HMS project deliverable 3

Team Members :

Ashish Kumar Gupta	180050013	180050013@iitb.ac.in
Jagdish Suthar	180050039	jsuthar@iitb.ac.in
Kaushal U	180050047	180050047@iitb.ac.in
Shivanshu Gour	180050099	180050099@iitb.ac.in

ORIGINAL SCHEMA:

In the original schema only these two relations were not normalised but other relations were already in **BCNF** So we are only showing these two relations before normalisation.

1. Appointment(

ID	int	
timestamp_booking	datetime	Not Null
date_appoint	date	Not Null
status	text	Not Null
Doctor_id	int	
Patient_id	int	
Slot_name	text	
Day	text	
From	time	
To	time	

Primary key(ID)
Foreign Key(Doctor_id) references Doctor on delete set Null
Foreign Key(Patient_id) references Patient on delete set Null
Foreign Key(Slot_name, Day, from) references Slot_Interval on delete set Null
)

Candidate keys (ID) (timestamp, date, doctor_id, patient_id, from)
FD's {ID -> R
Doctor_id->Slot_name (Doctor_id is neither superkey nor Slot_name is part of any candidate key)
Date-> Day (Date is neither superkey nor Day is part of any candidate key)}
So this Relation is neither in **3NF nor in BCNF**

2. Test_appointment (

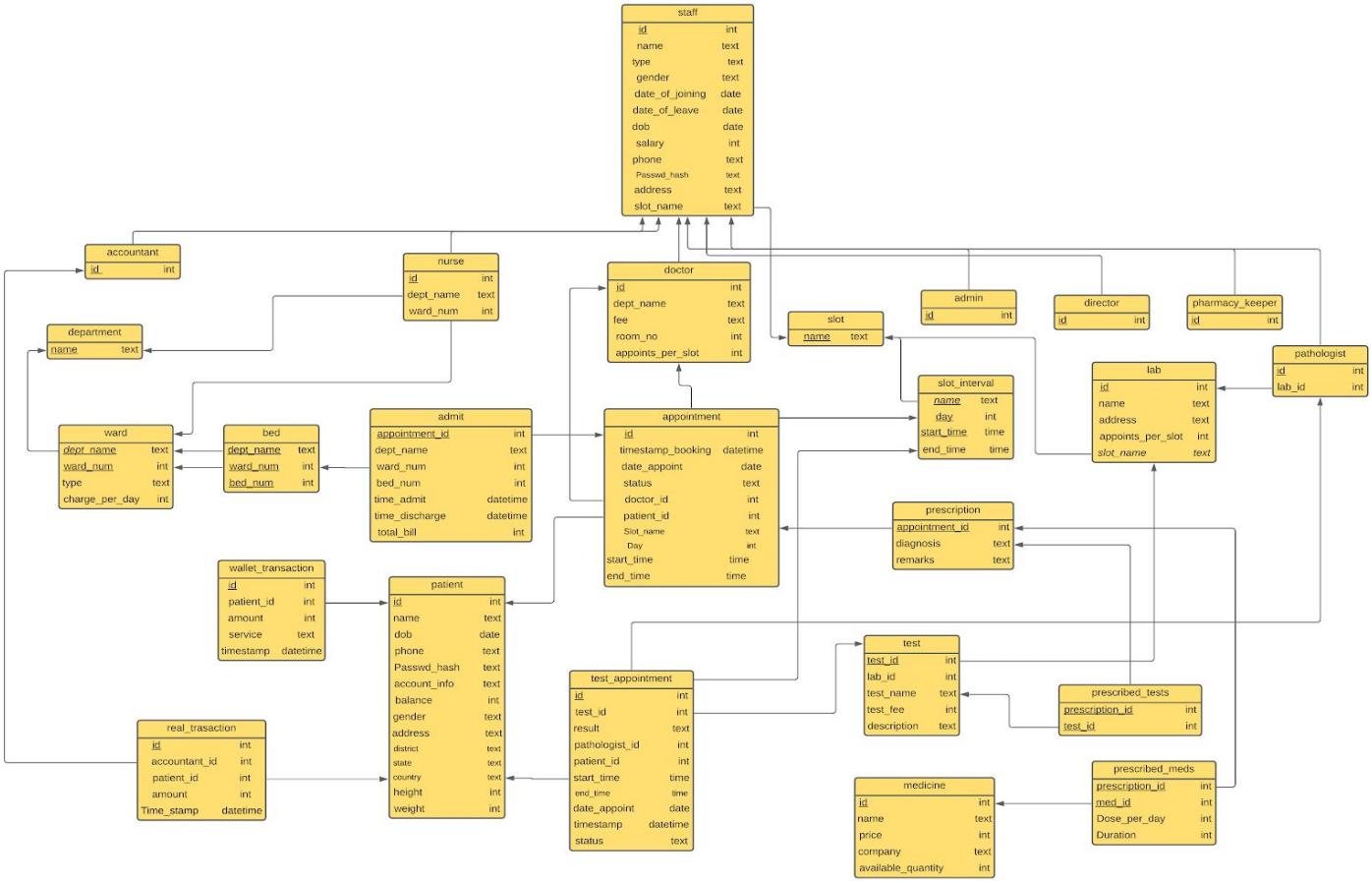
ID	int
Test_id	int
Result	text
Pathologist_id	int
Patient_id	int
Slot_name	text
Day	text
From	time
Date	date
Timestamp	datetime
Status	text

Primary key(ID)
Foreign Key(Test_id) references Test on delete set Null
Foreign Key(Patient_id) references Patient on delete set Null
Foreign Key(Slot_name, Day, from) references Slot_Interval on delete set Null
)

Candidate keys: (ID) (timestamp, date, test_id, from, patient_id)
ID -> R

Test_id->Lab_id and Lab_id->Slot_name => Test_id -> Slot_name
Where (Test_id is neither superkey nor Slot_name is part of any candidate key)
Date-> Day (Date is neither superkey nor Day is part of any candidate key)
So this Relation is neither in **3NF nor in BCNF**

NORMALISED SCHEMA DESIGN (with integrity constraints**):-**



1. Staff(

ID	int	
name	text	Not null
Type	text	Not Null
gender	text	(Male, Female, Other)
date_of_joining	date	Not null
date_of_leave	date	// (null => currently working)
dob	date	Not null
salary	int	Not null (>= 0)
phone	text	Not Null, **Unique**
Passwd_hash	text	Not Null
address	text	// null => address not specified

Slot_name text // null => Duty has not been assigned(i.e. for newly recruited staff)
Primary Key(ID)
Foreign Key(Slot_name) references Slot
)

ID -> R (ID is superkey)
Phone -> R (phone is superkey)
In **BCNF**

2. Accountant(

ID	int	

Primary key(ID)
Foreign Key(ID) references Staff on delete set Cascade
)

ID->R (ID is superkey)
In **BCNF**

3. Nurse(

ID	int	
Dept_name	text	Not null
Ward_Num	int	Not null

Primary key(ID)
Foreign Key(ID) references Staff on delete set Cascade
Foreign Key(Dept_name, Ward_Num) references Ward
)

ID -> R
In **BCNF**

4. Pathologist(

ID	int	
Lab_id	int	Not null

Primary key(ID)
Foreign Key(ID) references Staff on delete set Cascade
Foreign Key(Lab_id) references Lab
)

ID -> R (ID is superkey)
In **BCNF**

5. Pharmacy_keeper(

ID	int	

Primary key(ID)
Foreign Key(ID) references Staff on delete set Cascade
)

ID->R
IN **BCNF**

6. Director(

ID	int	

Primary key(ID)
Foreign Key(ID) references Staff on delete set Cascade
)

ID->R
IN **BCNF**

7. Admin(

ID	int	

Primary key(ID)
Foreign Key(ID) references Staff on delete set Cascade
)

ID->R
IN **BCNF**

8. Doctor(

ID int
Dept_name text Not null
Fee int Not null (>=0)
Room_no int // null => no specific room
Appoints_per_slot int Not null (>=0)
Primary key(ID)
Foreign key(ID) references Staff on delete set Cascade
Foreign key(Dept_name) references Department
)
ID->R
IN **BCNF**

9. Department (
Name text
Primary key (ID)
)
Name -> R(Id is superkey)
In **BCNF**

10. Ward (
Dept_name text
Ward_Num int
type text Not null (general, ICU)
Charge_per_day int Not null (>= 0)
Primary key (Dept_name, Ward_Num)
Foreign key(Dept_name) references Department
)
Dept_name, ward_num -> R ((dept_name,ward_num) is superkey)
In **BCNF**

11. Bed (
Dept_name text
Ward_Num int
Bed_num int
Primary key (Dept_name,Ward_Num,Bed_num)
Foreign Key(Dept_name, Ward_Num) references Ward
)
Dept_name, ward_num, bed_num -> R ((dept_name, ward_num, bed_num) is superkey)
In **BCNF**

12. Slot (
Name text
Primary key (Name)
)
name->name (trivial) in **BCNF**

13. Slot_Interval (
Name text
Day text
From time
To time Not null
Primary key(Name, Day, from)
Foreign key(Name) references Slot on delete Cascade
)
name, day, from -> R (name, day,from) is a superkey
In **BCNF**

// Day and slot_name is removed during normalization

14. Appointment(
ID int

timestamp_booking	datetime	Not Null
date_appoint	date	Not Null
status	text	(scheduled, complete, delayed, 'cancelled by doctor', cancelled)
Doctor_id	int	Not null
Patient_id	int	Not null
From	time	Not null
To	time	Not null

Primary key(ID)
Foreign Key(Doctor_id) references Doctor
Foreign Key(Patient_id) references Patient
)
Candidate keys (ID) (timestamp, date, doctor_id, patient_id, from)
ID -> R
In **BCNF**

15. Patient (

ID	int	
name	text	
dob	date	**Not Null**
Phone	text	**Not Null, Unique**
Passwd_hash	text	**Not Null**
Account_info	text	// null => not given
Balance	int	(>= 0)
Gender	text	(Male,Female, Other)
Address	text	// null => not specified
District	text	// null => not specified
State	text	// null => not specified
Country	text	// null => not specified
Height	int	(> 0) or null // null => not specified
Weight	int	(> 0) or null // null => not specified

Primary Key(ID)
)
ID -> R (ID is superkey)
Phone->R (phone is superkey)
In **BCNF**

16. Wallet_transaction (

Id	int		
Patient_id	int		// null => patient deleted
Amount	int	Not null	// +ve means credit, -ve debit
Service	text	Not Null	
Timestamp	datetime	Not Null	

Primary Key(id)
Foreign Key(Patient_id) references Patient on delete set Null
)
ID->R
IN **BCNF**

17. Real_transaction (

Id	int		
Accountand_id	int		// null => no accountant involved (direct bank-to-bank trns) // or accountant deleted
Patient_id	int		// null => patient deleted
Amount	int	Not null	// +ve => credit , -ve => debit
Time_stamp	datetime	Not Null	

Primary key(id)

Foreign Key(Patient_id) references Patient on delete set Null
Foreign Key(Accountant_id) references Accountant on delete set Null
)

ID->R
IN **BCNF**

18. Admit (

Appoinment_id	int		
Dept_name	text	Not null	
Ward_Num	int	Not null	
Bed_num	int	Not null	
Time_admit	datetime	Not null	
Time_discharge	datetime		// null => not discharged
Total_bill	int	(>=0)	// null => not discharged

Primary key(Appointment_id, Dept_id, Ward_Num, Bed_num)
Foreign Key(appointment_id) references Appointment on delete Cascade
Foreign Key(Dept_name, Ward_Num, Bed_Num) references Bed
Check (time_discharge is not null OR (dept_name is not null AND ward_num is not null AND Bed_num is not null)) // make sure that if patient is not discharged then bed should exist
)
Appointment_id-> R (Appointment_id is superkey)
In **BCNF**

19. Prescription(

Appointment_id	int		
Diagnosis	text	Not null	
Remarks	text		// can be null

Primary key(Appointment_id)
Foreig Key (appointment_id) references Appointment on delete Cascade
)
Appointment_id->R (Appointment_id is superkey)
IN **BCNF**

20. Lab(

ID	int	
Name	text	Not null
Address	text	Not null
Appoints_per_slot	int	Not null (>= 0)
Slot_name	text	Not null

Primary key(ID)
Foreign key (Slot_name) references Slot
)
ID->R (ID is superkey)
Name->Address,Appoints_per_slot, Slot_name (Name is superkey)
In **BCNF**

21. Test (

Test_id	int	
Lab_id	int	Not null
Test_name	text	Not null
Test_fee	int	Not null (>= 0)
Description	text	Not null

Primary key(Test_id)
Foreign Key(Lab_id) references Lab
)
Test_id -> R (Test_id is superkey)
IN **BCNF**

22. Medicine (

ID	int	
Name	text	Not null
Price	int	Not null (>=0)
Company	text	Not null
Available_quantity	int	Not null (>= 0)

Primary key(id)
)
ID->R (ID is superkey)
IN **BCNF**

23. Prescribed_Tests (

Prescription_id	int
Test_id	int

Primary key(Prescription_id,Test_id)
Foreign Key(Prescription_id) references Prescription on delete Cascade
Foreign Key(Test_id) references Test on delete set Null
)
Prescription_id, Test_id -> R ((Prescription_id, Test_id) is superkey)
IN **BCNF**

24. Prescribed_Meds (

Prescription_id	int		
Med_id	int		
Dose_per_day	int	(>=0)	// null => special medicine for emergency condition
Duration	int	(>=0)	// null => same as above

Primary key(Prescription_id,Med_id)
Foreign Key(Prescription_id) references Prescription on delete Cascade
Foreign Key(Med_id) references Med on delete set Null
)
Prescription_id,Med_id -> R
IN **BCNF**

// Day and slot_name is removed during normalization

25. Test_appointment (

ID	int		
Test_id	int	Not null	
Result	text		// null => report not yet ready
Pathologist_id	int		// null => test not done yet
Patient_id	int	Not null	
Start_time	time	Not null	
Date	date	Not null	
Timestamp	datetime	Not null	
Status	text	(scheduled, sample_taken, complete, delayed, cancelled, cancelled by pathologist)	

Primary key(ID)
Foreign Key(Test_id) references Test
Foreign Key(Patient_id) references Patient
)
ID -> R
IN **BCNF**

Denormalization for performance:-

These two schemas are again denormalized for performance reasons.

1. Appointment(

ID	int		
timestamp_booking	datetime	Not Null	
date_appoint	date	Not Null	
status	text	(scheduled, complete, delayed, 'cancelled by doctor', cancelled)	
Doctor_id	int	Not null	
Patient_id	int	Not null	
Slot_name	text	Not null	// added
Day	text	Not null	// added
From	time	Not null	
To	time	Not null	

Primary key(ID)
Foreign Key(Doctor_id) references Doctor
Foreign Key(Patient_id) references Patient
Foreign Key(Slot_name, Day, from) references Slot_Interval
)

2. Test_appointment (

ID	int		
Test_id	int	Not null	
Result	text		// null => report not yet ready
Pathologist_id	int		// null => test not done yet
Patient_id	int	Not null	
Slot_name	text	Not null	// added
Day	text	Not null	// added
Start_time	time	Not null	
Date	date	Not null	
Timestamp	datetime	Not null	
Status	text	(scheduled, sample_taken, complete, delayed, cancelled, cancelled by pathologist)	

Primary key(ID)
Foreign Key(Test_id) references Test
Foreign Key(Patient_id) references Patient
Foreign Key(Pathologist_id) references Pathologist
Foreign Key(Slot_name, Day, from) references Slot_Interval
)

Analytics & Queries

- Appointments/Test_appontments booking per day/week/month per doctor/test/combindly

```
select date_part('year',date_appoint) as year,
date_part('month',date_appoint) as month, count(*) as Num_of_appointments
from appointment
group by date_part('year',date_appoint),date_part('month',date_appoint);
```

```
select date_part('year',date_appoint) as year,
date_part('week',date_appoint) as week, count(*) as Num_of_appointments
from appointment
group by date_part('year',date_appoint),date_part('week',date_appoint);


with R as
   (select doctor_id, count(*) as Num_of_appointments
       from appointment
       group by doctor_id)
   select id, name,Num_of_appointments from
   R join staff on (R.doctor_id = staff.id);


select date_part('year',date_appoint) as year,
date_part('month',date_appoint) as month, count(*) as Num_of_appointments
from test_appointment
group by date_part('year',date_appoint),date_part('month',date_appoint);

select date_part('year',date_appoint) as year,
date_part('week',date_appoint) as week, count(*) as Num_of_appointments
from test_appointment
group by date_part('year',date_appoint),date_part('week',date_appoint);


with R as
   (select test_id, count(*) as Num_of_appointments
       from test_appointment
       group by test_id)
   select test.test_id, test_name,Num_of_appointments from
   R join test on (R.test_id = test.test_id);
```

- No_of_patients for each disease (per month , per year) grouping by age, gender, location

```
select P.diagnosis as disease, count(patient_id) as Num_of_patients
from appointment as A join prescription as P on (A.id = P.appointment_id)
group by diagnosis;

select date_part('year', A.date_appoint) as Year, P.diagnosis as disease,
count(A.patient_id) as Num_of_patients
from appointment as A join prescription as P on (A.id = P.appointment_id)
group by date_part('year', A.date_appoint),P.diagnosis;

select date_part('month', A.date_appoint) as Month, P.diagnosis as disease,
count(A.patient_id) as Num_of_patients
from appointment as A join prescription as P on (A.id = P.appointment_id)
group by date_part('month', A.date_appoint),P.diagnosis;

select P.diagnosis as disease, PT.gender, count(A.patient_id) as
Num_of_patients
from (appointment as A join prescription as P on (A.id = P.appointment_id))
join patient as PT on (PT.id = A.patient_id )
group by P.diagnosis, PT.gender;



select P.diagnosis as disease,
   case
   when (CURRENT_DATE- PT.dob) / 365.25 > 50 then '51 & over'
```

```
   when (CURRENT_DATE- PT.dob) / 365.25 > 19 then '20 - 30'
   when (CURRENT_DATE- PT.dob) / 365.25 > 30 then '31 - 50'
   else 'under 20'
 end as age_group
,count(A.patient_id) as Num_of_patients
from (appointment as A join prescription as P on (A.id = P.appointment_id))
join patient as PT on (PT.id = A.patient_id )
group by P.diagnosis,  age_group;
```

- Trending diseases in every season (which disease has most no of cases in particular season)

```
select date_part('month', A.date_appoint) as Month, P.diagnosis as disease,
count(A.patient_id) as Num_of_patients
from appointment as A join prescription as P on (A.id = P.appointment_id)
group by date_part('month', A.date_appoint),P.diagnosis
order by Num_of_patients DESC;
```

- No of Admitted patient per ward per department

```
select dept_name, ward_num, Count(appointment_id) as Num_of_patients
from admit
group by dept_name, ward_num;
```

- #Transactions and amount_transaction per day/month/year and for appointment booking with doctor, for lab_test, at pharmacy_store

```
select date_part('year',  timestamp_) as Year, count(*) as
Num_of_transaction
from real_transaction
group by date_part('year',  timestamp_);

select date_part('year',  timestamp_) as Year, date_part('month',
timestamp_) as Month, count(*) as Num_of_transaction
from real_transaction
group by date_part('year',  timestamp_),date_part('month',  timestamp_);


select date_part('year',  timestamp_) as Year, sum(amount) as NET_Cash_flow
from real_transaction
group by date_part('year',  timestamp_);

select date_part('year',  timestamp_) as Year, date_part('month',
timestamp_) as Month, sum(amount) as NET_Cash_flow
from real_transaction
group by date_part('year',  timestamp_),date_part('month',  timestamp_);
```

- Which medicines are having more demands(quantity sold per month/year)

```
select M.id , count(*) as Quantity_sold
from prescribed_meds as PM join medicine as M on(PM.med_id = M.id)
group by M.id;
```

Triggers:

- Triggers are added to check if any scheduled **appointment** and **test_appointment** is delayed, will be triggered before any insert/update/delete on respective tables.

```
-- ================== Triggers ====================
-- trigger to check for any delayed appointment
-- before any INSERT/UPDATE/DELETE
CREATE OR replace FUNCTION check_appoint_delayed()
  RETURNS TRIGGER
  LANGUAGE PLPGSQL
AS $$
BEGIN
  -- trigger logic
   update appointment set status = 'delayed'
   where status = 'scheduled'
   and (date_appoint < current_date) OR (date_appoint = current_date and
end_time < current_time);
   return new;
END;
$$;

create trigger check_delayed AFTER insert OR update OR delete
on appointment
for each statement
when (pg_trigger_depth() = 0)
execute procedure check_appoint_delayed();

-- trigger to check for any delayed test_appointment
-- before any INSERT/UPDATE/DELETE
CREATE OR replace FUNCTION check_test_appoint_delayed()
  RETURNS TRIGGER
  LANGUAGE PLPGSQL
AS $$

BEGIN
  -- trigger logic
   update test_appointment set status = 'delayed'
   where status = 'scheduled'
   and (date_appoint < current_date) OR (date_appoint = current_date and
end_time < current_time);
   return new;
END;
$$;

create trigger check_test_delayed AFTER insert OR update OR delete
on test_appointment
for each statement
when (pg_trigger_depth() = 0)
execute procedure check_test_appoint_delayed();
```

Transactions & SQL Queries:-

1) Doctor:-

a) Login into system as a staff (Doctor)

i) Select * from staff where phone=? and passwd_hash=?

b) Can see,cancel and mark complete each appointments

// to see all appointments

i) Select * from appointment where Doctor_id = doctor_id;
Select * from appointment where Doctor_id = doctor_id and date =?;

// appointment related to a particular patient

ii) Select * from appointment where Doctor_id = doctor_id and Patient_id = patient_id;

// cancelling appointment

iii) BEGIN TRANSACTION;
Update appointment
Set status = “cancelled by doctor”
Where id = appoint_id;

update patient set balance = balance + fee where id = p_id;

Insert into wallet_transaction (patient_id, amount, service)
values (p_id, fee, 'refund due to cancellation by doctor');
END TRANSACTION;

iv)

// marking appointment as complete

v) Update appointment set status = ‘complete’
Where id = appoint_id;

c) Write,modify and see prescription

i) Insert into prescription values(?,?,?)

//adding various medicine for that prescription

ii) Insert into Prescribed_meds(pres_id,med_id)

//adding various medicine for that prescription

iii) Insert into Prescribed_tests(pres_id,test_id)

iv) UPDATE prescription SET diagnosis = changed_diag, remarks = updated_remarks;

v) UPDATE Prescribed_meds SET med_id = new_med_id, Dose_per_day = new_dose,
Durations = new_duration;

vi) UPDATE Prescribed_tests SET test_id = new_test_id;

// see appointment details with prescription and meds and tests

vii) With R as (
Select * from appointment as A join prescription as P using(appointment_id)
Where patient_id = p_id;
)
SELECT * from (R as P JOIN Prescribed_Meds as PM on(P.appointment_id =
PM.prescription_id)) JOIN Prescribed_Tests as PT on (P.appointment_id =
PT.prescription_id)
ORDER BY date DESC

d) Can See test reports

```
SELECT * FROM test_appointments where patient_id = p_id;
```

e) Get Patients info

```
SELECT * FROM patients where patient_id = p_id;
```

2) Accountant

a) Add money to patient's wallet

i)
```
BEGIN TRANSACTION;
Insert into real_transaction(accountant_id, patient_id, amount) values (a_id, p_id,  amount);
Update patient set balance = balance + amount where id = p_id;
COMMIT TRANSACTION;
```

b) give back money from wallet

i)
```
BEGIN TRANSACTION;
Insert into real_transaction(accountant_id, patient_id, amount) values (a_id, p_id, -amount);
Update patient set balance = balance - amount where id = p_id;
COMMIT TRANSACTION;
```

3) Pathologist

a) Access prescription and previous reports of patient

```
// to get prescription details
```

i)
```
SELECT *
FROM appointment as A JOIN prescription as P ON (A.id = P.appointment_id)
WHERE patient_id = ?
// to get previous reports
```

ii)
```
SELECT *
From test_appointmet where  patient_id = p_id;
```

b) Take sample

i)
```
Update test_appointment
Set status = “sample_taken”
Where id = appoint_id;
```

c) Add report to patient's account

i)
```
Update test_appointment
Set result = “path_to_file”, status = “complete”
Where id = appoint_id;
```

d) Cancel appointment

i)
```
BEGIN TRANSACTION;
Update test_appointment
Set status = “cancelled by pathologist”
Where id = appoint_id;

update patient set balance = balance + fee where id = p_id;
```

Insert into wallet_transaction (patient_id, amount, service)
values (p_id, fee, 'refund due to cancellation by pathologist');
END TRANSACTION;

4) Pharmacy_keeper

a) Updating the medicine entity for the new supply
   i) UPDATE medicine SET available_quantity = ?,
   Name = ?, price = ?, company=?
   WHERE id = ?

   ii) INSERT INTO medicine values(?,?,?,?,?)

b) Checking availability of the medicine before purchasing
   i) SELECT * FROM medicine
   WHERE id = ? // name = ? ;

c) Make payment and deduct money from patient's wallet
   i) BEGIN TRANSACTION;
   update patient set balance = balance - amount where id = p_id;

   Insert into wallet_transaction (patient_id, amount, service)
   values (p_id, - amount, 'payment made at pharmacy');
   END TRANSACTION;

5) Director

a) Add a new staff member
   i) INSERT INTO staff values(?,?,?,?,?,?,?,?,?,?,?);
   //then insert the staff id into another table like doctor,admin etc
   ii) INSERT INTO admin values(?);
   iii) INSERT INTO pharmacy_keeper values(?);
   iv) INSERT INTO pathologist values(?);
   v) INSERT INTO doctor values(?,?,?,?,?);
   vi) INSERT INTO nurse values(?,?,?);
   vii) INSERT INTO accountant values(?);

b) Remove a staff member
   i) UPDATE staff SET relieving_date = ? WHERE id = ?
c) Can see statistics and analytics
   i) //various analytics queries

6) Admin

a) Can see statistics and analytics
   ii) //various analytics queries

7) Patient

a) Login to dashboard

i) Select * from patient where phone = ? and passwd_hash = ?;

b) Get appointment and medical tests history

i) Select * from appointment a inner join doctor d on(a.doctor_id = d.id)
where patient_id = ?
Order by date_appoint;

ii) Select * from test_appointment a inner join pathologist p on(a.pathologist_id = p.id)
Inner join test t using(test_id)
where patient_id = ?
Order by date_appoint;

c) A patient can book an appointment with a **doctor** by paying money from their wallet and can choose a slot from available ones.

// **Booking can be done at most 1 week earlier**

i) Select * from staff s inner join doctor d using(id);

// get available slot_intervals GIVEN (id)

ii)
```
with max_appoints as(
   select appoints_per_slot from doctor where id = ?),
   d_slot_name as(
      select slot_name from staff where id = ?),
   slots(slot_name, day, start_time, end_time) as(
      select *
      from slot_interval
      where name = d_slot_name),
   slot_count(slot_name, day, start_time, count) as(
      select slot_name, day, start_time , count(*)
      from appointment
      where date_appoint >= current_date and time_start > current_time
      and doctor_id = ?
      group by slot_name, day, start_time)

select slot_name, day, start_time, end_time, coalesce(count, 0),
      (case when (day = extract(dow from current_date) and start_time =< current_time)
            then current_date + 7
         when (day = extract(dow from current_date) and start_time > current_time)
            then current_date
         else current_date + MOD(day - extract(dow from current_date) + 7, 7)
      end) as "date"
from slots left outer join slot_count using(slot_name, day, start_time)
where (count < max_appoints OR count is null)
```

// transaction to book appointment

iii)
```
BEGIN TRANSACTION;
with max_appoints as(
   select appoints_per_slot from doctor where id = ?),
   d_slot_name as(
      select slot_name from staff where id = ?),
```

```
slots(slot_name, day, start_time, end_time) as(
   select *
   from slot_interval
   where name = d_slot_name and day = ? and start_time = ? ),
slot_count(slot_name, day, start_time, count) as(
   select slot_name, day, start_time , count(*)
   from appointment
   where date_appoint >= current_date and time_start > current_time
      and doctor_id = ? and day = ? and start_time = ?
   group by slot_name, day, start_time),
cur_appoints as(
   select coalesce(count, 0)
   from slots left outer join slot_count using(slot_name, day, start_time))
```

// this function will add transaction and update balance **definition is in DDL.sql** file

```
select book_appoint (max_appoints, cur_appoints, date(?), time(?), p_id(?), d_id(?),
fee(?));
COMMIT TRANSACTION;
```

d) A patient can book an appointment for a **Laboratory Test prescribed by a doctor** by paying money from their wallet and can choose a slot from available ones.
   i) Almost same queries as above for appointment with doctor with just change of relation name in the query

e) Add money to the wallet from bank
   i)
```
BEGIN TRANSACTION;
insert into real_transaction(accountant_id, patient_id, amount) values (null, ?, ?);
update patient set balance = balance + ? where id = ?;
COMMIT TRANSACTION;
```

f) Withdraw money from the wallet
   i)
```
BEGIN TRANSACTION;pat
insert into real_transaction(accountant_id, patient_id, amount) values (null, ?, ?);
update patient set balance = balance - ? where id = ?;
COMMIT TRANSACTION;
```

Indexes:-

```
// there are many queries to fetch appointments by doctor_id so an index on doctor_id will make it faster
        Create index doct_appointment on appointment(doctor_id)
// there are many queries to fetch appointments by patient_id so an index on patient_id will make it faster
        Create index patient_appointment on appointment(patient_id)
// there are many queries to fetch appointments by within a range of date so an index on date will make it faster
        Create index doct_appointment_date on appointment(date_appoint)\
// there are triggers and queries which filter appointments by their status so an index on status will make them faster
        Create index st_appointment on appointment(status)

// there are many queries on test_appointments to filter by patient_id and pathologist_id so an index on patient_id
and pathologist_id will make it faster
```

```
Create index patient_test_appointment on test_appointment(patient_id)
Create index patient_test_appointment on test_appointment(pathologist_id)
// there are many queries to fetch test_appointments by within a range of date so an index on date will make it faster
Create index patient_test_appointment on test_appointment(date_appoint)
// there are triggers and queries which filter appointments by their status so an index on status will make them faster
Create index st_test_appointment on test_appointment(status)
```

Complete Screen Design:-

USER :- DOCTOR

Use Case :- Can see,cancel and mark complete each appointments

Your Upcoming Appointments

Patient Id	Name	Date	Time	
4	Sandra Jensen	2021-02-17	3:00:00	see
5	Drew Howell	2021-02-18	3:00:00	see
6	Wing Powers	2021-02-18	16:00:00	see

Your Previous Appointments

Patient Id	Name	Date	Time	
1	Laurel Fischer	2021-02-11	3:00:00	see
2	Chaim Rollins	2021-02-13	3:00:00	see
3	Merritt Talley	2021-02-15	16:00:00	see

USER :- DOCTOR

Use Case :- 1) Can see,cancel and mark complete each appointments

2) Write,modify and see prescription

Appointment Details

Patient ID: 4

Name: Sandra Jensen Patient info

Date: 2021-02-17

Time: 3:00:00

Status: Scheduled

Diagnosis: N/A edit

Prescribed Medicine:N/A edit
Prescribed Tests: N/A edit
Remarks: N/A edit

See Reports

Cancel Appointment

Mark as Complete

On clicking edit for Diagnosis etc. a text box appears

Diagnosis...

Done

USER :- DOCTOR

Use Case :- Can See test reports

Reports:-

Patient ID: 4
Name: Sandra Jensen

CT Scan

MRI

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

USER :- DOCTOR

Use Case :- Can See Patient Details

Patient Details

Patient ID: 4
Name: Sandra Jensen
DOB: 12/02/1995
Age: 25
Gender : F
Height: 164 cm
Weight: 56 kg

USER :-Patient

Use Case :- *A patient can book an appointment with a **doctor** by paying money from their wallet and can choose a slot from available ones.*

Book An Appointment With a Doctor

Doctor id

slot_id

Date

Book

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

USER :-Patient

Use Case :- *A patient can book an appointment with a **doctor** by paying money from their wallet and can choose a slot from available ones.*

Payment

To Pay : xxxxxx
Wallet Balance : xxxxxx
Service: Doctor Appointment

Book Appointment

USER :-Patient

Use Case :- *A patient can book an appointment for a **Laboratory Test prescribed by a doctor** by paying money from their wallet and can choose a slot from available ones.*

Book A Test Appointment

prescription id

test_id

slot_id

Date

Book

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

USER :-Patient

Use Case :- *A patient can book an appointment for a **Laboratory Test prescribed by a doctor** by paying money from their wallet and can choose a slot from available ones.*

To Pay : xxxxxx
Wallet Balance : xxxxxx
Service: Test Appointment

Book Appointment

USER :-Patient

Use Case :- *Add money to the wallet from bank*

Add Money to Wallet

Amount

Add

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

USER :-Patient

Use Case :- *Withdraw money from the wallet*

Withdraw Money from Wallet

Amount

Withdraw

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

USER :-Patient

Use Case :- *Verify a payment*

Verify Your Payment

Invoice

Medicine	Quantity	Price
xxxxxxx	2	250
xxxxxxx	3	300
Total		550

Your Balance: xxxxxxxx

Verify

USER :-Patient

Patients Dashboard

Book An Appointment With A Doctor

Book An Appointment For A Test

Add Money To Your Wallet

Withdraw Money From Your Wallet

Verify Payment

USER :Accountant

Use Case :- Add money to patient's wallet

Add Money To Patient's Wallet

Patient_id

Amount

Add

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

USER :Accountant

Use Case :- give back money from wallet

Removing Money from Wallet

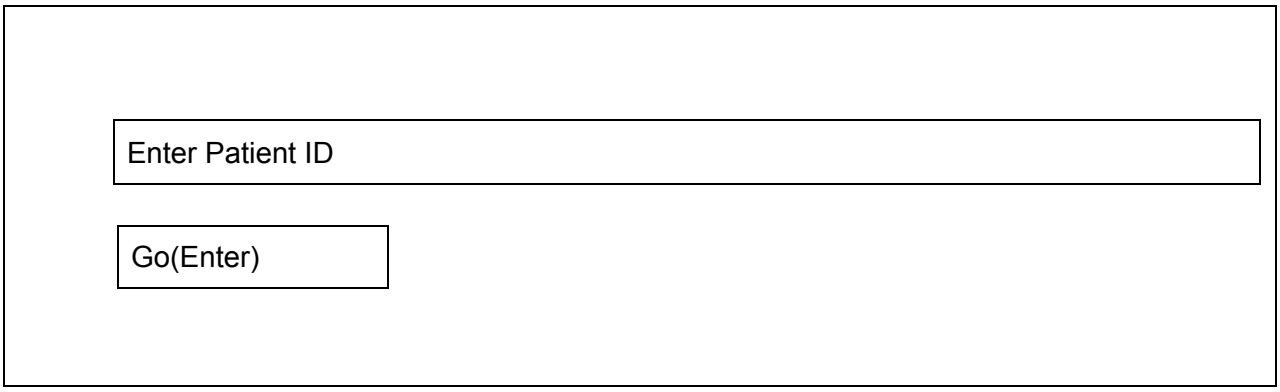
Patient_id

Amount

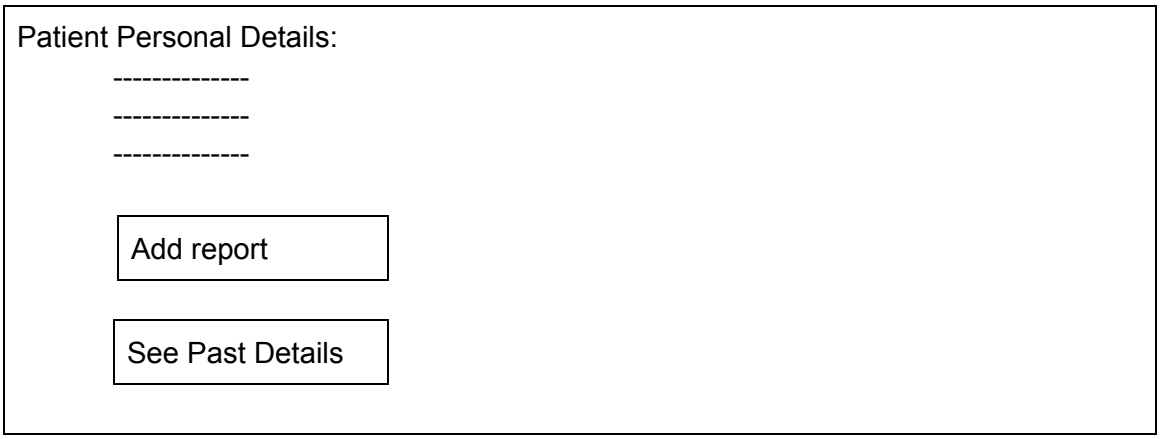
Remove

USER : Pathologist

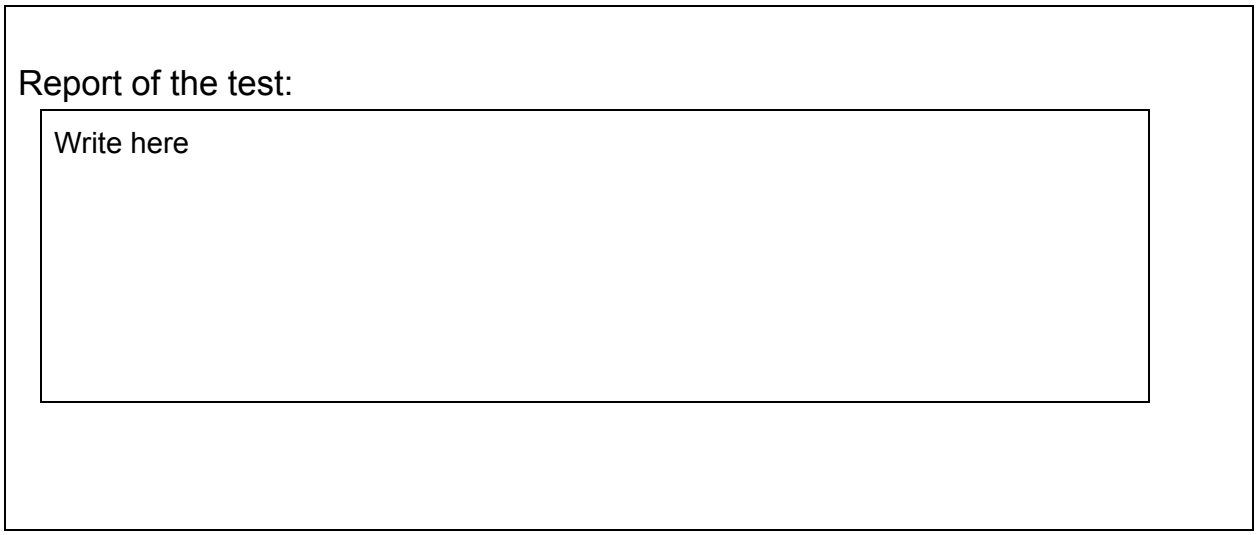
Use Case: Add report to patient's account



On Click:



Use Case: Add report



Use Case: Access prescription and previous reports of patient

Prescriptions Dropdown:

Presc1
Presc2
Presc3

Reports Dropdown:

Rep1
Rep2
Rep3

USER : Pharmacy keeper

Use case: Updating the medicine entity for the new supply

Update Medicine Stock

Sell Medicine

Update Medicine Stock:

Medicine name

Medicine Company

Added Stock quantity

Add

Use Case: initiate a payment for medicine

Patient ID

Medicine Name

Medicine Company

Quantity

Add (more)

Medicine	Quantity	Price
xxxxxxx	2	250
xxxxxxx	3	300
Total		550

Proceed to Payment

Director page:

Add Staff member

Fire Staff member

View Statistics and Analysis

UseCase: Add Staff member

Enter Staff Details

Name:	----------
Designation:	----------
Gender:	----------
Date of JOining:	----------
Phone:	------
Salary:	----------

Etc details

Confirm to add

UseCase: Fire Staff member

Enter staff ID

Confirm to fire

Admin page:

UseCase: Can see statistics and analytics

View and Analyse the System

Nurse page:

//there will be use cases in future if we implement the additional features

Data Generation and loading:-

Parameters of Data Generation:

- Staff : The date_of_joining is kept after the year 2016. As of now, the date_of_leaving is kept NULL. DOB is kept between 1975 and 1995. Salary is kept constant for all records.
- appoints_per_slot in each relation is kept 2
- Ward: For each department, 2 wards are present. In each ward, 10 beds are present
- Appointment : The Appointment dates are kept in the year 2021.
- Slot : There are 2 slots: general slot, night slot. General Slot: Between 6AM to 10 PM. Night Slot: Between 10 PM to 6 AM.
- Slot interval: Each slot interval is kept 1 hour long.
- Lab: The slot for all 5 labs is kept general for now.
- Pathologist: Each lab has 1 pathologist.
- Prescription: The remarks attribute states the prescribed tests for now.
- Test : Description attribute contains only the test name. Test fee is kept either 500 or 1000 for a record.
- Test Appointment: Appointment date is kept in 2021. Results of CT scan, MRI, X-Ray, ECG are mentioned as image link, result of all other tests is currently NULL.
- Patient: Used the generation tool and set the account balance: 100 to 10000 and other attributes(name,address, dob:1901 to till date,accountant_info(16 digits))are auto generated
- Admit: admit_id foriegn key of appointment
- Wallet_transactions:id ranging above 351 to 399
- Real_transactions: ranging from 301 to 350

- Prescribed_meds: duration is in hours (1 to 24)
- Medicine: id ranging from 201 and above till 299 company name and name are auto generated randomly available quantity is > 0.

Data load Script:-

InsertData.sql

```
delete from test_appointment;
delete from prescribed_meds;
delete from prescribed_tests;
delete from medicine;
delete from test;
delete from prescription;
delete from admit;
delete from real_transaction;
delete from   wallet_transaction;
delete from appointment;
delete from bed;
delete from doctor;
delete from admin;
delete from director;
delete from pharmacy_keeper;
delete from pathologist;
delete from accountant;
delete from nurse;
delete from staff;
delete from patient;
delete from lab;
delete from ward;
delete from department;
delete from slot_interval;
delete from slot;


\set localpath `pwd`'/data/slot.csv'
copy slot from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/slot_interval.csv'
copy slot_interval from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/department.csv'
copy department from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/ward.csv'
copy ward from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/lab.csv'
copy lab from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/patient.csv'
copy patient from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/staff.csv'
copy staff from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/nurse.csv'
copy nurse from :'localpath' delimiter ',' csv header NULL 'NULL';
```

```
\set localpath `pwd`'/data/accountant.csv'
copy accountant from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/pathologist.csv'
copy pathologist from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/pharmacy_keeper.csv'
copy pharmacy_keeper from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/director.csv'
copy director from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/admin.csv'
copy admin from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/doctor.csv'
copy doctor from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/bed.csv'
copy bed from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/appointment.csv'
copy appointment from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/wallet_transaction.csv'
copy wallet_transaction from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/real_transaction.csv'
copy real_transaction from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/admit.csv'
copy admit from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/prescription.csv'
copy prescription from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/test.csv'
copy test from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/medicine.csv'
copy medicine from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/prescribed_tests.csv'
copy prescribed_tests from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/prescribed_meds.csv'
copy prescribed_meds from :'localpath' delimiter ',' csv header NULL 'NULL';

\set localpath `pwd`'/data/test_appointment.csv'
copy test_appointment from :'localpath' delimiter ',' csv header NULL 'NULL';
```

Appendix:

DDL.sql :

```
DROP TRIGGER IF EXISTS check_delayed on appointment;
DROP TRIGGER IF EXISTS check_test_delayed on test_appointment;

DROP TABLE IF EXISTS    test_appointment;
DROP TABLE IF EXISTS    prescribed_meds;
DROP TABLE IF EXISTS    prescribed_tests;
DROP TABLE IF EXISTS    medicine;
DROP TABLE IF EXISTS    test;
DROP TABLE IF EXISTS    prescription;
DROP TABLE IF EXISTS    admit;
DROP TABLE IF EXISTS    real_transaction;
DROP TABLE IF EXISTS    wallet_transaction;
DROP TABLE IF EXISTS    appointment;
DROP TABLE IF EXISTS    bed;
DROP TABLE IF EXISTS    doctor;
DROP TABLE IF EXISTS    admin;
DROP TABLE IF EXISTS    director;
DROP TABLE IF EXISTS    pharmacy_keeper;
DROP TABLE IF EXISTS    pathologist;
DROP TABLE IF EXISTS    accountant;
DROP TABLE IF EXISTS    nurse;
DROP TABLE IF EXISTS    staff;
DROP TABLE IF EXISTS    patient;
DROP TABLE IF EXISTS    lab;
DROP TABLE IF EXISTS    ward;
DROP TABLE IF EXISTS    department;
DROP TABLE IF EXISTS    slot_interval;
DROP TABLE IF EXISTS    slot;
-- 1
CREATE TABLE slot (
       name                                    text,
       Primary key (name)
);

-- 2
CREATE TABLE slot_interval (
       name                          text,
       day                               integer check(day >=0 and day < 7),
       start_time                        time,
       end_time                          time   Not null,
       Primary key(name, day, start_time),
       Foreign key(name) references slot(name) on delete Cascade
);
-- 3
CREATE TABLE lab(
       id                                      integer,
       name                              text    Not null,
       address                           text    Not null,
       appoints_per_slot                 integer Not Null        check(appoints_per_slot
>= 0 ),
       slot_name                         text    Not null,
       Primary key(id),
       Foreign key (slot_name) references slot
);
```

```
-- 4
CREATE TABLE department (
       name                        text,
       Primary key (name)
);

-- 5
CREATE TABLE ward (
       dept_name               text,
       ward_num                    integer,
       type                        text        Not null        check (type in
('General', 'ICU')),
       charge_per_day          integer Not null        check(charge_per_day >= 0 ),
       Primary key (dept_name,ward_num),
       Foreign key(dept_name) references department(name)
);
-- 6
CREATE TABLE patient (
       id                              integer,
       name                        text,
       dob                         date        Not Null,
       phone                       text        Not Null        Unique,
       passwd_hash                             text    Not null,
              account_info             text,
       balance                     integer     Not null        check (balance >= 0 ),
       gender                  text    Not null        Check (gender
in('male','female','other')),
              address                      text,
              district                                text,
              state                                   text,
              country                                 text,
       height                      integer     check (height > 0 or height is null ) ,
       weight                      integer     check (weight > 0 or weight is null),


       Primary Key(id)
);

-- 7
CREATE TABLE staff (
       id                      integer,
       name                    text    NOT NULL,
       type                                    text    Not Null,
       gender                  text    Not null Check (gender
in('male','female','other')),
       date_of_joining                 date    NOT NULL,
       date_of_leave                   date,   /*(null allowed => currently working)*/
       dob                             date    Not null,
       salary                          integer Not null        check (salary >= 0),

       phone                           text    NOT NULL UNIQUE,
              passwd_hash                              text    Not null,
       address                         text,
       slot_name                       text,
       Primary Key(id),
       Foreign Key(slot_name) references slot(name)
);

-- 8
```

```
CREATE TABLE accountant (
       id                              integer,
       Primary key(id),
       Foreign Key(id) references staff(id) on delete Cascade
);

-- 9
CREATE TABLE nurse (
       id                              integer,
       dept_name                       text,
       ward_num                        integer,
       Primary key(id),
       Foreign Key(id) references staff(id) on delete Cascade,
       Foreign Key(dept_name, ward_num) references ward(dept_name,ward_num)
);

-- 10
CREATE TABLE pathologist (
       id                                  integer,
       lab_id                          integer,
       Primary key(id),
       Foreign Key(id) references staff on delete Cascade,
       Foreign Key(lab_id) references lab(id)
);

-- 11
CREATE TABLE pharmacy_keeper (
       id                                  integer,
       Primary key(id),
       Foreign Key(id) references staff(id) on delete Cascade
);
-- 12
CREATE TABLE director (
       id                                  integer,
       Primary key(id),
       Foreign Key(id) references staff(id) on delete Cascade
);
-- 13
CREATE TABLE admin (
       id                                  integer,
       Primary key(id),
       Foreign Key(id) references staff on delete Cascade
);
-- 14
CREATE TABLE  doctor(
       id                              integer,
       dept_name                       text,
       fee                             integer Not null        check (fee >= 0),
       room_no                  integer ,
       appoints_per_slot           integer Not null    check(appoints_per_slot >=0 ),
       Primary key(id),
       Foreign key(id) references staff on delete Cascade,
       Foreign key(dept_name) references department(name)
);


-- 15
CREATE TABLE bed (
       dept_name                           text,
       ward_num                    integer,
```

```
       bed_num                 integer,
       Primary key (dept_name,ward_num,bed_num),
       Foreign Key(dept_name, ward_num) references ward(dept_name, ward_num) on delete
Cascade
);


-- 16
CREATE TABLE appointment(
       id                                  integer,
       timestamp_                          timestamp without time zone         Default
current_timestamp,
       date_appoint                date        Not Null,
       status                      text        Not Null check(status in ('scheduled',
'complete', 'delayed', 'cancelled by doctor', 'cancelled')),
       doctor_id                   integer     Not null,
       patient_id                  integer     Not null,

       slot_name                   text        Not null,
       slot_day                    integer     Not null,
       start_time                      time without time zone  Not null,
       end_time                        time without time zone,

       Primary key(id),
       Foreign Key(doctor_id) references doctor,
       Foreign Key(patient_id) references patient,
       Foreign Key(slot_name, slot_day, start_time) references
slot_interval(name,day,start_time) on delete set Null
);


-- 17
CREATE TABLE wallet_transaction (
       id                                  integer,
       patient_id                  integer,
       amount                      integer    NOT NULL,
       service                     text               Not Null,
       timestamp_                  timestamp without time zone        Default
current_timestamp,
       Primary Key(id),
       Foreign Key(patient_id) references patient on delete set Null
);

-- 18
CREATE TABLE real_transaction (
       id                                  integer,
       accountant_id            integer,
       patient_id               integer,
       amount                      integer    Not Null,
       timestamp_                  timestamp without time zone        Default
current_timestamp,
       Primary key(id),
       Foreign Key(patient_id) references patient on delete set Null,
       Foreign Key(accountant_id) references accountant(id) on delete set Null
);
-- 19
CREATE TABLE admit (
       appointment_id          integer,
       dept_name               text,
       ward_num                integer,
```

```
	bed_num			integer,
	time_admit		timestamp without time zone		Default
current_timestamp,
	time_discharge		timestamp without time zone,
	total_bill		integer		check(total_bill>=0),
	Primary key(appointment_id),
	Foreign Key(appointment_id) references appointment on delete set NULL,
	Foreign Key(dept_name,ward_num, bed_num) references bed on delete set Null,
		Check (time_discharge is not null OR (dept_name is not null AND ward_num
is not null AND bed_num is not null))
		-- make sure that if patient is not discharged then bed should exist
);

-- 20
CREATE TABLE prescription(
	appointment_id		integer,
		diagnosis				text	Not null,
	remarks			text,
	Primary key(appointment_id),
	Foreign key(appointment_id) references appointment on delete Cascade
);


-- 21
CREATE TABLE test (
	test_id			integer,
	lab_id				integer		Not Null,
	test_name			text		Not null,
	test_fee			integer	Not null		check (test_fee >= 0 ),
		description				text,
	Primary key(test_id),
	Foreign Key(lab_id) references lab
);

-- 22
CREATE TABLE medicine (
	id				integer,
	name				text		Not null,
	price				  integer	Not null		check(price >=0 ),
	company			text		Not null,
	available_quantity		integer		Not null
check(available_quantity >= 0 ),
	Primary key(id)
);
-- 23
CREATE TABLE prescribed_tests (
	prescription_id		integer,
	test_id				integer,
	Primary key(prescription_id,test_id),
	Foreign Key(prescription_id) references prescription on delete Cascade,
	Foreign Key(test_id) references Test on delete set Null
);

-- 24
CREATE TABLE prescribed_meds (
	prescription_id		integer,
	med_id				integer,
	dose_per_day		integer		check(dose_per_day >= 0 OR dose_per_day
is null),
```

```
	duration                        integer         check(duration >= 0 OR duration
is null),
	Primary key(prescription_id,med_id),
	Foreign Key(prescription_id) references prescription on delete Cascade,
	Foreign Key(med_id) references medicine on delete set Null
);

-- 25
CREATE TABLE test_appointment (
	id                              integer,
	test_id                     integer             Not null,
	timestamp_                  timestamp without time zone Default
current_timestamp,
	pathologist_id          integer,
	patient_id              integer         Not null,

	slot_name                   text                Not null,
	slot_day                            integer         Not null,
	start_time                  time without time zone,
	end_time                    time without time zone,
	date_appoint                        date                Not null,
	status                      text        Not null check(status in ('scheduled',
'sample_taken', 'complete', 'delayed', 'cancelled', 'cancelled by pathologist')),
	result                  text,                   /*null while result not
published*/

	Primary key(id),
	Foreign Key(test_id) references Test,
	Foreign Key(patient_id) references patient,
	Foreign Key(pathologist_id) references pathologist,
	Foreign Key(slot_name,day,start_time) references
slot_interval(name,day,start_time) on delete set Null
);



-- ================= Triggers ====================
-- trigger to check for any delayed appointment
-- before any INSERT/UPDATE/DELETE
CREATE OR replace FUNCTION check_appoint_delayed()
  RETURNS TRIGGER
  LANGUAGE PLPGSQL
AS $$
BEGIN
  -- trigger logic
   update appointment set status = 'delayed'
   where status = 'scheduled'
   and (date_appoint < current_date) OR (date_appoint = current_date and end_time <
current_time);
   return new;
END;
$$;

create trigger check_delayed AFTER insert OR update OR delete
on appointment
for each statement
when (pg_trigger_depth() = 0)
execute procedure check_appoint_delayed();
```

```
-- trigger to check for any delayed test_appointment
-- before any INSERT/UPDATE/DELETE
CREATE OR replace FUNCTION check_test_appoint_delayed()
  RETURNS TRIGGER
  LANGUAGE PLPGSQL
AS $$
BEGIN
  -- trigger logic
   update test_appointment set status = 'delayed'
   where status = 'scheduled'
   and (date_appoint < current_date) OR (date_appoint = current_date and end_time <
current_time);
   return new;
END;
$$;

create trigger check_test_delayed AFTER insert OR update OR delete
on test_appointment
for each statement
when (pg_trigger_depth() = 0)
execute procedure check_test_appoint_delayed();



-- pay and confirm slot booking GIVEN(date, day, slot_name, start_time, end_time,
patient_id, fee)
-- creating a function to do booking
create or replace function book_appoint(max_appoints int, cur_appoints int,
       appoint_date date, start_time time, end_time time, p_id int, d_id int, slot_name
text, day int, fee int)
returns int
language plpgsql
as
$$
declare
  a integer;
begin
   select 0 into a;
   if (cur_appoints >= max_appoints) then
       raise exception 'slot is full';

   elseif ((select balance from patient where id = p_id) < fee) then
       raise exception 'insufficient balance in your wallet, recharge it';
   else
       insert into appointment(date_appoint, status, doctor_id, patient_id, slot_name,
slot_day, start_time, end_time)
       values (appoint_date, 'schedualed', d_id, p_id, slot_name, day, appoint_time,
end_time);

       update patient set balance = balance - fee where id = p_id;

       insert into wallet_transaction (patient_id, amount, service)
       values (p_id, -fee, 'appointment booking');

       select 1 into a;
   end if;
   return a;
end;
$$;
```