Rakshitha Devi J
superset id: 5369940
Saveetha engineering college

Coding Challenge
Hospital Management System

1.create sql schema from the following classes class, use the class attributes for table column names.

```
create database hospital_db;
use hospital_db;

create table patient (
    patientid int primary key auto_increment,
    firstname varchar(50) not null,
    lastname varchar(50) not null,
    dateofbirth date not null,
    gender varchar(10),
    contactnumber varchar(15),
    address text
);
create table doctor (
    doctorid int primary key auto_increment,
    firstname varchar(50) not null,
    lastname varchar(50) not null,
    specialization varchar(100),
    contactnumber varchar(15)
);
create table appointment (
    appointmentid int primary key auto_increment,
    patientid int,
    doctorid int,
    appointmentdate datetime not null,
    description text,
    foreign key (patientid) references patient(patientid) on delete cascade,
    foreign key (doctorid) references doctor(doctorid) on delete set null
```

);

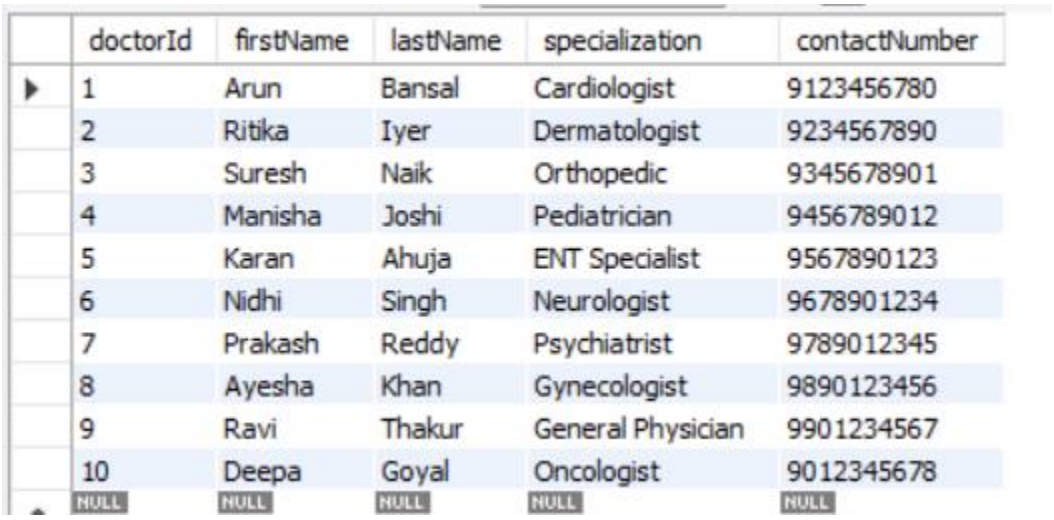
insert into patient (firstname, lastname, dateofbirth, gender, contactnumber, address) values
('rahul', 'sharma', '1990-05-12', 'male', '9876543210', 'delhi'),
('priya', 'verma', '1988-09-23', 'female', '9823456789', 'mumbai'),
('amit', 'kumar', '1985-11-15', 'male', '9812345678', 'kolkata'),
('sneha', 'patel', '1992-03-07', 'female', '9871203040', 'ahmedabad'),
('vikram', 'singh', '1989-07-30', 'male', '9845671230', 'jaipur'),
('pooja', 'mehta', '1991-12-20', 'female', '9898989898', 'pune'),
('rajesh', 'rao', '1983-06-10', 'male', '9765432101', 'hyderabad'),
('neha', 'das', '1995-04-18', 'female', '9781234567', 'bhubaneswar'),
('sandeep', 'mishra', '1993-10-05', 'male', '9776543210', 'lucknow'),
('anjali', 'kapoor', '1996-08-12', 'female', '9799999999', 'chandigarh');

select * from patient;

	patientId	firstName	lastName	dateOfBirth	gender	contactNumber	addre
▶	1	Rahul	Sharma	1990-05-12	Male	9876543210	Delhi
	2	Priya	Verma	1988-09-23	Female	9823456789	Mumb
	3	Amit	Kumar	1985-11-15	Male	9812345678	Kolkat
	4	Sneha	Patel	1992-03-07	Female	9871203040	Ahme
	5	Vikram	Singh	1989-07-30	Male	9845671230	Jaipur
	6	Pooja	Mehta	1991-12-20	Female	9898989898	Pune
	7	Rajesh	Rao	1983-06-10	Male	9765432101	Hyder
	8	Neha	Das	1995-04-18	Female	9781234567	Bhuba
	9	Sandeep	Mishra	1993-10-05	Male	9776543210	Luckn
	10	Anjali	Kapoor	1996-08-12	Female	9799999999	Chand

insert into doctor (firstname, lastname, specialization, contactnumber) values
('arun', 'bansal', 'cardiologist', '9123456780'),
('ritika', 'iyer', 'dermatologist', '9234567890'),
('suresh', 'naik', 'orthopedic', '9345678901'),
('manisha', 'joshi', 'pediatrician', '9456789012'),
('karan', 'ahuja', 'ent specialist', '9567890123'),
('nidhi', 'singh', 'neurologist', '9678901234'),
('prakash', 'reddy', 'psychiatrist', '9789012345'),
('ayesha', 'khan', 'gynecologist', '9890123456'),
('ravi', 'thakur', 'general physician', '9901234567'),

('deepa', 'goyal', 'oncologist', '9012345678');
select * from doctor;



doctorId	firstName	lastName	specialization	contactNumber
1	Arun	Bansal	Cardiologist	9123456780
2	Ritika	Iyer	Dermatologist	9234567890
3	Suresh	Naik	Orthopedic	9345678901
4	Manisha	Joshi	Pediatrician	9456789012
5	Karan	Ahuja	ENT Specialist	9567890123
6	Nidhi	Singh	Neurologist	9678901234
7	Prakash	Reddy	Psychiatrist	9789012345
8	Ayesha	Khan	Gynecologist	9890123456
9	Ravi	Thakur	General Physician	9901234567
10	Deepa	Goyal	Oncologist	9012345678
NULL	NULL	NULL	NULL	NULL

insert into appointment (patientid, doctorid, appointmentdate, description) values
(1, 1, '2025-04-10 10:00:00', 'heart checkup'),
(2, 2, '2025-04-11 11:30:00', 'skin allergy'),
(3, 3, '2025-04-12 09:45:00', 'knee pain'),
(4, 4, '2025-04-13 14:00:00', 'child fever'),
(5, 5, '2025-04-14 13:15:00', 'ear infection'),
(6, 6, '2025-04-15 15:30:00', 'migraine issues'),
(7, 7, '2025-04-16 10:45:00', 'anxiety treatment'),
(8, 8, '2025-04-17 12:00:00', 'routine pregnancy check'),
(9, 9, '2025-04-18 16:00:00', 'cold and cough'),
(10, 10, '2025-04-19 11:00:00', 'cancer consultation');

select * from appointment;

appointmentId	patientId	doctorId	appointmentDate	description
1	1	1	2025-04-10 10:00:00	Heart checkup
2	2	2	2025-04-11 11:30:00	Skin allergy
3	3	3	2025-04-12 09:45:00	Knee pain
4	4	4	2025-04-13 14:00:00	Child fever
5	5	5	2025-04-14 13:15:00	Ear infection
6	6	6	2025-04-15 15:30:00	Migraine issues
7	7	7	2025-04-16 10:45:00	Anxiety treatment
8	8	8	2025-04-17 12:00:00	Routine pregnancy ch
9	9	9	2025-04-18 16:00:00	Cold and cough
10	10	10	2025-04-19 11:00:00	Cancer consultation
NULL	NULL	NULL	NULL	NULL

2.Implement the following for all model classes. Write default constructors and overload the constructor with parameters, getters and setters, method to print all the member variables and values.

entity:
entity/__init__.py:
entity/Patient.py:

```
class Patient:
   def __init__(self, patientId=None, firstName="", lastName="",
dateOfBirth="", gender="", contactNumber="", address=""):
       self.__patientId = patientId
       self.__firstName = firstName
       self.__lastName = lastName
       self.__dateOfBirth = dateOfBirth
       self.__gender = gender
       self.__contactNumber = contactNumber
       self.__address = address

   # Getters and Setters
   def get_patientId(self): return self.__patientId
   def set_patientId(self, patientId): self.__patientId = patientId

   def get_firstName(self): return self.__firstName
   def set_firstName(self, firstName): self.__firstName = firstName

   def get_lastName(self): return self.__lastName
   def set_lastName(self, lastName): self.__lastName = lastName
```

```
    def get_dateOfBirth(self): return self.__dateOfBirth
    def set_dateOfBirth(self, dob): self.__dateOfBirth = dob

    def get_gender(self): return self.__gender
    def set_gender(self, gender): self.__gender = gender

    def get_contactNumber(self): return self.__contactNumber
    def set_contactNumber(self, contact): self.__contactNumber = contact

    def get_address(self): return self.__address
    def set_address(self, address): self.__address = address

    def __str__(self):
        return f"Patient[ID={self.__patientId}, Name={self.__firstName}
{self.__lastName}, DOB={self.__dateOfBirth}, Gender={self.__gender},
Contact={self.__contactNumber}, Address={self.__address}]"
```

entity/Doctor.py:

```
class Doctor:
    def __init__(self, doctorId=None, firstName="", lastName="",
specialization="", contactNumber=""):
        self.__doctorId = doctorId
        self.__firstName = firstName
        self.__lastName = lastName
        self.__specialization = specialization
        self.__contactNumber = contactNumber

    def get_doctorId(self): return self.__doctorId
    def set_doctorId(self, doctorId): self.__doctorId = doctorId

    def get_firstName(self): return self.__firstName
    def set_firstName(self, firstName): self.__firstName = firstName

    def get_lastName(self): return self.__lastName
    def set_lastName(self, lastName): self.__lastName = lastName

    def get_specialization(self): return self.__specialization
    def set_specialization(self, specialization): self.__specialization =
specialization

    def get_contactNumber(self): return self.__contactNumber
    def set_contactNumber(self, contact): self.__contactNumber = contact

    def __str__(self):
```

```
        return f"Doctor[ID={self.__doctorId}, Name={self.__firstName}
{self.__lastName}, Specialization={self.__specialization},
Contact={self.__contactNumber}]"
```

entity/Appointment.py:

```
class Appointment:
    def __init__(self, appointmentId=None, patientId=None, doctorId=None,
appointmentDate="", description=""):
        self.__appointmentId = appointmentId
        self.__patientId = patientId
        self.__doctorId = doctorId
        self.__appointmentDate = appointmentDate
        self.__description = description

    def get_appointmentId(self): return self.__appointmentId
    def set_appointmentId(self, appointmentId): self.__appointmentId =
appointmentId

    def get_patientId(self): return self.__patientId
    def set_patientId(self, patientId): self.__patientId = patientId

    def get_doctorId(self): return self.__doctorId
    def set_doctorId(self, doctorId): self.__doctorId = doctorId

    def get_appointmentDate(self): return self.__appointmentDate
    def set_appointmentDate(self, appointmentDate): self.__appointmentDate =
appointmentDate

    def get_description(self): return self.__description
    def set_description(self, description): self.__description = description

    def __str__(self):
        return f"Appointment[ID={self.__appointmentId},
PatientID={self.__patientId}, DoctorID={self.__doctorId},
Date={self.__appointmentDate}, Description={self.__description}]"
```

3.Define IHospitalService interface/abstract class with following methods to interact with database Keep the interfaces and implementation classes in package dao.Define HospitalServiceImpl class and implement all the methods IHospitalServiceImpl .

Dao:

dao/IHospitalService.py:

```
from abc import ABC, abstractmethod
from entity.Appointment import Appointment

class IHospitalService(ABC):

    @abstractmethod
    def getAppointmentById(self, appointmentId: int) -> Appointment:
        pass

    @abstractmethod
    def getAppointmentsForPatient(self, patientId: int) -> list:
        pass

    @abstractmethod
    def getAppointmentsForDoctor(self, doctorId: int) -> list:
        pass

    @abstractmethod
    def scheduleAppointment(self, appointment: Appointment) -> bool:
        pass

    @abstractmethod
    def updateAppointment(self, appointment: Appointment) -> bool:
        pass

    @abstractmethod
    def cancelAppointment(self, appointmentId: int) -> bool:
        pass
```

dao/HospitalServiceImpl.py:

```
import mysql.connector
from entity.Appointment import Appointment
from dao.IHospitalService import IHospitalService
from exception.PatientNumberNotFoundException import
PatientNumberNotFoundException
from util.DBConnUtil import DBConnUtil


class HospitalServiceImpl(IHospitalService):
    def __init__(self):
        self.conn = DBConnUtil.getConnection()
        self.cursor = self.conn.cursor()
```

```
    def getAppointmentById(self, appointmentId: int) -> Appointment:
        query = "SELECT * FROM Appointment WHERE appointmentId = %s"
        self.cursor.execute(query, (appointmentId,))
        row = self.cursor.fetchone()
        if row:
            return Appointment(*row)
        else:
            return None

    def getAppointmentsForPatient(self, patientId: int) -> list:
        query = "SELECT * FROM Appointment WHERE patientId = %s"
        self.cursor.execute(query, (patientId,))
        rows = self.cursor.fetchall()
        if not rows:
            raise PatientNumberNotFoundException(f"Patient with ID {patientId}
not found.")
        return [Appointment(*row) for row in rows]

    def getAppointmentsForDoctor(self, doctorId: int) -> list:
        query = "SELECT * FROM Appointment WHERE doctorId = %s"
        self.cursor.execute(query, (doctorId,))
        rows = self.cursor.fetchall()
        return [Appointment(*row) for row in rows]

    def scheduleAppointment(self, appointment: Appointment) -> bool:
        query = """
        INSERT INTO Appointment (patientId, doctorId, appointmentDate,
description)
        VALUES (%s, %s, %s, %s)
        """
        self.cursor.execute(query, (
            appointment.get_patientId(),
            appointment.get_doctorId(),
            appointment.get_appointmentDate(),
            appointment.get_description()
        ))
        self.conn.commit()
        return True

    def updateAppointment(self, appointment: Appointment) -> bool:
        query = """
        UPDATE Appointment
        SET patientId=%s, doctorId=%s, appointmentDate=%s, description=%s
        WHERE appointmentId=%s
        """
        self.cursor.execute(query, (
            appointment.get_patientId(),
            appointment.get_doctorId(),
            appointment.get_appointmentDate(),
```

```
            appointment.get_description(),
            appointment.get_appointmentId()
        ))
        self.conn.commit()
        return self.cursor.rowcount > 0

    def cancelAppointment(self, appointmentId: int) -> bool:
        query = "DELETE FROM Appointment WHERE appointmentId = %s"
        self.cursor.execute(query, (appointmentId,))
        self.conn.commit()
        return self.cursor.rowcount > 0
```

4.Create a utility class DBConnection in a package util with a static variable connection of Type Connection and a static method getConnection() which returns connection. Connection properties supplied in the connection string should be read from a property file.Create a utility class PropertyUtil which contains a static method named getPropertyString() which reads a property fie containing connection details like hostname, dbname, username, password, port number and returns a connection string.

Db.properties:

```
host=localhost
port=3306
database=hospital_db
user=root
password=rakshi430
```

util:
util/DBConnUtil.py:

```
import mysql.connector
from util.DBPropertyUtil import getPropertyString

class DBConnUtil:
    @staticmethod
    def getConnection():
        props = getPropertyString('db.properties')
        if props is None:
            return None
        try:
            conn = mysql.connector.connect(
                host=props['host'],
                port=props['port'],
```

```
                database=props['database'],
                user=props['user'],
                password=props['password']
            )
            return conn
        except mysql.connector.Error as err:
            print(f"Error connecting to the database: {err}")
            return None
```

util/DBPropertyUtil.py:

```
def getPropertyString(file_name):
    props = {}
    try:
        with open(file_name, 'r') as file:
            for line in file:
                if '=' in line and not line.startswith('#'):
                    key, value = line.strip().split('=', 1)
                    props[key.strip()] = value.strip()
    except FileNotFoundError:
        print("Properties file not found.")
        return None

    conn_str = {
        "host": props.get("host"),
        "port": props.get("port"),
        "database": props.get("database"),
        "user": props.get("user"),
        "password": props.get("password")
    }

    return conn_str
```

5.Create the exceptions in package myexceptions Define the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method

exception/PatientNumberNotFoundException.py:

```
class PatientNumberNotFoundException(Exception):
    def __init__(self, message="Patient ID not found in the database."):
        super().__init__(message)
```

6. Create class named MainModule with main method in package mainmod. Trigger all the methods in service implementation class.

main.py:

```
from dao.HospitalServiceImpl import HospitalServiceImpl
from entity.Appointment import Appointment
from exception.PatientNumberNotFoundException import
PatientNumberNotFoundException

def main():
    service = HospitalServiceImpl()

    while True:
        print("\n------- Hospital Management System -------")
        print("1. Get appointment by ID")
        print("2. Get appointments for patient")
        print("3. Get appointments for doctor")
        print("4. Schedule appointment")
        print("5. Update appointment")
        print("6. Cancel appointment")
        print("7. Exit")

        choice = input("Enter your choice: ")

        try:
            if choice == '1':
                aid = int(input("Enter Appointment ID: "))
                appt = service.getAppointmentById(aid)
                print(appt if appt else "Appointment not found.")

            elif choice == '2':
                pid = int(input("Enter Patient ID: "))
                appts = service.getAppointmentsForPatient(pid)
                for appt in appts:
                    print(appt)

            elif choice == '3':
                did = int(input("Enter Doctor ID: "))
                appts = service.getAppointmentsForDoctor(did)
                for appt in appts:
                    print(appt)

            elif choice == '4':
                pid = int(input("Enter Patient ID: "))
                did = int(input("Enter Doctor ID: "))
                date = input("Enter Appointment Date (YYYY-MM-DD): ")
                desc = input("Enter Description: ")
                appt = Appointment(None, pid, did, date, desc)
```

```
                success = service.scheduleAppointment(appt)
                print("Appointment Scheduled!" if success else "Failed to
schedule.")

            elif choice == '5':
                aid = int(input("Enter Appointment ID to Update: "))
                pid = int(input("Enter new Patient ID: "))
                did = int(input("Enter new Doctor ID: "))
                date = input("Enter new Date (YYYY-MM-DD): ")
                desc = input("Enter new Description: ")
                appt = Appointment(aid, pid, did, date, desc)
                success = service.updateAppointment(appt)
                print("Appointment Updated!" if success else "Update failed.")

            elif choice == '6':
                aid = int(input("Enter Appointment ID to Cancel: "))
                success = service.cancelAppointment(aid)
                print("Appointment Cancelled!" if success else "Cancellation
failed.")

            elif choice == '7':
                print("Exiting. Goodbye!")
                break

            else:
                print("Invalid choice. Please select again.")

        except PatientNumberNotFoundException as e:
            print("Error:", e)

        except Exception as e:
            print("Unexpected Error:", e)

if __name__ == "__main__":
    main()
```

OUTPUT:

1.Get appointment by ID:

```
------ Hospital Management System -------
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 1
Enter Appointment ID: 5
Appointment[ID=5, PatientID=5, DoctorID=5, Date=2025-04-14 13:15:00, Description=Ear infection]
```

2.Get appointments for patient:

```
------ Hospital Management System -------
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 2
Enter Patient ID: 2
Appointment[ID=2, PatientID=2, DoctorID=2, Date=2025-04-11 11:30:00, Description=Skin allergy]
```

3.Get appointments for doctor:

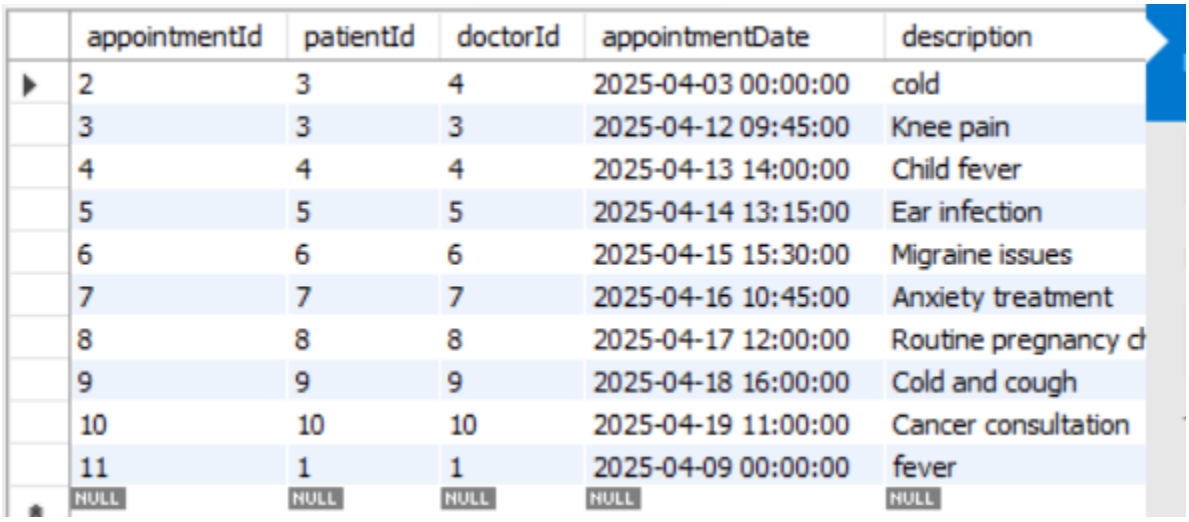
```
------ Hospital Management System -------
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 3
Enter Doctor ID: 4
Appointment[ID=4, PatientID=4, DoctorID=4, Date=2025-04-13 14:00:00, Description=Child fever]
```

4. Schedule appointment:

```
------ Hospital Management System -------
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 4
Enter Patient ID: 1
Enter Doctor ID: 1
Enter Appointment Date (YYYY-MM-DD): 2025-04-09
Enter Description: fever
Appointment Scheduled!
```

5. Update appointment:

```
------ Hospital Management System -------
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 5
Enter Appointment ID to Update: 2
Enter new Patient ID: 3
Enter new Doctor ID: 4
Enter new Date (YYYY-MM-DD): 2025-04-03
Enter new Description: cold
Appointment Updated!
```



appointmentId	patientId	doctorId	appointmentDate	description
2	3	4	2025-04-03 00:00:00	cold
3	3	3	2025-04-12 09:45:00	Knee pain
4	4	4	2025-04-13 14:00:00	Child fever
5	5	5	2025-04-14 13:15:00	Ear infection
6	6	6	2025-04-15 15:30:00	Migraine issues
7	7	7	2025-04-16 10:45:00	Anxiety treatment
8	8	8	2025-04-17 12:00:00	Routine pregnancy ch
9	9	9	2025-04-18 16:00:00	Cold and cough
10	10	10	2025-04-19 11:00:00	Cancer consultation
11	1	1	2025-04-09 00:00:00	fever
NULL	NULL	NULL	NULL	NULL

6. Cancel appointment:

```
------ Hospital Management System -------
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 6
Enter Appointment ID to Cancel: 1
Appointment Cancelled!
```

appointmentId	patientId	doctorId	appointmentDate	description
2	3	4	2025-04-03 00:00:00	cold
3	3	3	2025-04-12 09:45:00	Knee pain
4	4	4	2025-04-13 14:00:00	Child fever
5	5	5	2025-04-14 13:15:00	Ear infection
6	6	6	2025-04-15 15:30:00	Migraine issues
7	7	7	2025-04-16 10:45:00	Anxiety treatment
8	8	8	2025-04-17 12:00:00	Routine pregnancy ch
9	9	9	2025-04-18 16:00:00	Cold and cough
10	10	10	2025-04-19 11:00:00	Cancer consultation
11	1	1	2025-04-09 00:00:00	fever
NULL	NULL	NULL	NULL	NULL

7.EXIT:

```
------ Hospital Management System -------
1. Get appointment by ID
2. Get appointments for patient
3. Get appointments for doctor
4. Schedule appointment
5. Update appointment
6. Cancel appointment
7. Exit
Enter your choice: 7
Exiting. Goodbye!

Process finished with exit code 0
```