**1.How do you group data by month and year?**

You can use EXTRACT or DATE\_FORMAT to get the year and month, then GROUP BY those values:

SELECT

EXTRACT(YEAR FROM order\_date) AS year,

EXTRACT(MONTH FROM order\_date) AS month,

SUM(total\_revenue) AS monthly\_revenue

FROM orders

GROUP BY EXTRACT(YEAR FROM order\_date), EXTRACT(MONTH FROM order\_date)

ORDER BY year, month;

Or in MySQL:

SELECT

DATE\_FORMAT(order\_date, '%Y-%m') AS year\_month,

SUM(total\_revenue) AS monthly\_revenue

FROM orders

GROUP BY year\_month

ORDER BY year\_month;

**2. Difference between COUNT(\*) and COUNT(DISTINCT col)**

| **Function** | **Purpose** |
| --- | --- |
| COUNT(\*) | Counts **all rows**, including duplicates and NULLs. |
| COUNT(DISTINCT col) | Counts **unique non-NULL values** in a column. |

Example:

SELECT COUNT(\*) AS total\_orders,

COUNT(DISTINCT customer\_id) AS unique\_customers

FROM orders;

**3. How do you calculate monthly revenue?**

Use SUM() to aggregate revenue per month:

SELECT

EXTRACT(YEAR FROM order\_date) AS year,

EXTRACT(MONTH FROM order\_date) AS month,

SUM(total\_revenue) AS monthly\_revenue

FROM orders

GROUP BY year, month

ORDER BY year, month;

**4. What are aggregate functions in SQL?**

Aggregate functions summarize multiple rows into a single value:

* SUM(col) → total of a column
* AVG(col) → average
* COUNT(col) → number of rows
* MAX(col) → maximum value
* MIN(col) → minimum value

**5. How to handle NULLs in aggregates?**

* COUNT(col) **ignores NULLs**
* SUM(col) and AVG(col) **ignore NULLs**
* You can use COALESCE to replace NULLs:

SELECT SUM(COALESCE(total\_revenue, 0)) AS total\_revenue

FROM orders;

**6. Role of ORDER BY and GROUP BY**

* **GROUP BY** → Groups rows based on one or more columns to perform aggregates.
* **ORDER BY** → Sorts the final result by one or more columns (ascending/descending).

Example:

SELECT customer\_id, SUM(total\_revenue) AS revenue

FROM orders

GROUP BY customer\_id

ORDER BY revenue DESC;

**7. How do you get the top 3 months by sales?**

You can use ORDER BY with LIMIT (MySQL/PostgreSQL) or TOP (SQL Server):

-- MySQL/PostgreSQL

SELECT

DATE\_FORMAT(order\_date, '%Y-%m') AS year\_month,

SUM(total\_revenue) AS monthly\_revenue

FROM orders

GROUP BY year\_month

ORDER BY monthly\_revenue DESC

LIMIT 3;

-- SQL Server

SELECT TOP 3

FORMAT(order\_date, 'yyyy-MM') AS year\_month,

SUM(total\_revenue) AS monthly\_revenue

FROM orders

GROUP BY FORMAT(order\_date, 'yyyy-MM')

ORDER BY monthly\_revenue DESC;