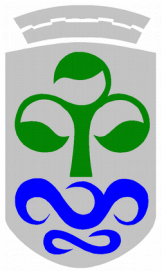


UNIVERSIDADE
DE VIGO

ESCOLA SUPERIOR DE ENXEÑERÍA INFORMÁTICA

PRÁCTICA 7: PLSQL

GRUPO: BDII6_2

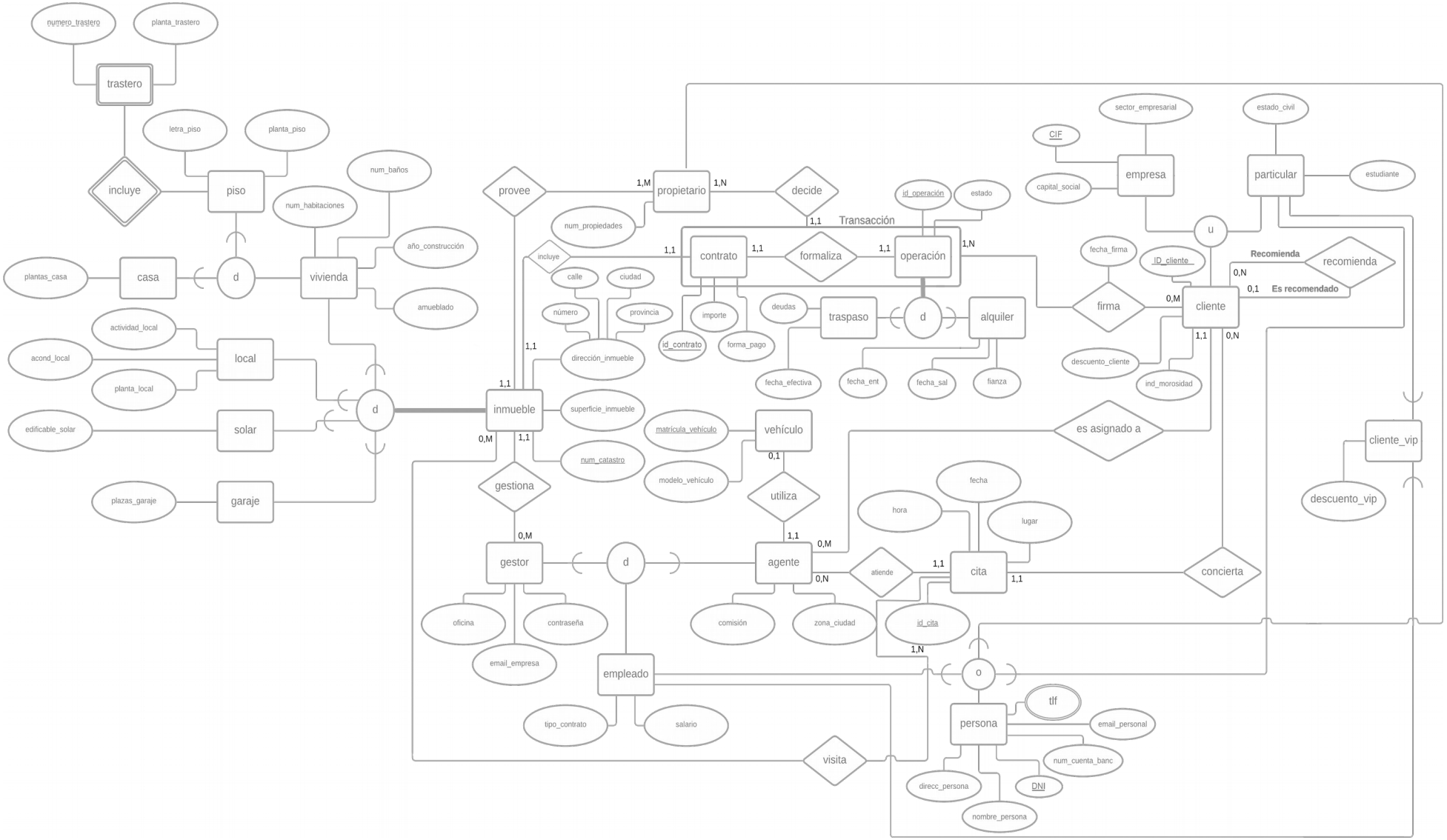
TEMÁTICA: INMOBILIARIA

DNI: 77460353X **NOMBRE: Gómez Dopazo, Rubén**
DNI: 34629985N **NOMBRE: Pérez Gómez, Cristian**
DNI: 34275678C **NOMBRE: Pérez Iglesia, Luis**
DNI: 76734153N **NOMBRE: Gómez González, Alejandro**

BASES DE DATOS II 2020-2021

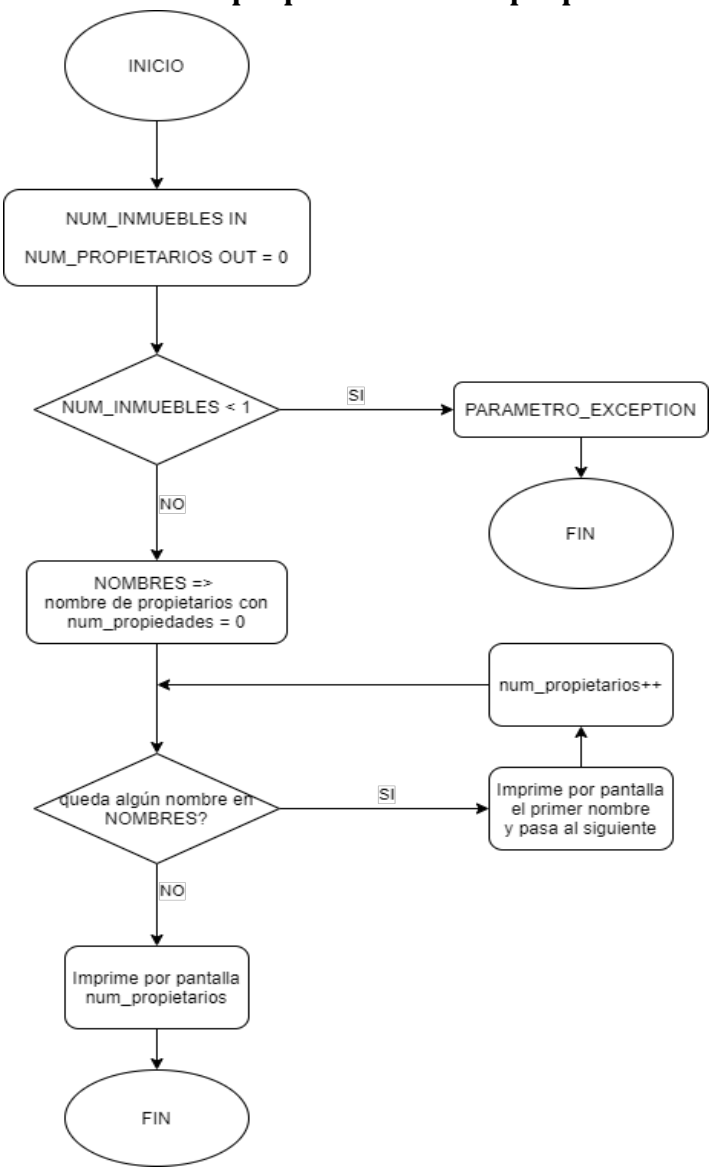
CALIFICACIÓN

DIAGRAMA ERR DEFINITIVO

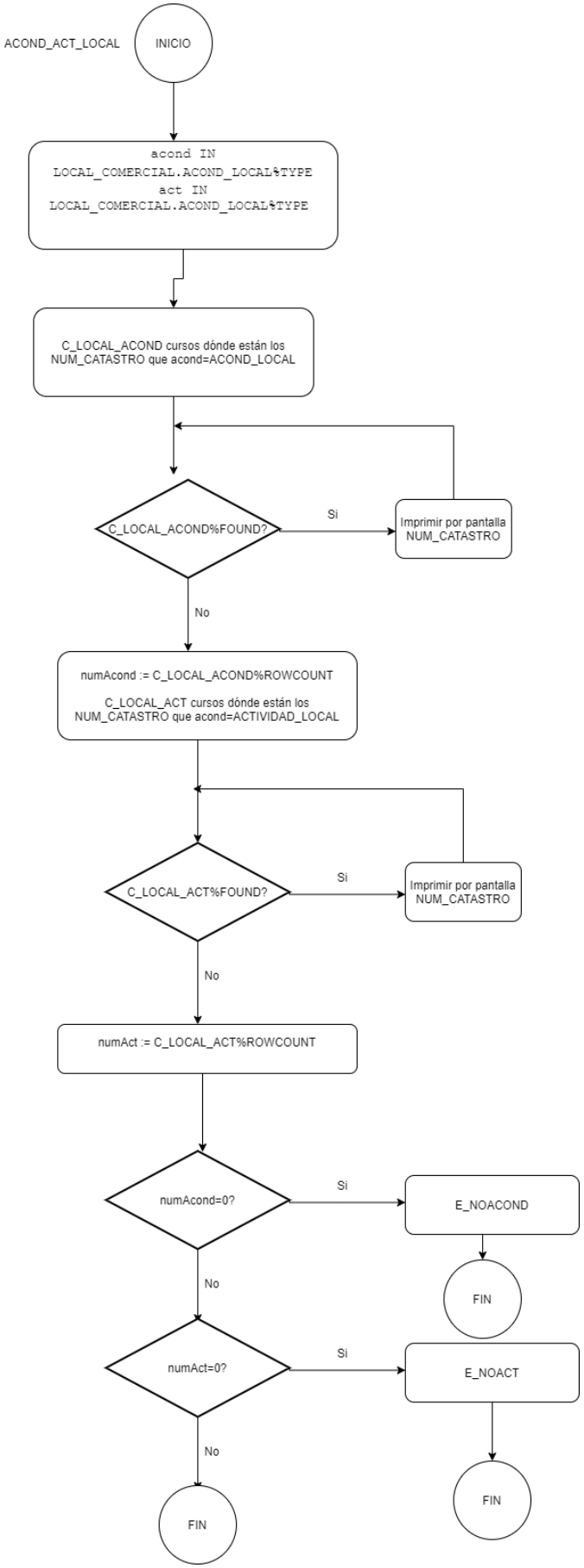


DIAGRAMAS DE FLUJO DE PROCEDIMIENTOS Y FUNCIONES

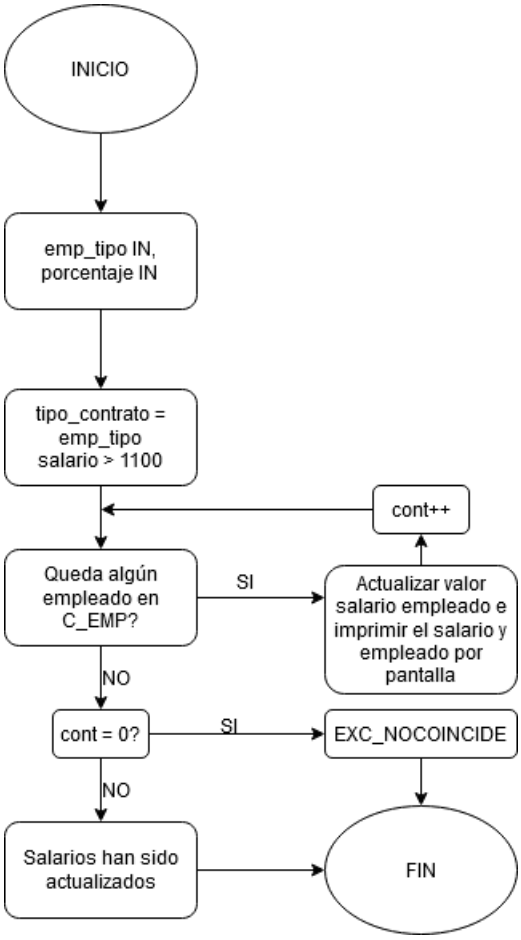
Procedimiento 1: Número de propietarios con X propiedades.



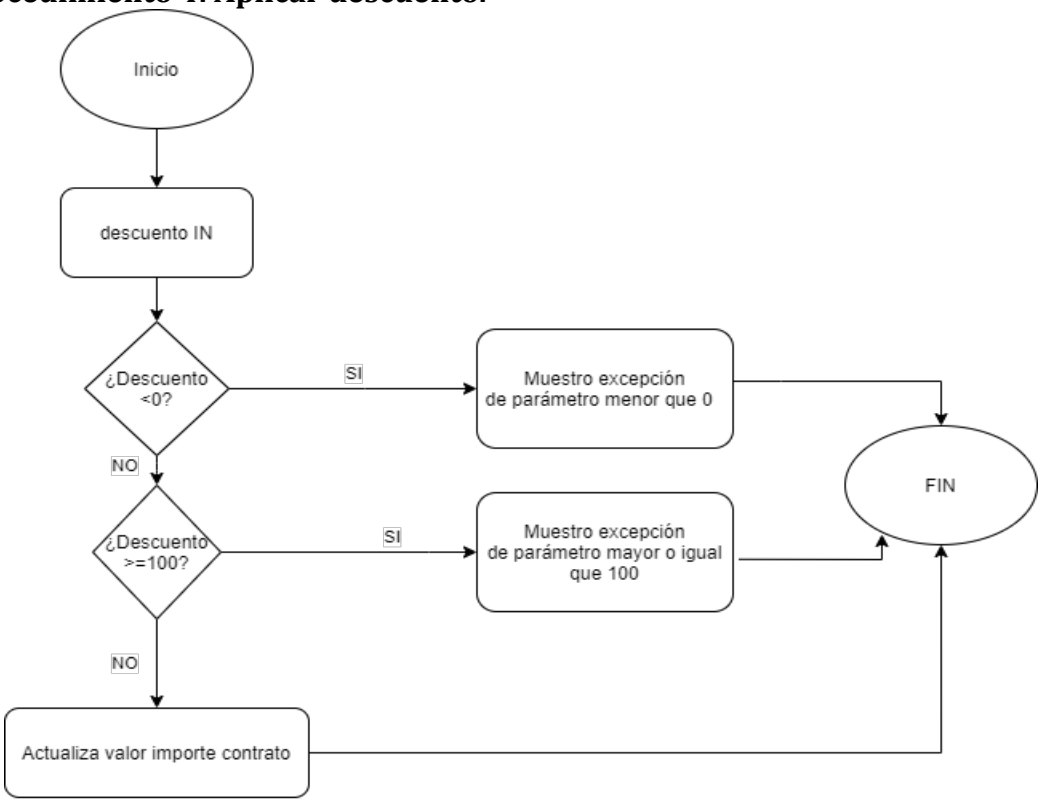
Procedimiento 2: Acondicionamiento actual del local.



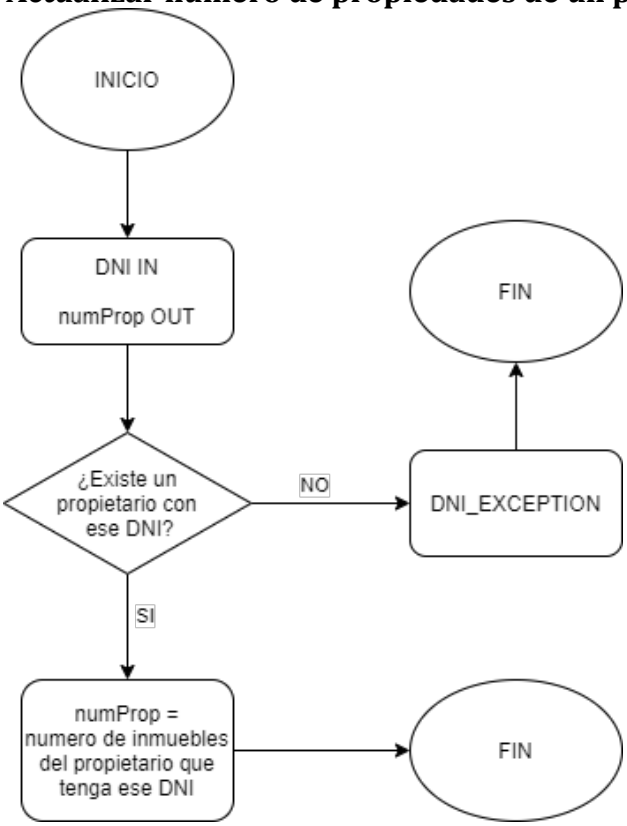
Procedimiento 3: Bajar sueldo a los empleados con X tipo de contrato.



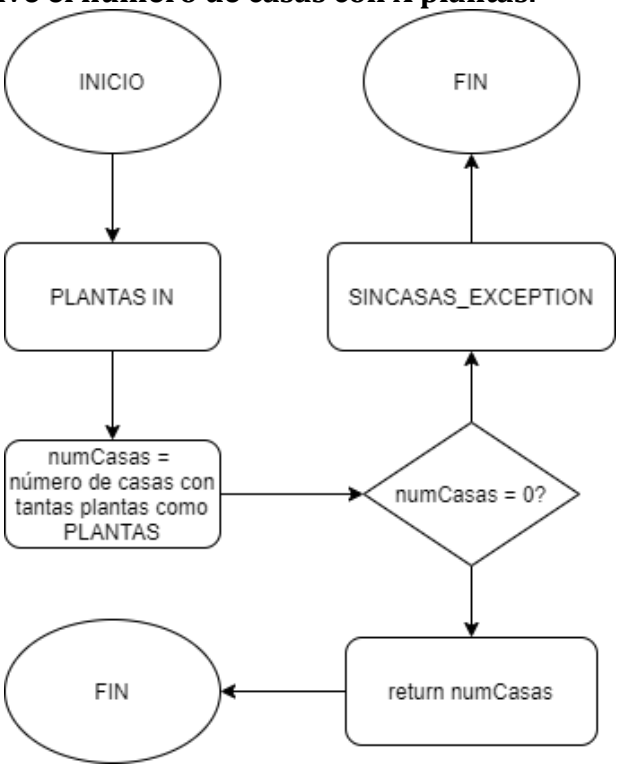
Procedimiento 4: Aplicar descuento.



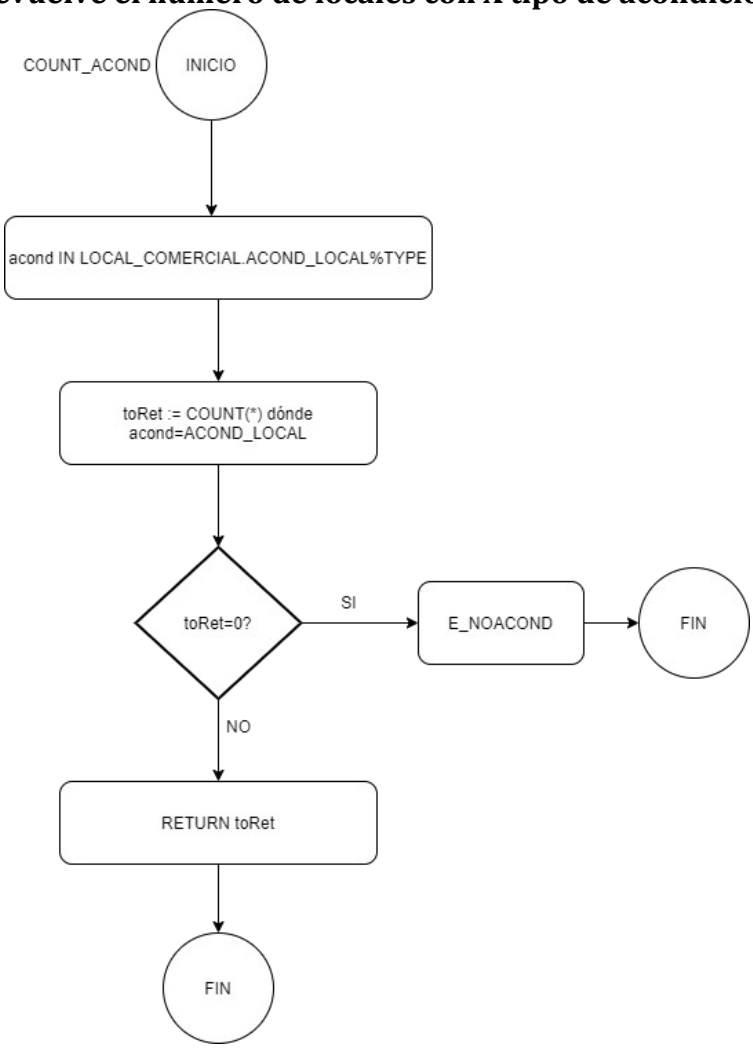
Procedimiento 5: Actualizar número de propiedades de un propietario.



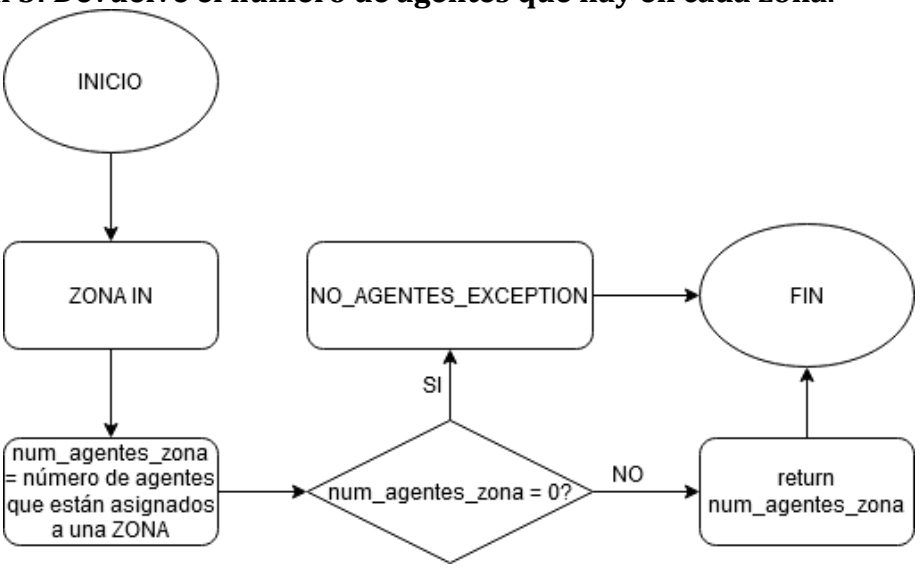
Función 1: Devuelve el número de casas con X plantas.



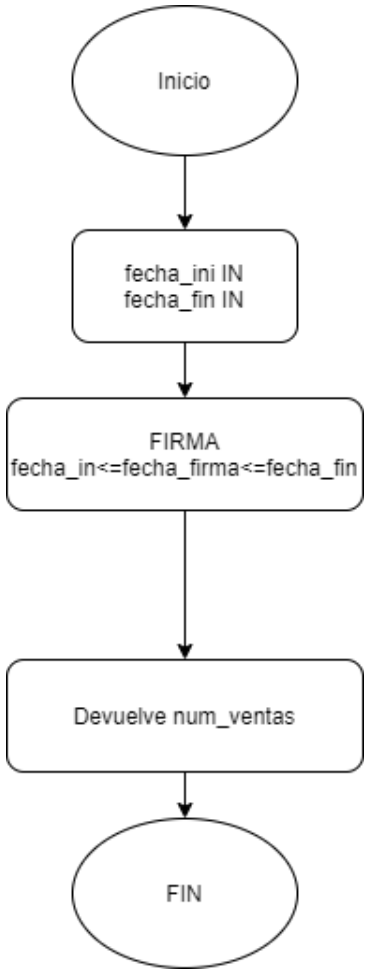
Función 2: Devuelve el número de locales con X tipo de acondicionamiento.



Función 3: Devuelve el número de agentes que hay en cada zona.



Función 4: Devuelve el número de ventas entre dos fechas dadas.



SENTENCIAS DE DEFINICIÓN DE PROCEDIMIENTOS Y FUNCIONES

--Procedimiento 1:

```
CREATE OR REPLACE PROCEDURE PROPIETARIOS(NUM_INMUBLES IN NUMBER,
NUM_PROPIETARIOS OUT NUMBER)
    IS
        PARAMETRO_EXCEPTION EXCEPTION;
        NOMBREPROP PERSONA.NOMBRE_PERSONA%TYPE;
        CURSOR NOMBRES IS
            SELECT A.nombre_persona
            FROM PERSONA A, PROPIETARIO B
            WHERE A.DNI_persona=B.DNI_propietario AND
B.num_Propiedades=NUM_INMUBLES;
    BEGIN
        IF (NUM_INMUBLES < 1) THEN
            RAISE PARAMETRO_EXCEPTION;
        END IF;
        NUM_PROPIETARIOS := 0;
        OPEN NOMBRES;
        LOOP
            FETCH NOMBRES INTO NOMBREPROP;
            EXIT WHEN NOMBRES%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE(NOMBREPROP);
            NUM_PROPIETARIOS := NUM_PROPIETARIOS + 1;
        END LOOP;
        DBMS_OUTPUT.PUT_LINE('Propietarios con ' || NUM_INMUBLES
        || ' inmuebles (' || NOMBRES%ROWCOUNT || ')');
        CLOSE NOMBRES;

    EXCEPTION
        WHEN PARAMETRO_EXCEPTION THEN
           DBMS_OUTPUT.PUT_LINE('La cantidad de inmuebles debe
           ser mayor que 0');
        WHEN OTHERS THEN
           DBMS_OUTPUT.PUT_LINE('Código: ' || SQLCODE);
END Propietarios;
/
show errors
```

--Procedimiento 2:

```
CREATE OR REPLACE PROCEDURE ACOND_ACT_LOCAL(acond IN
LOCAL_COMERCIAL.ACOND_LOCAL%TYPE, act IN
LOCAL_COMERCIAL.ACTIVIDAD_LOCAL%TYPE,numLocales OUT NUMBER)
    IS
        CURSOR C_LOCAL_ACOND IS
            SELECT NUM_CATASTRO
            FROM LOCAL_COMERCIAL
            WHERE ACOND_LOCAL = acond
            FOR UPDATE;
        CURSOR C_LOCAL_ACT IS
```

```
            SELECT NUM_CATASTRO
            FROM LOCAL_COMERCIAL
            WHERE ACTIVIDAD_LOCAL = act
            FOR UPDATE;

        i  LOCAL_COMERCIAL.NUM_CATASTRO%TYPE;
        NUM_CATASTRO LOCAL_COMERCIAL.NUM_CATASTRO%TYPE;
        numAcond NUMBER;
        numAct NUMBER;
        E_NOACOND EXCEPTION;
        E_NOACT EXCEPTION;
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Locales con acondicinamiento '||
        acond);
        NumLocales := 0;
        OPEN C_LOCAL_ACOND;
        FETCH C_LOCAL_ACOND INTO NUM_CATASTRO;
        WHILE C_LOCAL_ACOND%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(NUM_CATASTRO);
            FETCH C_LOCAL_ACOND INTO NUM_CATASTRO;
        END LOOP;
        numAcond := C_LOCAL_ACOND%ROWCOUNT;
        CLOSE C_LOCAL_ACOND;
        DBMS_OUTPUT.PUT_LINE('Locales con actividad '|| act);
        OPEN C_LOCAL_ACT;
        FETCH C_LOCAL_ACT INTO NUM_CATASTRO;
        WHILE C_LOCAL_ACT%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(NUM_CATASTRO);
            FETCH C_LOCAL_ACT INTO NUM_CATASTRO;
            NumLocales := NumLocales + 1;
        END LOOP;
        numAct := C_LOCAL_ACT%ROWCOUNT;
        CLOSE C_LOCAL_ACT;
        IF (numAcond = 0) THEN
            RAISE E_NOACOND;
        END IF;
        IF (numAct = 0) THEN
            RAISE E_NOACT;
        END IF;
    EXCEPTION
        WHEN E_NOACOND THEN
            DBMS_OUTPUT.PUT_LINE('Excepción: No coincide ningún
           acondicionamiento');
        WHEN E_NOACT THEN
            DBMS_OUTPUT.PUT_LINE('Excepción: No coincide con
            ninguna actividad');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Código: ' || SQLCODE);
END ACOND_ACT_LOCAL;
/
show errors
```

--Procedimiento 3:

```
CREATE OR REPLACE PROCEDURE EMP_BAJAR_SUELDO(emp_tipo IN
VARCHAR, porcentaje IN NUMBER)

    IS
        EXC_NOCOINCIDE EXCEPTION;
        CURSOR C_EMP IS
            SELECT *
            FROM EMPLEADO
            WHERE tipo_contrato LIKE emp_tipo AND salario > 1100
            FOR UPDATE;
        regEmp EMPLEADO%ROWTYPE;
        cont NUMBER;
    BEGIN
        cont := 0;
        FOR regEmp IN C_EMP LOOP
            DBMS_OUTPUT.PUT('Empleado: ' || regEmp.DNI_empleado
            || '; Salario actual: ' || regEmp.Salario);
            UPDATE EMPLEADO SET SALARIO = SALARIO*(1 -
             porcentaje)
            WHERE CURRENT OF C_EMP;
            DBMS_OUTPUT.PUT_LINE('; Salario nuevo: ' ||
             regEmp.Salario);
            cont := cont + 1;
        END LOOP;
        IF cont = 0 THEN RAISE EXC_NOCOINCIDE;
        END IF;
        DBMS_OUTPUT.PUT_LINE(cont || ' SALARIOS HAN SIDO
         ACTUALIZADOS');
    EXCEPTION
        WHEN EXC_NOCOINCIDE THEN
            DBMS_OUTPUT.PUT_LINE('ERROR: El tipo de contrato
            introducido no coincide con ninguno de los
            existentes.');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Codigo: ' || SQLCODE);
            DBMS_OUTPUT.PUT_LINE('Mensaje: ' || SUBSTR(SQLERRM,
            11,100));
END EMP_BAJAR_SUELDO;
/
show errors
```

--Procedimiento 4:

```
CREATE OR REPLACE PROCEDURE APLICAR_DESCUENTO(descuento IN
NUMBER)

    IS
        PARAMETRO_MENOR_EXCEPTION EXCEPTION;
        PARAMETRO_MAYOR_EXCEPTION EXCEPTION;
        CURSOR C_CONTRATO IS
            SELECT *
            FROM CONTRATO
            FOR UPDATE;
        RegContrato CONTRATO%ROWTYPE;
```

```
    BEGIN
    IF (descuento < 0)THEN
            RAISE PARAMETRO_MENOR_EXCEPTION;
        END IF;
    IF (descuento >= 100)THEN
            RAISE PARAMETRO_MAYOR_EXCEPTION;
        END IF;
    FOR RegContrato IN C_CONTRATO LOOP
        DBMS_OUTPUT.PUT('ID contrato: ' ||
RegContrato.ID_contrato);
        DBMS_OUTPUT.PUT_LINE('; Precio original: ' ||
RegContrato.importe);
        UPDATE CONTRATO SET importe =
         importe*((100-descuento)/100) WHERE CURRENT OF
         C_CONTRATO;
    END LOOP;
    FOR RegContrato IN C_CONTRATO LOOP
        DBMS_OUTPUT.PUT('ID contrato: ' ||
RegContrato.ID_contrato);
        DBMS_OUTPUT.PUT_LINE('; Precio actual: ' ||
RegContrato.importe);
    END LOOP;
    EXCEPTION
       WHEN PARAMETRO_MENOR_EXCEPTION THEN
          DBMS_OUTPUT.PUT_LINE('El descuento debe ser mayor que
          0');
        WHEN PARAMETRO_MAYOR_EXCEPTION THEN
          DBMS_OUTPUT.PUT_LINE('El descuento debe ser menor que
          100');
       WHEN OTHERS THEN
          DBMS_OUTPUT.PUT_LINE('Código: ' || SQLCODE);
END APLICAR_DESCUENTO;
/
show errors
```

--Procedimiento 5:

```
CREATE OR REPLACE PROCEDURE ACTUALIZAR_PROPIETARIO(DNI IN
PROPIETARIO.DNI_propietario%TYPE, numProp OUT NUMBER)

    IS
        CURSOR C_INMUEBLE IS
            SELECT COUNT(num_catastro)
            FROM INMUEBLE
            WHERE DNI_propietario = DNI;
        DNI_EXCEPTION EXCEPTION;
    BEGIN
        OPEN C_INMUEBLE;
        FETCH C_INMUEBLE INTO numProp;
        IF C_INMUEBLE%NOTFOUND THEN
            RAISE DNI_EXCEPTION;
        END IF;
        UPDATE PROPIETARIO SET num_propiedades = numProp
            WHERE DNI_propietario = DNI;
```

```
        CLOSE C_INMUEBLE;
        DBMS_OUTPUT.PUT_LINE('Número de propiedades: ' ||
         numProp);
    EXCEPTION
        WHEN DNI_EXCEPTION THEN
            DBMS_OUTPUT.PUT_LINE('No se han encontrado al
             propietario con DNI ' || DNI);
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Codigo: ' || SQLCODE);
            DBMS_OUTPUT.PUT_LINE('Mensaje: ' || SUBSTR(SQLERRM,
            11,100));
END ACTUALIZAR_PROPIETARIO;
/
show errors
```

--Función 1:

```
CREATE OR REPLACE FUNCTION CASAS_POR_PLANTAS(PLANTAS IN NUMBER)

    RETURN NUMBER
    IS
       numCasas NUMBER;
       SINCASAS_EXCEPTION EXCEPTION;
    BEGIN
        SELECT COUNT(*) INTO numCasas
        FROM CASA
        WHERE plantas_casa=PLANTAS;
        IF numCasas = 0 THEN
            RAISE SINCASAS_EXCEPTION;
        END IF;
        RETURN numCasas;
    EXCEPTION
        WHEN SINCASAS_EXCEPTION THEN
            DBMS_OUTPUT.PUT_LINE('No se han encontrado casas con
            ' || PLANTAS || ' plantas.');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Codigo: ' || SQLCODE);
            DBMS_OUTPUT.PUT_LINE('Mensaje: ' || SUBSTR(SQLERRM,
            11,100));
END CASAS_POR_PLANTAS;
/
show errors
```

--Función 2:

```
CREATE OR REPLACE FUNCTION COUNT_ACOND(acond IN
LOCAL_COMERCIAL.ACOND_LOCAL%TYPE)

    RETURN NUMBER
    IS
    toRet NUMBER;
    E_NOACOND EXCEPTION;
    BEGIN
        SELECT COUNT(*) INTO toRet
        FROM LOCAL_COMERCIAL
        WHERE ACOND_LOCAL = acond;
        IF(toRet = 0)THEN
            RAISE E_NOACOND;
        END IF;
        RETURN toRet;
    EXCEPTION
        WHEN E_NOACOND THEN
            DBMS_OUTPUT.PUT_LINE('Excepción: No coincide ningún
            acondicionamiento');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Código: ' || SQLCODE);
END COUNT_ACOND;
/
show errors
```

--Función 3:

```
CREATE OR REPLACE FUNCTION AGENTE_POR_ZONA(zona IN VARCHAR)

    RETURN NUMBER
    IS
        num_agentes_zona NUMBER;
        NO_AGENTES_EXCEPTION EXCEPTION;
    BEGIN
        SELECT COUNT(*) INTO num_agentes_zona
        FROM AGENTE
        WHERE zona_ciudad = zona;
        IF num_agentes_zona = 0 THEN
                RAISE NO_AGENTES_EXCEPTION;
        END IF;
        RETURN num_agentes_zona;
    EXCEPTION
        WHEN NO_AGENTES_EXCEPTION THEN
            DBMS_OUTPUT.PUT_LINE('No se han encontrado agentes
            en la zona indicada (' || zona || ')');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Codigo: ' || SQLCODE);
            DBMS_OUTPUT.PUT_LINE('Mensaje: ' || SUBSTR(SQLERRM,
            11,100));
END AGENTE_POR_ZONA;
/
show errors
```

--Función 4:

```
CREATE OR REPLACE FUNCTION NUM_VENTAS_FECHA(fecha_ini IN DATE,
fecha_fin IN DATE)
RETURN NUMBER
IS
    BADFORMAT_EXCEPTION EXCEPTION;
    num_ventas NUMBER;
BEGIN
    SELECT COUNT(*) INTO num_ventas
    FROM FIRMA
    WHERE fecha_firma>=fecha_ini AND fecha_firma<=fecha_fin;
    RETURN num_ventas;
    EXCEPTION
        WHEN BADFORMAT_EXCEPTION THEN
            DBMS_OUTPUT.PUT_LINE('El formato de fecha es
            incorrecto');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Codigo: ' || SQLCODE);
            DBMS_OUTPUT.PUT_LINE('Mensaje: ' || SUBSTR(SQLERRM,
            11,100));
END NUM_VENTAS_FECHA;
/
show errors
```

--Función 5:

```
/*
CREATE OR REPLACE FUNCTION DEVOLVER_FECHA
RETURN TIME
IS
BEGIN
    SELECT CONVERT (date, SYSDATETIME())
    ,CONVERT (date, SYSDATETIMEOFFSET())
    ,CONVERT (date, SYSUTCDATETIME())
    ,CONVERT (date, CURRENT_TIMESTAMP)
    ,CONVERT (date, GETDATE())
    ,CONVERT (date, GETUTCDATE()) INTO fecha_hoy;
    RETURN fecha_hoy;
END DEVOLVER_FECHA;
/
show errors

*/
```

SENTENCIAS DE DEFINICIÓN DE BLOQUES DE PRUEBAS

```
SET SERVEROUTPUT ON
DECLARE
      toRet NUMBER;
BEGIN
      DBMS_OUTPUT.NEW_LINE;
```

--procedimientos

```
   BEGIN
-- PROPIETARIOS
      DBMS_OUTPUT.PUT_LINE('======>INICIO PROCEDIMIENTO:
      PROPIETARIOS');
      PROPIETARIOS(3, toRet);
      DBMS_OUTPUT.PUT_LINE('Numero de propietarios con 3
      inmuebles: ' || toRet);
      DBMS_OUTPUT.PUT_LINE('======>FIN PROCEDIMIENTO:
      PROPIETARIOS');
      DBMS_OUTPUT.NEW_LINE;
   EXCEPTION
      WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('[EXCEPCIÓN]');
         DBMS_OUTPUT.PUT_LINE('[Código]: ' || SQLCODE);
         DBMS_OUTPUT.PUT_LINE('[Mensaje]: ' || SUBSTR(SQLERRM,
         11, 100));
   END;

   BEGIN
-- ACOND_ACT_LOCAL
      DBMS_OUTPUT.PUT_LINE('======>INICIO PROCEDIMIENTO:
      ACOND_ACT_LOCAL');
      ACOND_ACT_LOCAL('VACIO', 'ALMACEN', toRet);
      DBMS_OUTPUT.PUT_LINE('Numero de locales: ' || toRet);
      DBMS_OUTPUT.PUT_LINE('======>FIN PROCEDIMIENTO:
      ACOND_ACT_LOCAL');
      DBMS_OUTPUT.NEW_LINE;
   EXCEPTION
      WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('[EXCEPCIÓN]');
         DBMS_OUTPUT.PUT_LINE('[Código]: ' || SQLCODE);
         DBMS_OUTPUT.PUT_LINE('[Mensaje]: ' || SUBSTR(SQLERRM,
         11, 100));
   END;

   BEGIN
-- EMP_BAJAR_SUELDO
      DBMS_OUTPUT.PUT_LINE('======>INICIO PROCEDIMIENTO:
      EMP_BAJAR_SUELDO');
      EMP_BAJAR_SUELDO('temporal', 10);
      DBMS_OUTPUT.PUT_LINE('======>FIN PROCEDIMIENTO:
      EMP_BAJAR_SUELDO');
      DBMS_OUTPUT.NEW_LINE;
```

```
   EXCEPTION
      WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('[EXCEPCIÓN]');
         DBMS_OUTPUT.PUT_LINE('[Código]: ' || SQLCODE);
         DBMS_OUTPUT.PUT_LINE('[Mensaje]: ' || SUBSTR(SQLERRM,
         11, 100));
   END;

   BEGIN
-- APLICAR_DESCUENTO
      DBMS_OUTPUT.PUT_LINE('======>INICIO PROCEDIMIENTO:
      APLICAR_DESCUENTO');
      APLICAR_DESCUENTO(10);
      DBMS_OUTPUT.PUT_LINE('======>FIN PROCEDIMIENTO:
      APLICAR_DESCUENTO');
      DBMS_OUTPUT.NEW_LINE;
   EXCEPTION
      WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('[EXCEPCIÓN]');
         DBMS_OUTPUT.PUT_LINE('[Código]: ' || SQLCODE);
         DBMS_OUTPUT.PUT_LINE('[Mensaje]: ' || SUBSTR(SQLERRM,
         11, 100));
   END;

   BEGIN
-- ACTUALIZAR_PROPIETARIO
      DBMS_OUTPUT.PUT_LINE('======>INICIO PROCEDIMIENTO:
      ACTUALIZAR_PROPIETARIO');
      ACTUALIZAR_PROPIETARIO('00734219J', toRet);
      DBMS_OUTPUT.PUT_LINE('Numero de propiedades del
      propietario 00734219J: ' || toRet);
      DBMS_OUTPUT.PUT_LINE('======>FIN PROCEDIMIENTO:
      ACTUALIZAR_PROPIETARIO');
      DBMS_OUTPUT.NEW_LINE;
   EXCEPTION
      WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('[EXCEPCIÓN]');
         DBMS_OUTPUT.PUT_LINE('[Código]: ' || SQLCODE);
         DBMS_OUTPUT.PUT_LINE('[Mensaje]: ' || SUBSTR(SQLERRM,
         11, 100));
   END;
```

--funciones

```
   BEGIN
-- CASAS_POR_PLANTAS
      DBMS_OUTPUT.PUT_LINE('======>INICIO FUNCIÓN:
      CASAS_POR_PLANTAS');
      toRet := CASAS_POR_PLANTAS(2);
      DBMS_OUTPUT.PUT_LINE('Casas con 2 plantas: ' || toRet);
      DBMS_OUTPUT.PUT_LINE('======>FIN FUNCIÓN:
      CASAS_POR_PLANTAS');
      DBMS_OUTPUT.NEW_LINE;
   EXCEPTION
      WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('[EXCEPCIÓN]');
```

```
      DBMS_OUTPUT.PUT_LINE('[Código]: ' || SQLCODE);
      DBMS_OUTPUT.PUT_LINE('[Mensaje]: ' || SUBSTR(SQLERRM,
      11, 100));
END;

BEGIN
-- COUNT_ACOND
   DBMS_OUTPUT.PUT_LINE('======>INICIO FUNCIÓN:
   COUNT_ACOND');
   toRet := COUNT_ACOND('VACIO');
   DBMS_OUTPUT.PUT_LINE('Locales vacios: ' || toRet);
   DBMS_OUTPUT.PUT_LINE('======>FIN FUNCIÓN: COUNT_ACOND');
   DBMS_OUTPUT.NEW_LINE;
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('[EXCEPCIÓN]');
      DBMS_OUTPUT.PUT_LINE('[Código]: ' || SQLCODE);
      DBMS_OUTPUT.PUT_LINE('[Mensaje]: ' || SUBSTR(SQLERRM,
      11, 100));
END;

BEGIN
-- AGENTE_POR_ZONA
   DBMS_OUTPUT.PUT_LINE('======>INICIO FUNCIÓN:
   AGENTE_POR_ZONA');
   toRet := AGENTE_POR_ZONA('centro');
   DBMS_OUTPUT.PUT_LINE('Agentes asignados a la zona centro:
   ' || toRet);
   DBMS_OUTPUT.PUT_LINE('======>FIN FUNCIÓN:
   AGENTE_POR_ZONA');
   DBMS_OUTPUT.NEW_LINE;
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('[EXCEPCIÓN]');
      DBMS_OUTPUT.PUT_LINE('[Código]: ' || SQLCODE);
      DBMS_OUTPUT.PUT_LINE('[Mensaje]: ' || SUBSTR(SQLERRM,
      11, 100));
END;

BEGIN
-- NUM_VENTAS_FECHA
   DBMS_OUTPUT.PUT_LINE('======>INICIO FUNCIÓN:
   NUM_VENTAS_FECHA');
   toRet := NUM_VENTAS_FECHA('2020-01-01','2020-10-01');
   DBMS_OUTPUT.PUT_LINE('Número de ventas entre enero y
   octubre de 2020: ' || toRet);
   DBMS_OUTPUT.PUT_LINE('======>FIN FUNCIÓN:
   NUM_VENTAS_FECHA');
   DBMS_OUTPUT.NEW_LINE;
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('[EXCEPCIÓN]');
      DBMS_OUTPUT.PUT_LINE('[Código]: ' || SQLCODE);
      DBMS_OUTPUT.PUT_LINE('[Mensaje]: ' || SUBSTR(SQLERRM,
      11, 100));
END;
```

```
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('[EXCEPCIÓN NO TRATADA EN EL BLOQUE
       PRINCIPAL]');
      DBMS_OUTPUT.PUT_LINE('[Código]: ' || SQLCODE);
      DBMS_OUTPUT.PUT_LINE('[Mensaje]: ' || SUBSTR(SQLERRM, 11,
       100));
END;
/
```

HOJA DE FIRMAS

DNI: 77460353X **NOMBRE: Gómez Dopazo, Rubén**

DNI: 34629985N **NOMBRE: Pérez Gómez, Cristian**

DNI: 34275678C **NOMBRE: Pérez Iglesia, Luis**

DNI: 76734153N **NOMBRE: Gómez González, Alejandro**