LinQ (Language Integrated Query)

1. What is LINQ?

LINQ (**Language Integrated Query**) is a set of features in C# that lets you query data in a **type-safe**, **readable**, and **declarative** way — directly inside C#.
It works with in-memory collections, EF Core database queries, XML, JSON, etc.

Instead of writing SQL:

```
SELECT * FROM Books WHERE Price > 20 ORDER BY Title
```

You can write:

```
var result = _context.Books
   .Where(b => b.Price > 20)
   .OrderBy(b => b.Title)
   .ToList();
```

---

2. LINQ Syntax Styles

- **Query Syntax**: SQL-like

```
var query = from b in _context.Books
        where b.Price > 20
        orderby b.Title
        select b;
```

- **Method Syntax**: Fluent API

```
var query = _context.Books
   .Where(b => b.Price > 20)
   .OrderBy(b => b.Title);
```

Both produce the same result — method syntax is more common with EF Core.

---

3. Core LINQ Operators (Categories)

Category	Examples	Purpose
Filtering	Where, OfType	Narrow down data
Sorting	OrderBy, OrderByDescending, ThenBy	Sort data

Category	Examples	Purpose
Projection	Select, SelectMany	Transform data
Joining	Join, GroupJoin	Combine sequences
Grouping	GroupBy	Group elements
Aggregation	Count, Sum, Average, Min, Max	Aggregate values
Quantifiers	Any, All, Contains	Boolean checks
Element	First, FirstOrDefault, SingleOrDefault	Get one element
Set	Distinct, Union, Intersect, Except	Set operations
Generation	Range, Repeat	Create sequences
Partitioning	Skip, Take, SkipWhile, TakeWhile	Take a part of data (pagination)

---

4. Applied Examples in Service Layer

```
public class BookstoreService
{
  private readonly BookstoreRepository _repo;
  public BookstoreService(BookstoreRepository repo)
  {
    _repo = repo;
  }
  // Example 1: Filtering
  public IEnumerable<Book> GetBooksAbovePrice(decimal price)
  {
    return _repo.GetAllBooks()
          .Where(b => b.Price > price)
          .OrderBy(b => b.Title);
  }
```

```
    // Example 2: Projection
    public IEnumerable<string> GetAllBookTitles()
    {
        return _repo.GetAllBooks()
                    .Select(b => b.Title);
    }

    // Example 3: Join
  public IEnumerable<BookAuthorDto> GetBooksWithAuthors()
{
    return _repo.GetAllBooks()
        .Join(_repo.GetAllAuthors(),
            b => b.AuthorId,
            a => a.AuthorId,
            (b, a) => new BookAuthorDto { Title = b.Title, Name = a.Name });
}
    // Example 4: Grouping
public IEnumerable<BooksByAuthorDto> GetBooksGroupedByAuthor()
{
    var authors = _repo.GetAllAuthors().ToList();
    var books = _repo.GetAllBooks();

    var grouped = books
        .GroupBy(b => b.AuthorId)
        .Select(g =>
        {
            var author = authors.FirstOrDefault(a => a.AuthorId == g.Key);
            return new BooksByAuthorDto
            {
                AuthorId = g.Key,
```

```
            AuthorName = author?.Name ?? "Unknown",
            BookTitles = g.Select(b => b.Title).ToList(),
            BookCount = g.Count()
        };
    });

    return grouped;
}
    // Example 5: Aggregation
    public decimal GetTotalStockValue()
    {
        return _repo.GetAllBooks()
                .Sum(b => b.Price * b.Stock);
    }

    // Example 6: Quantifiers
    public bool AnyBooksOutOfStock()
    {
        return _repo.GetAllBooks().Any(b => b.Stock == 0);
    }

    // Example 7: Element operators
    public Book GetMostExpensiveBook()
    {
        return _repo.GetAllBooks()
                .OrderByDescending(b => b.Price)
                .FirstOrDefault();
    }
```

```
    // Example 8: Complex Join (Orders + Customers + Books)
    public IEnumerable<object> GetOrderDetails()
    {
        return _repo.GetAllOrders()
                .Join(_repo.GetAllCustomers(),
                    o => o.CustomerId,
                    c => c.CustomerId,
                    (o, c) => new { o, c })
                .Join(_repo.GetAllOrderItems(),
                    oc => oc.o.OrderId,
                    oi => oi.OrderId,
                    (oc, oi) => new { oc, oi })
                .Join(_repo.GetAllBooks(),
                    oci => oci.oi.BookId,
                    b => b.BookId,
                    (oci, b) => new
                    {
                        oci.oc.c.Name,
                        b.Title,
                        oci.oi.Quantity,
                        TotalPrice = b.Price * oci.oi.Quantity
                    });
    }
}
```

```
// Distinct
public IEnumerable<string> GetUniqueAuthorCountries()
{
    return _repo.GetAllAuthors()
            .Select(a => a.Country)
            .Distinct();
```

```
}

// Union
public IEnumerable<string> GetBookTitlesAndAuthorNames()
{
   var bookTitles = _repo.GetAllBooks().Select(b => b.Title);
   var authorNames = _repo.GetAllAuthors().Select(a => a.Name);
   return bookTitles.Union(authorNames);
}

// Intersect
public IEnumerable<Book> GetBooksAbove20AndInStock()
{
   var expensiveBooks = _repo.GetAllBooks().Where(b => b.Price > 20);
   var inStockBooks = _repo.GetAllBooks().Where(b => b.Stock > 0);
   return expensiveBooks.Intersect(inStockBooks);
}

//  Except
public IEnumerable<Book> GetBooksInStockOnly()
{
   var allBooks = _repo.GetAllBooks();
   var outOfStock = _repo.GetAllBooks().Where(b => b.Stock == 0);
   return allBooks.Except(outOfStock);
}
// Take
public IEnumerable<Book> GetTop5CheapestBooks()
{
   return _repo.GetAllBooks()
            .OrderBy(b => b.Price)
            .Take(5);}}
```

```
// Skip
public IEnumerable<Book> SkipTopSellingBook()
{
   return _repo.GetAllBooks()
            .OrderByDescending(b => b.Stock)
            .Skip(1);
}

//  TakeWhile
public IEnumerable<Book> GetBooksWhilePriceBelow30()
{
   return _repo.GetAllBooks()
            .OrderBy(b => b.Price)
            .TakeWhile(b => b.Price < 30);
}

// SkipWhile
public IEnumerable<Book> SkipWhilePriceBelow30()
{
   return _repo.GetAllBooks()
            .OrderBy(b => b.Price)
            .SkipWhile(b => b.Price < 30);
}
// ToList
public List<Book> GetBooksAsList()
{
   return _repo.GetAllBooks().ToList();
}
```