**3PROYECTO DE BASE DE DATOS**

**SISTEMA DE GESTION DE TIENDA ONLINE**

**INTRODUCCIÓN AL PROBLEMA Y OBJETIVOS**.

Una tienda online de productos electrónicos necesita un sistema para gestionar sus operaciones, incluyendo productos, clientes, pedidos, reseñas de productos y categorías.

Parte central de este sistema es su base de datos relacional que cumpla con los siguientes **requerimientos funcionales**:

1. **Gestión de Productos.** Registrar productos con nombre, descripción, precio, stock y categoría.
2. **Gestión de Clientes.** Registrar clientes con nombre, correo electrónico, dirección y número de teléfono.
3. **Pedidos**. Gestionar pedidos, incluyendo fecha, estado (pendiente, enviado, entregado) y productos incluidos.
4. **Reseñas**. Permitir reseñas de productos con calificación (1-5 estrellas) y comentarios, solo por clientes que hayan comprado el producto.
5. **Categorías**. Organizar productos en categorías (por ejemplo, teléfonos, laptops, accesorios) con descripción.
6. **Restricciones**:

* Máximo 5 pedidos pendientes por cliente.
* Stock de productos no negativo.
* Reseñas sólo de clientes que hayan comprado el producto.

El sistema de gestión también debe cumplir los siguientes **requerimientos no funcionales**, que se deben refinar con el dueño de la tienda online para poder cuantificar cada requerimiento:

1. **Desempeño**. Lograr un tiempo de respuesta “adecuado” en la ejecución de consultas y procedimientos almacenados. Ya en operación, permitir accesos concurrentes a la base de datos por diferentes usuarios, sin degradar los tiempos de respuesta.
2. **Seguridad**. Proteger los datos en la base de datos, especialmente de clientes, productos y pedidos. Autenticar a las personas con acceso a las distintas tablas de la base de datos. Controlar las acciones que las personas autorizadas pueden realizar sobre las tablas de la base de datos (consultar, insertar, borrar, actualizar registros) en base a su rol.
3. **Disponibilidad**. La base de datos debe estar disponible 24x7x365. Ya en operación, contar con una réplica de la base de datos. Realizar respaldos completos cada semana e incrementales cada día.
4. **Facilidad de mantenimiento**. Documentar los scripts de creación y llenado de la base de datos, las consultas y los procedimientos almacenados, a fin de facilitar sus actualizaciones.

El objetivo es diseñar una base de datos normalizada, implementar consultas y al menos 8 procedimientos almacenados, optimizar con índices y validar con datos de prueba.

**DIAGRAMA ER Y JUSTIFICACIÓN DE NORMALIZACIÓN**

En el diseño de la base de datos se empleó la estructura sugerida de las tablas. En esta estructura sugerida de tablas ya se indicaban las seis tablas de la base de datos, sus atributos, claves primarias y foráneas. Sólo se les agregaron índices a los atributos que sirven de base a las consultas y procedimientos almacenados.

La estructura de las tablas sugerida ya estaba en tercera forma normal (3FN), cumpliendo también los requisitos de las 1FN y 2FN.

Todas las tablas cumplen con los requisitos de la 1FN:

* No tienen atributos multivaluados
* No tienen atributos compuestos
* Todos los atributos son atómicos

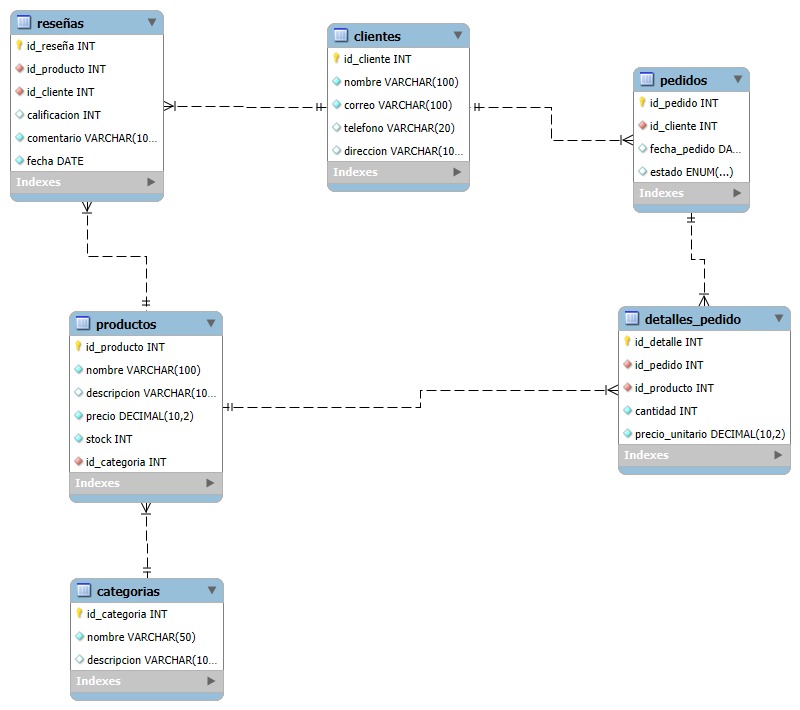
También todas las tablas cumplen con los requisitos de la 2FN:

* Están en la primera forma normal
* Todos los atributos no clave dependen completamente de la clave primaria

Y todas las tablas cumplen con los requisitos de la 3FN:

* Están en segunda forma normal
* No hay dependencias transitivas entre los atributos no clave y la clave primaria

El diagrama entidad relación se hizo con MySQL Workbench y muestra los atributos, claves primarias y foráneas de cada tabla, así como las relaciones entre tablas, incluyendo su multiplicidad.



**Relaciones entre tablas:**

clientes a pedidos: Uno a Muchos (Un cliente puede tener muchos pedidos)

pedidos a detalles\_pedido: Uno a muchos (Un pedido puede tener muchos artículos)

producto a detalles\_pedido: Uno a muchos (Un producto puede aparecer como artículo en muchos pedidos)

productos a reseñas: Uno a muchos (Un producto puede tener muchas reseñas)

clientes a reseñas: Uno a muchos (Un cliente puede escribir muchas reseñas)

categorías a productos: Una a muchos (Una categoría puede incluir muchos productos)

**SCRIPTS SQL COMPLETOS** (tablas, índices, datos de prueba, consultas, procedimientos).

***Script para crear la base de datos***.

-- Crear base de datos

CREATE DATABASE IF NOT EXISTS tienda\_electronicos;

USE tienda\_electronicos;

-- Tabla de Clientes

CREATE TABLE clientes (

id\_cliente INT PRIMARY KEY AUTO\_INCREMENT,

nombre VARCHAR(100) NOT NULL,

correo VARCHAR(100) UNIQUE NOT NULL,

telefono VARCHAR(20),

direccion VARCHAR(100),

INDEX indice\_correo (correo) -- indice para busquedas por correo

);

-- Tabla de Categorías de Productos

CREATE TABLE categorias (

id\_categoria INT PRIMARY KEY AUTO\_INCREMENT,

nombre VARCHAR(50) NOT NULL,

descripcion VARCHAR(100),

INDEX indice\_nombre\_categoria (nombre) -- indice para busqueda por nombre de categoria

);

-- Tabla de Productos

CREATE TABLE productos (

id\_producto INT PRIMARY KEY AUTO\_INCREMENT,

nombre VARCHAR(100) NOT NULL,

descripcion VARCHAR(100),

precio DECIMAL(10,2) NOT NULL,

stock INT NOT NULL DEFAULT 0 CHECK (stock >= 0), --restriccion stock no negativo,

id\_categoria INT NOT NULL,

FOREIGN KEY (id\_categoria) REFERENCES categorias(id\_categoria),

INDEX indice\_nombre\_producto (nombre), -- indice para busqueda por nombre de producto

INDEX indice\_categoria\_producto (id\_categoria) -- indice para busqueda por categoria

);

--Tabla de Pedidos

CREATE TABLE pedidos (

id\_pedido INT PRIMARY KEY AUTO\_INCREMENT,

id\_cliente INT NOT NULL,

fecha\_pedido DATE DEFAULT (CURRENT\_DATE),

estado ENUM('pendiente', 'procesando', 'enviado', 'entregado', 'cancelado') DEFAULT 'pendiente',

FOREIGN KEY (id\_cliente) REFERENCES clientes(id\_cliente),

INDEX indice\_cliente\_pedido (id\_cliente), -- iindice para búsquedas de pedidos por cliente

INDEX indice\_estado\_pedido (estado) -- indice para busqueda por estado de pedido

);

--Tabla de Detalles de Pedidos

CREATE TABLE detalles\_pedido (

id\_detalle INT PRIMARY KEY AUTO\_INCREMENT,

id\_pedido INT NOT NULL,

id\_producto INT NOT NULL,

cantidad INT NOT NULLi CHECK (cantidad > 0), -- restriccion para cantidad positiva

precio\_unitario DECIMAL(10,2) NOT NULL,

FOREIGN KEY (id\_pedido) REFERENCES pedidos(id\_pedido),

FOREIGN KEY (id\_producto) REFERENCES productos(id\_producto),

INDEX indice\_pedido\_detalle (id\_pedido) -- indice para busquedas de detalles por pedidoa

);

--Tabla de Reseñas

CREATE TABLE reseñas (

id\_reseña INT PRIMARY KEY AUTO\_INCREMENT,

id\_producto INT NOT NULL,

id\_cliente INT NOT NULL,

calificacion INT NOT NULL CHECK (calificacion BETWEEN 1 AND 5),

comentario VARCHAR(100),

fecha DATE DEFAULT (CURRENT DATE),

FOREIGN KEY (id\_producto) REFERENCES productos(id\_producto),

FOREIGN KEY (id\_cliente) REFERENCES clientes(id\_cliente),

INDEX indice\_producto\_reseña (id\_producto) -- indice para busquedas de reseñas por producto

);

Notas.

* Los atributos con el id de cada tabla se definieron como AUTO\_INCREMENT.
* Los atributos que no pueden quedar vacíos se definieron como NOT NULL.
* Los atributos que no pueden tener valores repetidos se definieron como UNIQUE.
* Se puso restricción al atributo stock de la tabla productos para que el stock no sea negativo CHECK (stock >= 0).
* Se puso restricción al atributo cantidad de la tabla detalle\_pedido para que la cantidad ordenada sea positiva CHECK (cantidad > 0).
* Se puso restricción al atributo calificación de la tabla reseñas para que la calificación sea de 1 a 5 estrellas CHECK (calificacion BETWEEN 1 AND 5).
* Se pusieron valores de default a los campos de fecha DEFAULT (CURRENT\_DATE) y al de stock DEFAULT 0.
* Los estados de un pedido se definieron como una enumeración de cinco posibles estados ENUM('pendiente', 'procesando', 'enviado', 'entregado', 'cancelado'), con un valor de DEFAULT 'pendiente'.

***Script para insertar los datos de prueba en cada tabla***.

Se insertaron las cantidades mínimas de datos de prueba por tabla, con valores realistas de categorías, modelos, descripciones y precios de productos electrónicos.

Los inserts en las tablas comienzan con las tablas que no tienen dependencia de otras tablas (que no tienen llaves foráneas).

USE tienda\_electronicos;

-- Insertar categorías

INSERT INTO categorias (nombre, descripcion) VALUES

('Smartphones', 'Teléfonos inteligentes de última generación'),

('Laptops', 'Computadoras portátiles para trabajo y entretenimiento'),

('Televisores', 'Pantallas para el hogar con diferentes tecnologías'),

('Audio', 'Equipos de sonido y accesorios'),

('Gaming', 'Consolas y accesorios para videojuegos'),

('Electrodomésticos', 'Aparatos electrónicos para el hogar');

-- Insertar productos (30 productos)

INSERT INTO productos (nombre, descripcion, precio, stock, id\_categoria) VALUES

-- Smartphones

('iPhone 15 Pro', 'Apple iPhone 15 Pro 256GB Plata', 24999.00, 50, 1),

('Samsung Galaxy S23 Ultra', 'Samsung Galaxy S23 Ultra 256GB Negro', 22999.00, 4, 1),

('Xiaomi Redmi Note 12', 'Xiaomi Redmi Note 12 128GB Azul', 5499.00, 80, 1),

('Motorola Edge 40', 'Motorola Edge 40 256GB Verde', 8999.00, 30, 1),

('Huawei P60 Pro', 'Huawei P60 Pro 512GB Oro', 18999.00, 25, 1),

-- Laptops

('MacBook Pro 14" M2', 'Apple MacBook Pro 14" M2 512GB Plata', 37999.00, 20, 2),

('Dell XPS 15', 'Dell XPS 15 i7 16GB 512GB Negro', 32999.00, 1, 2),

('HP Spectre x360', 'HP Spectre x360 i5 8GB 256GB Azul', 21999.00, 18, 2),

('Lenovo ThinkPad X1', 'Lenovo ThinkPad X1 Carbon i7 16GB 1TB Negro', 28999.00, 12, 2),

('Asus ROG Zephyrus', 'Asus ROG Zephyrus G15 Ryzen 9 32GB 1TB Gris', 41999.00, 10, 2),

-- Televisores

('LG OLED C3 55"', 'LG OLED C3 55" 4K Smart TV', 24999.00, 25, 3),

('Samsung QN90B 65"', 'Samsung Neo QLED QN90B 65" 4K', 32999.00, 1, 3),

('Sony X80K 50"', 'Sony Bravia X80K 50" 4K Android TV', 12999.00, 30, 3),

('TCL 55C735 55"', 'TCL 55C735 55" QLED Google TV', 15999.00, 22, 3),

('Hisense U7H 65"', 'Hisense U7H 65" ULED Android TV', 21999.00, 15, 3),

-- Audio

('Sony WH-1000XM5', 'Audífonos inalámbricos Sony WH-1000XM5', 6999.00, 40, 4),

('Bose QuietComfort 45', 'Audífonos Bose QuietComfort 45', 6499.00, 3, 4),

('JBL Flip 6', 'Bocina portátil JBL Flip 6 Azul', 1999.00, 60, 4),

('Sony HT-S400', 'Soundbar Sony HT-S400 2.1 canales', 3999.00, 25, 4),

('Samsung HW-Q600A', 'Soundbar Samsung HW-Q600A 3.1.2 canales', 5999.00, 20, 4),

-- Gaming

('PlayStation 5', 'Consola PlayStation 5 825GB', 12999.00, 30, 5),

('Xbox Series X', 'Consola Xbox Series X 1TB', 11999.00, 2, 5),

('Nintendo Switch OLED', 'Consola Nintendo Switch OLED Blanco', 8999.00, 35, 5),

('Logitech G Pro X', 'Audífonos gaming Logitech G Pro X', 3499.00, 50, 5),

('Razer BlackWidow V3', 'Teclado mecánico Razer BlackWidow V3', 2999.00, 45, 5),

-- Electrodomésticos

('LG InstaView', 'Refrigerador LG InstaView 23 pies', 32999.00, 10, 6),

('Samsung Bespoke', 'Lavadora Samsung Bespoke 20kg', 17999.00, 2, 6),

('Mabe RME1430XMN', 'Microondas Mabe 1.4 pies', 2999.00, 30, 6),

('Philips Airfryer', 'Freidora de aire Philips HD9650', 4999.00, 25, 6),

('Oster Xpert', 'Licuadora Oster Xpert 8 velocidades', 1999.00, 40, 6);

-- Insertar clientes (15 clientes)

INSERT INTO clientes (nombre, correo, telefono, direccion) VALUES

('Juan Pérez López', 'juan.perez@gmail.com', '5551234567', 'Av. Insurgentes Sur 123, Col. Roma Norte, CDMX'),

('María González Hernández', 'maria.gonzalez@hotmail.com', '5559876543', 'Calle Morelos 45, Col. Centro, Guadalajara'),

('Carlos Sánchez Martínez', 'carlos.sanchez@yahoo.com', '5554567890', 'Blvd. Díaz Ordaz 789, Col. Del Valle, Monterrey'),

('Ana Rodríguez García', 'ana.rodriguez@gmail.com', '5552345678', 'Av. Revolución 567, Col. Tacubaya, CDMX'),

('Luis Fernández Jiménez', 'luis.fernandez@outlook.com', '5558765432', 'Calle Hidalgo 34, Col. Americana, Guadalajara'),

('Patricia Ramírez Díaz', 'paty.ramirez@gmail.com', '5553456789', 'Av. Constitución 890, Col. Moderna, Puebla'),

('Jorge Torres Ruiz', 'jorge.torres@hotmail.com', '5557654321', 'Blvd. Benito Juárez 123, Col. Narvarte, CDMX'),

('Sofía Mendoza Castro', 'sofia.mendoza@yahoo.com', '5556543210', 'Av. Universidad 456, Col. Del Carmen, Coyoacán'),

('Miguel Ángel Vargas', 'miguel.vargas@gmail.com', '5554321098', 'Calle 5 de Mayo 67, Col. Centro, Querétaro'),

('Laura Gómez Silva', 'laura.gomez@outlook.com', '5553210987', 'Av. López Mateos 789, Col. San Pedro, Tijuana'),

('Roberto Navarro Ortega', 'roberto.navarro@hotmail.com', '5552109876', 'Blvd. Aguacaliente 345, Col. Chapultepec, Aguascalientes'),

('Isabel Cruz Mendoza', 'isabel.cruz@gmail.com', '5551098765', 'Av. Tecnológico 901, Col. Linda Vista, Chihuahua'),

('Fernando Herrera Luna', 'fer.herrera@yahoo.com', '5550987654', 'Calle Zaragoza 234, Col. Juárez, Hermosillo'),

('Adriana Morales Ríos', 'adri.morales@hotmail.com', '5559876543', 'Av. Las Torres 567, Col. Las Brisas, León'),

('Ricardo Castro Méndez', 'ricardo.castro@gmail.com', '5558765432', 'Blvd. Luis Donaldo Colosio 890, Col. Las Palmas, Cancún');

-- Insertar pedidos (20 pedidos)

INSERT INTO pedidos (id\_cliente, fecha\_pedido, estado) VALUES

(1, '2023-10-15', 'entregado'),

(2, '2023-10-16', 'entregado'),

(3, '2023-10-17', 'enviado'),

(4, '2023-10-18', 'procesando'),

(5, '2023-10-19', 'pendiente'),

(6, '2023-10-20', 'entregado'),

(7, '2023-10-21', 'cancelado'),

(8, '2023-10-22', 'entregado'),

(9, '2023-10-23', 'enviado'),

(10, '2023-10-24', 'procesando'),

(11, '2023-10-25', 'pendiente'),

(12, '2023-10-26', 'entregado'),

(13, '2023-10-27', 'enviado'),

(14, '2023-10-28', 'procesando'),

(15, '2023-10-29', 'pendiente'),

(1, '2023-10-30', 'entregado'),

(3, '2023-10-31', 'enviado'),

(5, '2023-11-01', 'procesando'),

(7, '2023-11-02', 'cancelado'),

(9, '2023-11-03', 'entregado');

-- Insertar detalles de pedido (25 detalles)

INSERT INTO detalles\_pedido (id\_pedido, id\_producto, cantidad, precio\_unitario) VALUES

-- Pedido 1

(1, 1, 1, 24999.00),

(1, 16, 1, 6999.00),

-- Pedido 2

(2, 6, 1, 37999.00),

-- Pedido 3

(3, 2, 1, 22999.00),

(3, 21, 1, 12999.00),

(3, 25, 1, 2999.00),

-- Pedido 4

(4, 3, 2, 5499.00),

-- Pedido 5

(5, 7, 1, 32999.00),

(5, 17, 1, 6499.00),

-- Pedido 6

(6, 11, 1, 24999.00),

-- Pedido 7

(7, 22, 1, 11999.00),

-- Pedido 8

(8, 4, 1, 8999.00),

(8, 18, 2, 1999.00),

-- Pedido 9

(9, 8, 1, 21999.00),

(9, 23, 1, 8999.00),

-- Pedido 10

(10, 12, 1, 32999.00),

-- Pedido 11

(11, 5, 1, 18999.00),

(11, 19, 1, 3999.00),

-- Pedido 12

(12, 9, 1, 28999.00),

-- Pedido 13

(13, 13, 1, 12999.00),

(13, 24, 1, 3499.00),

-- Pedido 14

(14, 14, 1, 15999.00),

-- Pedido 15

(15, 10, 1, 41999.00),

(15, 20, 1, 5999.00),

-- Pedido 16

(16, 26, 1, 32999.00);

-- Insertar reseñas (10 reseñas)

INSERT INTO reseñas (id\_producto, id\_cliente, calificacion, comentario, fecha) VALUES

(1, 1, 2, 'Imma keep it a buck fifty with you, G, the phone is dog water. The only good thing about it are the basic phone functions.', '2023-10-20'),

(6, 2, 4, 'Vey well made laptop, however, it is slightly expensive', '2023-10-21'),

(11, 6, 5, 'This is indubitably the greatest intelligent phone I have ever used', '2023-10-25'),

(16, 1, 5, 'The headphones cancel all external noise perfectly', '2023-10-22'),

(21, 3, 4, 'PS5 functions quite well, but it has limited games', '2023-10-28'),

(2, 3, 3, 'Keep it a stack, phone is great, BUT the battery runs out fast', '2023-10-30'),

(7, 5, 5, 'I LOVE refrigeratorssss ', '2023-11-02'),

(12, 8, 4, 'Good evening sir. Would you like to hear about our Lord and Saviour, Jesus Christ--JehovaWitness', '2023-10-27'),

(17, 9, 5, 'Was that the bite of 87???!!!', '2023-11-01'),

(22, 7, 2, 'I am not in danger, Skyler, I AM THE DANGER! A guy opens his door, and gets shot, and you think that´s me? No. I AM THE ONE WHO KNOCKS!--bluecancer420', '2023-10-29');

***Script con consultas y procedimientos almacenados.***

Script con las 3 consultas y 8 procedimientos almacenados solicitados. En todos los procedimientos almacenados se incluyó un argumento de salida p\_resultado para almacenar un mensaje que indica si el procedimiento se ejecutó con éxito o el mensaje de error que corresponda.

USE tienda\_electronicos;

-- -----------------------------------------------------

-- CONSULTAS

-- -----------------------------------------------------

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

SELECT

c.nombre AS categoria,

p.nombre AS producto,

p.descripcion,

p.precio,

p.stock

FROM

productos p

JOIN

categorias c ON p.id\_categoria = c.id\_categoria

WHERE

p.stock > 0

ORDER BY

c.nombre, p.precio;

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

SELECT

cl.id\_cliente,

cl.nombre,

cl.correo,

COUNT(p.id\_pedido) AS pedidos\_pendientes,

SUM(dp.cantidad \* dp.precio\_unitario) AS total\_compras

FROM

clientes cl

JOIN

pedidos p ON cl.id\_cliente = p.id\_cliente

JOIN

detalles\_pedido dp ON p.id\_pedido = dp.id\_pedido

WHERE

p.estado = 'pendiente'

GROUP BY

cl.id\_cliente, cl.nombre, cl.correo;

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

SELECT

p.id\_producto,

p.nombre,

AVG(r.calificacion) AS calificacion\_promedio,

COUNT(r.id\_reseña) AS total\_reseñas

FROM

productos p

JOIN

reseñas r ON p.id\_producto = r.id\_producto

GROUP BY

p.id\_producto, p.nombre

ORDER BY

calificacion\_promedio DESC

LIMIT 5;

-- -----------------------------------------------------

-- PROCEDIMIENTOS ALMACENADOS

-- -----------------------------------------------------

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

DELIMITER //

CREATE PROCEDURE registrar\_pedido(

IN p\_id\_cliente INT,

IN p\_id\_producto INT,

IN p\_cantidad INT,

OUT p\_resultado VARCHAR(200)

)

BEGIN

DECLARE pedidos\_pendientes INT;

DECLARE stock\_actual INT;

DECLARE precio\_producto DECIMAL(10,2);

-- Verificar pedidos pendientes del cliente

SELECT COUNT(\*) INTO pedidos\_pendientes

FROM pedidos

WHERE id\_cliente = p\_id\_cliente AND estado = 'pendiente';

-- Verificar stock disponible

SELECT stock, precio INTO stock\_actual, precio\_producto

FROM productos

WHERE id\_producto = p\_id\_producto;

IF pedidos\_pendientes >= 5 THEN

SET p\_resultado = 'Error: El cliente ya tiene 5 pedidos pendientes';

ELSEIF stock\_actual < p\_cantidad THEN

SET p\_resultado = CONCAT('Error: Stock insuficiente. Disponible: ', stock\_actual);

ELSE

-- Registrar el pedido

INSERT INTO pedidos (id\_cliente, estado) VALUES (p\_id\_cliente, 'pendiente');

-- Registrar detalle del pedido

INSERT INTO detalles\_pedido (id\_pedido, id\_producto, cantidad, precio\_unitario)

VALUES (LAST\_INSERT\_ID(), p\_id\_producto, p\_cantidad, precio\_producto);

-- Actualizar stock

UPDATE productos SET stock = stock - p\_cantidad WHERE id\_producto = p\_id\_producto;

SET p\_resultado = 'Pedido registrado exitosamente';

END IF;

END //

DELIMITER ;

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

DELIMITER //

CREATE PROCEDURE registrar\_reseña(

IN p\_id\_cliente INT,

IN p\_id\_producto INT,

IN p\_calificacion INT,

IN p\_comentario VARCHAR(100),

OUT p\_resultado VARCHAR(200)

)

BEGIN

DECLARE cliente\_compro\_producto INT;

-- Verificar si el cliente compró el producto

SELECT COUNT(\*) INTO cliente\_compro\_producto

FROM pedidos p

JOIN detalles\_pedido dp ON p.id\_pedido = dp.id\_pedido

WHERE p.id\_cliente = p\_id\_cliente AND dp.id\_producto = p\_id\_producto;

IF cliente\_compro\_producto = 0 THEN

SET p\_resultado = 'Error: El cliente no ha comprado este producto';

ELSEIF p\_calificacion < 1 OR p\_calificacion > 5 THEN

SET p\_resultado = 'Error: La calificación debe estar entre 1 y 5';

ELSE

-- Registrar la reseña

INSERT INTO reseñas (id\_producto, id\_cliente, calificacion, comentario)

VALUES (p\_id\_producto, p\_id\_cliente, p\_calificacion, p\_comentario);

SET p\_resultado = 'Reseña registrada exitosamente';

END IF;

END //

DELIMITER ;

-- 3. Actualizar el stock de un producto después de un pedido (siempre disminuye)

DELIMITER //

CREATE PROCEDURE actualizar\_stock\_pedido(

IN p\_id\_producto INT,

IN p\_cantidad INT,

OUT p\_resultado VARCHAR(200)

)

BEGIN

DECLARE stock\_actual INT;

SELECT stock INTO stock\_actual FROM productos WHERE id\_producto = p\_id\_producto;

IF stock\_actual < p\_cantidad THEN

SET p\_resultado = CONCAT('Error: Stock insuficiente. Disponible: ', stock\_actual);

ELSE

-- Siempre disminuir el stock

UPDATE productos SET stock = stock - p\_cantidad WHERE id\_producto = p\_id\_producto;

SET p\_resultado = CONCAT('Stock actualizado. Nuevo stock: ',

(SELECT stock FROM productos WHERE id\_producto = p\_id\_producto));

END IF;

END //

DELIMITER ;

-- 4. Cambiar el estado de un pedido

DELIMITER //

CREATE PROCEDURE cambiar\_estado\_pedido(

IN p\_id\_pedido INT,

IN p\_nuevo\_estado VARCHAR(20),

OUT p\_resultado VARCHAR(200)

)

BEGIN

DECLARE estado\_actual VARCHAR(20);

SELECT estado INTO estado\_actual FROM pedidos WHERE id\_pedido = p\_id\_pedido;

IF estado\_actual IS NULL THEN

SET p\_resultado = 'Error: Pedido no encontrado';

ELSEIF estado\_actual = 'cancelado' THEN

SET p\_resultado = 'Error: No se puede modificar un pedido cancelado';

ELSEIF p\_nuevo\_estado NOT IN ('pendiente', 'procesando', 'enviado', 'entregado', 'cancelado') THEN

SET p\_resultado = 'Error: Estado no válido';

ELSE

UPDATE pedidos SET estado = p\_nuevo\_estado WHERE id\_pedido = p\_id\_pedido;

SET p\_resultado = CONCAT('Estado del pedido ', p\_id\_pedido, ' cambiado a ', p\_nuevo\_estado);

END IF;

END //

DELIMITER ;

-- 5. Eliminar reseñas de un producto (puede ser una o todas) con promedio actualizado

DELIMITER //

CREATE PROCEDURE eliminar\_reseñas\_producto(

IN p\_id\_producto INT,

IN p\_id\_reseña INT, -- NULL para eliminar todas

OUT p\_resultado VARCHAR(200)

)

BEGIN

DECLARE total\_reseñas INT;

DECLARE reseñas\_restantes INT;

DECLARE promedio\_actual DECIMAL(3,2);

-- Verificar si el producto existe

SELECT COUNT(\*) INTO total\_reseñas FROM reseñas WHERE id\_producto = p\_id\_producto;

IF total\_reseñas = 0 THEN

SET p\_resultado = 'Error: El producto no tiene reseñas';

ELSE

IF p\_id\_reseña IS NULL THEN

-- Eliminar todas las reseñas del producto

DELETE FROM reseñas WHERE id\_producto = p\_id\_producto;

SET p\_resultado = 'Todas las reseñas del producto eliminadas. No quedan reseñas.';

ELSE

-- Eliminar solo la reseña específica

DELETE FROM reseñas WHERE id\_reseña = p\_id\_reseña AND id\_producto = p\_id\_producto;

-- Verificar si se eliminó alguna reseña

IF ROW\_COUNT() = 0 THEN

SET p\_resultado = 'Error: No se encontró la reseña especificada para este producto';

ELSE

-- Calcular reseñas restantes y promedio

SELECT

COUNT(\*) AS count,

IFNULL(AVG(calificacion), 0) AS avg

INTO

reseñas\_restantes,

promedio\_actual

FROM reseñas

WHERE id\_producto = p\_id\_producto;

SET p\_resultado = CONCAT(

'Reseña eliminada. ',

'Reseñas restantes: ', reseñas\_restantes, '. ',

'Promedio actual: ', ROUND(promedio\_actual, 2)

);

END IF;

END IF;

END IF;

END //

DELIMITER ;

-- 6. Agregar un nuevo producto, verificando que no exista un duplicado

DELIMITER //

CREATE PROCEDURE agregar\_producto(

IN p\_nombre VARCHAR(100),

IN p\_descripcion VARCHAR(100),

IN p\_precio DECIMAL(10,2),

IN p\_stock INT,

IN p\_id\_categoria INT,

OUT p\_resultado VARCHAR(200)

)

BEGIN

DECLARE producto\_existente INT;

-- Verificar si ya existe un producto con el mismo nombre y categoría

SELECT COUNT(\*) INTO producto\_existente

FROM productos

WHERE nombre = p\_nombre AND id\_categoria = p\_id\_categoria;

IF producto\_existente > 0 THEN

SET p\_resultado = 'Error: Ya existe un producto con ese nombre en esta categoría';

ELSEIF p\_precio <= 0 THEN

SET p\_resultado = 'Error: El precio debe ser mayor que cero';

ELSEIF p\_stock < 0 THEN

SET p\_resultado = 'Error: El stock no puede ser negativo';

ELSE

-- Insertar el nuevo producto

INSERT INTO productos (nombre, descripcion, precio, stock, id\_categoria)

VALUES (p\_nombre, p\_descripcion, p\_precio, p\_stock, p\_id\_categoria);

SET p\_resultado = CONCAT('Producto "', p\_nombre, '" agregado exitosamente con ID: ', LAST\_INSERT\_ID());

END IF;

END //

DELIMITER ;

-- 7. Actualizar la información de un cliente

DELIMITER //

CREATE PROCEDURE actualizar\_cliente(

IN p\_id\_cliente INT,

IN p\_nuevo\_telefono VARCHAR(20),

IN p\_nueva\_direccion VARCHAR(100),

OUT p\_resultado VARCHAR(200)

)

BEGIN

DECLARE cliente\_existente INT;

SELECT COUNT(\*) INTO cliente\_existente FROM clientes WHERE id\_cliente = p\_id\_cliente;

IF cliente\_existente = 0 THEN

SET p\_resultado = 'Error: Cliente no encontrado';

ELSE

UPDATE clientes

SET telefono = p\_nuevo\_telefono, direccion = p\_nueva\_direccion

WHERE id\_cliente = p\_id\_cliente;

SET p\_resultado = CONCAT('Información del cliente ', p\_id\_cliente, ' actualizada exitosamente');

END IF;

END //

DELIMITER ;

-- 8. Generar un reporte de productos con stock bajo

DELIMITER //

CREATE PROCEDURE reporte\_stock\_bajo()

BEGIN

SELECT

p.id\_producto,

p.nombre AS producto,

c.nombre AS categoria,

p.stock,

p.precio

FROM

productos p

JOIN

categorias c ON p.id\_categoria = c.id\_categoria

WHERE

p.stock < 5

ORDER BY

p.stock ASC;

END //

DELIMITER ;

**RESULTADOS DE PRUEBAS Y ANÁLISIS DE RENDIMIENTO**

Las pruebas de los scripts, consultas y procedimientos se realizaron con MySQL Workbench y se muestran los resultados a continuación:

**Salida de la ejecución del script de creación de la base de datos:**

# Time Action Message Duration / Fetch

3 1 20:17:54 CREATE DATABASE IF NOT EXISTS tienda\_electronicos 1 row(s) affected 0.125 sec

3 2 20:17:54 USE tienda\_electronicos 0 row(s) affected 0.000 sec

3 3 20:17:54 CREATE TABLE clientes (

id\_cliente INT PRIMARY KEY AUTO\_INCREMENT,

nombre VARCHAR(100) NOT NULL,

correo VARCHAR(100) UNIQUE NOT NULL,

telefono VARCHAR(20),

direccion VARCHAR(100),

INDEX indice\_correo (correo) -- indice para busquedas por correo

) 0 row(s) affected 0.203 sec

3 4 20:17:54 CREATE TABLE categorias (

id\_categoria INT PRIMARY KEY AUTO\_INCREMENT,

nombre VARCHAR(50) NOT NULL,

descripcion VARCHAR(100),

INDEX indice\_nombre\_categoria (nombre) -- indice para busqueda por nombre de categoria

) 0 row(s) affected 0.079 sec

3 5 20:17:54 CREATE TABLE productos (

id\_producto INT PRIMARY KEY AUTO\_INCREMENT,

nombre VARCHAR(100) NOT NULL,

descripcion VARCHAR(100),

precio DECIMAL(10,2) NOT NULL,

stock INT NOT NULL DEFAULT 0 CHECK (stock >= 0), -- restriccion stock no negativo,

id\_categoria INT NOT NULL,

FOREIGN KEY (id\_categoria) REFERENCES categorias(id\_categoria),

INDEX indice\_nombre\_producto (nombre), -- indice para busqueda por nombre de producto

INDEX indice\_categoria\_producto (id\_categoria) -- indice para busqueda por categoria

) 0 row(s) affected 0.140 sec

3 6 20:17:54 CREATE TABLE pedidos (

id\_pedido INT PRIMARY KEY AUTO\_INCREMENT,

id\_cliente INT NOT NULL,

fecha\_pedido DATE DEFAULT (CURRENT\_DATE),

estado ENUM('pendiente', 'procesando', 'enviado', 'entregado', 'cancelado') DEFAULT 'pendiente',

FOREIGN KEY (id\_cliente) REFERENCES clientes(id\_cliente),

INDEX indice\_cliente\_pedido (id\_cliente), -- iindice para búsquedas de pedidos por cliente

INDEX indice\_estado\_pedido (estado) -- indice para busqueda por estado de pedido

) 0 row(s) affected 0.141 sec

3 7 20:17:55 CREATE TABLE detalles\_pedido (

id\_detalle INT PRIMARY KEY AUTO\_INCREMENT,

id\_pedido INT NOT NULL,

id\_producto INT NOT NULL,

cantidad INT NOT NULL CHECK (cantidad > 0), -- restriccion para cantidad positiva

precio\_unitario DECIMAL(10,2) NOT NULL,

FOREIGN KEY (id\_pedido) REFERENCES pedidos(id\_pedido),

FOREIGN KEY (id\_producto) REFERENCES productos(id\_producto),

INDEX indice\_pedido\_detalle (id\_pedido) -- indice para busquedas de detalles por pedidoa

) 0 row(s) affected 0.187 sec

3 8 20:17:55 CREATE TABLE reseñas (

id\_reseña INT PRIMARY KEY AUTO\_INCREMENT,

id\_producto INT NOT NULL,

id\_cliente INT NOT NULL,

calificacion INT NOT NULL CHECK (calificacion BETWEEN 1 AND 5),

comentario VARCHAR(100),

fecha DATE DEFAULT (CURRENT\_DATE),

FOREIGN KEY (id\_producto) REFERENCES productos(id\_producto),

FOREIGN KEY (id\_cliente) REFERENCES clientes(id\_cliente),

INDEX indice\_producto\_reseña (id\_producto) -- indice para busquedas de reseñas por producto

) 0 row(s) affected 0.219 sec

**Salida de la ejecución del script de inserción de datos de prueba:**

# Time Action Message Duration / Fetch

3 1 14:06:50 USE tienda\_electronicos 0 row(s) affected 0.000 sec

3 2 14:06:50 INSERT INTO categorias (nombre, descripcion) VALUES

('Smartphones', 'Teléfonos inteligentes de última generación'),

('Laptops', 'Computadoras portátiles para trabajo y entretenimiento'),

('Televisores', 'Pantallas para el hogar con diferentes tecnologías'),

('Audio', 'Equipos de sonido y accesorios'),

('Gaming', 'Consolas y accesorios para videojuegos'),

('Electrodomésticos', 'Aparatos electrónicos para el hogar') 6 row(s) affected

Records: 6 Duplicates: 0 Warnings: 0 0.016 sec

3 3 14:06:50 INSERT INTO productos (nombre, descripcion, precio, stock, id\_categoria) VALUES

-- Smartphones

('iPhone 15 Pro', 'Apple iPhone 15 Pro 256GB Plata', 24999.00, 50, 1),

('Samsung Galaxy S23 Ultra', 'Samsung Galaxy S23 Ultra 256GB Negro', 22999.00, 4, 1),

('Xiaomi Redmi Note 12', 'Xiaomi Redmi Note 12 128GB Azul', 5499.00, 80, 1),

('Motorola Edge 40', 'Motorola Edge 40 256GB Verde', 8999.00, 30, 1),

('Huawei P60 Pro', 'Huawei P60 Pro 512GB Oro', 18999.00, 25, 1),

-- Laptops

('MacBook Pro 14" M2', 'Apple MacBook Pro 14" M2 512GB Plata', 37999.00, 20, 2),

('Dell XPS 15', 'Dell XPS 15 i7 16GB 512GB Negro', 32999.00, 1, 2),

('HP Spectre x360', 'HP Spectre x360 i5 8GB 256GB Azul', 21999.00, 18, 2),

('Lenovo ThinkPad X1', 'Lenovo ThinkPad X1 Carbon i7 16GB 1TB Negro', 28999.00, 12, 2),

('Asus ROG Zephyrus', 'Asus ROG Zephyrus G15 Ryzen 9 32GB 1TB Gris', 41999.00, 10, 2),

-- Televisores

('LG OLED C3 55"', 'LG OLED C3 55" 4K Smart TV', 24999.00, 25, 3),

('Samsung QN90B 65"', 'Samsung Neo QLED QN90B 65" 4K', 32999.00, 1, 3),

('Sony X80K 50"', 'Sony Bravia X80K 50" 4K Android TV', 12999.00, 30, 3),

('TCL 55C735 55"', 'TCL 55C735 55" QLED Google TV', 15999.00, 22, 3),

('Hisense U7H 65"', 'Hisense U7H 65" ULED Android TV', 21999.00, 15, 3),

-- Audio

('Sony WH-1000XM5', 'Audífonos inalámbricos Sony WH-1000XM5', 6999.00, 40, 4),

('Bose QuietComfort 45', 'Audífonos Bose QuietComfort 45', 6499.00, 3, 4),

('JBL Flip 6', 'Bocina portátil JBL Flip 6 Azul', 1999.00, 60, 4),

('Sony HT-S400', 'Soundbar Sony HT-S400 2.1 canales', 3999.00, 25, 4),

('Samsung HW-Q600A', 'Soundbar Samsung HW-Q600A 3.1.2 canales', 5999.00, 20, 4),

-- Gaming

('PlayStation 5', 'Consola PlayStation 5 825GB', 12999.00, 30, 5),

('Xbox Series X', 'Consola Xbox Series X 1TB', 11999.00, 2, 5),

('Nintendo Switch OLED', 'Consola Nintendo Switch OLED Blanco', 8999.00, 35, 5),

('Logitech G Pro X', 'Audífonos gaming Logitech G Pro X', 3499.00, 50, 5),

('Razer BlackWidow V3', 'Teclado mecánico Razer BlackWidow V3', 2999.00, 45, 5),

-- Electrodomésticos

('LG InstaView', 'Refrigerador LG InstaView 23 pies', 32999.00, 10, 6),

('Samsung Bespoke', 'Lavadora Samsung Bespoke 20kg', 17999.00, 2, 6),

('Mabe RME1430XMN', 'Microondas Mabe 1.4 pies', 2999.00, 30, 6),

('Philips Airfryer', 'Freidora de aire Philips HD9650', 4999.00, 25, 6),

('Oster Xpert', 'Licuadora Oster Xpert 8 velocidades', 1999.00, 40, 6) 30 row(s) affected

Records: 30 Duplicates: 0 Warnings: 0 0.015 sec

3 4 14:06:50 INSERT INTO clientes (nombre, correo, telefono, direccion) VALUES

('Juan Pérez López', 'juan.perez@gmail.com', '5551234567', 'Av. Insurgentes Sur 123, Col. Roma Norte, CDMX'),

('María González Hernández', 'maria.gonzalez@hotmail.com', '5559876543', 'Calle Morelos 45, Col. Centro, Guadalajara'),

('Carlos Sánchez Martínez', 'carlos.sanchez@yahoo.com', '5554567890', 'Blvd. Díaz Ordaz 789, Col. Del Valle, Monterrey'),

('Ana Rodríguez García', 'ana.rodriguez@gmail.com', '5552345678', 'Av. Revolución 567, Col. Tacubaya, CDMX'),

('Luis Fernández Jiménez', 'luis.fernandez@outlook.com', '5558765432', 'Calle Hidalgo 34, Col. Americana, Guadalajara'),

('Patricia Ramírez Díaz', 'paty.ramirez@gmail.com', '5553456789', 'Av. Constitución 890, Col. Moderna, Puebla'),

('Jorge Torres Ruiz', 'jorge.torres@hotmail.com', '5557654321', 'Blvd. Benito Juárez 123, Col. Narvarte, CDMX'),

('Sofía Mendoza Castro', 'sofia.mendoza@yahoo.com', '5556543210', 'Av. Universidad 456, Col. Del Carmen, Coyoacán'),

('Miguel Ángel Vargas', 'miguel.vargas@gmail.com', '5554321098', 'Calle 5 de Mayo 67, Col. Centro, Querétaro'),

('Laura Gómez Silva', 'laura.gomez@outlook.com', '5553210987', 'Av. López Mateos 789, Col. San Pedro, Tijuana'),

('Roberto Navarro Ortega', 'roberto.navarro@hotmail.com', '5552109876', 'Blvd. Aguacaliente 345, Col. Chapultepec, Aguascalientes'),

('Isabel Cruz Mendoza', 'isabel.cruz@gmail.com', '5551098765', 'Av. Tecnológico 901, Col. Linda Vista, Chihuahua'),

('Fernando Herrera Luna', 'fer.herrera@yahoo.com', '5550987654', 'Calle Zaragoza 234, Col. Juárez, Hermosillo'),

('Adriana Morales Ríos', 'adri.morales@hotmail.com', '5559876543', 'Av. Las Torres 567, Col. Las Brisas, León'),

('Ricardo Castro Méndez', 'ricardo.castro@gmail.com', '5558765432', 'Blvd. Luis Donaldo Colosio 890, Col. Las Palmas, Cancún') 15 row(s) affected

Records: 15 Duplicates: 0 Warnings: 0 0.016 sec

3 5 14:06:50 INSERT INTO pedidos (id\_cliente, fecha\_pedido, estado) VALUES

(1, '2023-10-15', 'entregado'),

(2, '2023-10-16', 'entregado'),

(3, '2023-10-17', 'enviado'),

(4, '2023-10-18', 'procesando'),

(5, '2023-10-19', 'pendiente'),

(6, '2023-10-20', 'entregado'),

(7, '2023-10-21', 'cancelado'),

(8, '2023-10-22', 'entregado'),

(9, '2023-10-23', 'enviado'),

(10, '2023-10-24', 'procesando'),

(11, '2023-10-25', 'pendiente'),

(12, '2023-10-26', 'entregado'),

(13, '2023-10-27', 'enviado'),

(14, '2023-10-28', 'procesando'),

(15, '2023-10-29', 'pendiente'),

(1, '2023-10-30', 'entregado'),

(3, '2023-10-31', 'enviado'),

(5, '2023-11-01', 'procesando'),

(7, '2023-11-02', 'cancelado'),

(9, '2023-11-03', 'entregado') 20 row(s) affected

Records: 20 Duplicates: 0 Warnings: 0 0.000 sec

3 6 14:11:04 INSERT INTO detalles\_pedido (id\_pedido, id\_producto, cantidad, precio\_unitario) VALUES

-- Order 1

(1, 1, 1, 24999.00),

(1, 16, 1, 6999.00),

-- Order 2

(2, 6, 1, 37999.00),

-- Order 3

(3, 2, 1, 22999.00),

(3, 21, 1, 12999.00),

(3, 25, 1, 2999.00),

-- Order 4

(4, 3, 2, 5499.00),

-- Order 5

(5, 7, 1, 32999.00),

(5, 17, 1, 6499.00),

-- Order 6

(6, 11, 1, 24999.00),

-- Order 7

(7, 22, 1, 11999.00),

-- Order 8

(8, 4, 1, 8999.00),

(8, 18, 2, 1999.00),

-- Order 9

(9, 8, 1, 21999.00),

(9, 23, 1, 8999.00),

-- Order 10

(10, 12, 1, 32999.00),

-- Order 11

(11, 5, 1, 18999.00),

(11, 19, 1, 3999.00),

-- Order 12

(12, 9, 1, 28999.00),

-- Order 13

(13, 13, 1, 12999.00),

(13, 24, 1, 3499.00),

-- Order 14

(14, 14, 1, 15999.00),

-- Order 15

(15, 10, 1, 41999.00),

(15, 20, 1, 5999.00),

-- Order 16

(16, 26, 1, 32999.00) 25 row(s) affected

Records: 25 Duplicates: 0 Warnings: 0 0.000 sec

3 7 14:21:02 INSERT INTO reseñas (id\_producto, id\_cliente, calificacion, comentario, fecha) VALUES

(1, 1, 2, 'Imma keep it a buck fifty with you, G, the phone is dog water.', '2023-10-20'),

(6, 2, 4, 'Vey well made laptop, however, it is slightly expensive', '2023-10-21'),

(11, 6, 5, 'This is indubitably the greatest intelligent phone I have ever used', '2023-10-25'),

(16, 1, 5, 'The headphones cancel all external noise perfectly', '2023-10-22'),

(21, 3, 4, 'PS5 functions quite well, but it has limited games', '2023-10-28'),

(2, 3, 3, 'Keep it a stack, phone is great, BUT the battery runs out fast', '2023-10-30'),

(7, 5, 5, 'I LOVE refrigeratorssss ', '2023-11-02'),

(12, 8, 4, 'Good evening sir. Would you like to hear about our Lord and Saviour, Jesus Christ--JehovaWitness', '2023-10-27'),

(17, 9, 5, 'Was that the bite of 87???!!!', '2023-11-01'),

(22, 7, 2, 'Say my name. You´re Heisenberg. You´re goddamn right', '2023-10-29') 10 row(s) affected

Records: 10 Duplicates: 0 Warnings: 0 0.046 sec

**Salida de la ejecución de consultas y procedimientos almacenados:**

-- -----------------------------------------------------

-- CONSULTAS

-- -----------------------------------------------------

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

3 32 14:53:29 SELECT

c.nombre AS categoria,

p.nombre AS producto,

p.descripcion,

p.precio,

p.stock

FROM

productos p

JOIN

categorias c ON p.id\_categoria = c.id\_categoria

WHERE

p.stock > 0

ORDER BY

c.nombre, p.precio

LIMIT 0, 1000 30 row(s) returned 0.000 sec / 0.000 sec

categoria producto descripcion precio stock

Audio JBL Flip 6 Bocina portátil JBL Flip 6 Azul 1999.00 60

Audio Sony HT-S400 Soundbar Sony HT-S400 2.1 canales 3999.00 25

Audio Samsung HW-Q600A Soundbar Samsung HW-Q600A 3.1.2 canales 5999.00 20

Audio Bose QuietComfort 45 Audífonos Bose QuietComfort 45 6499.00 3

Audio Sony WH-1000XM5 Audífonos inalámbricos Sony WH-1000XM5 6999.00 40

Electrodomésticos Oster Xpert Licuadora Oster Xpert 8 velocidades 1999.00 40

Electrodomésticos Mabe RME1430XMN Microondas Mabe 1.4 pies 2999.00 30

Electrodomésticos Philips Airfryer Freidora de aire Philips HD9650 4999.00 25

Electrodomésticos Samsung Bespoke Lavadora Samsung Bespoke 20kg 17999.00 2

Electrodomésticos LG InstaView Refrigerador LG InstaView 23 pies 32999.00 10

Gaming Razer BlackWidow V3 Teclado mecánico Razer BlackWidow V3 2999.00 45

Gaming Logitech G Pro X Audífonos gaming Logitech G Pro X 3499.00 50

Gaming Nintendo Switch OLED Consola Nintendo Switch OLED Blanco 8999.00 35

Gaming Xbox Series X Consola Xbox Series X 1TB 11999.00 2

Gaming PlayStation 5 Consola PlayStation 5 825GB 12999.00 30

Laptops HP Spectre x360 HP Spectre x360 i5 8GB 256GB Azul 21999.00 18

Laptops Lenovo ThinkPad X1 Lenovo ThinkPad X1 Carbon i7 16GB 1TB Negro 28999.00 12

Laptops Dell XPS 15 Dell XPS 15 i7 16GB 512GB Negro 32999.00 1

Laptops MacBook Pro 14" M2 Apple MacBook Pro 14" M2 512GB Plata 37999.00 20

Laptops Asus ROG Zephyrus Asus ROG Zephyrus G15 Ryzen 9 32GB 1TB Gris 41999.00 10

Smartphones Xiaomi Redmi Note 12 Xiaomi Redmi Note 12 128GB Azul 5499.00 80

Smartphones Motorola Edge 40 Motorola Edge 40 256GB Verde 8999.00 30

Smartphones Huawei P60 Pro Huawei P60 Pro 512GB Oro 18999.00 25

Smartphones Samsung Galaxy S23 Ultra Samsung Galaxy S23 Ultra 256GB Negro 22999.00 4

Smartphones iPhone 15 Pro Apple iPhone 15 Pro 256GB Plata 24999.00 50

Televisores Sony X80K 50" Sony Bravia X80K 50" 4K Android TV 12999.00 30

Televisores TCL 55C735 55" TCL 55C735 55" QLED Google TV 15999.00 22

Televisores Hisense U7H 65" Hisense U7H 65" ULED Android TV 21999.00 15

Televisores LG OLED C3 55" LG OLED C3 55" 4K Smart TV 24999.00 25

Televisores Samsung QN90B 65" Samsung Neo QLED QN90B 65" 4K 32999.00 1

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

3 33 15:13:41 SELECT cl.id\_cliente, cl.nombre, cl.correo, COUNT(p.id\_pedido) AS pedidos\_pendientes, SUM(dp.cantidad \* dp.precio\_unitario) AS total\_compras FROM clientes cl JOIN pedidos p ON cl.id\_cliente = p.id\_cliente JOIN detalles\_pedido dp ON p.id\_pedido = dp.id\_pedido WHERE p.estado = 'pendiente' GROUP BY cl.id\_cliente, cl.nombre, cl.correo LIMIT 0, 1000 3 row(s) returned 0.016 sec / 0.000 sec

Id\_cliente nombre correo pedidos\_pendientes total\_compras

5 Luis Fernández Jiménez luis.fernandez@outlook.com 2 39498.00

11 Roberto Navarro Ortega roberto.navarro@hotmail.com 2 22998.00

15 Ricardo Castro Méndez ricardo.castro@gmail.com 2 47998.00

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

3 34 15:27:22 SELECT

p.id\_producto,

p.nombre,

AVG(r.calificacion) AS calificacion\_promedio,

COUNT(r.id\_reseña) AS total\_reseñas

FROM

productos p

JOIN

reseñas r ON p.id\_producto = r.id\_producto

GROUP BY

p.id\_producto, p.nombre

ORDER BY

calificacion\_promedio DESC

LIMIT 5 5 row(s) returned 0.000 sec / 0.000 sec

Id nombre calificación total reseñas

7 Dell XPS 15 5.0000 1

11 LG OLED C3 55" 5.0000 1

16 Sony WH-1000XM5 5.0000 1

17 Bose QuietComfort 45 5.0000 1

6 MacBook Pro 14" M2 4.0000 4.0000 1

-- -----------------------------------------------------

-- PROCEDIMIENTOS ALMACENADOS

-- -----------------------------------------------------

*1. registrar\_pedido*

-- Prueba de registro de pedido exitoso

CALL registrar\_pedido(1, 5, 1, @resultado);

17:03:51 CALL registrar\_pedido(1, 5, 1, @resultado) 1 row(s) affected 0.062 sec

SELECT @resultado;

'Pedido registrado exitosamente'

-- Prueba de intento de registro de un pedido de un producto que no tiene stock suficiente

CALL registrar\_pedido(1, 5, 100, @resultado);

17:25:10 CALL registrar\_pedido(1, 5, 100, @resultado) 1 row(s) affected 0.047 sec

SELECT @resultado;

'Error: Stock insuficiente. Disponible: 24'

-- Prueba cuando hay muchos pedidos pendientes de un cliente (es necesario registrar más pedidos antes de hacer la prueba para que se cumpla la condición de error de que no se puede hacer un pedido cuando el cliente ya tiene 5 pedidos pendientes)

*2. registrar\_reseña*

-- Prueba de registro de reseña válido

CALL registrar\_reseña(1, 5, 4, 'Excelente producto', @resultado);

17:40:50 CALL registrar\_reseña(1, 5, 4, 'Excelente producto', @resultado) 1 row(s) affected 0.047 sec

SELECT @resultado;

'Reseña registrada exitosamente'

-- Prueba de intento de registro de reseña de un cliente que no ha comprado el producto

CALL registrar\_reseña(15, 1, 5, 'No lo he comprado', @resultado);

17:44:05 CALL registrar\_reseña(15, 1, 5, 'No lo he comprado', @resultado) 1 row(s) affected 0.000 sec

SELECT @resultado;

'Error: El cliente no ha comprado este producto'

-- Prueba de intento de registro de reseña con una calificación fuera del rango

CALL registrar\_reseña(1, 5, 6, ‘Calificación\_inválida’, @resultado);

08:54:42 CALL registrar\_reseña(1, 5, 6, 'Calificación\_inválida', @resultado) 1 row(s) affected 0.000 sec

SELECT @resultado;

@resultado

'Error: La calificación debe estar entre 1 y 5'

*3. actualizar\_stock\_pedido*

-- Imprime el stock antes de ejecutar el procedimiento almacenado

SELECT stock FROM productos WHERE id\_producto = 5;

17:51:57 SELECT stock FROM productos WHERE id\_producto = 5 LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec

Stock

24

-- Prueba de una actualización del stock válida

CALL actualizar\_stock\_pedido(5, 2, @resultado);

18:05:56 CALL actualizar\_stock\_pedido(5, 2, @resultado) 1 row(s) affected 0.015 sec

SELECT @resultado;

'Stock actualizado. Nuevo stock: 22'

-- Imprime el stock después de ejecutar el procedimiento almacenado

SELECT stock FROM productos WHERE id\_producto = 5;

18:14:05 SELECT stock FROM productos WHERE id\_producto = 5 LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec

stock

22

-- Prueba de intento de actualizar el stock cuando es insuficiente

CALL actualizar\_stock\_pedido(5, 1000, @resultado);

18:18:28 SELECT @resultado LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec

SELECT @resultado;

@resultado

'Stock actualizado. Nuevo stock: 22'

*4. cambiar\_estado\_pedido*

-- Prueba de cambio de estado de pedido exitoso

CALL cambiar\_estado\_pedido(3, 'enviado', @resultado);

20:37:42 CALL cambiar\_estado\_pedido(3, 'enviado', @resultado) 0 row(s) affected 0.031 sec

SELECT @resultado;

'Estado del pedido 3 cambiado a enviado'

-- Verifica que el cambio del estado del pedido

SELECT estado FROM pedidos WHERE id\_pedido = 3;

20:39:21 SELECT estado FROM pedidos WHERE id\_pedido = 3 LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec

estado

enviado

-- Prueba de intento de cambio de estado de un pedido, usando un estado inválido

CALL cambiar\_estado\_pedido(3, ‘estado\_invalido‘, @resultado);

21:13:47 CALL cambiar\_estado\_pedido(3, 'estado\_invalido', @resultado) 1 row(s) affected 0.000 sec

SELECT @resultado;

'Estado del pedido 3 cambiado a enviado'

*5. eliminar\_reseñas\_producto*

-- Agregando reseñas de producto antes de probar el procedimiento almacenado

INSERT INTO reseñas (id\_producto, id\_cliente, calificacion, comentario) VALUES

(5, 1, 5, 'Excelente'),

(5, 2, 4, 'Muy bueno'),

(5, 3, 3, 'Regular');

21:18:29 INSERT INTO reseñas (id\_producto, id\_cliente, calificacion, comentario) VALUES (5, 1, 5, 'Excelente'), (5, 2, 4, 'Muy bueno'), (5, 3, 3, 'Regular') 3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0 0.062 sec

-- Verifica las reseñas del producto 5 antes de probar el procedimiento almacenado

SELECT \* FROM reseñas WHERE id\_producto = 5;

id\_reseña, id\_producto, id\_cliente, calificacion, comentario, fecha

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 11 | 5 | 1 | 4 | Excelente producto | 2025-08-02 |
| 12 | 5 | 1 | 5 | Excelente | 2025-08-02 |
| 13 | 5 | 2 | 4 | Muy bueno | 2025-08-02 |
| 14 | 5 | 3 | 3 | Regular | 2025-08-02 |
|  |  |  |  |  |  |

-- Prueba exitosa de eliminar una reseña

CALL eliminar\_reseñas\_producto(5, LAST\_INSERT\_ID(), @resultado);

22:09:02 CALL eliminar\_reseñas\_producto(5, LAST\_INSERT\_ID(), @resultado) 1 row(s) affected 0.031 sec

SELECT @resultado;

'Error: Estado no válido'

-- Verifica las reseñas restantes de este producto y su promedio

SELECT \* FROM reseñas WHERE id\_producto = 5;

id\_reseña, id\_producto, id\_cliente, calificacion, comentario, fecha

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 11 | 5 | 1 | 4 | Excelente producto | 2025-08-02 |
| 12 | 5 | 1 | 5 | Excelente | 2025-08-02 |
| 13 | 5 | 2 | 4 | Muy bueno | 2025-08-02 |
| 14 | 5 | 3 | 3 | Regular | 2025-08-02 |
| 15 | 5 | 1 | 5 | Excelente | 2025-08-02 |
| 16 | 5 | 2 | 4 | Muy bueno | 2025-08-02 |
| 17 | 5 | 3 | 3 | Regular | 2025-08-02 |
|  |  |  |  |  |  |

SELECT AVG(calificacion) FROM reseñas WHERE id\_producto = 5;

AVG(calificacion)

'4.0000'

-- Prueba de borrar todas las reseñas restantes del producto

CALL eliminar\_reseñas\_producto(5, NULL, @resultado);

22:09:11 CALL eliminar\_reseñas\_producto(5, NULL, @resultado) 6 row(s) affected 0.031 sec

SELECT @resultado;

@resultado

'Error: Estado no válido'

-- Verifica que el producto ya no tiene reseñas

SELECT \* FROM reseñas WHERE id\_producto = 5;

id\_reseña, id\_producto, id\_cliente, calificacion, comentario, fecha

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 11 | 5 | 1 | 4 | Excelente producto | 2025-08-02 |
| 12 | 5 | 1 | 5 | Excelente | 2025-08-02 |
| 13 | 5 | 2 | 4 | Muy bueno | 2025-08-02 |
| 14 | 5 | 3 | 3 | Regular | 2025-08-02 |
| 15 | 5 | 1 | 5 | Excelente | 2025-08-02 |
| 16 | 5 | 2 | 4 | Muy bueno | 2025-08-02 |
| 17 | 5 | 3 | 3 | Regular | 2025-08-02 |
|  |  |  |  |  |  |

*6. agregar\_producto*

-- Prueba exitosa de agregar un producto

CALL agregar\_producto('Producto de prueba’, 'Descripcion producto de prueba’, 999.99, 10, 1, @resultado);

22:48:17 CALL agregar\_producto('Producto\_de\_prueba', 'Descripcion\_producto\_de\_prueba', 999.99, 10, 1, @resultado) 1 row(s) affected 0.062 sec

SELECT @resultado;

@resultado

'Producto \"Producto\_de\_prueba\" agregado exitosamente con ID: 31'

-- Verifica que se agregó el producto

SELECT \* FROM productos WHERE nombre = 'Producto de prueba‘;

id\_producto, nombre, descripcion, precio, stock, id\_categoria

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31 | Producto\_de\_prueba | Descripcion\_producto\_de\_prueba | 999.99 | 10 | 1 |
|  |  |  |  |  |  |

-- Intento de agregar producto duplicado

CALL agregar\_producto('Producto de prueba‘, 'Otra descripcion', 888.88, 5, 1, @resultado);

22:55:26 CALL agregar\_producto('Producto\_de\_prueba', 'Otra\_descripcion', 888.88, 5, 1, @resultado) 1 row(s) affected 0.032 sec

SELECT @resultado;

@resultado

'Error: Ya existe un producto con ese nombre en esta categoría'

-- Intento de agregar producto con precio negativo

CALL agregar\_producto('Producto Inválido', 'Test', -10, 5, 1, @resultado);

22:59:35 CALL agregar\_producto('Producto\_Invalido', 'Test', -10, 5, 1, @resultado) 1 row(s) affected 0.000 sec 'Error: El precio debe ser mayor que cero'

SELECT @resultado;

@resultado

'Error: El precio debe ser mayor que cero'

-- Intento de agregar producto con stock negativo

CALL agregar\_producto('Producto\_Inválido2', 'Test', 1999.99, -5, 1, @resultado);

23:02:31 CALL agregar\_producto('Producto\_Invalido2', 'Test', 1999.99, -5, 1, @resultado) 1 row(s) affected 0.000 sec

SELECT @resultado;

@resultado

'Error: El stock no puede ser negativo'

*7. actualizar\_cliente*

-- Despliega la información del cliente 1 antes de ejecutar la prueba del procedimiento almacenado

SELECT \* FROM clientes WHERE id\_cliente = 1;

id\_cliente, nombre, correo, telefono, direccion

'1', 'Juan Pérez López', 'juan.perez@gmail.com', '5551234567', 'Av. Insurgentes Sur 123, Col. Roma Norte, CDMX'

-- Prueba exitosa de actualizar el teléfono y la dirección del cliente

CALL actualizar\_cliente(1, '5551112233', 'Nueva dirección test', @resultado);

08:41:20 CALL actualizar\_cliente(1, '5551112233', 'Nueva dirección test', @resultado) 1 row(s) affected 0.046 sec

SELECT @resultado;

@resultado

'Información del cliente 1 actualizada exitosamente'

-- Verifica la actualización de los datos del cliente

SELECT \* FROM clientes WHERE id\_cliente = 1;

id\_cliente, nombre, correo, telefono, direccion

'1', 'Juan Pérez López', 'juan.perez@gmail.com', '5551112233', 'Nueva dirección test'

-- Intento de actualizar los datos de un cliente que no existe

CALL actualizar\_cliente(999, '5550000000', 'No existe', @resultado);

08:42:52 CALL actualizar\_cliente(999, '5550000000', 'No existe', @resultado) 1 row(s) affected 0.000 sec

SELECT @resultado;

@resultado

'Error: Cliente no encontrado'

*8. reporte\_stock\_bajo*

-- Prueba el procedimiento almacenado

CALL reporte\_stock\_bajo();

id\_producto, producto, categoria, stock, precio

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | Dell XPS 15 | Laptops | 1 | 32999.00 |
| 12 | Samsung QN90B 65" | Televisores | 1 | 32999.00 |
| 27 | Samsung Bespoke | Electrodomésticos | 2 | 17999.00 |
| 22 | Xbox Series X | Gaming | 2 | 11999.00 |
| 17 | Bose QuietComfort 45 | Audio | 3 | 6499.00 |
| 2 | Samsung Galaxy S23 Ultra | Smartphones | 4 | 22999.00 |

**CONCLUSIONES Y LECCIONES APRENDIDAS**.

Se creó y se pobló exitosamente la base de datos. Se cumplieron los objetivos solicitados y los objetivos funcionales.

Áreas de mejora: Manejo de transacciones en los almacenamientos que se requieran para mantener la integridad de la B.D.

Por ejemplo, en el procedimiento “registrar pedido” se debe insertar un registro en pedidos, un registro en la tabla de “detalles de pedidos” y hacer un update a la tabla de productos para actualizar la cantidad del inventario; las tres instrucciones deben de ejecutarse todas o ninguna (true or false; 0 or 1) por lo que hay que usar transacciones. El código quedaría de la siguiente manera:

DELIMITER //

CREATE PROCEDURE registrar\_pedido(

IN p\_id\_cliente INT,

IN p\_id\_producto INT,

IN p\_cantidad INT,

OUT p\_resultado VARCHAR(200)

BEGIN

DECLARE pedidos\_pendientes INT;

DECLARE stock\_actual INT;

DECLARE precio\_producto DECIMAL(10,2);

DECLARE v\_id\_pedido INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = CONCAT('Error: ', IFNULL(SQLERRM, 'Transacción fallida'));

END;

-- Verificar restricciones antes de iniciar transacción

SELECT COUNT(\*) INTO pedidos\_pendientes

FROM pedidos

WHERE id\_cliente = p\_id\_cliente AND estado = 'pendiente';

SELECT stock, precio INTO stock\_actual, precio\_producto

FROM productos

WHERE id\_producto = p\_id\_producto

IF pedidos\_pendientes >= 5 THEN

SET p\_resultado = 'Error: El cliente ya tiene 5 pedidos pendientes';

ELSEIF stock\_actual < p\_cantidad THEN

SET p\_resultado = CONCAT('Error: Stock insuficiente. Disponible: ', stock\_actual);

ELSE

START TRANSACTION;

-- 1. Registrar el pedido

INSERT INTO pedidos (id\_cliente, estado)

VALUES (p\_id\_cliente, 'pendiente');

SET v\_id\_pedido = LAST\_INSERT\_ID();

-- 2. Registrar detalle del pedido

INSERT INTO detalles\_pedido (id\_pedido, id\_producto, cantidad, precio\_unitario)

VALUES (v\_id\_pedido, p\_id\_producto, p\_cantidad, precio\_producto);

-- 3. Actualizar stock

UPDATE productos

SET stock = stock - p\_cantidad

WHERE id\_producto = p\_id\_producto;

COMMIT;

SET p\_resultado = CONCAT('Pedido ', v\_id\_pedido, ' registrado exitosamente');

END IF;

END //

DELIMITER ;

¿Cuándo se deben de usan los índices?

Los índices se recomiendan de usar cuando hay al menos mil registros en una tabla, lo que todavía no cumplen las tablas en nuestra base de datos. Sí puedes usar índices cuando hay menos registros si hay tablas de referencia que son consultadas frecuentemente.

Los índices creados en las tablas de la base de datos y su propósito son los siguientes:

|  |  |  |
| --- | --- | --- |
| Nombre\_Tabla | Nombre índice | Propósito |
| Clientes | INDEX indice\_correo (correo) | índice para búsquedas por correo |
| Categorías | INDEX indice\_nombre\_categoria (nombre) | índice para búsqueda por nombre de categoría |
| Productos | INDEX indice\_nombre\_producto (nombre),  INDEX indice\_categoria\_producto (id\_categoria) | índice para búsqueda por nombre de producto  índice para búsqueda por categoría |
| Pedidos | INDEX indice\_cliente\_pedido (id\_cliente)  INDEX indice\_estado\_pedido (estado) | índice para búsquedas de pedidos por cliente  índice para búsqueda por estado de pedido |
| Detalles\_pedido | INDEX indice\_pedido\_detalle (id\_pedido) | índice para búsquedas de detalles por pedido |
| Reseñas | INDEX indice\_producto\_reseña (id\_producto) | índice para búsquedas de reseñas por producto |
|  |  |  |

EVALUACIÓN DEL DESEMPEÑO DE LAS CONSULTAS USANDO EXPLAINS

Los explains te muestran cómo se ejecutan consultas en MySQL

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

EXPLAIN

SELECT

c.nombre AS categoria,

p.nombre AS producto,

p.descripcion,

p.precio,

p.stock

FROM

productos p

JOIN

categorias c ON p.id\_categoria = c.id\_categoria

WHERE

p.stock > 0

ORDER BY

c.nombre, p.precio;

SALIDA:

id, select\_type, table, partitions, type, possible\_keys, key, key\_len, ref, rows, filtered, Extra

1 SIMPLE c index PRIMARY indice\_nombre\_categoria 202 6 100.00 Using index; Using temporary; Using filesort

1 SIMPLE p ref indice\_categoria\_producto indice\_categoria\_producto 4 tienda\_electronicos.c.id\_categoria 1 33.33 Using where

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

EXPLAIN

SELECT

cl.id\_cliente,

cl.nombre,

cl.correo,

COUNT(p.id\_pedido) AS pedidos\_pendientes,

SUM(dp.cantidad \* dp.precio\_unitario) AS total\_compras

FROM

clientes cl

JOIN

pedidos p ON cl.id\_cliente = p.id\_cliente

JOIN

detalles\_pedido dp ON p.id\_pedido = dp.id\_pedido

WHERE

p.estado = 'pendiente'

GROUP BY

cl.id\_cliente, cl.nombre, cl.correo;

SALIDA

id, select\_type, table, partitions, type, possible\_keys, key, key\_len, ref, rows, filtered, Extra

1 SIMPLE p ref PRIMARY,indice\_cliente\_pedido,indice\_estado\_pedido indice\_estado\_pedido 2 const 4 100.00 Using index condition; Using temporary

1 SIMPLE cl eq\_ref PRIMARY PRIMARY 4 tienda\_electronicos.p.id\_cliente 1 100.00

1 SIMPLE dp ref indice\_pedido\_detalle indice\_pedido\_detalle 4 tienda\_electronicos.p.id\_pedido 1 100.00

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

EXPLAIN

SELECT

p.id\_producto,

p.nombre,

AVG(r.calificacion) AS calificacion\_promedio,

COUNT(r.id\_reseña) AS total\_reseñas

FROM

productos p

JOIN

reseñas r ON p.id\_producto = r.id\_producto

GROUP BY

p.id\_producto, p.nombre

ORDER BY

calificacion\_promedio DESC

LIMIT 5;

SALIDA:

id, select\_type, table, partitions, type, possible\_keys, key, key\_len, ref, rows, filtered, Extra

1 SIMPLE r ALL indice\_producto\_reseña 10 100.00 Using temporary; Using filesort

1 SIMPLE p eq\_ref PRIMARY,indice\_nombre\_producto PRIMARY 4 tienda\_electronicos.r.id\_producto 1 100.00

LECCIONES APRENDIDAS

En la sección de consultas y procesos de almacenamiento requiero investigarlo más a fondo; aunque ya no es la misma dificultad que antes, me falta la práctica.

NOTES

While I do intend to study databases, alongside other programming topics, this vacation period, I am completely burnt out.   
  
 The second trimester was the start of the downwards spiral towards loathing programming, since I did terribly in html/css and beginner programming because the teachers (who shall be unanimous) weren’t the best. Ever since that I got into learning Japanese 日本語 (ni-hon-go), trying to teach English　英語 (ei-go), and working out (activities that I wanted to do for quite a while [especially learning a third language that contrasts the Romantic and Germanic languages of Spanish & English {Mixteco Alto es el siguiente} ] to balance out the difficulties of learning programming, but it has failed.

The fifth trimester fiercely crashed against me, and the three weeks of vacation did jack-all, since I still was studying programming. By the time the sixth one rolled in, I did not have it in me, and by force of will went headfirst into the firery pit; working out (and boxing) keeps me mentally sane from wanting to smash my computer, and dealing with the half-wits who ridicule me for not knowing as much programming (even though I *could* do the same to them for butchering English).

Programming is so monotonic. What is the gist of it? Sure, you get filthy rich, but at the cost of burning out (more than any job) and fake relationships.

My dad, mom (retired), and uncle work like assets, not like humans; they arrive home at 11pm, and sometimes get called to work EVEN MORE until the early morning because the database went down, or something.

I asked my father the other day how he conquered my mom, and he told me that it was because of material wealth. My uncle receives a bunch of gifts just because of his position, not because of him; it’s sad.

Regardless, I’ll take your word, teacher, in trying to find love for programming. Maybe there’s a middle ground between shark science and programming, who knows. I want to throw off this weight of hostility towards programming that depletes my energy, happiness, and concentration; must find *PEACE.*