

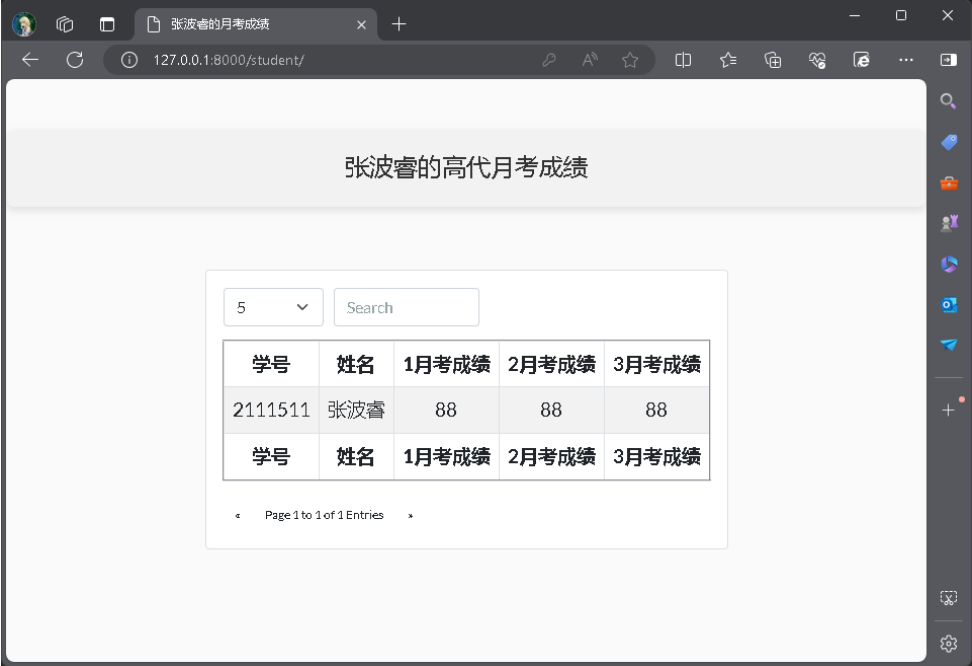
# 数据库工程作业

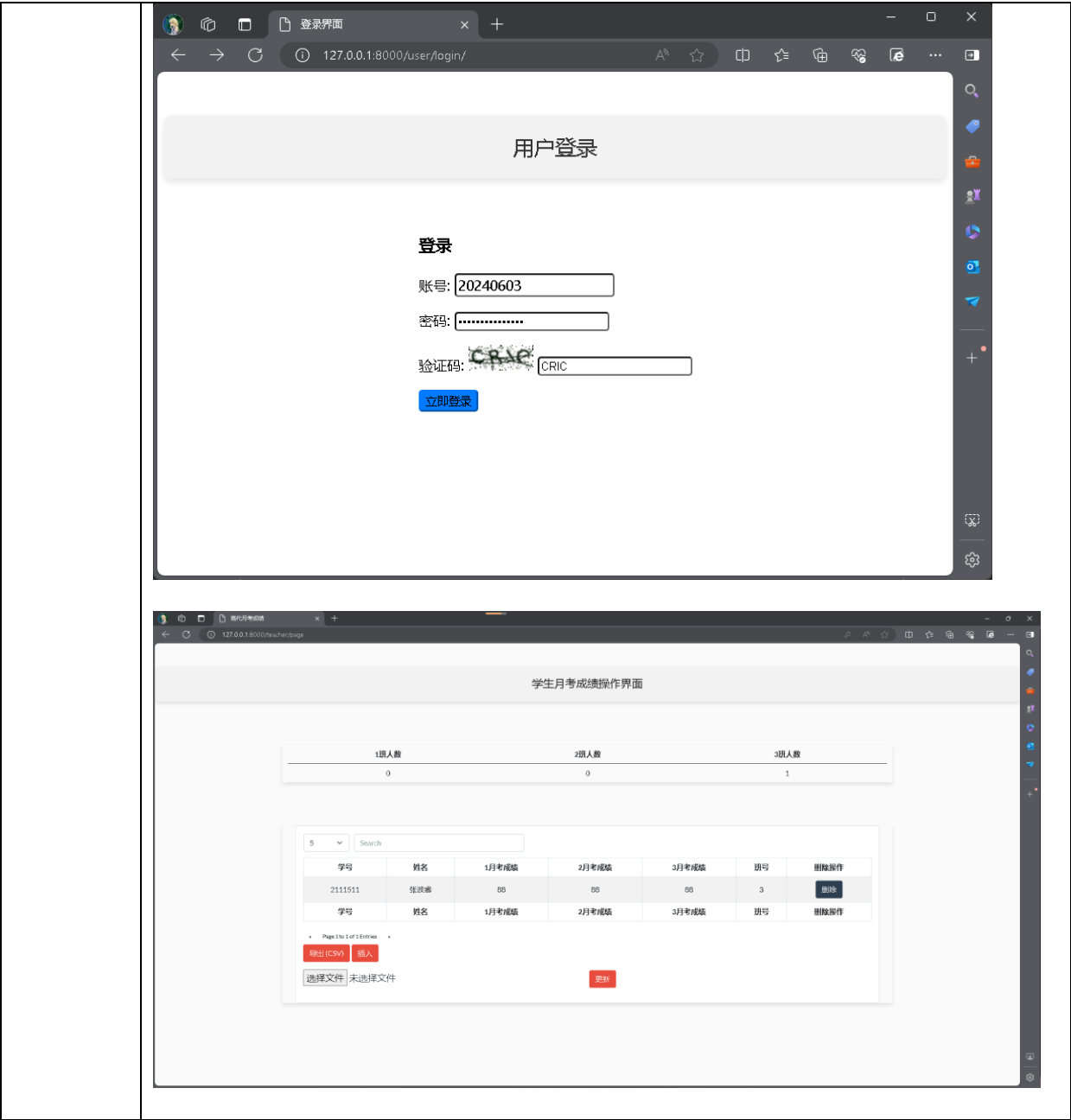
要求:

- 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	2111511	姓名	张波睿	专业	计算机科学（辅修）
项目名称	高等代数月考成绩管理系统				
必备环境	MySQL==8.0.36 python==3.8.0 django==4.2.13 mysqlclient==2.2.4				
系统主要功能简介（4 分）	登录页面可以通过账号、密码、验证码进行登录操作，老师和学生会跳转到不同页面。 学生页面可以查询学生自己的三次月考成绩。 老师页面可以查看所有学生的学号、姓名、三次月考成绩、班号，查看全部班级的人数；可以导出 csv 文件；可以插入整条新学生数据；可以删除整条学生数据；可以通过上传 excel 表导入更新学生成绩班级数据。				
系统主要页面截图（6 分）					



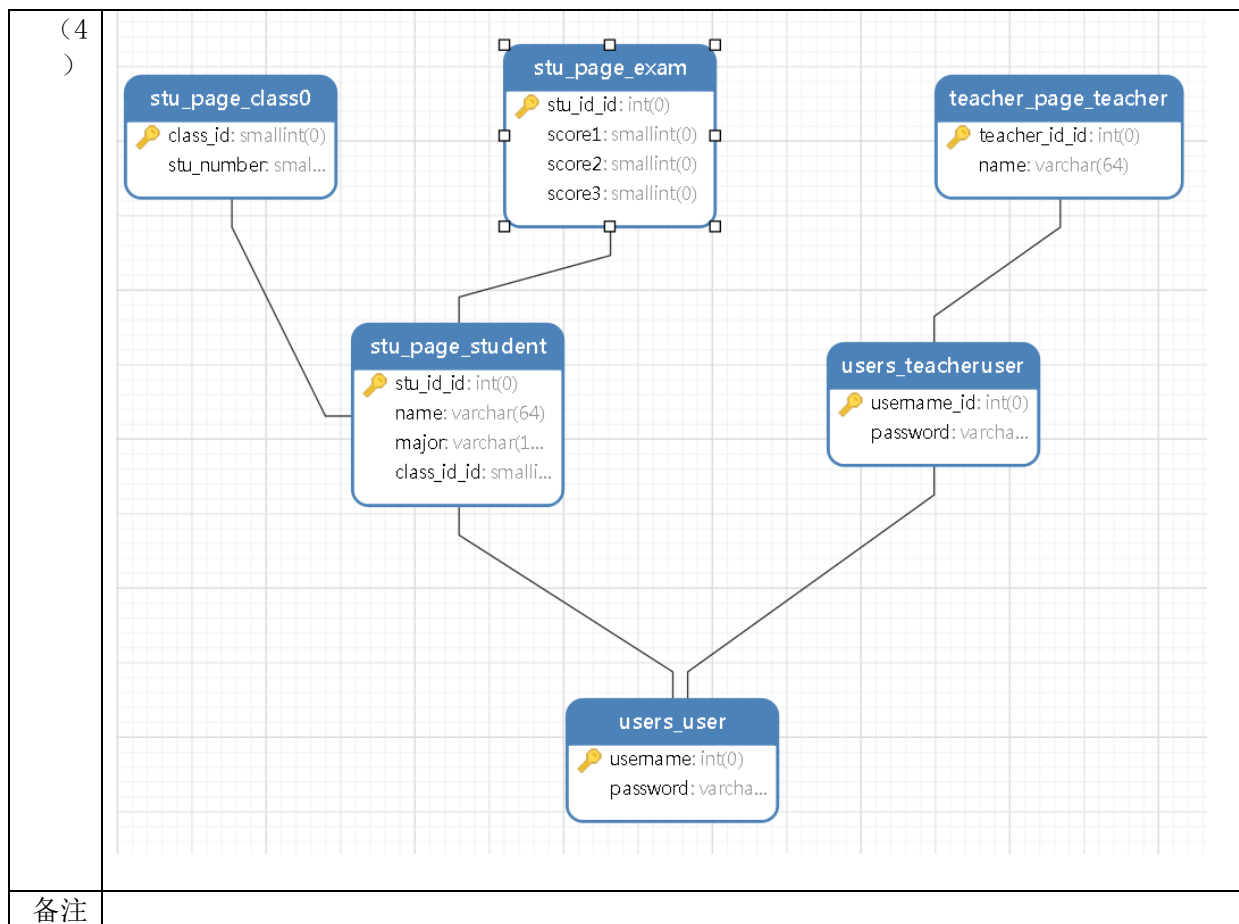
2. 系统配置（10 分）

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. 安装 MySQL			
		2. 建立一个 database 名为 dbs_homework			
	高级 语言	1. 用 conda 开辟一个虚拟空间（python==3.8.0）			
		2. 安装 django			
		3. 安装 mysqlclient			
		4. 安装 jupyter			
连接串 分析 (6 分)	序 号	名 称	功能说明	取 值	
	1	ENGINE	指定了 Django 将使用哪个数据库后端。其	django.db.backen	

			内容表示 Django 将使用其内置的 MySQL 数据库适配器来与数据库交互	ds.mysql
	2	NAME	指定了所用数据库名称	db homework
	3	USER	指定了连接数据库所使用的用户名	root
	4	PASSWORD	指定了连接数据库所使用的密码	woai_zhdd021025
	5	HOST	指定了数据库服务器的主机地址，这里利用本人电脑，数据库和 django 运行在同一地址，因此内容为右侧所示	127.0.0.1
	6	PORT	指定了数据库监听的端口号，默认为 3306	3306
连接串代码 (截屏) (2 分)	<pre> 80 81 DATABASES = { 82     "default": { 83         "ENGINE": "django.db.backends.mysql", 84         "NAME": "db homework", 85         "USER": "root", 86         "PASSWORD": "woai_zhdd021025", 87         "HOST": "127.0.0.1", 88         "PORT": "3306" 89     } 90 }           </pre>			
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	users_user(username, password)	username		
	2	users_teacheruser(username_id, password)	username_id	username_id	users_user(username)
	3	teacher_page_teacher(teacher_id_id, name)	teacher_id_id	teacher_id_id	users_teacheruser(username_id)
	4	stu_page_class0(class_id, stu_number)	class_id		
	5	stu_page_student(stu_id_id, name, major, class_id_id)	stu_id_id	stu_id_id class_id_id	users_user(username) stu_page_class0(class_id)
关系图	6	stu_page_exam(stu_id_id, score1, score2, score3)	stu_id_id	stu_id_id	stu_page_student(stu_id_id)



#### 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）	老师网页中对应学生块中点击删除按钮，则删除该学生的成绩和对应信息，但不删除账号。	
涉及的表（2 分）	stu_page_student stu_page_exam	
表连接涉及字段（1 分）	stu_page_student.stu_id_id = stu_page_exam.stu_id_id	
删除条件字段描述（1 分）	字段	规则
	stu_page_student.stu_id_id=?	当接收到学号数据时，删除在两个表中的这个学号的学生全部数据

代码 (4 分)	<pre>def delete_student(stu_id):     try:         #开启事务管理         with transaction.atomic():             student = Student.objects.get(stu_id_id=stu_id) #如果没有该学生，删除操作停止              if Exam.objects.filter(stu_id_id=stu_id).exists(): #是否存在成绩数据                 Exam.objects.get(stu_id_id=stu_id).delete() #删除成绩数据              student.delete() #删除学生数据      except Student.DoesNotExist:         print('删除学生不存在！')         logger.error(f"尝试删除不存在学生学号：{stu_id}")      except Exception as e:         logger.error(f"在删除学号 {stu_id}学生时有错误{e}发生")</pre>																																																	
程序演示 (4 分)	<table><tr><th>学号</th><th>姓名</th><th>1月考成绩</th><th>2月考成绩</th><th>3月考成绩</th><th>班号</th><th>删除操作</th></tr><tr><td>112</td><td>2</td><td>None</td><td>None</td><td>None</td><td>1</td><td><div>删除</div></td></tr><tr><td>2111511</td><td>张波豪</td><td>88</td><td>88</td><td>88</td><td>3</td><td><div>删除</div></td></tr><tr><th>学号</th><th>姓名</th><th>1月考成绩</th><th>2月考成绩</th><th>3月考成绩</th><th>班号</th><th>删除操作</th></tr></table> <p>点击删除后</p> <table><tr><th>学号</th><th>姓名</th><th>1月考成绩</th><th>2月考成绩</th><th>3月考成绩</th><th>班号</th><th>删除操作</th></tr><tr><td>2111511</td><td>张波豪</td><td>88</td><td>88</td><td>88</td><td>3</td><td><div>删除</div></td></tr><tr><th>学号</th><th>姓名</th><th>1月考成绩</th><th>2月考成绩</th><th>3月考成绩</th><th>班号</th><th>删除操作</th></tr></table>	学号	姓名	1月考成绩	2月考成绩	3月考成绩	班号	删除操作	112	2	None	None	None	1	<div>删除</div>	2111511	张波豪	88	88	88	3	<div>删除</div>	学号	姓名	1月考成绩	2月考成绩	3月考成绩	班号	删除操作	学号	姓名	1月考成绩	2月考成绩	3月考成绩	班号	删除操作	2111511	张波豪	88	88	88	3	<div>删除</div>	学号	姓名	1月考成绩	2月考成绩	3月考成绩	班号	删除操作
学号	姓名	1月考成绩	2月考成绩	3月考成绩	班号	删除操作																																												
112	2	None	None	None	1	<div>删除</div>																																												
2111511	张波豪	88	88	88	3	<div>删除</div>																																												
学号	姓名	1月考成绩	2月考成绩	3月考成绩	班号	删除操作																																												
学号	姓名	1月考成绩	2月考成绩	3月考成绩	班号	删除操作																																												
2111511	张波豪	88	88	88	3	<div>删除</div>																																												
学号	姓名	1月考成绩	2月考成绩	3月考成绩	班号	删除操作																																												
备注																																																		

## 5. 触发器控制下的添加操作（20 分）

说明	(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。	
功能描述 (1 分)	在进行学生数据插入、删除、修改操作时, 同时改变 stu_page_class0 中的学生人数。 在进行学生数据插入时, 检查插入成绩是否为 0-100, 若超出范围则停止操作。	
触发器描述 (2 分)	在进行学生数据插入、删除、修改操作时, 由触发器控制 stu_page_class0 中学生人数的增加减少修改。 在进行学生数据插入时, 由触发器控制检查插入成绩是否为 0-100, 若超出范围则由触发器停止操作。 下面内容只讲述插入操作。	
涉及的表 (1 分)	stu_page_student, stu_page_class0	
输入数据 (2 分)	字段	规则
	users_user.username=?	INT
	stu_page_student.name=?	VARCHAR(64)
	stu_page_student.class_id_id=?	INT, 已经在 stu_page_class.class_id 中出现
	stu_page_exam.score1=?	INT, >0, <100
	stu_page_exam.score2=?	INT, >0, <100
	stu_page_exam.score3=?	INT, >0, <100
插入操作源码 (3 分)		

	<pre> def insert(request):     # 获取POST数据     data = request.POST     stu_id = data.get('stu_id')     name = data.get('name')     score1 = data.get('score1')     score2 = data.get('score2')     score3 = data.get('score3')     class_id = data.get('class_id')      # 将'NaN'字符串转换为None, 否则转换为int     score1 = None if score1 == 'NaN' else int(score1) if score1 else None     score2 = None if score2 == 'NaN' else int(score2) if score2 else None     score3 = None if score3 == 'NaN' else int(score3) if score3 else None      # 获取班级信息, 如果不存在则返回错误响应     try:         class0 = Class0.objects.get(class_id=class_id)     except Class0.DoesNotExist:         return HttpResponse('不存在这样的班级', status=400)      # 尝试获取或创建用户     try:         user0, created = User.objects.get_or_create(             username=stu_id,             defaults={'password': '123456'}         )     except Exception as e:         return HttpResponse(f'用户创建/获取失败: {str(e)}', status=500)      # 确保学生信息在创建考试成绩前存在     try:         student0, student_created = Student.objects.get_or_create(             stu_id=user0,             defaults={'name': name, 'class_id': class0}         )     except Exception as e:         return HttpResponse(f'学生信息创建/获取失败: {str(e)}', status=500)      # 创建考试成绩     try:         Exam.objects.create(             stu_id=student0,             score1=score1,             score2=score2,             score3=score3         )     except Exception as e:         return HttpResponse(f'考试成绩创建失败: {str(e)}', status=500)      return HttpResponse('数据已成功插入!', status=201) </pre>
触发器 源码 (3 分)	

```
mysql> DELIMITER $$
mysql> CREATE TRIGGER trg_stu_update
-> AFTER UPDATE ON stu_page_student
-> FOR EACH ROW
-> BEGIN
->     UPDATE stu_page_class0 SET stu_number = stu_number + 1
->     WHERE class_id = NEW.class_id_id;
->     UPDATE stu_page_class0 SET stu_number = stu_number - 1
->     WHERE class_id = OLD.class_id_id;END$$
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
mysql> DELIMITER $$
mysql> CREATE TRIGGER trg_stu_insert
-> AFTER INSERT ON stu_page_student
-> FOR EACH ROW
-> BEGIN
->     UPDATE stu_page_class0
->     SET stu_number = stu_number + 1
->     WHERE class_id = NEW.class_id_id;
-> END$$
Query OK, 0 rows affected (0.02 sec)

mysql> DELIMITER ;
mysql>
mysql> DELIMITER $$
mysql> CREATE TRIGGER trg_stu_delete
-> AFTER DELETE ON stu_page_student
-> FOR EACH ROW
-> BEGIN
->     UPDATE stu_page_class0
->     SET stu_number = stu_number - 1
->     WHERE class_id = OLD.class_id_id;
-> END$$
Query OK, 0 rows affected (0.02 sec)

mysql> DELIMITER ;
mysql>
```

另一个源码忘记拍照了，只会找出这个

```
***** 1 row *****
      TRIGGER_CATALOG: def
      TRIGGER_SCHEMA: dbs_homework
      TRIGGER_NAME: check_score
      EVENT_MANIPULATION: INSERT
      EVENT_OBJECT_CATALOG: def
      EVENT_OBJECT_SCHEMA: dbs_homework
      EVENT_OBJECT_TABLE: stu_page_exam
      ACTION_ORDER: 1
      ACTION_CONDITION: NULL
      ACTION_STATEMENT: BEGIN
      IF NEW.score1 IS NOT NULL AND (NEW.score1 < 0 OR NEW.score1 > 100) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '成绩必须在0-100!';
      END IF;

      IF NEW.score2 IS NOT NULL AND (NEW.score2 < 0 OR NEW.score2 > 100) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '成绩必须在0-100!';
      END IF;

      IF NEW.score3 IS NOT NULL AND (NEW.score3 < 0 OR NEW.score3 > 100) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '成绩必须在0-100!';
      END IF;
    END
      ACTION_ORIENTATION: ROW
      ACTION_TIMING: BEFORE
      ACTION_REFERENCE_OLD_TABLE: NULL
      ACTION_REFERENCE_NEW_TABLE: NULL
      ACTION_REFERENCE_OLD_ROW: OLD
      ACTION_REFERENCE_NEW_ROW: NEW
      CREATED: 2024-06-03 19:50:59.64
      SQL_MODE: ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION
      DEFINER: root@localhost
      CHARACTER_SET_CLIENT: gbk
      COLLATION_CONNECTION: gbk_chinese_ci
      DATABASE_COLLATION: utf8mb3_general_ci
1 row in set (0.01 sec)
```

```
2024-06-03 02:42:29.45 | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION | root@localhost | gbk | NEW | gbk_chinese_ci |
| utf8mb3_general_ci |
| def | dbs_homework | check_score | INSERT | def | dbs_homework | stu_page_exam | 1 | NULL | BEGIN
      IF NEW.score1 IS NOT NULL AND (NEW.score1 < 0 OR NEW.score1 > 100) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '成绩必须在0-100!';
      END IF;

      IF NEW.score2 IS NOT NULL AND (NEW.score2 < 0 OR NEW.score2 > 100) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '成绩必须在0-100!';
      END IF;

      IF NEW.score3 IS NOT NULL AND (NEW.score3 < 0 OR NEW.score3 > 100) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '成绩必须在0-100!';
      END IF;
    END | ROW | BEFORE | NULL | OLD
| NEW | gbk_chinese_ci | utf8mb3_general_ci | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION | root@localhost | gbk
```

程序演示





## 6. 存储过程控制下的更新操作（18 分）

说明	（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。	
功能描述 （1 分）	导入一个符合规范的 excel 表，然后按照表中内容对学生姓名班级成绩信息进行更新操作。	
存储过程 功能描述 （1 分）	存储过程用于对某个学生更新基本信息和考试成绩。首先验证学生 ID 存在性和成绩的有效范围，在事务保护下更新学生姓名和班级信息，同时根据学生是否已有成绩记录决定新增或者更新考试成绩，确保数据一致性和完整性。	
涉及的关系表 （2 分）	stu_page_student, stu_page_exam	
表连接涉及 字段 （1）	stu_page_student.stu_id_id = stu_page_exam.stu_id_id	
更改字段 （2 分）	字段	规则
	stu_page_student.name	VARCHAR(64)
	stu_page_student.class_id_id	INT, 已经在 stu_page_class.class_id 中出现
	stu_page_exam.score1	INT, >0, <100, 如果没有成绩表就改成插入操作
	stu_page_exam.score2	INT, >0, <100, 如果没有成绩表就改成插入操作
	stu_page_exam.score3	INT, >0, <100, 如果没有成绩表就改成插入操作
更新代码 （3 分）	<pre> def upload_result(request):     if request.method == 'POST' and len(request.FILES) &gt; 0:         f = request.FILES['file']         excel_type = f.name.split('.')[1]         if excel_type in ['xlsx', 'xls']:             data = pd.read_excel(f)             for i in range(len(data)):                 stu_id = data.iloc[i,0]                 name = data.iloc[i,1]                 score1 = data.iloc[i,2]                 score2 = data.iloc[i,3]                 score3 = data.iloc[i,4]                 class_id = data.iloc[i,5]                 information = [stu_id, name, score1, score2, score3, class_id]                  try:                     call_stored_procedure(information) #调用存储过程                 except:                     print(f"数据{information}, 可能不存在该学生或更新成绩超出实际范围!")             else:                 error = '上传文件类型错误!'                 print(error)          return render(request, 'teacher_page/index2.html') </pre>	
创建存储		

过程源码  
(3分)

SQL 预览

```
1 CREATE DEFINER='root'@'localhost' PROCEDURE `UpdateStudent`(  
2     IN p_stu_id INT,  
3     IN p_name VARCHAR(64),  
4     IN p_score1 INT,  
5     IN p_score2 INT,  
6     IN p_score3 INT,  
7     IN p_class_id INT)  
8 BEGIN  
9     DECLARE student_exists INT DEFAULT 0;  
10  
11     START TRANSACTION;  
12  
13     SELECT COUNT(*) INTO student_exists  
14     FROM stu_page_student  
15     WHERE stu_id_id = p_stu_id;  
16  
17     IF student_exists = 0  
18         OR p_score1 IS NOT NULL AND (p_score1>100 OR p_score1<0)  
19         OR p_score2 IS NOT NULL AND (p_score2>100 OR p_score2<0)  
20         OR p_score3 IS NOT NULL AND (p_score3>100 OR p_score3<0)  
21     THEN  
22         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '学生ID不存在, 无法更新。';  
23         ROLLBACK;  
24     END IF;  
25  
26     UPDATE stu_page_student  
27     SET name = p_name,  
28         class_id_id = p_class_id  
29     WHERE stu_id_id = p_stu_id;  
30  
31     IF NOT EXISTS (  
32         SELECT 1 FROM stu_page_exam  
33         WHERE stu_id_id = p_stu_id  
34     ) THEN  
35         INSERT INTO stu_page_exam (stu_id_id, score1, score2, score3)  
36         VALUES (p_stu_id, p_score1, p_score2, p_score3);  
37     ELSE  
38         UPDATE stu_page_exam  
39         SET score1 = p_score1,  
40             score2 = p_score2,  
41             score3 = p_score3  
42         WHERE stu_id_id = p_stu_id;  
43     END IF;  
44     COMMIT;  
45 END
```

存储过程  
执行源码  
(1分)

```
from django.db import connection  
  
def call_stored_procedure(information):  
    with connection.cursor() as cursor:  
        cursor.callproc('UpdateStudent', information)
```

程序演示  
(2分)

导入 excel 表

学号	姓名	1月考试成绩	2月考试成绩	3月考试成绩	班号	删除操作
111111		None	None	None	1	<a href="#">删除</a>
333333		None	None	None	1	<a href="#">删除</a>
444444		None	None	None	1	<a href="#">删除</a>
555555		None	None	None	1	<a href="#">删除</a>
888888		None	None	None	1	<a href="#">删除</a>
学号	姓名	1月考试成绩	2月考试成绩	3月考试成绩	班号	删除操作

Page 1 to 2 of 10 Entries

[导出\(CSV\)](#) [插入](#)

[选择文件](#) test.xlsx

[更新](#)

更新后

	<table><tr><th>学号</th><th>姓名</th><th>1月考成绩</th><th>2月考成绩</th><th>3月考成绩</th><th>班号</th><th>删除操作</th></tr><tr><td>111111</td><td>测试</td><td>100</td><td>20</td><td>33</td><td>1</td><td><button>删除</button></td></tr><tr><td>333333</td><td></td><td>None</td><td>None</td><td>None</td><td>1</td><td><button>删除</button></td></tr><tr><td>555555</td><td>测试5</td><td>60</td><td>60</td><td>33</td><td>1</td><td><button>删除</button></td></tr><tr><td>888888</td><td>测试8</td><td>30</td><td>90</td><td>33</td><td>1</td><td><button>删除</button></td></tr><tr><td>222222</td><td>测试2</td><td>90</td><td>30</td><td>33</td><td>2</td><td><button>删除</button></td></tr><tr><th>学号</th><th>姓名</th><th>1月考成绩</th><th>2月考成绩</th><th>3月考成绩</th><th>班号</th><th>删除操作</th></tr></table>	学号	姓名	1月考成绩	2月考成绩	3月考成绩	班号	删除操作	111111	测试	100	20	33	1	<button>删除</button>	333333		None	None	None	1	<button>删除</button>	555555	测试5	60	60	33	1	<button>删除</button>	888888	测试8	30	90	33	1	<button>删除</button>	222222	测试2	90	30	33	2	<button>删除</button>	学号	姓名	1月考成绩	2月考成绩	3月考成绩	班号	删除操作
学号	姓名	1月考成绩	2月考成绩	3月考成绩	班号	删除操作																																												
111111	测试	100	20	33	1	<button>删除</button>																																												
333333		None	None	None	1	<button>删除</button>																																												
555555	测试5	60	60	33	1	<button>删除</button>																																												
888888	测试8	30	90	33	1	<button>删除</button>																																												
222222	测试2	90	30	33	2	<button>删除</button>																																												
学号	姓名	1月考成绩	2月考成绩	3月考成绩	班号	删除操作																																												
程序演示 (2分)	<div>[03/Jun/2024 23:13:07] "POST /teacher/insert HTTP/1.1" 200 9875 [03/Jun/2024 23:13:07] "GET /teacher/page HTTP/1.1" 200 15666 数据[333333, '测试3', 80, ' ', 33, 3], 可能不存在该学生或更新成绩超出实际范围! 数据[123456, '错误学号测试', 11, 11, 11, 0], 可能不存在该学生或更新成绩超出实际范围! 数据[666666, '测试6', 10, 70, 133, 2], 可能不存在该学生或更新成绩超出实际范围! 数据[999999, '测试9', 120, 100, 33, 2], 可能不存在该学生或更新成绩超出实际范围! [03/Jun/2024 23:14:02] "POST /teacher/upload HTTP/1.1" 200 9875 [03/Jun/2024 23:14:02] "GET /teacher/page HTTP/1.1" 200 15666</div>																																																	
备注																																																		

## 7. 含有视图的查询操作（15 分）

说明	(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。
操作功能描述 (1 分)	查询全部学生的学号、姓名、成绩, 并展示某个学号的那条信息。 查询全部学生的学号、姓名、成绩、班号, 并按照班号排序, 并展示全部信息。
视图功能描述 (1 分)	学生查询视图: 连接学生和成绩两个表后展示学号、姓名、成绩 老师查询视图: 连接学生和成绩两个表后展示学号、姓名、成绩、班号, 并按班号排列
涉及的关系表 (2 分)	stu_page_student, stu_page_exam
表连接字段 (1 分)	stu_page_student.stu_id_id = stu_page_exam.stu_id_id
创建视图代码 (3 分)	<pre> 1  SELECT 2    's'.stu_id_id AS stu_id_id, 3    's'.name AS name, 4    'e'.score1 AS score1, 5    'e'.score2 AS score2, 6    'e'.score3 AS score3 7  FROM 8    ( `stu_page_student` `s` LEFT JOIN `stu_page_exam` `e` ON ( ( `s`.stu_id_id = `e`.stu_id_id ) ) ) 9  ORDER BY 10   `s`.stu_id_id </pre> <pre> SELECT `s`.stu_id_id AS `stu_id_id`, `s`.name AS `name`, `e`.score1 AS `score1`, `e`.score2 AS `score2`, `e`.score3 AS `score3`, `s`.class_id_id AS `class_id_id` FROM ( `stu_page_student` `s` LEFT JOIN `stu_page_exam` `e` ON ( ( `s`.stu_id_id = `e`.stu_id_id ) ) ) ORDER BY `s`.class_id_id, `s`.stu_id_id </pre>
查询代码 (3 分)	<pre> class ShowScore(models.Model):     stu_id_id = models.IntegerField(primary_key=True)     name = models.CharField(max_length=64)     score1 = models.SmallIntegerField(null=True)     score2 = models.SmallIntegerField(null=True)     score3 = models.SmallIntegerField(null=True)      class Meta:         db_table = 'showscore2'         managed = False </pre>

	<pre>class ShowScoreWithClass(models.Model):     stu_id_id = models.IntegerField(primary_key=True)     name = models.CharField(max_length=64)     score1 = models.SmallIntegerField(null=True)     score2 = models.SmallIntegerField(null=True)     score3 = models.SmallIntegerField(null=True)     class_id_id = models.SmallIntegerField()      class Meta:         db_table = 'showscorewithclass2'         managed = False</pre>																																						
程序演示 (4分)	<div><div><p>张波睿的高代月考试成绩</p><div><div>5</div><div>Search</div></div><table><tr><th>学号</th><th>姓名</th><th>1月考试成绩</th><th>2月考试成绩</th><th>3月考试成绩</th></tr><tr><td>2111511</td><td>张波睿</td><td>88</td><td>88</td><td>88</td></tr><tr><th>学号</th><th>姓名</th><th>1月考试成绩</th><th>2月考试成绩</th><th>3月考试成绩</th></tr></table><div>Page 1 to 1 of 1 Entries</div></div></div> <div><div><p>学生月考试成绩操作界面</p><div><div>1班人数0</div><div>2班人数0</div><div>3班人数1</div></div><div><div><div>5</div><div>Search</div></div><table><tr><th>学号</th><th>姓名</th><th>1月考试成绩</th><th>2月考试成绩</th><th>3月考试成绩</th><th>班号</th><th>删除操作</th></tr><tr><td>2111511</td><td>张波睿</td><td>88</td><td>88</td><td>88</td><td>3</td><td>删除</td></tr><tr><th>学号</th><th>姓名</th><th>1月考试成绩</th><th>2月考试成绩</th><th>3月考试成绩</th><th>班号</th><th>删除操作</th></tr></table><div>Page 1 to 1 of 1 Entries</div><div><div>删除CSV</div><div>导入</div><div>选择文件</div><div>未选择文件</div><div>添加</div></div></div></div></div> <tr><td>备注</td><td></td></tr>	学号	姓名	1月考试成绩	2月考试成绩	3月考试成绩	2111511	张波睿	88	88	88	学号	姓名	1月考试成绩	2月考试成绩	3月考试成绩	学号	姓名	1月考试成绩	2月考试成绩	3月考试成绩	班号	删除操作	2111511	张波睿	88	88	88	3	删除	学号	姓名	1月考试成绩	2月考试成绩	3月考试成绩	班号	删除操作	备注	
学号	姓名	1月考试成绩	2月考试成绩	3月考试成绩																																			
2111511	张波睿	88	88	88																																			
学号	姓名	1月考试成绩	2月考试成绩	3月考试成绩																																			
学号	姓名	1月考试成绩	2月考试成绩	3月考试成绩	班号	删除操作																																	
2111511	张波睿	88	88	88	3	删除																																	
学号	姓名	1月考试成绩	2月考试成绩	3月考试成绩	班号	删除操作																																	
备注																																							