# EF Core 8.0 Hands-On Labs

**Roll No.- 6363054**

## Lab 1: Understanding ORM

ORM (Object-Relational Mapping) allows developers to map C# classes to relational database tables. EF Core provides a way to interact with the database using objects instead of writing SQL directly.

## Lab 2: Setting Up the Database Context

### Category.cs

public class Category  
{  
 public int Id { get; set; }  
 public string Name { get; set; } = string.Empty;  
 public List<Product> Products { get; set; } = new();  
}

### Product.cs

public class Product  
{  
 public int Id { 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!;  
}

### AppDbContext.cs

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-22051323\SQLEXPRESS;Database=RetailInventoryDB;Trusted\_Connection=True;Encrypt=False;");  
 }  
}

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

Commands used:

dotnet ef migrations add InitialCreate

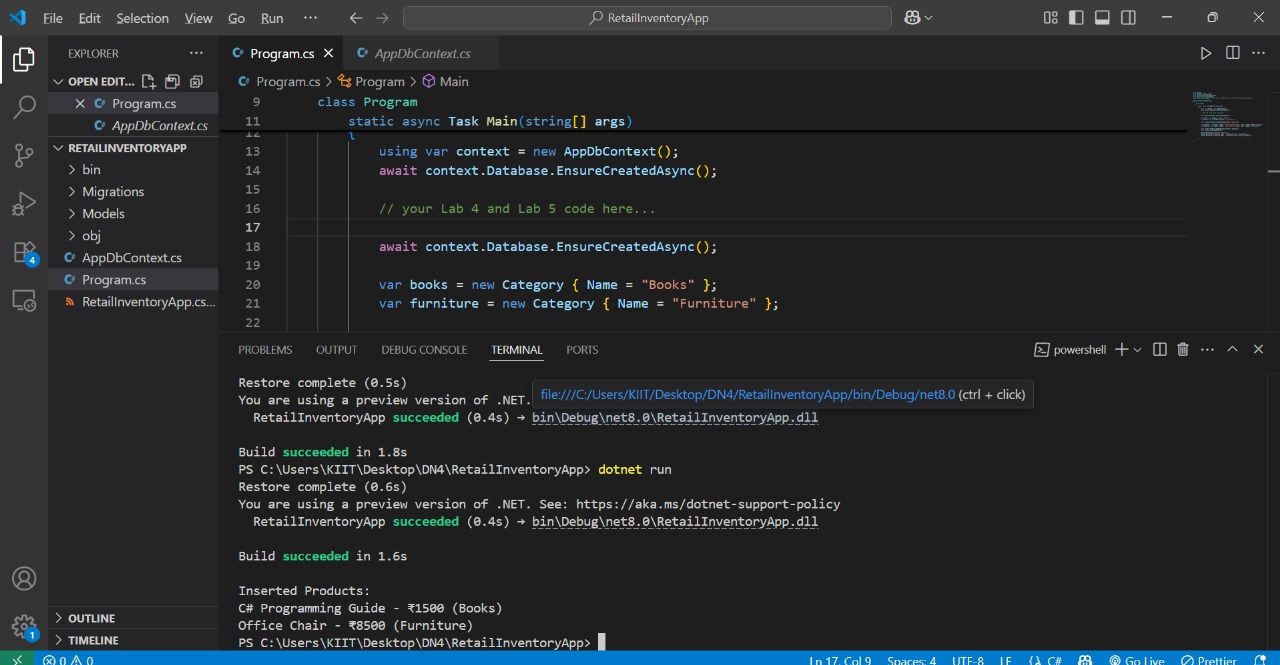
dotnet ef database update

## Lab 4: Inserting Initial Data into the Database

### Program.cs

var books = new Category { Name = "Books" };  
var furniture = new Category { Name = "Furniture" };  
  
await context.Categories.AddRangeAsync(books, furniture);  
  
var product1 = new Product { Name = "C# Programming Guide", Price = 1500, Category = books };  
var product2 = new Product { Name = "Office Chair", Price = 8500, Category = furniture };  
  
await context.Products.AddRangeAsync(product1, product2);  
await context.SaveChangesAsync();

### Output:



C# Programming Guide - ₹1500 (Books)  
Office Chair - ₹8500 (Furniture)

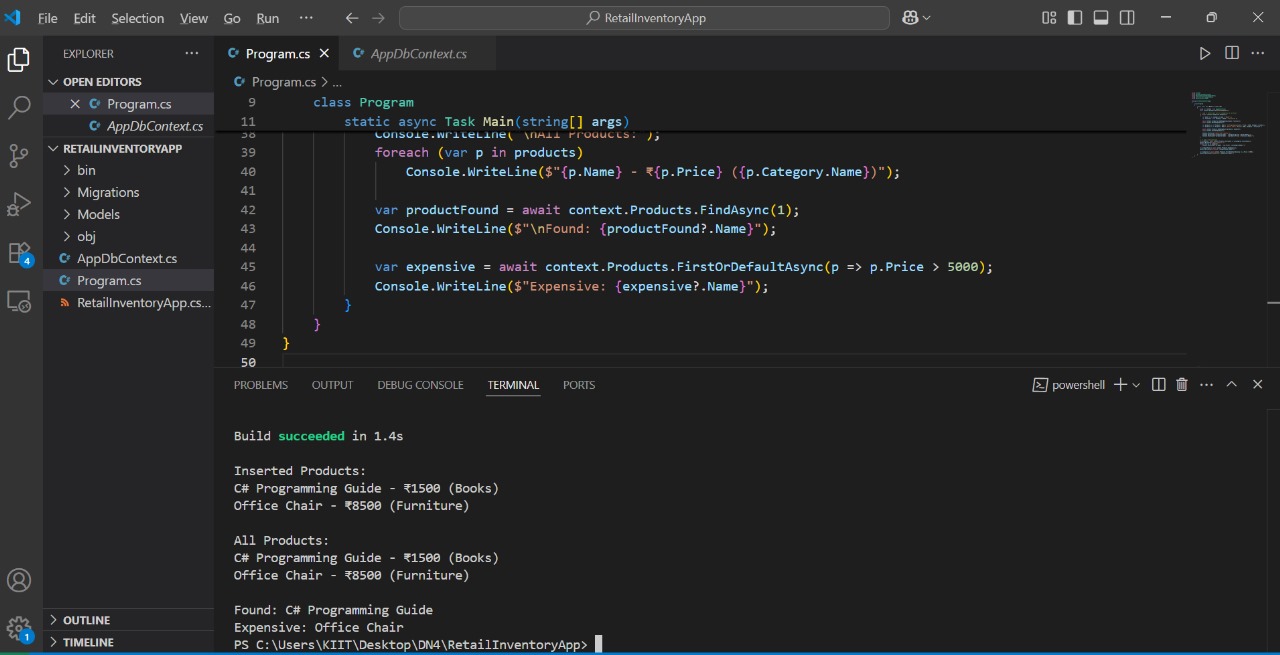
## Lab 5: Retrieving Data from the Database

var products = await context.Products.Include(p => p.Category).ToListAsync();  
foreach (var p in products)  
 Console.WriteLine($"{p.Name} - ₹{p.Price} ({p.Category.Name})");

var product = await context.Products.FindAsync(1);  
Console.WriteLine($"Found: {product?.Name}");

var expensive = await context.Products.FirstOrDefaultAsync(p => p.Price > 5000);  
Console.WriteLine($"Expensive: {expensive?.Name}");

### Output:



C# Programming Guide - ₹1500 (Books)  
Office Chair - ₹8500 (Furniture)  
  
Found: C# Programming Guide  
Expensive: Office Chair