import tkinter  
import pickle  
import tkinter as tk  
from tkinter import messagebox  
from tkinter import ttk  
import psycopg2  
import openpyxl  
from openpyxl import Workbook  
import os  
from datetime import datetime  
  
try:  
 conn = psycopg2.connect('dbname=Kalonov user=postgres password=25749681 host=localhost port=5432')  
 print("Успешное подключение к базе данных PostgreSQL")  
except Exception as e:  
 print(f"Ошибка подключения к базе данных PostgreSQL: {e}")  
  
  
def category():  
 # Создаем графический интерфейс  
  
 root = tk.Tk()  
 root.title("document")  
 root.resizable(False, False) # Это запретит изменение размеров окна по ширине и по высоте  
  
 # Создание таблицы Отдел  
 tree = ttk.Treeview(root)  
 tree["columns"] = ('ID', 'category')  
 tree.heading('#0', text='№')  
 tree.heading('ID', text='ID')  
 tree.heading('category', text='Категория')  
  
 # Устанавливаем размеры столбцов  
 tree.column("#0", width=40)  
 tree.column("ID", width=100, anchor=tk.CENTER)  
 tree.column("category", width=170, anchor=tk.CENTER)  
  
 tree.pack()  
  
  
 try:  
 # удаление устаревших данных  
 tree.delete(\*tree.get\_children())  
  
 cursor = conn.cursor()  
 cursor.execute("SELECT \* FROM \"Category\"")  
  
 rows = cursor.fetchall()  
  
 for i, row in enumerate(rows):  
 tree.insert("", "end", text=str(i), values=row)  
  
 except Exception as e:  
 print(f"Ошибка при загрузке данных: {e}")  
  
 def save\_to\_excel():  
 try:  
 wb = openpyxl.Workbook()  
 ws = wb.active  
 ws.append([column[0] for column in cursor.description]) # Заголовки столбцов  
  
 for row\_data in rows:  
 ws.append(row\_data)  
  
 desktop\_path = os.path.join(os.path.join(os.environ['USERPROFILE']), 'Desktop')  
 timestamp = datetime.now().strftime("%Y-%m-%d\_%H-%M-%S")  
 save\_path = os.path.join(desktop\_path, f"department\_data\_{timestamp}.xlsx")  
  
 if save\_path:  
 wb.save(save\_path)  
 print("Данные успешно сохранены в файл Excel!")  
 except Exception as e:  
 print(f"Ошибка при сохранении данных в Excel: {e}")  
  
 # Добавление данных в таблицу из бд  
 def insert():  
 try:  
 # удаление устаревших данных  
 tree.delete(\*tree.get\_children())  
  
 cursor = conn.cursor()  
 cursor.execute("SELECT \* FROM \"Category\"")  
 rows = cursor.fetchall()  
  
 for i, row in enumerate(rows):  
 tree.insert("", "end", text=str(i), values=row)  
  
 except Exception as e:  
 print(f"Ошибка при загрузке данных: {e}")  
  
 # Добавление новой строки в таблицу  
 def add\_string():  
 idc = entry.get()  
 cat = entry2.get()  
  
 try:  
 cursor = conn.cursor()  
 cursor.execute("INSERT INTO \"Category\" (\"idc\", \"category\" )"  
 " VALUES (%s, %s)",  
 (idc, cat))  
 conn.commit()  
  
 insert() # Обновить вывод таблицы после добавления строки  
 entry.delete(0, tk.END)  
 entry2.delete(0, tk.END)  
  
 except Exception as e:  
 # Если произошла ошибка, откатываем транзакцию  
 conn.rollback()  
  
 error\_msg = f"Ошибка при загрузке данных: {e}"  
 print(error\_msg)  
 messagebox.showerror("Ошибка", error\_msg)  
  
 def on\_enter(event):  
 # Поместите здесь ваше действие  
 add\_string()  
  
 # Привязываем клавишу Enter к функции on\_enter  
 root.bind('<Return>', on\_enter)  
  
 def delete\_all\_records():  
 try:  
 selected\_item = tree.selection()  
  
 for item in selected\_item:  
 values = tree.item(item, 'values')  
 ID\_department = values[0]  
  
 cursor = conn.cursor()  
  
 # Начало транзакции  
 cursor.execute("START TRANSACTION;")  
  
 try:  
  
 # Удаление данных из таблицы 'department'  
 cursor.execute("DELETE FROM \"Documents\" WHERE \"categoryid\" = %s", (ID\_department,))  
  
 # Удаление данных из таблицы 'department'  
 cursor.execute("DELETE FROM \"Category\" WHERE \"idc\" = %s", (ID\_department,))  
  
 conn.commit()  
  
 insert() # Обновить вывод таблицы после удаления всех записей  
 except Exception as e:  
 conn.rollback()  
 error\_msg = f"Ошибка при удалении данных из таблиц: Убедитесь, что вы удалили всех сотрудников из данного отдела {e}"  
 print(error\_msg)  
 messagebox.showerror("Ошибка", error\_msg)  
 return  
  
 except Exception as e:  
 conn.rollback()  
 error\_msg = f"Ошибка при удалении данных из таблиц: {e}"  
 print(error\_msg)  
 messagebox.showerror("Ошибка", error\_msg)  
  
 def confirm\_action():  
 result = messagebox.askyesno("Confirmation", "Вы уверены, что хотите очистить?")  
 if result:  
 # Put your action here  
 delete\_all\_records()  
 else:  
 print("Action canceled.")  
  
 def update\_string():  
 id\_value = entry.get()  
 name = entry2.get()  
  
 if not id\_value:  
 messagebox.showerror("Ошибка",  
 "Пожалуйста, введите корректные значения для ID отдела и количества сотрудников.")  
 return  
  
 try:  
 id\_value = int(id\_value)  
  
 cursor = conn.cursor()  
 cursor.execute(  
 "UPDATE \"Category\" SET \"category\" = %s WHERE \"idc\" = %s",  
 (name, id\_value))  
 conn.commit()  
  
 # Обновляем поля после успешного обновления  
 entry.delete(0, tk.END)  
 entry.insert(0, id\_value)  
 entry2.delete(0, tk.END)  
  
 insert() # Может потребоваться обновить данные на экране  
 entry.delete(0, tk.END)  
  
 except ValueError:  
 messagebox.showerror("Ошибка", "ID отдела и количество сотрудников должны быть числовыми значениями.")  
 except Exception as e:  
 conn.rollback()  
 error\_msg = f"Ошибка при обновлении данных: {e}"  
 print(error\_msg)  
 messagebox.showerror("Ошибка", error\_msg)  
  
 return entry, entry2  
  
 def on\_double\_click(event):  
 selected\_item = tree.selection()[0] # Получаем ID выбранной строки  
 item\_values = tree.item(selected\_item, "values") # Получаем значения выбранной строки  
 if item\_values:  
 entry.delete(0, tk.END)  
 entry.insert(0, item\_values[0]) # Пример: ID отдела в первое поле  
 entry2.delete(0, tk.END)  
 entry2.insert(0, item\_values[1]) # Пример: Название отдела во второе поле  
  
 tree.bind("<Double-1>", on\_double\_click)  
  
  
 def delete():  
 # удаление устаревших данных  
 tree.delete(\*tree.get\_children())  
  
  
 entry\_frame10 = tk.Frame(root)  
 entry\_frame10.pack(side='right')  
 # Создание кнопки "добавить строку"  
 btn2 = tk.Button(entry\_frame10, text="Add String", padx=5, pady=1, width=10, height=1, command=add\_string,  
 )  
 btn2.pack(side=tk.TOP, padx=10, pady=2)  
 # Создание кнопки "очистить строку"  
 btn4 = tk.Button(entry\_frame10, text="Clear String", padx=5, pady=1, width=10, height=1, command=confirm\_action,  
 )  
 btn4.pack(side=tk.TOP, padx=10, pady=2)  
 # Создание кнопки "изменить"  
 btn5 = tk.Button(entry\_frame10, text="Сhange", padx=5, pady=1, width=10, height=1, command=update\_string,  
 )  
 btn5.pack(side=tk.TOP, padx=10, pady=2)  
  
 entry\_frame11 = tk.Frame(root)  
 entry\_frame11.pack(side='right')  
 # Создание кнопки "импорт"  
 btn1 = tk.Button(entry\_frame11, text="Import Table", padx=5, pady=1, width=10, height=1, command=insert,  
 )  
 btn1.pack(side=tk.TOP, padx=10, pady=2)  
 # Создание кнопки "очистить"  
 btn3 = tk.Button(entry\_frame11, text="Clear Page", padx=5, pady=1, width=10, height=1, command=delete,  
 )  
 btn3.pack(side=tk.TOP, padx=10, pady=2)  
 # Создание кнопки "сохранить"  
 btn6 = tk.Button(entry\_frame11, text="Save", padx=5, pady=1, width=10, height=1, command=save\_to\_excel,  
 )  
 btn6.pack(side=tk.TOP, padx=10, pady=2)  
  
  
  
 # Создание строки для ввода новых данных  
 entry\_frame = tk.Frame(root)  
 entry\_frame.pack()  
  
 entry\_label = tk.Label(entry\_frame, text="Введите id:", width=23, height=3,anchor='w')  
 entry\_label.pack(side='left')  
  
 entry = tk.Entry(entry\_frame, width=30) # Создаем поле ввода шириной 30 символов  
 entry.pack(side='left') # Размещаем поле ввода на frame1  
  
 entry\_frame2 = tk.Frame(root)  
 entry\_frame2.pack()  
  
 entry\_label2 = tk.Label(entry\_frame2, text="Введите название:", width=23, height=2,  
 anchor='w')  
 entry\_label2.pack(side='left')  
  
 entry2 = tk.Entry(entry\_frame2, width=30) # Создаем поле ввода шириной 30 символов  
 entry2.pack(side='left') # Размещаем поле ввода на frame1  
  
  
 return entry, entry2 # Возвращаем созданное поле ввода  
  
def author():  
 # Создаем графический интерфейс  
 root3 = tk.Tk()  
 root3.title("Author")  
 root3.resizable(False, False) # Это запретит изменение размеров окна по ширине и по высоте  
  
 # Создание таблицы Отдел  
 tree = ttk.Treeview(root3)  
 tree["columns"] = ('ID', 'FIO', 'Email', 'Number phone')  
 tree.heading('#0', text='№')  
 tree.heading('ID', text='ID employees')  
 tree.heading('FIO', text='FIO')  
 tree.heading('Email', text='Email')  
 tree.heading('Number phone', text='Number phone')  
  
 # Устанавливаем размеры столбцов  
 tree.column("#0", width=40)  
 tree.column("ID", width=100, anchor=tk.CENTER)  
 tree.column("FIO", width=200, anchor=tk.CENTER)  
 tree.column("Email", width=200, anchor=tk.CENTER)  
 tree.column("Number phone", width=200, anchor=tk.CENTER)  
  
  
 tree.pack()  
  
 try:  
 tree.delete(\*tree.get\_children()) # удаление устаревших данных  
  
 cursor = conn.cursor()  
 cursor.execute("SELECT \* FROM \"Authors\"")  
 rows = cursor.fetchall()  
  
 for i, row in enumerate(rows):  
 tree.insert("", "end", text=str(i), values=row, )  
  
 except Exception as e:  
 print(f"Ошибка при загрузке данных: {e}")  
  
 # Функция для сортировки по отделу и вывода результата в отдельное окно  
  
 def save\_to\_excel():  
 try:  
 wb = openpyxl.Workbook()  
 ws = wb.active  
 ws.append([column[0] for column in cursor.description]) # Заголовки столбцов  
  
 for row\_data in rows:  
 ws.append(row\_data)  
  
 desktop\_path = os.path.join(os.path.join(os.environ['USERPROFILE']), 'Desktop')  
 timestamp = datetime.now().strftime("%Y-%m-%d\_%H-%M-%S")  
 save\_path = os.path.join(desktop\_path, f"employees\_data\_{timestamp}.xlsx")  
  
 if save\_path:  
 wb.save(save\_path)  
 print("Данные успешно сохранены в файл Excel!")  
 except Exception as e:  
 print(f"Ошибка при сохранении данных в Excel: {e}")  
  
 # Создание кнопок  
 def insert4():  
 try:  
 tree.delete(\*tree.get\_children()) # удаление устаревших данных  
  
 cursor = conn.cursor()  
 cursor.execute("SELECT \* FROM \"Authors\"")  
 rows = cursor.fetchall()  
  
 for i, row in enumerate(rows):  
 tree.insert("", "end", text=str(i), values=row,)  
  
 except Exception as e:  
 print(f"Ошибка при загрузке данных: {e}")  
  
 def add\_string():  
 id1 = entry.get()  
 fio = entry2.get()  
 email = entry3.get()  
 number = entry4.get()  
  
 try:  
 cursor = conn.cursor()  
 cursor.execute("INSERT INTO \"Authors\" (\"ida\", \"name\", \"email\",\"phone\") VALUES (%s, %s, %s, %s)",  
 (id1, fio, email, number))  
 conn.commit()  
  
 insert4() # Обновить вывод таблицы после добавления строки  
 entry.delete(0, tk.END)  
 entry2.delete(0, tk.END)  
 entry3.delete(0, tk.END)  
 entry4.delete(0, tk.END)  
  
  
 except Exception as e:  
 # Если произошла ошибка, откатываем транзакцию  
 conn.rollback()  
  
 error\_msg = f"Ошибка при загрузке данных: {e}"  
 print(error\_msg)  
 messagebox.showerror("Ошибка", error\_msg)  
  
 def on\_enter(event):  
 # Поместите здесь ваше действие  
 add\_string()  
  
 # Привязываем клавишу Enter к функции on\_enter  
 root3.bind('<Return>', on\_enter)  
  
 def delete\_selected\_employee():  
 try:  
 selected\_item = tree.selection()  
  
 for item in selected\_item:  
 values = tree.item(item, 'values')  
 ID\_employee = values[0]  
  
 cursor = conn.cursor()  
  
 # Начало транзакции  
 cursor.execute("START TRANSACTION;")  
  
 try:  
  
 # Удалить из accounting  
 cursor.execute("DELETE FROM \"Documents\" WHERE \"authorid\" = %s", (ID\_employee,))  
  
 # Удалить из employees последним  
 cursor.execute("DELETE FROM \"Authors\" WHERE \"ida\" = %s", (ID\_employee,))  
  
 # Применить все изменения  
 conn.commit()  
 except Exception as e:  
 conn.rollback()  
 error\_msg = f"Ошибка при удалении данных из одной из таблиц: {e}"  
 print(error\_msg)  
 messagebox.showerror("Ошибка", error\_msg)  
 return  
  
 insert4() # Обновить вывод таблицы после удаления строк  
 except Exception as e:  
 conn.rollback()  
 error\_msg = f"Ошибка при выполнении удаления из базы данных: {e}"  
 print(error\_msg)  
 messagebox.showerror("Ошибка", error\_msg)  
  
 def confirm\_action():  
 result = messagebox.askyesno("Confirmation", "Вы уверены, что хотите очистить?")  
 if result:  
 # Put your action here  
 delete\_selected\_employee()  
 else:  
 print("Action canceled.")  
  
 def update\_record():  
 try:  
 selected\_item = tree.selection()  
  
 if not selected\_item:  
 messagebox.showinfo("Информация", "Пожалуйста, выберите строку для изменения.")  
 return  
  
 id1 = entry.get()  
 fio = entry2.get()  
 email = entry3.get()  
 number = entry4.get()  
  
  
 if not id1 or not fio or not number or not email:  
 messagebox.showinfo("Информация", "Пожалуйста, заполните все поля для обновления.")  
 return  
  
 cursor = conn.cursor()  
  
 for item in selected\_item:  
 values = tree.item(item, 'values')  
 employee\_id = values[0]  
  
 # Начало транзакции  
 cursor.execute("START TRANSACTION;")  
  
 # Выполнение SQL-запроса для обновления строки по определенному идентификатору  
 cursor.execute(  
 "UPDATE \"Authors\" SET \"ida\" = %s, \"name\" = %s, \"email\" = %s, \"phone\" = %s",  
 (id1, fio, email, number))  
  
 conn.commit()  
 insert4() # Обновить вывод таблицы после обновления строки  
 print("Запись успешно обновлена в базе данных.")  
  
 except Exception as e:  
 conn.rollback()  
 error\_msg = f"Ошибка при обновлении строки в базе данных: {e}"  
 print(error\_msg)  
 messagebox.showerror("Ошибка", error\_msg)  
  
 # Очистка полей ввода после обновления  
 entry.delete(0, tk.END)  
 entry2.delete(0, tk.END)  
 entry4.delete(0, tk.END)  
  
  
 def on\_double\_click(event):  
 selected\_item = tree.selection()[0] # Получаем ID выбранной строки  
 item\_values = tree.item(selected\_item, "values") # Получаем значения выбранной строки  
 if item\_values:  
 entry.delete(0, tk.END)  
 entry.insert(0, item\_values[0]) # Пример: ID отдела в первое поле  
 entry2.delete(0, tk.END)  
 entry2.insert(0, item\_values[1]) # Пример: Название отдела во второе поле  
 entry3.delete(0, tk.END)  
 entry3.insert(0, item\_values[2]) # Пример: Название отдела во второе поле  
 entry4.delete(0, tk.END)  
 entry4.insert(0, item\_values[3]) # Пример: Название отдела во второе поле  
  
  
 tree.bind("<Double-1>", on\_double\_click)  
  
 def delete():  
 # удаление устаревших данных  
 tree.delete(\*tree.get\_children())  
  
 entry\_frame10 = tk.Frame(root3)  
 entry\_frame10.pack(side='right')  
 # Создание кнопки "добавить строку"  
 btn2 = tk.Button(entry\_frame10, text="Add String", padx=5, pady=1, width=10, height=1, command=add\_string)  
 btn2.pack(side=tk.TOP, padx=60, pady=2)  
 # Создание кнопки "очистить строку"  
 btn4 = tk.Button(entry\_frame10, text="Clear String", padx=5, pady=1, width=10, height=1, command=confirm\_action)  
 btn4.pack(side=tk.TOP, padx=25, pady=2)  
 # Создание кнопки "изменить"  
 btn5 = tk.Button(entry\_frame10, text="Сhange", padx=5, pady=1, width=10, height=1, command=update\_record)  
 btn5.pack(side=tk.TOP, padx=25, pady=2)  
  
 entry\_frame11 = tk.Frame(root3)  
 entry\_frame11.pack(side='right')  
 # Создание кнопки "импорт"  
 btn1 = tk.Button(entry\_frame11, text="Import Table", padx=5, pady=1, width=10, height=1, command=insert4)  
 btn1.pack(side=tk.TOP, padx=60, pady=2)  
 # Создание кнопки "очистить"  
 btn3 = tk.Button(entry\_frame11, text="Clear Page", padx=5, pady=1, width=10, height=1, command=delete)  
 btn3.pack(side=tk.TOP, padx=25, pady=2)  
 # Создание кнопки "сохранить"  
 btn6 = tk.Button(entry\_frame11, text="Save", padx=5, pady=1, width=10, height=1, command=save\_to\_excel)  
 btn6.pack(side=tk.TOP, padx=25, pady=2)  
  
  
 entry\_frame12 = tk.Frame(root3)  
 entry\_frame12.pack(side='right')  
  
 entry\_frame1 = tk.Frame(root3)  
 entry\_frame1.pack()  
  
 entry\_label = tk.Label(entry\_frame1, text="Введите id:", width=30, height=2,  
 anchor='w')  
 entry\_label.pack(side='left')  
  
 entry = tk.Entry(entry\_frame1, width=30) # Создаем поле ввода шириной 30 символов  
 entry.pack(side='left') # Размещаем поле ввода на frame1  
  
 entry\_frame2 = tk.Frame(root3)  
 entry\_frame2.pack()  
  
 entry\_label2 = tk.Label(entry\_frame2, text="Введите ФИО:", width=30, height=1,  
 anchor='w')  
 entry\_label2.pack(side='left')  
  
 entry2 = tk.Entry(entry\_frame2, width=30) # Создаем поле ввода шириной 30 символов  
 entry2.pack(side='left') # Размещаем поле ввода на frame1  
  
 entry\_frame3 = tk.Frame(root3)  
 entry\_frame3.pack()  
  
 entry\_label3 = tk.Label(entry\_frame3, text="Введите почту:", width=30, height=2,  
 anchor='w')  
 entry\_label3.pack(side='left')  
  
 entry3 = tk.Entry(entry\_frame3, width=30) # Создаем поле ввода шириной 30 символов  
 entry3.pack(side='left') # Размещаем поле ввода на frame1  
  
 entry\_frame4 = tk.Frame(root3)  
 entry\_frame4.pack()  
  
 entry\_label4 = tk.Label(entry\_frame4, text="Введите номер телефона:", width=30, height=1,  
 anchor='w')  
 entry\_label4.pack(side='left')  
  
 entry4 = tk.Entry(entry\_frame4, width=30) # Создаем поле ввода шириной 30 символов  
 entry4.pack(side='left') # Размещаем поле ввода на frame1  
  
  
def document():  
 # Создаем графический интерфейс  
 root3 = tk.Tk()  
 root3.title("Employees")  
 root3.resizable(False, False) # Это запретит изменение размеров окна по ширине и по высоте  
  
 # Создание таблицы Отдел  
 tree = ttk.Treeview(root3)  
 tree["columns"] = ('ID', 'title', 'content', 'category', 'autor', 'date')  
 tree.heading('#0', text='№')  
 tree.heading('ID', text='ID')  
 tree.heading('title', text='Название')  
 tree.heading('content', text='Контент')  
 tree.heading('category', text='Категория')  
 tree.heading('autor', text='Автор')  
 tree.heading('date', text='Дата')  
  
 # Устанавливаем размеры столбцов  
 tree.column("#0", width=40)  
 tree.column("ID", width=100, anchor=tk.CENTER)  
 tree.column("title", width=170, anchor=tk.CENTER)  
 tree.column("content", width=100, anchor=tk.CENTER)  
 tree.column("category", width=65, anchor=tk.CENTER)  
 tree.column("autor", width=55, anchor=tk.CENTER)  
 tree.column("date", width=100, anchor=tk.CENTER)  
  
 tree.pack()  
  
 try:  
 tree.delete(\*tree.get\_children()) # удаление устаревших данных  
  
 cursor = conn.cursor()  
 cursor.execute("SELECT \* FROM \"Documents\"")  
 rows = cursor.fetchall()  
  
 for i, row in enumerate(rows):  
 tree.insert("", "end", text=str(i), values=row, )  
  
 except Exception as e:  
 print(f"Ошибка при загрузке данных: {e}")  
  
 # Функция для сортировки по отделу и вывода результата в отдельное окно  
 def sort\_by\_department():  
 selected\_department\_id = sort\_entry.get()  
  
 sorted\_employees = []  
 for child in tree.get\_children():  
 values = tree.item(child, 'values')  
 if values and values[4] == selected\_department\_id:  
 sorted\_employees.append(values)  
  
 # Создание нового окна для вывода отсортированных данных  
 result\_window = tk.Toplevel(root3)  
 result\_window.title("Sorted Document")  
 result\_window.resizable(False, False) # Это запретит изменение размеров окна по ширине и по высоте  
  
 result\_tree = ttk.Treeview(result\_window)  
 result\_tree["columns"] = ('ID', 'title', 'content', 'category', 'autor', 'date')  
  
 # Создание колонок таблицы  
 for i, column in enumerate(('ID', 'title', 'content', 'category', 'autor', 'date')):  
 result\_tree.heading(i, text=column)  
 result\_tree.column(i, width=180, minwidth=100, anchor=tk.CENTER, stretch=True)  
  
 for idx, employee in enumerate(sorted\_employees):  
 result\_tree.insert("", "end", text=str(idx), values=employee)  
  
 result\_tree.pack()  
  
 # Функция сохранения результатов сортировки  
 def save\_to\_excel():  
 wb = Workbook()  
 ws = wb.active  
 for i, column in enumerate(('ID', 'title', 'content', 'category', 'autor', 'date')):  
 ws.cell(row=1, column=i + 1, value=column)  
  
 for idx, employee in enumerate(sorted\_employees):  
 for i, value in enumerate(employee):  
 ws.cell(row=idx + 2, column=i + 1, value=value)  
  
 desktop\_path = os.path.join(os.path.join(os.environ['USERPROFILE']), 'Desktop')  
 timestamp = datetime.now().strftime("%Y-%m-%d\_%H-%M-%S")  
 save\_path = os.path.join(desktop\_path, f"employees\_data\_{timestamp}.xlsx")  
  
 wb.save(save\_path)  
 print("Данные успешно сохранены в файле employees\_data.xlsx. на рабочий стол")  
  
 # Создание кнопки "Сохранить"  
 save\_button = tk.Button(result\_window, text="Save", command=save\_to\_excel, width=10, height=1)  
 save\_button.pack()  
  
 def save\_to\_excel():  
 try:  
 wb = openpyxl.Workbook()  
 ws = wb.active  
 ws.append([column[0] for column in cursor.description]) # Заголовки столбцов  
  
 for row\_data in rows:  
 ws.append(row\_data)  
  
 desktop\_path = os.path.join(os.path.join(os.environ['USERPROFILE']), 'Desktop')  
 timestamp = datetime.now().strftime("%Y-%m-%d\_%H-%M-%S")  
 save\_path = os.path.join(desktop\_path, f"employees\_data\_{timestamp}.xlsx")  
  
 if save\_path:  
 wb.save(save\_path)  
 print("Данные успешно сохранены в файл Excel!")  
 except Exception as e:  
 print(f"Ошибка при сохранении данных в Excel: {e}")  
  
 # Создание кнопок  
 def insert4():  
 try:  
 tree.delete(\*tree.get\_children()) # удаление устаревших данных  
  
 cursor = conn.cursor()  
 cursor.execute("SELECT \* FROM \"Documents\"")  
 rows = cursor.fetchall()  
  
 for i, row in enumerate(rows):  
 tree.insert("", "end", text=str(i), values=row,)  
  
 except Exception as e:  
 print(f"Ошибка при загрузке данных: {e}")  
  
 def add\_string():  
 id1 = entry.get()  
 tit = entry2.get()  
 cat = entry4.get()  
 cont = entry5.get()  
 auth = entry6.get()  
 date = entry7.get()  
  
 try:  
 cursor = conn.cursor()  
 cursor.execute("INSERT INTO \"Documents\" (\"idd\", \"title\", \"content\",\"categoryid\", \"authorid\", \"date\") VALUES (%s, %s, %s, %s, %s, %s)",  
 (id1, tit, cat, cont, auth, date))  
 conn.commit()  
  
 insert4() # Обновить вывод таблицы после добавления строки  
 entry.delete(0, tk.END)  
 entry2.delete(0, tk.END)  
 entry4.delete(0, tk.END)  
 entry5.delete(0, tk.END)  
 entry6.delete(0, tk.END)  
 entry7.delete(0, tk.END)  
  
 except Exception as e:  
 # Если произошла ошибка, откатываем транзакцию  
 conn.rollback()  
  
 error\_msg = f"Ошибка при загрузке данных: {e}"  
 print(error\_msg)  
 messagebox.showerror("Ошибка", error\_msg)  
  
 def on\_enter(event):  
 # Поместите здесь ваше действие  
 add\_string()  
  
 # Привязываем клавишу Enter к функции on\_enter  
 root3.bind('<Return>', on\_enter)  
  
 def delete\_selected\_employee():  
 try:  
 selected\_item = tree.selection()  
  
 for item in selected\_item:  
 values = tree.item(item, 'values')  
 ID\_doc = values[0]  
  
 cursor = conn.cursor()  
  
 # Начало транзакции  
 cursor.execute("START TRANSACTION;")  
  
 try:  
  
 # Удалить из employees последним  
 cursor.execute("DELETE FROM \"Documents\" WHERE \"idd\" = %s", (ID\_doc,))  
  
 # Применить все изменения  
 conn.commit()  
 except Exception as e:  
 conn.rollback()  
 error\_msg = f"Ошибка при удалении данных из одной из таблиц: {e}"  
 print(error\_msg)  
 messagebox.showerror("Ошибка", error\_msg)  
 return  
  
 insert4() # Обновить вывод таблицы после удаления строк  
 except Exception as e:  
 conn.rollback()  
 error\_msg = f"Ошибка при выполнении удаления из базы данных: {e}"  
 print(error\_msg)  
 messagebox.showerror("Ошибка", error\_msg)  
  
 def confirm\_action():  
 result = messagebox.askyesno("Confirmation", "Вы уверены, что хотите очистить?")  
 if result:  
 # Put your action here  
 delete\_selected\_employee()  
 else:  
 print("Action canceled.")  
  
 def update\_record():  
 try:  
 selected\_item = tree.selection()  
  
 if not selected\_item:  
 messagebox.showinfo("Информация", "Пожалуйста, выберите строку для изменения.")  
 return  
  
 id1 = entry.get()  
 tit = entry2.get()  
 cat = entry4.get()  
 cont = entry5.get()  
 auth = entry6.get()  
 date = entry7.get()  
  
 if not id1 or not tit or not cat or not cont or not date:  
 messagebox.showinfo("Информация", "Пожалуйста, заполните все поля для обновления.")  
 return  
  
 cursor = conn.cursor()  
  
 for item in selected\_item:  
 values = tree.item(item, 'values')  
 employee\_id = values[0]  
  
 # Начало транзакции  
 cursor.execute("START TRANSACTION;")  
  
 # Выполнение SQL-запроса для обновления строки по определенному идентификатору  
 cursor.execute(  
 "UPDATE \"Documents\" SET \"idd\" = %s, \"title\" = %s, \"content\" = %s,\"categoryid\" = %s, \"authorid\" = %s, \"date\" = %s",  
 (id1, tit, cat, cont, auth, date))  
  
 conn.commit()  
 insert4() # Обновить вывод таблицы после обновления строки  
 print("Запись успешно обновлена в базе данных.")  
  
 except Exception as e:  
 conn.rollback()  
 error\_msg = f"Ошибка при обновлении строки в базе данных: {e}"  
 print(error\_msg)  
 messagebox.showerror("Ошибка", error\_msg)  
  
 # Очистка полей ввода после обновления  
 entry.delete(0, tk.END)  
 entry2.delete(0, tk.END)  
 entry4.delete(0, tk.END)  
 entry5.delete(0, tk.END)  
 entry6.delete(0, tk.END)  
 entry7.delete(0, tk.END)  
  
 def on\_double\_click(event):  
 selected\_item = tree.selection()[0] # Получаем ID выбранной строки  
 item\_values = tree.item(selected\_item, "values") # Получаем значения выбранной строки  
 if item\_values:  
 entry.delete(0, tk.END)  
 entry.insert(0, item\_values[0]) # Пример: ID отдела в первое поле  
 entry2.delete(0, tk.END)  
 entry2.insert(0, item\_values[1]) # Пример: Название отдела во второе поле  
 entry4.delete(0, tk.END)  
 entry4.insert(0, item\_values[2]) # Пример: Название отдела во второе поле  
 entry5.delete(0, tk.END)  
 entry5.insert(0, item\_values[3]) # Пример: ID отдела в первое поле  
 entry6.delete(0, tk.END)  
 entry6.insert(0, item\_values[4]) # Пример: Название отдела во второе поле  
 entry7.delete(0, tk.END)  
 entry7.insert(0, item\_values[5]) # Пример: Название отдела во второе поле  
  
 tree.bind("<Double-1>", on\_double\_click)  
  
 def delete():  
 # удаление устаревших данных  
 tree.delete(\*tree.get\_children())  
  
 entry\_frame10 = tk.Frame(root3)  
 entry\_frame10.pack(side='right')  
 # Создание кнопки "добавить строку"  
 btn2 = tk.Button(entry\_frame10, text="Add String", padx=5, pady=1, width=10, height=1, command=add\_string)  
 btn2.pack(side=tk.TOP, padx=60, pady=2)  
 # Создание кнопки "очистить строку"  
 btn4 = tk.Button(entry\_frame10, text="Clear String", padx=5, pady=1, width=10, height=1, command=confirm\_action)  
 btn4.pack(side=tk.TOP, padx=25, pady=2)  
 # Создание кнопки "изменить"  
 btn5 = tk.Button(entry\_frame10, text="Сhange", padx=5, pady=1, width=10, height=1, command=update\_record)  
 btn5.pack(side=tk.TOP, padx=25, pady=2)  
  
 entry\_frame11 = tk.Frame(root3)  
 entry\_frame11.pack(side='right')  
 # Создание кнопки "импорт"  
 btn1 = tk.Button(entry\_frame11, text="Import Table", padx=5, pady=1, width=10, height=1, command=insert4)  
 btn1.pack(side=tk.TOP, padx=60, pady=2)  
 # Создание кнопки "очистить"  
 btn3 = tk.Button(entry\_frame11, text="Clear Page", padx=5, pady=1, width=10, height=1, command=delete)  
 btn3.pack(side=tk.TOP, padx=25, pady=2)  
 # Создание кнопки "сохранить"  
 btn6 = tk.Button(entry\_frame11, text="Save", padx=5, pady=1, width=10, height=1, command=save\_to\_excel)  
 btn6.pack(side=tk.TOP, padx=25, pady=2)  
 # Создание кнопки "сортировать"  
 btn7 = tk.Button(entry\_frame11, text="Sort", padx=5, pady=1, width=10, height=1, command=sort\_by\_department)  
 btn7.pack(side=tk.TOP, padx=25, pady=2)  
  
 entry\_frame12 = tk.Frame(root3)  
 entry\_frame12.pack(side='right')  
  
 entry\_frame1 = tk.Frame(root3)  
 entry\_frame1.pack()  
  
 entry\_label = tk.Label(entry\_frame1, text="Введите id:", width=30, height=2,  
 anchor='w')  
 entry\_label.pack(side='left')  
  
 entry = tk.Entry(entry\_frame1, width=30) # Создаем поле ввода шириной 30 символов  
 entry.pack(side='left') # Размещаем поле ввода на frame1  
  
 entry\_frame2 = tk.Frame(root3)  
 entry\_frame2.pack()  
  
 entry\_label2 = tk.Label(entry\_frame2, text="Введите название:", width=30, height=1,  
 anchor='w')  
 entry\_label2.pack(side='left')  
  
 entry2 = tk.Entry(entry\_frame2, width=30) # Создаем поле ввода шириной 30 символов  
 entry2.pack(side='left') # Размещаем поле ввода на frame1  
  
 entry\_frame4 = tk.Frame(root3)  
 entry\_frame4.pack()  
  
 entry\_label4 = tk.Label(entry\_frame4, text="Введите контент:", width=30, height=1,  
 anchor='w')  
 entry\_label4.pack(side='left')  
  
 entry4 = tk.Entry(entry\_frame4, width=30) # Создаем поле ввода шириной 30 символов  
 entry4.pack(side='left') # Размещаем поле ввода на frame1  
  
 entry\_frame5 = tk.Frame(root3)  
 entry\_frame5.pack()  
  
 entry\_label5 = tk.Label(entry\_frame5, text="Введите категорию:", width=30, height=2,  
 anchor='w')  
 entry\_label5.pack(side='left')  
  
 entry5 = tk.Entry(entry\_frame5, width=30) # Создаем поле ввода шириной 30 символов  
 entry5.pack(side='left') # Размещаем поле ввода на frame1  
  
 entry\_frame6 = tk.Frame(root3)  
 entry\_frame6.pack()  
  
 entry\_label6 = tk.Label(entry\_frame6, text="Введите автора:", width=30, height=1,  
 anchor='w')  
 entry\_label6.pack(side='left')  
  
 entry6 = tk.Entry(entry\_frame6, width=30) # Создаем поле ввода шириной 30 символов  
 entry6.pack(side='left') # Размещаем поле ввода на frame1  
  
 entry\_frame7 = tk.Frame(root3)  
 entry\_frame7.pack()  
  
 entry\_label7 = tk.Label(entry\_frame7, text="Введите дату:", width=30, height=1,  
 anchor='w')  
 entry\_label7.pack(side='left')  
  
 entry7 = tk.Entry(entry\_frame7, width=30) # Создаем поле ввода шириной 30 символов  
 entry7.pack(side='left') # Размещаем поле ввода на frame1  
  
 # Создание поля для сортировки  
  
 sort\_entry\_frame = tk.Frame(root3)  
 sort\_entry\_frame.pack()  
  
 sort\_label = tk.Label(sort\_entry\_frame, text="Введите ID автора:", width=30, height=2,  
 anchor='w')  
 sort\_label.pack(side='left')  
  
 sort\_entry = tk.Entry(sort\_entry\_frame, width=30, background="light gray")  
 sort\_entry.pack(side='left')  
  
 return entry, entry2, entry4, entry5, entry6, sort\_entry # Возвращаем созданное поле ввода  
  
def info():  
  
  
 # Создаем графический интерфейс  
 root2 = tk.Tk()  
 root2.title("Information")  
 root2.resizable(False, False) # Это запретит изменение размеров окна по ширине и по высоте  
  
 # Создание таблиц  
  
 # Создание таблицы Отдел  
 tree = ttk.Treeview(root2)  
 tree["columns"] = ('tit', 'cat', 'aut')  
 tree.heading('#0', text='№')  
 tree.heading('tit', text='FIO')  
 tree.heading('cat', text='Department')  
 tree.heading('aut', text='Number')  
  
  
 # Устанавливаем размеры столбцов  
 tree.column("#0", width=40)  
 tree.column("tit", width=200, anchor=tk.CENTER)  
 tree.column("cat", width=200, anchor=tk.CENTER)  
 tree.column("aut", width=200, anchor=tk.CENTER)  
  
 tree.pack()  
  
 try:  
 tree.delete(\*tree.get\_children()) # удаление устаревших данных  
  
 cursor = conn.cursor()  
 cursor.execute("SELECT \"Documents\".\"title\", \"Category\".\"category\", \"Authors\".\"name\""  
 "FROM \"Documents\" "  
 "JOIN \"Category\" ON \"Category\".\"idc\" = \"Documents\".\"categoryid\""  
 "JOIN \"Authors\" ON \"Documents\".\"authorid\" = \"Authors\".\"ida\";")  
 rows = cursor.fetchall()  
  
 for i, row in enumerate(rows):  
 tree.insert("", "end", text=str(i), values=row, )  
  
 except Exception as e:  
 print(f"Ошибка при загрузке данных: {e}")  
 # Создание кнопок  
  
# Создание графического интерфейса  
def data():  
  
 def close():  
 boom.destroy()  
  
 boom = tk.Tk()  
 boom.title("Data Base")  
 boom.geometry("450x550")  
 boom.resizable(False, False) # Это запретит изменение размеров окна по ширине и по высоте  
  
  
 label1 = tk.Label(boom, text="Kalonov data base", font=('Arial', 12, 'normal'), foreground="gray")  
 label1.pack(pady=5)  
  
 colon\_l = tk.Frame(boom, padx=50, pady=1)  
 colon\_l.pack()  
  
 # Создание и размещение кнопки для открытия окна с данными  
 open\_button = tk.Button(colon\_l, text="Author", command=author, width=20, height=2, font=('Arial', 20, 'normal'), borderwidth="6")  
 open\_button.pack()  
 open\_button = tk.Button(colon\_l, text="Category", command=category, width=20, height=2, font=('Arial', 20, 'normal'), borderwidth="6")  
 open\_button.pack()  
 open\_button = tk.Button(colon\_l, text="Document", command=document, width=20, height=2, font=('Arial', 20, 'normal'), borderwidth="6")  
 open\_button.pack()  
 open\_button = tk.Button(colon\_l, text="Information", command=info, width=20, height=2, font=('Arial', 20, 'normal'), borderwidth="6")  
 open\_button.pack()  
 open\_button = tk.Button(colon\_l, text="Exit", command=close, width=20, height=2, font=('Arial', 20, 'normal'), borderwidth="6")  
 open\_button.pack()  
  
# Функция для сохранения пользователей в файл  
def save\_users():  
 with open('users.pickle', 'wb') as file:  
 pickle.dump(users, file)  
  
# Загрузка пользователей из файла или создание нового словаря, если файла нет  
try:  
 with open('users.pickle', 'rb') as file:  
 users = pickle.load(file)  
except FileNotFoundError:  
 users = {"admin": "password"}  
  
def reg():  
 def register():  
 username = entry\_username.get()  
 password = entry\_password.get()  
  
 # Регистрация нового пользователя  
 if username not in users:  
 users[username] = password  
 save\_users() # Сохраняем пользователей после добавления нового  
 label\_status.config(text="Пользователь зарегистрирован.")  
 else:  
 label\_status.config(text="Пользователь уже существует.")  
  
 root8 = tk.Tk()  
 root8.title("Регистрация")  
 root8.geometry("200x170")  
  
 label\_username = tk.Label(root8, text="")  
 label\_username.pack()  
  
 label\_username = tk.Label(root8, text="Имя пользователя:")  
 label\_username.pack()  
 entry\_username = tk.Entry(root8)  
 entry\_username.pack()  
  
 label\_password = tk.Label(root8, text="Пароль:")  
 label\_password.pack()  
 entry\_password = tk.Entry(root8, show="\*")  
 entry\_password.pack()  
  
 button\_register = tk.Button(root8, text="Зарегистрироваться", command=register, foreground="black",  
 borderwidth="2")  
 button\_register.pack(pady=10)  
  
 label\_status = tk.Label(root8, text="")  
 label\_status.pack()  
  
 root8.mainloop()  
  
  
def log():  
 def login():  
 username = entry\_username.get()  
 password = entry\_password.get()  
  
 # Проверка логина и пароля  
 if username in users and users[username] == password:  
 data()  
 root7.destroy()  
 entrance.destroy()  
 else:  
 label\_status.config(text="Неверный логин или пароль.")  
  
 root7 = tk.Tk()  
 root7.title("Авторизация")  
 root7.geometry("200x170")  
  
 label\_username = tk.Label(root7, text="")  
 label\_username.pack()  
  
 label\_username = tk.Label(root7, text="Имя пользователя:")  
 label\_username.pack()  
 entry\_username = tk.Entry(root7)  
 entry\_username.pack()  
  
 label\_password = tk.Label(root7, text="Пароль:")  
 label\_password.pack()  
 entry\_password = tk.Entry(root7, show="\*")  
 entry\_password.pack()  
  
 button\_login = tk.Button(root7, text="Войти", command=login, foreground="black", borderwidth="2")  
 button\_login.pack(pady=10)  
  
 label\_status = tk.Label(root7, text="")  
 label\_status.pack()  
  
 root7.mainloop()  
  
def close\_add\_ed():  
 entrance.destroy()  
  
# Начало работы, то, что с самого начал выводится на экран:  
  
  
# Создание главного окна  
  
entrance = tk.Tk()  
entrance.title("Entrance")  
entrance.geometry("300x330")  
entrance.resizable(False, False)  
  
label = tk.Label(entrance, text=" data base ", font=('Arial', 14, 'normal'))  
label.pack(pady=10)  
  
login1=tk.Button(entrance, text ="Login", width=10, height=1, foreground="black",  
 borderwidth="8", font=('Verdana', 20, 'normal'), cursor="hand2", command=log)  
login1.pack(pady=10)  
  
register1=tk.Button(entrance, text ="Registration", width=10, height=1, foreground="black",  
 borderwidth="8", font=('Verdana', 20, 'normal'), cursor="hand2", command=reg)  
register1.pack(pady=10)  
  
exit1=tk.Button(entrance, text ="Exit", width=10, height=1, foreground="black",  
 borderwidth="8", font=('Verdana', 20, 'normal'), cursor="hand2", command=close\_add\_ed)  
exit1.pack(pady=10)  
  
entrance.mainloop()

**СКРИПТ SQL**

-- This script was generated by the ERD tool in pgAdmin 4.

-- Please log an issue at https://github.com/pgadmin-org/pgadmin4/issues/new/choose if you find any bugs, including reproduction steps.

BEGIN;

CREATE TABLE IF NOT EXISTS public."Documents"

(

idd integer NOT NULL,

title character(100) NOT NULL,

content character(100) NOT NULL,

categoryid integer NOT NULL,

authorid integer NOT NULL,

date character(100) NOT NULL,

PRIMARY KEY (idd)

);

CREATE TABLE IF NOT EXISTS public."Authors"

(

ida integer NOT NULL,

name character(100) NOT NULL,

email character(100) NOT NULL,

phone character(100) NOT NULL,

PRIMARY KEY (ida)

);

CREATE TABLE IF NOT EXISTS public."Category"

(

idc integer NOT NULL,

category character(100) NOT NULL,

PRIMARY KEY (idc)

);

ALTER TABLE IF EXISTS public."Documents"

ADD CONSTRAINT id1 FOREIGN KEY (categoryid)

REFERENCES public."Category" (idc) MATCH FULL

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public."Documents"

ADD CONSTRAINT id2 FOREIGN KEY (authorid)

REFERENCES public."Authors" (ida) MATCH FULL

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

END;