Lab 1: Understanding ORM with a Retail Inventory System

1. What is ORM?
ORM (Object-Relational Mapping) is a technique used in software development to map programming language objects (like C# classes) to database tables. It allows you to interact with a database using C# code instead of writing raw SQL queries.
Example Mapping:

C# Class	SQL Table
Product	Products
Name property	Name column

The ORM framework handles the conversion between your object-oriented code and the relational database structure behind the scenes.

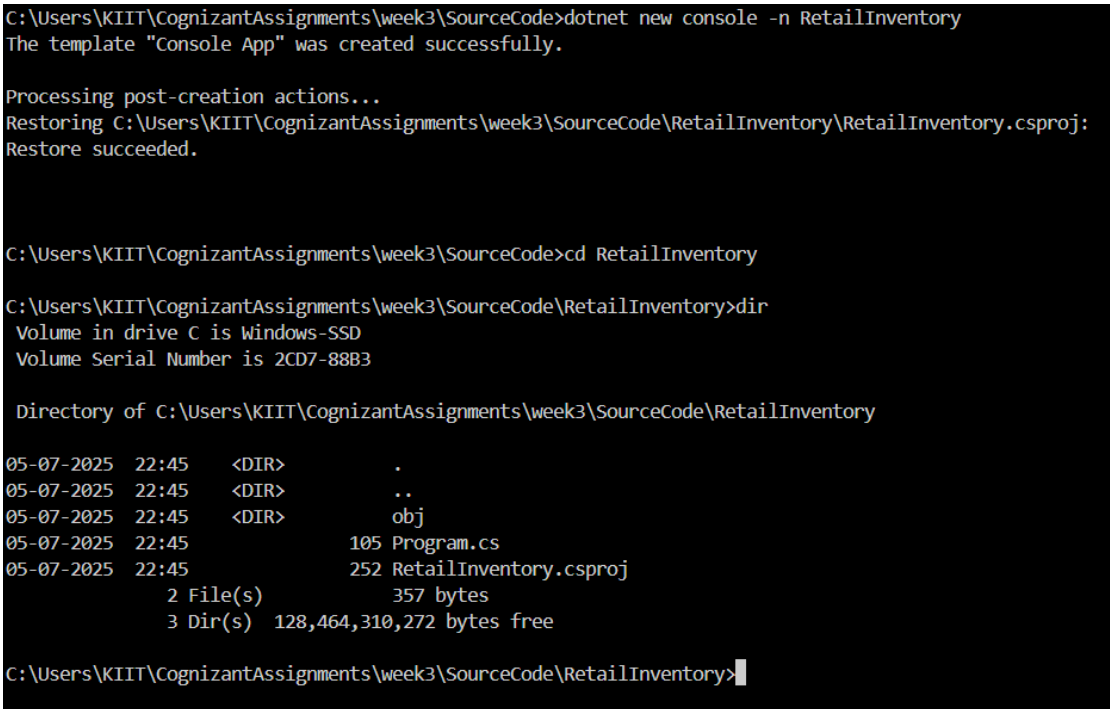
Benefits of ORM:
Productivity: Developers can focus on writing C# code without worrying about SQL queries.
Maintainability: Centralized models make the codebase easier to update and manage.
Abstraction: Hides SQL logic behind C# methods like .Add(), .Remove(), or .SaveChanges().

2. EF Core vs EF Framework (EF6)

Feature	EF Core	EF Framework (EF6)
Platform	Cross-platform (.NET Core/.NET 5+)	Windows-only
Lightweight	Yes	No (heavier)
Performance	Faster due to modern optimizations	Slower in comparison
LINQ & Async Support	Full support	Partial/less efficient
Compiled Queries	Supported (faster performance)	Not available
Use Case	Recommended for new development	Legacy support only

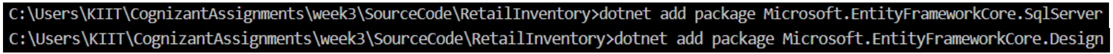
3. EF Core 8.0 Features
EF Core 8.0 brings several enhancements for better performance and modern use cases:
JSON Column Mapping: You can store and retrieve entire objects as JSON in a single SQL column.
Compiled Models: Reduces app startup time by avoiding model re-parsing.
Interceptors: Lets you hook into database commands for logging, auditing, etc.
Bulk Operations: Better handling of large inserts/updates in batches.

4. Create a .NET Console App:

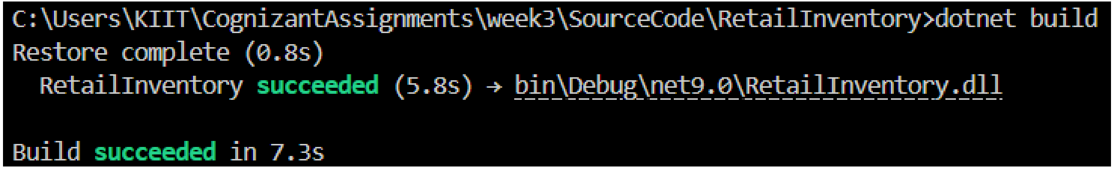


```
C:\Users\KIIT\CognizantAssignments\week3\SourceCode>dotnet new console -n RetailInventory
The template "Console App" was created successfully.

Processing post-creation actions...
Restoring C:\Users\KIIT\CognizantAssignments\week3\SourceCode\RetailInventory\RetailInventory.csproj:
Restore succeeded.



C:\Users\KIIT\CognizantAssignments\week3\SourceCode>cd RetailInventory

C:\Users\KIIT\CognizantAssignments\week3\SourceCode\RetailInventory>dir
 Volume in drive C is Windows-SSD
 Volume Serial Number is 2CD7-88B3

 Directory of C:\Users\KIIT\CognizantAssignments\week3\SourceCode\RetailInventory

05-07-2025  22:45    <DIR>          .
05-07-2025  22:45    <DIR>          ..
05-07-2025  22:45    <DIR>          obj
05-07-2025  22:45               105 Program.cs
05-07-2025  22:45               252 RetailInventory.csproj
               2 File(s)            357 bytes
               3 Dir(s)  128,464,310,272 bytes free

C:\Users\KIIT\CognizantAssignments\week3\SourceCode\RetailInventory>
```

5. Install EF Core Packages:

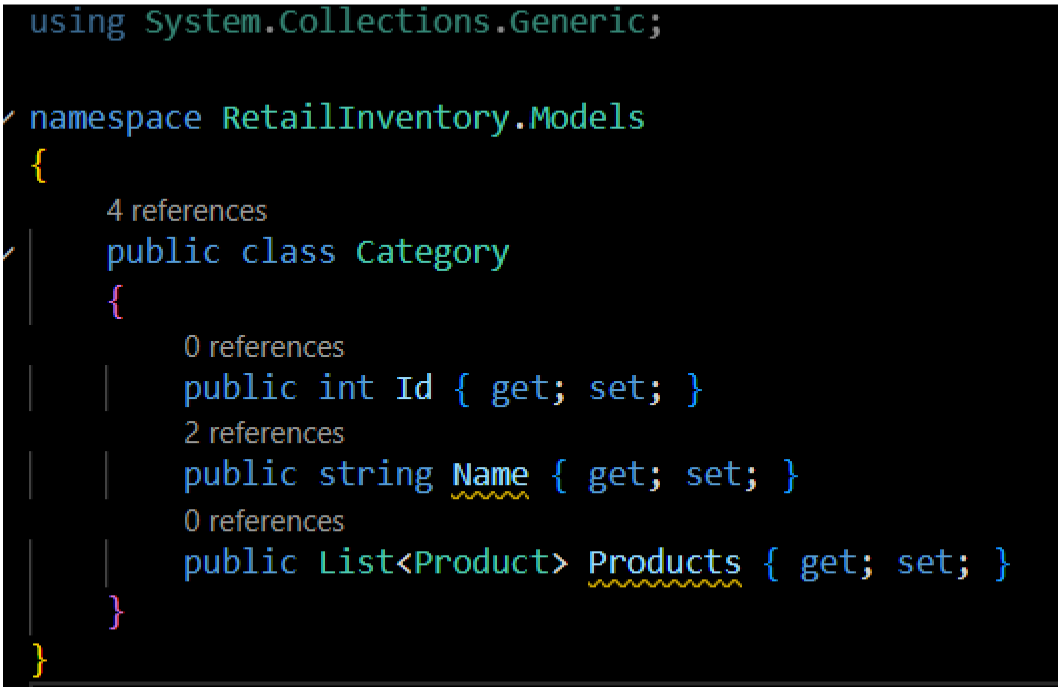


```
C:\Users\KIIT\CognizantAssignments\week3\SourceCode\RetailInventory>dotnet add package Microsoft.EntityFrameworkCore.SqlServer
C:\Users\KIIT\CognizantAssignments\week3\SourceCode\RetailInventory>dotnet add package Microsoft.EntityFrameworkCore.Design
```

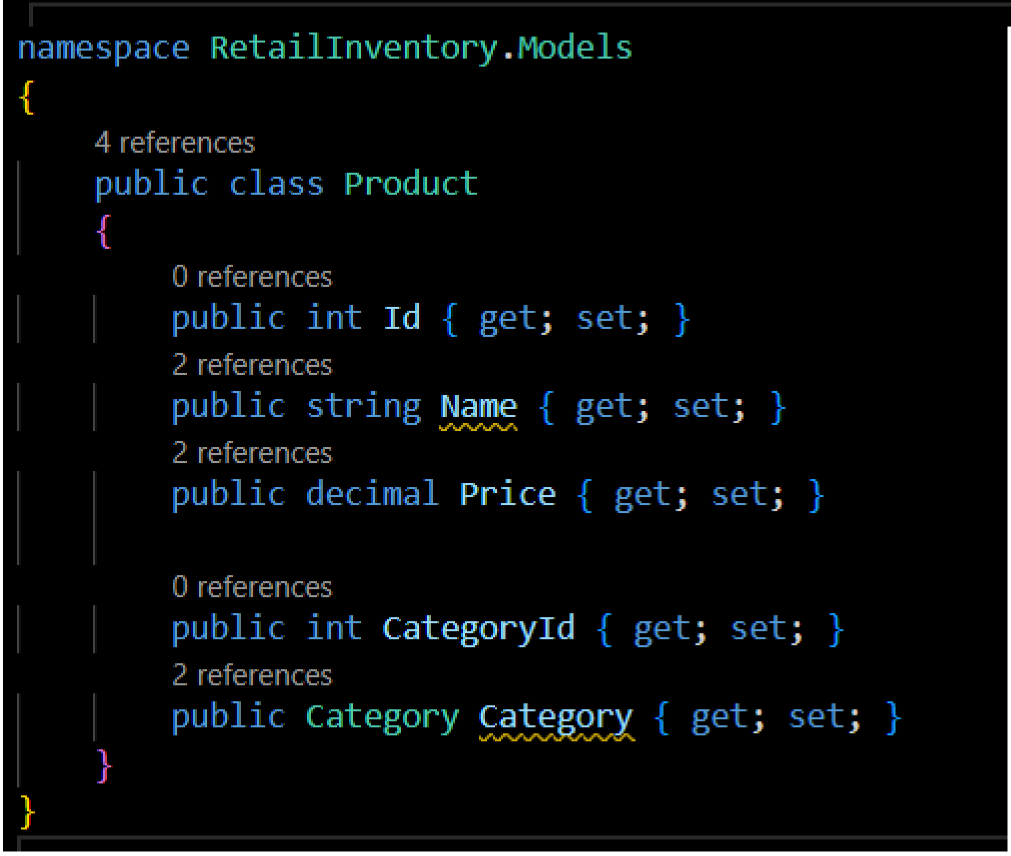


```
C:\Users\KIIT\CognizantAssignments\week3\SourceCode\RetailInventory>dotnet build
Restore complete (0.8s)
  RetailInventory succeeded (5.8s) → bin\Debug\net9.0\RetailInventory.dll

Build succeeded in 7.3s
```

Lab 2: Setting Up the Database Context for a retail Store

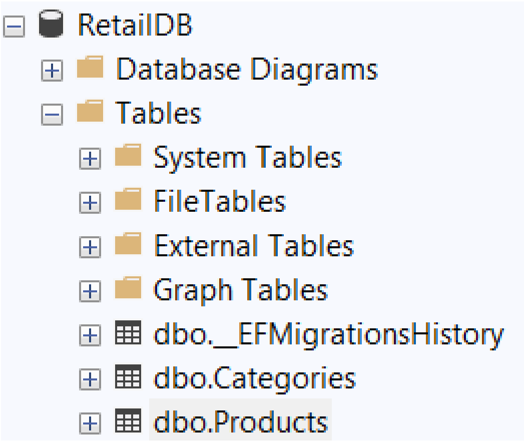


```
using System.Collections.Generic;

namespace RetailInventory.Models
{
        4 references
        public class Category
        {
                0 references
                public int Id { get; set; }
                2 references
                public string Name { get; set; }
                0 references
                public List<Product> Products { get; set; }
        }
}
```



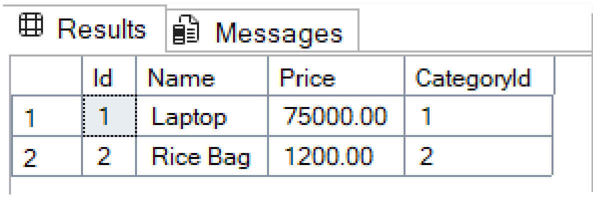
```
namespace RetailInventory.Models
{
        4 references
        public class Product
        {
                0 references
                public int Id { get; set; }
                2 references
                public string Name { get; set; }
                2 references
                public decimal Price { get; set; }

                0 references
                public int CategoryId { get; set; }
                2 references
                public Category Category { get; set; }
        }
}
```

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

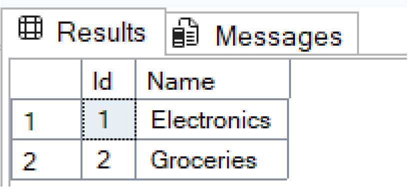


- RetailDB
  - Database Diagrams
  - Tables
    - System Tables
    - FileTables
    - External Tables
    - Graph Tables
    - dbo.__EFMigrationsHistory
    - dbo.Categories
    - dbo.Products

Lab 4: Inserting Initial Data into the Database

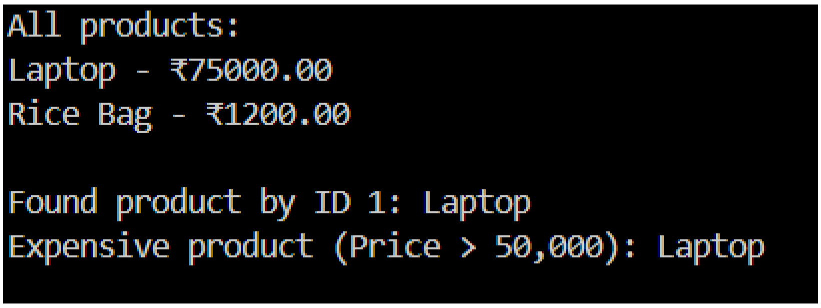


	Id	Name	Price	CategoryId
1	1	Laptop	75000.00	1
2	2	Rice Bag	1200.00	2



	Id	Name
1	1	Electronics
2	2	Groceries

Lab 5: Retrieving Data from the Database



```
All products:
Laptop - ₹75000.00
Rice Bag - ₹1200.00

Found product by ID 1: Laptop
Expensive product (Price > 50,000): Laptop
```