数据库基础及应用课程

大作业

项目名称： 学校信息管理系统

完成人： 汪洋

完成日期： 2023 年 5 月 3 日

指导教师： 赵英良

目录

1 软件功能

1.1 学生信息管理

1. **添加学生信息**：允许教务处录入学生学籍信息。适用于入学季新生入籍、交换生学籍交接、转学申请等。
2. **查询学生信息**：允许用户根据学生的某一项信息或某几项信息，查询相应的学生信息。适用于活动信息统计、中学反馈等。
3. **更新学生信息**：允许用户根据学生的某一项信息或某几项信息，更新相应的学生信息。适用于学籍修改、转专业、转班级等。
4. **删除学生信息**：允许教务处根据学生的某一项信息或某几项信息，删除相应的学生信息。适用于学生转学、退学等。

1.2 课程信息管理

1. **添加课程信息**：允许教务处录入课程信息，将这些信息存储到数据库中。适用于学院开课等。
2. **查询课程信息**：允许用户根据课程的某一项信息或某几项信息，查询相应的课程信息。适用于学生选课参考、教务处安排课表等。
3. **更新课程信息**：允许用户根据课程的某一项信息或某几项信息，更新相应的课程信息。适用于教师调换、学分修改、先修课程修改等。
4. **删除课程信息**：允许用户根据课程的某一项信息或某几项信息，删除相应的课程信息。适用于教师退休、课程停开等。

1.3 选课信息管理

1. **添加选课信息**：允许学生选课。适用于学生选课等。
2. **查询选课信息**：允许学生查看他们已经选择的课程，教师查看选择他们课程的学生。适用于学生上课统计、教师考勤管理等。
3. **更新选课信息**：允许学生修改他们已经选择的课程。适用于学生改选课等。
4. **删除选课信息**：允许学生取消他们已经选择的课程。适用于学生退选课等。

1.4 成绩管理

1. **添加成绩信息**：允许教师录入学生成绩。适用于教师登记学生成绩。
2. **查询成绩信息**：允许用户根据选课的某一项信息或某几项信息，查询学生成绩、查询课程成绩。适用于学生、教师查成绩。
3. **更新成绩信息**：允许用户根据选课的某一项信息或某几项信息，更新相应的成绩信息。适用于教师修改成绩。
4. **删除成绩信息**：允许用户根据选课的某一项信息或某几项信息，删除相应的成绩信息。适用于因特殊原因取消学生考试成绩。
5. **成绩信息分析**：
   (1) 学生成绩分析：允许用户根据学号、学年、学期，计算学生最高分、最低分、平均分（考虑课程学分权重）、绩点（最高 4.3）、优秀率（90 分以上）、及格率（60 分以上）、不及格率（60 分以下）、优秀门数（90 分以上）、及格门数（60 分以上）、不及格门数（60 分以下）、总门数。
   (2) 课程成绩分析：允许根据课程号、学年、学期，计算课程最高分、最低分、平均分、优秀率（90 分以上）、及格率（60 分以上）、不及格率（60 分以下）、优秀人数（90 分以上）、及格人数（60 分以上）、不及格人数（60 分以下）、总人数。
   注：绩点对应：95-100 分为 4.3，90-95 分为 4.0，85-90 分为 3.7，81-85 分为 3.3，78-81 分为 3.0，

75-78 分为 2.7，72-75 分为 2.3，68-72 分为 2.0，64-68 分为 1.7，60-64 分为 1.3，小于 60 分为 0，GPA 采取平均学分绩计算方法。

2 软件的运行环境

1. Python3
2. 华为 ECS，openEuler OS
3. openGauss(Sqlite)：DB
4. putty：远程连接工具
5. Python 第三方库：
   (1) tkinter：GUI
   (2) messagebox：消息窗口库
   (3) tkfont：字体库
   (4) ttk：下拉菜单库
   (5) scrolledtext：滚动文本库
   (6) datetime：日期函数库
   (7) sqlite3：连接 sqlite3 的库
   (8) psycopg2：连接 opengauss 的库

3 环境搭建说明

1. 安装 Python3：https://www.python.org/。
2. 安装 Putty：https://www.putty.be/。
3. 申请华为云 ECS，安装 OpenGauss 数据库：https://www.huaweicloud.com/。
4. 完成 Putty 远程连接 OpenGauss 数据库的相关配置（详见文档）。
5. 完成 Python 连接 OpenGauss 数据库的相关设置。
   (1) 设置防火墙安全规则：控制台 → 弹性云服务器 → 安全组 → 入方向规则 → 添加规则：优先级：10，策略：允许，协议端口：TCP、26000，类型：IPv4，源地址：0.0.0.0/0。
   (2) 设置 DB 加密方式和监听 IP
   修改密码加密方式

```
[omm@ecs -c9bf ~]$ gs_guc set -I all -c "password_encryption_type=1"
```

   修改 DB 监听地址

```
[omm@ecs -c9bf ~]$ vi /gaussdb/data/db1/postgresql.conf
listen_addresses='10.0.0.15'
local_bind_address ='10.0.0.15'
#修改为
listen_addresses='*'
local_bind_address='*'
```

   修改登录主机信任方式（允许远程访问，在 IPV4 下面）

```
vi /gaussdb/data/db1/pg_hba.conf
#增加
host all all 192.168.56.1/24 md5 #虚拟机使用
host all all 0.0.0.0/0 md5 #ECS使用
```

6. 在 OpenGauss 中设计数据库。
7. 在 Python 中设计程序。

4 软件设计说明

4.1 数据库设计说明

数据库名称：dbapp。
数据库用户名：xjtudba，密码：xjtu@123。(可以使用 sqlite3, 不需密码)
数据表名称：student、course、sc。
数据表结构：

列名	说明	数据类型	约束
sid	学号	字符串，20	主键
sname	姓名	字符串，40	非空
gender	性别	字符串，10	男或女
age	年龄	int	
sclass	班级	字符串，20	
school	学院	字符串，40	
birthday	生日	date	
province	籍贯	字符串，40	

表 1: student

列名	说明	数据类型	约束
cid	课程号	字符串，20	主键
cname	课程名	字符串，40	
teacher	教师	字符串，20	
credit	学分	float	
cpid	先修课程	字符串，20	

表 2: course

列名	说明	数据类型	约束
sid	学号	字符串，20	联合主键，外键，引用 student 表 sid
cid	课程号	字符串，20	联合主键，外键，引用 course 表 cid
year	学年	字符串，10	
term	学期	字符串，20	第一学期、第二学期或小学期
grade	成绩	int	

表 3: sc

数据库设计代码：

```
[omm@ecs-5138 ~]$ gs_om -t start
[omm@ecs-5138 ~]$ gsql -p 26000 -d postgres -U omm -r
postgres=# create user xjtudba with password 'xjtu@123';
postgres=# create database dbapp owner xjtudba;
postgres=# \c dbapp
dbapp=# create schema xjtudba authorization xjtudba;
dbapp=# set search_path to xjtudba;

```

```
dbapp=# create table student (
sid varchar(20) primary key,
sname varchar(40) not null,
gender varchar(10), check (gender = '男' or gender = '女'),
age int,
sclass varchar(20),
school varchar(40),
birthday date,
province varchar(40));
dbapp=# create table course (
cid varchar(20) primary key,
cname varchar(40),
teacher varchar(20),
credit float,
cpid varchar(20));
dbapp=# create table sc (
sid varchar(20),
cid varchar(20),
year varchar(10),
term varchar(20), check (term = '第一学期' or term = '第二学期' or term = '小学期'),
grade int,
primary key (sid, cid),
foreign key (sid) references student(sid),
foreign key (cid) references course(cid));
```

4.2 函数设计说明

```
import tkinter as tk
from tkinter import messagebox #消息窗口库
import tkinter.font as tkfont #字体库
from tkinter import ttk #下拉菜单库
from tkinter import scrolledtext #滚动文本库
import datetime #日期函数库
#import sqlite3 #连接sqlite3的库
import psycopg2 #连接opengauss的库

#连接数据库(OpenGauss)
def connect_database():
    con = psycopg2.connect(
        host='123.249.97.41',
        port=26000,
        user='xjtudba',
        password='xjtu@123',
        database='dbapp')
    cur = con.cursor();
    return con, cur
#连接数据库(Sqlite3)
#def connect_database():
    con = sqlite3.connect("addressbook.db")
    cur=con.cursor()
    return con,cur

#显示查询结果
def show_result(result):
    result_window = tk.Toplevel(root)
    result_window.geometry('800x400+250+150')
    text = scrolledtext.ScrolledText(result_window, width=800, height=400)
    for row in result:
        text.insert(tk.END, '  '.join(map(str, row)) + '\n') #将每一行元素转换为字符串，用空格连接，最后加上换行符
    text.pack()
```

```
    text.config(state=tk.DISABLED) #使用config方法将text部件设置为只读
    return

#添加学生信息
def add_student_window():
    add_student_window = tk.Toplevel(root)
    add_student_window.geometry('300x500+550+150')
    add_student_window.title('添加学生信息')
    def add_student():
        #获取用户输入的学生信息
        sid = sid_entry.get()
        sname = sname_entry.get()
        gender = gender_cmb.get()
        age = age_entry.get()
        sclass = sclass_entry.get()
        school = school_entry.get()
        birthday = birthday_entry.get()
        province = province_entry.get()
        #检查学生信息有效性
        if not sid:
            tk.messagebox.showerror('错误', '学号不能为空！')
            return
        if not sname:
            tk.messagebox.showerror('错误', '姓名不能为空！')
            return
        try:
            age = int(age)
        except ValueError:
            tk.messagebox.showerror('错误', '年龄必须是整数！')
            return
        try:
            birthday = datetime.datetime.strptime(birthday, '%Y-%m-%d').date()
        except ValueError:
            tk.messagebox.showerror('错误', '生日必须是"YYYY-MM-DD"格式的日期！')
            return
        #将学生信息保存到数据库
        con, cur = connect_database()
        cur.execute('insert into student values(%s, %s, %s, %s, %s, %s, %s, %s)', (sid, sname,
            gender, age, sclass, school, birthday, province))
        con.commit()
        cur.close()
        con.close()
        tk.messagebox.showinfo('提示', '数据插入成功！')
        # 清空表单
        sid_entry.delete(0, 'end')
        sname_entry.delete(0, 'end')
        gender_cmb.delete(0, 'end')
        age_entry.delete(0, 'end')
        sclass_entry.delete(0, 'end')
        school_entry.delete(0, 'end')
        birthday_entry.delete(0, 'end')
        province_entry.delete(0, 'end')
    #学号输入框
    sid_label = tk.Label(add_student_window, text='学号:', font=font3)
    sid_label.pack()
    sid_entry = tk.Entry(add_student_window)
    sid_entry.pack()
    #姓名输入框
    sname_label = tk.Label(add_student_window, text='姓名:', font=font3)
    sname_label.pack()
    sname_entry = tk.Entry(add_student_window)
    sname_entry.pack()
```

```
    #性别下拉菜单
    gender_label = tk.Label(add_student_window, text='性别:', font=font3)
    gender_label.pack()
    gender_cmb = ttk.Combobox(add_student_window, width=15, font=font3)
    gender_cmb.pack()
    gender_cmb['value'] = ('男','女')
    #年龄输入框
    age_label = tk.Label(add_student_window, text='年龄:', font=font3)
    age_label.pack()
    age_entry = tk.Entry(add_student_window)
    age_entry.pack()
    #班级输入框
    sclass_label = tk.Label(add_student_window, text='班级:', font=font3)
    sclass_label.pack()
    sclass_entry = tk.Entry(add_student_window)
    sclass_entry.pack()
    #学院输入框
    school_label = tk.Label(add_student_window, text='学院:', font=font3)
    school_label.pack()
    school_entry = tk.Entry(add_student_window)
    school_entry.pack()
    #生日输入框
    birthday_label = tk.Label(add_student_window, text='生日:', font=font3)
    birthday_label.pack()
    birthday_entry = tk.Entry(add_student_window)
    birthday_entry.pack()
    #籍贯输入框
    province_label = tk.Label(add_student_window, text='籍贯:', font=font3)
    province_label.pack()
    province_entry = tk.Entry(add_student_window)
    province_entry.pack()
    #确定和取消按钮
    confirm_button = tk.Button(add_student_window, text='确定', command=add_student, font=font3,
        fg='red', cursor='hand2')
    confirm_button.pack()
    cancel_button = tk.Button(add_student_window, text='取消',command=add_student_window.destroy,
        font=font3, fg='red', cursor='hand2')
    cancel_button.pack()

#查询学生信息
def select_student_window():
    select_student_window = tk.Toplevel(root)
    select_student_window.geometry('300x500+550+150')
    select_student_window.title('查询学生信息')
    def select_student():
        #获取用户输入的学生信息
        sid = sid_entry.get()
        sname = sname_entry.get()
        gender = gender_cmb.get()
        age = age_entry.get()
        sclass = sclass_entry.get()
        school = school_entry.get()
        birthday = birthday_entry.get()
        province = province_entry.get()
        #检查学生信息有效性
        if age:
            try:
                age = int(age)
            except ValueError:
                tk.messagebox.showerror('错误', '年龄必须是整数! ')
                return
        if birthday:
```

```
            try:
                birthday = datetime.datetime.strptime(birthday, '%Y-%m-%d').date()
            except ValueError:
                tk.messagebox.showerror('错误', '生日必须是"YYYY-MM-DD"格式的日期！')
                return
        #根据输入的学生信息进行查询
        con, cur = connect_database()
        sql = 'select * from student where 1'
        params = []
        if sid:
            sql += ' and sid like %s'
            params.append('%' + sid + '%')
        if sname:
            sql += ' and sname like %s'
            params.append('%' + sname + '%')
        if gender:
            sql += ' and gender like %s'
            params.append(gender)
        if age:
            sql += ' and age like %s'
            params.append('%' + str(age) + '%')
        if sclass:
            sql += ' and sclass like %s'
            params.append('%' + sclass + '%')
        if school:
            sql += ' and school like %s'
            params.append('%' + school + '%')
        if birthday:
            sql += ' and birthday like %s'
            params.append('%' + str(birthday) + '%')
        if province:
            sql += ' and province like %s'
            params.append('%' + province + '%')
        cur.execute(sql, params)
        result = cur.fetchall()
        show_result(result)
        con.commit()
        cur.close()
        con.close()
        # 清空表单
        sid_entry.delete(0, 'end')
        sname_entry.delete(0, 'end')
        gender_cmb.delete(0, 'end')
        age_entry.delete(0, 'end')
        sclass_entry.delete(0, 'end')
        school_entry.delete(0, 'end')
        birthday_entry.delete(0, 'end')
        province_entry.delete(0, 'end')
    #学号输入框
    sid_label = tk.Label(select_student_window, text='学号:', font=font3)
    sid_label.pack()
    sid_entry = tk.Entry(select_student_window)
    sid_entry.pack()
    #姓名输入框
    sname_label = tk.Label(select_student_window, text='姓名:', font=font3)
    sname_label.pack()
    sname_entry = tk.Entry(select_student_window)
    sname_entry.pack()
    #性别下拉菜单
    gender_label = tk.Label(select_student_window, text='性别:', font=font3)
    gender_label.pack()
    gender_cmb = ttk.Combobox(select_student_window, width=15, font=font3)
```

```
    gender_cmb.pack()
    gender_cmb['value'] = ('男','女')
    #年龄输入框
    age_label = tk.Label(select_student_window, text='年龄:', font=font3)
    age_label.pack()
    age_entry = tk.Entry(select_student_window)
    age_entry.pack()
    #班级输入框
    sclass_label = tk.Label(select_student_window, text='班级:', font=font3)
    sclass_label.pack()
    sclass_entry = tk.Entry(select_student_window)
    sclass_entry.pack()
    #学院输入框
    school_label = tk.Label(select_student_window, text='学院:', font=font3)
    school_label.pack()
    school_entry = tk.Entry(select_student_window)
    school_entry.pack()
    #生日输入框
    birthday_label = tk.Label(select_student_window, text='生日:', font=font3)
    birthday_label.pack()
    birthday_entry = tk.Entry(select_student_window)
    birthday_entry.pack()
    #籍贯输入框
    province_label = tk.Label(select_student_window, text='籍贯:', font=font3)
    province_label.pack()
    province_entry = tk.Entry(select_student_window)
    province_entry.pack()
    #确定和取消按钮
    confirm_button = tk.Button(select_student_window, text='确定', command=select_student, font=
        font3, fg='red', cursor='hand2')
    confirm_button.pack()
    cancel_button = tk.Button(select_student_window, text='取消', command=select_student_window.
        destroy, font=font3, fg='red', cursor='hand2')
    cancel_button.pack()

#更新学生信息
def update_student_window():
    update_student_window = tk.Toplevel(root)
    update_student_window.geometry('500x300+450+150')
    update_student_window.title('更新学生信息')
    #标签和字段名的映射
    field_map = {'学号':'sid', '姓名':'sname', '性别':'gender', '年龄':'age', '班级':'sclass', '学
        院':'school', '生日':'birthday', '籍贯':'province'}
    def update_student():
        #获取用户输入的学生信息
        for field, old_entry, new_entry in entries:
            old_value = old_entry.get()
            new_value = new_entry.get()
            field = field_map[field]  #从映射中获取字段名
            #根据输入的学生信息进行更新
            con, cur = connect_database()
            if old_value and new_value:
                cur.execute('update student set ' + f'{field}' + ' = %s where ' + f'{field}' + ' =
                    %s', (new_value, old_value))
                con.commit()
            elif old_value or new_value:
                tk.messagebox.showerror('错误', '旧值和新值都必须填写！')
                return
        cur.close()
        con.close()
        tk.messagebox.showinfo('成功', '更新成功！')
        #清空表单
```

```
        for _, old_entry, new_entry in entries: #用_表示不关心第一个元素field
            old_entry.delete(0, 'end')
            new_entry.delete(0, 'end')

    #标签和输入框
    entries = []
    for i, field_label in enumerate(field_map.keys()):
        label = tk.Label(update_student_window, width=7)
        label.grid(row=i, column=0)
        old_label = tk.Label(update_student_window, text=f'原{field_label}:', font=font3)
        old_label.grid(row=i, column=1)
        new_label = tk.Label(update_student_window, text=f'新{field_label}:', font=font3)
        new_label.grid(row=i, column=3)
        old_entry = tk.Entry(update_student_window)
        old_entry.grid(row=i, column=2)
        new_entry = tk.Entry(update_student_window)
        new_entry.grid(row=i, column=4)
        entries.append((field_label, old_entry, new_entry))
    #确定和取消按钮
    confirm_button = tk.Button(update_student_window, text='确定', command=update_student, font=
        font3, fg='red', cursor='hand2')
    confirm_button.grid(row=8, column=2)
    cancel_button = tk.Button(update_student_window, text='取消', command=update_student_window.
        destroy, font=font3, fg='red', cursor='hand2')
    cancel_button.grid(row=8, column=4)

#删除学生信息
def delete_student_window():
    delete_student_window = tk.Toplevel(root)
    delete_student_window.geometry('300x500+550+150')
    delete_student_window.title('删除学生信息')
    def delete_student():
        #获取用户输入的学生信息
        sid = sid_entry.get()
        sname = sname_entry.get()
        gender = gender_cmb.get()
        age = age_entry.get()
        sclass = sclass_entry.get()
        school = school_entry.get()
        birthday = birthday_entry.get()
        province = province_entry.get()
        #检查学生信息有效性
        if age:
            try:
                age = int(age)
            except ValueError:
                tk.messagebox.showerror('错误', '年龄必须是整数！')
                return
        if birthday:
            try:
                birthday = datetime.datetime.strptime(birthday, '%Y-%m-%d').date()
            except ValueError:
                tk.messagebox.showerror('错误', '生日必须是"YYYY-MM-DD"格式的日期！')
                return
        #根据输入的学生信息进行查询
        confirm = tk.messagebox.askyesno('确认', '你确定要删除此学生信息吗？')
        if confirm:
            con, cur = connect_database()
            sql = ' from student where 1'
            params = []
            if sid:
                sql += ' and sid like %s'
```

```
                params.append('%' + sid + '%')
            if sname:
                sql += ' and sname like %s'
                params.append('%' + sname + '%')
            if gender:
                sql += ' and gender like %s'
                params.append('%' + gender + '%')
            if age:
                sql += ' and age=%s'
                params.append('%' + str(age) + '%')
            if sclass:
                sql += ' and sclass like %s'
                params.append('%' + gender + '%')
            if school:
                sql += ' and school like %s'
                params.append('%' + school + '%')
            if birthday:
                sql += ' and birthday like %s'
                params.append('%' + str(birthday) + '%')
            if province:
                sql += ' and province like %s'
                params.append('%' + province + '%')
            #检查要删除的学生信息是否存在
            cur.execute('select *'+sql, params)
            result = cur.fetchall()
            if result is None:
                tk.messagebox.showerror('错误', '您要删除的学生信息不存在！')
            else:
                cur.execute('delete'+sql, params)
                con.commit()
                cur.close()
                con.close()
                tk.messagebox.showinfo('成功', '删除成功！')
                # 清空表单
                sid_entry.delete(0, 'end')
                sname_entry.delete(0, 'end')
                gender_cmb.delete(0, 'end')
                age_entry.delete(0, 'end')
                sclass_entry.delete(0, 'end')
                school_entry.delete(0, 'end')
                birthday_entry.delete(0, 'end')
                province_entry.delete(0, 'end')
    #学号输入框
    sid_label = tk.Label(delete_student_window, text='学号:', font=font3)
    sid_label.pack()
    sid_entry = tk.Entry(delete_student_window)
    sid_entry.pack()
    #姓名输入框
    sname_label = tk.Label(delete_student_window, text='姓名:', font=font3)
    sname_label.pack()
    sname_entry = tk.Entry(delete_student_window)
    sname_entry.pack()
    #性别下拉菜单
    gender_label = tk.Label(delete_student_window, text='性别:', font=font3)
    gender_label.pack()
    gender_cmb = ttk.Combobox(delete_student_window, width=15, font=font3)
    gender_cmb.pack()
    gender_cmb['value'] = ('男','女')
    #年龄输入框
    age_label = tk.Label(delete_student_window, text='年龄:', font=font3)
    age_label.pack()
    age_entry = tk.Entry(delete_student_window)
```

```
    age_entry.pack()
    #班级输入框
    sclass_label = tk.Label(delete_student_window, text='班级:', font=font3)
    sclass_label.pack()
    sclass_entry = tk.Entry(delete_student_window)
    sclass_entry.pack()
    #学院输入框
    school_label = tk.Label(delete_student_window, text='学院:', font=font3)
    school_label.pack()
    school_entry = tk.Entry(delete_student_window)
    school_entry.pack()
    #生日输入框
    birthday_label = tk.Label(delete_student_window, text='生日:', font=font3)
    birthday_label.pack()
    birthday_entry = tk.Entry(delete_student_window)
    birthday_entry.pack()
    #籍贯输入框
    province_label = tk.Label(delete_student_window, text='籍贯:', font=font3)
    province_label.pack()
    province_entry = tk.Entry(delete_student_window)
    province_entry.pack()
    #确定和取消按钮
    confirm_button = tk.Button(delete_student_window, text='确定', command=delete_student, font=
        font3, fg='red', cursor='hand2')
    confirm_button.pack()
    cancel_button = tk.Button(delete_student_window, text='取消', command=delete_student_window.
        destroy, font=font3, fg='red', cursor='hand2')
    cancel_button.pack()

#添加课程信息
def add_course_window():
    add_course_window = tk.Toplevel(root)
    add_course_window.geometry('300x500+550+150')
    add_course_window.title('添加课程信息')
    def add_course():
        #获取用户输入的课程信息
        cid = cid_entry.get()
        cname = cname_entry.get()
        teacher = teacher_entry.get()
        credit = credit_entry.get()
        cpid = cpid_entry.get()
        #检查课程信息有效性
        if not cid:
            tk.messagebox.showerror('错误', '课程号不能为空！')
            return
        if not cname:
            tk.messagebox.showerror('错误', '课程名不能为空！')
            return
        #将课程信息保存到数据库
        con, cur = connect_database()
        cur.execute('insert into course values(%s, %s, %s, %s, %s)', (cid, cname, teacher, credit,
            cpid))
        con.commit()
        cur.close()
        con.close()
        tk.messagebox.showinfo('提示', '数据插入成功！')
        # 清空表单
        cid_entry.delete(0, 'end')
        cname_entry.delete(0, 'end')
        teacher_entry.delete(0, 'end')
        credit_entry.delete(0, 'end')
        cpid_entry.delete(0, 'end')
```

```
    #课程号输入框
    cid_label = tk.Label(add_course_window, text='课程号:', font=font3)
    cid_label.pack()
    cid_entry = tk.Entry(add_course_window)
    cid_entry.pack()
    #课程名输入框
    cname_label = tk.Label(add_course_window, text='课程名:', font=font3)
    cname_label.pack()
    cname_entry = tk.Entry(add_course_window)
    cname_entry.pack()
    #教师输入框
    teacher_label = tk.Label(add_course_window, text='教师:', font=font3)
    teacher_label.pack()
    teacher_entry = tk.Entry(add_course_window)
    teacher_entry.pack()
    #学分输入框
    credit_label = tk.Label(add_course_window, text='学分:', font=font3)
    credit_label.pack()
    credit_entry = tk.Entry(add_course_window)
    credit_entry.pack()
    #先修课程输入框
    cpid_label = tk.Label(add_course_window, text='先修课程:', font=font3)
    cpid_label.pack()
    cpid_entry = tk.Entry(add_course_window)
    cpid_entry.pack()
    #确定和取消按钮
    confirm_button = tk.Button(add_course_window, text='确定', command=add_course, font=font3, fg=
        'red', cursor='hand2')
    confirm_button.pack()
    cancel_button = tk.Button(add_course_window, text='取消',command=add_course_window.destroy,
        font=font3, fg='red', cursor='hand2')
    cancel_button.pack()

#查询课程信息
def select_course_window():
    select_course_window = tk.Toplevel(root)
    select_course_window.geometry('300x500+550+150')
    select_course_window.title('查询课程信息')
    def select_course():
        #获取用户输入的课程信息
        cid = cid_entry.get()
        cname = cname_entry.get()
        teacher = teacher_entry.get()
        credit = credit_entry.get()
        cpid = cpid_entry.get()
        #根据输入的学生信息进行查询
        con, cur = connect_database()
        sql = 'select * from course where 1'
        params = []
        if cid:
            sql += ' and cid like %s'
            params.append('%' + cid + '%')
        if cname:
            sql += ' and cname like %s'
            params.append('%' + cname + '%')
        if teacher:
            sql += ' and teacher like %s'
            params.append('%' + teacher + '%')
        if credit:
            sql += ' and credit like %s'
            params.append('%' + credit + '%')
        if cpid:
```

```
            sql += ' and cpid like %s'
            params.append('%' + cpid + '%')
        cur.execute(sql, params)
        result = cur.fetchall()
        show_result(result)
        con.commit()
        cur.close()
        con.close()
        # 清空表单
        cid_entry.delete(0, 'end')
        cname_entry.delete(0, 'end')
        teacher_entry.delete(0, 'end')
        credit_entry.delete(0, 'end')
        cpid_entry.delete(0, 'end')
        return result
    #课程号输入框
    cid_label = tk.Label(select_course_window, text='课程号:', font=font3)
    cid_label.pack()
    cid_entry = tk.Entry(select_course_window)
    cid_entry.pack()
    #课程名输入框
    cname_label = tk.Label(select_course_window, text='课程名:', font=font3)
    cname_label.pack()
    cname_entry = tk.Entry(select_course_window)
    cname_entry.pack()
    #教师输入框
    teacher_label = tk.Label(select_course_window, text='教师:', font=font3)
    teacher_label.pack()
    teacher_entry = tk.Entry(select_course_window)
    teacher_entry.pack()
    #学分输入框
    credit_label = tk.Label(select_course_window, text='学分:', font=font3)
    credit_label.pack()
    credit_entry = tk.Entry(select_course_window)
    credit_entry.pack()
    #先修课程输入框
    cpid_label = tk.Label(select_course_window, text='先修课程:', font=font3)
    cpid_label.pack()
    cpid_entry = tk.Entry(select_course_window)
    cpid_entry.pack()
    #确定和取消按钮
    confirm_button = tk.Button(select_course_window, text='确定', command=select_course, font=
        font3, fg='red', cursor='hand2')
    confirm_button.pack()
    cancel_button = tk.Button(select_course_window, text='取消', command=select_course_window.
        destroy, font=font3, fg='red', cursor='hand2')
    cancel_button.pack()

#更新课程信息
def update_course_window():
    update_course_window = tk.Toplevel(root)
    update_course_window.geometry('500x300+450+150')
    update_course_window.title('更新课程信息')
    #创建标签和字段名的映射
    field_map = {'课程号':'cid', '课程名':'cname', '教师':'teacher', '学分':'credit', '先修课程':'
        cpid'}
    def update_course():
        #获取用户输入的课程信息
        for field, old_entry, new_entry in entries:
            old_value = old_entry.get()
            new_value = new_entry.get()
            field = field_map[field]  #从映射中获取字段名
```

```
575                 #根据输入的学生信息进行更新
576                 con, cur = connect_database()
577                 if old_value and new_value:
578                     cur.execute('update course set ' + f'{field}' + ' = %s where ' + f'{field}' + ' =
                            %s', (new_value, old_value))
579                     con.commit()
580                 elif old_value or new_value:
581                     tk.messagebox.showerror('错误', '旧值和新值都必须填写！')
582                     return
583             cur.close()
584             con.close()
585             tk.messagebox.showinfo('成功', '更新成功！')
586             #清空表单
587             for _, old_entry, new_entry in entries: #用_表示不关心第一个元素field
588                 old_entry.delete(0, 'end')
589                 new_entry.delete(0, 'end')
590 
591         #标签和输入框
592         entries = []
593         for i, field_label in enumerate(field_map.keys()):
594             label = tk.Label(update_course_window, width=3)
595             label.grid(row=i, column=0)
596             old_label = tk.Label(update_course_window, text=f'原{field_label}:', font=font3)
597             old_label.grid(row=i, column=1)
598             new_label = tk.Label(update_course_window, text=f'新{field_label}:', font=font3)
599             new_label.grid(row=i, column=3)
600             old_entry = tk.Entry(update_course_window)
601             old_entry.grid(row=i, column=2)
602             new_entry = tk.Entry(update_course_window)
603             new_entry.grid(row=i, column=4)
604             entries.append((field_label, old_entry, new_entry))
605         #确定和取消按钮
606         confirm_button = tk.Button(update_course_window, text='确定', command=update_course, font=
                font3, fg='red', cursor='hand2')
607         confirm_button.grid(row=5, column=2)
608         cancel_button = tk.Button(update_course_window, text='取消', command=update_course_window.
                destroy, font=font3, fg='red', cursor='hand2')
609         cancel_button.grid(row=5, column=4)
610 
611 #删除课程信息
612 def delete_course_window():
613     delete_course_window = tk.Toplevel(root)
614     delete_course_window.geometry('300x500+550+150')
615     delete_course_window.title('删除课程信息')
616     def delete_course():
617         #获取用户输入的课程信息
618         cid = cid_entry.get()
619         cname = cname_entry.get()
620         teacher = teacher_entry.get()
621         credit = credit_entry.get()
622         cpid = cpid_entry.get()
623         #根据输入的课程信息进行删除
624         confirm = tk.messagebox.askyesno('确认', '您确定要删除此课程信息吗？')
625         if confirm:
626             con, cur = connect_database()
627             sql = ' from course where 1'
628             params = []
629             if cid:
630                 sql += ' and cid like %s'
631                 params.append('%' + cid + '%')
632             if cname:
633                 sql += ' and cname like %s'
```

```
                params.append('%' + cname + '%')
            if teacher:
                sql += ' and teacher like %s'
                params.append('%' + teacher + '%')
            if credit:
                sql += ' and credit like %s'
                params.append('%' + credit + '%')
            if cpid:
                sql += ' and cpid like %s'
                params.append('%' + cpid + '%')
            #检查要删除的课程信息是否存在
            cur.execute('select *'+sql, params)
            result = cur.fetchall()
            if result is None:
                tk.messagebox.showerror('错误', '您要删除的课程信息不存在！')
            else:
                cur.execute('delete'+sql, params)
                con.commit()
                cur.close()
                con.close()
                tk.messagebox.showinfo('成功', '删除成功！')
                # 清空表单
                cid_entry.delete(0, 'end')
                cname_entry.delete(0, 'end')
                teacher_entry.delete(0, 'end')
                credit_entry.delete(0, 'end')
                cpid_entry.delete(0, 'end')
    #课程号输入框
    cid_label = tk.Label(delete_course_window, text='课程号:', font=font3)
    cid_label.pack()
    cid_entry = tk.Entry(delete_course_window)
    cid_entry.pack()
    #课程名输入框
    cname_label = tk.Label(delete_course_window, text='课程名:', font=font3)
    cname_label.pack()
    cname_entry = tk.Entry(delete_course_window)
    cname_entry.pack()
    #教师输入框
    teacher_label = tk.Label(delete_course_window, text='教师:', font=font3)
    teacher_label.pack()
    teacher_entry = tk.Entry(delete_course_window)
    teacher_entry.pack()
    #学分输入框
    credit_label = tk.Label(delete_course_window, text='学分:', font=font3)
    credit_label.pack()
    credit_entry = tk.Entry(delete_course_window)
    credit_entry.pack()
    #先修课程输入框
    cpid_label = tk.Label(delete_course_window, text='先修课程:', font=font3)
    cpid_label.pack()
    cpid_entry = tk.Entry(delete_course_window)
    cpid_entry.pack()
    #确定和取消按钮
    confirm_button = tk.Button(delete_course_window, text='确定', command=delete_course, font=
        font3, fg='red', cursor='hand2')
    confirm_button.pack()
    cancel_button = tk.Button(delete_course_window, text='取消', command=delete_course_window.
        destroy, font=font3, fg='red', cursor='hand2')
    cancel_button.pack()

#添加选课信息
def add_sc_window():
```

```
    add_sc_window = tk.Toplevel(root)
    add_sc_window.geometry('300x500+550+150')
    add_sc_window.title('添加选课信息')
    def add_sc():
        #获取用户输入的选课信息
        sid = sid_entry.get()
        cid = cid_entry.get()
        year = year_entry.get()
        term = term_cmb.get()
        #检查选课信息有效性
        if not sid:
            tk.messagebox.showerror('错误', '学号不能为空！')
            return
        if not cid:
            tk.messagebox.showerror('错误', '课程号不能为空！')
            return
        if not year:
            tk.messagebox.showerror('错误', '学年不能为空！')
            return
        if not term:
            tk.messagebox.showerror('错误', '学期不能为空！')
            return
        #将选课信息保存到数据库
        con, cur = connect_database()
        cur.execute('insert into sc(sid, cid, year, term) values(%s, %s, %s, %s)', (sid, cid, year
            , term))
        con.commit()
        cur.close()
        con.close()
        tk.messagebox.showinfo('提示', '数据插入成功！')
        # 清空表单
        sid_entry.delete(0, 'end')
        cid_entry.delete(0, 'end')
        year_entry.delete(0, 'end')
        term_cmb.delete(0, 'end')
    #学号输入框
    sid_label = tk.Label(add_sc_window, text='学号:', font=font3)
    sid_label.pack()
    sid_entry = tk.Entry(add_sc_window)
    sid_entry.pack()
    #课程号输入框
    cid_label = tk.Label(add_sc_window, text='课程号:', font=font3)
    cid_label.pack()
    cid_entry = tk.Entry(add_sc_window)
    cid_entry.pack()
    #学年输入框
    year_label = tk.Label(add_sc_window, text='学年:', font=font3)
    year_label.pack()
    year_entry = tk.Entry(add_sc_window)
    year_entry.pack()
    #学期下拉菜单
    term_label = tk.Label(add_sc_window, text='学期:', width=17, font=font3)
    term_label.pack()
    term_cmb = ttk.Combobox(add_sc_window, width=15, font=font3)
    term_cmb.pack()
    term_cmb['value'] = ('第一学期', '第二学期', '小学期')
    #确定和取消按钮
    confirm_button = tk.Button(add_sc_window, text='确定', command=add_sc, font=font3, fg='red',
        cursor='hand2')
    confirm_button.pack()
    cancel_button = tk.Button(add_sc_window, text='取消', command=add_sc_window.destroy, font=
        font3, fg='red', cursor='hand2')
```

```
    cancel_button.pack()

#查询选课信息
def select_sc_window():
    select_sc_window = tk.Toplevel(root)
    select_sc_window.geometry('300x500+550+150')
    select_sc_window.title('添加选课信息')
    def select_sc():
        #获取用户输入的选课信息
        sid = sid_entry.get()
        cid = cid_entry.get()
        year = year_entry.get()
        term = term_cmb.get()
        #根据输入的选课信息进行查询
        con, cur = connect_database()
        sql = 'select sid, cid, year, term from sc where 1'
        params = []
        if sid:
            sql += ' and sid like %s'
            params.append('%' + sid + '%')
        if cid:
            sql += ' and cname like %s'
            params.append('%' + cid + '%')
        if year:
            sql += ' and year like %s'
            params.append('%' + year + '%')
        if term:
            sql += ' and term like %s'
            params.append('%' + term + '%')
        cur.execute(sql, params)
        result = cur.fetchall()
        show_result(result)
        con.commit()
        cur.close()
        con.close()
        # 清空表单
        sid_entry.delete(0, 'end')
        cid_entry.delete(0, 'end')
        year_entry.delete(0, 'end')
        term_cmb.delete(0, 'end')
    #学号输入框
    sid_label = tk.Label(select_sc_window, text='学号:', font=font3)
    sid_label.pack()
    sid_entry = tk.Entry(select_sc_window)
    sid_entry.pack()
    #课程号输入框
    cid_label = tk.Label(select_sc_window, text='课程号:', font=font3)
    cid_label.pack()
    cid_entry = tk.Entry(select_sc_window)
    cid_entry.pack()
    #学年输入框
    year_label = tk.Label(select_sc_window, text='学年:', font=font3)
    year_label.pack()
    year_entry = tk.Entry(select_sc_window)
    year_entry.pack()
    #学期下拉菜单
    term_label = tk.Label(select_sc_window, text='学期:', width=17, font=font3)
    term_label.pack()
    term_cmb = ttk.Combobox(select_sc_window, width=15, font=font3)
    term_cmb.pack()
    term_cmb['value'] = ('第一学期', '第二学期', '小学期')
    #确定和取消按钮
```

```
    confirm_button = tk.Button(select_sc_window, text='确定', command=select_sc, font=font3, fg='red', cursor='hand2')
    confirm_button.pack()
    cancel_button = tk.Button(select_sc_window, text='取消', command=select_sc_window.destroy, font=font3, fg='red', cursor='hand2')
    cancel_button.pack()

#更新选课信息
def update_sc_window():
    update_sc_window = tk.Toplevel(root)
    update_sc_window.geometry('500x300+450+150')
    update_sc_window.title('添加选课信息')
    #标签和字段名的映射
    field_map = {'学号':'sid', '课程号':'cid', '学年':'year', '学期':'term'}
    def update_sc():
        #获取用户输入的课程信息
        for field, old_entry, new_entry in entries:
            old_value = old_entry.get()
            new_value = new_entry.get()
            field = field_map[field]  #从映射中获取字段名
            #根据输入的学生信息进行更新
            con, cur = connect_database()
            if old_value and new_value:
                cur.execute('update sc set ' + f'{field}' + ' = %s where ' + f'{field}' + ' = %s', (new_value, old_value))
                con.commit()
            elif old_value or new_value:
                tk.messagebox.showerror('错误', '旧值和新值都必须填写！')
                return
        cur.close()
        con.close()
        tk.messagebox.showinfo('成功', '更新成功！')
        #清空表单
        for _, old_entry, new_entry in entries: #用_表示不关心第一个元素field
            old_entry.delete(0, 'end')
            new_entry.delete(0, 'end')

    #标签和输入框
    entries = []
    for i, field_label in enumerate(field_map.keys()):
        label = tk.Label(update_sc_window, width=4)
        label.grid(row=i, column=0)
        old_label = tk.Label(update_sc_window, text=f'原{field_label}:', font=font3)
        old_label.grid(row=i, column=1)
        new_label = tk.Label(update_sc_window, text=f'新{field_label}:', font=font3)
        new_label.grid(row=i, column=3)
        old_entry = tk.Entry(update_sc_window)
        old_entry.grid(row=i, column=2)
        new_entry = tk.Entry(update_sc_window)
        new_entry.grid(row=i, column=4)
        entries.append((field_label, old_entry, new_entry))
    #确定和取消按钮
    confirm_button = tk.Button(update_sc_window, text='确定', command=update_sc, font=font3, fg='red', cursor='hand2')
    confirm_button.grid(row=4, column=2)
    cancel_button = tk.Button(update_sc_window, text='取消', command=update_sc_window.destroy, font=font3, fg='red', cursor='hand2')
    cancel_button.grid(row=4, column=4)

#删除选课信息
def delete_sc_window():
    delete_sc_window = tk.Toplevel(root)
```

```
    delete_sc_window.geometry('300x500+550+150')
    delete_sc_window.title('删除选课信息')
    def delete_sc():
        #获取用户输入的选课信息
        sid = sid_entry.get()
        cid = cid_entry.get()
        year = year_entry.get()
        term = term_cmb.get()
        #根据输入的选课信息进行删除
        confirm = tk.messagebox.askyesno('确认', '你确定要删除此选课信息吗？')
        if confirm:
            con, cur = connect_database()
            sql = ' from sc where 1'
            params = []
            if sid:
                sql += ' and sid like %s'
                params.append('%' + cid + '%')
            if cid:
                sql += ' and cid like %s'
                params.append('%' + cid + '%')
            if year:
                sql += ' and year like %s'
                params.append('%' + year + '%')
            if term:
                sql += ' and term like %s'
                params.append('%' + term + '%')
            #检查要删除的课程信息是否存在
            cur.execute('select *'+sql, params)
            result = cur.fetchall()
            if result is None:
                tk.messagebox.showerror('错误', '您要删除的课程信息不存在！')
            else:
                cur.execute('delete'+sql, params)
                con.commit()
                cur.close()
                con.close()
                tk.messagebox.showinfo('成功', '删除成功！')
                #清空表单
                sid_entry.delete(0, 'end')
                cid_entry.delete(0, 'end')
                year_entry.delete(0, 'end')
                term_cmb.delete(0, 'end')
    #学号输入框
    sid_label = tk.Label(delete_sc_window, text='学号:', font=font3)
    sid_label.pack()
    sid_entry = tk.Entry(delete_sc_window)
    sid_entry.pack()
    #课程号输入框
    cid_label = tk.Label(delete_sc_window, text='课程号:', font=font3)
    cid_label.pack()
    cid_entry = tk.Entry(delete_sc_window)
    cid_entry.pack()
    #学年输入框
    year_label = tk.Label(delete_sc_window, text='学年:', font=font3)
    year_label.pack()
    year_entry = tk.Entry(delete_sc_window)
    year_entry.pack()
    #学期输入框
    term_label = tk.Label(delete_sc_window, text='学期:', font=font3)
    term_label.pack()
    term_cmb = ttk.Combobox(delete_sc_window, width=15, font=font3)
    term_cmb.pack()
```

```
    term_cmb['value'] = ('第一学期', '第二学期', '小学期')
    #确定和取消按钮
    confirm_button = tk.Button(delete_sc_window, text='确定', command=delete_sc, font=font3, fg='
        red', cursor='hand2')
    confirm_button.pack()
    cancel_button = tk.Button(delete_sc_window, text='取消', command=delete_sc_window.destroy,
        font=font3, fg='red', cursor='hand2')
    cancel_button.pack()

#添加成绩信息
def add_grade_window():
    add_grade_window = tk.Toplevel(root)
    add_grade_window.geometry('300x500+550+150')
    add_grade_window.title('添加成绩信息')
    def add_grade():
        #获取用户输入的选课信息
        sid = sid_entry.get()
        cid = cid_entry.get()
        year = year_entry.get()
        term = term_cmb.get()
        grade = grade_entry.get()
        #检查成绩信息有效性
        if not sid:
            tk.messagebox.showerror('错误', '学号不能为空！')
            return
        if not cid:
            tk.messagebox.showerror('错误', '课程号不能为空！')
            return
        try:
            grade = float(grade)
        except ValueError:
            tk.messagebox.showerror('错误', '成绩必须是数值！')
            return
        #将选课信息保存到数据库
        con, cur = connect_database()
        cur.execute('update sc set grade=%s where sid=%s and cid=%s and year=%s and term=%s', (
            grade, sid, cid, year, term))
        con.commit()
        cur.close()
        con.close()
        tk.messagebox.showinfo('提示', '数据插入成功！')
        #清空表单
        sid_entry.delete(0, 'end')
        cid_entry.delete(0, 'end')
        year_entry.delete(0, 'end')
        term_cmb.delete(0, 'end')
    #学号输入框
    sid_label = tk.Label(add_grade_window, text='学号:', font=font3)
    sid_label.pack()
    sid_entry = tk.Entry(add_grade_window)
    sid_entry.pack()
    #课程号输入框
    cid_label = tk.Label(add_grade_window, text='课程号:', font=font3)
    cid_label.pack()
    cid_entry = tk.Entry(add_grade_window)
    cid_entry.pack()
    #学年输入框
    year_label = tk.Label(add_grade_window, text='学年:', font=font3)
    year_label.pack()
    year_entry = tk.Entry(add_grade_window)
    year_entry.pack()
    #学期下拉菜单
```

```
    term_label = tk.Label(add_grade_window, text='学期:', width=17, font=font3)
    term_label.pack()
    term_cmb = ttk.Combobox(add_grade_window, width=15, font=font3)
    term_cmb.pack()
    term_cmb['value'] = ('第一学期', '第二学期', '小学期')
    #成绩输入框
    grade_label = tk.Label(add_grade_window, text='成绩:', font=font3)
    grade_label.pack()
    grade_entry = tk.Entry(add_grade_window)
    grade_entry.pack()
    #确定和取消按钮
    confirm_button = tk.Button(add_grade_window, text='确定', command=add_grade, font=font3, fg='red', cursor='hand2')
    confirm_button.pack()
    cancel_button = tk.Button(add_grade_window, text='取消', command=add_grade_window.destroy, font=font3, fg='red', cursor='hand2')
    cancel_button.pack()


#查询成绩信息
def select_grade_window():
    select_grade_window = tk.Toplevel(root)
    select_grade_window.geometry('300x500+550+150')
    select_grade_window.title('查询成绩信息')
    def select_grade():
        #获取用户输入的选课信息
        sid = sid_entry.get()
        cid = cid_entry.get()
        year = year_entry.get()
        term = term_cmb.get()
        #根据输入的选课信息进行查询
        con, cur = connect_database()
        sql = 'select sid, cid, year, term, coalesce(grade,0) as grade from sc where 1' #coalesce 实现成绩为NULL时返回结果为0
        params = []
        if sid:
            sql += ' and sid like %s'
            params.append('%' + sid + '%')
        if cid:
            sql += ' and cid like %s'
            params.append('%' + cid + '%')
        if year:
            sql += ' and year like %s'
            params.append('%' + year + '%')
        if term:
            sql += ' and term like %s'
            params.append('%' + term + '%')
        cur.execute(sql, params)
        result = cur.fetchall()
        show_result(result)
        con.commit()
        cur.close()
        con.close()
        #清空表单
        sid_entry.delete(0, 'end')
        cid_entry.delete(0, 'end')
        year_entry.delete(0, 'end')
        term_cmb.delete(0, 'end')
    #学号输入框
    sid_label = tk.Label(select_grade_window, text='学号:', font=font3)
    sid_label.pack()
    sid_entry = tk.Entry(select_grade_window)
```

```
    sid_entry.pack()
    #课程号输入框
    cid_label = tk.Label(select_grade_window, text='课程号:', font=font3)
    cid_label.pack()
    cid_entry = tk.Entry(select_grade_window)
    cid_entry.pack()
    #学年输入框
    year_label = tk.Label(select_grade_window, text='学年:', font=font3)
    year_label.pack()
    year_entry = tk.Entry(select_grade_window)
    year_entry.pack()
    #学期下拉菜单
    term_label = tk.Label(select_grade_window, text='学期:', width=17, font=font3)
    term_label.pack()
    term_cmb = ttk.Combobox(select_grade_window, width=15, font=font3)
    term_cmb.pack()
    term_cmb['value'] = ('第一学期', '第二学期', '小学期')
    #确定和取消按钮
    confirm_button = tk.Button(select_grade_window, text='确定', command=select_grade, font=font3,
         fg='red', cursor='hand2')
    confirm_button.pack()
    cancel_button = tk.Button(select_grade_window, text='取消', command=select_grade_window.
        destroy, font=font3, fg='red', cursor='hand2')
    cancel_button.pack()

#更新成绩信息
def update_grade_window():
    update_grade_window = tk.Toplevel(root)
    update_grade_window.geometry('300x500+550+150')
    update_grade_window.title('更新成绩信息')
    def update_grade():
        #获取用户输入的选课信息
        sid = sid_entry.get()
        cid = cid_entry.get()
        year = year_entry.get()
        term = term_cmb.get()
        grade = grade_entry.get()
        #检查成绩信息有效性
        try:
            grade = float(grade)
        except ValueError:
            tk.messagebox.showerror('错误', '成绩必须是数值! ')
            return
        #根据输入的选课信息进行查询
        con, cur = connect_database()
        sql = 'update sc set grade=%s where 1'
        params = [grade]
        if sid:
            sql += ' and sid like %s'
            params.append('%' + sid + '%')
        if cid:
            sql += ' and cid like %s'
            params.append('%' + cid + '%')
        if year:
            sql += ' and year like %s'
            params.append('%' + year + '%')
        if term:
            sql += ' and term like %s'
            params.append('%' + term + '%')
        cur.execute(sql, params)
        con.commit()
        cur.close()
```

```
            con.close()
            tk.messagebox.showinfo('成功', '更新成功！')
            #清空表单
            sid_entry.delete(0, 'end')
            cid_entry.delete(0, 'end')
            year_entry.delete(0, 'end')
            term_cmb.delete(0, 'end')
            grade_entry.delete(0, 'end')
        #学号输入框
        sid_label = tk.Label(update_grade_window, text='学号:', font=font3)
        sid_label.pack()
        sid_entry = tk.Entry(update_grade_window)
        sid_entry.pack()
        #课程号输入框
        cid_label = tk.Label(update_grade_window, text='课程号:', font=font3)
        cid_label.pack()
        cid_entry = tk.Entry(update_grade_window)
        cid_entry.pack()
        #学年输入框
        year_label = tk.Label(update_grade_window, text='学年:', font=font3)
        year_label.pack()
        year_entry = tk.Entry(update_grade_window)
        year_entry.pack()
        #学期下拉菜单
        term_label = tk.Label(update_grade_window, text='学期:', width=17, font=font3)
        term_label.pack()
        term_cmb = ttk.Combobox(update_grade_window, width=15, font=font3)
        term_cmb.pack()
        term_cmb['value'] = ('第一学期', '第二学期', '小学期')
        #成绩输入框
        grade_label = tk.Label(update_grade_window, text='成绩:', font=font3)
        grade_label.pack()
        grade_entry = tk.Entry(update_grade_window)
        grade_entry.pack()
        #确定和取消按钮
        confirm_button = tk.Button(update_grade_window, text='确定', command=update_grade, font=font3,
             fg='red', cursor='hand2')
        confirm_button.pack()
        cancel_button = tk.Button(update_grade_window, text='取消', command=update_grade_window.
            destroy, font=font3, fg='red', cursor='hand2')
        cancel_button.pack()

#删除成绩信息
def delete_grade_window():
    delete_grade_window = tk.Toplevel(root)
    delete_grade_window.geometry('300x500+550+150')
    delete_grade_window.title('删除成绩信息')
    def delete_grade():
        #获取用户输入的选课信息
        sid = sid_entry.get()
        cid = cid_entry.get()
        year = year_entry.get()
        term = term_cmb.get()
        grade = grade_entry.get()
        #根据输入的选课信息进行删除
        confirm = messagebox.askyesno('确认', f'您确定要删除此成绩信息吗？')
        if confirm:
            con, cur = connect_database()
            sql = ' where 1'
            params = []
            if sid:
                sql += ' and sid like %s'
```

```
                params.append('%' + sid + '%')
            if cid:
                sql += ' and cid like %s'
                params.append('%' + cid + '%')
            if year:
                sql += ' and year like %s'
                params.append('%' + year + '%')
            if term:
                sql += ' and term like %s'
                params.append('%' + term + '%')
            if grade:
                sql += ' and grade like %s'
                params.append('%' + grade + '%')
            #检查要删除的课程信息是否存在
            cur.execute('select * from sc'+sql, params)
            result = cur.fetchall()
            if result is None:
                tk.messagebox.showerror('错误', '您要删除的成绩信息不存在！')
            else:
                cur.execute('update sc set grade=null'+sql, params)
                con.commit()
                cur.close()
                con.close()
                tk.messagebox.showinfo('成功', '删除成功！')
                #清空表单
                sid_entry.delete(0, 'end')
                cid_entry.delete(0, 'end')
                year_entry.delete(0, 'end')
                term_cmb.delete(0, 'end')
                grade_entry.delete(0, 'end')
    #学号输入框
    sid_label = tk.Label(delete_grade_window, text='学号:', font=font3)
    sid_label.pack()
    sid_entry = tk.Entry(delete_grade_window)
    sid_entry.pack()
    #课程号输入框
    cid_label = tk.Label(delete_grade_window, text='课程号:', font=font3)
    cid_label.pack()
    cid_entry = tk.Entry(delete_grade_window)
    cid_entry.pack()
    #学年输入框
    year_label = tk.Label(delete_grade_window, text='学年:', font=font3)
    year_label.pack()
    year_entry = tk.Entry(delete_grade_window)
    year_entry.pack()
    #学期下拉菜单
    term_label = tk.Label(delete_grade_window, text='学期:', width=17, font=font3)
    term_label.pack()
    term_cmb = ttk.Combobox(delete_grade_window, width=15, font=font3)
    term_cmb.pack()
    term_cmb['value'] = ('第一学期', '第二学期', '小学期')
    #成绩输入框
    grade_label = tk.Label(delete_grade_window, text='成绩:', font=font3)
    grade_label.pack()
    grade_entry = tk.Entry(delete_grade_window)
    grade_entry.pack()
    #确定和取消按钮
    confirm_button = tk.Button(delete_grade_window, text='确定', command=delete_grade, font=font3,
         fg='red', cursor='hand2')
    confirm_button.pack()
    cancel_button = tk.Button(delete_grade_window, text='取消', command=delete_grade_window.
        destroy, font=font3, fg='red', cursor='hand2')
```

```
    cancel_button.pack()

#成绩信息分析
def analysis_grade_window():
    analysis_grade_window = tk.Toplevel(root)
    analysis_grade_window.geometry('300x500+550+150')
    analysis_grade_window.title('删除成绩信息')
    #学生的最高分
    def student_max_grade(sid, year, term):
        con, cur = connect_database()
        cur.execute('select max(grade) from sc where sid = %s and year = %s and term = %s', (sid,
            year, term))
        result = cur.fetchone()
        student_max_grade = result[0] if result else None
        con.commit()
        cur.close()
        con.close()
        return student_max_grade
    #学生的最低分
    def student_min_grade(sid, year, term):
        con, cur = connect_database()
        cur.execute('select min(grade) from sc where sid=%s and year=%s and term=%s', (sid, year,
            term))
        result = cur.fetchone()
        student_min_grade = result[0] if result else None
        con.commit()
        cur.close()
        con.close()
        return student_min_grade
    #学生的平均分
    def student_avg_grade(sid, year, term):
        con, cur = connect_database()
        cur.execute('select sum(c.grade*b.credit)/sum(b.credit) from course b inner join sc c on b
            .cid=c.cid where c.sid = %s and c.year = %s and c.term = %s', (sid, year, term))
        result = cur.fetchone()
        student_avg_grade = result[0] if result else None
        con.commit()
        cur.close()
        con.close()
        return student_avg_grade
    #学生的绩点
    def student_gpa(sid, year, term):
        con, cur = connect_database()
        cur.execute('select c.grade, b.credit from course b inner join sc c on b.cid=c.cid where c
            .sid=%s and c.year=%s and c.term=%s', (sid, year, term))
        result = cur.fetchall()
        con.commit()
        cur.close()
        con.close()
        total_credit = 0
        total_g = 0
        for grade, credit in result:
            if grade is not None:
                if grade >= 95:
                    g = 4.3
                elif grade >= 90:
                    g = 4.0
                elif grade >= 85:
                    g = 3.7
                elif grade >= 81:
                    g = 3.3
                elif grade >= 78:
```

```
                    g = 3.0
                elif grade >= 75:
                    g = 2.7
                elif grade >= 72:
                    g = 2.3
                elif grade >= 68:
                    g = 2.0
                elif grade >= 64:
                    g = 1.7
                elif grade >= 60:
                    g = 1.3
                else:
                    g = 0
                total_g += g * credit
                total_credit += credit
        if total_credit == 0:
            return None
        else:
            gpa = total_g/total_credit
            return gpa
    #学生的优秀率
    def student_excellent_rate(sid, year, term):
        con, cur = connect_database()
        cur.execute('select count(*) from sc where sid=%s and year=%s and term=%s and grade is not
             null', (sid, year, term))
        total_count = cur.fetchone()[0]
        cur.execute('select count(*) from sc where sid=%s and year=%s and term=%s and grade>=90',
            (sid, year, term))
        excellent_count = cur.fetchone()[0]
        con.commit()
        cur.close()
        con.close()
        if total_count > 0:
            student_excellent_rate = excellent_count/total_count
            return format(student_excellent_rate, '.2%')
        else:
            return None
    #学生的及格率
    def student_pass_rate(sid, year, term):
        con, cur = connect_database()
        cur.execute('select count(*) from sc where sid=%s and year=%s and term=%s and grade is not
             null', (sid, year, term))
        total_count = cur.fetchone()[0]
        cur.execute('select count(*) from sc where sid=%s and year=%s and term=%s and grade>=60',
            (sid, year, term))
        pass_count = cur.fetchone()[0]
        con.commit()
        cur.close()
        con.close()
        if total_count > 0:
            student_pass_rate = pass_count/total_count
            return format(student_pass_rate, '.2%')
        else:
            return None
    #学生的不及格率
    def student_fail_rate(sid, year, term):
        if student_pass_rate(sid, year, term) is not None:
            student_fail_rate = 1 - float(student_pass_rate(sid, year, term).rstrip('%'))/100
            return format(student_fail_rate, '.2%')
        else:
            return None
    #优秀门数
```

```
1348      def student_excellent_count(sid, year, term):
1349          con, cur = connect_database()
1350          cur.execute('select count(*) from sc where sid=%s and year=%s and term=%s and grade>=90',
                  (sid, year, term))
1351          student_excellent_count = cur.fetchone()[0]
1352          con.commit()
1353          cur.close()
1354          con.close()
1355          return student_excellent_count
1356      #及格门数
1357      def student_pass_count(sid, year, term):
1358          con, cur = connect_database()
1359          cur.execute('select count(*) from sc where sid=%s and year=%s and term=%s and grade>=60',
                  (sid, year, term))
1360          student_pass_count = cur.fetchone()[0]
1361          con.commit()
1362          cur.close()
1363          con.close()
1364          return student_pass_count
1365      #不及格门数
1366      def student_fail_count(sid, year, term):
1367          student_fail_count = student_total_count(sid, year, term) - student_pass_count(sid, year,
                  term)
1368          return student_fail_count
1369      #总门数
1370      def student_total_count(sid, year, term):
1371          con, cur = connect_database()
1372          cur.execute('select count(*) from sc where sid=%s and year=%s and term=%s and grade is not
                   null', (sid, year, term))
1373          student_total_count = cur.fetchone()[0]
1374          con.commit()
1375          cur.close()
1376          con.close()
1377          return student_total_count
1378      #课程的最高分
1379      def course_max_grade(cid, year, term):
1380          con, cur = connect_database()
1381          cur.execute('select max(grade) from sc where cid = %s and year = %s and term = %s', (cid,
                  year, term))
1382          result = cur.fetchone()
1383          course_max_grade = result[0] if result else None
1384          con.commit()
1385          cur.close()
1386          con.close()
1387          return course_max_grade
1388      #课程的最低分
1389      def course_min_grade(cid, year, term):
1390          con, cur = connect_database()
1391          cur.execute('select min(grade) from sc where cid=%s and year=%s and term=%s', (cid, year,
                  term))
1392          result = cur.fetchone()
1393          course_min_grade = result[0] if result else None
1394          con.commit()
1395          cur.close()
1396          con.close()
1397          return course_min_grade
1398      #课程的平均分
1399      def course_avg_grade(cid, year, term):
1400          con, cur = connect_database()
1401          cur.execute('select avg(grade) from sc where cid = %s and year = %s and term = %s', (cid,
                  year, term))
1402          result = cur.fetchone()
```

```
1403            course_avg_grade = result[0] if result else None
1404            con.commit()
1405            cur.close()
1406            con.close()
1407            return course_avg_grade
1408        #课程的优秀率
1409        def course_excellent_rate(cid, year, term):
1410            con, cur = connect_database()
1411            cur.execute('select count(*) from sc where cid=%s and year=%s and term=%s and grade is not
                    null', (cid, year, term))
1412            total_count = cur.fetchone()[0]
1413            cur.execute('select count(*) from sc where cid=%s and year=%s and term=%s and grade>=90',
                (cid, year, term))
1414            excellent_count = cur.fetchone()[0]
1415            con.commit()
1416            cur.close()
1417            con.close()
1418            if total_count > 0:
1419                course_excellent_rate = excellent_count/total_count
1420                return format(course_excellent_rate, '.2%')
1421            else:
1422                return None
1423        #课程的及格率
1424        def course_pass_rate(cid, year, term):
1425            con, cur = connect_database()
1426            cur.execute('select count(*) from sc where cid=%s and year=%s and term=%s and grade is not
                    null', (cid, year, term))
1427            total_count = cur.fetchone()[0]
1428            cur.execute('select count(*) from sc where cid=%s and year=%s and term=%s and grade>=60',
                (cid, year, term))
1429            pass_count = cur.fetchone()[0]
1430            con.commit()
1431            cur.close()
1432            con.close()
1433            if total_count > 0:
1434                course_pass_rate = pass_count/total_count
1435                return format(course_pass_rate, '.2%')
1436            else:
1437                return None
1438        #课程的不及格率
1439        def course_fail_rate(cid, year, term):
1440            if course_pass_rate(cid, year, term) is not None:
1441                course_fail_rate = 1 - float(course_pass_rate(cid, year, term).rstrip('%'))/100
1442                return format(course_fail_rate, '.2%')
1443            else:
1444                return None
1445        #优秀人数
1446        def course_excellent_count(cid, year, term):
1447            con, cur = connect_database()
1448            cur.execute('select count(*) from sc where cid=%s and year=%s and term=%s and grade>=90',
                (cid, year, term))
1449            course_excellent_count = cur.fetchone()[0]
1450            con.commit()
1451            cur.close()
1452            con.close()
1453            return course_excellent_count
1454        #及格人数
1455        def course_pass_count(cid, year, term):
1456            con, cur = connect_database()
1457            cur.execute('select count(*) from sc where cid=%s and year=%s and term=%s and grade>=60',
                (cid, year, term))
1458            course_pass_count = cur.fetchone()[0]
```

```
            con.commit()
            cur.close()
            con.close()
            return course_pass_count
        #不及格人数
        def course_fail_count(cid, year, term):
            course_fail_count = course_total_count(cid, year, term) - course_pass_count(cid, year,
                term)
            return course_fail_count
        #总人数
        def course_total_count(cid, year, term):
            con, cur = connect_database()
            cur.execute('select count(*) from sc where cid=%s and year=%s and term=%s and grade is not
                 null', (cid, year, term))
            course_total_count = cur.fetchone()[0]
            con.commit()
            cur.close()
            con.close()
            return course_total_count
        #显示学生成绩分析
        def student_grade_window():
            student_grade_window = tk.Toplevel(analysis_grade_window)
            student_grade_window.geometry('600x600+550+150')
            student_grade_window.title('学生成绩信息分析结果')
            #获取用户输入的选课信息
            sid = sid_entry.get()
            year = year_entry.get()
            term = term_cmb.get()
            #检查选课信息有效性
            if not sid:
                messagebox.showerror('错误', '学号不能为空！')
                return
            if not year:
                messagebox.showerror('错误', '学年不能为空！')
                return
            if not term:
                messagebox.showerror('错误', '学期不能为空！')
                return
            #获取学生成绩分析计算值
            max_grade = student_max_grade(sid, year, term)
            min_grade = student_min_grade(sid, year, term)
            avg_grade = student_avg_grade(sid, year, term)
            gpa = student_gpa(sid, year, term)
            excellent_rate = student_excellent_rate(sid, year, term)
            pass_rate = student_pass_rate(sid, year, term)
            fail_rate = student_fail_rate(sid, year, term)
            excellent_count = student_excellent_count(sid, year, term)
            pass_count = student_pass_count(sid, year, term)
            fail_count = student_fail_count(sid, year, term)
            total_count = student_total_count(sid, year, term)
            result = f"""
            最高分：{max_grade}
            最低分：{min_grade}
            平均分：{avg_grade}
            绩点：{gpa}
            优秀率：{excellent_rate}
            及格率：{pass_rate}
            不及格率：{fail_rate}
            优秀门数：{excellent_count}
            及格门数：{pass_count}
            不及格门数：{fail_count}
            总门数：{total_count}
```

```
            """
            text = scrolledtext.ScrolledText(student_grade_window, width=800, height=400)
            text.insert(tk.END, result)
            text.pack()
            text.config(state=tk.DISABLED) #使用config方法将text部件设置为只读
        #显示课程成绩分析
        def course_grade_window():
            course_grade_window = tk.Toplevel(analysis_grade_window)
            course_grade_window.geometry('600x600+550+150')
            course_grade_window.title('课程成绩信息分析结果')
            #获取用户输入的选课信息
            cid = cid_entry.get()
            year = year_entry.get()
            term = term_cmb.get()
            #检查选课信息有效性
            if not cid:
                messagebox.showerror('错误', '学号不能为空！')
                return
            if not year:
                messagebox.showerror('错误', '学年不能为空！')
                return
            if not term:
                messagebox.showerror('错误', '学期不能为空！')
                return
            #获取课程成绩分析计算值
            max_grade = course_max_grade(cid, year, term)
            min_grade = course_min_grade(cid, year, term)
            avg_grade = course_avg_grade(cid, year, term)
            excellent_rate = course_excellent_rate(cid, year, term)
            pass_rate = course_pass_rate(cid, year, term)
            fail_rate = course_fail_rate(cid, year, term)
            excellent_count = course_excellent_count(cid, year, term)
            pass_count = course_pass_count(cid, year, term)
            fail_count = course_fail_count(cid, year, term)
            total_count = course_total_count(cid, year, term)
            result = f"""
            最高分：{max_grade}
            最低分：{min_grade}
            平均分：{avg_grade}
            优秀率：{excellent_rate}
            及格率：{pass_rate}
            不及格率：{fail_rate}
            优秀人数：{excellent_count}
            及格人数：{pass_count}
            不及格人数：{fail_count}
            总人数：{total_count}
            """
            text = scrolledtext.ScrolledText(course_grade_window, width=800, height=400)
            text.insert(tk.END, result)
            text.pack()
            text.config(state=tk.DISABLED) #使用config方法将text部件设置为只读
        #学号输入框
        sid_label = tk.Label(analysis_grade_window, text='学号:', font=font3)
        sid_label.pack()
        sid_entry = tk.Entry(analysis_grade_window)
        sid_entry.pack()
        #课程号输入框
        cid_label = tk.Label(analysis_grade_window, text='课程号:', font=font3)
        cid_label.pack()
        cid_entry = tk.Entry(analysis_grade_window)
        cid_entry.pack()
        #学年输入框
```

```
    year_label = tk.Label(analysis_grade_window, text='学年:', font=font3)
    year_label.pack()
    year_entry = tk.Entry(analysis_grade_window)
    year_entry.pack()
    #学期下拉菜单
    term_label = tk.Label(analysis_grade_window, text='学期:', width=17, font=font3)
    term_label.pack()
    term_cmb = ttk.Combobox(analysis_grade_window, width=15, font=font3)
    term_cmb.pack()
    term_cmb['value'] = ('第一学期', '第二学期', '小学期')
    #确定和取消按钮
    confirm_student_button = tk.Button(analysis_grade_window, text='学生成绩分析', command=
        student_grade_window, font=font3, fg='red', cursor='hand2')
    confirm_student_button.pack()
    confirm_course_button = tk.Button(analysis_grade_window, text='课程成绩分析', command=
        course_grade_window, font=font3, fg='red', cursor='hand2')
    confirm_course_button.pack()
    cancel_button = tk.Button(analysis_grade_window, text='取消', command=analysis_grade_window.
        destroy, font=font3, fg='red', cursor='hand2')
    cancel_button.pack()

root = tk.Tk()
root.geometry('900x500+250+150')
root.title('学校信息管理系统')
#字体
font1=tkfont.Font(family='华文中宋', size=20, weight=tkfont.BOLD)
font2=tkfont.Font(family='微软雅黑', size=15)
font3=tkfont.Font(family='微软雅黑', size=10)
#标题
title = tk.Label(root, text='学校信息管理系统', font=font1)
title.pack()

#6个frame
frame1 = tk.Frame(root, width=900, height=90, bd=25)
frame1.pack()
frame2 = tk.Frame(root, width=900, height=90, bd=10)
frame2.pack()
frame3 = tk.Frame(root, width=900, height=90, bd=10)
frame3.pack()
frame4 = tk.Frame(root, width=900, height=90, bd=10)
frame4.pack()
frame5 = tk.Frame(root, width=900, height=90, bd=10)
frame5.pack()
frame6 = tk.Frame(root, width=900, height=90, bd=10)
frame6.pack()

#frame1:信息管理标签
label_student = tk.Label(frame1, text='学生信息管理', font=font2)
label_student.grid(row=0, column=0)
label1_1 = tk.Label(frame1, width=11)
label1_1.grid(row=0, column=1)
label_course = tk.Label(frame1, text='课程信息管理', font=font2)
label_course.grid(row=0, column=2)
label1_2 = tk.Label(frame1, width=11)
label1_2.grid(row=0, column=3)
label_sc = tk.Label(frame1, text='选课信息管理', font=font2)
label_sc.grid(row=0, column=4)
label1_3 = tk.Label(frame1, width=11)
label1_3.grid(row=0, column=5)
label_grade = tk.Label(frame1, text='成绩信息管理', font=font2)
label_grade.grid(row=0, column=6)

```

```
#frame2:添加信息按钮
add_student_button = tk.Button(frame2, text='添加学生信息', command=add_student_window, font=font3, fg='red', padx=30, cursor='hand2')
add_student_button.grid(row=0,column=0)
label2_1 = tk.Label(frame2, width=8)
label2_1.grid(row=0, column=1)
add_course_button = tk.Button(frame2, text='添加课程信息', command=add_course_window, font=font3, fg='red', padx=30, cursor='hand2')
add_course_button.grid(row=0,column=2)
label2_2 = tk.Label(frame2, width=8)
label2_2.grid(row=0, column=3)
add_sc_button = tk.Button(frame2, text='添加选课信息', command=add_sc_window, font=font3, fg='red', padx=30, cursor='hand2')
add_sc_button.grid(row=0,column=4)
label2_3 = tk.Label(frame2, width=8)
label2_3.grid(row=0, column=5)
add_grade_button = tk.Button(frame2, text='添加成绩信息', command=add_grade_window, font=font3, fg='red', padx=30, cursor='hand2')
add_grade_button.grid(row=0,column=6)

#frame3:查询信息按钮
select_student_button = tk.Button(frame3, text='查询学生信息', command=select_student_window, font=font3, fg='red', padx=30, cursor='hand2')
select_student_button.grid(row=0, column=0)
label3_1 = tk.Label(frame3, width=8)
label3_1.grid(row=0, column=1)
select_course_button = tk.Button(frame3, text='查询课程信息', command=select_course_window, font=font3, fg='red', padx=30, cursor='hand2')
select_course_button.grid(row=0, column=2)
label3_2 = tk.Label(frame3, width=8)
label3_2.grid(row=0, column=3)
select_sc_button = tk.Button(frame3, text='查询选课信息', command=select_sc_window, font=font3, fg='red', padx=30, cursor='hand2')
select_sc_button.grid(row=0, column=4)
label3_3 = tk.Label(frame3, width=8)
label3_3.grid(row=0, column=5)
select_grade_button = tk.Button(frame3, text='查询成绩信息', command=select_grade_window, font=font3, fg='red', padx=30, cursor='hand2')
select_grade_button.grid(row=0,column=6)

#frame4:更新信息按钮
update_student_button = tk.Button(frame4, text='更新学生信息', command=update_student_window, font=font3, fg='red', padx=30, cursor='hand2')
update_student_button.grid(row=0, column=0)
label4_1 = tk.Label(frame4, width=8)
label4_1.grid(row=0, column=1)
select_course_button = tk.Button(frame4, text='更新课程信息', command=update_course_window, font=font3, fg='red', padx=30, cursor='hand2')
select_course_button.grid(row=0, column=2)
label4_2 = tk.Label(frame4, width=8)
label4_2.grid(row=0, column=3)
select_sc_button = tk.Button(frame4, text='更新选课信息', command=update_sc_window, font=font3, fg='red', padx=30, cursor='hand2')
select_sc_button.grid(row=0, column=4)
label4_3 = tk.Label(frame4, width=8)
label4_3.grid(row=0, column=5)
select_grade_button = tk.Button(frame4, text='更新成绩信息', command=update_grade_window, font=font3, fg='red', padx=30, cursor='hand2')
select_grade_button.grid(row=0,column=6)

#frame5:删除信息按钮
```

```
1689 delete_student_button = tk.Button(frame5, text='删除学生信息', command=delete_student_window, font
         =font3, fg='red', padx=30, cursor='hand2')
1690 delete_student_button.grid(row=0, column=0)
1691 label5_1 = tk.Label(frame5, width=8)
1692 label5_1.grid(row=0, column=1)
1693 select_course_button = tk.Button(frame5, text='删除课程信息', command=delete_course_window, font=
         font3, fg='red', padx=30, cursor='hand2')
1694 select_course_button.grid(row=0, column=2)
1695 label5_2 = tk.Label(frame5, width=8)
1696 label5_2.grid(row=0, column=3)
1697 select_sc_button = tk.Button(frame5, text='删除选课信息', command=delete_sc_window, font=font3, fg
         ='red', padx=30, cursor='hand2')
1698 select_sc_button.grid(row=0, column=4)
1699 label5_3 = tk.Label(frame5, width=8)
1700 label5_3.grid(row=0, column=5)
1701 select_grade_button = tk.Button(frame5, text='删除成绩信息', command=delete_grade_window, font=
         font3, fg='red', padx=30, cursor='hand2')
1702 select_grade_button.grid(row=0,column=6)
1703
1704 #frame6:成绩信息分析
1705 label6 = tk.Label(frame6, width=89)
1706 label6.grid(row=0, column=0)
1707 grade_analysis_button = tk.Button(frame6, text='成绩信息分析', command=analysis_grade_window, font
         =font3, fg='red', padx=30, cursor='hand2')
1708 grade_analysis_button.grid(row=0,column=1)
1709
1710 root.mainloop() #启动主事件循环
```

5 软件使用说明

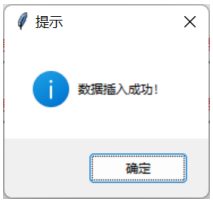
如果使用 OpenGauss 数据库运行，可以直接使用以上代码。如果使用 Sqlite 数据库运行，需要将第 11 行 connect_database() 函数前加 #，并删去第 21 行前面的 #。

用户初始界面如下：

本学校信息管理系统目前可运行功能说明如下：

5.1 学生信息管理

1. **添加学生信息**：单击“添加学生信息”按钮，输入学生的学号、姓名、性别、年龄、班级、学院、生日、籍贯信息，将这些信息存储到数据库中。



学生信息要求：

(1) 学号不能为空。

(2) 姓名不能为空。

(3) 年龄必须是整数。

(4) 生日必须是”YYYY-MM-DD” 格式的日期。

2. **查询学生信息**：单击“查询学生信息”按钮，输入学生的某一项信息或某几项信息，查询相应学生信息。

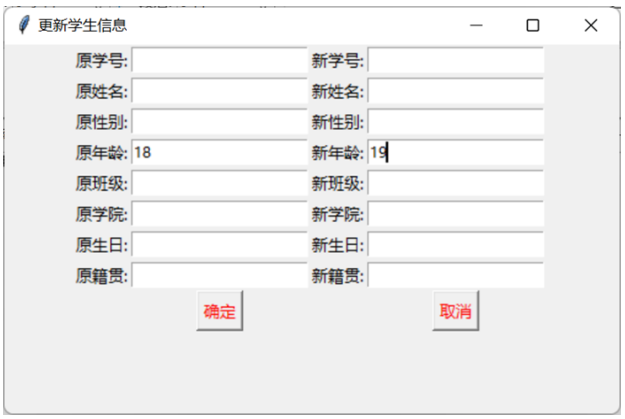
查询信息要求：

(1) 年龄必须是整数。

(2) 生日必须是”YYYY-MM-DD” 格式的日期。

在姓名栏输入某个字，会查询名字中带这个字的人。

3. **更新学生信息**：单击“更新学生信息”按钮，输入学生的某一项信息或某几项信息，更新相应学生信息。



旧值和新值不能为空。

4. **删除学生信息**：单击“删除学生信息”按钮，输入学生的某一项信息或某几项信息，删除相应学生信息。

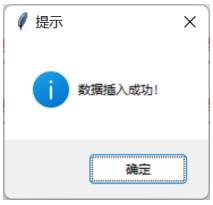
删除信息要求：

(1) 年龄必须是整数。

(2) 生日必须是”YYYY-MM-DD” 格式的日期。

5.2 课程信息管理

1. **添加课程信息**：单击“添加课程信息”按钮，输入课程名称、课程号、教师、学分、先修课程信息，将这些信息存储到数据库中。

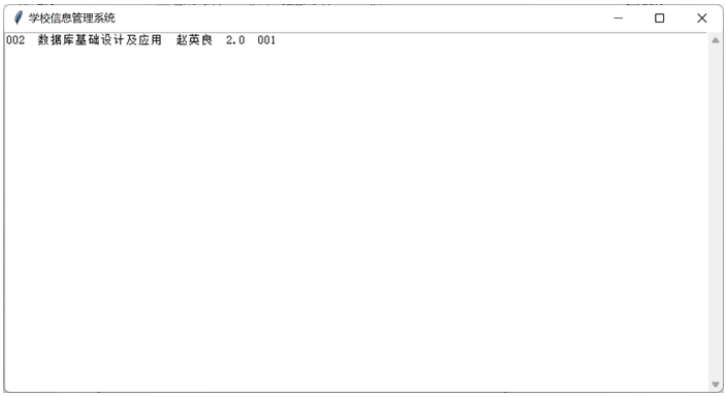


课程信息要求：

(1) 课程号不能为空。

(2) 课程名不能为空。

2. **查询课程信息**：单击“查询课程信息”按钮，输入课程的某一项信息或某几项信息，查询相应课程信息。



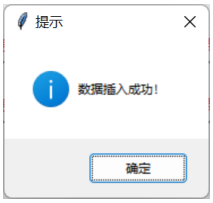
3. **更新课程信息**：单击“更新课程信息”按钮，输入课程的某一项信息或某几项信息，更新相应课程信息。

旧值和新值不能为空。

4. **删除课程信息**：单击“删除课程信息”按钮，输入课程的某一项信息或某几项信息，删除相应课程信息。

5.3 选课信息管理

1. **添加选课信息**：单击“添加选课信息”按钮，输入学号、课程号、学年、学期，将这些信息存储到数据库中。



选课信息要求：

(1) 学号不能为空。

(2) 课程号不能为空。

(3) 学年不能为空。

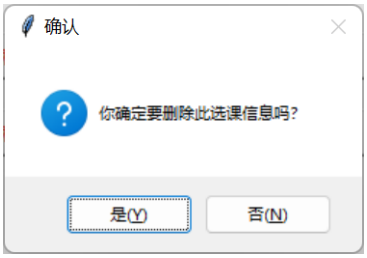
(4) 学期不能为空。

2. **查询选课信息**：单击“查询选课信息”按钮，输入选课的某一项信息或某几项信息，查询相应选课信息。

3. **更新选课信息**：单击“更新选课信息”按钮，输入选课的某一项信息或某几项信息，更新相应选课信息。

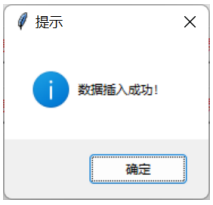
旧值和新值不能为空。

4. **删除选课信息**：单击“删除选课信息”按钮，输入选课的某一项信息或某几项信息，删除相应选课信息。



5.4 成绩管理

1. **添加成绩信息**：单击“添加成绩信息”按钮，输入学生号、课程号、学年、学期、成绩，将成绩信息存储到数据库中。



成绩信息要求：

(1) 学号不能为空。

(2) 课程号不能为空。

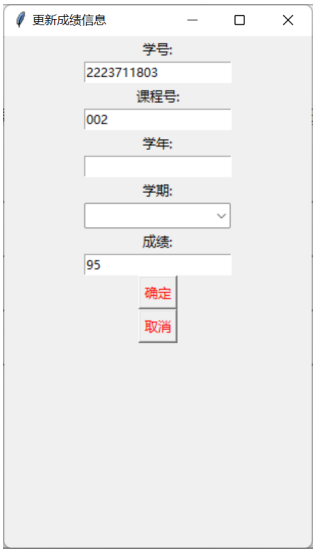
(3) 成绩必须是数值。

2. **查询成绩信息**：单击“添加成绩信息”按钮，输入选课的某一项信息或某几项信息，查询相应成绩信息。

(1) 查询学生成绩信息。

(2) 查询课程成绩信息。

3. **更新成绩信息**：单击“添加成绩信息”按钮，输入选课的某一项信息或某几项信息，更新相应成绩信息。

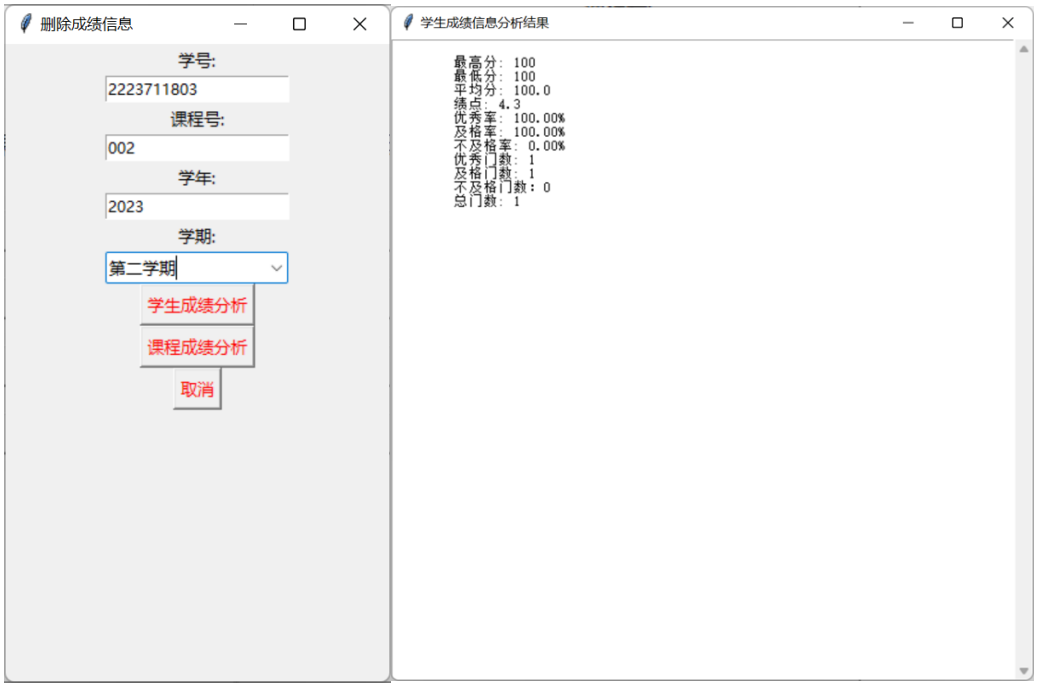


旧值和新值不能为空。

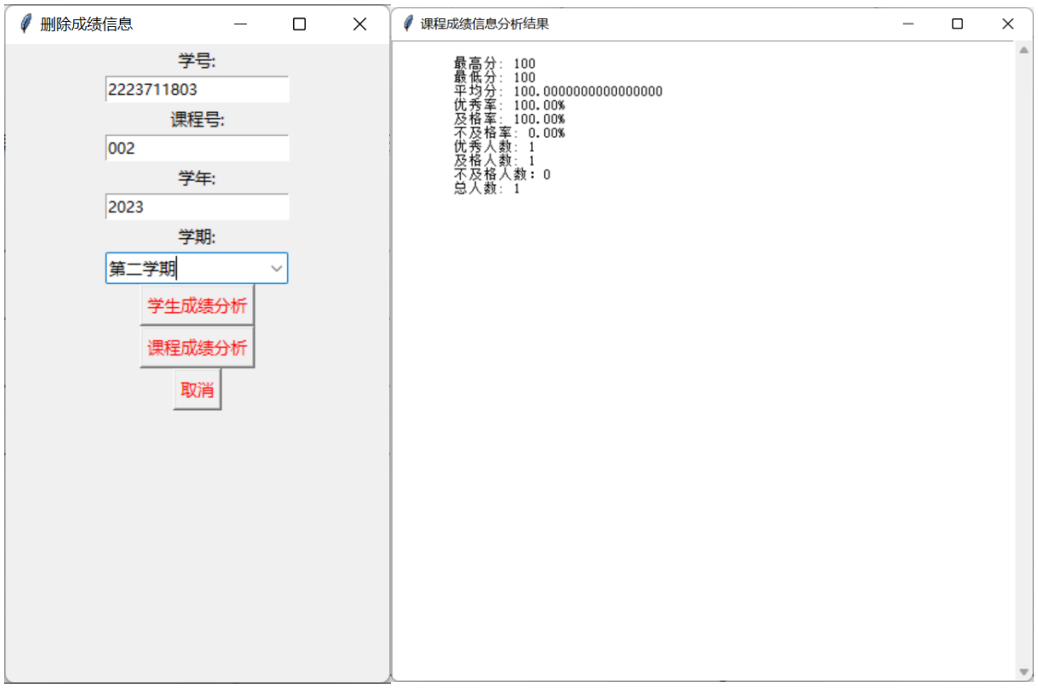
4. **删除成绩信息**：单击“添加成绩信息”按钮，输入选课的某一项信息或某几项信息，删除相应成绩信息。

5. **成绩信息分析**：单击“成绩信息分析”按钮。

(1) 学生成绩分析：输入学号、学年、学期，计算学生最高分、最低分、平均分（考虑课程学分权重）、绩点（最高 4.3）、优秀率（90 分以上）、及格率（60 分以上）、不及格率（60 分以下）、优秀门数（90 分以上）、及格门数（60 分以上）、不及格门数（60 分以下）、总门数。



(2) 课程成绩分析：输入课程号、学年、学期，计算课程最高分、最低分、平均分、优秀率（90 分以上）、及格率（60 分以上）、不及格率（60 分以下）、优秀人数（90 分以上）、及格人数（60 分以上）、不及格人数（60 分以下）、总人数。



注：绩点对应：95-100 分为 4.3，90-95 分为 4.0，85-90 分为 3.7，81-85 分为 3.3，78-81 分为 3.0，75-78 分为 2.7，72-75 分为 2.3，68-72 分为 2.0，64-68 分为 1.7，60-64 分为 1.3，小于 60 分为 0，GPA 采取平均学分绩计算方法。

6 后记

本程序是基于 Python3 的 tkinter 库设计的可视化数据库操作窗口。作为学校信息管理系统 1.0，不是最优程序，仍有很大优化空间。

可能的优化方向：

1. 输入框的循环优化。
2. 用户登录窗口的实现。
3. 成绩信息实现平时成绩、期中成绩、期末成绩的区分。
4. 信息更新与删除界面实现教师端与学生端的分离。
5. 用户界面美化。

经初步调试，本程序可正常运行。虑及本人水平有限，潜在的代码错误与冗余在所难免，用户如在使用时发现 bug，或者有关于本程序的修改意见，欢迎与我联系 wangyangdh2015@126.com。

7 致谢

感谢赵英良老师在教学过程中给予的悉心指导。