Interview Questions

1. What is the difference between WHERE and HAVING?

- **WHERE**: Filters rows *before* any grouping/aggregation. Used to filter individual records.
- **HAVING**: Filters groups *after* aggregation (like after `GROUP BY`). Used to filter summary results.

Example:

```
SELECT product_id, SUM(quantity)
FROM sales
WHERE sale_date >= '2025-08-01'  -- Filters rows first
GROUP BY product_id
HAVING SUM(quantity) > 20;       -- Filters aggregated results
```

2. What are the different types of joins?

- **INNER JOIN**: Only rows with matching keys in both tables.
- **LEFT JOIN (LEFT OUTER JOIN)**: All rows from the left table + matched rows from the right; fills with NULL if no match.
- **RIGHT JOIN (RIGHT OUTER JOIN)**: All rows from the right table + matched rows from the left.
- **FULL JOIN (FULL OUTER JOIN)**: All rows from both tables; NULLs where no match.
- **CROSS JOIN**: Cartesian product (every row of left × every row of right).

3. How do you calculate average revenue per user in SQL?

If you have a `users` table and a `sales` table with `user_id` and `amount`:

```
SELECT AVG(user_revenue) AS avg_revenue_per_user
FROM (
  SELECT user_id, SUM(amount) AS user_revenue
  FROM sales
  GROUP BY user_id
) sub;
```

- This finds the total revenue per user, then averages those totals.

4. What are subqueries?

- A **subquery** is a query *nested* inside another query.
- It can be in the `SELECT`, `FROM`, or `WHERE` clause, and is used to perform operations that depend on results from another query.

Example:

```
SELECT name
FROM users
WHERE id IN (SELECT user_id FROM sales WHERE amount > 1000);
```

5. How do you optimize a SQL query?

- Use proper indexes on columns used in `WHERE`, `JOIN`, `ORDER BY`.
- Select only needed columns (`SELECT col1, col2`), not `SELECT *`.
- Avoid unnecessary subqueries or complex joins.
- Use `EXPLAIN` to analyze query performance plan.
- Minimize use of functions in WHERE—use them in SELECT if possible.
- Partition large tables, or break queries into batches for huge datasets.

6. What is a view in SQL?

- A **view** is a *virtual table* based on the result of an SQL statement.
- It shows data from one or more tables, making complex queries reusable.
- You can query a view like a table:

```
CREATE VIEW high_value_sales AS
 SELECT * FROM sales WHERE amount > 1000;
```

- Views don’t store data—just the query logic.

7. How would you handle null values in SQL?

- Use the `IS NULL` or `IS NOT NULL` conditions to filter for null/non-null.
- Use `COALESCE(a, b)` or `IFNULL(a, b)` to substitute a default value when a field is null:

```
SELECT name, COALESCE(email, 'no email provided') FROM users;
```

- In aggregations, functions skip nulls, but always check logic (e.g., `COUNT(field)` ignores nulls).