**UNION & UNION ALL** deyimi iki veya daha fazla sorgunun sonuçlarını birleştirmek için kullanırız. Bu iki veya daha fazla sorgu sonucu oluşan değerler tek bir tablo üzerinde return edilir.

UNION & UNION ALL deyimi tamamen iki benzer tablonun ilgili kolonlarını birleştirmek için kullanılır.

UNION & UNION ALL kullanırken dikkat! : listelenecek kolonların aynı türde, birbiri ile uygun türde veri tiplerine sahip olması gerekiyor.

**INTERSECT**: UNION gibi iki SELECT ifadesini birleştirmek için kullanıyoruz, ancak INTERSECT ifadesinin döndürdüğü veri kümesi, iki SELECT ifadesinin veri kümelerinin kesişimi olacaktır.

Yani INTERSECT deyimi yalnızca her iki SELECT deyiminde **ortak** satırları döndürür.

**EXCEPT:** iki farklı sql sorgusundan birinin sonuç kümesinde olup diğerinin sonuç kümesinde olmayan ( A tablosunda olup da B tablosunda olmayan) kayıtları listeleyen bir komuttur.

**CASE** fonksiyonu: herhangi bir şartı sağlayan veriyi istediğimiz şekilde gösterebilmemize olanak sağlamaktadır. İki şekilde kullanılır: **Simple Case** ve **Searched case**.

**Simple Case :**Mevcut bir veri satırında yer alan değerin daha sonra istediğimiz şekilde görüntülenmesini sağlıyoruz.

Syntax:

**CASE** test ifadesi  
**WHEN** karşılaştırılacak ifade 1 **THEN** Bunun Geri Dönüş Değeri  
**WHEN** karşılaştırılacak ifade 2 **THEN** Bunun Geri Dönüş Değeri  
**ELSE** karşılaştırılacak son değer **END AS** sütun başlığı.

Örnek 1:

SELECT Title, CASE Title

WHEN 'Mr.' THEN 'Bay'

WHEN 'Ms.' THEN 'Bayan'

WHEN 'Mrs.' THEN 'Bayan'

WHEN 'Miss' THEN 'Bayan'

ELSE 'Bilinmiyor' END AS Cinsiyet

FROM Person.Person

WHERE BusinessEntityID IN (1,3,5,6,345,357,423)

Örnek 2 :

**Create a new column with the meaning of the values in the Order\_Status column. 1 = Pending; 2 = Processing; 3 = Rejected; 4 = Completed**

SELECT order\_id, order\_status,  
 CASE order\_status  
 WHEN 1 THEN 'Pending'  
 WHEN 2 THEN 'Processing'  
 WHEN 3 THEN 'Rejected'  
 ELSE 'Completed'  
 END AS ord\_status\_mean  
FROM sale.orders

**Searched Case :**  Eğer ifadeler oldukça karışık ise başvurulan bir yöntemdir. Sadece tek bir sütuna değil, birden fazla sütuna condition ugulamamıza izin veriyor. Ayrıca sadece eşittir değil, diğer logical operatörlerle de çalışıyor.

Syntax:

**CASE WHEN** test edilecek ifade1 **THEN** değer1  
**WHEN** test edilecek ifade 2 **THEN** değer 2  
**ELSE** değer3 **END**

Örnek 1:

SELECT Title,

**CASE WHEN** Title IN ('Ms.','Mrs.','Miss') **THEN** 'Bayan'

**WHEN** Title = 'Mr.' **THEN** 'Erkek'

**ELSE** 'Bilinmiyor'

**END** **AS** Cinsiyet

FROM Person.Person

WHERE BusinessEntityID IN (1,3,5,6,345,357,423)

Örnek 2:

Create a new column that shows which email service provider

("Gmail", "Hotmail", "Yahoo" or "Other" ).

Create  a new column with the meaning of the  values in the Order\_Status column.  
-- 1 = Pending; 2 = Processing; 3 = Rejected; 4 = Completed

SELECT order\_id, order\_status,  
 CASE  
 WHEN order\_status = 1 THEN 'Pending'  
 WHEN order\_status = 2 THEN 'Processing'  
 WHEN order\_status = 3 THEN 'Rejected'  
 ELSE 'Completed'  
 END AS ord\_status\_mean  
FROM sale.orders

**CASE ile karşılaştırma yapmak :** CASE deyimi ile karşılaştırma da yapabiliriz.

Örneğin seyehat bilgilerinin yer aldığı bir tablomuz olsun. Seyahat bilgilerinden bir sütunda otelde kalma süresi diğer bir sütunda yolda geçecek süre olsun. Bu iki sütundan hangisi büyük veya hangisi küçük olduğunu seçip ayrı bir kolonda tıpkı yeni bir sütun gibi gösterebiliriz.

SELECT VacationHours, SickLeaveHours,

**CASE**

**WHEN** VacationHours > SickLeaveHours THEN VacationHours

**ELSE** SickLeaveHours

**END AS** 'Büyük Değer'

FROM HumanResources.Employee

**WINDOW FUNCTION:**

Veri setinde mevcut satırla bir şekilde ilişkili olan bir dizi satırda bir işlem gerçekleştirmemizi sağlar.

Window fonksiyonları sql sorgusu ile elde edilen sonuç setini her fonksiyonun kendi karakterine göre parçalara ayırarak yine bu parçalara kendi fonksiyonlarına göre değer üretirler.

Bu değerler SELECT listesinde veya ORDER BY sıralama kriterleri içinde kullanılabilirler.

Ancak bu fonksiyonlar (window functions) WHERE cümleciği içinde kullanılamazlar. Çünkü window fonksiyonlarının uygulanacağı kayıt setini SELECT sorgusu üzerindeki WHERE bölümündeki koşullar belirler.

Group by fonksiyonundan farklı olarak diğer satırlardaki verileri de hesaplamaya dahil edebiliriz.

Hareketli ortalama-kümülatif toplam gibi işlemleri bu fonksiyonlarla grup bazında kolayca yapabiliriz.

Window fonksiyonları kullanılırken **OVER** anahtarı ile kayıt setinin parçalara bölünmesi sağlanır.

GROUP BY İLE KIYASLAMASI:

GROUP BY--> distinct kullanmıyoruz, distinct'i zaten kendi içinde yapıyor  
Window Funct.--> optioanal olarak yapabiliyoruz.

GROUP BY -->  aggregate mutlaka gerekli,  
Window Funct.> aggregate optional

GROUP BY --> Ordering invalid  
Window Funct.--> ordering valid

GROUP BY --> performansı düşük  
Window Funct.--> performanslı

**3 farklı bileşen kullanılır;**

* **PARTITION BY** : Veriyi gruplara ayırır. Opsiyoneldir.
* **ORDER BY :** Her bir grup için satırların sıralamasını yapmayı sağlar. Zorunludur.
* **ROW\_veya\_RANGE :** Veri gruplarının tüm verisi ile değil belirli bir alandaki verileri için hesaplama yapabilmemizi sağlar. Özellikle hareketli ortalama hesaplayabilmek için çok kullanışlıdır. Opsiyoneldir.

1. **Unbounded Preceding** (öncesi sınırlandırılmamış):  current row'a kadar olan bölümde fonksiyonu uygula.
2. **Unbounded Following** (sonrası sınırlandırılmamış) : current row'dan sonuna kadar olan bölümde fonksiyonu uygula.

**(**Hepsi Current Row'a kadar olan bölümü grupluyor.)

**İki tip WINDOW fonksiyonu tanımlanmıştır.**  
  
 **Ranking Window Functions**

**Aggregate Window Functions**

**Ranking Window Functions**

kayıt seti içinde derecelendirme fonksiyonlarıdır. Bunlar:

**ROW\_NUMBER**

kayıt setinin herbir bölümüne atamak üzere 1 'den başlayarak sıralı giden bir sayı üretir.

**RANK**  
**DENSE\_RANK**

RANK fonksiyonu ile bölümlendirilmiş kayıt setindeki her bir bölüme bir rank numarası verilir. Bu kayıt setinde alt bölüm içindeki her kayıt aynı rank numarasına sahiptir. Bu numaralandırma yine 1'den başlar. Ancak ROW\_NUMBER fonksiyonunda olduğu gibi ardarda gelen bir sıralandırma ile gitmek zorunda değildir. Bir bölüm içinde birden fazla kayıt varsa rank numaralarında atlamalar olacaktır.

**NTILE**

NTILE fonksiyonu SELECT ifadenizde WHERE koşuluna uyan kayıtları OVER ve ORDER BY ile belirtilen sıralamaya göre dizilmiş şekilde sizin parametre olarak geçeceğiniz bir sayıya bölerek her bölüme bir sıra numarası verir.