Lab 1: Understanding ORM with a Retail Inventory System

1. What is ORM?

ORM (Object-Relational Mapping) is a technique that allows developers to interact with a relational database using object-oriented programming languages like C#. Instead of writing raw SQL queries, you can use objects to perform CRUD operations.

How ORM maps C# classes to DB tables:

- A C# class (e.g., `Product`) becomes a database table.
- Class properties (e.g., `Name, Price`) become columns in that table.
  - EF Core handles the conversion between your C# objects and the SQL data behind the scenes.

2. Benefits of Using ORM

- **Productivity**: Write less boilerplate code.
- **Maintainability**: Centralize logic in your models.
- **Abstraction**: Avoid complex SQL; use LINQ instead.
- **Portability**: Easily switch databases.

3. EF Core vs EF Framework

Feature EF Core EF Framework (EF6) Platform Cross-platform (.NET Core)

Windows-only (.NET Framework) Performance Lightweight, faster Heavier, more stable LINQ

+ Async Support Yes Limited

Compiled Queries Yes No

Maturity Newer More mature

4. EF Core 8.0 New Features

- **JSON Column Mapping**: Store complex objects directly in a single column. ●

Compiled Models: Speeds up startup performance for large databases. ●

Interceptors: Hook into database calls for logging or validation. ● **Bulk**

Operations Improvements: More efficient insert/update/delete.

5. Project Setup

Create a .NET Console App:

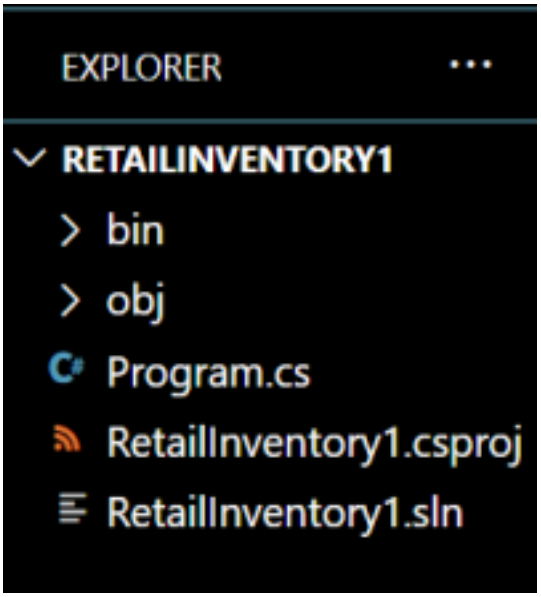
```
dotnet new console -n RetailInventory
```

Install EF Core Packages:

```
dotnet add package Microsoft.EntityFrameworkCore.SqlServer
dotnet add package Microsoft.EntityFrameworkCore.Design
```

This sets up EF Core in your project and prepares it to work with a SQL Server database.

After Setup:



Rahul Srivastava
Superset id : 6360503