Documentación TiendaOnline

1. Introducción al problema y objetivos

- **Problema**: Resolver la gestión de las operaciones de una tienda online, esto incluye: clientes, pedidos, reseñas de productos y categorías.
- **Objetivo general**: Generar el diseño y la implementación de una base de datos relacional, funcional y normalizada para la tienda online.
- **Objetivos específicos**:
    - Análisis de requerimientos, identificación de entidades, atributos y relaciones para poder crear un diseño conceptual: Modelo E-R.
    - Generar el diseño logico, convertir el Modelo ER a tablas o esquemas.
    - Aplicar principios de normalización hasta 3FN.
    - Implementar las tablas en SQL con sus respectivas restricciones.
    - Poblar la base de datos con datos de prueba.
    - Crear consultas y procedimientos almacenados.
    - Validación y optimización: ajustar el diseño para que sea eficiente y coherente.
    - Evaluar el rendimiento y funcionamiento del diseño.

2.- Análisis de requerimientos, identificación de entidades, atributos y relaciones.

Análisis de los requerimientos:

- 1.- La base de datos debe tener el registro de los productos con la siguiente información: Nombre, descripción, precio, stock,categoria.
- 2.- Debe contenter la infomación de los clientes: Nombre, Correo electrónico, dirección y numero de telefono.
- 3.- La base de datos debe guardar la infomación relacionada a los pedidos: fechaa, estado(pendiente, enviado, entregado) y los productos que formar el pedido.
- 4.-Debe ser capaz de guardar reseñas, es decir almacenar la calificacion de productos y comentario sobre el producto.

  NOTA: Solo el cliente que compro el producto puede realizar reseñas.

    - 5.- Debe almacenar las categorias de los productos, por ejemplo: telefonos, laptos, accesorios y la descripcion de cada categoria.
- 6.-El stock del inventario no puede estar en negativo.
- 7.-Se debe asegurar la unicidad de los correos electrónicos de los usuarios.

Después de revisar las requerimientos del proyecto, determiné las posibles entidades y sus atributos

Posibles entidades:

- Productos
- Clientes
- Pedidos
- Reseñas de productos
- Categorías

Posibles entidades con sus atributos:

- **Productos:** Nombre, descripción, precio, Stock(piezas existentes), categoria.
- **Clientes:** Nombre, correo electrónico, dirrección y No. Telefonico.
- **Pedidos:** Fecha??, estado(pediente, enviado, entregado), productos que forman el pedido.
- **Reseñas:** calificacion del producto comprado, cometario.
- **Categorias:** nombre, y descripcion de cada categoria.

Posibles foreing key de cada entidad

- **Productos:** Nombre, descripción, precio, Stock(piezas existentes), categoría. ***Foreing key de la entidad productos:*** categoria, permite una relacion con la entidad categorias.
- **Clientes:** Nombre, correo electrónico, dirrección y No. Telefonico. No tiene FK, candidata. Pero debemos buscar una forma de relacionarlos. Pues el cliente hace pedidos y reseñas, de ahi que las Fk candidatas puedan ser: ***pedidos y reseñas.***
- **Pedidos:** Fecha??, estado(pediente, enviado, entregado), productos que forman el pedido. ***FK candidata de la entidad pedido:*** FK_Productos. Permite relacionar pedidos con productos.
- **Reseñas de productos:** calificación del producto comprado, comentario. Sin fk candidata, pero debemos proponer una forma de rlacion la tabla reseñas con pruductos y clientes.

- **Categorías**: nombre, y descripción de cada categoria. Se debe buscar la forma de relacionar categorías como productos mediante un FK.

3.-Diagrama Entidad-Relación (ER) y justificación de la normalización

Después del análisis de los requerimientos y la determinación de entidades y atributos, el primer modelo de entidad relación es el siguiente:

Modelo ER: Enlace1

Esquema lógico: Enlace2

4.- Normalización y Justificación

- **Primera Forma Normal (1FN)**

  La Primera Forma Normal (1FN) establece que todos los atributos de una tabla deben contener valores atómicos, es decir, indivisibles. También implica que no deben existir grupos repetitivos ni columnas multivaluadas.

  Al revisar el modelo que tenemos actualmente, la entidad pedidos contiene al atributo productos, este atributo puede ser multivaluado al momento en que un cliente compre diversos productos en un mismo pedido, y eso rompe con la regla de la 1FN.

  Esto haría que productos no tenga un valor atómico.

  Solución:

  Para cumplir con la 1FN y representar correctamente la relación entre pedidos y productos, se creó la tabla intermedia: Detalles_Pedido con los siguientes campos: Esta nueva tabla permite al usuario tener un pedido con diversos productos, solucionala relación que las tablas productos y clientes tenian de muchos a muchos.

```
CREATE TABLE Detalles_Pedido (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    Id_Pedido INT,
    Id_Producto INT,
    Cantidad INT,
    Precio_Unitario DECIMAL,
    FOREIGN KEY (Id_Pedido) REFERENCES Pedidos(Id),
    FOREIGN KEY (Id_Producto) REFERENCES Productos(Id)
);
```

Podriamos aumentar la granularidad en la normalización, por ejemplo en la tabla clientes, descomponer el nombre, entre apellido paterno o materno. También podriamos crear una tabla especifica de contacto para tener correo electrónico en otra tabla, en caso de que el cliente tenga más de un correo, sin embargo la normalización con la tabla Detalles_Pedidos cumple con la funcionalidad, los requerimientos y las reglas de negocio del sistema solicitado.

- **Segunda Forma Normal (2FN)**

  La Segunda Forma Normal (2FN) establece que una tabla debe estar en Primera Forma Normal (1FN) Y no tener dependencias parciales, es decir, ningún atributo no clave debe depender solo de una parte de la clave primaria compuesta.

  Todas las tablas del modelo actual ya cumplen con la Segunda Forma Normal (2FN), dado que no presentan claves compuestas con dependencias parciales.

- **Tercera Forma Normal (3FN)**

  La Tercera Forma Normal (3FN) exige que una tabla esté en 2FN y que no existan dependencias transitivas entre atributos no clave.

  En nuestro modelo, todas las tablas cumplen con esta forma normal, ya que:

  - Cada atributo no clave depende directamente de su clave primaria.
  - No se repite información derivada de otros atributos no clave.

5.- Modelo y esquema final.

Despues de hacer la normalización volvi a generar el modelo ER y su esquema:

Modelo ER Normalizado: Enlace3

Esquema lógico Enlace4

6.-Scripts SQL

Creación de tablas y restricciones a considerar:

Antes de realizar los scrips de creación de la base de datos basados en el esquema anterior ya normalizado, es importante considerar las restricciones que tendrá la base de datos:

Restricciones:

- 1.- El stock no debe estar en negativo. Para solucionar esa restricción en la base de datos, nos situaremos en la tabla producto en el atributo stock, usando un check, para restringir que el atributo stock acepte valores negativos, la condicion que debe de cumplir es que el valor int insertado debe ser superior o igual a 0. La linea quedaría así:
  Stock INT CHECK (Stock >= 0),
- 2.- Unicidad en lo correos electrónicos de los clientes. Solucionaremos la unicidad de correos electrónicos ocupando la restricción unique, para garantizar que el atributo no se pueda insertar o actualizar valores duplicados, es decir que ya los este usando otro usuario.

La linea quedaria asi:

```
Correo_Electronico VARCHAR(150) UNIQUE,
```

- 3.-Un cliente no puede tener más de 5 pedidos pendientes.

Para agregar esa restricción podríamos hacer un store procedure que validara la cantidad almancenada de pedidos pendientes que tiene el usuario, pero al hacer la insercion de un nuevo pedido tendría que llamar al store procedure, para optimizar el apartado ocuparemos un trigger que se ejecute automátizamente después de que se inserte un nuevo pedido.

```
```sql

    DELIMITER //

    CREATE TRIGGER vpedidos_pendientes
    BEFORE INSERT ON Pedidos
    FOR EACH ROW
    BEGIN
        DECLARE totalPendientes INT;

        IF NEW.Estado = 'Pendiente'
        THEN
            SELECT COUNT(*) INTO totalPendientes
            FROM Pedidos
            WHERE Id_Cliente = NEW.Id_Cliente AND Estado = 'Pendiente';

            IF totalPendientes >= 5 THEN
                SIGNAL SQLSTATE '4500'
                SET MESSAGE_TEXT = 'El cliente ya tiene 5 pedidos pendientes.';
            END IF;
        END IF;
    END;
    //

    DELIMITER ;
```

Una vez completada la parte de las restricciones de la base de datos, procederemos a los scrips de creacion:

Scrips de creación:

Enlace 5

Poblar base de datos:

Para poblar la base de datos use la herramienta en linea: https://www.mockaroo.com/, que me permite seleccionar el tipo de dato,la cantidad de filas, y me devuelve un scrip de creación para usar en terminal e insertar. En esta parte tuve un problema al relacionar los valores por sus claves foraneas, por eso en los scrips tengo duplicada la población.

Scrips de poblacion generado: Enlace 6

Scrips usado con datos de Mockaroo: Enlace 7

7.- Generaremos las Consultas y los procedimientos almacenados:

Consultas
```

- **1.-Listar productos por categoría, ordenados por precio:**

```
Select c.nombre, p.nombre, p.precio
From Productos p
Inner join Categorias c
on p.Categoria_id = c.id
Order by p.precio;


SELECT c.nombre AS Categoria, p.nombre AS Producto, p.precio
FROM Productos p
INNER JOIN Categorias c ON p.Categoria_id = c.id
GROUP BY p.nombre, p.precio, c.nombre
ORDER BY p.precio;
```

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

```
    SELECT
    c.Nombre AS "Clientes con pedidos pendientes",
    SUM(d.Cantidad * d.Precio_Unitario) AS TotaldeCompra
FROM
    Clientes c
INNER JOIN
    Pedidos p ON c.Id = p.Id_Cliente
INNER JOIN
    Detalles_Pedido d ON p.Id = d.Id_Pedido
WHERE
    p.Estado = 'Pendiente'
GROUP BY
    c.Id, c.Nombre
ORDER BY
    TotaldeCompra DESC;
```

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

```
    SELECT
        p.Nombre AS Producto
    FROM
        Productos p
    INNER JOIN
        Resenas r ON p.Id = r.Id_Producto
    GROUP BY
        p.Id, p.Nombre
    ORDER BY
        AVG(r.Calificacion_Producto) DESC
    LIMIT 5;
```

Procedimiento almacenados:

*1.-Registrar un nuevo pedido, verificando el limite de 5 pedidos pendientes y stock sufuciente**

Este procedimiento fue solucionado en la parte de las restricciones con un trigger que verifica el limite de pedidos después de hacer un insert en la tabla pedidos, pero crearemos el store procedure.

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

```
DELIMITER $

CREATE PROCEDURE RegistrarResenia(
    IN p_Id_Cliente INT,
    IN p_Id_Producto INT,
    IN p_Calificacion INT,
```

```
    IN p_Comentario VARCHAR(255)
)
BEGIN
    IF (
        SELECT COUNT(*)
        FROM Pedidos p
        INNER JOIN Detalles_Pedido dp
            ON p.Id = dp.Id_Pedido
        WHERE p.Id_Cliente = p_Id_Cliente
        AND dp.Id_Producto = p_Id_Producto
    ) = 0
    THEN
        SELECT 'El cliente no ha comprado éste producto y no puede realizar un reseña';
    ELSE
        INSERT INTO Resenas (Calificacion_Producto, Comentario, Fecha, Id_Producto, Id_Cliente)
        VALUES (p_Calificacion, p_Comentario, CURRENT_DATE(), p_Id_Producto, p_Id_Cliente);
    END IF;
END $

DELIMITER ;
```

- **3.-Actualizar el stock de un producto despues de un pedido.**

```
 DELIMITER $

CREATE TRIGGER ActualizarStock
AFTER INSERT ON Detalles_Pedido
FOR EACH ROW
BEGIN
    UPDATE Productos
    SET Stock = Stock - NEW.Cantidad
    WHERE Id = NEW.Id_Producto;
END $

DELIMITER ;
```

- **4.-Cambiar el estado de un pedido. De pediente a enviado.**

```
 DELIMITER $

CREATE PROCEDURE CambiarEstado (
    IN p_Id_Pedido INT
)
BEGIN
    UPDATE Pedidos
    SET Estado = 'Enviado'
    WHERE Id = p_Id_Pedido AND Estado = 'Pendiente';
END $

DELIMITER ;
```

- **5.-Eliminar reseñas de un producto especifico, actualizando el promedio de calificaciónes.**

Para este procedimiento almancenado tenemos dos opciones, generar un nuevo atributo en la tabla resenas para que se pueda actualizar el promedio o simplemente calcular el promedio después de eliminar las reseñas, pero si eliminamos las reseñas de un producto especifico el resultado seria 0. Haré simplemente la eliminacion de reseñas de un producto especifico.

```
DELIMITER $

CREATE PROCEDURE EliminarResenias(
    IN p_Id_Producto INT
)
BEGIN
    DELETE FROM Resenas
    WHERE Id_Producto = p_Id_Producto;
END $
```

```
DELIMITER ;
```

- **6.- Agregar un nuevo producto, verificando que no exista un duplicado (mismo nombre y categoria)**

Para este procedimiento podriamos usar una retricción UNIQUE al crear los scrips de la base de datos, como lo hicimos con el correo electrónico, pero desarrollaremos el store procedure;

```
DELIMITER $

CREATE PROCEDURE AgregarSiNoExiste (
    IN p_Nombre VARCHAR(100),
    IN p_Descripcion VARCHAR(255),
    IN p_Precio FLOAT,
    IN p_Stock INT,
    IN p_Categoria_Id INT
)
BEGIN
    IF (
        SELECT COUNT(*)
        FROM Productos
        WHERE Nombre = p_Nombre
        AND Categoria_Id = p_Categoria_Id
    ) > 0 THEN
        SELECT 'El producto ya existe en esta categoría';
    ELSE
        INSERT INTO Productos (Nombre, Descripcion, Precio, Stock, Categoria_Id)
        VALUES (p_Nombre, p_Descripcion, p_Precio, p_Stock, p_Categoria_Id);
    END IF;
END $

DELIMITER ;
```

- **7.- actualizar la infomación de un cliente(por ejemplo, dirección o telefono)**

```
DELIMITER $

CREATE PROCEDURE ActualizarIfoCliente (
    IN p_IdCliente INT,
    IN p_NuevaDireccion VARCHAR(100),
    IN p_NuevoTelefono VARCHAR(25)
)
BEGIN
    UPDATE Clientes
    SET
        Direccion=p_NuevaDireccion,
        Telefono=p_NuevoTelefono
    WHERE Id=p_IdCliente;
END $

DELIMITER ;
```

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

```
SELECT
    p.Id,
    p.Nombre AS Producto,
    p.Stock,
    c.Nombre AS Categoria
FROM  Productos p
INNER JOIN Categorias c ON p.Categoria_Id = c.Id
WHERE
    p.Stock < 5
ORDER BY
    p.Stock ASC;
```

8.- Validación y Optimización.

Resultados de consultas ejecutadas con datos de prueba.

En esta parte ocupe la herramienta boost-tool, que convierte el texto plano que genera la consulta en la terminal a una tabla que se pueda igresar en MarckDown.

- **1.- Listar productos por categoria, ordenados por precio:**

Categoria	Producto	precio
Alimentos - Lácteos	Zanahorias baby	28.9
Salud	Semillas de calabaza	35
Exterior	Yogur probiótico	39.5
Juguetes	Audífonos inalámbricos deportivos	41.93
Alimentos - Condimentos	Té fermentado	42
Alimentos - Aderezos	Aderezo de miso	47
Salud	Mezcla para panqueques	48.9
Salud	Mezcla para panque	48.9
Juguetes	Hummus de ajo	55.2
Alimentos - Proteína	Pasta italiana	56.8
Electrónica	Botana de queso	62.5
Mascotas	Pretzels con chocolate	68
Alimentos - Snacks	Arroz frito congelado	73.9
Exterior	Rallador multifuncional	79.9
Alimentos - Lácteos	Ravioles de espinaca	84.75
Alimentos - Panadería	Ravioles de calabaza	88
Fotografía	Pelota interactiva para perro	129.38
Automotriz	Hamburguesas de res	132
Fitness	Manta suave	189
Alimentos - Aderezos	Galletas de avena con chispas	218.56
Automotriz	Loción corporal de coco	219.69
Alimentos - Comidas Preparadas	Salsa BBQ ahumada	241.52
Ropa - Ropa Exterior	Cuaderno con diseño artístico	255.7
Mascotas	Crema hidratante facial	273.98
Cocina	Kit de construcción creativa	299
Alimentos - Comidas Congeladas	Jumpsuit cruzado	349
Alimentos - Comida Congelada	Leche vegetal de almendra	376.54
Cocina	Tenis transpirables	399
Alimentos - Granos	Aceite de oliva extra virgen	419.63
Alimentos - Repostería	Alfombra decorativa redonda	423.13
Belleza	Mermelada de frutos rojos	426.78
Alimentos - Comida Congelada	Collar con dije minimalista	441.69
Automotriz	Juego de cuchillos de chef	457.33
Ropa - Accesorios	Pan artesanal integral	490.65
Alimentos - Condimentos	Pelota de yoga antideslizante	530.65
Electrónica	Barra energética de avena	602.13
Exterior	Cereal de granola natural	684.12

Categoria	Producto	precio
Mascotas	Papel fotográfico brillante	878
Alimentos - Desayuno	Bolso cruzado de piel sintética	904.95
Alimentos - Postres Congelados	Batería portátil solar	969.11
Alimentos - Repostería	Set de pesas ajustables	999

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

Clientes con pedidos pendientes	TotaldeCompra
Silvana Scouller	12800.8203125
Barbaraanne Dowderswell	9386.499755859375
Babara Starbuck	9098.2001953125
Abby O'Henehan	7568.0400390625
Jermayne Poulgreen	7407.35986328125
Arron Loader	6652.33984375
Saunderson Nimmo	4961.210105895996
Daron Shildrick	4102.320022583008
Bertina Mc Carrick	4072.2099609375
Jasmina Brien	3367.3798828125
Verney Baumann	1083
Caddric Sebyer	813.9000015258789
Abigail Hedylstone	613
Emelina Kovelmann	467
Mandel Skitt	286.4000015258789
Lotty Wye	231
Erhard Knellen	108.9000015258789
Thibaud Coopper	97.29999923706055

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

Producto
Rallador multifuncional
Pretzels con chocolate
Hamburguesas de res
Pasta italiana
Tenis transpirables

9.-Explicación de indices creados y su impacto (Usando Explain Mysql)

- **1.-Busqueda por Categoria**

  CREATE INDEX idx_categoria_id ON Productos(Categoria_Id);

  ```
  EXPLAIN
  SELECT Nombre, Precio
  FROM Productos
  WHERE Categoria_Id = 4
  ```

  Agregué el índice tabla de productos en el atributo categoria_id, esto permite reducir el tiempo de las consultas que filtran productos según su categoría.

Si este indice no estuviera creado, el SMDB tendría que hacer un recorrido sobre toda la tabla de Productos. Con este indice se accede directamente a categoría en futuras consultas. Usando Explain para verficar que se este usando el indice la terminal no regresa:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	Productos	NULL	ref	idx_categoria_id	idx_categoria_id	5	const	1	100.00	NULL

- **2.-Consultar el estado de los pedidos.** Este indice permite reducir el tiempo de consulta al saber el estado de los pedidos en su situación actual: Pediente, Enviado, Entregado o Cancelado, es decir mejorar el rendiminento en la consulta. Sin el indice, cada vez que se le haga una consulta para saber el estado de los pedidos, el SGBD tendrá que realizar un barrido por toda la tabla.

```
CREATE INDEX idx_estado_pedido ON Pedidos(Estado);


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_pedido	idx_estado_pedido	203	const	27	100.00	NULL

3.-Indice para buscar productos por el nombre de usuario.

```
CREATE INDEX idx_nombre_producto ON Productos(Nombre);
```

La creación de este indice sobre nombre del atributo productos ayuda a mejorar el tiempo de respuesta en la consultas que requieran buscar por nombre del producto.

```
EXPLAIN SELECT * FROM Productos WHERE Nombre = 'Aderezo de miso';
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	Productos	NULL	ref	idx_nombre_producto	idx_nombre_producto	403	const	4	100.00	NULL

10.-Pruebas de procedimientos almacenados

1.-Registrar un nuevo pedido, verificando el limite de 5 pedidos pendientees y stock sufuciente

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

```
DELIMITER $

CREATE PROCEDURE RegistrarResenia(
    IN p_Id_Cliente INT,
    IN p_Id_Producto INT,
    IN p_Calificacion INT,
    IN p_Comentario VARCHAR(255)
)
BEGIN
    IF (
        SELECT COUNT(*)
        FROM Pedidos p
        INNER JOIN Detalles_Pedido dp
            ON p.Id = dp.Id_Pedido
        WHERE p.Id_Cliente = p_Id_Cliente
        AND dp.Id_Producto = p_Id_Producto
    ) = 0
    THEN
        SELECT 'El cliente no ha comprado éste producto y no puede realizar un reseña';
    ELSE
        INSERT INTO Resenas (Calificacion_Producto, Comentario, Fecha, Id_Producto, Id_Cliente)
        VALUES (p_Calificacion, p_Comentario, CURRENT_DATE(), p_Id_Producto, p_Id_Cliente);
    END IF;
END $

DELIMITER ;
```

Para probarlo desde la terminar primero haremos un inner join que nos permita saber qué productos ha comprado el cliente con el id = 1.

Id_Producto	Producto	Precio	Categoria
10	Rallador multifuncional	79.9	Exterior
15	Hamburguesas de res	132	Automotriz
16	Jumpsuit cruzado	349	Alimentos - Comidas Congeladas
21	Mezcla para panqueques	48.9	Salud
3	Ravioles de espinaca	84.75	Alimentos - Lácteos

Después mediante un llama dal store procedure trataremos de agregar una reseña a un producto comprado y a otro que no, es decir ocupareoms dirente id:

- **1.- CALL RegistrarResenia(1, 10, 5, 'Excelente calidad');** Query OK, 1 row affected (0.09 sec)
- **2.-CALL RegistrarResenia(1, 200, 5, 'Excelente calidad');**

  El cliente no ha comprado éste producto y no puede realizar un reseña
- **3.-Actualizar el stock de un producto despues de un pedido.**

```
 DELIMITER $

CREATE TRIGGER ActualizarStock
AFTER INSERT ON Detalles_Pedido
FOR EACH ROW
BEGIN
    UPDATE Productos
    SET Stock = Stock - NEW.Cantidad
    WHERE Id = NEW.Id_Producto;
END $

DELIMITER ;
```

Para poder verificar si funciona primero hacemos un select a la tabla productos:

Id	Nombre	Descripcion	Precio	Stock	Categoria_Id
1	Mezcla para panqueques	Mezcla lista para preparar panqueques con chispas de chocolate.	48.9	17	3
2	Set de pesas ajustables	Mancuernas ajustables que ahorran espacio para entrenamiento de fuerza.	999	25	14
3	Ravioles de espinaca	Ravioles rellenos de espinaca fresca y ricotta cremosa.	84.75	26	68
4	Hummus de ajo	Dip cremoso de garbanzo con ajo intenso.	55.2	24	81
5	Manta suave	Cobija ligera y cálida ideal para noches frías.	189	11	64
6	Kit de construcción creativa	Set de bloques para fomentar la imaginación infantil.	299	10	69
7	Botana de queso	Crujientes bocadillos de queso que se deshacen en la boca.	62.5	12	56
8	Ravioles de calabaza	Ravioles rellenos de calabaza asada y especias.	88	23	26
9	Semillas de calabaza	Botana crujiente de semillas de calabaza tostadas.	35	27	3
10	Rallador multifuncional	Rallador de acero para quesos y verduras.	79.9	7	77

Haremos un insert en detalles pedidos para actualizar el stock del producto mezcla para panqueques.

```
INSERT INTO Detalles_Pedido (Id_Pedido, Id_Producto, Cantidad, Precio_Unitario)
-> VALUES (51, 1, 7, 48.9);
```

y volvemos a hacer el select para revisar el stock del producto con el id 1

Id	Nombre	Descripcion	Precio	Stock	Categoria_Id
1	Mezcla para panqueques	Mezcla lista para preparar panqueques con chispas de chocolate.	48.9	10	3
2	Set de pesas ajustables	Mancuernas ajustables que ahorran espacio para entrenamiento de fuerza.	999	25	14

Id	Nombre	Descripcion	Precio	Stock	Categoria_Id
3	Ravioles de espinaca	Ravioles rellenos de espinaca fresca y ricotta cremosa.	84.75	26	68
4	Hummus de ajo	Dip cremoso de garbanzo con ajo intenso.	55.2	24	81

Listo, se actualizo el stock.

- **4.-Cambiar el estado de un pedido. De pediente a enviado.**

```
 DELIMITER $

CREATE PROCEDURE CambiarEstado (
    IN p_Id_Pedido INT
)
BEGIN
    UPDATE Pedidos
    SET Estado = 'Enviado'
    WHERE Id = p_Id_Pedido AND Estado = 'Pendiente';
END $

DELIMITER ;
```

Para hacer la prueba del store procedure, primero haremos un select a la tabla pedidos:

Id	Fecha_pedido	Estado	Id_Cliente
51	2025-07-02	Entregado	2
52	2025-07-03	Entregado	3
53	2025-07-04	Pendiente	4

Trataremos ahora de actualizar el pedido con el id 51 llamando al store procedure:

```
CALL CambiarEstado(53);
```

Id	Fecha_pedido	Estado	Id_Cliente
51	2025-07-02	Entregado	2
52	2025-07-03	Entregado	3
53	2025-07-04	Enviado	4

- **5.-Eliminar reseñas de un producto especifico, actualizando el promedio de calificaciónes.**

Para este procedimiento almancenado tenemos 2 opciones, generar un nuevo atributo en la tabla resenas para que se pueda actualizar el promedio o simplemente calcular el promedio después de eliminar las reseñas, pero si eliminamos las reseñas de un producto especifico el resultado seria 0. Haré simplemente la eliminacion de reseñas de un producto especifico.

```
DELIMITER $

CREATE PROCEDURE EliminarResenias(
    IN p_Id_Producto INT
)
BEGIN
    DELETE FROM Resenas
    WHERE Id_Producto = p_Id_Producto;
END $

DELIMITER ;
```

- **6.- Agregar un nuevo producto, verificando que no exista un duplicado (mismo nombre y categoria)**

Para este procedimiento podriamos usar una retricción UNIQUE al crear los scrips de la base de datos, como lo hicimos con el correo electronico, pero desarrollaremos el store procedure;

```
DELIMITER $

CREATE PROCEDURE AgregarSiNoExiste (
    IN p_Nombre VARCHAR(100),
    IN p_Descripcion VARCHAR(255),
    IN p_Precio FLOAT,
    IN p_Stock INT,
    IN p_Categoria_Id INT
)
BEGIN
    IF (
        SELECT COUNT(*)
        FROM Productos
        WHERE Nombre = p_Nombre
        AND Categoria_Id = p_Categoria_Id
    ) > 0 THEN
        SELECT 'El producto ya existe en esta categoría';
    ELSE
        INSERT INTO Productos (Nombre, Descripcion, Precio, Stock, Categoria_Id)
        VALUES (p_Nombre, p_Descripcion, p_Precio, p_Stock, p_Categoria_Id);
    END IF;
END $

DELIMITER ;
```

Para probar este procedimiento insertaré realizar un nuevo producto, después volveré a insertar el mismo producto, por último harémos un select a la tabla productos para confimar que se agrego correctamente.

```
CALL AgregarProductoSiNoExiste(
'Laptop HP',
'Laptop de 15 pulgadas con 8GB RAM',
1500,
10,
1
);
```

Id	Nombre	Descripcion	Precio	Stock	Categoria_Id
99	Mermelada de frutos rojos	Fórmula especializada para cuidado diario.	426.78	20	28
100	Bolso cruzado de piel sintética	Aporta beneficios prácticos y decorativos.	904.95	10	49
101	Laptop HP	Laptop de 15 pulgadas con 8GB RAM	1500	10	1

- **7.- Actualizar la infomación de un cliente(por ejemplo, dirección o telefono)**

  ```
  DELIMITER $

  CREATE PROCEDURE ActualizarIfoCliente (
      IN p_IdCliente INT,
      IN p_NuevaDireccion VARCHAR(100),
      IN p_NuevoTelefono VARCHAR(25)
  )
  BEGIN
      UPDATE Clientes
      SET
          Direccion=p_NuevaDireccion,
          Telefono=p_NuevoTelefono
      WHERE Id=p_IdCliente;
  END $

  DELIMITER ;
  ```

  Actualizaremosla infomación de un cliente, select para elegir un cliente.

Id	Nombre	Correo_Electronico	Direccion	Telefono

Id	Nombre	Correo_Electronico	Direccion	Telefono
1	Silvana Scouller	sscouller0@e-recht24.de	15th Floor	399-379-4012
2	Bertina Mc Carrick	bmc1@people.com.cn	Apt 28	852-972-5846
3	Zechariah De Ruggero	zde2@uol.com.br	10th Floor	763-973-0836

actualizamos el id 3:

```
CALL ActualizarClienteSimple(3, 'Av. Reforma 123, CDMX', '555-123-4567');
```

Id	Nombre	Correo_Electronico	Direccion	Telefono
1	Silvana Scouller	sscouller0@e-recht24.de	15th Floor	399-379-4012
2	Bertina Mc Carrick	bmc1@people.com.cn	Apt 28	852-972-5846
3	Zechariah De Ruggero	zde2@uol.com.br	Av. Reforma 123, CDMX	555-123-4567

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

```
SELECT
    p.Id,
    p.Nombre AS Producto,
    p.Stock,
    c.Nombre AS Categoria
FROM  Productos p
INNER JOIN Categorias c ON p.Categoria_Id = c.Id
WHERE
    p.Stock < 5
ORDER BY
    p.Stock ASC;
```

11.-Propuestas de mejoras:

Podemos generar indices adicionales que mejoren la base de datos, podriamos pensar por ejemplo en la información de contacto, generar un indice sobre el teléfono o el correo del cliente para poder contactarlo.

Podríamos mejorar el atributo stock, agregarlo a un trigger para que se actualice automaticamente después de una entrega o venta.

Se podria mejorar la unicidad de la infomación de los clientes o de los productos agregando la restricción UNIQUE.

Validar que Calificacion_Producto en Resenas solo acepte valores del 1 al 5, al igual para Estado en Pedidos, que solo acepte 'Pendiente', 'Enviado' o 'Entregado':

Agregar un store procedure que se ejecute todos los dias y notifique el stock bajo.

12.-Conclusiones y lecciones aprendidas.

Puedo concluir que fue un proyecto que me sentí satisfecho de realizarlo, además me permitió comprender el proceso que conlleva diseñar, implementar y probar una base de datos relacional. Desde la creación de tablas y relaciones hasta la inserción de datos y la ejecución de consultas.

Una de las principales lecciones fue reconocer que la práctica constante no solo mejora la comprensión, sino que también acelera la capacidad para resolver errores. Sin embargo, también aprendí que antes de escribir cualquier línea de código es crucial tener claro el problema que se desea resolver y estructurar una lógica precisa desde el inicio para resolverlo.

Durante el desarrollo, enfrenté desafíos como poblar la base de datos, crear stored procedures funcionales y sobre todo mejorar la habilidad logica de poder resolver un proceso. Esto me mostró que el aprendizaje es reiterativo y que, aunque por momeentos me frustré, debo tratar de reenfocar la concentración para poder avanzar.

Estoy en proceso de encontrar una metodología de trabajo personal que se adapte a mis necesidades y ritmo. Este proyecto me ayudó a identificar que una de las áreas donde más debo trabajar es en mejorar mi concentración.

En conclusión, fue un proyecto que me retó en lo técnico y en lo personal. Me permitió conocer y adquirir habilidades para el desarrollo de bases de datos.

De antemano, Gracias :V