Name-Sangram Mandal

Superset Id-6363848

Week-3(Handson-Exercise)

1.**Entity Framework Core 8.0:**

**LAB-1: Understanding ORM with a RetailInventory System:-**

**Code:**

**Creating Models file:**

**In Product.cs:**

namespace RetailInventory.Models;

public class Product

{

    public int ProductId { get; set; }

    public string Name { get; set; }

    public int Stock { get; set; }

    public int CategoryId { get; set; }

    public Category Category { get; set; }

}

**In Category.cs:**

namespace RetailInventory.Models;

public class Category

{

    public int CategoryId { get; set; }

    public string Name { get; set; }

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

}

**Creating Data folder:**

**In 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 optionsBuilder)

    {

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

    }

}

**In Program.cs:**

using RetailInventory.Data;

using RetailInventory.Models;

using Microsoft.EntityFrameworkCore;

var context = new AppDbContext();

context.Database.EnsureCreated();

if (!context.Categories.Any())

{

    context.Categories.AddRange(

        new Category { Name = "Electronics" },

        new Category { Name = "Groceries" },

        new Category { Name = "Clothing" }

    );

    context.SaveChanges();

}

bool running = true;

while (running)

{

    Console.WriteLine("\nRetail Inventory Menu");

    Console.WriteLine("1. View Products");

    Console.WriteLine("2. Add Product");

    Console.WriteLine("3. Update Product");

    Console.WriteLine("4. Delete Product");

    Console.WriteLine("5. Exit");

    Console.Write("Select an option: ");

    var choice = Console.ReadLine();

    switch (choice)

    {

        case "1":

            ViewProducts();

            break;

        case "2":

            AddProduct();

            break;

        case "3":

            UpdateProduct();

            break;

        case "4":

            DeleteProduct();

            break;

        case "5":

            running = false;

            break;

        default:

            Console.WriteLine("Invalid option.");

            break;

    }

}

void ViewProducts()

{

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

    if (!products.Any())

    {

        Console.WriteLine("No products found.");

        return;

    }

    Console.WriteLine("\nProduct List:");

    foreach (var p in products)

        Console.WriteLine($"{p.ProductId}. {p.Name} ({p.Category.Name}) - Stock: {p.Stock}");

}

void AddProduct()

{

    Console.Write("Enter product name: ");

    var name = Console.ReadLine();

    Console.Write("Enter stock: ");

    if (!int.TryParse(Console.ReadLine(), out int stock))

    {

        Console.WriteLine("Invalid stock.");

        return;

    }

    Console.Write("Enter category name: ");

    var categoryName = Console.ReadLine();

    var category = context.Categories.FirstOrDefault(c => c.Name.ToLower() == categoryName.ToLower());

    if (category == null)

    {

        category = new Category { Name = categoryName };

        context.Categories.Add(category);

        context.SaveChanges();

        Console.WriteLine($"Category '{categoryName}' created.");

    }

    var product = new Product

    {

        Name = name,

        Stock = stock,

        CategoryId = category.CategoryId

    };

    context.Products.Add(product);

    context.SaveChanges();

    Console.WriteLine("Product added.");

}

void UpdateProduct()

{

    ViewProducts();

    Console.Write("Enter Product ID to update: ");

    if (!int.TryParse(Console.ReadLine(), out int id))

    {

        Console.WriteLine("Invalid ID.");

        return;

    }

    var product = context.Products.Find(id);

    if (product == null)

    {

        Console.WriteLine("Product not found.");

        return;

    }

    Console.Write("New name: ");

    var name = Console.ReadLine();

    if (!string.IsNullOrWhiteSpace(name))

        product.Name = name;

    Console.Write("New stock: ");

    var stockInput = Console.ReadLine();

    if (int.TryParse(stockInput, out int newStock))

        product.Stock = newStock;

    context.SaveChanges();

    Console.WriteLine("Product updated.");

}

void DeleteProduct()

{

    ViewProducts();

    Console.Write("Enter Product ID to delete: ");

    if (!int.TryParse(Console.ReadLine(), out int id))

    {

        Console.WriteLine("Invalid ID.");

        return;

    }

    var product = context.Products.Find(id);

    if (product == null)

    {

        Console.WriteLine("Product not found.");

        return;

    }

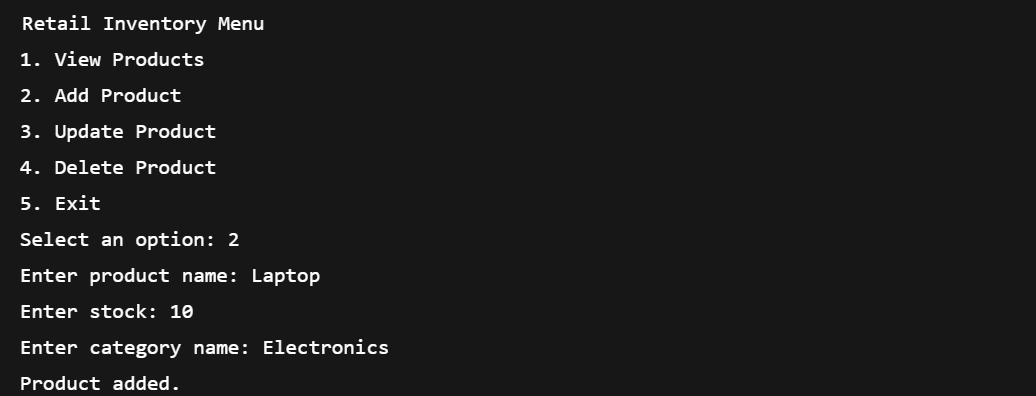
    context.Products.Remove(product);

    context.SaveChanges();

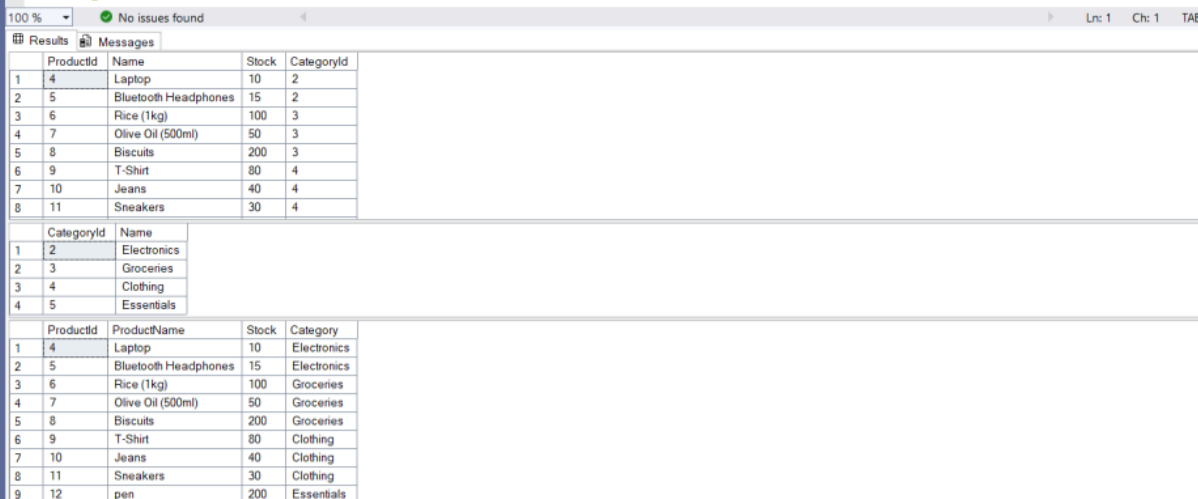
    Console.WriteLine("Product deleted.");

}

Output:



Output in SSMS:



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

**Code: (In Product.cs):**

namespace RetailInventory.Models;

public class Product

{

    public int ProductId { get; set; }

    public string Title { get; set; }

    public decimal Cost { get; set; }

    public int CategoryRefId { get; set; }

    public Category Category { get; set; }

}

**In Category.cs:**

namespace RetailInventory.Models;

public class Category

{

    public int CategoryId { get; set; }

    public string CategoryName { get; set; }

    public string Description { get; set; }

    public DateTime CreatedAt { get; set; } = DateTime.Now;

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

}

**In Program.cs:**

using RetailInventory.Data;

using RetailInventory.Models;

var context = new AppDbContext();

context.Database.EnsureCreated();

if (!context.Categories.Any())

{

    var electronics = new Category

    {

        CategoryName = "Electronics",

        Description = "Devices like phones, laptops, and accessories."

    };

    var groceries = new Category

    {

        CategoryName = "Groceries",

        Description = "Everyday food and household items."

    };

    var clothing = new Category

    {

        CategoryName = "Clothing",

        Description = "Men's and women's apparel."

    };

    context.Categories.AddRange(electronics, groceries, clothing);

    context.Products.AddRange(

        new Product

        {

            ProductName = "Laptop",

            ProductPrice = 59999,

            Category = electronics

        },

        new Product

        {

            ProductName = "Smartphone",

            ProductPrice = 39999,

            Category = electronics

        },

        new Product

        {

            ProductName = "Rice - 5kg",

            ProductPrice = 450,

            Category = groceries

        },

        new Product

        {

            ProductName = "T-Shirt",

            ProductPrice = 799,

            Category = clothing

        }

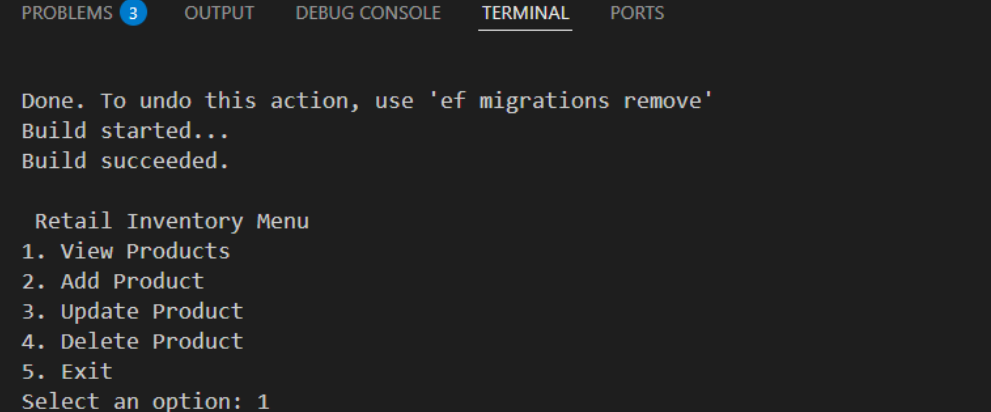
    );

    context.SaveChanges();

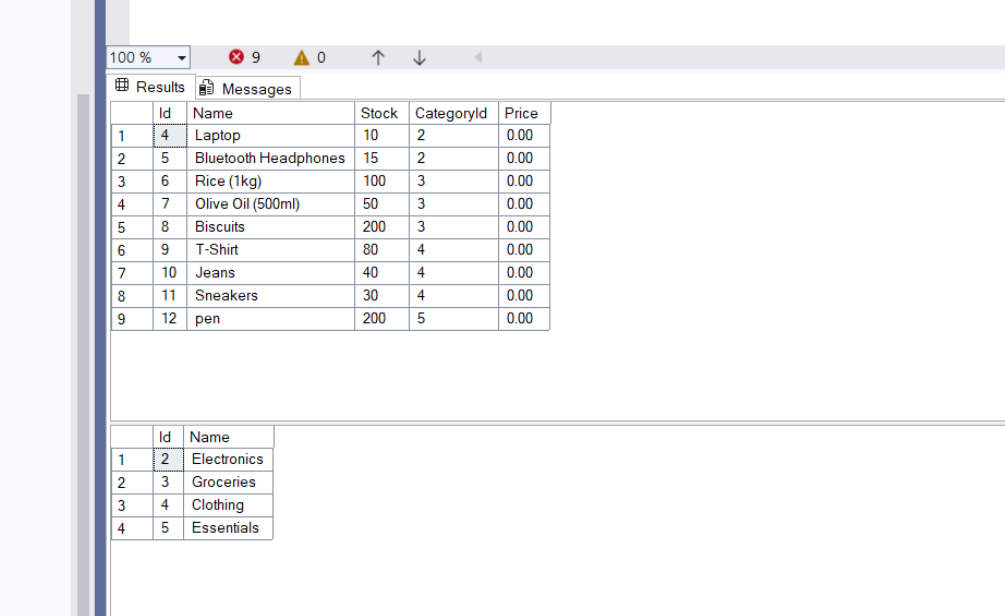
    Console.WriteLine("Initial categories and products added to the database.");

}

**Output:**



In SSMS :



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

**Code:**

dotnet tool install --global dotnet-ef

dotnet ef migrations add InitialCreate

dotnet ef database update

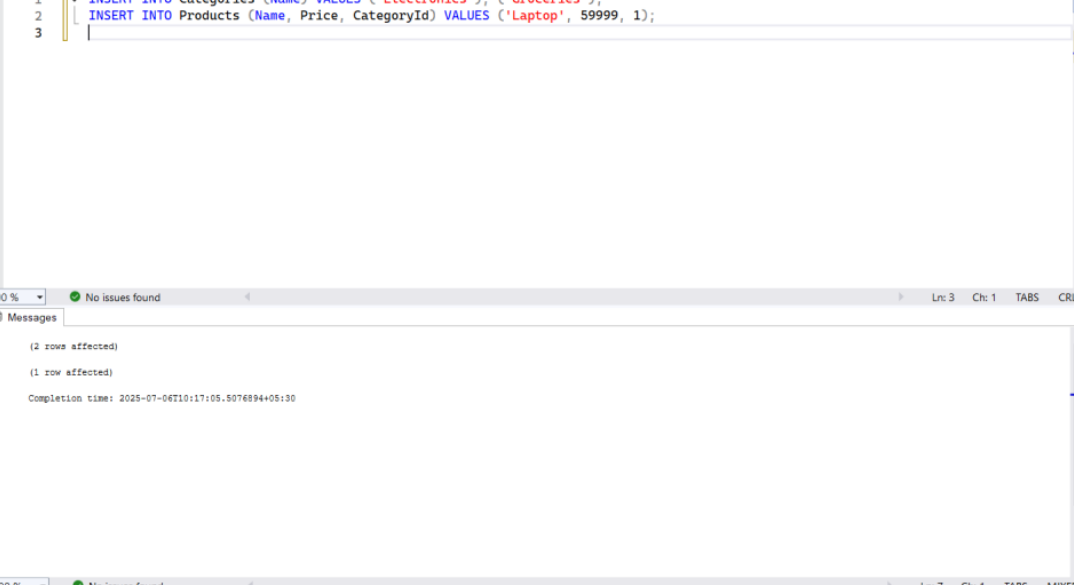
* **Opened SSMS**
* **Connected to: localdb**
* **Navigated to RetailDB → Tables**
* **Verified the presence of:**
* **Products**
* **Categories**
* **EFMigrationsHistory**

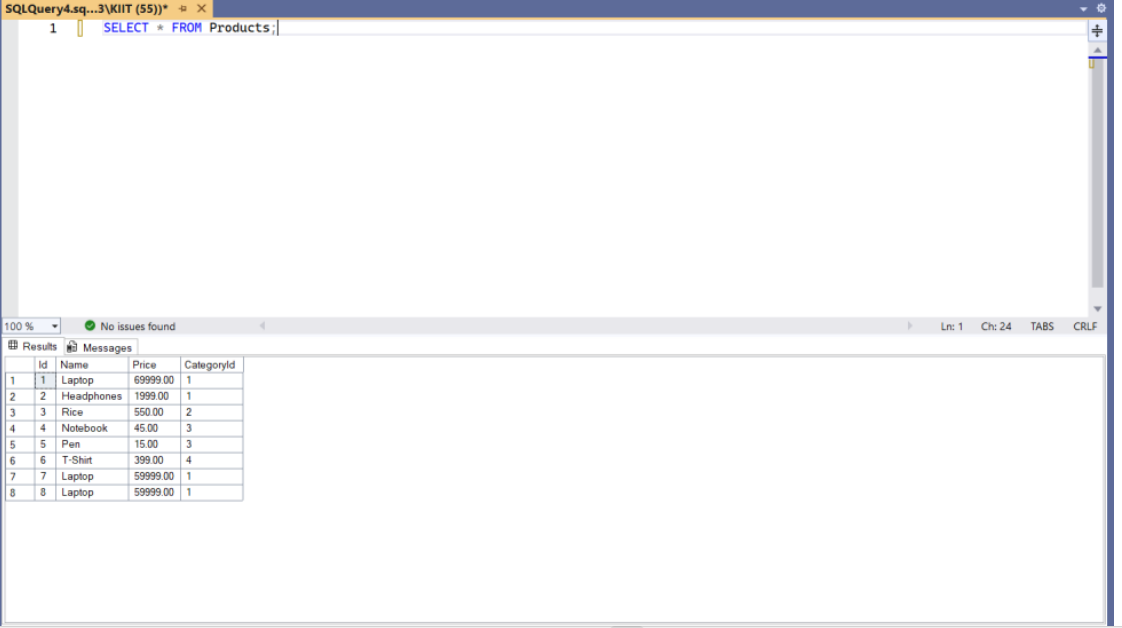
**Output:**

INSERT INTO Categories (Name) VALUES ('Electronics'), ('Groceries');

INSERT INTO Products (Name, Price, CategoryId) VALUES ('Laptop', 59999, 1);

IN SSMS :





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

**Code:**

**In Program.cs:**

using RetailInventory.Data;

using RetailInventory.Models;

using var context = new AppDbContext();

var electronics = new Category { CategoryName = "Electronics", Description = "Devices and gadgets" };

var groceries = new Category { CategoryName = "Groceries", Description = "Daily essentials and food items" };

var clothing = new Category { CategoryName = "Clothing", Description = "Apparel and garments" };

var furniture = new Category { CategoryName = "Furniture", Description = "Household and office furniture" }

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

var product1 = new Product { ProductName = "Laptop", ProductPrice = 75000, Stock = 15, Category = electronics };

var product2 = new Product { ProductName = "Smartphone", ProductPrice = 55000, Stock = 25, Category = electronics };

var product3 = new Product { ProductName = "Rice Bag - 10kg", ProductPrice = 1200, Stock = 50, Category = groceries };

var product4 = new Product { ProductName = "Milk (1L)", ProductPrice = 60, Stock = 100, Category = groceries };

var product5 = new Product { ProductName = "T-Shirt", ProductPrice = 799, Stock = 40, Category = clothing };

var product6 = new Product { ProductName = "Dining Table", ProductPrice = 15999, Stock = 5, Category = furniture };

var product7 = new Product { ProductName = "Office Chair", ProductPrice = 3499, Stock = 20, Category = furniture };

await context.Products.AddRangeAsync(product1, product2, product3, product4, product5, product6, product7);

await context.SaveChangesAsync();

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

Console.WriteLine($"Total categories: {await context.Categories.CountAsync()}");

Console.WriteLine($"Total products: {await context.Products.CountAsync()}");

**Output:**



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

**Code:**

**In Program.cs:**

using System;

using System.Linq;

using Microsoft.EntityFrameworkCore;

using RetailInventory.Data;

using RetailInventory.Models;

using var context = new AppDbContext();

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

Console.WriteLine("All Products:");

foreach (var p in products)

    Console.WriteLine($"{p.ProductName} - ₹{p.ProductPrice} ({p.Category.CategoryName})");

Console.WriteLine();

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

if (product != null)

    Console.WriteLine($"Product with ID 1: {product.ProductName} - ₹{product.ProductPrice}")

Console.WriteLine();

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

if (expensive != null)

    Console.WriteLine($"First product over ₹50000: {expensive.ProductName}");

Console.WriteLine();

var groceryItems = await context.Products

    .Where(p => p.Category.CategoryName == "Groceries")

    .ToListAsync();

Console.WriteLine("Groceries:");

foreach (var g in groceryItems)

    Console.WriteLine($"{g.ProductName} - ₹{g.ProductPrice}");

Console.WriteLine();

var sorted = await context.Products

    .OrderByDescending(p => p.ProductPrice)

    .ToListAsync();

Console.WriteLine("Products by Price (High to Low):");

foreach (var s in sorted)

    Console.WriteLine($"{s.ProductName} - ₹{s.ProductPrice}");

Console.WriteLine();

var productSummaries = await context.Products

    .Select(p => new { p.ProductName, p.Stock })

    .ToListAsync();

Console.WriteLine("Product Stocks:");

foreach (var item in productSummaries)

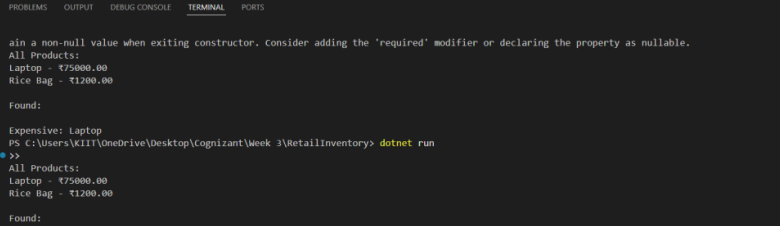
    Console.WriteLine($"{item.ProductName}: {item.Stock} in stock");

Console.WriteLine();

var total = await context.Products.CountAsync();

Console.WriteLine($"Total Products: {total}");

Output :



In SSMS :

