**QUESTÃO DISCURSIVA**

Considere o modelo para responder aos itens abaixo.

Diagrama

Descrição gerada automaticamente

**Observações:**

* Na tabela “**notas\_fiscais**”, as colunas “*cpf*” e “*matricula*” são chaves estrangeiras e referenciam, respectivamente, as tabelas “**clientes”** (*cpf*) e “**vendedores”** (*matricula*);
* Na tabela “**vendedores**”, a coluna “*chefe*” é chave estrangeira e referencia a própria tabela “**vendedores**” (*matricula*);
* Na tabela “**itens\_notas\_fiscais**”, as colunas “*numero*” e “*codigo\_do\_produto*” são a chave primária composta da tabela e referenciam, respectivamente, as tabelas “**notas\_fiscais**” (*numero*) e “**produtos**” (*codigo*).
* Na tabela “**itens\_notas\_fiscais**”, a coluna “*preco*” resulta da multiplicação da coluna “*quantidade*” pela coluna “preco\_de\_lista” da tabela “**produtos”**.

1. Escreva uma consulta SQL que retorne o nome de todos os vendedores e dos respectivos chefes desses vendedores (quando houver);
2. Por um erro na aplicação que é responsável por popular as tabelas acima, a coluna “*preço*” da tabela “**itens\_notas\_fiscais**” não foi computada, ou seja, todos os valores da coluna estão como “*null*”. Felizmente a equipe de programadores já realizou a correção na aplicação, porém os registros já inseridos ainda estão com valor “*null*” para essa coluna. Considerando as observações acima, escreva uma instrução SQL que corrija esse problema para todos os registros já existentes;
3. Utilizando apenas o operador “EXISTS” para realizar associação entre tabelas (proibido uso de junções e subconsultas que não utilizem tal operador), escreva uma consulta SQL que retorne todos os produtos que não tenham sido comprados pelo cliente de nome “*Leonardo Chapisco*”, considere que só há um cliente com tal nome;
4. Escreva uma consulta SQL que retorne o nome dos clientes e o valor total gasto por esses clientes (ex: “Leonardo Chapisco”, “400.0”) considerando todas as compras já realizadas. Além disso, a consulta só deve mostrar os clientes que compraram mais de R$ 100,00 (cem) reais em produtos e o resultado deve ser ordenado pelo valor total gasto em ordem decrescente.

SUGESTÕES DE SOLUÇÕES:

1. Escreva uma consulta SQL que retorne o nome de todos os vendedores e dos respectivos chefes desses vendedores (quando houver);

SELECT v1.nome AS vendedor, v2.nome AS chefe

    FROM vendedores v1

    LEFT OUTER JOIN vendedores v2

    ON v1.chefe = v2.matricula;

Percebe-se pelo modelo que há um autorrelacionamento da tabela “Vendedores”, e que o campo “chefe” nada mais é do que uma chave estrangeira que referencia a chave primária “matricula”.

Também é possível inferir que estamos diante de um relacionamento opcional de ambos os lados do relacionamento (bolinha branca/vazada), dessa forma pode existir vendedor sem chefe, bem como chefe sem chefiar ninguém.

Em outras palavras, devemos retornar uma consulta que liste todos os vendedores (LEFT JOIN) e mostre os respectivos chefes (quando houver). Para tanto, utilizamos na condição de junção o atributo “chefe” da tabela vendedor (lado esquerdo) e, no lado direito, o campo “matricula”;

1. Por um erro na aplicação que é responsável por popular as tabelas acima, a coluna “*preço*” da tabela “**itens\_notas\_fiscais**” não foi computada, ou seja, todos os valores da coluna estão como “*null*”. Felizmente a equipe de programadores já realizou a correção na aplicação, porém os registros já inseridos ainda estão com valor “*null*” para essa coluna. Considerando as observações acima, escreva uma instrução SQL que popule essa coluna corretamente para todos os registros já existentes;

UPDATE itens\_notas\_fiscais inf

   SET inf.preco = cast(inf.quantidade AS DECIMAL(10,2)) \*

     (SELECT p.preco\_de\_lista FROM produtos p WHERE p.codigo = inf.codigo\_do\_produto);

COMMIT; -- execute o commit para manter os dados atualizados, caso contrário o SQL Fiddle não mostrará os dados atualizados em requisições subsequentes;

No update acima, o Oracle não faz conversão de tipo implicitamente, sendo necessário o uso da função “cast()”, caso tenha esquecido desse detalhe, considere a solução como correta.

A instrução “commit” não faz parte da solução, adicionei ela ao código para que garantisse a persistência dos preços alterados para requisições subsequentes, pois a solução do item 4 considera que a coluna preço da tabela “**itens\_notas\_fiscais**” não estejam nulas.

1. Utilizando apenas o operador “EXISTS” para realizar associação entre tabelas (proibido uso de junções e subconsultas que não utilizem tal operador), escreva uma consulta SQL que retorne todos os produtos que não tenham sido comprados pelo cliente de nome “*Leonardo Chapisco*”, considere que só há um cliente com tal nome;

SELECT p.nome FROM produtos p

WHERE NOT EXISTS

    (SELECT 1 FROM itens\_notas\_fiscais inf

    WHERE EXISTS

        (SELECT 1 FROM notas\_fiscais nf

        WHERE EXISTS

            (SELECT 1 FROM clientes c

                WHERE p.codigo = inf.codigo\_do\_produto

                AND inf.numero = nf.numero

                AND nf.cpf = c.cpf

                AND c.nome = 'Leonardo Chapisco'

            )

        )

    )

O entendimento de consultas com a cláusula EXISTS pode ser melhor explicado pelos vídeos abaixo:

[Exists e Not Exists (parte 1)](https://www.youtube.com/watch?v=PM9tO8IKmPg)

[Exists e Not Exists (parte 2)](https://www.youtube.com/watch?v=5UPS06LOE3A)

1. Escreva uma consulta SQL que retorne o nome dos clientes e o valor total gasto por esses clientes (ex: “Leonardo Chapisco”, “400.0”) considerando todas as compras já realizadas. Além disso, a consulta só deve mostrar os clientes que compraram mais de R$ 100,00 (cem) reais em produtos e o resultado deve ser ordenado pelo valor total gasto em ordem decrescente.

SELECT c.nome, sum(inf.preco) AS total FROM itens\_notas\_fiscais inf

    INNER JOIN notas\_fiscais nf

        ON inf.numero = nf.numero

    INNER JOIN clientes c

        ON nf.cpf = c.cpf

    GROUP BY c.nome

    HAVING sum(inf.preco) > 100

    ORDER BY total DESC;

* Todas as soluções podem ser devidamente testadas no site [SQL Fiddle](https://d.docs.live.net/4ad7631881056c26/Documentos/sqlfiddle.com) utilizando o Oracle 11g.
* Os códigos de DDL e DML estão no repositório <https://github.com/pierryangelo/sql_chapisco>