**CURSO T-SQL – ALURA:**

**SQL Server: T-SQL com Microsoft SQL Server 2017**

<https://cursos.alura.com.br/course/tsql-com-sql-server-2017/task/41174>

**03. ESCREVENDO FUNÇÕES**

Nesta aula, aprendemos:

* Como funciona as funções no SQL Server;
* Como ter uma tabela como resultado de uma função;
* Como usar funções dentro de um *loop*;
* Como alterar e excluir 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**.

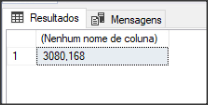
2) Se você executar o comando abaixo:

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:

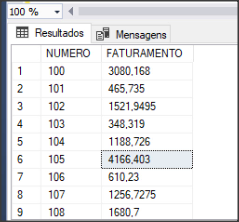
RETURN @FATURAMENTO

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

SELECT NUMERO,

[dbo].[FaturamentoNota](NUMERO) AS FATURAMENTO

FROM [NOTAS FISCAIS]



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

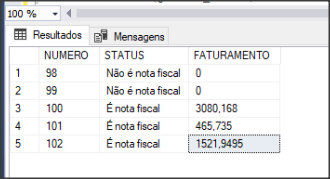
SELECT \* FROM @TABELA\_NUMEROS

11) Modifique os parâmetros iniciais do script:

SET @LIMITE\_MINIMO = 98

SET @LIMITE\_MAXIMO = 102

E executando-o, você terá:



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,

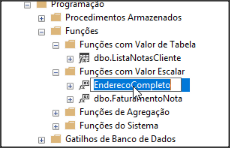
ESTADO,

CEP

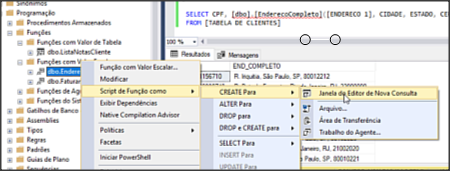
) AS END\_COMPLETO

FROM [TABELA DE CLIENTES]

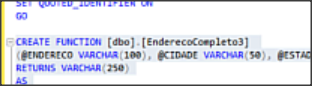
21) Exclua uma função. Se você for na lista de funções, terá:



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**:

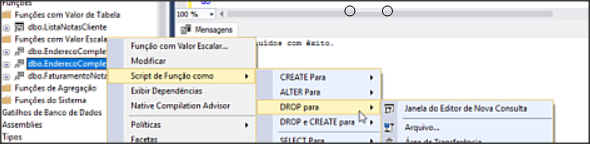


23) No script apresentado, modifique o nome da função para **EnderecoCompleto3**:



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