**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<Product> 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<Product> Products { get; set; }

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

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

    {

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

    }

}

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

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

**Migration:**

**20250706163627\_InitialCreate.cs :**

using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace RetailInv.Migrations

{

*/// <inheritdoc />*

    public partial class InitialCreate : Migration

    {

*/// <inheritdoc />*

        protected override void Up(MigrationBuilder migrationBuilder)

        {

            migrationBuilder.CreateTable(

                name: "Categories",

                columns: table => new

                {

                    Id = table.Column<int>(type: "int", nullable: false)

                        .Annotation("SqlServer:Identity", "1, 1"),

                    Name = table.Column<string>(type: "nvarchar(max)", nullable: true)

                },

                constraints: table =>

                {

                    table.PrimaryKey("PK\_Categories", x => x.Id);

                });

            migrationBuilder.CreateTable(

                name: "Products",

                columns: table => new

                {

                    Id = table.Column<int>(type: "int", nullable: false)

                        .Annotation("SqlServer:Identity", "1, 1"),

                    Name = table.Column<string>(type: "nvarchar(max)", nullable: true),

                    Price = table.Column<decimal>(type: "decimal(18,2)", nullable: false),

                    CategoryId = table.Column<int>(type: "int", nullable: false)

                },

                constraints: table =>

                {

                    table.PrimaryKey("PK\_Products", x => x.Id);

                    table.ForeignKey(

                        name: "FK\_Products\_Categories\_CategoryId",

                        column: x => x.CategoryId,

                        principalTable: "Categories",

                        principalColumn: "Id",

                        onDelete: ReferentialAction.Cascade);

                });

            migrationBuilder.CreateIndex(

                name: "IX\_Products\_CategoryId",

                table: "Products",

                column: "CategoryId");

        }

*/// <inheritdoc />*

        protected override void Down(MigrationBuilder migrationBuilder)

        {

            migrationBuilder.DropTable(

                name: "Products");

            migrationBuilder.DropTable(

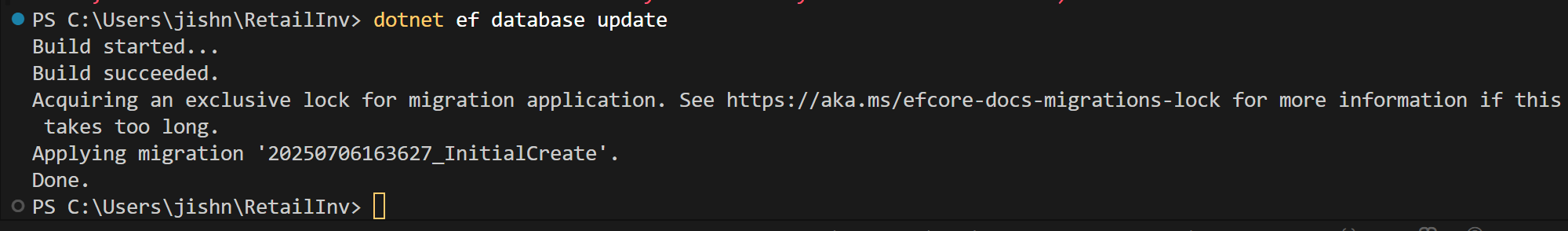
                name: "Categories");

        }

    }

}

**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 = "Electronics" };

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

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

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

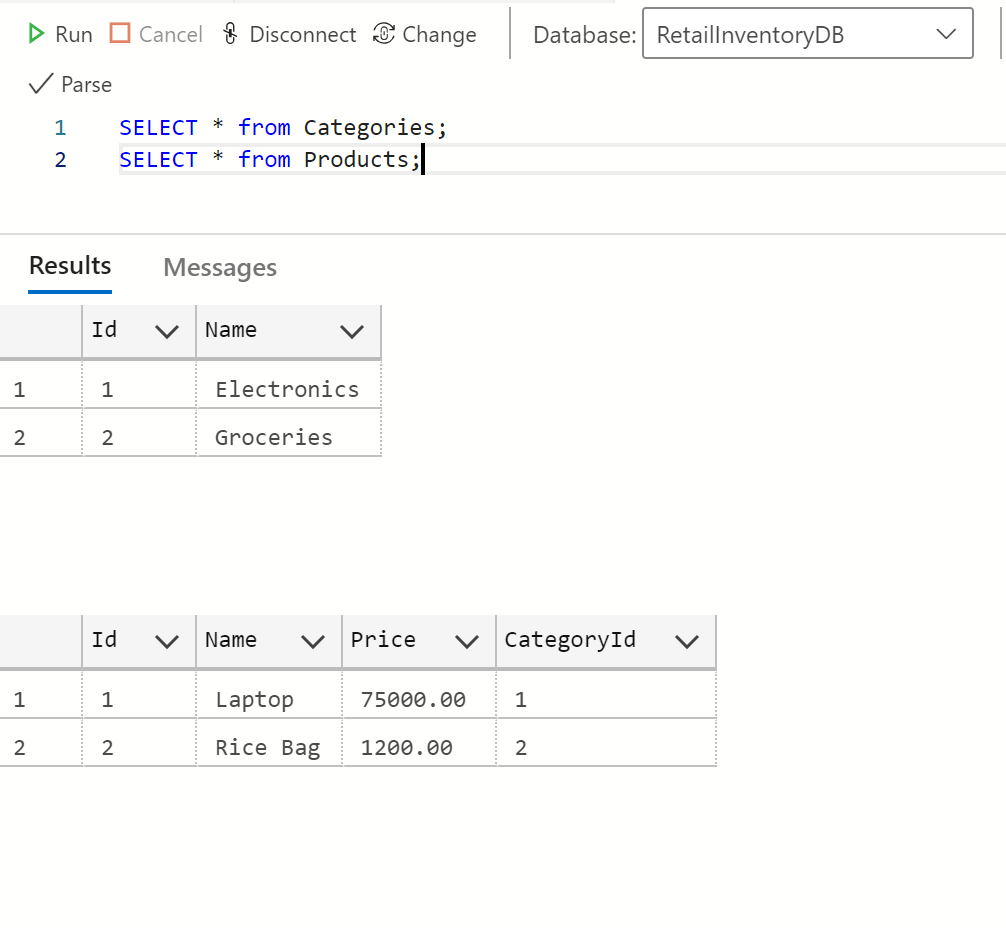
var product2 = new Product { Name = "Rice Bag", Price = 1200, Category = groceries };

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("\nAll Products:");

*foreach* (var p *in* allProducts)

{

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

}

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

*if* (foundProduct != null)

{

    Console.WriteLine($"\n Found product with ID 1: {foundProduct.Name} - ₹{foundProduct.Price}");

}

var expensiveProduct = await context.Products

    .FirstOrDefaultAsync(p => p.Price > 50000);

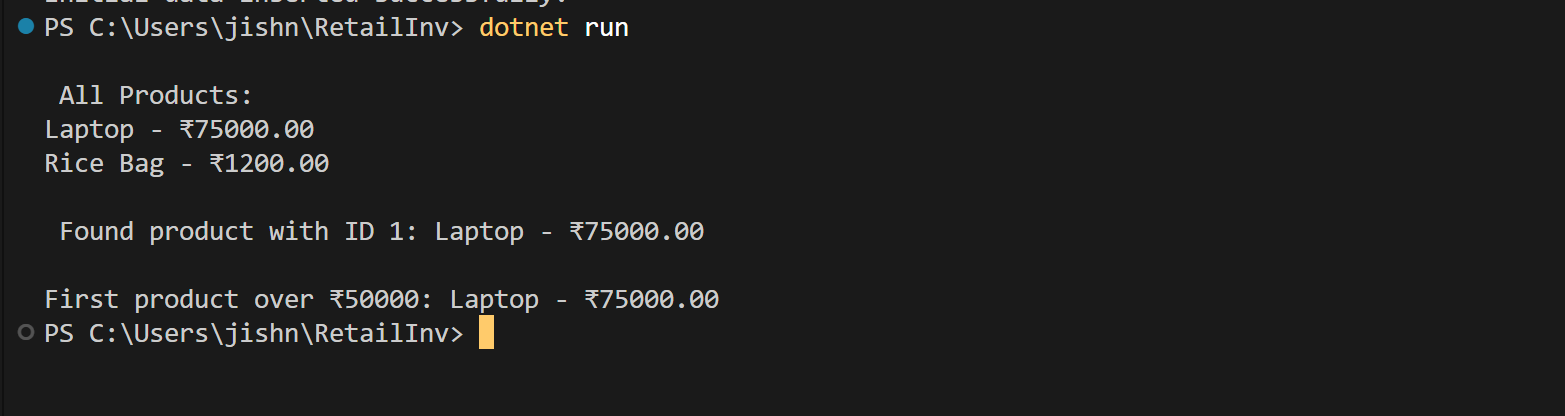
*if* (expensiveProduct != null)

{

    Console.WriteLine($"\nFirst product over ₹50000: {expensiveProduct.Name} - ₹{expensiveProduct.Price}");

}

**Output:**

****