



Faculdade de Informática e Administração Paulista

LISTA DE EXERCÍCIOS

PL/SQL

PROCEDURAL LANGUAGE/STRUCTURED QUERY LANGUAGE

PROFESSOR MESTRE ALEXANDRE BARCELOS

Índice

Exercícios

01-Introdução ao PL/SQL	4
02-Declarando Variáveis PL/SQL	5
03-Instruções DML no PL/SQL	7
04-Estruturas de Controle.....	9
05-Registros PL/SQL	10
06-Criando Cursores Explícitos	11
07-Tratando Exceções.....	13
08-Procedures	14
09-Functions	15
10-Packages.....	16
11-Triggers	17

Índice

Soluções

01-Introdução ao PL/SQL	19
02-Declarando Variáveis PL/SQL	20
03-Instruções DML no PL/SQL	21
04-Estruturas de Controle	22
05-Registros PL/SQL	23
06-Criando Cursores Explícitos	24
07-Tratando Exceções	26
08-Procedures	27
09-Functions	28
10-Packages	29
11-Triggers	30

Lista de Exercícios

Esses exercícios estão disponíveis no final de cada apostila.

Cap 01-Introdução ao PL/SQL

1. Qual dos seguintes blocos PL/SQL é executado corretamente?
 - a.

```
BEGIN  
END;
```
 - b.

```
DECLARE  
amount INTEGER(10);  
END;
```
 - c.

```
DECLARE  
BEGIN  
END;
```
 - d.

```
DECLARE  
amount INTEGER(10);  
BEGIN  
DBMS_OUTPUT.PUT_LINE(amount);  
END;
```
2. Crie e execute um bloco anônimo simples que exiba "Hello World". Execute e salve este script como lab_01_02_soln.sql.

Cap 02-Declarando Variáveis PL/SQL

1. Identifique se os nomes dos identificadores são válidos ou inválidos:
 - a. today
 - b. last_name
 - c. today's_date
 - d. Number_of_days_in_February_this_year
 - e. Isleap\$year
 - f. #number
 - g. NUMBER#
 - h. number1to7

2. Identifique se a declaração de variável e a inicialização é válida ou inválida:
 - a. printer_name constant VARCHAR2(10);
 - b. deliver_to VARCHAR2(10):=Johnson;
 - c. by_when DATE:= SYSDATE+1;

3. Examine o seguinte bloco anônimo e escolha a resposta correta apropriada.


```
SET SERVEROUTPUT ON
DECLARE
    fname VARCHAR2(20);
    lname VARCHAR2(15) DEFAULT 'fernandez';
BEGIN
    DBMS_OUTPUT.PUT_LINE( FNAME || ' ' ||lname);
END;
/
```

 - a. O bloco será executado com sucesso mas não será impresso nada.
 - b. O bloco retornará um erro porque a variável fname é declarada sem ser iniciada.
 - c. O bloco será executado com sucesso e irá imprimir 'fernandez'.
 - d. O bloco retornará um erro porque você não pode usar a palavra-chave DEFAULT para inicializar uma variável do tipo VARCHAR2.
 - e. O bloco retornará um erro porque a variável FNAME não foi declarada.

4. Crie um bloco anônimo. Carregue o script da aula 01 lab_01_02_soln.sql, que você criou na pergunta 2 do exercício 1.
 - a. Adicione a seção DECLARE nesse bloco PL/SQL. Na seção DECLARE declare as seguintes variáveis:
 1. Variável today do tipo DATE. Inicializar today com SYSDATE
 2. Variável tomorrow do tipo today. Use o atributo%TYPE para declarar essa variável.
 - b. Na seção executável inicialize a variável tomorrow com uma expressão que calcula a data de tomorrow (adicione um ao valor de today). Exiba o valor de today e tomorrow após de imprimir 'Hello World'
 - c. Execute e salve este script como lab_02_04_soln.sql. Um exemplo da saída é mostrada a seguir.

```
Hello World
TODAY IS : 12-JAN-04
TOMORROW IS : 13-JAN-04
PL/SQL procedure successfully completed.
```

5. Edite o script lab_02_04_soln.sql.
 - a. Crie as bind variables basic_percent e pf_percent do tipo NUMBER.
 - b. Na seção executável do bloco PL/SQL atribua os valores 45 e 12 a basic_percent e pf_percent, respectivamente.
 - c. Encerre o bloco PL/SQL com "/" e exiba o valor das bind variables utilizando o comando PRINT.
 - d. Execute e salve seu arquivo de script como lab_02_05_soln.sql.

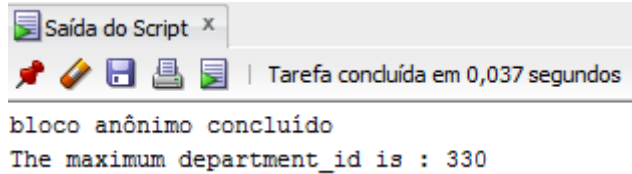
Na seção executável do bloco PL/SQL atribua os valores 45 e 12 a basic_percent e pf_percent, respectivamente.

Encerre o bloco PL/SQL com "/" e exibir o valor das variáveis de ligação usando o comando PRINT.

Execute e salve seu arquivo de script como lab_02_05_soln.sql.

Cap 03-Instruções DML no PL/SQL

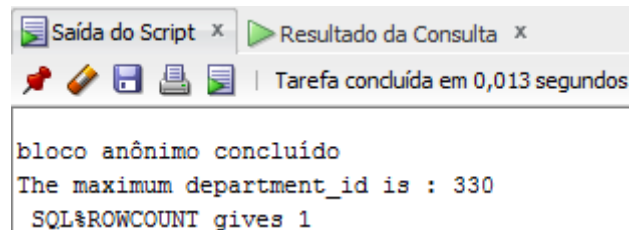
1. Crie um bloco PL/SQL que seleciona o número máximo de departamento na tabela `DEPARTMENTS` e o armazena em uma variável. Imprima os resultados na tela. Salve o bloco PL/SQL em um arquivo nomeado `lab_04_01_soln.sql`.



```

bloco anônimo concluído
The maximum department_id is : 330
  
```

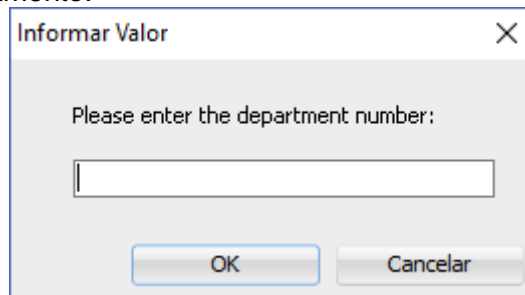
2. Modifique o bloco PL/SQL que você criou no exercício 1 (`lab_04_01_soln.sql`) para inserir um novo departamento na tabela `DEPT`. Salve o bloco PL/SQL em um arquivo nomeado (`lab_04_02_soln.sql`).
 - a. Em vez de imprimir o número do departamento recuperado do exercício 1, adicione 10 a ele (Se o número retornado foi 330 então adicione 10) e use-o como o número do departamento do novo departamento. O novo departamento deve ser definido como: `EDUCATION`
 - b. Use uma variável de substituição do tipo `BIND VARIABLE` para definir o número do departamento novo departamento.
 - c. Deixe um valor nulo nas colunas como `MANAGER_ID` e como o `LOCATION_ID` do novo departamento.
 - d. Execute o bloco PL/SQL.
 - e. Exiba as seguintes mensagens como a seguir demonstrado. (Utilize o atributo de cursor `SQL%ROWCOUNT` para exibir a quantidade de linhas manipuladas.)



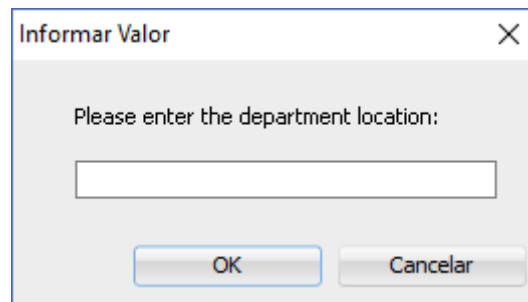
```

bloco anônimo concluído
The maximum department_id is : 330
SQL%ROWCOUNT gives 1
  
```

3. Crie um bloco PL/SQL que atualize a localização (`location_id`) para um departamento existente (`department_id`) (Tabela: `DEPARTMENTS`). Salve o bloco PL/SQL em um arquivo denominado `lab_04_03_soln.sql`.
 - a. Use uma variável de substituição (`VARIABLE` ou `ACCEPT`) para informar o número de departamento.



- b. Use uma variável de substituição (`VARIABLE` ou `ACCEPT`) para informar a localização de departamento. (Consulte a tabela `LOCATIONS` para ter um número de localização válido)

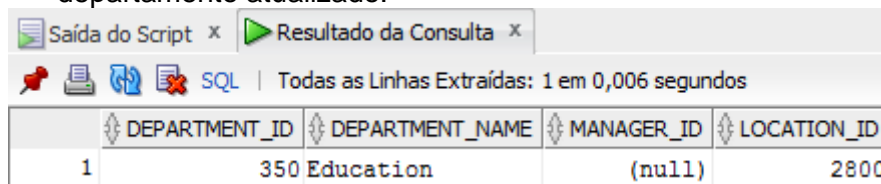


Informar Valor

Please enter the department location:

OK Cancelar

- c. Teste o bloco PL/SQL.
- d. Exiba o nome e o número do departamento, além da localização do departamento atualizado.

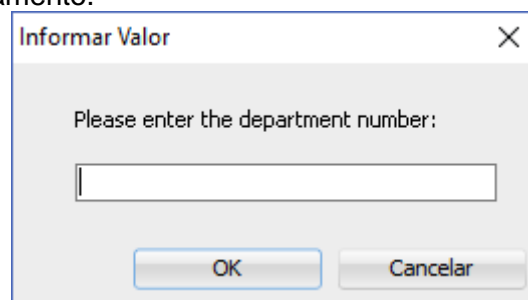


Saída do Script x Resultado da Consulta x

SQL | Todas as Linhas Extraídas: 1 em 0,006 segundos

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	350 Education	(null)	2800

4. Crie um bloco PL/SQL que delete o departamento criado no exercício 2. Salve o bloco PL/SQL em um arquivo denominado lab_04_04_soln.sql.
 - a. Use uma variável de substituição (VARIABLE ou ACCEPT) para informar para o número do departamento.



Informar Valor

Please enter the department number:

OK Cancelar

- b. Imprima o número de linhas afetadas na tela. (Para isso utilize uma `BIND VARIABLE` e exiba essa mensagem após o teste do bloco. Não use `dbms_output.put_line`)
 - c. Teste o bloco PL/SQL.
 - d. O que acontece se você informar um número de departamento inexistente?
 - e. Confirme que o departamento foi deletado.

Cap 04-Estruturas de Controle

1. Execute o comando a seguir para criar a tabela messages.

```
CREATE TABLE MESSAGES
(TEXT VARCHAR(50));
```

- a. Crie um bloco PL/SQL que insira na tabela MESSAGES os números de 1 até 10, excluindo os números 6 e 8.

Após a execução do bloco faça SELECT na tabela MESSAGES para verificar se os números foram inseridos corretamente.

2. Antes de iniciar execute o script a seguir:

Utilize o script a seguir para criar a tabela EMP que será uma cópia da tabela EMPLOYEES

```
drop table emp;
create table emp
as select * from employees;

alter table emp
add stars varchar(50);
```

Crie um bloco PL/SQL que premie um funcionário, anexando um asterisco à coluna STARS da tabela EMP para cada US\$1000 do salário do funcionário. **(2,5 pontos)**

- Se o funcionário recebe um salário (salary) de US\$24000, a string de asteriscos deve conter 24 (vinte e quatro) asteriscos.
- Se o funcionário recebe um salário (salary) de US\$9600, a string de asteriscos deve conter 10 (dez) asteriscos.
- Se o funcionário recebe um salário (salary) de US\$4200, a string de asteriscos deve conter 4 (quatro) asteriscos.

Atualize a coluna STARS da tabela de funcionários (EMP) com a string de asteriscos, de acordo com a regra anterior.

Faça uma atualização por vez, então considere que o código do funcionário deve ser informado.

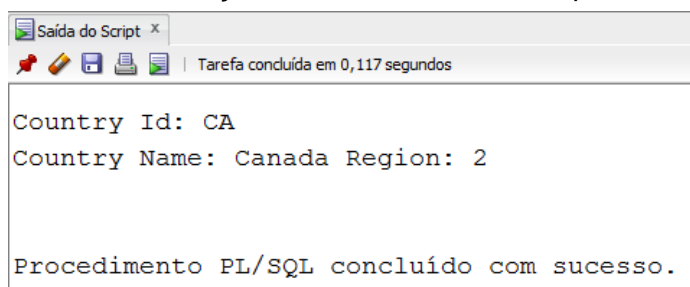
Exemplos de seleção de funcionários:

```
SELECT EMPLOYEE_ID, SALARY, STARS, LENGTH(STARS) STARS_COUNT
FROM EMP
WHERE EMPLOYEE_ID in (100,170,107);
```

EMPLOYEE_ID	SALARY	STARS	STARS_COUNT
100	24000	*****	24
107	4200	****	4
170	9600	*****	10

Cap 05-Registros PL/SQL

1. Escreva um bloco PL/SQL para imprimir informações sobre um determinado país. Declare um registro PL/SQL baseado na estrutura da tabela de countries.
 - a. Use o comando DEFINE para definir uma variável c **countryid**. Atribua CA a countryid. Passe o valor para o bloco PL/SQL através de uma variável de substituição.
 - b. Na seção declare, use o atributo%ROWTYPE e declare a variável country_record de tipo countries.
 - c. Na seção executável, obter todas as informações da tabela **countries** usando countryid. Mostra as informações selecionadas sobre o país.



```
Saída do Script x
Tarefa concluída em 0,117 segundos

Country Id: CA
Country Name: Canada Region: 2

Procedimento PL/SQL concluído com sucesso.
```

Cap 06-Criando Cursores Explícitos

Execute o script a seguir para criar uma nova tabela de armazenamento de funcionários e salários.

```
CREATE TABLE top_dogs
(name VARCHAR2(25),
salary NUMBER(11,2));
```

1. Crie um bloco PL/SQL que determine os funcionários com os maiores salários.
 - a) Aceite um número n como entrada de usuário com um parâmetro de substituição.

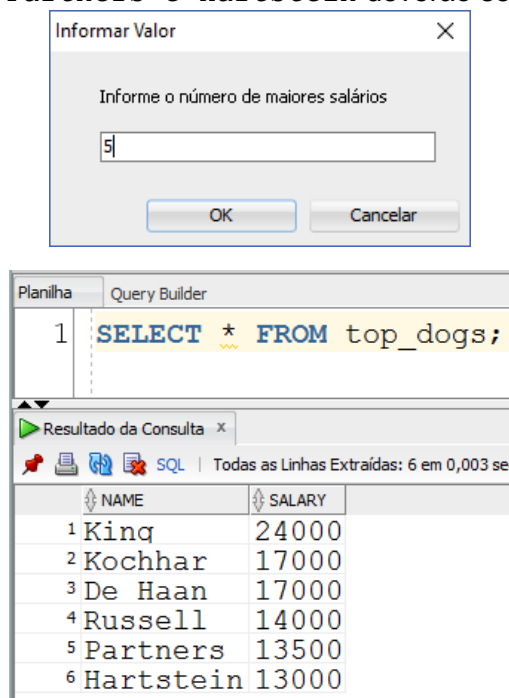
- b) Em um loop, obtenha os sobrenomes (last_name) e salários (salary) dos n funcionários com os maiores salários na tabela EMPLOYEES.
 - c) Armazene os nomes e os salários na tabela TOP_DOGS.
 - d) Suponha que nenhum dos funcionários tenha salário igual ao do outro.
 - e) Teste vários casos especiais, como $n = 0$ ou em que n seja maior que o número de funcionários na tabela EMPLOYEES.

Esvazie a tabela TOP_DOGS depois de cada teste (TRUNCATE TABLE TOP_DOGS;).

Planilha		Query Builder
1		SELECT * FROM top_dogs;
Resultado da Consulta x		
Todas as Linhas Extraídas: 5 em 0,003 segundos		
	NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Russell	14000
5	Partners	13500

2. Considere o caso em que vários funcionários recebem o mesmo salário. Se uma pessoa estiver listada, todas as pessoas com o mesmo salário também deverão estar listadas.

a) Por exemplo, se o usuário informar um valor 5 para n , King, Kochhar, De Haan, Russell, Partners e Hartstein deverão ser exibidos.

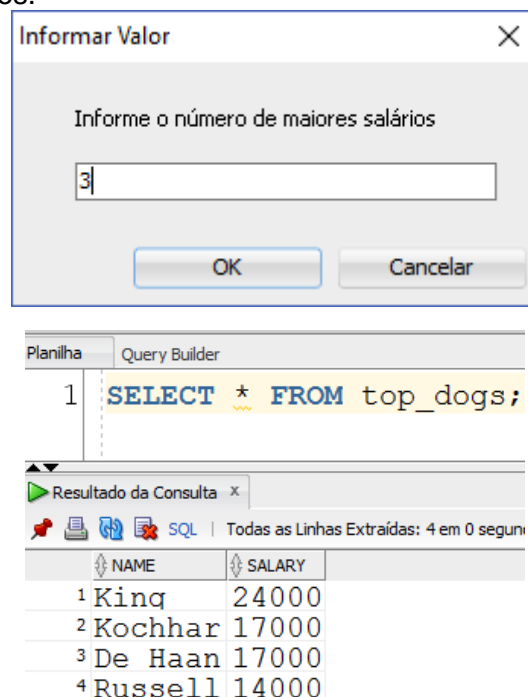


The screenshot shows a dialog box titled "Informar Valor" with a close button (X). The text inside says "Informe o número de maiores salários". A text input field contains the number "5". Below the input field are "OK" and "Cancelar" buttons.

Below the dialog box is a screenshot of a database application interface. The top bar shows "Planilha" and "Query Builder". The query editor shows a single query: `SELECT * FROM top_dogs;`. Below the query editor is a tab labeled "Resultado da Consulta" with a close button (X). The status bar indicates "Todas as Linhas Extraídas: 6 em 0,003 seg". The result is displayed in a table with two columns: "NAME" and "SALARY".

	NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Russell	14000
5	Partners	13500
6	Hartstein	13000

b) Se o usuário informar um valor 3, King, Kochhar, De Haan e Russell deverão ser exibidos.



The screenshot shows a dialog box titled "Informar Valor" with a close button (X). The text inside says "Informe o número de maiores salários". A text input field contains the number "3". Below the input field are "OK" and "Cancelar" buttons.

Below the dialog box is a screenshot of the same database application interface. The query editor still shows `SELECT * FROM top_dogs;`. The "Resultado da Consulta" tab shows the status bar indicating "Todas as Linhas Extraídas: 4 em 0 segundos". The result table now only contains the top 4 rows:

	NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Russell	14000

c) Delete todas as linhas da tabela TOP_DOGS e teste o exercício.

Cap 07-Tratando Exceções

1- Crie um bloco PL/SQL para selecionar o nome do funcionário com um determinado salário.

```
CREATE TABLE messages  
(results VARCHAR(200));
```

- a) Se o salário informado retornar mais de uma linha, trate a exceção com um handler de exceção apropriado e insira, na tabela MESSAGES, a mensagem "More than one employee with a salary of <salário>".
- b) Se o salário informado não retornar qualquer linha, trate a exceção com um handler de exceção apropriado e insira, na tabela MESSAGES, a mensagem "No employee with a salary of <salário>".
- c) Se o salário informado retornar apenas uma linha, insira, na tabela MESSAGES, o nome do funcionário e o valor do salário.
- d) Trate qualquer outra exceção com um handler de exceção apropriado e insira, na tabela MESSAGES, a mensagem "Some other error occurred".
- e) Teste o bloco para vários casos.

Cap 08-Procedures

1. Crie uma procedure chamada CAD_EMPLOYEES que realiza o cadastramento de um FUNCIONARIO. Para isso utilize a tabela EMPLOYEES. Crie as exceções necessárias (pelo menos duas) para o funcionamento correto desse procedimento.

Utilize a estrutura a seguir para a tabela de empregados:

Nome	Nulo?	Tipo
-----	-----	-----
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

2. Crie uma procedure chamada CAD_DEPARTMENTS Que realiza o cadastramento de um DEPARTAMENTO. Para isso utilize a tabela DEPARTMENTS. Crie as exceções necessárias (pelo menos duas) para o funcionamento correto desse procedimento.

Utilize a estrutura a seguir para a tabela de departamentos:

Nome	Nulo?	Tipo
-----	-----	-----
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

Cap 09-Functions

1. Crie uma função chamada `GET_SALARY_TAX` para retornar o imposto descontado do salário mensal de um empregado. Utilize a tabela `EMPLOYEES`.
 - a. A função deve aceitar como parâmetro o código do funcionário .
 - b. A função deve retornar o valor do salário taxado

Use a fórmula a seguir para calcular a taxa anual:
 $(\text{salary} * 0.08)$

2. Crie uma função chamada `GET_ANNUAL_COMP` para retornar o salário anual calculado a partir do salário mensal de um empregado. Utilize a tabela `EMPLOYEES`.
 - a. A função deve aceitar como parâmetros de entrada o código do funcionário.
 - b. A função deve retornar o valor do salário taxado

OBSERVAÇÃO: O valor da comissão pode ser nulo pois existem funcionários que não são comissionados. A função deve retornar um salário anual não nulo.

Use a fórmula a seguir para calcular o salário anual:
 $(\text{salary} * 12) + (\text{commission_pct} * \text{salary} * 12)$

Cap 10-Packages

1. Crie um pacote chamado `PKG_EXERCICIO` que contenha a procedure `CAD_DEPARTMENTS` e a procedure `GET_ANNUAL_COMP`. Teste a execução da procedure e da function que foram empacotadas

Cap 11-Triggers

Construa um TRIGGER que capture todas as alterações de linha (DML – INSERT, UPDATE e DELETE) na tabela chamada EMPLOYEES_TRG (Cópia da Tabela EMPLOYEES) para conformidade com auditoria. Os registros de auditoria devem ser feitos na tabela AUDIT_EMPLOYEES.

Criação da tabela de auditoria:

```
create table AUDIT_EMPLOYEES
(AUD_WHO          VARCHAR2(20),
AUD_WHEN          DATE,
AUD_OPERATION     VARCHAR2(1),
AUD_MODULE        VARCHAR2(30),
EMPLOYEE_ID       NUMBER(6),
FIRST_NAME        VARCHAR2(20),
LAST_NAME         VARCHAR2(25) NOT NULL,
EMAIL             VARCHAR2(25),
PHONE_NUMBER      VARCHAR2(20) NOT NULL,
HIRE_DATE         DATE NOT NULL,
JOB_ID            VARCHAR2(10) NOT NULL,
SALARY            NUMBER(8,2),
COMMISSION_PCT    NUMBER(2,2),
MANAGER_ID        NUMBER(6),
DEPARTMENT_ID     NUMBER(4),
VACATION_BALANCE  NUMBER(6,2)
);
```

Criação da Cópia da tabela EMPLOYEES

```
CREATE TABLE employees_trg
AS SELECT *
FROM employees;
```

Utilize as seguintes instruções para testar o TRIGGER:

```
DELETE FROM employees_trg
WHERE   employee_id = 200;
COMMIT;

INSERT INTO employees_trg VALUES
(200, 'Jennifer', 'Whalen', 'JWHALEN', '515.123.4444', TO_DATE('17-
SET-1987', 'dd-MON-yyyy'), 'AD_ASST', 4400, NULL, 101, 10);
COMMIT;

SELECT   *
FROM     audit_employees;

UPDATE employees_trg
SET      phone_number = '515.123.333'
WHERE    employee_id = 200;
COMMIT;
```


Lista de Respostas

Cap 01-Introdução ao PL/SQL

1.

Alternativa D

2.

```
SET SERVEROUTPUT ON
BEGIN
    DBMS_OUTPUT.PUT_LINE('Hello World');
END;
/
```

Cap 02-Declarando Variáveis PL/SQL

1.

- a. válida
- b. válida
- c. inválida
- d. inválida
- e. válida
- f. inválida
- g. válida
- h. válida

2.

- a. inválida
- b. inválida
- c. válida

3.

Alternativa c

4.

```
SET SERVEROUTPUT ON
DECLARE
    today DATE:=SYSDATE;
    tomorrow today%TYPE;
BEGIN
    tomorrow:=today +1;
    DBMS_OUTPUT.PUT_LINE(' Hello World ');
    DBMS_OUTPUT.PUT_LINE('TODAY IS : '|| today);
    DBMS_OUTPUT.PUT_LINE('TOMORROW IS : ' || tomorrow);
END;
```

5.

```
VARIABLE basic_percent NUMBER
VARIABLE pf_percent NUMBER

SET SERVEROUTPUT ON
DECLARE
    today DATE:=SYSDATE;
    tomorrow today%TYPE;
BEGIN
    :basic_percent:=45;
    :pf_percent:=12;

    tomorrow:=today +1;
    DBMS_OUTPUT.PUT_LINE(' Hello World ');
    DBMS_OUTPUT.PUT_LINE('TODAY IS : '|| today);
    DBMS_OUTPUT.PUT_LINE('TOMORROW IS : ' || tomorrow);
END;
/
PRINT basic_percent
PRINT pf_percent
```

Cap 03-Instruções DML no PL/SQL

1.


```

SET SERVEROUTPUT ON
DECLARE
    max_deptno  NUMBER;
BEGIN
    SELECT MAX(department_id)  INTO max_deptno  FROM
        departments;
    DBMS_OUTPUT.PUT_LINE('The maximum department_id is : ' ||
max_deptno);
END;
```
2.


```

VARIABLE dept_id NUMBER
SET SERVEROUTPUT ON
DECLARE
    dept_name departments.department_name%TYPE:= 'Education';
    max_deptno  NUMBER;
BEGIN
    SELECT MAX(department_id)  INTO max_deptno  FROM departments;
    DBMS_OUTPUT.PUT_LINE('The maximum department_id is : ' ||
max_deptno);
    :dept_id := 10 + max_deptno;
    INSERT INTO departments (department_id, department_name,
location_id)
    VALUES (:dept_id,dept_name, NULL);
    DBMS_OUTPUT.PUT_LINE (' SQL%ROWCOUNT gives ' || SQL%ROWCOUNT);
END;
/
SELECT * FROM  departments  WHERE  department_id=:dept_id;
```
3.


```

ACCEPT p_department_id  PROMPT 'Please enter the department
number'
ACCEPT p_location_id    PROMPT 'Please entre the department
location'
DECLARE
    v_department_id
departments.department_id%TYPE:=&p_department_id;
    v_location_id    departments.location_id%TYPE:=&p_location_id;
BEGIN
    UPDATE departments
    SET location_id=v_location_id
    WHERE department_id=v_department_id;
END;
/
```
4.


```

ACCEPT p_department_id  PROMPT 'Please enter the department
number'
DECLARE
    v_department_id
departments.department_id%TYPE:=&p_department_id;
BEGIN
    delete from departments
    WHERE  department_id=v_department_id;
END;
/
```

Cap 04-Estruturas de Controle

1.

```

BEGIN
FOR i in 1..10 LOOP
  IF i = 6 or i = 8 THEN
    null;
  ELSE
    INSERT INTO messages(results)
      VALUES (i);
  END IF;
END LOOP;
COMMIT;
END;
/
SELECT * FROM messages;

```

2.

```

ACCEPT p_empno PROMPT 'Informe o número do funcionário: '
DECLARE
  v_empno      emp.employee_id%TYPE := TO_NUMBER(&p_empno);
  v_asterisk   emp.stars%TYPE := NULL;
  v_sal        emp.salary%TYPE;
BEGIN
  SELECT NVL(ROUND(salary/1000), 0)
    INTO v_sal
  FROM emp
  WHERE employee_id = v_empno;

  FOR i IN 1..v_sal LOOP
    v_asterisk := v_asterisk || '*';
  END LOOP;

  UPDATE emp
  SET stars = v_asterisk
  WHERE employee_id = v_empno;

  COMMIT;
END;
/

```

Cap 05-Registros PL/SQL

1.

```
SET SERVEROUTPUT ON
SET VERIFY OFF
DEFINE countryid = CA
DECLARE
    country_record countries%ROWTYPE;
BEGIN
    SELECT *
    INTO   country_record
    FROM   countries
    WHERE  country_id = UPPER('&countryid');

    DBMS_OUTPUT.PUT_LINE ('Country Id: ' ||
country_record.country_id ||
    ' Country Name: ' || country_record.country_name
    || ' Region: ' || country_record.region_id);

END;
```

Cap 06-Criando Cursores Explícitos

1.

```
DROP TABLE top_dogs;
```

```
CREATE TABLE hr.top_dogs
(name VARCHAR2(25),
salary NUMBER(11,2));
```

```
ACCEPT p_num PROMPT 'Informe o número de maiores salários '
DECLARE
```

```
    v_num    NUMBER(3) := &p_num;
    v_ename   employees.last_name%TYPE;
    v_sal     employees.salary%TYPE;
    CURSOR emp_cursor IS
        SELECT last_name, salary
        FROM employees
        ORDER BY 2 DESC;
```

```
BEGIN
```

```
    OPEN emp_cursor;
    FETCH emp_cursor INTO v_ename, v_sal;
    WHILE emp_cursor%ROWCOUNT <= v_num AND emp_cursor%FOUND LOOP
        INSERT INTO top_dogs (name, salary)
        VALUES (v_ename, v_sal);
        FETCH emp_cursor INTO v_ename, v_sal;
    END LOOP;
    CLOSE emp_cursor;
    COMMIT;
```

```
END;
```

```
/
```

```
SELECT * FROM top_dogs;
```

2.

```
truncate table top_dogs;
```

```
ACCEPT p_num PROMPT 'Informe o número de maiores salários '
DECLARE
```

```
    v_num NUMBER(3) := &p_num;
    v_ename   employees.last_name%TYPE;
    v_current_sal employees.salary%TYPE;
    v_last_sal   v_current_sal%TYPE;
    CURSOR emp_cursor IS
        SELECT last_name, salary
        FROM employees
        ORDER BY 2 DESC;
```

```
BEGIN
```

```
    OPEN emp_cursor;
    FETCH emp_cursor INTO v_ename, v_current_sal;
    WHILE emp_cursor%ROWCOUNT <= v_num AND emp_cursor%FOUND LOOP
        INSERT INTO top_dogs (name, salary)
        VALUES (v_ename, v_current_sal);
        v_last_sal := v_current_sal;
        FETCH emp_cursor INTO v_ename, v_current_sal;
        IF v_last_sal = v_current_sal THEN
            v_num := v_num + 1;
        END IF;
```



```
        END LOOP;  
        CLOSE emp_cursor;  
    COMMIT;  
END;  
/  
SELECT * FROM top_dogs;
```

Cap 07-Tratando Exceções

```
DELETE FROM MESSAGES;
```

```
SET VERIFY OFF
```

```
DEFINE sal = 6000
```

```
DECLARE
```

```
    ename employees.last_name%TYPE;
```

```
    emp_sal          employees.salary%TYPE := &sal;
```

```
BEGIN
```

```
    SELECT      last_name
```

```
    INTO        ename
```

```
    FROM        employees
```

```
    WHERE       salary = emp_sal;
```

```
    INSERT INTO messages (results)
```

```
    VALUES (ename || ' - ' || emp_sal);
```

```
EXCEPTION
```

```
    WHEN no_data_found THEN
```

```
        INSERT INTO messages (results)
```

```
        VALUES ('No employee with a salary of ' ||
```

```
TO_CHAR(emp_sal));
```

```
    WHEN too_many_rows THEN
```

```
        INSERT INTO messages (results)
```

```
        VALUES ('More than one employee with a salary of ' ||
```

```
TO_CHAR(emp_sal));
```

```
    WHEN others THEN
```

```
        INSERT INTO messages (results)
```

```
        VALUES ('Some other error occurred.');
```

```
END;
```

```
/
```

```
SELECT * FROM messages;
```

Cap 08-Procedures

1.

```

create or replace procedure CAD_EMPLOYEES
(P_EMPLOYEE_ID          NUMBER,
P_FIRST_NAME            VARCHAR2,
P_LAST_NAME             VARCHAR2,
P_EMAIL                 VARCHAR2,
P_PHONE_NUMBER          VARCHAR2,
P_HIRE_DATE             DATE,
P_JOB_ID                VARCHAR2,
P_SALARY                NUMBER,
P_COMMISSION_PCT        NUMBER,
P_MANAGER_ID            NUMBER,
P_DEPARTMENT_ID         NUMBER
)
is
BEGIN
    INSERT INTO EMPLOYEES
    VALUES
    (P_EMPLOYEE_ID,P_FIRST_NAME,P_LAST_NAME,P_EMAIL,P_PHONE_NUMBE
R,P_HIRE_DATE,P_JOB_ID,
    P_SALARY,
P_COMMISSION_PCT,P_MANAGER_ID,P_DEPARTMENT_ID);
    COMMIT;
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    DBMS_OUTPUT.PUT_LINE('Empregado já cadastrado');
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Ocorreu um erro'|| SQLERRM);
END ;
/

```

2.

```

create or replace procedure CAD_DEPARTMENTS
(P_DEPARTMENT_ID        NUMBER ,
P_DEPARTMENT_NAME       VARCHAR2 ,
P_MANAGER_ID            NUMBER,
P_LOCATION_ID           NUMBER)
is
BEGIN
    INSERT INTO DEPARTMENTS
    VALUES
    (P_DEPARTMENT_ID,P_DEPARTMENT_NAME,P_MANAGER_ID,P_LOCATION_ID
);
    COMMIT;
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    DBMS_OUTPUT.PUT_LINE('Departamento já cadastrado');
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Ocorreu um erro'|| SQLERRM);
END ;

```

Cap 09-Functions**1.**

```

create or replace function get_salary_tax (
p_ei employees.employee_id%type)
return number
is
    v_sal employees.salary%type;
begin
    select salary * 0.8
    into v_sal
    from employees
    where employee_id = p_ei;

    return v_sal;
end get_salary_tax;
/

```

```

select employee_id,last_name,salary,get_salary_tax(employee_id)

from employees;

```

2.

```

create or replace function  GET_ANNUAL_COMP
(p_ei employees.employee_id%type)
return number
is
    v_saltax employees.salary%type;
begin
    select (salary*12) + (nvl(commission_pct,0)*salary*12)
    into    v_saltax
    from    employees
    where employee_id=p_ei;

    return v_saltax;
end GET_ANNUAL_COMP;
/

```

```

select employee_id,last_name,salary,GET_ANNUAL_COMP(employee_id)
ann_sal
from employees
where department_id=90;

```

Cap 10-Packages

```

create or replace package PKG_EXERCICIO is
procedure CAD_DEPARTMENTS
(P_DEPARTMENT_ID      NUMBER ,
 P_DEPARTMENT_NAME    VARCHAR2 ,
 P_MANAGER_ID         NUMBER,
 P_LOCATION_ID        NUMBER);
FUNCTION get_annual_comp(
    sal  IN employees.salary%TYPE,
    comm IN employees.commission_pct%TYPE)
RETURN NUMBER;
end;
/

create or replace package body PKG_EXERCICIO is
procedure CAD_DEPARTMENTS
(P_DEPARTMENT_ID      NUMBER ,
 P_DEPARTMENT_NAME    VARCHAR2 ,
 P_MANAGER_ID         NUMBER,
 P_LOCATION_ID        NUMBER)
is
BEGIN
    INSERT INTO DEPARTMENTS
    VALUES
(P_DEPARTMENT_ID,P_DEPARTMENT_NAME,P_MANAGER_ID,P_LOCATION_ID
);
    COMMIT;
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
    DBMS_OUTPUT.PUT_LINE('Departamento já cadastrado');
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Ocorreu um erro'|| SQLERRM);
END ;

FUNCTION hr.get_annual_comp(
    sal  IN employees.salary%TYPE,
    comm IN employees.commission_pct%TYPE)
RETURN NUMBER IS
BEGIN
    RETURN (NVL(sal,0) * 12 + (NVL(comm,0) * nvl(sal,0) * 12));
END get_annual_comp;
end;
/

```

Cap 11-Triggers

```

create trigger TRG_AUDIT_EMPLOYEES;
after insert or update or delete on EMPLOYEES
for each row
declare
    l_operation varchar2(1) :=
        case when updating then 'U'
              when deleting then 'D'
              else 'I' end;
begin
    if updating or inserting then
        insert into AUDIT_EMPLOYEES
            (aud_who
            ,aud_when
            ,aud_operation
            ,aud_module
            ,employee_id
            ,first_name
            ,last_name
            ,email
            ,phone_number
            ,hire_date
            ,job_id
            ,salary
            ,commission_pct
            ,manager_id
            ,department_id)
        values
            (user
            ,sysdate
            ,l_operation
            ,sys_context('USERENV','MODULE')
            ,:new.employee_id
            ,:new.first_name
            ,:new.last_name
            ,:new.email
            ,:new.phone_number
            ,:new.hire_date
            ,:new.job_id
            ,:new.salary
            ,:new.commission_pct
            ,:new.manager_id
            ,:new.department_id);
    else
        insert into AUDIT_EMPLOYEES
            (aud_who
            ,aud_when
            ,aud_operation
            ,aud_module
            ,employee_id
            ,first_name

```

```
,last_name
,email
,phone_number
,hire_date
,job_id
,salary
,commission_pct
,manager_id
,department_id)
values
  (user
  ,sysdate
  ,l_operation
  ,sys_context('USERENV','MODULE')
  ,:old.employee_id
  ,:old.first_name
  ,:old.last_name
  ,:old.email
  ,:old.phone_number
  ,:old.hire_date
  ,:old.job_id
  ,:old.salary
  ,:old.commission_pct
  ,:old.manager_id
  ,:old.department_id);
end if;
end;
/
```