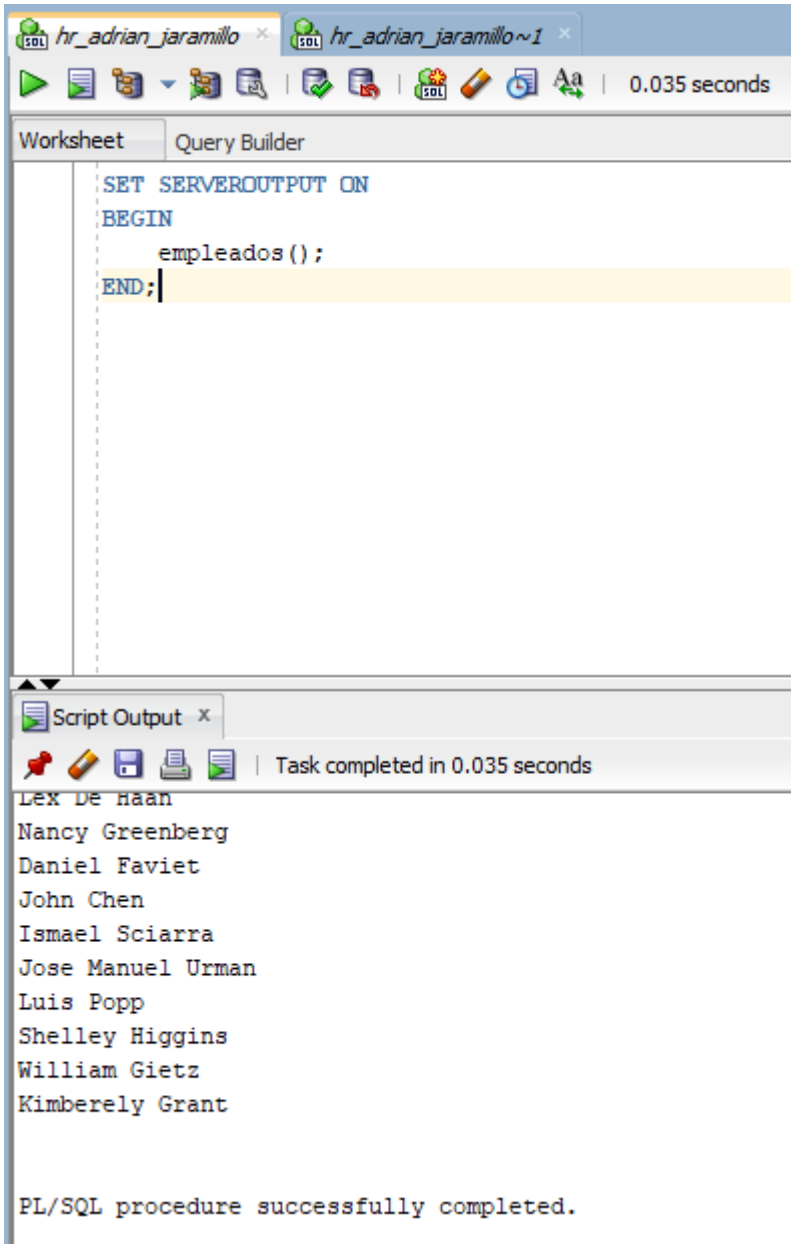
1. Desarrollar un procedimiento que visualice el nombre y apellido de todos los empleados ordenados por número de departamento.

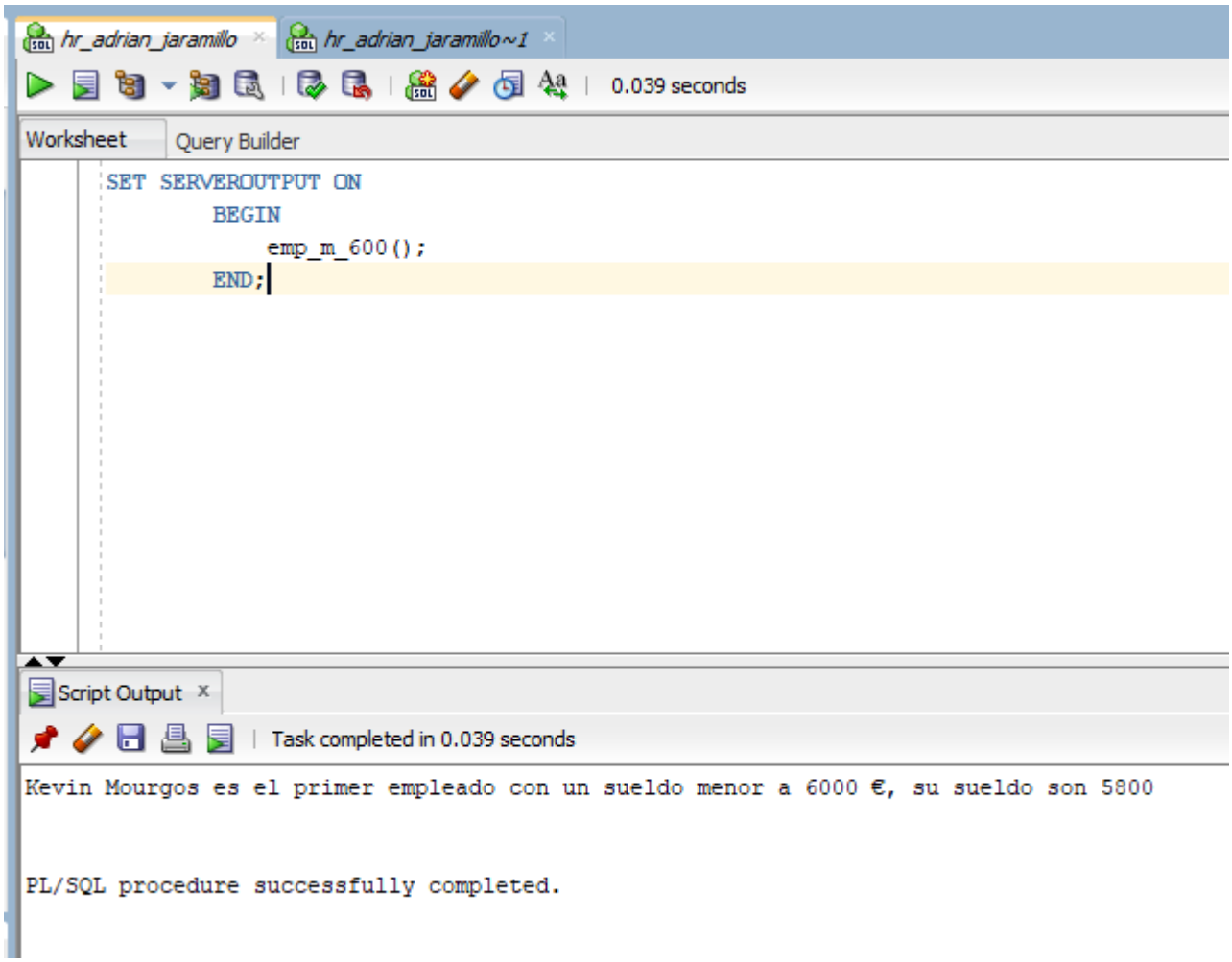
```
-- ############ --
-- PARTE CÓDIGO --
-- ############ --

SET SERVEROUTPUT ON
BEGIN
    empleados();
END;

-- ################### --
-- PARTE PROCEDIMIENTO --
-- ################### --

CREATE OR REPLACE PROCEDURE EMPLEADOS AS
CURSOR c_empleados IS
SELECT first_name, last_name
    FROM employees
    ORDER BY department_id;

v_reg_cursor c_empleados%ROWTYPE;

BEGIN
    OPEN c_empleados;
    FETCH c_empleados INTO v_reg_cursor;

    WHILE c_empleados%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(v_reg_cursor.first_name || ' ' ||v_reg_cursor.last_name);
        FETCH c_empleados INTO v_reg_cursor;
    END LOOP;

    CLOSE c_empleados;
END EMPLEADOS;
```



2. Desarrollar un procedimiento que encuentre el primer empleado con un sueldo menor de 6.000 €.

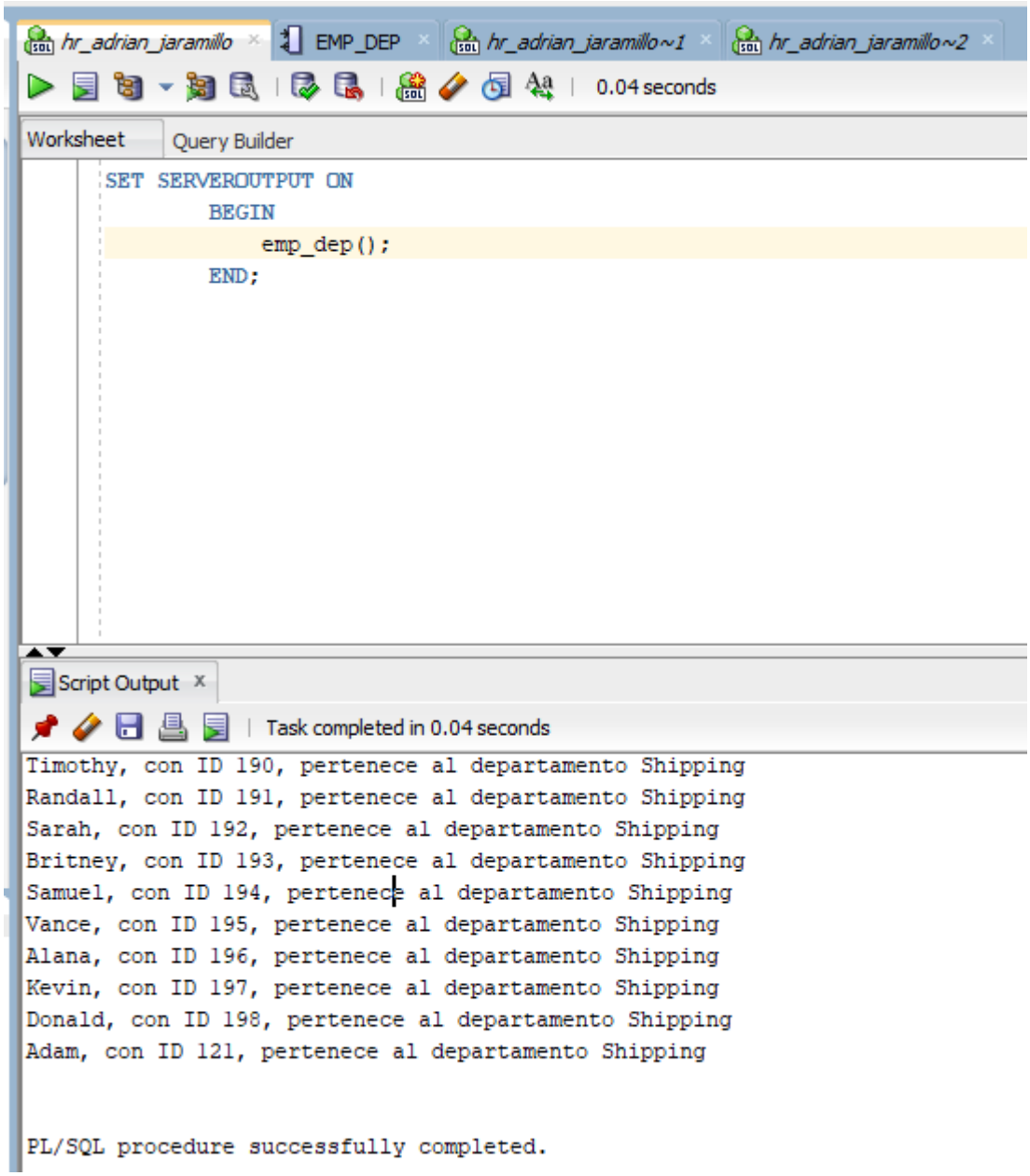
```
-- ############ --
-- PARTE CÓDIGO --
-- ############ --

SET SERVEROUTPUT ON
BEGIN
    emp_m_600();
END;

-- ################### --
-- PARTE PROCEDIMIENTO --
-- ################### --

create or replace PROCEDURE emp_m_600 AS
```

```
        CURSOR c_emp_m_600 IS
        SELECT first_name, last_name, salary
            FROM employees
            WHERE salary < 6000
            ORDER BY salary DESC;

        v_reg_cursor c_emp_m_600%ROWTYPE;

        BEGIN

            OPEN c_emp_m_600;
            FETCH c_emp_m_600 INTO v_reg_cursor;

            DBMS_OUTPUT.PUT_LINE(v_reg_cursor.first_name || ' ' || v_reg_cursor.last_name
|| ' es el primer empleado con un sueldo menor a 6000 €, su sueldo son ' ||
v_reg_cursor.salary);

            CLOSE c_emp_m_600;

        END emp_m_600;
```



3. Realizar un procedimiento que visualice el número y nombre de un empleado, así como el nombre de su departamento, ordenados por nombre de departamento.

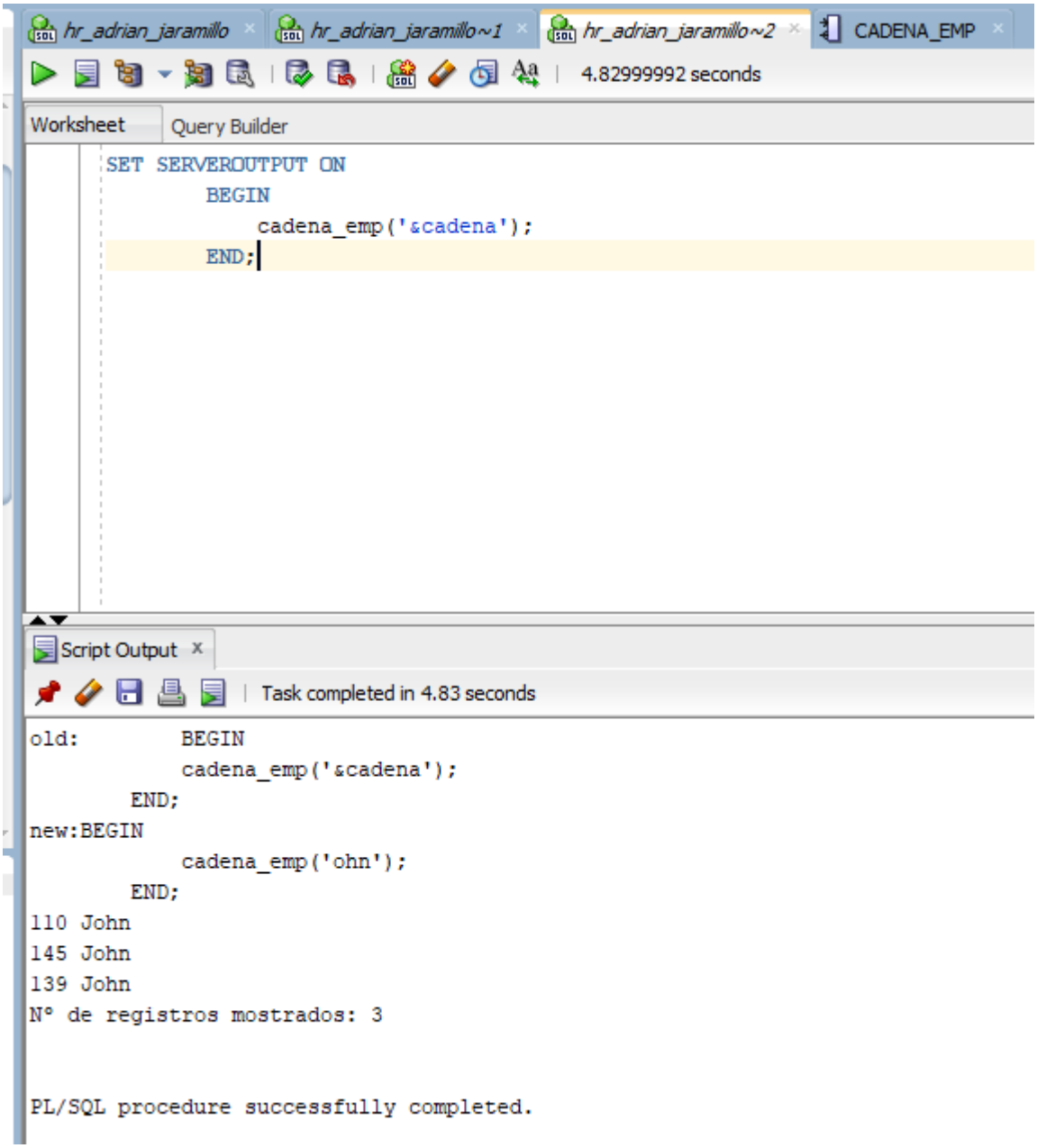
```
        -- ############ --
        -- PARTE CÓDIGO --
        -- ############ --

        SET SERVEROUTPUT ON
        BEGIN
            emp_dep();
        END;

        -- ################### --
        -- PARTE PROCEDIMIENTO --
        -- ################### --

        create or replace PROCEDURE emp_dep AS

        CURSOR c_emp_dep IS
        SELECT e.employee_id, e.first_name, d.department_name
            FROM employees e
            INNER JOIN departments d ON
            e.department_id = d.department_id
            ORDER BY d.department_name;

        v_reg_cursor c_emp_dep%ROWTYPE;

        BEGIN

            OPEN c_emp_dep;
            FETCH c_emp_dep INTO v_reg_cursor;

            WHILE c_emp_dep%FOUND LOOP

                DBMS_OUTPUT.PUT_LINE(v_reg_cursor.first_name || ', con ID ' ||
v_reg_cursor.employee_id || ', pertenece al departamento ' || v_reg_cursor.department_name
);
                FETCH c_emp_dep INTO v_reg_cursor;

            END LOOP;

            CLOSE c_emp_dep;

        END emp_dep;
```



4. Escribir un procedimiento que reciba una cadena y visualice el nombre y el número de empleado de todos los empleados cuyo nombre contenga la cadena especificada. Al finalizar visualizar el número de empleados mostrados.

```
-- ############# --
-- PARTE CÓDIGO --
-- ############# --

SET SERVEROUTPUT ON
BEGIN
    cadena_emp('&cadena');
END;

-- ################### --
-- PARTE PROCEDIMIENTO --
```

```
        -- ################### --

        create or replace PROCEDURE cadena_emp
        (
            CADENA IN VARCHAR2
        ) AS

        CURSOR c_cadena_emp IS
        SELECT employee_id, first_name
            FROM employees
            WHERE
                first_name LIKE '%'||CADENA||'%' OR
                upper(first_name) LIKE '%'||CADENA||'%' OR
                lower(first_name) LIKE '%'||CADENA||'%';

        v_reg_cursor c_cadena_emp%ROWTYPE;

        BEGIN

            OPEN c_cadena_emp;
            FETCH c_cadena_emp INTO v_reg_cursor;

            WHILE c_cadena_emp%FOUND LOOP

                DBMS_OUTPUT.PUT_LINE(v_reg_cursor.employee_id || ' '
||v_reg_cursor.first_name);
                FETCH c_cadena_emp INTO v_reg_cursor;

            END LOOP;

            DBMS_OUTPUT.PUT_LINE( 'Nº de registros mostrados: ' || c_cadena_emp%ROWCOUNT);

            CLOSE c_cadena_emp;

        END cadena_emp;
```



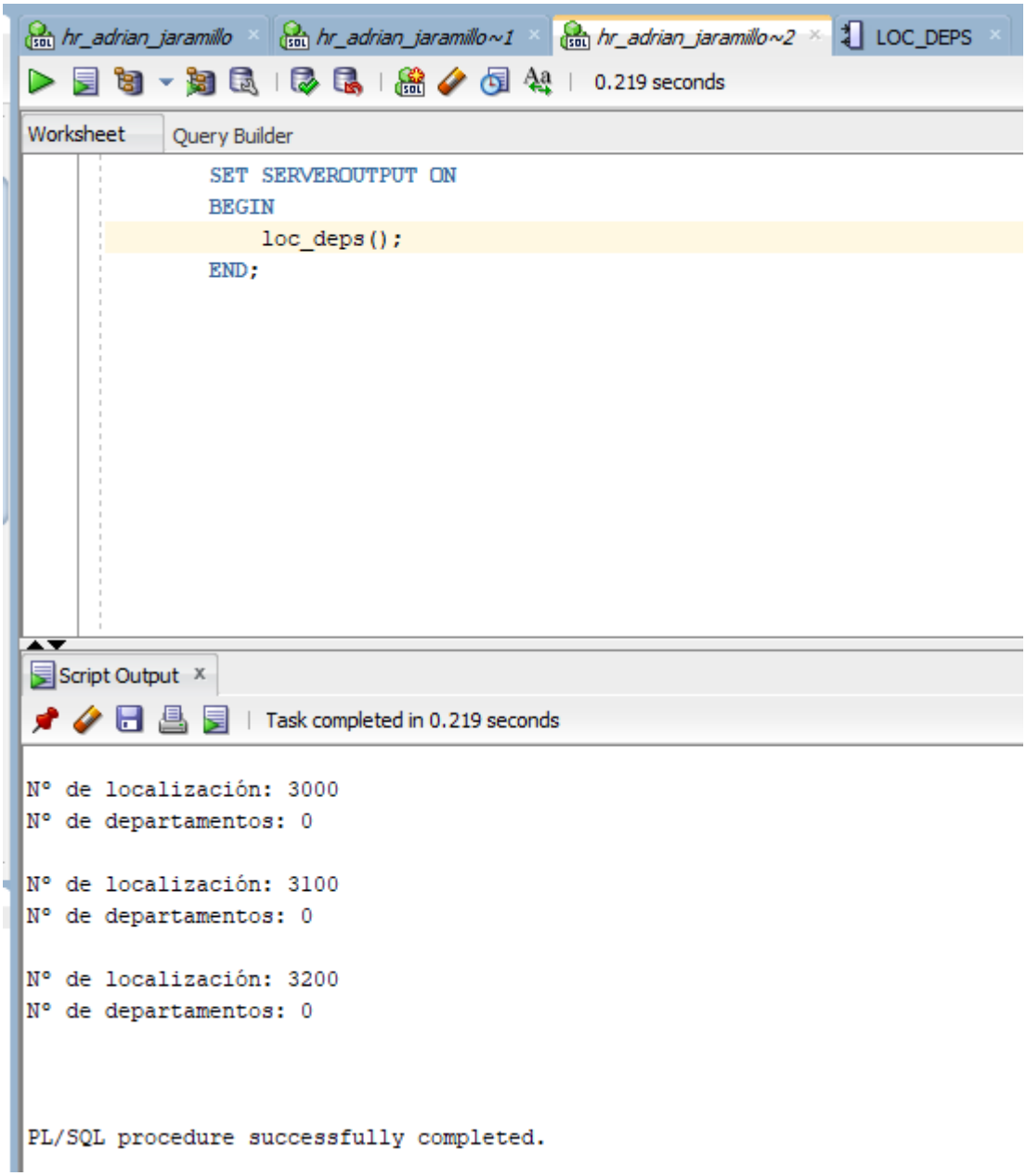
5. Crear un procedimiento que muestre las localizaciones (dirección y ciudad) y el número de departamentos que tiene (incluso si no tiene departamentos).

```
-- ############ --
-- PARTE CÓDIGO --
-- ############ --

SET SERVEROUTPUT ON
BEGIN
    loc_deps();
END;

-- ################## --
-- PARTE PROCEDIMIENTO --
-- ################## --
```

```
        create or replace PROCEDURE loc_deps AS

        --Cursor primer ID localización
        CURSOR c_idloc_min IS
        SELECT location_id
            FROM locations
            WHERE ROWNUM = 1;

        v_reg_cursor1 c_idloc_min%ROWTYPE;

        --Cursor último ID localización
        CURSOR c_idloc_max IS
        SELECT location_id
            FROM locations
            WHERE ROWNUM = 1
            ORDER BY location_id DESC;

        v_reg_cursor2 c_idloc_max%ROWTYPE;

        --Cursor filtrar registros por localización
        CURSOR c_locs IS
        SELECT d.department_id dep_id, d.location_id loc_d, l.location_id loc_l,
l.street_address street, l.city city
            FROM departments d
            RIGHT JOIN locations l ON
            d.location_id = l.location_id;

        v_reg_cursor3 c_locs%ROWTYPE;

        --Cursor contiene todos los ID localización posibles
        CURSOR c_id_locs IS
        SELECT location_id
            FROM locations;

        v_reg_cursor4 c_id_locs%ROWTYPE;

        v_booleana varchar2(10) := 'false';
        v_existe varchar2(10);

        --Variable contadora de registros con el mismo ID de localización
        cont number(10);



        BEGIN

            --Abrimos los cursores necesarios para los valores del for
            OPEN c_idloc_min;
            OPEN c_idloc_max;

            --Recorremos los cursores abiertos para obtener sus valores
```

```
FETCH c_idloc_min INTO v_reg_cursor1;
FETCH c_idloc_max INTO v_reg_cursor2;



--Aquí especificamos cuántos registros hay de cada ID de
--localización en el select hecho arriba, sabiendo así
--cuántos departamentos hay en cada localización
FOR i IN v_reg_cursor1.location_id .. v_reg_cursor2.location_id LOOP

    --Aquí comprobamos que el ID de localización generado por el for existe.
    --Si no existe pasamos a comprobar el siguiente ID de localización

    WHILE v_booleana = 'false' LOOP

        OPEN c_id_locs;
        FETCH c_id_locs INTO v_reg_cursor4;

        WHILE c_id_locs%FOUND LOOP
            IF v_reg_cursor4.location_id = i THEN
                v_existe := 's';
            END IF;
            FETCH c_id_locs INTO v_reg_cursor4;
        END LOOP;

        v_booleana := 'true';
        CLOSE c_id_locs;

    END LOOP;




    --En el caso de que existiera el ID de localización generado,
    --v_existe toma el valor de s, o "sí existe", y entramos al código

    IF v_existe = 's' THEN

        --Iniciamos siempre este contador a 0 para resetear el valor
        --anterior y que empiece una nueva cuenta de registros
        cont := 0;
        OPEN c_locs;
        FETCH c_locs INTO v_reg_cursor3;


        WHILE c_locs%FOUND LOOP

            IF v_reg_cursor3.loc_l = i AND v_reg_cursor3.loc_d = i THEN
                cont := cont+1;
            END IF;

            FETCH c_locs INTO v_reg_cursor3;
```

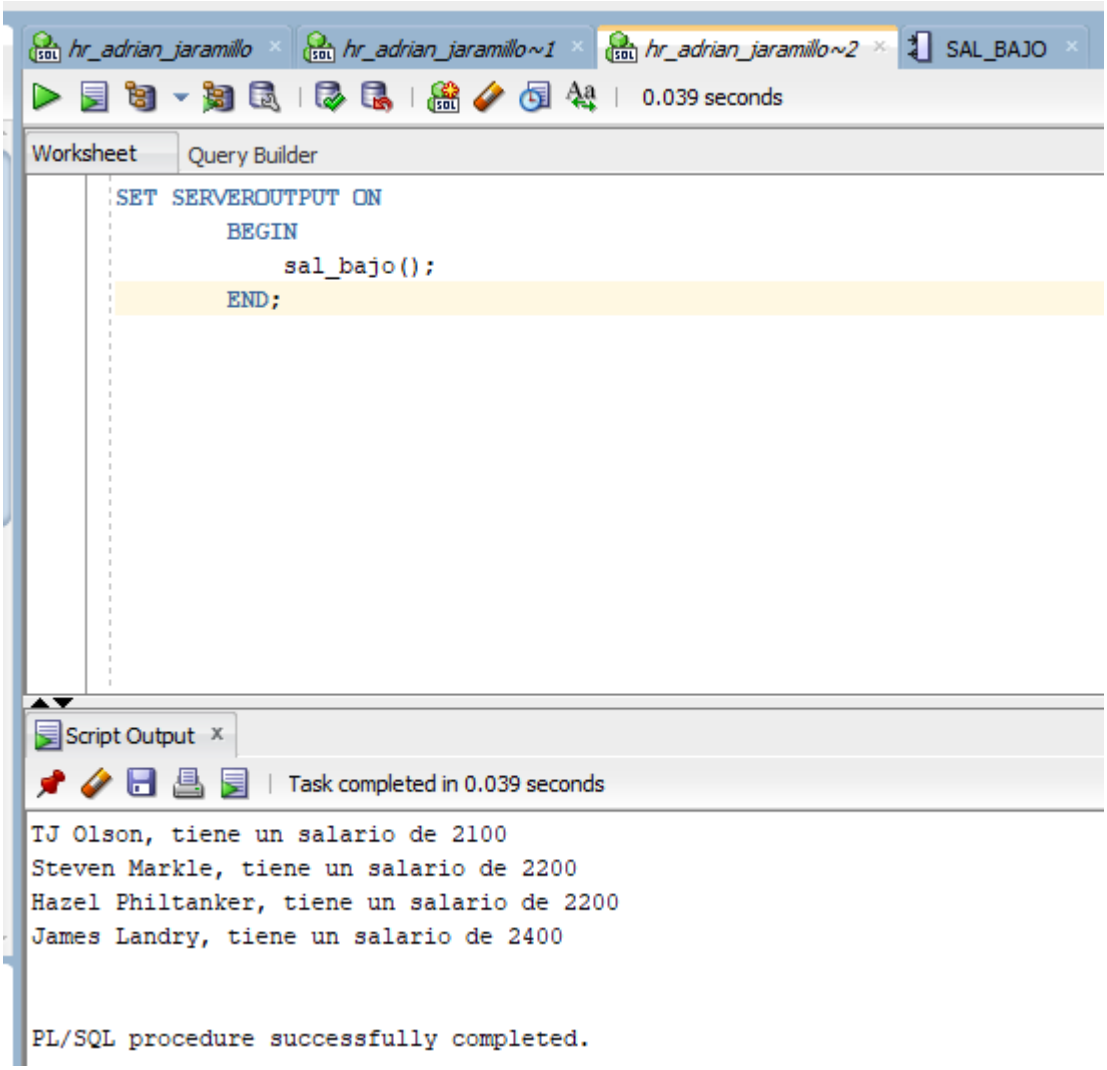
```
                END LOOP;

                DBMS_OUTPUT.PUT_LINE('Nº de localización: ' || i);
                DBMS_OUTPUT.PUT_LINE('Nº de departamentos: ' || cont);
                DBMS_OUTPUT.PUT_LINE('');

                CLOSE c_locs;

            END IF;

            --Cambiamos éstos valores a como estaban anteriormente para
            --que al volver arriba con for, se pueda seguir comprobando
            --si la localización existe
            v_booleana := 'false';
            v_existe := '';

        END LOOP;--Fin del FOR


        CLOSE c_idloc_min;
        CLOSE c_idloc_max;


    END loc_deps;
```



6. Escribir un programa que visualice el nombre, apellido y el salario de los cuatro empleados que tienen el salario más bajo.

```
-- ############# --
-- PARTE CÓDIGO --
-- ############# --

SET SERVEROUTPUT ON
BEGIN
    sal_bajo();
END;

-- ################### --
-- PARTE PROCEDIMIENTO --
-- ################### --

create or replace PROCEDURE sal_bajo AS
```

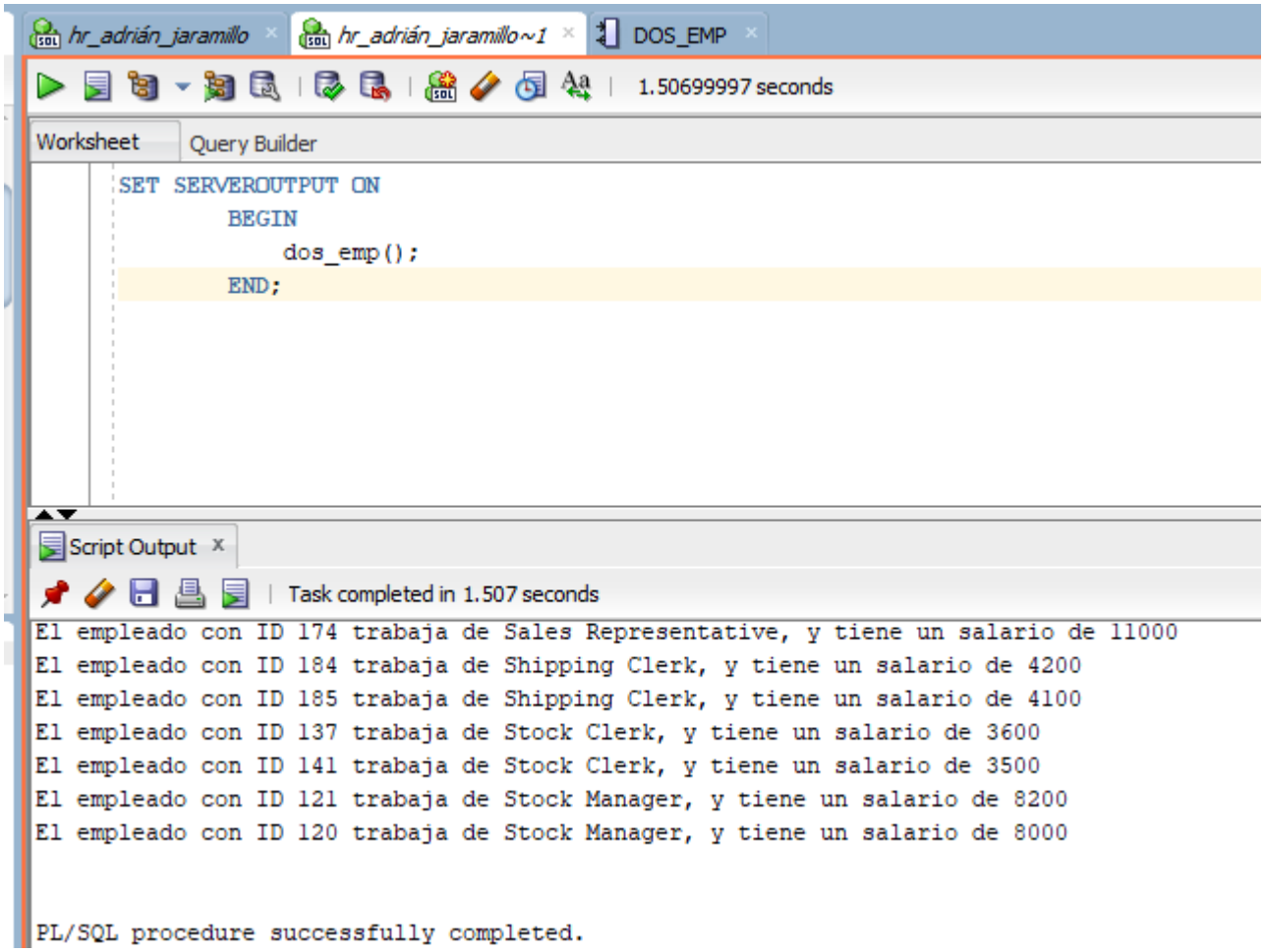
```
        CURSOR c_sal_bajo IS
        SELECT first_name, last_name, salary
            FROM employees
            ORDER BY salary;

        v_reg_cursor c_sal_bajo%ROWTYPE;


        BEGIN

            OPEN c_sal_bajo;
            FETCH c_sal_bajo INTO v_reg_cursor;


            FOR i IN 1 .. 4 LOOP

            DBMS_OUTPUT.PUT_LINE(v_reg_cursor.first_name || ' ' || v_reg_cursor.last_name
|| ', tiene un salario de ' || v_reg_cursor.salary);
            FETCH c_sal_bajo INTO v_reg_cursor;

            END LOOP;


        END sal_bajo;
```



7. Codificar un programa que visualice los dos empleados que ganan más de cada oficio.

```
-- ############ --
-- PARTE CÓDIGO --
-- ############ --

SET SERVEROUTPUT ON
BEGIN
    dos_emp();
END;

-- ################### --
-- PARTE PROCEDIMIENTO --
-- ################### --

create or replace PROCEDURE dos_emp AS

CURSOR c_dos_emp IS
SELECT e.employee_id e_id, j.job_title j_title, e.salary e_salary
    FROM employees e
    INNER JOIN jobs j ON
```

```
            e.job_id = j.job_id
            ORDER BY j.job_title ASC, e.salary DESC;

        v_reg_cursor c_dos_emp%ROWTYPE;

        --Variable contador
        v_contador NUMBER(10) := 0;
        v_trabajo jobs.job_title%TYPE;

        BEGIN

            OPEN c_dos_emp;
            FETCH c_dos_emp INTO v_reg_cursor;

            v_trabajo := v_reg_cursor.j_title;

            WHILE c_dos_emp%FOUND LOOP

                IF v_reg_cursor.j_title != v_trabajo THEN

                   v_trabajo := v_reg_cursor.j_title;
                   v_contador := 0;

                END IF;

                v_contador := v_contador+1;

                IF v_contador < 3 THEN
                    DBMS_OUTPUT.PUT_LINE('El empleado con ID ' || v_reg_cursor.e_id || '
trabaja de ' || v_reg_cursor.j_title || ', y tiene un salario de ' ||
v_reg_cursor.e_salary);
                END IF;

                FETCH c_dos_emp INTO v_reg_cursor;

            END LOOP;


        END dos_emp;
```


hr_adrián_jaramillo
hr_adrián_jaramillo~1
DOS_EMP
1.50699997 seconds
Worksheet
Query Builder
SET SERVEROUTPUT ON
BEGIN
dos_emp();
END;
Script Output
Task completed in 1.507 seconds
El empleado con ID 174 trabaja de Sales Representative, y tiene un salario de 11000
El empleado con ID 184 trabaja de Shipping Clerk, y tiene un salario de 4200
El empleado con ID 185 trabaja de Shipping Clerk, y tiene un salario de 4100
El empleado con ID 137 trabaja de Stock Clerk, y tiene un salario de 3600
El empleado con ID 141 trabaja de Stock Clerk, y tiene un salario de 3500
El empleado con ID 121 trabaja de Stock Manager, y tiene un salario de 8200
El empleado con ID 120 trabaja de Stock Manager, y tiene un salario de 8000
PL/SQL procedure successfully completed.