Hospital management system

- All the tables are in 1NF as all have an atomic data, 2NF as all non-prime attributes fully dependent on the prime attribute and 3NF as there is no transitive dependency. While normalization up to 3NF is a standard practice, it's essential to strike a balance. Over-normalization can lead to complex query structures so that tables are normalized up to 3NF.
- **Tables:**

1. User table

user_id	role_id	first_name	last_name	contact_number	email
10	1	Aniket	shinde	8594933221	aniketshinde@gmail.com
NULL	NULL	NULL	NULL	NULL	NULL

2. User role table

role_id	role_name
1	Doctor
NULL	NULL

3. Insurance table

insurance_id	insurance_company	expiry_date
1000	LIC	2025-10-21
1001	LIC	2022-05-03

4. Patient table

patient_id	first_name	last_name	birth_date	gender	contact_number	email	insurance_id
100	Aditya	patil	2024-01-13	Male	9484736493	adityapatil@gmail.com	1000
101	Amit	pawar	2024-08-13	Male	9484768893	amitpawar@gmail.com	1001

5. Diagnosis table

diagnosis_id	patient_id	doctor_id	reason	diagnosis_date	diagnosis_amount
10000	100	10	kidney stone	2024-01-15	50000
10001	101	10	Fracture	2024-03-18	30000

6. Bill table

bill_id	patient_id	bill_date	bill_amount
1	100	2024-01-20	50000

- **Questions**

1. Write necessary queries to register new user roles and personas. Here user roles can be different like doctor, nurse, cashier even different kinds of doctors like dentist, surgeon etc.

   Query:

   insert into user_role values (1,'Doctor');

insert into users values (10,1, 'Aniket', 'shinde', '8594933221', 'aniketshinde@gmail.com');

Result:

user_id	role_id	first_name	last_name	contact_number	email
10	1	Aniket	shinde	8594933221	aniketshinde@gmail.com
NULL	NULL	NULL	NULL	NULL	NULL

role_id	role_name
1	Doctor
NULL	NULL

2. Write necessary queries to add to the list of diagnosis of the patient tagged by date.

   Query:

   insert into diagnosis values(10000,100,10,'kidney stone', '2024-1-15',50000);
   insert into diagnosis values(10001,101,10,'Fracture', '2024-3-18',30000);

   Result:

diagnosis_id	patient_id	doctor_id	reason	diagnosis_date	diagnosis_amount
10000	100	10	kidney stone	2024-01-15	50000
10001	101	10	Fracture	2024-03-18	30000

3. Write necessary queries to fetch required details of a particular patient. Here patient data from patient, diagnosis and insurance tables is shown together.

   Query:

   select p.*,d.doctor_id,d.reason,d.diagnosis_date, i.insurance_company, i.expiry_date from patient p join diagnosis d on p.patient_id = d.patient_id join insurance i on i.insurance_id = p.insurance_id where p.patient_id = 100;

   Result:

patient_id	first_name	last_name	birth_date	gender	contact_number	email	insurance_id	doctor_id	reason	diagnosis_date	insurance_company	expiry_date
100	Aditya	patil	2024-01-13	Male	9484736493	adityapatil@gmail.com	1000	10	kidney stone	2024-01-15	LIC	2025-10-21

4. Write necessary queries to prepare bill for the patient at the end of checkout. We can fill the bill data for same patient id, same patient can come to the hospital multiple times.

Query: insert into bill(patient_id,bill_date,bill_amount) values (100,'2024-1-20', 50000);

5. Write necessary queries to fetch and show data from various related tables (Joins). It gives data from patients name, doctors name, insurance and bill details;
   Query:
   select concat(p.first_name,p.last_name) as Patient_name,d.doctor_id,d.reason,d.diagnosis_date, i.insurance_company, i.expiry_date,b.bill_date,b.bill_amount from patient p join diagnosis d on p.patient_id = d.patient_id join insurance i on i.insurance_id = p.insurance_id join bill b on b.patient_id = p.patient_id where p.patient_id = 100;

   Result:

Patient_name	Doctors_name	reason	diagnosis_date	insurance_company	expiry_date	bill_date	bill_amount
Aditya patil	Aniket shinde	kidney stone	2024-01-15	LIC	2025-10-21	2024-01-20	50000

6. Optimize repeated read operations using views/materialized views. This view provides patients and diagnosis details.
   Query:
   create view patient_diagnosis as select concat(p.first_name," ",p.last_name) as 'Patients name', d.*, concat(u.first_name," ",u.last_name) as 'Doctors name' from diagnosis d join patient p on p.patient_id = d.patient_id join users u on u.user_id = d.doctor_id where p.patient_id = 100;
   #To view data in view
   select * from patient_diagnosis;

   Result:

Patients name	diagnosis_id	patient_id	doctor_id	reason	diagnosis_date	Doctors name
Aditya patil	10000	100	10	kidney stone	2024-01-15	Aniket shinde

7. Optimize read operations using indexing wherever required. (Create index on at least 1 table)
   Query: create index idx_patient_id on diagnosis(patient_id);

8. Try optimizing bill generation using stored procedures.

Query:

```
delimiter //
create procedure bill_generation(in patient_id int)
begin
 declare total_bill int;
 select d.diagnosis_amount into total_bill from diagnosis d where
d.patient_id = patient_id;
 insert into bill(patient_id,bill_date,bill_amount) values
(patient_id,current_date(),total_bill);
 End //
 delimiter ;
```

#To call procedure

```
Call bill_generation(100);
```

9. Add necessary triggers to indicate when patients medical insurance limit has expired.

Query:

```
delimiter //
create trigger check_insurance_expiry before insert on bill for each
row
begin
        declare exp_date date;
   select expiry_date into exp_date from insurance where
insurance.insurance_id = (select insurance_id from patient where
patient_id =
   NEW.patient_id);
   if NEW.bill_date > exp_date then
    SIGNAL SQLSTATE '45000'
  SET MESSAGE_TEXT = 'Patients insurance is expired bill not
possible!';
   end if;
   end; //
delimiter ;
```

Result: If insurance is not expired, new record is inserted into bill table else following message is shown when we call procedure or insert into bill table:

0 68 18:27:49	call bill_generation(101)	Error Code: 1644. Billing cannot be done for a patient with expired insurance.	0.000 sec