PES UNIVERSITY

100 feet Ring Road, BSK 3rd Stage, Bengaluru 560085

Department of Computer Science and Engineering
Jan – May 2020

UE18CS252
Database Management Systems

Project Report

Hospital Management System

PES1201800229   Poorvi Sameep Kunja
4th Sem Sec. [I]   Roll No. 12

PROJECT SUMMARY

My project is modelled after a Hospital Management System. It contains multiple tables including Patients, Doctors, Departments, Room, Comments, Prescriptions and Appointments. One of my triggers calculates the date a medication runs out when the date of pick up of the medicine is updated and inserts an entry into a table called Runs_Out, referencing the Precription_Id. My secondary trigger calculates the discharge status of a patient by using the attributes of the Room table and writes the results in the Release_Status table. My queries are closely modelled on the real world. The first one lists all the doctors that a specific patient has consulted. The second query helps us identify the doctor that has attended to most patients. The third query tackles the real-life problem of limited availability of a specific drug at a particular time. On availability of the drug, it updates the Picked_Up_Date of all patients requesting that particular drug. The final query models the situation of medication delivery where it displays the names of the medications the employee must carry and also calculates the exact quantity required when he enters the area that he will be delivering to.

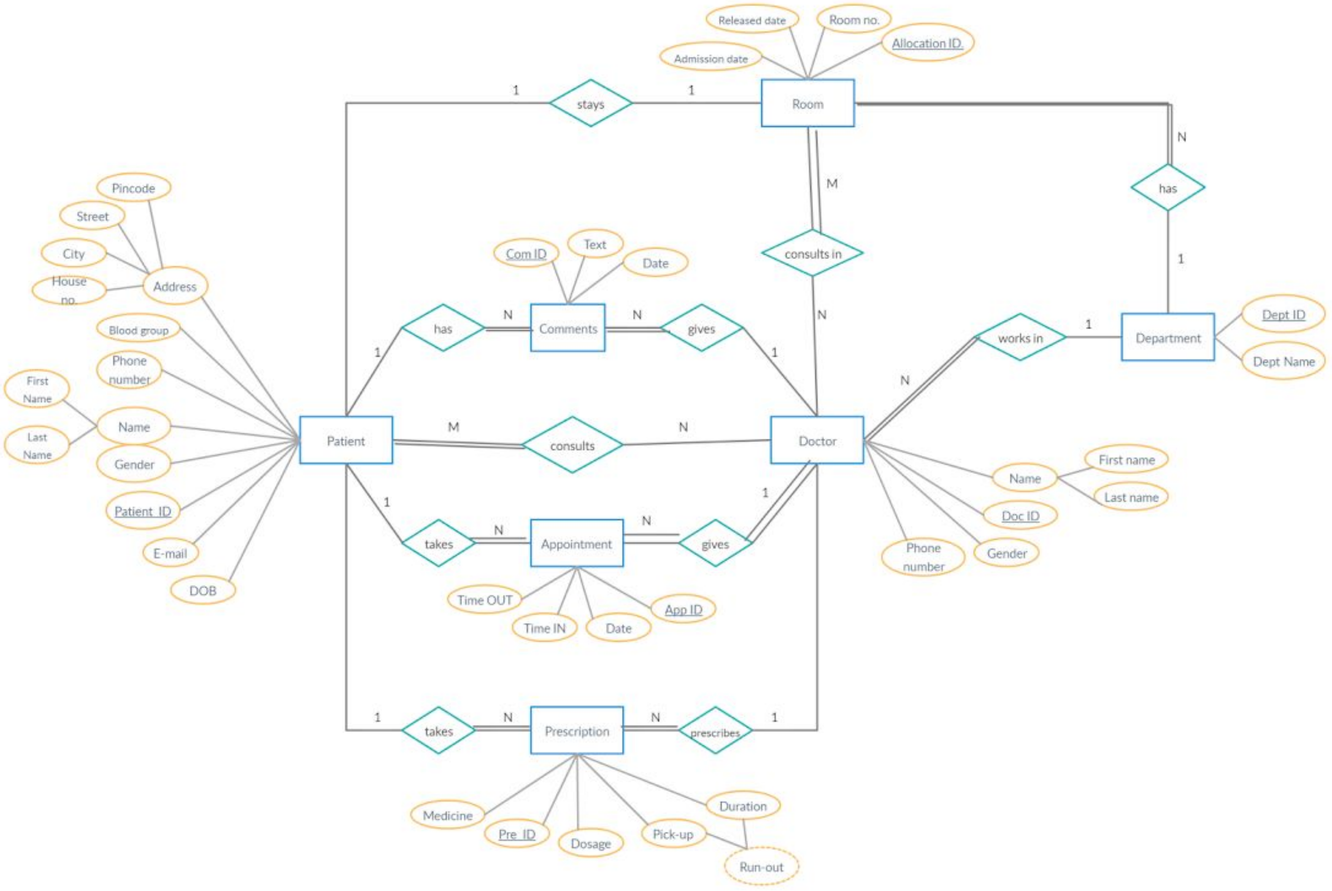
All in all, I believe that my model closely resembles the real world Hospital and can tackle most of the issues that they face.

Introduction

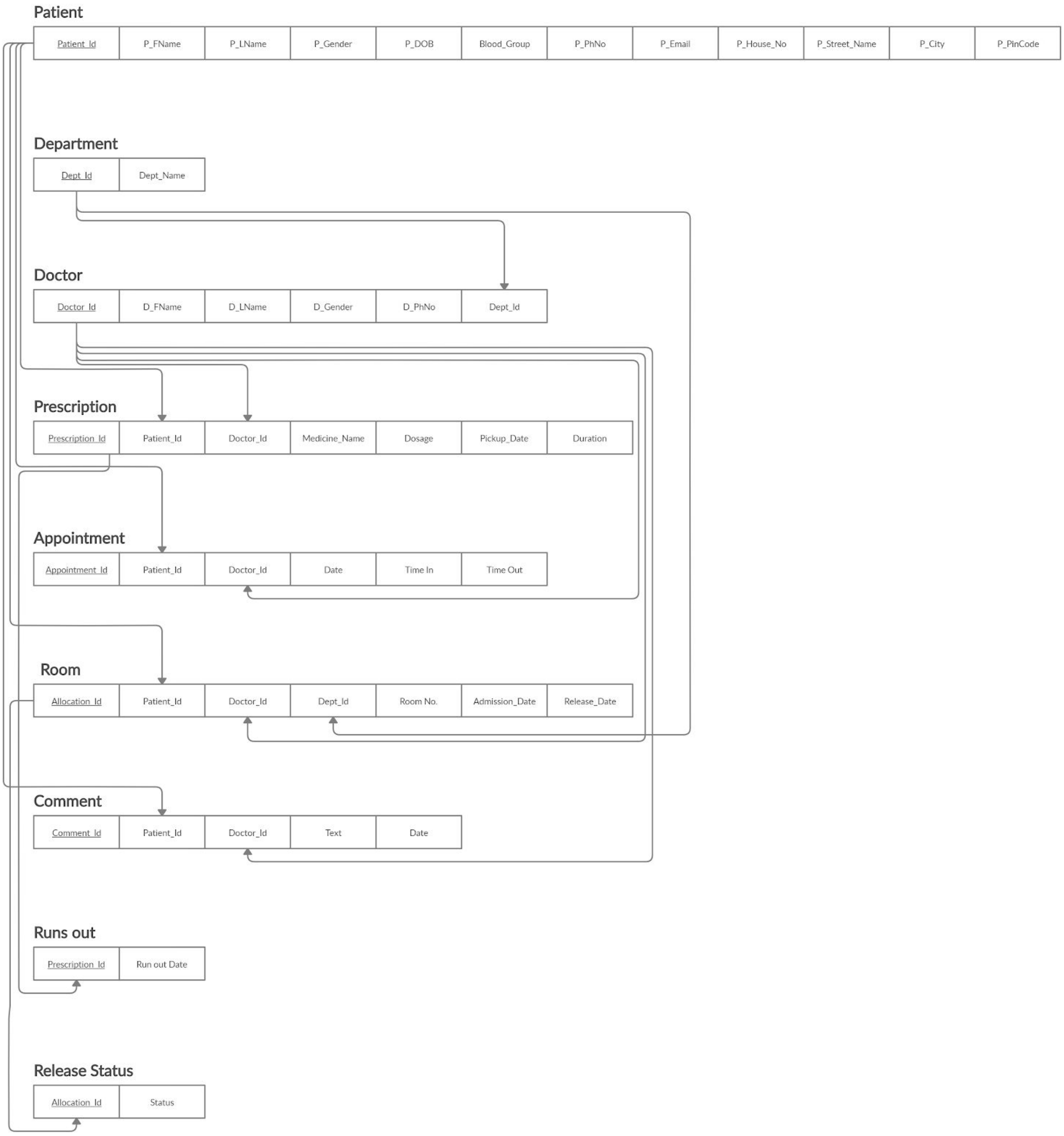
My project is modelled after a Hospital Management System. It accepts patient data (First name, Last name, gender, Blood group, Date of Birth, Email, Phone Number and four fields to accept their address). Each patient is uniquely identified by a Patient_id which is auto-incremented (use of serial data type). We have a similar table for accepting doctor data (First Name, Last Name, Phone number, Department they work under and gender) that differentiates doctors based on the id. We have a Department table that uniquely identifies Departments based on their id and has an additional attribute of a name. Whenever a doctor prescribes a medication to a patient, a new record is added to the Prescriptions table which identifies each record by a Prescription_id and also references Doctor_Id and Patient_Id. Further, it has the Medication Name, dosage and duration for which the medication must be taken. All the aforementioned attributes must have some value (NOT NULL). The final attribute of the Prescription table is the Picked_Up_Date which is updated when the patient picks up the medication from the pharmacy. Moving onto the next table of Appointments, it contains an Appointment_Id, references Doctor_Id and Patient_Id and also requires the Date and Time In and Time Out of the appointment slot. Whenever a patient is admitted to the hospital, a new record is added to the Room table (Room_id, Doctor_Id, Patient_Id, Department_Id). Additionally, it requires the date of Admission of the patient and o optionally, a discharge date which helps us determine the discharge status of the patient. The final attribute is the room number in which the patient is admitted. Typically, a doctor writes down comments during a patient appointment, which are stored in the comments table referencing the doctor's id and the patient's id and identified by its own comment_id. It also asks for the date of entry of comment and the comment text.

Data Model

<u>ER Diagram:</u>



Relational Schema:



Patient

Patient_Id	P_FName	P_LName	P_Gender	P_DOB	Blood_Group	P_PhNo	P_Email	P_House_No	P_Street_Name	P_City	P_PinCode

Department

Dept_Id	Dept_Name

Doctor

Doctor_Id	D_FName	D_LName	D_Gender	D_PhNo	Dept_Id

Prescription

Prescription_Id	Patient_Id	Doctor_Id	Medicine_Name	Dosage	Pickup_Date	Duration

Appointment

Appointment_Id	Patient_Id	Doctor_Id	Date	Time In	Time Out

Room

Allocation_Id	Patient_Id	Doctor_Id	Dept_Id	Room No.	Admission_Date	Release_Date

Comment

Comment_Id	Patient_Id	Doctor_Id	Text	Date

Runs out

Prescription_Id	Run out Date

Release Status

Allocation_Id	Status

FD and Normalization

F:{F1, F2, F3, F4, F5, F6, F7, F8, F9, F10}

F1: {Patient_Id} -> {P_FName, P_LName, P_Gender, P_DOB, Blood_Group, P_PhNo, P_Email, P_House_No, P_Street_Name, P_City, P_PinCode}
in Patient

F2: {Dep_Id} -> {Dept_Name}
in Department

F3: {Doctor_Id} -> {D_FName, D_LName, D_Gender, D_PhNo, Dept_Id}
in Doctor

F4: {Prescription_Id} -> {Patient_Id, Doctor_Id, Medicine_Name, Dosage, Pickup_Date, Duration}
in Prescription

F5: {Appointment_Id} -> {Patient_Id, Doctor_Id, Date, Time_In, Time_Out}
in Appointment

F6: {Allocation_Id} -> {Patient_Id, Doctor_Id, Dept_Id, Room_No, Admission_Date, Release_Date}
in Room

F7:{Comment_Id} -> {Patient_Id, Doctor_Id, Text, Date}
in Comment

F8: {P_PhNo} -> {Patient_Id, P_FName, P_LName, P_Gender, P_DOB, Blood_Group, P_Email, P_House_No, P_Street_Name, P_City, P_PinCode}
in Patient

F9: {Dept_Name} -> {Dep_Id}
in Department

F10: {D_PhNo} -> {Doctor_Id, D_FName, D_LName, D_Gender, Dept_Id}
in Doctor

F11: {Patient_Id, Doctor_Id} -> { Patient_Id, Doctor_Id } (Trivial FD)
in Consults

Candidate Keys:

1. Patient:

On applying attribute closure, we get:
$[\text{Patient\_Id}]^+$ = {P_FName, P_LName, P_Gender, P_DOB, Blood_Group, P_PhNo, P_Email, P_House_No, P_Street_Name, P_City, P_PinCode}
It can be seen that the closure for the attribute Patient_Id covers all attributes in the table. Hence, **Patient_Id** is a candidate key.

On applying attribute closure, we get:
$[\text{P\_PhNo}]^+$ = {Patient_Id, P_FName, P_LName, P_Gender, P_DOB, Blood_Group, P_Email, P_House_No, P_Street_Name, P_City, P_PinCode}
It can be seen that the closure for the attribute P_PhNo covers all attributes in the table. Hence, **P_PhNo** is a candidate key.

2. Department:

On applying attribute closure, we get:
$[\text{Dept\_Id}]^+$ = {Dept_Name}
It can be seen that the closure for the attribute Dept_Id covers all attributes in the table. Hence, **Dept_Id** is a candidate key.

On applying attribute closure, we get:
$[\text{Dept\_Name}]^+$ = {Dept_Id}
It can be seen that the closure for the attribute Dept_Name covers all attributes in the table. Hence, **Dept_Name** is a candidate key.

3. Doctor:

On applying attribute closure, we get:
$[\text{Doctor\_Id}]^+$ = {D_FName, D_LName, D_Gender, D_PhNo, Dept_Id}
It can be seen that the closure for the attribute Doctor_Id covers all attributes in the table. Hence, **Doctor_Id** is a candidate key.

On applying attribute closure, we get:
$[\text{D\_PhNo}]^+$ = {Doctor_Id, D_FName, D_LName, D_Gender, Dept_Id}
It can be seen that the closure for the attribute D_PhNo covers all attributes in the table. Hence, **D_PhNo** is a candidate key.

4. Prescription

On applying attribute closure, we get:
$[\text{Prescription\_Id}]^+$ = {Patient_Id, Doctor_Id, Medicine_Name, Dosage, Pickup_Date, Duration}
It can be seen that the closure for the attribute Prescription_Id covers all attributes in the table. Hence, **Prescription_Id** is a candidate key.

5. Appointment

On applying attribute closure, we get:

$[\text{Appointment\_Id}]^{+}$ = {Patient_Id, Doctor_Id, Date, Time_In, Time_Out}

It can be seen that the closure for the attribute Appointment_Id covers all attributes in the table. Hence, **Appointment_Id** is a candidate key.

6. Room

On applying attribute closure, we get:

$[\text{Allocation\_Id}]^{+}$ = {Patient_Id, Doctor_Id, Dept_Id, Room_No, Admission_Date, Release_Date}

It can be seen that the closure for the attribute Allocation_Id covers all attributes in the table. Hence, **Allocation_Id** is a candidate key.

7. Comment

On applying attribute closure, we get:

$[\text{Comment\_Id}]^{+}$ = {Patient_Id, Doctor_Id, Text, Date}

It can be seen that the closure for the attribute Comment_Id covers all attributes in the table. Hence, **Comment_Id** is a candidate key.

8. Consults

On applying attribute closure, we get:

$[\text{Patient\_Id, Doctor\_Id}]^{+}$ = {Patient_Id, Doctor_Id}

It can be seen that the closure for the attribute (Patient_Id, Doctor_Id) covers all attributes in the table. Hence, **(Patient_Id, Doctor_Id)** is a candidate key.

Normalization and Testing for lossless join property:

Normalization:

The currently obtained database structure is in 3NF as well as BCNF as there are no transitive dependencies and the attributes of the table are functionally dependent only on the key. The Third Normal Form (3NF) can be violated where there is a scope of transitive dependency i.e. a non-prime attribute determining another non-prime attribute.

For example: If we consider that the Doctor table has another column named Dept_Name, 3NF is violated because Dept_Id -> Dept_Name exists. In the doctor table, Dept_Id is a non-prime attribute and therefore leads to transitive dependency.

For example: If we consider that the Doctor table has another column named Dept_Name and take our candidate key as (Doctor_Id, Dept_Id), 2NF is violated because Dept_Id -> Dept_Name exists and leads to partial dependency.

Lossless Join Property:

A relation, R is said to be lossless, if, upon decomposition into two relations, R1 and R2, the natural join of R1 and R2 gives the relation R again. This implies that there has been no loss of data/no redundant rows. Thus upon checking we come to the conclusion that none of the tables that we have in the current scenario can be decomposed into two more tables such that its a lossless decomposition. This is because our tables are already in an appropriate normal form.

DDL

```
create table PATIENTS (
        Patient_Id SERIAL PRIMARY KEY,
        P_FName VARCHAR(50) NOT NULL,
        P_LName VARCHAR(50) NOT NULL,
        P_Gender VARCHAR(6) NOT NULL CHECK (P_Gender IN ('Male', 'Female', 'Other')),
        P_Email VARCHAR(50),
        P_DOB DATE NOT NULL,
        P_Blood_Group VARCHAR(3) NOT NULL CHECK (P_Blood_Group IN
('A+','A-','AB+','AB-','O+','O-','B+','B-')),
        P_PhNo VARCHAR(10) UNIQUE NOT NULL,
        P_House_No VARCHAR(50) NOT NULL,
        P_Street_Name VARCHAR(50) NOT NULL,
        P_City VARCHAR(50) NOT NULL,
        P_PinCode VARCHAR(6) NOT NULL CHECK (P_PinCode LIKE '______')
);

insert into PATIENTS (P_FName, P_LName, P_Gender, P_Email, P_DOB, P_Blood_Group,
P_PhNo, P_House_No, P_Street_Name, P_City, P_PinCode) values ('Yul', 'Rossi', 'Male',
'yrossi0@shareasale.com', '1961-08-26', 'B+', '9013074524', '55', 'Barby', 'Bengaluru', '560064');

insert into PATIENTS (P_FName, P_LName, P_Gender, P_Email, P_DOB, P_Blood_Group,
P_PhNo, P_House_No, P_Street_Name, P_City, P_PinCode) values ('Alyson', 'Arber', 'Female',
null, '2015-12-1', 'AB-', '8941625934', '70474', 'Hallows', 'Mumbai', '560022');

insert into PATIENTS (P_FName, P_LName, P_Gender, P_Email, P_DOB, P_Blood_Group,
P_PhNo, P_House_No, P_Street_Name, P_City, P_PinCode) values ('Binky', 'Kimbrey', 'Male',
'bkimbrey2@discovery.com', '1960-6-13', 'O+', '4925766032', '5949', 'Mariners Cove', 'Bengaluru',
'560052');

insert into PATIENTS (P_FName, P_LName, P_Gender, P_Email, P_DOB, P_Blood_Group,
P_PhNo, P_House_No, P_Street_Name, P_City, P_PinCode) values ('Rubina', 'Elfleet', 'Other',
'relfleet3@smh.com.au', '1959-9-23', 'O-', '3683515056', '4700', 'Longview', 'Bengaluru', '560032');

insert into PATIENTS (P_FName, P_LName, P_Gender, P_Email, P_DOB, P_Blood_Group,
P_PhNo, P_House_No, P_Street_Name, P_City, P_PinCode) values ('Mendel', 'Block', 'Male',
'mblock4@jiathis.com', '1940-5-28', 'A+', '8464535851', '7264', 'Graceland', 'Kolkata', '560035');

insert into PATIENTS (P_FName, P_LName, P_Gender, P_Email, P_DOB, P_Blood_Group,
P_PhNo, P_House_No, P_Street_Name, P_City, P_PinCode) values ('Jaime', 'Glaysher', 'Female',
'jglaysher5@amazonaws.com', '1964-12-30', 'B+', '3657051327', '7196', 'Bengaluru', 'Sadar Bazar',
'560010');

insert into PATIENTS (P_FName, P_LName, P_Gender, P_Email, P_DOB, P_Blood_Group,
P_PhNo, P_House_No, P_Street_Name, P_City, P_PinCode) values ('Phoebe', 'Eddison', 'Female',
null, '1998-8-27', 'B+', '9226048821', '8855', 'Lukken', 'Delhi', '560015');
```

```
insert into PATIENTS (P_FName, P_LName, P_Gender, P_Email, P_DOB, P_Blood_Group,
P_PhNo, P_House_No, P_Street_Name, P_City, P_PinCode) values ('Allegra', 'Huish', 'Other',
'ahuish7@whitehouse.gov', '1941-9-12', 'B-', '8235057523', '2', 'Tennessee', 'Mumbai', '560027');

insert into PATIENTS (P_FName, P_LName, P_Gender, P_Email, P_DOB, P_Blood_Group,
P_PhNo, P_House_No, P_Street_Name, P_City, P_PinCode) values ('Ashly', 'Spenton', 'Female',
'aspenton8@cocolog-nifty.com', '1988-1-8', 'B+', '3297972315', '31', 'Oneill', 'Bengaluru', '560035');

insert into PATIENTS (P_FName, P_LName, P_Gender, P_Email, P_DOB, P_Blood_Group,
P_PhNo, P_House_No, P_Street_Name, P_City, P_PinCode) values ('Georgia', 'Faier', 'Female',
'gfaier9@google.nl', '2006-11-25', 'O+', '7993496495', '05', 'Manitowish', 'Bengaluru', '560064');




CREATE TABLE DEPARTMENTS(
Department_Id SERIAL PRIMARY KEY,
Department_Name VARCHAR(50) NOT NULL UNIQUE
);


INSERT INTO DEPARTMENTS (Department_Name) VALUES('Pathology');

INSERT INTO DEPARTMENTS (Department_Name) VALUES ('General');

INSERT INTO DEPARTMENTS (Department_Name) VALUES ('Dentistry');

INSERT INTO DEPARTMENTS (Department_Name) VALUES ('Anesthesiology');

INSERT INTO DEPARTMENTS (Department_Name) VALUES ('Intensive Care Unit');

INSERT INTO DEPARTMENTS (Department_Name) VALUES ('Nutrition');

INSERT INTO DEPARTMENTS (Department_Name) VALUES ('Cancer Care');

INSERT INTO DEPARTMENTS (Department_Name) VALUES ('Dermatology');

INSERT INTO DEPARTMENTS (Department_Name) VALUES ('Emergency');

INSERT INTO DEPARTMENTS (Department_Name) VALUES ('Cardiology');




create table DOCTORS (
	Doctor_Id SERIAL PRIMARY KEY,
	D_FName VARCHAR(50) NOT NULL,
	D_LName VARCHAR(50) NOT NULL,
	D_Gender VARCHAR(6) NOT NULL CHECK (D_Gender IN ('Male', 'Female', 'Other')),
	D_PhNo VARCHAR(50) UNIQUE NOT NULL,
	Dept_Id INT REFERENCES DEPARTMENTS(Department_Id) on delete cascade NOT NULL
);

insert into DOCTORS (D_FName, D_LName, D_Gender, D_PhNo, Dept_Id) values ('Pippa', 'Binge', 'Female', '7681710428',5);

insert into DOCTORS (D_FName, D_LName, D_Gender, D_PhNo, Dept_Id) values ('Igor', 'Cranfield', 'Male', '3308024410',3);

insert into DOCTORS (D_FName, D_LName, D_Gender, D_PhNo, Dept_Id) values ('Artie', 'Squibbes', 'Male', '7897393835',8);

insert into DOCTORS (D_FName, D_LName, D_Gender, D_PhNo, Dept_Id) values ('Dell', 'Chancelier', 'Male', '3059278557',5);

insert into DOCTORS (D_FName, D_LName, D_Gender, D_PhNo, Dept_Id) values ('Rosemaria', 'Sheivels', 'Female', '7253755432',9);

insert into DOCTORS (D_FName, D_LName, D_Gender, D_PhNo, Dept_Id) values ('Carline', 'Sink', 'Female', '1625766744',9);

insert into DOCTORS (D_FName, D_LName, D_Gender, D_PhNo, Dept_Id) values ('Raoul', 'Banes', 'Male', '2592725910',1);

insert into DOCTORS (D_FName, D_LName, D_Gender, D_PhNo, Dept_Id) values ('Dud', 'Cavet', 'Male', '6405927189',2);

insert into DOCTORS (D_FName, D_LName, D_Gender, D_PhNo, Dept_Id) values ('Carolann', 'Lonsbrough', 'Female', '5689012112',7);

insert into DOCTORS (D_FName, D_LName, D_Gender, D_PhNo, Dept_Id) values ('Lou', 'Fendlen', 'Male', '8959653839',10);



create table PRESCRIPTIONS (
	Prescription_Id SERIAL PRIMARY KEY,
	Patient_Id INT REFERENCES PATIENTS(Patient_Id) on delete cascade NOT NULL,
	Doctor_Id INT REFERENCES DOCTORS(Doctor_Id) on delete cascade NOT NULL,
	MedName VARCHAR(50) NOT NULL,
	Dosage INT NOT NULL,
	Picked_Up_Date DATE,
	Duration INT NOT NULL

);
```

```
insert into PRESCRIPTIONS (Patient_Id,Doctor_Id,MedName,Dosage,Duration) values
(3,4,'Paracetamol',2,5);

insert into PRESCRIPTIONS (Patient_Id,Doctor_Id,MedName,Dosage,Duration) values
(7,4,'Brufen',1,6);

insert into PRESCRIPTIONS (Patient_Id,Doctor_Id,MedName,Dosage,Duration) values
(5,2,'Crocin',2,7);

insert into PRESCRIPTIONS (Patient_Id,Doctor_Id,MedName,Dosage,Duration) values
(3,5,'Disprin',1,10);

insert into PRESCRIPTIONS (Patient_Id,Doctor_Id,MedName,Dosage,Duration) values
(9,2,'Norflox',2,9);

insert into PRESCRIPTIONS (Patient_Id,Doctor_Id,MedName,Dosage,Duration) values
(6,7,'Avomin',3,8);

insert into PRESCRIPTIONS (Patient_Id,Doctor_Id,MedName,Dosage,Duration) values
(2,4,'Paracetamol',2,5);

insert into PRESCRIPTIONS (Patient_Id,Doctor_Id,MedName,Dosage,Duration) values
(5,4,'Paracetamol',2,6);

insert into PRESCRIPTIONS (Patient_Id,Doctor_Id,MedName,Dosage,Duration) values
(1,4,'Paracetamol',2,3);

insert into PRESCRIPTIONS (Patient_Id,Doctor_Id,MedName,Dosage,Duration) values
(8,4,'Paracetamol',2,2);



create table APPOINTMENTS (
        Appointment_Id SERIAL PRIMARY KEY,
        Patient_Id INT REFERENCES PATIENTS(Patient_Id) on delete cascade NOT NULL,
        Doctor_Id INT REFERENCES DOCTORS(Doctor_Id) on delete cascade NOT NULL,
        Date DATE NOT NULL,
        TimeIn TIME NOT NULL,
        TimeOut TIME NOT NULL

);


insert into APPOINTMENTS (Patient_Id,Doctor_Id,Date,TimeIn,TimeOut) values
(3,4,'2020-5-30','10:15:00','11:15:00');

insert into APPOINTMENTS (Patient_Id,Doctor_Id,Date,TimeIn,TimeOut) values
(2,6,'2020-5-27','11:15:00','12:15:00');
```

```
insert into APPOINTMENTS (Patient_Id,Doctor_Id,Date,TimeIn,TimeOut) values
(9,7,'2020-5-6','12:15:00','13:15:00');

insert into APPOINTMENTS (Patient_Id,Doctor_Id,Date,TimeIn,TimeOut) values
(7,9,'2020-5-8','13:15:00','14:15:00');

insert into APPOINTMENTS (Patient_Id,Doctor_Id,Date,TimeIn,TimeOut) values
(4,4,'2020-5-10','16:15:00','17:15:00');

insert into APPOINTMENTS (Patient_Id,Doctor_Id,Date,TimeIn,TimeOut) values
(2,1,'2020-5-18','17:15:00','18:15:00');

insert into APPOINTMENTS (Patient_Id,Doctor_Id,Date,TimeIn,TimeOut) values
(4,6,'2020-5-12','16:15:00','17:15:00');

insert into APPOINTMENTS (Patient_Id,Doctor_Id,Date,TimeIn,TimeOut) values
(4,7,'2020-5-11','16:15:00','17:15:00');

insert into APPOINTMENTS (Patient_Id,Doctor_Id,Date,TimeIn,TimeOut) values
(4,8,'2020-5-13','16:15:00','17:15:00');



create table ROOM (
        Allocation_Id SERIAL PRIMARY KEY,
        Patient_Id INT REFERENCES PATIENTS(Patient_Id) on delete cascade NOT NULL,
        Doctor_Id INT REFERENCES DOCTORS(Doctor_Id) on delete cascade NOT NULL,
        Dept_Id INT REFERENCES DEPARTMENTS(Department_Id) on delete cascade NOT
NULL,
        Admission_Date DATE NOT NULL,
        Release_Date DATE,
        Room_No INT NOT NULL

);

insert into ROOM (Patient_Id,Doctor_Id,Dept_Id,Admission_Date,Release_Date,Room_No) values
(2,1,3,'2020-5-18','2020-5-23',201);

insert into ROOM (Patient_Id,Doctor_Id,Dept_Id,Admission_Date,Room_No) values
(3,4,3,'2020-5-23',201);

insert into ROOM (Patient_Id,Doctor_Id,Dept_Id,Admission_Date,Room_No) values
(5,6,3,'2020-5-25',201);

insert into ROOM (Patient_Id,Doctor_Id,Dept_Id,Admission_Date,Release_Date,Room_No) values
(9,3,3,'2020-5-27',’2020-5-30’,201);



create table COMMENT(
        Comment_Id SERIAL PRIMARY KEY,
        Patient_Id INT REFERENCES PATIENTS(Patient_Id) on delete cascade NOT NULL,
        Doctor_Id INT REFERENCES DOCTORS(Doctor_Id) on delete cascade NOT NULL,
        Text VARCHAR(150) NOT NULL,
        Date DATE NOT NULL

);


insert into COMMENT (Patient_Id,Doctor_Id,Text,Date) values (9,2,'High Fever. Paracetamol
prescribed','2020-5-27');

insert into COMMENT (Patient_Id,Doctor_Id,Text,Date) values (4,3,'Root canal
required','2020-5-27');

insert into COMMENT (Patient_Id,Doctor_Id,Text,Date) values (10,8,'Prominent
Acne','2020-5-27');

insert into COMMENT (Patient_Id,Doctor_Id,Text,Date) values (1,1,'Alzheimers Disease
positive','2020-5-27');

insert into COMMENT (Patient_Id,Doctor_Id,Text,Date) values (5,6,'Vitamin A
Deficiency','2020-5-27');



create table RUNS_OUT(
        Prescription_Id INT REFERENCES PRESCRIPTIONS(Prescription_Id) on delete cascade
NOT NULL,
        Runs_Out_Date DATE NOT NULL
);



create table Release_Status(
        Allocation_Id INT REFERENCES ROOM(Allocation_Id) on delete cascade NOT NULL,
        Status VARCHAR(50) NOT NULL
);
```

Triggers

1. **Calculates and adds the date a particular prescription runs out (and prescription id) when the date of pick up of medicine is mentioned in the PRESCRIPTIONS table.**

```
CREATE OR REPLACE FUNCTION RU_Date()
RETURNS TRIGGER AS
$BODY$
begin
if NEW.Picked_Up_Date IS NOT NULL THEN
INSERT INTO RUNS_OUT(Prescription_Id,Runs_Out_Date)VALUES
(OLD.Prescription_Id, NEW.Picked_Up_Date + OLD.DURATION);
end if;
RETURN NEW;
end;
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER RU_Date
BEFORE UPDATE ON PRESCRIPTIONS
FOR EACH ROW
EXECUTE PROCEDURE RU_Date();
```

2. **Inserts the discharge status of a patient into the Release_State Table on insertion of values into the ROOM Table**

```
CREATE OR REPLACE FUNCTION R_Status()
RETURNS TRIGGER AS
$BODY$
begin
if NEW.Release_Date IS NOT NULL THEN
INSERT INTO Release_Status(Allocation_Id,Status)VALUES
(NEW.Allocation_Id,'Discharged');
else
INSERT INTO Release_Status(Allocation_Id,Status)VALUES (NEW.Allocation_Id,'Not
Discharged');
end if;
RETURN NEW;
end;
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER R_Status
AFTER INSERT ON ROOM
FOR EACH ROW
EXECUTE PROCEDURE R_Status();
```

SQL Queries

Find the names of the doctors that the Patient with id = 4 has consulted

```
SELECT D_FName, D_LName FROM DOCTORS, APPOINTMENTS where
(DOCTORS.Doctor_Id = APPOINTMENTS.Doctor_Id AND APPOINTMENTS.Patient_Id = 4);
```

Find the most in-demand doctor in the hospital

```
SELECT D_FName, D_LName FROM DOCTORS WHERE Doctor_Id = (SELECT Doctor_Id
FROM APPOINTMENTS GROUP BY Doctor_Id HAVING count(Doctor_Id) = ((SELECT
MAX(DCOUNT) FROM (SELECT COUNT(Doctor_Id) AS DCOUNT FROM APPOINTMENTS
Group By Doctor_Id ) AS MAXCOUNT )));
```

A specific medicine is finally in-stock. Update the pick-up date for every patient who has been prescribed that medicine.

```
UPDATE PRESCRIPTIONS SET Picked_Up_Date = now()::DATE WHERE (MedName =
'Paracetamol' AND Picked_Up_Date IS NULL);
```

You have to deliver medicines to a certain area. Find the medicines that you need to take.

```
SELECT MedName, (sum(Dosage)*sum(Duration)) AS Quantity FROM
PRESCRIPTIONS,PATIENTS WHERE PATIENTS.Patient_Id = PRESCRIPTIONS.Patient_Id
AND PATIENTS.P_City = 'Bengaluru' GROUP BY MedName;
```

Conclusion

In conclusion, my project closely resembles a real-life model of a Hospital. Moreover, the triggers have also been created in such a way so as to help the hospital in their future.

CAPABILITIES

For example, one of the queries identifies the doctor attending to most patients. This will help the hospital in hiring purposes (if the doctor's department is understaffed and hence workload has increased) or help improve the practice of other doctors (in case of high recommendations leading to rise in number of patients) among others. The third query tackles the real-life problem of limited availability of a specific drug at a particular time and the final query models the situation of medication delivery. Due to the way our database has been implemented, it also has pre-existing conditions/diseases stored (through comments) making it easier for doctors to understand the patients better.

LIMITATIONS

1. There may be a clash of timings in the Appointments table.
2. There may be overbooking of rooms as raising exceptions is beyond the current scope of this project.

FUTURE SCOPE

1. We can extend this project by adding an application functionality that will allow appointment booking.
2. Similar functionality can be developed into a medication delivery system.
3. Since we can easily track the medical trends, we can also continue to do so in a manner that will help identify similar symptoms and trends in other family members.

VOTE OF THANKS

I would like to thank Raghu Sir for giving me the opportunity to learn about and how to implement a project in PostgreSQL and for being extremely accommodating with my doubts. Additionally, I would like to thank the Department of Computer Science, PES for supporting me through this project.