SQL Sorgu Örnekleri

Soru 1: Her müşterinin toplam sipariş sayısını ve ortalama sipariş tutarını gösteren sorgu.

```
SELECT m.MusteriAdi + ' ' + m.MusteriSoyadi AS MusteriAdiSoyadi,

   COUNT(s.SiparisID) AS ToplamSiparisSayisi,

   AVG(s.ToplamTutar) AS OrtalamaSiparisTutari

FROM Musteriler m

LEFT JOIN Siparisler s ON m.MusteriID = s.MusteriID

GROUP BY m.MusteriID,

   m.MusteriAdi,

   m.MusteriSoyadi;
```

Açıklama:

```
SELECT m.MusteriAdi + ' ' + m.MusteriSoyadi AS MusteriAdiSoyadi,
```

- **Amaç:** Müşteri adını ve soyadını birleştirerek tek bir sütunda göstermek.
- + ' ' + ifadesi iki metni araya boşluk koyarak birleştirir.
- AS MusteriAdiSoyadi: Bu sütuna anlamlı bir isim verir, yani bu alanın başlığı "MusteriAdiSoyadi" olur.

COUNT(s.SiparisID) AS ToplamSiparisSayisi,

- **Amaç:** Müşterinin kaç siparişi olduğunu saymak.
- COUNT(): Satır sayısını verir. Burada her müşterinin sipariş sayısını hesaplıyoruz.
- AS ToplamSiparisSayisi: Sonuç sütununa isim verir.

AVG(s.ToplamTutar) AS OrtalamaSiparisTutari

- **Amaç:** Müşterinin siparişlerinin ortalama tutarını hesaplamak.
- AVG(): Belirtilen sütunun ortalamasını alır.
- ToplamTutar: Her siparişin toplam tutarını tutan sütundur.

FROM Musteriler m

- **Amaç:** Verileri alacağımız ana tablo.
- Musteriler: Müşteri bilgilerini tutan tablo.
- m: Bu tabloya kısa bir takma ad (alias). Böylece yazarken daha kısa ifade kullanabiliyoruz.

LEFT JOIN Siparisler s ON m.MusteriID = s.MusteriID

- **Amaç:** Müşterilerle onların siparişlerini birleştirmek.
- LEFT JOIN: Müşterilerin hepsi listelensin, siparişi olmayanların da adı görünsün.
- s: Siparisler tablosunun kısa adı.
- ON m.MusteriID = s.MusteriID: İki tabloyu birleştirirken kullanılacak ortak alan (foreign key ilişkisi).

GROUP BY m.MusteriID, m.MusteriAdi, m.MusteriSoyadi;

- **Amaç:** Her müşteri için ayrı ayrı toplama ve ortalama işlemi yapabilmek.
- GROUP BY: Verileri gruplandırır, böylece her müşteri için bir satır oluşur.

Genel Mantık:

Bu sorgu her müşteri için:

- Kaç sipariş verdiğini,
- Verdiği siparişlerin ortalama tutarını gösteriyor. Siparişi olmayan müşteriler de listede yer alıyor çünkü LEFT JOIN kullanıldı.

Soru 2: Son 3 gün içinde en az 2 sipariş veren müşterileri ve toplam harcamalarını gösteren sorgu

```
SELECT m.MusteriAdi + ' ' + m.MusteriSoyadi AS MusteriAdiSoyadi,

   COUNT(s.SiparisID) AS SiparisSayisi,

   SUM(s.ToplamTutar) AS ToplamHarcama

FROM Musteriler m

JOIN Siparisler s ON m.MusteriID = s.MusteriID

WHERE s.SiparisTarihi >= DATEADD(DAY, -3, GETDATE())

GROUP BY m.MusteriID,

   m.MusteriAdi,

   m.MusteriSoyadi

HAVING COUNT(s.SiparisID) >= 2;
```

Açıklama:

SELECT m.MusteriAdi + ' ' + m.MusteriSoyadi AS MusteriAdiSoyadi,

- Tıpkı 1. soruda olduğu gibi, müşterinin adı ve soyadını birleştirip kullanıcıya tek bir sütunda gösteriyoruz.

COUNT(s.SiparisID) AS SiparisSayisi,

- **Amaç:** Her müşterinin son 3 gündeki sipariş sayısını saymak.
- COUNT(): Siparişlerin toplam sayısını verir.

SUM(s.ToplamTutar) AS ToplamHarcama

- **Amaç:** Müşterinin bu sürede yaptığı harcamaların toplamını hesaplar.
- SUM(): Belirtilen sütunun toplamını alır.
- ToplamTutar: Her bir siparişin tutarı.

FROM Musteriler m

- Ana tablomuz müşteri bilgileri.

JOIN Siparisler s ON m.MusteriID = s.MusteriID

- JOIN burada INNER JOIN anlamındadır. Yani sadece **siparişi olan** müşteriler alınır.
- ON: Hangi alanlar üzerinden bağlandığını belirtir.

WHERE s.SiparisTarihi >= DATEADD(DAY, -3, GETDATE())

- **Amaç:** Sipariş tarihi son 3 gün olanları filtrelemek
- GETDATE(): Bugünün tarihi.
- DATEADD(DAY, -3, GETDATE()): Bugünden 3 gün önceki tarihi hesaplar.
- >=: Bu tarihten sonra olanları alır.

GROUP BY m.MusteriID, m.MusteriAdi, m.MusteriSoyadi

- Gruplandırma işlemi, her müşteri için tek bir satır çıkması için yapılır.

HAVING COUNT(s.SiparisID) >= 2;

- **HAVING**, GROUP BY ile birlikte çalışır. Gruplanmış sonuçlara filtre uygulamak için kullanılır.
- Bu satır şunu der: "Sadece 2 veya daha fazla sipariş verenleri getir."

Genel Mantık:

Bu sorgu, **son 3 gün içinde en az 2 sipariş** vermiş olan müşterileri bulur ve onların:

- Kaç sipariş verdiğini,
- Ne kadar harcadığını gösterir.

Soru 3: Her müşterinin en son sipariş tarihini ve tutarını gösteren sorgu

```
WITH SonSiparisler AS (

  SELECT MusteriID,

    SiparisTarihi,

        ToplamTutar,

        ROW_NUMBER() OVER (PARTITION BY MusteriID ORDER BY SiparisTarihi DESC) AS
Sira

  FROM Siparisler

)

SELECT m.MusteriAdi + ' ' + MusteriSoyadi AS MusteriAdiSoyadi,

           s.SiparisTarihi AS SonSiparisTarihi,

           s.ToplamTutar AS SonSiparisTutari

FROM Musteriler m

JOIN SonSiparisler s ON m.MusteriID = s.MusteriID

WHERE s.Sira = 1;
```

Açıklama:

WITH SonSiparisler AS (...)

- Bu satır bir **CTE** (Common Table Expression) tanımlar.
- Yani geçici bir sanal tablo oluştururuz.
- Bu yapı kodu daha okunabilir ve düzenli hale getirir.
- Adı: SonSiparisler, içerideki sorgunun sonucu bu isimle çağrılabilir.

CTE Sorgusu:

```
SELECT MusteriID,

   SiparisTarihi,

   ToplamTutar,

   ROW_NUMBER() OVER (PARTITION BY MusteriID ORDER BY SiparisTarihi DESC) AS Sira

FROM Siparisler
```

Ne yapıyor?

- ROW_NUMBER() her müşteriye ait siparişleri **sipariş tarihine göre sıraya koyuyor**.
- PARTITION BY MusteriID: Her müşteri kendi içinde ayrı ayrı sıralanıyor.
- ORDER BY SiparisTarihi DESC: En yeni sipariş en üstte olur.
- AS Sira: Sıralama sonucunu "Sira" adıyla verir.

🎯 Yani: Her müşterinin en yeni siparişi "Sira = 1" olacak.

Ana Sorgu:

```
SELECT m.MusteriAdi + ' ' + MusteriSoyadi AS MusteriAdiSoyadi,

   s.SiparisTarihi AS SonSiparisTarihi,

   s.ToplamTutar AS SonSiparisTutari

FROM Musteriler m

JOIN SonSiparisler s ON m.MusteriID = s.MusteriID

WHERE s.Sira = 1;
```

Açıklamalar:

- JOIN: Müşteriler ile siparişleri birleştiriyoruz.
- WHERE s.Sira = 1: Sadece en yeni sipariş alınır.
- SonSiparisTarihi, SonSiparisTutari: Bu siparişe ait tarih ve tutar.

Genel Mantık:

- Bu sorgu, her müşterinin **en son yaptığı siparişi** bulur.
- ROW_NUMBER() ile sıraya koyarız.
- Sonra sadece 1. sıradaki siparişi çekeriz.

Soru 4: Ödeme durumu 'Beklemede' olan siparişlerin toplam tutarını müşteri bazında gösteren sorgu

```
SELECT m.MusteriAdi + ' ' + m.MusteriSoyadi AS MusteriAdiSoyadi,

   COUNT(s.SiparisID) AS ToplamSiparisSayisi,

   SUM(s.ToplamTutar) AS ToplamBekleyenTutar

FROM Musteriler m

JOIN Siparisler s ON m.MusteriID = s.MusteriID

WHERE s.OdemeDurumu = 'Beklemede'

GROUP BY m.MusteriID,

   m.MusteriAdi,

   m.MusteriSoyadi;
```

Açıklama:

SELECT m.MusteriAdi + ' ' + m.MusteriSoyadi AS MusteriAdiSoyadi,

- Müşteri adı ve soyadını birleştiriyoruz.
- + ' ' + → Araya boşluk koyarak daha okunabilir hale getiriyoruz.

COUNT(s.SiparisID) AS ToplamSiparisSayisi

- Beklemede olan siparişlerin adedini verir.
- Yani her müşteri için "kaç siparişi beklemede" onu gösteriyor.

SUM(s.ToplamTutar) AS ToplamBekleyenTutar

- Beklemede olan siparişlerin **toplam tutarını** toplar.
- Örnek: Ali'nin 2 siparişi beklemede, biri 100₺ diğeri 50₺ → toplam 150₺ yazar.

FROM Musteriler m

- Ana tablomuz yine müşteriler.

JOIN Siparisler s ON m.MusteriID = s.MusteriID

- Müşterileri siparişlerle eşleştiriyoruz.
- Sadece eşleşen kayıtlar gelir (INNER JOIN).

WHERE s.OdemeDurumu = 'Beklemede'

- Filtre burada: Sadece ödeme durumu "Beklemede" olan siparişler dahil ediliyor.

GROUP BY m.MusteriID, m.MusteriAdi, m.MusteriSoyadi;

- Müşteri bazında gruplama yapılır.
- Böylece her müşteri için tek satır olur ve üstteki COUNT() / SUM() düzgün çalışır.

Genel Mantık:

Bu sorgu, beklemede olan siparişleri alıyor ve her müşteri için:

- Kaç tane bekleyen sipariş var,
- Toplam ne kadar tutuyor gibi bilgileri çıkarıyor.

Soru 5: Her müşterinin sipariş geçmişini, sipariş tarihine göre sıralı şekilde gösteren ve her sipariş için bir önceki siparişle arasındaki gün farkını hesaplayan sorgu

```
SELECT m.MusteriAdi + ' ' + m.MusteriSoyadi AS MusteriAdiSoyadi,

   s.SiparisTarihi,,

   s.ToplamTutar,

   s.OdemeDurumu,

      DATEDIFF(

       DAY,

       LAG(s.SiparisTarihi) OVER (PARTITION BY m.MusteriID ORDER BY
s.SiparisTarihi),

       s.SiparisTarihi

   ) AS OncekiSiparisOlanGunFarki

FROM Musteriler m

JOIN Siparisler s ON m.MusteriID = s.MusteriID

ORDER BY m.MusteriID, s.SiparisTarihi;
```

Açıklama:

SELECT m.MusteriAdi + '' + m.MusteriSoyadi AS MusteriAdiSoyadi,

- Ad + Soyad birleştiriyoruz, klasikleşmiş müşteri etiketi.

s.SiparisTarihi, s.ToplamTutar, s.OdemeDurumu,

- Siparişin **tarihi**, **toplam tutarı** ve **ödeme durumu** gibi detayları direkt çekiyoruz.

DATEDIFF(...) AS OncekiSiparisOlanGunFarki

```
DATEDIFF(
    DAY,
    LAG(s.SiparisTarihi) OVER (PARTITION BY m.MusteriID ORDER BY s.SiparisTarihi),
    s.SiparisTarihi
)
```

Ne bu?

- DATEDIFF(DAY, önceki_tarih, şimdiki_tarih) → İki tarih arasındaki gün farkını verir.
- Biz burada "önceki sipariş tarihi" ile "şu anki sipariş tarihi" arasındaki farkı buluyoruz.

LAG(...) OVER (...)

LAG(s.SiparisTarihi):

- Bu, bir önceki satırdaki **sipariş tarihini** alır.
- Yani bir müşterinin önceki sipariş tarihine geri dönüp bakar.

OVER (PARTITION BY m.MusteriID ORDER BY s.SiparisTarihi):

- Her müşteri kendi içinde değerlendirilir (**PARTITION BY**).
- Sipariş tarihine göre sıralanır (**ORDER BY s.SiparisTarihi**).

Sonuç: Her müşteri için, siparişler sırayla listelenir ve her satırda, bir önceki sipariş tarihi alınır.

JOIN ve ORDER BY

```
JOIN Siparisler s ON m.MusteriID = s.MusteriID
ORDER BY m.MusteriID, s.SiparisTarihi;
```

- Müşteri ve sipariş tablosunu eşleştiriyoruz.
- Sonuçları müşteri ID’ye göre ve sipariş tarihine göre sıralıyoruz → Bu sıralama sayesinde LAG() doğru çalışıyor.

İlk siparişin "öncesi" yok → NULL

Sonraki siparişlerde gün farkı yazıyor.

Bu sorgu sayesinde, müşteri sipariş alışkanlıklarını analiz edebiliriz.
Mesela:

- Kaç günde bir sipariş veriyor?
- Düzenli mi sipariş veriyor?
- Uzun aralıklar var mı?
