**Ejercicio-1**

Crea un procedimiento que visualice el nombre y la fecha de alta de todos los empleados ordenados por nombre.

```
CREATE OR REPLACE PROCEDURE verEmpleado1
AS
    CURSOR cEmple IS  SELECT apell, fechalt FROM empleado ORDER BY apell;
    vapell VARCHAR2(10);
    vFecha DATE;
BEGIN
    OPEN cEmple;
    FETCH cEmple INTO vapell, vFecha;
    WHILE cEmple%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE( vApell||'    '||vFecha);
    FETCH cEmple INTO vapell, vFecha;
    END LOOP;
    CLOSE cEmple;
END verEmpleado1;
```

--De otra manera más fácil.

```
CREATE OR REPLACE PROCEDURE verEmpleado2
AS
    CURSOR cEmple IS SELECT apell, fechalt FROM empleado ORDER BY apell;
BEGIN
    FOR vReg IN cEmple LOOP
        DBMS_OUTPUT.PUT_LINE( vReg.apell ||'   '||vReg.fechalt);
    END LOOP;
END verEmpleado2;
```

--Bloque para probar el procedimiento

```
BEGIN
    Verempleado1;
END;
```

**Ejercicio-2**

Procedimiento que muestre el nombre de cada departamento y el número de empleados que tiene, incluidos los departamentos que no tienen empleados.

```
CREATE OR REPLACE PROCEDURE verEmpleDepart
AS
    CURSOR cEmple IS
    SELECT nombredep, COUNT(numemp)
        FROM empleado, departamento  WHERE numedep = depnume(+) GROUP BY nombredep;
    vdnombre departamento.nombredep%TYPE;
    vnumemple INTEGER;
BEGIN
    OPEN cEmple;
    FETCH cEmple INTO vdnombre, vnumemple;
    WHILE cEmple%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(vdnombre||' --> '||vnumemple);
        FETCH cEmple INTO vdnombre, vnumemple;
    END LOOP;
    CLOSE cEmple;
END verEmpleDepart;
```

--Usando FOR… LOOP

```
CREATE OR REPLACE PROCEDURE verEmpleDepart2
AS CURSOR cEmpleDepart IS
    SELECT nombredep, COUNT(numemp) total
        FROM empleado, departamento WHERE numedep = depnume(+) GROUP BY nombredep;
    vnomDept   departamento.nombredep%TYPE;
    vnumemple  number;
BEGIN
      FOR vreg IN  cEmpleDepart LOOP
          vnomDept:= vreg.nombredep;
          vnumemple:= vreg.total;
          DBMS_OUTPUT.PUT_LINE ('Departamento: '||vnomDept||'  Total emple: '||vnumemple);
      END LOOP;
END verEmpleDepart2;

--Bloque para probarlos
BEGIN
      verEmpleDepart2;
END;
```

**Ejercicio-3**

Modificar el bloque anterior para que al final muestre también el número de empleados que tiene la empresa.

**Solución**

```
CREATE OR REPLACE PROCEDURE verEmpleDepart3  AS
    CURSOR cEmpleDepart IS
        SELECT nombredep, COUNT(numemp) Total
            FROM empleado, departamento  WHERE numedep = depnume(+) GROUP BY nombredep;
    vdnombre departamento.nombredep%TYPE;
    vnumemple NUMBER;
    vtotalEmple  NUMBER DEFAULT 0;
BEGIN
    FOR vreg IN  cEmpleDepart LOOP
        vdnombre:= vreg.nombredep;
        vnumemple:= vreg.Total;
        DBMS_OUTPUT.PUT_LINE ('Departamento: '||vdnombre||'  Total emple: '||vnumemple);
    END LOOP;
    SELECT COUNT(*) INTO vtotalEmple FROM empleado;
    DBMS_OUTPUT.PUT_LINE ('Total empleados en la empresa: '||vtotalEmple);
END verEmpleDepart3;


--Bloque para probarlos
BEGIN
    verEmpleDepart3;
END;
```

**Ejercicio-4**

Procedimiento  que visualice el apellido y el salario de los cinco empleados que tienen el salario más alto

**Solución**

```
CREATE OR REPLACE PROCEDURE empe5maxsal AS
    CURSOR cemple IS SELECT apell, salario FROM empleado ORDER BY NVL(salario,0)  DESC;
    vEmple cemple%ROWTYPE;
    conta NUMBER;
BEGIN
    conta:=1;
    OPEN cemple;
    FETCH cemple INTO vEmple;
    WHILE cemple%FOUND AND conta<=5 LOOP
    DBMS_OUTPUT.PUT_LINE(vEmple.apell ||' -> '|| vEmple.salario);
    FETCH cemple INTO vEmple;
```

```
        conta:= conta+1;
        END LOOP;
        CLOSE cemple;
    END empe5maxsal;

    --Bloque para probarlo
    BEGIN
        empe5maxsal;
    END;
```

**Ejercicio-5**

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

**Solución**

```
    CREATE OR REPLACE PROCEDURE empe2minsal AS
        CURSOR cemp IS SELECT apell, oficio, salario FROM empleado ORDER BY oficio, salario;
        vregemp cemp%ROWTYPE;
        coficio   empleado.oficio%TYPE;
        conta  NUMBER;
    BEGIN
          OPEN cemp;
          coficio:= '*';
          FETCH cemp INTO vregemp;
          WHILE cemp%FOUND LOOP
              IF coficio <> vregemp.oficio THEN
                    coficio := vregemp.oficio;
                    conta:= 1;
              END IF;
              IF conta <= 2 THEN
                  DBMS_OUTPUT.PUT_LINE (vregemp.oficio||' -> '||vregemp.apell||' -> '||vregemp.salario);
              END IF;
              FETCH cemp INTO vregemp;
              conta:= conta+1;
          END LOOP;
          CLOSE cemp;
    END empe2minsal;

    --Bloque para probarlo
    BEGIN
    empe2minsal;
    END;
```

De otra manera.

```
CREATE OR REPLACE PROCEDURE empe2minsal2 IS
    CURSOR coficio IS SELECT DISTINCT oficio FROM empleado;
    CURSOR cempoficio (poficio varchar2) IS
    SELECT numemp, apell, salario FROM empleado  WHERE oficio = poficio ORDER BY salario;
    regcoficio    coficio%ROWTYPE;
    regcempoficio  cempoficio%ROWTYPE;
    conta number;
BEGIN
      FOR regcoficio IN coficio LOOP
            conta:=0;
            DBMS_OUTPUT.PUT_LINE ('Empleados del oficio : '||regcoficio.oficio);
            OPEN cempoficio(regcoficio.oficio);
            FETCH cempoficio INTO regcempoficio;
            WHILE cempoficio%FOUND AND conta<2 LOOP
                  conta:= conta+1;
                  DBMS_OUTPUT.PUT_LINE('Numero empleado '||regcempoficio.numemp
                  ||'  Nombre empleado '||regcempoficio.apell || '  Salario '||regcempoficio.salario );
                  fetch cempoficio INTO regcempoficio;
            END LOOP;
            CLOSE cempoficio;
            DBMS_OUTPUT.PUT_LINE(' ');
      END LOOP;
END empe2minsal2;


BEGIN
    empe2minsal2;
END;
```

**Ejercicio-6**

Realizar un procedimiento que pasado un departamento como parámetro, muestre los empleados que tiene ordenado por apellidos.

**Solución**

```
CREATE OR REPLACE PROCEDURE pempledepart(dep number) IS
        CURSOR cursor1(dep NUMBER) IS SELECT * FROM empleado WHERE depnume = dep;
BEGIN
FOR vReg IN cursor1(dep) LOOP
        DBMS_OUTPUT.PUT_LINE (vReg.numemp||' ' ||vReg.apell||' '||vReg.oficio||' ' ||vReg.depnume);
```

```
END LOOP;
END pempledepart;

--Bloque que comprueba el funcionamiento
BEGIN
pempledepart(20);
END;
```