1:

```
var outOfStockProducts = from p in ListGenerators.Products
                         where p.UnitsInStock == 0
                         select p;
var expensiveInStock = from p in ListGenerators.Products
                       where p.UnitsInStock > 0 && p.UnitPrice > 3.00M
                       select p;
string[] Arr = { "zero", "one", "two", "three", "four",
                 "five", "six", "seven", "eight", "nine" };

var shortNames = from num in Arr.Select((name, index) => new { name, index })
                 where name.Length < index
                 select name;
```

2:

```
var firstOutOfStock = ListGenerators.Products
                      .First(p => p.UnitsInStock == 0);
var expensiveProduct = ListGenerators.Products
                       .FirstOrDefault(p => p.UnitPrice > 1000);
int[] Arr = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };

var secondGreaterThanFive = Arr.Where(n => n > 5)
                            .ElementAt(1);
```

3:

```
int[] Arr = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };
int oddCount = Arr.Count(n => n % 2 != 0);
Console.WriteLine($"عدد الأعداد الفردية = {oddCount}");
```

4:

```
var customerOrdersCount = from c in ListGenerators.Customers
                          select new
                          {
                            c.CustomerName,
                            OrdersCount = c.Orders.Count()
                          };

foreach (var c in customerOrdersCount)
   Console.WriteLine($"{c.CustomerName} - {c.OrdersCount}");
var categoryProductsCount = from p in ListGenerators.Products
                            group p by p.Category into g
                            select new
                            {
                              Category = g.Key,
                              ProductsCount = g.Count()
                            };

foreach (var c in categoryProductsCount)
   Console.WriteLine($"{c.Category} - {c.ProductsCount}");
int[] Arr = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };

int total = Arr.Sum();

Console.WriteLine($"مجموع الأعداد = {total}");
string[] words = File.ReadAllLines("dictionary_english.txt");
```

```
int totalChars = words.Sum(w => w.Length);

Console.WriteLine($"إجمالي عدد الحروف = {totalChars}");
```

6:

```
var totalUnitsPerCategory = from p in ListGenerators.Products
                            group p by p.Category into g
                            select new
                            {
                               Category = g.Key,
                               TotalUnits = g.Sum(p => p.UnitsInStock)
                            };

foreach (var c in totalUnitsPerCategory)
   Console.WriteLine($"{c.Category} - {c.TotalUnits}");

string[] words = File.ReadAllLines("dictionary_english.txt");

int shortestWordLength = words.Min(w => w.Length);

Console.WriteLine($"أقصر كلمة طولها = {shortestWordLength}");

var cheapestPricePerCategory = from p in ListGenerators.Products
                               group p by p.Category into g
                               select new
                               {
```

```
                    Category = g.Key,
                    MinPrice = g.Min(p => p.UnitPrice)
                };

foreach (var c in cheapestPricePerCategory)
    Console.WriteLine($"{c.Category} - {c.MinPrice}");

var cheapestProducts = from p in ListGenerators.Products
                group p by p.Category into g
                let minPrice = g.Min(p => p.UnitPrice)
                from p in g
                where p.UnitPrice == minPrice
                select new
                {
                    Category = g.Key,
                    Product = p.ProductName,
                    Price = p.UnitPrice
                };

foreach (var item in cheapestProducts)
    Console.WriteLine($"{item.Category} - {item.Product} - {item.Price}");
int longestWordLength = words.Max(w => w.Length);

Console.WriteLine($"أطول كلمة طولها = {longestWordLength}");

var mostExpensivePricePerCategory = from p in ListGenerators.Products
```

```
                    group p by p.Category into g
                    select new
                    {
                        Category = g.Key,
                        MaxPrice = g.Max(p => p.UnitPrice)
                    };

foreach (var c in mostExpensivePricePerCategory)
    Console.WriteLine($"{c.Category} - {c.MaxPrice}");
var mostExpensiveProducts = from p in ListGenerators.Products
                            group p by p.Category into g
                            let maxPrice = g.Max(p => p.UnitPrice)
                            from p in g
                            where p.UnitPrice == maxPrice
                            select new
                            {
                                Category = g.Key,
                                Product = p.ProductName,
                                Price = p.UnitPrice
                            };

foreach (var item in mostExpensiveProducts)
    Console.WriteLine($"{item.Category} - {item.Product} - {item.Price}");

double avgWordLength = words.Average(w => w.Length);
```

```
Console.WriteLine($"متوسط طول الكلمات = {avgWordLength}");

var avgPricePerCategory = from p in ListGenerators.Products
                          group p by p.Category into g
                          select new
                          {
                             Category = g.Key,
                             AvgPrice = g.Average(p => p.UnitPrice)
                          };

foreach (var c in avgPricePerCategory)
   Console.WriteLine($"{c.Category} - {c.AvgPrice}");
```

```
var productsByName = from p in ListGenerators.Products
                     orderby p.ProductName
                     select p;

foreach (var p in productsByName)
   Console.WriteLine(p.ProductName);

string[] Arr = { "aPPLE", "AbAcUs", "bRaNcH", "BlUeBeRrY", "ClOvEr", "cHeRry" };

var caseInsensitiveSort = Arr.OrderBy(w => w, StringComparer.OrdinalIgnoreCase);

foreach (var w in caseInsensitiveSort)
   Console.WriteLine(w);
```

```
var productsByStock = from p in ListGenerators.Products
                      orderby p.UnitsInStock descending
                      select p;

foreach (var p in productsByStock)
   Console.WriteLine($"{p.ProductName} - {p.UnitsInStock}");

string[] Arr = { "zero", "one", "two", "three", "four",
                 "five", "six", "seven", "eight", "nine" };

var digitsSorted = from d in Arr
                   orderby d.Length, d
                   select d;

foreach (var d in digitsSorted)
   Console.WriteLine(d);

string[] words = { "aPPLE", "AbAcUs", "bRaNcH", "BlUeBeRrY", "ClOvEr", "cHeRry" };

var wordsSorted = words.OrderBy(w => w.Length)
                       .ThenBy(w => w, StringComparer.OrdinalIgnoreCase);

foreach (var w in wordsSorted)
   Console.WriteLine(w);

var productsByCategoryPrice = from p in ListGenerators.Products
```

```
                    orderby p.Category, p.UnitPrice descending
                    select p;

foreach (var p in productsByCategoryPrice)
   Console.WriteLine($"{p.Category} - {p.ProductName} - {p.UnitPrice}");

string[] Arr = { "aPPLE", "AbAcUs", "bRaNcH", "BlUeBeRrY", "ClOvEr", "cHeRry" };

var wordsSortedDesc = Arr.OrderBy(w => w.Length)
                .ThenByDescending(w => w, StringComparer.OrdinalIgnoreCase);

foreach (var w in wordsSortedDesc)
   Console.WriteLine(w);

string[] Arr = { "zero", "one", "two", "three", "four",
            "five", "six", "seven", "eight", "nine" };

string[] Arr = { "zero", "one", "two", "three", "four",
            "five", "six", "seven", "eight", "nine" };

var digitsWithSecondI = (from d in Arr
                where d.Length > 1 && d[1] == 'i'
                select d).Reverse();

foreach (var d in digitsWithSecondI)
   Console.WriteLine(d);
```

```
var productNames = from p in ListGenerators.Products
                   select p.ProductName;

foreach (var name in productNames)
   Console.WriteLine(name);

string[] words = { "aPPLE", "BlUeBeRrY", "cHeRry" };

var upperLowerWords = from w in words
                      select new { Upper = w.ToUpper(), Lower = w.ToLower() };

foreach (var item in upperLowerWords)
   Console.WriteLine($"Upper: {item.Upper}, Lower: {item.Lower}");

var productsWithPrice = from p in ListGenerators.Products
                        select new
                        {
                           p.ProductName,
                           Price = p.UnitPrice,
                           p.Category
                        };

foreach (var item in productsWithPrice)
   Console.WriteLine($"{item.ProductName} - {item.Category} - {item.Price}");

int[] Arr = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };
```

```
var numberInPlace = Arr.Select((num, index) => new
{
   Number = num,
   InPlace = (num == index)
});

foreach (var item in numberInPlace)
   Console.WriteLine($"{item.Number}: {item.InPlace}");

int[] numbersA = { 0, 2, 4, 5, 6, 8, 9 };
int[] numbersB = { 1, 3, 5, 7, 8 };

var pairs = from a in numbersA
        from b in numbersB
        where a < b
        select new { A = a, B = b };

Console.WriteLine("Pairs where a < b:");
foreach (var p in pairs)
   Console.WriteLine($"{p.A} is less than {p.B}");
```

```
int[] Arr = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };

int oddCount = Arr.Count(n => n % 2 != 0);
```

```
Console.WriteLine($"عدد الأعداد الفردية = {oddCount}");

var customerOrdersCount = from c in ListGenerators.Customers
                          select new
                          {
                            c.CustomerName,
                            OrdersCount = c.Orders.Count()
                          };

foreach (var c in customerOrdersCount)
  Console.WriteLine($"{c.CustomerName} - {c.OrdersCount}");

var categoryProductsCount = from p in ListGenerators.Products
                            group p by p.Category into g
                            select new
                            {
                              Category = g.Key,
                              ProductsCount = g.Count()
                            };

foreach (var c in categoryProductsCount)
  Console.WriteLine($"{c.Category} - {c.ProductsCount}");

string[] words = File.ReadAllLines("dictionary_english.txt");

int totalChars = words.Sum(w => w.Length);
```

```
Console.WriteLine($"إجمالي عدد الحروف = {totalChars}");

var totalUnitsPerCategory = from p in ListGenerators.Products
                            group p by p.Category into g
                            select new
                            {
                               Category = g.Key,
                               TotalUnits = g.Sum(p => p.UnitsInStock)
                            };

foreach (var c in totalUnitsPerCategory)
   Console.WriteLine($"{c.Category} - {c.TotalUnits}");

int shortestWordLength = words.Min(w => w.Length);
Console.WriteLine($"أقصر كلمة طولها = {shortestWordLength}");

var cheapestPricePerCategory = from p in ListGenerators.Products
                               group p by p.Category into g
                               select new
                               {
                                  Category = g.Key,
                                  MinPrice = g.Min(p => p.UnitPrice)
                               };

foreach (var c in cheapestPricePerCategory)
```

```
   Console.WriteLine($"{c.Category} - {c.MinPrice}");

var cheapestProducts = from p in ListGenerators.Products
                       group p by p.Category into g
                       let minPrice = g.Min(p => p.UnitPrice)
                       from p in g
                       where p.UnitPrice == minPrice
                       select new
                       {
                          Category = g.Key,
                          Product = p.ProductName,
                          Price = p.UnitPrice
                       };

foreach (var item in cheapestProducts)
   Console.WriteLine($"{item.Category} - {item.Product} - {item.Price}");

int longestWordLength = words.Max(w => w.Length);
Console.WriteLine($"أطول كلمة طولها = {longestWordLength}");

var mostExpensivePricePerCategory = from p in ListGenerators.Products
                                    group p by p.Category into g
                                    select new
                                    {
                                       Category = g.Key,
                                       MaxPrice = g.Max(p => p.UnitPrice)
```

```
                    };

foreach (var c in mostExpensivePricePerCategory)
   Console.WriteLine($"{c.Category} - {c.MaxPrice}");

var mostExpensiveProducts = from p in ListGenerators.Products
                    group p by p.Category into g
                    let maxPrice = g.Max(p => p.UnitPrice)
                    from p in g
                    where p.UnitPrice == maxPrice
                    select new
                    {
                       Category = g.Key,
                       Product = p.ProductName,
                       Price = p.UnitPrice
                    };

foreach (var item in mostExpensiveProducts)
   Console.WriteLine($"{item.Category} - {item.Product} - {item.Price}");

double avgWordLength = words.Average(w => w.Length);
Console.WriteLine($"متوسط طول الكلمات = {avgWordLength}");

var avgPricePerCategory = from p in ListGenerators.Products
                    group p by p.Category into g
                    select new
```

```
                {
                    Category = g.Key,
                    AvgPrice = g.Average(p => p.UnitPrice)
                };

foreach (var c in avgPricePerCategory)
    Console.WriteLine($"{c.Category} - {c.AvgPrice}");
```

```
var productsByName = from p in ListGenerators.Products
                     orderby p.ProductName
                     select p;

foreach (var p in productsByName)
    Console.WriteLine(p.ProductName);

string[] Arr = { "aPPLE", "AbAcUs", "bRaNcH", "BlUeBeRrY", "ClOvEr", "cHeRry" };

var caseInsensitiveSort = Arr.OrderBy(w => w, StringComparer.OrdinalIgnoreCase);

foreach (var w in caseInsensitiveSort)
    Console.WriteLine(w);

var productsByStock = from p in ListGenerators.Products
                      orderby p.UnitsInStock descending
                      select p;

foreach (var p in productsByStock)
```

```
   Console.WriteLine($"{p.ProductName} - {p.UnitsInStock}");


string[] Arr = { "zero", "one", "two", "three", "four",
                 "five", "six", "seven", "eight", "nine" };


var digitsSorted = from d in Arr
                   orderby d.Length, d
                   select d;


foreach (var d in digitsSorted)
   Console.WriteLine(d);


string[] Arr = { "aPPLE", "AbAcUs", "bRaNcH", "BlUeBeRrY", "ClOvEr", "cHeRry" };


var wordsSorted = Arr.OrderBy(w => w.Length)
                     .ThenBy(w => w, StringComparer.OrdinalIgnoreCase);


foreach (var w in wordsSorted)
   Console.WriteLine(w);


var productsByCategoryPrice = from p in ListGenerators.Products
                              orderby p.Category, p.UnitPrice descending
                              select p;


foreach (var p in productsByCategoryPrice)
   Console.WriteLine($"{p.Category} - {p.ProductName} - {p.UnitPrice}");
```

```
string[] Arr = { "aPPLE", "AbAcUs", "bRaNcH", "BlUeBeRrY", "ClOvEr", "cHeRry" };

var wordsSortedDesc = Arr.OrderBy(w => w.Length)
                  .ThenByDescending(w => w, StringComparer.OrdinalIgnoreCase);

foreach (var w in wordsSortedDesc)
   Console.WriteLine(w);
```

```
var first3OrdersWA = (from c in ListGenerators.Customers
                 where c.Region == "WA"
                 from o in c.Orders
                 select o)
                .Take(3);

foreach (var o in first3OrdersWA)
   Console.WriteLine($"OrderID: {o.OrderID}, Date: {o.OrderDate}");

var skip2OrdersWA = (from c in ListGenerators.Customers
                 where c.Region == "WA"
                 from o in c.Orders
                 select o)
                .Skip(2);

foreach (var o in skip2OrdersWA)
   Console.WriteLine($"OrderID: {o.OrderID}, Date: {o.OrderDate}");
```

```
int[] numbers = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };

var result = numbers.TakeWhile((num, index) => num >= index);

foreach (var n in result)
   Console.WriteLine(n);

var result = numbers.SkipWhile(n => n % 3 != 0);

foreach (var n in result)
   Console.WriteLine(n);

var result = numbers.SkipWhile((num, index) => num >= index);

foreach (var n in result)
   Console.WriteLine(n);
```

```
var first3OrdersWA = (from c in ListGenerators.Customers
                      where c.Region == "WA"
                      from o in c.Orders
                      select o)
                     .Take(3);

foreach (var o in first3OrdersWA)
   Console.WriteLine($"OrderID: {o.OrderID}, Date: {o.OrderDate}");

var skip2OrdersWA = (from c in ListGenerators.Customers
```

```
            where c.Region == "WA"
            from o in c.Orders
            select o)
           .Skip(2);

foreach (var o in skip2OrdersWA)
   Console.WriteLine($"OrderID: {o.OrderID}, Date: {o.OrderDate}");

int[] numbers = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };

var result = numbers.TakeWhile((num, index) => num >= index);

foreach (var n in result)
   Console.WriteLine(n);

var result = numbers.SkipWhile(n => n % 3 != 0);

foreach (var n in result)
   Console.WriteLine(n);

var result = numbers.SkipWhile((num, index) => num >= index);

foreach (var n in result)
   Console.WriteLine(n);
```

```
string[] words = File.ReadAllLines("dictionary_english.txt");
```

```
bool hasEi = words.Any(w => w.Contains("ei"));

Console.WriteLine(hasEi
   ? "يوجد كلمات تحتوي على 'ei'"
   : "لا يوجد كلمات تحتوي على 'ei'");

var categoriesWithOutOfStock = from p in ListGenerators.Products
                    group p by p.Category into g
                    where g.Any(p => p.UnitsInStock == 0)
                    select new
                    {
                       Category = g.Key,
                       Products = g
                    };

foreach (var c in categoriesWithOutOfStock)
{
   Console.WriteLine($"Category: {c.Category}");
   foreach (var p in c.Products)
      Console.WriteLine($"   {p.ProductName} - {p.UnitsInStock}");
}

var categoriesAllInStock = from p in ListGenerators.Products
                  group p by p.Category into g
                  where g.All(p => p.UnitsInStock > 0)
                  select new
```

```
                {
                    Category = g.Key,
                    Products = g
                };

foreach (var c in categoriesAllInStock)
{
    Console.WriteLine($"Category: {c.Category}");
    foreach (var p in c.Products)
        Console.WriteLine($"   {p.ProductName} - {p.UnitsInStock}");
}
```