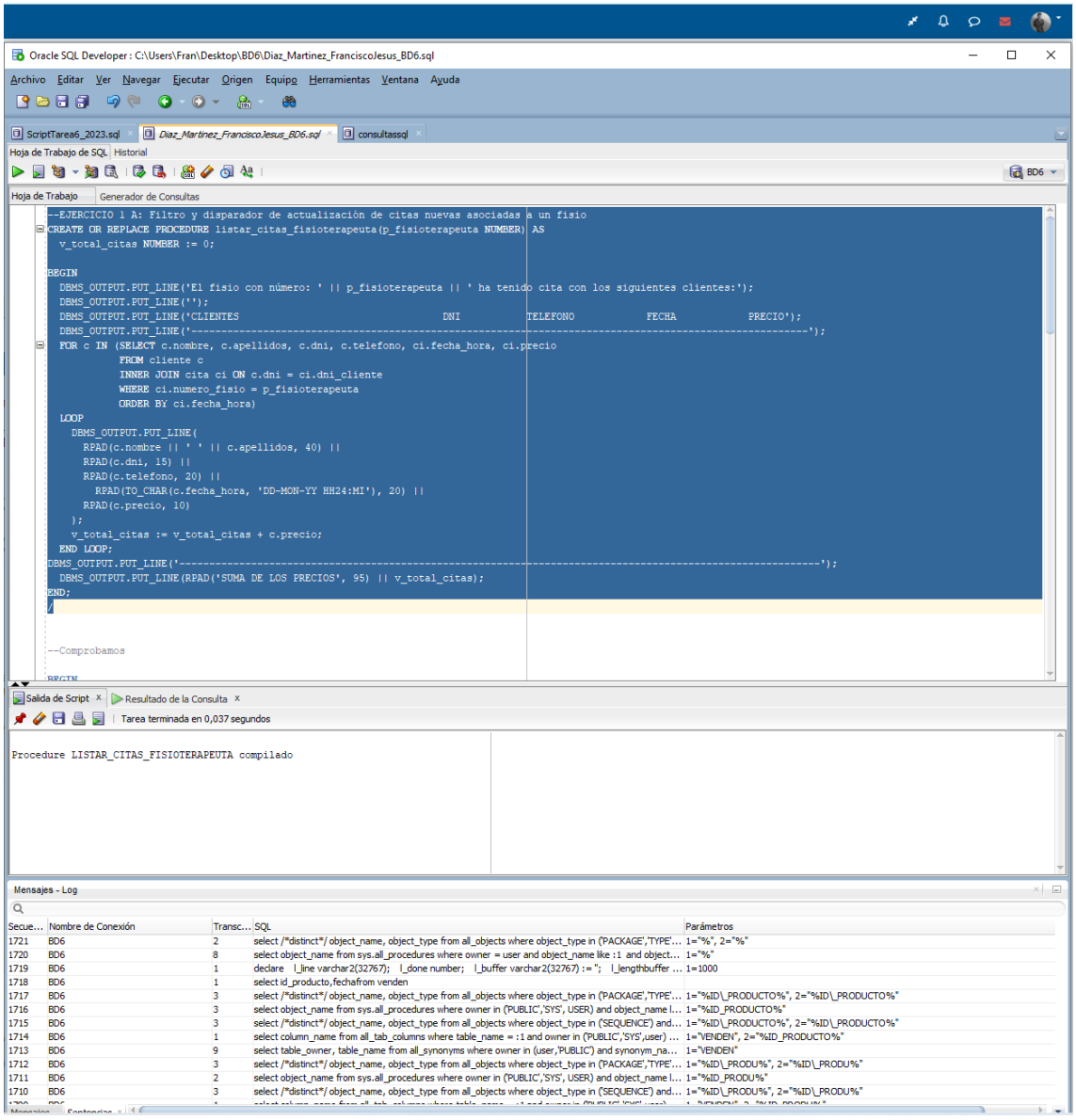
BD:ACTIVIDAD 6

Francisco Jesús Díaz Martínez

IES AGUADULCE DAM 1

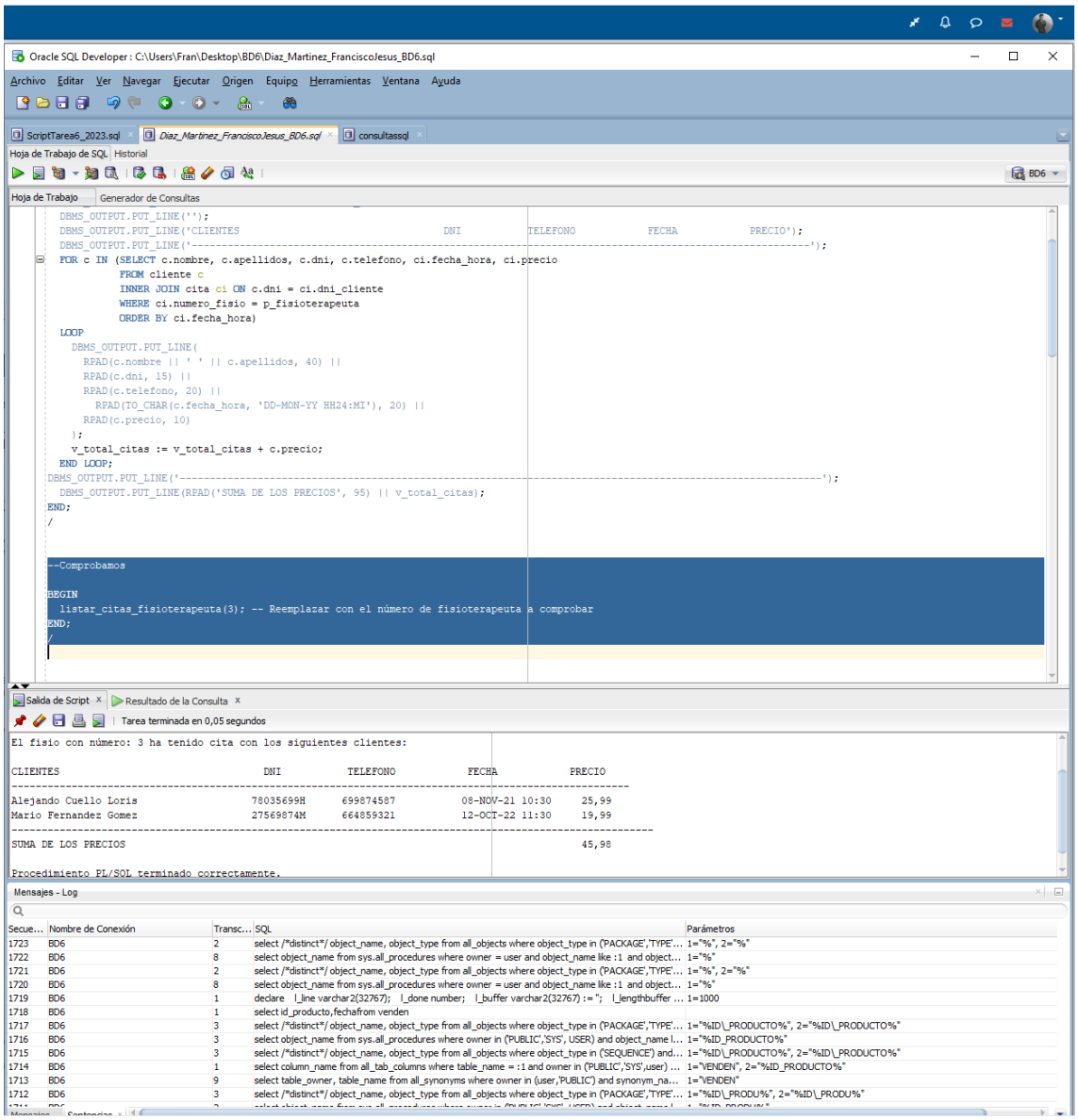
ACTIVIDAD 1:

En esta primera actividad nos pedían que confeccionásemos un script para recoger los datos de los clientes de un fisio y que los ordenásemos en una tabla con un determinado formato, por último, debíamos facilitar la suma de todos los pagos de los clientes por el servicio.



El código ha sido compilado con éxito como se puede ver en la pestaña de output. Para ello debemos de seleccionar el código por completo y hacer clic sobre el botón verde (ejecutar).

También la primera vez que introducimos la base de datos que nos proporcionan en las actividades deberemos de ejecutarla, porque cuando confeccionamos los scripts nos debemos de basar en ella para su correcta implementación y funcionamiento.

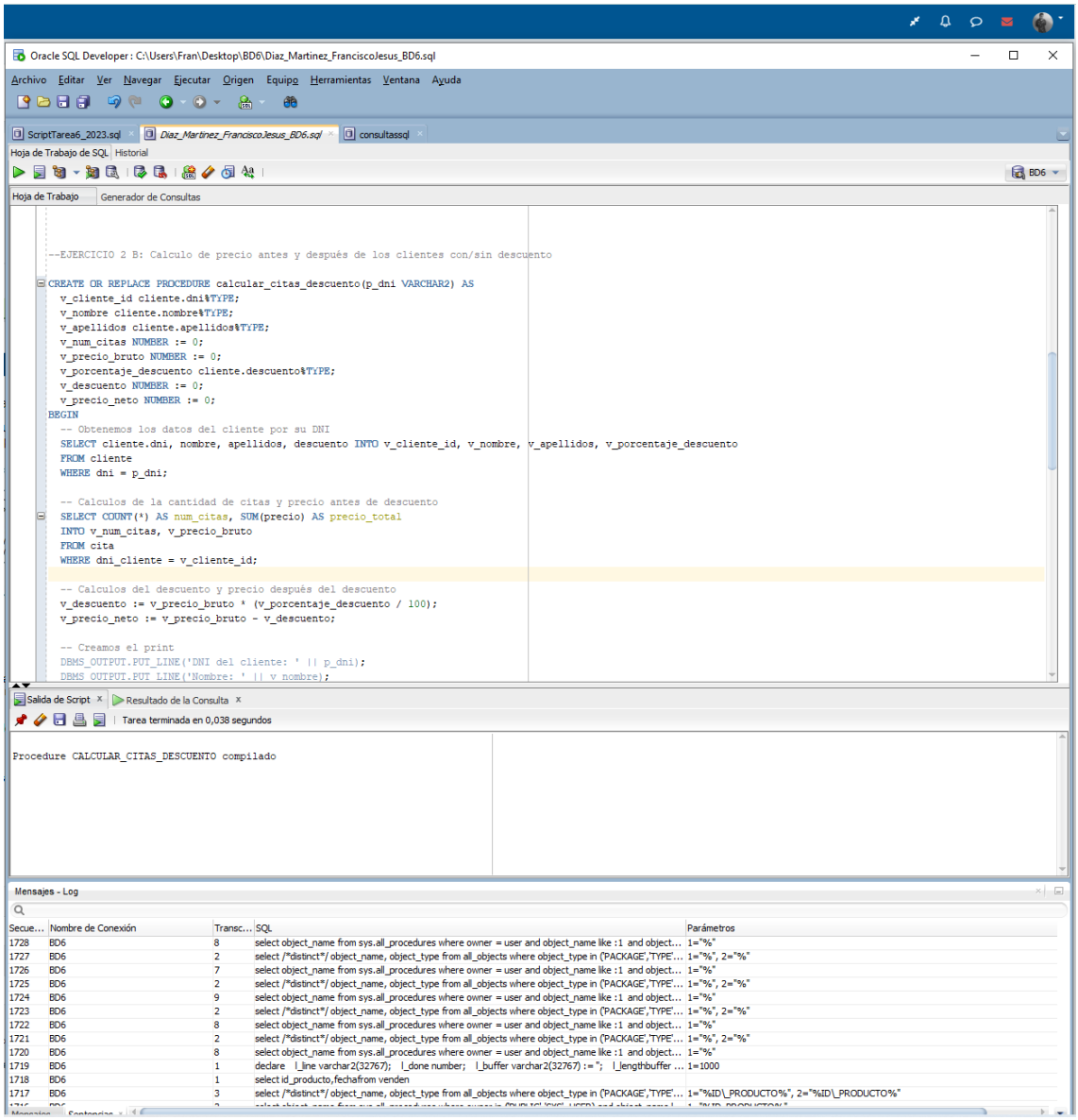


Aquí tenemos la comprobación y salida del script, introducimos en el parámetro de constructor el número de fisioterapeuta que queremos comprobar.

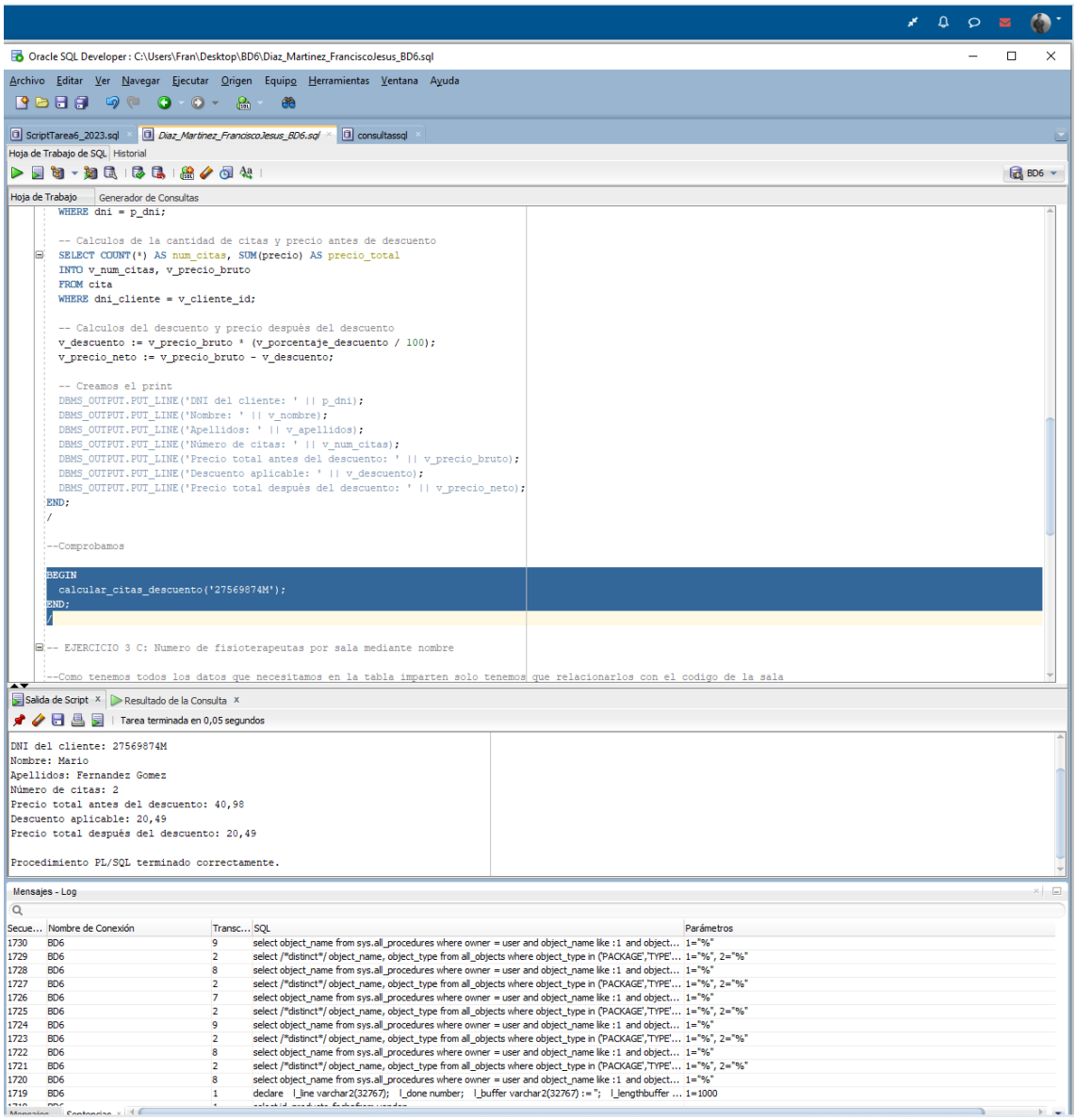
Hay pocas aclaraciones sobre el script, hemos usado las tablas cliente y citas donde están recogidos todos los datos del cliente, así como la identificación del fisio y fecha de la cita y precio.

Luego hemos hecho un “inner join” entre las identificaciones de los clientes en ambas tablas para obtener los datos y poder obtener la salida deseada. Por último, hemos hecho un operatorio simple de suma para obtener el precio total.

ACTIVIDAD 2:



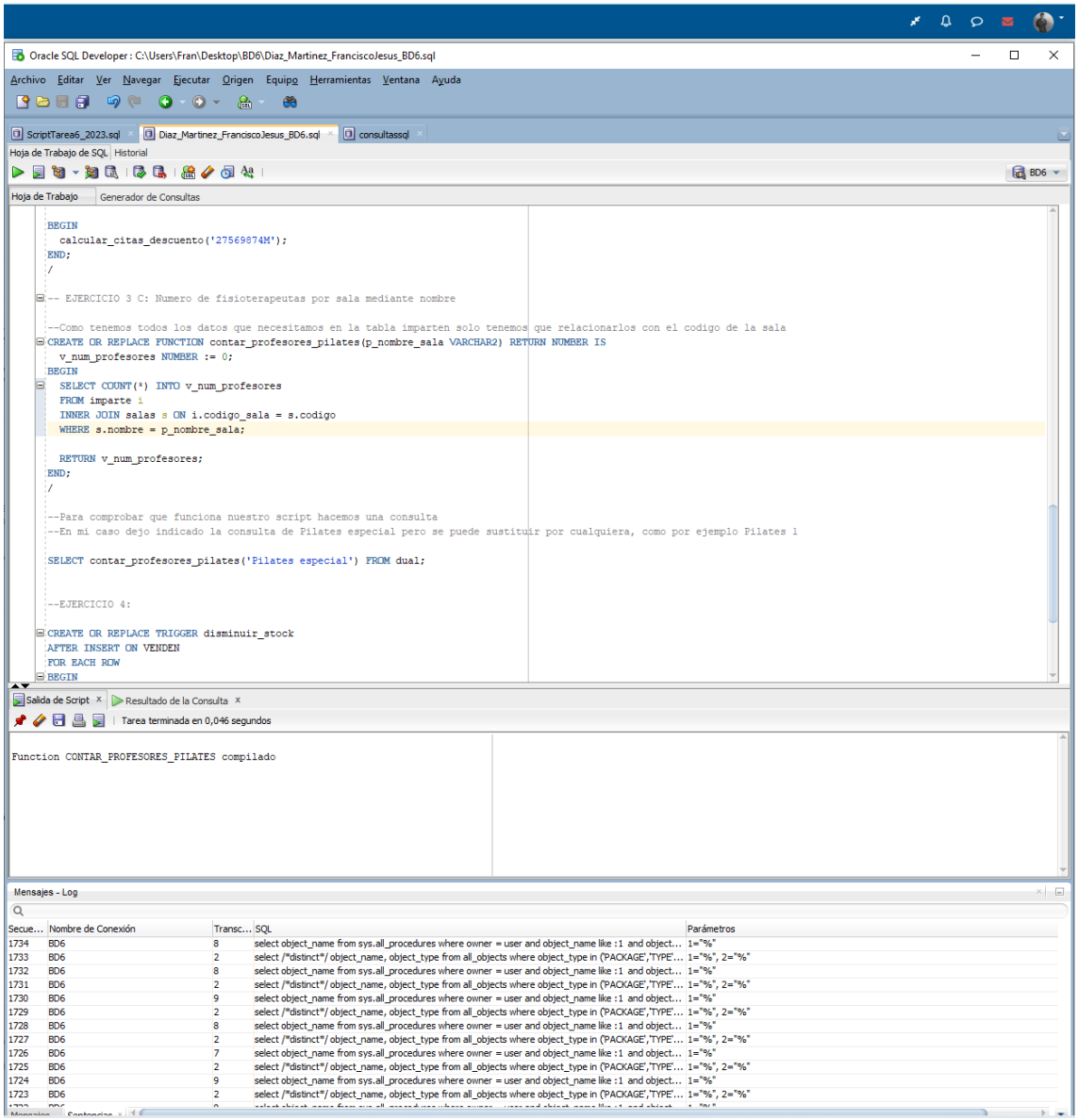
Aquí podemos observar que la compilación ha sido correcta. Hemos usado todos los datos de la tabla cliente y citas. Nos solicitaban obtener el número de las citas, el monto total de su coste, el descuento que podría tener ese cliente, y el precio después de aplicarla. Para ello primero hemos obtenido todos los datos de las tablas, y posteriormente hemos procedido a hacer los cálculos.



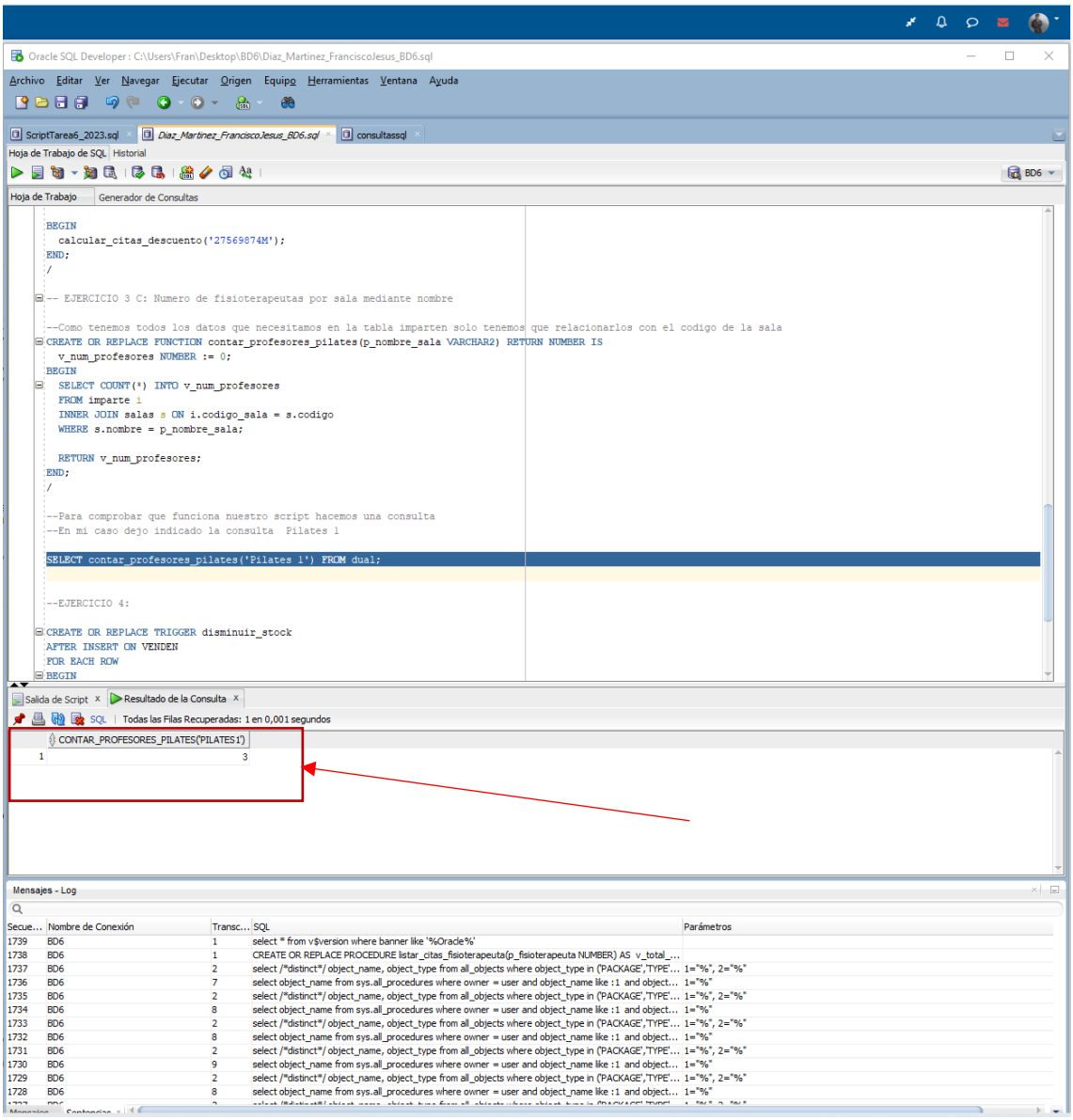
Para comprobar el funcionamiento de mi script, ejecuto una llamada al DNI de un cliente que es el mismo que se expone como ejemplo en la actividad.

Como se puede ver el output es como el que se muestra en el ejercicio, en mi base de datos solo tiene 2 citas, cuyo valor es de 40.98, después de calcular el descuento, se resta y se muestra el total después del descuento.

ACTIVIDAD 3

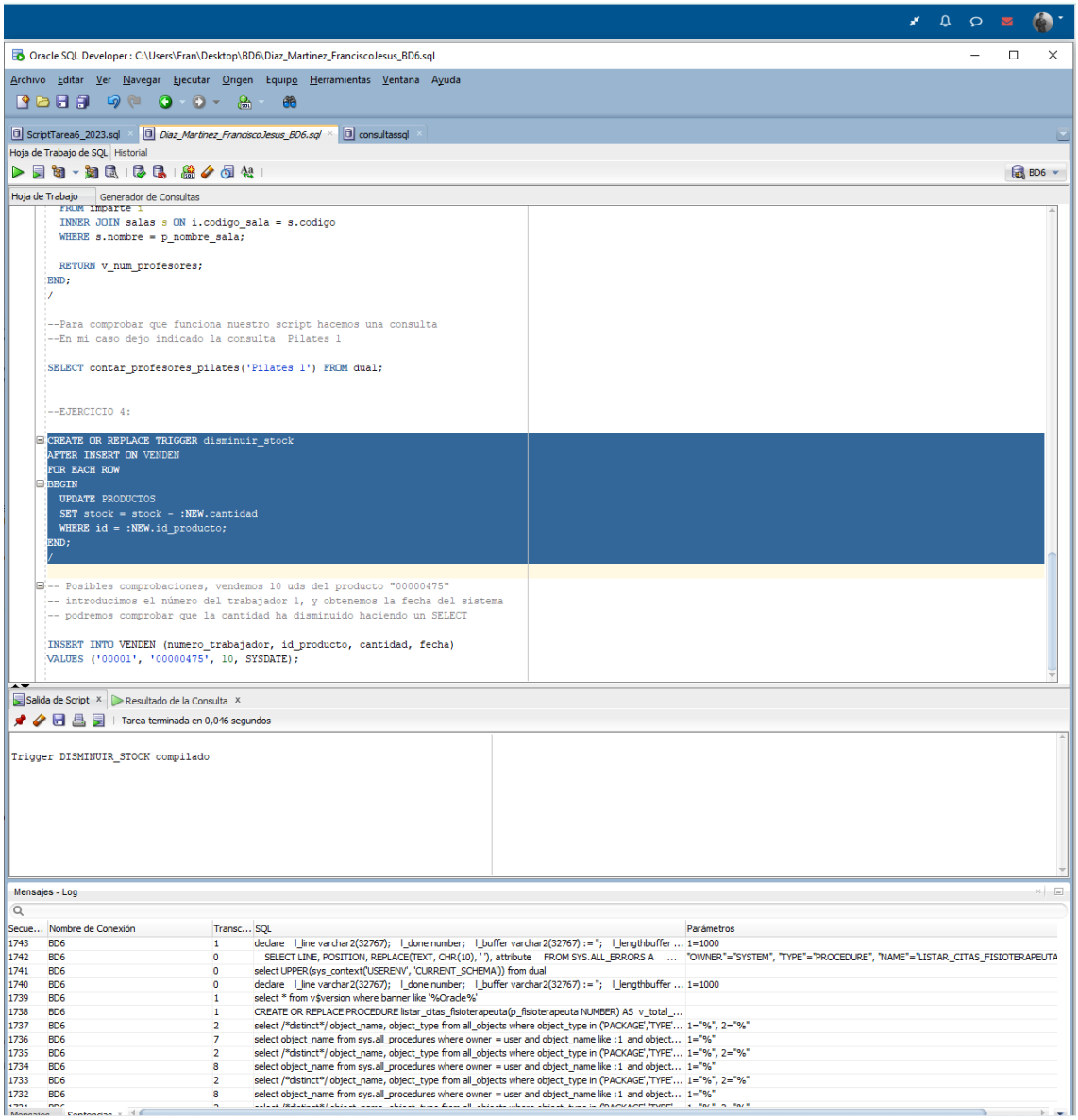


Aquí tenemos la compilación con éxito del conteo de los profesores de pilates según la sala que se compruebe. Este es uno de los apartados más cortos porque la tabla “imparte” recoge la mayoría de los datos y solo tenemos que usar la tabla “salas” para hacer un INNER JOIN entre los códigos de identificación. Hecho esto, definimos lo que nos devolverá: el número de profesores que han usado la sala “X”.



Para la comprobación, debemos de hacer un select de nuestra función para ello usamos (dual) que se usa para acceder a tablas ficticias, indicamos el nombre de una de las salas de pilates, como por ejemplo “Pilates 1” y como vemos en el output nos indica que hay 3 profesores que han usado esa sala en concreto.

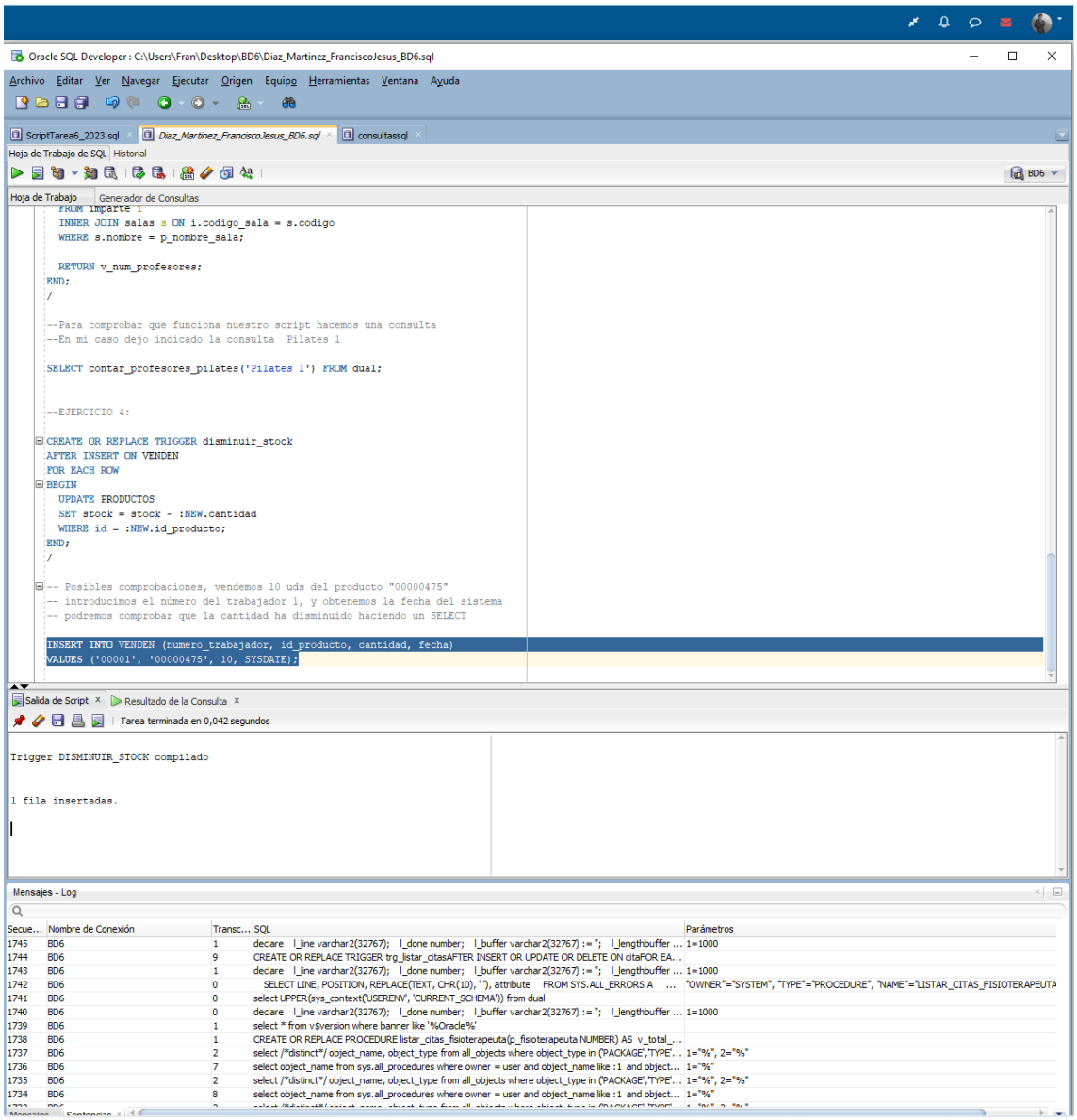
ACTIVIDAD 4



Por último, compilamos el disparador para que cuando se venda un producto, se disminuya la cantidad del mismo.

Lo que estamos indicando con el script es que después de cada venta (tabla “VENDEN”) se actualice en la tabla productos los valores de id y stock.

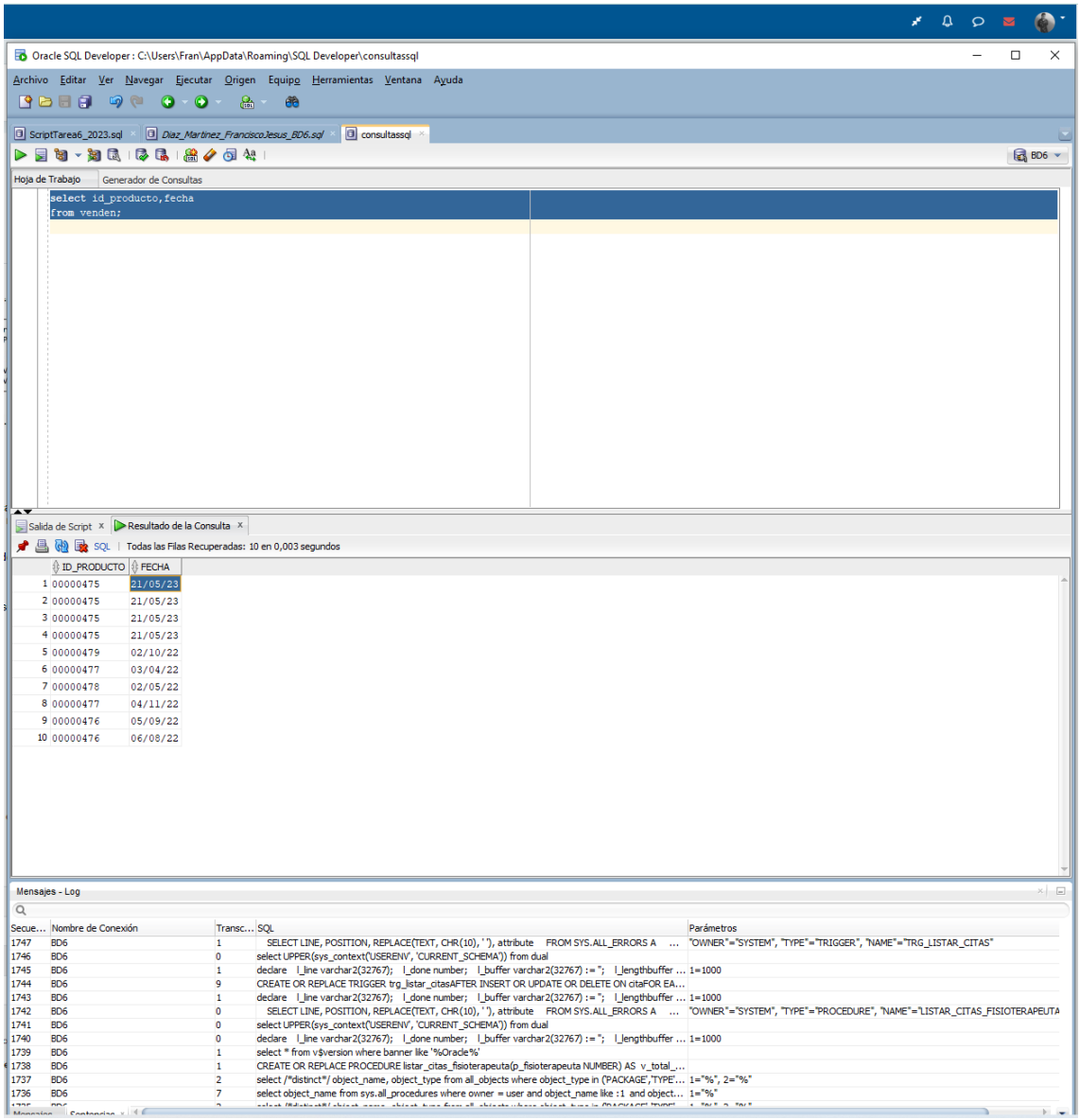
Para ello solo citamos los valores originales e indicamos una nueva identificación para los nuevos.



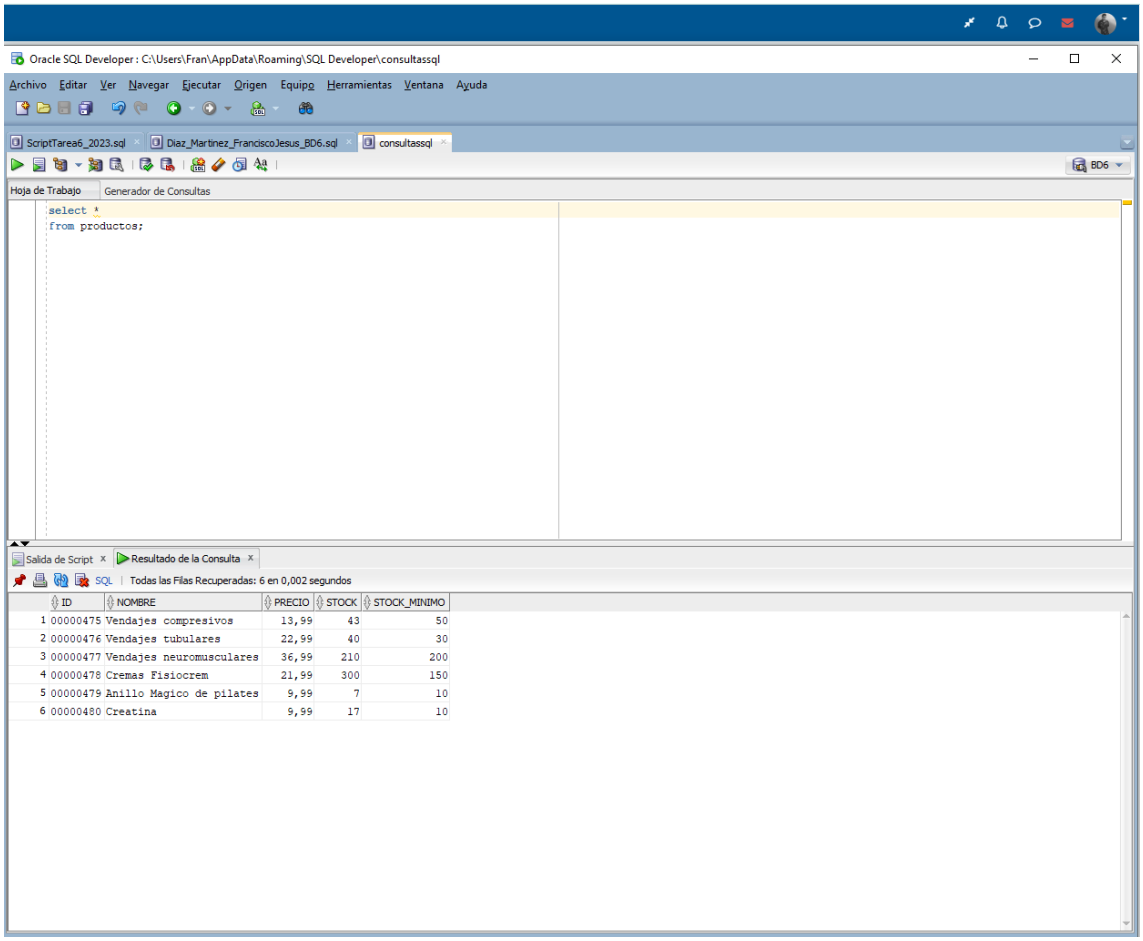
Para la comprobación de este gatillo, lo que podemos hacer es crear una venta en la base de datos para ver como se modifican los valores. Para ello hacemos un “INSERT” en la tabla “VENDEN” con los constructores de número de trabajador, id del producto, cantidad y fecha.

En mi caso he usado todos los valores que están registrados en nuestra base de datos, salvo la fecha y la cantidad, las cuales he indicado de forma arbitraria con la fecha de mi sistema.

Ahora al ejecutar la inserción, podemos ver como el output no nos muestra más que una línea indicando que se ha insertado con éxito. A continuación, adjunto dos capturas con consultas para comprobar que ha surtido efecto.



Primero hacemos una consulta de id y fecha en la tabla venden. Como podemos observar, hay 4 nuevas entradas de la fecha en la que realicé este trabajo. Hay 4 porque hice varias pruebas para comprobar que funciona correctamente, pero como se puede observar todas corresponden al mismo producto y a la fecha del sistema.



Por último a modo de confirmación, hacemos un select de todos los productos, y comprobamos que el stock de “Vendajes compresivos” ha disminuido en 40 unidades, respecto a la base de datos que se proporcionó originalmente.

A continuación el código generado:

```
--EJERCICIO 1 A: Filtro y disparador de actualización de citas nuevas
asociadas a un fisio
CREATE OR REPLACE PROCEDURE
listar_citas_fisioterapeuta(p_fisioterapeuta NUMBER) AS
  v_total_citas NUMBER := 0;

BEGIN
  DBMS_OUTPUT.PUT_LINE('El fisio con número: ' || p_fisioterapeuta ||
' ha tenido cita con los siguientes clientes:');
  DBMS_OUTPUT.PUT_LINE('');
  DBMS_OUTPUT.PUT_LINE('CLIENTES                                    DNI
TELEFONO            FECHA            PRECIO');
  DBMS_OUTPUT.PUT_LINE('-----------------------------------------------
------------------------------------------------------');
  FOR c IN (SELECT c.nombre, c.apellidos, c.dni, c.telefono,
ci.fecha_hora, ci.precio
            FROM cliente c
            INNER JOIN cita ci ON c.dni = ci.dni_cliente
            WHERE ci.numero_fisio = p_fisioterapeuta
            ORDER BY ci.fecha_hora)
  LOOP
    DBMS_OUTPUT.PUT_LINE(
      RPAD(c.nombre || ' ' || c.apellidos, 40) ||
      RPAD(c.dni, 15) ||
      RPAD(c.telefono, 20) ||
        RPAD(TO_CHAR(c.fecha_hora, 'DD-MON-YY HH24:MI'), 20) ||
      RPAD(c.precio, 10)
    );
    v_total_citas := v_total_citas + c.precio;
  END LOOP;
DBMS_OUTPUT.PUT_LINE('-----------------------------------------------
------------------------------------------------------');
  DBMS_OUTPUT.PUT_LINE(RPAD('SUMA DE LOS PRECIOS', 95) ||
v_total_citas);
END;
/


--Comprobamos

BEGIN
  listar_citas_fisioterapeuta(3); -- Reemplazar con el número de
fisioterapeuta a comprobar
END;
/
```

```
--EJERCICIO 2 B: Calculo de precio antes y después de los clientes
con/sin descuento

CREATE OR REPLACE PROCEDURE calcular_citas_descuento(p_dni VARCHAR2)
AS
  v_cliente_id cliente.dni%TYPE;
  v_nombre cliente.nombre%TYPE;
  v_apellidos cliente.apellidos%TYPE;
  v_num_citas NUMBER := 0;
  v_precio_bruto NUMBER := 0;
  v_porcentaje_descuento cliente.descuento%TYPE;
  v_descuento NUMBER := 0;
  v_precio_neto NUMBER := 0;
BEGIN
  -- Obtenemos los datos del cliente por su DNI
  SELECT cliente.dni, nombre, apellidos, descuento INTO v_cliente_id,
v_nombre, v_apellidos, v_porcentaje_descuento
  FROM cliente
  WHERE dni = p_dni;

  -- Calculos de la cantidad de citas y precio antes de descuento
  SELECT COUNT(*) AS num_citas, SUM(precio) AS precio_total
  INTO v_num_citas, v_precio_bruto
  FROM cita
  WHERE dni_cliente = v_cliente_id;

  -- Calculos del descuento y precio después del descuento
  v_descuento := v_precio_bruto * (v_porcentaje_descuento / 100);
  v_precio_neto := v_precio_bruto - v_descuento;

  -- Creamos el print
  DBMS_OUTPUT.PUT_LINE('DNI del cliente: ' || p_dni);
  DBMS_OUTPUT.PUT_LINE('Nombre: ' || v_nombre);
  DBMS_OUTPUT.PUT_LINE('Apellidos: ' || v_apellidos);
  DBMS_OUTPUT.PUT_LINE('Número de citas: ' || v_num_citas);
  DBMS_OUTPUT.PUT_LINE('Precio total antes del descuento: ' ||
v_precio_bruto);
  DBMS_OUTPUT.PUT_LINE('Descuento aplicable: ' || v_descuento);
  DBMS_OUTPUT.PUT_LINE('Precio total después del descuento: ' ||
v_precio_neto);
END;
/

--Comprobamos

BEGIN
  calcular_citas_descuento('27569874M');
END;
/
```

```
-- EJERCICIO 3 C: Numero de fisioterapeutas por sala mediante nombre

--Como tenemos todos los datos que necesitamos en la tabla imparten
solo tenemos que relacionarlos con el codigo de la sala
CREATE OR REPLACE FUNCTION contar_profesores_pilates(p_nombre_sala
VARCHAR2) RETURN NUMBER IS
  v_num_profesores NUMBER := 0;
BEGIN
  SELECT COUNT(*) INTO v_num_profesores
  FROM imparte i
  INNER JOIN salas s ON i.codigo_sala = s.codigo
  WHERE s.nombre = p_nombre_sala;

  RETURN v_num_profesores;
END;
/

--Para comprobar que funciona nuestro script hacemos una consulta
--En mi caso dejo indicado la consulta  Pilates 1

SELECT contar_profesores_pilates('Pilates 1') FROM dual;


--EJERCICIO 4:

CREATE OR REPLACE TRIGGER disminuir_stock
AFTER INSERT ON VENDEN
FOR EACH ROW
BEGIN
  UPDATE PRODUCTOS
  SET stock = stock - :NEW.cantidad
  WHERE id = :NEW.id_producto;
END;
/

-- Posibles comprobaciones, vendemos 10 uds del producto "00000475"
-- introducimos el número del trabajador 1, y obtenemos la fecha del
sistema
-- podremos comprobar que la cantidad ha disminuido haciendo un SELECT

INSERT INTO VENDEN (numero_trabajador, id_producto, cantidad, fecha)
VALUES ('00001', '00000475', 10, SYSDATE);
```