const sqlite3 = require('sqlite3').verbose();

const db = new sqlite3.Database('./empresa.db', (err) => {

  if (err) return console.error('❌ Error al conectar:', err.message);

  console.log('✅ Conectado a la base de datos empresa.db');

});

db.serialize(() => {

  // =======================

  // Ejercicio 1.1 - Tabla usuarios

  // =======================

  db.run(`

    CREATE TABLE IF NOT EXISTS usuarios (

      id INTEGER PRIMARY KEY AUTOINCREMENT,

      nombre TEXT NOT NULL,

      email TEXT NOT NULL UNIQUE

    )

  `, (err) => {

    if (err) return console.error('❌ Error al crear tabla usuarios:', err.message);

    console.log('✅ Tabla "usuarios" lista.');

  });

  // =======================

  // Ejercicio 1.2 - Tabla productos

  // =======================

  db.run(`

    CREATE TABLE IF NOT EXISTS productos (

      id INTEGER PRIMARY KEY AUTOINCREMENT,

      nombre TEXT NOT NULL,

      precio REAL NOT NULL,

      stock INTEGER DEFAULT 0

    )

  `, (err) => {

    if (err) return console.error('❌ Error al crear tabla productos:', err.message);

    console.log('✅ Tabla "productos" lista.');

  });

  // =======================

  // Ejercicio 2.1 - Insertar un usuario

  // =======================

  db.run(`INSERT INTO usuarios (nombre, email) VALUES (?, ?)`,

    ["Ana López", "ana.lopez@example.com"],

    function(err) {

      if (err) console.error('❌ Insert Ana:', err.message);

      else console.log(`✅ Ana López insertada con ID: ${this.lastID}`);

    }

  );

  // =======================

  // Ejercicio 2.2 - Insertar múltiples usuarios

  // =======================

  const nuevosUsuarios = [

    { nombre: "Carlos García", email: "carlos.garcia@example.com" },

    { nombre: "Laura Martínez", email: "laura.martinez@example.com" },

    { nombre: "Pedro Rodríguez", email: "pedro.rodriguez@example.com" }

  ];

  nuevosUsuarios.forEach(u => {

    db.run(`INSERT INTO usuarios (nombre, email) VALUES (?, ?)`,

      [u.nombre, u.email],

      function(err) {

        if (err) console.error(`❌ Error insertando ${u.nombre}:`, err.message);

        else console.log(`✅ ${u.nombre} insertado con ID: ${this.lastID}`);

      }

    );

  });

  // =======================

  // Ejercicio 2.3 - Insertar productos

  // =======================

  const nuevosProductos = [

    { nombre: "Laptop Gamer", precio: 1250.75, stock: 15 },

    { nombre: "Teclado Mecánico", precio: 150.50, stock: 30 },

    { nombre: "Monitor 27 pulgadas", precio: 300.00, stock: 25 }

  ];

  nuevosProductos.forEach(p => {

    db.run(`INSERT INTO productos (nombre, precio, stock) VALUES (?, ?, ?)`,

      [p.nombre, p.precio, p.stock],

      function(err) {

        if (err) console.error(`❌ Error insertando producto ${p.nombre}:`, err.message);

        else console.log(`✅ Producto ${p.nombre} insertado con ID: ${this.lastID}`);

      }

    );

  });

  // =======================

  // Ejercicio 3.1 - Obtener todos los usuarios ordenados por nombre

  // =======================

  db.all(`SELECT \* FROM usuarios ORDER BY nombre`, [], (err, rows) => {

    if (err) return console.error('❌ Error consultando usuarios:', err.message);

    console.log("\n👥 Usuarios ordenados:");

    rows.forEach(row => console.log(`- ${row.nombre} (${row.email})`));

  });

  // =======================

  // Ejercicio 3.2 - Obtener un usuario por ID

  // =======================

  db.get(`SELECT \* FROM usuarios WHERE id = ?`, [2], (err, row) => {

    if (err) return console.error('❌ Error buscando ID 2:', err.message);

    console.log("\n🔍 Usuario con ID 2:", row ? row.nombre : "No encontrado");

  });

  db.get(`SELECT \* FROM usuarios WHERE id = ?`, [999], (err, row) => {

    if (err) return console.error('❌ Error buscando ID 999:', err.message);

    console.log("🔍 Usuario con ID 999:", row ? row.nombre : "No encontrado");

  });

  // =======================

  // Ejercicio 3.3 - Buscar usuarios por dominio @example.com

  // =======================

  db.all(`SELECT \* FROM usuarios WHERE email LIKE ?`, ['%@example.com'], (err, rows) => {

    if (err) return console.error('❌ Error buscando emails:', err.message);

    console.log("\n📧 Usuarios con email @example.com:");

    rows.forEach(row => console.log(`- ${row.nombre} (${row.email})`));

  });

  // =======================

  // Ejercicio 3.4 - Buscar productos por rango de precio (100 a 500)

  // =======================

  db.all(`SELECT \* FROM productos WHERE precio BETWEEN ? AND ?`, [100, 500], (err, rows) => {

    if (err) return console.error('❌ Error buscando productos:', err.message);

    console.log("\n💰 Productos entre $100 y $500:");

    rows.forEach(row => console.log(`- ${row.nombre}: $${row.precio}`));

  });

  // =======================

  // Ejercicio 3.5 - Procesar usuarios uno por uno

  // =======================

  db.each(`SELECT nombre FROM usuarios`, [], (err, row) => {

    if (err) return console.error('❌ Error en db.each:', err.message);

    console.log(`👋 Hola, ${row.nombre}`);

  }, (err, count) => {

    if (err) return console.error('❌ Error final de db.each:', err.message);

    console.log(`✅ Se ha saludado a todos los usuarios. Total: ${count}`);

  });

  // =======================

  // Cierre de conexión

  // =======================

  db.close((err) => {

    if (err) return console.error('❌ Error al cerrar DB:', err.message);

    console.log('\n🔒 Conexión cerrada correctamente.');

  });

});

// =======================

// Ejercicio 4.1 - Actualizar el Email de un Usuario

// =======================

db.run(

  `UPDATE usuarios SET email = ? WHERE id = ?`,

  ["nuevo.email@example.com", 2],

  function(err) {

    if (err) return console.error("❌ Error actualizando email:", err.message);

    if (this.changes === 0) {

      console.log("⚠️ No se encontró el usuario con ID 2 para actualizar.");

    } else {

      console.log(`✅ Email actualizado correctamente para el usuario con ID 2.`);

    }

  }

);

// =======================

// Ejercicio 4.2 - Incrementar Stock de un Producto

// =======================

db.run(

  `UPDATE productos SET stock = stock + ? WHERE id = ?`,

  [10, 1],

  function(err) {

    if (err) return console.error("❌ Error actualizando stock:", err.message);

    if (this.changes === 0) {

      console.log("⚠️ No se encontró el producto con ID 1 para actualizar stock.");

    } else {

      console.log("✅ Stock incrementado en 10 unidades para el producto con ID 1.");

    }

  }

);

**Dónde incluir el ejercicio 4.1 y 4.2**

Agrega estos bloques **antes del cierre de conexión (db.close(...))**, para que se ejecuten después de las inserciones y consultas de las secciones anteriores.

// =======================

// Ejercicio 5.1 - Eliminar un Usuario por Email

// =======================

db.run(

  `DELETE FROM usuarios WHERE email = ?`,

  ["pedro.rodriguez@example.com"],

  function(err) {

    if (err) return console.error("❌ Error al eliminar usuario:", err.message);

    if (this.changes === 0) {

      console.log("⚠️ No se encontró usuario con ese email para eliminar.");

    } else {

      console.log(`✅ Usuario con email "pedro.rodriguez@example.com" eliminado.`);

    }

  }

);

// =======================

// Ejercicio 5.2 - Eliminar Productos sin Stock

// =======================

db.run(

  `DELETE FROM productos WHERE stock = 0`,

  [],

  function(err) {

    if (err) return console.error("❌ Error al eliminar productos sin stock:", err.message);

    if (this.changes === 0) {

      console.log("ℹ️ No había productos con stock = 0 para eliminar.");

    } else {

      console.log(`✅ ${this.changes} producto(s) sin stock eliminado(s).`);

    }

  }

);

**Al igual que con los ejercicios anteriores**

 coloca estos bloques **antes del db.close()** en tu [app.js](http://app.js).

 // =======================

  // Sección 6: Cierre de la Conexión

  // =======================

  // =======================

  // Cierre de conexión

  // =======================

  db.close((err) => {

    if (err) return console.error('❌ Error al cerrar DB:', err.message);

    console.log('\n🔒 Conexión cerrada correctamente.');

  });

});

**Este paso ya lo estamos haciendo** correctamente en el db.close(...) al final del db.serialize(...).

// =======================

// Ejercicio 7.1 - Crear tabla posts con FOREIGN KEY

// =======================

db.run(`

  CREATE TABLE IF NOT EXISTS posts (

    id INTEGER PRIMARY KEY AUTOINCREMENT,

    titulo TEXT NOT NULL,

    contenido TEXT NOT NULL,

    autor\_id INTEGER,

    FOREIGN KEY (autor\_id) REFERENCES usuarios(id)

  )

`, (err) => {

  if (err) return console.error("❌ Error al crear tabla posts:", err.message);

  console.log("✅ Tabla 'posts' lista.");

});

// Insertar algunos posts

const posts = [

  { titulo: "Bienvenidos al blog", contenido: "Este es el primer post", autor\_id: 1 },

  { titulo: "Tips de programación", contenido: "Usa async/await en Node.js", autor\_id: 2 }

];

posts.forEach(post => {

  db.run(

    `INSERT INTO posts (titulo, contenido, autor\_id) VALUES (?, ?, ?)`,

    [post.titulo, post.contenido, post.autor\_id],

    function(err) {

      if (err) return console.error(`❌ Error insertando post "${post.titulo}":`, err.message);

      console.log(`✅ Post "${post.titulo}" insertado con ID: ${this.lastID}`);

    }

  );

});

// Consulta con JOIN para mostrar nombre del autor

db.all(`

  SELECT posts.titulo, posts.contenido, usuarios.nombre AS autor

  FROM posts

  JOIN usuarios ON posts.autor\_id = usuarios.id

`, [], (err, rows) => {

  if (err) return console.error("❌ Error en consulta JOIN:", err.message);

  console.log("\n📝 Posts con nombre de autor:");

  rows.forEach(row => {

    console.log(`- "${row.titulo}" por ${row.autor}: ${row.contenido}`);

  });

});

// =======================

// Ejercicio 7.2 - Función query con Promesas + async/await

// =======================

function query(sql, params = []) {

  return new Promise((resolve, reject) => {

    db.all(sql, params, (err, rows) => {

      if (err) reject(err);

      else resolve(rows);

    });

  });

}

async function mostrarUsuariosConPromesa() {

  console.log("\n📋 Usuarios usando Promesas:");

  try {

    const usuarios = await query(`SELECT \* FROM usuarios ORDER BY nombre`);

    usuarios.forEach(u => {

      console.log(`- ${u.nombre} (${u.email})`);

    });

  } catch (err) {

    console.error("❌ Error en mostrarUsuariosConPromesa:", err.message);

  }

}

mostrarUsuariosConPromesa();

// =======================

// Ejercicio 7.3 - Buscar posts por email del autor (JOIN + WHERE)

// =======================

db.all(`

  SELECT posts.titulo, posts.contenido, usuarios.email

  FROM posts

  JOIN usuarios ON posts.autor\_id = usuarios.id

  WHERE usuarios.email = ?

`, ["laura.martinez@example.com"], (err, rows) => {

  if (err) return console.error("❌ Error buscando posts por email:", err.message);

  console.log("\n🔎 Posts escritos por laura.martinez@example.com:");

  if (rows.length === 0) {

    console.log("ℹ️ No hay posts de este autor.");

  } else {

    rows.forEach(row => {

      console.log(`- "${row.titulo}": ${row.contenido}`);

    });

  }

});

⚠️ Importante: Para que la conexión no se cierre **antes** de que se ejecuten las promesas, debes mover db.close(...) al final del async function