PL8 PL9 - PL/SQL – FUNÇÕES, BLOCOS ANÓNIMOS, CURSORES E EXCEÇÕES

Category	Exercícios
Files	INSERTS.sql CREATES.sql PL08-PL_SQL-Funcoes-Ex Livraria.pdf
URL	https://moodle.isep.ipp.pt/pluginfile.php/301054/course/section/31926/PL08.S2-INSERT.sql

▼ Create

```
-- ** eliminar tabelas se existentes **
-- CASCADE CONSTRAINTS para eliminar as restri��es de integridade da
-- PURGE elimina a tabela da base de dados e da "reciclagem"
DROP TABLE cliente              CASCADE CONSTRAINTS PURGE;
DROP TABLE codigoPostal         CASCADE CONSTRAINTS PURGE;
DROP TABLE cartaoCliente        CASCADE CONSTRAINTS PURGE;
DROP TABLE venda                CASCADE CONSTRAINTS PURGE;
DROP TABLE livro                CASCADE CONSTRAINTS PURGE;
DROP TABLE categoria            CASCADE CONSTRAINTS PURGE;
DROP TABLE edicaoLivro          CASCADE CONSTRAINTS PURGE;
DROP TABLE idioma               CASCADE CONSTRAINTS PURGE;
DROP TABLE autor                CASCADE CONSTRAINTS PURGE;
DROP TABLE autorEdicaoLivro     CASCADE CONSTRAINTS PURGE;
DROP TABLE nacionalidadeAutor   CASCADE CONSTRAINTS PURGE;
DROP TABLE pais                 CASCADE CONSTRAINTS PURGE;
DROP TABLE editora              CASCADE CONSTRAINTS PURGE;
DROP TABLE precoEdicaoLivro     CASCADE CONSTRAINTS PURGE;

-- ## tabela cliente ##
CREATE TABLE cliente (
    nifCliente      NUMERIC(9)  CONSTRAINT pkClienteNifCliente PRIMARY
    codPostal       CHAR(8)     CONSTRAINT nnClientesCodPostal NOT NUL
    nome            VARCHAR(60) CONSTRAINT nnClienteNome NOT NULL,
    dataNascimento  DATE,
    morada          VARCHAR(50) CONSTRAINT nnClienteMorada NOT NULL,
    nrTelemovel     NUMERIC(9)  CONSTRAINT ckClienteNrTelemovel CHECK(
);

-- ## tabela codigoPostal ##
CREATE TABLE codigoPostal (
```

```
    codPostal   CHAR(8)      CONSTRAINT pkCodigoPostalCodPostal PRIMAR
    localidade  VARCHAR(25)  CONSTRAINT nnCodigoPostalLocalidade NOT N

    CONSTRAINT ckCodigoPostalCodPostal CHECK(REGEXP_LIKE(codPostal, '^
);

-- ## tabela cartaoCliente ##
CREATE TABLE cartaoCliente (
    nrCartao       INTEGER          CONSTRAINT pkCartaoClienteNrCartao
    nifCliente     NUMERIC(9)       CONSTRAINT nnCartaoClientesNif NOT
                                    CONSTRAINT ckCartaoClienteNif CHEC
                                    CONSTRAINT ukCartaoCliente UNIQUE,
    dataAdesao     DATE             CONSTRAINT nnCartaoClienteDataAdes
    saldoAtual     NUMERIC(5,2),
    saldoAcumulado NUMERIC(10,2)
);

-- ## tabela venda ##
CREATE TABLE venda (
    nrVenda     INTEGER     CONSTRAINT pkVendaNrVenda PRIMARY KEY,
    nifCliente  INTEGER     CONSTRAINT nnVendaNifCliente NOT NULL,
    isbn        CHAR(14)    CONSTRAINT nnVendaIsbn NOT NULL,
    dataHora    DATE        CONSTRAINT nnVendaDataHora NOT NULL,
    quantidade  INTEGER     CONSTRAINT nnVendaQuantidade NOT NULL
);

-- ## tabela livro ##
CREATE TABLE livro (
    idLivro      INTEGER     CONSTRAINT pkLivroIdLivro PRIMARY KEY,
    idCategoria  INTEGER     CONSTRAINT nnLivroIdCategoria NOT NULL,
    titulo       VARCHAR(50) CONSTRAINT nnLivroTitulo NOT NULL
);

-- ## tabela categoria ##
CREATE TABLE categoria (
    idCategoria   INTEGER     CONSTRAINT pkCategoriaIdCategoria PRIMAR
    designacao    VARCHAR(20) CONSTRAINT nnCategoriaDesignacao NOT NUL
);

-- ## tabela edicaoLivro ##
CREATE TABLE edicaoLivro (
    isbn        CHAR(14)            CONSTRAINT pkEdicaoLivroIsbn PRIMA
    idLivro     INTEGER             CONSTRAINT nnEdicaoLivroIdLivro NO
    idEditora   INTEGER             CONSTRAINT nnEdicaoLivroIdEditora
    nrEdicao    INTEGER             CONSTRAINT nnEdicaoLivroNrEdicao N
    codIdioma   CHAR(2)             CONSTRAINT nnEdicaoLivroCodIdioma
```

```
    mesEdicao   NUMERIC(2)          CONSTRAINT nnEdicaoLivroMesEdicao
                                    CONSTRAINT ckEdicaoLivroMesEdicao
    anoEdicao   NUMERIC(4)          CONSTRAINT nnEdicaoLivroAnoEdicao
    stockMin    INTEGER DEFAULT 10  CONSTRAINT nnEdicaoLivroStockMin N
    stock       INTEGER             CONSTRAINT nnEdicaoLivroStock NOT

    CONSTRAINT ckEdicaoLivroIsbn CHECK(REGEXP_LIKE(isbn, '^\d{3}-\d{10
);

-- ## tabela idioma ##
CREATE TABLE idioma (
    codIdioma   CHAR(2)        CONSTRAINT pkIdiomaCodIdioma PRIMARY KE
    designacao  VARCHAR(20)    CONSTRAINT nnIdiomaDesignacao NOT NULL
);

-- ## tabela autor ##
CREATE TABLE autor (
    idAutor INTEGER     CONSTRAINT pkAutorIdAutor PRIMARY KEY,
    nome    VARCHAR(35) CONSTRAINT nnAutorNome NOT NULL
);

-- ## tabela autorEdicaoLivro ##
CREATE TABLE autorEdicaoLivro (
    isbn    CHAR(14),
    idAutor INTEGER,

    CONSTRAINT pkAutorEdicaoLivroIsbnIdAutor PRIMARY KEY (isbn, idAuto
);

-- ## tabela nacionalidadeAutor ##
CREATE TABLE nacionalidadeAutor (
    idAutor  INTEGER,
    codPais  CHAR(2),

    CONSTRAINT pkNacionalidadeAutorIdAutorCodPais PRIMARY KEY(idAutor,
);

-- ## tabela pais ##
CREATE TABLE pais (
    codPais  CHAR(2)        CONSTRAINT pkPaisCodPais PRIMARY KEY,
    nome     VARCHAR(30)    CONSTRAINT nnPaisNome NOT NULL
);

-- ## tabela editora ##
CREATE TABLE editora (
    idEditora   INTEGER     CONSTRAINT pkEditorIdEditora PRIMARY KEY,
```

```
    nome        VARCHAR(35) CONSTRAINT nnEditorNome NOT NULL,
    codPais     CHAR(2)     CONSTRAINT nnEditorCodPais NOT NULL
);

-- ## tabela precoEdicaoLivro ##
CREATE TABLE precoEdicaoLivro (
    isbn        CHAR(14),
    dataInicio  DATE            CONSTRAINT nnPrecoEdicaoLivroDataInici
    preco       NUMERIC(5,2)    CONSTRAINT nnPrecoEdicaoLivroPreco NOT
                                CONSTRAINT ckPrecoEdicaoLivroPreco CHE

    CONSTRAINT pkPrecoEdicaoLivroIsbnDataInicio PRIMARY KEY (isbn, dat
);

-- ** alterar tabelas para defini��o de chaves estrangeiras **
ALTER TABLE cliente             ADD CONSTRAINT fkClienteCodPostal
ALTER TABLE cartaoCliente       ADD CONSTRAINT fkCartaoClienteNifClien
ALTER TABLE venda               ADD CONSTRAINT fkVendaNifCliente
ALTER TABLE venda               ADD CONSTRAINT fkVendaIsbn
ALTER TABLE livro               ADD CONSTRAINT fkLivroIdCategoria
ALTER TABLE edicaoLivro         ADD CONSTRAINT fkEdicaoLivroIdLivro
ALTER TABLE edicaoLivro         ADD CONSTRAINT fkEdicaoLivroCodIdioma
ALTER TABLE edicaoLivro         ADD CONSTRAINT fkEdicaoLivroIdEditora
ALTER TABLE autorEdicaoLivro    ADD CONSTRAINT fkAutoresEdicaoLivroIsb
ALTER TABLE autorEdicaoLivro    ADD CONSTRAINT fkAutoresEdicaoLivroIdA
ALTER TABLE nacionalidadeAutor  ADD CONSTRAINT fkNacionalidadeAutorIdA
ALTER TABLE nacionalidadeAutor  ADD CONSTRAINT fkNacionalidadeAutorCod
ALTER TABLE editora             ADD CONSTRAINT fkEditoraCodPais
ALTER TABLE precoEdicaoLivro    ADD CONSTRAINT fkPrecoEdicaoLivroIsbn

-- ** guardar em DEFINITIVO as altera��es na base de dados, se a op�
-- COMMIT;
```

▼ Inserts

```
-- ** inserir dados nas tabelas **

-- ## tabela Autor ##
INSERT INTO autor(idAutor, nome) VALUES (100, 'John');
INSERT INTO autor(idAutor, nome) VALUES (101, 'Sofia');
INSERT INTO autor(idAutor, nome) VALUES (102, 'Socrates');
INSERT INTO autor(idAutor, nome) VALUES (103, 'Gabriel');
INSERT INTO autor(idAutor, nome) VALUES (104, 'Judite');
INSERT INTO autor(idAutor, nome) VALUES (105, 'Steven Feuerstein');
INSERT INTO autor(idAutor, nome) VALUES (106, 'Bill Pribyl');
```

```
INSERT INTO autor(idAutor, nome) VALUES (107, 'Luis Damas');
INSERT INTO autor(idAutor, nome) VALUES (108, 'Feliz Gouveia');
INSERT INTO autor(idAutor, nome) VALUES (109, 'Thomas Connolly');
INSERT INTO autor(idAutor, nome) VALUES (110, 'Carolyn Begg');
INSERT INTO autor(idAutor, nome) VALUES (111, 'Deborah Perry Piscione'

-- ## tabela Categoria ##
INSERT INTO categoria(idCategoria, designacao) VALUES (1, 'Economia');
INSERT INTO categoria(idCategoria, designacao) VALUES (2, 'Filosofia')
INSERT INTO categoria(idCategoria, designacao) VALUES (3, 'Historia');
INSERT INTO categoria(idCategoria, designacao) VALUES (4, 'Informatica

-- ## tabela CodigoPostal ##
INSERT INTO codigoPostal(codPostal, localidade) VALUES ('4200-197', 'P
INSERT INTO codigoPostal(codPostal, localidade) VALUES ('4460-393', 'S
INSERT INTO codigoPostal(codPostal, localidade) VALUES ('4450-282', 'M
INSERT INTO codigoPostal(codPostal, localidade) VALUES ('4420-601', 'G
INSERT INTO codigoPostal(codPostal, localidade) VALUES ('4445-273', 'E
INSERT INTO codigoPostal(codPostal, localidade) VALUES ('4444-909', 'V
INSERT INTO codigoPostal(codPostal, localidade) VALUES ('4500-527', 'E
INSERT INTO codigoPostal(codPostal, localidade) VALUES ('4400-129', 'V

-- ## tabela Cliente ##
INSERT INTO cliente(nifCliente, codPostal, nome, dataNascimento, morad
INSERT INTO cliente(nifCliente, codPostal, nome, dataNascimento, morad
INSERT INTO cliente(nifCliente, codPostal, nome, dataNascimento, morad
INSERT INTO cliente(nifCliente, codPostal, nome, dataNascimento, morad
INSERT INTO cliente(nifCliente, codPostal, nome, dataNascimento, morad
INSERT INTO cliente(nifCliente, codPostal, nome, dataNascimento, morad
INSERT INTO cliente(nifCliente, codPostal, nome, dataNascimento, morad
INSERT INTO cliente(nifCliente, codPostal, nome, dataNascimento, morad
INSERT INTO cliente(nifCliente, codPostal, nome, dataNascimento, morad

-- ## tabela CartaoCliente ##
INSERT INTO cartaoCliente(nrCartao, nifCliente, dataAdesao, saldoAtual
INSERT INTO cartaoCliente(nrCartao, nifCliente, dataAdesao, saldoAtual
INSERT INTO cartaoCliente(nrCartao, nifCliente, dataAdesao, saldoAtual
INSERT INTO cartaoCliente(nrCartao, nifCliente, dataAdesao, saldoAtual
INSERT INTO cartaoCliente(nrCartao, nifCliente, dataAdesao, saldoAtual

-- ## tabela Pais ##
INSERT INTO pais(codPais, nome) VALUES ('DE','Alemanha');
INSERT INTO pais(codPais, nome) VALUES ('BR','Brasil');
INSERT INTO pais(codPais, nome) VALUES ('US','Estados Unidos da Americ
INSERT INTO pais(codPais, nome) VALUES ('ES','Espanha');
INSERT INTO pais(codPais, nome) VALUES ('FR','França');
```

```
INSERT INTO pais(codPais, nome) VALUES ('PT','Portugal');
INSERT INTO pais(codPais, nome) VALUES ('GB','Reino Unido');
INSERT INTO pais(codPais, nome) VALUES ('GR','Grecia');

-- ## tabela NacionalidadeAutor ##
INSERT INTO nacionalidadeAutor(idAutor, codPais) VALUES (100, 'US');
INSERT INTO nacionalidadeAutor(idAutor, codPais) VALUES (101, 'PT');
INSERT INTO nacionalidadeAutor(idAutor, codPais) VALUES (102, 'GR');
INSERT INTO nacionalidadeAutor(idAutor, codPais) VALUES (103, 'BR');
INSERT INTO nacionalidadeAutor(idAutor, codPais) VALUES (103, 'PT');
INSERT INTO nacionalidadeAutor(idAutor, codPais) VALUES (104, 'PT');
INSERT INTO nacionalidadeAutor(idAutor, codPais) VALUES (105, 'US');
INSERT INTO nacionalidadeAutor(idAutor, codPais) VALUES (106, 'US');
INSERT INTO nacionalidadeAutor(idAutor, codPais) VALUES (107, 'PT');
INSERT INTO nacionalidadeAutor(idAutor, codPais) VALUES (108, 'PT');
INSERT INTO nacionalidadeAutor(idAutor, codPais) VALUES (109, 'GB');
INSERT INTO nacionalidadeAutor(idAutor, codPais) VALUES (110, 'GB');
INSERT INTO nacionalidadeAutor(idAutor, codPais) VALUES (111, 'US');

-- ## tabela Editora ##
INSERT INTO editora(idEditora, nome, codPais) VALUES (1400, 'Livros do
INSERT INTO editora(idEditora, nome, codPais) VALUES (1500, 'Bertrand'
INSERT INTO editora(idEditora, nome, codPais) VALUES (1600, 'McGraw-Hi
INSERT INTO editora(idEditora, nome, codPais) VALUES (1700, 'Prentice
INSERT INTO editora(idEditora, nome, codPais) VALUES (1800, 'O''Reilly
INSERT INTO editora(idEditora, nome, codPais) VALUES (1900, 'FCA', 'PT
INSERT INTO editora(idEditora, nome, codPais) VALUES (2000, 'Pearson',
INSERT INTO editora(idEditora, nome, codPais) VALUES (2100, 'Addison-W
INSERT INTO editora(idEditora, nome, codPais) VALUES (2200, 'St. Marti

-- ## tabela Livro ##
INSERT INTO livro(idLivro, idCategoria, titulo) VALUES (200, 2, 'Etica
INSERT INTO livro(idLivro, idCategoria, titulo) VALUES (201, 1, 'Secre
INSERT INTO livro(idLivro, idCategoria, titulo) VALUES (202, 1, 'Bitco
INSERT INTO livro(idLivro, idCategoria, titulo) VALUES (203, 4, 'Oracl
INSERT INTO livro(idLivro, idCategoria, titulo) VALUES (204, 2, 'Exist
INSERT INTO livro(idLivro, idCategoria, titulo) VALUES (205, 4, 'Datab
INSERT INTO livro(idLivro, idCategoria, titulo) VALUES (206, 2, 'Logic
INSERT INTO livro(idLivro, idCategoria, titulo) VALUES (207, 1, 'Empre
INSERT INTO livro(idLivro, idCategoria, titulo) VALUES (208, 4, 'Funda
INSERT INTO livro(idLivro, idCategoria, titulo) VALUES (209, 4, 'SQL')

-- ## tabela Idioma ##
INSERT INTO idioma(codIdioma, designacao) VALUES ('PT', 'Portugu�s');
INSERT INTO idioma(codIdioma, designacao) VALUES ('EN', 'Ingl�s');
INSERT INTO idioma(codIdioma, designacao) VALUES ('FR', 'Franc�s');
```

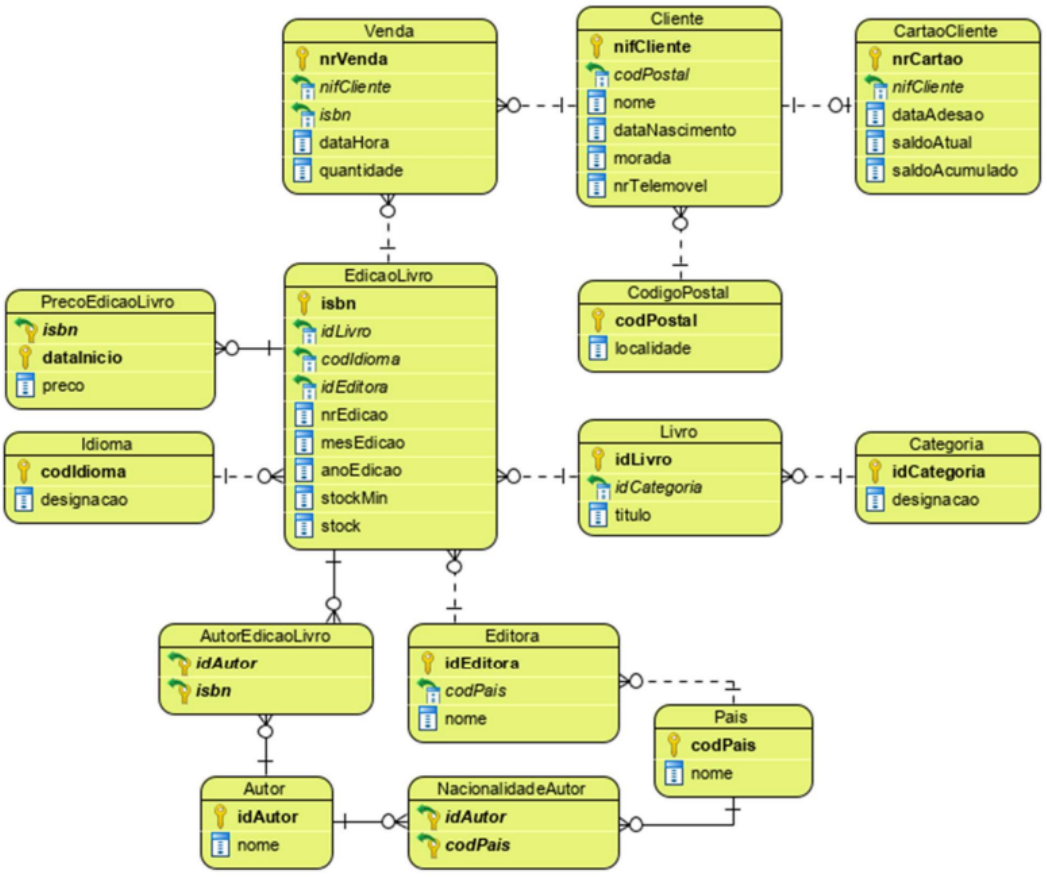
```
INSERT INTO idioma(codIdioma, designacao) VALUES ('ES', 'Espanhol');

-- ## tabela EdicaoLivro ##
INSERT INTO edicaoLivro(isbn, idLivro, idEditora, nrEdicao, codIdioma,
INSERT INTO edicaoLivro(isbn, idLivro, idEditora, nrEdicao, codIdioma,
INSERT INTO edicaoLivro(isbn, idLivro, idEditora, nrEdicao, codIdioma,
INSERT INTO edicaoLivro(isbn, idLivro, idEditora, nrEdicao, codIdioma,
INSERT INTO edicaoLivro(isbn, idLivro, idEditora, nrEdicao, codIdioma,
INSERT INTO edicaoLivro(isbn, idLivro, idEditora, nrEdicao, codIdioma,
INSERT INTO edicaoLivro(isbn, idLivro, idEditora, nrEdicao, codIdioma,
INSERT INTO edicaoLivro(isbn, idLivro, idEditora, nrEdicao, codIdioma,
INSERT INTO edicaoLivro(isbn, idLivro, idEditora, nrEdicao, codIdioma,
INSERT INTO edicaoLivro(isbn, idLivro, idEditora, nrEdicao, codIdioma,
INSERT INTO edicaoLivro(isbn, idLivro, idEditora, nrEdicao, codIdioma,
INSERT INTO edicaoLivro(isbn, idLivro, idEditora, nrEdicao, codIdioma,
INSERT INTO edicaoLivro(isbn, idLivro, idEditora, nrEdicao, codIdioma,
INSERT INTO edicaoLivro(isbn, idLivro, idEditora, nrEdicao, codIdioma,
INSERT INTO edicaoLivro(isbn, idLivro, idEditora, nrEdicao, codIdioma,
INSERT INTO edicaoLivro(isbn, idLivro, idEditora, nrEdicao, codIdioma,
INSERT INTO edicaoLivro(isbn, idLivro, idEditora, nrEdicao, codIdioma,

-- ## tabela PrecoEdicaoLivro ##
INSERT INTO precoEdicaoLivro(isbn, dataInicio, preco) VALUES ('500-123
INSERT INTO precoEdicaoLivro(isbn, dataInicio, preco) VALUES ('978-113
INSERT INTO precoEdicaoLivro(isbn, dataInicio, preco) VALUES ('500-121
INSERT INTO precoEdicaoLivro(isbn, dataInicio, preco) VALUES ('978-144
INSERT INTO precoEdicaoLivro(isbn, dataInicio, preco) VALUES ('500-123
INSERT INTO precoEdicaoLivro(isbn, dataInicio, preco) VALUES ('500-123
INSERT INTO precoEdicaoLivro(isbn, dataInicio, preco) VALUES ('500-123
INSERT INTO precoEdicaoLivro(isbn, dataInicio, preco) VALUES ('500-123
INSERT INTO precoEdicaoLivro(isbn, dataInicio, preco) VALUES ('500-123
INSERT INTO precoEdicaoLivro(isbn, dataInicio, preco) VALUES ('500-123
INSERT INTO precoEdicaoLivro(isbn, dataInicio, preco) VALUES ('500-123
INSERT INTO precoEdicaoLivro(isbn, dataInicio, preco) VALUES ('978-059
INSERT INTO precoEdicaoLivro(isbn, dataInicio, preco) VALUES ('978-059
INSERT INTO precoEdicaoLivro(isbn, dataInicio, preco) VALUES ('500-127
INSERT INTO precoEdicaoLivro(isbn, dataInicio, preco) VALUES ('978-972
INSERT INTO precoEdicaoLivro(isbn, dataInicio, preco) VALUES ('978-972
INSERT INTO precoEdicaoLivro(isbn, dataInicio, preco) VALUES ('978-972

-- ## tabela AutorEdicaoLivro ##
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('500-1234567891',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('500-1234567891',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('978-1137279170',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('500-1211111191',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('978-1449777452',
```

```
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('978-1449777452',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('978-0596556464',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('978-0596003814',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('978-0596003814',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('500-1234447891',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('500-1234222891',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('500-1234222891',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('500-1234522891',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('500-1234522891',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('500-1234666891',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('500-1234567991',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('500-1234533891',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('500-1277777891',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('978-9727227990',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('978-9727228394',
INSERT INTO autorEdicaoLivro(isbn, idAutor) VALUES ('978-9727224432',

-- ## tabela Venda ##
INSERT INTO venda(nrVenda, nifCliente, isbn, dataHora, quantidade) VAL
INSERT INTO venda(nrVenda, nifCliente, isbn, dataHora, quantidade) VAL
INSERT INTO venda(nrVenda, nifCliente, isbn, dataHora, quantidade) VAL
INSERT INTO venda(nrVenda, nifCliente, isbn, dataHora, quantidade) VAL
INSERT INTO venda(nrVenda, nifCliente, isbn, dataHora, quantidade) VAL
INSERT INTO venda(nrVenda, nifCliente, isbn, dataHora, quantidade) VAL
INSERT INTO venda(nrVenda, nifCliente, isbn, dataHora, quantidade) VAL
INSERT INTO venda(nrVenda, nifCliente, isbn, dataHora, quantidade) VAL
INSERT INTO venda(nrVenda, nifCliente, isbn, dataHora, quantidade) VAL
INSERT INTO venda(nrVenda, nifCliente, isbn, dataHora, quantidade) VAL
INSERT INTO venda(nrVenda, nifCliente, isbn, dataHora, quantidade) VAL
INSERT INTO venda(nrVenda, nifCliente, isbn, dataHora, quantidade) VAL
INSERT INTO venda(nrVenda, nifCliente, isbn, dataHora, quantidade) VAL
INSERT INTO venda(nrVenda, nifCliente, isbn, dataHora, quantidade) VAL

-- ** guardar em DEFINITIVO as altera��es na base de dados, se a op�
-- COMMIT;
```

▼ Ficha

https://prod-files-secure.s3.us-west-2.amazonaws.com/3b829444-cd0b-4b9b-a3c6-750b0f6c18d6/9e35ad4a-0d6f-4075-8682-8e88e4098d11/PL08-PL_SQL-Funcoes-Ex_Livraria.pdf

▼ Modelo Relacional



EXERCÍCIOS

▼ 3)

Criar um script com código PL/SQL para implementar uma função, designada fncTotalClientes, para retornar o número total de clientes. Caso não existam clientes registados, a função deve retornar o valor NULL. Testar adequadamente a função implementada, através de blocos anónimos. Para visualizar o resultado, executar o comando `SET SERVEROUTPUT ON`

```
CREATE OR REPLACE FUNCTION fncTotalClientes
RETURN NUMBER IS
    totClientes NUMBER := 0;

BEGIN

    SELECT COUNT(nifCliente) INTO totClientes FROM cliente;

    IF totClientes = 0 THEN
        RETURN NULL;
    END IF;
```

```
    RETURN totClientes;

END;
/
```

```
DECLARE
    totalClientes NUMBER;
BEGIN
    totalClientes := fncTotalClientes();
    DBMS_OUTPUT.PUT_LINE('Total de Clientes: ' || totalClientes);
END;
```

▼ 4)

Criar um novo script PL/SQL para implementar uma função, designada fncTemLivrosEditora, para verificar se existem livros de uma dada editora em stock. A função deve receber, por parâmetro, o identificador da editora e tem de retornar um valor booleano, true ou false. Se o parâmetro fornecido for inválido, a função deve retornar o valor NULL, usando o mecanismo de exceções. Testar adequadamente a função implementada, através de blocos anónimos.

```
CREATE OR REPLACE FUNCTION fncTemLivrosEditora(idEditora NUMBER)
RETURN BOOLEAN IS
    returnValue BOOLEAN := FALSE;

    naoExisteLivroEitora EXCEPTION;

BEGIN
    -- verificar se identificador da editora é valido
    IF idEditora IS NULL THEN
        -- lançcar exceção se identificador for nulo
        RAISE_APPLICATION_ERROR(-2001, 'Identificador da Editora inval
        returnValue := FALSE;
        RETURN returnValue;
    end if;

    -- verificar se existem livros em stock da editora
    BEGIN
        SELECT 1 INTO returnValue
        FROM livro
        WHERE idEditora AND ROWNUM = 1;
    END;

    IF returnValue = FALSE THEN
```

```
        RAISE naoExisteLivroEitora;
    end if;

    RETURN returnValue;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            returnValue := FALSE;
            RETURN returnValue;

       WHEN naoExisteLivroEitora THEN
           returnValue := FALSE;
           RETURN returnValue;

END;
/
```

```
DECLARE
    temLivros BOOLEAN;
    idEditoraTeste NUMBER := 1; -- Substitua pelo ID da editora que de

BEGIN
    temLivros := fncTemLivrosEditora(idEditoraTeste);

    IF temLivros IS NOT NULL THEN
        IF temLivros THEN
            DBMS_OUTPUT.PUT_LINE('Existem livros em stock da Editora e
        ELSE
            DBMS_OUTPUT.PUT_LINE('Não existem livros em stock da Edito
        END IF;
    ELSE
        DBMS_OUTPUT.PUT_LINE('Identificador da Editora inválido.');
    END IF;
END;
/
```

▼ 5)

Criar um novo script PL/SQL para implementar uma função, designada fncClienteInfo, para retornar a informação pessoal de um dado cliente, recebido por parâmetro. Se o parâmetro fornecido for inválido, a função deve retornar o valor NULL, usando o mecanismo de exceções. Testar adequadamente a função implementada, através de blocos anónimos.

```
CREATE OR REPLACE FUNCTION fncClienteInfo(nifCliente NUMBER)
RETURN VARCHAR2 IS

    nomeCliente VARCHAR2(100);
    infoCliente VARCHAR2(500);

    clienteNaoEncontrado EXCEPTION;
    clienteInvalido EXCEPTION;

BEGIN

    -- verificar se nifCliente é valido
    IF nifCliente IS NULL THEN
        RAISE_APPLICATION_ERROR(-20001, 'NIF DO CLIENTE INVÁLIDO');
    END IF;

    -- verificar se o cliente existe
    BEGIN
        SELECT NOME
        INTO nomeCliente
        FROM cliente
        WHERE nifCliente = nifCliente AND ROWNUM = 1;


    infoCliente := 'Informação do Cliente: ' || nomeCliente || '(NIF:

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE clienteNaoEncontrado;
    END;

    RETURN infoCliente;

    EXCEPTION

    WHEN clienteNaoEncontrado THEN
        RAISE_APPLICATION_ERROR(-20002, 'Cliente não encontrado.');
        RETURN NULL;

    WHEN clienteInvalido THEN
        RETURN NULL; -- ou tratar de outra forma, dependendo do requis

END;
/
```

```
-- Bloco anônimo para testar a função
DECLARE
    clienteInfo VARCHAR2(500);
    nifClienteTeste NUMBER := 123456789; -- Substitua pelo NIF do clie
BEGIN
    clienteInfo := fncClienteInfo(900800100);

    IF clienteInfo IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE(clienteInfo);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Cliente não encontrado ou parâmetro invá
    END IF;
END;
/
```

▼ 6)

Criar um novo script PL/SQL para implementar uma função, designada fncStockAnoEditora, para retornar o stock dos livros editados por uma dada editora num dado ano. A função deve receber, por parâmetro, o identificador da editora e o ano. Este último parâmetro deve ser opcional na invocação da função e o seu valor por omissão deve ser o ano atual. Se qualquer parâmetro fornecido for inválido, a função deve retornar o valor NULL, usando o mecanismo de exceções. Testar adequadamente a função implementada, através de blocos anónimos

```
CREATE OR REPLACE FUNCTION fncStockAnoEditora(
    idEditora NUMBER,
    ano IN NUMBER DEFAULT NULL
) RETURN NUMBER IS
    totalStock NUMBER := 0;
    anoAtual NUMBER;
    editoraNaoEncontrada EXCEPTION;
    parametroInvalido EXCEPTION;

BEGIN
    -- Verificar se o identificador da editora é válido
    IF idEditora IS NULL THEN
        RAISE_APPLICATION_ERROR(-20001, 'Identificador da Editora invá
    END IF;

    -- Verificar se o ano é válido
    IF ano IS NULL THEN
        -- Se ano for nulo, utilizar o ano atual
        SELECT EXTRACT(YEAR FROM SYSDATE) INTO anoAtual FROM DUAL;
    ELSE
```

```
        anoAtual := ano;
    END IF;

    -- Verificar se a editora existe
    BEGIN
        SELECT 1
        INTO totalStock
        FROM livro l
        WHERE l.id_editora = idEditora AND EXTRACT(YEAR FROM l.data_pu
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE editoraNaoEncontrada;
    END;

    RETURN totalStock;

EXCEPTION
    WHEN editoraNaoEncontrada THEN
        RAISE_APPLICATION_ERROR(-20002, 'Editora não encontrada ou sem
        RETURN NULL;
    WHEN parametroInvalido THEN
        RETURN NULL; -- ou tratar de outra forma, dependendo do requis
END;
/
```

▼ 7)

Criar um novo script PL/SQL para implementar uma função, designada fncPrecoVenda, para retornar o preço aplicado a uma dada venda, recebida por parâmetro. Se o parâmetro fornecido for inválido, a função deve retornar o valor NULL, usando o mecanismo de exceções. Testar adequadamente a função implementada, através de blocos anónimos

```

```

▼ 8)

Criar um novo script PL/SQL para implementar uma função, designada fncRegistarNovoCliente, para registar um novo cliente, em que os dados são recebido como parâmetros. Em caso de sucesso, a função deve retornar o identificador do novo cliente. Se os parâmetros fornecidos forem inválidos, a função deve retornar o valor NULL, usando o mecanismo de exceções. Testar adequadamente a função implementada, através de blocos anónimos.

```

```

▼ 9)

Criar um novo script PL/SQL para implementar uma função, designada fncLivrosEditora, que deve retornar um cursor com livros de uma dada editora em stock. A função deve receber, por parâmetro, o identificador da editora. Se o parâmetro fornecido for inválido, a função deve retornar um cursor vazio. Use o mecanismo de exceções. Testar adequadamente a função implementada, através de blocos anónimos.

▼ 10)

Criar um novo script PL/SQL para implementar uma função, designada fncClientesAtivos, que deve retornar um cursor com os clientes com vendas num dado período, incluindo o respetivo valor total. A função deve receber, por parâmetro, o período em análise. Se o período fornecido for inválido ou não houve vendas nesse período, a função deve retornar um cursor vazio. Testar adequadamente a função implementada, através de blocos anónimos.