Casa abierta al tiempo

UNIVERSIDAD AUTÓNOMA METROPOLITANA

UNIDAD CUAJIMALPA

LICENCIATURA EN TECNOLOGÍAS Y SISTEMAS DE INFORMACIÓN.

PROYECTO DE BASES DE DATOS:

SISTEMA DE GESTION DE TIENDA ONLINE

UNIDAD DE ENSEÑANZA

BASES DE DATOS

Dr. Guillermo Monroy

Alumno:
Luis Antonio Salinas Mata

1. Descripción del Problema

Se nos pide la creación de una tienda online suponiendo que los productos electrónicos necesita un sistema para gestionar sus operaciones incluyendo:

- productos.
- clientes.
- pedidos (con detalles y estado).
- reseñas de productos.
- categorías.

Requisitos funcionales:

- **Gestión de Productos:** Nombre, descripción, precio, stock (no negativo), categoría.
- **Gestión de Clientes:** Nombre, correo (único), dirección, número de teléfono.
- **Pedidos:** Fecha, productos incluidos.
- **Reseñas:** Calificación (1–5 estrellas), comentario. Solo por clientes que hayan comprado el producto.
- **Categorías:** Clasificación de productos (ej. teléfonos, laptops, accesorios).

Se decidió agregar dos tablas más:

- **detalle_pedido:** cantidad productos, precio total.
- **estado_pedido:** estado (pendiente, enviado, entregado)

Estas tablas adicionales permiten cumplir con la **Tercera Forma Normal (3NF)**

Restricciones clave:

- Máximo 5 pedidos *pendientes* por cliente.
- El stock de productos no debe ser negativo.
- Las reseñas solo pueden ser hechas por clientes que compraron el producto.

Objetivo General:

Diseñar una **base de datos relacional normalizada**, implementar consultas y al menos **8 procedimientos almacenados**, optimizarla con índices y validarla con datos de prueba.

2. Requisitos del Proyecto:

- Diseñar una base de datos relacional con al menos 5 tablas (Productos, Clientes, Pedidos, Reseñas, Categorías).

Entregables:

- Diagrama Entidad-Relación (ER) que modele entidades y relaciones.
- Esquema en tercera forma normal (3NF), con justificación de normalización.
- Identificación de claves primarias, foráneas y candidatas.

Análisis para cumplir los criterios de entrega:

Para cumplir con el modelo Entidad-Relación (ER) tenemos que tomar en cuenta lo siguiente:

- **Entidades**: cajas (categoria, productos, clientes, pedidos, detalle_pedido, reseñas, estado_pedido).
- **Atributos**: cada campo de las tablas.
- **Relaciones**: líneas entre entidades.
- **Cardinalidad**: indicación 1:N, N:N, etc.

Ahora para ver que cumpla con la tercera Forma normal (3NF) primero debemos analizar si cumple con la 1NF:

- LLave primaria en cada una de las tablas
- No debe haber grupos repetitivos de datos dentro de una misma fila, y todos sus atributos son atómicos. Por ejemplo, un detalle_pedido tiene una cantidad y un "precio_total" especificos para una línea de pedido, no una lista de productos.

Con esto se asegura que cumpla con la "1NF" ahora debemos asegurar que cumpla con las "2NF":

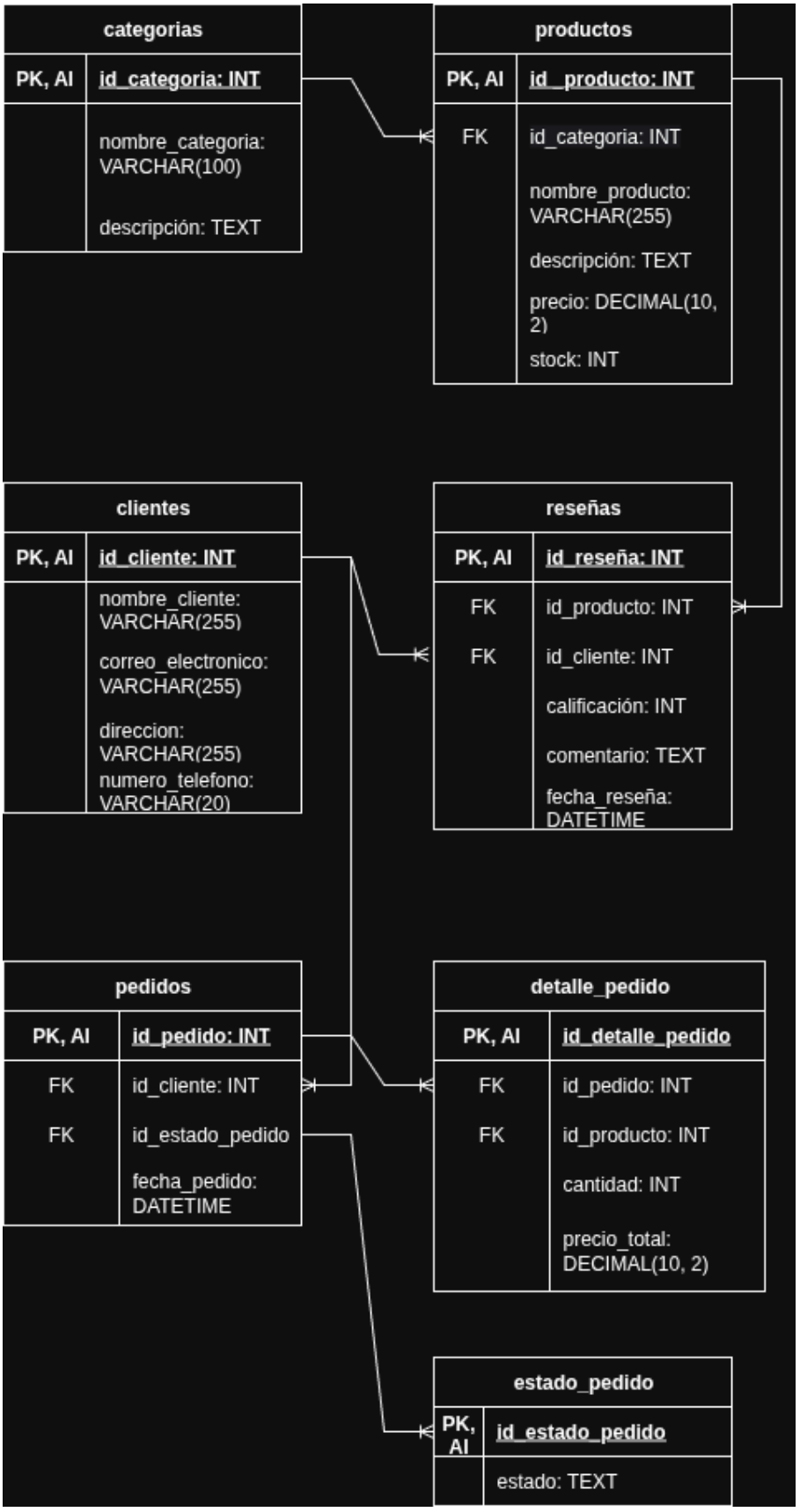
- Debe estar en "1NF" y todos los atributos no clave deben depender completamente de la clave primaria.

Y por último requisito para que cumpla con la "3NF":

- Debe estar en "2NF" y no debe haber dependencias transitivas. (Una dependencia transitiva ocurre cuando un atributo no clave depende de otro atributo no clave, en lugar de depender directamente de la clave primaria.)

En el diagrama que muestro a continuación se ve claramente que en todas las tablas, cada atributo no clave dependen directamente de la clave primaria de esa tabla, y no de otro atributo no clave dentro de la misma tabla. Por ejemplo, en Productos, nombre, precio, stock dependen directamente de "id_producto", no hay ningún atributo que dependa de nombre en lugar de "id_producto". Así cumpliendo con todos los requisitos del entregable.

Diagrama base de datos relacional de tienda online



3. Implementación de la Base de Datos

Se utilizó **MySQL** como SGBD.

Entregables:

- Script SQL para crear tablas, claves y restricciones.
- Al menos 3 índices (productos por nombre, por categoría, pedidos por cliente).
- Script SQL para insertar datos de prueba.

Configuración Inicial Generada Automáticamente:

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
```

- Evita que un campo con AUTO_INCREMENT pueda tener valor 0 automáticamente.

Una vez terminadas las configuraciones iniciales empezamos con la creación de las tablas necesarias.

Tablas:

Creación de la tabla categorías que incluye lo siguiente:

- id_categoria: identificador único de categoría.
- nombre_categoria: aquí se coloca el nombre de la categoría
- descripcion: aquí se agrega una breve descripción de la categoría.

```
CREATE TABLE categorias (
  id_categoria int(11) NOT NULL AUTO_INCREMENT,
  nombre_categoria varchar(100) NOT NULL,
  descripcion text NOT NULL,
  PRIMARY KEY (id_categoria)
);
```

Creación de la tabla clientes que incluye lo siguiente:

- id_clientes: identificador único de clientes.
- nombre_cliente: aquí es donde se coloca el nombre del cliente.
- correo_electronico: aquí se coloca el correo electrónico del cliente.
- direccion: aqui se coloca la dirección del cliente.
- numero_telefono: aquí se coloca el número de teléfono del cliente.

```
CREATE TABLE clientes (
  id_clientes int(11) NOT NULL AUTO_INCREMENT,
  nombre_cliente varchar(255) NOT NULL,
  correo_electronico varchar(255) NOT NULL UNIQUE,
  direccion varchar(255) NOT NULL,
  numero_telefono varchar(20) NOT NULL,
  PRIMARY KEY (id_clientes)
);
```

Creación de la tabla “detalle_pedido” incluye lo siguiente:

- id_detalle_pedido: identificador único de la tabla “detalle_pedido”.
- id_pedido: llave foránea enlazada a la tabla "pedidos".
- id_producto: llave foranea enlazada a la tabla "productos".
- cantidad: aquí se inserta la cantidad productos que hay en el pedido.
- precio_total: aquí se inserta el precio final del pedido.

NOTA: Esta tabla tiene un enlace a la tabla "pedidos" y "productos"

```
CREATE TABLE detalle_pedido (
  id_detalle_pedido int(11) NOT NULL AUTO_INCREMENT,
  id_pedido int(11) NOT NULL,
  id_producto int(11) NOT NULL,
  cantidad int(11) NOT NULL,
  precio_total decimal(10,2) NOT NULL,
  PRIMARY KEY (id_detalle_pedido),
  KEY fk_detalle_pedido_pedido (id_pedido),
  KEY fk_detalle_pedido_producto (id_producto),
  CONSTRAINT fk_detalle_pedido_pedido FOREIGN KEY (id_pedido) REFERENCES pedidos (id_pedido) ON UPDATE CASCADE,
  CONSTRAINT fk_detalle_pedido_producto FOREIGN KEY (id_producto) REFERENCES productos (id_producto) ON UPDATE CASCAD
);
```

Creación de la tabla estado_pedido que incluye lo siguiente:

- id_estado_pedido: identificador único de la tabla pedido.
- estado: aquí contendrá el estado del pedido(enviado, cancelado, en proceso).

```
CREATE TABLE estado_pedido (
  id_estado_pedido int(11) NOT NULL AUTO_INCREMENT,
  estado varchar(50) NOT NULL,
  PRIMARY KEY (id_estado_pedido)
);
```

Creación de la tabla pedidos que incluye lo siguiente:

- id_pedido: identificador único de la tabla pedidos.
- id_cliente: llave foránea enlazada a la tabla "clientes".
- id_estado_pedido: llave foránea enlazada a la tabla "estado_pedido".
- fecha_pedido: aquí se guardará la fecha del pedido.

```
CREATE TABLE pedidos (
  id_pedido int(11) NOT NULL AUTO_INCREMENT,
  id_cliente int(11) NOT NULL,
  id_estado_pedido int(11) NOT NULL,
  fecha_pedido datetime NOT NULL,
  PRIMARY KEY (id_pedido),
  KEY fk_pedidos_clientes (id_cliente),
  KEY fk_pedidos_estado (id_estado_pedido),
  CONSTRAINT fk_pedidos_clientes FOREIGN KEY (id_cliente) REFERENCES clientes (id_clientes) ON UPDATE CASCADE,
  CONSTRAINT fk_pedidos_estado FOREIGN KEY (id_estado_pedido) REFERENCES estado_pedido (id_estado_pedido) ON UPDATE C
);
```

Creación de la tabla productos

- id_producto: identificador único de la tabla productos.
- id_categoria: llave foránea enlazada a la tabla "categorias".
- nombre_producto: aquí se colocará el nombre del producto.
- descripcion: aquí se colocara una breve descripción del producto.
- precio: aquí se colocará el precio del producto.
- stock: aquí se colocará el stock disponible del producto.

```
CREATE TABLE productos (
  id_producto int(11) NOT NULL AUTO_INCREMENT,
  id_categoria int(11) NOT NULL,
  nombre_producto varchar(255) NOT NULL,
  descripcion text NOT NULL,
  precio decimal(10,2) NOT NULL,
  stock int(11) NOT NULL,
  PRIMARY KEY (id_producto),
  KEY fk_productos_categorias (id_categoria),
  CONSTRAINT fk_productos_categorias FOREIGN KEY (id_categoria) REFERENCES categorias (id_categoria) ON UPDATE CASCAD
);
```

Creación de la tabla resenas

- id_resena: identificador único de la tabla resena.
- id_producto: llave foránea enlazada a la tabla "productos".
- id_cliente: llave foránea enlazada a la tabla "cliente".
- calificacion: aqui se le colocara la reseña con calificacion del 1 al 5.
- comentario: aquí se dejará un breve comentario para la reseña.
- fecha_resena: aquí se colocara la fecha de cuando se realizó la reseña.

```
CREATE TABLE resenas (
  id_resena int(11) NOT NULL AUTO_INCREMENT,
  id_producto int(11) NOT NULL,
  id_cliente int(11) NOT NULL,
  calificacion int(11) NOT NULL,
  comentario text NOT NULL,
  fecha_resena datetime NOT NULL,
  PRIMARY KEY (id_resena),
  KEY fk_resenas_producto (id_producto),
  KEY fk_resenas_cliente (id_cliente),
  CONSTRAINT fk_resenas_producto FOREIGN KEY (id_producto) REFERENCES productos (id_producto) ON UPDATE CASCADE,
  CONSTRAINT fk_resenas_cliente FOREIGN KEY (id_cliente) REFERENCES clientes (id_clientes) ON UPDATE CASCADE
);
```

ANALISIS Y VALIDACION.

Resultados de la ejecución de las consultas.

1. Listar productos disponibles por categoría, ordenados por precio.

```
mysql>
mysql> DELIMITER ;
mysql> SELECT p.nombre_producto, p.precio, p.stock, c.nombre_categoria
    -> FROM productos p
    -> JOIN categorias c ON p.id_categoria = c.id_categoria
    -> WHERE p.stock > 0
    -> ORDER BY c.nombre_categoria, p.precio;
+--------------------------+----------+-------+------------------+
| nombre_producto          | precio   | stock | nombre_categoria |
+--------------------------+----------+-------+------------------+
| Cable HDMI               |   199.00 |    60 | Accesorios       |
| Funda para iPhone        |   199.00 |    80 | Accesorios       |
| Hub USB 4 puertos        |   249.00 |    90 | Accesorios       |
| Soporte para laptop      |   299.00 |    25 | Accesorios       |
| Cargador USB-C           |   299.00 |   100 | Accesorios       |
| Lámpara LED escritorio   |   349.00 |    60 | Accesorios       |
| Audífonos con micrófono  |   399.00 |    30 | Accesorios       |
| Power Bank 10000mAh      |   399.00 |    40 | Accesorios       |
| Audífonos Bluetooth      |   499.00 |    70 | Accesorios       |
| Mouse gamer              |   599.00 |    50 | Accesorios       |
| Teclado inalámbrico      |   699.00 |    30 | Accesorios       |
| Webcam HD                |   799.00 |    22 | Accesorios       |
| Chromebook               |  6999.00 |    10 | Laptops          |
| Acer Aspire              |  7999.00 |    15 | Laptops          |
| Lenovo IdeaPad 3         |  8999.00 |    25 | Laptops          |
| HP Pavilion              | 13999.00 |    20 | Laptops          |
| HP Envy                  | 15999.00 |     8 | Laptops          |
| LG Gram                  | 17999.00 |     5 | Laptops          |
| Dell XPS 13              | 18999.00 |    12 | Laptops          |
| MacBook Air M1           | 23999.00 |    15 | Laptops          |
| Asus ROG                 | 29999.00 |     7 | Laptops          |
| Realme C25               |  4499.00 |    35 | Teléfonos        |
| Motorola G9              |  4999.00 |    50 | Teléfonos        |
| Nokia 5.4                |  5999.00 |    20 | Teléfonos        |
| Xiaomi Redmi Note 10     |  6999.00 |    30 | Teléfonos        |
| Huawei P30               | 10999.00 |    10 | Teléfonos        |
| OnePlus Nord             | 12999.00 |    13 | Teléfonos        |
| iPhone 12                | 14999.00 |    10 | Teléfonos        |
| Samsung Galaxy S21       | 15999.00 |    18 | Teléfonos        |
| iPhone 13                | 17999.00 |    25 | Teléfonos        |
+--------------------------+----------+-------+------------------+
30 rows in set (0,00 sec)
```

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

```
mysql> SELECT cl.id_clientes, cl.nombre_cliente, cl.correo_electronico,
    ->        COUNT(p.id_pedido) AS pedidos_pendientes,
    ->        COALESCE(SUM(dp.cantidad * pr.precio), 0) AS total_compras
    -> FROM clientes cl
    -> JOIN pedidos p ON cl.id_clientes = p.id_cliente
    -> JOIN estado_pedido ep ON p.id_estado_pedido = ep.id_estado_pedido
    -> LEFT JOIN detalle_pedido dp ON p.id_pedido = dp.id_pedido
    -> LEFT JOIN productos pr ON dp.id_producto = pr.id_producto
    -> WHERE ep.estado = 'pendiente'
    -> GROUP BY cl.id_clientes
    -> ORDER BY cl.id_clientes;
+-------------+-----------------+---------------------+--------------------+---------------+
| id_clientes | nombre_cliente  | correo_electronico  | pedidos_pendientes | total_compras |
+-------------+-----------------+---------------------+--------------------+---------------+
|           1 | Juan Pérez      | juan1@gmail.com     |                  2 |      18597.00 |
|           2 | Ana Torres      | ana2@gmail.com      |                  1 |       9998.00 |
|           3 | Luis Gómez      | luis3@gmail.com     |                  2 |       7198.00 |
|           5 | Carlos Sánchez  | carlos5@gmail.com   |                  2 |       9698.00 |
|           8 | Lucía Herrera   | lucia8@gmail.com    |                  1 |        597.00 |
|          11 | Ricardo Lara    | ricardo11@gmail.com |                  1 |        399.00 |
|          14 | Valeria Ramos   | valeria14@gmail.com |                  1 |       6999.00 |
+-------------+-----------------+---------------------+--------------------+---------------+
7 rows in set (0,00 sec)
```

Resultado: La tabla muestra correctamente los clientes con pedidos pendientes y el precio total pagado por sus compras.

3. Reporte con los 5 productos con mejor calificación promedio en reseñas.

```
mysql> SELECT pr.nombre_producto, AVG(r.calificacion) AS calificacion_promedio
    -> FROM productos pr
    -> JOIN resenas r ON pr.id_producto = r.id_producto
    -> GROUP BY pr.id_producto
    -> ORDER BY calificacion_promedio DESC
    -> LIMIT 5;
+---------------------+----------------------+
| nombre_producto     | calificacion_promedio |
+---------------------+----------------------+
| iPhone 13           |               5.0000 |
| MacBook Air M1      |               5.0000 |
| Realme C25          |               5.0000 |
| iPhone 12           |               5.0000 |
| Audífonos Bluetooth |               4.0000 |
+---------------------+----------------------+
5 rows in set (0,00 sec)
```

Resultado: Muestra correctamente los 5 productos con mayor calificación promedio en reseñas.

PRUEBAS DE PROCEDIMIENTOS ALMACENADOS CON DIFERENTES

ESCENARIOS.

1. Registrar un nuevo pedido, verificando el límite de 5 pedidos pendientes y stock suficiente.

ESCENARIO: sin pasarse de los 5 pedidos pendientes y con stock

```
mysql> CALL sp_registrar_pedido(3, 1, 2, '2025-07-30');
Query OK, 1 row affected (0,03 sec)

mysql> SELECT * FROM pedidos WHERE id_cliente = 3;
+-----------+------------+------------------+---------------------+
| id_pedido | id_cliente | id_estado_pedido | fecha_pedido        |
+-----------+------------+------------------+---------------------+
|         3 |          3 |                1 | 2025-07-03 00:00:00 |
|        18 |          3 |                2 | 2025-07-18 00:00:00 |
|        21 |          3 |                1 | 2025-07-30 00:00:00 |
+-----------+------------+------------------+---------------------+
3 rows in set (0,00 sec)
```

```
mysql> SELECT * FROM detalle_pedido  WHERE id_pedido = 21;
+-------------------+-----------+-------------+----------+--------------+
| id_detalle_pedido | id_pedido | id_producto | cantidad | precio_total |
+-------------------+-----------+-------------+----------+--------------+
|                26 |        21 |           1 |        2 |     35998.00 |
+-------------------+-----------+-------------+----------+--------------+
1 row in set (0,00 sec)
```

```
mysql> SELECT * FROM productos WHERE id_producto = 1;
+-------------+--------------+-----------------+---------------------+----------+-------+
| id_producto | id_categoria | nombre_producto | descripcion         | precio   | stock |
+-------------+--------------+-----------------+---------------------+----------+-------+
|           1 |            1 | iPhone 13       | Smartphone de Apple | 17999.00 |    23 |
+-------------+--------------+-----------------+---------------------+----------+-------+
1 row in set (0,00 sec)
```

ESCENARIO: Registrando pedidos donde se piden más productos que el stock actual.

```
mysql> CALL sp_registrar_pedido(3, 1, 6, '2025-07-30');
ERROR 1644 (45000): No hay suficiente stock para este producto
```

2. Registrar una reseña, verificando que el cliente haya pedido el producto.

ESCENARIO: El cliente reseña un producto que sí pidió.

```
mysql> CALL sp_registrar_resena(1, 1, 1, 5, 'Excelente producto', '2025-07-05');
Query OK, 1 row affected (0,01 sec)
```

```
mysql> SELECT * FROM resenas WHERE id_cliente = 1;
+-----------+-------------+------------+--------------+----------------------------------+---------------------+
| id_resena | id_producto | id_cliente | calificacion | comentario                       | fecha_resena        |
+-----------+-------------+------------+--------------+----------------------------------+---------------------+
|         1 |           1 |          1 |            5 | Excelente teléfono, muy rápido.  | 2025-07-05 00:00:00 |
|        11 |           1 |          1 |            4 | Muy buen producto                | 2025-07-30 00:00:00 |
|        12 |           1 |          1 |            5 | Excelente producto               | 2025-07-05 00:00:00 |
+-----------+-------------+------------+--------------+----------------------------------+---------------------+
3 rows in set (0,00 sec)
```

ESCENARIO: Hacer reseña sin comprar el producto

```
mysql> CALL sp_registrar_resena(1, 1, 2, 5, 'Excelente producto', '2025-08-06');
ERROR 1644 (45000): El cliente no ha comprado este producto
```

3. Actualizar el stock de un producto después de un pedido.

ESCENARIO: Se vendieron 5 teclados inalámbricos, se tiene que actualizar el stock.
El producto tiene 25 productos en stock:

```
mysql> CALL sp_actualizar_stock(14,5);
Query OK, 1 row affected (0,01 sec)

mysql> SELECT id_producto, nombre_producto, stock FROM productos WHERE id_producto=14;
+-------------+----------------------+-------+
| id_producto | nombre_producto      | stock |
+-------------+----------------------+-------+
|          14 | Teclado inalámbrico  |    25 |
+-------------+----------------------+-------+
```

ESCENARIO: no hay suficiente stock.

```
mysql> CALL sp_actualizar_stock(14,25);
ERROR 1644 (45000): Este producto no tiene stock suficiente
```

4. Cambiar el estado de un pedido.

ESCENARIO: Se cambia el estado(id_estado) de un pedido de pendiente (1) a entregado (3).

```
mysql> SELECT * FROM pedidos WHERE id_pedido =1;
+-----------+------------+------------------+---------------------+
| id_pedido | id_cliente | id_estado_pedido | fecha_pedido        |
+-----------+------------+------------------+---------------------+
|         1 |          1 |                3 | 2025-07-01 00:00:00 |
+-----------+------------+------------------+---------------------+
1 row in set (0,00 sec)
```

```
mysql> SELECT * FROM pedidos WHERE id_estado_pedido =1 AND id_pedido=1;
+-----------+------------+------------------+---------------------+
| id_pedido | id_cliente | id_estado_pedido | fecha_pedido        |
+-----------+------------+------------------+---------------------+
|         1 |          1 |                1 | 2025-07-01 00:00:00 |
+-----------+------------+------------------+---------------------+
1 row in set (0,00 sec)
```

ESCENARIO: Se ingresó un estado inexistente.

```
mysql> CALL sp_cambiar_estado(4,6);
ERROR 1644 (45000): Estado no existente (1-4)
```

5. Eliminar reseñas de un producto en específico.

ESCENARIO: Se quieren eliminar las reseñas del producto con id=1:

```
mysql> SELECT * FROM resenas WHERE id_producto=1;
+-----------+-------------+------------+--------------+----------------------------------+---------------------+
| id_resena | id_producto | id_cliente | calificacion | comentario                       | fecha_resena        |
+-----------+-------------+------------+--------------+----------------------------------+---------------------+
|         1 |           1 |          1 |            5 | Excelente teléfono, muy rápido.  | 2025-07-05 00:00:00 |
|        11 |           1 |          1 |            4 | Muy buen producto                | 2025-07-30 00:00:00 |
|        12 |           1 |          1 |            5 | Excelente producto               | 2025-07-05 00:00:00 |
|        13 |           1 |          1 |            5 | Excelente producto               | 2025-07-02 00:00:00 |
+-----------+-------------+------------+--------------+----------------------------------+---------------------+
```

```
mysql> CALL sp_eliminar_resena(1);
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM resenas WHERE id_producto=1;
Empty set (0.00 sec)
```

ESCENARIO: Se quieren eliminar las reseñas de un producto, pero se ingresa una id que no tiene reseñas.

```
mysql> CALL sp_eliminar_resena(12);
ERROR 1644 (45000): No hay resenas de ese producto
```

6. Agregar un nuevo producto verificando que no exista un duplicado (mismo nombre y categoría)

ESCENARIO: Se agrega un producto normal sin nombre ni categoría duplicados:

```
mysql> CALL sp_agregar_producto('Nintendo Switch 2', 'Consola de nintendo ', 13999.00, 10, 5);
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM productos WHERE nombre='Nintendo Switch 2';
+-------------+------------------+----------------------+----------+-------+--------------+
| id_producto | nombre           | descripcion          | precio   | stock | id_categoria |
+-------------+------------------+----------------------+----------+-------+--------------+
|          36 | Nintendo Switch 2 | Consola de nintendo  | 13999.00 |    10 |            5 |
+-------------+------------------+----------------------+----------+-------+--------------+
1 row in set (0.00 sec)
```

ESCENARIO: Se quiere agregar un producto con mismo nombre y categoría que uno existente.

```
mysql> CALL sp_agregar_producto('Nintendo Switch 2', 'Consola de nintendo ', 13999.00, 10, 5);
ERROR 1644 (45000): Ya existe un producto con ese nombre y caracteristicas
mysql>
```

7. Actualizar el teléfono de un cliente.

ESCENARIO: Un cliente quiere actualizar su número del teléfono celular:

```
mysql> SELECT id_cliente,nombre,telefono FROM clientes WHERE id_cliente=1
+------------+-------------+------------+
| id_cliente | nombre      | telefono   |
+------------+-------------+------------+
|          1 | Luis Garcia | 5512122101 |
+------------+-------------+------------+
1 row in set (0.00 sec)
```

```
mysql> CALL sp_actualizar_telefono_cliente(1,'5588128812');
Query OK, 1 row affected (0.02 sec)

mysql> SELECT id_cliente,nombre,telefono FROM clientes WHERE id_cliente=1;
+------------+------------+------------+
| id_cliente | nombre     | telefono   |
+------------+------------+------------+
|          1 | Luis Garcia | 5588128812 |
+------------+------------+------------+
1 row in set (0.00 sec)
```

ESCENARIO: Un cliente cambia su teléfono pero otro cliente ya tiene ese número registrado.

```
mysql> CALL sp_actualizar_telefono_cliente(15,'5588128812');
ERROR 1644 (45000): El telefono ya esta registrado por otro cliente
```

El teléfono ya estaba registrado por el usuario del escenario anterior.

8.- Generar un reporte de productos con stock bajo (menos de 5 unidades)

A este procedimiento no le encontré algún escenario en específico cuando se el call te devuelve la tabla con los productos con 5 o menos de 5 productos de stock.

```
mysql> CALL sp_reporte_stock();
+---------------------+-------+
| nombre              | stock |
+---------------------+-------+
| HP Pavilion 15      |     2 |
| Philips TV          |     2 |
| Motorola G84        |     3 |
| Hisense A6H         |     3 |
| PlayStation 5       |     3 |
| MacBook Air M2      |     4 |
| Samsung Smart TV 50 |     4 |
| Xiaomi 24i          |     4 |
| HyperX Cloud Stinger |     4 |
| Mario Kart 8 Deluxe |     4 |
| Asus VivoBook 14    |     5 |
| MSI Modern 15       |     5 |
| Samsung Galaxy A54  |     5 |
| AOC 24B1XHS         |     5 |
| Redragon K552       |     5 |
| Nintendo Switch OLED |     5 |
+---------------------+-------+
16 rows in set (0.00 sec)

Query OK, 0 rows affected (0.02 sec)
```

PROPUESTAS DE MEJORAS

Durante el desarrollo del proyecto, identifiqué varias formas en que la base de datos podría optimizarse para mejorar el rendimiento y la integridad de los datos:

Índices adicionales:

Para mejorar el rendimiento de las consultas frecuentes, se pueden agregar índices en:

- La columna “fecha_pedido” en la tabla pedidos, para búsquedas por fecha.
- La columna “id_estado_pedido” en pedidos, para búsquedas por estado del pedido.
- La columna “id_pedido” en “detalle_pedido”, para consultar rápidamente los productos comprados en un pedido.
- La columna “correo_electronico” en clientes, para localizar clientes por su email (aunque ya es UNIQUE, también funciona como índice).

Triggers en lugar de procedimientos:

Algunos procedimientos podrían reemplazarse o complementarse con triggers. Por ejemplo:

- Crear un trigger BEFORE INSERT en la tabla “esenas” para validar automáticamente que el cliente realmente compró el producto antes de permitir registrar la reseña.
- Crear un trigger BEFORE INSERT en la “tabla pedidos” para evitar que un cliente tenga más de 5 pedidos pendientes.

Conclusiones y aprendizajes

El desarrollo de esta base de datos fue una experiencia enriquecedora que me permitió consolidar los conocimientos adquiridos durante el curso. Algunos de los aprendizajes más importantes que obtuve son:

- **Importancia del diseño**:
  El diseño correcto de una base de datos es fundamental. Una estructura bien pensada no solo permite que el sistema funcione correctamente, sino que también mejora la eficiencia en el acceso y manipulación de los datos.

- **Optimización desde el modelado**:
  Usar claves foráneas, restricciones, índices y procedimientos bien diseñados facilita que los datos se mantengan consistentes y permite automatizar reglas de negocio importantes directamente en la base.

- **Preparación para el mundo real**:
  Este proyecto me ayudó a comprender cómo se integran las bases de datos dentro de sistemas más grandes, y cómo su correcto diseño tiene un impacto directo en el rendimiento, mantenimiento y escalabilidad de las aplicaciones.

En conclusión, aprender a construir y optimizar bases de datos es una competencia clave en el entorno laboral actual. Esta experiencia no solo fortaleció mis conocimientos técnicos, sino que también me dejó con herramientas prácticas que podré aplicar en proyectos profesionales futuros.