Complemento practica 9.

La **inyección SQL**, cómo sucede, y qué significa **"usar consultas parametrizadas"**, con ejemplos simples y comprensibles en Python usando sqlite3.

## 🛑 ¿Qué es una inyección SQL?

Es una técnica en la que un atacante escribe **código SQL malicioso** dentro de un campo de entrada (por ejemplo, un formulario), para **manipular la base de datos**.

### 🎯 Objetivo del atacante:

* Leer información privada.
* Borrar datos.
* Iniciar sesión como otro usuario.
* Modificar la base de datos sin permiso.

## 🧪 Ejemplo inseguro — Código vulnerable

import sqlite3

conn = sqlite3.connect("usuarios.db")

cursor = conn.cursor()

usuario = input("Ingresa tu nombre de usuario: ")

consulta = f"SELECT \* FROM usuarios WHERE nombre = '{usuario}'"

cursor.execute(consulta)

resultados = cursor.fetchall()

print(resultados)  
  
NOTA: en Python es una **f-string** (formato de cadena literal) que permite **insertar variables o expresiones dentro de una cadena de texto**, usando llaves {}  
  
nombre = "Juan"

consulta = f"SELECT \* FROM usuarios WHERE nombre = '{nombre}'"

print(consulta)  
Salida.. SELECT \* FROM usuarios WHERE nombre = 'Juan'

### ¿Qué pasa si el atacante escribe esto?

' OR '1'='1

La consulta se convierte en:

SELECT \* FROM usuarios WHERE nombre = '' OR '1'='1'

'1'='1' **siempre es verdadero**, entonces se devuelven **todos los usuarios**, aunque no tenga un nombre válido.  
Esto da acceso a todos los usuarios (inyección SQL).

## ✅ Solución segura — Consultas parametrizadas

python

import sqlite3

conn = sqlite3.connect("usuarios.db")

cursor = conn.cursor()

usuario = input("Ingresa tu nombre de usuario: ")  
if usuario == “ “

consulta = f "SELECT \* FROM usuarios WHERE nombre = ?"

cursor.execute(consulta, (usuario)) 👈 Aquí está la diferencia

resultados = cursor.fetchall()

print(resultados)

### ¿Qué es una ****consulta parametrizada****?

Es cuando el **valor** que el usuario ingresa **no se concatena directamente al texto SQL**, sino que se envía **por separado**.  
El motor de base de datos trata ese valor **solo como datos**, no como código ejecutable.

## ✅ Beneficios de usar consultas parametrizadas

* Protege contra inyección SQL.
* Se limpian (escapan) los caracteres peligrosos automáticamente.
* Mejor para mantenimiento del código.
* Funciona con todas las bases de datos modernas (SQLite, MySQL, PostgreSQL, etc.).