Database nedir

Bu kurs, SQL'in (Yapılandırılmış Sorgu Dili) temellerini tanıtacaktır.

Bu kursun sonunda,

* veritabanlarıyla etkileşime geçmek için temel SQL'i öğrenmiş olacaksınız.
* İlişkisel Veritabanlarının temellerini
* Tek veya birden çok tablodan veri almayı
* Fonksiyonlar ile verileri işlemeyi
* Alt sorgularla verileri işlemeyi
* Tablolar oluşturumayı ve değişiklikler yapmayı

Öğrenmiş olacaksınız

Bugünün oturum konularını tanıyalım.

* Database nedir?, Bir veritabanının tanımıyla başlayacağız.
* Database in içindekiler nedir, Ardından, veritabanı kavramını ve günlük faaliyetlerimizde kullanımını öğreneceksiniz.
* SQL nedir—, Sonra veritabanının içinde ne olduğunu keşfedeceğiz

Yapılandırılmış sorgulama dili (bir database olacak, bir veri olacak ve biz bu bilgileri sorgulayacağız.)

* SQL dili elements

Ardından, veritabanıyla iletişim kurmak için SQL'i (Structured Query Language) giriş yapacağız.

Database denince aklımıza ne geliyor

Bir veri tabanı, bilgilerin düzenli bir şekilde saklandığı ve yönetildiği bir bilgisayar sistemi içindeki yapılandırılmış bir veri topluluğudur. Büyük miktardaki verilerin verimli ve güvenilir bir şekilde saklanması, geri alınması ve yönetilmesi için tasarlanmış bir yazılım sistemidir.

Bir veri tabanı, kullanıcıların verileri hızlı ve kolay bir şekilde depolayıp erişmelerini sağlayan bir elektronik dosyalama sistemine benzetilebilir.

Veritabanı, bir bilgisayar sisteminde organize bir şekilde dataların saklanması..”

Burada bir veri olacak, o veriler organize olacak, yapılandırılacak ya da yapılandırılmayacak, bir de bunlar elektronik ortamda saklanacak. (aksi halde veri tabanı olmaz)

Veri tabanı, bilgisayar sistemlerinizde verilerinizin küme olarak, bir bütün olarak saklanmış hali de denebilir.

Bu veriler, text olabilir, resim olabilir, video olabilir, tablo olabilir. Bunu sadece text olarak düşünmeyin. Dolayısıyla organize olarak depolanmış veriler bütünü, bir de bunların bilgisayar sistemlerinde saklanmış olduğunu düşüneceğiz. Veri tabanı dediğimizde aklımıza gelebilecek iki önemli şey bunlar.

Database kullanılmayan bir program yok, her yerde kullanılıyor.

Veritabanları, çevrimiçi alışveriş, bankacılık, sağlık hizmetleri, lojistik ve birçok farklı uygulama alanında kullanılmaktadır. Modern bilgi işlemde vazgeçilmezdir ve neredeyse her işletme ve hükümet faaliyetinin desteklenmesinde kullanılırlar.

e-devlet, e-nabız bunlara baktığımızda çok büyük bir database sistemi vardır.

e-ticaret sitelerine baktığımızda, sosyal medya, üniversite ders kayıtları, bunlar veri tabanı üzerinden yapılıyor.

Günümüzde özel ya da kamu kuruluşlarının tümü, içerdikleri bilgileri kendi veri tabanlarında tutmaktadır.

Nüfus müdürlükleri, ulaşım merkezleri, bankalar, okullar ve üniversiteler kayıtlı kişiler arasından istenilen bilgilere saniyeler içinde ulaşabiliyorsa bu veri tabanı sistemleri sayesindedir.

Dünya verilerle dolu.

İnsanlar günlük yaşam aktivitelerinde farkında olarak veya olmayarak veri ve veritabanlarını kullanırlar.

bankamız, bakkalınız, cep telefonunuzdaki bir uygulama, web siteleri, verilerinizi takip etmek için veritabanlarını kullanır.

\*\*\*

Bir web sitesine eriştiğinizde, web sitesi verilerinizi (ör. erişim tarihi ve saati, konumunuz, tarayıcı bilgileriniz) toplamaya ve veri tabanında saklamaya başlar.

Başka bir örnek verelim.

Ticari bir web sitesinde bir ürün sipariş ettiğinizde, siparişiniz bir veri tabanında saklanır.

Banka hesabınızdan para çektiniz.

Bankanız bu işlemi veri tabanında saklar.

Facebook, Instagram gibi sosyal medya platformları,

Twitter, üyeler, arkadaşları, üye etkinlikleri, mesajları, reklamları vb. verileri depolamak için veri tabanlarını kullanır.

Database Menegement System

Bir veri tabanı tipik olarak bir veri tabanı yönetim sistemi (DBMS) tarafından kontrol edilir.

Veri tabanı, ve onu yönetecek bir yazılım gerekiyor, o veri tabanlarını yönetecek, giriş çıkış kullanıcı işlemleri, güvenlik gibi işlerden sorumlu.

DBMS, kurulu olan veri tabanının güvenlik ayarı, kullanıcı ve yetki ayarları ve bunun gibi konfigürasyonları kontrol edebileceğimiz bir yazılım sistemi.

Veritabanı Yönetim Sistemi (Database Management System - DBMS), kullanıcıların veri depolama, veri erişimi, veri yönetimi ve veri güvenliği için bir dizi yazılım uygulamasıdır. DBMS, kullanıcıların veri tabanlarını oluşturmasına, veri eklemesine, güncellemesine, sorgulamasına ve silmesine izin verir.

Bir DBMS, veri tabanı ile son kullanıcılar veya programlar arasında bir arabirim görevi görerek kullanıcıların bilgileri düzenlemesine ve optimize etmesine, almasına, güncellemesine ve yönetmesine olanak tanır.

Veriler ve DBMS, bunlarla ilişkili uygulamalarla birlikte bir veri tabanı sistemi olarak adlandırılır ve genellikle yalnızca veri tabanı olarak kısaltılır.

Bir veri tabanı tipik olarak veri tabanı yönetim sistemi (DBMS) olarak bilinen kapsamlı bir veri tabanı yazılım programı gerektirir.

SQL ----- ilişkisel veri tabanlarının dilidir. ( Releyşınıl ) (non

IBM'de çalışırken veritabanı yönetimi için ilişkisel modeli icat etti.

Aslında "ilişkisel veritabanı" terimi, 1970 yılında IBM'den E. F. Codd tarafından icat edildi. Codd, bu terimi "Büyük Paylaşılan Veri Bankaları için İlişkisel Veri Modeli" adlı araştırma makalesinde tanıttı.

Bu makalesinde ve daha sonraki makalelerinde "ilişkisel" ile neyi kastettiğini tanımladı.

Ancak Boyce ve Chamberlin, bu yeni moda "ilişkisel veri tabanlarından" bilgi talep etmenin standart bir yolunu oluşturmak için bir araya geldi. Buna kısaca "Yapılandırılmış İngilizce Sorgu Dili" veya "SEQUEL" adını verdiler. Daha sonra daha da kısaldı ve “Yapı Sorgulama Dili” veya “SQL” oldu.

Kullanıcılar browser den veya haberleşme programları sayesinde buradaki database lere erişmek için SQL kullanılıyor.

Kullanıcı ile veri tabanı arasında bir yazılım daha var;

APİ ------- Appilication programing interface

Uygulama programlama arayüzü

Bu da bir yazılım --- görselliği olmayan bir yazılım, bir ara program (garson )

API clint tarafındaki işlemleri kolaylaştırıyor. Clint in (teferruatta boğulmasını ) SQL ile uğraşmasını engelliyor. Karmaşıklığı azaltmış oluyor.

API (Application Programming Interface), farklı yazılım sistemleri arasındaki iletişimi sağlayan bir arayüzdür. Bu arayüz, bir yazılımın başka bir yazılımdan veri almasını, veri göndermesini veya birlikte çalışmasını sağlar.

API'lar, bir yazılımın işlevselliğini genişletmek veya başka bir uygulamayla entegre etmek için kullanılabilir. API'lar, yazılım geliştiricilerinin ve uygulama sahiplerinin farklı sistemler arasında veri alışverişi yapmasını kolaylaştırır. Bu sayede yazılım sistemleri birbirleriyle uyumlu hale gelir ve daha işlevsel hale gelir.

diyagram, şematik içeren bir resim

Açıklama otomatik olarak oluşturuldu

SQL nedir, bir sorgulama dilidir

SQL (Structured Query Language), yapılandırılmış sorgu dili anlamına gelir. Veri tabanı yönetim sistemlerinde (DBMS) kullanılan bir programlama dilidir ve bu dil aracılığıyla veri tabanlarına erişilir, verileri saklanır, düzenlenir ve sorgulanır.

SQL, veri tabanı sistemlerinde kullanılan birçok işlemi gerçekleştirmek için kullanılır. Bu işlemler arasında veri tablolarının oluşturulması, verilerin ekleme, silme ve güncelleme işlemleri, veri tablolarının sorgulanması ve veri tabanları arasında ilişkilerin kurulması yer alır.

SQL, kullanımı kolay ve standart bir dil olduğu için birçok farklı veri tabanı yönetim sistemleri tarafından desteklenir. Bu nedenle, birçok farklı programlama diliyle birlikte kullanılabilir ve veri tabanı işlemlerini gerçekleştirmek için yaygın bir seçenek olarak kabul edilir.

Çoğu ticari veri tabanı sistemi SQL dilini kullanır.

SQL ile veri tabanında saklanan verilere erişebilir veya bunları değiştirebilirsiniz.

İlişkisel veri tabanları, nesne yönelimli veri tabanları, grafik veri tabanları (object-oriented databases, graph databases ) ve daha birçok türde Data Warehouses veri tabanı bulunmaktadır. Her veri tabanı türü, farklı türde verileri ve iş yüklerini yönetmek için tasarlanmıştır ve yeni bir uygulama veya sistem oluşturulurken doğru veri tabanı türünü seçmek önemli bir düşünce olabilir.

SQL (Yapılandırılmış Sorgu Dili anlamına gelir), standart İlişkisel Veri tabanı Yönetim Sistemi (RDBMS) dili olarak kabul edilir.

Bu yüzden genellikle İlişkisel Veri tabanını SQL olarak ve İlişkisel Olmayan veri tabanını NoSQL olarak adlandırmayı tercih ederiz.

(Not Only SQL)İlişkisel veritabanı, birbiriyle ilişkili veri noktalarını depolayan ve bunlara erişim sağlayan bir veri tabanı türüdür.

İlişkisel veri tabanları, verileri tablolarda temsil etmenin sezgisel ve basit bir yolu olan ilişkisel modele dayalıdır.

Tablonun sütunları, verilerin özniteliklerini tutar ve her kaydın genellikle her öznitelik için bir değeri vardır, bu da veri noktaları arasında ilişkilerin kurulmasını kolaylaştırır.

NewSQL, NoSQL veritabanı sistemlerinin aynı ölçeklenebilir performansını sunarken ilişkisel veri modelini destekleyen ürün tekliflerini tanımlamak için kullanılan bir terimdir.

NewSQL veritabanları, geleneksel çevrimiçi işlem işleme (OLTP) RDBMS ile ilgili bazı önemli sorunları çözen modern SQL veritabanlarıdır.

Geleneksel veritabanı yönetim sistemlerinin faydalarını korurken, NoSQL veritabanlarının ölçeklenebilirliğini ve gelişmiş performansını elde etmeyi amaçlıyorlar.

Başka bir deyişle, NewSQL veritabanları, NoSQL'in OLTP, yüksek performans ve ölçeklenebilirliğini birleştiren ilişkisel veritabanı sistemleridir.

Geleneksel DBMS'nin ASİT (Atomiklik, Tutarlılık, İzolasyon ve Dayanıklılık) garantilerini korurlar.

ACID işlemleri, eksiksiz iş süreçlerini, eşzamanlı işlemleri, sistem arızaları veya hataları durumunda verilerin hayatta kalmasını ve bir işlemden önce ve sonra tutarlılığı sağlar.

<https://statisticsanddata.org/data/the-most-popular-databases-2006-2023/>

SQLite

SQLite, küçük, hızlı, bağımsız, sunucusuz, yüksek güvenilirlikli, tam özellikli bir SQL veritabanı motorunu uygulayan bir C dili kitaplığıdır. SQLite, dünyada en çok kullanılan veritabanı motorudur. SQLite, tüm cep telefonlarında ve çoğu bilgisayarda yerleşiktir ve insanların her gün kullandığı sayısız başka uygulamanın içinde paket halinde gelir.

PostgreSQL (Postgres olarak da bilinir), genişletilebilirliği ve SQL uyumluluğunu vurgulayan ücretsiz ve açık kaynaklı bir ilişkisel veritabanı yönetim sistemidir (RDBMS)

Veritabanındaki bilgiler tablolar halinde gruplandırılmıştır.

Veritabanındaki bir tabloya veritabanı tablosu denir.

Tablolar, veritabanlarında veri depolamanın temel birimidir.

Yapılandırılmış veri terimini kullandık. Yapılandırılmış veri burada tablo anlamına gelir. Bir tablo sütunlardan ve satırlardan oluşur. Bunu bir Excel veya Google elektronik tablosu olarak düşünebilirsiniz. Veritabanı tablosu ile Excel/Google elektronik tablosu arasında benzerlikler olsa da, bunlar farklı şeylerdir. Burada ayrıntılara girmeyeceğiz. Bir tablo sütunlardan ve satırlardan oluşur. Sütun, tablo tarafından depolanan bir veri parçasıdır. Satır, tek bir şeyin niteliklerini tanımlayan tek bir sütun kümesidir. Sütunların benzersiz bir adı olmalıdır. Sütunlar ve satırlar birlikte bir tablo oluşturur.

Bu bir örnektir;

Bir şirket veritabanı örneğini ele alalım. Diyelim ki iki tablo var. Biri çalışanlar, diğeri departmanlar. Çalışanlar tablosuna daha yakından bakalım. İşte tablonun dökümü.

Tablonun adı çalışanlardır.

Tablonun yedi sütunu vardır (diğer adıyla alanlar veya nitelikler)

Tablonun on satırı vardır (aka, kayıtlar veya demetler)

Tablonun sütun başlık adları şunlardır: emp\_id, ad\_ad, soyad, maaş, iş\_ünvanı, cinsiyet, işe alma\_tarihi

Çalışanlar tablosunda, şirketteki her çalışan hakkında veriler bulunur.

Elbette, gerçek dünyadaki bir şirketin çok daha fazla çalışanı olacaktır. Kavramları göstermek için küçük bir tablo kullanıyoruz.

Tablo, sütunlar ve satırlar biçiminde depolanan düzenli bir veri (ilişkisel) koleksiyonudur. Sütunlar dikey, satırlar yatay olarak kategorize edilebilir. Bir tablodaki sütunlar, alanlar (öznitelik) olarak adlandırılırken, satırlar kayıtlar olarak adlandırılabilir.

Tablolar neden fazla---------------Performans ve hız

Tablolar ne kadar uzunsa erişim çok uzun olur

IT dünyasında her şeyi böleceksiniz, küçülteceğiz

SQL COMPONENTS

* **Data Manipulation Language** (**DML**)
* **Data Definition Language** (**DDL**)
* **Data Control Language** (**DCL**)
* **Transaction Control Language** (**TCL**)

SQL ile, veritabanında saklanan verilere erişebilir veya bunları değiştirebilirsiniz. T

İşte farklı erişim türleri.

Bunlar:

• Veritabanından read-----en önemli kısmı burası ve ilk göreceğimiz yer burası.

• Veritabanına yeni create

• Veritabanındaki verileri update

• Veri tabanından verilerin delete

SQL Language Elements

SQL Dilinin kuralları (Python kuralları gibi )

* Büyük küçük harf farketmez
* İnglizce gramer yazı şeklinde sıralama olacak ( sanki cümle kurar gibi, hangi koşulda ve aynı zamanda sırala(order by) --- neye göre number\_of\_employees’e göre)
* (;) olmalı sonda
* Tavsiyeler--- büyük küçük yazım
* Verileri tırnak içinde yazıyoruz

BNF (notasyonu) gösterimi nedir

Bilgisayar biliminde, Backus-Naur formu genellikle bilgisayar programlama dilleri, belge formatları, talimat setleri ve iletişim gibi bilgi işlemde kullanılan dillerin sözdizimini tanımlamak için kullanılan, bağlamdan bağımsız gramerler için bir meta sözdizimi notasyonudur. protokoller.

Değişkenlere böyle isim verirsem

Class-name -------🡪 kebab case

className----Camel case (deve gibi----- java gibi diller)

sneek case ----🡪 alt tre şeklinde yazılıyor (Python gibi)

SQL de verileri okuyabilmek için syntax ımız bu şekilde olmalı.

Keyword

Stetimınt----ifade

**Selecting column /columns/ all columns**

Artık veritabanımız ile SQL ile iletişim kurma zamanı.

Lütfen unutmayın, bu kursun amacı bir veritabanı modellemek veya tasarlamak değildir, bu tamamen farklı bir şeydir ve önce temelleri anlamanız gerekir.

Bu kursun odak noktası size veritabanları ile iletişim kurmanızı sağlayacak Yapılandırılmış Sorgulama Dilini temel düzeyde ve biraz da orta düzeyde öğretmektir. Yolunuza devam ettiğinizde farklı veritabanları kullanıyor olacaksınız. Ve onlarla iletişim kurarken kullanacağınız dil SQL olacaktır.

Örnekleri uygulayalım

/\* carsalesEmployees tablosundan name getir \*/

SELECT name FROM carsalesEmployees;

--carsalesEmployees tablosundan name ve surname sütunlarını getir

SELECT name, surname FROM carsalesEmployees;

--hepsi gelsin istiyorsak

SELECT \* FROM carsalesEmployees;

/\* Distinct filtresini kullanarak tekrarları kaldırıyoruz \*/

SELECT DISTINCT gender, dep FROM carsalesEmployees;

--where örneği

SELECT \*

FROM carsalesEmployees

WHERE gender='F';

--limit denemesi ve AND - OR örnekleri

SELECT \* FROM carsalesEmployees WHERE salary>2100 AND gender='M' LIMIT 1;

--order by sıralama işlemi yapacağız, DEFAULT ASC, sırayla önce neye göre sıralayacaksak onu sıralar

Verileri alfabetik veya sayısal sırayla almak istemeniz durumunda ORDER BY anahtar kelimesini kullanırız.

Varsayılan olarak ORDER BY anahtar kelimesi, kayıtları artan düzende sıralar.

Kayıtları azalan düzende sıralamak için DESC anahtar kelimesini kullanın. Verileri artan düzende sıralamak için ASC'yi açıkça kullanabilirsiniz.

SELECT \*

FROM carsalesEmployees

ORDER BY name, surname DESC;

SELECT gender, name, surname, salary FROM carsalesEmployees ORDER BY gender DESC, salary DESC;

--burada bu işleme bir de şart (where) koyalım

Bu bölümde ORDER BY'yi WHERE deyimiyle birlikte kullanacağız. ORDER BY cümlesi WHERE cümlesinden sonra gelir.

Teknik olarak herhangi bir SQL ifadesi tek bir satıra yazılabilir. Ancak uzun sorgular yazmaya başladığınızda okumak zorlaşacaktır. Bu gibi durumlarda çözüm, kodu sadece yatay değil dikey olarak da düzenlemektir. Buna güzelleştirme denir. Yukarıdaki sözdizimini yeniden yazalım.

SELECT gender, name, surname, salary

FROM carsalesEmployees

WHERE salary>3000

ORDER BY gender DESC, salary DESC;

--and, or, not örnekleri

SELECT \* FROM carsalesEmployees WHERE dep='Mechanic' AND salary>3000;

SELECT \* FROM carsalesEmployees WHERE dep='Manager' AND date='21.02.2018';

SELECT \* FROM carsalesEmployees;

SELECT \* FROM carsalesEmployees WHERE id<5 ;

--order by denemesi ile siralama yapmış olduk

SELECT name, surname FROM carsalesEmployees ORDER BY name ASC;

SELECT name, surname FROM carsalesEmployees ORDER BY name DESC;

SELECT gender, name, surname FROM carsalesEmployees ORDER BY gender DESC, name ASC;

SELECT name, surname, gender, salary FROM carsalesEmployees where salary>=3100 ORDER BY salary DESC, name ASC;

SELECT \* FROM carsalesEmployees WHERE dep='Mechanic' AND salary>3000;

SELECT \* FROM carsalesEmployees WHERE dep='Mechanic' AND date='22.05.2019';

--between örnekleri

SELECT \* FROM carsalesEmployees WHERE salary>=2000 AND salary<=3000;

SELECT \* FROM carsalesEmployees WHERE salary BETWEEN 2500 and 3000;

NOT BETMEEN de kullanabiliriz.

BETWEEN operatörünü tarihlerle birlikte kullanmak da mümkündür. Lütfen tarih değerlerinizi tek tırnak (') içine alın ve sorgunuzda YYYY-AA-GG tarih biçimini kullanın.

logo içeren bir resim

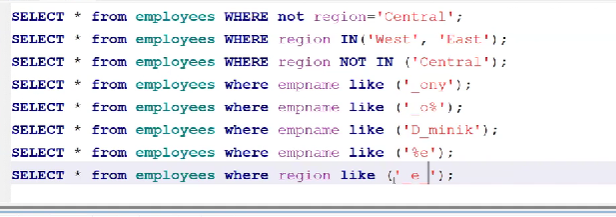
Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu



metin içeren bir resim

Açıklama otomatik olarak oluşturuldu



1. Dersin sonu

Genel Tekrar

* **SELECT** \* **FROM** employees**;**
* SELECT DISTINCT dep FROM employees; ---------DISTINCT, selectten hemen sonra yazılır.
* SELECT \* FROM employees LIMIT 2;
* SELECT column\_name(s) FROM table\_name WHERE condition(s);

**SELECT** empname,salary **FROM** employees **WHERE** salary>3000; -----where tablo isminden sonra yazılır.

**SELECT** \* **FROM** employees **WHERE** gender='M' **LIMIT** 2;

* SELECT empname, gender, salary FROM employees ORDER BY gender, salary DESC;
* SELECT \* FROM employees WHERE job='Mechanic' OR gender='F';
* SELECT \* FROM employees WHERE dep='Manager' AND gender='M'
* SELECT \* FROM employees WHERE NOT job='Salesperson'

SELECT \* FROM carsalesEmployees WHERE dep='Manager' AND date='21.02.2018';

SELECT \* FROM carsalesEmployees WHERE region='West' OR dep='Manager';

SELECT \* FROM carsalesEmployees WHERE NOT gender='F';

--alttaki iki satir ayni işlemi yapıyor

SELECT \* FROM carsalesEmployees where salary>=2500 AND salary<=3000;

SELECT \* FROM carsalesEmployees WHERE salary BETWEEN 2500 AND 3000;

* SELECT \* FROM employees WHERE salary BETWEEN 2500 AND 3000; ----- or yerine kullanılabiliyor

SELECT \* FROM sales WHERE name BETWEEN 'A' and 'B';

* SELECT \* FROM sales where carBrand='Mercedes' and salesDate='2022-02-01';
* SELECT \* FROM sales where salesDate BETWEEN '2022-02-01' and '2022-09-14';

--bir sütun içerisinde birden fazla seçenek seçebiliyorum (iki satır da aynı sonucu verir)

SELECT \* FROM carsalesEmployees WHERE region IN('West', 'East');

SELECT \* FROM carsalesEmployees WHERE region='West' OR region='East';

SELECT \* FROM carsalesEmployees WHERE region NOT IN('West', 'East');

* **SELECT \* FROM** employees **WHERE** job **IN** ('Manager','Mechanic');
* SELECT \* FROM employees WHERE job NOT IN ('Manager','Mechanic');
* SELECT empname,empsurname FROM employees WHERE empsurname LIKE 'W%';
* SELECT empname,empsurname FROM employees WHERE empname LIKE '\_a%';

SELECT \* FROM carsalesEmployees WHERE name like 'R%';

SELECT \* FROM carsalesEmployees WHERE dep like 'M%c';

Aggregate Functions

Bazı SQL ifadeleri yazarken, arama yaparken bazı fonk ihtiyaç duyarız

Elimdeki kayıtların ortalamasını getir diyebiliyoruz

Satır, bir kaydı temsil ediyordu

Sütun neydi, o kayıtlara ait özelliklerdi.

Belli bir özelliğe ait bilgileri bir fonksiyona tabi tutarak bir sonuç elde ediyoruz.

SUM----toplamı

AVG----ortalamasını

Slaytla birlikte örnek yapalım

COUNT---sayısını verecek

İnsert, data (veri) ekleyelim----NULL değeri nedir, hiç, boş demek-------CTRL + S

--toplam kaç personel var, nul olmayan değeri saydırmalıyım ki gerçek değeri bulayım

SELECT count (\*) FROM carsalesEmployees;

SELECT \* FROM carsalesEmployees;

SELECT count(name) FROM carsalesEmployees;

SELECT count(dep) FROM carsalesEmployees;

--sadece null olanların sayısını ver deyince

SELECT count(name) FROM carsalesEmployees WHERE dep IS NULL;

--count a takma isim verebiliyoruz, tepesine yazacağı isimi kendimiz belirleyebiliyoruz

SELECT count(\*) as toplam\_personel\_sayısı FROM carsalesEmployees;

Count DİSTİNC

--sayarken tekrar edenleri listeden çıkaralım

SELECT count(DISTINCT region) as bolge\_sayısı FROM carsalesEmployees;

Min, max, avg---ortalama, round---sayının virgülden sonrasını alma demek veya

round(sayı,2) -- round(avg(salary),2) --- sayının virgülden sonraki kısmının 2 basamağını al sadece demek

min(sayı)-max(sayı) ---- bize sonucunu verir, yani çıkarma işlemi yapar

SELECT min(salary) FROM carsalesEmployees;

SELECT max(salary) FROM carsalesEmployees;

SELECT avg(salary) FROM carsalesEmployees;

SELECT min(salary) as andusuk, max(salary) as en\_yuksek, avg(salary) as ortalama

FROM carsalesEmployees;

SELECT min(salary) as andusuk, max(salary) as en\_yuksek, round(avg(salary),2) as ortalama

FROM carsalesEmployees;

--min ve max aldığımız sayılarla matematiksel işlem yapabiliriz

SELECT min(salary)-max(salary) as fark FROM carsalesEmployees;

SELECT salary, salary\*1.15 as zamli\_maas FROM carsalesEmployees;

--maasları yuzde 15 zam yapmanın sirkete getirdigi maaliyet

SELECT sum(salary)-sum(salary\*1.15) FROM carsalesEmployees;

--en eski personel kimmis

SELECT name, min(date) FROM carsalesEmployees;

* Ortlama maaşı hesapla
* ortalama maas max maas dan ne kadar az, min dan ne kadar fazla
* bunu farklı başlık ismi ile ve sadece isim, soyisim ve maaş şeklinde sorgula
* min maas hesapla

SELECT name, max(salary) FROM carsalesEmployees;

SELECT name, round(avg(salary)) FROM carsalesEmployees;

SELECT name, max(salary)-round(avg(salary)) FROM carsalesEmployees;

SELECT min(salary)-round(avg(salary)) as maasortalamasi\_minmaas\_farki FROM carsalesEmployees;

SELECT sum(salary) FROM carsalesEmployees;

SELECT sum(salary)/ count(id) FROM carsalesEmployees;

SELECT sum(salary)/ count(salary) FROM carsalesEmployees;

SELECT avg(salary) FROM carsalesEmployees;

SELECT count(name) FROM carsalesEmployees WHERE dep IS NULL;

SELECT \* from carsalesEmployees ORDER BY salary LIMIT 3;

SELECT \* from carsalesEmployees WHERE salary NOT NULL ORDER BY salary LIMIT 3;

Gruplar

Gurupladıktan sonra matematiksel işlem yapar

SELECT \* FROM carsalesEmployees;

SELECT count(\*) as toplamsayi FROM carsalesEmployees GROUP BY gender;

SELECT region, count(\*) as herbolgedekipersonelsayisi FROM carsalesEmployees GROUP BY region;

SELECT dep, count(\*) as hermeslektekipersonelsayisi FROM carsalesEmployees GROUP BY dep;

SELECT region, gender, count(\*) as herbolgedekipersonelsayisicinsiyetegore FROM carsalesEmployees GROUP BY region, gender;

--maasa gore gurupla desem

--veya her bolgedeki en yuksek maasli personel

SELECT name, region, max(salary) FROM carsalesEmployees GROUP BY region;

**SELECT** gender, **MAX** (salary) **AS** maxsalary

**FROM** employees

**GROUP BY** gender

**ORDER BY** maxsalary **DESC**;

VSCodda açmak için

SQLite

Ve 1. SQLite 2. Sqlite wiever extenşınları yükleyelim

Wiev ----command palette

SQLite-open palette

Altta SQL Explorer--- alttaki tabloya sağ tıkla ---new query

Select i yazdık tan sonra run query yap

Çoklu tablolarda Sorgulama

Herhangi bir veri tabanı çalışması yapılacaksa içeriği dami data olan veri tabanları mevcut, oturup siz tek tek uğraşmıyorsunuz da hazır girilmiş veriler var siz onu sadece uygulamanıza bağlama veya SQL sorgulaması yapmak için kullanıyorsunuz.

Onlardan birisi de SQLite için hazırlanmış chinook (plak şirketi) veri tabanı.

İşte dijital müzik mağazası veritabanının ERD Şeması.

ERD nedir biliyor musun bu arada?

ERD, ER Diyagramı veya ER modeli olarak da bilinen Varlık İlişkisi Şeması, veritabanı tasarımında kullanılmak üzere bir tür yapısal diyagramdır. Bir ERD, iki önemli bilgiyi görselleştiren farklı semboller ve bağlayıcılar içerir: Sistem kapsamındaki ana varlıklar ve bu varlıklar arasındaki karşılıklı ilişkiler. İşte bu yüzden "Varlık" "İlişki" diyagramı (ERD) olarak adlandırılır!

1. Trac tablosunu kullanarak trac isimlerini getiren bir sorgu yaz.

SELECT Name FROM tracks;

1. Tracs tablosundan name ve composer (besteci) bilgilerini getir

SELECT Name, Composer FROM tracks;

1. Album tablosundaki tüm bilgileri getir

Select \* from albums;

1. Tract tablosundan AlbumId ve MediaTypId yi getirsin ama Album Id ye göre filtrelesin (DISTINCT)

SELECT DISTINCT AlbumId, MediaTypeId FROM tracks;

1. Parça tablosunu kullanarak her parçanın bestecisinin adını bulun
2. Tracs da bestecisi Jımi hendrix olan tüm bilgileri getir

SELECT \* FROM tracks WHERE Composer='Jimi Hendrix';

1. İnvoices tablosundaki tüm bilgileri getir, 10 dan büyük olanlar olsun

SELECT \* FROM invoices WHERE total>10;

1. İnvoices tablosundaki tüm bilgileri getir, 10 dan büyük olanlar olsun, limit 4 tane olacak

SELECT \* FROM invoices WHERE total>10 limit 4;

SELECT \* FROM invoices WHERE total>10 order by total limit 4;

1. Toplam tutarı 10 ABD Dolarından fazla olan faturaların tüm bilgilerini bulun. Ardından bunları toplam miktara göre azalan düzende sıralayın.

SELECT \* FROM invoices WHERE total>10 order by total decs;

1. Fatura ülkesinin ABD olmadığı faturaların tüm bilgilerini bulun. Ardından bunları toplam miktara göre artan düzende sıralayın

SELECT \* FROM invoices WHERE BillingCountry not in ('USA') ORDER by total;

SELECT \* FROM invoices WHERE BillingCountry is not 'USA' ORDER by total;

SELECT \* FROM invoices WHERE BillingCountry <> 'USA' ORDER by total;

1. 2009 ile 2011 arasındaki fatura tarihleri ​​arasından en yeni fatura tarihini bulun.

SELECT \* FROM invoices WHERE InvoiceDate BETWEEN '2009-01-01' and '2011-01-01' ORDER by InvoiceDate DESC LIMIT 1

SELECT max(invoiceDate) FROM invoices WHERE InvoiceDate BETWEEN '2009-01-01' and '2011-01-01';

SELECT max(invoiceDate) FROM invoices WHERE InvoiceDate>'2009-01-01' and InvoiceDate<'2011-01-01';

1. Belçika, Norveç, Kanada ve ABD'den sipariş veren müşterilerin ad ve soyadlarını bulun

Customers tablosunda

SELECT Country, FirstName, LastName FROM customers

WHERE Country IN ('Belgium', 'Norway', 'Canada', 'USA') ORDER BY Country;

1. Bach ın parçalarının ismini soruyor

SELECT Composer, name FROM tracks WHERE Composer like '%Bach';

1. Dijital müzik mağazasında kaç fatura var? (invoices)

SELECT count(\*) as fatura\_sayisi FROM invoices;

Her composer e ait kaç tract var ( kaç tane null var görebiliriz)

SELECT Composer, count(\*) FROM tracks WHERE Composer is NULL;

SELECT Composer, count(\*) FROM tracks GROUP BY Composer;

1. Dijital müzik mağazasında kaç tane besteci var?

SELECT count(DISTINCT Composer) as besteci\_sayisi FROM tracks;

1. Minimum süreye sahip parça adını bulun

SELECT \* FROM tracks ORDER BY Milliseconds LIMIT 1;

SELECT name, min(Milliseconds) FROM tracks;

1. Maksimum süreye sahip parça adını bulun.

SELECT name, max(Milliseconds) FROM tracks;

1. Mağazamız ne kadar para kazandı?

SELECT sum(total) as toplam\_kazanilan\_para FROM invoices;

1. Süresi ortalama süreden daha uzun olan parçaları bulun (iç içe sorgularda bakacağız)

Önce ortalamayı bulacağız sonra where ile büyüğünü bulacağız

1. Her bestecinin parçasının toplam sayısını bulun. Sonucunuz bestecinin adını ve numarasını içerecektir

SELECT Composer, count(\*) FROM tracks GROUP BY Composer;

1. Her ülkeden kaç müşterimiz var? Sonucunuz ülke adını ve numarasını içerecektir.

SELECT BillingCountry, count(\*) FROM invoices GROUP BY BillingCountry;

SELECT BillingCountry, count(DISTINCT CustomerId) FROM invoices GROUP BY BillingCountry;

SELECT Country, count(\*) FROM customers GROUP BY Country;

Müşteri sayısıyla fatura kestiğim müşteri sayısı aynı olduğu için sonuç aynı çıkıyor

1. Her albüm için minimum parça süresini bulun. Sonucunuz albüm kimliğini ve minimum süreyi içerecektir

SELECT AlbumId FROM tracks GROUP BY AlbumId;

SELECT AlbumId, min(Milliseconds) FROM tracks GROUP BY AlbumId;

1. Her ülke için toplam fatura tutarını bulun. Sonucunuz ülke adını ve toplam tutarı içerecektir

SELECT BillingCountry, sum(total) as fatura\_toplami FROM invoices GROUP BY BillingCountry;