UFCG/CEEI/ Bach. em Ciência da Computação
Período: 2024.2
Disciplina: Banco de Dados 1
Professor: Cláudio Baptista, Ph.D.

Projeto em Grupo - Fase 3

Responda às questões relacionadas ao seu grupo e submeta a resposta no *Classroom*.

Grupo 1

1 - Implemente uma **function** PL/SQL chamada *calcular_media_compras_cliente*. Essa **function** deverá receber um código de cliente e retornar a média dos valores de suas compras já finalizadas.

2 - Implemente uma **function** PL/SQL chamada *get_total_prod_vencidos*. Ela deverá receber uma data e retornar o total em estoque de produtos que possuem data de validade superior à recebida como parâmetro.

3 - Implemente uma **procedure** PL/SQL chamada *aplicar_desconto_categoria*. Esta procedure recebe o código de uma categoria e um percentual de desconto. Ela deverá aplicar esse desconto no preço atual de todos os produtos dessa categoria.

4 - Implemente uma **procedure** PL/SQL chamada *remover_historico_cliente*. Essa procedure recebe o código de um cliente e deverá remover o histórico de produtos visualizados há mais de 1 ano.

5 - Implemente uma **procedure** PL/SQL chamada *atualizar_data_validade*. Esta procedure recebe o código de uma categoria e uma quantidade em dias, e atualiza o valor da data de vencimento de todos os produtos da categoria passada para que seja igual a sua data de fabricação mais a quantidade de dias passados como parâmetro.

6 - Crie uma **view** com o nome *v_frete_por_transportadora* que exibe, por transportadora, a soma total dos valores de frete de todas as ordens de compra já finalizadas, mostrando o código e nome da transportadora e o total.

7 - Crie uma **view** com o nome *v_fornecedor_sem_produto* que exibe o nome e código dos fornecedores que não possuem nenhum produto cadastrado no sistema.

8 - Crie um **trigger** com o nome *tg_verificar_preco_produto* para garantir que, ao atualizar o preço de um produto, o novo preço seja pelo menos o dobro do preço anterior. Caso contrário, lance uma exceção.

9 - Crie um **trigger** com o nome *tg_ajustar_nome_cliente* que, ao inserir um cliente, adicione

o texto ‘ - NOVATO’ ao seu nome.

10 - Crie um **trigger** com o nome tg_*garantir_intervalo_nota* que, antes de inserir uma nova avaliação, garanta que a nota esteja entre 0 e 5. Caso contrário, lance uma exceção.

Grupo 2

1 - Implemente uma **function** PL/SQL chamada *calcula_media_cidade* que deverá receber o nome da cidade e retornar o valor médio das ordens de compra efetuadas com endereço de entrega para a cidade informada.

2 - Implemente uma **function** PL/SQL chamada *get_total_prod_comprados*. Esta função deverá receber uma data inicial e uma data final, e retornar a soma dos valores dos produtos comprados entre essas datas.

3 - Crie uma **procedure** PL/SQL chamada *adicionar_produto_estoque*, que receba o código de um produto e o código de um centro de distribuição e adicione 50 unidades desse produto ao estoque no centro de distribuição. Caso o estoque não esteja zerado, a quantidade deve ser atualizada, adicionando mais 50 à quantidade existente.

4 - Crie uma **procedure** PL/SQL chamada *atualizar_status_compra* que, ao receber o código de uma ordem de compra, altere seu status para "EM SEPARAÇÃO", desde que o status atual seja "AGUARDANDO PAGAMENTO".

5 - Crie uma **procedure** PL/SQL chamada *atualizar_preco_produto* que, dado o código de uma categoria, aplica um desconto de 10% no preço de venda dos produtos dessa categoria que tem vencimento para os próximos 30 dias.

6 - Crie uma **view** com o nome *v_produtos_estoque* que liste todos os produtos com seu respectivo nome, preço, quantidade em estoque e o nome da categoria.

7 - Crie uma **view** com o nome *v_media_compras_clientes* que exibe o valor médio das compras realizadas por cada cliente. A view deve apresentar o código do cliente e o valor médio das compras (exibindo 0 para clientes que ainda não realizaram compras), ordenado do menor valor médio para o maior.

8 - Crie um **trigger** com o nome *tg_ajustar_data_indicacao* que, ao inserir um novo cliente, defina a data de indicação como a data atual caso o cliente não tenha sido indicado por ninguém.

9 - Crie um **trigger** com o nome *tg_verificar_data_validade* que, ao inserir ou atualizar um produto, valide se a data de validade do produto não é anterior à data atual, e lance uma exceção caso isso ocorra.

10 - Crie um **trigger** com o nome *tg_ajustar_especificacao_nula* que, ao inserir ou atualizar um produto, define ESPECIFICACAO = ‘Sem informações’ quando uma tupla for inserida com especificação de produto nula.

Grupo 3

1 - Implemente uma **procedure** PL/SQL chamada *ajusta_preco*. Esta procedure recebe o código de uma categoria e aplica um aumento de R$50,00 no preço (venda e compra) de todos os produtos da categoria informada.

2 - Implemente uma **procedure** PL/SQL chamada *aumenta_pontos_cliente*. Esta procedure deverá receber um CPF e um valor de pontos, adicionando os pontos informados ao campo PONTOS do cliente com o CPF informado. Caso o cliente não exista, a procedure deverá exibir uma mensagem de erro apropriada.

3 - Crie uma **view** chamada *levantamento_de_compras_papelaria*, que lista todos os clientes e as quantidades de compras realizadas por eles, de todos os produtos da categoria "Papelaria".

4 - Implemente uma **função** PL/SQL chamada *calcula_frete_categoria*. Esta função deverá receber um código de categoria e retornar o valor total dos fretes das ordens de compras que possuem produtos daquela categoria.

5 - Crie um **trigger** para modificar o nome da transportadora deixando a primeira letra no nome sempre maiuscula quando esse dado for inserido ou atualizado.

6 - Crie um **trigger** que não permita a inserção de um produto no carrinho se a quantidade solicitada for maior que a quantidade disponibilizada pelo Centro de Distribuição daquele produto. Ao não permitir a inserção nesses casos, deve ser exibido uma mensagem de erro.

7 - Implemente uma **procedure** PL/SQL chamada *remover_fornecimento*. Esta procedure recebe o código de um produto e um CEP, e apaga do banco qualquer relação de fornecimento entre o produto indicado e fornecedores que possuam o CEP passado como parâmetro.

8 - Implemente uma **função** PL/SQL chamada *get_total_prod_comprados*. Esta função deverá receber uma data e retornar a soma dos valores dos produtos que foram comprados em uma data igual ou anterior a data recebida como parâmetro.

9 - Crie um **trigger** para baixar o estoque de um produto quando ele for vendido.

10 - Crie uma **view** que exibe o código, o nome do produto e o preço juntamente com a média das avaliações por produto.

Grupo 4

1 - Implemente uma **procedure** PL/SQL chamada *devolver_estoque*. Esta procedure recebe o código de uma ordem de compra e devolve todos os produtos comprados para os estoques do centro de distribuição dele, ou seja, ele soma a quantidade atual no estoque com a quantidade que previamente havia sido comprada.

2 - Implemente uma **função** PL/SQL chamada *get_qtd_prox_vencimentos*. Esta função recebe o código de um centro de distribuição e retorna a quantidade total de produtos que estão próximos ao vencimento. Considere que um produto próximo ao vencimento deva possuir uma data de validade que expira em 5 dias.

3 - Crie um **trigger** para modificar o nome do fornecedor deixando a primeira letra no nome sempre maiúscula quando esse dado for inserido ou atualizado.

4 - Crie um **trigger** para substituir o valor do campo de número por 's/n' na tabela de ENDERECO quando, no momento da inserção, uma tupla vier com esse valor vazio.

5 - Crie uma **view** que lista os fornecedores e a quantidade de produto por categoria fornecidos por eles. A view deve exibir o nome dos fornecedores, a categoria do produto e a quantidade de produtos por categoria.

6 - Implemente uma **procedure** PL/SQL chamada *remover_avaliacao_falsa*. Esta procedure recebe o código de um produto e remove do banco todas as avaliações ligadas a ordens de compras com status diferentes de 'FINALIZADA'.

7 - Crie um **trigger** para não permitir que uma ordem de compra tenha uma quantidade negativa de qualquer item. Ao não permitir isso, deve ser exibido uma mensagem de erro.

8 - Implemente uma **procedure** PL/SQL chamada *decrementar_preco*. Esta procedure recebe o código de uma categoria e um valor em porcentagem, e diminui o preço de venda e compra de todos os produtos pertencentes à categoria especificada com base na porcentagem passada.

9 - Crie uma **view** que exibe o código do cliente, o nome e o valor total das compras (quantidade x preço do produto na venda) efetuadas por ele e que foram finalizadas.

10 - Crie uma **view** que liste o nome das transportadoras que já transportaram o equivalente a mais de 1 milhão de reais em produtos (quantidade x preço do produto na venda).

Grupo 5

1 - Implemente uma **função** PL/SQL chamada *calcular_entregas_periodo*. Esta função deverá receber um código de transportadora e uma data inicial, e retornar a quantidade de ordens de compras ligadas à transportadora selecionada a partir do período de tempo informado.

2 - Implemente uma **procedure** PL/SQL chamada *atualizar_endereço*. Esta procedure recebe o código de uma cliente e atualiza o endereço de todas as suas ordens de compras para o endereço atual de seu cadastro, mas somente se o status dela for 'AGUARDANDO PAGAMENTO'.

3 - Crie uma **view** chamada *v_clientes_indicados* que liste o nome dos clientes que foram indicados por algum cliente que possua menos de 2000 reais em compras.

4 - Crie uma **view** chamada *v_emails_ativos*, que mostre todos os clientes que abriram um email e clicaram no seu conteúdo. A view deve exibir o nome e código do cliente, bem como o assunto do email e a data em que este foi enviado.

5 - Escreva um **trigger** chamado *trg_no_update_cliente*, que não permita a alteração de dados na tabela CLIENTE.

6 - Crie um **trigger** chamado *trg_abrevia_nome_sobrenome*, para, quando receber o valor SILVA ou OLIVEIRA, como sobrenome de algum cliente, abreviar para 'S.' ou 'O.', respectivamente.

7 - Implemente uma **function** PL/SQL chamada *calcula_media_cliente*. Esta função deverá receber o código de um cliente, calcular e retornar o valor médio das notas fiscais do respectivo cliente.

8 - Implemente uma **function** PL/SQL chamada *calcula_produto_mais_vendido*. Esta função deverá receber uma data, e retornar o produto que mais foi vendido a partir desse dia.

9- Implemente uma **procedure** PL/SQL chamada *barateia_produtos_mal_avaliados*. Esta procedure recebe uma nota e diminui 10% do valor de venda do produto, que possua uma avaliação média menor ou igual à nota passada como parâmetro.

10 - Crie um **trigger** chamado *trg_limita_aumento_preco* que impeça a atualização do preço de venda de um produto para um valor que seja o dobro do anterior.

Grupo 6

1 - Crie uma **view** chamada *v_produto_categoria_fornecedor*, que exibe o nome de cada produto, o nome da sua respectiva categoria e o nome do fornecedor. Em caso de múltiplos fornecedores, exiba o primeiro por ordem alfabética.

2 - Crie uma **view** chamada *v_enderecos_cliente*, que exibe o nome do cliente e cada um dos endereços distintos de suas ordens de compra, mostrando todos os atributos referentes ao endereço (rua, número, bairro, cidade, estado e CEP).

3 - Crie uma **trigger** chamada *trg_forcar_status_pedido* que, antes de inserir uma nova ordem de compra na tabela, vai definir automaticamente o campo STATUS como 'AGUARDANDO PAGAMENTO', independentemente do valor informado no comando INSERT.

4- Crie um **trigger** chamado *trg_limita_quantidade_carrinho* que impeça a inserção de um produto no carrinho se a quantidade solicitada for maior que a disponível no item de inventário do produto.

5 - Implemente uma **procedure** PL/SQL chamada *calcular_valor_nota*. Esta procedure recebe o código da ordem de compra e atualiza o VALOR_TOTAL da sua respectiva nota fiscal, utilizando a fórmula:
(preço de cada produto comprado x quantidade do respectivo produto) + frete - desconto.

6 - Implemente uma **função** PL/SQL chamada *calcula_total_fornecedor*. Esta função deverá receber o código do fornecedor e retornar o valor total da soma dos preços de compra dos produtos fornecidos por ele.

7 - Crie uma **view** chamada *v_compras_por_localidade*, que exibe o nome da cidade, o bairro, o CEP e o número total de compras finalizadas naquela localidade. Caso nenhuma compra tenha sido realizada, exiba o valor 0.

8 - Crie um **trigger** chamado *trg_proibe_eniedson_osorio*, para, quando receber o valor de nome “Eniedson” e sobrenome “Osório” de algum cliente, não permitir a criação.

9 - Crie uma **view** chamada *v_clientes_acima_5000*, que lista os clientes que realizaram um total de compras acima de 5 mil reais. A visão deve exibir o nome, codigo e o valor total gasto em compras.

10 - Crie uma **trigger** chamado *trg_valida_data_nascimento*, que impeça a inserção ou atualização de um cliente cuja data de nascimento seja superior à data atual.