LOVE LETTER

Analitik SQL Nedir ?

SQL bir “DataBase” yönetim yazılımıdır. Aslında donanımlar seviyesine inecek olursa donanımı önemli kılan “veri” denen ifadeyi Programlarımızla buluşturmaya ve buna göre işlemeye yarayan yazılımdır. Analitik SQL’de aslında Bilgisayar Bilimleri yıllarca geliştikçe veri miktarı artmıştır. Veriye olan açlık doyum noktasına gelmiş ve verilerin filtrelenmesi verinin faydalı bilgi haline alması için bir ihtiyaç doğmuştur. Analitik SQL burada ortaya çıkmış ve 2012’de hayatımıza girmiştir.

Analitik SQL Nerelerde Kullanılır ?

Sirketler, analitik SQL sorgulari kullanarak işletme performansini degerlendirebilir, trendleri analiz edebilir ve karar destek sistemleri için veri sağlayabilir.

Örneğin , bir e ticaret şirketi , satış verilerini analiz ederek en çok satılan ürünleri belirleyebilir, pazarlama analitiği yapabilir.

Bir banka, müşteri hareketlerini analiz ederek kredi
riskini degerlendirebilir ve müşteri terkini önlemek için önlemler alınabilir.

Bir şirket gelir tablosu ve tablosunu analiz ederek gelir ve gider trendlerini çalıştırır, mali rasyoları hesaplayabilir ve bütçe tahminleri yapabilir.

Bir lojistik şirketi, nakliye hizmetlerini analiz ederek rotalama optimizasyonunu yapabilir, envanter yönetimini optimize edebilir ve teslimat yolunu iyileştirebilir.

Bir sigorta şirketi, polis teşkilatını analiz ederek potansiyel dolandırıcılık vakalarını işletmer ve müşteri profillerini tanımlayabilir.

Büyük veri kümelerindeki desenleri, ilişkileri ve trendleri kullanmak için analitik SQL sorguları kullanılabilir.

Analitik SQL, karmaşık koruyucular ve kullandıkları modelleri kullanmak için kullanılabilir. Örneğin, tahminleme, optimizasyon, simülasyon ve mekanizma analizi gibi işlemleri gerçekleştirebilir.

Hadoop (Açık kaynak kodlu bir kütüphane) gibi büyük veri platformları üzerinde çalışan veri analitiği için analitik SQL kullanılabilir.

Sorgu Örnekleri (her grup üyesine 1 sorgu örneği olacak şekilde)

OVER()
Over fonksiyonu daha çok ROW_NUMBER ORDER BY fonksiyonları gibileriyle birlikte kullanılır. Belirli bir veri grubunu işaretlemek için eşik değeri belirlemeyi kolaylaştıran bir fonksiyondur. Mesela elimizde “yıl,etkinlik, bölge verileri ve row” şeklinde bir tablomuz var. Bu tablo üzerinde eşik değeri kullanarak ve kullanmadan analiz etmeye kalkarsak sonuçlar şu şekilde olmaktadır.

```
1  SELECT
2   Year,Event,Country,
3   ROW_NUMBER() OVER(Order by Year Desc,Country ASC) AS Row_N
4   from summer_medals
```

year	event	country	row_n
2012	58 - 68 KG	AFG	1
2012	1500M	ALG	2
2012	Hockey	ARG	3
2012	Hockey	ARG	4
2012	Hockey	ARG	5

```
1  SELECT
2    Year,Event,Country,
3    -- Assign numbers to each row
4   ROW_NUMBER()OVER(Order by Year Desc,Country ASC) AS Row_N
5  FROM Summer_Medals
6  ORDER BY Event,Row_N ASC;
```

year	event	country	row_n
2012	+ 100KG	BRA	215
2012	+ 100KG	FRA	643
2012	+ 100KG	GER	864
2012	+ 100KG	RUS	1514
2008	+ 100KG (Heavyweight)	CUB	2552

(Yukarıda yapmak istenilen örnek yıllara göre azalan ülke ismine göre artan bir sıralama hedeflenmiştir.)

ROW_NUMBER() ve ORDER BY()
Tablodaki güvenlik sıralamasını belirleme ve her satıra numara atamak için kullanılırlar. ORDER BY() sorgu sonucu dönen verilerin sıra düzenini belirler.(Artan azalan gibi) ROW NUMBER() ise belirli bir sıralama düzenine göre her satıra bir numara atar.Örnek verecek olursak satış temsilcilerinin yıla göre satış miktarlarını sorgulama işlemini yapabiliriz.

```
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	FirstName	LastName	Sales YTD
1	Linda	Mitchell	4251368,54
2	Jae	Pak	4116871,22
3	Michael	Blythe	3763178,17
4	Jillian	Carson	3189418,36
5	Ranjit	Varkey Chudukatil	3121616,32
6	José	Saraiva	2604540,71
7	Shu	Ito	2458535,61
8	Tsvi	Reiter	2315185,61
9	Rachel	Valdez	1827066,71
10	Tete	Mensa-Annan	1576562,19
11	David	Campbell	1573012,93
12	Garrett	Vargas	1453719,46
13	Lynn	Tsoflias	1421810,92
14	Pamela	Ansman-Wolfe	1352577,13

Burada “ROW_NUMBER() OVER(ORDER BY SalesYTD DESC)” kısmı ile satış miktarlarını azalan sırada sıralayıp numaralandırmaktadır.

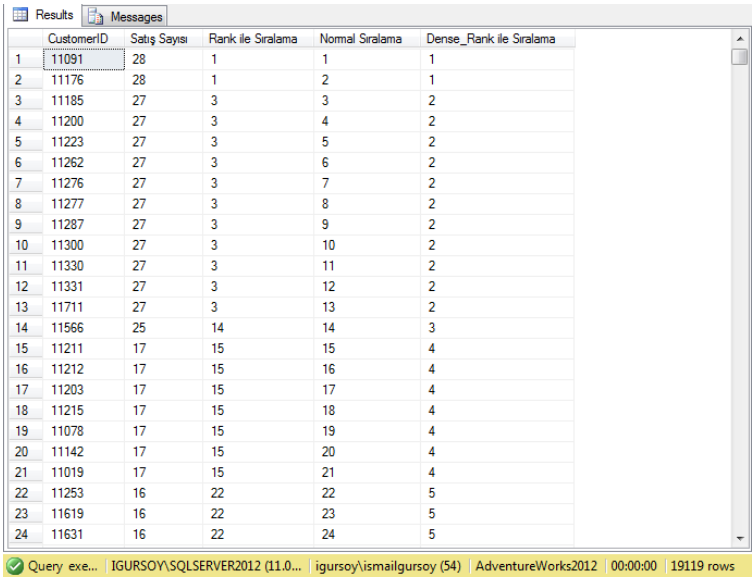
RANK() - DENSE_RANK()

RANK() fonksiyonu ile veri setlerindeki tekrar eden satırlara aynı sıralama değeri döndürülür ve bu sayede benzersiz değerler tespit edilir. Eğer öğeler eşit sıralama değerlerine sahip olur ise bir sonraki sıralama değeri atlanır.

DENSE_RANK() fonksiyonu ise RANK() fonksiyonundan farklı olarak numara atama işlemini yapmadan numaralam işlemine kaldığı yerden devam eder.

Burada anlatılanlara bir örnek ile anlatmak gerekirse;

```
SELECT CustomerID, COUNT(*) AS "Satış Sayısı",
RANK() OVER(ORDER BY COUNT(*) DESC) AS "Rank ile Sıralama",
ROW_NUMBER() OVER(ORDER BY COUNT(*) DESC) AS "Normal Sıralama",
DENSE_RANK() OVER(ORDER BY COUNT(*) DESC) AS "Dense_Rank ile Sıralama"
FROM Sales.SalesOrderHeader
GROUP BY CustomerID
ORDER BY COUNT(*) DESC;
```



Results | Messages

	CustomerID	Satış Sayısı	Rank ile Sıralama	Normal Sıralama	Dense_Rank ile Sıralama
1	11091	28	1	1	1
2	11176	28	1	2	1
3	11185	27	3	3	2
4	11200	27	3	4	2
5	11223	27	3	5	2
6	11262	27	3	6	2
7	11276	27	3	7	2
8	11277	27	3	8	2
9	11287	27	3	9	2
10	11300	27	3	10	2
11	11330	27	3	11	2
12	11331	27	3	12	2
13	11711	27	3	13	2
14	11566	25	14	14	3
15	11211	17	15	15	4
16	11212	17	15	16	4
17	11203	17	15	17	4
18	11215	17	15	18	4
19	11078	17	15	19	4
20	11142	17	15	20	4
21	11019	17	15	21	4
22	11253	16	22	22	5
23	11619	16	22	23	5
24	11631	16	22	24	5

Query exe... | IGURSOY\SQLSERVER2012 (11.0... | igursoy\ismailgursoy (54) | AdventureWorks2012 | 00:00:00 | 19119 rows

LEAD() LAG()

LAG ile LEAD fonksiyonu aslında bize aynı formatta veri içinde arama döndürür. Ama verinin farklı şekilde yorumlanmasında yardımcı olur. LAG fonksiyonu veri seti içerisindeki hedef veriden bir önceki veriyi hedef alarak işlem yapmamızı sağlarken LEAD fonksiyonu hedef veri'den sonraki herhangi bir veri'yi hedef alarak bize analiz yapma imkanı sunar. Metin olarak örnekleyecek olursak, bir robotu x tane odalı alanda hareket sayısı, oda metrekaresini tutan veriye sahibiz, bu metrekare alanda birim başına düşen hareket sayısının en fazla olduğu değeri bulmak istiyoruz. Burada yapmamız gerekenler yeni tablo açıp hesaplamak yerine robotun hareket sayısını OVER fonksiyonu ile belirli eşik değeri belirleyip oradan dönen sonucu odaların metrekaresine göre sıralayıp hareket sayısını LAG fonksiyonu ile önceki satır ile karşılaştırmasını sağlayarak (hatalı işlem) oranı hakkında analiz yapabiliriz. Veya LEAD fonksiyonu ile bir sonraki satıra göre Doğru işlem oranı hakkında analiz yapabiliriz.
Daha başka bir örnekle belirtecek olursak eğer;

```
SELECT ActivityId, IncidentId, CreatedOn,

LEAD(CreatedOn, 1, NULL) OVER (PARTITION BY IncidentId ORDER BY CreatedOn) AS
ActivityClosedOn

FROM @Activities

ORDER BY IncidentId, CreatedOn ASC
```

ActivityId	IncidentId	CreatedOn	ActivityClosedOn
126	2005	2014-01-01	2014-01-15
115	2005	2014-01-15	
142	2006	2014-03-01	2014-03-08
189	2006	2014-03-08	2014-04-01
275	2006	2014-04-01	2014-05-10
236	2006	2014-05-10	

(LEAD FONKSİYONU)

```
SELECT ActivityId, IncidentId, CreatedOn,

LEAD(CreatedOn, 1, NULL) OVER (PARTITION BY IncidentId ORDER BY CreatedOn) AS
ActivityClosedOn,

LAG(CreatedOn, 1, NULL) OVER (PARTITION BY IncidentId ORDER BY CreatedOn) AS
PreviousActivityCreatedOn

FROM @Activities

ORDER BY IncidentId, CreatedOn ASC
```

ActivityId	IncidentId	Created On	Activity Closed On	Previous Activity Created On
126	2005	2014-01-01	2014-01-15	
115	2005	2014-01-15		2014-01-01
142	2006	2014-03-01	2014-03-08	
189	2006	2014-03-08	2014-04-01	2014-03-01
275	2006	2014-04-01	2014-05-10	2014-03-08
236	2006	2014-05-10		2014-04-01

(LAG FONKSİYONU)

FIRST_VALUE() & LAST_VALUE()

LAST_VALUE() işlemi , veri kümesi içinde bir sütundaki son değer geçişi sağlar. Yani son değeri yürütür.Bu işlev belirli bir sıralama düzenine (örneğin , bir zaman sırası veya sıra numarası) dayalı olarak son değeri yürütür. FIRST_VALUE() ise sütundaki ilk değeri yürütür. Sütundaki ilk değer geçişini sağlar.

```sql
SELECT C.CustID, C.FirstName, C.LastName, P.ProductName, P.Price as 'Product Price',
O.Qty, O.OrderDate,
LAST_VALUE(P.ProductName)
OVER (PARTITION BY O.CustID ORDER BY O.OrderDate ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) as LastItemPurchased
FROM Orders as O
INNER JOIN Customers AS C
ON O.CustID = C.CustID
INNER JOIN Products as P
ON O.ProdID = P.ProductID
```

Results | Messages

CustID	FirstName	LastName	ProductName	Product Price	Qty	OrderDate	LastItemPurchased
50	Joshua	Porter	Side Tables	265.20	1	2021-07-24 00:00:00.000	Side Tables
50	Joshua	Porter	Large Bench	198.00	1	2021-07-01 00:00:00.000	Side Tables
55	Andrew	Bluefield	Side Tables	265.20	1	2021-07-18 00:00:00.000	Side Tables
55	Andrew	Bluefield	Coat Rack	45.00	1	2021-07-06 00:00:00.000	Side Tables
55	Andrew	Bluefield	Small Bench	169.40	1	2021-06-01 00:00:00.000	Side Tables
60	Jack	Donovan	Coat Rack	45.00	2	2021-06-06 00:00:00.000	Coat Rack
65	Cindy	Thatcher	Coffee Table	220.00	1	2021-07-14 00:00:00.000	Coffee Table
70	Gordon	Acres	Small Bench	169.40	3	2021-09-01 00:00:00.000	Small Bench
70	Gordon	Acres	Coffee Table	220.00	1	2021-08-06 00:00:00.000	Small Bench
70	Gordon	Acres	Side Tables	265.20	1	2021-08-06 00:00:00.000	Small Bench
75	Gretchen	Hamilton	Side Tables	265.20	1	2021-06-13 00:00:00.000	Side Tables

```sql
SELECT C.CustID, C.FirstName, C.LastName, P.ProductName, P.Price as 'Product Price',
O.Qty, O.OrderDate,
FIRST_VALUE(P.ProductName) OVER (ORDER BY O.OrderDate) as 'First Product Purchased Ever'
FROM Orders as O
INNER JOIN Customers AS C
ON O.CustID = C.CustID
INNER JOIN Products as P
ON O.ProdID = P.ProductID
```

Results | Messages

CustID	FirstName	LastName	ProductName	Product Price	Qty	OrderDate	First Product Purchased Ever
55	Andrew	Bluefield	Small Bench	169.40	1	2021-06-01 00:00:00.000	Small Bench
60	Jack	Donovan	Coat Rack	45.00	2	2021-06-06 00:00:00.000	Small Bench
75	Gretchen	Hamilton	Side Tables	265.20	1	2021-06-13 00:00:00.000	Small Bench
50	Joshua	Porter	Large Bench	198.00	1	2021-07-01 00:00:00.000	Small Bench
55	Andrew	Bluefield	Coat Rack	45.00	1	2021-07-06 00:00:00.000	Small Bench
65	Cindy	Thatcher	Coffee Table	220.00	1	2021-07-14 00:00:00.000	Small Bench
55	Andrew	Bluefield	Side Tables	265.20	1	2021-07-18 00:00:00.000	Small Bench
50	Joshua	Porter	Side Tables	265.20	1	2021-07-24 00:00:00.000	Small Bench
70	Gordon	Acres	Coffee Table	220.00	1	2021-08-06 00:00:00.000	Small Bench
70	Gordon	Acres	Side Tables	265.20	1	2021-08-06 00:00:00.000	Small Bench
70	Gordon	Acres	Small Bench	169.40	3	2021-09-01 00:00:00.000	Small Bench

PERCENT_RANK

Verileri yüzdelik olarak sıralamak için, genelde matematiksel işlemler yapmak üzere ihtiyaç duyulan fonksiyondur. 0'dan başlayıp 1'e doğru giden değerler üzerinden dağılım yapar. İlk satır her zaman 0, son satır her zaman 1 kabul edilir.

Her satır için uygulanan formül:

(satır sırası - 1) / (toplam satır sayısı - 1)

Aşağıda yer alan kod bloğunda; mal grupları 1 olan ürünlerin birim fiyatına göre küçükten büyüğe listelenmesi ve PR isimli sütunda percent_rank değerinin tutulması beklenmiştir.

```
SELECT
    URUN_ID,
    URUN_ADI,
    URETIM_RECETESI,
    MAL_GRUPLARI,
    BIRIM_FIYAT,
    PERCENT_RANK () OVER(PARTITION BY MAL_GRUPLARI ORDER BY BIRIM_FIYAT) AS PR
FROM
    URUNLER_20212425001
WHERE
    MAL_GRUPLARI = 1
ORDER BY
    MAL_GRUPLARI, pr;
```

Kod bloğunun çıktısı aşağıdaki şekilde olacaktır.

Results Messages

	URUN_ID	URUN_ADI	URETIM_RECETESI	MAL_GRUPLARI	BIRIM_FIYAT	PR
1	1	Iphone7	2	1	1000.00	0
2	7	Iphone 8	3	1	3000.00	0,5
3	11	Samsung Galaxy S3	2	1	6000.00	1

STRINGAGG() - LISTAGG()

2 veya daha üst derece normalizasyonlarda Foreign Key yapısının kullanılmasıyla; belirli bir gruba veya kategoriye ait veriler tek satırda gösterilmek istenebilir. Bu işlemi yapabilmek için, kullanılan SQL sunucusuna bağlı olarak “stringagg” veya “listagg” fonksiyonları kullanılmalıdır.

Aşağıda yer alan kod bloğunda, belirli mal gruplarına göre kategorize edilmiş ürünler arasından hangi ürünün hangi mal grubunda yer aldığını tek satırda görebilmek istenmiştir.

```
SELECT
    MAL_GRUPLARI,
    STRING_AGG (URUN_ADI, ', ') WITHIN GROUP (ORDER BY MAL_GRUPLARI) AS URUNLER
FROM
    URUNLER_20212425001
GROUP BY
    MAL_GRUPLARI;
```

Kod bloğunun çıktısı aşağıdaki gibi olacaktır.

Results | Messages

	MAL_GRUPLARI	URUNLER
1	1	Iphone7, Iphone 8, Samsung Galaxy S3
2	2	Xaomi Mini Tablet, Apple Tablet, Samsung Tablet