Proyecto: Sistema de Gestión de Tienda en Línea

Descripción:

Este proyecto consiste en un sistema completo para la gestión de una tienda en línea, que incluye la administración de productos, categorías, clientes, pedidos y pagos. Es ideal para practicar habilidades en bases de datos y SQL.

Objetivos:

- Crear una base de datos relacional que permita gestionar todas las operaciones de una tienda en línea.
- Implementar consultas SQL para extraer información relevante de la base de datos.

Esquema de la Base de Datos

El sistema está compuesto por seis tablas principales:

1. Categorías: Define las diferentes categorías de productos.
2. Productos: Contiene los detalles de los productos disponibles para la venta.
3. Clientes: Registra la información básica de los clientes.
4. Pedidos: Gestiona los pedidos realizados por los clientes.
5. Detalles_Pedido: Almacena información sobre los productos incluidos en cada pedido.
6. Pagos: Registra los pagos realizados por los pedidos.

Creación de las tablas:

```sql
CREATE TABLE Categorías (
    categoria_id INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    descripcion TEXT
);

CREATE TABLE Productos (
    producto_id INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(200) NOT NULL,
    descripcion TEXT,
    precio DECIMAL(10,2) NOT NULL,
    stock INT NOT NULL,
    categoria_id INT,
    FOREIGN KEY (categoria_id) REFERENCES Categorías(categoria_id)
);

CREATE TABLE Clientes (
    cliente_id INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    apellido VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    telefono VARCHAR(15),
    direccion TEXT
);

CREATE TABLE Pedidos (
    pedido_id INT PRIMARY KEY AUTO_INCREMENT,
    cliente_id INT,
```

```
    fecha_pedido DATETIME DEFAULT CURRENT_TIMESTAMP,
    estado ENUM('Pendiente', 'Enviado', 'Entregado', 'Cancelado'),
    coste_envio DECIMAL(10, 2),
    FOREIGN KEY (cliente_id) REFERENCES Clientes(cliente_id)
);

CREATE TABLE Detalles_Pedido (
    detalle_id INT PRIMARY KEY AUTO_INCREMENT,
    pedido_id INT,
    producto_id INT,
    cantidad INT NOT NULL,
    precio_unitario DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (pedido_id) REFERENCES Pedidos(pedido_id),
    FOREIGN KEY (producto_id) REFERENCES Productos(producto_id)
);

CREATE TABLE Pagos (
    pago_id INT PRIMARY KEY AUTO_INCREMENT,
    pedido_id INT,
    monto_total DECIMAL(10,2) NOT NULL,
    fecha_pago DATETIME DEFAULT CURRENT_TIMESTAMP,
    metodo_pago ENUM('Tarjeta', 'Transferencia', 'Efectivo'),
    estado_pago ENUM('Completado', 'Pendiente', 'Fallido'),
    FOREIGN KEY (pedido_id) REFERENCES Pedidos(pedido_id)
);
```

Insertar datos a la tabla:

```sql
-- Insertar categorías
INSERT INTO Categorías (nombre, descripcion) VALUES
('Electrónica', 'Dispositivos electrónicos y gadgets'),
('Ropa', 'Vestimenta y accesorios'),
('Hogar', 'Artículos para el hogar'),
('Libros', 'Libros de diferentes géneros'),
('Deportes', 'Artículos deportivos');


-- Insertar productos
INSERT INTO Productos (nombre, descripcion, precio, stock, categoria_id)
VALUES
('Smartphone XYZ', 'Smartphone con pantalla de alta resolución.', 299.99,
50, 1),
('Camiseta Básica', 'Camiseta de algodón cómoda.', 19.99, 100, 2),
('Sofá Modular', 'Sofá cómodo y elegante para el hogar.', 499.99, 20, 3),
('El Señor de los Anillos', 'Novela de fantasía épica.', 29.99, 75, 4),
('Balón de Fútbol', 'Balón de fútbol profesional.', 49.99, 30, 5),
('Laptop Gaming', 'Laptop de alto rendimiento para juegos.', 1299.99, 15,
1),
('Pantalones Vaqueros', 'Pantalones vaqueros clásicos.', 59.99, 60, 2),
('Lámpara de Mesa', 'Lámpara de mesa moderna.', 79.99, 40, 3),
('Cien Años de Soledad', 'Novela literaria.', 19.99, 90, 4),
('Raqueta de Tenis', 'Raqueta de tenis profesional.', 99.99, 25, 5);


-- Insertar detalles del pedido (se amplían los ejemplos con más productos
y cantidades)
INSERT INTO Detalles_Pedido (pedido_id, producto_id, cantidad,
precio_unitario) VALUES
(1, 1, 1, 299.99),
(1, 2, 2, 19.99),
```

```
(2, 3, 1, 499.99),
(2, 4, 1, 29.99),
(1,5,1, 49.99),
(2,6,1,1299.99),
(1,7,1,59.99),
(2,8,1,79.99);
```

```
-- Insertar clientes (con dirección)
INSERT INTO Clientes (nombre, apellido, email, telefono, direccion) VALUES
('Juan', 'Pérez', 'juan.perez@example.com', '123456789', 'Calle Ejemplo
123'),
('María', 'Gómez', 'maria.gomez@example.com', '987654321', 'Avenida
Principal 456');

-- Insertar pedidos (con coste_envio)
INSERT INTO Pedidos (cliente_id, estado, coste_envio) VALUES
(1, 'Pendiente', 5.99),
(2, 'Enviado', 7.50);


-- ... (Código anterior para insertar datos en Detalles_Pedido)

-- Insertar pagos (con estado_pago)
INSERT INTO Pagos (pedido_id, monto_total, metodo_pago, estado_pago) VALUES
(1, (299.99 + (19.99 * 2) + 5.99), 'Tarjeta', 'Completado'),  -- Se incluye
el coste de envío
(2, (499.99 + 7.50), 'Transferencia', 'Completado'); -- Se incluye el coste
de envío
```

Consultas

3. Consultas con Agregaciones

a) Total de productos en stock por categoría

```
SELECT c.nombre AS categoria, SUM(p.stock) AS total_stock
FROM Productos p
JOIN Categorías c ON p.categoria_id = c.categoria_id
GROUP BY c.nombre;
```

b) Total de ingresos por pedido

```
SELECT dp.pedido_id, SUM(dp.cantidad * dp.precio_unitario) AS
total_ingresos
FROM Detalles_Pedido dp
GROUP BY dp.pedido_id;
```

c) Número de pedidos por cliente

```
SELECT c.cliente_id, c.nombre, c.apellido, COUNT(p.pedido_id) AS
total_pedidos
FROM Clientes c
LEFT JOIN Pedidos p ON c.cliente_id = p.cliente_id
GROUP BY c.cliente_id, c.nombre, c.apellido;
```

4. Consultas Avanzadas

a) Obtener los pedidos con su cliente y el total pagado

```sql
SELECT p.pedido_id, c.nombre AS cliente, c.apellido, pa.monto_total,
pa.metodo_pago
FROM Pedidos p
JOIN Clientes c ON p.cliente_id = c.cliente_id
JOIN Pagos pa ON p.pedido_id = pa.pedido_id;
```

b) Productos más vendidos (ordenados por cantidad total)

```sql
SELECT p.nombre AS producto, SUM(dp.cantidad) AS total_vendido
FROM Detalles_Pedido dp
JOIN Productos p ON dp.producto_id = p.producto_id
GROUP BY p.nombre
ORDER BY total_vendido DESC;
```

c) Clientes que han gastado más dinero

```sql
SELECT c.cliente_id, c.nombre, c.apellido, SUM(pa.monto_total) AS
total_gastado
FROM Clientes c
JOIN Pedidos p ON c.cliente_id = p.cliente_id
JOIN Pagos pa ON p.pedido_id = pa.pedido_id
GROUP BY c.cliente_id, c.nombre, c.apellido
ORDER BY total_gastado DESC;
```

d) Pedidos con más de 3 productos

```sql
SELECT dp.pedido_id, COUNT(dp.detalle_id) AS total_productos
FROM Detalles_Pedido dp
GROUP BY dp.pedido_id
HAVING total_productos > 3;
```

5. Consultas de Actualización

a) Actualizar el stock de un producto

```sql
UPDATE Productos
SET stock = stock - 5
WHERE producto_id = 1;
```

b) Cambiar el estado de un pedido a "Entregado"

```sql
UPDATE Pedidos
SET estado = 'Entregado'
WHERE pedido_id = 1;
```

6. Consultas de Eliminación

a) Eliminar un cliente (y sus pedidos relacionados)

```sql
DELETE FROM Clientes
WHERE cliente_id = 1;
```

b) Eliminar productos sin stock

```sql
DELETE FROM Productos
```

```sql
WHERE stock = 0;
```

7. Consultas con Subconsultas

a) Obtener el producto más caro

```sql
SELECT nombre, precio
FROM Productos
WHERE precio = (SELECT MAX(precio) FROM Productos);
```

b) Clientes que han realizado al menos un pedido

```sql
SELECT *
FROM Clientes
WHERE cliente_id IN (SELECT cliente_id FROM Pedidos);
```

c) Pedidos con un monto total mayor al promedio de todos los pedidos

```sql
SELECT p.pedido_id, pa.monto_total
FROM Pedidos p
JOIN Pagos pa ON p.pedido_id = pa.pedido_id
WHERE pa.monto_total > (SELECT AVG(monto_total) FROM Pagos);
```