EJERCICIOS PÁGINA 45

1.- Determinar cuales de los siguientes identificadores son equivalentes en PL/SQL.

a) Identificador1 NUMBER;
b) Identificador_1 NUMBER; DISTINTO a)
c) identificador1 NUMBER; IGUAL a)
d) IdEntificador_1 NUMBER; IGUAL b)
e) IDENTIFICADOR1 NUMBER; IGUAL a)

2.- Determinar cual de los siguientes identificadores en PL/SQL es el válido:

Primera variable VARCHAR2(40);
end BOOLEAN;
Una_variable VARCHAR2(40); VÁLIDO
Otra-variable VARCHAR2(40);

3.- ¿ Funcionaría la siguiente sentencia en PL/SQL?. En caso negativo proponer como resolverlo.

```
DECLARE
nombre VARCHAR2(80);
direccion VARCHAR2(80);
tipo empleado VARCHAR2(4);
BEGIN
SELECT name, address, type
INTO nombre, direccion, tipo empleado
FROM emp;
END;
```

NO FUNCIONARÍA YA QUE “TIPO EMPLEADO” NO ESTÁ BIEN DEFINIDO NI BIEN USADO EN EL “INTO”. UNA POSIBLE SOLUCIÓN SERIA DEFINIRLO DE ALGUNA DE LAS SIGUIENTES MANERAS:

tipoempleado
tipo_empleado …. O CUALQUIERA DE LAS FORMAS ACEPTADAS POR ORACLE.

4- En las siguientes sentencias de definición de subtipos, hay una que no es válida... Indicar cual es, y como solucionarlo.

```
DECLARE
SUBTYPE numero1 IS NUMBER;
SUBTYPE cadena IS VARCHAR2(10);
total numero1(6,2);
```

LA DEFINICIÓN CORRECTA SERIA:

DECLARE
SUBTYPE numero1 IS NUMBER(6,2);
SUBTYPE cadena IS VARCHAR2(10);
total numero1;

5.- Indicar del siguiente juego de declaraciones, cuales son correctas y cuales no, indicando además por qué no son correctas.

DECLARE
cierto BOOLEAN:=FALSE; CORRECTA
id_externo NUMBER(4) NOT NULL; INCORRECTA. “Not null” y no tiene un valor asignado
cosa NUMBER:=2; CORRECTA
producto NUMBER:=2*cosa; CORRECTA
suma NUMBER:=cosa + la_otra; INCORRECTA. “La_otra” se hace uso de una variable aún no declarada.
la_otra NUMBER:=2; CORRECTA
tipo_uno NUMBER(7,2) NOT NULL:=3; CORRECTA
tipo_otro tipo_uno%TYPE; CORRECTA

6.- Suponiendo que tenemos una tabla llamada EMP, en la cual existen tres campos: nombre VARCHAR2(40), direccion VARCHAR2(255), telefono NUMBER(10)... ¿Qué valores podrán tomar las variables que definimos en la siguiente declaración?

DECLARE
valor1 emp%ROWTYPE;
valor2 emp.nombre%TYPE;
valor3 emp.telefono%TYPE;

CURSOR c1 IS SELECT nombre,direccion FROM emp;
valor4 c1%ROWTYPE;
valor5 emp%ROWTYPE;

VALOR1 nombre VARCHAR2(40), direccion VARCHAR2(255), telefono NUMBER(10)
…
VALOR2 VARCHAR2(40)
VALOR3 NUMBER(10)
VALOR4 nombre VARCHAR2(40), direccion VARCHAR2(255)
VALOR5 nombre VARCHAR2(40), direccion VARCHAR2(255), telefono NUMBER(10)
…

7.- En base al enunciado anterior, ¿Sería correcta la siguiente asignación?

valor1:=valor5; CORRECTA YA QUE LOS DOS TIENEN EL MISMO TIPO, COGEN EL DE LA TABLA EMP;

¿y esta?

valor4:=valor1; INCORRECTA VALOR4 SOLO TIENE DOS CAMPOS Y VALOR1 TODOS LOS DEL TIPO DE LA TABLA EMP.

Razonar el por qué en ambos casos.

8.-¿ Es esta declaración correcta en PL/SQL?

```
DECLARE --
i, j NUMBER;
```
INCORRECTA. LAS DECLARACIONES DE VARIABLES SE HACEN INDIVIDUALMENTE.

¿Y si la escribieramos así?

```
DECLARE
i, j NUMBER, NUMBER;
```
INCORRECTA. LAS DECLARACIONES DE VARIABLES SE HACEN INDIVIDUALMENTE.

9- Escribir un bloque PL/SQL que escriba el texto “Hola”.

```
BEGIN
 DBMS_OUTPUT.PUT_LINE('HOLA');
END;
```

10 – Escribe un bloque PL/SQL que sume dos número y te diga que números ha sumado y cuál es el resultado.

```
DECLARE
NUM1 NUMBER:=6;
NUM2 NUMBER:=2;
BEGIN
 DBMS_OUTPUT.PUT_LINE(‘LA SUMA DE ’||NUM1||' + '||NUM2||' ES IGUAL A '||
(NUM1 + NUM2));
END;
```

11- Escribe un bloque PL/SQL donde declares una variable, inicializada a la fecha actual, que imprimes por pantalla.

```
DECLARE
FECHA DATE:=SYSDATE;
BEGIN
 DBMS_OUTPUT.PUT_LINE(FECHA);
END;
```

12- Escribe un boque PL/SQL que almacene en una variable numérica el año actual. Muestra por pantalla el año. Puedes usar las funciones to_number y to_char.

```
DECLARE
ANYO VARCHAR2(4):= TO_CHAR(SYSDATE, 'YYYY');
BEGIN
 DBMS_OUTPUT.PUT_LINE(ANYO);
END;
```

EJERCICIOS PÁGINA 79

1.- ¿Es correcta la siguiente sintaxis General de la sentencia IF-THEN ELSE?,¿Por qué?, ¿Cómo la escribirías?.

```
BEGIN
IF condicion1 THEN
BEGIN
secuencia_de_instrucciones1;
ELSE
secuencia_de_instrucciones2;
ENDIF;
END;
```

NO HAY QUE PONER EL SEGUNDO “BEGIN” QUE VA DESPUÉS DEL “THEN”. EL “ENDIF” DEL FINAL VA SEPARADO “END IF”.

LA FORMA CORRECTA SERIA:

```
BEGIN
IF condicion1 THEN
secuencia_de_instrucciones1;
ELSE
secuencia_de_instrucciones2;
END IF;
END;
```

2.- ¿Qué resultado nos daría la siguiente comparación?

```
create or replace function f1 return boolean is
identificador1 VARCHAR2(10):='Hola Pepe';
identificador2 VARCHAR2(10):='Hola pepe';
BEGIN
IF identificador1<>identificador2 THEN
RETURN TRUE;
ELSE
RETURN FALSE;
END IF;
END;
```

DARÍA ERROR YA QUE LA SENTENCIA RETURN SOLO SE UTILIZA EN FUNCIONES. SI FUNCIONASE DEVOLVERÍA TRUE

3.- Indicar que errores existen en el siguiente código fuente:

```
DECLARE
a NUMBER:=1;
b NUMBER:=6;
salida_bucle BOOLEAN;
BEGIN
salida_bucle:='FALSE';   INCORRECTO. FALSE SE ESCRIBE SIN COMILLAS
WHILE NOT salida_bucle
LOOP
BEGIN      INCORRECTO. AQUÍ NO VA UN BEGIN
IF a>=b THEN
salida_bucle:='TRUE';   INCORRECTO. TRUE SE ESCRIBE SIN COMILLAS
ELSE
a:=(a+1);
END IF;
END LOOP;
END;
```

4- ¿Qué valor contendrá la variable 'sumador' al salir del bucle?, ¿Por qué?

```
DECLARE
sumador NUMBER;
BEGIN
FOR i IN 1..100 LOOP
sumador:=sumador+i;
END LOOP;
END;
```

NO SE PUEDE SABER YA QUE NO ESTA SUMADOR INICIALIZADO Y NO TIENE VALOR DEFINIDO.

5.- ¿Qué resultado dará la ejecución del siguiente código?

```
DECLARE
temp NUMBER;
SUBTYPE numero IS temp%TYPE;
valor numero;
BEGIN
WHILE valor<20 LOOP
valor:=valor+1;
END LOOP;
dbms_output.put_line(valor);
END;
```

NO SE PUEDE SABER YA QUE NO ESTA VALOR INICIALIZADO Y NO TIENE UN VALOR DEFINIDO.

6- ¿ Funcionaría el siguiente trozo de código?, ¿Por qué?, ¿Cómo arreglarlo?

```
DECLARE
mi_valor NUMBER;
cierto BOOLEAN:=FALSE;
BEGIN
WHILE NOT cierto
LOOP
IF mi_valor=NULL THEN
mi_valor:=1;
ELSE
mi_valor:=mi_valor+1;
END IF;
IF mi_valor>100 THEN cierto:=TRUE; END IF;
EXIT WHEN cierto;
END LOOP;
END;
```

NO FUNCIONARÍA PORQUE SE COMPARA UNA VARIABLE DE TIPO NUMBER CON NULL. PARA QUE FUNCIONE SE PODRIA INICIALIZAR A 0 MI_VALOR O CAMBIAR “IF mi_valor=NULL THEN” POR “IF mi_valor IS NULL THEN”

7.- Escribir la sintaxis General de un código que evalúe si se cumple una condición, en caso de cumplirse que ejecute una serie de sentencias, en caso contrario que evalúe otra, que de cumplirse ejecute otras instrucciones, si ésta no se cumple que evalúe una tercera condición.. y así N veces. En caso de existir varias soluciones, comentarlas y escribir la más óptima o clara.

```
IF condición1 THEN sentencia1;
ELSIF condición2 THEN sentencia2;
ELSIF condición3 THEN sentencia3;
.
.
.
ELSE  sentenciaN
END IF;
```

TAMBIEN SE PUEDE HACER MEDIANTE EL CASE U OTRA POSIBILIDAD MEDIANTE IF-THEN-ELSE ANIDADOS.

8.- Implementar en PL/SQL un bucle infinito que vaya sumando valores en una variable de tipo NUMBER.

```
DECLARE
valor NUMBER:=0;
BEGIN
WHILE TRUE
LOOP
valor := valor + 1;
END LOOP;
END;
```

9.- En base al bucle anterior, añadirle la condición de que salga cuando la variable sea mayor que 10.000.

```
DECLARE
valor NUMBER:=0;
BEGIN
WHILE valor <= 10000 LOOP
valor := valor + 1;
END LOOP;
END;
```

10.- Implementar un bucle en PL/SQL mediante la sentencia WHILE, en el cual vayamos sumando valores a una variable mientras ésta sea menor que 10, y asegurándonos de que el bucle se ejecuta por lo menos una vez.

```
DECLARE
valor NUMBER:= 0;
BEGIN
WHILE valor<10 LOOP
valor := valor +1;
END LOOP;
END;
```

11.- Implementar en PL/SQL, el código necesario de un programa que al final de su ejecución haya almacenado en una variable llamada 'cadena', el siguiente valor: cadena:='10*9*8*7*6*5*4*3*2*1'

```
DECLARE
valor NUMBER:= 10;
cadena VARCHAR2(20) := '';
BEGIN
FOR valor IN REVERSE 1 .. 10 LOOP
cadena := cadena||valor;
IF valor > 1 THEN cadena := cadena||'*';
END IF;
END LOOP;
DBMS_OUTPUT.PUT_LINE(CADENA);
END;
```

EJERCICIOS PÁGINA 106

1.- ¿Funcionaria el siguiente código?. Explicar por qué y cómo solucionarlo (si se os ocurren varias formas de arreglarlo, explicarlas todas).

```
DECLARE
base NUMBER:=100;
BEGIN
FOR dept IN 1..10 LOOP
UPDATE dept SET nom_dept=base+dept
WHERE cod_dept=base+dept;
END LOOP;
END;
```

NO FUNCIONARÍA YA QUE PL/SQL SUPONE QUE DEPT REFERENCIA AL CONTADOR DEL FOR Y NO A LA TABLA. SERÍA MEJOR CAMBIAR EL NOMBRE DEL CONTADOR DEL BUCLE FOR YA QUE ES IGUAL AL NOMBRE DE LA TABLA.

TAMBIÉN PUEDES PONER EL NOMBRE COMPLETO DE LA TABLA USUARIO.DEPT.

2- ¿Qué ocurriría al ejecutar el siguiente código?, ¿Por qué?, ¿Cómo arreglarlo? (Si existen varias posibilidades, comentarlas e indicar cuál sería más eficiente).

```
DECLARE
ap1_emp VARCHAR2(40):='Fernandez';
BEGIN
DELETE FROM emp WHERE ap1_emp=ap1_emp;
COMMIT;
END;
```

FUNCIONARÍA Y BORRARÍA TODOS LOS DATOS DE LA TABLA EMP, NO SOLO LOS QUE EL “APELLIDO1” SEA FERNANDEZ YA QUE ORACLE CREE QUE LOS DOS API_EMP QUE APARECEN EN LA SENTENCIA WHERE REFERENCIAN A LA COLUMNA DE LA BASE DE DATOS. LA SOLUCIÓN ES CAMBIAR EL NOMBRE DE LA VARIABLE O USAR UNA ETIQUETA.

3.- ¿Es correcto el siguiente código en PL/SQL?, ¿Por qué? Nota:Ignorar el funcionamiento del código (no hace nada), ceñirse exclusivamente a la sintaxis válida en PL/SQL.

```
FOR ctr IN 1..10 LOOP
IF NOT fin THEN
INSERT INTO temp VALUES (ctr, 'Hola');
COMMIT;
factor:=ctr*2;
ELSE ctr:=10;
END IF;
END LOOP;
```

FALTARÍA DEFINIR LA VARIABLE FIN.

4.- ¿Qué resultado provoca la ejecución del siguiente código PL/SQL?

```
DECLARE
variable NUMBER:=1;
almacenamos NUMBER:=1;
BEGIN
FOR i IN 5..variable LOOP
almacenamos:=almacenamos+i;
END LOOP;
INSERT INTO traza VALUES(TO_CHAR(almacenamos));
COMMIT;
END;
```

Inserta una fila en la tabla "TRAZA" con un valor de "almacenamos" que se calcula en un bucle, con una suma del valor de la variable "almacenamos" más el valor de la variable "i". Aunque el bucle no se ejecuta ninguna vez ya que el valor de “variable” es menor que 5 (valor mínimo del bucle FOR). Deberia escribirse FOR i IN REVERSE 5..variable LOOP. O darle un valor a variable mayor que 5.

5.- ¿Qué da este bloque?

```
DECLARE
V_num NUMBER;
BEGIN
SELECT COUNT(*) INTO V_num FROM productos;
DBMS_OUTPUT.PUT_LINE(V_num);
END;
```

El código realiza una consulta que cuenta las filas de la tabla 'productos’ y coloca el valor en la variable 'v_num', luego la variable 'v_num' se muestra por pantalla. El resultado será el número de filas.

6.- Crea una tabla MENSAJES con un solo campo VALOR, de tipo VARCHAR2(5). Crea un bloque que inserte 8 elementos en la tabla con valores del 1 al 10, excepto el 4 y 5.

```
CREATE TABLE MENSAJES (
        VALOR VARCHAR(5)
);

BEGIN
  FOR i IN 1 .. 10 LOOP
    IF i != 4 AND i != 5 THEN
      INSERT INTO MENSAJES VALUES(i);
      COMMIT;
    END IF;
  END LOOP;
END;
```

7.- Vuelve a usar la tabla anterior con los datos ya rellenos, y si el número que insertas es menor que 4 entonces debes sacar un mensaje de error si ya existe, si el número es mayor o igual que 4, debes insertarlo si no existe y actualizarlo sumándole 1 si ya existe.

```
DECLARE
  esta MENSAJES.valor%TYPE;
  nuevovalor MENSAJES.valor%TYPE:=&valornuevo;

BEGIN
```

```
SELECT valor into esta FROM MENSAJES where nuevovalor=valor;
IF nuevovalor >= 1 AND nuevovalor < 4 THEN dbms_output.put_line('ERROR YA
EXISTE');
ELSE UPDATE MENSAJES SET valor=valor+1 WHERE valor=nuevovalor;
END if;

EXCEPTION
when no_data_found then dbms_output.put_line('NO ENCONTRADO EL VALOR');
INSERT INTO MENSAJES VALUES(nuevovalor);
end;
```

8.- Crea una tabla PRODUCTO(CODPROD, NOMPROD, PRECIO), usando SQL (no uses un bloque PL/SQL). Añade un producto a la tabla usando una sentencia insert dentro de un bloque PL/SQL.

```
CREATE TABLE PRODUCTO(
  CODPROD NUMBER,
  NOMPROD VARCHAR2(100),
  PRECIOPROD NUMBER(6,2),
  CONSTRAINT PK_producto PRIMARY KEY(CODPROD)
);

BEGIN
  INSERT INTO PRODUCTO VALUES (1, 'Balon Futbol', 23.75);
  COMMIT;
END;
```

9.- Añade otro producto, ahora utilizando una lista de variables en la sentencia insert.

```
DECLARE
  codigo PRODUCTO.CODPROD%TYPE := 2;
  nombre PRODUCTO.NOMPROD%TYPE  := 'Botas Futbol';
  precio PRODUCTO.PRECIOPROD%TYPE  := 42.50;
BEGIN
  INSERT INTO PRODUCTO VALUES (codigo, nombre, precio);
  COMMIT;
END;
```

10.- Añade, ahora usando un registro PL/SQL, dos productos más.

```
DECLARE
  producto_registro PRODUCTO%ROWTYPE;
BEGIN
  FOR i IN 1 .. 2 LOOP
    producto_registro.codprod := 5+i;
```

```
    producto_registro.nomprod := 'Pantalón' || i;
    producto_registro.precioprod := 10.0 + i;
    INSERT INTO PRODUCTO VALUES (producto_registro.codprod,
producto_registro.nomprod,
      producto_registro.precioprod);
    COMMIT;
  END LOOP;
END;
```

11.- Borra el primer producto insertado, e incrementa el precio de los demás en un 5%.

```
BEGIN
 DELETE FROM PRODUCTO WHERE codprod=1;
 UPDATE PRODUCTO SET PRECIOPROD=PRECIOPROD*1.05;
END;
```

12.- Obtén y muestra por pantalla el número de productos que hay almacenados, usando select ... Into y el mensaje “Hay n productos”.

```
DECLARE
 suma NUMBER;
BEGIN
  SELECT COUNT(*) INTO suma FROM PRODUCTO;
  DBMS_OUTPUT.PUT_LINE('Hay ' || suma || ' productos');
END;
```

13.- Obtén y muestra todos los datos de un producto (busca a partir de la clave primaria, escogiendo un código de producto existente. Usa select ... into y un registro PL/SQL.

```
DECLARE
  producto_registro PRODUCTO%ROWTYPE;
BEGIN
  SELECT * INTO  producto_registro FROM PRODUCTO WHERE codprod = 3;
  DBMS_OUTPUT.PUT_LINE('CÓDIGO PRODUCTO:' || producto_registro.codprod);
  DBMS_OUTPUT.PUT_LINE('NOMBRE PRODUCTO:' ||  producto_registro.nomprod);
  DBMS_OUTPUT.PUT_LINE('PRECIO PRODUCTO:' ||  producto_registro.precioprod);
END;
```

14.- Haz un bloque PL/SQL, de forma que se indique que no hay productos, que hay pocos (si hay entre 1 y 3) o el número exacto de productos existentes (si hay más de 3).

```
DECLARE
  numproductos NUMBER;
  mensaje VARCHAR2(50) := '';
```

```
BEGIN
  SELECT COUNT(*) INTO numproductos FROM PRODUCTO;
  IF numproductos = 0 THEN
    mensaje := 'No hay productos';
  ELSIF numproductos >=1 AND numproductos <= 3 THEN
    mensaje := 'Hay pocos';
  ELSE
    mensaje := 'Hay ' || numproductos || ' productos';
  END IF;
  DBMS_OUTPUT.PUT_LINE(mensaje);
END;
```

EJERCICIOS PÁGINA 126

1.- Crear una función pl/sql que duplica la cantidad recibida como parámetro.

```
CREATE OR REPLACE FUNCTION Duplicar (N IN NUMBER) RETURN NUMBER IS

resultado NUMBER;

BEGIN
RETURN (N*2)
END Duplicar;
```

Otra posibilidad

```
BEGIN
resultado:=N*2;
return(resultado);
END DUPLICAR;
```

LLAMADA A LA FUNCIÓN

```
DECLARE
VALOR2 NUMBER;
VALOR NUMBER:='&NUMERO';
BEGIN
VALOR2:=DUPLICAR(VALOR);
dbms_output.put_line(VALOR2);
END;
```

2.- Crear una función pl/sql llamada factorial que devuelva el factorial de un número, por ejemplo 5! = 1 * 2 * 3 * 4 * 5 = 120.

```
CREATE or replace FUNCTION factorial (n IN NUMBER)
  RETURN NUMBER
```

```
  IS BEGIN
    if n = 0 then
      return (1) ;
    else
      return ((n * factorial (n-1))) ;
    end if ;
    END ;

DECLARE
v NUMBER := &valor;
f NUMBER;
BEGIN
f := Factorial(v);
DBMS_OUTPUT.PUT_LINE ('Factorial de ' || v || ' = '|| f );
END;
```

3.- Crear un procedimiento pl/sql que muestra los números desde el 1 hasta el valor pasado como parámetro.

```
CREATE OR REPLACE PROCEDURE cadena (vcadena IN NUMBER) IS

mostrar varchar2:=’’;
BEGIN
FOR i IN 1..vcadena LOOP
mostrar := mostrar || i;
END LOOP;
DBMS_OUTPUT.PUT_LINE(mostrar);
END cadena;


DECLARE
NUM NUMBER:=('&NUMERO');
BEGIN
cadena(NUM);
END;
```

4.- Modificar el procedimiento del ejercicio 3 para que muestre números desde un valor inferior hasta uno superior con cierto salto, que por defecto será 1.

```
CREATE OR REPLACE PROCEDURE muestranum2(
  V_INI IN number,
  V_FIN IN NUMBER,
  SALTO IN number DEFAULT 1
) IS
  CONT number;
   mostrar varchar2:=’’;
```

```
BEGIN
  CONT := V_INI;
  loop
    exit when CONT > V_FIN;
    mostrar=mostrar || CONT;
    CONT := CONT + SALTO;
  end loop;
 dbms_output.put_line(mostrar);
END;
```

5.- Modificar el procedimiento del ejercicio 3 para que inserte los números en una tabla.

```
create table TABLA_NUM(numero NUMBER);

CREATE OR REPLACE PROCEDURE cadena2 (vcadena IN NUMBER) IS

mostrar varchar2:=’’;
BEGIN
FOR i IN 1..vcadena LOOP
insert into TABLA_NUM values(i);
mostrar := mostrar || i;
END LOOP;
DBMS_OUTPUT.PUT_LINE(mostrar);
END cadena2;
```

6- Crea una función llamada, PVP que toma como argumento un código de producto, una descripción y un coste del producto, y realice una inserción en una tabla PRODUCTOS si el código de producto (PK) no existe y en caso de existir actualice los datos de descripción y coste y devuelva el precio de venta al público, que resulta de aplicarle a ese precio de coste un margen comercial del 20%.

TABLA PRODUCTOS

```
CODIGOPRODUCTO  VARCHAR2(15)  --------  Clave primaria
NOMBRE  VARCHAR2(70)
PROVEEDOR  VARCHAR2(50)
DESCRIPCION  CLOB
CANTIDAD  NUMBER(5.0)
PRECIOVENTA  NUMBER(15.2)
COSTE  NUMBER(15,2)

create or replace FUNCTION pvp2 (
codigpro IN VARCHAR2,
descrip IN CLOB,
preciocoste IN NUMBER)  RETURN NUMBER IS

esta PRODUCTOS.CODIGOPRODUCTO%TYPE;
```

```
BEGIN
SELECT CODIGOPRODUCTO INTO ESTA FROM PRODUCTOS WHERE
CODIGOPRODUCTO = CODIGPRO;

UPDATE PRODUCTOS SET DESCRIPCION = DESCRIP
WHERE CODIGOPRODUCTO = ESTA;

UPDATE PRODUCTOS SET COSTE=PRECIOCOSTE
WHERE CODIGOPRODUCTO = ESTA;

RETURN PRECIOCOSTE*1.2;

EXCEPTION
when no_data_found then
INSERT INTO PRODUCTOS(CODIGOPRODUCTO, DESCRIPCION, COSTE)
VALUES(CODIGPRO, DESCRIP, PRECIOCOSTE);
RETURN PRECIOCOSTE*1.2;
end PVP2;
```

7.- Dado este PL/SQL:

```
CREATE OR REPLACE PROCEDURE modificar_precio_producto (codigoprod
NUMBER, nuevo NUMBER) AS Precioant NUMBER(5);
BEGIN
SELECT precio_uni INTO precioant
FROM productos
WHERE cod_producto = codprod;
IF (precioant * 0.20) > ABS( precioant – nuevoprecio) then
UPDATE productos SET precio_uni = nuevoprecio
WHERE cod_producto = codigoprod;
ELSE DBMS_OUTPUT.PUT_LINE(‘Error, modificación superior a 20%’);
END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE (‘No encontrado el producto ‘ || codigoprod);
END modificar_precio_producto;
```

Responde a las siguientes preguntas:

a) ¿Se trata de una función o de un procedimiento, por qué?, habría que cambiar para que fuera lo otro?

Es un procedimiento, porque pone "PROCEDURE" y no devuelve ningún valor, para cambiarlo a función habría que cambiar procedure por function y devolver un valor.

b) ¿Cuál es la cabecera del procedimiento?

modificar_precio_producto (codigoprod NUMBER, nuevo NUMBER)

c) ¿Qué es el precioant?

Una variable que almacena el precio del producto a modificar, antes de ser modificado.

d) ¿Qué es el nuevoprecio?

Una variable que almacena el precio del producto ya modificado.

e) ¿Qué es el precio_uni?

Un campo de la tabla productos

f) ¿Cuáles son los parámetros del procedimiento?

codigoprod y nuevo

g) ¿Qué es NO_DATA_FOUND?

Sirve para atrapar los errores si no encuentra datos.

h) ¿Cuál es el nombre del procedimiento?

modificar_precio_producto

i) ¿Dónde comienza el bloque?

Es todo un bloque de procedimiento

j) ¿Qué hace la cláusula into?

Una cláusula para guardar datos de los campos en variables

k) ¿qué hace la condición del IF?

Comprobar si el nuevo precio supera el incremento del 20% del precio antiguo

l) ¿Porque no tiene la cláusula declare?¿Qué tiene en su lugar?

Porque tiene AS en su lugar.

8.- Corregir los errores de sintaxis en esta función. Esta función PL/SQL devuelve el número PI (3,141592653589793238462...). Calculado mediante el algoritmo que ideó John Wallis en 1665.

```
CREATE FUNCTION piWallis(pIteraciones number) number IS
vCont number
vRet number
vCont = 0
vRet = 1
loop
vCont = vCont + 1;
when vCont > pIteraciones exit loop;
if (vCont % 2) = 0
vRet := vRet * vCont / (vCont + 1);
else vRet := vRet * (vCont + 1) / vCont;
endif;
end loop
return (2 * vRet);
END;
```

SOLUCIÓN

```
CREATE FUNCTION piwallis(piteraciones NUMBER) NUMBER
  vcont NUMBER      -- los valores se declaran antes del IS
  vret NUMBER
IS
  vCont = 0
  vRet = 1
  loop
    vcont = vcont + 1;
    exit WHEN vcont > piteraciones;   – se declara el exit del loop así
    IF MOD(vcont, 2) = 0 -- necesitamos el resto, no la división, se hace con la función MOD
      vret := vret * vcont / (vcont + 1);
    else
      vret := vret * (vcont + 1) / vcont;
    end if;     – end if va separado
  end loop;
  RETURN (2 * vret);
END;
```

9.- Introduces todas las funciones y procedimentos de los ejercicios anteriores en un PACKAGE llamado FUNCIONES_PROCE

```
CREATE OR REPLACE PACKAGE FUNCIONES_PROCE IS

--AQUÍ SE DEFINEN LOS TIPOS QUE UTILIZAREMOS, LAS EXCEPCIONES PROPIAS Y LOS ERRORES A TRATAR--
```

```
FUNCTION Duplicar (N NUMBER) RETURN NUMBER;

FUNCTION factorial (n NUMBER)  RETURN NUMBER;

PROCEDURE cadena (vcadena NUMBER);

PROCEDURE muestranum2(V_INI number, V_FIN NUMBER, SALTO number);

PROCEDURE cadena2 (vcadena NUMBER);

FUNCTION pvp2 (codigpro VARCHAR2, descrip CLOB, preciocoste NUMBER)
RETURN NUMBER;

END FUNCIONES_PROCE;
/

CREATE OR REPLACE PACKAGE BODY FUNCIONES_PROCE AS

– AQUÍ IMPLEMENTO LAS FUNCIONES Y PROCEDIMIENTOS DEFINIDOS
ANTERIORMENTE CON TODAS LAS VARIABLES DE ENTRADA QUE ME HARÁN
FALTA --
.
.
.
.
.
.
END FUNCIONES_PROCE;
/
```

10.- Ejecuta los procedimientos y funciones del nuevo paquete desde un bloque anónimo.

PARA EJECUTARLOS HAY QUE REFERENCIARLOS MEDIANTE EL NOMBRE DEL PAQUETE SEGUIDO DE UN PUNTO Y DEL PROCEDIMIENTO/FUNCIÓN CORRESPONDIENTE.

```
BEGIN
 ...
 FUNCIONES_PROCE.CADENA(20);
 ...
END;
```

EJERCICIOS PÁGINA 151

1- Realiza un cursor sobre la tabla Bancos y sucursales, donde indiques el nombre del banco y el de cada sucursal. Hazlo con Fecth y con FOR.

```
DECLARE
  CURSOR banc_cursor IS
  SELECT nombre_banc BANCO, nombre_suc SUCURSAL FROM bancos, sucursales
  where bancos.cod_banco = sucursales.cod_banco;
  reg_banc  banc_cursor%rowtype;
BEGIN
  OPEN banc_cursor;
  FETCH banc_cursor INTO reg_banc;
  while banc_cursor%found loop
        dbms_output.put_line(reg_banc.banco||' --  '||reg_banc.sucursal);
        fetch banc_cursor INTO reg_banc;
  END loop;
  CLOSE banc_cursor;
END;
```

```
DECLARE
  CURSOR banc_cursor IS
  SELECT nombre_banc BANCO, nombre_suc SUCURSAL FROM bancos, sucursales
  where bancos.cod_banco = sucursales.cod_banco;
BEGIN
  FOR reg_banc IN banc_cursor loop
   dbms_output.put_line(reg_banc.banco||' --  '||reg_banc.sucursal);
  END loop;
end;
```

2- Crea un campo en la tabla Cuenta, llamándolo “Rentable”, recorre la tabla Cuenta y si el haber es mayor que el debe actualizara S, de lo contrario a N, sumando las rentables y las no rentables y sacándolas por pantalla al final. Usa ROWID para hacerlo (identificador único para cada registro de una BD).

```
ALTER TABLE cuentas ADD rentable VARCHAR2(1);
```

```
DECLARE
  counter_s NUMBER := 0;
  counter_n NUMBER := 0;
  CURSOR rent_cursor IS
    SELECT saldo_haber, saldo_debe, cod_banco FROM cuentas
    FOR UPDATE;
BEGIN
  FOR rent_cuent IN rent_cursor loop
    IF rent_cuent.saldo_haber > rent_cuent.saldo_debe THEN
```

```
      UPDATE cuentas SET rentable = 's' WHERE CURRENT OF rent_cursor;
      counter_s := counter_s + 1;
    ELSE
      UPDATE cuentas SET rentable = 'n' WHERE CURRENT OF rent_cursor;
      counter_n := counter_n + 1;
   END IF;
  END loop;
  dbms_output.put_line('Hay '||to_char(counter_s)||' rentables y '||to_char(counter_n)||' no
rentables');
  dbms_output.put_line('En total hay '||to_char(counter_s+counter_n));
end;
```

4.- Utiliza un cursor y un bucle LOOP simple para recuperar y mostrar los datos de todos los productos. Indica al final cuantos productos hay.

```
DECLARE
  CURSOR c_productos IS
    SELECT * FROM productos;
    reg_productos  c_productos%rowtype;
    TOTAL NUMBER;
BEGIN
  OPEN c_productos;
  dbms_output.put_line('CODIGO PRODUCTO || '||'NOMBRE || '||'PROVEEDOR ||
'||'DESCRIPCIÓN || '||'CANTIDAD || '||'PRECIO VENTA || '||'COSTE ||');
  loop
    fetch c_productos INTO reg_productos;
    exit when c_productos%notfound;
    dbms_output.put_line(reg_productos.codigoproducto||' || '||reg_productos.nombre||' || '||
reg_productos.proveedor||' || '||reg_productos.descripcion||' || '||reg_productos.cantidad||' || '||
reg_productos.precioventa||' || '||reg_productos.coste);
  END loop;
  CLOSE c_productos;
SELECT COUNT(*) INTO  TOTAL FROM PRODUCTOS;
dbms_output.put_line(TOTAL);
END;
```

5- Usa el cursor del loop directamente en el bucle -- (lo hacemos con for)

```
DECLARE
  CURSOR c_productos IS
    SELECT * FROM productos;
BEGIN
  dbms_output.put_line('CODIGO PRODUCTO || '||'NOMBRE || '||'PROVEEDOR ||
'||'DESCRIPCIÓN || '||'CANTIDAD || '||'PRECIO VENTA || '||'COSTE ||');
 for reg_productos in c_productos loop
```

```
    dbms_output.put_line(reg_productos.codigoproducto||' || '||reg_productos.nombre||' || '||
reg_productos.proveedor||' || '||reg_productos.descripcion||' || '||reg_productos.cantidad||' || '||
reg_productos.precioventa||' || '||reg_productos.coste);
  END loop;
END;
```

6.- Usa ahora un while.

```
DECLARE
  CURSOR c_productos IS
    SELECT * FROM productos;
    reg_productos  c_productos%rowtype;
BEGIN
  OPEN c_productos;
  fetch c_productos INTO reg_productos;
  dbms_output.put_line('CODIGO PRODUCTO || '||'NOMBRE || '||'PROVEEDOR ||
'||'DESCRIPCIÓN || '||'CANTIDAD || '||'PRECIO VENTA || '||'COSTE ||');
  while c_productos%found loop
    dbms_output.put_line(reg_productos.codigoproducto||' || '||reg_productos.nombre||' || '||
reg_productos.proveedor||' || '||reg_productos.descripcion||' || '||reg_productos.cantidad||' || '||
reg_productos.precioventa||' || '||reg_productos.coste);
    fetch c_productos INTO reg_productos;
  END loop;
  CLOSE c_productos;
END;
```

7.- Dadas las siguientes tablas:

```
-- tabla para almacenar todos los alumnos de la
BD CREATE TABLE Alumnos
   (numMatricula NUMBER PRIMARY KEY,
   nombre VARCHAR2(15),
    apellidos VARCHAR2(30),
   titulacion VARCHAR2(15),
   precioMatricula NUMBER);

-- tabla para los alumnos de informática
CREATE TABLE AlumnosInf
(IDMatricula NUMBER PRIMARY KEY,
nombre_apellidos VARCHAR2(50),
precio NUMBER);
```

Inserte los siguientes datos de prueba en la tabla ALUMNOS.

numMatricula	*nombre*	*apellidos*	*titulacion*	*precioMatricula*
1	Juan	Álvarez	Administrativo	1000
2	José	Jiménez	Informatica	1200
3	Maria	Pérez	Administrativo	1000
4	Elena	Martínez	Informatica	1200

Construya un cursor que inserte sólo los alumnos de informática en la tabla ALUMNOSINF, teniendo en cuenta la estructura de esta tabla, asi por ejemplo, debe tener en cuenta que el atributo nombre_apellidos resulta de la concatenación de los atributos nombre y apellidos. Antes de la inserción de cada tupla en la tabla ALUMNOSINF debe mostrar por pantalla el nombre y el apellido que va a insertar.

```
--Inserte los siguientes datos de prueba en la tabla alumnos
INSERT INTO alumnos VALUES(1,'Juan','Álvarez','Administrativo',1000);
INSERT INTO alumnos VALUES(2,'José','Jiménez','Informática',1200);
INSERT INTO alumnos VALUES(3,'María','Pérez','Administrativo',1000);
INSERT INTO alumnos values(4,'Elena','Martínez','Informática',1200);

DECLARE
CURSOR c_alumnos IS
select * from alumnos;
BEGIN
  FOR reg_alumnos IN c_alumnos loop
    IF reg_alumnos.titulacion = 'Informática' THEN
      INSERT INTO alumnosinf
      values(reg_alumnos.numMatricula,reg_alumnos.nombre||'_'||
      reg_alumnos.apellidos,reg_alumnos.precioMatricula);
      dbms_output.put_line(reg_alumnos.numMatricula||' || '||reg_alumnos.nombre||'_'||
      reg_alumnos.apellidos||' || '||reg_alumnos.precioMatricula);
    end if;
  END loop;
END;
```

8. Dadas las siguientes tablas:

```
CREATE TABLE Tabla_Departamento (
Num_Depart Number(2) PRIMARY KEY,
Nombre_Depart VARCHAR2(15),
Ubicación VARCHAR2(15),
Presupuesto NUMBER(10,2),
Media_Salarios NUMBER(10,2),
Total_Salarios NUMBER(10,2));

CREATE TABLE Tabla_Empleado(
Num_Empleado Number(4) PRIMARY KEY,
Nombre_Empleado VARCHAR(25),
Categoria VARCHAR(10), -- Gerente, Comercial, …
Jefe Number(4),
Fecha_Contratacion DATE,
Salario Number(7,2),
Comision Number(7,2),
Num_Depart NUMBER(2),
FOREIGN KEY (Jefe) REFERENCES Tabla_Empleado,
FOREIGN KEY (Num_Depart) REFERENCES Tabla_Departamento);
```

a) Construya un bloque que calcule el presupuesto del departamento para el año próximo. Se almacenará el mismo en la tabla **Tabla_Departamento** en la columna **Presupuesto**. Hay que tener en cuenta las siguientes subidas de sueldo:

Gerente	+	20%
Comercial	+	15%

Los demás empleados que no estén en ninguna de las categorías anteriores se les subirá el sueldo un 10%.

b) Construya un bloque que actualice el campo Total_Salarios y el campo Media_Salarios de la tabla **Tabla_Departamento**, siendo el total la suma del salario de todos los empleados, igualmente con la media. Para ello:

- Cree un cursor **C1**, que devuelva todos los departamentos
- Cree un cursor **C2**, que devuelva el salario y el código de todos los empleados de su departamento.

a)

```
DECLARE
  p_total NUMBER := 0;
  sub NUMBER;
  CURSOR c_emp IS
  SELECT * FROM tabla_empleado;
BEGIN
   FOR reg_emp IN c_emp loop
     IF reg_emp.num_depart = 01 then   -- se tiene que indicar el número de depart aquí y abajo
       IF reg_emp.categoria = 'gerente' THEN
```

```
        sub := reg_emp.salario * 0.2;
        p_total := p_total + reg_emp.salario + sub;
       END IF;
       IF reg_emp.categoria = 'comercial' THEN
        sub := reg_emp.salario * 0.15;
        p_total := p_total + reg_emp.salario + sub;
       END IF;
       IF reg_emp.categoria != 'comercial' AND reg_emp.categoria != 'gerente' THEN
        sub := reg_emp.salario * 0.1;
        p_total := p_total + reg_emp.salario + sub;
       END IF;
      END IF;
   END loop;
  dbms_output.put_line(p_total);
  UPDATE tabla_departamento
  SET presupuesto = p_total where num_depart = 01; -- se tiene que indicar el número de
depart aquí y arriba
END;


b) DECLARE
stotal NUMBER;
countM number;
CURSOR c_dept IS SELECT * FROM tabla_departamento;
CURSOR c_emp IS SELECT * FROM tabla_empleado;
BEGIN
  FOR reg_dept IN c_dept loop --recorre todos los departamentos
    dbms_output.put_line('Departamento '||reg_dept.num_depart);
    stotal :=0;
    countM :=0; --pongo a 0 el total y contador por cada departamento
      FOR reg_emp IN c_emp loop -- recorre todos los empleados
        IF reg_emp.num_depart = reg_dept.num_depart THEN -- si el num_depart del
empleado coincide con el depart que se está recorriendo, se muestra
          stotal := stotal + reg_emp.salario;
          countM := countM+1;
        END IF;
      END loop;
     dbms_output.put_line('El salario total de este departamento es '||stotal||' con una media de
'||stotal/countm||' por empleado'); -- muestro por pantalla los resultados
    UPDATE tabla_departamento
    SET total_salarios = stotal WHERE num_depart = reg_dept.num_depart;       --updates
para meterlo en las tablas
    UPDATE tabla_departamento
    set media_salarios = stotal/countM where num_depart = reg_dept.num_depart;
  end loop;
END;
```

9.- Cree una tabla NOTAS con los atributos que estime necesarios y construya un bloque que inserte en la tabla NOTAS cuatro filas para cada alumno matriculado, estas filas corresponderán a las tres convocatorias ordinarias y la última para la convocatoria extraordinaria de junio. Antes de insertar se comprobará que no están ya creadas. Las filas deberán inicializarse a nulo.

```
CREATE TABLE alumnos_notas(
numeromatricula  NUMBER PRIMARY KEY,
eval1 number,
eval2 number,
eval3 number,
extraordinaria NUMBER
);

DECLARE
CURSOR c_alumnosnotas IS
select nummatricula from alumnos1;
numalumno alumnos1.nummatricula%type;

BEGIN

OPEN c_alumnosnotas;
FETCH c_alumnosnotas INTO numalumno;
while  c_alumnosnotas%found loop
       INSERT INTO alumnos_notas values (numalumno,null,null,null,null);
     FETCH c_alumnosnotas INTO numalumno;
  END loop;
close c_alumnosnotas;
END;
```