I.E.S. Jacarandá
Gestión de Base de Datos
PL/SQL
Relación de ejercicios 5

1. Realiza un procedimiento que reciba un código de un cliente y muestra por pantalla su nombre y el pueblo y la provincia donde vive.

```
CREATE OR REPLACE PROCEDURE EJER1 (codigo CLIENTES.CODCLI%TYPE)
IS
nombre_cliente clientes.nombre%TYPE;
nombre_pueblo pueblos.nombre%TYPE;
nombre_provincia provincias.nombre%TYPE;
BEGIN
SELECT C.Nombre, P.NOMBRE, PRO.NOMBRE INTO nombre_cliente, nombre_pueblo, nombre_provincia
FROM CLIENTES C, PUEBLOS P, PROVINCIAS PRO
WHERE C.CODPUE = P.CODPUE AND P.CODPRO = PRO.CODPRO AND C.CODCLI = codigo;

dbms_output.put_line ('El cliente ' || nombre_cliente || ' vive en el pueblo ' || nombre_pueblo || ' y provincia ' || nombre_provincia);

exception
when no_data_found then
dbms_output.put_line('No existe');

END;
/

BEGIN
EJER1(3);
end;
```

2. Realiza una función devolver_gastos que reciba un nombre de un cliente y devuelva la suma de todo lo que se ha gastado en todas sus facturas.

```
CREATE OR REPLACE FUNCTION devolver_gastos(nombre_cliente clientes.nombre%TYPE) return number
IS
dinerito NUMBER(8);
BEGIN
SELECT sum(lineas_fac.PRECIO * lineas_fac.CANT) INTO dinerito
FROM clientes, facturas, lineas_fac
WHERE clientes.codcli = facturas.codcli
and facturas.codfac = lineas_fac.codfac
and clientes.NOMBRE = nombre_cliente;
dbms_output.put_line('El cliente ' || nombre_cliente || ' no tiene facturas');
return dinerito;
end;
/

declare
aux number;
begin
aux := devolver_gastos('FOLCH CASABO, MARIA MERCEDES');
dbms_output.put_line(aux);
end;
/
```

3. Realiza un procedimiento llamado HallarNumPueblos que recibiendo un nombre de una provincia, muestre en pantalla el número de pueblos de dicha provincia.

```
CREATE OR REPLACE PROCEDURE HallarNumPueblos(nombre_ provincias.nombre%TYPE)
IS
numeros NUMBER(2);
BEGIN
SELECT count(pueblos.codpue) INTO numeros
FROM pueblos, provincias
WHERE provincias.codpro = pueblos.codpro
and provincias.nombre = nombre_;
DBMS_OUTPUT.PUT_LINE(numeros);
end;
/

SET SERVEROUTPUT ON
begin
HallarNumPueblos('AVILA');
end;
/
```

4. Hacer un procedure que reciba tendrá tres parámetros: un código de cliente, la provincia y el pueblo. Los dos últimos parámetros serán de entrada y salida y deberá devolver el nombre de la provincia y el nombre del prueblo en los parámetros.

```
CREATE OR REPLACE PROCEDURE EJ4(vcod CLIENTES.CODCLI%TYPE, vnompro in out PROVINCIAS.NOMBRE%TYPE,
vnompue in out PUEBLOS.NOMBRE%TYPE)
IS
vcodpue PUEBLOS.CODPUE%TYPE;
vcodpro PROVINCIAS.CODPRO%TYPE;
BEGIN
SELECT codpue into vcodpue from clientes where codcli = vcod;
select nombre, codpro into vnompue, vcodpro from pueblos where codpue = vcodpue;
select nombre into vnompro from provincias where codpro = vcodpro;
exception
when no_data_found then
vnompro := null;
vnompue := null;
END;
/

SET SERVEROUTPUT ON
declare
nombreprovincia PROVINCIAS.NOMBRE%TYPE;
nombrepueblo PUEBLOS.NOMBRE%TYPE;
BEGIN
EJ4(6, nombreprovincia, nombrepueblo);
dbms_output.put_line('El cliente vive en ' || nombrepueblo || ' provincia de ' || nombreprovincia);
END;
/
```

5. Codificar un procedimiento que permita borrar un cliente cuyo número se pasará en la llamada.

```
ALTER TABLE FACTURAS DISABLE CONSTRAINT FK0_54;

CREATE OR REPLACE PROCEDURE borrarCli(codigoCli CLIENTES.CODCLI%TYPE)
IS
BEGIN
DELETE
FROM CLIENTES
WHERE CODCLI = codigoCli;
END;
/

BEGIN
borrarCli(6);
END;
/
```

6. Escribir un procedimiento que modifique la provincia de un pueblo. El procedimiento

recibirá como parámetros el código del pueblo y el nombre de la provincia nueva.

```
CREATE OR REPLACE PROCEDURE modPro(codigo PUEBLOS.CODPUE%TYPE, nombrePro PROVINCIAS.NOMBRE%TYPE)
IS
BEGIN
UPDATE PROVINCIAS
SET PROVINCIAS.NOMBRE = nombrePro
WHERE PROVINCIAS.CODPRO = (SELECT PUEBLOS.CODPRO FROM PUEBLOS WHERE PUEBLOS.CODPUE = codigo);
END;
/

BEGIN
modPro('01651', 'SUPER');
END;
/
```

7. Crear un procedimiento que en la tabla artículos incrementar el precio un porcentaje que se le pasará como argumento, junto con el código del artículo. Este procedimiento también recibirá un parámetro de entrada salida en el que se deberá devolver la cantidad incrementada. Realizar un procedure o un bloque anónimo para llamar a este procedure.

```
CREATE OR REPLACE PROCEDURE EJ7(codigoAr ARTICULOS.CODART%TYPE, porcentajeAr NUMBER, cantidadAr in out
NUMBER)
IS
vprecio ARTICULOS.PRECIO%TYPE;
BEGIN
SELECT precio into vprecio from articulos where codart = codigoAr;
UPDATE ARTICULOS
SET precio = ((vprecio * porcentajeAr) + vprecio)
WHERE articulos.codart = codigoAr;
cantidadAr := vprecio * porcentajeAr;
END;
/

declare
cantidad NUMBER;
begin
EJ7('L76424', 0.2, cantidad);
DBMS_OUTPUT.PUT_LINE('Esta es la cantidad incrementa de mi arti-culo que he modificadoooo: ' || cantidad);
END;
/
```

8. Haz una función llamada DevolverCodProvincia que reciba el nombre de una provincia y devuelva su código.

```
CREATE OR REPLACE FUNCTION EJ8(nombrePro PROVINCIAS.NOMBRE%TYPE) RETURN PROVINCIAS.CODPRO%TYPE
IS
codigoPro PROVINCIAS.CODPRO%TYPE;
BEGIN
SELECT codpro into codigoPro from provincias where nombre = nombrePro;
return codigoPro;
END;
/

begin
DBMS_OUTPUT.PUT_LINE(EJ8('BADAJOZ'));
END;
/
```

9. Crear una función que inserte un cliente en la tabla clientes. Su código será superior a los existentes y el l resto de los valores se les pasará como argumento. Debe devolver el código del cliente insertado. Además deberá garantizar que todos los datos se almacenan en mayúscula independientemente de como los reciba en el parámetro.

```
CREATE OR REPLACE FUNCTION EJ9(nombreCli CLIENTES.NOMBRE%TYPE, direccionCli CLIENTES.DIRECCION%TYPE,
codigoPostCli CLIENTES.CODPOSTAL%TYPE, codPueCli CLIENTES.CODPUE%TYPE) RETURN CLIENTES.CODCLI%TYPE
IS
codigoCli CLIENTES.CODCLI%TYPE;
BEGIN
SELECT max(codcli)+1 INTO codigoCli from clientes;
INSERT INTO clientes(codcli, nombre, direccion, codpostal, codpue)
VALUES (codigoCli, UPPER(nombreCli), UPPER(direccionCli), UPPER(codigoPostCli), UPPER(codPueCli));
RETURN codigoCli;
END;
/

BEGIN
DBMS_OUTPUT.PUT_LINE(EJ9('MORENA CORBATA, JUAN', 'SANT RAFAEL, 158-18', '38894', '31289'));
END;
/
```

10. Realiza un procedimiento Mostrarsodatosarticulosque reciba el nombre de un artículo al revés y muestre toda la información de ese artículo. Es recormendable realizar una función para darle la vuelta al artículo.