Dell IT Academy

Exercício completo

Consultas Query Syntax

Projeto C# com Entity Framework Core e Consultas LINQ

Após configurar e popular o banco de dados, você pode usar LINQ para realizar consultas.

a) Selecionar Todos os Produtos:

```
using (var context = new LojaDbContext())
{
        Console.WriteLine("CONSULTA a)\n");
        var produtos = (from p in context.Produtos
                        select p).ToList();
        foreach (var produto in produtos)
        {
                Console.WriteLine($"{produto.Nome} - R$ {produto.Preco}");
        }
}
```

b) Filtrar Produtos por Categoria “Eletrônicos”:

```
using (var context = new LojaDbContext())
{
        Console.WriteLine("CONSULTA b) n");
        var eletronicos = (from p in context.Produtos
                           where p.Categoria.Nome == "Eletrônicos"
                           select p).ToList();

        foreach (var produto in eletronicos)
        {
                Console.WriteLine(produto.Nome);
        }
}
```

c) Buscar Produtos com Preço Acima de Um Valor (1.000):

```
using (var context = new LojaDbContext())
{
        Console.WriteLine("CONSULTA c)\n");
        var produtosCaros = (from p in context.Produtos
                             where p.Preco > 1000
                             select p).ToList();

        foreach (var produto in produtosCaros)
        {
                Console.WriteLine($"{produto.Nome} - R$ {produto.Preco}");
        }
}
```

d) Ordenar Produtos por Preço:

```
using (var context = new LojaDbContext())
{
        Console.WriteLine("CONSULTA d)\n");
        var produtosOrdenados = (from p in context.Produtos
                                 orderby p.Preco
                                 select p).ToList();

        foreach (var produto in produtosOrdenados)
        {
                 Console.WriteLine($"{produto.Nome} - R$ {produto.Preco}");
        }
}
```

e) Agrupar Produtos por Categoria:

```
using (var context = new LojaDbContext())
{
        Console.WriteLine("CONSULTA e)\n");
        var produtosPorCategoria = (from p in context.Produtos
                                    group p by p.Categoria.Nome into g
                                    select new
                                    {
                                           Categoria = g.Key,
                                           Produtos = g.ToList()
                                    }).ToList();

        foreach (var grupo in produtosPorCategoria)
        {
               Console.WriteLine($"Categoria: {grupo.Categoria}");
               foreach (var produto in grupo.Produtos)
               {
                      Console.WriteLine($" - {produto.Nome}");
               }
        }
}
```

f) Calcular a Receita Total de Vendas:

```
using (var context = new LojaDbContext())
{
        Console.WriteLine("CONSULTA f)\n");
        var receitaTotal = (from v in context.Vendas
                            select v.Quantidade * v.Produto.Preco).Sum();
         Console.WriteLine($"Receita Total: R$ {receitaTotal}\n\n");
}
```

g) Listar todos os produtos junto com o nome de sua categoria (INNER JOIN).

```
using (var context = new LojaDbContext())
{
        Console.WriteLine("CONSULTA g)\n");
        var produtosComCategorias = from p in context.Produtos
                                    join c in context.Categorias on p.CategoriaId equals c.Id
                                    select new
                                    {
                                           ProdutoNome = p.Nome,
                                           CategoriaNome = c.Nome,
                                           Preco = p.Preco
                                     };

            Console.WriteLine("Produtos com suas categorias:");
                   foreach (var item in produtosComCategorias)
                   {
                       Console.WriteLine($"Produto: {item.ProdutoNome}, Categoria: {item.CategoriaNome}, Preço: R$ {item.Preco}");
                   }
}
```

h) Listar todos os produtos, incluindo aqueles que não possuem uma categoria associada (LEFT JOIN).

```
using (var context = new LojaDbContext())
{
    Console.WriteLine("CONSULTA h)\n");
    var produtosComCategoriasOuNao = from p in context.Produtos
                                     join c in context.Categorias on p.CategoriaId equals c.Id into categoriaJoin
                                     from c in categoriaJoin.DefaultIfEmpty()
                                     select new
                                     {
                                         ProdutoNome = p.Nome,
                                         CategoriaNome = c != null ? c.Nome : "Sem Categoria",
                                         Preco = p.Preco
                                     };

    Console.WriteLine("Produtos com ou sem categorias:");
    foreach (var item in produtosComCategoriasOuNao)
    {
        Console.WriteLine($"Produto: {item.ProdutoNome}, Categoria: {item.CategoriaNome}, Preço: R$ {item.Preco}");
    }
}
```

i) Listar as vendas com informações do produto e da categoria associada a cada venda (INNER JOIN – 3 TABELAS).

```
using (var context = new LojaDbContext())
{
    Console.WriteLine("CONSULTA i)\n");
    var vendasDetalhadas = from v in context.Vendas
                           join p in context.Produtos on v.ProdutoId equals p.Id
                           join c in context.Categorias on p.CategoriaId equals c.Id
                           select new
                           {
                               VendaId = v.Id,
                               ProdutoNome = p.Nome,
                               CategoriaNome = c.Nome,
                               Quantidade = v.Quantidade,
                               Total = v.Quantidade * p.Preco
                           };

    Console.WriteLine("Detalhes das Vendas:");
    foreach (var venda in vendasDetalhadas)
    {
        Console.WriteLine($"Venda ID: {venda.VendaId}, Produto: {venda.ProdutoNome}, Categoria: {venda.CategoriaNome}, Quantidade: {venda.Quantidade},
Total: R$ {venda.Total}");
```