

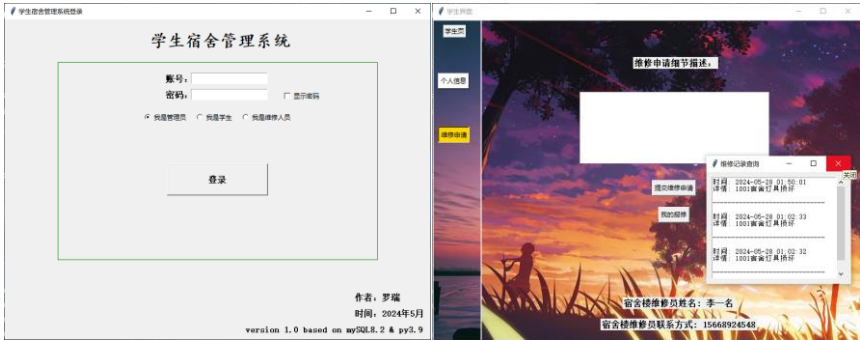
数据库工程作业

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）

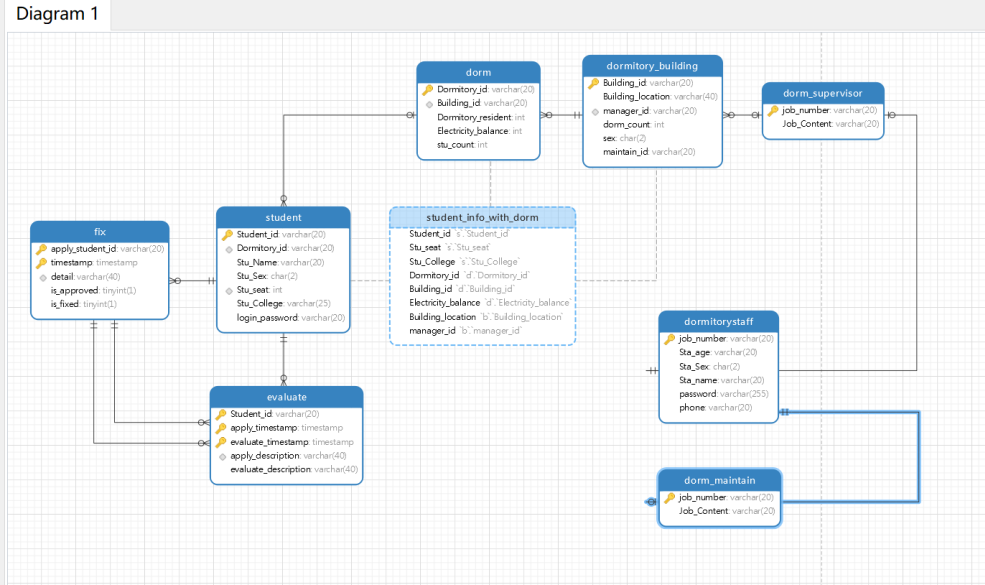
学号	2210529	姓名	罗瑞	专业	密码科学与技术
项目名称	学生宿舍管理系统				
必备环境	MySQL8.2, Python 3.9				
系统主要功能简介（4 分）	<p>该系统为一个完整的学生宿舍管理系统，分为三个用户模式，且具有完整的权限管理。应用情景为一所学校有两个建筑集群，有若干宿舍楼，每栋宿舍楼配有一名管理人员和一名维修人员，学生拥有已分配宿舍和未分配宿舍两种状态。</p> <p>功能描述：</p> <p><1>管理员模式：二级管理员拥有对应宿舍楼的操作权限，一级管理员拥有所有宿舍楼的操作权限。操作种类有：（对学生操作）模糊查询匹配学生、审批学生的报修、重置学生的系统密码、更新学生的一些基本信息、更换学生的宿舍或床位；（对宿舍操作）模糊查询匹配宿舍，查询宿舍的所有基本信息，给指定学生分配到当前宿舍、将指定学生移出当前宿舍、增加宿舍、删除宿舍；（对自己操作）查看、修改管理员信息。</p> <p><2>学生模式：宿舍报修、查看保修状态、评价维修；查看、修改个人信息、查看舍友信息。</p> <p><3>维修人员模式：查看报修、修改报修状态、查看维修评价；查看、修改个人信息</p>				
系统主要页面截图（6 分）					



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	1. MySQL 8.0.32 存储各张表数据。 2. SQL 语句建立各种触发器或储存结构、视图等。			
	高级语言	1. python 用于建立客户端系统的可视化操作界面，且用于客户端系统与数据库系统的连接与进行各种增、删、查、改操作的“指挥”。			
连接串 分析 (6 分)		序 号	名 称	功能说明	取 值
		1	user	数据库账号	'root'
		2	password	数据库密码	'123456'
		3	host	指定服务器地址	'localhost'
		4	database	指定数据库中的 mydormitory 数据库系统	'mydormitory'
连接串代码 (截屏) (2 分)					
备注		端口默认为 3306、且查询返回格式在高级语言中默认为元组，这里不需要再特别设置，所以本部分只需要较少的设置就能完全达到开发目的。			

3. 数据库设计（14 分）

说明	<p>（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；</p> <p>（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。</p>				
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	dormitory_building	Building_id	无	无
	2	dorm	Dormitory_id	Building_id	dormitory_building (Building_id)
	3	dormitorystaff	job_number	无	无
	4	dorm_supervisor	job_number	job_number	Dormitorystaff (job_number)
	5	dorm_maintain	job_number	job_number	Dormitorystaff (job_number)
	6	Student	Student_id	Dormitory_id	Dorm (Dormitory_id)
	7	fix	(apply_student_id, timestamp)	apply_student_id	Student (Student_id)
	8	evaluate	(Student_id, apply_timestamp, evaluate_timestamp)	(Student_id, apply_timestamp, apply_description)	Fix (apply_student_id, Timestamp, detail)
关系图 (4)	<p>Diagram 1</p> 				
备注	<p>Fix 是以申请报修学生 id 和时间戳组合为主键的，evaluate 是外键引用 Fix 主键的基础上再组合评价时间戳为主键的，这是因为评价是在一次报修的基础上，但是可能出现一对多的情况。一栋宿舍对应一个维修人员，维修人员的界面中只会显示对应楼宇的学生报修申请和相应维修评价。</p>				

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)	此操作将删除指定宿舍及其相关的所有记录。当我们提供一个宿舍号作为输入时, 它将从四个表 (student,dorm) 中删除与该宿舍号相关的所有记录, 另外该宿舍所在的宿舍楼的宿舍个数属性也会自动减 1。这是一个事务操作, 这意味着如果在删除过程中的任何一步发生错误, 所有的删除操作都将被回滚, 数据库将返回到操作开始之前的状态。	
涉及的表 (2 分)	student、dorm、dormitory_building	
表连接涉及字段 (1 分)	Student.dormitory_id=Dorm.dormitory_id Dorm.building_id=dormitory_building.building_id	
删除条件 字段描述 (1 分)	字段	规则
	Student.dormitory_id	Student.dormitory_id=NULL
	Student.stu_seat	Student.stu_seat=NULL
	Dorm.Dormitory_id	delete
	Dorm.Building_id	delete
	Dorm.Dormitory_resident	delete
	Dorm.Electricity_balance	delete
	Dorm.stu_count	delete
	Dormitory_building.dorm_count	dorm_count=dorm_count-1

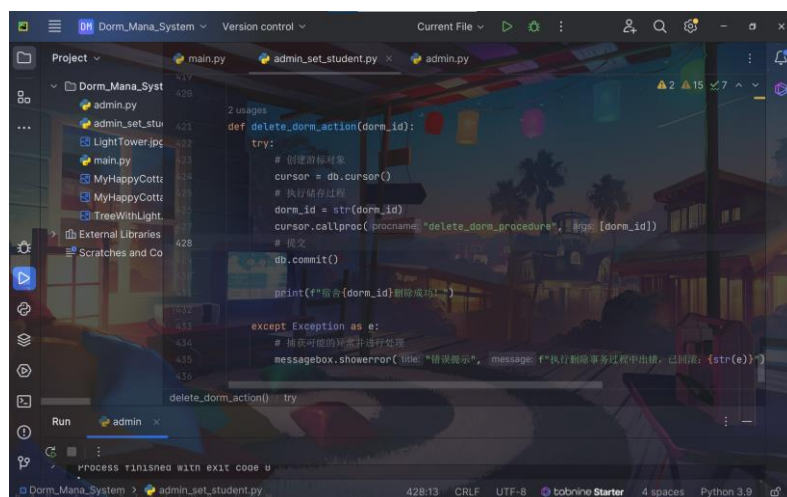
代码
(4分)

```

保存 查询创建工具 美化 SQL 代码段 创建图表
localhost 3306 mydormitory 运行 停止 解释

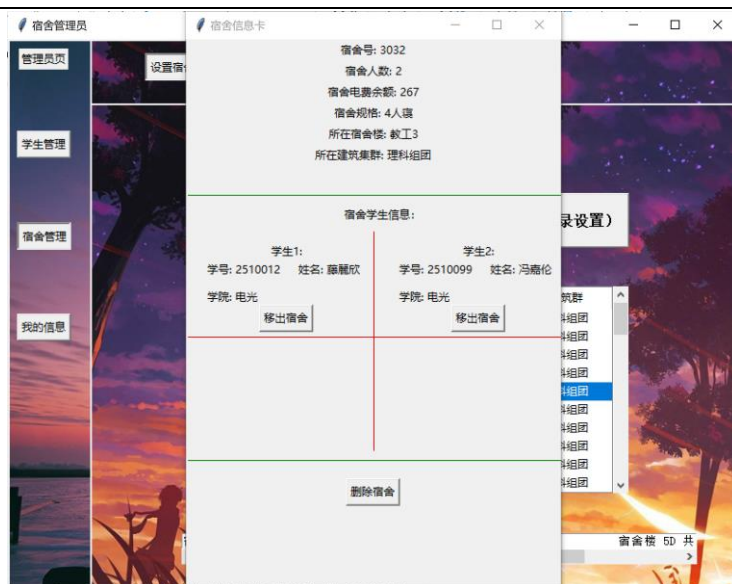
1 DELIMITER //
2
3 -- 重新创建存储过程
4 CREATE PROCEDURE delete_dorm_procedure(IN dorm_id VARCHAR(20))
5 BEGIN
6     DECLARE v_building_id VARCHAR(20);
7     DECLARE v_dorm_count INT;
8     -- 获取被删除宿舍所在的宿舍楼ID及其宿舍数量
9     SELECT building_id, dorm_count INTO v_building_id, v_dorm_count FROM
dormitory_building WHERE building_id = (SELECT building_id FROM dorm WHERE
dormitory_id = dorm_id);
10    -- 开始事务
11    START TRANSACTION;
12    -- 更新宿舍所在宿舍楼的宿舍数量减1
13    UPDATE dormitory_building SET dorm_count = v_dorm_count - 1 WHERE building_id =
v_building_id;
14    -- 更新宿舍所有学生的宿舍号与床位位置为空
15    UPDATE student SET dormitory_id = NULL, stu_seat = NULL WHERE dormitory_id =
dorm_id;
16    -- 删除宿舍
17    DELETE FROM dorm WHERE dormitory_id = dorm_id;
18    -- 提交事务
19    COMMIT;
20 END //
21
22 DELIMITER ;
23

```

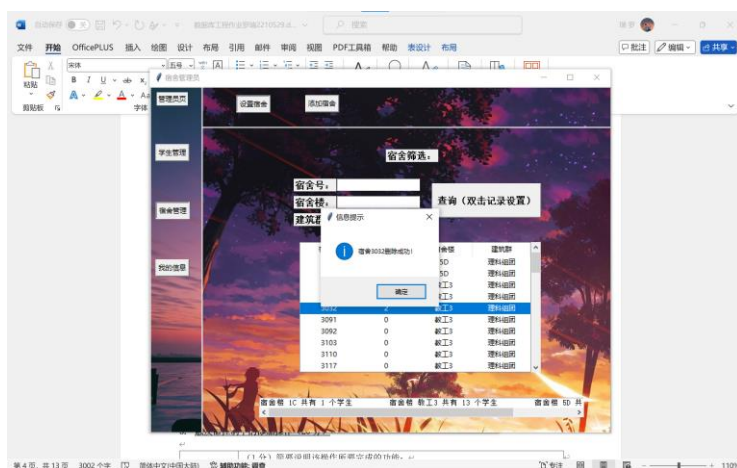


程序
演示
(4分)





点击删除宿舍：



再次点击查询刷新，发现刚才的宿舍 3032 对应的记录消失，并且注意到底部的滚动框教工 3 宿舍楼宿舍人数由 13 减少为 11：



此时还应注意，查询刚才删除掉的宿舍中的两个学生，均显示无宿舍状态

该操作有一定权限，如果当前管理人员不是一级管理员或者宿舍所在宿舍楼对应的二级管理员，则无操作权限。这里为控制篇幅只简单展示：

备注

5. 触发器控制下的添加操作（20 分）

说明	<p>(1 分) 简要说明该操作所要完成的功能；</p> <p>(2 分) 简要说明该触发器所要完成的功能</p> <p>(1 分) 该操作会涉及的表（以“表名”的形式给出）。</p> <p>(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；</p> <p>(6 分) 实现该操作的关键代码（高级语言、SQL），截图即可；</p> <p>(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。</p>	
功能描述 (1 分)	学生添加维修表中的保修记录时，为了保证系统资源稳定且合理，要限制同一个同学提交的报修记录在连续的 24 小时内不能超过 3 条。	
触发器描述 (2 分)	触发器的功能是在学生向维修表中添加记录时，系统会先检查维修表中由该学生提交的记录的个数和相应的时间戳信息，若在从当前时刻向前计时的 24 小时内该学生已提交过三条报修记录，则当前抛出错误，阻止记录插入操作，否则提交记录成功。	
涉及的表 (1 分)	Fix	
输入数据 (2 分)	字段	规则
	Fix_apply_student_id	Fix_apply_student_id=self.id(Stu_id)
	detail	Fix.detail=detail

插入操作
源码
(3分)

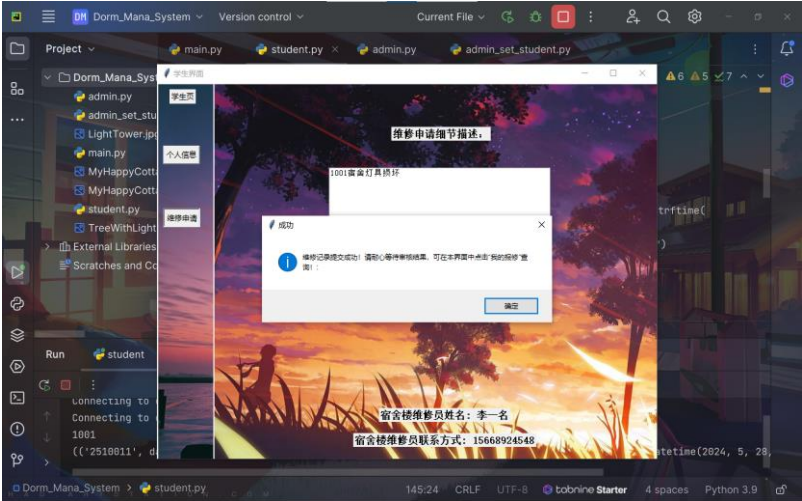
```
main.py student.py admin.py admin_set_student.py
125 def fix_action(self, detail):
126     try:
127         # 创建游标对象
128         cursor = db.cursor()
129
130         # 执行插入操作
131         cursor.execute(query="INSERT INTO fix (apply_student_id, detail) VALUES (%s, %s)"
132                        args=(self.id, detail))
133
134         # 提交事务
135         db.commit()
136         messagebox.showinfo(title="成功", message="维修记录提交成功! 请耐心等待审核结果, 可在本界
137
138     except Exception as e:
139         messagebox.showerror(title="错误", message="维修记录提交失败: " + str(e))
140
141 # 个人信息界面, 逻辑比较简单, 不再详细注释
```

触发器源
码
(3分)

```
DELIMITER //
CREATE TRIGGER check_fix_init
BEFORE INSERT ON fix
FOR EACH ROW
BEGIN
    DECLARE record_count INT;
    -- 获取当前学生在过去24小时内提交的记录数量
    SELECT COUNT(*) INTO record_count
    FROM fix
    WHERE apply_student_id = NEW.apply_student_id
    AND timestamp >= DATE_SUB(NOW(), INTERVAL 24 HOUR);
    -- 如果记录数量大于等于3, 则抛出错误
    IF record_count >= 3 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '您在过去24小时内提交记录数量超过频率限制, 请耐心等待近期提交的审核结果!';
    END IF;
END //
```

说明: 不违背触发器能够执行插入操作。

程序演示
(4分)



	
程序演示 (4分)	<p>说明：违背触发器要求，不能够执行插入操作，系统报错：</p> 
备注	

6. 存储过程控制下的更新操作（18 分）

说明	<p>（1 分）简要说明该操作所要完成的功能；</p> <p>（1 分）简要说明该存储过程所要完成的功能；</p> <p>（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）</p> <p>（1 分）表连接涉及字段描述（描述方式为“表 1. 属性=表 2. 属性”）</p> <p>（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；</p> <p>（6 分）实现该操作的关键代码（高级语言、SQL），截图即可；</p> <p>（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。</p>	
功能描述（1 分）	<p>对学生进行转宿舍操作，更新原新宿舍的人数信息，即原宿舍人数减一，新宿舍人数加 1，并更新 student 表中相应记录的 dormitory_id 属性的更新。执行更新操作前首先检查输入的宿舍号是否合法，然后检查新宿舍是否还有增加学生的空间，再检查性别是否一致，最后再检查输入的床位号是否合法，当这些条件都符合时才成功执行转宿舍操作。</p>	
存储过程功能描述（1 分）	<p>该操作主要完成的是更新 dorm 表 stu_count 属性以及 student 表中的 dormitory_id 属性，该存储过程接受新宿舍号、安排床位号为输入参数，来完成 dormitory_id 属性与原宿舍 stu_count 减 1 和新宿舍 stu_count 加 1 的更新。在更新之前，会检查宿舍号输入是否存在，再检查新旧宿舍是否相同，然后检查新宿舍 stu_count 属性是否等于 dormitory_count 属性，再检查转入的宿舍所在宿舍楼的性别是否与当前学生性别一致，再检查输入的整数床位号是否在 1 到新宿舍最大人数容量之间，最后再检查提交的床位号是否与原有床位号冲突，当这六个条件判断依次为 true、false、false、true、true、false 才成功执行转宿舍操作，否则抛出错误。</p>	
涉及的关系表（2 分）	Student /dorm /dormitory_building	
表连接涉及字段（1）	<p>Student.dormitory_id=dorm.dormitory_id</p> <p>dorm.building_id=dormitory_building.building_id</p>	
更改字段（2 分）	字段	规则
	Dorm.stu_count	新宿舍人数+1，原宿舍人数-1
	Student.dormitory_id	更新学生的宿舍号
	Student.stu_seat	更新学生的床位号

更新代码
(3分)


```
-- 更新原宿舍的人数
UPDATE dorm
SET stu_count = old_dorm_count - 1
WHERE dormitory_id = old_dorm_id;

-- 更新学生的宿舍信息
UPDATE student
SET dormitory_id = new_dorm_id, stu_seat = new_seat
WHERE student_id = stu_id;

-- 更新新宿舍的人数
UPDATE dorm
SET stu_count = new_dorm_count + 1
WHERE dormitory_id = new_dorm_id;
```

创建存储过程
源码
(3分)

```
1 DELIMITER //
2
3 CREATE PROCEDURE change_dorm_proc(
4     IN new_dorm_id VARCHAR(20),
5     IN new_seat INT,
6     IN stu_id VARCHAR(20),
7     OUT success BOOLEAN
8 )
9
10 BEGIN
11     #原宿舍号
12     DECLARE old_dorm_id INT;
13     #原、新宿舍人数
14     DECLARE old_dorm_count INT;
15     DECLARE new_dorm_count INT;
16     #新宿舍容量
17     DECLARE new_dorm_cap INT;
18     #学生性别
19     DECLARE stu_gender CHAR(2);
20     #新宿舍性别
21     DECLARE dorm_gender CHAR(2);
22     #用于检查宿舍号是否存在和判断床位是否冲突的变量
23     DECLARE dorm_count INT;
24     DECLARE is_seat_conflict INT;
25
26
27 -- 获取原宿舍信息
28 SELECT dormitory_id INTO old_dorm_id
29 FROM student
30 WHERE student_id = stu_id;
31
32 -- 获取原宿舍的当前人数
33 SELECT stu_count INTO old_dorm_count
34 FROM dorm
35 WHERE dormitory_id = old_dorm_id;
36
37 -- 获取新宿舍的当前人数和性别
38 SELECT stu_count, sex, Dormitory_resident
39 INTO new_dorm_count, dorm_gender, new_dorm_cap
40 FROM dorm natural join dormitory_building
41 WHERE dormitory_id = new_dorm_id;
42
43 -- 开始事务
44 START TRANSACTION;
45
46 -- 1.检查宿舍号输入是否存在
47 SELECT COUNT(*) INTO dorm_count FROM dorm WHERE dormitory_id = new_dorm_id;
48 IF dorm_count = 0 THEN
49     SET success = FALSE;
50     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '输入的宿舍不存在!';
51 END IF;
52
53 -- 2.检查新旧宿舍是否相同
54 IF old_dorm_id = new_dorm_id THEN
55     SET success = FALSE;
56     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '新旧宿舍不能相同!';
57     ROLLBACK;
58 END IF;
59
60 -- 3.检查新宿舍stu_count属性是否等于dormitory_count属性
61 IF new_dorm_cap = (SELECT stu_count FROM dorm WHERE dormitory_id = new_dorm_id) THEN
62     SET success = FALSE;
63     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '新宿舍容量已满!';
64 END IF;
65
66 -- 4.检查新宿舍是否与学生性别匹配
67 IF dorm_gender != (SELECT stu_sex FROM student WHERE student_id = stu_id) THEN
68     SET success = FALSE;
69     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '请安排到性别一致的宿舍!';
70     ROLLBACK;
71 END IF;
72
73 -- 5.检查床位号是否在新宿舍的容量范围内
74 IF new_seat < 1 OR new_seat > new_dorm_cap THEN
75     SET success = FALSE;
76     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '新床位不合法!';
77     ROLLBACK;
78 END IF;
```

	<pre>78 79 -- 6.检查提交的床位号是否与原有床位号冲突 80 SELECT COUNT(*) INTO is_seat_conflict 81 FROM student 82 WHERE dormitory_id = new_dorm_id AND stu_seat = new_seat; 83 84 IF is_seat_conflict > 0 THEN 85 SET success = FALSE; 86 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '输入床位与已安排床位冲突!'; 87 ROLLBACK; 88 END IF; 89 -- 更新原宿舍的人数 90 UPDATE dorm 91 SET stu_count = old_dorm_count - 1 92 WHERE dormitory_id = old_dorm_id; 93 -- 更新学生的宿舍信息 94 UPDATE student 95 SET dormitory_id = new_dorm_id, stu_seat = new_seat 96 WHERE student_id = stu_id; 97 -- 更新新宿舍的人数 98 UPDATE dorm 99 SET stu_count = new_dorm_count + 1 100 WHERE dormitory_id = new_dorm_id; 101 SET success = TRUE; 102 -- 提交事务 103 COMMIT; 104 END // 105 DELIMITER ;</pre>
存储过程 执行 源码 (1 分)	
程序演 示 (2 分)	 <p>说明：不违背存储过程，能够执行更新操作，点击确定后系统自动刷新界面，可以看到信息已经正确地更新：</p>

更新学生信息

基本信息修改

学院：

姓名：

性别：

更新信息

移出宿舍

更换床位或宿舍

床位号：3

宿舍号：3001

新床位号：

更换床位

新宿舍号：

新床位号：

更换宿舍

说明：违背存储过程：（条件 1），系统报错；

更新学生信息

基本信息修改

学院:

姓名:

床位号:

宿舍号:

移出宿舍:

更换床位或宿舍:

错误提示

执行存储过程时出错: (1644, 输入的宿舍不存在!)

确定


新宿舍号:

新床位号:

移出宿舍:

更换宿舍:

说明：违背存储过程：（条件 2），系统报错；



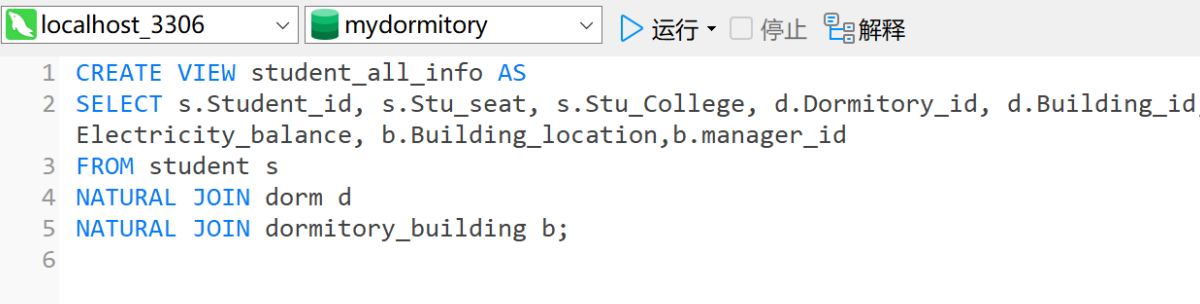
The screenshot shows the 'Basic Information Modification' page. The 'School' field is filled with 'Electronic Information and Optical Engineering College'. The 'Room Number' field is filled with '3002'. A red error dialog box is displayed in the center, with the message: '执行存储过程时出错: (1644, '新旧宿舍不能相同。')' (Error executing stored procedure: (1644, 'Old and new dormitories cannot be the same.')). The dialog box has a red 'X' icon and a '确定' (OK) button. The background page is partially obscured by the dialog box.

说明：违背存储过程：（条件3），系统报错；

程序演
示
(2
分)

	<div data-bbox="316 192 837 564"><p>更新学生信息</p><div><div>基本信息修改</div><div>更换床位或宿舍</div></div><div><div>学院: 历史学院</div><div>床位号: 2</div></div><div><div>姓名: 李明</div><div>宿舍号: 3001</div></div><div><div>性别: 男</div><div></div></div><div><div>移出宿舍</div><div>更换宿舍</div></div><div><div>新宿舍号: 3002</div><div>新床位号: 1</div></div><div><div>确定</div></div><div><div>错误提示</div><div>执行存储过程时出错: (1644, '新宿舍容量已满!')</div></div></div> <p>说明：违背存储过程：（条件 5），系统报错；</p> <div data-bbox="316 627 837 1001"><p>更新学生信息</p><div><div>基本信息修改</div><div>更换床位或宿舍</div></div><div><div>学院: 电子信息与光学工程学院</div><div>床位号: 1</div></div><div><div>姓名: 韦宇宇</div><div>宿舍号: 3002</div></div><div><div>移出宿舍</div><div>更换床位</div></div><div><div>新宿舍号: 3001</div><div>新床位号: 5</div></div><div><div>确定</div></div><div><div>错误提示</div><div>执行存储过程时出错: (1644, '新床位不合法!')</div></div></div> <p>说明：违背存储过程：（条件 6），系统报错；</p> <div data-bbox="316 1081 837 1458"><p>更新学生信息</p><div><div>基本信息修改</div><div>更换床位或宿舍</div></div><div><div>学院: 电子信息与光学工程学院</div><div>床位号: 1</div></div><div><div>姓名: 韦宇宇</div><div>宿舍号: 3002</div></div><div><div>移出宿舍</div><div>更换床位</div></div><div><div>新宿舍号: 3001</div><div>新床位号: 2</div></div><div><div>确定</div></div><div><div>错误提示</div><div>执行存储过程时出错: (1644, '输入床位与已安排床位冲突!')</div></div></div>
备注	该操作也有权限，跟上述删除宿舍管理人员权限逻辑相同。

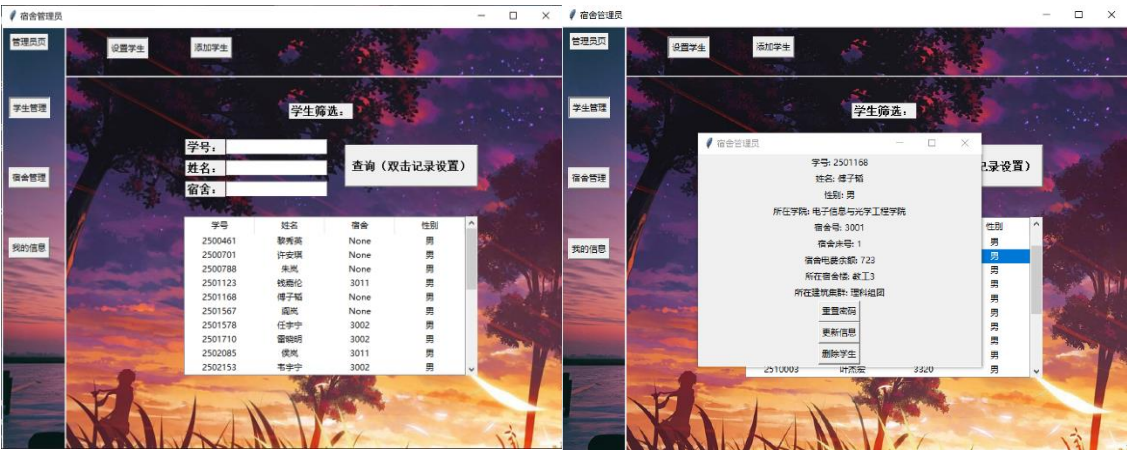
7. 含有视图的查询操作（15 分）

说明	<p>(1 分) 简要说明该操作所要完成的功能；</p> <p>(1 分) 简要说明建立的该视图的功能；</p> <p>(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）</p> <p>(1 分) 简要说明表连接涉及的字段（以“表 1. 属性=表 2. 属性”）</p> <p>(6 分) 实现该操作的关键代码（高级语言、SQL），截图即可；</p> <p>(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。</p>
操作功能描述 (1 分)	该操作视图显示每个学生的详细信息，包括学号、姓名、性别、所在学院、宿舍号、宿舍床号、宿舍电费余额、所在宿舍楼、所在建筑集群。这个操作通过三表连接并执行子查询来获得这些信息。
视图功能描述 (1 分)	如果学生已分配宿舍，该视图可用于方便地展示被查询学生的学号、姓名、性别、所在学院、宿舍号、宿舍床号、宿舍电费余额、所在宿舍楼、所在建筑集群。
涉及的关系表 (2 分)	Student/dorm/dormitory_building
表连接字段 (1 分)	<p>Student.Dormitory_id=dorm.Dormitory_id</p> <p>Dorm.Building_id=dormitory_building.Building_id</p>
创建视图代码 (3 分)	 <pre> 1 CREATE VIEW student_all_info AS 2 SELECT s.Student_id, s.Stu_seat, s.Stu_College, d.Dormitory_id, d.Building_id, Electricity_balance, b.Building_location, b.manager_id 3 FROM student s 4 NATURAL JOIN dorm d 5 NATURAL JOIN dormitory_building b; 6 </pre>

查询代码
(3分)

```
main.py admin.py admin_set_student.py
169 else:
170     cursor.execute(
171         query: 'SELECT Stu_seat, stu_college, building_id, Electricity_balance, Building_location
172         'FROM student_info_with_dorm '
173         'WHERE Student_id = %s', stu_id
174     )
175     student_info = cursor.fetchone()
176     stu_seat = student_info[0]
177     stu_college = student_info[1]
178     stu_building = student_info[2]
179     stu_balance = student_info[3]
180     stu_location = student_info[4]
181     # 显示选定的学生信息
182     tk.Label(new_window, text="学号: " + str(stu_id)).pack()
183     tk.Label(new_window, text="姓名: " + str(stu_name)).pack()
184     tk.Label(new_window, text="性别: " + str(stu_sex)).pack()
185     tk.Label(new_window, text="所在学院: " + str(stu_college)).pack()
186     tk.Label(new_window, text="宿舍号: " + str(stu_dorm)).pack()
187     tk.Label(new_window, text="宿舍床号: " + str(stu_seat)).pack()
188     tk.Label(new_window, text="宿舍电费余额: " + str(stu_balance)).pack()
189     tk.Label(new_window, text="所在宿舍楼: " + str(stu_building)).pack()
190     tk.Label(new_window, text="所在建筑集群: " + str(stu_location)).pack()
AdminSystem on_student_query_select() else
```

程序演示
(4分)



备注

只有当查看已分配宿舍的学生的详细信息时才会调用该视图，当学生处于未分配宿舍状态时，代码会直接调用数据库表 student 即可完成信息的查询，且母查询为模糊匹配。