TRƯỜNG ĐẠI HỌC VĂN LANG
KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP 2

LẬP TRÌNH PYTHON NÂNG CAO

Chủ đề:

WINFORM QUẢN LÝ SÁCH CÓ KẾT NỐI DATABASE

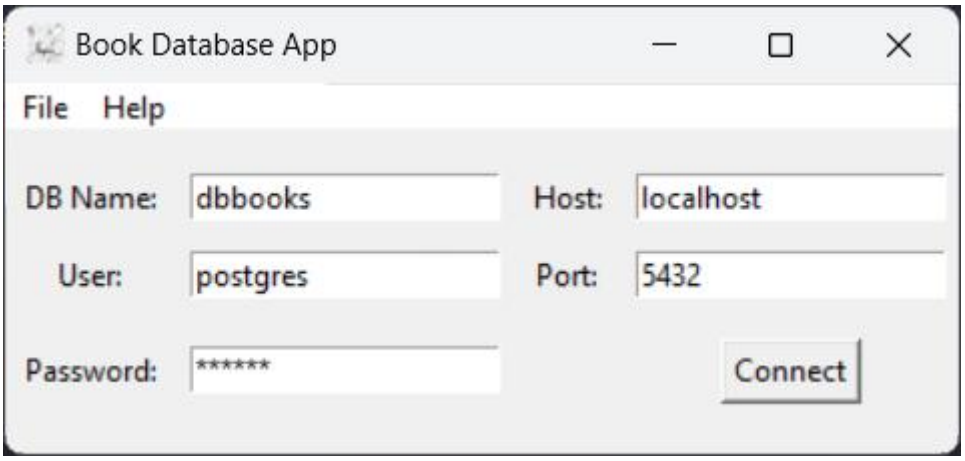
SVTH: Nguyễn Liên Nhi - 2274802010612
LỚP: 241_71ITSE31003_02
GVHD:Thầy Huỳnh Thái Học

TP. Hồ Chí Minh – 4 / 2024

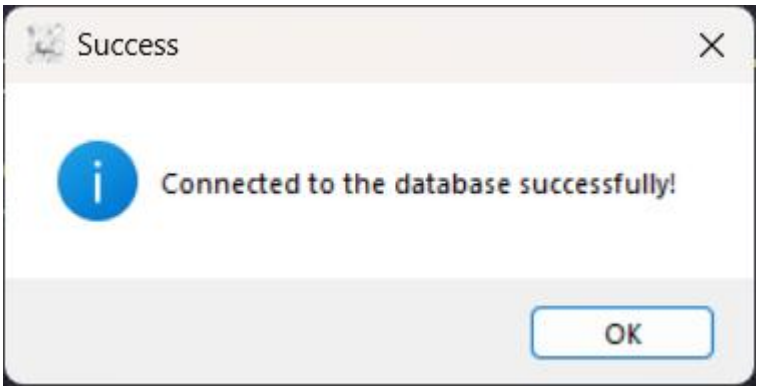
Mục lục

1. Giao diện

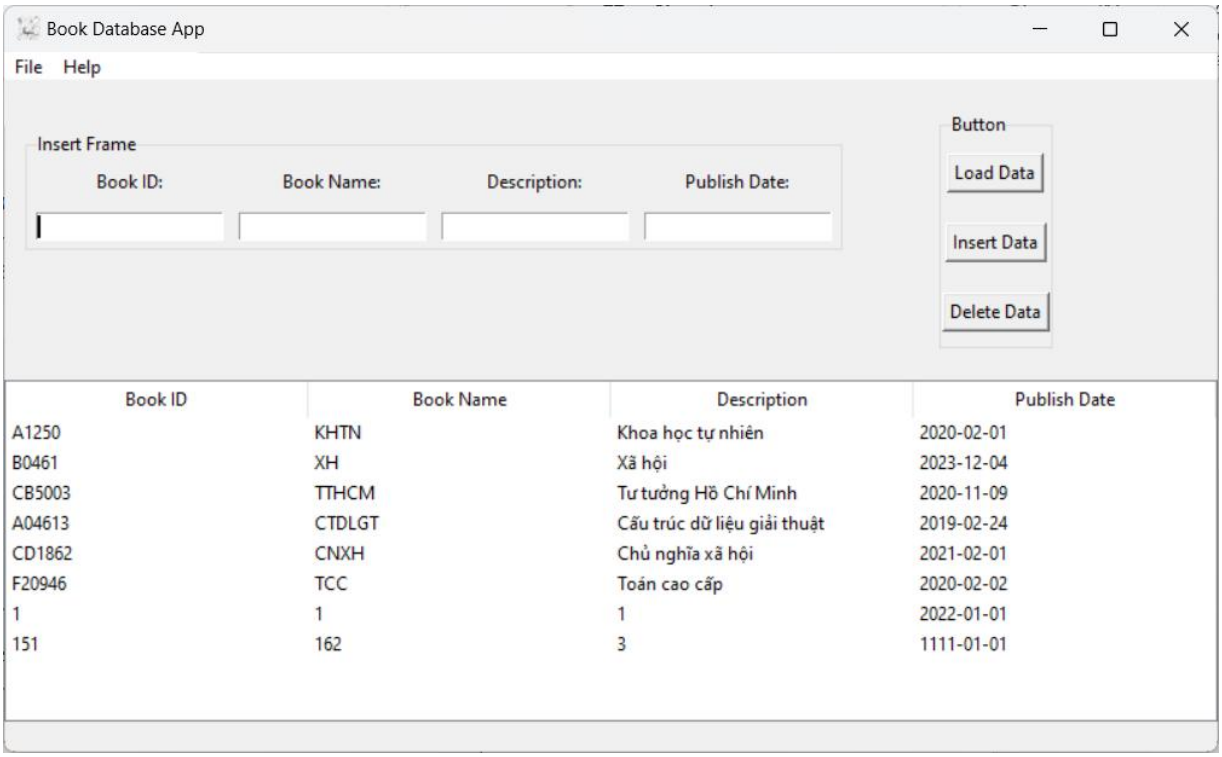
Giao diện connect database:



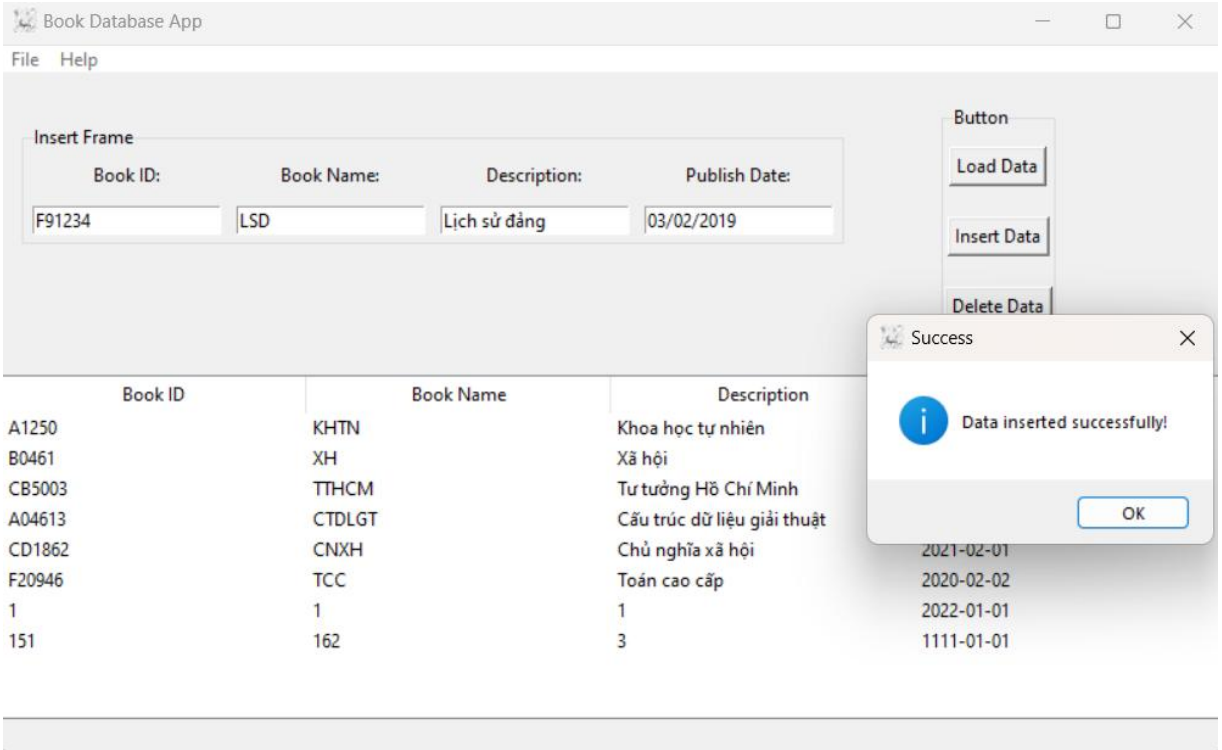
Thông báo khi connect thành công:



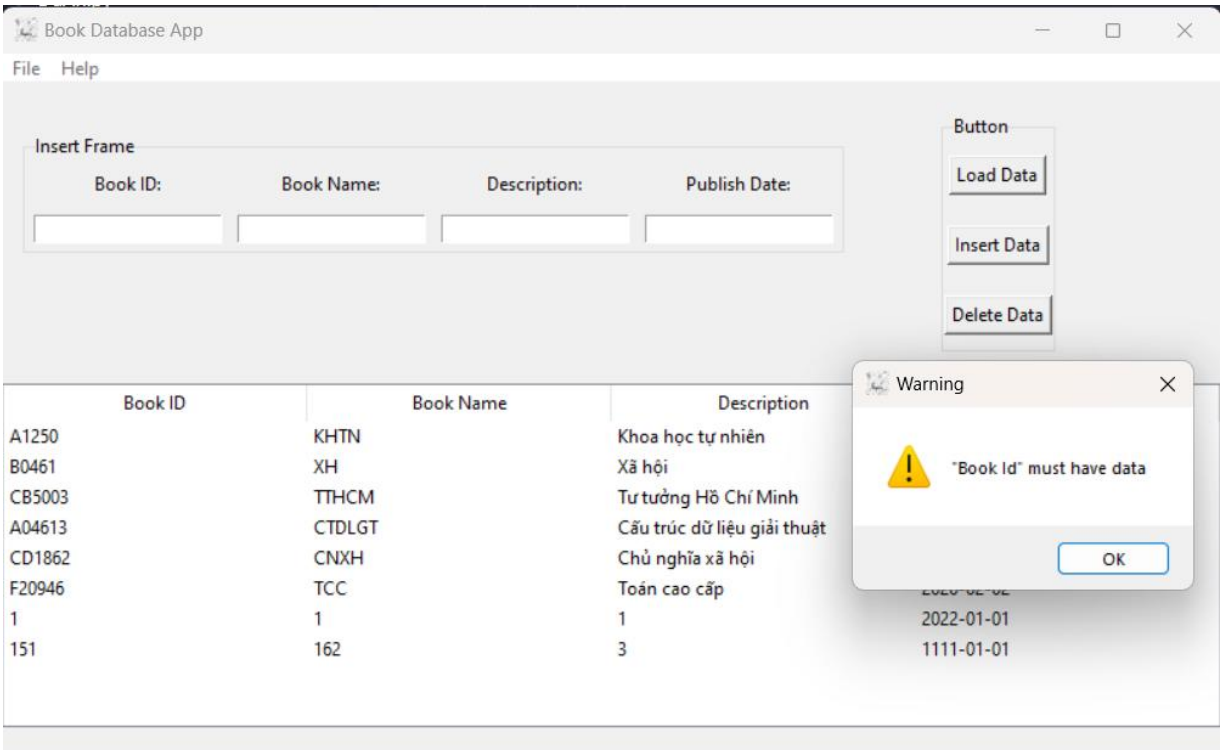
Giao diện chính:



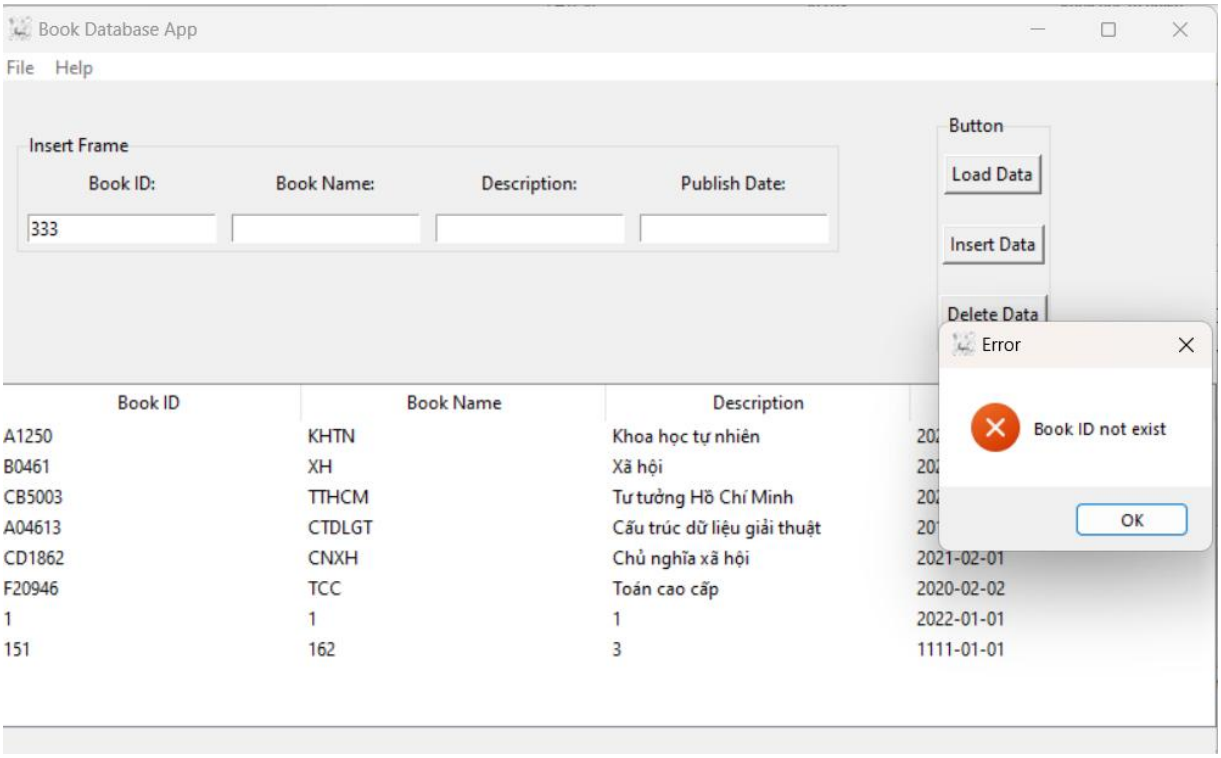
Thông báo thêm nội dung thành công:



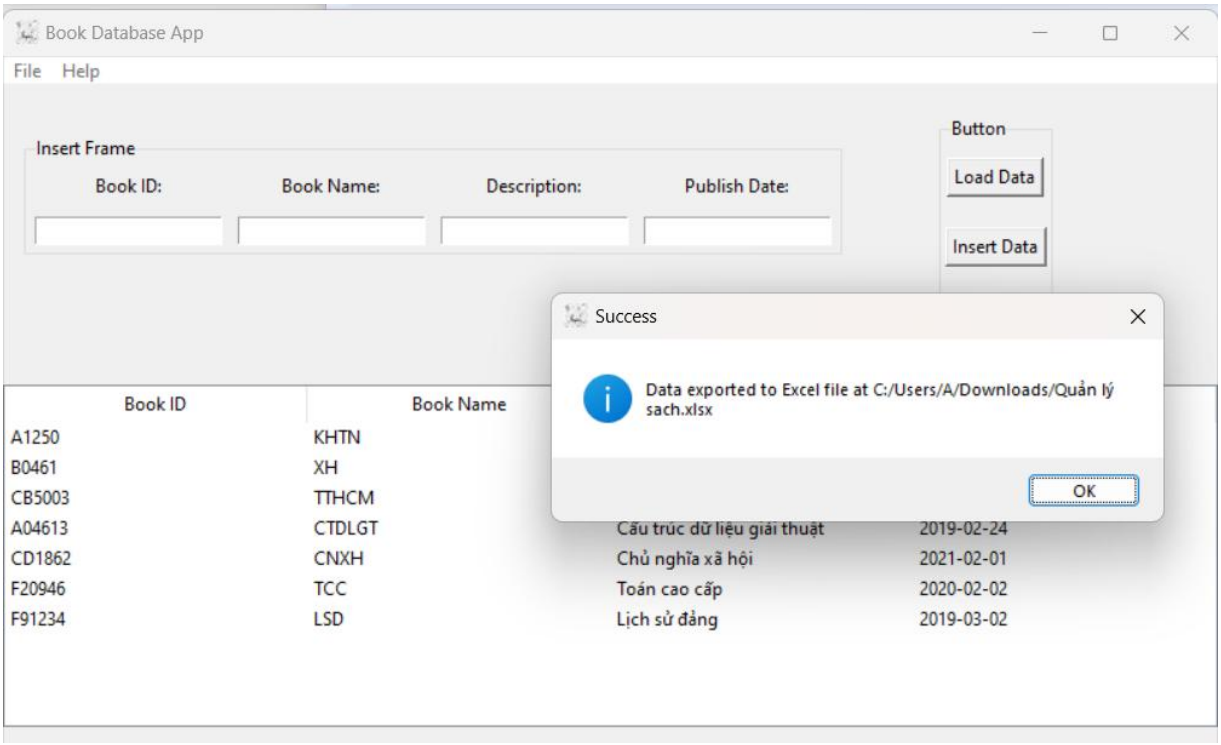
Thông báo xoá nội dung thất bại khi không nhập id - mã sách cần xoá:



Thông báo xoá nội dung thất bại khi không nhập id - mã sách không tồn tại:



Thông báo sau khi xuất toàn bộ nội dung thành file excel



2. Chức năng

2.1. Kết nối - Connect database

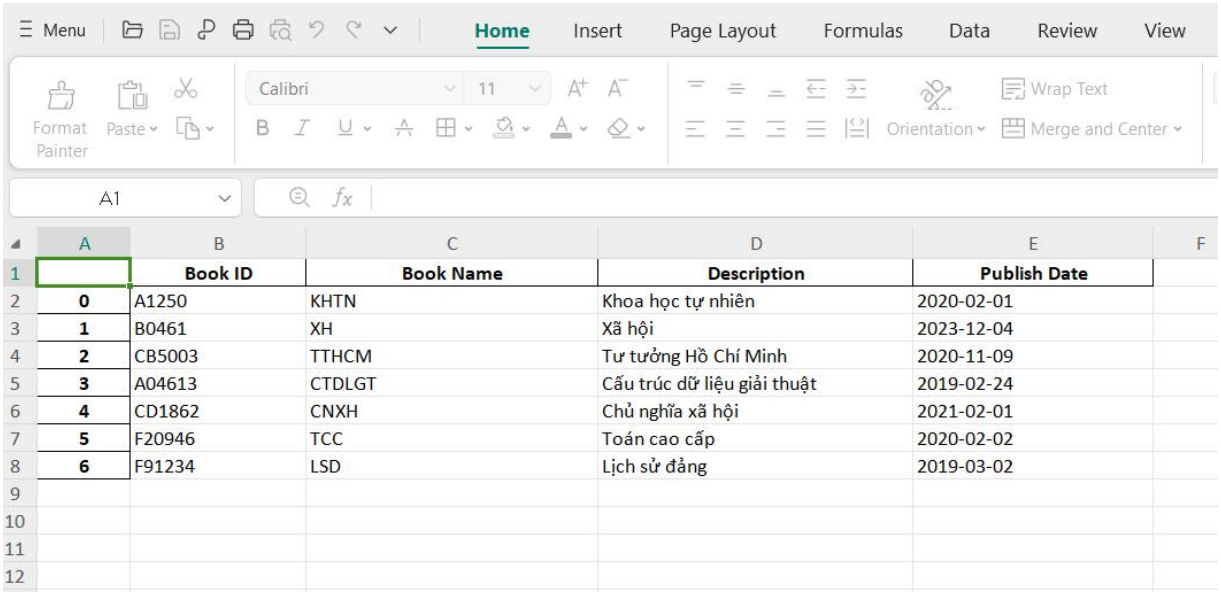
Kết nối thành công với database.

2.2. Quản lý thông tin sách theo dạng bảng

- Cho phép thêm, xoá nội dung thông tin sách.
- Báo lỗi khi không nhập thông tin thêm sách hoặc thông tin sách mới bị trùng lặp.
- Báo lỗi khi không nhập thông tin xoá sách hoặc nhập thông tin xoá sách không tồn tại.

2.3. Xuất thông tin thành file Excel:

Cột đầu tiên sẽ tự động đánh số thứ tự số hàng nội dung bắt đầu từ số 0.



	A	B	C	D	E
1		**Book ID**	**Book Name**	**Description**	**Publish Date**
2	**0**	A1250	KHTN	Khoa học tự nhiên	2020-02-01
3	**1**	B0461	XH	Xã hội	2023-12-04
4	**2**	CB5003	TTHCM	Tư tưởng Hồ Chí Minh	2020-11-09
5	**3**	A04613	CTDLGT	Cấu trúc dữ liệu giải thuật	2019-02-24
6	**4**	CD1862	CNXH	Chủ nghĩa xã hội	2021-02-01
7	**5**	F20946	TCC	Toán cao cấp	2020-02-02
8	**6**	F91234	LSD	Lịch sử đảng	2019-03-02
9					
10					
11					
12					

3. Mã nguồn

```python
import tkinter as tk
from tkinter import Menu, ttk
from tkinter import messagebox, filedialog
import psycopg2
from psycopg2 import sql
import pandas as pd

class BookDatabaseApp:
    def __init__(self, root):
        self.root = root
        self.root.iconbitmap('icon.ico')
        self.root.title("Book Database App")

        # Frame for menu
        self.menu_bar = Menu(self.root)
        file_menu = Menu(self.menu_bar, tearoff=0)
        self.menu_bar.add_cascade(label="File", menu=file_menu)
        file_menu.add_command(label="Export to Excel", command=self.export_to_excel)
        file_menu.add_command(label="Exit",command=self.exit_widgets)

        help_menu = Menu(self.menu_bar,tearoff=0)
        self.menu_bar.add_cascade(label="Help",menu=help_menu)
        help_menu.add_command(label="About",command=self.msg_about)

        self.root.config(menu=self.menu_bar)

        # Database connection fields
        self.db_name = tk.StringVar(value='dbbooks')
        self.user = tk.StringVar(value='postgres')
        self.password = tk.StringVar(value='123456')
        self.host = tk.StringVar(value='localhost')
        self.port = tk.StringVar(value='5432')
        self.table_name = tk.StringVar(value='books')

        # Show connection frame
        self.connection_frame()

    def msg_about(self):
        messagebox.showinfo("Infromation","This GUI made by Nhi Nguyen 19/10/2024")

```

```python
def connection_frame(self):
        # Connection frame
        self.connection_frame = tk.Frame(self.root)
        self.connection_frame.grid(pady=10)

        tk.Label(self.connection_frame, text="DB Name:").grid(row=0, column=0, padx=5, pady=5)
        tk.Entry(self.connection_frame, textvariable=self.db_name).grid(row=0, column=1, padx=5, pady=5)

        tk.Label(self.connection_frame, text="User:").grid(row=1, column=0, padx=5, pady=5)
        tk.Entry(self.connection_frame, textvariable=self.user).grid(row=1, column=1, padx=5, pady=5)

        tk.Label(self.connection_frame, text="Password:").grid(row=2, column=0, padx=5, pady=5)
        tk.Entry(self.connection_frame, textvariable=self.password, show="*").grid(row=2, column=1, padx=5, pady=5)

        tk.Label(self.connection_frame, text="Host:").grid(row=0, column=2, padx=5, pady=5)
        tk.Entry(self.connection_frame, textvariable=self.host).grid(row=0, column=3, padx=5, pady=5)

        tk.Label(self.connection_frame, text="Port:").grid(row=1, column=2, padx=5, pady=5)
        tk.Entry(self.connection_frame, textvariable=self.port).grid(row=1, column=3, padx=5, pady=5)

        connect_button = tk.Button(self.connection_frame, text="Connect", command=self.trans)
        connect_button.grid(row=2, column=3, columnspan=2, pady=10)

```

```python
def show_db_interface(self):
        # Frame for database interface
        self.db_interface_frame = tk.Frame(self.root)
        self.db_interface_frame.grid(pady=10)

        insert_frame = ttk.LabelFrame(self.db_interface_frame,text="Insert Frame")
        insert_frame.grid(pady=10,column=0)

        self.column1 = tk.StringVar()
        self.column2 = tk.StringVar()
        self.column3 = tk.StringVar()
        self.column4 = tk.StringVar()

        tk.Label(insert_frame, text="Book ID:").grid(row=0, column=0, padx=5, pady=5)
        tk.Entry(insert_frame, textvariable=self.column1).grid(row=1, column=0, padx=5, pady=5)

        tk.Label(insert_frame, text="Book Name:").grid(row=0, column=1, padx=5, pady=5)
        tk.Entry(insert_frame, textvariable=self.column2).grid(row=1, column=1, padx=5, pady=5)

        tk.Label(insert_frame, text="Description:").grid(row=0, column=2, padx=5, pady=5)
        tk.Entry(insert_frame, textvariable=self.column3).grid(row=1, column=2, padx=5, pady=5)

        tk.Label(insert_frame, text="Publish Date:").grid(row=0, column=3, padx=5, pady=5)
        tk.Entry(insert_frame, textvariable=self.column4).grid(row=1, column=3, padx=5, pady=5)

        button_frame = ttk.LabelFrame(self.db_interface_frame,text="Button")
        button_frame.grid(padx=5,pady=10,column=2 , row=0 , rowspan=3)
        tk.Button(button_frame, text="Load Data", command=self.load_data).grid(row=0, column=0,columnspan=2, pady=10)
        tk.Button(button_frame, text="Insert Data", command=self.event_insert).grid(row=1, column=0, columnspan=2, pady=10)
        tk.Button(button_frame, text="Delete Data", command=self.event_delete).grid(row=2, column=0, columnspan=2, pady=10)

```

```python
# Treeview to display data
        self.tree = ttk.Treeview(self.db_interface_frame, columns=("masach", "tensach", "mota", "ngayxuatban"), show="headings")
        self.tree.heading("masach", text="Book ID")
        self.tree.heading("tensach", text="Book Name")
        self.tree.heading("mota", text="Description")
        self.tree.heading("ngayxuatban", text="Publish Date")
        self.tree.grid(pady=10, columnspan=6)

        # Load data after connection
        self.load_data()
```

```python
def connect_db(self):
        try:
            # Attempt to connect to the database
            self.conn = psycopg2.connect(
                dbname=self.db_name.get(),
                user=self.user.get(),
                password=self.password.get(),
                host=self.host.get(),
                port=self.port.get()
            )
            self.cur = self.conn.cursor()

            # Show success message
            messagebox.showinfo("Success", "Connected to the database successfully!")
            return True
        except Exception as e:
            messagebox.showerror("Error", f"Error connecting to the database: {e}")
            return False

    def check_exist(self, bookid , bookname):
        # Lấy tất cả các hàng (ID của các mục) trong Treeview
        children = self.tree.get_children()
        # Duyệt qua từng hàng
        for child in children:
            # Lấy dữ liệu của từng hàng (ở đây giả trị Books id có thể nằm trong cột đầu tiên)
            values = self.tree.item(child, 'values')
            if values[0] == bookid or values[1] == bookname :
                return True
        return False

    def event_load_data(self, rows):
        self.clear_input()
        for item in self.tree.get_children():
            self.tree.delete(item)

        for row in rows:
            self.tree.insert('', tk.END, values=row)

    def load_data(self):
        try:
            query = sql.SQL("SELECT * FROM {}").format(sql.Identifier(self.table_name.get()))
            self.cur.execute(query)
            rows = self.cur.fetchall()
            self.event_load_data(rows)

        except Exception as e:
            self.conn.rollback()
            messagebox.showerror("Error", f"Error loading data: {e}")

    def event_insert(self):
        if self.validation_masach() :
            if self.check_exist(self.column1.get(), self.column2.get()):
                messagebox.showerror("Error","Book ID existed")
            else :
                self.insert_data()
                self.clear_input()
        else:
            messagebox.showwarning("Warning",' "Book Id" must have data')


    def insert_data(self):
        try:
            query = sql.SQL("INSERT INTO {} (masach, tensach, mota, ngayxuatban) VALUES (%s, %s, %s, %s)").format(sql.Identifier(self.table_name.get()))
            data = (self.column1.get(), self.column2.get(), self.column3.get(), self.column4.get())
            self.cur.execute(query, data)
            self.conn.commit()
            messagebox.showinfo("Success", "Data inserted successfully!")
        except Exception as e:
            self.conn.rollback()
            messagebox.showerror("Error", f"Error inserting data: {e}")
        self.load_data()

    def event_delete(self):
        if self.validation_masach():
            if self.check_exist(self.column1.get(),self.column2.get()):
                self.delete_data()
                self.clear_input()
            else:
                messagebox.showerror("Error","Book ID not exist")

        else:
            messagebox.showwarning("Warning",' "Book Id" must have data')

```

```python
def delete_data(self):
        try:
            delete_query = sql.SQL("DELETE FROM {} WHERE masach = %s").format(sql.Identifier(self.table_name.get()))
            data_to_delete = (self.column1.get(),)
            self.cur.execute(delete_query, data_to_delete)
            self.conn.commit()
            messagebox.showinfo("Success", "Data deleted successfully!")
        except Exception as e:
            self.conn.rollback()
            messagebox.showerror("Error", f"Error deleting data: {e}")
        self.load_data()


    def trans(self):
        if self.connect_db() :
             # Hide connection frame and display database interface
            self.connection_frame.grid_forget()
            self.show_db_interface()

            # Load data after connection

    def validation_masach(self):
       if self.column1.get().strip() != "":
           return True
       else :
           return False

    def clear_input(self):
        self.column1.set("")
        self.column2.set("")
        self.column3.set("")
        self.column4.set("")


    def exit_widgets(self):
        self.root.destroy()

    def export_to_excel(self):
        try:
            rows = []
            for item in self.tree.get_children():
                row_data = self.tree.item(item)['values']
                rows.append(row_data)

            if not rows:
                messagebox.showwarning("Warning", "No data to export!")
                return

            col_names = [self.tree.heading(col)['text'] for col in self.tree["columns"]]

            export_excel = pd.DataFrame(rows, columns=col_names)

            file_path = filedialog.asksaveasfilename(defaultextension=".xlsx",
                                                     filetypes=[("Excel files", "*.xlsx"), ("All files", "*.*")])
            if file_path:
                export_excel.to_excel(file_path)
                messagebox.showinfo("Success", f"Data exported to Excel file at {file_path}")
            else:
                messagebox.showwarning("Warning", "You didn't select a file location.")
        except Exception as e:
            messagebox.showerror("Error", f"An error occurred while exporting data to Excel: {e}")

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

```

4. Github

https://github.com/NHIIIIIIIIII/Project_AdvancePython.git