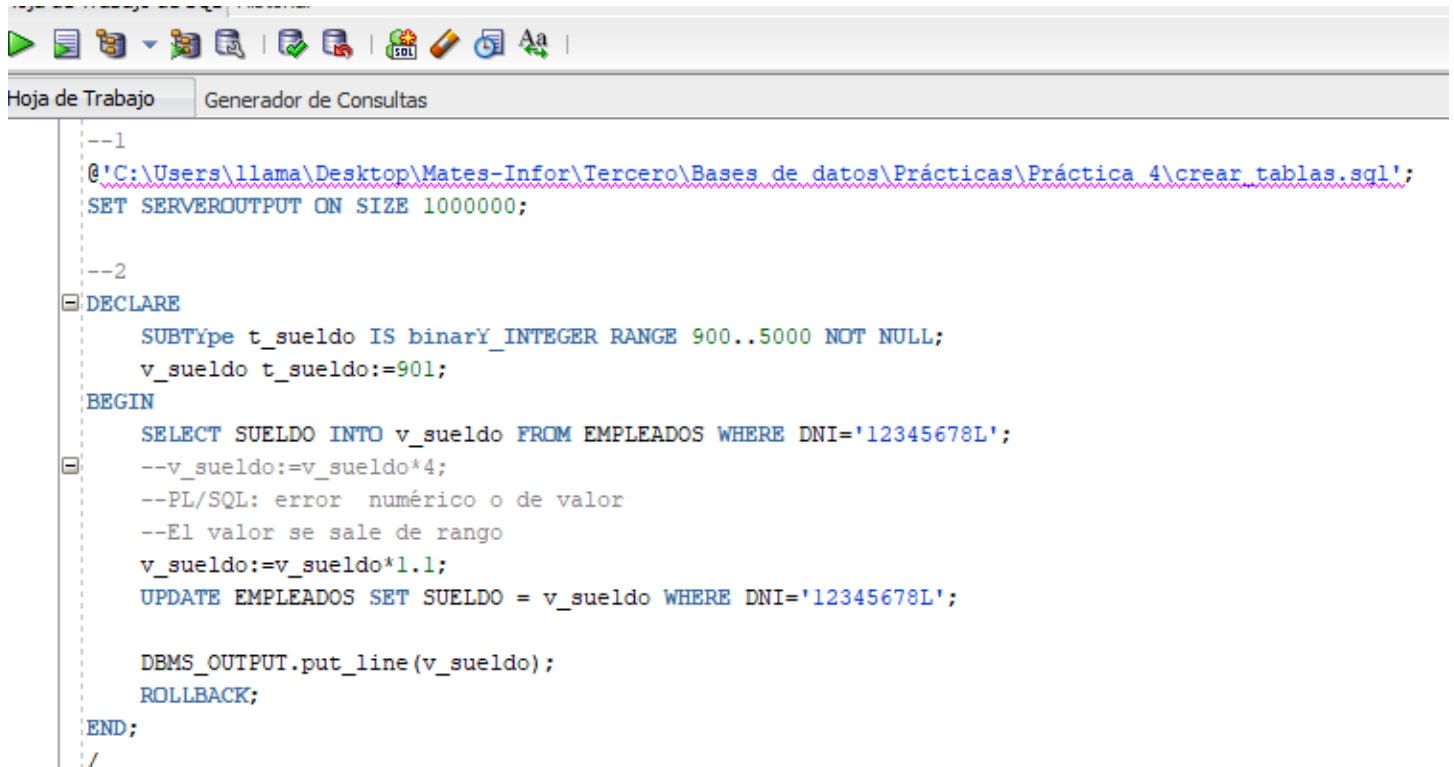


PRÁCTICA 4

JUAN CARLOS LLAMAS NÚÑEZ 3ºDG

En este documento se presentan las capturas del código para facilitar la legibilidad, así como la salida por consola que generan las consultas. Al final del documento se añade el texto plano del archivo .sql y la salida de consola.

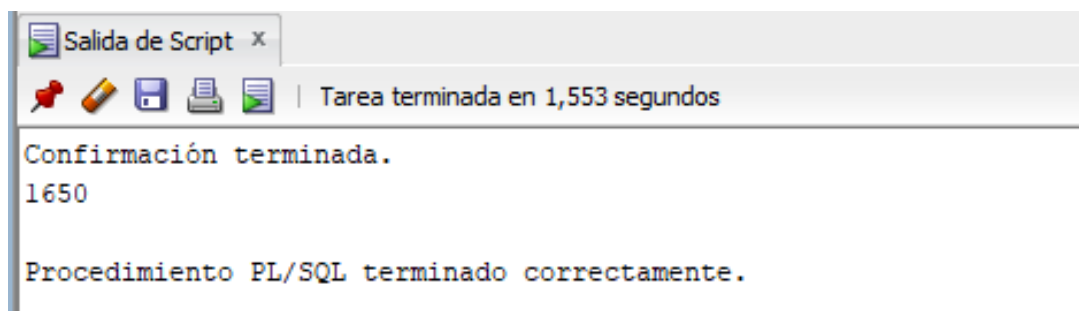
1 y 2



```
--1
@'C:\Users\llama\Desktop\Mates-Infor\Tercero\Bases de datos\Prácticas\Práctica 4\crear tablas.sql';
SET SERVEROUTPUT ON SIZE 1000000;

--2
DECLARE
    SUBTípe t_sueldo IS binary_INTEGER RANGE 900..5000 NOT NULL;
    v_sueldo t_sueldo:=901;
BEGIN
    SELECT SUELDO INTO v_sueldo FROM EMPLEADOS WHERE DNI='12345678L';
    --v_sueldo:=v_sueldo*4;
    --PL/SQL: error numérico o de valor
    --El valor se sale de rango
    v_sueldo:=v_sueldo*1.1;
    UPDATE EMPLEADOS SET SUELDO = v_sueldo WHERE DNI='12345678L';

    DBMS_OUTPUT.put_line(v_sueldo);
    ROLLBACK;
END;
/
```



```
Salida de Script x
Tarea terminada en 1,553 segundos

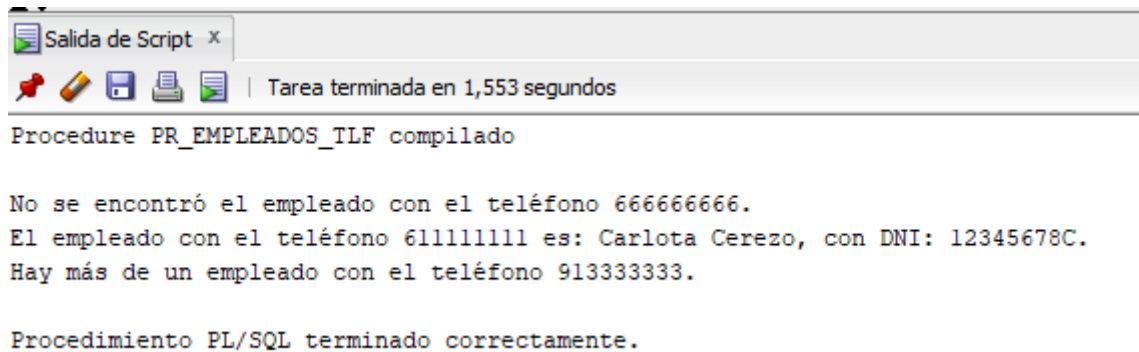
Confirmación terminada.
1650

Procedimiento PL/SQL terminado correctamente.
```

3

```
--3
CREATE OR REPLACE PROCEDURE pr_empleados_tlf
(p_tlf IN VARCHAR2)
IS
    v_dni EMPLEADOS.DNI%TYPE;
    v_nombre EMPLEADOS.NOMBRE%TYPE;
BEGIN
    SELECT DNI, NOMBRE
    INTO v_dni, v_nombre
    FROM EMPLEADOS NATURAL JOIN TELÉFONOS
    WHERE TELÉFONO=p_tlf;

    DBMS_OUTPUT.put_line('El empleado con el teléfono '||p_tlf||' es: '||v_nombre||', con DNI: '||v_dni||');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.put_line('No se encontró el empleado con el teléfono '||p_tlf||');
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.put_line('Hay más de un empleado con el teléfono '||p_tlf||');
END;
/
DECLARE
BEGIN
    pr_empleados_tlf('666666666');
    pr_empleados_tlf('611111111');
    pr_empleados_tlf('913333333');
END;
/
```



4

```
--4
CREATE OR REPLACE PROCEDURE pr_comprobar_poblaciones
IS
    v_poblacion "Códigos postales".POBLACIÓN%TYPE;
BEGIN
    SELECT POBLACIÓN INTO v_poblacion FROM "Códigos postales" GROUP BY POBLACIÓN HAVING COUNT(*)>1;
    DBMS_OUTPUT.put_line('A la población '||v_poblacion||' no le corresponde siempre la misma provincia.');
```

EXCEPTION

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.put_line('No hay dos o más provincias que compartan la misma población.');

END;

/

DECLARE

BEGIN

pr_comprobar_poblaciones();

INSERT INTO "Códigos postales" VALUES ('41008','Arganda','Sevilla');

pr_comprobar_poblaciones();

ROLLBACK;

END;

/



Procedure PR_COMPROBAR_POBLACIONES compilado

No hay dos o más provincias que compartan la misma población.

A la población Arganda no le corresponde siempre la misma provincia.

Procedimiento PL/SQL terminado correctamente.

5

```
--5
CREATE OR REPLACE PROCEDURE pr_empleados_CP
IS
    CURSOR c_cod_postal IS SELECT "Código postal", AVG(SUELDO) FROM
        ("Códigos postales" NATURAL JOIN DOMICILIOS) NATURAL JOIN EMPLEADOS
    GROUP BY "Código postal" ORDER BY "Código postal";

    CURSOR c_dni (cod_post DOMICILIOS."Código postal"%TYPE) IS
    SELECT NOMBRE, CALLE, SUELDO FROM DOMICILIOS NATURAL JOIN EMPLEADOS
    WHERE "Código postal"=cod_post;

    TYPE codigo_postal IS RECORD(
        cod_postal "Códigos postales"."Código postal"%TYPE,
        sueldo_medio NUMBER
    );
    TYPE persona IS RECORD(
        nombre EMPLEADOS.NOMBRE%TYPE,
        calle DOMICILIOS.CALLE%TYPE,
        sueldo EMPLEADOS.SUELDO%TYPE
    );
    v_cod codigo_postal;
    v_persona persona;
    i INT:=0;
    j INT:=0;
BEGIN
    OPEN c_cod_postal;
    LOOP
        FETCH c_cod_postal INTO v_cod;
        EXIT WHEN c_cod_postal%NOTFOUND;

        OPEN c_dni(v_cod.cod_postal);
        LOOP
            FETCH c_dni INTO v_persona;
            EXIT WHEN c_dni%NOTFOUND;
            --Al menos un empleado vive en el domicilio asociado a ese código postal
            i:=i+1;
            IF i=1 THEN
                DBMS_OUTPUT.put_line('Código postal: '||v_cod.cod_postal);
            END IF;
            DBMS_OUTPUT.put_line(' '||v_persona.nombre||', '||v_persona.calle||', '||v_persona.sueldo);

            j:=j+1;
        END LOOP;
        CLOSE c_dni;
        IF i>=1 THEN
            DBMS_OUTPUT.put_line('    N° empleados: ' || i ||', Sueldo medio: '||v_cod.sueldo_medio);
        END IF;
        i:=0;
    END LOOP;
    CLOSE c_cod_postal;
    DBMS_OUTPUT.put_line('Total empleados: '||j);
END;
/
DECLARE
BEGIN
    pr_empleados_CP();
END;
```

```
Salida de Script x
Tarea terminada en 1,553 segundos

Procedure PR_EMPLEADOS_CP compilado

Código postal: 14200
  Laura López, Diamante, 1500
  Pedro Pérez, Diamante, 2000
  N° empleados: 2, Sueldo medio: 1750
Código postal: 14900
  Pedro Pérez, Carbón, 2000
  N° empleados: 1, Sueldo medio: 2000
Código postal: 28004
  Antonio Arjona, Cántaro, 5000
  N° empleados: 1, Sueldo medio: 5000
Código postal: 28040
  Antonio Arjona, Avda. Complutense, 5000
  N° empleados: 1, Sueldo medio: 5000
Total empleados: 5

Procedimiento PL/SQL terminado correctamente.
```

6

```
--6
CREATE OR REPLACE FUNCTION f_c_ba RETURN NUMBER
IS
BEGIN
    RETURN 2.920050977316;
END;
/
CREATE OR REPLACE FUNCTION f_n (p_n INTEGER) RETURN NUMBER
IS
    devolver NUMBER;
    fn NUMBER;
BEGIN
    IF p_n=1 THEN
        devolver:=f_c_ba();
    ELSE
        fn:=f_n(p_n-1);
        devolver:=FLOOR(fn) * (fn-FLOOR(fn)+1);
    END IF;
    RETURN devolver;
END;
/
CREATE OR REPLACE FUNCTION f_primo (p_n INTEGER) RETURN NUMBER
IS
BEGIN
    return FLOOR(f_n(p_n));
END;
/
DECLARE
BEGIN
    FOR i IN 1..10
    LOOP
        DBMS_OUTPUT.put_line(i||'.- '||f_primo(i));
    END LOOP;
END;
/
```



Function F_C_BA compilado

Function F_N compilado

Function F_PRIMO compilado

```
1.- 2
2.- 3
3.- 5
4.- 7
5.- 11
6.- 13
7.- 17
8.- 19
9.- 23
10.- 29
```

Procedimiento PL/SQL terminado correctamente.

7

```
--7
CREATE OR REPLACE PROCEDURE pr_jefes (p_DNI IN VARCHAR2)
IS
    v_dni EMPLEADOS.JEFE%TYPE;
    v_nombre EMPLEADOS.NOMBRE%TYPE;
BEGIN
    SELECT JEFE INTO v_dni FROM EMPLEADOS WHERE DNI=p_DNI;
    IF v_dni IS NOT NULL THEN
        SELECT NOMBRE INTO v_nombre FROM EMPLEADOS WHERE DNI=v_dni;
        DBMS_OUTPUT.put_line(v_nombre||' '||v_dni);
        pr_jefes(v_dni);
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.put_line('El empleado con DNI '||p_DNI||' no existe');
END;
/

DECLARE
BEGIN
    pr_jefes('12345678C');
    pr_jefes('12345688C');
END;
/

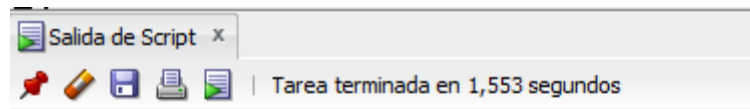
--Si ponemos union en lugar de UNION ALL lanza esta excepción:
--PL/SQL: ORA-32040: la cláusula recursiva WITH debe utilizar una operación UNION ALL
CREATE OR REPLACE PROCEDURE pr_jefes_nr(p_DNI IN VARCHAR2)
IS
    CURSOR c_jefes IS
    WITH tiene_jefe (empleado, jefe) AS (
        SELECT dni, jefe FROM empleados WHERE dni = p_dni
        UNION ALL
        SELECT tiene_jefe.empleado, empleados.jefe FROM tiene_jefe INNER JOIN empleados ON tiene_jefe.jefe = empleados.dni
        WHERE empleados.jefe IS NOT NULL
    )
    SELECT empleado, jefe FROM tiene_jefe;

    v_dni empleados.jefe%TYPE;
    v_aux empleados.jefe%TYPE;
    v_nombre empleados.nombre%TYPE;

BEGIN
    OPEN c_jefes;
    LOOP
        FETCH c_jefes INTO v_aux,v_dni;
        EXIT WHEN c_jefes%notfound;
        SELECT nombre INTO v_nombre FROM empleados WHERE dni = v_dni;
        dbms_output.put_line(v_nombre||' '||v_dni);
    END LOOP;

    CLOSE c_jefes;
    --Procedimiento artificial para comprobar que el dni que se proporciona está en la tabla EMPLEADOS
    --Si no lo estuviera el SELECT INTO lanza una excepción y se ejecuta el bloque EXCEPTION
    SELECT jefe INTO v_aux FROM empleados WHERE b dni = p_dni;
EXCEPTION
    WHEN no_data_found THEN
        dbms_output.put_line('El empleado con DNI '||p_dni||' no existe');
END;
/

DECLARE BEGIN
    pr_jefes_nr('12345678C');
    pr_jefes_nr('12345688C');
END;
```



```
Procedure PR_JEFES compilado

Laura López 12345678L
Pedro Pérez 12345678P
Antonio Arjona 12345678A
El empleado con DNI 12345688C no existe

Procedimiento PL/SQL terminado correctamente.

Procedure PR_JEFES_NR compilado

Laura López 12345678L
Pedro Pérez 12345678P
Antonio Arjona 12345678A
El empleado con DNI 12345688C no existe

Procedimiento PL/SQL terminado correctamente.
```

ARCHIVO .SQL EN TEXTO PLANO

```
--1
@'C:\Users\llama\Desktop\Mates-Infor\Tercero\Bases de datos\Prácticas\Práctica
4\crear_tablas.sql';
SET SERVEROUTPUT ON SIZE 1000000;

--2
DECLARE
  SUBTYpe t_sueldo IS binary_INTEGER RANGE 900..5000 NOT NULL;
  v_sueldo t_sueldo:=901;
BEGIN
  SELECT SUELDO INTO v_sueldo FROM EMPLEADOS WHERE DNI='12345678L';
  --v_sueldo:=v_sueldo*4;
  --PL/SQL: error numérico o de valor
  --El valor se sale de rango
  v_sueldo:=v_sueldo*1.1;
  UPDATE EMPLEADOS SET SUELDO = v_sueldo WHERE DNI='12345678L';

  DBMS_OUTPUT.put_line(v_sueldo);
  ROLLBACK;
END;
/

--3
CREATE OR REPLACE PROCEDURE pr_empleados_tlf
(p_tlf IN VARCHAR2)
IS
```



```

v_dni EMPLEADOS.DNI%TYPE;
v_nombre EMPLEADOS.NOMBRE%TYPE;
BEGIN
    SELECT DNI,NOMBRE
    INTO v_dni, v_nombre
    FROM EMPLEADOS NATURAL JOIN TELÉFONOS
    WHERE TELÉFONO=p_tlf;

    DBMS_OUTPUT.put_line('El empleado con el teléfono '||p_tlf||' es: '||v_nombre||', con
DNI: '||v_dni||');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.put_line('No se encontró el empleado con el teléfono '||p_tlf||');
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.put_line('Hay más de un empleado con el teléfono '||p_tlf||');
END;
/
DECLARE
BEGIN
    pr_empleados_tlf('666666666');
    pr_empleados_tlf('611111111');
    pr_empleados_tlf('913333333');
END;
/
--4
CREATE OR REPLACE PROCEDURE pr_comprobar_poblaciones
IS
    v_poblacion "Códigos postales".POBLACIÓN%TYPE;
BEGIN
    SELECT POBLACIÓN INTO v_poblacion FROM "Códigos postales" GROUP BY POBLACIÓN
HAVING COUNT(*)>1;
    DBMS_OUTPUT.put_line('A la población '||v_poblacion||' no le corresponde siempre la
misma provincia.');
```

EXCEPTION

```

    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.put_line('No hay dos o más provincias que compartan la misma
población.');
```

END;

```

/
DECLARE
BEGIN
    pr_comprobar_poblaciones();
    INSERT INTO "Códigos postales" VALUES ('41008','Arganda','Sevilla');
    pr_comprobar_poblaciones();
    ROLLBACK;
END;
/
--5
CREATE OR REPLACE PROCEDURE pr_empleados_CP
IS
    CURSOR c_cod_postal IS SELECT "Código postal", AVG(SUELDO) FROM
("Códigos postales" NATURAL JOIN DOMICILIOS) NATURAL JOIN EMPLEADOS
```

```

GROUP BY "Código postal" ORDER BY "Código postal";

CURSOR c_dni (cod_post DOMICILIOS."Código postal"%TYPE)IS
SELECT NOMBRE, CALLE, SUELDO FROM DOMICILIOS NATURAL JOIN EMPLEADOS
WHERE "Código postal"=cod_post;

TYPE codigo_postal IS RECORD(
    cod_postal "Códigos postales"."Código postal"%TYPE,
    sueldo_medio NUMBER
);
TYPE persona IS RECORD(
    nombre EMPLEADOS.NOMBRE%TYPE,
    calle DOMICILIOS.CALLE%TYPE,
    sueldo EMPLEADOS.SUELDO%TYPE
);
v_cod codigo_postal;
v_persona persona;
i INT:=0;
j INT:=0;
BEGIN
    OPEN c_cod_postal;
    LOOP
        FETCH c_cod_postal INTO v_cod;
        EXIT WHEN c_cod_postal%NOTFOUND;

        OPEN c_dni(v_cod.cod_postal);
        LOOP
            FETCH c_dni INTO v_persona;
            EXIT WHEN c_dni%NOTFOUND;
            --Al menos un empleado vive en el domicilio asociado a ese código postal
            i:=i+1;
            IF i=1 THEN
                DBMS_OUTPUT.put_line('Código postal: '||v_cod.cod_postal);
            END IF;
            DBMS_OUTPUT.put_line(' '||v_persona.nombre||', '||v_persona.calle||',
' ||v_persona.sueldo);

            j:=j+1;
        END LOOP;
        CLOSE c_dni;
        IF i>=1 THEN
            DBMS_OUTPUT.put_line(' Nº empleados: '|| i ||', Sueldo medio:
' ||v_cod.sueldo_medio);
        END IF;
        i:=0;
    END LOOP;
    CLOSE c_cod_postal;
    DBMS_OUTPUT.put_line('Total empleados: '||j);
END;
/
DECLARE
BEGIN

```

```

    pr_empleados_CP();
END;

/
--6
CREATE OR REPLACE FUNCTION f_c_ba RETURN NUMBER
IS
BEGIN
    RETURN 2.920050977316;
END;

/
CREATE OR REPLACE FUNCTION f_n (p_n INTEGER) RETURN NUMBER
IS
    devolver NUMBER;
    fn NUMBER;
BEGIN
    IF p_n=1 THEN
        devolver:=f_c_ba();
    ELSE
        fn:=f_n(p_n-1);
        devolver:=FLOOR(fn)*(fn-FLOOR(fn)+1);
    END IF;
    RETURN devolver;
END;

/
CREATE OR REPLACE FUNCTION f_primo (p_n INTEGER) RETURN NUMBER
IS
BEGIN
    return FLOOR(f_n(p_n));
END;

/
DECLARE
BEGIN
    FOR i IN 1..10
    LOOP
        DBMS_OUTPUT.put_line(i || '- ' || f_primo(i));
    END LOOP;
END;

/
--7
CREATE OR REPLACE PROCEDURE pr_jefes (p_DNI IN VARCHAR2)
IS
    v_dni EMPLEADOS.JEFE%TYPE;
    v_nombre EMPLEADOS.NOMBRE%TYPE;
BEGIN
    SELECT JEFE INTO v_dni FROM EMPLEADOS WHERE DNI=p_DNI;
    IF v_dni IS NOT NULL THEN
        SELECT NOMBRE INTO v_nombre FROM EMPLEADOS WHERE DNI=v_dni;
        DBMS_OUTPUT.put_line(v_nombre || ' ' || v_dni);
        pr_jefes(v_dni);
    END IF;
EXCEPTION

```

```

    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.put_line('El empleado con DNI ' || p_dni || ' no existe');
END;
/
DECLARE
BEGIN
    pr_jefes('12345678C');
    pr_jefes('12345688C');
END;
/
--Si ponemos union en lugar de UNION ALL lanza esta excepción:
--PL/SQL: ORA-32040: la cláusula recursiva WITH debe utilizar una operación UNION ALL
CREATE OR REPLACE PROCEDURE pr_jefes_nr(p_dni IN VARCHAR2)
IS
    CURSOR c_jefes IS
        WITH tiene_jefe (empleado, jefe) AS (
            SELECT dni, jefe FROM empleados WHERE dni = p_dni
            UNION ALL
            SELECT tiene_jefe.empleado, empleados.jefe FROM tiene_jefe INNER JOIN empleados ON
tiene_jefe.jefe = empleados.dni
            WHERE empleados.jefe IS NOT NULL
        )
        SELECT empleado, jefe FROM tiene_jefe;

    v_dni empleados.jefe%TYPE;
    v_aux empleados.jefe%TYPE;
    v_nombre empleados.nombre%TYPE;

BEGIN
    OPEN c_jefes;
    LOOP
        FETCH c_jefes INTO v_aux,v_dni;
        EXIT WHEN c_jefes%notfound;
        SELECT nombre INTO v_nombre FROM empleados WHERE dni = v_dni;
        dbms_output.put_line(v_nombre || ' ' || v_dni);
    END LOOP;

    CLOSE c_jefes;
    --Procedimiento artificial para comprobar que el dni que se proporciona está en la tabla
EMPLEADOS
    --Si no lo estuviera el SELECT INTO lanza una excepción y se ejecuta el bloque EXCEPTION
    SELECT jefe INTO v_aux FROM empleados WHERE b dni = p_dni;
EXCEPTION
    WHEN no_data_found THEN
        dbms_output.put_line('El empleado con DNI ' || p_dni || ' no existe');
END;
/

DECLARE BEGIN
    pr_jefes_nr('12345678C');
    pr_jefes_nr('12345688C');
END;

```

SALIDA POR CONSOLA EN TEXTO PLANO

1650

Procedimiento PL/SQL terminado correctamente.

Procedure PR_EMPLEADOS_TLF compilado

No se encontró el empleado con el teléfono 666666666.

El empleado con el teléfono 611111111 es: Carlota Cerezo, con DNI: 12345678C.

Hay más de un empleado con el teléfono 913333333.

Procedimiento PL/SQL terminado correctamente.

Procedure PR_COMPROBAR_POBLACIONES compilado

No hay dos o más provincias que compartan la misma población.

A la población Arganda no le corresponde siempre la misma provincia.

Procedimiento PL/SQL terminado correctamente.

Procedure PR_EMPLEADOS_CP compilado

Código postal: 14200

Laura López, Diamante, 1500

Pedro Pérez, Diamante, 2000

Nº empleados: 2, Sueldo medio: 1750

Código postal: 14900

Pedro Pérez, Carbón, 2000

Nº empleados: 1, Sueldo medio: 2000

Código postal: 28004

Antonio Arjona, Cántaro, 5000

Nº empleados: 1, Sueldo medio: 5000

Código postal: 28040

Antonio Arjona, Avda. Complutense, 5000

Nº empleados: 1, Sueldo medio: 5000

Total empleados: 5

Procedimiento PL/SQL terminado correctamente.

Function F_C_BA compilado

Function F_N compilado

Function F_PRIMO compilado

1.- 2

2.- 3

3.- 5

4.- 7

- 5.- 11
- 6.- 13
- 7.- 17
- 8.- 19
- 9.- 23
- 10.- 29

Procedimiento PL/SQL terminado correctamente.

Procedure PR_JEFES compilado

Laura López 12345678L
Pedro Pérez 12345678P
Antonio Arjona 12345678A
El empleado con DNI 12345688C no existe

Procedimiento PL/SQL terminado correctamente.

Procedure PR_JEFES_NR compilado

Laura López 12345678L
Pedro Pérez 12345678P
Antonio Arjona 12345678A
El empleado con DNI 12345688C no existe

Procedimiento PL/SQL terminado correctamente.