SQL Sorguları

// insert

INSERT INTO user VALUES( null, 'Ali', 'Bilmem', 'ali@mail.com', '12345', 30 )

// Toplu insert

INSERT INTO user VALUES( null, 'Erkan', 'Bil', 'erkan@mail.com', '12345', 25 );

INSERT INTO user VALUES( null, 'Serkan', 'Bilki', 'serkan@mail.com', '12345', 33 );

// sadece bazı elemanların ekleneceği insert işlemi

INSERT INTO user( name, email, password ) VALUES ( 'Zehra', 'zehra@mail.com', '54321' )

// MD5 fonksiyonunu kullanarak insert yapma

INSERT INTO user VALUES( null, 'Ali', 'Bilmem', 'ali@mail.com', MD5('12345'), 30 )

// tarihi otomatik hesaplayıp ekleme yapma

INSERT INTO product VALUES( null, 'Buzdolabı', 15000, now() )

// ara tablo veri ekleme

INSERT INTO basket values( null, 1, 3, now() )

// farklı bir tablo içindeki datalara göre ekleme yapma

// genel olarak öneri önce mevcut tablonun temizlenmesidir. bu yüzden TRUNCATE kullanılır.

TRUNCATE userbackup;

INSERT INTO userbackup(name, email, date) SELECT CONCAT(name, ' ', surname), email, now() FROM USER;

// Enum

// tekil değerler kaydetmek istediğinizde kullanacağınız veri türüdür. -> 'work'

// SET

// Birden fazla özelliği seçerek veriyi kayıt yapabileceğiniz veri türüdür. -> 'red,green'

INSERT INTO `address` (`aid`, `uid`, `title`, `city`, `date`, `status`, `colors`) VALUES (NULL, '10', 'İş adresi', 'İstanbul', '2022-03-24 13:49:04.000000', 'work', 'red,green');

// update

UPDATE user SET name = 'Ahmet', surname = 'Bilmem', email = 'ali@mail.com', password = '12345', age = 30 WHERE uid = 1

// sadece belli sütunların değerlerini değiştirme

UPDATE user SET name = 'Ali', surname = 'Bilmem' WHERE uid = 1;

// tüm unsurları değiştirme

UPDATE user SET name = 'Ahmet', surname = 'Bilmem', password = '12345', age = 30

// mail adresi ali@mail.com olan kişilerin yaşını 40 yap

UPDATE user SET age = 40 WHERE email = 'ali@mail.com'

// delete

// uid değeri 4 olan satırı sil

DELETE FROM user WHERE uid = 4

// adı "Ali" soyadı "Bilmem" olan satırları sil

DELETE FROM user WHERE name = 'Ali' and surname = 'Bilmem'

// yaşı 30 dan büyük olanları sil

DELETE FROM user WHERE age > 30

// tablodaki tüm verileri temizleme id değerleri dahil her şeyi sıfırlar

TRUNCATE \* FROM `userbackup`

// id'lder hariç data içerisinde sıfırlama yapar.

DELETE FROM `userbackup`;

// Select

// tüm dataları alma

SELECT \* FROM user

// yaşı 30 veya 30 dan büyük olanları al

SELECT \* FROM user WHERE age >= 30

// limit kullanımı

// 0. satırdan başla ve 2 tane satır getir.

SELECT \* FROM user LIMIT 0, 2

// yaşı 25 ten büyük olanları için 2 satır getir.

SELECT \* FROM user WHERE age > 25 LIMIT 0, 2

// string ifadelerde arama yaparak select işlemi

// LIKE büyük küçük karakter duyarlılığı olmayan bir fonksiyondur.

// surname değeri "Bil" olanları getir

SELECT \* FROM user WHERE surname LIKE 'Bil'

// % sol tarafta ise aranan ifadede sol taraf önemli değil sağ tarafa uyanları getir.

SELECT \* FROM user WHERE surname LIKE '%Bil'

// % sağ tarafta kullanılır ise sol tarafki aranana uyanları getir

SELECT \* FROM user WHERE surname LIKE 'Bil%'

// % solda ve sağda kullanıldığında arada ne varsa aranacak ifadede bunu arar ve bulduklarını getirir.

// surname biriminde içinde "kim" geçenleri getir.

SELECT \* FROM user WHERE surname LIKE '%kim%'

// Karakter aralığına göre arama yapma

// başlangıcı A bitişi R ile biten ve aralıkta 14 karakterin olması gerektiği title elemanlarına ait satırları bana getir.

SELECT \*from film WHERE title LIKE 'A\_\_\_\_\_\_\_\_\_\_\_\_\_\_R'

// join -> inner join

// farklı tabloların birleştirilmesi için kullanılır.

// inner ->bire bir eşleşen tabloları getir

SELECT \* FROM basket INNER JOIN product ON basket.pid = product.pid INNER JOIN user ON basket.uid = user.uid

// Left Join -> sol tarafdaki tablonun karşılığı olmayan satırları içinde datalarını getir.

SELECT \* FROM `basket` LEFT JOIN product ON basket.pid = product.pid

// Right join -> sağ tarafdaki tablonun karşılığı olmayan satırları içinde datalarını getir.

SELECT \* FROM `basket` RIGHT JOIN product ON basket.pid = product.pid;

// Full Join -> şart olmaksızın tüm satırları diğer tablolarla birleştir

SELECT \* FROM `basket` FULL JOIN product;

// örnek join

SELECT \* FROM `payment` as p INNER JOIN customer as c ON p.customer\_id = c.customer\_id INNER JOIN address as a ON c.address\_id = a.address\_id INNER JOIN rental as r ON r.rental\_id = p.rental\_id INNER JOIN inventory as i ON i.inventory\_id = r.inventory\_id INNER JOIN film as f ON f.film\_id = i.film\_id

// as anahtar kelimesi -> tablo isimlerini değiştirmek için kullanılır.

SELECT \* FROM basket as b INNER JOIN product ON b.pid = product.pid INNER JOIN user ON b.uid = user.uid

// as anhatar kelimesi ile yazılmış join sorgusu

SELECT \* FROM basket as b INNER JOIN product as p ON b.pid = p.pid INNER JOIN user as u ON b.uid = u.uid

// as kullanmadan sütun isimlerini değişdirme " " boşluk ile olabilir.

SELECT first\_name name, last\_name surname from actor

// bu sorgu belli sütunların datalarını getirerek 3 farklı tabloyu birleşitirir

SELECT b.bid, b.date, p.title, p.price, u.name, u.surname, u.email FROM basket as b INNER JOIN product as p ON b.pid = p.pid INNER JOIN user as u ON b.uid = u.uid

// join işlemlerinde where yada limit gibi fonksiyonlar kullanbilir.

SELECT b.bid, b.date, p.title, p.price, u.name, u.surname, u.email FROM basket as b INNER JOIN product as p ON b.pid = p.pid INNER JOIN user as u ON b.uid = u.uid WHERE p.price > 10000 LIMIT 0, 2

// sıralama yapma

// asc, desc

// String yada tam sayı yada diğer türlerin hepsinde geçerlidir.

// DESC büyükten küçüğe doğru sıralama için kullanılır.

SELECT \* FROM film ORDER BY film\_id DESC

// ASC küçükten büyüğe doğru sıralam için kullanılır.

SELECT \* FROM film ORDER BY film\_id ASC

// and or ifadeleri kullanımı

// And -> solda bulunan ve sağda bulunan şartların sağlanmasını gerektirir

SELECT \*from film WHERE `rental\_rate` > 1 AND `length` > 60

// OR -> sağda yada solda bulunan şartlardan bir tanesinin geçerli olduğu durumlarda kullanılır.

SELECT \*from film WHERE `rental\_rate` > 1 OR `length` > 60

// concat -> sütunları birleştirmek için kullanılır.

SELECT a.actor\_id, concat( a.first\_name, ' ' ,a.last\_name ) as name , a.last\_update FROM actor as a

// null değerlere göre getirme

// isnull veya is not null

SELECT \* FROM film WHERE `original\_language\_id` is null

SELECT \* FROM film WHERE `original\_language\_id` is NOT null

// Aralık belirterek sorgu yapma

// sol taraftaki değişken küçük, sağ tarafta bulunan değişken büyük olmalıdır.

// between -> rental\_rate değeri 1.0 ile 5.0 arasında olan film satırlarını getir

SELECT \* FROM film WHERE `rental\_rate` BETWEEN 1.0 and 5.0

// 2 tarih arasındaki dataları getirme

SELECT \* FROM film WHERE `last\_update` BETWEEN '2006-02-15 05:03:42' and '2008-02-15 05:03:42';

// <> farklılık yaparak sql datalarını getirme

SELECT \* FROM film WHERE `length` <> 50;

SELECT \* FROM film WHERE `length` != 50

// in kullanımı ( select in select table )

// bir aramada where karşılığında birden fazla arama yapmak kullanılır.

// film\_id değer (1,2,3,4,5) olan satırları getir.

SELECT \* FROM film WHERE `film\_id` IN (1,2,3,4,5)

// tablo içinde farklı bir tablodaki değerlere göre "in" ile sorgu yapılabilir.

SELECT \* FROM country WHERE `country\_id` IN ( SELECT country\_id FROM city )

// count -> bir tablo içindeki satırtların sayısını getirmek için kullanılır.

SELECT COUNT(`country\_id`) AS size FROM country

// Sum -> sütun değerlerinin toplanmasını sağlar.

SELECT SUM(`amount`) as totalAmount FROM payment

// avg -> ortalama alma

SELECT AVG(`rental\_rate`) avgData FROM `film`

// max -> tablo içindeki en yüksek değeri getirir

SELECT MAX(`amount`) FROM `payment`;

// min -> tablo içindeki en düşük değeri getirir

SELECT MIN(`amount`) FROM `payment`

// subString -> 1,10 -> "1" -> başlangıç karakteri, "10" ise toplamda kaç karakter geleceğini ifade eder.

SELECT film\_id, SUBSTRING(title, 1,10) as title FROM `film`

// fullText index araması

// bir cümle içindeki geçen benzer kelimelerin aranması için kullanılır.

// bulduğu sonuçları en benzer şekilde üst sıralarda tutar

// Kurallar -> kullanılacak tablonun motoru MyISAM olarak işaretlenmelidir.

SELECT \* FROM `film\_text` WHERE MATCH(`title`, `description`) AGAINST('BANG drama filmi izlemek istiyor' IN NATURAL LANGUAGE MODE );

// Group By -> Benzer olan sonuçları gruplamak için kullanılır.

SELECT \* FROM `address` GROUP BY `city\_id`;

// Procedure -> Saklı yordam

// slq içerisindeki daha önceden hazırlamış olduğumuz queryleri çalıştırmak için kullanılır.

SELECT \* FROM `film\_text` WHERE MATCH(`title`, `description`) AGAINST( search IN NATURAL LANGUAGE MODE );

// Trigger Olulşturma

// bir satıra ekleme, düzenleme yada silme işlemlerinden sonra veya önce farklı bir satırı etkileme işlemidir.

// "new" anahtar kelimesi yeni etkilenen satırın değerine ulaşmak için kullanılır.

CREATE TRIGGER `addressAddTrigger` AFTER INSERT ON `user` FOR EACH ROW BEGIN SET @uid = new.uid; INSERT INTO address VALUES( null, uid, '', '', now() ); END

// View

// sanal tablo oluşturmak için kullanılır.

// büyük sql sorgularında çalıştırma anında bu tablolar üretlir ve kullanıma hazır hale gelir.

SELECT p.payment\_id, p.amount, c.first\_name, c.last\_name, c.email, a.address, f.title FROM `payment` as p INNER JOIN customer as c ON p.customer\_id = c.customer\_id INNER JOIN address as a ON c.address\_id = a.address\_id INNER JOIN rental as r ON r.rental\_id = p.rental\_id INNER JOIN inventory as i ON i.inventory\_id = r.inventory\_id INNER JOIN film as f ON f.film\_id = i.film\_id;

// function

// java veya c# gibi dillerde erişim mümkün değildir.

// içerisine parametre gönderilebilir.

// geriye bir tür dönderebilir ama sql tablosu gönderimi mümkün değildir.

BEGIN

DECLARE lname VARCHAR(20);

SELECT name INTO lname FROM language WHERE language\_id = languageId LIMIT 0, 1;

RETURN lname;

END;

// FOREIGN KEY ->

ALTER TABLE `phone` ADD CONSTRAINT `fk\_user\_phone` FOREIGN KEY (`uid`) REFERENCES `user`(`uid`) ON DELETE RESTRICT ON UPDATE RESTRICT;

// Rand() -> karışık sırada değerler oluşturmak için kullanılır

SELECT \* FROM `user` ORDER BY Rand();

// her kullanıcı için özel bir random değer üreterek listeleme işi yapar

SELECT \*, CAST( (Rand() \* 100) as Integer ) as ref FROM `user`