

Problem Statement:

In this assignment, using the MySQL database Workbench the following To-Do-List was to be completed: Create a database named hospital_portal

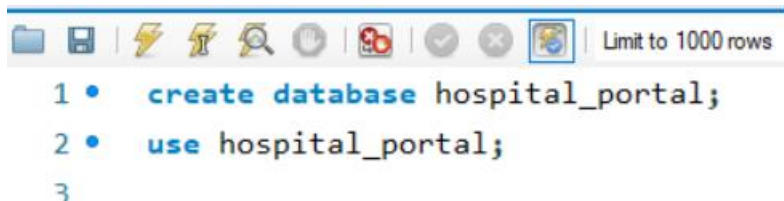
- Develop a patients table with the following attributes:
 - patient_id (int, not null, unique, auto_increment, primary key)
 - patient_name (varchar(45), not null)
 - age (int, not null)
 - admission_date (date)
 - discharge_date (date)
- Construct an Appointments table with the following attributes
 - appointment_id (int, not null, unique, auto_increment, primary key)
 - patient_id (int, not null)
 - doctor_id (int, not null)
 - appointment_date (date, not null)
 - appointment_time (decimal, not null)
- Insert some sample data into the patients' table.
- Develop stored procedures for appointment scheduling and adding patients
- Develop stored procedures for discharging patients and viewing all patient and doctor records.
- Create a table for doctors and insert sample data.
- Create a view containing the joint of doctors, appointments, and patients tables, respectively.

In addition, the integration of the above constructed database and a Server file, developed with Python's programming language, was provided at the onset of the assignment and was to be integrated and implemented using little to no knowledge of the Python to MySQL integration training. I suppose the lack of knowledge is to add to the challenge of the assignment and to sus out the individual student's resourcefulness in the face of perhaps real-world encounters with programming scenarios that may not have already known answers. Nevertheless, the following was to be completed:

- the modification of the portalDatabase.py file was to be tested by running the portalServer.py to ensure integration of the two files
- Access to the web platform at localhost:8000/
- Implementation of the following methods in portalDatabase.py file:
 - addPatient
 - scheduleAppointment
 - viewAppointments
 - dischargePatient

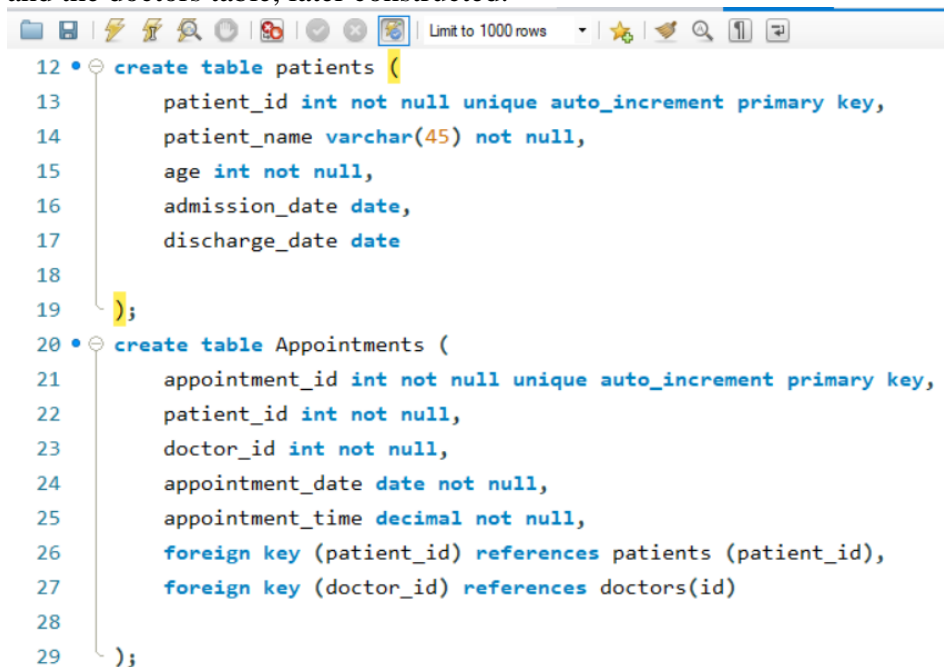
Implementation :

The 1st implementation was to create a database named hospital_portal and was done by first logging into the MySQL Workbench and using the CREATE DATABASE statement to create a new SQL database with the name 'hospital_portal'. I then used the USED statement to access the newly created database.



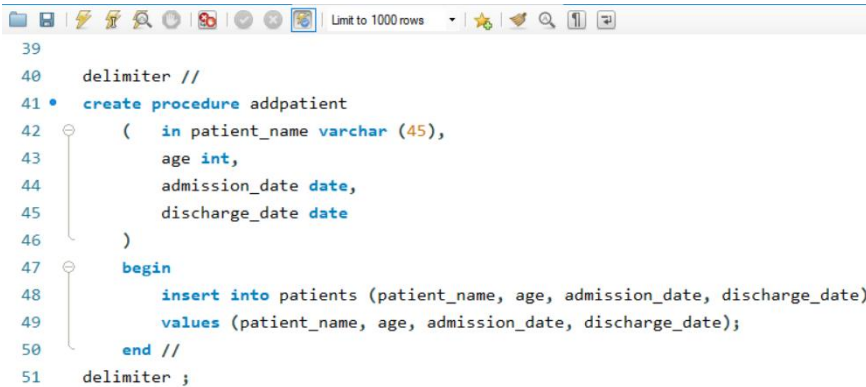
```
1 • create database hospital_portal;
2 • use hospital_portal;
3
```

With the newly created database, two tables were created labelled 'patients' and 'appointments', respectively each containing 5 attributes. In the Patients table the int, varchar, and date data types were used and in the Appointments table the datatype decimal was utilized as well. Likewise, in the Appointments table two foreign keys were assigned to the primary keys of the patients table and the doctors table, later constructed.

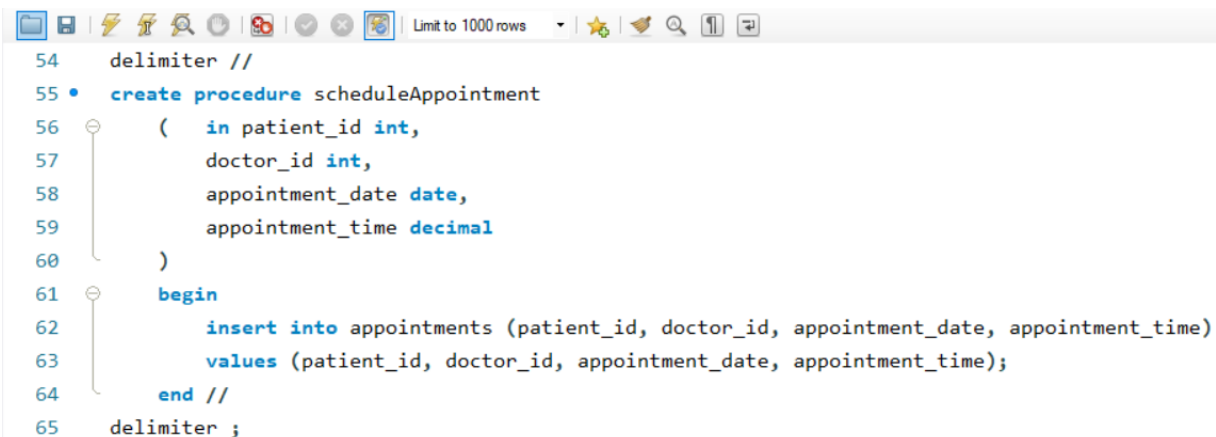


```
12 • create table patients (
13     patient_id int not null unique auto_increment primary key,
14     patient_name varchar(45) not null,
15     age int not null,
16     admission_date date,
17     discharge_date date
18 );
19
20 • create table Appointments (
21     appointment_id int not null unique auto_increment primary key,
22     patient_id int not null,
23     doctor_id int not null,
24     appointment_date date not null,
25     appointment_time decimal not null,
26     foreign key (patient_id) references patients (patient_id),
27     foreign key (doctor_id) references doctors(id)
28 );
29
```

Continuing to add to the structure of the hospital_portal database, sample data values were inserted into the patients' table and store procedures were developed to add new patients, add appointments, and show patients that had been discharged. A delimiter was used when creating the stored procedures to ensure that conflicts with semicolons within the SQL statements would be avoided.

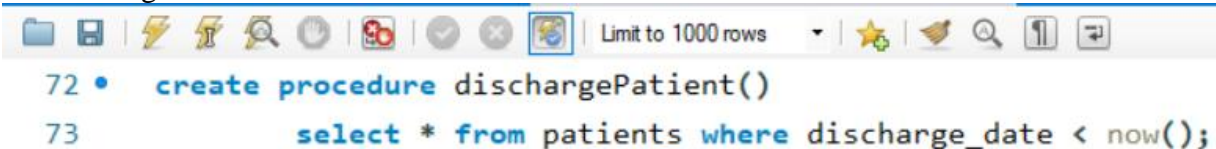


```
39
40 delimiter //
41 • create procedure addpatient
42 ( in patient_name varchar (45),
43   age int,
44   admission_date date,
45   discharge_date date
46 )
47 begin
48   insert into patients (patient_name, age, admission_date, discharge_date)
49   values (patient_name, age, admission_date, discharge_date);
50 end //
51 delimiter ;
```



```
54 delimiter //
55 • create procedure scheduleAppointment
56 ( in patient_id int,
57   doctor_id int,
58   appointment_date date,
59   appointment_time decimal
60 )
61 begin
62   insert into appointments (patient_id, doctor_id, appointment_date, appointment_time)
63   values (patient_id, doctor_id, appointment_date, appointment_time);
64 end //
65 delimiter ;
```

When creating procedures that were single lined no delimiter was used as in the cases of creating the dischargePatient view.



```
72 • create procedure dischargePatient()
73   select * from patients where discharge_date < now();
```

When creating a view containing the join of multiple tables (doctors, appointments, and patients) I used a new way of joining the three because when I attempted to join with the inner or left statements there was an overlapping conflict with the attribute names patient_id or doctor_id. As a result I found that by joining the tables with the USING() function I would not run into an error message.

```

50 • create view joinedTables
51 as
52 select * from doctors
53 join (appointments JOIN patients USING(patient_id)) USING(id);

```

Once completed I then focused most of my time on the integration and implementation of the MySQL database and the pythonServer. This is perhaps where I had the most difficulty in accomplishing completion of the tasks assigned.

To begin, The Server file was initially not functioning properly and gave the Error message within the file when I attempted to run it that and except or finally statement was expected. Finally figuring out that the EXCEPT and PASS statements were needed in the last line before RUN function, the file once running then printed out in the shell that the database has no module named 'hospitalDatabase'

```

= RESTART: C:\Users\cvrob\Documents\LehmanClasses\2023\FallTerm\CIS344\Final Project\portalServer (1).py
Traceback (most recent call last):
  File "C:\Users\cvrob\Documents\LehmanClasses\2023\FallTerm\CIS344\Final Project\portalServer (1).py", line 3, in <module>
    from hospitalDatabase import Database
ModuleNotFoundError: No module named 'hospitalDatabase'

```

To fix this issue, I realized that in the initial import statement in the portalServer.py file 'hospitalDatabase' was importing from the wrong file name, which should have been 'portalDatabase', the name of the file we are actually importing. These two initial debugging issues set field for other issues when trying to implement the requested functions.

For instance, although the implementation of the addPatient function was figured out by using the provided code to copy and paste into other sections of the body of code and tweaking the labeling of variables and input and output types, the same process could not be used with the implementation of the Schedule Appointment function. Clearly there is some oversight for which I cannot see or figure out that causes the same code used for the addPatient to be implemented, that cannot be used to implement the schedulingAppointment function. Instead, I face the white screen of death.

On a good note, however, this same screen of death was given to me repeatedly when I attempted to viewAppointments and finally seeing that the given code was the culprit again,

```

if self.path == '/viewAppointment':
    self.send_response(200)
    self.send_header('Content-type', 'text/html')
    self.end_headers()
    self.wfile.write(b"<html><head><title> Hospital's Portal")
    self.wfile.write(b"<body>")

```

I corrected the initial path designation from `viewAppointment` to `viewAppointment[s]`. I was elated to FINALLY make something work and at the same time thinking how crazy one letter of makes a huge difference. Moving forward I continued to have issues with both the `portalDatabase.py` file and the `portalServer.py` file. As stated, before the very code that would work for an implementation of one function and should technically do the same for a another, or so I think, does not.

Epilogue:

In the end, completion of the integration and implementation of what I created in the MySQL database for this assignment was uneventful. What caused the most grief was not the creation of the database itself but indeed the implementation of its methods and functions. In addition, having little knowledge of Python's server capabilities and implementations, made the task much more difficult to complete with the given time. What I did enjoy about the entire frustrating process, however, was the knowledge I gained about my strengths and weakness in this field. I have much more to learn and practice.