数据库工程作业

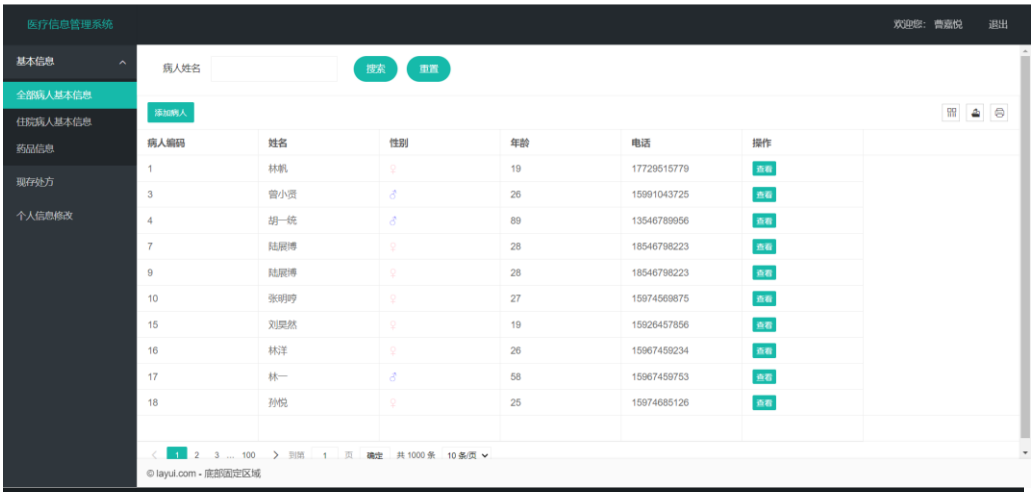
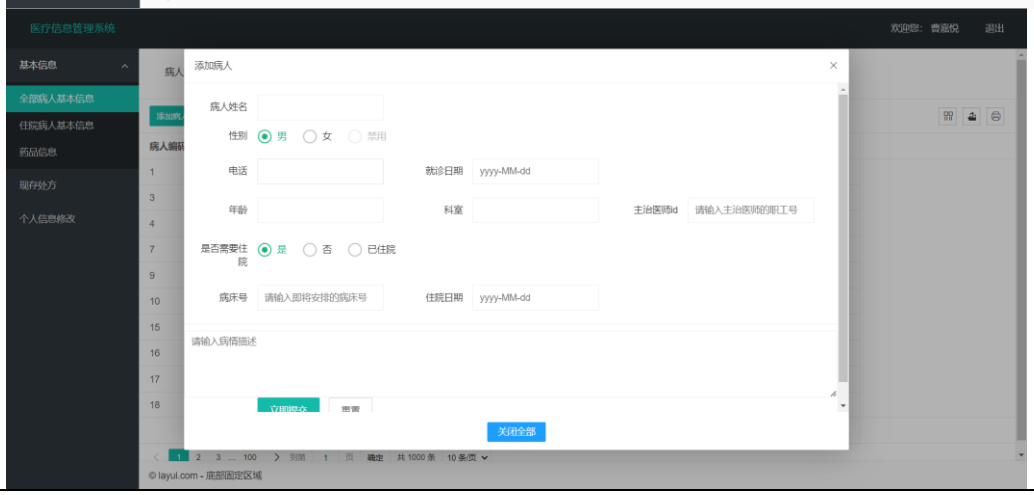
要求：

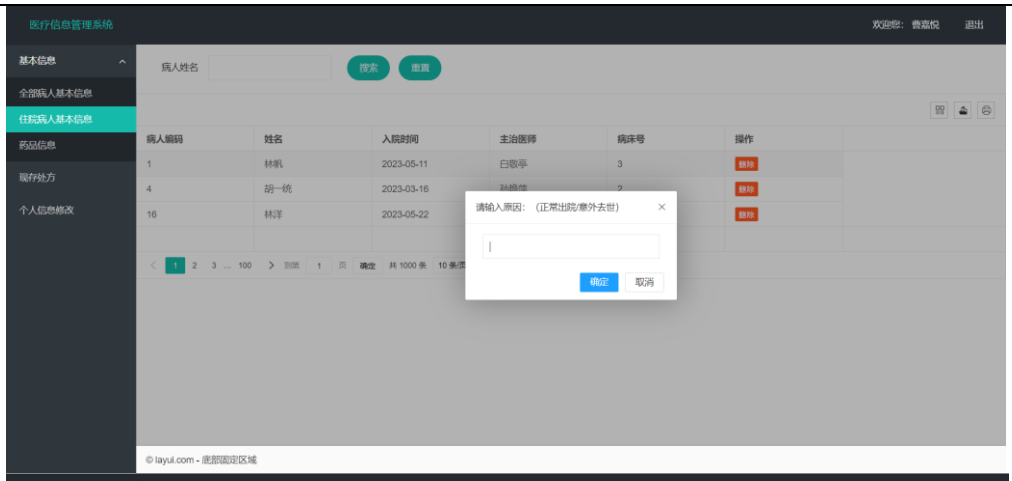
1. 完成一个小型的数据库信息管理系统（或部分功能），并填写工程作业报告；程序和报告请在规定时间之内上传。
2. 开发模式（B/S 或 C/S）、开发高级语言任选，后台数据库使用大型数据库管理系统（SQL Server、Oracle、MySQL 等），不要使用桌面数据库。
3. 报告中所列举的四种操作，每种操作举一个例子即可。
4. 作业成绩按照报告中的标准评分，程序只实现报告中涉及的部分即可。
5. 作业完成后，请将工程作业报告和程序打包提交给助教老师，并联系助教老师进行系统说明和演示，回答相关问题。

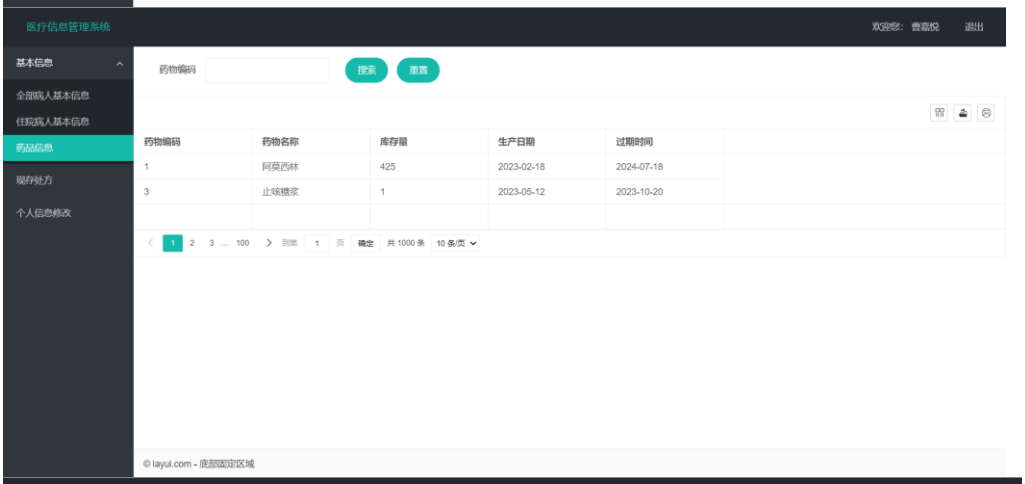
工程作业报告

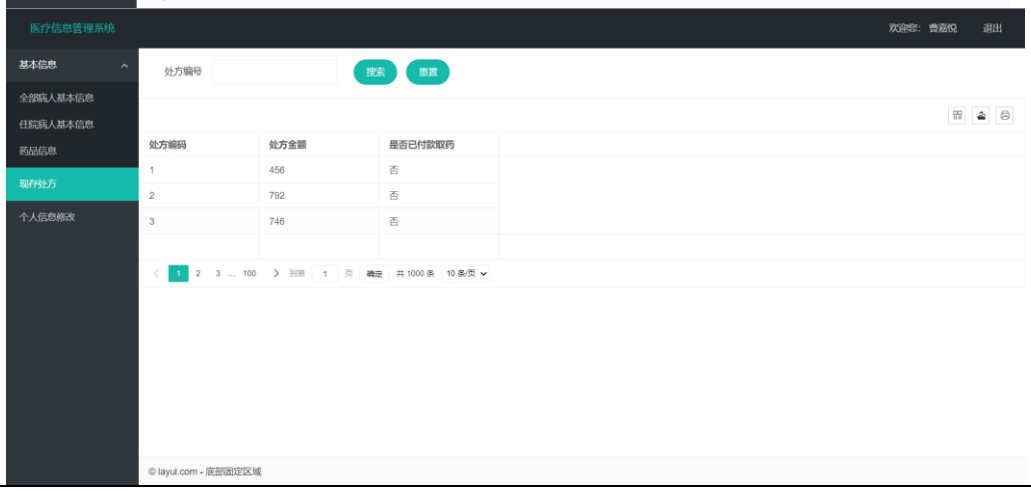
1. 项目信息（10 分）

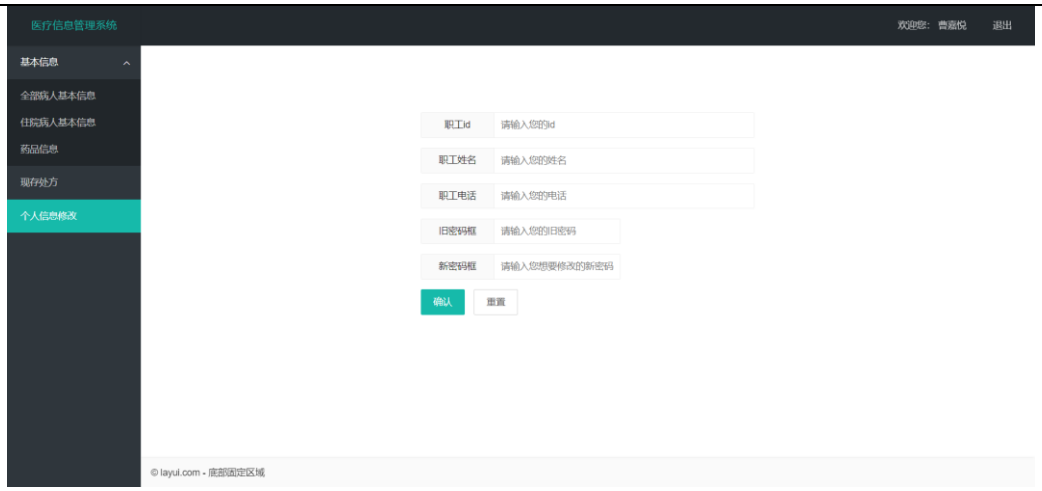
学号	2113881	姓名	曹嘉悦	专业	计算机科学与技术
项目名称	简易医院信息管理系统				
必备环境	MySQL、Python3.7				
系统主要功能简介（4 分）	1、医院职工根据自己的职工号和密码登入医疗信息管理系统界面； 2、进入界面后，在全部病人基本信息界面，我们可以通过输入姓名查找病人信息，也可以导出打印信息表；点击查看操作，我们可以查看病人的过往就诊记录，包括就诊时间、主管医生、挂号部门、病情描述；点击添加病人操作，我们输入新病人的相关信息会同时在病历表、病人信息表（以前没有就诊过）中加入一行的信息，且如果新病人选择住院，我们同时会在住院病人表中添加新的信息。 3、进入界面后，在住院病人基本信息界面，我们仍可以通过输入姓名查找病人信息，也可以导出打印信息表；点击删除界面，系统会让我们输入删除的原因，如果是正常出院，我们只需要把他从住院病人表中删除，在病人信息表中仍然保留信息。但如果是在手术未做之前意外死亡，我们需要在所有相关的表中删除这位病人的数据信息。 4、进入界面后，在现存处方表界面，我们可以通过处方编号搜索对应的处方；搜索到想要的处方之后，我们需要更新处方的取药情况，系统要求这里只能输入是，不能输入其他多余或错误的东西。在输入“是”之后，会更新药品库存。如果库存充足，系统会显示更新库存成功，反之，会显示库存不足！ 5、药品信息除了会因为处方信息更新，还会随着时间的推移，如果过期时间已到或库存为 0，会删除这段记录。 6、进入界面后，在个人信息修改界面,我们可以修改自己的电话和密码，其中修改密码的前提是输入旧密码正确。 7、点击“退出”，会返回登录界面。				

系统主要页面截图（6 分）	  


医疗信息管理系统
欢迎您：曹嘉悦
退出
基本信息
全部病人基本信息
住院病人基本信息
药品信息
现存处方
个人信息修改
病人姓名
搜索
重置
病人编码
姓名
入院时间
主治医师
病床号
操作
1
林帆
2023-05-11
白敬亭
3
删除
4
胡一统
2023-03-16
孙悟空
2
删除
16
林洋
2023-05-22
删除
请输入原因：（正常出院/意外去世）
确定
取消
© layui.com - 底部固定区域


医疗信息管理系统
欢迎您：曹嘉悦
退出
基本信息
全部病人基本信息
住院病人基本信息
药品信息
现存处方
个人信息修改
药物编码
搜索
重置
药物编码
药物名称
库存量
生产日期
过期时间
1
阿莫西林
425
2023-02-18
2024-07-18
3
止咳糖浆
1
2023-05-12
2023-10-20
© layui.com - 底部固定区域


医疗信息管理系统
欢迎您：曹嘉悦
退出
基本信息
全部病人基本信息
住院病人基本信息
药品信息
现存处方
个人信息修改
处方编号
搜索
重置
处方编码
处方金额
是否已付款取药
1
456
否
2
792
否
3
746
否
© layui.com - 底部固定区域

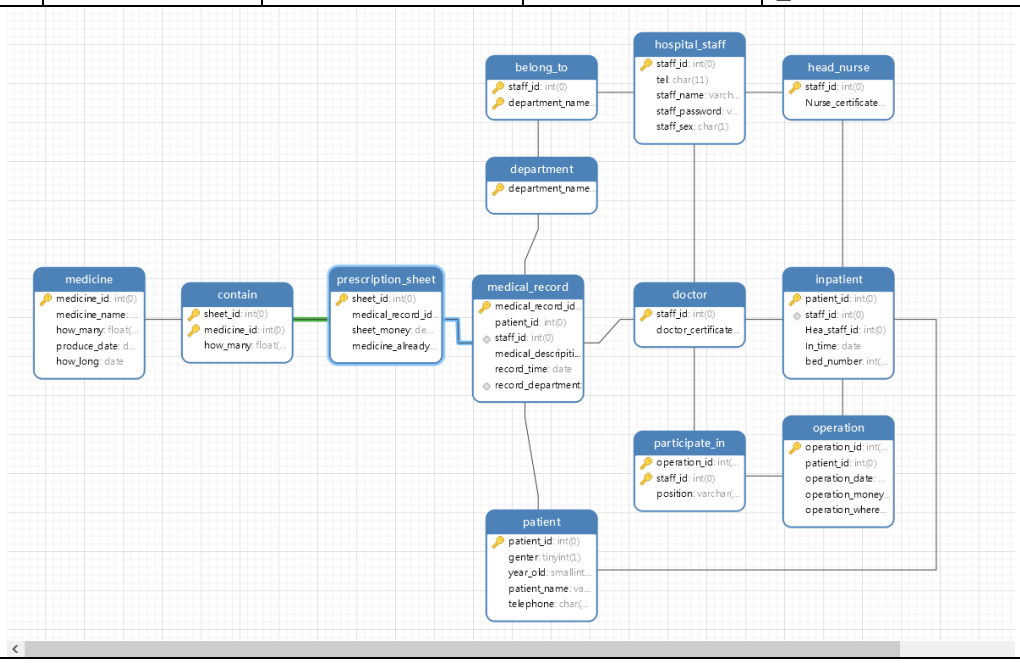
	

2. 系统配置（10 分）

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. 启动 MySQL80			
		2. 使用 Navicat 在 localhost3306 上创建数据库 medical_information			
	高级语言	1. import pymysql 导入 python 相关包			
连接串分析（6 分）		序号	名称	功能说明	取值
		1	host	连接到本地主机上的数据库服务器	localhost
		2	user	指定连接数据库时使用的用户名	root
		3	password	指定连接数据库时使用的密码	123456
		4	database	指定要连接的数据库名称	medical_information
		5	charset	指定数据库连接的字符集编码	utf8
连接串代码（截屏）（2 分）		``` db = pymysql.connect(host='localhost', user='root', password=config['MYSQL_PASSWORD'], database=config['DATABASE_NAME'], charset='utf8') cur = db.cursor() ```			
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	medicine	medicine_id		
	2	department	department_name		
	3	hospital_staf	staff_id		

		f			
	4	belong_to	(staff_id, department_name)	staff_id, department_name	hospital_staff(staff_id) department(department_name)
	5	patient	patient_id		
	6	inpatient	patient_id	patient_id	patient(patient_id)
	7	doctor	staff_id	staff_id	hospital_staff(staff_id)
	8	head_nurse	staff_id	staff_id	hospital_staff(staff_id)
	9	operation	operation_id	patient_id	inpatient(patient_id)
	10	medical_record	medical_record_id	patient_id staff_id	patient(patient_id) doctor(doctor_id)
	11	prescription_sheet	sheet_id	medical_record_id	medical_record(medical_record_id)
	12	contain	(sheet_id,medicine_id)	sheet_id,medicine_id	medicine(medicine_id) prescription_sheet(sheet_id)
关系图（4）					
备注					

4. <u>含有事务应用的删除操作（13 分）</u>

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）	住院病人被从住院病人表中删除一般有两种情况。一种是正常出院，我们只需要直接删除即可；还有一种是手术前的意外死亡。第二种情况我们需要把相关的表单内容全部进行删除。

<table>
<tr><td>涉及的表
（2 分）</td><td colspan="2">inpatient/operation/patient/medical_record/participate_in</td></tr>
<tr><td>表连接涉及字段
（1 分）</td><td colspan="2">operation.patient_id = inpatient.patient_id
operation.patient_id = patient.patient_id</td></tr>
<tr><td rowspan="2">删除条件字段描述
（1 分）</td><td>字段</td><td>规则</td></tr>
<tr><td>operation.patient_id
inpatient.patient_id
patient.patient_id</td><td>=patient_id='%s' % (patient_id)</td></tr>
<tr><td>代码
（4 分）</td><td colspan="2">

```python
@app.route('/order/delete',methods=['GET'])
def inpatient_delete():
    patient_id = request.args.get('patient_id')
    reason = request.args.get('reason')
    print(patient_id,reason)
    if reason=='正常出院':
        sql="DELETE FROM inpatient where patient_id='%s'"%(patient_id)
        query.delete(sql)
        return jsonify({'success': 1})
    elif reason=='意外去世':
        sql="DELETE FROM operation where patient_id='%s'"%(patient_id)
        query.delete(sql)
        sql="DELETE FROM inpatient where patient_id='%s'"%(patient_id)
        query.delete(sql)
        sql = "DELETE FROM patient where patient_id='%s'" % (patient_id)
        query.delete(sql)
        return jsonify({'success': 1})
    else:
        return jsonify({'success': 0})
```

</td></tr>
<tr><td>程序演示
（4 分）</td><td colspan="2">

在程序运行之前，如下表所示：

participate_in 表：

operation_id	staff_id	position
2	3	主刀医生
2	5	主刀医生
9	3	麻醉

operation 表：

operation_id	patient_id	operation_date	operation_money	operation_where
2	4	2023-05-16	872	403
9	7	2023-05-04	897	402

inpatient 表：

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

patient_id	staff_id	Hea_staff_id	In_time	bed_number
1	5	4	2023-05-11	3
4	2	4	2023-03-16	2
16	1	4	2023-05-22	14
7	2	4	2023-05-09	4

patient 表：

patient_id	genter	year_old	patient_name	telephone
1	0	19	林帆	17729515779
3	1	26	曾小贤	15991043725
4	1	89	胡一统	13546789956
7	0	28	陆展博	18546798223
10	0	27	张明哼	15974569875
15	0	19	刘昊然	15926457856
16	0	26	林洋	15967459234
17	1	58	林一	15967459753
18	0	25	孙悦	15974685126

在程序运行之后，如下表所示：

participate_in 表：

operation_id	staff_id	position
2	3	主刀医生
2	5	主刀医生

operation 表：

operation_id	patient_id	operation_date	operation_money	operation_where
2	4	2023-05-16	872	403

inpatient 表：

patient_id	staff_id	Hea_staff_id	In_time	bed_number
1	5	4	2023-05-11	3
4	2	4	2023-03-16	2

patient 表：

patient_id	genter	year_old	patient_name	telephone
1	0	19	林帆	17729515779
3	1	26	曾小贤	15991043725
4	1	89	胡一统	13546789956
10	0	27	张明哼	15974569875
15	0	19	刘昊然	15926457856
16	0	26	林洋	15967459234
17	1	58	林一	15967459753
18	0	25	孙悦	15974685126

备注

(1)tient_id=7 的陆展博进行意外去世处理，对 patient_id=16 的林洋进行正常出院处理。

(2)sql = """
DELETE operation, inpatient, patient
FROM operation
LEFT JOIN inpatient ON operation.patient_id = inpatient.patient_id
LEFT JOIN patient ON operation.patient_id = patient.patient_id
WHERE operation.patient_id = '%s'

	""" % (patient_id) query.delete(sql)（这里是我写报告的时候突然考虑到的代码中的 sql 语句的合并。） (3)在设置外键的时候我给两个表之间设置了级联删除。 ALTER TABLE participate_in ADD CONSTRAINT FK_participate_in FOREIGN KEY (operation_id) REFERENCES operation (operation_id) ON DELETE CASCADE; ALTER TABLE medical_record ADD CONSTRAINT FK_about FOREIGN KEY (patient_id) REFERENCES patient (patient_id) ON DELETE CASCADE; 这样我在删除 operation 和 patient 表的内容会同时删除这两个表对应的内容。

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">点击添加按钮，添加新的病人时，我们需要同时添加新的病历记录和病人的个人信息。在添加病人的个人信息时，要求是在数据库原信息没有的条件下才可以添加（以避免重复添加）；在添加病人的病历记录时，要求所输入的科室和职员是存在的才可以。</td></tr>
<tr><td>触发器描述
（2 分）</td><td colspan="2">当 patient 表中不存在这个人时才可以进行添加，如果已经存在，触发器不会让重复创建。当输入的 staff_id 和 department_name 不在现有表内存在，则会在触发器的作用下添加被拒绝。</td></tr>
<tr><td>涉及的表
（1 分）</td><td colspan="2">patient/department/ hospital_staff/medical_record</td></tr>
<tr><td rowspan="4">输入数据
（2 分）</td><td>字段</td><td>规则</td></tr>
<tr><td>NEW.staff_id</td><td>NOT IN (SELECT staff_id FROM hospital_staff)</td></tr>
<tr><td>NEW.record_department</td><td>NOT IN (SELECT department_name FROM department)</td></tr>
<tr><td>NEW.patient_name
NEW.telephone</td><td>EXISTS (SELECT * FROM patient WHERE patient_name =NEW.patient_name AND telephone = NEW.telephone)</td></tr>
</table>

插入操作源码（3 分）	

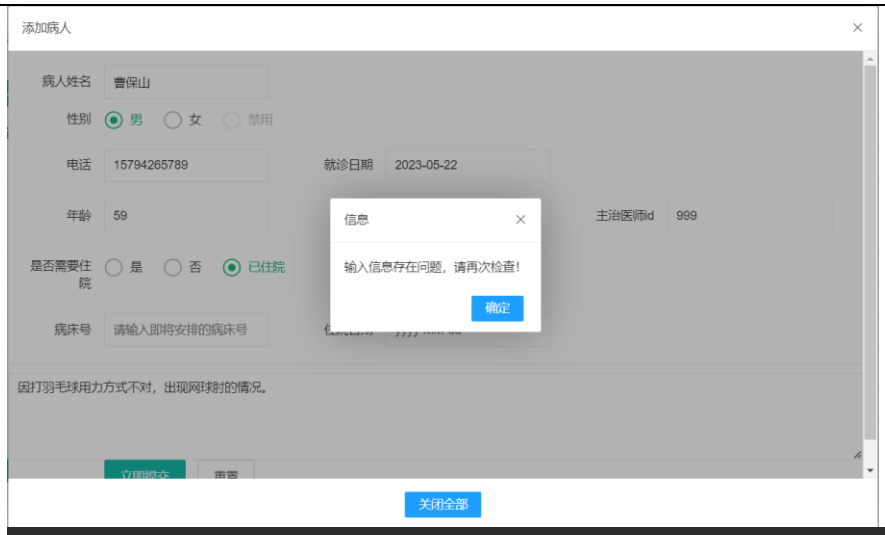
```
@app.route('/order/save',methods=['POST'])
def save_newpatient():
    name=request.form.get('patient_name')
    sex=request.form.get('sex')
    tel=request.form.get('telephone')
    date=request.form.get('date')
    year_old=request.form.get('year_old')
    department=request.form.get('department')
    doc_id=request.form.get('doctor_id')

    answer=request.form.get('need')
    bed=request.form.get('number')
    orderdate=request.form.get('orderdate')
    des=request.form.get('patient_description')
#先添加新的病人信息
    query.create_trigger1()#创建触发器用于这个病人是否已经在信息表里
    sql="INSERT INTO patient(patient_name,genter,year_old,telephone) VALUES('
    query.add(sql)
    sql="SELECT patient_id FROM patient WHERE patient_name='%s' AND telephone
    result = query.query(sql)
    patient_id=result[0][0]
#更新病历
    query.create_trigger2()
    sql="INSERT INTO medical_record(patient_id,staff_id,medical_description,
    ##创建触发器用于插入合适的值进入record表中,记录有了
    ans=query.add(sql)
    #print('the result of add list',ans)
```

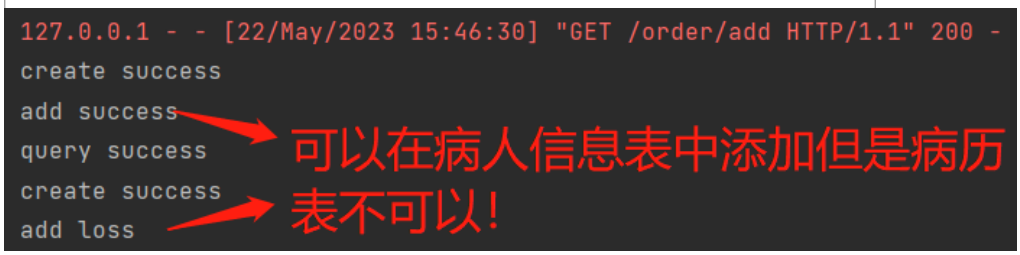
触发器源码（3 分）

```python
def create_trigger1():
    db = pymysql.connect(host='localhost', user='root', password=config['MYSQL_PASSWORD'], database=config['DATABASE_NAME'], charset='utf8')
    cur = db.cursor()
    try:
        cur.execute("DROP TRIGGER IF EXISTS add_patient;")
        # 创建触发器
        create_trigger_query = """
        CREATE TRIGGER add_patient
        BEFORE INSERT ON patient
        FOR EACH ROW
        BEGIN
            IF EXISTS (SELECT * FROM patient WHERE patient_name = NEW.patient_name AND telephone = NEW.telephone) THEN
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '数据已经存在，不需要再插入';
            END IF;
        END ;

        """

        cur.execute(create_trigger_query)
        db.commit()

        print('create success')
        result='success'

    except:
        print('create loss')
        result='loss'
        db.rollback()
    cur.close()
    db.close()
    return result
```

```python
def create_trigger2():
    db = pymysql.connect(host='localhost', user='root', password=config['MYSQL_PASSWORD'],
                         database=config['DATABASE_NAME'], charset='utf8')
    cur = db.cursor()
    try:
        cur.execute("DROP TRIGGER IF EXISTS add_medical_record;")
        # 创建触发器
        create_trigger_query = """
        CREATE TRIGGER add_medical_record
BEFORE INSERT ON medical_record
FOR EACH ROW
BEGIN
  IF NEW.staff_id NOT IN (SELECT staff_id FROM hospital_staff)
  OR NEW.record_department NOT IN (SELECT department_name FROM department)
  THEN
  SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '插入被拒绝';
  END IF;
END;
        """
        cur.execute(create_trigger_query)
        db.commit()
        print('create success')
        result = 'success'
    except:
        print('create loss')
        result = 'loss'
        db.rollback()
    cur.close()
    db.close()
    return result
```

程序演示 （4 分）	
程序演示 （4 分）	





接下来如果我们更正出正确的科室和主治医师职工号，由于刚才已经在病历表中添加成功，所以会出现以下情况：

备注	

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>我们针对处方明细上的病人是否取药，需要对数据库里的药物用量进行更新。随着取药过程的进行，我们需要减少库存药量。当库存不足时需要进行显示。库存为 0 和过期的需要删除。一切结束后，也更新处方单，标记已经交费取药。</td></tr>
<tr><td>存储过程功能描述（1 分）</td><td>输入指定的处方单号，如果处方单号对应的药品数量小于库存数量，则更新不被允许。反之，我们会依据输入的 sheet_id 对对应的药品进行更新，并标记处方表已缴费。</td></tr>
<tr><td>涉及的关系表（2 分）</td><td>sheet/contain/medicine</td></tr>
<tr><td>表连接涉及字段（1）</td><td>contain.medicine_id = medicine.medicine_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>medicine.how_many</td><td>= medicine.how_many - contain.how_many</td></tr><tr><td>……</td><td>……</td></tr></table></td></tr>
<tr><td>更新代码（3 分）</td><td>

```sql
UPDATE medicine, contain
SET medicine.how_many = medicine.how_many - contain.how_many
WHERE medicine.medicine_id = contain.medicine_id
AND contain.sheet_id = inputID;

UPDATE prescription_sheet
SET medicine_already = 1
WHERE sheet_id = inputID;
```

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

创建存储过程源码（3分）	

```
        # 创建过程
        create_PROCEDURE_query = """

CREATE PROCEDURE update_info(IN inputID INT)
BEGIN
    IF EXISTS (
        SELECT medicine.medicine_id
        FROM contain
        JOIN medicine ON contain.medicine_id = medicine.medicine_id
        WHERE contain.sheet_id = inputID
        AND contain.how_many > medicine.how_many
    )
    THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'UPDATE IS NOT ACCEPTED AS 库
    END IF;

    UPDATE medicine, contain
    SET medicine.how_many = medicine.how_many - contain.how_many
    WHERE medicine.medicine_id = contain.medicine_id
    AND contain.sheet_id = inputID;

    UPDATE prescription_sheet
    SET medicine_already = 1
    WHERE sheet_id = inputID;
END;

        """
```

存储过程执行源码（1分）

```python
#用于记录付款的处方单然后进一步更新
@app.route('/pay/answer',methods=['POST'])
def update_pay_medicine():
    value=request.form.get('value')
    sheet_id=request.form.get('id')
    print(value,sheet_id)

    #更新处方和库房信息
    if value=='是':
        result=query.create_call_procedure(sheet_id)
        print(result)
        if result=='success':
            response_data = {'success': 1}
        else:
            response_data = {'success': 0}
    print(response_data)
    return jsonify(response_data)
```

程序演示（2分）

下面我会展示更新前的数据情况：

药物编码	药物名称	库存量	生产日期	过期时间
1	阿莫西林	425	2023-02-18	2024-07-18
3	止咳糖浆	1	2023-05-12	2023-10-20

设置初始库存量

contain 表：

sheet_id	medicine_id	how_many
1	1	25
1	3	2
3	3	6

sheet 表：

sheet_id	medical_record_id	sheet_money	medicine_already
1	1	456	0
2	2	792	0
3	3	746	0

在界面填入“是”后：

更新并没有完成：

处方编码	处方金额	是否已付款取药
1	456	否
2	792	否
3	746	否

违背了存储过程：

```
127.0.0.1 - - [22/May/2023 16:25:45] "POST /pay/answer HTTP/1.1" 200 -
call procedure_loss
loss
{'success': 0}
```

程序演示（2分）

对于上面的情况，我们改变止咳糖浆的库存量：

medicine_id	medicine_name	how_many	produce_date	how_long
1	阿莫西林	425	2023-02-18	2024-07-18
3	止咳糖浆	100	2023-05-12	2023-10-20

再次重复上述过程，得到结果如下：

medicine 表：

药物编码	药物名称	库存量	生产日期	过期时间
1	阿莫西林	400	2023-02-18	2024-07-18
3	止咳糖浆	98 =100-2	2023-05-12	2023-10-20

sheet 表：

处方编码	处方金额	是否已付款取药
1	456	是
2	792	否
3	746	否

程序后台：

```
127.0.0.1 - - [22/May/2023 16:26:15] "GET /order/sheet?page=1&limit=10 HTTP/1.1" 200 -
是 1
call procedure_success
success
{'success': 1}
```

备注

我们对 sheet_id=1 的表进行取药后更新。第一个程序演示为更新失败的情景；第二个程序演示为更新成功的情景。

7. 含有视图的查询操作（15 分）

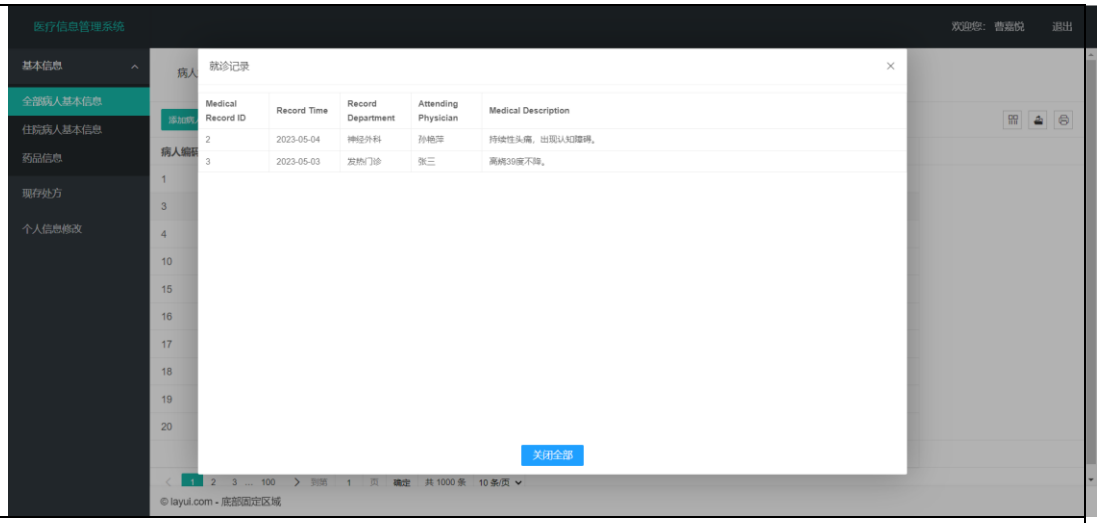
<table>
<tr><td>说明</td><td>（1 分）简要说明该操作所要完成的功能；
（1 分）简要说明建立的该视图的功能；
（2 分）简要说明该操作涉及的关系数据表（以“表名”的形式给出）
（1 分）简要说明表连接涉及的字段（以“表 1.属性=表 2.属性”）
（6 分）实现该操作的关键代码（高级语言、SQL），截图即可；
（4 分）如何执行该操作，按所述方法能够正常演示程序则给分。</td></tr>
<tr><td>操作功能描述（1 分）</td><td>我们建立视图连接病历表、医生基本信息表和病人个人信息表，从而用于在病人信息栏中显示，通过查找病人的个人基本信息，也能找到他的过往病历记录和对应的主治医生姓名等。</td></tr>
<tr><td>视图功能描述（1 分）</td><td>查询属于某个 patient_id 的过往病史信息。在 hospital_staff 表中获 staff_name。在 medical_record 表中获得 patient_id/medical_record_id/medial_descripition/record_time/record_department。</td></tr>
<tr><td>涉及的关系表（2 分）</td><td>medical_record/hospital_staff</td></tr>
<tr><td>表连接字段（1 分）</td><td>medical_record.staff_id=hospital_staff.staff_id</td></tr>
<tr><td>创建视图代码（3 分）</td><td>

```
judge_answer = query.judge_exists_view(sql)
print(judge_answer[0][0])
if (judge_answer[0][0] == 0):
    sql = """CREATE VIEW disease_details AS SELECT
    medical_record.patient_id, medical_record.medical_record_id,
    medical_record.medical_descripition,medical_record.record_time,
    medical_record.record_department, hospital_staff.staff_name, hospital_staff.staff_id
    FROM medical_record
    JOIN hospital_staff ON
    medical_record.staff_id = hospital_staff.staff_id"""
    query.create_view(sql)
```

</td></tr>
<tr><td>查询代码（3 分）</td><td>

```
#利用视图查看病人既往病史
@app.route('/old_diseases')
def old_diseases():
    id = request.args.get('patient_id')
    print("我获得",id)
    sql = "SELECT EXISTS (SELECT 1 FROM information_schema.views WHERE table_schema = 'medical_information'AND tabl
    judge_answer = query.judge_exists_view(sql)
    print(judge_answer[0][0])
    if (judge_answer[0][0] == 0):
        sql = """CREATE VIEW disease_details AS SELECT
        medical_record.patient_id, medical_record.medical_record_id,
        medical_record.medical_descripition,medical_record.record_time,
        medical_record.record_department, hospital_staff.staff_name, hospital_staff.staff_id
        FROM medical_record
        JOIN hospital_staff ON
        medical_record.staff_id = hospital_staff.staff_id"""
        query.create_view(sql)
    sql = "SELECT * FROM DISEASE_DETAILS WHERE patient_id='%s'" % (id)
    result = query.query(sql)
    categories = ['patient_id', 'medical_record_id', 'medical_descripition', 'record_time', 'record_department',
                  'staff_name']
    data = []
```

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

程序演示 （4 分）	 医疗信息管理系统 欢迎您：曹嘉锐 退出 基本信息 全部病人基本信息 住院病人基本信息 药品信息 现存处方 个人信息修改 就诊记录 Medical Record ID Record Time Record Department Attending Physician Medical Description 2 2023-05-04 神经外科 孙艳萍 持续性头痛，出现认知障碍。 3 2023-05-03 发热门诊 张三 高烧39度不降。 关闭全部 © layui.com - 底部固定区域
备注	