

Tienda Inventario

Esta es una aplicación web para la gestión de inventario de componentes de ordenadores. Soporta bases de datos PostgreSQL y MySQL/MariaDB usando PHP con PDO.



Creación de una base de datos llamada **Tienda** con tablas para componentes de hardware (**almacenamiento, cpu, gpu, memoria**)

✓ 1. Diseño Relacional Mejorado

Actualmente tienes las 4 tablas principales, pero para un diseño más **escalable y mantenible**, podrías agregar relaciones y estructurar la BD así:

♦ Tablas propuestas:

- **producto**: tabla común que une todos los componentes (almacenamiento, cpu, etc.)
- **categoría**: define si es CPU, GPU, etc.
- **marca**: fabricantes (Intel, AMD, Samsung, etc.)
- **almacenamiento / cpu / gpu / memoria**: detalles técnicos específicos

🎯 **Ventaja**: separas lo común (precio, stock, nombre) de lo específico (núcleos, velocidad, etc.).

🔧 2. Comandos SQL de creación de tablas, insert y relaciones

Aquí va un ejemplo usando PostgreSQL o MySQL:

```
-- 1. Tabla de categorías

CREATE TABLE categoria (

    id_categoria SERIAL PRIMARY KEY,

    nombre VARCHAR(50) UNIQUE NOT NULL

);

-- 2. Tabla de marcas

CREATE TABLE marca (
```

```
id_marca SERIAL PRIMARY KEY,

nombre VARCHAR(50) UNIQUE NOT NULL

);

-- 3. Tabla de productos

CREATE TABLE producto (

    id_producto SERIAL PRIMARY KEY,

    nombre VARCHAR(100) NOT NULL,

    precio DECIMAL(10,2) NOT NULL,

    stock INT NOT NULL DEFAULT 0,

    id_categoria INT REFERENCES categoria(id_categoria),

    id_marca INT REFERENCES marca(id_marca)

);

-- 4. Tabla CPU

CREATE TABLE cpu (

    id_cpu SERIAL PRIMARY KEY,

    id_producto INT UNIQUE REFERENCES producto(id_producto),

    nucleos INT,

    hilos INT,

    frecuencia_base DECIMAL(4,2),

    frecuencia_turbo DECIMAL(4,2),

    passmark_score INT

);
```

```
-- 5. Tabla GPU

CREATE TABLE gpu (

    id_gpu SERIAL PRIMARY KEY,

    id_producto INT UNIQUE REFERENCES producto(id_producto),

    vram_gb INT,

    passmark_score INT

);


-- 6. Tabla Memoria

CREATE TABLE memoria (

    id_memoria SERIAL PRIMARY KEY,

    id_producto INT UNIQUE REFERENCES producto(id_producto),

    capacidad_gb INT,

    tipo VARCHAR(10),

    velocidad_mhz INT

);


-- 7. Tabla Almacenamiento

CREATE TABLE almacenamiento (

    id_almacenamiento SERIAL PRIMARY KEY,

    id_producto INT UNIQUE REFERENCES producto(id_producto),

    capacidad_gb INT,

    tipo VARCHAR(10)

);
```

```
-- Inserciones para la tabla categoria
```

```
INSERT INTO categoria (nombre) VALUES
```

```
('CPU'),  
('GPU'),  
('Memoria'),  
('Almacenamiento'),  
('Placa Madre'),  
('Fuente de Poder'),  
('Refrigeración'),  
('Gabinete'),  
('Monitor'),  
('Teclado'),  
('Ratón'),  
('Auriculares'),  
('Tarjeta de Sonido'),  
('Tarjeta de Red'),  
('Unidad Óptica'),  
('Ventilador'),  
('Disipador'),  
('Cableado'),  
('Adaptadores'),  
('Impresora');
```

```
-- Inserciones para la tabla marca
```

```
INSERT INTO marca (nombre) VALUES
```

```
('Intel'),  
('AMD'),  
('NVIDIA'),  
('ASUS'),  
('MSI'),  
('Gigabyte'),  
('Corsair'),  
('Kingston'),  
('G.Skill'),  
('Crucial'),  
('Samsung'),  
('Western Digital'),  
('Seagate'),  
('EVGA'),
```

```

('Cooler Master'),
('NZXT'),
('Logitech'),
('Razer'),
('HP'),
('Dell');

-- Inserciones para la tabla producto
INSERT INTO producto (nombre, precio, stock, id_categoria, id_marca)
VALUES
('Intel Core i5-12400F', 180.00, 15, 1, 1),
('AMD Ryzen 5 5600X', 199.99, 10, 1, 2),
('NVIDIA GeForce RTX 3060', 329.00, 8, 2, 3),
('ASUS TUF Gaming RTX 4070', 599.00, 5, 2, 4),
('Corsair Vengeance LPX 16GB DDR4', 75.00, 20, 3, 7),
('G.Skill Ripjaws V 32GB DDR4', 120.00, 12, 3, 9),
('Samsung 970 EVO Plus 1TB SSD', 99.00, 18, 4, 11),
('Seagate Barracuda 2TB HDD', 55.00, 25, 4, 13),
('MSI B550 Tomahawk', 139.00, 7, 5, 5),
('Gigabyte X670 AORUS Elite', 229.00, 6, 5, 6),
('Corsair RM750x PSU 750W', 119.00, 10, 6, 7),
('Cooler Master MWE 650W', 89.00, 9, 6, 15),
('Cooler Master Hyper 212', 44.00, 14, 7, 15),
('NZXT Kraken X63', 149.99, 4, 7, 16),
('NZXT H510 ATX Case', 79.00, 11, 8, 16),
('Logitech G Pro X Headset', 99.00, 6, 12, 17),
('Razer BlackWidow V3', 129.00, 5, 10, 18),
('Logitech G502 HERO Mouse', 49.99, 10, 11, 17),
('HP 27" Full HD Monitor', 179.99, 7, 9, 19),
('Dell UltraSharp 24" IPS', 229.99, 4, 9, 20);

INSERT INTO cpu (id_producto, nucleos, hilos, frecuencia_base,
frecuencia_turbo, passmark_score) VALUES
(1, 6, 12, 2.50, 4.40, 17000), -- Intel Core i5-12400F
(2, 6, 12, 3.70, 4.60, 19500); -- AMD Ryzen 5 5600X

INSERT INTO gpu (id_producto, vram_gb, passmark_score) VALUES
(3, 12, 17000), -- RTX 3060
(4, 12, 22000); -- RTX 4070

```

```
INSERT INTO memoria (id_producto, capacidad_gb, tipo, velocidad_mhz)
VALUES
(5, 16, 'DDR4', 3200),    -- Corsair Vengeance LPX 16GB DDR4
(6, 32, 'DDR4', 3600);   -- G.Skill Ripjaws V 32GB DDR4

INSERT INTO almacenamiento (id_producto, capacidad_gb, tipo) VALUES
(7, 1000, 'SSD'),        -- Samsung 970 EVO Plus 1TB SSD
(8, 2000, 'HDD');        -- Seagate Barracuda 2TB HDD
```



3. Consultas básicas

Consultar stock y precios:

```
SELECT p.nombre, c.nombre AS categoria, m.nombre AS marca, p.precio,
p.stock
FROM producto p
JOIN categoria c ON p.id_categoria = c.id_categoria
JOIN marca m ON p.id_marca = m.id_marca;
```



4. Pantalla de ingreso y consulta de inventario

Si estás usando **HTML + JS (frontend)** con **PHP, Python Flask/Django o Node.js (backend)**, una interfaz sencilla puede tener:

- **Formulario de ingreso de producto:**
 - Categoría (dropdown)
 - Marca (dropdown)
 - Nombre, precio, stock
 - Campos específicos (frecuencia para CPU, vram para GPU, etc.)
- **Tabla de consulta:**
 - Filtros por categoría y marca

- Búsqueda por nombre
- Ordenar por stock o precio

Objetivo

Tener una página web con:

1. **Formulario de ingreso** de productos, incluyendo campos generales y específicos (como vRAM, frecuencia, etc.).
2. **Tabla de consulta** de inventario, con filtros por categoría y marca, y orden por precio o stock.

Estructura general del proyecto

C:\Users\fanny\Desktop\tienda_proyecto_php>tree /f en CMD

```
TIENDA_PROYECTO_PHP
| categorias.php
| conectar.php
| conectar_mysql.php
| conexion_pg.php
| exportar_historial_csv.php
| index.html
| index.php
| info.php
| ingresar.php
| insertar_producto.php
| insert_datatienda_inventario.sql
| inventario.php
| inventario_schema_create_mysql.sql
| marcas.php
| obtener_categorias.php
| obtener_marcas.php
| obtener_productos.php
| productos.php
| script.js
| tienda_inventario.sql
|
|——css
|   estilo.css
|
```


└─img
imagen.jpg

✓ Paso 1: Base de datos (PostgreSQL)

✓ Paso 2: Backend en PHP

- **backend/conectar.php (conexión básica)**

conexion_pg.php

```
<?php
// Parámetros de conexión
$host = "localhost";
$port = "5432";
$dbname = "mi_base_de_datos";
$user = "mi_usuario";
$password = "mi_contraseña";

// Cadena de conexión
$conn_string = "host=$host port=$port dbname=$dbname user=$user
password=$password";

// Intentar conexión
$conn = pg_connect($conn_string);

// Verificar si se conectó correctamente
if ($conn) {
    echo "Conexión exitosa a PostgreSQL.";
} else {
    echo "Error al conectar a PostgreSQL.";
}
?>
```

conectar.php

```
<?php
$host = "localhost";
$dbname = "Tienda_Inventario";
$user = "postgres";
```

```

$pass = ""; // déjalo vacío si no tienes contraseña

try {
    $pdo = new PDO("pgsql:host=$host;dbname=$dbname", $user, $pass);
} catch (PDOException $e) {
    echo "✗ Error de conexión: " . $e->getMessage();
    exit;
}
?>

```

ingresar.php

```

<?php
require 'conectar.php'; // Asegúrate de que este archivo tiene tu
conexión a PostgreSQL

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Recoger datos del formulario
    $nombre = $_POST['nombre'];
    $precio = $_POST['precio'];
    $stock = $_POST['stock'];
    $id_categoria = $_POST['id_categoria'];
    $id_marca = $_POST['id_marca'];

    try {
        // Preparar la consulta SQL
        $sql = "INSERT INTO producto (nombre, precio, stock,
id_categoria, id_marca)
            VALUES (:nombre, :precio, :stock, :id_categoria,
:id_marca)";

        $stmt = $pdo->prepare($sql);
        $stmt->execute([
            ':nombre' => $nombre,
            ':precio' => $precio,
            ':stock' => $stock,
            ':id_categoria' => $id_categoria,
            ':id_marca' => $id_marca
        ]);

        echo "Producto guardado exitosamente. <a
href='index.php'>Volver</a>";
    } catch (PDOException $e) {

```

```

        echo "Error al insertar el producto: " . $e->getMessage();
    }
} else {
    echo "Acceso no permitido.";
}
?>

```

Para confirmar, abre en tu navegador colocar :

http://localhost:8000

Verifica en la base de datos

Puedes comprobar que el producto fue insertado con éxito ejecutando:

sql

SELECT * FROM producto;

Inventario.php

```

<?php
require 'conectar.php'; // Conexión a PostgreSQL
?>

<!DOCTYPE html>
<html>
<head>
    <title>Inventario de Productos</title>
</head>
<body>
    <h1>Inventario</h1>
    <a href="index.php">← Volver al formulario</a>
    <br><br>

    <table border="1" cellpadding="5" cellspacing="0">
        <thead>
            <tr>
                <th>Nombre</th>
                <th>Precio</th>

```

```

        <th>Stock</th>
        <th>Categoría</th>
        <th>Marca</th>
    </tr>
</thead>
<tbody>
<?php
    try {
        $stmt = $pdo->query("SELECT p.nombre, p.precio, p.stock,
c.nombre AS categoria, m.nombre AS marca
                                FROM producto p
                                JOIN categoria c ON p.id_categoria =
c.id_categoria
                                JOIN marca m ON p.id_marca =
m.id_marca
                                ORDER BY p.nombre");

        while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
            echo "<tr>
                <td>{$row['nombre']}</td>
                <td>{$row['precio']}</td>
                <td>{$row['stock']}</td>
                <td>{$row['categoria']}</td>
                <td>{$row['marca']}</td>
            </tr>";
        }
    } catch (PDOException $e) {
        echo "<tr><td colspan='5'>Error al obtener datos: " .
$e->getMessage() . "</td></tr>";
    }
?>
</tbody>
</table>
</body>
</html>

```

obtener_categorias.php

```

<?php
include 'conectar.php';

try {

```

```

        $stmt = $conn->query("SELECT id, nombre FROM categorias ORDER BY
nombre ASC");
        $categorias = $stmt->fetchAll(PDO::FETCH_ASSOC);
        echo json_encode($categorias);
    } catch (PDOException $e) {
        echo json_encode(['error' => $e->getMessage()]);
    }
?>

```

obtener_marcas.php

```

<?php
include 'conectar.php';

try {
    $stmt = $conn->query("SELECT id, nombre FROM marcas ORDER BY nombre
ASC");
    $marcas = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($marcas);
} catch (PDOException $e) {
    echo json_encode(['error' => $e->getMessage()]);
}
?>

```

✓ Paso 3: Frontend en HTML + JS

index.html

html

```

<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Inventario Tienda</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        form { margin-bottom: 30px; }
        table { border-collapse: collapse; width: 100%; }
        th, td { border: 1px solid #ccc; padding: 8px; text-align:
left; }
        th { background-color: #f4f4f4; }
    </style>
</head>

```

```
<body>

  <h1>Ingreso de Producto</h1>
  <form id="formProducto">
    <label>Nombre: <input type="text" name="nombre"
required></label><br>
    <label>Precio: <input type="number" name="precio" step="0.01"
required></label><br>
    <label>Stock: <input type="number" name="
```

script.js

javascript

```
// script.js
document.addEventListener('DOMContentLoaded', () => {
  const form = document.getElementById('formProducto');
  const tabla = document.querySelector('#tablaInventario tbody');

  form.addEventListener('submit', async (e) => {
    e.preventDefault();

    const datos = new FormData(form);
    const response = await fetch('insertar_producto.php', {
      method: 'POST',
      body: datos
    });

    const resultado = await response.text();
    alert(resultado);
    form.reset();
    cargarProductos();
  });

  async function cargarProductos() {
    const response = await fetch('obtener_productos.php');
    const productos = await response.json();

    tabla.innerHTML = '';
    productos.forEach(p => {
      const fila = `<tr>
```

```

                <td>${p.nombre}</td>
                <td>${p.categoria}</td>
                <td>${p.marca}</td>
                <td>${p.precio}</td>
                <td>${p.stock}</td>
            </tr>`;
            tabla.innerHTML += fila;
        });
    }

    cargarProductos(); // al inicio
});

```

Cargar categorías y marcas al cargar **index.html**

En tu **script.js**, al cargar la página, debes hacer llamadas AJAX/fetch a **obtener_categorias.php** y **obtener_marcas.php** para llenar los **<select>** del formulario.

marcas.php

```

<?php

require 'conexion_pg.php'; // O 'conectar.php' si usas ese archivo con
PDO

?>

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <title>Lista de Marcas</title>

```

```
<style>

    table {

        border-collapse: collapse;

        width: 50%;

        margin-top: 20px;

    }

    th, td {

        border: 1px solid #999;

        padding: 8px;

        text-align: left;

    }

    th {

        background-color: #eee;

    }

</style>

</head>

<body>

    <h1>Marcas registradas</h1>

    <a href="index.php">← Volver al formulario</a>

    <table>

        <thead>

            <tr>
```



```

        <th>ID</th>

        <th>Nombre de la marca</th>

    </tr>

</thead>

<tbody>

    <?php

        try {

            $stmt = $pdo->query("SELECT id_marca, nombre FROM marca
ORDER BY id_marca");

            while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {

                echo "<tr>

                    <td>{$row['id_marca']}</td>

                    <td>{$row['nombre']}</td>

                </tr>";

            }

        } catch (PDOException $e) {

            echo "<tr><td colspan='2'>Error al obtener datos: " .
$e->getMessage() . "</td></tr>";

        }

    ?>

</tbody>

</table>

</body>

</html>

```

categorias.php

```
<?php

require 'conectar.php'; // Asegúrate de que este archivo define $pdo
correctamente con PDO y PostgreSQL

?>

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <title>Lista de Categorías</title>

    <style>

        table {

            border-collapse: collapse;

            width: 50%;

            margin-top: 20px;

        }

        th, td {

            border: 1px solid #999;

            padding: 8px;

            text-align: left;

        }

    </style>

</head>

<body>

    <div>

        <table>

            <thead>

                <tr>

                    <th>Categoría</th>

                </tr>

            </thead>

            <tbody>

                <tr>

                    <td>Categoría 1</td>

                </tr>

                <tr>

                    <td>Categoría 2</td>

                </tr>

                <tr>

                    <td>Categoría 3</td>

                </tr>

                <tr>

                    <td>Categoría 4</td>

                </tr>

                <tr>

                    <td>Categoría 5</td>

                </tr>

            </tbody>

        </table>

    </div>

</body>

</html>
```

```
        th {

            background-color: #eee;

        }

    </style>

</head>

<body>

    <h1>Categorías registradas</h1>

    <a href="index.php">← Volver al formulario</a>

    <table>

        <thead>

            <tr>

                <th>ID</th>

                <th>Nombre de la categoría</th>

            </tr>

        </thead>

        <tbody>

            <?php

                try {

                    $stmt = $pdo->query("SELECT id_categoria, nombre FROM
categoria ORDER BY id_categoria");

                    while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {

                        echo "<tr>

                            <td>{$row['id_categoria']}</td>
```

```

        <td>{$row['nombre']}</td>

        </tr>";

    }

    } catch (PDOException $e) {

        echo "<tr><td colspan='2'>Error: " . $e->getMessage() .
"</td></tr>";

    }

    ?>

</tbody>

</table>

</body>

</html>

```

Modificar la estructura con:

1. Agregar claves foráneas a la tabla **producto**

```

ALTER TABLE producto

ADD CONSTRAINT fk_categoria

FOREIGN KEY (id_categoria) REFERENCES categoria(id_categoria) ON DELETE
SET NULL;

ALTER TABLE producto

ADD CONSTRAINT fk_marca

```

```
FOREIGN KEY (id_marca) REFERENCES marca(id_marca) ON DELETE SET NULL;
```

✓ 2. Crear trigger para registrar cambios de stock

Supongamos que quieres registrar los cambios en la tabla **log_cambios_stock**.

a) Función trigger para registrar cambios de stock

```
CREATE OR REPLACE FUNCTION registrar_cambio_stock()

RETURNS TRIGGER AS $$

BEGIN

    IF (NEW.stock IS DISTINCT FROM OLD.stock) THEN

        INSERT INTO log_cambios_stock(id_producto, stock_anterior,
stock_nuevo, fecha)

        VALUES (OLD.id_producto, OLD.stock, NEW.stock, NOW());

    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;
```

b) Trigger que llama a la función al actualizar **producto**

```
CREATE TRIGGER trg_cambio_stock
AFTER UPDATE ON producto
FOR EACH ROW
WHEN (OLD.stock IS DISTINCT FROM NEW.stock)
EXECUTE FUNCTION registrar_cambio_stock();
```

✓ 3. Crear una vista importante: **vista_productos_detalle**

Una vista que combine producto con su categoría, marca, y stock actual:

```
CREATE OR REPLACE VIEW vista_productos_detalle AS

SELECT

    p.id_producto,

    p.nombre,

    p.precio,

    p.stock,

    c.nombre AS categoria,

    m.nombre AS marca

FROM

    producto p

LEFT JOIN categoria c ON p.id_categoria = c.id_categoria

LEFT JOIN marca m ON p.id_marca = m.id_marca;
```

Esta vista sirve para:

- Mostrar información en el frontend sin hacer múltiples joins.
- Consultas en dashboards o reportes.

inventario-php

```
<?php

require 'conectar.php'; // Conexión a PostgreSQL

?>
```

```
<!DOCTYPE html>

<html>

<head>

    <title>Inventario de Productos</title>

</head>

<body>

    <h1>Inventario</h1>

    <a href="index.php">← Volver al formulario</a>

    <br><br>

    <table border="1" cellpadding="5" cellspacing="0">

        <thead>

            <tr>

                <th>Nombre</th>

                <th>Precio</th>

                <th>Stock</th>

                <th>Categoría</th>

                <th>Marca</th>

            </tr>

        </thead>

        <tbody>

            <?php

            try {

                $stmt = $pdo->query("SELECT p.nombre, p.precio, p.stock,
c.nombre AS categoria, m.nombre AS marca
```

```

        FROM producto p

        JOIN categoria c ON p.id_categoria =
c.id_categoria

        JOIN marca m ON p.id_marca =
m.id_marca

        ORDER BY p.nombre");

while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {

    echo "<tr>

        <td>{$row['nombre']}

```




Observaciones

- El HTML tiene un formulario con selects de categorías y marcas.
- Al cargar la página, los selects se llenan desde el backend.
- El formulario al enviarse, inserta el producto vía `insertar_producto.php`.
- Después de insertar, recarga la tabla llamando a `obtener_productos.php` y muestra los productos.
- Todo funciona con fetch + JSON y PHP + PostgreSQL.



Tabla de historial de precios o stock

Esta tabla registra cada vez que cambia el precio o el stock de un producto, para tener un histórico.

sql

```
CREATE TABLE historial_precios_stock (  
    id SERIAL PRIMARY KEY,  
    producto_id INT NOT NULL,  
    fecha_cambio TIMESTAMP DEFAULT NOW(),  
    precio_anterior NUMERIC(10,2),  
    precio_nuevo NUMERIC(10,2),  
    stock_anterior INT,  
    stock_nuevo INT  
    id_categoria INT,  
    id_marca INT  
);
```

```
INSERT INTO historial_precios_stock  
(producto_id, fecha_cambio, precio_anterior, precio_nuevo,  
stock_anterior, stock_nuevo, id_categoria, id_marca)  
VALUES  
(1, '2025-06-01 10:00:00', 150.00, 140.00, 30, 28, 2, 1),  
(2, '2025-06-02 14:30:00', 220.00, 215.00, 20, 18, 1, 2),
```

```
(3, '2025-06-03 09:45:00', 320.00, 310.00, 15, 15, 3, 3),
(1, '2025-06-04 16:20:00', 140.00, 135.00, 28, 25, 2, 1),
(4, '2025-06-05 11:15:00', 400.00, 395.00, 10, 9, 4, 4),
(2, '2025-06-06 13:50:00', 215.00, 210.00, 18, 16, 1, 2),
(5, '2025-06-07 08:30:00', 500.00, 490.00, 5, 5, 5, 5);
```



Exportar datos a CSV desde PHP

Código PHP para exportar la tabla **historial_precios_stock** a CSV

Guarda este archivo, por ejemplo, como **exportar_historial_csv.php** en tu proyecto:

```
<?php
require 'conectar.php'; // Asegúrate que aquí $pdo es la conexión PDO a
PostgreSQL

// Nombre del archivo CSV que se descargará
$filename = "historial_precios_stock_" . date('Ymd') . ".csv";

// Headers para indicar que es un archivo descargable CSV
header('Content-Type: text/csv; charset=utf-8');
header('Content-Disposition: attachment; filename=' . $filename);

// Abrir "salida" estándar para escribir el CSV
$output = fopen('php://output', 'w');

// Escribir la cabecera del CSV
fputcsv($output, array('ID', 'Producto ID', 'Fecha Cambio', 'Precio
Anterior', 'Precio Nuevo', 'Stock Anterior', 'Stock Nuevo', 'ID
Categoria', 'ID Marca'));

try {
    // Consulta para obtener los datos
    $stmt = $pdo->query("SELECT * FROM historial_precios_stock ORDER BY
fecha_cambio DESC");

    // Escribir cada fila en el CSV
    while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
        fputcsv($output, $row);
    }
} catch (PDOException $e) {
```

```

    // En caso de error, mostrar mensaje
    echo "Error al obtener datos: " . $e->getMessage();
    exit;
}




fclose($output);
exit;
?>

```

Cómo usarlo

1. Abre en el navegador:
`http://localhost:8000/exportar_historial_csv.php`
2. Se descargará automáticamente un archivo `.csv` con los datos de la tabla `historial_precios_stock`.
3. Ábrelo con Excel, LibreOffice Calc o cualquier editor de texto.

Script completo para la tabla `historial_precios_stock` que:

1.  **Agrega las columnas `id_categoria` e `id_marca`.**
2.  **Limpia datos inválidos**, es decir, pone en `NULL` los valores de `id_categoria` o `id_marca` que no tengan correspondencia en las tablas `categoria` y `marca`.
3.  **Crea claves foráneas** con integridad referencial.

SCRIPT SQL PARA `historial_precios_stock`

sql

```

-- . Actualizar id_categoria a NULL cuando no existe en
log_cambios_stock
UPDATE log_cambios_stock
SET id_categoria = NULL
WHERE id_categoria IS NOT NULL
    AND id_categoria NOT IN (SELECT id_categoria FROM categoria);

ALTER TABLE log_cambios_stock

```

```
ADD COLUMN id_categoria INT;

ALTER TABLE log_cambios_stock
ADD COLUMN id_marca INT;

-- Solo después de agregar las columnas
UPDATE log_cambios_stock
SET id_categoria = NULL
WHERE id_categoria IS NOT NULL
    AND id_categoria NOT IN (SELECT id_categoria FROM categoria);

-- . Limpiar datos inválidos en id_categoria
UPDATE log_cambios_stock
SET id_categoria = NULL
WHERE id_categoria IS NOT NULL
    AND id_categoria NOT IN (SELECT id_categoria FROM categoria);

-- . Limpiar datos inválidos en id_marca
UPDATE log_cambios_stock
SET id_marca = NULL
WHERE id_marca IS NOT NULL
    AND id_marca NOT IN (SELECT id_marca FROM marca);

-- . Agregar claves foráneas (si no existen ya)
ALTER TABLE log_cambios_stock
ADD CONSTRAINT fk_log_categoria
FOREIGN KEY (id_categoria) REFERENCES categoria(id_categoria)
ON DELETE SET NULL;

ALTER TABLE log_cambios_stock
ADD CONSTRAINT fk_log_marca
FOREIGN KEY (id_marca) REFERENCES marca(id_marca)
ON DELETE SET NULL;

-- . Limpiar datos inválidos en id_categoria
UPDATE historial_precios_stock
SET id_categoria = NULL
WHERE id_categoria IS NOT NULL
    AND id_categoria NOT IN (SELECT id_categoria FROM categoria);

-- . Limpiar datos inválidos en id_marca
UPDATE historial_precios_stock
SET id_marca = NULL
```

```

WHERE id_marca IS NOT NULL
    AND id_marca NOT IN (SELECT id_marca FROM marca);

-- . Agregar claves foráneas (si no existen)
ALTER TABLE historial_precios_stock
ADD CONSTRAINT fk_historial_categoria
FOREIGN KEY (id_categoria) REFERENCES categoria(id_categoria)
ON DELETE SET NULL;




ALTER TABLE historial_precios_stock
ADD CONSTRAINT fk_historial_marca
FOREIGN KEY (id_marca) REFERENCES marca(id_marca)
ON DELETE SET NULL;

-- . Limpiar datos inválidos en producto_id
UPDATE historial_precios_stock
SET producto_id = NULL
WHERE producto_id IS NOT NULL
    AND producto_id NOT IN (SELECT id_producto FROM producto);

-- . Agregar clave foránea
ALTER TABLE historial_precios_stock
ADD CONSTRAINT fk_historial_producto
FOREIGN KEY (producto_id)
REFERENCES producto(id_producto)
ON DELETE SET NULL;

```

SCRIPT PARA RELACIONAR **producto_id** CON **producto(id_producto)**

1.  **Verifica si existen valores inválidos.**
2.  **Limpia** los **producto_id** que no existen en la tabla **producto**, asignando **NULL**.
3.  **Agrega la clave foránea** con **ON DELETE SET NULL**.

```
-- . Limpiar datos inválidos en producto_id
UPDATE historial_precios_stock
SET producto_id = NULL
WHERE producto_id IS NOT NULL
    AND producto_id NOT IN (SELECT id_producto FROM producto);

-- . Agregar clave foránea
ALTER TABLE historial_precios_stock
ADD CONSTRAINT fk_historial_producto
FOREIGN KEY (producto_id)
REFERENCES producto(id_producto)
ON DELETE SET NULL;
```

✅ Backup con **pgAdmin** (interfaz gráfica)

1. Abrir pgAdmin.

2. En el panel izquierdo, haz clic derecho en tu base de datos **Tienda_Inventario**.

3. Selecciona **Backup...**

4. Configura así:

- **Format:** Custom o Tar
- **Filename:** Ruta donde guardar el archivo (**.backup**)
- Opciones adicionales:
 - Marcar **Data**, **Schema**, **Blobs**, **Privileges**, etc., si aparecen.

5. Haz clic en "Backup" para guardar

Carpeta ubicada en el escrito C:\Users\fanny\Desktop\Tienda_Inventario, archivo tienda.sql

✓ ¿Por qué se restaura una base de datos?

Restaurar sirve para **recuperar** todo el trabajo en estos casos:

🔄 1. Cambias de PC o reinstalas PostgreSQL

Si reinstalas o usas otra máquina, puedes restaurar tu backup y continuar trabajando sin perder nada.

💥 2. Cometes un error grave

Si borras tablas, datos, relaciones o haces un **DROP DATABASE** por error, puedes restaurar el backup para volver al punto seguro.

🐛 3. Fallas en disco o sistema

Un daño en el disco o el sistema puede borrar la base. Con el backup, puedes restaurar todo como estaba.

🧠 Cuando se usa :

El **pg_restore** solo se usa cuando tú lo necesitas, por ejemplo:

- Para migrar tu base a otro equipo.
- Para recuperar datos tras un fallo.
- Para probar algo sin miedo, restaurando luego.

📌 Pasos detallados para restaurar un **.backup**:

1. Abre CMD (símbolo del sistema) como administrador.
2. Escribe el siguiente comando

```
pg_restore -U postgres -h localhost -p 5432 -d Tienda_Inventario -v  
"C:\Users\fanny\Desktop\Tienda_Inventario\tienda.sql"
```

📌 Busca los errores del **pg_restore**

Para ver **exactamente qué falló**, ejecuta **pg_restore** redirigiendo la salida a un archivo de log:

```
pg_restore -U postgres -h localhost -p 5432 -d Tienda_Inventario -v  
"C:\Users\fanny\Desktop\Tienda_Inventario\tienda.sql" > restore_log.txt 2>&1
```

Este comando crea un archivo llamado `restore_log.txt` con todos los errores y advertencias. Abrirlo y revisar qué falló.

✓ ¿Cómo verlo?

1. Abrir el **Explorador de archivos**.
2. Navegar a: `C:\Users\fanny\`
3. Busca un archivo llamado: `restore_log.txt`
4. **Haz doble clic** para abrirlo (usa Bloc de notas o cualquier editor de texto).

Perfecto. A continuación te explico todos los pasos detalladamente para:

✓ 1. Preparar la Conexión PDO para MySQL

a. Instalar el driver PDO para MySQL en Windows

En XAMPP o WAMP ya viene instalado. Si usas PHP independiente:

- Verifica que la extensión esté habilitada en `php.ini`:

```
ini
extension=pdo_mysql
```

b. Crear nuevo archivo de conexión `conectar_mysql.php`

```
php
Copiar código
<?php
ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);

$host = 'localhost';
$dbname = 'tienda_inventario_mysql';
$user = 'root'; // o tu usuario
$password = ''; // tu contraseña
```



```
try {
    $pdo = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8",
    $user, $password, [
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
    ]);
    echo "Conexión exitosa a MySQL.";
} catch (PDOException $e) {
    echo "Error al conectar a MySQL: " . $e->getMessage();
}
?>
```

🟡 **No necesitas crear otra carpeta.** Puedes reutilizar los archivos y solo cambiar el archivo de conexión entre `conectar.php` (PostgreSQL) y `conectar_mysql.php`.

✅ 2. Exportar esquema PostgreSQL y convertir a MySQL

a. Exportar la estructura de PostgreSQL:

bash

Copiar código

```
pg_dump -U postgres -h localhost -p 5432 --schema-only -f
tienda_pg.sql Tienda_Inventario
```

b. Convertir tienda_pg.sql a MySQL

- Usa una de estas herramientas:
 - 🧠 <https://sqlines.com/online> (conversión de PostgreSQL a MySQL)
 - 🛠️ Aplicaciones como **SQL Workbench**, **Full Convert**, o **DBConvert**
- Guarda el resultado como:

pgsql

Copiar código

```
tienda_inventario_mysql.sql
```

c. Crear base de datos en MySQL

sql

Copiar código

```
CREATE DATABASE tienda_inventario_mysql CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
```

d. Importar el archivo convertido:

```
bash
Copiar código
mysql -u root -p tienda_inventario_mysql <
tienda_inventario_mysql.sql
```

✓ 3. Permitir que tu app soporte ambas BD

Puedes tener una estructura como esta:

```
bash
Copiar código
/conectar_postgres.php
/conectar_mysql.php
/config.php ← define qué conexión usar
/index.php
/inventario.php
...
```

Se hace dos con `conectar.php` podrías hacer algo así:

```
php
$pdo = new PDO("mysql:host=$host;port=$port;dbname=$dbname", $user,
$password);

$pdo = new PDO("pgsql:host=$host;dbname=$db", $user, $pass);
```

✓ OPCIÓN RECOMENDADA: SQLines Command Line – Gratis y sin licencia

🔧 PASOS PARA USAR SQLINES (VERSIÓN GRATUITA)

Puedes usar la herramienta gratuita llamada **SQLines Command Line Tool**, que convierte archivos `.sql` de PostgreSQL a MySQL sin registro.

1. Descargar la versión gratuita

- Ve a: <https://sqlines.com/download>
- Descarga: **SQLines SQL Converter - Command Line**
- No requiere instalación, solo descomprimir el **.zip**.

2. Prepara tu archivo PostgreSQL

Asegúrate de tener el archivo SQL de PostgreSQL exportado, por ejemplo:

makefile

Copiar código

`C:\Users\fanny\Desktop\Tienda_Inventario\tienda_pg.sql`

-

3. Abrir CMD en la carpeta donde está SQLines

- Descomprime el archivo **.zip** descargado.
- Entra a esa carpeta y copia la ruta.

Abre CMD y navega a esa carpeta:

bash

Copiar código

`cd "C:\ruta\donde\descomprimiste\sqlines"`

-

4. Ejecuta el comando de conversión

bash

Copiar código

`sqlines -s=postgresql -t=mysql`

`-in="C:\Users\fanny\Desktop\Tienda_Inventario\tienda_pg.sql"`

`-out="C:\Users\fanny\Desktop\Tienda_Inventario\tienda_mysql.sql"`

- `-s=postgresql`: define el origen (PostgreSQL).
- `-t=mysql`: define el destino (MySQL).

- **-in**: ruta del archivo SQL de entrada.
 - **-out**: archivo convertido que se generará.
-

✓ Resultado:

- Se creará un archivo: **tienda_mysql.sql** en tu escritorio, listo para importar en MySQL.
- Puedes usar **MySQL Workbench** o **phpMyAdmin** para importar ese archivo.

Puedes usar **phpMyAdmin** para importar el archivo **tienda_mysql.sql** a tu servidor MySQL. Aquí te explico el **paso a paso completo** para hacerlo bien y **evitar errores comunes**:

✓ REQUISITOS PREVIOS

Antes de empezar, asegúrate de:

- Tener phpMyAdmin funcionando (por ejemplo, a través de **XAMPP**, **WAMP**, **Laragon**, etc.).
- Haber creado una base de datos vacía donde se va a importar el contenido.

PASOS PARA IMPORTAR **tienda_mysql.sql** EN PHPMYADMIN

1. Iniciar tu servidor local

- Abre XAMPP/WAMP/Laragon y **asegúrate de que Apache y MySQL estén activos**.

Abre tu navegador y ve a:

arduino
<http://localhost/phpmyadmin>

2. 📁 Crear una base de datos vacía

- En el panel izquierdo, haz clic en **"Nueva"** o **"New"**.

Escribe el nombre de tu nueva base de datos, por ejemplo:

nginx
[tienda_inventario](#)

Selecciona el cotejamiento (collation):

nginx
[utf8_general_ci](#)

- Haz clic en **Crear**.

3. 📁 Importar el archivo [.sql](#)

- Una vez creada, haz clic en el nombre de la base de datos en el panel izquierdo.
- Ahora ve a la pestaña **"Importar"** en el menú superior.
- En "Archivo a importar":
 - Haz clic en **"Elegir archivo"** / **"Choose file"**.

Busca y selecciona el archivo:

makefile
[C:\Users\fanny\Desktop\Tienda_Inventario\tienda_mysql.sql](#)

-
- Asegúrate de que esté seleccionado:
 - Formato del archivo: **SQL**
 - Codificación: [utf-8](#) (por defecto está bien)

- Haz clic en "**Continuar**" / "**Go**" (parte inferior).

4. Esperar el resultado

Si todo va bien, verás el mensaje:

mathematica

Copiar código

La importación se ha ejecutado correctamente / Import has been successfully finished.

- En el panel izquierdo ahora verás todas las tablas importadas.

SI APARECEN ERRORES

Si ves errores como:

- "Error de sintaxis"
- "Table already exists"
- "Unknown data type"

Entonces:

1. Asegúrate de que la base de datos estaba **vacía** antes de importar.
2. Abre el archivo `tienda_mysql.sql` con un editor (como Notepad++) y revisa si:
 - Tiene sentencias PostgreSQL sin convertir (por ejemplo, `SERIAL`, `BOOLEAN`, `NOW()` en vez de `CURRENT_TIMESTAMP`, etc.).

Usar **HeidiSQL** con **MariaDB** perfectamente para importar tu archivo `tienda_mysql.sql`.
A continuación te doy el **paso a paso detallado** para hacerlo bien y evitar errores:

✓ Requisitos previos

Antes de comenzar, asegúrate de que:

- Tienes **MariaDB** ejecutándose (por ejemplo con XAMPP, WAMP, Laragon, etc.).
 - Tienes **HeidiSQL** instalado.
 - El archivo `tienda_mysql.sql` está ubicado en tu escritorio (`C:\Users\fanny\Desktop\Tienda_Inventario\tienda_mysql.sql`).
-

🗺️ Pasos para importar en HeidiSQL

1. 🔑 Abre HeidiSQL y conecta a tu servidor

- Inicia **HeidiSQL**.
 - Selecciona tu sesión (servidor local o remoto).
 - Hostname/IP: `127.0.0.1` o `localhost`
 - Usuario: `root` (por defecto)
 - Contraseña: (déjala en blanco o la que hayas definido)
 - Puerto: `3306` (por defecto en MariaDB/MySQL)
 - Haz clic en **Conectar**.
-

2. 📁 Crear la base de datos (si no existe)

- En el panel izquierdo (explorador de bases de datos), haz clic derecho y selecciona **"Crear nueva base de datos"**.
- Ponle un nombre, por ejemplo: `tienda_inventario`
- Escoge el **collation** `utf8_general_ci`
- Presiona **OK**.

3. 📁 Importar el archivo SQL

Una vez dentro de la base de datos creada:

- Haz clic en la base de datos `tienda_inventario` para seleccionarla.
- Luego haz clic en **Archivo > Cargar archivo SQL** o presiona `Ctrl + Shift + O`

Busca el archivo:

```
makefile  
C:\Users\fanny\Desktop\Tienda_Inventario\tienda_mysql.sql
```

- Se abrirá el contenido en un nuevo editor de consultas.

4. ▶ Ejecutar el script SQL

- Verifica que arriba diga: **Base de datos seleccionada: tienda_inventario**
- Luego haz clic en el botón **"Ejecutar"** (ícono de rayo), o presiona `F9`.
- Espera a que finalice. Puede tardar unos segundos según el tamaño del archivo.

5. ✅ Verifica la importación

- En el panel izquierdo, despliega la base de datos `tienda_inventario`.
- Asegúrate de que las tablas y datos se hayan creado correctamente.

6. ✅ Para verificar rápido:

1. Confirma que la imagen está en:

```
bash  
tienda_proyecto_php/img/fondo.jpg
```


2. Que el CSS esté en:

bash

```
tienda_proyecto_php/css/estilo.css
```

3. Que en tu `index.php` el CSS se enlace así:

html

```
<link rel="stylesheet" href="css/estilo.css">
```