ID:6363695

Lab 1: Understanding ORM with a Retail Inventory System

//product.cs

namespace RetailInventory.Models

{

public class Product

{

public int Id { get; set; }

public string? Name { get; set; }

public int Stock { get; set; }

public int CategoryId { get; set; }

public Category? Category { get; set; }

}

}

//category.cs

namespace RetailInventory.Models

{

public class Category

{

public int Id { get; set; }

public string? Name { get; set; }

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

}

}

//APPDbContext.cs

using Microsoft.EntityFrameworkCore;

using RetailInventory.Models;

namespace RetailInventory.Data

{

public class AppDbContext : DbContext

{

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

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

protected override void OnConfiguring(DbContextOptionsBuilder options)

{

options.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=RetailDb;Trusted\_Connection=True;");

}

}

}

//Program.cs

using Microsoft.EntityFrameworkCore;

using RetailInventory.Data;

using RetailInventory.Models;

using var context = new AppDbContext();

// Seed data

if (!context.Categories.Any())

{

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

context.Categories.Add(electronics);

context.SaveChanges();

var phone = new Product

{

Name = "Smartphone",

Stock = 10,

CategoryId = electronics.Id,

Category = electronics

};

context.Products.Add(phone);

context.SaveChanges();

}

// Display

var products = context.Products.Include(p => p.Category).ToList();

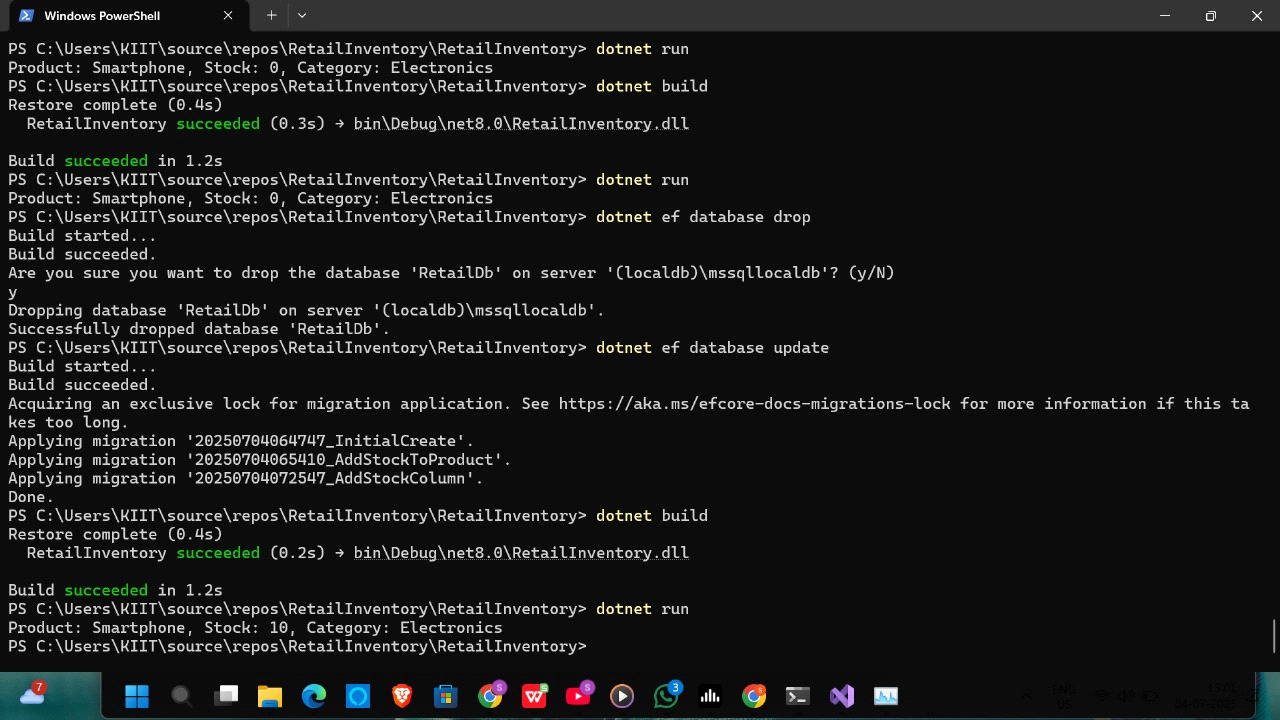
foreach (var p in products)

{

Console.WriteLine($"Product: {p.Name}, Stock: {p.Stock}, Category: {p.Category?.Name}");

}

OUTPUT:



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

//product

namespace RetailInventory.Models

{

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!;

}

}

//category

using System.Collections.Generic;

namespace RetailInventory.Models

{

public class Category

{

public int Id { get; set; }

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

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

}

}

//AppDbContext

using Microsoft.EntityFrameworkCore;

using RetailInventory.Models;

namespace RetailInventory.Data

{

public class AppDbContext : DbContext

{

public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

public DbSet<Product> Products => Set<Product>();

public DbSet<Category> Categories => Set<Category>();

}

}

//AppDbContextFactory

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Design;

namespace RetailInventory.Data

{

public class AppDbContextFactory : IDesignTimeDbContextFactory<AppDbContext>

{

public AppDbContext CreateDbContext(string[] args)

{

var optionsBuilder = new DbContextOptionsBuilder<AppDbContext>();

optionsBuilder.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=RetailDb;Trusted\_Connection=True;");

return new AppDbContext(optionsBuilder.Options);

}

}

}

//program

using Microsoft.EntityFrameworkCore;

using RetailInventory.Data;

using RetailInventory.Models;

using System;

using System.Linq;

class Program

{

static void Main()

{

var options = new DbContextOptionsBuilder<AppDbContext>()

.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=RetailDb;Trusted\_Connection=True;")

.Options;

using var context = new AppDbContext(options);

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

var product = new Product { Name = "Laptop", Price = 999.99m, Category = category };

context.Categories.Add(category);

context.Products.Add(product);

context.SaveChanges();

var products = context.Products.Include(p => p.Category).ToList();

foreach (var p in products)

{

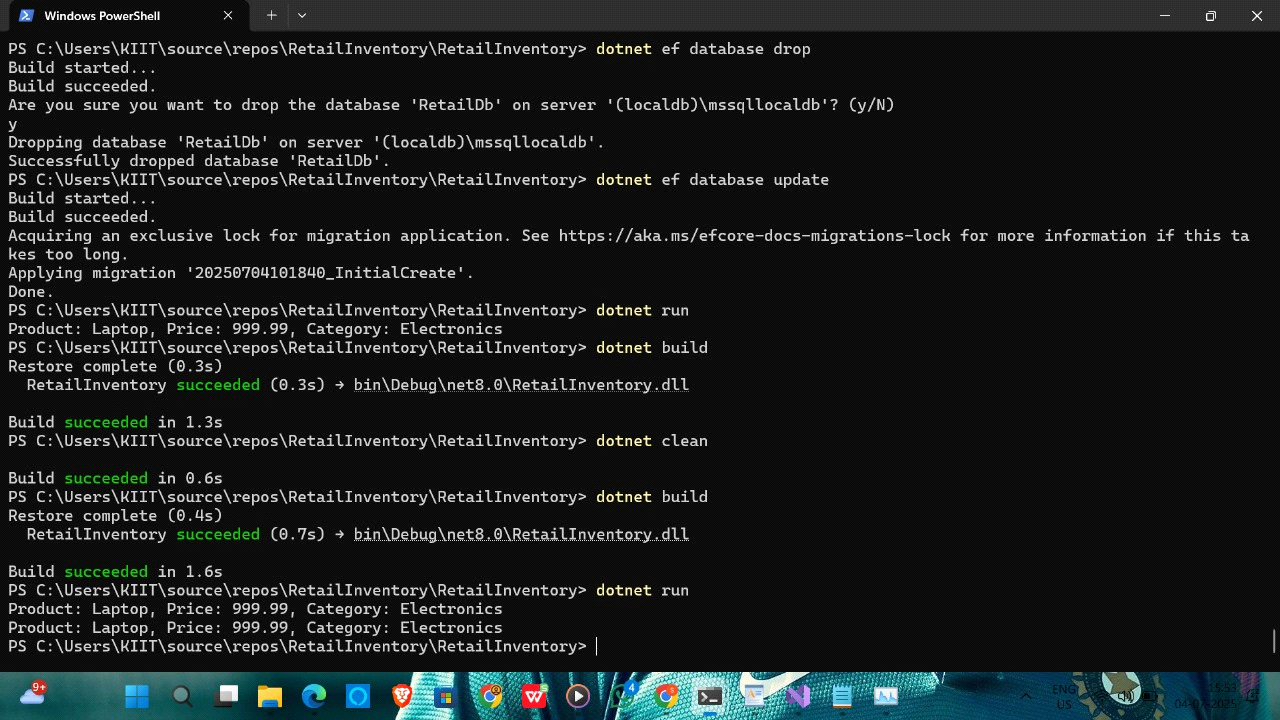
Console.WriteLine($"Product: {p.Name}, Price: {p.Price}, Category: {p.Category.Name}");

}

}

}

OUTPUT:



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

//program

using System;

using System.Linq;

using Microsoft.EntityFrameworkCore;

using RetailInventory.Data;

using RetailInventory.Models;

class Program

{

static void Main(string[] args)

{

var optionsBuilder = new DbContextOptionsBuilder<AppDbContext>();

optionsBuilder.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=RetailDb;Trusted\_Connection=True;");

using var context = new AppDbContext(optionsBuilder.Options);

// Add sample data if not exists

if (!context.Categories.Any())

{

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

var product = new Product { Name = "Laptop", Price = 59999.99m, Category = category };

context.Categories.Add(category);

context.Products.Add(product);

context.SaveChanges();

}

// Fetch and display

var products = context.Products.Include(p => p.Category).ToList();

foreach (var p in products)

{

Console.WriteLine($"Product: {p.Name}, Category: {p.Category.Name}, Price: ₹{p.Price}");

}

}

}

//product

namespace RetailInventory.Models

{

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!;

}

}

//category

using System.Collections.Generic;

namespace RetailInventory.Models

{

public class Category

{

public int Id { get; set; }

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

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

}

}

//AppDbContext

using Microsoft.EntityFrameworkCore;

using RetailInventory.Models;

namespace RetailInventory.Data

{

public class AppDbContext : DbContext

{

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

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

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

// 👇 This is where the connection string is used

optionsBuilder.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=RetailDb;Trusted\_Connection=True;");

}

}

}

//AppDbConextFactory

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Design;

namespace RetailInventory.Data

{

public class AppDbContextFactory : IDesignTimeDbContextFactory<AppDbContext>

{

public AppDbContext CreateDbContext(string[] args)

{

var optionsBuilder = new DbContextOptionsBuilder<AppDbContext>();

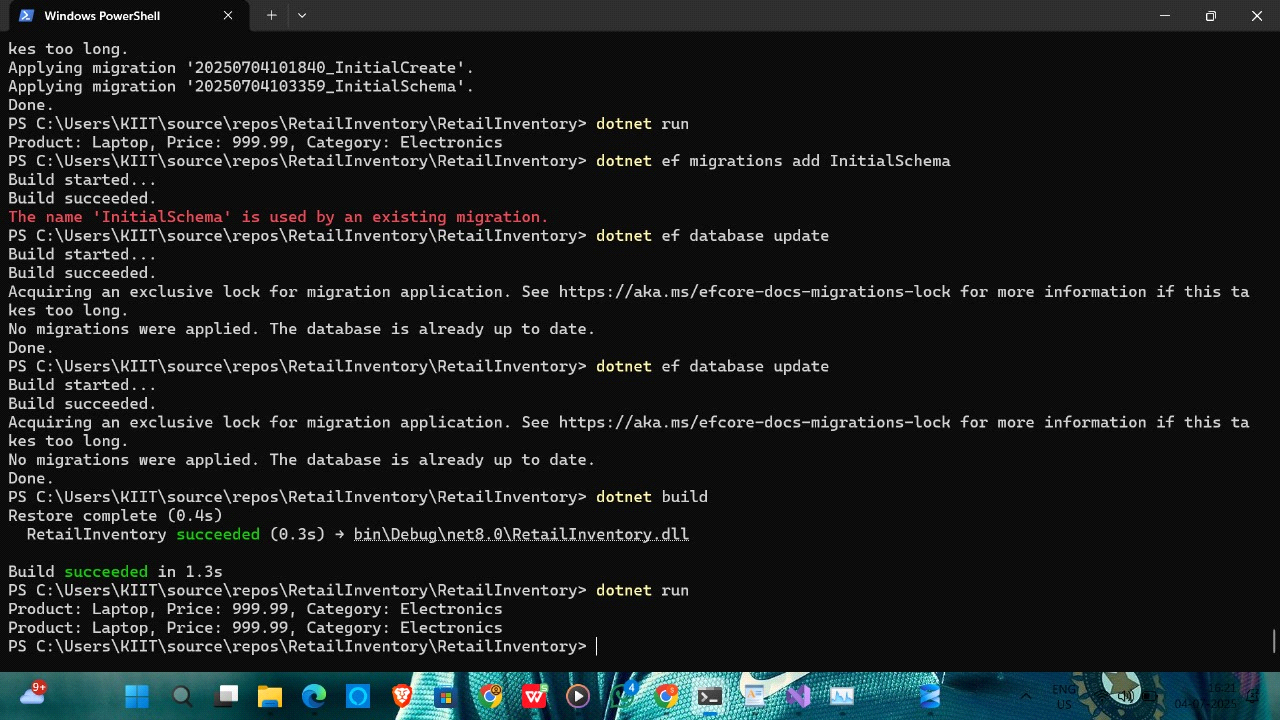
optionsBuilder.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=RetailDb;Trusted\_Connection=True;");

return new AppDbContext(optionsBuilder.Options);

}

}

}



Lab 4: Inserting Initial Data into the Database

//program.cs

using Microsoft.EntityFrameworkCore;

using RetailInventory.Data;

using RetailInventory.Models;

using System;

using System.Linq;

using System.Threading.Tasks;

class Program

{

static async Task Main(string[] args)

{

var optionsBuilder = new DbContextOptionsBuilder<AppDbContext>();

optionsBuilder.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=RetailDb;Trusted\_Connection=True;");

using var context = new AppDbContext(optionsBuilder.Options);

await context.Products.ExecuteDeleteAsync();

await context.Categories.ExecuteDeleteAsync(); 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();

var products = context.Products.Include(p => p.Category).ToList();

foreach (var p in products)

{

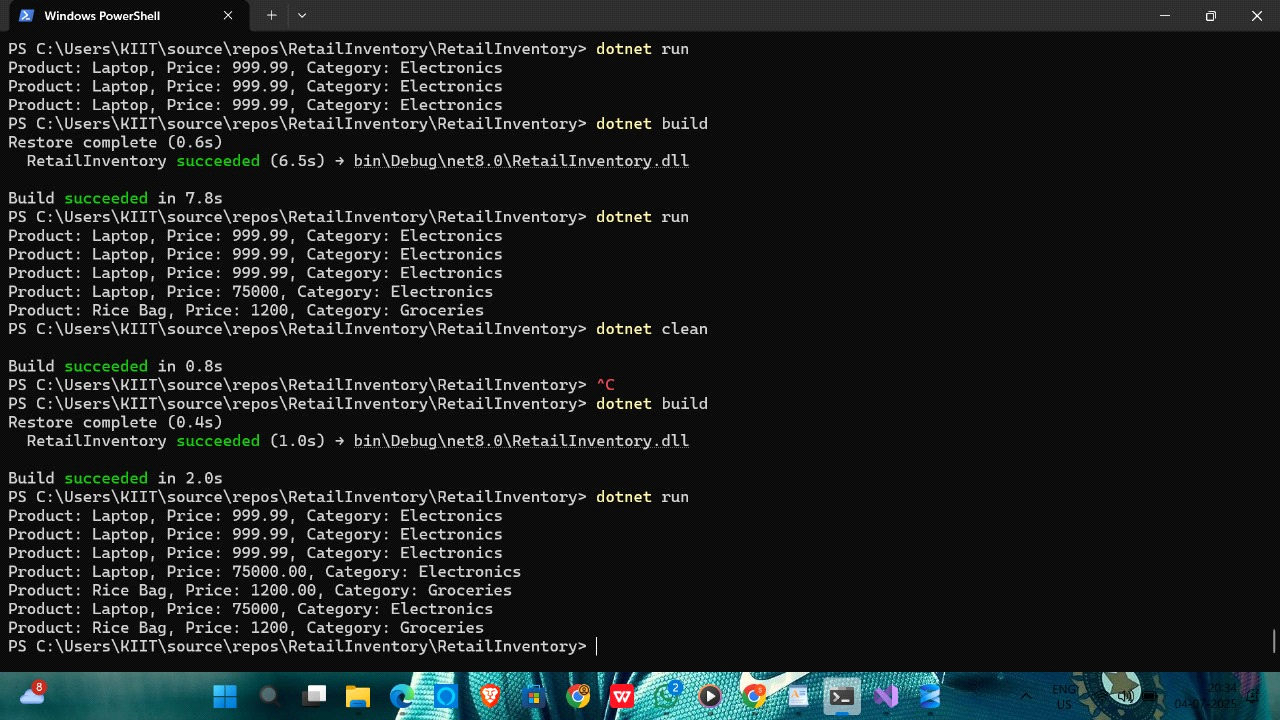
Console.WriteLine($"Product: {p.Name}, Price: {p.Price}, Category: {p.Category.Name}");

}

}

}

OUTPUT:



Lab 5: Retrieving Data from the Database

//Program.cs

using Microsoft.EntityFrameworkCore;

using RetailInventory.Data;

using RetailInventory.Models;

using System;

using System.Threading.Tasks;

class Program

{

static async Task Main(string[] args)

{

var optionsBuilder = new DbContextOptionsBuilder<AppDbContext>();

optionsBuilder.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=RetailDb;Trusted\_Connection=True;");

using var context = new AppDbContext(optionsBuilder.Options);

// 1. Retrieve all products

var products = await context.Products.Include(p => p.Category).ToListAsync();

foreach (var p in products)

{

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

}

// 2. Find by ID

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

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

// 3. FirstOrDefault with condition

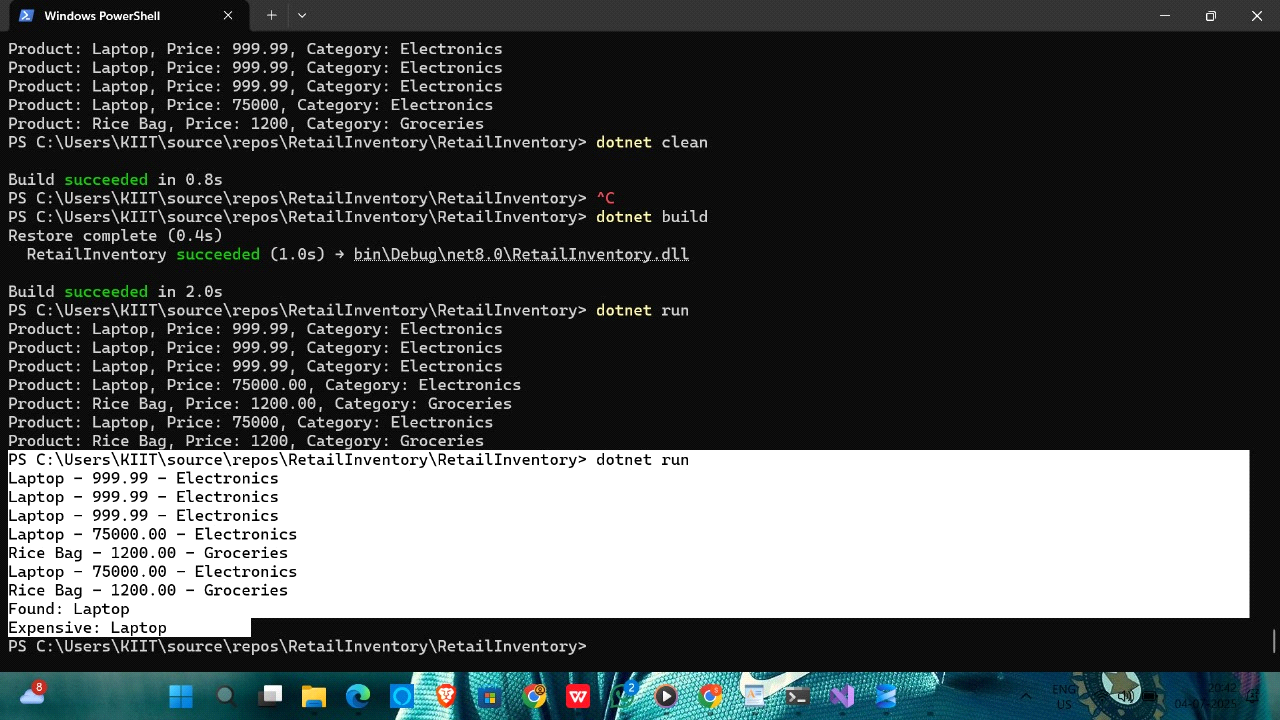
var expensive = await context.Products.FirstOrDefaultAsync(p => p.Price > 50000);

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

}

}

OUTPUT:



Lab 6: Updating and Deleting Records

//program.cs

using Microsoft.EntityFrameworkCore;

using RetailInventory.Data;

using RetailInventory.Models;

using System;

using System.Threading.Tasks;

class Program

{

static async Task Main(string[] args)

{

var optionsBuilder = new DbContextOptionsBuilder<AppDbContext>();

optionsBuilder.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=RetailDb;Trusted\_Connection=True;");

using var context = new AppDbContext(optionsBuilder.Options);

// 1. UPDATE Product price

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

if (product != null)

{

product.Price = 70000;

await context.SaveChangesAsync();

Console.WriteLine("Updated Laptop price to 70000");

}

else

{

Console.WriteLine("Laptop not found for update.");

}

// 2. DELETE 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 Rice Bag from products.");

}

else

{

Console.WriteLine("Rice Bag not found for deletion.");

}

// 3. View remaining products

var products = await context.Products.Include(p => p.Category).ToListAsync();

Console.WriteLine("\nRemaining Products:");

foreach (var p in products)

{

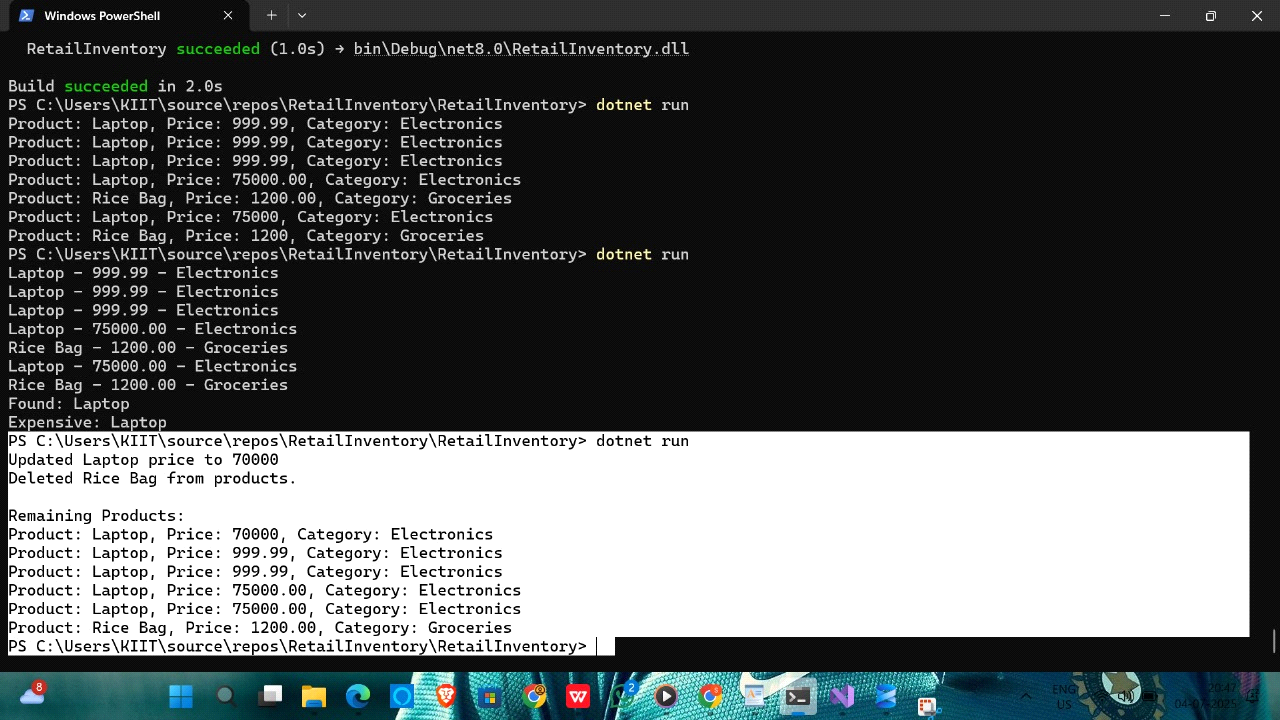
Console.WriteLine($"Product: {p.Name}, Price: {p.Price}, Category: {p.Category?.Name}");

}

}

}

OUTPUT:



Lab 7: Writing Queries with LINQ

//program.cs

using RetailInventory.Models;

using Microsoft.EntityFrameworkCore;

using RetailInventory.Data;

using System;

using System.Linq;

using System.Threading.Tasks;

class Program

{

static async Task Main(string[] args)

{

var optionsBuilder = new DbContextOptionsBuilder<AppDbContext>();

optionsBuilder.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=RetailDb;Trusted\_Connection=True;");

using var context = new AppDbContext(optionsBuilder.Options);

// 1. Filter and sort products with price > 1000 and sort by price descending

var filteredProducts = await context.Products

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

.OrderByDescending(p => p.Price)

.Include(p => p.Category)

.Select(p => new ProductDto

{

Name = p.Name,

Price = p.Price,

CategoryName = p.Category.Name

})

.ToListAsync();

Console.WriteLine("Filtered and Sorted Products:");

foreach (var p in filteredProducts)

{

Console.WriteLine($"Name: {p.Name}, Price: {p.Price}, Category: {p.CategoryName}");

}

}

}

//product.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace RetailInventory.Models

{

public class Product

{

public int Id { get; set; }

[Required]

public string Name { get; set; }

[Column(TypeName = "decimal(18,2)")]

public decimal Price { get; set; }

public int CategoryId { get; set; }

public Category Category { get; set; }

}

}

//category.cs

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace RetailInventory.Models

{

public class Category

{

public int Id { get; set; }

[Required]

public string Name { get; set; }

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

}

}

//AppDbContext.cs

using Microsoft.EntityFrameworkCore;

using RetailInventory.Models;

namespace RetailInventory.Data

{

public class AppDbContext : DbContext

{

public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)

{

}

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

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

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

optionsBuilder.UseSqlServer("Your\_Connection\_String\_Here");

}

}

}

}

//AppDbContextFactory.cs

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Design;

namespace RetailInventory.Data

{

public class AppDbContextFactory : IDesignTimeDbContextFactory<AppDbContext>

{

public AppDbContext CreateDbContext(string[] args)

{

var optionsBuilder = new DbContextOptionsBuilder<AppDbContext>();

optionsBuilder.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=RetailDb;Trusted\_Connection=True;");

return new AppDbContext(optionsBuilder.Options);

}

}

}

//ProductDto.cs

namespace RetailInventory.Models

{

public class ProductDto

{

public string Name { get; set; }

public decimal Price { get; set; }

public string CategoryName { get; set; }

}

}

OUTPUT:

