1.- Dado el siguiente procedimiento almacenado:

```
CREATE OR REPLACE PROCEDURE CrearDepart (VNumDep depart.dept_no%type, VNombreD
depart.dnombre%type DEFAULT 'PROVISIONAL',VNUMCE depart.numce%type DEFAULT 7)
IS

BEGIN

INSERT INTO DEPART  (DEPT_NO, DNOMBRE, NUMCE) VALUES
(VNumDep,VNombreD,VNUMCE);

DBMS_OUTPUT.PUT_LINE('SE HA CREADO UN NUEVO DEPARTAMENTO '||VNOMBRED);

end;
```

Determinar cuál de las siguientes llamadas son correctas:

a. **CrearDepart;**
b. **CrearDepart(70);**
c. **CrearDepart('Compras');**
d. **CrearDepart(70,'Compras');**
e. **CrearDepart('Compras',7);**
f. **CrearDepart(70,'Compras',5);**
g. **CREARDEPART(90,null,3);**

2.- Codifica un procedimiento que reciba una cadena y la visualice al revés. E invocar al procedimiento.

```
create or replace PROCEDURE VerCadInvertida (cad IN VARCHAR2) IS
BEGIN
   dbms_output.put_line('La cadena '||cad||' invertida es:');
FOR VAR IN REVERSE 1..LENGTH(cad)
LOOP
    dbms_output.put(SUBSTR(cad,VAR,1));
END LOOP;
  dbms_output.put_line(''); -- Para liberar el buffer
END;
```

```
create or replace PROCEDURE VerCadInvertida (cad IN VARCHAR2) IS

VAR INT;

BEGIN

   dbms_output.put_line('La cadena '||cad||' invertida es:');

VAR:=LENGTH(CAD);

WHILE VAR >=1

LOOP

    dbms_output.put(SUBSTR(cad,VAR,1));

    VAR:=VAR-1;

END LOOP;
```

```
  dbms_output.put_line(''); -- Para liberar el buffer

END;
```

```
-- Llamada al método
set serveroutput on;
begin
   vercadinvertida('hola que tal');
end;
/
```

3.- Realiza una función que reciba una cadena y devuelva la cadena invertida. Poned un ejemplo de llamada a la función.

```
create or replace FUNCTION CadInvertida (cad IN VARCHAR2) RETURN VARCHAR2 IS
inv VARCHAR2(40);
BEGIN

FOR VAR IN REVERSE 1..LENGTH(cad)
LOOP
   INV:=INV||SUBSTR(cad,VAR,1);
END LOOP;
  RETURN INV;
END;
```

```
create or replace FUNCTION CadInvertida (cad IN VARCHAR2) RETURN VARCHAR2 IS

inv VARCHAR2(40):='';

VAR INT:=LENGTH(CAD);

BEGIN

  WHILE VAR>=1

LOOP

   INV:=INV||SUBSTR(cad,VAR,1);

   VAR:=VAR-1;

END LOOP;

  RETURN INV;

END;
```

```
-- Llamada a la funcion
set serveroutput on;
declare
cad varchar2(20):='Hoy es Viernes';
begin
dbms_output.put_line('La cadena '||cad||' invertida es :' ||
cadinvertida(cad));
```

```
end;
/
```

4.- Procedimiento que permita borrar un empleado cuyo número se pasará en la llamada. Se visualizará un mensaje con el nombre del empleado borrado o un mensaje de error.

```
CREATE OR REPLACE PROCEDURE BorraEmple (nEmple emple.emp_no%type) AS
VNOMBRE EMPLE.NOMBRE%TYPE;
BEGIN
  DBMS_OUTPUT.PUT_LINE(' SE PROCEDE A BORRAR AL EMPLEADO '||NEMPLE);
  SELECT NOMBRE INTO VNOMBRE FROM EMPLE WHERE EMP_NO=NEMPLE;
  DELETE FROM EMPLE WHERE EMP_NO=nEmple;
  DBMS_OUTPUT.PUT_LINE(' EMPLEADO '||VNOMBRE||' BORRADO ');
END;
```

```
set serveroutput on;
DECLARE
NEMP EMPLE.EMP_NO%TYPE;
BEGIN
NEMP:=&DAME_EMPLEADO_A_BORRAR;
BORRAEMPLE(NEMP);
EXCEPTION
WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE(' NO EXISTE EL EMPLEADO '||NEMP);
WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(' ERROR '||SQLERRM);
END;
/
ROLLBACK;
```

Visualiza todos los procedimientos y funciones del usuario almacenados en la BD y su situación (valid/invalid):

```
Select object_name, object_type, status from user_objects where object_type in
('PROCEDURE','FUNCTION');
```

5.- Crear un procedimiento **pverempleoficio** al que le paso un oficio y muestra ~~el nombre, apellidos, oficio, nombre departamento, nombre del departamento de los empleados que tienen ese oficio~~ el número de empleados de ese oficio.

```
create or replace FUNCTION fvercuantosoficio (POFICIO emple.OFICIO%type)
RETURN INT
AS
TYPE INFO IS RECORD
(VNOMBRE EMPLE.NOMBRE%TYPE,
VAPEL EMPLE.APELLIDO%TYPE,
VOFICIO EMPLE.OFICIO%TYPE,
VNOMBRED DEPART.DNOMBRE%TYPE);
infoEmple INFO;
 CONT INT;
BEGIN
 SELECT COUNT(*) INTO CONT FROM EMPLE WHERE upper(oficio)=upper(pOFICIO);
 IF CONT !=0 THEN
```

```
 SELECT NOMBRE,apellido,oficio,DNOMBRE INTO infoEmple  FROM EMPLE E,
 depart d WHERE e.dept_no=d.dept_no and UPPER(e.oficio)=UPPER(POFICIO);
 Dbms_Output.Put_Line (Infoemple.Vnombre||' '||Infoemple.Vapel||' '||
Infoemple.Voficio||' '||Infoemple.Vnombred);
 END IF;
RETURN CONT;
EXCEPTION
WHEN TOO_MANY_ROWS
THEN dbms_output.put_line('DEMASIADOS REGISTROS, HAY QUE USAR UN CURSOR !!!');
WHEN NO_DATA_FOUND THEN dbms_output.put_line('NO HAY NINGUN REGISTRO, DE
OFICIO : '||POFICIO);
WHEN OTHERS THEN dbms_output.put_line('ERROR: '||SQLERRM);
RETURN CONT;
END;
**nota: Si hay más de un empleado con ese oficio concreto se producirá el
error TOO_MANY_ROWS. Habría que utilizar un cursor.
```

```
SET SERVEROUTPUT ON;
DECLARE
OFICIO EMPLE.OFICIO%TYPE;
BEGIN
OFICIO:='&DAME_OFICIO';
pverempleoficiot(OFICIO);
EXCEPTION
WHEN TOO_MANY_ROWS THEN dbms_output.put_line('DEMASIADOS REGISTROS, HAY QUE
USAR UN CURSOR !!!');

END;
/
```

```
create or replace PROCEDURE pveroficio (POFICIO emple.OFICIO%type)
AS
numE int:=0;
BEGIN
SELECT COUNT(*) INTO numE FROM  EMPLE WHERE OFICIO=POFICIO;
DBMS_OUTPUT.PUT_LINE ('EL NUMERO DE EMPLEADOS DE '||POFICIO||' ES '||numE);
END;
```

6.- Crear un procedimiento pcalcularsueldosdep al que se le pasa un número de departamento y muestra el total de sueldos, el total de comisiones, cuantos empleados hay.

```
create or replace Procedure Pcalcularsueldosdep (Ndep Emple.Dept_No%Type) As
Numemp Number;
Sums Number;
Sumc  Number;

Begin
 Select Count(*)   , Sum(Salario) , NVL(Sum(Comision),0)   Into
Numemp,Sums,Sumc From Emple  Where Dept_No=Ndep;
 Dbms_Output.Put_Line ('NUMERO DE EMPLEADOS: '||NUMEMP||' DEL DEPARTAMENTO '||
NDEP||' TOTAL DE SUELDOS '||SUMS||' TOTAL COMISIONES '||SUMC);
End;
```

7.- Crear el procedimiento paumentosalario2 se le pasa el número de departamento ~~y va mostrando uno por uno los empleados~~ y les actualice el sueldo en un 10%.

~~- Al presidente no se le sube el salario.~~
~~- Si son Analistas un 0,5 % el salario, si son Vendedores un 0,3%, si son Directores un 1% y si son otro tipo de oficio un 0,2%.~~

```
Create Or Replace Procedure Paumentosalario2 (Ndep Emple.Dept_No%Type) As
Begin
UPDATE EMPLE SET SALARIO=SALARIO+salario*0.1 WHERE dept_no=Ndep;
Dbms_Output.Put_Line ('NUMERO DE EMPLEADOS DEL DEPARTAMENTO : '||Ndep||' ES
'||SQL%Rowcount);
End;
```

```
SET SERVEROUTPUT ON;
DECLARE
NDEP EMPLE.DEPT_NO%TYPE;
BEGIN
NDEP:=&DAME_NUMDEPT;
paumentosalario2(NDEP);
EXCEPTION
WHEN OTHERS THEN dbms_output.put_line('eRROR: '||SQLERRM);
end;
/
```

UTILIZA UNA FUNCIÓN QUE DEVUELVE TRUE SI EL DEPARTAMENTO QUE RECIBE COMO PARÁMETRO EXISTE Y FALSE EN CASO CONTRARIO.

```
Create Or Replace Procedure Paumentosalario2 (Ndep Emple.Dept_No
%Type)

As

CONT INT;

Begin

IF FEXISTE_DEPTO(Ndep) THEN

SELECT COUNT(*) INTO CONT FROM EMPLE WHERE dept_no=Ndep;

  IF CONT =0 THEN

    Dbms_Output.Put_Line ('NO HAY EMPLEADOS EN EL DEPTO. '||Ndep);

  ELSE

            UPDATE EMPLE SET SALARIO=SALARIO+salario*0.1 WHERE
dept_no=Ndep;

  Dbms_Output.Put_Line ('NUMERO DE EMPLEADOS DEL DEPARTAMENTO : '||
Ndep||' ACTUALIZADOS ES '
```

```
    ||SQL%Rowcount);
  END IF;
ELSE
Dbms_Output.Put_Line ('EL DEPTO '||Ndep||' NO EXISTE !!');
END IF;

End;

CREATE OR REPLACE FUNCTION FEXISTE_DEPTO (NDEP DEPART.DEPT_NO
%TYPE)
RETURN BOOLEAN
IS
RET BOOLEAN:=TRUE;
CONT INT;
BEGIN
SELECT COUNT(*) INTO CONT FROM DEPART WHERE DEPT_NO=NDEP;
IF CONT =0 THEN
  RET:=FALSE;
END IF;
RETURN RET;
END;
```

8. Desarrolla un procedimiento que visualice el apellido y la fecha de alta de todos los empleados ordenados por apellido. (Necesita Cursor).

```
CREATE OR REPLACE PROCEDURE VIS_FECHA_ALTA
IS
CURSOR cursorEmple IS SELECT * FROM EMPLE ORDER BY APELLIDO;
VREG emple%rowtype;
```

```
BEGIN

OPEN cursorEmple;

FETCH cursorEmple INTO VREG;

WHILE cursorEmple%FOUND LOOP

DBMS_OUTPUT.PUT_LINE(VREG.APELLIDO||', '||VREG.NOMBRE||' '||
VREG.FECHA_ALT);

FETCH cursorEmple INTO VREG;

END LOOP;

DBMS_OUTPUT.PUT_LINE('SE HAN PROCESADO '||cursorEmple
%ROWCOUNT||' Registros');

CLOSE cursorEmple;

END;

– LLAMADA:

set serveroutput on;

BEGIN

VIS_FECHA_ALTA();

END;
```

9**.- Escribe un procedimiento que reciba una cadena y visualice el número de empleados cuyo apellido contenga la cadena especificada.**

```
create or replace PROCEDURE pverCuantosApellido (PAPELLIDO emple.APELLIDO
%type) AS
cont int:=0;
CAD VARCHAR2(20);
BEGIN
CAD:='%'||PAPELLIDO||'%';
SELECT COUNT(*) INTO cont FROM EMPLE WHERE APELLIDO LIKE CAD;
DBMS_OUTPUT.PUT_LINE (CONT||' EMPLEADOS CONTIENEN '||PAPELLIDO||' EN SU
APELLIDO');
END;
```

```
SET SERVEROUTPUT ON;
DECLARE
APEL EMPLE.APELLIDO%TYPE;
```

```
BEGIN
APEL:='&DAME_APELLIDO';
pverCuantosApellido(APEL);
EXCEPTION
WHEN OTHERS THEN dbms_output.put_line('eRROR: '||SQLERRM);
END;
/
```

10.- Codifica un procedimiento que muestre el **nombre del departamento pasado por parámetro** y el número de empleados que tiene, **utilizando una función que devuelva el número de empleados del departamento que se le pase como parámetro.**

```
create or replace PROCEDURE DepCuantosEmpleado(dep emple.dept_no%type)
IS
vnombreD depart.dnombre%type;
cont number;
BEGIN
SELECT DNOMBRE INTO vnombreD FROM DEPART WHERE DEPT_NO=dep;
DBMS_OUTPUT.PUT_LINE (VNOMBRED||' => '||CuantosEmple(dep));
EXCEPTION
WHEN NO_DATA_FOUND THEN dbms_output.put_line('ERROR: '|| SQLERRM ||' NO EXISTE
EL DEPARTAMENTO '||NEP);
END;
```

```
create or replace function cuantosEmple (dep emple.dept_no%type) RETURN INT
IS
N INT:=0;
BEGIN
SELECT COUNT(*) INTO N FROM EMPLE WHERE DEPT_NO=DEP;
RETURN N;
END;
```

```
SET SERVEROUTPUT ON;
DECLARE
NDEP EMPLE.DEPT_NO%TYPE;
BEGIN
NDEP:=&DAME_NUMDEPT;
DepCuantosEmpleado(NDEP);

end;
/
```

11.- Escribe un programa que visualice el apellido y el salario del empleado con salario mayor ~~de los cinco empleados que tienen el salario más alto~~.

```
create or replace PROCEDURE salarioMayorEmpleado
IS
nombre EMPLE.nombre%type;
maxSueldo emple.salario%type;

BEGIN
SELECT MAX(salario) into maxSueldo FROM EMPLE;
```

```
SELECT NOMBRE INTO nombre FROM EMPLE WHERE SALARIO=maxSueldo;
DBMS_OUTPUT.PUT_LINE (NOMBRE||' => '||maxSueldo);
EXCEPTION
WHEN TOO_MANY_ROWS THEN DBMS_OUTPUT.PUT_LINE ('HAY MAS DE UN EMPLEADO CON
SALRIO MAXIMO'||maxSueldo);
END;
```

```
begin
salarioMayorEmpleado;
end;
```

12.- Codifica un programa que visualice los dos empleados que ganan menos de cada oficio. CURSOR

13.- Escribe un procedimiento que reciba todos los datos de un nuevo empleado y procese la transacción de alta, gestionando posibles errores. El procedimiento deberá gestionar en concreto los siguientes puntos:

- no_existe_departamento.
- no_existe_director.
- numero_empleado_duplicado.
- Salario nulo: con RAISE_APPLICATION_ERROR
- Otros posibles errores de Oracle visualizando códigos de error y el mensaje de error.

```
create or replace PROCEDURE INSERTA_EMPLE (VEMPLE IN EMPLE%ROWTYPE) IS
BEGIN

IF EXISTE_DEP(VEMPLE.DEPT_NO) THEN
  IF EXISTE_DIRECTOR THEN
      IF EXISTE_EMPLE(VEMPLE.EMP_NO)=FALSE THEN
        IF VEMPLE.SALARIO IS NULL THEN RAISE_APPLICATION_ERROR(-20001,'ERROR,
SALARIO NULO !!');
        ELSE
             DBMS_OUTPUT.PUT_LINE('SE INTRODUCE EL NUEVO EMPLEADO '||
VEMPLE.APELLIDO);
    INSERT INTO EMPLE VALUES
(VEMPLE.EMP_NO,VEMPLE.NOMBRE,VEMPLE.APELLIDO,VEMPLE.OFICIO,VEMPLE.FECHA_ALT,VE
MPLE.SALARIO,0,VEMPLE.DEPT_NO,NULL,NULL,0);
    DBMS_OUTPUT.PUT_LINE('SE HA INTRODUCIDO '||SQL%ROWCOUNT ||' EMPLEADOS');
         END IF;
    ELSE
     DBMS_OUTPUT.PUT_LINE('EMPLEADO: '||VEMPLE.EMP_NO||' YA EXISTE');
    END IF;
  ELSE
      DBMS_OUTPUT.PUT_LINE('NO HAY DIRECTOR ' );
  END IF;
ELSE
      DBMS_OUTPUT.PUT_LINE('DEPARTAMENTO '||' '||VEMPLE.DEPT_NO|| ' NO  EXISTE
' );
END IF;
```

```
END;
```

```
CREATE OR REPLACE FUNCTION EXISTE_DEP (DEP DEPART.DEPT_NO%TYPE) RETURN BOOLEAN
IS
RET BOOLEAN:=FALSE;
N INT;
BEGIN
SELECT COUNT(*) INTO N FROM DEPART WHERE DEPT_NO=DEP;
IF N!=0 THEN
RET:=TRUE;
END IF;
RETURN RET;
END;
```

```
CREATE OR REPLACE FUNCTION EXISTE_DIRECTOR RETURN BOOLEAN
IS
RET BOOLEAN:=FALSE;
N INT;
BEGIN
SELECT COUNT(*) INTO N FROM EMPLE WHERE OFICIO='DIRECTOR';
IF N!=0 THEN
RET:=TRUE;
END IF;
RETURN RET;
END;
```

```
CREATE OR REPLACE FUNCTION EXISTE_DEP (DEP DEPART.DEPT_NO%TYPE) RETURN BOOLEAN
IS
RET BOOLEAN:=FALSE;
N INT;
BEGIN
SELECT COUNT(*) INTO N FROM DEPART WHERE DEPT_NO=DEP;
IF N!=0 THEN
RET:=TRUE;
END IF;
RETURN RET;
END;
```

```
CREATE OR REPLACE FUNCTION EXISTE_EMPLE (EMP EMPLE.EMP_NO%TYPE) RETURN BOOLEAN
IS
RET BOOLEAN:=FALSE;
N INT;
BEGIN
SELECT COUNT(*) INTO N FROM EMPLE WHERE EMP_NO=EMP;
IF N!=0 THEN
RET:=TRUE;
END IF;
RETURN RET;
END;
```

```
SET SERVEROUTPUT ON;
DECLARE
VAR EMPLE%ROWTYPE;
BEGIN
VAR.EMP_NO:=&NUMERO_EMPLEADO;
VAR.NOMBRE:='&NOMBRE_EMPLEADO';
VAR.APELLIDO:='&APELLIDO_EMPLEADO';
```

```
VAR.OFICIO:='&OFICIO_EMPLEADO';
VAR.SALARIO:=&SUELDO_EMPLEADO;
VAR.FECHA_ALT:=SYSDATE();
VAR.DEPT_NO:=&NUMERO_DEPARTAMENTO;
DATOSEMPLE(VAR);
end;
/
```

14.- Desarrolla un procedimiento que permita insertar nuevos departamentos según las siguientes especificaciones.

- Se pasará al procedimiento el nombre del departamento y la localidad.
- El procedimiento insertará la fila nueva asignando como número de departamento la decena siguiente al número mayor de la tabla
- Se incluirá la gestión de posibles errores.

15.- Escribe 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 por cien de la diferencia entre el salario del y la media de su oficio. Se deberá hacer que la transacción no se quede a medias, y se gestionarán los posibles errores. CURSOR