Ejercicio-1

Codificar una función que reciba por teclado una cadena y la visualice al revés, A continuación realizar un bloque para comprobar que funciona.

Solución

```
CREATE OR REPLACE FUNCTION fcadenaReves(vcadena VARCHAR2) RETURN varchar2
   IS vcadReves VARCHAR2(80);
BEGIN
   FOR i IN REVERSE 1..LENGTH(vcadena) LOOP
      vcadReves:=vcadReves||  SUBSTR (vcadena,i,1);
   END LOOP;
   RETURN vcadReves;
END fcadenaReves;

--Bloque para probar la función
ACCEPT cadena PROMPT 'Introduce una cadena de caracteres'
DECLARE
   resultado varchar2(80);
BEGIN
   DBMS_OUTPUT.PUT_LINE ('La cadena es: '||'&cadena');
   resultado:= fcadenaReves('&cadena');
   DBMS_OUTPUT.PUT_LINE ('La cadena al revés es : '||resultado);
END;
```

Ejercicio-2

Escribir una función que reciba una fecha y devuelva, en número, solo el año correspondiente a esa fecha. Escribir un bloque PL/SQL que haga uso de la función anterior. Por ejemplo, a partir de la fecha de hoy, que devuelve el año y lo visualice.

Solución

```
CREATE OR REPLACE FUNCTION fanio     (fecha DATE)    RETURN NUMBER
AS
   vanio NUMBER (4);
BEGIN
   vanio:= TO_NUMBER(TO_CHAR(fecha, 'YYYY'));
   RETURN vanio;
END fanio;

--Bloque para probar la función
DECLARE
   anio         NUMBER(4);
```

```
    fecha       date;
BEGIN
    fecha:= SYSDATE;          -- como prueba introduzco la fecha del sistema actual
    anio:= fanio(fecha);      --invoco a la función f_anio pasándole la fecha
    DBMS_OUTPUT.PUT_LINE ('En la fecha: '|| fecha || ' el año es : '|| anio);
END;
```

Ejercicio-3

Diseñar una función llamada **devuelveNumemp** que permita devolver el **número** de un determinado empleado a partir de su apellido, el departamento y el oficio que tiene.

Diseñar a continuación un bloque que permita comprobar la función anterior

Solución

```
CREATE OR REPLACE FUNCTION devuelveNumemp (papellido VARCHAR2, pdepnume NUMBER, poficio
VARCHAR2) RETURN empleado.numemp%TYPE
IS
    vnumemp empleado.numemp%TYPE;
BEGIN
    -- para evitar problemas de mayúsculas o minúsculas usamos el UPPER
    SELECT numemp INTO vnumemp FROM empleado
    WHERE UPPER(apell)=UPPER(papellido) AND UPPER(oficio)=UPPER(poficio) AND depnume=
    pdepnume;
    return vnumemp;
END devuelveEmpno;

--Bloque para probar la función

--Se supone que los datos se introducirán correctamente
ACCEPT papellido PROMPT 'Introduce el nombre:  '
ACCEPT pdepnume  PROMPT 'Introduce el número de departamento: '
ACCEPT poficio PROMPT 'Introduce el oficio  :'
DECLARE
      vnumeEmp empleado.numemp%TYPE;
BEGIN
      vnumeEmp := devuelveNumemp ('&papellido', &pdepnume , '&poficio');
      DBMS_OUTPUT.PUT_LINE ('El número de empleado de '|| '&papellido' ||
        ' del departamento ' ||&pdepnume||' y de oficio '|| '&poficio'||' es '|| vnumeEmp);
END;
```

Ejercicio-4

Crear un procedimiento llamado **nuevoEmpleado** para insertar un empleado nuevo en la tabla empleado.

El procedimiento llamará a una función llamada **validarDept** para comprobar que el departamento del nuevo empleado existe en la tabla departamento.

Solución

```
CREATE OR REPLACE FUNCTION validaDepto (pnumerodept IN departamento.numedep %TYPE)
      RETURN BOOLEAN IS
      existe CHAR;
BEGIN
   SELECT 'S' INTO existe FROM departamento WHERE numedep = pnumerodept;
      RETURN (TRUE);
   EXCEPTION
      WHEN NO_DATA_FOUND THEN
            RETURN (FALSE);
END validaDepto;


CREATE OR REPLACE PROCEDURE nuevoEmpleado (numero empleado.numemp%TYPE, nombre
   empleado.apell%TYPE, departa empleado.depnume%TYPE) IS
BEGIN
   IF validaDepto(departa) THEN
         INSERT INTO empleado (numemp,apell,depnume) VALUES (numero,nombre,departa);
   ELSE
         DBMS_OUTPUT.PUT_LINE ('Departamento inválido ');
   END IF;
END nuevoEmpleado;

--Bloque para probar la función
BEGIN
   nuevoEmpleado (11,'ANA',20);
END;
```