MSSQL İLE SORGULAMA İŞLEMLERİ

1-Products tablosu

```
CREATE TABLE Products (
    ProductId INT PRIMARY KEY,
    Name NVARCHAR(100),
    Price DECIMAL(18, 2),
    CategoryId INT
);
```

Products tablosuna veri ekleme

```
INSERT INTO Products (ProductId, Name, Price, CategoryId)
VALUES
    (1, 'Keyboard', 49.99, 1),
    (2, 'Mouse', 29.99, 1),
    (3, 'Monitor', 199.99, 2),
    (4, 'Printer', 149.99, 3),
    (5, 'Laptop', 999.99, 2);
```

2-Categories tablosu

```
CREATE TABLE Categories (
    CategoryId INT PRIMARY KEY,
    Name NVARCHAR(50)
);
```

Categories tablosuna veri ekleme

```
INSERT INTO Categories (CategoryId, Name)
```

```
VALUES
   (1, 'Computer Accessories'),
   (2, 'Computer Monitors'),
   (3, 'Printers');
```

3-Orders tablosu

```
CREATE TABLE Orders (
   OrderId INT PRIMARY KEY,
   OrderDate DATE,
   CustomerId INT
);
```

Orders tablosuna veri ekleme

```
INSERT INTO Orders (OrderId, OrderDate, CustomerId)
VALUES
   (1, '2024-05-20', 1),
   (2, '2024-05-21', 2),
   (3, '2024-05-22', 1);
```

4-Customers tablosu

```
CREATE TABLE Customers (
   CustomerId INT PRIMARY KEY,
   CustomerName NVARCHAR(100)
);
```

Customers tablosuna veri ekleme

```
INSERT INTO Customers (CustomerId, CustomerName)
```

```
VALUES
    (1, 'Alice'),
    (2, 'Bob'),
    (3, 'Charlie');
```

Sorgular

1. INNER JOIN

INNER JOIN, iki veya daha fazla tabloyu belirli bir koşula göre birleştirmek için kullanılır. İki tablo arasında ortak bir alan veya alanlar (genellikle primary key ve foreign key ilişkisiyle) kullanılarak birleştirme yapılır. INNER JOIN, birleşim koşulunu sağlayan kayıtları döndürür.

Örneğin, aşağıdaki sorguda Products ve Categories tabloları CategoryId alanı üzerinden INNER JOIN ile birleştirilmiştir:

```
SELECT p.ProductId, p.Name AS ProductName, p.Price, c.Name AS CategoryName
FROM Products p
INNER JOIN Categories c ON p.CategoryId = c.CategoryId;
```

Bu sorgu, Products tablosundaki ürünleri Categories tablosundaki kategori isimleriyle birleştirir ve ProductId, ProductName, Price ve CategoryName alanlarını içeren sonuçları döndürür.

2. LEFT JOIN

LEFT JOIN, sol tablonun (ilk tablo) tüm kayıtlarını sağ tabloya (ikinci tablo) göre birleştirir. Sol tablodaki her kayıt, sağ tabloda eşleşen bir kayıt bulunmasa bile sonuç kümesine dahil edilir. Eğer sağ tabloda eşleşen bir kayıt yoksa, sağ tablo için NULL değerler döner.

Örneğin, aşağıdaki sorguda Orders ve Customers tabloları CustomerId alanı üzerinden LEFT JOIN ile birleştirilmiştir:

```
SELECT o.OrderId, o.OrderDate, c.CustomerName
FROM Orders o
LEFT JOIN Customers c ON o.CustomerId = c.CustomerId;
```

Bu sorgu, Orders tablosundaki siparişleri Customers tablosundaki müşteri adlarıyla birleştirir. Eğer bir siparişin müşterisi yoksa, müşteri adı NULL olarak döner.

3. HAVING İşlemi

HAVING, GROUP BY ile gruplanmış veriler üzerinde filtreleme yapmak için kullanılır. HAVING, WHERE sorgusundan farklı olarak, GROUP BY ile gruplanan sonuçlar üzerinde çalışır.

Örneğin, aşağıdaki sorguda, ürün kategorilerinin toplam fiyatları 100 TL'nin üzerinde olan kategorileri buluruz:

```
SELECT c.CategoryId, c.Name AS CategoryName, SUM(p.Price) AS TotalPrice

FROM Products p

INNER JOIN Categories c ON p.CategoryId = c.CategoryId

GROUP BY c.CategoryId, c.Name

HAVING SUM(p.Price) > 100;
```

Bu sorgu, Products tablosundan alınan ürün fiyatlarını, Categories tablosunda gruplayarak kategoriye göre toplar. Sonra, HAVING koşulu ile sadece toplam fiyatı 100 TL'nin üzerinde olan kategorileri filtreler.

4. LIKE İşlemi

LIKE, metin tabanlı arama sorgularında kullanılır. Metinlerin belirli bir desene göre eşleşip eşleşmediğini kontrol etmek için kullanılır. % joker karakteri olarak kullanılır ve herhangi bir karakter dizisini temsil eder.

Örneğin, aşağıdaki sorgularda Products tablosundan ürün adı içinde 'keyboard' geçen ürünleri ve Customers tablosundan müşteri adı 'A' ile başlayan müşterileri buluruz:

```
-- Ürün adında 'keyboard' geçen ürünleri bulma (case insensitive)

SELECT ProductId, Name, Price

FROM Products

WHERE LOWER(Name) LIKE '%keyboard%';

-- Müşteri adı 'A' ile başlayan müşterileri bulma

SELECT CustomerId, CustomerName

FROM Customers

WHERE CustomerName LIKE 'A%';
```

Bu sorgular, LIKE operatörünü kullanarak belirli desenlere göre veri aramayı gösterir.

5. Toplu İşlemler (Transaction)

Transaksiyonlar, bir veya birden fazla SQL işlemi gruplandırarak, tüm işlemlerin başarıyla tamamlanıp tamamlanmadığını kontrol etmenizi sağlar. BEGIN TRANSACTION, COMMIT ve ROLLBACK ifadeleri ile

işlemler gruplandırılır ve işlemlerin tamamı başarılıysa COMMIT, hata oluşursa ROLLBACK kullanılarak geri alınabilir.

Örneğin, aşağıdaki sorgu, belirli bir kategoriye sahip ürünleri siler ve sonra bu kategoriyi kaldırır:

```
BEGIN TRANSACTION;

-- Belirli bir kategoriye sahip ürünleri silme
DELETE FROM Products
WHERE CategoryId IN (SELECT CategoryId FROM Categories WHERE Name = 'Printers');

-- Printers kategorisini silme
DELETE FROM Categories WHERE Name = 'Printers';

COMMIT TRANSACTION;
```

Bu sorgu, bir TRANSACTION içinde iki işlemi bir arada işler ve eğer birinde hata olursa, diğerini geri alabilir.

6. MIN ve MAX Kullanımı

```
-- Products tablosunda en düşük ve en yüksek fiyatlı ürünleri bulma
SELECT MIN(Price) AS MinPrice, MAX(Price) AS MaxPrice
FROM Products;
```

Bu sorguda MIN() fonksiyonu en düşük fiyatı, MAX() fonksiyonu ise en yüksek fiyatı bulmak için kullanılmıştır.

7. COUNT Kullanımı

```
-- Orders tablosundaki siparişlerin toplam sayısını bulma
SELECT COUNT(*) AS TotalOrders
FROM Orders;
```

COUNT() fonksiyonu, belirtilen sorgu veya gruplama sonucunda kaç adet kayıt olduğunu döndürür. Bu örnekte, Orders tablosundaki toplam sipariş sayısını bulmak için kullanılmıştır.

8. AVG Kullanımı

```
-- Products tablosundaki ürünlerin ortalama fiyatını bulma
SELECT AVG(Price) AS AvgPrice
```

```
FROM Products;
```

AVG() fonksiyonu, belirtilen sorgu veya gruplama sonucunda ortalama değeri hesaplar. Bu örnekte, Products tablosundaki ürünlerin ortalama fiyatını bulmak için kullanılmıştır.

9. GROUP BY ve ORDER BY Kullanımı

```
-- Categories tablosundaki her kategori için kaç ürün olduğunu ve toplam fiyatını bulma
SELECT c.Name AS CategoryName, COUNT(p.ProductId) AS TotalProducts, SUM(p.Price) AS TotalPrice
FROM Categories c
LEFT JOIN Products p ON c.CategoryId = p.CategoryId
GROUP BY c.Name
ORDER BY TotalPrice DESC;
```

GROUP BY ve ORDER BY ifadeleri kullanılarak kategoriye göre gruplama yapılır ve gruplanmış sonuçlar belirli bir sıraya göre sıralanır.

GROUP BY c.Name: Kategorilere göre gruplama yapılır.

COUNT(p.ProductId) AS TotalProducts: Her kategori için ürün sayısını sayar.

SUM(p.Price) AS TotalPrice: Her kategori için toplam fiyatı hesaplar.

ORDER BY TotalPrice DESC: Toplam fiyata göre azalan şekilde sıralama yapar.

10. Alt Sorgu ile En Çok Sipariş Veren Müşteriyi Bulma

```
-- En çok sipariş veren müşteriyi bulma
SELECT CustomerId, CustomerName
FROM Customers
WHERE CustomerId = (
    SELECT CustomerId
    FROM Orders
    GROUP BY CustomerId
    ORDER BY COUNT(*) DESC
    LIMIT 1
```

```
);
```

Bu sorguda, iç içe geçmiş bir alt sorgu (subquery) kullanılarak en çok sipariş veren müşterinin CustomerId ve CustomerName bilgileri bulunur.

11. BETWEEN Kullanımı

```
-- Products tablosundaki belirli fiyat aralığında (between) olan ürünleri bulma

SELECT ProductId, Name, Price

FROM Products

WHERE Price BETWEEN 50 AND 200;
```

BETWEEN operatörü, belirli bir aralıktaki değerleri döndürmek için kullanılır. Bu örnekte, Products tablosundaki fiyatı 50 ile 200 arasında olan ürünleri bulmak için kullanılmıştır.

12. IN Kullanımı

```
-- Belirli kategoriye ait ürünlerin listesini bulma (in)

SELECT ProductId, Name, Price

FROM Products

WHERE CategoryId IN (SELECT CategoryId FROM Categories WHERE Name = 'Computer Monitors');
```

IN operatörü, bir değerin bir alt sorgu sonucunda döndürülen değerler listesinde olup olmadığını kontrol etmek için kullanılır. Bu örnekte, Categories tablosundan 'Computer Monitors' kategorisine ait ürünleri bulmak için kullanılmıştır.

13. JOIN ve WHERE Kullanımı

```
-- Müşteri adı 'Alice' olan kişinin verdiği siparişleri bulma

SELECT o.OrderId, o.OrderDate

FROM Orders o

INNER JOIN Customers c ON o.CustomerId = c.CustomerId

WHERE c.CustomerName = 'Alice';
```

Bu sorguda, INNER JOIN ile Orders ve Customers tabloları birleştirilir ve WHERE koşulu ile müşteri adı 'Alice' olan kişinin verdiği siparişler bulunur.

14. WHERE Kullanımı ile Belirli Kategorideki Ürünlerin Listesini Bulma

-- Bir kategorideki ürünlerin listesini bulma

```
SELECT p.ProductId, p.Name, p.Price

FROM Products p

INNER JOIN Categories c ON p.CategoryId = c.CategoryId

WHERE c.Name = 'Laptops';
```

Bu sorguda, INNER JOIN ile Products ve Categories tabloları birleştirilir ve WHERE koşulu ile 'Laptops' kategorisindeki ürünlerin listesi bulunur.

15. Toplu İşlemler (Transaction)

```
BEGIN TRANSACTION;

-- Belirli bir kategoriye ait ürünleri silme

DELETE FROM Products

WHERE CategoryId IN (SELECT CategoryId FROM Categories WHERE Name = 'Computer Accessories');

-- Silinen kategoriyi kaldırma

DELETE FROM Categories WHERE Name = 'Computer Accessories';

COMMIT TRANSACTION;
```

Bu sorguda, BEGIN TRANSACTION, COMMIT ve ROLLBACK ifadeleri kullanılarak bir transaksiyon içinde iki ayrı işlem gruplandırılmıştır. Eğer her iki işlem de başarılı olursa, COMMIT TRANSACTION ile onaylanır, aksi takdirde ROLLBACK TRANSACTION ile geri alınabilir.

Bu açıklamalar ile MSSQL sorgulama elemanlarını ve işlemlerini detaylı bir şekilde anlamış olmalısınız. Her bir sorgu, belirli bir senaryoya veya işlem türüne odaklanarak, veritabanı yönetiminde nasıl kullanılabileceğini gösteriyor. Bu sorguları kendi veritabanınıza uygun şekilde adapte edebilir ve işlemlerinizi yönetebilirsiniz.