1

1.1) The large amount of data in the table slows down our query. At the same time, keeping all this big data in a single table is one of the reasons why the query performs poorly.

1.2) My first suggestion is to implement an archiving system. The current year's data can be kept in the main table, while the data from previous years can be archived and accessed only when needed for example, tables like, Islem_2020 , Islem_2021 etc.
Another suggestion is to store repetitive data like procedure codes or patient info in separate tables using IDs. Also, we can add indexes to the columns we frequently search by that would make queries shorter and faster.

1.3) It might work in the beginning when we assume there won't be too much data, but we need to be able to anticipate that the table will grow over time. With such a large amount of data, this current structure is not sustainable. It would be much more efficient if we could foresee this from the start and plan accordingly.

2

2.1 Using SELECT * to fetch the entire table is unnecessary. It's enough to select only the columns you actually need.
If you're looking for someone named Ahmet, writing Ahmet% or using a CONTAINS query might be sufficient.
Keep in mind that if you want to create indexes, functions like LOWER or YEAR don't work well with indexing.

2.2

```
SELECT AdSoyad, KayitTarihi FROM HastaKayit
WHERE AdSoyad LIKE 'ahmet%'
AND KayitTarihi >= '2024-01-01'
AND KayitTarihi < '2025-01-01';
```

Or, you can add indexes and do something like:

```
CREATE INDEX idx_AdSoyad ON HastaKayit(AdSoyad);
CREATE INDEX idx_KayitTarihi ON HastaKayit(KayitTarihi);

SELECT AdSoyad, KayitTarihi
FROM HastaKayit
WHERE AdSoyad LIKE 'ahmet%'
AND KayitTarihi >= '2024-01-01'
AND KayitTarihi < '2025-01-01';
```

2.3. We can convert the user input to lowercase (toLowerCase()) or uppercase (toUpperCase()) before sending it to SQL to ensure consistent searches.

3

3.1 I'm extracting only the year from the SatisTarihi column in the Satis table and displaying it in a column called Yil.I'm selecting the UrunAdi column from the Urun table.I'm summing the values from the Adet column in the Satis table and showing the result in a column named ToplamAdet.
By multiplying Satis.Adet with Urun.Fiyat, I calculate the total sales amount and display it in a column called ToplamTutar.I'm performing an INNER JOIN between the Urun and Satis tables using the UrunID field in other words, I'm joining the two tables on matching product IDs.
Finally, I group the data by year (YEAR(SatisTarihi)) and product name (UrunAdi) using a GROUP BY clause.

```
SELECT
YEAR(Satis.SatisTarihi) AS Yil,
Urun.UrunAdi,
SUM(Satis.Adet) AS ToplamAdet,
SUM(Satis.Adet * Urun.Fiyat) AS ToplamTutar
FROM Urun
INNER JOIN Satis ON Urun.UrunID = Satis.UrunID
GROUP BY
YEAR(Satis.SatisTarihi),
Urun.UrunAdi
```

3.2

```
SELECT *
FROM (
SELECT
YEAR(Satis.SatisTarihi) AS Yil,
Urun.UrunAdi,
SUM(Satis.Adet * Urun.Fiyat) AS ToplamTutar
FROM
Urun
JOIN
Satis ON Urun.UrunID = Satis.UrunID
GROUP BY
YEAR(Satis.SatisTarihi),
Urun.UrunAdi
) AS YillikUrunSatislari
WHERE ToplamTutar = (
SELECT MAX(SUM(Satis.Adet * Urun.Fiyat))
FROM
Urun
JOIN
Satis ON Urun.UrunID = Satis.UrunID
WHERE YEAR(Satis.SatisTarihi) = YillikUrunSatislari.Yil
GROUP BY
Urun.UrunAdi,
YEAR(Satis.SatisTarihi)
)
```

3.3 We use left join to combine tables based on matching IDs.
If there's no matching data for example, if a product has no sales the corresponding fields appear as NULL.In that case, we filter out the records where the product ID is NULL.

```
SELECT *
FROM Urun
LEFT JOIN Satis ON Urun.UrunID = Satis.UrunID
WHERE Satis.UrunID IS NULL
```