**LAB 7: EF CORE 8.0 HOL**

**Aim:** Writing Queries with LINQ

**Scenario:** The store wants to filter and sort products for reporting.

**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 ProductId { get; set; }

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

public int QuantityAvailable { get; set; }

public DateTime LastChecked { get; set; }

}

**Models\Supplier.cs:**

namespace RetailInventoryApp.Models;

public class Supplier

{

public int SupplierId { get; set; }

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

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

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

}

**Data\RetailContext.cs:**

using Microsoft.EntityFrameworkCore;

using RetailInventoryApp.Models;

namespace RetailInventoryApp.Data;

public class RetailContext : DbContext

{

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

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

public DbSet<Supplier> Suppliers => Set<Supplier>();

public DbSet<Stock> Stocks => Set<Stock>();

protected override void OnConfiguring(DbContextOptionsBuilder options)

=> options.UseSqlite("Data Source=retail.db");

}

**Program.cs:**

using RetailInventoryApp.Data;

using RetailInventoryApp.Models;

using Microsoft.EntityFrameworkCore;

using var context = new RetailContext();

await context.Database.EnsureCreatedAsync();

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

{

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

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

var supplier = new Supplier { Name = "Tech Supplier", Email = "tech@supplier.com" };

var p1 = new Product

{

Name = "Laptop",

Price = 75000,

Category = electronics,

Supplier = supplier,

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

};

var p2 = new Product

{

Name = "Rice Bag",

Price = 1200,

Category = groceries,

Supplier = supplier,

Stock = new Stock { QuantityAvailable = 100, LastChecked = DateTime.Now }

};

var p3 = new Product

{

Name = "Smartphone",

Price = 30000,

Category = electronics,

Supplier = supplier,

Stock = new Stock { QuantityAvailable = 30, LastChecked = DateTime.Now }

};

await context.AddRangeAsync(electronics, groceries, supplier, p1, p2, p3);

await context.SaveChangesAsync();

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

}

var filtered = await context.Products

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

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

.ToListAsync();

Console.WriteLine(" Filtered Products ((double)Price > ₹1000):");

foreach (var p in filtered)

{

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

}

var productDTOs = await context.Products

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

.ToListAsync();

Console.WriteLine("\n DTO Projection (Name + Price):");

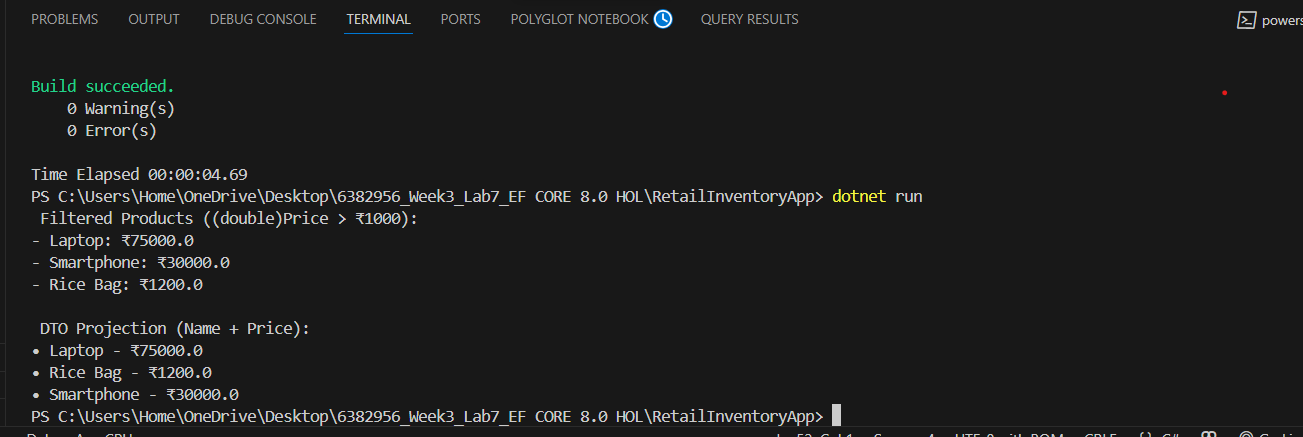
foreach (var dto in productDTOs)

{

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

}

**Output:**

****