15

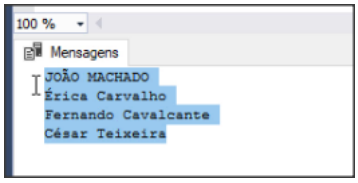
Mãos na massa: Cursor

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 acessando a base **SUCOS_VENDAS** , e digite os comandos abaixo para mostrar o uso do *cursor*:

```
DECLARE @NOME VARCHAR(200)
DECLARE CURSOR1 CURSOR FOR SELECT TOP 4 NOME
FROM [TABELA DE CLIENTES]
OPEN CURSOR1
FETCH NEXT FROM CURSOR1 INTO @NOME
WHILE @@FETCH_STATUS = 0
BEGIN
    PRINT @NOME
    FETCH NEXT FROM CURSOR1 INTO @NOME
END
CLOSE CURSOR1
DEALLOCATE CURSOR1
```

2) Se você executar os comandos acima, terá a exibição dos nomes 4 primeiros clientes:



3) Você pode associar mais de um campo a um *cursor*. Veja um exemplo abaixo, crie uma nova consulta no **SQL Server Management Studio** associada à base de dados **SUCOS_VENDAS** , com o código abaixo:

```
DECLARE @NOME VARCHAR(200)
DECLARE @ENDERECO VARCHAR(MAX)
DECLARE CURSOR1 CURSOR FOR
SELECT NOME, ([ENDERECO 1] + ' - ' + BAIRRO + ' - ' +
    CIDADE + ' - ' + ESTADO + ' - ' + CEP) ENDCOMPLETO
FROM [TABELA DE CLIENTES]
OPEN CURSOR1
FETCH NEXT FROM CURSOR1 INTO @NOME, @ENDERECO
WHILE @@FETCH_STATUS = 0
BEGIN
    PRINT @NOME + ' Endereço: ' + @ENDERECO
    FETCH NEXT FROM CURSOR1 INTO @NOME, @ENDERECO
END
```

4) Execute o código e veja que o resultado irá apresentar os dados concatenados originários do *cursor*:

	NOME	ENDCOMPLETO
1	JOÃO MACHADO	RUA PROJETADA A - MADUREIRA - Rio de Janeiro - ...
2	Érica Carvalho	R. Iriquitia - Jardins - São Paulo - SP - 80012212
3	Fernando Cavalcante	R. Dois de Fevereiro - Água Santa - Rio de Janeiro - RJ ...
4	César Teixeira	Rua Conde de Bonfim - Tijuca - Rio de Janeiro - RJ - 22...
5	Marcos Nougueira	Av. Pastor Martin Luther King Junior - Inhauma - Rio de ...
6	Eduardo Jorge	R. Volta Grande - Tijuca - Rio de Janeiro - RJ - 22012002
7	Abel Silva	Rua Humaitá - Humaitá - Rio de Janeiro - RJ - 22000212
8	Petra Oliveira	R. Benício de Abreu - Lapa - São Paulo - SP - 88192029
9	Paulo César Mattos	Rua Hélio Beltrão - Tijuca - Rio de Janeiro - RJ - 210020...

5) Crie um código usando T-SQL que tem como objetivo criar uma venda fictícia. O processo a ser construído irá selecionar um cliente, um produto e um vendedor aleatoriamente do cadastro existente, e também buscar um número de itens de nota fiscais aleatórios, criar quantidades aleatórias e incluir na tabela de notas fiscais e de itens de notas. Logo, crie uma nova consulta no **SQL Server Management Studio**, associada à base `SUCOS_VENDAS` e, o primeiro passo, será criar uma função que gere números aleatórios entre dois determinados números.

6) Se você executar o comando abaixo:

```
SELECT RAND()
```

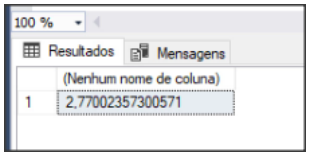
Você irá selecionar um número aleatório entre 0 e 1.

7) Para gerar um número aleatório entre X e Y, basta fazer a seguinte fórmula matemática:

```
((X - Y - 1) * RAND() + Y)
```

Para um número entre 10 e 1, então basta executar:

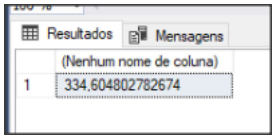
```
SELECT ((10 - 1 - 1) * RAND() + 1)
```



	(Nenhum nome de coluna)
1	2,77002357300571

8) Para um número entre 500 e 100, faça:

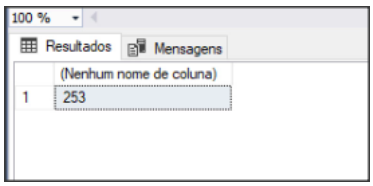
```
SELECT ((500 - 100 - 1) * RAND() + 100)
```



	(Nenhum nome de coluna)
1	334,604802782674

9) Para arrendondá-lo, ou seja, sem casas decimais, use a função `ROUND` :

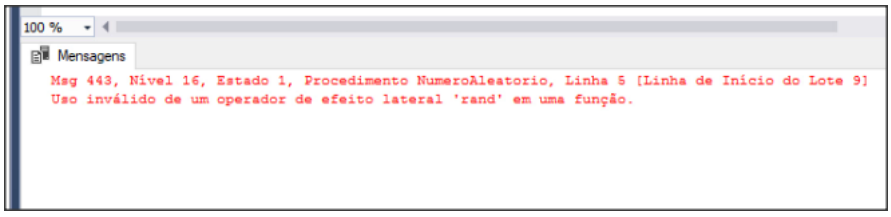
```
SELECT ROUND(((500 - 100 - 1) * RAND() + 100), 0)
```



10) Agora, crie uma função que obtém os números aleatórios entre dois valores. Os parâmetros desta função serão o valor inicial e final do intervalo:

```
CREATE FUNCTION NumeroAleatorio (
    @VAL_INIC INT,
    @VAL_FINAL INT
) RETURNS INT
AS
BEGIN
DECLARE @ALEATORIO INT
SET @ALEATORIO =
    ROUND(((@VAL_FINAL - @VAL_INIC - 1) *
        RAND() + @VAL_INIC), 0)
RETURN @ALEATORIO
END
```

11) Ao executar os comandos acima para a criação da função, ocorrerá um erro, como mostrado abaixo:



`RAND()` não pode ser usada dentro da criação de uma função **UDF**.

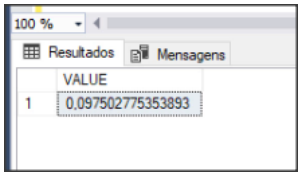
12) Para contornar este problema, crie uma `VIEW` , como mostrado abaixo:

```
CREATE VIEW VW_ALEATORIO AS SELECT RAND() AS VALUE
```

13) Executando a criação da `VIEW` :

```
SELECT * FROM VW_ALEATORIO
```

Você irá obter o mesmo resultado que a execução da função `RAND()` :

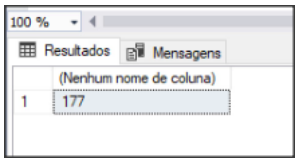


14) Agora você pode editar a criação da função, usando, em vez de `RAND` , a `VIEW` :

```
CREATE FUNCTION NumeroAleatorio (
    @VAL_INIC INT,
    @VAL_FINAL INT
) RETURNS INT
AS
BEGIN
DECLARE @ALEATORIO INT
DECLARE @ALEATORIO_FLOAT FLOAT
SELECT @ALEATORIO_FLOAT = VALUE FROM VW_ALEATORIO
SET @ALEATORIO =
    ROUND(((@VAL_FINAL - @VAL_INIC - 1) *
        @ALEATORIO_FLOAT + @VAL_INIC), 0)
RETURN @ALEATORIO
END
```

15) Após a criação da função, você pode usá-la, executando, por exemplo, um simples **SELECT** chamando a função **UDF**:

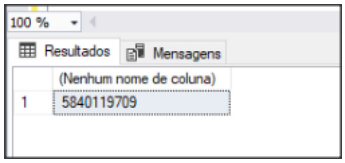
```
SELECT [dbo].[NumeroAleatorio](1,100)
```



16) Agora use a função criada acima para obter um cliente aleatório a partir da lista de clientes contida na tabela. Obtenha um número aleatório entre 1 e o número total de clientes. Então percorra a tabela através de um *cursor* para buscar o cliente na posição correspondente ao número aleatório:

```
DECLARE @CLIENTE_ALEATORIO VARCHAR(12)
DECLARE @VAL_INICIAL INT
DECLARE @VAL_FINAL INT
DECLARE @ALEATORIO INT
DECLARE @CONTADOR INT

SET @CONTADOR = 1
SET @VAL_INICIAL = 1
SELECT @VAL_FINAL = COUNT(*) FROM [TABELA DE CLIENTES]
SET @ALEATORIO =
    [dbo].[NumeroAleatorio](@VAL_INICIAL, @VAL_FINAL)
DECLARE CURSOR1 CURSOR FOR SELECT CPF
    FROM [TABELA DE CLIENTES]
OPEN CURSOR1
FETCH NEXT FROM CURSOR1 INTO @CLIENTE_ALEATORIO
WHILE @CONTADOR < @ALEATORIO
BEGIN
    FETCH NEXT FROM CURSOR1 INTO @CLIENTE_ALEATORIO
    SET @CONTADOR = @CONTADOR + 1
END
CLOSE CURSOR1
DEALLOCATE CURSOR1
SELECT @CLIENTE_ALEATORIO
```

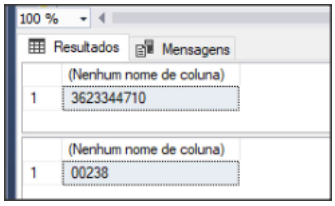
Executando, você obtém um cliente aleatório:



17) Você pode repetir a lógica acima para obter um vendedor aleatório:

```
DECLARE @CLIENTE_ALEATORIO VARCHAR(12)
DECLARE @VENDEDOR_ALEATORIO VARCHAR(12)
DECLARE @VAL_INICIAL INT
DECLARE @VAL_FINAL INT
DECLARE @ALEATORIO INT
DECLARE @CONTADOR INT

SET @CONTADOR = 1
SET @VAL_INICIAL = 1
SELECT @VAL_FINAL = COUNT(*) FROM [TABELA DE CLIENTES]
SET @ALEATORIO =
    [dbo].[NumeroAleatorio](@VAL_INICIAL, @VAL_FINAL)
DECLARE CURSOR1 CURSOR FOR SELECT CPF
    FROM [TABELA DE CLIENTES]
OPEN CURSOR1
FETCH NEXT FROM CURSOR1 INTO @CLIENTE_ALEATORIO
WHILE @CONTADOR < @ALEATORIO
BEGIN
    FETCH NEXT FROM CURSOR1 INTO @CLIENTE_ALEATORIO
    SET @CONTADOR = @CONTADOR + 1
END
CLOSE CURSOR1
DEALLOCATE CURSOR1
SELECT @CLIENTE_ALEATORIO

SET @CONTADOR = 1
SET @VAL_INICIAL = 1
SELECT @VAL_FINAL = COUNT(*) FROM [TABELA DE VENDEDORES]
SET @ALEATORIO =
    [dbo].[NumeroAleatorio](@VAL_INICIAL, @VAL_FINAL)
DECLARE CURSOR1 CURSOR FOR SELECT MATRICULA
    FROM [TABELA DE VENDEDORES]
OPEN CURSOR1
FETCH NEXT FROM CURSOR1 INTO @VENDEDOR_ALEATORIO
WHILE @CONTADOR < @ALEATORIO
BEGIN
    FETCH NEXT FROM CURSOR1 INTO @VENDEDOR_ALEATORIO
    SET @CONTADOR = @CONTADOR + 1
END
CLOSE CURSOR1
DEALLOCATE CURSOR1
SELECT @VENDEDOR_ALEATORIO
```

Executando, você terá um cliente e vendedor aleatório:



18) Junte o processo de obter a entidade (cliente, produto ou vendedor) aleatória em uma única função. A única diferença é que a tabela que você tem que analisar (para obter o número máximo de membros) e que tem que percorrer no cursor será uma para cada tipo de entidade. Inicie a função digitando sua declaração e as variáveis auxiliares:

```
CREATE FUNCTION EntidadeAleatoria (@TIPO VARCHAR(12))
    RETURNS VARCHAR(20)
AS
BEGIN
DECLARE @ENTIDADE_ALEATORIO VARCHAR(12)
DECLARE @TABELA TABLE (CODIGO VARCHAR(20))
DECLARE @VAL_INICIAL INT
DECLARE @VAL_FINAL INT
DECLARE @ALEATORIO INT
DECLARE @CONTADOR INT
```

O parâmetro `@TIPO` da função determinará que tipo de entidade você irá analisar (cliente, produto ou vendedor). A tabela `@TABELA` vai obter os dados da entidade, dependendo do valor de `@TIPO` .

19) Você irá determinar com que tabela irá trabalhar baseado no valor de `@TIPO` . Logo, digite os comandos abaixo:

```
IF @TIPO = 'CLIENTE'
BEGIN
    INSERT INTO @TABELA (CODIGO) SELECT CPF AS CODIGO
        FROM [TABELA DE CLIENTES]
END
IF @TIPO = 'VENDEDOR'
BEGIN
    INSERT INTO @TABELA (CODIGO) SELECT MATRICULA
        FROM [TABELA DE VENDEDORES]
END
IF @TIPO = 'PRODUTO'
BEGIN
    INSERT INTO @TABELA (CODIGO) SELECT [CODIGO DO PRODUTO]
        FROM [TABELA DE PRODUTOS]
END
```

Neste ponto, a variável `@TABELA` terá, ou a listagem de clientes, ou de vendedores, ou de produtos.

20) Agora, obtenha um valor aleatório sobre `@TABELA` , e percorra a mesma através de um *cursor*:

```
SET @CONTADOR = 1
SET @VAL_INICIAL = 1
SELECT @VAL_FINAL = COUNT(*) FROM @TABELA
SET @ALEATORIO =
    [dbo].[NumeroAleatorio](@VAL_INICIAL, @VAL_FINAL)
DECLARE CURSOR1 CURSOR FOR SELECT CODIGO FROM @TABELA
```

```
OPEN CURSOR1
FETCH NEXT FROM CURSOR1 INTO @ENTIDADE_ALEATORIO
WHILE @CONTADOR < @ALEATORIO
BEGIN
    FETCH NEXT FROM CURSOR1 INTO @ENTIDADE_ALEATORIO
    SET @CONTADOR = @CONTADOR + 1
END
CLOSE CURSOR1
DEALLOCATE CURSOR1
```

21) Feche a função com o `RETURN` e, então , você terá a função completa:

```
CREATE FUNCTION EntidadeAleatoria (@TIPO VARCHAR(12))
    RETURNS VARCHAR(20)
AS
BEGIN
DECLARE @ENTIDADE_ALEATORIO VARCHAR(12)
DECLARE @TABELA TABLE (CODIGO VARCHAR(20))
DECLARE @VAL_INICIAL INT
DECLARE @VAL_FINAL INT
DECLARE @ALEATORIO INT
DECLARE @CONTADOR INT

IF @TIPO = 'CLIENTE'
BEGIN
    INSERT INTO @TABELA (CODIGO) SELECT CPF AS CODIGO
        FROM [TABELA DE CLIENTES]
END
IF @TIPO = 'VENDEDOR'
BEGIN
    INSERT INTO @TABELA (CODIGO) SELECT MATRICULA
        FROM [TABELA DE VENDEDORES]
END
IF @TIPO = 'PRODUTO'
BEGIN
    INSERT INTO @TABELA (CODIGO) SELECT [CODIGO DO PRODUTO]
        FROM [TABELA DE PRODUTOS]
END

SET @CONTADOR = 1
SET @VAL_INICIAL = 1
SELECT @VAL_FINAL = COUNT(*) FROM @TABELA
SET @ALEATORIO = [dbo].[NumeroAleatorio](@VAL_INICIAL, @VAL_FINAL)
DECLARE CURSOR1 CURSOR FOR SELECT CODIGO FROM @TABELA
OPEN CURSOR1
FETCH NEXT FROM CURSOR1 INTO @ENTIDADE_ALEATORIO
WHILE @CONTADOR < @ALEATORIO
BEGIN

    FETCH NEXT FROM CURSOR1 INTO @ENTIDADE_ALEATORIO
    SET @CONTADOR = @CONTADOR + 1
END
CLOSE CURSOR1
DEALLOCATE CURSOR1
RETURN @ENTIDADE_ALEATORIO
END
```

22) Execute a criação da função e, agora, você pode usá-la, por exemplo, em um `SELECT`, como mostrado abaixo:

```
SELECT [dbo].[EntidadeAleatoria]('CLIENTE'),
    [dbo].[EntidadeAleatoria]('PRODUTO'),
    [dbo].[EntidadeAleatoria]('VENDEDOR')
```

23) Agora você irá determinar outros dados da nota fiscal. Estes dados serão:

- Número da nota - Será um sequencial baseado no maior número existente;
- Data da nota - Será um parâmetro fixo;
- Imposto - Valor fixo;
- Número de itens da nota - Será um número aleatório entre 2 e 10;
- Quantidade - Também um valor aleatório, mas entre 5 e 100.
- Preço - Será o preço obtido da tabela de produtos.

24) Logo, digite os comandos abaixo:

```
DECLARE @CLIENTE VARCHAR(12)
DECLARE @VENDEDOR VARCHAR(12)
DECLARE @PRODUTO VARCHAR(12)
DECLARE @DATA DATE
DECLARE @NUMERO INT
DECLARE @IMPOSTO FLOAT
DECLARE @NUM_ITENS INT
DECLARE @CONTADOR INT
DECLARE @QUANTIDADE INT
DECLARE @PRECO FLOAT

SET @DATA = '20180521'
SET @CLIENTE = [dbo].[EntidadeAleatoria]('CLIENTE')
SET @VENDEDOR = [dbo].[EntidadeAleatoria]('VENDEDOR')
SELECT @NUMERO = MAX(NUMERO) + 1 FROM [NOTAS FISCAIS]
SET @IMPOSTO = 0.18
SET @CONTADOR = 1
SET @NUM_ITENS = [dbo].[NumeroAleatorio](2, 10)
```

Você declara as variáveis, obtêm o cliente e vendedor aleatórios, e os dados associados ao cabeçalho da nota fiscal.

25) Insira os dados na tabela de notas fiscais:

```
INSERT INTO [NOTAS FISCAIS] (CPF, MATRICULA, DATA, NUMERO, IMPOSTO)
    VALUES (@CLIENTE, @VENDEDOR, @DATA, @NUMERO, @IMPOSTO)
```

26) Com o número de itens obtido, faça um *loop* para determinar os dados de cada item da nota (produto, quantidade e preço):

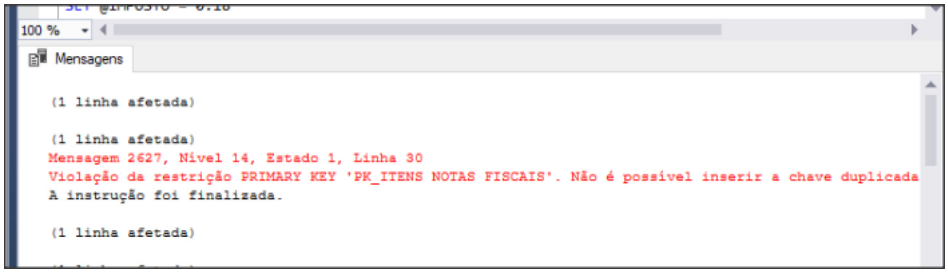
```
WHILE @CONTADOR <= @NUM_ITENS
BEGIN
    SET @PRODUTO = [dbo].[EntidadeAleatoria]('PRODUTO')
```

```
        SET @QUANTIDADE = [dbo].[NumeroAleatorio](5, 100)
        SELECT @PRECO = [PREÇO DE LISTA] FROM [TABELA DE PRODUTOS]
            WHERE [CODIGO DO PRODUTO] = @PRODUTO

        INSERT INTO [ITENS NOTAS FISCAIS]
            (NUMERO, [CODIGO DO PRODUTO], QUANTIDADE, PREÇO)
            VALUES (@NUMERO, @PRODUTO, @QUANTIDADE, @PRECO)

        SET @CONTADOR = @CONTADOR + 1
    END
```

27) Por fim, você terá o script completo. Execute-o:

```
DECLARE @CLIENTE VARCHAR(12)
DECLARE @VENDEDOR VARCHAR(12)
DECLARE @PRODUTO VARCHAR(12)
DECLARE @DATA DATE
DECLARE @NUMERO INT
DECLARE @IMPOSTO FLOAT
DECLARE @NUM_ITENS INT
DECLARE @CONTADOR INT
DECLARE @QUANTIDADE INT
DECLARE @PRECO FLOAT

SET @DATA = '20180521'
SET @CLIENTE = [dbo].[EntidadeAleatoria]('CLIENTE')
SET @VENDEDOR = [dbo].[EntidadeAleatoria]('VENDEDOR')
SELECT @NUMERO = MAX(NUMERO) + 1 FROM [NOTAS FISCAIS]
SET @IMPOSTO = 0.18
SET @CONTADOR = 1
SET @NUM_ITENS = [dbo].[NumeroAleatorio](2, 10)

INSERT INTO [NOTAS FISCAIS] (CPF, MATRICULA, DATA, NUMERO, IMPOSTO)
    VALUES (@CLIENTE, @VENDEDOR, @DATA, @NUMERO, @IMPOSTO)

WHILE @CONTADOR <= @NUM_ITENS
BEGIN
    SET @PRODUTO = [dbo].[EntidadeAleatoria]('PRODUTO')
    SET @QUANTIDADE = [dbo].[NumeroAleatorio](5, 100)
    SELECT @PRECO = [PREÇO DE LISTA] FROM [TABELA DE PRODUTOS]
        WHERE [CODIGO DO PRODUTO] = @PRODUTO

    INSERT INTO [ITENS NOTAS FISCAIS]
        (NUMERO, [CODIGO DO PRODUTO], QUANTIDADE, PREÇO)
        VALUES (@NUMERO, @PRODUTO, @QUANTIDADE, @PRECO)

    SET @CONTADOR = @CONTADOR + 1
END
```

28) Ao executar a rotina várias vezes, ocorre um erro de PK:



Ele ocorre porque, dependendo do número de itens aleatórios obtidos, você pode tentar incluir um item com o mesmo número e mesmo produto na tabela de itens de nota. Logo, você terá que testar se o produto aleatório já existe na tabela com o mesmo número de nota. Se existir, você busca outro produto.

29) Crie duas variáveis. Uma que será auxiliar para testar o produto antes de incluí-lo na tabela e outra será do tipo tabela, onde você guardará os produtos incluídos:

```
DECLARE @LISTAPRODUTOS TABLE (PRODUTO VARCHAR(20))
DECLARE @AUXPRODUTO INT
```

30) Dentro do *loop*, teste se o produto obtido de forma aleatória já existe na tabela de produtos temporária (`@LISTAPRODUTOS`):

```
SELECT @AUXPRODUTO = COUNT(*) FROM @LISTAPRODUTOS
    WHERE PRODUTO = @PRODUTO
```

31) Teste o valor de `@AUXPRODUTO` , se ele for 0, significa que o produto ainda não foi incluído para esta nota fiscal, então inclua-o na tabela de itens. Se for diferente de 0, não faça nada. Para isso, acrescente um `IF` ao script:

```
IF @AUXPRODUTO = 0
BEGIN
    SET @QUANTIDADE = [dbo].[NumeroAleatorio](5, 100)
    SELECT @PRECO = [PREÇO DE LISTA] FROM [TABELA DE PRODUTOS]
        WHERE [CODIGO DO PRODUTO] = @PRODUTO
    INSERT INTO [ITENS NOTAS FISCAIS]
        (NUMERO, [CODIGO DO PRODUTO], QUANTIDADE, PREÇO)
    VALUES (@NUMERO, @PRODUTO, @QUANTIDADE, @PRECO)
    SET @CONTADOR = @CONTADOR + 1
END
```

32) Inclua o produto atual na `@LISTAPRODUTOS` para os testes seguintes:

```
INSERT INTO @LISTAPRODUTOS (PRODUTO) VALUES (@PRODUTO)
```

33) O script ficará assim:

```
DECLARE @CLIENTE VARCHAR(12)
DECLARE @VENDEDOR VARCHAR(12)
DECLARE @PRODUTO VARCHAR(12)
DECLARE @DATA DATE
DECLARE @NUMERO INT
DECLARE @IMPOSTO FLOAT
DECLARE @NUM_ITENS INT
```

```
DECLARE @NUM_ITENS INT
DECLARE @CONTADOR INT
DECLARE @QUANTIDADE INT
DECLARE @PRECO FLOAT
DECLARE @LISTAPRODUTOS TABLE (PRODUTO VARCHAR(20))
DECLARE @AUXPRODUTO INT

SET @DATA = '20180521'
SET @CLIENTE = [dbo].[EntidadeAleatoria]('CLIENTE')
SET @VENDEDOR = [dbo].[EntidadeAleatoria]('VENDEDOR')
SELECT @NUMERO = MAX(NUMERO) + 1 FROM [NOTAS FISCAIS]
SET @IMPOSTO = 0.18
SET @CONTADOR = 1
SET @NUM_ITENS = [dbo].[NumeroAleatorio](2, 10)

INSERT INTO [NOTAS FISCAIS] (CPF, MATRICULA, DATA, NUMERO, IMPOSTO)
    VALUES (@CLIENTE, @VENDEDOR, @DATA, @NUMERO, @IMPOSTO)

WHILE @CONTADOR <= @NUM_ITENS
BEGIN
    SET @PRODUTO = [dbo].[EntidadeAleatoria]('PRODUTO')
    SELECT @AUXPRODUTO = COUNT(*) FROM @LISTAPRODUTOS
        WHERE PRODUTO = @PRODUTO
    IF @AUXPRODUTO = 0
    BEGIN
        SET @QUANTIDADE = [dbo].[NumeroAleatorio](5, 100)
        SELECT @PRECO = [PREÇO DE LISTA] FROM [TABELA DE PRODUTOS]
            WHERE [CODIGO DO PRODUTO] = @PRODUTO
        INSERT INTO [ITENS NOTAS FISCAIS]
            (NUMERO, [CODIGO DO PRODUTO], QUANTIDADE, PREÇO)
            VALUES (@NUMERO, @PRODUTO, @QUANTIDADE, @PRECO)
        SET @CONTADOR = @CONTADOR + 1
    END
    INSERT INTO @LISTAPRODUTOS (PRODUTO) VALUES (@PRODUTO)
END
```