Hospital Database Full Report

Prepared for:

Jalal Omer, Ph.D.

Prepared by:

Team leader: Rafael Gayahan (RMG180000),

Virtue Adowei (VEA190000),

Alekhya Pinnamaneni (AXP190109),

Aloksai Choudari (AXC190063)

CS 4347.502

8 December 2022

Table Of Contents

Introduction…….. **3**
System Description…….…. 3
Roles…….. 3
System Requirements……..…. **4**
Context Diagram……. 4
Functional Requirements…….. 4
Non-Functional Requirements…… 5
Conceptual Design…….. **6**
Database Schema……. 6
Entity Relationship model…… 7
Business rules and integrity constraints…… 7
Logical Database Schema……. **9**
Implementation of tables……9
SQL statements for database construction and population……. 16
Expected database operations and estimated data volumes…….. 21
Functional Dependencies and Database Normalization……. **21**
The Database System……. 27
Additional Queries and Views…… **32**
Queries……. 32
Views…… 33
User application Interface…….. 36
Conclusions and Future Work…… 36
References……. 37

Introduction

The purpose of this project is to design and implement a full stack hospital database application through which parametric users can enter information to run commands such as inserting a patient, finding a schedule, and displaying various statistics that might be useful for a hospital.

System Description:

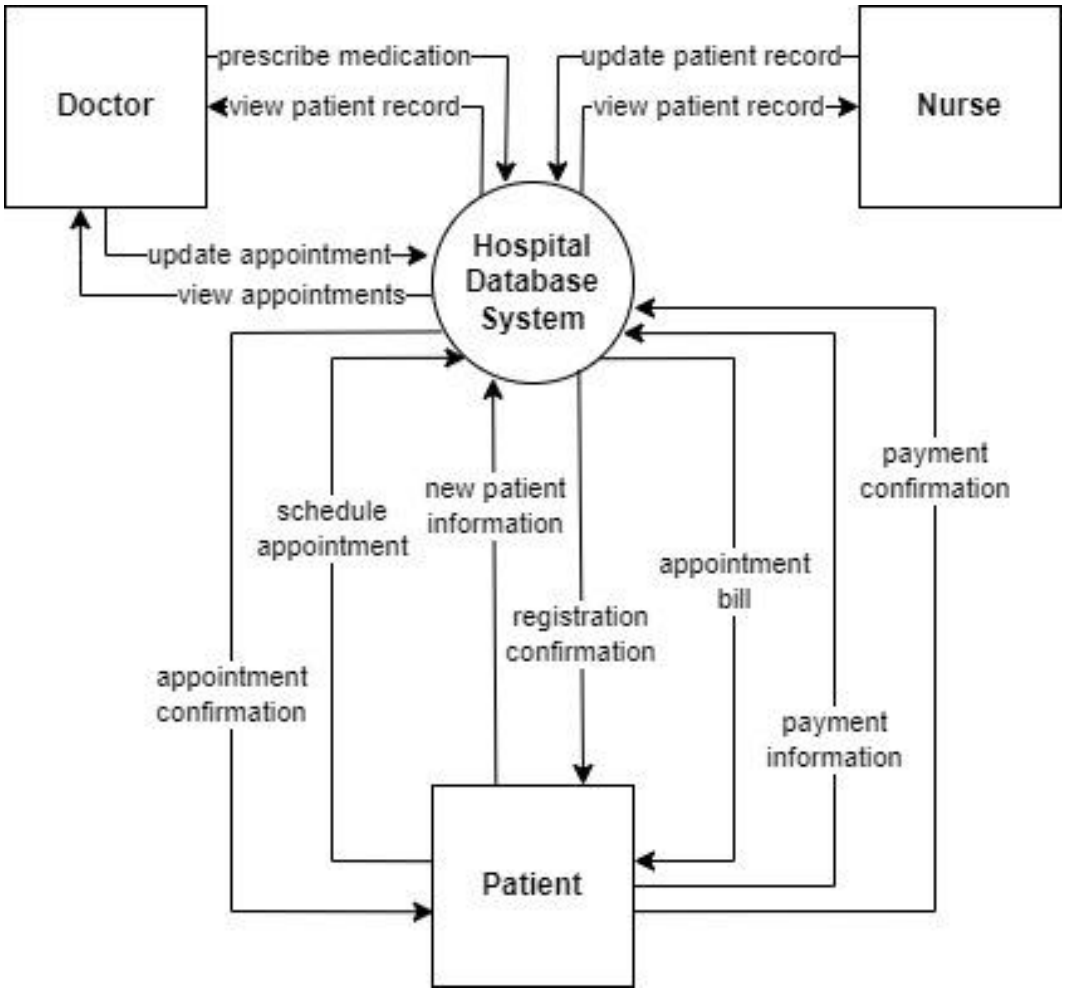
A hospital is using a database to help keep track of all information on doctors, nurses, patients, and supplies. Doctors should have personal information about them stored along with their department, specialization, and a doctor ID. Patients should have personal information, emergency contact information, and a medical record. Each patient has an assigned nurse who is able to update patient information, as well as add new patients to the registry. Each patient also has their own pharmacy. Nurses are unable to prescribe or diagnose as that should be restricted only to doctors. Doctors and nurses should also have their schedules be stored.

Roles:

Rafael was the team leader, and he worked on the system description, ER diagram, the implementation of tables in the target DBMS, and the Java SQL queries for the application . Aloksai worked on the functional requirements, business rules and integrity constraints, database views and queries, and the SQL injection. Alekhya worked on the context diagram, database schema, SQL statements for database construction and data population. Virtue worked on the non-functional requirements, interface requirements, functional dependencies and normalization, and the application GUI.

System Requirements

Context Diagram:



Functional Requirements:

- **Patients** → First Name, Last Name, Social Security Number, Patient ID, DOB (Age), Address, Contact Number, Emergency Contact First Name, Emergency Contact Last Name, Emergency Contact Number, Email
- **Medical Record** → Patient ID, Current Diagnosis, Previous Illnesses, Current Medications, Allergies, Blood Type

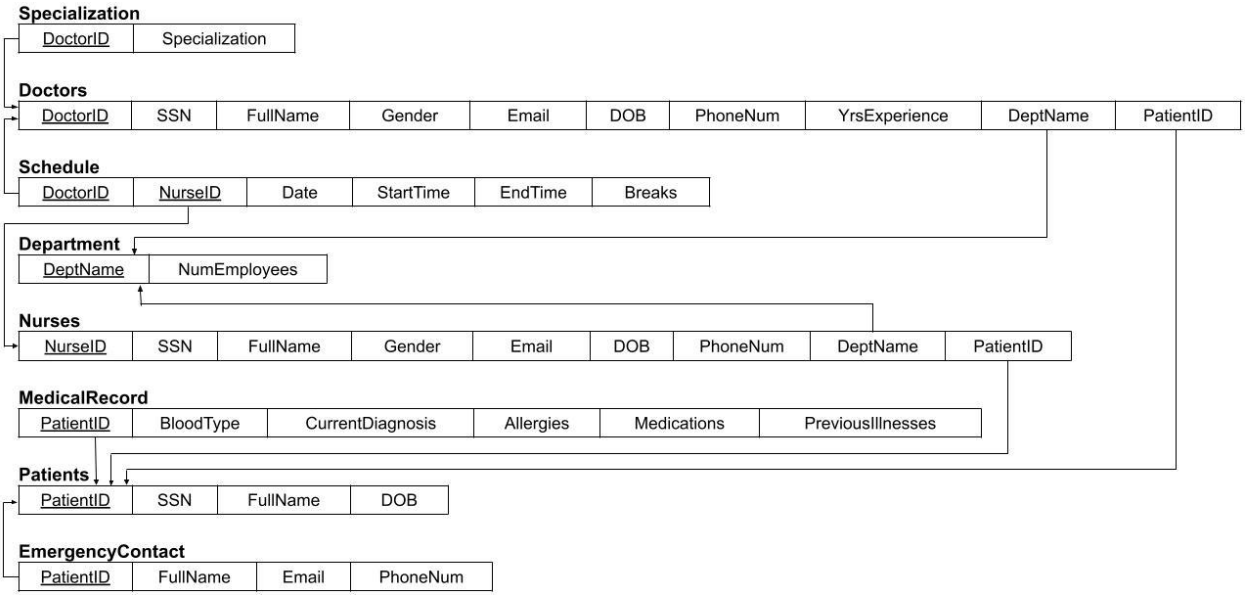
- **Doctors/Nurses** → First Name, Last Name, Social Security Number, Gender, DOB (Age), Department, Specialization, Start Year (Years of Experience), Doctor ID, Number, Email
- **Schedules** → Date, Start Time, End Time, Breaks
- Doctors/Nurses should be able to input patient information using the data above
- Doctors/Nurses should be able to query and update patient information
- Doctor information must be created to give a brief description of the doctor, along with a day-to-day schedule for the doctor with information about patients worked with
- Doctors/Nurses can create and update their own schedules based on availability

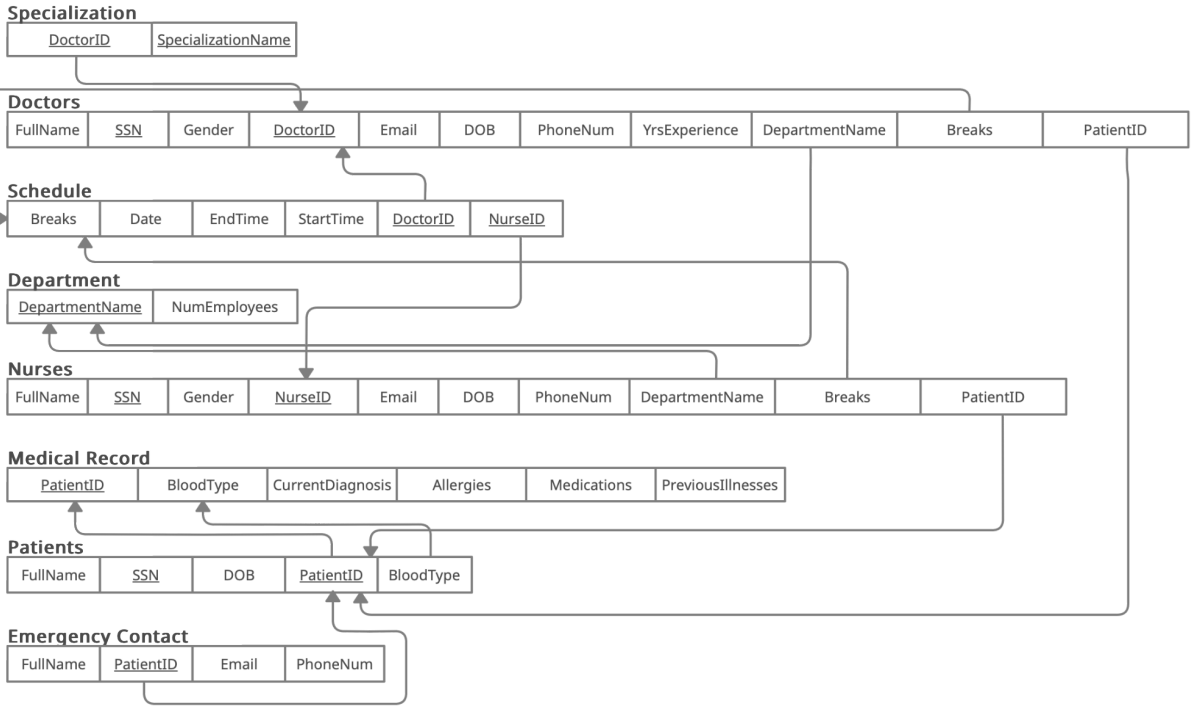
Non-functional Requirements:

- There should be data validation for each entry to ensure data integrity (elimination of uncontrolled redundancy and conflicting data entries)
- Multiple users should be able to use the system at once
- The information stored in the database should be secure
- There should be data hiding from the end user
- Different users should only have access to certain data entries and functions, depending on what sort of user they are (eg DBA vs naive user, doctor vs patient)
- The DBMS should be a relational database following a three schema architecture, which would support data independence as well as multiple views of the data
- Commands written to the DBMS should execute within a reasonable amount of time

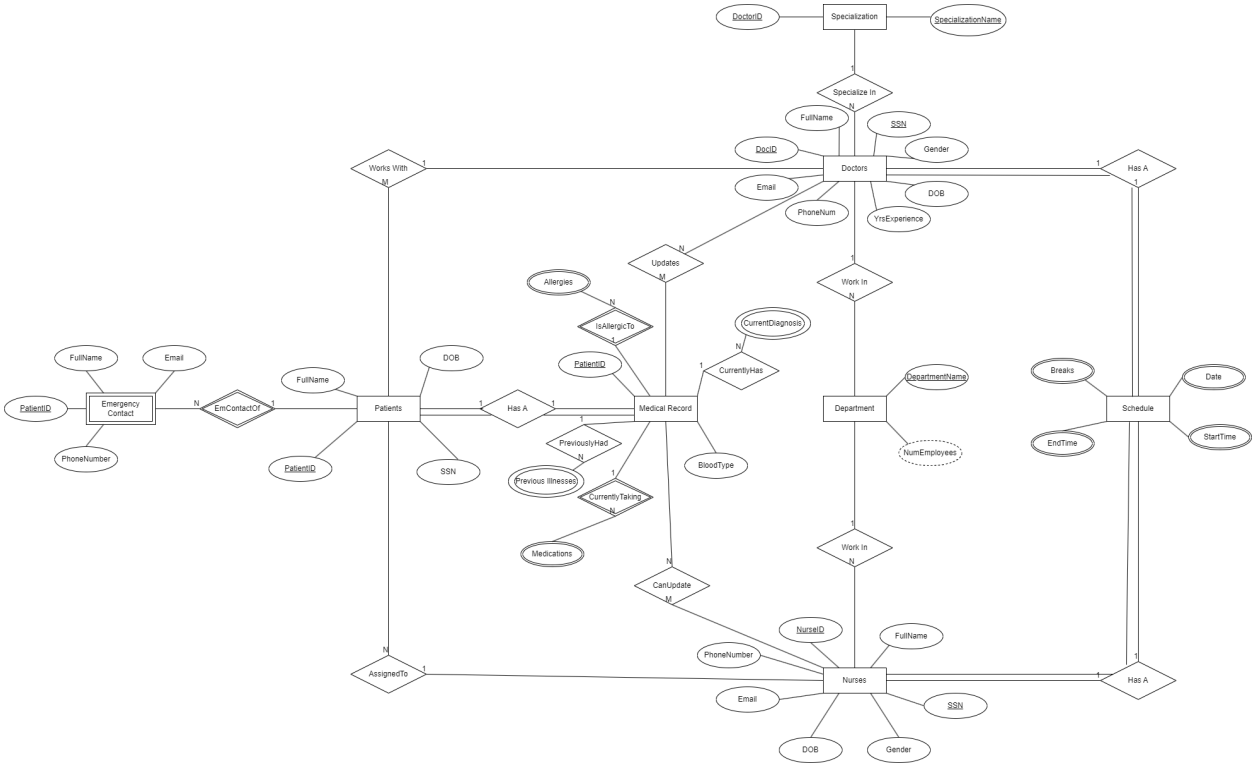
Conceptual Design

Database Schema:





Entity Relationship model:



Business rules and integrity constraints:

- Business Rules:
    - Specialization
        - DoctorID - PK, FK
    - Doctors
        - DoctorID - PK
        - SSN - SK
        - Email - SK
        - PhoneNum - SK
        - DeptName - FK
        - PatientID - FK
    - Department
        - DeptName - PK
        - NumEmployees - Derived from cardinality of the relationship between “Doctors” working in “Department”
    - Nurses
        - NurseID - PK
        - SSN - SK
        - Email - SK
        - PhoneNum - SK
        - DeptName - FK
        - PatientID - FK

    - Medical Record
        - PatientID - PK, FK
    - Patients
        - PatientID - PK
        - SSN - SK
    - Emergency Contact
        - PatientID - PK, FK
- Domain Integrity Constraints
    - Each attribute has specific data types that must be used to store information
    - For example, attributes such as FullName, Allergies, CurrentDiagnosis, Medication, Email, Gender, etc should contain only character or string data with a length of 100 characters or so
    - Similarly, integer data types for attributes representing numeric values such as DOB, SSN, PatientID, PhoneNumber, YrsExperience, DocID, NumEmployees, etc should contain only integer values
- Entity Integrity Constraints
    - Primary keys must have a value (cannot be NULL) as they will be used for tuples in relations of the database
    - Values such as PID, DoctorID, SpecializationName, SSN, PatientID, etc must have their respective data type values to represent the information in the database
- Referential Integrity Constraints
    - There must be validity in the relationships of the database
    - For example, the entity “Specialization” refers to the primary attribute DoctorID from the entity “Doctors” in order to designate the appropriate doctor to a specialization in the hospital
    - Referring to the previous constraints, the variables in each referential attribute must have the correct type and value that it is referring to, as these will act as a foreign key or a primary key to different entities
    - For example, PatientID will be a primary key attribute in the “Medical Record” entity, but refers to the “Patients” entity to a particular patient, which must be correctly allotted to
- Key Constraints
    - As stated in the referential integrity constraints, primary key attributes will be referred to across various sections in this hospital database, but each primary key must be a unique, non-NULL value
    - For example, DepartmentName is a primary attribute for the “Department” entity in this hospital database, but it must be a non-repeating value in order to prevent confusion and redundancy

Interface requirements:

- The database management system shall be able to interface with the Windows operating system, the Mac operating system, and the Linux operating system to support data independence
- The database management system shall be able to interface with the applications that will be used by the doctors, nurses, and patients
- There shall be different user interfaces for different types of end-users (doctor vs nurse vs patient), and that user interface will be determined from the user login. The user interface shall also be altered to fit the device the user accesses it from (phone vs. computer)
- For the web application, the database server should be able to interface with the application server, which acts as a conduit for partially processed data
- The user interface for the database system applications should be simple and usable, and should be a mix of form-based entries with input validation and menu-b

Logical Database Schema

Implementation of tables in the target DBMS (snapshots of tables in DBMS):
All of these tables are to be shown pre-normalization

Department Table

Result Grid | Filter Rows:

DeptName	NumEmployees
NULL	NULL

```
INSERT INTO Department(DeptName) VALUES ('Outpatient'), ('Inpatient'), ('Paramedical'),
('Rehabilitation'), ('Operation'), ('Radiolody'), ('Pharmacy');
```

DeptName	NumEmployees
Inpatient	5
Operation	2
Outpatient	2
Paramedical	1
Pharmacy	2
Radiolody	2
Rehabilitation	4
NULL	NULL

Doctor Table

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

DoctorID	SSN	FullName	Gender	Email	DOB	PhoneNum	YrsExperience	DeptName	PatientID
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
INSERT INTO Doctors VALUES
('1000100001', '000101000', 'Layken Cohen', 'F', 'cohenl@hospital.org',
'1983-02-20', '6149572626', 9, 'Outpatient', '0000000000'),
('1111111111', '000111000', 'Charlie Wynwood', 'F', 'wynwoodc@hospital.org',
'1992-05-30', '7394756298', 5, 'Inpatient', '0000000001'),
('2222222222', '000222000', 'Lily Bloom', 'F', 'blooml@hospital.org', '1977-06-21',
'2148273547', 3, 'Radiology', '0000000002'),
('3333333333', '000333000', 'Fallon ONeill', 'F', 'oneillf@hospital.org',
'1990-11-11', '4696568365', 14, 'InPatient', '0000000003'),
('4444444444', '000444000', 'Atlas Corrigan', 'M', 'corrigan@hospital.org',
'1989-03-01', '6148563527', 0, 'Rehabilitation', '0000000004'),
('5555555555', '000555000', 'Sky Davis', 'F', 'daviss@hospital.org', '1975-12-25',
'9726593836', 2, 'Pharmacy', '0000000005'),
('6666666666', '000666000', 'Megan Andrews', 'F', 'andrewsm@hospital.org',
'1983-09-19', '9726481100', 8, 'Operation', '0000000006'),
('7777777777', '000777000', 'Lowen Ashleigh', 'F', 'ashleighl@hospital.org',
'1994-10-31', '8386797000', 4, 'Paramedical', '0000000007'),
('8888888888', '000888000', 'Tate Collins', 'F', 'collinst@hospital.org',
'1981-01-12', '8269164747', 11, 'Rehabilitation', '0000000008');
```

DoctorID	SSN	FullName	Gender	Email	DOB	PhoneNum	YrsExperience	DeptName	PatientID
1000100001	000101000	Layken Cohen	F	cohenl@hospital.org	1983-02-20	6149572626	9	Outpatient	0000000000
1111111111	000111000	Charlie Wynwood	F	wynwoodc@hospital.org	1992-05-30	7394756298	5	Inpatient	0000000001
2222222222	000222000	Lily Bloom	F	blooml@hospital.org	1977-06-21	2148273547	3	Radiology	0000000002
3333333333	000333000	Fallon ONeill	F	oneillf@hospital.org	1990-11-11	4696568365	14	Inpatient	0000000003
4444444444	000444000	Atlas Corrigan	M	corrigan@hospital.org	1989-03-01	6148563527	0	Rehabilitation	0000000004
5555555555	000555000	Sky Davis	F	daviss@hospital.org	1975-12-25	9726593836	2	Pharmacy	0000000005
6666666666	000666000	Megan Andrews	F	andrewsm@hospital.org	1983-09-19	9726481100	8	Operation	0000000006
7777777777	000777000	Lowen Ashleigh	F	ashleighl@hospital.org	1994-10-31	8386797000	4	Paramedical	0000000007
8888888888	000888000	Tate Collins	F	collinst@hospital.org	1981-01-12	8269164747	11	Rehabilitation	0000000008

Emergency Contact Table

Result Grid | Filter Rows:

PatientID	FullName	Email	PhoneNum
NULL	NULL	NULL	NULL

```
INSERT INTO EmergencyContact VALUES
('0000000000', 'Jill Doe', 'jilldoe@gmail.com', '1234567890'),
('0000000001', 'James Doe', 'james1234@hotmail.com', '2141112345'),
('0000000002', 'Michael Mouse', 'mouse@gmail.com', '8121112323'),
('0000000003', 'Don Duck', 'dontheduck@yahoo.com', '8006563434'),
('0000000004', 'Collin Hoover', 'collinhoover@gmail.com', '1231231234'),
('0000000005', 'Harper Crawford', 'harperc@gmail.com', '6141234567'),
('0000000006', 'Chastin Crawford', 'chastinc@gmail.com', '8123945467'),
('0000000007', 'Ella Scissorhands', 'ilovescissors@yahoo.com', '6147697533'),
('0000000008', 'Maddie Rogers', 'maddierogers777@gmail.com', '2144338797');
```

PatientID	FullName	Email	PhoneNum
0000000000	Jill Doe	jilldoe@gmail.com	1234567890
0000000001	James Doe	james1234@hotmail.com	2141112345
0000000002	Michael Mouse	mouse@gmail.com	8121112323
0000000003	Don Duck	dontheduck@yahoo.com	8006563434
0000000004	Collin Hoover	collinhoover@gmail.com	1231231234
0000000005	Harper Crawford	harperc@gmail.com	6141234567
0000000006	Chastin Crawford	chastinc@gmail.com	8123945467
0000000007	Ella Scissorhands	ilovescissors@yahoo.com	6147697533
0000000008	Maddie Rogers	maddierogers777@gmail.com	2144338797
NULL	NULL	NULL	NULL

Medical Record Table

Result Grid | Filter Rows: | Edit: | Export/Import:

PatientID	BloodType	CurrentDiagnosis	Allergies	Medications	PreviousIllnesses
NULL	NULL	NULL	NULL	NULL	NULL

```
INSERT INTO MedicalRecord VALUES
    ('0000000000', 'O+', 'Pneuomonia', 'Sulfonamides', 'Fever reducer', 'Pneumonia'),
('0000000001', 'AB-', 'Osteoarthritis', 'N/A', 'Analgesic', 'N/A'),
('0000000002', 'B-', 'Throat Cancer', 'Aspirin, Ibuprofen', 'Chemotherapy', 'N/A'),
('0000000003', 'O+', 'Speticemia', 'Latex', '', 'N/A'),
('0000000004', 'A+', 'Cardiac dysrhythmias', 'N/A', 'Vasopressors', 'Blood clots'),
('0000000005', 'B+', 'Anemia', 'N/A', 'N/A', 'N/A'),
('0000000006', 'O-', 'Pancreatitis', 'N/A', 'N/A', 'Gallstones'),
('0000000007', 'AB+', 'Stroke', 'Penicillin', 'Anticoagulants', 'N/A'),
('0000000008', 'A+', 'Asthma Exacerbation', 'Pollen, Mold', 'Bronchodilator', 'N/A');
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Cont

PatientID	BloodType	CurrentDiagnosis	Allergies	Medications	PreviousIllnesses
0000000000	O+	Pneuomonia	Sulfonamides	Fever reducer	Pneumonia
0000000001	AB-	Osteoarthritis	N/A	Analgesic	N/A
0000000002	B-	Throat Cancer	Aspirin, Ibuprofen	Chemotherapy	N/A
0000000003	O+	Speticemia	Latex		N/A
0000000004	A+	Cardiac dysrhythmias	N/A	Vasopressors	Blood clots
0000000005	B+	Anemia	N/A	N/A	N/A
0000000006	O-	Pancreatitis	N/A	N/A	Gallstones
0000000007	AB+	Stroke	Penicillin	Anticoagulants	N/A
0000000008	A+	Asthma Exacerbation	Pollen, Mold	Bronchodilator	N/A
NULL	NULL	NULL	NULL	NULL	NULL

Nurses Table

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

NurseID	SSN	FullName	Gender	Email	DOB	PhoneNum	DeptName	PatientID
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
INSERT INTO Nurses VALUES
('100000001', '001000100', 'Ryle Kincaid', 'M', 'kincaidr@hospital.org',
'2022-11-12', '2145831486', 'Rehabilitation', '0000000000'),
('100111001', '001010100', 'Miles Archer', 'M', 'archerm@hospital.org',
'2022-11-12', '4698271515', 'Pharmacy', '0000000001'),
('200222002', '002020200', 'Ben Kessler', 'M', 'kesslerb@hospital.org',
'2022-11-12', '6141486200', 'Radiology', '0000000002'),
('300333003', '003030300', 'Dean Holder', 'M', 'holderd@hospital.org',
'2022-11-12', '4690208822', 'Inpatient', '0000000003'),
('400444004', '004040400', 'Graham Wells', 'M', 'grahamw@hospital.org',
'2022-11-12', '9721203447', 'Inpatient', '0000000004'),
('500555005', '005050500', 'Will Cooper', 'M', 'cooperw@hospital.org',
'2022-11-12', '9723248755', 'Outpatient', '0000000005'),
('600666006', '006060600', 'Silas Nash', 'M', 'nashs@hospital.org', '2022-11-12',
'6145980421', 'Operation', '0000000006'),
('700777007', '007070700', 'Leeds Gabriel', 'M', 'gabriell@hospital.org',
'2022-11-12', '6142178201', 'Inpatient', '0000000007'),
('800888008', '008080800', 'Kel Cohen', 'M', 'cohenk@hospital.org', '2022-11-12',
'8171820000', 'Rehabilitation', '0000000008');
```

NurseID	SSN	FullName	Gender	Email	DOB	PhoneNum	DeptName	PatientID
100000001	001000100	Ryle Kincaid	M	kincaidr@hospital.org	2022-11-12	2145831486	Rehabilitation	0000000000
100111001	001010100	Miles Archer	M	archerm@hospital.org	2022-11-12	4698271515	Pharmacy	0000000001
200222002	002020200	Ben Kessler	M	kesslerb@hospital.org	2022-11-12	6141486200	Radiology	0000000002
300333003	003030300	Dean Holder	M	holderd@hospital.org	2022-11-12	4690208822	Inpatient	0000000003
400444004	004040400	Graham Wells	M	grahamw@hospital.org	2022-11-12	9721203447	Inpatient	0000000004
500555005	005050500	Will Cooper	M	cooperw@hospital.org	2022-11-12	9723248755	Outpatient	0000000005
600666006	006060600	Silas Nash	M	nashs@hospital.org	2022-11-12	6145980421	Operation	0000000006
700777007	007070700	Leeds Gabriel	M	gabriell@hospital.org	2022-11-12	6142178201	Inpatient	0000000007
800888008	008080800	Kel Cohen	M	cohenk@hospital.org	2022-11-12	8171820000	Rehabilitation	0000000008
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Patient Table

Result Grid | Filter Rows:

PatientID	SSN	FullName	DOB
NULL	NULL	NULL	NULL

```
INSERT INTO Patients VALUES
('0000000000', '000000000', 'John Doe', '2022-11-12'),
('0000000001', '111111111', 'Jane Doe', '2022-11-11'),
('0000000002', '222222222', 'Mickey Mouse', '2022-11-10'),
('0000000003', '333333333', 'Donald Duck', '2022-11-09'),
('0000000004', '444444444', 'Minnie Mouse', '2022-11-08');
```

PatientID	SSN	FullName	DOB
0000000000	000000000	John Doe	2022-11-12
0000000001	111111111	Jane Doe	2022-11-11
0000000002	222222222	Mickey Mouse	2022-11-10
0000000003	333333333	Donald Duck	2022-11-09
0000000004	444444444	Minnie Mouse	2022-11-08
0000000005	555555555	Verity Crawford	1957-09-21
0000000006	666666666	Crew Crawford	2015-02-13
0000000007	777777777	Edward Scissorhands	1966-04-24
0000000008	888888888	Maggie Rogers	1983-12-04
NULL	NULL	NULL	NULL

ScheduleTable

Result Grid | Filter Rows: | Edit:

DoctorID	NurseID	Date	StartTime	EndTime	Breaks
NULL	NULL	NULL	NULL	NULL	NULL

```
INSERT INTO schedule VALUES
('1000100001', '100000001', '2022-11-13', '04:00:00', '10:00:00','30:00'),
('1111111111', '100111001', '2022-11-13', '10:00:00', '16:00:00','30:00'),
('2222222222', '200222002', '2022-11-13', '16:00:00', '22:00:00','30:00'),
('3333333333', '300333003', '2022-11-14', '22:00:00', '04:00:00','30:00'),
('4444444444', '400444004', '2022-11-14', '04:00:00', '10:00:00','30:00'),
('5555555555', '500555005', '2022-11-14', '10:00:00', '16:00:00','30:00'),
('6666666666', '600666006', '2022-11-14', '16:00:00', '22:00:00','30:00'),
('7777777777', '700777007', '2022-11-15', '22:00:00', '04:00:00','30:00'),
('8888888888', '800888008', '2022-11-15', '04:00:00', '10:00:00','30:00');
```

Result Grid | Filter Rows: | Edit: | Export/I

DoctorID	NurseID	Date	StartTime	EndTime	Breaks
1000100001	100000001	2022-11-13	04:00:00	10:00:00	30:00:00
1111111111	100111001	2022-11-13	10:00:00	16:00:00	30:00:00
2222222222	200222002	2022-11-13	16:00:00	22:00:00	30:00:00
3333333333	300333003	2022-11-14	22:00:00	04:00:00	30:00:00
4444444444	400444004	2022-11-14	04:00:00	10:00:00	30:00:00
5555555555	500555005	2022-11-14	10:00:00	16:00:00	30:00:00
6666666666	600666006	2022-11-14	16:00:00	22:00:00	30:00:00
7777777777	700777007	2022-11-15	22:00:00	04:00:00	30:00:00
8888888888	800888008	2022-11-15	04:00:00	10:00:00	30:00:00
NULL	NULL	NULL	NULL	NULL	NULL

Specialization Table

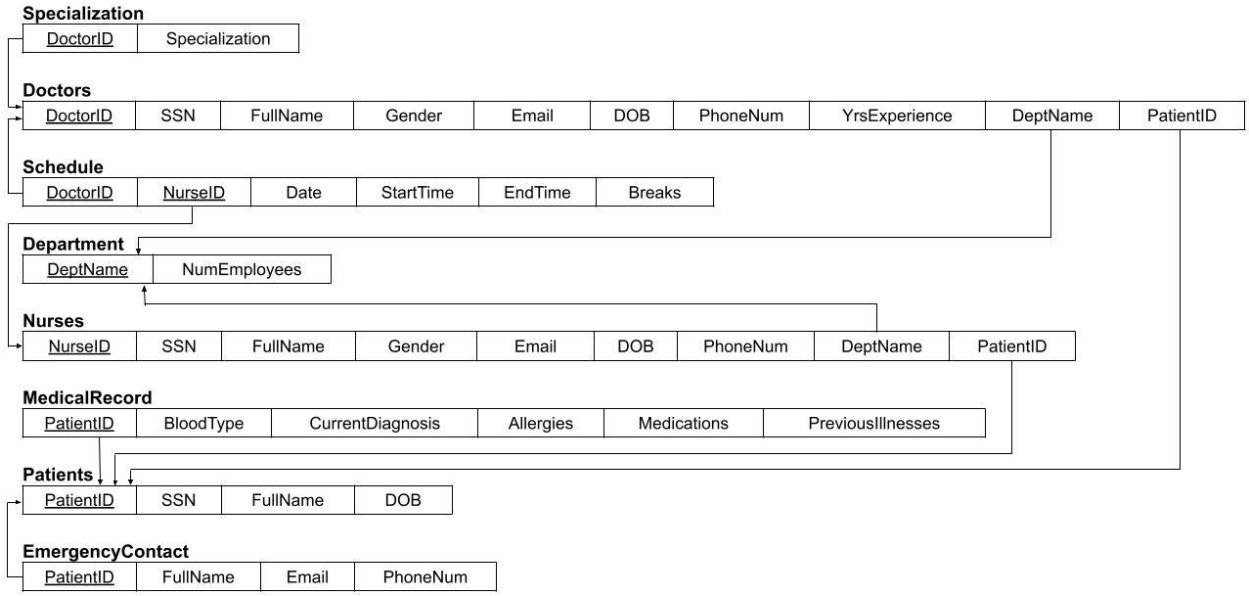
Result Grid | Filter Rows:

DoctorID	SpecializationName
NULL	NULL

```
INSERT INTO Specialization VALUES
('1000100001', 'Pulmonology'),
('1111111111', 'Rheumatology'),
('2222222222', 'Otolaryngology'),
('3333333333', 'Intensive Care'),
('4444444444', 'Cardiology'),
('5555555555', 'Hematology'),
('6666666666', 'Pancreatology'),
('7777777777', 'Vascular'),
('8888888888', 'Immunology');
```

DoctorID	SpecializationName
1000100001	Pulmonology
1111111111	Rheumatology
2222222222	Otolaryngology
3333333333	Intensive Care
4444444444	Cardiology
5555555555	Hematology
6666666666	Pancreatology
7777777777	Vascular
8888888888	Immunology
NULL	NULL

SQL statements for database construction and data population:



- **Construct the database**
  - CREATE TABLE Department (
    - DeptName VARCHAR(15) NOT NULL,
    - NumEmployees INT NOT NULL DEFAULT 0,
    - PRIMARY KEY (DeptName));
  - CREATE TABLE Patients (
    - PatientID CHAR(10) NOT NULL,

- SSN CHAR(9) NOT NULL,
- FullName VARCHAR(20) NOT NULL,
- DOB DATE NOT NULL,
- PRIMARY KEY (PatientID),
- UNIQUE (SSN));

- CREATE TABLE MedicalRecord (
  - PatientID CHAR(10) NOT NULL,
  - BloodType CHAR(3) NOT NULL,
  - CurrentDiagnosis VARCHAR(20),
  - Allergies VARCHAR(20),
  - Medications VARCHAR(30),
  - PreviousIllnesses VARCHAR(30),
  - PRIMARY KEY (PatientID),
  - FOREIGN KEY (PatientID) REFERENCES Patients (PatientID)
    - ON DELETE CASCADE ON UPDATE CASCADE);
- CREATE TABLE EmergencyContact (
  - PatientID CHAR(10) OT NULL,
  - FullName VARCHAR(20) NOT NULL,
  - Email VARCHAR(25) NOT NULL,
  - PhoneNum CHAR(10) NOT NULL,
  - PRIMARY KEY (PatientID),
  - FOREIGN KEY (PatientID) REFERENCES Patients (PatientID)
    - ON DELETE CASCADE ON UPDATE CASCADE);
- CREATE TABLE Schedule (
  - DoctorID CHAR(10) NOT NULL,
  - NurseID CHAR(10) NOT NULL,
  - Date DATE NOT NULL,
  - StartTime TIME NOT NULL,
  - EndTime TIME NOT NULL,
  - Breaks INTERVAL,
  - PRIMARY KEY(DoctorID, NurseID));
- CREATE TABLE Nurses (
  - NurseID CHAR(10) NOT NULL,
  - SSN CHAR(9) NOT NULL,
  - FullName VARCHAR(20) NOT NULL,
  - Gender CHAR(1) NOT NULL,
  - Email VARCHAR(25) NOT NULL,
  - DOB DATE NOT NULL,
  - PhoneNum CHAR(10) NOT NULL,
  - DeptName VARCHAR(15) NOT NULL,
  - PatientID CHAR(10) NOT NULL,
  - PRIMARY KEY (NurseID),
  - UNIQUE (SSN, Email, PhoneNum),

- FOREIGN KEY (DeptName) REFERENCES Department (DeptName)
  - ON DELETE RESTRICT ON UPDATE CASCADE,
- FOREIGN KEY (PatientID) REFERENCES Patients (PatientID)
  - ON DELETE SET NULL ON UPDATE CASCADE);

- CREATE TABLE Doctors (
  - DoctorID CHAR(10) NOT NULL,
  - SSN CHAR(9) NOT NULL,
  - FullName VARCHAR(20) NOT NULL,
  - Gender CHAR(1) NOT NULL,
  - Email VARCHAR(25) NOT NULL,
  - DOB DATE NOT NULL,
  - PhoneNum CHAR(10) NOT NULL,
  - YrsExperienceINT NOT NULL,
  - DeptName VARCHAR(15) NOT NULL,
  - PatientID CHAR(10),
  - PRIMARY KEY (DoctorID),
  - UNIQUE (SSN, Email, PhoneNum),
  - FOREIGN KEY (DeptName) REFERENCES Department (DeptName)
    - ON DELETE RESTRICT ON UPDATE CASCADE,
  - FOREIGN KEY (PatientID) REFERENCES Patients (PatientID)
    - ON DELETE SET NULL ON UPDATE CASCADE);
- CREATE TABLE Specialization (
  - DoctorID CHAR(10) NOT NULL,
  - Specialization VARCHAR(15) NOT NULL,
  - PRIMARY KEY (DoctorID),
  - FOREIGN KEY (DoctorID) REFERENCES Doctors (DoctorID)
    - ON DELETE CASCADE ON UPDATE CASCADE);
- ALTER TABLE Schedule
  - ADD FOREIGN KEY (DoctorID) REFERENCES Doctors (DoctorID)
    - ON DELETE RESTRICT ON UPDATE CASCADE;
- ALTER TABLE Schedule
  - ADD FOREIGN KEY (NurseID) REFERENCES Nurses (NurseID)
    - ON DELETE RESTRICT ON UPDATE CASCADE;

- **Populate the database:**
  - INSERT INTO Department (DeptName) VALUES (‘Outpatient’), (‘Inpatient’), (‘Paramedical’),(‘Rehabilitation’), (‘Operation’), (‘Radiology’), (‘Pharmacy’);
  - INSERT INTO Patients VALUES
    ('0000000000', '000000000', 'John Doe', '2021-01-01'),
    ('0000000001', '111111111', 'Jane Doe', '2008-10-31'),
    ('0000000002', '222222222', 'Mickey Mouse', '1972-11-17'),
    ('0000000003', '333333333', 'Donald Duck', '1995-03-09'),

('0000000004', '444444444', 'Colleen Hoover', '2020-11-28'),
('0000000005', '555555555', 'Verity Crawford', '1957-09-21'),
('0000000006', '666666666', 'Crew Crawford', '2015-02-13'),
('0000000007', '777777777', 'Edward Scissorhands', '1966-04-24'),
('0000000008', '888888888', 'Maggie Rogers', '1983-12-04');

- INSERT INTO MedicalRecord VALUES
  ('0000000000', 'O+', 'Pneuomonia', 'Sulfonamides', 'Fever reducer', 'Pneumonia'),
  ('0000000001', 'AB-', 'Osteoarthritis', 'N/A', 'Analgesic', 'N/A'),
  ('0000000002', 'B-', 'Throat Cancer', 'Aspirin, Ibuprofen', 'Chemotherapy', 'N/A'),
  ('0000000003', 'O+', 'Speticemia', 'Latex', ‘N/A’, 'N/A'),
  ('0000000004', 'A+', 'Cardiac dysrhythmias', 'N/A', 'Vasopressors', 'Blood clots'),
  ('0000000005', 'B+', 'Anemia', 'N/A', 'N/A', 'N/A'),
  ('0000000006', 'O-', 'Pancreatitis', 'N/A', 'N/A', 'Gallstones'),
  ('0000000007', 'AB+', 'Stroke', 'Penicillin', 'Anticoagulants', 'N/A'),
  ('0000000008', 'A+', 'Asthma Exacerbation', 'Pollen, Mold', 'Bronchodilator', 'N/A');
- INSERT INTO EmergencyContact VALUES
  ('0000000000', 'Jill Doe', 'jilldoe@gmail.com', '1234567890'),
  ('0000000001', 'James Doe', 'james1234@hotmail.com', '2141112345'),
  ('0000000002', 'Michael Mouse', 'mouse@gmail.com', '8121112323'),
  ('0000000003', 'Don Duck', 'dontheduck@yahoo.com', '8006563434'),
  ('0000000004', 'Collin Hoover', 'collinhoover@gmail.com', '1231231234'),
  ('0000000005', 'Harper Crawford', 'harperc@gmail.com', '6141234567'),
  ('0000000006', 'Chastin Crawford', 'chastinc@gmail.com', '8123945467'),
  ('0000000007', 'Ella Scissorhands', 'ilovescissors@yahoo.com', '6147697533'),
  ('0000000008', 'Maddie Rogers', 'maddierogers777@gmail.com', '2144338797');
- INSERT INTO Doctors VALUES
  ('1000100001', '000101000', 'Layken Cohen', 'F', 'cohenl@hospital.org',
  '1983-02-20', '6149572626', 9, 'Outpatient', '0000000000'),
  ('1111111111', '000111000', 'Charlie Wynwood', 'F', 'wynwoodc@hospital.org',
  '1992-05-30', '7394756298', 5, 'Inpatient', '0000000001'),
  ('2222222222', '000222000', 'Lily Bloom', 'F', 'blooml@hospital.org', '1977-06-21',
  '2148273547', 3, 'Radiology', '0000000002'),
  ('3333333333', '000333000', 'Fallon ONeill', 'F', 'oneillf@hospital.org',
  '1990-11-11', '4696568365', 14, 'Inpatient', '0000000003'),
  ('4444444444', '000444000', 'Atlas Corrigan', 'M', 'corrigan@hospital.org',
  '1989-03-01', '6148563527', 0, 'Rehabilitation', '0000000004'),
  ('5555555555', '000555000', 'Sky Davis', 'F', 'daviss@hospital.org', '1975-12-25',
  '9726593836', 2, 'Pharmacy', '0000000005'),
  ('6666666666', '000666000', 'Megan Andrews', 'F', 'andrewsm@hospital.org',
  '1983-09-19', '9726481100', 8, 'Operation', '0000000006'),
  ('7777777777', '000777000', 'Lowen Ashleigh', 'F', 'ashleighl@hospital.org',
  '1994-10-31', '8386797000', 4, 'Paramedical', '0000000007'),
  ('8888888888', '000888000', 'Tate Collins', 'F', 'collinst@hospital.org',
  '1981-01-12', '8269164747', 11, 'Rehabilitation', '0000000008');

- INSERT INTO Nurses VALUES
  ('100000001', '001000100', 'Ryle Kincaid', 'M', 'kincaidr@hospital.org',
  '2022-11-12', '2145831486', 'Rehabilitation', '0000000000'),
  ('100111001', '001010100', 'Miles Archer', 'M', 'archerm@hospital.org',
  '2022-11-12', '4698271515', 'Pharmacy', '0000000001'),
  ('200222002', '002020200', 'Ben Kessler', 'M', 'kesslerb@hospital.org',
  '2022-11-12', '6141486200', 'Radiology', '0000000002'),
  ('300333003', '003030300', 'Dean Holder', 'M', 'holderd@hospital.org',
  '2022-11-12', '4690208822', 'Inpatient', '0000000003'),
  ('400444004', '004040400', 'Graham Wells', 'M', 'grahamw@hospital.org',
  '2022-11-12', '9721203447', 'Inpatient', '0000000004'),
  ('500555005', '005050500', 'Will Cooper', 'M', 'cooperw@hospital.org',
  '2022-11-12', '9723248755', 'Outpatient', '0000000005'),
  ('600666006', '006060600', 'Silas Nash', 'M', 'nashs@hospital.org', '2022-11-12',
  '6145980421', 'Operation', '0000000006'),
  ('700777007', '007070700', 'Leeds Gabriel', 'M', 'gabriell@hospital.org',
  '2022-11-12', '6142178201', 'Inpatient', '0000000007'),
  ('800888008', '008080800', 'Kel Cohen', 'M', 'cohenk@hospital.org', '2022-11-12',
  '8171820000', 'Rehabilitation', '0000000008');
- INSERT INTO Specialization VALUES
  ('1000100001', 'Pulmonology'),
  ('1111111111', 'Rheumatology'),
  ('2222222222', 'Otolaryngology'),
  ('3333333333', 'Intensive Care'),
  ('4444444444', 'Cardiology'),
  ('5555555555', 'Hematology'),
  ('6666666666', 'Pancreatology'),
  ('7777777777', 'Vascular'),
  ('8888888888', 'Immunology');
- INSERT INTO Schedule (DoctorID, NurseID, Date, StartTime, EndTime)
  VALUES
  (‘1000100001’, ‘100000001’, ‘2022-11-13’, ‘04:00’, ‘10:00’),
  (‘11111111111’, ‘100111001’, ‘2022-11-13’, ‘10:00’, ‘16:00’),
  (‘2222222222’, ‘200222002’, ‘2022-11-13’, ‘16:00’, ‘22:00’),
  (‘3333333333’, ‘300333003’, ‘2022-11-14’, ‘22:00’, ‘04:00’),
  (‘4444444444’, ‘400444004’, ‘2022-11-14’, ‘04:00’, ‘10:00’),
  (‘5555555555’, ‘500555005’, ‘2022-11-14’, ‘10:00’, ‘16:00’),
  (‘6666666666’, ‘600666006’, ‘2022-11-14’, ‘16:00’, ‘22:00’),
  (‘7777777777’, ‘700777007’, ‘2022-11-15’, ‘22:00’, ‘04:00’),
  (‘8888888888’, ‘800888008’, ‘2022-11-15’, ‘04:00’, ‘10:00’);
- UPDATE Department
  SET NumEmployees = (
  SELECT COUNT(*)
  FROM Doctors

```
WHERE Doctors.DeptName = Department.DeptName) +
(SELECT COUNT(*)
FROM Nurses
WHERE Nurses.DeptName = Department.DeptName);
```

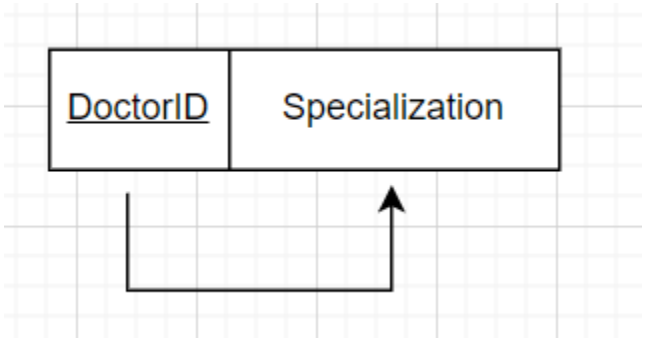
Expected database operations and estimated data volumes:

Based on our schema, we expect database operations such as inserting, looking for, modifying, or deleting a patient, doctor, or nurse, and we expect that those operations will be used several times a day, so there will most likely be a high data volume for those operations. We also anticipate the need to aggregate data and run statistics on different data items such as finding out who is on schedule, how many patients have a certain blood type, what sort of specializations the doctors have, and other data that would be of use to a hospital. These operations may not be used as frequently as insertions or updates, but we anticipate that they might be used daily for the regular operations of the hospital, so it would have a moderate data volume.

Functional Dependencies and Database Normalization

Specialization table:

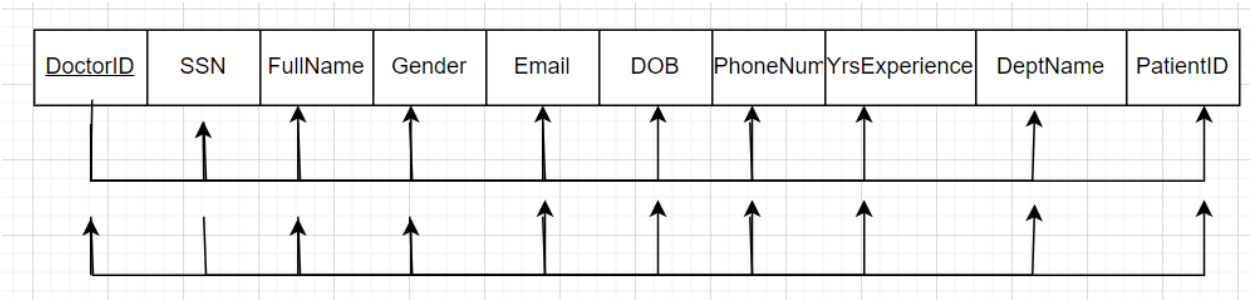
- Specialization is dependent on DoctorID



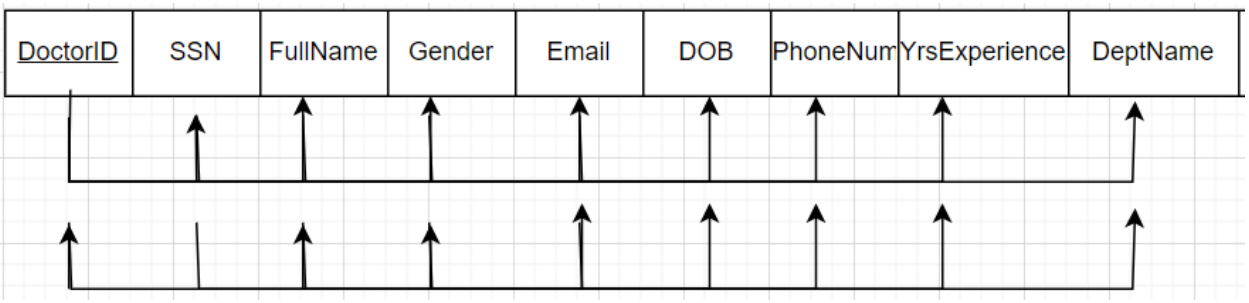
The above table is in 1NF because it does not have composite attributes, multivalued attributes, nested relations, or attributes whose values for an individual tuple are non-atomic. It is in 2NF because it is in 1NF, and there are no partial dependencies. It is 3NF because it is in 2NF and it does not have any transitive dependencies of none-prime attributes on a candidate key through another non-prime attribute. It is in BCNF because it is in 3NF and attributes are only dependent on superkeys.

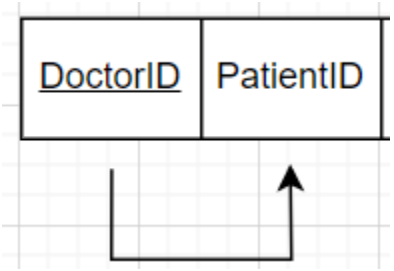
Doctors table

- SSN, Fullname, Gender, Email, DOB, PhoneNum, YrsExperience, DeptName, and PatientID all depend on the DoctorID
- DoctorID, Fullname, Gender, Email, DOB, PhoneNum, YrsExperience, DeptName, and PatientID all depend on the SSN



The above table is not in 1NF because PatientID can have multiple values. The normalized tables are below

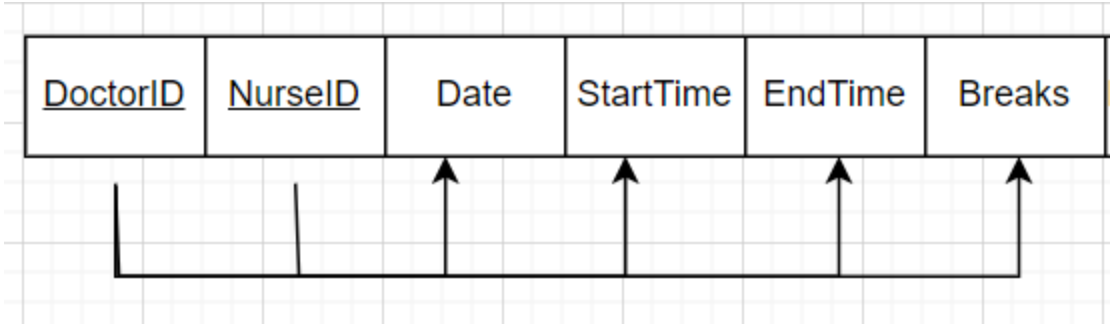




The above tables are in 1NF because they do not have composite attributes, multivalued attributes, nested relations, or attributes whose values for an individual tuple are non-atomic. They are in 2NF because they are in 1NF, and there are no partial dependencies. They are in 3NF because they are in 2NF and they do not have any transitive dependencies of none-prime attributes on a candidate key through another non-prime attribute. They are in BCNF because they are in 3NF and attributes are only dependent on superkeys. I assume a doctor has just one phone number and email address on file.

Schedule

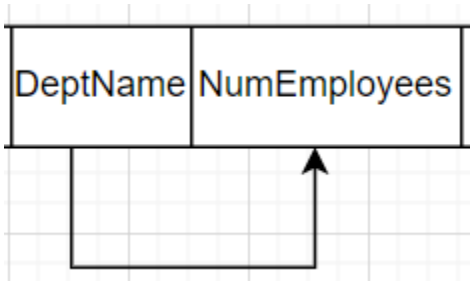
- Date, start time, end time, and breaks all depend upon both DoctorID and NurseID



The above table is in 1NF because it does not have composite attributes, multivalued attributes, nested relations, or attributes whose values for an individual tuple are non-atomic. It is in 2NF because it is in 1NF, and there are no partial dependencies. It is 3NF because it is in 2NF and it does not have any transitive dependencies of none-prime attributes on a candidate key through another non-prime attribute. It is in BCNF because it is in 3NF and attributes are only dependent on superkeys. We assume that each doctor-nurse pair has only one break in their combined shift.

Department

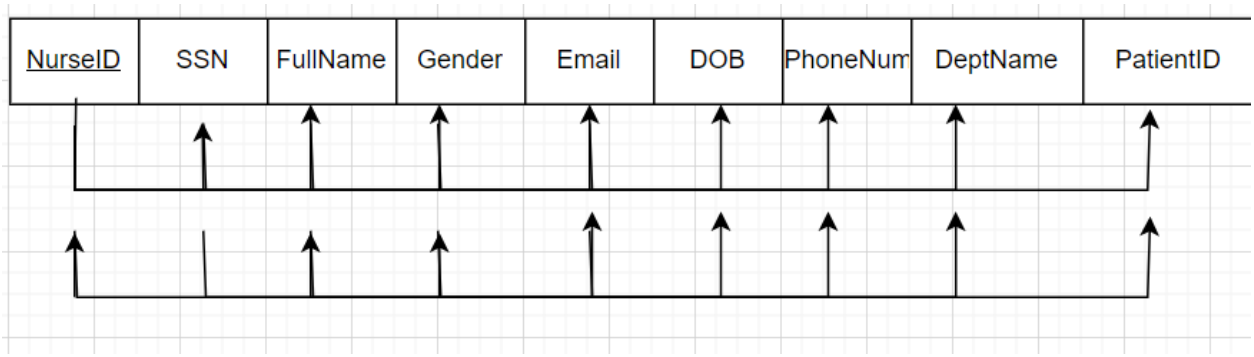
- NumEmployees depends on the DepartmentName



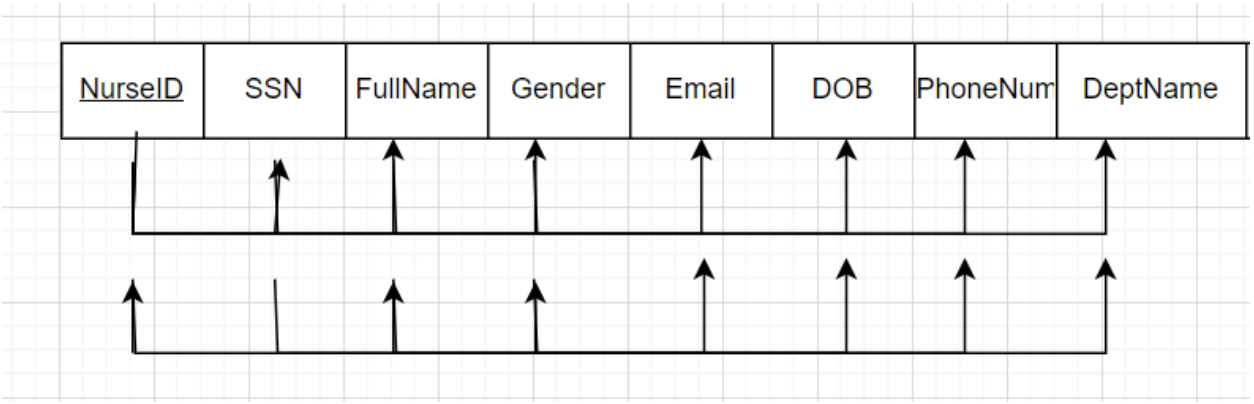
The above table is in 1NF because it does not have composite attributes, multivalued attributes, nested relations, or attributes whose values for an individual tuple are non-atomic. It is in 2NF because it is in 1NF, and there are no partial dependencies. It is 3NF because it is in 2NF and it does not have any transitive dependencies of none-prime attributes on a candidate key through another non-prime attribute. It is in BCNF because it is in 3NF and attributes are only dependent on superkeys.

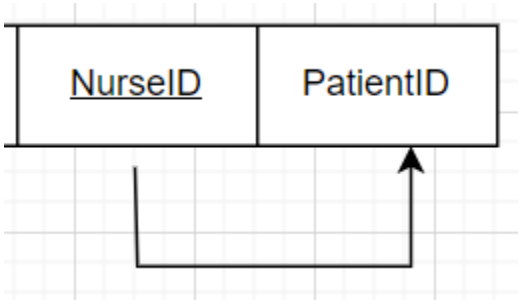
Nurses

- SSN, Fullname, Gender, Email, DOB, PhoneNum, DeptName, and PatientID all depend on the NurseID
- NurseID, Fullname, Gender, Email, DOB, PhoneNum, DeptName, and PatientID all depend on the SSN



The above table is not in 1NF because PatientID can have multiple values. The normalized tables are below

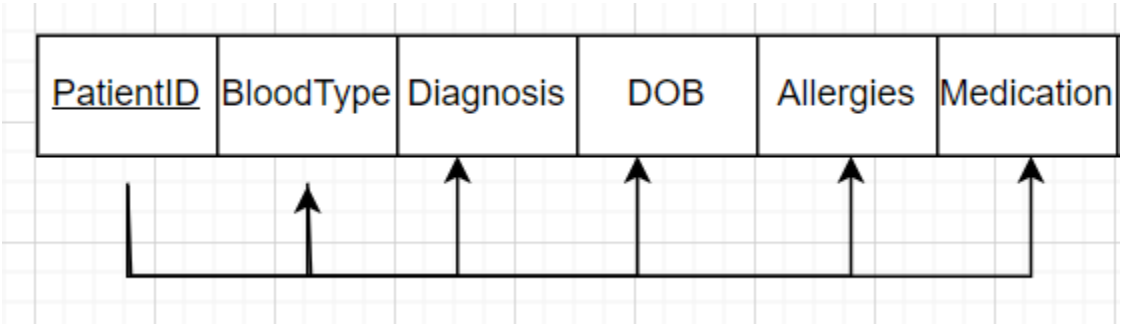




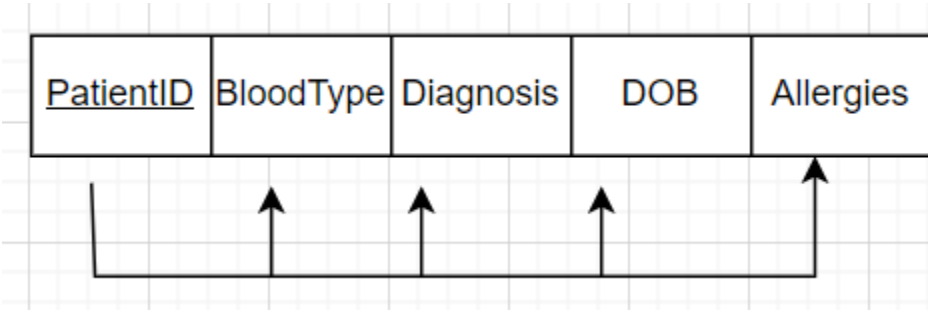
The above tables are in 1NF because they do not have composite attributes, multivalued attributes, nested relations, or attributes whose values for an individual tuple are non-atomic. They are in 2NF because they are in 1NF, and there are no partial dependencies. They are in 3NF because they are in 2NF and they do not have any transitive dependencies of none-prime attributes on a candidate key through another non-prime attribute. They are in BCNF because they are in 3NF and attributes are only dependent on superkeys. I assume a nurse has just one phone number and email address on file.

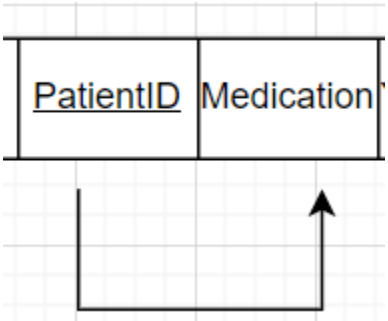
Medical Record

- BloodType, CurrentDiagnosis, Allergies, Medications, and PreviousIllnesses all depend on the PatientID

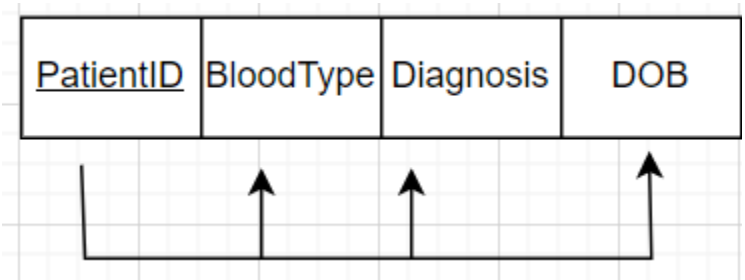


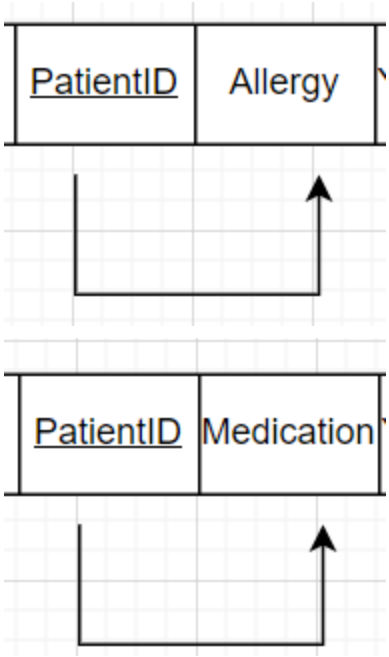
The above table is not in 1NF because Medication is multivalued. The normalized tables are below





The second table is in 1NF, but the first table is still not in 1NF because Allergies is multivalued. The normalized tables are below

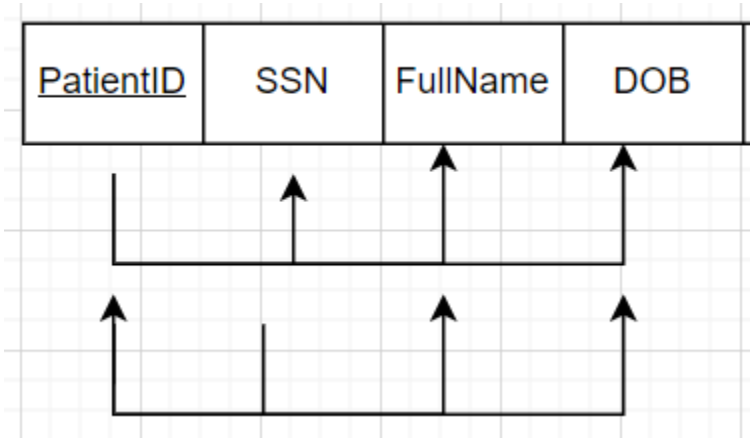




The above tables are in 1NF because they do not have composite attributes, multivalued attributes, nested relations, or attributes whose values for an individual tuple are non-atomic. They are in 2NF because they are in 1NF, and there are no partial dependencies. They are in 3NF because they are in 2NF and they do not have any transitive dependencies of none-prime attributes on a candidate key through another non-prime attribute. They are in BCNF because they are in 3NF and attributes are only dependent on superkeys. I assume a nurse has just one phone number and email address on file.

Patients

- SSN, FullName, and DOB all depend on the PatientID
- PatientID , FullName, and DOB all depend on the SSN

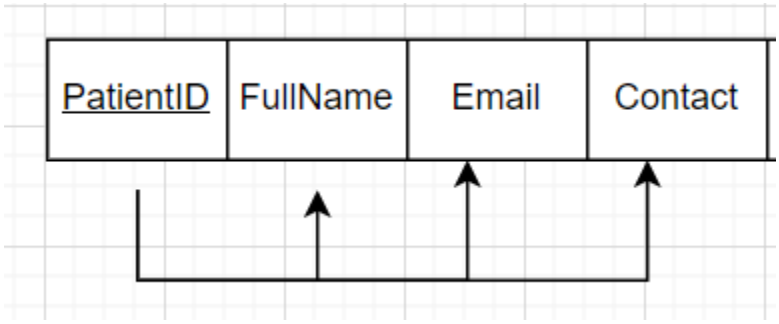


The above table is in 1NF because it does not have composite attributes, multivalued attributes, nested relations, or attributes whose values for an individual tuple are non-atomic. It is in 2NF because it is in 1NF, and there are no partial dependencies. It is 3NF because it is in 2NF and it does not have any transitive dependencies of none-prime attributes on a candidate key through

another non-prime attribute. It is in BCNF because it is in 3NF and attributes are only dependent on superkeys.

Emergency Contact

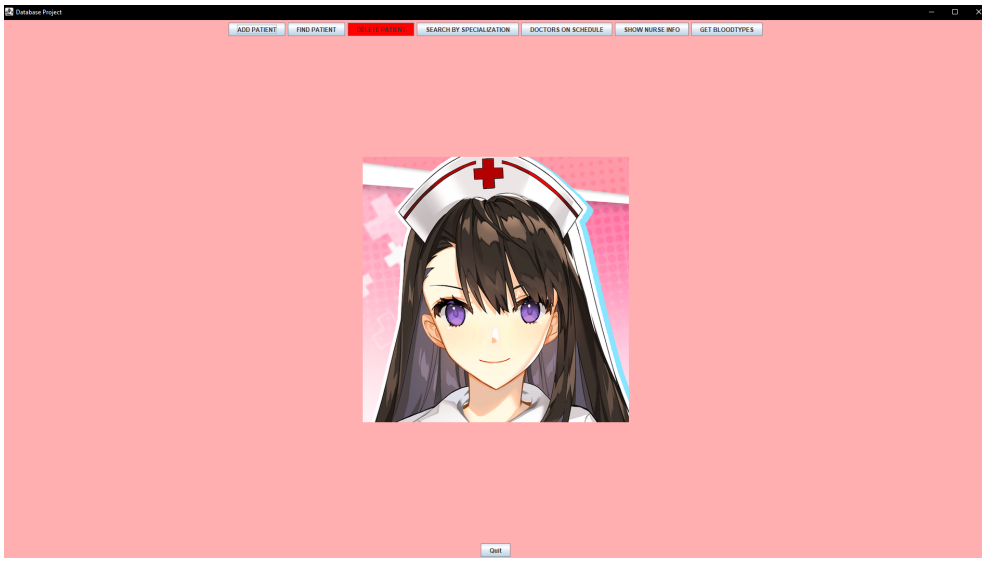
- FullName, Email, and Contact all depend on PatientID



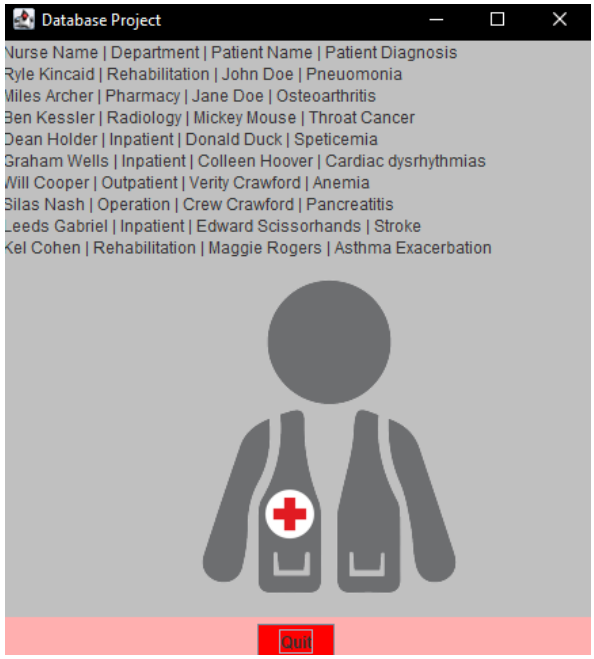
The above table is in 1NF because it does not have composite attributes, multivalued attributes, nested relations, or attributes whose values for an individual tuple are non-atomic. It is in 2NF because it is in 1NF, and there are no partial dependencies. It is 3NF because it is in 2NF and it does not have any transitive dependencies of non-prime attributes on a candidate key through another non-prime attribute. It is in BCNF because it is in 3NF and attributes are only dependent on superkeys.

The Database System

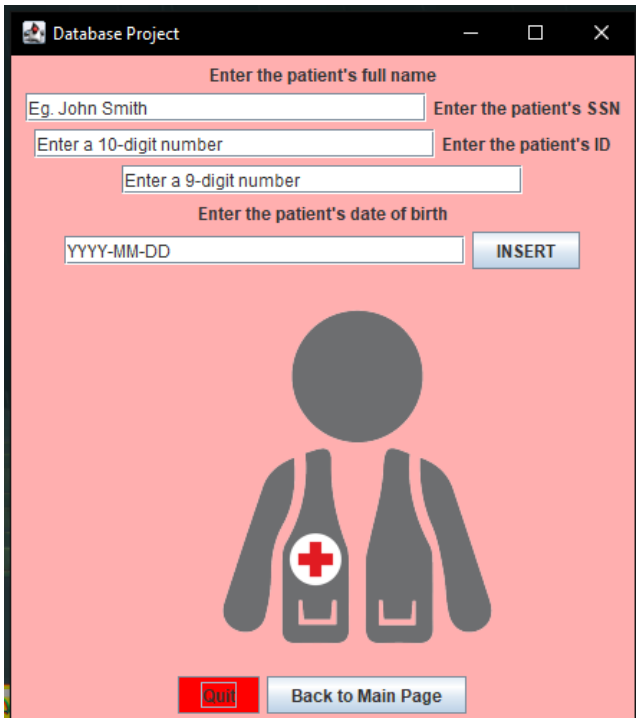
The database requires you to have a local instance of the database running on your computer using MySql Workbench. The local instance must be running on local host or else the program java program will not run. To run the program, you press run in the java program and a gui with multiple selections will appear.



Once this screen appears, youll be able to click any of the selections at the top. For example, if i were to click Show Nurse Info, you would get a selection of all nurses, their department, their patient, and their patient's illness as shown below.

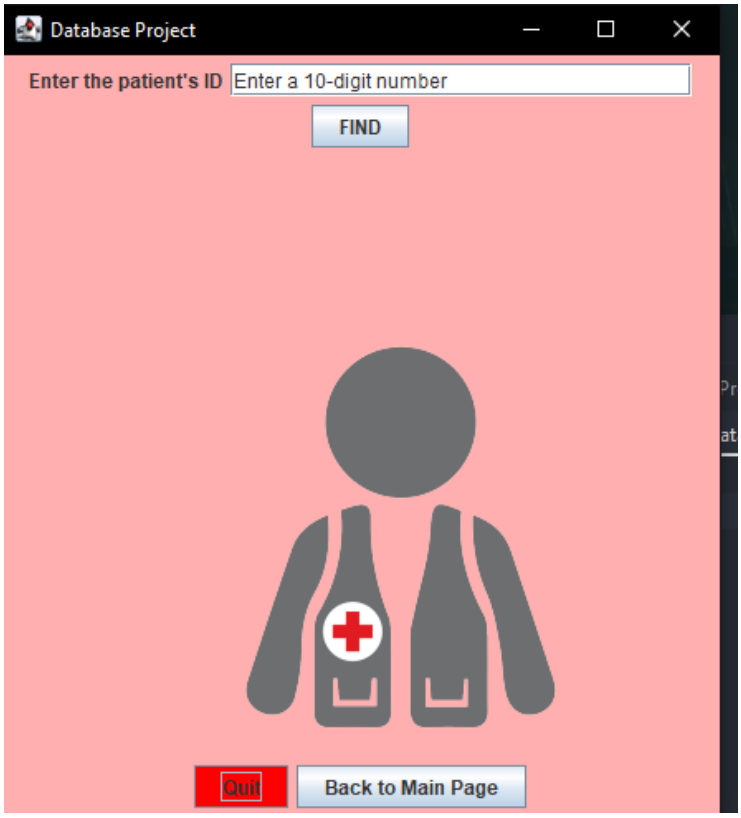


For another type, there are certain selections where you need to enter in fields to run the program. For example, add patient requires you to list a first name, social security number, patient ID, and date of birth to be able to add it to the database.

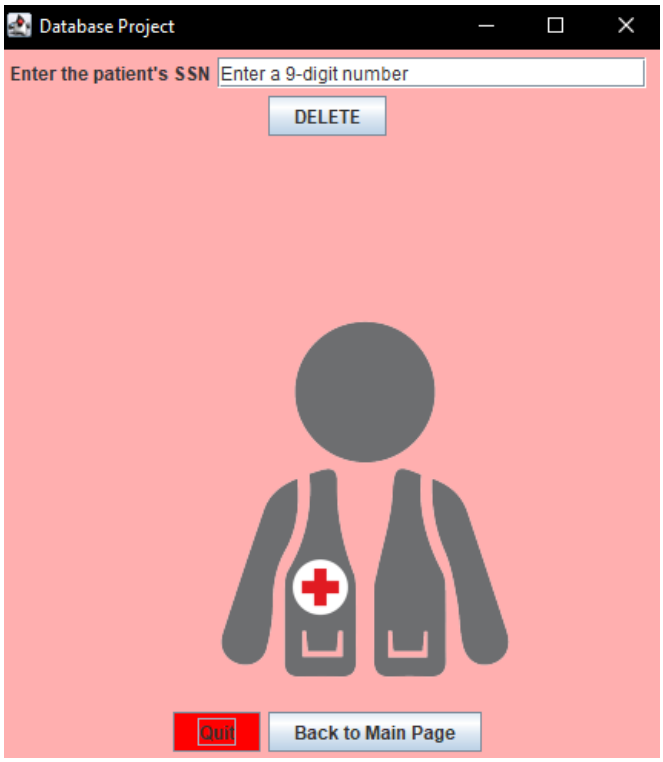


The other selections are very similar in that you either enter fields or you just press the button and it lists the output right away.

Find Patient



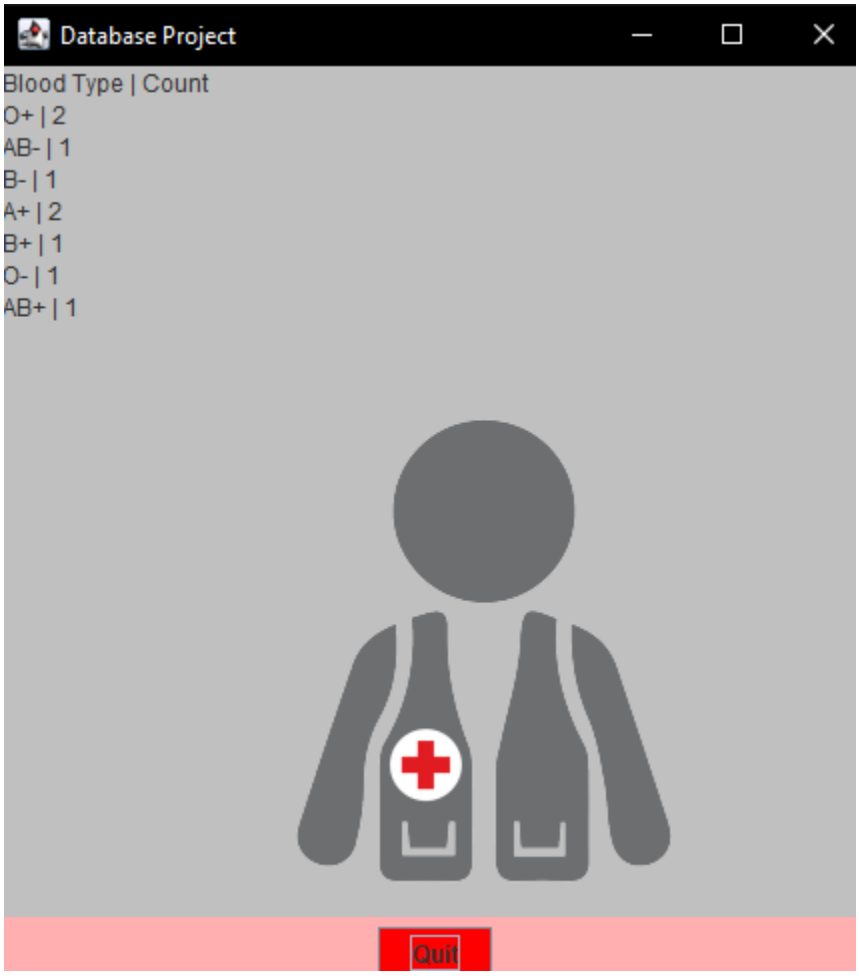
Delete patient



Find by specialization

Find doctors

Get blood types



Additional queries and views:

QUERIES:

- **Retrieve the DoctorID, FullName, and DeptName of all the Doctors with at least 5 years of experience**
  - SELECT Doctors.DoctorID, Doctors.FullName, Doctors.DeptName
    FROM Doctors
    WHERE Doctors.YrsExperience >= 5;

```
hospital=# SELECT Doctors.DoctorID, Doctors.FullName, Doctors.DeptName
hospital-# FROM Doctors
hospital-# WHERE Doctors.YrsExperience >= 5;
  doctorid  |    fullname     |    deptname
------------+-----------------+----------------
 1000100001 | Layken Cohen    | Outpatient
 1111111111 | Charlie Wynwood | Inpatient
 3333333333 | Fallon ONeill   | Inpatient
 6666666666 | Megan Andrews   | Operation
 8888888888 | Tate Collins    | Rehabilitation
(5 rows)
```

- **Retrieve the PatientID and FullName of all the patients that have PreviousIllnesses**
  - SELECT Patients.PatientID, Patients.FullName
    FROM (Patients JOIN MedicalRecord ON Patients.PatientID = MedicalRecord.PatientID)
    WHERE MedicalRecord.PreviousIllnesses != ‘N/A’;

```
hospital=# SELECT Patients.PatientID, Patients.FullName
FROM (Patients JOIN MedicalRecord ON Patients.PatientID =
MedicalRecord.PatientID)
WHERE MedicalRecord.PreviousIllnesses != 'N/A';
 patientid  |    fullname
------------+----------------
 0000000000 | John Doe
 0000000004 | Colleen Hoover
 0000000006 | Crew Crawford
(3 rows)
```

- **Retrieve the DoctorID and FullName of all the doctors who are male**
  - SELECT Doctors.DoctorID, Doctors.FullName
    FROM Doctors
    WHERE Doctors.Gender = ‘M’;

```
hospital=# SELECT Doctors.DoctorID, Doctors.FullName
FROM Doctors
WHERE Doctors.Gender = 'M';
  doctorid  |    fullname
------------+----------------
 4444444444 | Atlas Corrigan
(1 row)
```

- **Retrieve the DeptName and NumEmployees of all the Departments with at least 3 employees**
  - SELECT Department.DeptName, Department.NumEmployees
    FROM Department
    WHERE Department.NumEmployees >= 3;

```
hospital=# SELECT Department.DeptName, Department.NumEmployees
hospital-# FROM Department
hospital-# WHERE Department.NumEmployees >= 3;
    deptname    | numemployees
----------------+--------------
 Inpatient      |            5
 Rehabilitation |            4
(2 rows)
```

VIEWS:

- **Creates a view that shows the number of patients admitted in the hospital grouped by blood type**
  - CREATE VIEW BloodTypeCount
    AS SELECT MedicalRecord.BloodType, COUNT(*)
    FROM MedicalRecord
    GROUP BY MedicalRecord.BloodType;

```
hospital=# CREATE VIEW BloodTypeCount
hospital-# AS SELECT MedicalRecord.BloodType, COUNT(*)
hospital-# FROM MedicalRecord
hospital-# GROUP BY MedicalRecord.BloodType;
CREATE VIEW
hospital=# select * from bloodtypecount;
 bloodtype | count
-----------+-------
 O-        |     1
 AB-       |     1
 AB+       |     1
 A+        |     2
 B-        |     1
 B+        |     1
 O+        |     2
(7 rows)
```

- **Creates a view that shows the name, department, patient's name, and patient's diagnosis for every nurse in the hospital**
  - CREATE VIEW NursesPatients(Name, Dept, PatientName, PatientDiagnosis)
    AS SELECT N.FullName, N.DeptName, P.FullName, M.CurrentDiagnosis
    FROM ((Nurses as N JOIN Patients AS P ON N.PatientID = P.PatientID) JOIN MedicalRecord AS M ON P.PatientID = M.PatientID);

```
hospital=# CREATE VIEW NursesPatients(Name, Dept, PatientName, PatientDiagnosis)

hospital-# AS SELECT N.FullName, N.DeptName, P.FullName, M.CurrentDiagnosis
hospital-# FROM ((Nurses as N JOIN Patients AS P ON N.PatientID = P.PatientID) J
OIN MedicalRecord AS M ON P.PatientID = M.PatientID);
CREATE VIEW
hospital=# select * from nursespatients;
     name      |      dept      |     patientname     |   patientdiagnosis
---------------+----------------+---------------------+----------------------
 Ryle Kincaid  | Rehabilitation | John Doe            | Pneuomonia
 Miles Archer  | Pharmacy       | Jane Doe            | Osteoarthritis
 Ben Kessler   | Radiology      | Mickey Mouse        | Throat Cancer
 Dean Holder   | Inpatient      | Donald Duck         | Speticemia
 Graham Wells  | Inpatient      | Colleen Hoover      | Cardiac dysrhythmias
 Will Cooper   | Outpatient     | Verity Crawford     | Anemia
 Silas Nash    | Operation      | Crew Crawford       | Pancreatitis
 Leeds Gabriel | Inpatient      | Edward Scissorhands | Stroke
 Kel Cohen     | Rehabilitation | Maggie Rogers       | Asthma Exacerbation
(9 rows)
```

- **Creates a view that shows the ID, full name, department, and specialization of each doctor in the hospital**
    - CREATE VIEW SpecializationInfo
      AS SELECT Doctors.DoctorID, Doctors.FullName, Doctors.DeptName, Specialization.Specialization
      FROM Doctors, Specialization
      WHERE Doctors.DoctorID = Specialization.DoctorID;

```
hospital=# CREATE VIEW SpecializationInfo
hospital-# AS SELECT Doctors.DoctorID, Doctors.FullName, Doctors.DeptName, Speci
alization.Specialization
hospital-# FROM Doctors, Specialization
hospital-# WHERE Doctors.DoctorID = Specialization.DoctorID;
CREATE VIEW
hospital=# select * from specializationinfo;
  doctorid  |    fullname     |    deptname    | specialization
------------+-----------------+----------------+----------------
 1000100001 | Layken Cohen    | Outpatient     | Pulmonology
 1111111111 | Charlie Wynwood | Inpatient      | Rheumatology
 2222222222 | Lily Bloom      | Radiology      | Otolaryngology
 3333333333 | Fallon ONeill   | Inpatient      | Intensive Care
 4444444444 | Atlas Corrigan  | Rehabilitation | Cardiology
 5555555555 | Sky Davis       | Pharmacy       | Hematology
 6666666666 | Megan Andrews   | Operation      | Pancreatology
 7777777777 | Lowen Ashleigh  | Paramedical    | Vascular
 8888888888 | Tate Collins    | Rehabilitation | Immunology
(9 rows)
```

- **Creates a view that shows the emergency contacts' name of each patient in the hospital**
    - CREATE VIEW PatientContacts (PatientID, PatientName, EmergencyContact)
      AS SELECT Patients.PatientID, Patients.FullName, EmergencyContact.FullName
      FROM Patients, EmergencyContact

WHERE Patients.PatientID = EmergencyContact.PatientID;

```
hospital=# CREATE VIEW PatientContacts (PatientID, PatientName, EmergencyContact
)
hospital-# AS SELECT Patients.PatientID, Patients.FullName, EmergencyContact.Ful
lName
hospital-# FROM Patients, EmergencyContact
hospital-# WHERE Patients.PatientID = EmergencyContact.PatientID;
CREATE VIEW
hospital=# CREATE VIEW PatientContacts (PatientID, PatientName, EmergencyContact
)
hospital-# AS SELECT Patients.PatientID, Patients.FullName, EmergencyContact.Ful
lName
hospital-# FROM Patients, EmergencyContact
hospital-# WHERE Patients.PatientID = EmergencyContact.PatientID;
ERROR:  relation "patientcontacts" already exists
hospital=# select * from patientcontacts;
 patientid  |     patientname     | emergencycontact
------------+---------------------+------------------
 0000000000 | John Doe            | Jill Doe
 0000000001 | Jane Doe            | James Doe
 0000000002 | Mickey Mouse        | Michael Mouse
 0000000003 | Donald Duck         | Don Duck
 0000000004 | Colleen Hoover      | Collin Hoover
 0000000005 | Verity Crawford     | Harper Crawford
 0000000006 | Crew Crawford       | Chastin Crawford
 0000000007 | Edward Scissorhands | Ella Scissorhands
 0000000008 | Maggie Rogers       | Maddie Rogers
(9 rows)
```

- **Creates a view that shows the current diagnosis and allergies of the patient, from the medical record, with the patient's name and ID**
  - CREATE VIEW PatientRecord (PatientID, PatientName, CurrentDiagnosis, Allergies)
    AS SELECT Patients.PatientID, Patients.FullName, MedicalRecord.CurrentDiagnosis, MedicalRecord.Allergies
    FROM Patients, MedicalRecord
    WHERE Patients.PatientID = MedicalRecord.PatientID;

```
hospital=# CREATE VIEW PatientRecord (PatientID, PatientName, CurrentDiagnosis,
Allergies)
hospital-# AS SELECT Patients.PatientID, Patients.FullName, MedicalRecord.Curren
tDiagnosis, MedicalRecord.Allergies
hospital-# FROM Patients, MedicalRecord
hospital-# WHERE Patients.PatientID = MedicalRecord.PatientID;
CREATE VIEW
hospital=# select * from patientrecord;
 patientid  |     patientname     |   currentdiagnosis   |     allergies
------------+---------------------+----------------------+--------------------
 0000000000 | John Doe            | Pneuomonia           | Sulfonamides
 0000000001 | Jane Doe            | Osteoarthritis       | N/A
 0000000002 | Mickey Mouse        | Throat Cancer        | Aspirin, Ibuprofen
 0000000003 | Donald Duck         | Speticemia           | Latex
 0000000004 | Colleen Hoover      | Cardiac dysrhythmias | N/A
 0000000005 | Verity Crawford     | Anemia               | N/A
 0000000006 | Crew Crawford       | Pancreatitis         | N/A
 0000000007 | Edward Scissorhands | Stroke               | Penicillin
 0000000008 | Maggie Rogers       | Asthma Exacerbation  | Pollen, Mold
(9 rows)
```

User Application Interface

For the system interface, we created a graphical user interface that has several options for the type of operation the user wants to perform on the database. We assume that the user is a staff at the hospital, in which case the user is presented the option to: insert a patient, retrieve a patient's information, delete a patient, search for doctors of a certain specialization, display nurse information, find information about the blood types present/available at the hospital, and retrieve the doctors in schedule. These options are presented in the form of buttons on the "main screen," and after a user clicks on a button they are presented a different screen where they might be prompted for input. After potentially entering the input and clicking another button to submit the query, the corresponding operation will be performed on the backend as a query and the results will be returned to the user via another application screen.

The functions themselves are implemented using Prepared Statements to make all the queries safe and less prone to having SQL injection be read as input. Each function takes in a connection, and then any other data that is required to output the data. Each function will then convert the ResultSet data type into a string that can be returned and set to the GUI to be displayed as output.

Conclusions and Future Work

This project was a good introduction to full stack development because we were able to build a cohesive application that had a pleasant GUI and was fully functional with its operations on the database. We all worked together to map out and implement our project, and in the end we were able to create a formidable application.

In terms of improvement, we thankfully received feedback during our demo that illuminated some issues we had with our design and deepened our understanding of database applications. For example, some work we could do in the future is that we could use the auto increment function to assign IDs to the doctors, nurses, and patients to ensure uniqueness and to increase the ease of using the system. Another thing we can change in the future is that we can ensure that the primary key to each table is unique and not randomly entered by the user, in order to ensure that integrity constraints are upheld.

References

[1] Oracle, “A Visual Guide to Layout Managers.” Accessed Nov 30, 2022 [Online]. Available: https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

[2] R. Elmasri, S. Navathe, “*The Fundamentals of Database Systems*,” Hoboken, NJ, USA: Pearson, 2016.