Lab 5: Retrieving Data from the Database

# Objective

Learn how to retrieve data from your SQL Server database using EF Core’s async methods: `ToListAsync`, `FindAsync`, and `FirstOrDefaultAsync`.

# Scenario

The retail store wants to display product details on the dashboard. You need to fetch all products, find a product by its ID, and get the first expensive product.

# Step-by-Step Implementation

## Code: Program.cs

using Microsoft.EntityFrameworkCore;  
using System;  
using System.Threading.Tasks;  
  
class Program  
{  
 static async Task Main(string[] args)  
 {  
 using var context = new AppDbContext();  
  
 // 1. Retrieve All Products  
 var products = await context.Products.ToListAsync();  
 Console.WriteLine("All Products:");  
 foreach (var p in products)  
 Console.WriteLine($"{p.Name} - ₹{p.Price}");  
  
 // 2. Find by ID  
 var product = await context.Products.FindAsync(1);  
 Console.WriteLine($"  
Found: {product?.Name}");  
  
 // 3. FirstOrDefault with Condition  
 var expensive = await context.Products.FirstOrDefaultAsync(p => p.Price > 50000);  
 Console.WriteLine($"  
Expensive: {expensive?.Name}");  
 }  
}

# Explanation

## 1. Retrieve All Products

var products = await context.Products.ToListAsync();  
foreach (var p in products)  
 Console.WriteLine($"{p.Name} - ₹{p.Price}");

\* `ToListAsync()` fetches all rows from the Products table as a list.

\* The `foreach` loop prints each product’s name and price.

## 2. Find by ID

var product = await context.Products.FindAsync(1);  
Console.WriteLine($"Found: {product?.Name}");

\* `FindAsync(1)` retrieves the product with primary key value 1.

\* The `?.` operator ensures no exception if the product is not found.

## 3. FirstOrDefault with Condition

var expensive = await context.Products.FirstOrDefaultAsync(p => p.Price > 50000);  
Console.WriteLine($"Expensive: {expensive?.Name}");

\* `FirstOrDefaultAsync` returns the first product with a price greater than ₹50,000, or `null` if none exist.

# Best Practices

\* Always use `await` with async EF Core methods to avoid blocking your application.

\* Use `FindAsync` for quick retrieval by primary key.

\* Use `FirstOrDefaultAsync` for conditional queries.

\* Always check for `null` when using `FirstOrDefaultAsync` or `FindAsync`.

# Summary Table

|  |  |  |
| --- | --- | --- |
| Method | Purpose | Example Usage |
| ToListAsync() | Get all rows from a table | await context.Products.ToListAsync() |
| FindAsync(key) | Find by primary key | await context.Products.FindAsync(1) |
| FirstOrDefaultAsync() | Get first row matching a condition | await context.Products.FirstOrDefaultAsync(p => p.Price > 50000) |

# References

\* Microsoft Docs: Querying Data - EF Core: https://learn.microsoft.com/en-us/ef/core/querying/

\* Microsoft Docs: Asynchronous Query Execution: https://learn.microsoft.com/en-us/ef/core/querying/async