Ejercicios PL/SQL

1. Escribir una función que reciba dos números y devuelva su suma. A continuación, escribir un procedimiento que muestre la suma al usuario.

2. Codificar un procedimiento que reciba una cadena de texto y la visualice al revés.

3. Escribir una función que reciba una fecha y devuelva el año de la fecha (como número).

4. Dado el siguiente procedimiento:

   ```
   CREATE OR REPLACE PROCEDURE crear_depart (
           v_num_dept depart.dept_no%TYPE,
           v_dnombre depart.dnombre%TYPE DEFAULT 'PROVISIONAL',
           v_loc depart.loc%TYPE DEFAULT ‘PROVISIONAL’)
   AS
   $$ BEGIN
           INSERT INTO depart VALUES (v_num_dept, v_dnombre, v_loc);
   END; $$
   ```

   Indicar cuáles de las siguientes llamadas son correctas y cuáles no:

   1º. crear_depart;
   2º. crear_depart(50);
   3º. crear_depart('COMPRAS');
   4º. crear_depart(50,'COMPRAS');
   5º. crear_depart('COMPRAS', 50);
   6º. crear_depart('COMPRAS', 'VALENCIA');
   7º. crear_depart(50, 'COMPRAS', 'VALENCIA');
   8º. crear_depart('COMPRAS', 50, 'VALENCIA');
   9º. crear_depart('VALENCIA', ‘COMPRAS’);
   10º. crear_depart('VALENCIA', 50);

5. Codificar un procedimiento que reciba una lista de hasta 5 números y visualice su suma.

6. Realizar los siguientes procedimientos y funciones sobre la base de datos de jardinería:

   1. Función: calcular_precio_total_pedido

      Descripción: Dado un código de pedido la función debe calcular la suma total del pedido. Tenga en cuenta que un pedido puede contener varios productos diferentes y varias cantidades de cada producto.
      Parámetros de entrada: codigo_pedido (INT)
      Salida: el precio total del pedido (FLOAT)

   2. Función: calcular_suma_pedidos_cliente

      Descripción: Dado un código de cliente la función debe calcular la suma total de todos los pedidos realizados por el cliente. Deberá hacer uso de la función calcular_precio_total_pedido que ha desarrollado en el apartado anterior.
      Parámetros de entrada: codigo_cliente (INT)

Salida: La suma total de todos los pedidos del cliente (FLOAT)

3. Función: calcular_suma_pagos_cliente

   Descripción: Dado un código de cliente la función debe calcular la suma total de los pagos realizados por ese cliente.
   Parámetros de entrada: codigo_cliente (INT)
   Salida: la suma total de todos los pagos del cliente (FLOAT)

4. Procedimiento: calcular_pagos_pendientes

   Descripción: Deberá calcular los pagos pendientes de todos los clientes. Para saber si un cliente tiene algún pago pendiente deberemos calcular cuál es la cantidad de todos los pedidos y los pagos que ha realizado. Si la cantidad de los pedidos es mayor que la de los pagos entonces ese cliente tiene pagos pendientes.
   Deberá insertar en una tabla llamada clientes_con_pagos_pendientes los siguientes datos: id_cliente, suma_total_pedidos, suma_total_pagos, pendiente_de_pago (si la tabla no existe se debe crear)

7. Escribir un procedimiento que modifique la localidad de una oficina de la base de datos de jardinería. El procedimiento recibirá como parámetros el número y la localidad nueva.

8. En la base de datos de departamentos, empleados y proyectos, codificar un procedimiento que reciba como parámetros un número de departamento, un importe y un porcentaje; y suba el salario a todos los empleados del departamento indicado en la llamada. La subida será el porcentaje o el importe indicado en la llamada (el que sea más beneficioso para el empleado en cada caso empleado).

9. En la misma base de datos del ejercicio anterior, escribir un procedimiento que suba el sueldo de todos los empleados que ganen menos que el salario medio de su oficio. La subida será del 50% de la diferencia entre el salario del empleado y la media de su oficio. Se deberá asegurar que la transacción no se quede a medias, y se gestionarán los posibles errores.

10. Escribir un disparador en la base de datos de los ejercicios anteriores que haga fallar cualquier operación de modificación del apellido o del número de un empleado, o que suponga una subida de sueldo superior al 10%.

11. Cambiar la solución del ejercicio anterior para permitir la eliminación físicamente del registro de la tabla empleados pero guardar una copia del registro eliminado en una tabla llamada ex_empleados, guardando también en esa tabla la fecha de la baja.

12. En la base de datos de jardinería, queremos que no se puedan eliminar físicamente los pedidos. Por tanto, en vez de eliminarlos, se marcarán como baja. Para ello debemos añadir a la tabla de pedidos un campo baja que contendrá un valor lógico TRUE o FALSE (no podrá contener ningún otro valor). Por defecto estará puesto a FALSE (no se ha borrado) y cuando se intente borrar el pedido, en vez de borrar el pedido se cambiará el valor de este campo.

13. Queremos que se guarde en una tabla altas_empleados el historial de inserciones de registros realizadas en la tabla empleados, además de los datos del empleado se deberá guardar en la tabla el usuario que realizó la inserción del empleado y la fecha/hora de la operación. La

primera vez que se ejecute el disparador deberá crear la tabla si no existe y rellenarla con los empleados que contenga la base de datos en ese momento.

14. Hacer que se actualicen automáticamente las existencias de los productos cuando se inserte un nuevo pedido o cuando se rectifique la cantidad de uno existente. Se supone que un pedido produce una reducción del *stock* (existencias) del producto.