Arquivo: limpa_db.py

```
import sqlite3
import tkinter as tk
from tkinter import filedialog, messagebox, ttk

class DatabaseCleanerApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Gerenciador de Banco SQLite")

        # Variáveis
        self.conn = None
        self.cursor = None
        self.tabelas = []
        self.tabela_selecionada = tk.StringVar()

        # Botão escolher banco
        btn_escolher = tk.Button(root, text="Escolher Banco de Dados", command=self.abrir_banco)
        btn_escolher.pack(pady=10)

        # Combobox para selecionar tabela
          self.combo = ttk.Combobox(root, textvariable=self.tabela_selecionada, state="readonly",
width=50)
        self.combo.pack(pady=10)

        # Frame para botões de ação
        frame_botoes = tk.Frame(root)
        frame_botoes.pack(pady=10)

                        btn_apagar   =   tk.Button(frame_botoes,   text="Apagar   Informações",
command=self.apagar_informacoes, bg="orange")
        btn_apagar.grid(row=0, column=0, padx=5)

         btn_excluir = tk.Button(frame_botoes, text="Excluir Tabela", command=self.excluir_tabela,
bg="red", fg="white")
        btn_excluir.grid(row=0, column=1, padx=5)

    def abrir_banco(self):
        caminho = filedialog.askopenfilename(
            title="Selecione o banco de dados SQLite",
                filetypes=[("SQLite Database", "*.db *.sqlite *.sqlite3"), ("Todos os arquivos",
"*.*")]
        )

        if not caminho:
            return

        try:
            self.conn = sqlite3.connect(caminho)
            self.cursor = self.conn.cursor()
```

```
            # Buscar tabelas
            self.cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
            self.tabelas = [t[0] for t in self.cursor.fetchall() if t[0] != "sqlite_sequence"]

            if not self.tabelas:
                messagebox.showwarning("Aviso", "Nenhuma tabela encontrada nesse banco.")
                return

            self.combo["values"] = self.tabelas
            self.combo.current(0)

                 messagebox.showinfo("Sucesso", f"Banco '{caminho}' aberto!\n\n{len(self.tabelas)}
tabelas encontradas.")

        except Exception as e:
            messagebox.showerror("Erro", f"Falha ao abrir o banco:\n{e}")

    def apagar_informacoes(self):
        tabela = self.tabela_selecionada.get()
        if not tabela:
            messagebox.showwarning("Aviso", "Selecione uma tabela primeiro.")
            return

        try:
            self.cursor.execute(f"DELETE FROM {tabela};")
            self.conn.commit()
                 messagebox.showinfo("Sucesso", f"Todas as informações da tabela '{tabela}' foram
apagadas!")
        except Exception as e:
            messagebox.showerror("Erro", f"Falha ao apagar informações:\n{e}")

    def excluir_tabela(self):
        tabela = self.tabela_selecionada.get()
        if not tabela:
            messagebox.showwarning("Aviso", "Selecione uma tabela primeiro.")
            return

          resposta = messagebox.askyesno("Confirmar", f"Tem certeza que deseja excluir a tabela
'{tabela}'?")
        if not resposta:
            return

        try:
            self.cursor.execute(f"DROP TABLE {tabela};")
            self.conn.commit()

            # Atualizar lista
            self.tabelas.remove(tabela)
            self.combo["values"] = self.tabelas
            if self.tabelas:
                self.combo.current(0)
            else:
                self.combo.set("")
```

```
            messagebox.showinfo("Sucesso", f"Tabela '{tabela}' foi excluída!")
        except Exception as e:
            messagebox.showerror("Erro", f"Falha ao excluir tabela:\n{e}")

if __name__ == "__main__":
    root = tk.Tk()
    app = DatabaseCleanerApp(root)
    root.mainloop()
```