11

Mãos na massa: Funções

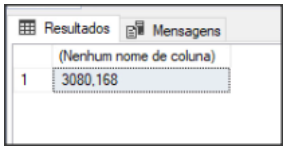
Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

1) Crie uma nova consulta para a base **SUCOS_VENDAS** .

2) Se você executar o comando abaixo:

```
SELECT SUM(QUANTIDADE * [PREÇO])
FROM [ITENS NOTAS FISCAIS]
WHERE NUMERO = 100
```

Você irá verificar o faturamento da nota fiscal 100:



3) Crie uma função que retornará o faturamento da nota. O parâmetro a ser passado será o número da nota fiscal:

```
CREATE FUNCTION FaturamentoNota (@NUMERO AS INT) RETURNS FLOAT
AS
BEGIN
    DECLARE @FATURAMENTO FLOAT
    SELECT @FATURAMENTO = SUM(QUANTIDADE * [PREÇO])
        FROM [ITENS NOTAS FISCAIS]
    WHERE NUMERO = @NUMERO
    RETURN @FATURAMENTO
END
```

- Você tem a declaração da função:

```
CREATE FUNCTION FaturamentoNota (@NUMERO AS INT) RETURNS FLOAT
AS
```

- Entre o **BEGIN** e **END** , o código de implementação:

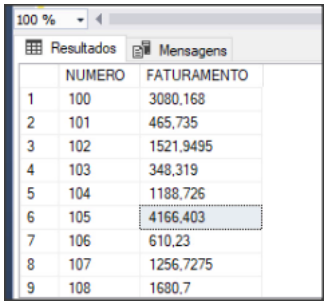
```
BEGIN
    DECLARE @FATURAMENTO FLOAT
    SELECT @FATURAMENTO = SUM(QUANTIDADE * [PREÇO])
        FROM [ITENS NOTAS FISCAIS]
    WHERE NUMERO = @NUMERO
    RETURN @FATURAMENTO
END
```

- Destaque para o comando **RETURN** , para retornar o valor da variável para ser o resultado da função:

```
RETURN @FATURAMENTO
```

4) Depois, você pode usar o comando `SELECT` com a função:

```
SELECT NUMERO,
    [dbo].[FaturamentoNota](NUMERO) AS FATURAMENTO
FROM [NOTAS FISCAIS]
```



	NUMERO	FATURAMENTO
1	100	3080,168
2	101	465,735
3	102	1521,9495
4	103	348,319
5	104	1188,726
6	105	4166,403
7	106	610,23
8	107	1256,7275
9	108	1680,7

5) Crie agora um função usando um *loop*. Crie uma nova consulta no SQL Server associados a base `SUCOS_VENDAS` .

6) Edite o script mostrado abaixo:

```
DECLARE @LIMITE_MINIMO INT,
    @LIMITE_MAXIMO INT,
    @CONTADOR_NOTAS INT
DECLARE @TABELA_NUMEROS TABLE (
    [NUMERO] INT,
    [STATUS] VARCHAR(200)
)

SET @LIMITE_MINIMO = 1
SET @LIMITE_MAXIMO = 100000

SET NOCOUNT ON
WHILE @LIMITE_MINIMO <= @LIMITE_MAXIMO
BEGIN
    SELECT @CONTADOR_NOTAS = COUNT(*) FROM [NOTAS FISCAIS]
        WHERE [NUMERO] = @LIMITE_MINIMO
    IF @CONTADOR_NOTAS > 0
    BEGIN
        INSERT INTO @TABELA_NUMEROS ([NUMERO], [STATUS])
            VALUES (@LIMITE_MINIMO, 'É nota fiscal')
    END
    ELSE
    BEGIN
        INSERT INTO @TABELA_NUMEROS ([NUMERO], [STATUS])
            VALUES (@LIMITE_MINIMO, 'Não é nota fiscal')
    END
    SET @LIMITE_MINIMO = @LIMITE_MINIMO + 1
END

SELECT * FROM @TABELA_NUMEROS
```

Acrescente mais um campo na tabela, que representará o faturamento da nota fiscal. Use a função do vídeo anterior para obter isso.

7) Modifique a declaração da variável que representa a tabela para:

```
DECLARE @TABELA_NUMEROS TABLE (
    [NUMERO] INT,
    [STATUS] VARCHAR(200),
    [FATURAMENTO] FLOAT
)
```

8) Quando o número contido na variável de contagem, dentro do *loop*, representar uma nota fiscal existente, insira a função no campo `FATURAMENTO`, da variável da tabela:

```
IF @CONTADOR_NOTAS > 0
BEGIN
    INSERT INTO @TABELA_NUMEROS ([NUMERO], [STATUS], [FATURAMENTO])
        VALUES (
            @LIMITE_MINIMO,
            'É nota fiscal',
            [dbo].[FaturamentoNota](@LIMITE_MINIMO)
        )
END
```

9) Em contrapartida, no comando `ELSE`, insira 0 quando aquele número não representar uma nota fiscal válida:

```
ELSE
BEGIN
    INSERT INTO @TABELA_NUMEROS ([NUMERO], [STATUS], [FATURAMENTO])
        VALUES (@LIMITE_MINIMO, 'Não é nota fiscal', 0)
END
```

10) Você terá o seguinte script final:

```
DECLARE @LIMITE_MINIMO INT,
    @LIMITE_MAXIMO INT,
    @CONTADOR_NOTAS INT
DECLARE @TABELA_NUMEROS TABLE (
    [NUMERO] INT,
    [STATUS] VARCHAR(200),
    [FATURAMENTO] FLOAT
)

SET @LIMITE_MINIMO = 1
SET @LIMITE_MAXIMO = 1000000

SET NOCOUNT ON
WHILE @LIMITE_MINIMO <= @LIMITE_MAXIMO
BEGIN
    SELECT @CONTADOR_NOTAS = COUNT(*) FROM [NOTAS FISCAIS]
        WHERE [NUMERO] = @LIMITE_MINIMO
    IF @CONTADOR_NOTAS > 0
```

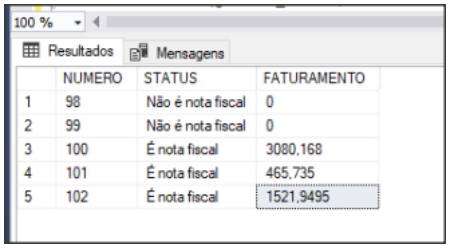
```
    BEGIN
        INSERT INTO @TABELA_NUMEROS ([NUMERO], [STATUS], [FATURAMENTO])
            VALUES (
                @LIMITE_MINIMO,
                'É nota fiscal',
                [dbo].[FaturamentoNota](@LIMITE_MINIMO)
            )
    END
    ELSE
    BEGIN
        INSERT INTO @TABELA_NUMEROS ([NUMERO], [STATUS], [FATURAMENTO])
            VALUES (@LIMITE_MINIMO, 'Não é nota fiscal', 0)
    END
    SET @LIMITE_MINIMO = @LIMITE_MINIMO + 1
END

SELECT * FROM @TABELA_NUMEROS
```

11) Modifique os parâmetros iniciais do script:

```
SET @LIMITE_MINIMO = 98
SET @LIMITE_MAXIMO = 102
```

E executando-o, você terá:



	NUMERO	STATUS	FATURAMENTO
1	98	Não é nota fiscal	0
2	99	Não é nota fiscal	0
3	100	É nota fiscal	3080,168
4	101	É nota fiscal	465,735
5	102	É nota fiscal	1521,9495

12) Faça com que o retorno da função seja uma tabela. Crie uma nova consulta usando a base de dados **SUCOS_VENDAS** .

13) Se você executar a consulta abaixo:

```
SELECT * FROM [NOTAS FISCAIS] WHERE CPF = '1471156710'
```

Terá a listagem de todas as notas fiscais de um cliente.

14) Você pode ter este retorno usando uma função cujo parâmetro a ser passado será o CPF do cliente. Para isso, faça o script abaixo:

```
CREATE FUNCTION ListaNotasCliente (
    @CPF AS VARCHAR(12)
) RETURNS TABLE
AS
RETURN SELECT * FROM [NOTAS FISCAIS] WHERE CPF = @CPF
```

Execute para criar a função.

15) Agora, a função vai funcionar como se fosse uma tabela:

```
SELECT * FROM [dbo].[ListaNotasCliente]('1471156710')

SELECT COUNT(*) FROM [dbo].[ListaNotasCliente]('1471156710')
```

16) Inclusive usando-a em um `JOIN`:

```
SELECT A.CPF, A.NUM_NOTA, B.TOTAL_FATURAMENTO FROM (
SELECT CPF, (
    SELECT COUNT(*) FROM
        [dbo].[ListaNotasCliente](CPF)
) AS NUM_NOTA
FROM [TABELA DE CLIENTES]) A INNER JOIN (
    SELECT CPF,
    SUM([dbo].[FaturamentoNota](NUMERO))
        AS TOTAL_FATURAMENTO
FROM [NOTAS FISCAIS] GROUP BY CPF) B
ON A.CPF = B.CPF
```

17) Crie uma função que concatena o endereço do cliente. Para isso, execute o código abaixo:

```
CREATE FUNCTION EnderecoCompleto (
    @ENDERECO VARCHAR(100),
    @CIDADE VARCHAR(50),
    @ESTADO VARCHAR(50),
    @CEP VARCHAR(20)
)
RETURNS VARCHAR(250)
AS
BEGIN
    DECLARE @ENDERECO_COMPLETO VARCHAR(250)
    SET @ENDERECO_COMPLETO = @ENDERECO + ' - ' +
        @CIDADE + ' - ' + @ESTADO + ' - ' + @CEP
    RETURN @ENDERECO_COMPLETO
END
```

18) Se você executar a consulta abaixo:

```
SELECT CPF, [dbo].[EnderecoCompleto]([ENDERECO 1], CIDADE, ESTADO, CEP) AS END_COMPLETO
FROM [TABELA DE CLIENTES]
```

Terá a listagem dos clientes com o endereço completo.

19) Altere a função acima, o endereço não terá mais o "-" entre cada campo que compõe o endereço. Substitua por ",". Para isso, execute o comando `ALTER TABLE`, conforme mostrado abaixo:

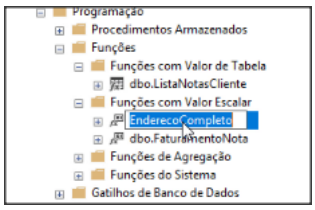
```
ALTER FUNCTION EnderecoCompleto (
    @ENDERECO VARCHAR(100),
    @CIDADE VARCHAR(50),
```

```
    @ESTADO VARCHAR(50),
    @CEP VARCHAR(20)
)
RETURNS VARCHAR(250)
AS
BEGIN
    DECLARE @ENDERECO_COMPLETO VARCHAR(250)
    SET @ENDERECO_COMPLETO = @ENDERECO + ', ' +
        @CIDADE + ', ' + @ESTADO + ', ' + @CEP
    RETURN @ENDERECO_COMPLETO
END
```

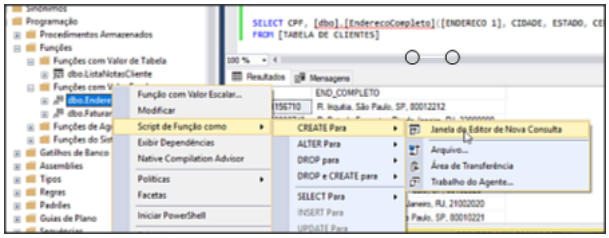
20) Executando novamente a consulta, você verá que o endereço está com o separador "," entre seus elementos:

```
SELECT CPF, [dbo].[EnderecoCompleto] (
    [ENDERECO 1],
    CIDADE,
    ESTADO,
    CEP
) AS END_COMPLETO
FROM [TABELA DE CLIENTES]
```

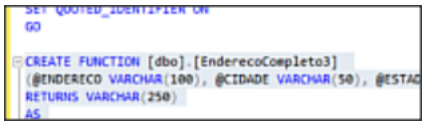
21) Exclua uma função. Se você for na lista de funções, terá:



22) Clique com o botão da direita do mouse sobre a função `EnderecoCompleto` e escolha a opção **Script de Função como --> CREATE Para --> Janela do Editor da Nova Consulta**:

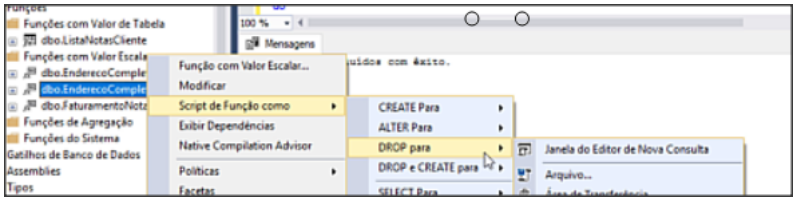


23) No script apresentado, modifique o nome da função para `EnderecoCompleto3` :



24) Execute o comando e uma nova função será criada.

25) Volte à lista das funções e, sobre o nome da função `EnderecoCompleto3` clique com o botão da direita do mouse e selecione **Script de Função como --> DROP Para --> Janela do Editor da Nova Consulta**:



26) Você verá o comando para excluir a função, que se chama `DROP FUNCTION`.

27) Você pode também usar o `IF` para testar se a função existe antes de criá-la. Em caso positivo, você pode apagá-la antes de sua criação:

```
IF OBJECT_ID ('EnderecoCompleto3', 'FN') IS NOT NULL
DROP FUNCTION [dbo].[EnderecoCompleto3]

CREATE FUNCTION [dbo].[EnderecoCompleto3] (
    @ENDERECO VARCHAR(100),
    @CIDADE VARCHAR(50),
    @ESTADO VARCHAR(50),
    @CEP VARCHAR(20)
)
RETURNS VARCHAR(250)
AS
BEGIN
    DECLARE @ENDERECO_COMPLETO VARCHAR(250)
    SET @ENDERECO_COMPLETO = @ENDERECO + ', ' +
        @CIDADE + ', ' + @ESTADO + ', ' + @CEP
    RETURN @ENDERECO_COMPLETO
END
```