SQL Server Ranking Fonksiyonları

Ranking fonksiyonları sıralama işlevinde satırlara sıra numarası vermek için kullanılan fonksiyonlardır. Kullanılan fonksiyona göre bazı sıra numaraları aynı olabilir.

ROW_NUMBER

Sorgu sonucunu numaralandırır. Kullanılan yönteme göre bütün sorgu sonucunu tek grup olarak algılayıp numaralandırabileceği gibi her grubu kendi içinde tekrar birden başlayarak numaralandırabilir.

Kullanım Şekli

```
ROW_NUMBER ( )
     OVER ( [ PARTITION BY value_expression , ... [ n ] ] order_by_clause )
```

Örnekler

Basit Örnek

Aşağıdaki kod sistem veri tabanlarını listeler. “ROW_NUMBER() OVER(ORDER BY name ASC) AS Row#” komut parçası veri tabanı isimlerini küçükten büyüğe sıralayarak numarasını verir.

```
SELECT
  ROW_NUMBER() OVER(ORDER BY name ASC) AS Row#,
  name, recovery_model_desc
FROM sys.databases
WHERE database_id < 5;
```

Row#	name	recovery_model_desc
1	master	SIMPLE
2	model	FULL
3	msdb	SIMPLE
4	tempdb	SIMPLE

“[PARTITION BY value_expression , … [n]]” kısmı kullanıldığında value-expression değerlerine göre sonucu gruplara ayırıp her grubu kendi içerisinde birden başlayarak numaralandırır. Gruplama için seçilen değer değiştiğinde sayaç tekrar bire alınır. Örneğin aşağıdaki sorguda işlemi “recovery_model_desc” kolonuna göre gruplama işlemi yapılmış.

```
SELECT
  ROW_NUMBER() OVER(PARTITION BY recovery_model_desc ORDER BY
name ASC)
    AS Row#,
  name, recovery_model_desc
FROM sys.databases WHERE database_id < 5;
```

Row#	name	recovery_model_desc
1	model	FULL
1	master	SIMPLE
2	msdb	SIMPLE
3	tempdb	SIMPLE

Sonuçta da gördüğümüz gibi recovery_model_desc kolonu “FULL” değeri için 1 verilmişken, “SIMPLE” değeri de kendi içinde numaralandırılmıştır.

Satış Temsilcilerinin Satış Miktarlarının Alınması

Başka bir örnek olarak aşağıdaki sorgu, satış temsilcilerinin yıla göre satış miktarını sorguluyor. Sorguyu yaparken de “ROW_NUMBER() OVER(ORDER BY SalesYTD DESC)” kısmı ile satış miktarlarını azalan sırada sıralayıp numaralandırmaktadır.

```
USE AdventureWorks2012;
GO
SELECT ROW_NUMBER() OVER(ORDER BY SalesYTD DESC) AS Row,
    FirstName, LastName, ROUND(SalesYTD,2,1) AS "Sales YTD"
FROM Sales.vSalesPerson
WHERE TerritoryName IS NOT NULL AND SalesYTD <> 0;
```


ROW_NUMBER

ROW_NUMBER’IN PARTITION İLE KULLANILMASI

Yukarıdaki sorguyu bu sefer satış temsilcilerini bölgelerine göre gruplayıp kendi grupları içerisinde sıralamak için değiştirirsek;

```
USE AdventureWorks2012;
GO
SELECT FirstName, LastName, TerritoryName, ROUND(SalesYTD,2,1)
AS SalesYTD,
```

```
ROW_NUMBER() OVER(PARTITION BY TerritoryName ORDER BY SalesYTD
DESC)
  AS Row
FROM Sales.vSalesPerson
WHERE TerritoryName IS NOT NULL AND SalesYTD <> 0
ORDER BY TerritoryName;
```

ROW_NUMBER PARTITION

Sonuç ekranında da göründüğü gibi bölge adı (TerritoryName) değiştikçe sayaç bire eşitleniyor.

RANK

Çalışması ve kullanımı ROW_NUMBER gibidir. Farklı olarak ROW_NUMBER her bir satırı artan sırada numaralandırırken (1,2,3,4,5) RANK eşit değere sahip satırları aynı numara ile numaralandırmaktadır (1,2,2,3,4) dolayısı ile aynı değerleri dönderebilir.

Kullanım Şekli

RANK ( ) OVER ([partition_by_clause] order_by_clause)

Eğer bir değer tekrar ederse sonra ki değer tekrar edenlerin sayısı toplamı kadar olacaktır. Örneğin 1, 2, 2, 2, 4, 5 vb. İki değerinden sonra 4 değerinin gelmesinin nedeni 4 değerinin gerçekten dördüncü sırada olmasıdır ve kendisinden önce bir tane birinci ve üç tana ikinci eleman bulunmaktadır. Bu

özelliğinin sonucu olarak RANK fonksiyonu ardışık numaralar döndürmeyebilir.

Aşağıdaki örnek, maaşlarına göre sıralanmış ilk on çalışanı döndürmektedir. PARTITION BY deyimi belirtilmediğinden, RANK işlevi sonuç kümesindeki tüm satırlara uygulandı.

```
USE AdventureWorks2012
SELECT TOP(10) BusinessEntityID, Rate,
       RANK() OVER (ORDER BY Rate DESC) AS RankBySalary
FROM HumanResources.EmployeePayHistory AS eph1
WHERE RateChangeDate = (SELECT MAX(RateChangeDate)
                        FROM HumanResources.EmployeePayHistory
AS eph2
                           WHERE eph1.BusinessEntityID =
eph2.BusinessEntityID)
ORDER BY BusinessEntityID;
```

RANK

Aşağıdaki örnek sorguda da ürünleri stok durumlarına ve konumlarına göre sıralar. Ürünleri LocationID değerine göre gruplayıp Quantity değerine göre azalan sırada listeler. Listenin ilk iki ürünü olan 494 ve 495 numaralı ürünlerin aynı RANK değerine sahip olduğuna dikkat edin. Sonra gelen değer tekrar eden iki 1'den dolayı üç olmuştur. Bir diğer nokta PARTITION BY ile kullanılan LocationID değeri değiştiğinde sayacın bire eşitlendiğine dikkat edin.

```
USE AdventureWorks2012;
GO
SELECT i.ProductID, p.Name, i.LocationID, i.Quantity
    ,RANK() OVER
```

```
    (PARTITION BY i.LocationID ORDER BY i.Quantity DESC) AS
Rank
FROM Production.ProductInventory AS i
INNER JOIN Production.Product AS p
    ON i.ProductID = p.ProductID
WHERE i.LocationID BETWEEN 3 AND 4
ORDER BY i.LocationID;
GO
```

RANK PARTITION

DENSE_RANK

Kullanımı RANK ile birebir aynıdır. RANK fonksiyonundan farkı, RANK fonksiyonunda tekrar eden değerden sonra kayıtların sayısı kadar olan değer geliyorken DENSE_RANK fonksiyonunda tekrar eden değerlerden sonra ardışık değer gelmektedir. Yukarıdaki sorgunun DENSE_RANK ile yazılmış hali:

```
USE AdventureWorks2012;
GO
SELECT i.ProductID, p.Name, i.LocationID, i.Quantity
    ,DENSE_RANK() OVER
    (PARTITION BY i.LocationID ORDER BY i.Quantity DESC) AS
Rank
FROM Production.ProductInventory AS i
INNER JOIN Production.Product AS p
    ON i.ProductID = p.ProductID
WHERE i.LocationID BETWEEN 3 AND 4
ORDER BY i.LocationID;
GO
```

DENSE_RANK