**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/41179**](https://cursos.alura.com.br/course/tsql-com-sql-server-2017/task/41179)

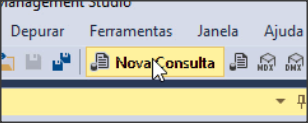
**02. ELEMENTOS DO T-SQL**

Nesta aula, aprendemos:

* Algumas estruturas principais do T-SQL, como a declaração de variáveis;
* Como atribuir valores a estas variáveis;
* O uso de desvios de fluxo (**IF**);
* Como usar o *loop*;
* As tabelas temporárias;
* Como depurar o código.

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

1) Abra o **SQL Server Management Studio 2017**, conecte-se ao seu servidor, e clique sobre **Nova Consulta**:



2) Você pode declarar variáveis no T-SQL:

DECLARE @IDADE INT

DECLARE @NOME VARCHAR(200)

DECLARE @DATA DATE

3) Toda esta declaração pode ser feita na mesma linha:

DECLARE @IDADE2 INT, @NOME2 VARCHAR(200), @DATA2 DATE

4) Você pode associar valores à variável:

SET @IDADE = 20

SET @NOME = 'JOÃO DA SILVA'

SET @DATA = '20180517'

5) Se você executar o comando **PRINT**, você pode observar os valores das variáveis:

SET @IDADE = 20

PRINT @IDADE

SET @NOME = 'JOÃO DA SILVA'

PRINT @NOME

SET @DATA = '20180517'

PRINT @DATA

6) Você pode usar o conteúdo da variável e usá-la em comandos, por exemplo, de consultas na base SQL Server:

DECLARE @CPF VARCHAR(12)

SET @CPF = '2600586709'

SELECT NOME, [DATA DE NASCIMENTO],

IDADE FROM [TABELA DE CLIENTES]

WHERE CPF = @CPF

7) Você pode também atribuir valores às variáveis, através do comando SQL:

DECLARE @IDADE2 INT, @NOME2 VARCHAR(200), @DATA2 DATE

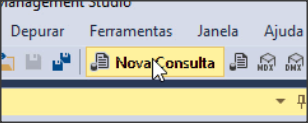
SELECT @NOME2 = NOME, @DATA2 = [DATA DE NASCIMENTO],

@IDADE2 = IDADE

FROM [TABELA DE CLIENTES]

WHERE CPF = @CPF

8) Novamente no **SQL Server Management Studio 2017**, clique sobre **Nova Consulta**:



9) Certifique-se que você esteja acessando a base **SUCOS\_VENDAS**.

10) Verifique como usar o controle de fluxo para testar se uma tabela existe. Se você executar o comando:

CREATE TABLE TABELA\_TESTE (ID VARCHAR(10))

Você irá criar uma tabela.

11) Porém, se você executar o comando acima novamente, ocorrerá um erro, pois você não pode criar uma tabela que já existe. Para isso, você deve executar o comando **DROP** antes de recriar a tabela:

DROP TABLE TABELA\_TESTE

CREATE TABLE TABELA\_TESTE (ID VARCHAR(10))

12) Se você executar estes comandos sempre nesta ordem, nenhum problema acontecerá. Mas você pode usar o comando **IF** do T-SQL para testar se a tabela existe ou não. Então, antes de apagar a tabela, teste se ela existe e, antes de criá-la, teste se ela não existe. Para isso, você pode executar os comandos abaixo:

IF OBJECT\_ID('TABELA\_TESTE', 'U') IS NOT NULL

DROP TABLE TABELA\_TESTE

IF OBJECT\_ID('TABELA\_TESTE', 'U') IS NULL

CREATE TABLE TABELA\_TESTE (ID VARCHAR(10))

Assim você garante que estes comandos sempre funcionarão, mesmo que executados independentemente.

13) Crie uma nova consulta. Relembrando algumas funções vistas em cursos anteriores, se você executar o comando abaixo:

SELECT GETDATE()

Você irá obter a data do sistema.

14) Já o comando abaixo:

SELECT DATENAME(WEEKDAY, DATEADD(DAY, 6, GETDATE()))

Te retorna o nome do dia da semana 6 dias após o dia de hoje.

15) Inclua comandos que irão testar o dia selecionado e verificar se ele é um dia de semana ou se é um dia do fim de semana. Se você digitar os comandos abaixo:

DECLARE @DIA\_SEMANA VARCHAR(20)

DECLARE @NUMERO\_DIAS INT

SET @NUMERO\_DIAS = 9

SET @DIA\_SEMANA =

DATENAME(WEEKDAY, DATEADD(DAY, @NUMERO\_DIAS, GETDATE()))

PRINT @DIA\_SEMANA

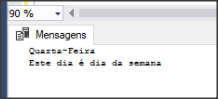
IF @DIA\_SEMANA = 'Domingo' OR @DIA\_SEMANA = 'Sábado'

PRINT 'Este dia é fim de semana'

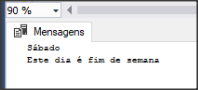
ELSE

PRINT 'Este dia é dia da semana'

Você irá obter este resultado, por exemplo:



Ou:



16) Crie uma nova consulta. Obtenha a soma dos limites de créditos de clientes de um determinado bairro e compare-a com um valor máximo previamente determinado. Se a soma dos limites for maior que este valor, haverá uma resposta, se não, haverá outra:

DECLARE @LIMITE\_MAXIMO FLOAT

DECLARE @BAIRRO VARCHAR(20)

SET @BAIRRO = 'Água Santa'

SET @LIMITE\_MAXIMO = 50000

IF @LIMITE\_MAXIMO <= (SELECT SUM([LIMITE DE CREDITO])

FROM [TABELA DE CLIENTES] WHERE BAIRRO = @BAIRRO)

BEGIN

PRINT 'Valor estourou. Não é possível abrir novos créditos'

END

ELSE

BEGIN

PRINT 'Valor não estourou. É possível abrir novos créditos'

END

Você terá o seguinte resultado:

Ou:

17) Crie uma nova consulta. Gere um *loop* para exibir um contador de números entre 1 e um valor máximo. Para isso, declare algumas variáveis e inclua os parâmetros do procedimento:

DECLARE @LIMITE\_MINIMO INT, @LIMITE\_MAXIMO INT

SET @LIMITE\_MINIMO = 1

SET @LIMITE\_MAXIMO = 10

18) Em seguida, use a estrutura de **WHILE** para percorrer os valores de 1 até o valor máximo e exibir o resultado na saída do SQL Server:

WHILE @LIMITE\_MINIMO <= @LIMITE\_MAXIMO

BEGIN

PRINT @LIMITE\_MINIMO

SET @LIMITE\_MINIMO = @LIMITE\_MINIMO + 1

END

Você terá o seguinte resultado:



19) Você pode incluir o comando **BREAK** para parar o *loop* em um determinado valor. Para isso, adicione mais uma variável, que determinará o ponto de parada, na declaração das variáveis:

DECLARE @LIMITE\_MINIMO INT,

@LIMITE\_MAXIMO INT,

@LIMITE\_BREAK INT

20) E inicialize-a:

SET @LIMITE\_MINIMO = 1

SET @LIMITE\_MAXIMO = 10

SET @LIMITE\_BREAK = 12

21) Por fim, determine o ponto de parada dentro do *loop*:

WHILE @LIMITE\_MINIMO <= @LIMITE\_MAXIMO

BEGIN

PRINT @LIMITE\_MINIMO

SET @LIMITE\_MINIMO = @LIMITE\_MINIMO + 1

IF @LIMITE\_MINIMO = @LIMITE\_BREAK

BEGIN

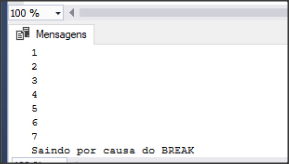
PRINT 'Saindo por causa do BREAK'

BREAK

END

END

22) Você terá como resposta:



23) Crie uma nova consulta onde você irá percorrer um limite mínimo e máximo de números e, para cada um deles, verifique se é ou não o número de alguma nota fiscal. Se for, escreva na tabela o número e um *label* dizendo que é uma nota. Se não, escreva também o número, mas o *label* dirá que ele não é uma nota:

IF OBJECT\_ID ('TABELA DE NUMEROS','U') IS NOT NULL

DROP TABLE [TABELA DE NUMEROS]

CREATE TABLE [TABELA DE NUMEROS] (

[NUMERO] INT,

[STATUS] VARCHAR(200)

)

DECLARE @LIMITE\_MINIMO INT,

@LIMITE\_MAXIMO INT,

@CONTADOR\_NOTAS INT

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

INSERT INTO [TABELA DE NUMEROS] ([NUMERO], [STATUS])

VALUES (@LIMITE\_MINIMO, 'É nota fiscal')

ELSE

INSERT INTO [TABELA DE NUMEROS] ([NUMERO], [STATUS])

VALUES (@LIMITE\_MINIMO, 'Não é nota fiscal')

SET @LIMITE\_MINIMO = @LIMITE\_MINIMO + 1

END

SELECT \* FROM [TABELA DE NUMEROS]

Com a seguinte saída:

24) Crie uma nova consulta para a base **SUCOS\_VENDAS**, baseada na consulta anterior, e insira as tabelas temporárias:

DECLARE @LIMITE\_MINIMO INT,

@LIMITE\_MAXIMO INT,

@CONTADOR\_NOTAS INT

DECLARE @TABELA\_NUMEROS TABLE (

[NUMERO] INT,

[STATUS] VARCHAR(200)

)

CREATE TABLE #TABELA\_NUMEROS2 (

[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')

INSERT INTO #TABELA\_NUMEROS2 ([NUMERO], [STATUS])

VALUES (@LIMITE\_MINIMO, 'É nota fiscal')

END

ELSE

BEGIN

INSERT INTO @TABELA\_NUMEROS ([NUMERO], [STATUS])

VALUES (@LIMITE\_MINIMO, 'Não é nota fiscal')

INSERT INTO #TABELA\_NUMEROS2 ([NUMERO], [STATUS])

VALUES (@LIMITE\_MINIMO, 'Não é nota fiscal')

END

SET @LIMITE\_MINIMO = @LIMITE\_MINIMO + 1

END

SELECT \* FROM @TABELA\_NUMEROS

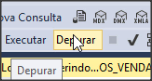
SELECT \* FROM #TABELA\_NUMEROS2

25) Pra facilitar a depuração, altere os parâmetros para os valores abaixo:

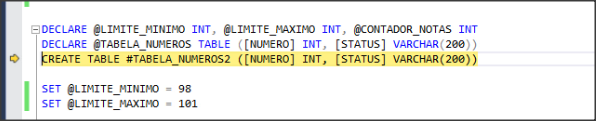
SET @LIMITE\_MINIMO = 98

SET @LIMITE\_MAXIMO = 101

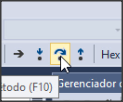
26) Clique no botão **Depurar**:



27) Você verá o início da depuração com a linha atual selecionada com amarelo:

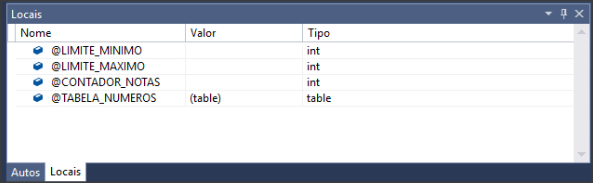


28) Se você clicar no botão destacado abaixo (**Pular Método**):

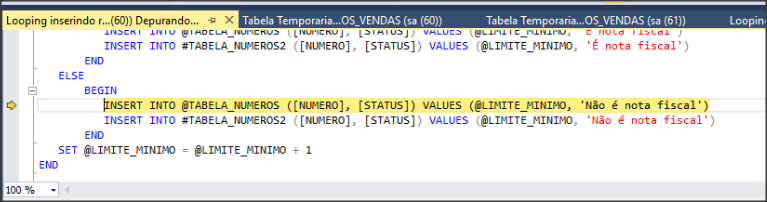


Fará com que o depurador passa para a linha seguinte.

29) Abaixo você pode ver o valor das variáveis a cada momento:



30) Clicando no botão **Pular Método**, você pode ir avançando passo a passo no código fonte da rotina:



31) Acompanhe a execução dos comandos passo a passo até o seu final.