ENTITY FRAME WORK:

using System;

using System.Collections.Generic;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

namespace RetailInventory

{

public class Category

{

public int Id { get; set; }

public string Name { get; set; }

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

}

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

public int CategoryId { get; set; }

public Category Category { get; set; }

}

public class AppDbContext : DbContext

{

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

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

protected override void OnConfiguring(DbContextOptionsBuilder builder)

{

builder.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=RetailDb;Trusted\_Connection=True;");

}

}

class Program

{

static async Task Main()

{

// Apply migrations and create/update database

using (var ctx = new AppDbContext())

{

await ctx.Database.MigrateAsync();

}

// Insert initial data

using (var ctx = new AppDbContext())

{

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

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

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

var laptop = new Product { Name = "Laptop", Price = 75000M, Category = electronics };

var rice = new Product { Name = "Rice Bag", Price = 1200M, Category = groceries };

await ctx.Products.AddRangeAsync(laptop, rice);

await ctx.SaveChangesAsync();

}

// Retrieve & display products

using (var ctx = new AppDbContext())

{

Console.WriteLine("\*\* All Products \*\*");

var all = await ctx.Products

.Include(p => p.Category)

.ToListAsync();

foreach (var p in all)

Console.WriteLine($"{p.Id}: {p.Name} — {p.Category.Name} — ₹{p.Price}");

Console.WriteLine("\n\*\* Product with ID = 1 \*\*");

var prod = await ctx.Products.FindAsync(1);

Console.WriteLine(prod != null

? $"{prod.Id}: {prod.Name} — ₹{prod.Price}"

: "Not found");

Console.WriteLine("\n\*\* Most Expensive Product (>₹50000) \*\*");

var expensive = await ctx.Products

.FirstOrDefaultAsync(p => p.Price > 50000M);

Console.WriteLine(expensive != null

? $"{expensive.Name} — ₹{expensive.Price}"

: "No such product");

}

}

}

}

Output:

