01

1 - Linq to entities min max avg

Transcrição

O último pedido do cliente é calcular o "preço da maior", "menor venda" e "venda média". Para fazer isso é preciso, primeiro, entender o significado de venda. Uma venda consiste em:

1. Escolher os produtos;
2. Colocá-los no carrinho;
3. Efetuar o pagamento;
4. A loja aprovar o pedido;
5. Recebimento do e-mail de confirmação com nota fiscal;

Para o sistema, uma nota fiscal é a representação de uma venda.

Para realizar o pedido do cliente vamos utilizar a entidade "Notas fiscais" que será utilizada para calcular os valores desejados. Primeiro, vamos construir o código da maior venda. Para isso começaremos por declarar o contexto, assim, vamos escrever `using()` e usaremos a seguinte linha:

```
var contexto = new AluraTunesEntities()
```

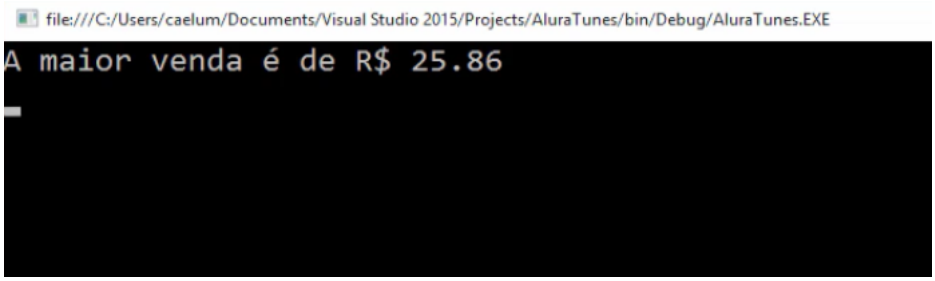
A seguir, declaramos uma variável que representa a maior venda, a `maiorVenda` , e igualaremos isso a `contexto.NotasFiscais` , junto disso colocaremos a função `Max()` que retornará o maior valor de `NotasFiscais` do sistema. Nós vamos inserir dentro do `Max()` o `nf` e a propriedade `Total` . Desta forma, teremos `nf => nf.Total` . Falta imprimir o resultado, para tanto, escreveremos `Console.WriteLine()` e imprimiremos a `maiorVenda` . Também adicionaremos o formato `"A maior venda é de R$ {0}"` . Por último vamos acrescentar o `Console.ReadKey()` . Nosso código ficará da seguinte maneira:

```
class Program
{
    static void Main(string[] args)
    {
        using (var contexto = new AluraTunesEntities())
        {
            var maiorVenda = contexto.NotasFiscais.Max(nf => nf.Total);

            Console.WriteLine("A maior venda é de R$ {0}", maiorVenda);
        }

        Console.ReadKey();
    }
}
```

Rodando isso temos o seguinte resultado:



Agora que a maior venda já está calculada, vamos encontrar a menor venda. Por isso, vamos inserir a linha:

```
var menorVenda = contexto.NotasFiscais
```

Junto disso, adicionaremos a função `Min()` que deve trazer o valor mínimo. Passaremos para `Min()` a expressão `lambda` que pega a propriedade de nota fiscal e o total:

```
var menorVenda = contexto.NotasFiscais.Min(nf => nf.Total)
```

Ebaixo do `Console.WriteLine()` colocaremos outro `Console.WriteLine()`:

```
Console.WriteLine("A menor venda é de R$ {0}", menorVenda);
```

Com isso, temos o seguinte código:

```
class Program
{
    static void Main(string[] args)
    {
        using (var contexto = new AluraTunesEntities())
        {
            var maiorVenda = contexto.NotasFiscais.Max(nf => nf.Total);
            var menorVenda = contexto.NotasFiscais.Max(nf => nf.Total);

            Console.WriteLine("A maior venda é de R$ {0}", maiorVenda);
            Console.WriteLine("A menor venda é de R$ {0}", menorVenda);
        }

        Console.ReadKey();
    }
}
```

Rodando isso, alcançaremos o seguinte resultado:

Agora temos os resultados de maior e menor venda respectivamente!

O próximo passo é inserir outra variável para representar a venda média:

```
var vendaMedia = contexto.NotasFiscais
```

Junto disso, adicionaremos a `Average()` que calcula a média e passamos também o `nf => nf.Total` . Teremos:

```
var vendaMedia = contexto.NotasFiscais.Average(nf => nf.Total)
```

Abaixo disso colocaremos o `Console.WriteLine("A venda média é de R$ {0}", menorVenda)` . O código ficará da seguinte maneira:

```
class Program
{
    static void Main(string[] args)
    {
        using (var contexto = new AluraTunesEntities())
        {
            var maiorVenda = contexto.NotasFiscais.Max(nf => nf.Total);
            var menorVenda = contexto.NotasFiscais.Max(nf => nf.Total);
            var vendaMedia = contexto.NotasFiscais.Average(nf => nf.Total);

            Console.WriteLine("A maior venda é de R$ {0}", maiorVenda);
            Console.WriteLine("A menor venda é de R$ {0}", menorVenda);
            Console.WriteLine("A venda média é de R$ {0}", vendaMedia);
        }

        Console.ReadKey();
    }
}
```

Rodando a aplicação o resultado é o seguinte:

Portanto, conseguimos realizar o pedido do cliente!

O próximo passo é descobrir como as consultas são feitas no banco de dados! Para tanto, vamos inserir uma instrução que pede ao contexto para que ele jogue todos os `logs` de `script` no `Console` . Assim, tudo que é gerado de consulta no `SQL` será mostrado no `Console` . Portanto, acima das variáveis vamos inserir:

```
contexto.Database.Log = Console.WriteLine
```

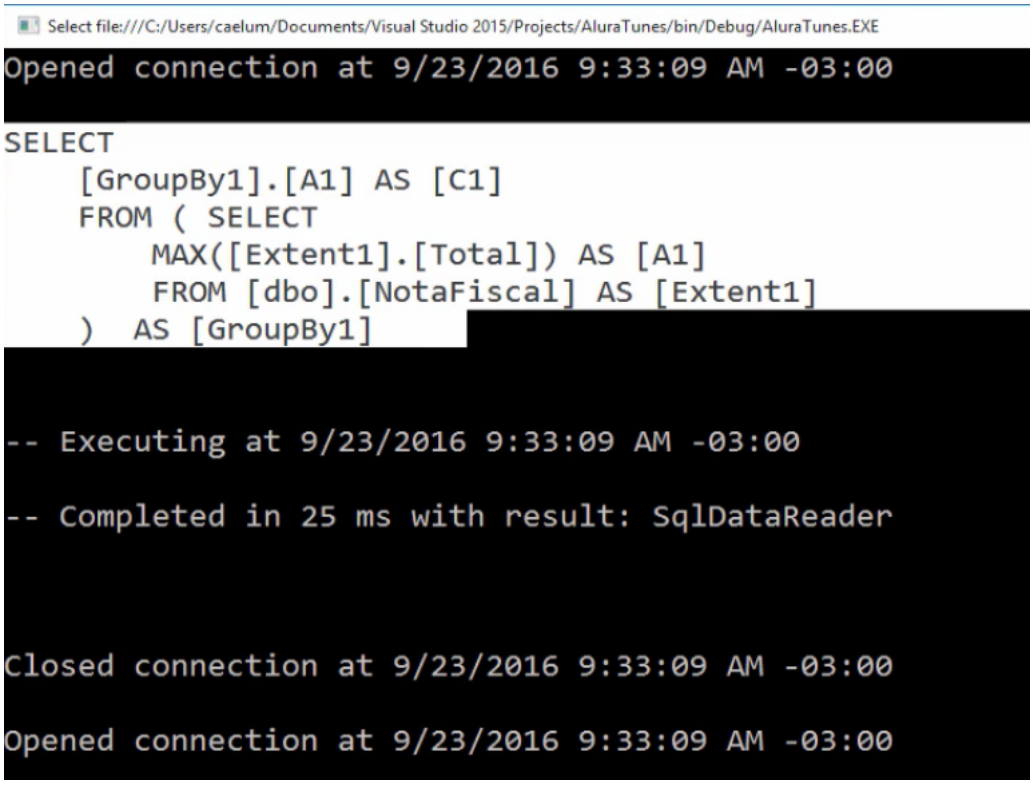
O código ficará da seguinte maneira:

```
using (var contexto = new AluraTunesEntities())
        {
            contexto.Database.Log = Console.WriteLine();

            var maiorVenda = contexto.NotasFiscais.Max(nf => nf.Total);
            var menorVenda = contexto.NotasFiscais.Max(nf => nf.Total);
            var vendaMedia = contexto.NotasFiscais.Average(nf => nf.Total);
```

```
            Console.WriteLine("A maior venda é de R$ {0}", maiorVenda);
            Console.WriteLine("A menor venda é de R$ {0}", menorVenda);
            Console.WriteLine("A venda média é de R$ {0}", vendaMedia);
        }
```

Rodando isso temos:



O resultado mostra três consultas sendo que cada uma possui uma tarefa distinta: uma calcula o valor máximo, a segunda o mínimo e a terceira o médio.

Seria bacana se conseguíssemos em apenas uma consulta trazer essas três informações. Desta forma, economizaremos no processamento do servidor SQL, no tráfico de rede e no tempo de processamento.

Para fazer uma consulta única declaramos a variável `vendas` e como vamos trazer as informações de Notas Fiscais, iremos inserir o `contexto.NotasFiscais` e abaixo disso colocaremos o `select nf` . Vamos inserir também a impressão das notas com o:

```
foreach (var venda in vendas)
```

Além do `Console.WriteLine()` .

Teremos:

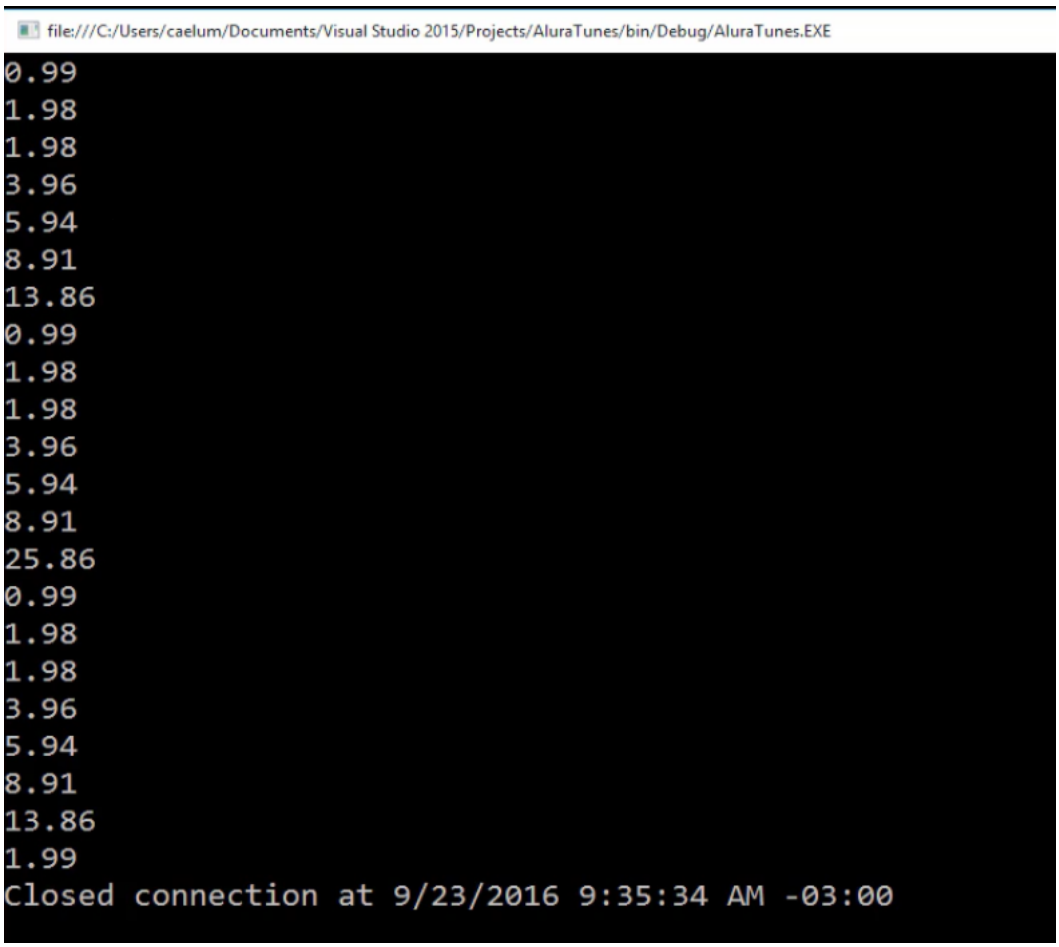
```
Console.WriteLine("A maior venda é de R$ {0}", maiorVenda);
Console.WriteLine("A menor venda é de R$ {0}", menorVenda);
Console.WriteLine("A venda média é de R$ {0}", vendaMedia);

var vendas = from nf in contexto.NotasFiscais
             select nf;

foreach (var vend in vendas)
```

```
{
    Console.WriteLine(venda.Total)
}
```

Rodando isso o resultado são todos os valores:



Agora, vamos trazer as maiores, menores e médias vendas. Para fazer isso vamos utilizar um pequeno truque, passaremos para o `group nf by` não uma propriedade, mas uma constante, a `1`. Passaremos também um objeto anônimo, o `select new`. Falta adicionar junto do `group nf by` o `into agrupado`. O `agrupado` contem informações de `agrupamento` da consulta, por isso, vamos inserir dentro do `select new` os valores: `maiorVenda = agrupado.Max(nf => nf.Total)`, o `menorVenda = agrupado.Min(nf => nf.Total)` e `vendaMedia = agrupado.Average(nf => nf.Total)`:

```
var vendas = from nf in contexto.NotasFiscais
             group nf by 1 into agrupado
             select nf;
             {
                 maiorVenda = agrupado.Max(nf => nf.Total),
                menorVenda = agrupado.Min(nf => nf.Total),
                vendaMedia = agrupado.Average(nf => nf.Total)
             }
```

Agora, é preciso transformar a `query` em objeto. Para fazer isso vamos envolver o código em parênteses e inserimos o método `Single` que é capaz de converter consulta em objeto:

```
var vendas =(from nf in contexto.NotasFiscais
             group nf by 1 into agrupado
```

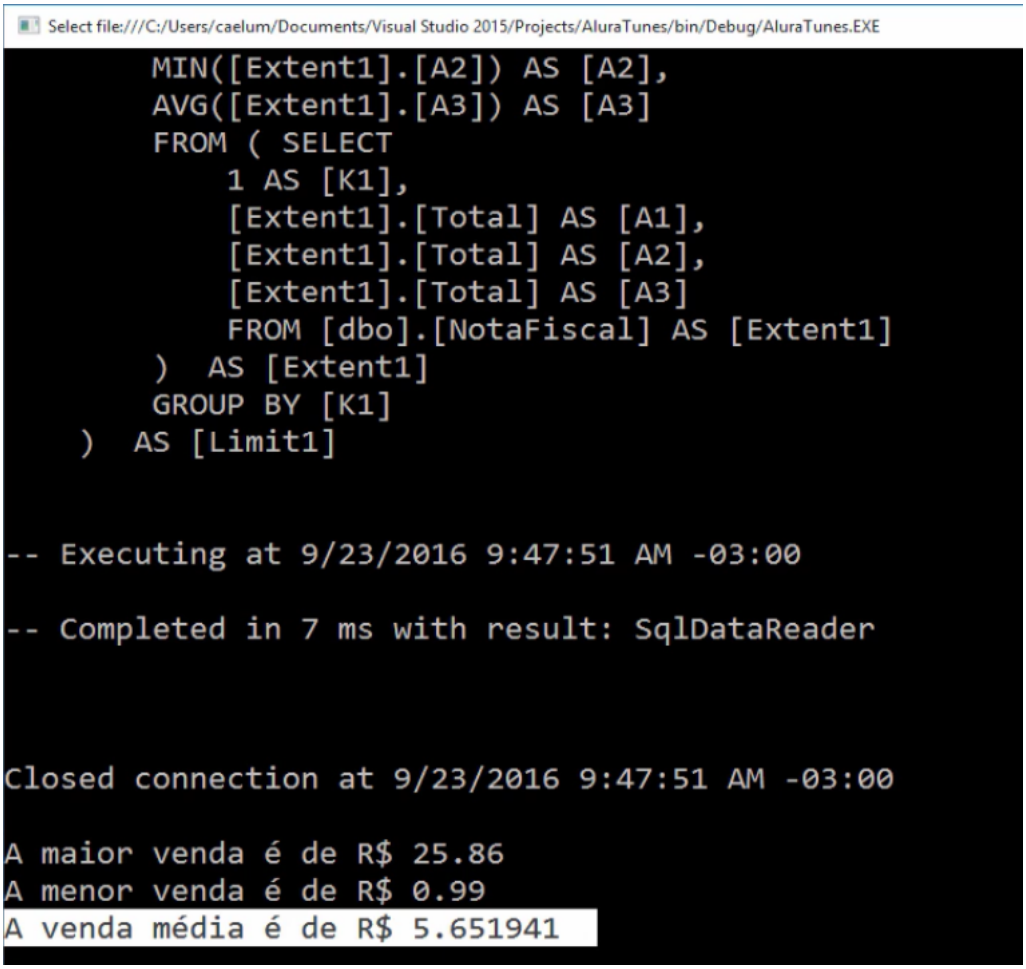
```
                select nf;
                {
                    maiorVenda = agrupado.Max(nf => nf.Total),
                   menorVenda = agrupado.Min(nf => nf.Total),
                   vendaMedia = agrupado.Average(nf => nf.Total)
                }).Single();
```

O objeto recém criado tem três propriedades: maiorVenda, menor venda e venda média. Vamos copiar as linhas referentes ao `Console` e substituir os valores de `maiorVenda` , `menorVenda` e `vendaMedia` por `vendas` :

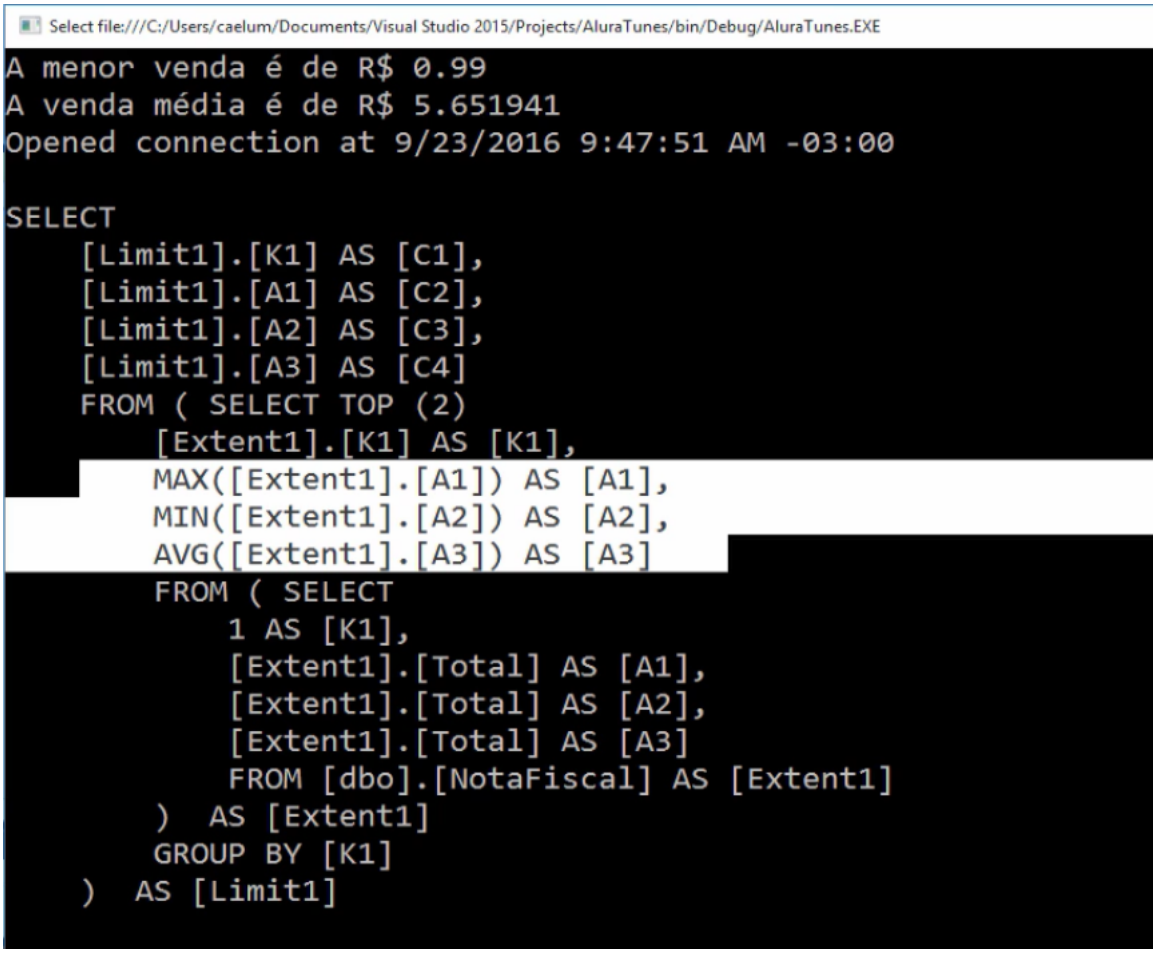
```
var vendas =(from nf in contexto.NotasFiscais
             group nf by 1 into agrupado
             select nf;
             {
                 maiorVenda = agrupado.Max(nf => nf.Total),
                menorVenda = agrupado.Min(nf => nf.Total),
                vendaMedia = agrupado.Average(nf => nf.Total)
             }).Single();

Console.WriteLine("A maior venda é de R$ {0}", vendas.maiorVenda);
Console.WriteLine("A menor venda é de R$ {0}", vendas.menorVenda);
Console.WriteLine("A venda média é de R$ {0}", vendas.vendaMedia);
```

Rodando isso temos exatamente o mesmo resultado de antes:



Observe também o `SQL` gerado. Através dele encontramos os valores máximo, mínimo e médio:



Ao fim desta aula, conseguimos chegar a uma única consulta no lugar das três iniciais! Dessa maneira economizamos tempo, tráfico de rede e processamento da aplicação. Também calculamos o mínimo, máximo e médio da consulta `LINQ` para `Entity Framework`.