**Week 3**

**EF Core 8.0 HOL**

**Lab 1**

What is ORM?

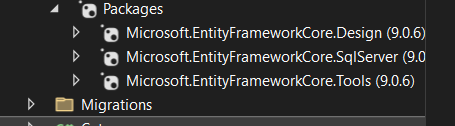
ORM (Object-Relational Mapping) is a technique that connects C# classes to database tables, allowing developers to work with data as objects instead of writing raw SQL.

How it maps C# classes to tables:

* Each class becomes a table.
* Each property becomes a column.
* Relationships are handled with navigation properties and foreign keys.

Benefits of ORM:

* Productivity: Write less boilerplate SQL code.
* Maintainability: Easier to update and refactor using C# code.
* Abstraction from SQL: Focus on business logic without worrying about database syntax.



**Lab 2**

**Category.cs-**

using System.Collections.Generic;

namespace RetailInventory

{

public class Category

{

public int Id { get; set; }

public string Name { get; set; }

// One-to-many relationship

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

}

}

**Product.cs-**

namespace RetailInventory

{

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

public int CategoryId { get; set; }

public required Category Category { get; set; }

}

}

**AppDbContext.cs-**

using Microsoft.EntityFrameworkCore;

namespace RetailInventory

{

public class AppDbContext : DbContext

{

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

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

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

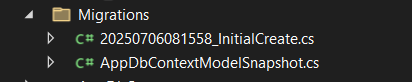
optionsBuilder.UseSqlServer("Server=AYUSHMAN\\SQLEXPRESS;Database=RetailStoreDb;Trusted\_Connection=True;Encrypt=False;");

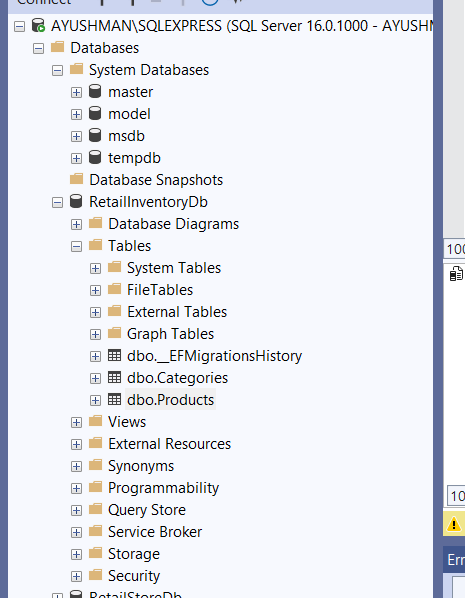
}

}

}

**LAB 3**





**LAB 4**

Program.cs-

using System;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

using RetailInventory;

class Program

{

static async Task Main()

{

using var context = new AppDbContext();

// Prevent duplicate seed

if (await context.Products.AnyAsync()) return;

var electronics = new Category { Name = "Electronics", Products = new List<Product>() };

var groceries = new Category { Name = "Groceries", Products = new List<Product>() };

await context.Categories.AddRangeAsync(electronics, groceries);

var product1 = new Product { Name = "Laptop", Price = 75000, Category = electronics };

var product2 = new Product { Name = "Rice Bag", Price = 1200, Category = groceries };

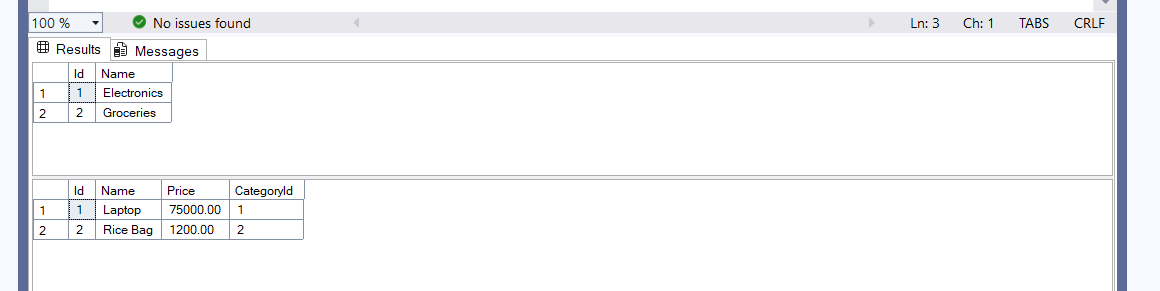
await context.Products.AddRangeAsync(product1, product2);

await context.SaveChangesAsync();

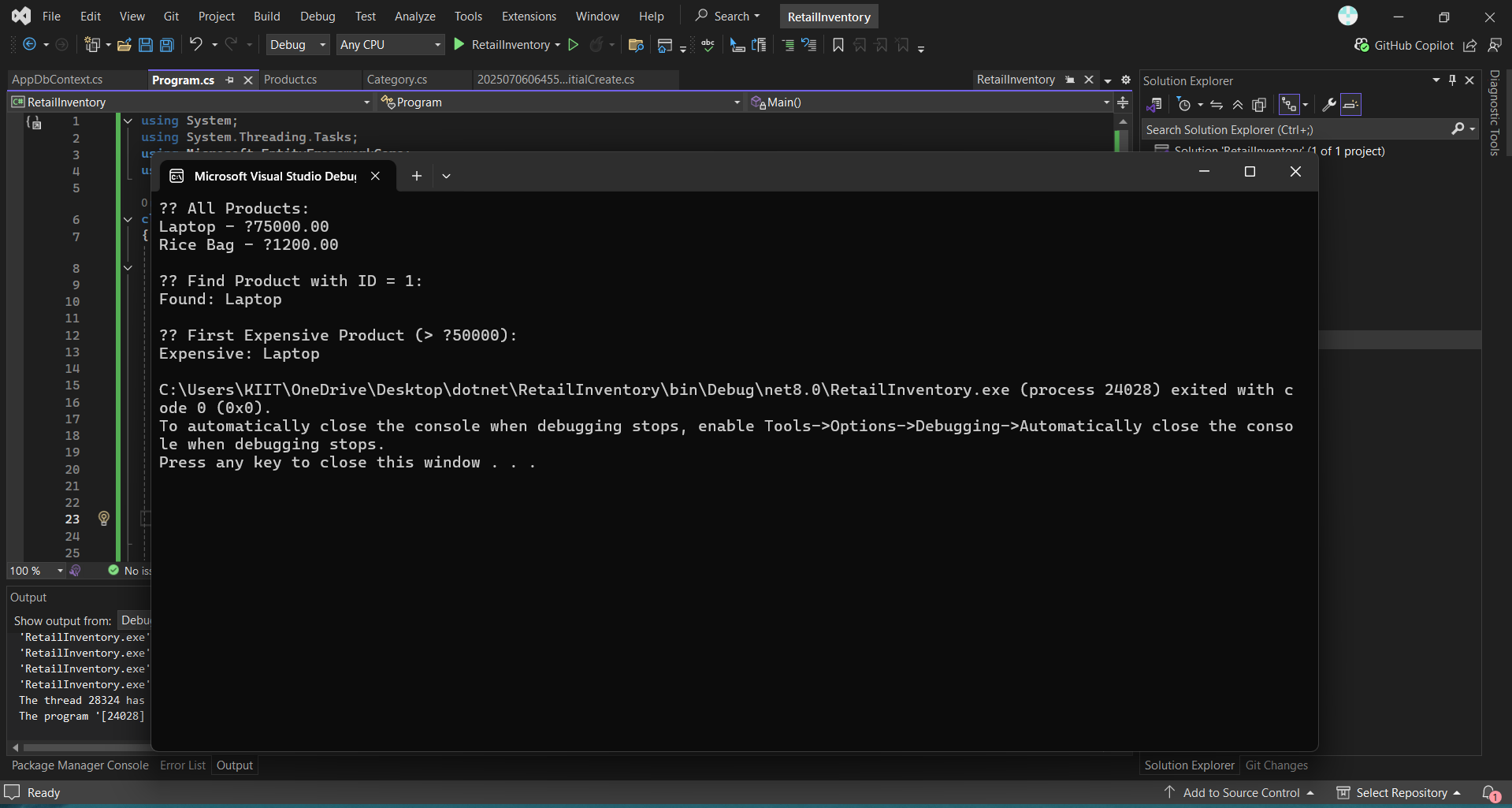
Console.WriteLine("Initial data inserted successfully.");

}

}



**LAB 5**



**Program.cs-**

using System;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

using RetailInventory;

class Program

{

static async Task Main()

{

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 with ID = 1:");

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

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

Console.WriteLine("\n First Expensive Product (> ₹50000):");

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

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

}

}

**LAB 6-**

**Program.cs-**

using System;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

using RetailInventory;

class Program

{

static async Task Main()

{

using var context = new AppDbContext();

// Update Product Price

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

if (product != null)

{

Console.WriteLine($"Old Price of {product.Name}: ₹{product.Price}");

product.Price = 70000;

await context.SaveChangesAsync();

Console.WriteLine($"Updated Price of {product.Name}: ₹{product.Price}");

}

// Delete a Product

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

if (toDelete != null)

{

context.Products.Remove(toDelete);

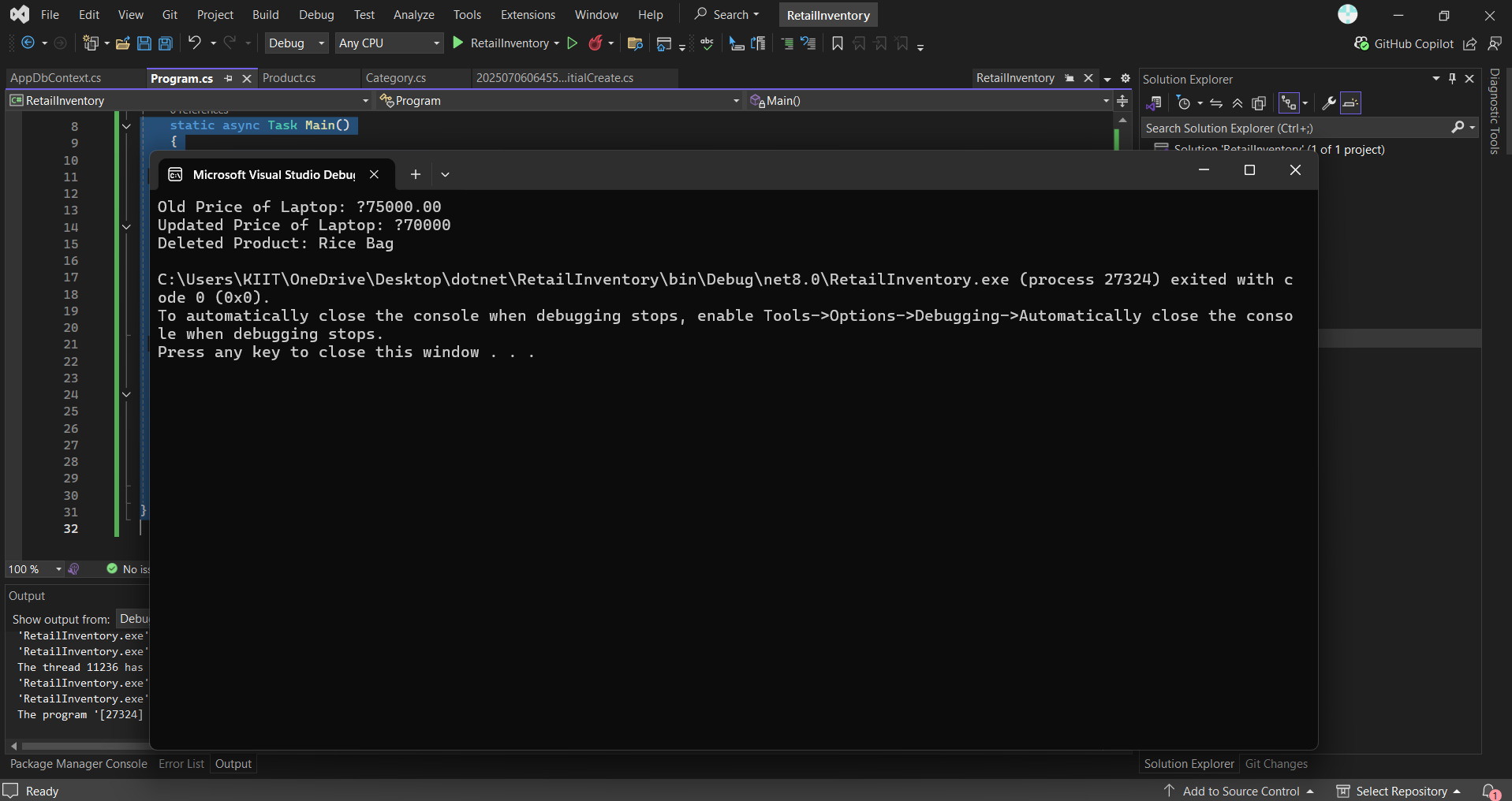
await context.SaveChangesAsync();

Console.WriteLine($"Deleted Product: {toDelete.Name}");

}

}

}

****

**LAB 7**

**Program.cs-**

using System;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

using RetailInventory;

class Program

{

static async Task Main()

{

using var context = new AppDbContext();

//Filter and Sort Products

Console.WriteLine(" Products with Price > ₹1000 (Descending):");

var filtered = await context.Products

.Where(p => p.Price > 1000)

.OrderByDescending(p => p.Price)

.ToListAsync();

foreach (var product in filtered)

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

//Project into DTO

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

var productDTOs = await context.Products

.Select(p => new { p.Name, p.Price })

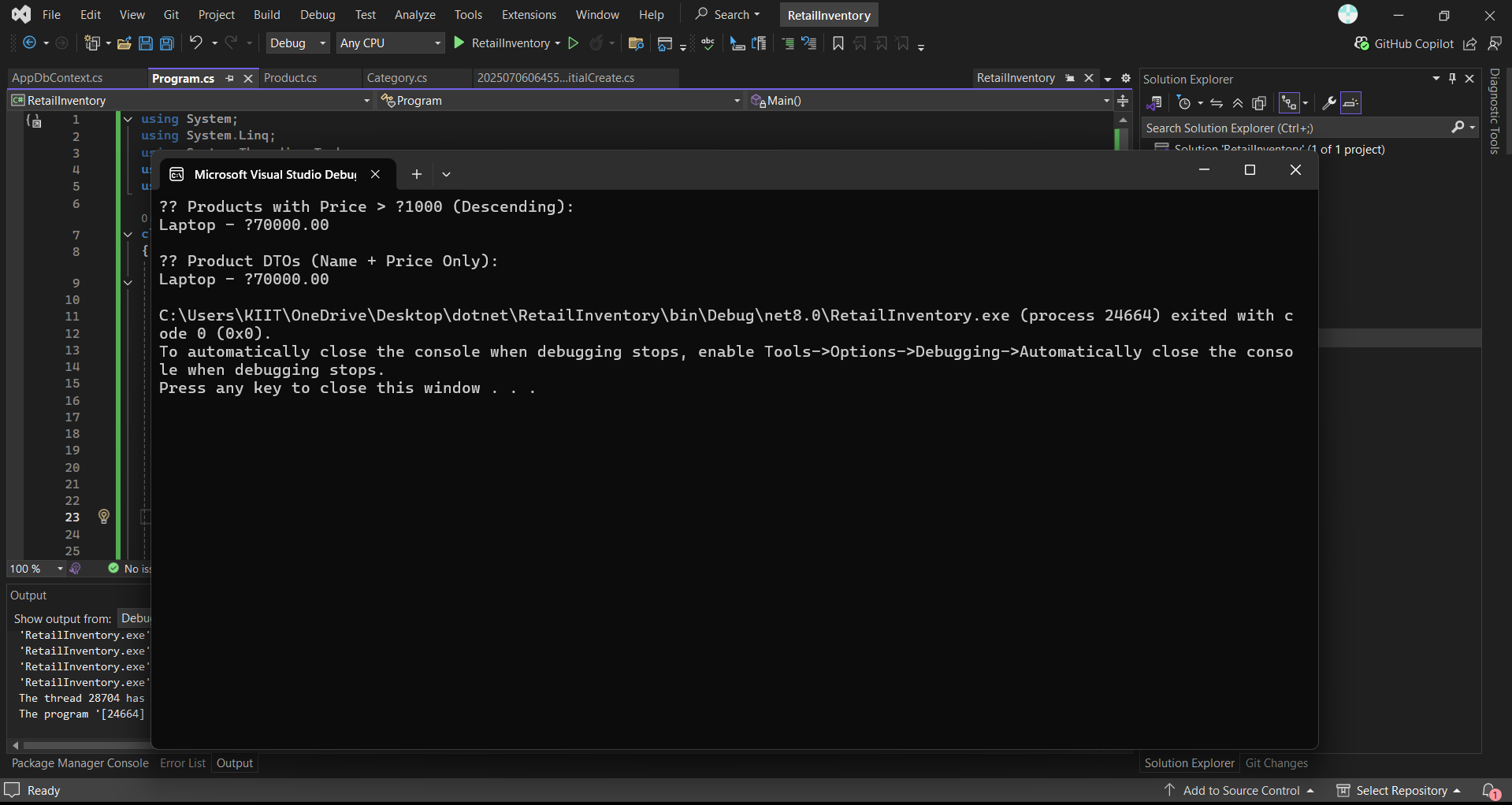
.ToListAsync();

foreach (var dto in productDTOs)

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

}

}

****