LINQ Alıştırmaları ve Ödevleri-1

Tüm Ödevlerde Kullanılacak Veri Seti

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Category { get; set; }
    public decimal Price { get; set; }
    public int StockQuantity { get; set; }
}

public class Customer
{
    public int Id { get; set; }
    public string FullName { get; set; }
    public string City { get; set; }
}

public class Order
{
    public int OrderId { get; set; }
    public int CustomerId { get; set; }
    public int ProductId { get; set; }
    public DateTime OrderDate { get; set; }
    public int Quantity { get; set; }
}

// Data Lists
List<Product> products = new List<Product>
{
    new Product { Id = 1, Name = "Akıllı Telefon", Category =
"Elektronik", Price = 4500.00m, StockQuantity = 50 },
    new Product { Id = 2, Name = "Dizüstü Bilgisayar", Category =
"Elektronik", Price = 12000.00m, StockQuantity = 25 },
    new Product { Id = 3, Name = "Kahve Makinesi", Category = "Mutfak
Eşyaları", Price = 1200.00m, StockQuantity = 10 },
    new Product { Id = 4, Name = "Roman Kitabı", Category = "Kitap",
Price = 80.00m, StockQuantity = 200 },
```

```
    new Product { Id = 5, Name = "Kablosuz Kulaklık", Category =
"Elektronik", Price = 950.00m, StockQuantity = 75 },
    new Product { Id = 6, Name = "Tarih Ansiklopedisi", Category =
"Kitap", Price = 250.00m, StockQuantity = 80 },
    new Product { Id = 7, Name = "Blender Seti", Category = "Mutfak
Eşyaları", Price = 1800.00m, StockQuantity = 40 }
};

List<Customer> customers = new List<Customer>
{
    new Customer { Id = 1, FullName = "Ali Veli", City = "İstanbul" },
    new Customer { Id = 2, FullName = "Ayşe Yılmaz", City = "Ankara" },
    new Customer { Id = 3, FullName = "Mehmet Kaya", City = "İzmir" },
    new Customer { Id = 4, FullName = "Zeynep Demir", City = "Ankara" },
    new Customer { Id = 5, FullName = "Fatma Öztürk", City = "Bursa" } //
Siparişi olmayan müşteri
};

List<Order> orders = new List<Order>
{
    new Order { OrderId = 101, CustomerId = 1, ProductId = 2, OrderDate =
new DateTime(2025, 10, 1), Quantity = 1 },
    new Order { OrderId = 102, CustomerId = 2, ProductId = 4, OrderDate =
new DateTime(2025, 10, 5), Quantity = 3 },
    new Order { OrderId = 103, CustomerId = 1, ProductId = 5, OrderDate =
new DateTime(2025, 10, 8), Quantity = 2 },
    new Order { OrderId = 104, CustomerId = 3, ProductId = 1, OrderDate =
new DateTime(2025, 10, 10), Quantity = 1 },
    new Order { OrderId = 105, CustomerId = 4, ProductId = 7, OrderDate =
new DateTime(2025, 11, 12), Quantity = 1 },
    new Order { OrderId = 106, CustomerId = 2, ProductId = 2, OrderDate =
new DateTime(2025, 11, 15), Quantity = 1 },
    new Order { OrderId = 107, CustomerId = 1, ProductId = 4, OrderDate =
new DateTime(2025, 11, 21), Quantity = 2 },
    new Order { OrderId = 108, CustomerId = 3, ProductId = 5, OrderDate =
new DateTime(2025, 12, 1), Quantity = 1 },
    new Order { OrderId = 109, CustomerId = 2, ProductId = 6, OrderDate =
new DateTime(2025, 12, 4), Quantity = 1 },
    new Order { OrderId = 110, CustomerId = 4, ProductId = 2, OrderDate =
new DateTime(2025, 12, 8), Quantity = 1 }
};
```

Ödev 1: Temel Filtreleme ve Sıralama

Senaryo: E-ticaret sitesinin ana sayfasında, fiyatı 1000 TL'den az olan ve stoğunda 50'den fazla ürün bulunan ürünleri, fiyata göre azalan şekilde (pahalıdan ucuza) sıralayarak listelemek istiyoruz.
İstenenler:

1. `products` listesini filtreleyerek fiyatı 1000 TL'den az olanları alın.
2. Bu filtrelenmiş ürünler arasından stok adedi (`StockQuantity`) 50'den fazla olanları seçin.
3. Sonuçları fiyata göre pahalıdan ucuza doğru sıralayın.
4. Ekrana ürünün Adını, Fiyatını ve Stok Adedini yazdırın.
   İpucu: Sıralama için `OrderByDescending()` kullanın.
   Beklenen Çıktı:

```
Ürün Adı: Kablosuz Kulaklık, Fiyat: 950.00 TL, Stok: 75
Ürün Adı: Tarih Ansiklopedisi, Fiyat: 250.00 TL, Stok: 80
Ürün Adı: Roman Kitabı, Fiyat: 80.00 TL, Stok: 200
```

Ödev 2: Projeksiyon (Veri Şekillendirme)

Senaryo: Pazarlama ekibi, tüm ürünlerin stok durumunu ve KDV dahil fiyatını içeren basitleştirilmiş bir liste istiyor.
İstenenler:

1. `products` listesindeki her bir ürün için yeni bir anonim tip (anonymous type) oluşturun.
2. Bu yeni tip `ProductName`, `StockStatus` ve `PriceWithVat` özelliklerini içermelidir.
3. `StockStatus` özelliği, stok adedi 20'den az ise "Kritik Stok", değilse "Stok Yeterli" değerini almalıdır.
4. `PriceWithVat` özelliği, ürünün normal fiyatına %18 KDV eklenerek hesaplanmalıdır.
5. Oluşturulan bu yeni listeyi ekrana yazdırın.
   İpucu: `Select()` metodu içinde `new { ... }` kullanarak anonim tipler oluşturabilirsiniz. Koşullu atama için `?:` (ternary) operatörü işinizi kolaylaştırır.
   Beklenen Çıktı:

```
Product Name: Akıllı Telefon, Price with VAT: 5310.00, Stock Status: Stok Yeterli
Product Name: Dizüstü Bilgisayar, Price with VAT: 14160.00, Stock Status: Stok Yeterli
Product Name: Kahve Makinesi, Price with VAT: 1416.00, Stock Status: Kritik Stok
... (diğer ürünler için de benzer formatta)
```

Ödev 3: Gruplama (`GroupBy`) ve Gruplanmış Veri Üzerinde İşlem

Senaryo: Raporlama departmanı, her bir kategorideki en pahalı ürünün hangisi olduğunu gösteren bir rapor istiyor.
İstenenler:

1. `products` listesini `Category` özelliğine göre gruplayın.
2. Her bir grup için kategori adını ve o kategorideki en pahalı ürünün adını ve fiyatını bulun.
   İpucu: `GroupBy()` sonrası `Select()` ile her grup (`g`) için `g.OrderByDescending(p => p.Price).First()` gibi bir ifade kullanarak en pahalı ürünü bulabilirsiniz.
   Beklenen Çıktı:

```
Kategori: Elektronik, En Pahalı Ürün: Dizüstü Bilgisayar (12000.00 TL)
Kategori: Mutfak Eşyaları, En Pahalı Ürün: Blender Seti (1800.00 TL)
Kategori: Kitap, En Pahalı Ürün: Tarih Ansiklopedisi (250.00 TL)
```

Ödev 4: Üç Listeyi Birleştirme (`Join`)

Senaryo: Muhasebe departmanı, her bir siparişin detayını içeren (müşteri adı, şehir, ürün adı, adet, birim fiyat ve sipariş tarihi) kapsamlı bir rapor istiyor.
İstenenler:

1. `orders`, `customers` ve `products` listelerini uygun ID'ler üzerinden birleştirin.
2. Her bir sipariş için `CustomerName`, `City`, `ProductName`, `Quantity`, `Price` ve `OrderDate` bilgilerini içeren yeni bir anonim tip oluşturun.
3. Bu birleştirilmiş ve şekillendirilmiş listeyi ekrana yazdırın.
   İpucu: `Join` işlemini zincirleme olarak iki kez kullanmanız gerekecek. `orders.Join(...).Join(...)` gibi.
   Beklenen Çıktı:

```
Müşteri: Ali Veli (İstanbul), Ürün: Dizüstü Bilgisayar, Adet: 1, Fiyat: 12000.00, Tarih: 01.10.2025
Müşteri: Ayşe Yılmaz (Ankara), Ürün: Roman Kitabı, Adet: 3, Fiyat: 80.00, Tarih: 05.10.2025
... (toplam 10 sipariş için benzer formatta)
```

Ödev 5: Birleştirme ve Filtreleme (`Join` & `Where`)

Senaryo: Müşteri hizmetleri, Ankara'daki müşterilerin verdiği tüm siparişleri görmek istiyor.
İstenenler:

1. `orders` ve `customers` listelerini `CustomerId` üzerinden birleştirin.
2. Sadece şehri "Ankara" olan müşterilerin siparişlerini filtreleyin.
3. Bu siparişlerin `OrderId`, `FullName` ve `OrderDate` bilgilerini ekrana yazdırın.
   İpucu: `Join` işleminden sonra `Where()` metodu ile filtreleme yapın.
   Beklenen Çıktı:

```
Sipariş ID: 102, Müşteri: Ayşe Yılmaz, Tarih: 05.10.2025
Sipariş ID: 105, Müşteri: Zeynep Demir, Tarih: 12.11.2025
Sipariş ID: 106, Müşteri: Ayşe Yılmaz, Tarih: 15.11.2025
Sipariş ID: 109, Müşteri: Ayşe Yılmaz, Tarih: 04.12.2025
Sipariş ID: 110, Müşteri: Zeynep Demir, Tarih: 08.12.2025
```

Ödev 6: Karmaşık Gruplama ve Toplama (En İyi Müşteri)

Senaryo: Yönetim, toplamda en çok para harcayan müşteriyi (ve ne kadar harcadığını) öğrenmek istiyor.
İstenenler:

1. `orders` listesini `products` listesi ile birleştirerek her siparişin toplam tutarını (`Quantity * Price`) hesaplayın.
2. Bu sonuçları `CustomerId`'ye göre gruplayın.
3. Her müşteri için toplam harcamayı (`Sum`) hesaplayın.
4. Sonuçları toplam harcamaya göre azalan şekilde sıralayıp en üstteki müşteriyi alın.
5. Müşterinin adını ve toplam harcamasını ekrana yazdırın.
   İpucu: Bu çok adımlı bir sorgudur. `Join`, `GroupBy`, `Select` (içinde `Sum` ile) ve `OrderByDescending().First()` kombinasyonunu kullanacaksınız.
   Beklenen Çıktı:

```
En Çok Harcama Yapan Müşteri: Ayşe Yılmaz, Toplam Harcama: 12490.00 TL
```

Ödev 7: Eleman Operatörleri (`FirstOrDefault`, `SingleOrDefault`)

Senaryo: ID'si 6 olan ürünün bilgilerini getirmek istiyoruz. Eğer bu ID'ye sahip bir ürün yoksa programın hata vermemesi, bunun yerine bir bilgilendirme mesajı göstermesi gerekiyor.
İstenenler:

1. `products` listesi içinde ID'si 6 olan ürünü `FirstOrDefault()` kullanarak bulun.
2. Eğer ürün bulunduysa (yani sonuç `null` değilse), ürünün adını ve kategorisini ekrana yazdırın.
3. Eğer ürün bulunamadıysa, "ID'si 6 olan ürün bulunamadı." mesajını yazdırın.
4. Aynı işlemi ID'si 99 olan bir ürün için de tekrarlayın.
   İpucu: `FirstOrDefault` geriye `null` dönebilir, bu yüzden bir `if` kontrolü yapmak iyi bir pratiktir.
   Beklenen Çıktı:

```
Ürün Adı: Tarih Ansiklopedisi, Kategori: Kitap
ID'si 99 olan ürün bulunamadı.
```

Ödev 8: SayfaLama Operatörleri (`Skip` & `Take`)

Senaryo: Ürün listeleme sayfasında, ürünleri pahalıdan ucuza sıraladıktan sonra ikinci sayfadaki ürünleri göstermek istiyoruz. Her sayfada 3 ürün bulunsun.
İstenenler:

1. `products` listesini fiyata göre pahalıdan ucuza (`OrderByDescending`) sıralayın.
2. İlk sayfadaki 3 ürünü atlamak için `Skip(3)` kullanın.
3. İkinci sayfadaki 3 ürünü almak için `Take(3)` kullanın.
4. Seçilen ürünlerin adını ve fiyatını ekrana yazdırın.
   Beklenen Çıktı:

```
Ürün Adı: Blender Seti, Fiyat: 1800.00 TL
Ürün Adı: Kahve Makinesi, Fiyat: 1200.00 TL
Ürün Adı: Kablosuz Kulaklık, Fiyat: 950.00 TL
```

Ödev 9: Set Operatörleri (`Any`)

Senaryo: Hiç sipariş vermemiş müşterileri tespit edip onlara özel bir kampanya maili göndermek istiyoruz.
İstenenler:

1. `customers` listesi üzerinde, `orders` listesinde kendilerine ait (`CustomerId`) hiçbir kaydın (`Any`) olmadığı müşterileri filtreleyin.
2. Bulunan müşterilerin adını ve şehrini ekrana yazdırın.
   İpucu: `customers.Where(c => !orders.Any(o => o.CustomerId == c.Id))` gibi bir yapı kullanabilirsiniz. `Any` metodu, bir koleksiyonda belirli bir koşulu sağlayan en az bir eleman olup olmadığını kontrol eder. `!` (değil) operatörü ile bu durumu tersine çeviriyoruz.
   Beklenen Çıktı:

```
Hiç Sipariş Vermemiş Müşteri: Fatma Öztürk (Bursa)
```

Ödev 10: Karmaşık Sorgu (Belirli Bir Ayda En Çok Satan Ürün)

Senaryo: Ekim (10. ay) 2025'te toplamda en çok satan (adet bazında) ürünün hangisi olduğunu bulmak istiyoruz.
İstenenler:

1. `orders` listesinden sadece Ekim 2025'te verilen siparişleri filtreleyin (`Where`).
2. Bu filtrelenmiş siparişleri `ProductId`'ye göre gruplayın (`GroupBy`).
3. Her bir ürün grubu için toplam satış adedini (`Sum(o => o.Quantity)`) hesaplayın.
4. Bu sonuçları toplam adede göre azalan şekilde sıralayıp (`OrderByDescending`) en üsttekini alın (`First`).
5. En çok satan ürünün ID'sini kullanarak `products` listesinden ürünün adını bulun ve ekrana toplam satış adediyle birlikte yazdırın.
   İpucu: Bu da birkaç adımlı bir sorgudur. Son adımda bulduğunuz `ProductId` ile `products` listesinde bir `First()` sorgusu daha yapmanız gerekecek.
   Beklenen Çıktı:

```
Ekim 2025'in En Çok Satan Ürünü: Roman Kitabı, Toplam Satış: 3 adet
```