EJERCICIO JARDINERIA

```
-- Mostrar un listado de los empleados y cuanto Ha vendido a sus clientes
DECLARE

   CURSOR C_CLI_CIU IS
      SELECT E.NOMBRE, SUM(DP.CANTIDAD*DP.PRECIOUNIDAD)
      FROM EMPLEADOS E, CLIENTES C, PEDIDOS P, DETALLEPEDIDOS DP
      WHERE E.CODIGOEMPLEADO = C.CODIGOEMPLEADOREPVENTAS
      AND C.CODIGOCLIENTE = P.CODIGOCLIENTE
      AND P.CODIGOPEDIDO = DP.CODIGOPEDIDO
      GROUP BY E.NOMBRE;

   V_NOM CLIENTES.NOMBRECLIENTE%TYPE;
   V_GASTO NUMBER;


BEGIN

   OPEN C_CLI_CIU;
   LOOP
      FETCH C_CLI_CIU INTO V_NOM, V_GASTO;
      EXIT WHEN C_CLI_CIU%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('EL EMPLEADO '||V_NOM||' HA VENDIDO '||V_GASTO||'
€');

   END LOOP;

   CLOSE C_CLI_CIU;

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

END;



SELECT E.NOMBRE, SUM(DP.CANTIDAD*DP.PRECIOUNIDAD)
FROM EMPLEADOS E, CLIENTES C, PEDIDOS P, DETALLEPEDIDOS DP
WHERE E.CODIGOEMPLEADO = C.CODIGOEMPLEADOREPVENTAS
AND C.CODIGOCLIENTE = P.CODIGOCLIENTE
AND P.CODIGOPEDIDO = DP.CODIGOPEDIDO
GROUP BY E.NOMBRE;

--- MOSTAR LOS EMPLEADOS QUE HAYAN VENDIDO CERO TAMBIEN DEBEN
--SER MOSTRADOS.
DECLARE
```

```
   CURSOR C_EMP IS
      SELECT CODIGOEMPLEADO, NOMBRE
      FROM EMPLEADOS E;

   V_NOM EMPLEADOS.NOMBRE%TYPE;
   V_COD EMPLEADOS.CODIGOEMPLEADO%TYPE;
   V_GASTO NUMBER;


BEGIN

   OPEN C_EMP;
   LOOP
      FETCH C_EMP INTO V_COD, V_NOM;
      EXIT WHEN C_EMP%NOTFOUND;

      --- AQUI PUEDE HABER MUCHO MUCHO CODIGO
      V_GASTO := 0;

      BEGIN

         SELECT NVL(SUM(DP.CANTIDAD*DP.PRECIOUNIDAD),0) INTO V_GASTO
         FROM EMPLEADOS E, CLIENTES C, PEDIDOS P, DETALLEPEDIDOS DP
         WHERE C.CODIGOEMPLEADOREPVENTAS = V_COD
         AND C.CODIGOCLIENTE = P.CODIGOCLIENTE
         AND P.CODIGOPEDIDO = DP.CODIGOPEDIDO;
      EXCEPTION
      WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Ocurrió el error ' ||SQLCODE ||' ---mensaje: ' ||
SQLERRM);
      END;

      DBMS_OUTPUT.PUT_LINE('EL EMPLEADO '||V_NOM||' HA VENDIDO '||V_GASTO||'
€');

   END LOOP;

   CLOSE C_EMP;

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

END;
```

BD ALUMNOS
/MOSTAR POR PANTALLA, CADA UNO DE LOS DEPARTAMENTOS CON EL NOMBRE Y CUANTO GANA EL EMPLEADO QUE MAS COBRA CON SUS DATOS DE COBRO/
--EJEMPLO: 10 ACCOUNTIG NEW YORK KING 5000 0 PRESIDENT

<<<<<<<<<<<<<clase 9>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
DECLARE

```
    CURSOR C_DEPT IS
         SELECT DEPTNO, DNAME, LOC
         FROM DEPT
         WHERE DEPTNO IN (10,20,30);

    V_CODEPT DEPT.DEPTNO%TYPE;
    V_DNAME DEPT.DNAME%TYPE;
    V_LOC DEPT.LOC%TYPE;

    V_ENAME EMP.ENAME%TYPE;
    V_SAL EMP.SAL%TYPE;
    V_COMM EMP.COMM%TYPE;
    V_TRABAJO EMP.JOB%TYPE;


BEGIN

OPEN C_DEPT;

LOOP
   FETCH C_DEPT INTO V_CODEPT,V_DNAME,V_LOC;
   EXIT WHEN C_DEPT%NOTFOUND;

   -- PARA CADA CODIGO DE DEPARTAMENTO CALCULO EL NOTA QUE COBRA MAS CON SUS DATOS.
   SELECT ENAME, SAL, COMM,JOB INTO V_ENAME, V_SAL, V_COMM, V_TRABAJO
   FROM EMP
   WHERE SAL = (SELECT MAX(SAL) FROM EMP WHERE DEPTNO = V_CODEPT)
   AND DEPTNO = V_CODEPT
   AND ROWNUM =1;

   dbms_output.put_line(V_CODEPT||' '||V_DNAME||' '||V_LOC||' '||V_ENAME||' '||V_SAL||' '||V_COMM||' '||V_TRABAJO);

END LOOP;

CLOSE C_DEPT;
END;
```

CURSOR IMPLICITO

```
DECLARE

   CURSOR C_NOMBRES IS
      SELECT NOMBRE,APELLIDOS
      FROM ALUMNOS;

   V_NOMBRE ALUMNOS.NOMBRE%TYPE;
   V_APE ALUMNOS.APELLIDOS%TYPE;


BEGIN

   OPEN C_NOMBRES;

   LOOP
      FETCH C_NOMBRES INTO V_NOMBRE,V_APE;
      EXIT WHEN C_NOMBRES%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE( 'EL NOMBRE ES: ' || V_NOMBRE || ' ' ||V_APE);
   END LOOP;

   CLOSE C_NOMBRES;

END;

DECLARE
   V_IMPLICITO NUMBER;
   v_coddept dept.deptno%type :=&CODIGO_DEPT;
BEGIN

   SELECT nvl(SUM(E.SAL+NVL(E.COMM,0)),0) AS GASTOSUELDO INTO V_IMPLICITO
   FROM EMP E
   WHERE E.DEPTNO=v_coddept;

   DBMS_OUTPUT.PUT_LINE(V_IMPLICITO);
EXCEPTION

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


--- OBTENER CUANDO SE HA GASTADO CADA DEPARTAMENTO EN SUELDOS,
TENIENDO EN CUENTA LA COMISIÓN, QUE SI NO TIENE ES CERO


--- MOSTRAR UNA LINEA POR CADA DEPARTAMENTO CON SU NOMBRE Y SU GASTO
EN SUELDO
--- EJEMPLO: EL DEPARTAMENTO ACCOUNTING SE HA GASTADO EN SUELDO 7500
EUROS.
```

--- SE DEBE REALIZAR DE 2 FORMAS, UNA VEZ CON UN CURSO EXPLICITO E ITERANDO POR EMPLEADO Y OTRA VEZ CON UN CURSOR IMPLICITO.

```
DECLARE

   CURSOR C_SALARIO IS
      SELECT ENAME,E.SAL+NVL(E.COMM,0)
      FROM EMP E;

   V_SALARIO NUMBER;
   V_NOMBRE EMP.ENAME%TYPE;

   V_SALARIO_TOTAL NUMBER:=0;

BEGIN


   OPEN C_SALARIO;
   LOOP
      FETCH C_SALARIO INTO V_NOMBRE,V_SALARIO;
      EXIT WHEN C_SALARIO%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(V_NOMBRE||' COBRA '||V_SALARIO);
      V_SALARIO_TOTAL :=  V_SALARIO_TOTAL +V_SALARIO;

   END LOOP;

   CLOSE C_SALARIO;
   DBMS_OUTPUT.PUT_LINE('EL TOTAL ES '||V_SALARIO_TOTAL);

EXCEPTION

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


END;
```

MUESTRE POR PANTALLA
-- EL NOMBRE DEL DEPARTAMENTO Y CUANDO SE HA GASTADO PARA TODOS ELLOS.

```
DECLARE

 CURSOR C_SAL_DEPT IS
   SELECT DNAME,F_SALARIO_DEPT(DEPTNO)
   FROM DEPT;

  V_NOMBRE DEPT.DNAME%TYPE;
```

```
  V_SALARIO NUMBER;


BEGIN
   OPEN C_SAL_DEPT;
   LOOP
      FETCH C_SAL_DEPT INTO V_NOMBRE, V_SALARIO;
      EXIT WHEN C_SAL_DEPT%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('EL DEPARTAMENTO ' ||V_NOMBRE ||' HA GASTADO
EN SALARIO ' || V_SALARIO);

   END LOOP;
   CLOSE C_SAL_DEPT;


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


--Crea una función que reciba como parámetros el número de un departamento y una fecha,
-- y que devuelva el número de empleados que tenía ese departamento en esa fecha.
-- si no tiene que devuelva cero.
SET SERVEROUTPUT ON;
/

CREATE OR REPLACE FUNCTION fNumDept (p_numdept  DEPT.DEPTNO%TYPE,
P_fecha DATE)
RETURN INTEGER
IS
v_cantidad_empleados integer;

BEGIN
   SELECT COUNT(EMPNO) into v_cantidad_empleados
   FROM EMP
   WHERE TO_DATE(HIREDATE, 'DD/MM/YY')= p_fecha
   AND DEPTNO= P_numdept;

return v_cantidad_empleados;
EXCEPTION
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('Ocurrió el error ' ||SQLCODE ||' ---mensaje: ' || SQLERRM);
   RETURN -1;
END;
/
begin
 dbms_output.put_line(
FnUMDEPT(10, TO_DATE('17/11/81', 'DD/MM/YY')));
```

```
end;
/
SELECT COUNT(EMPNO)
FROM EMP
WHERE TO_DATE(HIREDATE, 'DD/MM/YY')= TO_DATE('17/11/81', 'DD/MM/YY')
AND DEPTNO=10;

/
-- EN JARDINERIA.
--Mostrar un listado de todos los clientes (nombre y ciudad) que no hayan hecho pagos

DECLARE

   CURSOR C_CLIENTES IS
      SELECT C.NOMBRECLIENTE, C.CIUDAD
      FROM CLIENTES C
      WHERE C.CODIGOCLIENTE NOT IN (SELECT CODIGOCLIENTE FROM PAGOS);

   V_NOMBRE CLIENTES.NOMBRECLIENTE%TYPE;
   V_CIUDAD CLIENTES.CIUDAD%TYPE;

BEGIN

   OPEN C_CLIENTES;

   LOOP
      FETCH C_CLIENTES INTO V_NOMBRE,V_CIUDAD;
      EXIT WHEN C_CLIENTES%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE( 'EL NOMBRE ES: ' || V_NOMBRE || ' ' || ' Y LA CIUDAD: '
||V_CIUDAD);
   END LOOP;

   CLOSE C_CLIENTES;
EXCEPTION

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

END;
/

-- Mostrar un listado de los empleados y cuanto ha vendido a sus clientes.

DECLARE

   CURSOR C_EMPLEADOSVENTAS IS
      SELECT E.NOMBRE AS EMPLEADO, SUM(DP.CANTIDAD *DP.PRECIOUNIDAD)
      FROM CLIENTES C, PEDIDOS P, DETALLEPEDIDOS DP, EMPLEADOS E
```

```
      WHERE E.CODIGOEMPLEADO = C.CODIGOEMPLEADOREPVENTAS
      AND C.CODIGOCLIENTE =P.CODIGOCLIENTE
      AND P.CODIGOPEDIDO = DP.CODIGOPEDIDO
      GROUP BY E.NOMBRE;

   V_NOMBRE EMPLEADOS.NOMBRE%TYPE;
   V_VENDIDO DETALLEPEDIDOS.PRECIOUNIDAD%TYPE;

BEGIN

   OPEN C_EMPLEADOSVENTAS;

   LOOP
      FETCH C_EMPLEADOSVENTAS INTO V_NOMBRE,V_VENDIDO;
      EXIT WHEN C_EMPLEADOSVENTAS%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE( 'EL EMPLEADO ES: ' || V_NOMBRE || ' ' || ' Y HA
VENDIDO: ' ||V_VENDIDO);
   END LOOP;

   CLOSE C_EMPLEADOSVENTAS;
EXCEPTION

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

END;
/

set serveroutput on;

--1. Desarrollar un procedimiento que visualice el apellido y la fecha de alta de todos los
--empleados ordenados por apellido.

CREATE OR REPLACE PROCEDURE spEmp
IS

        CURSOR C_EMP IS
        SELECT ENAME, HIREDATE
        FROM EMP;

        V_NOM EMP.ENAME%TYPE;
        V_FEC EMP.HIREDATE%TYPE;

BEGIN

        OPEN C_EMP;
        LOOP
        FETCH C_EMP INTO V_NOM, V_FEC;
```

```
        EXIT WHEN C_EMP%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE( V_NOM ||' '||V_FEC);
        END LOOP;
        CLOSE C_EMP;


EXCEPTION

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

END;

BEGIN
spEmp;
END;


--2. Desarrollar un procedimiento que encuentre el primer
--empleado con un sueldo mayor de 2.000   .

select * from emp;

CREATE OR REPLACE PROCEDURE spSueld
IS

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

BEGIN

        SELECT ENAME, SAL INTO V_NOM, V_SAL
        FROM EMP
        WHERE SAL > 2000
        AND ROWNUM <=1;

        DBMS_OUTPUT.PUT_LINE( V_NOM ||' '||V_SAL);


EXCEPTION

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

END;
```

```
BEGIN
spSueId;
END;

--3. Realizar un procedimiento que visualice el número y apellido de un empleado, así como
--la localidad de su departamento, ordenado por el nombre de la localidad

CREATE OR REPLACE PROCEDURE spLoc
IS

        CURSOR C_EMP IS
        SELECT E.ENAME, E.EMPNO, D.LOC
        FROM EMP E, DEPT D
        WHERE E.DEPTNO = E.DEPTNO;

        V_NOM EMP.ENAME%TYPE;
        V_COD EMP.EMPNO%TYPE;
        V_LOC DEPT.LOC%TYPE;

BEGIN

        OPEN C_EMP;
        LOOP
        FETCH C_EMP INTO V_NOM, V_COD, V_LOC;
        EXIT WHEN C_EMP%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE( V_NOM ||' '||V_COD||' '||V_LOC);
        END LOOP;
        CLOSE C_EMP;


EXCEPTION

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

END;

BEGIN
spLoc;
END;


--4. En la tabla EMP incrementar el salario el 10% a los empleados que tengan una
comisión
---superior al 5% del salario.
```

```
DECLARE

	CURSOR C_COM IS
	SELECT SAL, COMM, ENAME
	FROM EMP;

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

BEGIN

	OPEN C_COM;
	LOOP
	FETCH C_COM INTO V_SAL, V_COM, V_COD;
	EXIT WHEN C_COM%NOTFOUND;

	UPDATE emp
	SET
		SAL = V_SAL*1.1
	WHERE
		SAL*0.05 < V_COM
		AND ENAME = V_COD;


	END LOOP;
	CLOSE C_COM;

EXCEPTION

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

END;

SELECT SAL, COMM
FROM EMP;

ROLLBACK;


--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.


DECLARE
```

```
        CURSOR C_COM IS
        SELECT SAL, NVL(COMM,0), EMPNO, ENAME
        FROM EMP;

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

BEGIN

        OPEN C_COM;
        LOOP
        FETCH C_COM INTO V_SAL, V_COM, V_COD, V_NOM;
        EXIT WHEN C_COM%NOTFOUND;

        UPDATE emp
        SET
                SAL = V_SAL*1.1
        WHERE
                SAL*0.05 < V_COM
                AND EMPNO = V_COD;

        DBMS_OUTPUT.PUT_LINE('SALARIO ANTIGUO:'||V_NOM ||' '||V_SAL||' '||V_COM);

        END LOOP;
        CLOSE C_COM;

        OPEN C_COM;
        LOOP
        FETCH C_COM INTO V_SAL, V_COM, V_COD, V_NOM;
        EXIT WHEN C_COM%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('SALARIO NUEVO:'||V_NOM ||' '||V_SAL||' '||V_COM);
        END LOOP;
        CLOSE C_COM;


EXCEPTION

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

END;

--6. Escribir un procedimiento que reciba una cadena y visualice el apellido y el número de
--empleado de todos los empleados cuyo apellido contenga la cadena especificada. Al
```

```
--finalizar visualizar el número de empleados mostrados.

CREATE OR REPLACE PROCEDURE spCuentEmp(p_cad EMP.ENAME%TYPE)
IS

        CURSOR C_CAD IS
        SELECT INSTR(upper(ENAME), upper(p_cad), 1)
        FROM EMP;

        V_CONT NUMBER := 0;
        V_NUM NUMBER;

BEGIN
        OPEN C_CAD;
        LOOP
        FETCH C_CAD INTO V_NUM;
        EXIT WHEN C_CAD%NOTFOUND;

        IF V_NUM > 0 THEN

        V_CONT := V_CONT +1;

        END IF;

        END LOOP;
        CLOSE C_CAD;

        DBMS_OUTPUT.PUT_LINE('HAY ' ||V_CONT||' EMPLEADOS QUE CONTIENEN ' ||
p_cad);

END;

begin
spCuentEmp('in');
end;



SELECT INSTR(upper(ENAME), upper('in'), 1)
FROM emp;


--7. Crear un procedimiento que muestre el nombre de todos los departamentos y el
--número de empleados que tiene (incluso si no tiene empleados).

SELECT D.DNAME, COUNT(E.EMPNO)
FROM DEPT D, EMP E
WHERE D.DEPTNO = E.DEPTNO(+)
```

```
GROUP BY D.DNAME;


CREATE OR REPLACE PROCEDURE spDept
IS

        CURSOR C_DEPT IS
        SELECT D.DNAME, COUNT(E.EMPNO)
        FROM DEPT D, EMP E
        WHERE D.DEPTNO = E.DEPTNO(+)
        GROUP BY D.DNAME;

        V_NOM EMP.ENAME%TYPE;
        V_NUM INTEGER;

BEGIN

        OPEN C_DEPT;
        LOOP
        FETCH C_DEPT INTO V_NOM, V_NUM;
        EXIT WHEN C_DEPT%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE( V_NOM ||' '||V_NUM);
        END LOOP;
        CLOSE C_DEPT;

END;

BEGIN
spDept;
END;



--8. Buscar todos los empleados que tienen un salario + comisión superior a 2000 y
--asignarles como nuevo salario esta suma. Sólo para los que tienen comisión.

DECLARE

        CURSOR C_SUMASAL IS
        SELECT ENAME, SAL, NVL(COMM,0), EMPNO
        FROM EMP;

        V_NOM EMP.ENAME%TYPE;
        V_SAL EMP.SAL%TYPE;
        V_COMM EMP.COMM%TYPE;
        V_COD EMP.EMPNO%TYPE;
        V_SUM INTEGER;
```

```
BEGIN

        OPEN C_SUMASAL;
        LOOP
        FETCH C_SUMASAL INTO V_NOM, V_SAL, V_COMM, V_COD;
        EXIT WHEN C_SUMASAL%NOTFOUND;

        V_SUM := V_SAL+V_COMM;

        IF V_SUM > 2000 AND V_COMM != 0 THEN
        UPDATE emp
        SET
                SAL = V_SUM
        WHERE
                empno = V_COD;
        END IF;

        END LOOP;
        CLOSE C_SUMASAL;

END;

SELECT ENAME, SAL, COMM FROM EMP;

ROLLBACK;


--9. Escribir un programa que visualice el apellido y el salario de los cinco empleados que
--tienen el salario más alto.

SELECT ENAME, SAL
FROM EMP
ORDER BY 2 DESC;

SELECT ENAME, SAL
FROM (  SELECT ENAME, SAL
        FROM EMP
        ORDER BY 2 DESC)
WHERE ROWNUM <=5;


DECLARE

        CURSOR C_CINCO IS
        SELECT ENAME, SAL
        FROM (  SELECT ENAME, SAL
                FROM EMP
                ORDER BY 2 DESC)
```

```
        WHERE ROWNUM <=5;

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

BEGIN

        OPEN C_CINCO;
        LOOP
        FETCH C_CINCO INTO V_NOM, V_SAL;
        EXIT WHEN C_CINCO%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE( V_NOM ||' '||V_SAL);
        END LOOP;
        CLOSE C_CINCO;

END;
```

--10.Codificar un programa que visualice los dos empleados que ganan menos de cada oficio.

```
DECLARE

        CURSOR C_JOB IS
        SELECT DISTINCT JOB
        FROM EMP;

        CURSOR C_MENOS(p_job EMP.JOB%TYPE) IS
        SELECT ENAME, SAL
        FROM (  SELECT ENAME, SAL
                FROM EMP
                WHERE UPPER(JOB) = upper(p_job)
                ORDER BY 2)
        WHERE ROWNUM <=2;

        V_NOM EMP.ENAME%TYPE;
        V_SAL EMP.SAL%TYPE;
        V_JOB EMP.JOB%TYPE;


BEGIN

        OPEN C_JOB;
        LOOP
        FETCH C_JOB INTO V_JOB;
        EXIT WHEN C_JOB%NOTFOUND;
```

```
        OPEN C_MENOS(V_JOB);
        LOOP
        FETCH C_MENOS INTO V_NOM, V_SAL;
        EXIT WHEN C_MENOS%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE(V_JOB||' '|| V_NOM ||' '||V_SAL);
        END LOOP;
        CLOSE C_MENOS;

        END LOOP;
        CLOSE C_JOB;


END;


-- 11. 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 JOB, ROUND(AVG(SAL),2)
FROM EMP
GROUP BY JOB;

CREATE OR REPLACE PROCEDURE spSalJob
IS

        CURSOR C_MEDIA IS
        SELECT JOB, ROUND(AVG(SAL),2)
        FROM EMP
        GROUP BY JOB;

        CURSOR C_EMP IS
        SELECT EMPNO, SAL
        FROM EMP;

        V_MEDIA EMP.SAL%TYPE;
        V_JOB EMP.JOB%TYPE;
        V_COD EMP.EMPNO%TYPE;
        V_SAL EMP.SAL%TYPE;
        V_DIF INTEGER;

BEGIN
```

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

        OPEN C_EMP;
        LOOP
        FETCH C_EMP INTO V_COD, V_SAL;
        EXIT WHEN C_EMP%NOTFOUND;
        IF V_MEDIA > V_SAL THEN
                V_DIF := V_MEDIA -V_SAL;

                UPDATE emp
                SET
                SAL = SAL + V_DIF*0.5
                WHERE
                SAL < V_MEDIA
                AND job = V_JOB;
        ELSE
                V_DIF := V_SAL -V_MEDIA;

                UPDATE emp
                SET
                SAL = SAL + V_DIF*0.5
                WHERE
                SAL < V_MEDIA
                AND job = V_JOB;
        END IF;
        END LOOP;
        CLOSE C_EMP;

        END LOOP;
        CLOSE C_MEDIA;

END;

--12.Crear un procedimiento que inserte un empleado en la tabla EMP. Su número será el
--posterior al del empleado de mayor número y la fecha de incorporación a la empresa
--será la actual. Se le pondrá un salario igual al salario medio. El departamento será el
--de Sevilla, y el apellido se recibirá como parámetro.



--13.Realizar un procedimiento para borrar un empleado recibiendo como parámetro el
--número de empleado.

CREATE OR REPLACE PROCEDURE spBorrar(p_num EMP.EMPNO%TYPE)
IS
```

```
BEGIN

        DELETE
        FROM EMP
        WHERE EMPNO = p_num;

END;


/*
Crear un cursor para ver todos los clientes que no hayan
hecho pagos. Hazlo con un loop.
*/
DECLARE

   CURSOR c_clientes_sin_pago IS
      select distinct c.nombrecliente
      from clientes c, pagos p
      where c.codigocliente= p.codigocliente(+)
      and p.codigocliente is null;

   V_nombrecliente clientes.nombrecliente%TYPE;

BEGIN

   OPEN c_clientes_sin_pago;
   LOOP
      FETCH c_clientes_sin_pago INTO  V_nombrecliente;
      EXIT WHEN c_clientes_sin_pago%NOTFOUND;

      DBMS_OUTPUT.PUT_LINE('EL cliente '|| V_nombrecliente||' no ha hecho pago');

   END LOOP;

   CLOSE c_clientes_sin_pago;

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

END;


--8. Buscar todos los empleados que tienen un salario + comisi�n superior a 2000 y
--asignarles como nuevo salario esta suma. S�lo para los que tienen comisi�n.

DECLARE
```

```
   CURSOR C_SUMASAL IS
      SELECT ENAME, SAL, NVL(COMM,0), EMPNO
      FROM EMP;

   V_NOM EMP.ENAME%TYPE;
   V_SAL EMP.SAL%TYPE;
   V_COMM EMP.COMM%TYPE;
   V_COD EMP.EMPNO%TYPE;
   V_SUM INTEGER;

BEGIN

   OPEN C_SUMASAL;
   LOOP
      FETCH C_SUMASAL INTO V_NOM, V_SAL, V_COMM, V_COD;
      EXIT WHEN C_SUMASAL%NOTFOUND;

      V_SUM := V_SAL+V_COMM;

      IF V_SUM > 2000 AND V_COMM != 0 THEN
         UPDATE emp
         SET
            SAL = V_SUM
         WHERE
            empno = V_COD;
      END IF;

   END LOOP;
   CLOSE C_SUMASAL;

END;<
```