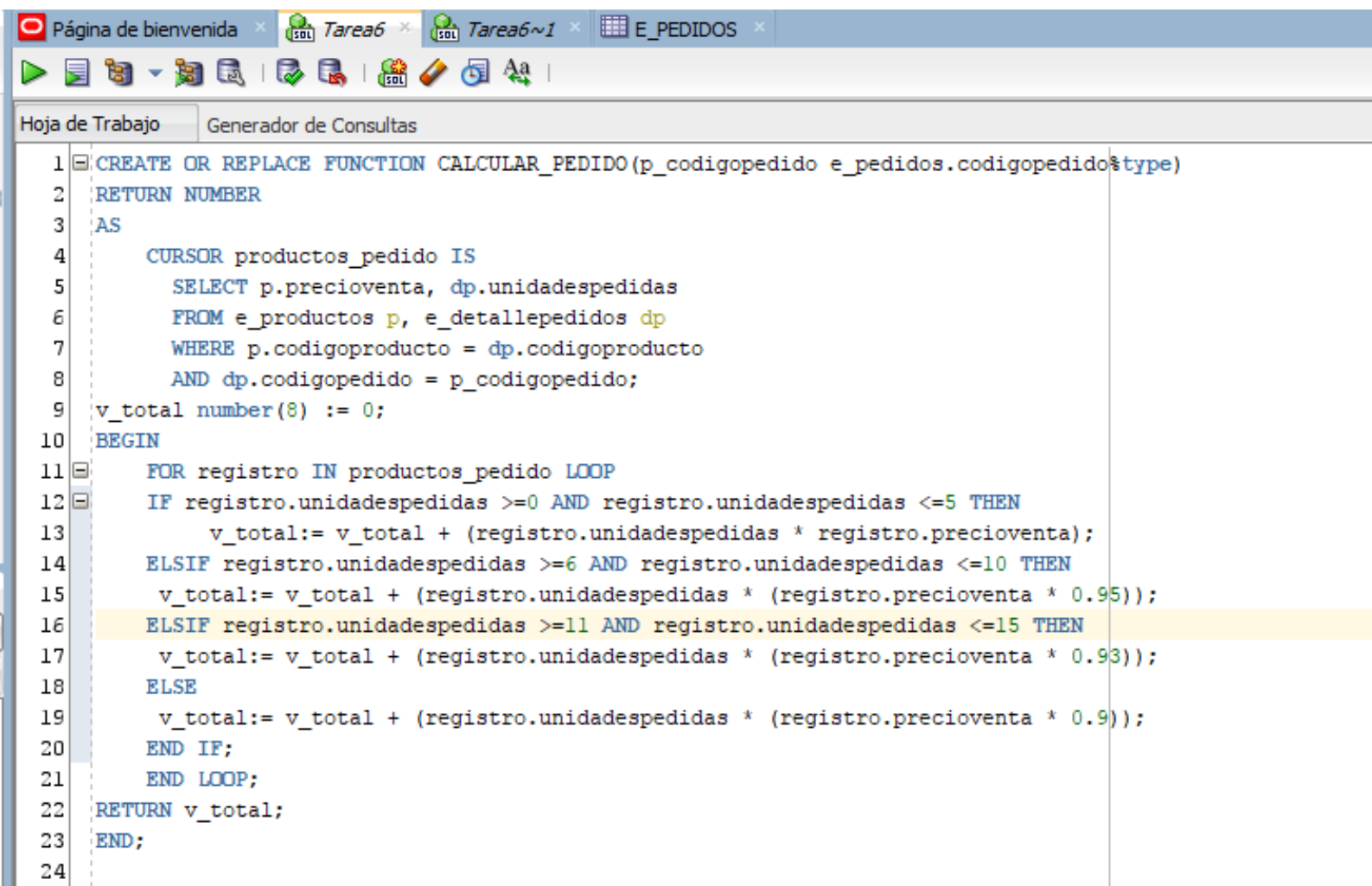


TAREA PARA BD06.**ACTIVIDAD 1.**

Realizaremos una **función** llamada **CALCULAR_PEDIDO** a la que le daremos un código de pedido como entrada y nos devuelva como salida el total de ese pedido. A cada producto del pedido se le realizará un descuento sobre el precio de venta que dependerá del nº de unidades pedidas:

- Si el nº de unidades está entre 0 y 5: no habrá descuento
- Si el nº de unidades pedidas está entre 6 y 10 el descuento es del 5%
- Si el nº de unidades pedidas está entre 11 y 15 el descuento es del 7%
- Si el nº de unidades pedidas es >15 el descuento será del 10%.

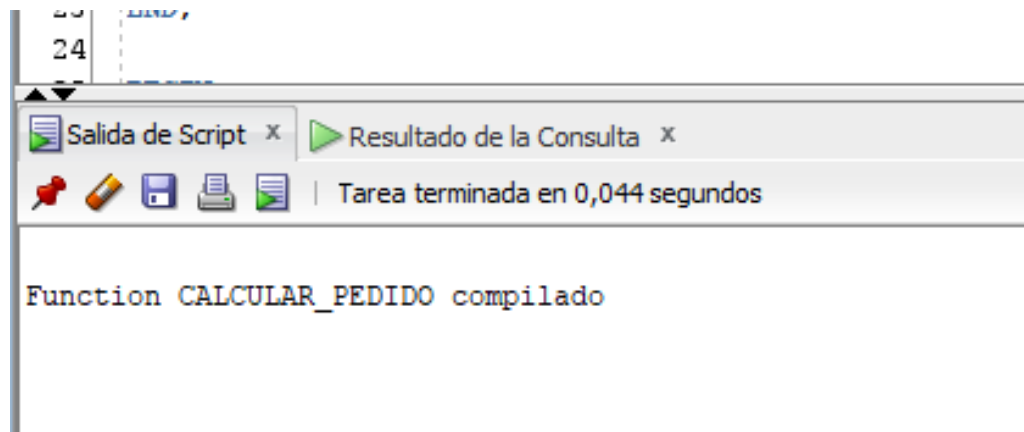
El total de un producto será el resultado de multiplicar el precio de venta (con el descuento realizado) por el nº de unidades pedidas.



The screenshot shows a SQL IDE window with the title bar containing 'Página de bienvenida', 'Tarea6', 'Tarea6~1', and 'E_PEDIDOS'. The main window has two tabs: 'Hoja de Trabajo' and 'Generador de Consultas'. The 'Generador de Consultas' tab is active, displaying a SQL script for creating a function named 'CALCULAR_PEDIDO'. The script is as follows:

```
1 CREATE OR REPLACE FUNCTION CALCULAR_PEDIDO(p_codigopedido e_pedidos.codigopedido%type)
2 RETURN NUMBER
3 AS
4     CURSOR productos_pedido IS
5         SELECT p.precioventa, dp.unidadespedidas
6         FROM e_productos p, e_detallepedidos dp
7         WHERE p.codigoproducto = dp.codigoproducto
8         AND dp.codigopedido = p_codigopedido;
9     v_total number(8) := 0;
10 BEGIN
11     FOR registro IN productos_pedido LOOP
12         IF registro.unidadespedidas >=0 AND registro.unidadespedidas <=5 THEN
13             v_total:= v_total + (registro.unidadespedidas * registro.precioventa);
14         ELSIF registro.unidadespedidas >=6 AND registro.unidadespedidas <=10 THEN
15             v_total:= v_total + (registro.unidadespedidas * (registro.precioventa * 0.95));
16         ELSIF registro.unidadespedidas >=11 AND registro.unidadespedidas <=15 THEN
17             v_total:= v_total + (registro.unidadespedidas * (registro.precioventa * 0.93));
18         ELSE
19             v_total:= v_total + (registro.unidadespedidas * (registro.precioventa * 0.9));
20         END IF;
21     END LOOP;
22     RETURN v_total;
23 END;
```

Al ejecutar la función nos dice que se compila correctamente:



Para añadir el código del pedido como entrada añado un DBMS_OUTPUT y le paso como parámetro un 1.

```

        v_total:= v_total + (registro.unidadespedidas * (registro.precioventa * 0.9)),
    ELSE
        v_total:= v_total + (registro.unidadespedidas * (registro.precioventa * 0.9));
    END IF;
    END LOOP;
RETURN v_total;
END;

BEGIN
    DBMS_OUTPUT.PUT_LINE(CALCULAR_PEDIDO(1));
END;

```

El pedido 1 se corresponde con la siguiente tabla

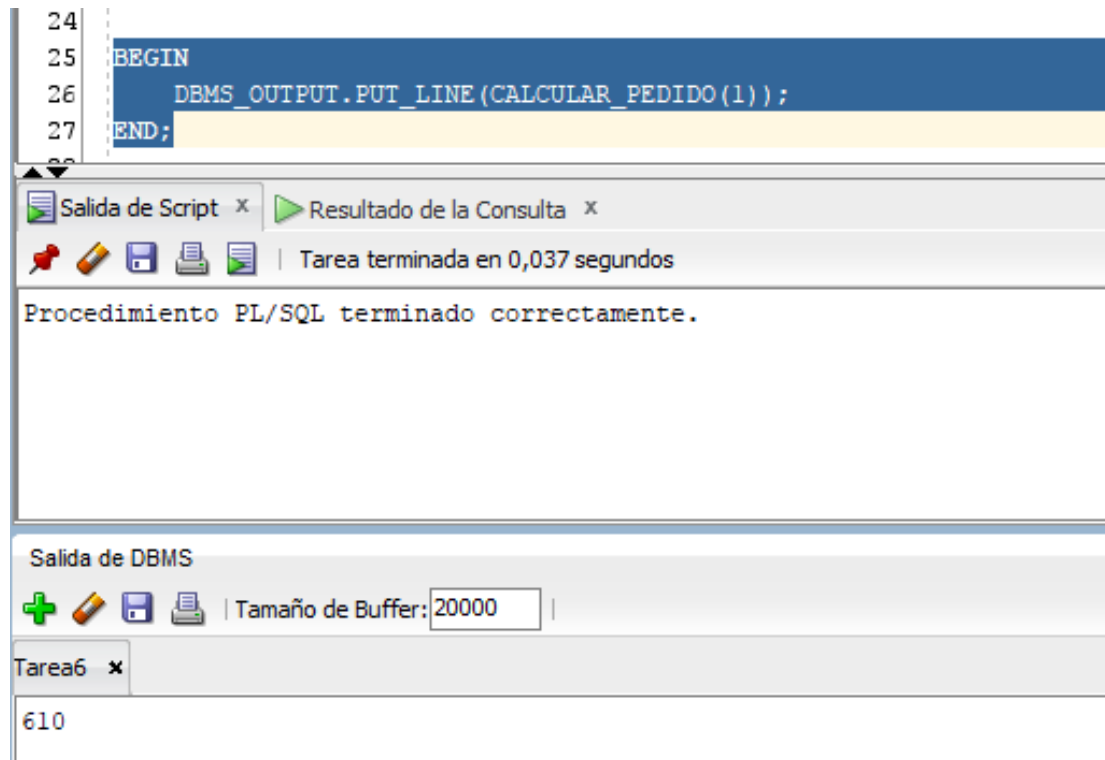
29

Salida de Script x Resultado de la Consulta x

Todas las Filas Recuperadas: 3 en 0,039 seg

	PRECIOVENTA	UNIDADES PEDIDAS
1	10	4
2	30	10
3	50	6

Ejecución DBMS_OUTPUT:



COMPROBACIÓN:

El resultado es 610, la suma de:

- $10 * 4 = 40$ No hay descuento
- $(30 * 0,95) * 10 = 285$
- $(50 * 0,95) * 6 = 285$

SENTENCIA:

```
CREATE OR REPLACE FUNCTION    CALCULAR_PEDIDO(p_codigopedido  
e_pedidos.codigopedido%type)
```

```
RETURN NUMBER
```

```
AS
```

```
    CURSOR productos_pedido IS
```

```
        SELECT p.precioventa, dp.unidadespedidas
```

```
        FROM e_productos p, e_detallepedidos dp
```

```
        WHERE p.codigoproducto = dp.codigoproducto
```

```
        AND dp.codigopedido = p_codigopedido;
```

```
v_total number(8) := 0;
```

```
BEGIN

FOR registro IN productos_pedido LOOP

IF registro.unidadespedidas >=0 AND registro.unidadespedidas <=5 THEN

    v_total:= v_total + (registro.unidadespedidas * registro.precioventa);

ELSIF registro.unidadespedidas >=6 AND registro.unidadespedidas <=10 THEN

    v_total:= v_total + (registro.unidadespedidas * (registro.precioventa * 0.95));

ELSIF registro.unidadespedidas >=11 AND registro.unidadespedidas <=15 THEN

    v_total:= v_total + (registro.unidadespedidas * (registro.precioventa * 0.93));

ELSE

    v_total:= v_total + (registro.unidadespedidas * (registro.precioventa * 0.9));

END IF;

END LOOP;

RETURN v_total;

END;
```

```
BEGIN

    DBMS_OUTPUT.PUT_LINE(CALCULAR_PEDIDO(1));

END;
```

ACTIVIDAD 2

Realizaremos un **procedimiento** llamado **CALCULAR_CLIENTE** que dándole como entrada un código de cliente y un año, nos devuelva como salida dos parámetros : el total facturado por los pedidos que haya pagado y el total facturado por los pedidos que no haya pagado durante ese año ese cliente. Sólo se tendrán en cuenta los pedidos que se hayan entregado ya.

Sólo tendremos en cuenta los clientes que hayan realizado más de dos pedido en ese año.

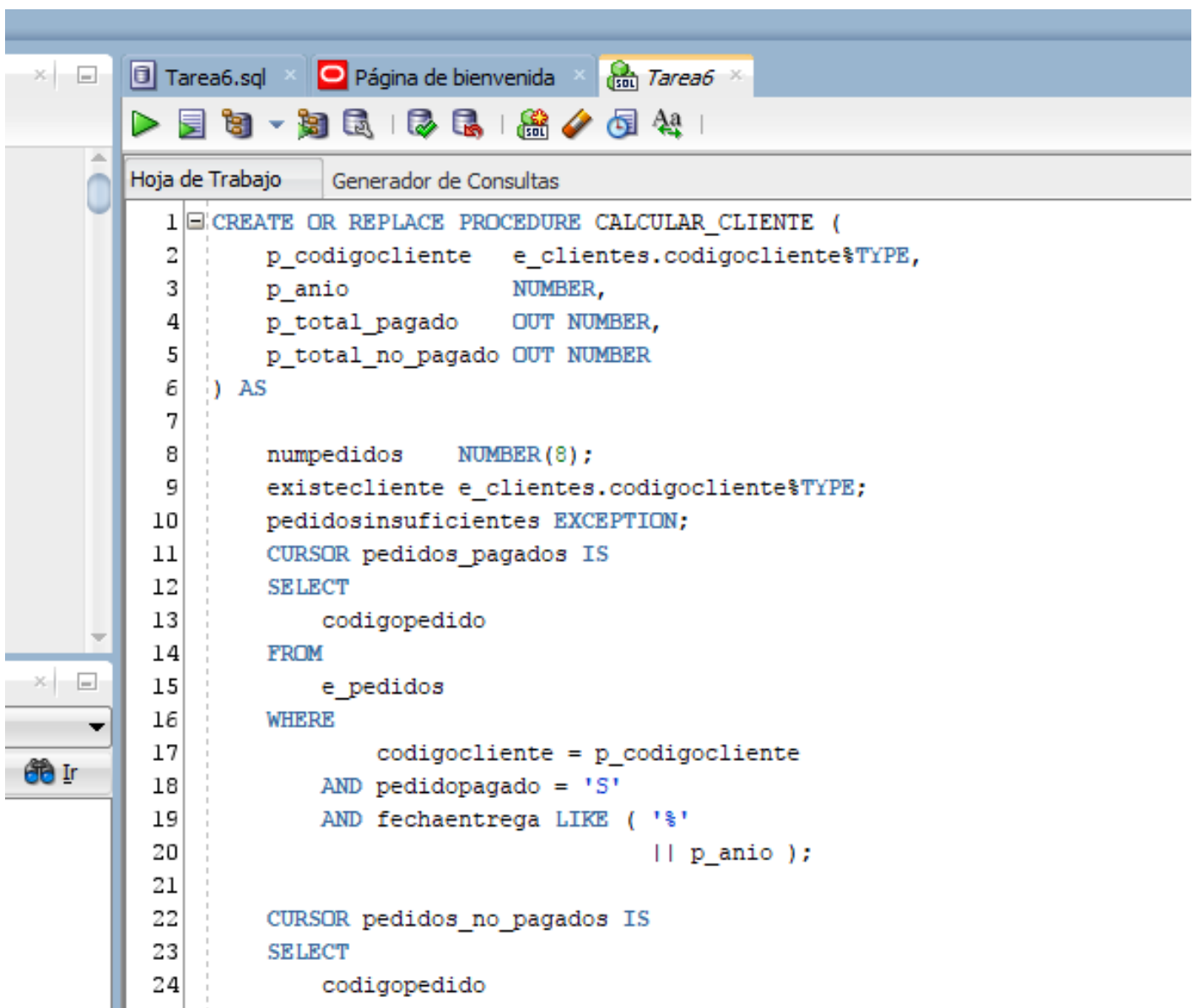
Si el cliente no existe saltaremos una excepción, visualizaremos un mensaje de que ese cliente no existe y finalizará el procedimiento devolviendo como salida - 1 en ambos totales.

Si el cliente no tiene pedidos durante ese año, saltaremos una excepción y visualizaremos un mensaje de que ese cliente no tiene pedidos ese año y finalizará el procedimiento devolviendo como salida -1 en ambos totales

Por cada pedido realizaremos una llamada a la función CALCULAR_PEDIDO, realizada en la actividad 1, que nos calculará el total de cada pedido. Estos totales se irán acumulando en el total de los pedidos pagados o en el total de los pedidos no pagados.

Finalmente devolveremos como parámetros dichos totales.

CAPTURAS DE PANTALLA

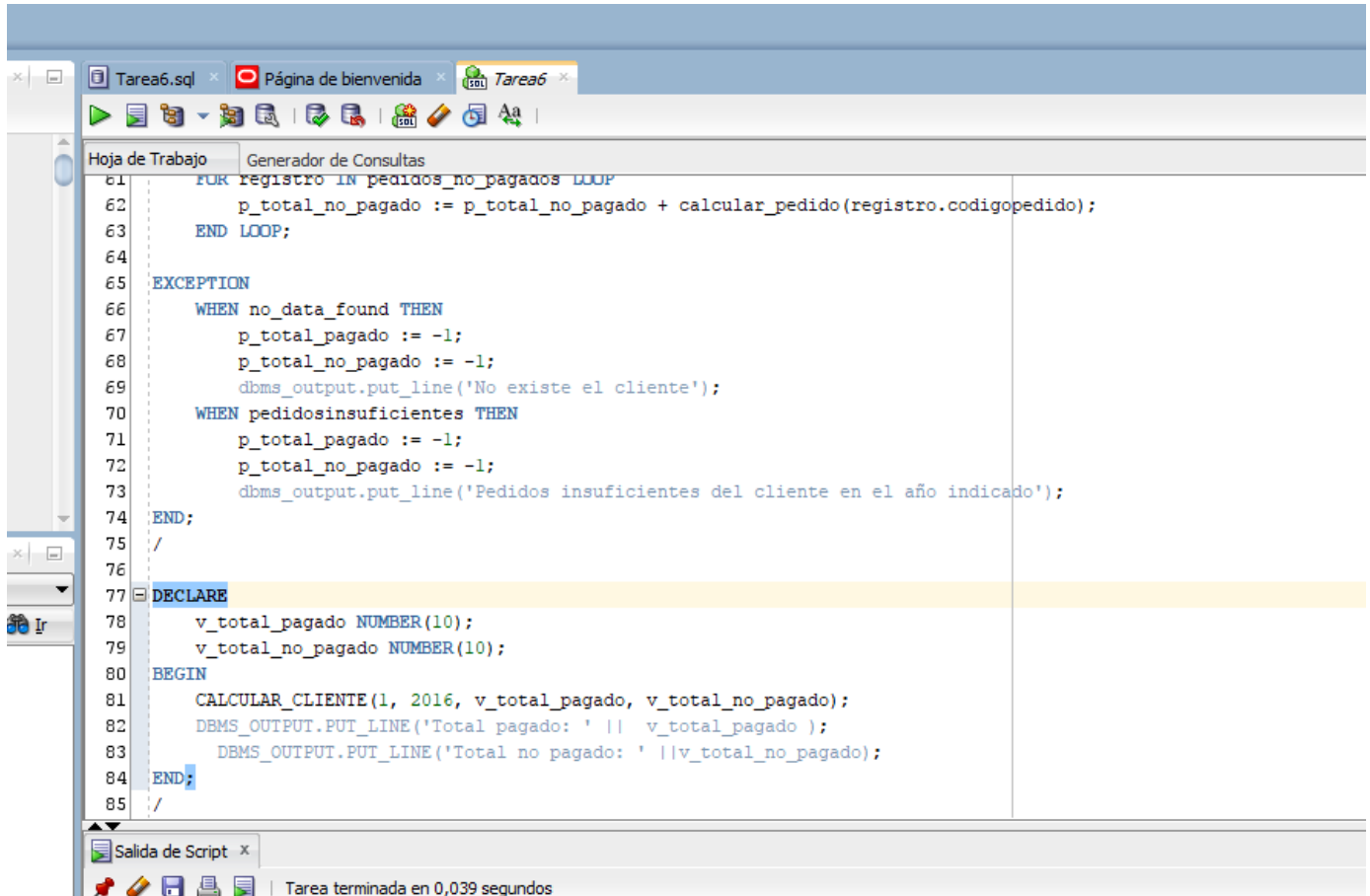


```
1 CREATE OR REPLACE PROCEDURE CALCULAR_CLIENTE (  
2     p_codigocliente  e_clientes.codigocliente%TYPE,  
3     p_anio           NUMBER,  
4     p_total_pagado   OUT NUMBER,  
5     p_total_no_pagado OUT NUMBER  
6 ) AS  
7  
8     numpedidos      NUMBER(8);  
9     existecliente   e_clientes.codigocliente%TYPE;  
10    pedidosinsuficientes EXCEPTION;  
11    CURSOR pedidos_pagados IS  
12    SELECT  
13        codigopedido  
14    FROM  
15        e_pedidos  
16    WHERE  
17        codigocliente = p_codigocliente  
18        AND pedidopagado = 'S'  
19        AND fechaentrega LIKE ( '%' || p_anio );  
20  
21    CURSOR pedidos_no_pagados IS  
22    SELECT  
23        codigopedido  
24    FROM
```

```
Hoja de Trabajo  Generador de Consultas
24      codigopedido
25  FROM
26      e_pedidos
27  WHERE
28      codigocliente = p_codigocliente
29      AND pedidopagado = 'N'
30      AND fechaentrega LIKE ( '%'
31                          || p_anio );
32
33  BEGIN
34      p_total_pagado := 0;
35      p_total_no_pagado := 0;
36  SELECT
37      codigocliente
38  INTO existecliente
39  FROM
40      e_clientes
41  WHERE
42      codigocliente = p_codigocliente;
43
44  SELECT
45      COUNT(*)
46  INTO numpedidos
47  FROM
48      e_pedidos
```

```
Tarea6.sql  Página de bienvenida  Tarea6
[Icons]
Hoja de Trabajo  Generador de Consultas
46  INTO numpedidos
47  FROM
48      e_pedidos
49  WHERE
50      codigocliente = p_codigocliente
51      AND fechaentrega LIKE ( '%'
52                          || p_anio );
53
54  IF numpedidos < 2 THEN
55      RAISE pedidosinsuficientes;
56  END IF;
57  FOR registro IN pedidos_pagados LOOP
58      p_total_pagado := p_total_pagado + calcular_pedido(registro.codigopedido);
59  END LOOP;
60
61  FOR registro IN pedidos_no_pagados LOOP
62      p_total_no_pagado := p_total_no_pagado + calcular_pedido(registro.codigopedido);
63  END LOOP;
64
65  EXCEPTION
66      WHEN no_data_found THEN
67          p_total_pagado := -1;
68          p_total_no_pagado := -1;
69          dbms_output.put_line('No existe el cliente');
70      WHEN pedidosinsuficientes THEN
```

Fin del procedimiento y añadido un nuevo bloque para pasar los parámetros de entrada del código y el año y que devuelva el total facturado por los pedidos que haya pagado(v_total_pagado) y el total facturado por los pedidos que no haya pagado(v_total_no_pagado) durante ese año ese cliente.



```

61  FOR registro IN pedidos_no_pagados LOOP
62      p_total_no_pagado := p_total_no_pagado + calcular_pedido(registro.codigopedido);
63  END LOOP;
64
65  EXCEPTION
66      WHEN no_data_found THEN
67          p_total_pagado := -1;
68          p_total_no_pagado := -1;
69          dbms_output.put_line('No existe el cliente');
70      WHEN pedidosinsuficientes THEN
71          p_total_pagado := -1;
72          p_total_no_pagado := -1;
73          dbms_output.put_line('Pedidos insuficientes del cliente en el año indicado');
74  END;
75  /
76
77  DECLARE
78      v_total_pagado NUMBER(10);
79      v_total_no_pagado NUMBER(10);
80  BEGIN
81      CALCULAR_CLIENTE(1, 2016, v_total_pagado, v_total_no_pagado);
82      DBMS_OUTPUT.PUT_LINE('Total pagado: ' || v_total_pagado);
83      DBMS_OUTPUT.PUT_LINE('Total no pagado: ' || v_total_no_pagado);
84  END;
85  /

```

Salida de Script x

Tarea terminada en 0,039 segundos

SENTENCIA:

```

CREATE OR REPLACE PROCEDURE CALCULAR_CLIENTE (

    p_codigocliente  e_clientes.codigocliente%TYPE,

    p_anio           NUMBER,

    p_total_pagado   OUT NUMBER,

    p_total_no_pagado OUT NUMBER

) AS

```

```
numpedidos    NUMBER(8);  
  
existecliente e_clientes.codigocliente%TYPE;  
  
pedidosinsuficientes EXCEPTION;  
  
CURSOR pedidos_pagados IS  
  
SELECT  
  
    codigopedido  
  
FROM  
  
    e_pedidos  
  
WHERE  
  
    codigocliente = p_codigocliente  
  
    AND pedidopagado = 'S'  
  
    AND fechaentrega LIKE ( '%' || p_anio );
```

```
CURSOR pedidos_no_pagados IS  
  
SELECT  
  
    codigopedido  
  
FROM  
  
    e_pedidos  
  
WHERE  
  
    codigocliente = p_codigocliente  
  
    AND pedidopagado = 'N'  
  
    AND fechaentrega LIKE ( '%' || p_anio );
```

```
BEGIN
```



```
p_total_pagado := 0;
p_total_no_pagado := 0;

SELECT
    codigocliente
INTO existecliente
FROM
    e_clientes
WHERE
    codigocliente = p_codigocliente;

SELECT
    COUNT(*)
INTO numpedidos
FROM
    e_pedidos
WHERE
    codigocliente = p_codigocliente
    AND fechaentrega LIKE ( '%'
                            || p_anio );

IF numpedidos < 2 THEN
    RAISE pedidosinsuficientes;
END IF;

FOR registro IN pedidos_pagados LOOP
    p_total_pagado := p_total_pagado +
    calcular_pedido(registro.codigopedido);
```

```
END LOOP;

FOR registro IN pedidos_no_pagados LOOP

    p_total_no_pagado      :=      p_total_no_pagado      +
    calcular_pedido(registro.codigopedido);

END LOOP;

EXCEPTION

    WHEN no_data_found THEN

        p_total_pagado := -1;

        p_total_no_pagado := -1;

        dbms_output.put_line('No existe el cliente');

    WHEN pedidosinsuficientes THEN

        p_total_pagado := -1;

        p_total_no_pagado := -1;

        dbms_output.put_line('Pedidos insuficientes del cliente en el año indicado');

END;

/

DECLARE

    v_total_pagado NUMBER(10);

    v_total_no_pagado NUMBER(10);

BEGIN

    CALCULAR_CLIENTE(1, 2015, v_total_pagado, v_total_no_pagado);

    DBMS_OUTPUT.PUT_LINE('Total pagado: ' || v_total_pagado );

    DBMS_OUTPUT.PUT_LINE('Total no pagado: ' || v_total_no_pagado);
```

END;

/

ACTIVIDAD 3.

Queremos crear los siguientes disparadores:

3.1. Un disparador llamado **DISP_PEDIDOS** que salte al insertar o actualizar en la tabla **pedidos** para que no nos deje insertar o actualizar si se produce uno de los siguientes casos:

- La fecha de pedido debe ser menor que la fecha esperada de entrega y la fecha de entrega.
- El estado de un pedido sólo puede tener los valores P, E o D (Pendiente de entregar, Entregado o Devuelto)
- La columna PedidoPagado sólo puede tener los valores : S o N (Pagado Si, Pagado No)

Se lanzará una excepción mediante la cual el registro no se inserte y visualiza el mensaje adecuado

Como la tabla es propiedad de SYS, tengo que crear un nuevo usuario ya que da error al ejecutar disparadores ya que el usuario SYS es el usuario del sistema y se usa para realizar tareas críticas del sistema.

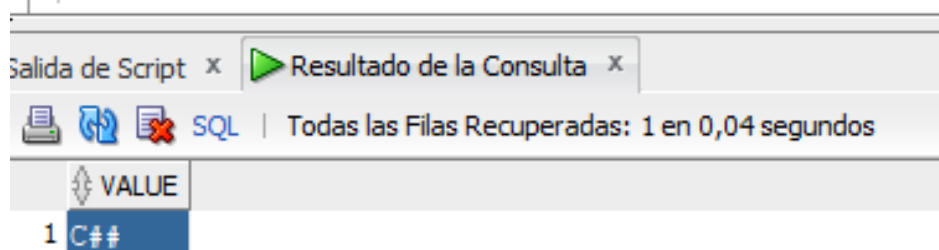
Creo un usuario con el prefijo del sistema

```
SELECT value FROM v$parameter WHERE name = 'common_user_prefix';
```

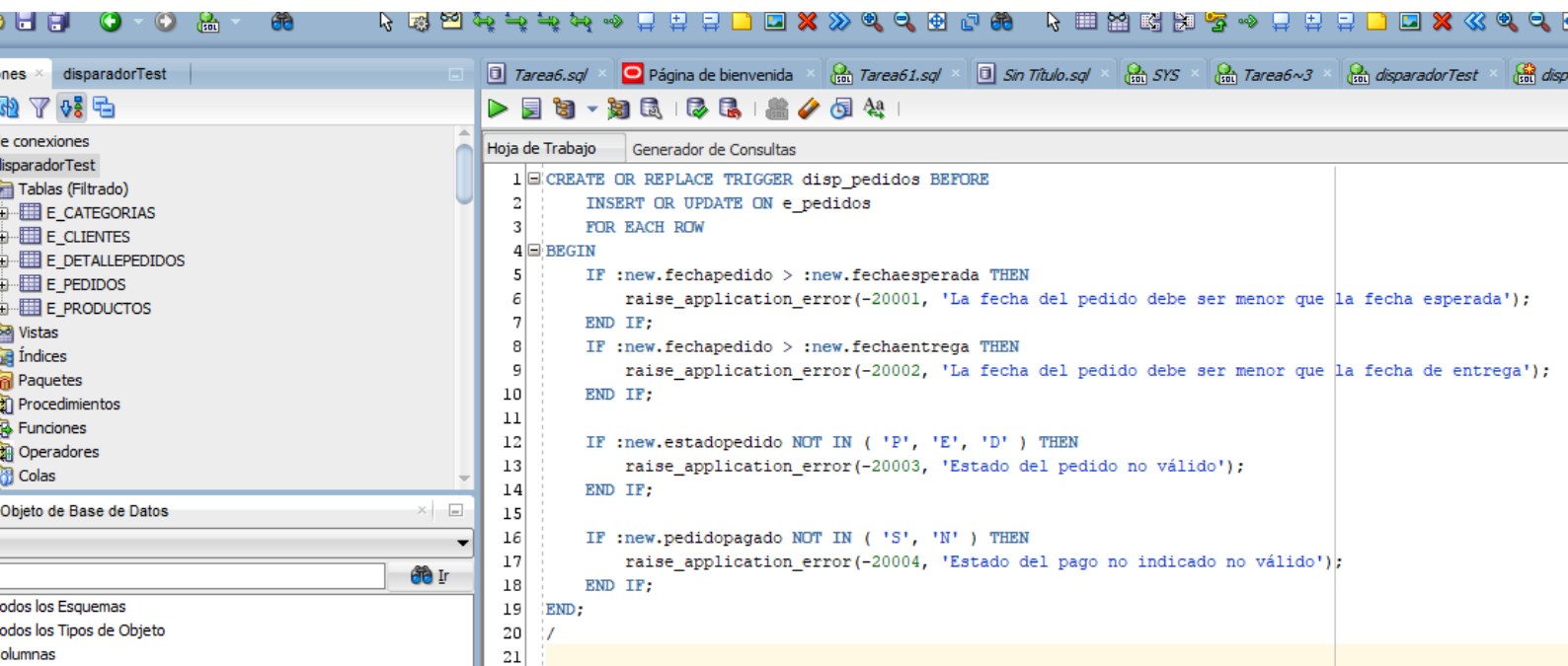
```
CREATE USER C##trigger IDENTIFIED BY "1234";
```

```
GRANT CREATE TRIGGER TO C##trigger;
```

```
1 CREATE USER C##trigger IDENTIFIED BY "1234";
2 GRANT CREATE TRIGGER TO C##trigger;
3
```



Creo el usuario con todas las tablas, función y procedimiento para poder ejecutar el disparador... y lo testeo insertando valores en la tabla e_pedidos con un insert.



SENTENCIA:

CREATE OR REPLACE TRIGGER disp_pedidos BEFORE

INSERT OR UPDATE ON e_pedidos

FOR EACH ROW

BEGIN

IF :new.fechapedido > :new.fechaesperada THEN

raise_application_error(-20001, 'La fecha del pedido debe ser menor que la fecha esperada');

END IF;

IF :new.fechapedido > :new.fechaentrega THEN

raise_application_error(-20002, 'La fecha del pedido debe ser menor que la fecha de entrega');

END IF;

```
IF :new.estadopedido NOT IN ( 'P', 'E', 'D' ) THEN
```

```
    raise_application_error(-20003, 'Estado del pedido no válido');
```

```
END IF;
```

```
IF :new.pedidopagado NOT IN ( 'S', 'N' ) THEN
```

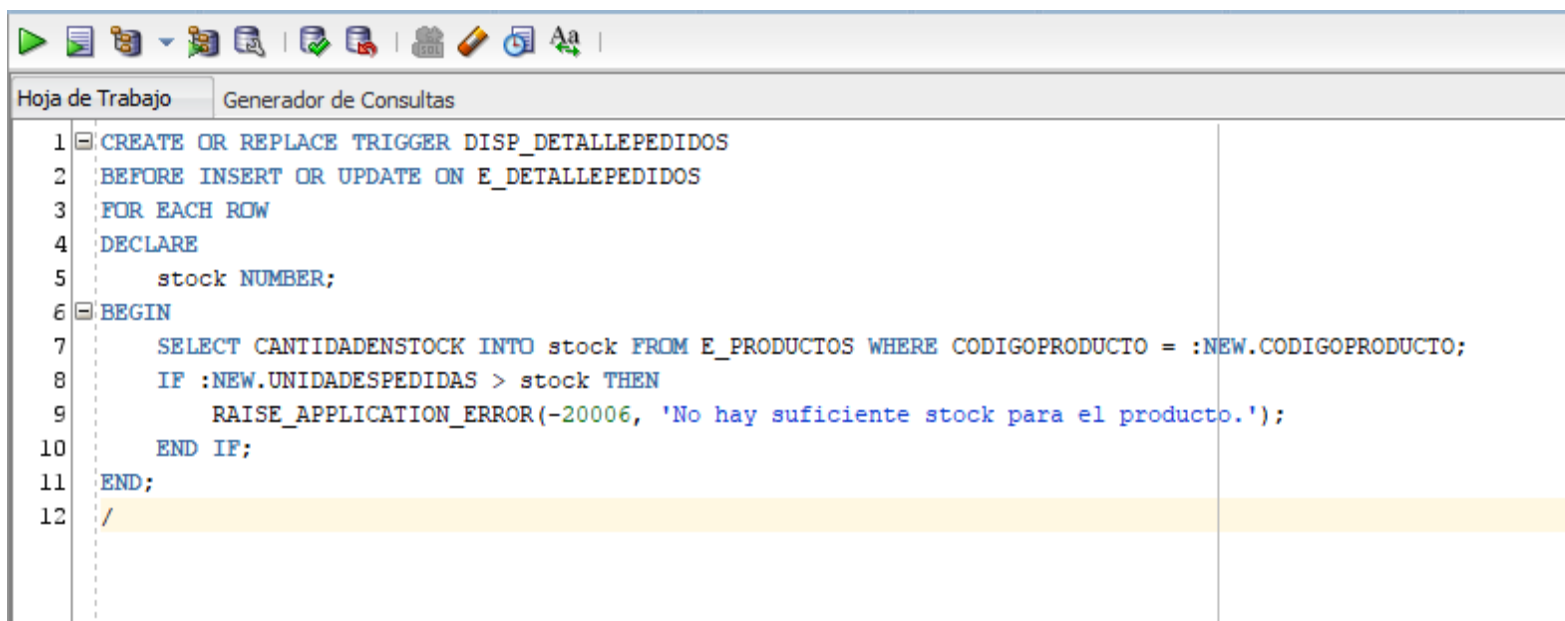
```
    raise_application_error(-20004, 'Estado del pago no indicado no válido');
```

```
END IF;
```

```
END;
```

```
/
```

- 3.2. Un disparador llamado **DISP_DETALLEPEDIDOS**, en el que al insertar o actualizar en la tabla **detallepedidos** controle que la cantidad pedida del producto sea menor que la cantidad en stock de dicho producto. Se lanzará una excepción mediante la cual el registro no se inserte y visualiza el mensaje adecuado.



The screenshot shows a SQL IDE window with a toolbar at the top and a tab labeled 'Hoja de Trabajo' and 'Generador de Consultas'. The main area contains the following SQL code:

```
1 CREATE OR REPLACE TRIGGER DISP_DETALLEPEDIDOS
2 BEFORE INSERT OR UPDATE ON E_DETALLEPEDIDOS
3 FOR EACH ROW
4 DECLARE
5     stock NUMBER;
6 BEGIN
7     SELECT CANTIDADENSTOCK INTO stock FROM E_PRODUCTOS WHERE CODIGOPRODUCTO = :NEW.CODIGOPRODUCTO;
8     IF :NEW.UNIDADES PEDIDAS > stock THEN
9         RAISE_APPLICATION_ERROR(-20006, 'No hay suficiente stock para el producto.');
```

The code is highlighted in yellow, and the line numbers 1 through 12 are visible on the left side of the editor.

SENTENCIA:

```
CREATE OR REPLACE TRIGGER DISP_DETALLEPEDIDOS  
BEFORE INSERT OR UPDATE ON E_DETALLEPEDIDOS  
FOR EACH ROW  
DECLARE  
    stock NUMBER;  
BEGIN  
    SELECT CANTIDADENSTOCK INTO stock FROM E_PRODUCTOS WHERE  
        CODIGOPRODUCTO = :NEW.CODIGOPRODUCTO;  
    IF :NEW.UNIDADESPEDIDAS > stock THEN  
        RAISE_APPLICATION_ERROR(-20006, 'No hay suficiente stock para el  
        producto.');
```

END IF;

END;

/