1. **(4,5 puntos)** Crea un objeto que dada una temporada (ALTA, MEDIA o BAJA) presente un listado de las categorías de habitación que han sido reservadas, mostrando el número de habitaciones de la categoría ocupadas. Además, para cada categoría, deberá aparecer el listado de reservas de habitaciones ordenadas por número de habitación de menor a mayor:

    Listado de categorías reservadas para la TEMPORADA XXXX:

    Categoría C1, Reservas de habitaciones: H1:

        Habitación nº N1, reservada Fecha DD/MM/YYYY, con régimen R1

        Habitación nº N2, reservada Fecha DD/MM/YYYY, con régimen R2

    Donde XXXX es el nombre de la temporada, C1 es es el nombre de la categoría de la habitación, H1 el número de reservas de habitaciones para dicha categoría. Nº es el número de la habitación, DD/MM/YYYY la fecha de la reserva de dicha habitación en formato día/mes/año, y R1 el régimen alimenticio de la reserva asociada.

    Se deberán controlar las siguientes situaciones de excepción. Si la temporada pasada por parámetro no existe deberá mostrarse el mensaje "Error, la temporada indicada no es válida, inserte el valor ALTA, MEDIA o BAJA". Si la temporada existe pero no hay reservas para dicha temporada deberá indicarse el mensaje "La temporada indicada no tiene reservas".

    SOL:

```
CREATE OR REPLACE PROCEDURE lista_temporada(IN_TEMP temporada.nombre%type) IS
aux_temp temporada.nombre%type;
sin_reservas exception;

CURSOR c1 IS SELECT CATEGORIA.nombre, count(habitacion.numero) as habitaciones
FROM CALENDRESERVAS
JOIN CALENDARIO ON (calendreservas.fecha = calendario.fecha)
JOIN HABITACION ON (calendreservas.habitacion = habitacion.numero)
JOIN CATEGORIA ON (habitacion.categoria = categoria.nombre)
WHERE CALENDARIO.temporada = IN_TEMP
GROUP BY CATEGORIA.nombre;

aux_categoria categoria.nombre%type;

CURSOR C2 IS SELECT Habitacion.numero, calendreservas.fecha, calendreservas.alimentacion FROM CALENDRESERVAS
JOIN HABITACION ON (habitacion.numero = calendreservas.habitacion)
JOIN CALENDARIO ON (calendreservas.fecha = calendario.fecha)
WHERE habitacion.categoria = aux_categoria AND calendario.temporada = IN_TEMP
```

```
ORDER BY habitacion.numero ASC;

BEGIN
SELECT nombre into aux_temp from TEMPORADA WHERE nombre = IN_TEMP;

escribir('Listado de reservas para la temporada ' || IN_TEMP);
FOR regc1 in C1 LOOP
   escribir ('Categoria ' || regc1.nombre || ', Reservas de habitaciones:' ||
regc1.habitaciones);
   aux_categoria := regc1.nombre;
   FOR regc2 in C2 LOOP
         escribir('Habitacion nº ' || regc2.numero || ', reservada en ' || regc2.fecha || '
con régimen ' ||  regc2.alimentacion);
   END LOOP;
        escribir('');
END LOOP;

EXCEPTION
WHEN no_data_found THEN
   escribir('Error, la temporada no es válida, inserte el valor ALTA, MEDIA o BAJA');
WHEN sin_reservas THEN
   escribir('La temporada indicada no tiene reservas');
END;

– exec lista_temporada('BAJA');
```

2. **(3 puntos)** Crea un objeto que dado un nif cliente y una fecha devuelva el conteo de habitaciones que ha reservado para dicha fecha. Se deberán de controlar dos situaciones de excepción. Si no existe el cliente en la base de datos se mostrará el mensaje "Error, el cliente con nif NIF no existe", donde NIF es el nif del cliente pasado por parámetro. Por otro lado, si el cliente existe pero no tiene reservas para la fecha indicada se mostrará el mensaje "Error, el cliente no tiene reservas en la fecha DD/MM/YYYY".

   SOL:

```
CREATE OR REPLACE FUNCTION cuenta_habitaciones(IN_NIF cliente.nif%type,
IN_FECHA date) RETURN number
IS
aux_cliente cliente.nombre%type;
aux_reservas number(3);
aux_habitaciones number(3);
no_reservas exception;
BEGIN

SELECT nombre into aux_cliente FROM cliente WHERE cliente.nif = IN_NIF;

SELECT count(codigo) into aux_reservas FROM reserva
JOIN calendreservas ON (calendreservas.reserva = reserva.codigo)
```

```
WHERE cliente = IN_NIF AND fecha = IN_FECHA;

IF aux_reservas = 0 THEN
        raise no_reservas;
END IF;

SELECT count(habitacion) into aux_habitaciones FROM reserva
JOIN calendreservas ON (calendreservas.reserva = reserva.codigo)
WHERE cliente = IN_NIF AND fecha = IN_FECHA;

return aux_habitaciones;

EXCEPTION
WHEN no_data_found THEN
        escribir('Error, el cliente con nif ' || IN_NIF || ' no existe');
        return null;
WHEN no_reservas THEN
        escribir('Error, el cliente no tiene reservas para la fecha ' || to_char(IN_FECHA,
'dd/mm/yyyy'));
        return null;
END;


– select cuenta_habitaciones('21446688A','15/11/2012') FROM dual;
– select cuenta_habitaciones('21446688A','10/12/2013') FROM dual;
– select cuenta_habitaciones('22222222A','10/10/2012') FROM dual;
```

3. **(2,5 puntos)** Crea un objeto que controle las citas de los clientes para los tratamientos. Cuando se dé de alta una nueva cita, se modifique o se elimine, se debe de mostrar por pantalla cuánto es el dinero que deberá abonar o deberá devolverse al cliente mediante los mensajes "Cita tratamiento registrada, el cliente debe abonar X€" o "Cita de tratamiento registrada, se debe devolver al cliente Y€", donde X e Y es el precio correspondiente al tratamiento de la cita añadida, eliminada o modificada.

```
CREATE OR REPLACE TRIGGER trigger_citas
BEFORE INSERT OR UPDATE OR DELETE ON cita
FOR EACH ROW
DECLARE
aux_precio_nuevo tratamiento.precio%type :=0;
aux_precio_antiguo tratamiento.precio%type := 0;
BEGIN
        IF INSERTING OR UPDATING THEN
          select tratamiento.precio into aux_precio_nuevo FROM tratamiento
        WHERE tratamiento.codigo = :new.tratamiento;
        END IF;
        IF DELETING OR UPDATING THEN
```

```
        select tratamiento.precio into aux_precio_antiguo FROM tratamiento
        WHERE tratamiento.codigo = :old.tratamiento;
        END IF;

        IF INSERTING THEN
                escribir('Cita de tratamiento registrada, el cliente debe abonar ' ||
        aux_precio_nuevo || '€');
        ELSIF UPDATING THEN
                IF(aux_precio_nuevo - aux_precio_antiguo > 0) THEN
                escribir('Cita de tratamiento registrada, el cliente debe abonar ' ||
        (aux_precio_nuevo - aux_precio_antiguo) || '€');
                ELSE
                escribir('Cita de tratamiento registrada, se debe devolver al cliente ' ||
        (aux_precio_antiguo - aux_precio_nuevo) || '€');
                END IF;
        ELSIF DELETING THEN
                escribir('Cita de tratamiento registrada, se debe devolver al cliente ' ||
        (aux_precio_antiguo - aux_precio_nuevo) || '€');
        END IF;
END;

– UPDATE CITA SET TRATAMIENTO = 'DEP01' WHERE FECHA =
to_date('15/11/12','dd/mm/yy') AND momento = 1;
– UPDATE CITA SET TRATAMIENTO = 'MAS01' WHERE FECHA =
to_date('15/11/12','dd/mm/yy') AND momento = 1;
– INSERT INTO CITA VALUES('16/11/12',1,'77777777S','21446688A','MAS01');
– DELETE FROM CITA WHERE FECHA = to_date('16/11/12') AND empleado =
'77777777S';
```