**1. What is the difference between WHERE and HAVING?**

| **WHERE** | **HAVING** |
| --- | --- |
| Filters rows **before** grouping | Filters groups **after** grouping |
| Can’t use aggregate functions | Can use aggregate functions (e.g. SUM) |
| Used with SELECT, UPDATE, DELETE | Only used with SELECT + GROUP BY |

✅ Example:

SELECT department, COUNT(\*)

FROM employees

WHERE status = 'active'

GROUP BY department

HAVING COUNT(\*) > 5;

**2. What are the different types of joins?**

* **INNER JOIN**: Only matching rows in both tables.
* **LEFT JOIN (LEFT OUTER)**: All rows from the left table, and matched rows from the right.
* **RIGHT JOIN (RIGHT OUTER)**: All rows from the right table, and matched rows from the left.
* **FULL JOIN (FULL OUTER)**: All rows from both tables, matched where possible. *(Not natively supported in MySQL — use UNION of LEFT and RIGHT joins.)*
* **CROSS JOIN**: Cartesian product — every row of A with every row of B.
* **SELF JOIN**: Join a table with itself.

**3. How do you calculate average revenue per user (ARPU) in SQL?**

Assuming:

* users(user\_id)
* transactions(user\_id, amount)

SELECT

SUM(t.amount) / COUNT(DISTINCT u.user\_id) AS arpu

FROM

users u

LEFT JOIN

transactions t ON u.user\_id = t.user\_id;

**4. What are subqueries?**

A **subquery** is a query nested inside another SQL query.

Types:

* **Scalar subquery**: Returns a single value.
* **Row subquery**: Returns a single row.
* **Table subquery**: Returns multiple rows and columns.
* **Correlated subquery**: Depends on the outer query.

✅ Example:

SELECT name

FROM customers

WHERE id IN (

SELECT customer\_id FROM orders WHERE total > 500

);

**5. How do you optimize a SQL query?**

✅ Tips:

* Use **indexes** on columns used in WHERE, JOIN, and ORDER BY.
* Avoid SELECT \* — select only needed columns.
* Use **EXPLAIN** to analyze query performance.
* Avoid functions on indexed columns (e.g., WHERE YEAR(date)).
* Minimize subqueries and use **JOINs** where better.
* Use **LIMIT** with large datasets for pagination.
* Use **appropriate data types** and keep tables normalized.

**6. What is a view in SQL?**

A **view** is a virtual table created by a query. It doesn't store data but simplifies complex logic and improves readability.

✅ Example:

CREATE VIEW active\_users AS

SELECT \* FROM users WHERE status = 'active';

Then you can use:

SELECT \* FROM active\_users;

**7. How would you handle null values in SQL?**

* Use IS NULL or IS NOT NULL to check for nulls.
* Use COALESCE(column, default\_value) to replace nulls.
* Use IFNULL() (MySQL-specific) for default values.

✅ Example:

SELECT name, COALESCE(email, 'no\_email@example.com')