

T.C.
MİLLÎ EĞİTİM BAKANLIĞI



MEGEP

(MESLEKÎ EĞİTİM VE ÖĞRETİM SİSTEMİNİN GÜÇLENDİRİLMESİ PROJESİ)

BİLİŞİM TEKNOLOJİLERİ

İNTERNET PROGRAMCILIĞI 5

ANKARA
2007

İÇİNDEKİLER

AÇIKLAMALAR

KOD	
ALAN	Bilişim Teknolojileri
DAL/MESLEK	Web Programcılığı
MODÜLÜN ADI	İnternet Programcılığı 5
MODÜLÜN TANIMI	Bu modül gerekli ortam sağlandığında; programlama komutları yardımıyla veri tabanı işlemleri yapabilecek, konuların verildiği modüldür.
SÜRE	40/32
ÖN KOŞUL	İnternet Programcılığı 4 modülünü başarmış olmak
YETERLİK	1. Programlama içinde form ve nesnelere kullanmak
MODÜLÜN AMACI	GENEL AMAÇ: Öğrenci bu modül ile gerekli ortam sağlandığında; hazırlanan program ile veritabanı işlemleri yapabileceksiniz. Amaçlar: 1. Veritabanı programlama dilini (MySQL) kullanımını öğrenerek, uygulamalar yapabileceksiniz. 2. Veritabanı programlama diliyle etkileşimli web (php ile) uygulamaları gerçekleştirebileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	Ortam: Bilişim Teknolojileri laboratuvarı, işletme ortamı. Donanım: Projeksiyon, bilgisayar... Web programlama yazılımlarını çalıştırabilecek yeterlikte bilgisayar, bilgisayar laboratuvarı, internet bağlantısı
ÖLÇME VE DEĞERLENDİRME	<ul style="list-style-type: none">➤ Her faaliyet sonrasında o faaliyetle ilgili değerlendirme soruları ile kendi kendinizi değerlendireceksiniz.➤ Her öğrenme faaliyeti sonunda birerde uygulama faaliyeti sizden istenecektir.➤ Öğretmen modül sonunda size ölçme aracı (uygulama, soru-cevap) uygulayarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek değerlendirecektir.➤ Modül sonunda Sınıf geçme yönetmeliğine uygun olarak ölçme ve değerlendirmeye tabi tutulacaksınız.

GİRİŞ

Sevgili Öğrenci;

Programlama ile birlikte insanlar tüm verilerini bilgisayar ortamında tutmak istemişlerdir. Bunun için hemen hemen tüm programlama dillerinde dosyalama işlemi yapılabilmektedir. Bu ortamlardaki kısıtlamalar, her şey istenildiği gibi yapılamadığından, sadece veri kontrollerinde iyi olan programlama dilleri geliştirildi.

Veritabanı programları bir kurumun çalışanlarının, kimlik, adres, kurumu vb. bilgilerin tutulup, işlendiği, değişiklikler yapılan bilen programlama dilleridir.

MySQL Linux ve PHP ile çok iyi anlaşılabilen bir veritabanı programlama dilidir. Günümüzde zaten web tasarımcısı olan birisi, tasarımını bir veritabanıyla mutlaka birleştirmelidir. Çünkü artık insanlar işlerini web ortamına taşımışlardır. İnternet üzerinden alışveriş yapmak, okullara kayıt yaptırmak veya bilgilerini görmek, uçak veya otobüs bileti almak gibi işlemlerde sizin bilgileriniz hep bu veritabanlarında tutulmaktadır.

İnternet ortamında veritabanı işlemi için de bazı web programlama dilleri kendilerine uygun olan, anlaşılabilir veritabanı programlama dilini seçer. PHP bir çok veritabanı dilini destekler. MySQL de birçok web programlama dilinde çalışır. Ama sanki PHP MySQL, MySQL de PHP için geliştirilmiştir gibidir. Gibidir değil zaten öyledir. İkisi de açık kod, ucuz veya bedava olan dillerdir.

ÖĞRENME FAALİYETİ - 1

AMAÇ

Bilgileri kâğıt kalem kullanarak dosyalarda saklamak elverişli bir yöntem değildir. Bu yöntem yazılım ve programlama geliştikçe kâğıt kalem yerine bilgisayarlara bırakılmıştır. Tabii bilgisayar programlamada da özellikle veritabanı programcılığına bırakılmıştır. Veritabanı, herhangi bir programda saklayabileceğimizden çok daha fazla veriyi kolayca muhafaza edebilir. Veritabanları, bize son derece güvenilir ortamlar sunmaktadır.

Veritabanları, programcılığa açık bir yapı getirmektedir. Yazılım geliştiriciler; dünya çapında kabul görmüş standartlar sayesinde, verilere istedikleri veritabanını (SQL, MySQL gibi) kullanarak kolayca erişebilirler. Bu programları kullananlar, veritabanına ait hiçbir teknik bilgiye sahip olmadan kolayca veri girişi yapabilir, veritabanındaki verileri okuyabilir. Veritabanları milyonlarca bilgileri saklamak, işlemek amacıyla kullanılır. Veritabanı programcılığı da bu bilgilerin tutulmasını, işlenmesini sağlayan kodlardan oluşur. Bu faaliyette dünyanın en önemli veritabanı programlarından MySQL veritabanı programını öğreneceksiniz. PHP, MySQL dışında birçok veritabanı yazılımı (PostgreSQL, MsSQL, Oracle, dBase...) ile birlikte sorunsuz çalışabilmesine rağmen en yüksek performans PHP+MySQL+Apache üçlüsünden elde edilmiştir.

ARAŞTIRMA

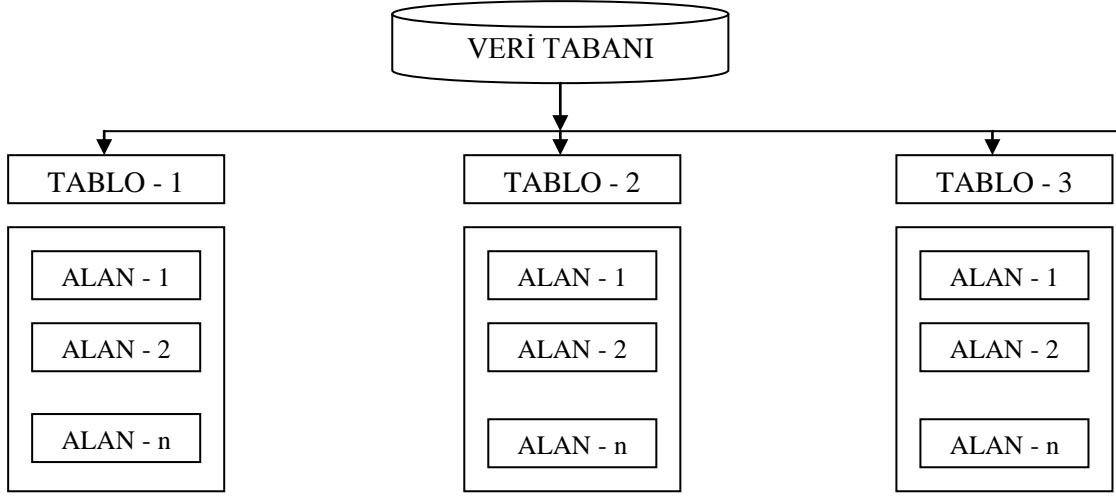
- Dünya çapında kullanılan veritabanı programlarını araştırınız.
- MySQL ve diğer veri tabanlarını avantaj ve dezavantaj olarak karşılaştırınız.
- Herhangi bir alışveriş web sitesinin nasıl veri kaydı yaptığını araştırınız.

Araştırma işlemleri için üyelik gerektiren web sitelerinin üyelik bilgilerinin nasıl tutulduğunu, alışveriş sitelerinin nasıl hazırlandığını düşünerek böyle birkaç web sitesini inceleyiniz.

1. MySQL VERİTABANI YÖNETİMİ

1.1 Veritabanı Oluşturma (Create Database)

Veritabanı (database), verilerin bilgisayar ortamında saklanması olarak tanımlanabilir. Aynı tür bilgileri içeren veriler birkaç gruba ayrılır ve bu gruplarına her birine tablo (table) denir. Veritabanı tablo veya tablolardan oluşur. Tablolar da her bir bilginin saklandığı alanlardan (fields) oluşur.



Şekil 1.1

Bir veritabanı adından da anlaşılacağı gibi bilgilerin depolandığı hiyerarşik bir yapıdır. Biz web üzerinde kullanıcı adı, şifresi, e-posta adresi gibi bilgileri bu veritabanlarında saklarız. Aynı bilgileri bir dosyaya yazıp gerektiğinde dosyayı açıp bilgileri okutmak da bir çözüm gibi görünse de bu işlem hem daha karmaşık ve zahmetli hem de daha yavaştır. Aslında az çok İngilizce bilen birisi için MySQL kullanmak pek zor olmayacaktır.



Resim 1.1

Veritabanlarında tutulacak bilgilerin doğru olarak girilmesi, tekrarlı kayıtların olmaması, verilerin güvenliği gibi özellikleri ancak bir veritabanı yöntemiyle sağlanmaktadır. Bunları sağlayan birçok veritabanı sunucusu vardır. Progress, Oracle, Sybase, ProgreSQL, MySQL, Access bunlardandır.

MySQL’de veritabanı oluşturmak oldukça kolaydır. Bunun için **CREATE DATABASE** komutu kullanılır.

Kullanımı;

CREATE DATABASE veritabanı_adi;

veritabanı_adi yerine oluşturmak istediğiniz veritabanının adını yazmanız gerekir.

Örnek; Aşağıdaki komutlarla ogrenci adlı bir veritabanı oluşturulmuştur.

```
mysql>create database ogrenci;
Query OK,1row affected (0.03 sec)
```

Bu veri tabanının kolaylıkla oluşturulduğunu ve her şeyin yolunda olduğunu gösterir.

Veritabanı Silme (Drop)

Artık ihtiyacınız olmadığına inandığınız bir veritabanını DROP DATABASE komutuyla silebilirsiniz. Bu komutu uygulamadan önce çok iyi düşünmeli ve bu veritabanına ihtiyacınızın olmadığından emin olmalısınız. Çünkü bu sorgu ile tüm tabloları, tüm indexleri, ve veritabanının kendisini silmiş olacaksınız. Yani kayıtlar geri gelmeyecektir.

Kullanımı;

```
DROP DATABASE veritabanı_adı;
```

veritabanı_adı yerine silmek istediğiniz veritabanının adını yazmanız gerekir.

Örnek; Aşağıdaki komutlarla ogrenci adlı bir veritabanı silinmiştir.

```
mysql>drop database ogrenci;  
Query OK,1 row affected (0.03 sec)
```

Bu veri tabanının kolaylıkla silindiğini gösterir.

1.1.1 MySQL Veritabanı Sunucusu

MySQL veritabanı ve sunucusunun en büyük özelliği ücretsiz oluşudur. Eğer ticari amaçla kullanılacaksa az miktar ücret ödenmesi gerekebilir. MySQL sunucusunun en büyük özelliklerinden birisi de PHP (web programlama dili) ile oldukça iyi çalışmasıdır. Sanki iki üründe bir biri için oluşturulmuş gibidir. MySQL sunucusu işletim sisteminden bağımsız olarak çalışmaktadır.

1.1.2 MySQL Kurulumu

İnternet programcılığı 1 modülünde de anlatıldığı gibi MySQL veritabanı sunucusu <http://www.mysql.com> adresinden indirilebilir. İndirilen sıkıştırılmış dosyayı herhangi bir boş klasöre açılmalıdır. Açılan dosyalardan setup.exe programı çalıştırılır. MySQL windows'ta standart olarak C:\mysql klasörüne kurulacaktır.

1.1.3 MySQL Yazım Kuralları

MySQL komut satırında dikkat edilmesi gereken kurallar;

- Komut satırında yazılan ifadelerin bittiğini belirtmek için noktalı virgül (;) kullanılır.
mysql>show databases;
- Bazı durumlarda noktalı virgül (;) kullanılmaz.
mysql>quit
- Komutlar büyük veya küçük harfle yazılabilir .
mysql>select current_time();
mysql>SELECT CURRENT_TIME();
- Komutlar birden fazla satırda yazılabilir (sonuna noktalı virgül konulan komutlar, kelimeler bölünmemek kaydıyla)

Doğru Yazım;

```
mysql> select
> current_time();
+-----+
|currnet time()|
+-----+
|20:23:56      |
+-----+
1 row in set (0.00 sec)
mysql>
```

Hatalı Yazım;

```
mysql>select current_
>time();
ERROR 1064: You have an error in your SQL syntax near `)` at line 2
mysql>
```

- Komutlarda Türkçe karakterler (ğ,Ğ,ı,İ,ş,Ş,ü,Ü,ö,Ö,ç,Ç) kullanılmaz.

1.1.4 MySQL Yeni Kullanıcı Oluşturma

MySQL veritabanı sistemi birçok farklı veritabanı destekleyebilir. Genellikle uygulama başına bir veritabanı olacaktır. MySQL’de en kolay kısımlarından biri **Veritabanı oluşturmaktır**. Bunun içinse MySQL sunucusunda tam yetkili bir kullanıcı olmanız gerekir. Kurlumu siz yaptıysanız zaten yetkili sizsinizdir. Kurulum aşamasındaki şifrenizi unutmayınız. Şimdi yeni bir yetkili kullanıcı eklemek için aşağıdaki komutlar kullanılır.

```
mysql>GRANT ALL PRIVILEGES ON *.* TO miho@localhost IDENTIFIED BY
`mihoparola` WITH GRANT OPTION;
```

MySQL sunucusuna okuma, ekleme, düzeltme ve silme hakkı olan bir kullanıcıyı eklemek için aşağıdaki komutlar kullanılır.

```
mysql>insert into user
>host,
>user,
>password,
>Select_priv,
>Insert_priv,
>Update_priv,
>Delete_priv,
>Create_priv,
>Drop_priv,
>Reload_priv,
>Shutdown_priv,
>Process_priv,
>File_priv,
>Grant_priv,
>References_priv,
>Index_priv,
>Alter_priv)
VALUES
```



```
(`#`,`ahmet`,PASSWORD(`3348`),`Y`, `Y`, `Y`, `Y`,`N`, `N`, `N`, `N`, `N`, `N`, `N`, `N`, `N`, `N`,`N`);
```

Kullanıcı adı “ahmet”, şifresi “3348” olan tüm yetkilere sahip bir kullanıcı oluşturulur. Yeni kullanıcıların tanımlarının geçerli olabilmesi için MySQL sunucusunda çıkılmalı ve aşağıdaki komut satırı uygulanmalıdır.

MySQL sunucusunda database oluşturma hakkı olan bir kullanıcı Create database komutuyla yeni bir database oluşturabilir

Kullanımı;

```
mysql>create database <veritabanı adı>;
```

Örnek;

```
mysql> create database mezunlar;  
Query OK, 1 row affected (0.00 sec)
```

Veritabanını seçme (use);

Artık mezunlar diye bir veri tabanı oluşturulmuştur. Fakat bu veri tabanını kullanabilmek için seçmemiz gerekmektedir. Veritabanını seçmek için use komutu kullanılır.

Kullanımı;

```
mysql>use <database adı>;
```

Örnek;

```
mysql>use mezunlar;  
database changed  
Mezunlar veritabanı seçilmiş oldu.
```

1.2 Tablo Oluşturma (Create Table)

Veritabanı oluşturmada bir sonraki adım, tablo oluşturmaktır. Bunu, **create table** MySQL komutunu kullanarak yapabiliriz. Tablo oluşturma işlemi yapılmadan önce mutlaka veritabanı seçilmelidir (use komutuyla). Bir **create table** ifadesinin genel formu şöyledir.



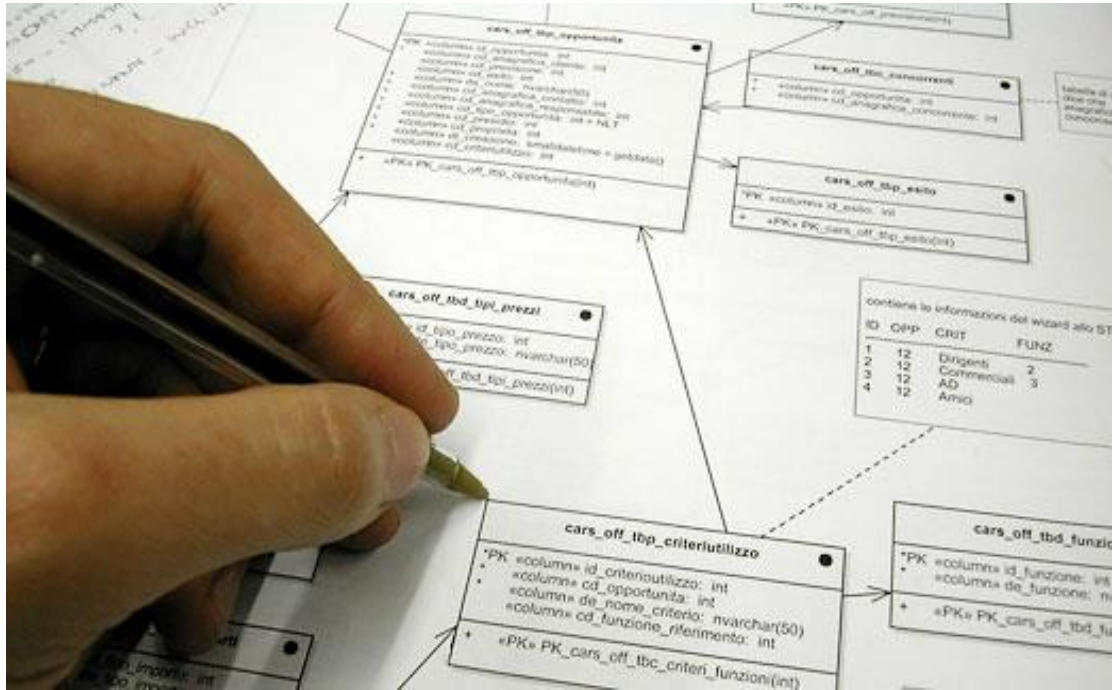
Resim 1.2

Kullanımı;

```
Create Table <tablo adı>  
alan_adi veri_turu [Default ifade][alan_kısıtlaması],  
...  
...  
[tablo kısıtlaması]);
```

Tablo Adı	:	Tabloya verilecek isim (örnek: mezunkayit),
Alan Adı	:	Tabloda yer alacak bilgi alanlarının adı (örnek: ogr_no, adi, soyadi),
Alan Veri Türü	:	O alana girilecek bilginin türü (sayı, metin, tarih),
Default İfade	:	O alan belirtilmezse başlangıçta atanacak değer,
Alan Kısıtlaması	:	O alanla ilgili kısıtlama (boş olmaması, o alana sadece E veya K girilmesi gibi kısıtlamalar),
Tablo Kısıtlaması	:	Her kaydın belirli alanlara göre kısıtlanması (kayıtların tek olmasını sağlamak, başka bir tabloyla ilişkilendirmek)

MySQL’de üç temel veri tipi vardır. Bunlar; sayısal (numeric), tarih ve saat (date and time) ve karakter katarıdır (string). Bu kategorilerin her birinin içinde de pek çok tip bulunur. Her bir tipin hafızada çeşitli depolama boyutları vardır. Veri türleri ve özelliklerini aşağıdaki tabloda görülmektedir.



Resim 1.3 Örnek veritabanı algoritması

Alan Veri Türleri		
Sayısal (Numeric) Veri Türleri		
Veri Türü	Aralık	Kapladığı Yer
TINYINT [(m)][UNSIGNED] [ZEROFILL]	-128 ile 127 arasında	1 byte
SMALLINT [(m)][UNSIGNED] [ZEROFILL]	-32768 ile 32767 arasında	2 byte
MEDIUM (m)][UNSIGNED][ZEROFIL	-8388608 ile 8388607 arasında	3 byte

L]		
INT[(m)][UNSIGNED][ZEROFILL]	-2147483648 ile 2147483647 arasında	4 byte
BIGINT [(m)][UNSIGNED][ZEROFILL]	-9223372036854775808 ile 9223372036854775807 arasında	8 byte
FLOAT	Virgülden sonra 4 yada 8 haneli	4 byte
FLOAT (m,n)	İşaretsiz Küçük Reel Sayı	4 byte
DOUBLE PRECISION [(m,n)]	İşaretsiz Normal Reel Sayı	8 byte
REAL [(m,n)]	Double ile aynıdır	8 byte
DECIMAL [(m,n)]	Double ile aynıdır	(m+n) byte
NUMERIC [(m,n)]	Decimal ile aynıdır	(m+n) byte
Tarih ve Saat (Date and Time) Veri Türleri		
Veri Türü	Aralık	Kapladığı Yer
TIMESTAMP [(m)]	O andaki sistem zamanını tutar (tarih ve saat)	4 byte
DATE	YYYY-MM-DD,YY-MM-DD,YYMMDD formatında 0000-0-00 ile 9999-12-31 arasında	4 byte
TIME	HH:MM:SS,HHMMSS,HHMM,HH	3 byte
DATETIME	YYYY-MM-DD HH:MM:SS	8 byte
Alfasayısal (Karakter-String) Veri Türleri		
Veri Türü	Aralık	Kapladığı Yer
CHAR (m)[BINARY]	M:1 ile 255 arası BINARY kullanılırsa aramalarda büyük/küçük harf arımıdır.	m byte
VARCHAR	(m) [BINARY] Char ile aynıdır, fakat kapladığı alan açısından farklıdır.	(Girilen uzunluk+1) byte
TINYTEXT(m),[BINARY]	Char ile aynıdır, fakat kapladığı yer açısından farklıdır.	(Girilen uzunluk+1) byte
TINYTEXT ve TINYBLOB	Varchar ile aynıdır. Küçük/büyük harf ayrımı vardır. Uzunluk belirtilmez.	(Girilen uzunluk+1) byte
TEXT ve BLOB	Küçük/Büyük harf ayrımı vardır. Uzunluk belirtilmez. Max. 65535 karakter	(Girilen uzunluk+2) byte
MEDIUMTEXT ve MEDIUMBLOB	Küçük/Büyük harf ayrımı vardır. Uzunluk belirtilmez. Max. 16777216 karakter.	(Girilen uzunluk + 3) byte
LONGTEXT ve LONGBLOB	Küçük/Büyük harf ayrımı vardır. Uzunluk belirtilmez. Max. 4294967295 karakter	(Girilen uzunluk+4) byte
ENUM('değer1', 'değer2', ..., 'değern')	Verilen değerlerden bir tanesi mutlaka seçilidir. (Alan kısıtlamasında kullanılır)	En fazla 2 byte
SET ('değer1', 'değer2', ..., 'değern')	Verilen değerlerden birden çok seçilebilir.	1-8 byte arasında

Tablo 1.1. Alan Veri Türü

Örnek; Mezun olan öğrenciler için mezun bilgilerinin tutulacağı bir tablo oluşturulmuş. Bu tablo için aşağıdaki alan adları bulunacaktır. Tablo adını kayıt olarak kabul edelim.

Alan adı- Açıklama

adi : Öğrenci adı,
soyadi : Öğrenci soyadı,
ogrno : Öğrenci No,

Kayıt tablosunun oluşturulması;

```
mysql> create table kayit
->( ad varchar(15) not null,
->soyad varchar(15)not null,
->ogrno smallint(5) unsigned,
->constraint ogrno_pk primary key (ogrno)
->);
Query OK, 0 rows affected (0.08 sec)
```

Örnek; Bir alışveriş sitesinde müşteri bilgilerini tutacak bir tablo için aşağıdaki alanlar bulunacaktır. Tablo adı musterisi olsun.

Alan adı - Açıklama

Musteri_id :müşteri numarası
adsoyad : Müşteri adı ve soyadı,
adres : Müşteri adresi,
sehir : Şehir,
tlfon : Telefon,

Musteri tablosunu oluşturulması;

```
mysql> create table musterisi
->(musteri_id int unsigned not null auto_increment primary key,
->adsoyad varchar(50) not null,
->adres varchar(100) not null,
->sehir varchar(15) not null,
->tlfon varchar(11) not null
->);
Query OK, 0 rows affected (0.08 sec)
```

Örnek; Bir satış sitesine konulacak ürünlerin tutulacağı **urun** adlı tabloyu aşağıdaki gibi oluşturalım.

Alan adı	Açıklama
Urun_kodu	: Satılacak ürün kodu,
Urun_markasi	: Satılacak ürünün markası,
Urun_turu	: Satılacak ürün türü,
Urun_fiyati	: Satılacak ürünün fiyatı,

Urun tablosunun oluşturulması;

```
mysql> create table urun
->( urun_kodu integer(5) zerofill not null,
->urun_markasi varchar(20)not null default 'BELİRSİZ',
->urun_turu varchar(20)not null default 'BELİRSİZ',
->urun_fiyati bigint(10) unsigned,
->constraint urun_kodu_pk primary key (urun_kodu),
->);
Query OK, 0 rows affected (0.07 sec)
```

Tabloları Silmek (Drop Table)

Bazen bir tablonun tamamından kurtulmak isteyebilirsiniz. Bunun için DROP TABLE ifadesiyle yapabilirsiniz. Genel kullanımı aşağıdaki gibidir.

Kullanımı;

DROP TABLE tablo_adi

Örnek; müşteri tablosunu silmek için aşağıdaki ifade kullanılır.

```
mysql> drop table müşteri;
```

Bu sorgu tablodaki tüm satırları ve tablonun kendisini sileceğinden, kullanırken dikkatli olunmalıdır.

1.3 Index Yapıları (Create Index);

Bir index, veri tabanı ortamında tablo gibi bir nesnedir ve ilişkili olarak kullanıldığı tablonun indexleme alanı (primary key) olarak kullanılan kolondaki verilere göre sıralanmış biçimde işleme sokulmasını sağlar. Bir tablo indexlenmiş ise bu tablo içinde gerçekleştirilecek bir arama ya da koşullu listeleme işlemi çok daha hızlı biçimde gerçekleştirilebilecektir. CREATE INDEX belirtilen bir tablo üzerinden bir indeks oluşturur. İndekslerin birincil kullanım amacı veritabanı başarımını arttırmaktır (ancak, uygunsuz kullanımı başarımın düşmesiyle sonuçlanır).

İndeks için anahtar alanları sütun isimleri olarak ya da parantez içinde yazılmış ifadeler olarak belirtilir. İndeksleme yöntemi çoksütunlu indeksleri destekliyorsa, çok sayıda alan belirtilebilir. Bir indeks alanı tablonun satırındaki bir veya daha fazla sütun değerinden hesaplanan bir ifade olabilir. Bu özellik, bazı temel veri dönüşümlerini temel alan veriye daha hızlı erişim sağlamak için kullanılabilir.

Auto_increment, tamsayı sütunlarında kullanabileceğiniz özel bir MySQL özelliğidir. Tabloya satırlar eklerken ilgili alanı boş bıraktığımızda MySQL otomatik olarak benzersiz bir tanımlayıcı değer oluşturacaktır. Bu değer, sütundaki mevcut maksimum değerleri bir fazla olacaktır. Her tabloda yalnızca bir tane bulunabilir. Auto_increment içeren sütunlar indexlenmelidir.

Bir sütun adının ardından gelen primary key, bu sütunun tablo için birincil anahtar olduğunu belirtir. Bu sütuna yapılan girişlerin benzersiz olması gerekir (TC kimlik no, okuldaki öğrenci no gibi). MySQL bu sütunu otomatik olarak indexler. Yukarıdaki müşteri tablosundaki **musteri_id** ile birlikte kullanıldığında Auto_increment ile birlikte görülür. Birincil anahtar üzerindeki otomatik index auto_increment için gereken indexle ilgilenir.

Bir sütun adının ardından birincil anahtar (primary key) belirleme işlemi sadece tek sütunluk birincil anahtarlar için yapılabilir. Birincil anahtarların belirtilmesi bu sütunlarda indexlerin oluşturulmasını sağlar.

Birincil anahtarlar ya da indexler olmadan da tablo oluşturmak mümkündür. Yeni başlayan MySQL kullanıcılarının karşılaştıkları sorunlardan biri çok hızlı olduğunu duymuş oldukları bu veritabanından yeterli performans elde edememektir. Bu performans sorunuyla karşılaşmalarının nedeni veritabanlarında hiçbir index oluşturmamış olmalarıdır.

MySQL tarafından otomatik olarak oluşturulan indexler başlangıç için işimizi görecektir. Eğer anahtar olmayan bir sütunda çok fazla sorgu çalıştırdığınızı fark ederseniz, performansı arttırmak için bu sütuna bir index eklemek isteyebilirsiniz. Bunu CREATE INDEX ifadesini kullanarak yapabilirsiniz. Bu ifadenin genel formu aşağıdadır.

Kullanımı;

```
CREATE [ UNIQUE ] INDEX isim ON tablo [ USING yöntem ]
  ( { sütun | ( ifade ) } [ işleç_sınıfı ] [, ...] )
  [ TABLESPACE tablo_alanı ]
  [ WHERE dayanak ]
```

Parametreler

UNIQUE

İndeks oluşturulurken ve her veri eklenişinde, tabloda birbirinin aynı değerler bulunmaması için sistemin sınama yapmasını sağlar. Girdilerin yinelenmesine sebep olacak bir veri girme veya güncelleme işleminin yapılmaya çalışılması bir hata üretecektir.

İsim : Oluşturulacak indeksin ismi; burada şema nitelemeli isimler kullanılamaz. İndeks daima tabloyu içeren şemada oluşturulur.

Tablo : İndekslenecek tablonun ismi (şema nitelemeli olabilir).

Yöntem : İndeks için kullanılacak yöntemin ismi. Değer olarak, btree, hash, rtree ve gist verilebilir. btree öntanımlı yöntemdir.

Sütun : Tablo sütunun ismidir.

İfade : Tablonun bir ya da daha fazla sütünü ile ilintili bir ifade. İfade yukarıdaki sözdiziminde gösterildiği gibi parantez içinde yazılmalıdır. Ancak, ifade bir işlev çağırısı biçimindeyse parantez içine alınmayabilir.

işleç_sınıfı: Bir işleç sınıfının ismidir. Ayrıntılar için aşağıya bakınız.

tablo_alanı: İndeksin oluşturulacağı tablo alanıdır. Belirtilmezse, default_tablespace yapılandırma değişkeninin değeri; bu değişkene bir değer atanmamışsa veritabanının öntanımlı tablo alanı kullanılır.

Dayanak : Bir kısmî indeks için kısıt ifadesi.

Örnekler;

films tablosunun title sütunu üzerinde bir B-tree indeksi oluşturmak için:

```
mysql> CREATE UNIQUE INDEX title_idx ON films (title);
```

films tablosunun code sütunu üzerinde bir indeks oluşturup, bu indeksin indexspace tablo alanında kalması için:

```
mysql> CREATE INDEX code_idx ON films(code) TABLESPACE indexspace;
```

```
mysql> CREATE INDEX namex ON "Depo.Dbf" (malz_adi)
```

İşletmede çalışan personeli brüt maaşlarına göre azalan sırada (yüksek maaştan düşük maaşa doğru) listelemek istenirse, brüt alanına göre aşağıdaki gibi index oluşturmak gerekir.

```
mysql> CREATE INDEX per_maas ON personel (brut DESC);
```

Bir okuldaki öğrencileri öncelikle adlarına göre, aynı adda olanları soyadlarına göre, hem adı hem soyadı aynı olanların ortalamalarına göre sıralanmış olarak (listelenmesi) istenirse aşağıdaki komutlar kullanılmalıdır.

```
mysql> CREATE INDEX ogr_ad_soyad_ort ON ogrenci (ad,soyad,ort);
```

Mevcut Index'in Silinmesi (Index Drop);

Bir tablo üzerinde tanımlanmış her hangi bir index, o tablonun veritabanından silinmesi ile otomatik olarak silinecektir. Tablo silinmeksizin, o tablo üzerinde oluşturulan indeksin içinse, DROP INDEX komutu kullanılır.

```
mysql>INDEX DROP ogr_ad_soyad_ort;
```

Böylece ogrenci tablosu üzerinde oluşturulmuş ogr_ad_soyad_ort adlı indeks, ogrenci tablosu veritabanında kaldığı halde silinecektir.

1.4 Insert Kullanımı

Bir veritabanı ile işlem yapabilmek için önce ona veri girmemiz gerekir. Bunu yapmanın en yaygın yolunda MySQL'in INSERT komutudur. Bir tablodaki her satır normalde gerçek bir nesne veya ilişkiyi tarif eder ve o satırın sütun değerleri bu gerçek nesne hakkındaki bilgileri depolar, INSERT ifadesini veritabanına veri satırları eklemek için kullanabiliriz.

Kullanımı;

```
INSERT INTO tablo [(kolon, kolon, ...)] VALUES (değer-1, değer-2, ...)
```

Ya da

```
INSERT INTO tablo [(kolon, kolon, ...)] SELECT ....
```

Görüldüğü gibi, tabloya iki türde veri nakledilebiliyor:

Örnek;

- mysql>insert into musteri(ad, soyad, adres, sehir, posta_kodu, telefon) values('Özgür','Dönmez','Arı Koop.2/2 Batıkent','Ankara','06130','03122560123');
- mysql>insert into musteri values (null,'ali er','12.cad no:3 Emek','Ankara','03123335566');
- mysql>INSERT INTO musteri (musteri_no,adsoyad,sehir) values(3,'Davut ÖZTÜRK','Mersin');
- mysql>INSERT INTO iller (cod_il) VALUES (33);
- mysql>INSERT INTO eski_uyeler VALUES('M','Karabulut','mk@karya.net','1979-03-29');
- INSERT INTO musteri VALUES ("M. Selcuk Batal" , "61" , "Refahiye" , "Erzincan");
- INSERT INTO mus_hesap VALUES ("1471" , "100" , {05/02/03});

Tablo yerine, içine veri girmek istediğimiz gerçek tablonun adını koyduğumuza ve değerlerin yerine de gerçek değerler girdiğimize dikkat edin. Bu örneklerdeki değerlerin tümü tek tırnak içine alınmıştır. MySQL'deki karakter katarlarının her zaman çift ve ya tek tırnak içinde olması gerekir. Sayılara ve tarihler tırnak gerekmez.

Insert ile birlikte sadece birkaç varyasyon daha kullanılabilir. Insert sözcüğünün sonuna LOW_PRIORITY yada DELAYED eklenebilir. LOW_PRIORITY anahtar sözcüğü, sistemin bekleyip verileri daha sonra, tablodan okunmadıkları sırada girilebileceği anlamına gelir. DELAYED anahtar sözcüğü, girilen verilerinizin tampon belleğe alınacağı anlamına gelir. Sunucu meşgulse, insert işleminin tamamlanmasını beklemek zorunda kalmadan sorgu çalıştırmaya devam edebilirsiniz.

1.5 Select Kullanımı

Bir veritabanından veri elde etmeyi (almayı); tablonun belirli kriterleriyle eşleşen satırları seçerek yaparız. Bu işi yapan MySQL komutu; SELECT komutudur. SELECT ifadesinin kullanımının birçok seçeneği ve farklı yolu vardır. Bunlar;

```
SELECT [ALL|DISTINCT] {*|alan_adi_listesi}
[INTO{OUTFILE|DUMPFİLE}'dosya_adi'export_option]
FROM tablo_adi
[WHERE koşul]
[GROUP BY alan_adi1[,alan_adi2]...]
[HAVING search-condition]
[ORDER BY siralama_alanlari]
Şeklinde bir ifadesi vardır.
```

Şimdi öncelikle isteğe bağlı cümlecisi olmayan belirli bir tablodan seçim yapan sorgulara bakalım. Tipik olarak bu öğeler tablonun sütunlarıdır. (Her türlü MySQL ifadesinin sonucu da olabilirler.) Bu sorgu müşteri tablosundaki ad ve şehir sütunları içeriklerini listeler.

```
mysql>select ad, sehir from musteri;
```

Kodları müşteri tablosundaki ad ve şehir sütunlarındaki bilgileri aşağıdaki gibi listeleyecektir.

```
+-----+-----+
|ad          |sehir      |
+-----+-----+
|Ahmet      |Mersin     |
|Ayşe      |Hatay      |
|Ahsen     |Şanlıurfa  |
+-----+-----+
```

Yukarıda da görüldüğü gibi, müşteri tablosundan seçtiğimiz (ad ve şehir) öğeleri içeren bir tablomuz oldu. Bu veri, müşteri tablosundaki tüm satırlar için gösterilmektedir.

Bir tablodan select anahtar sözcüğünün ardından listeleyerek istediğiniz sayıda sütun belirtebilirsiniz. Başka bazı öğeleri belirtmek de mümkündür. Bunlardan bir tanesi de joker karakter olan "*" işaretidir. Bu işaret belirtilmiş tablodan ve tablolardaki tüm sütunlarla eşleşir. Örneğin siparis adlı bir tablodaki tüm sütunları listelemek için aşağıdaki kodları yazılır.

```
mysql>select * from siparis; kodları siparis tablosundaki tüm sütunlarındaki bilgileri aşağıdaki gibi listeleyecektir.
```

```
+-----+-----+-----+-----+
|musteri_no  |siparis_tar |urun      | fiyat |
+-----+-----+-----+-----+
|001        |23.03.2007 |HDD       |120YTL |
|0034       |12.03.2007 |256MB RAM |85YTL |
|003        |28.03.2007 |Power Supply |28YTL |
+-----+-----+-----+-----+
```

Bir tablonun alt kümelerine erişmek için (şartlı sorgulama yapmak için), bazı seçim kriterleri belirtmemiz gerekir. Bunu **WHERE** parametresi ile yapabiliriz.

Kullanımı;

```
SELECT <alan_adi [alan_adi1,...]> FROM <tablo_adi> WHERE <koşullar>
```



```
mysql>select from musteriler where adi='ayşe';
```

musteriler tablosundan tüm sütunlara ama sadece adi ayşe olan satırları seçecektir. Bunun çıktısı:

```
+-----+-----+-----+-----+
|ad      |adres      |şehir      |telefon    |
+-----+-----+-----+-----+
|ayşe    |orhaniye mh. |MUĞLA      |2143526    |
|ayşe    |emek cd.     |ANKARA     |5214556    |
|ayşe    |güneyli köyü |MERSİN     |4522356    |
+-----+-----+-----+-----+
```

WHERE belirli satırları seçerken kullanılan ölçütleri belirtir. Bu örnekte adi ayşe olan satırları seçtik. Tek eşittir işareti, eşitliği kontrol etmek için kullanıldı. Bunun PHP’de farklı olduğunu ve bunları birlikte kullanırken karıştırılabilecekleri unutulmamalıdır.

MySQL eşitliği ek olarak operatörleri bu düzenli deyimlerin tümünü destekler. WHERE cümlecikleri içinde kullanılan operatör ve deyimler aşağıdaki tabloda verilmiştir.

WHERE Cümlecikleri İçin Kullanışlı Karşılaştırma Operatörleri			
OPERATÖRLER	AD(wardsa)	ÖRNEK	AÇIKLAMA
=	Eşittir	ogrno=125	İki değerin eşit olup olmadığını kontrol eder.
>	Büyüktür.	Fiyat>45	Bir değerin diğerinden büyük olup olmadığını kontrol eder.
<	Küçüktür.	Nt<44	Bir değerin diğerinden küçük olup olmadığını kontrol eder.
>=	Büyüktür veya eşittir.	Nt>=45	Bir değerin diğerinden büyük yada eşit olup olmadığını kontrol eder.
<=	Küçüktür veya eşittir.	Nt<=44	Bir değerin diğerinden küçük yada eşit olup olmadığını kontrol eder
!= veya <>	Eşit değildir.	Ad!=’ahmet’	İki değer eşit değil mi diye bakar
IS NOT NULL		Adres is not null	Alanın içinde değer bulunup bulunmadığını kontrol eder
IS NULL		Adres is null	Alanın içinde değer yokmu diye bakar.
BETWEEN		Nt between 50 and 90	Bir değerin minimum değerden büyük veya eşit, maksimum değerden küçük veya eşit olup olmadığını kontrol eder.
IN		Şehir in (“mersin”,”muğla”)	Bir değerin belirli bir kümede olup olmadığını kontrol eder.
NOT IN		Şehir not in (“ankara”,”istanbul”)	Bir değer bir kümede değil mi diye bakar.
LIKE	Örnek eşleştirme	İsim like (“ayşe%”)	Basit SQL eşleştirme işlemini kullanarak bir değerin bir örnekle eşleşip eşlenemediğini kontrol eder.

NOT LIKE	Örnek eşleştirme	İsim not like("oya%")	Bir değer bir örnekle eşleşiyor mu diye bakar.
----------	------------------	-----------------------	--

Tablo 1.2 WHERE Cümlecikleri İçin Kullanışlı Karşılaştırma Operatörleri

Tablodaki son iki satır like ile ilgilidir. Bunlar örnek eşleştirme formlarıdır. Like basit sql örnek (pattern) eşleştirilmesi kullanılır. Örnekler normal metin artı herhangi bir sayıdaki karakter ile eşleşebilecek joker karakteri gösteren bir % karakterinden ve birtek karakteri ile eşleşen _ (alt çizgi) karakterinden oluşabilir.

Birden fazla kriteri kontrol etmek için basit operatörleri ve örnek eşleştirme söz dizimini kullanabilir ve bunları and ve or ile daha karmaşık bir kriterler halinde birleştirebilirsiniz.

Örneğin;

```
mysql> Select * form musteriler where adi='ayşe' or adi='fatma';
```

Bu örnekte musteriler tablosunda adi ayşe veya Fatma olanları listeleyecektir .

Örnek; Adı, Halil veya Nural veya Ahmet olan öğretmenlerin adının ve adreslerinin görüntülenmesi için aşağıdaki ifade yazılır.

```
mysql>select adi, adres from ogretmen where in ('Halil','Nural','Ahmet');
```

Örnek; Soyadı "ÇATAK" ve cinsiyeti "E" olan öğrencilerin ad ve numaralarının görüntülenmesi için aşağıdaki ifade kullanılır.

```
mysql>select ad, ogrno from ogrenci where soyad='ÇATAK' and cinseyti='E';
```

Örnek; Sınavdan 45'ten küçük olan öğrencilerin ad ve numarasının görüntülenmesi için aşağıdaki ifade kullanılır.

```
mysql>select adi, ogrno from ogrenci where not1<45;
```

Örnek; Sınav notundan 70 ile 100 arası not alan öğrencilerin ad ve numarasının görüntülenmesi için aşağıdaki ifade kullanılır.

```
mysql>select adi, ogrno from ogrenci where not2 between 70 and 100;
```

Örnek; Doğum tarihi 30/03/1995 olan öğrencilerin ad ve numarasının görüntülenmesi için aşağıdaki ifade kullanılır.

```
mysql>select adi, ogrno from ogrenci where dogum_tarihi='1995/03/30';
```

Örnek; Soyadı 'ER' ile başlayan öğrencilerin ad ve numarasının görüntülenmesi için aşağıdaki ifade kullanılır.

```
mysql>select adi, ogrno from ogrenci where soyadi like 'ER%';
```

Örnek; Soyadı 'AN' ile biten öğrencilerin ad ve numarasının görüntülenmesi için aşağıdaki ifade kullanılır.

```
mysql>select adi, ogrno from ogrenci where soyadi like '%AN';
```

Örnek; Adının içinde 'M' geçen öğrencilerin ad ve numarasının görüntülenmesi için aşağıdaki ifade kullanılır.

```
mysql>select adi, ogrno from ogrenci where adi like '%M%';
```

Örnek; Adının içinde 'M' geçmeyen öğrencilerin ad ve numarasının görüntülenmesi için aşağıdaki ifade kullanılır.

```
mysql>select adi, ogrno from ogrenci where adi not like '%M%';
```

Örnek; Adının NULL (boş) olmayan öğrencilerin ad ve numarasının görüntülenmesi için aşağıdaki ifade kullanılır.

```
mysql>select adi, ogrno from ogrenci where adi is not null;
```

Bir tablo sorgulandığı zaman sonuçlar birincil anahtara (primary key) göre sıralı gelmektedir. Sonuçların sıralamasını değiştirmek mümkündür. ORDER BY bütün MySQL ifadelerinde kullanılır.

Kullanımı;

```
Select <alan adi> from <tablo adi> [where <koşullar>] [order by alan1[, alan2, ...]{asc|desc}];
```

Örnek; Öğrenci tablosundaki kayıtları öğrencilerin adına göre sıralayarak adının, soyadının ve numarasının görüntülenmesini sağlayan aşağıdaki gibi yazılır.(bu sıralamada öğrenci bilgileri adına göre a'dan z'ye doğru sıralanır.)

```
mysql>select adi,soyadi, ogrno from ogrenci where order by adi;
```

Örnek; Öğrenci tablosundaki kayıtları öğrencilerin adına göre tersten (büyükten küçüğe) sıralayarak adının, soyadının ve numarasının görüntülenmesini sağlayan aşağıdaki gibi yazılır.(bu sıralamada öğrenci bilgileri adına göre z'den a'ya doğru sıralanır.)

```
mysql>select adi, soyadi, ogrno from ogrenci where order by adi desc;
```

İki Veya Daha Fazla Tablonun Beraber Sorgulanması (JOIN)

Birden fazla tablodan aynı anda bilgi getirilmesi (alınması) gerektiğinde, ortak alanlar üzerinden birleştirme yapılır. Birleştirme işlemi koşullar bölümünde yapılır, ortak olan alanlar eşleştirilir. MySQL ifadelerinde alan isimlerinin önüne tablo adı yazılır. Tablo adı ile alan adı arasına "." (nokta) konulur.

Kullanımı;

```
SELECT alan1[,alan2,...] FROM tablo1,tablo2[tablo3,...] WHERE tablo1.alan1=tablo2.alan1
[AND tablo2.alan2=tablo2.alan2,...];
```

Örnek; müşteri ve sipariş adlı iki tablo bulunmaktadır. Müşterilerin adlarının ve sipariş verdikleri ürünlerin kodlarının, fiyatının ve tutarının görüntülenmesi için aşağıdaki ifade kullanılır.

```
mysql>SELECT müşteri.adi,siparis.urunkodu,siparis.fiyat,siparis.tutar FROM
müşteri,siparis, WHERE müşteri.müşteri_no=siparismüşteri_no;
```

Örnek; öğrenci ve notlar adlı iki tablodan Ahmet adlı öğrencinin iki yazılı bir sözlü notlarının görüntülenmesi için aşağıdaki ifade kullanılır.

```
mysql>SELECT öğrenci.adi, notlar.y1, notlar.y2, notlar.s1 FROM öğrenci,notlar,
WHERE öğrenci_no=notlar_no AND öğrenci.adi='Ahmet';
```

Tablo Adlarına Takma Ad (Alias) Verme

Tablolara başka isimlerle seslenmek çoğunlukla kullanışlı, bazen de gereklidir. Tabloların diğer adlarına lakap (aliens) denir. Bunları bir sorgunun dışında oluşturabilir ve sonrada kullanmaya devam edebilirsiniz. Çoğunlukla kısa yazılırlar ve tercihen tablo adlarının ilk harfleri kullanılır. Kullanacağımız tabloları tanımlarken ilgili tablonun lakabını tanımlamak için AS cümlecığı ekleyebiliriz. Takma ad (lakap) için FROM'dan sonra tablo adı bir boşluk ve takma ad azılır. Takma ad verildikten sonra bilgi alanları önüne bu takma ad yazılır. Takma ad ile alan adı arasına yine “.” Karakteri konulur.

Örnek; Müşterilerin adının, vermiş olduğu siparişlerin tarihlerinin ve tutarının görüntülenmesi için aşağıdaki ifade yazılır.

```
mysql>SELECT m.adi,s.urunkodu,s.fiyat,s.tutar FROM müşteri m, siparis s, WHERE
m.müşteri_no=s.müşteri_no;
```

Kayıtları Gruplama (GROUP BY)

Çoğunlukla belirli bir kümenin içine kaç satırın düştüğünü veya bazı sütunların ortalama değerlerini (örneğin öğrenci not ortalaması gibi) bilmek istersiniz. MySQL'de bu tip bir sorguyu yanıtlamada kullanılan bir dizi gruplama fonksiyonu bulunmaktadır. Bu gruplama fonksiyonları bir tabloya veya bir tablo içindeki veri gruplarına uygulanabilir. Bu gruplar üzerinde çeşitli işlemler yapılabilir. Listelenecek bilgi alanları mutlaka GROUP BY ifadesinden sonra belirtilmelidir. SUM, COUNT, AVG, MIN, MAX, gibi fonksiyonlarla kullanılan alanlar GROUP BY'dan sonra belirtilmez.

Kullanımı;

```
SELECT alan1[,alan2,...] FROM tablo1 GROUP BY alan1[,alan2,...];
```

Örnek; Her müşterinin müşteri numarasını ve yapmış olduğu siparişlerin toplam tutarının görüntülenmesi için aşağıdaki ifade yazılır.

```
mysql>SELECT müşteri_no, SUM (tutar) FROM siparis GROUP BY müşteri_no;
```

Örnek; Bir okuldaki kız ve erkek öğrenci sayısının görüntülenmesi için aşağıdaki ifade yazılır.

```
mysql>SELECT cinsiyeti, COUNT (*) FROM ongrenci GROUP BY cinsiyeti;
```

Örnek; Bir okulun her sınıfında kaç öğrenci olduğunun görüntülenmesi için aşağıdaki ifade yazılır.

```
mysql>SELECT sinif, COUNT (*) FROM sinif GROUP BY sinif;
```

Gruplar Üzerinde Koşullu Sorgulama (HAVING)

Gruplanmış veriler içerisinde belli koşula uyanlar sorgulanacaksa MySQL cümlesinde HAVING ifadesi ile koşullar yazılır. Bu tür SQL cümlelerinde WHERE ile yazılan koşullar varsa öncelikle bunlar göz önüne alınır, sonra gruplama işlemi yapılır. Gruplama sonunda da HAVING ile verilen koşullara uygun kayıtlar listelenir. HAVING ile verilecek koşullar mutlaka GROUP BY yanına yazılan alanlar olmalıdır.

Örnek; 5 ten az siparişte bulunan müşterilerin numarasının görüntülenmesi için aşağıdaki ifade yazılır.

```
mysql>SELECT musteri_no, COUNT(*)FROM siparis GROUP BY musteri_no HAVING count(*)<5;
```

1.6 Delete Kullanımı

MySQL veritabanında kayıt silmek çok kolaydır. Girilen bilgileri silmek için delete komutu kullanılır.

Kullanımı;

```
DELETE [LOW_PRIORITY] [QUICK ] [IGNORE] FROM tablo_adi  
[WHERE koşul]  
[ORDER BY satır_sütun]  
[LIMIT sayı]
```

Örnek; Tablodaki tüm verileri (satırları) silmek için aşağıdaki ifadeyi yazmak yeterlidir. Tüm veriler silineceğinde dikkatli kullanılmalıdır.(musteri tablosundaki tüm bilgiler silinir.)

```
mysql> delete * from musteri;
```

Örnek; Aşağıdaki SQL komutu kullanıcılar tablosundaki bolum bilgisi “bilgisayar” olan tüm kullanıcılar ile ilgili satırları silecektir.

```
mysql> delete * from kullanicilar where bolum='bilgisayar';
```

Örnek; Aşağıdaki SQL komutu öğrenci tablosundaki ad bilgisi “mesut” olan tüm öğrenciler ile ilgili satırları silecektir.

```
mysql> delete * from ogrenci where ad='mesut';
```

1.7 Update Kullanımı

Veritabanından veri almanın yanı sıra, çoğunlukla bu verileri değiştirmek de isteriz. Örneğin öğrenci veritabanındaki öğrenci veli adres bilgisini değiştirmek isteyebiliriz. Bu ve benzeri güncelleme (değiştirme) işlemleri için **UPDATE** ifadesi kullanılır.

Kullanımı;

```
UPDATE tablo_adi SET ala1=değer1,[,alan2=değer2,...] [WHERE koşul cümlesi] ;
```

Temel olarak tablo_adi adlı tabloyu güncellemek, adlandırılan sütunların her birini uygun deyimle ayarlamaktır. Bir UPDATE, bir WHERE cümlecisi kullanılarak belirli satırlarla sınırlandırılabilir.

NOT; Burada Where ifadesi kullanılmaz ise tablodaki bütün kayıtlar güncellenir, koşul yazılırsa o koşula uygun kayıtlar güncellenir. Koşul kısmına dikkat edilmesi gereklidir, çünkü istenmeyen sonuçlar doğurabilir. Bir kayıttaki düzeltme yapacağınız yere bütün kayıtların içeriklerini değiştirebilirsiniz.

Örnek; Bir kitap satışı yapan mağaza kitap fiyatlarını %15 arttırmak istiyor. Bunun için kitap veritabanındaki fiyat alanı üzerinde güncelleme yapmak gerekecektir.

```
mysql>update kitap set fiyat=fiyat*1.15;
```

Örnek; Öğrenci numarası 127 olan öğrencinin doğum tarihini 03/12/1997 olarak değiştirecek MySQL kodu aşağıdaki gibi olacaktır.

```
mysql>update ogrenci set dogum_tarihi='1995/03/12'  
mysql>where ogrno=127;
```

1.8 Alter Kullanımı

Alter Table (Tablo Güncelleme)

Alter Table; Bir tablonun tanımını değiştirir. Veritabanındaki satırları güncellemeye ek olarak, veritabanı içindeki tabloların yapısını da değiştirmek gerekebilir. Bu amaç için, ALTER TABLE ifadesi kullanılır. MySQL ile bir tabloda istediğiniz kadar değişiklik yapabilirsiniz. Değiştirme cümleciklerinin her biri tablonun farklı yönlerini değiştirmek için kullanılabilir. Alter Table'nin çeşitli alt kullanımları vardır.

Kullanımı;

ALTER [IGNORE] TABLE *tbl_name*
alter_specification [, *alter_specification*] ...

alter_specification:

table_option ...
 | ADD [COLUMN] *column_definition* [FIRST | AFTER *col_name*]
 | ADD [COLUMN] (*column_definition*,...)
 | ADD {INDEX|KEY} [*index_name*] [*index_type*] (*index_col_name*,...)
 | ADD [CONSTRAINT [*symbol*]]PRIMARY KEY [*index_type*] (*index_col_name*,...)
 | ADD [CONSTRAINT [*symbol*]]UNIQUE [INDEX|KEY] [*index_name*] [*index_type*]
 (*index_col_name*,...)
 | ADD [CONSTRAINT [*symbol*]]FOREIGN KEY [*index_name*]
 (*index_col_name*,...)[*reference_definition*]
 | ALTER [COLUMN] *col_name* {SET DEFAULT *literal* | DROP DEFAULT}
 | CHANGE [COLUMN] *old_col_name* *column_definition* [FIRST|AFTER *col_name*]
 | MODIFY [COLUMN] *column_definition* [FIRST | AFTER *col_name*]
 | DROP [COLUMN] *col_name*
 | DROP PRIMARY KEY
 | DROP {INDEX|KEY} *index_name*
 | DROP FOREIGN KEY *fk_symbol*
 | DISABLE KEYS
 | ENABLE KEYS
 | RENAME [TO] *new_tbl_name*
 | ORDER BY *col_name* [, *col_name*] ...
 | CONVERT TO CHARACTER SET *charset_name* [COLLATE *collation_name*]
 | [DEFAULT] CHARACTER SET *charset_name* [COLLATE *collation_name*]
 | DISCARD TABLESPACE
 | IMPORT TABLESPACE

index_col_name:

col_name [(*length*)] [ASC | DESC]

index_type:

USING {BTREE | HASH}

ALTER TABLE İfadesiyle Yapılabilen Değişiklikler	
Söz Dizimi	Açıklama
ADD [COLUMN] <i>column_definition</i> [FIRST AFTER <i>col_name</i>]	Belirtilen konuma yeni bir sütun ekler. <i>column_definition</i> 'ın bir ad ve tipe ihtiyaç duyar.
ADD [COLUMN] (<i>column_definition</i> ,...)	Tablonun sonuna bir veya daha fazla sütun ekler.
ADD {INDEX KEY} [<i>index_name</i>] [<i>index_type</i>] (<i>index_col_name</i> ,...)	Tabloya belirtilen sütun veya sütunlar üzerinde bir index oluşturur.
ADD [CONSTRAINT [<i>symbol</i>]]PRIMARY KEY [<i>index_type</i>] (<i>index_col_name</i> ,...)	Belirtilen sütun veya sütunları tablonun birincil anahtarı yapar.
ADD [CONSTRAINT [<i>symbol</i>]]UNIQUE [INDEX KEY] [<i>index_name</i>] [<i>index_type</i>] (<i>index_col_name</i> ,...)	Belirtilen sütun ve sütunlar üzerinde tabloya benzersiz bir index ekler.
ADD [CONSTRAINT [<i>symbol</i>]]FOREIGN KEY [<i>index_name</i>]	Bir InnDB tablosuna yabancı bir anahtar ekler.

<code>(index_col_name,...)[reference_definition]</code>	
<code>ALTER [COLUMN] col_name {SET DEFAULT literal DROP DEFAULT}</code>	Belirli bir sütunun varsayılan değerini ekler veya kaldırır.
<code>CHANGE [COLUMN] old_col_name column_definition [FIRST AFTER col_name]</code>	Column adlı sütunu, listelenmiş tanıma sahip olacak şekilde listeler.
<code>MODIFY [COLUMN] column_definition [FIRST AFTER col_name]</code>	CHANGE benzer. Adları değil, sütun tiplerini değiştirmek için kullanılır.
<code>DROP [COLUMN] col_name</code>	Belirtilen sütunu siler
<code>DROP PRIMARY KEY</code>	Birincil indexi siler.
<code>DROP {INDEX KEY} index_name</code>	Belirtilen indexi siler.
<code>DROP FOREIGN KEY fk_symbol</code>	Yabancı anahtarı siler.
<code>DISABLE KEYS</code>	İndex güncelleme özelliğinin etkinliğini kaldırır.
<code>ENABLE KEYS</code>	İndex güncelleme özelliğinin etkinleştirir.
<code>RENAME [TO] new_tbl_name</code>	Bir tabloyu yeniden adlandırır.
<code>ORDER BY col_name [, col_name] ...</code>	Tabloyu satırlar belirli bir şekilde sıralanmış şekilde yeniden oluşturur.
<code>CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]</code>	Metin tabanlı sütunların hepsini belirtilen kümesine dönüştürür.
<code>[DEFAULT] CHARACTER SET charset_name [COLLATE collation_name]</code>	Varsayılan karakter kümesini ayarlar.
<code>DISCARD TABLESPACE</code>	Bir InnDB dosyasının temelini oluşturan tablespace dosyasını siler.
<code>IMPORT TABLESPACE</code>	Bir InnDB dosyasının temelini oluşturan tablespace dosyasını yeniden oluşturur..

Tablo 1.3 Alter Table İfadesiyle Yapılabilen Değişiklikler

Örnek; Müşteri tablosunda adların 25 karaktere kadar uzun olmasına izin verilmiş olsun. Verileri almaya başladıktan sonra bazı isimlerin çok uzun olduğu görülsün. Bu durumu, sütunu 50 karakter uzunluğunda olacak şekilde değiştirilerek düzeltilebilir.

```
mysql>alter table musteriler modify ad char(50) not null;
```

Çok sık ortaya çıkan bir durum, bir sütun ekleme ihtiyacıdır.

Örnek; Müşteri tablosunda müşterilerin e-mail adreslerini de almak istediğimiz düşünelim. Ve tablomuzda böyle bir sütun önceden yoktu. Bu durumu, yeni bir sütun eklenecektir.

```
mysql>alter table musteriler add email char(50) not null;
```

Eklenen bir sütundan kurtulma isteği de yine çok sık karşılaşılan bir durumdur. Yukarıda eklediğimiz sütunu aşağıdaki gibi silebiliriz.

```
mysql>alter table musteriler drop email;
```

Örnek; Öğrenci kayıt bilgileri için oluşturulmuş olan öğrenci tablosunun adını, ogr olarak değiştirmek için şu kodlar yazılır..


```
mysql>alter table ogrenci rename ogr;
```

Örnek; Bir kirtasiye veritabanındaki kitap adlı alan adını book olarak değiştirelim.

```
mysql>alter table kirtasiye change kitap book;
```

1.9 MySQL Fonksiyonları

1.9.1 Standart fonksiyonlar

AVG()

Verilen alanın aritmetik ortalamasını alır.

```
mysql>SELECT AVG(yas) FROM employee where bolum='Halkla İlişkiler';
```

Yukarıdaki sorgu halkla ilişkiler bölümünde çalışan personelin yaş ortalamasını verecektir. AVG() fonksiyonu istenen alandaki verilerin aritmetik ortasını görüntüler.

SUM()

Verilen alanların toplamını alır.

```
mysql>SELECT SUM(maas) FROM employee WHERE bolum='Bilgi İşlem';
```

Yukarıdaki sorgu bilgi işlem bölümünde çalışan tüm personelin maaşlarının toplamını görüntüler.

MIN()

Verilen alanın kayıtlar arasındaki en küçük değerini verir.

```
mysql>SELECT MIN(maas) AS enazmaas FROM employee WHERE bolum='Bilgi İşlem';
```

Yukarıdaki sorgu bilgi işlem bölümünde çalışan ve en düşük maaşı alan personeli görüntüler.

MAX()

Verilen alanın kayıtlar arasındaki en büyük değerini verir.

```
mysql>SELECT MAX(maas) FROM employee WHERE bolum='Bilgi İşlem';
```

Yukarıdaki sorgu bilgi işlem bölümünde çalışan ve en yüksek maaşı alan personelin maaşını görüntüler.

COUNT()

Verilen alanın kayıtlar arasında kaç defa yazıldığını bulur

Eğer şirketin basın yayın bölümünde çalışan evli ve maaşı 500 YTL'den az olan kişilerin sayısını öğrenmek istiyorsak aşağıdaki sorgu işimizi görecektir.

```
mysql>SELECT COUNT(*) FROM employee WHERE maas<500 AND medeni_hal='evli' AND bolum='Basın Yayın'
```

ROUND ()

Verilen alanın virgülden sonraki değerini yuvarlarlar. Tamsayı kısmını verir.

```
mysql> select round(15.56);  
>16
```

1.9.2 Tarih Ve Zaman Fonksiyonları

ADDDATE

Bu fonksiyon tarih hesaplama işlemleri için kullanılır. Bu fonksiyonla aynı çalışan DATE_SUB() DATE_ADD() SUBDATE() Fonksiyonları vardır. ADDDATE() ve SUBDATE(), DATE_ADD() ve DATE_SUB() fonksiyonlarının eş anlamlıdır. Bunu kullanmanız için yazım formatının sağ tarafındaki tip DATE veya DATETIME olmalıdır. Burada kullanılan **tarih**, DATETIME (tarih zaman) ya da DATE (tarih) değeridir. Bunlar başlangıç değerini belirler. Yazım Formatı INTERVAL değerini belirler... Bu değer başlangıç değerine eklenir veya çıkarılır.



Resim 1.4

Yazım Formatı; bir işaretler grubudur... Bu negatif INTERVAL'ler için '-' ile başlayabilir. **tip**; ise bir anahtar kelimedir... Bu yazım formatının nasıl yorumlanacağını gösterir. EXTRACT(tip FROM tarih) fonksiyonuyla INTERVAL tipini öğrenebilirsiniz.

Yazım Formatında; Gün, Ay, Yıl, Saat, Dakika, Saniye gibi Değerleri Ayırmada Kullanılan İşaretler Yerine, İstedığınız Noktalama İşaretlerini Kullanabilirsiniz. Aşağıda Verilen Parametreler Bölümündeki Yazım Şekli Önerilendir. 'Saatler: Dakikalar: Saniyeler' Yerine 'Saatler. Dakikalar. Saniyeler' veya 'Saatler, Dakikalar, Saniyeler' gibi kullanabilirsiniz.

Fonksiyona girilen tarih değeri DATE ise ve hesaplamalarda kullandığınız tip YEAR, MONTH ve DAY ise (Yani TIME bölümleri değilse); hesaplamalar sonucunda döndürülen değer DATE formatındadır.

Bunun dışındakilerde döndürülen değer TIME formatındadır.

Kullanımı:

ADDDATE(tarih,INTERVAL expr tip)

ADDDATE(expr,gün)

ADDDATE(tarih, INTERVAL yazım formatı tip)

Bu fonksiyonla aynı çalışan diğer fonksiyonlar

DATE_SUB(tarih, INTERVAL yazım formatı tip)

SUBDATE(tarih, INTERVAL yazım formatı tip)

DATE_ADD (tarih, INTERVAL yazım formatı tip)

Örnek;

```
mysql> SELECT DATE_ADD('1998-01-02', INTERVAL 31 DAY);  
-> '1998-02-02'  
mysql> SELECT ADDDATE('1998-01-02', INTERVAL 31 DAY);  
-> '1998-02-02'
```

Parametreler;

Tip Adı	YAZIM FORMATI
MICROSECOND	: Mikrosaniyeler
SECOND	: Saniyeler
MINUTE	: Dakikalar
HOUR	: Saatler
DAY	: Günler
WEEK	: Haftalar
MONTH	: Aylar
QUARTER	: Çeyrekler
YEAR	: Yıllar
SECOND_MICROSECOND	: Saniyeler.Mikrosaniyeler
MINUTE_MICROSECOND	: Dakikalar.Mikrosaniyeler
MINUTE_SECOND	: Dakikalar:Saniyeler
HOUR_MICROSECOND	: Saatler.Mikrosaniyeler
HOUR_SECOND	: Saatler:Dakikalar:Saniyeler
HOUR_MINUTE	: Saatler:Dakikalar
DAY_MICROSECOND	: Günler.Mikrosaniyeler
DAY_SECOND	: Günler Saatler:Dakikalar:Saniyeler
DAY_MINUTE	: Günler Saatler:Dakikalar
DAY_HOUR	: Günler Saatler
YEAR_MONTH	: Yıllar-Aylar

Tablo 1.4 Tarih zaman parametreleri

Örnek;

```
<?php  
$tarih = '2005-01-31';  
$sql = mysql_query("SELECT ADDDATE('$tarih', INTERVAL 1 DAY) AS tarih");  
$sonuc = mysql_fetch_object($sql);  
echo $sonuc->tarih;  
//Sonuç: 2005-02-01  
?>
```

ADDTIME

ADDTIME(expr,expr2)

ADDTIME() Fonksiyonu expr2 değerini expr değerine ekler ve sonucu döndürür...

Kullanımı;

expr : TIME (zaman) ya da DATETIME (tarihzaman) olabilir..

expr2 : ise sadece TIME (zaman) dır...

ADDTIME(expr,expr2)

Örnek;

```
mysql> SELECT ADDTIME('1997-12-31 23:59:59.999999', '1 1:1:1.000002');  
-> '1998-01-02 01:01:01.000001'
```

```
mysql> SELECT ADDTIME('01:00:00.999999', '02:00:00.999998');  
-> '03:00:01.999997'
```

CURDATE

Bu günün tarihini 'YYY-MM-DD' veya 'YYYYMMDD' formatında döndürür.

Kullanımı;

```
mysql> SELECT CURDATE();  
-> '1997-12-15'  
mysql> SELECT CURDATE() + 0;  
-> 19971215
```

CURRENT_DATE

Bu günün tarihini 'YYYY-MM-DD' veya 'YYYYMMDD' formatında verir. CURRENT_DATE ve CURRENT_DATE(), CURDATE() fonksiyonunun eşanlamlıdır.

Kullanımı;

```
mysql> SELECT CURRENT_DATE();  
-> '2005-10-03'  
mysql> SELECT CURRENT_DATE() + 0;  
-> 20051003
```

CURRENT_TIME

CURRENT_TIME ve CURRENT_TIME() Fonksiyonları; CURTIME Fonksiyonunun eşanlamlısıdır.

CURRENT_TIMESTAMP

CURRENT_TIMESTAMP ve CURRENT_TIMESTAMP(), NOW() Fonksiyonunun Eşanlamlıdır. Bunlar şimdiki tarihi ve zamanı verirler.

Kullanımı;

```
mysql>select CURRENT_TIMESTAMP();  
->2005-10-03 14:50:59  
mysql>select CURRENT_TIMESTAMP;  
->2005-10-03 14:50:59
```

CURTIME

Geçerli saati 'HH: MM: SS' veya 'HHMMSS' formatında döndürür.

Kullanımı:

```
mysql> SELECT CURDATE();
-> '1997-12-15'
mysql> SELECT CURDATE() + 0;
-> 19971215
```

DATE

DATE (Tarih) veya DATETIME (Tarih Zaman) İfadesinden Tarihi Çıkarır. Bu fonksiyon MySQL 4.1.1 de kullanılır.

Kullanımı:

```
mysql> SELECT DATE('2003-12-31 01:02:03');
-> '2003-12-31'
```

DATEDIFF

İki tarih arasındaki farkı alır.

Kullanımı:

```
mysql> SELECT DATEDIFF('1997-12-31 23:59:59','1997-12-30');
-> 1
mysql> SELECT DATEDIFF('1997-11-30 23:59:59','1997-12-31');
-> -31
```

DATE_ADD

Bu fonksiyon tarih hesaplama işlemleri için kullanılır.

Bu fonksiyonla aynı çalışan DATE_SUB(), ADDDATE() ,SUBDATE() Fonksiyonları vardır.ADDDATE() ve SUBDATE(), DATE_ADD() ve DATE_SUB() fonksiyonlarının eş anlamlıdır.

Yazım Formatı INTERVAL değerini belirler. Bu değer başlangıç değerine eklenir veya çıkarılır.

Yazım Formatında; Gün, Ay, Yıl, Saat, Dakika, Saniye gibi Değerleri Ayırmada Kullanılan İşaretler Yerine, İstedığınız Noktalama İşaretlerini Kullanabilirsiniz.. Aşağıda Verilen Parametreler Bölümündeki Yazım Şekli Önerilendir. 'Saatler: Dakikalar: Saniyeler' Yerine 'Saatler. Dakikalar. Saniyeler' veya 'Saatler, Dakikalar, Saniyeler' gibi kullanabilirsiniz. Fonksiyona girilen tarih değeri DATE ise, ve hesaplamalarda kullandığınız tip YEAR, MONTH ve DAY ise hesaplamalar sonucunda döndürülen değer DATE formatındadır. Bunun dışındakilerde döndürülen değer TIME formatındadır.

```
DATE_ADD(date,INTERVAL expr type)
DATE_ADD (tarih, INTERVAL yazım formatı tip)
```

```
Bu fonksiyonla aynı çalışan diğer fonksiyonlar
DATE_SUB(tarih, INTERVAL yazım formatı tip)
ADDDATE(tarih, INTERVAL yazım formatı tip)
SUBDATE(tarih, INTERVAL yazım formatı tip)
```

Örnek: MySQL'de Tarih Hesaplama İşlemleri

```

<?php
mysql> SELECT "1997-12-31 23:59:59" + INTERVAL 1 SECOND;
-> 1998-01-01 00:00:00 ->Verilen DATETIME Değerine 1 Saniye Ekleniyor...
mysql> SELECT INTERVAL 1 DAY + "1997-12-31";
-> 1998-01-01 -> Verilen DATE Değerine 1 Gün Ekleniyor
mysql> SELECT "1998-01-01" - INTERVAL 1 SECOND;
-> 1997-12-31 23:59:59 -
> Verilen DATE veya DATETIME Değerine 1 Saniye Ekleniyor ve Sonucu DATETIME
Formatında Veriyor.
mysql> SELECT DATE_ADD("1997-12-31 23:59:59",INTERVAL 1 SECOND);
-> 1998-01-01 00:00:00 -> Verilen DATETIME Değerine 1 Saniye Ekleniyor
mysql> SELECT DATE_ADD("1997-12-31 23:59:59", INTERVAL 1 DAY);
-> 1998-01-01 23:59:59 ->Verilen DATETIME Değerine 1 Gün Ekleniyor
mysql> SELECT DATE_ADD("1997-12-31 23:59:59",
INTERVAL "1:1" MINUTE_SECOND);
-> 1998-01-01 00:01:00 -
> Verilen DATETIME Değerine 1 Dakika 1 Saniye Ekleniyor
mysql> SELECT DATE_SUB("1998-01-01 00:00:00",
INTERVAL "1 1:1:1" DAY_SECOND);
-> 1997-12-30 22:58:59 -
> Verilen DATETIME Değerinden 1 Gün 1 Saat:1Dakika:1 Saniye Çıkarılıyor
mysql> SELECT DATE_ADD("1998-01-01 00:00:00",
INTERVAL "-1 10" DAY_HOUR);
-> 1997-12-30 14:00:00 -
> Verilen DATETIME Değerinden 1 Gün, 1 Saat Çıkarılıyor
mysql> SELECT DATE_SUB("1998-01-02", INTERVAL 31 DAY);
-> 1997-12-02 -> Verilen DATE Değerinden 31 Gün Çıkarılıyor

?>

```

Örnek: DATE_ADD() Kullanım Örnekleri

```

<?php
/*
MySQL'deki tarih alanı datetime olarak ayarlandığını varsayarsak....
+-----+
| tarihalani      |
+-----+
| 2004-12-31 23:59:59 |
+-----+
*/
$sorgu = mysql_query("SELECT tarihalani + INTERVAL 1 SECOND"); //Tarihe 1 Sa
niye Ekler
// '2005-01-01 00:00:00'
$sorgu = mysql_query("SELECT INTERVAL 1 DAY + tarihalani"); //Tarihe 1 Gün Ek
ler
// '2005-01-01'
$sorgu = mysql_query("SELECT tarihalani -
INTERVAL 1 SECOND; //Tarihten 1 Saniye Çıkarır
// '2004-12-31 23:59:58'
$sorgu = mysql_query("SELECT DATE_ADD(tarihalani,INTERVAL 1 SECOND); //Ta
rihe 1 Saniye Ekler
// '2005-01-01 00:00:00'
$sorgu = mysql_query("SELECT DATE_ADD(tarihalani,INTERVAL 1 DAY)"); //Tarih
e 1 Gün Ekler

```

```

        // '2005-01-01 23:59:59'
        $sorgu = mysql_query("SELECT DATE_ADD(tarihalani,INTERVAL '1:1' MINUTE_SEC
OND)");//Tarihe 1 Dakika 1 Saniye Ekler
        // '2005-01-01 00:01:00'
        $sorgu = mysql_query("SELECT DATE_ADD(tarihalani,INTERVAL '-
1 10' DAY_HOUR)"); //Tarihden 1 Gün 10 Saat Çıkarır
        // '2004-12-30 14:59:59'
        $sorgu = mysql_query("SELECT DATE_ADD('1992-12-
31 23:59:59.000002',INTERVAL '1.999999' SECOND_MICROSECOND)");//Tarihden
1 Saniye 999999 Mikrosaniye Ekler
        // '1993-01-01 00:00:01.000001'
?>

```

DATE_FORMAT

MySQL'de tarihleri dönüştürmek için kullanılır. DATE_FORMAT() fonksiyonu PHP fonksiyonuna benzer şekilde çalışır, ancak farklı biçimlendirme kodları kullanır. MySQL'deki tarih ve saatler ISO 8601 biçimi kullanılarak işlenir. ISO 8601'de tarihlerin, yıl önce olacak şekilde girilmesi gerekir. YYYY-AA-GG SS:DD:SS Örnek 2005'in Haziran ayı için 2005-06-01 olarak girilmelidir. Bu ISO 8601 tarih formatını, GG.AA.YYYY şekline getirmek için bu fonksiyondan yararlanabilirsiniz

Parametreler:

%M : Ay, tüm metin (January...December)
%W : Gün adı, tüm metin. (Sunday...Saturday)
%D : Ayın günü, sayısal, metin soneksiyle birlikte (örneğin, 1st)
%Y : Yıl, sayısal, 4-basamaklı
%y : Yıl, sayısal, 2-basamaklı
%a : Gün adı, 3-karakter (Sun...Sat)
%d : Ayın günü, sayısal, başta sıfır var.. (00...31)
%e : Ayın günü, sayısal, başta sıfır yok...(0...31)
%m : Ay, sayısal, başta sıfır var.. (00...12)
%c : Ay, sayısal, başta sıfır yok... (0...12)
%b : Ay, metin, 3-karakter(Jan...Dec)
%j : Yılın günü, sayısal.
%H : Saat, 24-saat, başta sıfır var
%k : Saat, 24-saat, başta sıfır yok
%h ya da **%I** : Saat, 12-saat, başta sıfır var
%l : Saat, 12-saat, başta sıfır yok.
%i : Dakika, sayısal, başta sıfır var.
%r : Zaman, 12-saat (hh:mm:ss [AM|PM])
%T : Zaman, 24-saat (hh:mm:ss)
%S ya da **%s** : Saniye, sayısal, başta sıfır var... (00...59)
%p : AM veya PM
%w : Haftanın günü, sayısal, 0'dan (Sunday) 6'ya (Saturday) kadar...

Örnek: MySQL Tarih Biçimini Şekillendirme

```
<?php
/*
```

```
Dosyalar veritabanındaki uyeler Tablosunu Kullandığınızı Varsayarsak
```

```

+-----+-----+
| uyeadi | kayittarihi      |
+-----+-----+
| ali   | 2005-05-14 15:20:00 |
+-----+-----+
| veli  | 2005-06-14 15:20:00 |
+-----+-----+
*/
$sorgu = mysql_query("SELECT DATE_FORMAT(kayittarihi, '%d.%m.%Y %H:%i:
%s') AS kayittarihi FROM uyeler",$baglanti); //MySQL Tarih Alanı Biçimlendiriliyor
while($veri = mysql_fetch_array($sorgu)
{
    echo $veri["kayittarihi"];
}

//Sonuç
//14.05.2005 15:20:00
//14.06.2005 15:20:00
?>

```

Örnek: UNIXTIME Alanını Düzenleme

```

<?php
/*
Dosyalar Klasöründeki uyeler Tablosunu Kullandığınızı Varsayarsak...
+-----+-----+
| uyeadi | kayit_tarihi      |
+-----+-----+
| ali   | 1116076800        |
+-----+-----+
*/
require("ayar.php");
require("baglan.php");
$sorgu = mysql_query("SELECT DATE_FORMAT(FROM_UNIXTIME(kayit_tarihi), '%
d.%m.%Y %H:%i:%s') AS kayit_tarihi FROM uyeler"); //Unixtime Zaman Biçimi Dü
zenleniyor
$veri = mysql_fetch_array($sorgu)

    echo $veri["kayit_tarihi"];

//Sonuç
//14.05.2005 15:20:00
?>

```

DATE_SUB

Bu fonksiyon tarih hesaplama işlemleri için kullanılır. Bu fonksiyonla aynı çalışan DATE_ADD() ADDBDATE() SUBDATE() Fonksiyonları vardır. ADDBDATE() ve SUBDATE(), DATE_ADD() ve DATE_SUB() fonksiyonlarının eşanlamlıdır.

Burada kullanılan **tarih**, DATETIME (tarihzaman) ya da DATE (tarih) değeridir. Bunlar başlangıç değerini belirler. Yazım Formatı INTERVAL değerini belirler. Bu değer başlangıç değerine eklenir veya çıkarılır.

Kullanımı:

DATE_SUB(date,INTERVAL expr type)
DATE_SUB(tarih, INTERVAL yazım formatı tip)

Bu fonksiyonla aynı çalışan diğer fonksiyonlar
DATE_ADD (tarih, INTERVAL yazım formatı tip)
ADDDATE(tarih, INTERVAL yazım formatı tip)
SUBDATE(tarih, INTERVAL yazım formatı tip)

DAY

Bu fonksiyon DAYOFMONTH() Fonksiyonunun eşanlamlısıdır.

DAYNAME

Tarih için hafta günlerinin ismini verir. (İngilizce Olarak)

DAYNAME(tarih)

```
mysql> select DAYNAME("2005-08-22");  
-> 'Monday'
```

DAYOFMONTH

Geçerli tarih için ayın gününü verir. (1-31 aralığında).

DAYOFMONTH(tarih)

```
mysql> select DAYOFMONTH('1998-02-03');  
-> 3
```

DAYOFWEEK

DAYOFWEEK(tarih)

Haftanın gününü döndürür.

Geçerli tarih için: (1=Pazar, 2= Pazartesi,7=Cumartesi) Bu değerler ODBC Standardıyla aynıdır.

```
mysql> select DAYOFWEEK('1998-02-03');  
-> 3
```

Kullanımı:

```
mysql> select DAYOFWEEK('1998-02-03');  
-> 3
```

DAYOFYEAR

Geçerli tarih için yılın gün sayısını döndürür. (1-366)

DAYOFYEAR(tarih)

```
mysql> select DAYOFYEAR('1998-02-03');  
-> 34
```

EXTRACT

Bu fonksiyon, aynı DATE_ADD() ve DATE_SUB() fonksiyonlarındaki INTERVAL tiplerini kullanır. Yalnız tarih hesaplamaları yerine, tarihten payları çıkarır.

EXTRACT(typ FROM tarih)

Örnek: EXTRACT() Kullanım Örnekleri

```

<?php
$sql1 = mysql_query("SELECT EXTRACT(YEAR FROM '2005-10-03') AS tarih");
$sonuc = mysql_fetch_object($sql1);
echo $sonuc->tarih; //2005
$sql2 = mysql_query("SELECT EXTRACT(YEAR_MONTH FROM '2005-10-03 15:25:00') AS tarih");
$sonuc = mysql_fetch_object($sql2);
echo $sonuc->tarih; //200510
$sql3 = mysql_query("SELECT EXTRACT(DAY_MINUTE FROM '2005-10-03 15:25:00') AS tarih");
$sonuc = mysql_fetch_object($sql3);
echo $sonuc->tarih; //31525
?>

```

HOUR

Geçerli zamanın saatini döndürür. (0–23 arasında).

HOUR(zaman)

```
mysql> select HOUR('10:05:03');
-> 10
```

MINUTE

Geçerli zaman için dakikayı verir. (0-59);

MINUTE(zaman)

```
mysql> select MINUTE('98-02-03 10:05:03');
-> 5
```

MONTH

Datetime formatındaki tarihin ayını 1-12 şeklinde döndürür.

Kullanımı:

MONTH(tarih)

```
mysql> SELECT MONTH('1998-02-03');
-> 2
```

Örnek: MONTH () Kullanımı

```

<?php
/*
MySQL'deki tarih alanı datetime olarak ayarlandığını varsayarsak.
+-----+
| tarihalani      |
+-----+
| 2005-06-14 15:20:00 |
+-----+
*/
$sorgu = mysql_query("SELECT MONTH(tarihalani) AS tarih_alani FROM tabloadi");
;
$sonuc = mysql_fetch_array($sorgu);
echo $sonuc["tarih_alani"];
//Çıktısı 6 Olacaktır
?>

```

MONTHNAME

Tarih için ayların ismini verir. (İngilizce olarak)

MONTHNAME(tarih)

```
mysql> select MONTHNAME("1998-02-05");  
-> 'February'
```

NOW

Şimdiki zamanı 'YYYY-MM-DD HH:MM:SS' veya YYYYMMDDHHMMSS formatında verir.
NOW()

Örnek: NOW() Kullanım Örnekleri

```
mysql> SELECT NOW();  
-> '1997-12-15 23:50:26'  
mysql> SELECT NOW() + 0;  
-> 19971215235026
```

QUARTER

Geçerli tarih için yılın kaçınıcı çeyreği olduğunu (1-4) döndürür.

QUARTER(tarih)

```
mysql> select QUARTER('98-04-01');  
-> 2
```

SECOND

Zaman için saniyeyi verir (0-59).

SECOND(zaman)

```
mysql> select SECOND('10:05:03');  
-> 3
```

TIME_TO_SEC

Fonksiyona verilen Saat argümanını saniyeye çevirir.

TIME_TO_SEC(saat)

```
mysql> SELECT TIME_TO_SEC('22:23:00');  
-> 80580  
mysql> SELECT TIME_TO_SEC('00:39:38');  
-> 2378
```

WEEK

Fonksiyon tek argümanla geçerli tarihin kaçınıcı hafta olduğunu verir...(0-53 aralığında). (Bazı yerlerde 53 haftanın başlangıcı da olabilir) Bazı yerlerde, Pazar haftanın ilk günüdür. İkinci argümanla WEEK() Fonksiyonunda, haftanın pazar veya pazartesi ile başlayacağını belirleyebilirsiniz. İkinci argüman sıfır (0) olursa hafta pazar günleri başlar, 1 olursa pazartesi günleri başlar.

WEEK(tarih), WEEK(tarih,ilkgün)

```
mysql> select WEEK('1998-02-20');  
-> 7  
mysql> select WEEK('1998-02-20',0);
```

```
-> 7
mysql> select WEEK('1998-02-20',1);
-> 8
mysql> select WEEK('1998-12-31',1);
-> 53
```

WEEKDAY

Geçerli tarih için, haftanın gününü verir. (0 = Pazartesi, 1=Salı,6=Pazar)

WEEKDAY(tarih)

```
mysql> select WEEKDAY('1997-10-04 22:23:00');
-> 5
mysql> select WEEKDAY('1997-11-05');
-> 2
```

YEAR

Geçerli tarihin yılını döndürür. (1000-9999)

```
YEAR(tarih) mysql> select YEAR('98-02-03'); -> 1998
```

YEARWEEK

Geçerli tarihin yıl ve hatfasını döndürür. İkinci argüman, WEEK() fonksiyonundaki ikinci argüman ile aynı çalışır. (ikinci argüman sıfır (0) olursa hafta pazar günleri başlar, 1 olursa pazartesi günleri başlar) Dikkat edilmesi gereken, senenin ilk ve son haftasındaki yıl, argümanda verilen tarihteki seneden farklı olabilir.

YEARWEEK(tarih), YEARWEEK(tarih,ilk)

```
mysql> select YEARWEEK('1987-01-01'); -> 198653
```

1.9.3 Karşılaştırma Fonksiyon ve Operatörleri

Eşittir “=”

```
mysql> SELECT 1 = 0;
-> 0
mysql> SELECT '0' = 0;
-> 1
mysql> SELECT '0.0' = 0;
-> 0
mysql> SELECT '0.01' = 0;
-> 0
mysql> SELECT '.01' = 0.1 ;
-> 1
```

Eşit Değildir “<>”, “!=”

```
mysql> SELECT '.01' <> '0.01';
-> 1
mysql> SELECT .01 <> '0.01';
-> 0
mysql> SELECT 'zapp' <> 'zappp';
-> 1
```

Küçüktür veya Eşittir “<=”

```
mysql>SELECT 0.1 <= 2;  
-> 1
```

Küçüktür “<”

```
mysql> SELECT 2 < 2;  
-> 1
```

Büyüktür veya eşittir “>=”

```
mysql> SELECT 2 >= 2;  
-> 1
```

Büyüktür “>”

```
mysql>SELECT 2 > 2;  
-> 1
```

GREATEST(değer1,değer2,.....)

İki veya daha fazla değer verildiğinde en büyük olan değeri seçer. LEAST() komutuyla aynı şekilde kullanılır.

```
mysql>SELECT GREATEST (2 , 0);  
->2  
mysql>SELECT GREATEST (34.0 , 3.0 , 5.0 , 767.0);  
->767.0
```

MySQL 5.0.13 ‘den önce, GREATEST() fonksiyonu sadece diğer değerlerin hepsi geçersiz duruma geçtiğinde geçersiz oluyordu. 5.0.13’te değerlerden herhangi biri geçersiz olduğunda, hemen geçersiz oluyor.

IN

expr **IN** (değer,...)

Verilen expr değerlerin herhangi birine eşitse “1” değilse “0” verir.

```
mysql> SELECT 2 IN (0, 3, 5, 6);  
-> 0  
mysql> SELECT 'wefwf' IN ('weee', 'wefwf', 'whfk');  
-> 1
```

INTERVAL (N,N1,N2,N3...)

Eğer N<N1 ise “0”; eğer N<N2 ise “+1”; eğer N NULL ise “-1“ değerini alır ve alınan değerler toplanır. Fakat bu fonksiyonun doğru olarak çalışabilmesi için N1<N2<N3.....<Nn olmalıdır.

```
mysql> SELECT INTERVAL ( 23, 1, 15, 17, 30, 44, 200 );  
-> 3  
mysql> SELECT INTERVAL ( 10, 1, 10, 100, 1000 );  
-> 2  
mysql> SELECT INTERVAL ( 22, 23, 30, 44, 200 );  
-> 0
```

LEAST (*değer1, değer2*)

İki yada daha fazla değerlikten en küçük olanını seçer. Eğer girilen değerler *string* ise harf sırasına göre bulacaktır.

```
mysql> SELECT LEAST ( 2, 0 );
-> 0
mysql> SELECT LEAST ( 3.0, 34.0, 5.0, 767.0 );
-> 3.0
mysql> SELECT LEAST ( 'B', 'A', 'C' )
-> 'A'
```

1.9.4 Mantıksal Operatörler

SQL'de mantıksal operatörler **TRUE**, **FALSE** VE **NULL** terimlerinden oluşur. MySQL'de ise 1(**TRUE**), 0(**FALSE**) ve **NULL** terimleri kullanılır.

NOT (DEĞİL) “!”

Lojik **DEĞİL KAPISI** 'na eşittir. 1 girilirse çıkışı 0; 0 girilirse, çıkışı 1 yapar.

```
mysql> SELECT NOT 10 ;
-> 0
mysql> SELECT NOT 0 ;
-> 1
mysql> SELECT NOT NULL;
-> NULL
mysql> SELECT NOT ! (1+1);
-> 0
mysql> SELECT NOT ! 1 + 1;
-> 1
```

En son verilen örnekte sonuç '1' dir çünkü “(! 1) + 1” değerine eşittir.

AND (VE) “&&”

Lojik **VE KAPISI**'na eşittir. Girişlerden herhangi biri '0' ise çıkış değerini '0' verir.

```
mysql> SELECT 1 && 1 ;
-> 1
mysql> SELECT 1 && 0 ;
-> 0
mysql> SELECT 1 && NULL ;
-> NULL
mysql> SELECT 0 && NULL ;
-> 0
mysql> SELECT NULL && 0 ;
-> 0
```

OR (VEYA) “|”

Lojik **VEYA KAPISI**'na eşittir. Girişlerden herhangi birisi '1' olduğunda çıkış değerini '1' verir.

```
mysql> SELECT 1 || 1 ;
-> 1
mysql> SELECT 1 || 0 ;
-> 1
mysql> SELECT 0 || 0 ;
-> 0
mysql> SELECT 0 || NULL ;
-> NULL
mysql> SELECT 1 || NULL ;
-> 1
```

XOR (ÖZEL VEYA)

Lojik olarak **ÖZEL VEYA KAPISI**'nin yaptığı işi görür. Girişler aynı olduğunda çıkış değerini '0'; diğer durumlarda '1' verir.

```
mysql> SELECT 1 XOR 1 ;
-> 0
mysql> SELECT 1 XOR 0 ;
-> 1
mysql> SELECT 1 XOR NULL ;
-> NULL
mysql> SELECT 1 XOR 1 XOR 1 ;
-> 1
```

Matematiksel olarak “(a AND (NOT b)) OR ((NOT a) AND b) ” değerine eşittir

1.9.5. Kontrol Mekanizmaları ve Karakter Fonksiyonları

1. **CASE** *değer* **WHEN** [*değeri karşılaştır*] **THEN** *sonuç* [**WHEN** [*değeri karşılaştır*] **THEN** *sonuç* ...] **[ELSE** *sonuç*] **END**

2. **CASE WHEN** [*durum*] **THEN** *sonuç* [**when** [*durum*] **THEN** *sonuç* ...] **[ELSE** *sonuç*]

Birincisinde eğer değer karşılaştırılacak değere eşitse sonuçta söylenen durumu yapar; eğer değer karşılaştırılacak durumlara eşit değilse ELSE komutu ile söylenen sonucu yaptırır. İkincisinde ise, eğer durum gerçekleşiyorsa sonuçta söyleneni yapar; eğer durum gerçekleşmiyorsa yine ELSE komutundan sonra söylenen işi yapar.

```
mysql> SELECT CASE 1 WHEN 1 THEN 'one'
-> WHEN 2 THEN 'two' ELSE 'more' END;
-> 'one'
mysql> SELECT CASE 1>0 THEN 'true' ELSE 'false' END;
-> 'true'
mysql> SELECT CASE BINARY 'B'
-> WHEN 'a' THEN 1 WHEN 'b' THEN 2 END;
-> NULL
```

IF(*expr1*, *expr2*, *expr3*)

expr1 'de yazan eğer doğruysa *expr2*; değilse *expr3* 'de yazanı uygular.

```
Mysql> SELECT IF ( 1 > 2 , 1, 0);
-> 0
mysql> SELECT IF ( 1 < 2, 'YES', 'NO' );
-> 'YES'
mysql> SELECT IF (STRCMP('test', 'test1'), 'no', 'yes' );
-> 'no'
```

ASCII (str)

Eğer girilen değer null bir ifade değilse değerini asclı kodunu verir(0 – 255)

BIN(N)

Girilen sayının BINARY sayı sistemine çevirir.

```
mysql> SELECT BIN ( 12 );  
-> '1100'
```

BIT_LENGTH (str)

Girilen değer kaç bitlik olduğunu bulur.

```
mysql> SELECT BIT_LENGTH ('text')  
-> 32
```

CHAR (N,...)

Girilen asclı kodları karakterlere dönüştürür.

```
mysql> SELECT CHAR ( 77, 121, 83, 81, '76' );  
-> 'MySQL'  
mysql> SELECT CHAR ( 77, 77.3, '77.3' );  
-> 'MMM'
```

CONCAT(str1, str2,...)

String toplama yapar. Eğer değerlerden herhangi birisi null bir ifade ise null olan ifadeyi seçer.

```
mysql> SELECT CONCAT ( 'My', 'S', 'QL' );  
-> 'MySQL'  
mysql> SELECT CONCAT ( 'My', NULL, 'QL' );  
-> NULL  
mysql> SELECT CONCAT ( 14.3 );  
-> 14.3
```

CONCAT_WS (separator, str1, str2,...)

Tıpkı CONCAT fonksiyonunda olduğu gibi string toplama yapar fakat bunun tek farkı değerler arasına istediğimiz bir karakter koyabilmemizdir. Eğer değerlerin arasında bir null değer varsa onu pas geçer ve bir sonrakiyle birleştirir.

```
mysql> SELECT CONCAT_WS ( ',', 'First name', 'Second name', 'Last name' );  
-> 'First name, Second name, Last name'  
mysql> SELECT CONCAT_WS ( ',', 'First name', NULL, 'Last name' );  
-> 'First name, Last name'
```

ELT (N, str1, str2, str_m...)

N=1 ise str1'i, N=2 ise str2'i, N=n ise str_n'i seçer.

```
mysql> SELECT ELT ( 1, 'ej', 'der', 'alp' );  
-> 'ej'  
mysql> SELECT ELT ( 4, 'ej', 'der', 'alp' );  
-> 'alp'
```

FIELD (str, str1, str2, str3,...)

Eğer str'deki yazı diğerlerinin içinde varsa kaç numaralı str' de olduğunu verir. Eğer str diğerlerinin içinde yoksa '0' verir.


```
mysql> SELECT FIELD ( 'ej', 'hej', 'ej', 'foo');
-> 2
mysql> SELECT FIELD ( 'ej', 'hej', 'lej', 'foo');
-> 0
```

FIND_IN_SET (str, strlist)

Verilen str'nin strlist içinde kaçınıcı sırada olduğunu bulur.

```
mysql> SELECT FIND_IN_TEXT ( 'b', 'a, b, c, d');
-> 2
```

FORMAT (X, D)

'#,###,###.##' diye bir X sayısının olduğunu varsayalım. D sayısı bu sayının ondalıklı kısmını yuvarlar veya '0' koyarak ondalıklı kısımdaki D tane sayı olmasını sağlar. Eğer D sıfır ise ondalıklı kısım kalkar.

```
Mysql> SELECT FORMAT ( 12332.123456, 4 );
-> 12,332.1235
mysql> SELECT FORMAT ( 12332.1, 4);
-> 12,332.1000
mysql> SELECT FORMAT ( 12332.2, 0);
-> 12,332
```

INSERT(str,pos,len,newstr)

Verilen metin içerisine, istenilen pozisyondan itibaren ' ' işareti içinde verilen değeri yazar.

```
mysql> SELECT INSERT('Quadratic', 3, 4, 'What');
-> 'QuWhattic'
mysql> SELECT INSERT('Quadratic', -1, 4, 'What');
-> 'Quadratic'
mysql> SELECT INSERT('Quadratic', 3, 100, 'What');
-> 'QuWhat'
```

LEFT(str,len)

Parantez içerisinde verilen metnin, verilen karakter sayısı kadarını soldan itibaren alır.

```
mysql> SELECT LEFT('foobarbar', 5);
-> 'fooba'
```

LENGTH(str)

' ' (tırnak) içinde verilen string ifadenin sayısını verir.

```
mysql> SELECT LENGTH('text');
-> 4
```

LOWER(str)

Büyük harflerle yazılan string ifadeyi küçük harflere çevirir.

```
mysql> SELECT LOWER('QUADRATICALLY');
-> 'quadratically'
```

LTRIM(str)

Yazılan string ifadenin solunda bulunan boşluğu siler.

```
mysql> SELECT LTRIM(' barbar');
-> 'barbar'
```

RTRIM(str)

Yazılan string ifadenin sağında bulunan boşluğu siler.

```
mysql> SELECT LTRIM('barbar ');
```

```
-> 'barbar'
```

MID(str,pos,len)

Verilen string soldan belirtilen pozisyonundan itibaren, belirtilen kadar karakterini alır.

REPEAT(str,count)

Tırnak içinde verilen karakteri istendiği kadar yazdırır.

```
mysql> SELECT REPEAT('MySQL', 3);  
-> 'MySQLMySQLMySQL'
```

REVERSE(str)

Verilen string ifadeyi tersinden yazdırır.

```
mysql> SELECT REVERSE('abc');  
-> 'cba'
```

RIGHT(str,len)

Verilen string içinden sağdan verilen değer kadar olan karakteri alır

```
mysql> SELECT RIGHT('foobarbar', 4);  
-> 'rbar'
```

SPACE(N)

Verilen sayısal değer kadar boşluk verir.

```
mysql> SELECT SPACE(6);  
-> '      '
```

UPPER(str)

Verilen string ifadeyi tamamen büyük karakterlere çevirir.

```
mysql> SELECT UPPER('Hej');  
  
-> 'HEJ'
```

1.9.6. Aritmetik Operatörler

Toplama

```
mysql> SELECT 3+5;  
-> 8
```

Çıkarma

```
mysql> SELECT 3-5;  
-> -2
```

Negatif sayı (devamındaki sayıyı negatif yapar)

```
mysql> SELECT - 2;  
-> -2
```

Çarpma

```
mysql> SELECT 3*5;  
-> 15
```

```
mysql> SELECT 18014398509481984*18014398509481984.0;  
-> 324518553658426726783156020576256.0
```

Bölme

```
mysql> SELECT 3/5;  
-> 0.60
```

Sıfıra bölme hatası olduğunda sonuçta **NULL** değeri döner.

```
mysql> SELECT 102/(1-1);
```

-> NULL

DIV

Tam bölme; Bölme işlemi sonunda virgöl kısmı atılır, tam kısmı yazar.

```
mysql> SELECT 5 DIV 2;
```

-> 2

N % M

Kalan Bulma; N bölü M. işlemiden kalanı verir.

1.9.7. Matematiksel Fonksiyonlar

ABS(X)

Verilen sayısal değerin mutlak değerini verir.

```
mysql> SELECT ABS(2);
```

-> 2

```
mysql> SELECT ABS(-32);
```

-> 32

ACOS(X)

Verilen değerin arc cosine değerini verir.

```
mysql> SELECT ACOS(1);
```

-> 0

```
mysql> SELECT ACOS(1.0001);
```

-> NULL

```
mysql> SELECT ACOS(0);
```

-> 1.5707963267949

ASIN(X)

Verilen değerin arc sinus değerini verir

```
mysql> SELECT ASIN(0.2);
```

-> 0.20135792079033

```
mysql> SELECT ASIN('foo');
```

+-----+

| ASIN('foo') |

+-----+

| 0 |

+-----+

1 row in set, 1 warning (0.00 sec)

```
mysql> SHOW WARNINGS;
```

+-----+-----+-----+-----+-----+-----+

| Level | Code | Message |

+-----+-----+-----+-----+-----+-----+

| Warning | 1292 | Truncated incorrect DOUBLE value: 'foo' |

+-----+-----+-----+-----+-----+-----+

ATAN(X)

Verilen değerin arc tangent değerini verir.

```
mysql> SELECT ATAN(2);
```

-> 1.1071487177941

```
mysql> SELECT ATAN(-2);
```

-> -1.1071487177941

COS(X)

Radians değeri verilen değerin cosinusini verir.

```
mysql> SELECT COS(PI());  
-> -1
```

COT(X)

Verilen değerin cotangant değerini verir.

```
mysql> SELECT COT(12);  
-> -1.5726734063977  
mysql> SELECT COT(0);  
-> NULL
```

DEGREES(X)

Verilen değişkenini değerini radyandan dereceye dönüştürür.

```
mysql> SELECT DEGREES(PI());  
-> 180  
mysql> SELECT DEGREES(PI() / 2);  
-> 90
```

LN(X) LOG(X), LOG(B,X)

X değişken değerinin doğal logaritmasını verir.

```
mysql> SELECT LN(2);  
-> 0.69314718055995  
mysql> SELECT LN(-2);  
-> NULL  
mysql> SELECT LOG(2);  
-> 0.69314718055995  
mysql> SELECT LOG(-2);  
-> NULL
```

MOD(N,M), N % M, N MOD M

N değerinin *M*. değerine bölümünden kalanını verir.

```
mysql> SELECT MOD(234, 10);  
-> 4  
mysql> SELECT 253 % 7;  
-> 1  
mysql> SELECT MOD(29,9);  
-> 2  
mysql> SELECT 29 MOD 9;  
-> 2
```

PI (X)

PI sayısının değerini verir.

```
mysql> SELECT PI();  
-> 3.141593  
mysql> SELECT PI()+0.00000000000000000000;  
-> 3.141592653589793116
```

RADIANS(X)

X değişkenin değeri dereceden radyana dönüştürür.

```
mysql> SELECT RADIANS(90);  
-> 1.5707963267949
```

TAN(X)

Radyan olarak verilen değerin Tanjantının verir.

```
mysql> SELECT TAN(PI());  
-> -1.2246063538224e-16  
mysql> SELECT TAN(PI()+1);  
-> 1.5574077246549
```

SQRT(X)

X değerini karekökünü verir.(negatif sayıların karekökü olmaz)

```
mysql> SELECT SQRT(4);  
-> 2  
mysql> SELECT SQRT(20);  
-> 4.4721359549996  
mysql> SELECT SQRT(-16);  
-> NULL
```

SIGN(X)

Değişkenlerin işaretlerini verir. (-1 → sayı negatifse , 0 → sıfır olduğunda, veya 1 → sayı pozitif olduğunda)

```
mysql> SELECT SIGN(-32);  
-> -1  
mysql> SELECT SIGN(0);  
-> 0  
mysql> SELECT SIGN(234);  
-> 1
```

SIN(X)

Radyan olarak verilen değerin sinüsünü verir.

```
mysql> SELECT SIN(PI());  
-> 1.2246063538224e-16  
mysql> SELECT ROUND(SIN(PI()));  
-> 0
```

ÖLÇME VE DEĞERLENDİRME

Öğrenme faaliyetinde kazandığınız bilgileri ölçebileceğiniz kısma geldiniz. Bu bölümde yer alan ölçme sorularını dikkatlice okuyarak cevaplandırınız.

A. ÖLÇME SORULARI

Aşağıda çoktan seçmeli sorular bulunmaktadır. Soruları okuyarak doğru bulduğunuz seçeneği işaretleyiniz. Test soruları bitirince Cevap Anahtarı ile karşılaştırınız.

- 1- Aşağıdakilerden hangisi bir veritabanı programıdır?
A) Photoshop B) Dreamweaver C) MySQL D) Apache
- 2- MySQL’de bir veritabanı hangi komutla oluşturulur?
A) Select Database
B) Create Database
C) Create Table
D) Create Index
- 3- Oluşturulan bir veritabanı hangi komutla silinir?
A) Drop B) Delete C) Del D) Hiçbiri
- 4- Aşağıdaki MySQL komut uygulaması sonucunda hangisi hata verir?
A) mysql>select sin(45);
B) mysql>create table müşteri;
C) mysql>quit
D) mysql>use ahmer
- 5- Bir MySQL veritabanı üzerinde işlem yapabilmek için, o veritabanını seçmek gerekir. Böyle bir veritabanını seçmek için hangi komut kullanılır?
A) use B) select C) create D) drop
- 6- Aşağıdakilerden veri türlerinden hangisi metin (karakter) veri türüdür?
A) Datetime B) Real C) Int D) VarChar
- 7- MySQL’de index işlemi ne amaçla yapılır?
A) Veritabanında veri aramak için,
B) Veritabanından veri silmek için,
C) Veritabanındaki verileri sıralamak için,
D) Veritabanına veri eklemek için,
- 8- Bir veritabanından veri almak (sorgulamak) için hangi komut kullanılır?
A) Select B) Update C) Insert D) Alter
- 9- mysql>select from okul where tur='meslek' ;
MySQL komut satırı ne iş yapar.
A) Okul veritabanında, tur değeri “meslek” olanları listeler,
B) Okul veritabanında, tur değeri “meslek” olanları siler,
C) Okul veritabanında, tur değeri “meslek” olanları listeler,
D) Okul veritabanında, tur değeri “meslek” olanları tablo oluşturur,

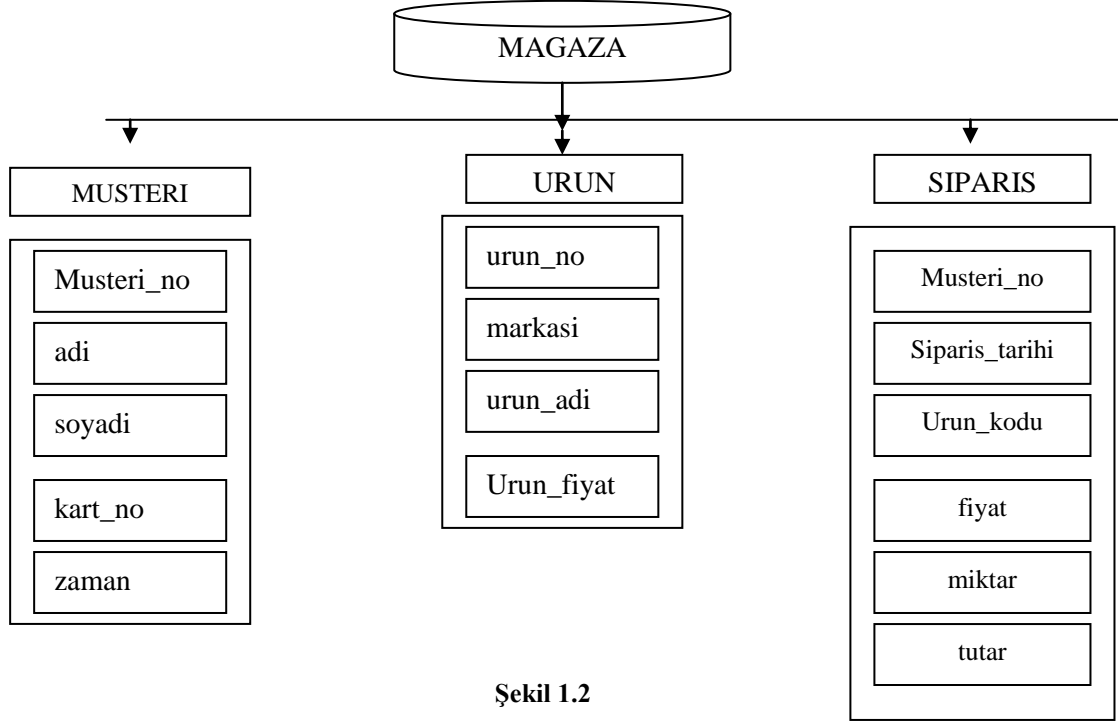
10- Veritabanında bir alanın deęerini deęiřtirmek iin hangi MySQL komutu kullanılır?
A) Update B) Alter C) Insert D) Hepsi

11-Adının iinde ‘M’ geen ğrencilerin ad ve numarasının grntlenmesi iin gerekli olan ifadeyi yazınız.

DEęERLENDİRME

Cevaplarınızı cevap anahtarı ile karřılařtırınız. Bu test sizin kendi kendinizi denemeniz iin yapılmıřtır. Test iinde cevaplandıramadıęınız, yanlıř cevaplandırdıęınız veya kendinizi bilgi bakımından eksik hissettięiniz sorular iin bilgi sayalarına tekrar dnp ęrenme faaliyetini gzden geirmeniz tavsiye olunur.

UYGULAMA FAALİYETİ-1



Şekil 1.2

İŞLEM BASAMAKLARI	ÖNERİLER
➤ Yukarıdaki şekle göre MAGAZA adlı bir veri tabanı oluşturunuz.	
➤ muster_i, urun ve siparis adlı tabloları oluşturunuz.	
➤ Her tabloya 5(beş) 'er kayıt giriniz.	Uygulamalar için kayıtlarda benzer değerler girebilirsiniz. (aynı isimli, fiyatları aynı olan ürünler gibi)
➤ Girilen kayıtlardan urun fiyatı 25YTL olan ürünleri listeleyin.	
➤ müşteri adı Mesut ve sipariş tarihi 30/04/2007 olan kayıtları görüntülesin.	
➤ Girmiş olduğunuz müşteri kayıtlarından ikisini siliniz.	
➤ Ürünlerden bazılarının fiyatlarını değiştiriniz.	
➤ Ürün fiyatı en pahalı olan ürün hangisi?	
➤ Tabloları silin.	
➤ Mağaza veritabanını silin.	

ÖĞRENME FAALİYETİ – 2

AMAÇ

PHP ile yapılan önceki çalışmalarda, verileri depolamak ve elde etmek için düz dosyalar kullanılmıştı. Düz dosya dediğimiz basit bir metin (text) dosyasıdır. Bu bölümde ilişkisel veritabanı sistemlerinin depolama ve elde etme işlemlerini bir web uygulamasında çok daha kolay, güvenli ve kullanışlı hale geldiğini göreceksiniz. Bir veritabanı oluşturmak için MySQL kullanıldıktan sonra bu veritabanını web tabanlı bir ön uca taşıyabiliriz.

Bu bölümde, PHP kullanılarak bir MySQL veritabanına nasıl erişileceğini, ayrıca veritabanından okuyup veritabanına yazmayı öğreneceksiniz. Bu kısımda bir veritabanına bir kayıttın nasıl ekleneceği, sorgulama yapılacağı, silineceği düzeltileceği anlatılacaktır.

ARAŞTIRMA

- PHP ve MySQL’le hazırlanmış web sitelerini inceleyiniz.
- İnternet’te veri gönderme hakkında bilgi toplayınız.
- İnternet’te, kullanıcılardan gelen bilgiler nereye, nasıl kaydediliyor olabilir?

2. PHP İLE MySQL VERİTABANINA ERİŞMEK

2.1. PHP İle Veritabanı Etkileşimleri

PHP için şimdiye kadar anlatılan konular, bundan sonra anlatılacak işlere yardımcı olacak komutları içermektedir. Bu kısımda anlatılacak konular HTML kısmında anlatılan formlarla da ilgili olduğundan öncelikle o kısmı tekrar gözden geçirilmelidir.

İnternet üzerinde birçok sayfada form üzerinden bilgi girişi yapılmaktadır. Örneğin: Bir mail okuma sayfasından kullanıcı adı ve şifre girilen sayfa, bir arama motorunda bir konuyu ararken kullanılan sayfalar formlara örnek verilebilir. Bu tipte sayfalar genelde bir veritabanından bilgi sorgularlar ve sorgulamanın sonucunda yeni sonuçlar döndürürler.

PHP bir script dili olduğu için sürükle bırak mantığıyla çalışan nesnelere yoktur. Bu yüzden HTML nesnelere kullanılır.

HTML formlarında metin kutusu, onay kutusu, buton gibi nesnelere vardır. Bu nesnelere form nesnelere olarak kullanılacak ve gerekli bilgi giriş işlemleri yapılmaktadır.

HTML formlar için iki yöntem olduğunu bir önceki modülde görmüştünüz.

GET: Client (istemci) tarafından gönderilen bilgiler web browserın adres satırında görüntülenir.

Örnek;

```
<html>
  <FORM ACTION="formyolla.php" METHOD="GET">
    Adınız: <input type="text" name =ad" ></br>
    Soyadınız: < input type="submit" value="Gönder">
  </FORM>
</html>
```

Bu örnek getform.html adıyla kaydedilmeli ve web browser'ın adres satırında <http://localhost/webmagaza/getform.html> yazılarak çağırılmalıdır. İlgili alanları doldurduktan sonra Gönder butonunu basılır. Adre satırında formda girilen bilgiler karışık bir şekilde görülmektedir.

...../formyolla.php?ad="halil"&soyad="halepli"&.....

Burada <form> tagının action parametresiyle çağrılan php dosyasının adı,input type ile tanımlanan nesnelere adlarını ve browserden girilen değerler görülmektedir. Action parametresiyle gönderilen dosya adından sonra "?" daha sonra ilk nesnenin "name"parametresiyle verilmiş adı, değeri ve sonraki nesne adlarının ve değerlerini birleştirmek için kullanılan"&" işaretleri görülür.

POST : Client (istemci) tarafından gönderilen bilgiler browserın adres satırında görüntülenmeden yollanır.

Örnek;

```
<html>
  <FORM ACTION= formyolla.php" METHOD="POST">
    Adınız :<input type= "text" name="ad"></br>
    Soyadınız : <input type= "text" name="soyad"></br>
    <input type="Submit" value="Gönder">
  </FORM>
</html>
```

Bu örnek postform.html adıyla kaydedilmeli ve web browser'ın adres satırında <http://localhost/webmagaza/postform.html> yazılarak çağırılmalıdır.

İlgili alanları doldurduktan sonra Gönder butonuna basılır. Bu örnekte sadece çağrılan sayfanın adresi görünmektedir.(Değişkenler ve girilen değerler görünmez)

<http://localhost/webmagaza/formyolla.php> şeklinde adres satırı görünür.

Yukarıdaki örnekler incelendiğinde FORM tagının ACTION parametresinin çağrılacak yeni PHP dosyasının adı yer almaktadır.

Veritabanı üzerinde işlem yaparken PHP'nin MySQL için gerekli olan fonksiyonlar kullanılacaktır. Bu fonksiyonlardan en gereklileri devam eden konularda anlatılacaktır.

PHP ile yazılacak kodlarda bir veritabanı işlemi yapılıyorsa, zorunlu olarak sırasıyla şunlar yapılmalıdır.

- 1.MySQL bağlantısının gerçekleştirilmelidir,
- 2.MySQL sunucusundaki kullanılacak veritabanı seçilmelidir,
- 3.Bir SQL (Structure Query Language) ifadesi yazılmalıdır,
- 4.Bu SQL ifadesi çalıştırılmalıdır,

5. SQL işleminin sonucuna göre işlemler tamamlanmalıdır,
6. Bağlantı kapatılmalıdır.

Buradaki altı adımın her birinde kullanılan komutlar aşağıda örneklerle açıklanmıştır.

2.2. MySQL Sunucusuna Bağlantı

PHP 5'te MySQL'e bağlanmak için yeni bir kütüphane bulunur. Bu kütüphane `mysqli` adını taşır. (*i* harfi, improved, yani geliştirilmiş anlamına gelir.). `mysqli` kütüphanesi nesne yönelimli ya da prosedürel söz dizimi kullanmanıza imkân verir.

Kullanımı;

```
$db=mysqli_connect(<Sunucusu_adi>,<Kullanıcı_adi>,<Kullanıcı_sifre>);
```

Bağlantı parametreleri yazılarak bu ifade bir değişkene aktarılır.

Örnek;

```
$baglan=mysqli_connect ("localhost" , "root" , "manisa");
```

2.3 Veritabanı Seçimi

MySQL'i bir komut satırı arabiriminden kullanırken şöyle bir komutla hangi veritabanını kullanmayı planladığınızı söylemeniz gerekir.

```
mysql>use kitaplar;
```

Webden bağlanırken de bu yapmamız gerekir. Kullanılacak olan veritabanı, **mysqli** yapılandırıcısından ya da **mysqli_connect ()** fonksiyonunun bir parametresi olarak belirtilir. Varsayılan veritabanını değiştirmek istiyorsanız, bunu **mysqli_select_db ()** fonksiyonu ile yapabilirsiniz.

Kullanımı;

```
mysqli_select_db(<veritabanı_adi>.<Bağlantı_adi>);
```

Örnek;

```
$sec = mysqli_select_db("magaza",$baglan);  
If ($sec)  
{  
    print "Veritabanı Sizin İçin Seçildi";  
else  
{  
    print "Veritabanı Seçilemedi";
```

2.4 Veritabanını Sorgulamak

Sorguyu çalıştırmak için **mysql_query ()** fonksiyonunu kullanabilirsiniz. Oluşturulan SQL ifadesinin hangi bağlantı için çalıştırılacağı belirtilerek bir değişkene atanır.

```
degisken=mysql_query(<sql_ifadesi>,<baglati_adi>);
```

Örnek;

```
$sorgu=mysql_query($sql,$baglan);
```

Örnek;

```
$query=("select * from kitap where yazar='Ali'", $baglanti);
```

2.5 SQL Sunucu Üzerinde İşlemler

Sonuç tanımlayıcısından sonuçların farklı şekillerde elde edilmesini sağlayan pek çok farklı fonksiyon vardır. Sonuç nesnesi ya da tanımlayıcısı, sorgu tarafından döndürülür, en satırlara erişim için anahtar vazifesi görür. Burada dönen sonuçlar MySQL sunucu yönetiminden alınan SQL sonuçları gibi ekrana yazdırılmazlar. PHP komutlar ile bunlar üzerinde işlemler yapılır. Update, Insert ve Delete işlemleri geriye sonuç döndürmezler. Sadece Select ifadelerinde geriye sonuçlar döner. Bu sonuçlar üzerinde işlem yapmak için aşağıdaki iki fonksiyondan yararlanılır.

```
mysql_num_field(<calistirilan_sorgu_adi>)
```

Bu komut SQL ifadesinde seçilen bilgi alanlarının sayısını bulmak için kullanılır.

```
mysql_fetch_row(<calistirilan_sorgu_adi>)
```

Bu komut SQL ifadesi çalıştırıldıktan sonra dönen araçlar içerisinde bir tanesini çağırmak için kullanılır. Bu komut ikinci kez çalıştırılır ise bir sonraki kayıt çağrılır. Sonuç, alan sayısı kadar elemanlı bir dizi oluşturularak atılır.

Örnek;

```
<?
$baglan=mysql_connect ("localhost","root","");
$sec = mysql_select_db("magazaé,$baglan);
If ($sec)
{
    print "Veritabanı Seçildi";
    $sql="select musteri_no,adi,soyadi,dogum_tarihi from musteri";
    $sorgu=mysql_query($sql,$baglan);
    $alan_sayisi = mysql_num_fields($sorgu); //sonuc 4
    $i=0;
    While ($satir = mysql_fetch_row($sorgu) )
    {
        $i++;
        for ($s=1;$s<$alan _sayisi;$s++)
        {
            $dizi [$i] [$s] = $satir[$s] ;
        }
    }

    for($i-1;$i<=count ($dizi);$i++)
    {
```

```

        print $dizi[$i][1].$dizi [$i][3]."<br>";
    }
}

mysql_close($baglan);
}
else
{
print "veritabanı Seçilemedi";
return;
}
?>

```

2.6 MySQL Bağlantısını Kapatma

Bir PHP uygulamasında veritabanı ile yapılacak işlemler tamamlandıktan sonra veritabanı bağlantısı kapatılmalıdır. Çünkü her veritabanına yapılacak bağlantı sayısı sınırlıdır. Bağlantı sayısını aşmamak için işi biten kullanıcının bağlantısı aşağıdaki komut ile kapatılır.

```
mysql_close(<baglanti_adi>);
```

Örnek;

```
mysql_close($baglanti);
```

Örnek Uygulama

Aşağıdaki uygulamada Firmadaki kullanıcı internet üzerinden müşteri bilgilerini girebilecek şekilde hazırlanmış bir web sayfasının kodları yazılmıştır. Uygulama iki kısımdan oluşmaktadır.

1.Kullanıcı arayüzünün HTML olarak hazırlanması (dosya adı muster_i.html)

```

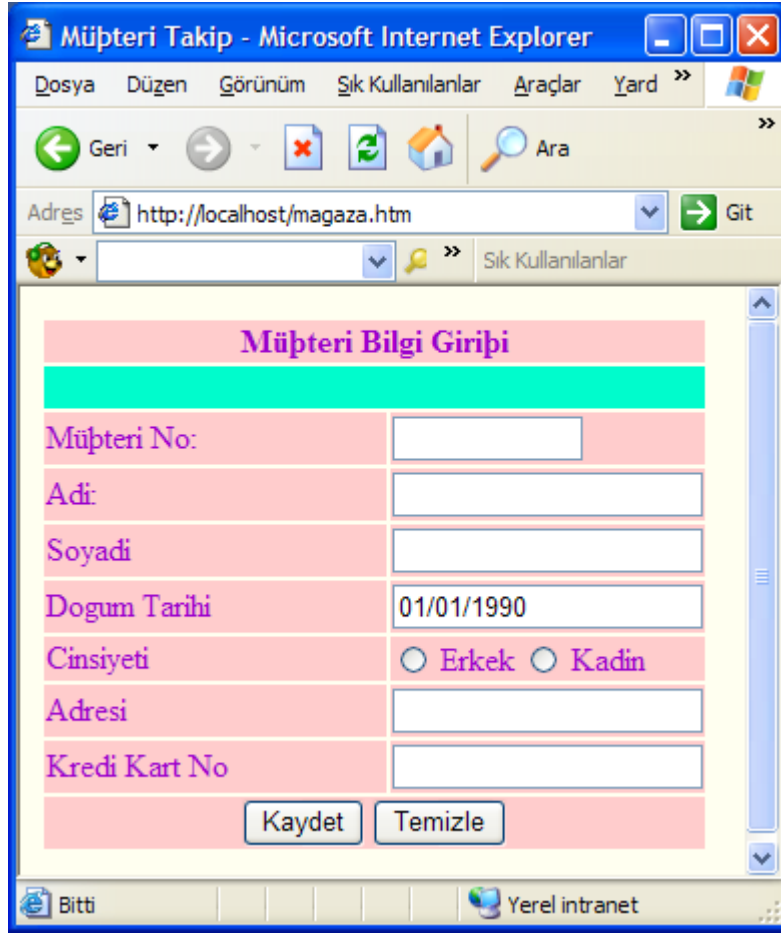
<html>
<head>
<title>e_Mađaza Müþteri Sipariþ Takip otomasyonu</title>
<meta http-equiv="Content-Type" content="text/html; charset=">
</head>
<body bgcolor="#FFFFFF" text="#000000">
<form name=" " method="get" action="muster_i.php">
<table border="0">
<tr bgcolor="#FFCCCC">
<td colspan="6"> <div align="center"><b>Müþteri Bilgi Giriþi</b></div></td>
</tr>
<tr bgcolor="#CCFFFF">
<td colspan="6">&nbsp;  </td>
</tr>
<tr bgcolor="#FFCCCC">
<td width="169">Müþteri No :</td>
<td width="147" colspan="5" bgcolor="#FFCCCC"> <input type="text" name="
xmuster_i_no" maxlength="5" size="10">
</td>
</tr>

```

```

<tr bgcolor="#FFCCCC">
  <td width="169">Adý :</td>
  <td width="147" colspan="5"> <input type="text" name="xadi"> </td>
</tr>
<tr bgcolor="#FFCCCC">
  <td width="169">Soyadý :</td>
  <td width="147" colspan="5"> <input type="text" name="xsoyadi"> </td>
</tr>
<tr bgcolor="#FFCCCC">
  <td width="169">Dođum Tarihi :</td>
  <td width="147" colspan="5"> <input type="text" name="xdogum_tarihi"
value="01/01/1900">
  </td>
</tr>
<tr bgcolor="#FFCCCC">
  <td width="169">Cinsiyeti
  <tdwidth="147" colspan="5"> <input type="radio" name="xcinsiyeti" value="e">
    Erkek
    <input type="radio" name="xcinsiyeti" value="k">
    Kadýn </td>
</tr>
<tr bgcolor="#FFCCCC">
  <td width="169">Adresi :</td>
  <td width="147" colspan="5"> <input name="xadres" type="text" id="xadres"
maxlength="50">
  </td>
</tr>
<tr bgcolor="#FFCCCC">
  <td with="169">Kredi kart no :</td>
  <td width="147" colspan="5"> <input type="text" name="xkartno"
maxlength="16">
  </td>
</tr>
<tr bgcolor="#FFCCCC">
  <td colspan="6"> <div align="center">
    <div align="center">
      <input type="submit" name="xgiris" value="KAYDET">
      <input type="reset" name="xtemizle" value="TEMÝZLE" >
    </div>
  </div></td>
</tr>
</table>
</form>
</body>
</html>

```



Resim 2.1 magaza.htm ekranı

2. Kullanıcı ara yüzünde girilen bilgilerin veri tabanına kaydedilmesi (dosya adı muster_i.php)

```
<?
$baglan = mysql_connect ("localhost", "root", "Ankara");
$sec = mysql_select_db ("magaza", $baglan) :
$xdogum_tarihi=tarihe_çevir($xdogum_tarihi) :
$sql="insert inso müşteri (müşteri_no, adı, soyadı, doğum_tarihi, cinsiyeti, adres,
kart_no) values '$xmüşteri_no','$xadı','$xsoyadı', '$zdoğum_tarihi', $xcinsiyeti', '
$xadres', '$xkart_no')";
$sorgu=mysql_query($sql, $baglan):
if ($sorgu)
    print "kayıt başarıyla girilmiştir" :
else
    print"lütfen girdiğiniz bilgileri kontrol ediniz":
mysql_close ($baglan):
}
?>
```

ÖLÇME VE DEĞERLENDİRME

Öğrenme faaliyetinde kazandığınız bilgileri ölçebileceğiniz kısma geldiniz. Bu bölümde yer alan ölçme sorularını dikkatlice okuyarak cevaplandırınız.

A. ÖLÇME SORULARI

Aşağıda çoktan seçmeli sorular bulunmaktadır. Soruları okuyarak doğru bulduğunuz seçeneği işaretleyiniz. Test soruları bitirince Cevap Anahtarı ile karşılaştırınız.

1. MySQL veritabanına bağlanmak için hangi fonksiyon kullanılır?

- A) mysql_connect
- B) myql_open
- C) mysql_bağlan
- D) mysql_select

2. Veritabanı bağlantısını kapatmak için hangi fonksiyon kullanılır?

- A) mysql_disconnect
- B) mysql_close
- C) mysql_kapat
- D) mysql_drop

3. Mysql_query fonksiyonu kaç tane parametre alır?

- A) 2 B) 3 C) 4 D) 5

4. “use deneme “ ifadesinin anlamı nedir?

- A) deneme bağlantısını kullan
- B) deneme sorgusunu kullan
- C) deneme veritabanını kullan
- D) deneme sunucusunu kullan

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Bu test sizin kendi kendinizi denemeniz için yapılmıştır. Test içinde cevaplandıramadığınız, yanlış cevaplandırduğunuz veya kendinizi bilgi bakımından eksik hissettiğiniz sorular için bilgi sayalarına tekrar dönüp öğrenme faaliyetini gözden geçirmeniz tavsiye olunur.

UYGULAMA FAALİYETİ -2

Bir kütüphaneye okuyucu kaydı online olarak yapılmak istenmektedir. Bunun için veritabanında aşağıdaki veritabanını oluşturunuz ve PHP ile etkileşimli hale getiriniz.

Okuyucu Veritabanı Tablosu

tc_no
soyadi
adi
bolum
sube
okul_no
dogum_tarihi

Tablo 2.1

İŞLEM BASAMAKLARI	ÖNERİLER
1- Veritabanı adını okuyucu olarak verin.	
2- okuyucu veritabanına yukardaki değişkenlerden oluşan bir oku tablosu oluşturun.	
3- uygulamanın okuyucu.html adıyla web (kullanıcı) arayüzünü hazırlayınız.	
4- Kullanıcı ara yüzünde girilen bilgilerin veritabanına kaydedilmesi için okuma.php adlı PHP dosyasını oluşturunuz.	

MODÜL DEĞERLENDİRME

PERFORMANS TESTİ (YETERLİK ÖLÇME)

Modül ile kazandığınız yeterliği, öğretmeniniz işlem basamaklarına göre kriterler evet ya da hayır şeklinde değerlendirecektir.

DEĞERLENDİRME KRİTERLERİ	EVET	HAYIR
➤ Şekil 1.2 ye göre MAGAZA adlı bir veri tabanı oluşturduunuz mu?		
➤ musteri, urun ve siparis adlı tabloları oluşturduunuz mu?		
➤ Her tabloya 5(beş) 'er kayıt girdiniz mi?		
➤ Girilen kayıtlardan urun fiyatı 25YTL olan ürünleri listelediniz mi?.		
➤ Müşteri adı Mesut ve sipariş tarihi 30/04/2007 olan kayıtları görüntülediniz mi?		
➤ Girmiş olduğunuz müşteri kayıtlarından ikisini sildiniz mi?		
➤ Ürünlerden bazılarının fiyatlarını değiştirdiniz mi?		
➤ Ürün fiyatı en pahalı olan ürün hangisidir, sorguladınız mı?		
➤ Tabloları sildiniz mi?		
➤ Magaza veritabanını sildiniz mi?		
➤ Tablo 2.1 i kullanarak Veritabanını oluşturduunuz mu?		
➤ Tablo 2.1 deki alanlardan oluşan bir oku isimli bir tablo oluşturduunuz mu?		
➤ Uygulamanın okuyucu.html adıyla web (kullanıcı) arayüzünü hazırladınız mı?.		
➤ Kullanıcı ara yüzünde girilen bilgilerin veri tabanına kaydedilmesi için okuma.php adlı PHP dosyasını oluşturduunuz mu?		

DEĞERLENDİRME

Yaptığınız değerlendirme sonucunda eksikleriniz varsa öğrenme faaliyetlerini tekrarlayınız.

Modülü tamamladınız, tebrik ederiz. Öğretmeniniz size çeşitli ölçme araçları uygulayacaktır. Öğretmeninizle iletişime geçiniz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ 1

1	C
2	B
3	A
4	B
5	A
6	D
7	C
8	A
9	A
10	A
11	select adi, ogrno from ogrenci where adi not like '%M%';

ÖĞRENME FAALİYETİ 2

1	A
2	B
3	A
4	C

ÖNERİLEN KAYNAKLAR

- ŞAMLI, Mehmet, PHP 5, İstanbul, 2006
- WELLING, Luke, PHP ve MYSQL Uzmanlar İçin, İstanbul, 2006
- <http://programci.wordpress.com>
- <http://www.capraz.net>
- <http://www.ceviz.net>
- <http://www.programlama.com>
- <http://www.mysql.com.tr>

KAYNAKÇA

- KÖSEOĞLU, Kerem, Veritabanı Mantığı, Pusula Yayıncılık, 2005, İstanbul
- ALTINKAYA, Muhittin, DEMİRCAN, Yahya, MySQL ve PHP Programlamaya Giriş
A.Ü.F.E.F. , Ankara, 2005
- WELLING, Luke, LAURA, Thomson, PHP ve MYSQL Uzmanlar İçin, Alfa Yayınları, İst,
2006
- OTANER, Kayra, PHP ve MySQL ile web yazılım geliştirme, İstanbul, 2002
- <http://www.belgeler.org>
- <http://www.ceviz.net>
- <http://www.trojan-tr.org>
- <http://www.mysql.com>
- <http://www.csharpnedir.com>