DotNet-FSE Mandatory Hands-On

WEEK-3 **NAME:** Sri Ranjani Priya P

EXERCISE 1:

Understanding ORM with a Retail Inventory System

CODE:(C#)
File: Program.cs

```
using RetailInventoryApp.Models;
using System;
using System.Linq;
class Program
{
    static void Main()
    {
        using var context = new RetailContext();
        context.Database.EnsureCreated();
        if (!context.Products.Any())        {
            context.Products.Add(new Product { Name = "Monitor",
Quantity = 5, Price = 9999 });
            context.Products.Add(new Product { Name = "Keyboard",
Quantity = 10, Price = 1999 });
            context.SaveChanges();
            Console.WriteLine("Products inserted.\n");
        }
        else
        {
            Console.WriteLine("Products already exist. Skipping
insert.\n");
        }
        Console.WriteLine("Current Inventory:");
        foreach (var p in context.Products)
        {
            Console.WriteLine($"{p.ProductId}: {p.Name} -
{p.Quantity} pcs @ ₹{p.Price}");        }   }}
```

File: Product.cs

```
namespace RetailInventoryApp.Models;

public class Product
{
    public int ProductId { get; set; }
    public string Name { get; set; }
    public int Quantity { get; set; }
    public decimal Price { get; set; }
}
```

File: RetailContext.cs

```
using Microsoft.EntityFrameworkCore;
using RetailInventoryApp.Models;

public class RetailContext : DbContext
{
    public DbSet<Product> Products { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)     {
              optionsBuilder.UseSqlite("Data Source=retail.db");
          }
}
```

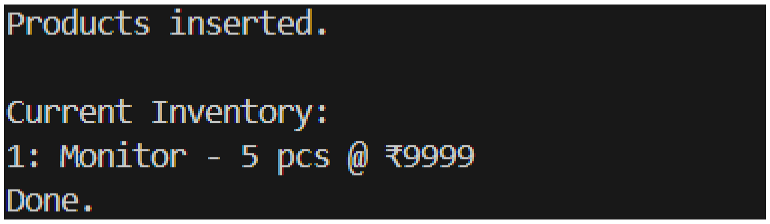
File: RetailInventoryApp.csproj

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net8.0</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.EntityFrameworkCore.Sqlite"
Version="8.0.0" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Design"
Version="8.0.0" />
```

```xml
  </ItemGroup>
</Project>
```

OUTPUT:



```
Products inserted.

Current Inventory:
1: Monitor - 5 pcs @ ₹9999
Done.
```

EXERCISE 2:

Setting Up the Database Context for a Retail Store

CODE:(C#)
File: RetailContext.cs

```csharp
using Microsoft.EntityFrameworkCore;
using RetailInventoryApp.Models;

public class RetailContext : DbContext
{
    public DbSet<Product> Products { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
    {
        optionsBuilder.UseSqlite("Data Source=retail.db");
    }
}
```

OUTPUT:

No direct output unless used with Program.cs

EXERCISE 3:

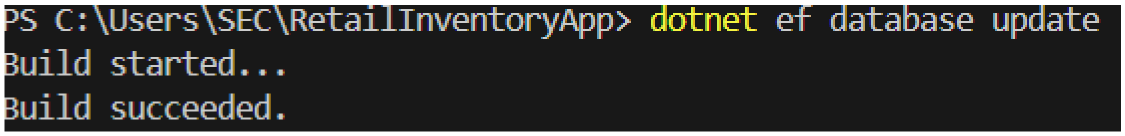
Using EF Core CLI to Create and Apply Migrations

CODE:

Commands:

```
dotnet ef migrations add InitialCreate
dotnet ef database update
```

OUTPUT:



```
PS C:\Users\SEC\RetailInventoryApp> dotnet ef database update
Build started...
Build succeeded.
```

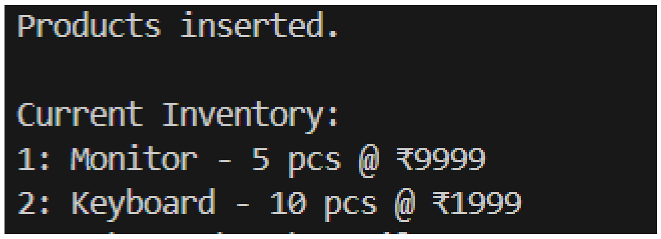
EXERCISE 4:

Inserting Initial Data into the Database

CODE:

Same code as Exercise 1

OUTPUT:



```
Products inserted.

Current Inventory:
1: Monitor - 5 pcs @ ₹9999
2: Keyboard - 10 pcs @ ₹1999
```

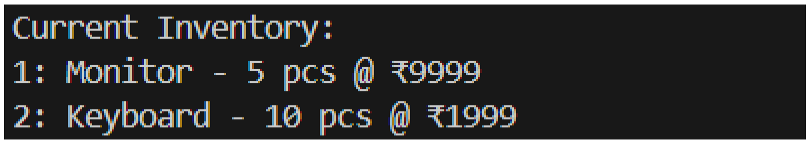
EXERCISE 5:

Retrieving Data from the Database

CODE:

```
Console.WriteLine("Current Inventory:");
        foreach (var p in context.Products)
        {
            Console.WriteLine($"{p.ProductId}: {p.Name} -
{p.Quantity} pcs @ ₹{p.Price}");
        }
```

OUTPUT:



```
Current Inventory:
1: Monitor - 5 pcs @ ₹9999
2: Keyboard - 10 pcs @ ₹1999
```

EXERCISE 6:

Updating and Deleting Records

CODE:

FILE: Program.cs

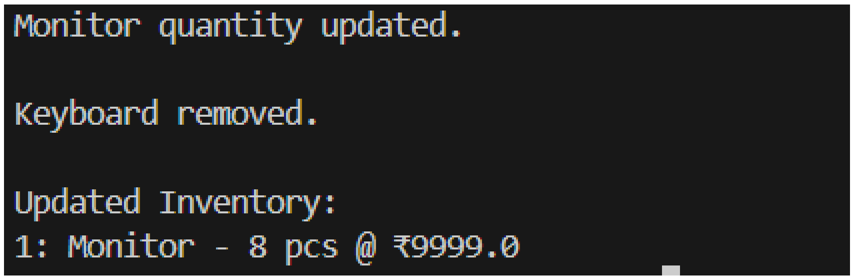
```
using RetailInventoryApp.Models;
using System;
using System.Linq;

class Program
{
    static void Main()
    {
        using var context = new RetailContext();
        context.Database.EnsureCreated();
         var monitor = context.Products.FirstOrDefault(p => p.Name
== "Monitor");
        if (monitor != null)
        {
            monitor.Quantity = 8;
            context.SaveChanges();
            Console.WriteLine("Monitor quantity updated.\n");
        }

                var keyboard = context.Products.FirstOrDefault(p =>
p.Name == "Keyboard");
        if (keyboard != null)
        {
            context.Products.Remove(keyboard);
            context.SaveChanges();
            Console.WriteLine("Keyboard removed.\n");
        }

            Console.WriteLine("Updated Inventory:");
        foreach (var p in context.Products)
        {
            Console.WriteLine($"{p.ProductId}: {p.Name} -
{p.Quantity} pcs @ ₹{p.Price}");
        }
    }
}
```

OUTPUT:



```
Monitor quantity updated.

Keyboard removed.

Updated Inventory:
1: Monitor - 8 pcs @ ₹9999.0
```

EXERCISE 7:

Writing Queries with LINQ

CODE:
FILE: Program.cs

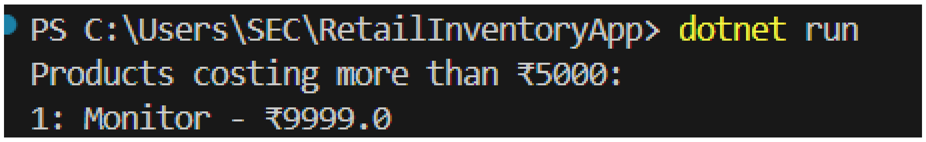
```
using RetailInventoryApp.Models;
using System;
using System.Linq;

class Program
{
    static void Main()
    {
        using var context = new RetailContext();
        context.Database.EnsureCreated();

        var expensiveProducts = context.Products
            .Where(p => p.Price > 5000)
            .ToList();

        Console.WriteLine("Products costing more than ₹5000:");
        foreach (var p in expensiveProducts)
        {
            Console.WriteLine($"{p.ProductId}: {p.Name} -
₹{p.Price}");
        }
    }
}
```

OUTPUT:



```
PS C:\Users\SEC\RetailInventoryApp> dotnet run
Products costing more than ₹5000:
1: Monitor - ₹9999.0
```