04

4 - Clientes compraram produto mais vendido

Transcrição

Nesta aula o pedido do cliente foi: gerar um relatório sobre indivíduos que compraram os produtos mais vendidos.

Criar uma consulta dessa dimensão parece um pouco complexo, portanto, primeiro construímos um relatório simples e depois incrementamos ele!

Para começar, é preciso descobrir quais são os produtos com recordes de vendas! Como um produto equivale a uma faixa de música, é preciso encontrar quais são as faixas mais vendidas. Para averiguar essa informação criamos uma consulta, a `var faixasQuery` e nela inserimos o `from f in contexto.Faixas` e `select f`. Conforme o que fizemos nas consultas anteriores, vamos imprimir o resultado através do `foreach` e `Console.WriteLine`:

```
var faixaQuery =
from f in contexto.Faixas
select f;

foreach (var faixa in faixaQuery)
{
    Console.WriteLine(faixa.Nome);
}

Console.ReadKey();
```

Ao rodar a aplicação o resultado mostrado são todas as músicas, portanto, é preciso refinar a `Query` para obter uma consulta com as demais informações. Assim, faz-se necessário modificar o `select f` para trazer também algumas propriedades específicas. Dessa maneira, vamos alterá-lo para `select new` e dentro disso introduzimos as seguintes características:

- Id da faixa: `f.FaixaId`
- Nome da faixa: `f.Nome`
- Total de vendas relacionadas à faixa: `f.ItemNotaFiscais`

Agora, vamos refletir um pouco:

Para cada faixa `f` existem muitos itens de Nota fiscal que estão relacionados. Para cada `inf` - que equivale a item de nota fiscal - temos um `total`. Portanto, é preciso obter a soma de todos os totais existentes, dessa forma, vamos utilizar o método `Sum` junto ao `f.ItemNotaFiscais`. Como não podemos somar objetos, apenas valores, vamos passar para o `Sum` uma expressão `lambda` cujo resultado a quantidade x preco: `inf => inf.Quantidade * inf.PrecoUnitario`. O código fica da seguinte maneira:

```
var faixasQuery =

from f in contexto.Faixas
select new
{
    f.FaixaId,
    f.Nome,
```

```
        Total = f.ItemNotaFiscais.Sum(inf => inf.Quantidade * inf.PrecoUnitario)

    };

    foreach (var faixa in faixaQuery)
    {
        Console.WriteLine(faixa.Nome);
    }

    Console.ReadKey();
```

O que estamos fazendo é uma projeção, o que significa que estamos pegando o `inf` - item de nota fiscal - e projetando na expressão que multiplica a quantidade pelo preço unitário.

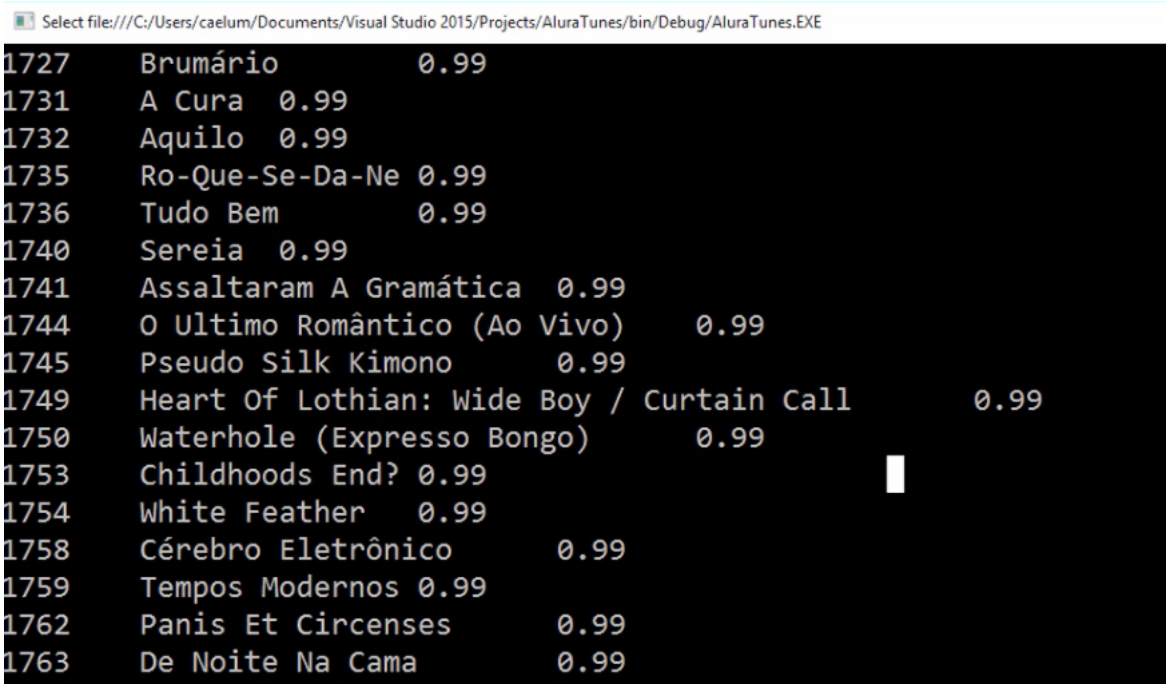
Após adicionar novas propriedades é preciso trazê-las também para o `select`. Assim, adicionamos `Console.WriteLine()`, `faixa.FaixaId`, `faixa.Nome` e `faixa.Total`. Além disso acrescentamos também formatação `"{0}t\{1}\t{2}"`. Teremos:

```
    foreach (var faixa in faixaQuery)
    {
        Console.WriteLine(faixa.Nome);
    }

    Console.ReadKey("{0}t\{1}\t{2}");
```

Entretanto, veremos que ocorrerá um erro! Essa falha acontece, pois existem algumas faixas que não possuem nenhuma venda realizada, portanto, não é possível somar os valores de uma coleção vazia. Assim, faz-se necessário filtrar a consulta para pegar apenas a faixas que possuem vendas realizadas. Dessa forma, vamos introduzir um filtro através do `where f.ItemNotaFiscais` e junto disso adicionamos o método `Count()` que irá trazer esses itens. Também introduzimos a condição de que os objetos sejam maiores que zero através de `> 0`. O código fica da seguinte maneira:

```
    var faixasQuery =
    from f in contexto.Faixas
    where f.ItemNotaFiscais.Count() > 0
    select new
```

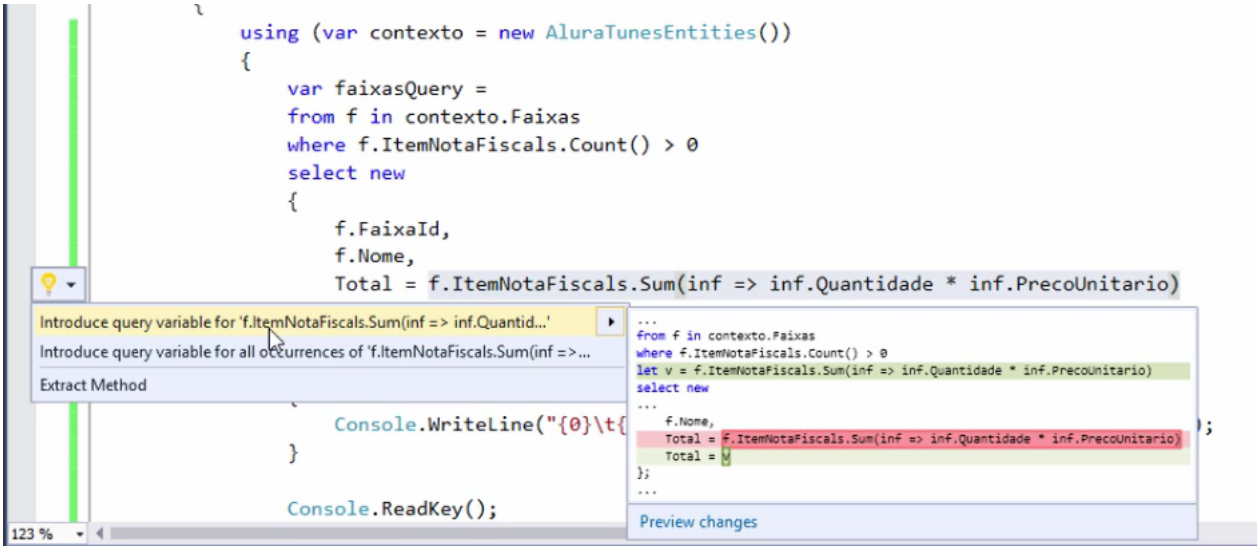
Agora, rodando a consulta o resultado são itens maiores do que zero:



Nosso objetivo também é ordenar a consulta do maior para o menor e assim verificar mais facilmente qual é o produto mais vendido na loja virtual. Para isso, vamos utilizar o `orderby` e junto podemos acrescentar a seguinte expressão:

```
f.ItemNotaFiscais.Sum(inf => inf.Quantidade * inf.PrecoUnitario)
```

Mas ao fazer isso, estaríamos repetindo-a em dois lugares. Para evitar que isso ocorra armazenamos a expressão em um lugar que permita seu reuso. Vamos guardá-la em uma variável interna a consulta. Então, iremos desfazer o `order by`, depois selecionaremos a expressão recém adicionada e vamos extrair a variável interna utilizando o comando `Ctrl ..` :



Assim, temos uma nova variável que renomeamos de `TotalDeVendas`. Abaixo do `where`, vamos declarar a variável:

```
let TotalDeVendas = f.ItemNotaFiscals.Sum(inf => inf.Quantidade * inf.PrecoUnitario)
```

No fim, adicionaremos o `Total` :

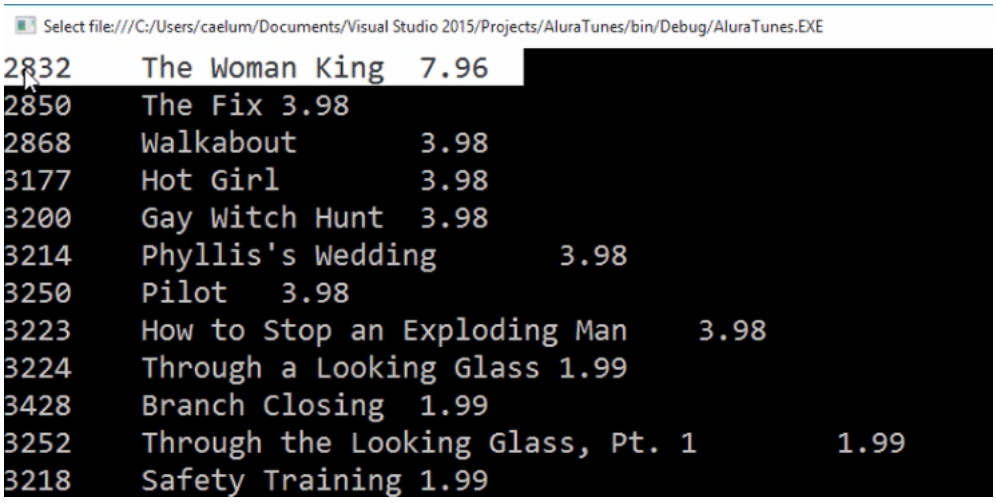
```
var faixaQuery =
from f in contexto.Faixas
where f.ItemNotaFiscals.Count() > 0
```

```
where f.ItemNotaFiscals.Count() > 0
let TotalDeVendas = f.ItemNotaFiscals.Sum(inf => inf.Quantidade * inf.PrecoUnitario)
select new
{
    f.faixaId,
    f.Nome,
    Total = TotalDeVendas
};

foreach (var faixa in faixasQuery)

{
    Console.WriteLine("{0}\t{1}\t{2}", faixa.FaixaId, faixa.Nome, faixa.Total);
}
```

Agora, vamos trabalhar com agrupamento de elementos, por isso, adicionaremos o `orderby` , juntamente com o `TotalDeVendas` e o `descending` - desta forma a ordem dos objetos será decrescente:

```
var faixaQuery =
from f in contexto.Faixas
where f.ItemNotaFiscals.Count() > 0
let TotalDeVendas = f.ItemNotaFiscals.Sum(inf => inf.Quantidade * inf.PrecoUnitario)
orderby TotalDeVendas descending
select new
```

Rodando a aplicação teremos um resultado que traz uma ordem do maior elemento para o menor:



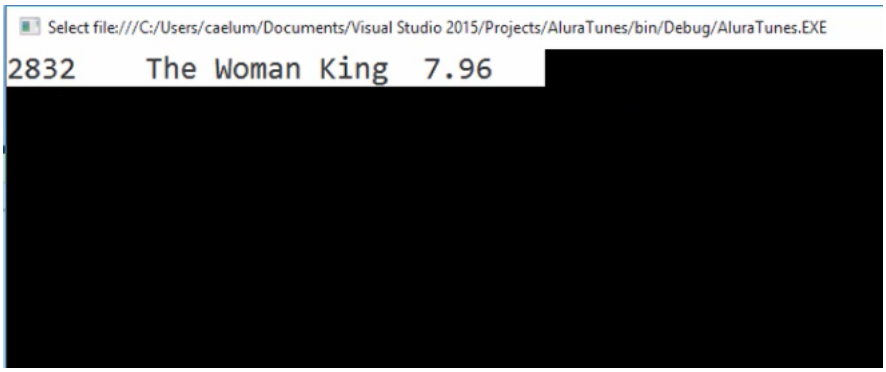
No topo da aplicação é possível verificar a faixa mais vendida de Id `2832` , nome `The Woman King` e valor `7.96` .

Uma vez que a lista está ordenada podemos descobrir qual é a faixa mais vendida, assim, vamos armazenar o resultado recém encontrado em uma `Query` . Desta maneira, excluímos o `foreach` e acrescentamos a variável `produtoMaisVendido` que será igual a `faixasQuery` . Como queremos pegar o primeiro item da lista adicionamos o método `First()` e imprimimos os dados do produto mais vendido, substituindo o nome da `faixa` por `produtoMaisVendido.FaixaId` , `produtoMaisVendido.Nome` e `produtoMaisVendido.Total` . O código ficará da seguinte maneira:

```
var faixaQuery =
from f in contexto.Faixas
where f.ItemNotaFiscals.Count() > 0
let TotalDeVendas = f.ItemNotaFiscals.Sum(inf => inf.Quantidade * inf.PrecoUnitario)
```

```
orderby TotalDeVendas descending
select new

{
    f.faixaId,
    f.Nome,
    Total = TotalDeVendas
};

var produtoMaisVendido = faixasQuery.First();

Console.WriteLine("{0}\t{1}\t{2}", produtoMaisVendido.FaixaId, produtoMaisVendido.Nome, produtoMais\

Console.ReadKey();
```

Ao rodar a aplicação o resultado mostra apenas os dados do produto mais vendido:



O próximo passo é encontrar os clientes que compraram o produto mais vendido. Para fazer isso é preciso navegar pelo item de nota fiscal e a partir disso chegar nessas pessoas. Desta forma, vamos criar uma nova consulta introduzindo uma nova variável de consulta: a `query`. Dentro dela, vamos inserimos o `from inf in contexto.ItemsNotaFiscal` e o `select inf`. Como o objetivo é filtrar as notas fiscais conforme o produto mais vendido, adicionamos a cláusula `where` junto do `inf.FaixaId` que será igual a `produtoMaisVendido.FaixaId`:

```
var query =
from inf in contexto.ItemsNotaFiscal
where inf.FaixaId == produtoMaisVendido.FaixaId
select inf
```

Falta modificar o `select` e incluir nele, em vez do objeto `ItemNotaFiscal`, os clientes que compraram os `Itens` mais vendidos. Desta forma, criamos um objeto anônimo, o `select new`. Dentro disso, incluiremos o nome do cliente que comprou o produto. Para fazer isso vamos declarar:

```
NomeCliente = inf.NotaFiscal.Cliente
```

Concatenaremos os elementos de primeiro nome e também sobrenome:

```
PrimeiroNome + " " + inf.NotaFiscal.Cliente.Sobrenome
```

Com as alterações o código ficará assim:

```
var query =
from inf in contexto.ItemsNotaFiscal
where inf.FaixaId == produtoMaisVendido.FaixaId
select new
{

    NomeCliente = inf.NotaFiscal.Cliente.PrimeiroNome + " " + inf.NotaFiscal.Cliente.Sobrenome
};
```

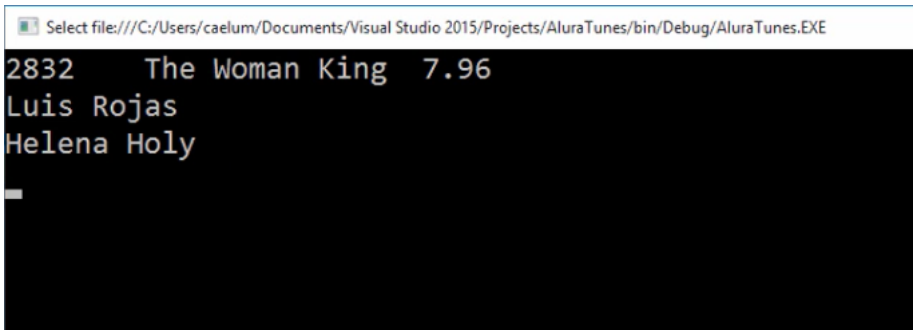
Na sequência, vamos imprimir os resultados na aplicação, portanto, adicionaremos o laço de tipo `foreach()` para o qual passaremos as informações a serem impressas na query. Ainda, adicionamos a seguinte linha:

```
Console.WriteLine(cliente.NomeCliente)
```

O trecho ficará assim:

```
var query =
from inf in contexto.ItemsNotaFiscal
where inf.FaixaId == produtoMaisVendido.FaixaId
select new
{
    NomeCliente = inf.NotaFiscal.Cliente.PrimeiroNome + " " + inf.NotaFiscal.Cliente.Sobrenome
};

foreach (var cliente in query)
{
    Console.WriteLine(cliente.NomeCliente);
}
```

Rodando o código temos o seguinte resultado:



O Luis Rojas e o Helena Holy são os clientes que adquiriram os produtos mais vendidos!

O que fizemos nesta aula?

Conhecemos o conceito de variável interna. Fizemos uso da variável interna em dois lugares diferentes e também para armazenar uma expressão mais complexa! Aprendemos, ainda, a fazer uma projeção de dados para calcular a soma de uma expressão.