UNIVERSIDAD AUTÓNOMA METROPOLITANA

UNIDAD CUAJIMALPA

LICENCIATURA EN TECNOLOGÍAS Y SISTEMAS DE INFORMACIÓN

BASE DE DATOS

JOSHUA BENJAMIN CABRIALES AGUILAR

Introducción

En este proyecto se resuelve un problema que tiene una tienda online el cual es que esta necesita un sistema para gestionar todas las operaciones que se realizan en el día a día incluyendo información de sus productos, los clientes junto a los pedidos, reseñas de estos y finalmente la categoría.

Se realizará una base de datos normalizada, junto con el diagrama ER, el esquema normalizado para evitar recurrencias, repetición de datos innecesarios, orden en las tablas y mejorar la optimización de estas para mostrar información de forma más clara.

Objetivo general

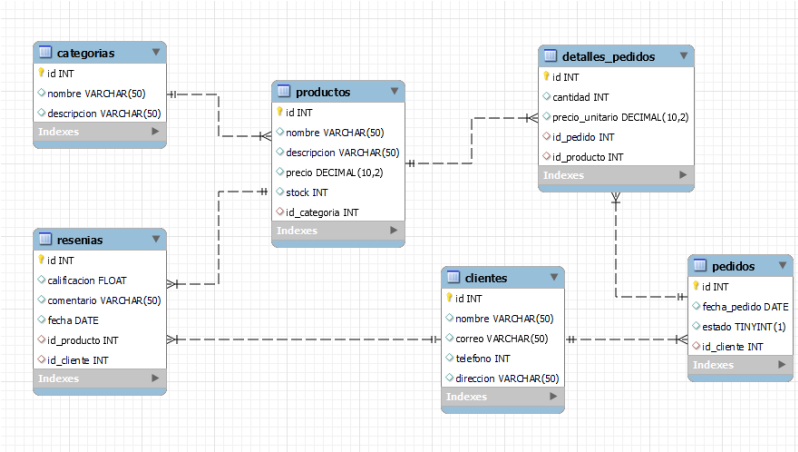
- Desarrollar un sistema en una base de datos donde se pueda gestionar y almacenar de forma clara y organizada la información de una tienda online.

Objetivos específicos

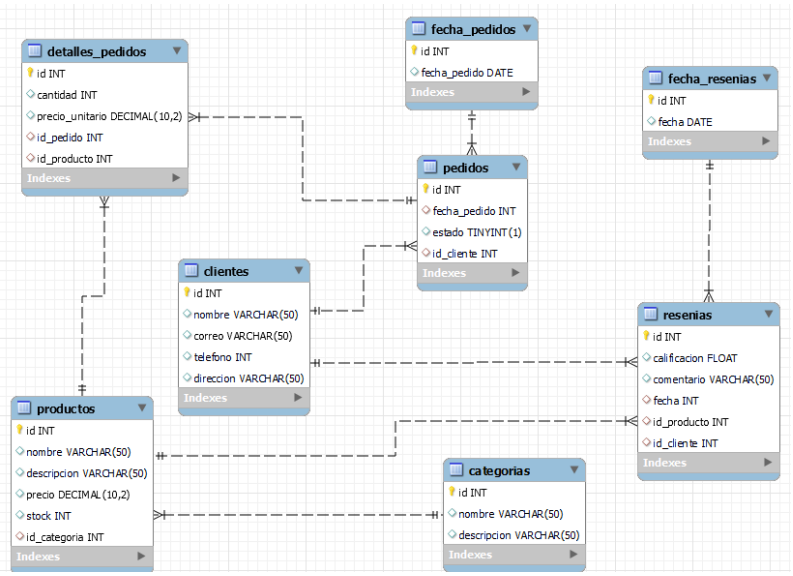
- Diseñar el diagrama ER para mostrar la conexión entre tablas, así como claves y cardinalidad.
- Crear el esquema de tablas normalizado en su tercera forma para evitar recurrencias y demás problemas comunes en BD.
- Identificar claves primarias y foráneas de las tablas, así como su relación.
- Optimizar la BD mediante procedimientos (store procedures) e índices (index)
- Realizar consultas con datos de prueba para comprobar la optimización, búsqueda y resolución de problemas mediante query hacia la BD.

Análisis y Diseño

1. Diagrama Entidad-Relación (ER)



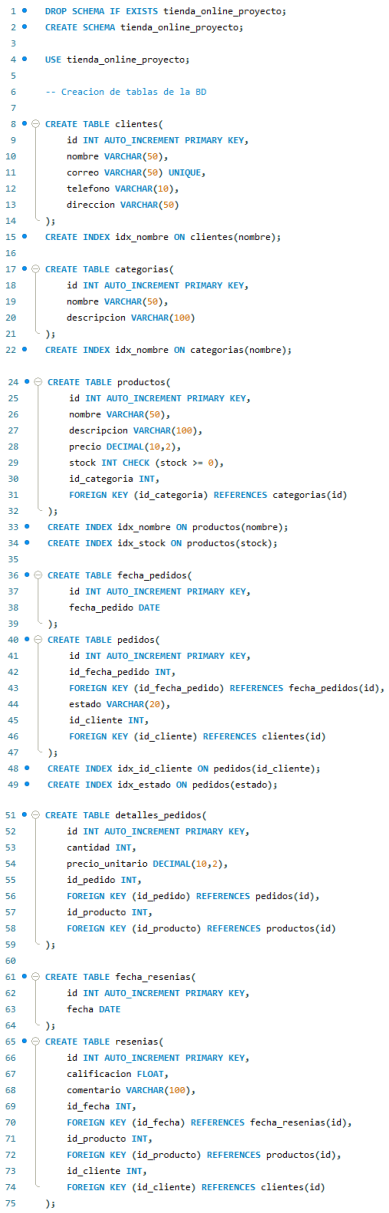
2. Esquema en 3NF: La justificación de este esquema en su 3NF es que las fechas se repetirían, ya que los pedidos y las reseñas pueden ser insertadas en fechas que ya existen, duplicando datos y haciéndolo menos claro, por lo que las fechas de las entidades de pedidos y reseñas pasan a tablas independientes para tener una relación “una fecha – muchos pedidos” y “una fecha – muchas reseñas”, con esto solo se tendría que identificar el **id** de la fecha y relacionarlo a un nuevo pedido o nueva reseña.



3. Identificación de claves
    - Claves primarias
        - clientes.id
        - categorias.id
        - productos.id
        - fecha_pedidos.id
        - pedidos.id
        - detalles_pedidos.id
        - fecha_resenias.id
        - resenias.id
    - Claves foráneas
        - productos.id_categoria
        - pedidos.id_fecha_pedido
        - pedidos.id_cliente
        - detalles_pedidos.id_pedido
        - detalles_pedidos.id_producto
        - resenias.id_fecha
        - resenias.id_producto
        - resenias.id_cliente
    - Claves candidatas
        - clientes.correo
        - clientes.telefono
        - categorias.nombre
        - productos.nombre

Implementación de la Base de Datos

4. Script que incluye
    a. Claves primarias y foráneas
    b. Restricciones (stock no debe aceptar números negativos, los correos no deben repetirse)
    c. Índices para optimización de consultas



```sql
DROP SCHEMA IF EXISTS tienda_online_proyecto;
CREATE SCHEMA tienda_online_proyecto;

USE tienda_online_proyecto;

-- Creacion de tablas de la BD

CREATE TABLE clientes(
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50),
    correo VARCHAR(50) UNIQUE,
    telefono VARCHAR(10),
    direccion VARCHAR(50)
);
CREATE INDEX idx_nombre ON clientes(nombre);

CREATE TABLE categorias(
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50),
    descripcion VARCHAR(100)
);
CREATE INDEX idx_nombre ON categorias(nombre);

CREATE TABLE productos(
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50),
    descripcion VARCHAR(100),
    precio DECIMAL(10,2),
    stock INT CHECK (stock >= 0),
    id_categoria INT,
    FOREIGN KEY (id_categoria) REFERENCES categorias(id)
);
CREATE INDEX idx_nombre ON productos(nombre);
CREATE INDEX idx_stock ON productos(stock);

CREATE TABLE fecha_pedidos(
    id INT AUTO_INCREMENT PRIMARY KEY,
    fecha_pedido DATE
);
CREATE TABLE pedidos(
    id INT AUTO_INCREMENT PRIMARY KEY,
    id_fecha_pedido INT,
    FOREIGN KEY (id_fecha_pedido) REFERENCES fecha_pedidos(id),
    estado VARCHAR(20),
    id_cliente INT,
    FOREIGN KEY (id_cliente) REFERENCES clientes(id)
);
CREATE INDEX idx_id_cliente ON pedidos(id_cliente);
CREATE INDEX idx_estado ON pedidos(estado);

CREATE TABLE detalles_pedidos(
    id INT AUTO_INCREMENT PRIMARY KEY,
    cantidad INT,
    precio_unitario DECIMAL(10,2),
    id_pedido INT,
    FOREIGN KEY (id_pedido) REFERENCES pedidos(id),
    id_producto INT,
    FOREIGN KEY (id_producto) REFERENCES productos(id)
);

CREATE TABLE fecha_resenias(
    id INT AUTO_INCREMENT PRIMARY KEY,
    fecha DATE
);
CREATE TABLE resenias(
    id INT AUTO_INCREMENT PRIMARY KEY,
    calificacion FLOAT,
    comentario VARCHAR(100),
    id_fecha INT,
    FOREIGN KEY (id_fecha) REFERENCES fecha_resenias(id),
    id_producto INT,
    FOREIGN KEY (id_producto) REFERENCES productos(id),
    id_cliente INT,
    FOREIGN KEY (id_cliente) REFERENCES clientes(id)
);
```

5. Script para poblar la base con datos de prueba

```sql
USE tienda_online_proyecto;

-- Clientes 15
INSERT INTO clientes(id, nombre, correo, telefono, direccion) VALUES(0, "Joshua", "joshua@correo.com", '5512345678', "atizapan");
INSERT INTO clientes(id, nombre, correo, telefono, direccion) VALUES(0, "Fernanda", "fernanda@correo.com", '5514235867', "lomas verdes");
INSERT INTO clientes(id, nombre, correo, telefono, direccion) VALUES(0, "Marlon", "marlon@correo.com", '5516792463', "cuspide");
INSERT INTO clientes(id, nombre, correo, telefono, direccion) VALUES(0, "Alberto", "alberto@correo.com", '5519342311', "satelite");
INSERT INTO clientes(id, nombre, correo, telefono, direccion) VALUES(0, "Pablo", "pablo@correo.com", '5516813647', "azcapotzalco");

INSERT INTO clientes(id, nombre, correo, telefono, direccion) VALUES(0, "Irving", "irving@correo.com", '5514793258', "coyoacan");
INSERT INTO clientes(id, nombre, correo, telefono, direccion) VALUES(0, "Esperanza", "esperanza@correo.com", '5514632028', "toluca");
INSERT INTO clientes(id, nombre, correo, telefono, direccion) VALUES(0, "Alejandro", "alejandro@correo.com", '5593485621', "cuajimalpa");
INSERT INTO clientes(id, nombre, correo, telefono, direccion) VALUES(0, "Paola", "paola@correo.com", '5578240169', "tlalpan");
INSERT INTO clientes(id, nombre, correo, telefono, direccion) VALUES(0, "Eder", "eder@correo.com", '5510236580', "polanco");

INSERT INTO clientes(id, nombre, correo, telefono, direccion) VALUES(0, "Mariel", "mariel@correo.com", '5579605781', "condesa");
INSERT INTO clientes(id, nombre, correo, telefono, direccion) VALUES(0, "Daniel", "daniel@correo.com", '5531025791', "interlomas");
INSERT INTO clientes(id, nombre, correo, telefono, direccion) VALUES(0, "Aldo", "aldo@correo.com", '5546578249', "xochimilco");
INSERT INTO clientes(id, nombre, correo, telefono, direccion) VALUES(0, "Catherine", "catherine@correo.com", '5569873020', "lerma");
INSERT INTO clientes(id, nombre, correo, telefono, direccion) VALUES(0, "Camila", "camila@correo.com", '5546057991', "roma");


 -- Categorias
 INSERT INTO categorias(id, nombre, descripcion) VALUES(0, "limpieza", "limpiar interiores del hogar");
 INSERT INTO categorias(id, nombre, descripcion) VALUES(0, "electronica", "equipos de computo, accesorios y tecnologia");
 INSERT INTO categorias(id, nombre, descripcion) VALUES(0, "alimentos y bebidas", "comida, frutas, verduras y bebidas");
 INSERT INTO categorias(id, nombre, descripcion) VALUES(0, "ropa", "Vestimentas para usar");
 INSERT INTO categorias(id, nombre, descripcion) VALUES(0, "electrodomesticos", "aparatos para el hogar");
 INSERT INTO categorias(id, nombre, descripcion) VALUES(0, "bolsas", "accesorios faciles de transportar y utiles para almacenar cosas");
 INSERT INTO categorias(id, nombre, descripcion) VALUES(0, "accesorios", "objetos variados para el uso diario");
 INSERT INTO categorias(id, nombre, descripcion) VALUES(0, "higiene", "utiles para matener una buena limpieza corporal");


-- Productos 30
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "suavitel", "jabon para lavar la ropa", 60, 25, 1);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "mochila Nike", "mochila para llevar utiles escolares", 2300, 43, 6);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "mouse", "periferico para controlar facilmente la computadora", 350, 12, 2);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "teclado", "periferico para escribir en computadora", 800, 10, 2);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "pantalla Samsung", "pantalla para visualizar contenido", 8200, 67, 2);

INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "iPhone 13", "dispositivo celular de Apple", 9600, 3, 2);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "escoba", "especial para barrer en interiores", 120, 65, 1);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "playera", "ropa para parte superior del cuerpo", 300, 94, 4);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "pantalon", "ropa para parte inferior del cuerpo", 750, 19, 4);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "refrigerador", "usado para mantener alimentos en temperaturas bajas", 5000, 98, 5);

INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "estufa", "electrodomestico para calentar alimentos y cocinar en casa", 3500, 32, 5);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "termo", "mantiene las bebidas con su temperatura", 230, 14, 7);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "xbox one", "consola de videojuegos", 4700, 61, 2);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "audifonos", "accesorio para escuchar musica", 3200, 45, 2);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "apple watch", "reloj inteligente para vincular a dispositivo celular", 2800, 81, 2);
```

```sql
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "laptop", "equipo de computo portatil especial para viajes", 7400, 90, 2);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "manzana", "fruta de temporada de excelente calidad", 37, 28, 3);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "zanahoria", "verdura en perfecto estado", 28, 19, 3);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "calcetines", "ropa interior comoda de algodon", 80, 83, 4);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "cable hdmi", "cable conector para pantallas y dispositivos electronicos", 110, 65, 2);

INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "gorra", "accesorio para la cabeza", 560, 60, 7);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "cafetera", "electrodomestico para preparar cafe", 2300, 34, 5);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "plancha", "", 450, 49, 5);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "pinol", "usado para lavar la ropa", 50, 98, 1);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "jabon de manos", "usado para lavar la ropa", 35, 72, 1);

INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "sabritas", "usado para lavar la ropa", 27, 51, 3);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "cepillo dental", "usado para lavar la ropa", 34, 73, 8);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "desodorante", "usado para lavar la ropa", 46, 36, 8);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "perfume", "usado para lavar la ropa", 78, 60, 8);
INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria) VALUES(0, "coca-cola", "usado para lavar la ropa", 32, 24, 3);
```

```sql
-- Fecha de pedidos
INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-06-14");
INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-07-20");
INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-05-03");
INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-03-17");
INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-01-28");

INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-04-06");
INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-02-12");
INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-07-21");
INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-06-09");
INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-05-11");

INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-03-23");
INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-01-01");
INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-04-15");
INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-02-26");
INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-07-07");

INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-06-19");
INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-05-22");
INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-04-02");
INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-03-18");
INSERT INTO fecha_pedidos(id, fecha_pedido) VALUES(0, "2025-01-24");
```

```sql
-- Pedidos 20
INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 2, 'pendiente', 2);
INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 1, 'pendiente', 4);
INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 4, 'enviado', 8);
INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 3, 'pendiente', 1);
INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 5, 'pendiente', 9);

INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 6, 'enviado', 11);
INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 7, 'pendiente', 10);
INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 10, 'pendiente', 3);
INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 8, 'enviado', 5);
INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 9, 'pendiente', 6);

INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 11, 'enviado', 15);
INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 15, 'pendiente', 12);
INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 14, 'pendiente', 14);
INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 13, 'pendiente', 13);
INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 12, 'enviado', 2);

INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 18, 'pendiente', 6);
INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 16, 'pendiente', 9);
INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 19, 'pendiente', 6);
INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 20, 'pendiente', 1);
INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, 17, 'enviado', 6);


-- Detalles de pedidos 25
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 3, 800, 2, 4);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 70, 34, 18, 27);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 5, 230, 5, 12);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 7, 300, 11, 8);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 10, 560, 9, 21);

INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 15, 37, 19, 17);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 22, 8200, 1, 5);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 4, 50, 12, 24);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 9, 3200, 4, 14);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 1, 60, 17, 1);

INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 6, 78, 7, 29);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 31, 4700, 13, 13);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 8, 2300, 3, 2);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 2, 27, 16, 26);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 5, 80, 6, 19);

INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 19, 120, 14, 7);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 13, 2300, 10, 22);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 9, 28, 15, 18);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 7, 750, 8, 9);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 2, 450, 20, 23);

INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 11, 9600, 1, 6);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 30, 35, 13, 25);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 17, 2800, 7, 15);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 6, 350, 18, 3);
INSERT INTO detalles_pedidos(id, cantidad, precio_unitario, id_pedido, id_producto) VALUES(0, 16, 46, 4, 28);
```

```sql
-- Fecha de resenias
INSERT INTO fecha_resenias(id, fecha) VALUES(0, '2025-06-14');
INSERT INTO fecha_resenias(id, fecha) VALUES(0, '2025-06-2');
INSERT INTO fecha_resenias(id, fecha) VALUES(0, '2025-06-29');
INSERT INTO fecha_resenias(id, fecha) VALUES(0, '2025-06-17');
INSERT INTO fecha_resenias(id, fecha) VALUES(0, '2025-06-5');

INSERT INTO fecha_resenias(id, fecha) VALUES(0, '2025-06-26');
INSERT INTO fecha_resenias(id, fecha) VALUES(0, '2025-06-12');
INSERT INTO fecha_resenias(id, fecha) VALUES(0, '2025-06-8');
INSERT INTO fecha_resenias(id, fecha) VALUES(0, '2025-06-21');
INSERT INTO fecha_resenias(id, fecha) VALUES(0, '2025-06-13');
```

```sql
-- Resenias 10
INSERT INTO resenias(id, calificacion, comentario, id_fecha, id_producto, id_cliente) VALUES(0, 8.5, 'Muy buen producto, llegó a tiempo.', 3, 12, 5);
INSERT INTO resenias(id, calificacion, comentario, id_fecha, id_producto, id_cliente) VALUES(0, 7.0, 'Cumple con lo básico.', 7, 8, 11);
INSERT INTO resenias(id, calificacion, comentario, id_fecha, id_producto, id_cliente) VALUES(0, 9.0, 'Excelente, superó mis expectativas.', 1, 3, 2);
INSERT INTO resenias(id, calificacion, comentario, id_fecha, id_producto, id_cliente) VALUES(0, 5.5, 'No era lo que esperaba.', 6, 20, 9);
INSERT INTO resenias(id, calificacion, comentario, id_fecha, id_producto, id_cliente) VALUES(0, 3.0, 'Producto defectuoso.', 2, 25, 13);

INSERT INTO resenias(id, calificacion, comentario, id_fecha, id_producto, id_cliente) VALUES(0, 8.0, 'Buena relación calidad-precio.', 9, 15, 7);
INSERT INTO resenias(id, calificacion, comentario, id_fecha, id_producto, id_cliente) VALUES(0, 5.5, 'Regular, tiene puntos a mejorar.', 5, 6, 1);
INSERT INTO resenias(id, calificacion, comentario, id_fecha, id_producto, id_cliente) VALUES(0, 9.8, 'Me encantó, lo volvería a comprar.', 8, 18, 10);
INSERT INTO resenias(id, calificacion, comentario, id_fecha, id_producto, id_cliente) VALUES(0, 6.0, 'Envío muy lento.', 10, 22, 4);
INSERT INTO resenias(id, calificacion, comentario, id_fecha, id_producto, id_cliente) VALUES(0, 4.8, 'Decente por el precio.', 4, 30, 14);
```

Consultas y Procedimientos Almacenados

6. Script con consultas

    a. Listar productos por categoría, ordenados por precio.

```sql
SELECT p.nombre AS producto, c.nombre AS categoria, p.precio AS precio -- Nombre de columnas
FROM productos p -- Tabla principal
INNER JOIN categorias c ON p.id_categoria = c.id -- Unicamente las categorias que se encuentren enlazadas con los productos actuales
ORDER BY categoria ASC, precio ASC; -- Ordenar por orden alfabetico para las categorias y el precio de forma ascendente (de menor a mayor precio)
```

```
mysql> SELECT p.nombre AS producto, c.nombre AS categoria, p.precio AS precio FROM productos p INNER JOIN categorias c ON p.id_categoria = c.id ORDER BY categoria ASC, precio ASC;
+------------------+---------------------+---------+
| producto         | categoria           | precio  |
+------------------+---------------------+---------+
| termo            | accesorios          |  230.00 |
| gorra            | accesorios          |  560.00 |
| sabritas         | alimentos y bebidas |   27.00 |
| zanahoria        | alimentos y bebidas |   28.00 |
| coca-cola        | alimentos y bebidas |   32.00 |
| manzana          | alimentos y bebidas |   37.00 |
| mochila Nike     | bolsas              | 2300.00 |
| plancha          | electrodomesticos   |  450.00 |
| cafetera         | electrodomesticos   | 2300.00 |
| estufa           | electrodomesticos   | 3500.00 |
| refrigerador     | electrodomesticos   | 5000.00 |
| cable hdmi       | electronica         |  110.00 |
| mouse            | electronica         |  350.00 |
| teclado          | electronica         |  800.00 |
| apple watch      | electronica         | 2800.00 |
| audifonos        | electronica         | 3200.00 |
| xbox one         | electronica         | 4700.00 |
| laptop           | electronica         | 7400.00 |
| pantalla Samsung | electronica         | 8200.00 |
| iPhone 13        | electronica         | 9600.00 |
| cepillo dental   | higiene             |   34.00 |
| desodorante      | higiene             |   46.00 |
| perfume          | higiene             |   78.00 |
| jabon de manos   | limpieza            |   35.00 |
| pinol            | limpieza            |   50.00 |
| suavitel         | limpieza            |   60.00 |
| escoba           | limpieza            |  120.00 |
| calcetines       | ropa                |   80.00 |
| playera          | ropa                |  300.00 |
| pantalon         | ropa                |  750.00 |
+------------------+---------------------+---------+
30 rows in set (0.00 sec)
```

b. Mostrar clientes con pedidos pendientes y total de compras.

```sql
SELECT cl.nombre AS cliente, -- Nombre de columna
COUNT(DISTINCT CASE WHEN p.estado = 'pendiente' THEN p.id END) AS pendientes, -- Cuenta unicamente los pedidos con estado pendiente
SUM(d.cantidad * d.precio_unitario) AS total_compras -- Suma las compras de cada cliente, multiplicando la cantidad por el precio unitario de cada pedido hecho por el cliente
FROM clientes cl -- Tabla principal
INNER JOIN pedidos p ON cl.id = p.id_cliente -- Unicamente los pedidos que tengan un cliente enlazado
LEFT JOIN detalles_pedidos d ON p.id = d.id_pedido -- Unicamente los pedidos que tengan productos disponibles
GROUP BY cl.nombre -- Agrupa por nombre de clientes ya que se usan funciones COUNT y SUM y es necesario
HAVING pendientes > 0; -- Se muestran solo los pedidos con más de 0 pedidos pendientes
```

```
mysql> SELECT cl.nombre AS cliente, COUNT(DISTINCT CASE WHEN p.estado = 'pendiente' THEN p.id END) AS pendientes, SUM(d.cantidad * d.precio_unitario
) AS total_compras FROM clientes cl INNER JOIN pedidos p ON cl.id = p.id_cliente LEFT JOIN detalles_pedidos d ON p.id = d.id_pedido GROUP BY cl.nomb
re HAVING pendientes > 0;
+-----------+------------+----------------+
| cliente   | pendientes | total_compras |
+-----------+------------+----------------+
| Alberto   |          1 |       2400.00 |
| Aldo      |          1 |       2280.00 |
| Catherine |          1 |     146750.00 |
| Daniel    |          1 |        200.00 |
| Eder      |          1 |      48068.00 |
| Fernanda  |          1 |     286252.00 |
| Irving    |          3 |      35334.00 |
| Joshua    |          2 |      30091.00 |
| Marlon    |          1 |       5250.00 |
| Paola     |          2 |       1210.00 |
+-----------+------------+----------------+
10 rows in set (0.00 sec)
```

c. Reporte de los 5 productos con mejor calificación promedio en reseñas.

```sql
SELECT p.nombre AS producto, AVG(r.calificacion) AS prom_calif -- Nombre de columnas
FROM productos p -- Tabla principal
INNER JOIN resenias r ON p.id = r.id_producto -- Solo las reseñas con productos enlazados
GROUP BY producto -- Agrupa por producto ya que se usa funcion AVG para el promedio de las calificaciones
ORDER BY prom_calif DESC -- Se ordena por el promedio de calificaciones de forma descendente
LIMIT 5; -- Solo aparecen los primeros 5 productos
```

```
mysql> SELECT p.nombre AS producto, AVG(r.calificacion) AS prom_calif, r.comentario AS comentario, f_r.fecha AS fecha_resenia, cl.nombre AS nombre_cliente FROM resenias r INNER JOIN producto
s p ON r.id_producto = p.id INNER JOIN fecha_resenias f_r ON r.id_fecha = f_r.id INNER JOIN clientes cl ON r.id_cliente = cl.id GROUP BY producto, comentario, fecha, nombre_cliente ORDER BY
prom_calif DESC LIMIT 5;
+-------------+-----------------+-----------------------------------+---------------+----------------+
| producto    | prom_calif      | comentario                        | fecha_resenia | nombre_cliente |
+-------------+-----------------+-----------------------------------+---------------+----------------+
| zanahoria   | 9.800000190734863 | Me encantó, lo volvería a comprar. | 2025-06-08    | Eder           |
| mouse       |               9 | Excelente, superó mis expectativas. | 2025-06-14    | Fernanda       |
| termo       |             8.5 | Muy buen producto, llegó a tiempo. | 2025-06-29    | Pablo          |
| apple watch |               8 | Buena relación calidad-precio.    | 2025-06-21    | Esperanza      |
| cable hdmi  |               7 | Cumple con lo básico.             | 2025-06-12    | Mariel         |
+-------------+-----------------+-----------------------------------+---------------+----------------+
5 rows in set (0.00 sec)
```

d. Procedimientos

i. Registrar un nuevo pedido: Después de usar el procedimiento para insertar nuevos pedidos, se probaron los mensajes de alerta para dos casos, si la cantidad solicitada era mayor a la disponible en el stock y si la cantidad de pedidos era alcanzada.

```
CREATE PROCEDURE sp_nuevo_pedido(
    IN sp_id_cliente INT,
    IN sp_cantidad INT,
    IN sp_id_producto INT,
    IN sp_id_fecha_pedido INT
)
BEGIN
    IF(
        SELECT stock FROM productos WHERE id = sp_id_producto -- Si existe el producto con el id indicado que tenga stock mayor a la cantidad
    ) > sp_cantidad
    THEN
        IF(
            SELECT COUNT(*) FROM pedidos WHERE id_cliente = sp_id_cliente -- Si el cliente ha hecho más de 5 pedidos
        ) >= 5
        THEN -- Manda alerta de límite de pedidos
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Límite de pedidos alcanzado';
        ELSE -- Si no alcanzó el límite de pedidos, inserta y pasa el pedido con los datos
            INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, sp_id_fecha_pedido, 'pendiente', sp_id_cliente);
        END IF;
    ELSE -- Si no cumple la condición, se ejecuta la alerta
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Stock inferior a cantidad solicitada';
    END IF;
END;
```

```
USE tienda_online_proyecto;
CALL sp_nuevo_pedido(1, 2, 1, 1);
CALL sp_nuevo_pedido(1, 1000, 1, 1)
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	15:08:28	USE tienda_online_proyecto	0 row(s) affected	0.000 sec
2	15:08:28	CALL sp_nuevo_pedido(1, 1, 1, 1)	1 row(s) affected	0.844 sec
3	15:09:49	USE tienda_online_proyecto	0 row(s) affected	0.000 sec
4	15:09:49	CALL sp_nuevo_pedido(1, 1000, 1, 1)	Error Code: 1644. Stock inferior a cantidad solicitada	1.656 sec
5	15:11:38	USE tienda_online_proyecto	0 row(s) affected	0.000 sec
6	15:11:38	CALL sp_nuevo_pedido(1, 2, 1, 1)	1 row(s) affected	0.344 sec
7	15:11:38	CALL sp_nuevo_pedido(1, 1000, 1, 1)	Error Code: 1644. Stock inferior a cantidad solicitada	0.000 sec
8	15:11:52	USE tienda_online_proyecto	0 row(s) affected	0.000 sec
9	15:11:52	CALL sp_nuevo_pedido(1, 2, 1, 1)	1 row(s) affected	0.078 sec
10	15:11:52	CALL sp_nuevo_pedido(1, 1000, 1, 1)	Error Code: 1644. Stock inferior a cantidad solicitada	0.000 sec
11	15:11:57	USE tienda_online_proyecto	0 row(s) affected	0.000 sec
12	15:11:57	CALL sp_nuevo_pedido(1, 2, 1, 1)	Error Code: 1644. Límite de pedidos alcanzado	0.000 sec

ii. Registrar una reseña: Se registra la reseña solo cuando hay una coincidencia entre el cliente y el producto pedido, si no existe un producto hecho por el cliente, manda la alerta, si se cumple, entonces hace la reseña.

```
CREATE PROCEDURE sp_nuevo_pedido(
    IN sp_id_cliente INT,
    IN sp_cantidad INT,
    IN sp_id_producto INT,
    IN sp_id_fecha_pedido INT
)
BEGIN
    IF(
        SELECT stock FROM productos WHERE id = sp_id_producto -- Si existe el producto con el id indicado que tenga stock mayor a la cantidad
    ) > sp_cantidad
    THEN
        IF(
            SELECT COUNT(*) FROM pedidos WHERE id_cliente = sp_id_cliente -- Si el cliente ha hecho más de 5 pedidos
        ) >= 5
        THEN -- Manda alerta de límite de pedidos
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Límite de pedidos alcanzado';
        ELSE -- Si no alcanzó el límite de pedidos, inserta y pasa el pedido con los datos
            INSERT INTO pedidos(id, id_fecha_pedido, estado, id_cliente) VALUES(0, sp_id_fecha_pedido, 'pendiente', sp_id_cliente);
        END IF;
    ELSE -- Si no cumple la condición, se ejecuta la alerta
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Stock inferior a cantidad solicitada';
    END IF;
END;
```

```
USE tienda_online_proyecto;

CALL sp_registrar_resenia(1, 14, 5.28, 'EL producto me gustó pero lo esperaba diferente en cuanto al color', 1);
CALL sp_registrar_resenia(1, 1, 5.28, 'No me gustó el producto, porque yo no lo pedí', 1)

```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	15:35:51	USE tienda_online_proyecto	0 row(s) affected	0.000 sec
2	15:35:51	CALL sp_registrar_resenia(1, 14, 5.28, 'EL producto me gustó pero lo esperaba diferente e...	1 row(s) affected	0.031 sec
3	15:35:51	CALL sp_registrar_resenia(1, 1, 5.28, 'No me gustó el producto, porque yo no lo pedí', 1)	Error Code: 1644. No existe un pedido del producto por ese cliente	0.000 sec

iii. Actualizar el stock de producto: Se inserta el id donde se encuentran los detalles de un pedido, para extraer la cantidad solicitada en el pedido y restarla al stock actual del producto, si no existe el pedido manda una alerta.

```sql
CREATE PROCEDURE sp_actualizar_stock(
    IN sp_id_detalles_pedidos INT
)
BEGIN
    -- Declaracion de variables para guardar la cantidad que se pidio y el id del producto
    DECLARE v_cantidad INT;
    DECLARE v_id_producto INT;

    IF EXISTS(
        SELECT cantidad FROM detalles_pedidos WHERE id = sp_id_detalles_pedidos -- Si existen los detalles del pedido con el mismo id
    )
    THEN
        SELECT cantidad, id_producto -- Selecciona las columnas que contienen los datos
        INTO v_cantidad, v_id_producto -- Se almacenan los datos en las variables
        FROM detalles_pedidos -- Tabla de donde se extraen los datos
        WHERE id = sp_id_detalles_pedidos; -- Unicamente del id dado

        UPDATE productos SET stock = stock - v_cantidad WHERE id = v_id_producto; -- Se actualiza el stock del producto
    ELSE -- Si no existe el pedido manda alerta
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Pedido no encontrado';
    END IF;
END;
```

```sql
USE tienda_online_proyecto;

CALL sp_actualizar_stock(1);
CALL sp_actualizar_stock(26)
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	15:44:10	USE tienda_online_proyecto	0 row(s) affected	0.000 sec
2	15:44:10	CALL sp_actualizar_stock(1)	1 row(s) affected	0.015 sec
3	15:49:18	USE tienda_online_proyecto	0 row(s) affected	0.000 sec
4	15:49:18	CALL sp_actualizar_stock(1)	1 row(s) affected	0.141 sec
5	15:49:18	CALL sp_actualizar_stock(26)	Error Code: 1644. Pedido no encontrado	0.000 sec

iv. Cambiar el estado de pedido: Recibe el id de un pedido realizado, si existe cambia el estado, si no, manda alerta.

```sql
USE tienda_online_proyecto;

DELIMITER $$

CREATE PROCEDURE sp_cambiar_estado(
    IN sp_id_pedido INT
)
BEGIN
    IF EXISTS(
        SELECT * FROM pedidos WHERE id = sp_id_pedido -- Si existe el pedido con el id indicado
    )
    THEN
        UPDATE pedidos SET estado = 'enviado' WHERE id = sp_id_pedido; -- Actualiza la columna de estado del pedido a 'enviado'
    ELSE -- Si no existe el pedido, manda alerta
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Pedido no existe';
    END IF;
END;

END $$
DELIMITER ;
```

```sql
USE tienda_online_proyecto;

CALL sp_cambiar_estado(19);
CALL sp_cambiar_estado(21);
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	15:55:06	USE tienda_online_proyecto	0 row(s) affected	0.000 sec
2	15:55:06	CALL sp_cambiar_estado(19)	1 row(s) affected	1.656 sec
3	15:55:08	CALL sp_cambiar_estado(21)	Error Code: 1644. Pedido no existe	0.000 sec

v. Eliminar reseñas: Se da el id del producto para buscar las reseñas asociadas, si se encuentran, se eliminan para actualizar la tabla de reseñas, si no, se manda la alerta.

```sql
USE tienda_online_proyecto;

DELIMITER $$

CREATE PROCEDURE sp_eliminar_resenia(
    IN sp_id_producto INT
)
BEGIN
    DECLARE id_resenia INT; -- variable para guardar el id de la reseña donde este el producto

    IF EXISTS(
        SELECT * FROM resenias WHERE resenias.id_producto = sp_id_producto -- Si existe una reseña que tenga el id del producto
    )
    THEN
        DELETE FROM resenias WHERE id_producto = sp_id_producto; -- Se elimina la reseña con el id del producto
    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No existen resenias de ese producto';
    END IF;
END;

END $$
DELIMITER ;
```

```sql
USE tienda_online_proyecto;

CALL sp_eliminar_resenia(20);
CALL sp_eliminar_resenia(1)
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	16:09:49	USE tienda_online_proyecto	0 row(s) affected	0.000 sec
2	16:09:49	CALL sp_eliminar_resenia(20)	2 row(s) affected	0.312 sec
3	16:09:49	CALL sp_eliminar_resenia(1)	Error Code: 1644. No existen resenias de ese producto	0.000 sec

vi. Agregar un nuevo producto: Si no encuentra el mismo producto en la categoría, lo añade a la tabla, si existe un duplicado, manda alerta de duplicado, si no existe la categoría elegida, manda un aviso.

```sql
CREATE PROCEDURE sp_agregar_producto(
    IN sp_nombre VARCHAR(50),
    IN sp_descripcion VARCHAR(100),
    IN sp_precio DECIMAL(10,2),
    IN sp_stock INT,
    IN sp_id_categoria INT
)
BEGIN
    IF NOT EXISTS(
        SELECT * FROM productos
        WHERE nombre = sp_nombre AND id_categoria = sp_id_categoria -- Si no existe ya un producto con el mismo nombre y la misma categoria
    )
    THEN
        IF EXISTS(
            SELECT * FROM categorias WHERE id = sp_id_categoria -- Si existe el id de la categoria deseada
        )
        THEN
            INSERT INTO productos(id, nombre, descripcion, precio, stock, id_categoria)
            VALUES(0, sp_nombre, sp_descripcion, sp_precio, sp_stock, sp_id_categoria); -- Se inserta el nuevo producto
        ELSE
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'No existe esa categoria'; -- Si no existe la categoria, se manda el aviso
        END IF;
    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Ya se encuentra disponible ese producto en la categoria'; -- Si ya existe un duplicado, manda alerta
    END IF;
END;
```

```sql
USE tienda_online_proyecto;

CALL sp_agregar_producto('crema', 'prueba para producto nuevo', 50, 5, 1);
CALL sp_agregar_producto('eucalipto', 'prueba para producto con categoria inexistente', 50, 5, 10);
CALL sp_agregar_producto('suavitel', 'prueba para producto y categoria duplicado', 50, 5, 1);
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	16:49:00	USE tienda_online_proyecto	0 row(s) affected	0.000 sec
2	16:49:00	CALL sp_agregar_producto('crema', 'prueba para producto nuevo', 50, 5, 1)	1 row(s) affected	0.203 sec
3	16:49:00	CALL sp_agregar_producto('eucalipto', 'prueba para producto con categoria inexistente', 50, 5, 10)	Error Code: 1644. No existe esa categoria	0.000 sec
4	16:49:30	USE tienda_online_proyecto	0 row(s) affected	0.000 sec
5	16:49:30	CALL sp_agregar_producto('suavitel', 'prueba para producto y categoria duplicado', 50, 5, 1)	Error Code: 1644. Ya se encuentra disponible ese producto en la categoria	0.000 sec

vii. Actualizar la información de cliente: Se mandan los datos nuevos del cliente, si existe por el nombre se actualizan, si no manda un aviso.

```sql
CREATE PROCEDURE sp_actualizar_cliente(
    IN sp_nombre VARCHAR(50),
    IN sp_correo VARCHAR(50),
    IN sp_telefono VARCHAR(10),
    IN sp_direccion VARCHAR(50)
)
BEGIN
    DECLARE v_id_cliente INT; -- variable para poder actualizar el cliente

    IF EXISTS(
        SELECT * FROM clientes WHERE nombre = sp_nombre -- Si existe el cliente
    )
    THEN
        SELECT id
        INTO v_id_cliente -- Guarda el id del cliente en una variable para actualizar sus datos
        FROM clientes
        WHERE nombre = sp_nombre;

        UPDATE clientes -- Se actualizan todos sus datos con los nuevos
        SET correo = sp_correo, telefono = sp_telefono, direccion = sp_direccion
        WHERE id = v_id_cliente;
    ELSE -- Si no existe el cliente, manda un aviso
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cliente no existe';
    END IF;
END;
```

```sql
USE tienda_online_proyecto;

CALL sp_actualizar_cliente('joshua', 'joshua.cabriales@correo.com', '5539608499', 'santa fe');
CALL sp_actualizar_cliente('balu', 'balu@correo.com', '1234567', 'casa')
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	17:02:26	USE tienda_online_proyecto	0 row(s) affected	0.000 sec
2	17:02:26	CALL sp_actualizar_cliente('joshua', 'joshua.cabriales@correo.com', '5539608499', 'santa fe')	1 row(s) affected	0.187 sec
3	17:02:26	CALL sp_actualizar_cliente('balu', 'balu@correo.com', '1234567', 'casa')	Error Code: 1644. Cliente no existe	0.000 sec

viii. Generar un reporte de productos con stock bajo: Se muestran los productos con todos sus datos, pero solo aquellos con un stock por debajo de 5, con su respectiva categoría.

```sql
USE tienda_online_proyecto;

DELIMITER $$

CREATE PROCEDURE sp_reporte_stock()
BEGIN
    SELECT p.nombre AS producto, p.descripcion AS descripcion, p.precio AS precio, p.stock AS stock, c.nombre AS categoria
    FROM productos p -- Tabla principal para datos de los productos
    INNER JOIN categorias c ON p.id_categoria = c.id -- Union con la categoria para mostrar su nombre
    WHERE stock <= 5
    ORDER BY stock ASC; -- Solo los productos con stock por debajo de 5
END;

END $$
DELIMITER ;
```

```sql
USE tienda_online_proyecto;

CALL sp_reporte_stock()
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

producto	descripcion	precio	stock	categoria
iPhone 13	dispositivo celular de Apple	9600.00	3	electronica
teclado	periferico para escribir en computadora	800.00	4	electronica
crema	prueba para producto nuevo	50.00	5	limpieza

Validación y Optimización

7. Explicación de índices creados y su impacto
    a. Índice para clientes (nombre): En casos donde se decida buscar los pedidos en base a un cliente por su nombre o los detalles de los pedidos de este.

```
mysql> EXPLAIN SELECT * FROM clientes WHERE nombre = 'Joshua';
+----+-------------+----------+------------+------+---------------+------------+---------+-------+------+----------+-------+
| id | select_type | table    | partitions | type | possible_keys | key        | key_len | ref   | rows | filtered | Extra |
+----+-------------+----------+------------+------+---------------+------------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | clientes | NULL       | ref  | idx_nombre    | idx_nombre | 203     | const |    1 |   100.00 | NULL  |
+----+-------------+----------+------------+------+---------------+------------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

    b. índice para categorías (nombre): En casos donde se decida filtrar por el nombre de cierta categoría.

```
mysql> EXPLAIN SELECT * FROM categorias WHERE nombre = 'electronica';
+----+-------------+-----------+------------+------+---------------+------------+---------+-------+------+----------+-------+
| id | select_type | table     | partitions | type | possible_keys | key        | key_len | ref   | rows | filtered | Extra |
+----+-------------+-----------+------------+------+---------------+------------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | categorias | NULL       | ref  | idx_nombre    | idx_nombre | 203     | const |    1 |   100.00 | NULL  |
+----+-------------+-----------+------------+------+---------------+------------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

    c. Índice para productos (nombre): En casos donde se requiera hacer una búsqueda relacionando datos con el nombre de cierto producto.

```
mysql> EXPLAIN SELECT * FROM productos WHERE nombre = 'suavitel';
+----+-------------+-----------+------------+------+---------------+------------+---------+-------+------+----------+-------+
| id | select_type | table     | partitions | type | possible_keys | key        | key_len | ref   | rows | filtered | Extra |
+----+-------------+-----------+------------+------+---------------+------------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | productos | NULL       | ref  | idx_nombre    | idx_nombre | 203     | const |    1 |   100.00 | NULL  |
+----+-------------+-----------+------------+------+---------------+------------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

    d. Índice para productos (stock): En casos donde sea necesario ubicar ciertos productos con stock bajo o los productos que tengan más stock para realizar acciones operativas correspondientes.

```
mysql> EXPLAIN SELECT * FROM productos WHERE stock = 60;
+----+-------------+-----------+------------+------+---------------+-----------+---------+-------+------+----------+-------+
| id | select_type | table     | partitions | type | possible_keys | key       | key_len | ref   | rows | filtered | Extra |
+----+-------------+-----------+------------+------+---------------+-----------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | productos | NULL       | ref  | idx_stock     | idx_stock | 5       | const |    2 |   100.00 | NULL  |
+----+-------------+-----------+------------+------+---------------+-----------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

    e. Índice para pedidos (id_cliente): En casos donde se ocupe identificar los pedidos de un cliente en específico.

```
mysql> EXPLAIN SELECT * FROM pedidos WHERE id_cliente = 1;
+----+-------------+---------+------------+------+---------------+----------------+---------+-------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys | key            | key_len | ref   | rows | filtered | Extra |
+----+-------------+---------+------------+------+---------------+----------------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | pedidos | NULL       | ref  | idx_id_cliente | idx_id_cliente | 5       | const |    2 |   100.00 | NULL  |
+----+-------------+---------+------------+------+---------------+----------------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

f. Índice para pedidos (estado): En casos donde sea necesario identificar los pedidos para cambiar su estado o para saber cuales ya fueron enviados y cuales aún están pendientes.

```
mysql> EXPLAIN SELECT * FROM pedidos WHERE estado = 'pendiente';
+----+-------------+---------+------------+------+---------------+------------+---------+-------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys | key        | key_len | ref   | rows | filtered | Extra |
+----+-------------+---------+------------+------+---------------+------------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | pedidos | NULL       | ref  | idx_estado    | idx_estado | 83      | const |   14 |   100.00 | NULL  |
+----+-------------+---------+------------+------+---------------+------------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

8. Validación

Las validaciones con procedimientos se encuentran en el área de procedimientos para visualizar la diferencia cuando se realizaban correctamente y cuando estos fallaban por sus condiciones.

Se añadieron otros índices, además de los propuestos ya que estos serán usados automáticamente en los procedimientos, lo cual agilizara los procesos de búsqueda y filtrado, como un índice para el estado de pedidos o índice para el stock de los productos. Además de incluir condiciones en los procedimientos que evitaran procesos innecesarios o inútiles al momento de llamar un procedimiento en particular, el cual podría realizar acciones o actualizaciones y cargar la memoria del equipo sin hacer verificaciones previas. También se dividieron las fechas en tablas de pedidos y reseñas para tener la 3NF y para evitar duplicidad en los datos de dichas tablas.

Conclusión

Se consiguió crear un sistema mediante una base de datos que almacena la información de la tienda, datos como los clientes, sus pedidos, además de toda la información relacionada a sus productos como el precio, stock disponible, descripción y categoría de estos para segmentar. Se diseñó el diagrama ER para saber los enlaces y vínculos que tendría la información de la BD y se realizó un esquema en 3NF para no tener duplicidad de datos y también poder evitar que los datos dependan de una sola llave primaria, teniendo una mayor seguridad y protección para la información.

Se logró implementar correctamente los procedimientos para agilizar ciertos procesos realizados frecuentemente en la BD como lo son insertar nuevos productos, actualizar información de clientes o pedidos y eliminar o solicitar ciertos datos para trabajar con ellos posteriormente. Además de optimizar las búsquedas de datos y el rendimiento de los filtrados a través de los índices, los cuales ayudan a hacer búsquedas más rápidas y enfocadas a escenarios precisos.

Todo esto fue importante ya que se logra tener un orden en la BD, también se consigue una validación de información para evitar datos duplicados o inserciones incompletas que puedan afectar las tablas y la información que ya contienen.

Algo que se observa para mejorar, es implementar la BD junto a un sistema Web para que su manejo sea más sencillo, con esto además se lograría hacer escalable, ya que la BD es capaz de soportar estas acciones y se podría usar con una interfaz amigable para usuarios promedios para posteriormente dar a conocer los productos y que los usuarios puedan crear sus perfiles de compra y agregar los productos para comprar sin necesitar empleados directos de la tienda.

Lo que fue más complicado, al menos en la lógica, fueron los procedimientos ya que se necesita poder observar todas las posibilidades al momento de ejecutar un procedimiento para saber todo lo necesario para que este lograra trabajar correctamente y como se esperaba.

Como aprendizaje, ahora se pueden considerar mayores posibilidades al momento de diseñar e implementar un sistema de BD para empresas o compañías grandes que necesitan procesar y trabajar con grandes cantidades de datos, a mantener la información separada, poder validar para evitar cargas innecesarias o problemáticas en las tablas, además de poder hacer más eficientes tareas que se llevan a cabo de forma continua en el sistema para agilizar los procesos.

Enlace a repositorio: https://github.com/JoshuaCabriales28SS/ProyectoBD