INTERVIEW PREPARATION QUESTIONS

1 Difference between WHERE and HAVING

- **WHERE**: Filters rows **before** aggregation. Used with individual records.
- **HAVING**: Filters **after** aggregation. Used with GROUP BY and aggregate functions.

Example:

```
SELECT * FROM sales
WHERE amount > 100;

-- HAVING filters after grouping
SELECT customer_id, SUM(amount) AS total_sales
FROM sales
GROUP BY customer_id
HAVING total_sales > 1000;
```

2 Different types of joins

- **INNER JOIN**: Returns rows that match in both tables.
- **LEFT JOIN**: All rows from the left table + matching rows from right table.
- **RIGHT JOIN**: All rows from the right table + matching rows from left table.
- **FULL OUTER JOIN**: All rows from both tables (matches + non-matches).
- **CROSS JOIN**: Cartesian product of both tables.
- **SELF JOIN**: A table joined with itself.

3 Calculate Average Revenue Per User (ARPU) in SQL

```
SELECT
   SUM(revenue) / COUNT(DISTINCT user_id) AS avg_revenue_per_user
FROM transactions;
```

- **SUM(revenue)** → total revenue
- **COUNT(DISTINCT user_id)** → total unique users

4 What are subqueries?

- A **subquery** is a query **inside another query**.
- Used to return data to the main query for filtering, aggregation, or joining.

Example:

```
SELECT *
FROM products
WHERE price > (SELECT AVG(price) FROM products);
```

5 How to optimize a SQL query

- Use **indexes** on frequently filtered/joined columns.
- Avoid SELECT * — select only required columns.
- Use **JOINs** instead of subqueries where possible.
- Use proper **data types**.
- Filter early using WHERE clauses.
- Avoid unnecessary sorting (ORDER BY).
- Analyze query execution plan for bottlenecks.

6 What is a view in SQL?

- A **view** is a **saved SQL query** that acts like a virtual table.
- It doesn't store data physically (unless it's a materialized view).
- Useful for simplifying complex queries and enhancing security.

Example:

```
CREATE VIEW TopCustomers AS
SELECT customer_id, SUM(amount) AS total_spent
FROM sales
GROUP BY customer_id
HAVING total_spent > 5000;
```

7️⃣ **How to handle NULL values in SQL**

- **IS NULL / IS NOT NULL** for filtering:

SELECT * FROM customers WHERE phone IS NULL;

- **COALESCE()** → Replace NULL with default value:

SELECT COALESCE(phone, 'Not Provided') FROM customers;

- **IFNULL()** or **CASE WHEN** for conditional handling.
- Avoid NULL in aggregates by using functions that skip NULLs (e.g., COUNT(column) ignores NULLs).