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

```
CREATE OR REPLACE PROCEDURE NombreFecha
IS
CURSOR c_nombre IS
SELECT ename, hiredate FROM emp ORDER BY hiredate DESC;
v_nombre c_nombre%ROWTYPE; /*USAS ROWTYPE*/
BEGIN
        OPEN c_nombre;
        FETCH c_nombre INTO v_nombre;
        WHILE c_nombre %FOUND LOOP
       dbms_output.put_line (' el nombre y la fecha son: '|| v_nombre.ename || ' y '||
v_nombre.hiredate);
        FETCH c_nombre INTO v_nombre;
        END LOOP;
        CLOSE c_nombre;
        EXCEPTION
                WHEN INVALID_CURSOR THEN /*TIPO DE EXCEPCIONES
HABITUALES EN UN PROCEDIMIENTO*/
                dbms_output.put_line (' operación con el cursor invalida');
                WHEN OTHERS THEN
                dbms_output.put_line (' error en el procedimiento NombreFecha');
END;
/

set serveroutput on;

EXECUTE NombreFecha ();
```

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

```
CREATE OR REPLACE PROCEDURE EmpleadoConSueldoMayor
IS
CURSOR c_nombre IS
SELECT ename FROM emp WHERE sal>2000 ORDER BY sal ASC;
v_nombre c_nombre%ROWTYPE;
BEGIN
        OPEN c_nombre;
        FETCH c_nombre INTO v_nombre;
          WHILE c_nombre%FOUND AND c_nombre%ROWCOUNT <=1 LOOP
/*USAS ROWCOUNT EN EL WHILE*/
           dbms_output.put_line (' el nombre del empleado con mayor sueldo es: '||
v_nombre.ename);
        FETCH c_nombre INTO v_nombre;
        END LOOP;
      IF c_nombre%ROWCOUNT <1 THEN /*USAS UN IF PARA COMPROBAR
QUE SI NO TIENE MAS DE UN EMPLEADO, NO EMPLEA EL ELSE*/
                        dbms_output.put_line('no tiene mas de un empleado ' );
        END IF;
        CLOSE c_nombre;
        EXCEPTION
                WHEN INVALID_CURSOR THEN
```

```
                dbms_output.put_line (' operación con el cursor invalida');
                WHEN OTHERS THEN
                                dbms_output.put_line (' error en el procedimiento
EmpleadoConSueldoMayor');
END;
/

/* OTRA FORMA DE HACERLO SIN EL WHILE
BEGIN
        OPEN c_nombre;
        FETCH c_nombre INTO v_nombre;
          dbms_output.put_line (' el nombre del empleado con mayor sueldo es: '||
v_nombre.ename);
        CLOSE c_nombre;
        EXCEPTION
                WHEN INVALID_CURSOR THEN
                dbms_output.put_line (' operación con el cursor invalida');
                WHEN OTHERS THEN
                                dbms_output.put_line (' error en el procedimiento
EmpleadoConSueldoMayor');
END;


OTRA FORMA!!
i NUMBER;
BEGIN
        i=1;
        OPEN c_nombre;
        FETCH c_nombre INTO v_nombre;
        WHILE c_nombre%FOUND AND i=1 LOOP
          dbms_output.put_line (' el nombre del empleado con mayor sueldo es: '||
v_nombre.ename);
        FETCH c_nombre INTO v_nombre;
        i:= i+1;
        END LOOP;
        IF c_nombre%ROWCOUNT <1 THEN
                        dbms_output.put_line('no tiene mas de un empleado ' );
        END IF;
        CLOSE c_nombre;
        EXCEPTION
                WHEN INVALID_CURSOR THEN
                dbms_output.put_line (' operación con el cursor invalida');
                WHEN OTHERS THEN
                                dbms_output.put_line (' error en el procedimiento
EmpleadoConSueldoMayor');
END;
*/
EXECUTE EmpleadoConSueldoMayor ();
```

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 NumeroNombreLocalidadEmple
```

```
IS
CURSOR c_datos IS
SELECT empno, ename, loc FROM emp, dept where emp.deptno=dept.deptno
ORDER BY loc; /*LO HACE CON UN JOIN*/
v_empno emp.empno%TYPE;
v_ename emp.ename%TYPE;
v_loc dept.loc%TYPE;
BEGIN
        OPEN c_datos;
        FETCH c_datos INTO v_empno, v_ename, v_loc;
        WHILE c_datos%FOUND LOOP
            dbms_output.put_line (' número: '|| v_empno|| ' nombre: '|| v_ename||
'localidad: '|| v_loc);
        FETCH c_datos INTO v_empno, v_ename, v_loc;
        END LOOP;
        CLOSE c_datos;
        EXCEPTION
                WHEN INVALID_CURSOR THEN
                dbms_output.put_line (' operación con el cursor invalida');
                WHEN OTHERS THEN
                                dbms_output.put_line (' error en el procedimiento
NumeroNombreLocalidadEmple');
END;
/

EXECUTE NumeroNombreLocalidadEmple ();
```

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

```
CREATE OR REPLACE PROCEDURE IncrementarSalario2
IS
CURSOR c_incremento IS
SELECT sal FROM emp WHERE comm> 0.05*sal FOR UPDATE; /*SE PONE LA
CONDICIÓN PARA ACTUALIZAR*/
v_sal emp.sal%TYPE;
BEGIN
        OPEN c_incremento;
        FETCH c_incremento INTO v_sal;
        WHILE c_incremento%FOUND LOOP
             UPDATE emp SET sal=sal*1.1 WHERE CURRENT OF c_incremento;
/*SE PONE LA CONDICIÓN PARA ACTUALIZAR*/
        FETCH c_incremento INTO v_sal;
        END LOOP;
        CLOSE c_incremento;
        EXCEPTION
                WHEN INVALID_CURSOR THEN
                dbms_output.put_line (' operación con el cursor invalida');
                WHEN OTHERS THEN
                                dbms_output.put_line (' error en el procedimiento
IncrementarSalario2');
END;
```

```
/

EXECUTE IncrementarSalario2 ();
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.

```
CREATE OR REPLACE PROCEDURE IncrementarSalarioDatos
IS
CURSOR c_datos IS
SELECT ename, sal, comm FROM emp WHERE comm> 0.05*sal FOR UPDATE;
v_ename emp.ename%TYPE;
v_sal emp.sal%TYPE;
v_comm emp.comm%TYPE;
v_salnuev emp.sal%TYPE;
BEGIN
        OPEN c_datos;
        FETCH c_datos INTO v_ename, v_sal, v_comm;
        WHILE c_datos%FOUND LOOP
                UPDATE emp SET sal=sal*1.1 WHERE CURRENT OF c_datos;
               SELECT sal INTO v_salnuev FROM emp WHERE ename=v_ename;
/*DESPUÉS DEL UPDATE SE HACE UN SELECT PARA EL CALCULO DEL SALARIO NUEVO*/
         dbms_output.put_line (' nombre: '|| v_ename|| ' salario: '|| v_sal|| 'comision: '|| v_comm);
        dbms_output.put_line (' nombre: '|| v_ename|| ' salario nuevo: '|| v_salnuev|| 'comision: '|| v_comm);
        FETCH c_datos INTO v_ename, v_sal, v_comm;
        END LOOP;
        CLOSE c_datos;
        EXCEPTION
        WHEN INVALID_CURSOR THEN
                dbms_output.put_line (' operación con el cursor invalida');
        WHEN TWO_MANY_ROOWS THEN
                dbms_output.put_line (' demasiadas filas seleccionadas');
        WHEN NOT_DATA_FOUND THEN
                        dbms_output.put_line (' no existe ningún empleado de nombre' ||v_ename);
        WHEN OTHERS THEN
                                dbms_output.put_line (' error en el procedimiento IncrementarSalarioDatos');
END;
/
EXECUTE IncrementarSalarioDatos ();
ROLLBACK;
```

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 CadenaNombreNumero (p_cadena IN VARCHAR2)
```

```
IS
v_cad VARCHAR2 (12);
CURSOR c_cadena IS
SELECT ename, empno FROM emp WHERE ename LIKE v_cad; /*USANDO LIKE*/
v_datos c_cadena%ROWTYPE;
BEGIN
        v_cad:='%'||p_cadena||'%';
        OPEN c_cadena;
        FETCH c_cadena INTO v_datos;
        WHILE c_cadena%FOUND LOOP
        dbms_output.put_line (' nombre: '|| v_datos.ename|| ' numero de empleado: '|| v_datos.empno);
        FETCH c_cadena INTO v_datos;
        END LOOP;
       dbms_output.put_line (' numero de empleados: '|| c_cadena%ROWCOUNT);
        CLOSE c_cadena;
        EXCEPTION
        WHEN INVALID_CURSOR THEN
                dbms_output.put_line (' operación con el cursor invalida');
                WHEN OTHERS THEN
                        dbms_output.put_line (' error en el procedimiento CadenaNombreNumero');
END;
/
/*SE PODRIA HABER PUESTO EN EL LIKE EL PORCENTAJE LIKE ('%'||v_cad||'%')*/
EXECUTE CadenaNombreNumero ('S');
```

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

```
CREATE OR REPLACE PROCEDURE DepartamentosEmpleados2
IS
CURSOR c_depar IS
SELECT deptno, dname FROM dept;
v_deptno dept.deptno%TYPE;
v_dname dept.dname%TYPE;
v_num_emp NUMBER;
BEGIN
OPEN c_depar;
        FETCH c_depar INTO v_deptno, v_dname;
        WHILE c_depar%FOUND LOOP
         v_num_emp:=devolverNumEmpleado (v_deptno); /*LLAMANDO A UNA FUNCIÓN PONIENDO LA VARIABLE*/
       dbms_output.put_line (' el departamento ' ||v_dname|| ' tiene ' ||v_num_emp);
        FETCH c_depar INTO v_deptno, v_dname;
        END LOOP;
        CLOSE c_depar;
        EXCEPTION
        WHEN INVALID_CURSOR THEN
                dbms_output.put_line (' operación con el cursor invalida');
```

```
        WHEN OTHERS THEN
                        dbms_output.put_line (' error en el procedimiento
DepartamentosEmpleados');
END;
/

EXECUTE DepartamentosEmpleados2 ();

CREATE OR REPLACE FUNCTION devolverNumEmpleado(p_deptno
dept.deptno%TYPE)
RETURN NUMBER
IS
v_numemp NUMBER;
BEGIN
        SELECT COUNT (*)  INTO v_numemp
        FROM emp
        WHERE deptno=p_deptno; /*IGUALANDO EL PARAMETRO*/
        RETURN v_numemp;
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.

```
CREATE OR REPLACE PROCEDURE SalarioMasComision
IS
CURSOR c_suma IS
SELECT ename FROM emp WHERE(sal + NVL(comm,0))>2000 and (0 +
NVL(comm,0))=0 FOR UPDATE;
v_ename emp.ename%TYPE;
BEGIN
        OPEN c_suma;
        FETCH c_suma INTO v_ename;
        WHILE c_suma%FOUND LOOP
        UPDATE emp
        SET sal=sal+comm
        WHERE CURRENT OF c_suma;
                dbms_output.put_line (' empleados con salario y comisión mayor a
2000' || v_ename);
        FETCH c_suma INTO v_ename;
        END LOOP;
        CLOSE c_suma;
        EXCEPTION
        WHEN INVALID_CURSOR THEN
                dbms_output.put_line (' operación con el cursor invalida');
        WHEN OTHERS THEN
                        dbms_output.put_line (' error en el procedimiento
SalarioMasComision');
END;
/

EXECUTE SalarioMasComision ();
```

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

```
CREATE OR REPLACE PROCEDURE emp_5maxsal
AS
CURSOR c_emp IS
SELECT apellido, salario FROM emple
ORDER BY salario DESC;
vr_emp c_emp%ROWTYPE;
i NUMBER;
BEGIN
i:=1;
OPEN c_emp;
FETCH c_emp INTO vr_emp;
WHILE c_emp%FOUND AND i<=5 LOOP
DBMS_OUTPUT.PUT_LINE(vr_emp.apellido ||
' * '|| vr_emp.salario);
FETCH c_emp INTO vr_emp;
i:=I+1;
END LOOP;
CLOSE c_emp;
END emp_5maxsal;
```

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

```
CREATE OR REPLACE PROCEDURE emp_2minsal
AS
CURSOR c_emp IS
SELECT apellido, oficio, salario FROM emple
ORDER BY oficio, salario;
vr_emp c_emp%ROWTYPE;
oficio_ant EMPLE.OFICIO%TYPE;
i NUMBER;
BEGIN
OPEN c_emp;
oficio_ant:='*';
FETCH c_emp INTO vr_emp;
WHILE c_emp%FOUND LOOP
IF oficio_ant <> vr_emp.oficio THEN
oficio_ant := vr_emp.oficio;
i := 1;
END IF;
IF i <= 2 THEN
DBMS_OUTPUT.PUT_LINE(vr_emp.oficio||' * '
||vr_emp.apellido||' * '
||vr_emp.salario);
END IF;
FETCH c_emp INTO vr_emp;
i:=I+1;
END LOOP;
CLOSE c_emp;
END emp_2minsal;
```

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.

```
CREATE OR REPLACE PROCEDURE subida_50pct
AS
CURSOR c_ofi_sal IS
SELECT oficio, AVG(salario) salario FROM emple
GROUP BY oficio;
CURSOR c_emp_sal IS
```

```
SELECT oficio, salario FROM emple E1
WHERE salario <
(SELECT AVG(salario) FROM emple E2
WHERE E2.oficio = E1.oficio)
ORDER BY oficio, salario FOR UPDATE OF salario;

vr_ofi_sal c_ofi_sal%ROWTYPE;
vr_emp_sal c_emp_sal%ROWTYPE;
v_incremento emple.salario%TYPE;

BEGIN
COMMIT;
OPEN c_emp_sal;
FETCH c_emp_sal INTO vr_emp_sal;
OPEN c_ofi_sal;
FETCH c_ofi_sal INTO vr_ofi_sal;
WHILE c_ofi_sal%FOUND AND c_emp_sal%FOUND LOOP
/* calcular incremento */
v_incremento :=
(vr_ofi_sal.salario - vr_emp_sal.salario) / 2;
/* actualizar */
UPDATE emple SET salario = salario + v_incremento
WHERE CURRENT OF c_emp_sal;
/* siguiente empleado */
FETCH c_emp_sal INTO vr_emp_sal;
/* comprobar si es otro oficio */
IF c_ofi_sal%FOUND and
vr_ofi_sal.oficio <> vr_emp_sal.oficio THEN
FETCH c_ofi_sal INTO vr_ofi_sal;
END IF;
END LOOP;
CLOSE c_emp_sal;
CLOSE c_ofi_sal;
COMMIT;
EXCEPTION
WHEN OTHERS THEN
ROLLBACK WORK;
RAISE;
END subida_50pct;
```

12. Añadir la columna total2 y en ella escribir la suma del salario y la comisión de los empleados con comisión distinta de 0.

```
ALTER TABLE EMPLE ADD(TOTAL2 NUMBER(10));
DECLARE
CURSOR CURSOR2 IS SELECT COMISION,SALARIO FROM EMPLE
WHERE COMISION <>0
FOR UPDATE;
BEGIN
FOR REG IN CURSOR2 LOOP
UPDATE EMPLE
SET TOTAL2 = SALARIO+COMISION
WHERE CURRENT OF CURSOR2;
END LOOP;
END;
/
```

13. Crear un procedimiento que inserte un empleado en la tabla EMPLE. Su número será el posterior al del empleado de mayor número y la fecha de incorporación a la empresa será la actual.

```
SQL> DECLARE
2 num_emple EMPLE.EMP_NO%TYPE;
3 fecha EMPLE.FECHA_ALT%TYPE;
4 BEGIN
5 SELECT MAX(emp_no) INTO num_emple FROM EMPLE;
6 SELECT SYSDATE INTO fecha FROM DUAL;
7 num_emple := num_emple+1;
8 INSERT INTO EMPLE VALUES (num_emple, 'TARATIEL','EMPLEADO',
7700, fecha, 200000, NULL, 40);
9 END;
PL/SQL procedure successfully completed.
```

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

```
SQL> DECLARE
2 NUM_EMPLE EMPLE.EMP_NO%TYPE;
3 BEGIN
4 SELECT EMP_NO INTO NUM_EMPLE FROM EMPLE
5 WHERE EMP_NO=&numero_empleado;
6 DELETE FROM EMPLE WHERE EMP_NO=NUM_EMPLE;
7 END;
8 / Enter value for numero_empleado:
7935old 5: WHERE EMP_NO=&numero_empleado;new 5: WHERE EMP_NO=7
935;
PL/SQL procedure successfully completed.
```

15. Realizar un procedimiento para modificar la localidad de un departamento. El procedimiento recibe como parámetros la localidad y el número de departamento.

```
SQL> DECLARE
2 v_LOC DEPART.LOC%TYPE;
3 v_DEP DEPART.DEPT_NO%TYPE;
4 BEGIN
5 SELECT LOC, DEPT_NO INTO v_LOC, v_DEP
6 FROM DEPART WHERE DEPT_NO=&NumeroDepartamento;
7 UPDATE DEPART SET LOC='&NuevaLocalidad'
8 WHERE DEPT_NO=v_DEP;
9 END;
10 /
 Enter value for numerodepartamento: 10
old 6: FROM DEPART WHERE DEPT_NO=&NumeroDepartamento;
new 6: FROM DEPART WHERE DEPT_NO=10;
Enter value for nuevalocalidad: Sevilla
old 7: UPDATE DEPART SET LOC='&NuevaLocalidad'
new 7: UPDATE DEPART SET LOC='Sevilla'

PL/SQL procedure successfully completed.
```