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

**Install EF Core Packages:**



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

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RetailInventory

{

public class Category

{

public int Id { get; set; }

public string Name { get; set; }

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

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RetailInventory

{

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; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Microsoft.EntityFrameworkCore;

using System.Threading.Tasks;

namespace RetailInventory

{

public class AppDbContext : DbContext

{

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

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

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer(

"Server=BT-22051965\\SQLEXPRESS;Database=RetailDB;Trusted\_Connection=True;Encrypt=False;"

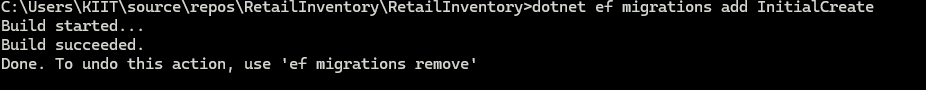
);

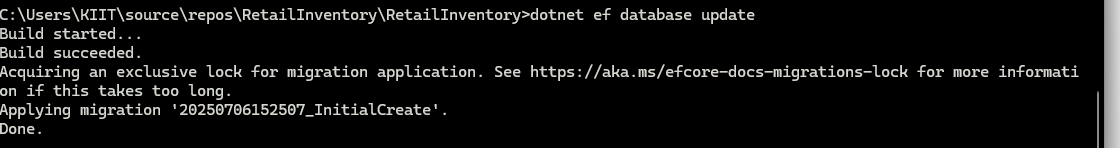
}

}

}

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





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

using System;

using System.Threading.Tasks;

namespace RetailInventory

{

internal class Program

{

static async Task Main(string[] args)

{

using var context = new AppDbContext();

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 = 7500, Category = electronics };

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

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

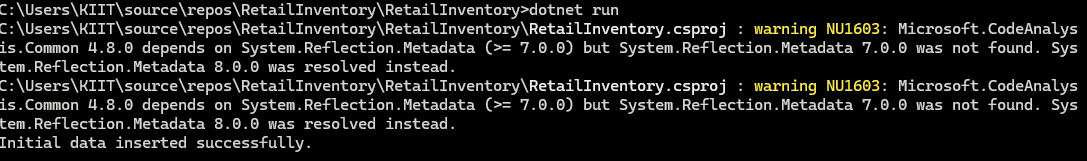
await context.SaveChangesAsync();

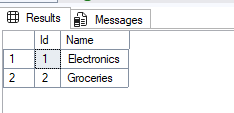
Console.WriteLine("Initial data inserted successfully.");

}

}

}







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

using Microsoft.EntityFrameworkCore;

using System;

using System.Threading.Tasks;

namespace RetailInventory

{

internal class Program

{

static async Task Main(string[] args)

{

using var context = new AppDbContext();

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 (1) ===");

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

Console.WriteLine($"Found: {product?.Name}");

Console.WriteLine("\n=== First Product with Price > 50,000 ===");

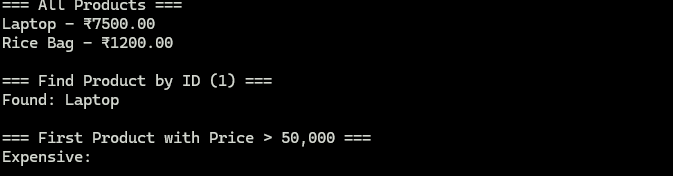
var expensive = await context.Products.FirstOrDefaultAsync(p => p.Price > 50000);

Console.WriteLine($"Expensive: {expensive?.Name}");

}

}

}



**Lab 6: Updating and Deleting Records**

using Microsoft.EntityFrameworkCore;

using System;

using System.Threading.Tasks;

namespace RetailInventory

{

internal class Program

{

static async Task Main(string[] args)

{

using var context = new AppDbContext();

Console.WriteLine("=== Updating Product Price ===");

var product = await context.Products.FirstOrDefaultAsync(p => p.Name == "Laptop");

if (product != null)

{

product.Price = 70000;

await context.SaveChangesAsync();

Console.WriteLine("Laptop price updated to ₹70000.");

}

Console.WriteLine("\n=== Deleting Product ===");

var toDelete = await context.Products.FirstOrDefaultAsync(p => p.Name == "Rice Bag");

if (toDelete != null)

{

context.Products.Remove(toDelete);

await context.SaveChangesAsync();

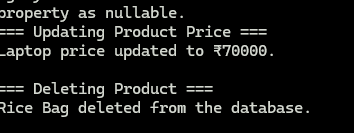
Console.WriteLine("Rice Bag deleted from the database.");

}

}

}

}



**Lab 7: Writing Queries with LINQ**

using Microsoft.EntityFrameworkCore;

using System;

using System.Threading.Tasks;

namespace RetailInventory

{

internal class Program

{

static async Task Main(string[] args)

{

using var context = new AppDbContext();

Console.WriteLine("=== Filtered & Sorted Products (Price > ₹1000) ===");

var filtered = await context.Products

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

.OrderByDescending(p => p.Price)

.ToListAsync();

foreach (var product in filtered)

{

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

}

Console.WriteLine("\n=== Projected into DTOs ===");

var productDTOs = await context.Products

.Select(p => new

{

p.Name,

p.Price

})

.ToListAsync();

foreach (var dto in productDTOs)

{

Console.WriteLine($"Name: {dto.Name}, Price: ₹{dto.Price}");

}

}

}

}

