```
/*
REALIZAR UN PROCEDIMIENTO, QUE RECORRA TODOS LOS EQUIPOS Y QUE
ACTUALICE LOS REGISTROS DE EQUIPOS_COPIA EN EL CASO DE QUE NO ESTE,
QUE LO INSERTE  Y EN EL CASO DE QUE ESTE QUE LO ACTUALICE
*/
CREATE OR REPLACE PROCEDURE spMantenerEquipoCopia
IS

   CURSOR C_EQ IS
      SELECT *
      FROM EQUIPOS;

   V_REG_EQUIPOS EQUIPOS%ROWTYPE;
   V_EXISTE INTEGER;

BEGIN

   OPEN C_EQ;
   LOOP
     FETCH C_EQ INTO  V_REG_EQUIPOS;
     EXIT WHEN C_EQ%NOTFOUND;

     SELECT NVL(COUNT(*),0) INTO V_EXISTE
     FROM EQUIPOS_COPIA
     WHERE NOMBRE = V_REG_EQUIPOS.NOMBRE;

     IF V_EXISTE = 0 THEN
        --INSERTO
        INSERT INTO equipos_copia (nombre,ciudad,division, conferencia)
        values (v_reg_EQUIPOS.nombre,v_reg_EQUIPOS.ciudad,
              v_reg_EQUIPOS.division, v_reg_EQUIPOS.conferencia);
      ELSE
        --ACTUALIZO
         UPDATE EQUIPOS_COPIA
           SET ciudad =v_reg_EQUIPOS.CIUDAD,
              division =v_reg_EQUIPOS.DIVISION,
              conferencia = v_reg_EQUIPOS.CONFERENCIA
          WHERE NOMBRE = v_reg_EQUIPOS.NOMBRE;

      END IF;

    END LOOP;
    CLOSE C_EQ;
  EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Ocurrió el error ' ||SQLCODE ||' ---mensaje: ' ||
SQLERRM);
END;
```

```
SELECT * FROM EQUIPOS_COPIA;

EXECUTE spMantenerEquipoCopia;

UPDATE EQUIPOS
SET CIUDAD = 'SEVILLA'
WHERE NOMBRE ='Celtics';



/*Haz un procedimiento que, dada una temporada y un equipo (ambos pasados como
parámetros de entrada) MUESTRE:
Cuántos partidos en total ha ganado ese equipo como local
Y Para cada partido: Decir cuántos puntos anotó.*/
CREATE OR REPLACE PROCEDURE spEquipoGanador(pTemp partidos.temporada%type,
pEquipo equipos.nombre%type)
is
    v_partidos_ganados number;

    cursor c_part is
        select pa.codigo, pa.puntos_local
        from partidos pa
        where upper(pa.equipo_local) = upper(pEquipo)
        and pa.temporada = pTemp;

    v_cod partidos.codigo%type;
    v_puntos_local partidos.puntos_local%type;

begin

    open c_part;
    loop
        fetch c_part into v_cod, v_puntos_local;
        exit when c_part%notfound;

        DBMS_OUTPUT.PUT_LINE('El equipo '|| pEquipo || ' en el partido '||v_cod||' metio '||
v_puntos_local);

    end loop;
    close c_part;

    --- y cuantos partidos ha ganado
    select count(*) into v_partidos_ganados
    from partidos pa
    where pa.puntos_visitante < pa.puntos_local
```

```
   and upper(pa.equipo_local) = upper(pEquipo)
   and pa.temporada = pTemp;

   DBMS_OUTPUT.PUT_LINE('El equipo '|| pEquipo || ' ganó '||v_partidos_ganados||'
partidos ');

EXCEPTION
   WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('Ocurrió el error ' ||SQLCODE ||' ---mensaje: ' || SQLERRM);
end;


/*
<<<CURSOS IMPLICITO>>
Realizar un método o procedimiento que muestre el total en
euros de un pedido, pasale el codigo por parametro.
*/

CREATE OR REPLACE PROCEDURE sp_total_euros_pedido (
   p_cod_dp              IN    detallepedidos.codigopedido%TYPE
) IS
   v_total_euros_pedido detallepedidos.preciounidad%TYPE;
BEGIN
   SELECT
      SUM(dp.preciounidad * dp.cantidad)
   INTO v_total_euros_pedido
   FROM
      detallepedidos   dp,
      pedidos          p
   WHERE
      p.codigopedido = dp.codigopedido
      AND dp.codigopedido = p_cod_dp;

   dbms_output.put_line('EL TOTAL DE EUROS DE ' || p_cod_dp || ' ES '
|| v_total_euros_pedido);

EXCEPTION
   WHEN no_data_found THEN
      dbms_output.put_line('EL CODIGO NO EXISTE');
   WHEN OTHERS THEN
      dbms_output.put_line('Ocurrió el error '
                           || sqlcode
                           || ' ---mensaje: '
                           || sqlerrm);
END;

<<<COMO SE EJECUTA>>>>
DECLARE
```

```
p_cod_dp  detallepedidos.codigopedido%TYPE:= 1;

BEGIN
SP_TOTAL_EUROS_PEDIDO(p_cod_dp);
END;


--- PROCEDMIENTO  (BD ALUMNOS)
CREAR UN PROCEDIMIENTO BORRE UN ALUMNO
CREATE OR REPLACE PROCEDURE spBorradoAlumno(p_codigo alumnos.codigo%type)
IS

BEGIN
   delete from alumnos
   where codigo = p_codigo;
END;

Begin
   spBorradoAlumno(8);
end;

CREAR UN PROCEDIMIENTO AGREGAR UN PRODUCTO (RECUERDA QUE DEBES
PONER UN REGISTRO EN LA TABLA PADRE GAMASPRODUCTOS PARA QUE
PUEDAS INSERTAR EL NUEVO PRODUCTO EN LA TABLA PRODUCTOS PORQUE
SINO TE VA A APARECER CLAVE VIOLADA)
CREATE OR REPLACE PROCEDURE spinsertarproducto (
   p_codigoproducto    productos.codigoproducto%TYPE,
   p_nombre            productos.nombre%TYPE,
   p_gama              productos.gama%TYPE,
   p_cantidadenstock   productos.cantidadenstock%TYPE,
   p_precioventa       productos.precioventa%TYPE
) IS
BEGIN
   INSERT INTO productos (
      codigoproducto,
      nombre,
      gama,
      cantidadenstock,
      precioventa
   ) VALUES (
      p_codigoproducto,
      p_nombre,
      p_gama,
      p_cantidadenstock,
      p_precioventa
   );

END;
```

```
begin
spinsertarproducto('100', 'muñeca inflable', 'productos sexuales',
20, 15);
end;


select *
from productos
where codigoproducto= '100';
```

INSERTAR ALUMNOS

```
CREATE OR REPLACE PROCEDURE spInsertarAlumno
    (p_codigo alumnos.codigo%type,
     p_nombre alumnos.nombre%type,
     p_ape alumnos.apellidos%type,
     p_fecnac alumnos.fecnac%type,
     p_curso alumnos.curso%type,
     p_sexo alumnos.sexo%type)
IS
  v_edad alumnos.edad%type;
BEGIN

    v_edad := trunc(months_between(sysdate,p_fecnac)/12);


    INSERT INTO alumnos (
        codigo,
        nombre,
        apellidos,
        fecnac,
        curso,
        sexo,
        edad
    ) VALUES (
      p_codigo,
      p_nombre,
      p_ape,
      p_fecnac,
      p_curso,
      p_sexo,
      v_edad
    );
```

```
END;

begin

spInsertarAlumno(169,'IRENE', 'SANCHEZ
RUANO',TO_DATE('27041992','DDMMYYYY'),'1DAW','M');


end;

-- INSERTAR ALUMNO
create or replace  PROCEDURE spInsertarAlumnoDevCodigo
    (p_codigo IN out alumnos.codigo%type,
     p_nombre alumnos.nombre%type,
     p_ape alumnos.apellidos%type,
     p_fecnac alumnos.fecnac%type,
     p_curso alumnos.curso%type,
     p_sexo alumnos.sexo%type)
IS
  v_edad alumnos.edad%type;
BEGIN

    v_edad := trunc(months_between(sysdate,p_fecnac)/12);

    select max(codigo) +1 into p_codigo from alumnos;


    INSERT INTO alumnos (
        codigo,
        nombre,
        apellidos,
        fecnac,
        curso,
        sexo,
        edad
    ) VALUES (
      p_codigo,
      p_nombre,
      p_ape,
      p_fecnac,
      p_curso,
      p_sexo,
      v_edad
    );

END;
```

```
declare

    v_codalu alumnos.codigo%type;

begin

spInsertarAlumnoDevCodigo(v_codalu,'ELVIS2','COSTELLO',TO_DATE('28/12/1996','DD/M
M/YYYY'),'1DAW','H');

DBMS_OUTPUT. PUT_LINE ('EL ALUMNO ELVIS HA SIDO REGISTRADO CON EL
CODIGO '||V_CODALU);


end;

-- CREAR UN PROCEDIMIENTO QUE PERMITA INSERTAR UN DEPARTAMENTO Y QUE
DESPUES DE INSERTAR TE DEVUELVA EL CODIGO DE ESTE. DICHO CODIGO DE
DEPARTAMENTO TIENE QUE SER EL SIGUIENTE MULTIPLO DE 10 SUPERIOR AL
ULTIMO DADO


create or replace  PROCEDURE spInsertarDeptDevCodigo
    (p_codigo out dept.deptno%type,
     p_nombre dept.dname%type,
     p_loc dept.loc%type)
IS
  v_codigo_mas_10 dept.deptno%type;
BEGIN

-- calcular el codigo con el que voy a insertar
select max(deptno)+10
    into v_codigo_mas_10
from dept;

--p_codigo := v_codigo_mas_10 - mod(v_codigo_mas_10,10);
p_codigo := trunc(v_codigo_mas_10/10,0) * 10;

--insertar
INSERT INTO dept (
    deptno,
    dname,
    loc
) VALUES (
    p_codigo,
    p_nombre,
    p_loc
);
```

```
--fin
end;

declare

   v_dept dept.deptno%type;

begin

spInsertarDeptDevCodigo(v_dept,'DAW2','SEVILLA2');

DBMS_OUTPUT. PUT_LINE ('EL DEPT EN SEVILLA2 HA SIDO REGISTRADO CON EL
CODIGO '||v_dept);


end;

-- Crear un procedimiento que despida (borre) al empleado más novel de cada
departamento.
create or replace PROCEDURE spBorrarNovel
IS

 CURSOR C_DEP IS
   SELECT DEPTNO,MAX(HIREDATE)
   FROM EMP
   GROUP BY DEPTNO;

   V_DEPTNO EMP.DEPTNO%TYPE;
   V_MAX_FEC DATE;


BEGIN

   OPEN C_DEP;
   LOOP
      FETCH C_DEP INTO V_DEPTNO,V_MAX_FEC;
      EXIT WHEN C_DEP%NOTFOUND;

      DELETE FROM EMP
      WHERE DEPTNO = V_DEPTNO
      AND HIREDATE = V_MAX_FEC;

   END LOOP;
   CLOSE C_DEP;

EXCEPTION
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('Ocurrió el error ' ||SQLCODE ||' ---mensaje: ' || SQLERRM);
```

```
END;

SELECT *
FROM EMP
ORDER BY HIREDATE DESC;

BEGIN
spBorrarNovel;
END;

-- Crear un procedimiento que muestre el nombre de todos los departamentos y el
-- número de empleados que tiene (incluso si no tiene empleados).
create or replace PROCEDURE spBorrarNovel21(P_coddept emp.deptno%type)
IS

 CURSOR C_DEP IS
   SELECT DEPTNO,MAX(HIREDATE)
   FROM EMP
   GROUP BY DEPTNO;

   V_DEPTNO EMP.DEPTNO%TYPE;
   V_MAX_FEC DATE;


BEGIN

   OPEN C_DEP;
   LOOP
      FETCH C_DEP INTO V_DEPTNO,V_MAX_FEC;
      EXIT WHEN C_DEP%NOTFOUND;

      if v_deptno = p_coddept or p_coddept is null then
         DELETE FROM EMP
         WHERE DEPTNO = V_DEPTNO
         AND HIREDATE = V_MAX_FEC;
      end if;

   END LOOP;
   CLOSE C_DEP;

EXCEPTION
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('Ocurrió el error ' ||SQLCODE ||' ---mensaje: ' || SQLERRM);


END;
```

```
create or replace PROCEDURE spBorrarNovel22(P_coddept emp.deptno%type)
IS

 CURSOR C_DEP IS
   SELECT DEPTNO,MAX(HIREDATE)
   FROM EMP
   where deptno = nvl(p_coddept,deptno)
   GROUP BY DEPTNO;

   V_DEPTNO EMP.DEPTNO%TYPE;
   V_MAX_FEC DATE;
BEGIN

   OPEN C_DEP;
   LOOP
      FETCH C_DEP INTO V_DEPTNO,V_MAX_FEC;
      EXIT WHEN C_DEP%NOTFOUND;

      DELETE FROM EMP
      WHERE DEPTNO = V_DEPTNO
      AND HIREDATE = V_MAX_FEC;

   END LOOP;
   CLOSE C_DEP;

EXCEPTION
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('Ocurrió el error ' ||SQLCODE ||' ---mensaje: ' || SQLERRM);


END;

-- Crear un procedimiento que muestre el nombre de todos los departamentos y el
-- número de empleados que tiene (incluso si no tiene empleados).

CREATE OR REPLACE PROCEDURE SP_DEPT_NUM
IS
   CURSOR C_DEPT_NUM IS
      SELECT D.DNAME, NVL(COUNT(E.EMPNO),0)
      FROM DEPT D, EMP E
      WHERE D.DEPTNO = E.DEPTNO(+)
      GROUP BY D.DNAME;

   V_NOM DEPT.DNAME%TYPE;
   V_EMPS NUMBER;

BEGIN
```

```
   OPEN C_DEPT_NUM;
   LOOP
      FETCH C_DEPT_NUM INTO V_NOM, V_EMPS;
      EXIT WHEN C_DEPT_NUM%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('EL DEPT '||V_NOM||' TIENE '||V_EMPS||  '
EMPLEADOS');
   END LOOP;
   CLOSE C_DEPT_NUM;

EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Ocurrió el error ' ||SQLCODE ||' ---mensaje: ' ||
SQLERRM);
END;
```

Escribir un programa que visualice el apellido y el salario de los cinco empleados que tienen el salario más alto.

```
DECLARE

   CURSOR C_EMP IS
      SELECT ENAME, SAL FROM EMP
      ORDER BY SAL DESC;

   V_NOMBRE EMP.ENAME%TYPE;
   V_SALARIO EMP.SAL%TYPE;
   contador integer :=0;


BEGIN

   OPEN C_EMP;
   LOOP
      FETCH C_EMP INTO V_NOMBRE, V_SALARIO;
      EXIT WHEN C_EMP%NOTFOUND;

      contador := contador+1;

      if contador = 5 then
         exit;
       end if;
      DBMS_OUTPUT.PUT_LINE('EMPLEADO -> '|| V_NOMBRE || ' COBRA: ' ||
V_SALARIO);

   END LOOP;
   CLOSE C_EMP;
```

```
exception
when others then
   DBMS_OUTPUT.PUT_LINE('Ocurrio el error' || SQLCODE || '--- mensaje:');
end;


-- Crea una procedimiento que reciba como parámetro el número de un departamento
-- y que devuelva el salario máximo y el salario mínimo de los empleados que trabajan en él.
-- controlar que el departamento puede que no existe
CREATE OR REPLACE PROCEDURE spDevMinMaxSalDept(pCodDept IN
dept.deptno%type, pSalMax OUT EMP.SAL%TYPE, pSalMin OUT EMP.SAL%TYPE)
IS
   v_nombre DEPT.Dname%type;
BEGIN

   SELECT DNAME INTO V_NOMBRE
   FROM DEPT
   WHERE DEPTNO = PCODDEPT;

   select nvl(max(sal),0), nvl(min(sal),0) into psalmax, psalmin
   from emp
   where deptno = pCodDept;



EXCEPTION
WHEN NO_DATA_FOUND THEN
   DBMS_OUTPUT.PUT_LINE('EL DEPARTAMENTO NO EXISTE');
   pSalMAX :=-1;
   pSalMin :=-1;
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('Ocurrió el error ' ||SQLCODE ||' ---mensaje: ' || SQLERRM);
   pSalMAX :=-1;
   pSalMin :=-1;

END;
--- COMO LO PROBAMOS

DECLARE

   V_SALMAX EMP.SAL%TYPE;
   V_SALMIN EMP.SAL%TYPE;
   V_CODIGO dept.deptno%type:=44;

BEGIN
   spDevMinMaxSalDept(V_CODIGO, V_SALMAX, V_SALMIN);
```

```
   DBMS_OUTPUT.PUT_LINE('EL CODIGO '|| V_CODIGO||' TIENE SALARIO MAXIMO '||
V_SALMAX|| ' Y MINIMO '||V_SALMIN);
END;

   select max(sal), min(sal)
   from emp
   where deptno = 48;

/*5. Realizar un procedimiento que incremente el salario el 10% a los empleados que
tengan una comisión superior al 5% del salario, y visualice el nombre, comisión y
salario antiguo, y el nombre, comisión y salario nuevo de todos los empleados.*/

CREATE OR REPLACE PROCEDURE spIncrementaSal
is
  CURSOR C_EMP IS
   SELECT ENAME, COMM, SAL
   FROM EMP
   WHERE NVL(COMM,0)  >= SAL*(5/100);

   V_NOM EMP.ENAME%TYPE;
   V_COM EMP.COMM%TYPE;
   V_SAL EMP.SAL%TYPE;

   --ESTE ES EL SALARIO NUEVO
   V_SAL_NUEVO EMP.SAL%TYPE;

begin
   OPEN C_EMP;

   LOOP
      FETCH C_EMP INTO V_NOM, V_COM, V_SAL;
      EXIT WHEN C_EMP%NOTFOUND;

      V_SAL_NUEVO := V_SAL *(1.10);
      DBMS_OUTPUT.PUT_LINE('NOMBRE ' || V_NOM  ||' , SAL = ' || V_SAL||' , SAL
NUEVO = '||V_SAL_NUEVO);

   END LOOP;
   CLOSE C_EMP;
EXCEPTION

WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('Ocurrió el error ' ||SQLCODE ||' ---mensaje: ' || SQLERRM);


END;
```

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.

```
SELECT ROUND(AVG(SAL),0), JOB
FROM EMP
GROUP BY JOB;

UPDATE emp
SET
   SAL = SAL + (V_SAL-SAL)*0.5
WHERE
   sal < V_SAL
   AND job = V_JOB;


CREATE OR REPLACE PROCEDURE spMedia
IS

   CURSOR C_MEDIA IS
      SELECT ROUND(AVG(SAL),0), JOB
      FROM EMP
      GROUP BY JOB;
   V_SAL NUMBER;
   V_JOB EMP.JOB%TYPE;

BEGIN

   OPEN C_MEDIA;
      LOOP
         FETCH C_MEDIA INTO V_SAL,V_JOB;
         EXIT WHEN C_MEDIA%NOTFOUND;

         UPDATE emp
         SET
            SAL = SAL + (V_SAL-SAL)*0.5
         WHERE
            sal < V_SAL
            AND job = V_JOB;

      END LOOP;
   CLOSE C_MEDIA;

/*
```

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.
*/

```
CREATE OR REPLACE PROCEDURE spMedia
IS
   CURSOR C_EMP IS
      SELECT EMPNO, SAL,JOB
      FROM EMP;

   V_EMPNO EMP.EMPNO%TYPE;
   V_SAL EMP.SAL%TYPE;
   V_JOB EMP.JOB%TYPE;

   V_MEDIA_JOB NUMBER;

BEGIN

   OPEN C_EMP;
   LOOP
      FETCH C_EMP INTO V_EMPNO, V_SAL, V_JOB;
      EXIT WHEN C_EMP%NOTFOUND;

      SELECT AVG(SAL) INTO V_MEDIA_JOB
      FROM EMP
      WHERE UPPER(JOB) = UPPER(V_JOB);


      IF V_SAL <= V_MEDIA_JOB THEN

         UPDATE EMP
            SET SAL  = SAL *1.5
         WHERE EMPNO = V_EMPNO;

      END IF;
   END LOOP;
   CLOSE C_EMP;
END;
```