Exercício 5 – Dashboard de Relatórios com JOINs e Tratamento de Exceções

# Função 1 – fnc\_percentual\_desconto

Crie uma função chamada `fnc\_percentual\_desconto` que, ao receber o código de um pedido, calcule o percentual de desconto aplicado sobre o valor total do pedido. A função deve utilizar `JOIN` com a tabela `item\_pedido` para validar que o pedido possui ao menos um item, e deve tratar possíveis exceções como: pedido inexistente, divisão por zero e erro genérico.

Critérios de avaliação:

* - Uso correto de JOIN
* - Retorno numérico com duas casas decimais (opcional, mas desejável)
* - Tratamento com EXCEPTION

SCRIPT:

CREATE OR REPLACE FUNCTION fnc\_percentual\_desconto (

p\_cod\_pedido NUMBER

) RETURN NUMBER IS

val\_desc NUMBER;

BEGIN

SELECT

SUM(b.val\_desconto\_item) / 100

INTO val\_desc

FROM

pedido a

INNER JOIN item\_pedido b ON a.cod\_pedido = b.cod\_pedido

WHERE a.cod\_pedido = p\_cod\_pedido;

RETURN ROUND(val\_desc, 2); -- retorna o percentual com 2 casas

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

raise\_application\_error(-20001, 'NÃO ENCONTROU NADA');

WHEN OTHERS THEN

raise\_application\_error(-20002, 'ERRO DESCONHECIDO');

END;

SELECT fnc\_percentual\_desconto(162890) FROM dual;

EXECUÇÃO:



# Função 2 – fnc\_media\_itens\_por\_pedido

Crie uma função que retorne a média de itens por pedido considerando todos os pedidos com itens registrados. Utilize `JOIN` com a tabela `historico\_pedido` para garantir que os pedidos são válidos. Implemente tratamento de erro para divisão por zero e erro genérico.

Critérios de avaliação:

* - Uso correto de agregações (COUNT, DISTINCT)
* - Uso de JOIN
* - Tratamento de exceções

SCRIPT:

CREATE OR REPLACE FUNCTION fnc\_media\_itens\_por\_pedido

RETURN NUMBER

IS

v\_total\_itens NUMBER := 0;

v\_total\_pedidos NUMBER := 0;

v\_media NUMBER := 0;

BEGIN

-- Total de itens (linhas da item\_pedido com pedido válido)

SELECT COUNT(\*) INTO v\_total\_itens

FROM item\_pedido ip

INNER JOIN historico\_pedido hp ON ip.cod\_pedido = hp.cod\_pedido;

-- Total de pedidos diferentes com pelo menos um item

SELECT COUNT(DISTINCT ip.cod\_pedido) INTO v\_total\_pedidos

FROM item\_pedido ip

INNER JOIN historico\_pedido hp ON ip.cod\_pedido = hp.cod\_pedido;

-- Se for tiver como dividir, calcula a média

IF v\_total\_pedidos > 0 THEN

v\_media := v\_total\_itens / v\_total\_pedidos;

END IF;

RETURN ROUND(v\_media, 2);

-- tratando as exceções

EXCEPTION

WHEN ZERO\_DIVIDE THEN

RETURN -1; -- erro por divisão por zero

WHEN OTHERS THEN

RETURN -2; -- erro desconhecido

END;

SELECT fnc\_media\_itens\_por\_pedido FROM dual;

EXECUÇÃO:



# Procedimento 1 – prc\_relatorio\_estoque\_produto

Implemente um procedimento que receba o código de um produto e exiba, via `DBMS\_OUTPUT`, o total de unidades movimentadas e a data da última movimentação. Utilize um `LEFT JOIN` com a tabela `produto\_composto` para mostrar que o produto pode fazer parte de composições, mesmo que não tenha. Inclua tratamento para ausência de dados e erro genérico.

Critérios de avaliação:

* - Utilização de agregações (SUM, MAX)
* - Uso de LEFT JOIN
* - Impressão no terminal com DBMS\_OUTPUT

SCRIPT:

CREATE OR REPLACE PROCEDURE prc\_relatorio\_estoque\_produto(p\_cod\_produto NUMBER) AS

v\_total\_movimentado NUMBER := 0;

v\_ultima\_data DATE;

BEGIN

-- Consulta com LEFT JOIN para garantir que o produto aparece mesmo sem composição

SELECT SUM(me.QTD\_MOVIMENTACAO\_ESTOQUE), MAX(me.DAT\_MOVIMENTO\_ESTOQUE)

INTO v\_total\_movimentado, v\_ultima\_data

FROM produto p

LEFT JOIN produto\_composto pc ON p.cod\_produto = pc.cod\_produto

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

WHERE p.cod\_produto = p\_cod\_produto;

-- Verificar se houve movimentações

IF v\_total\_movimentado IS NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Produto ' || p\_cod\_produto || ' não possui movimentações registradas.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Produto: ' || p\_cod\_produto);

DBMS\_OUTPUT.PUT\_LINE('Total movimentado: ' || v\_total\_movimentado);

DBMS\_OUTPUT.PUT\_LINE('Última movimentação: ' || TO\_CHAR(v\_ultima\_data, 'DD/MM/YYYY'));

END IF;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erro ao gerar o relatório do produto.');

END;

EXECUÇÃO:

BEGIN

prc\_relatorio\_estoque\_produto(101);

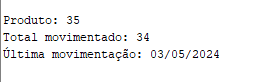
END;



BEGIN

prc\_relatorio\_estoque\_produto(35);

END;



# Procedimento 2 – prc\_relatorio\_composicao\_ativa

Crie um procedimento que, dado o código de um produto, exiba os componentes ativos que fazem parte de sua composição. Use JOIN com a tabela `movimento\_estoque` para relacionar os componentes com movimentações. Implemente o procedimento utilizando um `FOR LOOP` com `CURSOR IMPLÍCITO`, e trate erros genéricos.

Critérios de avaliação:

* - Uso de JOIN
* - Laço FOR ... IN (SELECT ...) LOOP
* - Impressão formatada com DBMS\_OUTPUT

SCRIPT:

CREATE OR REPLACE PROCEDURE prc\_relatorio\_composicao\_ativa(p\_cod\_produto NUMBER) AS

BEGIN

FOR comp IN (

SELECT pc.cod\_produto\_relacionado,

pc.COD\_PRODUTO ,

me.QTD\_MOVIMENTACAO\_ESTOQUE,

me.DAT\_MOVIMENTO\_ESTOQUE

FROM produto\_composto pc

INNER JOIN movimento\_estoque me

ON pc.cod\_produto = me.cod\_produto

WHERE pc.cod\_produto\_relacionado = p\_cod\_produto

AND pc.sta\_ativo = 'S'

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Composição Ativa: Produto Principal = ' || comp.cod\_produto\_relacionado ||

' | Componente = ' || comp.cod\_produto ||

' | Qtde Movimentada = ' || comp.QTD\_MOVIMENTACAO\_ESTOQUE ||

' | Data = ' || TO\_CHAR(comp.DAT\_MOVIMENTO\_ESTOQUE , 'DD/MM/YYYY')

);

END LOOP;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erro ao exibir a composição ativa do produto.');

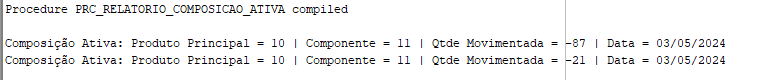
END;

EXECUÇÃO:

BEGIN

prc\_relatorio\_composicao\_ativa(10);

END;



# Procedimento 3 – prc\_relatorio\_pedido

Desenvolva um procedimento que exiba informações detalhadas de um pedido, como: valor total, desconto e status de entrega (`ENTREGUE` ou `PENDENTE`). O procedimento deve aceitar o código do pedido como parâmetro e utilizar JOIN com a tabela `item\_pedido` para garantir que o pedido possua itens. Inclua tratamento para pedido inexistente e erros genéricos.

Critérios de avaliação:

* - Lógica condicional (IF/ELSE) para status
* - Uso de JOIN
* - Uso correto de DBMS\_OUTPUT.PUT\_LINE

SCRIPT:

CREATE OR REPLACE PROCEDURE prc\_relatorio\_pedido(p\_cod\_pedido NUMBER) AS

v\_val\_total pedido.val\_total\_pedido%TYPE;

v\_val\_desconto pedido.val\_desconto%TYPE;

v\_status pedido.status%TYPE;

v\_qtd\_itens NUMBER;

BEGIN

-- Verifica se o pedido tem pelo menos um item usando JOIN com item\_pedido

SELECT COUNT(\*) INTO v\_qtd\_itens

FROM pedido p

JOIN item\_pedido ip ON p.cod\_pedido = ip.cod\_pedido

WHERE p.cod\_pedido = p\_cod\_pedido;

IF v\_qtd\_itens = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('O pedido não possui itens registrados.');

RETURN;

END IF;

-- Busca os dados do pedido

SELECT val\_total\_pedido, val\_desconto, status

INTO v\_val\_total, v\_val\_desconto, v\_status

FROM pedido

WHERE cod\_pedido = p\_cod\_pedido;

-- Exibe os dados

DBMS\_OUTPUT.PUT\_LINE('Pedido: ' || p\_cod\_pedido);

DBMS\_OUTPUT.PUT\_LINE('Valor com desconto: R$' || v\_val\_total);

DBMS\_OUTPUT.PUT\_LINE('Desconto aplicado: R$' || v\_val\_desconto);

-- Interpreta o status

IF LOWER(v\_status) = 'concluído' THEN

DBMS\_OUTPUT.PUT\_LINE('Status: ENTREGUE');

ELSIF LOWER(v\_status) = 'pendente' THEN

DBMS\_OUTPUT.PUT\_LINE('Status: PENDENTE');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Status: OUTRO');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Pedido não encontrado.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erro ao consultar o pedido.');

END;

EXECUÇÃO:

BEGIN

prc\_relatorio\_pedido(162890);

END;

