Tutorial Completo de Bases de Datos y SQL: De Cero a Experto

ADAPTACION Y COMPLEMENTACION DE PROGRAMAS. PROFE: AGUSTIN GATICA

PARTE 1: TEORÍA FUNDAMENTAL DE BASES DE DATOS

1.1 ¿Qué es una Base de Datos?

Una **Base de Datos (BD)** es un conjunto organizado de datos relacionados que se almacenan y se pueden recuperar de manera eficiente. Es como un archivo digital muy potente que permite:

- **Almacenar** grandes volúmenes de información
- **Organizar** datos de forma estructurada
- **Recuperar** información rápidamente
- **Actualizar** datos fácilmente
- **Proteger** la información

Ejemplo Real: Una base de datos de una tienda contiene información de productos, clientes, pedidos, etc.

1.2 Conceptos Clave

Tabla

Es la estructura principal donde se organizan los datos en filas y columnas.

ID_Cliente	Nombre	Email	Ciudad
1	Juan	juan@email.com	Madrid
2	María	maria@email.com	Barcelona
3	Pedro	pedro@email.com	Valencia

Registro (Fila)

Cada fila representa un conjunto completo de datos sobre un tema (ej: un cliente específico)

Campo (Columna)

Cada columna representa un tipo específico de información (ej: Nombre, Email)

Clave Primaria (Primary Key)

Un campo único que identifica cada registro. En el ejemplo anterior: **ID_Cliente**

Clave Foránea (Foreign Key)

Un campo que vincula una tabla con otra. Crea relaciones entre tablas.

1.3 Tipos de Bases de Datos

1. Bases de Datos Relacionales (SQL)

- Datos organizados en tablas
- Usa SQL (Lenguaje de Consulta Estructurado)
- **Ejemplos:** MySQL, PostgreSQL, SQL Server, Microsoft Access, Oracle
- Mejor para datos estructurados

2. Bases de Datos NoSQL

- Datos en formatos flexibles (documentos, grafos, etc.)
- **Ejemplos:** MongoDB, Firebase, Cassandra
- Mejor para datos no estructurados

*Para este tutorial nos enfocamos en **Bases Relacionales***

1.4 Relaciones entre Tablas

Relación Uno a Muchos (1:M)

Un cliente puede tener muchos pedidos

```
CLIENTES (1) ──────→ (M) PEDIDOS
ID_Cliente           ID_Pedido, ID_Cliente
```

Relación Muchos a Muchos (M:M)

Un estudiante puede inscribirse en muchos cursos, un curso tiene muchos estudiantes

```
ESTUDIANTES (M) ──────→ (M) CURSOS
              ↓ Tabla Intermedia ↓
           INSCRIPCIONES
```

1.5 Reglas de Normalización (Normas de Diseño)

Primera Forma Normal (1FN):

- Cada campo contiene un único valor (no listas)
- No hay campos repetidos

Segunda Forma Normal (2FN):

- Cumple 1FN
- Todos los campos no clave dependen completamente de la clave primaria

Tercera Forma Normal (3FN):

- Cumple 2FN
- Elimina dependencias transitivas entre campos

Esto evita datos duplicados y mantiene la integridad

PARTE 2: INTRODUCCIÓN A SQL

2.1 ¿Qué es SQL?

SQL (Structured Query Language) es el lenguaje estándar para:

- Consultar bases de datos
- Insertar datos
- Actualizar datos
- Eliminar datos
- Crear tablas

2.2 Tipos de Comandos SQL

Tipo	Comandos	Función
DDL (Definición)	CREATE, ALTER, DROP	Crear/modificar estructura
DML (Manipulación)	SELECT, INSERT, UPDATE, DELETE	Modificar datos

Tipo	Comandos	Función
DCL (Control)	GRANT, REVOKE	Control de acceso

PARTE 3: PRÁCTICA CON MICROSOFT ACCESS

3.1 Descarga e Instalación

Microsoft Access viene incluido en Microsoft Office 365 o puedes comprarlo por separado.

3.2 Crear Primera Base de Datos en Access

Paso 1: Abre Microsoft Access

Paso 2: Selecciona "Base de datos en blanco"

Paso 3: Nombra tu BD: `MiPrimeraDB` y haz clic en "Crear"

Paso 4: Access abre con una tabla en blanco

3.3 Crear Tabla de Clientes

En la cinta "Crear":

1. Haz clic en **"Tabla"** (se abre con campos por defecto)
2. Personaliza los campos:

Campo	Tipo Datos	Propiedades
ID_Cliente	Numeración automática	Clave primaria
Nombre	Texto corto	Requerido
Email	Texto corto	-
Teléfono	Texto corto	-
Ciudad	Texto corto	-
Fecha_Registro	Fecha/Hora	-

Para cambiar nombre y tipo:

- Haz clic derecho en el encabezado del campo
- Selecciona "**Propiedades de Campo**"

- Configura según la tabla anterior

3.4 Insertar Datos de Ejemplo

Modo Hoja de Datos:

ID_Cliente	Nombre	Email	Teléfono	Ciudad	Fecha_Registro
1	Juan García	juan@email.com	912345678	Madrid	2026-01-15
2	María López	maria@email.com	933456789	Barcelona	2026-01-20
3	Pedro Rodríguez	pedro@email.com	961234567	Valencia	2026-01-25
4	Ana Martínez	ana@email.com	945678901	Sevilla	2026-02-01
5	Carlos Fernández	carlos@email.com	918765432	Bilbao	2026-02-10

3.5 Crear Tabla de Pedidos

Campo	Tipo	Propiedades
ID_Pedido	Numeración automática	Clave primaria
ID_Cliente	Número	Clave foránea
Fecha_Pedido	Fecha/Hora	-
Total	Moneda	-
Estado	Texto corto	-

Datos:

ID_Pedido	ID_Cliente	Fecha_Pedido	Total	Estado
101	1	2026-02-01	150.50	Entregado
102	2	2026-02-03	200.00	Pendiente
103	1	2026-02-05	75.25	Entregado
104	3	2026-02-06	320.00	En proceso
105	2	2026-02-08	95.75	Entregado

ID_Pedido	ID_Cliente	Fecha_Pedido	Total	Estado
106	4	2026-02-10	450.00	Pendiente

PARTE 4: CONSULTAS SQL BÁSICAS EN ACCESS

4.1 Crear tu Primera Consulta

Paso 1: Ve a la pestaña **"Crear"**

Paso 2: Haz clic en **"Consulta de diseño"**

Paso 3: Se abre un diálogo, selecciona la tabla **"Clientes"** y haz clic en **"Agregar"**

Paso 4: Verás el diseño de consulta (Pasa a Vista SQL)

4.2 Cambiar a Vista SQL

En la cinta, haz clic en la flecha junto a **"Vista"** → **"Vista SQL"**

Ahora escribiremos código SQL directamente.

PARTE 5: CONSULTAS SQL PASO A PASO

5.1 Consulta SELECT - Lo Más Importante

$$\text{SELECT}$$ es el comando fundamental. Te permite recuperar datos.

Sintaxis Básica:

```
SELECT campo1, campo2, ...
FROM nombre_tabla;
```

Consulta 1: Obtener todos los clientes

```
SELECT *
FROM Clientes;
```

- `*` = todos los campos
- Resultado: Muestra todos los clientes con toda su información

Consulta 2: Solo nombres y emails

```
SELECT Nombre, Email
FROM Clientes;
```

Resultado:

Nombre	Email
Juan García	juan@email.com
María López	maria@email.com

5.2 WHERE - Filtrar Datos

Filtra registros según condiciones.

Sintaxis:

```
SELECT campos
FROM tabla
WHERE condición;
```

Consulta 3: Clientes de Madrid

```
SELECT Nombre, Ciudad
FROM Clientes
WHERE Ciudad = 'Madrid';
```

Resultado:

Nombre	Ciudad
Juan García	Madrid

Consulta 4: Pedidos mayores a 100€

```
SELECT ID_Pedido, Total
FROM Pedidos
WHERE Total > 100;
```

5.3 Operadores de Comparación

Operador	Significado	Ejemplo
`=`	Igual	`Ciudad = 'Madrid'`
`!=` o `<>`	Diferente	`Estado <> 'Entregado'`
`>`	Mayor que	`Total > 100`
`<`	Menor que	`Total < 50`
`>=`	Mayor o igual	`Total >= 150`
`<=`	Menor o igual	`Total <= 200`

5.4 AND, OR, NOT - Múltiples Condiciones

AND - Se cumplen TODAS las condiciones

```
SELECT Nombre, Ciudad, Email
FROM Clientes
WHERE Ciudad = 'Madrid' AND Nombre = 'Juan García';
```

OR - Se cumple AL MENOS UNA condición

```
SELECT Nombre, Ciudad
FROM Clientes
WHERE Ciudad = 'Madrid' OR Ciudad = 'Barcelona';
```

Resultado: Clientes de Madrid O Barcelona

NOT - Negación

```
SELECT Nombre, Ciudad
FROM Clientes
WHERE NOT Ciudad = 'Madrid';
```

Equivalente a: `WHERE Ciudad <> 'Madrid'`

5.5 ORDER BY - Ordenar Resultados

Sintaxis:

```
SELECT campos
FROM tabla
ORDER BY campo ASC|DESC;
```

- `ASC` = Ascendente (A→Z, 1→9) [Por defecto]
- `DESC` = Descendente (Z→A, 9→1)

Consulta 5: Clientes ordenados por nombre

```
SELECT Nombre, Ciudad
FROM Clientes
ORDER BY Nombre ASC;
```

Resultado:

Nombre
Ana Martínez
Carlos Fernández
Juan García
María López
Pedro Rodríguez

Consulta 6: Pedidos por mayor monto

```
SELECT ID_Pedido, Total, Estado
FROM Pedidos
ORDER BY Total DESC;
```

5.6 DISTINCT - Valores Únicos

Elimina duplicados.

Consulta 7: ¿Cuántas ciudades diferentes tenemos?

```
SELECT DISTINCT Ciudad
FROM Clientes;
```

Resultado:

Ciudad
Madrid
Barcelona
Valencia
Sevilla
Bilbao

5.7 LIMIT - Limitar Resultados

Consulta 8: Los 3 primeros clientes

```
SELECT TOP 3 Nombre, Email
FROM Clientes;
```

En Access usamos TOP en lugar de LIMIT

5.8 LIKE - Búsqueda de Patrones

Busca texto parcial.

Sintaxis:

```
WHERE campo LIKE 'patrón'
```

- `%` = Cualquier cantidad de caracteres
- `_` = Un único carácter

Consulta 9: Clientes cuyo nombre empieza con 'M'

```
SELECT Nombre, Email
FROM Clientes
WHERE Nombre LIKE 'M%';
```

Resultado: María López

Consulta 10: Emails que contienen 'email'

```sql
SELECT Nombre, Email
FROM Clientes
WHERE Email LIKE '%email%';
```

5.9 IN - Múltiples Valores

Consulta 11: Pedidos que NO están entregados

```sql
SELECT ID_Pedido, Estado, Total
FROM Pedidos
WHERE Estado IN ('Pendiente', 'En proceso');
```

Equivalente a:

```sql
WHERE Estado = 'Pendiente' OR Estado = 'En proceso'
```

5.10 BETWEEN - Rango de Valores

Consulta 12: Pedidos entre 100 y 300 euros

```sql
SELECT ID_Pedido, Total
FROM Pedidos
WHERE Total BETWEEN 100 AND 300;
```

PARTE 6: FUNCIONES DE AGREGACIÓN

Estas funciones calculan valores sobre grupos de datos.

6.1 COUNT - Contar Registros

Consulta 13: ¿Cuántos clientes tenemos?

```sql
SELECT COUNT(*) AS Total_Clientes
FROM Clientes;
```

Resultado: 5

Consulta 14: ¿Cuántos pedidos están entregados?

```sql
SELECT COUNT(*) AS Pedidos_Entregados
FROM Pedidos
WHERE Estado = 'Entregado';
```

6.2 SUM - Suma

Consulta 15: Total de ventas

```sql
SELECT SUM(Total) AS Ventas_Totales
FROM Pedidos;
```

Resultado: 1,291.50

Consulta 16: Total de ventas entregadas

```sql
SELECT SUM(Total) AS Ventas_Entregadas
FROM Pedidos
WHERE Estado = 'Entregado';
```

6.3 AVG - Promedio

Consulta 17: Valor promedio de pedidos

```sql
SELECT AVG(Total) AS Promedio_Pedido
FROM Pedidos;
```

Resultado: 215.25

6.4 MAX y MIN - Máximo y Mínimo

Consulta 18: Pedido más caro y más barato

```sql
SELECT
  MAX(Total) AS Pedido_Maximo,
  MIN(Total) AS Pedido_Minimo
FROM Pedidos;
```

Resultado:

Pedido_Maximo	Pedido_Minimo
450.00	75.25

PARTE 7: GROUP BY - Agrupar Datos

Agrupa registros por un campo y aplica funciones.

Sintaxis:

```
SELECT campo, FUNCION(campo2)
FROM tabla
GROUP BY campo;
```

Consulta 19: Cantidad de pedidos por cliente

```
SELECT ID_Cliente, COUNT(*) AS Total_Pedidos
FROM Pedidos
GROUP BY ID_Cliente;
```

Resultado:

ID_Cliente	Total_Pedidos
1	2
2	2
3	1
4	1

Consulta 20: Total gastado por cada cliente

```
SELECT ID_Cliente, SUM(Total) AS Gasto_Total
FROM Pedidos
GROUP BY ID_Cliente
ORDER BY Gasto_Total DESC;
```

PARTE 8: HAVING - Filtrar Grupos

Filtra grupos (como WHERE pero para GROUP BY).

Consulta 21: Clientes con más de 1 pedido

```
SELECT ID_Cliente, COUNT(*) AS Total_Pedidos
FROM Pedidos
GROUP BY ID_Cliente
HAVING COUNT(*) > 1;
```

Resultado: Solo clientes 1 y 2 (tienen 2 pedidos cada uno)

PARTE 9: JOINS - Combinar Tablas

Una de las partes más importantes. Permite conectar datos de múltiples tablas.

9.1 INNER JOIN

Devuelve solo registros que coinciden en ambas tablas.

Sintaxis:

```
SELECT campos
FROM tabla1
INNER JOIN tabla2
ON tabla1.clave = tabla2.clave;
```

Consulta 22: Clientes y sus pedidos

```
SELECT
  c.Nombre,
  c.Email,
  p.ID_Pedido,
  p.Total,
  p.Estado
FROM Clientes c
INNER JOIN Pedidos p
ON c.ID_Cliente = p.ID_Cliente;
```

Resultado:

Nombre	Email	ID_Pedido	Total	Estado
Juan García	juan@email.com	101	150.50	Entregado
Juan García	juan@email.com	103	75.25	Entregado
María López	maria@email.com	102	200.00	Pendiente
María López	maria@email.com	105	95.75	Entregado
Pedro Rodríguez	pedro@email.com	104	320.00	En proceso
Ana Martínez	ana@email.com	106	450.00	Pendiente

9.2 LEFT JOIN

Devuelve todos los registros de la tabla izquierda + coincidencias de la derecha.

Consulta 23: Todos los clientes y sus pedidos (incluso sin pedidos)

```
SELECT
  c.Nombre,
  COUNT(p.ID_Pedido) AS Total_Pedidos
FROM Clientes c
LEFT JOIN Pedidos p
ON c.ID_Cliente = p.ID_Cliente
GROUP BY c.Nombre;
```

9.3 RIGHT JOIN

Devuelve todos los registros de la tabla derecha + coincidencias de la izquierda.

```
SELECT
  c.Nombre,
  p.ID_Pedido
FROM Clientes c
RIGHT JOIN Pedidos p
ON c.ID_Cliente = p.ID_Cliente;
```

PARTE 10: CONSULTAS AVANZADAS

10.1 Subconsultas

Una consulta dentro de otra.

Consulta 24: Clientes que han gastado más que el promedio

```
SELECT Nombre, Email
FROM Clientes c
WHERE c.ID_Cliente IN (
  SELECT ID_Cliente
  FROM Pedidos
  GROUP BY ID_Cliente
  HAVING SUM(Total) > (
    SELECT AVG(Total_Gasto)
    FROM (
      SELECT ID_Cliente, SUM(Total) AS Total_Gasto
      FROM Pedidos
      GROUP BY ID_Cliente
    )
  )
);
```

10.2 UNION - Combinar Resultados

Combina resultados de múltiples SELECT.

```
SELECT Nombre AS Persona, 'Cliente' AS Tipo
FROM Clientes
UNION
SELECT Nombre AS Persona, 'Empleado' AS Tipo
FROM Empleados;
```

PARTE 11: CREAR BASES DE DATOS Y TABLAS CON SQL

11.1 CREATE DATABASE - Crear Base de Datos

En **phpMyAdmin**, necesitarás código diferente. Veamos primero Access.

En Microsoft Access:

Access crea automáticamente la BD cuando creas una nueva. Para SQL puro:

```
-- En MySQL/phpMyAdmin:
CREATE DATABASE MiTienda;
USE MiTienda;
```

11.2 CREATE TABLE - Crear Tabla

Sintaxis:

```
CREATE TABLE nombre_tabla (
  campo1 tipo_dato propiedades,
  campo2 tipo_dato propiedades,
  ...
  PRIMARY KEY (campo_clave)
);
```

Ejemplo: Crear tabla de Categorías

```
CREATE TABLE Categorias (
  ID_Categoria INT AUTO_INCREMENT,
  Nombre_Categoria VARCHAR(50) NOT NULL,
  Descripcion VARCHAR(200),
  PRIMARY KEY (ID_Categoria)
);
```

Tipos de datos comunes:

- `INT` = Número entero
- `DECIMAL(10,2)` = Decimal (precio)
- `VARCHAR(50)` = Texto variable (máx 50)
- `TEXT` = Texto largo
- `DATE` = Fecha
- `DATETIME` = Fecha y hora
- `BOOLEAN` = Verdadero/Falso

Propiedades:

- `NOT NULL` = Campo obligatorio
- `AUTO_INCREMENT` = Se incrementa automáticamente
- `DEFAULT valor` = Valor por defecto
- `UNIQUE` = Valores únicos

11.3 Crear Tabla con Clave Foránea

```
CREATE TABLE Productos (
  ID_Producto INT AUTO_INCREMENT,
  Nombre_Producto VARCHAR(100) NOT NULL,
  Precio DECIMAL(10,2) NOT NULL,
```

```
  ID_Categoria INT NOT NULL,
  Stock INT DEFAULT 0,
  PRIMARY KEY (ID_Producto),
  FOREIGN KEY (ID_Categoria) REFERENCES Categorias(ID_Categoria)
);
```

11.4 ALTER TABLE - Modificar Tabla

```
-- Agregar campo
ALTER TABLE Productos
ADD COLUMN Fecha_Creacion DATE DEFAULT CURDATE();

-- Modificar campo
ALTER TABLE Productos
MODIFY COLUMN Precio DECIMAL(12,2);

-- Eliminar campo
ALTER TABLE Productos
DROP COLUMN Fecha_Creacion;
```

11.5 INSERT - Insertar Datos

```
INSERT INTO Categorias (Nombre_Categoria, Descripcion)
VALUES ('Electrónica', 'Productos electrónicos');

INSERT INTO Categorias (Nombre_Categoria, Descripcion)
VALUES
  ('Ropa', 'Prendas de vestir'),
  ('Libros', 'Libros diversos'),
  ('Alimentos', 'Productos alimenticios');
```

11.6 UPDATE - Actualizar Datos

```
-- Cambiar precio de un producto
UPDATE Productos
SET Precio = 99.99
WHERE ID_Producto = 1;

-- Aumentar precio 10%
UPDATE Productos
SET Precio = Precio * 1.10
WHERE ID_Categoria = 1;
```

11.7 DELETE - Eliminar Datos

```sql
-- Eliminar producto específico
DELETE FROM Productos
WHERE ID_Producto = 5;

-- Eliminar todos los productos sin stock
DELETE FROM Productos
WHERE Stock = 0;
```

11.8 DROP TABLE - Eliminar Tabla Completa

```sql
DROP TABLE Productos;
```

⚠️ **CUIDADO:** Esto elimina la tabla y todos sus datos.

PARTE 12: TRABAJAR CON phpMyAdmin

12.1 Acceso a phpMyAdmin

phpMyAdmin es una interfaz web para gestionar MySQL/MariaDB.

Ubicación típica: `http://localhost/phpmyadmin`

12.2 Crear Base de Datos en phpMyAdmin

Paso 1: Haz clic en "**Bases de datos**"

Paso 2: Ingresa el nombre: `tienda_online`

Paso 3: Selecciona colación: `utf8mb4_unicode_ci`

Paso 4: Haz clic en "**Crear**"

12.3 Crear Tabla en phpMyAdmin

Paso 1: Selecciona tu BD

Paso 2: Haz clic en "**Crear tabla**"

Paso 3: Ingresa nombre y cantidad de columnas

Campo	Tipo	Longitud	Atributos	Índice
id_cliente	INT	-	AUTO_INCREMENT	PRIMARY
nombre	VARCHAR	100	NOT NULL	-
email	VARCHAR	100	UNIQUE	-
fecha_registro	DATETIME	-	DEFAULT CURRENT_TIMESTAMP	-

12.4 Insertar Datos en phpMyAdmin

Método 1: Interfaz gráfica

- Haz clic en la tabla
- Pestaña "**Insertar**"
- Completa los campos
- "**Continuar**"

Método 2: SQL directo

- Pestaña "**SQL**"
- Pega tu código INSERT

```
INSERT INTO Clientes (nombre, email, fecha_registro)
VALUES
  ('Juan García', 'juan@email.com', NOW()),
  ('María López', 'maria@email.com', NOW());
```

12.5 Ejecutar Consultas en phpMyAdmin

Paso 1: Haz clic en la tabla

Paso 2: Pestaña "**SQL**"

Paso 3: Escribe tu consulta:

```
SELECT nombre, email, fecha_registro
FROM Clientes
WHERE YEAR(fecha_registro) = 2026
ORDER BY fecha_registro DESC;
```

Paso 4: Botón "**Continuar**" o Ctrl+Enter

PARTE 13: PROYECTO PRÁCTICO COMPLETO

13.1 Descripción del Proyecto

Crearemos un sistema de **Gestión de Tienda Online** con:

- Clientes
- Categorías de productos
- Productos
- Pedidos
- Detalles de pedidos

13.2 Script SQL Completo

```
-- Crear la base de datos
CREATE DATABASE tienda_online;
USE tienda_online;

-- Tabla de Categorías
CREATE TABLE Categorias (
  id_categoria INT AUTO_INCREMENT PRIMARY KEY,
  nombre_categoria VARCHAR(100) NOT NULL,
  descripcion TEXT,
  fecha_creacion DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Tabla de Clientes
CREATE TABLE Clientes (
  id_cliente INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL,
  telefono VARCHAR(15),
  ciudad VARCHAR(50),
  fecha_registro DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Tabla de Productos
CREATE TABLE Productos (
  id_producto INT AUTO_INCREMENT PRIMARY KEY,
  nombre_producto VARCHAR(150) NOT NULL,
  descripcion TEXT,
  precio DECIMAL(10,2) NOT NULL,
  stock INT DEFAULT 0,
  id_categoria INT NOT NULL,
  fecha_creacion DATETIME DEFAULT CURRENT_TIMESTAMP,
```

```sql
  FOREIGN KEY (id_categoria) REFERENCES Categorias(id_categoria)
);

-- Tabla de Pedidos
CREATE TABLE Pedidos (
  id_pedido INT AUTO_INCREMENT PRIMARY KEY,
  id_cliente INT NOT NULL,
  fecha_pedido DATETIME DEFAULT CURRENT_TIMESTAMP,
  total DECIMAL(10,2) NOT NULL,
  estado ENUM('Pendiente', 'En proceso', 'Enviado', 'Entregado', 'Cancelado'),
  FOREIGN KEY (id_cliente) REFERENCES Clientes(id_cliente)
);

-- Tabla de Detalles de Pedidos (Tabla de relación M:M)
CREATE TABLE Detalles_Pedido (
  id_detalle INT AUTO_INCREMENT PRIMARY KEY,
  id_pedido INT NOT NULL,
  id_producto INT NOT NULL,
  cantidad INT NOT NULL,
  precio_unitario DECIMAL(10,2) NOT NULL,
  subtotal DECIMAL(10,2) GENERATED ALWAYS AS (cantidad * precio_unitario) STORED,
  FOREIGN KEY (id_pedido) REFERENCES Pedidos(id_pedido),
  FOREIGN KEY (id_producto) REFERENCES Productos(id_producto)
);

-- Insertar datos de ejemplo
INSERT INTO Categorias (nombre_categoria, descripcion) VALUES
('Electrónica', 'Dispositivos y aparatos electrónicos'),
('Ropa', 'Prendas de vestir'),
('Libros', 'Libros y publicaciones'),
('Alimentos', 'Productos alimenticios');

INSERT INTO Clientes (nombre, email, telefono, ciudad) VALUES
('Juan García', 'juan@email.com', '912345678', 'Madrid'),
('María López', 'maria@email.com', '933456789', 'Barcelona'),
('Pedro Rodríguez', 'pedro@email.com', '961234567', 'Valencia'),
('Ana Martínez', 'ana@email.com', '945678901', 'Sevilla');

INSERT INTO Productos (nombre_producto, descripcion, precio, stock, id_categoria) VALUES
('Laptop Dell XPS 13', 'Laptop ultradelgada de 13 pulgadas', 999.99, 10, 1),
('Mouse Logitech MX', 'Mouse inalámbrico de precisión', 99.99, 50, 1),
('Camiseta Algodon', 'Camiseta 100% algodón', 29.99, 200, 2),
('Python para Principiantes', 'Libro de programación', 39.99, 30, 3),
('Café Premium 1kg', 'Café molido de excelente calidad', 15.99, 100, 4);

INSERT INTO Pedidos (id_cliente, total, estado) VALUES
(1, 1129.97, 'Entregado'),
(2, 99.99, 'Pendiente'),
(3, 1059.98, 'En proceso'),
(4, 89.96, 'Entregado');

INSERT INTO Detalles_Pedido (id_pedido, id_producto, cantidad, precio_unitario) VALUES
(1, 1, 1, 999.99),
```

```
(1, 2, 1, 99.99),
(1, 3, 1, 29.99),
(2, 2, 1, 99.99),
(3, 1, 1, 999.99),
(3, 2, 1, 99.99),
(3, 4, 1, 39.99),
(4, 3, 2, 29.99),
(4, 5, 1, 15.99);
```

13.3 Consultas Útiles para la Tienda

Consulta 1: Ver todos los pedidos de un cliente con detalles

```
SELECT
  c.nombre,
  p.id_pedido,
  p.fecha_pedido,
  prod.nombre_producto,
  dp.cantidad,
  dp.precio_unitario,
  dp.subtotal,
  p.total,
  p.estado
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
INNER JOIN Productos prod ON dp.id_producto = prod.id_producto
WHERE c.nombre = 'Juan García'
ORDER BY p.fecha_pedido DESC;
```

Consulta 2: Productos más vendidos

```
SELECT
  prod.nombre_producto,
  cat.nombre_categoria,
  SUM(dp.cantidad) AS Total_Vendido,
  SUM(dp.subtotal) AS Ingresos_Totales
FROM Productos prod
INNER JOIN Detalles_Pedido dp ON prod.id_producto = dp.id_producto
INNER JOIN Categorias cat ON prod.id_categoria = cat.id_categoria
GROUP BY prod.id_producto
ORDER BY Total_Vendido DESC;
```

Consulta 3: Ingresos por categoría

```
SELECT
  cat.nombre_categoria,
```

```
  COUNT(DISTINCT p.id_pedido) AS Total_Pedidos,
  SUM(dp.subtotal) AS Ingresos,
  AVG(dp.subtotal) AS Promedio_Venta
FROM Categorias cat
INNER JOIN Productos prod ON cat.id_categoria = prod.id_categoria
INNER JOIN Detalles_Pedido dp ON prod.id_producto = dp.id_producto
INNER JOIN Pedidos p ON dp.id_pedido = p.id_pedido
GROUP BY cat.id_categoria
ORDER BY Ingresos DESC;
```

Consulta 4: Clientes más valiosos

```
SELECT
  c.nombre,
  c.email,
  COUNT(p.id_pedido) AS Total_Pedidos,
  SUM(p.total) AS Gasto_Total,
  AVG(p.total) AS Promedio_Pedido
FROM Clientes c
INNER JOIN Pedidos p ON c.id_cliente = p.id_cliente
GROUP BY c.id_cliente
ORDER BY Gasto_Total DESC;
```

RESUMEN: DIFERENCIAS ACCESS vs phpMyAdmin

Aspecto	Microsoft Access	phpMyAdmin
Interfaz	Escritorio	Web
Base datos	Archivo .accdb	Servidor MySQL
Escalabilidad	Hasta ~2GB	Ilimitada
Multiusuario	Limitado	Excelente
SQL	Variante de SQL	MySQL estándar
Costo	Pago (Office 365)	Gratuito
Ideal para	Pequeños proyectos	Producción/Web