

Name : Aniket khandelwal

Mail id : 2206156@kiit.ac.in

Superset id : 6365314

University Roll : 2206156

## Week - 03

# EF Core 8.0 Guided Hands-On Exercises

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

### Scenario:

You're building an inventory management system for a retail store. The store wants to

track products, categories, and stock levels in a SQL Server database.

### Ans) Code:-

1. Create a .NET Console App:

```
dotnet new console -n RetailInventory
```

```
cd RetailInventory
```

2. Install EF Core Packages:

```
dotnet add package Microsoft.EntityFrameworkCore.SqlServer
```

```
dotnet add package Microsoft.EntityFrameworkCore.Design
```

### Output:-

```
PS C:\Users\forso\EF Core 8.0> dotnet new console -n RetailInventory
The template "Console App" was created successfully.

Processing post-creation actions...
Restoring C:\Users\forso\EF Core 8.0\RetailInventory\RetailInventory.csproj:
Restore succeeded.

PS C:\Users\forso\EF Core 8.0> cd RetailInventory
PS C:\Users\forso\EF Core 8.0\RetailInventory> dotnet add package Microsoft.EntityFrameworkCore.SqlServer

Build succeeded in 0.5s
info : X.509 certificate chain validation will use the default trust store selected by .NET for code signing.
info : X.509 certificate chain validation will use the default trust store selected by .NET for timestamping.
info : Adding PackageReference for package 'Microsoft.EntityFrameworkCore.SqlServer' into project 'C:\Users\forso\EF Core 8.0\RetailInventory\RetailInventory.csproj'.
info : GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.sqlserver/index.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.sqlserver/index.json 381ms
info : GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.sqlserver/page/0.0.1-alpha/3.1.2.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.sqlserver/page/0.0.1-alpha/3.1.2.json 297ms
info : GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.sqlserver/page/3.1.3/6.0.0-preview.6.21352.1.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.sqlserver/page/3.1.3/6.0.0-preview.6.21352.1.json 369ms
info : GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.sqlserver/page/6.0.0-preview.7.21378.4/7.0.17.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.sqlserver/page/6.0.0-preview.7.21378.4/7.0.17.json 330ms
info : GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.sqlserver/page/7.0.18/10.0.0-preview.5.25277.114.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.sqlserver/page/7.0.18/10.0.0-preview.5.25277.114.json 319ms
info : Restoring packages for C:\Users\forso\EF Core 8.0\RetailInventory\RetailInventory.csproj...
info : GET https://api.nuget.org/v3/vulnerabilities/index.json
info : OK https://api.nuget.org/v3/vulnerabilities/index.json 668ms
info : GET https://api.nuget.org/v3/vulnerabilities/2025.06.28.11.49.20/vulnerability.base.json
info : GET https://api.nuget.org/v3/vulnerabilities/2025.06.28.11.49.20/2025.07.05.05.49.44/vulnerability.update.json
info : OK https://api.nuget.org/v3/vulnerabilities/2025.06.28.11.49.20/2025.07.05.05.49.44/vulnerability.update.json 315ms
info : OK https://api.nuget.org/v3/vulnerabilities/2025.06.28.11.49.20/2025.07.05.05.49.44/vulnerability.update.json 834ms
info : Package 'Microsoft.EntityFrameworkCore.SqlServer' is compatible with all the specified frameworks in project 'C:\Users\forso\EF Core 8.0\RetailInventory\RetailInventory.csproj'.
info : PackageReference for package 'Microsoft.EntityFrameworkCore.SqlServer' version '9.0.6' added to file 'C:\Users\forso\EF Core 8.0\RetailInventory\RetailInventory.csproj'.
info : Generating MSBuild file C:\Users\forso\EF Core 8.0\RetailInventory\obj\RetailInventory.csproj.nuget.g.props.
info : Generating MSBuild file C:\Users\forso\EF Core 8.0\RetailInventory\obj\RetailInventory.csproj.nuget.g.targets.
info : Writing assets file to disk. Path: C:\Users\forso\EF Core 8.0\RetailInventory\obj\project.assets.json
log  : Restored C:\Users\forso\EF Core 8.0\RetailInventory\RetailInventory.csproj (in 2.31 sec).
```

```

PS C:\Users\forso\EF Core 8.0\RetailInventory> dotnet add package Microsoft.EntityFrameworkCore.Design
Build succeeded in 0.5s
info : X.509 certificate chain validation will use the default trust store selected by .NET for code signing.
info : X.509 certificate chain validation will use the default trust store selected by .NET for timestamping.
info : Adding PackageReference for package 'Microsoft.EntityFrameworkCore.Design' into project 'C:\Users\forso\EF Core 8.0\RetailInventory\RetailInventory.csproj'.
info : GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.design/index.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.design/index.json 741ms
info : GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.design/page/0.0.1-alpha/3.1.3.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.design/page/0.0.1-alpha/3.1.3.json 310ms
info : GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.design/page/3.1.4/6.0.0-preview.7.21378.4.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.design/page/3.1.4/6.0.0-preview.7.21378.4.json 306ms
info : GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.design/page/6.0.0-rc.1.21452.10/7.0.18.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.design/page/6.0.0-rc.1.21452.10/7.0.18.json 281ms
info : GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.design/page/7.0.19/10.0.0-preview.5.25277.114.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.design/page/7.0.19/10.0.0-preview.5.25277.114.json 292ms
info : Restoring packages for C:\Users\forso\EF Core 8.0\RetailInventory\RetailInventory.csproj...
info : CACHE https://api.nuget.org/v3/vulnerabilities/index.json
info : CACHE https://api.nuget.org/v3/vulnerabilities/2025.06.28.11.49.20/vulnerability.base.json
info : CACHE https://api.nuget.org/v3/vulnerabilities/2025.06.28.11.49.20/2025.07.05.05.49.44/vulnerability.update.json
info : Package 'Microsoft.EntityFrameworkCore.Design' is compatible with all the specified frameworks in project 'C:\Users\forso\EF Core 8.0\RetailInventory\RetailInventory.csproj'.
info : PackageReference for package 'Microsoft.EntityFrameworkCore.Design' version '9.0.0' added to file 'C:\Users\forso\EF Core 8.0\RetailInventory\RetailInventory.csproj'.
info : Generating MSBuild file C:\Users\forso\EF Core 8.0\RetailInventory\obj\RetailInventory.csproj.nuget.g.props.
info : Generating MSBuild file C:\Users\forso\EF Core 8.0\RetailInventory\obj\RetailInventory.csproj.nuget.g.targets.
info : Writing assets file to disk. Path: C:\Users\forso\EF Core 8.0\RetailInventory\obj\project.assets.json
log : Restored C:\Users\forso\EF Core 8.0\RetailInventory\RetailInventory.csproj (in 440 ms).
PS C:\Users\forso\EF Core 8.0\RetailInventory>

```

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

### Scenario:

The retail store wants to store product and category data in SQL Server.

Ans) Code:-

#### Models/Product.cs

```

using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace RetailInventory.Models
{
    public class Product
    {
        public int Id { get; set; }

        [Required]
        [MaxLength(200)]
        public string Name { get; set; } = string.Empty;

        [Column(TypeName = "decimal(18,2)")]
        public decimal Price { get; set; }

        public int CategoryId { get; set; }
        public Category Category { get; set; } = null!;
    }
}

```

#### Models/Category.cs

```

using System.ComponentModel.DataAnnotations;

namespace RetailInventory.Models
{
    public class Category
    {
        public int Id { get; set; }

        [Required]
        [MaxLength(100)]
        public string Name { get; set; } = string.Empty;

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

```

```
}
```

### 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=.;Database=RetailInventoryDB;Trusted_C
onnection=true;TrustServerCertificate=true;");
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            // Configure relationships
            modelBuilder.Entity<Product>()
                .HasOne(p => p.Category)
                .WithMany(c => c.Products)
                .HasForeignKey(p => p.CategoryId);
        }
    }
}
```

### Output:-

```
PS C:\Users\forso\OneDrive\Desktop\Digital-Nurture-Solutions\Week3 - Entity Framework Core 8.0\Lab2> dotnet build
Restore complete (0.5s)
  Lab2 succeeded (0.8s) -> bin\Debug\net9.0\Lab2.dll
Build succeeded in 2.1s
```

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

### Scenario:

The retail store's database needs to be created based on the models you've defined.

You'll use EF Core CLI to generate and apply migrations.

### Ans) Code:-

#### Program.cs

```
using Lab2.Data;
using Lab2.Models;
```

```

using Microsoft.EntityFrameworkCore;

Console.WriteLine("=== Retail Inventory System ===\n");

try
{
    using var context = new AppDbContext();
    await context.Database.EnsureCreatedAsync();
    if (!await context.Categories.AnyAsync())
    {
        Console.WriteLine("Inserting initial data...\n");

        var electronics = new Category { Name = "Electronics" };
        var groceries = new Category { Name = "Groceries" };
        var clothing = new Category { Name = "Clothing" };

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

        var products = new List<Product>
        {
            new Product { Name = "Laptop", Price = 75000, Category =
electronics },
            new Product { Name = "Smartphone", Price = 45000, Category =
electronics },
            new Product { Name = "Rice Bag (25kg)", Price = 1200, Category
= groceries },
            new Product { Name = "Cooking Oil (1L)", Price = 180, Category
= groceries },
            new Product { Name = "T-Shirt", Price = 899, Category =
clothing },
            new Product { Name = "Jeans", Price = 2499, Category =
clothing }
        };

        await context.Products.AddRangeAsync(products);
        int recordsAffected = await context.SaveChangesAsync();

        Console.WriteLine($"✅ Successfully inserted {recordsAffected}
records!\n");
    }
    else
    {
        Console.WriteLine("⚠️ Data already exists. Skipping
insertion.\n");
    }
}
catch (Exception ex)
{
    Console.WriteLine($"❌ Error: {ex.Message}");
}

```

**Output:-**

```

PS C:\Users\forso\OneDrive\Desktop\Digital-Nurture-Solutions\Week3 - Entity Framework Core 8.0\Lab2> dotnet build
Restore complete (0.3s)
Lab2 succeeded (0.2s) → bin\Debug\net9.0\Lab2.dll

Build succeeded in 1.0s
PS C:\Users\forso\OneDrive\Desktop\Digital-Nurture-Solutions\Week3 - Entity Framework Core 8.0\Lab2> dotnet ef migrations add InitialCreate
Build started...
Build succeeded.
The name 'InitialCreate' is used by an existing migration.
PS C:\Users\forso\OneDrive\Desktop\Digital-Nurture-Solutions\Week3 - Entity Framework Core 8.0\Lab2> dotnet ef database update
Build started...
Build succeeded.
Acquiring an exclusive lock for migration application. See https://aka.ms/efcore-docs-migrations-lock for more information if this takes too long.
No migrations were applied. The database is already up to date.
Done.
PS C:\Users\forso\OneDrive\Desktop\Digital-Nurture-Solutions\Week3 - Entity Framework Core 8.0\Lab2> dotnet run
>>
DbContext initialized successfully!

```

MACBUKPRO\SQLEXPRESS01 (SQL Server 16.0.1000 - MACBUKPRO\forso)

- [-] Databases
  - [+] System Databases
  - [+] Database Snapshots
  - [+] EmployeeManagement
  - [+] EmployeeManagementSystem
  - [+] new
  - [+] ProductDB
  - [+] Products
  - [+] ProductsDB
  - [-] RetailInventoryDB
    - [+] Database Diagrams
    - [-] Tables
      - [+] System Tables
      - [+] FileTables
      - [+] External Tables
      - [+] Graph Tables
      - [+] dbo.\_EFMigrationsHistory
      - [+] **dbo.Categories**
      - [+] dbo.Products

SQLQuery4.sql...forso (73) SQLQuery3.sql...O\forso (70) SQLQuery2.sql...O\forso (66)

```

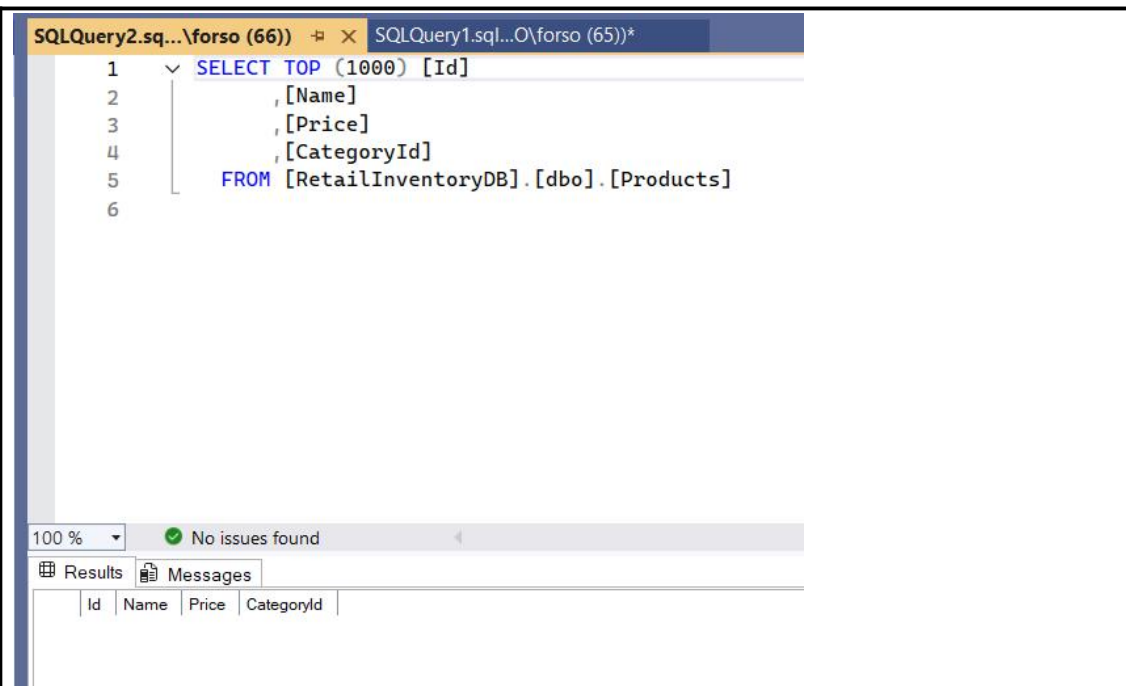
1  SELECT TOP (1000) [Id]
2    , [Name]
3    FROM [RetailInventoryDB].[dbo].[Categories]
4

```

100 % No issues found

Results Messages

Id	Name
----	------



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

##### Scenario:

The store manager wants to add initial product categories and products to the system.

Ans) Code:-

##### Program.cs

```
using Lab2.Data;
using Lab2.Models;
using Microsoft.EntityFrameworkCore;

Console.WriteLine("=== Retail Inventory System ===\n");

try
{
    using var context = new AppDbContext();

    // Ensure database is created
    await context.Database.EnsureCreatedAsync();

    // Check if data already exists
    if (!await context.Categories.AnyAsync())
    {
        Console.WriteLine("Inserting initial data...\n");

        // Create categories
        var electronics = new Category { Name = "Electronics" };
        var groceries = new Category { Name = "Groceries" };

        var clothing = new Category { Name = "Clothing" };
    }
}
```



```

        await context.Categories.AddRangeAsync(electronics, groceries,
clothing);
        // Create products
        var products = new List<Product>
        {
            new Product { Name = "Laptop", Price = 75000, Category =
electronics },
            new Product { Name = "Rice Bag (25kg)", Price = 1200, Category
= groceries },
            new Product { Name = "Jeans", Price = 2499, Category =
clothing },
        };

        await context.Products.AddRangeAsync(products);

        // Save changes
        int recordsAffected = await context.SaveChangesAsync();
        Console.WriteLine($"Successfully inserted {recordsAffected}
records!\n");
    }
    else
    {
        Console.WriteLine("Data already exists. Skipping insertion.\n");
    }
}
catch (Exception ex)
{
    Console.WriteLine($"Error: {ex.Message}");
}
}

```

### RetailInventoryDB:

```

USE RetailInventoryDB;
GO
SELECT * FROM Categories;
SELECT * FROM Products;

```

### Output:-

```

PS C:\Users\forso\OneDrive\Desktop\Digital-Nurture-Solutions\Week3 - Entity Framework Core 8.0\Lab2> dotnet run
=== Retail Inventory System ===

Inserting initial data...
✅ Successfully inserted 5 records!

PS C:\Users\forso\OneDrive\Desktop\Digital-Nurture-Solutions\Week3 - Entity Framework Core 8.0\Lab2> dotnet run
=== Retail Inventory System ===

Data already exists. Skipping insertion.

PS C:\Users\forso\OneDrive\Desktop\Digital-Nurture-Solutions\Week3 - Entity Framework Core 8.0\Lab2> dotnet run
=== Retail Inventory System ===

Data already exists. Skipping insertion.

```

```

1  USE RetailInventoryDB;
2  GO
3  SELECT * FROM Categories;
4  SELECT * FROM Products;
5

```

100 % No issues found

Results Messages

2	2	Groceries
3	3	Clothing

	Id	Name	Price	CategoryId
1	1	Laptop	75000.00	1
2	2	Rice Bag (25kg)	1200.00	2

## Lab 5: Retrieving Data from the Database

### Scenario:

The store wants to display product details on the dashboard.

Ans) Code:-

### Lab5/Program.cs

```

using Lab2.Data;
using Lab2.Models;
using Microsoft.EntityFrameworkCore;

Console.WriteLine("=== Data Retrieval Examples ===\n");

try
{
    using var context = new AppDbContext();

    // 1. Retrieve All Products
    Console.WriteLine("📄 ALL PRODUCTS:");
    Console.WriteLine("".PadRight(50, '-'));

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

    foreach (var product in products)
    {
        Console.WriteLine($"{product.Name} - ₹{product.Price:N0} ({product.Category.Name}");
    }

    // 2. Find by ID
    Console.WriteLine($"🔍 FIND BY ID (ID: 1):");
    Console.WriteLine("".PadRight(50, '-'));
}

```



```

var productById = await context.Products
    .Include(p => p.Category)
    .FirstOrDefaultAsync(p => p.Id == 1);

if (productById != null)
{
    Console.WriteLine($"Found: {productById.Name} -
{productById.Price:N0}");
}
else
{
    Console.WriteLine("Product not found!");
}

// 3. FirstOrDefault with Condition
Console.WriteLine($"\\n EXPENSIVE PRODUCTS (Price > ₹50,000):");
Console.WriteLine("".PadRight(50, '-'));

var expensiveProducts = await context.Products
    .Include(p => p.Category)
    .Where(p => p.Price > 50000)
    .ToListAsync();

if (expensiveProducts.Any())
{
    foreach (var product in expensiveProducts)
    {
        Console.WriteLine($" {product.Name} - ₹{product.Price:N0}");
    }
}
else
{
    Console.WriteLine("No expensive products found!");
}

// 4. Count and Statistics
Console.WriteLine($"\\n STATISTICS:");
Console.WriteLine("".PadRight(50, '-'));

var totalProducts = await context.Products.CountAsync();
var totalCategories = await context.Categories.CountAsync();
var avgPrice = await context.Products.AverageAsync(p => p.Price);

Console.WriteLine($"Total Products: {totalProducts}");
Console.WriteLine($"Total Categories: {totalCategories}");
Console.WriteLine($"Average Price: ₹{avgPrice:N2}");
}
catch (Exception ex)
{
    Console.WriteLine($"Error: {ex.Message}");
}
}

```

**Output:-**

SQLQuery11.s... \forso (85)) SQLQuery10.sql...O\forso (64))

```

1  SELECT TOP (1000) [Id]
2      , [Name]
3  FROM [RetailInventoryDB].[dbo].[Categories]
4

```

100 % No issues found

Results Messages

	Id	Name
1	1	Electronics
2	2	Groceries
3	3	Clothing

SQLQuery10.s... \forso (64)) SQLQuery9.sql...O\forso (79))

```

1  SELECT TOP (1000) [Id]
2      , [Name]
3      , [Price]
4      , [CategoryId]
5  FROM [RetailInventoryDB].[dbo].[Products]
6

```

100 % No issues found

Results Messages

	Id	Name	Price	CategoryId
1	1	Laptop	75000.00	1
2	2	Rice Bag (25kg)	1200.00	2

```

PS C:\Users\forso\OneDrive\Desktop\Digital-Nurture-Solutions\Week3 - Entity Framework Core 8.0\Lab5> dotnet run
=== Data Retrieval Examples ===

ALL PRODUCTS:
-----
Laptop - ₹75,000 (Electronics)
Rice Bag (25kg) - ₹1,200 (Groceries)

FIND BY ID (ID: 1):
-----
Found: Laptop - 75,000

EXPENSIVE PRODUCTS (Price > ₹50,000):
-----
Laptop - ₹75,000

STATISTICS:
-----
Total Products: 2
Total Categories: 3
Average Price: ₹38,100.00

```