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

Profesor:

Dr. Guillermo Monroy

Alumno:

Jose Abraham Marin Sánchez

INTRODUCCION AL PROBLEMA

Se requiere diseñar e implementar una base de datos relacional para una tienda online de aparatos electrónicos que quiere gestionar las operaciones de la tienda por lo que los requisitos para la base de datos que pide el cliente son:

1.- Que se pueda hacer gestión de los productos de la tienda, específicamente el registro de productos los cuales deben contar con nombre, descripción, precio, stock y categoría.

2.- La gestión de los clientes de la tienda con nombre, dirección, correo electrónico, teléfono y dirección los cuales ayudaran a identificarlo para pedidos y reseñas.

3.- La capacidad de crear y gestionar pedidos los cuales tengan características como la fecha, el estado del pedido y los productos que incluye.

4.- La gestión de reseñas de clientes a productos que han comprado donde puedan dar calificación de entre 1 y 5 estrellas además de que se pueda dejar su comentario.

5.- Categorías donde se puedan organizar los productos según sus características como computadoras, videojuegos y accesorios.

Además, existen algunas restricciones como no permitir stock negativo en los productos, que el correo electrónico que ya tenga un cliente no se permita registrar otra vez, que cada cliente no pueda tener mas de 5 pedidos pendientes y que las reseñas solo las puedan hacer aquellos clientes que hayan comprado el producto.

OBJETIVOS

Los objetivos de este proyecto son cumplir con los requerimientos del cliente usando los conocimientos obtenidos en clase, haciendo en primer lugar un análisis y diseño de la base de datos relacional incluyendo lograr una normalización a la tercera forma normal e identificando sus claves primarias, foráneas y candidatas las cuales serán importantes al crear los scripts.

Se escribirán varios scripts en SQL en el que vendrán la creación de las tablas requeridas, script con los índices para optimizar consultas frecuentes, script para poblar nuestra base de datos. También se crearán consultas y procedimientos

almacenados en scripts a los cuales posteriormente se les harán pruebas en diferentes escenarios.

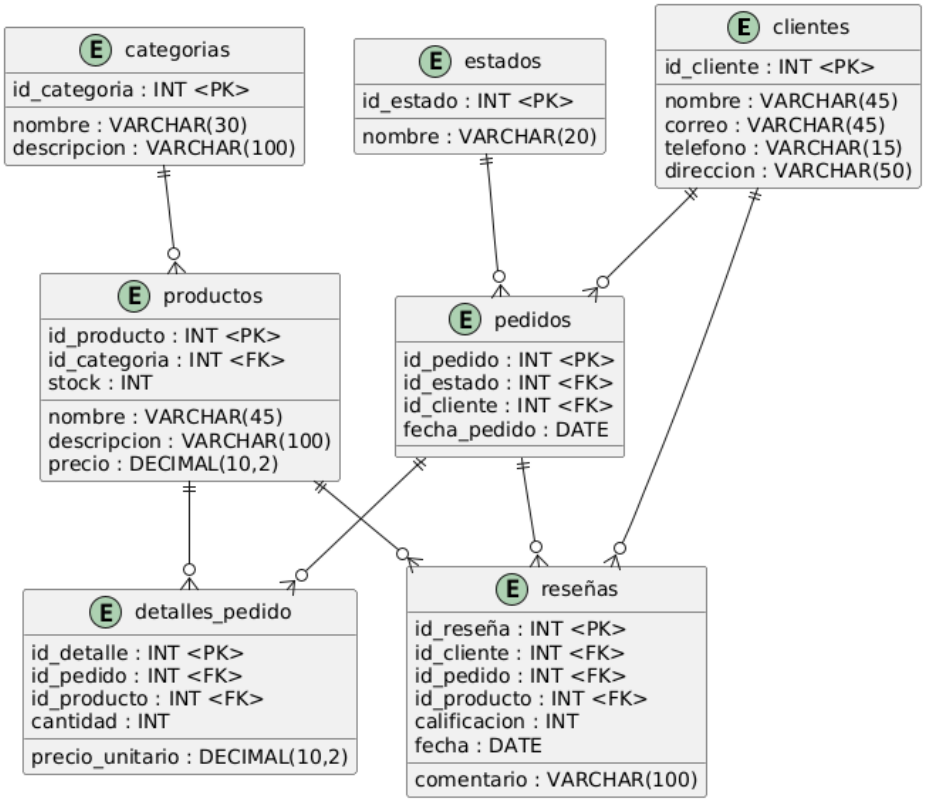
Finalmente, todo este proceso estará documentado aquí incluyendo explicación de los diferentes apartados, informe de las pruebas y algunas propuestas de mejoras para la base de datos.

ANALISIS Y DISEÑO

DIAGRAMA ENTIDAD RELACION:

Mi diagrama cuenta con mis 7 tablas: categorías, estados, clientes, productos, pedidos, detalles_pedido y reseñas. Y las relaciones entre las tablas son las siguientes:

- Una categoría puede pertenecer a varios productos.
- Un estado se le puede dar a varios pedidos.
- Un cliente puede tener varios pedidos.
- Un cliente puede dejar varias reseñas.
- Un producto puede tener varias reseñas.
- Un pedido puede tener varios detalles (de varios productos)



ESQUEMA DE FORMAS NORMALES:

Estas son las tres primeras formas normales:

1NF: La cumple si no hay atributos multivaluados, compuestos y todos son atómicos.

2NF: La cumple si está en 1NF y la clave primaria es simple y todos los atributos no clave dependen de ella.

3NF: La cumple si está en 2NF y porque no hay dependencias transitivas entre atributos no clave y la clave primaria

Se agregaron dos tablas más al diseño para cumplir con la normalización: **estado y detalles_pedido**

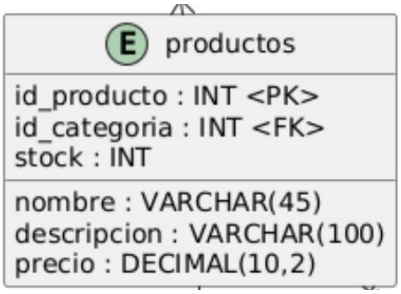
Tabla clientes.

1NF: La cumple ya que no hay atributos multivaluados, compuestos y todos son atómicos (correo y teléfono son únicos).

2NF: La cumple porque esta en 1NF y la clave primaria id_cliente es simple y todos los atributos no clave dependen de ella.

3NF: La cumple porque está en 2NF y porque no hay dependencias transitivas entre atributos no clave y la clave primaria

Tabla productos

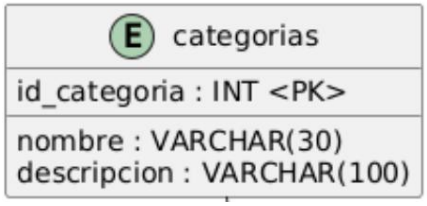


1NF: La cumple ya que nombre, descripción, precio y stock son atómicos.

2NF: La cumple porque está en 1NF y la clave primaria id_producto es simple y todos los atributos no clave dependen de ella.

3NF: La cumple porque está en 2NF y no hay atributos que dependan de otros atributos no clave.

Tabla categorías

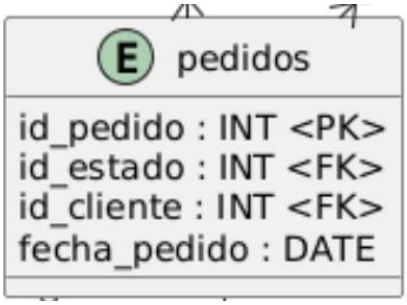


1NF: La cumple ya que no hay atributos multivaluados, compuestos y todos son atómicos

2NF: La cumple porque está en 1NF y los atributos nombre y descripción dependen de id_categoria.

3NF: La cumple porque está en 2NF y no hay atributos que dependan de otros atributos no clave.

Tabla pedidos

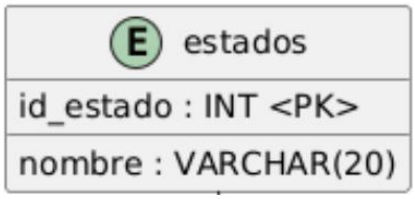


1NF: La cumple ya que todos los datos son atómicos y no hay atributos multivaluados o compuestos
*Cree la tabla estados para cumplir la normalización.

2NF: La cumple porque está en 1NF y la clave primaria id_pedido es simple y el atributo fecha_pedido depende de ella. Los otros atributos son FK.

3NF: La cumple porque está en 2NF y no hay atributos que dependan de otros atributos no clave.

Tabla estados

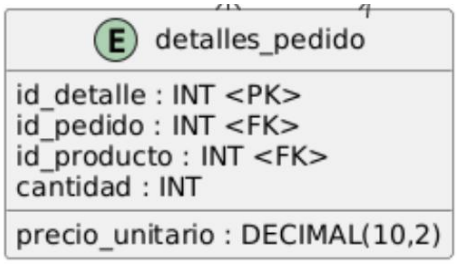


1NF: La cumple ya que todos los datos son atómicos y no hay atributos multivaluados o compuestos

2NF: La cumple porque está en 1NF y el atributo nombre depende solo de la PK id_estado

3NF: La cumple porque está en 2NF y no hay atributos no clave que dependan entre si

Tabla detalles_pedido



1NF: La cumple ya que todos los datos son atómicos y no hay atributos multivaluados o compuestos.

2NF: La cumple porque está en 1NF y todos los atributos no clave (cantidad, precio_unitario y las FK) dependen de la clave primaria id_detalle

3NF: La cumple porque está en 2NF y no hay dependencias transitivas.

Tabla reseñas

1NF: La cumple ya que todos los datos son atómicos y no hay atributos multivaluados o compuestos.

2NF: La cumple porque está en 1NF y todos los atributos no clave dependen de la clave primaria id_reseña

3NF: La cumple porque está en 2NF y no hay atributos no clave que dependan entre sí.

IDENTIFICACION DE CLAVES PRIMARIAS, FORANEAS Y CANDIDATAS

- Claves primarias:
  En cada una de las tablas tengo una clave primaria que es su id (**id_categoria, id_estado, id_cliente, id_producto, id_pedido, id_detalle y id_reseña**) que sirven para identificar de manera única ya sea un producto, pedido, cliente, etc.

- Claves foráneas:

  productos:
  La tabla de productos necesita de **id_categoria** para saber a qué categoría pertenece cierto producto

  pedidos:
  Necesita tener las claves foráneas de **id_estado** para saber si ya fue entregado, esta en proceso o fue cancelado, y necesita de **id_cliente** para identificar de que cliente es el pedido.

  detalles_pedido:
  Al ser de cierta manera una extensión del pedido requiere saber el **id_pedido** para conocer el estado y fecha, y además necesita información del producto por lo que usamos **id_producto**.

  reseñas:
  La reseña necesita saber de quien es (**id_cliente**), de que producto es la reseña (**id_producto**) y saber información de ese pedido (**id_pedido**).

- Claves candidatas:

  Las que considero podrían ser claves candidatas son solamente **correo** de la tabla clientes que al ser único podría usarse como identificador, además tal vez el **nombre** en categoría y estado ya que son únicos.
  Podría ser también **nombre** en producto, pero podrían existir dos nombres iguales.

ANALISIS Y VALIDACION.

-Resultados de la ejecución de las consultas.

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

```
mysql> SELECT
    -> c.nombre AS categoria,
    -> p.nombre AS producto,
    -> p.precio,
    -> p.stock
    -> FROM productos p
    -> INNER JOIN categorias c ON p.id_categoria = c.id_categoria
    -> WHERE p.stock > 0
    -> ORDER BY c.nombre ASC, p.precio ASC;
+-------------+----------------------+----------+-------+
| categoria   | producto             | precio   | stock |
+-------------+----------------------+----------+-------+
| Accesorios  | Sony SRS-XB13        |   749.00 |     8 |
| Accesorios  | Logitech MK270       |   799.00 |     9 |
| Accesorios  | HP OMEN 600          |   849.00 |     6 |
| Accesorios  | Redragon K552        |   999.00 |     5 |
| Accesorios  | HyperX Cloud Stinger |  1149.00 |     4 |
| Accesorios  | Razer DeathAdder V2  |  1199.00 |     7 |
| Accesorios  | Logitech G435        |  1499.00 |    10 |
| Laptops     | Asus VivoBook 14     | 10299.50 |     5 |
| Laptops     | Lenovo IdeaPad 3     | 11499.00 |     8 |
| Laptops     | HP Pavilion 15       | 12999.99 |     4 |
| Laptops     | Acer Aspire 5        | 13750.00 |     6 |
| Laptops     | MSI Modern 15        | 14200.00 |     5 |
| Laptops     | Lenovo Legion Slim 5 | 15499.00 |     7 |
| Laptops     | MacBook Air M2       | 18999.99 |     4 |
| Pantallas   | AOC 24B1XHS          |  2649.00 |     5 |
| Pantallas   | Xiaomi 24i           |  2899.00 |     4 |
| Pantallas   | Philips TV           |  2999.00 |     2 |
| Pantallas   | BenQ GW2480          |  2999.00 |     7 |
| Pantallas   | LG Monitor 27MK400H  |  3299.99 |     6 |
| Pantallas   | Samsung Smart TV 50  |  8499.00 |     4 |
| Pantallas   | Hisense A6H          |  8699.00 |     3 |
| Smartphones | Poco 11T             |  4399.99 |    11 |
| Smartphones | Honor X8             |  5999.00 |     8 |
| Smartphones | Xiaomi Redmi Note 13 |  6599.99 |     8 |
| Smartphones | Motorola G84         |  7199.00 |     3 |
| Smartphones | Huawei Nova 11       |  7999.00 |     6 |
| Smartphones | Samsung Galaxy A54   |  8499.50 |     5 |
| Smartphones | iPhone 13            | 17499.00 |    10 |
| Videojuegos | Elden Ring           |   899.00 |     8 |
| Videojuegos | Mario Kart 8 Deluxe  |  1199.00 |     4 |
| Videojuegos | EA FC25              |  1399.00 |     7 |
| Videojuegos | DualSense Controller |  1599.00 |    10 |
| Videojuegos | Xbox Series S        |  7499.00 |     6 |
| Videojuegos | Nintendo Switch OLED |  8899.00 |     5 |
| Videojuegos | PlayStation 5        | 13499.00 |     3 |
+-------------+----------------------+----------+-------+
35 rows in set (0.00 sec)
```

Resultado: Muestra de manera correcta los productos primero ordenados por las categorías y entre las categorías por precio ascendente.

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

```
mysql> SELECT
    -> c.id_cliente,
    -> c.nombre,
    -> COUNT(p.id_pedido) AS pedidos_pendientes,
    -> SUM(dp.cantidad * dp.precio_unitario) AS total_compras
    -> FROM clientes c
    -> INNER JOIN pedidos p ON c.id_cliente = p.id_cliente
    -> INNER JOIN detalles_pedido dp ON p.id_pedido = dp.id_pedido
    -> WHERE p.id_estado = 1
    -> GROUP BY c.id_cliente, c.nombre;
+------------+---------------+-------------------+---------------+
| id_cliente | nombre        | pedidos_pendientes | total_compras |
+------------+---------------+-------------------+---------------+
|          3 | Laura Sanchez |                 2 |      29998.99 |
|          2 | Raul Ortiz    |                 2 |      16597.00 |
|          4 | Pedro Perez   |                 1 |      13750.00 |
|         13 | Alma Guerrero |                 1 |       2547.00 |
|          1 | Luis Garcia   |                 1 |      15499.00 |
|          6 | Jaime Nuno    |                 1 |       1399.00 |
+------------+---------------+-------------------+---------------+
6 rows in set (0.02 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
    -> p.nombre,
    -> AVG(r.calificacion) AS calificacion_promedio
    -> FROM resenas r
    -> INNER JOIN productos p ON r.id_producto = p.id_producto
    -> GROUP BY p.id_producto, p.nombre
    -> ORDER BY calificacion_promedio DESC
    -> LIMIT 5;
+----------------+----------------------+
| nombre         | calificacion_promedio |
+----------------+----------------------+
| Logitech MK270 |               5.0000 |
| Xbox Series S  |               5.0000 |
| Sony SRS-XB13  |               4.0000 |
| HP OMEN 600    |               4.0000 |
| Hisense A6H    |               3.0000 |
+----------------+----------------------+
5 rows in set (0.02 sec)
```

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

EXPLICACION DE INDICES CREADOS Y SU IMPACTO.

Se crearon estos índices:

1. índice para buscar producto por nombre

CREATE INDEX idx_nombre_producto ON productos(nombre);

El objetivo de este índice es que al hacer una consulta como **SELECT * FROM productos WHERE nombre = 'iPhone 13';** la búsqueda sea más rápida ya que mi tabla productos puede tener muchos productos.

Primero hice el EXPLAIN de la consulta sin ingresar el índice:

```
mysql> EXPLAIN  SELECT  *  FROM
    -> productos WHERE nombre = 'iPhone 13';
+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+
| id | select_type | table     | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra       |
+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | productos | NULL       | ALL  | NULL          | NULL | NULL    | NULL |   35 |    10.00 | Using where |
+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+
```

Le tomo 35 filas poder encontrar el producto.

Ahora hice la misma consulta después de crear el índice:

```
mysql> EXPLAIN SELECT * FROM productos WHERE nombre = 'iPhone 13';
+----+-------------+-----------+------------+------+-----------------------+-----------------------+---------+-------+------+----------+-------+
| 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 | 183     | const |    1 |   100.00 | NULL  |
+----+-------------+-----------+------------+------+-----------------------+-----------------------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

Se puede ver que el índice se ocupo ya que se ve en la columna key el nombre del índice y la optimización se nota en la columna rows ya que solo reviso una fila para encontrar ese producto

2. Indice para obtener productos por categoría.

CREATE INDEX idx_categoria_producto ON productos(id_categoria);

Busca optimizar las consultas cuando buscamos productos por el id de la categoría, el índice optimizara consultas como **SELECT * FROM productos WHERE id_categoria = 2;**

```
mysql> EXPLAIN SELECT * FROM productos WHERE id_categoria = 2;
+----+-------------+-----------+------------+------+-------------------------+-------------------------+---------+-------+------+----------+-------+
| id | select_type | table     | partitions | type | possible_keys           | key                     | key_len | ref   | rows | filtered | Extra |
+----+-------------+-----------+------------+------+-------------------------+-------------------------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | productos | NULL       | ref  | idx_categoria_producto | idx_categoria_producto | 5       | const |    7 |   100.00 | NULL  |
+----+-------------+-----------+------------+------+-------------------------+-------------------------+---------+-------+------+----------+-------+
```

```
mysql> SELECT * FROM productos WHERE id_categoria = 2;
+-------------+----------------------+-----------------------------------------------------------+----------+-------+--------------+
| id_producto | nombre               | descripcion                                               | precio   | stock | id_categoria |
+-------------+----------------------+-----------------------------------------------------------+----------+-------+--------------+
|           8 | iPhone 13            | Smartphone de 128GB, camara dual, iOS 17                  | 17499.00 |    10 |            2 |
|           9 | Samsung Galaxy A54   | 6.4 pulgadas, pantalla AMOLED, 256GB                      |  8499.50 |     5 |            2 |
|          10 | Xiaomi Redmi Note 13 | Smartphone 5G, camara 108MP, 8GB RAM,                     |  6599.99 |     8 |            2 |
|          11 | Motorola G84         | Smatphone con pantalla OLED,128GB, 12GB RAM               |  7199.00 |     3 |            2 |
|          12 | Huawei Nova 11       | Smartphone con 128GB de almacenamiento, Android 13        |  7999.00 |     6 |            2 |
|          13 | Honor X8             | Smartphone huawei, 8gb RAM, Snapdragon 680                |  5999.00 |     8 |            2 |
|          14 | Poco 11T             | Pantalla HD+, bateria 5000mAh, 265GB almacenamiento, 6GB RAM |  4399.99 |    11 |            2 |
+-------------+----------------------+-----------------------------------------------------------+----------+-------+--------------+
7 rows in set (0.00 sec)
```

El índice también se uso como muestra la columna key y reviso solo 7 filas para obtener los 7 productos que tengo es esa categoría.

3. Indice para obtener pedidos por cliente.

CREATE INDEX idx_cliente_pedido ON pedidos(id_cliente);

El índice busca optimizar las consultas cuando queremos buscar los pedidos de x cliente, una consulta que sería más rápida seria **SELECT * FROM pedidos WHERE id_cliente = 7;**

```
mysql> EXPLAIN SELECT * FROM pedidos WHERE id_cliente = 7;
+----+-------------+---------+------------+------+--------------------+--------------------+---------+-------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys      | key                | key_len | ref   | rows | filtered | Extra |
+----+-------------+---------+------------+------+--------------------+--------------------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | pedidos | NULL       | ref  | idx_cliente_pedido | idx_cliente_pedido | 5       | const |    2 |   100.00 | NULL  |
+----+-------------+---------+------------+------+--------------------+--------------------+---------+-------+------+----------+-------+
```

También hace uso del índice y solo reviso dos filas ya que ese usuario solo tiene dos pedidos;

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 (0.02 sec)

mysql> SELECT * FROM pedidos WHERE id_cliente = 3;
+-----------+--------------+-----------+------------+
| id_pedido | fecha_pedido | id_estado | id_cliente |
+-----------+--------------+-----------+------------+
|         1 | 2025-07-01   |         1 |          3 |
|        21 | 2025-07-30   |         1 |          3 |
+-----------+--------------+-----------+------------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM detalles_pedido WHERE id_pedido = 21;
+------------+----------+-----------------+-----------+-------------+
| id_detalle | cantidad | precio_unitario | id_pedido | id_producto |
+------------+----------+-----------------+-----------+-------------+
|         26 |        2 |        12999.99 |        21 |           1 |
+------------+----------+-----------------+-----------+-------------+
```

Tenia 4 productos en stock y bajo a 2.

```
mysql> SELECT * FROM productos WHERE id_producto = 1;
+-------------+---------------+-----------------------------+----------+-------+-------------+
| id_producto | nombre        | descripcion                 | precio   | stock | id_categoria |
+-------------+---------------+-----------------------------+----------+-------+-------------+
|           1 | HP Pavilion 15 | Laptop con Intel i5 y 8GB RAM | 12999.99 |     2 |           1 |
+-------------+---------------+-----------------------------+----------+-------+-------------+
```

ESCENARIO: Registrando pedido donde se piden mas 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 si pidió.

```
mysql> CALL sp_registrar_resena(3, 1, 1, 4, 'Muy buen producto', '2025-07-30');
Query OK, 1 row affected (0.02 sec)
```

```
mysql> SELECT * FROM resenas WHERE id_pedido = 1;
+-----------+------------+-----------+-------------+-------------+------------------+------------+
| id_resena | id_cliente | id_pedido | id_producto | calificacion | comentario       | fecha      |
+-----------+------------+-----------+-------------+-------------+------------------+------------+
|        10 |          3 |         1 |           1 |           4 | Muy buen producto | 2025-07-30 |
+-----------+------------+-----------+-------------+-------------+------------------+------------+
```

```
mysql> SELECT * FROM detalles_pedido WHERE id_producto = 1 AND id_pedido= 1;
+-------------+----------+-----------------+-----------+-------------+
| id_detalle | cantidad | precio_unitario | id_pedido | id_producto |
+-------------+----------+-----------------+-----------+-------------+
|          1 |        1 |        12999.99 |         1 |           1 |
+-------------+----------+-----------------+-----------+-------------+
```

ESCENARIO: Hacer reseña sin comprar el producto

```
mysql> CALL sp_registrar_resena(4, 1, 1, 4, 'Muy buen producto', '2025-07-30');
ERROR 1644 (45000): El cliente no ha comprado este producto
mysql>
```

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

ESCENARIO: Se vendieron 5 teléfonos Poco 11T, se tiene que actualizar el stock.

El producto tiene 11 productos en stock:

```
|          14 | Poco 11T              |    11 |
```

Se usa el procedimiento y se muestra el nuevo stock:

```
mysql> CALL sp_actualizar_stock(14,5);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT id_producto,nombre,stock FROM productos WHERE id_producto=14;
+-------------+----------+-------+
| id_producto | nombre   | stock |
+-------------+----------+-------+
|          14 | Poco 11T |     6 |
+-------------+----------+-------+
```

ESCENARIO: no hay suficiente stock.

```
mysql> CALL sp_actualizar_stock(14,10);
ERROR 1644 (45000): Este producto no tiene stock
```

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_estado=1 AND id_pedido=4;
+-----------+--------------+-----------+------------+
| id_pedido | fecha_pedido | id_estado | id_cliente |
+-----------+--------------+-----------+------------+
|         4 | 2025-07-04   |         1 |          2 |
+-----------+--------------+-----------+------------+
1 row in set (0.00 sec)

mysql> CALL sp_cambiar_estado(4,3);
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM pedidos WHERE id_pedido=4;
+-----------+--------------+-----------+------------+
| id_pedido | fecha_pedido | id_estado | id_cliente |
+-----------+--------------+-----------+------------+
|         4 | 2025-07-04   |         3 |          2 |
+-----------+--------------+-----------+------------+
```

ESCENARIO: Se ingreso 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 especifico

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

```
mysql> SELECT * FROM resenas WHERE id_producto=1;
+-----------+------------+-----------+-------------+--------------+-------------------+------------+
| id_resena | id_cliente | id_pedido | id_producto | calificacion | comentario        | fecha      |
+-----------+------------+-----------+-------------+--------------+-------------------+------------+
|        10 |          3 |         1 |           1 |            4 | Muy buen producto | 2025-07-30 |
+-----------+------------+-----------+-------------+--------------+-------------------+------------+
1 row in set (0.00 sec)
```

```
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 categoria 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 numero 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 especifico 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

Creo se podrían agregar más índices para consultas comunes para la tienda como pueden ser:

- Búsqueda de pedido por fecha o por estado.

- Búsqueda de los detalles del pedido por el id del pedido.

- Busqueda de la información del usuario por el correo.

Se podrían integrar unos triggers para que en lugar del procedimiento de revisar si el usuario compro el producto para hacer la reseña desde el trigger al querer hacer la reseña revise si ese usuario compro el producto y solo lo deje ingresar la reseña si compro ese producto, también podría existir el trigger para limitar que un usuario tenga máximo 5 pedidos con estado pendiente en lugar del procedimiento.

Tal vez se podrían agregar alguna vista para evitar o más bien facilitar el escribir consultas largas o frecuentes.

CONCLUSIONES Y APRENDIZAJES.

El hacer este proyecto creo fue interesante y útil ya que me ayudo a fortalecer los temas aprendidos en clase y además me deja con varios aprendizajes.

Principalmente creo que con esta base de datos me di cuenta de la importancia de diseñar correctamente una base de datos para cualquier sistema que se nos requiera ya que pues en todas o la mayoría de los casos los sistemas se construyen para utilizar, analizar y manipular los datos que guardamos en ella.

Cuando una base de datos esta bien construida ayudara a sus usuarios a poder manejar su información de una manera mas eficiente, lo que es muy importante para cualquier persona o empresa ya que en la industria el tiempo es un aspecto clave y es muy importante tanto para los negocios como para los usuarios que se gestione de la mejor manera.

En conclusión, como profesionales en formación que somos, el saber crear y manejar bases de datos de cualquier tipo será fundamental para cuando nos adentremos al mundo laboral ya que en un área en constante crecimiento y competencia habilidades como las que estamos desarrollando pueden marcar la diferencia en nuestro desarrollo profesional.