ESCUELA POLITECNICA NACIONAL

Bases de Datos

Nombre: Joshua Morocho

1. Crear la Base de Datos y Tablas:

Crea una base de datos llamada tienda_online.

Dentro de la base de datos, crea las siguientes tablas:

- Clientes: Contendrá información básica sobre los clientes (id, nombre, apellido, email, teléfono, fecha de registro).
- Productos: Contendrá información sobre los productos (id, nombre, precio, stock, descripción).
- Pedidos: Registra los pedidos realizados por los clientes (id, cliente_id, fecha del pedido, total).
- Detalles_Pedido: Registra los detalles de cada pedido (id, pedido_id, producto_id, cantidad, precio unitario).

2. Restricciones:

- No se permiten valores nulos en campos como nombre, apellido, email, precio, y cantidad.
- Los precios deben ser positivos.
- El stock de los productos no puede ser negativo.
- Los nombres de los productos no deben repetirse.
- El email de los clientes debe ser único

```sql
-- Tabla Clientes
CREATE TABLE Clientes (
    id INTEGER PRIMARY KEY,
    nombre TEXT NOT NULL,
    apellido TEXT NOT NULL,
    email TEXT NOT NULL UNIQUE,
    telefono TEXT,
    fecha_registro DATETIME
);
--Tabla Productos
CREATE TABLE Productos (
    id INTEGER PRIMARY KEY ,
    nombre TEXT NOT NULL UNIQUE,
    precio REAL NOT NULL CHECK (precio > 0),
    stock INTEGER NOT NULL CHECK (stock >= 0),
    descripcion TEXT
);
-- Tabla Pedidos
CREATE TABLE Pedidos (
    id INTEGER PRIMARY KEY,
    cliente_id INTEGER NOT NULL,
    fecha_pedido DATETIME,
    total REAL NOT NULL CHECK (total >= 0),
    FOREIGN KEY (cliente_id) REFERENCES Clientes(id)
);
```

```sql
-- Tabla Detalles_Pedido
CREATE TABLE Detalles_Pedido (
    id INTEGER PRIMARY KEY,
    pedido_id INTEGER NOT NULL,
    producto_id INTEGER NOT NULL,
    cantidad INTEGER NOT NULL CHECK (cantidad > 0),
    precio_unitario REAL NOT NULL CHECK (precio_unitario > 0),
    FOREIGN KEY (pedido_id) REFERENCES Pedidos(id),
    FOREIGN KEY (producto_id) REFERENCES Productos(id)
);
```

Registros:

```sql
INSERT INTO Clientes (id,nombre, apellido, email, telefono)
VALUES
(001,'Alexander', 'Pérez', 'juan.perez@gmail.com', '093456789'),
(002,'Josias', 'Gómez', 'maria.gomez@gmail.com', '097654321'),
(003,'Carlos', 'Zambrano', 'carlos.lopez@gmail.com', '0956789123'),
(004,'Leticis', 'Martínez', 'ana.martinez@gmail.com', '0921654987'),
(005,'Florinda', 'Fernández', 'luis.fernandez@gmail.com', '0954321789');

INSERT INTO Productos (id,nombre, precio, stock, descripcion)
VALUES
(11,'Laptop', 800.00, 10, 'Laptop de 16GB de RAM'),
(12,'Smartphone', 500.00, 20, 'Smartphone pantalla de 6.5 pulgadas'),
(13,'Tablet', 300.00, 15, 'Tablet 64GB de almacenamiento'),
(14,'Auriculares', 100.00, 50, 'Auriculares inalámbricos'),
(15,'Reloj inteligente', 250.00, 30, 'Reloj con monitor de salud');

INSERT INTO Pedidos (id, cliente_id, total)
VALUES
(1,001, 800.00),
(2,002, 500.00),
(3,003, 300.00),
(4,004, 100.00),
(5,005, 250.00);
```

```sql
INSERT INTO Detalles_Pedido (pedido_id, producto_id, cantidad, precio_unitario)
VALUES
(1, 11, 1, 800.00),
(2, 12, 1, 500.00),
(3, 13, 1, 300.00),
(4, 14, 1, 100.00),
(5, 15, 1, 250.00);
```

○ Función y consultas para obtener el nombre completo de un cliente:

```sql
-- Funciones
-- Función para obtener el nombre completo de un cliente:
SELECT nombre || ' ' || apellido AS nombreCompleto FROM Clientes
WHERE id = 002;
```

nombreCompleto
Josias Gómez

○ Función para calcular el descuento de un producto:

```
-- Función para calcular el descuento de un producto:
SELECT precio * (1 - 0.15) AS precio_con_descuento FROM Productos
WHERE id =12;
```

	precio_con_descuento
1	425.0

○ Función para calcular el total de un pedido:

```
-- Función para calcular el total de un pedido
SELECT SUM(dp.cantidad * dp.precio_unitario) AS total FROM Detalles_Pedido dp
JOIN Pedidos p ON dp.pedido_id = p.id WHERE p.id = 2;
```

	total
1	500.0

○ Función para verificar la disponibilidad de stock de un producto:

```
-- Función para verificar la disponibilidad de stock de un producto:
SELECT CASE
            WHEN stock >= 0 THEN 'TRUE'
            ELSE 'FALSE'
        END AS disponibilidad_Stock_Producto
FROM Productos WHERE id = 13;

-- Función para calcular la antigüedad de un cliente
SELECT (julianday('now') - julianday(fechaRegistro)) / 365 AS antiguedad
```

	disponibilidad_Stock_Producto
1	TRUE

○ Función para calcular la antigüedad de un cliente:

```
-- Función para calcular la antigüedad de un cliente
SELECT (julianday('now') - julianday(fechaRegistro)) / 365 AS antiguedad
FROM Clientes WHERE id=003;
```

antiguedad
NULL

Parte 2

EJERCICIO 1

Se transcribe la imagen y se explica el codigo

```sql
-- Parte 2
-- 1
CREATE FUNCTION CalcularTotalOrden(id_orden INT)
RETURNS DECIMAL(16,2)
DETERMINISTIC
BEGIN
    DECLARE total DECIMAL(16,2);
    DECLARE iva DECIMAL(16,2);

    SET iva = 0.15;

    SELECT SUM(P.precio * O.cantidad) INTO total
    FROM Ordenes O
    JOIN Productos P ON O.producto_id = P.ProductoID
    WHERE O.OrdenID = id_orden;

    SET total = total + (total * iva);

    RETURN total;
END//

DELIMITER ;
```

Explicación:

Calcula el total de una orden específica, incluyendo el IVA. Toma como entrada el ID de la orden, cambia el valor del iva al 0.15 (15%), suma los precios de los productos multiplicados por sus cantidades, y luego añade un 15% de IVA al total antes de devolverlo.

EJERCICIO 2

```sql
-- 2
DELIMITER $$

CREATE FUNCTION CalcularEdad(fecha_nacimiento DATE)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE edad INT;
    SET edad = TIMESTAMPDIFF(YEAR, fecha_nacimiento, CURDATE());
    RETURN edad;
END $$

DELIMITER ;
```

Explicación:

Calcula la edad de una persona en años a partir de su fecha de nacimiento. Toma como entrada la fecha de nacimiento, utiliza la función TIMESTAMPDIFF para calcular la diferencia en años entre la fecha de nacimiento proporcionada (fecha_nacimiento) y la fecha actual (CURDATE()). El resultado se almacena en la variable edad y devuelve el entero que representa la edad.

EJERCICIO 3

```sql
-- 3
DELIMITER $$

CREATE FUNCTION VerificarStock(producto_id INT)
RETURNS BOOLEAN
DETERMINISTIC
BEGIN
    DECLARE stock INT;
    SELECT Existencia INTO stock
    FROM Productos
    WHERE ProductoID = producto_id;
    IF stock > 0 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END $$
DELIMITER ;
```

Explicación:

Esta función verifica si un producto tiene stock disponible. Toma como entrada el ID de un producto, declara una variable llamada stock que almacenará la cantidad de stock del producto consultado, Si el stock es mayor que cero, la función devuelve TRUE, indicando que hay stock disponible. Si el stock no es mayor que cero, se ejecuta la siguiente línea:

```sql
ELSE
    RETURN FALSE;
END IF;
```

y devuelve un valor booleano (TRUE o FALSE) dependiendo de si la cantidad de stock es mayor que cero.

Ejercicio 4

```sql
-- 4
DELIMITER $$
CREATE FUNCTION CalcularSaldo(id_cuenta INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE saldo DECIMAL(10,2);

    SELECT SUM(CASE
        WHEN tipo_transaccion = 'deposito' THEN monto
        WHEN tipo_transaccion = 'retiro' THEN -monto
        ELSE 0
    END) INTO saldo
    FROM Transacciones
    WHERE cuenta_id = id_cuenta;

    RETURN saldo;
END $$
DELIMITER ;
```

Explicación:

Calcula el saldo de una cuenta bancaria a partir de las transacciones registradas. Se crea la función con el valor de entrada de id_cuenta luego la lonea:

RETURNS DECIMAL(10,2)

Indica que la función devolverá un valor de tipo DECIMAL con una precisión de 10 dígitos en total, de los cuales 2 serán decimales.

Utiliza una expresión CASE para determinar cómo se debe calcular cada transacción:

- WHEN tipo_transaccion = 'deposito' THEN monto: Si la transacción es un depósito, se suma el monto.
- WHEN tipo_transaccion = 'retiro' THEN -monto: Si la transacción es un retiro, se resta el monto (se suma un valor negativo).
- ELSE 0: Si el tipo de transacción no es ni depósito ni retiro, se considera como 0.
- INTO saldo: Almacena el resultado de la suma en la variable saldo.
- FROM Transacciones: Indica que la información se obtiene de la tabla Transacciones.
- WHERE cuenta_id = id_cuenta: Filtra las transacciones para que solo se consideren aquellas que pertenecen a la cuenta especificada por id_cuenta.

Finalmente devuelve un valor decimal que representa el saldo total.