UF3_PAC02

Ejercicios

1. Programa anónimo que visualice el nombre y la localidad de todos los departamentos.

```
declare
    cursor nombreLocalidad is select dept.dnombrebre, dept.lugar
from dept;
  nombre dept.dnombrebre%type;
  localidad dept.lugar%type;
begin
  open nombreLocalidad;
  Loop
    fetch nombreLocalidad into nombre, localidad;
      exit when nombreLocalidad%notfound;
      DBMS_OUTPUT.PUT_LINE(nombre||' - '|| localidad);
    end loop;
   Close nombrelocalidad;
end;
```

2. Programa que muestre los apellidos de los empleados que pertenecen al departamento de VENTAS. Hay que numerar cada línea secuencialmente.

```
Ccreate or replace procedure verEmpVentas
as
    cursor cursorEmpVentas is select apellido from EMP where
emp.dept_no=(select dept_no from dept where dnombrebre='VENTAS');
  varApellido emp.apellido%type;
  iterador number;
begin
  open cursorEmpVentas;
  iterador:=1;
  Loop
    fetch cursorEmpVentas into varApellido;
    exit when cursorEmpVentas%notfound;
    DBMS_OUTPUT.PUT_Line(iterador||' - ' || varApellido);
    iterador:=iterador+1;
  end loop;
  close cursorEmpVentas;
end;
```

3. Crea un procedimiento que visualice por pantalla los apellidos de los empleados del departamento que se recibe como argumento (se recibe el código del departamento). Haz dos versiones, una usando un " for " y otra sin usarlo. Ambas versiones tienen que informar en el caso de que no se muestre ningún empleado (bien porque el departamento pedido no exista o que no tenga empleados).

```
create or replace procedure verEmpVentasParametro (numeroDept
number)
as
    cursor cursorEmpVentas is select apellido from EMP where
emp.dept_no=numeroDept;
  varApellido emp.apellido%type;
  iterador number;
begin
  open cursorEmpVentas;
  iterador:=1;
  Loop
    fetch cursorEmpVentas into varApellido;
    exit when cursorEmpVentas%notfound;
    DBMS_OUTPUT.PUT_Line(iterador||' - ' || varApellido);
    iterador:=iterador+1;
  end loop;
  IF iterador=0 then
     DBMS_OUTPUT.PUT_Line('No se han encontrado empleados.');
  end if;
  close cursorEmpVentas;
end;
```

4. Crear una función que sirva para calcular cuántos empleados del departamento que pasamos por parámetro cobran comisión (significa comisión mayor que 0).
Pasamos por parámetro el código del departamento.
Hazlo sin utilizar la función de agregación COUNT ( ).
Si el departamento no existe la función debe devolver el valor NULL.
Si el departamento no tiene ningún empleado que cobre comisión la función debe devolver un 0.

```
create or replace function cuentacomisiones(numeroDepartamento
number)
return number
is
   cursor empleadillos( deptoNumber number) is select * from emp
where emp.dept_no=deptoNumber;
  empleado emp%rowtype;
  contador number;
begin
  contador:=0;
  open empleadillos(numeroDepartamento);
  Loop
    FETCH empleadillos INTO empleado;
    exit when empleadillos%notfound;
    if(empleado.comision>0) then
      contador:=contador+1;
    end if;
  end loop;
  close empleadillos;
  return contador;
end;
```

5. Haz un procedimiento que utilizando la función del ejercicio 4 muestre por pantalla el código y nombre de los departamentos en los que menos de la mitad de su empleados cobran comisión.

Si el departamento no tiene ningún empleado debe mostrar un mensaje informativo.

```
create or replace procedure menosComisiones
as
  cursor departamentos is select * from dept;
  auxDept dept%rowtype;
  auxTotal number;
begin

  for auxDept in departamentos loop
           select   count(*)   into   auxTotal   from   emp   where
DEPT_NO=auxDept.dept_no;
    if cuentacomisiones(auxDept.dept_no)< (auxTotal/2) then
            dbms_output.put_line(auxDept.dept_no  ||  '  --  '  ||
auxDept.DNOMBREBRE);
    end if;
  end loop;

end;
```

6. Programa que muestre por cada departamento una línea con el número de empleados y la suma de los salarios del departamento. A continuación la lista de los apellidos y salarios de los empleados de este departamento. Al final del listado: El número total de empleados de la empresa y suma de todos los salarios.

```
create or replace procedure listaEmpleados
as
  cursor departamentos is select * from dept;
  auxDept dept%rowtype;
   cursor empleados(parametroDeptNo number) is select * from emp
where dept_no=parametroDeptNo;
  empleado emp%rowtype;
  totalEmpTemp number;
  totalSalarioTemp number;
  totalEmpFinal number;
  totalSalarioFinal number;
begin
  totalEmpFinal:=0;
  totalSalarioFinal:=0;
  for auxDept in departamentos loop
        select count(salario), sum(salario) into totalEmpTemp,
totalSalarioTemp from emp where emp.dept_no=auxDept.dept_no;
      dbms_output.put_line('DEPARTAMENTO. '|| auxDept.dept_no ||'
Total Emp: ' ||totalEmpTemp || ' Suma salario: ' ||
totalSalarioTemp);

    for empleado in empleados(auxDept.dept_no) loop
         dbms_output.put_line('            '||empleado.apellido ||
'---'|| empleado.salario);
    end loop;

    totalEmpFinal:=totalEmpFinal+totalEmpTemp;
    if totalSalarioTemp is not null then
      totalSalarioFinal:=totalSalarioFinal+totalSalarioTemp;
    end if;
  end loop;
   dbms_output.put_line('TOTAL EMPLEADOS--> '|| totalEmpFinal ||
'----TOTAL SALARIOS--> '|| totalSalarioFinal);
end;
```

7. Hacer un procedimiento que reciba por parámetro el nombre de un departamento y el porcentaje de subida de salario. El procedimiento incrementará el salario el porcentaje que le indicamos a los empleados que pertenecen al departamento recibido e informará de cuántos empleados se les ha modificado el salario. El porcentaje se recibirá en tanto por 1 (es decir, el 10% sería 0,1 etc) .

```
create or replace procedure subeSueldos (deptNumber number, inc
number)
as
  cursor empleados(parametreoDeptNo number) is select salario from
emp where DEPT_NO=parametreoDeptNo for update;
  thisSalario number;
begin

  for thisSalario in empleados(deptNumber) loop
    update emp
    set salario=salario+ (Salario*inc)
    where current of empleados;
  end loop;
end;
```

8. Crear un procedimiento que reciba el número de empleado y la cantidad que se incrementa el salario del empleado correspondiente. Utilice dos excepciones: una definida por el usuario (salario_null) y otra predeterminada (NO_DATA_FOUND)

```
create or replace PROCEDURE subeUnSueldo(empNumber
emp.emp_no%type, incremento number)
AS
valor_cero exception;
salarioOld emp.salario%type;
BEGIN
  select salario into salarioOld from emp where emp_no=empNumber;
      update emp set salario=salarioOld+incremento where
emp_no=empNumber;
  if salarioOld is null then
       raise valor_cero;
   end if;
exception
when no_data_found then dbms_output.put_line ('no datos');
when valor_cero then dbms_output.put_line ('no existe emple');
END;
```

9. Procedimiento que permita subir 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. Hay que terminar la transacción y gestionar los posibles errores.

```
CREATE OR REPLACE PROCEDURE subeSueldosMenores
AS
cursor c is select salario, oficio from emp for update;
salarioMedio number(15,4);
BEGIN
   FOR cfile IN c
   LOOP
        select avg(salario) into salarioMedio from emp where
oficio=cfile.oficio;
        if cfile.salario<salarioMedio then
        update emp set
salario=salario+((salarioMedio-cfile.salario)/2) where current of
c;
        end if;
   END LOOP;
 exception
when no_data_found then dbms_output.put_line ('no datos');
END;
```