数据库工程作业

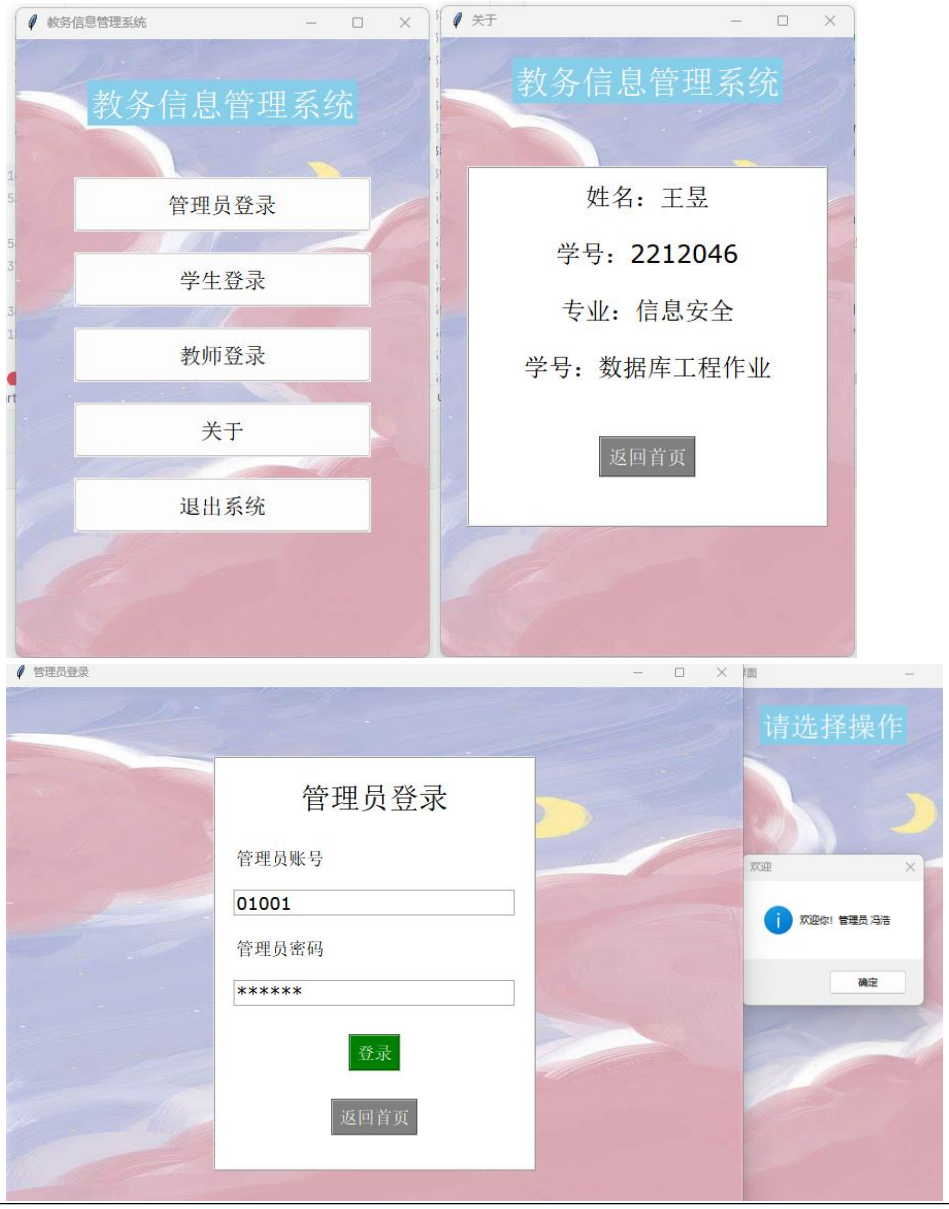
要求：

1. 完成一个小型的数据库信息管理系统（或部分功能），并填写工程作业报告；程序和报告请在规定时间之内上传。
2. 开发模式（B/S 或 C/S）、开发高级语言任选，后台数据库使用大型数据库管理系统（SQL Server、Oracle、MySQL 等），不要使用桌面数据库。
3. 报告中所列举的四种操作，每种操作举一个例子即可。
4. 作业成绩按照报告中的标准评分，程序只实现报告中涉及的部分即可。
5. 作业完成后，请将工程作业报告和程序打包提交给助教老师，并联系助教老师进行系统说明和演示，回答相关问题。

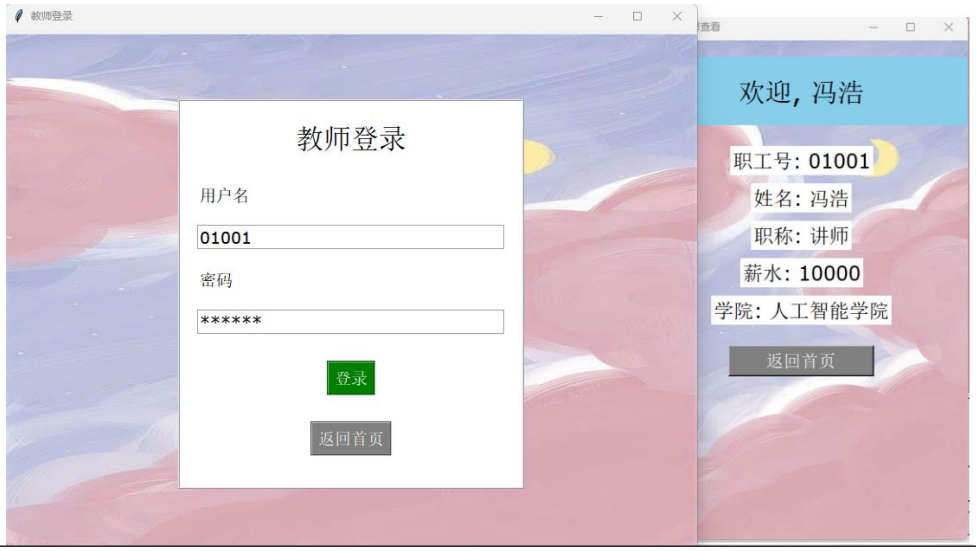
工程作业报告

1. 项目信息（10 分）

学号	2212046	姓名	王昱	专业	信息安全
项目名称	教务信息管理系统				
必备环境	Python3.11+Pymysql+Tkinter,MySQL,Navicat,PyCharm				
系统主要功能简介（4 分）	运行程序后，首先进入主界面，这里共有三种登录方式，分别是管理员登录，学生登录与教师登录，分别输入各自的 id 与密码即可进入系统。学生与教师登录进入系统后可以看到自己的基本信息。管理员登录后可以操作教师信息，学生信息，课程信息，选课信息，授课信息，以及学生课程信息总体概览。具体来说，管理员可以增删改查学生、教师、课程、选课等信息，还可以通过查询学号来查询一个学生的全部信息。总体来说，该数据库实现了如下主要的功能： 1. 学生管理 学生信息记录：存储每个学生的学号、姓名、性别、入学时间、所属学院等基本信息。 学生登录：存储学生的登录信息，包括学号和密码。 选课信息：记录学生所选的课程信息，包括课程代码、教室、成绩等。 2. 教师管理 教师信息记录：存储每个教师的职工号、姓名、职称、薪水、所属学院等基本信息。 教师登录：存储教师的登录信息，包括职工号和密码。 授课信息：记录教师所教授的课程信息，包括课程代码、教室、结课方式等。 3. 课程管理 课程信息记录：存储每门课程的课程代码、课程名称、学分、课时等基本信息。 课程安排：记录每门课程的授课教师、上课教室等信息。 4. 学院及专业管理 学院信息：存储每个学院的名称和相关信息。 专业信息：存储每个专业的专业代码、专业名称、培养方式、所属学院等信息。 5. 管理员功能 查看学生课程信息：通过视图查看所有学生的课程信息，包括学号、姓名、性别、入学日期、学院、课程 ID、课程名、学分、学时、成绩、班级号等。				

	增删改查功能：管理员可以增删改查学生、教师、课程、选课等信息。
系统主要页面截图（6 分）	

课程代码	课程名称	学分	课时
0107	留学与学术交流应用技能	2	16
0303	毛中特	2	16
0912	人工智能导论	2.5	16
1012	数据库系统	3.5	30
1017	算法设计与分析	3.5	20
1023	软件安全	2	16
1025	信安数基	3.5	20
1031	计算机组成原理	3.5	20
1329	芯片封装	2	16
3001	网球初级	1	16
3311	大学物理学（一）	4	16
L001	数据结构	3	48
L002	操作系统	3	48
L003	计算机网络	3	48
L004	数据库原理	3	48
L005	人工智能导论	2	32
L006	算法设计与分析	3	48



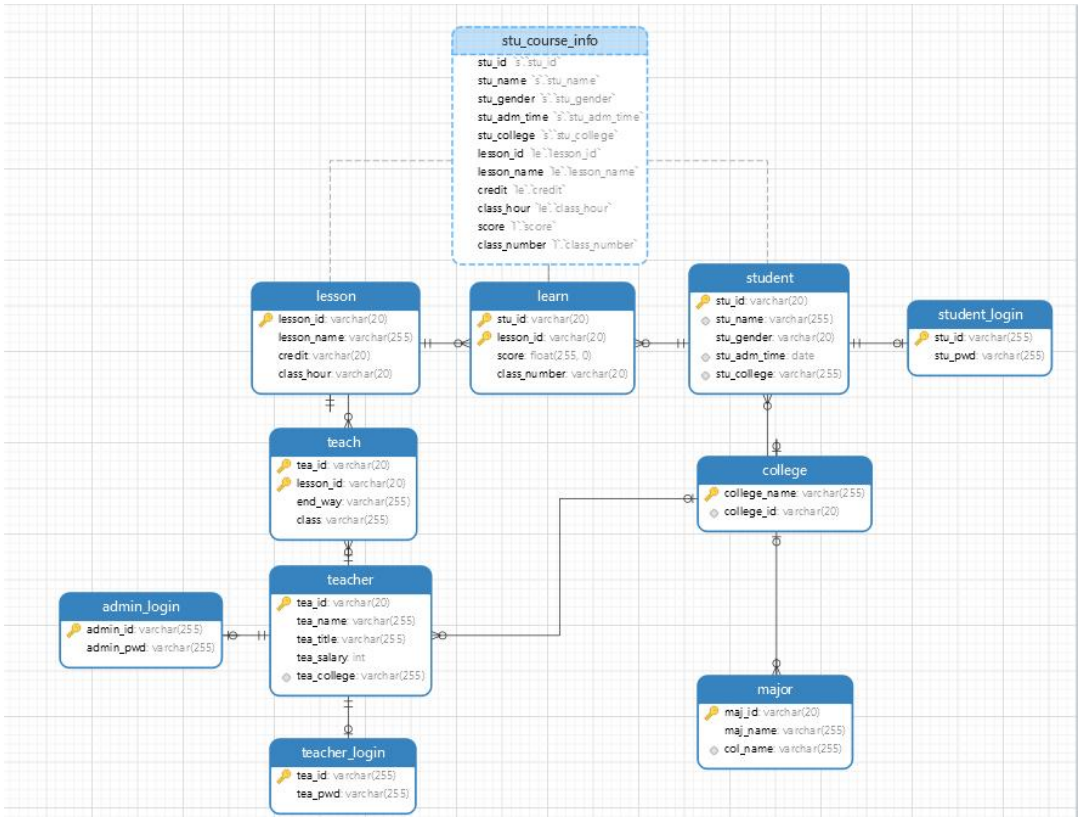
2. 系统配置（10 分）

说明		（2 分）请说明系统配置情况（后台数据库，高级语言）； （8 分）请使用连接串连接高级语言和数据库，并分析字符串的各个部分。		
配置步骤 2 分	DBMS	MySQL 8.0（用于存储表）		
	高级语言	1.python 3.11（用于开发图形用户界面）		
		2.SQL（用于与 MySQL 数据库进行交互，执行查询、插入、更新和删除操作）		
连接串分析（6 分）		序号	名称	功能说明	取值
		1	host	数据库服务器地址，指定连接的 MySQL 数据库服务器的 IP 地址或主机名	localhost
		2	user	数据库用户名，用于登录 MySQL 数据库	root

	3	password	数据库密码，用于验证数据库用户的身份	wy1211..
	4	database	数据库名称，指定要连接的具体数据库	emp_project
	5	port	数据库端口号，指定 MySQL 数据库的服务端口，默认是 3306	3306
	6	charset	字符编码	utf8
连接串代码（截屏）（2 分）				
备注				

3. 数据库设计（14 分）

说明	（10 分）按照数据表的创建顺序，依次给出所涉及数据表的信息，其中参照字段以“（字段 1，字段 2，……，字段 n）”的形式给出，被参照字段以“表名（字段 1，字段 2，……，字段 n）”的形式给出； （4 分）一般 DBMS 都可以为数据库生成关系图，请将该图片截屏并粘贴到表格中。				
数据表（10）	创建顺序	数据表名称	主键	参照属性	被参照表及属性
	1	admin_login	admin_id	admin_id	teacher(tea_id)
	2	college	college_name	-	-
	3	lesson	lesson_id	-	-
	4	student	stu_id	-	-
	5	learn	stu_id	stu_id	student(stu_id)
			lesson_id	lesson_id	lesson(lesson_id)
	6	major	maj_id	col_name	college(college_name)
	7	student_login	stu_id	stu_id	student(stu_id)
	8	teach	tea_id	tea_id	teacher(tea_id)
			lesson_id	lesson_id	lesson(lesson_id)
	9	teacher	tea_id	tea_college	college(college_name)
	10	teacher_login	tea_id	tea_id	teacher(tea_id)

关系图（4）	
备注	图中钥匙代表主键，灰色标记代表外键，连接的线表示表与表之间的关系。

4. **含有事务应用的删除操作（13 分）**

说明	（1 分）简要说明该操作所要完成的功能； （2 分）该操作会涉及的表（必须含有两张或两张以上的关系表，同时以“表名”的形式给出） （1 分）表连接涉及字段描述（描述方式为“表 1.属性=表 2.属性”） （1 分）删除条件涉及的字段描述(以“表名.属性=？”形式给出) （4 分）实现该操作的关键代码（高级语言、SQL），截图即可；（其中如果删除语句中不包含任何形式的事务应用将扣除 3 分） （4 分）如何执行该操作，按所述方法能够正常演示程序则给分。
功能描述（1 分）	我们有三个表：student、student_login 和 learn。当删除 student 表中的一条记录时，同时删除 student_login 表中对应的登录信息和 learn 表中对应的选课信息。这些删除操作在一个事务中执行，以确保数据的一致性。如果任何删除操作失败，就回滚（ROLLBACK）整个事务，以保持数据库的完整性。
涉及的表（2 分）	涉及的表：student,student_login,learn
表连接涉及字段	三个表通过学生 id（stu_id）进行连接： student.stu_id=student_login.stu_id student.stu_id=learn.stu_id

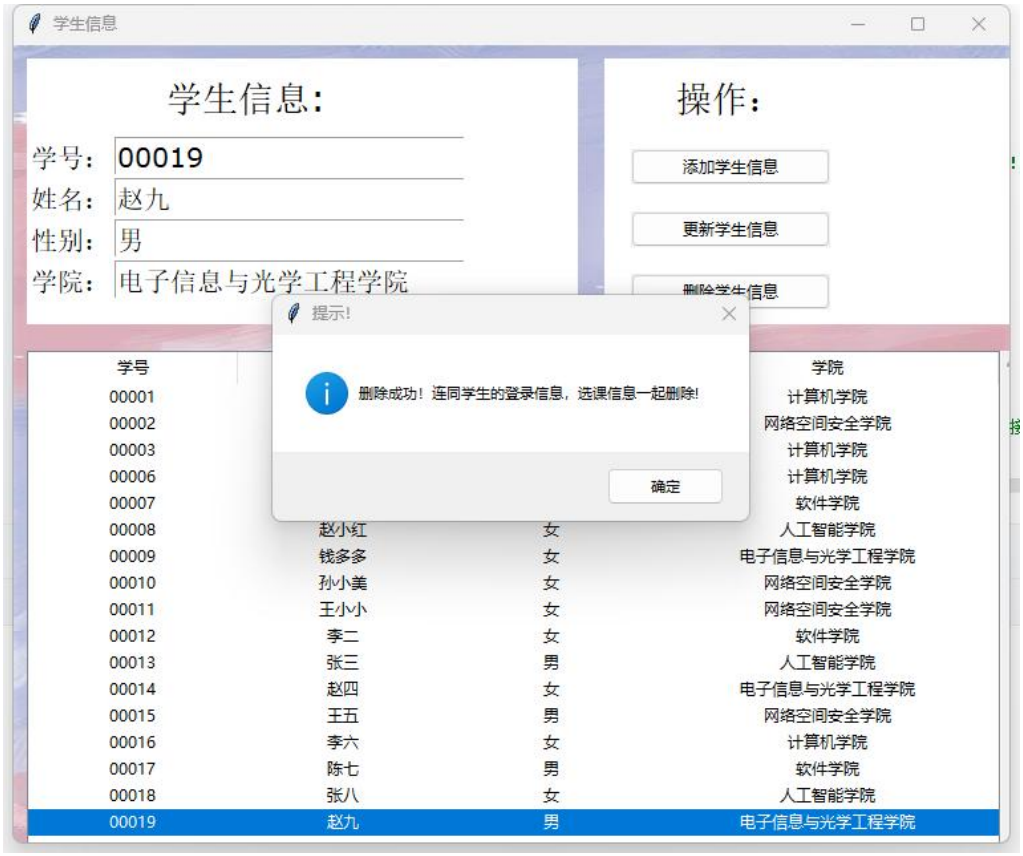
<table>
<tr><td>（1分）</td><td></td></tr>
<tr><td rowspan="4">删除条件字段描述（1分）</td><td>字段</td><td>规则</td></tr>
<tr><td>student.stu_id</td><td>这里填入要删除学生的 ID</td></tr>
<tr><td>student_login.stu_id</td><td>判断是否 student_login.stu_id=student.stu_id，若相等则删除对应记录</td></tr>
<tr><td>learn.stu_id</td><td>判断是否 learn.stu_id=student.stu_id，若相等则删除对应记录</td></tr>
<tr><td>代码（4分）</td><td colspan="2">触发器代码：

```
DELIMITER //
CREATE PROCEDURE delete_student_with_related_records(IN student_id VARCHAR(20))
BEGIN
    DECLARE total_records INT;
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        -- 发生异常时回滚事务
        ROLLBACK;
    END;

    START TRANSACTION;
    DELETE FROM student_login WHERE stu_id = student_id;
    DELETE FROM learn WHERE stu_id = student_id;
    DELETE FROM student WHERE stu_id = student_id;
    SET total_records = (SELECT COUNT(*) FROM student WHERE stu_id = student_id)
                      + (SELECT COUNT(*) FROM student_login WHERE stu_id = student_id)
                      + (SELECT COUNT(*) FROM learn WHERE stu_id = student_id);
    IF total_records = 0 THEN
        COMMIT;
    ELSE
        ROLLBACK;
    END IF;
END
//
DELIMITER ;
```

删除操作代码：

```
def del_row(self):
    res = messagebox.askyesnocancel( title: '警告！', message: '是否删除所选数据？')
    if res:
        db = pymysql.connect(host="localhost", user="root", password="wy1211..", database="emp_project", po
        cursor = db.cursor()
        sql="BEGIN"
        sql1 = "DELETE FROM student WHERE stu_id = '%s'" % (self.row_info[0])
        sql2 = "DELETE FROM student_login WHERE stu_id =' %s'" % (self.row_info[0])
        sql3 = "DELETE FROM learn WHERE stu_id = '%s'" % (self.row_info[0])
        sql4 = "COMMIT"
        try:
            cursor.execute(sql)
            cursor.execute(sql1)
            cursor.execute(sql2)
            cursor.execute(sql3)
            cursor.execute(sql4)
            db.commit()
            messagebox.showinfo( title: '提示！', message: '删除成功！连同学生的登录信息，选课信息一起删除！')
```
</td></tr>
</table>

程序演示（4分）	 随后查询该学生选课信息，可以看到所有记录已经被删除掉了： 
备注	通过在触发器中使用 START TRANSACTION、COMMIT 和 ROLLBACK，可以确保在发生异常时回滚事务，保持数据库的一致性。如果没有异常，事务将被提交，所有删除操作都会生效。

5. **触发器控制下的添加操作（20 分）**

<table>
<tr><td>说明</td><td colspan="2">（1 分）简要说明该操作所要完成的功能；
（2 分）简要说明该触发器所要完成的功能
（1 分）该操作会涉及的表（以“表名”的形式给出）。
（2 分）该操作输入数据以及输入数据应该满足的条件，如：数值范围、是否为空；
（6 分）实现该操作的关键代码（高级语言、SQL），截图即可；
（8 分）如何执行该操作，按所述方法能够正常演示程序则给分。</td></tr>
<tr><td>功能描述
（1 分）</td><td colspan="2">该功能实现了在选课信息界面中显示学生的选课信息，包括学生学号、课程代码、教室、成绩和学分，同时可以进行选课信息的插入、更新和删除操作，同时保证了每个学生在选了新的课之后总学分不超过 20 学分。</td></tr>
<tr><td>触发器描述
（2 分）</td><td colspan="2">在执行插入操作时，触发器会检查加入了新课程的总学分是否大于 20，如果大于 20 那么就回滚，使每个学生的总学分不超过 20 学分。</td></tr>
<tr><td>涉及的表
（1 分）</td><td colspan="2">learn、lesson</td></tr>
<tr><td rowspan="4">输入数据
（2 分）</td><td>字段</td><td>规则</td></tr>
<tr><td>learn.lesson_id</td><td>输入数据，通过课程 id 在 lesson 表中寻找该课程的学分</td></tr>
<tr><td>learn.stu_id</td><td>输入数据，与 lesson_id 结合起来计算当前学生的总学分是多少，判断是否大于 20，若大于 20 则违背触发器，不能执行插入操作。（如果是第一次插入，那么总学分默认为 0）</td></tr>
<tr><td>lesson.credit</td><td>通过 lesson_id 在 lesson 表中查询到 credit</td></tr>
<tr><td>插入操作
源码
（3 分）</td><td colspan="2">

```
sql4 = ("SELECT SUM(lesson.credit) "
        "FROM learn JOIN lesson ON learn.lesson_id = lesson.lesson_id "
        "WHERE learn.stu_id = %s")
cursor.execute(sql4, args: (self.var_id.get(),))
total_credits = cursor.fetchone()[0]
if total_credits is None:
    total_credits = 0
sql5 = ("SELECT lesson.credit "
        "FROM lesson "
        "WHERE lesson.lesson_id = %s")
cursor.execute(sql5, args: (self.var_les.get(),))
new_credits = cursor.fetchone()[0]

db.close()

if self.var_id.get() == '' or self.var_les.get() == '':
    messagebox.showinfo( title: '警告！', message: '请输入选课信息')
elif total_credits + float(new_credits) > 20:
    messagebox.showinfo( title: '警告！', message: '总学分不能超过20学分！')
```

</td></tr>
</table>

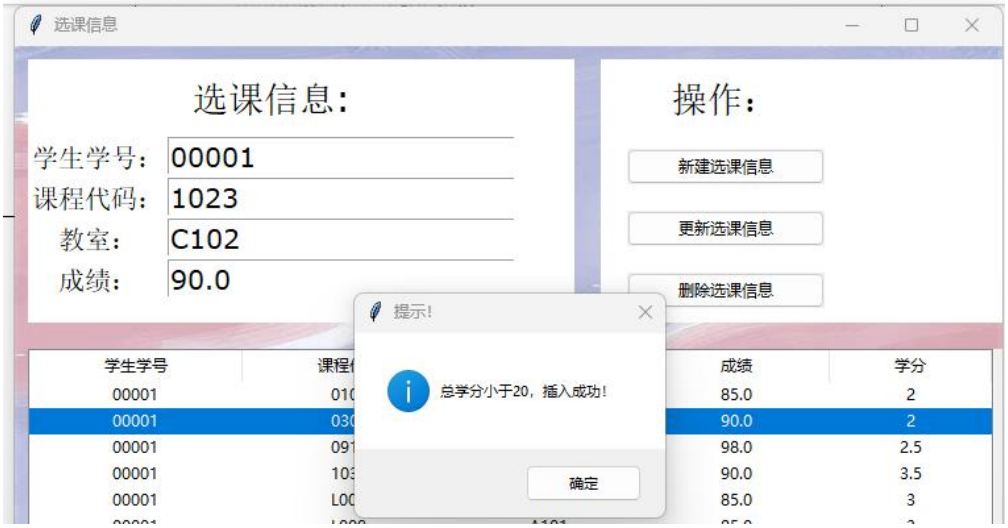
```
else:
    if self.var_id.get() not in stu_id or self.var_les.get() not in les_id:
        messagebox.showinfo( title: '警告！', message: '该学生或该课程不存在！')
    elif self.var_les.get() not in les_id:
        messagebox.showinfo( title: '警告！', message: '无此课程！')
    else:
        db = pymysql.connect(host="localhost", user="root", password="wy1211..", database="emp_project",
                             port=3306)
        cursor = db.cursor()  # 使用cursor()方法获取操作游标
        sql = "INSERT INTO learn(stu_id, lesson_id, score, class_number) VALUES (%s, %s, %s, %s)"
        try:
            cursor.execute(sql, args: (self.var_id.get(), self.var_les.get(), self.var_scr.get(), self.var_cla.ge
            db.commit()
            self.id.append(self.var_id.get())
            self.les.append(self.var_les.get())
            self.cla.append(self.var_cla.get())
            self.scr.append(self.var_scr.get())
            self.tree.insert( parent: '', len(self.id) - 1, values=(
                self.id[len(self.id) - 1], self.les[len(self.id) - 1], self.cla[len(self.id) - 1],
                self.scr[len(self.id) - 1], new_credits))
            self.tree.update()
            messagebox.showinfo( title: '提示！', message: '总学分小于20，插入成功！')
        except Exception as e:
            db.rollback()
            messagebox.showinfo( title: '警告！', message: f'总学分大于20，插入失败！{e}')
        db.close()
```

触发器源码（3分）

```
DROP TRIGGER IF EXISTS `before_learn_insert`;
delimiter ;;
CREATE TRIGGER `before_learn_insert` BEFORE INSERT ON `learn` FOR EACH ROW BEGIN
    DECLARE total_credits INT;
    SET total_credits = (SELECT SUM(le.credit)
                         FROM learn l
                         JOIN lesson le ON l.lesson_id = le.lesson_id
                         WHERE l.stu_id = NEW.stu_id);
    SET total_credits = total_credits +
    (SELECT le.credit
    FROM lesson le
    WHERE le.lesson_id = NEW.lesson_id);
    -- 检查总学分是否超过 20
    IF total_credits > 20 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '学生选课总学分不能超过 20 学分';
    END IF;
END
;;
delimiter ;
```

程序演示（4分）

当前学号为00001学生已经有了17学分，若插入2学分的软件安全课（1023）：

<table>
<tr><td></td><td> 
</td></tr>
<tr><td>程序演示
（4分）</td><td>此时已经19学分，若再插入5学分的高等数学课(L011)：
 
</td></tr>
<tr><td>备注</td><td>该触发器保证了每个学生的课总学分都不会超过20学分。除了这个触发器以外，我还设置了其他关于增加的触发器，例如当插入一个新的学生信息的时候，会自动为学生创建一个账户（在student表添加该学生，默认密码为123456）。</td></tr>
</table>

6. **存储过程控制下的更新操作（18 分）**

<table>
<tr><td>说明</td><td>（1 分）简要说明该操作所要完成的功能；
（1 分）简要说明该存储过程所要完成的功能；
（2 分）说明该操作涉及操作的表（必须包含两张或两张以上的关系表，以“表名形式”描述）
（1 分）表连接涉及字段描述（描述方式为“表 1.属性=表 2.属性”）
（2 分）该操作会修改字段（以“表名.字段名”的形式给出），以及修改规则，如新数值的计算方法、在何种条件下予以修改等；
（6 分）实现该操作的关键代码（高级语言、SQL），截图即可；
（5 分）如何执行该操作，按所述方法能够正常演示程序则给分。</td></tr>
<tr><td>功能描述(1 分)</td><td>更新学生的学员信息，也就是转专业操作的时候，要保证学生的平均学分绩大于 80 分才可以转专业；未达到要求的将拒绝转专业。</td></tr>
<tr><td>存储过程功能描述（1 分）</td><td>该存储过程要求传入学生 id 和专业名称作为输入参数，来更新学生的专业名称。它会在更新之前，计算该学生的平均分数，并检查是否达到 80 分。如果学生的平均分数低于 80 分，则更新失败。</td></tr>
<tr><td>涉及的关系表（2 分）</td><td>涉及到的表：
student,learn,college</td></tr>
<tr><td>表连接涉及字段（1）</td><td>连接的字段：
student.stu_id=learn.stu_id</td></tr>
<tr><td>更改字段（2 分）</td><td><table><tr><td>字段</td><td>规则</td></tr><tr><td>student.college</td><td>修改学生的学院，要保证其平均学分绩大于 80 分才可以修改，并且要转入的专业要存在</td></tr></table></td></tr>
<tr><td>更新代码（3 分）</td><td><pre>
IF avg_score >= 80 THEN
    START TRANSACTION;
    UPDATE student
    SET student.stu_college= new_col
    WHERE student.stu_id = student_id;
    COMMIT;
</pre></td></tr>
</table>

<table>
<tr><td></td><td>

```
def updata_row(self):
    res = messagebox.askyesnocancel( title: '警告！', message: '是否更新所填数据？')
    if res:
        if self.var_id.get() == self.row_info[0]:
            db = pymysql.connect(host="localhost", user="root", password="wy1211..", database="emp_project",
                                 port=3306)
            cursor = db.cursor()
            sql = "UPDATE student SET stu_name = %s, stu_gender = %s, stu_college = %s WHERE stu_id = %s"
            sql_update_major = "CALL update_student_major(%s, %s)"
            try:
                if self.var_college.get() != self.row_info[3]:
                    cursor.execute(sql_update_major, args: (self.var_id.get(), self.var_college.get()))
                cursor.execute(sql, args: (
                self.var_name.get(), self.var_gender.get(), self.var_college.get(), self.var_id.get()))
                db.commit()
                messagebox.showinfo( title: '提示！', message: '转专业成功！')

                id_index = self.id.index(self.row_info[0])
                self.name[id_index] = self.var_name.get()
                self.gender[id_index] = self.var_gender.get()
                self.college[id_index] = self.var_college.get()

                self.tree.item(self.tree.selection()[0], values=(
                    self.var_id.get(), self.var_name.get(), self.var_gender.get(), self.var_college.get()))
            except pymysql.MySQLError as e:
                db.rollback()
                messagebox.showinfo( title: '警告！', str(e))
            finally:
                db.close()
```

</td></tr>
<tr><td>创建存储过程源码（3分）</td><td>

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `update_student_major`(
    IN student_id VARCHAR(20),
    IN new_col  VARCHAR(255)
)
BEGIN
    DECLARE avg_score FLOAT;
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '更新失败：平均分数未达到80分';
    END;
    SELECT AVG(sc.score) INTO avg_score
    FROM learn sc
    WHERE sc.stu_id = student_id;
    IF avg_score >= 80 THEN
        START TRANSACTION;
        UPDATE student
        SET student.stu_college= new_col
        WHERE student.stu_id = student_id;
        COMMIT;
    ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '更新失败：平均分数未达到80分';
    END IF;
END
```

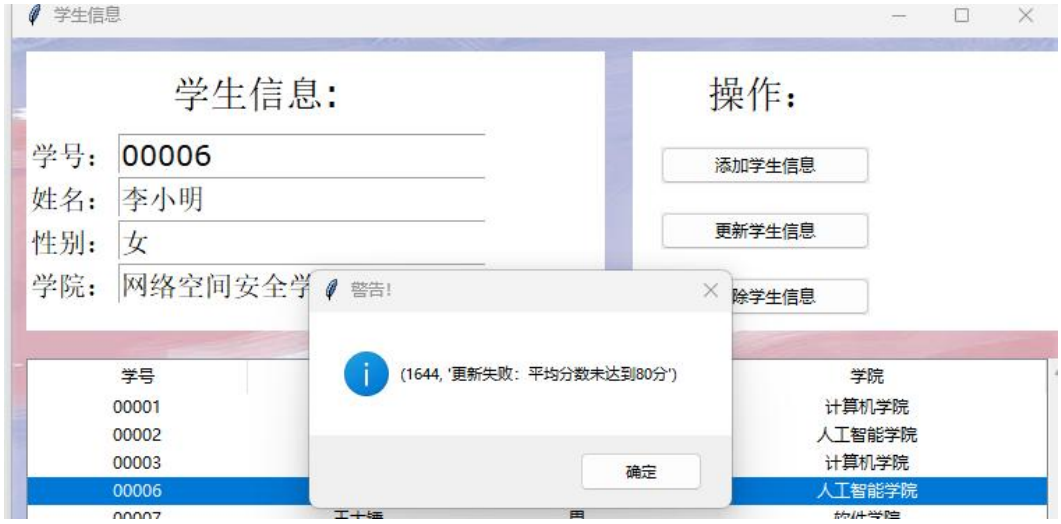
</td></tr>
<tr><td>存储过程执行源码（1分）</td><td>

```
sql2 = "CALL update_class_number_and_teach(%s,%s,%s)"
try:
    cursor.execute(sql, args: (self.var_id.get(), self.var_les.get(), self.var_scr.get()))
    cursor.execute(sql2, args: (self.var_id.get(),self.var_les.get(),self.var_cla.get()))
    db.commit()
```

</td></tr>
<tr><td>程序演示(2分）</td><td>说明：该学生平均成绩大于 80 分</td></tr>
</table>

<table>
<tr><td></td><td> 

转学院成功：

</td></tr>
<tr><td>程序演示
（2 分）</td><td>说明：该学生平均成绩小于 80 分：
 
</td></tr>
<tr><td>备注</td><td>这样保证了只有平均学分绩大于 80 的人才可以转专业</td></tr>
</table>

7. **含有视图的查询操作（15 分）**

<table>
<tr><td>说明</td><td>（1 分）简要说明该操作所要完成的功能；
（1 分）简要说明建立的该视图的功能；
（2 分）简要说明该操作涉及的关系数据表（以“表名”的形式给出）
（1 分）简要说明表连接涉及的字段（以“表 1.属性=表 2.属性”）
（6 分）实现该操作的关键代码（高级语言、SQL），截图即可；
（4 分）如何执行该操作，按所述方法能够正常演示程序则给分。</td></tr>
<tr><td>操作功能描述(1分)</td><td>该操作创建一个视图，该视图涵盖了学生的全部信息以及他所选修课程的信息，通过多个表连接进行操作。</td></tr>
<tr><td>视图功能描述(1分)</td><td>该视图连接了三个表，让我们可以更加直观地看到学生课程信息的一个涵盖。</td></tr>
<tr><td>涉及的关系表(2分)</td><td>learn,student,lesson</td></tr>
<tr><td>表连接字段（1 分）</td><td>三个表连接的方式如下:
student.stu_id = learn.stu_id
learn.lesson_id = lesson.lesson_id</td></tr>
<tr><td>创建视图代码(3分)</td><td><pre>select `s`.`stu_id` AS `stu_id`,`s`.`stu_name` AS `stu_name`,`s`.`stu_gender` AS `stu_gender`,`s`.`stu_adm_time` AS `stu_adm_time`,`s`.`stu_college` AS `stu_college`,`le`.`lesson_id` AS `lesson_id`,`le`.`lesson_name` AS `lesson_name`,`le`.`credit` AS `credit`,`le`.`class_hour` AS `class_hour`,`l`.`score` AS `score`,`l`.`class_number` AS `class_number` from ((`student` `s` join `learn` `l` on((`s`.`stu_id` = `l`.`stu_id`))) join `lesson` `le` on((`l`.`lesson_id` = `le`.`lesson_id`)))</pre></td></tr>
<tr><td>查询代码（3 分）</td><td><pre>def load_data(self):
    # 打开数据库连接
    db = pymysql.connect(host="localhost", user="root", password="wy1211..", database="emp_project", port=330
    cursor = db.cursor()
    sql = ("SELECT * "
           "FROM stu_course_info")
    try:
        cursor.execute(sql)
        results = cursor.fetchall()
        for row in results:
            self.tree.insert(parent: '', index: 'end', values=row)
    except Exception as e:
        print("Error: unable to fetch data")
        messagebox.showinfo(title: '警告！', message: f'数据库连接失败！\n错误信息：{e}')
    db.close() # 关闭数据库连接</pre></td></tr>
</table>

程序演示 （4 分）	
备注	输入学号还可以进行查询某个人的信息：