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

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

1. **What is ORM?**
   * ORM stands for Object-Relational Mapping
   * A programming technique that maps objects in an object-oriented programming language (like C#) to data in a relational database (like SQL Server).
   * C# category class are mapped to SQL category table
   * C# product class are mapped to SQL product table
   * **Benefits:**
     + **productivity:** reduce the need of writing repetitive SQL. Speeds up development
     + **Maintainability:** clean code and easy to understand .
     + **Abstraction:** Everyone can easily perform basic CRUD operation
2. **EF Core vs EF Framework:**
   * **EF Core:**
     + cross platform :Runs on Windows, macOS, and Linux.   
       Lightweight: A more modular and performance-oriented rewrite.
     + Modern features: Supports LINQ (Language Integrated Query), async queries (for non-blocking I/O), compiled queries (for better performance), and more.
   * **EF Framework (EF6):**
     + Windows-only: Primarily designed for .NET Framework applications on Windows.
     + More mature: Has a longer history and may have some niche features not yet in EF Core.
     + Less flexible: Can be heavier and less performant in some scenarios compared to EF Core.
3. **EF Core 8.0 Features :**
   * JSON column mapping: Allows you to map complex C# objects directly to JSON columns in your database, making it easier to store semi-structured data.
   * Improved performance with compiled models: Generates highly optimized code for database access, leading to faster query execution.
   * Interceptors and better bulk operations: Provides more hooks to customize EF Core's behavior and offers improved support for performing large-scale data operations efficiently.
4. **Create a .NET Console App:**
   * create a project in vs code and choose console app
   * click next and name your project as retailInventory
5. **Install EF Core Packages:**
   * in solution click your project and select manage Nuget packages
   * install all essential packages or goto terminal and add this command  
     + dotnet add package Microsoft.EntityFrameworkCore.SqlServer
     + dotnet add package Microsoft.EntityFrameworkCore.Design.

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

* **Model:**
  + **File: 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; }  }  } |
| --- |

* + **file:product.cs**

| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using Microsoft.EntityFrameworkCore;  using RetailInventory.Models;  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; }  }  } |
| --- |

* **Create AppDbContext:** 
  + **File: Data -> AppDbContext**

| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using Microsoft.EntityFrameworkCore;  using RetailInventory.Models;  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; }  }  } |
| --- |

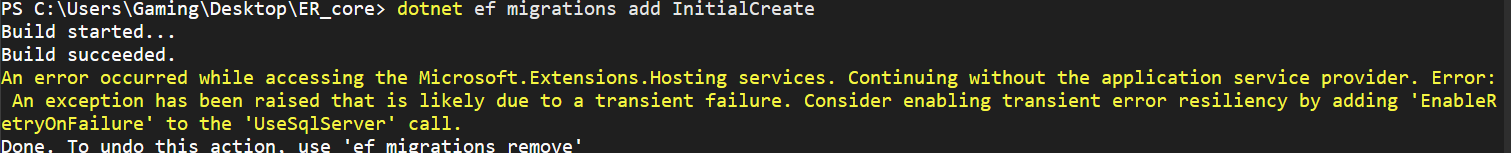
**Lab 3: Using EF Core CLI to Create and Apply Migrations**

1. **install EF CLI Tool:**

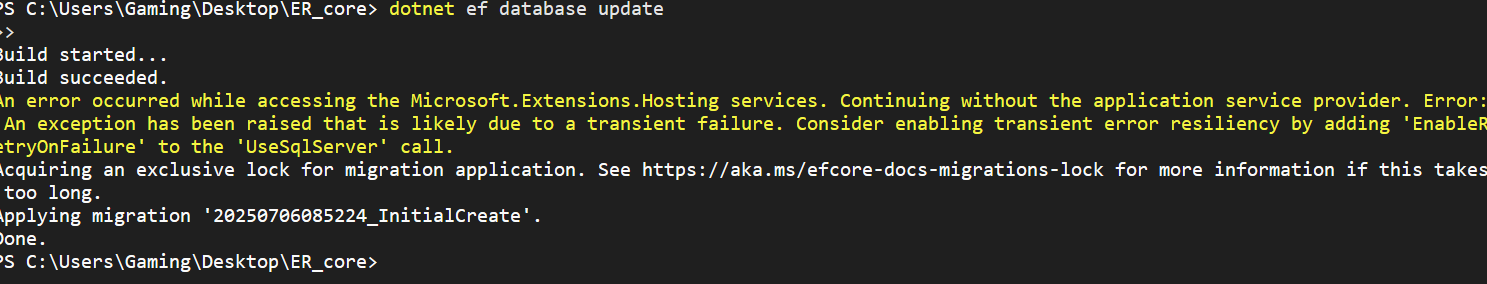
dotnet tool install --global dotnet-ef

1. **Create migration:**

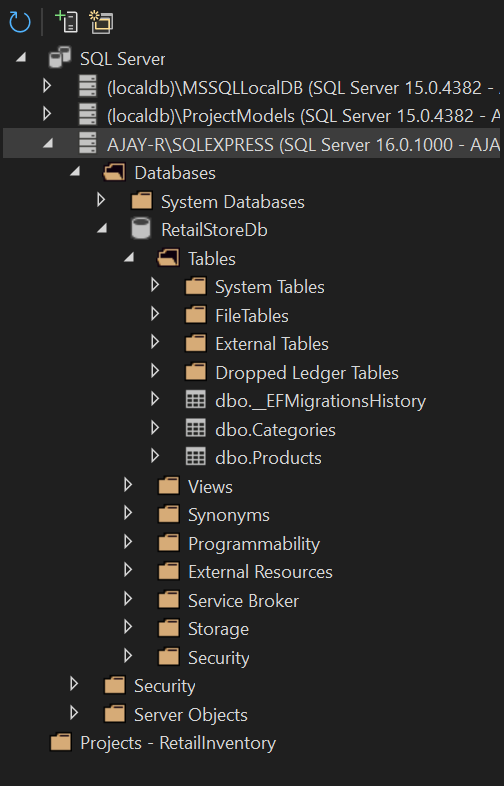
dotnet ef migrations add InitialCreate



1. **Apply migration:**

**dotnet ef database update**

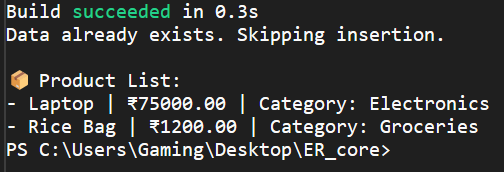
1. **Verify in SQL:**

****

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

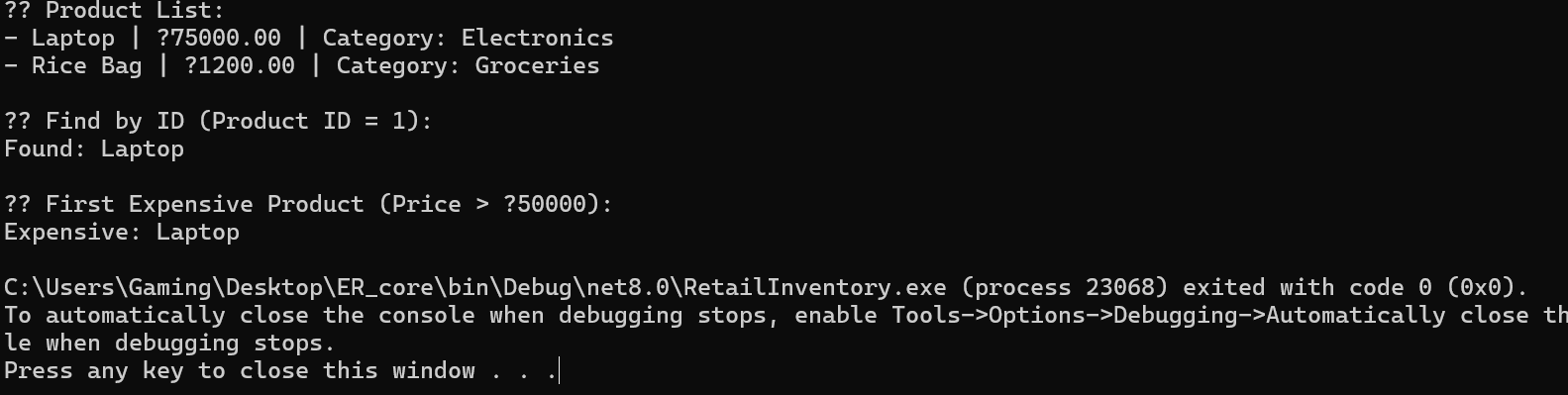
1. **Insert Data in Program.cs:**

| using Microsoft.EntityFrameworkCore;  using RetailInventory.Data;  using RetailInventory.Models;  using System;  using System.Threading.Tasks;  class Program  {  static async Task Main()  {  using var context = new AppDbContext();  Console.WriteLine(" Product List:");  var products = await context.Products.Include(p => p.Category).ToListAsync();  foreach (var p in products)  {  Console.WriteLine($"- {p.Name} | ₹{p.Price:0.00} | Category: {p.Category?.Name}");  }  Console.WriteLine("\n Find by ID (Product ID = 1):");  var product = await context.Products.FindAsync(1);  Console.WriteLine(product != null ? $"Found: {product.Name}" : "Product not found");  Console.WriteLine("\n First Expensive Product (Price > ₹50000):");  var expensive = await context.Products.FirstOrDefaultAsync(p => p.Price > 50000);  Console.WriteLine(expensive != null ? $"Expensive: {expensive.Name}" : "No expensive product found");  }  } |
| --- |



**Lab 5: Retrieving Data from the Database**

* **Retrieve All Products:**
* **Find by ID**
* **FirstOrDefault with Condition:**

****

****