**ENTITY FRAMEWORK CORE 8.0**

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

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.

**Commands:**

dotnet new console -n RetailInventory

cd RetailInventory

dotnet add package Microsoft.EntityFrameworkCore.SqlServer

dotnet add package Microsoft.EntityFrameworkCore.Design

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

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

**Code:**

**Category.cs:**

namespace RetailInventory.Models;

public class Category

{

    public int Id { get; set; }

    public string? Name { get; set; }

    public List&lt;Product&gt; Products { get; set; } = new();

}

**Product.cs:**

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 Microsoft.EntityFrameworkCore;

using RetailInventory.Models;

public class AppDbContext : DbContext

{

    public DbSet&lt;Product&gt; Products { get; set; }

    public DbSet&lt;Category&gt; Categories { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

    {

optionsBuilder.UseSqlServer(&quot;Server=localhost;Database=RetailInventoryDB;Trusted\_Conn

ection=True;TrustServerCertificate=True&quot;);

    }

}

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

The retail stores database needs to be created based on the models you’ve defined. You’ll use EF Core CLI to generate and apply migrations

Migration:

20250706163627\_InitialCreate.cs :

using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace RetailInv.Migrations

{

    /// &lt;inheritdoc /&gt;

    public partial class InitialCreate : Migration

    {

        /// &lt;inheritdoc /&gt;

        protected override void Up(MigrationBuilder migrationBuilder)

        {

            migrationBuilder.CreateTable(

                name: &quot;Categories&quot;,

                columns: table =&gt; new

                {

                    Id = table.Column&lt;int&gt;(type: &quot;int&quot;, nullable: false)

                        .Annotation(&quot;SqlServer:Identity&quot;, &quot;1, 1&quot;),

                    Name = table.Column&lt;string&gt;(type: &quot;nvarchar(max)&quot;, nullable: true)

                },

                constraints: table =&gt;

                {

                    table.PrimaryKey(&quot;PK\_Categories&quot;, x =&gt; x.Id);

                });

            migrationBuilder.CreateTable(

                name: &quot;Products&quot;,

                columns: table =&gt; new

                {

                    Id = table.Column&lt;int&gt;(type: &quot;int&quot;, nullable: false)

                        .Annotation(&quot;SqlServer:Identity&quot;, &quot;1, 1&quot;),

                    Name = table.Column&lt;string&gt;(type: &quot;nvarchar(max)&quot;, nullable: true),

                    Price = table.Column&lt;decimal&gt;(type: &quot;decimal(18,2)&quot;, nullable: false),

                    CategoryId = table.Column&lt;int&gt;(type: &quot;int&quot;, nullable: false)

                },

                constraints: table =&gt;

                {

                    table.PrimaryKey(&quot;PK\_Products&quot;, x =&gt; x.Id);

                    table.ForeignKey(

                        name: &quot;FK\_Products\_Categories\_CategoryId&quot;,

                        column: x =&gt; x.CategoryId,

                        principalTable: &quot;Categories&quot;,

                        principalColumn: &quot;Id&quot;,

                        onDelete: ReferentialAction.Cascade)

      });

            migrationBuilder.CreateIndex(

                name: &quot;IX\_Products\_CategoryId&quot;,

                table: &quot;Products&quot;,

                column: &quot;CategoryId&quot;);

        }

        /// &lt;inheritdoc /&gt;

        protected override void Down(MigrationBuilder migrationBuilder)

        {

            migrationBuilder.DropTable(

                name: &quot;Products&quot;);

            migrationBuilder.DropTable(

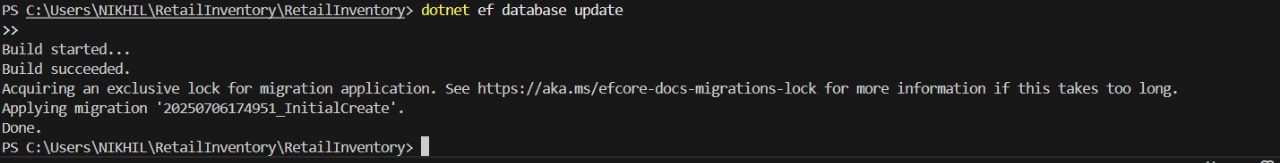
                name: &quot;Categories&quot;);

        }

    }

}

**OUTPUT :**

****

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

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

**Program.cs:**

using RetailInventory.Models;

using var context = new AppDbContext();

var electronics = new Category { Name = &quot;Electronics&quot; };

var groceries = new Category { Name = &quot;Groceries&quot; };

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

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

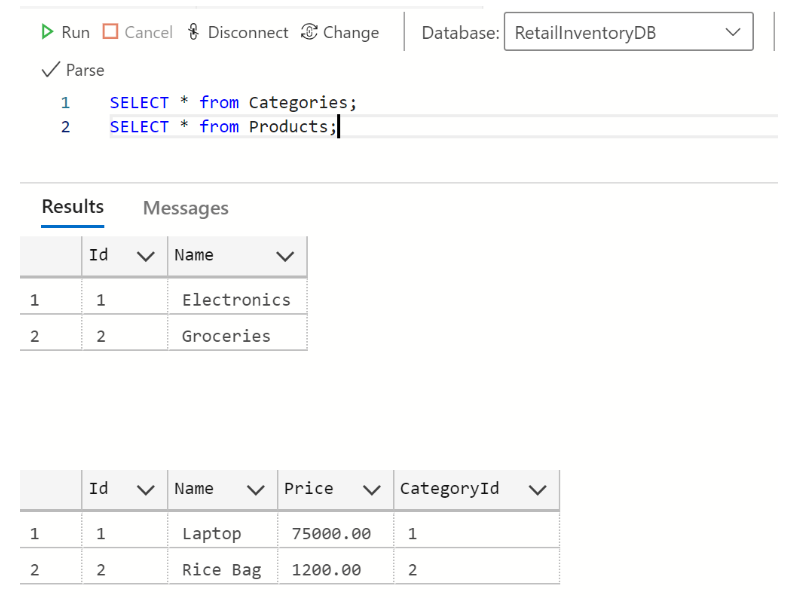
var product2 = new Product { Name = &quot;Rice Bag&quot;, Price = 1200, Category = groceries };

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

await context.SaveChangesAsync();

Console.WriteLine(“Initial data inserted successfully.”);

**OUTPUT :**

****

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

The store wants to display product details on the dashboard

**Program.cs:**

using RetailInventory.Models;

using Microsoft.EntityFrameworkCore;

using var context = new AppDbContext();

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

Console.WriteLine(&quot;\nAll Products:&quot;);

foreach (var p in allProducts)

{

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

}

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

if (foundProduct != null)

{

    Console.WriteLine($&quot;\n Found product with ID 1: {foundProduct.Name} -

₹{foundProduct.Price}&quot;);

}

var expensiveProduct = await context.Products

    .FirstOrDefaultAsync(p =&gt; p.Price &gt; 50000);

if (expensiveProduct != null)

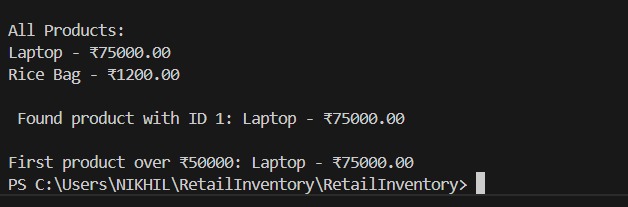
{

    Console.WriteLine($&quot;\nFirst product over ₹50000: {expensiveProduct.Name} -

₹{expensiveProduct.Price}&quot;);

}

**OUTPUT :**

****