

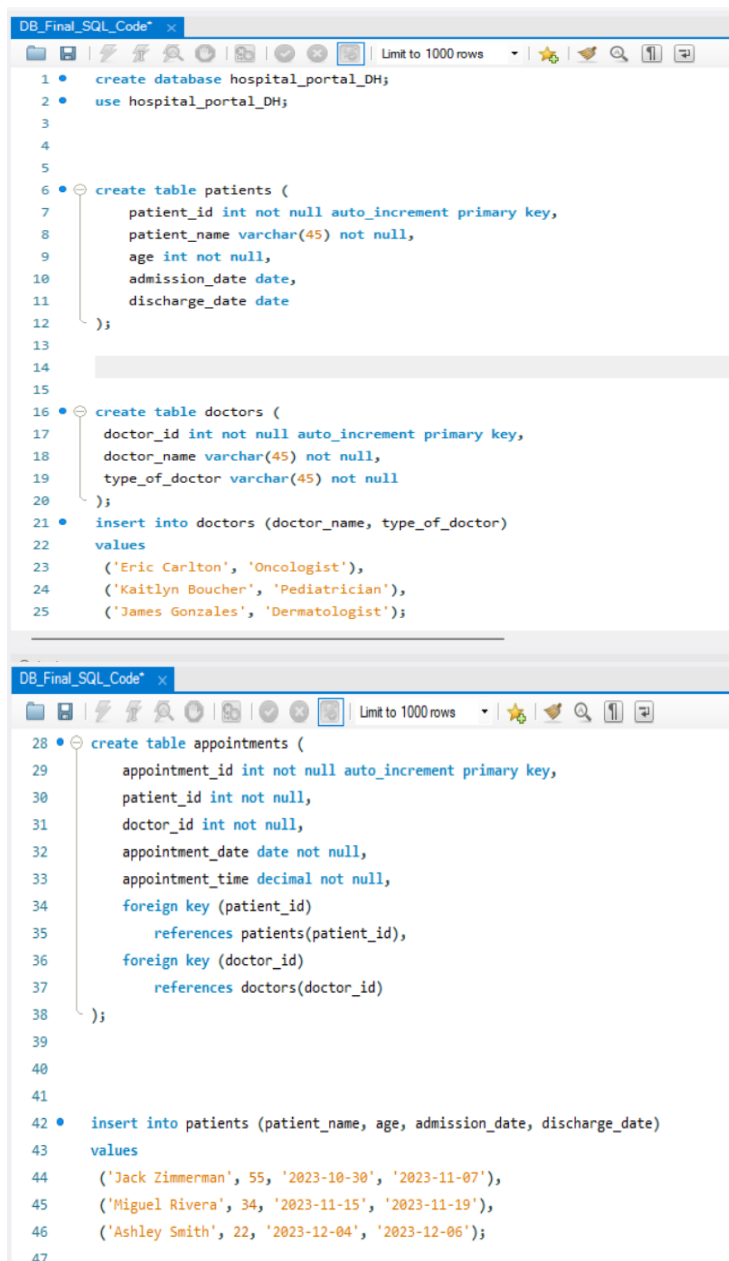
Darbert Herrand

Professor Yanilda Peralta Ramos

CIS 344

12/19/2023

DB_FINAL_REPORT

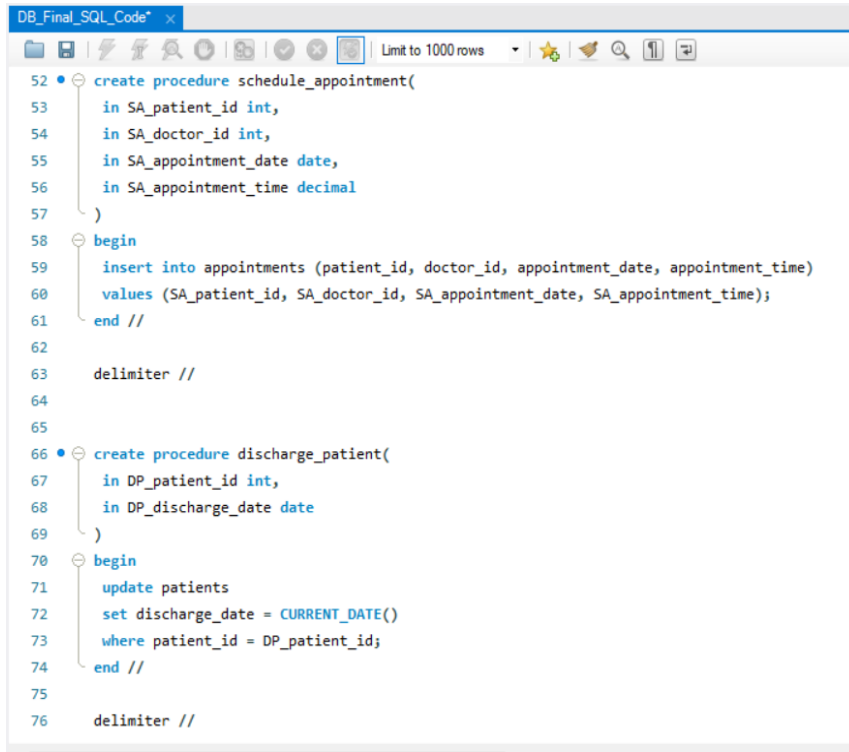


```

DB_Final_SQL_Code
1 • create database hospital_portal_DH;
2 • use hospital_portal_DH;
3
4
5
6 • create table patients (
7     patient_id int not null auto_increment primary key,
8     patient_name varchar(45) not null,
9     age int not null,
10    admission_date date,
11    discharge_date date
12 );
13
14
15
16 • create table doctors (
17     doctor_id int not null auto_increment primary key,
18     doctor_name varchar(45) not null,
19     type_of_doctor varchar(45) not null
20 );
21 • insert into doctors (doctor_name, type_of_doctor)
22 values
23     ('Eric Carlton', 'Oncologist'),
24     ('Kaitlyn Boucher', 'Pediatrician'),
25     ('James Gonzales', 'Dermatologist');
26
27
28 • create table appointments (
29     appointment_id int not null auto_increment primary key,
30     patient_id int not null,
31     doctor_id int not null,
32     appointment_date date not null,
33     appointment_time decimal not null,
34     foreign key (patient_id)
35         references patients(patient_id),
36     foreign key (doctor_id)
37         references doctors(doctor_id)
38 );
39
40
41
42 • insert into patients (patient_name, age, admission_date, discharge_date)
43 values
44     ('Jack Zimmerman', 55, '2023-10-30', '2023-11-07'),
45     ('Miguel Rivera', 34, '2023-11-15', '2023-11-19'),
46     ('Ashley Smith', 22, '2023-12-04', '2023-12-06');
47
  
```

SQL SECTION

I started my final by creating a MySQL file within my localhost server 3306. Within this file, I created the database hospital_portal_dh to represent my initials. After having this new database on my server I started to make new tables, patients, doctors, and appointments which can be seen below. I also added values to the patients and doctors tables and in all they were fairly easy to make so I had no real complications.



```

52 • create procedure schedule_appointment(
53     in SA_patient_id int,
54     in SA_doctor_id int,
55     in SA_appointment_date date,
56     in SA_appointment_time decimal
57 )
58 begin
59     insert into appointments (patient_id, doctor_id, appointment_date, appointment_time)
60     values (SA_patient_id, SA_doctor_id, SA_appointment_date, SA_appointment_time);
61 end //
62
63 delimiter //
64
65
66 • create procedure discharge_patient(
67     in DP_patient_id int,
68     in DP_discharge_date date
69 )
70 begin
71     update patients
72     set discharge_date = CURRENT_DATE()
73     where patient_id = DP_patient_id;
74 end //
75
76 delimiter //
  
```

Next was to tackle creating procedures, which were slightly harder but still not too bad. I created two stored procedures in my SQL one called `schedule_appointment` and the other was called `discharge_patient`. The webpage would call upon these to do those very actions. As shown below I created both procedures with

parameters that had different prefixes to be able to separate them from each other and the attributes in the tables. SA for the parameters in `schedule_appointment` and DP for the parameters in `discharge_patient`. I also spaced them with delimiters.

```

def scheduleAppointment(self, patient_id, doctor_id, appointment_date, appointment_time):
    ''' Method to schedule an appointment '''
    if self.connection.is_connected():
        self.cursor = self.connection.cursor()
        query = "CALL schedule_appointment(%s, %s, %s, %s);"
        self.cursor.execute(query, (patient_id, doctor_id, appointment_date, appointment_time))
        self.connection.commit()
        return

    # Implement the functionality

def viewAppointments(self):
    ''' Method to view all appointments '''
    if self.connection.is_connected():
        self.cursor = self.connection.cursor()
        query = "SELECT * FROM appointments"
        self.cursor.execute(query)
        records = self.cursor.fetchall()
        return records

    # Implement the functionality
  
```

```

def dischargePatient(self, patient_id):
    ''' Method to discharge a patient '''
    if self.connection.is_connected():
        self.cursor = self.connection.cursor()
        query = "CALL discharge_patient(%s);"
        self.cursor.execute(query, (patient_id,))
        self.connection.commit()
        return

    # Implement the functionality

# Add more methods as needed for hospital operations

def viewAllDoctors(self):
    if self.connection.is_connected():
        self.cursor = self.connection.cursor()
        query = "SELECT * FROM doctors"
        self.cursor.execute(query)
        records = self.cursor.fetchall()
        return records

```

PYTHON SECTION

In both the portalDatabase.py and the portalServer.py the previously written code helped a great deal in figuring out what would be needed of me. The easiest part of the portalDatabase.py was adding the credentials from MySQL. Although it took some trial and error developing the methods in my code became easier once I realized that the purpose of my

viewAppointments and viewAllDoctors methods were similar to the getAllPatients method that was given. Congruently, the scheduleAppointment and dischargePatient methods were similar to the already provided code for addPatient.

In the portalServer.py the hardest part was implementing and understanding what was going on in the do_GET. I looked over the provided code and reapplied a lot of it but many issues and errors were still occurring. It took a lot of rereading the code until it finally clicked and made sense. After I realized the specific goals of what the written code was trying to achieve, I applied the same code to the parts that aligned closely with my goals. For Example my code for viewAllDoctors.

```

if self.path == '/viewAllDoctors':
    data=[]
    records = self.database.viewAllDoctors()
    print(records)
    data=records
    self.send_response(200)
    self.send_header('Content-type','text/html')
    self.end_headers()
    self.wfile.write(b"<html><head><title> Hospital's Portal </title></head>")
    self.wfile.write(b"<body>")
    self.wfile.write(b"<center><h1>Hospital's Portal</h1>")
    self.wfile.write(b"<hr>")
    self.wfile.write(b"<div> <a href='/'>Home</a>| \
| | | | <a href='/addPatient'>Add Patient</a>|\
| | | | <a href='/scheduleAppointment'>Schedule Appointment</a>|\
| | | | <a href='/viewAppointments'>View Appointments</a>|\
| | | | <a href='/viewAllDoctors'>Doctors</a>|\
| | | | <a href='/dischargePatient'>Discharge Patient</a></div>")
    self.wfile.write(b"<hr><h2>All Doctors</h2>")

    self.wfile.write(b"<table border=2> \
| | | | <tr><th> Doctor ID </th>\
| | | | <th> Doctor Name </th>\
| | | | <th> Type of Doctor </th></tr>")

    for row in data:
        self.wfile.write(b' <tr> <td>')
        self.wfile.write(str(row[0]).encode())
        self.wfile.write(b'</td><td>')
        self.wfile.write(str(row[1]).encode())
        self.wfile.write(b'</td><td>')
        self.wfile.write(str(row[2]).encode())
        self.wfile.write(b'</td></tr>')

    self.wfile.write(b"</center></body></html>")
    return

```

This entire section of code was inspired by the given code for getAllPatients

WEBPAGE SECTION

This for me was the hardest part of the entire final. At first, I didn't have the correct add-ons and couldn't find a way to see my webpage. After installing the mysql.connector correctly and making sure the right credentials were placed in the portalDatabase.py I still couldn't get the webpage to appear. It turns out that in my portalSever.py where I was attempting to run the code on the used localhost 3306, I was actually making a mistake. I thought that in order to connect my SQL and portalServer.py I needed to directly express the relationship but it was already done through the portalDatabase.py. After changing my port back to 8000 I was able to view my webpage by putting localhost:8000 into my browser.

```
def run(server_class=HTTPServer, handler_class=HospitalPortalHandler, port=8000):
    server_address = ('localhost', port)
    httpd = server_class(server_address, handler_class)
    print('Starting httpd on port {}'.format(port))
    httpd.serve_forever()

run()
```

Hospital's Portal

[Home](#) | [Add Patient](#) | [Schedule Appointment](#) | [View Appointments](#) | [Doctors](#) | [Discharge Patient](#)

All Patients

Patient ID	Patient Name	Age	Admission Date	Discharge Date
1	Jack Zimmerman	55	2023-10-30	2023-11-07
2	Miguel Rivera	34	2023-11-15	2023-11-19
3	Ashley Smith	22	2023-12-04	2023-12-06
4	Cassandra Lopez	14	2023-12-19	2023-12-22