**PRÁCTICA 1.**

**VARIABLES Y BLOQUES EN PL/SQL**.

**1. Evalúa cada una de las declaraciones de variables siguientes y determina cuales no son válidas y por qué.**

DECLARE

**v\_id NUMBER(4);** → CORRECTO

**v\_x, v\_y, v\_z VARCHAR2(10);**  → Incorrecto, porque no puede declarar múltiples variables a la vez, aunque puede ponerlos en la misma línea, siempre ya que tiene tipo de datos y punto y coma después de cada uno.

**v\_birthdate DATE NOT NULL;** → Incorrecto, porque si no va a ser nulo, entonces debe tener un valor inicial, así

**v\_in\_stock BOOLEAN := 1;** → Incorrecto, porque el tipo de datos y el valor deben coincidir. Debería ser BOOLEAN: = verdadero o NUMBER: = 1.

**emp\_record emp\_record\_type; →** CORRECTO

**TYPE name\_table\_type IS TABLE OF VARCHAR2(20); INDEX BY BINARY\_INTEGER; dept\_name\_table name\_table\_type; ....**

**2. Indica el valor de las variables del siguiente bloque PL/SQL en cada momento:**

DECLARE v\_customer VARCHAR2(50) := ‘ Womansport ‘;

v\_weight NUMBER(3) := 600;

v\_message VARCHAR2(255) := ‘ Producto 10012 ‘;

BEGIN DECLARE v\_customer NUMBER(7) := 201;

v\_message VARCHAR2(255) := ‘ Producto 11001 ‘;

v\_new\_locn VARCHAR2(50) := ‘ Europa ‘;

BEGIN v\_weight := v\_weight + 1;

v\_new\_locn := ‘ Oeste ‘ || v\_new\_locn;

END;

v\_weight := v\_weight + 1; v\_message := v\_message || ‘ está en stock ‘; v\_customer := ‘ Carmen ‘ || v\_customer ; v\_new\_locn := ‘ Oeste ‘ || v\_new\_locn;

END;

Introducción a PL/SQL

**3. Construye un bloque PL/SQL que pida el precio de un producto y el valor del IVA a aplicar, e imprima el valor del total del producto.**

set serveroutput on;

declare

nom\_art articulos.articulo%type;

precio articulos.precio\_venta%type;

iva number(10):= 1.21;

total number(10);

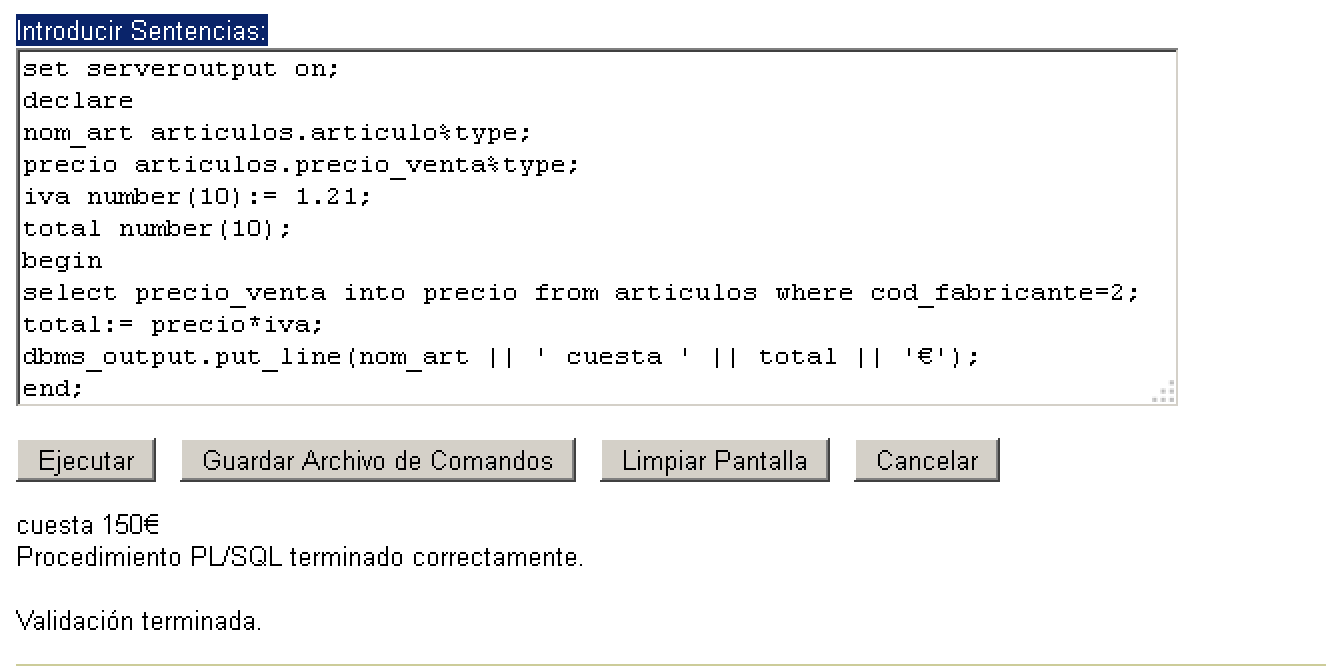
begin

select precio\_venta into precio from articulos where cod\_fabricante=2;

total:= precio\*iva;

dbms\_output.put\_line(nom\_art || ' cuesta ' || total || '€');

end;



**4. Construye un bloque PL/SQL que seleccione el artículo de mayor PVP en la tabla ARTICULOS y almacene su valor en una variable de SQL para imprimirlo a continuación.**

set serveroutput on;

declare

nom\_art articulos.articulo%type;

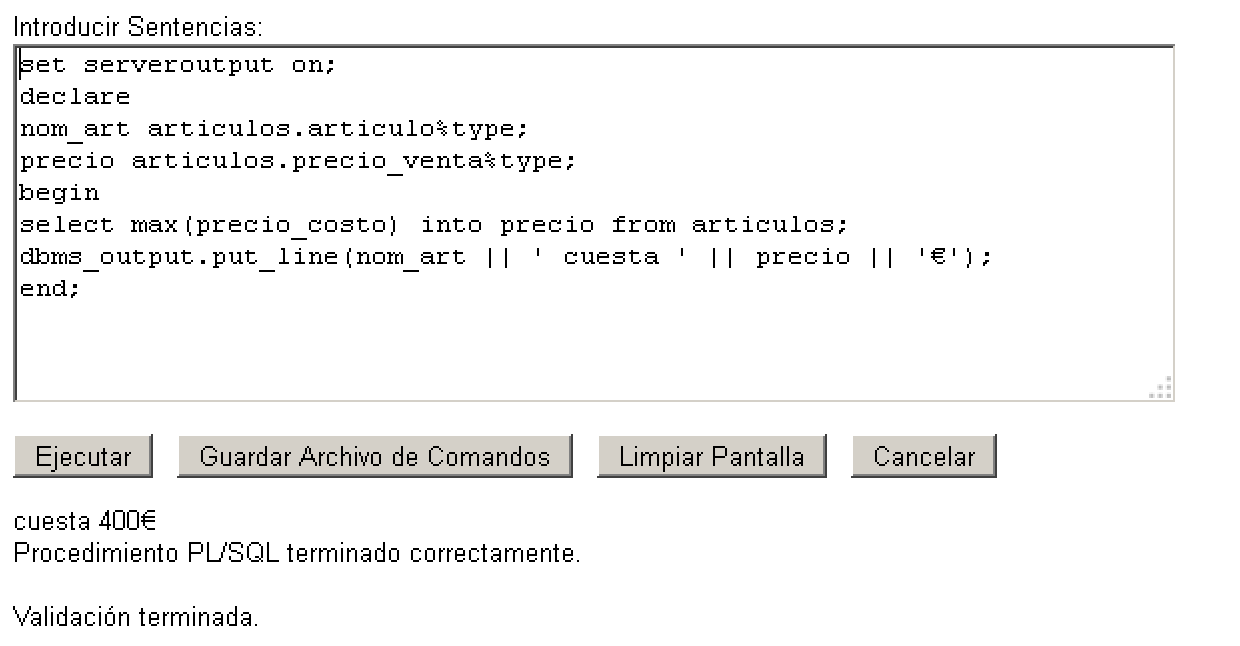
precio articulos.precio\_venta%type;

begin

select max(precio\_costo) into precio from articulos;

dbms\_output.put\_line(nom\_art || ' cuesta ' || precio || '€');

end;



**5. Crea y almacena en un fichero un bloque PL/SQL que inserte un nuevo artículo en la tabla ARTICULOS. Los datos del código y nombre deben pedirse previamente por teclado para darlos como argumentos.**

declare

narticulo varchar2(20):=&narticulo;

nfabricante number(38):=&nfabricante;

npeso number(38):=&npeso;

ncategoria varchar2(10):=&ncategoria;

nprecio\_venta number(38):=&nprecio\_venta;

nprecio\_costo number(38):=&nprecio\_costo;

nexistencias number(38):=&nexistencias;

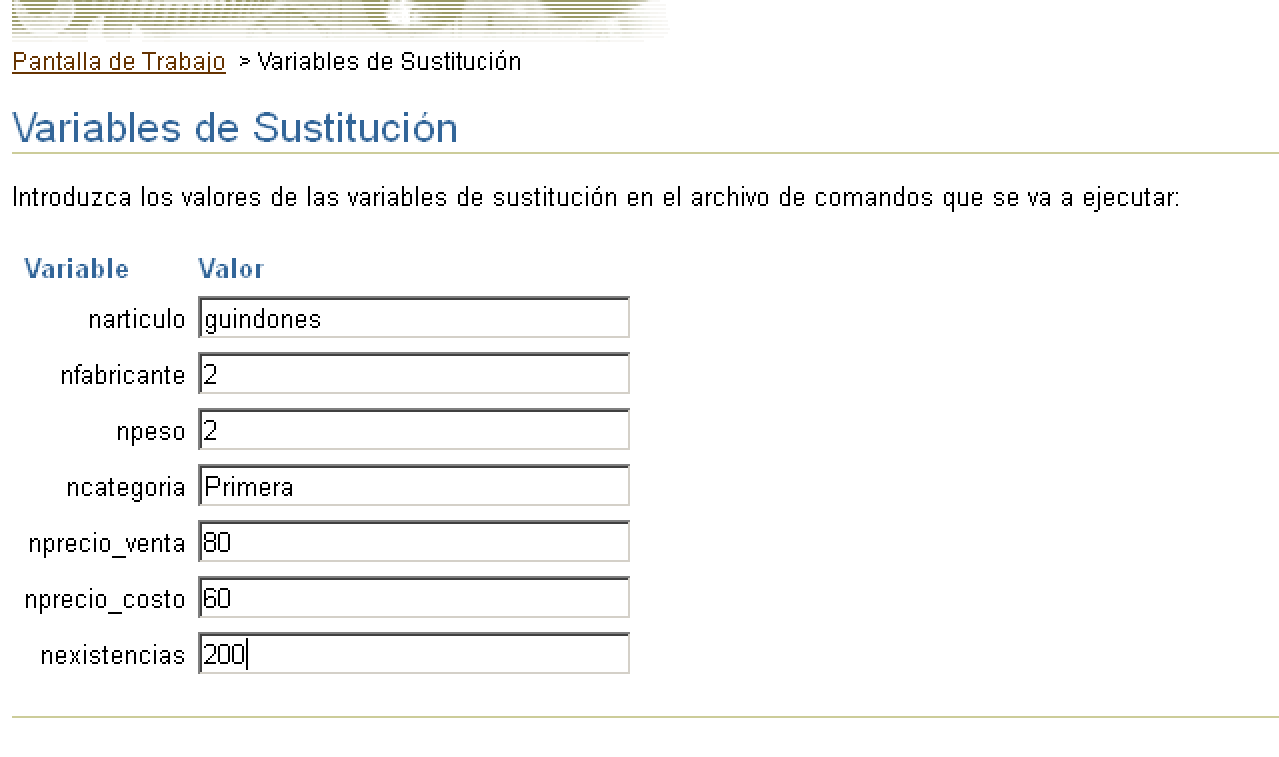
begin

insert into articulos (ARTICULO,COD\_FABRICANTE,PESO,CATEGORIA,PRECIO\_VENTA,PRECIO\_COSTO,EXISTENCIAS)

VALUES (narticulo,nfabricante,npeso,ncategoria,nprecio\_venta,nprecio\_costo,nexistencias);

DBMS\_OUTPUT.PUT\_LINE('Se ha introducido nuevo producto ');

END;

****

**6. Un bloque PL/SQL debe actualizar la dirección de un cliente en la tabla CLIENTES. Para ello, el bloque pedirá el código del cliente y su nueva dirección.**

Declare

nif number(38) := &amp;nif;

nueva\_dir VARCHAR2(30) := &amp;nueva\_dir;

Begin

UPDATE Cliente

set DIRCLI = nueva\_dir

where NIFCLI = nif;

DBMS\_OUTPUT.PUT\_LINE(&#39;La nueva dirección es: &#39; || nueva\_dir);

END;

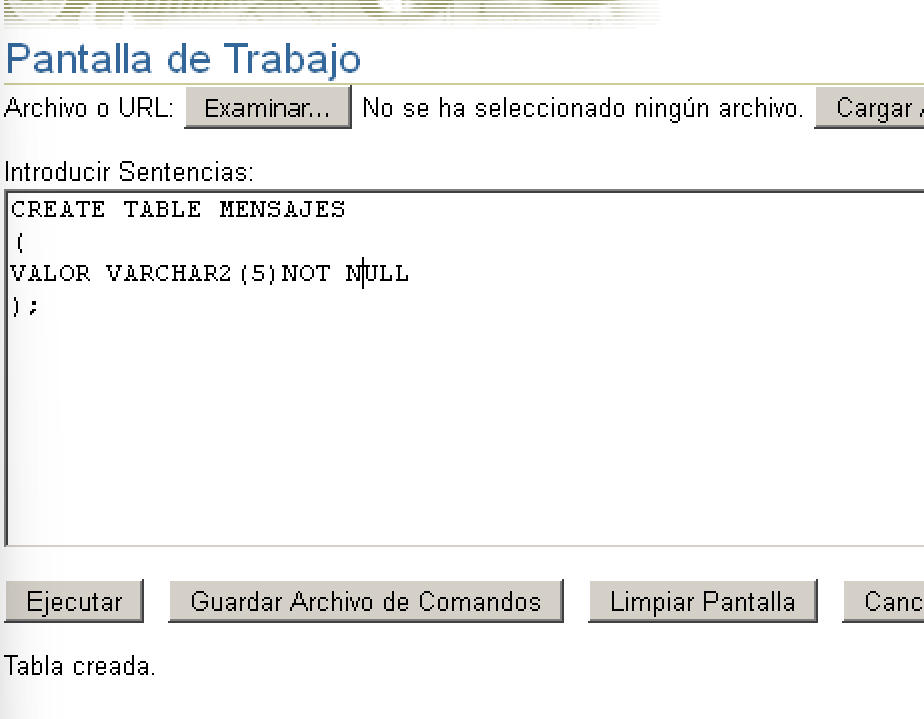
**7. Crea un bloque que borre todos los proveedores de un país que se pedirá por teclado. Debe imprimirse el número de proveedores que se han borrado.**

**PRÁCTICA 2. ESTRUCTURAS DE CONTROL EN BLOQUES**

**PL/SQL**

**1. 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 VARCHAR2(5)NOT NULL

);

BEGIN

FOR numero IN 1..10 LOOP

IF numero &lt;&gt; 4 AND numero &lt;&gt; 5 THEN

INSERT INTO MENSAJES VALUES (numero);

END IF;

END LOOP;

END;

**2. Un bloque PL/SQL debe aceptar el código de un artículo y actualizar su precio según las siguientes indicaciones. Si el artículo tiene un PVP menor a 100 ptas., su precio debe ser aumentado en 5 ptas. Si está entre 100 y 1.000 ptas. su precio subirá 15 y si excede las 1.000 ptas., el precio del producto debe subir 50 ptas. Si el PVP fuese NULL, el aumento sería 0.**

ALTER TABLE ARTICULOS

ADD ID\_COD VARCHAR2(10) ;

INSERT INTO ARTICULOS (ARTICULO, COD\_FABRICANTE, PESO,CATEGORIA,

PRECIO\_VENTA, PRECIO\_COSTO, EXISTENCIAS, ID\_COD)

VALUES (‘MIEL’, 2, 3,‘Primera’, 20, 5, 34, 263);

DECLARE

codart articulos.id%TYPE :=&amp;codart;

precio articulos.precio\_costo%TYPE;

precio\_nuevo articulos.precio\_costo%TYPE;

pvp articulos.precio\_venta%TYPE;

BEGIN

SELECT precio\_venta INTO pvp FROM ARTICULOS where id=codart;

SELECT precio\_costo INTO precio FROM ARTICULOS where id=codart;

IF (pvp &lt; 100) THEN

precio\_nuevo := precio + 5;

ELSIF (pvp BETWEEN 100 AND 1000) THEN

precio\_nuevo := precio + 15;

ELSIF (pvp &gt; 1000) THEN

precio\_nuevo := precio + 50;

ELSE

precio\_nuevo := precio;

END IF;

UPDATE ARTICULOS SET precio\_costo=precio\_nuevo where id=codart;

DBMS\_OUTPUT.PUT\_LINE(‘Se ha actualizado el precio de art nº ‘ || codart || ‘. Precio

anterior: ‘|| precio || ‘. Precio nuevo: ‘ || precio\_nuevo);

END;

### **Ejercicio 02 : Sentencia Loop While**

**Desarrollar un programa que sume «n» veces un número aleatorio.**

**Imprimir en pantalla los números aleatorios generados y la suma obtenida**

**La cantidad de iteraciones deberá generarse de manera aleatoria, considerando los valores del 1 al 10. Los números aleatorios generados en cada iteración deberán ser del 10 al 100.**

**DECLARE**

**v\_sum NUMBER := 0;**

**v\_x NUMBER :=0;**

**v\_num NUMBER;**

**v\_limit NUMBER;**

**BEGIN**

**v\_limit := FLOOR(DBMS\_RANDOM.value(1,10));**

**LOOP**

**v\_x := v\_x + 1;**

**v\_num := FLOOR(DBMS\_RANDOM.value(10,100));**

**DBMS\_OUTPUT.PUT\_LINE('Número aleatorio '||v\_x||' -> '|| v\_num);**

**v\_sum := v\_sum + v\_num;**

**EXIT WHEN v\_x = v\_limit;**

**END LOOP;**

**DBMS\_OUTPUT.PUT\_LINE('Suma Total = '||v\_sum);**

**END;**

### **Ejercicio 03 : Sentencia FOR**

**Desarrollar un programa que devuelva la cantidad de números múltiplos de 3 que existen del 1 al 100.**

**DECLARE**

**v\_multiple\_3 NUMBER := 0;**

**BEGIN**

**FOR v\_num IN 1..100 LOOP**

**IF (MOD(v\_num,3)=0) THEN**

**v\_multiple\_3 := v\_multiple\_3 + 1;**

**END IF;**

**END LOOP;**

**DBMS\_OUTPUT.PUT\_LINE('Números multiplos de 3 del 1 al 100: '||v\_multiple\_3);**

**END;**

**-- Output:**

**-- Números multiplos de 3 del 1 al 100: 3**

### **Ejercicio 04: Cursores**

**Elabore un bloque anónimo PL/SQL que almacene en un objeto cursor la lista de empleados (código, nombres, apellidos y salario) que tiene un sueldo menor e igual $ 2,800. Mostrar el resultado por cada registro. Utilizar el esquema de base de datos HR (Human Resources) que viene por defecto en Live SQL.**

**DECLARE**

**CURSOR employees\_cursor IS**

**SELECT employee\_id,first\_name,last\_name,salary FROM hr.employees WHERE salary<=2800;**

**v\_id hr.employees.employee\_id%TYPE;**

**v\_name hr.employees.first\_name%TYPE;**

**v\_surname hr.employees.last\_name%TYPE;**

**v\_salary hr.employees.salary%TYPE;**

**v\_num NUMBER := 0;**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE(LPAD('-',50,'-'));**

**DBMS\_OUTPUT.PUT\_LINE('INFORME DE EMPLEADOS');**

**DBMS\_OUTPUT.PUT\_LINE(LPAD('-',50,'-'));**

**OPEN employees\_cursor;**

**LOOP**

**FETCH employees\_cursor INTO v\_id,v\_name,v\_surname,v\_salary;**

**EXIT WHEN employees\_cursor%NOTFOUND;**

**v\_num := v\_num+1;**

**DBMS\_OUTPUT.PUT\_LINE('[' || v\_num || '] Empleado N° '|| v\_id || ' - ' || v\_name || ' ' || v\_surname || ' tiene un sueldo de' ||TO\_CHAR(v\_salary,'$99,999.00'));**

**END LOOP;**

**CLOSE employees\_cursor;**

**END;**

### **Ejercicio 05: Sentencia IF**

**Elaborar un bloque anónimo PL/SQL que devuelva un reporte de los empleados que tienen un salario menor e igual a $ 2,800.00. El reporte debe mostrarse agrupado por departamentos. Los valores de la columna deben ser por ancho fijo.**

**DECLARE**

**CURSOR dpto\_employees\_cursor IS**

**SELECT e.department\_id,d.department\_name,e.employee\_id,e.first\_name,e.last\_name,e.salary**

**FROM hr.employees e**

**INNER JOIN hr.departments d ON e.department\_id=d.department\_id**

**WHERE salary<2800**

**ORDER BY d.department\_name DESC, salary DESC;**

**v\_last\_department\_id hr.departments.department\_id%TYPE:=0;**

**v\_num NUMBER := 0;**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE(LPAD('-',70,'-'));**

**DBMS\_OUTPUT.PUT\_LINE('INFORME DE EMPLEADOS POR DEPARTAMENTOS');**

**DBMS\_OUTPUT.PUT\_LINE(LPAD('-',70,'-'));**

**FOR emp IN dpto\_employees\_cursor**

**LOOP**

**IF v\_last\_department\_id<>emp.department\_id THEN**

**DBMS\_OUTPUT.PUT\_LINE(chr(13)||'DEPARTAMENTO: ' || UPPER(emp.department\_name));**

**DBMS\_OUTPUT.PUT\_LINE(LPAD('-',70,'-'));**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE(rpad(emp.employee\_id, 10) || rpad(emp.first\_name|| ' ' || emp.last\_name, 20) ||TO\_CHAR(emp.salary,'$999,999.00'));**

**END IF;**

**v\_last\_department\_id:=emp.department\_id;**

**END LOOP;**

**END;**

**3. Añade una nueva columna a la tabla ARTICULOS para almacenar cadenas de ‘\*’. Crea un bloque PL/SQL que pida el código de un artículo y almacene un ‘\*’ por cada 500 ptas. del precio del producto en la nueva columna de la tabla.**

declare

codigo ARTICULOS.ID\_COD%TYPE:=&colocar\_cadena;

precio ARTICULOS.PRECIO\_VENTA%TYPE;

ASTER ARTICULOS.ARTICULO%TYPE;

begin

select precio\_venta into precio FROM ARTICULOS WHERE ID\_COD=codigo;

WHILE precio >500 LOOP

cadena:= cadena + "\*";

precio:= precio-500;

END LOOP;

INSERT INTO ARTICULOS(ASTER) values(cadena);

END;

**PRÁCTICA 3. CURSORES EN BLOQUES PL/SQL**.

**1. Crea un bloque que almacene en la tabla AUX\_ARTICULOS (previamente debe**

**vaciarse) un número dado de los artículos con mayor PVP de los existentes en la tabla**

**ARTICULOS. El número de artículos a almacenarse debe ser dado por teclado.**

**2. Escribe un bloque que recupere todos los proveedores por países. El resultado debe**

**aparecer en una nueva tabla que permita almacenar datos del tipo:**

**‘Proveedor: ONCE – País: ESPAÑA’**

**Utiliza un cursor para recuperar cada país de la tabla de PROVEEDORES y pasar dicho**

**país a un cursor que obtenga el nombre de los proveedores en él. Una vez que se tiene el**

**nombre del proveedor y su país, debe añadirse a la nueva tabla en el formato**

**especificado.**

**3. Modifica el ejercicio 3 de la práctica 2 mediante el uso de un cursor así como las**

**funcionalidades FOR UPDATE y WHERE CURRENT OF para éste.**

**PRÁCTICA 4. EXCEPCIONES EN BLOQUES PL/SQL.**

**1. Escribe un bloque PL/SQL que seleccione el nombre de un proveedor a partir de una ciudad dada por teclado. El resultado de cada consulta debe almacenarse en una nueva tabla con una sola columna que sea una cadena de caracteres, de forma que:**

**a) Si la ciudad introducida sólo proporciona una fila, la entrada a la nueva tabla será del tipo: ‘ Nombre\_Proveedor – Ciudad ‘.**

**b) Si la ciudad introducida no proporciona ningún proveedor, crear la excepción apropiada e insertar en la nueva tabla el mensaje: ‘No existen proveedores en Ciudad‘.**

**c) Si la ciudad dada tiene varios proveedores, añadir la excepción adecuada e insertar en la nueva tabla el mensaje: ‘ Existen varios proveedores en Ciudad ‘.**

**d) En cualquier otro caso, añadir una nueva excepción y añadir en la nueva tabla un nuevo mensaje: ‘Se ha producido un error ‘.**

**2. Modificar la solución al ejercicio 1.6 añadiendo un controlador de excepciones que controle cuando se introduce un código de cliente que no existe.**

**3. Escribe un bloque PL/SQL que imprima el número de artículos cuyo PVP esté entre 100 ptas. más o menos que un valor dado por teclado. En caso de que no haya artículos con precio en este rango, se debe imprimir un mensaje indicándolo. Si hay más uno o más artículos en este rango, el mensaje debe indicar cuántos artículos son los que están en este rango de valores y en caso de producirse cualquier otro error, debe crearse la excepción adecuada que indique la existencia de un error.**

**PRÁCTICA 5. PROCEDIMIENTOS Y FUNCIONES EN**

**PL/SQL**

**1. Crea un procedimiento que inserte un nuevo artículo en la tabla ARTICULOS.**

**2. Crea un procedimiento que permita modificar la descripción de un artículo. Incluye el control de errores con las excepciones oportunas.**

**3. Escribe un procedimiento para borrar un artículo de la tabla ARTICULOS. Añade el control de errores con las excepciones adecuadas.**

**4. Obtén un procedimiento para consultar la tabla de ARTICULOS y recupere el nombre y PVP para un código de artículo dado. Proporciona el control de errores adecuado.**

**5. Escribe una función que devuelva la descripción de un artículo cuyo código se pedirá por teclado.**

**6. Crea una función para obtener el precio de un artículo descontando el IVA. Para ello, la función aceptará el PVP y el IVA y devolverá el precio sin IVA.**

**7. Escribe un procedimiento para insertar una nueva factura en la tabla FACTURAS. Este procedimiento debe hacer las llamadas a funciones que permitan indicar si el código de artículo y el código de cliente existen en las tablas correspondientes. Las funciones a crear devolverán un valor booleano. Si las llamadas a las dos funciones devuelven un valor TRUE debe insertarse la nueva factura como un nuevo registro. Pero si alguna de**

**las llamadas retorna un valor FALSE, el procedimiento debe indicar al usuario con el mensaje apropiado.**