



## **Fundamentos de Bases de Datos**

**Sistema de Gestión de Inventario para una tienda de tecnología**

**Estudiante**

Maria Isabel Vallejos Rodriguez

**Profesor:**

Brayner Salmeron Castillo

Noviembre 2024

<b>Introducción</b>	<b>3</b>
<b>Descripción del problema</b>	<b>3</b>
Objetivos del sistema	3
<b>Modelo Entidad-Relación</b>	<b>4</b>
<b>Modelo Relacional</b>	<b>6</b>
<b>Estándares de Bases de Datos</b>	<b>7</b>
Convenciones Generales	7
Tablas	7
Columnas	7
<b>Modelo Físico</b>	<b>8</b>
<b>Objetos Programados</b>	<b>8</b>
Tablas	8
Productos	8
Proveedores	9
Compras	10
Ventas	11
Clientes	12
Categorías	13
MetodosPagos	14
Devoluciones	15
Rel_ProductosVentas	16
Rel_ProductosCompras	17
Rel_ProductosProveedores	18
Funciones Escalares	19
fn_ValorTotalInventario	19
fn_ProductosVendidosPeriodo	19
Función de Tabla	20
fn_ProductosStockBajo	20
Procedimientos Almacenados	20
sp_RegistrarVenta	20
sp_RegistrarCompra	21
Vistas	22
vw_ProductosStock	22
vw_VentasClientesFecha	22
vw_ComprasProveedoresFecha	23
Triggers	24
tr_ActualizarStockVenta	24
tr_ActualizarStockCompra	24
<b>Conclusiones</b>	<b>25</b>
<b>Recomendaciones</b>	<b>25</b>
<b>Fuentes de información</b>	<b>26</b>

# Introducción

En el presente documento se detalla el proceso de diseño y de implementación de una base de datos para un sistema de gestión de inventario para una tienda de tecnología. Este sistema está diseñado para gestionar eficientemente el control de inventario de productos, proveedores, ventas y la relación con los clientes, proporcionando una solución integral para las necesidades operativas del negocio.

El sistema propuesto utiliza una base de datos robusta, implementada con las mejores prácticas de diseño y seguridad. La implementación se realizará utilizando el Gestor de Bases de Datos SQL Server, y el lenguaje T-SQL para los scripts de creación de tablas, restricciones y objetos avanzados.

Este documento detalla todo el proceso, desde el modelado de la base de datos hasta la implementación de funciones, procedimientos almacenados, vistas y triggers.

Todo el proyecto se puede ver en el siguiente repositorio de Github:  
[https://github.com/isvallrod/ProyectoBasesDeDatos\\_Nov24](https://github.com/isvallrod/ProyectoBasesDeDatos_Nov24)

## Descripción del problema

La tienda necesita un sistema de gestión de inventarios que permita controlar de manera eficiente el stock de productos, gestionar proveedores, registrar compras y ventas, y generar informes que apoyen la toma de decisiones. Este sistema debe ser robusto y escalable, permitiendo manejar un gran volumen de datos y adaptarse al crecimiento de la tienda.

## Objetivos del sistema

1. **Control de Inventario:** Mantener un registro actualizado y preciso del inventario.
2. **Gestión de Proveedores:** Administrar información y transacciones con proveedores de manera eficiente.
3. **Registro de Ventas:** Detallar las ventas realizadas, con información de los clientes y productos vendidos.
4. **Generación de Informes:** Producir reportes que apoyen la toma de decisiones.

# Modelo Entidad-Relación

La primera etapa del proyecto consiste en obtener el modelo Entidad-Relación. Este modelo se obtiene mediante el análisis de los requerimientos del negocio.

Además, como parte de la implementación se solicitó agregar dos entidades nuevas para mejorar la solución.

Se decidió implementar la entidad de Métodos de Pago para gestionar los diferentes métodos de pagos disponibles, además también se agregó la entidad de Devolución que es una característica muy importante en todas las tiendas de venta.

## **Categorías-Productos (Uno a Muchos)**

- Una categoría puede contener múltiples productos
- Cada producto pertenece a una única categoría

## **Productos-Ventas (Muchos a Muchos con tabla intermedia)**

### **Justificación:**

- Un producto puede estar en múltiples ventas
- Una venta puede contener múltiples productos
- Tabla intermedia (rel\_ProductosVentas)

## **Productos-Compras (Muchos a Muchos con tabla intermedia)**

### **Justificación:**

- Un producto puede estar en múltiples compras de inventario
- Una compra puede contener múltiples productos
- Tabla intermedia (rel\_ProductosCompras)

## **Productos-Proveedores (Muchos a Muchos con tabla intermedia)**

### **Justificación:**

- Un producto puede ser suministrado por múltiples proveedores
- Un proveedor puede suministrar múltiples productos
- Tabla intermedia (rel\_ProductosVentas)

## **Productos-Devoluciones (Uno a Muchos)**

### **Justificación:**

- Un producto puede tener múltiples devoluciones
- Cada devolución está asociada a un único producto

## **Ventas-Devoluciones (Uno a Muchos)**

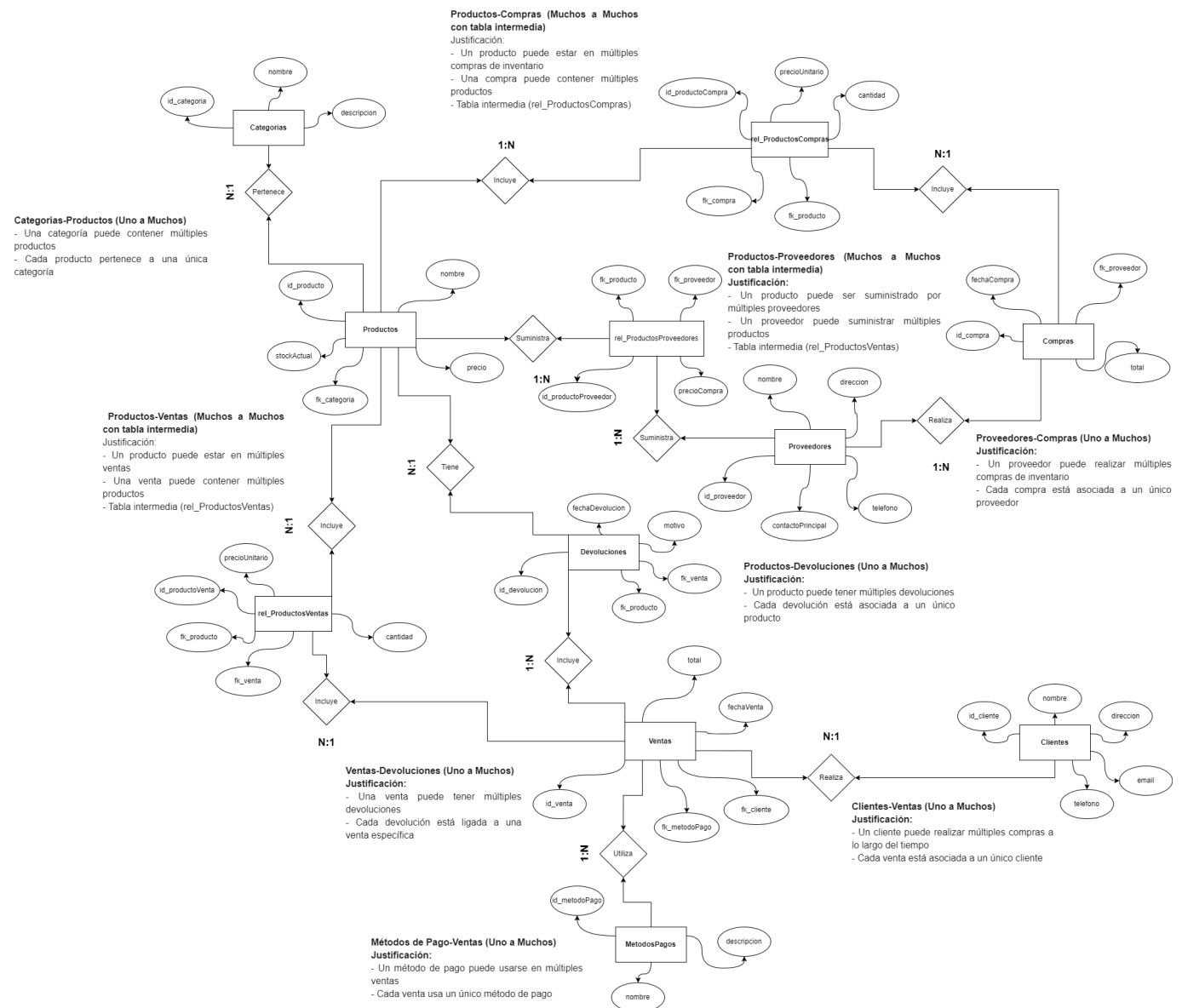
### **Justificación:**

- Una venta puede tener múltiples devoluciones
- Cada devolución está ligada a una venta específica

- Un cliente puede realizar múltiples compras a lo largo del tiempo
- Cada venta está asociada a un único cliente

- Un método de pago puede usarse en múltiples ventas
- Cada venta usa un único método de pago

- Un proveedor puede realizar múltiples compras de inventario
- Cada compra está asociada a un único proveedor



# Modelo Relacional

En las siguientes imágenes se muestra el Modelo Relacional, donde podemos ver las relaciones entre las tablas anteriormente mencionadas. Además se muestra el tipo de datos a utilizar para cada uno de los campos, así como las restricciones asociadas en caso de ser necesario, por ejemplo si se permite o no tener valores NULL.



# Estándares de Bases de Datos

Para garantizar la consistencia y el entendimiento del proyecto, se seguirán los siguientes estándares:

## Convenciones Generales

La convención de nomenclatura permitida para todos los nombres será PascalCase y camelCase.

No se utilizarán caracteres especiales ni palabras reservadas.

Se utilizarán nombres en español para mantener la consistencia.

Longitud máxima de nombres: 30 caracteres.

## Tablas

- Nombres en plural
- La convención de nombres será PascalCase
- Prefijo según tipo de tabla:
  - Tablas base: ninguno (ej: Producto)
  - Tablas de relación: rel\_ (ej: rel\_ProductoProveedor)

## Columnas

- Nombres descriptivos y concisos en singular
- La convención de nombres será camelCase
- Prefijos según tipo:
  - Primary Key: id\_ (ej: id\_producto)
  - Foreign Key: fk\_ seguido del nombre de la tabla referenciada (ej: fk\_categoria)

# Modelo Físico

En el Modelo Físico se detalla la estructura física de la base de datos:

- Descripción de cada tabla: columnas, tipos de datos, si aceptan nulos, restricciones, etc.
- Explicación de los índices implementados para optimizar el rendimiento.
- Capturas de pantalla o diagramas del diseño físico en la herramienta seleccionada.

Cada uno de los puntos anteriores describe los objetos programados en SQL, por lo que en la siguiente sección se detallan los puntos anteriores.

## Objetos Programados

En la siguiente sección se describen las diferentes tablas utilizadas en el proyecto:

### Tablas

#### Productos

##### Descripción

Contiene el catálogo de productos disponibles en la tienda.

##### Columnas

`id_producto (INT, PK, Identity(1,1), NOT NULL)`  
`nombre (VARCHAR(100), NOT NULL, UNIQUE)`  
`precio (DECIMAL(10,2), NOT NULL)`  
`fk_categoria (INT, FK, NOT NULL)`  
`stockActual (INT, NOT NULL, DEFAULT 0)`

##### Referencias

Referencia a:  
Categorias (fk\_categoria)  
Referenciada por:  
rel\_ProductosVentas (fk\_producto)  
rel\_ProductosCompras (fk\_producto)  
rel\_ProductosProveedores (fk\_producto)  
Devoluciones (fk\_producto)



### Ejemplo de Datos:

```
INSERT INTO Productos (nombre, precio, fk_categoria, stock) VALUES  
('Laptop Dell XPS 13',999999.99, 1, 10),  
('Mouse Logitech MX', 49999.99, 2, 25),  
('Monitor LG 27', 299999.99, 3, 15);
```

### Diseño Físico:

	id_producto	nombre	precio	stockActual	fk_categoria
1	1	Laptop Dell XPS 13	1299.99	15	1
2	2	iPhone 15 Pro	999.99	25	2
3	3	Samsung Galaxy Tab S9	649.99	20	3
4	4	Logitech MX Master 3	99.99	30	4
5	5	LG 27" 4K Monitor	399.99	10	5
6	6	AMD Ryzen 9 5950X	549.99	8	6
7	7	Sony WH-1000XM4	349.99	12	7
8	8	PlayStation 5	499.99	5	8
9	9	Samsung 1TB SSD	129.99	40	9
10	10	TP-Link Archer AX6000	299.99	15	10

### Proveedores

#### Descripción

Almacena la información de los proveedores de la tienda.

#### Columnas

```
id_proveedor (INT, PK, Identity(1,1), NOT NULL)  
nombre (VARCHAR(100), NOT NULL)  
direccion (VARCHAR(100), NOT NULL)  
contactoPrincipal (VARCHAR(100), NULL)  
telefono (VARCHAR(20), NULL)
```

### Referencias

Referenciada por:

Compras (fk\_proveedor)

Rel\_ProductosProveedores (fk\_proveedor)

## Ejemplo de Datos

```
INSERT INTO Proveedores (nombre, direccion, contactoPrincipal, telefono)
VALUES
```

```
('TechSupply Inc', 'Costa Rica', 'John Smith', '8888-0001'),
('Global Electronics', 'Costa Rica', '8888-0002'),
('Digital Imports', 'Estados Unidos', '8888-0003');
```

## Diseño Físico:

	id_proveedor	nombre	direccion	contactoPrincipal	telefono
1	1	TechWorld Inc	Av. Silicon Valley 123, CA	John Smith	+1-555-0123
2	2	Electrónica Global	Calle Principal 456, Madrid	María García	+34-612345678
3	3	AsiaComp	Dragon Street 789, Hong Kong	Lee Wong	+852-98765432
4	4	InnoTech	Tech Park 321, Seúl	Kim Lee	+82-1012345678
5	5	Digital Solutions	Digital Ave 654, Toronto	Mike Johnson	+1-416-555-0199
6	6	MegaComp	Tech Boulevard 987, Berlín	Hans Mueller	+49-30-12345678
7	7	ElectroMex	Reforma 246, CDMX	Carlos Ruiz	+52-55-12345678
8	8	TechnoIndia	Tech Park 111, Bangalore	Raj Patel	+91-9876543210
9	9	BrasilTech	Av Paulista 333, São Paulo	Pedro Santos	+55-11-98765432
10	10	ArgentinaDigital	Av Corrientes 555, Buenos Aires	Laura Pérez	+54-911-12345678

## Compras

### Descripción

Registra las compras realizadas a proveedores.

### Columnas

```
id_compra (INT, PK, Identity(1,1), NOT NULL)
fechaCompra (DATE, NOT NULL, DEFAULT GETDATE())
total (DECIMAL(12,2), NOT NULL)
fk_proveedor (INT, FK, NOT NULL)
```

### Referencias

Referencia a:

Proveedores (fk\_proveedor)

Referenciada por:

Rel\_ProductosCompras (fk\_compra)

### Ejemplo de Datos

```
INSERT INTO Compras(fecha_compra, total, fk_proveedor) VALUES
('2024-03-20', 5000000.00, 1),
('2024-03-21', 3000000.00, 2),
('2024-03-22', 2500000.00, 3);
```

### Diseño Físico:

	id_compra	fechaCompra	total	fk_proveedor
1	1	2024-01-10 09:00:00.000	5000.00	1
2	2	2024-01-11 10:30:00.000	7500.00	2
3	3	2024-01-12 11:45:00.000	3500.00	3
4	4	2024-01-13 14:20:00.000	6000.00	4
5	5	2024-01-14 16:15:00.000	4500.00	5
6	6	2024-01-15 13:30:00.000	8000.00	6
7	7	2024-01-16 15:45:00.000	3000.00	7
8	8	2024-01-17 12:10:00.000	5500.00	8
9	9	2024-01-18 10:25:00.000	4000.00	9
10	10	2024-01-19 11:50:00.000	6500.00	10

### Ventas

#### Descripción

Registra las ventas realizadas a clientes.

#### Columnas

```
id_venta (INT, PK, Identity(1,1), NOT NULL)
fechaVenta (DATE, NOT NULL, DEFAULT GETDATE())
total (DECIMAL(12,2), NOT NULL)
fk_cliente (INT, FK, NOT NULL)
fk_metodoPago (INT, FK, NOT NULL)
```

### Referencias

```
Referencia a:
Clientes (fk_cliente)
MetodosPagos (fk_metodoPago)
Referenciada por:
Rel_Productos_Ventas (fk_venta)
Devoluciones (fk_venta)
```

### Ejemplo de Datos

```
INSERT INTO Ventas (fechaVenta, total, fk_cliente, fk_metodoPago) VALUES ('2024-03-20', 1049999.99, 1, 1), ('2024-03-21', 349999.99, 2, 2), ('2024-03-22', 999999.99, 3, 1);
```

### Diseño Físico:

	id_venta	fechaVenta	total	fk_cliente	fk_metodoPago
1	1	2024-01-15 10:30:00.000	1499.99	1	2
2	2	2024-01-16 15:45:00.000	999.99	2	1
3	3	2024-01-17 12:20:00.000	749.99	3	3
4	4	2024-01-18 16:30:00.000	1299.99	4	4
5	5	2024-01-19 11:15:00.000	449.99	5	2
6	6	2024-01-20 14:40:00.000	899.99	6	1
7	7	2024-01-21 13:25:00.000	599.99	7	3
8	8	2024-01-22 17:10:00.000	1099.99	8	5
9	9	2024-01-23 10:55:00.000	799.99	9	2
10	10	2024-01-24 15:30:00.000	649.99	10	4

### Clientes

#### Descripción

Almacena la información de los clientes de la tienda.

#### Columnas

```
id_cliente (INT, PK, Identity(1,1), NOT NULL)
nombre (VARCHAR(100), NOT NULL)
direccion (VARCHAR(100), NOT NULL)
email (VARCHAR(100), NOT NULL, UNIQUE)
telefono (VARCHAR(20), NULL)
```

### Referencias

```
Referenciada por: Ventas (fk_cliente)
```

### Ejemplo de Datos

```
INSERT INTO CLIENTES (nombre, direccion, email, telefono) VALUES ('Juan Pérez', 'juan.perez@email.com', 'Cartago', '8888-1234'), ('María López', 'maria.lopez@email.com', 'Guanacaste', '8888-5678'),
```

```
('Carlos Rodríguez', 'carlos.rodriguez@email.com', 'Alajuela', '8888-9012');
```

## Diseño Físico:

	id_cliente	nombre	direccion	email	telefono
1	1	Ana Martínez	Calle 123, San José	ana.martinez@email.com	+506-555-1234
2	2	Pedro González	Av Principal 456, Heredia	pedro.g@email.com	+506-612345677
3	3	María López	Plaza Mayor 789, Alajuela	maria.lopez@email.com	+506-87654321
4	4	Juan Rodríguez	Carrera 10 #45, Cartago	juan.r@email.com	+506-310-1234567
5	5	Carlos Silva	Rua Augusta 123, Liberia	carlos.s@email.com	+506-12345678
6	6	Laura Torres	Av Libertador 456, Puntarenas	laura.t@email.com	+506-87654321
7	7	José Ramírez	Calle Real 789, Limón	jose.r@email.com	+506-999-123456
8	8	Andrea Castro	Av República 321, San Carlos	andrea.c@email.com	+506-87654321
9	9	Miguel Flores	Calle Luna 654, Grecia	miguel.f@email.com	+506-99-1234567
10	10	Carmen Díaz	Av Sol 987, Pérez Zeledón	carmen.d@email.com	+506-414-1234567

## Categorías

### Descripción

Clasifica los productos en diferentes categorías

### Columnas

id\_categoria (INT, PK, Identity(1,1), NOT NULL)  
nombre (VARCHAR(50), NOT NULL, UNIQUE)  
descripcion (VARCHAR(200), NULL)

### Referencias

Referenciada por: Productos (fk\_categoria)

## Ejemplo de Datos

```
INSERT INTO Categorías (nombre, descripcion) VALUES  
( 'Laptops', 'Computadoras portátiles'),  
( 'Periféricos', 'Accesorios y dispositivos externos'),  
( 'Monitores', 'Pantallas y displays');
```

## Diseño Físico:

	id_categoria	nombre	descripcion
1	1	Laptops	Computadoras portátiles de diferentes marcas
2	2	Smartphones	Teléfonos inteligentes de última generación
3	3	Tablets	Tabletas de diversos tamaños y marcas
4	4	Accesorios	Periféricos y accesorios varios
5	5	Monitores	Pantallas y monitores de diferentes tamaños
6	6	Componentes PC	Componentes para computadoras de escritorio
7	7	Audio	Equipos de sonido y audio
8	8	Gaming	Productos para videojuegos
9	9	Almacenamiento	Dispositivos de almacenamiento
10	10	Redes	Equipos de conectividad y redes

## MetodosPagos

### Descripción

Catálogo de métodos de pago disponibles.

### Columnas

id\_metodoPago (INT, PK, Identity(1,1), NOT NULL)  
nombre (VARCHAR(50), NOT NULL, UNIQUE)  
descripcion (VARCHAR(200), NULL)

### Referencias

Referenciada por: Ventas (fk\_metodoPago)

### Ejemplo de Datos

```
INSERT INTO MetodosPagos (nombre, descripcion) VALUES
('Tarjeta', 'Pago con tarjeta de crédito o débito'),
('Efectivo', 'Pago en efectivo'),
('Transferencia', 'Transferencia bancaria');
```

### Diseño Físico:

	id_metodoPago	nombre	descripcion
1	1	Efectivo	Pago en efectivo al momento de la compra
2	2	Tarjeta Crédito	Pago con tarjeta de crédito
3	3	Tarjeta Débito	Pago con tarjeta de débito
4	4	PayPal	Pago a través de cuenta PayPal
5	5	Transferencia	Transferencia bancaria
6	6	Bitcoin	Pago con criptomoneda Bitcoin
7	7	Financiamiento	Pago en cuotas con financiamiento
8	8	Cheque	Pago con cheque bancario
9	9	Mercado Pago	Pago a través de Mercado Pago
10	10	Google Pay	Pago mediante Google Pay

# Devoluciones

## Descripción

Registra las devoluciones de productos vendidos.

## Columnas

```
id_devolucion (INT, PK, Identity(1,1), NOT NULL)
fechaDevolucion (DATE, NOT NULL, DEFAULT GETDATE())
motivo (VARCHAR(200), NOT NULL)
fk_venta (INT, FK, NOT NULL)
fk_producto (INT, FK, NOT NULL)
```

## Referencias

```
Referencia a:
Ventas(fk_venta)
Productos (fk_producto)
```

## Ejemplo de Datos

```
INSERT INTO Devoluciones (fecha_devolucion, motivo, fk_venta,
fk_producto) VALUES
('2024-03-22', 'Producto defectuoso', 1, 1),
('2024-03-23', 'No cumple expectativas', 2, 2),
('2024-03-24', 'Error en especificaciones', 3, 3);
```

## Diseño Físico:

	id_devolucion	fechaDevolucion	motivo	fk_venta	fk_producto
1	1	2024-01-17 11:30:00.000	Producto defectuoso	1	1
2	2	2024-01-18 14:45:00.000	No cumple expectativas	2	2
3	3	2024-01-19 16:20:00.000	Error en el pedido	3	3
4	4	2024-01-20 13:15:00.000	Producto dañado en transporte	4	4
5	5	2024-01-21 15:40:00.000	Falla técnica	5	5
6	6	2024-01-22 12:25:00.000	Incompatibilidad de sistema	6	6
7	7	2024-01-23 17:10:00.000	Producto incorrecto	7	7
8	8	2024-01-24 10:55:00.000	No funciona correctamente	8	8
9	9	2024-01-25 14:30:00.000	Daño físico	9	9
10	10	2024-01-26 16:15:00.000	Insatisfacción del cliente	10	10

## Rel\_ProductosVentas

### Descripción

Tabla intermedia que relaciona productos con ventas.

### Columnas

id\_productoVenta (INT, PK, Identity(1,1), NOT NULL)  
fk\_producto (INT, FK, NOT NULL)  
fk\_venta (INT, FK, NOT NULL)  
cantidad (INT, NOT NULL)  
precioUnitario (DECIMAL(10,2), NOT NULL)

### Referencias

Referencia a:  
Productos (fk\_producto)  
Ventas (fk\_venta)

### Ejemplo de Datos

```
INSERT INTO rel_ProductosVentas (fk_producto, fk_venta, cantidad,  
precioUnitario) VALUES  
(1, 1, 1, 999999.99),  
(2, 2, 1, 49999.99),  
(3, 3, 1, 299999.99);
```

### Diseño Físico:

	id_producto_venta	fk_producto	fk_venta	cantidad	precioUnitario
1	1	1	1	1	1299.99
2	2	2	2	1	999.99
3	3	3	3	1	649.99
4	4	4	4	2	99.99
5	5	5	5	1	399.99
6	6	6	6	1	549.99
7	7	7	7	1	349.99
8	8	8	8	2	499.99
9	9	9	9	3	129.99
10	10	10	10	1	299.99



## Rel\_ProductosCompras

### Descripción

Tabla intermedia que relaciona productos con compras.

### Columnas

id\_productoCompra (INT, PK, Identity(1,1), NOT NULL)  
fk\_producto (INT, FK, NOT NULL)  
fk\_compra (INT, FK, NOT NULL)  
cantidad (INT, NOT NULL)  
precioUnitario (DECIMAL(10,2), NOT NULL)

### Referencias

Referencia a:  
Productos (fk\_producto)  
Compras (fk\_compra)

### Ejemplo de Datos

```
INSERT INTO rel_ProductosCompras (fk_producto, fk_compra, cantidad,
precioUnitario) VALUES
(1, 1, 5, 800000.00),
(2, 2, 10, 35000.00),
(3, 3, 8, 250000.00);
```

### Diseño Físico:

	id_producto_compra	fk_producto	fk_compra	cantidad	precioUnitario
1	1	1	1	5	1000.00
2	2	2	2	10	750.00
3	3	3	3	7	500.00
4	4	4	4	15	75.00
5	5	5	5	5	300.00
6	6	6	6	4	400.00
7	7	7	7	6	250.00
8	8	8	8	3	400.00
9	9	9	9	20	100.00
10	10	10	10	8	200.00

## Rel\_ProductosProveedores

### Descripción

Tabla intermedia que relaciona productos con sus proveedores.

### Columnas

id\_productoProveedor (INT, PK, Identity(1,1), NOT NULL)  
fk\_producto (INT, FK, NOT NULL)  
fk\_proveedor (INT, FK, NOT NULL)  
precioCompra (DECIMAL(10,2), NOT NULL)

### Referencias

Referencia a:

Productos (fk\_producto)

Proveedores (fk\_proveedor)

### Ejemplo de Datos

```
INSERT INTO rel_ProductosProveedores (fk_producto, fk_proveedor,  
precioCompra) VALUES  
(1, 1, 800000.00),  
(2, 2, 35000.00),  
(3, 3, 250000.00);
```

### Diseño Físico:

	id_producto_proveedor	fk_producto	fk_proveedor	precioCompra
1	1	1	1	1000.00
2	2	2	2	750.00
3	3	3	3	500.00
4	4	4	4	75.00
5	5	5	5	300.00
6	6	6	6	400.00
7	7	7	7	250.00
8	8	8	8	400.00
9	9	9	9	100.00
10	10	10	10	200.00

## Funciones Escalares

### fn\_ValorTotalInventario

#### Propósito:

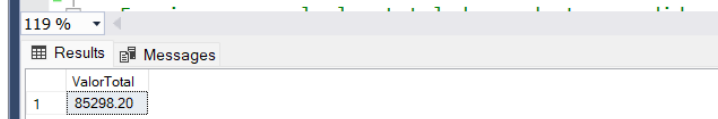
Calcular el valor total del inventario actual sumando el producto del precio unitario por la cantidad en stock de cada producto.

#### Lógica implementada:

- Multiplica el precio unitario por el stock actual de cada producto
- Suma todos los resultados para obtener el valor total
- Retorna un valor decimal con 2 decimales

#### Ejemplo de uso:

```
-- Consultar valor total del inventario
SELECT dbo.fn_ValorTotalInventario() as ValorTotal
```



ValorTotal
85298.20

### fn\_ProductosVendidosPeriodo

#### Propósito:

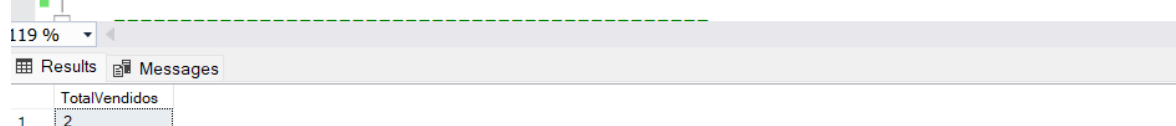
Calcular la cantidad total de productos vendidos durante un período específico definido por fechas.

#### Lógica implementada:

- Multiplica el precio unitario por el stock actual de cada producto
- Suma todos los resultados para obtener el valor total
- Retorna un valor decimal con 2 decimales

#### Ejemplo de uso:

```
-- Calcular productos vendidos en un periodo
SELECT dbo.fn_ProductosVendidosPeriodo('2024-01-01', '2024-01-17') as TotalVendidos
```



TotalVendidos
2

## Función de Tabla

### fn\_ProductosStockBajo

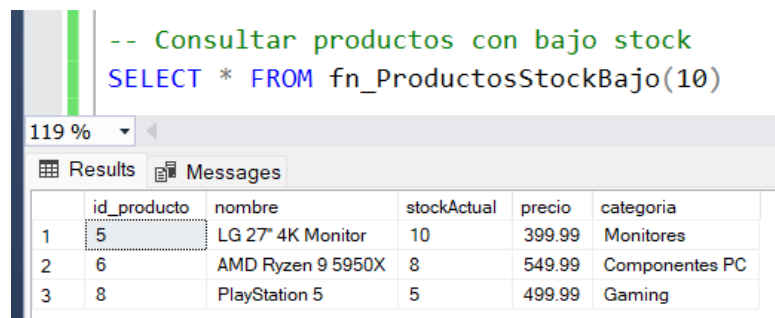
#### Propósito:

Identificar productos cuyo stock actual es igual o menor a un nivel mínimo ingresado.

#### Lógica implementada:

- Recibe un parámetro de stock mínimo
- Retorna una tabla con información detallada de productos
- Incluye join con la tabla de categorías para agregar información de los productos
- Filtra los productos con stock menor o igual al valor ingresado

#### Ejemplo de uso:



The screenshot shows a SQL query window with the following text:

```
-- Consultar productos con bajo stock  
SELECT * FROM fn_ProductosStockBajo(10)
```

Below the query window, the 'Results' tab is active, displaying a table with 6 columns: id\_producto, nombre, stockActual, precio, and categoria. The table contains 3 rows of data.

	id_producto	nombre	stockActual	precio	categoria
1	5	LG 27" 4K Monitor	10	399.99	Monitores
2	6	AMD Ryzen 9 5950X	8	549.99	Componentes PC
3	8	PlayStation 5	5	499.99	Gaming

## Procedimientos Almacenados

### sp\_RegistrarVenta

#### Propósito:

Se encarga del proceso de registrar una nueva venta, incluyendo la actualización de campos relacionados.

#### Lógica implementada:

- Registra la venta principal con el total calculado
- Registra los productos vendidos en la tabla de relación
- Utiliza SCOPE\_IDENTITY() para mantener la integridad referencial
- Maneja las transacciones

## Ejemplo de uso:

```
-- Registrar una venta
EXEC sp_RegistrarVenta 1, 1, 1, 2, 100.00
```

19 %

Messages

(1 row affected)

(1 row affected)

## sp\_RegistrarCompra

### Propósito:

Se encarga del proceso de registrar una nueva compra, incluyendo la actualización de campos relacionados.

### Lógica implementada:

- Registra la compra principal con el total calculado
- Registra los productos comprados en la tabla de relación
- Mantiene la integridad referencial
- Simplifica el proceso de registro de compras

## Ejemplo de uso:

```
-- Registrar una compra
EXEC sp_RegistrarCompra 1, 1, 5, 80.00
```

119 %

Messages

(1 row affected)

(1 row affected)

Completion time: 2024-11-29T22:12:23.2285681-06:00

## Vistas

## vw\_ProductosStock

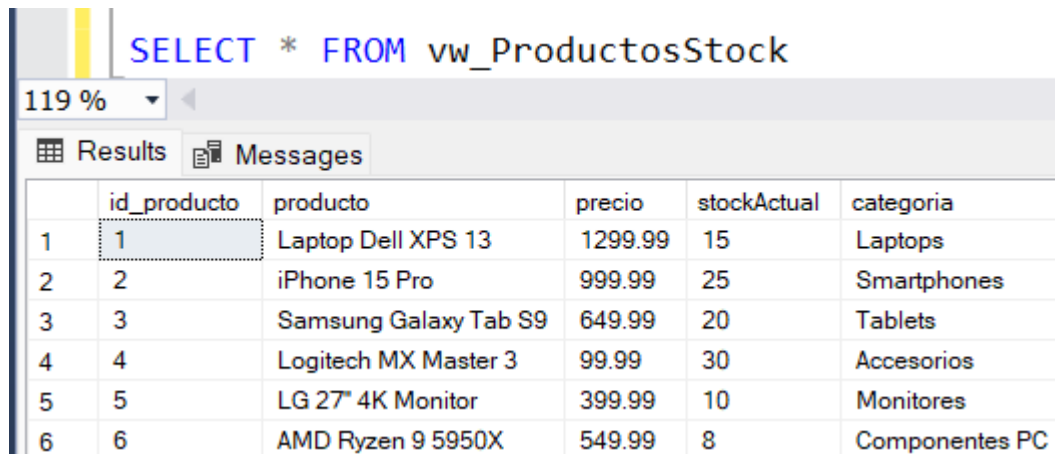
### Propósito:

Muestra una vista de todos los productos con su información de stock y categoría.

**Lógica implementada:**

- Combina información de productos y categorías mediante un JOIN
- Muestra todos los detalles importantes de productos y categorías

**Ejemplo de uso:**



```
SELECT * FROM vw_ProductosStock
```

	id_producto	producto	precio	stockActual	categoria
1	1	Laptop Dell XPS 13	1299.99	15	Laptops
2	2	iPhone 15 Pro	999.99	25	Smartphones
3	3	Samsung Galaxy Tab S9	649.99	20	Tablets
4	4	Logitech MX Master 3	99.99	30	Accesorios
5	5	LG 27" 4K Monitor	399.99	10	Monitores
6	6	AMD Ryzen 9 5950X	549.99	8	Componentes PC

**vw\_VentasClientesFecha**

**Propósito:**

Muestra una vista detallada de las ventas realizadas, incluyendo información del cliente y además se agrega información del método de pago para aprovechar que tenemos estos datos.

**Lógica implementada:**

- Relaciona ventas con clientes y métodos de pago mediante un JOIN
- Muestra información detallada de las ventas

**Ejemplo de uso:**

<pre>-- Ver ventas por cliente SELECT * FROM vw_VentasClientesFecha</pre>				
<div> <div>%</div> <div>Results Messages</div> </div>				
id_venta	fechaVenta	cliente	total	metodoPago
1	2024-01-15 10:30:00.000	Ana Martínez	1499.99	Tarjeta Crédito
2	2024-01-16 15:45:00.000	Pedro González	999.99	Efectivo
3	2024-01-17 12:20:00.000	María López	749.99	Tarjeta Débito
4	2024-01-18 16:30:00.000	Juan Rodríguez	1299.99	PayPal
5	2024-01-19 11:15:00.000	Carlos Silva	449.99	Tarjeta Crédito
6	2024-01-20 14:40:00.000	Laura Torres	899.99	Efectivo

vw\_ComprasProveedoresFecha

**Propósito:**

Muestra una vista de las compras realizadas a proveedores con información de la fecha y precio

**Lógica implementada:**

- Combina información de compras y proveedores mediante un JOIN
- Muestra información detallada de las compras y proveedores

**Ejemplo de uso:**

<pre>-- Ver compras por proveedor SELECT * FROM vw_ComprasProveedoresFecha</pre>				
<div> <div>19 %</div> <div>Results Messages</div> </div>				
	id_compra	fechaCompra	proveedor	total
1	1	2024-01-10 09:00:00.000	TechWorld Inc	5000.00
2	2	2024-01-11 10:30:00.000	Electrónica Global	7500.00
3	3	2024-01-12 11:45:00.000	AsiaComp	3500.00
4	4	2024-01-13 14:20:00.000	InnoTech	6000.00
5	5	2024-01-14 16:15:00.000	Digital Solutions	4500.00
6	6	2024-01-15 13:30:00.000	MegaComp	8000.00

# Triggers

## tr\_ActualizarStockVenta

### Propósito:

Actualizar el stock de un producto al registrarse una venta.

### Lógica implementada:

- Se activa después de insertar en rel\_ProductosVentas
- Reduce el stock actual del producto vendido
- Utiliza la tabla inserted para acceder a los datos de la transacción

### Ejemplo de uso:

```
-- El trigger se activa automáticamente después de:
INSERT INTO rel_ProductosVentas (fk_producto, fk_venta, cantidad, precioUnitario)
VALUES (1, 1, 5, 99.99);
```

9 %

Messages

(1 row affected)

(1 row affected)

Completion time: 2024-11-29T23:04:41.3635086-06:00

## tr\_ActualizarStockCompra

### Propósito:

Actualizar el stock de un producto al registrarse una compra.

### Lógica implementada:

- Se activa después de insertar en rel\_ProductosCompras
- Incrementa el stock actual del producto comprado
- Utiliza la tabla inserted para acceder a los datos de la transacción

### Ejemplo de uso:

```
-- El trigger se activa automáticamente después de:
INSERT INTO rel_ProductosCompras (fk_producto, fk_compra, cantidad, precioUnitario)
VALUES (1, 1, 50, 75.00);
```

9 %

Messages

(1 row affected)

(1 row affected)

Completion time: 2024-11-29T23:05:11.7236555-06:00



# Conclusiones

- El diseño robusto del modelo Entidad-Relación proporcionó las bases para un buen diseño para la implementación del modelo en SQL Server.
- Se logró implementar exitosamente un conjunto completo de objetos programados que cubren las necesidades fundamentales del negocio, incluyendo funciones, vistas, procedimientos almacenados y triggers.
- La estructura modular de los objetos permite una gestión eficiente del inventario, ventas y compras y además permite una escalabilidad de la base de datos en caso de ser necesario.
- La normalización implementada asegura la integridad de los datos y minimiza la redundancia.
- Las consultas y procedimientos fueron diseñados considerando las mejores prácticas de optimización, lo que resulta en un rendimiento eficiente del sistema.

# Recomendaciones

- Revisar periódicamente el rendimiento de las consultas más utilizadas
- Actualizar los procedimientos según las necesidades cambiantes del negocio
- Establecer un sistema de monitoreo para el rendimiento de las consultas y procedimientos.
- Considerar la implementación de:
  - Sistema de alertas automáticas para niveles bajos de inventario
  - Módulo de reportes avanzados para análisis de tendencias
  - Integración con sistemas de punto de venta
  - Funcionalidades de predicción de demanda
- Implementar un sistema robusto de roles y permisos para controlar el acceso a la información sensible.
- Realizar auditorías periódicas de seguridad y acceso a los datos.
- Mantener un registro de cambios en datos críticos mediante triggers adicionales.

# Fuentes de información

## Microsoft Learn: Recursos educativos de SQL Server

Microsoft. (n.d.). *Recursos educativos de SQL Server*. Recuperado de <https://learn.microsoft.com/es-es/sql/sql-server/educational-sql-resources?view=sql-server-ver16>

Descripción: Página oficial de Microsoft que recopila materiales educativos sobre SQL Server, incluyendo tutoriales, laboratorios prácticos y enlaces a cursos relacionados.

## W3Schools: Tutorial de SQL

W3Schools. (n.d.). *SQL Tutorial*. Recuperado de <https://www.w3schools.com/sql/default.asp>

Descripción: Una guía introductoria sobre SQL que abarca conceptos básicos y avanzados, con ejemplos prácticos y ejercicios interactivos para principiantes.

## Recursos adicionales sugeridos

### SQL Server Central

Redgate. (n.d.). *SQL Server Central*. Recuperado de <https://www.sqlservercentral.com>

Descripción: Una comunidad en línea para profesionales de SQL Server, que ofrece artículos, foros, desafíos diarios, y recursos educativos relacionados con la administración y desarrollo en SQL Server.

## Repositorio de Github del proyecto

[https://github.com/isvallrod/ProyectoBasesDeDatos\\_Nov24](https://github.com/isvallrod/ProyectoBasesDeDatos_Nov24)