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

| **Feature** | **WHERE** | **HAVING** |
| --- | --- | --- |
| Purpose | Filters rows **before** aggregation | Filters groups **after** aggregation |
| Used with | Individual rows | Aggregate functions like SUM, AVG |
| Example | sql SELECT \* FROM sales WHERE region='East'; | sql SELECT region, SUM(sales) FROM sales GROUP BY region HAVING SUM(sales) > 1000; |

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

| **Join Type** | **Description** |
| --- | --- |
| **INNER JOIN** | Returns only matching rows from both tables. |
| **LEFT (OUTER) JOIN** | Returns all rows from the left table, with matching rows from the right (NULL if no match). |
| **RIGHT (OUTER) JOIN** | Returns all rows from the right table, with matching rows from the left (NULL if no match). |
| **FULL OUTER JOIN** | Returns all rows when there is a match in one of the tables. |
| **CROSS JOIN** | Returns Cartesian product of both tables (all combinations). |
| **SELF JOIN** | A table joined with itself. |
| **NATURAL JOIN** | Automatically joins tables based on columns with the same name. |

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

ARPU = Total Revenue ÷ Number of Users

SELECT SUM(revenue)/COUNT(DISTINCT user\_id) AS avg\_revenue\_per\_user

FROM sales;

* SUM(revenue) → total revenue
* COUNT(DISTINCT user\_id) → number of unique users

**4. What are subqueries?**

* A **subquery** is a query inside another query.
* Can be used in SELECT, FROM, or WHERE clauses.
* Types: **Scalar**, **Correlated**, **Non-correlated**.

**Example:**

-- Find employees with salary above average

SELECT name, salary

FROM employees

WHERE salary > (SELECT AVG(salary) FROM employees);

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

Tips for SQL optimization:

1. Use proper **indexes** on frequently searched columns.
2. Avoid SELECT \*; select only required columns.
3. Use **JOINs** efficiently, prefer INNER JOIN if possible.
4. Filter data early using WHERE instead of HAVING.
5. Avoid unnecessary subqueries; use **CTEs** (Common Table Expressions) or temp tables.
6. Use **EXPLAIN** to check the query execution plan.

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

* A **view** is a virtual table based on a SQL query.
* It does **not store data physically**, but provides a convenient way to query complex data.

**Example:**

CREATE VIEW high\_sales AS

SELECT customer\_id, SUM(amount) AS total\_sales

FROM sales

GROUP BY customer\_id

HAVING SUM(amount) > 1000;

* Now, SELECT \* FROM high\_sales; will give filtered results easily.

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

1. **Filter out NULLs**:

SELECT \* FROM employees WHERE manager\_id IS NOT NULL;

1. **Replace NULL with default** using COALESCE or IFNULL:

SELECT name, COALESCE(phone, 'Not Provided') AS phone\_number

FROM employees;

1. **Aggregate functions** usually ignore NULLs (SUM, AVG, etc.) automatically.