

Queremos crear los siguientes subprogramas:

1.- **Procedimiento** que escriba los datos de un empleado a partir del número de empleado, el número de empleado es un parámetro de entrada del procedimiento. Hay que probarlo desde un bloque anónimo y desde otro procedimiento. Si no existe se controla mediante una EXCEPCIÓN

```
SQL Plus (Oracle y PL-SQL)
SQL> CREATE OR REPLACE PROCEDURE d_e (num_emp integer)
2
3 AS
4   d_e_TEMPLESROWTYPE;
5
6 BEGIN
7
8   SELECT * INTO d_e
9   FROM TEMPLE
10  WHERE NUMEMP=num_emp;
11
12  DBMS_OUTPUT.PUT_LINE('NºEmpleado: ' || d_e.NUMEMP || ' ; Departamento en el que trabaja: ' || d_e.DEPTE || ' ; Extensión telefónica: ' || d_e.EXTTEL || ' ; Fecha de nacimiento: ' || d_e.FECNIA);
13  DBMS_OUTPUT.PUT_LINE('Fecha de ingreso: ' || d_e.FECIN || ' ; Salario: ' || d_e.SALAR || ' ; Comisión: ' || d_e.COMIS || ' ; Nº de hijos: ' || d_e.NUMHI || ' ; Nombre del empleado: ' || d_e.NOPEM);
14
15 EXCEPTION
16
17   WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('El empleado no existe');
18
19 END;
20 /
Procedimiento creado.

SQL> CREATE OR REPLACE PROCEDURE prueba_d_e (num_emp integer)
2
3 AS
4
5 BEGIN
6   d_e(num_emp);
7
8 END;
9 /
Procedimiento creado.

SQL> DECLARE
1   num_emp integer;
2
3
4 BEGIN
5   num_emp:=110;
6   prueba_d_e(num_emp);
7
8 END;
9 /
Introduzca un valor para n: 110
antiguo 5: num_emp:=110;
nuevo 5: num_emp:=110;
NºEmpleado: 110 ; Departamento en el que trabaja: 121 ; Extensión telefónica: 350 ; Fecha de nacimiento: 10/11/29
Fecha de ingreso: 15/02/50 ; Salario: 3100 ; Comisión: ; Nº de hijos: 3 ; Nombre del empleado: PONS, CESAR
```

```
SQL Plus (Oracle y PL-SQL)
5
6 BEGIN
7
8   SELECT * INTO d_e
9   FROM TEMPLE
10  WHERE NUMEMP=num_emp;
11
12  DBMS_OUTPUT.PUT_LINE('NºEmpleado: ' || d_e.NUMEMP || ' ; Departamento en el que trabaja: ' || d_e.DEPTE || ' ; Extensión telefónica: ' || d_e.EXTTEL || ' ; Fecha de nacimiento: ' || d_e.FECNIA);
13  DBMS_OUTPUT.PUT_LINE('Fecha de ingreso: ' || d_e.FECIN || ' ; Salario: ' || d_e.SALAR || ' ; Comisión: ' || d_e.COMIS || ' ; Nº de hijos: ' || d_e.NUMHI || ' ; Nombre del empleado: ' || d_e.NOPEM);
14
15 EXCEPTION
16
17   WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('El empleado no existe');
18
19 END;
20 /
Procedimiento creado.

SQL> CREATE OR REPLACE PROCEDURE prueba_d_e (num_emp integer)
2
3 AS
4
5 BEGIN
6   d_e(num_emp);
7
8 END;
9 /
Procedimiento creado.

SQL> DECLARE
1   num_emp integer;
2
3
4 BEGIN
5   num_emp:=110;
6   prueba_d_e(num_emp);
7
8 END;
9 /
Introduzca un valor para n: 110
antiguo 5: num_emp:=110;
nuevo 5: num_emp:=110;
NºEmpleado: 110 ; Departamento en el que trabaja: 121 ; Extensión telefónica: 350 ; Fecha de nacimiento: 10/11/29
Fecha de ingreso: 15/02/50 ; Salario: 3100 ; Comisión: ; Nº de hijos: 3 ; Nombre del empleado: PONS, CESAR
Procedimiento PL/SQL terminado correctamente.

SQL>
```

2.- **Procedimiento** que devuelva, mediante un parámetro de salida, los datos de un departamento a partir del número de departamento, el número de departamento es un parámetro de entrada del procedimiento. Hay que probarlo desde un bloque anónimo y desde otro procedimiento. (Aquí no utilizéis EXCEPCIÓN)

```
SQL Plus (Oracle y PL-SQL)
SQL> CREATE OR REPLACE PROCEDURE d_dep (num_dep IN TDEPTO.NUMDETYPE, datos_dep OUT TDEPTO.DATATYPE)
2
3
4 AS
5 BEGIN
6   SELECT * INTO datos_dep
7   FROM TDEPTO
8   WHERE NUMDE=num_dep;
9
10  DBMS_OUTPUT.PUT_LINE('Nº de departamento: ' || datos_dep.NUMDE || ' ; Nº de centro: ' || datos_dep.NUMCE || ' ; Director del departamento: ' || datos_dep.DIREC);
11  DBMS_OUTPUT.PUT_LINE('En propiedad o funciones?: ' || datos_dep.TIDIR || ' ; Presupuesto: ' || datos_dep.PRESU || ' ; Depende del departamento ' || datos_dep.DEPOC);
12  DBMS_OUTPUT.PUT_LINE('Nombre del departamento: ' || datos_dep.NOMDE);
13
14 END;
15 /
Procedimiento creado.

SQL> CREATE OR REPLACE PROCEDURE prueba_d_dep (num_dep IN integer, datos_dep OUT TDEPTO.DATATYPE)
2
3 AS
4
5 BEGIN
6   d_dep(num_dep, datos_dep);
7
8 END;
9 /
Procedimiento creado.

SQL> DECLARE
2  num_dep integer;
3  datos_dep TDEPTO.DATATYPE;
4
5 BEGIN
6  num_dep:=110;
7  prueba_d_dep(num_dep, datos_dep);
8
9 EXCEPTION
10  WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('El departamento no existe');
11
12 END;
13 /
Introduce un valor para n: 110
antiguo 6: num_dep:=110;
nuevo 6: num_dep:=110;
Nº de departamento: 110 ; Nº de centro: 20 ; Director del departamento: 180
En propiedad o funciones?: P ; Presupuesto: 150 ; Depende del departamento 100
Nombre del departamento: DIRECCION COMERCIAL
```

```
SQL Plus (Oracle y PL-SQL)
5 BEGIN
6  SELECT * INTO datos_dep
7  FROM TDEPTO
8  WHERE NUMDE=num_dep;
9
10  DBMS_OUTPUT.PUT_LINE('Nº de departamento: ' || datos_dep.NUMDE || ' ; Nº de centro: ' || datos_dep.NUMCE || ' ; Director del departamento: ' || datos_dep.DIREC);
11  DBMS_OUTPUT.PUT_LINE('En propiedad o funciones?: ' || datos_dep.TIDIR || ' ; Presupuesto: ' || datos_dep.PRESU || ' ; Depende del departamento ' || datos_dep.DEPOC);
12  DBMS_OUTPUT.PUT_LINE('Nombre del departamento: ' || datos_dep.NOMDE);
13
14 END;
15 /
Procedimiento creado.

SQL> CREATE OR REPLACE PROCEDURE prueba_d_dep (num_dep IN integer, datos_dep OUT TDEPTO.DATATYPE)
2
3 AS
4
5 BEGIN
6   d_dep(num_dep, datos_dep);
7
8 END;
9 /
Procedimiento creado.

SQL> DECLARE
2  num_dep integer;
3  datos_dep TDEPTO.DATATYPE;
4
5 BEGIN
6  num_dep:=110;
7  prueba_d_dep(num_dep, datos_dep);
8
9 EXCEPTION
10  WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('El departamento no existe');
11
12 END;
13 /
Introduce un valor para n: 110
antiguo 6: num_dep:=110;
nuevo 6: num_dep:=110;
Nº de departamento: 110 ; Nº de centro: 20 ; Director del departamento: 180
En propiedad o funciones?: P ; Presupuesto: 150 ; Depende del departamento 100
Nombre del departamento: DIRECCION COMERCIAL

Procedimiento PL/SQL terminado correctamente.
SQL>
```

3.- **Función** que devuelve todos los datos de un departamento a partir del número de departamento, el número de departamento es un parámetro de entrada a la función. Hay que probarlo desde un bloque anónimo y desde otro procedimiento.

```

SQL Plus (Oracle y PL-SQL)
SQL> CREATE OR REPLACE FUNCTION d_dep_F (num_dep IN OUT TDEPTO.NUMDETYPE) RETURN TDEPTO.ROWTYPE
2
3 AS
4 datos_dep TDEPTO.ROWTYPE;
5
6 BEGIN
7 SELECT * INTO datos_dep
8 FROM TDEPTO
9 WHERE NUMDE=num_dep;
10
11 DBMS_OUTPUT.PUT_LINE('Nº de departamento: ' || datos_dep.NUMDE || ' ; Nº de centro: ' || datos_dep.NUMCE || ' ; Director del departamento: ' || datos_dep.DIREC);
12 DBMS_OUTPUT.PUT_LINE('En propiedad o funciones?: ' || datos_dep.TIDIR || ' ; Presupuesto: ' || datos_dep.PRESU || ' ; Depende del departamento ' || datos_dep.DEPOE);
13 DBMS_OUTPUT.PUT_LINE('Nombre del departamento: ' || datos_dep.NOMDE);
14
15 RETURN datos_dep;
16
17 END;
18 /
Función creada.

SQL> CREATE OR REPLACE PROCEDURE prueba_d_dep_F (num_dep IN OUT integer)
2
3 AS
4 function TDEPTO.ROWTYPE;
5
6 BEGIN
7 function:=d_dep_F(num_dep);
8
9 END;
10 /
Procedimiento creado.

SQL> DECLARE
2 num_dep integer;
3
4 BEGIN
5 num_dep:=110;
6 prueba_d_dep_F(num_dep);
7
8 END;
9 /
Introduzca un valor para n: 110
antiguo 5: num_dep:=110;
nuevo 5: num_dep:=110;
Nº de departamento: 110 ; Nº de centro: 20 ; Director del departamento: 180
En propiedad o funciones?: P ; Presupuesto: 150 ; Depende del departamento 100
Nombre del departamento: DIRECCION COMERCIAL

```

```

SQL Plus (Oracle y PL-SQL)
5
6 BEGIN
7 SELECT * INTO datos_dep
8 FROM TDEPTO
9 WHERE NUMDE=num_dep;
10
11 DBMS_OUTPUT.PUT_LINE('Nº de departamento: ' || datos_dep.NUMDE || ' ; Nº de centro: ' || datos_dep.NUMCE || ' ; Director del departamento: ' || datos_dep.DIREC);
12 DBMS_OUTPUT.PUT_LINE('En propiedad o funciones?: ' || datos_dep.TIDIR || ' ; Presupuesto: ' || datos_dep.PRESU || ' ; Depende del departamento ' || datos_dep.DEPOE);
13 DBMS_OUTPUT.PUT_LINE('Nombre del departamento: ' || datos_dep.NOMDE);
14
15 RETURN datos_dep;
16
17 END;
18 /
Función creada.

SQL> CREATE OR REPLACE PROCEDURE prueba_d_dep_F (num_dep IN OUT integer)
2
3 AS
4 function TDEPTO.ROWTYPE;
5
6 BEGIN
7 function:=d_dep_F(num_dep);
8
9 END;
10 /
Procedimiento creado.

SQL> DECLARE
2 num_dep integer;
3
4 BEGIN
5 num_dep:=110;
6 prueba_d_dep_F(num_dep);
7
8 END;
9 /
Introduzca un valor para n: 110
antiguo 5: num_dep:=110;
nuevo 5: num_dep:=110;
Nº de departamento: 110 ; Nº de centro: 20 ; Director del departamento: 180
En propiedad o funciones?: P ; Presupuesto: 150 ; Depende del departamento 100
Nombre del departamento: DIRECCION COMERCIAL
Procedimiento PL/SQL terminado correctamente.
SQL>

```

4.- **Procedimiento** que muestra un listado de empleados (a partir de su número de departamento), de la siguiente manera:

Nº Empleado	Nombre_empl	Salario	Nombre-Dpto	Nombre_centro
.....
.....

Hay que probarlo desde un bloque anónimo y desde otro procedimiento.

```
SQL Plus (Oracle y PL-SQL)
SQL> CREATE OR REPLACE PROCEDURE Listado_Empleados (num_dep integer)
2
3
4 AS
5     CURSOR empleados (num_dep integer) IS
6     SELECT TEMPLE.Nombre, TEMPLE.Nomem, TEMPLE.Salar, TDEPTO.Nombre, TCENTR.Nombre
7     FROM TEMPLE, TDEPTO, TCENTR
8     WHERE TEMPLE.NUMDE=TDEPTO.NUMDE AND TDEPTO.NUMKE=TCENTR.NUMKE AND Tdepto.Numde=num_dep;
9
10    n_emp integer;
11    nom_emp varchar2(30);
12    salario integer;
13    nom_dep varchar2(30);
14    nom_centro varchar2(30);
15    N_Empleado varchar2(20);
16    Name_Emp varchar2(30);
17    Sal varchar2(200);
18    Name_Dpto varchar2(30);
19    Name_Centro varchar2(30);
20    Espacio_blanco varchar2(20);
21
22 BEGIN
23     Espacio_blanco:= ' ';
24     N_Empleado:='NºEmpleado';
25     Name_Emp:='Nombre_empl';
26     Sal:='Salario';
27     Name_Dpto:='Nombre-Dpto';
28     Name_Centro:='Nombre_centro';
29
30     OPEN empleados(num_dep);
31     FETCH empleados INTO n_emp, nom_emp, salario, nom_dep, nom_centro;
32     DBMS_OUTPUT.PUT_LINE(N_Empleado || RPAD(Espacio_blanco, 10) || Name_Emp || RPAD(Espacio_blanco, 10) || Sal || RPAD(Espacio_blanco, 10) || Name_Dpto || RPAD(Espacio_blanco, 10) || Name_Centro);
33
34     WHILE empleados%FOUND LOOP
35         DBMS_OUTPUT.PUT_LINE(CHR(10));
36         DBMS_OUTPUT.PUT_LINE(n_emp || RPAD(Espacio_blanco, 17) || nom_emp || RPAD(Espacio_blanco, 10) || salario || RPAD(Espacio_blanco, 13) || nom_dep || RPAD(Espacio_blanco, 9) || nom_centro);
37         FETCH empleados INTO n_emp, nom_emp, salario, nom_dep, nom_centro;
38     END LOOP;
39
40     CLOSE empleados;
41
42 END;
43 /
Procedimiento creado.
SQL>
```

```
SQL Plus (Oracle y PL-SQL)
SQL> CREATE OR REPLACE PROCEDURE prueba_listado_empleados (num_dep integer)
2
3
4 AS
5 BEGIN
6     listado_empleados(num_dep);
7
8 END;
9 /
Procedimiento creado.
SQL> DECLARE
2     num_dep integer;
3
4 BEGIN
5     num_dep:='&num_dep';
6     prueba_listado_empleados(num_dep);
7
8 END;
9 /
Introduzca un valor para num_dep: 112
antiguo 5: num_dep:=&num_dep;
nuevo 5: num_dep:=112;
NºEmpleado Nombre_empl Salario Nombre-Dpto Nombre_centro
120 LASA, MARIO 3500 SECTOR SERVICIOS RELACION CON CLIENTES
130 TEROL, LUCIANO 2900 SECTOR SERVICIOS RELACION CON CLIENTES
270 GARCIA, OCTAVIO 3800 SECTOR SERVICIOS RELACION CON CLIENTES
330 DIEZ, AMELIA 2800 SECTOR SERVICIOS RELACION CON CLIENTES
180 MARTIN, MICHAELA 1800 SECTOR SERVICIOS RELACION CON CLIENTES
450 PEREZ, SABINA 2100 SECTOR SERVICIOS RELACION CON CLIENTES
490 TORRES, HORACIO 1800 SECTOR SERVICIOS RELACION CON CLIENTES
Procedimiento PL/SQL terminado correctamente.
```

5. - **Función** que compruebe si existe un departamento con el número que se le pase (parámetro de entrada). Devolverá verdadero o falso, 1 ó 0 lo que creáis conveniente.

```
SQL Plus (Oracle y PL-SQL)
SQL> CREATE OR REPLACE FUNCTION existe_departamento (num IN integer) RETURN integer
2
3 AS
4   dep INTEGER;
5
6 BEGIN
7
8   SELECT MUNDE INTO dep
9   FROM TDEPTO
10  WHERE MUNDE=num;
11
12  DBMS_OUTPUT.PUT_LINE('El departamento existe');
13  RETURN 1;
14
15 END;
16 /
Función creada.

SQL> DECLARE
2   num INTEGER;
3   fun integer;
4
5 BEGIN
6   num:='&n';
7   fun:=existe_departamento(num);
8
9 EXCEPTION
10
11 WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('El departamento no existe');
12
13 END;
14 /
Introduzca un valor para n: 135
antiguo 6: num:='&n';
nuevo 6: num:='135';
El departamento no existe

Procedimiento PL/SQL terminado correctamente.

SQL> DECLARE
2   num INTEGER;
3   fun integer;
4
5 BEGIN
6   num:='&n';
7   fun:=existe_departamento(num);
8
9 EXCEPTION
```

```
SQL Plus (Oracle y PL-SQL)
14
15 END;
16 /
Función creada.

SQL> DECLARE
2   num INTEGER;
3   fun integer;
4
5 BEGIN
6   num:='&n';
7   fun:=existe_departamento(num);
8
9 EXCEPTION
10
11 WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('El departamento no existe');
12
13 END;
14 /
Introduzca un valor para n: 135
antiguo 6: num:='&n';
nuevo 6: num:='135';
El departamento no existe

Procedimiento PL/SQL terminado correctamente.

SQL> DECLARE
2   num INTEGER;
3   fun integer;
4
5 BEGIN
6   num:='&n';
7   fun:=existe_departamento(num);
8
9 EXCEPTION
10
11 WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('El departamento no existe');
12
13 END;
14 /
Introduzca un valor para n: 110
antiguo 6: num:='&n';
nuevo 6: num:='110';
El departamento existe

Procedimiento PL/SQL terminado correctamente.

SQL>
```

6.- **Función** que compruebe si existe un empleado con el número que se le pase (parámetro de entrada). Devolverá verdadero o falso, 1 ó 0 lo que creáis conveniente.

```
SQL Plus (Oracle y PL-SQL)
SQL> CREATE OR REPLACE FUNCTION existe_empleado (num IN integer) RETURN boolean
2
3 AS
4   dep INTEGER;
5
6 BEGIN
7
8   SELECT NUMEN INTO dep
9   FROM TEMPLE
10  WHERE NUMEN=num;
11
12  RETURN true;
13
14 EXCEPTION
15
16   WHEN NO_DATA_FOUND THEN RETURN false;
17
18 END;
19 /
función creada.
SQL> DECLARE
2   num INTEGER;
3
4 BEGIN
5   num:='&n';
6
7   IF existe_empleado(num) THEN
8     DBMS_OUTPUT.PUT_LINE('El empleado ' || num || ' existe');
9   ELSE
10    DBMS_OUTPUT.PUT_LINE('El empleado ' || num || ' no existe');
11  END IF;
12
13 END;
14 /
Introduzca un valor para n: 555
antiguo 5: num:='&n';
nuevo 5: num:='555';
El empleado 555 no existe
Procedimiento PL/SQL terminado correctamente.
SQL> DECLARE
2   num INTEGER;
3
4 BEGIN
5   num:='&n';
6
```

```
SQL Plus (Oracle y PL-SQL)
SQL> DECLARE
2   num INTEGER;
3
4 BEGIN
5   num:='&n';
6
7   IF existe_empleado(num) THEN
8     DBMS_OUTPUT.PUT_LINE('El empleado ' || num || ' existe');
9   ELSE
10    DBMS_OUTPUT.PUT_LINE('El empleado ' || num || ' no existe');
11  END IF;
12
13 END;
14 /
Introduzca un valor para n: 110
antiguo 5: num:='&n';
nuevo 5: num:='110';
El empleado 110 existe
Procedimiento PL/SQL terminado correctamente.
SQL> _
```

7. **Procedimiento o bloque anónimo** que a partir de un número de empleado, si existe, nos muestre todos los datos del empleado y de su departamento utilizando los subprogramas anteriores.

```
SQL Plus (Oracle y PL-SQL)
SQL> DECLARE
  2 num_emp integer;
  3 num_dep integer;
  4 datos_dep TDEPTCOLORMYPE;
  5
  6 BEGIN
  7   num_emp:='&n';
  8
  9   SELECT NMDE INTO num_dep
 10   FROM TEMPLE
 11   WHERE NMDEH=num_emp;
 12
 13   prueba_d_s(num_emp);
 14   DBMS_OUTPUT.PUT_LINE('-----');
 15   prueba_d_dep(num_dep, datos_dep);
 16
 17 EXCEPTION
 18   WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('El empleado no existe');
 19
 20 END;
 21 /
Introduzca un valor para n: 110
antiguo 7: num_emp:='&n';
nuevo 7: num_emp:='110';
Empleado: 110 ; Departamento en el que trabaja: 121 ; Extensión telefónica: 350 ; Fecha de nacimiento: 10/11/29
Fecha de ingreso: 15/02/50 ; Salario: 3100 ; Comisión: ; Nº de hijos: 3 ; Nombre del empleado: PONS, CESAR
-----
Nº de departamento: 121 ; Nº de centro: 10 ; Director del departamento: 150
¿Ha propiedad o funciones?: P ; Presupuesto: 20 ; Depende del departamento 120
Nombre del departamento: PERSONAL
Procedimiento PL/SQL terminado correctamente.
SQL>
```

```
SQL Plus (Oracle y PL-SQL)
SQL> DECLARE
  2 num_emp integer;
  3 num_dep integer;
  4 datos_dep TDEPTCOLORMYPE;
  5
  6 BEGIN
  7   num_emp:='&n';
  8
  9   SELECT NMDE INTO num_dep
 10   FROM TEMPLE
 11   WHERE NMDEH=num_emp;
 12
 13   prueba_d_s(num_emp);
 14   DBMS_OUTPUT.PUT_LINE('-----');
 15   prueba_d_dep(num_dep, datos_dep);
 16
 17 EXCEPTION
 18   WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('El empleado no existe');
 19
 20 END;
 21 /
Introduzca un valor para n: 10
antiguo 7: num_emp:='&n';
nuevo 7: num_emp:='10';
El empleado no existe
Procedimiento PL/SQL terminado correctamente.
SQL>
```

8.- **Procedimiento** al que le pasamos un nombre de empleado como parámetro de entrada y debe obtener, mediante una función que también hay que crear, el departamento de ese empleado. Una vez tengamos el departamento el procedimiento hace un listado de los empleados de ese departamento utilizando otro procedimiento que también hay que crear (se le pasa el número de departamento), aumenta el salario de los empleados de ese departamento mediante otro procedimiento (también se le pasa el número de departamento) y se vuelven a escribir los empleados para comprobar que se ha cambiado el salario. Hay que probarlo desde un bloque anónimo y desde otro procedimiento.

```
SQL Plus (Oracle y PL-SQL)
SQL> CREATE OR REPLACE PROCEDURE lep (nom_emp varchar2)
2
3 AS
4 f integer;
5
6 BEGIN
7 f:=dep_emp(nom_emp);
8 list_emp_dep(f);
9 END;
10 /

Procedimiento creado.

SQL> CREATE OR REPLACE FUNCTION dep_emp (nom_emp varchar2) RETURN integer
2
3 AS
4 CURSOR dep(nom_emp varchar2) IS
5 SELECT Nombres, Nume
6 FROM Temple
7 WHERE Nombres=nom_emp;
8
9 num_dep integer;
10 nom_em varchar2(20);
11
12 BEGIN
13 OPEN dep(nom_emp);
14 FETCH dep INTO nom_em, num_dep;
15
16 WHILE dep%FOUND LOOP
17 DBMS_OUTPUT.PUT_LINE(CHR(10));
18 DBMS_OUTPUT.PUT_LINE('Departamento del empleado ' || nom_emp || ': ' || num_dep);
19 DBMS_OUTPUT.PUT_LINE(CHR(10));
20 FETCH dep INTO nom_em, num_dep;
21 END LOOP;
22 CLOSE dep;
23
24 list_emp_dep(num_dep);
25
26 RETURN 1;
27
28 END;
29 /

Función creada.

SQL> CREATE OR REPLACE PROCEDURE list_emp_dep (num_dep integer)
2
3 AS
4 CURSOR listado_dep_emp(num_dep integer) IS
```

```
SQL Plus (Oracle y PL-SQL)
SQL> CREATE OR REPLACE PROCEDURE list_emp_dep (num_dep integer)
2
3 AS
4 CURSOR listado_dep_emp(num_dep integer) IS
5 SELECT *
6 FROM TEMPLE
7 WHERE Nume=num_dep;
8
9 datos TEMPLE%ROWTYPE;
10
11 BEGIN
12 OPEN listado_dep_emp(num_dep);
13 FETCH listado_dep_emp INTO datos;
14
15 WHILE listado_dep_emp%FOUND LOOP
16 DBMS_OUTPUT.PUT_LINE('NºEmpleado: ' || datos.numem || ' Extensión Telefónica: ' || datos.extel || ' Fecha de Nacimiento: ' || datos.fecna || ' Fecha de Ingreso: ' || datos.fecin || ' Salario: ' ||
17 datos.salar || ' Comisión: ' || datos.comis || ' Nºde Hijos: ' || datos.numhi || ' Nombre del empleado: ' || datos.nombres);
18
19 FETCH listado_dep_emp INTO datos;
20 END LOOP;
21 CLOSE listado_dep_emp;
22
23 DBMS_OUTPUT.PUT_LINE(CHR(10));
24 aumentar_salario(num_dep);
25
26 END;
27 /

Procedimiento creado.

SQL> CREATE OR REPLACE PROCEDURE aumentar_salario (num_dep integer)
2
3 AS
4 CURSOR salario_nuevo_empleados(num_dep integer) IS
5 SELECT Nombres, Salar, Nombres
6 FROM TEMPLE
7 WHERE Nume=num_dep;
8
9 salario_nuevo integer;
10 N_Emp integer;
11 Name_Emp varchar2(20);
12
13 BEGIN
14
15 UPDATE TEMPLE
16 SET SALAR = SALAR + 200
17 WHERE Nume=num_dep;
18
19 OPEN salario_nuevo_empleados(num_dep);
20 FETCH salario_nuevo_empleados INTO N_Emp, salario_nuevo, Name_Emp;
```



```

SQL Plus (Oracle y PL-SQL)
SQL> CREATE OR REPLACE PROCEDURE aumentar_salario (num_dep integer)
2
3 AS
4 CURSOR salario_nuevo_empleados(num_dep integer) IS
5 SELECT Nombres, Salario, Nomem
6 FROM TEMPLE
7 WHERE Numde=num_dep;
8
9 salario_nuevo integer;
10 N_Emp integer;
11 Name_Emp varchar2(20);
12
13 BEGIN
14
15 UPDATE TEMPLE
16 SET SALAR = SALAR + 200
17 WHERE Numde=num_dep;
18
19 OPEN salario_nuevo_empleados(num_dep);
20 FETCH salario_nuevo_empleados INTO N_Emp, salario_nuevo, Name_Emp;
21
22 IF num_dep <> 1 THEN
23 DBMS_OUTPUT.PUT_LINE('Empleados con los salarios actualizados del departamento ' || num_dep || ' ': ' || CHR(10));
24 END IF;
25
26 WHILE salario_nuevo_empleados%FOUND LOOP
27 DBMS_OUTPUT.PUT_LINE('NºEmpleado: ' || N_Emp || ' Salario nuevo: ' || salario_nuevo || ' Nombre del empleado: ' || Name_Emp);
28 FETCH salario_nuevo_empleados INTO N_Emp, salario_nuevo, Name_Emp;
29 END LOOP;
30
31 END;
32 /
Procedimiento creado.

SQL> CREATE OR REPLACE PROCEDURE prueba_LEP (nom_emp varchar2)
2
3 AS
4
5 BEGIN
6 lep(nom_emp);
7
8 END;
9 /
Procedimiento creado.

SQL> DECLARE
2 nom_emp varchar2(20);

```

```

SQL Plus (Oracle y PL-SQL)
SQL> DECLARE
2 nom_emp varchar2(20);
3
4 BEGIN
5 nom_emp:='Empleado';
6 prueba_LEP(nom_emp);
7
8 EXCEPTION
9 WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('No se ha encontrado al empleado que buscabas');
10
11 END;
12 /
Introduzca un valor para empleado: PONS, CESAR
Antiguo S: nom_emp:='Empleado';
nuevo S: nom_emp:='PONS, CESAR';

Departamento del empleado PONS, CESAR: 121

NºEmpleado: 110 Extensión Telefónica: 350 Fecha de Nacimiento: 10/11/29 Fecha de Ingreso: 15/02/59 Salario: 4700 Comisión: 0 Nºde Hijos: 3 Nombre del empleado: PONS, CESAR
NºEmpleado: 150 Extensión Telefónica: 340 Fecha de Nacimiento: 10/08/30 Fecha de Ingreso: 15/01/48 Salario: 6000 Comisión: 0 Nºde Hijos: 0 Nombre del empleado: PEREZ, JULIO
NºEmpleado: 190 Extensión Telefónica: 350 Fecha de Nacimiento: 12/05/32 Fecha de Ingreso: 11/02/62 Salario: 4600 Comisión: 0 Nºde Hijos: 4 Nombre del empleado: VEIGA, JULIANA
NºEmpleado: 370 Extensión Telefónica: 360 Fecha de Nacimiento: 22/06/67 Fecha de Ingreso: 20/01/87 Salario: 3500 Comisión: 0 Nºde Hijos: 1 Nombre del empleado: RUIZ, FABIOLA

Empleados con los salarios actualizados del departamento 121:
NºEmpleado: 110 Salario nuevo: 4900 Nombre del empleado: PONS, CESAR
NºEmpleado: 150 Salario nuevo: 6200 Nombre del empleado: PEREZ, JULIO
NºEmpleado: 190 Salario nuevo: 4800 Nombre del empleado: VEIGA, JULIANA
NºEmpleado: 370 Salario nuevo: 3700 Nombre del empleado: RUIZ, FABIOLA

Procedimiento PL/SQL terminado correctamente.
SQL>

```

9.- **Función** que calcule el factorial de un número que se le pasa como parámetro. Se prueba desde un procedimiento y desde un bloque anónimo que utilizarán una variable de sustitución para pedir el número.

```
SQL Plus (Oracle y PL-SQL)
SQL> CREATE OR REPLACE FUNCTION Factorial (num IN integer) RETURN integer
1
2
3 AS
4     factorial integer;
5
6 BEGIN
7
8     factorial:=1;
9     FOR contador IN 1..num LOOP
10        factorial:=factorial*contador;
11     END LOOP;
12
13 DBMS_OUTPUT.PUT_LINE('El factorial del número ' || num || ' es ' || factorial);
14 RETURN 1;
15
16 END;
17 /
Función creada.

SQL> CREATE OR REPLACE PROCEDURE Factory (num integer)
1
2
3 AS
4     factory integer;
5
6 BEGIN
7     factory:=factorial(num);
8
9 END;
10 /
Procedimiento creado.

SQL> DECLARE
1     num integer;
2
3 BEGIN
4     num:=&n;
5     Factory(num);
6
7 END;
8 /
Introduzca un valor para n: 5
antiguo   5:  num:=5;
nuevo     5:  num:=5;
El factorial del número 5 es 120

Procedimiento PL/SQL terminado correctamente.
```