CURSORES

```
SELECT last_name
  INTO l_last_name
  FROM employees
 WHERE employee_id = 138;
```

-- retorna uma única linha

```
SELECT last_name
  INTO l_last_name
  FROM employees
```

-- retorna várias linhas, TOO_MANY_ROWS exception raised

Ciclo de vida de um cursor:

1. **DECLARE**
   CURSOR <cursor_name>[(parameters_list)] IS <select statement>;
2. **OPEN**
   OPEN <cursor_name>[(parameters_list)];
3. **FETCH, num ciclo LOOP, FOREACH**
   FETCH <cursor_name> INTO <variable>;
4. **CLOSE**
   CLOSE <cursor_name>;

EXCEÇÕES

Ciclo de vida de uma exceção:

1. **DECLARE**
   <exception_name> EXCEPTION;
2. **RAISE**
   RAISE <exception_name>;
3. **EXCEPTION HANDLING**
   EXCEPTION
   WHEN < exception_name > THEN <exception_handling_routine>;

1 - Listar para todos os produtos existentes num dado armazém a quantidade a encomendar para repor stocks. É política da empresa encomendar a quantidade necessária de cada produto para repor os stocks a um nível de referência, ou seja, encomendar uma quantidade igual à diferença entre o nível de referência e a quantidade em stock à data.

O nível de referência é igual a 12 vezes o stock mínimo do produto.

Para um produto que tenha um stock de 100 unidades e um stock mínimo de 10 unidades devem encomendar-se 20 unidades: 12*10 – 100 = 20.

Gerar exceções para “armazém inexistente” e para armazém em que não seja necessário encomendar nenhum produto, “vendas_reduzidas”.

2 - Listar quantidade de cada produto existente em cada armazéns. Gerar uma exceção “armazém_sem_produtos” se existir algum armazém sem qualquer produto.

```
-- tabelas
drop table ArmazemProduto;
drop table Produto;
drop table Armazem;

create table Produto(cod_produto integer primary key);
create table Armazem(cod_armazem integer primary key);
create table ArmazemProduto(
        cod_armazem integer,
        cod_produto integer,
        stock integer default 0 not null,
        stock_minimo integer default 0 not null,
        corredor varchar(10),
        prateleira varchar(10),
        constraint pk_ArmazemProduto primary key(cod_armazem, cod_produto),
        constraint fk_ArmazemProduto_Armazem foreign key (cod_armazem)
references Armazem(cod_armazem),
        constraint fk_ArmazemProduto_Produto foreign key (cod_produto)
references Produto(cod_produto)
);



-- dados para teste
insert into Armazem values(1);
insert into Armazem values(2);
insert into Armazem values(3);
insert into Armazem values(4);
insert into Armazem values(5);

insert into Produto values(1);
insert into Produto values(2);
insert into Produto values(3);
```

```
insert into Produto values(4);
insert into Produto values(5);
insert into Produto values(6);
insert into Produto values(7);
insert into Produto values(8);

insert into ArmazemProduto values(1, 1, 10, 5, 'A23', '2A');
insert into ArmazemProduto values(1, 2, 18, 10, 'A81', '1S');
insert into ArmazemProduto values(1, 3, 8, 1, 'A23', '1W');
insert into ArmazemProduto values(1, 4, 121, 50, 'A81', '2A');
insert into ArmazemProduto values(1, 5, 60, 5, 'A23', '2S');
insert into ArmazemProduto values(1, 6, 154, 10, 'A23', '2B');
insert into ArmazemProduto values(1, 7, 25, 20, 'A23', '2C');
insert into ArmazemProduto values(1, 8, 113, 10, 'A23', '2S');
insert into ArmazemProduto values(2, 2, 26, 5, '2B98', '22EW');
insert into ArmazemProduto values(2, 4, 87, 10, '2B12', '22B2');
insert into ArmazemProduto values(3, 3, 44, 12, '3B15', '32B3');
insert into ArmazemProduto values(3, 1, 110, 5, '3C13', '42B2');
insert into ArmazemProduto values(4, 7, 120, 10, '4C65', '42NW');
insert into ArmazemProduto values(4, 6, 90, 5, '4B35', '42SS');



-- procedimento
CREATE OR REPLACE PROCEDURE repor_stocks(armazem_a_reabastecer
ArmazemProduto.cod_armazem%type) AS
   qtd_a_encomendar ArmazemProduto.stock%type;
   c_produto ArmazemProduto.cod_produto%type;
   c_stock ArmazemProduto.stock%type;
   c_stock_minimo ArmazemProduto.stock_minimo%type;
   CURSOR c_produtos(armazem ArmazemProduto.cod_armazem%type) IS
      SELECT cod_produto, stock, stock_minimo
        FROM ArmazemProduto
        WHERE cod_armazem = armazem;
   armazem_inexistente EXCEPTION;
   low_sales EXCEPTION;
   vendas_reduzidas BOOLEAN := TRUE;
   dummy Armazem.cod_armazem%type;
BEGIN
   SELECT COUNT(*) INTO dummy FROM Armazem WHERE cod_armazem =
armazem_a_reabastecer;
   IF dummy = 0 THEN
        RAISE armazem_inexistente;
   ELSE
        OPEN c_produtos(armazem_a_reabastecer);
        LOOP
                FETCH c_produtos INTO c_produto, c_stock, c_stock_minimo;
                EXIT WHEN c_produtos%notfound;
                qtd_a_encomendar := 12 * c_stock_minimo - c_stock;
                IF qtd_a_encomendar > 0 THEN
                        Vendas_reduzidas := FALSE;
                        dbms_output.put_line(c_produto || ' - ' ||
qtd_a_encomendar);
                END IF;
        END LOOP;
        CLOSE c_produtos;
        IF vendas_reduzidas THEN
```

```
                RAISE low_sales;
        END IF;
    END IF;

    EXCEPTION
        WHEN armazem_inexistente THEN
                dbms_output.put_line('Não existe nenhum armazém com o código '
|| armazem_a_reabastecer);
        WHEN low_sales THEN
                dbms_output.put_line('O armazém ' || armazem_a_reabastecer ||
' tem vendas reduzidas.');
        WHEN others THEN
                dbms_output.put_line('Erro!');

END;
/



-- teste
SET serveroutput ON;
EXEC repor_stocks(1);
-- resultado esperado com os dados de teste
-- 1 - 50
-- 2 - 102
-- 3 - 4
-- 4 - 479
-- 7 - 215
-- 8 - 7

EXEC repor_stocks(2);
-- resultado esperado com os dados de teste
-- 2 - 34
-- 4 - 33

EXEC repor_stocks(3);
-- resultado esperado com os dados de teste
-- 3 - 100

EXEC repor_stocks(4);
-- resultado esperado com os dados de teste
-- O armazém 4 não tem stock de nenhum produto.

EXEC repor_stocks(5);
-- resultado esperado com os dados de teste
-- O armazém 5 não tem stock de nenhum produto.

EXEC repor_stocks(100);
-- resultado esperado com os dados de teste
-- Não existe nenhum armazém com o código 100
```