**Author - Utsav Saxena , Superset ID - 6361856**

**EF Core 8.0 Hands-On Exercises Solutions**

**Lab 1**

**Understanding ORM with a Retail Inventory System**

**Question 1 -**  What is ORM**?**

**Answer-** ORM - Object relational Mapping is a technique used to convert data between incompatible type systems in OOP languages and Relational databases.ORM maps database tables to classes, which leads to simplifying CRUD operations.

**Question 2** - EF Core vs EF Framework

**Answer -**

| **Feature** | **Entity Framework Core** | **Entity Framework 6 (EF 6)** |
| --- | --- | --- |

|  |  |  |
| --- | --- | --- |
| **Platform** | Cross-platform (.NET Core & .NET 5/6/7/8+) | Windows-only (.NET Framework) |

|  |  |  |
| --- | --- | --- |
| **Performance** | Generally faster and more lightweight | Slower compared to EF Core |

|  |  |  |
| --- | --- | --- |
| **Modularity** | Lightweight and extensible, modular design | Monolithic design, less modular |

|  |  |  |
| --- | --- | --- |
| **LINQ Support** | Improved LINQ translation and performance | Mature but limited compared to EF Core |

|  |  |  |
| --- | --- | --- |
| **Database Support** | Supports more modern and cloud databases (e.g., Azure Cosmos DB) | Mainly traditional relational databases |

|  |  |  |
| --- | --- | --- |
| **Migration Support** | Built-in code-based migrations | Also supports code-based migrations |

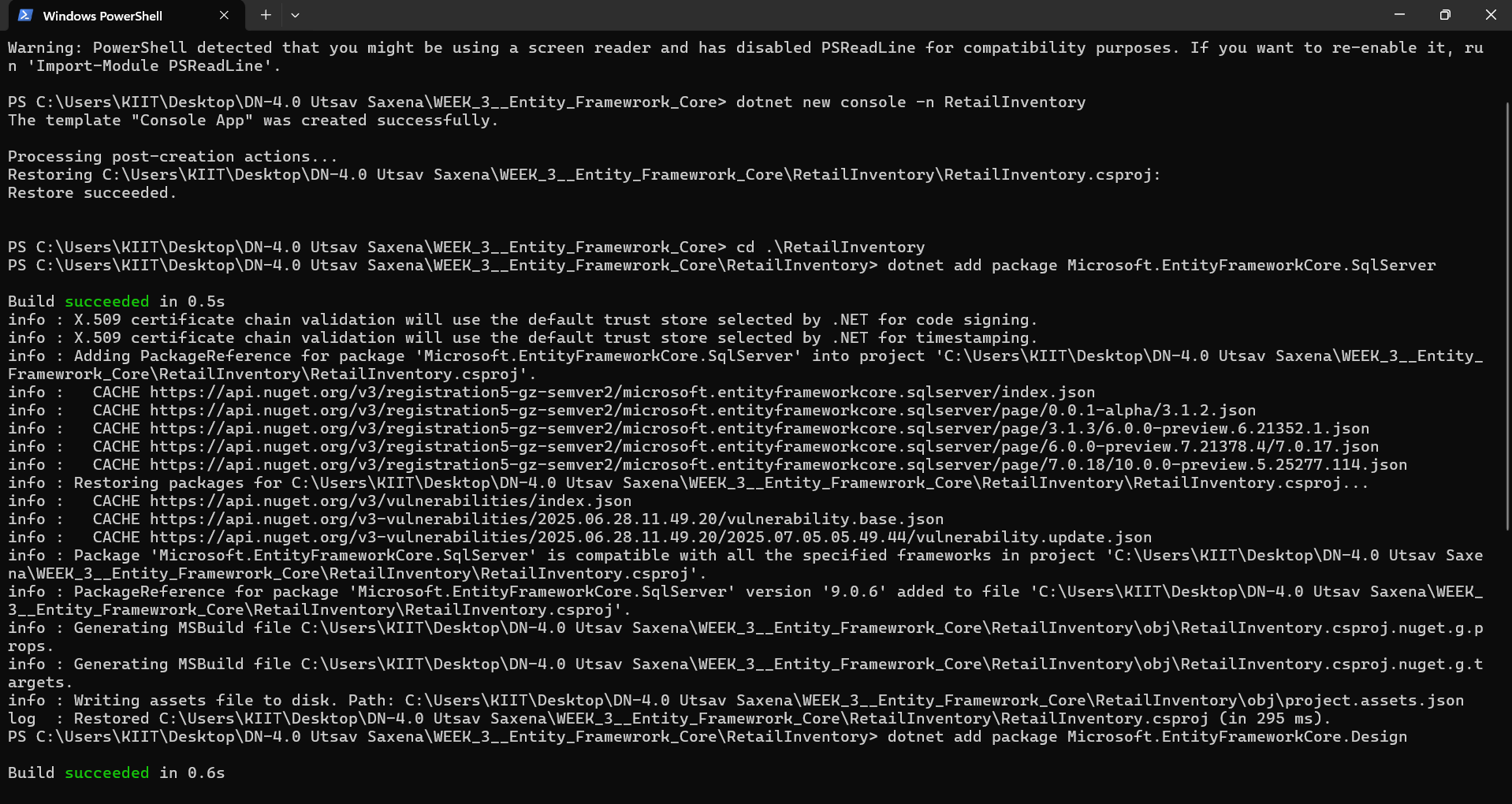
|  |  |  |
| --- | --- | --- |
| **Change Tracking** | More efficient in EF Core | Less optimized compared to EF Core |

**Question 3** - EF Core 8.0 Features.

**Answer**- Modern .NET Support,Improved LINQ translation, JSON Column Mapping, Better Bulk Operations.

**LAB 1 Code -**

**CLI Command execution**

****

**LAB 1 Output -**

**Created Folder(console app) in the Given path.**

**Lab 2**

**Setting Up the Database Context for a Retail Store**

**LAB 2 Code -**

**Models/Category.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RetailInventory.Models

{

public class Category

{

public int Id { get; set; }

public string Name { get; set; }

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

}

}

**Models/Product.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RetailInventory.Models

{

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; }

}

}

**./AppDbContext.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

using RetailInventory.Models;

using Microsoft.Extensions.Configuration;

namespace RetailInventory

{

public class AppDbContext : DbContext

{

public AppDbContext() { }

public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

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

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

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) {

optionsBuilder.UseSqlServer("Server=localhost\\SQLEXPRESS;Database=R etailDb;Trusted\_Connection=True;TrustServerCertificate=True;");

}

}

}

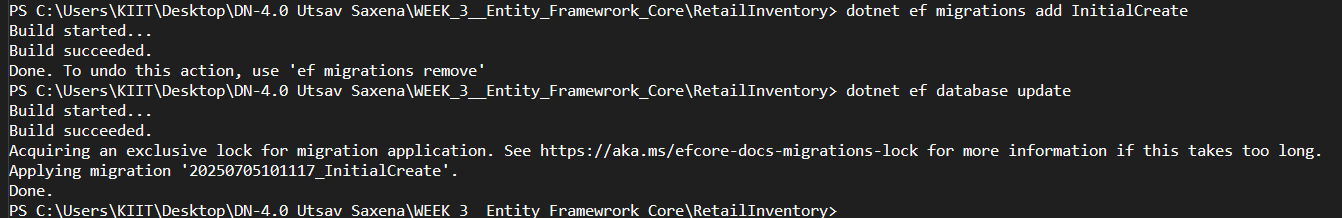
**LAB 2 Output -**

**No output produced in this code.**

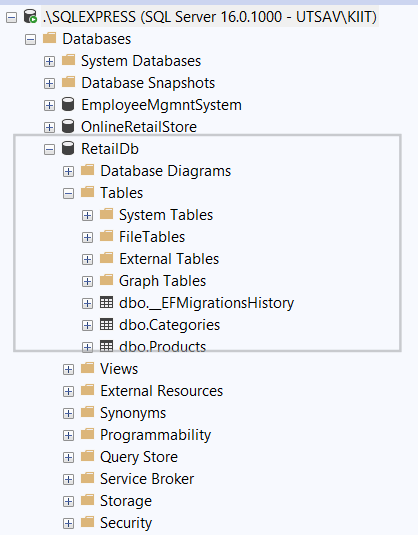
**Lab 3**

**Using EF Core CLI to Create and Apply Migrations**

**LAB 3 Code -**

****

**LAB 3 Output -**

****

**Lab 4**

**Inserting Initial Data into the Database**

**LAB 4 Code -**

using Microsoft.EntityFrameworkCore;

using RetailInventory;

using RetailInventory.Models;

using var context = new AppDbContext();

var electronics = new Category { Name = "Electronics" };

var groceries = new Category { Name = "Groceries" };

var author = new Category { Name = "Author" };

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

var product1 = new Product { Name = "Laptop", Price = 75000, Category = electronics };

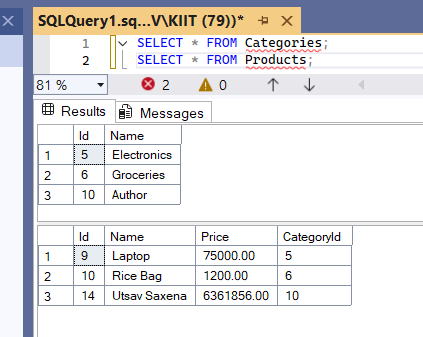
var product2 = new Product { Name = "Rice Bag", Price = 1200, Category = groceries };

var product3 = new Product { Name = "Utsav Saxena", Price = 6361856, Category = author };

await context.Products.AddRangeAsync(product1, product2, product3);

await context.SaveChangesAsync();

**LAB 4 Output-**

****

**Lab 5**

**Retrieving Data from the Database**

**LAB 5 Code -**

Console.WriteLine("Name - Utsav Saxena, Superset ID - 6361856 \n");

var products = await context.Products.ToListAsync();

foreach (var p in products)

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

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

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

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

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

**LAB 5 Output-**

