3. SUNU

Biz tabloda bazı istenmeyen durumlar var, bunlar nedir.

Veri tabanının, verilerin tutulduğu tabloların parçalanmasıyla ilgili normalizasyon dediğimiz işlemler yapılıyor. Pratik olarak bir tablo nasıl parçalanır, niye parçalanır bunu anlatmaya çalışacağız.

Bir tabloyu niye parçalamamız gerekir ona bakalım;

* Biz bir tabloya baktığımızda null kayıt görmeyi tercih etmeyiz.
* Veri tekrarı olmasını istemeyiz
* Bir de mantıksal olarak ilişkili verilerimizi bir araya toplamaya çalışırız.

Gender----tekrar ediliyor, biz bunun tekrar etmesini istemiyoruz nasıl yapacağız.

Region

Job

* Bunların tekrar etmesini istemiyorsak ayrı bir tabloda tutup buraya bir id vereceğiz ve o id den ilişki kurup 1 e 1 ilişki olacak.
* Byte olarak yerlerden tasarruf etmiş olacağız. String daha çok yer kaplayacak.
* İsimlerde, maaşlarda, tarih de bunu uygulamıyoruz.
* Null tekrarı olunca ne yapacağız, mesela adres bilgisini saklayan sütunumda null değerler olabilir.
* Biz genel tablomuzda bu adres bilgisini tutmayız. Farklı bir tabloda tutarız ama ana tablodaki adress sütununu kaldırırız çünkü yer kaplamasın diye, id ile ilişkilendirip oradan istersek bakabiliriz.

metin içeren bir resim

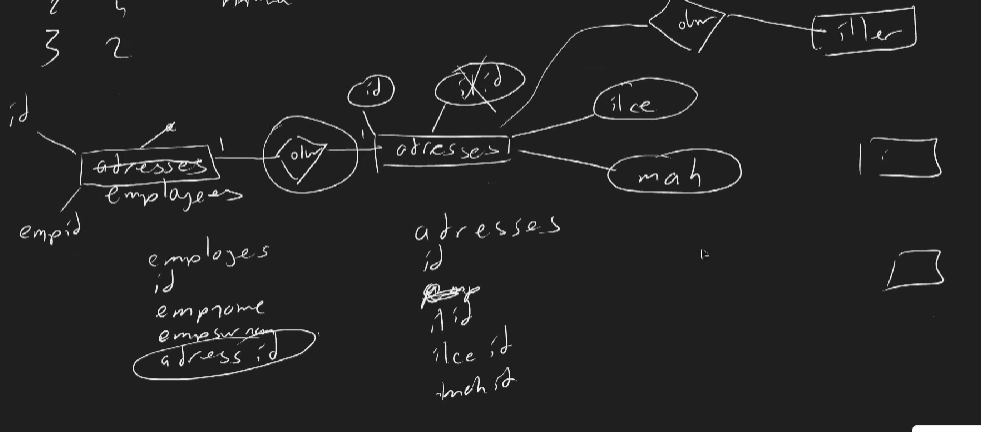
Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu



Mesela istanbul da 16 milyon insan yaşıyor, e-devlette 80 milyon insanın bilgisi var. Burada 16 milyon defa istanbul yazmıyorda 16 milyon tane 34 rakamı yazıyor. Bu şekilde ben rakamlar üzerinden adresime erişmiş oluyorum, daha doğrusu adresimi karşıladığım tablolar arası ilişkiyle oluşturmuş oluyorum. Bu biraz tecrübe istiyor. User bunları görmez, arka planda bunlar otomatik olarak işler.

metin, cihaz, metre, kontrol paneli içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek ; şirket'deki çalışanlar ve departmanlar tablolarını kullanacağız

Aşağıdaki ifade, çalışanların emp\_id, first\_name,last\_name,dept\_name ve dept\_id değerlerini döndürür.

SELECT employees.emp\_id, employees.first\_name, employees.last\_name, departments.dept\_name, departments.dept\_id FROM employees INNER JOIN departments ON employees.emp\_id = departments.emp\_id;

SELECT employees.first\_name, employees.last\_name, employees.gender, employees.salary, departments.dept\_name FROM employees JOIN departments ON employees.emp\_id = departments.emp\_id

(kadınları görmek istersek → WHERE staff.gender = 'Female';

(L → WHERE staff.first\_name LIKE 'L%' ile başlayan isimleri görmek istersek ne olur;)

(maaşı 80000'den fazla olan çalışanları görmek istersek → NEREde çalışanlar.maaş > 80000;)

Şimdi chinook veri tabanına geliyorum. Burada parçalanmış bir veri tabanı karşımıza çıkıyor.

Genres---türlere bakıyoruz. Türleri track tablosunda id olarak kullanmış.

Media-type

Albums

Fax ayrılmaz ama iletişim bilgileri ayrılabilir

Playlist—playtract, her bir playlist te hangi parçalar var, 1 numaralı playlistte şu parçalar varmış, tracid tutuyor bakin. Gidiyoruz track a oradan yazıyoruz hangi parça olduğunu buluyoruz.

Eğer bu şekilde olmasaydı , şurada kullanılan tüm bilgiler( ayrı tablolarda tutulan) yan yana yazılmış olacaktı. Hepsi tek bir tabloda olacaktı.

Müşteri ismi, adresi, playlistler…… hepsi olacaktı.

O yüzden bizim elimizdeki tablolara verileri, alt tablolara ayırarak veya ilişkisel tablolara ayırarak kullanırız.

Primary key ------ Bunları de her zaman bir anahtar kelime ile yaparız. Her bir müşterinin diğerinden farklı olmasını sağlar. Uniq dir, tekrar etmez.

Foregin key ---- ilişkili olan diğer anahtar, kullanıldığı tabloda uniq olabilir ama başka tablonun forein key i dir. Çünkü orada tekrarlanabilir.

Örnek

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Biz bunun tasarımını yapmayız ama mantığı genel itibariyle anlamak lazım. Primary key ve foregin key in ne anlam ifade ettiğini anlamak lazım.

SLAYTLARA GEÇİYORUZ 3. DERS için

JOIN

JOIN deyimi, iki veya daha fazla tabloyu tek bir tabloda birleştirmek için kullanılır.

Joins genellikle bu tablolar arasındaki ilişkiyi tanımlayan anahtarlara veya ortak alanlara göre uygulanır.

Çoğu durumda bu birleştirmeler, bir tablonun primary keyi ve ona katılmak istediğimiz diğer tablonun foreign keyi kullanılarak oluşturulur.

Anahtarlar üzerinde hareket ederek ilgili veriye ulaşabiliyorum.

JOIN TYPES

INNER JOIN: Her iki tablodaki ortak kayıtları döndürür. ( iki tablom var, bu tabloda sütunlar var, bu tablodaki kesişenleri getiriyor sadece.

LEFT OUTER JOIN: Sol tablodaki tüm kayıtları ve sağ tablodaki eşleşen kayıtları döndürür. ( Birinci tablodaki tüm verileri getiriyor, ikinci tablodaki de kesişenleri getiriyor).

SQLite INNER JOIN ve LEFT JOIN i destekliyor, RIGHT ve FULL OUTER JOIN i desteklemiyor.

RIGHT OUTER JOIN: Sağ tablodaki tüm kayıtları ve sol tablodaki eşleşen kayıtları döndürür.

FULL OUTER JOIN: Hem sol hem de sağ tabloların tüm kayıtlarını döndürür.

En fazla inner join kullanılıyor.

slayt 12

INNER JOIN, yalnızca ve yalnızca her iki tablo da ON yan tümcesinde belirtilen koşulları karşılıyorsa, sonuçta görünmesi için her iki katılımcı tablodaki tüm satırları seçer. Birleştirme koşulu doğru (veya 1) olarak değerlendirilirse, her iki tablodaki satırların sütunları sonuç kümesine dahil edilir.

INNER JOIN'in nasıl çalıştığını kolayca anlamak için ekrandaki görsel anlatımdan faydalanabiliriz. İki tablonun kesişimi, eşleşen satırları temsil eder.

Bu söz diziminde,-------

sütunlar: tablo\_A veya tablo\_B'den sütun adları.

tablo\_A, tablo\_B: Birleştirilen tabloların adları.

birleştirme\_koşulları: Her bir birleştirilmiş satır çifti için değerlendirilecek koşulları belirtir.

Bu ifadedeki işleç genellikle eşittir işaretidir (=), dolayısıyla birleştirme koşulları genellikle şu biçimi alır:

tablo\_A. common\_field = tablo\_B.common\_field

INNER anahtar sözcüğü, bu JOIN türü için isteğe bağlıdır.

Slayttan anlat biraz daha sonra uygulama yap SQLite da.

Bir örnek yapalım. Diyelim ki iki tablomuz var, bunlardan biri mevcut sınavları ve geçme puanlarını listeliyor. Diğeri ise bazı sınavlara giren öğrenciler ve notlarıdır. Sınavları kimin geçip geçmediğini daha iyi anlamak için öğrencilerin notlarını ve sınavların geçme puanını görmek için tabloları birleştirmek istiyorum.

Ortak alanımız sınav. O yüzden bu alandaki tabloları birleştireceğim.

SELECT deyiminde getirmek istediğimiz alanı belirtmek için tablo\_adı.sütun\_adı kullandığımıza lütfen dikkat edin.

Ortak alan sınav olduğu için eşleşen değerler ve bunlara karşılık gelen satırlar birleştirilir.

Gördüğünüz gibi, biri hariç tüm satırlar eşleşiyor; sınav : Network (Çünkü; Network ortak alan değil)

Öğrenciler tablosunda Network olmadığı için onu kaldırdık.

Testlerin sınav sütununu da sorgumuzda istemiyoruz. Yani, sonuç kümemizde olmayacak.

Ek olarak, INNER JOIN yan tümcesi kullanılarak üç veya daha fazla tablo birleştirilebilir. Üç veya daha fazla tabloyu birleştirmek için kullanılan sözdizimi şöyle olabilir

diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

1 öğrencinin bir tane notu olur diyorsam 1 e 1 ilişki oluyor demektir, 1 öğrencinin birden fazla ders alabilir dersem 1 e n ilişki olmuş oluyor.

Biz bunları hazırlamayacağız, bize bu hazır bir şekilde verilir biz bunun üzerinde sorgulama yaparız.

Student tablosundan score yi yazdıralım, kim yazdırmak ister ??????

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Hareketli tablolarda hep rakamlar vardır, biz ne yaparız bunun içindeki rakamları ifade eden bilgileri çekmek istersek bu şekilde sorgulama ifadeleri kullanırız.

Playlist tablosuna bak onların karşılığını bulmaya çalış.

SELECT playlists.Name, tracks.Name

from playlist\_track

inner join playlists on playlist\_track.PlaylistId=playlists.PlaylistId

inner join tracks on playlist\_track.TrackId=tracks.TrackId;

bir tablodaki primary key ile diğer tablodaki foregin key birleşiyor.

Google dan er diyagramlarını ara--------travel agents

Örnek ; chinook.db'deki sanatçı ve albüm tablolarını kullanacağız

Aşağıdaki ifade, albüm başlıklarını ve sanatçı adlarını döndürür.

SELECT Title, Name FROM albums INNER JOIN artists ON artists.ArtistId = albums.ArtistId;

Örnek;

SELECT tracks.Name, albums.Title FROM tracks INNER JOIN albums on tracks.AlbumId=albums.AlbumId;

INNER JOIN ----defaulttur, inner yazmasak da sadece joın yazsak da olur ama biz yine de yazıyoruz.

Başta yazarken hangi tabloda olduğunu belirtmek zorunda değiliz eğer tablodaki sütun ismi diğerlerinde yok ise.

Slayttaki örnekler

* Her bir parçanın türünü bulunuz

SELECT genres.Name, tracks.Name FROM tracks JOIN genres on tracks.GenreId=genres.GenreId;

* Her faturanın müşteri adını bulun. Sonucunuz Fatura kimliğini ve müşteri adını içerecektir.

SELECT invoices.InvoiceId, customers.FirstName FROM invoices JOIN customers on customers.CustomerId=invoices.CustomerId;

Tracks tablosundaki id olan sütunların üçünü de numara olarak görmek istemiyorum da gerçek değerlerini getirmek istiyorum. Ne yapacağım 3 ayrı tablodan ilişki kuracağım. Önce tek tek getirelim daha sonra birleştirelim bunları.

SELECT name, AlbumId, MediaTypeId, GenreId FROM tracks;

SELECT genres.Name, tracks.Name FROM tracks JOIN genres on tracks.GenreId=genres.GenreId;

SELECT albums.Title, tracks.Name FROM tracks JOIN albums on tracks.AlbumId=albums.AlbumId;

SELECT media\_types.Name, tracks.Name FROM tracks JOIN media\_types on tracks.MediaTypeId=media\_types.MediaTypeId;

İnner join ile;

SELECT tracks.Name, media\_types.Name, albums.Title, genres.Name

FROM tracks

INNER JOIN media\_types on tracks.MediaTypeId=media\_types.MediaTypeId

INNER JOIN albums on tracks.AlbumId=albums.AlbumId

INNER JOIN genres on tracks.GenreId=genres.GenreId;

As --- alias ile başlıklara isim verebiliriz

LEFT JOIN

Bu JOIN ifadesinde sorguda sol tablonun tüm kayıtları ve sağ tablonun ortak kayıtları döndürülür. JOIN işlemi sırasında sağdaki tabloda eşleşen satır bulunamazsa bu değerler NULL olarak atanır.

SELECT tests.exam, tests.passing\_score, students.name, students.score FROM tests LEFT JOIN students ON tests.exam = students.exam;

----

Sanatçıların albüm bilgilerini bulun----album isimleri

SELECT artists.name, albums.Title FROM artists LEFT JOIN albums on artists.ArtistId=albums.ArtistId;

Böyle arama yaparsak kısıtlama yapar inner join gibi

SELECT artists.name, albums.Title FROM albums LEFT JOIN artists on artists.ArtistId=albums.ArtistId;

mediaType tablosuna tip olarak mp3 ekle.

Daha sonra LEFT JOIN kullan

SELECT media\_types.Name, tracks.Name FROM media\_types

LEFT JOIN tracks on media\_types.MediaTypeId=tracks.MediaTypeId;

İnner join kullansaydım mp3 ü göremeyecektim.

SELECT media\_types.Name, tracks.Name FROM media\_types JOIN tracks on media\_types.MediaTypeId=tracks.MediaTypeId;

SELECT media\_types.Name, tracks.Name FROM tracks LEFT JOIN media\_types on media\_types.MediaTypeId=tracks.MediaTypeId;

SELECT albums.Title, artists.Name FROM artists LEFT JOIN albums on albums.ArtistId=artists.ArtistId;

Subqueries

Bir alt sorgu, başka bir ifadenin içine yerleştirilmiş bir SELECT ifadesidir. Alt sorgu, iç sorgu veya iç içe sorgu olarak da adlandırılır.

Subqueries, çevreleyen sorguya veri sağlayan iç içe geçmiş sorgulardır.

Alt sorgular, bir değer veya bir kayıt listesi döndürebilir

Alt sorgular parantez içine alınmalıdır

İç sorgu, dış sorgudan önce yürütülür. İç sorgunun sonuçları dış sorguya aktarılır.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Alt sorguyu kullanırken bazı kurallar vardır:

Bir alt sorgu parantez içine alınmalıdır.

ORDER BY deyiminin bir alt sorguda kullanılmasına izin verilmez.

BETWEEN operatörü bir alt sorgu ile kullanılamaz. Ancak alt sorgu içinde BETWEEN kullanabilirsiniz.

Aslında genel bir sözdizimi yoktur; alt sorgular, parantez içine yerleştirilmiş normal sorgulardır.

Alt sorgular, bir sorgu içinde farklı şekillerde ve farklı konumlarda kullanılabilir.

Bir alt sorgu şu durumlarda kullanılabilir:

* SELECT clause
* FROM clause
* WHERE clause

Ancak bir alt sorgu genellikle WHERE yan tümcesinin içine gömülür. WHERE yan tümcelerinde karşılaştırma operatörlerini kullanabilirsiniz.

Şimdi tek satırlık alt sorgularla başlayalım.

S: Alt Sorgu nedir?

C: Alt sorgu, iç içe sorgu veya iç sorgu olarak da bilinen başka bir sorgu içindeki sorgudur. Ana sorgu tarafından sorgulanacak verileri kısıtlamak veya geliştirmek, böylece ana sorgunun çıktısını sırasıyla kısıtlamak veya geliştirmek için kullanılır.

İki ana alt sorgu türü vardır:

* Single-row subqueries
* Multiple-row subqueries

Tek satırlı alt sorgular, yalnızca bir sütun içeren bir satır döndürür ve tipik olarak, özellikle WHERE yan tümcesinde =, >, >=, <=, <>, != gibi tek satır işleçleriyle kullanılır.

Bazı koşulları içerdiğinden çoğunlukla “WHERE” yan tümceleriyle birlikte kullanılırlar karşılaştırma işleçleri

Rodney Weaver'dan daha fazla maaş alan çalışanları bulacağız. Sorgumuz çalışanların ad, soyad ve maaş bilgilerini döndürmelidir.

ayrıntılı olarak görelim

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Önce iç sorgu yürütülür ve Rodney'nin maaşı olan 87000'i döndürür.

87000 değeri dış sorguya, özellikle WHERE yan tümcesine iletilir.

Bir diğer örnek;

**Find out the employees who get paid more than the average salary**

Ortalama maaştan daha fazla maaş alan çalışanları öğrenin

Tek bir sütunla tek bir satır döndürdüğü için kümelenmiş işlevleri de kullanabiliriz. Şimdi ortalama maaştan daha fazla maaş alan çalışanları bulalım. Sorgumuz çalışanların ad, soyad ve maaş bilgilerini döndürmelidir.

SELECT first\_name, last\_name, salary FROM employees

WHERE salary > (SELECT AVG(salary) FROM employees);

Sorguyu inceleyelim;:

WHERE yan tümcesi ile > operatörünü kullandık.

İç sorgu 83600 olan ortalama maaşı döndürür ve bu tek değeri dış sorguya iletir.

Dış sorgu, bu çalışanları filtreler ve maaşı 83600'den fazla olanları döndürür.

Toplam işlevleri alt sorgularla birleştirmenin, analitik sorularımızı yanıtlamak için daha fazla gücümüz olduğunu gördüğünüz gibi.

Kendimiz employees tablosundan örnekler yapalım

Rubie nin aldığı maaş

SELECT salary FROM carsalesEmployees WHERE name='Rubie';

Rubie den fazla maas alanlar

SELECT \* FROM carsalesEmployees WHERE salary>(SELECT salary FROM carsalesEmployees WHERE name='Rubie’);

\*\*

Slayttaki Örnekler----arkadaşlara yaptıralım

Parça kimliği, parça adı, ' Faceless ' Albüm başlığının albüm kimliği bilgilerini alın.

Retrieve track id, track name, album id info of the Album title ‘Faceless’. (fesılıs)

SELECT trackid, name, albumid FROM tracks

WHERE albumid = ( SELECT albumid FROM albums WHERE title = 'Faceless');

Burada SELECT albumid FROM albums WHERE title = 'Faceless'

sorgusunu ayrı çalıştır ve tek bir row döndürdüğünü göster.

\*\*Bunu yapmamızın sebebi iki tablo arasında ilişki kurarak yapmalıyız, onun için albümid ile ilişki kuruyoruz.

Birleştirme kullanan sorguların çoğu, bir alt sorgu (başka bir sorgunun içine yerleştirilmiş bir sorgu) kullanılarak yeniden yazılabilir ve çoğu alt sorgu, birleştirmeler olarak yeniden yazılabilir.

Join kullanarak yapalım

Parça kimliği, parça adı, ' Faceless ' Albüm başlığının albüm kimliği bilgilerini alın.

**SELECT** tracks.TrackId, tracks.Name, tracks.AlbumId, albums.Title

**FROM** tracks

**JOIN** albums **ON** tracks.AlbumId = albums.AlbumId **WHERE** albums.Title = 'Faceless';

Multiple-row Subqueries

Çok satırlı alt sorgular, satır kümelerini döndürür ve IN, NOT IN, ANY, ALL gibi çok satırlı işleçlerle kullanılır.

Örnek;

Operasyon departmanı (departmanlar tablosu) altında çalışan çalışanları (çalışanlar tablosundan ad, soyad) bulun

Bildiğiniz gibi tek tablodan oluşan şirket veri tabanına yeni bir tablo ekledik. Yeni tablomuz departmanlar tablosudur. Üç sütunu ve dokuz satırı vardır. Her satır, bir çalışanın departman bilgilerini temsil eder. İşte veritabanı tablolarımız.

ayrıntılı olarak görelim

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Ekrandaki sorguyu inceleyelim:

IN operatörünü WHERE yan tümcesi ile kullandık.

İç sorgu, Operasyon departmanı altında çalışan çalışan kimliklerini döndürür.

Daha sonra çalışan kimlikleri dış sorguya iletilir.

1. İç sorgu, Operasyon departmanı altında çalışan çalışanların kimliklerini döndürür.
2. Çalışan kimlikleri dış sorguya iletilir.

Dış sorgu, bu çalışanların kimliklerini filtreler ve sonuç kümesi olarak adlarını ve soyadlarını döndürür.

Operasyon departmanında kaç erkek çalışan var?

(Not: Sonuç tablonuz 'count\_male' adlı bir sütundan oluşacaktır)

SELECT COUNT(gender) as count\_male FROM employees WHERE gender='Male' AND emp\_id IN(select emp\_id from departments where dept\_name='Operations');

Join ile

SELECT COUNT(gender) as count\_male FROM employees

**JOIN** departments ON employees.emp\_id=departments.emp\_id

WHERE employees.gender = 'Male' AND employees.emp\_id

Slayttaki son örnek;

Retrieve track id, track name, album id info of the Album title ‘Faceless’ and ‘Let There Be Rock’

"Faceless" ve "Let There Be Rock" Albüm başlığının parça kimliğini, parça adını, albüm kimliği bilgilerini alın

SELECT trackid, name, albumid FROM tracks WHERE albumid

IN ( SELECT AlbumId FROM albums WHERE title IN ('Faceless', 'Let There Be Rock') );

-----------------------------burada kaldık 3. Ders sonu

**JOIN kullanarak yapınız**

SELECT tracks.trackid, tracks.name, tracks.albumid FROM tracks

**JOIN** albums ON tracks.AlbumId=albums.AlbumId WHERE title IN ('Faceless', 'Let There Be Rock');

* Veya bunları içermeyen diyebiliriz.
* Veya albüm ismi s ile başlayanlar

Tüm artistler albüm çıkartmış mı bakalım

* Hangi artist hangi albümü çıkarttı

SELECT artists.Name as artist\_adi, albums.Title as album\_adi from artists

INNER JOIN albums on artists.ArtistId=albums.ArtistId ORDER by albums.ArtistId;

Önemli bilgi;

SELECT parcalar.Name from tracks as parcalar;

Ben burada tablo ismimi değiştirebilirim.

* Albümü olmayan artistler kim ??????

SELECT artists.Name, albums.Title from artists LEFT JOIN albums on artists.ArtistId=albums.ArtistId

WHERE albums.Title is NULL;

* Hangi artisin kaç tane albümü var

SELECT artists.ArtistId, artists.Name, count(\*) from artists

INNER JOIN albums on artists.ArtistId=albums.ArtistId GROUP by artists.ArtistId;

* Kaç artistin albümü yok

SELECT count(artists.Name) from artists LEFT JOIN albums on artists.ArtistId=albums.ArtistId

WHERE albums.Title is NULL;

\*\*\*\*\*\*

HAVING

Group by ile where condisinini beraber kullanamayız, bunun yerine HAVİNG kullanmalıyız. Eğer gurupladıktan sonra bir işlem yapmak istiyorsak bunu kullanmalıyız.

Guruplamadan önce bir şart istiyorsam where kullanacağız.

Her bir composer in (besteci) kaç tane parçası var ??????2

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

--

SELECT Composer, count(\*) FROM tracks GROUP by Composer HAVING count(\*)>9;

SELECT Composer, count(\*) FROM tracks GROUP by Composer HAVING count(\*)>9 AND Composer like ('A%');

---

Belli bir tarihten sonraki faturaları getirip guruplayabilirim ---- istediğim sütuna göre

SELECT \* FROM invoices WHERE InvoiceDate>'2011-01-01';

SELECT \* FROM invoices WHERE InvoiceDate>'2011-01-01' GROUP by BillingCountry;

2011 den sonra fatura toplamı 50 den büyük olanlar

SELECT \*, sum(total) as top FROM invoices WHERE InvoiceDate>'2011-01-01' GROUP by BillingCountry HAVING top>50;

SELECT BillingCountry, count(\*), sum(total) as top FROM invoices WHERE InvoiceDate>'2011-01-01' GROUP by BillingCountry

HAVING top>50;

Alt sorgularla ilgili önemli bilgi;

Alt sorgulara başlamadan önce, sorguların nasıl yürütüldüğünü bilmelisiniz.

Yazdığınızdan farklı:

FROM ve JOIN ifadelerinden başlar

sonra onu WHERE yan tümcesinde filtreler

ardından GROUP BY ve HAVING

ardından buna göre sütun adlarını SEÇİN bu noktada gerekli sütun ve satırlara sahibiz

daha sonra ORDER BY deyimindeki verileri sıralar

son olarak sonuçları buna göre sınırlayın LIMIT

örnek çalışılabilecek bir site (chinook veri tabanı için)

<https://www.sqlitetutorial.net/sqlite-count-function/>