Práctica de PROCEDIMIENTOS Y PARÁMETROS

1- Crear un procedimiento llamado “visualizar” que visualice el nombre y salario de todos los empleados.

```
create  or replace procedure MostrarTodosLosEmpleados is
cursor empleado is select * from employees;
begin
    for empleados in empleado loop
    DBMS_OUTPUT.PUT_LINE('ID: ' || empleados.employee_id);
    DBMS_OUTPUT.PUT_LINE('Primer nombre: ' || empleados.first_name);
    DBMS_OUTPUT.PUT_LINE('Apellido: ' || empleados.last_name);
    DBMS_OUTPUT.PUT_LINE('Salario: ' || empleados.salary);
    end loop;

end;
```

```
begin
 MostrarTodosLosEmpleados;
end;
/*Práctica de FUNCIONES
```

```
Salario: 4000
ID: 193
Primer nombre: Britney
Apellido: Everett
Salario: 3900
ID: 194
Primer nombre: Samuel
Apellido: McCain
Salario: 3200
ID: 195
Primer nombre: Vance
Apellido: Jones
Salario: 2800
ID: 196
Primer nombre: Alana
```

2- Modificar el programa anterior para incluir un parámetro que pase el número de departamento para que visualice solo los empleados de ese departamento

- Debe devolver el número de empleados en una variable de tipo OUT

```
create  or replace procedure MostrarEmpleadosPorDepartamento(NUMERO_DEPARTAMENTO IN DEPARTMENTS.DEPARTMENT_ID%TYPE, NUMERO_EMPLEADOS OUT NUMBER) is
cursor empleado is select * from employees WHERE DEPARTMENT_ID=NUMERO_DEPARTAMENTO;
begin
    NUMERO_EMPLEADOS:=0;
    for empleados in empleado loop
    DBMS_OUTPUT.PUT_LINE('ID: ' || empleados.employee_id);
    DBMS_OUTPUT.PUT_LINE('Primer nombre: ' || empleados.first_name);
    DBMS_OUTPUT.PUT_LINE('Apellido: ' || empleados.last_name);
    DBMS_OUTPUT.PUT_LINE('Salario: ' || empleados.salary);
    NUMERO_EMPLEADOS:=NUMERO_EMPLEADOS+1;
    end loop;

end;
DECLARE
numeroEmpleados number:=0;
begin
 MostrarEmpleadosPorDepartamento(60,numeroEmpleados);
 DBMS_OUTPUT.PUT_LINE('');
 DBMS_OUTPUT.PUT_LINE('numero de empleados: ' || numeroEmpleados);
end;
```

```
Apellido: Austin
Salario: 4800
ID: 106
Primer nombre: Valli
Apellido: Pataballa
Salario: 4800
ID: 107
Primer nombre: Diana
Apellido: Lorentz
Salario: 4200

numero de empleados: 5
```

3- Crear un bloque por el cual se de formato a un número de cuenta

suministrado por completo, por ejmplo: 11111111111111111111

- Formateado a: 1111-1111-11-1111111111
- Debemos usar un parámetro de tipo IN-OUT

```
create or replace procedure formatoNumeroDeCuenta(numero_cuenta in out varchar2) IS
str1 varchar2(5):='';
str2 varchar2(5):='';
str3 varchar2(3):='';
str4 varchar2(10):='';
begin
str1:=substr(numero_cuenta,1,4)||'-';
str2:=substr(numero_cuenta,5,4)||'-';
str3:=substr(numero_cuenta,9,2)||'-';
str4:=substr(numero_cuenta,11,10);
numero_cuenta:=str1|| str2 || str3 || str4;
end;
```

```
DECLARE
numeroCuenta varchar(30):='01234567890123456789';
begin
formatoNumeroDeCuenta(numeroCuenta);
 DBMS_OUTPUT.PUT_LINE('NC: ' || numeroCuenta);

end;
```

```
NC: 0123-4567-89-0123456789

Procedimiento PL/SQL terminado correctamente.
```

Práctica de FUNCIONES

1. Crear una función que tenga como parámetro un número de departamento y que devuelve la suma de los salarios de dicho departamento. La imprimimos por pantalla.

- Si el departamento no existe debemos generar una excepción con dicho mensaje
- Si el departamento existe, pero no hay empleados dentro, también debemos generar una excepción para indicarlo

```
create or replace function SumaSalariosPorDepartamentoFunction(departmentId departments.department_id%TYPE)
RETURN NUMBER
IS
totalSalarios number:=0;
numeroEmpleados number:=0;
noHayEmpleados exception;
dataDep departments%rowtype;
begin
    SELECT * INTO dataDep from departments WHERE department_id=departmentId;
    select sum(salary) into totalSalarios from employees where department_id=departmentId;
    select count(*) into numeroEmpleados from employees where department_id=departmentId;
    if numeroEmpleados=0 then
    raise noHayEmpleados;
    end if;
    return totalSalarios;
Exception
    when NO_DATA_FOUND then
    dbms_output.put_line('ese departamento no existe');
    totalSalarios:=-1;
    return totalSalarios;
    when noHayEmpleados then
    dbms_output.put_line('no hay empleados en ese dept');
    totalSalarios:=-1;
    return totalSalarios;
END;
```

```
begin
    dbms_output.put_line(numeroDeEpleadosPorDepartamentoFunction(120));
end;
```

```
no hay empleados en ese dept
-1
```

```
begin
    dbms_output.put_line(SumaSalariosPorDepartamentoFunction(60));
end;
```

```
28800

Procedimiento PL/SQL terminado correctamente.
```

2. Modificar el programa anterior para incluir un parámetro de tipo OUT por el que vaya el número de empleados afectados por la query. Debe ser visualizada en el programa que llama a la función. De esta forma vemos que se puede usar este tipo de parámetros también en una función

```
create or replace function SumaSalariosPorDepartamentoFunction(departmentId departments.department_id%TYPE, numeroEmpleados out number)
RETURN NUMBER
IS
totalSalarios number:=0;
noHayEmpleados exception;
dataDep departments%rowtype;
begin
    numeroEmpleados:=0;
    SELECT * INTO dataDep from departments WHERE department_id=departmentId;
    select sum(salary) into totalSalarios from employees where department_id=departmentId;
    select count(*) into numeroEmpleados from employees where department_id=departmentId;
    if numeroEmpleados=0 then
    raise noHayEmpleados;
    end if;
    return totalSalarios;
Exception
    when NO_DATA_FOUND then
    dbms_output.put_line('ese departamento no existe');
    totalSalarios:=-1;
    return totalSalarios;
    when noHayEmpleados then
    dbms_output.put_line('no hay empleados en ese dept');
    totalSalarios:=-1;
    return totalSalarios;
END;
```

```
numeroEmpleadosAfectados number:=0;
begin
    dbms_output.put_line(SumaSalariosPorDepartamentoFunction(60,numeroEmpleadosAfectados));
    dbms_output.put_line('empleados afectados: '|| numeroEmpleadosAfectados);
end;
```

```
28800
empleados afectados: 5

Procedimiento PL/SQL terminado correctamente.
```

3. Crear una función llamada CREAR_REGION,

- A la función se le debe pasar como parámetro un nombre de región y debe devolver un número, que es el código de región que calculamos dentro de la función
- Se debe crear una nueva fila con el nombre de esa REGION
- El código de la región se debe calcular de forma automática. Para ello se debe averiguar cual es el código de región más alto que tenemos en la tabla en ese momento, le sumamos 1 y el resultado lo ponemos como el código para la nueva región que estamos creando.
- Si tenemos algún problema debemos generar un error
- La función debe devolver el número que ha asignado a la región

```
create or replace function CREAR_REGION(nombre_region varchar2)
return number
is
ultimoIdRegion number;
verificadorNombreExiste regions%rowtype;
begin
    SELECT * INTO verificadorNombreExiste FROM regions WHERE LOWER(region_name) = LOWER(nombre_region);
    dbms_output.put_line('ESE NOMBRE YA EXISTE');
    return -1;
exception
    when NO_DATA_FOUND THEN
    select max(region_id) into ultimoIdRegion from regions;
    insert into regions values(ultimoIdRegion+1, nombre_region);
    return ultimoIdRegion+1;
end;

declare
nombreACrear varchar2(200):='europe';
begin
dbms_output.put_line(CREAR_REGION(nombreACrear));
end;
```

```
ESE NOMBRE YA EXISTE
-1

Procedimiento PL/SQL terminado correctamente.
```

```
declare
nombreACrear varchar2(200):='kazajistan';
begin
dbms_output.put_line(CREAR_REGION(nombreACrear));
end;

7

Procedimiento PL/SQL terminado correctamente.
```