Tarea 5

EJERCICIO 1

Crear una base de datos llamada DBJefeHijo con DB40 con la siguiente información:

```
public static void main(String[] args){
        File fichero=new File("baseDatos");
        fichero.delete();
        /*Este código anterior lo ponemos por si la base de datos ya existiera y quisiéramos
        empezar desde el principio.*/
        ObjectContainer baseDatos=Db4oEmbedded.openFile("BDJefeHijo ");
        baseDatos.store(new Jefe("Ángel", 5, 53,new Hijo("Gustavo", 7)));
        baseDatos.store(new Jefe("Nieves", 3, 45,new Hijo("Iván", 3)));
        baseDatos.store(new Jefe("Jesús", 3, 5,new Hijo("Noelia", 3)));
        baseDatos.store(new Jefe("Dolores", 5,63,new Hijo("Sergio", 7)));
        baseDatos.store(new Jefe("Vicki", 3, 5,null));
        baseDatos.store(new Jefe("Fátima", 5,63,new Hijo("Lidia", 27)));
        baseDatos.store(new Jefe("Juan Luís", 3, 5,null));
        baseDatos.store(new Jefe("Elena", 1,42,new Hijo("David", 19)));
        baseDatos.store(new Jefe("Miguel", 20,45,new Hijo("Paula", 3)));
        baseDatos.store(new Jefe("Jesús", 19, 44,new Hijo("Rubén", 12)));
        baseDatos.close();
}
```

Realizar las siguientes consultas:

1. **Visualizar los jefes que tengan más de 55 años.**
2. **Modificar la edad de Miguel incrementando su edad un año más.**
3. **Borrar los jefes que llevan más de 6 años en la empresa.**
4. **Visualizar todos los jefes que quedan, incluidos sus hijos, que no han sido borrados anteriormente.**

En primer lugar, se ha procedido a crear la clase Jefe y la clase Hijo:

Clase Jefe:

```java
/**
 * Representa al objeto Jefe.
 * Cada jefe tiene un nombre, años de antigüedad en la empresa, edad y un posible hijo.
 */
public class Jefe {

    /** Nombre del jefe. */
    String nombre;

    /** Años de antigüedad en la empresa. */
    int añosAntiguedad;

    /** Edad del jefe. */
    int edad;

    /** Hijo del jefe (puede ser null si no tiene hijos). */
    Hijo hijo;

    /**
     * Constructor para crear un nuevo objeto Jefe.
     *
     * @param nombre Nombre del jefe.
     * @param añosAntiguedad Años que lleva en la empresa.
     * @param edad Edad del jefe.
     * @param hijo Hijo del jefe (puede ser null si no tiene hijos).
     */
    public Jefe(String nombre, int añosAntiguedad, int edad, Hijo hijo) {
        this.nombre = nombre;
        this.añosAntiguedad = añosAntiguedad;
        this.edad = edad;
        this.hijo = hijo;
    }

    //Métodos getter y setter

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getAñosAntiguedad() {
        return añosAntiguedad;
    }

    public void setAñosAntiguedad(int añosAntiguedad) {
        this.añosAntiguedad = añosAntiguedad;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }

    public Hijo getHijo() {
        return hijo;
    }

    public void setHijo(Hijo hijo) {
        this.hijo = hijo;
    }

    /**
     * Devuelve una representación en cadena del objeto Jefe.
     *
     * @return Cadena con la información del jefe, incluyendo su nombre, edad,
     *         años en la empresa y datos de su hijo (si tiene).
     */
    @Override
    public String toString() {
        return "Jefe: " + nombre + ", Años en empresa: " + añosAntiguedad + ", Edad: " + edad
                + (hijo != null ? ", Hijo: " + hijo.nombre + " (" + hijo.edad + " años)" : ", Sin hijos");
    }
}
```

Clase Hijo:

```java
/**
 * Representa a un hijo de un jefe dentro de la empresa.
 * Cada hijo tiene un nombre y una edad.
 */
public class Hijo {

    /** Nombre del hijo. */
    String nombre;

    /** Edad del hijo. */
    int edad;

    /**
     * Constructor para crear un nuevo objeto Hijo.
     *
     * @param nombre Nombre del hijo.
     * @param edad Edad del hijo.
     */
    public Hijo(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

    //Métodos getter y setter:

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }

    /**
     * Devuelve una representación en cadena del objeto Hijo.
     *
     * @return Cadena con la información del hijo.
     */
    @Override
    public String toString() {
        return "Hijo: " + nombre + " (" + edad + " años)";
    }
}
```

Clase principal

Antes de iniciar, se elimina cualquier archivo de la base de datos existente para evitar inconsistencias. Luego, se almacenan varios objetos de tipo Jefe, algunos con hijos y otros sin hijos.

```java
import com.db4o.*;
import com.db4o.query.Query;
import java.io.File;

/**
 * Clase principal que maneja una base de datos de objetos con información de Jefes y sus Hijos.
 * Utiliza db4o para almacenar, consultar, modificar y eliminar datos.
 */
public class Main {


    public static void main(String[] args) {
        // Eliminar el archivo de la base de datos si ya existe para empezar desde cero.
        File fichero = new File("BDJefeHijo");
        fichero.delete();

        // Abrir la base de datos y almacenar algunos objetos Jefe con hijos
        ObjectContainer baseDatos = Db4oEmbedded.openFile("BDJefeHijo");

        baseDatos.store(new Jefe("Ángel", 5, 53, new Hijo("Gustavo", 7)));
        baseDatos.store(new Jefe("Nieves", 3, 45, new Hijo("Iván", 3)));
        baseDatos.store(new Jefe("Jesús", 3, 5, new Hijo("Noelia", 3)));
        baseDatos.store(new Jefe("Dolores", 5, 63, new Hijo("Sergio", 7)));
        baseDatos.store(new Jefe("Vicki", 3, 5, null));
        baseDatos.store(new Jefe("Fátima", 5, 63, new Hijo("Lidia", 27)));
        baseDatos.store(new Jefe("Juan Luís", 3, 5, null));
        baseDatos.store(new Jefe("Elena", 1, 42, new Hijo("David", 19)));
        baseDatos.store(new Jefe("Miguel", 20, 45, new Hijo("Paula", 3)));
        baseDatos.store(new Jefe("Jesús", 19, 44, new Hijo("Rubén", 12)));

        baseDatos.close(); // Cerrar base de datos para asegurar la escritura.
```

Para realizar las siguientes consultas procedemos a abrir de nuevo la base de datos:

```java
// Reabrir la base de datos para realizar consultas y modificaciones
baseDatos = Db4oEmbedded.openFile("BDJefeHijo");
```

1. Visualizar los jefes que tengan más de 55 años.

```java
// 1. Visualizar los jefes con más de 55 años
System.out.println("Jefes con más de 55 años:");
Query query1 = baseDatos.query();
query1.constrain(Jefe.class);
query1.descend("edad").constrain(55).greater();
ObjectSet<Jefe> resultado1 = query1.execute();
for (Jefe j : resultado1) {
    System.out.println(j);
}
```

2. **Modificar la edad de Miguel incrementando su edad un año más.**

```
// 2. Modificar la edad de Miguel
Query query2 = baseDatos.query();
query2.constrain(Jefe.class);
query2.descend("nombre").constrain("Miguel");
ObjectSet<Jefe> resultado2 = query2.execute();
for (Jefe j : resultado2) {
    j.edad += 1; // Aumentar la edad en 1 año
    baseDatos.store(j); // Guardar el cambio en la base de datos
}
```

3. **Borrar los jefes que llevan más de 6 años en la empresa.**

```
// 3. Borrar los jefes que llevan más de 6 años en la empresa
Query query3 = baseDatos.query();
query3.constrain(Jefe.class);
query3.descend("añosAntiguedad").constrain(6).greater();
ObjectSet<Jefe> resultado3 = query3.execute();
for (Jefe j : resultado3) {
    System.out.println("\nEl Jefe con nombre " + j.nombre + " con " + j.añosAntiguedad + " años de antigüedad ha sido borrado.");
    baseDatos.delete(j); // Eliminar el objeto de la base de datos
}
```

4. **Visualizar todos los jefes que quedan, incluidos sus hijos, que no han sido borrados anteriormente.**

```
// 4. Visualizar todos los jefes restantes
System.out.println("\nJefes restantes:");
ObjectSet<Jefe> resultadoFinal = baseDatos.query(Jefe.class);
for (Jefe j : resultadoFinal) {
    System.out.println(j);
}
```

Finalmente cerramos la base de datos:

```
baseDatos.close(); // Cerrar BD al final
```

Resultado:

```
run:
Jefes con mas de 55 a�os:
Jefe: Dolores, A�os en empresa: 5, Edad: 63, Hijo: Sergio (7 a�os)
Jefe: F�tima, A�os en empresa: 5, Edad: 63, Hijo: Lidia (27 a�os)

El Jefe con nombre Miguelcon 20 a�os de antiguedad ha sido borrado.

El Jefe con nombre Jes�scon 19 a�os de antiguedad ha sido borrado.

Jefes restantes:
Jefe: �ngel, A�os en empresa: 5, Edad: 53, Hijo: Gustavo (7 a�os)
Jefe: Nieves, A�os en empresa: 3, Edad: 45, Hijo: Iv�n (3 a�os)
Jefe: Jes�s, A�os en empresa: 3, Edad: 5, Hijo: Noelia (3 a�os)
Jefe: Dolores, A�os en empresa: 5, Edad: 63, Hijo: Sergio (7 a�os)
Jefe: Vicki, A�os en empresa: 3, Edad: 5, Sin hijos
Jefe: F�tima, A�os en empresa: 5, Edad: 63, Hijo: Lidia (27 a�os)
Jefe: Juan Lu�s, A�os en empresa: 3, Edad: 5, Sin hijos
Jefe: Elena, A�os en empresa: 1, Edad: 42, Hijo: David (19 a�os)
BUILD SUCCESSFUL (total time: 0 seconds)
```

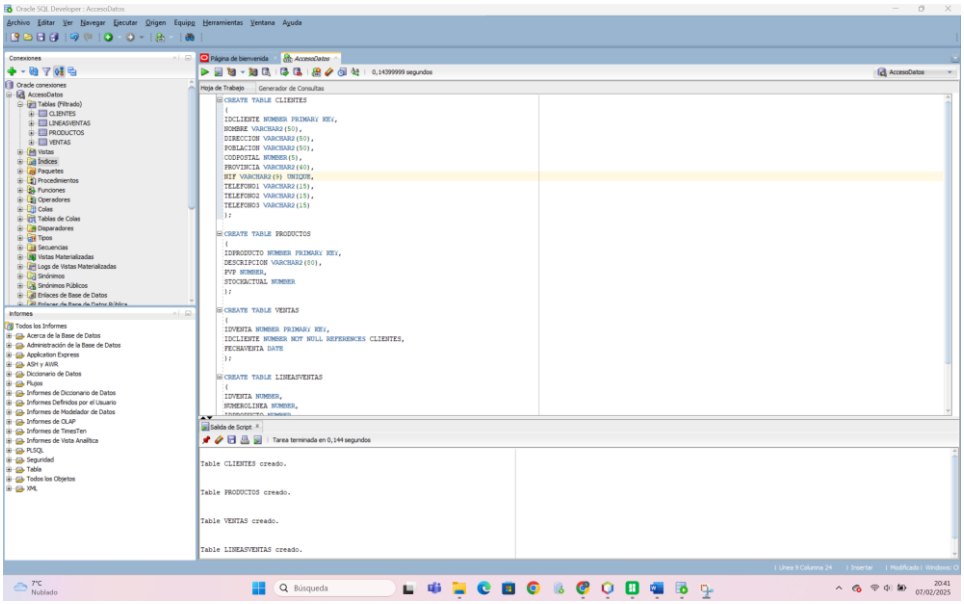
EJERCICIO 2

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));
```

Tarea 5

En primer lugar creamos una nueva conexión y las tablas correspondientes:



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

CREATE TYPE TIP_TELEFONOS AS VARRAY(3) OF VARCHAR2(15);

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

CREATE TYPE TIP_DIRECCION AS OBJECT (

CALLE VARCHAR2(50),

POBLACIÓN VARCHAR2(50),

CODPOSTAL NUMBER(5),

PROVINCIA VARCHAR2(40));

CREATE TYPE TIP_CLIENTE AS OBJECT(

IDCLIENTE NUMBER,

NOMBRE VARCHAR2(50),

DIREC TIP_DIRECCION,

NIF VARCHAR2(9),

TELEF TIP_TELEFONOS);

CREATE TYPE TIP_PRODUCTO AS OBJECT(

IDPRODUCTO NUMBER,

DESCRIPCION VARCHAR2(80),

PVP NUMBER,

STOCKACTUAL NUMBER);

```
CREATE TYPE TIP_LINEAVENTA AS OBJECT(
NUMEROLINEA NUMBER,
IDPRODUCTO REF TIP_PRODUCTO,
CANTIDAD NUMBER);
```

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

```
CREATE TYPE TIP_LINEAS_VENTA AS TABLE OF TIP_LINEAVENTA;
```

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

```
CREATE TYPE TIP_VENTA AS OBJECT (
IDVENTA NUMBER,
IDCLIENTE REF TIP_CLIENTE,
FECHAVENTA DATE,
LINEAS TIP_LINEAS_VENTA,
MEMBER FUNCTION TOTAL_VENTA RETURN NUMBER);
```

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.**

```
CREATE OR REPLACE TYPE BODY TIP_VENTA AS
MEMBER FUNCTION TOTAL_VENTA RETURN NUMBER IS
TOTAL NUMBER:=0;
LINEA TIP_LINEAVENTA;
PRODUCT TIP_PRODUCTO;
BEGIN
FOR I IN 1..LINEAS.COUNT LOOP
LINEA:=LINEAS(I) ;
SELECT DEREF(LINEA.IDPRODUCTO) INTO PRODUCT FROM DUAL;
TOTAL:=TOTAL + LINEA.CANTIDAD * PRODUCT.PVP;
END LOOP;
RETURN TOTAL;
END;
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 TIP_CLIENTE(
IDCLIENTE PRIMARY KEY,
NIF UNIQUE);
CREATE TABLE TABLA_PRODUCTOS OF TIP_PRODUCTO(
IDPRODUCTO PRIMARY KEY);
CREATE TABLE TABLA_VENTAS OF TIP_VENTA(
IDVENTA PRIMARY KEY)
NESTED TABLE LINEAS STORE AS TABLA_LINEAS;
```

7. **Inserta dos clientes y cinco productos.**

```
INSERT INTO TABLA_CLIENTES VALUES
(1, 'Pepe Perez' ,
TIP_DIRECCION ('C/ Falsa 1', 'Salamanca' , '37007', 'Salamanca'),
'12345678A', TIP_TELEFONOS ( '927131211', '658741589'));
INSERT INTO TABLA_CLIENTES VALUES
(2, 'Maria Rodriguez' ,
TIP_DIRECCION ('C/Falsa 2', 'Plasencia' , '10600', 'Caceres' ) ,
'76767667F', TIP_TELEFONOS ( '94980009', '646512984'));
INSERT INTO TABLA_PRODUCTOS VALUES (1, 'PS5', 600, 15);
INSERT INTO TABLA_PRODUCTOS VALUES (2, 'PATINETE ELECTRICO', 350, 7);
INSERT INTO TABLA_PRODUCTOS VALUES (3, 'XBOX SERIES S', 450, 5);
INSERT INTO TABLA_PRODUCTOS VALUES (4, 'IPHONE 15', 700, 20);
INSERT INTO TABLA_PRODUCTOS VALUES (5, 'SAMSUN S9', 500.99, 10);
```

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

```
INSERT INTO TABLA_VENTAS
SELECT 1, REF(C), SYSDATE, TIP_LINEAS_VENTA()
FROM TABLA_CLIENTES C WHERE C.IDCLIENTE=1;
```

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).**

```
INSERT INTO TABLE (SELECT V.LINEAS FROM TABLA_VENTAS V WHERE V.IDVENTA=1)
(SELECT 1, REF(P), 1 FROM TABLA_PRODUCTOS P WHERE P.IDPRODUCTO=1);

INSERT INTO TABLE (SELECT V.LINEAS FROM TABLA_VENTAS V WHERE V.IDVENTA=1)
(SELECT 2, REF(P), 2 FROM TABLA_PRODUCTOS P WHERE P.IDPRODUCTO=2);
```

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

```
INSERT INTO TABLA_VENTAS
SELECT 2, REF(C), SYSDATE, TIP_LINEAS_VENTA()
FROM TABLA_CLIENTES C WHERE C.IDCLIENTE=1;
```

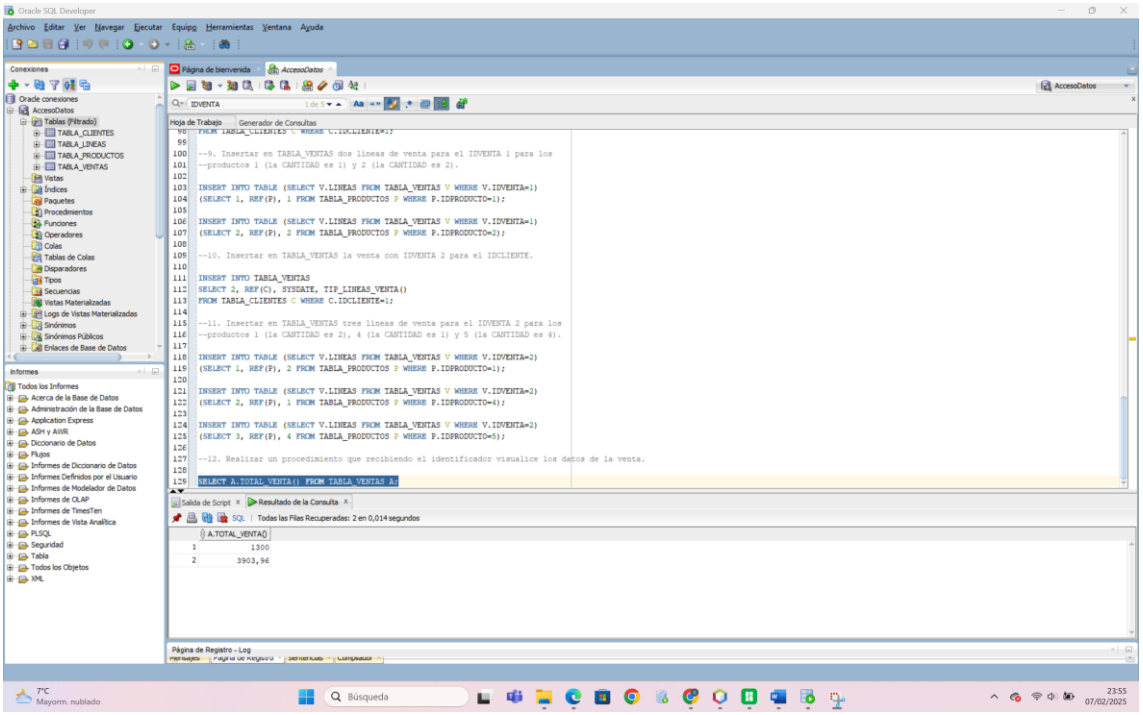
11. **Insertar en TABLA_VENTAS tres líneas de venta para el IDVENTA 2 para los productos 1 (la CANTIDAD es 2), 4 (la CANTIDAD es 1) y 5 (la CANTIDAD es 4).**

```
INSERT INTO TABLE (SELECT V.LINEAS FROM TABLA_VENTAS V WHERE V.IDVENTA=2)
(SELECT 1, REF(P), 2 FROM TABLA_PRODUCTOS P WHERE P.IDPRODUCTO=1);

INSERT INTO TABLE (SELECT V.LINEAS FROM TABLA_VENTAS V WHERE V.IDVENTA=2)
(SELECT 2, REF(P), 1 FROM TABLA_PRODUCTOS P WHERE P.IDPRODUCTO=4);

INSERT INTO TABLE (SELECT V.LINEAS FROM TABLA_VENTAS V WHERE V.IDVENTA=2)
(SELECT 3, REF(P), 4 FROM TABLA_PRODUCTOS P WHERE P.IDPRODUCTO=5);
```

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

```
SELECT A.TOTAL_VENTA() FROM TABLA_VENTAS A;
```

Tarea 5


--9. Insertar en TABLA_VENTAS dos lineas de venta para el IDVENTA 1 para los
--productos 1 (la CANTIDAD es 1) y 2 (la CANTIDAD es 2).
INSERT INTO TABLE (SELECT V.LINEAS FROM TABLA_VENTAS V WHERE V.IDVENTA=1)
(SELECT 1, REF(P), 1 FROM TABLA_PRODUCTOS P WHERE P.IDPRODUCTO=1);
INSERT INTO TABLE (SELECT V.LINEAS FROM TABLA_VENTAS V WHERE V.IDVENTA=1)
(SELECT 2, REF(P), 2 FROM TABLA_PRODUCTOS P WHERE P.IDPRODUCTO=2);
--10. Insertar en TABLA_VENTAS la venta con IDVENTA 2 para el IDCLIENTE.
INSERT INTO TABLA_VENTAS
SELECT 2, REF(C), SYSDATE, TIP_LINEAS_VENTA()
FROM TABLA_CLIENTES C WHERE C.IDCLIENTE=1;
--11. Insertar en TABLA_VENTAS tres lineas 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).
INSERT INTO TABLE (SELECT V.LINEAS FROM TABLA_VENTAS V WHERE V.IDVENTA=2)
(SELECT 1, REF(P), 2 FROM TABLA_PRODUCTOS P WHERE P.IDPRODUCTO=1);
INSERT INTO TABLE (SELECT V.LINEAS FROM TABLA_VENTAS V WHERE V.IDVENTA=2)
(SELECT 2, REF(P), 1 FROM TABLA_PRODUCTOS P WHERE P.IDPRODUCTO=4);
INSERT INTO TABLE (SELECT V.LINEAS FROM TABLA_VENTAS V WHERE V.IDVENTA=2)
(SELECT 3, REF(P), 4 FROM TABLA_PRODUCTOS P WHERE P.IDPRODUCTO=5);
--12. Realizar un procedimiento que recibiendo el identificador visualice los datos de la venta.
SELECT A.TOTAL_VENTA() FROM TABLA_VENTAS A;
A.TOTAL_VENTA()
1 1300
2 3903,96