



ÍNDICE

1. INTRODUCCIÓN AL PROBLEMA.
2. GLOSARIO DE TÉRMINOS.
3. VISIÓN GENERAL DEL SISTEMA.
4. CATÁLOGO DE REQUISITOS.
5. PRUEBAS DE ACEPTACIÓN.
6. MODELADO CONCEPTUAL.
7. MATRIZ DE TRAZABILIDAD.



1. INTRODUCCIÓN AL PROBLEMA.

La empresa solicitante de los servicios es una cadena de ópticas **Óptica Maguilla**, conformada por 3 franquicias. El propietario se encuentra con el problema de usar un sistema informático antiguo y obsoleto, que no le permite desempeñar correctamente su actividad laboral, lo que se traduce en retrasos, pérdidas económicas y de tiempo.

Con esto en mente, el propietario se puso en contacto con nuestro equipo para encontrar solución a los defectos de su sistema informático. Tras realizar una entrevista inicial, se llegaron a una serie de conclusiones sobre los problemas a abordar y se decidió crear un sistema desde cero con las funcionalidades que realmente se necesitan y solventando problemas que el anterior sistema no solucionaba.

2. GLOSARIO DE TÉRMINOS.

Cerca: Término que hace referencia a la hipermetropía.

Cilindro: El cilindro se expresa en dioptrías. Corrige el astigmatismo. La indicación del cilindro no siempre es obligatoria pero sí lo es en caso de deformación corneal, puede tener signo positivo o negativo.

Convergencia ocular: La convergencia ocular es un proceso que ocurre cuando queremos mirar objetos o puntos cercanos, para ello ambos ojos deben dirigirse hacia el nasal.

Descentramiento: Distancia desde el centro de la lente hasta el centro del ojo, si no están alineados pueden producir problemas de visión, hay cierto grado de tolerancia, dependiendo de la enfermedad ocular interesan diferentes configuraciones


Dioptrías: La dioptría es la unidad que con valores positivos o negativos expresa el poder de refracción de una lente o potencia de la lente.

Diplopía: Es una alteración visual que consiste en la percepción de visión doble. Esta alteración de la visión puede ser horizontal, diagonal u oblicua

Distancia cerca: Hace referencia a las dioptrías necesarias para corregir la hipermetropía.

Distancia lejos: Hace referencia a las dioptrías necesarias para corregir la miopía.

Eje: El valor del eje indica cómo deben colocarse las lentes en la montura, dónde se encuentran las partes más gruesas y dónde se encuentran las más finas. Dicho valor está comprendido entre 0 y 180 y no tiene signo positivo ni negativo, pero se indica siempre en grados. No puede ser inferior a 0 ni superior a 180. Los valores son independientes para cada ojo, es decir, puede tener un eje para el ojo derecho y otro para el izquierdo.



Prisma: Los prismas se utilizan en óptica para corregir las anomalías de convergencia ocular susceptibles de provocar una diplopía.

Esfera: La esfera se expresa en dioptrías. La esfera caracteriza el grado de miopía cuando va precedida por el signo - y el grado de hipermetropía cuando va precedida por el signo +.

Franquicia: Sistema de venta de productos de una firma comercial en una tienda de otro propietario y bajo ciertas condiciones económicas.

Graduación: La graduación de la vista determina la cantidad de corrección óptica que necesita una persona que padece uno o varios problemas refractivos (miopía, hipermetropía y/o astigmatismo y presbicia o vista cansada).

Lejos: Termino que hace referencia a la miopía.

Lentes oftálmicas y montura: Una lente oftálmica (o lente para gafa) es un objeto transparente compuesto por dos superficies, en la que al menos una de ellas, está curvada, una montura es armazón sobre la que se sostiene las lentes oftálmicas.

Miopía: La miopía es un defecto visual que afecta esencialmente a la visión de objetos lejanos. La imagen de los objetos observados se forma por delante de la retina de modo la persona miope ve mal de lejos y bien de cerca.

Queratometría: Es una prueba realizada a un paciente en la que se determinan los parámetros de su córnea, tales como la medida de sus radios de curvatura de sus superficies.

Tonometría: Procedimiento consistente en medir la tensión de un líquido que se encuentra alojado en una cavidad. Por lo general se utiliza para la determinación de la presión intraocular (PIO), que es la presión a la que se encuentra el humor acuoso, el líquido ubicado en el interior del ojo.



3. VISIÓN GENERAL DEL SISTEMA.

Se trata de un sistema que se encarga de gestionar datos relacionados con los clientes, gestión de stock de productos.

El sistema debe poder almacenar datos de los clientes, poder realizar encargos de productos, guardar datos de las graduaciones, y enviar ofertas a los clientes.

Otra de las funcionalidades del sistema consiste en el control de Stock de los productos de la empresa, a parte de poder ver la cantidad de objetos que se encuentra, se le facilitará al propietario del sistema la revisión de los productos, haciendo que el sistema avise cuando los niveles de ciertos objetos bajen de un umbral.

Este sistema diferenciará entre dos tipos de usuarios, el empleado y el propietario de la empresa, el empleado que precisará de información menos relevante que la del propietario, el propietario dispondrá además de herramientas necesarias para gestionar a sus empleados y clientes de diversas maneras.

4. CATÁLOGO DE REQUISITOS

Objetivo

Como: Propietario de la óptica

Quiero: Gestionar los clientes y el stock de las tres franquicias que componen mi empresa

Para: Agilizar la organización de la empresa

Requisitos de Información

RI-001 Información sobre los clientes

Como: Propietario de la óptica

Quiero: Disponer de la información correspondiente a los *clientes*: nombre, apellidos, dirección, sexo, código postal, provincia, localidad, razón por la que viene a la óptica, profesión, teléfono, DNI, email, firma y puede tener una fotografía

Para: Tener control de los clientes

RI-002 Información sobre la graduación

Como: Propietario de la óptica

Quiero: Disponer de la información correspondiente a la graduación: Graduado por, Atendido Por, y, para cada ojo, esf.ojo, cilindro, eje, esf.cerca, adición, lejos, cerca, SC, CC, queratometría y prisma

Para: Tener control de la salud ocular de los clientes

RI-003 Información sobre los encargos

Como: Propietario de la óptica

Quiero: Disponer de la información correspondiente a los *encargos realizados por los clientes*: id de referencia, descripción, unidades, precio unitario, precio total.

Para: Tener control de los encargos

RI-004 Información sobre las franquicias

Como: Propietario de la óptica

Quiero: Disponer de la información correspondiente a las *franquicias*:

nombre, dirección, dueño

Para: Tener control de las franquicias

RI-005 Ofertas

Como: Propietario de la óptica

Quiero: Poder realizar ofertas con los nuevos catálogos que los fabricantes.

Para: Aumentar las ventas

RI-006 Productos

Como: Propietario de la óptica

Quiero: Disponer de la información correspondiente a los *productos*: id. Los productos pueden ser monturas, lentillas, lentes...

Para: Conocer los productos

Reglas de Negocio

RN-001 Datos del cliente

Como: Propietario de la óptica

Quiero: Que se cumpla la siguiente regla de negocio:

Todo cliente debe tener asignada una fecha de nacimiento, nombre, apellido, sexo y una firma para ser creado en el sistema

Para: Cumplir requisitos legales y registrar un usuario

RN-002 Stock en cantidad negativa

Como: Propietario de la óptica

Quiero: Que se cumpla la siguiente regla de negocio:

Un producto puede ser pedido sin que esté en stock, por tanto, cuando esto suceda su cantidad debe ser negativa

Para: Llevar un recuento de los productos que se han pedido en estas condiciones

RN-003 Avisos de stock

Como: Propietario de la óptica

Quiero: Que se cumpla la siguiente regla de negocio:

Si el stock de un producto baja de una cantidad marcada, se enviará un aviso

Para: Evitar que los productos no se queden sin stock

Requisitos Funcionales

RF-001 Perfil de cliente

Como: Propietario de la óptica

Quiero: Obtener un perfil completo de los clientes de la óptica, con sus datos y sus historiales

Para: Llevar seguimiento de los clientes

RF-002 Listado de productos

Como: Propietario de la óptica

Quiero: Obtener un listado completo de los productos en stock

Para: Llevar seguimiento de los productos en stock

RF-003 Hoja de graduación

Como: Propietario de la óptica

Quiero: obtener una hoja con la información ocular del cliente

Para: Conocer el tipo de producto que necesita

5. PRUEBAS DE ACEPTACIÓN

RN-001 Datos del cliente – Pruebas de aceptación

El usuario se crea cuando se rellenan los campos de nombre, apellidos, sexo, firma y fecha de nacimiento.

El usuario no se crea si se rellena el campo de nombre, pero no el de fecha de nacimiento ni el de apellidos.

El usuario no se crea si alguno de los campos nombre, apellidos, sexo, firma o fecha de nacimiento faltan.

El usuario se crea si se rellenan todos los datos, incluidos nombre, apellidos, sexo, firma y fecha de nacimiento.

RN-002 Almacén en cantidad negativa – Pruebas de aceptación

Si se pide un producto que se encuentra en stock, se restan las unidades pedidas.

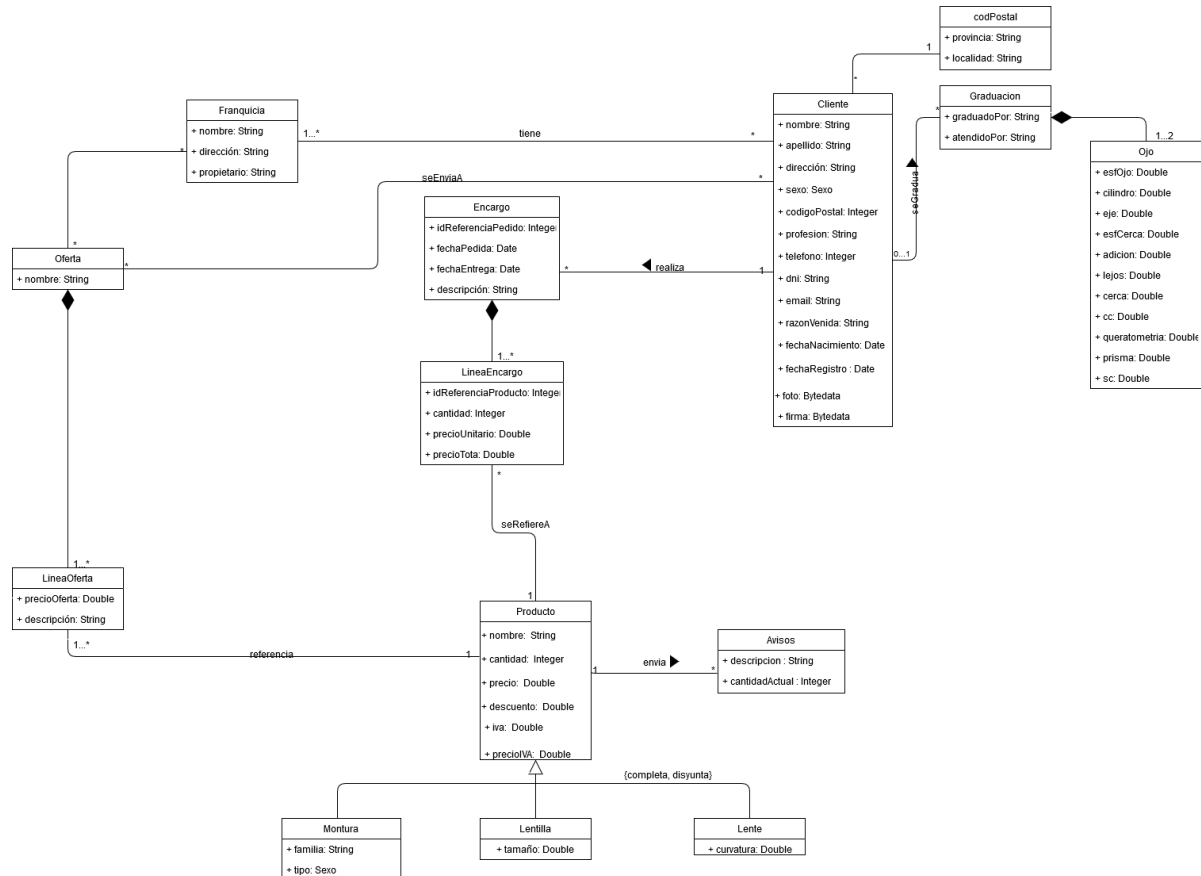
Si se pide un producto y si no hay stock del producto, la cantidad podrá ser negativa.

RN-003 Avisos de Stock – Pruebas de aceptación

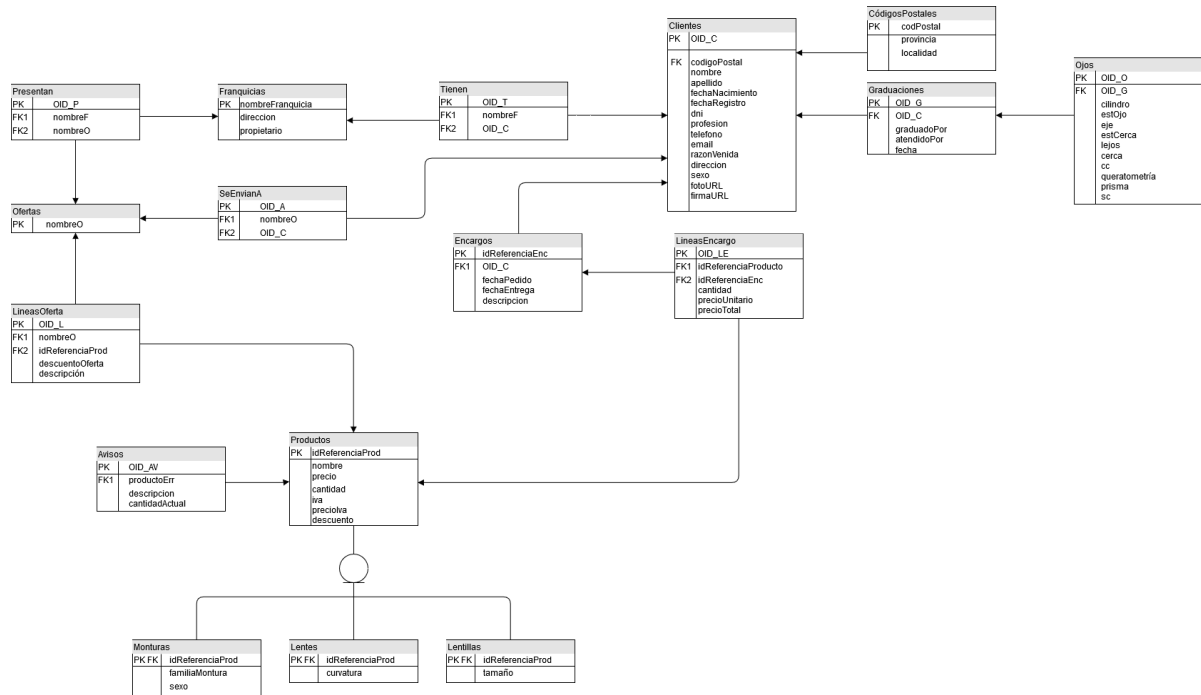
Si un producto se encuentra por encima de la cantidad marcada, no pasará nada.

Si un producto se encuentra por debajo de la cantidad marcada, se enviará un aviso a las franquicias

6. MODELO CONCEPTUAL



7. MODELO RELACIONAL



8. MATRIZ DE TRAZABILIDAD

	RI-001	RI-002	RI-003	RI-004	RI-005	RI-006	RN-001	RN-002	RN-003	RF-001	RF-002	RF-003
Cliente	✓	✓	✓	✓	✓		✓	✓		✓		✓
Graduación		✓										✓
Ojo		✓										✓
Encargo			✓					✓				
L.Encargo			✓					✓				
Producto			✓		✓	✓		✓	✓		✓	
Franquicia				✓	✓							
Oferta					✓							
L.Oferta					✓							
realiza			✓					✓				
seGradua		✓										
seEnviaA					✓							
tiene				✓								
contiene												
referencia					✓							
ofrece					✓							
Avisos									✓			
seRefiereA			✓					✓				

9. SCRIPTS - CREACION DE LAS TABLAS, SECUENCIAS Y TRIGGER DE SECUENCIA

-- Borrado de las tablas

DROP TABLE Avisos;
DROP TABLE Presentan;
DROP TABLE SeEnvianA;
DROP TABLE LineasOferta;
DROP TABLE Ofertas;
DROP TABLE Tienen;
DROP TABLE Franquicias;
DROP TABLE Lentillas;
DROP TABLE Lentes;
DROP TABLE Monturas;
DROP TABLE LineasEncargo;
DROP TABLE Productos;
DROP TABLE Encargos;
DROP TABLE Ojos;
DROP TABLE Graduaciones;
DROP TABLE Clientes;
DROP TABLE CodigosPostales;

-- Borrado de las secuencias

DROP SEQUENCE SEC_IDREFERENCIAPRODUCTO;
DROP SEQUENCE SEC_AVISOS;
DROP SEQUENCE SEC_LineasOferta;
DROP SEQUENCE SEC_Presentan;
DROP SEQUENCE SEC_SeEnvianA;
DROP SEQUENCE SEC_Tienen;
DROP SEQUENCE SEC_LINEAENCARGO;
DROP SEQUENCE SEC_OJO;
DROP SEQUENCE SEC_GRADUACION;
DROP SEQUENCE SEC_CLIENTE;
DROP SEQUENCE SEC_IDREFERENCIA_ENCARGO;

-- Creación de las tablas

CREATE TABLE CodigosPostales (
 codpostal **SMALLINT**,
 provincia **VARCHAR(50) NOT NULL**,
 localidad **VARCHAR(50) NOT NULL**,

PRIMARY KEY (codpostal)
);

CREATE TABLE Clientes (
 OID_C **SMALLINT**,
 codigoPostal **SMALLINT NOT NULL**,
 nombre **VARCHAR(50) NOT NULL**,
 apellido **VARCHAR(50) NOT NULL**,
 fechaNacimiento **DATE NOT NULL**,
 fechaRegistro **DATE NOT NULL**,
 dni **VARCHAR(9) UNIQUE**,
 profesion **VARCHAR(50)**,
 telefono **SMALLINT**,
 email **VARCHAR(50)**,
 razonVenida **VARCHAR(100)**,
 direccion **VARCHAR(50)**,
 sexo **VARCHAR(50) NOT NULL**,
 fotoURL **VARCHAR(100)**,
 firmaURL **VARCHAR(100) NOT NULL**,

CONSTRAINT SEXOFALLA **CHECK** (sexo **IN** ('Hombre','Mujer','Otro')),
 CONSTRAINT EMAILFALLA **CHECK** (REGEXP_LIKE(email, '.*@.*\..*')),
 CONSTRAINT DNIFALLA **CHECK** (REGEXP_LIKE(dni, '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][A-Z]')),
 CONSTRAINT URLFIRMAFALLA **CHECK** (REGEXP_LIKE(firmaURL, 'https://.*\..*/*')),
 CONSTRAINT URLFOTOFALLA **CHECK** (REGEXP_LIKE(fotoURL, 'https://.*\..*/*')),
 PRIMARY KEY (OID_C),
 FOREIGN KEY (codigoPostal) **REFERENCES** CodigosPostales(codpostal)
);

CREATE TABLE Graduaciones (
 OID_G **SMALLINT**,
 OID_C **SMALLINT NOT NULL**,
 graduadoPor **VARCHAR(50)**,
 atendidoPor **VARCHAR(50)**,
 fecha **DATE NOT NULL**,

 PRIMARY KEY (OID_G),
 FOREIGN KEY (OID_C) **REFERENCES** Clientes(OID_C) **ON DELETE CASCADE**
);

```

CREATE TABLE Ojos (
  OID_O SMALLINT,
  OID_G SMALLINT NOT NULL,
  cilindro NUMBER(12,2),
  estOjo NUMBER(12,2),
  eje NUMBER(12,2),
  esteCerca NUMBER(12,2),
  lejos NUMBER(12,2),
  cerca NUMBER(12,2),
  cc NUMBER(12,2),
  queratometria NUMBER(12,2),
  prisma NUMBER(12,2),
  sc NUMBER(12,2),

  CONSTRAINT CCDECIMAL CHECK(cc >= 0 AND cc <= 1),
  PRIMARY KEY (OID_O),
  FOREIGN KEY (OID_G) REFERENCES Graduaciones(OID_G) ON DELETE CASCADE
);

```

```

CREATE TABLE Encargos (
  idReferenciaEncargo SMALLINT,
  OID_C SMALLINT NOT NULL,
  fechaPedido DATE NOT NULL,
  fechaEntrega DATE,
  descripcion VARCHAR(100),

  PRIMARY KEY (idReferenciaEncargo),
  FOREIGN KEY (OID_C) REFERENCES Clientes(OID_C) ON DELETE CASCADE
);

```

```

CREATE TABLE Productos (
  idReferenciaProducto SMALLINT,
  nombre VARCHAR(50),
  cantidad SMALLINT NOT NULL,
  precio NUMBER (12,2)NOT NULL,
  iva NUMBER (12,2)NOT NULL,
  precioIVA NUMBER(12,2),
  descuento NUMBER(12,2) DEFAULT '0,0',

  CONSTRAINT PRECIONEGATIVO_P CHECK(precio >= 0),
  CONSTRAINT DESCUENTOPORCENTAJEPRODUCTOS CHECK(descuento >= 0 AND
descuento <=1),
  CONSTRAINT IVAPORCENTAJE CHECK(iva >= 0 AND iva <=1),
  PRIMARY KEY (idReferenciaProducto)

```


);

```

CREATE TABLE LineasEncargo (
    OID_LE SMALLINT,
    idReferenciaProducto SMALLINT NOT NULL,
    idReferenciaEncargo SMALLINT NOT NULL,
    cantidad SMALLINT,
    precioUnitario NUMBER(12,2),

    CONSTRAINT CANTIDADNEGATIVA_LE CHECK(cantidad > 0),
    CONSTRAINT PRECIONEGATIVO_LE CHECK(precioUnitario >= 0),
    PRIMARY KEY (OID_LE),
    FOREIGN KEY (idReferenciaProducto) REFERENCES Productos(idReferenciaProducto) ON
DELETE CASCADE,
    FOREIGN KEY (idReferenciaEncargo) REFERENCES Encargos(idReferenciaEncargo) ON
DELETE CASCADE
);
ALTER TABLE LineasEncargo ADD (precioTotal number(12,2) AS (precioUnitario *
cantidad)) ;
CREATE TABLE Monturas (
    idReferenciaProducto SMALLINT,
    familiaMontura VARCHAR(50),
    sexo VARCHAR(50),

    CONSTRAINT ERRORSEXO_MONTURAS CHECK(sexo IN ('Hombre','Mujer','Otro')),
    PRIMARY KEY (idReferenciaProducto),
    FOREIGN KEY (idReferenciaProducto) REFERENCES Productos(idReferenciaProducto) ON
DELETE CASCADE
);
-----
CREATE TABLE Lentes (
    idReferenciaProducto SMALLINT,
    curvatura NUMBER(12,2),

    PRIMARY KEY (idReferenciaProducto),
    FOREIGN KEY (idReferenciaProducto) REFERENCES Productos(idReferenciaProducto) ON
DELETE CASCADE
);
-----
CREATE TABLE Lentillas (
    idReferenciaProducto SMALLINT,
    tamaño NUMBER(12,2),

    CONSTRAINT TAMAÑONEGATIVO CHECK(tamaño > 0),
    PRIMARY KEY (idReferenciaProducto),
    FOREIGN KEY (idReferenciaProducto) REFERENCES Productos(idReferenciaProducto) ON
DELETE CASCADE
);

```

```
CREATE TABLE Franquicias (  
    nombreFranquicia VARCHAR(50),  
    direccion VARCHAR(50) NOT NULL,  
    propietario VARCHAR(50),
```

```
    PRIMARY KEY (nombreFranquicia)  
);
```

```
CREATE TABLE Tienen (  
    OID_T SMALLINT,  
    nombreFranquicia VARCHAR(50) NOT NULL,  
    OID_C SMALLINT NOT NULL,
```

```
    PRIMARY KEY (OID_T),  
    FOREIGN KEY (nombreFranquicia) REFERENCES Franquicias(nombreFranquicia) ON  
DELETE CASCADE,  
    FOREIGN KEY (OID_C) REFERENCES Clientes(OID_C) ON DELETE CASCADE  
);
```

```
CREATE TABLE Ofertas (  
    nombreOferta VARCHAR(50),
```

```
    PRIMARY KEY (nombreOferta)  
);
```

```
CREATE TABLE LineasOferta (  
    OID_LO SMALLINT,  
    nombreOferta VARCHAR(50) NOT NULL,  
    idReferenciaProducto SMALLINT NOT NULL,  
    descuentoOferta NUMBER(12,2) NOT NULL,  
    descripcion VARCHAR(100),
```

```
    PRIMARY KEY (OID_LO),  
    CONSTRAINT b_nombreOferta FOREIGN KEY (nombreOferta) REFERENCES  
Ofertas(nombreOferta),  
    FOREIGN KEY (idReferenciaProducto) REFERENCES Productos(idReferenciaProducto) ON  
DELETE CASCADE,
```

```
    CONSTRAINT DESCUENTOPORCENTAJELINEAOFERTA CHECK(descuentoOferta >= 0  
AND descuentoOferta <=1)  
);
```

```
CREATE TABLE SeEnvianA (  
    OID_SEA SMALLINT,  
    nombreOferta VARCHAR(50) NOT NULL,  
    OID_C SMALLINT NOT NULL,  
  
    PRIMARY KEY (OID_SEA),  
    FOREIGN KEY (nombreOferta) REFERENCES Ofertas(nombreOferta) ON DELETE  
CASCADE,  
    FOREIGN KEY (OID_C) REFERENCES Clientes(OID_C) ON DELETE CASCADE  
);
```

```
CREATE TABLE Presentan (  
    OID_P SMALLINT,  
    nombreOferta VARCHAR(50) NOT NULL,  
    nombreFranquicia VARCHAR(50) NOT NULL,  
  
    PRIMARY KEY (OID_P),  
    FOREIGN KEY (nombreOferta) REFERENCES Ofertas(nombreOferta) ON DELETE  
CASCADE,  
    FOREIGN KEY (nombreFranquicia) REFERENCES Franquicias(nombreFranquicia) ON  
DELETE CASCADE  
);
```

```
CREATE TABLE AVISOS (  
    OID_AV SMALLINT,  
    productoerr SMALLINT,  
    descripcion VARCHAR(50),  
    cantidadActual SMALLINT,  
  
    PRIMARY KEY(OID_AV),  
    FOREIGN KEY(productoerr) REFERENCES Productos(idReferenciaProducto) ON DELETE  
CASCADE  
);
```

-- Creación de las secuencias

CREATE SEQUENCE SEC_CLIENTE
INCREMENT BY 1
START WITH 1
MAXVALUE 99999999;

CREATE SEQUENCE SEC_GRADUACION
INCREMENT BY 1
START WITH 1
MAXVALUE 99999999;

CREATE SEQUENCE SEC_OJO
INCREMENT BY 1
START WITH 1
MAXVALUE 99999999;

CREATE SEQUENCE SEC_LINEAENCARGO
INCREMENT BY 1
START WITH 1
MAXVALUE 99999999;

CREATE SEQUENCE SEC_Tienen
INCREMENT BY 1
START WITH 1
MAXVALUE 99999999;

CREATE SEQUENCE SEC_SeEnvianA
INCREMENT BY 1
START WITH 1
MAXVALUE 99999999;

CREATE SEQUENCE SEC_Presentan
INCREMENT BY 1
START WITH 1
MAXVALUE 99999999;

CREATE SEQUENCE SEC_LineasOferta
INCREMENT BY 1
START WITH 1
MAXVALUE 99999999;

```
CREATE SEQUENCE SEC_IDREFERENCIA_ENCARGO
INCREMENT BY 1
START WITH 1
MAXVALUE 99999999;
```

```
CREATE SEQUENCE SEC_AVISOS
INCREMENT BY 1
START WITH 1
MAXVALUE 99999999;
```

```
CREATE SEQUENCE SEC_IDREFERENCIAPRODUCTO
INCREMENT BY 1
START WITH 1
MAXVALUE 99999999;
```

```
-- Creación de los trigger asociados a las secuencias
```

```
CREATE OR REPLACE TRIGGER TRIGGER_SEC_CLIENTE
BEFORE INSERT ON CLIENTES
FOR EACH ROW
DECLARE
    secuenciaCliente NUMBER := 0;
BEGIN
    SELECT SEC_CLIENTE.NEXTVAL INTO secuenciaCliente FROM DUAL;
    :NEW.OID_C := secuenciaCliente;
END;
/
```

```
CREATE OR REPLACE TRIGGER TRIGGER_SEC_GRADUACION
BEFORE INSERT ON GRADUACIONES
FOR EACH ROW
DECLARE
    secuenciaGraduacion NUMBER := 0;
BEGIN
    SELECT SEC_GRADUACION.NEXTVAL INTO secuenciaGraduacion FROM DUAL;
    :NEW.OID_G := secuenciaGraduacion;
END;
/
```

```
CREATE OR REPLACE TRIGGER TRIGGER_SEC_OJO
BEFORE INSERT ON OJOS
FOR EACH ROW
DECLARE
    secuenciaOjo NUMBER := 0;
BEGIN
    SELECT SEC_OJO.NEXTVAL INTO secuenciaOjo FROM DUAL;
    :NEW.OID_O := secuenciaOjo;
```

END;
/


```

-----
CREATE OR REPLACE TRIGGER TRIGGER_SEC_LINEAENCARGO
BEFORE INSERT ON LINEASENCARGO
FOR EACH ROW
DECLARE
    secuenciaLineaEncargo NUMBER := 0;
BEGIN
    SELECT SEC_LINEAENCARGO.NEXTVAL INTO secuenciaLineaEncargo FROM DUAL;
    :NEW.OID_LE := secuenciaLineaEncargo;
END;
/

```

```

-----
CREATE OR REPLACE TRIGGER TR_SEC_Tienen
BEFORE INSERT ON Tienen
FOR EACH ROW
DECLARE
    valorTienen NUMBER := 0;
BEGIN
    SELECT SEC_Tienen.NEXTVAL INTO valorTienen FROM DUAL;
    :NEW.OID_T:= valorTienen;
END;
/

```

```

-----
CREATE OR REPLACE TRIGGER TR_SEC_SeEnvianA
BEFORE INSERT ON SeEnvianA
FOR EACH ROW
DECLARE
    valorSeEnvianA NUMBER := 0;
BEGIN
    SELECT SEC_SeEnvianA.NEXTVAL INTO valorSeEnvianA FROM DUAL;
    :NEW.OID_SEA:= valorSeEnvianA;
END;
/

```

```

-----
CREATE OR REPLACE TRIGGER TR_SEC_Presentan
BEFORE INSERT ON Presentan
FOR EACH ROW
DECLARE
    valorPresentan NUMBER := 0;
BEGIN
    SELECT SEC_Presentan.NEXTVAL INTO valorPresentan FROM DUAL;
    :NEW.OID_P:= valorPresentan;
END;
/

```

```

-----
CREATE OR REPLACE TRIGGER TR_SEC_LineasOferta
BEFORE INSERT ON LineasOferta
FOR EACH ROW
DECLARE
    valorLineasOferta NUMBER := 0;
BEGIN
    SELECT SEC_LineasOferta.NEXTVAL INTO valorLineasOferta FROM DUAL;
    :NEW.OID_LO:= valorLineasOferta;
END;
/

```

```

-----
CREATE OR REPLACE TRIGGER TR_SEC_IDREFERENCIA_ENCARGO
BEFORE INSERT ON Encargos
FOR EACH ROW
DECLARE
    valorIdReferencia NUMBER := 0;
BEGIN
    SELECT SEC_IDREFERENCIA_ENCARGO.NEXTVAL INTO valorIdReferencia FROM
DUAL;
    :NEW.idReferenciaEncargo:= valorIdReferencia;
END;
/

```

```

-----
CREATE OR REPLACE TRIGGER TRIGGER_SEC_AVISOS
BEFORE INSERT ON AVISOS
FOR EACH ROW
DECLARE
    secuenciaAvisos NUMBER := 0;
BEGIN
    SELECT SEC_AVISOS.NEXTVAL INTO secuenciaAvisos FROM DUAL;
    :NEW.OID_AV := secuenciaAvisos;
END;
/

```

```

CREATE OR REPLACE TRIGGER TRIGGER_SEC_IDREFPROD
BEFORE INSERT ON Productos
FOR EACH ROW
DECLARE
    secuenciaProductos NUMBER := 0;
BEGIN
    SELECT SEC_IDREFERENCIAPRODUCTO.NEXTVAL INTO secuenciaProductos FROM
DUAL;
    :NEW.idReferenciaProducto := secuenciaProductos;
END;
/

```

10.SCRIPTS - CREACION DE LOS PROCEDIMIENTOS Y FUNCIONES

-- Creación de los procedimientos y funciones

--Registrar una franquicia

```
CREATE OR REPLACE PROCEDURE PR_registrar_franquicia (  
    w_nombreFranquicia IN Franquicias.nombreFranquicia%TYPE,  
    w_direccion IN Franquicias.direccion%TYPE,  
    w_propietario IN Franquicias.propietario%TYPE  
) IS  
    BEGIN  
        INSERT INTO Franquicias (nombreFranquicia, direccion, propietario)  
        VALUES (w_nombreFranquicia, w_direccion, w_propietario);  
        COMMIT;  
    END PR_registrar_franquicia;  
/
```

--Borrar una franquicia

```
CREATE OR REPLACE PROCEDURE PR_eliminar_franquicia (  
    w_nombreFranquicia IN Franquicias.nombreFranquicia%TYPE )  
IS  
    BEGIN  
  
        DELETE FROM Tienen WHERE nombreFranquicia = w_nombreFranquicia;  
        DELETE FROM Franquicias WHERE nombreFranquicia = w_nombreFranquicia;  
        COMMIT;  
    END PR_eliminar_franquicia;  
/
```

--Actualizar una franquicia

*--Registrar un cliente y lo agrega a una franquicia (El modelo presentan una relacion 1. *)*

```
CREATE OR REPLACE PROCEDURE PR_registrar_cliente (  
  w_nombreFranquicia IN Franquicias.nombreFranquicia%TYPE,  
  w_codigoPostal IN Clientes.codigoPostal%TYPE,  
  w_nombre IN Clientes.nombre%TYPE,  
  w_apellido IN Clientes.apellido%TYPE,  
  w_fechaNacimiento IN Clientes.fechaNacimiento%TYPE,  
  w_dni IN Clientes.dni%TYPE,  
  w_profesion IN Clientes.profesion%TYPE,  
  w_telefono IN Clientes.telefono%TYPE,  
  w_email IN Clientes.email%TYPE,  
  w_razonVenida IN Clientes.razonVenida%TYPE,  
  w_direccion IN Clientes.direccion%TYPE,  
  w_sexo IN Clientes.sexo%TYPE,  
  w_fotoURL IN Clientes.fotoURL%TYPE,  
  w_firmaURL IN Clientes.firmaURL%TYPE  
) IS  
  BEGIN  
    INSERT INTO Clientes (codigoPostal, nombre, apellido, fechaNacimiento, dni, profesion,  
telefono, email, razonVenida, direccion, sexo,fotoURL,firmaURL)  
      VALUES (w_codigoPostal, w_nombre, w_apellido, w_fechaNacimiento, w_dni, w_profesion,  
w_telefono, w_email, w_razonVenida, w_direccion, w_sexo,w_fotoURL,w_firmaURL);  
    COMMIT;  
  
    INSERT INTO Tienen(nombreFranquicia, OID_C)  
      VALUES(w_nombreFranquicia, SEC_CLIENTE.CURRVAL);  
    COMMIT;  
  
  END PR_registrar_cliente;  
/  

```

--Registra a un cliente en una franquicia (Un cliente puede estar en mas de una franquicia)

```
CREATE OR REPLACE PROCEDURE PR_agre_Cliente_Franquicia (  
  w_nombreFranquicia IN Franquicias.nombreFranquicia%TYPE,  
  w_OID_C IN Clientes.OID_C%TYPE  
) IS  
  BEGIN  
    INSERT INTO Tienen(nombreFranquicia, OID_C)  
      VALUES(w_nombreFranquicia, w_OID_C );  
    COMMIT;  
  END PR_agre_Cliente_Franquicia;  
/  

```

--Borra a un cliente de una franquicia

```
CREATE OR REPLACE PROCEDURE PR_elim_cliente_Franquicia (  
  w_OID_C IN Tienen.OID_C%TYPE,  
  w_nombreFranquicia IN Tienen.nombreFranquicia%TYPE  
)  
IS  
  BEGIN  
    DELETE FROM Tienen WHERE OID_C = w_OID_C AND nombreFranquicia =  
w_nombreFranquicia;  
    COMMIT;  
  END PR_elim_cliente_Franquicia;  
/
```

--Borrar un cliente

```
CREATE OR REPLACE PROCEDURE PR_eliminar_cliente (  
  w_OID_C IN Clientes.OID_C%TYPE )  
IS  
  BEGIN  
    DELETE FROM Clientes WHERE OID_C = w_OID_C;  
    COMMIT;  
  END PR_eliminar_cliente;  
/
```

--Realizar un encargo (Crea el encargo e introduce la primera linea)

```
CREATE OR REPLACE PROCEDURE PR_realizar_encargo(  
  w_OID_C IN Clientes.OID_C%TYPE,  
  w_ID_referenciaProducto IN Productos.idReferenciaProducto%TYPE,  
  w_cantidad SMALLINT  
) IS  
  BEGIN  
    INSERT INTO Encargos(OID_C,fechaPedido)  
    VALUES(w_OID_C,SYSDATE);  
    COMMIT;  
  
    INSERT INTO  
LineasEncargo(idReferenciaProducto,idReferenciaEncargo,cantidad,precioUnitario)  
  
    VALUES(w_ID_referenciaProducto,SEC_IDREFERENCIA_ENCARGO.CURRVAL,w_cantidad,  
(SELECT precioIVA FROM Productos WHERE idReferenciaProducto =  
w_ID_referenciaProducto));  
    COMMIT;  
  
    UPDATE PRODUCTOS SET CANTIDAD = CANTIDAD - w_cantidad where  
idReferenciaProducto = w_ID_referenciaProducto;  
    COMMIT;  
  END PR_realizar_encargo;  
/
```

--Borrar un encargo

```
CREATE OR REPLACE PROCEDURE PR_cancelar_encargo (  
  w_idReferenciaEncargo IN Encargos.idReferenciaEncargo%TYPE )  
IS  
  checker DATE;  
  x_1 NUMBER;  
  cantidadN NUMBER;  
  BEGIN  
    SELECT fechaEntrega INTO checker FROM Encargos WHERE idReferenciaEncargo =  
w_idReferenciaEncargo;  
    IF(checker is null)THEN  
      SELECT idReferenciaProducto,cantidad INTO x_1,cantidadN FROM LineasEncargo  
WHERE idReferenciaEncargo = w_idReferenciaEncargo;  
      UPDATE Productos SET cantidad = cantidad + cantidadN WHERE idReferenciaProducto  
= x_1;  
      DELETE FROM Encargos WHERE idReferenciaEncargo = w_idReferenciaEncargo;  
      COMMIT;  
    END IF;  
    EXCEPTION WHEN OTHERS THEN  
      DBMS_OUTPUT.put_line('No puedes cancelar un encargo ya entregado');  
      ROLLBACK;  
END PR_cancelar_encargo;  
/
```

```
CREATE OR REPLACE PROCEDURE PR_encargoRecibido (  
  w_idReferenciaEncargo IN Encargos.idReferenciaEncargo%TYPE ) IS  
  entrega DATE;  
  checker DATE;  
  BEGIN  
    SELECT fechaEntrega INTO checker FROM Encargos WHERE idReferenciaEncargo =  
w_idReferenciaEncargo;  
    IF(checker IS NULL) THEN  
      SELECT SYSDATE INTO entrega FROM DUAL;  
      UPDATE Encargos SET FECHAENTREGA = entrega WHERE IDREFERENCIAENCARGO  
= w_idReferenciaEncargo;  
      COMMIT;  
    END IF;  
  
END PR_encargoRecibido;  
/
```

--Registrar un producto

```
CREATE OR REPLACE PROCEDURE PR_registrar_producto (  
  w_nombre IN Productos.nombre%TYPE,  
  w_cantidad IN Productos.cantidad%TYPE,  
  w_precio IN Productos.precio%TYPE,  
  w_iva IN Productos.iva%TYPE)  
IS  
  BEGIN  
    INSERT INTO Productos(nombre,cantidad,precio,iva)  
VALUES(w_nombre,w_cantidad,w_precio,w_iva);  
    COMMIT;  
END PR_registrar_producto;  
/
```

--Registrar una lentilla

```
CREATE OR REPLACE PROCEDURE PR_registrar_lentilla (  
  w_nombre IN Productos.nombre%TYPE,  
  w_cantidad IN Productos.cantidad%TYPE,  
  w_precio IN Productos.precio%TYPE,  
  w_iva IN Productos.iva%TYPE,  
  w_tamaño IN Lentillas.tamaño%TYPE)  
IS  
  ProductoID SMALLINT;  
  BEGIN  
    INSERT INTO Productos(nombre,cantidad,precio,descuento,iva)  
VALUES(w_nombre,w_cantidad,w_precio,'0,0',w_iva);  
    COMMIT;  
    SELECT SEC_IDREFERENCIAPRODUCTO.CURRVAL INTO ProductoID FROM DUAL;  
    INSERT INTO LENTILLAS VALUES(productoID,w_tamaño);  
    COMMIT;  
END PR_registrar_lentilla;  
/
```

--Registrar una montura

```
CREATE OR REPLACE PROCEDURE PR_registrar_Montura (  
  w_nombre IN Productos.nombre%TYPE,  
  w_cantidad IN Productos.cantidad%TYPE,  
  w_precio IN Productos.precio%TYPE,  
  w_iva IN Productos.iva%TYPE,  
  w_familiaMontura IN Monturas.familiaMontura%TYPE,  
  w_sexo IN Monturas.sexo%TYPE)  
IS  
  ProductoID SMALLINT;  
  BEGIN  
    INSERT INTO Productos(nombre,cantidad,precio,descuento,iva)  
VALUES(w_nombre,w_cantidad,w_precio,'0,0',w_iva);  
    COMMIT;  
    SELECT SEC_IDREFERENCIAPRODUCTO.CURRVAL INTO ProductoID FROM DUAL;  
    INSERT INTO Monturas VALUES(productoID,w_familiaMontura,w_sexo);  
    COMMIT;  
END PR_registrar_Montura;  
/  
-----
```

--Registrar una Lente

```
CREATE OR REPLACE PROCEDURE PR_registrar_Lente (  
  w_nombre IN Productos.nombre%TYPE,  
  w_cantidad IN Productos.cantidad%TYPE,  
  w_precio IN Productos.precio%TYPE,  
  w_iva IN Productos.iva%TYPE,  
  w_curvatura IN Lentes.curvatura%TYPE)  
IS  
  ProductoID SMALLINT;  
  BEGIN  
    INSERT INTO Productos(nombre,cantidad,precio,descuento,iva)  
VALUES(w_nombre,w_cantidad,w_precio,'0,0',w_iva);  
    COMMIT;  
    SELECT SEC_IDREFERENCIAPRODUCTO.CURRVAL INTO ProductoID FROM DUAL;  
    INSERT INTO Lentes VALUES(productoID,w_curvatura);  
    COMMIT;  
END PR_registrar_Lente;  
/  
-----
```

--Borrar un producto

```
CREATE OR REPLACE PROCEDURE PR_eliminar_producto (  
  w_idreferenciaProducto IN Productos.idreferenciaProducto%TYPE )  
IS  
  BEGIN  
    DELETE FROM Productos WHERE idreferenciaProducto = w_idreferenciaProducto;  
    COMMIT;  
END PR_eliminar_producto;  
/  
-----
```

--Añadir un producto a un pedido (Se crea una linea de encargo)

```
CREATE OR REPLACE PROCEDURE PR_añadirProducto_encargo(  
w_idreferenciaProducto IN Productos.idReferenciaProducto%TYPE,  
w_idReferenciaEncargo IN Encargos.idReferenciaEncargo%TYPE,  
w_cantidad SMALLINT  
) IS  
  BEGIN  
    INSERT INTO  
LineasEncargo(idreferenciaProducto,idReferenciaEncargo,cantidad,precioUnitario)  
      VALUES(w_idreferenciaProducto,w_idReferenciaEncargo,w_cantidad,(SELECT precioIVA  
FROM Productos WHERE idReferenciaProducto = w_idReferenciaProducto));  
    COMMIT;  
    UPDATE PRODUCTOS SET CANTIDAD = CANTIDAD - w_cantidad where  
idReferenciaProducto = w_idreferenciaProducto;  
    COMMIT;  
  END PR_añadirProducto_encargo;  
/
```

--Eliminar un producto de un pedido (Se elimina una linea de encargo)

```
CREATE OR REPLACE PROCEDURE PR_eliminarProducto_encargo (  
w_OID_LE IN LineasEncargo.OID_LE%TYPE )  
IS  
  BEGIN  
UPDATE Productos SET Cantidad = (SELECT cantidad FROM LineasEncargo WHERE  
OID_LE = w_OID_LE) WHERE idReferenciaProducto = (SELECT idReferenciaProducto  
FROM LineasEncargo WHERE OID_LE = w_OID_LE);  
DELETE FROM LineasEncargo WHERE OID_LE = w_OID_LE;  
  
COMMIT;  
END PR_eliminarProducto_encargo;  
/
```

--Crea la graduación de un cliente

```
CREATE OR REPLACE PROCEDURE PR_graduar_ClienteReducido(  
w_OID_C IN Clientes.OID_C%TYPE,  
w_graduadoPor IN Graduaciones.graduadoPor%TYPE,  
w_atendidoPor IN Graduaciones.atendidoPor%TYPE,  
w_ejeI IN Ojos.eje%TYPE,  
w_lejosI IN Ojos.lejos%TYPE,  
w_cercaI IN Ojos. cerca%TYPE,  
w_ejeD IN Ojos.eje%TYPE,  
w_lejosD IN Ojos.lejos%TYPE,  
w_cercaD IN Ojos. cerca%TYPE  
) IS  
  BEGIN  
    INSERT INTO Graduaciones(OID_C,graduadoPor,atendidoPor,fecha)  
    VALUES(w_OID_C,w_graduadoPor,w_atendidoPor,SYSDATE);  
    COMMIT;  
  
    INSERT INTO Ojos(OID_G,eje,cerca,lejos)  
VALUES(SEC_GRADUACION.CURRVAL,w_ejeI,w_cercaI,w_lejosI);  
    INSERT INTO Ojos(OID_G,eje,cerca,lejos)  
VALUES(SEC_GRADUACION.CURRVAL,w_ejeD,w_cercaD,w_lejosD);  
    COMMIT;  
  END PR_graduar_ClienteReducido;  
/  
-----
```

--Elimina la graduación de un cliente

```
CREATE OR REPLACE PROCEDURE PR_elim_grad_Clie (  
w_OID_G IN Graduaciones.OID_G%TYPE  
) IS  
  BEGIN  
    DELETE FROM Graduaciones WHERE OID_G = w_OID_G;  
    COMMIT;  
  END PR_elim_grad_Clie;  
/  
-----
```

--Presenta una oferta (Crea la oferta e introduce la primera linea)

```
CREATE OR REPLACE PROCEDURE PR_presentar_Oferta (  
w_nombreOferta IN Ofertas.nombreOferta%TYPE,  
w_idReferenciaProducto IN Productos.idReferenciaProducto%TYPE,  
w_descuentoOferta IN LineasOferta.descuentoOferta%TYPE,  
w_descripcion IN LineasOferta.descripcion%TYPE,  
w_nombreFranquicia VARCHAR  
) IS  
  BEGIN  
    INSERT INTO Ofertas(nombreOferta)  
      VALUES (w_nombreOferta);  
    COMMIT;  
    INSERT INTO PRESENTAN(NombreOferta, nombreFranquicia)  
VALUES(w_nombreOferta, w_nombreFranquicia);  
    COMMIT;  
    INSERT INTO LineasOferta(nombreOferta,idReferenciaProducto, descuentoOferta,  
descripcion)  
      VALUES (w_nombreOferta,w_idReferenciaProducto,w_descuentoOferta, w_descripcion);  
    COMMIT;  
  
  END PR_presentar_Oferta;  
/  
-----
```

--Añade productos (Añade lineas de oferta)

```
CREATE OR REPLACE PROCEDURE PR_añadirProducto_Oferta(  
w_nombreOferta IN Ofertas.nombreOferta%TYPE,  
w_idReferenciaProducto IN Productos.idReferenciaProducto%TYPE,  
w_descuentoOferta IN LineasOferta.descuentoOferta%TYPE,  
w_descripcion IN LineasOferta.descripcion%TYPE  
) IS  
  BEGIN  
    INSERT INTO LineasOferta(nombreOferta,idReferenciaProducto, descuentoOferta,  
descripcion)  
      VALUES (w_nombreOferta,w_idReferenciaProducto,w_descuentoOferta, w_descripcion);  
    COMMIT;  
  END PR_añadirProducto_Oferta;  
/  
-----
```

--Elimina producto de una oferta (Elimina una linea de oferta)

```
CREATE OR REPLACE PROCEDURE PR_eliminarProducto_Oferta (  
  w_OID_LO IN LineasOferta.OID_LO%TYPE )  
IS  
  codP SMALLINT;  
  BEGIN  
    SELECT idReferenciaProducto INTO codP FROM LineasOferta WHERE OID_LO =  
w_OID_LO;  
    UPDATE PRODUCTOS SET descuento = '0,0' WHERE idReferenciaProducto = codP;  
    DELETE FROM LineasOferta WHERE OID_LO = w_OID_LO;  
    COMMIT;  
  END PR_eliminarProducto_Oferta;  
/  
-----
```

```
CREATE OR REPLACE PROCEDURE PR_eliminar_Oferta (  
  w_nombreOferta IN Ofertas.nombreOferta%TYPE )  
IS  
  CURSOR c1 IS SELECT idReferenciaProducto FROM LineasOferta WHERE nombreOferta =  
w_nombreOferta;  
  id_ SMALLINT;  
  BEGIN  
    FOR i_ IN c1 LOOP  
      UPDATE PRODUCTOS SET descuento = '0,0' WHERE idReferenciaProducto =  
i_.idReferenciaProducto;  
      DELETE FROM LineasOferta WHERE nombreOferta = w_nombreOferta;  
      DELETE FROM SeEnvianA WHERE nombreOferta = w_nombreOferta;  
    END LOOP;  
  
    DELETE FROM Ofertas WHERE nombreOferta = w_nombreOferta;  
  
END PR_eliminar_Oferta;  
/  
-----
```

--Envia una oferta a un cliente

```
CREATE OR REPLACE PROCEDURE PR_enviar_Oferta (  
  w_nombreOferta IN Ofertas.nombreOferta%TYPE,  
  w_OID_C IN Clientes.OID_C%TYPE  
) IS  
  BEGIN  
    INSERT INTO SeEnvianA(nombreOferta, OID_C)  
      VALUES(w_nombreOferta,w_OID_C);  
    COMMIT;  
  END PR_enviar_Oferta;  
/  
-----
```

--Elimina el envio de una oferta a un cliente

```
CREATE OR REPLACE PROCEDURE PR_elim_env_Oferta (  
w_OID_SEA IN SeEnvianA.OID_SEA%TYPE  
) IS  
  BEGIN  
    DELETE FROM SeEnvianA WHERE OID_SEA = w_OID_SEA;  
    COMMIT;  
  END PR_elim_env_Oferta;  
/
```

--Nuevos Clientes (Dado un periodo de tiempo, calcula cuantos clientes se registraron)

```
CREATE OR REPLACE FUNCTION FN_nuevos_Clientes (  
fechaInicio DATE, fechaFin DATE)  
RETURN NUMBER IS n_clientes SMALLINT;  
  BEGIN  
    SELECT COUNT(*) INTO n_clientes FROM Clientes WHERE fechaRegistro BETWEEN  
fechaInicio AND fechaFin;  
    RETURN n_clientes;  
END;  
/
```

--Producto mas caro (Sin IVA)

```
CREATE OR REPLACE FUNCTION FN_prod_masCaro  
RETURN NUMBER  
IS  
  mayor_precio NUMBER(12,2);  
  id_prod_mc SMALLINT;  
  BEGIN  
    SELECT MAX(precio) INTO mayor_precio FROM Productos;  
    SELECT idReferenciaProducto INTO id_prod_mc FROM Productos WHERE precio =  
mayor_precio;  
    RETURN id_prod_mc;  
END;  
/
```

--Producto mas barato (Sin IVA)

```
CREATE OR REPLACE FUNCTION FN_prod_masBarato  
RETURN NUMBER  
IS  
  menor_precio NUMBER(12,2);  
  id_prod_mnc SMALLINT;  
  BEGIN  
    SELECT MIN(precio) INTO menor_precio FROM Productos;  
    SELECT idReferenciaProducto INTO id_prod_mnc FROM Productos WHERE precio =  
menor_precio;  
    RETURN id_prod_mnc;
```

END;

/

--Oferta mas enviada

CREATE OR REPLACE FUNCTION FN_ofer_masEnv

RETURN VARCHAR IS o_me **VARCHAR**(50);

BEGIN

SELECT cnt1.nombreOferta **INTO** o_me

FROM (**SELECT COUNT**(*) **AS** total, nombreOferta

FROM SeEnvianA

GROUP BY nombreOferta) cnt1,

(**SELECT MAX**(total) **as** maxtotal

FROM (**SELECT COUNT**(*) **AS** total, nombreOferta **FROM** SeEnvianA **GROUP BY**

nombreOferta)) cnt2

WHERE cnt1.total = cnt2.maxtotal;

RETURN o_me;

END;

/

--Oferta menos enviada

CREATE OR REPLACE FUNCTION FN_ofer_menosEnv

RETURN VARCHAR IS o_mne **VARCHAR**(50);

BEGIN

SELECT cnt1.nombreOferta **INTO** o_mne

FROM (**SELECT COUNT**(*) **AS** total, nombreOferta

FROM SeEnvianA

GROUP BY nombreOferta) cnt1,

(**SELECT MIN**(total) **as** mintotal

FROM (**SELECT COUNT**(*) **AS** total, nombreOferta **FROM** SeEnvianA **GROUP BY**

nombreOferta)) cnt2

WHERE cnt1.total = cnt2.mintotal;

RETURN o_mne;

END;

/

11.SCRIPITS - TRIGGER NO ASOCIADOS A SECUENCIAS

-- Creación de los trigger NO asociados a secuencias

--Comprobar que graduacion no se realiza en el futuro

```
CREATE OR REPLACE TRIGGER TR_Graduaciones_FechaFutura  
BEFORE INSERT OR UPDATE ON Graduaciones  
FOR EACH ROW  
DECLARE  
BEGIN
```

```
    IF(:NEW.fecha> SYSDATE)
```

```
        THEN raise_application_error
```

```
            (-20050,'La fecha de graduacion no puede haber ocurrido antes que la fecha del dia');
```

```
    END IF;
```

```
END;
```

```
/
```

--Comprobar que la fecha de nacimiento no sea posterior al presente

```
CREATE OR REPLACE TRIGGER TR_Cliente_FechaActual  
BEFORE INSERT OR UPDATE ON Clientes  
FOR EACH ROW  
DECLARE  
    futuro NUMBER;  
BEGIN
```

```
    futuro := (SYSDATE - :NEW.fechaNacimiento) ;
```

```
    IF(futuro < 0)
```

```
        THEN raise_application_error
```

```
            (-20010,'Un cliente no puede haber nacido en el futuro');
```

```
    END IF;
```

```
END;
```

```
/
```

--Poner fecha de registro cuando se crea un client

```
CREATE OR REPLACE TRIGGER TR_Cliente_FechaActual  
BEFORE INSERT OR UPDATE ON Clientes  
FOR EACH ROW  
BEGIN  
    :NEW.fechaRegistro := SYSDATE;  
END;
```

/

--Comprobar que la fecha de entrega no sea anterior a la fecha de pedida

```
CREATE OR REPLACE TRIGGER TR_Encargos_FechaEntrega
BEFORE INSERT OR UPDATE ON Encargos
FOR EACH ROW
DECLARE
    entrega NUMBER;
BEGIN
    entrega := (:NEW.fechaEntrega - :NEW.fechaPedido ) ;
    IF(entrega < 0)
        THEN raise_application_error
            (-20020,'La fecha de entrega no puede haber ocurrido antes que la fecha de pedido');
        END IF;
END;
/
```

--RN-003 Avisos de stock -> Comprobar si el stock es bajo, para modificar la tabla de aviso

```
CREATE OR REPLACE TRIGGER TR_STOCK_PRODUCTO
AFTER INSERT OR UPDATE ON PRODUCTOS
FOR EACH ROW
BEGIN
    IF(:NEW.CANTIDAD < 5)
        THEN
            INSERT INTO AVISOS(productoerr,descripcion,cantidadActual)
            VALUES(:NEW.idReferenciaProducto, 'Stock bajo, menos de 5
unidades',:NEW.CANTIDAD);
        ELSE
            DELETE FROM AVISOS WHERE productoerr = :NEW.idReferenciaProducto;
        END IF;
END;
/
```

--Calculo del precio con el IVA aplicado y el descuento

```
CREATE OR REPLACE TRIGGER TR_precioIVA
BEFORE INSERT OR UPDATE ON PRODUCTOS
FOR EACH ROW
BEGIN
    :NEW.precioIva := :NEW.precio + (:NEW.precio * :NEW.iva) - (:NEW.precio
* :NEW.descuento);
END;
/
```

--Modificar el campo descuento del precio

```
CREATE OR REPLACE TRIGGER TR_nuevo_desc
AFTER INSERT OR UPDATE ON LineasOferta
FOR EACH ROW
BEGIN
    UPDATE Productos SET descuento = :NEW.descuentoOferta WHERE idReferenciaProducto =
:NEW.idReferenciaProducto;
```

END;
/

```

-----
-- Evitar que un cliente no esté en ninguna franquicia
CREATE OR REPLACE TRIGGER TR_sin_franquicia
BEFORE INSERT OR UPDATE ON Tienen
FOR EACH ROW
DECLARE
numeroFCliente SMALLINT;
BEGIN
    SELECT COUNT(*) INTO numeroFCliente FROM Tienen WHERE OID_C = :OLD.OID_C;
    IF(numeroFCliente = 1)
    THEN raise_application_error
        (-20030, 'No se puede eliminar a un cliente de su ultima franquicia, debe eliminar el cliente');
    END IF;

END;
/

```

```

-----
CREATE OR REPLACE TRIGGER TR_prod_yaoferta
BEFORE INSERT OR UPDATE ON LineasOferta
FOR EACH ROW
DECLARE
n_ SMALLINT;
BEGIN
    SELECT COUNT(*) INTO n_ FROM LineasOferta WHERE idReferenciaProducto
= :new.idReferenciaProducto;
    IF(n_ >= 1)
    THEN
        UPDATE LineasOferta SET NombreOferta = :NEW.nombreOferta WHERE OID_LO
= :NEW.OID_LO;
        UPDATE LineasOferta SET descripcion = :NEW.descripcion WHERE OID_LO
= :NEW.OID_LO;
    END IF;
END;
/

```

```

-----
CREATE OR REPLACE TRIGGER TR_ProductoYaEnLE
BEFORE INSERT ON LineasEncargo
FOR EACH ROW
DECLARE
n_ SMALLINT;
cant SMALLINT;
BEGIN
    SELECT COUNT(*) INTO n_ FROM LineasEncargo WHERE idReferenciaProducto
= :new.idReferenciaProducto;
    IF(n_ = 1)
    THEN
        SELECT cantidad INTO cant FROM LineasEncargo WHERE idReferenciaProducto
= :new.idReferenciaProducto;
        DELETE FROM LineasEncargo Where idReferenciaProducto = :new.idReferenciaProducto;
        :new.cantidad := :new.cantidad + cant;

    END IF;
END;
/

```

11.SCRIPTS - TRIGGER NO ASOCIADOS A SECUENCIAS

DELETE FROM ENCARGOS;
DELETE FROM PRODUCTOS;
DELETE FROM CLIENTES;
DELETE FROM CODIGOSPOSTALES;
DELETE FROM OFERTAS;
DELETE FROM FRANQUICIAS;

INSERT INTO CODIGOSPOSTALES **VALUES**(41110,'Sevilla','Bollullos');

EXECUTE PR_registrar_franquicia ('Optica Maguilla','C/Cristo de los gitanos,12','Jose Antonio');

EXECUTE PR_registrar_cliente ('Optica Maguilla',41110,'Antonio','Osborne',TO_DATE('2012-06-05','YYYY-MM-DD'),'27283847E','Cocinero',777183927,'sss@sss.ss','Me he quedado tuerto de ir por la autopista a 180','C/Gustavo Adolfo Becquer, 32','Otro','https://asdasd.asda/asdasd.html','https://asdasd.asda/asdasd.html');

EXECUTE PR_registrar_cliente ('Optica Maguilla',41110,'Alex','Osborne',TO_DATE('2012-06-05','YYYY-MM-DD'),'27283842F','Misterioso',777483927,'sss@sss.ss','Me he quedado tuerto de ir por la autopista a 180','C/Gustavo Adolfo Becquer, 32','Otro','https://asdasd.asda/asdasd.html','https://asdasd.asda/asdasd.html');

EXECUTE PR_registrar_cliente ('Optica Maguilla',41110,'Jose','Osborne',TO_DATE('2012-06-05','YYYY-MM-DD'),'27583842X','Peluquero',677483927,'sss@sss.ss','Me he quedado tuerto de ir por la autopista a 180','C/Gustavo Adolfo Becquer, 32','Otro','https://asdasd.asda/asdasd.html','https://asdasd.asda/asdasd.html');

EXECUTE PR_registrar_cliente ('Optica Maguilla',41110,'Fernado','Osborne',TO_DATE('2012-06-05','YYYY-MM-DD'),'27283647B','Profesor de iNgles',777883927,'sss@sss.ss','Me he quedado tuerto de ir por la autopista a 180','C/Gustavo Adolfo Becquer, 32','Otro','https://asdasd.asda/asdasd.html','https://asdasd.asda/asdasd.html');

EXECUTE PR_registrar_cliente ('Optica Maguilla',41110,'Victor','Osborne',TO_DATE('2012-06-05','YYYY-MM-DD'),'27283817P','Nini',777083927,'sss@sss.ss','Me he quedado tuerto de ir por la autopista a 180','C/Gustavo Adolfo Becquer, 32','Otro','https://asdasd.asda/asdasd.html','https://asdasd.asda/asdasd.html');

```
EXECUTE PR_registrar_lentilla('lentilla a',420,'5,7','0,2','0,3');
EXECUTE PR_registrar_montura('montura a',420,'5,7','0,2','Metalica','Hombre');
EXECUTE PR_registrar_lente('lente a',420,'5,7','0,2','12,5');
```

```
EXECUTE PR_registrar_producto ('a',2,'5,7','0,2');
EXECUTE PR_registrar_producto ('b',6,'5,7','0,2');
EXECUTE PR_registrar_producto ('c',6,'2,1','0,2');
EXECUTE PR_registrar_producto ('d',6,'5,7','0,2');
EXECUTE PR_registrar_producto ('e',6,'7,7','0,2');
EXECUTE PR_registrar_producto ('f',6,'5,7','0,2');
EXECUTE PR_registrar_producto ('g',420,'5,7','0,2');
```

--Presentar oferta / Eliminar oferta

```
EXECUTE PR_presentar_Oferta('Verano en
Sevilla',SEC_IDREFERENCIAPRODUCTO.CURRVAL,'0,2','Gafas fabricadas en Sevilla
rebajadas','Optica Maguilla');
```

```
EXECUTE PR_presentar_Oferta('Vision 2020',SEC_IDREFERENCIAPRODUCTO.CURRVAL -
1,'0,4','Gafas nuevas en oferta','Optica Maguilla');
```

```
EXECUTE PR_presentar_Oferta('Rebajas
Navideñas',SEC_IDREFERENCIAPRODUCTO.CURRVAL - 2,'0,5','Productos con variados
descuentos','Optica Maguilla');
```

```
EXECUTE PR_eliminar_Oferta('Rebajas Navideñas');
```

--Añadir producto oferta / Eliminar producto oferta

```
EXECUTE PR_añadirProducto_Oferta('Verano en
Sevilla',SEC_IDREFERENCIAPRODUCTO.CURRVAL-3,'0,4','Gafas fabricadas en Sevilla
rebajadas');
```

```
EXECUTE PR_añadirProducto_Oferta('Vision
2020',SEC_IDREFERENCIAPRODUCTO.CURRVAL - 4,'0,3','Gafas nuevas en oferta');
```

--Error: Intentar añadir a una oferta un producto que ya está en otra

```
EXECUTE PR_añadirProducto_Oferta('Vision
2020',SEC_IDREFERENCIAPRODUCTO.CURRVAL - 3,'0,3','Gafas nuevas en oferta cambiado');
```

```
EXECUTE PR_eliminarProducto_Oferta(SEC_LINEASOFERTA.CURRVAL - 1);
```

EXECUTE

```
PR_realizar_encargo(SEC_CLIENTE.CURRVAL,SEC_IDREFERENCIAPRODUCTO.CURRVAL
,417);
```

EXECUTE

```
PR_añadirProducto_encargo(SEC_IDREFERENCIAPRODUCTO.CURRVAL ,SEC_IDREFEREN  
CIA_ENCARGO.CURRVAL,100);
```

```
EXECUTE PR_EncargoRecibido(SEC_IDREFERENCIA_ENCARGO.CURRVAL);  
EXECUTE PR_Cancelar_Encargo(SEC_IDREFERENCIA_ENCARGO.CURRVAL);
```

--Enviar oferta / Eliminar envio oferta

```
EXECUTE PR_enviar_Oferta('Verano en Sevilla',SEC_CLIENTE.CURRVAL);  
EXECUTE PR_enviar_Oferta('Verano en Sevilla',SEC_CLIENTE.CURRVAL - 1);  
EXECUTE PR_enviar_Oferta('Vision 2020',SEC_CLIENTE.CURRVAL-2);
```

```
SET SERVEROUTPUT ON;
```

```
DECLARE
```

```
prodBarato NUMBER;
```

```
prodCaro NUMBER;
```

```
CientesNuevos NUMBER;
```

```
BEGIN
```

```
    SELECT FN_prod_masBarato INTO prodBarato FROM DUAL;
```

```
    Select FN_nuevos_Clientes(TO_DATE('2019-01-01','YYYY-MM-DD'),SYSDATE) INTO  
CientesNuevos From Dual;
```

```
    SELECT FN_prod_masCaro INTO prodCaro FROM DUAL;
```

```
    DBMS_OUTPUT.put_line('El producto mas caro es:      IdReferenciaProducto:' || prodCaro);
```

```
    DBMS_OUTPUT.put_line('El producto mas barato es:    IdReferenciaProducto:' || prodBarato);
```

```
    DBMS_OUTPUT.put_line('Numero de clientes nuevos:  Cantidad:' || CientesNuevos);
```

```
END;
```


11.SCRIPTS - PAQUETES

```
CREATE OR REPLACE FUNCTION ASSERT_EQUALS (salida BOOLEAN, salida_esperada
BOOLEAN)
RETURN VARCHAR2 AS
BEGIN
    IF(salida= salida_esperada) THEN
        RETURN 'EXITO';
    ELSE
        RETURN 'FALLO';
    END IF;
END ASSERT_EQUALS;
/
```

```
CREATE OR REPLACE PACKAGE Pruebas_Avisos AS
PROCEDURE Inicializar;
PROCEDURE Insertar(nombre_prueba VARCHAR2,w_idReferenciaProducto
NUMBER,salidaEsperada BOOLEAN);
PROCEDURE Actualizar_descripcion(nombre_prueba VARCHAR2,w_OID_AV
NUMBER,w_new_descripcion VARCHAR2,salidaEsperada BOOLEAN);
PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_OID_AV NUMBER,salidaEsperada
BOOLEAN);
END Pruebas_Avisos;
/
```

```
CREATE OR REPLACE PACKAGE BODY Pruebas_Avisos AS
PROCEDURE Inicializar AS
BEGIN

    DELETE FROM Productos;
    INSERT INTO PRODUCTOS(nombre,cantidad,precio,iva)VALUES ('gafas pull',
10,'21,2','0,21');
    DELETE FROM Avisos;

END Inicializar;
```

```

PROCEDURE Insertar(nombre_prueba VARCHAR2,w_idReferenciaProducto
NUMBER,salidaEsperada BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;

BEGIN

    UPDATE PRODUCTOS SET CANTIDAD = 2 WHERE idReferenciaProducto =
w_idReferenciaProducto;
    COMMIT;

    --Comprobar que todo se ha introducido OK.
    SELECT COUNT(*) INTO aux
        FROM Avisos WHERE PRODUCTOERR = w_idReferenciaProducto;

    IF(AUX<>1) THEN
        salida:=FALSE;
    END IF;
    COMMIT;

    --Mostrar resultado de la prueba
    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
    EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
        ROLLBACK;
    END Insertar;

PROCEDURE Actualizar_descripcion(nombre_prueba VARCHAR2,w_OID_AV
NUMBER,w_new_descripcion VARCHAR2,salidaEsperada BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
BEGIN

    UPDATE Avisos SET descripcion=w_new_descripcion WHERE OID_AV=w_OID_AV;

    --Comprobar que todo se ha introducido OK.
    SELECT COUNT(*) INTO aux
        FROM Avisos WHERE OID_AV=w_OID_AV AND descripcion = w_new_descripcion;

    IF(AUX<>1) THEN
        salida:=FALSE;
    END IF;
    COMMIT;

```

```

--Mostrar resultado de la prueba
DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
    DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
    ROLLBACK;
END Actualizar_descripcion;

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_OID_AV NUMBER,salidaEsperada
BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
BEGIN

    DELETE FROM Avisos WHERE OID_AV = w_OID_AV;
    COMMIT;

--Comprobar que todo se ha introducido OK.
SELECT COUNT(*) INTO aux
    FROM Avisos WHERE OID_AV=w_OID_AV;

IF(AUX<>0) THEN
    salida:=FALSE;
END IF;
COMMIT;

--Mostrar resultado de la prueba
DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
    DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
    ROLLBACK;
END Eliminar;
END Pruebas_Avisos;
/

```

CREATE OR REPLACE PACKAGE Pruebas_Cliente **AS**

PROCEDURE Inicializar;

PROCEDURE Insertar(nombrePrueba VARCHAR2,w_codigoPostal **SMALLINT** ,w_nombre VARCHAR,w_apellido VARCHAR ,w_fechaNacimiento **DATE** ,w_dni VARCHAR ,w_profesion VARCHAR,w_telefono **SMALLINT**,w_email VARCHAR,w_razonVenida VARCHAR,w_direccion VARCHAR,w_sexo VARCHAR,w_fotoURL VARCHAR,w_firmaURL VARCHAR,salidaEsperada **BOOLEAN**);

PROCEDURE Actualizar_telefono(nombre_prueba VARCHAR2,w_OID_C **NUMBER**,w_new_telefono **NUMBER**,salidaEsperada **BOOLEAN**);

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_OID_C **NUMBER**,salidaEsperada **BOOLEAN**);

END Pruebas_Cliente;

/

CREATE OR REPLACE PACKAGE BODY Pruebas_Cliente **AS**

PROCEDURE Inicializar **AS**

BEGIN

DELETE FROM CLIENTES;

DELETE FROM CODIGOSPOSTALES;

INSERT INTO CODIGOSPOSTALES **VALUES**(41110,'Sevilla','Carrion');

END Inicializar;

PROCEDURE Insertar(nombrePrueba VARCHAR2,w_codigoPostal **SMALLINT** ,w_nombre VARCHAR,w_apellido VARCHAR ,w_fechaNacimiento **DATE** ,w_dni VARCHAR ,w_profesion VARCHAR,w_telefono **SMALLINT**,w_email VARCHAR,w_razonVenida VARCHAR,w_direccion VARCHAR,w_sexo VARCHAR,w_fotoURL VARCHAR,w_firmaURL VARCHAR,salidaEsperada **BOOLEAN**) **AS**

salida **BOOLEAN** :=**TRUE**; --Salida esperada

aux **SMALLINT**;

aux_id **SMALLINT**;

BEGIN

--Insertar codigopostal

INSERT INTO

Clientes(codigoPostal,nombre,apellido ,fechaNacimiento ,dni ,profesion,telefono,email,razonVenida,direccion,sexo,fotoURL,firmaURL)

VALUES

(w_codigoPostal,w_nombre,w_apellido ,w_fechaNacimiento ,w_dni,w_profesion,w_telefono,w_email,w_razonVenida,w_direccion,w_sexo,w_fotoURL,w_firmaURL);

COMMIT;

aux_id := SEC_CLIENTE.CURRVAL;

--Comprobar que todo se ha introducido OK.

SELECT COUNT(*) **INTO** aux

FROM Clientes **WHERE** OID_C=aux_id;--Codigo postal es unico

IF(AUX<>1) **THEN**

```
    salida:=FALSE;  
END IF;  
COMMIT;
```

```

--Mostrar resultado de la prueba
DBMS_OUTPUT.put_line(nombrePrueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
    DBMS_OUTPUT.put_line(nombrePrueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
    ROLLBACK;
END Insertar;

PROCEDURE Actualizar_telefono(nombre_prueba VARCHAR2,w_OID_C
NUMBER,w_new_telefono NUMBER,salidaEsperada BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
BEGIN
    --Actualizar nombres
    UPDATE Clientes SET telefono=w_new_telefono WHERE OID_C=w_OID_C;

    --Comprobar que todo se ha introducido OK.
    SELECT COUNT(*) INTO aux
        FROM Clientes WHERE OID_C=w_OID_C AND telefono=w_new_telefono;--Codigo
postal es unico

    IF(AUX<>1) THEN
        salida:=FALSE;
    END IF;
    COMMIT;

    --Mostrar resultado de la prueba
    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
    EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
        ROLLBACK;
    END Actualizar_telefono;

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_OID_C NUMBER,salidaEsperada
BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
BEGIN
    --Insertar codigopostal
    DELETE FROM Clientes WHERE OID_C=w_OID_C;
    COMMIT;

```

```

--Comprobar que todo se ha introducido OK.
SELECT COUNT(*) INTO aux
FROM Clientes WHERE OID_C=w_OID_C;--Codigo postal es unico

IF(AUX<>0) THEN
    salida:=FALSE;
END IF;
COMMIT;

--Mostrar resultado de la prueba
DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
    DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
    ROLLBACK;
END Eliminar;

END Pruebas_Cliente;
/

-----

CREATE OR REPLACE PACKAGE Pruebas_CodPost AS
PROCEDURE Inicializar;
PROCEDURE Insertar(nombre_prueba VARCHAR2,w_codPostal NUMBER,w_provincia
VARCHAR2,w_localidad VARCHAR2,salidaEsperada BOOLEAN);
PROCEDURE Actualizar_cod(nombre_prueba VARCHAR2,w_codPostal
NUMBER,w_newcodPostal NUMBER,salidaEsperada BOOLEAN);
PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_codPostal NUMBER,salidaEsperada
BOOLEAN);
END Pruebas_CodPost;
/

-----

CREATE OR REPLACE PACKAGE BODY Pruebas_CodPost AS
PROCEDURE Inicializar AS
    BEGIN
        DELETE FROM CLIENTES;
        DELETE FROM CODIGOSPOSTALES;
    END Inicializar;

PROCEDURE Insertar(nombre_prueba VARCHAR2,w_codPostal NUMBER,w_provincia
VARCHAR2,w_localidad VARCHAR2,salidaEsperada BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
    BEGIN
        --Insertar codigopostal
        INSERT INTO CodigosPostales VALUES (w_codPostal,w_provincia,w_localidad);
        COMMIT;

```



```

--Comprobar que todo se ha introducido OK.
SELECT COUNT(*) INTO aux
  FROM CodigosPostales WHERE codPostal=w_codPostal;--Codigo postal es unico

IF(AUX<>1) THEN
  salida:=FALSE;
END IF;
COMMIT;

--Mostrar resultado de la prueba
DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
  DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
  ROLLBACK;
END Insertar;

PROCEDURE Actualizar_cod(nombre_prueba VARCHAR2,w_codPostal
NUMBER,w_newcodPostal NUMBER,salidaEsperada BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
BEGIN
  --Actualizar nombres
  UPDATE CodigosPostales SET codpostal=w_newcodPostal WHERE
codpostal=w_codPostal;

  --Comprobar que todo se ha introducido OK.
  SELECT COUNT(*) INTO aux
    FROM CodigosPostales WHERE codPostal=w_newcodPostal;--Codigo postal es unico

  IF(AUX<>1) THEN
    salida:=FALSE;
  END IF;
  COMMIT;

  --Mostrar resultado de la prueba
  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
  EXCEPTION WHEN OTHERS THEN
    DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
    ROLLBACK;
  END Actualizar_cod;

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_codPostal NUMBER,salidaEsperada
BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
BEGIN

```

--Insertar codigopostal

DELETE FROM CodigosPostales **WHERE** codPostal=w_codPostal;
COMMIT;

--Comprobar que todo se ha introducido OK.

SELECT COUNT(*) INTO aux
 FROM CodigosPostales **WHERE** codPostal=w_codPostal;*--Codigo postal es unico*

IF(AUX<>0) **THEN**

 salida:=**FALSE**;

END IF;

COMMIT;

--Mostrar resultado de la prueba

DBMS_OUTPUT.put_line(nombre_prueba || ':' || **ASSERT_EQUALS**(salida,salidaEsperada));

EXCEPTION WHEN OTHERS THEN

 DBMS_OUTPUT.put_line(nombre_prueba || ':' ||

ASSERT_EQUALS(**FALSE**,salidaEsperada));

ROLLBACK;

END Eliminar;

END Pruebas_CodPost;

/

CREATE OR REPLACE PACKAGE Pruebas_Encargos **AS**

PROCEDURE Inicializar;

PROCEDURE Insertar(nombre_prueba VARCHAR2,w_OID_C **NUMBER**,salidaEsperada
BOOLEAN) ;

PROCEDURE Actualizar_cli(nombre_prueba VARCHAR2,w_idReferenciaEncargo
NUMBER,w_new_OID_C **NUMBER**,salidaEsperada **BOOLEAN**);

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_idReferenciaEncargo
NUMBER,salidaEsperada **BOOLEAN**) ;

END Pruebas_Encargos;

/

CREATE OR REPLACE PACKAGE BODY Pruebas_Encargos **AS**

PROCEDURE Inicializar **AS**

BEGIN

DELETE FROM CLIENTES;

DELETE FROM CODIGOSPOSTALES;

INSERT INTO CODIGOSPOSTALES **VALUES**(41110,'Sevilla','Bollullos');

INSERT INTO

CLIENTES(CODIGOPOSTAL,NOMBRE,APELLIDO,FECHANACIMIENTO,DNI,PROFESION,
TELEFONO,EMAIL,RAZONVENIDA,DIRECCION,SEXO,FOTOURL,FIRMAURL)

VALUES(41110,'An','An',TO_DATE('2012-06-05','YYYY-MM-DD'),
'27283847E','An',777283927,'sss@sss.ss','An','An','Hombre','https://asdasd.asda/
asdasd.html','https://asdasd.asda/asdasd.html');

INSERT INTO

CLIENTES(CODIGOPOSTAL,NOMBRE,APELLIDO,FECHANACIMIENTO,DNI,PROFESION,TELEFONO,EMAIL,RAZONVENIDA,DIRECCION,SEXO,FOTOURL,FIRMAURL)

VALUES(41110,'An','An',TO_DATE('2012-06-05','YYYY-MM-DD'),'27283848Z','An',777283927,'sss@sss.ss','An','An','Hombre','https://asdasd.asda/asdasd.html','https://asdasd.asda/asdasd.html');

DELETE FROM Encargos;

END Inicializar;

PROCEDURE Insertar(nombre_prueba VARCHAR2,w_OID_C **NUMBER**,salidaEsperada **BOOLEAN**) **AS**

salida **BOOLEAN** :=**TRUE**; *--Salida esperada*

aux **SMALLINT**;

aux_id **SMALLINT**;

BEGIN

--Insertar graduaciones

INSERT INTO Encargos(OID_C,fechaPedido)

VALUES(w_OID_C,SYSDATE);

COMMIT;

--Comprobar que todo se ha introducido OK.

aux_id:=SEC_IDREFERENCIA_ENCARGO.CURRVAL;

SELECT COUNT(*) INTO aux

FROM Encargos **WHERE** idReferenciaEncargo= aux_id;

IF(AUX<>1) **THEN**

salida:=**FALSE**;

END IF;

COMMIT;

--Mostrar resultado de la prueba

DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

EXCEPTION WHEN OTHERS THEN

DBMS_OUTPUT.put_line(nombre_prueba || ':' ||

ASSERT_EQUALS(**FALSE**,salidaEsperada));

ROLLBACK;

END Insertar;

PROCEDURE Actualizar_cli(nombre_prueba VARCHAR2,w_idReferenciaEncargo **NUMBER**,w_new_OID_C **NUMBER**,salidaEsperada **BOOLEAN**) **AS**

salida **BOOLEAN** :=**TRUE**; *--Salida esperada*

aux **SMALLINT**;

BEGIN

--Actualizar nombres

UPDATE Encargos **SET** OID_C=w_new_OID_C **WHERE**

idReferenciaEncargo=w_idReferenciaEncargo;

--Comprobar que todo se ha introducido OK.

SELECT COUNT(*) INTO aux
FROM Encargos **WHERE** idReferenciaEncargo=w_idReferenciaEncargo;

IF(AUX<>1) **THEN**
 salida:=**FALSE**;
END IF;
COMMIT;

--Mostrar resultado de la prueba

DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
 DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(**FALSE**,salidaEsperada));
 ROLLBACK;
END Actualizar_cli;

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_idReferenciaEncargo
NUMBER,salidaEsperada **BOOLEAN**) **AS**
salida **BOOLEAN** :=**TRUE**; *--Salida esperada*
aux **SMALLINT**;

BEGIN

--Insertar codigopostal

DELETE FROM Encargos **WHERE** idReferenciaEncargo=w_idReferenciaEncargo;
COMMIT;

--Comprobar que todo se ha introducido OK.

SELECT COUNT(*) INTO aux
FROM Encargos **WHERE** idReferenciaEncargo=w_idReferenciaEncargo;

IF(AUX<>0) **THEN**
 salida:=**FALSE**;
END IF;
COMMIT;

--Mostrar resultado de la prueba

DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
 DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(**FALSE**,salidaEsperada));
 ROLLBACK;
END Eliminar;

END Pruebas_Encargos;

/

```

-----
CREATE OR REPLACE PACKAGE Pruebas_Franquicia AS
PROCEDURE Inicializar;
PROCEDURE Insertar(nombre_prueba VARCHAR2,w_nombreFranquicia
VARCHAR2,w_direccion VARCHAR2,w_propietario VARCHAR2,salidaEsperada BOOLEAN);
PROCEDURE Actualizar_nombre(nombre_prueba VARCHAR2,w_nombreFranquicia
VARCHAR2,w_new_nombreFranquicia VARCHAR2,salidaEsperada BOOLEAN);
PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_nombreFranquicia
VARCHAR2,salidaEsperada BOOLEAN);
END Pruebas_Franquicia;
/

```

```

-----
CREATE OR REPLACE PACKAGE BODY Pruebas_Franquicia AS
PROCEDURE Inicializar AS
    BEGIN

        DELETE FROM Franquicias;

    END Inicializar;

PROCEDURE Insertar(nombre_prueba VARCHAR2,w_nombreFranquicia
VARCHAR2,w_direccion VARCHAR2,w_propietario VARCHAR2,salidaEsperada BOOLEAN)
AS
    salida BOOLEAN :=TRUE; --Salida esperada
    aux SMALLINT;

    BEGIN

        INSERT INTO Franquicias VALUES (w_nombreFranquicia ,w_direccion ,w_propietario);
        COMMIT;

        --Comprobar que todo se ha introducido OK.
        SELECT COUNT(*) INTO aux
            FROM Franquicias WHERE nombreFranquicia= w_nombreFranquicia;

        IF(AUX<>1) THEN
            salida:=FALSE;
        END IF;
        COMMIT;

        --Mostrar resultado de la prueba
        DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
        EXCEPTION WHEN OTHERS THEN
            DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
            ROLLBACK;
        END Insertar;

PROCEDURE Actualizar_nombre(nombre_prueba VARCHAR2,w_nombreFranquicia

```

```

VARCHAR2,w_new_nombreFranquicia VARCHAR2,salidaEsperada BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
BEGIN

```

```

    UPDATE Franquicias SET nombreFranquicia=w_new_nombreFranquicia WHERE
nombreFranquicia=w_nombreFranquicia;

```

```

    --Comprobar que todo se ha introducido OK.

```

```

SELECT COUNT(*) INTO aux
FROM Franquicias WHERE nombreFranquicia=w_new_nombreFranquicia;

```

```

IF(AUX<>1) THEN
    salida:=FALSE;
END IF;
COMMIT;

```

```

    --Mostrar resultado de la prueba

```

```

    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
        ROLLBACK;
END Actualizar_nombre;

```

```

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_nombreFranquicia
VARCHAR2,salidaEsperada BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
BEGIN

```

```

    DELETE FROM Franquicias WHERE nombreFranquicia=w_nombreFranquicia;
COMMIT;

```

```

    --Comprobar que todo se ha introducido OK.

```

```

SELECT COUNT(*) INTO aux
FROM Franquicias WHERE nombreFranquicia=w_nombreFranquicia;

```

```

IF(AUX<>0) THEN
    salida:=FALSE;
END IF;
COMMIT;

```

```

    --Mostrar resultado de la prueba

```

```

    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
        ROLLBACK;
END Eliminar;

```

END Pruebas_Franquicia;

/

CREATE OR REPLACE PACKAGE Pruebas_Graduaciones **AS**

PROCEDURE Inicializar;

PROCEDURE Insertar(nombre_prueba VARCHAR2,w_OID_C **NUMBER**,w_graduadoPor VARCHAR2,w_atendidoPor VARCHAR2,w_fecha **DATE**,salidaEsperada **BOOLEAN**);

PROCEDURE Actualizar_cliente(nombre_prueba VARCHAR2,w_OID_G **NUMBER**,w_new_OID_C **NUMBER**,salidaEsperada **BOOLEAN**);

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_OID_G **NUMBER**,salidaEsperada **BOOLEAN**);

END Pruebas_Graduaciones;

/

CREATE OR REPLACE PACKAGE BODY Pruebas_Graduaciones **AS**

PROCEDURE Inicializar **AS**

BEGIN

DELETE CLIENTES;

DELETE CODIGOSPOSTALES;

INSERT INTO CODIGOSPOSTALES **VALUES**(41110,'Sevilla','Bollullos');

INSERT INTO

CLIENTES(CODIGOPOSTAL,NOMBRE,APELLIDO,FECHANACIMIENTO,DNI,PROFESION,TELEFONO,EMAIL,RAZONVENIDA,DIRECCION,SEXO,FOTOURL,FIRMAURL)

VALUES(41110,'An','An',TO_DATE('2012-06-05','YYYY-MM-DD'),'27283847E','An',777283927,'sss@sss.ss','An','An','Hombre','https://asdasd.asda/asdasd.html','https://asdasd.asda/asdasd.html');

DELETE FROM Graduaciones;

END Inicializar;

PROCEDURE Insertar(nombre_prueba VARCHAR2,w_OID_C **NUMBER**,w_graduadoPor VARCHAR2,w_atendidoPor VARCHAR2,w_fecha **DATE**,salidaEsperada **BOOLEAN**) **AS**
salida **BOOLEAN** :=**TRUE**; *--Salida esperada*

aux **SMALLINT**;

graduacion **NUMBER**;

BEGIN

--Insertar graduaciones

INSERT INTO Graduaciones **VALUES**

(SEC_GRADUACION.NEXTVAL,w_OID_C,w_graduadoPor,w_atendidoPor,w_fecha);

COMMIT;

graduacion := SEC_GRADUACION.CURRVAL;

--Comprobar que todo se ha introducido OK.

SELECT COUNT(*) INTO aux

FROM Graduaciones **WHERE** OID_G= graduacion;*--Codigo postal es unico*

IF(AUX<>1) **THEN**

salida:=**FALSE**;

```
END IF;  
COMMIT;
```

```
--Mostrar resultado de la prueba
```

```
DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
```

```
EXCEPTION WHEN OTHERS THEN
```

```
DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
```

```
ASSERT_EQUALS(FALSE,salidaEsperada));
```

```
ROLLBACK;
```

```
END Insertar;
```

```
PROCEDURE Actualizar_cliente(nombre_prueba VARCHAR2,w_OID_G  
NUMBER,w_new_OID_C NUMBER,salidaEsperada BOOLEAN) AS
```

```
salida BOOLEAN :=TRUE; --Salida esperada
```

```
aux SMALLINT;
```

```
BEGIN
```

```
--Actualizar nombres
```

```
UPDATE Graduciones SET OID_C=w_new_OID_C WHERE OID_G=w_OID_G;
```

```
--Comprobar que todo se ha introducido OK.
```

```
SELECT COUNT(*) INTO aux
```

```
FROM Graduciones WHERE OID_G=w_OID_G AND OID_C =w_new_OID_C ;--
```

```
Codigo postal es unico
```

```
IF(AUX<>1) THEN
```

```
salida:=FALSE;
```

```
END IF;
```

```
COMMIT;
```

```
--Mostrar resultado de la prueba
```

```
DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
```

```
EXCEPTION WHEN OTHERS THEN
```

```
DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
```

```
ASSERT_EQUALS(FALSE,salidaEsperada));
```

```
ROLLBACK;
```

```
END Actualizar_cliente;
```

```
PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_OID_G NUMBER,salidaEsperada  
BOOLEAN) AS
```

```
salida BOOLEAN :=TRUE; --Salida esperada
```

```
aux SMALLINT;
```

```
BEGIN
```

```
--Insertar codigopostal
```

```
DELETE FROM Graduciones WHERE OID_G=w_OID_G;
```

```
COMMIT;
```

```
--Comprobar que todo se ha introducido OK.
```

```
SELECT COUNT(*) INTO aux
```

```
FROM Graduciones WHERE OID_G=w_OID_G;--Codigo postal es unico
```



```

IF(AUX<>0) THEN
    salida:=FALSE;
END IF;
COMMIT;

--Mostrar resultado de la prueba
DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
    DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
    ROLLBACK;
END Eliminar;
END Pruebas_Graduaciones;
/

-----
CREATE OR REPLACE PACKAGE Pruebas_Lentes AS
PROCEDURE Inicializar;
PROCEDURE Insertar(nombre_prueba VARCHAR2,w_IDREFERENCIAPRODUCTO
NUMBER,w_Curvatura NUMBER,salidaEsperada BOOLEAN);

PROCEDURE Actualizar_Curvatura(nombre_prueba
VARCHAR2,w_IDREFERENCIAPRODUCTO NUMBER,w_new_Curvatura
NUMBER,salidaEsperada BOOLEAN);

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_IDREFERENCIAPRODUCTO
NUMBER,salidaEsperada BOOLEAN) ;

END Pruebas_Lentes;
/

CREATE OR REPLACE PACKAGE BODY Pruebas_Lentes AS
PROCEDURE Inicializar AS
BEGIN
DELETE FROM PRODUCTOS;
INSERT INTO PRODUCTOS(nombre,cantidad,precio,iva) VALUES ('Producto
Prueba',10,'12,2','0,21');

END Inicializar;

PROCEDURE Insertar(nombre_prueba VARCHAR2,w_IDREFERENCIAPRODUCTO
NUMBER,w_Curvatura NUMBER,salidaEsperada BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
aux_id SMALLINT;
BEGIN
--Insertar graduaciones
INSERT INTO Lentes VALUES(w_IDREFERENCIAPRODUCTO,w_Curvatura);
COMMIT;

```

```

SELECT COUNT(*) INTO aux
FROM LENTES WHERE IDREFERENCIAPRODUCTO=
w_IDREFERENCIAPRODUCTO;

```

```

IF(AUX<>1) THEN
    salida:=FALSE;
END IF;
COMMIT;

```

--Mostrar resultado de la prueba

```

DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
    DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
ROLLBACK;
END Insertar;

```

```

PROCEDURE Actualizar_Curvatura(nombre_prueba
VARCHAR2,w_IDREFERENCIAPRODUCTO NUMBER,w_new_Curvatura
NUMBER,salidaEsperada BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;

```

BEGIN

--Actualizar nombres

```

UPDATE Lentes SET Curvatura=w_new_Curvatura WHERE
IDREFERENCIAPRODUCTO=w_IDREFERENCIAPRODUCTO;

```

--Comprobar que todo se ha introducido OK.

```

SELECT COUNT(*) INTO aux
FROM Lentes WHERE
IDREFERENCIAPRODUCTO=w_IDREFERENCIAPRODUCTO AND
Curvatura=w_new_Curvatura;

```

```

IF(AUX<>1) THEN
    salida:=FALSE;
END IF;
COMMIT;

```

--Mostrar resultado de la prueba

```

DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
    DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
ROLLBACK;
END Actualizar_Curvatura;

```

```

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_IDREFERENCIAPRODUCTO
NUMBER,salidaEsperada BOOLEAN) AS

```

```

salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
BEGIN
    --Insertar codigopostal
    DELETE FROM Lentes WHERE
IDREFERENCIAPRODUCTO=w_IDREFERENCIAPRODUCTO;
    COMMIT;

    --Comprobar que todo se ha introducido OK.
    SELECT COUNT(*) INTO aux
    FROM Lentes WHERE
IDREFERENCIAPRODUCTO=w_IDREFERENCIAPRODUCTO;

    IF(AUX<>0) THEN
        salida:=FALSE;
    END IF;
    COMMIT;

    --Mostrar resultado de la prueba
    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
    EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
        ROLLBACK;
    END Eliminar;

END Pruebas_Lentes;
/

-----

CREATE OR REPLACE PACKAGE Pruebas_Lentillas AS
PROCEDURE Inicializar;
PROCEDURE Insertar(nombre_prueba VARCHAR2,w_IDREFERENCIAPRODUCTO
NUMBER,w_TAMAÑO NUMBER,salidaEsperada BOOLEAN);

PROCEDURE Actualizar_tamaño(nombre_prueba
VARCHAR2,w_IDREFERENCIAPRODUCTO NUMBER,w_new_TAMAÑO
NUMBER,salidaEsperada BOOLEAN);

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_IDREFERENCIAPRODUCTO
NUMBER,salidaEsperada BOOLEAN) ;

END Pruebas_Lentillas;
/

CREATE OR REPLACE PACKAGE BODY Pruebas_Lentillas AS
PROCEDURE Inicializar AS
    BEGIN

```

```

DELETE FROM PRODUCTOS;
INSERT INTO PRODUCTOS(nombre,cantidad,precio,iva) VALUES ('Producto
Prueba',10,'12,2','0,21');

END Inicializar;

PROCEDURE Insertar(nombre_prueba VARCHAR2,w_IDREFERENCIAPRODUCTO
NUMBER,w_TAMAÑO NUMBER,salidaEsperada BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
aux_id SMALLINT;
BEGIN
    --Insertar graduaciones
    INSERT INTO Lentillas VALUES(w_IDREFERENCIAPRODUCTO,w_TAMAÑO);
    COMMIT;

    SELECT COUNT(*) INTO aux
    FROM LENTILLAS WHERE IDREFERENCIAPRODUCTO=
w_IDREFERENCIAPRODUCTO;

    IF(AUX<>1) THEN
        salida:=FALSE;
    END IF;
    COMMIT;

    --Mostrar resultado de la prueba
    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
    EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
        ROLLBACK;
    END Insertar;

PROCEDURE Actualizar_tamaño(nombre_prueba
VARCHAR2,w_IDREFERENCIAPRODUCTO NUMBER,w_new_TAMAÑO
NUMBER,salidaEsperada BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
BEGIN
    --Actualizar nombres
    UPDATE Lentillas SET TAMAÑO=w_new_TAMAÑO WHERE
IDREFERENCIAPRODUCTO=w_IDREFERENCIAPRODUCTO;

    --Comprobar que todo se ha introducido OK.
    SELECT COUNT(*) INTO aux
    FROM Lentillas WHERE
IDREFERENCIAPRODUCTO=w_IDREFERENCIAPRODUCTO AND
TAMAÑO=w_new_TAMAÑO;

```

```

IF(AUX<>1) THEN
    salida:=FALSE;
END IF;
COMMIT;

```

--Mostrar resultado de la prueba

```

DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
    DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
    ROLLBACK;
END Actualizar_tamaño;

```

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_IDREFERENCIAPRODUCTO
NUMBER,salidaEsperada **BOOLEAN**) **AS**

salida **BOOLEAN** :=**TRUE**; *--Salida esperada*
aux **SMALLINT**;

BEGIN

--Insertar codigopostal

```

DELETE FROM Lentillas WHERE
IDREFERENCIAPRODUCTO=w_IDREFERENCIAPRODUCTO;
COMMIT;

```

--Comprobar que todo se ha introducido OK.

```

SELECT COUNT(*) INTO aux
FROM Lentillas WHERE
IDREFERENCIAPRODUCTO=w_IDREFERENCIAPRODUCTO;

```

```

IF(AUX<>0) THEN

```

```

    salida:=FALSE;

```

```

END IF;

```

```

COMMIT;

```

--Mostrar resultado de la prueba

```

DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
    DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
    ROLLBACK;
END Eliminar;

```

END Pruebas_Lentillas;

/

CREATE OR REPLACE PACKAGE Pruebas_LinEnc **AS**

PROCEDURE Inicializar;

PROCEDURE Insertar(nombre_prueba VARCHAR2,w_ID_referenciaProducto

NUMBER,w_idReferenciaEncargo **NUMBER**,w_cantidad **NUMBER**,salidaEsperada **BOOLEAN**);

```

PROCEDURE Actualizar_enc(nombre_prueba VARCHAR2,w_OID_LE NUMBER,w_new_Enc
NUMBER,salidaEsperada BOOLEAN);
PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_OID_LE NUMBER,salidaEsperada
BOOLEAN);
END Pruebas_LinEnc;
/

```

```

-----
CREATE OR REPLACE PACKAGE BODY Pruebas_LinEnc AS

```

```

PROCEDURE Inicializar AS

```

```

BEGIN

```

```

DELETE FROM CLIENTES;

```

```

DELETE FROM CODIGOSPOSTALES;

```

```

DELETE FROM PRODUCTOS;

```

```

DELETE FROM ENCARGOS;

```

```

INSERT INTO CODIGOSPOSTALES VALUES(41110,'Sevilla','Bollullos');

```

```

INSERT INTO

```

```

CLIENTES(CODIGOPOSTAL,NOMBRE,APELLIDO,FECHANACIMIENTO,DNI,PROFESION,
TELEFONO,EMAIL,RAZONVENIDA,DIRECCION,SEXO,FOTOURL,FIRMAURL)

```

```

VALUES(41110,'An','An',TO_DATE('2012-06-05','YYYY-MM-
DD'),'27283847E','An',777283927,'sss@sss.ss','An','An','Hombre','https://asdasd.asda/
asdasd.html','https://asdasd.asda/asdasd.html');

```

```

INSERT INTO PRODUCTOS(nombre,cantidad,precio,iva)VALUES ('gafas pull',
10,'21,2','0,21');

```

```

INSERT INTO Encargos(OID_C,fechaPedido)

```

```

VALUES(SEC_CLIENTE.CURRVAL,SYSDATE);

```

```

INSERT INTO Encargos(OID_C,fechaPedido)

```

```

VALUES(SEC_CLIENTE.CURRVAL,SYSDATE);

```

```

DELETE FROM Graduaciones;

```

```

END Inicializar;

```

```

PROCEDURE Insertar(nombre_prueba VARCHAR2,w_ID_referenciaProducto
NUMBER,w_idReferenciaEncargo NUMBER,w_cantidad NUMBER,salidaEsperada BOOLEAN)
AS

```

```

salida BOOLEAN :=TRUE; --Salida esperada

```

```

aux SMALLINT;

```

```

aux_id SMALLINT;

```

```

BEGIN

```

```

--Insertar graduaciones

```

```

INSERT INTO

```

```

LineasEncargo(idReferenciaProducto,idReferenciaEncargo,cantidad,precioUnitario)

```

```

VALUES(w_ID_referenciaProducto,w_idReferenciaEncargo,w_cantidad,(SELECT
precioIVA FROM Productos WHERE idReferenciaProducto = w_ID_referenciaProducto));
COMMIT;

```

```

UPDATE PRODUCTOS SET CANTIDAD = CANTIDAD - w_cantidad where
idReferenciaProducto = w_ID_referenciaProducto;

```

COMMIT;

--Comprobar que todo se ha introducido OK.

aux_id := SEC_LINEAENCARGO.CURRVAL;

SELECT COUNT(*) INTO aux

FROM LineasEncargo **WHERE** OID_LE= aux_id; *--Codigo postal es unico*

IF(AUX<>1) **THEN**

salida:=**FALSE**;

END IF;

COMMIT;

--Mostrar resultado de la prueba

DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

EXCEPTION WHEN OTHERS THEN

DBMS_OUTPUT.put_line(nombre_prueba || ':' ||

ASSERT_EQUALS(**FALSE**,salidaEsperada));

ROLLBACK;

END Insertar;

PROCEDURE Actualizar_enc(nombre_prueba VARCHAR2,w_OID_LE **NUMBER**,w_new_Enc
NUMBER,salidaEsperada **BOOLEAN**) **AS**

salida **BOOLEAN** :=**TRUE**; *--Salida esperada*

aux **SMALLINT**;

BEGIN

--Actualizar nombres

UPDATE LineasEncargo **SET** idReferenciaEncargo=w_new_Enc **WHERE** OID_LE=
w_OID_LE;

--Comprobar que todo se ha introducido OK.

SELECT COUNT(*) INTO aux

FROM LineasEncargo **WHERE** OID_LE= w_OID_LE ; *--Codigo postal es unico*

IF(AUX<>1) **THEN**

salida:=**FALSE**;

END IF;

COMMIT;

--Mostrar resultado de la prueba

DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

EXCEPTION WHEN OTHERS THEN

DBMS_OUTPUT.put_line(nombre_prueba || ':' ||

ASSERT_EQUALS(**FALSE**,salidaEsperada));

ROLLBACK;

END Actualizar_enc;

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_OID_LE **NUMBER**,salidaEsperada
BOOLEAN) **AS**

salida **BOOLEAN** :=**TRUE**; *--Salida esperada*

```

aux SMALLINT;
BEGIN
    --Insertar codigopostal
    DELETE FROM LineasEncargo WHERE OID_LE= w_OID_LE;
    COMMIT;

    --Comprobar que todo se ha introducido OK.
    SELECT COUNT(*) INTO aux
        FROM LineasEncargo WHERE OID_LE= w_OID_LE;--Codigo postal es unico

    IF(AUX<>0) THEN
        salida:=FALSE;
    END IF;
    COMMIT;

    --Mostrar resultado de la prueba
    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
    EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
        ROLLBACK;
    END Eliminar;

END Pruebas_LinEnc;
/

-----

CREATE OR REPLACE PACKAGE Pruebas_Monturas AS
PROCEDURE Inicializar;
PROCEDURE Insertar(nombre_prueba VARCHAR2,w_IDREFERENCIAPRODUCTO
NUMBER,w_FAMILIAMONTURA VARCHAR,w_SEXO VARCHAR,salidaEsperada
BOOLEAN);

PROCEDURE Actualizar_sexo(nombre_prueba VARCHAR2,w_IDREFERENCIAPRODUCTO
NUMBER,w_new_SEXO VARCHAR,salidaEsperada BOOLEAN);

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_IDREFERENCIAPRODUCTO
NUMBER,salidaEsperada BOOLEAN) ;

END Pruebas_Monturas;
/

CREATE OR REPLACE PACKAGE BODY Pruebas_Monturas AS
PROCEDURE Inicializar AS
BEGIN
    DELETE FROM PRODUCTOS;
    INSERT INTO PRODUCTOS(nombre,cantidad,precio,iva) VALUES ('Producto
Prueba',10,'12,2','0,21');

```


END Inicializar;

PROCEDURE Insertar(nombre_prueba VARCHAR2,w_IDREFERENCIAPRODUCTO
NUMBER,w_FAMILIAMONTURA VARCHAR,w_SEXO VARCHAR,salidaEsperada
BOOLEAN) **AS**

salida **BOOLEAN** :=**TRUE**; --Salida esperada

aux **SMALLINT**;

aux_id **SMALLINT**;

BEGIN

--Insertar graduaciones

INSERT INTO Monturas

VALUES(w_IDREFERENCIAPRODUCTO,w_FAMILIAMONTURA,w_SEXO);

COMMIT;

SELECT COUNT(*) INTO aux

FROM Monturas **WHERE** IDREFERENCIAPRODUCTO=
w_IDREFERENCIAPRODUCTO;

IF(AUX<>1) **THEN**

salida:=**FALSE**;

END IF;

COMMIT;

--Mostrar resultado de la prueba

DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

EXCEPTION WHEN OTHERS THEN

DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(**FALSE**,salidaEsperada));

ROLLBACK;

END Insertar;

PROCEDURE Actualizar_sexo(nombre_prueba VARCHAR2,w_IDREFERENCIAPRODUCTO
NUMBER,w_new_SEXO VARCHAR,salidaEsperada **BOOLEAN**) **AS**

salida **BOOLEAN** :=**TRUE**; --Salida esperada

aux **SMALLINT**;

BEGIN

--Actualizar nombres

UPDATE Monturas **SET** SEXO=w_new_SEXO **WHERE**

IDREFERENCIAPRODUCTO=w_IDREFERENCIAPRODUCTO;

--Comprobar que todo se ha introducido OK.

SELECT COUNT(*) INTO aux

FROM Monturas **WHERE**

IDREFERENCIAPRODUCTO=w_IDREFERENCIAPRODUCTO **AND** SEXO=w_new_SEXO;

IF(AUX<>1) **THEN**

salida:=**FALSE**;

END IF;

COMMIT;

```

--Mostrar resultado de la prueba
DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
    DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
    ROLLBACK;
END Actualizar_sexo;

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_IDREFERENCIAPRODUCTO
NUMBER,salidaEsperada BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
BEGIN
    --Insertar codigopostal
    DELETE FROM Monturas WHERE
IDREFERENCIAPRODUCTO=w_IDREFERENCIAPRODUCTO;
    COMMIT;

    --Comprobar que todo se ha introducido OK.
    SELECT COUNT(*) INTO aux
    FROM Monturas WHERE
IDREFERENCIAPRODUCTO=w_IDREFERENCIAPRODUCTO;

    IF(AUX<>0) THEN
        salida:=FALSE;
    END IF;
    COMMIT;

    --Mostrar resultado de la prueba
    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
    EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
        ROLLBACK;
    END Eliminar;

END Pruebas_Monturas;
/
-----
-----
CREATE OR REPLACE PACKAGE Pruebas_Ofertas AS
PROCEDURE Inicializar;
PROCEDURE Insertar(nombre_prueba VARCHAR2,w_nombreOferta
VARCHAR2,salidaEsperada BOOLEAN);
PROCEDURE Actualizar_nom(nombre_prueba VARCHAR2,w_nombreOferta
VARCHAR2,w_new_nom VARCHAR2,salidaEsperada BOOLEAN);
PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_nombreOferta
VARCHAR2,salidaEsperada BOOLEAN);

```

END Pruebas_Ofertas;

/

CREATE OR REPLACE PACKAGE BODY Pruebas_Ofertas **AS**

PROCEDURE Inicializar **AS**

BEGIN

DELETE FROM Ofertas;

END Inicializar;

PROCEDURE Insertar(nombre_prueba VARCHAR2,w_nombreOferta
VARCHAR2,salidaEsperada **BOOLEAN**) **AS**

salida **BOOLEAN** :=**TRUE**; --Salida esperada

aux **SMALLINT**;

BEGIN

--Insertar graduaciones

INSERT INTO Ofertas **VALUES**(w_nombreOferta);

COMMIT;

--Comprobar que todo se ha introducido OK.

SELECT COUNT(*) **INTO** aux

FROM Ofertas **WHERE** nombreOferta= w_nombreOferta;--Codigo postal es unico

IF(AUX<>1) **THEN**

salida:=**FALSE**;

END IF;

COMMIT;

--Mostrar resultado de la prueba

DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

EXCEPTION WHEN OTHERS THEN

DBMS_OUTPUT.put_line(nombre_prueba || ':' ||

ASSERT_EQUALS(**FALSE**,salidaEsperada));

ROLLBACK;

END Insertar;

PROCEDURE Actualizar_nom(nombre_prueba VARCHAR2,w_nombreOferta
VARCHAR2,w_new_nom VARCHAR2,salidaEsperada **BOOLEAN**) **AS**

salida **BOOLEAN** :=**TRUE**; --Salida esperada

aux **SMALLINT**;

BEGIN

--Actualizar nombres

UPDATE Ofertas **SET** nombreOferta=w_new_nom **WHERE**

nombreOferta=w_nombreOferta;

SELECT COUNT(*) **INTO** aux

FROM Ofertas **WHERE** nombreOferta= w_new_nom;--Codigo postal es unico

--Comprobar que todo se ha introducido OK.

IF(AUX<>1) **THEN**

```

        salida:=FALSE;
END IF;
COMMIT;

--Mostrar resultado de la prueba
DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
        ROLLBACK;
END Actualizar_nom;

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_nombreOferta
VARCHAR2,salidaEsperada BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
BEGIN
        --Insertar codigopostal
        DELETE FROM Ofertas WHERE nombreOferta=w_nombreOferta;
        COMMIT;

        --Comprobar que todo se ha introducido OK.
        SELECT COUNT(*) INTO aux
        FROM Ofertas WHERE nombreOferta=w_nombreOferta;--Codigo postal es unico

        IF(AUX<>0) THEN
                salida:=FALSE;
        END IF;
        COMMIT;

        --Mostrar resultado de la prueba
        DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
        EXCEPTION WHEN OTHERS THEN
                DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
                ROLLBACK;
        END Eliminar;

END Pruebas_Ofertas;
/

-----
-----
CREATE OR REPLACE PACKAGE Pruebas_Ojos AS
PROCEDURE Inicializar;
PROCEDURE Insertar(nombre_prueba VARCHAR2,w_OID_G NUMBER,w_CILINDRO
NUMBER,w_ESTOJO NUMBER,w_EJE NUMBER,w_ESTECERCA NUMBER
,w_LEJOS NUMBER,w_CERCA NUMBER,w_CC NUMBER,w_QUERATOMETRIA
NUMBER,w_PRISMA NUMBER,w_SC NUMBER,salidaEsperada BOOLEAN);
PROCEDURE Actualizar_lejos(nombre_prueba VARCHAR2,w_OID_O NUMBER,w_new_lejos

```

```

NUMBER,salidaEsperada BOOLEAN);
PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_OID_O NUMBER,salidaEsperada
BOOLEAN);
END Pruebas_Ojos;
/

```

```

-----
CREATE OR REPLACE PACKAGE BODY Pruebas_Ojos AS

```

```

PROCEDURE Inicializar AS

```

```

BEGIN

```

```

DELETE FROM Graduaciones;

```

```

DELETE FROM CLIENTES;

```

```

DELETE FROM CODIGOSPOSTALES;

```

```

INSERT INTO CODIGOSPOSTALES VALUES(41110,'Sevilla','Bollullos');

```

```

INSERT INTO

```

```

CLIENTES(CODIGOPOSTAL,NOMBRE,APELLIDO,FECHANACIMIENTO,DNI,PROFESION,
TELEFONO,EMAIL,RAZONVENIDA,DIRECCION,SEXO,FOTOURL,FIRMAURL)

```

```

VALUES(41110,'An','An',TO_DATE('2012-06-05','YYYY-MM-DD'),
'27283847E','An',777283927,'sss@sss.ss','An','An','Hombre',
'https://asdasd.asda/asdasd.html','https://asdasd.asda/asdasd.html');

```

```

INSERT INTO GRADUACIONES(OID_C,GRADUADOPOR,ATENDIDOPOR,FECHA)
VALUES(SEC_CLIENTE.CURRVAL,'ee','ee',SYSDATE);

```

```

DELETE FROM OJOS;

```

```

END Inicializar;

```

```

PROCEDURE Insertar(nombre_prueba VARCHAR2,w_OID_G NUMBER,w_CILINDRO
NUMBER,w_ESTOJO NUMBER,w_EJE NUMBER,w_ESTECERCA NUMBER
,w_LEJOS NUMBER,w_CERCA NUMBER,w_CC NUMBER,w_QUERATOMETRIA
NUMBER,w_PRISMA NUMBER,w_SC NUMBER,salidaEsperada BOOLEAN) AS

```

```

salida BOOLEAN :=TRUE; --Salida esperada

```

```

aux SMALLINT;

```

```

ojo NUMBER;

```

```

BEGIN

```

```

--Insertar graduaciones

```

```

INSERT INTO

```

```

OJOS(OID_G,CILINDRO ,ESTOJO ,EJE ,ESTECERCA ,LEJOS ,CERCA ,CC ,QUERATOMET
RIA ,PRISMA ,SC) VALUES

```

```

(w_OID_G ,w_CILINDRO ,w_ESTOJO ,w_EJE ,w_ESTECERCA ,w_LEJOS ,w_CERCA ,w_CC
,w_QUERATOMETRIA ,w_PRISMA ,w_SC);

```

```

COMMIT;

```

```

ojo := SEC_OJO.CURRVAL;

```

```

--Comprobar que todo se ha introducido OK.

```

```

SELECT COUNT(*) INTO aux

```

```

FROM OJOS WHERE OID_O= ojo;--Codigo postal es unico

```

```

IF(AUX<>1) THEN

```

```

salida:=FALSE;

```

```

END IF;

```

```

COMMIT;

```

```

--Mostrar resultado de la prueba

```

```

        DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
    EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
        ROLLBACK;
    END Insertar;

```

```

PROCEDURE Actualizar_lejos(nombre_prueba VARCHAR2,w_OID_O NUMBER,w_new_lejos
NUMBER,salidaEsperada BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;

```

```

    BEGIN
        --Actualizar nombres
        UPDATE OJOS SET lejos=w_new_lejos WHERE OID_O=w_OID_O;

        --Comprobar que todo se ha introducido OK.
        SELECT COUNT(*) INTO aux
            FROM OJOS WHERE OID_O=w_OID_O AND lejos =w_new_lejos ;--Codigo postal es
unico

```

```

        IF(AUX<>1) THEN
            salida:=FALSE;
        END IF;
        COMMIT;

```

```

        --Mostrar resultado de la prueba
        DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
    EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
        ROLLBACK;
    END Actualizar_lejos;

```

```

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_OID_O NUMBER,salidaEsperada
BOOLEAN) AS

```

```

salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;

```

```

    BEGIN
        --Insertar codigopostal
        DELETE FROM Ojos WHERE OID_O=w_OID_O;
        COMMIT;

```

```

        --Comprobar que todo se ha introducido OK.
        SELECT COUNT(*) INTO aux
            FROM OJOS WHERE OID_O=w_OID_O;--Codigo postal es unico

```

```

        IF(AUX<>0) THEN
            salida:=FALSE;
        END IF;

```

COMMIT;

--Mostrar resultado de la prueba

DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

EXCEPTION WHEN OTHERS THEN

DBMS_OUTPUT.put_line(nombre_prueba || ':' ||

ASSERT_EQUALS(**FALSE**,salidaEsperada));

ROLLBACK;

END Eliminar;

END Pruebas_Ojos;

/

CREATE OR REPLACE FUNCTION ASSERT_EQUALS (salida **BOOLEAN**, salida_esperada **BOOLEAN**)

RETURN VARCHAR2 **AS**

BEGIN

IF(salida= salida_esperada) **THEN**

RETURN 'EXITO';

ELSE

RETURN 'FALLO';

END IF;

END ASSERT_EQUALS;

/

CREATE OR REPLACE PACKAGE Pruebas_Presentan **AS**

PROCEDURE Inicializar;

PROCEDURE Insertar(nombre_prueba VARCHAR2,w_nombreOferta
VARCHAR2,w_nombreFranquicia VARCHAR2,salidaEsperada **BOOLEAN**);

PROCEDURE Actualizar_nom(nombre_prueba VARCHAR2,w_OID_P **NUMBER**,w_nFran
VARCHAR2,salidaEsperada **BOOLEAN**);

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_OID_P **NUMBER**,salidaEsperada
BOOLEAN);

END Pruebas_Presentan;

/

CREATE OR REPLACE PACKAGE BODY Pruebas_Presentan **AS**

PROCEDURE Inicializar **AS**

BEGIN

DELETE FROM Presentan;

DELETE FROM OFERTAS;

DELETE FROM FRANQUICIAS;

INSERT INTO Ofertas(nombreOferta) **VALUES** ('Oferta de Prueba');

INSERT INTO Franquicias (nombreFranquicia, direccion, propietario) **VALUES** ('Franquicia
de Prueba', 'C/Prueba', 'Tester');

INSERT INTO Franquicias (nombreFranquicia, direccion, propietario) **VALUES** ('Optica
Nombre Nuevo', 'C/Prueba', 'Tester');

END Inicializar;

```

PROCEDURE Insertar(nombre_prueba VARCHAR2,w_nombreOferta
VARCHAR2,w_nombreFranquicia VARCHAR2,salidaEsperada BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux_id SMALLINT;
aux_lin Presentan%ROWTYPE;
BEGIN
    --Insertar codigopostal
    INSERT INTO Presentan(nombreOferta, nombreFranquicia)
VALUES(w_nombreOferta,w_nombreFranquicia);
    COMMIT;

    --Comprobar que todo se ha introducido OK.
    aux_id :=SEC_Presentan.CURRVAL;
    SELECT OID_P ,NOMBREOFERTA , NOMBREFRANQUICIA INTO aux_lin
    FROM Presentan WHERE OID_P=aux_id;

    IF(aux_lin.nombreOferta<>w_nombreOferta OR
aux_lin.nombreFranquicia<>w_nombreFranquicia) THEN
        salida:=FALSE;
    END IF;
    COMMIT;

    --Mostrar resultado de la prueba
    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
    EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
        ROLLBACK;
    END Insertar;

PROCEDURE Actualizar_nom(nombre_prueba VARCHAR2,w_OID_P NUMBER,w_nFran
VARCHAR2,salidaEsperada BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux_lin NUMBER;
BEGIN
    --Actualizar nombres
    UPDATE Presentan SET nombreFranquicia=w_nFran WHERE OID_P=w_OID_P;
    COMMIT;

    --Comprobar que todo se ha introducido OK.
    SELECT COUNT(*) INTO aux_lin
    FROM Presentan WHERE OID_P=w_OID_P AND nombreFranquicia=w_nFran;

    IF(aux_lin<>1) THEN
        salida:=FALSE;
    END IF;
    COMMIT;

```



```

--Mostrar resultado de la prueba
DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
    DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
    ROLLBACK;
END Actualizar_nom;

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_OID_P NUMBER,salidaEsperada
BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
BEGIN
    --Insertar codigopostal
    DELETE FROM Presentan WHERE OID_P=w_OID_P;
    COMMIT;

    --Comprobar que todo se ha introducido OK.
    SELECT COUNT(*) INTO aux
        FROM Presentan WHERE OID_P=w_OID_P;--Codigo postal es unico

    IF(AUX<>0) THEN
        salida:=FALSE;
    END IF;
    COMMIT;

    --Mostrar resultado de la prueba
    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
    EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
        ROLLBACK;
    END Eliminar;
END Pruebas_Presentan;
/

-----
-----
CREATE OR REPLACE PACKAGE Pruebas_Producto AS
PROCEDURE Inicializar;
PROCEDURE Insertar(nombre_prueba VARCHAR2,w_nombre VARCHAR2,w_CANTIDAD
NUMBER,w_precio NUMBER,w_iva NUMBER,salidaEsperada BOOLEAN);
PROCEDURE Actualizar_nom(nombre_prueba VARCHAR2,w_OID_Prod
NUMBER,w_new_nom VARCHAR2,salidaEsperada BOOLEAN);
PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_OID_Prod NUMBER,salidaEsperada
BOOLEAN);
END Pruebas_Producto;
/

CREATE OR REPLACE PACKAGE BODY Pruebas_Producto AS

```

PROCEDURE Inicializar **AS**

BEGIN

DELETE FROM Productos;

END Inicializar;

PROCEDURE Insertar(nombre_prueba VARCHAR2,w_nombre VARCHAR2,w_CANTIDAD
NUMBER,w_precio NUMBER,w_iva NUMBER,salidaEsperada **BOOLEAN**) **AS**

salida **BOOLEAN** :=**TRUE**; --Salida esperada

aux **SMALLINT**;

aux_id number;

BEGIN

--Insertar graduaciones

INSERT INTO Productos(nombre,cantidad,precio,descuento,iva)

VALUES(w_nombre,w_cantidad,w_precio,'0,0',w_iva);

COMMIT;

--Comprobar que todo se ha introducido OK.

aux_id := SEC_IDREFERENCIAPRODUCTO.CURRVAL;

SELECT COUNT(*) **INTO** aux

FROM Productos **WHERE** IDREFERENCIAPRODUCTO= aux_id;--Codigo postal es

unico

IF(AUX<>1) **THEN**

salida:=**FALSE**;

END IF;

COMMIT;

--Mostrar resultado de la prueba

DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

EXCEPTION WHEN OTHERS THEN

DBMS_OUTPUT.put_line(nombre_prueba || ':' ||

ASSERT_EQUALS(**FALSE**,salidaEsperada));

ROLLBACK;

END Insertar;

PROCEDURE Actualizar_nom(nombre_prueba VARCHAR2,w_OID_Prod
NUMBER,w_new_nom VARCHAR2,salidaEsperada **BOOLEAN**) **AS**

salida **BOOLEAN** :=**TRUE**; --Salida esperada

aux_lin Productos%ROWTYPE;

aux **SMALLINT**;

BEGIN

--Actualizar nombres

UPDATE Productos **SET** nombre=w_new_nom **WHERE**

IDREFERENCIAPRODUCTO=w_OID_Prod;

--Comprobar que todo se ha introducido OK.

SELECT

```

IDREFERENCIAPRODUCTO ,NOMBRE ,CANTIDAD ,PRECIO ,IVA ,PRECIOIVA ,DESCUEN
TO  INTO aux_lin
    FROM Productos WHERE IDREFERENCIAPRODUCTO=w_OID_Prod;

    --Tambien podria ser count(*) WHERE IDREFERENCIAPRODUCTO=w_OID_Prod AND
    nombre=w_new_nom

    IF(aux_lin.NOMBRE<>w_new_nom) THEN
        salida:=FALSE;
    END IF;
    COMMIT;

    --Mostrar resultado de la prueba
    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
    EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
    ASSERT_EQUALS(FALSE,salidaEsperada));
        ROLLBACK;
    END Actualizar_nom;

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_OID_Prod NUMBER,salidaEsperada
BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
BEGIN
    --Insertar codigopostal
    DELETE FROM Productos WHERE IDREFERENCIAPRODUCTO=w_OID_Prod;
    COMMIT;

    --Comprobar que todo se ha introducido OK.
    SELECT COUNT(*) INTO aux
        FROM Productos WHERE IDREFERENCIAPRODUCTO=w_OID_Prod;--Codigo postal
    es unico

    IF(AUX<>0) THEN
        salida:=FALSE;
    END IF;
    COMMIT;

    --Mostrar resultado de la prueba
    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
    EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
    ASSERT_EQUALS(FALSE,salidaEsperada));
        ROLLBACK;
    END Eliminar;

END Pruebas_Producto;
/

```

```
CREATE OR REPLACE PACKAGE Pruebas_SeEnvianA AS
```

```
PROCEDURE Inicializar;
```

```
PROCEDURE Insertar(nombre_prueba VARCHAR2,w_nombreOferta VARCHAR2,w_OID_C  
NUMBER,salidaEsperada BOOLEAN);
```

```
PROCEDURE Actualizar_cod(nombre_prueba VARCHAR2,w_OID_SEA NUMBER,w_nOID_C  
NUMBER,salidaEsperada BOOLEAN);
```

```
PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_OID_SEA NUMBER,salidaEsperada  
BOOLEAN);
```

```
END Pruebas_SeEnvianA;
```

```
/
```

```
CREATE OR REPLACE PACKAGE BODY Pruebas_SeEnvianA AS
```

```
PROCEDURE Inicializar AS
```

```
  BEGIN
```

```
    DELETE FROM Clientes;
```

```
    DELETE FROM CodigosPostales;
```

```
    DELETE FROM OFERTAS;
```

```
    INSERT INTO Ofertas(nombreOferta) VALUES ('Oferta de Prueba');
```

```
    INSERT INTO CODIGOSPOSTALES VALUES(41110,'Sevilla','Bollullos');
```

```
    INSERT INTO
```

```
CLIENTES(CODIGOPOSTAL,NOMBRE,APELLIDO,FECHANACIMIENTO,DNI,PROFESION,  
TELEFONO,EMAIL,RAZONVENIDA,DIRECCION,SEXO,FOTOURL,FIRMAURL)
```

```
  VALUES(41110,'NENE','NANO',TO_DATE('2012-06-05','YYYY-MM-  
DD'),'23283847X','An',777281927,'www@sss.ss','An','An','Hombre','https://asdasd.asda/  
asdasd.html','https://asdasd.asda/asdasd.html');
```

```
    INSERT INTO
```

```
CLIENTES(CODIGOPOSTAL,NOMBRE,APELLIDO,FECHANACIMIENTO,DNI,PROFESION,  
TELEFONO,EMAIL,RAZONVENIDA,DIRECCION,SEXO,FOTOURL,FIRMAURL)
```

```
  VALUES(41110,'An','An',TO_DATE('2012-06-05','YYYY-MM-  
DD'),'27283847E','An',777283927,'sss@sss.ss','An','An','Hombre','https://asdasd.asda/  
asdasd.html','https://asdasd.asda/asdasd.html');
```

```
    DELETE FROM SeEnvianA
```

```
    COMMIT;
```

```
  END Inicializar;
```

```
PROCEDURE Insertar(nombre_prueba VARCHAR2,w_nombreOferta VARCHAR2,w_OID_C  
NUMBER,salidaEsperada BOOLEAN) AS
```

```
salida BOOLEAN := TRUE; --Salida esperada
```

```
aux_id SMALLINT;
```

```
aux SMALLINT;
```

```
  BEGIN
```

```

--Insertar codigopostal
INSERT INTO SeEnvianA(nombreOferta, OID_C) VALUES(w_nombreOferta,w_OID_C);
COMMIT;

--Comprobar que todo se ha introducido OK.
aux_id := SEC_SeEnvianA.CURRVAL;

SELECT COUNT(*) INTO aux FROM SEENVIANA WHERE OID_SEA=aux_id;

IF(aux<>1) THEN
    SALIDA:=FALSE;
END IF;
COMMIT;

--Mostrar resultado de la prueba
DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
EXCEPTION WHEN OTHERS THEN
    DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
    ROLLBACK;
END Insertar;

PROCEDURE Actualizar_cod(nombre_prueba VARCHAR2,w_OID_SEA NUMBER,w_nOID_C
NUMBER,salidaEsperada BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux_id SMALLINT;
aux SMALLINT;
BEGIN
    --Actualizar nombres
    UPDATE SeEnvianA SET OID_C=w_nOID_C WHERE OID_SEA=w_OID_SEA;

    --Comprobar que todo se ha introducido OK.
    SELECT COUNT(*) INTO aux FROM SEENVIANA WHERE OID_SEA=w_OID_SEA
AND OID_C = w_nOID_C;

    IF(aux<>1) THEN
        SALIDA:=FALSE;
    END IF;
    COMMIT;

    --Mostrar resultado de la prueba
    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
    EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
        ROLLBACK;
    END Actualizar_cod;

PROCEDURE Eliminar(nombre_prueba VARCHAR2,w_OID_SEA NUMBER,salidaEsperada

```

```

BOOLEAN) AS
salida BOOLEAN :=TRUE; --Salida esperada
aux SMALLINT;
BEGIN
    --Insertar codigopostal
    DELETE FROM SeEnvianA WHERE OID_SEA=w_OID_SEA;
    COMMIT;

    --Comprobar que todo se ha introducido OK.
    SELECT COUNT(*) INTO aux
        FROM SeEnvianA WHERE OID_SEA=w_OID_SEA;

    IF(AUX<>0) THEN
        salida:=FALSE;
    END IF;
    COMMIT;

    --Mostrar resultado de la prueba
    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
    EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(FALSE,salidaEsperada));
        ROLLBACK;
    END Eliminar;
END Pruebas_SeEnvianA;
/

```

12.SCRIPTS - PRUEBAS CON PAQUETES

```
SET SERVEROUTPUT ON;
```

```
DECLARE
```

```
aviso NUMBER;
```

```
producto NUMBER;
```

```
BEGIN
```

```
DBMS_OUTPUT.put_line('Paquete Avisos');
```

```
Pruebas_Avisos.Inicializar;
```

```
producto:= SEC_IDREFERENCIAPRODUCTO.CURRVAL;
```

```
Pruebas_Avisos.Insertar('Prueba1',producto,true);
```

```
aviso:= SEC_AVISOS.CURRVAL;
```

```
Pruebas_Avisos.Actualizar_descripcion('Prueba2',aviso,'eoeo',true);
```

```
Pruebas_Avisos.Actualizar_descripcion('Prueba3',aviso + 1,'eoeo',false);
```

```
Pruebas_Avisos.Eliminar('Prueba4',aviso,true);
```

```
END;
```

```
/
```

```
BEGIN
```

```
DBMS_OUTPUT.put_line('Paquete Clientes');
```

```
Pruebas_Cliente.Inicializar;
```

```
Pruebas_Cliente.Insertar('Prueba1',41110,'Ant','Per',SYSDATE -
```

```
1,'42357456R','Peletero',635276984,'aaa@aaa.aa','Veo poco','C/eo,89','Hombre','https://google.com/
```

```
flights','https://google.com/flights',true);
```

```
Pruebas_Cliente.Insertar('Prueba2',41110,'Jos','Tab',SYSDATE - 4,'42357426R','Financieramente
```

```
libre',737848991,'aaa@qwe.aa','Veo
```

```
poco','C/qwe,89','Hombre','https://google.com/flights','https://google.com/flights',true);
```

```
Pruebas_Cliente.Actualizar_telefono('Prueba3',SEC_CLIENTE.CURRVAL,666000888,true);
```

```
Pruebas_Cliente.Eliminar('Prueba4',SEC_CLIENTE.CURRVAL,true);
```

```
END;
```

```
/
```

```
BEGIN
```

```
DBMS_OUTPUT.put_line('Paquete CodigosPostales');
```

```
Pruebas_CodPost.Inicializar;
```

```
Pruebas_CodPost.Insertar('Prueba1',41120,'SEVILLA','GELVES',true);
```

```
Pruebas_CodPost.Actualizar_cod('Prueba2',41120,41130,true);
```

```
Pruebas_CodPost.Eliminar('Prueba3',41130,true);
```

```
END;
```

```
/
```

```
BEGIN
```

```

DBMS_OUTPUT.put_line('Paquete Encargos');
Pruebas_Encargos.Inicializar;
Pruebas_Encargos.Insertar('Prueba1',SEC_CLIENTE.CURRVAL,true);
Pruebas_Encargos.Actualizar_cli('Prueba2',SEC_IDREFERENCIA_ENCARGO.CURRVAL,SEC_
CLIENTE.CURRVAL,true);
Pruebas_Encargos.Eliminar('Prueba3',SEC_IDREFERENCIA_ENCARGO.CURRVAL,true);
END;
/

BEGIN
DBMS_OUTPUT.put_line('Paquete Franquicia');
Pruebas_Franquicia.Inicializar;
Pruebas_Franquicia.Insertar('Prueba1','Optica Maguilla','Av. Ronda de Triana 21','Jesus',true);
Pruebas_Franquicia.Actualizar_nombre('Prueba2','Optica Maguilla','Opticalia',true);
Pruebas_Franquicia.Eliminar('Prueba3','Optica Maguilla',true);

END;
/

DECLARE
cliente NUMBER;
graduacion NUMBER;

BEGIN
DBMS_OUTPUT.put_line('Paquete Graduaciones');
Pruebas_Graduaciones.Inicializar;
cliente:= SEC_CLIENTE.CURRVAL;
Pruebas_Graduaciones.Insertar('Prueba1',cliente,'eee','eee',SYSDATE,true);
graduacion:= SEC_GRADUACION.CURRVAL;
Pruebas_Graduaciones.Insertar('Prueba2',cliente,'eee','eee',SYSDATE + 1,false);
Pruebas_Graduaciones.Actualizar_cliente('Prueba3',graduacion,cliente,true);
Pruebas_Graduaciones.Eliminar('Prueba4',graduacion,true);

END;
/

DECLARE
IDprod NUMBER;
BEGIN
DBMS_OUTPUT.put_line('Paquete Lentes');
Pruebas_Lentes.Inicializar;
IDprod := SEC_IDREFERENCIAPRODUCTO.CURRVAL;
Pruebas_Lentes.Insertar('Prueba1',IDprod,'0,2',true);
Pruebas_Lentes.Actualizar_Curvatura('Prueba2',IDprod,'0,3',true);
Pruebas_Lentes.Eliminar('Prueba3',IDprod,true);

END;
/

DECLARE
IDprod NUMBER;
BEGIN
DBMS_OUTPUT.put_line('Paquete Lentillas');
Pruebas_Lentillas.Inicializar;

```



```

IDprod := SEC_IDREFERENCIAPRODUCTO.CURRVAL;
Pruebas_Lentillas.Insertar('Prueba1',IDprod,'0,2',true);
Pruebas_Lentillas.Actualizar_tamaño('Prueba2',IDprod,'0,3',true);
Pruebas_Lentillas.Eliminar('Prueba3',IDprod,true);

END;
/

BEGIN
DBMS_OUTPUT.put_line('Paquete LineasEncargo');
Pruebas_LinEnc.Inicializar;
Pruebas_LinEnc.Insertar('Prueba1',SEC_IDREFERENCIAPRODUCTO.CURRVAL,SEC_IDREFERENCIA_ENCARGO.CURRVAL,7,true);
Pruebas_LinEnc.Actualizar_enc('Prueba2',SEC_LINEAENCARGO.CURRVAL,SEC_IDREFERENCIA_ENCARGO.CURRVAL,true);
Pruebas_LinEnc.Eliminar('Prueba3',SEC_LINEAENCARGO.CURRVAL,true);

END;
/

DECLARE
IDprod NUMBER;
BEGIN
DBMS_OUTPUT.put_line('Paquete Monturas');
Pruebas_Monturas.Inicializar;
IDprod := SEC_IDREFERENCIAPRODUCTO.CURRVAL;
Pruebas_Monturas.Insertar('Prueba1',IDprod,'Metalica','Hombre',true);
Pruebas_Monturas.Actualizar_sexo('Prueba2',IDprod,'Mujer',true);
Pruebas_Monturas.Eliminar('Prueba3',IDprod,true);

END;
/

BEGIN
DBMS_OUTPUT.put_line('Paquete Ofertas');
Pruebas_Ofertas.Inicializar;
Pruebas_Ofertas.Insertar('Prueba1','Oferta de Prueba',true);
Pruebas_Ofertas.Actualizar_nom('Prueba2','Oferta de Prueba','-----',true);
Pruebas_Ofertas.Eliminar('Prueba3','-----',true);

END;
/

DECLARE
graduacion NUMBER;
ojo NUMBER;

BEGIN
DBMS_OUTPUT.put_line('Paquete Ojos');
Pruebas_Ojos.Inicializar;
graduacion:= SEC_GRADUACION.CURRVAL;
Pruebas_Ojos.Insertar('Prueba1',graduacion,'0,1','0,1','0,1','0,1','0,1','0,1','0,1','0,1','0,1','0,1',true);
ojo:= SEC_OJO.CURRVAL;
Pruebas_Ojos.Insertar('Prueba2',graduacion,'0,1','0,1','0,1','0,1','0,1','0,1','2','0,1','0,1','0,1',false);
Pruebas_Ojos.Actualizar_lejos('Prueba3',ojo,'0,2',true);

```

```

Pruebas_Ojos.Eliminar('Prueba4',ojo,true);

END;
/
BEGIN
DBMS_OUTPUT.put_line('Paquete Presentan');
Pruebas_Presentan.Inicializar;
Pruebas_Presentan.Insertar('Prueba1','Oferta de Prueba','Franquicia de Prueba',true);
Pruebas_Presentan.Actualizar_nom('Prueba2',SEC_Presentan.CURRVAL,'Optica Nombre
Nuevo',true);
Pruebas_Presentan.Eliminar('Prueba3',SEC_Presentan.CURRVAL,true);
END;
/
BEGIN
DBMS_OUTPUT.put_line('Paquete Producto');
Pruebas_Producto.Inicializar;
Pruebas_Producto.Insertar('Prueba1','a',2,'5,7','0,2',true);
Pruebas_Producto.Insertar('Prueba1','a',2,'5,7','0,2',true);
Pruebas_Producto.Actualizar_nom('Prueba2',SEC_IDREFERENCIAPRODUCTO.CURRVAL,'b',true);
Pruebas_Producto.Eliminar('Prueba3',SEC_IDREFERENCIAPRODUCTO.CURRVAL,true);
END;
/
BEGIN
DBMS_OUTPUT.put_line('Paquete SeEnvianA');
Pruebas_SeEnvianA.Inicializar;
Pruebas_SeEnvianA.Insertar('Prueba1','Oferta de Prueba',SEC_CLIENTE.CURRVAL,true);
Pruebas_SeEnvianA.Actualizar_cod('Prueba2',SEC_SeEnvianA.CURRVAL,SEC_CLIENTE.CURR
VAL-1,true);
Pruebas_SeEnvianA.Eliminar('Prueba3',SEC_SeEnvianA.CURRVAL,true);
END;
/

```

ANEXO - EJEMPLO DE FATCTURA

Nombre: [REDACTED]

Teléfono: [REDACTED] Móvil: [REDACTED]

Graduación OM

Nº Paciente: 7635

Viene por: [REDACTED]

Nº Paciente: 7635

Talón Nº: 468

Fecha: 22/05/2019

	Lejos	Cilindro	Eje	Cerca	Media	Adic.	Prisma	D.Lj	D.Cr	Fecha
D	-1,50	-0,25	165					30,0		22/05/2019
I	-1,50	-0,25	180					30,0		22/05/2019

Ficha: 5 Graduado: EM Atendido: E.MAGUILLA

Lejos	Altura	Importe
D CR-P HOY MAXXEE ORG150 HMC+ STOCK	0,0	0,00 €
I CR-P HOY MAXXEE ORG150 HMC+ STOCK	0,0	0,00 €
MOOM-PULL PBG1910		0,00 €

Talón: 468

Fecha: 22/05/2019

Fecha Entrega: 23/05/2019

Referencia	Descripción	IVA	Al	Ud	P.Unit	P.Total
2A04056	CR-P HOY MAXXEE ORG150 HMC+ STOCK	10,	2	1	0,00 €	0,00 €
2A04056	CR-P HOY MAXXEE ORG150 HMC+ STOCK	10,	2	1	0,00 €	0,00 €
1491390	MOOM-PULL PBG1910	10,	2	1	0,00 €	0,00 €

Total 0,00 €

Líquido Neto

Entregado 0,00 €

Pendiente 0,00 €

Total 0,00 €

Entregado 0,00 €

Pendiente 0,00 €

En este documento aportado por uno de los propietarios de la empresa, se nos facilita la obtención de campos necesarios a la hora de realizar el sistema, además de conocer los datos imprescindibles para la generación de una ficha de cliente y también los datos relacionados con la compra de los bienes ofrecidos por la empresa.