LINQ Alıştırmaları ve Ödevleri-1 Çözümleri

Ödev 1: Temel Filtreleme ve Sıralama

```
var filteredAndSortedProducts = products
    .Where(p => p.Price < 1000 && p.StockQuantity > 50)
    .OrderByDescending(p => p.Price);

foreach (var product in filteredAndSortedProducts)
{
    Console.WriteLine($"Ürün Adı: {product.Name}, Fiyat: {product.Price:F2} TL,
Stok: {product.StockQuantity}");
}
```

Ödev 2: Projeksiyon (Veri Şekillendirme)

```
var projectedProducts = products.Select(p => new
{
    ProductName = p.Name,
    PriceWithVat = p.Price * 1.18m,
    StockStatus = p.StockQuantity < 20 ? "Kritik Stok" : "Stok Yeterli"
});

foreach (var p in projectedProducts)
{
    Console.WriteLine($"Product Name: {p.ProductName}, Price with VAT:
{p.PriceWithVat:F2}, Stock Status: {p.StockStatus}");
}
```

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

```
var mostExpensiveProductsByCategory = products
    .GroupBy(p => p.Category)
    .Select(g =>
    {
        var mostExpensive = g.OrderByDescending(p => p.Price).First();
        return new
        {
            Category = g.Key,
            ProductName = mostExpensive.Name,
            Price = mostExpensive.Price
        };
    });

foreach (var item in mostExpensiveProductsByCategory)
{
    Console.WriteLine($"Kategori: {item.Category}, En Pahalı Ürün:
{item.ProductName} ({item.Price:F2} TL)");
}
```

Ödev 4: Üç Listeyi Birleştirme (Join)

```
var comprehensiveOrderReport = orders
    .Join(customers,
          order => order.CustomerId,
          customer => customer.Id,
          (order, customer) => new { order, customer })
    .Join(products,
          oc => oc.order.ProductId,
          product => product.Id,
          (oc, product) => new
          {
              CustomerName = oc.customer.FullName,
              City = oc.customer.City,
              ProductName = product.Name,
              Quantity = oc.order.Quantity,
              Price = product.Price,
              OrderDate = oc.order.OrderDate
          });

foreach (var reportItem in comprehensiveOrderReport)
{
    Console.WriteLine($"Müşteri: {reportItem.CustomerName} ({reportItem.City}),
Ürün: {reportItem.ProductName}, Adet: {reportItem.Quantity}, Fiyat:
{reportItem.Price:F2}, Tarih: {reportItem.OrderDate:dd.MM.yyyy}");
}
```

Ödev 5: Birleştirme ve Filtreleme (Join & Where)

```
var ankaraOrders = orders
    .Join(customers,
          o => o.CustomerId,
          c => c.Id,
          (o, c) => new { Order = o, Customer = c })
    .Where(x => x.Customer.City == "Ankara")
    .Select(x => new
    {
        OrderId = x.Order.OrderId,
        FullName = x.Customer.FullName,
        OrderDate = x.Order.OrderDate
    });

foreach (var order in ankaraOrders)
{
    Console.WriteLine($"Sipariş ID: {order.OrderId}, Müşteri: {order.FullName},
Tarih: {order.OrderDate:dd.MM.yyyy}");
}
```

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

```
var topCustomer = orders
    .Join(products, o => o.ProductId, p => p.Id, (o, p) => new { o.CustomerId,
TotalPrice = o.Quantity * p.Price })
```

```
    .GroupBy(x => x.CustomerId)
    .Select(g => new
    {
        CustomerId = g.Key,
        TotalSpending = g.Sum(x => x.TotalPrice)
    })
    .OrderByDescending(x => x.TotalSpending)
    .First();

var customerInfo = customers.First(c => c.Id == topCustomer.CustomerId);

Console.WriteLine($"En Çok Harcama Yapan Müşteri: {customerInfo.FullName},
Toplam Harcama: {topCustomer.TotalSpending:F2} TL");
```

Ödev 7: Eleman Operatörleri (FirstOrDefault, SingleOrDefault)

```
// ID'si 6 olan ürün için
var product6 = products.FirstOrDefault(p => p.Id == 6);
if (product6 != null)
{
    Console.WriteLine($"Ürün Adı: {product6.Name}, Kategori:
{product6.Category}");
}
else
{
    Console.WriteLine("ID'si 6 olan ürün bulunamadı.");
}

// ID'si 99 olan ürün için
var product99 = products.FirstOrDefault(p => p.Id == 99);
if (product99 != null)
{
    Console.WriteLine($"Ürün Adı: {product99.Name}, Kategori:
{product99.Category}");
}
else
{
    Console.WriteLine("ID'si 99 olan ürün bulunamadı.");
}
```

Ödev 8: Sayfalama Operatörleri (Skip & Take)

```
var pagedProducts = products
    .OrderByDescending(p => p.Price)
    .Skip(3)
    .Take(3);

foreach (var product in pagedProducts)
{
    Console.WriteLine($"Ürün Adı: {product.Name}, Fiyat: {product.Price:F2}
TL");
}
```

Ödev 9: Set Operatörleri (Any)

```
var customersWithNoOrders = customers
    .Where(c => !orders.Any(o => o.CustomerId == c.Id));

foreach (var customer in customersWithNoOrders)
{
    Console.WriteLine($"Hiç Sipariş Vermemiş Müşteri: {customer.FullName}
({customer.City})");
}
```

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

```
var topSellingProductInOctober = orders
    .Where(o => o.OrderDate.Year == 2025 && o.OrderDate.Month == 10)
    .GroupBy(o => o.ProductId)
    .Select(g => new
    {
        ProductId = g.Key,
        TotalQuantity = g.Sum(o => o.Quantity)
    })
    .OrderByDescending(x => x.TotalQuantity)
    .First();

var productInfo = products.First(p => p.Id ==
topSellingProductInOctober.ProductId);

Console.WriteLine($"Ekim 2025'in En Çok Satan Ürünü: {productInfo.Name}, Toplam
Satış: {topSellingProductInOctober.TotalQuantity} adet");
```

LINQ Alıştırmaları ve Ödevleri-2 Çözümleri

Ödev 1: “Ana Sayfa Akışı” Simülasyonu

```
int currentUserId = 1;

var followedUserIds = follows
    .Where(f => f.FollowerId == currentUserId)
    .Select(f => f.FollowingId)
    .ToList();

var feed = posts
    .Where(p => followedUserIds.Contains(p.UserId))
    .OrderByDescending(p => p.Timestamp)
    .Join(users, p => p.UserId, u => u.Id, (p, u) => new { PostContent =
p.Content, Username = u.Username });

foreach (var item in feed)
{
    Console.WriteLine($"Kullanıcı: {item.Username}, Gönderi:
{item.PostContent}");
```

```
}
```

Ödev 2: En Popüler Gönderiyi Bulma (Yoruma Göre)

```
var mostPopularPost = comments
    .GroupBy(c => c.PostId)
    .Select(g => new { PostId = g.Key, CommentCount = g.Count() })
    .OrderByDescending(x => x.CommentCount)
    .First();

var postInfo = posts.First(p => p.Id == mostPopularPost.PostId);

Console.WriteLine($"En Popüler Gönderi: \"{postInfo.Content}\"
({mostPopularPost.CommentCount} yorum)");
```

Ödev 3: Etkileşim Raporu (Kullanıcı Bazında)

```
var interactionReport = users
    .Select(u => new
    {
        Username = u.Username,
        PostCount = posts.Count(p => p.UserId == u.Id),
        CommentCount = comments.Count(c => c.UserId == u.Id)
    })
    .OrderByDescending(r => r.PostCount);

foreach (var report in interactionReport)
{
    Console.WriteLine($"Kullanıcı: {report.Username}, Gönderi Sayısı:
{report.PostCount}, Yorum Sayısı: {report.CommentCount}");
}
```

Ödev 4: “Influencer” Tespiti (Takipçiye Göre)

```
var influencer = follows
    .GroupBy(f => f.FollowingId)
    .Select(g => new { UserId = g.Key, FollowerCount = g.Count() })
    .OrderByDescending(x => x.FollowerCount)
    .First();

var userInfo = users.First(u => u.Id == influencer.UserId);

Console.WriteLine($"En Çok Takip Edilen Kullanıcı: {userInfo.Username}
({influencer.FollowerCount} takipçi)");
```

Ödev 5: Karşılıklı Takipleşme (“Arkadaşlar”)

```
var mutuals = follows
    .Where(f1 => f1.FollowerId < f1.FollowingId) // Tekrarları önlemek için
    .Where(f1 => follows.Any(f2 => f2.FollowerId == f1.FollowingId &&
f2.FollowingId == f1.FollowerId))
```

```
    .Join(users, f => f.FollowerId, u => u.Id, (f, u) => new { f, FollowerName = u.Username })
    .Join(users, f_u => f_u.f.FollowingId, u => u.Id, (f_u, u) => new { f_u.FollowerName, FollowingName = u.Username });

foreach (var mutual in mutuals)
{
    Console.WriteLine($"Karşılıklı Takipleşenler: {mutual.FollowerName} <-> {mutual.FollowingName}");
}
```

Ödev 6: "Pasif" Kullanıcılar

```
var passiveUsers = users
    .Where(u => !posts.Any(p => p.UserId == u.Id));

foreach (var user in passiveUsers)
{
    Console.WriteLine($"Hiç Gönderi Paylaşmamış Kullanıcı: {user.Username}, Üyelik Tarihi: {user.JoinDate:dd.MM.yyyy}");
}
```

Ödev 7: Şehirlere Göre Aktivite

```
var activityByCity = posts
    .Join(users, p => p.UserId, u => u.Id, (p, u) => new { u.City })
    .GroupBy(x => x.City)
    .Select(g => new { City = g.Key, PostCount = g.Count() })
    .OrderByDescending(x => x.PostCount);

foreach (var cityActivity in activityByCity)
{
    Console.WriteLine($"Şehir: {cityActivity.City}, Toplam Gönderi: {cityActivity.PostCount}");
}
```

Ödev 8: Gönderi ve Yorumları Birleştirme

```
int targetPostId = 102;
var post = posts.FirstOrDefault(p => p.Id == targetPostId);

if (post != null)
{
    Console.WriteLine($"Gönderi: {post.Content}");
    Console.WriteLine("--- Yorumlar ---");

    var postComments = comments
        .Where(c => c.PostId == targetPostId)
        .Join(users, c => c.UserId, u => u.Id, (c, u) => new { Username = u.Username, CommentContent = c.Content });

    foreach (var comment in postComments)
```

```
    {
        Console.WriteLine($"Yorum Yapan: {comment.Username}, Yorum:
{comment.CommentContent}");
    }
}
```

Ödev 9: Hashtag Analizi

```
string hashtag = "#gündem";

var postIdsWithHashtag = posts
    .Where(p => p.Content.Contains(hashtag))
    .Select(p => p.Id)
    .ToList();

var commentsOnHashtaggedPosts = comments
    .Where(c => postIdsWithHashtag.Contains(c.PostId));

Console.WriteLine($"\"{hashtag}\" Konulu Gönderilere Yapılan Yorumlar:");
foreach (var comment in commentsOnHashtaggedPosts)
{
    Console.WriteLine($"- {comment.Content}");
}
```

Ödev 10: En Son Etkileşim

Adım 1: Modeli Güncelleme Öncelikle `Comment` sınıfına `Timestamp` özelliği eklenmelidir.

```
public class Comment
{
    public int Id { get; set; }
    public int PostId { get; set; }
    public int UserId { get; set; }
    public string Content { get; set; }
    public DateTime Timestamp { get; set; } // Eklenen özellik
}
```

Adım 2: Veri Setini Güncelleme `comments` listesi, yeni `Timestamp` özelliği ile güncellenmelidir.

```
List<Comment> comments = new List<Comment>
{
    new Comment { Id = 201, PostId = 102, UserId = 1, Content = "Başarılar
dilerim Zeynep!", Timestamp = DateTime.Now.AddDays(-4) },
    new Comment { Id = 202, PostId = 102, UserId = 4, Content = "Merakla
bekliyoruz!", Timestamp = DateTime.Now.AddDays(-3) },
    new Comment { Id = 203, PostId = 103, UserId = 2, Content = "Kesinlikle
katılıyorum Ahmet.", Timestamp = DateTime.Now.AddHours(-4) },
    new Comment { Id = 204, PostId = 101, UserId = 3, Content = "Gerçekten de
öyle.", Timestamp = DateTime.Now.AddDays(-1) },
    new Comment { Id = 205, PostId = 104, UserId = 2, Content = "Ankara'yı
özledim...", Timestamp = DateTime.Now.AddHours(-12) }
};
```

Adım 3: LINQ Sorgusu

```
var lastPostTime = posts.Max(p => p.Timestamp);
var lastCommentTime = comments.Max(c => c.Timestamp);

if (lastPostTime > lastCommentTime)
{
    Console.WriteLine($"Platformdaki son aktivite {lastPostTime} tarihinde bir gönderi idi.");
}
else
{
    Console.WriteLine($"Platformdaki son aktivite {lastCommentTime} tarihinde bir yorum idi.");
}
```