SQL QUERY

1. **Average patients per month, week, and year**
   Write a query to calculate the average number of patients admitted per month, week, and year for each hospital.

```
WITH weekly_admissions AS (
    SELECT
        p.hospital_id,
        h.hospital_name,
        COUNT(*) AS patients_admitted
    FROM
        Patient p
    JOIN
        Hospital h ON p.hospital_id = h.hospital_id
    GROUP BY
        p.hospital_id,
        DATE_FORMAT(p.admission_datetime, '%Y-%u')
),

monthly_admissions AS (
    SELECT
        p.hospital_id,
        h.hospital_name,
        COUNT(*) AS patients_admitted
    FROM
        Patient p
    JOIN
        Hospital h ON p.hospital_id = h.hospital_id
    GROUP BY
        p.hospital_id,
        DATE_FORMAT(p.admission_datetime, '%m')
),

yearly_admissions AS (
    SELECT
        p.hospital_id,
        h.hospital_name,
        COUNT(*) AS patients_admitted
    FROM
        Patient p
    JOIN
        Hospital h ON p.hospital_id = h.hospital_id
    GROUP BY
        p.hospital_id,
        DATE_FORMAT(p.admission_datetime, '%Y')
)

SELECT
    w.hospital_id,
    w.hospital_name,
    ROUND(AVG(w.patients_admitted), 0) AS avg_patients_per_week,
    ROUND(AVG(m.patients_admitted), 0) AS avg_patients_per_month,
    ROUND(AVG(y.patients_admitted), 0) AS avg_patients_per_year
```

```
FROM
   weekly_admissions w
JOIN
   monthly_admissions m ON w.hospital_id = m.hospital_id
JOIN
   yearly_admissions y ON w.hospital_id = y.hospital_id
GROUP BY
   w.hospital_id, w.hospital_name
ORDER BY
   w.hospital_id;
```

Weekly_admissions CTE counts the number of patients admitted to each hospital grouped by week.

DATE_FORMAT formats the admission date into a year-week format ('%Y-%u').
The result includes hospital_id, hospital_name, and the count of admitted patients (patients_admitted) which is obtained using Count () function.

Here Join is used to join patient table and hospital table with the help of a common column hospital_id which is in both tables.

Group By groups the table by hospital_id and date_format

Monthly_admissions CTE counts the number of patients admitted to each hospital grouped by month.

DATE_FORMAT in this query formats the admission date to group by month.
Similar to the previous CTE, it returns hospital_id, hospital_name, and patients_admitted which is obtained using Count () function.

Group By groups the table by hospital_id and date_format.

Yearly_admissions CTE counts the number of patients admitted to each hospital grouped by year.

In this query, same logic is applied but here the DATE_FORMAT formats the admission date to group by year.

At last SELECT statement selects data from the three CTEs.
It calculates the average number of patients admitted:

- Average weekly admissions.
- Average monthly admissions.
- Average yearly admissions.

The JOIN clauses combine data from the three CTEs based on hospital_id.Further,the results are grouped by hospital_id and hospital_name to ensure each hospital appears once in the results.The results are ordered by hospital_id.

hospital_id	hospital_name	avg_patients_per_week	avg_patients_per_month	avg_patients_per_year
1	Massachusetts General Hospital	187	1654	6616
2	ACMH Hospital	188	1663	6650
3	University Hopital	190	1675	6701
4	Boston City Hospital	191	1689	6757
5	Apollo Hospital	187	1652	6609

2. **Hospital occupancy on a daily, weekly, monthly, and yearly basis**
   Write a query to calculate the daily, weekly, monthly, and yearly hospital occupancy (admission/discharge) rates.

```
CREATE VIEW vw_occupancy_rate AS
SELECT
    h.hospital_id,
    h.hospital_name,
    DATE_FORMAT(p.admission_datetime, '%Y-%m-%d') AS period_date,
    'Daily' AS period_type,
    COUNT(*) AS admissions,
    COUNT(p.discharge_datetime) AS discharges,
    ROUND(COUNT(*) / NULLIF(COUNT(p.discharge_datetime), 0), 2) AS
occupancy_rate
FROM
    Hospital h
JOIN
    Patient p ON h.hospital_id = p.hospital_id
GROUP BY
    h.hospital_id, period_date

UNION ALL

SELECT
    h.hospital_id,
    h.hospital_name,
    DATE_FORMAT(p.admission_datetime, '%Y-%u') AS period_date,
    'Weekly' AS period_type,
    COUNT(*) AS admissions,
    COUNT(p.discharge_datetime) AS discharges,
    ROUND(COUNT(*) / NULLIF(COUNT(p.discharge_datetime), 0), 2) AS
occupancy_rate
FROM
    Hospital h
JOIN
    Patient p ON h.hospital_id = p.hospital_id
GROUP BY
    h.hospital_id, period_date

UNION ALL

SELECT
    h.hospital_id,
```

```
    h.hospital_name,
    DATE_FORMAT(p.admission_datetime, '%Y-%m') AS period_date,
    'Monthly' AS period_type,
    COUNT(*) AS admissions,
    COUNT(p.discharge_datetime) AS discharges,
    ROUND(COUNT(*) / NULLIF(COUNT(p.discharge_datetime), 0), 2) AS
occupancy_rate
FROM
    Hospital h
JOIN
    Patient p ON h.hospital_id = p.hospital_id
GROUP BY
    h.hospital_id, period_date

UNION ALL

SELECT
    h.hospital_id,
    h.hospital_name,
    DATE_FORMAT(p.admission_datetime, '%Y') AS period_date,
    'Yearly' AS period_type,
    COUNT(*) AS admissions,
    COUNT(p.discharge_datetime) AS discharges,
    ROUND(COUNT(*) / NULLIF(COUNT(p.discharge_datetime), 0), 2) AS
occupancy_rate
FROM
    Hospital h
JOIN
    Patient p ON h.hospital_id = p.hospital_id
GROUP BY
    h.hospital_id, period_date;

SELECT * FROM vw_occupancy_rate
ORDER BY hospital_id, period_date, period_type;
```

View vw_occupancy_rate is created to aggregate data from the Hospital and Patient tables.It calculates hospital statistics at four different time periods:

- Daily
- Weekly
- Monthly
- Yearly

The view uses the UNION ALL statement to combine four queries, each one representing one of the periods mentioned above.

Daily Data Calculation

DATE_FORMAT function formats the admission date (admission_datetime) into a daily format ('%Y-%m-%d').

Further COUNT () function is used to count the total number of admissions and discharges for each hospital, grouped by hospital_id and period_date.

Occupancy rate is calculated as the ratio of admissions to discharges. The NULLIF(COUNT(p.discharge_datetime), 0) ensures that a division by zero is avoided by returning NULL if the discharge count is zero.

NULLIF() function compares two expression and returns null if they are equal.Otherwise returns first expression.

Weekly Data Calculation

In this query, same logic is applied, but the date is formatted by week ('%Y-%u'), which gives a year-week number.

Monthly Data Calculation

In this query, same logic is applied, but the admission date is formatted by **month** ('%Y-%m')), which gives a year-month number.

Yearly Data Calculation

In this query, same logic is applied, but the admission date is formatted by **year** ('%Y'), which gives a year number.

At last we select all data from the vw_occupancy_rate view. The ORDER BY clause ensures that the data is ordered by hospital ID, period date, and period type. This allows for easy comparison of hospital performance across different time periods (daily, weekly, etc.)

hospital_id	hospital_name	period_date	period_type	admissions	discharges	occupancy_rate
1	Massachusetts General Hospital	2022	Yearly	2616	2105	1.24
1	Massachusetts General Hospital	2022-09	Monthly	178	143	1.24
1	Massachusetts General Hospital	2022-09-23	Daily	13	7	1.86
1	Massachusetts General Hospital	2022-09-24	Daily	23	16	1.44
1	Massachusetts General Hospital	2022-09-25	Daily	25	21	1.19
1	Massachusetts General Hospital	2022-09-26	Daily	30	24	1.25
1	Massachusetts General Hospital	2022-09-27	Daily	15	13	1.15
1	Massachusetts General Hospital	2022-09-28	Daily	26	21	1.24
1	Massachusetts General Hospital	2022-09-29	Daily	16	15	1.07
1	Massachusetts General Hospital	2022-09-30	Daily	30	26	1.15
1	Massachusetts General Hospital	2022-10	Monthly	808	642	1.26
1	Massachusetts General Hospital	2022-10-01	Daily	20	13	1.54
1	Massachusetts General Hospital	2022-10-02	Daily	28	18	1.56
1	Massachusetts General Hospital	2022-10-03	Daily	27	22	1.23
1	Massachusetts General Hospital	2022-10-04	Daily	33	25	1.32
1	Massachusetts General Hospital	2022-10-05	Daily	25	20	1.25
1	Massachusetts General Hospital	2022-10-06	Daily	16	13	1.23
1	Massachusetts General Hospital	2022-10-07	Daily	35	28	1.25
1	Massachusetts General Hospital	2022-10-08	Daily	18	17	1.06
1	Massachusetts General Hospital	2022-10-09	Daily	20	18	1.11
1	Massachusetts General Hospital	2022-10-10	Daily	28	26	1.08
1	Massachusetts General Hospital	2022-10-11	Daily	24	19	1.26
1	Massachusetts General Hospital	2022-10-12	Daily	23	18	1.28
1	Massachusetts General Hospital	2022-10-13	Daily	26	22	1.18

3. **Age-wise categorization of patients**
   Write a query to group and count patients based on age categories (e.g., Child, Adult, Senior).

```
SELECT
   CASE
      WHEN TIMESTAMPDIFF(YEAR, dob, CURDATE()) < 18 THEN 'Child'
      WHEN TIMESTAMPDIFF(YEAR, dob, CURDATE()) BETWEEN 18 AND 64 THEN
'Adult'
      ELSE 'Senior'
   END AS age_category,
   COUNT(*) AS patient_count
FROM
   Patient
GROUP BY
   age_category
ORDER BY
   age_category;
```

In this query **CASE** expression has been used to create a new column called age_category, which will assign patients into categories based on their age.

TIMESTAMPDIFF function is used to calculate the patient's age in years by subtracting the date of birth (dob) from the current date (CURDATE()).

In the case expression conditions are passed,

- If the calculated age is less than 18 years, the patient is categorized as a 'Child'.
- If the calculated age is between 18 and 64 years inclusive, the patient is categorized as an 'Adult'.
- **ELSE** : If none of the above conditions are true then they are categorized as a **'Senior'**.
-

The **END** marks the end of the CASE expression and assigns the result to the new column age_category.

Count (*) counts the number of patients in each age_category. This will count how many patients fall under each of the three age groups: Child, Adult, and Senior.
All this data is retrieved from the patient table.

GROUP BY groups the patients by their age categories. This ensures that the COUNT (*) will count the number of patients in each age group separately.

The **ORDER BY** clause sorts the results by the age category, typically in ascending alphabetical order, meaning the results will be displayed with 'Adult', 'Child', and 'Senior' in that order

age_category	patient_count
Adult	48898
Child	18686
Senior	32416

4. **Most consumed medicine**
Write a query to find out which medicine is consumed the most by all patients.

```
SELECT
   medicine_name,
   COUNT(*) AS consumption_count
FROM
   Treatment
GROUP BY
   medicine_name
ORDER BY
   consumption_count DESC
LIMIT 1;
```

Here this query selects two columns medicine_name and consumption count

Count (*) here counts how many times each medicine appears in the Treatment table, effectively measuring how often each medicine has been administered to patients. Count (*) function counts the total number of rows (records) for each medicine_name.

The **GROUP BY** clause groups all the rows that have the same medicine_name together.

Order By sorts the grouped results by the consumption_count column in descending order. The most consumed medicine (the one with the highest count) will appear at the top of the results.

DESC means descending order - from highest to lowest. If it was ASC, it would sort from the lowest to highest.

The **LIMIT** 1 clause ensures that only one result is returned—the medicine with the highest consumption count. This is because as it is mentioned we only need the most consumed medicine, not all of them.

	medicine_name	consumption_count
▶	Ibuprofen	14953

5. **Most consumed medicine by diagnosis**
Write a query to find the most consumed medicine for each diagnosis type.

```
WITH MedicineConsumption AS (
   SELECT
      d.diagnosis_name,
      t.medicine_name,
      COUNT(*) AS consumption_count
   FROM
      Diagnosis d
   JOIN
      Patient p ON d.patient_id = p.patient_id
   JOIN
      Treatment t ON p.patient_id = t.patient_id
```

```
    GROUP BY
        d.diagnosis_name, t.medicine_name
),
RankedMedicines AS (
    SELECT
        diagnosis_name,
        medicine_name,
        consumption_count,
        ROW_NUMBER() OVER (PARTITION BY diagnosis_name ORDER BY
consumption_count DESC) AS rnk
    FROM
        MedicineConsumption
)

SELECT
    diagnosis_name,
    medicine_name,
    consumption_count
FROM
    RankedMedicines
WHERE
    rnk = 1
ORDER BY
    diagnosis_name;
```

In this query, CTE named Medicine Consumption has been created that contains the count of how many times each medicine was consumed for each diagnosis, it is a temporary result set that you can reference within a SELECT, INSERT, UPDATE OR DELETE statement.

Here Diagnosis table is joined with patient table using a common column patien_id. Patient table is joined with treatment table using a common table between both i.e. patient_id. This creates a dataset that links diagnoses to the medicines prescribed to each patient.

COUNT function here counts how many times each medicine was administered for a given diagnosis.

Here group by groups the results by diagnosis_name and medicine_name , so that the COUNT(*) provides the consumption count for each diagnosis-medicine combination.

Another CTE named Ranked Medicines has been created that ranks the medicines by their consumption count for each diagnosis.

ROW_NUMBER () window function generates a unique row number for each row within a partition, based on the ordering of the consumption_count.

Partition By divides the data into groups based on diagnosis_name. So, for each diagnosis, the medicines will be ranked separately.

Order By: Within each diagnosis group, the medicines are ordered by their consumption count in descending order. This means the medicine with the highest consumption count will get a rank of 1.

Final Select query retrieves only the most consumed medicine for each diagnosis from Ranked Medicines **WHERE clause** here filters the Ranked Medicines table to return only the medicines with a rank of 1. **Order by** ordered the results by diagnosis_name.

diagnosis_name	medicine_name	consumption_count
Acid Reflux	Ibuprofen	875
Alzheimer Disease	Atorvastatin	878
Anxiety Disorder	Atorvastatin	846
Bipolar Disorder	Atorvastatin	861
Cancer	Ibuprofen	819
Celiac Disease	Ibuprofen	804
Chronic Bronchitis	Ibuprofen	848
Chronic Fatigue Syndrome	Atorvastatin	851
Chronic Obstructive Pulmonary Disease (COPD)	Ibuprofen	881
Cirrhosis	Ibuprofen	885
Common Cold	Atorvastatin	835
Coronary Artery Disease	Ibuprofen	857
Cough	Aspirin	1
Depression	Atorvastatin	852
Diabetes Type 2	Ibuprofen	819
Fibromyalgia	Atorvastatin	846
Flu	Ibuprofen	859
Gastroesophageal Reflux Disease	Ibuprofen	819
Heart Disease	Atorvastatin	837
HIV/AIDS	Ibuprofen	840
Hyperlipidemia	Ibuprofen	807
Hypertension	Ibuprofen	852
Kidney Stones	Ibuprofen	896
Lupus	Atorvastatin	815
Mild Asthma	Ibuprofen	840

6. **Average days of hospitalization**
   Write a query to calculate the average number of days a patient is hospitalized.

```
SELECT
   ROUND(AVG(DATEDIFF(p.discharge_datetime, p.admission_datetime)),0) AS
average_days_hospitalized
FROM
   Patient p
WHERE
   p.discharge_datetime IS NOT NULL;
```

DATEDIFF function calculates the difference between the discharge datetime and admission datetime in days for each patient. It returns the number of days each patient stayed in the hospital.

AVG () function calculates the average of all the calculated day differences. It represents the average number of days that patients were hospitalized.

ROUND (number, decimals): This rounds the average to the nearest integer, here in the query we have rounded it to 0 decimal places.

WHERE condition in the above query filters out the patients who have not yet been discharged. Here only discharged patients are considered for the calculation.

AS assigns the alias average days hospitalized to the result of the calculation, so the output column will have this name.

The query calculates the average duration of hospital stays (in days) for patients who have been discharged. It excludes patients who are still in the hospital.

	average_days_hospitalized
▶	183

7. Monthly and yearly income, with a cash/credit split

Write a query to display the total income (monthly and yearly), with a breakdown by payment mode (cash/credit).

```
WITH MonthlyIncome AS (
    SELECT
        DATE_FORMAT(p.discharge_datetime, '%Y-%m') AS period,
        b.payment_mode,
        SUM(b.bill_amount) AS total_income
    FROM
        Billing b
    JOIN
        Patient p ON b.patient_id = p.patient_id
    WHERE
        p.discharge_datetime IS NOT NULL
    GROUP BY
        period, b.payment_mode
),
YearlyIncome AS (
    SELECT
        DATE_FORMAT(p.discharge_datetime, '%Y') AS period,
        b.payment_mode,
        SUM(b.bill_amount) AS total_income
    FROM
        Billing b
    JOIN
        Patient p ON b.patient_id = p.patient_id
    WHERE
        p.discharge_datetime IS NOT NULL
    GROUP BY
        period, b.payment_mode
)

SELECT
    period,
    payment_mode,
    total_income,
    'Monthly' AS period_type
FROM
    MonthlyIncome

UNION ALL
```

```
SELECT
    period,
    payment_mode,
    total_income,
    'Yearly' AS period_type
FROM
    YearlyIncome

ORDER BY
    period_type, period, payment_mode;
```

In this we have used **Common Table Expressions (CTEs) -** Monthly Income and Yearly Income - to organize the results before combining them in a single query using UNION ALL

Monthly Income CTE calculates the yearly income.

DATE_FORMAT () here extracts the year and month e.g. 2023-09 from the discharge_datetime column, which will be used as the period for grouping.

The **SUM ()** here aggregates the total income (billed amount) for each month and for each payment mode (cash/credit).

Group By groups the rows by period (the year and month) and the payment_mode to give the total monthly income split by payment method.
Yearly Income CTE calculates the yearly income

DATE_FORMAT () here extracts only the year e.g., 2023 from the discharge_datetime column for grouping.

The **SUM ()** here calculates the total income for each year, broken down by payment mode (cash/credit).

Rows are grouped by period (the year) and payment_mode to compute the total yearly income per payment method.

Further we have combined monthly and yearly date with Union All

The first SELECT statement retrieves data from the Monthly Income CTE, including the period (year-month), payment mode, total income, and a label 'Monthly' to indicate that this is monthly data.

The second SELECT retrieves data from the Yearly Income CTE, but this time with a label 'Yearly' to indicate that the results are annual totals.

UNION ALL combines both sets of results into one result set, allowing you to see both monthly and yearly income totals in the same output. The final result is ordered by the period type, the period itself and the payment mode.

	period	payment_mode	total_income	period_type
▶	2022	cash	440541.28	Yearly
	2022	credit	436525.65	Yearly
	2023	cash	10900861.49	Yearly
	2023	credit	10695042.73	Yearly
	2024	cash	30648457.00	Yearly
	2024	credit	30939255.30	Yearly
	2022-09	cash	5084.87	Monthly
	2022-10	cash	81874.58	Monthly
	2022-10	credit	71636.69	Monthly
	2022-11	cash	120528.49	Monthly
	2022-11	credit	145588.48	Monthly
	2022-12	cash	233053.34	Monthly
	2022-12	credit	219300.48	Monthly
	2023-01	cash	266092.01	Monthly
	2023-01	credit	294245.65	Monthly
	2023-02	cash	356505.03	Monthly
	2023-02	credit	347443.04	Monthly
	2023-03	cash	489114.55	Monthly
	2023-03	credit	510596.26	Monthly
	2023-04	cash	573313.25	Monthly
	2023-04	credit	548791.39	Monthly
	2023-05	cash	746171.47	Monthly
	2023-05	credit	655417.54	Monthly
	2023-06	cash	780132.33	Monthly
	2023-06	credit	734920.43	Monthly