

**UNIVERDIDAD MARIANO GÁLVEZ DE GUATEMALA**

*Ingeniería en sistemas de la información y ciencias de la computación*

Kevin Roberto Lacán Tzul

1490-21-13789

Cécily Karina Guinea Mejía

1490-21-3264

Trabajo:Actividad 2

6to. Semestre

Sección “A”

Autómatas y lenguajes Formales

---------------------------------------------BD PUNTO DE VENTA -------------------------------------------------------------------

-- Crear tabla para Clientes

CREATE TABLE Clientes (

ClienteID NUMBER PRIMARY KEY,

Nombre VARCHAR2(100) NOT NULL,

Direccion VARCHAR2(200),

Telefono VARCHAR2(20),

Correo VARCHAR2(100),

Sexo CHAR(1) CHECK (Sexo IN ('M', 'F', 'O')),

NIT VARCHAR2(20),

CUI VARCHAR2(20),

SeguroMedico VARCHAR2(50),

NumeroPoliza VARCHAR2(50)

);

-- Crear tabla para Categorías de Productos

CREATE TABLE Categorias (

CategoriaID NUMBER PRIMARY KEY,

Nombre VARCHAR2(50) NOT NULL,

Descripcion VARCHAR2(200)

);

-- Crear tabla para Productos

CREATE TABLE Productos (

ProductoID NUMBER PRIMARY KEY,

Codigo VARCHAR2(20) UNIQUE NOT NULL,

Descripcion VARCHAR2(200) NOT NULL,

PrecioUnitario NUMBER(10,2) NOT NULL,

ImpuestoAplicable NUMBER(5,2) DEFAULT 0,

CategoriaID NUMBER,

FOREIGN KEY (CategoriaID) REFERENCES Categorias(CategoriaID)

);

-- Crear tabla para Productos Específicos (con número de serie)

CREATE TABLE ProductosEspecificos (

ProductoEspecificoID NUMBER PRIMARY KEY,

ProductoID NUMBER,

NumeroSerie VARCHAR2(50) UNIQUE NOT NULL,

EstadoVenta VARCHAR2(20) DEFAULT 'Disponible',

FOREIGN KEY (ProductoID) REFERENCES Productos(ProductoID)

);

-- Crear tabla para Formas de Pago

CREATE TABLE FormasPago (

FormaPagoID NUMBER PRIMARY KEY,

Nombre VARCHAR2(50) UNIQUE NOT NULL

);

-- Crear tabla para Ventas

CREATE TABLE Ventas (

VentaID NUMBER PRIMARY KEY,

ClienteID NUMBER,

FechaHora TIMESTAMP DEFAULT SYSTIMESTAMP,

FormaPagoID NUMBER,

NumeroFactura VARCHAR2(20) UNIQUE NOT NULL,

DescuentoTotal NUMBER(10,2) DEFAULT 0,

TotalVenta NUMBER(10,2) NOT NULL,

EstadoCredito VARCHAR2(20) DEFAULT 'Pagado',

FOREIGN KEY (ClienteID) REFERENCES Clientes(ClienteID),

FOREIGN KEY (FormaPagoID) REFERENCES FormasPago(FormaPagoID)

);

-- Crear tabla para Detalle de Ventas

CREATE TABLE DetalleVentas (

DetalleVentaID NUMBER PRIMARY KEY,

VentaID NUMBER,

ProductoID NUMBER,

ProductoEspecificoID NUMBER NULL,

Cantidad NUMBER NOT NULL,

PrecioUnitario NUMBER(10,2) NOT NULL,

Subtotal NUMBER(10,2) NOT NULL,

FOREIGN KEY (VentaID) REFERENCES Ventas(VentaID),

FOREIGN KEY (ProductoID) REFERENCES Productos(ProductoID),

FOREIGN KEY (ProductoEspecificoID) REFERENCES ProductosEspecificos(ProductoEspecificoID)

);

-- Crear tabla para Devoluciones

CREATE TABLE Devoluciones (

DevolucionID NUMBER PRIMARY KEY,

VentaID NUMBER,

FechaDevolucion DATE DEFAULT SYSDATE,

Motivo VARCHAR2(200),

FOREIGN KEY (VentaID) REFERENCES Ventas(VentaID)

);

-- Crear tabla para Detalle de Devoluciones

CREATE TABLE DetalledDevoluciones (

DetalleDevolucionID NUMBER PRIMARY KEY,

DevolucionID NUMBER,

ProductoID NUMBER,

ProductoEspecificoID NUMBER NULL,

Cantidad NUMBER NOT NULL,

FOREIGN KEY (DevolucionID) REFERENCES Devoluciones(DevolucionID),

FOREIGN KEY (ProductoID) REFERENCES Productos(ProductoID),

FOREIGN KEY (ProductoEspecificoID) REFERENCES ProductosEspecificos(ProductoEspecificoID)

);

-- Crear tabla para Bancos

CREATE TABLE Bancos (

BancoID NUMBER PRIMARY KEY,

Nombre VARCHAR2(100) UNIQUE NOT NULL

);

-- Crear tabla para Pagos

CREATE TABLE Pagos (

PagoID NUMBER PRIMARY KEY,

VentaID NUMBER,

FechaPago DATE DEFAULT SYSDATE,

FormaPagoID NUMBER,

MontoPagado NUMBER(10,2) NOT NULL,

NumeroReferencia VARCHAR2(50),

BancoID NUMBER,

FOREIGN KEY (VentaID) REFERENCES Ventas(VentaID),

FOREIGN KEY (FormaPagoID) REFERENCES FormasPago(FormaPagoID),

FOREIGN KEY (BancoID) REFERENCES Bancos(BancoID)

);

-- Crear tabla para Inventario

CREATE TABLE Inventario (

InventarioID NUMBER PRIMARY KEY,

ProductoID NUMBER,

CantidadInicial NUMBER DEFAULT 0,

CantidadVendida NUMBER DEFAULT 0,

CantidadRecibida NUMBER DEFAULT 0,

CantidadExistencia NUMBER DEFAULT 0,

FOREIGN KEY (ProductoID) REFERENCES Productos(ProductoID)

);

-- Crear secuencias para los IDs auto-incrementales

CREATE SEQUENCE seq\_cliente START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_categoria START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_producto START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_producto\_especifico START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_forma\_pago START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_venta START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_detalle\_venta START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_devolucion START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_detalle\_devolucion START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_banco START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_pago START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_inventario START WITH 1 INCREMENT BY 1;

-- Crear trigger para actualizar el inventario después de una venta

CREATE OR REPLACE TRIGGER trg\_actualizar\_inventario\_venta

AFTER INSERT ON DetalleVentas

FOR EACH ROW

BEGIN

UPDATE Inventario

SET CantidadVendida = CantidadVendida + :NEW.Cantidad,

CantidadExistencia = CantidadExistencia - :NEW.Cantidad

WHERE ProductoID = :NEW.ProductoID;

END;

/

-- Crear trigger para actualizar el inventario después de una devolución

CREATE OR REPLACE TRIGGER trg\_actualizar\_inventario\_devolucion

AFTER INSERT ON DetalledDevoluciones

FOR EACH ROW

BEGIN

UPDATE Inventario

SET CantidadVendida = CantidadVendida - :NEW.Cantidad,

CantidadExistencia = CantidadExistencia + :NEW.Cantidad

WHERE ProductoID = :NEW.ProductoID;

END;

/

-- Crear procedimiento almacenado para realizar una venta

CREATE OR REPLACE PROCEDURE realizar\_venta(

p\_cliente\_id IN NUMBER,

p\_forma\_pago\_id IN NUMBER,

p\_numero\_factura IN VARCHAR2,

p\_descuento\_total IN NUMBER,

p\_productos IN VARCHAR2,

p\_cantidades IN VARCHAR2,

p\_venta\_id OUT NUMBER

)

AS

v\_total\_venta NUMBER := 0;

v\_producto\_id NUMBER;

v\_cantidad NUMBER;

v\_precio\_unitario NUMBER;

v\_subtotal NUMBER;

v\_producto\_especifico\_id NUMBER;

BEGIN

-- Insertar la venta

INSERT INTO Ventas (VentaID, ClienteID, FormaPagoID, NumeroFactura, DescuentoTotal, TotalVenta)

VALUES (seq\_venta.NEXTVAL, p\_cliente\_id, p\_forma\_pago\_id, p\_numero\_factura, p\_descuento\_total, 0)

RETURNING VentaID INTO p\_venta\_id;

-- Procesar cada producto

FOR i IN 1..REGEXP\_COUNT(p\_productos, ',') + 1 LOOP

v\_producto\_id := TO\_NUMBER(REGEXP\_SUBSTR(p\_productos, '[^,]+', 1, i));

v\_cantidad := TO\_NUMBER(REGEXP\_SUBSTR(p\_cantidades, '[^,]+', 1, i));

-- Obtener el precio unitario del producto

SELECT PrecioUnitario INTO v\_precio\_unitario

FROM Productos

WHERE ProductoID = v\_producto\_id;

v\_subtotal := v\_precio\_unitario \* v\_cantidad;

v\_total\_venta := v\_total\_venta + v\_subtotal;

-- Verificar si es un producto específico

BEGIN

SELECT ProductoEspecificoID INTO v\_producto\_especifico\_id

FROM ProductosEspecificos

WHERE ProductoID = v\_producto\_id AND EstadoVenta = 'Disponible'

AND ROWNUM = 1;

-- Actualizar el estado del producto específico

UPDATE ProductosEspecificos

SET EstadoVenta = 'Vendido'

WHERE ProductoEspecificoID = v\_producto\_especifico\_id;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

v\_producto\_especifico\_id := NULL;

END;

-- Insertar el detalle de la venta

INSERT INTO DetalleVentas (DetalleVentaID, VentaID, ProductoID, ProductoEspecificoID, Cantidad, PrecioUnitario, Subtotal)

VALUES (seq\_detalle\_venta.NEXTVAL, p\_venta\_id, v\_producto\_id, v\_producto\_especifico\_id, v\_cantidad, v\_precio\_unitario, v\_subtotal);

END LOOP;

-- Actualizar el total de la venta

UPDATE Ventas

SET TotalVenta = v\_total\_venta - p\_descuento\_total

WHERE VentaID = p\_venta\_id;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

RAISE;

END;

/

-- Crear función para calcular el total de ventas por período

CREATE OR REPLACE FUNCTION calcular\_total\_ventas(

p\_fecha\_inicio IN DATE,

p\_fecha\_fin IN DATE

) RETURN NUMBER

AS

v\_total\_ventas NUMBER;

BEGIN

SELECT SUM(TotalVenta)

INTO v\_total\_ventas

FROM Ventas

WHERE FechaHora BETWEEN p\_fecha\_inicio AND p\_fecha\_fin;

RETURN NVL(v\_total\_ventas, 0);

END;

/

-- Crear job para realizar copias de seguridad diarias

BEGIN

DBMS\_SCHEDULER.CREATE\_JOB (

job\_name => 'BACKUP\_DIARIO',

job\_type => 'PLSQL\_BLOCK',

job\_action => 'BEGIN DBMS\_BACKUP\_RESTORE.BACKUP(''FULL''); END;',

start\_date => SYSTIMESTAMP,

repeat\_interval => 'FREQ=DAILY; BYHOUR=2',

enabled => TRUE

);

END;

/

-- Insertar datos en la tabla Clientes

INSERT INTO Clientes (ClienteID, Nombre, Direccion, Telefono, Correo, Sexo, NIT, CUI, SeguroMedico, NumeroPoliza)

VALUES (seq\_cliente.NEXTVAL, 'Juan Pérez', 'Calle 1-23 Zona 1', '5555-1234', 'juan@email.com', 'M', '1234567-8', '1234567890101', 'Seguro A', 'POL-001');

INSERT INTO Clientes (ClienteID, Nombre, Direccion, Telefono, Correo, Sexo, NIT, CUI, SeguroMedico, NumeroPoliza)

VALUES (seq\_cliente.NEXTVAL, 'María García', 'Avenida 2-34 Zona 2', '5555-5678', 'maria@email.com', 'F', '2345678-9', '2345678901012', 'Seguro B', 'POL-002');

-- Insertar datos en la tabla Categorias

INSERT INTO Categorias (CategoriaID, Nombre, Descripcion)

VALUES (seq\_categoria.NEXTVAL, 'Electrónicos', 'Productos electrónicos y gadgets');

INSERT INTO Categorias (CategoriaID, Nombre, Descripcion)

VALUES (seq\_categoria.NEXTVAL, 'Ropa', 'Prendas de vestir y accesorios');

-- Insertar datos en la tabla Productos

INSERT INTO Productos (ProductoID, Codigo, Descripcion, PrecioUnitario, ImpuestoAplicable, CategoriaID)

VALUES (seq\_producto.NEXTVAL, 'PROD001', 'Smartphone XYZ', 5000, 12, 1);

INSERT INTO Productos (ProductoID, Codigo, Descripcion, PrecioUnitario, ImpuestoAplicable, CategoriaID)

VALUES (seq\_producto.NEXTVAL, 'PROD002', 'Camiseta Algodón', 150, 12, 2);

-- Insertar datos en la tabla ProductosEspecificos

INSERT INTO ProductosEspecificos (ProductoEspecificoID, ProductoID, NumeroSerie, EstadoVenta)

VALUES (seq\_producto\_especifico.NEXTVAL, 1, 'SER001XYZ', 'Disponible');

INSERT INTO ProductosEspecificos (ProductoEspecificoID, ProductoID, NumeroSerie, EstadoVenta)

VALUES (seq\_producto\_especifico.NEXTVAL, 1, 'SER002XYZ', 'Disponible');

-- Insertar datos en la tabla FormasPago

INSERT INTO FormasPago (FormaPagoID, Nombre)

VALUES (seq\_forma\_pago.NEXTVAL, 'Efectivo');

INSERT INTO FormasPago (FormaPagoID, Nombre)

VALUES (seq\_forma\_pago.NEXTVAL, 'Tarjeta de Crédito');

-- Insertar datos en la tabla Bancos

INSERT INTO Bancos (BancoID, Nombre)

VALUES (seq\_banco.NEXTVAL, 'Banco Nacional');

INSERT INTO Bancos (BancoID, Nombre)

VALUES (seq\_banco.NEXTVAL, 'Banco Internacional');

-- Insertar datos en la tabla Inventario

INSERT INTO Inventario (InventarioID, ProductoID, CantidadInicial, CantidadVendida, CantidadRecibida, CantidadExistencia)

VALUES (seq\_inventario.NEXTVAL, 1, 100, 0, 100, 100);

INSERT INTO Inventario (InventarioID, ProductoID, CantidadInicial, CantidadVendida, CantidadRecibida, CantidadExistencia)

VALUES (seq\_inventario.NEXTVAL, 2, 200, 0, 200, 200);

-- Realizar una venta

DECLARE

v\_venta\_id NUMBER;

BEGIN

realizar\_venta(

p\_cliente\_id => 1,

p\_forma\_pago\_id => 1,

p\_numero\_factura => 'FAC001',

p\_descuento\_total => 0,

p\_productos => '1,2',

p\_cantidades => '1,2',

p\_venta\_id => v\_venta\_id

);

END;

/

-- Insertar una devolución

DECLARE

v\_venta\_id NUMBER;

v\_devolucion\_id NUMBER;

BEGIN

-- Obtener el ID de la última venta

SELECT MAX(VentaID) INTO v\_venta\_id FROM Ventas;

-- Insertar la devolución

INSERT INTO Devoluciones (DevolucionID, VentaID, FechaDevolucion, Motivo)

VALUES (seq\_devolucion.NEXTVAL, v\_venta\_id, SYSDATE, 'Producto defectuoso')

RETURNING DevolucionID INTO v\_devolucion\_id;

-- Insertar el detalle de la devolución

INSERT INTO DetalledDevoluciones (DetalleDevolucionID, DevolucionID, ProductoID, ProductoEspecificoID, Cantidad)

VALUES (seq\_detalle\_devolucion.NEXTVAL, v\_devolucion\_id, 1, NULL, 1);

-- Actualizar el inventario

UPDATE Inventario

SET CantidadVendida = CantidadVendida - 1,

CantidadExistencia = CantidadExistencia + 1

WHERE ProductoID = 1;

COMMIT;

END;

/

-- Insertar un pago

INSERT INTO Pagos (PagoID, VentaID, FechaPago, FormaPagoID, MontoPagado, NumeroReferencia, BancoID)

VALUES (seq\_pago.NEXTVAL, 1, SYSDATE, 2, 5300, 'REF001', 1);

COMMIT;

-- 1. Producto más vendido (por cantidad)

SELECT

p.ProductoID,

p.Codigo,

p.Descripcion,

SUM(dv.Cantidad) AS TotalVendido

FROM

Productos p

JOIN DetalleVentas dv ON p.ProductoID = dv.ProductoID

GROUP BY

p.ProductoID, p.Codigo, p.Descripcion

ORDER BY

TotalVendido DESC

FETCH FIRST 1 ROW ONLY;

-- 2. Producto más vendido (por ingresos)

SELECT

p.ProductoID,

p.Codigo,

p.Descripcion,

SUM(dv.Subtotal) AS TotalIngresos

FROM

Productos p

JOIN DetalleVentas dv ON p.ProductoID = dv.ProductoID

GROUP BY

p.ProductoID, p.Codigo, p.Descripcion

ORDER BY

TotalIngresos DESC

FETCH FIRST 1 ROW ONLY;

-- 3. Día con más ventas

SELECT

TRUNC(v.FechaHora) AS Fecha,

COUNT(\*) AS NumeroVentas,

SUM(v.TotalVenta) AS TotalVentas

FROM

Ventas v

GROUP BY

TRUNC(v.FechaHora)

ORDER BY

TotalVentas DESC

FETCH FIRST 1 ROW ONLY;

-- 4. Cliente más frecuente

SELECT

c.ClienteID,

c.Nombre,

COUNT(\*) AS NumeroCompras,

SUM(v.TotalVenta) AS TotalGastado

FROM

Clientes c

JOIN Ventas v ON c.ClienteID = v.ClienteID

GROUP BY

c.ClienteID, c.Nombre

ORDER BY

NumeroCompras DESC

FETCH FIRST 1 ROW ONLY;

-- 5. Ventas por categoría de producto

SELECT

c.Nombre AS Categoria,

COUNT(\*) AS NumeroVentas,

SUM(dv.Subtotal) AS TotalVentas

FROM

Categorias c

JOIN Productos p ON c.CategoriaID = p.CategoriaID

JOIN DetalleVentas dv ON p.ProductoID = dv.ProductoID

GROUP BY

c.Nombre

ORDER BY

TotalVentas DESC;

-- 6. Tendencia de ventas mensuales

SELECT

TO\_CHAR(v.FechaHora, 'YYYY-MM') AS Mes,

COUNT(\*) AS NumeroVentas,

SUM(v.TotalVenta) AS TotalVentas

FROM

Ventas v

GROUP BY

TO\_CHAR(v.FechaHora, 'YYYY-MM')

ORDER BY

Mes;

-- 7. Productos con bajo stock (menos de 10 unidades)

SELECT

p.ProductoID,

p.Codigo,

p.Descripcion,

i.CantidadExistencia

FROM

Productos p

JOIN Inventario i ON p.ProductoID = i.ProductoID

WHERE

i.CantidadExistencia < 10

ORDER BY

i.CantidadExistencia;

-- 8. Top 5 productos más rentables

SELECT

p.ProductoID,

p.Codigo,

p.Descripcion,

SUM(dv.Subtotal - (dv.Cantidad \* p.PrecioUnitario)) AS Utilidad

FROM

Productos p

JOIN DetalleVentas dv ON p.ProductoID = dv.ProductoID

GROUP BY

p.ProductoID, p.Codigo, p.Descripcion

ORDER BY

Utilidad DESC

FETCH FIRST 5 ROWS ONLY;

-- 9. Efectividad de descuentos

SELECT

CASE

WHEN v.DescuentoTotal > 0 THEN 'Con Descuento'

ELSE 'Sin Descuento'

END AS TipoVenta,

COUNT(\*) AS NumeroVentas,

AVG(v.TotalVenta) AS PromedioVenta

FROM

Ventas v

GROUP BY

CASE

WHEN v.DescuentoTotal > 0 THEN 'Con Descuento'

ELSE 'Sin Descuento'

END;

-- 10. Análisis de devoluciones

SELECT

p.ProductoID,

p.Codigo,

p.Descripcion,

COUNT(\*) AS NumeroDevoluciones,

SUM(dd.Cantidad) AS CantidadDevuelta

FROM

Productos p

JOIN DetalledDevoluciones dd ON p.ProductoID = dd.ProductoID

JOIN Devoluciones d ON dd.DevolucionID = d.DevolucionID

GROUP BY

p.ProductoID, p.Codigo, p.Descripcion

ORDER BY

NumeroDevoluciones DESC;

-- 11. Ventas por forma de pago

SELECT

fp.Nombre AS FormaPago,

COUNT(\*) AS NumeroVentas,

SUM(v.TotalVenta) AS TotalVentas

FROM

FormasPago fp

JOIN Ventas v ON fp.FormaPagoID = v.FormaPagoID

GROUP BY

fp.Nombre

ORDER BY

TotalVentas DESC;

-- 12. Clientes que no han comprado en los últimos 3 meses

SELECT

c.ClienteID,

c.Nombre,

MAX(v.FechaHora) AS UltimaCompra

FROM

Clientes c

LEFT JOIN Ventas v ON c.ClienteID = v.ClienteID

GROUP BY

c.ClienteID, c.Nombre

HAVING

MAX(v.FechaHora) < ADD\_MONTHS(SYSDATE, -3) OR MAX(v.FechaHora) IS NULL

ORDER BY

UltimaCompra;

-----------------------------------------BD CONTABILIDAD---------------------------------------

-- Crear tabla para Clientes

CREATE TABLE C\_Clientes (

ClienteID NUMBER PRIMARY KEY,

Nombre VARCHAR2(100) NOT NULL,

Direccion VARCHAR2(200),

RFC VARCHAR2(20),

CorreoElectronico VARCHAR2(100),

Telefono VARCHAR2(20)

);

-- Crear tabla para Categorias de Productos

CREATE TABLE C\_Categorias (

CategoriaID NUMBER PRIMARY KEY,

Nombre VARCHAR2(50) NOT NULL,

Descripcion VARCHAR2(200)

);

-- Crear tabla para Proveedores

CREATE TABLE C\_Proveedores (

ProveedorID NUMBER PRIMARY KEY,

Nombre VARCHAR2(100) NOT NULL,

Direccion VARCHAR2(200),

RFC VARCHAR2(20),

CorreoElectronico VARCHAR2(100),

Telefono VARCHAR2(20)

);

-- Crear tabla para Productos

CREATE TABLE C\_Productos (

ProductoID NUMBER PRIMARY KEY,

Codigo VARCHAR2(20) UNIQUE NOT NULL,

Descripcion VARCHAR2(200) NOT NULL,

PrecioCompra NUMBER(10,2) NOT NULL,

PrecioVenta NUMBER(10,2) NOT NULL,

ImpuestoAplicable NUMBER(5,2) DEFAULT 0,

CategoriaID NUMBER,

ProveedorID NUMBER,

StockMinimo NUMBER,

StockMaximo NUMBER,

FOREIGN KEY (CategoriaID) REFERENCES C\_Categorias(CategoriaID),

FOREIGN KEY (ProveedorID) REFERENCES C\_Proveedores(ProveedorID)

);

-- Crear tabla para Ventas

CREATE TABLE C\_Ventas (

VentaID NUMBER PRIMARY KEY,

ClienteID NUMBER,

FechaHora TIMESTAMP DEFAULT SYSTIMESTAMP,

NumeroFactura VARCHAR2(20) UNIQUE NOT NULL,

FormaPago VARCHAR2(50),

DescuentoTotal NUMBER(10,2) DEFAULT 0,

ImpuestoTotal NUMBER(10,2) DEFAULT 0,

TotalVenta NUMBER(10,2) NOT NULL,

FOREIGN KEY (ClienteID) REFERENCES C\_Clientes(ClienteID)

);

-- Crear tabla para Detalle de Ventas

CREATE TABLE C\_DetalleVentas (

DetalleVentaID NUMBER PRIMARY KEY,

VentaID NUMBER,

ProductoID NUMBER,

Cantidad NUMBER NOT NULL,

PrecioUnitario NUMBER(10,2) NOT NULL,

Subtotal NUMBER(10,2) NOT NULL,

FOREIGN KEY (VentaID) REFERENCES C\_Ventas(VentaID),

FOREIGN KEY (ProductoID) REFERENCES C\_Productos(ProductoID)

);

-- Crear tabla para Compras

CREATE TABLE C\_Compras (

CompraID NUMBER PRIMARY KEY,

ProveedorID NUMBER,

FechaCompra DATE DEFAULT SYSDATE,

NumeroFactura VARCHAR2(20) UNIQUE NOT NULL,

FormaPago VARCHAR2(50),

ImpuestoTotal NUMBER(10,2) DEFAULT 0,

TotalCompra NUMBER(10,2) NOT NULL,

FOREIGN KEY (ProveedorID) REFERENCES C\_Proveedores(ProveedorID)

);

-- Crear tabla para Detalle de Compras

CREATE TABLE C\_DetalleCompras (

DetalleCompraID NUMBER PRIMARY KEY,

CompraID NUMBER,

ProductoID NUMBER,

Cantidad NUMBER NOT NULL,

PrecioUnitario NUMBER(10,2) NOT NULL,

Subtotal NUMBER(10,2) NOT NULL,

FOREIGN KEY (CompraID) REFERENCES C\_Compras(CompraID),

FOREIGN KEY (ProductoID) REFERENCES C\_Productos(ProductoID)

);

-- Crear tabla para Gastos

CREATE TABLE C\_Gastos (

GastoID NUMBER PRIMARY KEY,

Concepto VARCHAR2(100) NOT NULL,

Monto NUMBER(10,2) NOT NULL,

FechaGasto DATE DEFAULT SYSDATE,

Categoria VARCHAR2(50)

);

-- Crear tabla para Bancos

CREATE TABLE C\_Bancos (

BancoID NUMBER PRIMARY KEY,

Nombre VARCHAR2(100) NOT NULL

);

-- Crear tabla para Cuentas Bancarias

CREATE TABLE C\_CuentasBancarias (

CuentaBancariaID NUMBER PRIMARY KEY,

BancoID NUMBER,

NumeroCuenta VARCHAR2(50) UNIQUE NOT NULL,

Saldo NUMBER(10,2) DEFAULT 0,

FOREIGN KEY (BancoID) REFERENCES C\_Bancos(BancoID)

);

-- Crear tabla para Movimientos Bancarios

CREATE TABLE C\_MovimientosBancarios (

MovimientoBancarioID NUMBER PRIMARY KEY,

CuentaBancariaID NUMBER,

FechaMovimiento DATE DEFAULT SYSDATE,

Tipo VARCHAR2(20) CHECK (Tipo IN ('Ingreso', 'Egreso')),

Monto NUMBER(10,2) NOT NULL,

Concepto VARCHAR2(200),

FOREIGN KEY (CuentaBancariaID) REFERENCES C\_CuentasBancarias(CuentaBancariaID)

);

-- Crear tabla para Inventario

CREATE TABLE C\_Inventario (

InventarioID NUMBER PRIMARY KEY,

ProductoID NUMBER,

CantidadInicial NUMBER DEFAULT 0,

CantidadVendida NUMBER DEFAULT 0,

CantidadComprada NUMBER DEFAULT 0,

CantidadExistencia NUMBER DEFAULT 0,

ValorInventario NUMBER(10,2) DEFAULT 0,

FOREIGN KEY (ProductoID) REFERENCES C\_Productos(ProductoID)

);

-- Crear tabla para Cuentas por Cobrar

CREATE TABLE C\_CuentasPorCobrar (

CuentaPorCobrarID NUMBER PRIMARY KEY,

ClienteID NUMBER,

VentaID NUMBER,

MontoTotal NUMBER(10,2) NOT NULL,

MontoPendiente NUMBER(10,2) NOT NULL,

FechaVencimiento DATE,

FOREIGN KEY (ClienteID) REFERENCES C\_Clientes(ClienteID),

FOREIGN KEY (VentaID) REFERENCES C\_Ventas(VentaID)

);

-- Crear tabla para Cuentas por Pagar

CREATE TABLE C\_CuentasPorPagar (

CuentaPorPagarID NUMBER PRIMARY KEY,

ProveedorID NUMBER,

CompraID NUMBER,

MontoTotal NUMBER(10,2) NOT NULL,

MontoPendiente NUMBER(10,2) NOT NULL,

FechaVencimiento DATE,

FOREIGN KEY (ProveedorID) REFERENCES C\_Proveedores(ProveedorID),

FOREIGN KEY (CompraID) REFERENCES C\_Compras(CompraID)

);

-- Crear tabla para Notas de Débito

CREATE TABLE C\_NotasDebito (

NotaDebitoID NUMBER PRIMARY KEY,

ClienteID NUMBER,

VentaID NUMBER,

FechaEmision DATE DEFAULT SYSDATE,

Monto NUMBER(10,2) NOT NULL,

Motivo VARCHAR2(200),

FOREIGN KEY (ClienteID) REFERENCES C\_Clientes(ClienteID),

FOREIGN KEY (VentaID) REFERENCES C\_Ventas(VentaID)

);

-- Crear tabla para Notas de Crédito

CREATE TABLE C\_NotasCredito (

NotaCreditoID NUMBER PRIMARY KEY,

ClienteID NUMBER,

VentaID NUMBER,

FechaEmision DATE DEFAULT SYSDATE,

Monto NUMBER(10,2) NOT NULL,

Motivo VARCHAR2(200),

FOREIGN KEY (ClienteID) REFERENCES C\_Clientes(ClienteID),

FOREIGN KEY (VentaID) REFERENCES C\_Ventas(VentaID)

);

-- Crear secuencias para los IDs auto-incrementales

CREATE SEQUENCE seq\_c\_cliente START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_categoria START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_proveedor START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_producto START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_venta START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_detalle\_venta START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_compra START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_detalle\_compra START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_gasto START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_banco START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_cuenta\_bancaria START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_movimiento\_bancario START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_inventario START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_cuenta\_por\_cobrar START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_cuenta\_por\_pagar START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_nota\_debito START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_nota\_credito START WITH 1 INCREMENT BY 1;

-- Crear trigger para actualizar el inventario después de una venta

CREATE OR REPLACE TRIGGER trg\_actualizar\_c\_inventario\_venta

AFTER INSERT ON C\_DetalleVentas

FOR EACH ROW

BEGIN

UPDATE C\_Inventario

SET CantidadVendida = CantidadVendida + :NEW.Cantidad,

CantidadExistencia = CantidadExistencia - :NEW.Cantidad,

ValorInventario = (CantidadExistencia - :NEW.Cantidad) \*

(SELECT PrecioCompra FROM C\_Productos WHERE ProductoID = :NEW.ProductoID)

WHERE ProductoID = :NEW.ProductoID;

END;

/

CREATE OR REPLACE TRIGGER trg\_actualizar\_c\_inventario\_compra

AFTER INSERT ON C\_DetalleCompras

FOR EACH ROW

BEGIN

-- Verificar si el producto ya existe en el inventario

IF EXISTS (SELECT 1 FROM C\_Inventario WHERE ProductoID = :NEW.ProductoID) THEN

-- Actualizar el inventario existente

UPDATE C\_Inventario

SET CantidadComprada = CantidadComprada + :NEW.Cantidad,

CantidadExistencia = CantidadExistencia + :NEW.Cantidad,

ValorInventario = (CantidadExistencia + :NEW.Cantidad) \*

(SELECT PrecioCompra FROM C\_Productos WHERE ProductoID = :NEW.ProductoID)

WHERE ProductoID = :NEW.ProductoID;

ELSE

-- Si el producto no está en inventario, insertarlo

INSERT INTO C\_Inventario (InventarioID, ProductoID, CantidadInicial, CantidadComprada, CantidadExistencia, ValorInventario)

VALUES (seq\_c\_inventario.NEXTVAL, :NEW.ProductoID, :NEW.Cantidad, :NEW.Cantidad, :NEW.Cantidad,

:NEW.Cantidad \* (SELECT PrecioCompra FROM C\_Productos WHERE ProductoID = :NEW.ProductoID));

END IF;

END;

/

-- Crear tabla para Clientes

CREATE TABLE C\_Clientes (

ClienteID NUMBER PRIMARY KEY,

Nombre VARCHAR2(100) NOT NULL,

Direccion VARCHAR2(200),

RFC VARCHAR2(20),

CorreoElectronico VARCHAR2(100),

Telefono VARCHAR2(20)

);

-- Crear tabla para Categorias de Productos

CREATE TABLE C\_Categorias (

CategoriaID NUMBER PRIMARY KEY,

Nombre VARCHAR2(50) NOT NULL,

Descripcion VARCHAR2(200)

);

-- Crear tabla para Proveedores

CREATE TABLE C\_Proveedores (

ProveedorID NUMBER PRIMARY KEY,

Nombre VARCHAR2(100) NOT NULL,

Direccion VARCHAR2(200),

RFC VARCHAR2(20),

CorreoElectronico VARCHAR2(100),

Telefono VARCHAR2(20)

);

-- Crear tabla para Productos

CREATE TABLE C\_Productos (

ProductoID NUMBER PRIMARY KEY,

Codigo VARCHAR2(20) UNIQUE NOT NULL,

Descripcion VARCHAR2(200) NOT NULL,

PrecioCompra NUMBER(10,2) NOT NULL,

PrecioVenta NUMBER(10,2) NOT NULL,

ImpuestoAplicable NUMBER(5,2) DEFAULT 0,

CategoriaID NUMBER,

ProveedorID NUMBER,

StockMinimo NUMBER,

StockMaximo NUMBER,

FOREIGN KEY (CategoriaID) REFERENCES C\_Categorias(CategoriaID),

FOREIGN KEY (ProveedorID) REFERENCES C\_Proveedores(ProveedorID)

);

-- Crear tabla para Ventas

CREATE TABLE C\_Ventas (

VentaID NUMBER PRIMARY KEY,

ClienteID NUMBER,

FechaHora TIMESTAMP DEFAULT SYSTIMESTAMP,

NumeroFactura VARCHAR2(20) UNIQUE NOT NULL,

FormaPago VARCHAR2(50),

DescuentoTotal NUMBER(10,2) DEFAULT 0,

ImpuestoTotal NUMBER(10,2) DEFAULT 0,

TotalVenta NUMBER(10,2) NOT NULL,

FOREIGN KEY (ClienteID) REFERENCES C\_Clientes(ClienteID)

);

-- Crear tabla para Detalle de Ventas

CREATE TABLE C\_DetalleVentas (

DetalleVentaID NUMBER PRIMARY KEY,

VentaID NUMBER,

ProductoID NUMBER,

Cantidad NUMBER NOT NULL,

PrecioUnitario NUMBER(10,2) NOT NULL,

Subtotal NUMBER(10,2) NOT NULL,

FOREIGN KEY (VentaID) REFERENCES C\_Ventas(VentaID),

FOREIGN KEY (ProductoID) REFERENCES C\_Productos(ProductoID)

);

-- Crear tabla para Compras

CREATE TABLE C\_Compras (

CompraID NUMBER PRIMARY KEY,

ProveedorID NUMBER,

FechaCompra DATE DEFAULT SYSDATE,

NumeroFactura VARCHAR2(20) UNIQUE NOT NULL,

FormaPago VARCHAR2(50),

ImpuestoTotal NUMBER(10,2) DEFAULT 0,

TotalCompra NUMBER(10,2) NOT NULL,

FOREIGN KEY (ProveedorID) REFERENCES C\_Proveedores(ProveedorID)

);

-- Crear tabla para Detalle de Compras

CREATE TABLE C\_DetalleCompras (

DetalleCompraID NUMBER PRIMARY KEY,

CompraID NUMBER,

ProductoID NUMBER,

Cantidad NUMBER NOT NULL,

PrecioUnitario NUMBER(10,2) NOT NULL,

Subtotal NUMBER(10,2) NOT NULL,

FOREIGN KEY (CompraID) REFERENCES C\_Compras(CompraID),

FOREIGN KEY (ProductoID) REFERENCES C\_Productos(ProductoID)

);

-- Crear tabla para Gastos

CREATE TABLE C\_Gastos (

GastoID NUMBER PRIMARY KEY,

Concepto VARCHAR2(100) NOT NULL,

Monto NUMBER(10,2) NOT NULL,

FechaGasto DATE DEFAULT SYSDATE,

Categoria VARCHAR2(50)

);

-- Crear tabla para Bancos

CREATE TABLE C\_Bancos (

BancoID NUMBER PRIMARY KEY,

Nombre VARCHAR2(100) NOT NULL

);

-- Crear tabla para Cuentas Bancarias

CREATE TABLE C\_CuentasBancarias (

CuentaBancariaID NUMBER PRIMARY KEY,

BancoID NUMBER,

NumeroCuenta VARCHAR2(50) UNIQUE NOT NULL,

Saldo NUMBER(10,2) DEFAULT 0,

FOREIGN KEY (BancoID) REFERENCES C\_Bancos(BancoID)

);

-- Crear tabla para Movimientos Bancarios

CREATE TABLE C\_MovimientosBancarios (

MovimientoBancarioID NUMBER PRIMARY KEY,

CuentaBancariaID NUMBER,

FechaMovimiento DATE DEFAULT SYSDATE,

Tipo VARCHAR2(20) CHECK (Tipo IN ('Ingreso', 'Egreso')),

Monto NUMBER(10,2) NOT NULL,

Concepto VARCHAR2(200),

FOREIGN KEY (CuentaBancariaID) REFERENCES C\_CuentasBancarias(CuentaBancariaID)

);

-- Crear tabla para Inventario

CREATE TABLE C\_Inventario (

InventarioID NUMBER PRIMARY KEY,

ProductoID NUMBER,

CantidadInicial NUMBER DEFAULT 0,

CantidadVendida NUMBER DEFAULT 0,

CantidadComprada NUMBER DEFAULT 0,

CantidadExistencia NUMBER DEFAULT 0,

ValorInventario NUMBER(10,2) DEFAULT 0,

FOREIGN KEY (ProductoID) REFERENCES C\_Productos(ProductoID)

);

-- Crear tabla para Cuentas por Cobrar

CREATE TABLE C\_CuentasPorCobrar (

CuentaPorCobrarID NUMBER PRIMARY KEY,

ClienteID NUMBER,

VentaID NUMBER,

MontoTotal NUMBER(10,2) NOT NULL,

MontoPendiente NUMBER(10,2) NOT NULL,

FechaVencimiento DATE,

FOREIGN KEY (ClienteID) REFERENCES C\_Clientes(ClienteID),

FOREIGN KEY (VentaID) REFERENCES C\_Ventas(VentaID)

);

-- Crear tabla para Cuentas por Pagar

CREATE TABLE C\_CuentasPorPagar (

CuentaPorPagarID NUMBER PRIMARY KEY,

ProveedorID NUMBER,

CompraID NUMBER,

MontoTotal NUMBER(10,2) NOT NULL,

MontoPendiente NUMBER(10,2) NOT NULL,

FechaVencimiento DATE,

FOREIGN KEY (ProveedorID) REFERENCES C\_Proveedores(ProveedorID),

FOREIGN KEY (CompraID) REFERENCES C\_Compras(CompraID)

);

-- Crear tabla para Notas de Débito

CREATE TABLE C\_NotasDebito (

NotaDebitoID NUMBER PRIMARY KEY,

ClienteID NUMBER,

VentaID NUMBER,

FechaEmision DATE DEFAULT SYSDATE,

Monto NUMBER(10,2) NOT NULL,

Motivo VARCHAR2(200),

FOREIGN KEY (ClienteID) REFERENCES C\_Clientes(ClienteID),

FOREIGN KEY (VentaID) REFERENCES C\_Ventas(VentaID)

);

-- Crear tabla para Notas de Crédito

CREATE TABLE C\_NotasCredito (

NotaCreditoID NUMBER PRIMARY KEY,

ClienteID NUMBER,

VentaID NUMBER,

FechaEmision DATE DEFAULT SYSDATE,

Monto NUMBER(10,2) NOT NULL,

Motivo VARCHAR2(200),

FOREIGN KEY (ClienteID) REFERENCES C\_Clientes(ClienteID),

FOREIGN KEY (VentaID) REFERENCES C\_Ventas(VentaID)

);

-- Crear secuencias para los IDs auto-incrementales

CREATE SEQUENCE seq\_c\_cliente START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_categoria START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_proveedor START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_producto START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_venta START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_detalle\_venta START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_compra START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_detalle\_compra START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_gasto START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_banco START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_cuenta\_bancaria START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_movimiento\_bancario START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_inventario START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_cuenta\_por\_cobrar START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_cuenta\_por\_pagar START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_nota\_debito START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq\_c\_nota\_credito START WITH 1 INCREMENT BY 1;

-- Crear trigger para actualizar el inventario después de una venta

CREATE OR REPLACE TRIGGER trg\_actualizar\_c\_inventario\_venta

AFTER INSERT ON C\_DetalleVentas

FOR EACH ROW

BEGIN

UPDATE C\_Inventario

SET CantidadVendida = CantidadVendida + :NEW.Cantidad,

CantidadExistencia = CantidadExistencia - :NEW.Cantidad,

ValorInventario = (CantidadExistencia - :NEW.Cantidad) \*

(SELECT PrecioCompra FROM C\_Productos WHERE ProductoID = :NEW.ProductoID)

WHERE ProductoID = :NEW.ProductoID;

END;

/

-- Crear trigger para actualizar el inventario después de una compra

CREATE OR REPLACE TRIGGER trg\_actualizar\_c\_inventario\_compra

AFTER INSERT ON C\_DetalleCompras

FOR EACH ROW

BEGIN

UPDATE C\_Inventario

SET CantidadComprada = CantidadComprada + :NEW.Cantidad,

CantidadExistencia = CantidadExistencia + :NEW.Cantidad,

ValorInventario = (CantidadExistencia + :NEW.Cantidad) \*

(SELECT PrecioCompra FROM C\_Productos WHERE ProductoID = :NEW.ProductoID)

WHERE ProductoID = :NEW.ProductoID;

END;

/

------------------------------Vista por Clientees-----------------------------

SELECT

C.ClienteID,

C.Nombre,

COUNT(V.VentaID) AS TotalVentas,

SUM(V.TotalVenta) AS MontoTotalVendido

FROM C\_Clientes C

JOIN C\_Ventas V ON C.ClienteID = V.ClienteID

GROUP BY C.ClienteID, C.Nombre;

----------------------------Vista por productos-------------------------------

SELECT

P.ProductoID,

P.Descripcion,

P.PrecioVenta,

I.CantidadExistencia,

I.ValorInventario

FROM C\_Productos P

JOIN C\_Inventario I ON P.ProductoID = I.ProductoID;

-------------------------Vista de cuentas por cobrar-------------------------

SELECT

C.ClienteID,

C.Nombre,

CC.VentaID,

CC.MontoTotal,

CC.MontoPendiente,

CC.FechaVencimiento

FROM C\_Clientes C

JOIN C\_CuentasPorCobrar CC ON C.ClienteID = CC.ClienteID

WHERE CC.MontoPendiente > 0;

-------------------------Vista de cuentas por pagar-------------------------

SELECT

P.ProveedorID,

P.Nombre,

CP.CompraID,

CP.MontoTotal,

CP.MontoPendiente,

CP.FechaVencimiento

FROM C\_Proveedores P

JOIN C\_CuentasPorPagar CP ON P.ProveedorID = CP.ProveedorID

WHERE CP.MontoPendiente > 0;

-------------------------Nueva venta -------------------------

CREATE OR REPLACE PROCEDURE RegistrarVenta(

p\_ClienteID NUMBER,

p\_Productos C\_DetalleVentas%ROWTYPE,

p\_DescuentoTotal NUMBER,

p\_ImpuestoTotal NUMBER,

p\_FormaPago VARCHAR2

) IS

v\_TotalVenta NUMBER := 0;

v\_VentaID NUMBER;

BEGIN

-- Insertar la venta en C\_Ventas

INSERT INTO C\_Ventas (VentaID, ClienteID, FechaHora, NumeroFactura, FormaPago, DescuentoTotal, ImpuestoTotal, TotalVenta)

VALUES (seq\_c\_venta.NEXTVAL, p\_ClienteID, SYSTIMESTAMP, 'FAC-' || seq\_c\_venta.CURRVAL, p\_FormaPago, p\_DescuentoTotal, p\_ImpuestoTotal, v\_TotalVenta)

RETURNING VentaID INTO v\_VentaID;

-- Insertar detalles de la venta

FOR i IN p\_Productos.FIRST .. p\_Productos.LAST LOOP

INSERT INTO C\_DetalleVentas (DetalleVentaID, VentaID, ProductoID, Cantidad, PrecioUnitario, Subtotal)

VALUES (seq\_c\_detalle\_venta.NEXTVAL, v\_VentaID, p\_Productos(i).ProductoID, p\_Productos(i).Cantidad, p\_Productos(i).PrecioUnitario, p\_Productos(i).Subtotal);

-- Actualizar el total de la venta

v\_TotalVenta := v\_TotalVenta + p\_Productos(i).Subtotal;

END LOOP;

-- Actualizar el total de la venta en C\_Ventas

UPDATE C\_Ventas

SET TotalVenta = v\_TotalVenta

WHERE VentaID = v\_VentaID;

COMMIT;

END;

/

-------------------------Nueva compra -------------------------

CREATE OR REPLACE PROCEDURE RegistrarCompra(

p\_ProveedorID NUMBER,

p\_Productos C\_DetalleCompras%ROWTYPE,

p\_ImpuestoTotal NUMBER,

p\_FormaPago VARCHAR2

) IS

v\_TotalCompra NUMBER := 0;

v\_CompraID NUMBER;

BEGIN

-- Insertar la compra en C\_Compras

INSERT INTO C\_Compras (CompraID, ProveedorID, FechaCompra, NumeroFactura, FormaPago, ImpuestoTotal, TotalCompra)

VALUES (seq\_c\_compra.NEXTVAL, p\_ProveedorID, SYSDATE, 'FAC-C-' || seq\_c\_compra.CURRVAL, p\_FormaPago, p\_ImpuestoTotal, v\_TotalCompra)

RETURNING CompraID INTO v\_CompraID;

-- Insertar detalles de la compra

FOR i IN p\_Productos.FIRST .. p\_Productos.LAST LOOP

INSERT INTO C\_DetalleCompras (DetalleCompraID, CompraID, ProductoID, Cantidad, PrecioUnitario, Subtotal)

VALUES (seq\_c\_detalle\_compra.NEXTVAL, v\_CompraID, p\_Productos(i).ProductoID, p\_Productos(i).Cantidad, p\_Productos(i).PrecioUnitario, p\_Productos(i).Subtotal);

-- Actualizar el total de la compra

v\_TotalCompra := v\_TotalCompra + p\_Productos(i).Subtotal;

END LOOP;

-- Actualizar el total de la compra en C\_Compras

UPDATE C\_Compras

SET TotalCompra = v\_TotalCompra

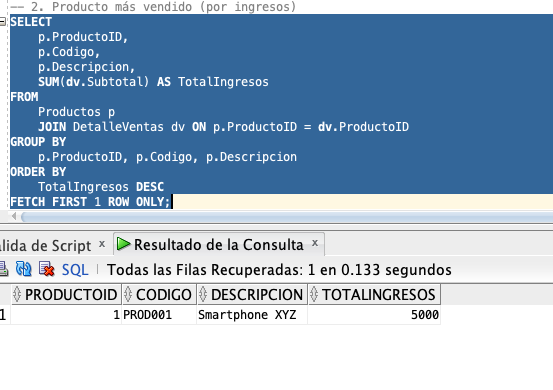
WHERE CompraID = v\_CompraID;

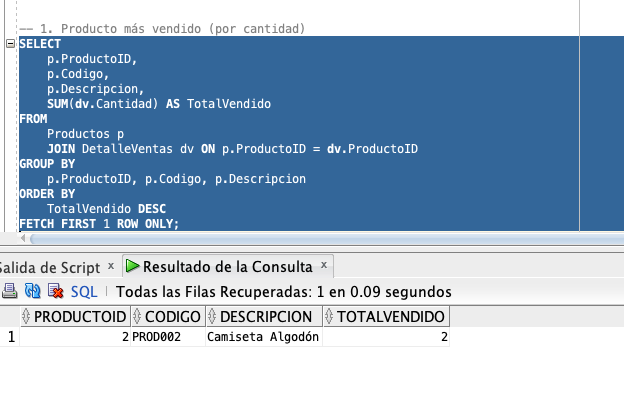
COMMIT;

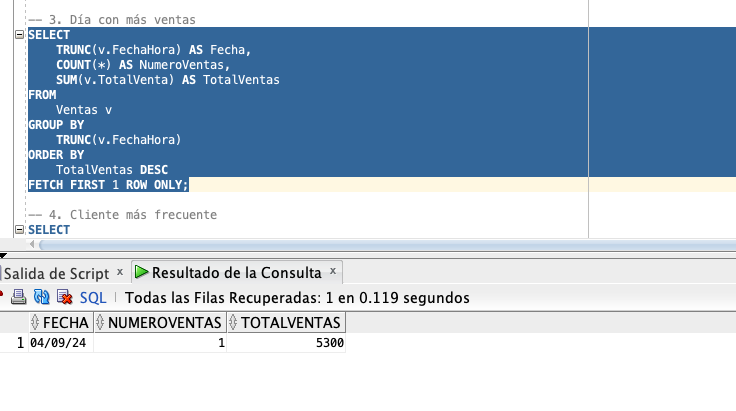
END;

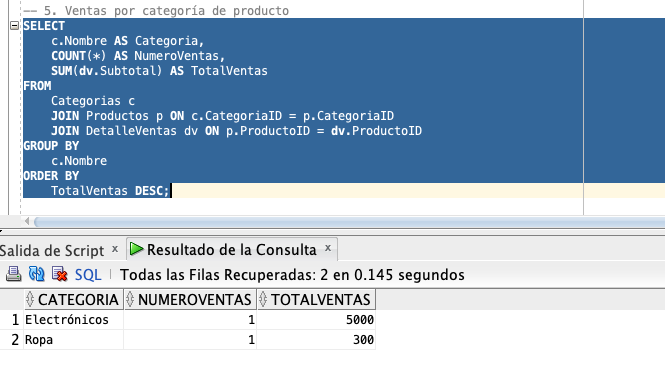
/

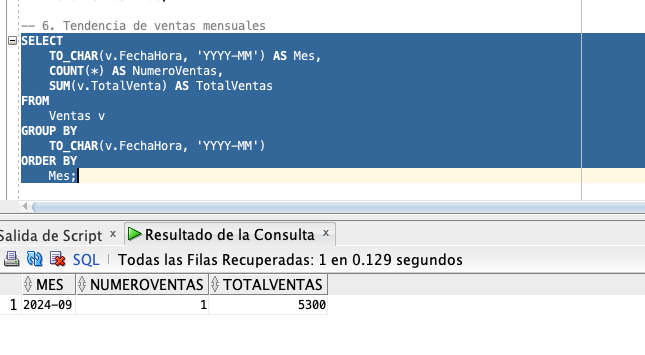
ANEXOS DE CAPTURAS DE PANTALLA DEL FUNCIONAMIENTO

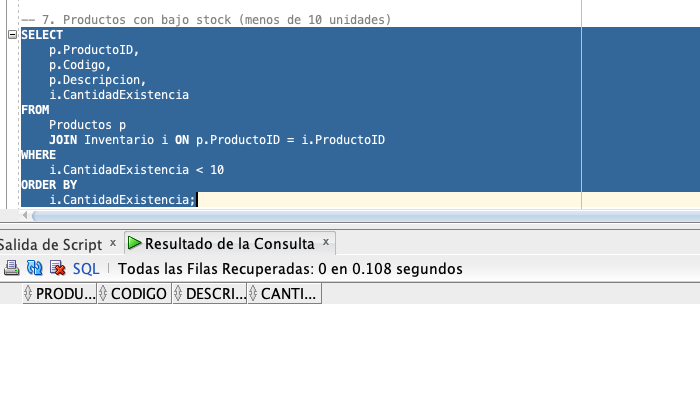




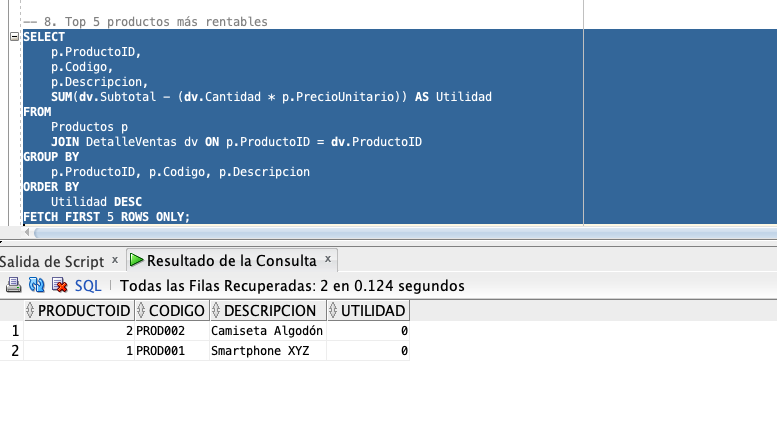


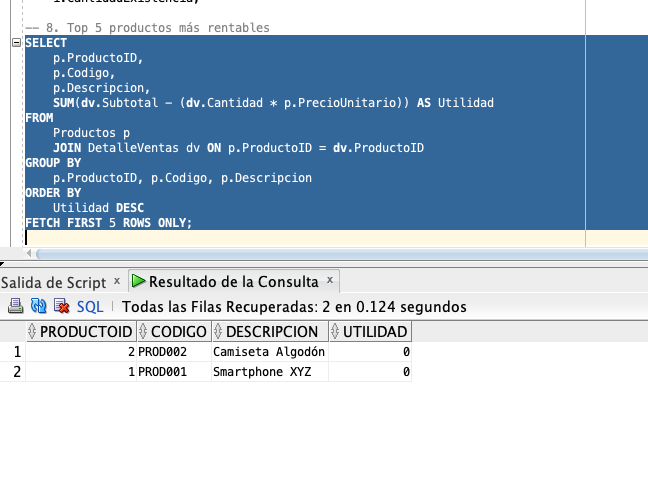


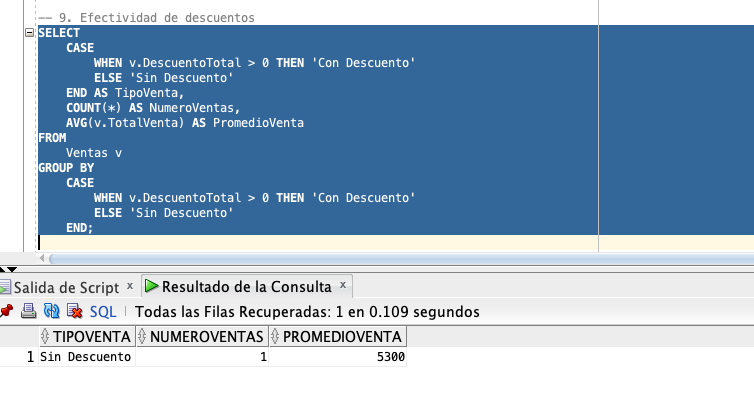


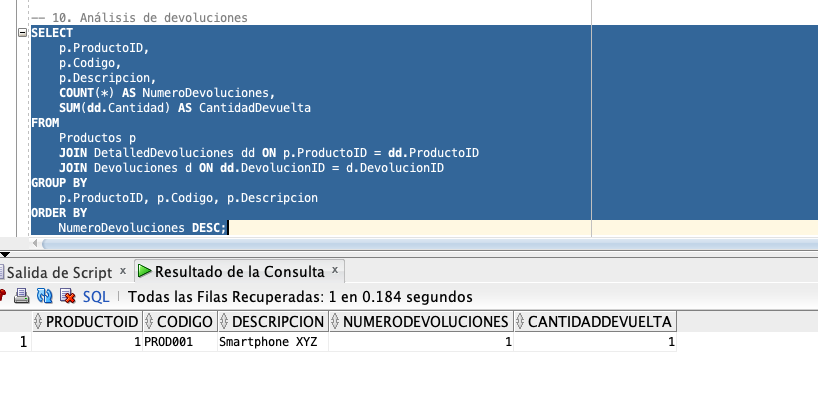


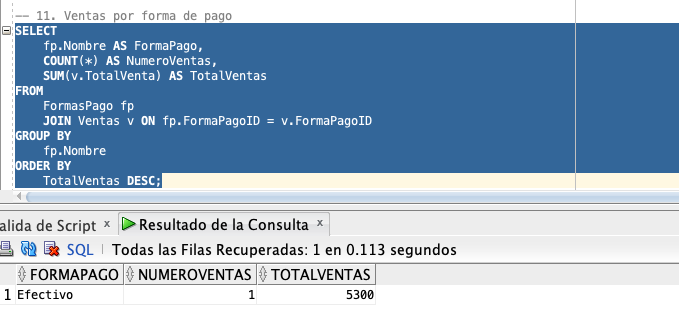
(Ningun producto tenia menos de 10 unidades en stock)

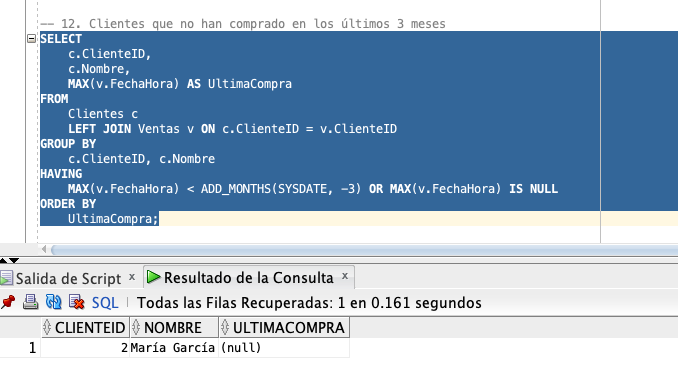


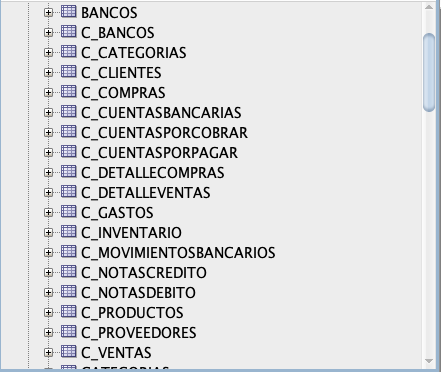


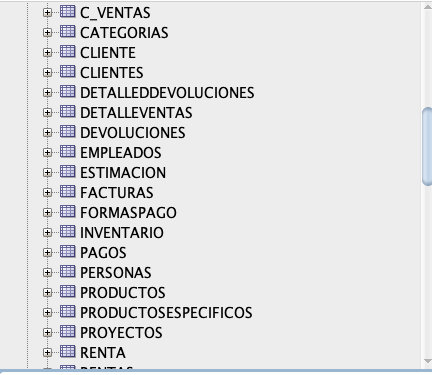












Nota. No todas las tablas pertenecen a los proyectos designados ya que la BD se a usado para diversas pruebas.

Conclusión:

Cada una de las BD se crearon tomando el enfoque de solucionar un problema en especifico, y ayudar con los procesos de registro y reporteria de cada una de las áreas en las cuales se enfoco el proyecto. Podemos concluir en que las tecnologías día con día nos facilitan las tareas diarias y reducen el margen de error hasta casi mitigarlos (si se toma en cuenta el error humano, ya que un sistema puede ser completamente sin errores)