# Week 3 Assignment Solutions (Entity Framework Core 8.0) EF Core 8.0 HOL

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

### 1. What is ORM (Object-Relational Mapping)?

ORM is a programming technique that lets developers interact with databases using object-oriented code instead of raw SQL. In .NET, libraries like Entity Framework (EF) Core serve as the ORM layer.

How ORM Works:

Maps C# classes to SQL database tables.

Maps class properties to columns.

Automatically translates LINQ queries into SQL under the hood.

### 2. Benefits of ORM:

| Benefit | Description |
|---|---|
| Productivity | No need to write complex SQL queries for CRUD operations. |
| Maintainability | Easy to refactor C# models; changes propagate to DB via migrations. |
| Abstraction | Focus on business logic; ORM handles the SQL, connection, transactions. |
| Portability | EF Core supports multiple databases (SQL Server, PostgreSQL, SQLite). |

### 3. EF Core vs EF Framework (EF6)

| Feature | EF Core 8.0 | EF Framework (EF6) |
|---|---|---|
| Cross-platform | Yes (Windows, Linux, macOS) | No (Windows only) |
| Performance | Lightweight, high-perf | Heavier runtime |
| Async LINQ support | Fully async/await support | Partial / complex setup |
| Modern APIs | LINQ, compiled queries, etc. | Legacy architecture |
| Migrations | Code-first migrations | Code-first and DB-first |
| Flexibility | More extensible, modular | Monolithic, limited hooks |

Use EF Core for new, modern apps.
Use EF6 only when maintaining legacy systems.

### 4. Create a .NET Console App:

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

### 5. Install EF Core Packages:

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

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

1. Models

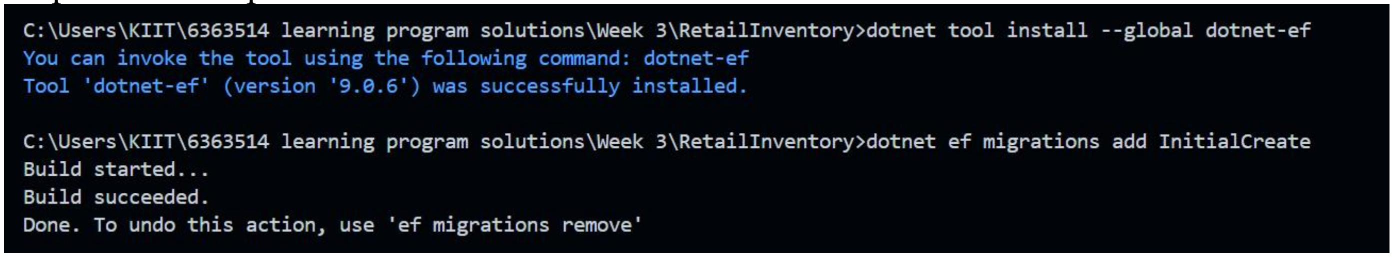
```
public class Category {
    public int Id { get; set; }
    public string Name { get; set; }
    public List<Product> Products { get; set; } = new();
}
public class Product {
    public int Id { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
    public int CategoryId { get; set; }
    public Category Category { get; set; }
}
```

2. AppDbContext.cs

```
using Microsoft.EntityFrameworkCore;

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=localhost\\SQLEXPRESS;Database=RetailInventory
Db;Trusted_Connection=True;Encrypt=False;");
    }
}
```

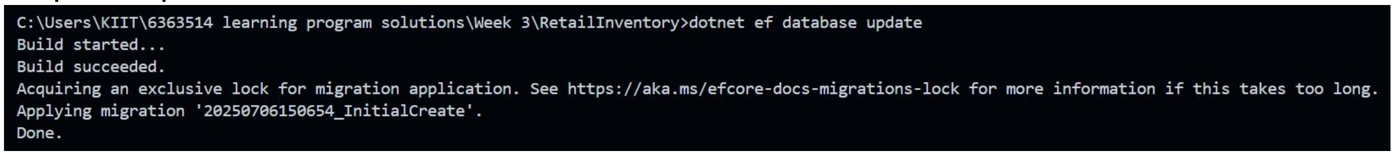
## Lab 3: Using EF Core CLI to Create and Apply Migrations

Step 1 and 2 output



```
C:\Users\KIIT\6363514 learning program solutions\Week 3\RetailInventory>dotnet tool install --global dotnet-ef
You can invoke the tool using the following command: dotnet-ef
Tool 'dotnet-ef' (version '9.0.6') was successfully installed.

C:\Users\KIIT\6363514 learning program solutions\Week 3\RetailInventory>dotnet ef migrations add InitialCreate
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'
```

Step 3 output

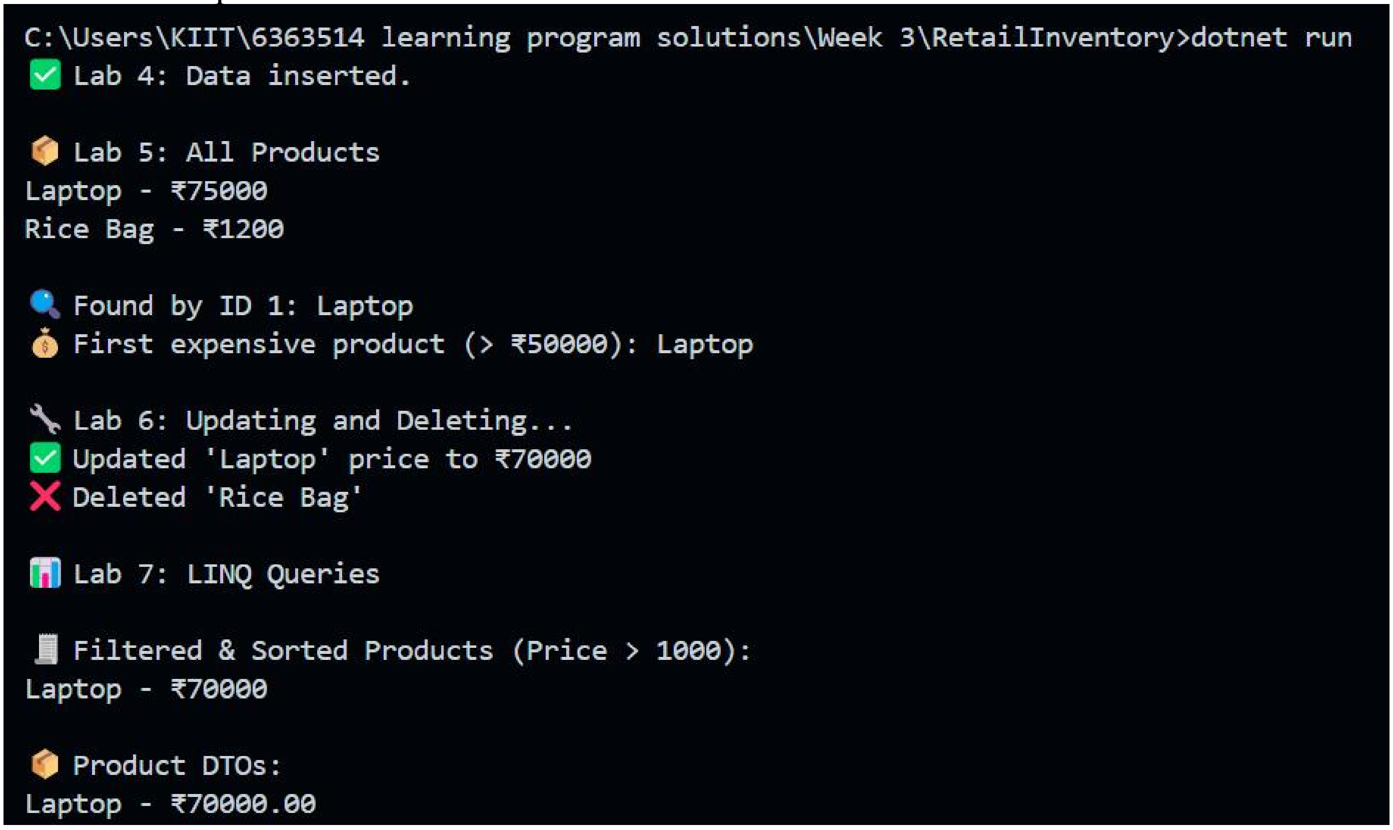


```
C:\Users\KIIT\6363514 learning program solutions\Week 3\RetailInventory>dotnet ef database update
Build started...
Build succeeded.
Acquiring an exclusive lock for migration application. See https://aka.ms/efcore-docs-migrations-lock for more information if this takes too long.
Applying migration '20250706150654_InitialCreate'.
Done.
```

Lab 4: Inserting Initial Data into the Database
Lab 5: Retrieving Data from the Database
Lab 6: Updating and Deleting Records
Lab 7: Writing Queries with LINQ

From the lab 4 to lab 7, I write the Product.cs for all fours with the output, run all of them once(first time), then comment out the data value in Database(for verification only)

Here's the output :



```
C:\Users\KIIT\6363514 learning program solutions\Week 3\RetailInventory>dotnet run
✅ Lab 4: Data inserted.

📦 Lab 5: All Products
Laptop - ₹75000
Rice Bag - ₹1200

🔍 Found by ID 1: Laptop
💰 First expensive product (> ₹50000): Laptop

🔧 Lab 6: Updating and Deleting...
✅ Updated 'Laptop' price to ₹70000
❌ Deleted 'Rice Bag'

📊 Lab 7: LINQ Queries

🧾 Filtered & Sorted Products (Price > 1000):
Laptop - ₹70000

📦 Product DTOs:
Laptop - ₹70000.00
```