Lab Report: Smart Retail Inventory System using EF Core and SQLite

# AIM:

To develop a Smart Retail Inventory System using C# with Entity Framework Core and SQLite. The system allows the retail store to track products, categories, stock levels, suppliers, and provides alerts for low stock and other analytics.

# SCENARIO:

We are building an inventory management system for a retail store using EF Core with SQLite. The system should store information about Products, Categories, Stock, and Suppliers.

It should provide features like:  
- Displaying all products  
- Identifying low stock items  
- Finding the most expensive product  
- Storing supplier and timestamp information

# MODEL CLASSES:

## Product.cs

namespace RetailInventory.Models  
{  
 public class Product  
 {  
 public int ProductId { get; set; }  
 public string? Name { get; set; }  
 public decimal Price { get; set; }  
 public DateTime AddedOn { get; set; } = DateTime.Now;  
  
 public int CategoryId { get; set; }  
 public Category? Category { get; set; }  
  
 public int SupplierId { get; set; }  
 public Supplier? Supplier { get; set; }  
  
 public Stock? Stock { get; set; }  
 }  
}

## Category.cs

namespace RetailInventory.Models  
{  
 public class Category  
 {  
 public int CategoryId { get; set; }  
 public string? Name { get; set; }  
 public List<Product>? Products { get; set; }  
 }  
}

## Stock.cs

namespace RetailInventory.Models  
{  
 public class Stock  
 {  
 public int StockId { get; set; }  
 public int Quantity { get; set; }  
  
 public int ProductId { get; set; }  
 public Product? Product { get; set; }  
 }  
}

## Supplier.cs

namespace RetailInventory.Models  
{  
 public class Supplier  
 {  
 public int SupplierId { get; set; }  
 public string? Name { get; set; }  
 public string? ContactInfo { get; set; }  
 public List<Product>? Products { get; set; }  
 }  
}

# DATABASE CONTEXT CLASS (RetailContext.cs):

using Microsoft.EntityFrameworkCore;  
using RetailInventory.Models;  
  
namespace RetailInventory.Data  
{  
 public class RetailContext : DbContext  
 {  
 public DbSet<Product> Products => Set<Product>();  
 public DbSet<Category> Categories => Set<Category>();  
 public DbSet<Stock> Stocks => Set<Stock>();  
 public DbSet<Supplier> Suppliers => Set<Supplier>();  
  
 protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)  
 {  
 optionsBuilder.UseSqlite("Data Source=retail\_inventory.db");  
 }  
  
 protected override void OnModelCreating(ModelBuilder modelBuilder)  
 {  
 modelBuilder.Entity<Product>()  
 .HasOne(p => p.Stock)  
 .WithOne(s => s.Product)  
 .HasForeignKey<Stock>(s => s.ProductId);  
 }  
 }  
}

# PROGRAM (Program.cs):

using RetailInventory.Models;  
using RetailInventory.Data;  
using Microsoft.EntityFrameworkCore;  
  
using (var context = new RetailContext())  
{  
 context.Database.EnsureCreated();  
 SeedData(context);  
 ShowMenu(context);  
}  
  
static void SeedData(RetailContext context)  
{  
 if (!context.Categories.Any())  
 {  
 var books = new Category { Name = "Books" };  
 var electronics = new Category { Name = "Electronics" };  
  
 var supplier1 = new Supplier { Name = "TechSupplier", ContactInfo = "tech@supply.com" };  
 var supplier2 = new Supplier { Name = "BookWorld", ContactInfo = "info@bookworld.com" };  
  
 var p1 = new Product { Name = "C# Guide", Price = 499, Category = books, Supplier = supplier2 };  
 var p2 = new Product { Name = "Laptop", Price = 55000, Category = electronics, Supplier = supplier1 };  
  
 var s1 = new Stock { Product = p1, Quantity = 5 };  
 var s2 = new Stock { Product = p2, Quantity = 50 };  
  
 context.AddRange(books, electronics, supplier1, supplier2, p1, p2, s1, s2);  
 context.SaveChanges();  
 }  
}  
  
static void ShowMenu(RetailContext context)  
{  
 while (true)  
 {  
 Console.WriteLine("\n🛒 Smart Retail Menu");  
 Console.WriteLine("1. Show All Products");  
 Console.WriteLine("2. Show Low Stock (below 10)");  
 Console.WriteLine("3. Most Expensive Product");  
 Console.WriteLine("4. Exit");  
 Console.Write("Choose an option: ");  
 string? option = Console.ReadLine();  
  
 if (option == "1") ShowProducts(context);  
 else if (option == "2") ShowLowStock(context);  
 else if (option == "3") ShowMostExpensive(context);  
 else if (option == "4") break;  
 else Console.WriteLine("❌ Invalid option.");  
 }  
}  
  
static void ShowProducts(RetailContext context)  
{  
 var items = context.Products.Include(p => p.Category).Include(p => p.Stock).Include(p => p.Supplier).ToList();  
 foreach (var p in items)  
 {  
 Console.WriteLine($"📦 {p.Name} | ₹{p.Price} | Category: {p.Category?.Name} | Stock: {p.Stock?.Quantity} | Supplier: {p.Supplier?.Name}");  
 }  
}  
  
static void ShowLowStock(RetailContext context)  
{  
 var items = context.Stocks.Include(s => s.Product).Where(s => s.Quantity < 10).ToList();  
 Console.WriteLine("⚠️ Low Stock Items:");  
 foreach (var s in items)  
 {  
 Console.WriteLine($"🔻 {s.Product?.Name} - Only {s.Quantity} left!");  
 }  
}  
  
static void ShowMostExpensive(RetailContext context)  
{  
 var product = context.Products.OrderByDescending(p => p.Price).FirstOrDefault();  
 if (product != null)  
 {  
 Console.WriteLine($"💰 Most Expensive: {product.Name} - ₹{product.Price}");  
 }  
}

# Output:

# output1