- The normal form which I have used is 3NF

  The reason is that, there should be not any non key attribute dependent on another non key attribute.

  e.g.

patientID	Diagnosis	equipment	Cost
01	Asthama	ECG	300
01	Blood test	Microscopes	500
02	Body Scan	X-Ray machine	899

  { patientId + Diagnosis } → primary key

  Here, equipment and cost are non key attribute and they both are fully dependent on primary key.

  But, cost is dependent on equipment and it is the non key attribute.

  { cost } → { equipment }

  i.e. one non key attribute is dependent on another non key attribute.

  So, to maintain the simplicity we define the new tables as below.

  Table1:-

patientID	Diagnosis	equipment
01	Asthama	ECG
01	Blood test	Microscopes
02	Body Scan	X-Ray machine
03	Body Scan	X-Ray macine

  Table2:-

equipment	Cost
ECG	300
Microscopes	500
X-Ray machine	899

Tables

1. Hospital table

Results | Messages

	name	city
1	Apple	Kolhapur
2	CPR	Kolhapur

2. Doctors table

Results | Messages

	doctor_id	name
1	d1	Kiran
2	d2	Omkar
3	d3	Suraj
4	d4	Alis
5	d5	Maya

3. Patients table

Results | Messages

	pid	fname	lname	insuranceExpired
1	p1	John	Smith	1
2	p2	Mary	Jones	1
3	p3	Rajesh	Kumar	0
4	p4	Lisa	Chen	0
5	p5	Love	Babbar	0

4. Insurance expiry table

Results | Messages

	pid	limitAmount	expiryDate
1	p1	5000	2024-01-13
2	p2	1000	2024-01-15
3	p3	8000	2024-02-25
4	p4	10000	2025-01-01
5	p5	15000	2024-06-13

5. PatientDisease table

Results | Messages | Client Statistic

	pid	disease
1	p1	Anxity
2	p2	Acne
3	p2	Asthama
4	p3	Bone Cancer

6. DoctorPatient Details

Results | Messages | Client Statistics

	doctor_id	pid
1	d1	p1
2	d1	p2
3	d2	p3

7. Nurses

Results | Messages | Client Statistics

	nurse_id	n_fname	n_lname
1	n1	Utkuuu	Ghatage
2	n2	Smitali	Erudkar
3	n3	Aishwarhya	Bacchan
4	n4	Lila	Henderson
5	n5	Aria	Mills
6	n6	Lina	Garcia
7	n7	Eva	Este

8. Equipments

Results | Messages | Client Stati

	eq_id	eq_name
1	e1	x-ray machine
2	e2	ECG
3	e3	ventilator
4	e4	scissors
5	e5	Microscopes

9. Diagnosis table

Results | Messages | Client Statistics

	patientId	diagnosis	DiagnosisDate	equipment
1	p1	Asthama	2024-01-30	ECG
2	p1	Blood test	2024-02-15	Microscopes
3	p2	Body Scan	2024-01-10	X-Ray machine
4	p3	Body Scan	2023-12-29	X-Ray machine

10. Diagnosis cost

Results | Messages | Client Statistics

	euquipment	cost
1	ECG	300
2	Microscopes	500
3	X-Ray machien	899

Q1. Write a necessary query to register new user roles and personas.

```
1. insert into doctor (doctor_id, name) values ('d1', 'Dr.Prakash');
2. insert into patient (pid, fname, lname) values ('p1', 'John', 'Smith');
3. insert into nurse(nurse_id, n_fname, n_lname) values('n1', 'Utkuuu', 'Ghatage');
```

Q2. Write necessary queries to add to the list of diagnosis of the patient tagged by date.

```
insert into diagnosis values
('p1', 'Asthama', '2024-01-30', 'ECG'),
('p1', 'Blood test', '2024-02-15', 'Microscopes'),
('p2', 'Body Scan', '2024-01-10', 'X-Ray machine'),
('p3', 'Body Scan', '2023-12-29', 'X-Ray machine');
```

Q3. Write necessary queries to fetch required details of a particular patient.

```sql
-- fetching details of name of patient using id
select pid, fname, lname from patient where pid = 'p1';
```

output:

	pid	fname	lname
1	p1	John	Smith

```sql
-- fetch diagnosis details of patient
select * from diagnosis where patientId='p1';
```

Results | Messages | Client Statistics

	patientId	diagnosis	DiagnosisDate	equipment
1	p1	Asthama	2024-01-30	ECG
2	p1	Blood test	2024-02-15	Microscopes

Q4. Write necessary queries to prepare bill for the patient at the end of checkout.

```sql
-- generating bill. We need to use join on three tables patient, diagnosis and cost.
-- (keep in mind that, give only that statements which can be grouped..)
SELECT
    p.pid AS PatientID,
    p.fname AS FirstName,
    p.lname AS LastName,

    sum(dc.cost) AS totalBill
FROM
    patient p
JOIN
    diagnosis d ON p.pid = d.patientId
JOIN
    diagnosisCost dc ON d.equipment = dc.euquipment
where
       p.pid = 'p1'
Group by
    p.pid, p.fname, p.lname;
```

Output:

Results | Messages | Client Statistics

	PatientID	FirstName	LastName	totalBill
1	p1	John	Smith	800

Q5. Write necessary queries to fetch and show data from various related tables (Joins).

```sql
-- Fetching data from multiple tables..
-- which doctor is giving treatment to which patient. we use doctor patient details table.
select
        doc.doctor_id,
        doc.name,
        p.pid,
        p.fname,
        p.lname
from
        doctor doc
join
        docPatientDetails  docDet on doc.doctor_id = docDet.doctor_id
join
        patient p on p.pid = docDet.pid;
```

Output:

Results | Messages | Client Statistics

	doctor_id	name	pid	fname	lname
1	d1	Dr.Kiran	p1	John	Smith
2	d1	Dr.Kiran	p2	Mary	Jones
3	d2	Dr.Omkar	p3	Rajesh	Kumar

Q6. Optimize repeated read operations using views/materialized views.

```sql
-- creating view to fech the detials of patients
create view fetchPatients as
        select
                p.pid,
                p.fname,
                p.lname,
                pd.disease
        from
                patient p
        join
                patientDisease pd on p.pid = pd.pid

-- this is how we execute the view
select * from fetchPatients;
```

Output:

Results | Messages | Client Statistics

	pid	fname	lname	disease
1	p1	John	Smith	Anxity
2	p2	Mary	Jones	Acne
3	p2	Mary	Jones	Asthama
4	p3	Rajesh	Kumar	Bone Cancer

Q7. Optimize read operations using indexing wherever required. (Create index on at least 1 table).

```sql
-- create index on nurse table
create index nurseIndex
on nurse (nurse_id);

select * from nurse where nurse_id = 'n3';
```

Messages | Execution plan | Client Statistics

Query 1: Query cost (relative to the batch): 100%
select * from nurse where nurse_id = 'n3'

SELECT
Cost: 0 %

Clustered Index Seek (Cluste…
[nurse].[PK__nurse__9BADE499…
Cost: 100 %

Q8. Try optimizing bill generation using stored procedures.

```sql
-- optimizing bill generation usign stored procedure
CREATE PROCEDURE generateBill
        @patient_id varchar(5)
AS
        SELECT
                p.pid AS PatientID,
                p.fname AS FirstName,
                p.lname AS LastName,
                sum(dc.cost) AS totalBill
        FROM
                patient p
        JOIN
                diagnosis d ON p.pid = d.patientId
        JOIN
                diagnosisCost dc ON d.equipment = dc.euquipment
        where
                p.pid = @patient_id
        Group by
                p.pid, p.fname, p.lname;
```

1. `exec generateBill @patient_id = 'p1'; -- this is how we execute the procedure..`

   output:

Results | Messages | Client Statistics

	PatientID	FirstName	LastName	totalBill
1	p1	John	Smith	800

2. `exec generateBill @patient_id = 'p2';`

   output:

Results | Messages | Client Statistics

	PatientID	FirstName	LastName	totalBill
1	p2	Mary	Jones	1399

Q9. Add necessary triggers to indicate when patients’ medical insurance limit has expired.

```sql
-- Create a trigger to update InsuranceExpired flag and log expiration
CREATE TRIGGER CheckInsuranceExpiry
ON InsuranceLimits
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    IF UPDATE(expiryDate)
    BEGIN
        DECLARE @PatientID varchar(5);

        SELECT @PatientID = pid FROM INSERTED;

        IF (SELECT ExpiryDate FROM INSERTED) < GETDATE()
        BEGIN
            -- Insurance limit has expired
            UPDATE patient
            SET insuranceExpired = 1
            WHERE pid = @PatientID;
        END
    END
END;
```

```sql
update InsuranceLimits
set expiryDate = '2024-01-13'
where pid = 'p1';

update InsuranceLimits
set expiryDate = '2024-01-15'
where pid = 'p2';

select * from InsuranceLimits;
select * from patient;
```

now, here I am updating the patient p1 and p2 expiray date, and as soon as I update it, it will automatically executes the trigger and chages are reflected into patient table.

Results | Messages

	pid	limitAmount	expiryDate
1	p1	5000	2024-01-13
2	p2	1000	2024-01-15
3	p3	8000	2024-02-25
4	p4	10000	2025-01-01
5	p5	15000	2024-06-13

	pid	fname	lname	insuranceExpired
1	p1	John	Smith	1
2	p2	Mary	Jones	1
3	p3	Rajesh	Kumar	0
4	p4	Lisa	Chen	0
5	p5	Love	Babbar	0