2022.1_BD2M - Avaliação Unidade 1

aeagf@discente.ifpe.edu.br Alternar conta

Rascunho salvo.

Seu e-mail será registrado quando você enviar este formulário.

*Obrigatório

Nome: *

Aldenis Everton Alves Guilherme de França

Matrícula: *

20211TADS-JG0694

Segue o script de criação e povoamento inicial do esquema. Para rodar o script e realizar suas respostas, devem utilizar a versão do SGBD PostgreSQL disponível em https://sqliteonline.com/.

```
create table produto (
   codigo integer PRIMARY key,
   descricao varchar(30) not null,
   volume varchar(30),
   marca_produto varchar(30),
   qtdeEstoque integer not null,
   classificacao_etaria integer,
   categoria_produto varchar(20)
);
create table cliente (
      id integer PRIMARY key,
   nome varchar(40)  NOT NULL,
   dt_nascimento date NOT NULL,
   sexo char,
   tipo varchar(20)
);
create table entradas (
      cod_produto integer,
   lote integer,
      qtdeEntrada integer not null,
   preco_unitario decimal (8,2) not null,
   dt_entrada DATE,
      primary key (cod_produto, lote),
   FOREIGN key (cod_produto) REFERENCES produto (codigo)
);
create table saidas (
      cod_produto integer,
      id_cliente integer,
   dt_saida DATE,
   preco_venda decimal (8,2) NOT NULL,
   qtdeSaida integer NOT NULL,
      PRIMARY key (cod_produto,id_cliente,dt_saida),
   FOREIGN key (cod_produto) REFERENCES produto (codigo),
   FOREIGN key (id_cliente) REFERENCES cliente (id)
);

INSERT INTO produto values
 (1, 'Agua Mineral', '500ml', 'Indaiá', 966, null, 'Bebida'),  (2, 'Agua Mineral', '20l', 'Indaiá', 115, 0, 'Bebida'),
```

```
 (3, 'Cerveja lata 350ml', 'Pack com 12', 'Skol', 100, 18, 'Alcoólico'),  (4, 'Vodca', '1L', 'Slova', 218, 18, 'Alcoólico'),
 (5, 'Coca-cola', '2,5l','Coca-cola', 0, null, 'Refrigerante'),  (6, 'Gelo em cubo', 'saco 10kg',null, 161, 0, 'Acompanhamento'),
 (7, 'Cerveja long neck', 'Pack com 6','Budweiser', 167, 18, 'Alcoólico'),
 (8, 'Cerveja lata 350ml', 'Pack com 12', 'Bohemia', 0, 18, 'Alcoólico'), (9, 'Fanta', '2l','Coca-cola', 507, null, 'Refrigerante');
insert into cliente VALUES
 (1, 'Maria do Socorro', '1989-09-29', 'F', 'VIP'),   (2, 'Tiago Sales', '2004-08-23', 'M', 'VIP'),
 (3, 'Gildo Bezerra', '1992-04-06', 'M', 'Comum'),   (4, 'Suellen Barboza', '2004-10-30', 'F', 'Comum'),
 (5, 'Lourenço Chagas', '1996-03-25', 'M', NULL);
insert into saidas VALUES
 (1,2,'2022-02-01',1.36,50),(1,1,'2022-01-29',1.87,40),(1,1,'2022-02-04',1.69,90),(1,5,'2022-03-25',1.41,200),
 (2,3,'2022-03-09',4.97,10),(2,3,'2022-03-28',5,8),(2,3,'2022-04-10',5.10,8), (4,3,'2022-03-20',17.96,48), (4,3,'2022-03-30',18,50),
 (4,1,'2022-03-20',18.69,10),(4,1,'2022-04-30',18.69,7),(4,5,'2022-04-20',19.69,12),
 (2,5,'2022-04-10',5.10,14),(2,4,'2022-04-11',5.10,10),(2,3,'2022-04-20',5.25,3),
 (6,1,'2022-03-10',8.96,10),(6,3,'2022-03-10',8.96,15),(6,1,'2022-04-12',9,14),
 (9,4,'2022-03-20',5.96,10),(9,5,'2022-03-23',6,15),(9,1,'2022-03-27',6.36,8);
insert into entradas VALUES
 (1,890,100,0.24,'2022-01-21'),(1,789,410,0.25,'2022-02-01'),(4,6985,200,10.63,'2022-03-15'), (4,2547,145,9.99,'2022-05-04'),
 (1,696,600,0.23,'2022-02-17'),(1,245,236,0.24,'2022-03-25'),(6,879,100,2.35,'2022-03-02'), (6,587,100,3.65,'2022-03-28'),
 (2,748,36,1.68,'2022-03-01'),(2,874,62,1.43,'2022-03-15'),(2,126,70,1.90,'2022-04-02'),
 (3,471,100,8.63,'2022-05-01'),(7,123,36,4.69,'2022-03-25'),(7,489,59,5,'2022-03-25'),(7,698,47,5.32,'2022-03-25'),
 (7,147,25,4.98,'2022-03-25'),(9,748,300,2.36,'2022-02-01'),(9,874,100,2.47,'2022-03-01'), (9,369,140,1.99,'2022-04-01');
```

VIEWS

Criar um relatório que exiba os nomes dos produtos que nunca foram vendidos e o preço gasto em suas entradas. 8 pontos

```
CREATE or REPLACE view VW_nao_vendidos AS
SELECT p.descricao, sum(preco_unitario*qtdeentrada)
FROM produto p left join entradas e
on p.codigo = e.cod_produto
WHERE p.codigo not in
(SELECT cod_produto from saidas)
GROUP by p.descricao
```

Criar um relatório que apresente o nome dos produtos, a quantidade em estoque, o quanto já se gastou com suas entradas e o quanto já se arrecadou com suas saídas. Além disso, deve calcular qual o valor do lucro/prejuízo (CÁLCULO: quanto se recebeu - quanto se gastou) com esse produto. 12 pontos

```
CREATE or REPLACE view VW_lucro_prejuizo AS
SELECT descricao, qtdeestoque, sum(preco_unitario*qtdeentrada) as compra,
sum(preco_venda*qtdesaida) as arrecadado,
(sum(preco_venda*qtdesaida) - sum(preco_unitario*qtdeentrada)) as lucro_prejuizo
from produto left join entradas on codigo = entradas.cod_produto
LEFT JOIN saidas on codigo = saidas.cod_produto
GROUP by produto.descricao, produto.qtdeestoque
GROUP by produto.descricao, produto.qtdeestoque
```

Criar um relatório com o valor gasto em saídas por tipo de cliente e por marca de produto. 10 pontos

```
CREATE or REPLACE view VW_gasto_cliente_produto AS
SELECT sum(preco_venda*qtdesaida) as arrecadado, tipo as tipo_cliente, marca_produto
from saidas LEFT JOIN cliente on id_cliente = cliente.id
LEFT JOIN produto on saidas.cod_produto = codigo
GROUP by tipo, marca_produto
```

Criar um relatório que apresente o nome do cliente, o nome do produto, a data, a quantidade e o valor total das saídas registradas no sistema. 10 pontos

```
CREATE or REPLACE view VW_consumo_cliente AS
SELECT cliente.nome, produto.descricao, saidas.dt_saida, saidas.qtdesaida,
sum(preco_venda*qtdesaida) as arrecadado
FROM saidas LEFT JOIN cliente on id_cliente = cliente.id
LEFt JOIN produto on saidas.cod_produto = produto.codigo
GROUP by cliente.nome, produto.descricao, saidas.dt_saida,saidas.qtdesaida
```

FUNÇÕES E PROCEDIMENTOS

Nessa ETAPA, a depender da solicitação, você escolhe criar uma função ou um procedimento.

Criar um subprograma que atualize o estoque de todos os produtos de uma marca passada como parâmetro. A quantidade do estoque deve ser atualizada com a soma de todas as quantidades das entradas subtraída da soma de todas as quantidades das saídas do produto. 10 pontos

```
CREATE or replace procedure atualize_estoque(nome_marca TEXT)
LANGUAGE PLPGSQL AS $$
begin
   UPDATE produto set
   qtdeestoque = COALESCE((select sum(qtdeentrada) from entradas WHERE
entradas.cod_produto = produto.codigo AND produto.marca_produto = nome_marca),0)
   - COALESCE((select sum(qtdesaida) from saidas WHERE saidas.cod_produto =
produto.codigo and produto.marca_produto = nome_marca),0)
   WHERE nome_marca = produto.marca_produto;
end;$$;
```

Criar um subprograma que retorne o catálogo contendo o nome, a categoria, a quantidade em estoque e a classificação etária dos produtos de uma marca passada como parâmetro. Caso a marca não exista, deve levantar um erro.

12 pontos

```
CREATE or replace FUNCTION retorna_catalogo(nome_marca TEXT)
RETURNS table (descricao varchar(30), categoria_produto varchar(30), qtdeestoque integer,
classificacao_etaria integer)
LANGUAGE PLPGSQL AS $$
DECLARE
      existe_marca TEXT;
BEGIN
      SELECT marca_produto FROM produto WHERE marca_produto = nome_marca into
existe_marca;
   if (existe_marca is NULL) THEN
      RAISE EXCEPTION 'Marca não existe';
   ELSE
      RETURN QUERY SELECT p.descricao, p.categoria_produto, p.qtdeestoque,
p.classificacao_etaria
       FROM produto p WHERE marca_produto = nome_marca;
   END if;
END;$$;
```

Criar um subprograma que retorne a quantidade de produtos de uma marca que deram entrada de uma data específica. Caso não tenha havido entrada nessa data para essa marca, deve retornar um erro.

12 pontos

```
CREATE or replace FUNCTION quantidade_produtos(data_entrada DATE, nome_marca
TEXT)
RETURNS integer
LANGUAGE PLPGSQL AS $$
DECLARE
      existe_data TEXT;
   qtd_produtos INTEGER;
BEGIN
      SELECT dt_entrada FROM entradas e, produto p
   WHERE e.cod_produto = p.codigo AND dt_entrada = data_entrada
   and p.marca_produto = nome_marca into existe_data;

   if (existe_data is NULL) THEN
      RAISE EXCEPTION 'Não houve entrada de produto nesta data';
   ELSE
      SELECT sum(e.qtdeentrada) FROM entradas e, produto p
       WHERE e.cod_produto = p.codigo AND e.dt_entrada = data_entrada and
p.marca_produto = nome_marca
       into qtd_produtos;
   END if;
   RETURN qtd_produtos;
END;$$;
```

TRIGGERS

As duas regras básicas desse negócio são:
+ a cada entrada de um produto, a quantidade em estoque desse produto deve ser incrementada pela quantidade adquirida.
+ a cada saída de um produto, a quantidade em estoque desse produto deve ser decrementada pela quantidade retirada. Além disso, a saída só pode ser registrada se a quantidade solicitada para saída seja menor ou igual a quantidade disponível em estoque.

Criar uma trigger que só permite a realização de uma saída quando há estoque disponível para a quantidade desejada. Caso não tenha, deve bloquear o registro da saída; caso haja produto suficiente, deve decrementar o estoque do produto com a quantidade retirada. 14 pontos

```
CREATE or replace FUNCTION atualiza_venda()
RETURNS TRIGGER
LANGUAGE PLPGSQL AS $$
DECLARE
	qtd_disponivel INTEGER;
BEGIN
	SELECT qtdeestoque FROM produto WHERE codigo = new.cod_produto into
qtd_disponivel;
   if qtd_disponivel < new.qtdesaida THEN
	RAISE EXCEPTION 'Quantidade disponível insuficiente!';
   ELSE
		UPDATE produto SET qtdeestoque = qtdeestoque - new.qtdesaida where codigo =
new.cod_produto;
   end if;
   RETURN NEW;
end;$$;

CREATE or replace TRIGGER decrementa_estoque
BEFORE insert or UPDATE ON saidas
FOR EACH ROW
EXECUTE PROCEDURE atualiza_venda();
```

Criar uma trigger que após o registro da entrada de um produto, deve se incrementar o valor do estoque com a quantidade registrada na entrada. 12 pontos

```
CREATE or replace FUNCTION atualiza_estoque()
RETURNS TRIGGER
LANGUAGE PLPGSQL AS $$
BEGIN
      UPDATE produto SET qtdeestoque = qtdeestoque + new.qtdeentrada
   where codigo = new.cod_produto;
   RETURN NEW;
end;$$;

CREATE or replace TRIGGER incrementa_estoque
AFTER insert or UPDATE ON entradas
FOR EACH ROW
EXECUTE PROCEDURE atualiza_estoque();
```

TRIGGER EXTRA

A última regra desse negócio é que a compra de produtos da categoria alcoólicas não seja realizada por clientes menores de 18 anos.
OBS: para calcular idade a partir de uma data, usar: "extract (year from age(campo_tipo_data))"

Então, solicito:

(EXTRA) Criar uma trigger que antes de cada saída de produto, verifica se o cliente que está associado à saída é de maior. Caso o cliente não seja de maior, deve ser proibida a venda, ou seja, deve bloquear o registro dessa saída. 10 pontos

```
CREATE or REPLACE FUNCTION verifica_maioridade()
RETURNS TRIGGER
LANGUAGE PLPGSQL AS $$
DECLARE
      idade_cliente INTEGER;
   permissao_faixa INTEGER;
BEGIN
      SELECT extract(year from age(dt_nascimento)) from cliente WHERE cliente.id =
NEW.id_cliente INTO idade_cliente;
   SELECT classificacao_etaria FROM produto WHERE produto.codigo = NEW.cod_produto
into permissao_faixa;
      if idade_cliente < 18 and permissao_faixa = 18 THEN
      RAISE EXCEPTION 'Produto não pode ser vendido. Cliente menor de idade!';
      end if;
   RETURN NEW;
end;$$;

CREATE or REPLACE TRIGGER maioridade_cliente
BEFORE INSERT on saidas
for EACH ROW
EXECUTE PROCEDURE verifica_maioridade();
```

Enviar Limpar formulário

Google Formulários