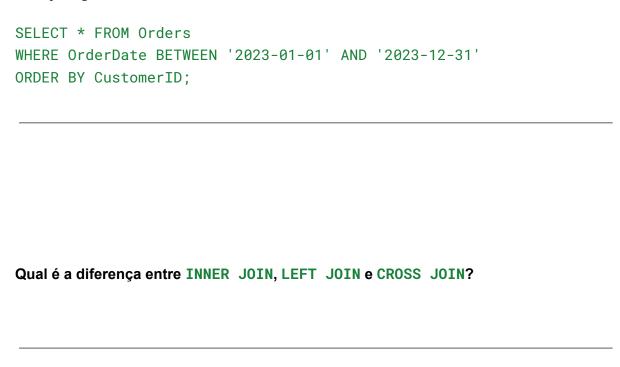
## Seção 1: SQL

| Escreva uma query que | otimize a seguinte | consulta para | grandes | volumes de | dados. |
|-----------------------|--------------------|---------------|---------|------------|--------|
| Query original:       |                    |               |         |            |        |



# Seção 2: Entity Framework

Explique a diferença entre os modos de carregamento Lazy, Eager e Explicit no Entity Framework.

Descreva um cenário onde o uso de AsNoTracking() seria necessário e explique por que.

### Seção 3: Arquitetura de Software

Explique a diferença entre arquitetura monolítica e arquitetura de microsserviços.

### Seção 4: Problema Prático

Sua tarefa é desenvolver uma API simples utilizando .NET Core e Entity Framework Core que permita o gerenciamento de produtos. A aplicação deve expor uma API RESTful que permita operações básicas de CRUD (Criar, Ler, Atualizar e Deletar) para produtos.

### **Requisitos Funcionais:**

- 1. Gerenciamento de Produtos:
  - Deve ser possível criar, atualizar, listar e deletar produtos.
  - Cada produto deve ter as seguintes propriedades:
    - ProductID: Identificador único (gerado automaticamente).
    - Name: Nome do produto (obrigatório).
    - Price: Preço do produto (obrigatório, deve ser maior que zero).
    - StockQuantity: Quantidade disponível em estoque (obrigatório, deve ser maior ou igual a zero).

### Requisitos Não Funcionais:

- Arquitetura: Utilize boas práticas de arquitetura, como a separação de camadas para Models, Repositories e Controllers.
- Validação: Implemente validações básicas para garantir que o nome do produto, preço e quantidade sejam válidos.
- Documentação da API: Utilize o Swagger para documentar os endpoints da API.

### Entrega:

- 1. Suba o projeto em um repositório público no GitHub.
- 2. Inclua um arquivo README.md que contenha:
  - o Breve descrição do projeto.
  - o Instruções para rodar o projeto localmente (usando dotnet run).
  - Passos para acessar o Swagger para testar a API.

### Critérios de Avaliação:

- Funcionalidade: O sistema atende aos requisitos funcionais propostos?
- Boas Práticas: O código segue boas práticas de desenvolvimento, como a separação de responsabilidades?
- Documentação: O projeto está bem documentado, e o Swagger está configurado corretamente?
- Controle de Versão: O repositório no GitHub está organizado e com commits claros?