|  |  |  |  |
| --- | --- | --- | --- |
| RM: | 557197 | Nome: | Larissa Muniz |
| RM: | 555678 | Nome: | João Victor Michaeli |
| RM: | 558062 | Nome: | Henrique Garcia |

**INSTRUÇÕES:**

**Avaliação Prática**

Leia atentamente cada questão.

Todas as questões devem ser respondidas em PL/SQL.

Utilize os comandos apropriados como funções, procedimentos, cursores, tratamento de exceções, entre outros.

Justifique suas escolhas sempre que possível.

Os nomes de objetos, colunas e tabelas utilizados nas questões são **exemplos ilustrativos** e têm como objetivo facilitar a compreensão e a execução das atividades. Para a resolução das questões, **utilize o modelo de dados apresentado em sala**, conforme já adotado nas demais atividades práticas do curso.

1. Crie uma função `fn\_categoria\_movimentada` que retorne a categoria mais movimentada (em quantidade total) no mês atual. Use JOINs, CASE e SYSDATE.

CREATE OR REPLACE FUNCTION fn\_categoria\_movimentada

RETURN VARCHAR2 IS

v\_nome\_produto VARCHAR2(100);

BEGIN

SELECT nom\_produto

INTO v\_nome\_produto

FROM (

SELECT

CASE

WHEN p.nom\_produto IS NULL THEN 'Produto sem nome'

ELSE p.nom\_produto

END AS nom\_produto,

SUM(m.qtd\_movimentacao\_estoque) AS total

FROM produto p

JOIN movimento\_estoque m ON p.cod\_produto = m.cod\_produto

WHERE m.dat\_movimento\_estoque

BETWEEN TRUNC(SYSDATE, 'MM') AND LAST\_DAY(SYSDATE)

GROUP BY

CASE

WHEN p.nom\_produto IS NULL THEN 'Produto sem nome'

ELSE p.nom\_produto

END

ORDER BY total DESC

)

WHERE ROWNUM = 1;

RETURN v\_nome\_produto;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

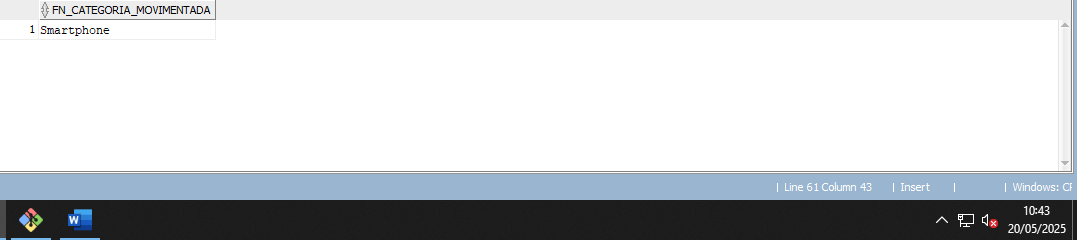
RETURN 'Sem movimentações este mês';

WHEN OTHERS THEN

RETURN 'Erro ao calcular produto';

END;

SELECT fn\_categoria\_movimentada FROM dual;



1. Crie uma procedure `sp\_valida\_produtos\_sem\_movimento` que liste produtos sem nenhum movimento nos últimos 6 meses. Use RIGHT JOIN, EXCEPTION e DBMS\_OUTPUT.

CREATE OR REPLACE PROCEDURE sp\_valida\_produtos\_sem\_movimento AS

v\_count NUMBER := 0;

BEGIN

FOR prod IN (

SELECT DISTINCT p.cod\_produto, p.nom\_produto

FROM movimento\_estoque m

RIGHT JOIN produto p ON m.cod\_produto = p.cod\_produto

WHERE m.dat\_movimento\_estoque IS NULL

OR m.dat\_movimento\_estoque < ADD\_MONTHS(SYSDATE, -6)

) LOOP

v\_count := v\_count + 1;

DBMS\_OUTPUT.PUT\_LINE('Produto sem movimento recente: ' ||

prod.cod\_produto || ' - ' || prod.nom\_produto);

END LOOP;

IF v\_count = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Todos os produtos tiveram movimentação nos últimos 6 meses.');

END IF;

EXCEPTION

WHEN OTHERS THEN

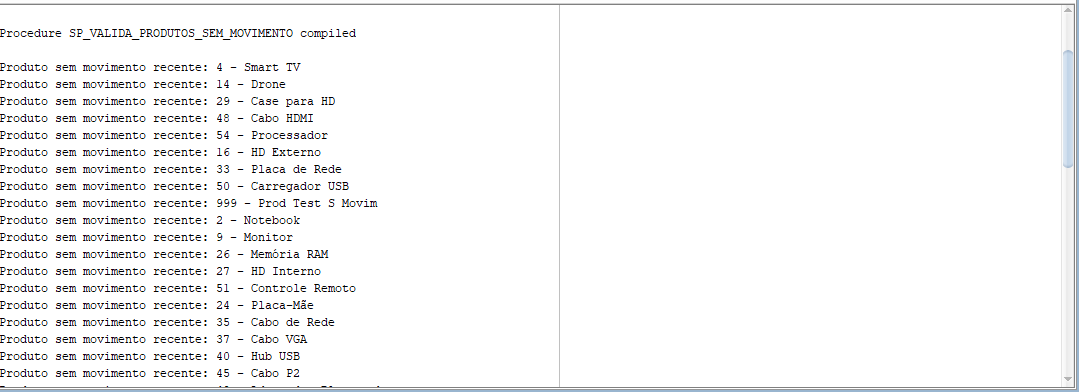
DBMS\_OUTPUT.PUT\_LINE('Erro ao verificar produtos sem movimento.');

END;

BEGIN

sp\_valida\_produtos\_sem\_movimento;

END;



1. Utilizando um cursor, liste todos os produtos com mais de 100 unidades movimentadas no total. Para cada um, calcule e exiba o total acumulado por meio de variável PL/SQL.

DECLARE

-- o total de movimentações de todos os produtos

v\_total\_geral NUMBER := 0;

-- busca produtos com mais de 100 unidades movimentadas

CURSOR c\_produtos IS

SELECT p.cod\_produto, p.nom\_produto, SUM(m.qtd\_movimentacao\_estoque) AS total\_mov

FROM produto p

JOIN movimento\_estoque m ON p.cod\_produto = m.cod\_produto

GROUP BY p.cod\_produto, p.nom\_produto

HAVING SUM(m.qtd\_movimentacao\_estoque) > 100;

BEGIN

FOR prod IN c\_produtos LOOP

-- acumula o total em variável

v\_total\_geral := v\_total\_geral + prod.total\_mov;

-- mostra os dados

DBMS\_OUTPUT.PUT\_LINE(

'Produto: ' || prod.cod\_produto || ' - ' || prod.nom\_produto ||

' | Total Movimentado: ' || prod.total\_mov

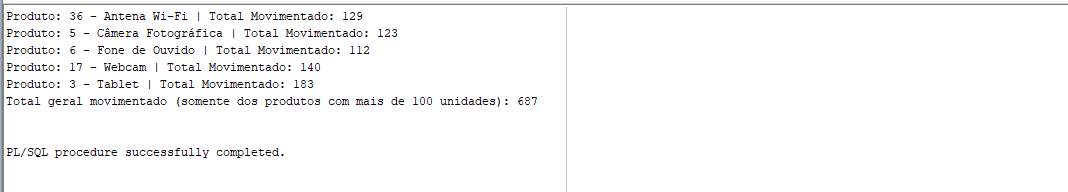
);

END LOOP;

-- exibe o acumulado final

DBMS\_OUTPUT.PUT\_LINE('Total geral movimentado (somente dos produtos com mais de 100 unidades): ' || v\_total\_geral);

END;



1. Implemente a procedure `sp\_qtd\_movimentos\_produto(p\_cod\_produto IN NUMBER, p\_total OUT NUMBER) ` que retorna a quantidade total de movimentos associados ao

CREATE OR REPLACE PROCEDURE sp\_qtd\_movimentos\_produto (

p\_cod\_produto IN NUMBER,

p\_total OUT NUMBER

) AS

BEGIN

-- Conta a quantidade de registros de movimentação para o produto

SELECT COUNT(\*)

INTO p\_total

FROM movimento\_estoque

WHERE cod\_produto = p\_cod\_produto;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

p\_total := 0; -- Se não encontrar nada, retorna 0

WHEN OTHERS THEN

p\_total := -1; -- Erro genérico

END;

DECLARE

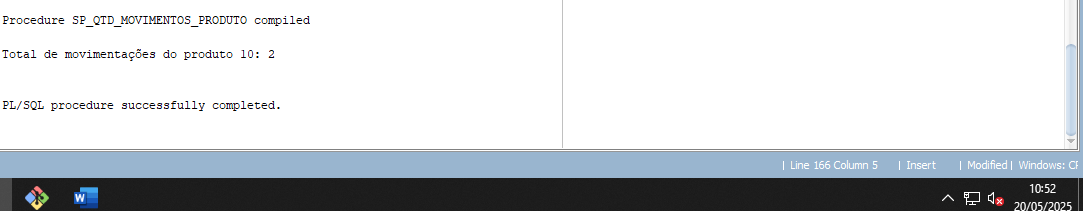
v\_total\_mov NUMBER;

BEGIN

sp\_qtd\_movimentos\_produto(10, v\_total\_mov);

DBMS\_OUTPUT.PUT\_LINE('Total de movimentações do produto 10: ' || v\_total\_mov);

END;



1. Atualize todos os produtos da categoria e de uma categoria específica, apenas se sua movimentação total for inferior a 50 unidades. Use cursor + LOOP com UPDATE.

DECLARE

CURSOR c\_produtos IS

SELECT p.cod\_produto

FROM produto p

LEFT JOIN movimento\_estoque m ON p.cod\_produto = m.cod\_produto

GROUP BY p.cod\_produto

HAVING SUM(NVL(m.qtd\_movimentacao\_estoque, 0)) < 50;

BEGIN

FOR prod IN c\_produtos LOOP

UPDATE produto

SET sta\_ativo = 'N'

WHERE cod\_produto = prod.cod\_produto;

DBMS\_OUTPUT.PUT\_LINE('Produto desativado: ' || prod.cod\_produto);

END LOOP;

COMMIT;

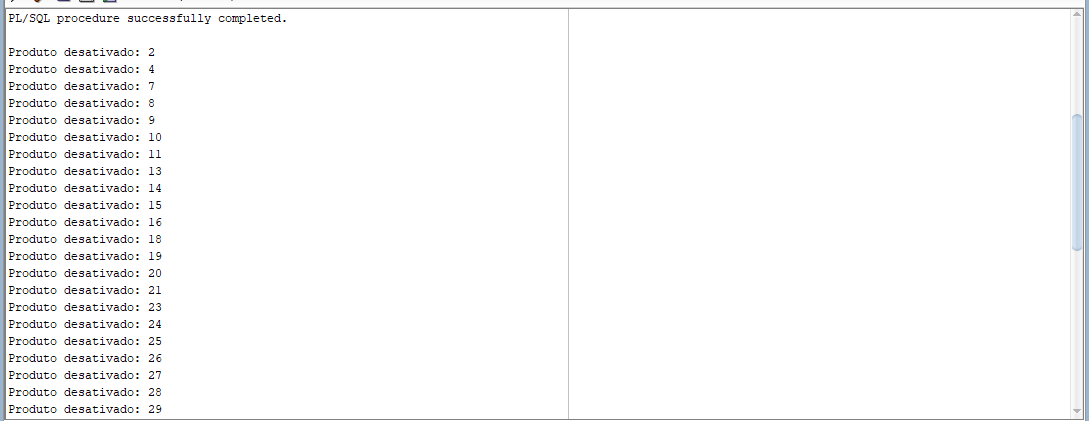
EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erro ao atualizar produtos.');

ROLLBACK;

END;



1. Crie uma procedure que para cada produto, busque as datas e quantidades dos movimentos e exiba no seguinte formato:  
    Produto: Teclado Gamer  
    -> 10/04/2025 | Entrada: 10  
    -> 12/04/2025 | Saída: -5  
    Utilize cursores aninhados.

CREATE OR REPLACE PROCEDURE sp\_relatorio\_movimentos\_produto AS

BEGIN

-- Cursor externo: lista todos os produtos

FOR prod IN (

SELECT cod\_produto, nom\_produto

FROM produto

) LOOP

-- Exibe o nome do produto

DBMS\_OUTPUT.PUT\_LINE('Produto: ' || prod.nom\_produto);

-- Cursor interno: movimentações para o produto atual

FOR mov IN (

SELECT dat\_movimento\_estoque, qtd\_movimentacao\_estoque

FROM movimento\_estoque

WHERE cod\_produto = prod.cod\_produto

ORDER BY dat\_movimento\_estoque

) LOOP

-- Define tipo da movimentação com base no sinal

DBMS\_OUTPUT.PUT\_LINE(

'-> ' || TO\_CHAR(mov.dat\_movimento\_estoque, 'DD/MM/YYYY') ||

' | ' ||

CASE

WHEN mov.qtd\_movimentacao\_estoque >= 0 THEN 'Entrada: '

ELSE 'Saída: '

END || mov.qtd\_movimentacao\_estoque

);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(''); -- linha em branco entre produtos

END LOOP;

EXCEPTION

WHEN OTHERS THEN

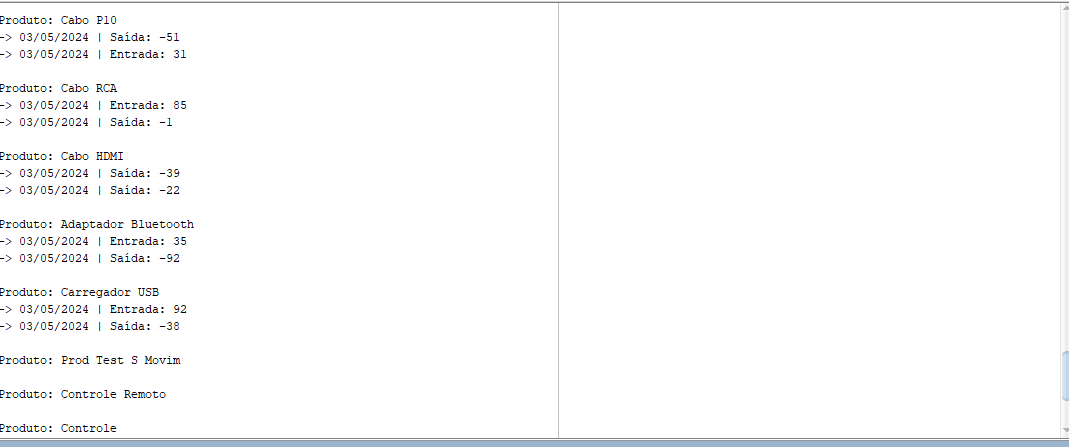
DBMS\_OUTPUT.PUT\_LINE('Erro ao gerar relatório de movimentações.');

END;

BEGIN

sp\_relatorio\_movimentos\_produto;

END;



1. Crie `sp\_relatorio\_geral\_movimento` que exiba o total movimentado por tipo de movimento (entrada/saída), inclusive aqueles sem registro associado (produto ou tipo). Use GROUP BY.

CREATE OR REPLACE PROCEDURE sp\_relatorio\_geral\_movimento AS

BEGIN

FOR reg IN (

SELECT t.DES\_TIPO\_MOVIMENTO\_ESTOQUE AS tipo\_mov,

SUM(m.QTD\_MOVIMENTACAO\_ESTOQUE) AS total

FROM tipo\_movimento\_estoque t

LEFT JOIN movimento\_estoque m ON t.COD\_TIPO\_MOVIMENTO\_ESTOQUE = m.COD\_TIPO\_MOVIMENTO\_ESTOQUE

GROUP BY t.DES\_TIPO\_MOVIMENTO\_ESTOQUE

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Tipo de Movimento: ' || reg.tipo\_mov || ' | Total Movimentado: ' || NVL(reg.total, 0));

END LOOP;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erro ao gerar relatório geral de movimentos.');

END;

BEGIN

sp\_relatorio\_geral\_movimento;

END;



1. Implemente um bloco anônimo que tenta inserir um novo produto já existente e capture o erro ORA-00001, exibindo uma mensagem customizada de violação de chave primária.

DECLARE

v\_cod\_produto produto.cod\_produto%TYPE := 1; -- codigo de um produto existente

v\_nome\_produto produto.nom\_produto%TYPE := 'Produto Duplicado';

BEGIN

INSERT INTO produto (cod\_produto, nom\_produto)

VALUES (v\_cod\_produto, v\_nome\_produto);

DBMS\_OUTPUT.PUT\_LINE('Produto inserido com sucesso.');

EXCEPTION

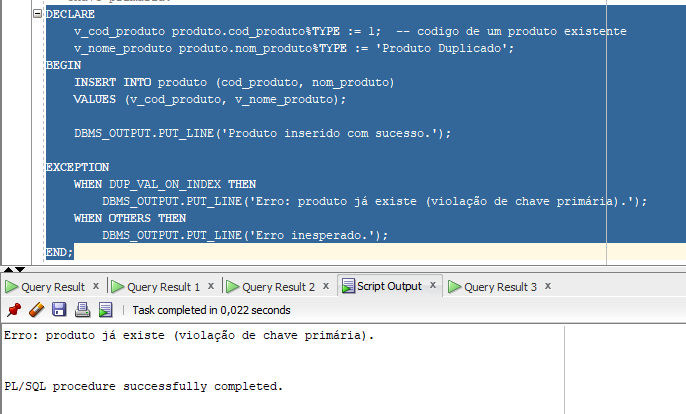
WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Erro: produto já existe (violação de chave primária).');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erro inesperado.');

END;



1. Crie `sp\_limpa\_movimentos\_antigos` que exclui movimentações com mais de 2 anos, mas faz rollback se a quantidade de registros excluídos ultrapassar 1000. Use ROLLBACK e COUNT(1).

CREATE OR REPLACE PROCEDURE sp\_limpa\_movimentos\_antigos AS

v\_qtd\_excluir NUMBER;

BEGIN

-- Verifica quantas movimentações têm mais de 2 anos

SELECT COUNT(1)

INTO v\_qtd\_excluir

FROM movimento\_estoque

WHERE dat\_movimento\_estoque < ADD\_MONTHS(SYSDATE, -24);

-- Exibe quantos seriam excluídos

DBMS\_OUTPUT.PUT\_LINE('Movimentações com mais de 2 anos: ' || v\_qtd\_excluir);

IF v\_qtd\_excluir <= 1000 THEN

-- Faz a exclusão e commita

DELETE FROM movimento\_estoque

WHERE dat\_movimento\_estoque < ADD\_MONTHS(SYSDATE, -24);

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Exclusão concluída com sucesso.');

ELSE

-- Rollback se passar do limite

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Rollback realizado. Mais de 1000 registros seriam excluídos.');

END IF;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

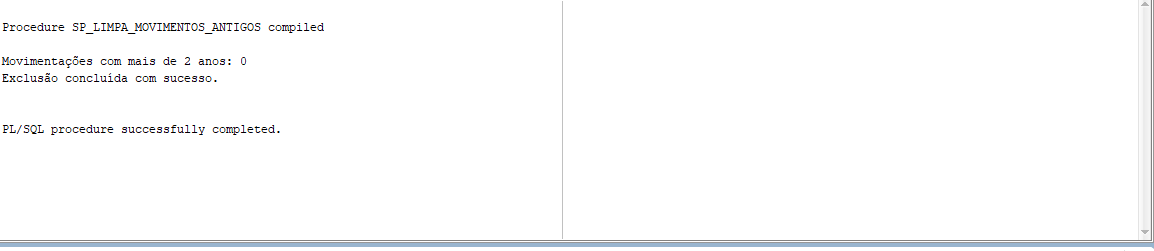
DBMS\_OUTPUT.PUT\_LINE('Erro durante o processo: ' || SQLERRM);

END;

BEGIN

sp\_limpa\_movimentos\_antigos;

END;



1. Crie a função `fn\_previsao\_movimentacao(p\_cod\_produto NUMBER) ` que, com base na média de movimentações nos últimos 3 meses, retorna uma previsão da movimentação esperada para o próximo mês.

CREATE OR REPLACE FUNCTION fn\_previsao\_movimentacao(p\_cod\_produto NUMBER)

RETURN NUMBER

IS

v\_total\_movimentado NUMBER := 0;

v\_qtd\_meses NUMBER := 3;

v\_media NUMBER := 0;

BEGIN

-- Soma a quantidade de movimentações dos últimos 3 meses

SELECT SUM(qtd\_movimentacao\_estoque)

INTO v\_total\_movimentado

FROM movimento\_estoque

WHERE cod\_produto = p\_cod\_produto

AND dat\_movimento\_estoque >= ADD\_MONTHS(SYSDATE, -v\_qtd\_meses);

-- Se houver movimentações, calcula a média

IF v\_total\_movimentado IS NOT NULL THEN

v\_media := ROUND(v\_total\_movimentado / v\_qtd\_meses, 2);

END IF;

RETURN v\_media;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0; -- Nenhuma movimentação

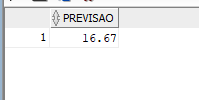
WHEN ZERO\_DIVIDE THEN

RETURN -1; -- Erro de divisão por zero

WHEN OTHERS THEN

RETURN -2; -- Erro inesperado

END;



Para todos os exercícios enviar o código sql e print de teste dos objetos.