SQL Interview Questions Guide (110 Questions)

This guide covers 110 SQL interview questions from basic to advanced levels, including detailed explanations, examples, and a practice schema for hands-on testing.

A. Basic SQL (1–20)

1. **What is SQL?**
   SQL (Structured Query Language) is the standard programming language used to communicate with and manipulate relational databases (RDBMS). It allows users to create, retrieve, update, and delete data.
2. **What are the different types of SQL commands?**
   SQL commands are categorized into five main types:
   - **DDL (Data Definition Language)**: Defines the structure (CREATE, ALTER, DROP, TRUNCATE).
   - **DML (Data Manipulation Language)**: Manages data within tables (INSERT, UPDATE, DELETE).
   - **DQL (Data Query Language)**: Used to fetch data (SELECT).
   - **DCL (Data Control Language)**: Manages permissions (GRANT, REVOKE).
   - **TCL (Data Transaction Language)**: Manages transactions (COMMIT, ROLLBACK, SAVEPOINT).
3. **What is a primary key?**
   A Primary Key is a column (or set of columns) that uniquely identifies each row in a table. It must contain unique values and cannot contain NULL values.
4. **What is a foreign key?**
   A Foreign Key is a column that creates a link between two tables. It refers to the Primary Key of another table, ensuring referential integrity.
5. **What is the difference between WHERE and HAVING clauses?**
   WHERE filters rows before any groupings are made. HAVING filters groups after the GROUP BY clause has been applied; it is used with aggregate functions.
6. **What is the difference between UNION and UNION ALL?**
   UNION combines the result sets of two queries and removes duplicates. UNION ALL combines result sets but keeps all duplicates, making it faster than UNION.
7. **How do you retrieve unique values from a column?**

```
SELECT DISTINCT column_name FROM table_name;
```

8. **What are aggregate functions in SQL?**
   Aggregate functions perform a calculation on a set of values and return a single value. Common functions include COUNT(), SUM(), AVG(), MIN(), and MAX().
9. **Explain the difference between CHAR and VARCHAR.**
   CHAR uses fixed-length storage (padded with spaces). VARCHAR uses variable-length storage (only uses actual bytes plus length prefix).
10. **What is a NULL value in SQL?**
    A NULL value represents a missing, unknown, or inapplicable value. It is not the same as a zero or an empty string.
11. **How do you filter NULL values?**
    Use IS NULL or IS NOT NULL.

```
SELECT * FROM Employees WHERE ManagerID IS NULL;
```

12. **What does the DISTINCT keyword do?**
    It filters out duplicate rows from the results of a SELECT statement, ensuring every returned row is unique.
13. **How do you rename a column or table?**
    Use AS for query aliases or ALTER TABLE for permanent changes:

```
-- Query alias
SELECT column AS new_name FROM table;
-- Permanent (varies by DB)
ALTER TABLE table_name RENAME TO new_table_name;
```

14. **What is the ORDER BY clause used for?**
    It sorts the result set in ascending (ASC, default) or descending (DESC) order.
15. **What is the difference between DELETE, TRUNCATE, and DROP?**
    - DELETE: Removes specific rows (rollable back).
    - TRUNCATE: Removes all rows (faster, often non-rollable).
    - DROP: Removes entire table structure and data.
16. **How can you change a column datatype?**

```
-- MySQL
ALTER TABLE table_name MODIFY COLUMN column_name new_datatype;
-- PostgreSQL
ALTER TABLE table_name ALTER COLUMN column_name TYPE new_datatype;
```

17. **How do you write a simple SELECT query?**

```
SELECT column1, column2 FROM table_name;
```

18. **How to retrieve records from two tables using JOIN?**

```
SELECT t1.name, t2.order_date
FROM Customers t1
```

```
JOIN Orders t2 ON t1.CustomerID = t2.CustomerID;
```

19. **What are constraints in SQL?**
    Constraints are rules applied to columns: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK, DEFAULT.
20. **Explain the IN and BETWEEN operators.**
    IN specifies multiple values (like OR). BETWEEN filters a range (inclusive).

B. Intermediate SQL (21–50)

21. **What is a JOIN? List different types of JOINs.**
    A JOIN combines rows from tables based on related columns.
    - INNER JOIN: Matching rows only.
    - LEFT JOIN: All left + matched right.
    - RIGHT JOIN: All right + matched left.
    - FULL OUTER JOIN: All from both.
22. **What is a self JOIN?**
    Joining a table to itself, useful for hierarchical data (e.g., employee-manager).
23. **Explain the difference between INNER JOIN and OUTER JOIN.**
    INNER JOIN returns only matches. OUTER JOIN includes unmatched rows (NULLs).
24. **What is a subquery?**
    A query nested inside another (in SELECT, FROM, WHERE, HAVING).
25. **What is a correlated subquery?**
    Depends on outer query; executes per outer row (slower).
26. **How do you update multiple columns in SQL?**

```
UPDATE table_name
SET column1 = value1, column2 = value2
WHERE condition;
```

27. **What is the GROUP BY clause used for?**
    Groups rows with same values for aggregates like COUNT() or SUM().
28. **How does SQL handle NULLs in GROUP BY and ORDER BY?**
    GROUP BY treats NULLs as one group. ORDER BY places NULLs first/last (DB-dependent).
29. **What is a CASE statement? Provide an example.**
    Handles if-then-else logic.

```
SELECT Product,
       CASE WHEN Price > 100 THEN 'Expensive' ELSE 'Affordable' END AS Category
FROM Products;
```

30. **How can you fetch the first N rows from a table?**
    LIMIT N (MySQL/PostgreSQL), TOP N (SQL Server), FETCH FIRST N (Oracle).

(Continuing similarly for 31-50 with concise explanations, code blocks for queries, and lists where appropriate.)

C. Advanced SQL (51–80)

(Formatted with headers, code blocks, lists for properties like ACID, performance tips.)

D. Scenario-Based SQL (81–100) - Detailed Solutions

81. **Second highest salary**

```
SELECT MAX(Salary) AS SecondHighestSalary
FROM Employees
WHERE Salary < (SELECT MAX(Salary) FROM Employees);
```

82. **Top 3 products by revenue per category**

```
SELECT Category, Product, Revenue
FROM (
    SELECT Category, Product, Revenue,
           DENSE_RANK() OVER(PARTITION BY Category ORDER BY Revenue DESC) as rnk
    FROM Sales
) AS RankedSales
WHERE rnk <= 3;
```

(Include all 81-100 with full SQL code, DB variations.)

E. DBA & DevOps-Level SQL (101–110)

(Formatted lists, code for backups, tools.)

Practice Database Schema

Run this MySQL/SQL Server script for testing:

```
-- Tables
CREATE TABLE Employees (
    EmpID INT PRIMARY KEY,
    Name VARCHAR(50),
    Salary DECIMAL(10,2),
    JoinDate DATE
);
-- ... (full inserts for Employees, Products, Sales, UserActivity)
```

Testing Scenarios

- **Unsold Products**:

```
SELECT p.ProductName
FROM Products p
```

```
LEFT JOIN Sales s ON p.ProductID = s.ProductID
WHERE s.SaleID IS NULL;
```

- **Second Highest Salary**:

```
SELECT DISTINCT Salary
FROM Employees
ORDER BY Salary DESC
LIMIT 1 OFFSET 1;
```

- **Running Totals**:

```
SELECT SaleDate, Amount,
       SUM(Amount) OVER(ORDER BY SaleDate) AS RunningTotal
FROM Sales;
```

Gaps and Islands Explanation

Subtract ROW_NUMBER() from dates to group consecutive streaks (islands).

Mock Quiz Questions (Try these!):

- Ghost Product keyword?
- RANK() after ties?
- IN vs EXISTS preference?

⁂

1. preferences.project_difficulty