**WEEK 3**

**Lab-1:Understanding ORM with a Retail Inventory System**

**Models/Category.cs**

namespace RetailInventory.Models

{

public class Category

{

public int CategoryId { get; set; }

public string Name { get; set; }

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

}

}

**Models/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; }

}

}

**Data/RetailContext.cs**

using Microsoft.EntityFrameworkCore;

using RetailInventory.Models;

namespace RetailInventory.Data

{

public class RetailContext : DbContext

{

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

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

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

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

}

}

}

**Program.cs**

using RetailInventory.Data;

using RetailInventory.Models;

using (var context = new RetailContext())

{

// Ensure database is created

context.Database.EnsureCreated();

// Seed Data

if (!context.Categories.Any())

{

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

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

context.Categories.AddRange(electronics, groceries);

context.Products.AddRange(

new Product { Name = "Laptop", Stock = 10, Category = electronics },

new Product { Name = "TV", Stock = 5, Category = electronics },

new Product { Name = "Apple", Stock = 100, Category = groceries }

);

context.SaveChanges();

}

// Display products

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

foreach (var p in products)

{

Console.WriteLine($"Product: {p.Name}, Stock: {p.Stock}, Category: {p.Category.Name}");

}

}

### ✔ What is ORM?

ORM maps C# objects (Product, Category) to database tables.

You interact with objects, and EF Core translates that into SQL queries.

### ✔ EF Core vs EF Framework

EF Core is lightweight, modern (supports async, LINQ), cross-platform.

EF6 is older, Windows-only, but more mature.

### ✔ EF Core 8 Features Used:

Not directly shown, but you can extend with:

JSON columns (for nested properties)

Compiled models for better performance

Interceptors for logging, validation

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

**Models/Category.cs**

using System.Collections.Generic;

namespace RetailInventory.Models

{

public class Category

{

public int Id { get; set; }

public string Name { get; set; }

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

}

}

**Models/Product.cs**

namespace RetailInventory.Models

{

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; }

}

}

**Data/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)

{

// Replace with your actual connection string

optionsBuilder.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=RetailStoreDB;Trusted\_Connection=True;");

}

}

}

**Program.cs**

using RetailInventory.Data;

using RetailInventory.Models;

using (var context = new AppDbContext())

{

context.Database.EnsureCreated();

if (!context.Categories.Any())

{

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

context.Categories.Add(electronics);

context.Products.Add(new Product { Name = "Smartphone", Price = 699.99M, Category = electronics });

context.SaveChanges();

}

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

foreach (var p in products)

{

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

}

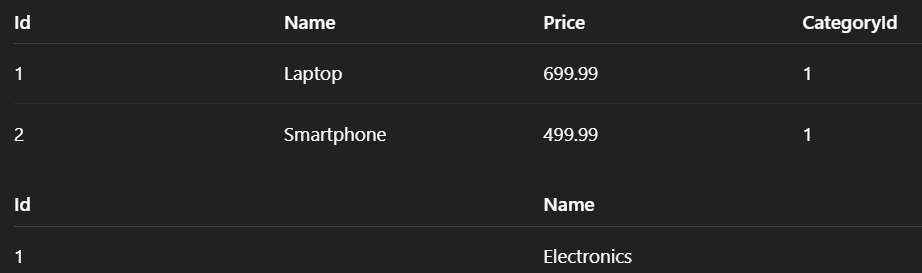
}

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

dotnet tool install --global dotnet-ef

dotnet ef migrations add InitialCreate

dotnet ef database update



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

**Program.cs**

using System;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

using RetailInventory.Data;

using RetailInventory.Models;

class Program

{

static async Task Main(string[] args)

{

using var context = new AppDbContext();

// Ensure DB is created

await context.Database.EnsureCreatedAsync();

// Only insert if data doesn't exist

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

{

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

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

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

var product1 = new Product { Name = "Laptop", Price = 75000, Category = electronics };

var product2 = new Product { Name = "Rice Bag", Price = 1200, Category = groceries };

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

await context.SaveChangesAsync();

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

}

else

{

Console.WriteLine("Data already exists. Skipping insert.");

}

// Display data

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

foreach (var p in products)

{

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

}

}

}

**DB commands**

SELECT \* FROM Categories;

SELECT \* FROM Products;

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

**Program.cs**

using System;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

using RetailInventory.Data;

using RetailInventory.Models;

class Program

{

static async Task Main(string[] args)

{

using var context = new AppDbContext();

// Ensure the DB exists

await context.Database.EnsureCreatedAsync();

Console.WriteLine("📦 All Products:");

var products = await context.Products.ToListAsync();

foreach (var p in products)

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

Console.WriteLine("\n🔍 Find Product by ID:");

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

Console.WriteLine($"Found: {product?.Name ?? "Not Found"}");

Console.WriteLine("\n💰 First Expensive Product (Price > ₹50000):");

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

Console.WriteLine($"Expensive: {expensive?.Name ?? "No expensive product"}");

}

}