**LAB 5:EF CORE 8.0 HOL**

**Aim:** Retrieving Data from the Database

**Scenario:** The store wants to display product details on the dashboard.

**Models\Product.cs:**

namespace RetailInventoryApp.Models

{

public class Product

{

public int ProductId { 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!;

public int SupplierId { get; set; }

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

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

}

}

**Models\category.cs:**

namespace RetailInventoryApp.Models

{

public class Category

{

public int CategoryId { get; set; }

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

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

}

}

**Models\Stock.cs:**

namespace RetailInventoryApp.Models

{

public class Stock

{

public int StockId { get; set; }

public int QuantityAvailable { get; set; }

public DateTime LastChecked { get; set; }

public int ProductId { get; set; }

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

}

}

**Models\Supplier.cs:**

namespace RetailInventoryApp.Models

{

public class Supplier

{

public int SupplierId { get; set; }

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

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

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

}

}

**Data\AppDbContext.cs:**

using Microsoft.EntityFrameworkCore;

using RetailInventoryApp.Models;

namespace RetailInventoryApp.Data

{

public class AppDbContext : DbContext

{

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

public DbSet<Supplier> Suppliers { get; set; }

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

public DbSet<Stock> Stocks { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlite("Data Source=retail\_inventory.db");

}

}

}

**Program.cs:**

using RetailInventoryApp.Data;

using RetailInventoryApp.Models;

using Microsoft.EntityFrameworkCore;

class Program

{

static async Task Main()

{

using var context = new AppDbContext();

await context.Database.EnsureDeletedAsync();

await context.Database.EnsureCreatedAsync();

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

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

var supplier1 = new Supplier { SupplierName = "TechZone", ContactEmail = "tech@zone.com" };

var supplier2 = new Supplier { SupplierName = "DailyMart", ContactEmail = "contact@dailymart.com" };

var product1 = new Product

{

Name = "Laptop",

Price = 75000,

Category = electronics,

Supplier = supplier1,

Stock = new Stock { QuantityAvailable = 10, LastChecked = DateTime.Now }

};

var product2 = new Product

{

Name = "Rice Bag",

Price = 1200,

Category = groceries,

Supplier = supplier2,

Stock = new Stock { QuantityAvailable = 50, LastChecked = DateTime.Now }

};

var product3 = new Product

{

Name = "Smartphone",

Price = 85000,

Category = electronics,

Supplier = supplier1,

Stock = new Stock { QuantityAvailable = 5, LastChecked = DateTime.Now }

};

await context.Products.AddRangeAsync(product1, product2, product3);

await context.SaveChangesAsync();

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

var products = await context.Products

.Include(p => p.Category)

.Include(p => p.Supplier)

.Include(p => p.Stock)

.ToListAsync();

Console.WriteLine(" All Products:");

foreach (var p in products)

{

Console.WriteLine($"{p.ProductId}: {p.Name} | ₹{p.Price} | Qty: {p.Stock.QuantityAvailable} | Supplier: {p.Supplier.SupplierName}");

}

Console.Write("\n Enter Product ID to search: ");

if (int.TryParse(Console.ReadLine(), out int searchId))

{

var found = await context.Products

.Include(p => p.Category)

.Include(p => p.Supplier)

.Include(p => p.Stock)

.FirstOrDefaultAsync(p => p.ProductId == searchId);

if (found != null)

{

Console.WriteLine($"\n Found: {found.Name}");

Console.WriteLine($"Price: ₹{found.Price}");

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

Console.WriteLine($"Supplier: {found.Supplier.SupplierName}");

Console.WriteLine($"Stock: {found.Stock.QuantityAvailable} (Last Checked: {found.Stock.LastChecked:dd-MM-yyyy})");

}

else

{

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

}

}

else

{

Console.WriteLine(" Invalid Product ID.");

}

var expensiveProduct = await context.Products

.Include(p => p.Stock)

.OrderByDescending(p =>(double) p.Price)

.FirstOrDefaultAsync();

if (expensiveProduct != null)

{

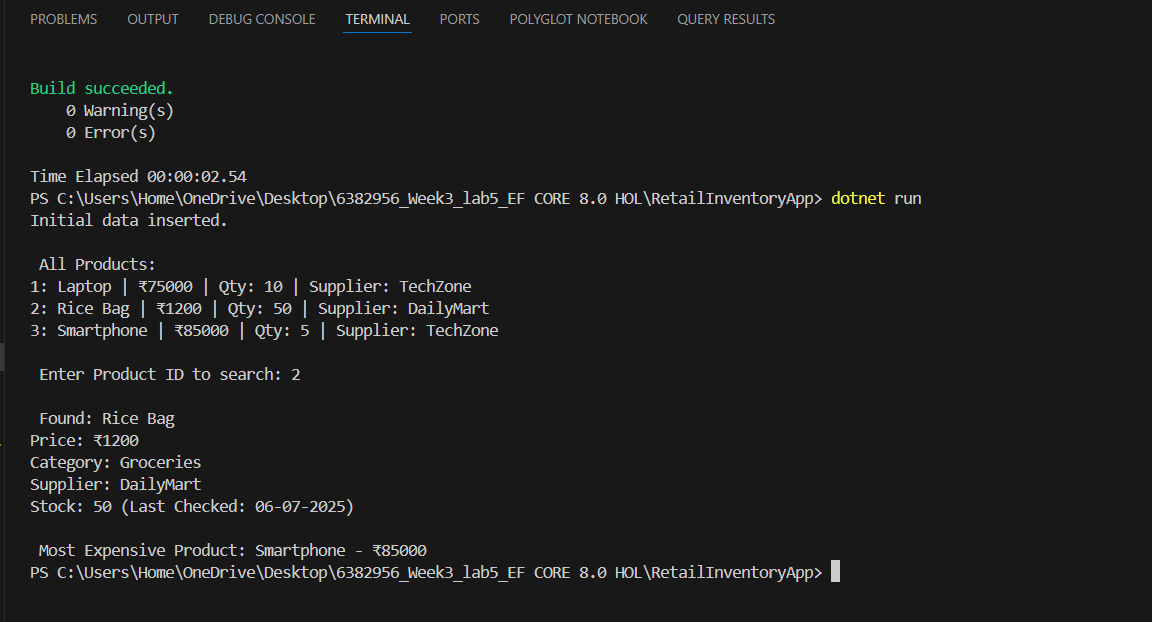
Console.WriteLine($"\n Most Expensive Product: {expensiveProduct.Name} - ₹{expensiveProduct.Price}");

}

}

}

**Output:**

****