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

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

1. What is ORM?

ORM uses metadata (defined through code, annotations, or config) to understand how to map:

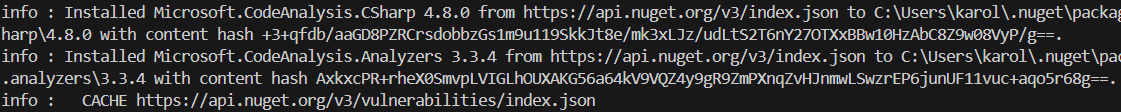
A C# class - a database table

A class property - a table column

Object relationships - foreign key constraints

The ORM framework creates a structured map that captures relationships between objects and tables, without needing to manually define SQL joins, constraints, or schema logic.

1. Creating .NET Console app & installing EF core packages.



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

1. Create Models:

**Category.cs:**

using System.Collections.Generic;

public class Category

{

public int Id { get; set; }

public string Name { get; set; }

public List<Product> Products { get; set; } = new List<Product>();

}

**Product.cs:**

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; }

}

1. Create **AppDbContext:**

using Microsoft.EntityFrameworkCore;

using System.Collections.Generic;

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=JOHANNA\\SQLEXPRESS;Database=RetailDb;Trusted\_Connection=True;TrustServerCertificate=True;");

}

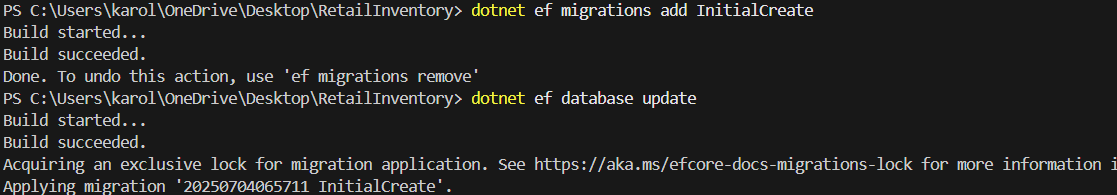
}

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

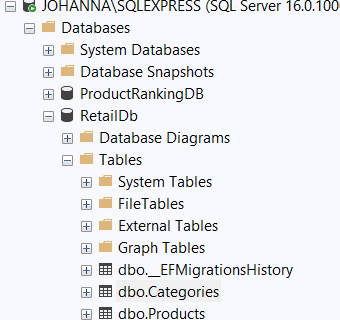
1. Install EF Core CLI :



1. Create Initial Migration & update to database:



1. Verify in SQL server:



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

1. Program.cs:

using System.Collections.Generic;

using System.Threading.Tasks;

class Program

{

    static async Task Main(string[] args)

    {

        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("Data inserted successfully!");

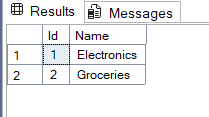
    }

}

1. Run the app:



1. Verify in SQL server:

****

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

1. Updated Program.cs for data retrieval:

using Microsoft.EntityFrameworkCore;

using RetailInventory.Models;

class Program

{

static async Task Main(string[] args)

{

using var context = new AppDbContext();

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

Console.WriteLine("All Products:");

foreach (var p in products)

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

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

Console.WriteLine($"\nFound by ID: {product?.Name}");

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

Console.WriteLine($"\nExpensive Product: {expensive?.Name}");

}

}

**FINAL OUTPUT:**

