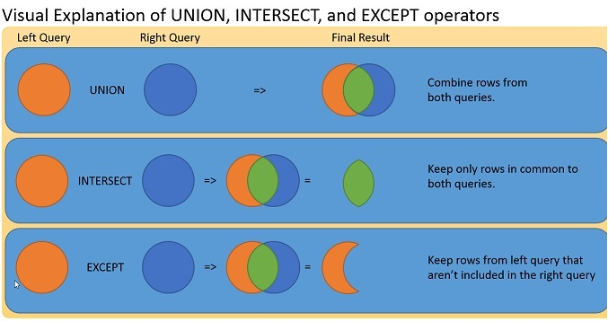
Küme işlemleri, birden çok sorgunun sonuçlarının tek bir sonuç kümesinde birleştirilmesine olanak tanır. Set operatörleri UNION, INTERSECT ve EXCEPT'i içerir.

UNION set operatörü, iki SELECT ifadesinin birleştirilmiş sonuçlarını döndürür. Esasen, sonuçlardan yinelenenleri kaldırır, yani yinelenen her sonuç için yalnızca bir satır listelenir.

Bu davranışa karşı koymak için, nihai sonuçta kopyaları tutan UNION ALL set operatörünü kullanın.

INTERSECT, yalnızca her iki SELECT sorgusu için ortak olan kayıtları listeler; EXCEPT set operatörü, ilk sorgunun sonuçlarında da bulunurlarsa, ikinci sorgunun sonuçlarını çıktıdan kaldırır. INTERSECT ve EXCEPT küme işlemleri yinelenmemiş sonuçlar üretir.



☝ Önemli:

Her iki SELECT deyimi de aynı sayıda sütun içermelidir.

SELECT ifadelerinde, karşılık gelen sütunlar aynı veri tipine sahip olmalıdır.

Sonuç kümesini sıralamak için konumsal sıralama kullanılmalıdır. Set operatörlerinde bireysel sonuç seti sıralamasına izin verilmez. ORDER BY, sorgunun sonunda bir kez görünebilir.

UNION ve INTERSECT operatörleri değişmelidir, yani sorguların sırası önemli değildir; nihai sonucu değiştirmez.

Performans açısından, UNION ALL, UNION'a kıyasla daha iyi performans gösterir, çünkü kaynaklar kopyaları filtrelemek ve sonuç kümesini sıralamak için boşa harcanmaz.

Set operatörleri alt sorguların parçası olabilir.

**Union Operatörü**

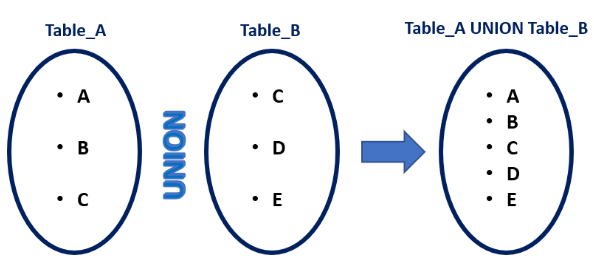
Bazı durumlarda, iki veya daha fazla tablodaki verileri bir sonuç kümesinde birleştirmeniz gerekebilir. Bu işlemi gerçekleştirmek için Union deyimi kullanılır. Birleştirmeniz gereken tablolar, aynı veri tabanında veya farklı veritabanlarında benzer veriler içeren tablolar olabilir.

UNION operatörünü, iki veya daha fazla sorgudan gelen satırları tek bir sonuç kümesinde birleştirmek için kullanacaksınız. UNION operatörü için temel sözdizimi şöyledir:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

UNION işlemini aşağıdaki resmi inceleyerek daha kolay anlayabilirsiniz.



"employees\_A" ve "employees\_B" adında iki tablomuz olduğunu varsayalım. İki çalışan tablosunu tek bir tabloda birleştirmek için UNION operatörünü kullanacağız:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

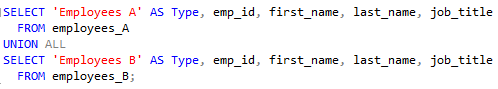
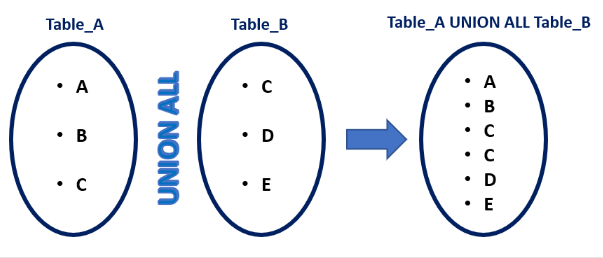
Çalışanlar\_A tablosundaki altı satır ve çalışanlar\_B tablosundaki altı satır UNION operatörü kullanılarak birleştirilir ve on satırlık bir tablo elde edilir. Ancak, bazı kayıtlar yinelenen kayıtlar oldukları için çıktı tablosunda tekrarlanmaz.

Union All Operator

UNION ALL yan tümcesi, iki tabloyu birleştirirken yinelenen kayıtlar dahil tüm kayıtları yazdırmak için kullanılır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

tablo içeren bir resim

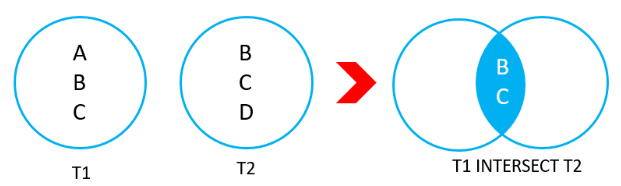
Açıklama otomatik olarak oluşturuldu

Çalışanlar\_A tablosundaki altı satır ve çalışanlar\_B tablosundaki altı satır, UNION ALL operatörü kullanılarak birleştirilir ve on iki satırlık bir tablo elde edilir. Ancak, emp\_id 49714 ve emp\_id 67323'ün çalışan kayıtları mükerrer kayıtlardır.

**Intersect Operator**

INTERSECT operatörü, iki sorgunun sonuç kümelerini karşılaştırır ve her iki sorgunun çıktısı olan farklı satırları döndürür.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldumetin içeren bir resim

Açıklama otomatik olarak oluşturuldu

tablo içeren bir resim

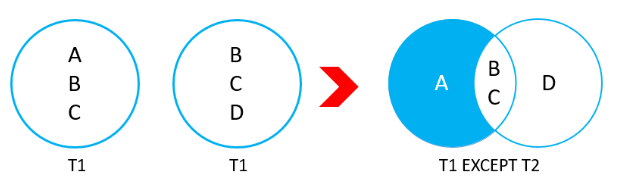
Açıklama otomatik olarak oluşturuldu

Gördüğünüz gibi sonuç setinde sadece iki çalışanın her iki tablo için ortak olan bilgileri döndürüldü.

**Except Operator**

EXCEPT operatörü, iki sorgunun sonuç kümelerini karşılaştırır ve önceki sorgunun sonraki sorgudan farklı olan satırlarını döndürür.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldumetin içeren bir resim

Açıklama otomatik olarak oluşturuldu

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Görüldüğü gibi sonuç kümesinde sadece çalışanlar\_A tablosunda olan ancak çalışanlar\_B tablosunda olmayan çalışanlar döndürüldü.

**CASE Expression**

CASE ifadesi, bir koşullar listesini değerlendirir ve ilk koşul karşılandığında bir değer döndürür. CASE ifadesi, diğer programlama dillerindeki IF-THEN-ELSE ifadesine benzer. CASE ifadesi, SQL'in if/then mantığını işleme şeklidir. Her CASE ifadesi END anahtar sözcüğüyle bitmelidir.

ELSE kısmı isteğe bağlıdır. ELSE kısmı yoksa ve hiçbir koşul doğru değilse, NULL döndürür. İki tür CASE ifadesi vardır:

Basit CASE ifadesi:

Basit CASE ifadesi, sonucu belirlemek için bir ifadeyi bir dizi basit ifadeyle karşılaştırır.

Aranan CASE ifadesi:

Aranan CASE ifadesi, sonucu belirlemek için bir dizi Boole ifadesini değerlendirir.

CASE, herhangi bir deyimde veya tümcede kullanılabilir. Örneğin, SELECT, UPDATE, DELETE ve SET gibi deyimlerde ve IN, WHERE, ORDER BY ve HAVING gibi tümcelerde CASE kullanabilirsiniz.

**Simple CASE Expression**

Simple CASE Expression, sonucu döndürmek için bir ifadeyi bir dizi ifadeyle karşılaştırır. İşte basit CASE ifadesi sözdizimi:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Basit CASE ifadesi, case\_expression ifadesini WHEN yan tümcelerindeki ifadelerle karşılaştırır. Ardından, birden çok olası sonuç ifadesinden birini döndürür. Hiçbir vaka ifadesi, while ifadesiyle eşleşmezse, CASE ifadesi, else\_result\_expression'ı döndürür. ELSE bölümü dahil edilmemişse, CASE ifadesi NULL değerini döndürür.

ELSE isteğe bağlıdır. Köşeli parantezler arasında gösterilmesinin nedeni budur. Köşeli parantez "isteğe bağlı" anlamına gelir. Yani CASE ifadelerimize ELSE kısmını dahil etmek zorunda değiliz.

Bir örneğe bakalım. Bilgi Teknolojileri (BT) alanı ile ilgili olup olmadıklarını bölümlere ayıracağız. Herhangi bir departman bu kategoriye girerse, onu 'BT' olarak, aksi takdirde 'diğerleri' olarak etiketleyin.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

SELECT ifadesinde CASE ifadesini kullandık. Gördüğünüz gibi 'category' adında yeni bir sütun oluşturuldu. AS anahtar sözcüğünü kullanarak END anahtar sözcüğünden sonra adlandırdık.

Yukarıdaki sorguda, dept\_name sütununun altındaki değerler WHEN yan tümcesindeki ('Bilgisayar Bilimi') ifadesiyle karşılaştırılır. Herhangi bir eşleşme varsa, 'IT' sonucu döner. Eşleşmeyen değerler 'diğerleri' olarak döner. Döndürülen tüm 'IT' veya 'others' değerleri, END AS'den sonra belirtilen farklı bir sütuna yerleştirilir. Bizim durumumuzda, yeni oluşturulan sütun adı 'kategori'dir. Oluşturulan yeni kolona istediğiniz ismi verebilirsiniz.

CASE ifadesi WHERE, HAVING, ORDER BY, SELECT yan tümcelerinde kullanılabilir.

**Searched CASE Expression**

Aranan CASE ifadesi, sonucu belirlemek için bir dizi ifadeyi değerlendirir. Basit CASE ifadesinde, yalnızca denklik için karşılaştırılırken, aranan CASE ifadesi herhangi bir karşılaştırma türünü içerebilir. Bu tip CASE deyiminde, CASE anahtar sözcüğünden hemen sonra herhangi bir ifade belirtmeyiz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bir örneğe bakalım. Çalışanların maaşlarını Yüksek, Orta, Düşük olmak üzere üç kategoriye ayıracağız.

İşte kriterlerimiz:

Maaş < 55.000 $ SONRA 'Düşük' olduğunda

Maaş 55.000 $ ile 80.000 $ arasında olduğunda SONRA 'Orta'

Maaş > 80.000 $ SONRA 'Yüksek' olduğunda

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

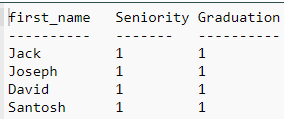
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

CASE ifadesinin toplama işlevleriyle kullanılması çoğu zaman sizi uzun sorgulardan kurtarır. İşte CASE Expression ile başka bir örnek:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu



Yukarıdaki sorguda Lisans mezunu olan deneyimli çalışanları listeledik.

Bu sorguda, SUM() işlevi ve CASE İfadesi kullanılarak yapılan filtrelemeden kaynaklanan sütunlar SELECT deyimine yazılmayabilir. Sonucu görmenizi istediğimiz için buraya yazdık.

Bir toplama işleminin sonucunu filtrelemek istedik. Yani, bunu HAVING deyiminde yaptık.