

# **INSTITUTO TECNOLÓGICO DEL PUTUMAYO**

**Bases de datos y almacenamiento masivo**

**Informe base de datos Sql**

**Ingry Nathaly Silva**

**Septiembre 2024**

## TABLA DE CONTENIDOS

<b>Bases de datos y almacenamiento masivo .....</b>	<b>1</b>
<b>Resumen Ejecutivo.....</b>	<b>3</b>
<b>Introducción.....</b>	<b>4</b>
<i>Contexto y Motivación.....</i>	<i>4</i>
<i>Alcance del Informe.....</i>	<i>4</i>
<i>Objetivos.....</i>	<i>4</i>
<b>Metodología.....</b>	<b>5</b>
<i>Herramientas Utilizadas .....</i>	<i>5</i>
<i>Procedimientos.....</i>	<i>5</i>
<b>Desarrollo del Informe.....</b>	<b>8</b>
<i>Descripción de la Base de Datos .....</i>	<i>8</i>
<i>Consultas SQL.....</i>	<i>9</i>
<i>Diseño de Base de Datos .....</i>	<i>13</i>
<b>Análisis y Discusión .....</b>	<b>15</b>
<i>Interpretación de Resultados .....</i>	<i>15</i>
<b>Conclusión.....</b>	<b>16</b>
<b>Recomendaciones .....</b>	<b>17</b>
<b>Referencias .....</b>	<b>18</b>

## **Resumen Ejecutivo**

Este informe proporciona un análisis detallado de la base de datos "Tienda", que se utiliza para gestionar la información de clientes, productos, pedidos, y detalles de pedidos en un entorno de comercio electrónico. Se han realizado varias consultas SQL para extraer, analizar y modificar los datos, proporcionando una visión integral del funcionamiento de la tienda y la gestión de inventario.

## **Introducción**

### **Contexto y Motivación**

El informe se realiza para comprender la estructura y funcionamiento de una base de datos de tienda en línea. La base de datos es fundamental para gestionar el inventario, los pedidos y la información de los clientes, permitiendo un análisis detallado de las operaciones comerciales y facilitando la toma de decisiones basada en datos.

### **Alcance del Informe**

El informe cubre aspectos esenciales de SQL, incluyendo la creación de tablas, relaciones entre tablas, ejecución de consultas básicas y complejas, y consideraciones de diseño de la base de datos. No se abordan temas de optimización avanzada ni gestión de bases de datos en entornos de producción.

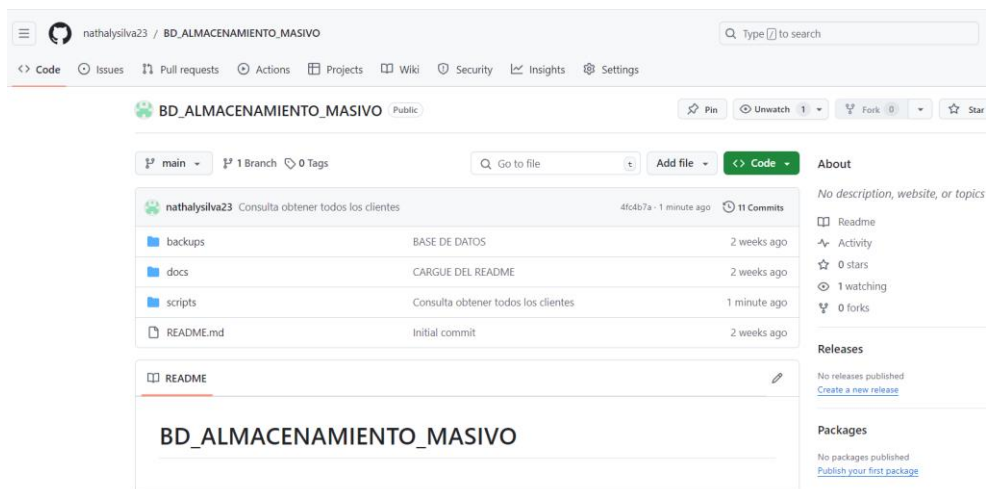
### **Objetivos**

- Analizar la estructura de la base de datos "Tienda".
- Ejecutar y explicar consultas SQL relevantes.
- Evaluar el diseño de la base de datos y proponer mejoras.

## Metodología

### Herramientas Utilizadas

- GitHub: Plataforma de desarrollo colaborativo basada en la web que utiliza el sistema de control de versiones Git, donde se subió el repositorio con la base de datos e información por medio de commits.



- MySQL Workbench: Para la modelación y ejecución de consultas SQL.
- SQLyog: Para la gestión y administración de la base de datos.

### Procedimientos

- Creación del Esquema de la Base de Datos: Se creó la base de datos y las tablas utilizando scripts SQL, definiendo las relaciones y restricciones.

#### *Creación de la base de datos:*

```
4 • /*!40111 SET @OLD_SQL_NOTES=@SQL_NOTES, SQL_NOTES=0 */;
5 • CREATE DATABASE /*!32312 IF NOT EXISTS*/`tienda` /*!4010
5
7 • USE `tienda`;
```

#### *Creación de las tablas de la base de datos (DDL):*

```

/*Table structure for table `clientes` */

• DROP TABLE IF EXISTS `clientes`;

• CREATE TABLE `clientes` (
  `id_cliente` int(11) NOT NULL AUTO_INCREMENT,
  `nombre` varchar(100) NOT NULL,
  `email` varchar(100) NOT NULL,
  `telefono` varchar(15) DEFAULT NULL,
  PRIMARY KEY (`id_cliente`),
  UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

/*Table structure for table `detallepedidos` */

• DROP TABLE IF EXISTS `detallepedidos`;

• CREATE TABLE `detallepedidos` (
  `id_detalle` int(11) NOT NULL AUTO_INCREMENT,
  `id_pedido` int(11) DEFAULT NULL,
  `id_producto` int(11) DEFAULT NULL,
  `cantidad` int(11) NOT NULL,
  `precio_unitario` decimal(10,2) NOT NULL,
  PRIMARY KEY (`id_detalle`),
  KEY `id_pedido` (`id_pedido`),
  KEY `id_producto` (`id_producto`),
  CONSTRAINT `detallepedidos_ibfk_1` FOREIGN KEY (`id_pedido`) REFERENCES `pedidos` (`id_pedido`),
  CONSTRAINT `detallepedidos_ibfk_2` FOREIGN KEY (`id_producto`) REFERENCES `productos` (`id_producto`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

/*Table structure for table `pedidos` */

• DROP TABLE IF EXISTS `pedidos`;

• CREATE TABLE `pedidos` (
  `id_pedido` int(11) NOT NULL AUTO_INCREMENT,
  `id_cliente` int(11) DEFAULT NULL,
  `fecha` date NOT NULL,
  `total` decimal(10,2) NOT NULL,
  PRIMARY KEY (`id_pedido`),
  KEY `id_cliente` (`id_cliente`),
  CONSTRAINT `pedidos_ibfk_1` FOREIGN KEY (`id_cliente`) REFERENCES `clientes` (`id_cliente`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

/*Table structure for table `productos` */

• DROP TABLE IF EXISTS `productos`;

• CREATE TABLE `productos` (
  `id_producto` int(11) NOT NULL AUTO_INCREMENT,
  `nombre` varchar(100) NOT NULL,
  `descripcion` text DEFAULT NULL,
  `precio` decimal(10,2) NOT NULL,
  `stock` int(11) NOT NULL,
  PRIMARY KEY (`id_producto`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

```

- Inserción de Datos: Se insertaron datos de ejemplo en las tablas para permitir la ejecución de consultas y análisis.

### *Inserción de datos en las tablas (DML):*

```
/*Data for the table `clientes` */
```

```
insert into `clientes`(`id_cliente`,`nombre`,`email`,`telefono`) values  
(1,'Juan Pérez','juan.perez@example.com','123456789'),  
(2,'Maria García','maria.garcia@example.com','987654321');
```

```
/*Data for the table `detallepedidos` */
```

```
insert into `detallepedidos`(`id_detalle`,`id_pedido`,`id_producto`,`cantidad`,`precio_unitario`) values  
(1,1,1,1,1200.00),  
(2,1,2,1,20.00),  
(3,2,3,1,50.00);
```

```
/*Data for the table `pedidos` */
```

```
insert into `pedidos`(`id_pedido`,`id_cliente`,`fecha`,`total`) values  
(1,1,'2024-09-01',1220.00),  
(2,2,'2024-09-01',50.00);
```

```
/*Data for the table `productos` */
```

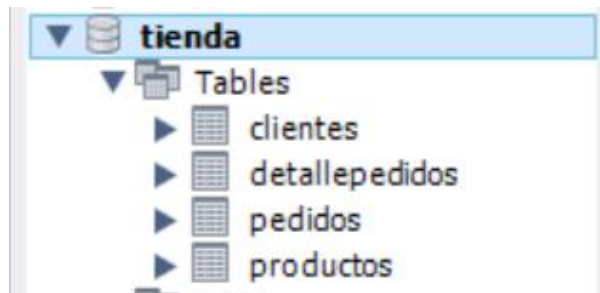
```
insert into `productos`(`id_producto`,`nombre`,`descripcion`,`precio`,`stock`) values  
(1,'Laptop','Laptop de alta gama',1200.00,10),  
(2,'Mouse','Mouse inalámbrico',20.00,100),  
(3,'Teclado','Teclado mecánico',50.00,50);
```

- Ejecución de Consultas SQL: Se realizaron consultas SQL para extraer, analizar y entender los datos almacenados.
- Análisis de Resultados: Se analizaron los resultados obtenidos de las consultas para proporcionar insights y conclusiones.

## Desarrollo del Informe

### Descripción de la Base de Datos

#### Esquema de la Base de Datos:



- Tablas:
  - **clientes:** Almacena datos de los clientes (id\_cliente, nombre, email, telefono).

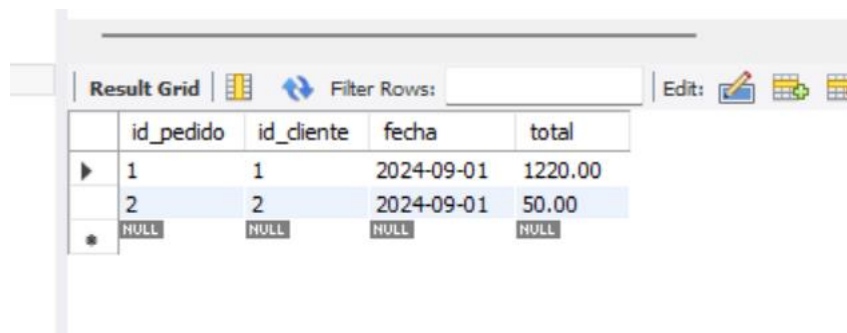
Result Grid				
	id_cliente	nombre	email	telefono
▶	1	Juan Pérez	juan.perez@example.com	123456789
	2	Maria García	maria.garcia@example.com	987654321
•	NULL	NULL	NULL	NULL

- **productos:** Contiene los productos disponibles (id\_producto, nombre, descripcion, precio, stock).

Result Grid					
	id_producto	nombre	descripcion	precio	stock
▶	1	Laptop	Laptop de alta gama	1200.00	10
	2	Mouse	Mouse inalámbrico	20.00	100
	3	Teclado	Teclado mecánico	50.00	50
•	NULL	NULL	NULL	NULL	NULL

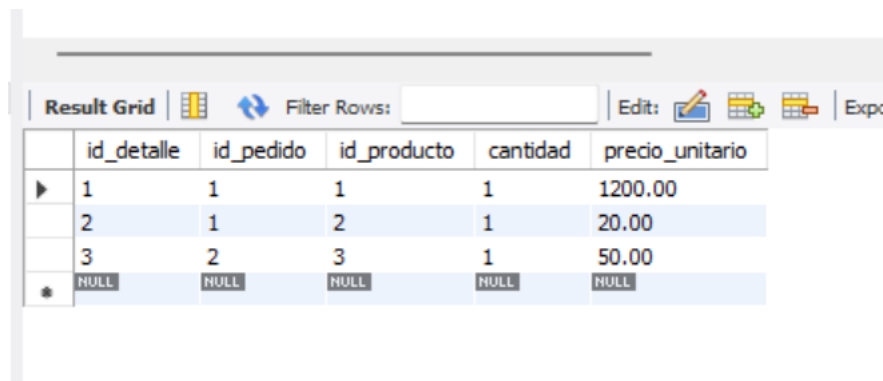
- **pedidos:** Registra los pedidos de los clientes (id\_pedido, id\_cliente, fecha, total).





	id_pedido	id_cliente	fecha	total
▶	1	1	2024-09-01	1220.00
	2	2	2024-09-01	50.00
*	NULL	NULL	NULL	NULL

- **detallepedidos:** Detalles de los productos en cada pedido (id\_detalle, id\_pedido, id\_producto, cantidad, precio\_unitario).



	id_detalle	id_pedido	id_producto	cantidad	precio_unitario
▶	1	1	1	1	1200.00
	2	1	2	1	20.00
	3	2	3	1	50.00
*	NULL	NULL	NULL	NULL	NULL

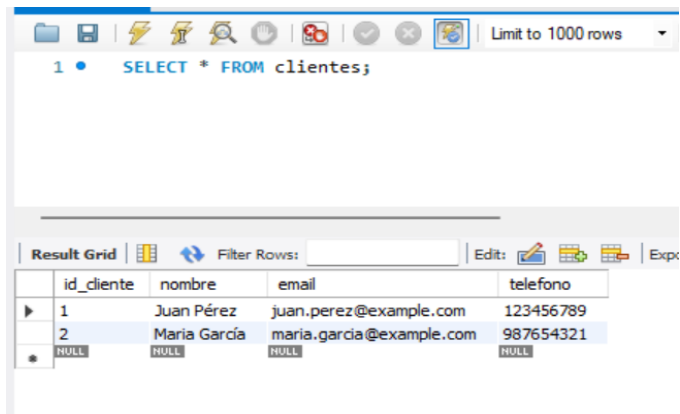
- Relaciones:
  - pedidos.id\_cliente → clientes.id\_cliente
  - detallepedidos.id\_pedido → pedidos.id\_pedido
  - detallepedidos.id\_producto → productos.id\_producto
- **Claves Principales:** id\_cliente, id\_producto, id\_pedido, id\_detalle
- **Claves Foráneas:** Relacionan las tablas detallepedidos y pedidos con clientes y productos.

## Consultas SQL

### Consultas Realizadas

- Obtener todos los clientes:

```
SELECT * FROM clientes;
```



SQL File 4\* x

Limit to 1000 rows

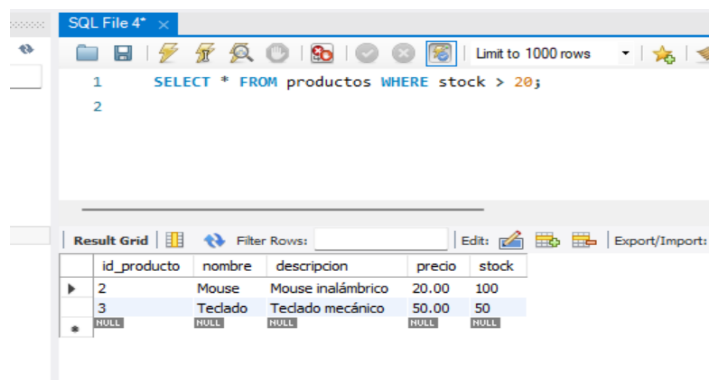
1 • SELECT \* FROM clientes;

Result Grid | Filter Rows: | Edit: | Export/Import:

	id_cliente	nombre	email	telefono
▶	1	Juan Pérez	juan.perez@example.com	123456789
▶	2	Maria García	maria.garcia@example.com	987654321
*	NULL	NULL	NULL	NULL

- Listar los productos con stock mayor a 20:

```
SELECT * FROM productos WHERE stock > 20;
```



SQL File 4\* x

Limit to 1000 rows

1 SELECT \* FROM productos WHERE stock > 20;  
2

Result Grid | Filter Rows: | Edit: | Export/Import:

	id_producto	nombre	descripcion	precio	stock
▶	2	Mouse	Mouse inalámbrico	20.00	100
▶	3	Teclado	Teclado mecánico	50.00	50
*	NULL	NULL	NULL	NULL	NULL

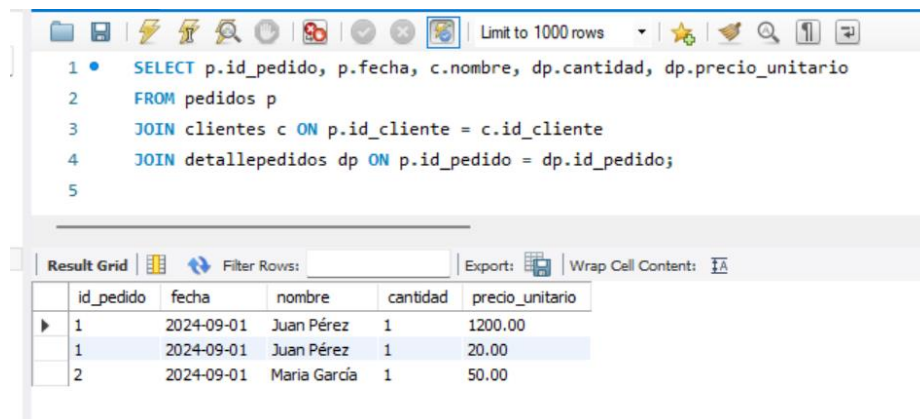
- Obtener pedidos con sus detalles:

```
SELECT p.id_pedido, p.fecha, c.nombre, dp.cantidad, dp.precio_unitario
```

```
FROM pedidos p
```

```
JOIN clientes c ON p.id_cliente = c.id_cliente
```

```
JOIN detallepedidos dp ON p.id_pedido = dp.id_pedido;
```



The screenshot shows a SQL IDE window with a query editor and a result grid. The query is a JOIN statement that combines the 'pedidos' table with 'clientes' and 'detallepedidos' tables. The result grid displays three rows of data.

```

1 • SELECT p.id_pedido, p.fecha, c.nombre, dp.cantidad, dp.precio_unitario
2 FROM pedidos p
3 JOIN clientes c ON p.id_cliente = c.id_cliente
4 JOIN detallepedidos dp ON p.id_pedido = dp.id_pedido;
5

```

	id_pedido	fecha	nombre	cantidad	precio_unitario
▶	1	2024-09-01	Juan Pérez	1	1200.00
	1	2024-09-01	Juan Pérez	1	20.00
	2	2024-09-01	Maria García	1	50.00

- Consulta con Join para mostrar pedidos y detalles de pedidos:

SELECT p.id\_pedido, c.nombre AS cliente, pr.nombre AS producto, dp.cantidad,

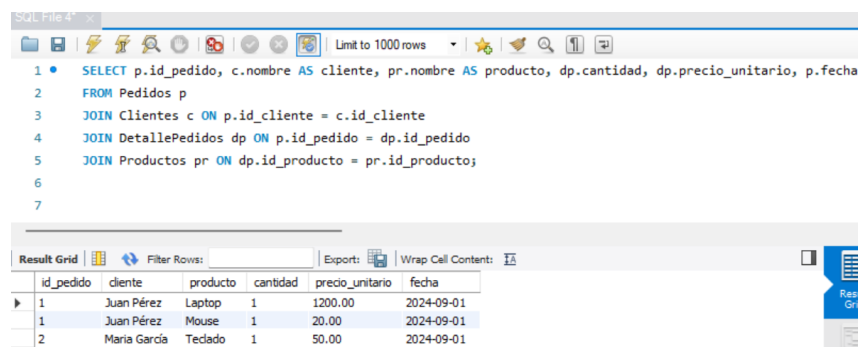
dp.precio\_unitario, p.fecha

FROM Pedidos p

JOIN Clientes c ON p.id\_cliente = c.id\_cliente

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

JOIN Productos pr ON dp.id\_producto = pr.id\_producto;



The screenshot shows a SQL IDE window with a query and its results. The query is a JOIN statement that combines the 'Pedidos' table with 'Clientes', 'DetallePedidos', and 'Productos' tables. The result grid displays three rows of data.

```

1 • SELECT p.id_pedido, c.nombre AS cliente, pr.nombre AS producto, dp.cantidad, dp.precio_unitario, p.fecha
2 FROM Pedidos p
3 JOIN Clientes c ON p.id_cliente = c.id_cliente
4 JOIN DetallePedidos dp ON p.id_pedido = dp.id_pedido
5 JOIN Productos pr ON dp.id_producto = pr.id_producto;
6
7

```

	id_pedido	cliente	producto	cantidad	precio_unitario	fecha
▶	1	Juan Pérez	Laptop	1	1200.00	2024-09-01
	1	Juan Pérez	Mouse	1	20.00	2024-09-01
	2	Maria García	Teclado	1	50.00	2024-09-01

- Actualización de stock de un producto después de una venta:

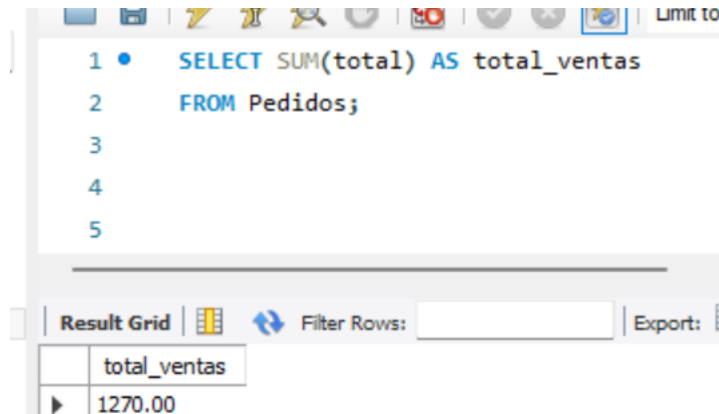
UPDATE Productos

SET stock = stock - 1

WHERE id\_producto = 1;

- Uso de funciones de agregación para ver el total de ventas:

```
SELECT SUM(total) AS total_ventas
FROM Pedidos;
```



### ***Resultados de Consultas***

- La primera consulta devuelve todos los clientes almacenados.
- La segunda muestra productos con un stock superior a 20.
- La tercera muestra una lista detallada de pedidos, incluyendo la fecha, el cliente, la cantidad y el precio unitario de cada producto en el pedido.
- La cuarta consulta devolverá un conjunto de filas en el que cada fila representa un detalle de pedido.
- La quinta consulta reducirá el valor del campo stock en la tabla productos en 1 para el producto con `id_producto = 1`. Por ejemplo, si el stock actual es 10, después de la ejecución de la consulta, el stock se actualizará a 9.
- La sexta consulta hace uso de funciones de agregación para ver el total de ventas

### ***Explicación de Consultas***

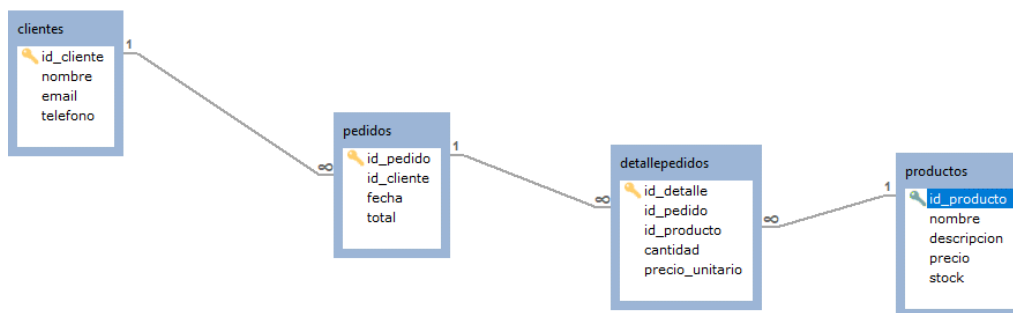
- Consulta 1: Selecciona todos los campos de la tabla clientes.
- Consulta 2: Selecciona productos de la tabla productos cuyo stock es mayor a 20.

- Consulta 3: Utiliza JOINS para combinar las tablas pedidos, clientes, y detallepedidos, permitiendo visualizar la relación entre los pedidos y los clientes.

## Diseño de Base de Datos

### *Modelo de Datos:*

- Entidad-Relación (ERD):
  - Entidades: Clientes, Productos, Pedidos, DetallePedidos.
  - Relaciones: Un cliente puede tener múltiples pedidos (1), un pedido puede tener múltiples detalles (1), y un detalle está asociado con un solo producto (N:1).



- Normalización: La base de datos se encuentra en la tercera forma normal (3NF) ya que cada tabla tiene una clave primaria única, los campos no clave dependen únicamente de la clave primaria, y no existen dependencias transitivas.

### *Consideraciones de Diseño:*

- Claves Primarias: Elegidas como identificadores únicos (id\_cliente, id\_producto, etc.).

- Relaciones: Definidas para mantener la integridad referencial, usando claves foráneas para conectar las tablas.
- Restricciones: Uso de UNIQUE en el campo email de la tabla clientes para evitar duplicados.

## **Análisis y Discusión**

### **Interpretación de Resultados**

Las consultas realizadas demuestran cómo la base de datos puede utilizarse para obtener información esencial para la operación de la tienda, como el seguimiento de pedidos y la gestión del inventario. La actualización automática del stock después de una venta es clave para mantener la precisión en la disponibilidad de productos.

## **Conclusión**

La base de datos "Tienda" se diseñó de manera efectiva para cumplir con los requisitos de un entorno de comercio electrónico básico. La correcta definición de las tablas y sus relaciones permite una gestión eficiente de la información de clientes, productos y pedidos. Las consultas SQL realizadas demuestran la capacidad de la base de datos para proporcionar insights valiosos sobre las operaciones comerciales.



### **Recomendaciones**

- Optimización: Implementar índices adicionales para mejorar el rendimiento de las consultas, especialmente en tablas con un gran volumen de datos.
- Seguridad: Incluir mecanismos de autenticación y control de acceso para proteger los datos sensibles.
- Escalabilidad: Evaluar la arquitectura de la base de datos para soportar un aumento en la cantidad de datos y usuarios.
- Implementar más consultas para análisis avanzados, como seguimiento de ventas y comportamiento del cliente.
- Automatizar la actualización del stock para manejar múltiples ventas simultáneamente.

## Referencias

- Repositorio con la información de la base de datos:  
[https://github.com/nathalysilva23/BD\\_ALMACENAMIENTO\\_MASIVO.git](https://github.com/nathalysilva23/BD_ALMACENAMIENTO_MASIVO.git)
- Documentación de MySQL: <https://dev.mysql.com/doc/>
- Documentación de SqlYog: <https://sqlyogkb.webyog.com/collection/1-sqlyog-docs>