



ipb

INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Base de Dados II

Engenharia Informática



Trabalho Prático 2

PL/SQL e APEX

José Guedes a56576

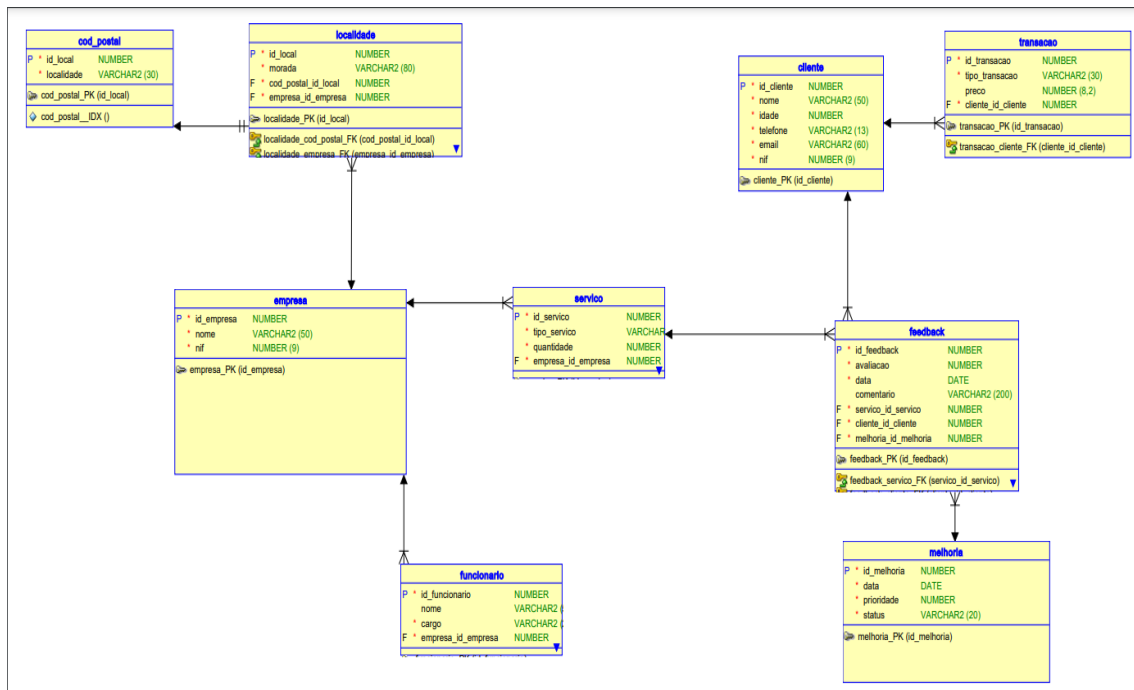
Marcelo Rocha a56584

Índice

Índice.....	2
Modelo ER	4
Utilizar as funções AVG, MIN, MAX, COUNT e SUM.....	5
LEFT e RIGHT JOIN (junções externas)	6
Criar um procedimento que utilize LEFT JOIN	6
Criar um procedimento que utilize RIGHT JOIN.....	7
Utilizar subconsultas.....	8
Criar um procedimento que utilize Single-Row Subqueries (pelo menos 1 parâmetro de entrada).	8
Criar um procedimento que utilize Multiple-Row Subqueries (pelo menos 1 parâmetro de entrada).	9
Utilizar as estruturas IF-THEN-ELSE e CASE.....	10
Criar um procedimento que utilize a estrutura IF-THEN-ELSE.....	10
4.2 Criar um procedimento que utilize a estrutura CASE (pelo menos 1 parâmetro de entrada).	11
Utilizar as estruturas LOOP, FOR e WHILE.....	12
LOOP	12
FOR.....	13
WHILE.....	14
Single-Row Functions	15
Criar um procedimento (ou função) que utilize pelo menos 2 funções DATE e TIME	15
Criar um procedimento (ou função) que utilize pelo menos 2 funções de manipulação de caracteres.....	17
Utilizar Cursores.....	18
Criar um procedimento que utilize 1 cursor e dados de diferentes tabelas.....	18
Criar um procedimento que utilize 2 cursores e dados de diferentes tabelas.....	19
Criar um procedimento que utilize cursores com subconsultas.....	20
Criar um procedimento que utilize um cursor variável (REF CURSOR).....	21
Inserir, alterar e remover registos	22
Criar um procedimento que permita inserir registos na base de dados	22
Criar um procedimento que permita eliminar registos na base de dados	23
Criar um procedimento que permita alterar registos na base de dados.....	24
Utilizar estruturas de dados do tipo Record.....	25
Criar um procedimento que utilize estruturas de dados do tipo RECORD.....	25
Criar um procedimento que utilize estruturas de dados do tipo RECORD e CURSOR..	27

Utilizar estruturas de dados do tipo Array	28
Criar um procedimento que utilize estruturas de dados do tipo ARRAY	28
Criar um procedimento que utilize estruturas de dados do tipo ARRAY e CURSOR	29
Utilizar TRIGGERS.....	30
Criar um trigger que utilize a declaração BEFORE.....	30
Crie um trigger que utilize a declaração AFTER.	31
Criar um trigger DDL (CREATE, ALTER, ou DROP).....	32
Packages.....	33
Criar um package que possua vários procedimentos (pelo menos 2) e pelo menos 1 função.....	33
Extra(Triggers).....	36
Problemas.....	37

Modelo ER



Nesta base de dados temos 9 tabelas (Cod_postal, Localidade, Empresa, Funcionário, Serviço, Feedback, Melhoria, Cliente e Transação).

Na tabela Clientes temos os dados dos clientes que irão realizar transações e dar o seu feedback. O feedback será sobre um serviço e poderão sugerir melhorias.

Na tabela Empresa armazenamos os dados de varias empresas, estas terão serviços associados a si, uma ou mais localidades e os seus funcionários.

Utilizar as funções AVG, MIN, MAX, COUNT e SUM

1) Utilizar as funções AVG, MIN, MAX, COUNT e SUM.

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex1(tipo_trans in varchar2) AS
    avg_price number;
    min_price number;
    max_price number;
    count_trans number;
    sum_price number;

BEGIN
    SELECT round(min(preco)),round(max(preco)),count(*),round(sum(preco)),
    round(avg(preco)) into min_price,max_price,count_trans,sum_price,avg_price
    FROM TRANSACAO
    WHERE TRANSACAO.tipo=tipo_trans group by tipo having count(*)>0;

    htp.p('tipo de transacao: '||tipo_trans||'<br>'||'preço mínimo: '||min_price||'<br>'||'preço máximo: '||max_price||'<br>'||'media preço: '|| avg_price||'<br>'||'total : '|| sum_price||'<br>');
    htp.p('nº de transações: '|| count_trans||'<br>');

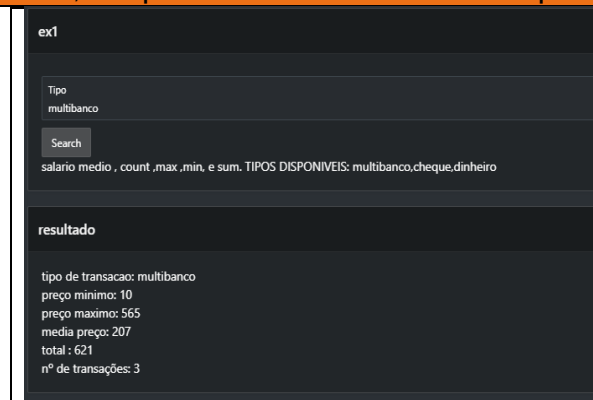
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            htp.p('Nenhuma transação encontrada para o tipo: ' || tipo_trans || '<br>');
END;
```

b) Resultado produzido pelo código.

```
tipo de transação: multibanco
preço mínimo: 10
preço máximo: 565
media preço: 207
total : 621

nº de transações: 3
```

c) captura do ecrã do formulário produzido no APEX.



- Utiliza o input para devolver preço medio, mínimo, máximo, a soma de todas as compras feitas com o tipo do input, e o numero de transações feitas com o tipo do input.

LEFT e RIGHT JOIN (junções externas)

Criar um procedimento que utilize LEFT JOIN

2) 2.1) Criar um procedimento que utilize LEFT JOIN.

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex2_1 AS
BEGIN
    FOR z IN ( SELECT c1.nif AS nif_cliente, e1.nif AS nif_empresa
              FROM cliente c1
              LEFT JOIN empresa e1 ON c1.nif = e1.nif)
    LOOP
        htp.p('NIF do Cliente: ' || z.nif_cliente || '<br>' || 'NIF da Empresa: ' || z.nif_empresa ||
        '<br>');
    END LOOP;
END;
```

b) Resultado produzido pelo código.

```
NIF do Cliente: 123456789
NIF da Empresa: 123456789

NIF do Cliente: 123456789
NIF da Empresa: 123456789

NIF do Cliente: 123123213
NIF da Empresa:

NIF do Cliente: 123123456
NIF da Empresa:

NIF do Cliente: 123123123
NIF da Empresa:

NIF do Cliente: 123456788
NIF da Empresa:
```

c) Captura do ecrã do formulário produzido no APEX.

resultado ex2_1

```
NIF do Cliente: 123456789
NIF da Empresa: 123456789
NIF do Cliente: 123456789
NIF da Empresa: 123456789
NIF do Cliente: 123123213
NIF da Empresa:
NIF do Cliente: 123123456
NIF da Empresa:
NIF do Cliente: 123123123
NIF da Empresa:
NIF do Cliente: 123456788
NIF da Empresa:
```

- Um procedimento **Left Join** onde mostra todos os NIF's da tabela cliente a esquerda e a direita mostra só os que correspondem a um igual ao de clientes.

Criar um procedimento que utilize RIGHT JOIN

2) 2.2) Criar um procedimento que utilize RIGHT JOIN.

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex2_2 AS
BEGIN
  FOR z IN (
    SELECT c1.nif AS nif_cliente, e1.nif AS nif_empresa
    FROM cliente c1
    RIGHT JOIN empresa e1 ON c1.nif = e1.nif
  ) LOOP
    htp.p('NIF do Cliente: ' || z.nif_cliente || '<br>' || 'NIF da Empresa: ' || z.nif_empresa || '<br>');
  END LOOP;
END;
```

b) Resultado produzido pelo código.

```
NIF do Cliente: 123456789NIF da Empresa: 123456789
NIF do Cliente: 123456789NIF da Empresa: 123456789
NIF do Cliente: NIF da Empresa: 123409876
NIF do Cliente: NIF da Empresa: 335443211
NIF do Cliente: NIF da Empresa: 112233445
```

c) Captura do ecrã do formulário produzido no APEX.

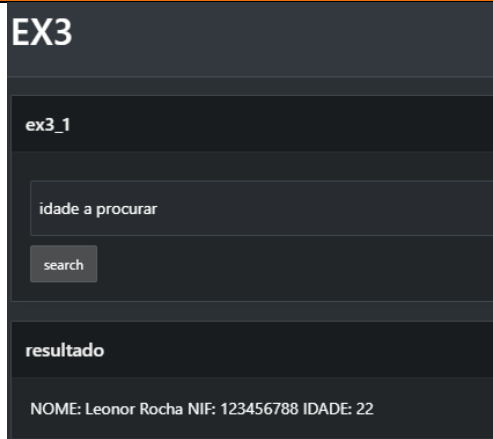
resultado ex2_2

```
NIF do Cliente: 123456789NIF da Empresa: 123456789
NIF do Cliente: 123456789NIF da Empresa: 123456789
NIF do Cliente: NIF da Empresa: 123409876
NIF do Cliente: NIF da Empresa: 335443211
NIF do Cliente: NIF da Empresa: 112233445
```

- Neste caso temos um **Right Join**, faz o mesmo que o **Left Join** mas agora a direita, mostrando depois a esquerda os que são iguais.

Utilizar subconsultas

Criar um procedimento que utilize Single-Row Subqueries (pelo menos 1 parâmetro de entrada).

3) 3.1) Criar um procedimento que utilize Single-Row Subqueries (pelo menos 1 parâmetro de entrada).	
a) Código PL/SQL utilizado.	
<pre>CREATE OR REPLACE PROCEDURE ex3_1(age NUMBER) AS x_nome CLIENTE.NOME%TYPE; x_nif CLIENTE.NIF%TYPE; x_idade CLIENTE.IDADE%TYPE; CURSOR c_clientes IS SELECT DISTINCT INITCAP(nome), nif, idade FROM cliente WHERE idade = (SELECT idade FROM cliente WHERE idade < age) ORDER BY idade; BEGIN OPEN c_clientes; LOOP FETCH c_clientes INTO x_nome, x_nif, x_idade; EXIT WHEN c_clientes%NOTFOUND; HTP.P('NOME: ' x_nome ' NIF: ' x_nif ' IDADE: ' x_idade '
'); END LOOP; CLOSE c_clientes; END;</pre>	
b) Resultado produzido pelo código.	
NOME: Leonor Rocha NIF: 123456788 IDADE: 22 **NOTA** Testar idade 23 anos, para retornar a pessoa mais nova;	
c) Captura do ecrã do formulário produzido no APEX.	
	<ul style="list-style-type: none"> O procedimento ex3_1 mostra informações do clientes com a idade menor que a fornecida.

Criar um procedimento que utilize Multiple-Row Subqueries (pelo menos 1 parâmetro de entrada).

3) 3.2) Criar um procedimento que utilize Multiple-Row Subqueries (pelo menos 1 parâmetro de entrada).

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex3_2(age NUMBER) AS
    id_cliente CLIENTE.ID%TYPE;

    CURSOR count_cli IS
        SELECT COUNT(ID)
        FROM CLIENTE
        WHERE idade IN (SELECT idade FROM CLIENTE WHERE IDADE < AGE);
BEGIN
    OPEN count_cli;

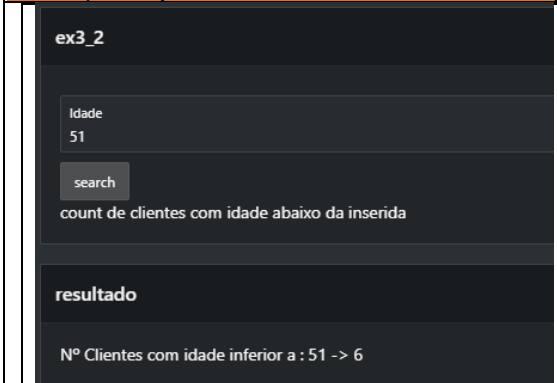
    LOOP
        FETCH count_cli INTO id_cliente;
        EXIT WHEN count_cli%NOTFOUND;
        htp.p('Nº Clientes com idade inferior a : ' || age || ' -> ' || id_cliente);
    END LOOP;

    CLOSE count_cli;
END;
```

b) Resultado produzido pelo código.

Nº Clientes com idade inferior a : 45 -> 4

c) Captura do ecrã do formulário produzido no APEX.

	<ul style="list-style-type: none"> O procedimento ex3_2 mostra a contagem de clientes com idade menor que a fornecida.
---	---

Utilizar as estruturas IF-THEN-ELSE e CASE

Criar um procedimento que utilize a estrutura IF-THEN-ELSE.

4) 4.1) Criar um procedimento que utilize a estrutura IF-THEN-ELSE.

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex4_1 AS
  CURSOR c_melhoria IS
    SELECT prioridade, status FROM MELHORIA;
    v_prioridade MELHORIA.prioridade%TYPE;
    v_status MELHORIA.status%TYPE;
BEGIN
  OPEN c_melhoria;
  LOOP
    FETCH c_melhoria INTO v_prioridade, v_status;
    EXIT WHEN c_melhoria%NOTFOUND;

    IF v_prioridade = 0 THEN
      http.p(v_prioridade || '-> Nao Preocupante ---STATUS: ' || v_status);
    ELSIF v_prioridade = 1 THEN
      http.p(v_prioridade || '-> Pouco Preocupante ---STATUS: ' || v_status);
    ELSIF v_prioridade = 2 THEN
      http.p(v_prioridade || '-> Media(baixa) Preocupação ---STATUS: ' || v_status);
    ELSIF v_prioridade = 3 THEN
      http.p(v_prioridade || '-> Media(Alta) Preocupação ---STATUS: ' || v_status);
    ELSIF v_prioridade = 4 THEN
      http.p(v_prioridade || '-> Alta Preocupação ---STATUS: ' || v_status);
    ELSIF v_prioridade = 5 THEN
      http.p(v_prioridade || '-> Altamente URGENTE ---STATUS: ' || v_status);
    END IF;
  END LOOP;

  CLOSE c_melhoria;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    http.p('Nenhuma melhoria encontrada.');
```

b) Resultado produzido pelo código.

```
1 -> Pouco Preocupante ---STATUS: Incompleto
5 -> Altamente URGENTE ---STATUS: resolvido
3 -> Media(Alta) Preocupação ---STATUS: Ativo
```

c) Captura do ecrã do formulário produzido no APEX.

resultado 4_1

```
1 -> Pouco Preocupante ---STATUS: Incompleto
5 -> Altamente URGENTE ---STATUS: resolvido
3 -> Media(Alta) Preocupação ---STATUS: Ativo
```

- O procedimento ex4_1 mostra a prioridade e status de cada melhoria

4.2 Criar um procedimento que utilize a estrutura CASE (pelo menos 1 parâmetro de entrada).

4) 4.2) Criar um procedimento que utilize a estrutura CASE (pelo menos 1 parâmetro de entrada).

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex4_2 (p_tipo IN VARCHAR2) AS
  CURSOR transacao_cur IS
    SELECT id, id_cliente, tipo,
      CASE
        WHEN preco >= 0 AND preco <= 49 THEN 'Aplicar Taxa' --Aplica taxa quando valor <49
        WHEN preco >= 50 THEN 'Taxa nao aplicavel'
        ELSE 'Preco Invalido'
      END AS precos
    FROM Transacao
    WHERE tipo = p_tipo;
  v_id Transacao.id%TYPE;
  v_id_cliente Transacao.id_cliente%TYPE;
  v_tipo Transacao.tipo%TYPE;
  v_precos VARCHAR2(30);

BEGIN
  OPEN transacao_cur;

  LOOP
    FETCH transacao_cur INTO v_id, v_id_cliente, v_tipo, v_precos;
    EXIT WHEN transacao_cur%NOTFOUND;


    http.p('ID: ' || v_id || ', ID Cliente: ' || v_id_cliente || ', Tipo: ' || v_tipo || ', Preços: ' || v_precos);
  END LOOP;

  CLOSE transacao_cur;
END;
```

b) Resultado produzido pelo código.

```
ID: 1, ID Cliente: 1, Tipo: multibanco, Preços: Aplicar Taxa
ID: 3, ID Cliente: 2, Tipo: multibanco, Preços: Aplicar Taxa
ID: 6, ID Cliente: 2, Tipo: multibanco, Preços: Taxa não aplicável
```

c) Captura do ecrã do formulário produzido no APEX.

	<ul style="list-style-type: none"> O procedimento ex4_2 mostra informações sobre as transações com base no tipo fornecido, e mostra se vai ser aplicada taxa ao preço.
---	---

Utilizar as estruturas LOOP, FOR e WHILE

LOOP

5) Utilizar as estruturas LOOP.

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex5_loop AS
  cursor cliente_cur IS
    SELECT nome, idade, nif
    FROM cliente;
  v_cliente cliente_cur%ROWTYPE;
BEGIN
  OPEN cliente_cur;

  LOOP
    FETCH cliente_cur INTO v_cliente;
    EXIT WHEN cliente_cur%NOTFOUND;

    htp.p('Nome: ' || v_cliente.nome || ' Idade: ' || v_cliente.idade || ' NIF: ' || v_cliente.nif);
  END LOOP;

  CLOSE cliente_cur;
END;
```

b) Resultado produzido pelo código.

```
Nome: Leonor Rocha Idade: 22 NIF: 123456788
Nome: João Silva Idade: 30 NIF: 123456789
Nome: Francisco Guedes Idade: 24 NIF: 123123213
Nome: António Barbosa Idade: 50 NIF: 123123123
Nome: Ze António Idade: 50 NIF: 123456789
```

c) Captura do ecrã do formulário produzido no APEX.

<div> <div>EX5_loop</div> <div></div> <div></div> <div></div> <div>resultado ex5_loop</div> <div> Nome: Leonor Rocha Idade: 22 NIF: 123456788 Nome: antonioassdsdadsdasdasda Idade: 23 NIF: 123123456 Nome: João Silva Idade: 30 NIF: 123456789 Nome: Francisco Guedes Idade: 24 NIF: 123123213 Nome: António Barbosa Idade: 50 NIF: 123123123 </div> </div>	<ul style="list-style-type: none"> Utiliza um ciclo LOOP e um cursor para mostrar todos os clientes.
--	---

FOR

5) Utilizar as estruturas FOR.

a) Código PL/SQL utilizado.

```
BEGIN
  FOR v_cliente IN (SELECT nome, idade FROM cliente) LOOP
    htp.p('Nome: ' || v_cliente.nome || '<br>' || '->Idade: ' || v_cliente.idade || '<br>');
  END LOOP;
END;
```

b) Resultado produzido pelo código.

```
Nome: Leonor Rocha
->Idade: 22

Nome: João Silva
->Idade: 30

Nome: Francisco Guedes
->Idade: 24

Nome: António Barbosa
->Idade: 50

Nome: Ze António
->Idade: 50
```

c) Captura do ecrã do formulário produzido no APEX.

EX5_FOR

resultado ex5_for

```
Nome: Leonor Rocha
->Idade: 22
Nome: antonioassadsdasdasdasdasda
->Idade: 23
Nome: João Silva
->Idade: 30
Nome: Francisco Guedes
->Idade: 24
Nome: António Barbosa
->Idade: 50
```

- Utiliza um ciclo FOR para mostrar nome e idade.

WHILE

5) Utilizar as estruturas WHILE.

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex5_while AS
  CURSOR cliente_cur IS
    SELECT nome, idade, nif FROM cliente;
  v_nome cliente.nome%TYPE;
  v_idade cliente.idade%TYPE;
  v_nif cliente.nif%TYPE;
  max_registos integer:=0;
BEGIN
  OPEN cliente_cur;

  WHILE max_registos <5 LOOP
    FETCH cliente_cur INTO v_nome, v_idade, v_nif;
    EXIT WHEN cliente_cur%NOTFOUND;
    http.p('Nome: '|| v_nome || ' Idade: '|| v_idade || ' NIF: '|| v_nif);
    max_registos:=max_registos+1;
  END LOOP;

  CLOSE cliente_cur;
END;
```

b) Resultado produzido pelo código.

```
Nome: Leonor Rocha Idade: 22 NIF: 123456788
Nome: acéfalo Idade: 13 NIF: 918756541
Nome: lara lopes Idade: 32 NIF: 123123123
Nome: Francisco Guedes Idade: 24 NIF: 123123213
Nome: António Barbosa Idade: 50 NIF: 123123123
```

c) Captura do ecrã do formulário produzido no APEX.

EX5_WHILE

resultado ex5_while

```
Nome: Leonor Rocha Idade: 22 NIF: 123456788
Nome: antonioassadsdasdasdasda Idade: 23 NIF: 123123456
Nome: João Silva Idade: 30 NIF: 123456789
Nome: Francisco Guedes Idade: 24 NIF: 123123213
Nome: António Barbosa Idade: 50 NIF: 123123123
```

- Utiliza o ciclo WHILE para mostrar no máximo 5 clientes.

Single-Row Functions

Criar um procedimento (ou função) que utilize pelo menos 2 funções DATE e TIME

6) 6.1) Criar um procedimento (ou função) que utilize pelo menos 2 funções DATE e TIME.

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex6_1 AS
    m_data melhoria.data%TYPE;
    m_lastday DATE;
    data_hoje DATE;
    dias_ate_hoje NUMBER;
    meses_depois date;

    CURSOR c_datas IS
        SELECT data FROM melhoria;
BEGIN
    OPEN c_datas;
    data_hoje := SYSDATE;
    LOOP
        FETCH c_datas INTO m_data;
        EXIT WHEN c_datas%NOTFOUND;

        m_lastday := LAST_DAY(m_data);
        dias_ate_hoje := round(data_hoje)-m_data;
        meses_depois:=add_months(m_data,6);
        htp.p('Ultimo dia do mes(dd-mm-aaaa): ' || TO_CHAR(m_lastday, 'DD-MM-YYYY') || '<br>');
        htp.p('Data da melhoria ate hoje: ' || TO_CHAR(dias_ate_hoje) || '<br>');
        htp.p('6 meses depois da melhoria: ' || to_char(meses_depois) || '<br>');
    END LOOP;
    CLOSE c_datas;
END;
```

b) Resultado produzido pelo código.

```
Ultimo dia do mes(dd-mm-aaaa): 30-09-2023

Dias da melhoria ate hoje: 254

6 meses depois da melhoria: 03/09/2024

Ultimo dia do mes(dd-mm-aaaa): 31-12-2004

Dias da melhoria ate hoje: 7088

6 meses depois da melhoria: 06/23/2005

Ultimo dia do mes(dd-mm-aaaa): 31-08-2022

Dias da melhoria ate hoje: 651

6 meses depois da melhoria: 02/08/2023
```

c) Captura do ecrã do formulário produzido no APEX.

ex6_1

Ultimo dia do mes(dd-mm-aaaa): 30-09-2023
Data da melhoria ate hoje: 260
6 meses depois da melhoria: 3/9/2024
Ultimo dia do mes(dd-mm-aaaa): 31-12-2004
Data da melhoria ate hoje: 7094
6 meses depois da melhoria: 6/23/2005
Ultimo dia do mes(dd-mm-aaaa): 31-08-2022
Data da melhoria ate hoje: 657
6 meses depois da melhoria: 2/8/2023

- Calcula as informações relacionadas à data das melhorias registradas, incluindo o último dia do mês em que ocorreram, o número de dias desde a data da melhoria até hoje e a data seis meses após a melhoria

Criar um procedimento (ou função) que utilize pelo menos 2 funções de manipulação de caracteres

6) 6.2) Criar um procedimento (ou função) que utilize pelo menos 2 funções de manipulação de caracteres.

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex6_2 AS
  c_telefone cliente.telefone%TYPE;
  CURSOR c_clientes IS
    SELECT telefone FROM cliente;
BEGIN
  OPEN c_clientes;
  LOOP
    FETCH c_clientes INTO c_telefone;
    EXIT WHEN c_clientes%NOTFOUND;
    c_telefone:= '+'||pad(c_telefone,12,351);
    htp.p('Numero atualizado: ' || c_telefone || '<br>');
    htp.p('posicao do numero: '||instr(c_telefone,'91')|| '<br>');
  END LOOP;
  CLOSE c_clientes;
END;
```

b) Resultado produzido pelo código.

```
Numero atualizado: +351914566478
posicao do numero: 5

Numero atualizado: +351123456789
posicao do numero: 0

Numero atualizado: +351912345678
posicao do numero: 5

Numero atualizado: +351918756541
posicao do numero: 5

Numero atualizado: +351918776548
posicao do numero: 5

Numero atualizado: +351919934211
posicao do numero: 5
```

c) Captura do ecrã do formulário produzido no APEX.

ex6_2

```
Numero atualizado: +351999444222
posicao do numero: 0
Numero atualizado: +351917889765
posicao do numero: 5
Numero atualizado: +351123456789
posicao do numero: 0
Numero atualizado: +351918756541
posicao do numero: 5
Numero atualizado: +351999444222
posicao do numero: 0
Numero atualizado: +351919934211
posicao do numero: 5
```

- Este procedimento atualiza os números de telefone dos clientes, adicionando o código de país (+351).

Utilizar Cursores

Criar um procedimento que utilize 1 cursor e dados de diferentes tabelas

7) 7.1) Criar um procedimento que utilize 1 cursor e dados de diferentes tabelas.

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex7_1(tipo_t IN VARCHAR2) AS
  CURSOR cliente_transacao IS
    SELECT nome,tipo,preco FROM cliente c
      JOIN transacao t ON c.id=t.id_cliente
    WHERE tipo=tipo_t;

  v_nome cliente.nome%TYPE;
  v_tipo transacao.tipo%TYPE;
  v_preco transacao.preco%TYPE;

BEGIN
  OPEN cliente_transacao;
  LOOP
    FETCH cliente_transacao INTO v_nome,v_tipo,v_preco;
    EXIT WHEN cliente_transacao%NOTFOUND;
    http.p('NOME: '||v_nome || ' TIPO: ' || v_tipo || ' PRECO: '||v_preco||'EUR. ');
  END LOOP;

  CLOSE cliente_transacao;
END;
```

b) Resultado produzido pelo código.

```
NOME: António Barbosa TIPO: multibanco PRECO: 10EUR.
NOME: Ze aziado TIPO: multibanco PRECO: 45.99EUR.
NOME: Ze aziado TIPO: multibanco PRECO: 565.34EUR.
```

c) Captura do ecrã do formulário produzido no APEX.

ex7_1

tipo transacção
multibanco

search tipo

resultado

```
NOME: António Barbosa TIPO: multibanco PRECO: 10EUR.
NOME: Ze aziado TIPO: multibanco PRECO: 45.99EUR.
NOME: Ze aziado TIPO: multibanco PRECO: 565.34EUR.
```

- Utiliza um cursor para guardar e mostrar dados de duas tabelas.

Criar um procedimento que utilize 2 cursores e dados de diferentes tabelas

7) 7.2) Criar um procedimento que utilize 2 cursores e dados de diferentes tabelas.

a) Código PL/SQL utilizado.


```
CREATE OR REPLACE PROCEDURE ex7_2(nome_cliente IN VARCHAR2) AS
  CURSOR cliente_transacao IS
    SELECT nome, tipo, preco
    FROM cliente c
    JOIN transacao t ON c.id = t.id_cliente
    WHERE nome=nome_cliente;
  v_nome cliente.nome%TYPE;
  v_tipo transacao.tipo%TYPE;
  v_preco transacao.preco%TYPE;
  CURSOR cliente_servico IS
    SELECT s.tipo
    FROM servico s
    JOIN cliente c ON s.id_cliente = c.id
    WHERE c.nome=nome_cliente;
  v_tipo_servico servico.tipo%TYPE;
BEGIN
  OPEN cliente_transacao;
  LOOP
    FETCH cliente_transacao INTO v_nome, v_tipo, v_preco;
    EXIT WHEN cliente_transacao%NOTFOUND;
    http.p('Cliente->' || v_nome || ', Tipo Pagamento->' || v_tipo || ', Preço->' || v_preco || ' EUR.');

```

b) Resultado produzido pelo código.

Cliente->Leonor Rocha, Tipo Pagamento->dinheiro, Preço->6000 EUR.

c) Captura do ecrã do formulário produzido no APEX.

	<ul style="list-style-type: none"> Inserindo o nome, temos a informação do tipo de pagamento e o preço.
---	--

Criar um procedimento que utilize cursores com subconsultas

7) 7.3) Criar um procedimento que utilize cursores com subconsultas.

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex7_3(nome_c IN VARCHAR2) AS
  CURSOR cliente_cur IS
    SELECT nome, idade, nif,
      (SELECT COUNT(*) FROM transacao t WHERE t.id_cliente = c.id) AS total_transacoes
    FROM cliente c
    WHERE c.nome = nome_c;
  v_nome cliente.nome%TYPE;
  v_idade cliente.idade%TYPE;
  v_nif cliente.nif%TYPE;
  v_total_transacoes NUMBER;
  cliente_nao_encontrado EXCEPTION;
BEGIN
  OPEN cliente_cur;
  FETCH cliente_cur INTO v_nome, v_idade, v_nif, v_total_transacoes;

  IF cliente_cur%NOTFOUND THEN
    RAISE cliente_nao_encontrado;
  END IF;

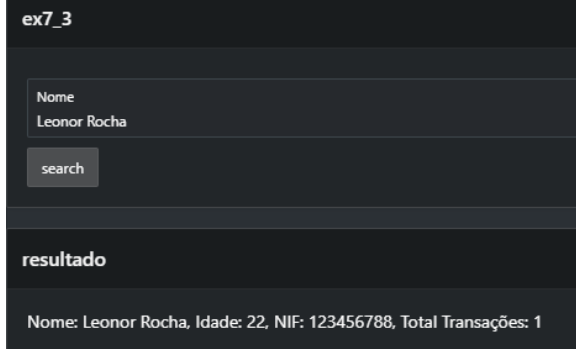
  http.p('Nome: '||v_nome||', Idade: '||v_idade||', NIF: '||v_nif||', Total Transações: '||v_total_transacoes);
  FETCH cliente_cur INTO v_nome, v_idade, v_nif, v_total_transacoes;

  CLOSE cliente_cur;
EXCEPTION
  WHEN cliente_nao_encontrado THEN
    http.p('Cliente não encontrado.');
```

b) Resultado produzido pelo código.

Nome: António Barbosa, Idade: 50, NIF: 123123123, Total Transações: 2

c) Captura do ecrã do formulário produzido no APEX.

	<ul style="list-style-type: none"> Inserindo o nome é nos mostrado a idade, NIF e o número total de transações correspondentes ao nome passado.
---	--

Criar um procedimento que utilize um cursor variável (REF CURSOR).

7) 7.4) Criar um procedimento que utilize um cursor variável (REF CURSOR)
a) Código PL/SQL utilizado.
EM BRANCO
b) Resultado produzido pelo código.
EM BRANCO
c) Captura do ecrã do formulário produzido no APEX.
EM BRANCO

Inserir, alterar e remover registos

Criar um procedimento que permita inserir registos na base de dados

8) 8.1) Criar um procedimento que permita inserir registos na base de dados.

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex8_1 (idc in number, nomec in varchar2, idadec in
number, telefonec in number, emailc in varchar2, nifc in varchar2) AS

BEGIN
    INSERT INTO cliente (id, nome, idade, telefone, email, nif)
    VALUES (idc,nomec,idadec,telefonec,emailc,nifc);

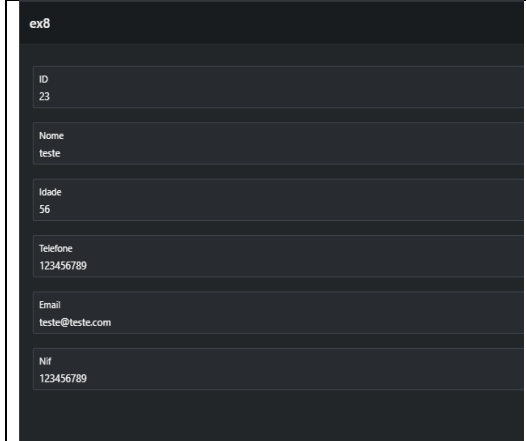
    http.p('Cliente inserido com sucesso: ID=' || idc || ', Nome=' || nomec || ', Idade=' || idadec || ',
    Telefone=' || telefonec || ', Email=' || emailc || ', NIF=' || nifc);

    EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        http.p('Valores duplicados, verifique os dados novamente');
END;
```

b) Resultado produzido pelo código.

Cliente inserido com sucesso: ID=100, Nome=Jose Francisco, Idade=24,
Telefone=80808080, Email=toma_la_e_da_ca@hotmail.com, NIF=080808080

c) Captura do ecrã do formulário produzido no APEX.

	<ul style="list-style-type: none"> • Permite inserir os registos na base de dados.
---	---

Criar um procedimento que permita eliminar registos na base de dados

8) 8.2) Criar um procedimento que permita eliminar registos na base de dados.

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex8_2 (idc IN NUMBER) AS
    v_id cliente%rowtype;
BEGIN
    SELECT * INTO v_id FROM Cliente WHERE id = idc;

    DELETE FROM cliente WHERE id = idc;

    http.p('Cliente com o ID : ' || v_id.id || ' e nome: ' || v_id.nome || ' apagado com sucesso');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        http.p('Erro: Nenhum registro encontrado com o ID fornecido.');
```

b) Resultado produzido pelo código.

Cliente com o ID : 100 e nome: Jose Francisco apagado com sucesso

c) Captura do ecrã do formulário produzido no APEX.

<p>ex8_2</p> <div> <div>Delete</div> <div>60</div> <div>delete</div> </div> <p>resultado</p> <p>Cliente com o ID : 60 e nome: antonioassadsdasdasdasda apagado com sucesso</p>	<ul style="list-style-type: none"> • Permite apagar o Cliente com base no seu ID.
--	--

Criar um procedimento que permita alterar registos na base de dados

8) 8.3) Criar um procedimento que permita alterar registos na base de dados.

a) Código PL/SQL utilizado.

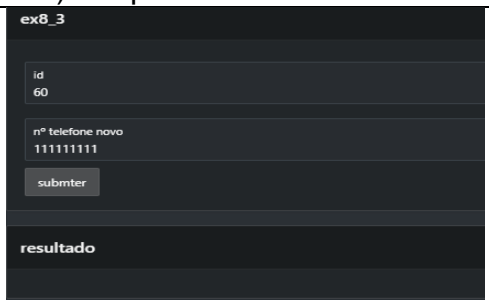
```
CREATE OR REPLACE PROCEDURE ex8_3 (idc in number, telefonec in number) AS
v_cliente cliente%ROWTYPE;
BEGIN
    SELECT * INTO v_cliente FROM Cliente WHERE id = idc;
    UPDATE cliente SET telefone=telefonec
    WHERE id=idc;
    http.p('ID: '||v_cliente.id || ', numero Novo: '||telefonec);

    EXCEPTION
    WHEN NO_DATA_FOUND THEN
        http.p('Erro: Nenhum registro encontrado com o ID fornecido.');
```

b) Resultado produzido pelo código.

ID: 6, numero Novo: 333333333

c) Captura do ecrã do formulário produzido no APEX.

	<ul style="list-style-type: none"> Alteramos o número de telefone.
--	---

d) Notas.

Para o ex8_3() correr normalmente, devemos desativar o trigger ex11_2().

```
ALTER TRIGGER ex11_2 DISABLE;
BEGIN
    ex8_3 (1, 999444222);
END;
ALTER TRIGGER ex11_2 ENABLE;
```


Utilizar estruturas de dados do tipo Record

Criar um procedimento que utilize estruturas de dados do tipo RECORD

9) 9.1) Criar um procedimento que utilize estruturas de dados do tipo RECORD.

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex9_1 (
  id_c IN NUMBER,
  nome_c IN VARCHAR2,
  idade_c IN VARCHAR2, telefone_c IN VARCHAR2, email_c IN VARCHAR2,
  nif_c IN NUMBER
) AS
  TYPE clientes_rec IS RECORD (
    id cliente.id%TYPE,
    nome cliente.nome%TYPE,
    idade cliente.idade%TYPE,
    telefone cliente.telefone%TYPE,
    email cliente.email%TYPE,
    nif cliente.nif%TYPE
  );
  clientes clientes_rec;
BEGIN
  clientes.id := id_c;
  clientes.nome := nome_c;
  clientes.telefone := telefone_c;
  clientes.nif := nif_c;
  clientes.email := email_c;
  clientes.idade := idade_c;

  HTP.P('ID: ' || clientes.id || ' ; ' || '<br>' || 'Nome: ' || clientes.nome || ' ; ' || '<br>' || 'Idade: ' || clientes.idade);
  HTP.P('Email: ' || clientes.email || ' ; ' || '<br>' || 'Telefone: ' || clientes.telefone || ' ; ' || '<br>' || 'NIF: ' || clientes.nif);
END;
```

b) Resultado produzido pelo código.

```
ID: 88;
Nome: José Guedes;
Idade: 24
Email: ja_vai@hotmail.com;
Telefone: 999888777;
NIF: 90898707
```

c) Captura do ecrã do formulário produzido no APEX.

<div><div>ex9_1</div><div><div>ID</div><div>1</div></div><div><div>Name</div><div>MARCELO ROCHA</div></div><div><div>Idade</div><div>23</div></div><div><div>Telefone</div><div>915899455</div></div><div><div>Email</div><div>rochamarcelo79223@gmail.com</div></div><div><div>Nif</div><div>278675554</div></div><div><div>mostrar</div></div></div> <div><div>resultado</div><div><div>ID: 1</div><div>Nome: MARCELO ROCHA</div><div>Idade: 23</div><div>Email: rochamarcelo79223@gmail.com</div><div>Telefone: 915899455</div><div>NIF: 278675554</div></div></div>		<ul style="list-style-type: none">Mostramos os dados que inserimos.
---	--	---

Criar um procedimento que utilize estruturas de dados do tipo RECORD e CURSOR

9) 9.2) Criar um procedimento que utilize estruturas de dados do tipo RECORD e CURSOR.

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex9_2 (
  id_c IN NUMBER) AS
  TYPE clientes_rec IS RECORD (
    id cliente.id%TYPE,
    nome cliente.nome%TYPE,
    idade cliente.idade%type,
    telefone cliente.telefone%TYPE,
    email cliente.email%type,
    nif cliente.nif%TYPE
  );
  cursor c_clientes is select id,nome,idade,telefone,email,nif from cliente where id=id_c;
  clientes_data clientes_rec;
BEGIN
  open c_clientes;
  fetch c_clientes into clientes_data;

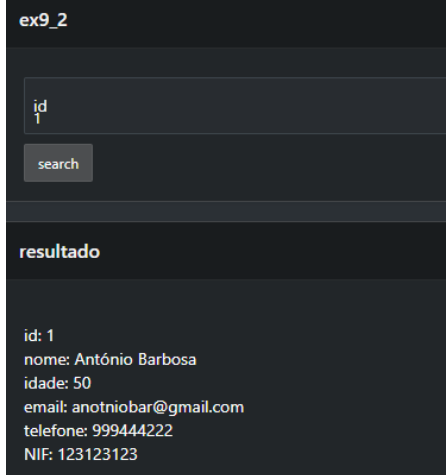
  if c_clientes%found then
    HTP.P(clientes_data.id || ', ' || clientes_data.nome || ', ' || clientes_data.idade);
    HTP.P(clientes_data.email || ', ' || clientes_data.telefone || ', ' || clientes_data.nif);
    else http.p('cliente nao encontrado');
  end if;

  close c_clientes;
END ex9_2;
```

b) Resultado produzido pelo código.

```
ID: 1;
Nome:Marcelo Rocha;
Idade: 23
Email:rochamarcelo79223@gmail.com;
Telefone: 999888777;
NIF: 27867554
```

c) Captura do ecrã do formulário produzido no APEX.

 <p>The screenshot shows a web application interface with a dark theme. At the top, it says 'ex9_2'. Below that is a form with a label 'id' and a text input field containing the number '1'. There is a 'search' button. Below the form, there is a section titled 'resultado'. Under 'resultado', the following information is displayed: 'id: 1', 'nome: António Barbosa', 'idade: 50', 'email: anotniobar@gmail.com', 'telefone: 999444222', and 'NIF: 123123123'.</p>	<ul style="list-style-type: none"> Este procedimento retorna os dados de um cliente com base no ID fornecido como parâmetro de entrada, incluindo nome, idade, telefone, e-mail e NIF. Se o cliente não for encontrado com o ID fornecido, uma mensagem indicando isso será exibida.
---	---

Utilizar estruturas de dados do tipo Array

Criar um procedimento que utilize estruturas de dados do tipo
ARRAY

10) 10.1) Criar um procedimento que utilize estruturas de dados do tipo ARRAY.

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex10_1 AS
  TYPE arrayNomes IS VARRAY(20) OF VARCHAR2(100);
  nomes arrayNomes := arrayNomes();

  i INTEGER := 0;

BEGIN
  FOR r IN (SELECT nome FROM cliente ) LOOP
    i := i + 1;
    nomes.EXTEND; -- Expande o array para adicionar um novo elemento
    nomes(i) := r.nome;
  END LOOP;

  FOR i IN 1 .. nomes.COUNT LOOP
    htp.p('Nome [' || i || ']: ' || nomes(i));
  END LOOP;

EXCEPTION
  WHEN VALUE_ERROR THEN
    htp.p('Erro de valor encontrado.');
```

b) Resultado produzido pelo código.

```
Nome [1]: Leonor Rocha
Nome [2]: Francisco Guedes
Nome [3]: António Barbosa
Nome [4]: Ze aziado
```

c) Captura do ecrã do formulário produzido no APEX.

ex10_1

resultado

Nome [1]: Leonor Rocha Nome [2]: antonioassadsdasdasdasda Nome [3]: João Silva Nome [4]: Francisco Guedes Nome [5]: António Barbosa Nome [6]: Ze azia

- Mostra os nomes dos clientes e o seu índice.

Criar um procedimento que utilize estruturas de dados do tipo ARRAY e CURSOR

10) 10.2) Criar um procedimento que utilize estruturas de dados do tipo ARRAY e CURSOR.

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE PROCEDURE ex10_2 AS
  TYPE arrayNomes IS VARRAY(20) OF VARCHAR2(100);
  nomes arrayNomes := arrayNomes();
  i INTEGER := 0;

  CURSOR c_cliente IS
    SELECT nome FROM cliente;
  r_nome cliente.nome%type;
BEGIN
  OPEN c_cliente;
  LOOP
    FETCH c_cliente INTO r_nome;
    EXIT WHEN c_cliente%NOTFOUND;
    i := i + 1;
    nomes.EXTEND; -- Expande o array e atribui o valor do nome
    nomes(i) := r_nome;
  END LOOP;
  CLOSE c_cliente;

  FOR i IN 1 .. nomes.COUNT LOOP
    htp.p('Nome [' || i || ']: ' || nomes(i));
  END LOOP;

EXCEPTION
  WHEN VALUE_ERROR THEN
    htp.p('Erro de valor encontrado.');
```

b) Resultado produzido pelo código.

```
Nome [1]: Leonor Rocha
Nome [2]: Francisco Guedes
Nome [3]: António Barbosa
Nome [4]: Ze aziado
```

c) Captura do ecrã do formulário produzido no APEX.

ex10_2

resultado

Nome [1]: Leonor Rocha Nome [2]: antonioassadsdasdasdasda Nome [3]: João Silva Nome [4]: Francisco Guedes Nome [5]: António Barbosa Nome [6]: Ze aziado

- Mostra os nomes dos cliente utilizando um cursor.
- Utiliza um for para ir atribuindo os nomes do cursor ao array.

Utilizar TRIGGERS

Criar um trigger que utilize a declaração BEFORE

11) 11.1) Criar um trigger que utilize a declaração BEFORE.
a) Código PL/SQL utilizado.
<pre>CREATE OR REPLACE TRIGGER ex11_1 BEFORE INSERT OR UPDATE OR DELETE ON cliente BEGIN IF TO_CHAR (SYSDATE, 'HH24') NOT BETWEEN '08' AND '18' THEN RAISE_APPLICATION_ERROR(-20205,'Alterações são permitidas apenas no horário entre as 8am e 6pm'); END IF; END ;</pre>
b) Resultado produzido pelo código.
Não Aplicável
c) Captura do ecrã do formulário produzido no APEX.
Não Aplicável

Crie um trigger que utilize a declaração AFTER.

11) 11.2) Crie um trigger que utilize a declaração AFTER.
a) Código PL/SQL utilizado.
<pre> CREATE OR REPLACE TRIGGER ex11_2 AFTER INSERT OR UPDATE OR DELETE ON cliente FOR EACH ROW DECLARE id_aud NUMBER; nome_cli VARCHAR2(100); tipo_operacao VARCHAR2(10); BEGIN SELECT logs_seq.NEXTVAL INTO id_aud FROM dual; IF INSERTING THEN nome_cli := :NEW.nome; tipo_operacao := 'INSERCAO'; ELSIF UPDATING THEN nome_cli := :NEW.nome; tipo_operacao := 'ATUALIZACAO'; ELSIF DELETING THEN nome_cli := :OLD.nome; tipo_operacao := 'REMOCAO'; END IF; INSERT INTO logs (id, nome_cliente, tipo_operacao, data) VALUES (id_aud, nome_cli, tipo_operacao, SYSDATE); END ; </pre>
b) Resultado produzido pelo código.
Não Aplicável
c) Captura do ecrã do formulário produzido no APEX.
Não Aplicável

Criar um trigger DDL (CREATE, ALTER, ou DROP)

11) 11.3) Criar um trigger DDL (CREATE, ALTER, ou DROP).

a) Código PL/SQL utilizado.

```
CREATE OR REPLACE TRIGGER ex11_3
AFTER CREATE OR ALTER OR DROP
ON SCHEMA
DECLARE
    id_aud NUMBER;
    ddl_operation VARCHAR2(20);
    object_name VARCHAR2(100);
BEGIN
    SELECT logs_seq.NEXTVAL INTO id_aud FROM dual;

    IF ora_sysevent = 'CREATE' THEN
        ddl_operation := 'CREATE';
    ELSIF ora_sysevent = 'ALTER' THEN
        ddl_operation := 'ALTER';
    ELSIF ora_sysevent = 'DROP' THEN
        ddl_operation := 'DROP';
    END IF;

    INSERT INTO Log_DDL (id, operacao, data)
    VALUES (id_aud, ddl_operation, SYSDATE);
END ;
```

b) Resultado produzido pelo código.

****Não Aplicável****

c) Captura do ecrã do formulário produzido no APEX.

****Não Aplicável****

Packages

Criar um package que possua vários procedimentos (pelo menos 2) e pelo menos 1 função.

12) 12.1) Criar um package que possua vários procedimentos (pelo menos 2) e pelo menos 1 função.

A) Código PL/SQL utilizado.

```
CREATE OR REPLACE PACKAGE ex12_1 AS
    PROCEDURE inserir_cliente(idc in number, nomec in varchar2, idadec in number, telefonec in
number, emailc in varchar2, nifc in varchar2);
    PROCEDURE alterar_numero(idc in number, telefonec in number);
    FUNCTION procurar_cliente(idc IN NUMBER) RETURN VARCHAR2;
END ex12_1;
/

CREATE OR REPLACE PACKAGE BODY ex12_1 IS
    PROCEDURE inserir_cliente(idc IN NUMBER, nomec IN VARCHAR2, idadec IN NUMBER, telefonec
IN NUMBER, emailc IN VARCHAR2, nifc IN VARCHAR2) IS
    BEGIN
        INSERT INTO cliente (id, nome, idade, telefone, email, nif)
        VALUES (idc, nomec, idadec, telefonec, emailc, nifc);
        http.p('Cliente inserido com sucesso: ID=' || idc || ', Nome=' || nomec || ', Idade=' || idadec || ',
Telefone=' || telefonec || ', Email=' || emailc || ', NIF=' || nifc);
    EXCEPTION
        WHEN DUP_VAL_ON_INDEX THEN
            http.p('Erro: Valores duplicados, verifique os dados novamente.');
```

```
END inserir_cliente;

    PROCEDURE alterar_numero(idc IN NUMBER, telefonec IN NUMBER) IS
        v_cliente cliente%ROWTYPE;
    BEGIN
        SELECT * INTO v_cliente FROM cliente WHERE id = idc;
        UPDATE cliente SET telefone = telefonec WHERE id = idc;
        http.p('ID: ' || v_cliente.id || ', número novo: ' || telefonec);
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            http.p('Erro: Nenhum registro encontrado com o ID fornecido.');
```

```
END alterar_numero;

    FUNCTION procurar_cliente(idc IN NUMBER) RETURN VARCHAR2 IS
        v_cli_name VARCHAR2(100);
    BEGIN
        SELECT nome INTO v_cli_name
        FROM cliente
        WHERE id = idc;

        RETURN v_cli_name;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            http.p('Erro: Nenhum registro encontrado com o ID fornecido.');
```

```
END procurar_cliente;
END ex12_1;
```

B) Resultado produzido pelo código.

A) PROCEDURE inserir_cliente

Cliente inserido com sucesso: ID=9, Nome=João Silva, Idade=30, Tele-
fone=123456789, Email=joadsado@adsd.com, NIF=123456789

B) PROCEDURE alterar_numero

ID: 1, número novo: 999444222

C) FUNCTION procurar_cliente

Nome: António Barbosa

C) Captura do ecrã do formulário produzido no APEX.

C.1) PROCEDURE inserir_cliente

ex12_1

id
112

nome
MARCELO ROCHA

idade
12

telefone
111111111

email
rochamarcelo793@gmail.com

nif
144144144

resultado

Cliente inserido com sucesso: ID=112, Nome=MARCELO ROCHA, Idade=12, Telefone=111111111, Email=rochamarcelo793@gmail.com, NIF=144144144

C.2) PROCEDURE alterar_numero

ex12_2

id
122

telefone
222222222

resultado

ID: 122, número novo: 222222222

C.3) FUNCTION procurar_cliente

ex12_1_search

id
3

resultado

nome: Francisco Guedes

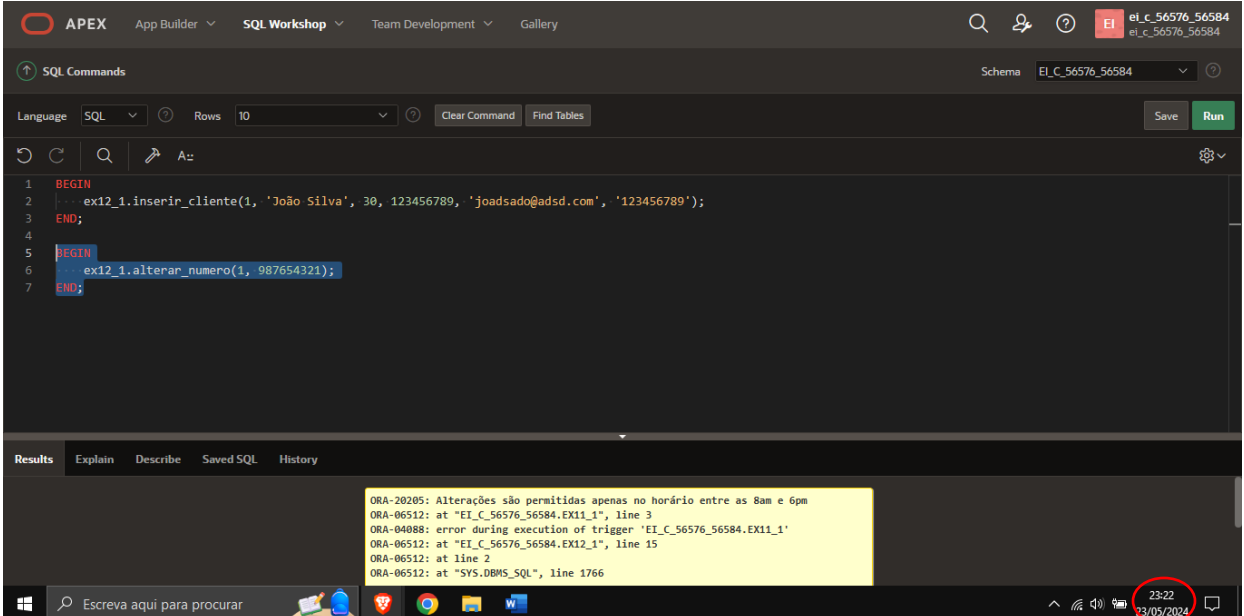
D) Notas.

Para o ex12_1.alterar_numero() correr normalmente, devemos desativar o trigger ex11_2().

```
ALTER TRIGGER ex11_2 DISABLE;  
BEGIN  
    ex12.1.alterar_numero(1, 999444222);  
END;  
ALTER TRIGGER ex11_2 ENABLE;
```

Extra(Triggers)

Trigger acionado ao tentar usar os procedimentos(inserir_cliente e alterar_numero) do package.



The screenshot shows the APEX SQL Workshop interface. The SQL Commands window contains the following code:

```
1 BEGIN
2   ex12_1.inserir_cliente(1, 'João Silva', 30, 123456789, 'joadsado@adsd.com', '123456789');
3 END;
4
5 BEGIN
6   ex12_1.alterar_numero(1, 987654321);
7 END;
```

The Results window shows the following error messages:

```
ORA-20205: Alterações são permitidas apenas no horário entre as 8am e 6pm
ORA-06512: at "EI_C_56576_56584.EX11_1", line 3
ORA-04088: error during execution of trigger 'EI_C_56576_56584.EX11_1'
ORA-06512: at "EI_C_56576_56584.EX12_1", line 15
ORA-06512: at line 2
ORA-06512: at "SYS.DBMS_SQL", line 1766
```

The system clock in the bottom right corner shows 23:22 on 23/05/2024.

Problemas

Enfrentamos vários desafios ao longo do trabalho, mas conseguimos solucioná-los com alguma pesquisa. No entanto, os exercícios `ex8_3()` e `ex12_1.alterar_numero()`, deparamo-nos com um problema relacionado ao trigger que não conseguimos resolver a tempo. Felizmente, ao desativar temporariamente o trigger, conseguimos contornar esse problema.

Problema com trigger ativado:

```
ORA-06502: PL/SQL: numeric or value error: character string buffer too small
ORA-06512: at "EI_C_56576_56584.EX11_2", line 12
ORA-04088: error during execution of trigger 'EI_C_56576_56584.EX11_2'
ORA-06512: at "EI_C_56576_56584.EX8_3", line 5 ORA-06512: at line 2
ORA-06512: at "SYS.DBMS_SQL", line 1766
1. BEGIN
2. ex8_3(4, 999444222);
3. END;
```