

EJERCICIO 2

Dado el siguiente modelo de datos:

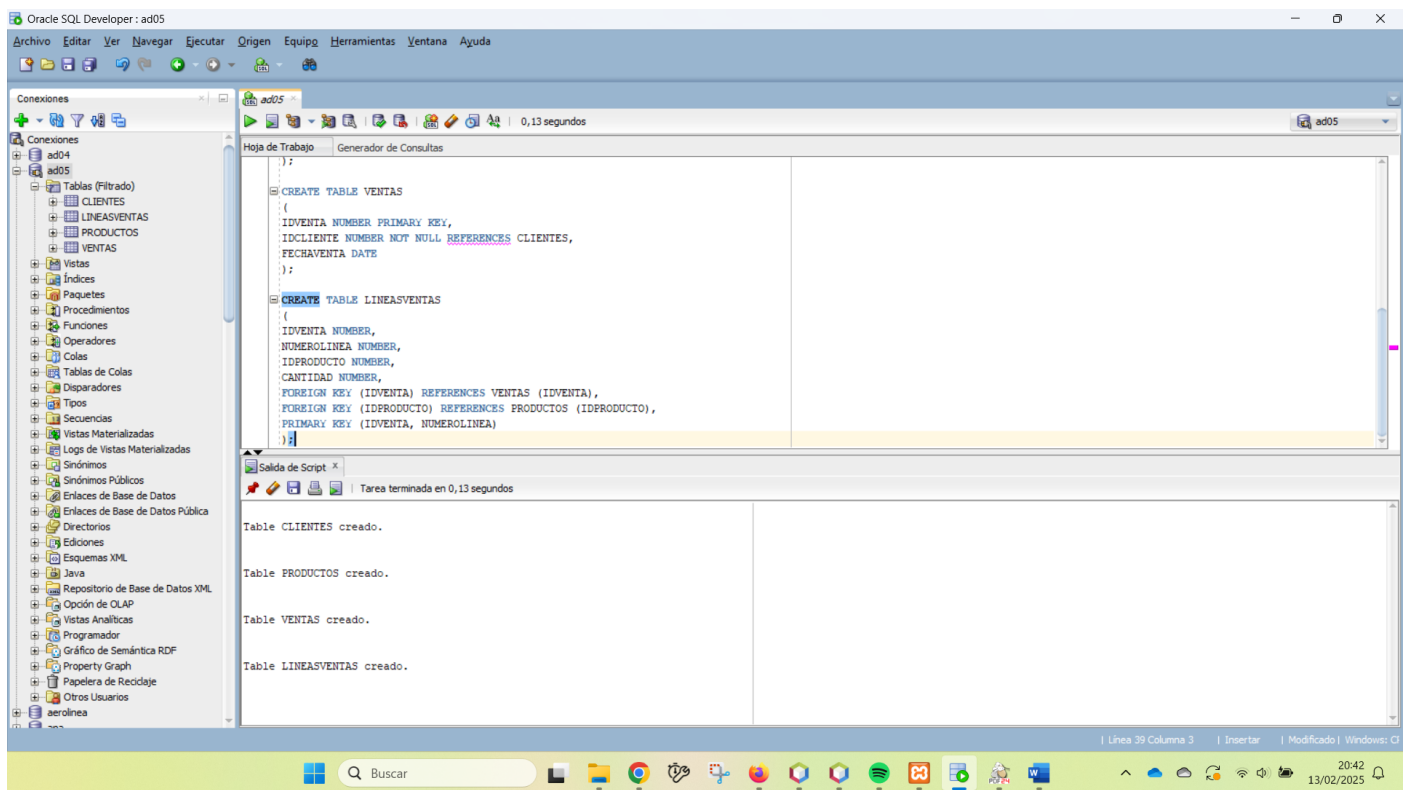
```
CREATE TABLE CLIENTES(
IDCLIENTE NUMBER PRIMARY KEY,
NOMBRE VARCHAR2(50),
DIRECCION VARCHAR2(50),
POBLACION VARCHAR2(50),
CODPOSTAL NUMBER(5),
PROVINCIA VARCHAR2(40),
NIF VARCHAR2(9) UNIQUE,
TELEFONO1 VARCHAR2(15),
TELEFONO2 VARCHAR2(15),
TELEFONO3 VARCHAR2(15)
);

CREATE TABLE PRODUCTOS(
IDPRODUCTO NUMBER PRIMARY KEY,
DESCRIPCION VARCHAR2(80),
PVP NUMBER,
STOCKACTUAL NUMBER
);

CREATE TABLE VENTAS(
IDVENTA NUMBER PRIMARY KEY,
IDCLIENTE NUMBER NOT NULL REFERENCES CLIENTES,
FECHAVENTA DATE
);

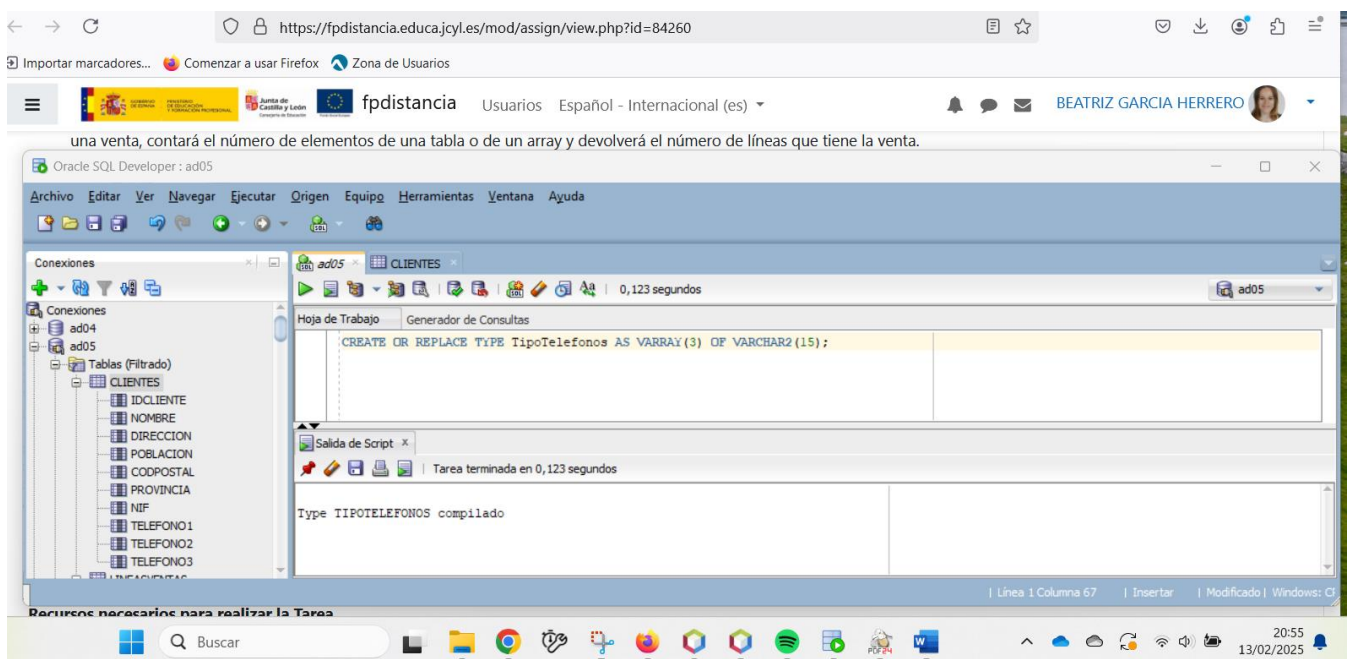
CREATE TABLE LINEASVENTAS(
IDVENTA NUMBER,
NUMEROLINEA NUMBER,
IDPRODUCTO NUMBER,
CANTIDAD NUMBER,
FOREIGN KEY (IDVENTA) REFERENCES VENTAS (IDVENTA),
FOREIGN KEY (IDPRODUCTO) REFERENCES PRODUCTOS (IDPRODUCTO),
PRIMARY KEY (IDVENTA, NUMEROLINEA) );
```

Creo la base de datos y sus tablas:



1. Definir un tipo varray de dimensión 3 para contener los teléfonos

CREATE OR REPLACE TYPE TipoTelefonos AS VARRAY(3) OF VARCHAR2(15);



2. Crear los tipos dirección, cliente, producto y línea de venta

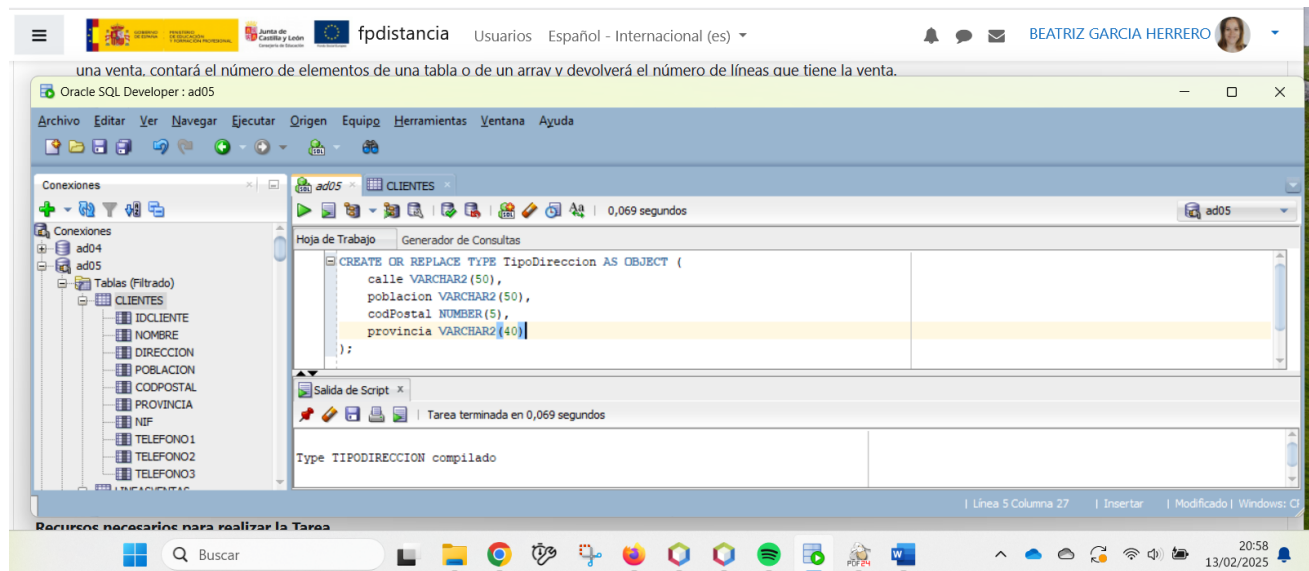
CREATE OR REPLACE TYPE **TipoDireccion** AS OBJECT (

calle VARCHAR2(50),

poblacion VARCHAR2(50),

codPostal NUMBER(5),

provincia VARCHAR2(40));



CREATE OR REPLACE TYPE **TipoCliente** AS OBJECT (

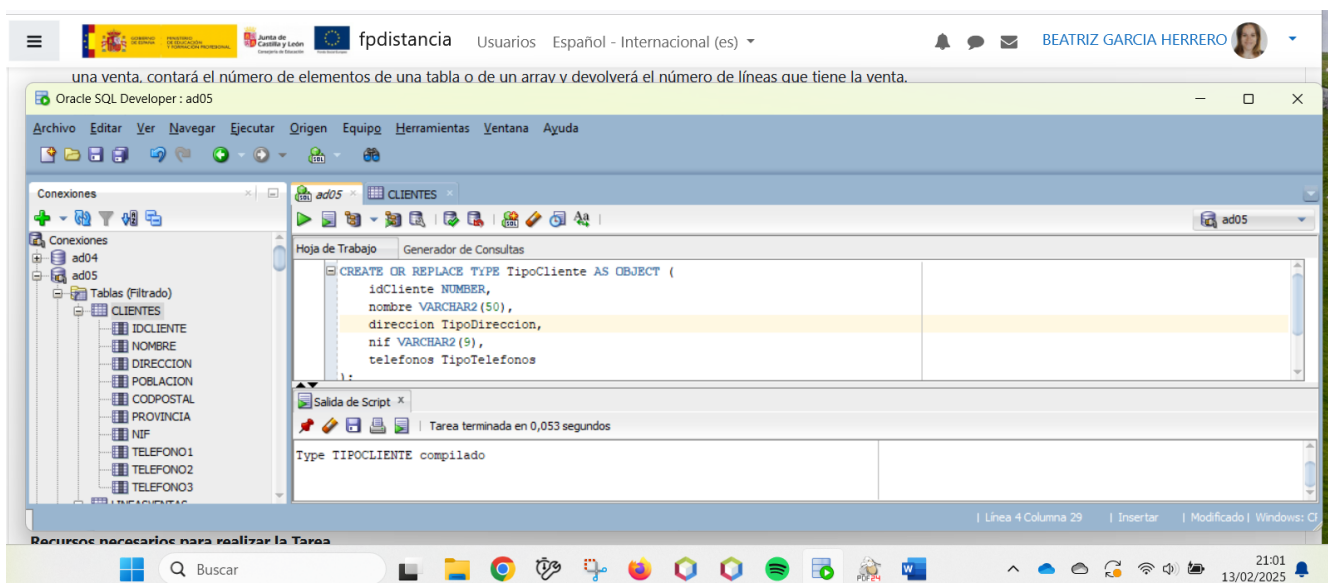
idCliente NUMBER,

nombre VARCHAR2(50),

direccion TipoDireccion,

nif VARCHAR2(9),

telefonos TipoTelefonos);



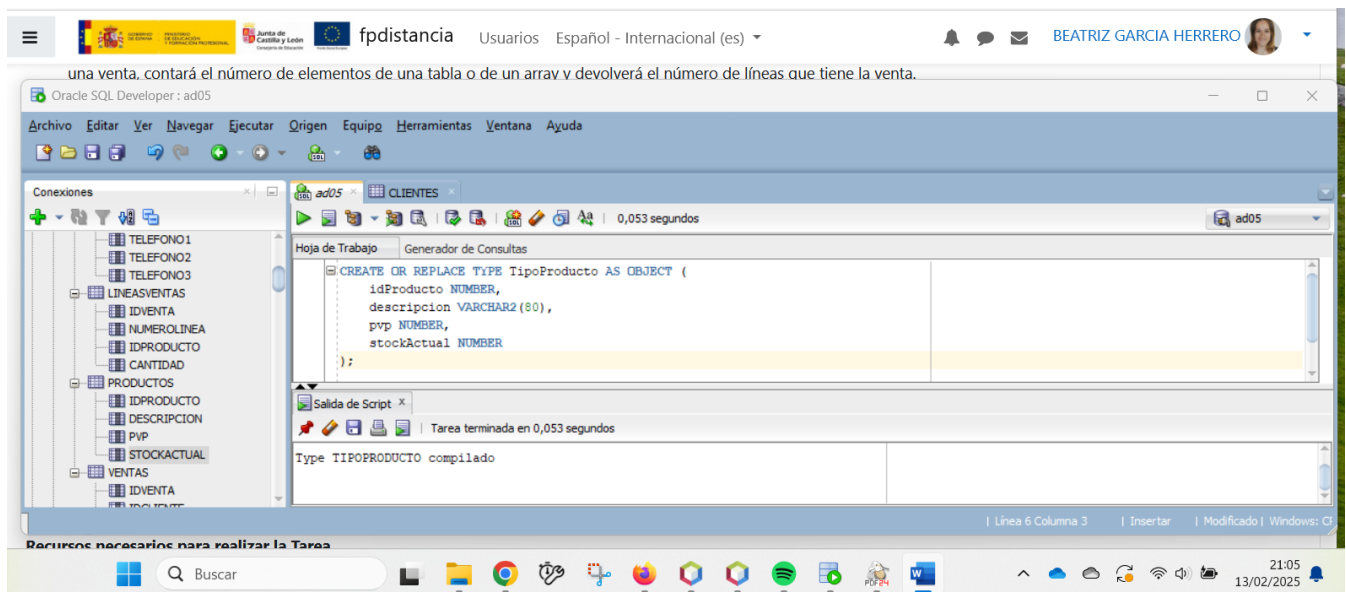
CREATE OR REPLACE TYPE **TipoProducto** AS OBJECT (

idProducto NUMBER,

descripcion VARCHAR2(80),

pvp NUMBER,

stockActual NUMBER);



CREATE OR REPLACE TYPE **TipoLineaVenta** AS OBJECT (

producto TipoProducto,

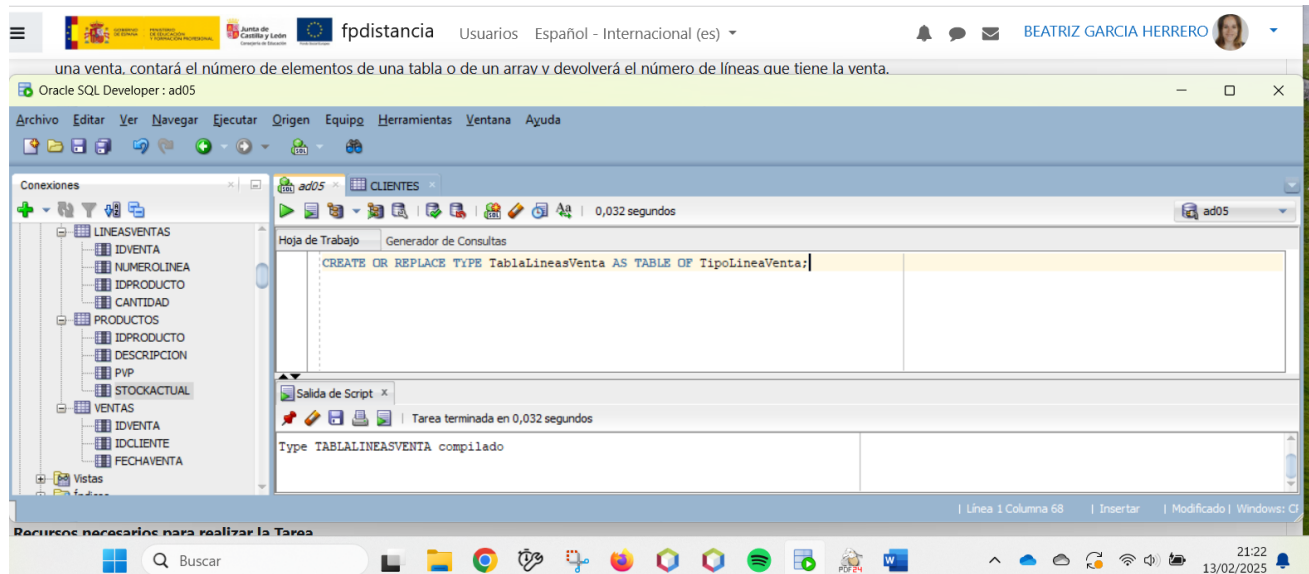
cantidad NUMBER);

```
CREATE OR REPLACE TYPE TipoLineaVenta AS OBJECT (  
    idProducto NUMBER,  
    cantidad NUMBER  
);
```

3. Crear un tipo tabla anidada para contener las líneas de una venta:

TablaLineasVenta es una colección de **TipoLineaVenta**, actuando como una tabla dentro de cada venta.

CREATE OR REPLACE TYPE TablaLineasVenta AS TABLE OF TipoLineaVenta;



4. Crear un tipo venta para los datos de las ventas, cada venta tendrá un atributo LINEAS del tipo tabla anidada definida anteriormente:

CREATE OR REPLACE TYPE TipoVenta AS OBJECT (

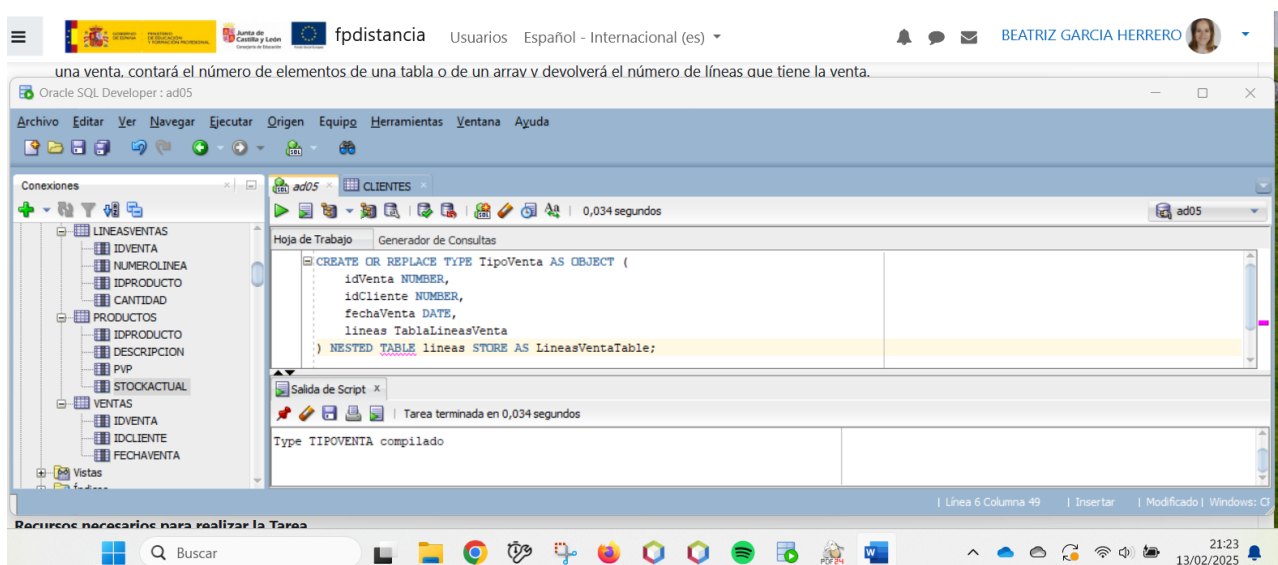
idVenta NUMBER,

idCliente NUMBER,

fechaVenta DATE,

lineas TablaLineasVenta

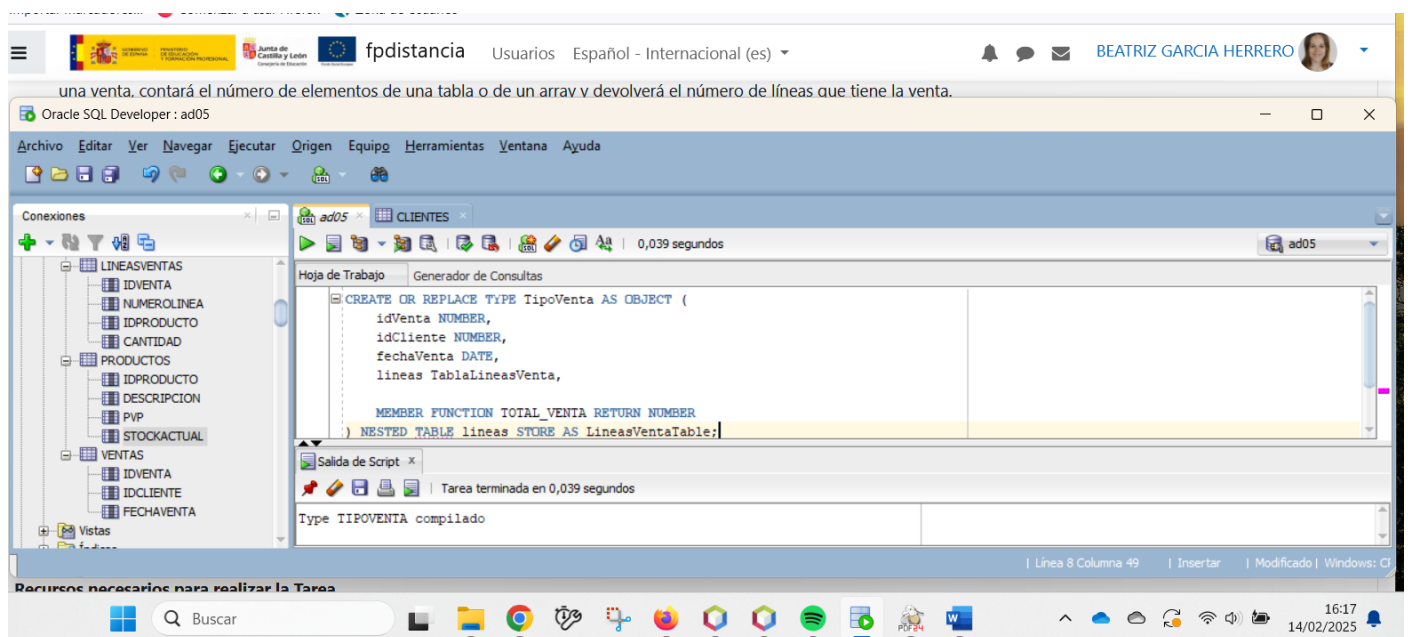
) NESTED TABLE lineas STORE AS LineasVentaTable;



5. Crea el cuerpo del tipo anterior, teniendo en cuenta que se definirá la función miembro **TOTAL_VENTA** que calcula el total de la venta de las líneas de venta que forman parte de una venta, contará el número de elementos de una tabla o de un array y devolverá el número de líneas que tiene la venta.

Incluyo la función TOTAL_VENTA dentro de TipoVenta.

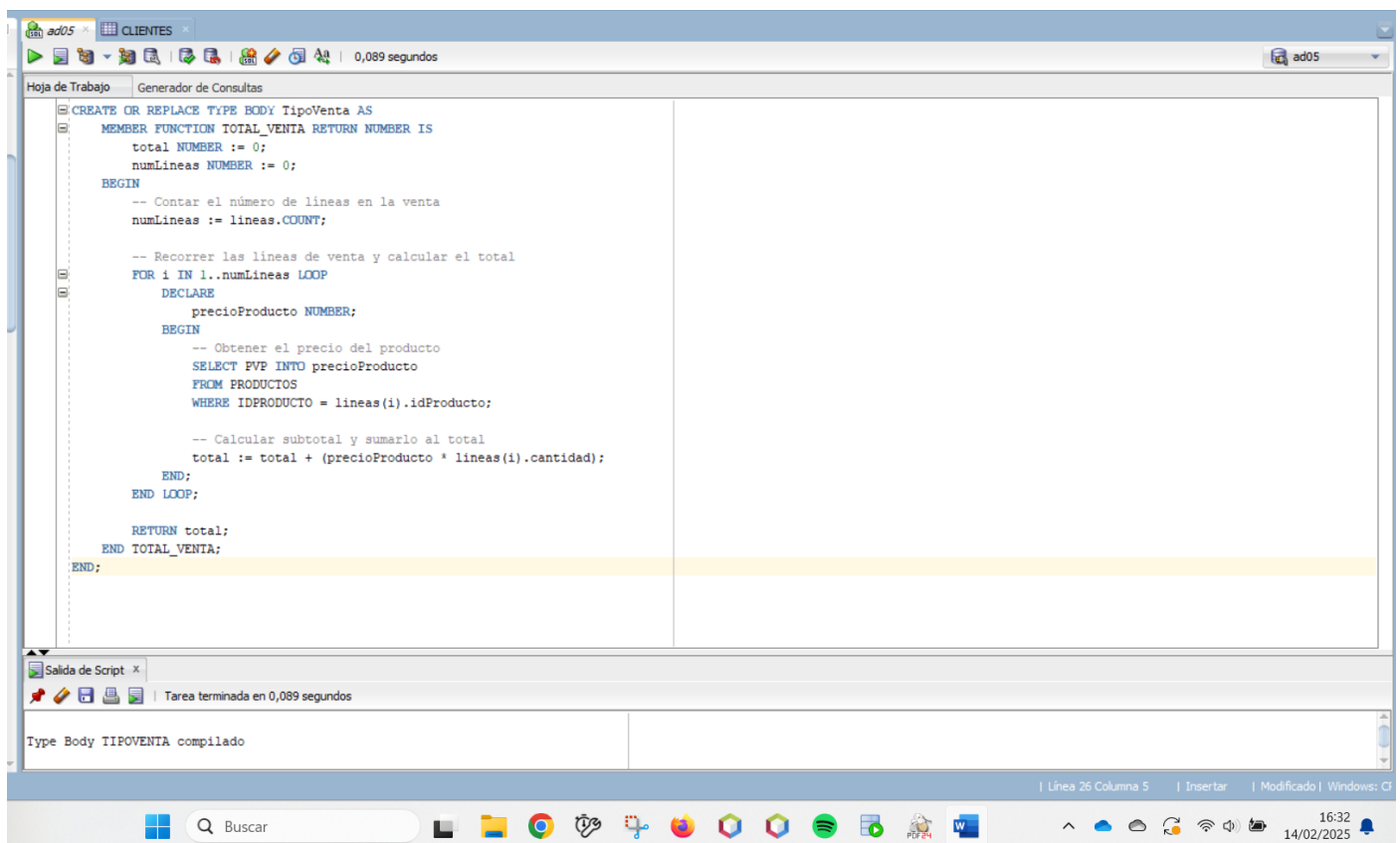
```
CREATE OR REPLACE TYPE TipoVenta AS OBJECT (
    idVenta NUMBER,
    idCliente NUMBER,
    fechaVenta DATE,
    lineas TablaLineasVenta,
    MEMBER FUNCTION TOTAL_VENTA RETURN NUMBER
);
```



```
CREATE OR REPLACE TYPE BODY TipoVenta AS
    MEMBER FUNCTION TOTAL_VENTA RETURN NUMBER IS
        total NUMBER := 0;
        numLineas NUMBER := 0;
    BEGIN
        -- Contar el número de líneas en la venta
        numLineas := lineas.COUNT;
```

```
-- Recorrer las líneas de venta y calcular el total
FOR i IN 1..numLineas LOOP
  DECLARE
    precioProducto NUMBER;
  BEGIN
    -- Obtener el precio del producto
    SELECT PVP INTO precioProducto
    FROM PRODUCTOS
    WHERE IDPRODUCTO = lineas(i).idProducto;

    -- Calcular subtotal y sumarlo al total
    total := total + (precioProducto * lineas(i).cantidad);
  END;
END LOOP;
RETURN total;
END TOTAL_VENTA;
END;
```

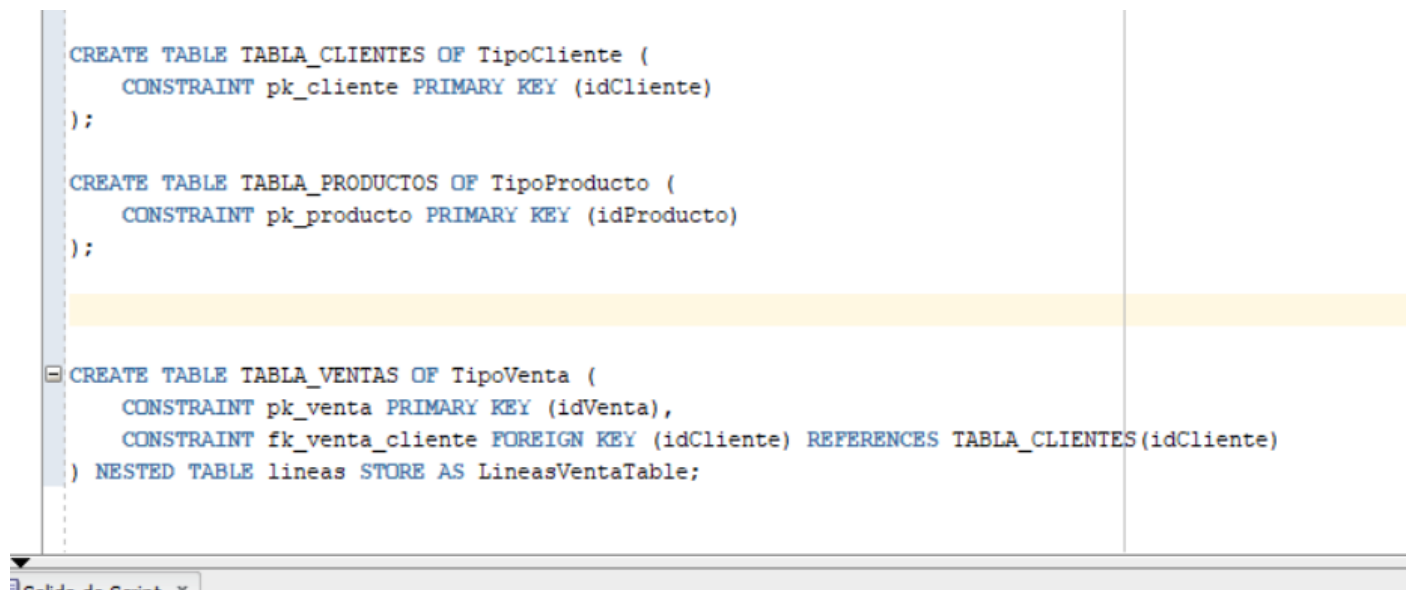


6. Crear las tablas donde almacenar los objetos de la aplicación. Se creará una tabla para clientes, otra para productos y otra para las ventas, en dichas tablas se definirán las oportunas claves primarias.

```
CREATE TABLE TABLA_CLIENTES OF TipoCliente (  
    CONSTRAINT pk_cliente PRIMARY KEY (idCliente)  
);
```

```
CREATE TABLE TABLA_PRODUCTOS OF TipoProducto (  
    CONSTRAINT pk_producto PRIMARY KEY (idProducto)  
);
```

```
CREATE TABLE TABLA_VENTAS OF TipoVenta (  
    CONSTRAINT pk_venta PRIMARY KEY (idVenta),  
    CONSTRAINT fk_venta_cliente FOREIGN KEY (idCliente) REFERENCES TABLA_CLIENTES(idCliente)  
) NESTED TABLE lineas STORE AS LineasVentaTable;
```



```
CREATE TABLE TABLA_CLIENTES OF TipoCliente (  
    CONSTRAINT pk_cliente PRIMARY KEY (idCliente)  
);  
  
CREATE TABLE TABLA_PRODUCTOS OF TipoProducto (  
    CONSTRAINT pk_producto PRIMARY KEY (idProducto)  
);  
  
CREATE TABLE TABLA_VENTAS OF TipoVenta (  
    CONSTRAINT pk_venta PRIMARY KEY (idVenta),  
    CONSTRAINT fk_venta_cliente FOREIGN KEY (idCliente) REFERENCES TABLA_CLIENTES(idCliente)  
) NESTED TABLE lineas STORE AS LineasVentaTable;
```


7. Inserta dos clientes y cinco productos.

```
INSERT INTO TABLA_CLIENTES VALUES (1, 'Beatriz García',  
TipoDireccion('Calle A, 123', 'Salamanca', 28001, 'Salamanca'),  
'12345678A',  
TipoTelefonos('600111222', '911223344', '612345678')  
);
```

```
INSERT INTO TABLA_CLIENTES VALUES (2, 'Enrique García',  
TipoDireccion('Av. Principal, 45', 'Badajoz', 08001, 'Badajoz'),  
'87654321B',  
TipoTelefonos('622333444', '933445566', '698765432')  
);
```

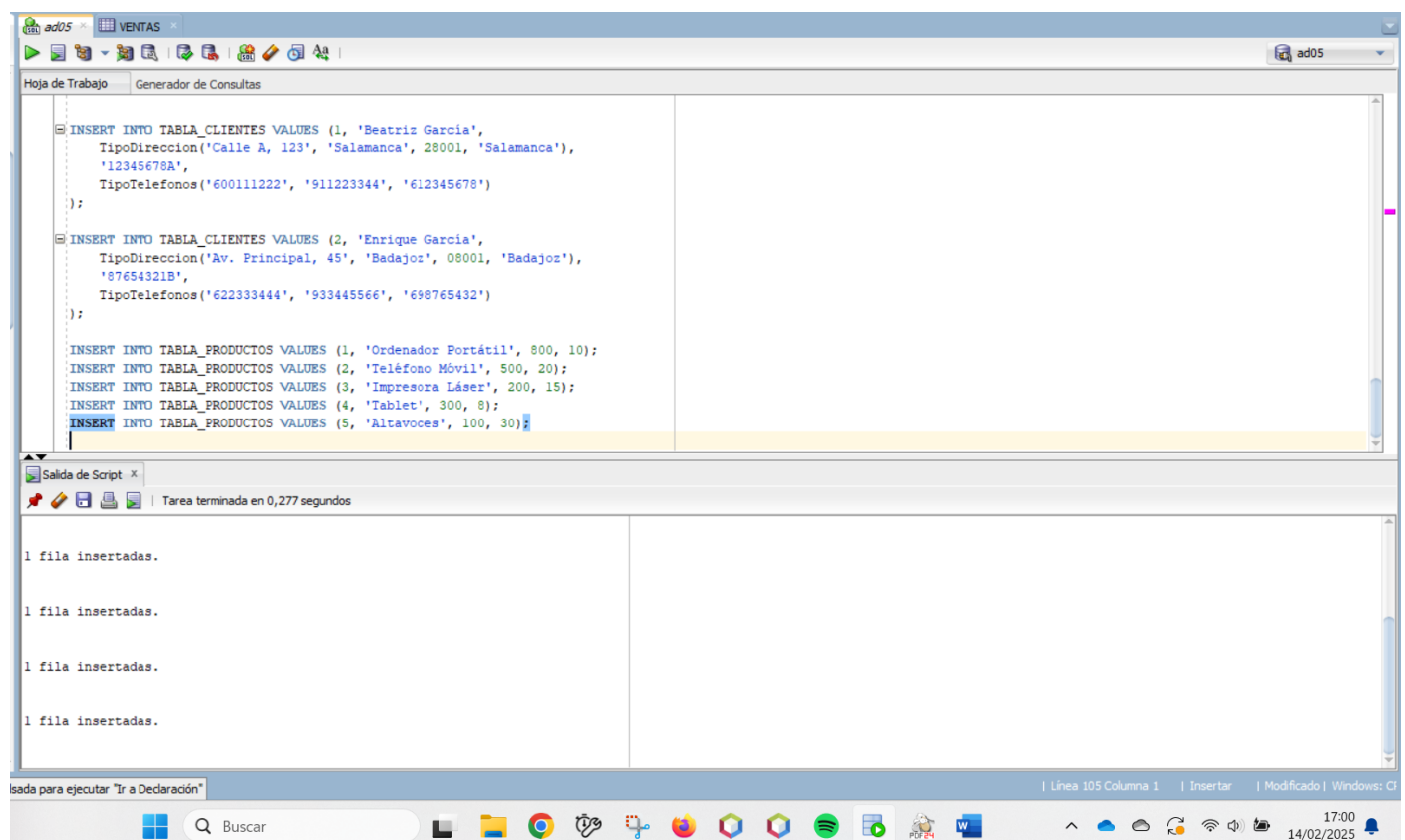
```
INSERT INTO TABLA_PRODUCTOS VALUES (1, 'Ordenador Portátil', 800, 10);
```

```
INSERT INTO TABLA_PRODUCTOS VALUES (2, 'Teléfono Móvil', 500, 20);
```

```
INSERT INTO TABLA_PRODUCTOS VALUES (3, 'Impresora Láser', 200, 15);
```

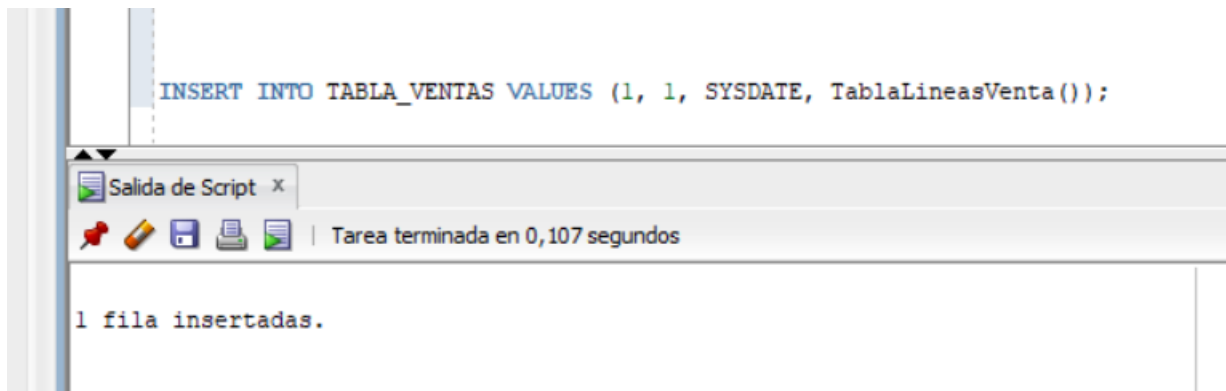
```
INSERT INTO TABLA_PRODUCTOS VALUES (4, 'Tablet', 300, 8);
```

```
INSERT INTO TABLA_PRODUCTOS VALUES (5, 'Altavoces', 100, 30);
```



8. Insertar en TABLA_VENTAS la venta con IDVENTA 1 para el IDCLIENTE 1

```
INSERT INTO TABLA_VENTAS VALUES (1, 1, SYSDATE, TablaLineasVenta());
```

**9. Insertar en TABLA_VENTAS dos líneas de venta para el IDVENTA 1 para los productos 1 (la CANTIDAD es 1) y 2 (la CANTIDAD es 2)**

```
UPDATE TABLA_VENTAS
```

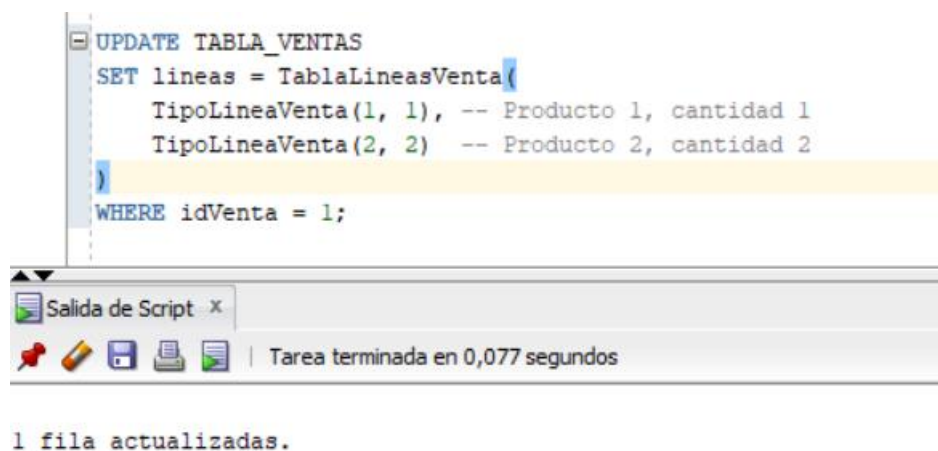
```
SET lineas = TablaLineasVenta(
```

```
    TipoLineaVenta(1, 1), -- Producto 1, cantidad 1
```

```
    TipoLineaVenta(2, 2) -- Producto 2, cantidad 2
```

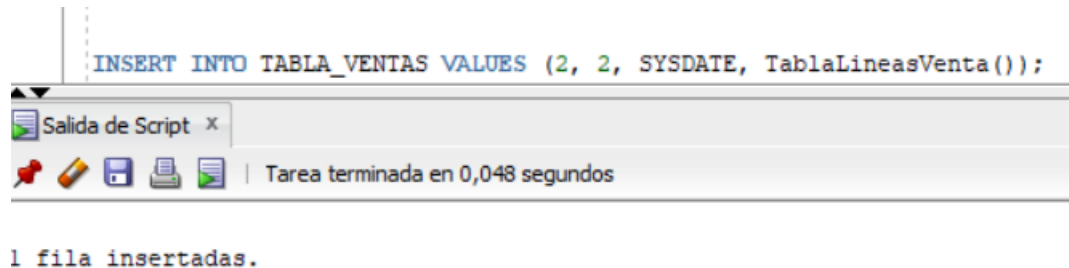
```
)
```

```
WHERE idVenta = 1;
```



10. Insertar en TABLA_VENTAS la venta con IDVENTA 2 para el IDCLIENTE

```
INSERT INTO TABLA_VENTAS VALUES (2, 2, SYSDATE, TablaLineasVenta());
```

**11. Insertar en TABLA_VENTAS tres líneas de venta para el IDVENTA 2 para los productos 1 (la CANTIDAD es 2), 4 (la CANTIDAD es 1) y 5 (la CANTIDAD es 4)**

```
UPDATE TABLA_VENTAS
```

```
SET lineas = TablaLineasVenta(
```

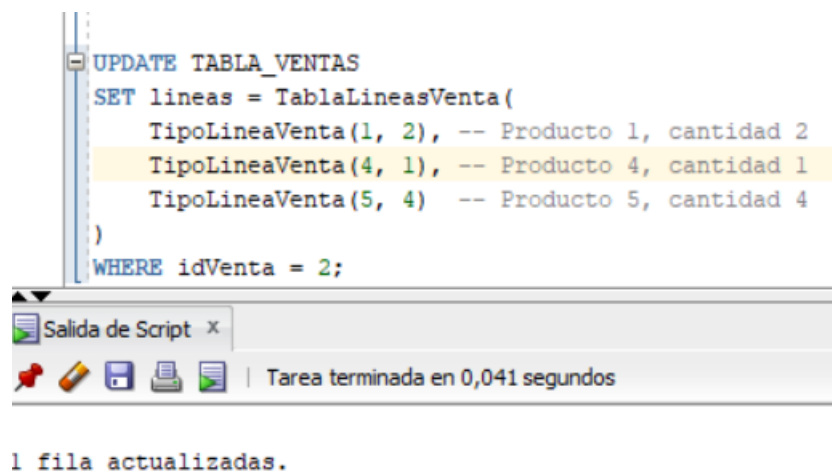
```
    TipoLineaVenta(1, 2), -- Producto 1, cantidad 2
```

```
    TipoLineaVenta(4, 1), -- Producto 4, cantidad 1
```

```
    TipoLineaVenta(5, 4) -- Producto 5, cantidad 4
```

```
)
```

```
WHERE idVenta = 2;
```



12. Realizar un procedimiento que recibiendo el identificador visualice los datos de la venta.

```

CREATE OR REPLACE PROCEDURE MostrarVenta(idVentaConsulta NUMBER) IS
    vVenta TipoVenta;
BEGIN
    -- Obtener los datos de la venta
    SELECT VALUE(v) INTO vVenta FROM TABLA_VENTAS v WHERE v.idVenta = idVentaConsulta;

    -- Mostrar datos de la venta
    DBMS_OUTPUT.PUT_LINE('Venta ID: ' || vVenta.idVenta);
    DBMS_OUTPUT.PUT_LINE('Cliente ID: ' || vVenta.idCliente);
    DBMS_OUTPUT.PUT_LINE('Fecha: ' || TO_CHAR(vVenta.fechaVenta, 'DD-MON-YYYY'));

    -- Mostrar líneas de venta
    DBMS_OUTPUT.PUT_LINE('Líneas de venta:');

    FOR i IN 1..vVenta.lineas.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE(' Producto ID: ' || vVenta.lineas(i).idProducto ||
                               ', Cantidad: ' || vVenta.lineas(i).cantidad);
    END LOOP;

    -- Mostrar total de la venta
    DBMS_OUTPUT.PUT_LINE('Total Venta: ' || vVenta.TOTAL_VENTA);
END;

```

The screenshot shows a SQL script editor with the following code:

```

CREATE OR REPLACE PROCEDURE MostrarVenta(idVentaConsulta NUMBER) IS
    vVenta TipoVenta;
BEGIN
    -- Obtener los datos de la venta
    SELECT VALUE(v) INTO vVenta FROM TABLA_VENTAS v WHERE v.idVenta = idVentaConsulta;

    -- Mostrar datos de la venta
    DBMS_OUTPUT.PUT_LINE('Venta ID: ' || vVenta.idVenta);
    DBMS_OUTPUT.PUT_LINE('Cliente ID: ' || vVenta.idCliente);
    DBMS_OUTPUT.PUT_LINE('Fecha: ' || TO_CHAR(vVenta.fechaVenta, 'DD-MON-YYYY'));

    -- Mostrar líneas de venta
    DBMS_OUTPUT.PUT_LINE('Líneas de venta:');
    FOR i IN 1..vVenta.lineas.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE(' Producto ID: ' || vVenta.lineas(i).idProducto ||
                               ', Cantidad: ' || vVenta.lineas(i).cantidad);
    END LOOP;

    -- Mostrar total de la venta
    DBMS_OUTPUT.PUT_LINE('Total Venta: ' || vVenta.TOTAL_VENTA);
END;

```

Below the script editor, a status bar indicates: "Salida de Script x" and "Tarea terminada en 0,078 segundos". At the bottom, a message states: "Procedure MOSTRARVENTA compilado".