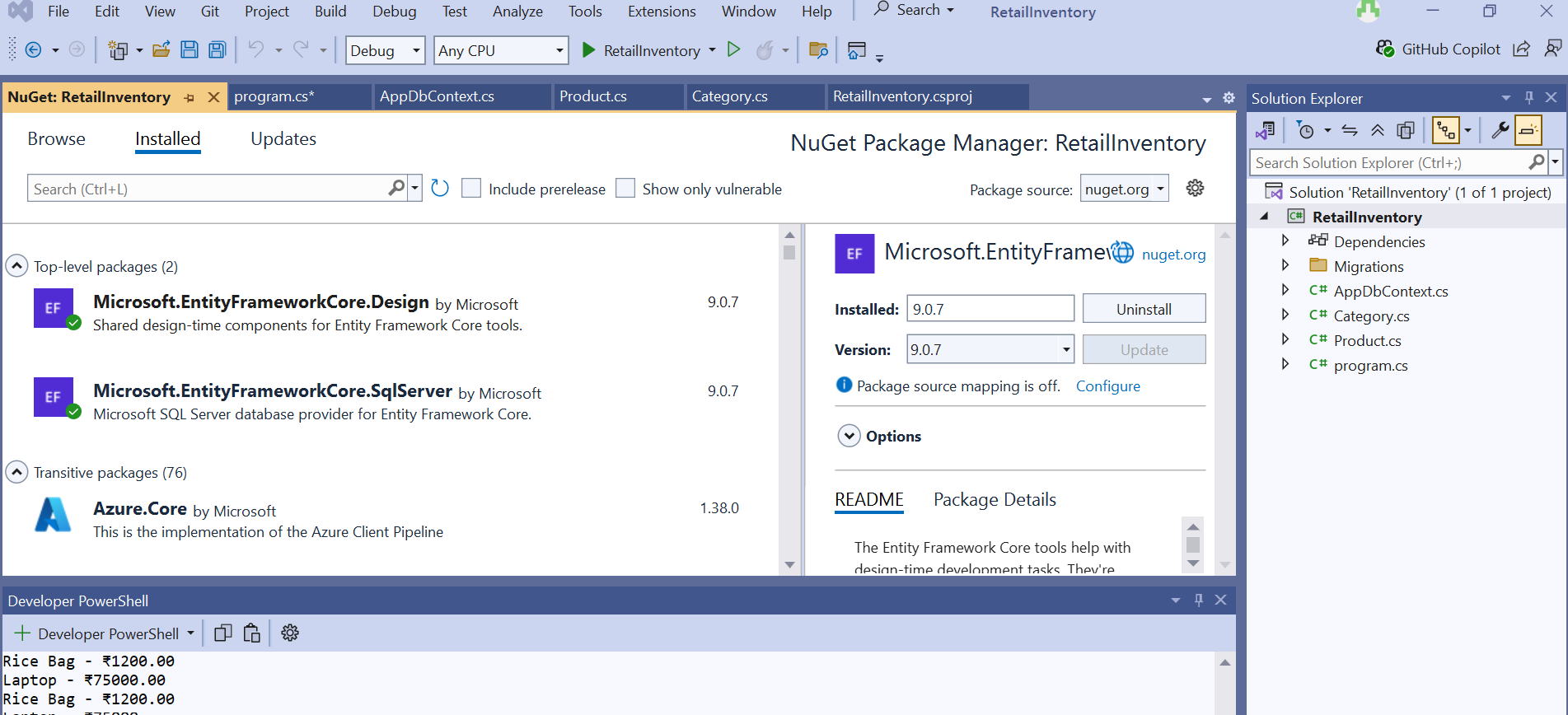
Week 4

Entirty Framework

Lab 1 : Understanding of the setup and installing frameworks for our solution .



Lab 2 : Setting up the Database context

Category:

using System.Text;

using System.Threading.Tasks;

namespace RetailInventory

{

public class Category

{

public int Id { get; set; }

public string Name { get; set; } = string.Empty;

public List<Product> Products { get; set; } = new List<Product>();

}

}

Products :

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RetailInventory

{

public class Product

{

public int Id { get; set; }

public string Name { get; set; } = string.Empty;

public decimal Price { get; set; }

public int CategoryId { get; set; }

public Category Category { get; set; } = null!;

}

}

AppDbContext :

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

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=(localdb)\\MSSQLLocalDB;Database=RetailDb;Trusted\_Connection=True;");

}

}

}

Connection String for my database:

"Server=(localdb)\\MSSQLLocalDB;Database=RetailDb;Trusted\_Connection=True;"

Lab 3 and 4 : Using EF Core CLI to Create and Apply Migrations and Inserting Initial Data into the Database

Installation of EF core CLI : dotnet tool install --global dotnet-ef

Program.cs

using Microsoft.EntityFrameworkCore;

using RetailInventory;

using System;

using System.Threading.Tasks;

class Program

{

static async Task Main(string[] args)

{

using var context = new AppDbContext();

var electronics = new Category { Name = "Electronics" };

var groceries = new Category { Name = "Groceries" };

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);

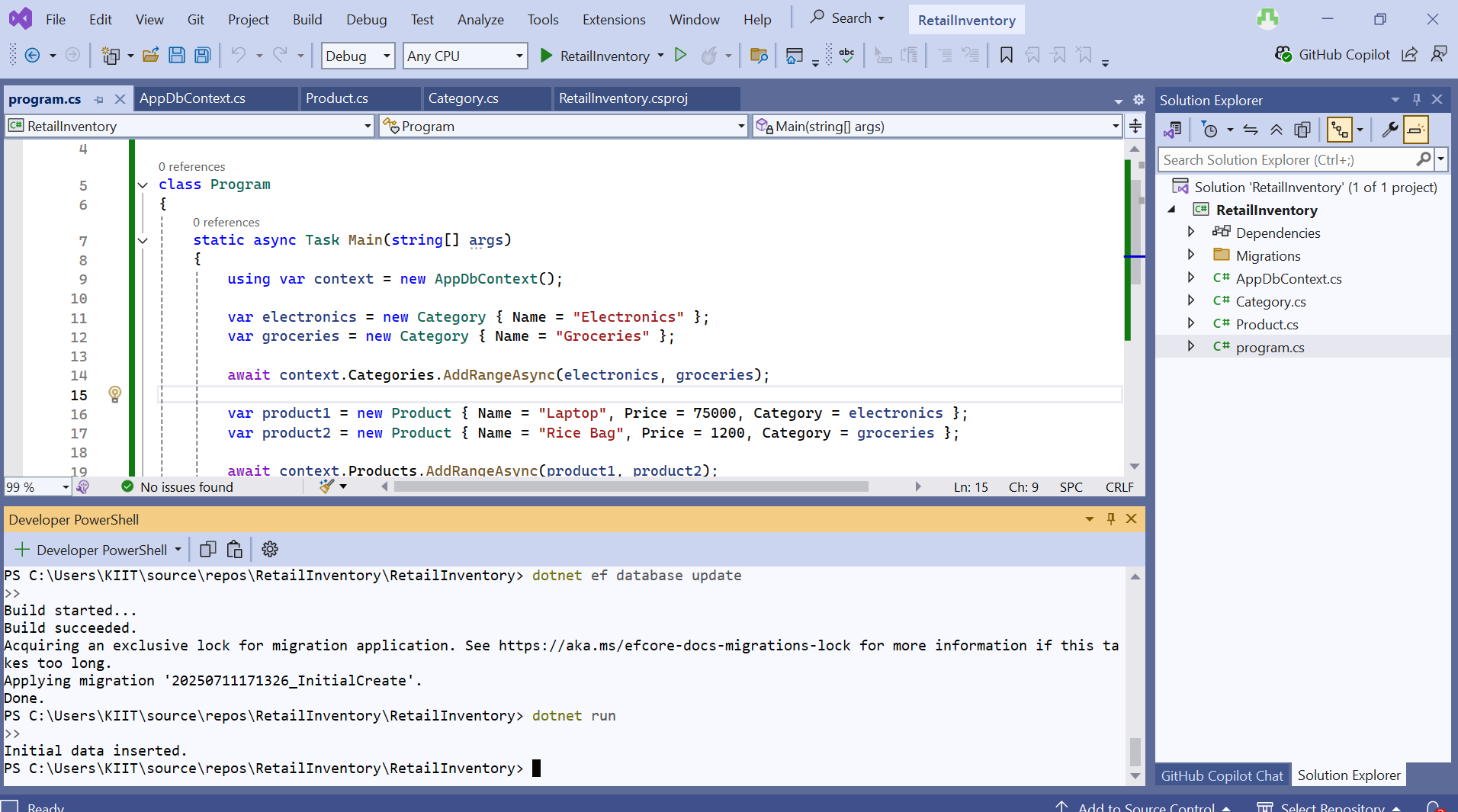
await context.SaveChangesAsync();

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

}

}

}



Lab 5: Retrieving Data from the Database

// Querying the database lab

// Lab 5: Retrieve data from the database

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 (ID = 1):");

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

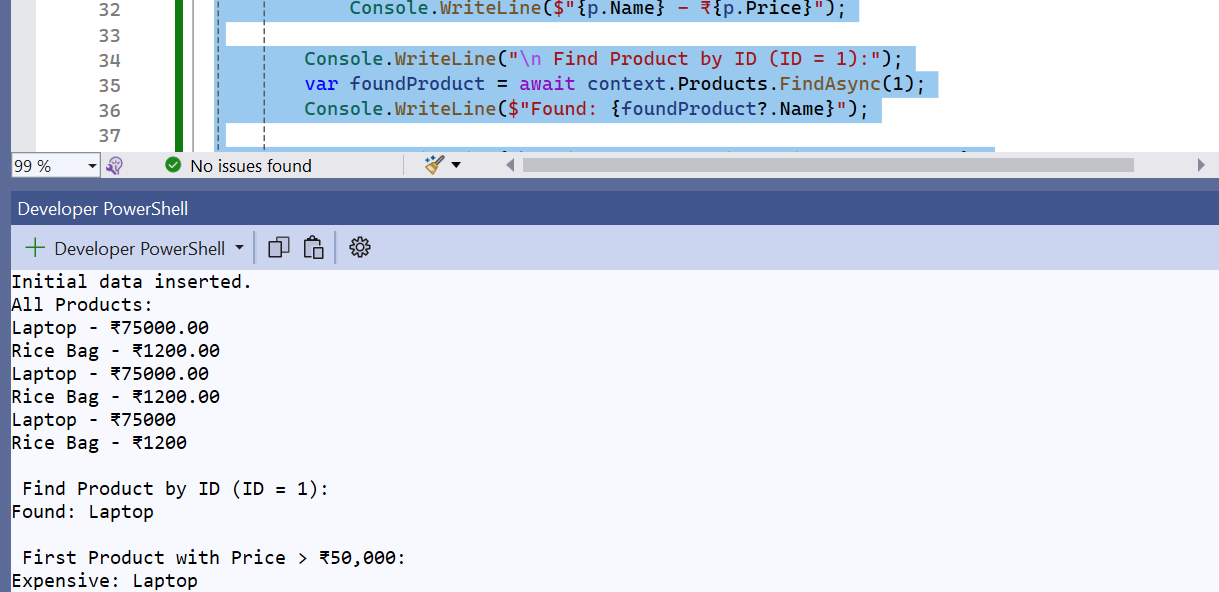
Console.WriteLine($"Found: {foundProduct?.Name}");

Console.WriteLine("\n First Product with Price > ₹50,000:");

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

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

Output :



Lab 6: Updating and Deleting Records

// Lab 6: Updating and Deleting Records

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

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

if (toDelete != null)

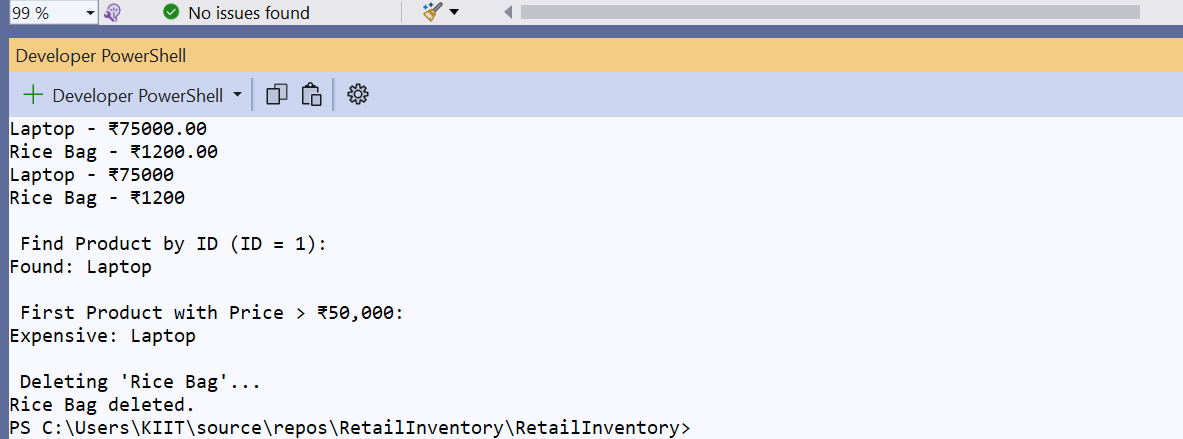
{

context.Products.Remove(toDelete);

await context.SaveChangesAsync();

Console.WriteLine("Rice Bag deleted.");

}



Lab 7: Writing Queries with LINQ

// Lab 7: LINQ Filter + Sort

Console.WriteLine("\n Filtered Products (Price > ₹1000, sorted):");

var filtered = await context.Products

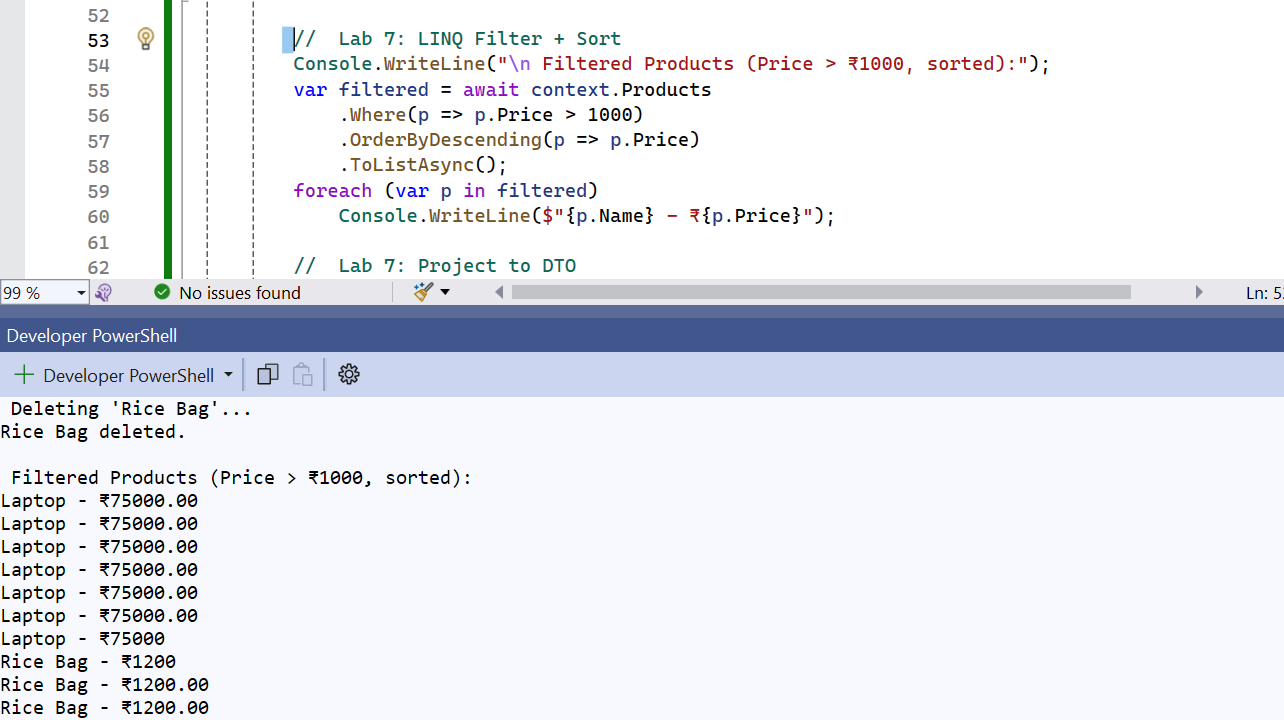
.Where(p => p.Price > 1000)

.OrderByDescending(p => p.Price)

.ToListAsync();

foreach (var p in filtered)

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



Lab 7: Project into DTO

// Project to DTO

Console.WriteLine("\n🧾 Product DTOs:");

var productDTOs = await context.Products

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

.ToListAsync();

foreach (var dto in productDTOs)

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

