# Lab 1: Understanding ORM with a Retail Inventory System

**1. What is ORM? (Object-Relational Mapping)**
ORM is a technique that maps C# classes (objects) to database tables so that you can interact with databases using code instead of SQL.

Eg:

| C# Class (Product) | SQL Table (Products) |
|---|---|
| int Id | INT Id PRIMARY KEY |
| string Name | VARCHAR Name |
| decimal Price | DECIMAL Price |

Benefits:
Productivity: No manual SQL needed.
Maintainability: Changes in models reflect in DB with migrations.
Abstraction: You work in C#, EF handles the SQL.

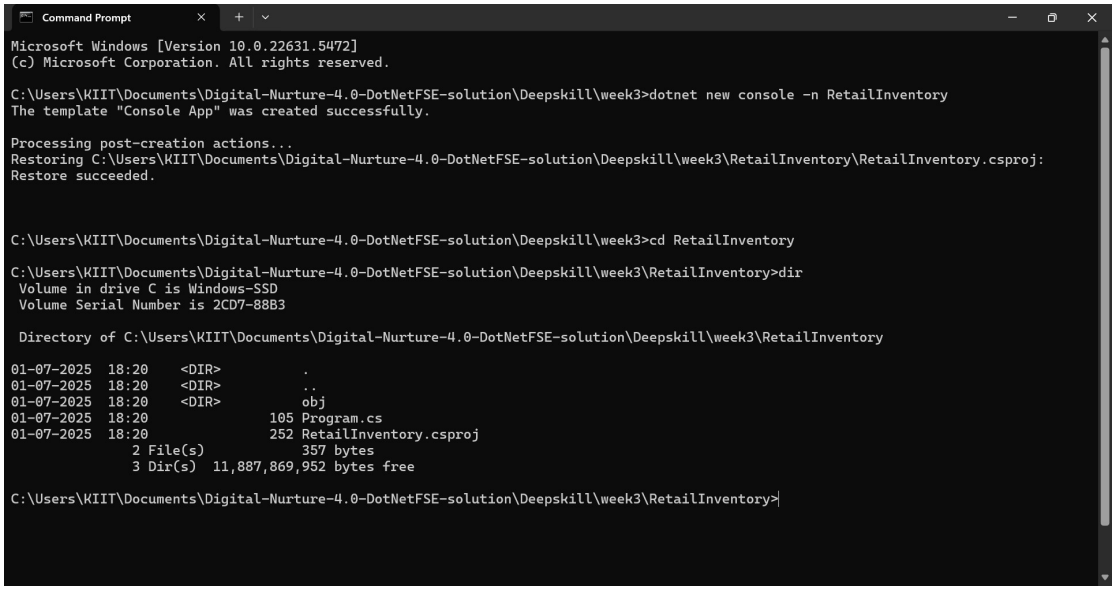
**2. EF Core vs EF Framework**

| Feature | EF Core (8.0) | EF Framework (6.x) |
|---|---|---|
| Cross-platform | Yes | No (Windows only) |
| Lightweight | Yes | Heavy |
| LINQ Support | Yes | Yes |
| Async Queries | Better supported | Limited |
| Performance | Compiled models = Faster | Slower |
| Future Support | Active development | Legacy support only |

**3. New EF Core 8.0 Features**

JSON Column Mapping: Map entire object graphs to JSON columns.
Compiled Models: Speeds up startup by avoiding model building at runtime.
Interceptors: Track, log, or modify DB behavior (e.g., queries, commands).
Bulk Operations: More efficient large inserts/updates.

**4. Create a .Net Console App**



### 5. Install EF Core Packages

```
C:\Users\KIIT\Documents\Digital-Nurture-4.0-DotNetFSE-solution\Deepskill\week3\RetailInventory>dotnet list package
Project 'RetailInventory' has the following package references
   [net9.0]:
   Top-level Package                          Requested   Resolved
   > Microsoft.EntityFrameworkCore.Design     9.0.6       9.0.6
   > Microsoft.EntityFrameworkCore.SqlServer  9.0.6       9.0.6


C:\Users\KIIT\Documents\Digital-Nurture-4.0-DotNetFSE-solution\Deepskill\week3\RetailInventory>
```

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

### 1. Create Models

### Models.cs

```
namespace RetailInventory
{
    public class Category
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public List<Product> Products { get; set; } = new();
    }

    public class Product
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public decimal Price { get; set; }

        // Foreign Key
        public int CategoryId { get; set; }

        // Navigation Property
        public Category Category { get; set; }
    }
}
```

### AppDbContext.cs

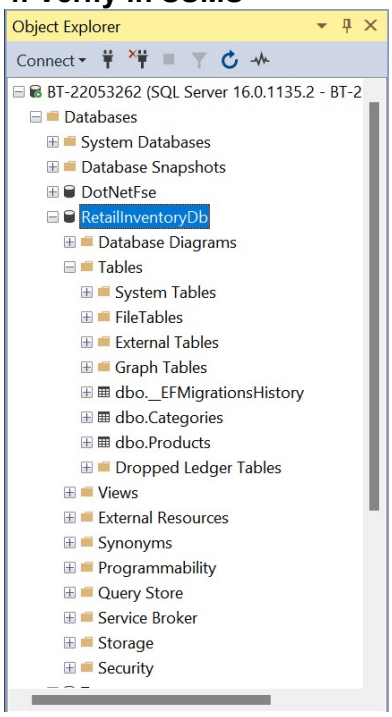
```
using Microsoft.EntityFrameworkCore;

namespace RetailInventory
{
    public class AppDbContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
        public DbSet<Category> Categories { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            // Replace this with your actual connection string
            optionsBuilder.UseSqlServer("Server=BT-
22053262;Database=RetailInventoryDb;Trusted_Connection=True;TrustServerCertificate=True;");
        }
    }
}
```

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

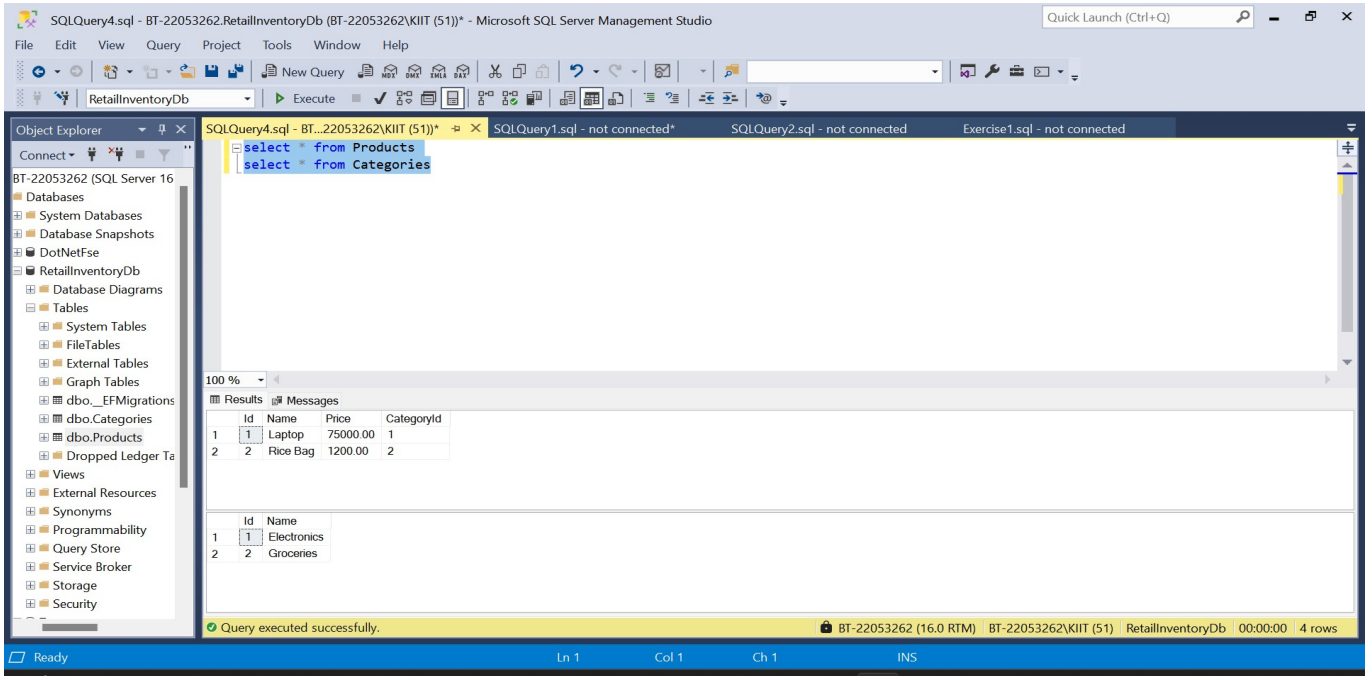
### 1. Install EF Core CLI
### 2. Create Initial Migration
### 3. Apply migration to initial Database

```
C:\Users\KIIT\Documents\Digital-Nurture-4.0-DotNetFSE-solution\Deepskill\week3\RetailInventory>dotnet tool install --global dotnet-ef
You can invoke the tool using the following command: dotnet-ef
Tool 'dotnet-ef' (version '9.0.6') was successfully installed.

C:\Users\KIIT\Documents\Digital-Nurture-4.0-DotNetFSE-solution\Deepskill\week3\RetailInventory>dotnet ef migrations add InitialCreate
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'

C:\Users\KIIT\Documents\Digital-Nurture-4.0-DotNetFSE-solution\Deepskill\week3\RetailInventory>dotnet ef database update
Build started...
Build succeeded.
Acquiring an exclusive lock for migration application. See https://aka.ms/efcore-docs-migrations-lock for more information if this ta
kes too long.
Applying migration '20250701133331_InitialCreate'.
Done.

C:\Users\KIIT\Documents\Digital-Nurture-4.0-DotNetFSE-solution\Deepskill\week3\RetailInventory>
```

### 4. Verify in SSMS




# Lab 4: Inserting Initial Data into the Database




# Lab 5: Retrieving Data from the Database