NAME – NIRBHIK MANDAL

**COGNIZANT WEEK 3 LEARNING**

**Entity Framework Core 8.0**

Lab 1: Understanding ORM with a Retail Inventory System

Code—

using System;

using System.Linq;

using Microsoft.EntityFrameworkCore;

using RetailInventory.Data;

using RetailInventory.Models;

using var context = new AppDbContext();

// Ask user how many products to add

Console.Write("How many products do you want to add? ");

int count = int.Parse(Console.ReadLine()!);

for (int i = 0; i < count; i++)

{

    Console.WriteLine($"\nProduct #{i + 1}");

    Console.Write("Name: ");

    string name = Console.ReadLine()!;

    Console.Write("Quantity: ");

    int qty = int.Parse(Console.ReadLine()!);

    Console.Write("Price: ");

    decimal price = decimal.Parse(Console.ReadLine()!);

    var product = new Product

    {

        Name = name,

        Quantity = qty,

        Price = price

    };

    context.Products.Add(product);

}

context.SaveChanges();

Console.WriteLine("\nProducts saved successfully!\n");

// Display all products

var products = context.Products.ToList();

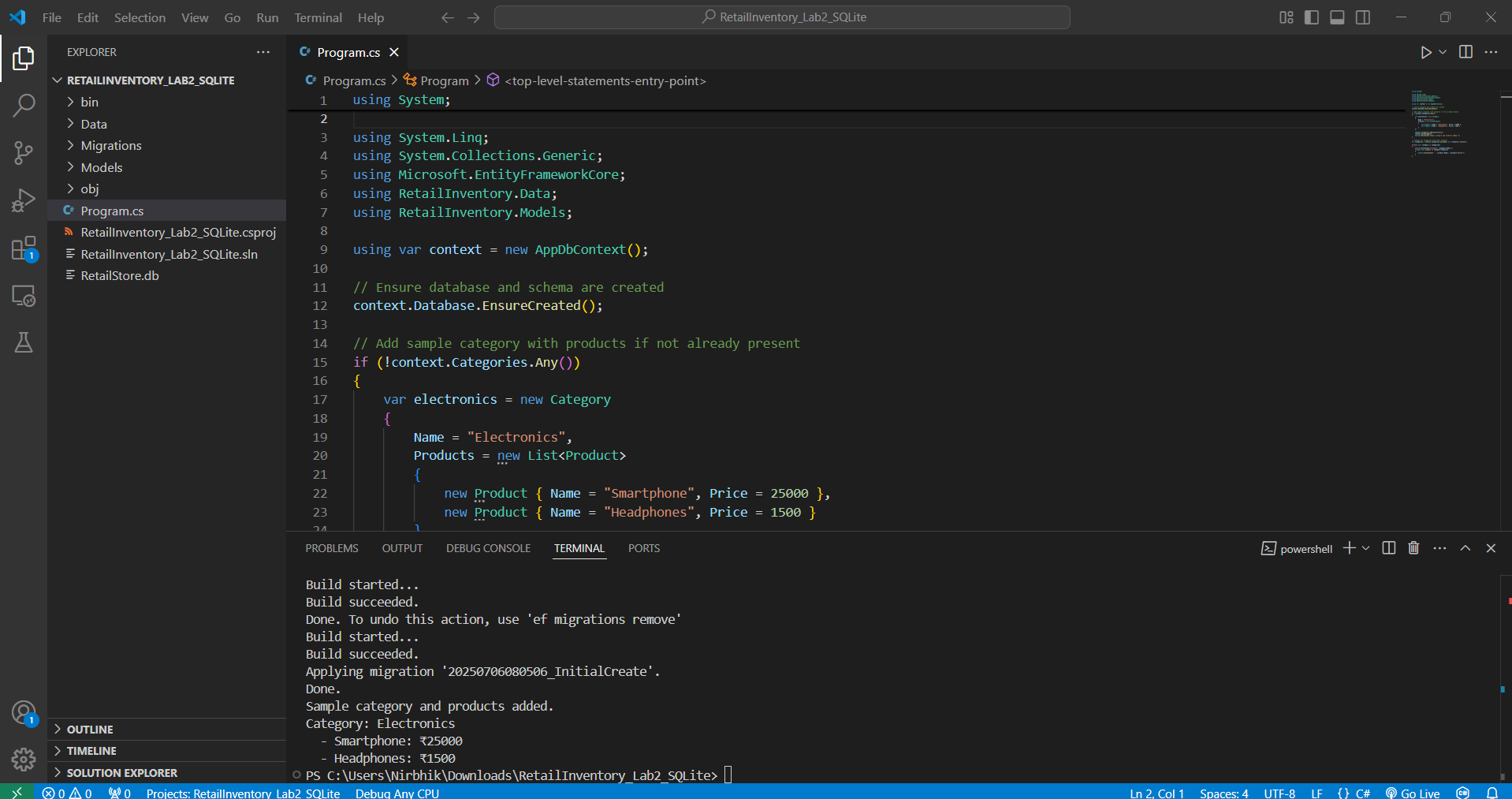
foreach (var p in products)

{

    Console.WriteLine($"{p.Name} - Qty: {p.Quantity}, Price: {p.Price}");

}

Output –



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

Code –

using System;

using System.Linq;

using System.Collections.Generic;

using Microsoft.EntityFrameworkCore;

using RetailInventory.Data;

using RetailInventory.Models;

using var context = new AppDbContext();

// Ensure database and schema are created

context.Database.EnsureCreated();

// Add sample category with products if not already present

if (!context.Categories.Any())

{

    var electronics = new Category

    {

        Name = "Electronics",

        Products = new List<Product>

        {

            new Product { Name = "Smartphone", Price = 25000 },

            new Product { Name = "Headphones", Price = 1500 }

        }

    };

    context.Categories.Add(electronics);

    context.SaveChanges();

    Console.WriteLine("Sample category and products added.");

}

// Display all categories with their products

var categories = context.Categories.Include(c => c.Products).ToList();

foreach (var category in categories)

{

    Console.WriteLine($"Category: {category.Name}");

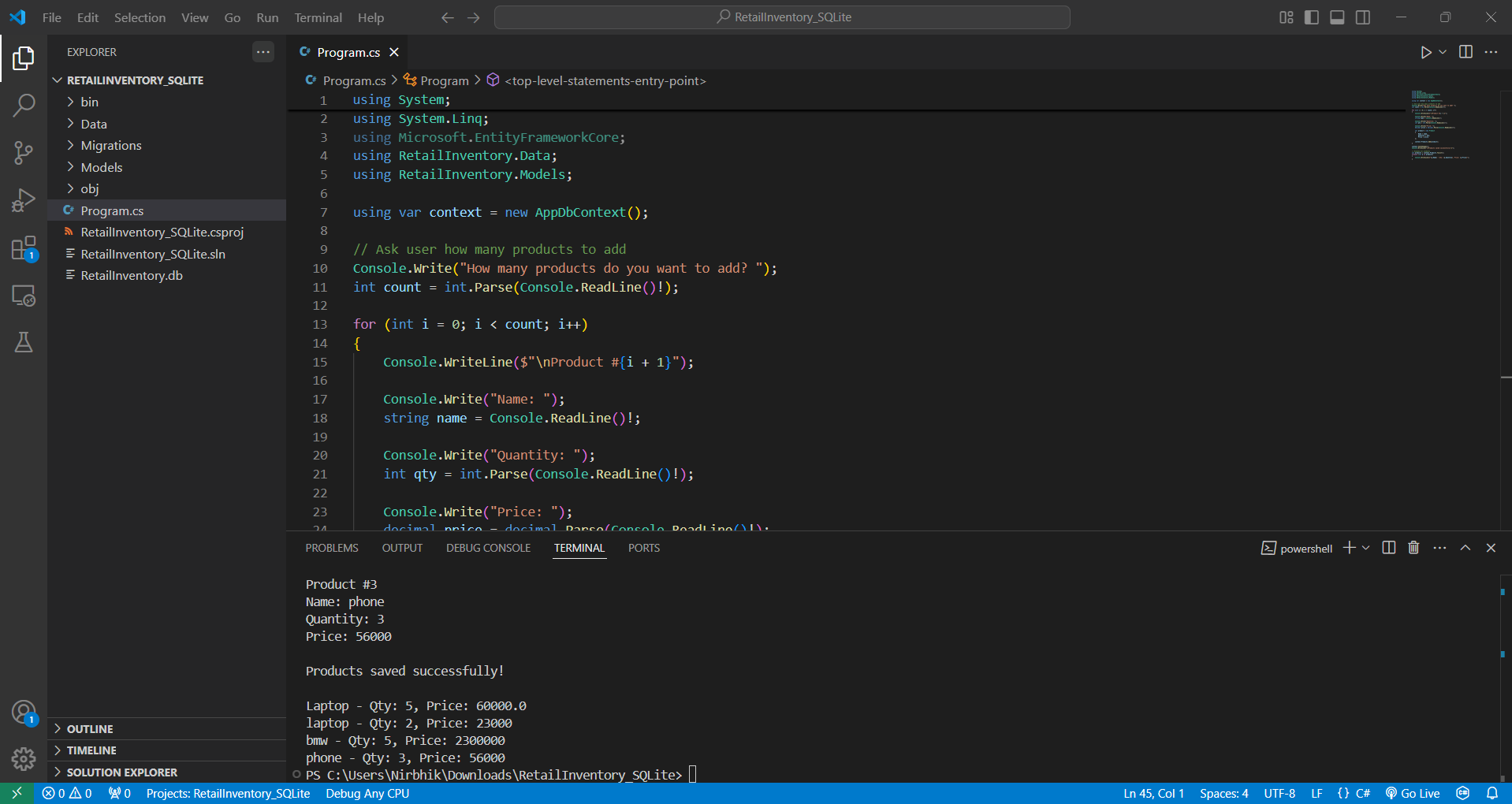
    foreach (var product in category.Products)

    {

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

    }

}

**Output –**

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

**Code –**

using System;

using System.Collections.Generic;

using System.Linq;

using Microsoft.EntityFrameworkCore;

using RetailInventory.Data;

using RetailInventory.Models;

using var context = new AppDbContext();

context.Database.EnsureCreated();

if (!context.Categories.Any())

{

    var electronics = new Category

    {

        Id = 1,

        Name = "Electronics",

        Products = new List<Product>

        {

            new Product { Id = 101, Name = "Smartphone", Price = 25000, CategoryId = 1 },

            new Product { Id = 102, Name = "Mouse", Price = 500, CategoryId = 1 }

        }

    };

    context.Categories.Add(electronics);

    context.SaveChanges();

    Console.WriteLine("Sample data added.");

}

var categories = context.Categories.Include(c => c.Products).ToList();

foreach (var category in categories)

{

    Console.WriteLine($"Category: {category.Name}");

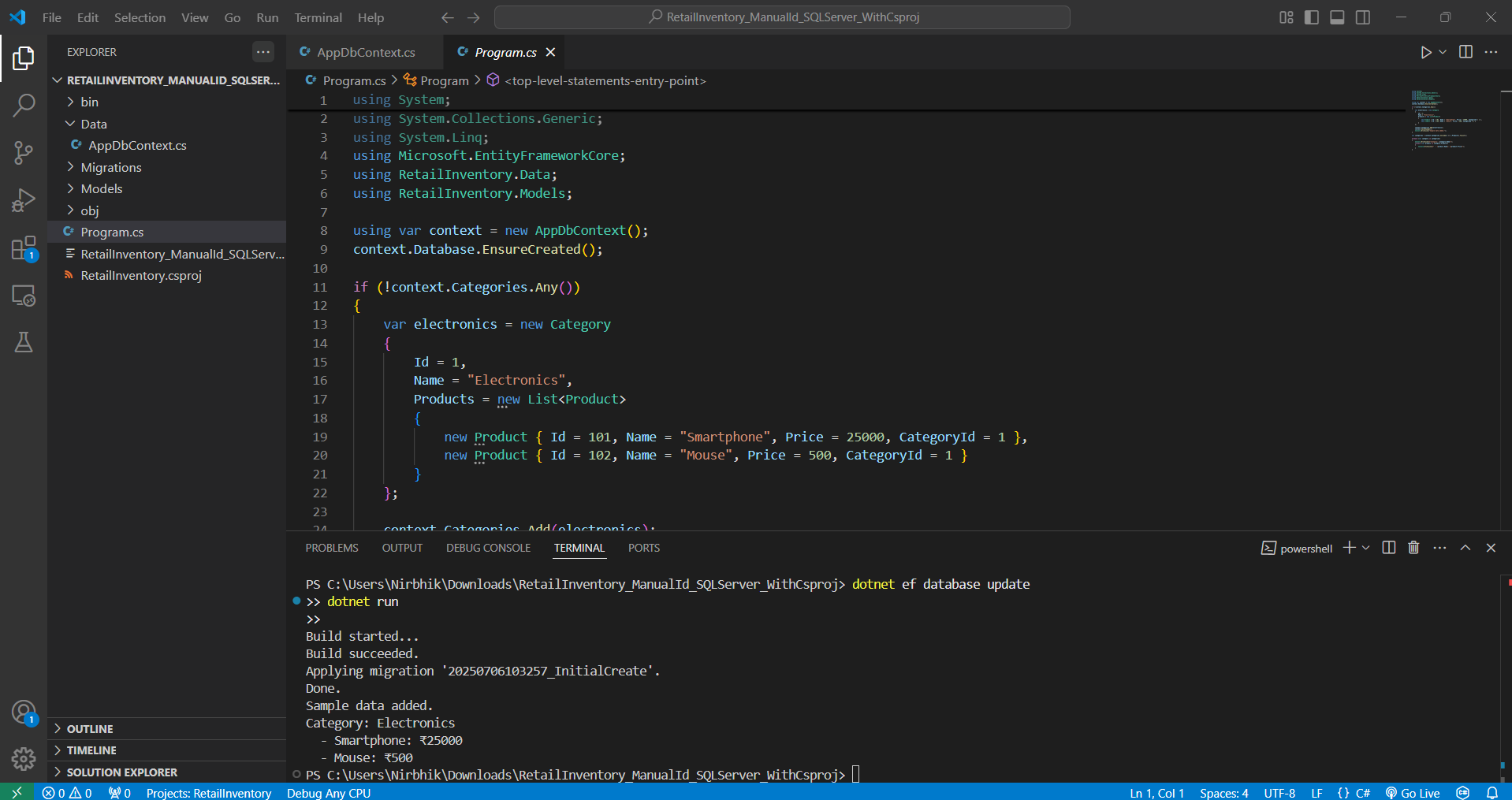
    foreach (var product in category.Products)

    {

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

    }

}

**Output –**

**Lab 4: Inserting Initial Data into the Database**

**Code –**

using RetailInventory.Data;

using RetailInventory.Models;

using Microsoft.EntityFrameworkCore;

var context = new AppDbContext();

// Optional: Ensure database is created

await context.Database.EnsureCreatedAsync();

// Insert initial data

if (!await context.Categories.AnyAsync())

{

    var electronics = new Category { Id = 1, Name = "Electronics" };

    var groceries = new Category { Id = 2, Name = "Groceries" };

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

    var product1 = new Product { Id = 101, Name = "Laptop", Price = 75000, Category = electronics };

    var product2 = new Product { Id = 102, Name = "Rice Bag", Price = 1200, Category = groceries };

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

    await context.SaveChangesAsync();

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

}

else

{

    Console.WriteLine(" Data already exists.");

}

// Display inserted data

var categories = await context.Categories.Include(c => c.Products).ToListAsync();

foreach (var category in categories)

{

    Console.WriteLine($"Category: {category.Name}");

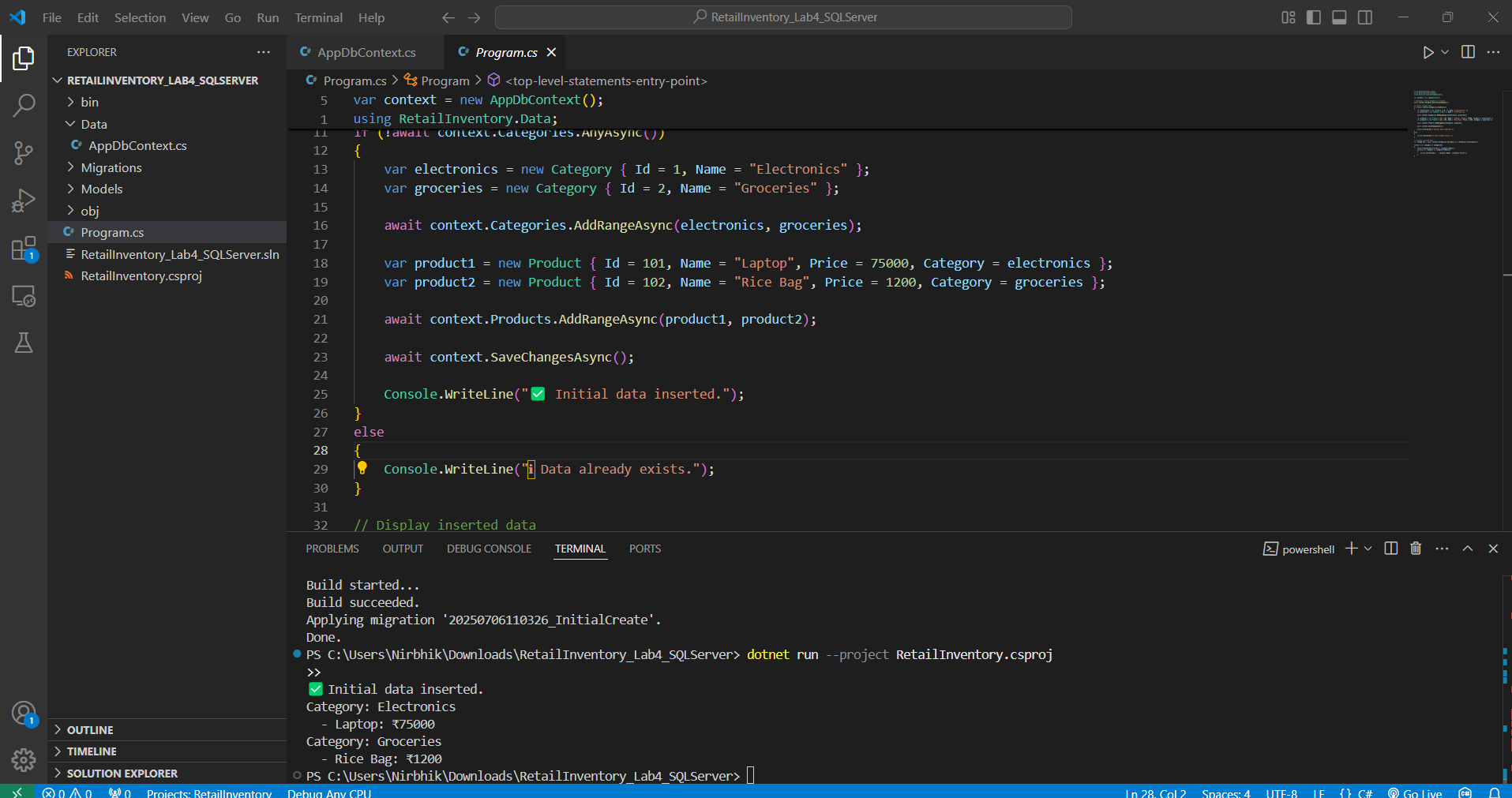
    foreach (var product in category.Products)

    {

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

    }

}

**Output –**

**Lab 5: Retrieving Data from the Database**

**Code –**

using Microsoft.EntityFrameworkCore;

using var context = new StoreContext();

// Seed if empty

if (!context.Products.Any())

{

    context.Products.AddRange(

        new Product { Name = "Laptop", Price = 60000 },

        new Product { Name = "Mouse", Price = 1200 },

        new Product { Name = "Keyboard", Price = 2500 }

    );

    await context.SaveChangesAsync();

}

// 1. Retrieve all products

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

foreach (var p in products)

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

// 2. Find by ID

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

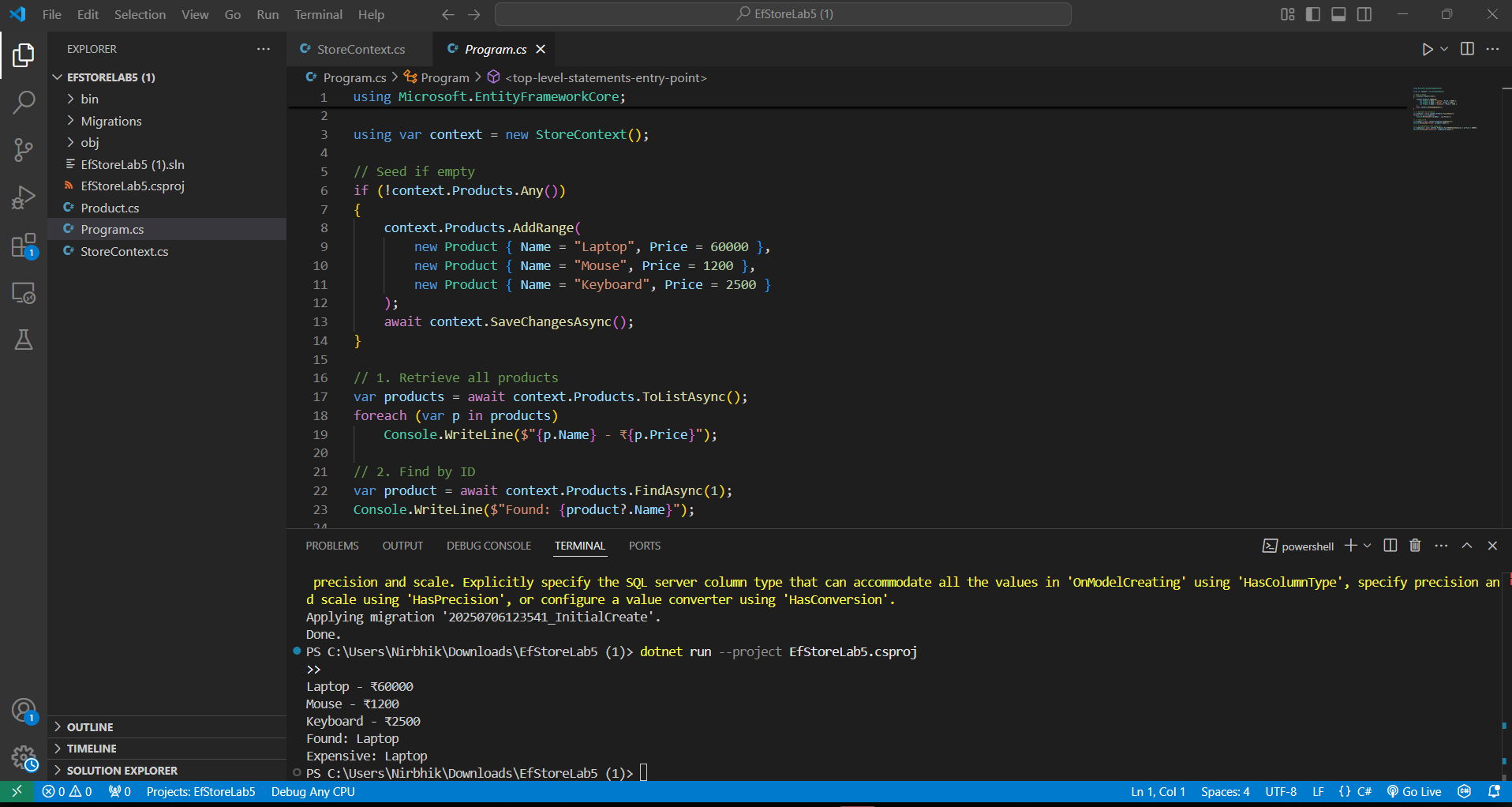
Console.WriteLine($"Found: {product?.Name}");

// 3. FirstOrDefault with condition

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

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

**Output --**

****