**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**

1. 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

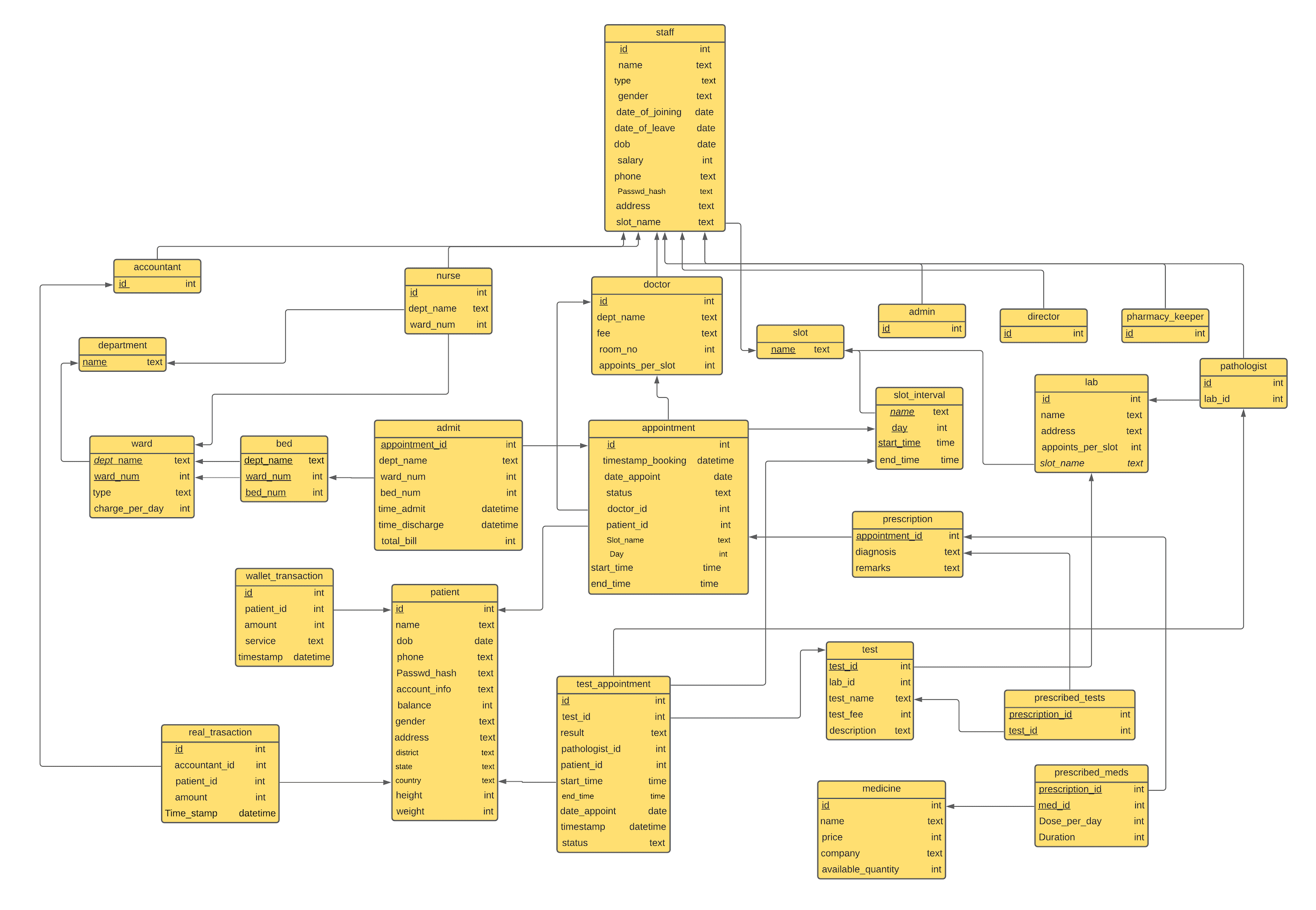
Test\_id->Lab\_id and Lab\_id->Slot\_name => Test\_id -> Slot\_name

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**

1. Accountant(

ID int

**Primary key(ID)**

**Foreign Key(ID) references Staff on delete set Cascade**

)

ID->R (ID is superkey)

In  **BCNF**

1. 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**

1. 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**

1. Pharmacy\_keeper(

ID int

**Primary key(ID)**

**Foreign Key(ID) references Staff on delete set Cascade**

)

ID->R

IN **BCNF**

1. Director(

ID int

**Primary key(ID)**

**Foreign Key(ID) references Staff on delete set Cascade**

)

ID->R

IN **BCNF**

1. Admin(

ID int

**Primary key(ID)**

**Foreign Key(ID) references Staff on delete set Cascade**

)

ID->R

IN **BCNF**

1. 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**

1. Department (

Name text

**Primary key (ID)**

)

Name -> R(Id is superkey)

In **BCNF**

1. 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**

1. 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**

1. Slot (

Name text

**Primary key (Name)**

)

name->name (trivial) in **BCNF**

1. 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

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

Fromtime 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**

1. 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**

1. 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**

1. 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**

1. 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**

1. 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**

1. 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**

1. 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**

1. 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**

1. 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**

1. 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

1. 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

Fromtime 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**

)

1. 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:-**
   1. Login into system as a staff (Doctor)
      1. Select \* from staff where phone=? and passwd\_hash=?
   2. Can see,cancel and mark complete each appointments

// to see all appointments

* + 1. Select \* from appointment where Doctor\_id = doctor\_id;

Select \* from appointment where Doctor\_id = doctor\_id and date =?;

// appointment related to a particular patient

* + 1. Select \* from appointment where Doctor\_id = doctor\_id and Patient\_id = patient\_id;

// cancelling appointment

* + 1. 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;

// marking appointment as complete

* + 1. Update appointment set status = ‘complete’

Where id = appoint\_id;

* 1. Write,modify and see prescription
     1. Insert into prescription values(?,?,?)

//adding various medicine for that prescription

* + 1. Insert into Prescribed\_meds(pres\_id,med\_id)

//adding various medicine for that prescription

* + 1. Insert into Prescribed\_tests(pres\_id,test\_id)
    2. UPDATE prescription SET diagnosis = changed\_diag, remarks = updated\_remarks;
    3. UPDATE Prescribed\_meds SET med\_id = new\_med\_id, Dose\_per\_day = new\_dose,

Durations = new\_duration;

* + 1. UPDATE Prescribed\_tests SET test\_id = new\_test\_id;

// see appointment details with prescription and meds and tests

* + 1. 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

* 1. Can See test reports

SELECT \* FROM test\_appointments where patient\_id = p\_id;

* 1. Get Patients info

SELECT \* FROM patients where patient\_id = p\_id;

1. **Accountant**
   1. Add money to patient’s wallet
      1. 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;

* 1. give back money from wallet
     1. 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;

1. **Pathologist**
   1. Access prescription and previous reports of patient

// to get prescription details

* + 1. SELECT \*

FROM appointment as A JOIN prescription as P ON (A.id = P.appointment\_id)

WHERE patient\_id = ?

// to get previous reports

* + 1. SELECT \*

From test\_appointmet where patient\_id = p\_id;

* 1. Take sample
     1. Update test\_appointment

Set status = “sample\_taken”

Where id = appoint\_id;

* 1. Add report to patient’s account
     1. Update test\_appointment

Set result = “path\_to\_file”, status = “complete”

Where id = appoint\_id;

* 1. Cancel appointment
     1. 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;

1. **Pharmacy\_keeper**
   1. Updating the medicine entity for the new supply
      1. UPDATE medicine SET available\_quantity = ?,

Name = ?, price = ?, company=?

WHERE id = ?

* + 1. INSERT INTO medicine values(?,?,?,?,?)
  1. Checking availability of the medicine before purchasing
     1. SELECT \* FROM medicine

WHERE id = ? // name = ? ;

* 1. Make payment and deduct money from patient’s wallet
     1. 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;

1. **Director**
   1. Add a new staff member
      1. INSERT INTO staff values(?,?,?,?,?,?,?,?,?,?,?);

//then insert the staff id into another table like doctor,admin etc

* + 1. INSERT INTO admin values(?);
    2. INSERT INTO pharmacy\_keeper values(?);
    3. INSERT INTO pathologist values(?);
    4. INSERT INTO doctor values(?,?,?,?,?);
    5. INSERT INTO nurse values(?,?,?);
    6. INSERT INTO accountant values(?);

* 1. Remove a staff member
     1. UPDATE staff SET relieving\_date = ? WHERE id = ?
  2. Can see statistics and analytics
     1. **//various analytics queries**

1. **Admin**
   1. Can see statistics and analytics
      1. **//various analytics queries**
2. **Patient**
   1. Login to dashboard
      1. Select \* from patient where phone = ? and passwd\_hash = ?;
   2. Get appointment and medical tests history
      1. Select \* from appointment a inner join doctor d on(a.doctor\_id = d.id)

where patient\_id = ?

Order by date\_appoint;

* + 1. 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;

* 1. 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**

* + 1. Select \* from staff s inner join doctor d using(id);

// get available slot\_intervals GIVEN (id)

* + 1. 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

* + 1. 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;

* 1. 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.
     1. Almost same queries as above for appointment with doctor with just change of relation name in the query
  2. Add money to the wallet from bank
     1. BEGIN TRANSACTION;

insert into real\_transaction(accountant\_id, patient\_id, amount) values (null, ?, ?);

update patient set balance = balance + ? where id = ?;

COMMIT TRANSACTION;

* 1. Withdraw money from the wallet
     1. 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 |

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

**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 |

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

**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 |

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

**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 |

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

**USER :-Patient**

**Use Case :-** *Withdraw money from the wallet*

Withdraw Money from Wallet

|  |
| --- |
| Amount |

|  |
| --- |
| Withdraw |

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

**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 |

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

**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

|  |  |  |
| --- | --- | --- |
| |  | | --- | | Enter Patient ID |  |  | | --- | | Go(Enter) | |

On Click:

|  |  |  |
| --- | --- | --- |
| Patient Personal Details:  --------------  --------------  --------------   |  | | --- | | Add report |  |  | | --- | | See Past Details | |

Use Case: Add report

|  |  |
| --- | --- |
| Report of the test:   |  | | --- | | Write here | |

**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;

$$;