Lab 1: Understanding ORM with a Retail Inventory System

Lab 2: Setting Up the Database Context for a Retail Stor

Lab 3: Using EF Core CLI to Create and Apply Migrations

**CODE:**

**Product.cs**

public class Product

{

    public int Id { get; set; }

    public string Name { get; set; }

    public decimal Price { get; set; }

    public int CategoryId { get; set; }

    public Category Category { get; set; }

    public int StockQuantity { get; set; }

}

**Category.cs**

public class Category

{

    public int Id { get; set; }

    public string Name { get; set; }

    public List<Product> Products { get; set; }

}

**AppDbContext.cs:**

using Microsoft.EntityFrameworkCore;

public class AppDbContext : DbContext

{

    public DbSet<Product> Products { get; set; }

    public DbSet<Category> Categories { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

    {

        optionsBuilder.UseSqlServer("Server=localhost\\sqlexpress;Database=RetailInventoryDb;Trusted\_Connection=True;Encrypt=True;TrustServerCertificate=True;");

    }

    protected override void OnModelCreating(ModelBuilder modelBuilder)

{

    // Category Seed

    modelBuilder.Entity<Category>().HasData(

        new Category { Id = 1, Name = "Electronics" },

        new Category { Id = 2, Name = "Groceries" }

    );

    // Product Seed

    modelBuilder.Entity<Product>().HasData(

        new Product { Id = 1, Name = "Smartphone", Price = 25000, StockQuantity = 50, CategoryId = 1 },

        new Product { Id = 2, Name = "Wheat Flour", Price = 800, StockQuantity = 100, CategoryId = 2 }

    );

}

}

**ProductDTO..cs:**

public class ProductDTO

{

    public string Name { get; set; }

    public decimal Price { get; set; }

}

**Program.cs:**

using System;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

class Program

{

    static async Task Main(string[] args)

    {

        using var context = new AppDbContext();

        Console.WriteLine("📦 All Products:");

        var products = await context.Products.ToListAsync();

        foreach (var p in products)

            Console.WriteLine($"🛒 {p.Name} - ₹{p.Price}");

        Console.WriteLine("\n🔎 Find Product by ID = 1:");

        var product = await context.Products.FindAsync(1);

        Console.WriteLine($"Found: {product?.Name}");

        Console.WriteLine("\n💰 Find Expensive Product (> ₹50,000):");

        var expensive = await context.Products.FirstOrDefaultAsync(p => p.Price > 50000);

        Console.WriteLine($"Expensive: {expensive?.Name}");

        // ✏️ UPDATE Laptop Price

        Console.WriteLine("\n✏️ Updating Laptop Price...");

        var laptop = await context.Products.FirstOrDefaultAsync(p => p.Name == "Laptop");

        if (laptop != null)

        {

            laptop.Price = 70000;

            await context.SaveChangesAsync();

            Console.WriteLine("✅ Laptop price updated to ₹70,000");

        }

        // ❌ DELETE Rice Bag

        Console.WriteLine("\n🗑️ Deleting Rice Bag...");

        var riceBag = await context.Products.FirstOrDefaultAsync(p => p.Name == "Rice Bag");

        if (riceBag != null)

        {

            context.Products.Remove(riceBag);

            await context.SaveChangesAsync();

            Console.WriteLine("✅ Rice Bag removed.");

        }

        // 🧾 Show Updated List

        Console.WriteLine("\n📦 Updated Product List:");

        var updatedProducts = await context.Products.ToListAsync();

        foreach (var p in updatedProducts)

            Console.WriteLine($"🛒 {p.Name} - ₹{p.Price}");

        Console.WriteLine("\n🔽 Filtered Products (Price > ₹1000, High to Low):");

        var filtered = await context.Products

            .Where(p => p.Price > 1000)                // only products > ₹1000

            .OrderByDescending(p => p.Price)           // sort from highest to lowest

            .ToListAsync();

        foreach (var p in filtered)

            Console.WriteLine($"🛒 {p.Name} - ₹{p.Price}");

        Console.WriteLine("\n📄 Product DTOs (Only Name & Price):");

        var productDTOs = await context.Products

            .Select(p => new ProductDTO                // make each product into a simpler object

            {

                Name = p.Name,

                Price = p.Price

            })

            .ToListAsync();

        foreach (var dto in productDTOs)

            Console.WriteLine($"📝 {dto.Name} - ₹{dto.Price}");

    Console.WriteLine("\n📦 Product Stock:");

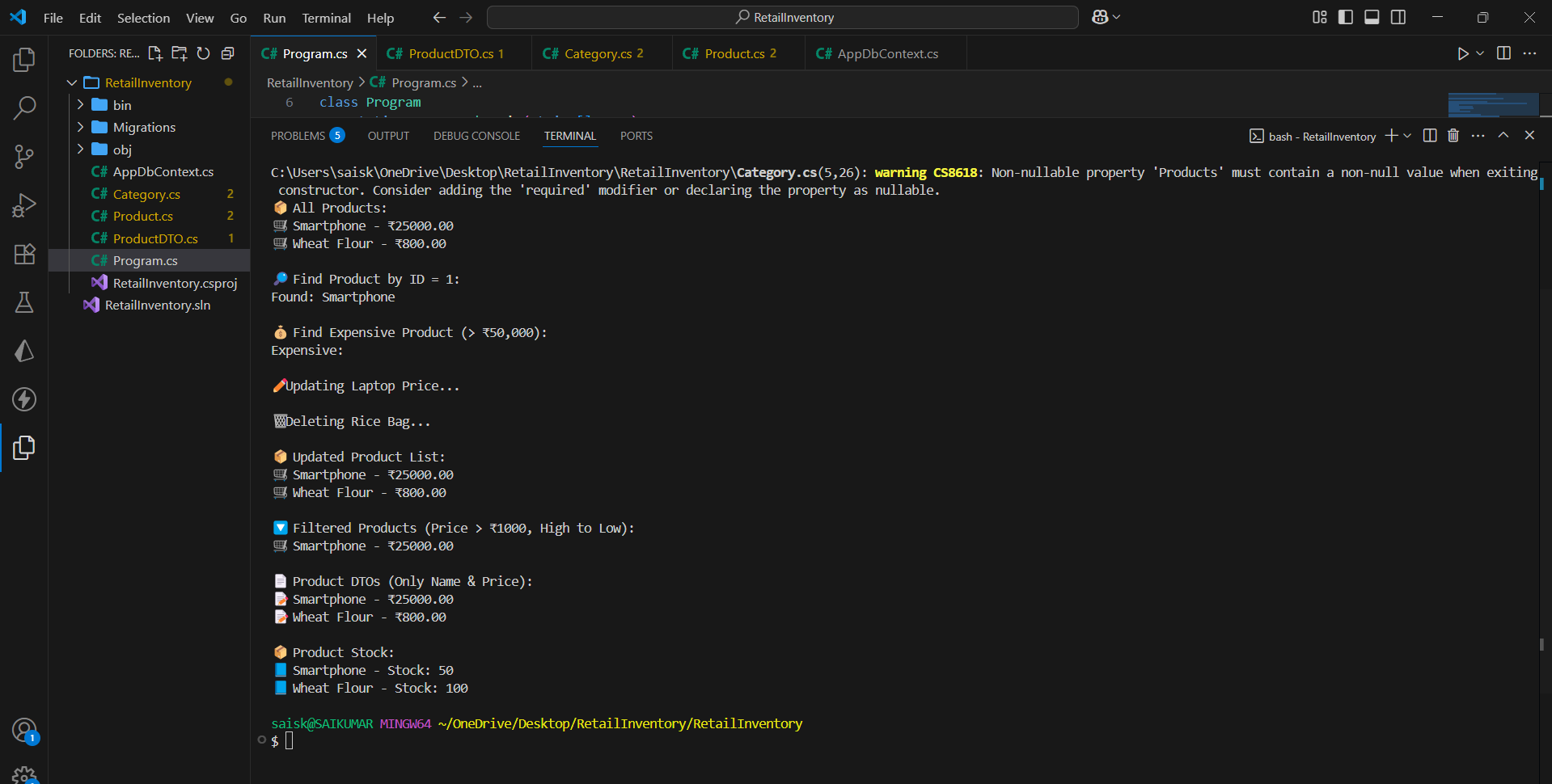
var stockList = await context.Products.ToListAsync();

foreach (var p in stockList)

    Console.WriteLine($"📘 {p.Name} - Stock: {p.StockQuantity}");

    }

}

**Output:**