıdaki gibi Personel isimli tablomuz olsun.

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	İstanbul	Finans Yöneticisi	3456789

Örnek1:

```
SELECT *
FROM Personel
WHERE Sehir LIKE 'İ%'
```

Burada dikkat edilecek nokta LIKE ifadesinden sonra % işaretinin kullanılmasıdır. Bu örnekte Sehir alanında İ harfi ile başlayan kayıtlar seçilmiştir. % işareti İ harfinden sonra kalan karakteri temsil eder. Yani bu sorgunun anlamı: Sehir alanındaki verilerden İ harfi ile başlayan kayıtları seç.

Çıktısı:

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	**İstanbul**	Bilgi İşlem Sorumlusu	1234567
3	Serkan ÖZGÜREL	**İstanbul**	Finan Yöneticisi	3456789

Örnek2:

```
SELECT *
FROM Personel
WHERE Bolum LIKE '%Yönetici%'
```

Bu kodda Bolum alanının herhangi bir yerinde (başında, ortasında veya sonunda) Yönetici kelimesini seçecektir.

Çıktısı:

2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	İstanbul	Finans Yöneticisi	3456789

<u>ÖNEMLİ BİLGİ:</u> NOT kelimmesi ile belirtilen değere sahip olmayan kayıtlar seçilir

Örnek3:

```
SELECT *
FROM Personel
WHERE Bolum NOT LIKE '%Yönetici%'
```

Bu kod Bolum alanının herhangi bir yerinde Yönetici <u>yazmayan</u> kayıtları seçer.

Çıktısı:

id	**Adi_soyadi**	**Sehir**	**Bolum**	**Meslek_Kodu**
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567

SQL LIKE İŞARETLERİNİN Kullanımı

LIKE operatörü tablomuzda bulunan kayıtlardan belirttiğimiz kriterler uygun olanları seçmek için kullanılır. Ancak bazı durumlarda farklı ihtiyaçlar söz konusu olabilir. Bu tip bir durumda LIKE ifadesinden sonra bazı özel işretler kullanılarak istenilen sonuçlara ulaşılabilir.

LIKE İşaretleri Tablosu

İşaret	Açıklaması
%	Birden çok karakterin yerine kullanılır
_	Sadece bir karakterin yerine kullanılır.
[karakter_listesi]	Köşeli parantez ile belirtilen harf listesi % işareti ile kullanılır.

LIKE bir operatördür ve WHERE ile kullanılır. Yani eşittir, büyüktür veya küçüktür işareti yerine kullanılır.

Örnek Tablo Uygulaması:

Örnek olarak aşağıdaki gibi Personel isimli tablomuz olsun.

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	İzmit	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	İzmir	Finans Yöneticisi	3456789
4	İlhan ÖZLÜ	İstanbul	Muhasebe	7765677

Örnek1:

```
SELECT *
FROM Personel
WHERE Sehir LIKE 'İ%'
```

Bu örnekte Sehir alanında İ harfi ile başlayan kayıtlar seçilmiştir. % işareti İ harfinden sonra kalan karakteri temsil eder. Yani bu sorgunun anlamı: Sehir alanındaki verilerden İ harfi ile başlayan kayıtları seç.

Çıktısı:

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	**İstanbul**	Bilgi İşlem Sorumlusu	1234567
3	İlhan ÖZLÜ	**İstanbul**	Muhasebe	7765677

Örnek2:

```
SELECT *
FROM Personel
WHERE Bolum LIKE '%Yönetici%'
```

Bu kodda Bolum alanının herhangi bir yerinde (başında, ortasında veya sonunda) Yönetici kelimesini seçecektir.

Çıktısı:

2	Ayhan ÇETİNKAYA	İzmit	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	İzmir	Finans Yöneticisi	3456789

ÖNEMLİ BİLGİ: NOT kelimmesi ile belirtilen değere sahip olmayan kayıtlar seçilir

Örnek3:

```
SELECT *
FROM Personel
WHERE Bolum NOT LIKE '%Yönetici%'
```

Bu kod Bolum alanının herhangi bir yerinde Yönetici yazmayan kayıtları seçer.

Çıktısı:

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
4	İlhan ÖZLÜ	İstanbul	Muhasebe	7765677

Örnek4:

```
SELECT *
FROM Personel
WHERE Sehir  LIKE 'İzmi_'
```

Bu kod Sehir alanının İzmi ile başlayan ve son harfi ne olursa olsun farketmeyen kayıtları seçer.

Çıktısı:

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
2	Ayhan ÇETİNKAYA	İzmit	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	İzmir	Finans Yöneticisi	3456789

Örnek5:

```
SELECT *
FROM Personel
WHERE Adi_soyadi  LIKE '[S,A]%'
```

Bu kod Adi_soyadi alanındaki ilk harfi S veya A ile başlayan kayıtları seçer. Son kayıt İ harfi ile başladığı için seçilmemiştir.

Çıktısı:

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	İzmit	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	İzmir	Finans Yöneticisi	3456789

Örnek6:

```
SELECT *
FROM Personel
WHERE Adi_soyadi  LIKE '[A-K]%'
```

Bu kod Adi_soyadi alanındaki, ilk harfi A ile K harfleri arasında ki herhangi bir harf ile başlayan kayıtları seçer. Bu koda uyan sadece bir kayıt vardır. Adi_soyadi alanındaki diğer kayıtlar S ve İ ile başladığından dolayı

Çıktısı:

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
2	Ayhan ÇETİNKAYA	İzmit	İdari İşler Yöneticisi	2345678

SQL IN Kullanımı

IN operatörü belirtilen tek bir alanda birden fazla değeri aramak için kullanılır.

IN Kullanım Biçimi

```
SELECT alan_ad(lari)
FROM tablo_adi
WHERE sorgulanacak_alan_adi IN (aranacak_deger1,aranacak_deger2,...)
```

IN bir operatördür ve WHERE ile kullanılır.

Örnek Tablo Uygulaması:

Örnek olarak aşağıdaki gibi Personel isimli tablomuz olsun.

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	Erzincan	Finans Yöneticisi	3456789
4	İlhan ÖZLÜ	İstanbul	Muhasebe	3456723

Örnek1:

```
SELECT *
FROM Personel
WHERE Sehir IN ( 'İstanbul','Kocaeli')
```

Bu kod ile Sehir alanında İstanbul ve Kocaeli yazan kayıtlar seçilir.

Çıktısı:

id	Adi_soyadi	Sehir	Bolum	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	2345678
4	İlhan ÖZLÜ	İstanbul	Muhasebe	3456723

SQL BETWEEN Kullanımı

Between operatörü ile bir alanda belirtilen aralıktaki değerleri aramak için kullanılır.

BETWEEN Kullanım Biçimi

```
SELECT alan_ad(lari)
FROM tablo_adi
WHERE sorgulanacak_alan_adi BETWEEN aranacak_deger1 AND aranacak_deger2
```

BETWEEN bir operatördür ve WHERE ile kullanılır.

Örnek Tablo Uygulaması:

Örnek olarak aşağıdaki gibi Personel isimli tablomuz olsun.

id	Adi_soyadi	Sehir	Bolum	Maas
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	900
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	1300
3	Serkan ÖZGÜREL	Erzincan	Finans Yöneticisi	1250
4	İlhan ÖZLÜ	İstanbul	Muhasebe	1500

Örnek1:

```
SELECT *
FROM Personel
WHERE Maas BETWEEN 1000 AND 1300
```

Bu kod ile Maaşı 1000 ile 1300 arasında olan kayıtlar seçilir. Burada dikkat ediliceği üzer Maas alanı rakamsal bir alan olduğu için 1000 ve 1300 degerleri tırnak işareti kullanılmadan yazılmıştır.

Çıktısı:

id	Adi_soyadi	Sehir	Bolum	Maas
2	Ayhan ÇETİNKAYA	Kocaeli	İdari İşler Yöneticisi	1300
3	Serkan ÖZGÜREL	Erzincan	Finans Yöneticisi	1250

Örnek2:

```
SELECT Adi_soyadi, Sehir
FROM Personel
WHERE Sehir BETWEEN 'A' AND 'K'
```

Bu kod ile Sehir alanındaki kayıtlardan A ile K harfi arasındaki harflerden herhangi bir harf ile başlayan kayıtlar seçilmektedir. Burada dikkat ediliceği üzer Bolum alanı metinsel bir alan olduğu için aranacak degerler tırnak işareti kullanılarak yazılmıştır.

Çıktısı:

Adi_soyadi	**Sehir**
Ayhan ÇETİNKAYA	Kocaeli
Serkan ÖZGÜREL	Erzincan

Örnek3:

```
SELECT Adi_soyadi, Sehir
FROM Personel
WHERE Sehir NOT BETWEEN 'A' AND 'K'
```

Bu kod ile Sehir alanındaki kayıtlardan A ile K harfi arasındaki harflerden herhangi bir harf ile başlamayan kayıtlar seçilmektedir. NOT kelimesi yapılan işlemin olumsuz yani şartlara uymayan halini işler.

Çıktısı:

Adi_soyadi	**Sehir**
Salih ESKİOĞLU	İstanbul
İlhan ÖZLÜ	İstanbul

SQL AS Alias Kullanımı

AS ifedesi ile uzun ve kullanımı zor olan tablo veya alan adlarına geçici olarak kısa isimler vererek bunları kodlamalarımızda kullanabiliriz. Böylece mevcut tablo yapımız bozmadan anlık olarak belirlediğimiz isimleri kullanabiliriz. Verilecek olan geçici ad eğer boşluk içeriyorsa köşeli parantez içinde yazılır. Tablodaki alan adlarında Türkçe karakter kullanımına izin verilmemektedir. Bu tip durumlarda AS ifadesi ile geçici bir isim verip yazdığımız uygulamada kullanabiliriz.

AS İfadesinin Alan Adlarında Kullanım Biçimi

```
SELECT alan_adi AS gecici_ad
FROM tablo_adi
```

AS İfadesinin Tablo adlarında Kullanım Biçimi

```
SELECT alan_adi
FROM tablo_adi AS gecici_ad
```

Örnek Tablo Uygulaması:

Birinci Tablomuz: Örnek olarak aşağıdaki gibi **Personel_Bilgileri** isimli tablomuz olsun.

id	Adi_soyadi	Yasadigi_sehir	Bolum_adi	Meslek_Kodu
1	Salih ESKİOĞLU	İstanbul	Bilgi İşlem Sorumlusu	1234567
2	Ayhan ÇETİNKAYA	İzmit	İdari İşler Yöneticisi	2345678
3	Serkan ÖZGÜREL	İzmir	Finans Yöneticisi	3456789
4	İlhan ÖZLÜ	İstanbul	Muhasebe	7765677

İkinci Tablomuz: Örnek olarak aşağıdaki gibi **Detay_Bilgileri** isimli tablomuz olsun.

id	Satilan_urun	Satis_fiyati
1	Buzdolabi	1000
1	Çamaşır Makinesi	900
4	Buzdolabı	1100
4	Televizyon	1800

Örnek1:

```
SELECT Adi_Soyadi AS isim, Yasadigi_sehir AS memleket
FROM Personel_bilgileri
```

Bu örnekte Adi_soyadi ve Yasadigi_sehir alaları için AS ifadesi ile geçici bir ad verilmiş olnur.

Çıktısı:

isim	**memleket**
Salih ESKİOĞLU	İstanbul
Ayhan ÇETİNKAYA	İzmit
Serkan ÖZGÜREL	İzmir
İlhan ÖZLÜ	İstanbul

Örnek2:

```
SELECT Adi_Soyadi AS isim, Yasadigi_sehir AS [Yaşadığı Şehir]
FROM Personel_bilgileri
WHERE [Yaşadığı Şehir]= 'İstanbul'
```

Bu kodda görüleceği üzere Yasadigi_sehir alanı için tanımlanan geçici ad WHERE ifadesinden sonra kullanılmaktadır. Artık sorgunun geri kalan kısımlarında [Yaşadığı Şehir] olarak kullanılabilir. Verilen geçici adda boşluk olduğu için köşeli parantez içinde yazılmalıdır.

Çıktısı:

isim	**Yaşadığı Şehir**
Salih ESKİOĞLU	İstanbul
İlhan ÖZLÜ	İstanbul

Örnek3:

```
SELECT personel.id, personeladi_soyadi AS isim, satislar.satilan_urun AS [Ürün],
satislar.satis_fiyati AS [Satış Fiyatı]
FROM Personel_bilgileri AS personel, Detay_bilgileri AS satislar
WHERE personel.id=satislar.id
```

İlk bakışta oldukça karışık bir kod olarak görünebilir. Burad iki tane tablo bir arada kullanılmıştır. İnceleyecek olursak;

FROM ifadesinden sonra Personel_bilgileri isimli tablonun adı kısaltılarak personel yapılmış. Aynı şekilde Detay_bilgileri tablosu ise satislar olarak adlandırılmış.
SELECT ifadesinden sonra, iki tane tablo kullanıldığı için alanadları yazılırken hangi tablodan olduğu belirtilmek zorundadır. İşte burada tabloya verilen kısa ad kullanılabilir.
WHERE ifadesi ile sorgu sonucu ortaya çıkan verilerde iki tablodaki id alanları eşit olan kayıtları seçerek hangi personelin hangi ürünü sattığı öğrenilebilir.

Çıktısı:

id	isim	Ürün	Satış Fiyatı
1	Salih ESKİOĞLU	Buzdolabi	1000
1	Salih ESKİOĞLU	Çamaşır Makinesi	900
4	İlhan ÖZLÜ	Buzdolabı	1100
4	İlhan ÖZLÜ	Televizyon	1800

SQL INNER JOIN Kullanımı

İki adet tablomuzdaki kayıtları belli bir kritere göre birleştirmek için INNER JOIN komutu kullanılır. Acces veritabanı programında kullanılan tablo arası ilişkileri gibidir.

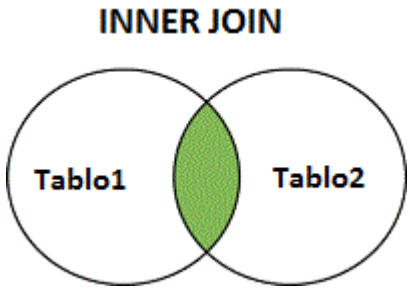
INNER JOIN Kullanım Biçimi

```
SELECT alan_ad(lari)
FROM tablo1 INNER JOIN tablo2
ON tablo1.alan_adi=tablo2.alan_adi
```

veya

```
SELECT alan_ad(lari)
FROM tablo1 JOIN tablo2
ON tablo1.alan_adi=tablo2.alan_adi
```

NOT: INNER JOIN yerine sadece JOIN kullanılabilir.
Burada görüleceği üzere From ifadesi ile birinci tablomuzu ve ardından JOIN (veya INNER JOIN) ile ikinci tablomuu belirtmiş oluyoruz. ON ile hangi alanların eşitleneceği gösterilmektedir. Birinci tabloda olan bir kayıt ikinci tabloda olmayabilir veya ikinci tabloda olan ir kayıt biirnci tabloda olmayabilir. Bu durumda sadece belirtilen alanlarda eşit olan kayıtlar seçilir. Mesele id_no alanına göre eşitleme yapılırsa birinci tabloda id_no alanında 1,2,3 kayıtları ve ikinci tabloda id_no alanında 1,2,3,4,5 kayıtları varsa; bu kayıtlardan sadece 1,2,3 olanlar işleme alınır. Her iki tabloda bulunan ortak kayıtlar seçilir. Yani kesişim kümesi baz alınır.



Örnek Tablo Uygulaması:

Örnek olarak aşağıdaki gibi Müşteriler isimli tablomuz olsun.

id	**Adi_soyadi**
1	Salih ESKİOĞLU
2	Ayhan ÇETİNKAYA
3	Serkan ÖZGÜREL
4	İlhan ÖZLÜ

İkinci tablomuz olan Satışlar ise aşağıdaki gibi olsun.

id	Satilan_mal	Satis_fiyati
1	Buzdolabi	1200
1	LCD TV	1800
2	LCD TV	1750
1	Çamaşır Makinesi	950

Örnek1:

```
SELECT *
FROM Müşteriler JOIN Satışlar
ON Musteriler.id=Satislar.id
```

Bu kodda iki tablodaki id alanları eşitlenmiş. id alanında ortak olan kayıtlar seçilecek ve diğer kayıtlar dikkate alınmayacaktır. 3 ve 4 nolu id lere sahip müşteriler çıktı kayıtlarında olmayacaktır. Çünkü Satışlar tablosunda bunlarla iligli bir kayıt bulunmamaktadır.

Çıktısı:

id	Adi_soyadi	Satilan_mal	Satis_fiyati
1	Salih ESKİOĞLU	Buzdolabı	1200
1	Salih ESKİOĞLU	LCD TV	1800
2	Ayhan ÇETİNKAYA	LCD TV	1750
1	Salih ESKİOĞLU	Çamaşır Makinesi	950

Örnek2:

```
SELECT Adi_soyadi, Satilan_mal
FROM Musteriler JOIN Satislar
ON Musteriler.id=Satislar.id
```

Bu kod ile sadece adı soyadı ve satılan mal alanları seçilmiş. Gene aynı şekilde id alanalr eşit olan kayıtlar işleme alınmış. Çıktıya dikkat edilirse id alanı çıktı alanları arasında yoktur. Çünkü select ile ilgili alan belirtilmemiştir.

Çıktısı:

Adi_soyadi	Satilan_mal
Salih ESKİOĞLU	Buzdolabı
Salih ESKİOĞLU	LCD TV
Ayhan ÇETİNKAYA	LCD TV
Salih ESKİOĞLU	Çamaşır Makinesi

Örnek3:

```
SELECT Adi_soyadi, Satilan_mal, musteriler.id AS id_no
FROM Musteriler JOIN Satislar
ON Musteriler.id=Satislar.id
ORDER BY id_no ASC
```

Bu kodda Select ifadesinden sonra musteri tablosundaki id alanı AS ifadesi ile özel tanım olarak belirtilmiştir. Yani SQL kodu içerisinde müşteri tablosunun id alanı kısaca id_no olarak kullanılacaktır. AS için detaylı kullanım bilgisine buradan ulaşabilirsiniz. ORDER BY ile seçilen kayıtlar id numarasına küçükten büyüğe sıralanmıştır.

Çıktısı:

id_no	Adi_soyadi	Sehir
1	Salih ESKİOĞLU	Buzdolabı
1	Salih ESKİOĞLU	LCD TV
1	Salih ESKİOĞLU	Çamaşır Makinesi
2	Ayhan ÇETİNKAYA	LCD TV

SQL LEFT JOIN Kullanımı

LEFT JOIN ile iki adet tablomuzdaki kayıtları belli bir kritere göre birleştirebilir. Burada asıl olan birinci tablondaki kayıtlardır. İkinci tablodan sadece birinci tabloda olan kayıtlar alınır. İkinci tabloda olupta birinci tabloda olmayan alanların değeri boş (NULL) olarak gelecektir.

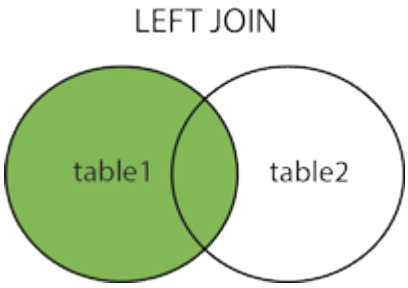
LEFT JOIN Kullanım Biçimi

```
SELECT alan_ad(lari)
FROM tablo1 LEFT JOIN tablo2
ON tablo1.alan_adi=tablo2.alan_adi
```

veya

```
SELECT alan_ad(lari)
FROM tablo1 LEFT OUTER JOIN tablo2
ON tablo1.alan_adi=tablo2.alan_adi
```

NOT: LEFT JOIN aynı zamanda LEFT OUTER JOIN olarak da kullanılabilir.
Burada görüleceği üzere From ifadesi ile birinci tablomuzu ve ardından LEFT JOIN (veya LEFT OUTER JOIN) ile ikinci tablomuu belirtmiş oluyoruz. ON ile hangi alanların eşitleneceği gösterilmektedir. Birinci tabloda ki bütün kayıtlar seçilir. Buna karşılık eşitlenen alanlara bakılarak ikinci tablodan sadece birinci tabloda olan kayıtlar seçilir. Mesela id_no alanına göre eşitleme yapılırsa birinci tabloda id_no alanında 1,2,3,4,5,6,7 kayıtları ve ikinci tabloda id_no alanında 3,4,9,10,11,15 kayıtları varsa; birinci tablodaki bütün kayıtlar alınırken ikinci tablodan sadece birinci tablodakine eşit olan kayıtlar alınır. Yani ikinci tablodan sadece 3 ve 4 id nolu kayıtlar alınır.



Örnek Tablo Uygulaması:

Örnek olarak aşağıdaki gibi Müşteriler isimli tablomuz olsun.

id	**Adi_soyadi**
1	Salih ESKİOĞLU
2	Ayhan ÇETİNKAYA
3	Serkan ÖZGÜREL
4	İlhan ÖZLÜ

İkinci tablomuz olan Satışlar ise aşağıdaki gibi olsun.

id	Satilan_mal	Satis_fiyati
1	Buzdolabi	1200
2	LCD TV	1800
5	LCD TV	1750
8	Çamaşır Makinesi	950

Örnek1:

```
SELECT *
FROM Mutseriler LEFT JOIN Satislar
ON Musteriler.id=Satislar.id
```

Bu kodda iki tablodaki id alanları eşitlenmiş. Birinci tablodan bütün kayıtlar alınacaktır. İkinci tablodan ise id alanında sadece 1 ve 2 yazan kayıtlar alınacaktır. id alanında 5 ve 8 yazan kayıtlar dikkate alınmayacaktır. Çünkü birinci tabloda 5 ve 8 id numarasına sahip kayıt bulunmamaktadır.

Çıktısı:

id	Adi_soyadi	Satilan_mal	Satis_fiyati
1	Salih ESKİOĞLU	Buzdolabı	1200
2	Ayhan ÇETİNKAYA	LCD TV	1800
3	Serkan ÖZGÜREL		
4	İlhan ÖZLÜ		

Örnek2:

```
SELECT Adi_soyadi, Satilan_mal
FROM Musteriler JOIN Satislar
ON Musteriler.id=Satislar.id
```

Bu kod ile sadece adı soyadı ve satılan mal alanları seçilmiş. Gene aynı şekilde id alanları eşit olan kayıtlar işleme alınmış. Çıktıya dikkat edilirse id alanı çıktı alanları arasında yoktur. Çünkü select ile ilgili alan belirtilmemiştir.

Çıktısı:

Adi_soyadi	Satilan_mal
Salih ESKİOĞLU	Buzdolabı
Ayhan ÇETİNKAYA	LCD TV
Serkan ÖZGÜREL	
İlhan ÖZLÜ	

Örnek3:

```
SELECT Adi_soyadi, Satilan_mal, musteriler.id AS id_no
FROM Musteriler JOIN Satislar
ON Musteriler.id=Satislar.id
ORDER BY id_no ASC
```

Bu kodda Select ifadesinden sonra musteri tablosundaki id alanı AS ifadesi ile özel tanım olarak belirtilmiştir. Yani SQL kodu içerisinde müşteri tablosunun id alanı kısaca id_no olarak kullanılacaktır. AS için detaylı kullanım bilgisine buradan ulaşabilirsiniz. ORDER BY ile seçilen kayıtlar id numarasına küçükten büyüğe sıralanmıştır.

Çıktısı:

id_no	Adi_soyadi	Sehir
1	Salih ESKİOĞLU	Buzdolabı
2	Ayhan ÇETİNKAYA	LCD TV
3	Serkan ÖZGÜREL	
4	İlhan ÖZLÜ	

SQL UNION Kullanımı

UNION ile iki adet tablomuzdaki seçeceğimiz alanları birleştirerek tek bir tablo alanıymış gibi kullanabiliriz. Union ile iki tablodaki alanlar birleştirilirken tekrarlayan kayıtlar bir defa alınır. Eğer tekrarlayan kayıtların alınması isteniyorsa UNION ALL kullanılmalıdır.

UNION Kullanım Biçimi

```
SELECT alan_ad(lari) FROM tablo1
UNION
SELECT alan_ad(lari) FROM tablo2
```

UNION ALL Kullanım Biçimi

```
SELECT alan_ad(lari) FROM tablo1
UNION ALL
SELECT alan_ad(lari) FROM tablo2
```

Görüleceği üzere iki tane SELECT ifadesi kullanılmaktadır. Yani iki ayrı sorgu yapısını UNION ile birleştirmiş oluyoruz. Burada dikkat edilecek olan nokta Select ifadesinden sonra yazılacak alan sayısı her iki sorgu ifadesinde de aynı olmalıdır. Alan adları farklı olabilir. Yani birinci select ifadesinde Şehir alanı kullanılırken diğer select ifadesinde Adres alanı kullanılabilir. Sonuçta anlamsız bir veri çıkabilir ancak yapı bu şekilde çalışmaktadır. Alanları birleştirirken yazım sırasına göre birleştirme yapmaktadır. Yani birinci Select ifadesinden sonra Adi_soyadi, Sehir yazıldıysa, çekilen verinin anlamlı olması için ikinci select ifadesinden sonra da Adi_soyadi, Sehir şeklinde yazılması gerekmektedir. Eğer ikinci bölüme Sehir, Adi_soyadi yazılırsa birinci tablodan adi_soyadi alanındaki veriler ile ikinci tablodan Sehir alanındaki veriler birleştirilir. Ancak bazı SQL editör programları böylesi bir durumun önün geçmek için kendi içlerinde kontrol mekanizması kurarak kullanıcıyı uyarabilmektedirler.

Örnek Tablo Uygulaması:
Örnek olarak aşağıdaki gibi Personel isimli tablomuz olsun.

id	Adi_soyadi	Sehir
1	Salih ESKİOĞLU	İstanbul
2	Ayhan ÇETİNKAYA	Kocaeli
3	Serkan ÖZGÜREL	Erzincan
4	İlhan ÖZLÜ	İstanbul

İkinci tablomuz olan Musteriler ise aşağıdaki gibi olsun.

id	Adi_soyadi	Sehir
1	Veysi Yamlı	Van
2	Sırrı Derman	Mersin

Örnek1:

```
SELECT Sehir FROM Personel
UNION
SELECT Sehir FROM Musteriler
```

Bu kod ile iki tabloda Sehir alanlarındaki veriler tekrar edenler bir defa alınmak suretiyle birleştirilmiş olunur. Dikkat edileceği üzere PErsonel tablosunda iki tane İstanbul bulunmaktadır.

Çıktısı:

Sehir
İstanbul
Kocaeli
Erzincan
Van
Mersin

Örnek2:

```
SELECT Sehir FROM Personel
UNION ALL
SELECT Sehir FROM Musteriler
```

Burada UNION ALL kulanılmıştır. Yani her iki tabloda Sehir alanında bulunan kayıtlar olduğu gibi alınmıştır. Personel tablosunda iki tane İstanbul kaydı vardır. Bunlar olduğu gibi alınacaktır.

Çıktısı:

Sehir
İstanbul
Kocaeli
Erzincan
İstanbul
Van
Mersin

Örnek3:

```
SELECT Adi_soyadi, Sehir FROM Personel
UNION ALL
SELECT Adi_soyadi,Sehir FROM Musteriler
```

İki tablo olduğu gibi birleştirilmiştir.

Çıktısı:

Adi_soyadi	**Sehir**
Salih ESKİOĞLU	İstanbul
Ayhan ÇETİNKAYA	Kocaeli
Serkan ÖZGÜREL	Erzincan
İlhan ÖZLÜ	İstanbul
Veysi Yamlı	Van
Sırrı Derman	Mersin

SQL SELECT INTO Kullanımı

SELECT INTO ifadesi ile bir tablodaki verileri alıp yeni bir tablo oluşturup içine kopyalayabiliriz. Sonuçta veritabanında yeni bir tablo oluşturulacağı için veritabanı üzerinde işlem yapan kullanıcının yeni bir tablo oluşturma yetkisine sahip olması gerekmektedir.

SELECT INTO Kullanım Biçimi

```
SELECT alan_ad(lari)
INTO yeni_tablo_adi [IN hedef_database]
FROM tablo1
```

NOT 1: Eğer yeni oluşturacağımız tablo aynı veritabanı içindeyse [IN hedef_database] ifadesi kullanılmaz. Eğer farklı bir veritabanı içine kopyasını alacaksak o zaman IN operatörü ile hedef veritabanını belirtmemiz gerekir.

NOT 2: Select Into yapısı ile yeni oluşturulacak olan tabloya, mevcut tablomuzdaki alanlar veri tipleri ve içindeki verilerle birlikte aynen kopyalanır. Eğer alan adını mevcut isminden farklı bir isimle oluşturmak istersek o zaman AS yapısı kullanabiliriz.

```
SELECT id, ad_soyad AS isim, yasadigi_sehir AS sehir
INTO personel_yedek
FROM personel
```

NOT 3: Eğer tablo içindeki verileri hariç tutup, sadece alan adları ve veri tiplerini almak istersek WHERE 1=0 eklememiz gerekir.

```
SELECT id, ad_soyad, sehir
INTO personel_yedek
FROM personel
WHERE 1=0
```

Select Into Örnekleri:

- Farklı ve harici veritabanına kopyalama: Acces veritabanlarında geçerli olan bir yapıdır. IN ifadesinden sonra belirtilen yoldaki veritabanına tablomuz kopyalanır.

```
SELECT *
INTO personel_yedek IN 'c:\backup.mdb'
FROM personel
```

- Aynı veritabanı içinde tablo kopyalama: Veritabanımızdaki işlem kayıtlarını yıllara göre tuttuğumuzu varsayalım. Her yeni yıl geldiğinde elimizde bulunan boş bir şablon veritabanını kullanarak bir kopyasını yeni çalışma yılı için oluşturabiliriz. Bu örnekte islemler_sablon isimli tablomuzun bir kopyası islemler_2014 adıyla oluşturuluyor. Böylece 2014 yılı için işlem kayıtlarının depolanacağı bir tablo elde etmiş oluruz.

```
SELECT *
INTO islemler_2014
FROM islemler_sablon
```

- Belli kritere göre seçilen kayıtları yeni bir tabloya kopyalama: Tablomuzda bulunan kayıtları istediğimiz bir kritere göre seçerek yeni bir tabloya kopyalayabiliriz. Aşağıdaki örnekte Personel tablosunda bulunan kayıtlardan Sehir alanında İstanbul yazanlar seçiliyor. Bu seçilen kayıtlardan sadece ad_soyad alanında bulunan kayıtlar istanbul_personelleri isimli yeni bir tabloya aktarılıyor.

```
SELECT ad_soyad
INTO istanbul_personelleri
FROM personel
WHERE sehir='istanbul'
```

SQL INSERT INTO Kullanımı

INSERT INTO ifadesi ile bir tablodaki verileri alıp varolan bir tablo içine kopyalayabiliriz. Hedefte belirttiğimiz tablonun var olması gerekmektedir. Hedef tabloda var olan alanlar silinmez. Var olan alanların yanına yeni alanlar eklenir.

INSERT INTO Kullanım Biçimi

```
INSERT INTO Hedef_tablo (alan_adi1,alan_adi2...)
SELECT alan_adi1,alan_adi2...
FROM tablo1
```

INSERT INTO Örnekleri:

- Tablomuzdaki bütün alanları kopyalama: Eğer tablomuzdaki bütün alanların hepsini almak istersek alan adlarını tek tek belirtmemize gerek yoktur. Aşağıdaki kod ile Personel tablomuzdaki bütün alanlar verileri ile birlikte olduğu gibi alınarak personel_yedek tablosuna (mevcut alanlar korunarak) eklenir.

```
INSERT INTO personel_yedek
SELECT *
FROM personel
```

- Tablomuzdaki alanların adını değiştirerek kopyalama: Tablomuzda bulunan alanları hedef tablomuza başka bir isimle oluşturabiliriz. Aşağıdaki örnekte Personel tablosundaki ad_soyad alanı Personel_yedek tablosuna isim olarak kopyalanıyor. Ancak Sehir alanı olduğu gibi kopyalanıyor.

```
INSERT INTO personel_yedek (isim, sehir)
SELECT  ad_soyad, sehir
FROM personel
```

- Belli kritere göre seçilen kayıtları kopyalama: Tablomuzda bulunan kayıtları istediğimiz bir kritere göre seçerek başka bir tabloya kopyalayabiliriz. Aşağıdaki örnekte Personel tablosunda bulunan kayıtlardan Sehir alanında İstanbul yazanlar seçiliyor. Bu seçilen kayıtlardan sadece ad_soyad alanında bulunan kayıtlar istanbul_personelleri isimli var olan tabloya isim olarak aktarılıyor.

```
INSERT INTO istanbul_personelleri (isim)
SELECT ad_soyad
FROM personel
WHERE sehir='istanbul'
```

SQL CREATE DATABASE Kullanımı

CREATE DATABASE ifadesi ile yeni bir veritabanı oluşturulur. Bu işlemi yapabilmek için mevcut kullanıcımızın veritabanı oluşturma yetkisine sahip olması gerekmektedir.

CREATE DATABASE Kullanım Biçimi

```
CREATE DATABASE veritabani_adi
```

Açıkçası kod bu kadar ve detaylandıracak bir şey yok.

Örnek: Kargo isminde yeni bir veritabanı oluşturmak için kullanılacak kod aşağıdaki gibidir.

CREATE DATABASE Kargo

SQL CREATE TABLE Kullanımı

CREATE TABLE ifadesi ile var olan veritabanımıza yeni bir tablo oluşturulur. Bu işlemi yapabilmek için mevcut kullanıcımızın tablo oluşturma yetkisine sahip olması gerekmektedir.

CREATE TABLE Kullanım Biçimi

```
CREATE TABLE
(
alan_adi1 veri_tipi(boyut),
alan_adi2 veri_tipi(boyut),
alan_adi3 veri_tipi(boyut),
....
)
```

alan_adi: *tablomuzdaki verinin depolanacağı alan adı belirtilir.*
veri_tipi: *ilgili alanın hangi veri tipinde olacağını belirtir (varchar, integer, double vb.)*
boyut: *ilgili alanın en fazla olabilecek boyutu belirtir.*
Kullanılabilecek veri tipleri ve açıklamalarına buradan ulaşabilirsiniz.

Örnek: Personel isminde yeni bir tablo oluşturmak için kullanılacak kod aşağıdaki gibidir.

```
CREATE TABLE Personel
(
id int,
adi_soyadi varchar(25),
sehir varchar(15),
bolum varchar(15),
medeni_durum bolean
)
```

Bu kod ile id isminde bir sayısal alan, boşluklar dahil olmak üzere numara ve harften oluşan 25 karakterli adi_soyadi, 15 karakterli sehir ve bolum ile medeni_durum adında bolean yani "evet-hayır", "var-yok", "1-0" gibi sadece iki seçenekli bir alan tanımlanıyor. Oluşan tabloda herhangi bir kayıt olamayacaktır.

Çıktısı:

id	adi_soyadi	sehir	bolum	medeni_durum

SQL CONSTRAINTS Kullanımı

CREATE TABLE ifadesi ile var olan veritabanımıza yeni bir tablo oluşturulur. Ancak oluşturulacak tablo alanlarının belli başlı kriterlere göre oluşmasını isteyebiliriz. Sadece bir alan için bunu yapabileceğimiz gibi bütün alanlar içinde uygulayabiliriz.

CONSTRAINTS Kullanım Biçimi

```
CREATE TABLE
(
alan_adi1 veri_tipi(boyut) constraint_adi,
alan_adi2 veri_tipi(boyut) constraint_adi,
alan_adi3 veri_tipi(boyut) constraint_adi,
....
)
```

Kullanılabilecek kriterler aşağıdaki gibidir:

NOT NULL: Alanında boş geçilemeyeceğini belirtir.
UNIQUE: Bu alana girilecek verilerin hiç biri birbirine benzeyemez. Yani tekrarlı kayıt içeremez.
PRIMERY KEY: Not Null ve Unique kriterlerinin her ikisini birden uygulanmasıdır.
FOREIGN KEY: Başka bir tablodaki kayıtlarla eşleştirmek için alandaki kayıtların tutarlılığını belirtir.
CHECK: Alandaki değerlerin belli bir koşulu sağlaması için kullanılır.
DEFAULT: Alan için herhangi bir değer girilmezse, varsıyalan olarak bir değer giremeyi sağlar.

Yuklarıdaki kriterlerin detaylı açıklamalarını sol taraftaki menüden ilgili kriter ismine tıklayarak öğrenebilirsiniz.

SQL NUT NULL Kullanımı

Tablomuza kayıt girerken NOT NULL kriteri ile belirleyeceğimiz alan veya alanların boş geçilmesini engelleyebiliriz. NOT NULL kriteri tabloyu ilk defa oluştururken belirlenebileceği gibi daha sonra ALTER yapısı ile de belirleyeceğimiz alanlara NOT NULL kriterini verebiliriz.

NOT NULL Kullanım Biçimi

```
CREATE TABLE
(
alan_adi1 veri_tipi(boyut) NOT NULL,
alan_adi2 veri_tipi(boyut) ,
alan_adi3 veri_tipi(boyut) ,
....
)
```

NOT NULL kriterini birden fazla alana uygulayabiliriz. Yukarıdaki kullanım biçiminde sadece bir alana nasıl uygulanacağı gösterilmiştir.

Örnek1:

```
Create Table
(id int NOT NULL,
adi_soyadi varchar(20) ,
Meslek varchar(10)
)
```

Yuklarıdaki örnekte, id numarası alanının boş geçilmemesi gerektiği için sadece id alanını NOT NULL kriteri ile yapıladırılmıştır.

Örnek2:

```
Create Table
(id int NOT NULL,
adi_soyadi varchar(20) NOT NULL ,
Meslek varchar(10)
)
```

Yuklarıdaki örnekte, id numarası ve isimin boş geçilmemesi gerektiği için id ve adi_soyadi alanları NOT NULL kriteri ile yapılandırılmıştır.

SQL UNIQUE Kullanımı

Tablomuzda bir alandaki verilerin tekrarlı olmasını istemiyorsak UNIQUE kriterini kullanmamız gerekir. Tablo tasarımını yaparken bir alanın UNIQUE olup olmayacağına iyi karar vermemiz gerekir. Çünkü daha sonradan ALTER komutu ile alanın özelliklerini değiştirirken, ilgili alanda tekrarlayan kayıtlar varsa UNIQUE değerini veremeyiz. PRIMARY KEY ile çokça karıştırılmaktadır. Aradaki farkları sıralayacak olursak:

- Birden fazla alan tek bir PRIMARY KEY ile tanımlanabilir. Ancak PRIMARY KEY yapısı her tabloda sadece bir tane olabilir. UNIQUE yapısı bir tabloda birden fazla olabilir.
- PRIMARY KEY yapısı ile boş kayıtlara izin verilmez. UNIQUE yapısında boş kayıtlara izi n verilir.
- PRIMARY KEY yapısı ile tablo üzerinde bir index tanımı oluşturulur her kaydın benzersiz bir tanımı yapılır. Böylece kullandığınız uygulama geliştirme ortamında (Ör: .NET) tablo üzerinde daha etkin sonuçlar elde edilebilir. UNIQUE yapısında ise alandaki değerlerin benzersiz olup olmadığına bakılır. Birden fazla alanda UNIQUE yapıldığında bunları bir index adıyla tanımlanması sağlaabilir ancak bu sadece bir tanımladır.

UNIQUE Kullanım Biçimi

SQL Server / Oracle / MS Acess ortamlarında sadece bir alanda kullanım biçimine örnek:

```
CREATE TABLE Personel
(
id int NOT NULL UNIQUE,
adi_soyadi varchar(20) ,
Sehir varchar(20)
)
```

UNIQUE kriterini sadece bir alana vereceksek nasıl kullanılacağı gösterilmiştir.

MySQL ortamında sadece bir alanda kullanım biçimine örnek:

```
CREATE TABLE Personel
(
id int NOT NULL,
adi_soyadi varchar(20) ,
Sehir varchar(20),
UNIQUE (id)
)
```

Görüldüğü gibi MySQL veritabanında işlem yapacaksanız UNIQUE ifadesini sonradan belirtmeniz gerekmektedir.

MySQL / SQL Server / Oracle / MS Acess ortamlarında birden fazla alanda kullanım biçimine örnek:

```
CREATE TABLE Personel
(
id int NOT NULL,
adi_soyadi varchar(20) NOT NULL ,
Sehir varchar(20),
CONSTRAINT id_no UNIQUE (id,adi_soyadi)
)
```

Burada görüleceği üzere birden fazla alana UNIQUE değeri verirken CONSTRAINT ifadesi ile bu işleme bir tanım giriliyor. Aslında bu tanım bizim tablomuzun index alanını oluşturmaktadır. İndexleme sayesinde tablomuzdaki verilerin bütülüğü daha sağlam olurken aramalarda da daha hızlı sonuçlar elde ederiz. UNIQUE ifadesinden sonra ise ilgili alanları virgül ile ayırarak yazarız.

Yuakrıdaki örnekler hep yeni bir veritabanı oluşturuken kullanılan örneklerdir. Ancak var olan bir veritabanında bir alanı UNIQUE yapmak istersek ALTER yapısını kullanmamız gerekir.

MySQL / SQL Server / Oracle / MS Acess ortamlarında bir alanda kullanım biçimine örnek:

```
ALTER TABLE Personel
ADD UNIQUE (id)
```

MySQL / SQL Server / Oracle / MS Acess ortamlarında birden fazla alanda kullanım biçimine örnek:

```
ALTER TABLE Personel
ADD CONSTRAINT  id_no UNIQUE (id,adi_soyadi)
```

Eğer UNIQUE olarak kriterlendirilmiş alanı normale çevirmek istersek DROP ifadesini kullanmamız gerekir.

SQL Server / Oracle / MS Acess ortamlarında UNIQUE yapısını kaldırmak:

```
ALTER TABLE Personel
DROP  CONSTRAINT id_no
```

Burada dikkat edilmesi gereken nokta eğer çoklu alanda UNIQUE işlemi yaptıysak, CONSTRAINT ifadesinden sonra tablomuzdaki alan adı değil, oluşturduğumuz index adı yazılmalıdır. Eğer tek bir alanda oluşturduysak o zaman CONSTRAINT ifadesinden sonra sadece alana adını yazabiliriz.

MySQL ortamlarında UNIQUE yapısını kaldırmak:

```
ALTER TABLE Personel
DROP  INDEX id_no
```

MySQL yapısından silerken tek fark CONSTRAINT yerine INDEX ifadesi kullanılır.

SQL PRIMARY KEY Kullanımı

PRIMARY KEY ile tablomuzdaki ilgili alanda benzersiz kayıtların tutulmasını istediğimiz durumlarda kullanılır. Yapısal olarak UNIQUE ile karıştırılabilir. Aradaki farkları sıralayacak olursak:

- Birden fazla alan tek bir PRIMARY KEY ile tanımlanabilir. Ancak PRIMARY KEY yapısı her tabloda sadece bir tane olabilir. UNIQUE yapısı bir tabloda birden fazla olabilir.
- PRIMARY KEY yapısı ile boş kayıtlara izin verilmez. UNIQUE yapısında boş kayıtlara izi n verilir.
- PRIMARY KEY yapısı ile tablo üzerinde bir index tanımı oluşturulur her kaydın benzersiz bir tanımı yapılır. Böylece kullandığınız uygulama geliştirme ortamında (Ör: .NET) tablo üzerinde daha etkin sonuçlar elde edilebilir. UNIQUE yapısında ise alandaki değerlerin benzersiz olup olmadığına bakılır. Birden fazla alanda UNIQUE yapıldığında bunları bir index adıyla tanımlanması sağlaabilir ancak bu sadece bir tanımladır.

PRIMARY KEY Kullanım Biçimi

SQL Server / Oracle / MS Acess ortamlarında sadece bir alanda kullanım biçimine örnek:

```
CREATE TABLE Personel
(
id int NOT NULL PRIMARY KEY,
adi_soyadi varchar(20) ,
Sehir varchar(20)
)
```

PRIMARY KEY kriterini sadece bir alana vereceksek nasıl kullanılacağı gösterilmiştir.

MySQL ortamında sadece bir alanda kullanım biçimine örnek:

```
CREATE TABLE Personel
(
id int NOT NULL,
adi_soyadi varchar(20) ,
Sehir varchar(20),
PRIMARY KEY (id)
)
```

Görüldüğü gibi MySQL veritabanında işlem yapacaksanız PRIMARY KEY ifadesini sonradan belirtmeniz gerekmektedir.

MySQL / SQL Server / Oracle / MS Acess ortamlarında birden fazla alanda kullanım biçimine örnek:

```
CREATE TABLE Personel
(
```

```
id int NOT NULL,
adi_soyadi varchar(20) NOT NULL ,
Sehir varchar(20),
CONSTRAINT id_no PRIMARY KEY  (id,adi_soyadi)
)
```

Burada görüleceği üzere birden fazla alan PRIMARY KEY yapısı içine alınıyor. CONSTRAINT ifadesi ile bu işleme bir tanım giriliyor. Aslında bu tanım bizim tablomuzun index alanını oluşturmaktadır. İndexleme sayesinde tablomuzdaki verilerin bütülüğü daha sağlam olurken aramalarda da daha hızlı sonuçlar elde ederiz. Ayrıca kullandığınz uygulama geliştirme ortamlarında (ör .Net) tablo üzerinde daha etkin kullanım imkanınız olacaktır. PRIMARY KEY ifadesinden sonra ise ilgili alanları virgül ile ayırarak yazarız.

Yuakrıdaki örnekler hep yeni bir veritabanı oluşturuken kullanılan örneklerdir. Ancak var olan bir veritabanında bir alanı UNIQUE yapmak istersek ALTER yapısını kullanmamız gerekir.

MySQL / SQL Server / Oracle / MS Acess ortamlarında bir alanda kullanım biçimine örnek:

```
ALTER TABLE Personel
ADD PRIMARY KEY (id)
```

MySQL / SQL Server / Oracle / MS Acess ortamlarında birden fazla alanda kullanım biçimine örnek:

```
ALTER TABLE Personel
ADD CONSTRAINT  id_no PRIMARY KEY (id,adi_soyadi)
```

Burada dikkat edilecek nokta; ALTER ile sonradan bir alana PRIMARY KEY kriteri tanımlanırken ilgili alanda veya alanlarda NULL yani boş kayıt olmamalıdır.

Eğer PRIMARY KEY olarak kriterlendirilmiş alanı normale çevirmek istersek DROP ifadesini kullanmamız gerekir.

SQL Server / Oracle / MS Acess ortamlarında PRIMARY KEY yapısını kaldırmak:

```
ALTER TABLE Personel
DROP  CONSTRAINT id_no
```

Burada dikkat edilmesi gereken nokta eğer çoklu alanda PRIMARY KEY işlemi yaptıysak, CONSTRAINT ifadesinden sonra tablomuzdaki alan adı değil, oluşturduğumuz index adı

yazılmalıdır. Eğer tek bir alanda oluşturduysak o zaman CONSTRAINT ifadesinden sonra sadece alana adını yazabiliriz.

MySQL ortamlarında UNIQUE yapısını kaldırmak:

```
ALTER TABLE Personel
DROP  PRIMARY KEY id_no
```

MySQL yapısından silerken tek fark direk olarak PRIMARY KEY ifadesi kullanılır.

SQL DEFAULT Kullanımı

DEFAULT kriteri ile oluşturduğumuz tabloda bir alanın değeri eğer kullanıcı tarafından boş olarak geçilirse, sistemin o alana otomatik olarak bir değer atamasını sağlayabiliriz.
Böylece verilerimizin bütünlüğü sağlanmış olacağı gibi ileride geliştireceğimiz uygulamalarda hata ile karşılaşma ihtimalimizde düşecektir.

DEFAULT Kullanım Biçimi

```
CREATE TABLE Kargo
(
id int,
adi_soyadi varchar(20) ,
adres varchar(20),
odeme_tipi varchar(25) DEFAULT 'Ücret Alıcı'
)
```

Yukarıdaki örnekte kargo takibi yapmak üzere Kargo isminde bir tablo oluşturulmuştur. Eğer yeni bir kargo takip kaydı girilirken odeme_tipi bilgisi boş geçilirse otomatik olarak ilgili alana 'Ücret Alıcı' ifadesi yazılacaktır. DEFAULT kriteri ile sadece belirleyeceğimiz bir veri değil, sistem verileride kullanılabilir.

```
CREATE TABLE Kargo
(
id int,
adi_soyadi varchar(20) ,
adres varchar(20),
kayit_tarihi date DEFAULT getdate()
)
```

Görüldüğü gibi kayit_tarihi alanına eğer bir değer belirtmezsek, otomatik oalrak günün tarihini atacaktır. Buradaki getdate() ile günü tarihi bilgisi elde edilmektedir.

Yuakrıdaki örnekler hep yeni bir veritabanı oluşturuken kullanılan örneklerdir. Ancak var olan bir veritabanında bir alanı DEFAULT yapmak istersek ALTER yapısını kullanmamız gerekir.

MySQL veritabanlarında kullanım biçimine örnek:

```
ALTER TABLE Kargo
ADD odeme_bilgisi SET DEFAULT 'Ücret Alıcı'
```

SQL Server / MS Acess veritabanlarında kullanım biçimine örnek:

```
ALTER TABLE Kargo
```

```
ADD COLUMN  odeme_bilgisi SET DEFAULT 'Ücret Alıcı'
```

Oracle veritabanlarında kullanım biçimine örnek:

```
ALTER TABLE Kargo
MODIFY odeme_bilgisi DEFAULT 'Ücret Alıcı'
```

Eğer DEFAULT olarak kriterlendirilmiş alanı normale çevirmek istersek DROP ifadesini kullanmamız gerekir.

SQL Server / Oracle / MS Acess ortamlarında DEFAULT yapısını kaldırmak:

```
ALTER TABLE Kargo
ALTER COLUMN odeme_bilgisi DROP DEFAULT
```

MySQL ortamlarında UNIQUE yapısını kaldırmak:

```
ALTER TABLE Kargo
ALTER COLUMN odeme_bilgisi DROP DEFAULT
```

Burada dikkat edilmesi gereken nokta iki tane ALTER ifadesinin kullanılmasıdır. Yani ALTER TABLE Kargo DROP odeme_bilgisi DEFAULT kullanımı yanlış olacaktır. Çünkü normal şartlarda direk olarak DROP yazılırsa belirtilen alan tablodan silinir. Bunun için iki defa ALTER kullanılmıştır.

SQL ISNULL(), NVL(), IFNULL(), COALESCE() Kullanımı

Tablomuzdaki bir alanda işlem yapmak istediğimiz zaman boş değerler sorun oluşturabilir. Bu durumda boş alanlara işlem sırasında değer verebilir ve işlemin sağlıklı yapılmasını sağlanabilir.

ISNULL: SQL Server ve MS Access 'de kullanılır.
IFNULL: MySQL 'de kullanılır.
NVL: Oracle'da kullanılır.
COALESCE: MySQL'de kullanılır.

Fonksiyonların Kullanım Biçimi
İlgili fonksiyonların kullanımını örnekler üzerinden açıklamak daha sağlıklı olacaktır.

Aşağıdaki gibi Urunler tablomuz olsun

id	Urun_adi	Fiyat	Stok
1	Buzdolabı	1500	10
2	LCD TV	1850	5
3	Çamaşır Makinesi 1000 Devir		1
4	Çamaşır Makinesi 800 Devir	850	1

Örnek1:

```
SELECT Urun_adi,(Fiyat*ISNULL(fiyat,0)) AS Stok_degeri
FROM Urunler
```

Çıktısı:

Urun_adi	Stok_degeri
Buzdolabı	15000
LCD TV	9250
Çamaşır Makinesi 1000 Devir	0
Çamaşır Makinesi 800 Devir	850

Burada kodun SQL Server üzerinde nasıl uygulandığını görüyoruz. Fiyatı boş almak yerine sıfır olarak alıyor.

Örnek2:

```
SELECT Urun_adi,(Fiyat*IFNULL(fiyat,0)) AS Stok_degeri
FROM Urunler
```

Çıktısı:

Urun_adi	**Stok_degeri**
Buzdolabı	15000
LCD TV	9250
Çamaşır Makinesi 1000 Devir	0
Çamaşır Makinesi 800 Devir	850

Burada MySQL yapısında ki kullanım biçimi görülmektedir. Çıktı birinci örnekle aynı olacaktır. Değişen sadece ISNULL yerine IFNULL kullanılmış olmasıdır.

SQL AVG() Kullanımı

AVG() fonksiyonu ile belirtilen alandaki değerlerin ortalaması elde edilir. Elimizdeki ürünlerin ortalama değerini bulmak için kullanılabilir.

Sadece sayısal alanlarda kullanılabilir.

AVG() Kullanım Biçimi

```
SELECT AVG(alan_adi) FROM tablo
```

Aşağıdaki gibi Urunler tablomuz olsun

id	Urun_adi	Fiyat
1	Buzdolabı	1500
2	LCD TV	1850
3	Çamaşır Makinesi 1000 Devir	950
4	Çamaşır Makinesi 800 Devir	850

Örnek1:

```
SELECT AVG(Fiyat)
FROM Urunler
```

Çıktısı:

Expr1000
1287,5

Burada görüldüğü üzere alan adı Expr1000 olarak görünmekedir. Aşağıdaki örnekte bu ismi daha anlamlı hale nasıl getirildiğinigörebilirsiniz.

Örnek2:

```
SELECT AVG(Fiyat) AS Camasir_Mak_Ortalama_Degeri
FROM Urunler
WHERE Urun_adi Like 'Çamaşır Makinesi%'
```

Çıktısı:

Camasir_Mak_Ortalama_Degeri
900

Burada AS ile ilgili alanın adı daha anlamlı bir hale getirilmiştir. Dikkat edileceği üzere WHERE kullanılarak elimizde bulunan bütün çamaşır makinelerinin oratalam değeri elde edilmiş oldu.

Örnek3:

```
SELECT Urun_adi,Fiyat
FROM Urunler
WHERE Fiyat>(SELECT AVG(Fiyat) FROM Urunler)
```

Çıktısı:

Urun_adi	Fiyat
Buzdolabi	1500
LCD TV	1850

Bu örnekte fiaytı ortlamanın üstünde olan ürünler listelenmektedir. Önce ikinci SELECT yapısı ile ortalama fiyat bilgisi elde ediliyor. Daha sonra bu bilgi WHERE ile yapısı ile fiyat alanına aktarılıyor. Kontrol işareti olarak > yani büyüktür işareti kullanılarak elde edilen ortalama fiyattan büyük olan değerlerin alınması sağlanıyor.

SQL COUNT() Kullanımı

COUNT() fonksiyonu belirtilen alandaki veya tablodaki toplam kayıt sayısını verir. Burada dikkat edilmesi gereken alan üzerindeki kayıt sayıları alınırken boş verilerin dikkate alınmamasıdır.

COUNT() Kullanım Biçimi

```
SELECT COUNT(alan_adi) FROM tablo
```

veya

```
SELECT COUNT(*) FROM tablo
```

veya

```
SELECT COUNT(DISTINCT alan_adi) FROM tablo
```

Birinci kullanım biçiminde belirttiğimiz alandaki boş olanlar hariç kaç tane kayıt olduğunu elde ederiz. İkinci kullanım biçiminde ise tablomuzdaki toplam kayıt sayısını elde ederiz. DISTNINCT kullanımında ise belirtilen alandaki tekrar eden kayıtlar sadece bir defa sayılır. DISTINCT kullanımı MS Access veritabanlarında desteklenmemektedir.

Aşağıdaki gibi Urunler tablomuz olsun

id	**Urun_adi**	**Fiyat**
1	Buzdolabı	1500
2	LCD TV	1850
3	Çamaşır Makinesi	
4	Çamaşır Makinesi	850

Örnek1:

```
SELECT COUNT(Fiyat)
FROM Urunler
```

Çıktısı:

Expr1000
3

Tablomuzda 4 tane kayıt olması rağmen kodumuz bize 3 değerini döndürdü. Çünkü fiyat alanındaki verilerden birisi boştur. Boş veriler dikkat alınmamaktadır. Burada görüldüğü üzere alan adı Expr1000 olarak görünmekedir. Aşağıdaki örnekte bu ismi daha anlamlı hale nasıl getirildiğini görebilirsiniz.

Örnek2:

```
SELECT COUNT(*) AS Camasir_Mak_Sayisi
FROM Urunler
WHERE Urun_adi='Çamaşır Makinesi'
```

Çıktısı:

Camasir_Mak_Sayisi
2

Burada AS ile ilgili alanın adı daha anlamlı bir hale getirilmiştir. Dikkat edileceği üzere WHERE kullanılarak elimizde bulunan çamaşır makinelerinin sayısı elde edilmiş oldu.

Örnek3:

```
SELECT COUNT(DITINCT Urun_adi) AS Tekrarsiz_kayit_sayisi
FROM Urunler
```

Çıktısı:

Tekrarsiz_kayit_sayisi
3

Tablomuzda Urun_adi alanında bulunan kayıtlardan tekrar etmeyen kayıtlar seçiliyor. Toplamda 4 tane kayıt olmasına rağmen Çamaşır makinesi iki defa olduğu için kodumuz bize sonuç olarak 3 değerini döndürmektedir.

SQL FIRST() Kullanımı

FIRST() fonksiyonu belirtilen alandaki ilk kayıt değerini verir. Bu fonksiyon sadece MS Access veritabanlarında çalışmaktadır. Diğer veritabanlarında ilk kayda ulaşmak için faklı sql kodları kullanılmaktadır.

FIRST() Kullanım Biçimi

```
SELECT FIRST(alan_adi)
FROM tablo
```

SQL Server Veritabanlarında ilk kaydı bulma:

```
SELECT TOP 1 alan_adi
FROM tablo
```

MySQL Veritabanlarında ilk kaydı bulma:

```
SELECT alan_adi
FROM tablo
LIMIT 1
```

ORACLE Veritabanlarında ilk kaydı bulma:

```
SELECT alan_adi
FROM tablo
WHERE ROWNUM <= 1
```

Aşağıdaki gibi Urunler tablomuz olsun

id	Urun_adi
1	Buzdolabı
2	LCD TV
3	Çamaşır Makinesi
4	Çamaşır Makinesi

Örnek1:

```
SELECT FIRST(Urun_adi)
FROM Urunler
```

Çıktısı:

Expr1000
Buzdolabi

Burada görüldüğü üzere alan adı Expr1000 olarak görünmekedir. Aşağıdaki örnekte bu ismi daha anlamlı hale nasıl getirildiğini görebilirsiniz.

Örnek2: SELECT FIRST(Urun_adi) AS Birinci_Kayit
FROM Urunler
Çıktısı:

Birinci_Kayit
Buzdolabı

Burada AS ile ilgili alanın adı daha anlamlı bir hale getirilmiştir.

Örnek3: SELECT TOP 1 Urun_adi
FROM Urunler
Çıktısı:

Urun_adi
Buzdoalbi

SELECT TOP 1 ile SQL Server veritabanlarında ilk kaydı bulabiliriz. Dikkat edileceği üzere alan adı Expr1000 değil, direk olarak veritabanındaki isimle geldi.

Örnek4: SELECT FIRST(id) AS Birinci_Kayit_id
FROM Urunler
WHERE Urun_adi='Çamaşır Makinesi'
Çıktısı:

Birinci_Kayit_id
3

Bu örnekte WHERE ile Urun_adi alanında Çamaşır Makinesi yazan kayıtlar seçilmiştir. Seçili olan bu kayıtlardan birinci olanın id alanındaki değer alınır.

Örnek5: SELECT FIRST(Urun_adi) AS Sirali_ilk_urun
FROM Urunler
ORDER BY Urun_adi DESC
Çıktısı:

Sirali_ilk_urun
LCD TV

Tablomuzdaki kayıtları Urun_adi alanına göre DESC kullanılarak büyükten küçüğe sıralayarak dizdiğimiz zaman ilk kayıt değişmektedir.

SQL LAST() Kullanımı

LAST() fonksiyonu belirtilen alandaki son kayıt değerini verir. Bu fonksiyon sadece MS Access veritabanlarında çalışmaktadır. Diğer veritabanlarında son kayda ulaşmak için faklı sql kodları kullanılmaktadır. Ancak bu kullanım veritabanının o anki don kaydına erişim sağlamaz. Kayıtları alafebetik ve numarasal olarak dizdirdikten sonra son kayıt değerine erişilir.

FIRST() Kullanım Biçimi

```
SELECT LAST(alan_adi)
FROM tablo
```

SQL Server Veritabanlarında ilk kaydı bulma:

```
SELECT TOP 1 alan_adi
FROM tablo
ORDER BY alan_adi DESC
```

MySQL Veritabanlarında ilk kaydı bulma:

```
SELECT alan_adi
FROM tablo
ORDER BY alan_adi DESC
LIMIT 1
```

ORACLE Veritabanlarında ilk kaydı bulma:

```
SELECT alan_adi
FROM tablo
ORDER BY alan_adi DESC
WHERE ROWNUM <= 1
```

Aşağıdaki gibi Urunler tablomuz olsun

id	**Urun_adi**
1	Buzdolabı
2	LCD TV
3	Çamaşır Makinesi
4	Çamaşır Makinesi

Örnek1:

```
SELECT LAST(Urun_adi)
FROM Urunler
```

Çıktısı:

Expr1000
Çamaşır Makinesi

Burada görüldüğü üzere alan adı Expr1000 olarak görünmekedir. Aşağıdaki örnekte bu ismi daha anlamlı hale nasıl getirildiğini görebilirsiniz.

Örnek2:

```
SELECT LAST(Urun_adi) AS Son_Kayit
FROM Urunler
```

Çıktısı:

Birinci_Kayit
Çamaşır Makinesi

Burada AS ile ilgili alanın adı daha anlamlı bir hale getirilmiştir.

Örnek3:

```
SELECT TOP 1 Urun_adi
FROM Urunler
ORDER BY Urun_adi DESC
```

Çıktısı:

Urun_adi
LCD TV

SELECT TOP 1 ile SQL Server veritabanlarında ilk kaydı bulabiliriz. Dikkat edileceği üzere alan adı Expr1000 değil, direk olarak veritabanındaki isimle geldi. Ancak dikkat edeceğiniz üzere bizim tablomuzdaki son kayıt Çamaşır Makinesi. DESC ilk kayıtlar büyükten küçüğe göre dizilince LCD TV başa geçmiş oldu. SELECT TOP 1 ile de son ilk kaydı yani aslında sıralanmış ilk kaydın değerini almış olduk

Örnek4:

```
SELECT LAST(id) AS Son_Kayit_id
FROM Urunler
WHERE Urun_adi<>'Çamaşır Makinesi'
```

Çıktısı:

Son_Kayit_id
2

Bu örnekte WHERE ile Urun_adi alanında Çamaşır Makinesi yazmayan kayıtlar seçilmiştir. Seçili olan bu kayıtlardan sonuncu olanın id alanındaki değer alınır.

Örnek5:

```
SELECT FIRST(Urun_adi) AS Sirali_ilk_urun
FROM Urunler
ORDER BY Urun_adi DESC
```

Çıktısı:

Sirali_ilk_urun
LCD TV

Tablomuzdaki kayıtları Urun_adi alanına göre DESC kullanılarak büyükten küçüğe sıralayarak dizdiğimiz zaman ilk kayıt değişmektedir.

SQL MAX() Kullanımı

MAX() fonksiyonu belirtilen alandaki en büyük değeri verir. Tablomuzdaki 100 lerce ürün kaydının olduğunu düşünün. En yüksek fiyatın ne olduğunu bulmak istediğimizi düşünün. Böyle bir durumda tek tek fiyatları kontrol edip en pahalı fiyatı bulabiliriz. Ancak bu oldukça büyük bir zaman kaybına yol açacaktır. Bunun yerine bir sql kodu ile sonuca direk ulaşabiliriz. Aynı şekilde muhasebe departmanı müdürü veya şirektin genel müdürü, personele ödediği en yüksek maaşı görmek isteyebilir. Bu durumda da bir sql kodu ile sonuca gidebiliriz. Veya sıcaklık ölçüm değerlerinin olduğu bir tablo düşünün. Ay içerisinde veya yıl içinde en yüksek sıcaklık değerinin ne olduğunu bulmak istediğimiz zaman MAX() fonksiyonu oldukça faydalı bir kullanım olacaktır. Sadece sayısal alanda değil aynı zamanda da metinsel alanlarda da kullanılabilir. Bu durumda metinsel veriyi A'dan Z'ye dizip en sondaki kaydı verecektir.

MAX() Kullanım Biçimi

```
SELECT MAX(alan_adi) FROM tablo
```

Aşağıdaki gibi Ürünler tablomuz olsun

İd	Urun_adi	Fiyat
1	Buzdolabı	1500
2	LCD TV	1850
3	Çamaşır Makinesi	950
4	Bulaşık Makinesi	850

Örnek1: SELECT Max(Fiyat)
FROM Urunler

Çıktısı:

Expr1000
1850

Burada görüldüğü üzere alan adı Expr1000 olarak görünmekedir. Aşağıdaki örnekte bu ismi daha anlamlı hale nasıl getirildiğinigörebilirsiniz.

Örnek2: SELECT Max(Fiyat) AS EnYuksekFiyat
FROM Urunler

Çıktısı:

EnYuksekFiyat
1850

Burada AS ile ilgili alanın adı daha anlamlı bir hale getirilmiştir.

SQL MIN() Kullanımı

MIN() fonksiyonu belirtilen alandaki en küçük değeri verir. Tablomuzdaki 100 lerce ürün kaydının olduğunu düşünün. En düşük fiyatın ne olduğunu bulmak istediğimizi düşünün. Böyle bir durumda tek tek fiyatları kontrol edip en ucuz fiyatı bulabiliriz. Ancak bu oldukça büyük bir zaman kaybına yol açacaktır. Bunun yerine bir sql kodu ile sonuca direk ulaşabiliriz. Aynı şekilde muhasebe departmanı müdürü veya şirektin genel müdürü, personele ödediği en düşük maaşı görmek isteyebilir. Bu durumda da bir sql kodu ile sonuca gidebiliriz. Veya sıcaklık ölçüm değerlerinin olduğu bir tablo düşünün. Ay içerisinde veya yıl içinde en düşük sıcaklık değerinin ne olduğunu bulmak istediğimiz zaman MIN() fonksiyonu oldukça faydalı bir kullanım olacaktır. Sadece sayısal alanda değil aynı zamanda da metinsel alanlarda da kullanılabilir. Bu durumda metinsel veriyi A'dan Z'ye dizip en baştaki kaydı verecektir.

MIN() Kullanım Biçimi

```
SELECT MIN(alan_adi) FROM tablo
```

Aşağıdaki gibi Ürünler tablomuz olsun

id	Urun_adi	Fiyat
1	Buzdolabı	1500
2	LCD TV	1850
3	Çamaşır Makinesi	950
4	Bulaşık Makinesi	850

Örnek1: SELECT Min(Fiyat)
FROM Urunler

Çıktısı:

Expr1000
850

Burada görüldüğü üzere alan adı Expr1000 olarak görünmekedir. Aşağıdaki örnekte bu ismi daha anlamlı hale nasıl getirildiğinigörebilirsiniz.

Örnek2: SELECT Min(Fiyat) AS EnDusukFiyat
FROM Urunler

Çıktısı:

EnDusukFiyat
850

Burada AS ile ilgili alanın adı daha anlamlı bir hale getirilmiştir.

SQL SUM() Kullanımı

SUM() fonksiyonu ile belirtilen alandaki değerlerin toplamı elde edilir. Elimizdeki ürnlerin toplam değerini bulmak için kullanılabilir. Veya elimizdeki toplam stok adedini bulabiliriz.

Sadece sayısal alanlarda kullanılabilir.

SUM() Kullanım Biçimi

```
SELECT SUM(alan_adi) FROM tablo
```

Aşağıdaki gibi Urunler tablomuz olsun

id	Urun_adi	Fiyat
1	Buzdolabı	1500
2	LCD TV	1850
3	Çamaşır Makinesi 1000 Devir	950
4	Çamaşır Makinesi 800 Devir	850

Örnek1: SELECT Sum(Fiyat)
FROM Urunler

Çıktısı:

Expr1000
5150

Burada görüldüğü üzere alan adı Expr1000 olarak görünmekedir. Aşağıdaki örnekte bu ismi daha anlamlı hale nasıl getirildiğinigörebilirsiniz.

Örnek2: SELECT Sum(Fiyat) AS Camasir_Mak_Toplam_Degeri
FROM Urunler
WHERE Urun_adi Like 'Çamaşır Makinesi%'

Çıktısı:

Camasir_Mak_Toplam_Degeri
1800

Burada AS ile ilgili alanın adı daha anlamlı bir hale getirilmiştir. Dikkat edileceği üzere WHERE kullanılarak elimizde bulunan bütün çamaşır makinelerinin toplam değeri elde edilmiş oldu.

SQL GROUP BY Kullanımı

Bir fonksiyonu kullanırken bazı durumlarda GROUP BY fonksiyonu ile belli alanlara göre gruplamak gerekebilir.

GROUP BY Kullanım Biçimi

```
SELECT alan_adi1, fonksiyon(alan_adi2)
FROM tablo
GROUP BY (alan_adi)
```

Aşağıdaki gibi Ürünler tablomuz olsun

id	Urun_adi	Fiyat
1	Buzdolabı	1500
2	LCD TV	1850
3	Çamaşır Makinesi	950
4	Çamaşır Makinesi	850

Örnek1: SELECT Urun_adi, SUM(Fiyat)
FROM Urunler
Group By Urun_adi

Çıktısı:

Urun_adi	Expr1000
Buzdolabi	1500
LCD TV	1850
Çamaşır Makinesi	1800

Örnekte Urun_adi alanına göre bir gruplama işlemi yapılmıştır. Ve her grubun fiyatı ayrı olarak toplanmıştır. Burada görüldüğü üzere alan adı Expr1000 olarak görünmekedir. Aşağıdaki örnekte bu ismi daha anlamlı hale nasıl getirildiğinigörebilirsiniz.

Örnek2: SELECT Urun_adi, Count(Urun_adi) AS Urun_adedi
FROM Urunler
Group By Urun_adi

Çıktısı:

Urun_adi	Urun_adedi
Buzdolabi	1
LDC TV	1

Çamaşır Makinesi	2

Burada ürünler gruplanıp, hangi üründen kaç tane olduğu bilgisi elde edilmiş. Burada AS ile ilgili alanın adı daha anlamlı bir hale getirilmiştir.

SQL Matematiksel Fonksiyonlar

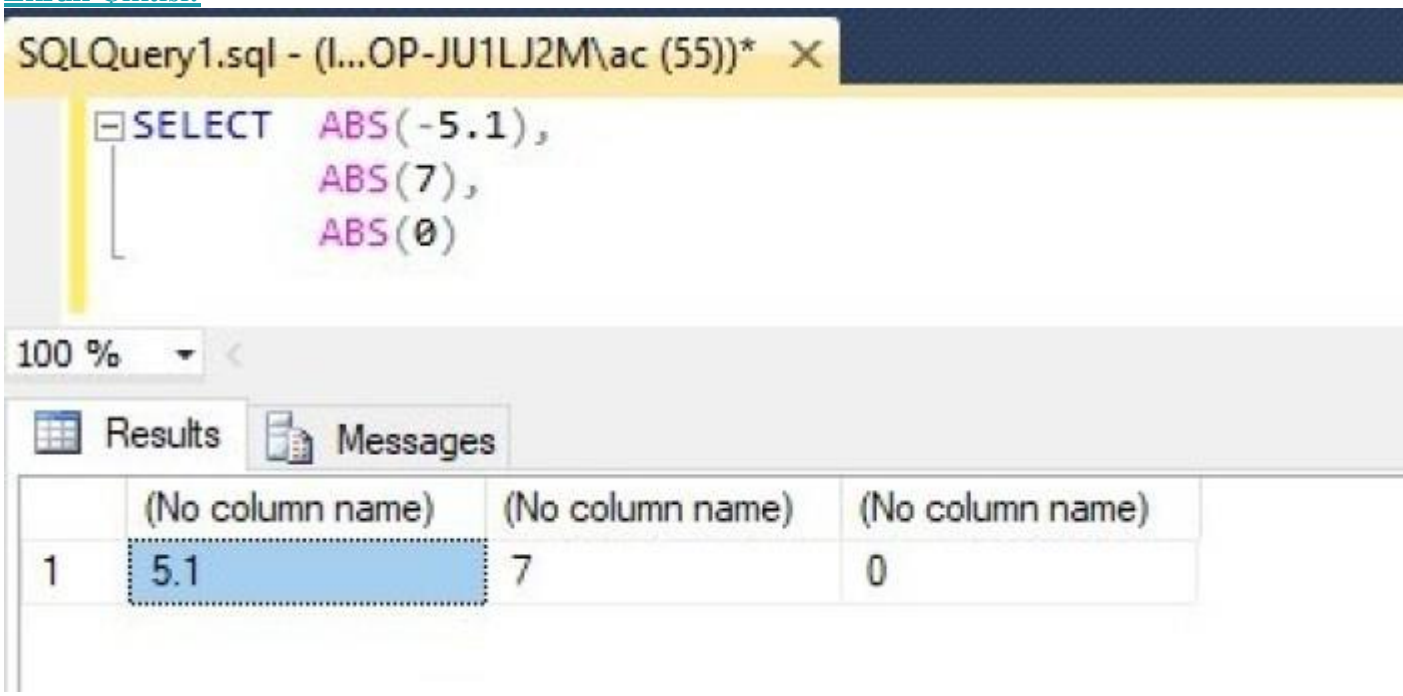
Sql‘ de verilen bir sayıyı üzerinde hesaplamaları yaparak tekrar geriye bir sayı değeri döndüren **fonksiyonlardır**. Bu yazımızda Sql’ de kullanılan **ABS(), CEILING(), FLOOR(), PI(), POWER(), RAND(), ROUND(), SIGN(), SQRT(), SQUARE(), ISNUMERIC()** fonksiyonlarının ne amaçla kullanıldığını görerek bu fonksiyonlarla ilgili örnekler oluşturacağız.

SQL – ABS() Fonksiyonu

Verilen sayının **mutlak değerini** geriye döndürür. Yani sayının pozitif halini geri döndürür.

```
1
2 SELECT ABS(-5.1),
3        ABS(7),
4        ABS(0)
5
```

Ekran Çıktısı:

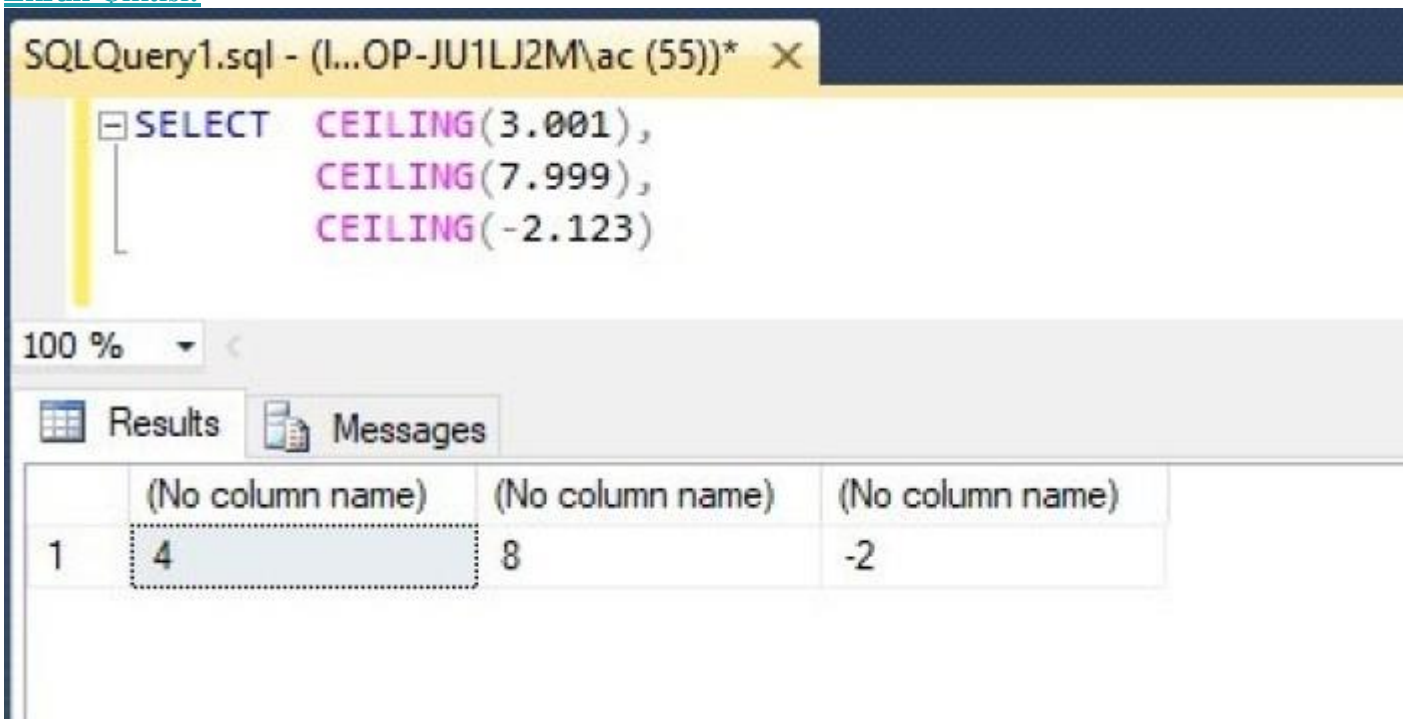


SQL – CEILING Fonksiyonu

Verilen **ondalık sayıyı** en yakın **üstüne** yuvarlar.

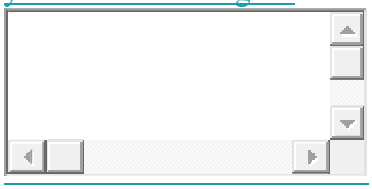
```
1
2 SELECT CEILING(3.001),
3        CEILING(7.999),
4        CEILING(-2.123)
5
```

Ekran Çıktısı:



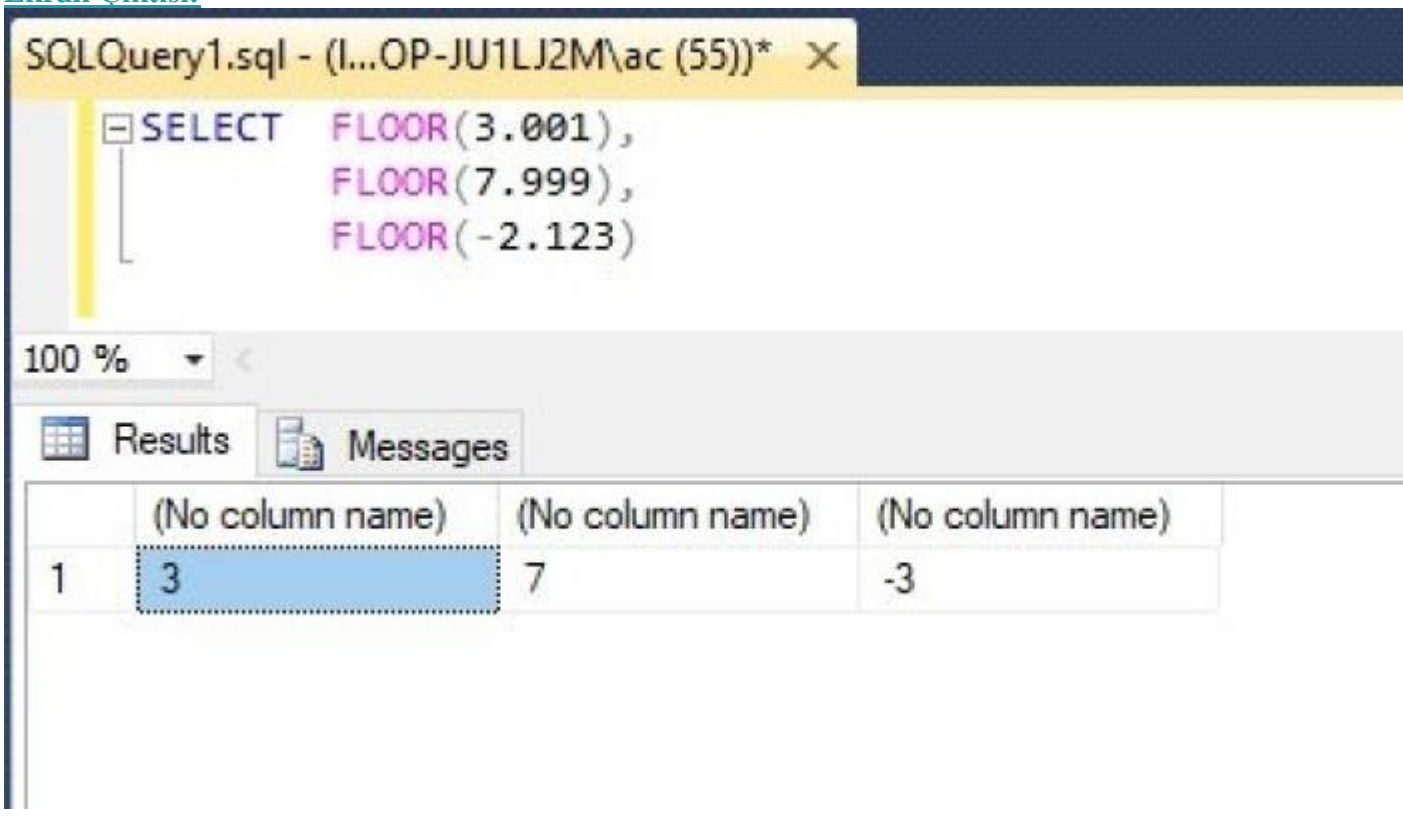
SQL – FLOOR() Fonksiyonu

Tabana yuvarlama işlemi yapar. Yani girilen sayının yakın **en küçük** tam sayıya yuvarlanmasını sağlar.



```
1

2 SELECT FLOOR(3.001),

3        FLOOR(7.999),

4        FLOOR(-2.123)

5
```

Ekran Çıktısı:



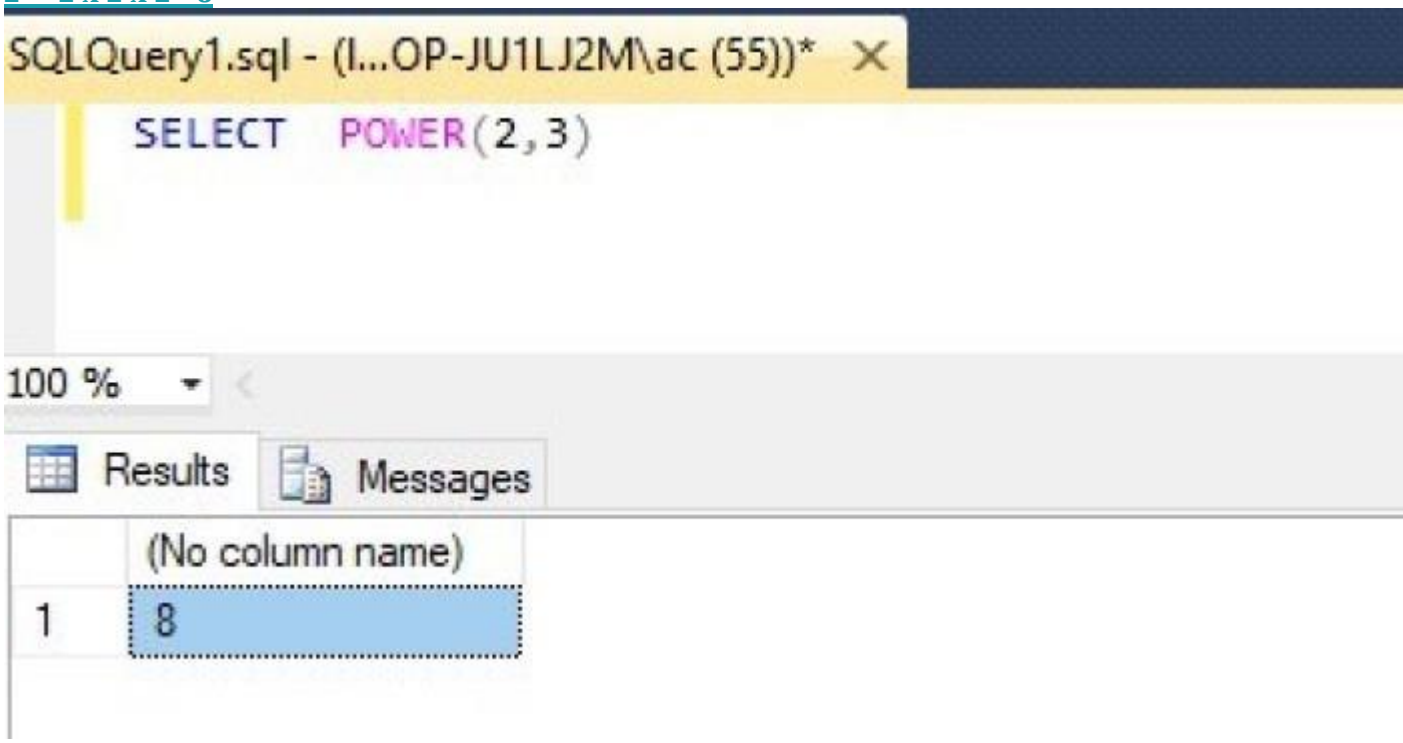
SQL – PI() Fonksiyonu

Matematikteki **pi (Π)** sayısını verir.

SQL – POWER() Fonksiyonu

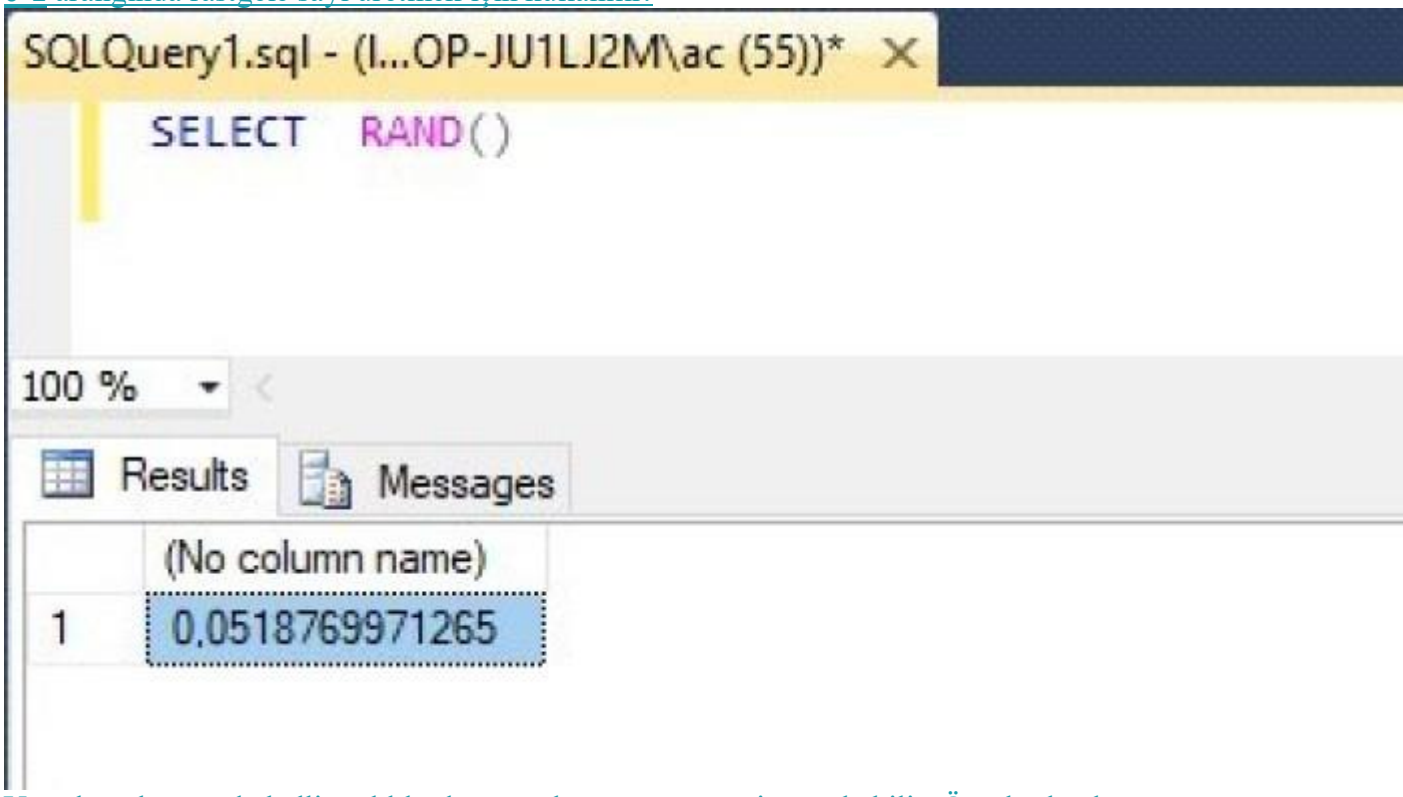
Verilen sayının, verilen değerde üssünü almak için kullanılır. Örnek olarak

$2^3 = 2 \times 2 \times 2 = 8$

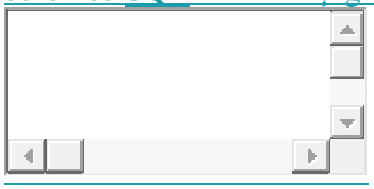


SQL- RAND() Fonksiyonu

0-1 aralığında rastgele sayı üretmek için kullanılır.



Uygulamalarımızda belli aralıklarda rastgele sayı üretmemiz gerekebilir. Örnek olarak **1-100** arası **rastgele sayı** üretmeye ihtiyaç duyduğumuzu varsayalım. Böyle bir durumda **SQL** komutu aşağıdaki gibi yazılabilir.



```
1

2 SELECT FLOOR(RAND()*100+1)

3
```

Ekran Çıktısı:

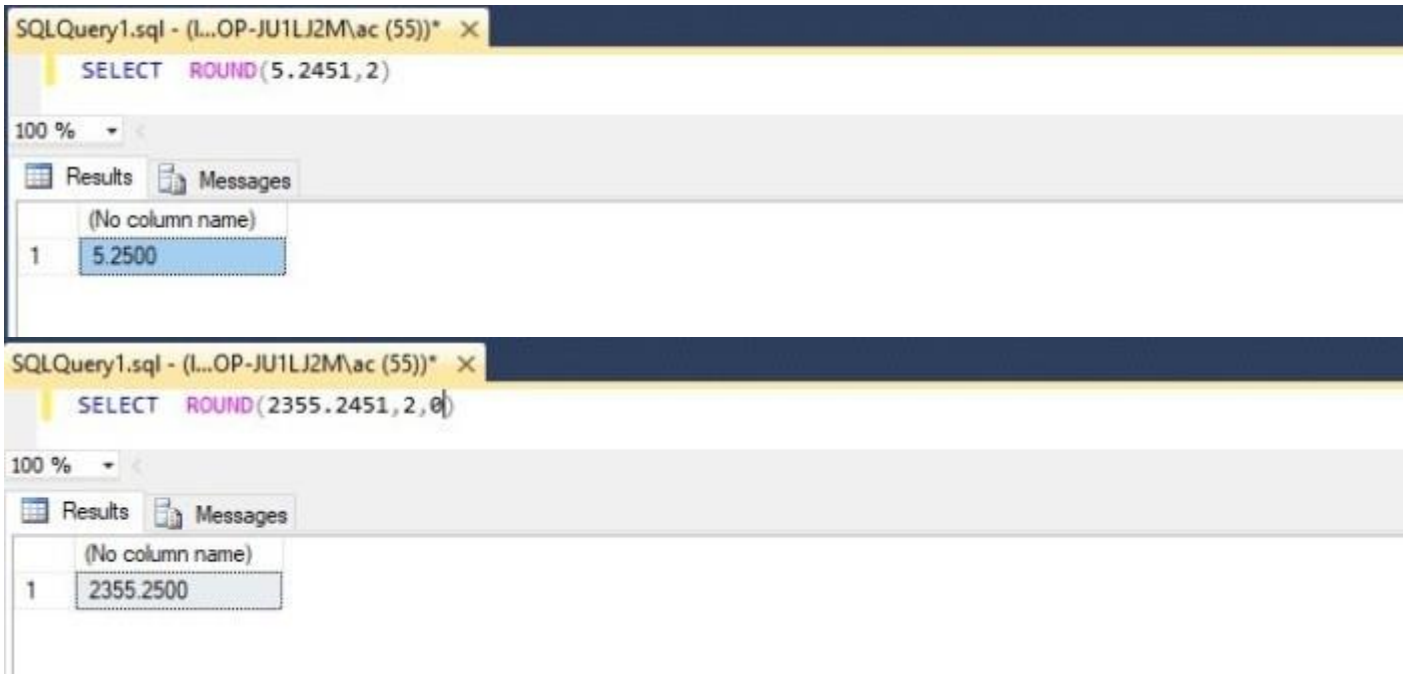

SQLQuery1.sql - (I...OP-JU1LJ2M\ac (55))*
SELECT FLOOR(RAND()*100+1)
100 %
Results
Messages
(No column name)
1
48

SQL -ROUND() Fonksiyonu

Verilen ondalıklı sayının virgülden sonra kaç basamak yuvarlanacağını belirleyebileceğiniz fonksiyondur. Bu fonksiyon 2 parametre ile kullanılabileceği gibi 3 parametre ile de kullanılabilir.

Birinci parametre yuvarlanacak olan sayıdır. İkinci parametre kaç basamağa yuvarlanacağıdır. Üçüncü parametre ise yuvarlama mı? Kırpma mı yapılacağı ile ilgilidir.

Örneklerle açıklayalım.

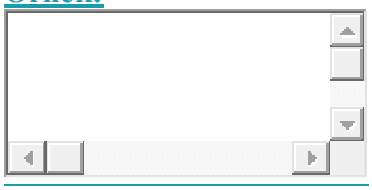


Yuvarlama değeri **negatif girilirse** sayının tam kısmı yani virgülün solundaki kısımda yuvarlanacaktır.

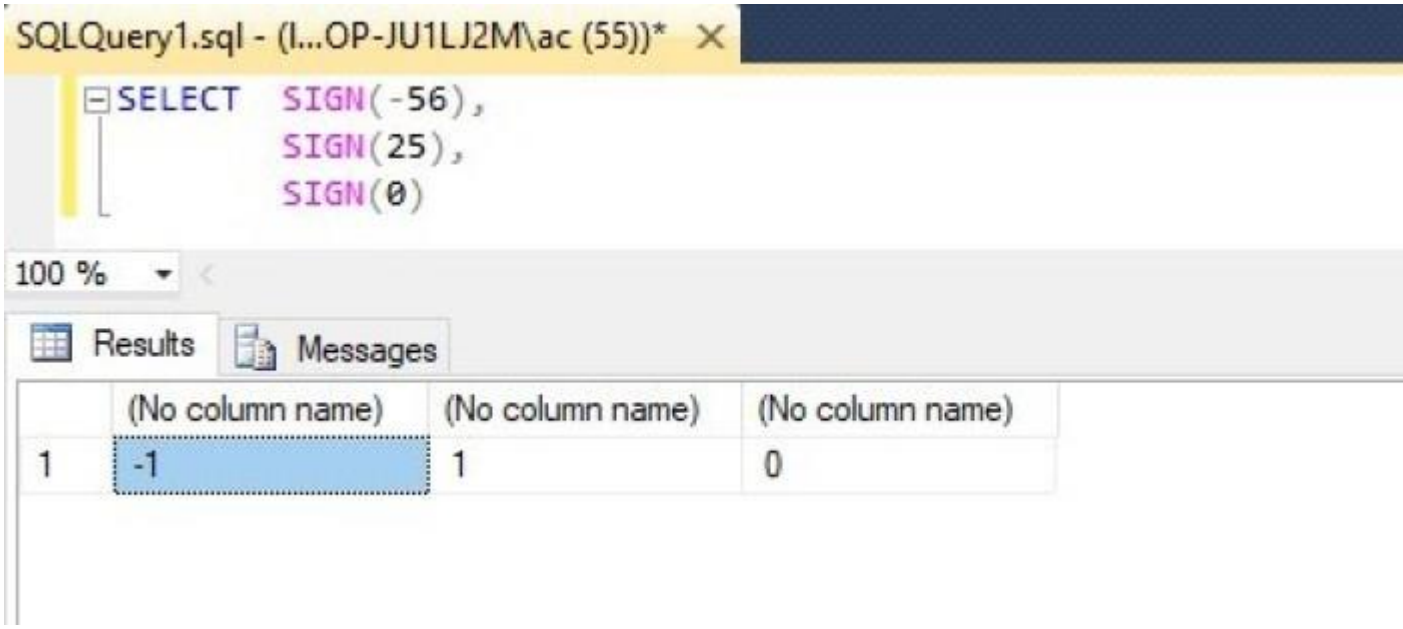
SQL – SIGN() Fonksiyonu

Bu fonksiyon veriye ait işaret bilgisini geriye döndürür.

Değer **negatifse -1,**
pozitifse 1,
Sıfırsa 0 değeri dönecektir.
Örnek:



```
1
2 SELECT SIGN(-56),
3        SIGN(25),
4        SIGN(0)
5
```



SQL – SQRT() Fonksiyonu

Verilen sayının **karekökünü** alır.

-

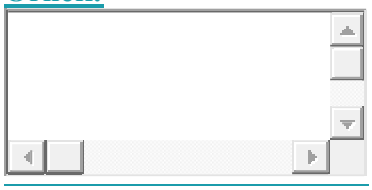
SQL – SQUARE() Fonksiyonu

Verilen sayının **karesini** döndüren fonksiyondur.

SQL – ISNUMERIC() Fonksiyonu

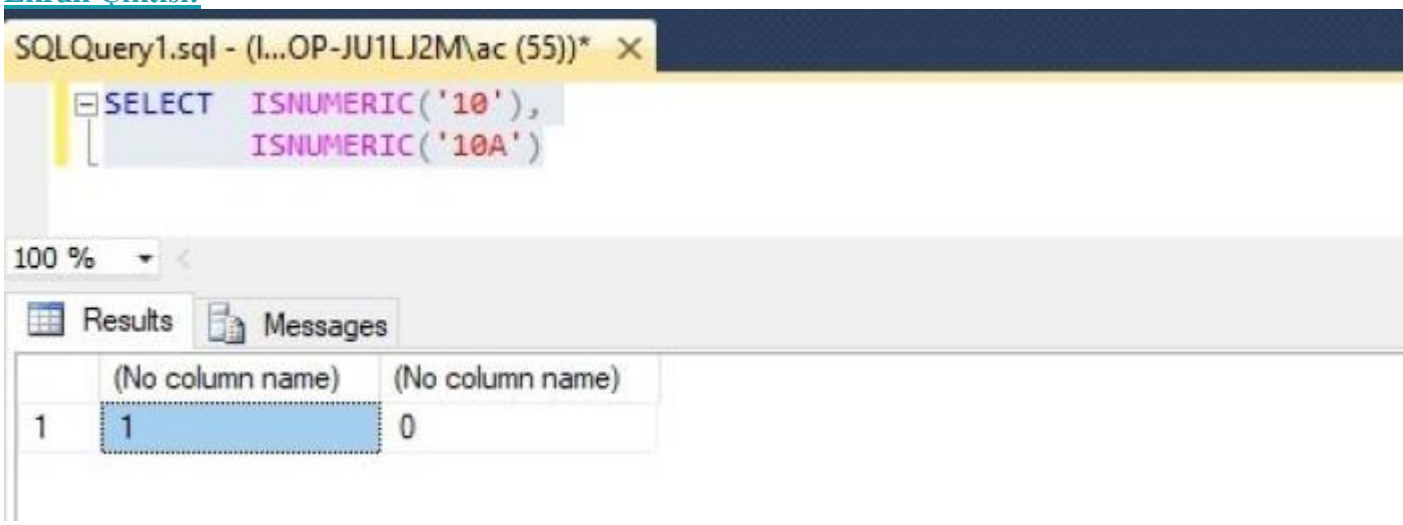
Verilen değerin **sayısal** olup olmadığı bilgisini elde etmek için kullanılır. **Sayısal**ise geriye "**1**" değeri dönecektir. Veri **sayısal değilse "0"** değeri dönecektir.

Örnek:



```
1

2 SELECT ISNUMERIC('10'),

3        ISNUMERIC('10A')

4
```

Ekran Çıktısı:



SIN : Bu fonksiyon float veri tipinden ve radyan cinsinden belirtilen bir açının sinüsünü float veri tipinden döndürür.

COS : Bu fonksiyon float veri tipinden bir ifadenin radyan cinsinden belirtilen açısının kosinüsünü döndürür.

TAN : Bu fonksiyon float veri tipinden bir ifadenin radyan cinsinden belirtilen açısının tanjantını döndürür.

COT : Bu fonksiyon float veri tipinden bir ifadenin radyan cinsinden belirtilen açısının kotanjantını döndürür.

DEGREES : Bu fonksiyon açı olarak belirtilen radyan cinsinden ifadenin derece cinsinden açısını döndürür.

EXP : Bu fonksiyon float veri tipinden belirtilen değerin üstel değerini döndürür.

FLOOR : Bu fonksiyon belirtilen değere eşit veya daha küçük en büyük tam sayıyı döndürür.

LOG : Bu fonksiyon float veri tipinden belirtilen değerin doğal logaritmasını döndürür.

LOG10 : Bu fonksiyon float veri tipinden belirtilen değerin base10 logaritmasını döndürür.

```
select ABS(-90) as “mutlak değeri -90”
select round(5.5666,3) -5.5670
select floor(5.5666) -5
select degrees(10) as “radyan 360”
select SIN(37)as “sinüs 37”
select COS(0) as “cosinüs 0”
select LOG10(10) as “log10 100”
select LOG(100) as “log 100”
select PI() as “PI SAYISI”
select POWER(2,8) as “2 üzeri 8”
select RAND(100)*100 as ” 1-100 arasi sayi”
select round(RAND()*100,0) as ” 1-100 arasi sayi”
select SQRT(9) as “9 un karekökü”
select Square(9) as “9 un karesi”
```

SQL'de sayısal değerler üzerinde işlem yapabilmek için matematiksel operatörleri kullanabiliriz. + ile toplama, – ile çıkarma, * ile çarpma, / ile bölme ve % ile bölmeden kalanı bulma (mod alma) gibi işlemleri yapabilmekteyiz.

Örnekler :

```
1  SELECT 3+5;
1  SELECT 6-1;
1  SELECT 3*5;
1  SELECT 8/2;
1  SELECT 8%3;
```