12

Consolidando o seu conhecimento

Chegou a hora de você seguir todos os passos realizados por mim durante esta aula. Caso já tenha feito, excelente. Se ainda não, é importante que você execute o que foi visto nos vídeos para poder continuar com os próximos cursos que tenham este como pré-requisito.

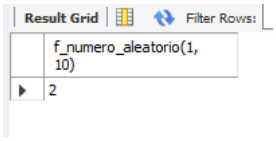
1) Vamos colocar nossos conhecimentos em prática. Iremos criar uma SP que irá criar uma venda aleatória.

2) Iniciamos com uma função que retorna um número aleatório entre dois valores. Para isso crie a função abaixo:

```
USE sucos_vendas;

DROP function IF EXISTS f_numero_aleatorio;


DELIMITER $$

USE sucos_vendas$$

CREATE FUNCTION f_numero_aleatorio(min INT, max INT) RETURNS int(11)

BEGIN

   DECLARE vRetorno INT;

   SELECT  FLOOR((RAND() * (max-min+1)) + min) INTO vRetorno;

RETURN vRetorno;

END$$


DELIMITER ;
```

3) Teste a função várias vezes para verificar que os números aleatórios são criados:

```
SELECT f_numero_aleatorio(1, 10);
```



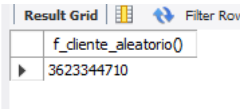
4) Usaremos a função abaixo para escolher um cliente de forma aleatória da tabela de clientes. Iremos escolher uma posição entre 1 e o número de registros da tabela e retornar o cliente escolhido conforme o número aleatório determinado. Digite e execute:

```
USE sucos_vendas;

DROP function IF EXISTS f_cliente_aleatorio;
```

```
DELIMITER $$

USE sucos_vendas$$

CREATE  FUNCTION f_cliente_aleatorio() RETURNS varchar(11) CHARSET utf8mb4

BEGIN

    DECLARE vRetorno VARCHAR(11);

    DECLARE num_max_tabela INT;

    DECLARE num_aleatorio INT;

    SELECT COUNT(*) INTO num_max_tabela FROM tabela_de_clientes;

    SET num_aleatorio = f_numero_aleatorio(1, num_max_tabela);

    SET num_aleatorio = num_aleatorio - 1;

    SELECT CPF INTO vRetorno FROM tabela_de_clientes

    LIMIT num_aleatorio, 1;

RETURN vRetorno;

END$$


DELIMITER ;
```

5) Teste a função para determinar um cliente aleatoriamente:

```
SELECT f_cliente_aleatorio();
```



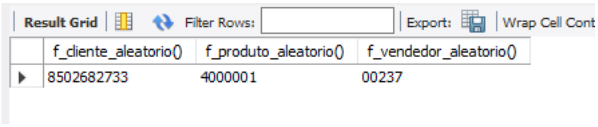
6) Nos exercícios associados a esta aula criamos a função para buscar, de forma aleatória, um produto e um vendedor. Mas caso não tenha sido feito o exercício proceda criando as duas funções conforme o código abaixo:

```
DELIMITER $$

USE `sucos_vendas`$$

CREATE DEFINER=`root`@`localhost` FUNCTION `f_produto_aleatorio`() RETURNS varchar(10) CHARSET utf8m

BEGIN

    DECLARE vRetorno VARCHAR(10);
```

```
    DECLARE num_max_tabela INT;

    DECLARE num_aleatorio INT;

    SELECT COUNT(*) INTO num_max_tabela FROM tabela_de_produtos;

    SET num_aleatorio = f_numero_aleatorio(1, num_max_tabela);

    SET num_aleatorio = num_aleatorio - 1;

    SELECT CODIGO_DO_PRODUTO INTO vRetorno FROM tabela_de_produtos

    LIMIT num_aleatorio, 1;

RETURN vRetorno;

END$$


DELIMITER ;

;


DELIMITER $$

USE sucos_vendas$$

CREATE DEFINER=root@localhost FUNCTION f_vendedor_aleatorio() RETURNS varchar(5) CHARSET utf8mb4

BEGIN

    DECLARE vRetorno VARCHAR(5);

    DECLARE num_max_tabela INT;

    DECLARE num_aleatorio INT;

    SELECT COUNT(*) INTO num_max_tabela FROM tabela_de_vendedores;

    SET num_aleatorio = f_numero_aleatorio(1, num_max_tabela);

    SET num_aleatorio = num_aleatorio - 1;

    SELECT MATRICULA INTO vRetorno FROM tabela_de_vendedores

    LIMIT num_aleatorio, 1;

RETURN vRetorno;

END$$


DELIMITER ;

;
```

7) Podemos testar as funções num único SELECT:

```
SELECT f_cliente_aleatorio(), f_produto_aleatorio(), f_vendedor_aleatorio();
```



8) Vamos, finalmente, criar a SP para incluir uma venda de forma aleatória. Digite e execute a criação da SP abaixo:

```
USE sucos_vendas;
DROP procedure IF EXISTS p_inserir_venda;

DELIMITER $$
USE sucos_vendas$$
CREATE PROCEDURE p_inserir_venda(vData DATE, max_itens INT,
max_quantidade INT)
BEGIN
DECLARE vCliente VARCHAR(11);
DECLARE vProduto VARCHAR(10);
DECLARE vVendedor VARCHAR(5);
DECLARE vQuantidade INT;
DECLARE vPreco FLOAT;
DECLARE vItens INT;
DECLARE vNumeroNota INT;
DECLARE vContador INT DEFAULT 1;
SELECT MAX(numero) + 1 INTO vNumeroNota from notas_fiscais;
SET vCliente = f_cliente_aleatorio();
SET vVendedor = f_vendedor_aleatorio();

INSERT INTO notas_fiscais (CPF, MATRICULA, DATA_VENDA, NUMERO, IMPOSTO)
VALUES (vCliente, vVendedor, vData, vNumeroNota, 0.18);
SET vItens = f_numero_aleatorio(1, max_itens);
WHILE vContador <= vItens
```

```
DO
   SET vProduto = f_produto_aleatorio();
   SET vQuantidade = f_numero_aleatorio(10, max_quantidade);
   SELECT PRECO_DE_LISTA INTO vPreco FROM tabela_de_produtos
   WHERE CODIGO_DO_PRODUTO = vProduto;
   INSERT INTO itens_notas_fiscais (NUMERO, CODIGO_DO_PRODUTO,
   QUANTIDADE, PRECO) VALUES (vNumeroNota, vProduto, vQuantidade, vPreco);
   SET vContador = vContador + 1;
END WHILE;
END$$
DELIMITER ;
```

9) Execute a SP para criar uma venda:

```
Call p_inserir_venda('20190517', 3, 100);
```

10) Se executarmos várias vezes a criação da venda teremos, num determinado momento, erro de PK.

11) Para resolver este problema precisamos testar se o produto aleatório determinado pela função já existe na tabela de vendas. Modifique a SP com o código abaixo:

```
USE sucos_vendas;
DROP procedure IF EXISTS p_inserir_venda;

DELIMITER $$
USE sucos_vendas$$
CREATE DEFINER=root@localhost PROCEDURE p_inserir_venda(vData DATE, max_itens INT,
max_quantidade INT)
BEGIN
DECLARE vCliente VARCHAR(11);
DECLARE vProduto VARCHAR(10);
DECLARE vVendedor VARCHAR(5);
```

```
DECLARE vQuantidade INT;
DECLARE vPreco FLOAT;
DECLARE vItens INT;
DECLARE vNumeroNota INT;
DECLARE vContador INT DEFAULT 1;
DECLARE vNumItensNota INT;
SELECT MAX(numero) + 1 INTO vNumeroNota from notas_fiscais;
SET vCliente = f_cliente_aleatorio();
SET vVendedor = f_vendedor_aleatorio();
INSERT INTO notas_fiscais (CPF, MATRICULA, DATA_VENDA, NUMERO, IMPOSTO)
VALUES (vCliente, vVendedor, vData, vNumeroNota, 0.18);
SET vItens = f_numero_aleatorio(1, max_itens);

WHILE vContador <= vItens
DO
   SET vProduto = f_produto_aleatorio();
   SELECT COUNT(*) INTO vNumItensNota FROM itens_notas_fiscais
   WHERE NUMERO = vNumeroNota AND CODIGO_DO_PRODUTO = vProduto;
   IF vNumItensNota = 0 THEN
      SET vQuantidade = f_numero_aleatorio(10, max_quantidade);
      SELECT PRECO_DE_LISTA INTO vPreco FROM tabela_de_produtos
      WHERE CODIGO_DO_PRODUTO = vProduto;
      INSERT INTO itens_notas_fiscais (NUMERO, CODIGO_DO_PRODUTO,
      QUANTIDADE, PRECO) VALUES (vNumeroNota, vProduto, vQuantidade, vPreco);
   END IF;
   SET vContador = vContador + 1;
END WHILE;
END$$

DELIMITER ;
```

12) No curso de manipulação de dados vimos, quando aprendemos `TRIGGERs`, que devemos criar uma para inclusão, uma para alteração e outra para exclusão. Se, dentro delas, tivemos que executar os mesmos comandos a manutenção será problemática porque qualquer mudança na regra de negócio deverá ser feita nas três `TRIGGERs`. Mas, se usarmos SP, as mudanças deverão ser feita apenas na SP e não em cada uma das `TRIGGERs`.

13) Criando a SP:

```
USE sucos_vendas;

DROP procedure IF EXISTS p_calculo_faturamento;


DELIMITER $$

USE sucos_vendas$$

CREATE PROCEDURE p_calculo_faturamento()

BEGIN

  DELETE FROM TAB_FATURAMENTO;

  INSERT INTO TAB_FATURAMENTO

  SELECT A.DATA_VENDA, SUM(B.QUANTIDADE * B.PRECO) AS TOTAL_VENDA FROM

  NOTAS_FISCAIS A INNER JOIN ITENS_NOTAS_FISCAIS B

  ON A.NUMERO = B.NUMERO

  GROUP BY A.DATA_VENDA;

END$$


DELIMITER ;
```

14) Criando as `TRIGGERs` usando a SP:

```
CREATE TABLE TAB_FATURAMENTO

(DATA_VENDA DATE NULL, TOTAL_VENDA FLOAT);


DELIMITER //

CREATE TRIGGER TG_CALCULA_FATURAMENTO_INSERT AFTER INSERT ON ITENS_NOTAS_FISCAIS

FOR EACH ROW BEGIN

  Call p_calculo_faturamento;

END//
```

```
DELIMITER //

CREATE TRIGGER TG_CALCULA_FATURAMENTO_UPDATE AFTER UPDATE ON ITENS_NOTAS_FISCAIS

FOR EACH ROW BEGIN

  Call p_calculo_faturamento;

END//


DELIMITER //

CREATE TRIGGER TG_CALCULA_FATURAMENTO_DELETE AFTER DELETE ON ITENS_NOTAS_FISCAIS

FOR EACH ROW BEGIN

  Call p_calculo_faturamento;


END//
```